FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Virtual Reality zum Anfassen

VR-System mit robotergestütztem haptischem Feedback

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Clemens Gatterer, BSc
Matrikelnummer 00627821

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Priv. Doz. Mag. Dr. Hannes Kaufmann
Mitwirkung: Dipl.-Ing. Mag. Emanuel Vonach, Bakk.

Wien, 5. Dezember 2017

Clemens Gatterer	Hannes Kaufmann

Erklärung zur Verfassung der Arbeit

Clemens Gatterer, BSc
Schlossmühlgasse 22
2351 Wiener Neudorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 5. Dezember 2017

Clemens Gatterer

Danksagung

An dieser Stelle möchte ich mich bei meinen Eltern, meiner Freundin und meinen Freunden bedanken, die mir geduldig mit moralischer Unterstützung und Anregungen zur Seite gestanden sind.

Weiterer Dank gilt Wolfgang, der mich mit seinem Tischlerhandwerk bei der Fertigung der Requisiten unterstützt hat.

Schließlich bedanke ich mich bei Priv. Doz. Mag. Dr. Hannes Kaufmann und Dipl.-Ing. Mag. Emanuel Vonach Bakk. für die Betreuung sowie bei allen Teilnehmern der Benutzerstudie.

Kurzfassung

Virtuelle Umgebungen werden dank stetig besser werdender Hardware immer realistischer. Die Immersion wird aber oft durch fehlendes oder unzureichendes haptisches Feedback bei der Interaktion mit virtuellen Objekten gestört. Bestehende Ansätze für haptisches Feedback, wie spezielle Eingabegeräte, Exoskelette oder Simulatoren, können dieses Problem nur eingeschränkt beheben, da das dargebotene Feedback entweder sehr rudimentär ist, die Bewegungsfreiheit eingeschränkt ist oder nur einzelne Anwendungen unterstützt werden. Einen Lösungsweg beschreiben spezielle haptische Feedbacksysteme, die mit unterschiedlichen Methoden, unter anderem auch Requisiten (*physical props*), verschiedene virtuelle Objekte in der Realität darstellen. Die existierenden Systeme haben allerdings einen großen Platzbedarf oder schränken aufgrund technischer Limitierungen die Bewegungsfreiheit innerhalb der virtuellen Welt ein. Diese Diplomarbeit beschreibt die Konzipierung und Implementierung eines immersiven *Virtual Reality* (*VR*)-Systems, das durch einen Roboterarm Requisiten in Reichweite der Benutzerin bzw. des Benutzers platziert und so haptisches Feedback bereitstellen kann. Dazu wurde im Zuge der Umsetzung ein Framework erstellt, das gebrauchsfertige VR-Produkte mit einem eigens angefertigten Roboter kombiniert. Eine eingesetzte Bewegungsplattform erlaubt die Simulation von uneingeschränkten Welten auf geringem Platz. Die einzelnen Softwarekomponenten werden gemeinsam mit einer Robotiklösung zu einem leicht zu verwendenden Unity3D-Framework verbunden, das als Basis für neue Anwendungen mit robotergestütztem haptischem Feedback dienen kann. Das System ist so konzipiert, dass es möglich ist, den verwendeten Roboterarm zu einem späteren Zeitpunkt leicht auszutauschen. Ebenso ist es möglich, nur ein Subset der unterstützten Hardwarekomponenten zu verwenden. Die durchgeführte Benutzerstudie und die technische Evaluierung belegen die Funktionsfähigkeit des erstellten Prototypen und zeigen Tendenzen zu positiven Auswirkungen von haptischem Feedback auf die VR-Erfahrung auf. Für eine gefahrlose Verwendung des Systems, werden Sicherheitsvorkehrungen beschrieben, deren Angemessenheit ebenfalls in der Benutzerstudie bestätigt werden konnte.

Abstract

Continuously improving hardware increases the realism in *Virtual Reality* (*VR*). However, the immersion is often disturbed due to the lack of appropriate haptic feedback when interacting with virtual objects. Existing approaches like haptic input devices or exoskeletons cannot fully solve this problem, as the provided feedback is either too simple or user movement is restricted. Simulators can provide suitable haptic feedback but they cannot be used for different applications. There are special haptic feedback systems that do not suffer most of these constraints. They use varying methods, e.g. physical props, to simulate the sensation during the interaction with an object. The main drawback of existing feedback systems is that the required physical space increases with larger virtual worlds. This thesis describes the design and the implementation of an immersive VR system that provides haptic feedback by using a robotic arm to position physical props around the user. As a basis for the system, a framework was developed that combines various out-of-the-box VR components with a custom-built robotic arm. A locomotion platform enables virtual worlds of unlimited size on a constant physical space. The system provides an easy to use Unity3D framework wrapping the required software libraries and a robotics system. This framework builds the basis for creating new VR applications with robotic haptic feedback. The framework allows to use only a subset of the supported VR devices without changing the application and also enables later modifications on the robotic arm. The performed technical evaluation and user study prove the functional capability of the system and indicate positive effects of haptic feedback on the VR experience. Safety guidelines for the use of the systems are specified to minimize the risk when using the system. The user study proves that these guidelines are satisfactory.

Inhaltsverzeichnis

KAPITEL 1

Einleitung

1.1 Motivation und Problemstellung

In den letzten Jahren gab es einige technologische Weiterentwicklungen im Bereich der *Virtual Reality* (*VR*). *Head-Mounted Display*s (*HMDs*), wie die Oculus Rift oder die HTC Vive, erzielen durch eine höhere Auflösung eine deutlich verbesserte Bildqualität. Mit den dazugehörenden Eingabegeräten können Benutzer bzw. Benutzerinnen die Bewegungen ihrer Hände in die virtuellen Welten übertragen und mit Objekten interagieren. Zusätzlich bieten optische Lösungen, wie beispielsweise die Leap Motion Orion, die Möglichkeit, die bloßen Hände selbst ohne daran befestigte Peripherie zu verwenden. Optische Ganzkörper-Trackingsysteme, wie zum Beispiel die Microsoft Kinect oder Motion-Capture-Anzüge, wie etwa der Perception Neuron, synchronisieren die Körperteile der Spielfigur mit ihren realen Gegenstücken. Auch die Fortbewegung kann dank Bewegungsplattformen auf eine fast natürliche Art und Weise stattfinden. Zu diesen Geräten zählen der Virtualizer von Cyberith und die Omni von Virtuix.

Alle diese Komponenten können dazu beitragen, bestimmte Aspekte der VR-Erfahrung realistischer zu machen. Das funktioniert solange gut, bis Benutzerinnen bzw. Benutzer mit virtuellen Objekten interagieren wollen. Durch die Peripherie bewegen sich die Hände des Avatars zwar exakt wie jene des Anwenders bzw. der Anwenderin, allerdings kann durch das berührte Objekt einfach hindurch gegriffen werden. Laut Robles-De-La-Torre[Rob06] erschwert dies ein schnelles Erlernen von Aufgaben die eine hohe Genauigkeit erfordern. Das Fehlen von haptischem Feedback fällt besonders negativ auf, wenn keine optimale visuelle Darstellung vorhanden ist, etwa weil die Benutzerin bzw. der Benutzer nicht direkt auf das Objekt blickt oder es teilweise verdeckt ist. Sallnäs et al.[SRS00] legen nahe, dass haptisches Feedback positive Auswirkungen auf die Ausführung von Aufgaben und auf die Präsenzwahrnehmung hat. Es gibt eine Reihe an Ansätzen, um durch haptisches Feedback virtuelle Objekte erfahrbar zu machen:

Haptische Eingabegeräte, wie die PHANTOM® Omni™, bieten eine Schnittstelle zwischen der virtuellen und der realen Welt. Sie schränken jedoch die Möglichkeiten deutlich ein, wie Benutzer bzw. Benutzerinnen mit der Welt interagieren können. So kann beispielsweise die PHANTOM® Omni™ als Zauberstab verwendet werden, mit dem die Konturen von Gegenständen erfühlt werden können.[And+06] Diese Objekte können aber nur mit dem Stab, nicht aber dem Unterarm oder gar den Beinen berührt werden. Ebenso ist es nicht möglich, die Oberfläche der Objekte zu spüren.

Um haptisches Feedback auch an anderen Körperteilen zu ermöglichen, können am Körper tragbare Geräte verwendet werden. Diese können Kleidungsstücke wie Handschuhe oder Westen sein, die das haptische Feedback durch Druck oder Vibration simulieren.[Lin+04; GRS11] Die Bewegungsfreiheit bleibt dabei großteils erhalten. Im Gegensatz dazu können durch Exoskelette Kräfte auf bestimmte Körperregionen ausgeübt werden. Dadurch können etwa Daumen und Zeigefinger nur soweit geschlossen werden, wie es die Kontur des virtuellen Objekts zulässt.[AJK03; Fon+13] Frisoli et al.[Fri+05] beschreiben ein System, das einen ganzen Arm umfasst und somit auch dort Feedback erzeugen kann. Zusätzlich bieten Exoskelette die Möglichkeit die Position der Finger oder des Arms exakt in die Anwendung zu übertragen. Der Nachteil dieser Systeme liegt erneut darin, dass das haptische Feedback auf eine spezielle Körperregion beschränkt ist. Weiters ist die Bewegungsfreiheit in der Regel eingeschränkt, wodurch es nicht möglich ist eine VR-Umgebung zu erstellen, in der sich die Anwenderin bzw. der Anwender frei bewegen kann.

Eine weitere Option ist der Einsatz eines Simulators. Dieser ist speziell für ein Szenario gebaut und kann viele der darin vorkommenden Elemente in der Realität nachbilden. Ein Beispiel hierfür ist ein Simulator für Kühlschranktüren[Shi+12]. Mit ihm kann der Widerstand beim Öffnen und Schließen unterschiedlich beschaffener Türen dargestellt werden. Der Nachteil einer solchen Lösung ist, dass für jede Welt ein eigener Simulator hergestellt werden muss.

Eine vielseitige Möglichkeit stellen speziell zugeschnittene Feedbacksysteme dar, die auf unterschiedliche Arten virtuelle Objekte in der realen Welt spürbar machen. Dazu können unter anderem Luftstöße[Sod+13] oder in die richtige Position gebrachte Requisiten dienen. Diese Requisiten können entweder durch Menschen bewegt werden[Che+15] oder Teil eines Roboterarms sein[Tac+94]. Der Vorteil dieser Systeme ist, dass eine Vielzahl an Objekten abgebildet werden kann, ohne dass der Anwender oder die Anwenderin ein spezielles Gerät verwenden muss. Weiters ist es möglich, dass Gegenstände mit der Benutzerin oder dem Benutzer interagieren, indem beispielsweise ein Ball zugeworfen wird. Bei der Verwendung von Luftstößen kann allerdings keine durchgängige Oberfläche dargestellt werden. Das von Cheng et al.[Che+15] beschriebene System(*TurkDeck*) benötigt mehrere Personen, die die Requisiten bedienen sowie einiges an Platz, da sich der Anwender bzw. die Anwenderin frei im Raum bewegen kann. Der Platz- und Personenbedarf des Systems von Tachi et al.[Tac+94] ist deutlich geringer, dafür ist das haptische Feedback erneut auf die Hände beschränkt.

Diese Diplomarbeit beschreibt ein Feedbacksystem, das durch einen Roboterarm Re-

quisiten in Reichweite der Benutzerin bzw. des Benutzers platziert und so haptisches Feedback bereitstellen kann. So kann beispielsweise eine Holzplatte verwendet werden, um die Illusion einer Wand zu erzeugen. Bewegt sich der Benutzer bzw. die Benutzerin in der virtuellen Welt auf die Wand zu, so verschiebt der Roboter die Platte entsprechend in seine bzw. ihre Richtung. Dazu werden eine Reihe von gebrauchsfertigen Produkten mit einem eigens angefertigten Roboter kombiniert. Die einzelnen Softwarekomponenten werden gemeinsam mit einer Robotiklösung zu einem leicht zu verwendenden Unity3D-Framework verbunden, das als Basis für neue Anwendungen mit passiv-haptischem Feedback dienen kann. Das System ist so konzipiert, dass der verwendete Roboterarm zu einem späteren Zeitpunkt leicht ausgetauscht werden kann. Ebenso ist es möglich, nur ein Subset der unterstützten Hardwarekomponenten zu verwenden. Wird in einem Szenario beispielsweise kein exaktes Handtracking benötigt, so kann die Leap Motion Orion einfach abgesteckt werden, ohne in der Anwendung etwas anpassen zu müssen. Weiters ist es auch möglich, die Robotersteuerung auf einen zweiten Computer auszulagern, um Ressourcen besser nützen zu können. Um all diese Anforderungen zu erfüllen, wurde eine gekapselte Software-Architektur gewählt, die aktuelle Softwarepraktiken berücksichtigt. Das Potential des Systems wurde mit einer technischen Auswertung und einer empirischen Benutzerstudie analysiert. Die Ergebnisse zeigen, dass das erstellte System in der Lage ist, Kollisionen mit Passanten in einer Stadt und eine Holzwand zu simulieren. Die funktionierende Abstraktion der Roboterhardware wird durch die einzelnen Testsitzungen bestätigt, da beliebig eine der beiden Aufgaben ohne Feedback durchgeführt wurde. Bei Vortests zur Benutzerstudie und Demonstrationen konnte gezeigt werden, dass die erstellten Anwendungen auch funktionieren, wenn nicht alle Komponenten des Setups im Einsatz sind. Dadurch ist der Vorbereitungsaufwand zwar geringer, die Genauigkeit des Körpertrackings wird aber entsprechend reduziert.

Die Auswertungen in Kapitel 5 zeigen aber auch Verbesserungspotenzial. Besonders Reaktionszeit und Leistung des verwendeten Roboterarms waren in manchen Fällen nicht zureichend. Die Reaktionszeit wird auch durch Vorgänge in *Open Robotics Automation Virtual Environment* (*OpenRAVE*) beeinflusst. Anpassungen etwa im Bereich der Serialisierung und Deserialisierung von Roboterpfaden können hier eine Steigerung bringen.

Diese Diplomarbeit ist im Rahmen eines Projekts der Interactive Media Systems Forschungsgruppe am Institut für Softwaretechnik und Interaktive Systeme entstanden und führte im Zuge dessen auch zu einer internationalen wissenschaftlichen Publikation[VGK17].

1.2 Ziele

Ziel dieser Arbeit ist die Erstellung eines VR-Systems das die Grundlage für die Erstellung von VR-Anwendungen bietet, die haptisches Feedback auf Basis von Requisiten unterstützen. Das dazu erstellte Framework soll die aktuell gebräuchliche VR-Peripherie unterstützen und um einen speziell zusammengestellten Roboterarm erweitert werden.

Der Fokus liegt auf einer guten Integration der einzelnen Softwarekomponenten in ein leicht zu verwendendes Framework für VR-Anwendungen mit robotergestütztem haptischem Feedback. Dieses Framework soll als Asset für Unity3D erstellt werden und eine Beispielumgebung enthalten, welche die Möglichkeiten demonstriert, wie der Roboterarm eingesetzt werden kann. Um den Vorbereitungsaufwand gering halten zu können, sollen mit diesem Framework erstellte Anwendungen auch dann funktionieren, wenn nicht alle Komponenten verwendet werden.

Laut Robles-De-La-Torre[Rob06] ist die Reaktionszeit der Feedbacklösung für realistisches haptisches Feedback von großer Bedeutung. Daher verwendet diese Diplomarbeit als Grundlage für die Robotersteuerung die Robotiklösung OpenRAVE. OpenRAVE erzeugt auf Basis eines 3D-Modells des Roboters einen *Inverse Kinematic* (*IK*)-Löser, wodurch die Berechnungsdauer zur Laufzeit deutlich reduziert wird.[Dia10] Auch die Kollisionsabfrage und die Planung eines Pfades für mehrere Roboter ist in OpenRAVE abgedeckt.

Diese Diplomarbeit kapselt die Robotiksoftware in ein leicht zu integrierendes Modul für C++- und C#-Anwendungen. Ein speziell angefertigtes Unity3D-Skript erleichtert die Verwendung innerhalb der 3D-Engine. Das Robotikmodul abstrahiert die Beschaffenheit und Anzahl der angebundenen Roboterarme. Dadurch ist es möglich den Arm auszutauschen, einen weiteren Arm hinzuzufügen oder den Feedbackmechanismus komplett zu deaktivieren, ohne die VR-Umgebung anpassen zu müssen.

Ein weiteres Ziel ist es, mit Hilfe dieses Systems herauszufinden, wie überzeugend ein Roboterarm virtuelle Objekte in der Realität erfahrbar machen kann. Dazu wird ein Versuchsaufbau mit einem Roboterarm erstellt. Der Arm ist an einer festen Position angebracht, wodurch nur in einem bestimmten Bereich Feedback möglich ist. Dies ist für die Auswertung dennoch ausreichend, da in diesem Bereich der volle Funktionsumfang des Roboters zur Verfügung steht. Die Eigenschaften des Arms werden in einer technischen Evaluierung ausgewertet, um etwaige Verbesserungsansätze zu lokalisieren. Eine empirische Benutzerstudie mit 34 Testpersonen soll eine Indikation darüber abgeben, wie das System und besonders der Roboterarm akzeptiert werden. Die dazu erstellten zwei Szenarien basieren auf der Beispielanwendung des beschrieben Unity3D-Frameworks.

In einer technischen Auswertung und einer empirischen Benutzerstudie sollen die Funktionstauglichkeit des Systems sowie die Auswirkungen des haptischen Feedbacks auf die Benutzererfahrung eruiert werden. Ebenso sollen anhand der Ergebnisse mögliche Schwachstellen aufgedeckt und Verbesserungsvorschläge erstellt werden.

1.3 Gliederung

Diese Diplomarbeit besteht aus 7 Kapiteln. Nach der Einleitung in Kapitel 1 beschreibt Kapitel 2 den aktuellen Stand der Technik im Bereich des haptischen Feedbacks. Um eine Entscheidungsbasis dafür zu bieten, wie das in dieser Arbeit erstellte System haptisches Feedback erzeugt, werden die einzelnen Kategorien analysiert. Dabei wird auf die Vor- und Nachteile der jeweiligen Ansätze sowie auf konkrete Umsetzungen eingegangen.

Kapitel 3 listet die Hard- und Softwarekomponenten auf, die in diesem System verwendet werden. Ein besonderer Fokus liegt auf der Robotikumgebung OpenRAVE, da diese das Kernstück der Steuerungslogik dieses Frameworks bildet. Neben den Eigenschaften und Funktionsweisen wird auch das Zusammenspiel der einzelnen Teile beschrieben. Auf Basis dieser Beschreibungen wird begründet, warum die jeweiligen Komponenten ausgewählt wurden.

Kapitel 4 zählt zum Hauptteil dieser Diplomarbeit und enthält die einzelnen Fertigungsschritte des vorgestellten VR-Systems. Zu Beginn wird erörtert wie die in Kapitel 3 erwähnten Komponenten eingesetzt werden. Dabei werden auch die Iterationen der Konzeptualisierung beschrieben, die notwendig waren, um die technischen Anforderungen abzudecken. Anschließend werden die evaluierten Konfigurationen des Roboterarms erläutert. Die unterschiedlichen Versionen werden anhand von Simulationen verglichen, wodurch jener Aufbau ausgewählt werden kann, der die gewünschte Funktionalität bietet. Die darauffolgenden Abschnitte erläutern den Einsatz von OpenRAVE und der 3D-Engine Unity3D, sowie die Implementierungsdetails des erstellten Frameworks. Dazu zählt auch eine Beschreibung der verwendeten Entwicklungskonzepte. Der letzte Teil des Kapitels behandelt die getroffenen Sicherheitsvorkehrungen und die möglichen Gefahren bei der Verwendung des Systems.

Kapitel 5 schildert die durchgeführten Evaluierungen. Die technische Analyse überprüft die funktionalen Eigenschaften des VR-Systems. Ein besonderes Augenmerk wird dabei auf die Auswertung des Roboterarms gelegt. Der zweite Teil befasst sich mit der Benutzerstudie. Dabei werden zuerst die erstellten Testszenarien beschrieben, bevor auf die Methodik und die Ergebnisse eingegangen wird.

In Kapitel 6 werden die Resultate des vorherigen Kapitels ausgewertet. Hierbei zeigt sich, dass das System die funktionalen Anforderungen erfüllt und die Sicherheitsvorkehrungen ausreichend sind. Auf Basis der Resultate werden auch die Schwächen der Umsetzung sowie mögliche Lösungsansätze erwähnt. Den Abschluss des Kapitels bildet ein Ausblick auf mögliche kommerzielle Anwendungsgebiete für das VR-System.

Kapitel 7 beinhaltet eine finale Zusammenfassung dieser Diplomarbeit und zählt einige Ansätze für weitere wissenschaftliche Arbeiten mit dem erstellten Feedbacksystem auf.

KAPITEL 2

Related Work

Die Bemühung, Haptik und *Virtual Reality* (*VR*) zu kombinieren, ist keine neue Bestrebung. Laut Benali-Khoudja[Ben04] lässt sich speziell nach 1990 eine Vielzahl an Veröffentlichungen zu diesem Thema finden.

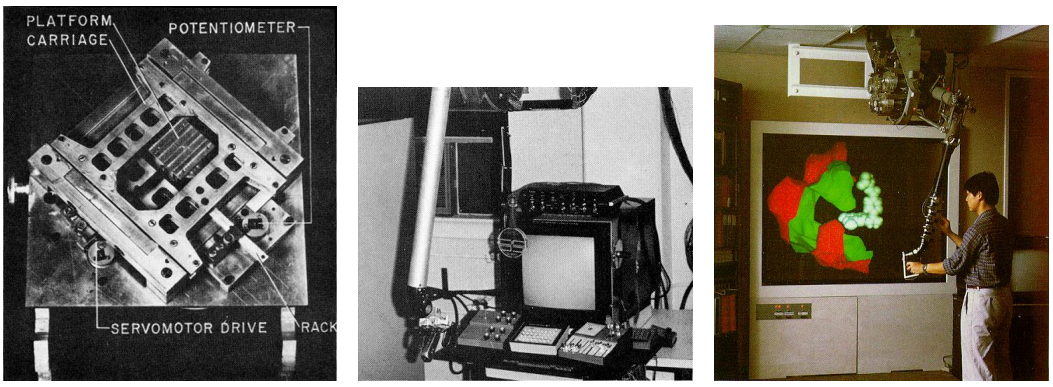
Dieses Kapitel gibt einen Überblick über einige Ansätze haptisches Feedback in VR zu bewerkstelligen und worin der Unterschied zu dieser Diplomarbeit besteht. Zur besseren Übersicht werden die Verfahren in vier Kategorien eingeteilt. Abschnitt 2.1 beschreibt spezielle Eingabegeräte, die sowohl eine Benutzereingabe, als auch haptisches Feedback bieten. Abschnitt 2.2 beschäftigt sich mit am Körper der Anwenderin bzw. des Anwenders angebrachte Geräte, die als Eingabemöglichkeit und Feedbackquelle dienen. Simulatoren, die die Eindrücke einer speziellen Tätigkeit simulieren, werden in Abschnitt 2.3 angeführt. In Abschnitt 2.4 werden andere spezielle Feedbackysteme dargestellt, die eine Vielzahl von Empfindungen erzeugen können und nicht auf die Simulation einer bestimmten Tätigkeit beschränkt sind.

2.1 Haptische Eingabegeräte

Es gibt eine Fülle an verschiedenen haptischen Eingabegeräten. Diese unterscheiden sich zum einen in der Anzahl der *Freiheitsgrade*, die für Eingabe und Feedback zur Verfügung stehen, und zum anderen, durch den Umstand, ob sie an einem bestimmten Platz montiert oder frei beweglich sind. Anhand dieser Kriterien sind die Geräte dieses Abschnitts gruppiert. Im Gegensatz zu dem Ansatz dieser Arbeit sind diese Geräte jeweils für bestimmte Körperteile vorgesehen. Abschnitt 2.1.1 beschreibt einige stationäre Geräte mit sechs Freiheitsgraden, während Abschnitt 2.1.2 jene mit weniger aufzählt. Abschnitt 2.1.3 nimmt abschließend Bezug auf frei bewegliche Geräte.

2.1.1 Stationäre Geräte mit sechs Freiheitsgraden

Stationäre Geräte mit haptischem Feedback gibt es bereits seit einiger Zeit. Die ältesten Geräte mit wissenschaftlicher Anwendung sind jene der von Brooks et al.[Bro+90] angeführten GROPE Reihe. GROPE-I ist ein Prototyp mit dem ein Cursor innerhalb eines virtuellen, zweidimensionalen Kräftefeldes bewegt werden kann. Die Auswirkungen der Kraftlinien im Feld werden mithilfe von Servomotoren an den Griff übermittelt. Dadurch ist es dem Anwender bzw. der Anwenderin möglich die Kräfte im Feld zu spüren und den Linien zu folgen. GROPE-I unterstützt keine sechs Freiheitsgrade. Dazu sind erst die weitaus komplexeren Weiterentwicklungen GROPE-II und GROPE-III in der Lage. Als Eingabegeräte dient ein modifizierter Teleoperator, der *Argonne Remote Manipulator* (*ARM*). Der ARM bietet sechs Freiheitsgrade: Translation und Rotation für x-, y- und z-Koordinate. Zusätzlich ist für jeden Freiheitsgrad Force Feedback verfügbar. Das bedeutet, dass in den Gelenken Motoren enthalten sind, die die Position und Ausrichtung des Griffstücks verändern können. Damit lassen sich vielerlei Aufgaben im virtuellen dreidimensionalen Raum erledigen. Als Beispielanwendungen erwähnt Brooks et al.[Bro+90] die Interaktion mit virtuellen Bausteinen und die Simulation von molekularem Docking. Die Geräte der GROPE Reihe ermöglichen im Unterschied zum Ansatz dieser Arbeit nur haptisches Feedback für die Hände mittels eines speziellen Griffs. Da sich das Gerät an einer festen Position befindet, ist der Nutzungsraum auf seine Reichweite beschränkt. Abbildung 1 zeigt die drei Versionen von GROPE.

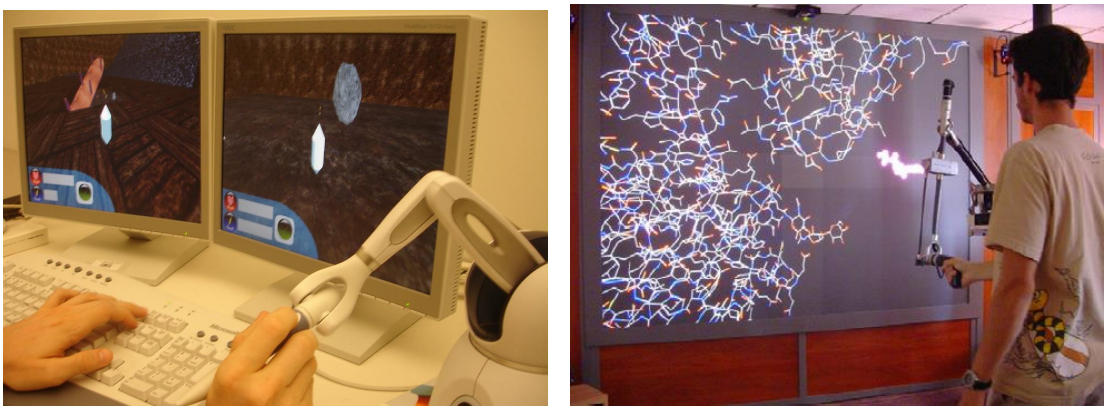


(a) (b) (c)

Abbildung 1: Alle GROPE Versionen: a) GROPE-I b) GROPE-II, c) GROPE-III im Einsatz für molekulares Docking.[Bro+90]

Mittlerweile gibt es platzsparendere Geräte, die ebenfalls sechs Freiheitsgrade für die Eingabe bieten. Zu den bekanntesten Vertretern zählen unter anderem die Geräte der PHANTOM®-Reihe von Sensable[Sen15] und die der Virtuoise™-Produktlinie von Haption SA[Hap15]. Die beiden Produkte sind in Abbildung 2 dargestellt. Beide Produkte liefern Feedback für das Verschieben im virtuellen Raum, aber nur die Virtuoise™-Geräte bieten zusätzlich Feedback für die Rotationen. Die Geräte finden eine Vielzahl von Anwen-

dungen, unter anderem in Videospielen[And+06] und bei Docking Simulationen[WM07; DR09]. In *HaptiCast*[And+06] steuert die Spielerin oder der Spieler verschiedene Zauberstäbe direkt mit einer PHANTOM® Omni™. Die verschiedenen Eigenschaften der Stäbe werden durch unterschiedliches Feedback direkt spürbar. Die Funktionsweise von HaptiCast ist in Abbildung 2a zu sehen. Wollacott et al.[WM07] verwenden gleichfalls eine PHANTOM® Omni™. Das Gerät wird verwendet, um diverse Andockpositionen an einem bestehenden Molekül auszuprobieren. Die Auswirkungen der Kräfte an jeder Position werden über das Gerät zurück übermittelt. Das ermöglicht es dem Forscher bzw. der Forscherin eine möglichst optimale Dockingstelle zu finden. Daunay et al.[DR09] verfolgen denselben Ansatz, allerdings kommt hier eine Virtuose™ 6D zum Einsatz. Dadurch steht der Forscherin bzw. dem Forscher auch Feedback über die Rotationskräfte zur Verfügung. Wie bei GROPE sind die Interaktionsmöglichkeiten nur mit einem Griff möglich und somit auf die Hände beschränkt. Ebenso ist keine freie Bewegung im Raum möglich. Abbildung 2b zeigt den Einsatz einer PHANTOM® Omni™ bei molekularem Docking.



(a) (b)

Abbildung 2: PHANTOM®- und Virtuose™-Geräte im Einsatz: a) PHANTOM® Omni™ als Zauberstab in dem Computerspiel HaptiCast[And+06], b) molekulares Docking mit einer Virtuose™ 6D[DR09].

Ein anderes Konzept, um ebenso sechs Freiheitsgrade zu erzielen, zeigen Murayama et al.[Mur+04]. Die verwendete SPIDAR G besteht aus einer Kugel die mittels Fäden schwebend in einem Gestell befestigt ist. Mithilfe dieser beweglichen Fäden können Rotationen und Translationen der Kugel registriert werden. Ebenso können diese Fäden die Kugel manipulieren, wodurch das haptische Feedback zustande kommt. Murayama et al.[Mur+04] verwenden zwei SPIDAR G gleichzeitig , um ein beidhändiges Arbeiten zu ermöglichen. Auch dieses Konzept kann nur mit den Händen verwendet werden und der Aktionsraum ist durch die Position der Geräte vorgegeben. Zusätzlich schränken die Fäden mögliche Bewegungen stark ein. Das SPIDAR G&G genannte Setup wird in Abbildung 3 gezeigt.

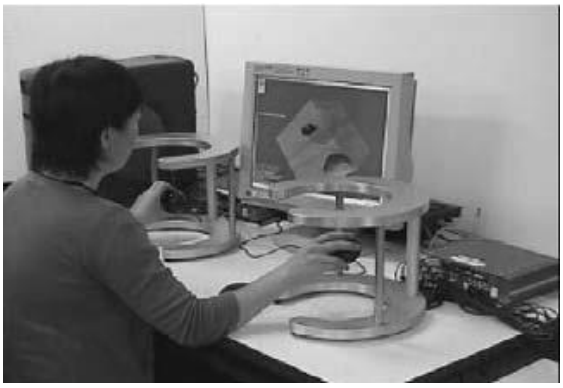


Abbildung 3: SPIDAR G&G in beidhändiger Verwendung.[Mur+04]

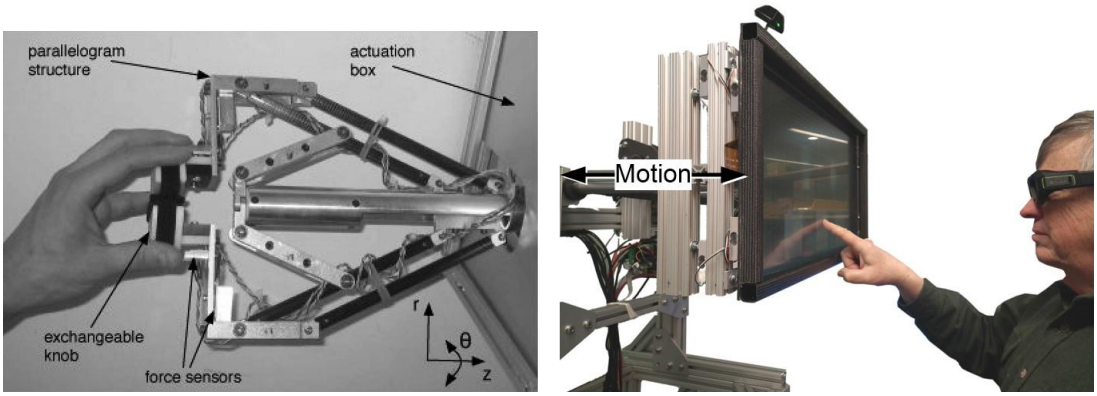
2.1.2 Stationäre Geräte mit weniger als sechs Freiheitsgraden

Alle bisher angeführten Objekte dienen hauptsächlich zur Navigation im Raum. Die Geräte in diesem Unterkapitel haben andere Anwendungsgebiete und kommen daher mit einer geringeren Anzahl an Freiheitsgraden aus.

Lambercy et al.[Lam+07] beschreiben eine Vorrichtung für die Greif- und Drehbewegungen der menschlichen Hand. Die Apparatur besteht aus einer Parallelogrammstruktur mit verschiedenen Aufsätzen für das Öffnen oder Schließen der Hand. Die Aufsätze können entweder mit der gesamten Hand oder mit einzelnen Fingern bedient werden. Die Struktur selbst kann um die eigene Achse rotieren und deckt so die Drehbewegungen ab. Diese zwei Freiheitsgrade reichen aus, um die meisten Handbewegungen zu erfassen. Sensoren messen die jeweiligen Bewegungen und geben die Daten weiter, das haptische Feedback wird mit Hilfe von Motoren bewerkstelligt. Dieses Gerät findet seinen Einsatz bei der Rehabilitation nach einem Schlaganfall. Die Kombination aus virtuellen Übungseinheiten und dem Gerät ermöglicht eine große Anzahl unterschiedlicher Trainingsabläufe. So kann zuerst die Handmotorik selbst trainiert werden, bevor in einer späteren Aufgabe das Benutzen eines Türknaufs geübt wird. Das Gerät ist in Abbildung 4a als Trainingsaufbau für die Bedienung eines Türknaufs mit zwei Freiheitsgraden dargestellt. Hierbei wird das Zugreifen und das anschließende Drehen der Hand geübt. Die Basisstruktur, bestehend aus zwei Parallelogrammen, ist in dieser Abbildung ebenfalls gut ersichtlich. Trotz der vielseitigen Aufsätze bleibt das Gerät, im Gegensatz zu dem Ansatz dieser Arbeit, auf die Hände beschränkt. Weiters können nur Szenarien dargestellt werden, in denen sich der Anwender bzw. die Anwenderin nicht frei bewegen kann.

Die meisten haptischen Eingabegeräte bieten zwar viele Freiheitsgrade, bringen aber nur relativ wenig Kraft auf. Aus diesem Grund beschreiben Sinclair et al.[SPB13] einen 3D-Touchscreen auf einem Roboterarm mit einem Freiheitsgrad(*TouchMover*), der bis zu 230 N erzeugen kann. Ein 16 kg schwerer Holzwürfel besitzt laut Sinclair et al.[SPB13] eine Haftreibung von 76 N. Der Arm ist gleichzeitig Sensor und Aktor. Anwenderinnen und Anwender können durch Druck auf den Monitor mit den dargestellten Objekten

interagieren. Somit ist es ihnen möglich Objekte zu bewegen oder ihre Oberfläche mit dem Finger abzufahren. Im Gegensatz zu den anderen Geräten in Abschnitt 2.1.1 bewegt der TouchMover keinen virtuellen Zeiger im Raum. Stattdessen ist der Finger selbst der Cursor und die virtuellen Objekte können direkt beeinflusst werden. Die Beschränkung auf eine Achse und der stationäre Aufbau schließen den TouchMover als Alternative zu dem System dieser Diplomarbeit allerdings aus. Der TouchMover und seine Funktionsweise sind in Abbildung 4b abgebildet. Bei diesem Ansätzen ist der Nutzungsraum erneut auf die Reichweite des fest verbauten Geräts eingeschränkt.



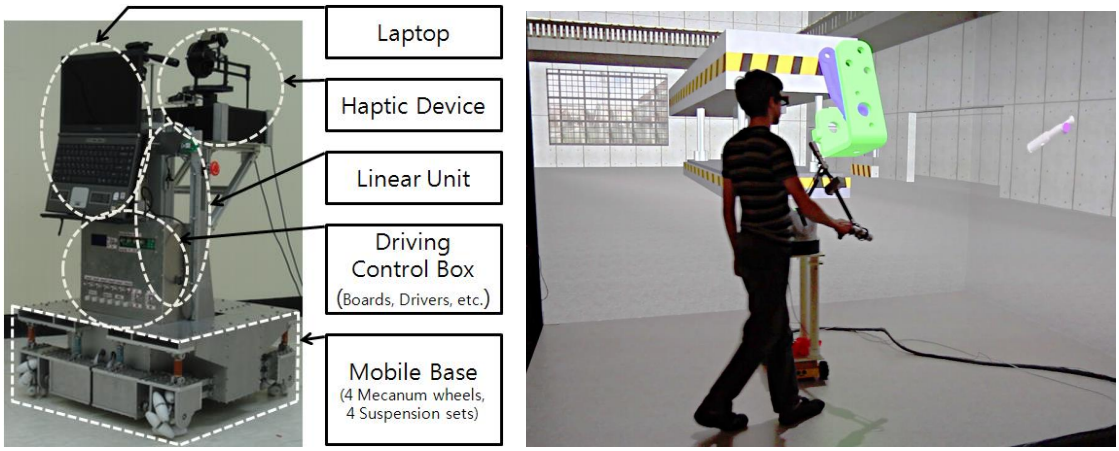
(a) (b)

Abbildung 4: Geräte mit weniger als 6 Freiheitsgraden:a) Türknauf mit zwei Freiheitsgraden[Lam+07], b) TouchMover mit einem Freiheitsgrad[SPB13].

