Bachelorthesis

Erstellung und Evaluation einer GUI-Anwendung zur Steuerung eines humanoiden Roboterkopfes

Johanna Pieper

Gutachter: Prof. Dr. Bernhard Nebel
Betreuer: Dr. Felix Lindner

Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik
Grundlagen der Künstlichen Intelligenz

28. August 2018

Bearbeitungszeit

30. 05. 2018 – 30. 8. 2018

Gutachter

Prof. Dr. Bernhard Nebel

Betreuer

Dr. Felix Lindner

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum	Unterschrift

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Frage, ob eine grafische Anwendung sich besser zur Steuerung eines humanoiden Roboterkopfes eignet, als eine text-basierte Programmiersprache. Um diese Frage zu untersuchen, wurde die grafische Anwendung RocGUI erstellt. Diese wurde als Benutzerschnittstelle für die Programmiersprache Robot-Control-Language (Roc) [1] entwickelt. Durch die visuelle Darstellung wurde ein Interface geschaffen, welches aus der klassischen Programmierumgebung eine für Einsteiger intuitive Desktop-Anwendung schafft. Damit wird eine Anwendergruppe angesprochen, die typische icon-basierte Anwendungen gewohnt ist.
Zur Beantwortung der ursprünglichen Frage wurde eine empirische Studie durchgeführt in der ein direkter Vergleich der Programmiersprache Roc mit der Anwendung RocGUI anstellt wird. Die Ergebnisse der Studie belegen einen deutlich geringeren Zeitaufwand bei der Benutzung der GUI-Anwendung. Die Produktivität kann somit mit der Verwendung der GUI-Anwendung deutlich angehoben werden. Bei der Untersuchung der Arbeitsbelastung konnte hingegen keine Entlastung festgestellt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Roboter sind seit Langem fester Bestandteil von großen Unternehmen. Mit zunehmender Tendenz ziehen sie jedoch auch in den Alltag der Menschen ein. Roboter zum Rasenmähen oder auch zum Putzen sind heutzutage im häuslichen Umfeld keine Besonderheit mehr. Aber auch vielseitigere Roboter, welche in den unterschiedlichsten Anwendungssituationen zum Einsatz kommen, werden immer erschwinglicher. Daraus entsteht das Bedürfnis einen Roboter individuell und einfach anpassen und steuern zu können. Oftmals wird die Benutzergruppe durch Voraussetzungen wie Programmiererfahrung oder Kenntnisse über die Hardware des Roboters eingeschränkt.
Diese Arbeit beschäftigt sich mit der Erstellung und Evaluation einer Anwendung mit grafischer Benutzungsoberfläche, welche eine einfache Bedienung eines menschlichen Roboterkopfes ermöglicht. Als Ausgangspunkt dient der Roboter Immanuel [2] der technischen Fakultät der Universität Freiburg. Er ist Teil des HERA Projekts [3] der Arbeitsgruppe Grundlagen der Künstlichen Intelligenz , welche sich mit der Formalisierung von Ethiken und ihrem Einsatz für Companion-Roboter beschäftigt. Als solcher wird er in Studien eingesetzt um beispielsweise mit Probanden über moralische Dilemmata zu sprechen. Die Versuchsleiter kommen vornehmlich aus Fachgebieten wie der Kognitionswissenschaft oder Psychologie. Voraussetzungen wie Programmierfähigkeiten oder tiefergehende Computerkenntnisse wie die Benutzung einer Linux-Kommandozeile werden nicht immer mitgebracht.
Momentan wird die Programmiersprache Robot-Control-Language (Roc) [1] verwendet. Ein Hauptproblem der Programmiersprache besteht in der Unübersichtlichkeit, da jedem einzelnen Motor in einer eigenen Datei Anweisungen gegeben werden müssen. Zusätzlich kann die Verwendung einer Programmiersprache abschreckend auf Laien wirken. Aus dieser Situation hat sich die Fragestellung entwickelt, ob eine GUI-Anwendung die Steuerung erleichtern würde.
Zur Beantwortung dieser Frage wird in der folgenden Arbeit der Entwicklungs- und Evaluationsprozess einer GUI-Anwendung beschrieben. Ziel ist eine Benutzungsoberfläche zu kreieren, die als Komplettpaket alle Aspekte der Steuerung zusammenfasst und auch Laien einen einfachen Zugang bietet. Die Programmiersprache Roc wird als

Schnittstelle zwischen der GUI und der vorhandenen Roboterplattform verwendet. Zur Evaluation des Ergebnisses wird eine Studie durchgeführt, welche einen direkten Vergleich mit der Programmiersprache Roc anstellt.

Aufbau der Arbeit Um einen Überblick über die Möglichkeiten der einsteigerfreundlichen Robotersteuerung zu geben, wird in Kapitel 2 eine Zusammenfassung der unterschiedlichen Ansätze gegeben. Im nächsten Schritt wird in Kapitel 3 die zu Beginn vorgefundene Situation beschrieben, welche die vorhandene Roboterplattform InMoov und Steuerungsmöglichkeit in Form der Programmiersprache Roc einschließt. Im weiteren Verlauf wird in Kapitel 4 die Erstellung der GUI-Anwendung beschrieben. Die Anwendung baut direkt auf der Programmiersprache Roc auf und erweitert diese durch eine graphische Repräsentation der Roboterbewegungen. Zur Evaluation der erstellten Anwendung wurde eine Studie durchgeführt, welche einen direkten Vergleich mit der Programmiersprache Roc implementiert. Hierzu wurde in Kapitel 5 die Studie in allen ihren Einzelteilen dokumentiert. Als Haupterkenntnis konnte gezeigt werden, dass bei der Benutzung der GUI-Anwendung deutlich weniger Zeit benötigt wird.

2. Einsteigerfreundliche Steuerung von Robotern

Roboter können auf vielfältige Weise gesteuert werden. Oftmals beschränkt sich die Nutzergruppe allerdings auf erfahrene Nutzer, welche beispielsweise bereits über Programmiererfahrung oder Wissen über die betreffende Roboterplattform verfügen. Es wurden eine Vielzahl von Anwendungen entworfen um den Prozess zu vereinfachen und somit eine breitere Anwendergruppe anzusprechen. Die verschiedenen Ansätze können in die zwei Kategorien, manuelle und automatische Programmiersysteme, unterteilt werden [4, 5]. Im Folgenden wird ein Überblick der unterschiedlichen Ansätze mit Hilfe von passenden Beispielen aus der Fachliteratur gegeben.

Automatische Programmiersysteme Bei automatischen Programmiersystemen wird die eigentliche Programmierung des Roboters vor dem Nutzer versteckt. Der Code zur Steuerung des Roboters wird aufgrund von Benutzereingaben automatisch durch das Interface generiert.
Programmierung-durch-Demonstration ist ein sehr etabliertes Beispiel für ein solches Vorgehen. Der Grundgedanke ist, dass ein Nutzer durch Beispiele einem Roboter bestimmte Verhaltensweisen beibringen kann. Im Allgemeinen werden die Komponenten des Roboter dabei von einem Nutzer sequenziell in unterschiedliche Positionen bewegt. Im Idealfall kann der Roboter das Verhalten nach einem oder mehreren Demonstrationsdurchgängen reproduzieren und auf ähnliche Situationen anwenden. Unter anderen wurde dieser Ansatz von Orendt, Fichtner und Henrich [6] untersucht. In ihrer Studie wurde ein Roboterarm zur Umsortierung und Stapelung von Bauklötzen benutzt. Dabei wurde speziell die einmalige Demonstration betrachtet und mit einem ähnlichen Ansatz von Weiss und Kollegen [7] mit mehreren Demonstrationsdurchgängen verglichen. Als Hauptaspekte wurden die Intuitivität der Bedienung und Robustheit der Reproduktionen des Roboters untersucht. Insgesamt wurde die Programmierung-durch-Demonstration-Methode von beiden Untersuchungen als eine sehr intuitives und leicht zugängliches Vorgehen bewertet. Die einmalige Demonstra-

tion des Verhaltens wurde dabei von Orendt, Fichtner und Henrich als besonders einsteigerfreundlich eingestuft, da ein geringer Aufwand zu einer Kernvoraussetzung einer einsteigerfreundlichen Steuerung zähle. Als Nachteil des Verfahrens wurden die häufigen fehlerbehafteten Reproduktionen des Roboters genannt. Durch die mehrfache Demonstration, wie in der Forschungsarbeit von Weiss beschrieben, kann der prozentuale Anteil der fehlerhaften Reproduktionen deutlich gesenkt werden.
Die Anwendung English2NAO von Buchina, Kamel und Barakova [8] verwendet natürliche Sprache zur Steuerung eines Roboters. Die Grundidee ist, dass Nutzer in normaler Umgangssprache dem Roboter Anweisungen erteilen können. In einem ersten Schritt wird die freie Formulierung eines Nutzers in eine formale Sprache umgewandelt, welche graphisch mit Hilfe von Ikons dargestellt wird. Im weiteren Verlauf kann der Nutzer diese graphische Repräsentation bearbeiten und weiter verfeinern. Das Vorgehen bietet eine sehr intuitive Handhabung und es können sehr schnell und mit geringem Aufwand einfache Bewegungsabläufe entworfen werden. Nachteilig ist hingegen, dass die Roboteraktionen auf eine Reihe vorgefertigter Verhaltensweisen, wie Nicken oder Tanzen beschränkt sind. Die exakte Kontrolle des Roboters ist durch die Ungenauigkeit der natürlichen Sprache nicht möglich. Auch ist die Implementierung von English2Nao hinsichtlich der Interpretation der Sprache noch nicht ausgereift. Das Problem ist, dass Formulierungen ein Stück weit wie ein gewöhnlicher Programmcode strukturiert werden müssen, um ein richtiges Ergebnis zu garantieren. Insgesamt wurde die Anwendung English2Nao von den Autoren als sehr effektives und intuitives Werkzeug zur Erstellung von linearen und kurzen Bewegungsabläufen eingestuft. Gerade Alltagssituationen in denen eine schnelle aber unpräzise Steuerung gefragt ist, könnten von der Sprachsteuerung profitieren. Zur Erstellung von komplexen und präzisen Bewegungen wurde das Konzept als zu unausgereift eingestuft.

Manuelle Programmiersysteme Bei der manuellen Programmierung von Robotersystemen erhält der Nutzer direkte Kontrolle über die Programmieranweisungen. Hierzu zählen die typischen text-basierten Roboterprogrammiersprachen wie RobotC, Kuka Robot Language oder Rapid, aber auch die gängigen höheren Programmiersprachen wie Python oder C++.
Bei visuellen Programmiersprachen wird zusätzlich mit graphischen Elementen und ihrer Anordnung gearbeitet. In großen Teilen der Fachliteratur wird die Meinung vertreten, dass die visuelle Darstellung besser geeignet ist für den Einstieg in die

Programmierung (z.B in [9, 10, 11]). Die Darstellung von Programmabläufen durch Ikons oder Flussdiagrammen wurde von Carlisle [9] als besonders gutes Mittel hervorgehoben. Saito und Kollegen [10] zeigten, dass die visuelle Darstellung motivierend auf Anfänger wirken kann, sodass ein erhöhtes Interesse an der Programmierung hervorgerufen wird. Schiffer [12] vertritt die These, dass bildliche Darstellungen einen anschaulichen Realitätsbezug herstellen können.
Eine sehr bekannte visuelle Programmiermethode kann unter dem Begriff Block-Programmierung zusammengefasst werden. Einzelne Codeteile wie Aktionen und Kontrollstrukturen werden wie ein Puzzle zusammengesetzt, wobei nur sinnvolle Programmteile zusammenpassen. Letztendlich entsteht ein kompletter Programmcode, der auch als solcher gelesen werden kann. Das Konzept wird beispielsweise in der erfolgreichen Programmierlernumgebung Scratch [13] eingesetzt. Selbst für Kindern in einem Alter von ca. 10 Jahren wurde die Benutzung der Plattform als einfach eingestuft [14]. In einer Studie von Maloney und Kollegen [15] wurde der Lernfortschritt von Programmierlaien im Alter von 8 bis 18 Jahren über einen Zeitraum von zwei Jahren dokumentiert. Die Teilnehmer erhielten keine Hilfestellung, wie eine Einführung in Programmierkonzepte oder der Benutzung von Scratch. Trotzdem wurden Kernvorstellungen der Programmierung, wie Schleifen oder Bedingungen von den Teilnehmern in Eigenleistung entwickelt. Insgesamt wurde hier die Intuitivität und Effektivität der Anwendung aufgezeigt. In anderen Anwendungen, wie Code3 [16] oder CoBlox [4] wurde das Prinzip auf die Roboterprogrammierung übertragen. Die Untersuchungen zu den beide Anwendungen haben gezeigt, dass das Konzept auch für Erwachsene geeignet ist und gut in einem industriellen Kontext eingesetzt werden kann.
Eine andere Möglichkeit einen Programmablauf zu strukturieren ist die Nutzung von Flussdiagrammen. Ein Beispiel hierfür ist die von Lego bereitgestellte Software zur Steuerung von EV3 Robotern [17]. Im Gegensatz zu der textbasierten Darstellung der Block-Programmierung wird hier auf stark symbolische Darstellungen gesetzt. Es existieren unterschiedliche Typen von Blöcken, wie Programmablaufs-, Aktions-, Sensor- oder Motorblöcke. Diese können wiederum zusammengefügt werden wodurch eine Art Ablaufdiagramm entsteht. Durch zusätzliche Verbindungslinien zwischen den Blöcken können beispielsweise Schleifen implementiert oder Daten ausgetauscht werden. Das Programm wurde vor allem als Lern- und Spielumgebung für Kinder entworfen. Die graphische Darstellung macht einen Einstieg einfach, jedoch sind der Komplexität des Programmcodes durch die einfache Darstellung auch Grenzen gesetzt.

