Lehrstuhl für Raumfahrttechnik

Prof. Prof. h.c. Dr. Dr. h.c.
Ulrich Walter

Technische Universität München

Bachelorarbeit

Entwicklung und Implementierung einer Nutzerschnittstelle für teleoperierte Nahbereichsoperationen von Satelliten in Virtual Reality

RT-BA 2018/10

Autor:

Xiaozhou Luo

Betreuer: Dipl.-Ing. (Univ.) Martin Dziura
Lehrstuhl für Raumfahrttechnik / Institute of Astronautics
Technische Universität München

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mir tatkräftige Unterstützung bei der Erstellung meiner Bachelorarbeit gegeben haben.

Mein besonderes Dankeschön geht an meinen Betreuer Martin Dziura, der mich bei meiner Arbeit so gut es geht unterstützt hat und bei Fragen oder Unklarheiten sich immer die dafür notwendige Zeit und Geduld nimmt, sie zweifelsfrei zu beantworten bzw. zu beseitigen.

Ebenso geht ein ausdrücklicher Dank an Maximilian Prexl, der mir besonders in Fragen bzgl. der VR-Anbindung und der Programmierung in Unity weitergeholfen hat und mir dabei Anregungen zur Entwicklung und Implementierung der Steuerungsmethode gegeben hat.

Natürlich bedanke ich mich an dieser Stelle auch bei allen Mitgliedern meines RACOON Jahrgangs und allen Mitarbeitern des LRT-Lehrstuhls für die wunderbare und produktive Zusammenarbeit in den letzten sechs Monaten.

Zum Schluss möchte ich mich noch bei denjenigen bedanken, die mir beim Korrekturlesen dieser Arbeit geholfen haben.

Zusammenfassung

Die NASA forschte bereits seit Beginn der 1990er Jahre an der Möglichkeit die VR-Technologie in der Raumfahrt einzusetzen. Das RACOON-Lab hat dafür ein VR-Programm entwickelt, welches ein On Orbit Servicing simuliert und es in einer virtuellen Umgebung darstellt. Das VR-Programm soll um eine Steuerungskomponente erweitert werden, um die Steuerungsoperationen auch in VR durchführen zu können.

Für die Interaktion mit der virtuellen Umgebung wird die VR-Brille HTC VIVE verwendet, die mit Hilfe von UnityEngine angesteuert wird.

Dabei wird auf die menschlichen Informationsverarbeitungsprozess eingegangen und die in der VR stimulierten Wahrnehmungsmöglichkeiten identifiziert. Daraus folgend werden die Anforderungen für die Steuerungsmethode und die der gesamten Nutzerschnittstelle definiert.

Auf Basis dieser Rahmenbedingungen werden zwei verschiedenen Eingabekonzepte entwickelt. Für die endgültige Implementierung wurde eine Symbiose aus den beiden Methoden getroffen. Dabei werden alle Komponenten für die Steuerungseingabe in den rechten Controller zusammengefasst und die für die Steuerung der Kameraperspektive zuständigen Komponenten in einer neuen Nutzerschnittstelle in den linken Controller zusammengefasst.

Zusätzlich zu der Steuerungsmethode wurde die Kamerasteuerung ebenfalls um eine Steuerungsfunktion erweitert, bei der die Kameraposition frei verschieblich ist. Zu Orientierung wurden verschiedene Hilfssysteme entwickelt und implementiert, die dem Nutzer bei der Steuerung unterstützet.

Das fertige VR-Programm wurde zum Schluss getestet und das Steuerungssystem funktioniert zuverlässig und alle Hilfssysteme erfüllen ihre vorgesehene Aufgabe. Lediglich die variable Kamerasteuerung muss zur Stabilisierung der Bildübertragung nochmal bearbeitet werden.

Abstract

The NASA has been researching and developing the virtual reality technology since the 1990s for the usage in astronautics. The RACOON-Lab has developed a virtual reality software as a visualization extension for their program that simulates On Orbit Servicing processes. In this thesis a flight control component will be added to the VR visualizer program.

The program is visualized on the VR-gear HTC VIVE and the headset is connected to the game engine Unity for data processing.

The process of cognition will be presented and all of types of methods of cognition that is used within the technology of virtual reality will be identified. Because of that requirements will be defined for the flight control component and for the user interface.

At this stage two different types of flight control input methods will be designed and evaluated. As a result a combination of these two types is chosen for the integration. Although all components of the flight control component will be stored in the controller on the right-hand side while all the components if the camera control are in the controller on the left-hand side.

In additional to the integration of the flight control component, the ability of the camera control unit will be expanded with the same function for higher variety of perspectives. For a better orientation within the control process, a handful auxiliary systems will be integrated.

At the end the finalized software has been tested with the result that the main control unit for the flight control and the auxiliary systems are fully working. Only the camera control unit suffers a malfunction and needs to be edited for a more reliable connection to the simulation software.

Inhaltsangabe

Abbildungsverzeichnis

Tabellenverzeichnis

Symbole und Formelzeichen

Symbol	SI-Einheit	Beschreibung
x	[m]	Wert der x-Koordinate
y	[m]	Wert der y-Koordinate
z	[m]	Wert der z-Koordinate
$\Delta v_{Rotation}$	[°/s]	relative Rotationsgeschwindigkeit

Abkürzungen

Abb.	Abbildung
ADI	Advanced Deviation Indicator
AR	Augmented Reality
BDI	Basic Deviation Indicator
CSA	Canadian Space Agency
DLR	Deutsches Zentrum für Luft- und Raumfahrt
EVA	Extra-Vehicular Activity
Fkt.	Funktion
FoV	Field of View
HFS	Haptic Feedback System
HIL	Hardware-in-the-Loop
HST	Hubble Space Telescope
HUD	Head Up Display
IEEE	Institute of Electrical and Electronics Engineers
IPC	Interprozesskommunikation
IVE	Immersive Virtual Environment
JSC	Johnson Space Center
KOSY	Koordinatensystem
LRT	Lehrstuhl für Raumfahrttechnik
MCC	Mission Control Center
NASA	National Aeronautics and Space Administration
OFS	Optical Feedback System
OOS	On Orbit Servicing
RACOON-Lab	Real-Time Attitude Control and On-Orbit Navigation Laboratory
RFC	Requirement of Flight Control
RRS	RACOON Reference Scenario
RUI	Requirement of the User Interface
SSRMS	Space Station Remote Manipulator System
STB	Software Technology Branch
Tab.	Tabelle
TUM	Technische Universität München
UI	User Interface

UIS	User Input System
VR	Virtual Reality
VR-OOS	Virtual Reality for On Orbit Servicing
VRT	Virtual Reality Trainer

1 Einleitung

In diesem Kapitel erfolgt die Hinführung zu der Kernaufgabe der Arbeit. Dabei wird nach der Motivation für die intensive Auseinandersetzung mit diesem Thema ein Überblick über den aktuellen Stand der Technik gewährt und die konkrete Aufgabenstellung und das Ziel dieses Dokuments festgelegt.

1.1 Motivation der Arbeit

Das Hubble Space Telescope (HST) ist das erste Weltraumteleskop, das im Rahmen des Great Observatory Projekts von der National Aeronautics and Space Administration (NASA) und der Europäischen Weltraumorganisation (ESA) ins Weltall gebracht wurde [1]. Bereits kurz nach dem Betriebsbeginn stellte sich heraus, dass das Teleskop einen optischen Fehler am Hauptspiegel aufweist und die Bilder, die das Teleskop aufgenommen hat, einen sehr geringen wissenschaftlichen Wert haben. Erst die erste Servicemission drei Jahre nach dem Aussetzen des Teleskops korrigierte die Abweichung und das Teleskop lieferte zum ersten Mal Bilder mit ausreichender Auflösung [2]. Die Servicemission SM1 markiert nicht nur eine neue Ära der Astronomie, sondern auch den Beginn einer neuen Art und Weise der Missionsvorbereitung.

