Gesten- und Sprachsteuerung für einen mobilen Roboter mittels Kinect for Windows unter Java

Studienarbeit

für die Prüfung zum

Bachelor of Science

der Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Simon Ebner, Volker Werling

Mai 2013

Bearbeitungszeitraum	12 Wochen
Matrikelnummer, Kurs	5837963 7012192, TAI10B2
Gutachter	Prof. Hans-Jörg Haubner

Erklärung

Wir erklären hiermit ehrenwörtlich:

1. dass wir unsere Studienarbeit mit dem Thema *Gesten- und Sprachsteuerung fuer einen mobilen Roboter mittels Kinect for Windows unter Java* ohne fremde Hilfe angefertigt haben;
2. dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet haben;
3. dass wir unsere Studienarbeit bei keiner anderen Prüfung vorgelegt haben;
4. dass die eingereichte elektronische Fassung exakt mit der eingereichten schriftlichen Fassung übereinstimmt.

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Karlsruhe, Mai 2013

Simon Ebner, Volker Werling

Zusammenfassung

Durch den Fortschritt der Technik wird der Wunsch des Menschen sich bestimmte Arbeiten zu erleichtern immer größer. Automaten, sogennante Roboter, werden eingesetzt um Menschen ihnen unliebsame, schwere oder auch gefährliche Aufgaben abzunehmen. Da Roboter mehr und mehr Eingang in den Alltag, durch immer weitere Anwendungsfelder, wie beispielsweise in der Pflege, finden, wird es wichtig sich mit der Interaktion von Mensch und Maschine zu befassen.
In der ersten Arbeit wurde die Umsetzung der Schnittstelle zwischen Mensch und Maschine behandelt.
Diese Arbeit beschäftigt sich mit der weiteren Entwicklung der Anwendung und im speziellen mit der Umsetzung der Schnittstelle zu einem mobilen Roboter, der durch das Gesten- und Sprachinterface ferngesteuert werden soll.

Die Autoren Ebner und Werling haben diese Arbeit gemeinsam verfasst, wobei die Kapitel jeweils von einem der Autoren geschreiben wurden. Auf Herrn Ebner entfallen Kapitel 2, 4, 7 und 8 und auf Herrn Werling die Kapitel 1, 3, 5, 6, und 9.

Inhaltsverzeichnis

1 Einleitung

Seit jeher entwickeln wir Menschen Maschinen zur leichteren und effizienteren Verrichtung von Arbeiten. Um die Verrichtung von Arbeiten von Menschen effizient zu übernehmen ist hier die denkbarste Form für einen Roboter eben auch die eines Menschen. Durch die Jahrhunderte haben Ingenieure und Wissenschaftler verschiedenste solcher Maschinen entwickelt. Die Entwicklung der ersten Automaten fand bereits in der Antike statt. Beispiele hierfüer sind die Wasserorgel von Heron von Alexandria, oder die fliegende Taube von Archytas von Tarent. Der arabische Erfinder al-Jazari, entwickelt zu Ende des zwölften Jahrhunderts verschiedene Automaten, wie zum Beispiel ein Teeservierender Roboter. Im Mittelalter, im Jahr 1495, entwarf Leonardo Da Vinci einen Mechanischen Ritter, als humanoiden Androiden. Die Schweizer Brüder Droz und Droz bauen im Jahr 1774 drei humanoide Roboter, einen Zeichner, einen Schreiber und eine Klavierspielerin.

Der Begriff Roboter wurde im Jahr 1920 vom tschechischen Schriftsteller Capek das erste mal in einem Schauspiel, Rossums Universal Robots[1], verwendet. Capek beschreibt hiermit Maschinenmenschen, welche Fronarbeit verrichten. Abgeleitet ist der Begriff vom tschechischen Wort für Fronarbeit, „rabota “.
Dieses Motiv der Menschmaschine, wurde von Science Fiction Autoren aufgegriffen. Der berühmte Autor Isaac Asimov erwähnte die Robotik als Studium der Roboter erstmals in seiner Kurzgeschichte Runaround[2], welche 1942 veröffentlicht wurde. Ebenso erfand Asimov in dieser Geschichte die 3 Robotergesetze:

- Ein Roboter darf keinen Menschen verletzen oder durch Untätigkeit zu Schaden kommen lassen.
- Ein Roboter muss den Befehlen eines Menschen gehorchen, es sei denn, solche Befehle stehen im Widerspruch zum ersten Gesetz.
- Ein Roboter muss seine eigene Existenz schützen, solange dieser Schutz nicht dem Ersten oder Zweiten Gesetz widerspricht.

Heute werden Roboter vor allem zur industriellen Fertigung eingesetzt. Sie ersetzen Menschen an Fließbändern, sowie weiteren Bereichen, wo die Arbeit durch einen Roboter effizienter und für den Menschen ungefährlicher durchgeführt werden kann. Die Logik dieser Roboter ist in der Regel fest einprogrammiert. Durch den immer breiteren Einsatz von Robotern, auch in zivilen Bereichen, wie beispielsweise der Pflege, wird es zunehmend wichtig auch hier neue Bedienkonzepte einzuführen. Da Roboter in diesen Bereichen zunehmend auch immer menschlicher dargestellt werden ist es hier wichtig auch dem Menschen natürliche Bedienschnittstellen zu Verfügung zu stellen. Die dem Menschen natürlichsten Arten der Kommunikation sind Sprache und Gestik. Dieser Teil wurde von den Authoren bereits in einer ersten Arbeit behandelt [EW13]. Diese Arbeit wird sich der Kombination der entwickelten Gesten- und Sprachbasierten Benutzerschnittstelle mit der Steuerung eines Roboters widmen.

[1]Karel Capek. Rossum Universal Robots von Capek, frei verfgbar bei der Universität Adeleide. ebooks.adelaide.edu.au. Abgerufen Mai 28, 2013

[2]Isaac Asimov. Runaround von Asimov bei der RCI Rutgers. rci.rutgers.edu. Abgerufen Mai 28, 2013

2 Aufgabenstellung

2.1 Problemstellung und Ziel der Arbeit

Aus dem ersten Teil der Arbeit ging die Entwicklung einer Schnittstelle zwischen Mensch und Roboter hervor. Dabei wurden ein erstes Konzept für die verwendeten Gesten und Sprachbefehle, sowie eine erste Umsetzung Stufe der Softwarearchitektur der Anwendung vorgestellt. Ziel des zweiten Teils ist, mit Hilfe diesen ersten Ergebnissen eine Steuerung eines mobilen Roboters umzusetzen. Dabei werden weiterhin die Bibliotheken *jnect* zur Steuerung der Kinect mittels Java, und *jahmm* zur Nutzung der Hidden Markov Models (HMMs) verwendet. Bislang offen sind die Punkte des zu verwendenden mobilen Roboters und die konkrete Implementierung der Anwendung mit den dafür benötigten Funktionalitäten.
Daraus ergeben sich nun folgende Ziele:

- Analyse verfügbarer Ausführungen von mobilen Robotern
- Auswahl eines Robotermodells anhand gegebener Kriterieen und Voraussetzungen
- Konzeption der Anwendung zur Steuerung des mobilen Roboters
- Implementierung der Anwendung
- Optimierung der Gestenerkennung (optional)
- Benutzerfreundlichkeit (Usability) der Anwendung

Dabei wird von der Problemstellung ausgegangen, dass im Rahmen dieser Studienarbeit die Wahl eines mobilen Roboters nur über vorhandene Modelle von mobilen Robotern möglich ist, die auch an der DHBW Karlsruhe verfügbar sind. Bei dieser Wahl sind auch die vorhandenen Restriktionen aus der bereits gewählten Programmiersprache, Java, zu beachten, da vermieden werden soll, mehrere Sprachen in der Entwicklung zu verwenden.
Das Kernziel, die korrekte Erkennung von Gesten und Sprache hängt von zahlreichen Faktoren ab, und benötigt ein hohes Maß an Testaufwand. Auf das Problem der Feineinstellung dieser Systeme durch empirische Ergebnisse wird eingegangen, ob das Ziel einer optimalen Lösung auch erreicht werden kann, ist ungewiss.

2.2 Geplantes Vorgehen

Aus dem ersten Teil der Studienarbeit ging unter anderem eine modular aufgebaute Rich client platform (RCP) Anwendung hervor. Da die einzelnen Module dieser Anwendung in Form von Eclipse Plugins separat verwendet werden können, kann die Arbeit an den unter 2.1 genannten Zielen, zu einem Großteil unabhängig voneinander in die Anwendung integriert werden.
Zu Beginn der Arbeit muss jedoch die Analyse und die Auswahl eines mobilen Roboters abgeschlossen werden, da eventuell auf Besonderheiten eines speziellen Robotermodells in der Anwendung gesonder eingegangen werden muss.
Nachdem also ein Modell ausgewählt wurde und die diversen Arbeiten an der Anwendung separat durchgeführt wurden, soll durch Tests und Optimierungen die Anwendung benutzerfreundlicher gestaltet werden, um eine höhere Qualität der Mensch-Computer-Interaktion zu erreichen.

2.3 Ausblick

Software die nicht weiterentwickelt wird, deren Fehler nicht behoben werden, die nicht auf Wünsche der User angepasst werden, und deren Funktionen nicht erweitert werden, werden auf kurz oder lang nicht mehr verwendet. Dies soll hier versucht werden, zu verhindern. Bereits während der Arbeiten an der Studienarbeit, steht der vorhandene Code der Anwendung unter einer kollaborativen Versionsverwaltung, wobei die Arbeit natürlich nur von den beiden Autoren durchgeführt wird. Um das Ziel dieser Studienarbeit, eine benutzerfreundliche Steuerung eines mobilen Roboters zu erstellen, über den Zeitraum der Studienarbeit weiter zu tragen, soll der Code der Anwendung im Anschluss der Studienarbeit frei verfügbar sein.

3 Stand der Technik

3.1 Roboter

Was genau ein Roboter ist, ist nicht einheitlich definiert. Nachfolgend werden einige der wichtigsten Definitionen genannt.

Die Japan Robot Assosication (JARA) unterteilt Roboter in die folgenden Kategorien[3] :

- Manual Manipulator: Handhabungsgerät, das kein Programm hat, sondern direkt vom Bediener geführt wird,
- Fixed Sequence Robot: Handhabungsgerät, das wiederholt nach einem konstanten Bewegungsmuster arbeitet. Das Ändern des Bewegungsmusters ist relativ aufwendig,
- Variable Sequence Robot: Handhabungsgerät, wie vorher beschrieben, jedoch mit der Möglichkeit, den Bewegungsablauf schnell und problemlos zu ändern,
- Playback Robot: Der Bewegungsablauf wird diesem Gerät einmal durch den Bediener vorgeführt und dabei im Programmspeicher gespeichert. Mit der im Speicher enthaltenen Information kann der Bewegungsablauf beliebig wiederholt werden,

[3]Website der Japan Roboter Association. jara.jp. Abgerufen Mai 25, 2013

- Numerical Control Robot: Dieses Handhabungsgerät arbeitet ähnlich wie eine NC-gesteuerte Maschine. Die Information über den Bewegungsablauf wird dem Gerät über Taster, Schalter oder Datenträger zahlenmäßig eingegeben,
- Intelligent Robot: Diese höchste Roboterklasse ist für Geräte gedacht, die über verschiedene Sensoren verfügen und damit in der Lage sind, den Programmablauf selbsttätig den Veränderungen des Werkstücks und der Umwelt anzupassen.