Viele Anwendungen kombinieren unterschiedliche Methoden miteinander, um in jeder erdenklichen Situation eine gute Lösungsmöglichkeit zu bieten. Die Anwendung Choreograph [18] wurde von der Firma Aldebaran-Robotic entworfen um den firmeneigenen, humanoiden Roboter Nao [19] zu steuern. Der Roboter besitzt 25 Freiheitsgrade, wodurch die Steuerung eine hohe Komplexität erreicht. Choreograph besteht aus mehreren Teilmodulen, unter anderen können einzelne Verhaltensweisen in einem Flussdiagramm zu einem komplexen Verhaltensmuster kombiniert werden. Eine andere Herangehensweise wird durch eine Zeitleiste implementiert. Dabei werden Bewegungen durch einen genauen zeitlichen Ablauf definiert. Der Zustand des Roboters wird zu jedem Zeitpunkt durch die Positionen der Motoren bestimmt, welche ebenfalls in der Zeitleiste einsehbar sind. Die beiden oben beschriebenen graphischen Darstellungen werden durch eine text-basierte Darstellung vervollständigt. So ist es ebenfalls möglich dem Roboter über ein Python-Skript Anweisungen zu geben. Die Kombination dieser verschiedenen Modi soll es Nutzern erlauben eine an die jeweilige Zielsetzung angepasste Vorgehensweise zu wählen.
Tangible-User-Interfaces (kurz TUI) sind Benutzerschnittstellen, die eine Interaktion mit einem Roboter über physische Objekte erlauben. Horn und Jacob [20] untersuchten zwei unterschiedliche „greifbare“ Programmiersprachen. Ähnlich wie in den vorherigen Abschnitten können einzelne Blöcke zu einem Programmablauf zusammengesetzt werden. Die von Sefidgar und Kollegen vorgestellte Situated-Tangible-Programmierung [21] implementiert ein Vorgehen, in dem unterschiedliche Blöcke in dem Arbeitsraum eines Roboters platziert werden. Dabei können Blöcke zur Deklaration von Objekten und Orten, wie auch Anweisungsblöcke verwendet werden. Insgesamt kann gesagt werden, dass Tangible-User-Interfaces vor allem für Kinder und Programmieranfänger vorteilhaft sind, da teilweise grundlegende Computererfahrungen fehlen und der Umgang mit physischen Objekten leichter fällt [22].

Die vorliegende Arbeit nutzt eine graphische Repräsentation um die Bewegungen des Roboters darzustellen. Ähnlich wie eine Komponente der Choreograph Anwendung wird eine Zeitleiste zur chronologischen Anordnung der Bewegungen des Roboters benutzt. Es handelt sich dabei in beiden Fällen um eine rein sequentielle Abfolge von Bewegungen, ohne jegliche Verzweigungsmöglichkeiten wie Schleifen oder if-Abfragen (Choreograph implementiert diese Kontrollstrukturen in anderen Bereichen der Anwendung).
Die Anwendung RocGUI wurde als Aufsatz auf der Programmiersprache Roc konzipiert. Daher ist jederzeit ein Darstellungswechsel zwischen der text-basierten und der

graphischen Repräsentation möglich. Mit einem vergleichbaren Darstellungswechsel wurde auch bei den Anwendung Choreograph gearbeitet. Wie auch die Anwendung Choreograph verspricht sich diese Arbeit eine bessere situationsbezogene Wahlmöglichkeit für die Nutzer. Beispielsweise könnte durch einen Darstellungswechsel das Debuggen von fehlerhaftem Verhalten erleichtert werden.

3. Programmierung des Roboters Immanuel

Um die genauen Voraussetzungen und Möglichkeiten der in dieser Arbeit entwickelten Anwendung zu verstehen, wird auf die vorgefundene Ausgangssituation eingegangen. Hierzu wird der verwendete Roboter Immanuel, welcher auf der Roboterplattform InMoov [23] beruht, beschrieben und die zur Steuerung verwendete Robot-Control-Language (Roc) [1].

3.1. Roboterplattform InMoov

Zur Demonstration der Anwendung RocGUI wird die Roboterplattform InMoov [23] verwendet. Es handelt sich dabei um ein Open-Source-Projekt, welches 2012 von Gaël Langevin einem französischen Designer und Bildhauer veröffentlicht wurde. Für den Zweck dieser Arbeit wird nur der Kopf und Nackenbereich des Roboters genutzt. Das Projekt verfügt über weitere Teile, wie den Oberkörper, Arme und Beine. Alle Komponenten außer den Beinen sind dabei motorisiert und können durch Servomotoren bewegt werden. Zur Steuerung der Motoren werden Mikrocontroller und ein Arduino Board Uno empfohlen. Alle nicht elektronischen Bauteile sind als Vorlagen im Internet verfügbar und können durch einen 3D-Drucker reproduziert werden. Die für diese Arbeit relevanten, beweglichen Komponenten werden im Folgenden näher beschrieben:

Augen Die Augen besitzen zwei Freiheitsgrade. Diese werden durch zwei unterschiedliche Servomotoren implementiert. Ein Motor sorgt dabei für die Links- und Rechts-, ein andere für die Auf- und Abbewegung der Augen. Beide Motoren agieren dabei unabhängig voneinander, wobei allerdings beide Augen immer synchron bewegt werden.

Kiefer Mundbewegungen können durch die Auf- und Abbewegung des Kiefers simuliert werden. Hierbei steht nur ein Freiheitsgrad zur Verfügung. Es wird ein

Servomotor verwendet um das komplette Bauteil unterhalb des Mundes zu bewegen.

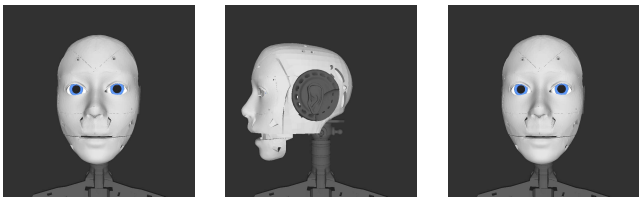
Kopf Der Kopf besitzt genau wie die Augen zwei Freiheitsgrade. Er lässt sich mit Hilfe von zwei unterschiedlichen Servomotoren nach links und rechts, wie auch nach oben und unten drehen.

3.2. Robot-Control-Language (Roc)

Die Programmiersprache Robot-Control-Language (Roc) [1] wurde ursprünglich im Zuge einer Bachelorarbeit von Dobrusin entwickelt. In den vergangenen Jahren wurde sie allerdings durch Dr. Lindner, einem Mitarbeiter der Arbeitsgruppe Grundlagen der Künstlichen Intelligenz der Universität Freiburg, modifiziert und an spezielle Anforderungen angepasst. Hauptgrund für die Änderungen war, dass keine uneingeschränkten parallelen Bewegungen der Motoren möglich war. Im Folgenden wird die derzeit verwendete Form der Programmiersprache beschrieben.
Roc wurde entwickelt um eine einfache Schnittstelle zwischen Nutzer und Roboter zu schaffen. Die Sprache ist angelehnt an die englische Sprache und dient zur Formulierung einfacher Befehle, welche interpretiert und durch einen Roboterkopf ausgeführt werden. Beispielsweise kann durch den strukturierten Befehl „severe left for 1000 milliseconds“ der Kopf des Roboters stark nach links ausgelenkt werden. Dabei muss immer klar sein, welcher Motor angesprochen werden soll, da es ansonsten zu Uneindeutigkeiten kommen kann.
Der Nutzer muss lediglich die Syntax der Programmiersprache erlernen. Die direkte Kommunikation mit dem Roboter übernimmt dabei das dahinterstehende System. Die Abfolge der Interpretation eines Roc-Befehls kann in mehrere Schritte aufgeteilt werden. Im ersten Schritt erzeugt der Roc-Compiler JSON-Dateien welche die Informationen der geforderten Bewegungen durch das Facial-Action-Coding-System [24] codiert. Im nächsten Schritt werden diese Dateien durch eine Software interpretiert und an das jeweilige Robotersystem weitergeleitet. Zur Anwendung der Programmiersprache auf andere Robotersysteme muss daher lediglich der zweite Schritt angepasst werden.

Syntax Jeder Roc-Befehl benutzt spezielle Schlüsselbegriffe, um die notwendigen Informationen zu kodieren. Um einen Bewegungsablauf des Roboters zu generieren, wird für jede Teilbewegung die Endposition und Zeitdauer der Bewegung benötigt. Die Endposition ergibt sich dabei aus der Richtung der Bewegung und der Aus-

File: head_leftright.roc
neutral position for 1500 milliseconds
maximum right for 500 milliseconds
maximum right for 500 milliseconds
neutral position for 1000 milliseconds



Zeitpunkt: 1,5 s | Zeitspanne: 2 - 2,5 s | Zeitpunkt: 3,5 s

Abbildung 3.1.: Roc-Befehle mit zugehörigen Momentaufnahmen des Bewegungsablaufs des InMoov-Roboterkopfes. Bildquelle: Dobrusin [1]

prägungsstärke. Die Datei, in dem sich der jeweilige Befehl befindet, definiert die Zugehörigkeit zu einem Motor des Roboters. Ein Beispiel für eine Befehlsabfolge mit daraus resultierenden Bewegungen des InMoov-Roboterkopfes ist in Abbildung 3.1 zu sehen.
Informell lässt sich die Syntax eines Standardbefehls durch den Ausdruck „⟨ Intensity ⟩ ⟨ Direction ⟩ for ⟨ Duration ⟩ milliseconds“ beschreiben. Eine Ausnahme existiert für die Positionierung in eine neutrale Haltung. Dazu wird der Ausdruck „neutral position for ⟨ Duration⟩ milliseconds“ verwendet. Die Wörter in den spitzen Klammern sind Platzhalter, für welche eine Reihe von festgelegten Wörtern eingesetzt werden können. ⟨ Intensity ⟩ kann durch die Wörter „trace of“, „slightly“, „pronounced“, „severe“ und „maximum“ ersetzt werden. Sie beschreiben die Ausprägungsstärke einer Bewegung. „trace of“ steht dabei für die geringste und „maximum“ für die größte Auslenkung. Durch ⟨ Direction ⟩, mit den möglichen Werten „left“, „right“, „up“ und „down“ wird die Richtung der Bewegung bestimmt. Mit einer bestimmten Kombination von Werten für ⟨ Direction ⟩, ⟨ Intensity ⟩ und dem Körperteil, welches bewegt werden soll, lässt sich eine genaue Endposition des betreffenden Körperteils berechnen. Mit ⟨ Duration ⟩ wird der genaue zeitliche Ablauf der Bewegung definiert. Der Wert, welcher eingesetzt wird, steht für die Anzahl der Millisekunden, die zur Verfügung stehen um in die entsprechende Endposition zu gelangen.
Die formelle Beschreibung der Sprache folgt der Definition in der Bachelorarbeit von Dobrusin [1]. Es wurden lediglich Anpassungen vorgenommen, um die derzeitig

Beschreibung	Symbol
Definition	::=
Alternative	|
Terminalsymbol	'...'
Nichtterminalsymbol	<...>
Gruppierung	(...)
Wiederholung (mindestens einmal)	(...)+

Tabelle 3.1.: Auflistung der relevanten Symbole der erweiterten Backus-Naur-Form

$\langle MotionSequence \rangle$	::=	$\langle Intensity \rangle$ $\langle Direction \rangle$ 'for' $\langle Duration \rangle$ 'milliseconds'
	|	'neutral position for' $\langle Duration \rangle$ 'milliseconds'
$\langle Direction \rangle$	::=	'left' | 'right' | 'up' | 'down'
$\langle Intensity \rangle$	::=	'trace of' | 'slightly' | 'pronounced' | 'severe' | 'maximum'
$\langle Duration \rangle$	::=	$n \in \mathbb{Z}_{>0}$

Abbildung 3.2.: Definition der Programmiersprache Roc in der erweiterten Backus-Naur-Form

verwendete Form der Sprache darzustellen. Die erweiterte Backus-Naur-Form (EBNF) ist ein Werkzeug zur präzisen Beschreibung von kontextfreien Grammatiken. Hierzu zählen auch die heutzutage gängigen höheren Programmiersprachen. Tabelle 3.1 zeigt die für diese Arbeit benötigten Symbole der erweiterten Backus-Naur-Form. Die Definition der Roc-Syntax in der erweiterten Backus-Naur-Form ist in Abbildung 3.2 zu sehen.
In der Praxis existieren einige Einschränkungen, die nicht direkt aus der Definition folgen. Um alle Motoren unabhängig von einander bewegen zu können, wird für jeden Motor eine eigene Roc-Datei benötigt. Da ein Motor jeweils nur einen Freiheitsgrad besitzt muss der jeweilige Wertebereich von ⟨ Direction ⟩ kontextbezogen auf „left“ und „right“ oder „up“ und „down“ eingeschränkt werden.

Facial-Action-Coding-System (FACS) FACS [24] ist ein Werkzeug zur Messung und Beschreibung von Gesichtsausdrücken. Einzelne oder mehrere Muskelbewegungen werden zu einer sogenannten Bewegungseinheiten (englisch: Action Unit (AU)) zusammengefasst. Die Mimik eines Menschen kann mit Hilfe der Bewegungseinheiten beschrieben werden. Beispielsweise wird das Zusammenziehen der Augenbrauen durch die Bewegungseinheit 4 kodiert. Zusätzlich existieren fünf unterschiedliche

Bewegungs--einheit	**Beschreibung**	**Ausprägungs-stärke**	**Beschreibung**
0	neutrales Gesicht	A	angedeutet
51	Linksdrehung des Kopfs	B	leicht
52	Rechtsdrehung des Kopfs	C	ausgeprägt
53	Aufwärtsdrehung des Kopfs	D	stark
54	Abwärtsdrehung des Kopfs	E	maximal
61	Linksdrehung der Augen		
62	Rechtsdrehung der Augen		
63	Aufwärtsdrehung der Augen		
64	Abwärtsdrehung der Augen		

Tabelle 3.2.: Übersicht der relevanten Bewegungseinheiten und Intensitäten des Facial-Action-Coding-Systems [24] zur Definition der Programmiersprache Roc. Freie Übersetzung der Originalversion.

Ausprägungsstärken (siehe Tabelle 3.2) der Bewegungen. Mit dem Ausdruck „AU 4D“ wird daher ein starkes Zusammenziehen der Augenbrauen beschrieben. Eine geschulte Person kann durch die Verwendung dieser Notation eine genaue Beschreibung der Miene eines Menschen anfertigen. Ziel ist oftmals eine objektive Einschätzung der Gefühle einer Person. Das System wurde 1978 von Paul Ekman und Wallace Friesen erstmals veröffentlicht.

Bei der Kompilierung von Roc-Befehlen wird, wie von Dobrusin [1] beschrieben, in einem ersten Schritt jedem Befehl eine Bewegungseinheit mit zusätzlicher Ausprägungsstärke zugeordnet. Die hierzu relevanten Bewegungseinheiten können aus Tabelle 3.2 entnommen werden. Beispielweise kann eine Bewegung der Augen von der maximalen Rechtsposition in die maximale Linksposition durch „AU 62E“ als Ausgangspunkt und „AU 61E“ als Endpunkt kodiert werden. Wenn man die zeitlichen Angaben der Bewegung hinzunimmt ergeben sich alle notwendigen Informationen um einen Bewegungsablauf des Roboters zu generieren. Abgespeichert werden diese Informationen in einer JSON-Datei. Abbildung 3.3 zeigt ein Beispiel für Roc-Dateien mit zugehörigen JSON-Output-Dateien.