Wegen der hohen Anzahl der Außenbordeinsätze, auch Extra-Vehicular-Activities (EVA) genannt, und ihrer hohen Komplexität mussten sowohl die Astronauten, die die eigentliche Arbeit im Weltall ausführten, als auch die Beteiligten am Boden im Mission Control Center (MCC) für die jeweiligen Situationen ausgebildet werden. Die NASA erforscht bereits seit dem Beginn der 1990er Jahre die Anwendung einer virtuellen Umgebung in der Raumfahrt und mit der SM1 wurde diese Technik zum ersten Mal in der Geschichte der Raumfahrt bei der Missionsvorbereitung angewendet. Dazu wurden insgesamt sechs EVA-Szenarien in eine Simulation implementiert und damit über 100 Missionsbeteiligte am Boden geschult. Eine nachträglich aufgenommene Studie belegt, dass die Schulung mittels der damals sich noch als Virtual Environment bezeichnende Technologie einen nachweisbaren Mehrwert in der Arbeitsleistung der Beteiligten bewirkt und das Verständnis bzgl. prozeduraler Aufgaben verstärkt [3].

Fortan wurde die virtuelle Realität (VR) in der Raumfahrt schwerpunktmäßig für die Missionstrainings weiterentwickelt und sie wird heutzutage beispielsweise für einen Teil der Vorbereitung von EVA-Einsätzen an der Internationalen Raumstation ISS verwendet [4, 5].

Parallel mit dem Voranschreiten der Technologie im VR-Bereich erfüllen zunehmend kommerziell erhältliche VR-Systeme die Anforderungen der NASA. Wurde früher eine Sonderanfertigung im fünfstelligen Dollarbetrag für die Visualisierung und Interaktion benötigt, ist es heutzutage möglich, die geforderten Spezifikationen mittels eines wesentlich günstigeren Serienmodells zu erfüllen [4]. Damit wird der Einsatz der VR-Technologie auch an universitäre Forschungsprojekte interessant. Eine ähnliche Entwicklung ist ebenfalls im Softwarebereich zu erkennen. Die NASA kehrt zunehmend von speziellen Eigenentwicklungen ab und setzt stattdessen *Unreal Engine*, das neben *Unity* für Entwicklung von Spielen konzipiert wurde, für die Simulation in VR ein [4, 6]. In dies wurde in Zusammenarbeit mit Unreal Engine eine

detailgetreue Umgebung von der ISS implementiert mit dem Ziel, die Astronauten künftig hiermit auf EVAs vorbereiten zu können [6].

Um das vollständige Eintauchen in die VR-Umgebung zu ermöglichen, ist in der Simulation eine detailgetreue Abbildung der Realität notwendig, um damit eine Immersive Virtual Environment (IVE) [7] zu erschaffen. Das Real-Time Attitude Control and On-Orbit Navigation Laboratory (RACOON-Lab) am Lehrstuhl für Raumfahrttechnik (LRT) an der Technischen Universität München (TUM) hat 2016 ein Programm entwickelt, das das Rendezvousmanöver zweier Satelliten im Erdorbit simuliert. Dabei wird mit Hilfe des Visualisierungsmoduls eine virtuelle Abbildung des ganzen Geschehens im Orbit ermöglicht [8]. Eine vorherige Arbeit in diesem Themengebiet erweiterte die Simulation bereits um eine VR-Schnittstelle und es wurde dazu eine erste graphische Nutzerschnittstelle (GUI) erstellt [9]. Für die Fertigstellung des VR-Systems, das völlig unabhängig von den Eingabegeräten der Simulation agieren kann, wird im Zuge der Bachelorarbeit eine Steuerungskomponente hinzugefügt und die Schnittstelle sowie GUI in Bezug auf die gesteigerten Anforderungen erneuert.

Die Entwicklung der Eingabelogik wird in **Kapitel 4** erklärt und die anschließende Implementierung erfolgt im **Kapitel 5**. Die Verifizierung der gesamten Schnittstelle und der erneuerten GUI erfolgt nach den im **Kapitel 3** aufgestellten Anforderungen später in **Kapitel 6** mit einer anschließenden Erörterung der Ergebnisse in **Kapitel 7**. Nach der Zusammenfassung der Arbeit im darauffolgenden **Kapitel 8** gibt **Kapitel 9** eine Auskunft darüber, wie das gesamte VR-System weiterentwickelt werden kann, sowie die möglichen Einsatzgebiete von VR und die der Schwestertechnologie Augmented Reality (AR) in der Zukunft der Raumfahrt sind.

1.2 Stand der Technik

In diesem Abschnitt werden die aktuell in der Raumfahrt standardisierten Steuerungseingabegeräte in Bezug auf die Lagekontrolle von Raumfahrzeugen und die bereits vorhandenen Anwendungen in VR und Simulationsprogramme vorgestellt.

1.2.1 Steuerungseingabegeräte in der Raumfahrt

In der heutigen Zeit ist die Raumfahrttechnik soweit automatisiert, dass manuelle Eingaben von Menschen, sei es am Boden im MCC oder am Bord eines Raumschiffes, in den meisten Fällen nicht mehr notwendig sind. Allerdings werden die kritischen Phasen wie zum Beispiel das finale Rendezvous zweier Raumschiffe immer noch per Hand gesteuert. Für das Andockmanöver von unbemannten Versorgungsraumfahrzeugen, wie das in Abbildung 1–2 dargestelltes Dragon V1 des Herstellers SpaceX an die ISS, ist das Space Station Remote Manipulator System (SSRMS) im Einsatz. Das SSRMS, auch Canadarm2 genannt, ist ein von der Canadian Space Agency (CSA) in Auftrag gegebener Roboterarm, der 2001 in die ISS integriert worden war [10]. Die maximale Länge bei der vollständigen Ausstreckung beträgt 17 m und sie wird für präzise Nahbereichsoperationen eingesetzt [11].

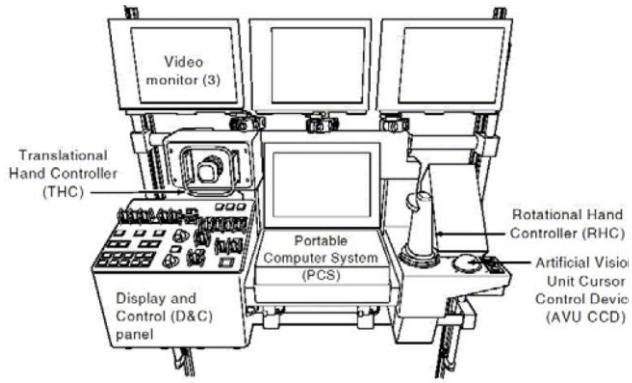

Abb. 1–1: Aufbau der Robotics Workstation [11]

Abb. 1–2: SpaceX Dragon am SSRMS [12]

Die Steuerung des SSRMS erfolgt entweder von der modularen Workstation des US-Amerikanischen Forschungsmoduls der ISS oder für eine bessere Rundumsicht direkt aus dem in Abbildung 1–3 dargestellten Kuppelmodul. Die Steuerungseinrichtung ist wie in der schematischen Abbildung 1–1 aufgebaut. Die Haupteingabegeräte bestehen aus der Kombination zweier Eingabegeräte. Wie schon bei der Dockingvorrichtung am Bord des Space-Shuttles ist der rechteckige Controller auf der linken Seite für die translatorische Bewegung des Roboterarms zuständig. Die rotatorische Eingabe übernimmt der Joystick auf der rechten Seite. Unterstützt wird der Nutzer zusätzlich von verschiedenen Videoaufnahmen und Telemetriedaten auf den drei Video Monitoren, die zusammen mit dem Portable Computer System verbunden sind. Die Interaktion mit dem Computer erfolgt mit Hilfe eines sphärischen Cursors auf der rechten Seite des Steuerungspultes, wobei die zentrale Verwaltung der Systeme durch die Kontrolleinheit auf der linken Seite geschieht.

Die Robotics Workstation ist ein repräsentatives Beispiel für die Flugregelung eines Raumfahrzeugs im Weltall. Sowohl das Orbital Maneuvering des ausgedienten Space Shuttle der NASA, als auch das der aktiven Sojus Kapsel basieren auf dieses

Zwei-Controller-Prinzip. Damit stellt das beschriebene Steuerungssystem einen allgemeingültigen Standard für das Manövrieren von Raumfahrzeuge im Orbit dar.