2.1.3 Frei bewegliche Geräte

Alle zuvor beschriebenen Geräte haben eine gemeinsame Einschränkung: Aufgrund ihrer festen Montage haben sie einen beschränkten Nutzungsraum. Dieser lässt sich zwar durch eine Verlängerung des Arms oder der Fäden erweitern, bleibt aber immer begrenzt.

Han et al.[Han+08] und Pavlik et al.[PVL13] umgehen diese Limitierung mit Hilfe einer mobilen Basis. Diese Basis ist ein Roboter mit omnidirektionalen Rädern, der das darauf montierte Eingabegerät durch den Raum bewegt. Dieser *Mobile Haptic Interface* (*MHI*) genannte Ansatz ermöglicht es mit großen virtuellen Umgebungen zu interagieren ohne auf das haptische Feedback einer PHANTOM® oder Virtuose™ zu verzichten. Han et al.[Han+08] beschreiben ein MHI mit einem integrierten Laptop als Steuerungsmodul. Als Eingabegerät dient eine PHANTOM® Premium™. Pavlik et al.[PVL13] verwenden Virtuose™ 6D für die Benutzerinteraktion. Die Robotersteuerung übernimmt die Simulationsumgebung, die mittels *Virtual-Reality Peripheral Network* (*VRPN*) Daten an den Mikrocontroller sendet. Beide Apparaturen können stabile Hindernisse darstellen indem die Räder blockieren. Eine Verwendung der Geräte ist weiterhin nur mit den Händen am Griffstück möglich. Abbildung 5 zeigt die beiden MHI-Systeme.



(a) (b)

Abbildung 5: Die MHI Systeme: a) MHI mit inkludiertem Laptop[Han+08], b) Einsatz bei einer Fertigungssimulation[PVL13].

Einen anderen Ansatz verfolgt Provancher[Pro14]. Anstatt bestehende haptische Geräte mobil zu machen werden frei bewegliche Geräte um haptisches Feedback erweitert. Ausgangslage bietet das STEM System[Six15], das kabellose Übertragung von Position und Rotation ermöglicht. Das Feedback wird durch drei bewegliche Plättchen am Griff erzeugt. Bewegen sich die Platten gleichzeitig in dieselbe Richtung, erwirkt das bei Benutzer und Benutzerinnen den Eindruck einer gerichteten Kraft in ebendieser Richtung. Gegengleiche Verschiebungen hingegen werden als Drehung empfunden. Die Simulation eines undurchdringlichen Widerstands ist allerdings nicht möglich. Provancher[Pro14] nennt dieses Prinzip *Reactive Grip*. Reactive Grip lässt sich auch auf andere nicht haptische Eingabegeräte anwenden. Im Gegensatz zu der Herangehensweise dieser Arbeit bleibt Reactive Grip erneut auf die Hände beschränkt. Die Funktionsweise der verschiebbaren Platten wird in Abbildung 6 erläutert.

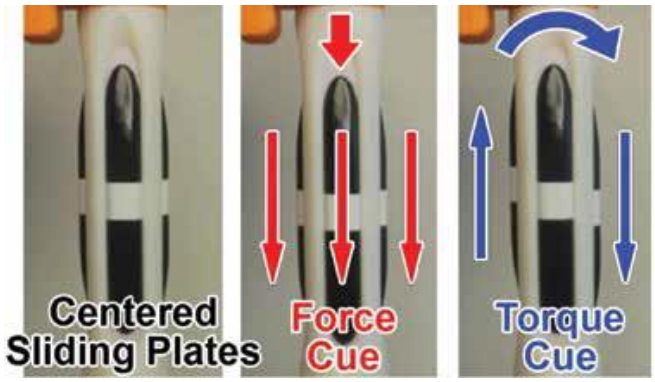
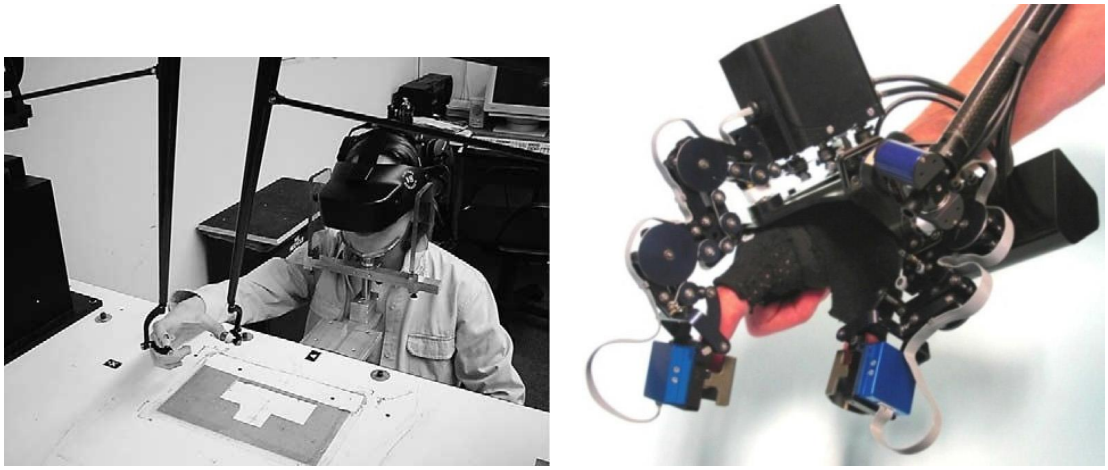


Abbildung 6: Funktionsweise von Reactive Grip.[Pro14]

2.2 Am Körper tragbare Geräte

Die Eingabegeräte aus Abschnitt 2.1 eignen sich gut für die Erkundung einer virtuellen Umgebung mit einem Cursor. Manchmal ist dieser Ansatz allerdings nicht ausreichend. Die Möglichkeiten zur direkten Interaktion mit Objekten sind sehr gering und oft nicht immersiv. Gegenstände können nicht in die Hand genommen sondern nur mittels Tastendruck am Zeiger befestigt werden. Weiters gibt es keine Mittel die Geometrie eines erfassten Objekts wahrzunehmen. Eine potentielle Lösung dieses Problems bieten Geräte, die direkt mit Körperteilen der Anwenderin bzw. des Anwenders verbunden sind. Je nach Größe eines solchen Systems ist Feedback für ein oder mehrere Körperteile möglich. Durch eine Kombination mehrerer Geräte zu einer Art Anzug wäre sogar Feedback für den gesamten Körper möglich. Im Gegensatz zu dem Ansatz dieser Diplomarbeit wäre dieser Anzug vermutlich schwer und würde die Bewegungsfreiheit deutlich einschränken.

Atkins et al.[AJK03] beschreiben ein System aus zwei PHANTOM® 3D, die an den Fingerspitzen von Daumen und Zeigefinger angebracht sind. Dadurch lassen sich die Positionen der Finger direkt in die virtuelle Welt übertragen, wodurch Greifgesten möglich sind. Das Feedback wird für jeden Finger separat berechnet und über die angebrachten PHANTOM®s übermittelt. So können Gegenstände nicht nur erfasst, sondern auch abgetastet werden. Durch die starre Montage der PHANTOM® Geräte lässt sich auch ein Widerstand beim Bewegen der Objekte abbilden. Der Aufbau ist in Abbildung 7a zu sehen. Atkins et al.[AJK03] verwenden diese Konstruktion, um die Zusammenhänge zwischen visuellen und gefühlten Distanzwahrnehmungen zu untersuchen. Fontana et al.[Fon+13] erwähnen ebenfalls ein Exoskelett für Daumen und Zeigefinger. Dieses liefert im Gegensatz zu dem Ansatz von Atkins et al.[AJK03] an mehreren Stellen pro Finger haptisches Feedback. Um das Gewicht zu reduzieren, wird der Impuls nicht direkt an den Fingergliedern, sondern am Handrücken erzeugt und mittels Stahlkabeln und Umlenkrollen weitergeleitet. Die Kabel dienen auch gleichzeitig dazu, die Stellung der Fingerglieder zurückzumelden. Dadurch können präzisere Greifbewegungen und detaillierteres Oberflächenfeedback erreicht werden. Abbildung 7b zeigt, wie die Motoren mit den Fingergliedern verbunden sind. Der Nachteil beider Ansätze besteht darin, dass das haptische Feedback nur für Daumen und Zeigefinger verfügbar ist. Durch die direkte Befestigung an den Fingern ist die Bewegungsfreiheit des Anwenders bzw. der Anwenderin stark eingeschränkt. Ebenso ist es nicht möglich die Oberflächenbeschaffenheit der dargestellten Objekte zu simulieren. Abbildung 7 stellt die beiden Geräte einander gegenüber.



(a) (b)

Abbildung 7: An Fingern befestigte Geräte: a) Tracking von Daumen und Zeigefinger mittels zwei PHANTOM® 3D[AJK03], b) Exoskelett für Daumen und Zeigefinger[Fon+13].

Die zuvor erwähnten Systeme eignen sich für die Interaktion mit eher leichten Objekten. Schwere Gegenstände sind aber nicht nur an der Hand, sondern am ganzen Arm spürbar.[Fri+05] Daher schlagen Frisoli et al.[Fri+05] ein Exoskelett für einen ganzen Arm vor. Dieses Gerät besitzt fünf Freiheitsgrade, wobei drei davon für das Kugelgelenk der Schulter verwendet werden. Für vier der Freiheitsgrade ist Force Feedback verfügbar, der fünfte dient nur zur Lagebestimmung. Die Gelenke sind wie in [Fon+13] durch Stahldrähte mit einer zentralen Stelle verbunden. Aufgrund der Masse von 11 kg ist das Gerät zusätzlich am Boden befestigt. Daumen und Zeigefinger können optional mit dem Skelett verbunden werden um Greifbewegungen zu gestatten. Frisoli et al.[Fri+05] gelingt es mit dieser Apparatur kontinuierliche Kräfte von 50 N zu erzeugen. Das Exoskelett ist in Abbildung 8 ersichtlich. Da das Exoskelett an einer festen Position montiert ist, ist es der Benutzerin bzw. dem Benutzer nicht möglich, sich komplett frei in der virtuellen Umgebung zu bewegen. Ebenso ist die Simulation von Oberflächen nicht möglich. Durch das relativ hohe Gewicht des Geräts, ist eine längere Verwendung unkomfortabel.

Abbildung 8: Ein Exoskelett für den Arm.[Fri+05]

Manche Szenarien setzen eine gewisse Bewegungsfreiheit voraus, die die oben genannten Geräte nicht bieten können. Dazu zählen virtuelle Welten, in denen der Benutzer bzw. die Benutzerin die Umgebung erkunden und dabei seinen bzw. ihren Oberkörper drehen kann. Ebenso wäre ein Exoskelett das haptisches Feedback am ganzen Körper bietet sehr komplex. Lindeman et al.[Lin+04] beschreiben aus diesem Grund eine Weste die mit 16 kleinen Vibrationsmodulen ausgestattet ist. Diese Module sind mit einer zentralen, batteriebetriebenen Steuereinheit verbunden, die via Funk Feedbackbefehle empfängt. Diese *TactaVest* selbst besteht aus mehreren Teilen um eine möglichst exakte Passform zu bieten. Damit können Lindeman et al.[Lin+04] mit relativ geringem Gewicht und fast ohne Einschränkung der Bewegungsfreiheit eine Vielzahl an Empfindungen simulieren. Die Rückansicht der TactaVest mit den Markierungen der Vibrationsmotoren ist in Abbildung 9a zu sehen. Ein ebenfalls auf Vibration basierendes Gerät stellen Gallotti et al.[GRS11] vor. Das Ziel ist, ein leichtes, kabelloses System zu entwickeln, das die Interaktion mit virtuellen Benutzeroberflächen ermöglicht. Dazu entwickeln Gallotti et al.[GRS11] einen Handschuh mit einem Vibrationsmotor an der Spitze des Zeigefingers(*v-Glove*). Das Modul wird über einen Arduino Mikrocontroller gesteuert, der gleichfalls drahtlos Befehle empfängt. Die Spitze des v-Gloves ist mit einem reflektierenden Klebeband markiert und kann dadurch ohne aktiven Emitter getrackt werden. Berührt der virtuelle Finger die Benutzeroberfläche, wird dies durch eine Vibration mitgeteilt. Abbildung 9b zeigt den v-Glove inklusive des Arduino Mikrocontrollers und des drahtlosen Empfängers. Der Nachteil dieser beiden Geräte ist, dass sie nur Vibrationen verwenden um haptisches Feedback zu erzeugen. Dadurch ist es nicht möglich die Beschaffenheit von Oberflächen zu simulieren.

(a) (b)

Abbildung 9: Kleidungsstücke mit haptischem Feedback: a) TactaVest[Lin+04], b) v-Glove[GRS11].

Da Vibrationen keine gerichteten Kräfte simulieren können, schlagen Minamizawa et al.[MF07] ein anderes tragbares Konzept vor. An den Fingerspitzen von Daumen und Zeigefinger wird ein Gummiband fixiert, das über zwei Rollen an der Oberseite der Finger bewegt werden kann. Die beiden Rollen können das Band spannen oder in eine Richtung ziehen. Je nachdem empfinden Anwenderinnen und Anwender einen Druck oder einen Zug. Damit können die Autoren der Arbeit die Trägheit eines gehaltenen Objekts simulieren. Diese Empfindungen sind laut Minamizawa et al.[MF07] ausreichend um den Eindruck von Schwerkraft und Masse an den Benutzer zu übermitteln ohne eine tatsächliche Kraft auf den Arm auszuüben.

Abbildung 10: Prototyp des GhostGlove.[Min+08]

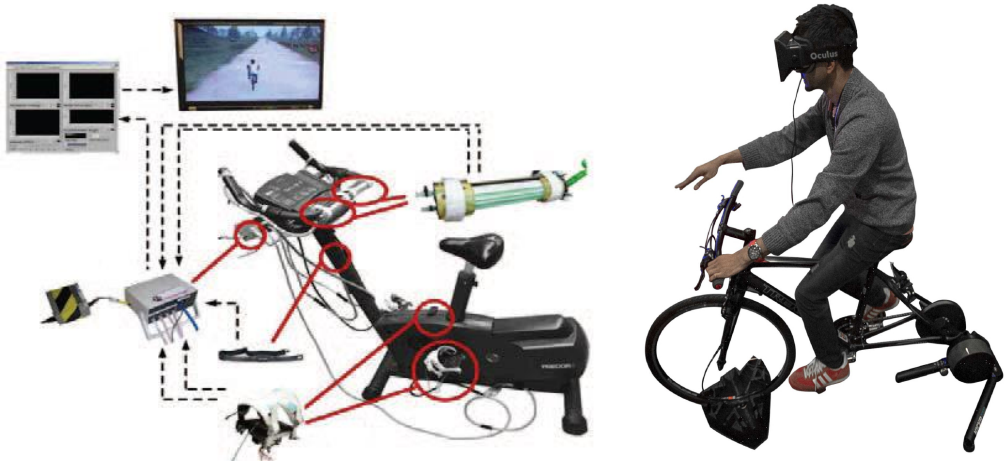
Die Weiterentwicklung dieses Ansatzes ist der ebenfalls von Minamizawa et al.[Min+08] vorgestellte *GhostGlove*. Er besteht aus je einem Band für jeden Finger und einem weiteren auf der Handfläche. Dadurch lassen sich unterschiedlich starke Berührungspunkte korrekt

auf der ganzen Hand darstellen. Im Gegensatz zum System dieser Diplomarbeit, ist das haptische Feedback bei diesem Ansatz auf die Hände beschränkt. Weiters ist es nicht möglich, unterschiedliche Oberflächen darzustellen. Der GhostGlove ist in Abbildung 10 ersichtlich.

2.3 Simulatoren

Geräte aus den vorherigen Abschnitten lassen sich gut für unterschiedliches Feedback auf einem bestimmten Körperbereich einsetzen. Die Apparaturen in dieser Sektion sind hingegen auf je ein spezielles Szenario ausgelegt, wodurch sie sich von dieser Arbeit unterscheiden. Dabei versuchen sie, sämtliche Eindrücke dieses Anwendungsfalls so akkurat wie möglich widerzuspiegeln.

Ein Einsatzgebiet ist die Steigerung der Motivation in Fitness-Spielen. Bolton et al.[Bol+14] und Ranky et al.[Ran+10] beschreiben jeweils Ansätze, das Training mit einem stationären Fahrrad ansprechender zu gestalten. Zu den ihrer Meinung nach wichtigsten Punkten zählt ein hoher Grad an Immersion. Benutzer und Benutzerinnen sollen das Gefühl haben, tatsächlich eine Fahrradtour zu unternehmen. Ranky et al.[Ran+10] erklären ein System, das herkömmliche Fahrradtrainer um eine virtuelle Umgebung erweitert. Dazu werden spezielle Griffe und Pedale verwendet, die sowohl die aufgewendete Kraft messen, als auch haptisches Feedback liefern. Die Geschwindigkeit wird nicht über das Fahrrad selbst, sondern ebenfalls über die Pedale registriert. Dadurch kann die virtuelle Umgebung erkennen, wenn eine Lenkbewegung durchgeführt wird oder die Fahrerin bzw. der Fahrer sich in einer Seitenlage befindet, und den Avatar entsprechend anpassen. Die Griffe können Druck auf die Handflächen ausüben um etwaige Erschütterungen zu simulieren. Die Pedale besitzen einen Vibrationsmotor, der bei Verlassen der Strecke zum Einsatz kommt, und Motoren, um die seitliche Neigung zu ändern. Die Neigungsänderung bewirkt den Eindruck einer Seitenlage in der Kurvenfahrt. Bolton et al.[Bol+14] zeigen eine VR-Version des ATARI Klassikers Paperboy. Sie erweitern ein herkömmliches Fahrrad um eine spezielle Halterung, die die Fahrtgeschwindigkeit an den PC übermittelt. Mittels Microsoft Kinect[Mic15a] wird der Oberkörper erfasst und an den Avatar übertragen. Im Gegensatz zu Ranky et al.[Ran+10] setzen Bolton et al.[Bol+14] ein *Head-Mounted Display* (*HMD*) von OCULUS VR[OCU15c] ein, um die Immersion zu erhöhen. Der erste Prototyp liefert noch kein Feedback zurück, Bolton et al.[Bol+14] zählen aber die Erhöhung des Tretwiderstandes als eines der nächsten Ziele. Damit ließen sich Veränderungen in der Geländesteigung sowie Hindernisse simulieren. Die beiden Trainingsgeräte sind in Abbildung 11 dargestellt.



(a) (b)

Abbildung 11: Fahrradsimulatoren: a) Fahrradsystem von Ranky et al.[Ran+10], b) Paperboy Simulators inklusive HMD[Bol+14].

Ein weiterer Simulator für ein Computerspiel wird von Humphries[Hum11] erwähnt. Hierbei handelt es sich um einen begehbaren Simulator für *First-Person-Shooter* (*FPS*) mit haptischem Feedback. Der Simulator besteht aus einer Kuppel, deren Zentrum ein omnidirektionales Laufband bildet. Dieses ermöglicht dem Spieler bzw. der Spielerin, sich in jede Richtung zu bewegen. Das an die Wand der Kuppel projizierte Bild des Spiels folgt dabei der Rotation. Horizontale Bewegungen wie Springen oder Ducken werden durch mehrere Microsoft Kinects erfasst. In dem Artikel wird der Simulator für das Spiel Battlefield 3[Ele15] verwendet. Das haptische Feedback wird mittels Paintball-Markierern bewerkstelligt. Diese sind in drei Gruppen zu je vier Markierern um das Zentrum aufgestellt und liefern Rückmeldung über erlittene Treffer.

Hornyak[Hor10] berichtet von einem Simulator für Beschleunigungskräfte des Max-Planck-Instituts für biologische Kybernetik[Max15]. Dieser basiert auf einem 5 Meter hohen KUKA Industrieroboter[KUK15]. Anstelle eines Greifers ist ein Sitz mit drei Monitoren befestigt. Dieser Sitz bildet das Cockpit eines Ferrari F2007 Formel 1 Autos. Mithilfe eines Force Feedback Lenkrads und Pedalen können Testpersonen das Auto auf einer virtuellen Version der Strecke von Monza steuern. Bei Beschleunigungen des virtuellen Boliden verändert der Roboter seine Lage, sodass die Schwerkraft in gleichem Maße auf den Körper wirkt. Zusätzlich dreht sich der Roboter analog zu dem virtuellen Rennwagen. Die Reaktionszeit des Arms beträgt nur 400 ms, wodurch eine starke Immersion möglich ist. Der Arm ist ebenfalls als Simulator für Flugzeuge oder Helikopter einsetzbar.[Tüb10a; Tüb10b]

Eine andere Form eines Simulators beschreiben Shin et al.[Shi+12]. Ihre Apparatur dient zur Nachbildung des Öffnungsverhaltens verschiedener Kühlschranktüren. Dies stellt eine kostengünstige Alternative zu der Erstellung von Prototypen dar, da nicht für

jede Türe ein neues Modell erstellt werden muss, sondern einfach die Parameter des Simulators angepasst werden. Das Gerät besteht aus einem Gestell mit den Dimensionen 60x60x140 cm. In 120 cm Höhe ist ein beweglicher Arm angebracht, an dessen Ende verschiedene Griffe montiert werden können. Der Widerstand des Arms wird durch einen Motor im Zentrum des Gestells reguliert. Ein wichtiger Teil des Geräts ist das Modell, das für die Berechnung des Widerstands verwendet wird. Dieses berücksichtigt die Anziehungskraft der magnetischen Dichtung, ihre Elastizität, die Trägheit der Türe und den Widerstand der Scharniere. Je nach Öffnungswinkel und aufgebrachter Kraft liefert dieses Modell einen entsprechenden Wert an den Motor. Das System wird durch die Darstellung einer virtuellen Küche auf einem 3D-Monitor komplettiert. Shin et al.[Shi+12] ist es damit möglich, ein annähernd realistisches Öffnungsgefühl zu erzeugen. Ein 3D-Modell des Setups ist in Abbildung 12 zu sehen.

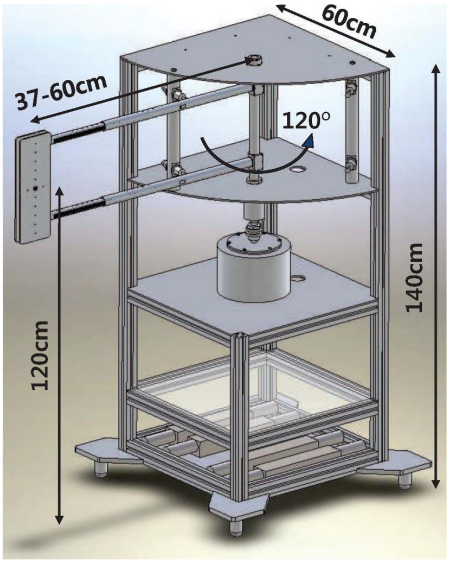


Abbildung 12: 3D-Modell des Kühlschranktür-Simulators.[Shi+12]

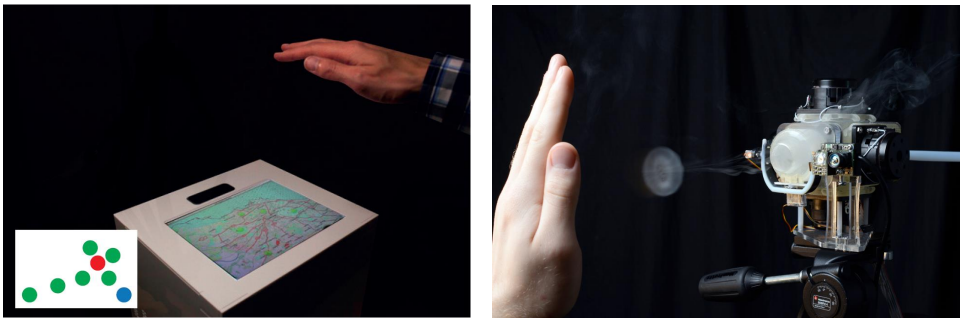
2.4 Andere Feedbacksysteme

Dieser Abschnitt enthält Systeme, die zur Abbildung verschiedener Empfindungen verwendet werden können. Diese Geräte müssen nicht gehalten oder angelegt werden und sind auch nicht auf ein spezielles Szenario beschränkt. Der Ansatz dieser Arbeit fällt ebenfalls in diese Kategorie.

Carter et al.[Car+13] beschreiben haptische Eindrücke, die durch Ultraschall erzeugt werden. Eine Anordnung aus vielen Ultraschall-Emittern sendet dabei gezielte Impulse aus, die von der Haut als Vibration oder Druck wahrgenommen werden. Potentielle Ziele, wie etwa einzelne Finger oder der Handballen, werden durch einen Tiefensensor lokalisiert. Das Feld aus Emittern ist an einen festen Platz gebunden und kann mehrere Druckpunkte

in bis zu 30 cm Höhe erzeugen. Das Gerät eignet sich daher gut, um stationären Anzeigen ein dreidimensionales Feedback zu verleihen. Der Vorteil der Ultraschall-Vibrationen ist, dass sie weder sichtbar noch hörbar sind. Allerdings lässt sich mit diesem Gerät, im Unterschied zu dieser Arbeit, kein tatsächliches Hindernis darstellen. In Abbildung 13a ist ein solches Display mit haptischem Feedback durch Ultraschall zu sehen. Die Punkte in dem weißen Feld stellen zu diesem Zeitpunkt emittierten Feedbackpositionen dar. Die unterschiedlichen Farben symbolisieren verschiedene Intensitäten.

Ein ebenfalls unsichtbares haptisches Feedback erwähnen Sodhi et al.[Sod+13]. Anstelle von Ultraschall setzt der *AIREAL* genannte Ansatz auf Luftwirbel. Die ringförmigen Wirbel werden von Geräten mit einer flexiblen Düse ausgestoßen. Die Emissionsrate kann dabei variiert werden, wodurch das Feedback unterschiedlich wahrgenommen wird. Luftwirbel besitzen eine höhere Reichweite als Ultraschall-Vibrationen. Die einzelnen Geräte können entweder alleine oder in einem Verbund agieren. Im Einzelmodus verwendet ein Gerät die integrierte Tiefenkamera, um die Handposition zu erkennen. Je nach Anwendung werden zu bestimmten Zeitpunkten Luftwirbel erzeugt, die auf der Hand als Druck wahrgenommen werden. Mehrere AIREAL Geräte können auch zusammen verwendet werden, um möglichst viele Positionen im Raum abzudecken. Damit lassen sich zum Beispiel am Kopf vorbeifliegende Vögel simulieren. Es ist aber auch möglich die Grenzen virtueller Objekte darzustellen und diese auch zu verschieben. Ähnlich zu [Car+13] lassen sich aber auch mit diesem System keine festen Objekte erzeugen. Abbildung 13b zeigt dieses System bei der Emission eines Luftwirbels. Zur besseren Sichtbarkeit wurde Rauch verwendet.



(a) (b)

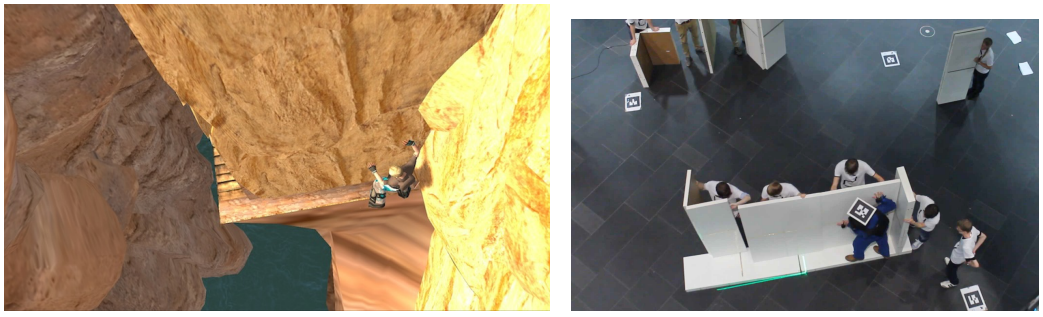
Abbildung 13: Systeme mit unsichtbarem haptischen Feedback: a) Haptisches Feedback durch Ultraschall[Car+13], b) Haptisches Feedback durch Luftwirbel[Sod+13].

Die beiden Systeme eignen sich gut für virtuelle Umgebungen in denen Benutzerinnen und Benutzer bewegt werden ohne dass sie sich in der physischen Welt bewegen. Bei Szenarien in denen die Bewegung in beiden Welten synchron stattfindet, ergeben sich allerdings Probleme. So können Anwender und Anwenderinnen beispielsweise durch virtuelle Wände hindurchgehen, wodurch die Position des Avatars nicht mehr übereinstimmt.[Ins01] Lindeman[Lin99] definiert dafür das Konzept des passiven haptischen Feedbacks.

„Passiv-haptische ‚Geräte' sind physische Objekte, die das Feedback nur durch ihre Form, Textur oder andere inhärente Eigenschaften liefern. Im Gegensatz zu aktiv-haptischen Feedbacksystemen, wird das von passiv-haptischen Geräten gebotene Feedback nicht von einem Computer gesteuert." [Lin99, S. 19 f.]

Insko[Ins01] zeigt, dass sich passives haptisches Feedback positiv auf die Präsenzwahrnehmung auswirkt und das Erlangen von räumlicher Orientierung erleichtert. Dazu wurden die relevanten Teile der virtuellen Umgebung physisch nachgebaut und konnten von den Benutzerinnen und den Benutzern erkundet werden. Im Gegensatz zu den meisten stationären Systemen kommt der haptische Eindruck nicht auf die Person zu, sondern umgekehrt. Dadurch sind nur Szenarien möglich, in denen die Bewegungen des Avatars mit jenen in der Realität übereinstimmen. Das Fahren mit einem Auto oder das Fliegen durch eine Landschaft ohne tatsächliche Bewegung würde die Kopplung von haptischem und virtuellem Objekt auflösen.[Ins01] Ein weiterer Nachteil dieses Vorgehens ist, dass die reellen Platzanforderungen mit der Größe der virtuellen Welt wachsen.

Cheng et al.[Che+15] beschreiben ein System namens *TurkDeck*, das die Tatsache ausnützt, dass Benutzer und Benutzerinnen nur mit einem kleinen Bereich der virtuellen Welt tatsächlich interagieren können. TurkDeck setzt dazu auf menschliche Aktorinnen und Aktore (*actuators*), die generische Requisiten bewegen und so die physische Welt dynamisch anpassen. Die Requisiten werden üblicherweise *props* oder *physical props* genannt. Die Bewegung der Person wird von Sensoren registriert und auf den Avatar übertragen, wodurch er sich in der Anwendung bewegt. Dabei berechnet das System etwaige Veränderungen der Landschaft und gibt den Aktoren mittels Lasermarkierungen und akustischen Signalen Anweisungen, die props neu zu positionieren. TurkDeck versucht, die weitere Bewegung des Avatars vorherzusagen, um die Reaktionszeit der Aktoren zu berücksichtigen. Im Gegensatz zu den zuvor beschriebenen passiv-haptischen Systemen, können durch das Recycling der nicht mehr benötigten Umgebungsabschnitte große Welten auf einer kleineren Fläche abgebildet werden. Weiters ist es auch möglich, nur den Avatar zu bewegen, da die Aktoren die Bauteile um den Anwender bzw. die Anwenderin bewegen können. Obwohl der Platzbedarf durch entsprechend aufgebaute virtuelle Welten minimiert werden kann, ist er dennoch deutlich höher als bei dem in dieser Arbeit vorgeschlagenen System. Die Verwendung von menschlichen Aktoren erhöht außerdem den logistischen Aufwand, da alle Personen zeitlich verfügbar sein müssen. Die Funktionsweise des Systems ist in Abbildung 14 anhand einer virtuellen Klippe dargestellt. Bewegt sich der Benutzer zur Kante, befiehlt das System den Aktoren, die Wand weiter aufzubauen. Dazu werden bereits verlassene Abschnitte verwendet, die dadurch abgebaut werden.[Che+15]



(a) (b)

Abbildung 14: Beispiel einer Klippe in TurkDeck: a) Sicht des Benutzers, b) Anordnung der props.[Che+15]

Cheng et al.[Che+15] haben TurkDeck absichtlich nicht komplett automatisiert, es gibt aber Ansätze, mit denen die Aufgabe der Aktoren durch Roboter abgebildet werden kann. McNeely[McN93] erstellt zwei Theorien über robotikbasiertes Feedback. Zum einen werden *Robitic Shape Displays* (*RSDs*) als Roboter definiert, die verschiedene Formen eines Objekts darstellen können. Dazu besitzt der Roboter ein Repertoire an Ecken, Kanten und Flächen, die abhängig von der Ausrichtung des virtuellen Objekts dargeboten werden. Die Lage des virtuellen Gegenstands wird relativ zur getrackten Position des Fingers oder der Hand berechnet. Der Roboter kann nicht nur die Oberfläche präsentieren, sondern auch Kraft ausüben oder seine Position verändern. Zum anderen beschreibt McNeely[McN93] *Roboxel*. Das ist ein System aus kleinen, autonom versorgten Robotern, die ihre Position und Orientierung selbst verändern können, um so virtuelle Gegenstände in der Realität nachzubauen. Im Gegensatz zum System dieser Diplomarbeit, beschränkt sich das haptische Feedback auf das Ertasten von Objekten. Es ist daher beispielsweise nicht möglich die Kollision mit Passanten auf einem Gehweg zu simulieren.

Hirota et al.[HH95] versuchen ein RSD mit von Roboxel inspirierten Elementen zu generieren. Dazu wird ein System beschrieben, das aus einem Feld von Metallstäben besteht. Jeder Metallstab besitzt einen eigenen Stellmotor, der festlegt, wie weit der Stab hervorsteht. Über den Stäben ist eine Schaumstoffschicht angebracht, damit die einzelnen Stäbe als eine Oberfläche erkennbar sind. Das Gerät selbst kann in zwei Richtungen gekippt werden. Diese Bewegung wird an die virtuelle Umgebung weitergegeben und verändert somit den Blickwinkel der Kamera auf ein Objekt. Die Oberfläche des Objekts wird mithilfe der Stäbe nachgebildet. Dadurch können beliebige Oberflächen nachgestellt werden, unter anderem auch abgerundete Ecken oder Löcher. Es ist ebenfalls möglich, aufgewendete Kraft als einen Lastanstieg in den einzelnen Motoren zu messen. Ein Nachteil dieses Ansatzes ist, dass die Größe der darstellbaren Objekte auf die Größe des Geräts beschränkt ist. Da dieses Gerät die Bewegung der Hand nicht verfolgen kann, ist es zum Beispiel nicht möglich eine ganze Wand darzustellen. Abbildung 15 zeigt einen Prototypen dieses Geräts.

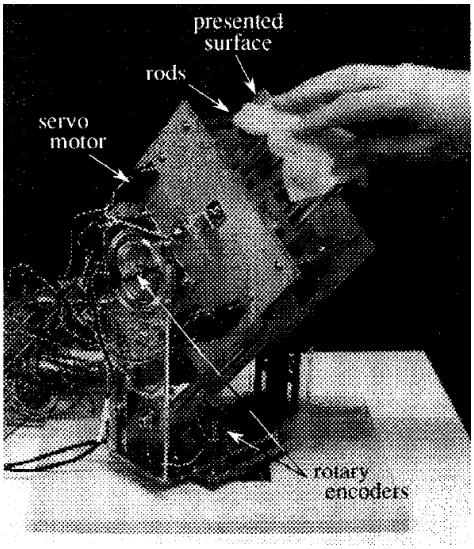
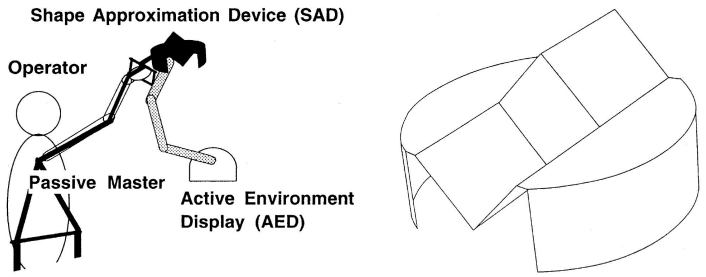


Abbildung 15: Gerät zur Simulation verschiedener Oberflächen.[HH95]

Ein weiteres RSD ist jenes von Tachi et al.[Tac+94]. Es besteht aus einem Roboter mit einem speziellen Endeffektor. Der Endeffektor wird *Shape Approximation Device* (*SAD*) genannt und besitzt Ecken, Kanten, Flächen sowie Rundungen. Es sind sowohl konvexe als auch konkave Ecken und Kanten vorhanden. Das System verfolgt die Position des Zeigefingers mithilfe eines am Arm angebrachten *passive master* mit 7 Freiheitsgraden. Das ist ein Roboterarm, dessen Gelenkwinkel nicht aktiv durch die Servomotoren, sondern durch die Anwenderin bzw. den Anwender geändert werden. Durch Auslesen der Motorwerte kann die Position des Endeffektors und somit auch die der Hand errechnet werden. Das VR-Programm eruiert anschließend den Ort des am nächsten gelegenen virtuellen Objekts und die dazu passende Fläche des SAD. Der Roboter dreht anschließend das SAD in die gewünschte Lage. Anwender und Anwenderinnen können somit die Oberfläche des virtuellen Objekts erforschen und Kraft darauf auswirken. Der Roboter verfolgt den Zeigefinger, wenn er sich entlang des virtuellen Objekts bewegt und passt gegebenenfalls die dargebotene Stelle des SAD an. Zusätzlich können auch Trägheit und Widerstand des Objekts durch eine entsprechende Bewegung des Roboters simuliert werden. Durch die Verbindung mit dem passive master wird die Bewegungsfreiheit eingeschränkt. Beispielsweise ist es der Benutzerin bzw. dem Benutzer nicht möglich sich zu drehen. Weiters kann das haptische Feedback mit keinem Körperteil außer der verbundenen Hand erfahren werden. Skizzen des Systems sowie des SAD sind in Abbildung 16 dargestellt.