File: head_leftright.roc

```
trace of left for 1500 milliseconds
maximum right for 2000 milliseconds
```

File: eyes_updown.roc

```
maximum up for 1000 milliseconds
maximum up for 2500 milliseconds
neutral position for 1000 milliseconds
```

File: head_leftright.json

```
[
    {
        "action_unit ": "51",
        "duration": 1500,
        "intensity": "A"
    },
    {
        "action_unit": "52",
        "duration": 2000,
        "intensity": "E"
    }
]
```

File: eyes_updown.json

```
[
    {
        "action_unit": "63",
        "duration": 1000,
        "intensity": "E"
    },
    {
        "action_unit": "63",
        "duration": 2500,
        "intensity": "E"
    },
    {
        "action_unit": "63",
        "duration": 1000,
        "intensity": "A"
    }
]
```

Abbildung 3.3.: Roc-Dateien im oberen Bereich mit zugehörigen JSON-Output-Dateien im unteren Bereich

4. Entwurf der GUI-Anwendung

4.1. Anforderungsanalyse

4.1.1. Benutzerbefragung

Am Anfang der Anforderungsanalyse stand eine Befragung von zwei Nutzern der Programmiersprache Roc. Als Hauptkritikpunkt wurde die fehlende Übersicht genannt, da jedem Motor in einer eigenen Datei Anweisung gegeben werden muss. Um Bewegungen von Teilkomponenten aufeinander abzustimmen, müssen die zeitlichen Informationen der Roc-Befehle aufsummiert werden. Eine der befragten Personen benutzte sogar ein externes Skript um sich die Rechenarbeit zu ersparen. Dasselbe Problem besteht bei der Synchronisation von Sprache und Bewegung des Roboters. Die Audiodatei und Motorbefehle für die Bewegung des Mundes wird mit dem externen Programm MaryTTS [25] erstellt. Ein gewöhnlicher Audioplayer wird verwendet, um die benötigten Zeiten zu ermitteln.
Als weitere Hürde wurde die häufige Benutzung der Linux-Kommandozeile genannt. Zur Kompilierung und Ausführung der Roc-Befehle, wie auch zur Benutzung der MaryTTS Software werden eine Reihe von Kommandozeilenbefehle benötigt. Grundsätzlich ist dies keine große Schwierigkeit nach einer Einführung und Einarbeitungsphase, jedoch kann es auf einen Laien eine abschreckende Wirkung haben. Von den Befragten wurde ein konkretes Beispiel genannt wo dies der Fall war und dadurch keine Zusammenarbeit mit dem Roboter zustande kam.

4.1.2. Benutzerprofil

Da Immanuel vor allem als moralische Instanz zum Einsatz kommt, wird er oftmals von Studierenden der Fachrichtungen Kognitionswissenschaften und Psychologie benutzt. Daher sind Erfahrungen mit Programmierung und Robotern in vielen Fällen nicht vorhanden. Die Steuerung des Roboter muss daher auch für komplette Anfänger möglich sein. Zusätzlich darf die Gruppe der erfahrenen Nutzer nicht

ausgeschlossen werden. Insofern sollte auch für diese Nutzer ein Mehrwert durch die Benutzungsoberfläche existieren.

4.1.3. Anforderungskatalog

Aus den oben aufgezählten Problemen, der Zielgruppe und der zu steuernden Robotorplattform ergeben sich die Anforderungen an die Anwendung.

Funktionalität Grundsätzlich ist das Ziel alle Funktionen der Programmiersprache Roc in die GUI-Anwendung zu übernehmen. Dazu zählt beispielsweise, dass alle Motoren sich vollkommen unabhängig voneinander bewegen können. Auch die Intensitätsstufen der einzelnen Bewegungen werden übernommen. Zur Kompilierung in eine für den Roboter verständliche Form soll in einem ersten Schritt die graphische Darstellung der Anwendung in Roc-Befehle überführt werden. Gleichzeitig soll das Einlesen von Roc-Befehlen in die graphische Anwendung möglich sein, wodurch sich der Nutzer jederzeit zwischen den beiden Darstellungsformen entscheiden kann.
Die Anwendung soll die bisherigen Einzelteile, wie die MaryTTS Software, das externe Kompilieren und Ausführen des Roboters zu einem ganzen und vollständigen System zusammenfassen. Bisher waren sehr viele Schritte außerhalb des Roc-Editors notwendig, um letztendlich den fertigen Bewegungsablauf des Roboters sehen zu können. Dies soll in der GUI-Anwendung im Hintergrund und ohne Wissen des Nutzers passieren.

Kompatibilität Um die Langlebigkeit der Software zu garantieren muss die Kompatibilität mit unterschiedlichen Computersystemen und Roboterplattformen garantiert werden. Es muss zur Entwicklung der Anwendung eine Programmiersprache oder Entwicklungsumgebung gewählt werden, welche alle gängigen Betriebssysteme abdeckt. Um andere Roboterplattformen steuern zu können muss die Anwendung anpassbar sein. Hierzu soll es möglich sein, die vorhandenen Informationen der Motoren zu konfigurieren und beliebig viele weitere Motoren einzubinden. An dieser Stelle muss gesagt werden, dass eine solche Anpassung nur durch die Weiterentwicklung der Programmiersprache Roc möglich ist.

Zuverlässigkeit Zu den wichtigsten Eigenschaften einer guten Anwendung gehört die Robustheit. Daher ist ein Ziel der Entwicklung eine Anwendung zu erstellen

welche tolerant gegenüber fehlerhaften Benutzereingaben oder Fehlverhalten von externer Software ist. Systemabstürze, fehlerhaftes Abspeichern oder Einlesen der Roc-Dateien dürfen nicht vorkommen.

Benutzbarkeit Jedem Nutzer soll eine einfache Steuerung des Roboters ermöglicht werden. Die gesamte Oberfläche muss verständlich und einfach aufgebaut sein, um eine intuitive Bedienung zu gewährleisten und einen langen Einarbeitungsprozess zu vermeiden. Der Zweck eines graphischen Elements muss durch die Gestaltung und Platzierung sichtbar werden. Zusätzlich soll die Bedienung der Anwendung möglichst komfortabel sein und eine effiziente Entwicklung eines Bewegungsablaufs ermöglichen.

4.2. Entwicklungsumgebung

Zur Entwicklung der Anwendung RocGUI wurde die Programmiersprache Livecode verwendet. Sie wurde von der schottischen Firma Livecode Ltd. mit Sitz in Edinburgh entwickelt. Livecode ist ebenfalls die Bezeichung der integrierte Entwicklungsumgebung, welche die Erstellung und das Testen einer Anwendung erlaubt. Es existiert eine kommerzielle wie auch voll funktionsfähige kostenlose Version der Software. Ein Hauptgrund für die Verwendung von Livecode war die Plattformunabhängigkeit, da Standardbetriebssysteme wie Windows, Linux und MacOs unterstützt werden. Außerdem ist ein grafischer GUI-Editor integriert, welcher eine sehr einfache Erstellung einer Benutzungsoberfläche erlaubt. Der Programmierstil der Sprache ist an das objektoriertierte Programmierparadigma angelehnt und wird im Folgenden beschrieben.

Grundlegende Funktionsweise Die Programmiersprache basiert auf den zwei Grundbausteinen, welche am besten durch die Begriffe Objekte und Nachrichten beschrieben werden. Nachrichten werden durch Ereignisse wie Benutzeraktivitäten oder Programmvorgänge ausgelöst. Objekte können auf Nachrichten reagieren, falls in dem Objekt ein entsprechender Programmcode hinterlegt wurde. Eine Nachricht wird immer an das naheliegendste Objekt gesendet. Falls dieses Objekt keinen Code zur Behandlung der Nachricht enthält, wird die Nachricht an das Objekt weitergereicht, welches in der Hierarchie eine Stufe über dem Ursprungsobjekt steht. Dieses Vorgehen implementiert eine Art Vererbung wie man sie von den typischen objektorientierten Programmiersprachen kennt. Abbildung 4.1 zeigt eine Übersicht über die hierar-

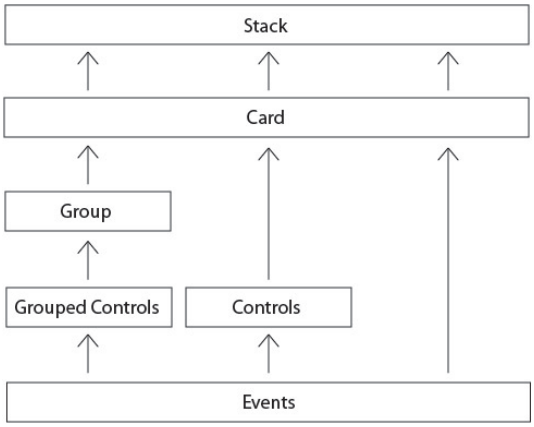


Abbildung 4.1.: Hierarchische Anordnung der Livecode-Objekte.
Quelle: in Anlehnung an [26]

chische Anordnung der Objekte. Als Hauptobjekt kann ein sogenannter „Stapel“ (engl. Stack) gesehen werden. Dieser kann wiederum beliebig viele „Karten“ (engl. Cards) enthalten, welche die typischen Programmfenster darstellen. Auf dieser Ebene werden die eigentlichen Bedienelemente (engl. Controls) wie beispielsweise Buttons, Textfelder und Scrollleisten platziert. Diese können entweder für sich alleine stehen oder durch eine Gruppe zusammengefasst werden. Diese Anordnung definiert den Weg einer Nachricht bis sie von einem Objekt abgefangenen wird. Wird beispielsweise ein Button gedrückt, so sucht Livecode erst einmal in dem Programmcode des Buttons nach einer entsprechenden Reaktion. Erst nachdem er hier nichts gefunden hat, durchsucht er nacheinander die entsprechende Gruppe (falls vorhanden), die Karte und den Stapel.
Das Skript eines Objektes kann Funktionen, wie man sie aus höheren Programmiersprachen kennt oder Reaktionen auf bestimmte Nachrichten enthalten. Außerdem können einem Objekt beliebig viele Attribute zugewiesen werden. Die meisten Objekte besitzen bereits bei ihrer Erstellung bestimmte Attribute, wie Länge, Breite und Position.

4.3. Implementierung

4.3.1. Interface

Abbildung 4.2 zeigt die graphische Benutzungsoberfläche des Hauptprogrammfensters. Der obere Bereich enthält eine Menüleiste mit Einstellungsmöglichkeiten, Kompilierbutton und einen Button zur Ausführung des Roboterkopfs. Der „Compile“ Button

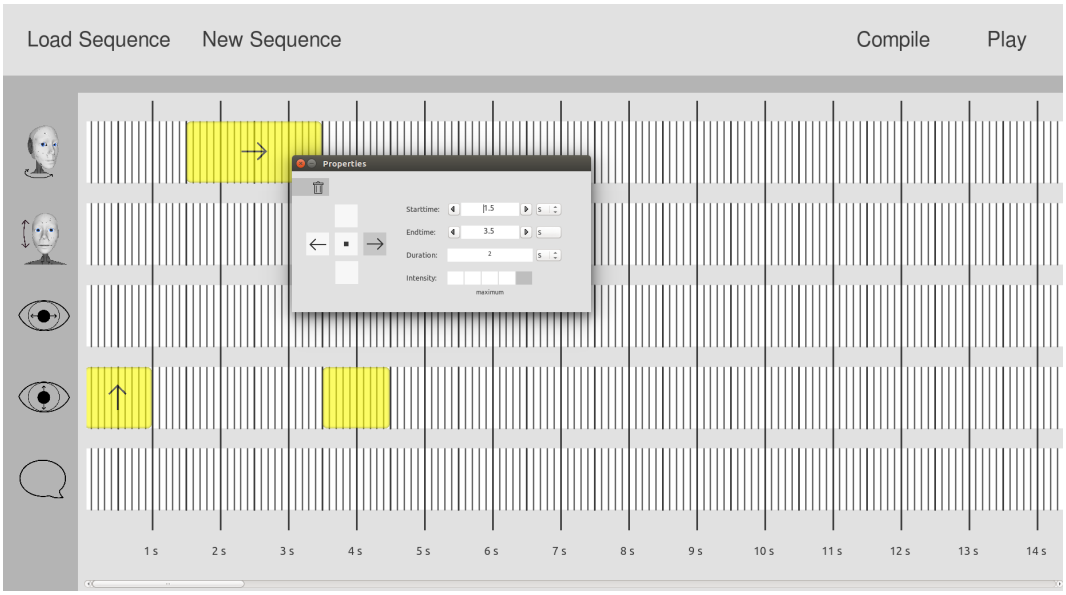


Abbildung 4.2.: Hauptprogrammfenster der Anwendung RocGUI mit Einstellungsfenster zur Festlegung der Eigenschaften einer Bewegung des Roboters. Bildquelle des InMoov-Roboterkopfes: Dobrusin [1]

speichert den erstellten Bewegungsablauf in Form von Roc-Dateien. Durch den „Play“ Button werden die abgespeicherten Dateien an den Roboter weitergegeben und ausgeführt. Durch den Button „New Sequence “ wird ein leeres Projekt angelegt und mit dem Button „Load Sequence“ kann ein früheres Projekt in die graphische Oberfläche geladen werden.
Der eigentliche Inhalt ist im unteren Bereich zu sehen. Dieser ist wiederum in mehrere Zeitleisten untergliedert. Durch die obersten vier Zeitleisten werden die Bewegungen des Roboters definiert. Jede Zeitleiste wird einem bestimmten Motor zugeordnet. Damit wird sichergestellt, dass alle Komponenten des Roboters parallel bewegt werden können. Innerhalb einer Zeitleiste werden Bewegungen durch farbige Rechtecke dargestellt. Diesen Rechtecken werden Eigenschaften, wie eine Richtung, Intensität und Anfangs- wie auch Endzeitpunkt zugeordnet. Der Anfangs- und Endzeitpunkt ist direkt bei der Betrachtung der Zeitleiste erkennbar, da die linke und rechte Kante entsprechend platziert sind. Die Richtung einer Bewegung kann durch die symbolische Beschriftung des entsprechenden Rechtecks abgelesen werden. Um alle Eigenschaften auf einmal sehen bzw. konfigurieren zu können muss mit der rechten Maustaste ein Einstellungsfenster (siehe Abbildung 4.2) geöffnet werden.
Durch die visuelle Darstellung der Richtungen und Zeitpunkte der Bewegungen

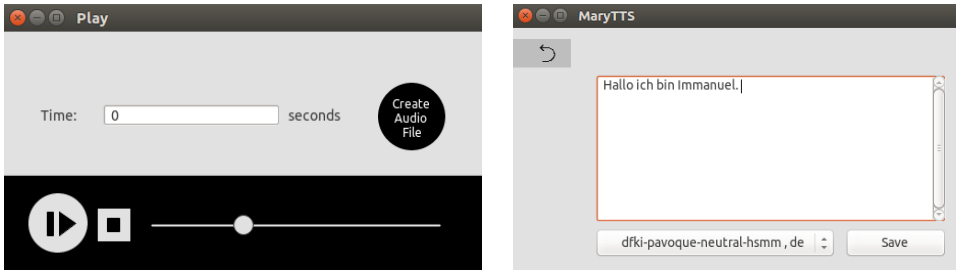


Abbildung 4.3.: Player und MaryTTS Programmfenster der Anwendung RocGUI

wird auf eine bessere Übersicht des kompletten Bewegungsablaufs abgezielt. Schmid gelangt in seiner Dissertation [11] zu dem Schluss, dass ein Vorteil der visuellen Darstellung in der Möglichkeit einer übersichtlicheren Darstellung von bestimmten Strukturen besteht. Hierzu wird der Punkt genannt, dass quantitative Eigenschaften teilweise wirkungsvoll visuell dargestellt werden können. Waren [27] unterstützt diese Ansicht durch die Aussage, dass große Mengen an Daten durch eine visuelle Darstellung komprimiert werden können. Mit der oben geschilderten Visualisierung der Richtungen, wie auch Start- und Endzeitpunkten der Bewegungen wird versucht sich diese Effekte zu nutze zu machen.
Die unterste Zeitleiste ist zur Konfiguration der Sprachausgabe des Roboters zuständig. Durch einen Linksklick wird ein Player (siehe Abbildung 4.3) geöffnet. Hier kann eine vorhandene Audiodatei abgespielt werden, was im Hauptfenster durch eine Markierung der derzeitigen Laufzeit in der untersten Zeitleiste sichtbar wird. Durch den Button „Create New Audio“ öffnet sich das Einstellungsfenster zur Erstellung der Audiosprachdatei (siehe Abbildung 4.3). In diesem Fenster kann der Text und die Stimme des Roboters festgelegt werden. Zur Erstellung der Audiodatei und der Mundbewegung des Roboters wird auf das externe Open-Source Programm MaryTTS [25] zurückgegriffen.