Die Definition der Robotic Industires Association[4], des Robotic Industries Association, ist die folgende:

> A robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks.

Diese Definitionen werden auch in der Literatur aufgegriffen und sind beispielsweise auch bei Nehmzow zu finden [Neh02].

3.2 Robotermodelle

Zur Umsetzung der Arbeit standen folgende mobile Robotermodelle zur Verfügung.

- Lego Mindstorms NXT 2.0
- Robotino
- Sphero

Alle verfügen über verschiedenste Funktionialitäten und Möglichkeiten der Fortbewegung und konnten in Rahmen dieser Arbeit zu Verwendung in Betracht gezogen werden. Näher erklärt werden die Modelle in Kapitel 5.3.

[4]Website der Robotic Industries Association. robotics.org. Abgerufen Mai 25, 2013

3.3 Kinect

Diese Arbeit stellt die Fortsetzung einer früheren Arbeit dar. Wie bereits in [EW13], in Kapitel 3.1 beschrieben, hat sich nichts am grundlegen Stand verändert. Das Kinect for Windows SDK ist mittlerweile in der Version 1.7 verfügbar[5] . In dieser Version unterstützt die Kinect Interaktionen. Diese erlauben eine "Push"sowie eine "Grip"Geste, sowie die allgemeine Verwendung der Hand als Cursor. Mit Kinect Fusion ist es möglich qualitativ hochwertige 3D-Renderings von Objekten oder Personen in Echtzeit darzustellen. Darüber hinaus hat Microsoft eine neue Iteration ihrer Spielkonsole XBox, die Xbox One, vorgestellt, mit der auch ein neuer Kinect Sensor veröffentlicht werden wird[6] .

3.4 Entwicklung der Software

Auch was die Entwicklung der Anwendung bleibt der Stand der gleiche wie bereits in der vorherigen Arbeit beschrieben. In der Entwicklung wurde jedoch auf eine Umsetzung der Benutzeroberfläche mittels der Lightweight Java Game Library (LWJGL)[7] abgesehen. Der Aufwand wäre hier dem Nutzen in einem unangemessenen Verhältnis gegenüber gestanden. Die Benutzeroberfläche wird wie in der vorherigen Arbeit besprochen mit dem Graphical Editing Framework (GEF)[8] und den üblichen Möglichkeiten zur Oberflächengestaltung bei einer RCP[9] umgesetzt.

[5] „Kinect for Windows - Whats New". microsoft.com. Abgerufen Mai 21, 2013
[6] Amanda Holpuch. „Microsoft unveils Xbox One console - as it happened". guardian.co.uk. Abgerufen Mai 22, 2013
[7] „LWJGL Lightweight Java Game Library". lwjgl.org. Abgerufen Mai 20, 2013
[8] „GEF (Graphical Editing Framework)". eclipse.org. Abgerufen Mai 20, 2013
[9] „Rich Client Platform". wiki.eclipse.org. Abgerufen May 20, 2013

4 Mensch-Computer-Interaktion

Die Mensch-Computer-Interaktion ist die Lehre der Interaktion zwischen dem Menschen (dem Benutzer) und Computern. Dieser Begriff ist nicht zu verwechseln mit Mensch-Maschine-Schnittstelle oder Benutzerschnittstelle, die Teil des Fachbereichs Mensch-Computer-Interaktion sind.
Einer der Wegbegründer auf diesem Gebiet war J. C. R. Licklider. Unter der Bezeichnung *Man-Computer Symbioses* hatte er bereits 1960 Ziele und Probleme dieses Teilgebiets der Informatik postuliert [Lic68].

4.1 Man-Computer Symbiosis

Unter Man-Computer Symbiosis verstand Licklider die Entwicklung einer kooperativen Interaktion zwischen Mensch und Computer. Dabei soll eine enge Partnerschaft zwischen Mensch und den elektronischen Geräten geschaffen werden, so Licklider weiter. Vergleichbar heute mit den fortgeschrittenen Smartphonemodellen, die dem Nutzer nicht nur das Telefonieren ermöglichen, sondern auch Internetzugang bereitstellen, als Navigationssystem dienen, und als Spielekonsole fungieren, und aus diesen Gründen für viele Menschen unersetzlich sind. Weiterhin sind zwei Ziele bei einer Symbiose nach Licklider zu erreichen:

1. Computern formuliertes Denken zu ermöglichen

2. Kooperation in der Entscheidungsfindung und der Handhabung von komplexen Situationen von Mensch und Computer, ohne inflexible Abhängigkeiten auf vordefinierte Programme.

Zum damaligen Zeitpunkt bestanden einzelne Rechner aus meterlangen Schranken, die ganze Räume umfassten und über spezielle Terminals gesteuert wurden, die dazu noch einen Bruchteil der Rechenleistung heutiger Mobilgeräte besitzen. Durch den rasanten Wandel der IT-Welt sind daher nicht mehr alle Erkenntnisse für heutige Anwendungen interessant.

4.2 Anwendung des Konzepts auf das Projekt

Im Rahmen der hier dargestellten Anwendung ist es wichtig das Konzept der Man-Computer Symbiosis von dem abzutrennen, was North [**?**] als *mechanically extended man* bezeichnet. Dabei gibt der Benutzer alle für die Handlung entscheidenden Kriterien, wie Richtung, und Integration vor. Die mechanischen Teile stellen ausschließlich eine Erweiterung des Benutzers dar. Die Anwendung soll zwar aus einer Steuerung für einen mobilen Roboter bestehen, es sollen aber noch weitere Elemente enthalten sein, die über den Typ einer Erweiterung hinaus gehen und eine From von Symbiose erreicht werden kann.
Ein Ziel dabei ist nach Licklider [Lic68], den Computer effektiv in Prozesse des *Denkens*, die in Echtzeit ablaufen müssen, zu integrieren. Aufgrund der Tatsache, dass Computer heut zu Tage aus mehreren Komponenten bestehen und auch zum Großteil durch ihre Perepherie bestimmt sind, die wiederum Rechenleistung des Computers zur vorgesehenen Nutzung benötigt, erweitern wir die Aussage von Licklider auf alle Komponenten, die an der Beziehung teilhaben. Diese werden im Folgenden näher betrachtet.

4.2.1 Die Anwendung RoCoVoMo

In der Anwendung *RoCoVoMo* geschieht dies durch die eingebaute Analyse der Gesten- und Sprachinformation durch HMMs. Die Komponenten Kinect und mobiler Roboter sind separate Teilnehmer der Mensch-Computer-Interaktion. Dabei unterstützt die Anwendung den Nutzer durch die stochastischen Auswertungen der Kinect-Daten und die intuitive Führung durch die Anwendung, in dem Gesten- und Spracherkennung dargestellt und entsprechend an dem mobilen Roboter weitergeleitet werden, und im Ernstfall(Blockade der Räder, fahren gegen ein Wandstück) gesonderte Routinen von der Anwendung ausgeführt werden, ohne das der Nutzer hierbei eingreifen kann, oder sollte. Eine Beschreibung der Anwendung und weitere Details zu deren Implementierung können unter Kapitel 7 und Kapitel 8 nachgelesen werden.

4.2.2 Eingabe- und Ausgabegeräte

Kinect

Die Kinect Komponente besitzt herkömmlichen Eingabegeräten gegenüber einige klare Vorteile in Bezug auf die Mensch-Computer-Interaktion. So kann mittels Kinect, ein Teil der formulierten Denkens, wie Licklider es nennt, auf den Computer übertragen werden, indem der Mensch Gesten und Sprachbefehle verwenden kann, die für den Menschen intuitiver zu handhaben sind, und darüber die Umsetzung der gewünschten Aktion dem Computer überlassen wird. Diese kann, der jeweilgen Aktion entsprechend, komplexe Funktionen umfassen, die der Mennschen nicht notwendigerweise durchführen muss. Dabei wird zugleich das zweite Ziel in Lickliders Man-Computer Symbiosis Symbiosis erfüllt, in dem durch die Gestensteuerung unnötige Zwischenschritte, wie diverse Auswahldialoge oder gesonderte Eingabegeräte zu einem Großteil ersetzt werden, und sogleich ein hohes Maßan Kooperation zwischen Mensch und Computer erreicht werden kann.

Mobiler Roboter

Der mobile Roboter wird direkt über die Anwendung RoCoVoMo gesteuert, und durch die Kinect-Steuerung soll die Verbindung zwischen Mensch und mobilen Roboter unmittelbar gestalten. Fr̈ den Roboter bedeutet dies, ein hohes Maßan flexibilität bezüglich Latenzzeit, Beschleunigung, und Bewegungsrichtung. Der Roboter muss unmittelbar reagieren, sobald eine Geste oder ein Sprachbefehl erkannt wird, so wie eine schnelle Bewegung gewünscht ist. Bezüglich Bewegung ist ein holonomer Roboter eine optimale Voraussetzung für eine intuitive Steuerung und einfache Gestenbefehle.

4.2.3 Mensch - User

Für eine effektive Mensch-Computer-Interaktion ist es für den Nutzer der Anwendung notwendig, einfach und schnell Gestenbefehle einzugeben, die genauso schnell in von der Anwendung in Aktionen umgesetzt werden sollen. Hierbei ist aber ebenso ein gewisses Training des Users nötig, da Gesten- und Sprachbefehle nicht die einfachste Form der Steuerung sind. Der Nutzer muss möglichst klar und ohne Dialekt Sprachbefehle geben, sowie er auch das Vokabular, dass die Anwendung umfasst, kennen muss.
Zu Beachten ist, das Sprache eine redundante Form der Kommunikation ist, und Befehle für den untrainierten Nutzer eine andere Aktion der Anwendung suggerieren, als die in Wirklichkeit der Fall ist.
In Hinsicht auf Gesten, ist es wichtig, dass der User konzentriert und in möglichst konzentrierter Haltung Gesten ausführt. Soweit Gesten simpel und intuitiv gewählt wurden, ist es mit wenig Trainer möglich die Anwendung über Gesten zu steuern. Jedoch muss sich der Nutzer sich soweit auf die Anwendung konzentrieren, in dem er sich bewusst sein muss, dass jede Bewegung, die er während aktiver Gestensteuerung ausfübt von der Anwendung wahrgenommen und analysiert und somit die Möglichkeit besteht, eine Aktion der Anwendung zu initiieren ohne dies beabsichtigt zu haben.