(a) (b)

Abbildung 16: *Robitic Shape Display* (*RSD*) von Tachi et al.[Tac+94]: a) Tracking der Hand, b) *Shape Approximation Device* (*SAD*).[Tac+94]

Ein ebenfalls auf der RSD-Theorie basierendes System wird von Yokokohji et al.[YHK96] vorgestellt. Das System wird *What You can See Is What You can Feel* (*WYSIWYF*) Display genannt. Es besteht aus einem Roboterarm, der fühlbare Gegenstände bereitstellt und einem Bildschirm mit integrierter Kamera. Die Kamera registriert am Roboter angebrachte Markierungen, wodurch die relative Position des Bildschirms zum virtuellen Gegenstand errechnet wird. Die Kamera zeichnet auch die Hand des Benutzers bzw. der Benutzerin auf. Die Hand wird mit Hilfe von farbbasierter Bildfreistellung aus dem aufgenommenen Bild extrahiert und auf dem Bildschirm mit der virtuellen Szene zusammengefügt. Der Roboterarm bleibt für Anwenderinnen und Anwender unsichtbar. Das System bietet einen *Surface Display Modus*, dessen Funktion der von Tachi et al.[Tac+94] ähnelt. Anstelle eines Passive Masters kommt allerdings ein Infrarot-Emitter an der Fingerspitze zum Einsatz. Es bietet aber auch einen zweiten Modus, der ohne ein Tracking der Hand auskommt. In diesem Modus ist das zu erfassende Objekt und die damit durchzuführende Interaktion fix vorgegeben. So soll beispielsweise ein Würfel mit einem daran befestigten Knauf verschoben werden. Am Roboterarm ist dazu eine Platte mit einem Knauf angebracht. Wird genügend Kraft aufgewendet, verschiebt der Roboter die Platte in die gewünschte Richtung. Trägheit und Widerstand sind auch hier frei einstellbar. Auch bei diesem System ist das haptische Feedback nicht mit allen Körperteilen erfahrbar, da nur ein Zeigefinger getracked wird. Zusätzlich ist es dem Anwender bzw. der Anwenderin nicht möglich sich mittels Bewegung in der physischen Welt durch die virtuelle Welt zu navigieren. Eine Darstellung des Funktionsprinzips ist in Abbildung 17 sichtbar.

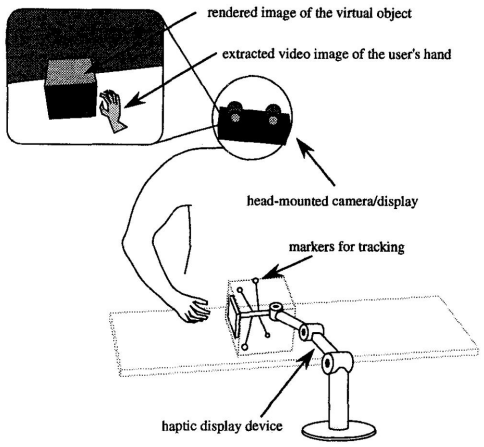


Abbildung 17: Prinzip des WYSIWYF Displays.[YHK96]

Das in dieser Arbeit vorgestellte System ähnelt dem Prinzip der RSD. Allerdings beschränkt sich die Anwendung nicht auf den Zeigefinger, sondern bietet abhängig vom Anwendungsfall Feedback für alle Körperregionen. Weiters wird die Idee des requisitenbasierten Feedbacks aufgegriffen, indem der Roboterarm aus mehreren props wählen kann und diese zum passenden Zeitpunkt präsentiert. Der Platzbedarf wird in diesem System durch die Verwendung einer Bewegungsplattform weiter reduziert.

KAPITEL 3

Konzeptentwicklung

Dieses Kapitel beschreibt die technischen Grundlagen des vorgeschlagenen Systems. Das Versuchssetup kombiniert verschiedene bestehende Geräte, um ein möglichst immersives VR-Erlebnis zu erzeugen. Abschnitt 3.1 enthält die Funktionalitäten jeder verwendeten Komponente und warum sie ausgewählt wurde. Abschnitt 3.2 gibt einen Überblick über die reinen Softwarekomponenten, die für Berechnungen und die Integration der einzelnen Teile eingesetzt werden. Weiters enthält er eine Beschreibung der Grundprinzipien der Robotikumgebung OpenRAVE und die Vorteile der Verwendung einer solchen Middleware.

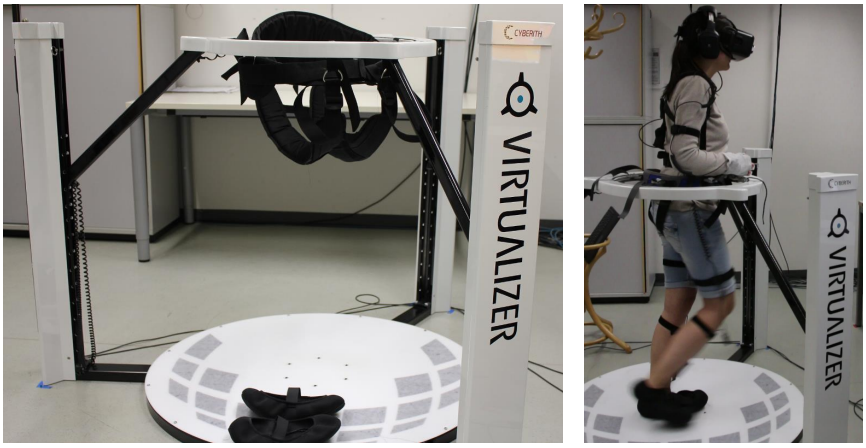
3.1 Hardwarekomponenten

Das erstellte Framework verwendet Hardwarekomponenten aus mehreren Bereichen. Abschnitt 3.1.1 beschreibt die eingesetzte Bewegungsplattform, mit deren Hilfe eine annähernd natürliche Fortbewegung in uneingeschränkten virtuellen Welten auf geringem Raum möglich ist. Abschnitt 3.1.2 erläutert die Funktionsweise des verwendeten HMD, während Abschnitt 3.1.3 das Tracking der Benutzerin bzw. des Benutzers schildert. Abschließend listet Abschnitt 3.1.4 den Roboterarm und die Spezifikation des PCs auf.

3.1.1 Fortbewegung

Ein wichtiger Bestandteil für eine glaubhafte virtuelle Welt ist die Möglichkeit sich darin zu bewegen. Gamepads oder haptische Eingabegeräte wie in Abschnitt 2.1 sind zwar leicht zu verwenden, bieten allerdings nicht das Gefühl, sich tatsächlich zu bewegen. Außerdem sind durch die Nutzung solcher Geräte die Hände des Anwenders bzw. der Anwenderin gebunden. Aus diesem Grund kommt der in [CH14] beschriebene *Virtualizer* zum Einsatz. Der Virtualizer ist eine omnidirektionale Bewegungsplattform, die zur Fortbewegung in virtuellen Welten dient. Er besteht aus einer runden Bodenplatte und einer Ringkonstruktion, die mittels dreier Steher an der Bodenplatte befestigt ist. Die

Oberfläche der Bodenplatte des Geräts besitzt eine geringe Reibung, sodass mit speziellen Überschuhen darüber geglitten werden kann. Die Benutzerin bzw. der Benutzer wird durch ein Gurtsystem in der Ringkonstruktion mittig über der Basis gehalten und kann so fast normal auf der Platte gehen. Abbildung 18 zeigt das Gerät, die Überschuhe und die Bewegung damit.



(a) (b)

Abbildung 18: Bewegungsplattform Virtualizer: a) Virtualizer und Überschuhe b) gehende Person.

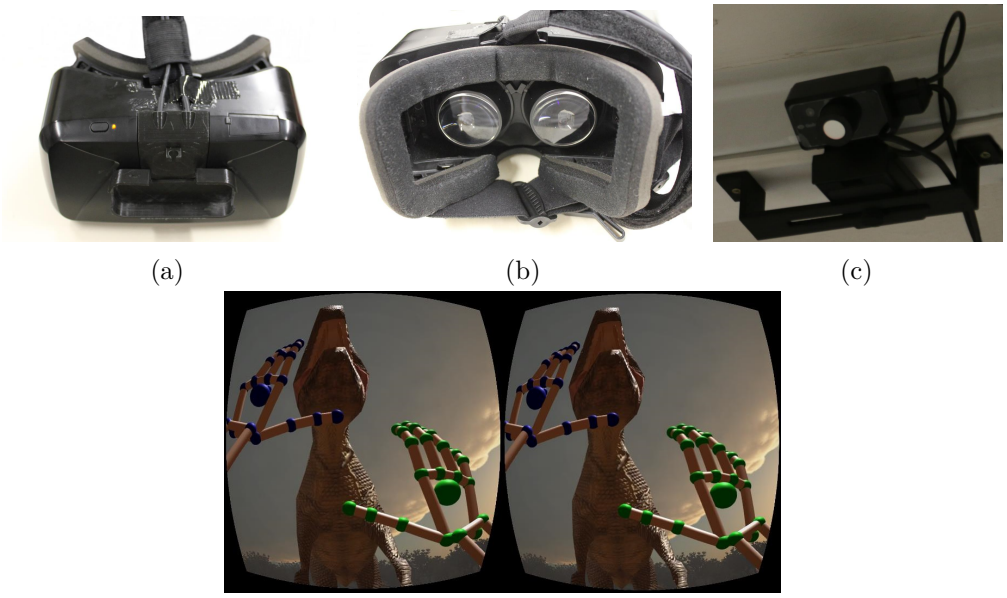
Sensoren in der Mitte der Platte registrieren die Bewegung der Füße und errechnen daraus die Bewegungsrichtung und -geschwindigkeit. Die Ringkonstruktion besteht aus einem starren äußeren und einem beweglichen inneren Ring. Der innere Ring ist durch das Gurtsystem mit der Hüfte verbunden und dreht sich bei Drehungen entsprechend mit. Diese Drehung wird ebenfalls registriert und als Körperorientierung an die Anwendung übergeben. Zusätzlich fließt sie in die Berechnung der Bewegungsrichtung ein, sodass die Gehrichtung immer relativ zur Körperrotation ist. Geht der Benutzer oder die Benutzerin vorwärts über die Platte und dreht sich dabei, ändert sich nur die Körperorientierung, die erkannte Bewegungsrichtung bleibt *vorwärts*. Die Ringkonstruktion kann vertikal bewegt werden, wodurch auf die aktuelle Hüfthöhe geschlossen werden kann. Veränderungen der Höhe können dabei als Springen oder Ducken erkannt werden. Die Höhe kann aber auch fixiert werden, wodurch auch sitzende Positionen, wie etwa beim Fahren eines Autos, möglich sind.

Der Virtualizer wird als Eingabegerät registriert und liefert Höhe, Orientierung, Gehrichtung und -geschwindigkeit als analoge Werte. Diese können entweder direkt in kompatiblen Anwendungen verarbeitet oder mittels spezieller Software als Tastenanschläge emuliert werden.[CH14; Cyb15]

3.1.2 Visualisierung

Ebenso wichtig wie die Fortbewegung ist eine immersive Darstellung der virtuellen Umgebung. Normale Bildschirme oder 3D-Monitore sind in Kombination mit 360° Drehungen nur bedingt einsetzbar. Zusätzlich muss der Blick stets auf den Monitor gerichtet bleiben, wodurch ein Umsehen in der Welt unmöglich ist. Ein HMD ist für diesen Anwendungszweck besser geeignet. Das in dieser Arbeit verwendete Gerät ist das *Oculus Rift Development Kit 2* (*DK2*) der Firma OCULUS VR[OCU15c]. Andere, zum Zeitpunkt der Konzeptionierung erhältliche, HMDs sind deutlich teurer als das DK2 und besitzen ein geringeres Sichtfeld. Das DK2 besitzt einen OLED-Bildschirm mit einer Auflösung von 1920x1080 Pixel. Dadurch ist für jedes Auge eine Auflösung von 960x1080 Pixel verfügbar. Die Bildauffrischungsrate liegt bei maximal 75Hz. Das dargestellte Bild besteht aus zwei Teilen, je einem pro Auge. Diese zwei Teilbilder sind so verzerrt, dass die zwei vor dem Display montierten Linsen die Ansicht korrekt entzerren können.

Die Bewegungen des Kopfes werden mit Gyroskopen, Magnetometern und Beschleunigungssensoren gemessen und an die Anwendung übertragen. Die Sensoren arbeiten mit einer Abtastrate von 1000 Hz, wodurch die Verzögerung möglichst gering gehalten wird. Eine zusätzliche Infrarotkamera beobachtet unter der Abdeckung des Geräts angebrachte Infrarotemitter, um die Position des Kopfes zu verfolgen. Abbildung 19 enthält die Komponenten des DK2, das Gerät selbst und die Infrarotkamera.[OCU15a; OCU15b; OCU15c]



(a) (b) (c)

(d)

Abbildung 19: *Head-Mounted Display* (*HMD*) *Oculus Rift Development Kit 2* (*DK2*): a) Außenseite mit Tiefensensorhalterung, b) Linsen zur Entzerrung des Bildes, c) Infrarotkamera zur Positioneserkennung, c) für VR verzerrtes Bild.

3.1.3 Körpertracking

Die letzte benötigte Komponente ist ein Trackingsystem, um den Körper korrekt in der virtuellen Welt abzubilden. Dies dient zum einen der Immersion, da Benutzerinnen und Benutzer ihre physischen Extremitäten durch das HMD nicht sehen können, zum anderen aber auch der Sicherheit. Sind die Positionen der einzelnen Gliedmaßen bekannt, kann dies in der Pfadberechnung des Roboters berücksichtigt werden, um Kollisionen zu vermeiden. Für die Realisierung eines solchen Trackings gibt es verschiedene Möglichkeiten, wie etwa optische Systeme oder am Körper angebrachte Sensoren. Optische Systeme haben die Nachteile, dass sie meist an festen Plätzen montiert sind, wodurch es zu Verdeckungen kommen kann, und dass die Genauigkeit des Trackings sehr stark vom Betrachtungswinkel abhängig ist. Andererseits müssen sie nicht angelegt oder kalibriert werden und es muss auch kein etwaiger Drift kompensiert werden. Diese Diplomarbeit unterstützt sowohl optische als auch getragene Tracking-Mechanismen.

Für simple Anwendungen, die kein exaktes Tracking benötigen, setzt diese Arbeit auf ein optisches Tracking basierend auf Microsofts Kinect-Technologie[Mic15b; Mic15c]. Im Vergleich mit anderen optischen Trackingsystemen, sind die Kinect-Geräte deutlich günstiger und problemloser in Unity3D zu integrieren. Dabei können sowohl Geräte der aktuellen, als auch jene der alten Kinect for Windows Generation verwendet werden. Das Funktionsprinzip ist bei beiden Systemen das gleiche: Ein Infrarotemitter gibt Infrarotstrahlen aus, deren Reflexionen von einem Infrarot-Sensor gemessen werden. Die Bildrate liegt bei 30 Hz. Die Infrarotstrahlen bilden ein spezielles Muster, wodurch der Sensor die einzelnen Strahlen identifizieren kann. Dadurch lässt sich ein Tiefenbild der Umgebung erstellen, auf dem das Kinect-SDK Personen erkennen kann. Die Identifikation der Personen funktioniert automatisch und erfordert keinerlei Kalibrierungspose. Erkannten Personen wird ein Skelett zugeordnet, dessen Gelenkspositionen und -winkel Softwareanwendungen zur Verfügung gestellt werden. Diese Daten können anschließend auf einen Avatar übertragen und für etwaige Kollisionsabfragen verwendet werden. Zusätzlich beinhalten die Geräte noch Farbkameras und Mikrofone, diese Features finden in dieser Arbeit allerdings keine Anwendung. Die Kinect2 bietet gegenüber ihrer Vorgängerversion eine höhere Auflösung und ein größeres Sichtfeld. Die maximale Entfernung beläuft sich auf bis zu fünf statt vier Meter und die Beleuchtung des Raumes hat weniger Einfluss auf das Tracking. Die Software der neueren Generation kann zudem weitere vier Personen gleichzeitig verfolgen und erkennt fünf Gelenke mehr: Beide Daumen, die entferntesten Fingerspitzen der Hände und das Genick. Weiters kann sie geschlossene oder geöffnete Hände unterscheiden. Einen Stellmotor für die Änderung des Blickwinkels besitzt allerdings nur die Kinect1. Da der Anwender bzw. die Anwenderin aber immer am selben Platz steht, ist eine nachträgliche Änderung des Neigungswinkels des Sensors auch nicht notwendig. Obwohl die Kinect2 eine etwas genauere Erkennung liefert, ist die Kinect1 für die Anwendung in einfachen Szenarien ebenso ausreichend. Die beiden Versionen der Kinect sind in Abbildung 20 zu sehen.[Mic15b; Mic15c; Mic15a]

Obwohl das optische Tracking für einige Szenarien ausreichend ist, benötigen komplexere Anwendungsfälle oft ein genaueres Tracking, dessen Qualität nicht vom Betrachtungswin-

(a) (b)

Abbildung 20: Unterstützte Kinect Sensoren: a) Kinect1 mit Halterung für eine Montage auf einem Stativ, b) Kinect2 auf einem Stativ.

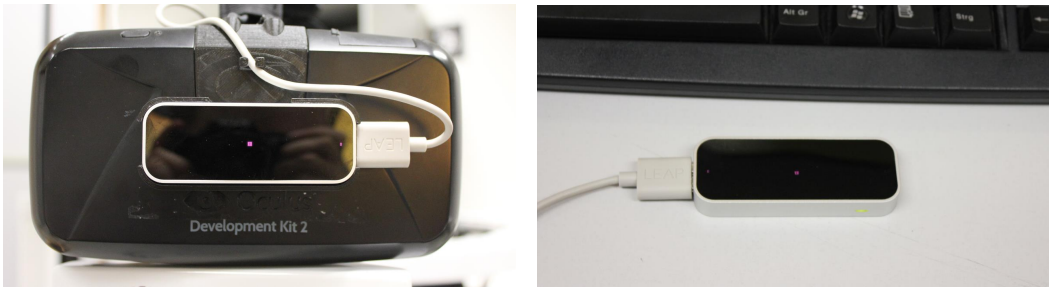
kel des Sensors abhängig ist. Hierfür verwendet diese Diplomarbeit einen Motion-Capture-Anzug namens *Perception Neuron*[Noi16]. Andere kameralose Motion-Capture-Anzuge sind entweder teurer oder sind anfälliger für Störungen, wie etwa der PrioVR™[YOS17]. Der Perception Neuron ist ein modularer Anzug, bestehend aus einer Basiseinheit und mehreren Sensoreinheiten. Die Sensoreinheiten besitzen ein Gyroskop, ein Magnetometer und einen Beschleunigungssensor mit jeweils drei Achsen. Diese *Neuron* genannten Sensoreinheiten können auf verschiedene Weisen miteinander verbunden werden, sodass entweder der ganze Körper oder nur einzelne Gliedmaßen getracked werden können. Die Sensoren werden dazu mit elastischen Bändern oder Handschuhen an den jeweiligen Körperregionen angebracht und messen dort die Veränderungen. Die Basiseinheit sammelt die Daten, verarbeitet sie und sendet sie via WLAN oder USB-Kabel an einen PC. Die Auffrischungsrate beträgt bei 18 oder weniger Sensoren 120 Hz, bei mehr Sensoren 60 Hz. Das dort laufende Programm *Axis Neuron* errechnet aus den Daten die Winkel der einzelnen Gelenke. Diese Winkel können von anderen Anwendungen ausgelesen und auf die Körpermodelle übertragen werden. Damit die errechneten Winkel möglichst exakt sind, muss nach dem Anlegen des Anzugs eine Kalibrierung mit dem Programm durchgeführt werden. Axis Neuron enthält mehrere vordefinierte Modelle, auf die die errechneten Winkel übertragen werden. Wird der gesamte Körper getracked, wird ein humanoider Avatar verwendet. Axis Neuron erkennt Gehbewegungen und verschiebt den Avatar entsprechend im Raum. Weiters ist es möglich, das Modell in der Anwendung zu drehen, oder an den Ursprung zurückzusetzen.

Sämtliche Daten des Modells, inklusive der errechneten Position im Raum, werden anschließend per Netzwerkprotokoll anderen Programmen zur Verfügung gestellt. Im Gegensatz zu optischem Tracking bietet diese Variante die Möglichkeit, jede Pose zu erkennen. Die Gelenkspositionen werden allerdings nicht gemessen, sondern ergeben sich aus dem Winkel und der Position des übergeordneten Gelenks. Daher ist es wichtig, dass das verwendete Modell den Maßen der Benutzerin bzw. des Benutzers entspricht. Eine weitere Einschränkung ist das Auftreten eines Drifts, da die Erkennung der Änderung nicht zu 100 Prozent genau ist. Deshalb stimmen die gemessenen Positionen nach einiger Zeit nicht mehr mit den tatsächlichen überein und es wird eine erneute Kalibrierung notwendig. Die Sensoren sind außerdem anfällig für Störungen, etwa durch magnetische Felder, wodurch es noch schneller zu einer Abweichung kommen kann. Abbildung 21 zeigt mögliche Anordnungen der Neurons.[Noi16].

(a) (b)

Abbildung 21: Mögliche Anwendungsgebiete von Perception Neuron: a) Konfiguration für die genaue Erkennung einer Hand, b) Ganzkörpertracking mit jeweils zwei Sensoren pro Hand.

In manchen Einsatzgebieten ist es notwendig, die gesamte Hand zu erfassen und nicht nur ihre Position. Mit dem zuvor genannten Motion Capture Anzug ist dies zwar ebenfalls möglich, es erfordert aber, dass jedem Gelenk auf der Hand ein eigener Sensor zugeordnet wird, wie in Abbildung 21a zu sehen ist. Dadurch muss zum einen mehr Haut mit einem Handschuh bedeckt werden, zum anderen schränkt die zusätzliche Verkabelung die Bewegungsfreiheit ein. Weiters wirken sich Ungenauigkeiten beim Tracking wesentlich stärker aus, wodurch zum Beispiel das Greifen kleiner Gegenstände erschwert wird. Daher verwendet diese Arbeit ein markerloses optisches Tracking namens *Leap Motion Orion*[Lea16b]. Das System besteht aus einer Hardwarekomponente, dem *Leap Motion* Sensor und einer PC-Anwendung. Die Hardware des Systems setzt sich aus einer kleinen Stereo-Infrarotkamera und drei Infrarotemittern zusammen. Die Auffrischungsrate kann bis zu 200 Hz betragen Die Emitter leuchten das nähere Umfeld der Kamera aus und die beiden Weitwinkellinsen nehmen jeweils ein Bild auf. Im Gegensatz zur Kinect wird kein Muster verwendet, mit dem die Kamera ein Tiefenbild erstellen kann. Stattdessen werden die beiden Einzelbilder an eine PC-Software weitergeleitet, welche die Bilder analysiert und mithilfe spezieller Tracking- und Filteralgorithmen ein 3D-Bild der erkannten Hände erstellt. Die Algorithmen sind in der Lage, bis zu einem gewissen Grad Hände auch dann zu erkennen, wenn sie teilweise verdeckt sind, oder die Infrarotkameras durch externe Quellen geblendet werden. Das so erstellte 3D-Bild weist eine höhere Genauigkeit auf als die des Perception Neuron und es besteht zudem keine Gefahr eines Drifts. Durch diese Art der Erkennung ist die Kamera sehr kompakt, weshalb sie sich gut auf einem HMD befestigen lässt. Alternativ kann der Sensor auf einem Tisch abgelegt werden, um eine Gestenerkennung für Programme zu liefern. Die beiden Montagemöglichkeiten sind in Abbildung 22 dargestellt. Wie bei allen optischen Tracking-Verfahren, besteht auch hier das Problem, dass die Hände nur dann erkannt werden können, wenn sie sich im Sichtbereich beider Linsen befinden.[Lea16a; Lea16b; Col14]



(a) (b)

Abbildung 22: Zwei Möglichkeiten, den Leap Motion Sensor zu verwenden: a) auf dem DK2 montiert, b) als weiteres Eingabegerät für Desktopanwendungen.

3.1.4 Roboterarm und PC

Die letzte verwendete Peripherie-Einheit ist zugleich das Kernstück der Arbeit: Ein speziell für diesen Anwendungsfall zusammengestellter *CrustCrawler Pro Series* Roboterarm. Dieser Arm wird für die Erzeugung des haptischen Feedbacks verwendet. Der genaue Aufbau und der Einsatz des Arms werden in Abschnitt 4.1.2 beschrieben.

Zusätzlich zu den oben genannten Komponenten ist ein aktueller PC notwendig, der die Hardware-Anforderungen der Peripherie erfüllt und das VR-Programm ausführt. Der verwendete PC besitzt einen 4GHz Intel Core i7 Prozessor, 32 GB Arbeitsspeicher und eine MSI GeForce GTX980 Ti Grafikkarte.

3.2 Softwarekomponenten

Abgesehen von den verwendeten Geräten und den dazugehörigen Anwendungen, sind auch reine Softwarekomponenten notwendig, um die einzelnen Teile zu verbinden. Dieser Abschnitt beschreibt die Eigenschaften der jeweiligen Komponenten und hebt ihre Einsatzgebiete hervor.

Kernthema dieser Arbeit ist die Integration eines Roboterarms in ein VR-Szenario. Um einen solchen Arm korrekt steuern zu können, ist es notwendig zu wissen, welche Winkel die einzelnen Gelenke einnehmen müssen, damit eine gewünschte Zielposition erreicht wird. Zur Berechnung der Gelenkwinkel wird eine *Inverse Kinematic* (*IK*) verwendet. Es gibt eine Vielzahl von Ansätzen, eine solche IK zu erstellen. Sie kann manuell erstellt oder mit einem Programm generiert werden. Für eine praktische Anwendung ist es aber oft notwendig, dass die IK in möglichst kurzer Zeit Lösungen findet. Dazu ist es wichtig, dass die IK für den jeweiligen Roboterarm optimiert ist. Diese Optimierungen erfordern komplexe Berechnungen und jede Änderung des Aufbaus erfordert eine neuerliche Anpassung der IK. Aus diesem Grund setzt diese Arbeit auf das Robotikframework *Open Robotics Automation Virtual Environment* (*OpenRAVE*)[Dia13d].

„*OpenRAVE bietet eine Umgebung, um Bewegungsplanungs-Algorithmen für Robotik-Anwendungen zu entwickeln, zu testen und einzusetzen. Der Hauptfokus liegt auf der Simulation und der Analyse der kinematischen und geometrischen Informationen im Bezug auf Bewegungsplanung.*“ [Dia13d]

Das Framework besitzt eine modulare Architektur, die in Abbildung 23 dargestellt ist und sich aus den folgenden Komponenten zusammensetzt: Der Kern (*Core*) stellt Schnittstellendefinitionen zur Verfügung und verwaltet die geladenen Umgebungen inklusive der enthaltenen Roboter und Objekte. Die Plugins können zur Laufzeit geladen und ausgetauscht werden. Durch Plugins (*Components*) können etwa neue Planungsalgorithmen oder auf einen bestimmten Roboter zugeschnittene Treiber implementiert werden, ohne die Kernanwendung anpassen zu müssen. Die Roboterdatenbanken (*Robot Database*) enthalten alle statischen Informationen über einen Roboter. Die Informationen werden in einem anfänglichen Analyseschritt erstellt und müssen nur nach einer Änderung der Geometrie des Roboters erneut gesammelt werden. Die Skriptunterstützung (*Scripting Environment*) ermöglicht das schnelle Erstellen und Testen von Umgebungen, ohne auf die C++-API zugreifen zu müssen. Es werden die Skriptsprachen Python und Octave/Matlab unterstützt. Ein besonders wichtiger Bestandteil des Frameworks ist IKFast.

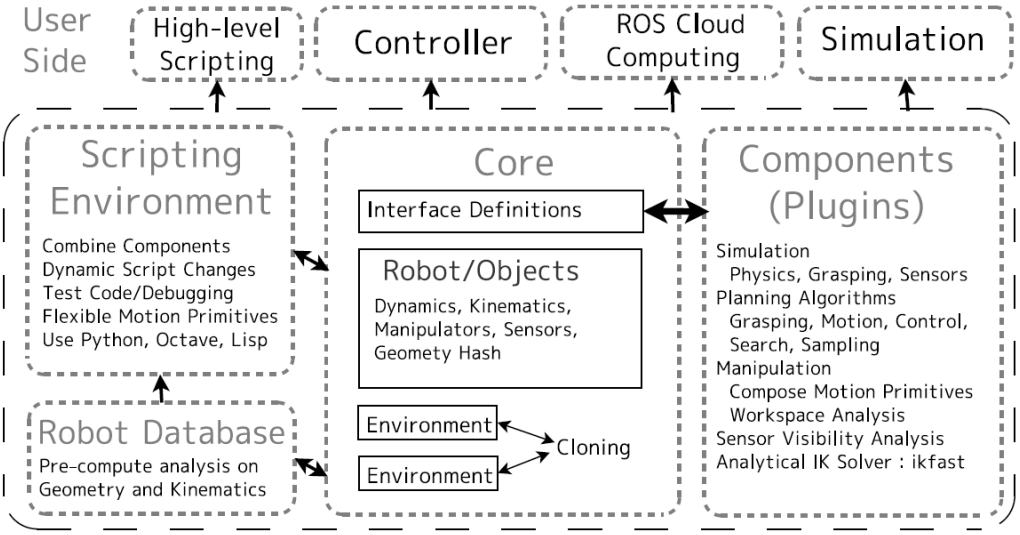


Abbildung 23: Architektur von OpenRAVE.[Dia10, Seite 235]

IKFast erstellt auf Basis der in der Roboterdatenbank enthaltenen Informationen einen speziell auf den Roboter zugeschnittenen analytischen IK-Löser. Dieser besteht aus einem optimiertem C++-Code, der zur Laufzeit geladen werden kann und im besten Fall 4 Mikrosekunden für die Suche nach der IK einer Pose benötigt. Somit reicht es aus, den Roboter im XML[Dia13a]- oder COLLADA[Dia13c]-Format in OpenRAVE zu definieren und ein Kontrollplugin zu erstellen, um die Steuerung mittels OpenRAVE zu verwenden.[Dia10; Dia13d; Dia13b] Die Integration von OpenRAVE in die virtuelle Umgebung

wird in Abschnitt 4.2.1 besprochen.

Eine ebenso wichtige Komponente stellt die 3D-Engine dar, da sie Funktionen bietet, um die virtuelle Welt sicht- und begehbar zu machen. Es gibt eine Reihe unterschiedlicher 3D-Engines; diese Arbeit verwendet *Unity*[Uni15b] in der kostenlosen Personal-Edition. Unity besitzt einen Editor, in dem das Terrain der Umgebung sowie graphische Primitive erstellt werden. Komplexere Modelle können aus Modellierungsanwendungen importiert werden, dazu zählen auch die in OpenRAVE verwendeten COLLADA Szenen. Zusätzlich stehen im *Asset-Store* über 170000 kostenlose und kostenpflichtige Erweiterungen und Modelle zur Verfügung. Allen Objekten in der Szene können Skripte zugeordnet werden, um die Programmlogik zu implementieren. Die Skripte können in C# und JavaScript erstellt werden, als Laufzeitumgebung wird Mono verwendet. Weiters werden auch plattformgebundene Bibliotheken unterstützt, die etwa zur Integration spezieller Hardware benötigt werden. Unity unterstützt die Ausgabe auf VR-Brillen; für alle anderen erwähnten Hardwarekomponenten existieren Assets, um sie in Unity einzubinden. Die erstellte Anwendung kann entweder direkt im Editor ausgeführt oder zu einem eigenständigen Programm kompiliert werden. Hierfür wird eine Vielzahl an unterschiedlichen Plattformen unterstützt, diese Arbeit beschränkt sich aber nur auf PC-Anwendungen unter Windows. Die einfache Verwendung und die bereits vorhandene Unterstützung der benötigten Geräte waren ausschlaggebend für die Verwendung von Unity als 3D-Engine.[Uni15b; Uni15a]

KAPITEL 4

Umsetzung

Dieses Kapitel beschreibt den Aufbau des immersiven VR-Systems. Die Umsetzung erfolgte im Rahmen eines Projekts der Interactive Media Systems Forschungsgruppe am Institut für Softwaretechnik und Interaktive Systeme dessen Ergebnisse in einer internationalen wissenschaftlichen Publikation[VGK17] veröffentlicht wurden. Abschnitt 4.1 beschäftigt sich mit dem Aufbau des Hardware-Setups und den Entstehungsschritten Abschnitt 4.2 beschreibt die Implementierung des Software-Frameworks und wie damit VR-Anwendungen mit haptischem Feedback erstellt werden können. Abschließend geht Abschnitt 4.3 auf mögliche Gefahren im Umgang mit dem System ein und wie diesen begegnet werden kann.

4.1 Hardware

Der Fertigungsprozess des Hardware-Aufbaus erfolgte in mehreren Iterationen. Abschnitt 4.1.1 erklärt, wie die Komponenten aus Kapitel 3 zusammenspielen und schildert den Ablauf der Realisierung. Abschnitt 4.1.2 gibt einen tieferen Einblick in die Spezifikation des Roboterarms.

4.1.1 Konstruktion

Dieses System kombiniert verschiedene gebrauchsfertige Geräte, um ein hohes Maß an Immersion zu erzielen. Je nach Anwendungsfall kommen aber nicht alle Komponenten zum Einsatz. Abbildung 24 zeigt den kompletten Aufbau unter Verwendung aller Teile.

Die Basis der Bewegungserkennung bildet die omnidirektionale Bewegungsplattform Cyberith Virtualizer (Abbildung 24a). In ihm kann sich der Benutzer oder die Benutzerin annähernd natürlich in alle Richtungen bewegen. Schritte, sowie Richtungs- und Höhenänderungen werden von der Plattform registriert und an den Computer weitergeleitet.

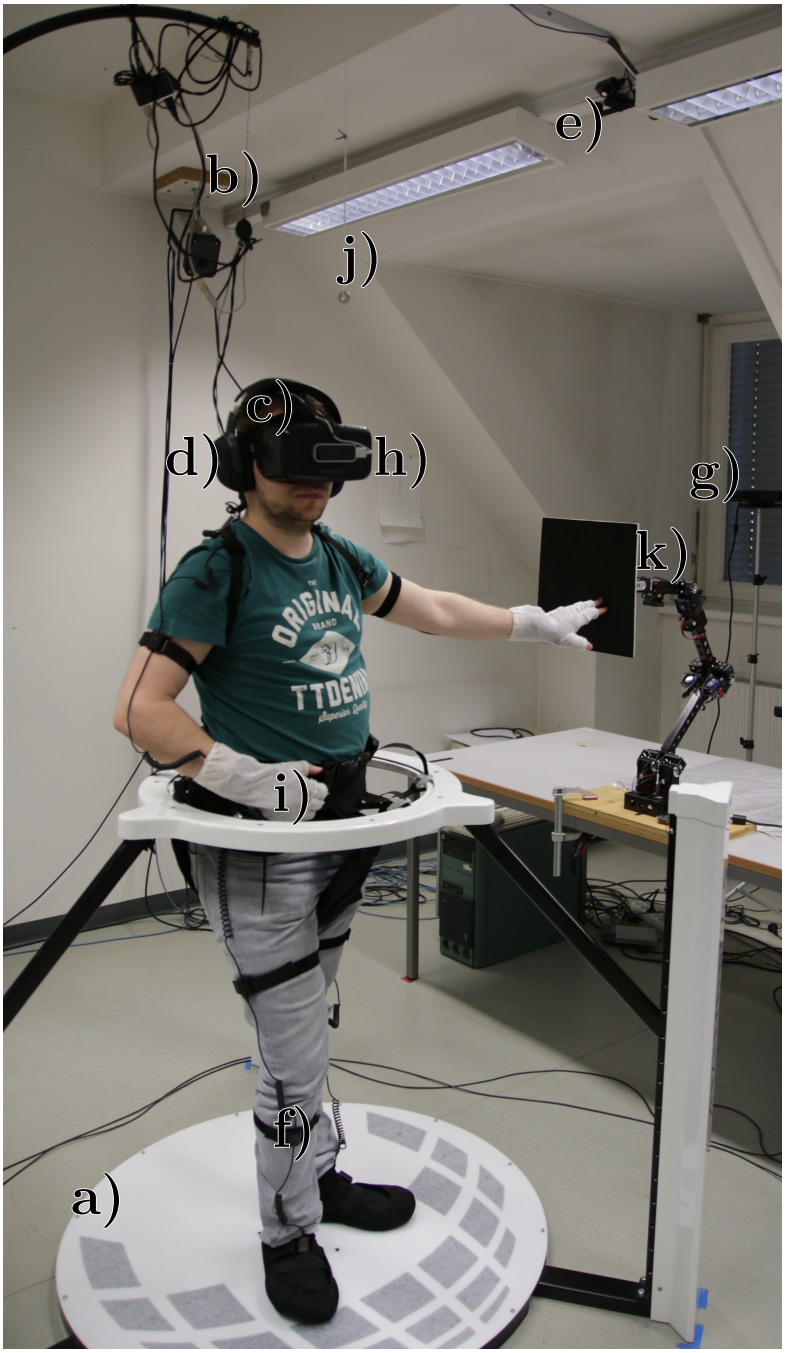


Abbildung 24: Aufbau des immersiven VR Systems: a) Cyberith Virtualizer, b) ausziehbare Kabelhalterung *The Arm*, c) *Oculus Rift Development Kit 2*, d) 7.1 USB-Headset, e) *Oculus Rift Development Kit 2* Infrarotkamera, f) Perception Neuron in der Ganzkörper-Konfiguration, g) Microsoft Kinect2, h) Leap Motion, i) atmungsaktive Überhandschuhe, j) Lot zur Ausrichtung der Kopfposition an der Nullstellung, k) CrustCrawler Pro-Series Roboterarm mit sieben Freiheitsgraden.