4.3.2. Architektur

Wie in Abschnitt 4.2 beschrieben, arbeitet Livecode mit einer hierarchischen Anordnung von Objekten. Um die Struktur der Anwendung RocGUI zu dokumentieren, wurden die Anordnung der Objekte des Hauptfensters in Abbildung 4.4 dargestellt. Der Hauptstack besitzt nur eine Karte, die mit „MainCard“ benannt wurde. Abbildung 4.2 zeigt die graphische Repräsentation der Karte. Sie enthält die drei Gruppen „timelines“, „menu“ und „layout“. Der „layout“ Gruppe wurden alle Bilder

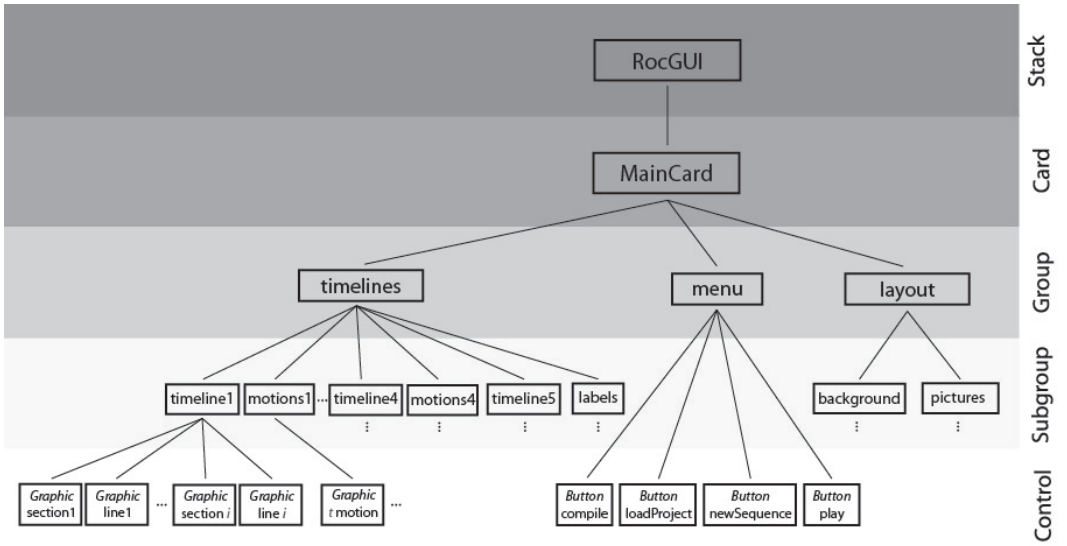


Abbildung 4.4.: Übersicht der Struktur des Hauptstacks. Die Abbildung wurde leicht vereinfacht um die wichtigsten Aspekte hervorzuheben.

und Hintergrundgraphiken zugeordnet. Die „timeline“ Gruppe enthält eine Reihe von Untergruppen um die Elemente der verschiedenen Zeitleisten zu sortieren. Die Untergruppen mit dem Präfix „timeline“ enthalten alle graphischen Elemente zur Darstellung der Zeitleisten. Die Zahlen am Ende deuten auf die graphische Anordnung der Zeitleisten hin. Jeder Zeitleiste mit Ausnahme der Sprachzeitleiste wurde zusätzlich eine Untergruppe mit Präfix „motion“ zugeordnet. Diese speichert die zugehörigen Graphiken um die Bewegungen des Roboters zu symbolisieren. Die verbleibende „labels“ Untergruppe speichert die Zeitleistenbeschriftung. Das Verhalten der Untergruppen wird durch den Programmcode der Gruppe „timelines“ definiert. Die Untergruppen selber enthalten keinen Code. Die Gruppe „menu“ enthält vier weitere Buttons, deren Funktion sich durch den jeweiligen Namen erklärt.
Außer dem Hauptstack wurden weiter zwei erwähnenswerte Substacks implementiert. Der „MotionProperties“-Stack enthält nur eine Karte, welche die Einstellungsmöglichkeiten der Bewegungen des Roboters enthält. Hier wurde jeweils eine Untergruppe für die graphischen Elemente der unterschiedlichen Einstellungsmöglichkeiten geschaffen. In Abbildung 4.2 ist die passende graphische Repräsentation der Karte zusehen.
Der „Voice“-Stack enthält zwei unterschiedliche Karten (siehe auch Abbildung 4.3). Der Programmcode der „MaryTTS “-Karte enthält die Prozeduren zur Bedienung des MaryTTS Programms. Beispielsweise der Start von MaryTTS und die Auflistung der installierten Stimmen. Die „Player“-Karte stellt einen gewöhlichen Audioplayer zu

Abspielung der MaryTTS Audiodateien dar. Beide Karte des „Voice“-Stack enthalten keine weiteren Gruppen, da die Anzahl der graphischen Elemente überschaubar ist.

4.3.3. Kompilierung

Im ersten Schritt wird für jede Zeitleiste (außer der Sprachzeitleiste) eine eigene Roc-Datei erstellt mit den entsprechenden Roc-Befehlen. Im Folgenden wird wie von Dobrusin [1] beschrieben vorgegangen. Jedem Roc-Befehl wird eine Bewegungseinheit und Intensität des Facial-Action-Coding-Systems [24] zugeordnet. Zusammen mit der jeweiligen Zeitdauer der Bewegung werden die Daten in Form einer baumartigen Strukturierung in eine JSON-Datei abgespeichert. Die Zwischenspeicherung dient dem Zweck, die Programmiersprache Roc von möglichen Implementierungsdetails des Zielsystems zu trennen. Zur Kontrolle einer speziellen Robotorplattform muss das jeweilige System die Möglichkeit haben die JSON-Dateien einzulesen und diese entsprechend zu verarbeiten. Weitere Details können in der Bachelorarbeit [1] von Dobrusin nachgelesen werden.
Abbildung 4.5 zeigt die passende Roc-Output-Datei zum Programmfenster in Abbildung 4.2.

4.3.4. Weiterentwicklungsmöglichkeiten

Die Erweiterungsmöglichkeit der Anwendung hängt eng mit der Programmiersprache Roc zusammen. Alle Zusatzfunktionen müssen auch in der Programmiersprache implementiert werden.
Prinzipiell können durch zusätzliche Zeitleisten weitere Freiheitsgrade des Roboters implementiert werden. Jedoch muss an dieser Stelle gesagt werden, dass die Anzahl der Zeitleisten nicht zu stark erhöhen werden kann, ohne eine verminderte Übersicht zu riskieren. Daher ist die GUI-Anwendung auf die Steuerung von Robotern mit geringer Komplexität eingeschränkt. Eine weitere Erweiterungsmöglichkeit besteht in der Einführung von mehr Intensitätsstufen, wodurch eine präzisere Steuerung der

File: head_leftright.roc	**File: eyes_updown.roc**
neutral position for 1500 milliseconds	maximum up for 1000 milliseconds
maximum right for 2000 milliseconds	maximum up for 2500 milliseconds
	neutral position for 1000 milliseconds

Abbildung 4.5.: Roc-Output-Dateien zu dem RocGUI-Programmfenster in Abbildung 4.2

Auslenkungen der Motoren möglich wäre. Eine andere sinnvolle Erweiterung wäre die Integration eines Text-Editors zur direkten Bearbeitung von Roc-Befehlen. Der nahtlose Übergang zwischen den beiden Darstellungsformen könnte den Nutzern die Möglichkeit bieten je nach Situation die passende Darstellung zu wählen.

5. Evaluation der GUI-Anwendung

Zur Evaluation der graphischen Anwendung RocGUI wurde eine Studie durchgeführt welche die Anwendung mit der domänspezifischen Programmiersprache Roc vergleicht. Wie bereits beschrieben, wurde die graphische Anwendung als Erweiterung der Programmiersprache entwickelt. Der Vergleich der beiden Anwendungen soll nun darüber Aufschluss geben, ob die Erweiterung einen Mehrwert gegenüber der ursprünglichen Implementierung besitzt. Insbesondere wurde dabei der Zeitaufwand und die Arbeitsbelastung zur Steuerung des Roboterkopfes untersucht.

5.1. Hypothesen

Bei der Auswertung des Zeitaufwands wird erwartet, dass ein Nutzer durchschnittlich weniger Zeit benötigt, wenn er die Anwendung RocGUI verwendet. Ein Grund für diese Annahme ist, dass Laien oftmals nur sehr wenig Programmiererfahrung besitzen aber im Gegenzug an graphische Programme wie RocGUI gewöhnt sind. Desweiteren könnten Eigenschaften der Programmiersprache Roc, wie die fehlende Gesamtübersicht oder der längere Lernprozess, zur Aneignung der Syntax einen höheren Zeitbedarf bewirken. Daher wird die folgende Hypothese untersucht:

H1 Zur Erstellung von Bewegungsabläufen des Roboterkopfes wird durchschnittlich bei der Verwendung der Anwendung RocGUI weniger Zeit benötigt als bei der Verwendung der Programmiersprache Roc.

Bezüglich der Arbeitsbelastung wurde ein explorativer Ansatz verfolgt. Der Arbeitsaufwand wird durch eine Reihe von unterschiedlichen Aspekten berechnet. Dabei können die einzelnen Aspekte in der Gesamtheit durchaus unterschiedlich ausfallen. Dieses Zusammenspiel macht es schwer eine eindeutige Tendenz von Anfang an festzulegen. Daher wird im Folgenden die Fragestellung verfolgt, welche der beiden Steuerungsmöglichkeiten für Nutzer eine geringere Arbeitsbelastung bedeutet. Dabei wird die folgende Hypothese untersucht:

H2 Die Höhe der Arbeitsbelastung bei der Erstellung von Bewegungsabläufen des Roboterkopfes unterscheidet sich bei der Verwendung der Anwendung RocGUI und Programmiersprache Roc.

Durch einen Fragebogen werden die Programmierkenntnisse der Probanden abgefragt, um ein besseres Bild über die jeweiligen Fähigkeiten und Vorerfahrungen zu erhalten. Hierzu stellt sich die Frage, ob Programmiererfahrung den Umgang mit der Programmiersprache Roc erleichtert. Die daraus resultierende Zeitersparnis beim Programmieren, würde für eine kleinere Differenz der Bearbeitungszeiten von Roc und RocGUI sprechen. Konkret wird durch einen Korrelationstest untersucht, ob die Differenzen der Bearbeitungszeiten von Roc und RocGUI mit Programmiererfahrungen zusammenhängt. Hierzu wird die folgende Hypothese untersucht:

H3 Die Differenz der Bearbeitungszeiten von Programmiersprache Roc und von Anwendung RocGUI steht im Zusammenhang mit der Programmiererfahrung einer Person.

5.2. Beschreibung der Studie

5.2.1. Probanden

Die Studie wurde mit 14 Probanden durchgeführt, davon 9 männliche und 5 weibliche Personen im Alter von 20 bis 24 Jahren mit einem Durchschnittsalter von 21.86 Jahren ($SD = 1.1$). Bei allen Personen handelt es sich um Studenten, vor allem aus den Fachrichtungen Mathematik (9) und Informatik (4). Drei Personen befinden sich in einem Informatik Studium, eine Person im 2-Fach-Bachelor mit Fach Informatik, alle anderen Versuchspersonen haben keinen direkten Bezug zur Informatik oder ähnlichen Fachgebieten. Die genauen Daten zu allen Probanden können der Tabelle A.2 im Anhang entnommen werden.
Die für diese Studie interessanten Fähigkeiten und Vorerfahrungen der Probanden wurden von den Probanden selber eingeschätzt. Es wurde ein Fragebogen verwendet, welcher die Programmiererfahrungen mit und ohne Bezug zur Robotik sowie die allgemeinen Erfahrungen mit Robotern abfragt. Nur ein Proband hat sich als Experte, weitere drei haben sich als fortgeschritten in der Programmierung eingestuft. Der Großteil der Probanden (10) hat keine oder geringe Vorerfahrungen.
Die meisten (12) Probanden hatten bisher wenig bis gar keinen Kontakt zu Robotern, was natürlich wenig oder keine Erfahrung in der Roboterprogrammierung impliziert. Von den zwei Probanden, welche sich als fortgeschritten hinsichtlich Robotern ein-

Erfahrungen mit:	Keine	Anfänger	Fortgeschritten	Experte
Programmierung	3	7	3	1
Robotern	9	3	2	0
Roboterprogrammierung	9	4	1	0

Tabelle 5.1.: Übersicht über das Vorwissen der Probanden

gestuft haben, gab lediglich einer an fortgeschritten in der Programmierung von Robotern zu sein. Der andere stufte sich als Anfänger ein. Tabelle 5.1 zeigt einen genauen Überblick der Vorerfahrungen der Probanden.
Insgesamt ergibt sich das Bild einer eher unerfahrenen Gruppe. Der Großteil der Probanden kann als Laien in der Steuerung von Robotern eingestuft werden. Dies spiegelt gut die Zielgruppe der GUI-Anwendung wider, wie sie in Abschnitt 4.1 beschrieben wurde.

5.2.2. Material

Fragebogen zur Person Um die Charakteristik und Vorwissen der einzelnen Probanden abzufragen, wurde ein Fragebogen benutzt, welcher in Abbildung A.1 im Anhang zu finden ist. Es wurde das Alter, Geschlecht und das jeweilige Tätigkeitsfeld abgefragt. Zusätzlich wurden Programmiererfahrungen, Kenntnisse in der Programmierung von Robotern, wie auch allgemeine Erfahrungen im Umgang mit Robotern abgefragt. Jeder dieser drei Punkte konnte mit den vier Antwortmöglichkeiten: Keine, Anfänger, Fortgeschritten und Experte beantwortet werden.