4.3 Herausforderungen der Mensch-Computer-Interaktion

Licklider beschreibt weiter Probleme und Herausforderungen der Man-Computer Symbiosis und der Mensch-Computer Kommunikation [LC62]. Auch wenn sich einige der Probleme heutzutage als neglierbar herausstellen, zeigt es doch einige Punkte auf, die bei der Konzeption einer Mensch-Computer-Interaktion zu beachten sind, denn selbst heute noch, werden Aufgaben, die schwer zu automatisieren sind, dem Menschen oft nur aus diesem Umstand übertragen, obgleich es sinnvoller wäre dies dem Computer zu überlassen und dem User nur die Aufgaben zuzuweisen, die für passend sind.
Da der Mensch in der Anwendung RoCoVoMo nicht einfach nur Bediener ist, sondern auch und vor allem zu erst den Umgang mit Gesten lernen muss stellen sich hierdurch auch eine Probleme, die behandelt werden müssen. Licklider betont hierbei insbesondere den kontinuierlichen Informationsaustausch zwischen Mensch und Computer. Das wichtigstes Nebenziel ist, so Licklider weiter, die Zufriedenheit, den Spaß und die Bestätigung des Nutzers zu maximieren. In der modernen IT-Welt würde man dies mit dem Begriff *Gamification* verbinden, in dem ner Nutzer positive Ressonanz durch seine Aktionen erfährt.
In Hinsicht auf den Informationsaustausch ist aber gleichzeitig darauf zu achten, diesen für den Menschen verständlich und zugänglich zu gestalten. In der Anwendung RoCoVoMo beispielsweise, ist sollten die Körperelemente in einer Grafik angezeigt werden und nicht deren Koordinaten ausgegeben werden. Weiteres zu dem Konzept, das aus den hier geschilderten Problemen ensteht, ist in Kapitel 7 vermerkt.
Von den fünf Problemen, die Licklider schildert [LC62], sind nur noch zwei in Hinsicht auf moderne Anwendungen und die Gestensteuerung als Ausgangspunkt von Interesse:

1. Entwicklung eines Programms das es ermöglicht Echtzeitdaten zu verarbeiten und Informationsverarbeitungsprezesse zwischen Mensch und Computer unterstützt. Das Sysyem muss weiterhin *train-and-error* Operationen erlauben. Es muss weiter mit fehlerhaften Eingaben dynamisch umgehen können.

2. Lösung des Problems der Kooperation von Menschen untereinandner bei der Entwicklung großer Anwendungen. Ohne effektives Teamwork ist Man-Computer Symbiosis nicht möglich, in dem Sinne, das verschiedene Anwendungsbereiche auf verschieden konzipiert sein können und somit Unterschiede im Design und des Programms und Verwirrungen bei Nutzer auftauchen. Wenn beispielsweise der 'Schließen'-Button in einem Modul links, in einem anderen rechts liegt, weil verschiedene Entwickler diese implementiert haben, so ist keine intuitive Führung durch die Anwendung mehr möglich.

4.4 Moderne Design Prinzipien

Eine aktuelle Analyse der Mensch-Computer-Interaktion führt Vikas Chahar [Cha12] durch. Dabei wird Mensch-Computer-Interaktion als die Überschneidung von Informatik, Verhaltenswissenschaft, Design und weiterer Wissenschaftsfeldern beschrieben. Sie beinhaltet Software und Hardware und und findet auf der Ebene der Benutzerschnittstelle statt. Weiterhin ist *MCI* auch von der Ergonomielehre abzugrenzen, die grö ßeren Fokus auf ein physische Ebene richtet.
Betrachtet man nun eine solche Benutzerschnittstelle (*eng. user interface*), so führt Chahar folgende Prinzipien auf, die zu beachten sind:

- Ein früher Fokus auf Benutzer und Aufgaben: Es ist zu ermitteln, wie viele Nutzer, welche Aufgaben ausführen müssen und wie diese Aufgaben zu modellieren sind
- Empirische Messungen: Das Interface früh mit Nutzern testen, die die Anwendung auch tatsächlich benutzen werden. Dabei ist es hilfreich statistische Auswertungen durchzuführen
- Iteratives Desgin: Den vorherigen Punkten anschließend, sollten folgende Schritte iterativ durchgeführt werden:
    1. Benutzerschnittstelle (re-)designen

2. Anwendung testen

3. Ergebnisse analysieren

4.5 Design Methodik

Diverse Methodiken verschiedener Techniken für Mensch-Computer-Interaktion existieren, meist daraus entstanden, wie Nutzer und technisches System interagieren. Aufgrund der vorliegenden Gesten- und Sprachsteuerung, stellt sich ein benutzerzentriertes Design (*eng. User-centered design (UCD)*) als sinnvoll heraus, da die Anwendung durch diese Befehle vom Nutzer aus bedient wird.

- User-centered design: UCD beruht auf der Idee, dass der Nutzer im Zentrum des Design einer Computer Anwendung steht. Dabei werden Wünsche, Notwendigkeiten und Grenzen des Benutzers eroiert und die Anwendung diesen Elementen entsprechend erstellt.
- Prinzipien für Benutzerschnittstellen (*eng. User Interface*) Design: Nach Chahar [Cha12] existieren sieben Prinzipien, die bei dem Design eines user interface zu jeder Zeit berücksichtigt werden sollten, nämlich Toleranz, Simplizität, Sichtbarkeit, Konsistenz, Struktur, Feedback, und Aufwand.

Oft werden diese Prinzipien und Begriffe verwendet, in dem man eine Anwendung einfach als ïntuitiv"bezeichnet. Jedoch ist die Bezeichnung intuitiv kein technischer Indikator für eine gute Man-Computer Symbiosis, da dieser stets unterschiedlich ausgelegt und bestimmt wird. Jef Raskin [Ras94] beschreibt den Begriff intuitiv in Bezug auf Mensch-Computer-Interaktion als ein Synonym für etwas bekanntes. In einem Beispiel, an einer Anwendung, die mit einer Computer Maus gesteuert wird, zeigt er, dass dieses Peripheriegerät erst dann *intuitv* in einem Programm nutzbar ist, sobald die Handhabung der Maus bekannt ist. In Bezug auf die Gesten- und Sprachsteuerung bedeutet dies, dass dem Nutzer erst das sprachliche Vokabular und die verwendbaren Gesten bekannt sein

müssen, bevor an eine dem Nutzer leicht fallende Verwendung der Anwendung zu denken ist.
Ebenso wichtig ist das Display Desgin. Zum Einen ist es wichtig, dem Nutzer alle nötigen Informationen und Rückmeldungen über und von der Anwendung anzuzeigen, zugleich muss vermieden werden, den Benutzer mit zu vielen Anzeigen und einem überladenen Bildschirmbereich zu verwirren.
Die Umsetzung und Auswirkung in der Anwendung RoCoVoMo all der zuvor aufgezeigten Prinzipien und Methoden werden im Kapitel 7 im Detail und an weiteren Beispielen beschrieben.

5 Auswahl des Roboters

5.1 Roboter

Diese Arbeit beschäftigt sich mit der Steuerung eines Roboters aus der Ferne. Als Steuerungsinterface wird in diesem Fall jedoch kein traditionelles Interface, wie beispielsweise ein Joystick, verwendet, sondern es kommt das, im ersten Teil dieser Studienarbeit vorgestellte Interface, welches auf Sprach- und Gestensteuerung basiert, zum Einsatz.

Diese Arbeit behandelt die Umsetzung mit einem mobilen Roboter. Wie zuvor bereits erwähnt gibt es keine eindeutige Kategoriesierung für Roboter im eigentlichen Sinne, aber beispielsweise mobile Roboter lassen sich gut durch ihre Art der Fortbewegung und Navigation unterscheiden. Hierzu bieten sich die folenden Kategorien an[10]:

- Manuell ferngesteuerter Roboter. Der Roboter wird von einem Menschen aus der Ferne ferngestuert. Im Idealfall verfügt der Roboter über Kamerasysteme, die es dem Nutzer erlauben die Umgebung des Roboters zu sehen, um so erst eine sinnvolle Navigation zu ermöglichen. Solche Roboter eignen sich vor allem um Arbeiten auszuführen an Orten die für Menschen gefährlich und nur mit sehr großer Schwierigkeit zu erreichen sind. Beispielsweise wäre der Einsaz als Bombenräumroboter denkbar.

[10] „Mobile Robot“. en.wikipedia.org. Abgerufen Mai 26, 2013

- Geschützter ferngesteurter Roboter. Auch dieser Roboter wird aus der Ferne gesteuert, jedoch besitzt dieser zusätzliche Sensoren, die es dem Roboter erlauben Gefahren für sich selbst aufzuspüren und diesen eventuell auszuweichen. Denkbar sind hier Berürungs- oder Abstandsssensoren.
- Linienfolgende Roboter. Solche Roboter sind nur dazu in der Lage einer fest vorgegebenen Bahn zu folgen. Diese ist entweder durch eine auf dem Boden aufgemalte Linie, welche durch Kamerasysteme erfasst und erkannt werden muss, oder aber auch durch Schienen, oder ein Magnetfeld, welchem gefolgt wird realisierbar. Diese Art der Fortbewegung bietet sich in einem vorher planerisch bekannnten Umfeld an. So kann beispielsweise die Fortbewegung von automatisierten Einheiten in einer Fabrik realisiert werden.
- Automatisierte Zufallsbasierte Roboter. Eine sehr primitive Art der Navigation, bei der sich der Roboter einfach nur nach Möglichkeit um erkannte Hindernisse herum bewegen, oder von diesen abprallen, ohne dabei eine tiefere sinnvolle Logik zu verfolgen.
- Autonom geführte Roboter. Diese Roboter verfügen über Sensoren die es ihnen erlauben eine Idee von ihrer Position zu entwickeln. Mittels verschiedener Verfahren, wie beispielsweise Triangulation mit Wegmarken oder dem Einsatz eines GPS ist es dieses Robotern möglich sich eigenständig durch unbekanntes, oder teilweise bekanntes Gelände zu bewegen.

5.2 Anforderungen

Die Anforderungen an einen Roboter im Rahmen des Einsatzes in diesem Projekt sind denkbar gering. Mobile Roboter verfügen in der Regel über Sensoren und Mechanismen zur Erkennung ihrer Umgebung und der selbstständigen Wegfindung in unbekanntem Terain. Dabei handelt es sich um so genannte Autonome Mobile Roboter. Diese sind auf

ihre Art mächtig, brächten jedoch eine Reihe an Funktionen mit, die in diesem Projekt nicht gebraucht werden.

Wie zuvor bereits erwähnt wird der Roboter von einem Nutzer durch ein Gesten- und Sprachbasiertes Interface angesprochen. Die Logik der Handlungen des Roboters kommt demnach vom Nutzer, wodurch der Roboter über keine eigene Intelligenz verfügen muss. Die einzige Anforderungen die demnach verbleiben sind, dass der Roboter mobil sein muss, also über eine Art des Antriebs und der Lenkung verfügen muss und eine Möglichkeit diesen aus der Ferne anzusteuern.

5.3 Mögliche Robotermodelle

Nachfolgend werden kurz die Robotermodelle vorgestellt, mit welchen die Realisierung möglich gewesen wäre. Alle Modelle bieten auf ihre Art eine Unterstützung für die Programmiersprache Java, wie es im Sinne dieses Projektes vorgesehen ist.

5.3.1 Robotino

Der Robotino ist ein Robotik-Lernsystem des Unternehmens Festo Didactic[11] . Der Roboter hat die folgende tecnische Spezifikation[12] :

[11] „Robotino® – Forschen und Lernen mit Robotern“. festo-didactic.com. Abgerufen Mai 26, 2013
[12] „Robotino“. de.wikipedia.orgm. Abgerufen Mai 26, 2013

Tabelle 5.1: Technische Daten des Robotino

Aktoren	Omnidirektionaler Antrieb, 4 Achsen Manipulator, kleiner BHA der Festo AG
Sensoren	Stoßleiste, Abstandssensorik, Kamera, Gyroskop, Laserscanner, Northstar
Rechner	Industrie-PC
Anschlüsse	USB, Ethernet
Kommunikation	WLAN
Energie	Blei-Gel-Akkumulatoren, NiMH-Akkumulatoren

Anwendungen die für den Robotino entwickelt werden können über die WLAN-Schnittstelle aus der Ferne auf dem Roboter ausgeführt werden. Laut Hersteller ist der Robotino über eine Programmierschnittstelle auch mit Java programmierbar. Durch seine holonomen Räder kann der Roboter sich problemlos omnidirektional fortbewegen. In der nachfolgenden Abbildung 5.1 ist der vorgestellte Roboter zu sehen.