Abb. 1–3: Robotics Workstation des SSRMS im Kuppelmodul [13]

1.2.2 VR im Bereich Raumfahrttechnik

Die virtuelle Realität ist schon seit geraumer Zeit in der Raumfahrt kein Fremdwort mehr und wird schwerpunktmäßig in Trainings von Astronauten eingesetzt.

Die VR-Technologie ist heutzutage im Einsatz an der ISS und wird für das Training und Auffrischung bereits absolvierter Trainingseinheiten in Bezug auf EVAs verwendet. Ein Beispiel stellt die korrekte Verwendung des Simplified Aid for EVA Rescue (SAFER) System dar. Es verhindert das unkontrollierte Abdriften während eines EVAs, falls die Sicherungsleine reißen sollte, und ermöglicht den Astronauten, wieder an das Raumschiff zu gelangen [14]. Das SAFER wurde zwar noch nie eingesetzt, aber nichtdestotrotz ist es notwendig, dass die Astronauten für den Notfall stets über die Bedienung dieses Systems Bescheid wissen [15]. Für die Anwendung im Weltraum wurde von der NASA eine Eigenentwicklung in Auftrag gegeben und keine auf dem Markt vorhandenen Geräte eingesetzt, da die Zertifizierung eines solchen Geräts für den Einsatz in Raumfahrzeugen einen langwierigen Prozess darstellt. Das in der Abbildung 1–4 zu sehende System ist das Resultat der Entwicklung und ist der Virtual Reality Trainer (VRT). Es ist eine mit VR-Optik ausgestattete Maske, bei der die graphische Darstellung mit Hilfe eines auf der ISS bereits vorhandenen Laptops mit der für diesen Einsatzzweck konzipierten Software erfolgt. Für den Betrieb muss der Laptop lediglich auf die VR-Maske aufgesetzt werden [5]. Die Interaktion mit dem VR-System erfolgt mit Hilfe eines rechteckigen Geräts, auf dem zwei Joysticks befestigt sind.

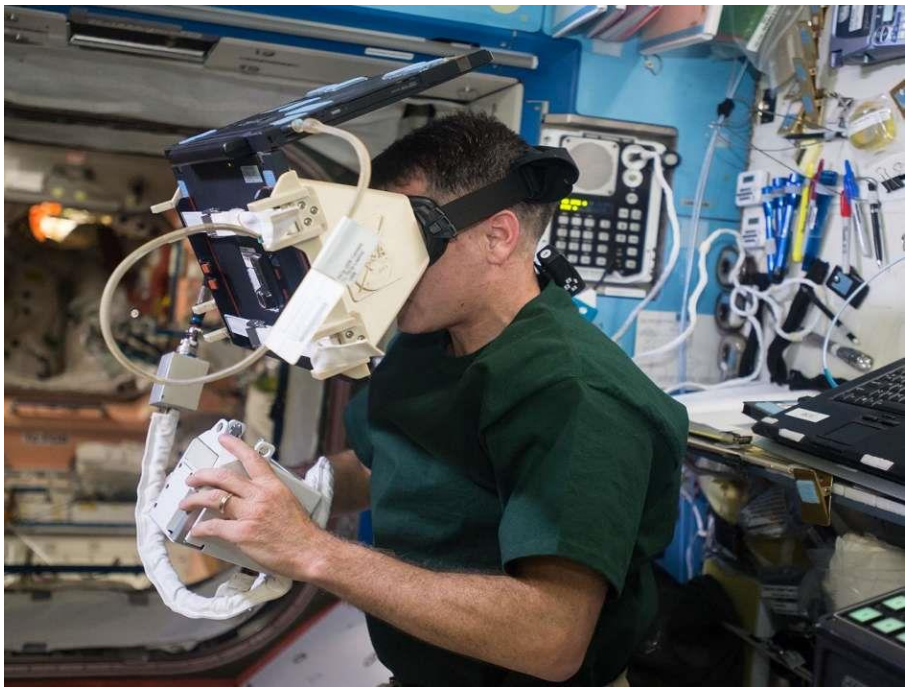

Abb. 1–4: VRT auf der ISS [4]

Das VRT gehört seit dem Sommer 2013 zur Standardprozedur des Trainings für die US-amerikanischen EVA Einsätze [5] und wurde im April 2018 durch ein handlicheres Modell der Firma Oculus ersetzt [4]. Für das Durchlaufen des Trainingsprogramms werden die Bilddaten auf einem Laptop berechnet und anschließend an die neue VR-Brille gesendet. Diese Trennung zwischen Datenverarbeitung und Visualisierung verbesserte die Ergonomie des gesamten Systems.

Parallel zu der Weiterentwicklung und Verbesserung der VR-Technologie wird die Möglichkeit in Betracht gezogen, sie für die Steuerung von On-Orbit-Servicing-Operationen (OOS) einzusetzen.

Das DLR erforscht mit dem Projekt „Virtual Reality for On Orbit Servicing“ (VR-OOS) seit Anfang 2010 die Anwendung von VR für eine teleoperierte Weltraummission [16]. Auf Basis der Anforderungen bei der Wartung und Instandsetzung von ausgedienten Satelliten wurde ein Nutzerinteraktionssystem mit haptischem Feedback entwickelt. Das System besteht aus drei verschiedenen Teilkomponenten. Die Visualisierung der IVE wird mittels eines Head Mounted Display (HMD) mit Trackingfunktion und einer realitätsnahen Simulation erreicht [7]. Die Eingabe erfolgt mit Hilfe des ebenfalls von der DLR entwickelten bimanuellem Interface HUG [7]. Dieses besteht aus zwei leichten Roboterarmen und besitzt eine ähnliche Beweglichkeit wie die menschlichen Arme [17]. Die Roboterarme sind dazu mit starken Elektromotoren an den Gelenken ausgestattet und somit ist es möglich, ein haptisches Feedback, das der Nutzer in der Realität bspw. bei der Kollision mit einem anderen Objekt erhalten würde, zu simulieren. Das HUG ist darüber hinaus mit verschiedenen Sensoren und Aktoren ausgestattet, die eine genaue Abbildung der menschlichen Arme ermöglichen und eine Vielzahl an Feedbacks bieten.

Abb. 1–5: DLR Robotersystem SpaceJustin [7]

Abb. 1–6: Bimanual Haptic Interface HUG [7]

Neben der reinen Simulation ist es möglich, das in Abbildung 1-5 abgebildete Robotersystem SpaceJustin anzusteuern [7]. Der humanoide Roboter wurde für den Einsatz von teleoperativen OOS Missionen konzipiert und bietet in Kombination mit dem in Abbildung 1-6 abgebildeten HUG Interface eine experimentelle Umgebung für die Anwendung und Verifikation der Methoden der Telepräsenz [18].

1.2.3 RACOON-Laboratory

Das RACOON-Lab am LRT der TUM ist ein End-to-End System für die Forschung und Entwicklung von neuen Methoden der Nahbereichsoperation, für die Bewertung und Evaluierung bereits vorhandener Technologien [8]. Der Fokus des Forschungsprojekts liegt auf dem Bereich der Teleoperation und Telerobotik für OOS-Anwendungen. Die gesamte Anwendung setzt sich aus insgesamt vier Teilsystemen zusammen, die miteinander durch einen gemeinsamen Datenserver kommunizieren. Die **Starrkörpersimulation** bildet den Kern des ganzen Systems und übermittelt die berechneten Daten an die Visualisierungskomponenten. Dabei ist es möglich, einzelne Keplerelemente des Orbits und die relativen als auch die globalen Positions- und Geschwindigkeitsparametern mit Hilfe einer graphisch dargestellten **Simulationssteuerung** zu verändern.

Für die Visualisierung sind zwei verschiedene Möglichkeiten implementiert. Die in Abbildung 1–7 zu sehende **Hardware-in-the-Loop** (HIL) Anlage ist eine Verkleinerung des gesamten Weltraumszenarios (Maßstab 1:20) [8]. Die ausführliche Beschreibung der Hardware Anlage ist unter [8] zu finden.