NASA-TLX-Fragebogen Der sogenannte Task-Load-Index-Fragebogen [28] wurde von der Arbeitsgruppe Human Performance Group der Bundesbehörde NASA der Vereinigten Staaten entwickelt. Nach einer dreijährigen Entwicklungsphase wurde er 1986 erstmals veröffentlicht. Nach mehr als 20 Jahren wurde von Hart [29], einer Mitentwicklerin des Fragebogens, die Wirksamkeit und Anwendungsgebiete des Werkzeugs untersucht.
Er dient zur Messung der Arbeitsbelastung einer Person während der Erfüllung bestimmter Aufgaben. Es werden die sechs Faktoren: geistige, körperliche und zeitliche Anforderung, sowie Leistung, Anstrengung und Frustration abgefragt. Die Arbeitsbelastung ergibt sich aus dem gewichteten Mittelwert dieser Teilkomponenten. Eine frühere Version enthielt neun Faktoren, welche jedoch reduziert wurden um Redundanz zu vermeiden und um das Werkzeug praktikabler zu gestalten. Der Fragebogen enthält für jeden der Faktoren eine 21-stufige Bewertungsskala von 0 bis 100.

Wenn man von Arbeitsbelastung spricht gehen die Meinung auseinander was genau darunter zu verstehen ist. Die Problematik besteht darin, dass jede Person eine individuelle Meinung und Empfindung bezüglich der einzelnen Faktoren hat. Die Autoren des Fragebogens haben sich daher entschieden, jeder Person selbst zu überlassen welche Faktoren mehr oder weniger wichtig sind.
Der Fragebogen sollte unmittelbar nach Beendigung einer Aufgabe ausgefüllt werden. Im ersten Schritt muss die betreffende Person eine Bewertung für alle sechs Faktoren abgeben. Als nächstes wird die persönliche Meinung abgefragt, wie die einzelnen Faktoren gewichtet werden sollen. Die betreffende Person bekommt eine Liste mit jeder möglichen Kombination von zwei unterschiedlichen Faktoren, insgesamt 15 mögliche Paare. In jedem Fall muss sie sich entscheiden, welchen der beiden Faktoren sie für bedeutsamer hält. Die Häufigkeit mit der ein Faktor gewählt wurde wird mit dem ursprünglich gewählten Wert des Faktors multipliziert. Auf diese Weise werden alle gewichteten Werte der Faktoren berechnet. Im letzten Schritt werden diese aufsummiert und danach durch 15 geteilt.
Der ursprüngliche Fragebogen wurde in Englisch verfasst. Da alle Probanden deutschsprachig sind wurde der Fragebogen übersetzt, um ein richtiges Verständnis zu garantieren. Die Übersetzung wurde von der Interaction Design Group der Hochschule Magdeburg-Stendal [30] übernommen. Die Beschreibungen der Faktoren wurden jedoch in einigen Fällen gekürzt. In der Beschreibung der körperlichen Anforderung wurden die Beispiele auf den Inhalt der Studie angepasst. Die Originalversion führt die Beispiele: Ziehen, Drücken, Drehen, Steuern und Aktivieren an. In der verwendetet Version wurden die Beispiele Tippen, Maus bewegen und Maustasten drücken verwendet um einen direkten Bezug zu den benötigten körperlichen Aktivitäten der Studie herzustellen. Die verwendete Version des Fragebogens kann in Abbildung A.8 und A.9 im Anhang eingesehen werden.

Aufgaben Es wurden zwei unterschiedliche Aufgabentypen verwendet. Der erste Aufgabentyp zielt einzig auf die reine Steuerung der Bewegung des Roboters ab. Der zweite Aufgabentyp bezieht darüber hinaus die Sprachausgabe des Roboters mit ein. Eine Separation erfolgte da nicht klar war, wie sehr der Aspekt der Sprache das Ergebnis beeinflussen würde.
Jeder Aufgabentyp wurde für beide Applikationen verwendet, daher wurden zwei unterschiedliche Versionen genutzt um einen Lerneffekt zu vermeiden. Die Zuteilung der jeweiligen Versionen auf die Testpersonen erfolgte pseudo-randomisiert. Da für

beide Aufgabentypen zwei unterschiedliche Versionen existieren ergeben sich daraus vier Kombinationsmöglichkeiten, welche abwechselnd zugeteilt wurden.
Die vollständige erste Aufgabe ist im Anhang, Abbildung A.4 (Version A) und Abbildung A.5 (Version B)zu finden. Der Proband bekommt drei verschiedene Positionen des Roboterkopfes. Diese wurden durch eine Beschreibung, wie auch Darstellungen von Kopf und Augen festgelegt. Zu jeder Position ist ein Zeitabschnitt angegeben. Aufgabe der Probanden war, dass der Roboter sich innerhalb des Zeitintervalls in der entsprechenden Position befindet. Die beiden Versionen der Aufgabe unterscheiden sich nur durch die verwendeten Positionen und den zugehörigen Zeitintervallen. Die Aufgabenstellung lässt nur sehr wenig Raum für Interpretation und Eigeninitiative, daher ist die Richtigkeit einer Lösung sehr einfach zu erkennen. Ziel war es zusätzliche Belastungen durch beispielsweise kreative Denkprozesse zu vermeiden, wodurch die Testergebnisse verfälscht würden.
Die zweite Aufgabe wurde wie eine typische Anwendungssituation gestaltet mit dem Ziel Bewegungen und Sprache des Roboters aufeinander abzustimmen. Die komplette Aufgabe kann im Anhang auf Abbildung A.6 (Version A) und A.7 (Version B) eingesehen werden. Zu einer vorweg eingespielten Sprachdatei sollte der Roboter passende Bewegungen ausführen. Hierzu wurden den Probanden unterschiedliche Stichworte der Audio mit zugehörigen Roboterpositionen gegeben. Mit Stichworten benannte Gegenstände, wie zum Beispiel ein Bild oder eine Blumenvase wurden räumlich passend zum Roboter platziert.

5.2.3. Ablauf

Der Ablauf der Studie erfolgte nach dem Schema in Abbildung 5.1. Die Einführung beinhaltete eine kurze Vorführung des verwendeten Roboterkopfs InMoov und Informationen, welche für beide Applikationen relevant sind. Die Vorführung diente dazu, dass die Probanden einen ersten Eindruck über die Möglichkeiten des Roboterkopfes erhielten. Weiterhin wurde die grobe Funktionsweise wie die Bewegungsrichtungen der unterschiedlichen Körperteile und der Aufbau des Roboters vermittelt.
An diesem Punkt wurde ein Pseudo-Randomisierungsverfahren verwendet, um die Reihenfolge der beiden Steuerungsmöglichkeiten, Roc und RocGUI festzulegen. Es wurde immer abwechselnd mit einer der beiden Applikationen angefangen.
Im ersten Schritt wurde eine kurze Einführung der jeweiligen Applikation gegeben. Für die Programmiersprache Roc stand vor allem die Syntax der Befehle im Mittelpunkt. Eine Übersicht mit Beispielen in Papierform wurde verwendet um einen

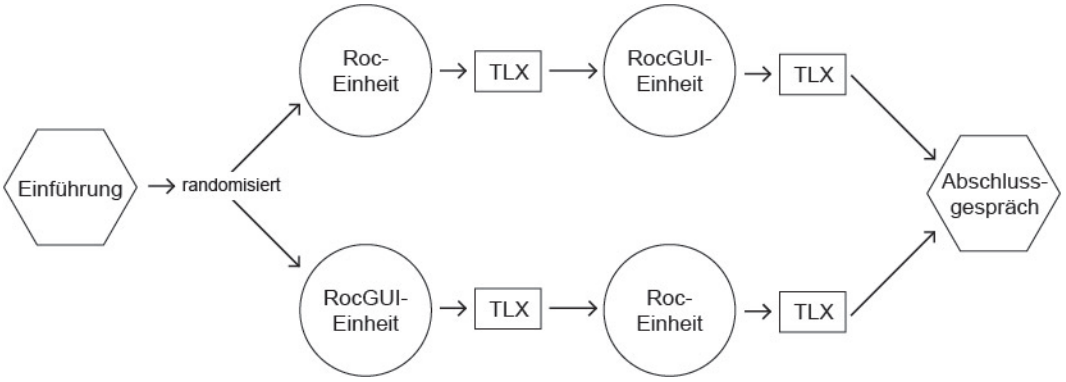


Abbildung 5.1.: Ablauf der Studie

schnellen Lernerfolg zu erzielen. Die Übersicht durfte von den Probanden während der Ausführung aller Aufgaben verwendet werden, da ansonsten die Merkfähigkeit der einzelnen Probanden stark in die Testergebnisse mit eingeflossen wäre. Das verwendete Dokument kann im Anhang Abbildung A.2 eingesehen werden. Außerdem wurde die Verwendung des Texteditors und die Weitergabe von Befehlen an den Roboterkopf gezeigt.
Zur Einführung der graphischen Anwendung RocGUI wurde beispielhaft eine neue Bewegung des Roboters erstellt und an dieser die unterschiedlichen Einstellungsmöglichkeiten gezeigt. Weiterhin wurde auf die beiden Buttons zur Sicherung und Weitergabe der Befehle an den Robotor aufmerksam gemacht.
Nach der Einführung zur jeweiligen Applikation wurde immer gleich vorgegangen: Im ersten Schritt wurden die Probanden angewiesen eine sehr kurze Trainingsaufgabe (siehe Abbildung A.3 im Anhang) zu bearbeiten. Alle Unklarheiten sollten an dieser Stelle ausgeräumt werden. Im nächsten Schritt wurden nacheinander Aufgabe 1 und 2 bearbeitet. Nach Beendigung beider Aufgaben wurde der TLX-Fragebogen ausgegeben mit der Anweisung die subjektiven Empfindungen gegenüber beiden Aufgaben wiederzugeben.
Nach der ersten Einheit (je nach Randomisierung Roc oder RocGui) wurde die jeweils andere Applikation auf genau die selbe Weise abgeprüft. Es wurde lediglich darauf geachtet die jeweils andere Version der vorherigen Aufgaben zu verwenden.

5.2.4. Messungen

Als Untersuchungsdesign wird mit einer Messwiederholung gearbeitet. Jeder Proband durchläuft dabei eine Einheit zur Programmiersprache Roc und eine weitere Einheit zur Anwendung RocGUI. Daher handelt es sich bei den gesammelten Daten um

abhängige Stichproben.
Innerhalb einer Einheit wurden den Probanden zwei unterschiedliche Aufgaben gestellt. Die Bearbeitungszeit wie auch die Anzahl der Aufrufe des Roboters wurden zu jeder einzelnen Aufgabe protokolliert. Je nach untersuchter Fragestellung wird entweder die Gesamtbearbeitungszeit oder die einzelnen Bearbeitungszeiten der beiden Aufgaben betrachtet. Zusätzlich wurde nach Beendigung einer kompletten Einheit der NASA-TLX-Fragebogen [28] ausgefüllt, um die Arbeitsbelastung festzuhalten. Diese kann Werte zwischen 0 und 100 haben, wobei ein kleiner Werte für eine geringe Arbeitsbelastung steht. Die Belastung wurde durch die unterschiedlichen Faktoren: Geistige, körperliche und zeitlich Anforderung, wie auch Leistung, Anstrengung und Frustration abgefragt. Durch das gewichtete arithmetische Mittel dieser Faktoren wurde der endgültige Wert festgelegt.
Durch einen Fragebogen am Anfang der Studie wurde Charakteristik und Fähigkeiten der Probanden abgefragt. Unter anderem erfolgte eine Einordnung der Programmiererfahrung in die vier Möglichkeiten: Keine, Anfänger, Fortgeschritten oder Experte. Zur Weiterverarbeitung wurden den Antwortmöglichkeiten die Werte von 0 bis 3 zugeordnet.

5.3. Ergebnisse

5.3.1. Quantitative Ergebnisse

Alle dokumentierten Daten können im Anhang Tabelle A.1 eingesehen werden. Zur Auswertung der Studie wurde die Programmiersprache R mit Umgebung RStudio verwendet. Es wurde das Signifikanzniveau von 5% festgelegt.
Im ersten Schritt wurden die Eigenschaften der Stichproben überprüft um geeignete statistische Tests festzulegen. Durch den Shapiro-Wilk-Test wurden alle Datensätze bzw. die Differenz der gepaarten Stichproben auf eine Normalverteilung geprüft. Da alle Stichproben außer der Programmiererfahrung durch den Test als annähernd normal verteilt klassifiziert wurden, konnte im weiteren Verlauf zum Vergleich der Mittelwerte auf den t-Test für verbundene Stichproben zurückgegriffen werden. Zur Testung des Zusammenhangs von Programmiererfahrung und dem Umgang mit der Programmiersprache Roc wurde der Spearman-Korrelationstest verwendet. Dieser ist robust gegenüber nicht normalverteilten Variablen wie der Programmiererfahrung.