Um eine stabilere Auftrittsfläche zu bieten, trägt die Anwenderin bzw. der Anwender Gymnastikschuhe in den Überschuhen. Im Zuge der Umsetzung des VR-Systems wurden zwei Versionen des Virtualizers verwendet. Zu Beginn kam der Prototyp 2 zum Einsatz. Der Aufbau dieses Prototyps ist dem aktuellen sehr ähnlich, allerdings bot dieser keine einheitliche Programmierschnittstelle. Die Weitergabe der Bewegungen erfolgte noch über Text-Kommandos auf einer TCP-Verbindung, die von einem Client in Tastenanschläge umgewandelt wurden. Dieser Client konnte zwar durch ein Unity-Script ersetzt werden, die TCP-Übertragung blieb aber weiterhin bestehen. Dadurch kannte das System nur einen Unterschied zwischen Gehen und Laufen, eine exakte Übertragung der tatsächlichen Gehgeschwindigkeit war nicht möglich. Weiters war die Gleitfähigkeit der Bodenplatte etwas geringer, was sich negativ auf die Natürlichkeit der Bewegung auswirkte. Durch Umstieg auf die aktuellere Version wurden viele dieser Probleme behoben. Das Virtualizer-SDK unterstützt eine stufenlose (analoge) Gehgeschwindigkeit und kommt ohne Client-Server-Kommunikation aus. In Kombination mit der verbesserten Gleitfähigkeit der Basisplatte und den dazugehörigen Überschuhen wurde die Bewegung im Virtualizer vereinfacht. Zusätzlich besitzt der aktuelle Virtualizer eine gebogene Stange mit einer ausziehbaren Halterung (*The Arm*), an der etwaige Kabel befestigt werden können. Dadurch können sich Benutzer und Benutzerinnen um 360° drehen, ohne durch die Verkabelung behindert zu werden (Abbildung 24b).

Die zweite Kernkomponente stellt das HMD dar. Wie bei der Bewegungsplattform gab es auch in diesem Bereich zu Beginn der Umsetzungsphase nur eine sehr eingeschränkte Auswahl an Geräten. In frühen Prototypen kam noch das *Oculus Rift Development Kit 1* (*DK1*) zum Einsatz, welches aber nur eine Übergangslösung darstellte. Ab dessen Verfügbarkeit wurde auf das *Oculus Rift Development Kit 2* (*DK2*) umgestiegen, da es deutliche Vorteile bot. Die Integration basiert auf dem Oculus Rift SDK und musste, abgesehen von Aktualisierungen, nicht angepasst werden. Gemeinsam mit einem 7.1 USB-Headset mit passiver Geräuschunterdrückung (Abbildung 24d) bietet das DK2 (Abbildung 24c) die audiovisuellen Eindrücke. Die notwendigen Kabel werden durch die Halterung des Virtualizers von der Anwenderin bzw. dem Anwender ferngehalten.

Die Implementierung des Tracking-Mechanismus erfolgte ebenfalls in mehreren Schritten. Als erste Version kam eine Kinect1 zum Einsatz. Obwohl sie leicht integriert und verwendet werden konnte, waren die Ergebnisse nicht zufriedenstellend. Die niedrige Auflösung bewirkte eine hohe Abweichung zwischen Avatar und dem Benutzer bzw. der Benutzerin. Weiters setzte das Tracking häufig aus, wenn Teile des Körpers verdeckt waren oder die Anwenderin bzw. der Anwender sich zu weit gedreht hatte. Eine geringfügige Verbesserung brachte die Umstellung auf eine Kinect2. Die verbesserte Auflösung verringerte die Ungenauigkeiten, die Problematik der Verdeckungen blieb allerdings bestehen. Für einige Szenarien, in denen etwa die Bewegungsrichtung eingeschränkt oder die Genauigkeit der Trackingdaten nicht essentiell ist, reicht das optische Tracking dennoch aus.

Um eine Trackinglösung zu implementieren, die mit Verdeckung und Rotation des Benutzers bzw. der Benutzerin umgehen kann, wurde Perception Neuron (Abbildung 24f) eingesetzt. Die 17 Sensoren des Motion-Capture-Anzugs liefern ein ausreichend genaues

Ergebnis, müssen jedoch aufwändiger kalibriert werden. Außerdem muss die Größe der Anwenderin oder des Anwenders mit der des Avatars exakt übereinstimmen, da die Sensoren nur die Winkel, nicht aber die Positionen der Gelenke liefern. Ein weiterer Schwachpunkt des Anzugs sind die relativ dünnen Verbindungskabel zwischen den einzelnen Neurons, die durch die Gurte des Virtualizers beschädigt werden können. Zum Schutz dieser Kabel werden mittels 3D-Drucker gefertigte Schienen verwendet, welche die Kabel zwischen Person und Gurt umhüllen. Die Kommunikation zwischen Anzug und PC-Anwendung erfolgt über WLAN, als Stromquelle dient ein USB-Akkupack am Rücken des Benutzers bzw. der Benutzerin. Da das Anlegen und die Kalibrierung des Perception Neuron zeitintensiv ist, wurde die Unterstützung beider Kinect-Varianten ebenfalls beibehalten. So kann die Kinect etwa als Notfallsystem (Abbildung 24g) bei Ausfall des Akkus das Tracking übernehmen, oder die initiale Vermessung der Anwenderin bzw. des Anwenders durchführen.

Der Tracking-Mechanismus registriert den gesamten Körper des Benutzers bzw. der Benutzerin. Anfängliche Umsetzungen verwendeten die erkannte Position des Kopfs zur Positionierung der *virtuellen Augen* in der Anwendung, während die Blickrichtung vom DK2 beigesteuert wurde. Das Kopf-Tracking und die Orientierung waren allerdings in vielen Fällen nicht kohärent, worunter die Kopplung zwischen virtueller und tatsächlicher Realität litt. Das führte in manchen Fällen zu einer eingeschränkteren Immersion oder zu Übelkeit. Um diesem Problem entgegenzuwirken, wird die Infrarot Kamera des DK2 eingesetzt. Diese ist in einer erhöhten Position montiert und überwacht von dort den Bereich des Virtualizers (Abbildung 24e). Sie erkennt im HMD integrierte Infrarot-Markierungen und kann so die absolute Kopfposition in der virtuellen Welt errechnen. Dadurch wurde die starre Kopplung zwischen dem Kopf des Avatars und den virtuellen Augen aufgelöst. Die Position des Kopfs wird weiterhin durch den Tracking-Mechanismus bestimmt und dient der Kollisionsvermeidung. Die Position der virtuellen Augen anhand der absoluten Position des DK2 berechnet.

Ein weiterer Punkt, bei dem das Tracking-System an seine Grenzen stieß, waren Interaktionen mit den Händen. Hierbei ist eine wesentlich höhere Genauigkeit erforderlich, als der Perception Neuron oder eine Kinect bieten können. Als optionale Ergänzung ist die Stereo-Infrarot-Kamera Leap Motion auf dem HMD angebracht (Abbildung 24h). Diese erkennt die Hände der Benutzerin oder des Benutzers und überträgt dies auf die virtuellen Körperteile. Dieses System ist als Ergänzung implementiert und kann auch deaktiviert werden, wenn beispielsweise keine exakte Erkennung notwendig ist. In diesem Fall wird die Handposition des Ganzkörper-Trackings verwendet. Das passiert ebenfalls, wenn die Kamera eine Hand nicht erkennen kann, weil etwa der Anwender oder die Anwenderin nicht zu ihren oder seinen Händen blickt. Während die Leap Motion ohne weitere Anpassungen mit dem Kinect-Tracking kompatibel war, mussten für den Perception Neuron spezielle Überhandschuhe gefertigt werden (Abbildung 24i). Die Handschuhe und Sensoren des Anzugs absorbieren das Infrarotlicht der Kamera, wodurch die Hände nicht korrekt erkannt werden können. Anfänglich wurden Latex-Handschuhe verwendet, bei denen die Fingerkuppen entfernt wurden. Diese boten zwar eine ausreichende Flexibilität

und Reflexion, waren allerdings nicht atmungsaktiv. Andere Handschuhe waren entweder nicht flexibel genug oder konnten von der Kamera nicht richtig erkannt werden. Aus diesem Grund wurden die verwendeten Überhandschuhe speziell zu diesem Zweck aus Verbandsmaterial gefertigt, welches ebenfalls das Infrarotlicht reflektiert. Dieses ist zum einen dehnbar genug, sodass nur eine Größe notwendig ist, zum anderen lässt das Material die Feuchtigkeit entweichen und schützt so die Sensoren.

Das Ganzkörper-Tracking positioniert die Gliedmaßen des Avatars immer relativ zur Körpermitte. Da Benutzer und Benutzerinnen stets mittig im Virtualizer fixiert sind, ist die Körpermitte immer bekannt. Im Gegensatz dazu werden die Hände der Leap Motion relativ zu den virtuellen Augen ausgerichtet. Diese werden wiederum durch die Kamera des DK2 positioniert. Die Kamera erkennt allerdings nur die Veränderungen von einer Initialposition. Dadurch ist es notwendig, den Kopf beim Start der Anwendung so auszurichten, dass sich die virtuellen Augen mit den tatsächlichen decken. Dazu ist ein höhenverstellbares Lot angebracht, um die Nullposition zu markieren (Abbildung 24j).

Das vielseitigste Element stellt der CrustCrawler Pro-Series 7-Achsen Roboterarm dar (Abbildung 24j). Dieser befindet sich an einer festen Position relativ zur Mitte des Virtualizers und kann sowohl Gegenstände simulieren, als auch mit der Benutzerin bzw. dem Benutzer interagieren. Dadurch wird er zur Schlüsselkomponente des Setups, da er das passiv-haptische Feedback bietet und die virtuelle Welt somit berührbar macht. Kommt der Anwender oder die Anwenderin in der virtuellen Realität in die Nähe eines berührbaren Elements, greift der Arm die passende Requisite und positioniert sie entsprechend. Da die Position des Arms in Relation zum Mittelpunkt des Virtualizers bekannt ist, lässt sich die Zielposition und -ausrichtung des props berechnen. Die Mitte des Virtualizers entspricht dem Zentrum des Avatars, die Zielpose ergibt sich daher durch Umrechnung der Weltkoordinaten in das lokale Koordinatensystem des Avatars. Eine Veränderung der Pose ist nur notwendig, wenn sich die Anwenderin oder der Anwender in der virtuellen Welt fortbewegt. Beugt er oder sie sich in Richtung des Objekts oder greift danach, so kann die relative Position beibehalten werden.

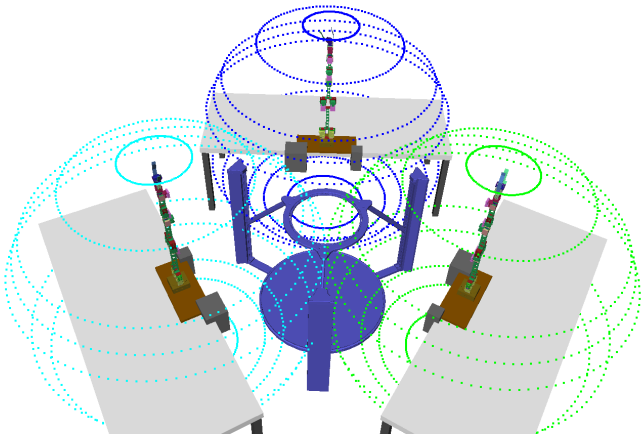


Abbildung 25: Setup mit drei Roboterarmen für haptisches Feedback von jeder Seite.

Um den kompletten Raum um den Virtualizer mit passiv-haptischem Feedback zu versorgen, sind mindestens drei derartige Arme notwendig. Ein solcher Aufbau ist in Abbildung 25 dargestellt. Die Arme müssen untereinander synchronisiert sein um Kollisionen im Überlappungsbereich zu vermeiden. Die Planungssoftware unterstützt die gleichzeitige Verwendung mehrerer Arme, das aktuelle Setup enthält allerdings nur einen Roboter, da dies für die Evaluierung des Konzepts ausreichend ist. Im Laufe der Entwicklung des Setups wurde der Arm immer weiter verbessert. Auf die genaue Spezifikation des Arms wird im folgenden Abschnitt eingegangen.

4.1.2 Spezifikation des Roboterarms

Als verwendeter Roboterarm wurde kein gebrauchsfertiges Modell gewählt, sondern speziell für den Einsatz im entwickelten VR-System zusammengestellt. Die Motoren und Verbindungsstücke sind so gewählt, dass eine gute Balance zwischen möglicher Last und Reichweite besteht. Dazu musste der Aufbau mehrmals adaptiert werden, um den Arbeitsbereich zu optimieren. Die verschiedenen Konfigurationen sind in Abbildung 26 dargestellt. Die blauen Punkte geben dabei an, ob eine Position erreicht werden kann, die roten Striche zeigen die möglichen Ausrichtungen an der jeweiligen Stelle. Initial bestand der Arm aus 9 Servomotoren der Dynamixel-Serie[ROB15], die durch spezielle Aluminiumträger und -halterungen von CrustCrawler[Cru] verbunden sind. Die Eigenschaften der Servos sind in Tabelle 1 angeführt.

Modell	**Drehmoment**	**Maximale Geschwindigkeit ohne Last**	**Schrittgröße**
AX-18A	1.8 Nm	582°/s	0.29°
MX-28T	2.5 Nm	330°/s	0.088°
AX-64T	6 Nm	378°/s	0.088°
AX-106T	8.4 Nm	270°/s	0.088°

Tabelle 1: Technische Daten der verwendeten Motoren.[ROB15]

Abhängig von den verwendeten Verbindungsstücken, verändert ein Motor entweder die Neigung oder die Drehung des Gelenks. Von der Basis beginnend waren die Bestandteile folgendermaßen angeordnet: ein MX-106T Aktor mit einer Drehplatte für die Rotation des Arms, zwei gleichgerichtete MX-106T, die zu einem Neigungsgelenk verbunden sind, ein auf bis zu 33,81 cm ausziehbarer Träger, ein Neigungsgelenk aus zwei MX-64T, ein 12,7 cm langer Träger, zwei aufeinanderfolgende MX-64T (Rotation und Neigung), ein 12,7 cm langer Träger, ein MX-28T (Rotation), und ein AX-18A Greifer. Die Aktoren des Arms sind in Serie zu einem Bus zusammengeschlossen, wobei jeder Motor einen eigenen Controller besitzt. Über ein dreipoliges Verbindungskabel werden sowohl Strom, als auch Steuersignale weitergegeben. Die Kommunikation ist eine serielle, asynchrone Halbduplexkommunikation für *Transistor-Transistor-Logik* (*TTL*). Die Steuerpakete werden auf einem PC erzeugt und über einen USB Adapter (*USB2Dynamixel*) an die Motoren gesendet. Ein Paket kann dabei an einen oder mehrere Servos adressiert sein, bei letzterer Variante werden die Befehle zeitgleich ausgeführt. Die Daten der Pakete

werden in den Speicher der einzelnen Motoren geschrieben, vom jeweiligen Controller ausgelesen und umgesetzt. Zusätzlich werden Antwortpakete zurückgesendet, die den Status der Motoren beinhalten.

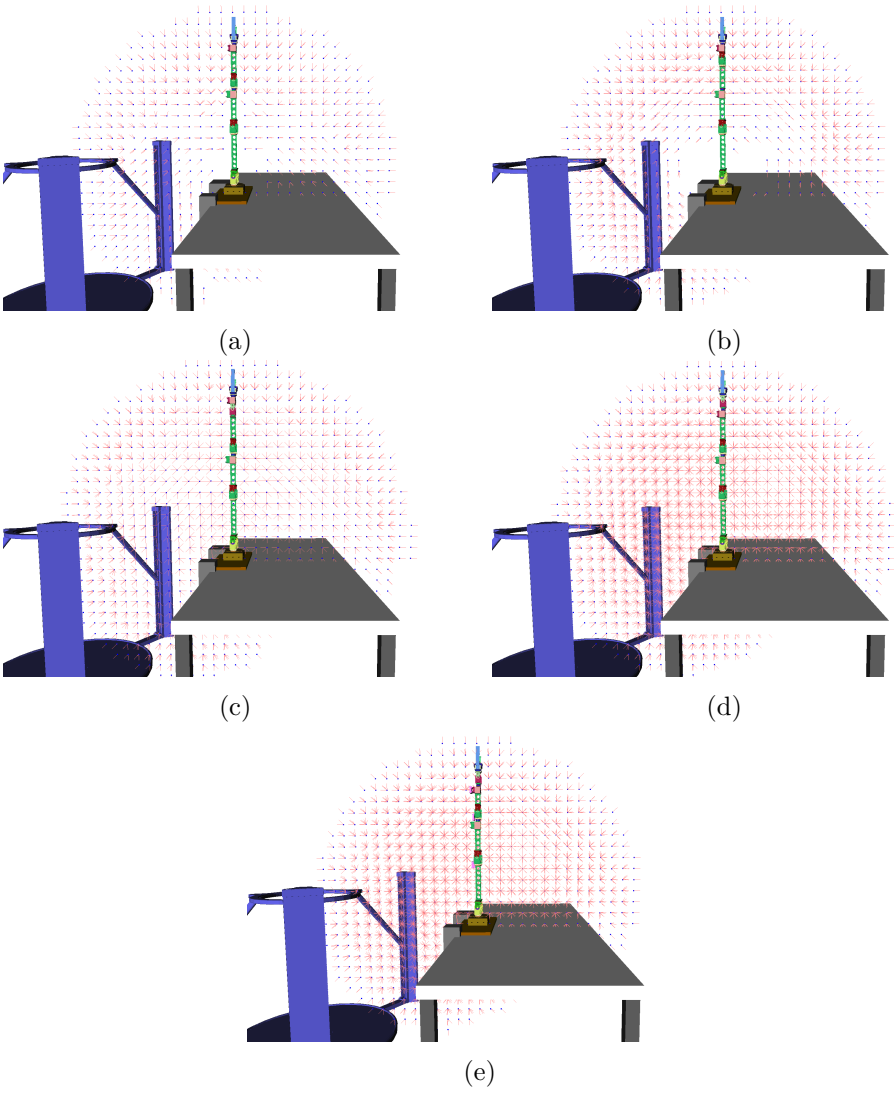


Abbildung 26: Arbeitsbereiche der unterschiedlichen Arm-Konfigurationen: a) Initiale Konfiguration, b) Verlagerung des einfachen Kippers hinter den Träger, c) ein weiteres Kippgelenk hinter dem Träger, d) ein weiteres Kippgelenk vor dem Greifer, e) finale Konfiguration: Ein weiteres Kippgelenk vor dem Greifer, ein kürzerer Träger und zusätzliche Ventilatoren zur Kühlung.

Wie in Abbildung 26a zu sehen ist, konnten in der Nähe der Basis kaum Positionen eingenommen werden. Zusätzlich war es nur an sehr wenigen Stellen möglich, Objekte parallel zu verschieben. Um die Veränderung der Position bei gleichbleibender Ausrichtung

zu ermöglichen, wurde das Kippgelenk zwischen den Träger und den MX-28T verschoben. Die Auswirkung dieser Änderung ist in Abbildung 26b dargestellt: Ab einer Entfernung von ca. 50 cm von der Basis war die Parallelverschiebung möglich, allerdings wurde die Anzahl der erreichbaren Positionen in der Nähe der Basis weiter reduziert. Diese Konfiguration war daher ebenso nicht zufriedenstellend. Um sowohl die Anzahl der erreichbaren Positionen als auch der möglichen Ausrichtungen zu erhöhen, musste ein weiteres Kippgelenk eingebaut werden. Abbildung 26c und Abbildung 26d zeigen die möglichen Varianten. Bei beiden Konfigurationen waren dieselben Punkte erreichbar, allerdings konnten bei der abwechselnden Anordnung von Dreh- und Kippgelenken deutlich mehr unterschiedliche Orientierungen eingenommen werden. Weiters war die IK bei der letzten Konfiguration deutlich simpler, was die Berechnungsdauer markant verkürzte. Der zusätzliche Motor verlängerte den Arm, wodurch die maximale Tragkraft reduziert wurde. Um dies auszugleichen, wurde der 12,7 cm Träger, der näher am Greifer liegt, durch einen nur 6,35 cm langen Träger ersetzt. In Abbildung 26e ist zu sehen, dass durch diese Maßnahme die maximale Reichweite etwas verringert wurde, während die möglichen Ausrichtungen nahezu ident blieben.

Abbildung 27 zeigt die finale Zusammenstellung des Arms: die MX106-T Drehplatte, das Neigungsgelenk aus zwei MX106-T, ein auf 21,11 cm Länge ausgezogener Träger, ein Neigungsgelenk aus zwei MX-64T, ein 12,7 cm langer Träger, zwei aufeinanderfolgende MX-64T (Rotation und Neigung), ein 6,35 cm langer Träger, zwei MX-28T (Rotation und Neigung), und ein AX-18A Greifer. Der Greifer besteht aus einem starren und einem beweglichen Teil. Je nach Anwendungsfall können am beweglichen Teil ein oder zwei *Finger* montiert werden. Der Arm besitzt eine Länge von der Tischplatte von 117,0 cm und eine Reichweite vom ersten Kippgelenk von 96,0 cm. Unter Verwendung eines 12 V Netzteils mit 222 W beträgt die maximale Tragkraft 1040 g. Dieser Wert beschreibt das äußerste Limit, da die Leistung der Motoren bei Last stark abnimmt und die Last je nach Lage des Arms ungleich auf die Servos verteilt wird. Die Leistungskurven der MX-Motoren sind in Abbildung 28 zu sehen. Wird ein zu schweres Objekt zu schnell bewegt, überhitzen die Motoren und schalten sich ab. Um der Überhitzung entgegenzuwirken sind an den Motoren die am stärksten belastet werden kleine Lüfter angebracht. Diese sind so befestigt, dass sie den Arbeitsbereich nur minimal beeinträchtigen. Sie verfügen über eine eigene Stromzufuhr und können einen Ausfall der Servos hinauszögern; ein dauerhafter Betrieb mit maximaler Last und Geschwindigkeit ist trotzdem nicht möglich. Hierfür muss entweder die Last und/oder das Arbeitstempo deutlich reduziert werden. Da die Geschwindigkeit maßgeblich an der Reaktionszeit des Arms beteiligt ist, werden nur sehr leichte Props mit einem Gewicht von max. 200 g verwendet.

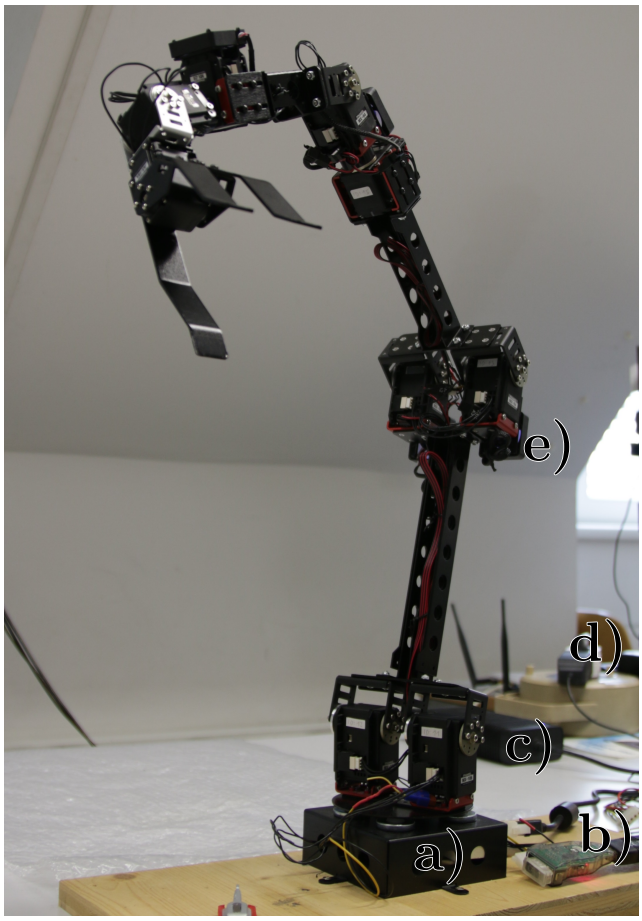
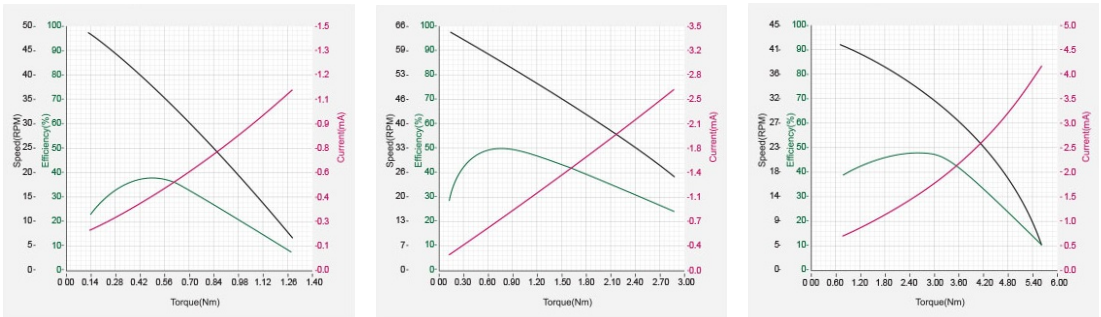


Abbildung 27: Komponenten des Roboterarms: a) 7-Achsen CrustCrawler Pro-Series Roboter Arm, b) Kommunikationsschnittstelle USB2Dynamixel, c) 12 V, max. 18,5 A Roboter-Netzteil (max. 222 W) mit Notausschalter, d) eigenständiges Netzteil zur Stromversorgung der Lüfter, e) zusätzliche Lüfter, um zu schneller Überhitzung vorzubeugen.



(a) MX-28T (b) MX-64T (c) MX-106T

Abbildung 28: Leistungskurven der MX-Motoren.[ROB15]

4.2 Framework

Als Basis für VR-Anwendungen, die das zuvor beschriebene Hardware-Setup einsetzen, wurde ein leicht zu verwendendes Framework erstellt. Es kapselt die Software-Komponenten der VR-Geräte und die Robotiklösung OpenRAVE zu einem Unity3D-Framework. Zusätzlich ist es möglich, den verwendeten Roboterarm zu einem späteren Zeitpunkt leicht auszutauschen. Ebenso kann auch nur ein Subset der unterstützten Hardwarekomponenten verwendet werden. Abbildung 29 zeigt die Software-Architektur des Frameworks: Die spezielle implementierte Steuerungsbibliothek (*OpenRAVE Library*) verwendet die C++-Schnittstellen von OpenRAVE. Sie abstrahiert die Komplexität der Schnittstellen und bietet Funktionen zur Steuerung der Roboter. Der Datenaustausch mit dem Roboter erfolgt über ein eigens implementiertes Controller-Plugin *CrustCrawler Controller*) über einen USB-Adapter (*USB2Dynamixel*). Eine C#-Klasse (*C# Wrapper*) kapselt diese um die Integration weiter zu erleichtern. Sie verwendet *PInvoke* um auf die exportierten Funktionen der OpenRAVE Library zuzugreifen und stellt diese für verwalteten Code wie .Net oder Mono zur Verfügung. Ein Unity3D-Script (*OpenRAVE Wrapper*) benutzt diesen Wrapper zur Kommunikation mit OpenRAVE und stellt selbst Funktionalitäten wie das parallele Planen von Pfaden während der Bewegung eines Roboters oder die Verwendung vorberechneter Pfade zur Verfügung. Die Daten der VR-Geräte werden ebenfalls von Unity3D-Scripten (*Leap VR Camera Control, Virtualizer Controller, Kinect Wrapper, Axis Wrapper*) über Kontrollprogramme (*LeapMotion Orion, Neuron Axis*) oder direkt über den Treiber von der Hardware (*Virtualizer, Kinect[1,2]*) ausgelesen. Die Scripte *Move Player* und *Avatar Controller* übertragen die getrackten Bewegungen an den Avatar.

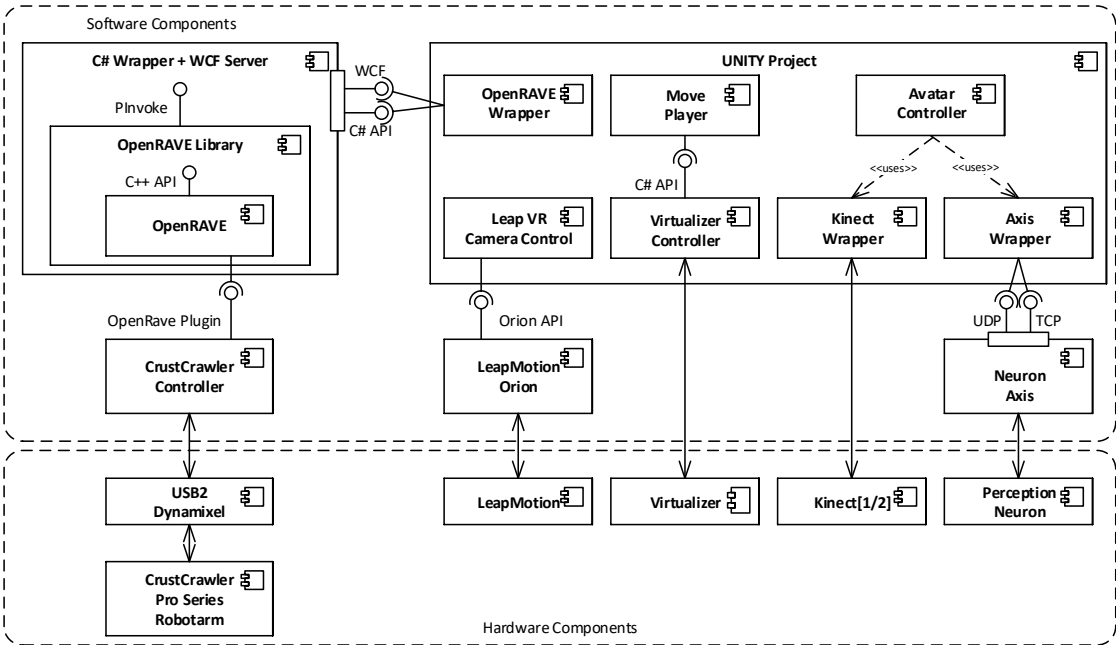


Abbildung 29: Abhängigkeiten der Softwarekomponenten.

Auf die einzelnen Komponenten des Frameworks wird in den folgenden Abschnitten im Detail eingegangen. Dabei erläutern Abschnitt 4.2.1 und Abschnitt 4.2.2 den Einsatz der Robotik-Software OpenRAVE als Steuerung und die Integration in die 3D-Engine Unity3D. Abschnitt 4.2.3 beschreibt die Entwicklung von VR-Anwendungen auf Basis dieses Frameworks anhand der Erstellung der Test-Szenarien.

4.2.1 Verwendung von OpenRave

Für die Abstraktion des Roboterarms wird OpenRAVE verwendet. Das Modell des Arms, das die Basis für die Generierung der IK darstellt, musste aufgrund der Einschränkungen des XML[Dia13a]- und COLLADA[Dia13c]-Formats in mehreren Schritten erstellt werden. Da das XML-Format wesentlich simpler ist, wurde damit der Aufbau des Arms beschrieben. Ein Auszug aus dem XML Modell ist in Listing 1 zu sehen. Hierbei werden einzelne Teile (*Bodies*) des Arms definiert, die aus einem oder mehreren 3D-Modellen bestehen. Die Modelle stammen entweder von Robotis[ROB10] oder wurden händisch erstellt. Letzteres war für die Props sowie für alle Komponenten notwendig die von CrustCrawler stammen, wie etwa die Verbindungsstreben oder den Drehteller. Diese Modelle dienen zum einen für die Visualisierung des Arms, zum anderen sind sie die Basis für die Kollisionsberechnung. Sie werden nicht in die XML-Datei eingebettet, sondern nur der jeweilige Dateipfad. Dadurch ist es möglich, dieselben Objekte öfters zu verwenden, ohne den Speicherplatzbedarf des Modells stark zu erhöhen. Die einzelnen Objekte werden beim Laden einer Szene in OpenRAVE für jeden Body zu einem Objekt kombiniert, das anschließend für die Kollisionsberechnung verwendet wird. Dadurch ist es nicht notwendig Überlappungen der einzelnen Teile innerhalb eines Bodies zu vermeiden, wodurch die Erstellung des Modells erleichtert wird. Diese Bodies werden anschließend durch *Joints* verknüpft. Zu den wichtigsten Eigenschaften eines Joints zählen die Rotationsachse (*axis*), die maximale Geschwindigkeit (*maxvel*), die maximale Beschleunigung (*maxaccel*) sowie der höchste und der niedrigste einnehmbare Rotationswert (*limitsdeg*). Ein aktivierter Joint entspricht dabei in der Regel einem Gelenk des Arms und somit einem oder mehreren Servomotoren. Die letzten beiden Werte werden allerdings durch die Geometrie des Arms übersteuert, das bedeutet, es können keine Drehwinkel eingenommen werden, in denen sich zwei oder mehr der 3D-Modelle überschneiden. OpenRAVE berücksichtigt hierbei Teile der 3D-Modelle die in einem Joint verbunden sind und sich bereits beim Laden überschneiden. Diese Überschneidungen zählen als Teil des Gelenks und schränken die Drehwinkel nicht ein. Im aktuellen Stadium muss für jede Requisite ein eigenes Modell erstellt werden, da sie den Zielpunkt des Arms verschieben und somit die IK verändern. In zukünftigen Versionen sollen die Props zur Laufzeit gewechselt werden können.

```
<?xml version="1.0" encoding="utf-8"?>
<KinBody name="TU-Crustcrawler_Holzplatte">
<Body name="Base" type="dynamic">
<Translation>0.0  0.0  0.0</Translation>
<Geom type="trimesh">
<data>meshes/base.WRL 1.</data>
</Geom>
</Body>
<Body name="Arm0" type="dynamic">
<offsetfrom>Base</offsetfrom>
<Translation>0.0 0.054 0</Translation>
<Geom type="trimesh">
<data>meshes/double_plate.WRL 1.</data>
</Geom>
<Geom type="trimesh">
<data>meshes/mx64t_106t_small_bracket.WRL 1.</data>
<Translation>-0.0035 0.002 0.038</Translation>
</Geom>
...
<RotationAxis>0 1 0 90</RotationAxis>
<!--HIERDURCH AENDERN SICH DIE ACHSEN! X->Z, Z->-X-->
</Body>
...
<Joint circular="false" name="Grip2" type="hinge">
<Body>GripperBase</Body>
<Body>MovingGripper</Body>
<offsetfrom>MovingGripper</offsetfrom>
<resolution>1</resolution>
<axis>1 0 0</axis>
<maxvel>0.7</maxvel>
<maxaccel>3.0</maxaccel>
<limitsdeg>4 -49</limitsdeg>
</Joint>
...
</KinBody>
```

Listing 1: Ausschnitt aus der XML-Definition des Roboterarms.