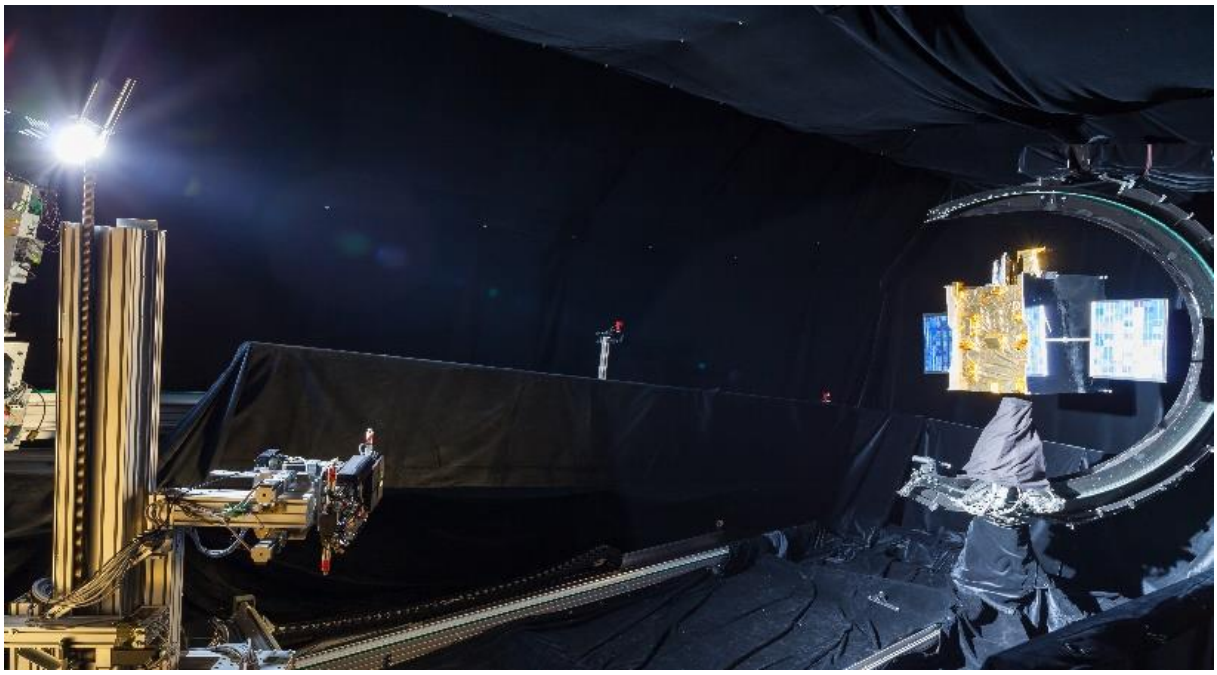

Abb. 1–7: RACOON-Lab HIL-Anlage

In der **Softwarevisualisierung** wird aus den Daten mittels der Multimediaprogrammschnittstelle DirectX 11 das betrachtete Szenario gerendert und im virtuellen Orbit dargestellt. Die Modelle inklusive der verwendeten Materialien der Raumfahrzeuge lassen sich beliebig und je nach Anforderung als CAD-Modelle aus dem Programm CATIA V5 integrieren. Insgesamt verfolgt die Softwarevisualisierung drei Kernpunkte:

1. Erstellung von vielfältigen graphischen Überwachungsmöglichkeiten,
2. Visuelles Feedback als Alternative zu Kamerabildern der HIL Anlage,
3. Einfache Visualisierung bei der Planung von OOS-Missionen.

Zwar sind die mit Hilfe der Kamera an der HIL aufgenommenen Bilder immer noch realistischer als die computergenerierten, dennoch ist die gerenderte Visualisierung

eine hinreichende und mit geringerem Aufwand realisierbare Alternative zu der Visualisierung durch die HIL-Anlage [8]. Um auch die Lichtverhältnisse in der realen Umgebung der OOS-Mission in der Erdumlaufbahn abzubilden, lassen sich die Einflüsse sowohl durch das Sonnenlicht, als auch durch das Erdalbedo simulieren.

Abb. 1–8: RACOON-Lab Software Visualisierung

Zum Hauptanwendungsfall der RACOON-Lab Simulation gehört unter anderem auch die Durchführung von Nutzerstudien [8]. Als Hauptaufgabe sollen die Probanden das Rendezvous- und Docking-Verfahren nach dem innerhalb von RACOON standardisiertem RACOON-Reference-Scenario (RRS), das in Kapitel 1.3.1 ausführlich erklärt wird, ausführen. Hierbei werden sie mit Hilfe von unterschiedlichen Hilfssystemen mit Informationen versorgt, die das Manöver erleichtern sollen. Für die Steuerung des anfliegenden Satelliten wird ein Drei-Freiheitsgradcontroller verwendet. Als eine Art Backupsystem ist es ebenfalls möglich, die Steuerung mit der Computertastatur auszuführen. Dafür wurden die einzelnen Translationen und Rotationen in verschiedenen Tasten untergebracht. Die ausführliche Veranschaulichung der Joysticksteuerung und die Tastenbelegung ist in [19, 20] zu finden. Das Ziel der Nutzerstudie ist die Evaluierung neuentwickelter Hilfssysteme. Im Rahmen einer Dissertation wurde mit Hilfe des Simulationsaufbaus das optische Hilfssystem „ThirdEye“ bewertet [21]. Ein andere durchgeführte Nutzerstudie war die Auswertung eines akustischen Telepräsenzsystems, das die multimodale RACOON Simulation, neben dem optischen Feedback, um ein Auditives ergänzen soll [22].

Die VR-Fähigkeit der RACOON-Simulation wurde im Rahmen von [9] implementiert. Es wurde eine Verbindung zu dem am LRT verfügbaren *HTC VIVE* errichtet und es ist möglich, die Simulation mit Hilfe des VR-Equipments zu betrachten und sich perspektivisch durch den Weltraum zu bewegen. Aufgrund dessen, dass die Ansteuerung der VR-Hardware mittels Gamesengine *Unity* geschieht, ist für den Datenaustausch eine Schnittstelle zwischen der Simulation und der VR-Anbindung notwendig.

Die Anforderungen an die Schnittstelle sind primär darauf ausgelegt, die Latenz des VR-Equipments bezogen auf die Weltraumsimulation so gering wie möglich zu halten, damit die Immersion, die Wahrnehmung sich in einer virtuellen Umgebung zu befinden, nicht gebrochen wird [9]. Für die Anbindung wurden verschiedene Techniken der Interprozesskommunikation (IPC) miteinander verglichen und nach der Übertragungsgeschwindigkeit bezogen auf die Dateigröße, der dafür notwendige Implementierungsaufwand und die Duplexfähigkeit der Datenübertragung eine Auswahl getroffen [9]. Die Entscheidung fiel auf eine Kombination aus Named Pipes und Shared Memory für die zeitkritischen Informationen [9], wobei der Performanceunterschied der beiden Techniken sich erst bei größeren Datenpaketen bemerkbar macht. Daher wurde zugunsten des Implementierungsaufwands für kleinere Datenmenge Named Pipes und für größere Datenmenge Shared Memory, die eine erheblich komplexere Implementierung erfordert, ausgewählt. Zeitlich unkritische Daten werden einfachheitshalber als Text-Datei ausgegeben und im Laufwerk gespeichert. Sie werden bei Bedarf vom Empfänger ausgelesen und weiterverarbeitet. Zusätzlich zu der Datenübertragung ist es für die VR wichtig, unterschiedliche Bilder für die einzelnen Augen bereitzustellen. Das bedeutet, dass die Monoansicht in der Simulation in eine Stereovariante mit einem Perspektivenversatz vom Abstand zwischen den beiden Augen umgewandelt werden muss. Dafür wurde innerhalb eines Zeitschritts des Renderprozesses mit einem zweiten Rendering um den vorher genannten Versatz erweitert [9]. Der gesamte Ablauf der VR-Anwendung ist in Abbildung 1–10 dargestellt.