Abbildung 5.1: Robotino des Unternehmens Festo Didactic, Quelle: Festo Didactic[13]

[13] „Robotino® – Szenarien und Lernfelder“. festo-didactic.com. Abgerufen Mai 26, 2013

5.3.2 Lego Mindstorms NXT

Vom Spielwarenhersteller LEGO werden unter dem Namen Mindstorms[14] Sets zur Entwicklung verschiedener Roboter vertrieben. Kern des Sets ist der sogennante NXT-Brick, ein intelligenter Baustein zur Ansteuerung der verschiedenen Sensoren und Aktoren, die ebenfalls zum Set gehören. Der NXT wird seit 2006 vertrieben, wird jedoch noch dieses Jahr durch einen aktuelleren Baustein, den EV3[15] abgelöest werden.

Die Hardware des Lego Mindstorms NXT in der Version 2.0 ist wie folgt spezifiziert[16]:

- Atmel-ARM-Prozessor, AT91SAM7S256; 256 kB Flash-Speicher, 64 KB RAM, 48 MHz
- Koprozessor: Atmel 8-Bit AVR, ATmega48; 4 KB Flash-Speicher, 512 Byte RAM, 8 MHz
- Bluetooth: CSR BlueCore 4 v2.0 +EDR; unterstützt das Serial Port Profile (SPP), 26 MHz
- USB-2.0-Anschluss, 12 Mbit/s
- 3 Motorausgänge mit Rückkanal
- 4 Sensoreingänge, analog und digital kombiniert der vierte Eingang kann als High-Speed-Port, entsprechend IEC 61158 Type 4/EN 50170, genutzt werden
- Punktmatrix LC-Anzeige; 100 x 64 Pixel, Abmessungen: 26 x 40,6 mm
- Soundausgabe mit 8-Bit-Auflösung und einer Samplingrate von 2 bis 16 kHz
- Open Source Firmware

[14] Website der LEGO Mindstorms. mindstorms.lego.com. Abgerufen Mai 26, 2013
[15] Announcing LEGO MINDSTORMS EV3. mindstorms.lego.com. Abgerufen Mai 26, 2013
[16] NXT - Hardware-Spezifikation des NXT-Steins. de.wikipedia.org. Abgerufen Mai 26, 2013

Sensoren für den Lego Roboter werden von der Firma HiTechnic entwickelt. In den folgenden Kategorien sind Sensoren verfügbar:

- Infrarot-Sucher
- Kreisel-Sensor
- Farbsensor
- Beschleunigungssensor
- Kompass-Sensor
- Temperatur-Sensor
- EOPD-Sensor

Trotz dessen, dass es sich bei diesem Modell um ein Spielzeug handelt, ist es aufgrund der vielen verfügbaren Sensoren, sowie der Unterstützung der Entwicklung durch Veröffentlichung der Hardware Spezifikationen, wie auch Developer-Toolkits, eine attraktive Platform zur Anwendung im Bereich der Robotik.

Abbildung 5.2: LEGO Mindstorms Set NXT 2.0, Quelle: Festo Didactic[17]

[17]Produktsteite des NXT. mindstorms.lego.com. Abgerufen Mai 26, 2013

5.3.3 Sphero

Der Sphero[18] des Unternehmens Orbotix ist eine Roboterkugel. Er kann mittels Bluetooth angesteuert werden und verfügt auch über Sensoren, ein Gyroskop, ein Beschleunigungssensor und ein Kompass[19]. Die Entwicklung von Anwendungen für den Sphero wird vor allem auf den mobilen Platformen Android und iOS unterstützt, es existieren jedoch auch inoffizielle Framewrosk zur Verwendung des Roboters in anderen Programmiersprachen und -umgebungen.

Abbildung 5.3: Roboter Kugel Sphero, Quelle: Wired[20]

[18] Produktsübersicht des Sphero. gosphero.com. Abgerufen Mai 26, 2013
[19] Angerollt: Roboter Kugel Sphero. heise.de. Abgerufen Mai 26, 2013
[20] Orbotix Sphero - The Shape of Things to Come. wired.com. Abgerufen Mai 26, 2013

Wie auch beim LEGO Roboter NXT handelt es sich hierbei im Grunde um ein Spielzeug. Doch auch hier gibt es Portierungen des SDK die die anderweitige Verwendung der Kugel ermöglichen. Auch dieser Roboter erfüllt grundsätzlich, die durch das Projekt vorgegebenen Anforderungen.

5.4 Umsetzung mit dem LEGO Mindstorms NXT

Im Rahmen dieses Projektes wird der NXT 2.0 aus der Lego Mindstorm Serie eingesetzt[21]. Im Kontext dieser Arbeit sind keine Sensoren notwendig, da der Roboter aus der Ferne gesteuert wird. Die grundsätzlichen Anforderungen werden von allen vorgestellten Robotermodellen unterstützt. Der Lego Mindstorms NXT wurde in erster Linie ausgewählt, da die Duale Hochschule hier über viele Modelle verfügt. Somit war es möglich den Roboter auszuleihen und wesentlich flexibler mit der Entwicklung der Schnittstelle umzugehen, als dies mit den anderen Modellen möglich gewesen wäre. Ebenfalls ausschlaggebend war die die Unterstützung durch Frameworks zur Entwicklung, welches im Falle des NXT durch das Framework leJOS optimal gegeben war. Mehr hierzu im nachfolgenden Kapitel 6.

Der eingesetzte Roboter ist in der nachfolgenden Abbildung 5.4 dargestellt. Das Modell ist so aufgebaut, dass mittels eines Motors der Antrieb, mit der anderen die Lenkung mit einer Lenkachse realisiert ist.

[21] „LEGO® MINDSTORMS® NXT 2.0“. mindstorms.lego.com. Abgerufen Mai 24, 2013

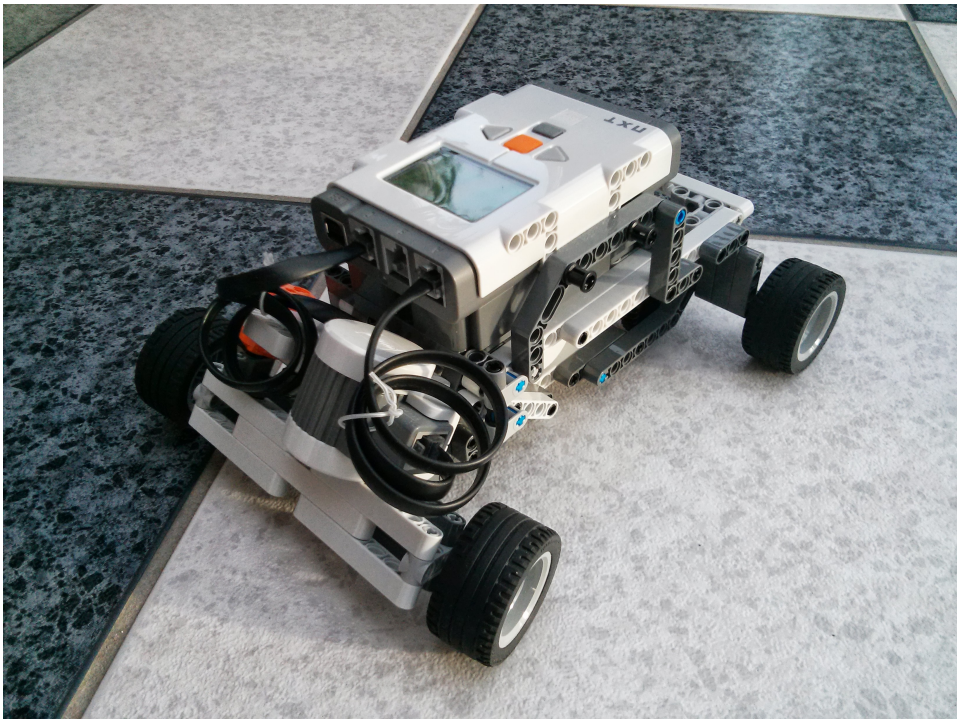


Abbildung 5.4: NXT Robotermodell mit Lenkachse

6 Programmierung des Lego Mindstorms NXT

6.1 Programmierung

Die Programmierung des NXT findet mittels der Entwicklungsumgebung NXT-G statt. Hierbei handelt es sich um einen grafischen Editor, mit dessen Hilfe die Logik des Roboters umgesetzt werden kann. Die Sensoren und Aktoren des Lego Roboters werden als Blöcke dargestellt, über die verschiedene Einstellungen vorgenommen werden können. Weitere Blöcke unterstützen simple Programmlogiken wie Schleifen und Bedingungen. Hiermit soll das Blockbausystem von LEGO immitiert, sowie ein leichter Einstieg in die Programmierung des Roboters geschaffen werden. In Abbildung 6.1 ist ein Ausschnitt eines Programms in der graphischen Oberfläche des Editors zu sehen.

Mit Hilfe dieser Anwendung ist zwar eine leichte Programmierung des NXT möglich, ist aber für dieses Projekt nicht sinnvoll einsetzbar. Die Firmware des NXT, sowie die Spezifikation der Sensoren und Aktoren, wurden von Lego veröffentlicht. Darüber hinaus auch diverse Developer Toolkits. Durch diese Unterstützung sind eine Vielzahl an Frameworks in verschieden Programmiersprachen und Umgebungen entstanden mit denen der NXT-Baustein programmiert oder auch aus der Ferne gesteuert werden kann.

[22] *NXT-G* debacher.de, Abgerufen Mai 22, 2013

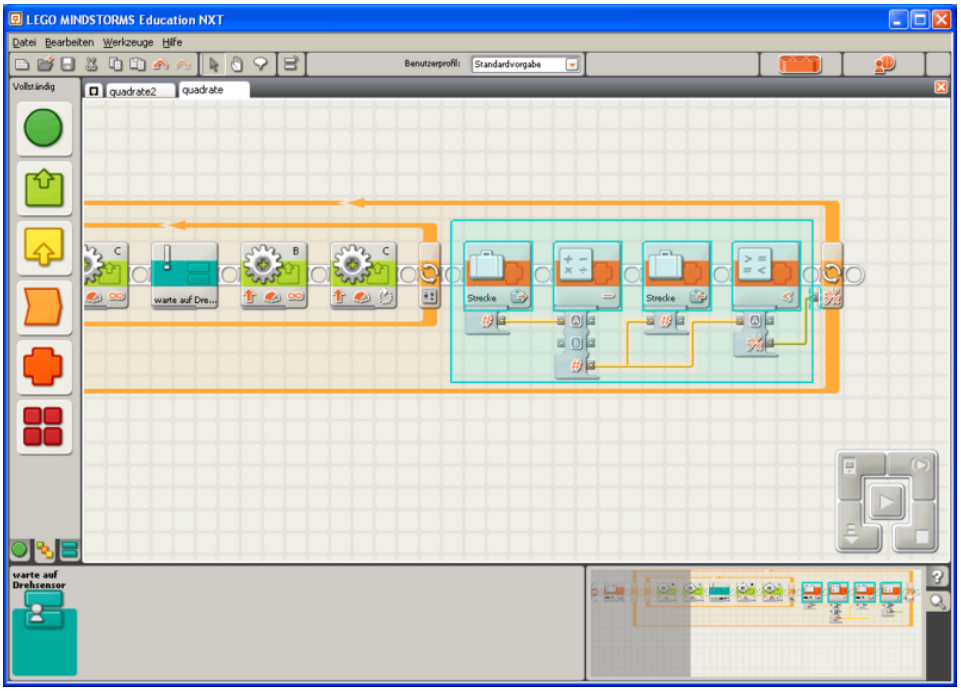


Abbildung 6.1: Oberfläche der Entwicklungsumgebung NXT-G (Quelle: debacher.de/wiki/NXT-G.[22])

6.2 Auswahl eines Frameworks

In dieser Arbeit soll nach Möglichkeit eine Umsetzung in der Programmiersprache Java erarbeitet werden. In der nachfolgenden Tabelle 6.1 ist eine Auswahl an Frameworks zu finden, welche eine solche Umsetzung in der gewünschten Sprache erlauben.