Bearbeitungszeiten Abbildung 5.2 zeigt eine Übersicht über die Ergebnisse der Zeitmessungen. Zur Auswertung der H1 Hypothese wurde die Gesamtbearbeitungszeit in

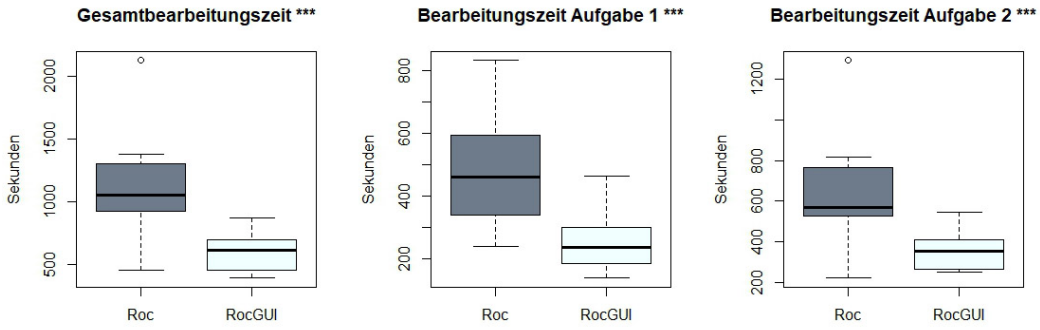


Abbildung 5.2.: Überblick über den Zeitaufwand zur Lösung der Aufgaben.
*: p<0.05 **: p<0.01 ***: p<0.001

Form der Summe der Bearbeitungszeiten von Aufgabe 1 und 2 betrachtet. Die durchschnittliche Bearbeitungszeit bei Verwendung von Roc beträgt 1088 Sekunden (18 Minuten und 8 Sekunden) ($SD = 409.15s$, $Minimum = 461s$, $Maximum = 2122s$). Bei der Verwendung der Anwendung RocGUI wurden im Mittel 600 Sekunden (10 Minuten) benötigt ($SD = 149.43$, $Minimum = 395s$, $Maximum = 871s$). Durch den einseitigen t-Test wurde Hypothese H1 bestätigt (t=5.26, p<0.001). Dies belegt einen deutlich geringeren Zeitaufwand bei der Verwendung der GUI-Anwendung.
Ursprünglich erfolgte eine Trennung der beiden Aufgabenstellungen, da nicht klar war, wie groß der Einfluss der Verwendung der Sprachausgabe des Roboters sein würde. Daher werden im Folgenden die Bearbeitungszeiten der Aufgaben im Einzelnen untersucht. Zur Bearbeitung von Aufgabe 1 mit der Programmiersprache Roc wurden durchschnittlich 469 Sekunden (7 Minuten und 49 Sekunden) benötigt ($SD = 161.96, Minimum = 239s, Maximum = 831s$). Mit der Anwendung RocGUI wurden durchschnittlich 245 Sekunden (4 Minuten und 5 Sekunden) zur Beendigung der Aufgabe benötigt ($SD = 89.56, Minimum = 138s, Maximum = 462s$). Mit Hilfe des einseitigen t-Tests für abhängige Stichproben wurde ein signifikant geringerer Zeitverbrauch der GUI-Anwendung festgestellt ($t = 6.1, p < 0.001$).
Zur Untersuchung der Zeiten von Aufgabe 2 wurde in gleicher Weise vorgegangen. Bei der Verwendung der Programmiersprache Roc wurde für Aufgabe 2 eine durchschnittliche Bearbeitungszeit von 619 Sekunden (10 Minuten und 19 Sekunden) ermittelt ($SD = 265.94, Minimum = 222s, Maximum = 1291s$). Die Anwendung RocGUI erzielte eine durchschnittliche Bearbeitungszeit von 356 Sekunden (5 Minuten und 56 Sekunden) ($SD = 85.54, Minmum = 253s, Maximum = 547s$). Auch hier besteht

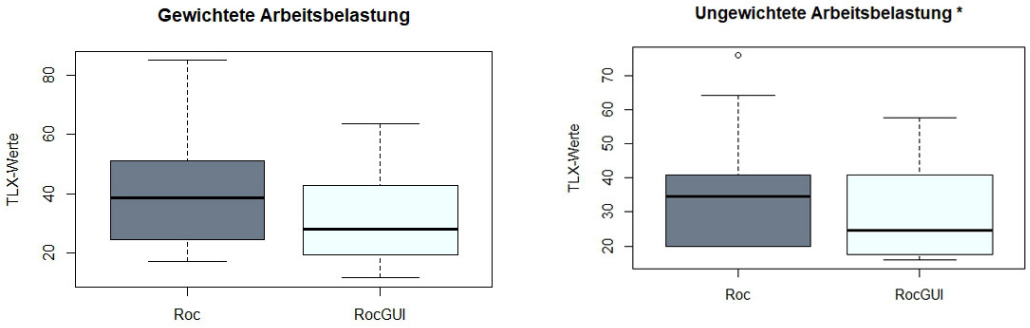


Abbildung 5.3.: Gewichtete und ungewichtete Auswertung des TLX-Fragebogens. *: p<0.05 **: p<0.01 ***: p<0.001

eine signifikant geringere Bearbeitungsdauer bei Verwendung der Anwendung RocGUI ($t = 4.04, p < 0.001$). Die Ergebnisse lassen den Rückschluss zu, dass in beiden Anwendungssituationen die Verwendung der graphischen Anwendung RocGUI den Zeitaufwand verringert.

Arbeitsbelastung Bei der Verwendung der Programmiersprache Roc beträgt der Durchschnittswert der Arbeitsbelastung 42.57 ($SD = 20.2$, $Minimum = 17.33$, $Maximum = 85$). Die Anwendung RocGUI wurde im Mittel mit einem Wert von 31.45 bewertet ($SD = 15.54, Minimum = 11.67, Maximum = 63.67$). Es konnte jedoch kein signifikanter Unterschied zwischen der Arbeitsbelastung der beiden Anwendungen festgestellt werden ($t = 2.13$, $p > 0.05$). Die Hypothese H2 wurde daher verworfen. Ein graphischer Vergleich der beiden Stichproben ist in Abbildung 5.3 zu sehen.
Allerdings ist die Exaktheit der zugrundeliegenden Daten fraglich. Im Verlauf der Studie hat sich gezeigt, dass vielen Probanden die Gewichtung der einzelnen Faktoren schwer gefallen ist. In Einzelfällen ist im Nachhinein aufgefallen, dass eher das eigene Können als die eigentliche Wichtigkeit der einzelnen Faktoren bewertet wurde. In einem weiteren Fall hat ein Proband im zweiten Durchlauf des Fragebogens festgestellt, dass er den ersten Fragebogen im Bezug auf die Gewichtung falsch ausgefüllt hat. Deshalb gibt es Grund zur Annahme, dass die ungewichteten Scores die tatsächliche Arbeitsbelastung besser widerspiegeln. Daher werden diese im Folgenden ebenfalls ausgewertet. Der zweiseitige t-Test hat einen signifikanten Unterschied zwischen der (ungewichteten) Arbeitsbelastung der beiden Anwendungen gezeigt ($t = 2.2, p < 0.05$). Abbildung 5.3 zeigt eine Übersicht über die ermittelten Daten.

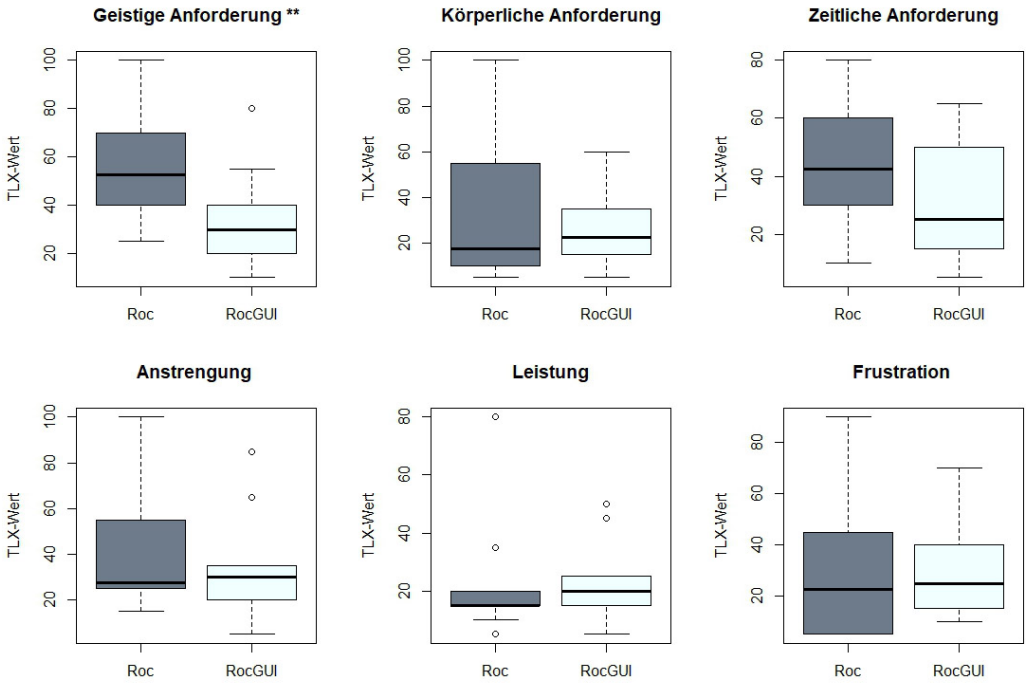


Abbildung 5.4.: Übersicht der Auswertung der NASA-TLX-Faktoren.
*: p<0.05 **: p<0.01 ***: p<0.001

Durchschnittlich wurde bei der Benutzung der Programmiersprache Roc ein Ergebnis von 36.96 ermittelt ($SD = 17.55, Minimum = 20, Maximum = 75.83$). Mit einem durchschnittlichen Ergebnis von 29.11 liegt die Anwendung RocGUI unterhalb dieses Wertes ($SD = 13.35, Minimum = 15.83, Maximum = 57.5$). Dies impliziert eine geringere Arbeitsbelastung der Anwendung RocGUI.

Im weiteren Verlauf wurden die einzelnen Faktoren der Arbeitsbelastung näher betrachtet, um das nicht signifikante Ergebnis der gewichteten Arbeitsbelastung näher zu beleuchten und eine differenziertere Bewertung zuzulassen. Eine Übersicht über die Ergebnisse der einzelnen Faktoren kann in Abbildung 5.4 eingesehen werden. Ein beidseitiger t-Test wurde auf die Daten aller sechs Faktoren des TLX-Fragebogens angewandt. Die geistige Anforderung sticht als einziger Faktor durch ein signifikantes Ergebnis heraus ($t = 3.64$, $p = 0.004$). Die durchschnittliche geistige Anforderung der Programmiersprache Roc beträgt 54.29 ($SD = 20.56, Minimum = 25, Maximum = 100$). Mit einem Durchschnittswert von 32.86 liegt bei der Verwendung der Anwendung RocGUI eine geringere geistige

Anforderung vor ($SD = 18.26, Minimum = 10, Maximum = 80$).
Bei den anderen Faktoren verzeichnet die Anwendung RocGUI jeweils einen leicht niedrigeren Durchschnittswert, jedoch konnten keine signifikanten Differenzen festgestellt werden.

Zusammenhang von Programmiererfahrung und Umgang der Programmiersprache Roc Der Zusammenhang der beiden unabhängigen Variablen Programmiererfahrung und Differenz der Bearbeitungszeiten der beiden Anwendungen wurden durch den Spearman-Korrelation Test untersucht. Zur einheitlichen Beurteilung der Bearbeitungszeiten wurde die Summe der Zeiten von Aufgabe 1 und 2 betrachtet. Hier konnte allerdings kein signifikanter Zusammenhang festgestellt werden ($rho = -0.24, p = 0.4$). Dies deutet darauf hin, dass vorhandene Programmiererfahrung nicht unbedingt eine erleichterte Benutzung der Programmiersprache Roc zur Folge hat.

5.3.2. Qualitative Ergebnisse

Die Programmiersprache wurde von einem Probanden als exakter wahrgenommen. „Bei der Programmiersprache durchdenke ich alles richtig“ begründete er seine Wahrnehmung. Zwei andere Probanden merkten an, dass sie ein größeres Erfolgserlebnis beim Bearbeiten der Aufgaben mit der Programmiersprache empfanden. Einer von ihnen meinte, dass er bei der direkten Programmierung einen höheren Eigenverdienst verspürt. Zusammenfassend lässt sich sagen, dass dieser Teil der Probanden die Bearbeitung mittels Programmiersprache als logikbasierter und deshalb befriedigender wahrgenommen hat. Dieses Gefühl könnte einen höheren Grad der Zufriedenheit mit der erbrachten Leistung durch die Programmiersprache mit sich bringen.
Fast alle Probanden fanden, dass die graphische Anwendung einfacher zu bedienen ist. Zwei nannten als Grund die bessere Übersicht über das komplette Geschehen. Bei der Programmiersprache sei es „schwierig nicht den Faden zu verlieren“. Dieser Aspekt könnte ein Grund für die deutlich kürzeren Bearbeitungszeiten der Anwendung RocGUI sein.
Die höheren Werte der geistigen Anstrengung der Programmiersprache könnten unter anderen durch die unbekannte Syntax der Roc-Befehle erklärt werden. Alle Probanden mussten in den ersten Minuten regelmäßig auf die bereitgestellte Übersicht der Syntax zurückgreifen. Nach den ersten Minuten hat ungefähr zwei Drittel der Probanden immer noch regelmäßig ihren Code mit der Übersicht verglichen. Ungefähr ein Viertel hat dies bis zum Ende getan. Im Besonderen sind Rechtschreibprobleme

bei den Wörtern „pronounced“ und „severe“ aufgefallen. Drei Probanden mussten mehrmals auf Rechtschreibfehler in ihrem Code aufmerksam gemacht werden bevor sie ihren Code ausgeführen konnten.
Ein weiterer Grund für die erhöhten Werte der geistigen Anforderung von Roc könnte der kognitive Aufwand zur Aufsummierung der Bewegungsdauern sein. Um den Zeitpunkt zu bestimmen an dem eine Bewegung beginnt und endet, müssen alle zeitlichen Informationen der vorherigen Codezeilen aufsummiert werden. Mehr als drei Viertel der Probanden hat zwischendurch auf Papier und Stift zurückgegriffen um den Vorgang zu vereinfachen.
Es wurden auch einige Usability-Probleme ausgemacht. Die größte Kritik erntete der interne Player der GUI-Anwendung. Durch fehlende Funktionen wie das Spulen der Audiodatei war ungefähr ein Viertel der Probanden sichtbar genervt. Solche Mängel haben vermutlich vor allem das Frustrationslevel der Probanden angehoben.

5.4. Diskussion

Das nicht signifikante Ergebnis der gewichteten Arbeitsbelastung und die Betrachtung der einzelnen Faktoren der Arbeitsbelastung machen klar, dass die GUI-Anwendung nicht in allen Bereichen eine Erleichterung darstellt. Die Mittelwerte der körperlichen Anforderung, Leistung, Anstrengung und Frustration zeigen nur einen geringen Unterschied zwischen den beiden Steuerungsmöglichkeiten. Den Beobachtungen innerhalb der Studie wurde entnommen, dass diese Faktoren sehr individuell bewertet wurden. Ungefähr ein Viertel der Probanden war frustriert von der Handhabung der GUI. Ein anderer Teil der Probanden war hingegen durch die Syntax und Tipparbeit der Programmiersprache genervt. Daher muss in dieser Sache je nach Vorlieben des Nutzers entschieden werden, welche Darstellungsweise besser geeignet ist. In zukünftigen Studien sollte versucht werden genauere Profile der Nutzer zu erstellen, um konkretere Aussagen treffen zu können.
Ein weiteres Ergebnis der Studie ist, dass kein direkter Zusammenhang zwischen Programmiererfahrungen und einem Peformanzgewinn der Programmiersprache Roc existiert. Die Fähigkeiten der Probanden wurde durch ein Selbsteinschätzungsverfahrung festgelegt. Bei dieser Erhebung von Daten kann es natürlich zu erheblichen Abweichungen kommen. Ein weiterer Grund für dieses Ergebnis könnte die spezielle Testgruppe der Studie sein. Ein großer Teil der Probanden sind Studenten aus dem Bereich der Mathematik. Als solche haben sie meistens keine oder wenig Programmiererfahrung, sehr wohl aber in den meisten Fällen eine sehr logische Denkweise.