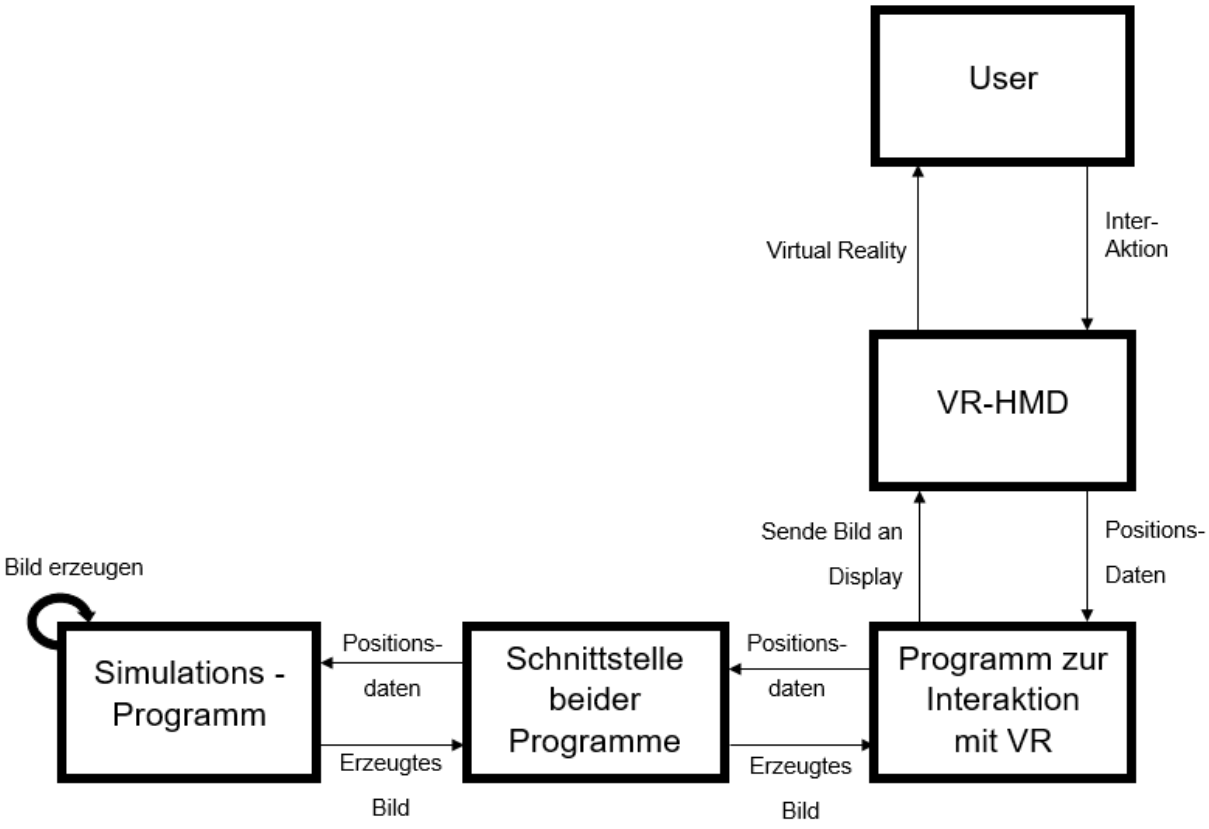

Abb. 1–9: Prinzipschema und Ablauf der VR-Schnittstelle [9]

Die Steuerung der VR-Perspektive ist in das UI des linken Motion-Controllers der *HTC VIVE* untergebrach. Mit seiner Hilfe ist es möglich, die Kameraperspektive entweder durch das Teleportationsmenü an jeweils fünf vorher festgelegten Positionen in der Nähe des Chaser- bzw. Targetsatelliten zu springen, oder sich translatorisch entlang der orbitalen Bahnrichtungen zu bewegen.

Für zusätzliche visuelle Hilfe sind im UI des rechten Motion-Controllers insgesamt vier Funktionen implementiert, wobei sich auf dem Auswahlmenü vier weitere belegbare Feldern für Einbindung zusätzlicher Funktionen befinden. Für die Verbesserung der Visualisierung in VR wurde eine Funktion zur Optimierung der Beleuchtung integriert. Eine weitere Funktion stellt das Pausieren der Simulation dar. Mit ihr ist es möglich, die Simulation direkt aus dem VR-Interface anzuhalten, um bspw. eine bestimmte Situation genauer ansehen zu können. Zusätzlich zu den Grundfunktionen der Simulation beinhaltet die Benutzeroberfläche optischen Hilfssysteme wie die Darstellung der Telemetriedaten durch ein graphisches Head-Up-Display (HUD) in Bezug auf den Target. Inspiriert wurde die graphische Anzeige durch das im Rahmen von ThirdEye entwickelte HUD [9]. Die Darstellung der relativen Positions- und Orientierungsinformationen ist auch in Zahlen möglich und stellt dem Benutzer, neben der Positionsgraphik in der HUD, eine weitere Methode zur Abschätzung der relativen Positions- und Geschwindigkeitsdaten der beiden Satelliten zur Verfügung.

Eine Steuerungsmöglichkeit für die Manipulation der Satellitenposition ist in der VR-UI nicht vorhanden. Sie erfolgt entweder mit Hilfe eines weiteren Controllers oder durch die Computertastatur.

1.3 Aufgabenstellung

Bevor mit der Auseinandersetzung mit dem eigentlichen Thema begonnen werden kann, müssen die zu erfüllenden Aufgaben und Ziele konkret definiert werden. Dazu ist es hilfreich, sich auf ein konkretes Szenario zu beziehen und die Ziele darauf basierend festzulegen.

1.3.1 Beschreibung des Missionsszenarios

Für die Entwicklung einer Steuerungskomponente ist es notwendig, die dafür vorgesehenen Operationen im Weltraum zu definieren. Dazu bezieht sich die Arbeit auf das RACOON Reference Scenario (RRS), das einen allgemeinen Standard im Rahmen der Missionsdurchführung der RACOON-Simulation bietet.

In [23] stehen sich ein Chasersatellit und ein Target- bzw. Zielsatellit gegenüber. Der Chaser, über der der Operator am Boden via Teleoperation die komplette Kontrolle in der Translation und Rotation verfügt, hat die Aufgabe, sich an einen anderen Satelliten aus einer beliebigen Richtung in Umlaufbahn um die Erde anzudocken. Das Target besitzt, je nach den ausgewählten Satelliten, eine Eigenrotation und verfügt im Gegensatz zu zum Chaser über keinerlei Steuerungsmöglichkeiten.

Das Vorhaben lässt sich in ein Flyaround und den eigentlichen Anflug unterteilen.

Die Inspizierung dient der Entscheidung für eine Anflugmethode je nach dem momentanen Zustand des Zielobjekts bzw. den relativen Rotationsgeschwindigkeiten des Zielobjekts gegenüber zum Chaser. Für dieses Vorhaben werden insgesamt zwei Flyaround aus 20 Metern und 8 Metern Entfernung benötigt. Das Flyaround aus 20 Metern gestaltet sich folgendermaßen: Der Chaser führt zunächst eine Kreisbahn einmal komplett um das Target in der Bahnebene aus. Danach folgt das gleiche Manöver, diesmal in der Orthogonalen zu der Bahnebene, wieder zum Startpunkt zurück. Mit diesem Flyaround werden die Bewegungen des Zielobjektes, die strukturelle Verfassung, die Trägheit des Chasersatelliten bzgl. Steuerungseingaben und, wenn gegeben, die „Keep Out Zones“ festgestellt. Darüber hinaus werden die Stellen, an denen der Chaser andocken kann, identifiziert. Als zweiter Schritt erfolgt eine Annäherung auf acht Meter. Dabei wird die Oberflächenbeschaffenheit des Satelliten auf strukturelle Schäden inspiziert, der Andockpunkt aus den verschiedenen vorher festgestellten Möglichkeiten ausgewählt und die Trägheit des Chaser abgeschätzt. Nach dem Erreichen der Position erfolgt ein weiterer Inspektionsflug nach dem bereits geschilderten Schema. Das Ziel dieses Flyarounds ist die Bestimmung des kinematischen Zustands des Targets. Dazu gehört die Lagebestimmung, die Achsen um der die Eigenrotation passiert, die dazugehörenden Geschwindigkeiten und, falls vorhanden, wie stark die Taumelbewegungen sind. Mit der Kenntnis des Istzustands des Zielsatelliten wird am Ende der Inspizierung aus insgesamt drei Anflugstrategien die Geeignetste ausgewählt. Sie werden in der Tabelle 1–1 aufgelistet und im Folgenden genauer erklärt.