Tabelle 6.1: Java Frameworks zur Programmierung des NXT[23]

Framework	**Sprache**
NXTGCC[24]	C/C++, Objective-C, Fortran, Java, Ada, others
leJOS[25]	Java
TinyVM[26]	Java
Gostai Urbi[27]	URBI, C++, Java, Matlab

Durch das erste genannte Framework NXTGCC, also einen Compiler, ist die Programmierung des Bausteins auf einer sehr hardware-nahen Ebene möglich. Dies bedeutet jedoch auch einen sehr hohen Aufwand bei der Umsetzung des vorhabens.

leJOS ist ein sehr ausgereiftes Framework zur Programmierung des NXT mit Java. Der Code wird auf einer eigens entwickelten Firmware auf dem NXT ausgeführt. Außerdem ist auch Codeausführung von einem Computer möglich. Da dies das Framework ist, welches zur Verwendung in dieser Arbeit ausgewählt wurde, wird es im nachfolgenden Kapitel näher beschrieben.

TinyVM kann eigentlich von vorneherein ausgeschlossen werden, da es sich nur zur Programmierung des RCX Bausteins von Lego handelt. Es wird hier jedoch trotzdem erwähnt, da der RCX der Vorgänger des NXT ist, sowie auch TinyVM der Vorläufer des zuvor genannten NXT darstellt. Wie auch bei leJOS bringt TinyVM eine eigene Firmware für den Lego Roboter mit.

[23] „Lego Mindstorms - Programming languages“. en.wikipedia.org. Abgerufen Mai 22, 2013
[24] „NXTGCC“. nxtgcc.sourceforge.net. Abgerufen Mai 22, 2013
[25] „leJOS“. lejos.sourceforge.net. Abgerufen Mai 22, 2013
[26] „Tiny VM“. tinyvm.sourceforge.net. Abgerufen Mai 22, 2013
[27] „Gostai Urbi - Mindstorms“. urbiforge.org. Abgerufen Mai 22, 2013

Urbi ist kein eigenes Framework zur Ansteuerung des Lego Roboters selbst, sondern eine Open Source Platform die sich der Steuerung von Robotern und komplexen System im allgemeinen zuwendet. Für den NXT existiert hier eine Schnittstelle. Da es sich hier um ein großes Framework mit sehr weitem Anspruch handelt wird von der Verwendung dieses Frameworks abgesehen.

6.3 leJOS - Java for LEGO Mindstorms

leJOS bietet eine breite Palette an Tools zur Programmierung des NXT mit der Programmiersprache Java. leJOS ging als Open Source Projekt aus TinyVM hervor, welches von José Solrzano entwickelt wurde. Kern von leJOS ist die Umsetzung der Java Virtual Machine auf dem NXT Baustein. Diese ist notwendig um Java Code überhaupt erst auszuführen. Die umgesetzten Features werden auf der Website wie folgt beschrieben[28] :

- Object oriented language (Java)
- Preemptive threads (tasks)
- Arrays, including multi-dimensional
- Recursion
- Synchronization
- Exceptions
- Java types including float, long, and String
- Most of the java.lang, java.util and java.io classes
- A Well-documented Robotics API

[28] „NXJ - Technologie“. lejos.sourceforge.net. Abgerufen Mai 22, 2013

Die erwähnte Robotics API ist die NXJ API. Sie stellt die Schnittstelle zu den Aktoren und Sensoren, sowie den Funktionen des Bausteins, wie beispielsweise Bluetooth, des Lego Roboters dar. Zur Verwendng muss die leJOS Firmware auf dem NXT installiert werden. Dies ist durch mitgelieferte Tools leicht möglich. Zur Entwicklung bringt das Framework ein Eclipse-Plugin mit sich, dies ist hilfreich, da die Entwicklung in diesem Projekt ebenfalls mit Eclipse erfolgt.

Die API ist auf zwei Arten realisiert. Einmal zur direkten Verwendung auf dem Baustein und zum anderen zum Ansteuern des NXT aus der Ferne. Es können also Programme zur direkten Ausführung auf dem Roboter oder über das Remote-Interface, zur Ausführung aus der Ferne mittels Bluetooth entwickelt werden. In der Verwendung ist hier keine andere Herangehensweise erforderlich. Es muss lediglich die entsprechend gewünschte Implementierung der API verwendet werden. Zur Verwendung der Bluetooth-Kommunikation ist auch kein erheblich großer Aufwand notwendig, dies wird komplett durch das Framework übernommen.

Das Framework wurde ausgewählt, da es viele Funktionen bereits abdeckt und somit erlaubt, sich im wesentlichen auf das Implementieren der gewüenschten Funktionen zu beschränken. Ebenfalls ist die Verwendung in ausführlicher Dokumentation geschildert.

7 Konzept und Steuerung der Anwendung

Ein Großteil der ersten Arbeit [EW13] betrafen die Konzeption und Entwicklung von Gesten- und Sprachbefehlen, die innerhalb einer Anwendung zur Steuerung eines mobilen Roboters benötigt werden und sinnvoll sind. Diese wurden werden der Arbeiten an der Anwendung nochmals einer Revision unterzogen und nur in Bezug des zugrundeliegenden HMM [BP66] [Rab89] in technischen Parametern für die Implementierung angepasst.
Das gesamte Vokalubar an Gesten- Sprachbefehlen ist fest in der Anwendung RoCoVoMo verankert, und lässt nur durch Anpassungen im Code ändern. Dies ist beabsichtigt, da nur durch diese Begrenzung ein stabiler Gebrauch der Anwendung garantiert werden kann.

7.1 Verwendete Gesten

Gesten, nach Kurtenbach und Hulteen [KH90], die innerhalb der Anwendung genutzt werden sollen, müssen innerhalb eines HMM repräsentiert werden. Aus der Beschreibung des HMM [EW13] müssen daher einige Komponenten in die Anwendung RoCoVoMo integriert werden. Das bedeutet, dass für jede Geste Trainingsdaten vorhanden sein, beziehungsweise in das Programm eingegeben werden müssen, um diese später im laufenden Betrieb zu erkennen.

Die verwendbaren Gesten werden im folgenden noch einmal aufgelistet. Die entsprechenden Klassifikationen und wissenschaftlichen Beschreibungen wurden bereits in der Ausarbeitung von Ebner und Werling [EW13] erörert und werden hier als bekannt vorausgesetzt.

7.1.1 Kreisbewegung

Diese Geste ermöglicht es den *Lego NXT* im Kreis fahren zu lassen. Sobald die Geste über das HMM und die vorhandenen Trainingsdaten abgegelichen wurde, wird die Anwendung die Aktion ausführen. Die Abbildung 7.1 zeigt die idealisierte Darstellung der Geste.

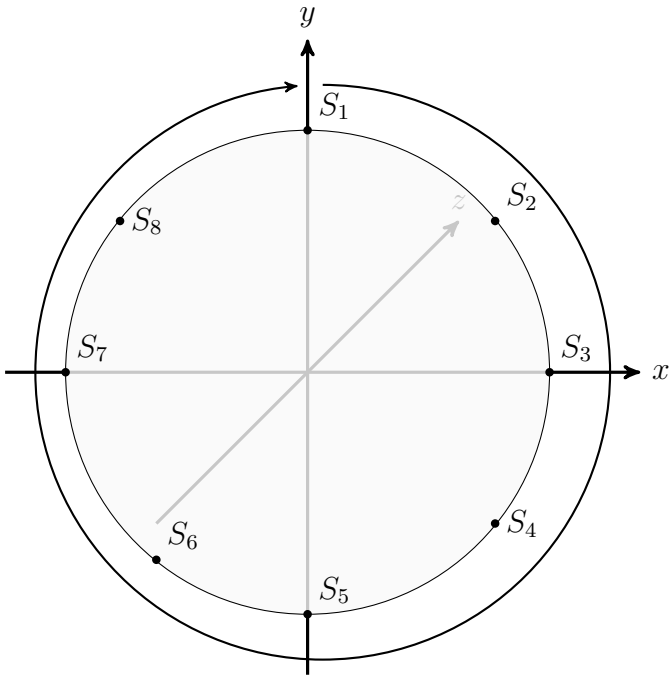


Abbildung 7.1: Abstrakte Darstellung einer Kreisbewegung im Koordinatensystem inklusiver ihrer 8 Zustandspunkte

Aufgrund der restriktionen aus dem mobilen Roboter, in dieser Arbeit der Lego NXT, ist aus dessen Motorspezifikationen und die hier gewälte Bauform, ist es nicht möglich

einen beliebig großen oder kleinen Kreis zu fahren. Weiterhin unterstützt dies die in Abbildung 7.1 gezeigte Geste und das darunterliegende genutzte HMM nicht.

7.1.2 Vorwärtsbewegung

Technisch ist die Vorwärtsbewegung in verschiedene Varianten aufgeteilt. Der Anwendungsnutzer soll bei der Verwendung der jeweiligen Geste aber nichts davon spüren, beziehungsweise, ihm soll diese Aufteilung nicht betreffen.
Dabei wird unterschieden, in der einfachen Variante einer geradlinigen Bewegung, dargestellt in Abbildung 7.2, und zwei erweiterten Gesten, die eine Richtungsänderung beinhalten, dargestellt in den Abbildungen 7.3(a) und 7.3(b).

Standardvariante

Diese Variante beschreibt die Geste für eine geradlinige Vorwärtsbewegung, in einen einfachen Aktionsbefehl für den Lego NXT umgesetzt wird.

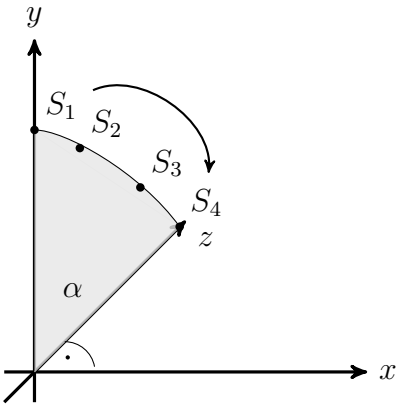


Abbildung 7.2: Abstrakte Darstellung einer Vorwärtsbewegung im Koordinatensystem inklusiver ihrer 4 Zustandspunkte

Die oben beschriebene Vorwärtsbewegung beinhaltet keine Information über einen Winkel, in dem sich der Roboter fortbewegen soll. Um diese Information zu erhalten, muss die Geste und das darunter liegende HMM angepasst und erweitert werden.