Das XML-Modell wird anschließend mit OpenRAVE in ein COLLADA-Modell umgewandelt, da es nur dort möglich ist, benutzerdefinierte Attribute bei Gelenken anzugeben. Dies ist notwendig, um spezielle Eigenschaften wie die Servo-ID oder den Stellwert in der Ausgangsstellung verfügbar zu machen. Aus der Ausgangsstellung kann der maximale Stellwert errechnet werden, da die Dynamixel-Motoren ihren Anfangswert immer bei der

Hälfte des Maximalwerts haben. Zusätzlich kann über diese Attribute auch der COM-Port festgelegt werden, über den mit dem Arm kommuniziert wird. Im Gegensatz zu XML-basierten Modellen sind im COLLADA-Modell die 3D-Objekte eingebunden, wodurch die Dateien deutlich größer sind. Dafür können sie einfach in anderen Programmen, wie etwa der 3D-Engine Unity3D, importiert werden. Um den benötigten Speicherplatz zu reduzieren, werden auch ZIP-komprimierte Dateien unterstützt. Diese benötigen allerdings eine etwas längere Ladezeit. Listing 2 zeigt die Definition der zusätzlichen Eigenschaften. Da bei den jeweiligen Gelenken unterschiedliche Servos zum Einsatz kommen, unterscheiden sich die Werte deutlich. Die COLLADA-Datei enthält alle Eigenschaften des Roboterarms, außer der Spezifikation des Endeffektors. Die Lage, Ausrichtung und der Aktionspunkt des Endeffektors werden in einer separaten XML-Datei beschrieben, die das COLLADA-Modell referenziert. Ist bereits ein IK-Löser für den Arm erzeugt worden, so kann der Pfad in dieser Datei ebenfalls angegeben werden. Dies ist notwendig, falls OpenRAVE nicht über ein Python-, sondern über ein C++-Programm geladen wird. Die Definition des Endeffektors und des IK-Lösers ist zwar auch in COLLADA möglich, dort ist diese aber deutlich umständlicher, weshalb die XML Variante bevorzugt wurde. Diese XML Datei bildet somit das vollständige Modell für den Roboterarm und dient als Basis für die IK-Berechnungen von OpenRAVE.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <COLLADA xmlns="http://www.collada.org/2008/03/COLLADASchema"
       version="1.5.0">
3    ...
4    <extra name="joint1" type="joint_info">
5      <technique profile="OpenRAVE">
6        <int_array name="dxl_id" count="2">11 12</int_array><!--
             DUAL JOINT, I.E. TWO MOTORS -->
7        <int_array name="zero_position" count="1">2048</int_array
             >
8      </technique>
9    </extra>
10   ...
11   <extra name="joint7" type="joint_info">
12     <technique profile="OpenRAVE">
13       <int_array name="dxl_id" count="1">19</int_array>
14       <int_array name="dxl_port_num" count="1">3</int_array>
15       <int_array name="zero_position" count="1">512</int_array>
16     </technique>
17   </extra>
18   ...
19 </COLLADA>
```

Listing 2: Benutzerdefinierte Eigenschaften im COLLADA-Modell des Roboterarms.

Der letzte Schritt ist die Modellierung der Umgebung für eine korrekte Kollisionsvermeidung. Dies geschieht erneut in XML, da dort mit einfachen Mitteln Objekte aus geometrischen Figuren erstellt werden können, ohne explizit ein 3D-Modell anzufertigen. Die Szene enthält neben dem Roboterarm alle relevanten Objekte: den Tisch, auf dem der Arm befestigt ist, den Virtualizer und einen Avatar, der mit dem Avatar aus der VR-Anwendung synchronisiert wird. Größe, Position und Ausrichtung der einzelnen Körperteile können dynamisch angepasst werden, um die aktuelle Haltung der Benutzerin oder des Benutzers einzunehmen. In Abbildung 30 ist eine Szene zu sehen, in der sich Arm und Avatar noch in ihrer Ausgangsstellung befinden. Alle angezeigten Objekte werden von OpenRAVE in der Pfadplanung berücksichtigt.

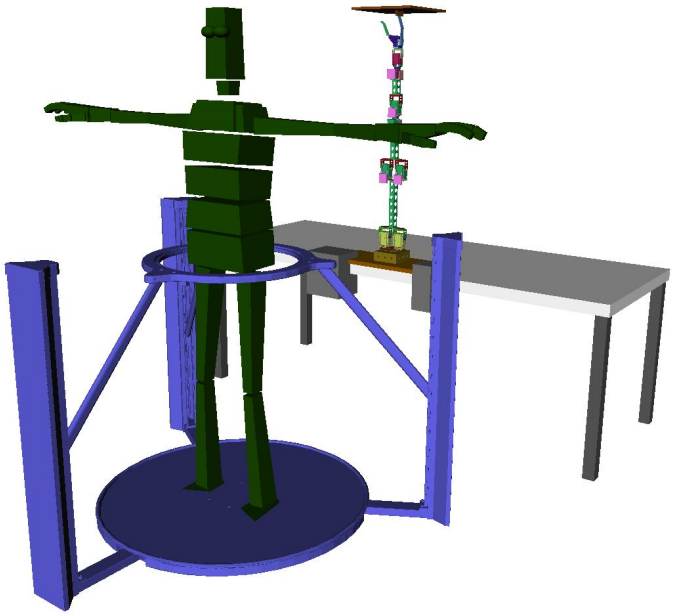


Abbildung 30: 3D-Modell der Umgebung in OpenRAVE.

Für schnelle Tests oder die Erstellung des IK-Lösers eignet sich die interaktive Python-Umgebung von OpenRAVE. Hierbei können direkt die Funktionen von OpenRAVE ausgeführt werden, ohne dass ein eigenes Programm erstellt werden muss. Eine direkte Integration in andere Programme ist jedoch nicht möglich. OpenRAVE stellt auch C++-Schnittstellen zur Verfügung, diese sind allerdings deutlich komplizierter. Daher wurde für die Integration eine eigene C++-Klasse (*OpenRAVE Library*) erstellt, die die Komplexität reduziert. Die Klasse ist ebenso wie OpenRAVE nur als 32 Bit Version verfügbar. Analog zu OpenRAVE werden auch bei der OpenRAVE Library die *Boost* C++-Bibliotheken[DAR16] eingesetzt. Die von OpenRAVE vorausgesetzte Version 1.4.4 weist Schwachstellen bei der Implementierung der *mutual exclusion* (*MUTEX*)-Mechanismen auf, die bei zu vielen gleichzeitigen Zugriffen zu einem Systemabsturz führen. Deshalb

wurde an den meisten Stellen auf die MUTEX der Win32-API (*Handle*) zurückgegriffen.

Wo es möglich war, wurde das *Ressourcen-Erwerb ist Initialisierung* (*RAII*)-Idiom umgesetzt, um sicherzustellen, dass verwendete Ressourcen auch wieder freigegeben werden. Hierbei wird die Anforderung einer Ressource dem Konstruktor und die Freigabe dem Destruktor des jeweiligen Objekts überlassen.[Wil05, Seite 166] Die meisten Klassen aus Boost sind ebenfalls nach dem RAII-Idiom konzipiert. Um die Win32-Handles ebenfalls nach diesem Prinzip verwenden zu können, wurde eine eigene RAII-Klasse für diese Zwecke erstellt. OpenRAVE kann innerhalb eines Prozesses nur einmal gestartet werden, dazu setzt die OpenRAVE Library auf das *Singleton*-Entwurfsmuster. Dadurch wird immer auf dieselbe OpenRAVE-Umgebung zugegriffen und auch die Zerstörung der Umgebung wird zentral gesteuert. Wurde die OpenRAVE-Umgebung zerstört, kann sie durch eine Einschränkung von OpenRAVE nicht mehr gestartet werden.

Eine weitere Einschränkung kommt von der verwendeten Visualisierungsbibliothek *QT-Coin*: Es kann nur einmal ein Betrachtungsfenster erzeugt werden; wird dieses geschlossen, so muss der gesamte Prozess beendet werden, um es erneut anzeigen zu können. Es ist allerdings nicht notwendig, das Fenster beim Wechsel einer Szene zu schließen, wodurch die Auswirkungen dieser Restriktion gering sind. Die Library ist weder in Bezug auf die Anzahl, noch auf die Typen der in der Szene verwendeten Roboter beschränkt, solange es sich um stationäre Arme handelt. Die einzige Voraussetzung ist, dass für jeden unterschiedlichen Robotertyp ein IK-Löser für Position und Endausrichtung (6D) erstellt wurde und dieser aus dem Modell referenziert wird.

In der OpenRAVE Library wurden folgende Funktionen für diese Aufgaben implementiert:

- **Laden einer Szene.** Dabei werden alle laufenden Bewegungen abgebrochen, jeder Roboter wird in seine Ausgangspose gebracht und verwendete Ressourcen werden freigegeben. Anschließend wird die neue Szene importiert und alle vorhanden Roboter nehmen ihre Ausgangslage ein. Wird das Simulationsfenster von OpenRAVE verwendet, so bleibt die Ausrichtung der Kamera unverändert.
- **Erzeugen eines Betrachtungsfensters zur Visualisierung der Szene.** Mithilfe dieses Fensters kann in der geladenen Szene navigiert werden um den aktuellen Zustand der darin befindlichen Objekte und Roboter zu erkennen. OpenRAVE verwendet diese Daten bei der Erkennung von Kollisionen. Durch eine Einschränkung von OpenRAVE kann dieses Fenster nur einmal geöffnet werden. Ein erneutes Öffnen ist nur möglich, wenn die Anwendung komplett neu gestartet wurde. Daher überprüft die Funktion, ob das Betrachtungsfenster seit dem Start der Applikation bereits geladen wurde. Ist dies der Fall wird kein Fenster geöffnet um Abstürze zu vermeiden.
- **Finden einer beliebigen IK-Lösung für die Zielausrichtung eines Roboters.** In den meisten Fällen ist dies auch die Lösung mit den geringsten Änderungen. Dazu verwendet die Library den geladenen IK-Löser aus OpenRAVE für den Roboter. In der aktuellen Version von OpenRAVE muss dieser bei der Verwendung der

C++-Schnittstelle bereits zuvor erstellt worden und in der XML-Datei eingetragen sein. Der Löser berücksichtigt alle in der Szene geladenen Objekte, sodass es zu keiner Kollision kommt. Einzig zwei durch ein Gelenk verbundene Teile eines Roboters sind von der Kollisionsüberprüfung ausgeschlossen, da Kollisionen durch die Einschränkungen der Gelenke verhindert werden. Dies erleichtert auch die Modellierung, da die einzelnen Teile des Gelenks überlappen können. Weiters wird sichergestellt dass die Limitierungen der Gelenke eingehalten werden.

- **Finden der IK-Lösung für die Zielausrichtung eines Roboters mit der geringsten Änderung zur aktuellen.** Diese Methode verwendet erneut den geladenen IK-Löser, der in der XML-Datei spezifiziert ist. Im Gegensatz zu der vorherigen Methode, errechnet sie aber alle möglichen Lösungen, die keine Kollision verursachen und die Limitierungen einhalten. Anschließend vergleicht sie die neuen Winkel mit den aktuell eingenommenen und wählt jene Konfiguration, die die kleinste Änderung aufweist. Die Ausführung dieser Methode dauert dadurch erheblich länger.

- **Erzeugen einer IK-Lösung für die Ausgangsposition eines Roboters.** Dabei wird jedes Gelenk auf 0 gesetzt. Diese Methode verwendet keinen IK-Löser, da die Ausgangsposition immer dieselbe ist und bereits beim Laden einer Szene automatisch eingenommen wird. Ebenso müssen die Limitierungen der Gelenke nicht berücksichtigt werden.

- **Öffnen und Schließen des Greifers.** Dazu werden alle Gelenke des Greifers anhand der in der XML-Datei spezifizierten Öffnungsrichtung (*closingdirection*) geöffnet bzw. geschlossen, bis entweder eine Kollision auftritt oder die Limitierungen der Gelenke erreicht werden. Eine Kollision tritt in der Regel dann auf, wenn ein Objekt erfasst wird. Für diese Operation wird kein IK-Löser benötigt.

- **Auslesen der Gelenkwinkel eines Arms.** Dadurch lässt sich die Ausrichtung eines Roboters an andere Programme übertragen. Diese Funktion wird auch bei der Berechnung von Pfaden zu einer Zielposition verwendet.

- **Bewegen eines Roboters zu einer bestimmten Pose.** Diese Methode berechnet eine IK-Lösung, erzeugt einen Pfad zwischen der aktuellen Haltung und der Zielhaltung. Gibt es eine Lösung und kann der Pfad ohne Kollision befahren werden, wird der Pfad an das Kontroll-Plugin (*Controller*) übergeben, der die tatsächlichen Steuerkommandos versendet.

- **Überprüfung, ob ein Roboter zu einer bestimmten Pose bewegt werden kann.** Diese Methode führt dieselben Schritte wie jene zuvor aus, ohne den Pfad aber an den Controller zu übergeben. Sie kann dazu verwendet werden mögliche Zielpositionen zu überprüfen und eine daraus auszuwählen.

- **Bewegen eines Roboters zu seiner initialen Pose.** Diese Methode erzeugt einen Pfad zwischen der aktuellen Haltung und jener in der Ausgangshaltung. Dazu

werden die Gelenkwinkel auf 0 gestellt. Gibt es einen kollisionsfreien Pfad, wird dieser an den Controller übergeben, der die Steuerkommandos versendet.

- **Planen eines Pfades zwischen zwei Gelenkkonfigurationen.** Dies geschieht unter Einhaltung der Geschwindigkeits- und Beschleunigungsbeschränkungen. Der Pfad ist eine quadratische Interpolation zwischen den beiden Positionen. Dadurch ist die Geschwindigkeit zu Beginn und Ende langsamer, was eine stabilere Bewegung bewirkt. Bei der Pfadberechnung wird erneut überprüft, ob weder Ziel- noch Ausgangskonfiguration eine Kollision hervorrufen. Etwaige Hindernisse entlang des Pfades werden umfahren. Der Pfadplaner von OpenRAVE ist in der Lage, die Pfade mehrerer Roboter zu synchronisieren, sodass es keine Beeinflussung gibt. Dazu müssen aber alle diese Pfade in einem gemeinsamen Funktionsaufruf erstellt werden.

- **Ausführen eines bereits geplanten Pfades für einen Roboter.** Hierbei wird vor der Ausführung nochmals überprüft, ob der Pfad ohne Kollision befahren werden kann. Ist dies der Fall, so wird der Pfad an den Controller der Roboters übergeben. Dieser berechnet die nächsten Stellwerte der Motoren anhand der Interpolation und übermittelt sie an den Roboter.

- **Verändern der Position und Größe eines Elements aus der Szene.** Da dies Auswirkungen auf die Kollisionsprüfungen hat, werden etwaige IK- oder Pfadberechnungen blockiert, bis der Vorgang abgeschlossen ist. Größenänderungen sind nur beim Laden eines 3D-Objekts möglich, daher wird zuerst überprüft ob das Objekt erneut geladen werden kann. Dazu muss es als einzelnen Objekt in der Szenendatei definiert sein und darf in keiner Hierarchie enthalten sein. Ist dies der Fall, wird das Objekt zerstört und mit der neuen Skalierung neu geladen. Positionsänderungen erfordern ebenfalls die Sperre der OpenRAVE-Umgebung, allerdings müssen die Objekte nicht zerstört werden. Dies kann u.a. dazu verwendet werden, einen Avatar zu synchronisieren.

- **Ändern und Zurücksetzen der Limitierungen für Beschleunigung, Geschwindigkeit sowie größter und kleinster einnehmbarer Winkel eines Gelenks.** Bei der ersten Änderung eines Limits, wird der ursprüngliche Wert gespeichert. Dadurch ist es möglich, diesen Wert wiederherzustellen. Die Limitierungen für Geschwindigkeit und Beschleunigung können frei gewählt werden. Die Einschränkungen des Winkels bleiben aus Sicherheitsgründen innerhalb der Originalwerte. Diese Limits werden bei der Planung und Kollisionsüberprüfung berücksichtigt, dadurch wird bei einer Änderung die OpenRAVE-Umgebung gesperrt. Die Limitierungen finden nur in OpenRAVE Anwendung und werden nicht an die Motoren übertragen, da die Limits dort je nach verwendetem Modell nur sehr grob oder gar nicht eingestellt werden können. Zusätzlich trägt dies dazu bei, dass der Roboter sich nicht langsamer bewegt, als es die Planung vorsieht und so die Simulation mit der Realität synchron bleibt.

- **Warten solange ein Roboter sich bewegt.** Diese Funktion liest in einer Schleife die Controller aller vorhandenen Roboter ab und überprüft, ob sich dieser in Bewegung befindet. Die Schleife wird erst verlassen, wenn sich kein Roboter bewegt. Dies dient vor allem dazu, erst dann einen neuen Pfad zu beginnen, wenn die Zielposition eingenommen ist. Auch die Synchronisation zwischen mehreren Robotern kann durch diese Funktion bewerkstelligt werden.

- **Zerstören der Umgebung und Freigabe der Ressourcen.** Diese Funktion stellt sicher, dass die Library korrekt beendet wird. Dazu zählt, dass zuerst alle Bewegungen von Robotern eingestellt werden und die Ausgangspositionen eingenommen werden. Erst danach wird die Verbindung über die Controller getrennt, da sonst ein unvorhersehbares Verhalten auftreten kann. Zusätzlich wird gewartet bis alle Arbeitsprozesse von OpenRAVE abgeschlossen sind, damit keine Ressourcen über die Lebensdauer der Anwendung belegt bleiben. Wird diese Funktion nicht verwendet, kann es passieren, dass Controller bestehen bleiben und somit keine erneute Verbindung zu einem Roboter aufgebaut werden kann.

Es wurde von allen Methoden zur Roboterkontrolle auch eine Version implementiert, die mehrere Roboter gleichzeitig bearbeitet. In diesem Fall wird bei der Pfadplanung auch der Pfad der anderen Roboter mitberücksichtigt, sodass es zu keiner Kollision kommt.

Wie zuvor erwähnt, unterscheidet die Library nicht zwischen Robotern mit unterschiedlichem Aufbau. Die einzige Voraussetzung für eine Verwendung ist, dass die Geometrie des Roboters von OpenRAVE gelesen werden kann, um einen IK-Löser zu erstellen. Die spezifischen Funktionen für einen Arm werden im Kontroll-Plugin abgebildet. Die Aufgabe des Controllers ist das Ausführen eines Pfades, das Setzen einer bestimmten Gelenkkonfiguration sowie das Auslesen der tatsächlichen Gelenkwerte. Für den in Abschnitt 4.1.2 beschriebenen Arm wurde im Zuge dieser Arbeit ein eigener Controller erstellt. Dieser verwendet das asynchrone, serielle Kommunikationsprotokoll 1.0 des Dynamixel SDK. Wird ein Pfad übergeben, so erzeugt der Controller interpolierte Zwischenkonfigurationen und überträgt diese in festen Abständen an die Servomotoren. Dabei wird immer ein Paket an alle Motoren gesendet, damit sie die Bewegung gleichzeitig ausführen. Um die korrekten Abstände zwischen den Paketen zu gewährleisten, setzt der Controller die Auflösung des Systemzeitgebers. Die IDs der Motoren sowie den Wert in der Ausgangsstellung liest der Controller aus dem Robotermodell aus. Ist im Modell kein COM-Port definiert, so wird der Standardport des USB2Dynamixel (COM3) verwendet. Enthält eine Szene mehrere Roboter, so müssen unterschiedliche COM-Ports vergeben werden. Beim Starten und Beenden des Controllers wird der Arm in die Initialposition gebracht. Abbildung 31 zeigt das Zusammenspiel zwischen OpenRAVE, der OpenRAVE Library, dem Controller und dem CrustCrawler Pro-Series Roboterarm. Das Controller-Plugin wird, wie alle anderen Plugins auch, von OpenRAVE beim Öffnen einer Szene geladen, wenn dort die Verwendung des Controller definiert ist. Ist kein spezieller Controller bei einem Roboter hinterlegt, so wird der Standardcontroller verwendet. Dieser simuliert nur die Bewegung innerhalb von OpenRAVE, übermittelt aber keine Daten an die Hardware.

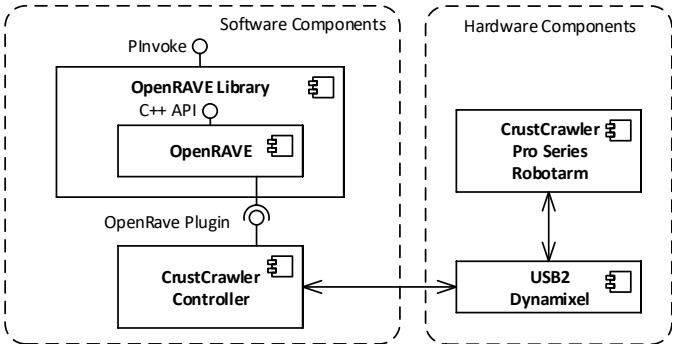


Abbildung 31: Struktur der Robotersteuerung.

4.2.2 Integration in Unity3D

Die im vorherigen Abschnitt beschriebene Programmbibliothek bietet eine gute Basis für die Integration der Steuerung in andere Umgebungen, die auf C++ basieren. Unity3D hingegen verwendet als Grundgerüst die quelloffene .Net-Alternative *Mono*. Um auch aus verwaltetem Code einfach auf die Library zugreifen zu können, wurde eine C#-Klasse erstellt, die die native Library kapselt (C#-Wrapper). Dazu exportiert die Library die meisten ihrer Funktionen in mehreren Ausführungen, sodass aus C# heraus mittels PInvoke darauf zugegriffen werden kann. Normalerweise werden hierzu statische Funktionen für jede zu exportierende Methode definiert, denen ein Objekt der nativen Klasse selbst übergeben wird. Auf diesem Objekt ruft die statische Funktion anschließend die nicht statische Version mit denselben Parametern auf. Um die Anzahl dieser zusätzlichen Methoden möglichst gering zu halten, wurde in dieser Arbeit der in [Spr08] angeführte Ansatz verfolgt. Hierbei wird die gesamte Klasse exportiert, die Einbindung in verwaltetem Code wird anschließend über so genannte *THIS-Calls* durchgeführt. Dazu ist die Definition eines Einsprungpunktes mit dem exportierten Namen der jeweiligen Funktion notwendig. Dieser enthält den voll-qualifizierten Namen der Klasse, den Namen der Methode sowie eine kodierte Auflistung der Datentypen der Parameter. Zusätzlich muss ebenso eine Instanz der Zielkasse übergeben werden. Neben dem Wegfall der zusätzlichen statischen Methoden bietet diese Variante auch eine geringfügig bessere Performance, die ebenso auf den Verzicht eines weiteren Aufrufs zurückzuführen ist. Ein weiterer Vorteil ist, dass durch die Verwendung der qualifizierten Einsprungpunkte alle Varianten einer überladenen Funktion referenziert werden können, ohne neue Namen vergeben zu müssen. Während die THIS-Call-Methoden problemlos aufgerufen werden können, stellte sich im Laufe der Verwendung eine schwerwiegende Einschränkung unter Mono heraus: Bei der Verwendung von nicht trivialen Datentypen, also etwa Zeichenketten, Zeigern oder Arrays, werden die Parameter nicht korrekt übergeben. Die einzige Lösung für dieses Problem ist die Verwendung statischer Funktionen, weshalb hierfür zusätzliche Methoden angelegt wurden. Diese wurden allerdings nur für jene Funktionen angelegt, bei denen es aufgrund der Datentypen der Parameter notwendig war. Die C# Klasse implementiert sowohl die THIS-Calls als auch die statischen Funktionen, wobei vor der

Verwendung festgelegt werden kann, welche Variante gewählt werden soll. Listing 3 zeigt die Verwendung einer Methode über einen THIS-Call und eine statische Hilfsfunktion. Die Definition von THIS-Call und statischer Methode unterscheidet sich nur durch die zusätzliche Angabe des Einstiegspunkts.