Tab. 1–1: Anflugstrategien des RRS

$\Delta v_{Rotation}$	**Anflugstrategie**	**Manöver**
= 0 °/s	Anflug aus der vertikalen Bahnebene	Final Approach
< 2.5 °/s	Anflug aus der Richtung der Rotationsachse	Move to Rotation Axis Final Approach
< 5 °/s	Vollständig synchronisierter Anflug	Move to Rotation Axis Synchronization Spin Up Final Approach

Die Anflugstrategien bestehen, je nach der relativen Rotationsgeschwindigkeit vom Target zum Chaser, aus bis zu drei Komponenten.

Move to Rotation Axis

In diesem Teil des Anflugs bewegt sich der Chaser von der Stelle, die nach dem letzten Flyaround erreicht wurde, auf die Rotationsachse des Targets zu. Dabei ist zu beachten, dass die Entfernung zum Target konstant acht Meter betragen soll.

Synchronization Spin Up

Nach dem Erreichen der Position auf der Rotationsachse des Targets erfolgt eine Synchronisation des Chaser mit dem Target. Das heißt, man passt die Rotationsgeschwindigkeit des Chasers um die nach „vorne“ zeigende z-Achse so lange an, bis die relative Rotationsgeschwindigkeit null ist und die beiden Satelliten sich, ausgehend vom körperfesten Koordinatensystem (KOSY) einer der beiden Satelliten, im quasistationären Zustand befinden.

Final Approach

Der letzte Schritt im Anflug ist, nach der bereits erfolgten Ausrichtung, die Verringerung der relativen Entfernung zwischen den beiden Satelliten. Ausgehend von einer Entfernung von acht Metern nähert sich der Chaser auf anderthalb Meter. Ab dieser Entfernung würde in der Realität ein mechanischer Greifarm zum Einsatz kommen, um die letzte Strecke zu überwinden. Dieses Manöver wurde im Rahmen der RACOON Weltraumsimulation nicht implementiert und wird folglich in dieser Arbeit vernachlässigt.

In der Abbildung 1–11 werden die möglichen Anflugstrategien nochmal in einer graphischen Darstellung zusammengestellt.

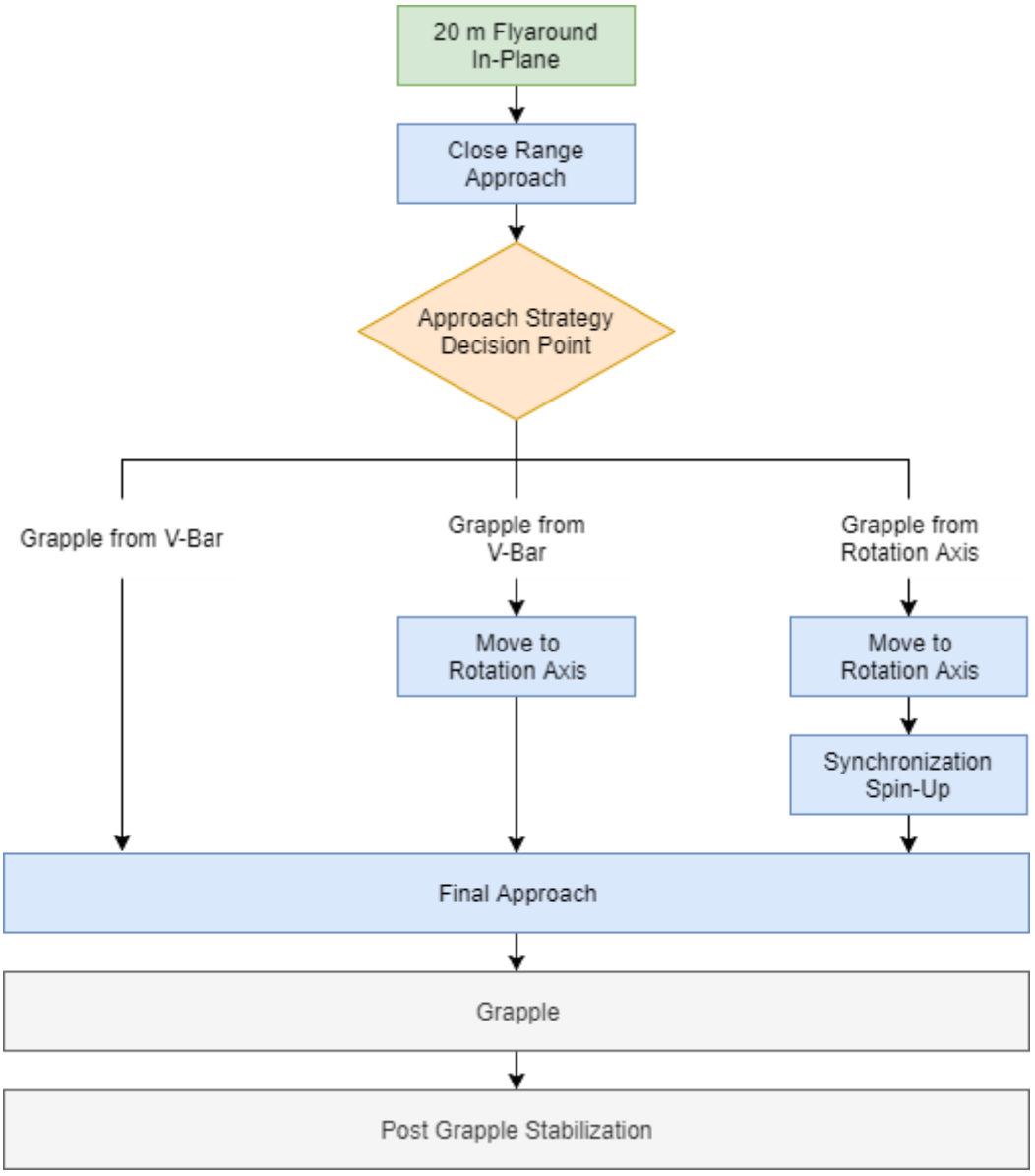

Abb. 1–10: Anflugstrategien nach RACOON Reference Scenario [23]

1.3.2 Ziele der Arbeit

Das übergeordnete Ziel dieser Bachelorarbeit ist die Erweiterung der bereits vorhandenen Schnittstelle zwischen der RACOON-Simulation und der HTC-VIVE um eine Steuerungsmöglichkeit. Dafür wird eine neue Methodik der Satellitensteuerung im Rahmen der Möglichkeiten der virtuellen Realität entwickelt und anschließend implementiert. Dabei gilt es auf die Anforderungen des vorher im Kapitel 1.3.1 vorgestellten Anflugszenarios einzugehen und die Steuerungsmöglichkeit auf ihrer Basis aufzubauen. Um die Steuerung zu unterstützen, ist es notwendig die bereits vorhandene Visualisierungs-/Kamerasteuerungskomponente anzupassen. Dabei sollte der Fokus auf Variabilität und Vielseitigkeit gelegt werden.

Neben der reinen Eingabelogik ist es notwendig dem Piloten mit verschiedenen Hilfssystems zu unterstützen. Die notwendigen Systeme soll im Rahmen der Bachelorarbeit herausgearbeitet und für Ihren Einsatz im Nutzerinterface angepasst werden.

Zusätzlich zum Primärziel der Arbeit wird die vorhandene Schnittstelle und das Interface überarbeitet. Der Schwerpunkt liegt hierbei auf die Stabilität des Gesamtsystems und die Verbesserung der Nutzerergonomie.