Erste Variante der erweiterten Vorwärtsbewegung

Durch die Erweiterung der Standardvariante um eine seitliche Bewegung wird es möglich Richtungsinformationen aus der Geste auszulesen und diese in eine gerichtete Bewegung für den Lego NXT umzusetzen. Dabei ist zu beachten, dass durch die relativ ungenaue Auswertung der Körperelemente durch die Kinect, die Anwendung nicht in der Lage ist einen exakten Richtungswinkel zu ermitteln. Es wird ein grober Richtwert verwendet, der den User bei der anfänglichen Nutzung etwas irritieren kann, was aber mit außreichen Training des Users auf diese Tatsache kompensierbar ist, in dem er eine Richtungsänderung anzeigt.

Zweite Variante der erweiterten Vorwärtsbewegung

In dieser Variante wird bereits mit Beginn der Geste eine Richtungsinformation mitgeliefert. Diese Form der Vorwärtsbewegung ist vor allem für versierte Nutzer gedacht und benötigt auch ein gewisses Training und die zuvor beschriebene Ungenauigkeit zu kompensieren.

7.1.3 Haltesignal

Das Anhalten des Lego NXT ist die wohl wichtigste Aktioin und auch in einer simplen Geste umgesetzt, zumindest für den User, denn auch hier muss auf der technischen Ebene der HMM differenziert werden. Da der Anwendungsnutzer nämlich aus den verschiedenen Formen der Vorwärtsbewegung einen Halt initiieren kann, muss dies auch

beim Haltesignal berücksichtigt werden. Abbildungen 7.5(a) und 7.5(b) zeigen beide Varianten auf.

7.1.4 Entriegeln — Blockieren

Um den Nutzer die Gestensteuerung zu erleichtern, ist es durch eine gesonderte Geste möglich, die Echtzeitverarbeitung seiner Bewegungen zu stoppen und den Lego NXT somit auch automisch zu einem Halt zu bringen. Abbildung **??** zeigt eine idealiserte Darstellung dieser Geste.

7.2 Trainingsmodul

Um die zuvor beschriebenen Gesten zu erkennen, müssen Trainingsdaten in das System eingelesen werden. Hierzu wurde ein Modul entwickelt und in die Anwendung integriert, dass das schreiben und einfügen von Trainingsdaten in RoCoVoMo ermöglicht. Dabei wird dei Kinect in einer Testumgebung gestartet und über das jnect Framework die Daten der gewünschten Geste, beziehungsweise des jeweilgen Body Elements eingelesen. Benötigte Informationen sind dabei nur die Koordinaten des jeweilgen Elements und die Anzahl der verwendeten HMM Zustandspunkte. Das jahmm Framework benötigt für die Errechnung eines HMM besonders gefilterte Daten, daher werden die Kinectinformationen in einem besonderen Format gespeichert, anschließend innerhalb der Anwendung ohne weitere Änderungen verwendet werden können. Die technischen Details dieses Moduls werden näher in Kapitel 8 beschrieben.
In Kapitel 4 wurde beschrieben, welche Merkmale entscheiden sind für eine gute Mensch-Computer-Interaktion und welche Gesichtspunkte berücksichtigt werden müssen. Die Kernfunktion, im Fall des Traininsmoduls, ist das Einlesen von Kinectdaten über ein Interface. Dabei ist das Design und die einfache Bedienung der grafischen Oberfläche sehr wichtig. In dem Modul selbst können zur Bedienung jedoch noch keine Gesten verwendet

werden, dies ist erst im Modul der Robotersteuerung möglich, da zuvor noch keine Trainingsdaten vorhanden sind. Da es sich weiterhin um ein User-centered design handelt ist es wichtig im Sinne der Man-Computer Symbiosis dem Benutzer eine Echtzeitverarbeitung und einfachen Informationsaustausch zu ermöglichen. Dies wird im Modul dadurch verwircklicht, in dem der Nutzer mit der Aufzeichnung seiner Bewegungen direkt und in Echtzeit eine Anzeige seiner Daten erhält und diese darüber hinaus manuell nachbearbeiten kann. Dabei wurde das Interface anhand der in Kapitel 4 vorgestellten Design Prinzipien von Chahar [Cha12] simpel und klar strukturiert und durch die manuelle Nachbearbeitung auf ein hohes Maß an Sichtbarkeit geachtet.

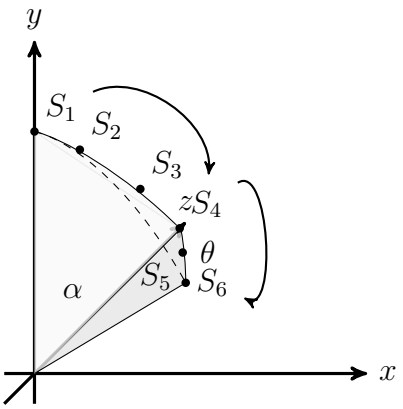
7.3 Kinect—Modul

Als ein Kernelement einer Man-Computer Symbiosis, gilt der Informationsaustausch. Jedoch muss dabei darauf geachtet werden, diesen in einer Form zu gestalten, dass der menschliche Teil der Beziehung dabei nicht von den Ausgaben überfordert wird, und er diese in für ihn passender Form erhält. Im Falle der Gesten- und Sprachsteuerung mittels Kinect, werden zunächst lediglich drei dimensionale Koordinaten der einzelnen Körperelemente übermittelt. Diese sind für den Menschen nur äußerst umstülich zu lesen und stellen keine geeignete Form der Darstellung dar. Daher wurde ein Modul erstellt, dass mittels diesen Koordinaten ein grafisches Bild der Körpers des Menschen mit allen erfassten Elementen anzeigt, deren Koordinaten von der Kinect übermittelt werden. Hiermit wird ebenfalls nahe am User-centered design gearbeitet und die Prinzipien, wie Simplizität und Feedback verfolgt, in dem der Nutzer ein einfaches Bild seiner selbst sieht.
Ein weiteres Merkmal, dass mittels des zugrundeliegenden OSGi-Frameworks möglich ist - bereits in Teil 1 der Studienarbeit [EW13] vorgestellt - ist, dass dieses Modul in jedem Teil der Anwendung aufrufbar ist und der Benutzer zu jeder Zeit Feedback und grafische Informationen über seine Erfassung durch die Kinect erhalten kann. Dies kann auch als Hilfe bei der Modellierung weiterer Gesten, der Weiterentwicklung der Anwendung und

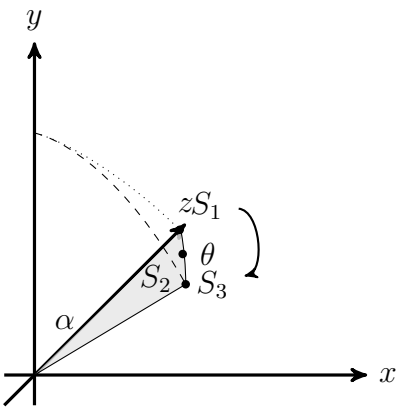
dem Auffinden von Fehlern und Fehlverhalten bei der Erkennung von Gesten und der Umsetzung in die entsprechende Aktion dienen.
Details zur Umsetzung dieses Moduls werden in Kapitel 8 beschrieben.

7.4 Roboterinterface

In diesem Modul kann der User die über Gesten den Lego NXT steuern. Dabei werden Anhand der Trainingsdaten bei Anwendungsstart die diversen HMM aller Gesten generiert und gespeichert. Sobald nun das Interface geöffnet wird, um den Roboter zu steuern, werden eben diese HMM verwendet und mit den Echtzeitdaten aus der Kinect abgegelichen und entsprechend Gesten erkannt und automatisch in Aktionen für den Lego NXT übertragen.
Diese Modul ist ebenso wie die vorherigen nach dem User-centered design konzipiert und richtet sich nach den Design Prinzipien von Chahar. Als besonders wichtig hierbei, stellt sich die grafische Darstellung des Roboters und der Ein- und Ausgabe heraus. Dem Menschen fällt es schwer, sich auf mehrere Dinge gleichzeitig zu konzentrieren, in Bezug auf die Anwendung, auf die grafische Anzeige am Computer und den Lego NXT im Raum. Daher wird mittels einer grafischen Anzeige innerhalb der Anwendung versucht, dieses Problem zu lösen, wobei der Nutzer hierbei nur eine idealisierte Ansicht der Realität vermittelt bekommt, die auch nur die Anzeige des Roboters und dessen Ein- und Ausgabe umfasst, und nicht etwaige Hindernisse im Raum.
Eine entsprechende Lösung dieses Problems, wäre die Installation einer Kamera auf dem Lego NXT, und die Ausgabe dieser Videoinformationen innerhalb der Anwendung.
Alle weiteren Informationen dieses Moduls und dessen Umsetzung in der Anwendung sind in Kapitel 8 zu finden.



(a) Darstellung der gesamten erweiterten Vorwärtsbewegung mit 6 Zuständen



(b) Detaildarstellung der Seitwärtsbewegung der erweiterten Vorwärtsbewegung und ihrer 2 Zustände

Abbildung 7.3: Idealisierte Darstellung der 1. Variante einer erweiterten Vorwärtsbewegung mit 6 Zuständen

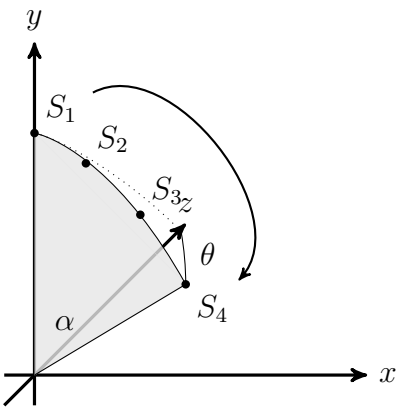
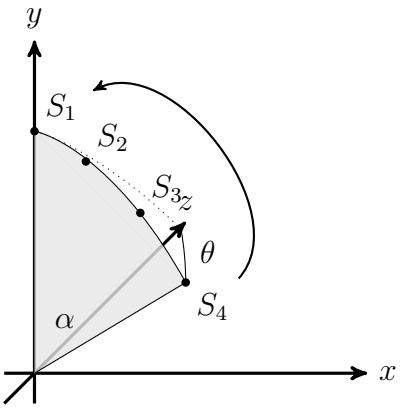
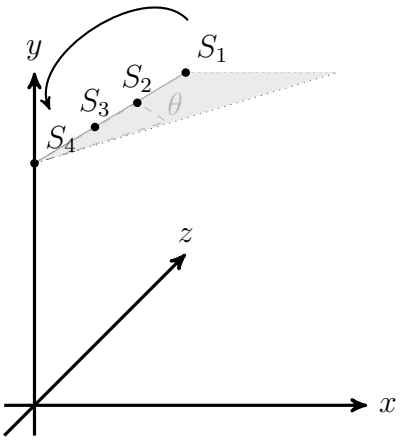


Abbildung 7.4: Idealisierte Darstellung der 2. Variante einer erweiterten Vorwärtsbewegung mit 4 Zuständen



(a) Abstrakte Darstellung des Haltesignals aus einer möglichen Vorwzärtsbewegung heraus, mit ihren 4 Zuständen



(b) Abstrakte Darstellung des Haltesignals aus einer möglichen Kreisbewegung heraus, mit ihren 4 Zuständen

Abbildung 7.5: Abstrakte Darstellung der Modelle für die Geste des Haltesignals mit jeweils 4 Zuständen

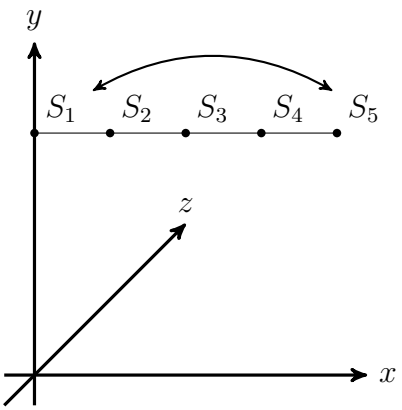


Abbildung 7.6: Idealisierte Darstellung der Geste zum Blockieren und Freigeben der Gesteneingabe mit 4 Zuständen

8 Implementierung

Die Anwendung RoCoVoMo wird in der Programmiersprache Java geschrieben und verwendet diverse Frameworks, die im folgenden kurz beschrieben werden:

- *OSGi*: Ermöglicht die Aufteilung der einzelnen Komponenten der Anwendung in sogenannte *Bundles* oder Module die getrennt voneinander verwendet werden können
- *jahmm*: Liefert alle Algorithmen und Komponenten, die zur Erstellung von HMMs benötigt werden
- *jnect*: Dient als Schnittstelle zwischen Kinect, Kinect SDK for Windows und der Java-Anwendung
- *lwjgl*: Ermöglicht die Verwendung von drei dimensionalen Grafiken der OpenGL Bibliothek

Darüber hinaus wird die gesamte Java-Anwendung innerhalb der Eclipse Integrated development environment (IDE) entwickelt und auf der aktuellen Juno 4.2 Plattform umgesetzt. Die Anwendung selbst wird in Form einer RCP entwickelt und damit das OSGi Framework von Eclipse, Equinox, verwendet.
Im Folgenden werden die einzelnen Bestandteile der Anwendung näher beschrieben.