```
1  // THIS-Call
2  /// <summary>Closes the fingers of the robot with the specified
       name in the scene.</summary>
3  [DllImport("crustcrawler_or_unmanaged_library", EntryPoint = "?
     CloseFingers@CrustCrawlerORLib@crustcrawler@tuwien@at@
4  @QAE_NPAD@Z", CallingConvention = CallingConvention.ThisCall)]
5  static private extern bool CloseFingers(IntPtr thisObj, string
     robotName);
6  ...
7  // static wrapper function
8  /// <summary>Closes the fingers of the robot with the specified
     . Static ddl export wrapper.</summary>
9  [DllImport("crustcrawler_or_unmanaged_library")]
10 static private extern bool CloseFingersStatic(IntPtr nativeObj,
       string robotName);
11 ...
12 //Function definition with runtime decision
13 /// <summary>Closes the fingers of the robot with the specified
       name in the scene.</summary>
14 public bool CloseFingers(string robotName)
15 {
16   if (UseStaticForStrings)
17     return CloseFingersStatic(nativeLib, robotName);
18   return CloseFingers(nativeLib, robotName);
19 }
```

Listing 3: Definitionen externer Funktion über *THIS-Call* und statische Methode.

Um den Singleton-Ansatz der nativen Library nachzuverfolgen, setzt der C#-Wrapper ebenfalls auf dieses Pattern. Zusätzlich zu den gekapselten Funktionen bietet die C#-Variante auch statische Methoden an, um Protokollierungsausgaben zu verwalten und zwischen statischen Funktionen und THIS-Calls zu wechseln. Da die Berechnungen für die Robotersteuerung sehr aufwändig sein können, ist es in manchen Fällen sinnvoll, OpenRAVE auf einem anderen Computer auszuführen. „*Windows Communication Foundation (*WCF*) ist ein Framework zur Erstellung dienstorientierter Anwendungen. Mit WCF können Sie Daten als asynchrone Nachrichten von einem Dienstendpunkt an einen anderen senden.*“ [Mic15e] Aus diesem Grund wurde im C#-Wrapper auch eine *WCF*-Schnittstelle implementiert. Dadurch muss bei der Integration in andere Programme nicht darauf geachtet werden, ob OpenRAVE lokal oder auf einem entfernten Computer läuft, da die Schnittstelle dieselbe ist.

Unity3D verwendet zur Kapselung einzelner Komponenten eigene C#-Klassen *(Scripts)*, die zusätzlich benötigte Funktionen für die Engine bereitstellen. Abbildung 32 zeigt jene Scripts die in einer mit diesem Framework erstellten virtuellen Welt verwendet werden. Ein Großteil davon musste im Zuge der Diplomarbeit erstellt oder adaptiert werden (weiß). Nur vier Scripts konnten unverändert integriert werden (grau).

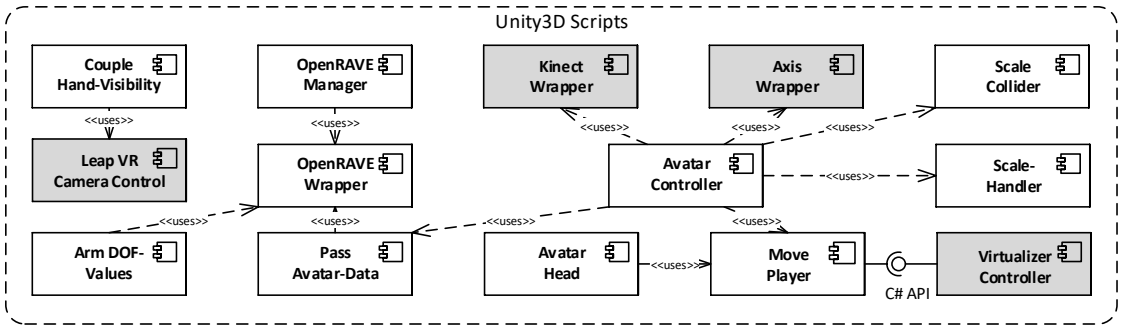


Abbildung 32: Neu angelegte (weiß) und unverändert eingesetzte (grau) Unity3D Scripts.

Das OpenRAVE-Unity-Wrapper-Script (*OpenRAVE-Wrapper*) ist die zentrale Komponente für die Kommunikation mit dem Roboterarm und verwaltet den Zugriff auf den C#-Wrapper. Alle Objekte, die Funktionen von OpenRAVE verwenden wollen, müssen dieses Script verwenden. Dabei kann sich ein Objekt den exklusiven Zugriff sichern, sodass kein anderes Objekt den Zustand des Roboters verändern kann. Der OpenRAVE-Wrapper übernimmt dabei die Aufrufe der Pfadplanung und überwacht die Bewegungen aller vorhandenen Roboter. Tritt während eines Pfades eine Kollision ein, so stoppt OpenRAVE den Pfad. Das Wrapper-Script versucht anschließend den Pfad wiederaufzunehmen. Die Anzahl an Wiederholversuchen kann frei definiert werden. Besteht die Kollision weiter, so wird der Pfad abgebrochen. Dauert eine Bewegung länger als eine bestimmte Zeitspanne, so wird dies registriert und kann für die Anzeige eines Ladesymbols verwendet werden. In der aktuellen Umsetzung beträgt diese Dauer zwei Sekunden. Das Ladesymbol ist in Abbildung 33 zu sehen.

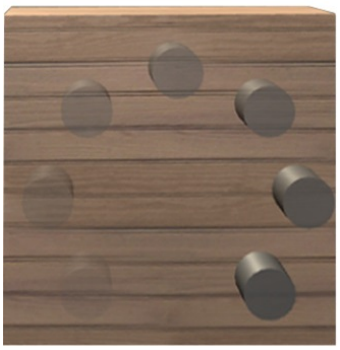


Abbildung 33: Animiertes Ladesymbol vor einer berührbaren Holzplatte.

Wenn es der Anwendungsfall zulässt, kann der nächste Pfad bereits geplant werden, während ein Arm noch in Bewegung ist. Bei wiederkehrenden Bewegungen kann auch ein vorberechneter Pfad übergeben werden, um die Reaktionszeit zu verbessern. Bei der Verwendung eines vorberechneten Pfades überprüft das Script, ob der zu bewegende Arm tatsächlich in der entsprechenden Ausgangsposition ist, andernfalls wird ein neuer Pfad von der aktuellen Position berechnet. Die IK für die Zielposition muss hierbei nicht errechnet werden, da sie aus dem Pfad extrahiert werden kann. Zusätzlich kümmert sich das Wrapper Script um die Übertragung der Daten des Avatars von Unity3D nach OpenRAVE. Dazu verwendet es das *Pass Avatar-Data* Script, das anhand des importieren COLLADA-Modell der OpenRAVE-Szene die einzelnen Bestandteile korrekt ausrichtet. Dieses Modell dient ebenso der Visualisierung der Roboter, da sämtliche Gelenkwerte vom OpenRAVE-Wrapper ausgelesen werden. Das *Arm DOF-Values* Script extrahiert diese Werte und richtet den importieren Roboterarm entsprechend aus. Da in der aktuellen Implementierung für jede verwendete Requisite eine neue OpenRAVE-Szene erstellt werden muss, ist es hilfreich mehrere COLLADA-Modelle zu importieren. Der *OpenRAVE-Manager* unterstützt dieses vorgehen indem er die Suche nach dem zu verwendenden Wrapper abstrahiert. Dazu müssen alle Modelle innerhalb eines gemeinsamen Objekts in Unity3D importiert werden. Anschließend werden alle Modelle bis auf das relevante deaktiviert. Der Manager sucht beim Laden der Umgebung nach dem einen aktiven Wrapper Script und stellt es anderen Scripts zur Verfügung. Da es nur einen Manager pro Unity3D Szene gibt, ist es ausreichend dieses Objekt beim Ladevorgang zu suchen und darüber auf den OpenRAVE-Wrapper zuzugreifen. Alle zuvor genannten Scripts wurde im Zuge der Diplomarbeit implementiert.

Die weiteren Scripts des Unity3D-Frameworks dienen der Steuerung des Avatar. Der speziell erstellteAvatar Controller verwendet die Scripts aus bestehenden Frameworks für Microsoft Kinect (*Kinect Wrapper*) und Perception Neuron (*Axis Wrapper*) um die Gelenkwinkel in die virtuelle Welt zu übernehmen. Aufgrund der höheren Genauigkeit werden die Daten des Perception Neuron bevorzugt, die Daten der Kinect werden nur als Backup-Maßnahme verwendet. Die Daten des Perception Neuron Anzugs können nicht direkt auf einen beliebigen Avatar übertragen werden, daher werden sie zuerst auf einen nicht sichtbaren Hilfsavatar aus dem Perception Neuron *Software Development Kit* (*SDK*) kopiert. Von diesem Avatar können anschließend die Daten übernommen werden, da sie von Unity3D bereits in das notwendige Format umgewandelt wurden. Müssen die Daten von der Kinect verwendet werden, muss berücksichtigt werden, dass die Daten nur dann eine ausreichende Qualität aufweisen, wenn die Kinect dem Benutzer bzw. der Benutzerin von vorne sieht. Dazu wurde ein frei definierbarer Rotationsbereich implementiert, in dem die Daten übernommen werden. Dreht sich die Benutzerin bzw. der Benutzer weiter, so wird der Avatar in eine aufrechte Position gebracht und die Arme werden nach unten abgewinkelt. Dadurch kommt es zu weniger fehlerhaften Planungen in OpenRAVE. Die Rotation wird dabei aus dem *Move Player* Script ausgelesen. Dieses wurde speziell implementiert um unterschiedliche Eingabegeräte für die Bewegung des Avatars zu ermöglichen. Die primäre Quelle der Daten ist der Virtualizer, es werden aber auch Eingaben per GamePad oder Tastatur unterstützt. Bei Verwendung des

Virtualizers ist es auch möglich zu springen oder sich zu ducken. Der Avatar Controller liefert neben den Positionsdaten auch die Größe des Anwenders oder der Anwenderin, um den Avatar entsprechend zu skalieren. Diese Größe kann entweder automatisiert mithilfe einer Kinect ermittelt werden, oder sie wird bei der Kalibrierung händisch eingetragen. Für die Skalierung der einzelnen Teile des Avatar sowie der Objekte zur Kollisionserkennung in Unity3D (*Collider*) wurden ebenfalls zwei neue Scripts erstellt. Der *Scale Handler* übernimmt die Größenanpassung der einzelnen Körperteile und stellt dabei sicher, dass die Gelenke korrekt verschoben werden. Auch die Collider müssen zusätzlich zur Skalierung entsprechend neu positioniert werden, was durch das *Scale Collider* Skript passiert. Die virtuellen Augen des Avatars werden von einem eigenen Script (*Avatar Head*) gesteuert. Dieses vereint die Daten des Move Player Scripts mit jenen des HMD. Die Ausgangsposition wird dazu anhand des Avatars berechnet, dazu wird die Veränderung addiert, die die Infrarotkamera des DK2 liefert. Als optionale Erweiterung des Avatars dienen die Hände aus dem Leap Motion Orion SDK. Erkennt die Kamera eine Hand, so wird die Hand des Avatars durch eine aus dem SDK ersetzt. Das Ausblenden der Gliedmaßen übernimmt das eigens dafür erstellte *Couple Hand-Visibilty* Script.

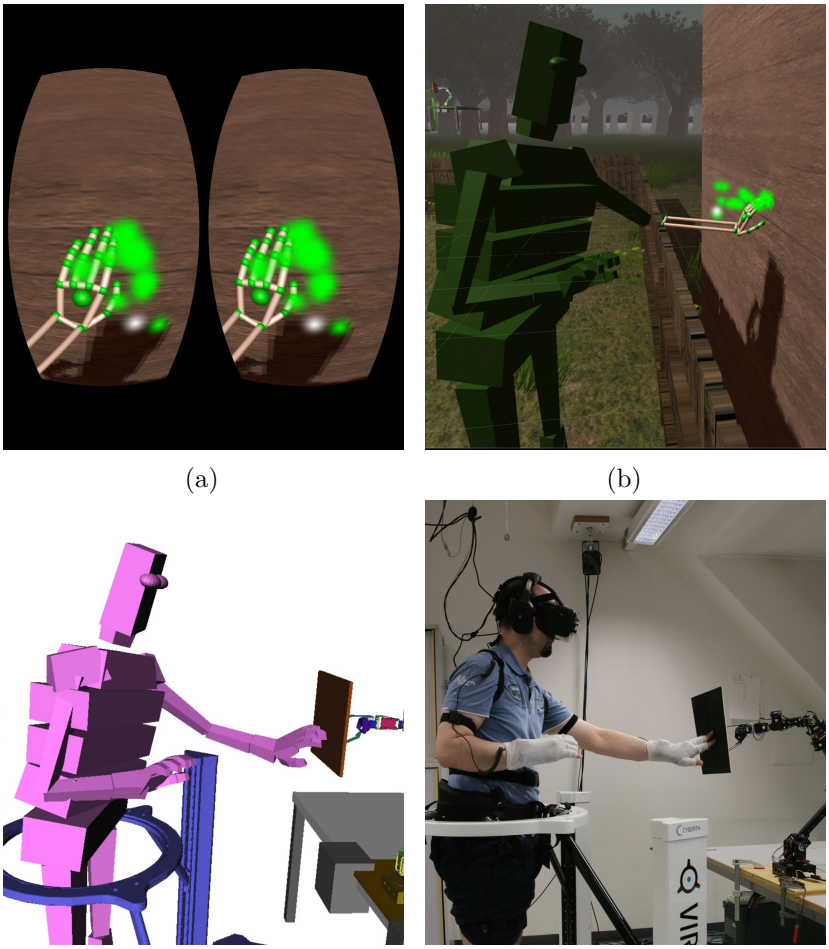
4.2.3 Entwicklung der Testszenarien

Um die Erstellung einer virtuellen Umgebung mit diesem System zu erleichtern, existiert eine Beispiel-Welt. Darin sind bereits alle Scripte konfiguriert und können als Basis für ein weiteres Projekt exportiert werden. Die Szene enthält außerdem Scripte, die mögliche Anwendungsfälle, wie etwa das Verwenden vorberechneter Pfade, Wartepositionen oder relative Positionierung, demonstrieren. Für die in Abschnitt 5.2 beschriebene Benutzerstudie wurden zwei VR-Anwendungen anhand dieses Frameworks erstellt.

Für die erste Aufgabe wurde ein virtueller Garten erstellt. An einer Seite des Zauns befindet sich eine 2,65 m hohe und 5 m breite Holzwand. Entlang dieser Holzwand werden Zielpunkte verteilt, von denen eine definierbare Anzahl zufällig ausgewählt wird. Die Auswahl kann so erfolgen, dass sie bei jedem Start der Anwendung dieselbe Folge liefert. Für dieses Szenario werden 7 Punkte ausgewählt. Diese Punkte entsprechen den Holzplatten aus der Beispiel-Welt und verfügen ebenfalls über das animierte Ladesymbol.

Für das haptische Feedback, welches die Holzwand repräsentiert, ist an der Spitze des Roboterarms eine 30 cm hohe und 30 cm breite Holzplatte befestigt. Befindet sich der Avatar weit vom Ziel entfernt, ist der Arm in einer aufrechten Position, um die Belastung für die Motoren gering zu halten. Nähert sich der Avatar, so wird die Holzplatte in eine Warteposition bewegt, deren Höhe mit der des Ziels übereinstimmt. Ab diesem Zeitpunkt beansprucht das Ziel exklusiven Zugriff auf den Roboterarm, wodurch gewährleistet wird, dass nicht zwischen zwei Positionen gesprungen wird. Die Platte wird, abhängig von der Position des Avatars, links oder rechts vor dem Arm positioniert, damit der Weg zur ersten tatsächlichen Position gering ist. Sieht die Benutzerin bzw. der Benutzer zum Beispiel den Punkt rechts von sich, so wird auch eine Warteposition, aus Sicht des Benutzers bzw. der Benutzerin, auf der rechten Seite eingenommen. Wird die Entfernung erreicht, ab

der eine Berührung möglich ist, positioniert der Roboterarm die Platte so, dass sich ihr Mittelpunkt mit dem Zielpunkt deckt. Um ein realistisches Feedback zu erhalten, müssen die Hände genau getracked werden, da sonst die Gefahr groß ist, an der Platte vorbei zu greifen. Daher wird zum zusätzlichen Handtracking die Leap Motion eingesetzt. Ist diese Pose erreicht, beginnen die Punkte stärker zu sprühen, da erst ab diesem Zeitpunkt etwas an der angezeigten Stelle zu spüren ist. Dazu verwendet das System die relative Position des Punkts im Hinblick auf die Position des Avatars in der virtuellen Welt. Ändert sich die Position des Avatars, so wird ein Pfad zu der neuen Zielposition berechnet, der auch die Körperteile der Testperson berücksichtigt und gegebenenfalls umfährt. Dauert die Bewegung lange, wird das Ladesymbol angezeigt.



(a) (b)

(c) (d)

Abbildung 34: Unterschiedliche Sichten des Systems beim Berühren eines Ziels: a) Sicht auf die Wand und das Ziel durch das DK2, b) Avatar vor der virtuellen Wand in Unity3D, c) Avatardaten inklusive der relativen Position in OpenRAVE, d) Positionierung der Holzplatte durch den Roboterarm in der realen Welt.

Um die Reaktionszeit zu verkürzen, wird ein etwaiger Folgepfad bereits berechnet, während der aktuelle Pfad noch befahren wird. Hierbei versucht das System immer, die aktuellste Position einzunehmen. Ist dies nicht möglich, wird sie verworfen und die nächst ältere wird versucht. Kommt währenddessen eine neue Position hinzu, wird sie zu Beginn der Warteschlange eingefügt. Kann eine Position eingenommen werden, so werden alle älteren Positionen verworfen. Dieses Vorgehen ist vor allem für das erste Zubewegen auf das Ziel notwendig, da sich hierbei die Position des Avatars sehr schnell ändert und sonst viele Zwischenschritte notwendig wären. Letztere können dadurch in den meisten Fällen übersprungen werden, was auch eine gleichmäßigere Bewegung des Arms zur Folge hat.

Berührt die Testperson die Platte oder versperrt dem Roboterarm den Weg, so wird aus Sicherheitsgründen keine Pfadplanung durchgeführt. Entfernt sich die Testperson wieder weiter von einem Ziel oder wird es erfolgreich berührt, begibt sich der Arm je nach Entfernung wieder in die neutrale Stellung oder in die Wartestellung. Abbildung 34 zeigt die Funktionsweise des Systems bei Berührung eines Punktes.

Die Anwendung für die zweite Aufgabe wurde nicht vom Autor erstellt. Da sie allerdings weitere Konzepte des Frameworks aufgreift, wird sie zur Vollständigkeit beschrieben. Für diese Anwendung wurde ein virtueller Häuserblock erstellt. Entlang eines speziellen 76 m langen Stücks des Gehwegs, begegnen der Testperson 10 Passanten. 4 Passanten weichen immer aus, die restlichen sind auf einen Kollisionskurs programmiert.

Für das haptische Feedback, das eine Kollision an der Schulter simuliert, wird ein weicher, nachgiebiger Boxhandschuh an der Spitze des Roboterarms verwendet. Im Gegensatz zu der Holzplatte aus dem ersten Szenario wird nur der Roboterarm, nicht jedoch der Handschuh in der Kollisionsvermeidung berücksichtigt, da die Testperson bewusst berührt werden soll. Der Zielpunkt für die Pfadberechnungen wurde daher entsprechend der Länge des Handschuhs um 200 mm versetzt. Da der Arm in dieser Aufgabe immer nur eine der Schultern treffen kann, werden zwei vorberechnete Pfade verwendet, anstatt bei jeder Kollision denselben Pfad erneut zu erstellen. Dadurch ist eine deutlich kürzere Reaktionszeit möglich, eine Gefährdung der Testperson liegt allerdings nicht vor, da, wie in Abschnitt 4.2.1 und Abschnitt 4.2.2 erwähnt, sowohl überprüft wird, ob die Ausgangsposition übereinstimmt, als auch, ob kein Hindernis den Pfad blockiert.

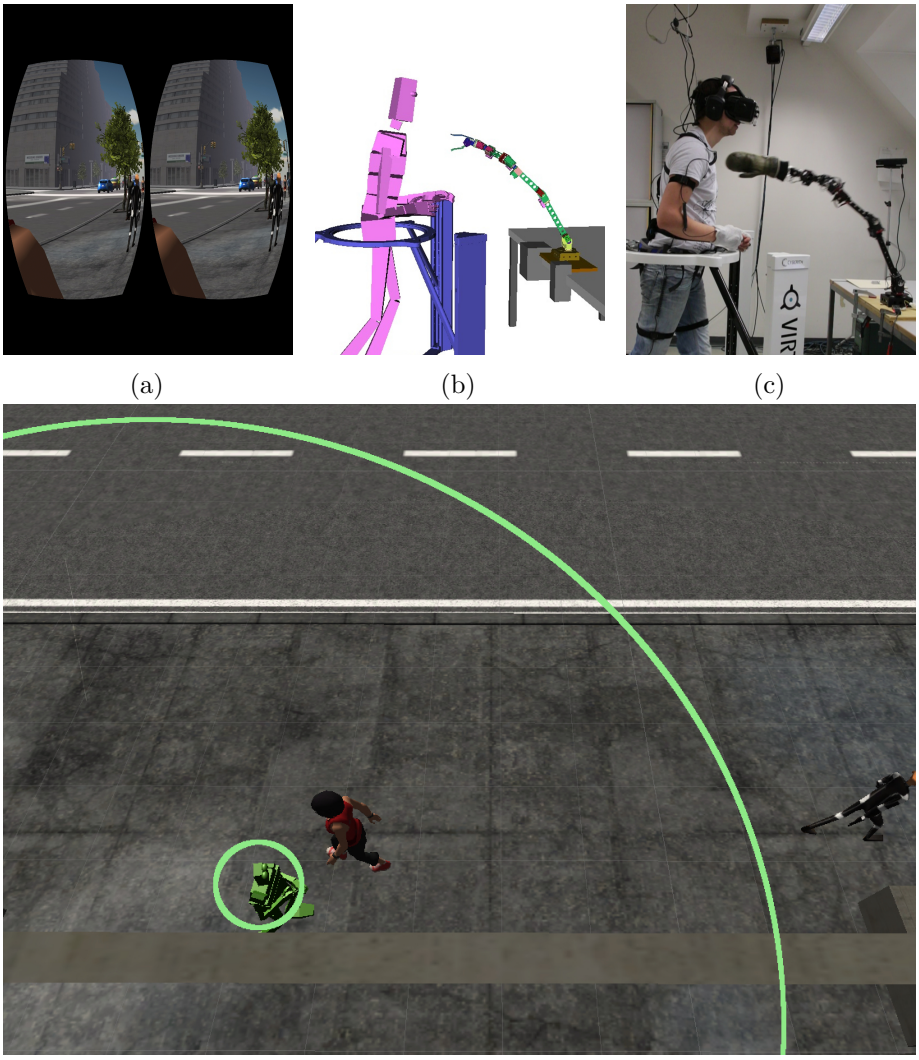
Eine Kollision mit einem Passanten besteht aus drei Stufen:

1. Nähert sich der Passant bis auf 5 m, so wird der Roboterarm in eine Warteposition versetzt, von der aus die beiden vorberechneten Pfade starten. Wie auch in der Gartenszene wird ein exklusiver Zugriff auf den Roboterarm erzwungen.

2. Auf Basis der relativen Geschwindigkeit, in der sich Testperson und Passant aufeinander zubewegen, wird berechnet, ab wann der Arm den Pfad befahren muss, damit die virtuelle Kollision mit der tatsächlichen übereinstimmt. Zu diesem Zeitpunkt wird auch die getroffene Schulter festgelegt und der Roboter beginnt, den entsprechenden Pfad zu befahren.

3. Nach der Kollision kehrt der Arm wieder entlang eines vorberechneten Pfades in die Ausgangsstellung zurück.

Abbildung 35 zeigt eine kurz bevorstehende Kollision in den verschiedenen Systemteilen. In Abbildung 35d sind die Entfernungen markiert, bei denen eine Aktion des Roboters ausgelöst wird. Der äußere Kreis gibt an, ab wann eine Kollision initiiert wird. Wird der innere Kreis erreicht, sollte der Pfad abgeschlossen sein.



(a) (b) (c)

(d)

Abbildung 35: Unterschiedliche Sichten des Systems bei einer Kollision in der Stadt: a) Sicht auf entgegenkommenden Avatar durch das DK2, b) Avatardaten in OpenRAVE, während der Roboterarm sich in Richtung Schulter bewegt, c) Roboterarm mit dem Boxhandschuh kurz vor der Berührung, d) Avatar kurz vor der Kollision.

4.3 Sicherheitsvorkehrungen

Da der Roboterarm im beschriebenen System direkt mit der Benutzerin bzw. dem Benutzer in Kontakt steht, existiert ein gewisses Gefahrenpotential. Um die Verwendung dennoch möglichst sicher zu gestalten, wurden die Risiken genau analysiert. Daraus wurden Sicherheitsvorkehrungen abgeleitet, die wie folgt gruppiert wurden:

- **Ständige Überwachung des Anwenders bzw. der Anwenderin:** Das System überwacht stets die Position der einzelnen Körperteile. Vor der Verwendung werden die Körperdimensionen exakt vermessen und der Avatar entsprechend angepasst. Dadurch ist dem System die Position jeder Gliedmaße ständig bekannt und es kann diese Information bei der Pfadplanung berücksichtigen. Um die Sicherheit weiter zu erhöhen, sind die Körperteile des Avatars um 5 cm größer als die realen Gegenstücke. OpenRAVE überprüft bei der Erstellung eines Pfades, ob sich Hindernisse auf dem Weg befinden und versucht, diese zu umgehen. Wenn dies nicht möglich ist, wird der Pfad verworfen und es findet keine Bewegung statt. Vor der Ausführung eines Pfades wird erneut kontrolliert, ob der Weg frei ist. Ist der Pfad blockiert, so wird er nicht befahren. Da es nicht möglich ist, die Bewegungen der Anwenderinnen und der Anwender zum Zeitpunkt der Pfadplanung vorherzusagen, besteht dennoch die Gefahr einer späteren Kollision, etwa wenn der Benutzer und die Benutzerin versucht, den Arm zu berühren. Darum wird der Pfad in vielen kleinen Schritten verfolgt. Wird bei einem dieser Schritte eine Kollision erkannt, so wird die Ausführung gestoppt. Der OpenRAVE-Wrapper kann dabei angewiesen werden, eine bestimmte Zeit lang zu versuchen, den Pfad erneut zu befahren. Ist dies nicht möglich, so wird der Pfad verworfen und der Fehler protokolliert.

- **Sicheres Design:** Sollte es dennoch zu einer ungewollten Kollision kommen, so ist der Roboterarm so konzipiert, dass das Verletzungsrisiko minimiert ist (passive Sicherheit). Der gesamte Arm wiegt nur 1,9 kg, dadurch bleiben die Auswirkungen der Trägheitskräfte bei einem plötzlichen Abbremsen gering. Weiters befindet sich die Basis des Arms weit genug vom Zentrum des Virtualizers entfernt, sodass er nicht in die Nähe des Kopfes kommen kann, solange die Benutzerin und der Benutzer aufrecht steht und größer als 1,35 m ist. Niedrigster und höchster Stellwert der Motoren sind so gewählt, dass keine Gliedmaßen eingeklemmt werden können.

- **Geringe Geschwindigkeit:** Um die Auswirkungen der Trägheitskräfte weiter zu reduzieren, wird der Arm in den meisten Anwendungen mit einer deutlich niedrigeren Geschwindigkeit verwendet, als technisch möglich wäre. Auch die Beschleunigung wurde entsprechend limitiert.

- **Beschränkung der Servoleistung:** Unter Umständen könnte es zu einem Abbruch der Verbindung zwischen dem USB2Dynamixel und der Anwendung kommen. Dadurch konnten keine neuen Steuersignale übermittelt werden, um eine Kollision zu vermeiden. Als Gegenmaßnahme registriert jeder Motor-Controller bei einer

Kollision einen Anstieg der Last. Überschreitet dieser Wert die Sicherheitslimits, so geht der betroffene Servo in den Überlast-Modus und der Motor wird deaktiviert. Diese Maßnahme wirkt auch, sollte aufgrund fehlerhafter Tracking-Daten eine Kollision von der Anwendung nicht erkannt werden.

- **Notfallschalter:** Als letzte Sicherheitsmaßnahme ist bei jeder Verwendung ein menschlicher Operator anwesend. Dieser kann im Falle eines unvorhergesehenen Verhaltens des Arms einen Notfallschalter betätigen, der den Roboter sofort deaktiviert. Anschließend kann der Roboter per Hand sicher wegbewegt werden. Ein solcher Fall wäre z.B. wenn Motoren aufgrund von Überhitzung deaktiviert werden, während der Anwender und die Anwenderin mit dem Arm interagiert. Hier kann es sonst zu einer Kettenreaktion kommen, da der Wegfall eines Motors die Last auf die übrigen Motoren erhöht und diese anschließend ebenfalls ausfallen.

KAPITEL 5

Evaluierung

Dieses Kapitel behandelt die subjektive und technische Evaluierung des Systems. Einige Ergebnisse dieser Evaluierung wurden auch in einer internationalen wissenschaftlichen Publikation[VGK17] veröffentlicht. Abschnitt 5.1 enthält die Auswertung der technischen Aspekte des Systems, insbesondere des Roboterarms. Abschnitt 5.2 beschreibt die durchgeführte Benutzerstudie zur Ermittlung, wie gut das System akzeptiert wird und welche Auswirkungen der Einsatz des haptischen Feedbacks auf die Anwendererfahrung hat.

5.1 Technische Evaluierung

Im Zuge der technischen Auswertung werden die Kerneigenschaften des Systems in der Konfiguration des Testsetups mit dem ausgewählten Roboterarm beleuchtet. Diese sind:

1. Welches Interaktionsvolumen bietet das System im aktuellen Aufbau?
2. Wie genau wird das haptische Feedback dargeboten (*Absolutgenauigkeit*)?
3. Welche Schwankungen weist das System bei mehreren Wiederholungen derselben Aufgabe auf (*Wiederholgenauigkeit*)?
4. Wie lange dauert es, bis das gewünschte Feedback an der korrekten Position zur Verfügung steht (*Reaktionszeit*)?

Die Reichweite wurde bereits in Abschnitt 4.1.2 beschrieben. Ohne eine Requisite beträgt sie maximal 96,0 cm. Dabei ist der Arm vollständig gestreckt, wodurch etwaige Props nicht mehr beliebig ausgerichtet werden können. Für ein realistisches Feedback ist es aber notwendig Requisiten parallel verschieben zu können, etwa um das Vorbeigehen an einer Wand zu simulieren. Ein weiteres Beispiel ist die Darstellung eines Objekts auf das sich die Anwenderin bzw. der Anwender zubewegt. Hierbei ändert sich die

Ausrichtung der Requisite nicht, sondern nur die Position. Abbildung 36 visualisiert die Reichweite des Arms anhand eines 5,0 cm Rasters in der Y-Z Ebene. Ein Punkt bedeutet, dass die Position prinzipiell erreicht werden kann; der Strich gibt dabei die möglichen Orientierungen an. Anhand dieser Darstellung ist deutlich erkennbar, dass weniger Ausrichtungen in einer Position möglich sind, je weiter sie von der Basis entfernt ist. Der Abstand zur Basis, bei dem Objekte unter Beibehaltung der Orientierung bewegt werden können, liegt bei 60,0 cm. Da die Basis des Roboters um ±180° gedreht werden kann und alle Hindernisse in dieser Darstellung berücksichtigt sind, kann aus ihr die dreidimensionale Reichweite abgeleitet werden.

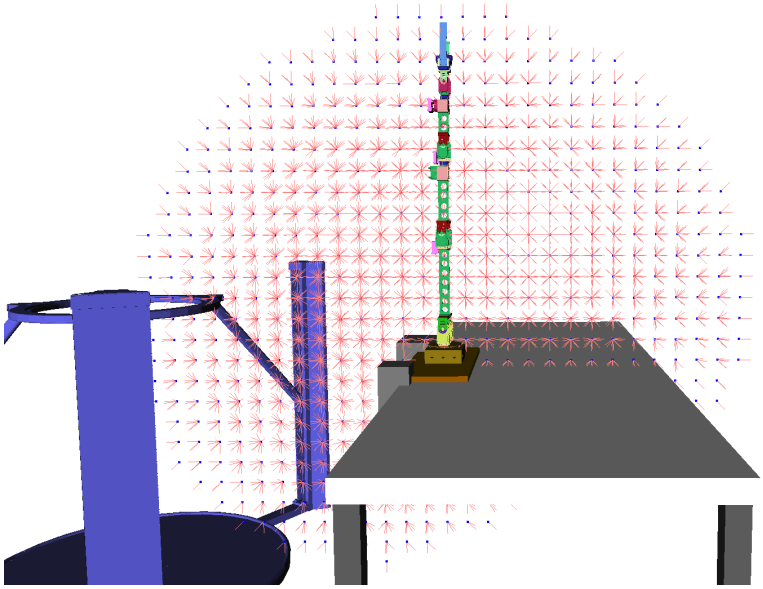


Abbildung 36: Interaktionsvolumen des Roboterarms.

Die Bewertung der Genauigkeit des Roboters basiert auf [Nic+13]. Hierbei wird im Arbeitsbereich des Roboters ein Würfel definiert, an dessen Kanten entlang jeweils fünf Punkte gleichmäßig verteilt werden. Da immer zwei Punkte pro Seite auf einen Eckpunkt fallen, enthält der Würfel insgesamt 44 Positionen. Der größtmögliche Würfel, der den Arm selbst nicht umschließt, hat eine Seitenlänge von 60 cm. Dadurch beträgt der Abstand zwischen zwei Punkten 15 cm. Im Zuge der Evaluierung fährt der Roboter jeden dieser Punkte an, wobei er zwischen zwei Punkten immer in die aufrechte Ausgangspose zurückkehrt. An jeder Position wird die tatsächliche Position im Raum sowie die aus den Daten der Sensoren errechnete Position festgehalten.

Für die Ermittlung der realen Position wurde das ARTTRACK1 Tracking System der Advanced Realtime Tracking GmbH verwendet. Dieses kommerzielle System verwendet Infrarotkameras, um retroreflektive Marker zu erfassen und ihre Positionen zu bestimmen. Die für die Auswertung verwendete Konfiguration bestand aus drei solcher Kameras, die den am Greifer des Arms befestigten Marker erkennen. Abbildung 37 zeigt eine

der Kameras sowie den Marker. Die durchschnittliche Ungenauigkeit des Systems in einem gesamten Raum liegt bei 1,0 mm. Im Zentrum des Tracking-Bereichs beträgt sie weniger als 0,5 mm.[Trü03] Um möglichst genaue Ergebnisse zu bekommen, wurde das Tracking-System direkt vor der Evaluierung so kalibriert, dass das Zentrum innerhalb des abzufahrenden Würfels lag.

(a) (b)

Abbildung 37: Tracking-System zur Messung der Genauigkeit: a) Retroreflektiver Marker an der Spitze des Roboterarms, b) eine der drei verwendeten Kameras.

Die so erhaltenen Daten wurden gemäß [Nic+13] nach drei Gesichtspunkten analysiert. Zusätzlich zu den absoluten Abweichungen wurden die Unterschiede auch für jede Raumachse berechnet. Die Differenz zwischen Zielposition und der tatsächlichen Position beschreibt die Absolutgenauigkeit (*accuracy*) des Systems. Für die x-Achse liegt der mittlere Fehler bei 15,33 mm (Standardabweichung (*SD*) 14,50), für y bei 46,20 mm (SD 24,43) und für z bei 11,30 mm (SD 23,03). Die absolute Abweichung beträgt im Durchschnitt 57,81 mm (SD 22,20).

Der zweite überprüfte Wert ist der Kodierer-Fehler. Dieser beschreibt die Differenz zwischen der Zielposition und jener Position, die anhand der Sensordaten der Motoren rückgemeldet wurde. Daraus kann auf die Genauigkeit der verwendeten Motoren und des IK-Modells geschlossen werden. Für die x-Achse liegt der mittlere Fehler bei 0,22 mm (SD 5,26), für y bei 23,40 mm (SD 12,58) und für z bei -2,93 mm (SD 11,07). Die absolute Abweichung beträgt im Durchschnitt 25,56 mm (SD 14,38).

Der letzte Faktor ist der durch die kinematische Kette entstandene Fehler. Der Fehler entsteht durch Verformungen der Bauteile des Roboters in der eingenommenen Pose, etwa wenn sich ein Träger unter der Last etwas biegt oder sich die Basisplatte hebt. Dieser beschreibt die Differenz zwischen der tatsächlichen Position und jener, die anhand der Motorstellwerte errechnet wurde. Für die x-Achse liegt der mittlere Fehler bei 15,11 mm (SD 11,35), für y bei 22,80 mm (SD 14,18) und für z bei 14,23 mm (SD 13,01). Die absolute Abweichung beträgt im Durchschnitt 36,68 mm (SD 10,22).

Die errechneten Fehler unterscheiden sich stark in den unterschiedlichen Punkten entlang des Würfels. Um dies zu verdeutlichen zeigt Abbildung 38 eine Visualisierung der absoluten Differenz in jeder Position. Hierbei ist erkennbar, dass die Abweichungen auf beiden Seiten des Arms nicht gleich sind. Dies liegt daran, dass die Positionen nicht durch dieselbe Pose eingenommen wurden, sondern immer durch jene, welche die geringste Änderung in den Stellwerten der Motoren bedeutete.

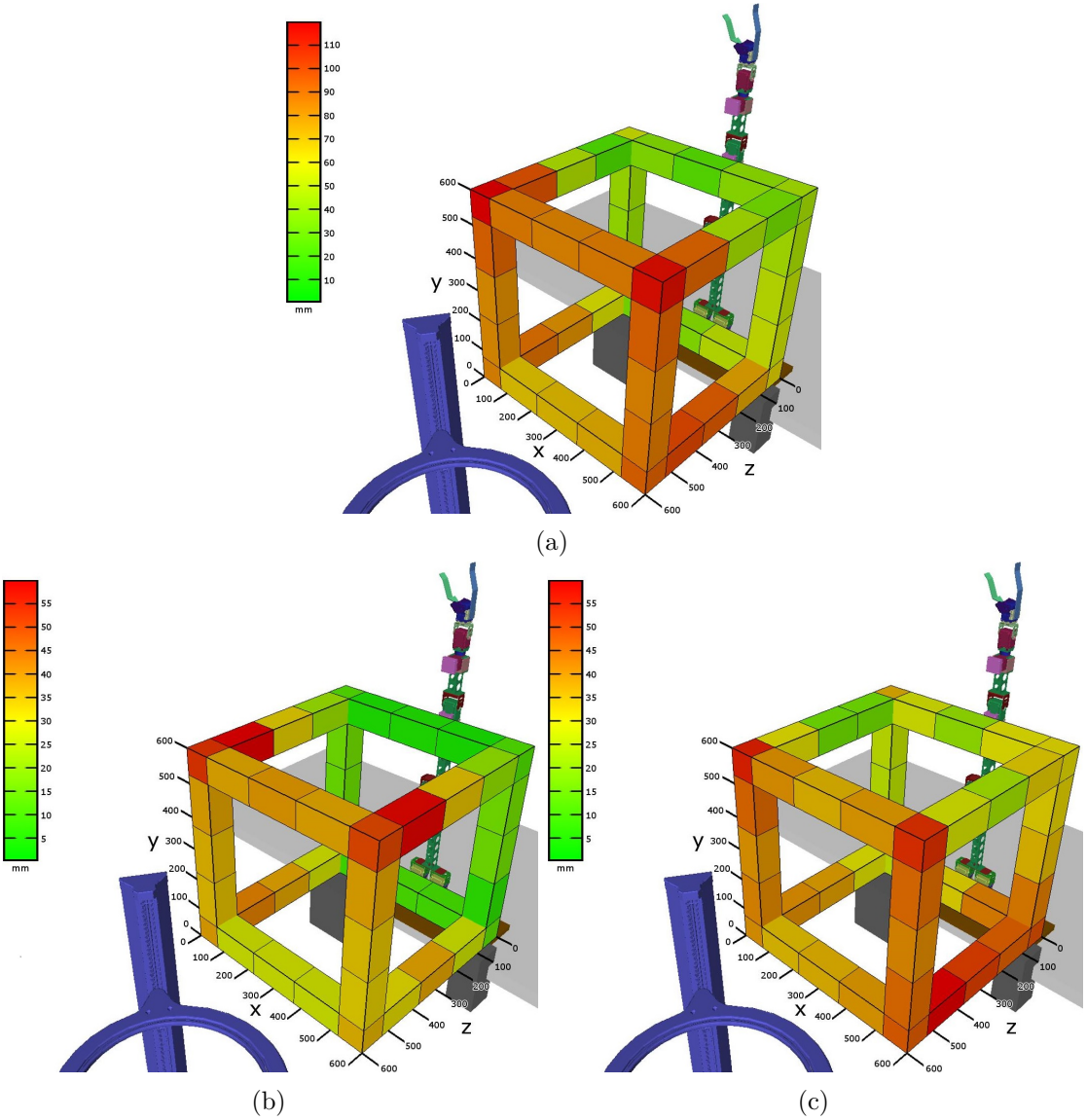
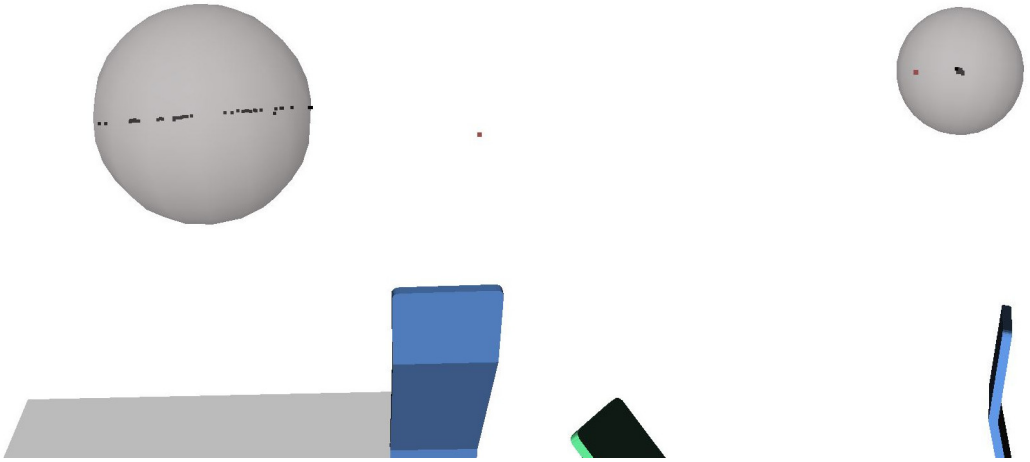


Abbildung 38: Übersicht über die absoluten Abweichungen in jeder Position des Würfels: a) Abweichung zwischen Soll- und Ist-Position (Absolutgenauigkeit), b) Abweichung zwischen Soll- und errechneter Position (Kodierer-Fehler), c) Abweichung zwischen Ist- und errechneter Position (Kinematischer Fehler).

Bei jedem der drei beschriebenen Faktoren betreffen die größten Abweichungen die y-Koordinate, was auch auf die Schwerkraft zurückzuführen ist. Dadurch wirkt sich auch das Gewicht der angebrachten Last sehr stark auf die Genauigkeit aus. Die Wiederholgenauigkeit (*repeatability*) ist davon nicht betroffen und wird daher auch als Maß für die Qualität eines Roboterarms herangezogen.[Nic+13] Für die Berechnung wird ein und derselbe Punkt mehrmals angefahren. Anschließend wird die kleinstmögliche Kugel errechnet, die alle tatsächlich eingenommenen Positionen umschließt. Der Radius dieser Kugel beschreibt die Wiederholgenauigkeit, wobei ein größerer Radius einen schlechteren Wiederholgenauigkeitswert bedeutet. Im Laufe der technischen Evaluierung wurde nach jeder Position im Würfel die aufrechte Ausgangsposition eingenommen. Die 44 gemessenen tatsächlichen Positionen können von einer Kugel mit einem 16,26 mm Radius umfasst werden. Abbildung 39 zeigt diese Kugel, die gemessenen Werte (schwarz) und die Zielposition (rot). Die Positionen liegen, bedingt durch den eingesetzten Marker des Tracking-Systems, oberhalb des Greifers. Die Position des Endeffektors wurde im OpenRAVE-Modell etwas nach oben versetzt, um eine Kollision des Markers mit der Tischplatte zu vermeiden. Es ist erkennbar, dass die gemessenen Punkte fast entlang einer Linie entlang der z-Achse liegen.



(a) Seitenansicht (x-Achse) auf das System. (b) Frontansicht (z-Achse) auf das System.

Abbildung 39: Unterschiedliche Positionen für die Ausgangspose.

Als letztes Bewertungskriterium wurde die Reaktionszeit des Systems analysiert. Diese ist die Zeitspanne zwischen dem Befehl, eine bestimmte Position einzunehmen, und dem tatsächlichen Erreichen selbiger. Sie setzt sich aus der Dauer für die Berechnung der Motorstellwerte und des Pfades sowie der Fahrzeit zusammen. Dieses Kriterium ist besonders interessant, da es beschreibt, wie lange es dauert, bis ein gewünschtes haptisches Feedback verfügbar ist. Hierzu wurden die Punkte aus der vorherigen Auswertung nacheinander angefahren, ohne in die Ausgangsposition zurückzukehren. Die zurückgelegten Wegstrecken können dabei in drei Kategorien eingeteilt werden: Kurz (150 mm), Mittel (600 mm - 671 mm) und Lang (848 mm - 1039 mm). Um vergleichbare Werte

zu erhalten, wurden für die Bewegung des Arms dieselben Limitierungen für Geschwindigkeit und Beschleunigung festgelegt, die in tatsächlichen Anwendungen gesetzt sind. Tabelle 2 zeigt die durchschnittlichen Zeitspannen für die Teilaufgaben sowie für die Reaktionszeit. Bei Pfaden mit einer Länge bis zu 150 mm beträgt die Planungsdauer im Mittel 0,83 s (SD 0,69) und die Bewegung benötigt 2,75 s (SD 2,05). Kurze Pfade sind jene Strecken zwischen zwei benachbarten Punkten entlang einer Kante des befahrenen Würfels. Die durchschnittliche Reaktionszeit liegt bei 3,58 s (SD 2,72). Die Planung benötigt bei Pfadlängen zwischen 600 mm und 671 mm mit durchschnittlich 2,43 s (SD 2,96) am längsten, da diese oft den Wechsel von einer Kante des Würfels zu einer benachbarten beschreiben. Dabei müssen deutlich mehr Gelenke auf einem kürzeren Weg verändert werden, ohne eine Kollision zu verursachen. Die mittlere Bewegungsdauer ist entsprechend der längeren Wegstrecke etwas höher und liegt bei 3,73 s (SD 1,12). Daraus ergibt sich eine durchschnittliche Reaktionszeit von 6,17 s (SD 2,89). Zu den langen Pfaden (848 mm - 1039 mm) zählen die Bewegungen von einer Kante des Würfels zu einer nicht benachbarten. Sie sind etwas simpler als die der mittleren Strecken und benötigen daher im Schnitt 1,6 s (SD 0,42). Die benötigte Zeit für die Bewegung ist aufgrund der Strecke am höchsten und beträgt durchschnittlich 4,10 s (SD 1,75). Aufgrund der kürzeren mittleren Planungszeit liegt die Reaktionszeit im Mittel bei 5,70 s (SD 2,12) und somit zwischen der für kurze und jener für mittlere Strecken. Die größte Auswirkung auf die Reaktionszeit hat die Komplexität des Pfades, welche durch die Einschränkungen der Gelenke sowie etwaige Hindernissen bestimmt wird. Dies zeigt sich in der minimal und maximal benötigten Zeit für jede Kategorie. Bei kurzen Strecken beträgt der kleinste Wert beispielsweise 0,74 s und der größte 12,43 s. Die Wertebereiche für mittlere und lange Wege fallen ähnlich aus (Mittel: 3,17 s - 12,42 s, Lang: 2,85 s - 8,59 s). Diese Extremwerte stellen allerdings eher Ausnahmen als den Normalfall dar.