1.3.3 Abgrenzung

Die Bachelorarbeit und die daraus folgende Entwicklung und Implementierung einer Steuerungskomponente wird im besonderen Hinblick auf das RACOON Reference Scenario ausgearbeitet. Dabei werden die Steuerungs- und Visualisierungskomponenten auf den Anflug eines Chasersatelliten auf einen Zielsatelliten optimiert. Somit berücksichtigt die Arbeit ausschließlich Anforderungen und Lösungsansätze für das im Kapitel 1.3.1 beschriebene Szenario. Das Thema Wartung und Reparatur wird in dieser Arbeit nicht berücksichtigt. Die Ausführung mit diesem Schwerpunkt wird im Rahmen der Bachelorarbeit [24] von Alexander Hug ausführlich behandelt.

Die Entwicklung des Steuerungssystems wird an einem handelsüblichen HTC-VIVE VR-System ausgeführt, das vom LRT an der TUM zur Verfügung gestellt wird. Deswegen unterliegt die Implementierung einer Rahmenbedingung und es können nur Systeme entwickelt und integriert werden, die die Systemgrenze der VR-Hardware nicht überschreiten. Folglich ist bspw. die Einführung eines Forced Feedback Option an der Steuerung nicht möglich und wird im Rahmen der Arbeit auch nicht erörtert.

Zur Verbesserung der Nutzerfreundlichkeit ist es möglich auch die auditive Wahrnehmung anzusprechen. Im Rahmen einer Studie am RACOON-Lab wurde die Auswirkung eines akustischen Feedbacks untersucht und es wurde festgestellt, dass der Einsatz eines solchen Hilfssystems nur sehr geringe Auswirkung auf die Missionsparametern hat [22]. Aus diesem Grund findet dieser Teil der menschlichen Wahrnehmungsmöglichkeit im Folgenden keine genauere Betrachtung.

2 Grundlagen der virtuellen Realität

Um in **Kapitel 4** die Steuerungskomponente zu entwickeln und sie anschließend in **Kapitel 5** zu implementieren, ist es erforderlich die grundlegende Funktionsweise von VR zu verstehen. Dazu ist es wichtig die menschliche Informations-verarbeitungsprozesse zu analysieren und die technische Umsetzung zur Manipulation der menschlichen Sinne in einer virtuellen Umgebung kennenzulernen.

2.1 Virtuelle Realität und menschliche Informationsverarbeitung

Die Technologie der virtuellen Realität erstellt eine immersive virtuelle Umgebung, bei der die Interaktion mit Hilfe von alltäglichen Bewegungen auf eine realistische Art und Weise stattfindet [25]. Für die Entwicklung einer Mensch-Maschine-Schnittstelle ist es daher sinnvoll, sich mit den Grundlagen der menschlichen Informationsverarbeitung auseinanderzusetzten, weil für die Simulation der perfekten Realität es essentiell ist, die Wahrnehmungsmöglichkeiten abzufangen und sie mit Hilfe der in der virtuellen Umgebung erstellten Stimulationselemente zu verknüpfen. Der Mensch nimmt mit Hilfe von insgesamt neun Sinnen seine Umgebung auf, wovon der visuelle sowie der akustische und der haptische Sinn die wichtigsten Wahrnehmungsmöglichkeiten im Zusammenhang mit der heutigen VR-Technologie sind [26].

Die Abbildung 2–1 stellt ein stark vereinfachtes Modell der menschlichen Informationsverarbeitung dar. Aufgrund dessen, dass die heutige VR-Technologie nur drei der insgesamt neun menschlichen Wahrnehmungsmöglichkeiten anspricht, werden im nachfolgenden die restlichen Sinneseindrücke vernachlässigt.

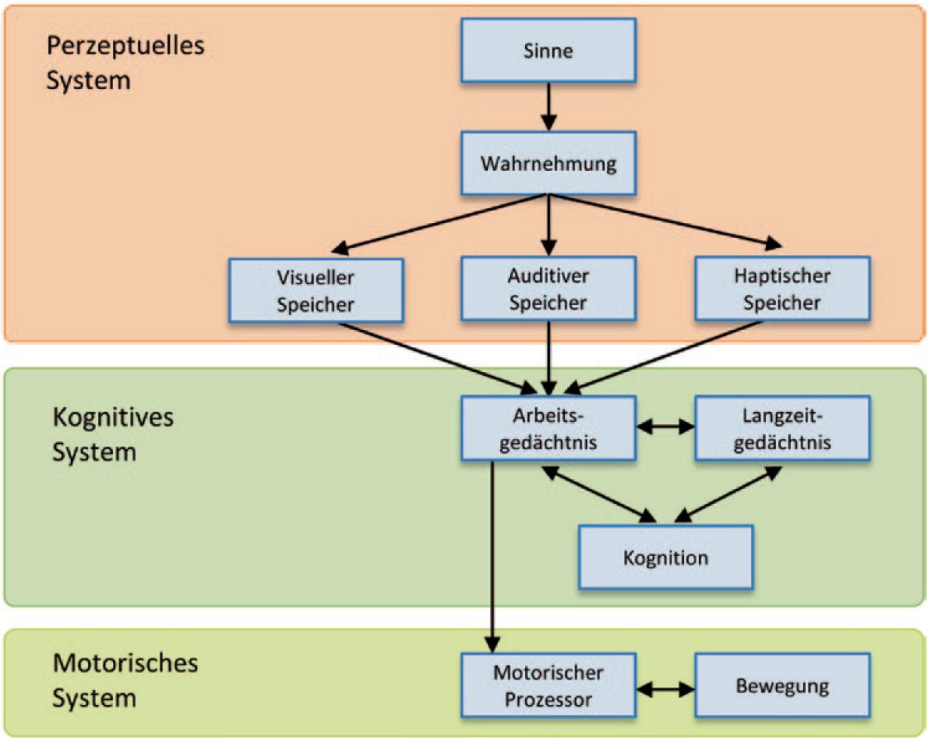

Abb. 2–1: Modell der Menschlichen Informationsverarbeitung [26]

Die menschliche Informationsverarbeitung durchläuft vom Auftreten eines Reizes bis zur Ausgabe einer Reaktion insgesamt drei verschiedenen Verarbeitungssysteme. Am

Anfang der Informationsverarbeitungskette steht das perzeptuelles System, bei dem äußere Reize mit Hilfe der drei an dieser Stelle betrachteten Sinne wahrgenommen werden, im Vordergrund [26]. Das Vorgehen bei der Perzeption ist vergleichbar mit einem technischen System, bei der Sensoren zur Messung von bestimmten Parametern eingesetzt werden. Die Sensoren repräsentieren dabei die menschlichen Sinne. Sie zeichnen verschiedenen Parametern eines äußeren Ereignisses auf, bei der jeder eine festgelegten Funktion besitzt. Für die Weiterverarbeitung der zuvor wahrgenommenen Sinneseindrücke werden sie im Arbeitsgedächtnis des kognitiven Systems zusammengefasst. Hier ist es möglich bei die Verarbeitung und Interpretation der wahrgenommen Reize auf das Langzeitgedächtnis zurückzugreifen und die Ausgabe zu planen. Die Ausgabe erfolgt, nach der Verarbeitung durch das kognitives System, mit Hilfe des motorischen Systems am Ende der Informationsverarbeitungskette, bei der eine Bewegung eingeleitet wird [26].

Für eine detailliertere Einblick in die menschliche Informationsverarbeitung werden im Folgenden die drei angesteuerten Wahrnehmungsmöglichkeiten erläutert.

2.1.1 Visuelle Wahrnehmung

Das visuelle System ist das Teil des menschlichen Nervensystems, der für die Verarbeitung optische Informationen verantwortlich ist [26]. Das Auge ist das Sinnesorgan für die Aufnahme visueller Informationen und verfügt insgesamt über 120 Millionen Fotorezeptoren [26]. Dabei werden sie in die für das skotopisches Sehen verantwortlichen Stäbchen, mit denen lediglich die Helligkeit gemessen werden kann, und in die für das photopische Wahrnehmung zuständigen Zapfen, die die Aufnahme von Farben bei ausreichender Helligkeit ermöglichen, unterteilt. Die Sehzellen sind auf der Netzhaut verteilt, auf der eine sowohl horizontal, als auch vertikal gespiegelte Abbildung der Realität durch die optischen Apparaturen des Auges erstellt wird. Das Auflösungsvermögen, das damit bei idealen Bedingungen erreicht werden kann, liegt zwischen 0,5 bis 1 Winkelminute [26]. Das ist zugleich der limitierende Faktor der heutigen VR-Technologie, da die Bildschirme der VR-Brillen eine recht geringe Entfernung zum Auge haben und bei einer Entfernung von sechs Zentimeter zur Vermeidung vom Fliegengittereffekt 30 bis 60 Pixeln in einer Breite von einem Millimeter benötigt werden. Mit Hilfe der visuellen Wahrnehmung ist es möglich, andere Objekte zu identifizieren und Informationen über deren räumliche Anordnung zu erhalten, bei der in der VR durch die Stereopsis und Tiefenhinweise erreicht wird. In der VR-Technologie ist der visuelle Sinn die wesentlichste Informationsquelle für die Wahrnehmung der virtuellen Welt [26].