8.1 Architektur — Design

Seit den Arbeiten an der Anwendung für den ersten Teil der Arbeit wurde das Design und die Architektur der Anwendung nochmals überarbeitet. Abbildung 8.1 zeigt den aktuellen Stand der Anwendung.
Die Arbeiten an der Architektur entfielen auf beide Autoren gleichermaßen, wobei Volker Werling einen Großteil der Überarbeitung des Designs für den zweiten Teil der Studienarbeit übernahm.

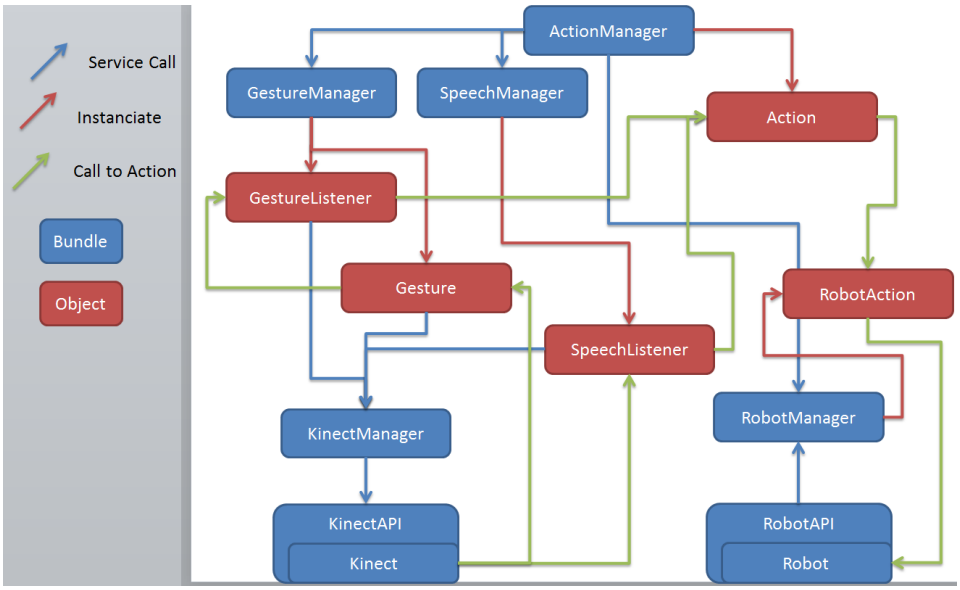


Abbildung 8.1: Architekturdiagramm der Anwendung RoCoVoMo

Die mit der Farbe Blau markierten Elemente stellen sogenannte OSGi-Bundles dar. Dabei handelt es sich vereinfacht gesprochen um einen Container, der eine bestimmte Logik beziehungsweise Fachlichkeit abdeckt und modular verwendbar ist. In der Anwendung RoCoVoMo werden mehrere solcher Bundles verwendet, die im Folgenden weiter beschrieben werden. Mit Rot markierte Elemente sind Java Klassen, die als Objekte innerhalb der Bundles verwendet werden.

8.1.1 ActionManager

In einer sogenannten *Managerklasse*, in diesem Fall dem *ActionManager* werden alle Aktionen verwaltet, die innerhalb der Anwendung ausführbar sind. Die Managerklasse führt dabei selbst keine Fachlichkeit aus, sondern delegiert diese an die darin verwalteten Objektinstanzen weiter, hier den einzelen Aktionen und ebenso an damit verbundene Bundles, hier *GestureManager*, *SpeechManager* und *RobotManager*.
Diese Aufteilung ermöglicht es, Aktionen flexibel in die Anwendung zu integrieren, abzuändern oder gar zu löschen. Weiter ist durch die loose Verbindung über sogenannte *Service Calls*, zum Beispiel des GestureManagers keine Abhängigkeit zu darin verwalteten Gesten vorhanden. Die einzige Verbindung zwischen Gesten und Aktionen findet im GestureListener statt, hierzu später mehr.
Gleiches gilt für Sprache und der Roboteransteuerung.

Action

Die *Action* ist so konzipiert, dass sie nicht direkt von einem Robotertyp abhängig ist. Diese Design ermöglicht sogar den relativ einfachen Umbau der Anwendung und Einsatz eines alternativen Robotermodells zum Lego NXT.
Wird eine Aktion ausgeführt, so ruft diese weiter die *RoboterAction* aus, die die weitere Steuerung des Roboters übernimmt, bisher der Lego NXT. Aufgerufen wird dieses Objekt von den Objekten *GestureListener*, und *Speechlistener*.

8.1.2 GestureManager

GestureManager verwaltet alle in der Anwendung verfügbaren Gesten. Dabei existiert pro *Gesture*-Objekt ein *GestureListener*.

GestureListener

GestureListener liest Kinectinformationen aus und gleicht diese mit den HMMs ab. Dabei werden über spiezelle *Recognizer*-Klassen, die Kinectdaten als HMM-Sequenzen eingelesen und Wahrscheinlichkeitsauswertungen im HMM die richtige Geste erkannt und das entsprechende *Gesture*-Objekt in Folge ausgeführt.

Gesture

Gesture beinhaltet zahlreiche Meta-Informationen über die entsprechende Geste und ist über einen *Actioncall* mit dem zugewiesenen *Action*-Objekt verbunden. Wird das Gesture-Objekt vom Listener ausgeführt, so ruft Gesture diese Aktion auf.

8.1.3 SpeechManager

SpeechManager verwaltet alle Sprachbefehle. Da Sprachbefehle bereits innerhalb der Kinect SDK integriert sind, und dieser lediglich über String-Objekte und der entsprechenden Sprache hinzugefügt werden müssen, wurde darauf verzichtet ein gesondertes Objekt Speech zu modellieren, da dieses ledigliche die String Information enthalten würde. Diese Daten sind im *SpeechListener* hinerlegt.

SpeechListener

SpeechListener führt, sobald ein Sprachbefehl von der Kinect erkannt wurde, das entsprechende *Action*-Objekt aus.

8.1.4 KinectManager

KinectManager verwaltet die Verwendung von und den Datenaustausch mit der Kinect. Dieser wird von den Objekten *GestureListener*, *Gesture*, und *SpeechListener* über Service Calls aufgerufen.
Durch den Manager wird die KinectAPI und darin enthaltene Kinect für alle Teilnehmer sichtbar und dadurch kann ohne direkte Abhängigkeiten auf die Kinectinformationen zugegriffen werden.

8.1.5 RobotManager

RobotManager verwaltet die Verwendung der RobotAPI und des darin enthaltenen Roboters. Bisher wird dadurch der Lego NXT verbunden. Durch diese Architektur ist jedoch auch möglich weitere Robotermodelle in die Anwendung RoCoVoMo zu integrieren und weitere Abhängigkeiten zu erhalten.
Der Manager verwaltet weiterhin die Objekte *RobotAction*, die dann aber je nach Robotermodell individuell geschrieben werden müssen, da diese direkt auf die Roboter Programmierschnittstelle zugreifen, und diese sich von Modell zu Modell unterscheiden kann.

8.2 Trainingsmodul

Das Trainingsmodul wurde konzeptionell bereits in Kapitel 7 beschrieben. In diesem Abschnitt werden die technischen Details dieser Komponente näher erläutert.
Abbildung 8.2 zeigt einen Ausschnitt des Moduls in der laufenden Anwendung, welche auf einer RCP basiert.
Um Trainingsdaten zu speichern muss zu Beginn eine Geste ausgewählt werden, und anschließend der Intervall Parameter angegeben werden, nach wie viel Sekunden die

Elementkoordinaten der Kinect gespeichert werden sollen. Sofern eine Datei zum Speichern der Werte ausgewält wurde, kann über Start die Aufzeichnung beginnen. In der linken Hälfte des Fensters werden anschließend die Werte angezeigt und über den Button Preview kann darüber hinaus der derzeitige Inhalt der Datei betrachtet werden, was für Experten eine Erleichterung bei Fehlersuche oder Korrekturen darstellt.
Dieser Vorgang wurde simpel konzipiert und es ist nicht schwer, sich in der Anzeige zurecht zu finden, jedoch erfordert das Aufzeichnen der Daten einiges an Training und Geduld, und muss mit Sorgfalt geschehen, da die nur mit korrekten und sauberen Trainingsdaten eine effektive Gestensteuerung ermöglicht werden kann. Darüber hinaus muss dieser Vorgang vielfach wiederholt werden, da nur über eine Vielzahl von Trainingsdaten ein stabiles HMM generiert werden kann. Als Richtwert sollten stets zwischen 80 und 100 Sequenzen in einem Trainingssatz für eine Geste vorhanden sein.
Die Arbeiten an diesem Modul wurden von Simon Ebner durchgeführt.

8.3 Kinect-Modul

Das Konzept des Kinect-Moduls wurde bereits in Kapitel 7 erläutert. Hier wird weiter auf die technischen Feinheiten dieser Komponente eingegangen.
Ein Aussschnitt des Moduls inst in Abbildung 8.3 dargestellt. Dieser zeigt die grafische Darstellung eines sogenannten *Skeletons* der Kinect, innerhalb der Anwendung RoCoVoMo.
Ein Großteil der Informationen die für die Darstellung dieses grafischen Objekts notwendig sind, liefert das *jahmm*-Framework. In der Anwendung RoCoVoMo werden diese Daten über das KinectManager Bundle bereitgestellt und können so in einem sogenannten *View* angezeigt werden. Diese Anzeige kann auch während der Trainingsaufzeichnung, oder der Robotersteuerung weiterlaufen und beeinflusst nicht die Performance der Anwendung.
Sobald das Modul gestartet wird, verbindet sich das View automatisch mit der Kinect

Instanz und die grafische Anzeige wird generiert.
Die Arbeiten an diesem Modul wurden von Simon Ebner durchgeführt.