Pfadlänge	**Planung**	**Bewegung**	**Reaktionszeit**
Kurz (150 mm)	0,83 s (SD 0,69)	2,75 s (SD 2,05)	3,58 s (SD 2,72)
Mittel (600 mm - 671 mm)	2,43 s (SD 2,96)	3,73 s (SD 1,12)	6,17 s (SD 2,89)
Lang (848 mm - 1039 mm)	1,6 s (SD 0,42)	4,10 s (SD 1,75)	5,70 s (SD 2,12)

Tabelle 2: Durchschnittliche Dauer der drei Pfadlängen.

5.2 Benutzerstudie

Die durchgeführte Benutzerstudie hatte zum Ziel, die subjektive Erfahrung bei der Verwendung des VR-Systems festzustellen. Dazu zählt, wie gut das System von den Anwendern und den Anwenderinnen angenommen wird, sowie die Auswirkungen des haptischen Feedbacks auf die VR-Erfahrung. An der Studie nahmen 12 Frauen und 22 Männer im Alter von 14 - 58 Jahren teil. Im Vorfeld gaben die Teilnehmerinnen und die Teilnehmer anhand einer Skala von 1 (gar keine) - 5 (sehr viel) an, wie oft sie einen Computer verwenden und ob sie bereits Erfahrung mit VR gemacht haben. Hierbei zeigte sich, dass alle Befragten eine Computernutzung von drei oder höher anführten, der

Mittelwert lag bei 4,5. Hinsichtlich der VR-Erfahrung gaben nur 14 Personen an, jemals Umgang mit VR-Technologien gehabt zu haben und nur zwei Personen nannten einen Wert höher als drei. Der Mittelwert bei dieser Frage lag bei 1,7.

Im Zuge der Benutzerstudie mussten die Tester und die Testerinnen zwei Aufgaben in eigens dafür erstellten VR-Szenarien erfüllen. Eine der beiden Aufgaben wurde mit haptischem Feedback durchgeführt, die andere ohne. Die Testpersonen wurden durch Zufall einer von zwei Gruppen zugeordnet, die sich dadurch unterschieden, welches Szenario mit haptischem Feedback durchgeführt wurde.

In den folgenden Unterabschnitten werden zuerst die Aufgaben beschrieben. Anschließend werden Methodik und Ablauf der Studie sowie der verwendete Fragebogen erläutert. Im abschließenden Teil werden die Ergebnisse aufbereitet und interpretiert.

5.2.1 Aufgaben

Für die beiden Aufgaben wurde jeweils eine eigene VR-Anwendung erstellt, die auf dem System dieser Arbeit basiert. Hierbei kann beim Start festgelegt werden, ob das haptische Feedback durch den Roboterarm aktiviert wird oder nicht. Für die Realisierung des Feedbacks wird der Arm mit einem zur Aufgabe passenden Prop ausgestattet. Dieser Abschnitt beschreibt den Aufbau der beiden Szenarien, die durchzuführende Aufgabe, sowie die Art des angebotenen Feedbacks.

Bei der ersten Aufgabe befindet sich die Testerin bzw. der Tester in einem virtuellen Garten. Darin befindet sich eine 2,65 m hohe und 5 m breite Holzwand. Die Testperson wird aufgefordert, sich zu einem Punkt auf der Wand zu bewegen, an dem weiße Partikel heraussprudeln. Abbildung 40 zeigt die Wand mit dem ersten Ziel. Ist der Abstand zu dem Zielpunkt gering genug, intensiviert sich das Sprudeln, um zu signalisieren, dass der Punkt in Reichweite ist. Berührt die Testperson anschließend den Punkt auf der Wand, so werden die weißen Partikel grün. Diese Position muss zwei Sekunden gehalten werden, dann verschwindet der Punkt und ein weiterer erscheint an einer anderen Stelle. Insgesamt sind 7 Punkte auf verschiedenen Höhen entlang der Wand verteilt, die alle berührt werden müssen. Nachdem der letzte Punkt verschwunden ist, erscheint ein Schriftzug, der das Ende der Aufgabe anzeigt. Abschließend wird dem Tester bzw. der Testerin angeboten, sich im Garten frei zu bewegen, um sich mit der Fortbewegung im System ein wenig vertrauter zu machen.

Als Requisite, die die Holzwand simuliert, ist eine 30 cm hohe und 30 cm breite Holzplatte am Roboterarm befestigt. Um die Belastung gering zu halten, wird sie nur dann in Position gebracht, wenn die Testperson in Reichweite ist. Ist die Entfernung gering genug wird der Arm so positioniert, dass sich der Mittelpunkt der Platte mit mit der Position des Zielpunkts deckt. Anschließend signalisiert ein intensiveres Sprudeln, dass die Wand berührt werden darf. Aus Sicherheitsgründen wird die Platte nicht bewegt, wenn die Platte berührt wird oder der Weg versperrt ist. Die Pfadberechnungen werden ebenso in der nicht haptischen Variante ausgeführt, allerdings wird die Roboterbewegung nur in

OpenRAVE durchgeführt. Dadurch wird sichergestellt, dass sich das System in beiden Ausprägungen gleich verhält und eine ähnliche Reaktionszeit an den Tag legt.

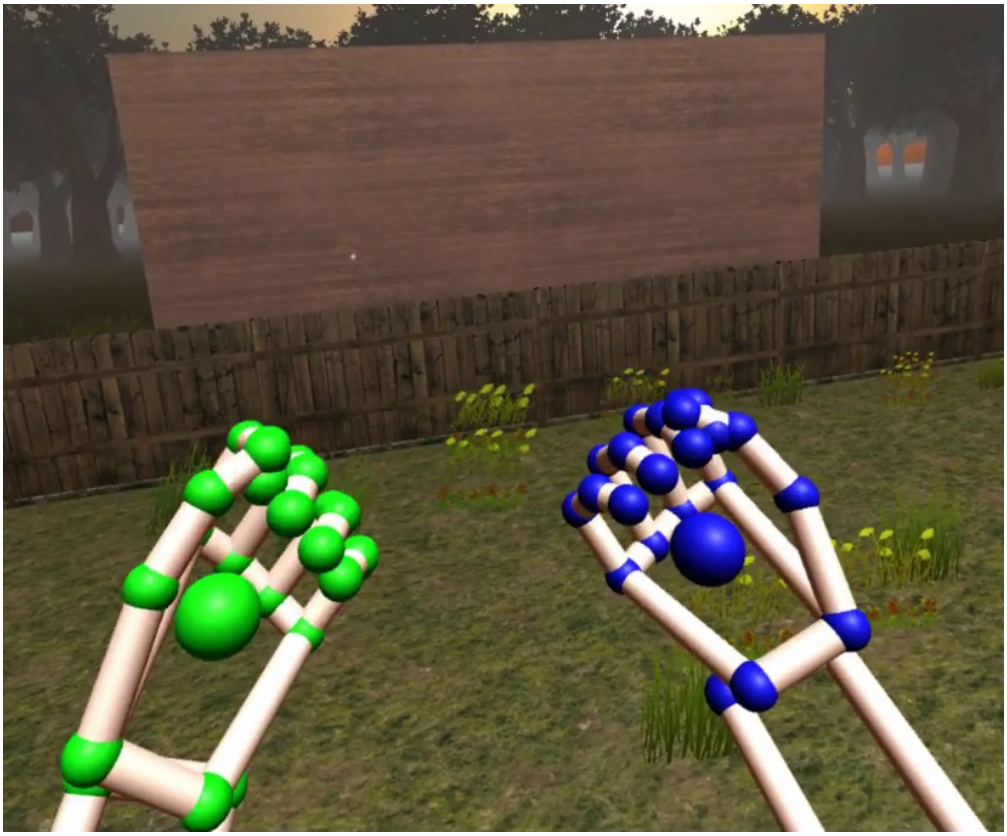


Abbildung 40: Wand mit Zielpunkt und virtuellen Händen.

Die zweite Aufgabe findet in einem virtuellen Häuserblock statt und wurde nicht vom Autor implementiert. Die Testerin bzw. der Tester startet auf dem Gehweg und soll ihn 32 m entlang bis zu einer Kreuzung gehen. Auf der anderen Seite der Kreuzung wartet ein winkender Avatar, der sich in Bewegung setzt, sobald die Testperson die Straße überquert hat. Die Testperson soll dem Avatar in gleichbleibendem Tempo für 76 m folgen. Auf dieser Strecke kommen dem Tester bzw. der Testerin 10 Passanten entgegen. 4 Passanten weichen immer aus, die restlichen sind auf einen Kollisionskurs programmiert. Abbildung 41 zeigt die Situation kurz vor einer solchen Kollision. Die weiteren Passanten und der zu verfolgende Avatar sind ebenfalls zu sehen. Das haptische Feedback, welches die Schulter der Passanten repräsentiert, erzeugt ein weicher, nachgiebiger Boxhandschuh, der an der Spitze des Roboterarms montiert ist. Die Aufgabe endet an einer weiteren Kreuzung, sobald der Avatar stehen bleibt und die Testperson ihn einholt. Insgesamt legt die Testerin bzw. der Tester 176 m zurück.

Abbildung 41: Bevorstehende Kollision mit einer entgegenkommenden Person.

Analog zu der Aufgabe im Garten läuft auch im Ablauf ohne haptisches Feedback die Simulation in OpenRAVE im Hintergrund. Da aber bei diesem Szenario nicht darauf gewartet werden muss, dass sich der Arm in einer korrekten Position befindet, ist dies für die Testperson nicht merkbar.

5.2.2 Methodik und Ablauf

Die Benutzerstudie bestand aus jeweils einer Testsitzung pro Testperson, die ungefähr eine Stunde dauerte. Bei jeder Sitzung waren zwei Beobachter anwesend, die durch den Ablauf führten.

Den Beginn stellte eine mündliche Einführung in die Studie dar. Die Testperson erfuhr, dass im Zuge des Experiments zwei Aufgaben in VR zu erfüllen sind und dabei ein Roboterarm haptisches Feedback liefern kann. Der Beobachter erörterte den Ablauf und klärte über die verwendeten Komponenten, die Sicherheitsvorkehrungen und das bestehende Restrisiko auf. Anschließend wurde der Testperson der Fragebogen ausgehändigt, mit

dem die Ergebnisse erhoben wurden. Hierbei erfolgte auch die zufällige Zuordnung zu einer der beiden Gruppen:

- Garten mit haptischem Feedback, Stadt ohne. (Gruppe G)
- Garten ohne haptischem Feedback, Stadt mit. (Gruppe C)

Zu Gruppe G wurden 18 Personen zugeordnet, zu Gruppe C 16. Die zugeordnete Gruppe sowie die Startzeit wurden vom Testleiter auf dem Fragebogen vermerkt. Die erste Seite des Fragebogens enthielt eine schriftliche Beschreibung des Experiments sowie eine Einverständniserklärung, dass die Risiken verstanden wurden und Foto- sowie Videomaterial aus der Sitzung anonymisiert verwendet und veröffentlicht werden dürfen.

Anschließend wurde die Testperson vermessen und die Daten in den Fragebogen sowie auf den Avatar der Testszenarien übertragen. Der Proband bzw. die Probandin beantwortete danach die einleitenden Fragen zum Experiment sowie einen *Fragebogen zur Erhebung von Simulatorkrankheit* (*SSQ*)[Ken+93]. Daraufhin wurden der Perception Neuron Anzug sowie das DK2 angelegt und kalibriert.

Nach der Kalibrierung half ein Testleiter der Person in den Virtualizer, während der andere die Szene vorbereitete. Die Testperson bekam eine kurze Einschulung über die Fortbewegung innerhalb der virtuellen Welten und über die Aufgabenstellung. Bis zu diesem Zeitpunkt war der Roboterarm immer in aufrechter Pose und ohne eine Requisite zu sehen. Erst nachdem die Testperson im Virtualizer das HMD und die Kopfhörer aufgesetzt hatte, wurde, falls sie der Gruppe G zugeordnet war, die Holzplatte montiert. Vor dem Start der ersten VR-Anwendung richtete ein Testleiter die Testperson noch anhand des Lotes aus. Anschließend wurde das Szenario gestartet. Nach Erfüllung der ersten Aufgabe beantwortete die Probandin bzw. der Proband erneut den SSQ, ohne den Virtualizer zu verlassen. Danach wurde mit der zweiten Aufgabe im zweiten Szenario fortgefahren, wobei das etwaige Prop (Gruppe C) wieder erst nach Aufsetzen des HMD montiert wurde.

Abschließend wurde die Testperson ein letztes Mal gebeten, den SSQ auszufüllen, sowie die restlichen Fragen des Fragebogens zu beantworten. Diese Fragen bezogen sich auf die beiden Aufgaben und auf die generelle Zufriedenheit mit dem System. Im Hinblick auf die Aufgaben wurden auch Fragen gestellt, die, auf Slater et al.[SUS95] basierend, die Fortbewegung und das Präsenzgefühl im System bewerten. Die Testpersonen befanden sich durchschnittlich jeweils fünf Minuten in den virtuellen Umgebungen. Das Geschehen während beider Aufgaben wurde sowohl in der realen als auch in der virtuellen Welt aufgezeichnet. Die Tests verliefen großteils störungsfrei. Vereinzelt auftretende Störungen stellten zu keiner Zeit eine Gefährdung der Testperson dar und konnten zumeist durch die Testleiter kompensiert werden.

5.2.3 Ergebnisse

Der verwendete Fragebogen bestand aus zwei Teilen. Zum einen beinhaltete er 55 Auswahlfragen und 7 Erörterungsfelder zum Testvorgang und den gestellten Aufgaben. Zum anderen enthielt er den SSQ, der 16 Fragen zum Wohlbefinden der Testpersonen umfasste. Bei jeder Frage wurde das Vorhandensein eines Symptoms der Simulatorkrankheit auf einer vierstufigen Skala (kein, leicht, moderat, stark) festgehalten.[Ken+93]

Dieser Abschnitt befasst sich mit der Auswertung der Antworten, um zu eruieren, wie gut das System angenommen wird und welche Auswirkungen das haptische Feedback hat. Bei zwei Testpersonen, jeweils einer aus den beiden Testgruppen, war der Fragebogen unvollständig, bei einer weiteren der Gruppe C wurde der Roboterarm nicht aktiviert. Die Angaben dieser Testpersonen wurden daher in den entsprechenden Auswertungen ausgeschlossen. Im folgenden Kapitel wird auf diese Fälle explizit hingewiesen. Dabei bezeichnen G1 und C1 die Testpersonen mit unvollständigem Fragebogen, C2 den Fall ohne aktivierten Roboterarm.

Wie gut die Erfahrung ist, die die Versuchspersonen mit dem System gemacht haben, ist ein Indikator dafür, wie gut die Komponenten des Systems zusammenspielen. Diese subjektive Bewertung wird nicht wesentlich vom haptischen Feedback beeinflusst und wird daher getrennt analysiert. Dabei basiert die Auswertung auf mehreren Faktoren, die im Fragebogen erfasst wurden. Ein interessanter Aspekt ist, ob das System als gefährlich wahrgenommen wird. Hierzu wurde jede Testperson vor und nach ihrer Sitzung gefragt, ob sie Vertrauen in das System hat und ob ihr die Vorstellung, mit dem Roboterarm zu interagieren, Unbehagen bereitet. Beide Fragen konnten anhand einer Skala von 1 - 5 beantwortet werden, wobei 1 kein Vertrauen bzw. kein Unbehagen und 5 sehr großes Vertrauen bzw. großes Unbehagen bedeutet. Die vor der Verwendung abgegebenen Antworten zeigen ein hohes Vertrauen (Mittelwert (M): 4,53 SD: 0,72) und geringes Unbehagen (M: 1,72 SD: 1,14). Die Ergebnisse nach Vollendung der Aufgaben weisen nur geringfügige Veränderungen auf. Die mittlere Änderung des Vertrauens ist 0,00 (SD: 0,98), das Unbehagen wurde im Mittel um 0,36 (SD: 1,12) geringer. Abbildung 42 visualisiert die abgegebenen Antworten vor und nach der Testsitzung.

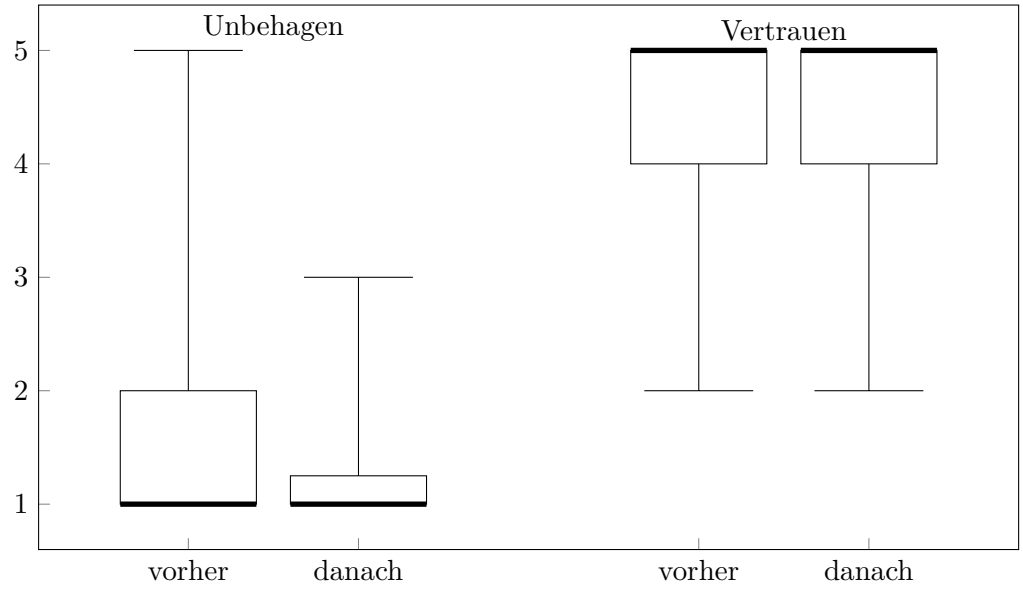


Abbildung 42: Vertrauen der Testpersonen in das System und ihr Unbehagen, mit dem Roboterarm zu interagieren, vor und nach dem Experiment.

Ein weiterer Punkt ist, wie gut die Fortbewegung im System funktioniert. Hierbei wurde, angelehnt an Slater et al.[SUS95], bei jeder Aufgabe gefragt, wie schwer bzw. leicht, unnatürlich bzw. natürlich und kompliziert bzw. unkompliziert die Fortbewegung war. Jede dieser Fragen bot eine Antwortskala von 1 - 7, bei der 1 die negativste Ausprägung war. Der Durchschnitt der Antworten auf diese insgesamt sechs Fragen gibt an, wie die Testperson die Fortbewegung empfunden hat. Im Mittel wurde die Fortbewegung mit 4,33 (SD: 1,17) eher positiv bewertet. In den entsprechenden Erörterungsfeldern gaben 8 Testpersonen als Hauptgrund für die mittelmäßige Bewertung an, dass das Gehen als unnatürlich, unangenehm und anstrengend empfunden wurde.

Zusätzlich zum Eindruck der Fortbewegung, gibt Slater et al.[SUS95] auch das Anwesenheitsgefühl in der virtuellen Welt als Indiz für eine gute Akzeptanz an. Dieses wurde im Fragebogen durch zwei Gruppen von Fragen erfasst: Einerseits, ob sich die Testperson präsent fühlt, und andererseits, ob die haptischen Elemente der Aufgaben als tatsächlich existent wahrgenommen wurden. In dieser Studie zeigte sich, dass beide Gruppen zwar zum Teil auch vom haptischen Feedback abhängen, allerdings war für das Präsenzgefühl die Identifikation mit dem Avatar eher ausschlaggebend. Da hierbei auch das Körpertracking von 7 Testpersonen als positiver Faktor erwähnt wurde und kein großer Unterschied zwischen den Werten aus den Aufgaben mit haptischem Feedback (M: 5,29 SD: 1,11, Testperson C2 ausgenommen) und jenen ohne (M: 5,17 SD: 0,96) festzustellen war, wird das Anwesenheitsgefühl der Testperson für die Bewertung des allgemeinen Eindrucks herangezogen. Der geringe Unterschied wird auch durch die Visualisierung in Abbildung 43 deutlich. G1 wurde bei der Auswertung mit haptischem

Feedback ausgenommen. Abgesehen von den Extremwerten sind die Antworten ähnlich ausgefallen.

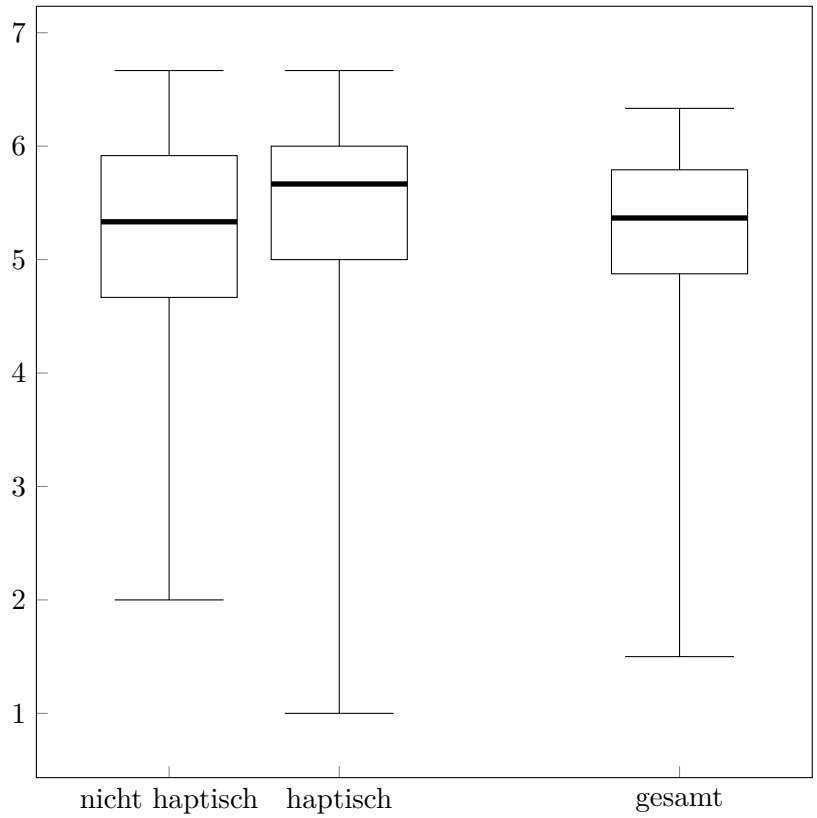


Abbildung 43: Präsenzgefühl der Testpersonen.

Für die Analyse des Anwesenheitsgefühls wurden pro Aufgabe drei Fragen gestellt. Diese basieren ebenfalls auf Slater et al.[SUS95] und stellen fest, inwieweit die Testperson sich tatsächlich anwesend gefühlt hat und ob sie den virtuellen Ort als ihre Realität angenommen hat. Weiters wurde gefragt, ob die Erlebnisse als betrachtete Bilder oder besuchte Orte empfunden wurden. Die Auswertung der Antworten nach Slater et al.[SUS95] liefert im Mittel ein Präsenzgefühl von 5,22 (SD: 0,88). Dieser relativ hohe Wert ist laut den Ausführungen auch auf die gut funktionierende Kombination von Körpertracking und HMD zurückzuführen. 6 Testpersonen erwähnten das Vorhandensein von Händen und die gute Übereinstimmung der Körperteile als positiven Eindruck. Den abschließenden Teil stellt die Auswertung des SSQ dar. Das Auftreten von Simulatorkrankheit beeinflusst das Wohlbefinden der Testpersonen enorm und wirkt sich daher auch auf die Erfahrung im System aus. Die Antworten werden gemäß Kennedy et al.[Ken+93] Gruppen zugeordnet. Die Werte der Antworten werden pro Gruppe aufsummiert und anschließend gewichtet. Die finale Bewertung entspricht der gewichteten Summe dieser drei Werte. Der Wert nach der ersten Aufgabe ist durchschnittlich um -35,94 (SD: 148,99) niedriger als der zu Beginn

der Sitzung. Nach der zweiten Aufgabe ist der Wert im Mittel 35,62 (SD: 175,71) höher als der zuvor. In der Summe ist die Bewertung stabil geblieben, die Schwankungen lassen sich aufgrund der textuellen Ausführungen auf Nervosität vor dem Test und die körperliche Anstrengung durch die ungewohnte Bewegung in der zweiten Aufgabe erklären. Einige Testpersonen gaben an, dass es ihnen nach dem Test besser ging als davor. Eine Person meldete ein Unwohlsein, was sich auch in ihrem deutlich höheren Simulatorkrankheitswert niederschlug.

Die Auswirkungen des haptischen Feedbacks beruhen auf zwei Gruppen von Fragen. Einerseits wurden die Testpersonen über ihre subjektiven Eindrücke in Bezug auf die Interaktionen mit den Zielobjekten befragt. Andererseits mussten sie Schätzfragen zu den Zielobjekten der jeweiligen Aufgaben ausfüllen. Betreffend der subjektiven Eindrücke wurde die Vermutung aufgestellt, dass die Zielobjekte realer wahrgenommen werden, wenn haptisches Feedback vorhanden ist.

Als erster Punkt wurde erneut das Präsenzgefühl nach [SUS95] herangezogen. Diesmal wurde jedoch gefragt, ob der Proband bzw. die Probandin die Zielobjekte als tatsächlich präsent wahrgenommen hat. Dazu wurden bei jeder Aufgabe vier Fragen gestellt, die sich an der Methodik von [SUS95] orientieren. Abbildung 44 veranschaulicht die statistische Auswertung der Antworten. G1, C1 und C2 wurden dabei ausgeschlossen. Bei der Gartenaufgabe liegt der Durchschnitt in der Version ohne Roboterarm bei 4,73 (SD: 1,31) und mit Arm bei 4,91 (SD: 1,03, G1 ausgenommen). Der Mittelwert für die Stadt beträgt ohne Feedback 4,39 (SD: 0,92) und mit Arm 4,64 (SD: 1,35, C1 und C2 ausgenommen). Damit befinden sich die Werte im Mittel in der oberen Hälfte der Skala, sind aber etwas unter denen des persönlichen Anwesenheitsgefühls. Weiters ist nur eine geringe Steigerung durch den Einsatz des Roboterarms erkennbar. Dies kann unter anderem darauf beruhen, dass der Unterschied zwischen den Fragen zum eigenen Präsenzgefühl und dem Eindruck der Existenz der Objekte nicht klar ersichtlich war. Weiters gaben einige Testpersonen an, dass die Reaktionen des Roboterarms zu langsam waren, was die Verbindung zwischen virtueller Welt und der Requisite beeinträchtigte.

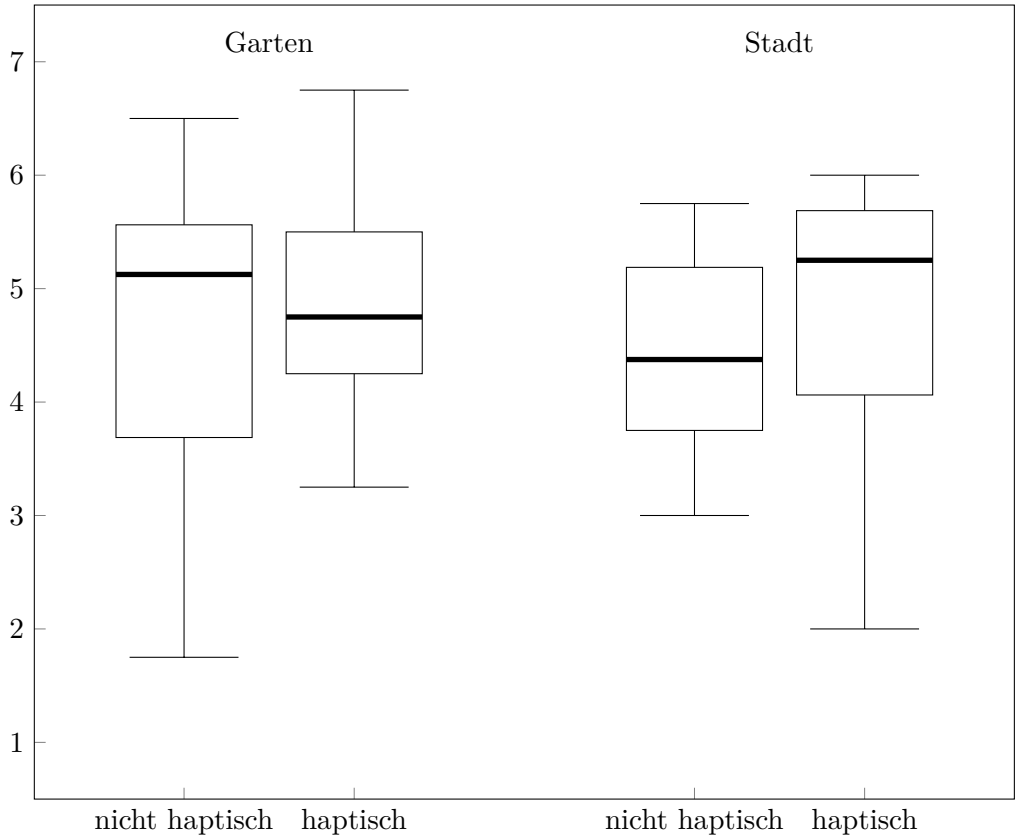


Abbildung 44: Wahrnehmung, wie stark die interaktiven Objekte der Szene als tatsächlich anwesend registriert wurden.

Die Möglichkeit von Unklarheiten bei den Präsenzfragen wurde in der Konzeption des Fragebogens berücksichtigt, indem noch weitere Detailfragen gestellt wurden. Um festzustellen, wie überzeugend die Simulation der Objekte ist, gaben die Testpersonen sowohl für die Wand als auch für die Passanten an, wie realistisch und wie glaubhaft die Objekte waren. Die gemittelten Ergebnisse dieser Fragestellung liegen bei den Szenen, die mit passiv-haptischem Feedback erlebt wurden, etwas über denen ohne. Für die Wand im Garten liegt der Wert ohne Feedback im Mittel bei 3,84 (SD: 2,04) und bei 4,78 (SD: 1,50), wenn der Roboterarm eingesetzt wurde. Bei den Passanten in der Stadt beträgt der Durchschnitt 3,78 (SD: 1,31) ohne Roboter und 4,22 (SD: 1,52, C2 ausgenommen) mit Arm. Auch in Abbildung 45 ist erkennbar, dass die Objekte in den Szenen mit haptischem Feedback überzeugender waren. In dieser Darstellung wurden G1 und C2 ausgeschlossen.

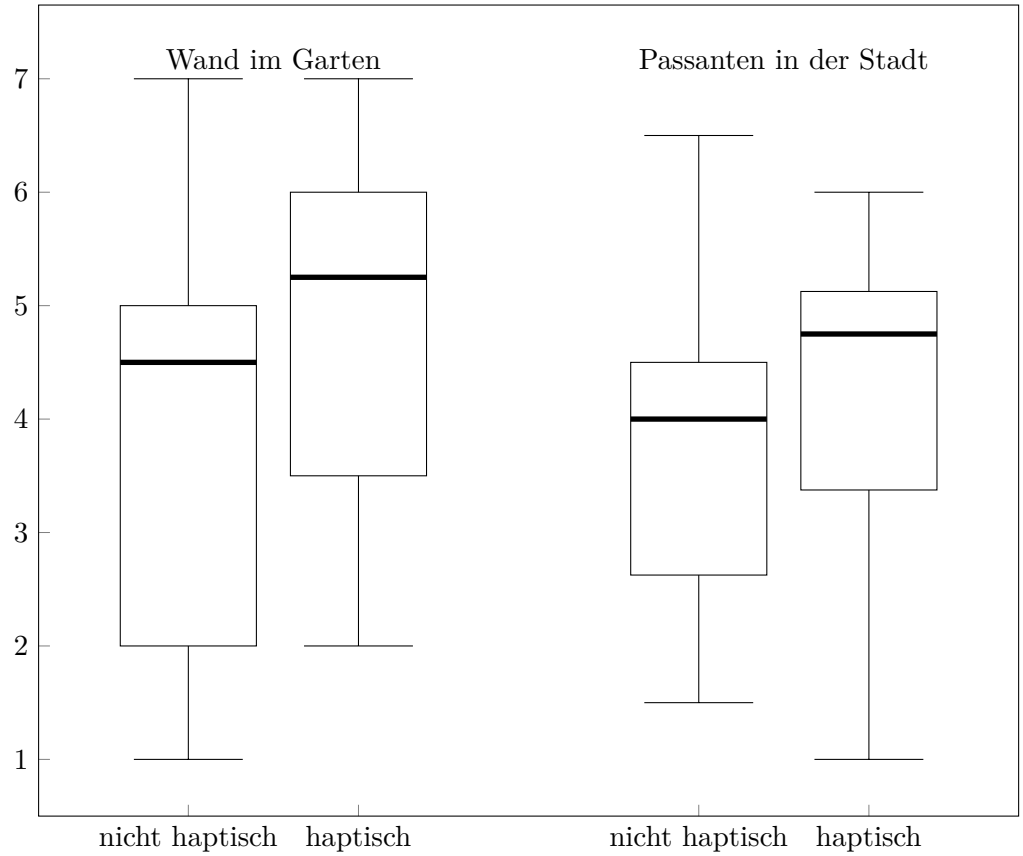


Abbildung 45: Auswertung, wie überzeugend die Objekte in der jeweiligen Aufgabe waren.

Als letztes Kriterium wurde gemessen, wie realistisch die Interaktion mit den Objekten war. Hierzu wurden die Testpersonen gefragt, wie natürlich die Kollisionen mit den Passanten und die Berührung der Wand empfunden wurden. Außerdem mussten sie angeben, wie ähnlich diese Interaktionen realen Berührungen und Kollisionen waren. Für diesen Aspekt wurde die größte Auswirkung durch den Roboterarm erwartet. Diese Hypothese wird durch die Ergebnisse bestärkt. Der Wert lag für die Berührung der Wand ohne Requisite durchschnittlich bei 3,09 (SD: 1,32) und mit der Holzplatte bei 4,03 (SD: 1,17). Auch in der Stadt erzeugte der Boxhandschuh einen Mittelwert von 3,63 (SD: 1,56, C2 ausgenommen) und die Interaktion war damit merklich realistischer als die rein visuelle Version mit 2,58 (SD: 1,39). Abbildung 46 stellt die Ergebnisse ohne C2 grafisch dar.

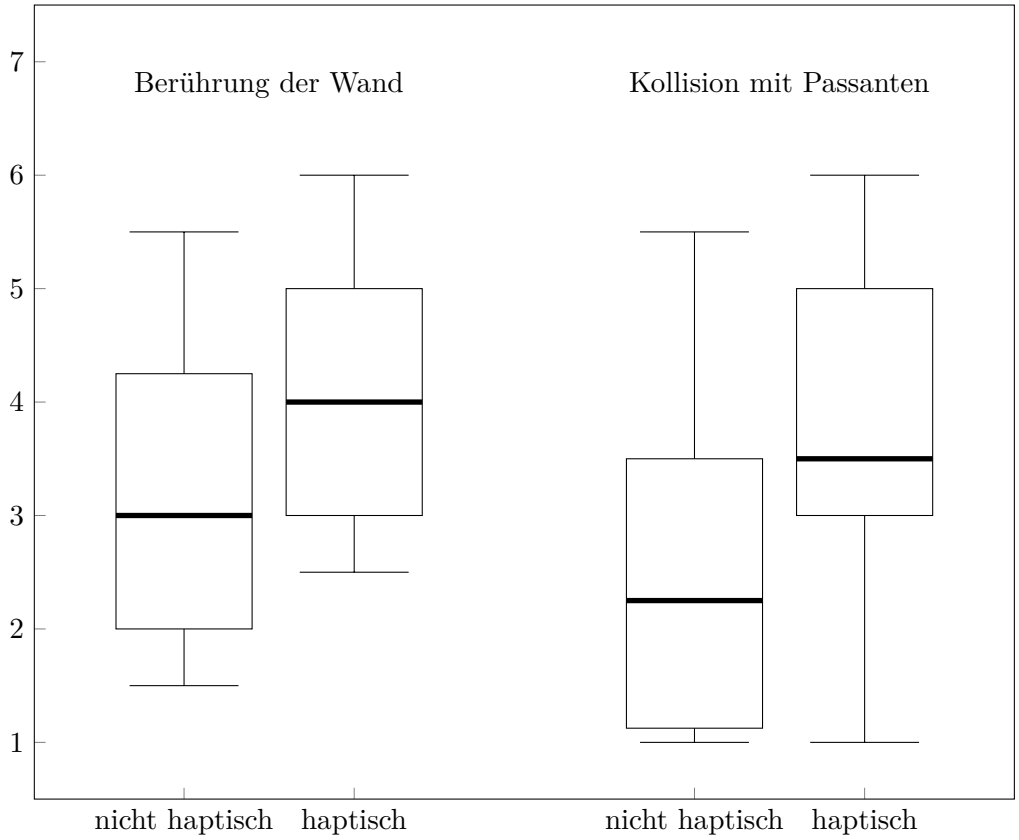


Abbildung 46: Auswertung darüber, wie realistisch die Interaktionen mit den Objekten in der jeweiligen Aufgabe waren.

Die Analyse dieser Punkte bestätigt teilweise die Hypothese, dass sich das haptische Feedback positiv auf die Qualität der Simulation auswirkt.

Die zweite Vermutung dieses Experiments war, dass die Verwendung des Feedbacks das Gespür der Testerinnen bzw. der Tester für die dargestellten Objekte verbessert. Dazu mussten die Testpersonen am Ende ihrer Testeinheiten folgende Schätzfragen beantworten:

1. Wie viele Ziele waren auf der Wand?
2. Wie breit war die Wand?
3. Wie hoch war die Wand?
4. Auf welcher Höhe befand sich das letzte Ziel?
5. Wie viele Personen kamen Ihnen in der Stadt entgegen?

6. Mit wie vielen Passanten kam es zu einer Kollision?

Die abgegebenen Antworten wurden mit den tatsächlichen Daten verglichen. Die Differenzen zwischen den geschätzten und den realen Werten sind bei der Wand in den beiden Varianten sehr gering, wodurch eine Verbesserung durch die haptische Variante nur schwer erkennbar ist. Die mittleren Abweichungen für die Fragen 1 - 4 betragen ohne haptisches Feedback 1,27 (SD: 1,83), -0,78 (SD: 3,3), -0,21 (SD: 0,61) bzw. 0,2 (SD: 0,36) und mit Feedback 0,94 (SD: 1,56), -1,16 (SD: 3,29), -0,03 (SD: 1,02) bzw. -0,08 (SD: 0,63). Auch die Anzahl der entgegenkommenden Personen wurde unter beiden Bedingungen ähnlich gut geschätzt(M: -0,83, SD: 6,3 ohne Feedback bzw. M: -1,79, SD: 4,46 mit Feedback, C1 und C2 ausgenommen). Einzig bei der Frage nach den Kollisionen unterscheidet sich die durchschnittliche Abweichung bei haptischem Feedback mit -0,14 (SD: 2,14, C1 und C2 ausgenommen) etwas mehr von der ohne (M: 3,56, SD: 1,62). Dies ist jedoch nicht ausreichend, um die zweite Vermutung zu bestätigen, dass haptisches Feedback positive Auswirkungen auf die Qualität der Simulation hat.

KAPITEL 6

Diskussion

Dieses Kapitel beschäftigt sich mit der Interpretation der Ergebnisse aus Kapitel 5. Abschnitt 6.1 behandelt die erkannten Probleme und beschreibt mögliche Lösungsansätze. Abschnitt 6.2 gibt abschließend einen Einblick in eventuelle kommerzielle Anwendungsszenarien für das System.

Die durchgeführten Auswertungen haben gezeigt, dass ein Roboterarm in einem VR-System passiv haptisches Feedback erzeugen kann. Generell lassen die Ergebnisse den Schluss zu, dass das gesamte System gut akzeptiert wird, die positiven Auswirkungen des Feedbacks sind allerdings nur Tendenzen und müssen noch weiter analysiert werden. In beiden Szenarien bekamen die interaktiven Objekte bessere Bewertungen in puncto überzeugender Simulation und Realismus, wenn haptisches Feedback vorhanden war. Die insgesamt sehr positive Resonanz lässt auch die Vermutung zu, dass die nur in Ausnahmefällen vorhandene VR-Erfahrung der Testpersonen zu den subjektiven Angaben beigetragen hat. Diese Tendenzen sind besonders im virtuellen Garten aufgefallen; bei den Passanten in der Stadt waren die Unterschiede zu der nicht haptischen Version geringer. Einige Testpersonen gaben als Begründung für diese Bewertung an, dass sie es als störend empfanden, dass Passanten durch sie oder durch Hindernisse hindurch gingen. Dies trat vor allem bei Personen auf, die versuchten, eine Kollisionen zu vermeiden, da die Passanten dahingehend programmiert waren, entweder auf jeden Fall mit der Testperson zusammenzustoßen oder ihnen auf jeden Fall auszuweichen.

Die Sicherheitsmaßnahmen erwiesen sich als angemessen. Es kam zu keinerlei nennenswerten negativen Vorfällen und das, schon zu Beginn der Testsitzung recht hohe, Vertrauen verschlechterte sich nicht. Auch das Unbehagen, mit dem Roboterarm zu interagieren, war von Anfang an sehr gering. Dadurch waren in diesen Punkten kaum Verbesserungen möglich und wurden auch nicht erwartet.

Einer der größeren Kritikpunkte betraf die Fortbewegung im System. Obwohl es keine Schwierigkeiten beim Erlernen der ungewohnten Bewegungsart gab, gaben einige Test-

personen an, dass es nur schwer möglich war sich exakt zu einer bestimmten Position zu Bewegen. Außerdem empfanden manche das Gehen als sehr unnatürlich und anstrengend. Ein weiterer negativer Punkt bezog sich auf die Interaktion mit dem Roboterarm in der Stadt. In manchen Fällen war die Kollision in der virtuellen Welt nicht synchron mit der Berührung durch den Handschuh. Vor allem die Dauer des Kontakts wurde in einigen Fällen als zu lange empfunden.

Die technische Evaluierung machte die Einschränkungen des verwendeten Arms deutlich. Sowohl bei der Auswertung der Genauigkeit als auch bei der Wiederholbarkeit zeigten sich die Schwächen des mit ca. 4.100,00 $ eher günstigen Roboters. Vor allem in ausgestreckten Posen wirken sich Schwerkraft und Last besonders stark auf den Arm aus. Auch die gemessenen Reaktionszeiten lagen in manchen Fällen deutlich über dem Durchschnitt. Die schlechten Werte für Genauigkeit und Reaktionszeit betreffen allerdings Extremsituationen, die in einer typischen Anwendung eher unwahrscheinlich sind. Die Leistung des Roboterarms war für das passiv-haptische Feedback in den meisten Fällen ausreichend.

6.1 Verbesserungsmöglichkeiten

Im Zuge der Implementierung dieses Systems und besonders im Laufe der Benutzerstudie offenbarten sich Verbesserungsmöglichkeiten. Manche können durch Ablaufänderungen umgesetzt werden, andere benötigen eine Anpassung in der Umsetzung.

Bereits zu Beginn der Konzeption wirkte sich die niedrige Leistung des Roboterarms aus. Bei zu schnellen Bewegungen oder schwereren Requisiten versagten manche der Servomotoren. Als Gegenmaßnahme wurde die maximale Geschwindigkeit eingeschränkt, was auch gleichzeitig der Sicherheit dient. Außerdem sollten weitere Props nur geringfügig mehr als 100 g wiegen, um einen reibungslosen Ablauf zu gewährleisten. Auch direkte Sonneneinstrahlung sollte vermieden werden, da dies die Ausgangstemperatur der Motoren erhöhen kann, was eine Überhitzung begünstigt. Die technische Evaluierung des Armes hat gezeigt, dass die Belastung bei weit von der Basis entfernten Punkten ansteigt. Daher sollte in Anwendungen darauf geachtet werden, diese Positionen nur so kurz wie möglich einzunehmen oder sie gar zu vermeiden.

Eine weitere Möglichkeit wäre der Einsatz eines anderen Roboterarms. Denkbare Kandidaten wären die Modelle aus der Stäubli TX2-Serie[Stä17]. Sie bieten bei einer ähnlichen Reichweite eine konstante Leistung im gesamten Arbeitsbereich. Die mögliche Traglast übersteigt die des verwendeten CrustCrawler Roboters bei weitem. Der spezielle Aufbau der Stäubli TX2 Arme ermöglicht es, mit einem Gelenk weniger auszukommen, wodurch auch die Komplexität der IK verringert wird.

Durch den allerdings deutlich höheren Preis ist ein System mit drei oder mehr Stäubli TX2 Roboterarmen eher unökonomisch. Als Kompensation hierfür wäre ein Aufbau denkbar, bei dem der Roboter um die Bewegungsplattform frei positioniert werden kann. Das ist beispielsweise durch ein Schienensystem realisierbar, auf dem der Roboter um den

Virtualizer verschoben wird. Eine solche Konstruktion beseitigt auch die Notwendigkeit, mehrere Arme zu synchronisieren, damit keine Kollisionen auftreten. Auch die komplexen Übergänge von dem Arbeitsbereich eines Roboters in den eines anderen fallen dadurch weg.

Die aktuelle Implementierung der OpenRAVE-Library unterstützt zwar das Öffnen und Schließen des Endeffektors, aufgenommene Gegenstände werden aber noch nicht in der Planung berücksichtigt. Bei bisherigen Anwendungen wurden die Requisiten zur Sicherheit fest mit dem Arm verbaut, da der CrustCrawler Roboter nicht genügend Kraft besitzt um diese verlässlich zu halten. Der Motor des Greifers benötigt bei langem Halten so viel Strom, dass die Leistung der anderen Motoren gedrosselt wird und sie somit leichter überlastet werden. Um einen reibungslosen Betrieb zu gewährleisten, müsste entweder das Gewicht der aufzunehmenden Objekte oder die maximale Geschwindigkeit der Motoren weiter reduziert werden. Daher wirkt sich die Einschränkung der OpenRAVE-Library nicht negativ aus. Bei der Verwendung eines stärkeren Roboterarms wäre eine Erweiterung um diese Funktion allerdings sinnvoll, um in einer Szene ein vielfältigeres Feedbackangebot zu ermöglichen. Bei der Erweiterung muss beachtet werden, dass die IK immer vom Fokuspunkt des Endeffektors abhängt. Bei dem in Abschnitt 5.2.1 beschrieben Stadt-Szenario liegt dieser Punkt in der Ausgangspose 20 cm über der Spitze des Roboters. Dadurch kann der Boxhandschuh mit der Testperson in der realen Welt kollidieren. Der Einsatz von vordefinierten Modellen für jede Requisite macht diese unterschiedlichen Fokuspunkte möglich. Bei dynamisch aufgenommenen Props bleibt der Punkt jedoch gleich, da die Geometrie des Arms nicht verändert wird. Um unterschiedliche Fokuspunkte zu unterstützen, kann ein virtueller Punkt verwendet werden. Anhand dieses Punktes kann eine Transformation in den echten Fokuspunkt erstellt werden. Diese wird anschließend auf jeden Zielpunkt angewendet, wodurch nur ein IK-Löser notwendig ist. OpenRAVE unterstützt das dynamische Laden und Entladen von 3D-Objekten aus XML-Dateien. Die gewünschten Objekte können aber auch zur Laufzeit anhand von Formen direkt in der Anwendung erzeugt werden.