Diese Denkweise könnte genau wie Programmierfähigkeiten einen Vorteil in der Benutzung einer Programmiersprache bedeuten. Diese Vermutung wird durch den Fakt unterstützt, dass die vier Probanden mit den geringsten Unterschieden in den Bearbeitungszeiten von GUI und Programmiersprache alle aus dem mathematischen Bereich kommen. Dies stellt natürlich in keinster Weise einen Beweis dar, liefert jedoch einen Ansatz an dem nähere Forschungs- und Erklärungsversuche ansetzen können. In zukünftigen Studien sollte auf die Diversität der Probanden geachtet werden. Fast alle Teilnehmer der Studie sind Studenten aus dem Bereich der Mathematik oder Informatik. Diese spezielle Population macht eine Verallgemeinerung der Ergebnisse unmöglich.
Ein Punkt der von einigen Probanden im Abschlussgespräch genannt wurde, ist die bessere Übersicht der Bewegungen des Roboters mit Hilfe der Anwendung RocGUI. In Abschnitt 4.3.1 wurde die Vermutung geäußert, dass die Übersichtlichkeit mit Hilfe der Visualisierung zunimmt. Jede Bewegung einer Komponente des Roboters wird durch ein farbiges Rechteck innerhalb einer Zeitleiste symbolisiert. Die Start- und Endzeitpunkte der Bewegungen werden durch die Positionierung und Länge der Rechtecke kodiert. Zusätzlich lässt sich aus der Positionierung und Beschriftung der Rechtecke die zu bewegende Komponente und die Richtung der Bewegung bestimmen. Diese Eigenschaften der RocGUI-Anwendung tragen vermutlich dazu bei, dass Nutzer einen schnelleren Überblick über alle Bewegungen des Roboters erhalten. Als Einschränkung muss an dieser Stelle gesagt werden, dass von den Probanden nur kurze und einfache Bewegungsabläufe erstellt wurden. In der Praxis werden jedoch oft sehr viel längere und komplexere Bewegungen benötigt. Daher können die Aussagen über die gute Übersicht der GUI-Anwendung nur mit Vorbehalt ausgewertet werden. Die Problematik bei langen Bewegungsabläufe könnte darin bestehen, dass der Nutzer auf eine Scrollleiste angewiesen ist um durch die Zeitleiste zu navigieren. Die Steuerung eines komplexeren Roboters mit mehr Freiheitsgraden könnte den selben Effekt haben. Für weitere Freiheitsgrade müssen mehr Zeitleisten implementiert werden. Sobald nicht mehr alle Zeitleisten untereinander passen, muss hier ebenfalls mit einer Scrollleiste gearbeitet werden. Beide Fälle könnten dazu führen, dass ein Nutzer zusätzlich verunsichert wird und den Überblick verliert. Diese Tatsache limitiert die Ergebnisse der Studie auf kurze Bewegungsabläufe und Roboterplattformen mit ähnlicher Komplexität wie der verwendete InMoov-Roboterkopf. Der Einfluss dieser beiden Faktoren könnte Gegenstand weiterführender Nachforschungen sein, um die Wirksamkeit der GUI-Anwendung näher zu untersuchen.

6. Fazit

Im Mittelpunkt dieser Arbeit steht die Frage ob eine text-basierte Programmiersprache oder eine grafische Anwendung eine leichtere Steuerung eines humanoiden Roboterkopfs ermöglicht. Um eine direkten Vergleich anzustellen wurde für die domänenspezifische Programmiersprache Robot-Control-Language (Roc) [1] eine grafische Benutzungsoberfläche entwickelt. Das Interface ist zur Generierung von sequenzielle Bewegungsabfolgen eines Roboterkopfs fähig. Alle Motoren können dabei vollkommen unabhängig und parallel bewegt werden. Zur Abspeicherung und Weitergabe der Befehle an den Roboter werden die Benutzereingaben in eine Befehlskette der Roc Programmiersprache umgewandelt und im Folgenden durch ein Robotersystem weiterverarbeitet. Die Bewegungen des Roboters werden in zeitlicher Abfolge auf der Benutzungsoberfläche angeordnet. Jedem Freiheitsgrad des Roboters wird eine eigene Zeitleiste zugewiesen, sodass alle beweglichen Komponenten des Roboters zeitlich unabhängig ansteuerbar sind. Bewegungen der Motoren werden durch farbliche Markierung innerhalb der Zeitleisten symbolisiert. Dieses Konzept lässt eine einfache Anpassung für andere Roboterplattformen zu. Prinzipiell müssen lediglich weitere Zeitleisten hinzugefügt werden, um mehr Freiheitsgrade eines Roboters zu implementieren.
Zu Beantwortung der ursprünglichen Frage wurde eine Studie durchgeführt, welche die text-basierte Programmiersprache mit der GUI-Anwendung vergleicht. Im Hinblick auf den Zeitaufwand, konnte durch die Benutzung der GUI-Anwendung eine deutliche Ersparnis nachgewiesen werden. Im Gegensatz dazu, konnte bei der Untersuchung der Arbeitsbelastung keine eindeutige Tendenz festgestellt werden. Allerdings ergaben nähere Untersuchungen im Bereich der geistigen Anforderung eine klare Entlastung der Probanden durch die GUI. Insgesamt lässt sich sagen, dass die entwickelte Anwendung durchaus einen Mehrwert gegenüber der text-basierten Steuerung besitzt.
Im Blick auf die derzeitige Entwicklung wird sich der Einsatz von Robotern in den persönlichen wie auch geschäftlichen Alltag vieler Menschen immer weiter verstärken. Im Zuge dieser Verbreitung wird die Nachfrage nach intuitiven und flexiblen

Steuerungen von Robotern immer größer. Viele Forschungsarbeiten befassen sich deshalb mit der Entwicklung von geeigneten Steuerungsmöglichkeit. Diese Arbeit trägt durch den empirischen Vergleich von einer text-basierten und einer visuellen Steuerungsmöglichkeit zu dieser Diskussion bei.

A. Anhang

Tabelle A.1.: Übersicht über die Daten der Studie

Probandennummer		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Aufgabe 1 Zeit	Roc	401	515	437	268	610	599	831	332	339	239	593	486	358	561
in Sekunden	RocGUI	209	462	147	142	327	236	301	280	190	138	237	184	247	324
Aufgabe 2 Zeit	Roc	805	552	528	246	693	531	1291	412	586	222	764	553	668	816
in Sekunden	RocGUI	381	409	325	253	405	547	400	313	265	262	417	255	427	319
Aufgabe 1	Roc	1	1	1	1	2	5	1	2	1	1	1	1	1	1
Roboter Aufrufe	RocGUI	1	2	1	1	1	2	2	1	1	1	1	1	1	1
Aufgabe 2	Roc	3	1	1	1	1	3	3	2	2	1	1	1	1	2
Roboter Aufrufe	RocGUI	2	2	1	1	1	6	2	1	1	1	1	1	1	1
TLX Werte*	Roc	51.3	85.0	38.7	17.3	35.7	24.7	60.3	36.3	22.0	38.7	74.3	18.7	45.7	47.3
	RocGUI	43.0	63.7	11.7	53.3	16.7	22.3	33.7	36.0	21.7	44.0	37.3	21.7	16.0	19.3
RTLX Werte*	Roc	35.8	75.8	33.3	20.0	28.3	20.0	56.7	26.7	20.0	40.8	64.2	20.0	37.5	38.3
	RocGUI	40.8	57.5	15.8	40.8	15.8	18.3	35.8	27.5	18.3	41.7	39.2	21.7	16.7	17.5
Geistige	Roc	25	100	45	30	65	40	65	45	40	70	70	35	60	70
Anforderung	RocGUI	40	80	10	55	20	40	30	30	25	35	40	15	15	25
Gewichtung	Roc	4	5	5	4	5	5	3	4	3	4	4	2	5	5
	RocGUI	1	5	3	5	2	5	3	4	1	0	5	5	3	4
Körperliche	Roc	5	100	5	10	20	10	70	20	15	60	55	20	10	15
Anforderung	RocGUI	60	40	30	15	20	15	35	5	25	55	35	20	5	10
Gewichtung	Roc	0	3	1	4	0	2	0	1	1	3	0	0	0	0
	RocGUI	4	0	1	0	3	0	2	1	4	4	0	0	4	2
Zeitliche	Roc	75	55	60	30	25	10	70	50	15	30	80	30	45	40
Anforderung	RocGUI	35	25	10	65	15	10	50	60	5	50	60	15	25	25
Gewichtung	Roc	5	1	4	0	2	0	4	5	4	2	3	3	4	2
	RocGUI	5	2	4	4	3	3	2	5	0	1	3	1	1	3
Leistung	Roc	80	10	20	10	20	20	15	15	15	5	35	15	15	35
	RocGUI	50	45	5	20	15	20	25	15	20	20	15	25	10	15
Gewichtung	Roc	3	2	3	4	3	4	2	3	5	5	1	5	3	4
	RocGUI	2	4	5	1	5	3	4	3	5	3	4	4	3	5
Anstrengung	Roc	25	100	25	20	20	25	70	25	30	35	55	15	65	45
	RocGUI	30	85	25	65	15	15	35	35	5	30	35	30	30	20
Gewichtung	Roc	2	4	2	1	4	1	5	2	2	1	2	2	2	2
	RocGUI	3	3	2	3	2	1	3	2	2	3	2	2	4	1
Frustration	Roc	5	90	45	20	20	15	50	5	5	45	90	5	30	25
	RocGUI	30	70	15	25	10	10	40	20	30	60	50	25	15	10
Gewichtung	Roc	1	0	0	2	1	3	1	0	0	0	5	3	1	2
	RocGUI	0	1	0	2	0	3	1	0	3	4	1	3	0	0

*Werte wurden auf eine Nachkommastelle gerundet

Abbildung A.1.: Fragebogen zur Person

Proband Nr.

Alter: _____

Geschlecht: _____

Studiengang/Beruf/Tätigkeit: _____

Programmiererfahrung:

☐ *Keine* ☐ *Anfänger* ☐ *Fortgeschritten* ☐ *Experte*

Erfahrung mit Robotern:

☐ *Keine* ☐ *Anfänger* ☐ *Fortgeschritten* ☐ *Experte*

Erfahrung mit Roboterprogrammierung:

☐ *Keine* ☐ *Anfänger* ☐ *Fortgeschritten* ☐ *Experte*

Abbildung A.2.: Übersicht und Beispiele der Roc-Syntax

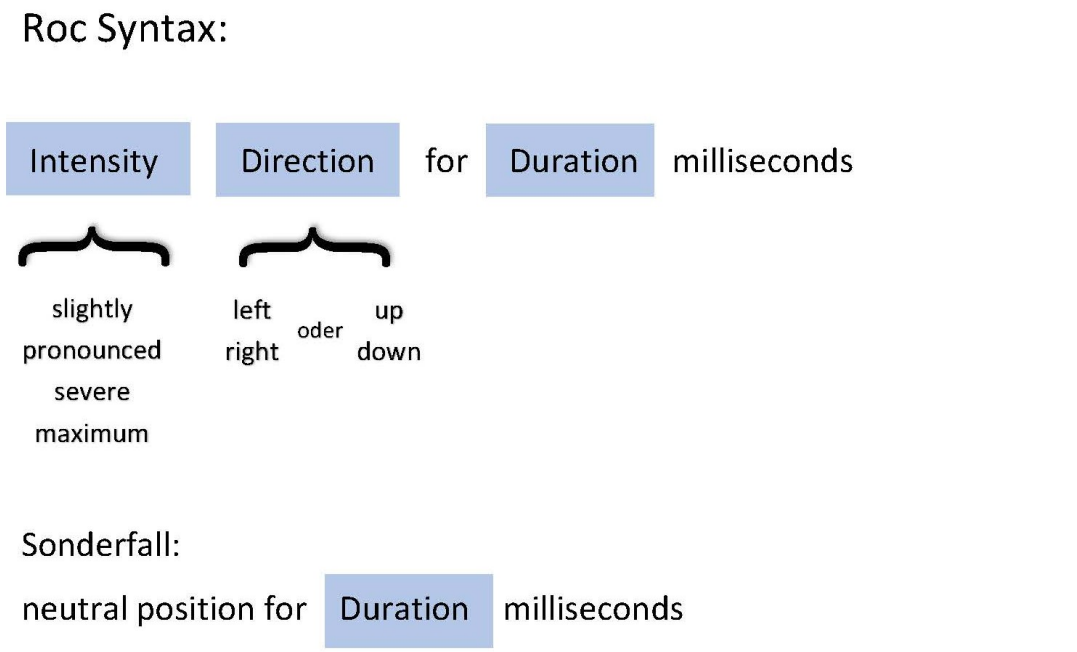


Tabelle A.2.: Probandendaten

Probanden-nummer	Geschlecht	Alter	Studiengang	Programier-erfahrung	Roboter-erfahrung	Roboterprogram-miererfahrung
1	weiblich	22	2-Fach Bachelor (Informatik,Wirtschaft)	Anfänger	Anfänger	Anfänger
2	weiblich	21	2-Fach-Bachelor (Mathe, Theologie)	Anfänger	Keine	Keine
3	männlich	23	Informatik Bachelor	Experte	Fortgeschritten	Anfänger
4	männlich	21	2-Fach-Bachelor (Mathe, Chemie)	Keine	Keine	Keine
5	männlich	21	Informatik Bachelor	Fortgeschritten	Fortgeschritten	Fortgeschritten
6	männlich	24	Informatik Master	Fortgeschritten	Anfänger	Anfänger
7	weiblich	23	Lehramt Sekundarstufe (Mathe, Theologie)	Keine	Keine	Keine
8	männlich	21	2-Fach-Bachelor (Mathe,Latein)	Anfänger	Keine	Keine
9	männlich	20	2-Fach-Bachelor (Mathe, Sport)	Anfänger	Keine	Keine
10	männlich	22	Mathe und Philosophie Bachelor	Fortgeschritten	Anfänger	Anfänger
11	weiblich	22	Mathe Bachelor	Anfänger	Keine	Keine
12	männlich	22	2-Fach-Bachelor (Mathe, Chemie)	Keine	Keine	Keine
13	männlich	23	Ingenieurspädagogik Bachelor	Anfänger	Keine	Keine
14	weiblich	21	Mathe Bachelor	Anfänger	Keine	Keine

Abbildung A.3.: Traningsaufgabe

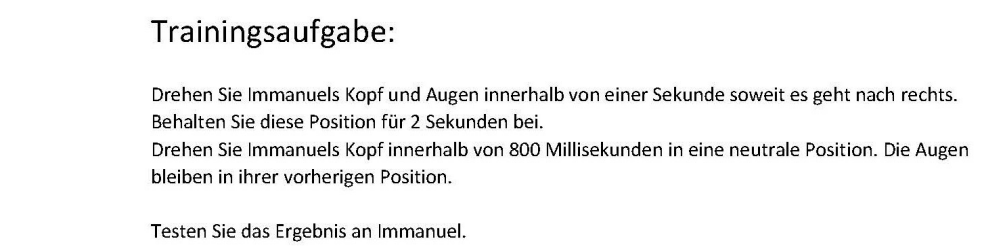


Abbildung A.4.: Aufgabe 1 Version A
Bildquelle des InMoov-Roboterkopfes: Dobrusin [1]

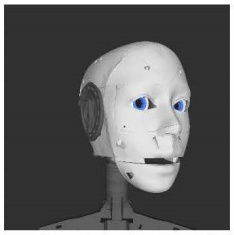
Version A

Aufgabe:

Im unteren Bereich finden Sie eine Reihe von Zeitabschnitten und zugehörige Positionen. Ziel ist es, dass Immanuel sich zu jedem Zeitpunkt innerhalb des Intervalls in der jeweiligen Position befindet.