8.4 Roboterinterface

Das Roboterinterface wurde konzeptionell bereits in Kapitel 7 beschrieben. In diesem Abschnitt werden die technischen Details dieser Komponente näher erläutert. Dabei muss jedoch angemerkt werden, dass die Umsetzung dieses Moduls in Form eines Graphical User Interface (GUI) aus Zeitgründen nicht möglich war. Die Arbeiten an diesem Modul wurden aufgrund des Umfangs auf beide Autoren verteilt. Volker Werling arbeitete an der Robotersteuerung und den Sprachbefehlen, und Simon Ebner an den Bereichen HMM, sowie Design und Gestaltung der Gesten.
Ziel der Arbeiten an diesem Modul war es ein drei dimensionales grafisches Interface zu gestalten, in dem der User eine vereinfachte Sicht der Roboters, sowie eine Anzeige seiner Eingaben, wie Gesten und Sprachbefehle, und eine Ausgabe der jeweilgen erkannten Aktion. Als Zeitraubend hat sich dabei die Kombination und Interkation der vielen Komponenten dieses Interfaces erwiesen. So konnten ohne Trainingsdaten nur Simulationen durchgeführt werden und die Robotersteuerung war auf einzelne vordefinierte sequenzielle Anweisungen beschränkt. Weiterhin stellte sich die Erstellung des grafischen Interfaces an sich als Herausforderung heraus, in dem Sinne, dass die Autoren bislang nur wenig Erfahrung mit 3D Design und Entwicklung von OpenGL Anwendungen hatten.

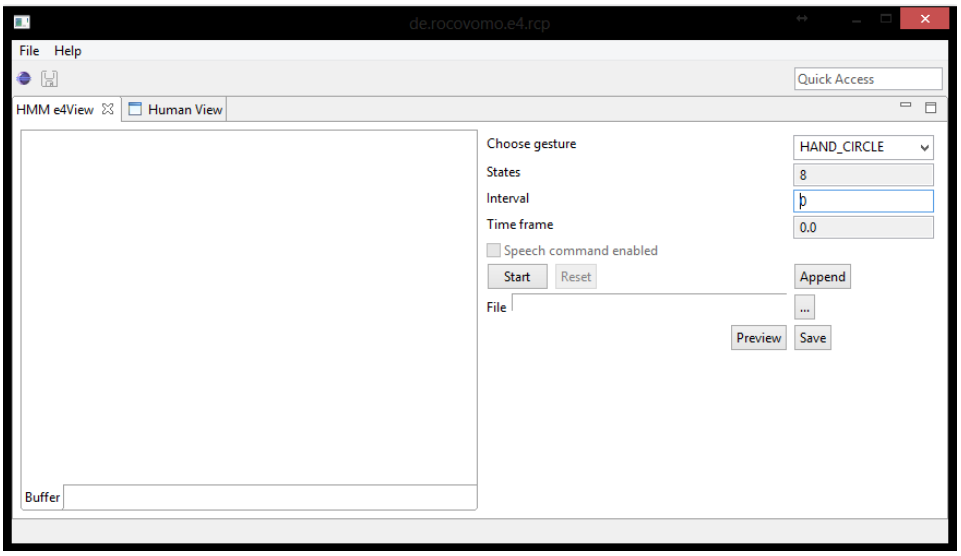


Abbildung 8.2: Ausschnitt des Trainingsmoduls der RoCoVoMo Anwendung

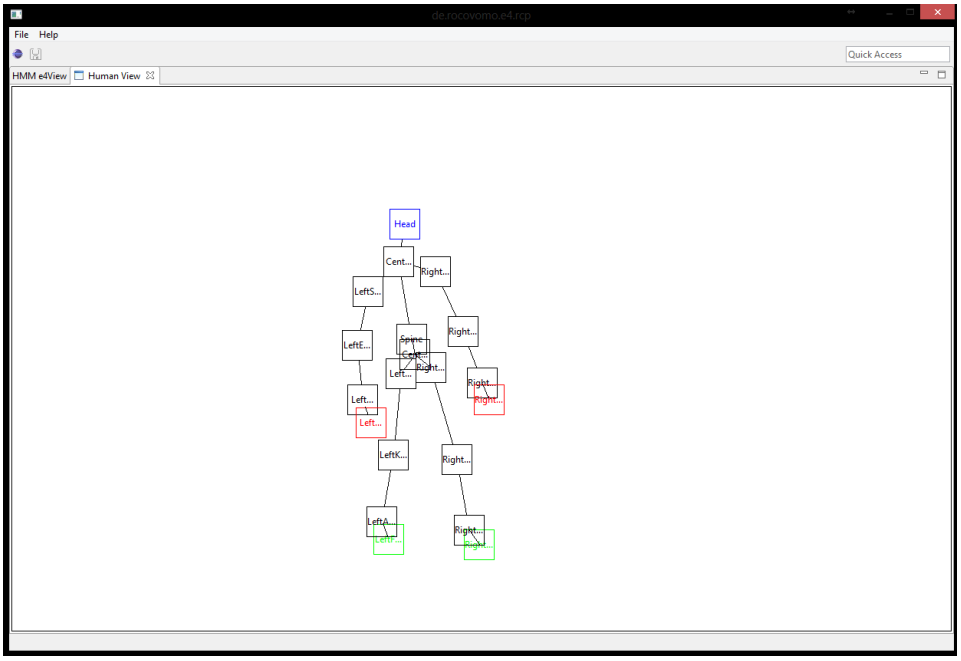


Abbildung 8.3: Ausschnitt des Kinect-Moduls der RoCoVoMo Anwendung

9 Ausblick - weitere Arbeiten

Im Rahmen dieser Arbeit wurde ein Anwendung und ein Framework zur Steuerung eines Roboters mittels Gesten und Sprachbefehlen entwickelt. Die dazu erforderlichen theoretischen Grundlagen wurden in einer ersten Arbeit [EW13], welche sich hauptsächlich mit der Umsetzung des User-Interface mit Gesten und Spracheingabe, erarbeitet und mit den Erkenntnissen dieser Arbeit, die sich mit der Umsetzung der Schnittstelle zu einem mobilen Roboter widmet, verknüpft. In der ersten Arbeit wurden ausführlich die verschiedenen Verfahren zur Erkennung von Gesten und Sprache erläutert, sowie im speziellen der verwendete Ansatz derversteckten Markov Modelle. In dieser Arbeit sind vor allem die Grundlagen der Entwicklung mit einem Roboter, in diesem Fall der Lego Mindstorms NXT dargelegt. Die entwickelte Softwarelösung soll der Dualen Hochschule Karlsruhe zur weiteren Verwendung zur Verfügung gestellt werden.

Wie im allgemeinen üblich ist eine Software niemals wirklich fertig gestellt, es wird immer noch Potential zur Verbesserung oder Weiterentwicklung geben. Das entwickelte Framework stellt eine gute Grundlage zur weiteren Verwendung der Kinect von Microsoft zur Entwicklung einer mächtigen Benutzerschnittstelle zur natürlichen Steuerung von Robotern. Die besondere Herausforderung lag hier vor allem in der Umsetzung der Hidden Markov Modells zur Erkennung von Gestern anhand von Koordinaten.

Noch notwendige Anpassungen und Verbesserungen werden von den Durchführenden, Herrn Ebner und Herrn Werling, soweit möglich auch nach Abschluss der Arbeit noch durchgeführt werden. Hierzu zählen unter anderem die Umsetung einer Graphischen

Oberfläche zur besseren Verwendung des Roboters, sowie das entfernen möglicher Fehlerquellen in den bereits entwickelten Komponenten und auch das anfertigen von besserer Dokumentation zu Verwendung.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[BP66] E. Leonard Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, Vol. 37, No. 6:1554–1563, 1966.

[Cha12] Vikas Chahar. An analytical study on hci. *International Journal of Computer Science and Management Studies*, 12(01):251–254, 2012.

[EW13] Simon Ebner and Volker Werling. Gesten- und sprachsteuerung fuer einen mobilen roboter mittels kinect for windows unter java. Vol. 1:1–101, 2013.

[KH90] G. Kurtenbach and E. A. Hulteen. Gestures in human-computer communication. In *The Art of Human-Computer Interface Design*. Addison-Wesley Publishing Co., May 1990.

[LC62] J. C. R. Licklider and Welden E. Clark. On-line man-computer communication. In *Proceedings of the May 1-3, 1962, spring joint computer conference*, AIEE-IRE '62 (Spring), pages 113–128, New York, NY, USA, 1962. ACM.

[Lic68] J. C. R. Licklider. Man-computer symbiosis. In W. D. Orr, editor, *Conversational Computers*, pages 3–5. Wiley, New York, 1968.

[Neh02] Ulrich Nehmzow. *Mobile Robotik: Eine praktische Einführung*. Springer, 2002.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, Vol. 77, No. 2:257–286, 1989.

[Ras94] Jef Raskin. Viewpoint: Intuitive equals familiar. *Commun. ACM*, 37(9):17–18, September 1994.

Glossar

Eclipse

Eclipse (von englisch eclipse ‚Sonnenfinsternis', ‚Finsternis', ‚Verdunkelung') ist ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Ursprünglich wurde Eclipse als integrierte Entwicklungsumgebung (IDE) für die Programmiersprache Java genutzt, aber mittlerweile wird es wegen seiner Erweiterbarkeit auch für viele andere Entwicklungsaufgaben eingesetzt. Für Eclipse gibt es eine Vielzahl sowohl quelloffener als auch kommerzieller Erweiterungen. 50

Equinox

Equinox (von englisch Tag- und Nachtgleiche) ist ein von der Eclipse Foundation entwickeltes Java-basiertes Framework, welches die OSGi-Kernspezifikation implementiert und das Gerüst der integrierten Entwicklungsumgebung Eclipse bildet. 50

Gamification

Als Gamification oder Gamifizierung (seltener auch Spielifizierung) bezeichnet man die Anwendung spieltypischer Elemente und Prozesse in spielfremdem Kontext. 19

Holonom

Ein holonomes System von Körpern zeichnet sich dadurch aus, dass sich die Lage der Körper durch n generalisierte Koordinaten $q_1, q_2, \ldots, q_n$ beschreiben lässt, die ganzlich unabhan

Man-Computer Symbiosis

Man-computer symbiosis is an expected development in cooperative interaction between men and electronic computers. It will involve very close coupling between the human and the electronic members of the partnership. 15–17, 19–21, 44

Mensch-Computer-Interaktion

Die Mensch-Computer-Interaktion (englisch *Human-Computer Interaction, HCI*) als Teilgebiet der Informatik beschäftigt sich mit der benutzergerechten Gestaltung von interaktiven Systemen und ihren Mensch-Maschine-Schnittstellen. 10, 15, 17–21, 43

Mensch-Maschine-Schnittstelle

Die Benutzerschnittstelle (nach Gesellschaft für Informatik, Fachbereich Mensch-Computer-Interaktion auch Benutzungsschnittstelle) ist die Stelle oder Handlung, mit der ein Mensch mit einer Maschine in Kontakt tritt. 15

OSGi

Die OSGi Alliance (früher Open Services Gateway initiative) spezifiziert eine hardwareunabh¨¨angige dynamische Softwareplattform, die es erleichtert, Anwendungen und ihre Dienste per Komponentenmodell („Bundle“/„Service“) zu modularisieren und zu verwalten („Service Registry“). Die OSGi-Plattform setzt eine Java Virtual Machine (JVM) voraus und bietet darauf aufbauend das OSGi-Framework. 44, 50, 51

Programmierschnittstelle

Eine Programmierschnittstelle (englisch application programming interface, API; deutsch Schnittstelle zur Anwendungsprogrammierung) ist ein Programmteil, der von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird, plural=API. 27, 54