In bestimmten Konstellationen benötigte das System deutlich länger für eine Reaktion als üblich. Dies lag zum einen an etwaigen Hindernissen, die großräumig umfahren wurden, zum anderen aber auch an der Zeit, die für die Serialisierung des Pfades benötigt wurde. Letzteres ist in der aktuellen Implementierung notwendig, um Pfade zwischen der 3D-Anwendung und OpenRAVE auszutauschen. Je länger dabei ein Pfad ist, desto zeitaufwändiger ist diese Operation. Dies tritt auch bei der Verwendung von vordefinierten Pfaden auf, da diese vor jeder Verwendung in OpenRAVE wieder deserialisiert werden müssen. Eine Möglichkeit, diesen Punkt zu umgehen, wäre, die erzeugten Pfade in der OpenRAVE-Library zu verwalten und nur Identifikationsnummern an die aufrufende Anwendung zu übermitteln. Die Sicherheitsfunktion, dass vor dem Befahren eines Pfades erneut überprüft wird, ob keine Hindernisse vorhanden sind, bleibt davon unbeeinflusst. Diese Maßnahme reduziert die langen Planungszeiten für komplexe Pfade nicht. Die Situationen, in denen diese Pfade auftreten, können aber durch die Wahl einer geeigneten Warteposition weitestgehend verhindert werden. Diese Warteposition wurden auch in

den in Abschnitt 5.2.1 erwähnten Szenarien verwendet. Wenn es die Anwendung erlaubt, sollte auch auf vorberechnete Pfade zurückgegriffen werden. Dies ist vor allem bei der Bewegung von der aufrechten Ausgangsposition zu einer Warteposition sinnvoll, da hier Start- und Zielposition bereits bekannt sind. Etwaige Pfadberechnungen können auch erst zur Laufzeit erfolgen, etwa wenn gerade keine Bewegung benötigt wird.

Das letzte Problem betrifft die eingesetzten Tracking-Methoden. Vor allem bei Vortests zur Benutzerstudie und bei längerer Verwendung zeigte sich ein Drift in den Daten des Perception Neuron. Dieser Effekt ist in der Nähe von metallischen Gegenständen besonders stark, was dazu führt, dass manche Körperteile mehr betroffen sind als andere. Dies erschwert eine programmatische Kompensation, da nicht erkennbar ist, wo ein Drift aufgetreten ist. Wie in Abschnitt 3.1 bereits erwähnt wurde, absorbieren die Materialien des Anzugs Infrarotlicht, wodurch auch optische Tracking-Systeme beeinträchtigt werden. Durch Überdecken dieser Materialien, wie etwa bei den verwendeten Überhandschuhen, kann dieser Nachteil allerdings ausgeglichen werden. Dadurch ist eine Kombination mit einem optischen Tracking-System möglich, das zur Erkennung einer möglichen Abweichung dienen kann. Das optische System muss dabei nicht zu jeder Zeit Daten liefern, da ein Drift nur langsam entsteht. Solange das optische System in mehr oder weniger regelmäßigen Abständen korrekte Daten liefert, können diese mit denen des Motion-Capture-Anzugs verglichen werden.
Eine weitere Möglichkeit wäre der Einsatz eines optischen Systems mit mehreren Kameras, wie etwa dem in Abschnitt 5.1 beschriebenen ARTTRACK1. Dabei muss darauf geachtet werden, dass das System mit eventuellen Verdeckungen umgehen kann, was Auswirkungen auf Position und Anzahl der Kameras hat. Ein Nachteil an der Verwendung eines solchen optischen Trackings ist allerdings, dass es meist an einen festen Ort gebunden ist. Die übrigen Komponenten des VR-Systems können hingegen leicht transportiert werden.

6.2 Anwendungsmöglichkeiten

Im aktuellen Ausbaugrad ist das physische Prop fest mit dem Arm verbaut. Daher hängen die Einsatzmöglichkeiten von der gewählten Requisite ab. Es kann sowohl als Simulator für statische Hindernisse, wie etwa eine Wand, verwendet werden, als auch als Feedback-System, das beispielsweise Berührungen fühlbar macht. Durch die noch relativ hohe Reaktionszeit und die Einschränkung auf einen Roboterarm ist die Abbildung von Treffern wie in [Hum11] nicht möglich. Mit den in Abschnitt 6.1 beschriebenen Adaptionen können noch komplexere Anwendungen realisiert werden.

Aufgrund des großen Platzbedarfs und der Vielzahl an verwendeten Komponenten eignet sich das System nicht für den Heimgebrauch. Der hohe Zeitaufwand für die Kalibrierung des Systems vor der Verwendung ist dazu noch ein weiteres Hindernis für den privaten Gebrauch. Eine denkbare Anwendung ist allerdings als vielseitige Simulationsumgebung ähnlich dem TurkDeck[Che+15]. Der geringere Platzbedarf und der niedrige personelle Aufwand dieses Systems ermöglichen dabei den Einsatz mehrerer Systeme auf derselben Fläche wie bei TurkDeck. Durch eine Veränderung an der Kabelführung kann auch auf die

Deckenmontage der Komponenten verzichtet werden. Damit lassen sich die Limitierungen durch Höhe und Beschaffenheit des Raumes weitgehend kompensieren. Bereits jetzt gibt es VR-Lounges und Cafés wie beispielsweise *Arcade - Virtual Reality Lounge*[Arc17]. Dort verwendete Virtualizer können durch die übrigen Komponenten dieses Systems leicht erweitert werden. Das Benutzererlebnis ist dabei stark von den virtuellen Welten abhängig, die wie bei TurkDeck speziell für das Feedback-System erstellt werden müssen.

Ein weiteres Anwendungsgebiet stellt jene als Therapieumgebung dar. Hierbei können zum Beispiel Alltags- bzw. Angstsituationen mit unterschiedlichen Realismus-Abstufungen umgesetzt werden. Als Einstieg kann nur das HMD verwendet werden, damit sich die Anwender und die Anwenderinnen an die visuellen Eindrücke gewöhnen können. In späteren Sitzungen können dann geräuschunterdrückende Kopfhörer, die Fortbewegung im Virtualizer, das Körpertracking und als Abschluss das passiv-haptische Feedback hinzu geschaltet werden. Dadurch lassen sich Anzahl und Intensität der Empfindungen an die Sensibilität der Benutzerin bzw. des Benutzers anpassen. Bei einem solchen Einsatz steht die Wiederholung derselben Aufgabe unter verschiedenen Gesichtspunkten im Vordergrund. Daher kann eine Therapieumgebung, verglichen mit einem VR-Café, mit weniger unterschiedlichen Anwendungen auskommen. Diese Szenarien können auch auf bestimmte Situationen beschränkt sein, was den Aufwand für die Erstellung weiter reduziert.

KAPITEL 7

Zusammenfassung und Ausblick

Diese Diplomarbeit beschreibt ein immersives VR-System für virtuelle Welten ohne räumliche Begrenzung. Dabei werden gebrauchsfertige Hardware-Komponenten mit einem speziell angefertigten Roboterarm kombiniert. Die hierbei erstellte Software-Lösung kapselt die Schnittstellen der einzelnen Geräte und ermöglicht so eine einfache Einbindung in neue Anwendungen. Als Ausgangspunkt für haptikunterstützte VR-Anwendungen dient eine Unity3D-Szene, die alle möglichen Komponenten enthält. Die erwähnten Sicherheitsvorkehrungen bilden die Grundlage für eine gefahrlose Verwendung des Systems. Einige der Ergebnisse wurden in einer internationalen wissenschaftlichen Publikation von Vonach et al.[VGK17] veröffentlicht.

In den durchgeführten Evaluierungen zeigte sich, dass ein Roboterarm in der Lage ist, passiv-haptisches Feedback zu präsentieren. Durch den Einsatz unterschiedlicher physischer Props können verschiedene Objekte dargestellt werden. Die Benutzerstudie hebt die Vorteile des Systems hervor. In einem Raum mit weniger als 16 m^2 Fläche konnte ein 15 m mal 15 m großer Garten mit einer fünf m breiten und 2,6 m hohen Wand simuliert werden. Die berührbare Höhe der Wand betrug, abhängig von der Entfernung zu ihr, bis zu 1,8 m. In demselben Raum konnten die Testpersonen über 100 m in einem virtuellen Häuserblock zurücklegen. Die verwendete Bewegungsplattform erlaubt dabei ein annähernd natürliches Bewegungsgefühl. Innerhalb dieser virtuellen Umgebungen konnten Testpersonen die Wand mit ihren Fingern ertasten oder Kollisionen mit entgegenkommenden Passanten spüren. Die Ergebnisse der Evaluierung deuten auf eine gute Akzeptanz des Systems hin.

Die Analyse der technischen Aspekte des Systems legte Verbesserungsmöglichkeiten offen. Die langsame Reaktionszeit des Roboters resultiert teilweise aus den Einschränkungen der maximalen Geschwindigkeit. Eine Gegenmaßnahme wäre die Definition eines Bereichs außerhalb der Reichweite des Anwenders bzw. der Anwenderin. In dieser Zone könnte der Roboter schneller bewegt werden, ohne die Verletzungsgefahr zu erhöhen. In der Nähe der Anwenderin bzw. des Anwenders bliebe die Geschwindigkeit eingeschränkt. Mit

einer kurzen Reaktionszeit wäre es möglich, einzelne Körperteile, wie etwa die Hände, zu verfolgen und somit an mehreren Orten haptisches Feedback zu erzeugen. Auch die teilweise geringe Genauigkeit des verwendeten Roboters fiel in der Evaluierung als Schwachstelle auf. Ein Austausch des Roboterarms durch ein stärkeres Modell würde die Genauigkeit der Bewegungen verbessern und gleichzeitig die Verwendung schwererer Requisiten ermöglichen.

Zukünftige Arbeiten können das Angebot an Feedbackmöglichkeiten ausbauen. Ein ausschlaggebendes Kriterium ist dabei die Richtung, aus der passiv-haptisches Feedback möglich ist. Drei oder mehr gleichmäßig um die Bewegungsplattform verteilte Roboterarme erlauben hierbei Feedback von jeder Seite des Anwenders bzw. der Anwenderin. Die größte Herausforderung stellt dabei die Synchronisation der Arme dar. Einerseits soll es im Überlappungsbereich zwischen zwei Roboterarmen nicht zu Kollisionen kommen, andererseits soll der Wechsel von einem Arm zu seinem Nachbarn möglichst nahtlos funktionieren. Im Idealfall bemerkt die Testperson nicht, ob sie mit einem oder mehreren Armen interagiert. Alternativ ist auch ein Schienensystem denkbar, auf dem der Roboter um die Bewegungsplattform bewegt werden kann. Dadurch ist die Reaktionszeit bei schnellen Richtungsänderungen zwar etwas größer, die aufwändige Synchronisation entfällt aber. Bei einer solchen Konstruktion muss besonders auf die Kabel geachtet werden, da der Arm beliebig oft um die Testperson fahren kann. Ein Lösungsansatz wäre, die Strom- und Datenleitungen von der Decke herab zu verlegen. Um den Aktionsbereich des Arms nicht einzuschränken, können die Kabel über eine Vorrichtung, ähnlich der Angel des Virtualizers, über den Arm hinweg gespannt bleiben.

Eine weitere Verbesserung ist der Einsatz verschiedener Oberflächen, wie etwa Gummi oder Stoff, bei neuen Props. Diese können mit den visuellen Eindrücken in der virtuellen Welt übereinstimmen oder ihnen bewusst widersprechen. Bei letzterem wäre es interessant, ob die Benutzerinnen bzw. die Benutzer diese Widersprüche akzeptieren oder, ob die Immersion dadurch beeinträchtigt wird. Mit weiteren Materialien ist es auch denkbar, komplexere Requisiten zu erstellen, die nicht nur aufgrund ihrer Oberfläche und Form Feedback erzeugen. Ein Beispiel wäre etwa ein Bedienfeld mit mehreren Schaltern, deren Betätigung wieder eine Aktion des Roboters auslöst. Diese Gegenstände können auch Sensoren beinhalten und somit Daten über die Interaktionen mit den Testpersonen sammeln.

Durch die Verbesserung der Roboterhardware sind auch Szenarien denkbar, die eine deutlich kürzere Reaktionszeit erfordern. Ein Beispiel dafür ist die Auswahl der verwendeten Requisite durch den Roboterarm in der laufenden Anwendung. Der Arm und die Planungssoftware müssen dabei schnell genug sein, den zusätzlichen Weg zwischen der Aufnahme der Requisite und der Positionierung zu kompensieren. Zusätzlich muss der Roboterarm über ausreichend Kraft besitzen, das Objekt so fest zu halten, dass es sich bei der Interaktion mit den Benutzer bzw. die Benutzerin nicht aus dem Griff löst. Dadurch wird die mögliche Vielfalt an darstellbaren Objekten erhöht. Durch kürzere Reaktionszeiten wäre es auch möglich, durchgehende Oberflächen darzustellen. Dazu können die Requisiten mittels der Trackingdaten der Hände positioniert werden, anstatt

an einer festen Position zu bleiben, solange sich die Person nicht im Raum bewegt.

Mit diesen Verbesserungen wäre eine Simulationsumgebung denkbar, die das Training bestimmter Aufgaben unter realistischen Bedingungen zulässt. So können beispielsweise Rettungskräfte verschiedene Situationen erleben, ohne dafür ganze Gebäude vorzubereiten. Zusätzlich kann eine solche Simulation deutlich schneller wiederholt werden und auch der Wechsel zu einer anderen Aufgabe lässt sich durch die Auswahl der entsprechenden VR-Anwendung leicht bewerkstelligen. Ein weiteres Beispiel ist die Ausbildung von Lehrlingen an verschiedenen Objekten. Ein Industrie-Lehrling kann so bereits im Vorfeld die Tätigkeiten üben, ohne Teile der Anlage zu belegen oder zu beschädigen.

Abbildungen

Tabellen

Listings

Glossar

Freiheitsgrad Freiheitsgrade beschreiben die Anzahl an unabhängigen Bewegungsmöglichkeiten.[BS13, S. 44]. 7–11, 14, 23, 38, 93

PInvoke „*Platform Invocation Services (*PInvoke*) erlauben es verwaltetem Code, nicht verwaltete Funktionen aufzurufen, die in einer* Dynamic Link Library *(*DLL*) implementiert sind.*“ [Mic15d]. 46, 55

Singleton Das Singleton Entwurfsmuster sichert ab, dass von einer Klasse genau ein Exemplar existiert und es einen globalen Zugriffspunkt darauf gibt.[Gam+11, S. 157]. 51, 56

WCF „*Windows Communication Foundation (*WCF*) ist ein Framework zur Erstellung dienstorientierter Anwendungen. Mit WCF können Sie Daten als asynchrone Nachrichten von einem Dienstendpunkt an einen anderen senden.*“ [Mic15e]. 56

Akronyme

ARM *Argonne Remote Manipulator*. 8

DK1 *Oculus Rift Development Kit 1*. 39

DK2 *Oculus Rift Development Kit 2*. 29, 33, 38–41, 59, 60, 62, 74, 93

DLL *Dynamic Link Library*. 95

FPS *First-Person-Shooter*. 18

HMD *Head-Mounted Display*. 1, 17, 18, 27, 29, 30, 32, 39, 40, 59, 74, 77, 87, 93

IK *Inverse Kinematic*. 4, 33, 34, 44, 47, 49–54, 58, 67, 84, 85

M Mittelwert. 75, 76, 82

MHI *Mobile Haptic Interface*. 11, 12, 93

MUTEX *mutual exclusion*. 50, 51

OpenRAVE *Open Robotics Automation Virtual Environment*. 3–5, 27, 33–35, 46–51, 53, 54, 56–58, 60, 62, 63, 69, 72, 73, 85, 93

RAII *Ressourcen-Erwerb ist Initialisierung*. 51

RSD *Robitic Shape Display*. 22–25, 93

SAD *Shape Approximation Device*. 23, 24

SD Standardabweichung. 67, 70, 75–80, 82

SDK *Software Development Kit*. 58, 59

SSQ *Fragebogen zur Erhebung von Simulatorkrankheit*. 74, 75, 77

TTL *Transistor-Transistor-Logik*. 42

VR *Virtual Reality*. vii, ix, 1–5, 7, 17, 23, 27, 29, 33, 35, 37, 39, 42, 46, 47, 50, 59, 70, 71, 73, 74, 83, 86, 87, 89, 91

VRPN *Virtual-Reality Peripheral Network*. 11

WYSIWYF *What You can See Is What You can Feel*. 24, 25, 93

Literatur

[And+06] Andrews, S. ; Mora, J. ; Lang, J. ; Lee, W. S.: Hapticast: A Physically-Based 3D Game With Haptic Feedback. In: *Proceedings of FuturePlay* (2006)

[Arc17] Arcade - Virtual Reality Lounge: *Arcade - Virtual Reality Lounge - Bad Hersfeld*. 2017. URL: http://arcade-lounge.de/ (besucht am 22. 04. 2017)

[AJK03] Atkins, J. E. ; Jacobs, R. a. ; Knill, D. C.: Experience-Dependent Visual Cue Recalibration Based On Discrepancies Between Visual And Haptic Percepts. In: *Vision Research* 43 (2003) Nr. 25, S. 2603–2613. ISSN: 00426989. DOI: 10.1016/S0042-6989(03)00470-X

[Ben04] Benali-Khoudja, M.: Tactile Interfaces: A State-Of-The-Art Survey. In: *Int. Symposium On Robotics* 31 (2004), S. 23–26. ISSN: 10974199. DOI: 10.1016/j.neuron.2010.01.010

[Bol+14] Bolton, J. ; Lambert, M. ; Lirette, D. ; Unsworth, B.: PaperDude: A Virtual Reality Cycling Exergame. In: *Proceedings of the Extended Abstracts of the 32nd Annual ACM Conference On Human Factors In Computing Systems - CHI EA '14*. New York, New York, USA: ACM Press, 2014, S. 475–478. ISBN: 9781450324748. DOI: 10.1145/2559206.2574827

[BS13] Brommundt, E. ; Sachs, G.: *Technische Mechanik: Eine Einführung*. Springer-Lehrbuch. Springer Berlin Heidelberg, 2013. ISBN: 9783642883590

[Bro+90] Brooks, F. P. ; Ouh-Young, M. ; Batter, J. J. ; Jerome Kilpatrick, P.: Project GROPE Haptic Displays For Scientific Visualization. In: *ACM SIGGRAPH Computer Graphics* 24 (1990) Nr. 4, S. 177–185. ISSN: 00978930. DOI: 10.1145/97880.97899

[CH14] Cakmak, T. ; Hager, H.: Cyberith Virtualizer: A Locomotion Device For Virtual Reality. In: *ACM SIGGRAPH 2014 Emerging Technologies*. 2014. ISBN: 9781450329613

[Car+13] Carter, T. ; Seah, S. A. ; Long, B. ; Drinkwater, B. ; Subramanian, S.: UltraHaptics. In: *Proceedings of the 26th Annual ACM Symposium On User Interface Software And Technology - UIST '13* (2013), S. 505–514. doi: 10.1145/2501988.2502018

[Che+15] Cheng, L.-P. ; Roumen, T. ; Rantzsch, H. ; Köhler, S. ; Schmidt, P. ; Kovacs, R. ; Jasper, J. ; Kemper, J. ; Baudisch, P.: TurkDeck: Physical Virtual Reality Based On People. In: *Proceedings of the 28th Annual ACM Symposium On User Interface Software & Technology - UIST '15* (2015), S. 417–426. doi: 10.1145/2807442.2807463

[Col14] Colgan, A.: *How Does the Leap Motion Controller Work?* 2014. url: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/ (besucht am 10. 09. 2016)

[Cru] CrustCrawler Inc: *Crustcrawler Pro-Series Robotic Arm components.* url: http://www.crustcrawler.com/products/ProRoboticArm/ (besucht am 13. 09. 2015)

[Cyb15] Cyberith GmbH: *Cyberith | Reality is not enough...* 2015. url: http://cyberith.com/ (besucht am 13. 09. 2015)

[DR09] Daunay, B. ; Régnier, S.: Stable Six Degrees of Freedom Haptic Feedback For Flexible Ligand-Protein Docking. In: *CAD Computer Aided Design* 41 (2009) Nr. 12, S. 886–895. issn: 00104485. doi: 10.1016/j.cad.2009.06.010

[DAR16] Dawnes, B. ; Abrahams, D. ; Rivera, R.: *Boost C++ Libraries.* 2016. url: http://www.boost.org/ (besucht am 30. 12. 2016)

[Dia10] Diankov, R.: *Automated Construction of Robotic Manipulation Programs.* Diss. 2010, S. 1–263

[Dia13a] Diankov, R.: *Format:XML - OpenRAVE.* 2013. url: http://openrave.programmingvision.com/wiki/index.php/Format:XML (besucht am 30. 12. 2016)

[Dia13b] Diankov, R.: *Introduction to the OpenRAVE Architecture 0.9.0.* 2013. url: http://openrave.org/docs/latest%5C_stable/coreapihtml/architecture%5C_concepts.html (besucht am 13. 09. 2015)

[Dia13c] Diankov, R.: *OpenRAVE | COLLADA Robot Extensions (Version 0.3.3) | OpenRAVE Documentation.* 2013. url: http://openrave.org/docs/latest%5C_stable/collada%5C_robot%5C_extensions/ (besucht am 30. 12. 2016)

[Dia13d] Diankov, R.: *OpenRAVE | Welcome to Open Robotics Automation Virtual Environment | OpenRAVE Documentation.* 2013. url: http://openrave.org/docs/latest%5C_stable/ (besucht am 16. 09. 2015)

[Ele15] Electronic Arts Inc: *Battlefield 3 (BF3) | Offizielle Site*. 2015. URL: http://www.battlefield.com/de/battlefield3 (besucht am 28. 08. 2015)

[Fon+13] Fontana, M. ; Fabio, S. ; Marcheschi, S. ; Bergamasco, M.: Haptic Hand Exoskeleton For Precision Grasp Simulation. In: *Journal of Mechanisms And Robotics* 5 (2013) Nr. 4, S. 41014. ISSN: 1942-4302. DOI: 10.1115/1.4024981

[Fri+05] Frisoli, A. ; Rocchi, F. ; Marcheschi, S. ; Dettori, A. ; Salsedo, F. ; Bergamasco, M.: A New Force-Feedback Arm Exoskeleton For Haptic Interaction In Virtual Environments. In: *First Joint Eurohaptics Conference And Symposium On Haptic Interfaces For Virtual Environment And Teleoperator Systems*. IEEE, 2005, S. 195–201. ISBN: 0-7695-2310-2. DOI: 10.1109/WHC.2005.15

[GRS11] Gallotti, P. ; Raposo, A. ; Soares, L.: V-Glove: A 3D Virtual Touch Interface. In: *2011 XIII Symposium On Virtual Reality*. IEEE, Mai 2011, S. 242–251. ISBN: 978-1-4577-0661-5. DOI: 10.1109/SVR.2011.21

[Gam+11] Gamma, E. ; Johnson, R. ; Helm, R. ; Vlissides, J.: *Entwurfsmuster: Elemente Wiederverwendbarer Objektorientierter Software*. 6. Aufl. Programmer's choice. München: Pearson Deutschland, 2011. ISBN: 9783827330437

[Han+08] Han, K.-l. ; Choi, O. K. ; Lee, I. ; Hwang, I. ; Lee, J. S. ; Choi, S.: Design And Control of Omni-Directional Mobile Robot For Mobile Haptic Interface. In: *International Conference On Control, Automation And Systems* (2008), S. 1290–1295

[Hap15] Haption SA: *Haption SA - Hardware*. 2015. URL: http://www.haption.com/site/index.php/en/products-menu-en/hardware-menu-en (besucht am 22. 08. 2015)

[HH95] Hirota, K. ; Hirose, M.: Simulation And Presentation of Curved Surface In Virtual Reality Environment Through Surface Display. In: *Proceedings Virtual Reality Annual International Symposium '95*. IEEE Comput. Soc. Press, 1995, S. 211–216. ISBN: 0-8186-7084-3. DOI: 10.1109/VRAIS.1995.512498

[Hor10] Hornyak, T.: *Engineers Turn Robot Arm into Formula 1 Simulator - IEEE Spectrum*. 2010. URL: http://spectrum.ieee.org/automaton/robotics/industrial-robots/engineers-turn-robot-arm-into-formula-1-simulator (besucht am 11. 08. 2015)

[Hum11] Humphries, M.: *The ultimate Battlefield 3 simulator has been created | Games | Geek.com*. 2011. URL: http://www.geek.com/games/the-ultimate-battlefield-3-simulator-has-been-created-1435731/ (besucht am 04. 08. 2015)

[Ins01] Insko, B. E.: *Passive Haptics Significantly Enhances Virtual Environments*. Diss. The University of North Carolina at Chapel Hill, 2001, S. 100. isbn: 0-493-17286-6

[Ken+93] Kennedy, R. S. ; Lane, N. E. ; Berbaum, K. S. ; Lilienthal, M. G.: Simulator Sickness Questionnaire: An Enhanced Method For Quantifying Simulator Sickness. In: *The International Journal of Aviation Psychology* 3 (Juli 1993) Nr. 3, S. 203–220. issn: 1050-8414. doi: 10.1207/s15327108ijap0303_3

[KUK15] KUKA Roboter GmbH: *KUKA Roboter GmbH: Industrieroboter*. 2015. url: http://www.kuka-robotics.com/de/

[Lam+07] Lambercy, O. ; Dovat, L. ; Gassert, R. ; Burdet, E. ; Teo, C. L. ; Milner, T.: A Haptic Knob For Rehabilitation of Hand Function. In: *IEEE Transactions On Neural Systems And Rehabilitation Engineering* 15 (2007) Nr. 1, S. 356–366. issn: 15344320. doi: 10.1109/TNSRE.2007.903913

[Lea16a] Leap Motion Inc: *Leap Motion | 3D Motion and Gesture Control for PC & Mac*. 2016. url: https://www.leapmotion.com/product/desktop (besucht am 28. 08. 2016)

[Lea16b] Leap Motion Inc: *Orion | Leap Motion Developers*. 2016. url: https://developer.leapmotion.com/orion (besucht am 28. 08. 2016)

[Lin+04] Lindeman, R. W. ; Page, R. ; Yanagida, Y. ; Sibert, J. L.: Towards Full-Body Haptic Feedback. In: *Proceedings of the ACM Symposium On Virtual Reality Software And Technology - VRST '04*. New York, New York, USA: ACM Press, 2004, S. 146. isbn: 1581139071. doi: 10.1145/1077534.1077562

[Lin99] Lindeman, R. W.: *Bimanual Interaction, Passive-Haptic Feedback, 3D Widget Representation, And Simulated Surface Constraints For Interaction In Immersive Virtual Environments*. Diss. The George Washington University, 1999, S. 148

[Max15] Max-Planck-Campus Tübingen: *Das Max-Planck-Institut für biologische Kybernetik*. 2015. url: http://www.kyb.tuebingen.mpg.de/de.html (besucht am 29. 08. 2015)

[McN93] McNeely, W.: Robotic Graphics: A New Approach To Force Feedback For Virtual Reality. In: *Proceedings of IEEE Virtual Reality Annual International Symposium*. IEEE, 1993, S. 336–341. isbn: 0-7803-1363-1. doi: 10.1109/VRAIS.1993.380761

[Mic15a] Microsoft: *Kinect - Windows app development*. 2015. url: https://dev.windows.com/en-us/kinect (besucht am 29. 08. 2015)

[Mic15b] Microsoft: *Kinect for Windows Sensor Components and Specifications*. 2015. URL: https://msdn.microsoft.com/en-us/library/jj131033.aspx (besucht am 16. 09. 2015)

[Mic15c] Microsoft: *Kinect hardware*. 2015. URL: https://dev.windows.com/en-us/kinect/hardware (besucht am 16. 09. 2015)

[Mic15d] Microsoft: *Platform Invoke Tutorial (C#)*. 2015. URL: https://msdn.microsoft.com/en-us/library/aa288468.aspx (besucht am 15. 09. 2015)

[Mic15e] Microsoft: *Was ist die Windows Communication Foundation*. 2015. URL: http://msdn.microsoft.com/de-de/library/ms731082.aspx (besucht am 15. 09. 2015)

[MF07] Minamizawa, K. ; Fukamachi, S.: Gravity Grabber: Wearable Haptic Display To Present Virtual Mass Sensation. In: *Proceedings of SIGGRAPH 2007* (2007), Article 8. DOI: 10.1145/1278280.1278289

[Min+08] Minamizawa, K. ; Kamuro, S. ; Fukamachi, S. ; Kawakami, N. ; Tachi, S.: GhostGlove: Haptic Existence of the Virtual World. In: *ACM SIGGRAPH 2008 New Tech Demos On - SIGGRAPH '08*. New York, New York, USA: ACM Press, 2008, S. 1–1. ISBN: 9781605584669. DOI: 10.1145/1401615.1401633

[Mur+04] Murayama, J. ; Luo, Y. ; Akahane, K. ; Hasegawa, S. ; Sato, M.: A Haptic Interface For Two-Handed 6DOF Manipulation-SPIDAR-G&G System. In: *IEICE Transactions On Information And Systems* E87-D (2004) Nr. 6, S. 1415–1421. ISSN: 09168532

[Nic+13] Nicolescu, A. ; Ivan, M. ; Avram, C. ; Dobrescu, T.: Volumetric Accuracy Experimental Evaluation And 3D Error Map Generation For A Kawasaki FS 10 E Articulated Arm Industrial Robot. In: *Recent Advances In Robotics, Aeronautical And Mechanical Engineering* (2013), S. 63–68

[Noi16] Noitom: *Perception Neuron by Noitom | Perception Neuron motion capture for virtual reality, animation, sports, gaming and film*. 2016. URL: https://neuronmocap.com/products/perception%5C_neuron (besucht am 28. 08. 2016)

[OCU15a] OCULUS VR LLC: *Developer Center — Documentation and SDKs | Oculus*. 2015. URL: https://developer.oculus.com/documentation/pcsdk/latest/concepts/dg-hardware-setup/

[OCU15b] OCULUS VR LLC: *Kickstarter » Oculus Rift: Step Into the Game von Oculus*. 2015. URL: https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game/posts/380099 (besucht am 15. 09. 2015)

[OCU15c] OCULUS VR LLC: *Oculus Rift | Oculus - Oculus VR*. 2015. URL: https://www.oculus.com/en-us/rift/

[PVL13] Pavlik, R. A. ; Vance, J. M. ; Luecke, G. R.: Interacting With A Large Virtual Environment By Combining A Ground-Based Haptic Device And A Mobile Robot Base. In: *Volume 2B: 33rd Computers And Information In Engineering Conference*. ASME, Aug. 2013, V02BT02A029. ISBN: 978-0-7918-5586-7. DOI: 10.1115/DETC2013-13441

[Pro14] Provancher, W. R.: Creating Greater VR Immersion By Emulating Force Feedback With Ungrounded Tactile Feedback. In: *IQT QUARTERLY FALL* 6 (2014) Nr. 2, S. 18–21

[Ran+10] Ranky, R. ; Sivak, M. ; Lewis, J. ; Gade, V. ; Deutsch, J. E. ; Mavroidis, C.: VRACK - Virtual Reality Augmented Cycling Kit: Design And Validation. In: *2010 IEEE Virtual Reality Conference (VR)*. IEEE, März 2010, S. 135–138. ISBN: 978-1-4244-6237-7. DOI: 10.1109/VR.2010.5444798

[Rob06] Robles-De-La-Torre, G.: The Importance of the Sense of Touch In Virtual And Real Environments. In: *IEEE Multimedia* 13 (2006) Nr. 3, S. 24–30

[ROB15] ROBOTIS INC: *ROBOTIS - DYNAMIXEL*. 2015. URL: http://en.robotis.com/index/product.php?cate%5C_code=101011 (besucht am 13. 09. 2015)

[ROB10] ROBOTIS INC: *ROBOTIS - Links to Drawings - Dynamixel, Bioloid, Frames*. 2010. URL: http://en.robotis.com/BlueAD/board.php?bbs%5C_id=downloads%5C&mode=view%5C&bbs%5C_no=26324%5C&page=1%5C&key=%5C&keyword=%5C&sort=%5C&scate=DRAWING (besucht am 30. 12. 2016)

[SRS00] Sallnäs, E.-L. ; Rassmus-Gröhn, K. ; Sjöström, C.: Supporting Presence In Collaborative Environments By Haptic Force Feedback. In: *ACM Transactions On Computer-Human Interaction* 7 (2000) Nr. 4, S. 461–476. ISSN: 10730516. DOI: 10.1145/365058.365086

[Sen15] Sensable: *Haptic Devices - Sensable*. 2015. URL: http://www.dentsable.com/products-haptic-devices.htm (besucht am 22. 08. 2015)

[Shi+12] Shin, S. ; Lee, I. ; Lee, H. ; Han, G. ; Hong, K. ; Yim, S. ; Lee, J. ; Park, Y. ; Byeong, K. K. ; Ryoo, D. H. ; Kim, D. W. ; Choi, S. ; Chung, W. K.: Haptic Simulation of Refrigerator Door. In: *2012 IEEE Haptics Symposium (HAPTICS)*. IEEE, März 2012, S. 147–154. ISBN: 978-1-4673-0809-0. DOI: 10.1109/HAPTIC.2012.6183783

[SPB13] Sinclair, M. ; Pahud, M. ; Benko, H.: TouchMover: Actuated 3D Touchscreen With Haptic Feedback. In: *Proceedings of the ACM International Conference On Interactive Tabletops And Surfaces (ITS '13)* (2013), S. 287–296. doi: 10.1145/2512349.2512805

[Six15] Sixense Entertainment, I.: *STEM System | Sixense*. 2015. url: http://sixense.com/wireless (besucht am 23. 08. 2015)

[SUS95] Slater, M. ; Usoh, M. ; Steed, A.: Taking Steps: the Influence of A Walking Technique On Presence In Virtual Reality. In: *ACM Transactions On Computer-Human Interaction* 2 (1995) Nr. 3, S. 201–219. issn: 10730516. doi: 10.1145/210079.210084

[Sod+13] Sodhi, R. ; Poupyrev, I. ; Glisson, M. ; Israr, A.: AIREAL: Interactive Tactile Experiences In Free Air. In: *ACM Transactions On Graphics* 32 (Juli 2013) Nr. 4, S. 1. issn: 07300301. doi: 10.1145/2461912.2462007

[Spr08] Springfield, J.: *Inheriting From a Native C++ Class in C# | Visual C++ Team Blog*. 2008. url: https://blogs.msdn.microsoft.com/vcblog/2008/12/08/inheriting-from-a-native-c-class-in-c/ (besucht am 29. 12. 2016)

[Stä17] Stäubli: *Roboter für niedrige Traglasten: TP, TS SCARA, TX und TX2 Sechsachser*. 2017. url: http://www.staubli.com/de/robotik/roboterarme/niedrige-traglasten/ (besucht am 18. 04. 2017)

[Tac+94] Tachi, S. ; Maeda, T. ; Hirata, R. ; Hoshino, H.: A Construction Method of Virtual Haptic Space. In: *Proceedings of the 4th International Conference On Artificial Reality And Tele-Existence (ICAT'94)* (1994), S. 131–138

[Trü03] Trübswetter, C.: *Analyzing And Monitoring Tracking Accuracy of An A.R.T. System*. Diss. TU München, 2003

[Tüb10a] Tübingen, M.-P.-C.: *Flugtraining der besonderen Art*. 2010. url: http://tuebingen.mpg.de/aktuelles-presse/pressemitteilungen/detail/flugtraining-der-besonderen-art.html (besucht am 29. 08. 2015)

[Tüb10b] Tübingen, M.-P.-C.: *Heli Trainer setzt neue Maßstäbe bei der Pilotenausbildung*. 2010. url: http://tuebingen.mpg.de/aktuelles-presse/pressemitteilungen/detail/heli-trainer-setzt-neue-massstaebe-bei-der-pilotenausbildung.html (besucht am 29. 08. 2015)

[Uni15a] Unity Technologies: *Unity - Editor*. 2015. url: https://unity3d.com/unity/editor (besucht am 13. 09. 2015)

[Uni15b] Unity Technologies: *Unity - Game Engine*. 2015. url: http://unity3d.com/ (besucht am 13. 09. 2015)

[VGK17] Vonach, E. ; Gatterer, C. ; Kaufmann, H.: VRRobot : Robot Actuated Props In An Infinite Virtual Environment. In: *IEEE Virtual Reality 2017*. 2017, S. 10

[Wil05] Willms, A.: *C++ Master Class: Einstieg Für Anspruchsvolle*. 1. Aufl. Master-Class. Pearson Studium, 2005. ISBN: 9783827321824

[WM07] Wollacott, A. M. ; Merz, K. M.: Haptic Applications For Molecular Structure Manipulation. In: *Journal of Molecular Graphics And Modelling* 25 (2007) Nr. 6, S. 801–805. ISSN: 10933263. DOI: 10.1016/j.jmgm.2006.07.005

[YHK96] Yokokohji, Y. ; Hollis, R. L. ; Kanade, T.: What You Can See Is What You Can Feel - Development of A Visual/Haptic Interface To Virtual Environment. In: *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium* (1996), S. 46–53

[YOS17] YOST LABS: *PrioVR Dev Kit – Yost Labs*. 2017. URL: https://yostlabs.com/priovr/ (besucht am 10. 09. 2017)