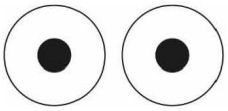
Zeitabschnitt: 1 000 – 2 500 Millisekunden (1 – 2,5 Sekunden)

Kopfposition:



Kopf stark (severe) nach links

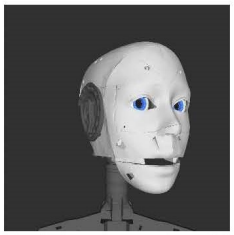
Augenposition:



neutrale Position

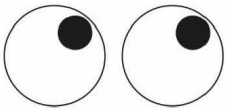
Zeitabschnitt: 2 800 – 5 500 Millisekunden (2,8 - 5,5 Sekunden)

Kopfposition:



Kopf stark (severe) nach links

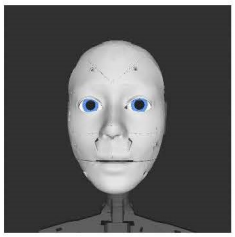
Augenposition:



Augen extrem (maximum) nach oben und stark (severe) nach links

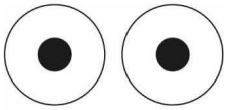
Zeitabschnitt: 6 000 – 6 500 Millisekunden (6 – 6 .5 Sekunden)

Kopfposition:



neutrale Position

Augenposition:



neutrale Position

Abbildung A.5.: Aufgabe 1 Version B
Bildquelle des InMoov-Roboterkopfes: Dobrusin [1]

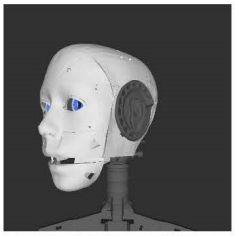
Version B

Aufgabe:

Im unteren Bereich finden Sie eine Reihe von Zeitabschnitten und zugehörige Positionen. Ziel ist es, dass Immanuel sich zu jedem Zeitpunkt innerhalb des Intervalls in der jeweiligen Position befindet.

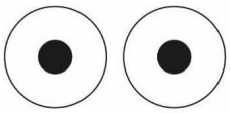
Zeitabschnitt: 1 000 – 2 000 Millisekunden (1 – 2 Sekunden)

Kopfposition:



Kopf stark (severe) nach rechts

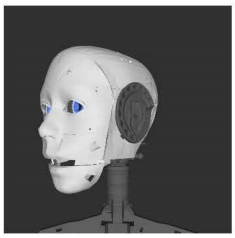
Augenposition:



neutrale Position

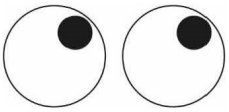
Zeitabschnitt: 2 300 – 5 500 Millisekunden (2,3 - 5,5 Sekunden)

Kopfposition:



Kopf stark (severe) nach rechts

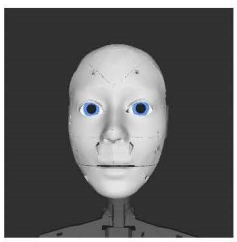
Augenposition:



Augen extrem (maximum) nach oben und stark (severe) nach links

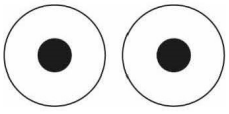
Zeitabschnitt: 6 500 – 7 000 Millisekunden (6,5 – 7 Sekunden)

Kopfposition:



neutrale Position

Augenposition:



neutrale Position

Abbildung A.6.: Aufgabe 2 Version A
Bildquelle des InMoov-Roboterkopfes: Dobrusin [1]

Version A

Aufgabe:

Jetzt geht es um das richtige Timing. Nachfolgend finden Sie einen Text, welchen Immanuel sprechen wird. Dieser Text enthält fettgedruckte Schlüsselwörter. Zu jedem dieser Schlüsselwörter finden Sie am Ende der Seite eine Position, welche Immanuel einnehmen soll sobald er das Wort ausspricht. Außerdem soll er diese Position mindestens eine halbe Sekunde halten bevor er sich weiterbewegt.

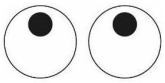
Gesprochener Text

„Ich wohne schon lange in der Technischen Fakultät. Diesen Raum kenne ich wie meine Westentasche. Dort steht der **Computer** und auf dieser Seite kann ich einen hübschen **Blumenstrauß** sehen. Mein Lieblingsgegenstand ist aber dieses **Bild**."

Stichworte und zugehörige Position

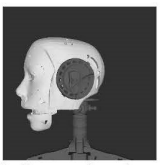
Computer

Kopf etwas (pronounced) nach links

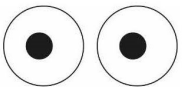


Augen stark (severe) nach oben

Blumenstrauß



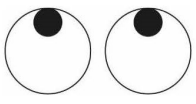
Kopf extrem (maximum) nach rechts



neutral position

Bild

Kopf etwas (pronounced) nach rechts



Augen extrem (maximum) nach oben

Abbildung A.7.: Aufgabe 2 Version B
Bildquelle des InMoov-Roboterkopfes: Dobrusin [1]

Version B

Aufgabe:

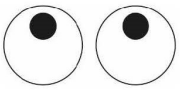
Jetzt geht es um das richtige Timing. Nachfolgend finden Sie einen Text, welchen Immanuel sprechen wird. Dieser Text enthält fettgedruckte Schlüsselwörter. Zu jedem dieser Schlüsselwörter finden Sie am Ende der Seite eine Position, welche Immanuel einnehmen soll sobald er das Wort ausspricht. Außerdem soll er diese Position mindestens eine halbe Sekunde halten bevor er sich weiterbewegt.

Gesprochener Text:
„Siehst du dieses **Bild**? Es zeigt meine beiden Roboterfreunde, die über einen Zebrastreifen gehen. Ich weiß auch wie das geht. Man muss zuerst nach **links** schauen und dann nach **rechts**. Wenn ein Auto kommt wartet man bis es anhält und geht dann über die Straße."

Stichworte und zugehörige Position

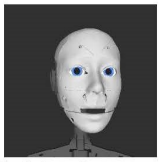
Bild

Kopf stark (severe) nach rechts

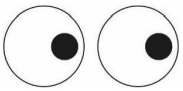


Augen stark (severe) nach oben

links

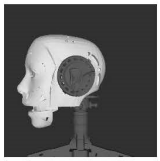


Kopf etwas (pronounced) nach links

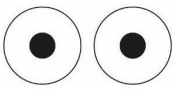


Augen stark (severe) nach links

rechts



Kopf extrem (maximum) nach rechts



neutral position

Abbildung A.8.: NASA-TLX-Fragebogen Seite 1
Quelle: In Anlehnung an [28]

Probanden Nr.

Beanspruchungshöhe

Geben Sie für jede der untenstehenden Kriterien an, wie hoch die Beanspruchung war. Markieren Sie dazu bitte auf den folgenden Skalen, in welchem Maße Sie sich in den sechs genannten Kriterien von der Aufgabe beansprucht oder gefordert gesehen haben:

Beispiel:

Geistige Anforderungen
Wie viel geistige Anstrengung war bei der Informationsaufnahme und -verarbeitung Erforderlich (z.B. Denken, Entscheiden, Rechnen, Erinnern, Suchen...)?

Körperliche Anforderungen
Wie viel körperliche Aktivität war erforderlich (z.B. Tippen, Maus bewegen, Maustasten drücken, ...)?
War die Aufgabe einfach oder anstrengend, erholsam oder mühselig?

Zeitliche Anforderungen
Wie viel Zeitdruck empfanden Sie hinsichtlich der Aufgabe?

Leistung
Wie erfolgreich haben Sie Ihrer Meinung nach die vom Versuchsleiter gesetzten Ziele erreicht?
Wie zufrieden sind Sie mit Ihrer Leistung?

Anstrengung
Wie hart mussten sie arbeiten, um Ihre Leistung zu erreichen?

Frustration
Wie unsicher, entmutigt, irritiert, gestresst und verärgert fühlten Sie sich während der Aufgabe?

Abbildung A.9.: NASA-TLX-Fragebogen Seite 2
Quelle: In Anlehnung an [28]

Umkreise Sie im nachfolgenden die Beanspruchungskriterien, welche für das Gesamtempfinden hinsichtlich der Aufgabe, die jeweils bedeutsamere war.

Geistige Anforderung oder Körperliche Anforderung	Anstrengung oder Leistung	Frustration oder Anstrengung
Zeitliche Anforderung oder Frustration	Körperliche Anforderung oder Frustration	Leistung oder Geistige Anforderung
Frustration oder Leistung	Anstrengung oder Geistige Anforderung	Zeitliche Anforderung oder Körperliche Anforderung
Leistung oder Körperliche Anforderung	Zeitliche Anforderung oder Anstrengung	Geistige Anforderung oder Zeitliche Anforderung
Geistige Anforderung oder Frustration	Körperliche Anforderung oder Anstrengung	Leistung oder Zeitliche Anforderung

Kontrollieren sie bitte, ob Sie zu allen Fragen Angaben gemacht haben.
Bei Unklarheiten wenden Sie sich bitte an die Versuchsleiterin / den Versuchsleiter.

Literaturverzeichnis

[1] I. Dobrusin, "Implementation of a domain-specific language for controlling an antropomorphic robot head," Bachelorthesis, Albert-Ludwigs-University Freiburg, Department of Computer Science, Research Group on the Foundation of Artificial Intelligence, 2016.

[2] F. Lindner and M. M. Bentzen, "The hybrid ethical reasoning agent immanuel," in *Proceedings of 2017 Conference on Human-Robot Interaction (HRI2017)*, pp. 187–188, 2017.

[3] F. Felix Lindner, M. M. Bentzen, and B. Nebel, "The hera approach to morally competent robots.," in *Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017)*, pp. 6991–6997, 2017.

[4] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. C. Shepherd, and D. Franklin, "Evaluating coblox: A comparative study of robotics programming environments for adult novices," in *Proceedings of 2018 CHI Conference on Human Factors in Computing Systems*, pp. 366:1–366:12, 2018.

[5] G. Biggs and B. MacDonald, "A survey of robot programming systems," in *Proceedings of the Australasian conference on robotics and automation*, 2003.

[6] E. M. Orendt, M. Fichtner, and D. Henrich, "Robot programming by non-experts: Intuitiveness and robustness of one-shot robot programming," in *Proceedings of 25th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 192–199, 2016.

[7] A. Weiss, J. Igelsböck, S. Calinon, A. Billard, and M. Tscheligi, "Teaching a humanoid: A user study with hoap-3 on learning by demonstration," in *Proceedings of 18th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 147–152, 2009.

[8] N. Buchina, S. Kamel, and E. Barakova, "Design and evaluation of an end-user

friendly tool for robot programming," in *Proceedings of 25th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 185–191, 2016.

[9] M. C. Carlisle, "Raptor: A visual programming environment for teaching object-oriented programming," *Journal of Computing Sciences in Colleges*, vol. 24, pp. 275–281, 2009.

[10] D. Saito, H. Washizaki, and Y. Fukazawa, "Comparison of text-based and visual-based programming input methods for first-time learners," *Journal of Information Technology Education-Research*, vol. 16, pp. 209–226, 2017.

[11] C. Schmidt, *Generierung von Struktureditoren für anspruchsvolle visuelle Sprachen.* Dissertation, Universität Padaborn, 2006.

[12] S. Schiffer, "Visuelle Programmierung - Potential und Grenzen," in *Jahrestagung der GI (Gesellschaft für Informatik)*, pp. 267–286, 1996.

[13] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: programming for all," *Communications of the ACM*, pp. 60–67, 2009.

[14] F. Kalelioğlu and Y. Gülbahar, "The effects of teaching programming via scratch on problem solving skills: A discussion from learners' perspective.," *Informatics in Education*, vol. 13, pp. 33–50, 2014.

[15] J. H. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk, "Programming by choice: Urban youth learning programming with scratch," in *Annual Meeting of SIGCSE (Special Interest Group on Computer Science Education)*, pp. 367–371, 2008.

[16] J. Huang and M. Cakmak, "Code3 a system for end-to-end programming of mobile manipulator robots for novices and experts," in *Proceedings of 2017 IEEE International Conference on Human-Robot Interaction*, pp. 453–462, 2017.

[17] S. H. Kim and J. W. Jeon, "Programming lego mindstorms nxt with visual programming," in *Proceedings of 2007 International Conference on Control, Automation and Systems*, pp. 2468–2472, 2007.

[18] E. Pot, J. Monceaux, R. Gelin, and B. Maisonnier, "Choregraphe: a graphical tool for humanoid robot programming," in *Proceedings of 18th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 46–51, 2009.

[19] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "Mechatronic design of nao humanoid," in *Proceedings of r2009 IEEE International Conference on Robotics and Automation*, pp. 769–774, 2009.

[20] M. S. Horn and R. J. K. Jacob, "Designing tangible programming languages for classroom use," in *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, pp. 159–162, 2007.

[21] Y. Sefidgar S, P. Agarwal, and M. Cakmak, "Situated tangible robot programming," in *Proceedings of IEEE International Conference on Human-Robot Interaction*, pp. 473–482, 2017.

[22] T. Sapounidis, S. Demetriadis, and I. Stamelos, "Evaluating children performance with graphical and tangible robot programming tools," *Personal and Ubiquitous Computing*, vol. 19, pp. 225–237, 2015.

[23] G. Langevin, "Inmoov, open source 3d printed life-size robot." http://inmoov.fr/, Online; accessed: 14.08.2018, 2012.

[24] E. Friesen and P. Ekman, "Facial action coding system. a technique for the measurement of facial movement," *Consulting Psychologists Press, Palo Alto*, 1978.

[25] M. Schröder and J. Trouvain, "The german text-to-speech synthesis system mary: A tool for research, development and teaching," *International Journal of Speech Technology*, vol. 6, pp. 365–377, 2003.

[26] Livecode Ltd, "Livecode lessons - the livecode message path and the structure of a livecode application - picture." http://lessons.livecode.com/m/4603/l/565724-the-livecode-message-path. Online, accessed 10.8.2018.

[27] C. Ware, *Information visualization: perception for design*, p. 3. Elsevier, 2012.

[28] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*, vol. 52, pp. 139–183, 1988.

[29] S. G. Hart, "Nasa-task load index (nasa-tlx); 20 years later," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, pp. 904–908, 2006.

[30] Interaction-Design-Group, “NASA-TLX (Kurzfassung deutsch),” Fragebogen, HS Magdeburg-Stendal, 2016.