2.1.2 Auditive Wahrnehmung

Neben dem visuellen System der Wahrnehmung

Die akustischen Reize werden im Gehör in Form von Schallwellen durch das Trommelfell aufgenommen und mit Hilfe der Gehörknöchelchen an die Schnecke weitergeleitet. Dorthin angelangt werden die mechanischen Bewegungen mit Hilfe von Sinneszellen in der Schnecke zu elektrische Signale verarbeitet und an das Gehirn für die zentrale Verwaltung der Sinneseindrücke weitergeleite [26]. Bei einem Vergleich mit der visuellen Wahrnehmung fällt auf, dass die auditive Wahrnehmung eine geringere räumliche Auflösung besitzt, während auf der anderen Seite die zeitliche

Auflösung viel feiner unterteilt ist. Der Grund dafür liegt in der Art und Weise der Informationsverarbeitung des Gehörs, bei der auf die Laufzeitdifferenzen und Intensitätsunterschiede der empfangenen Schallwellen eingegangen wird [27]. Die räumliche Orientierung wird hierbei mit Hilfe der Laufzeitdifferenz ermittelt, wobei eine zeitliche Diskrepanz von 10 bis 30 Millisekunden für eine eindeutige Unterscheidung ausreichend ist [27]. Die Sensitivität jedoch aufgrund der Informationsverarbeitungsmethode durch die Unterscheidung von Frequenz und Intensität recht gering und eine Differenzierung von zwei verschiedenen Geräuschquellen nur erreicht werden kann, wenn sie sich um mehrere Grade unterscheiden [26].

2.1.3 Haptische Wahrnehmung

Die haptische Wahrnehmung ist ein Oberbegriff für sensorische und motorische Aktivitäten, die die Aufnahme von Objekteigenschaften wie z.B. die Geometrie oder die Oberflächenbeschaffenheit mit der durch den Tastsinn empfundenen Sinneseindrücke. Sie lässt sich in drei Unterkategorien unterteilen: Die taktile Wahrnehmen, die für die Oberflächensensibilität zuständig ist, die kinästhetische Wahrnehmung bzw. die Propriozeption für die Tiefensensibilität und die Temperatur- und Schmerzwahrnehmung [26]. Dabei sind die taktile Wahrnehmungsmöglichkeit und die Tiefensensibilität die beiden haptischen Sinne, die innerhalb der gängigen VR-Technologie stimuliert werden. Das Erstgenannte lässt sich mit Hilfe von Mechanorezeptoren empfangen und wird im Rahmen der VR vorwiegend durch Vibration übertragen. Die Tiefensensibilität beschreibt nicht wie die Oberflächensensibilität die Wahrnehmung, die durch äußere Reize entstanden sind, sondern die wahrgenommenen Reize aus dem Inneren des Körpers. Es wird hierbei eine Unterscheidung zwischen Propriozeption, die alle Empfindungen im Zusammenhang mit der Körperposition umfasst, und der Kinästhesie getroffen. Das Letztgenannte ist eine Teilmenge der Propriozeption und beinhaltet nur Empfindungen bei der aktiven Bewegung [26].

2.2 Stand der Technik des VR-Interaktionsmediums

Die Hardware besitzt neben der softwaremäßigen Umsetzung eine entscheidende Rolle die Immersion der virtuellen Realität zu erhalten. Der aktuelle Markt für VR-Equipment reicht von zusammenfaltbarem Gehäuse aus Pappe für die Wiedergabe von VR-Inhalte mit Hilfe eines Smartphones bis hin zu komplexen Systemen mit permanentem Tracking der Eingabe- und Ausgabegeräte wie z.B. die *Oculus Rift* oder *HTC VIVE*. Das LRT verwendet das letztgenannte System für die Forschung und Entwicklung im Rahmen des CopKa Projektes und des RACOON-Labs.

Das VR-System ist eine Gemeinschaftsentwicklung von *HTC* und *Valve Corp.* [25] und setzt sich insgesamt aus drei verschiedenen Komponenten zusammen.

Das **Trackingsystem** besteht aus zwei auch *Lighthouse* genannte Trackingstationen, das die Position der Eingabegeräte mit Hilfe einer laserbasierenden Technologie bestimmt. Dabei deckt sie eine Fläche von bis zu 4 x 5 Meter ab und es ist möglich innerhalb der gesamten Fläche sich zu bewegen und mit der VR-Umgebung zu interagieren [25]. Dies hat den Vorteil, dass die Lokomotion mit Hilfe natürlicher

Bewegungen geschehen kann und die Teleportation nur in Ausnahmefälle verwendet werden muss. Um die Genauigkeit und Zuverlässigkeit der Positionsbestimmung zu verbessern ist es zusätzlich möglich die Trackingstationen kabelgebunden miteinander für die Synchronisation zu verbinden [28].

Die direkte Eingabe in die VR-Umgebung ist mit Hilfe der beiden **Motion-Controller** möglich. Sie ist in zweifacher Ausführung im Lieferumfang enthalten und ist das Haupteingabegerät innerhalb des Gesamtsystems. Das in der Abbildung 2–2 dargestellte Controller besitzt am Griff mit fünf Eingabefelder einer Vielzahl an Inputmöglichkeiten. Der **Trigger** ist ein Element auf der Rückseite des VIVE-Controllers und lässt sich am besten mit dem Zeigerfinger betätigen. Dabei ist er mit dem Gaspedal eines mit einem Automatikgetriebe ausgestattetes Pkws zu vergleichen. Wie das Gaspedal ist er im Besitz eines variablen Auslenkungswegs und am Ende des Auslenkungswegs einen Knopf, das beim Gaspedal des Pkws die Kick-Down-Funktion auslöst. Auf der Oberseite des Controllers ist ein großes kreisförmiges **Touchpad** verbaut, das mit einer berührungsempfindliche und drucksensitive Oberfläche ausgestattet ist. Damit ist es möglich, wie schon beim Trigger, eine variable Eingabe zu tätigen und sie ist in Kombination mit ihrer Größe und Position auf der Vorderseite des Controllers das Hauptinteraktionsmedium des Gesamtsystems. Des Weiteren sind auf der gleichen Seite noch zwei Knöpfe integriert, die die Aufgabe haben direkt zum allgemeinen Hauptmenü oder zum VR-System-Menü zu schalten. Der Knopf auf der Seite trägt die Bezeichnung **Grip** und ist ein frei belegbarer Knopf, der ergonomisch mit dem Mittel- oder Ringfinger erreichbar ist. Aufgrund dessen, dass die *HTC VIVE* keine hardwaremäßige Unterscheidung zwischen der linken und rechten Controller macht, ist das gesamte Eingabegerät achsensymmetrisch gestaltet. Das ist die Erklärung dafür, dass die Grip-Taste sowohl auf der linken Seite, als auch auf der rechten Seite des Controllers vorhanden ist, aber insgesamt nur mit einer Funktion belegbar ist. Der halofömige Ring am oberen Ende des Controllers ist, neben dem eindeutigen Erkennungsmerkmal der *VIVE*, der entscheidende Bereich für das Positionstracking. Mit Hilfe der in den Einkerbungen vorhandenen Trackingsensoren werden die exakte Position der Objekte erfasst und die Daten im Anschluss an für die Weiterverarbeitung weitergegeben. Für das erzeugen eines haptischen Feedbacks ist unter dem Touchpad einen Vibrationsmotor untergebracht, die sich variabel ansteuern lässt.

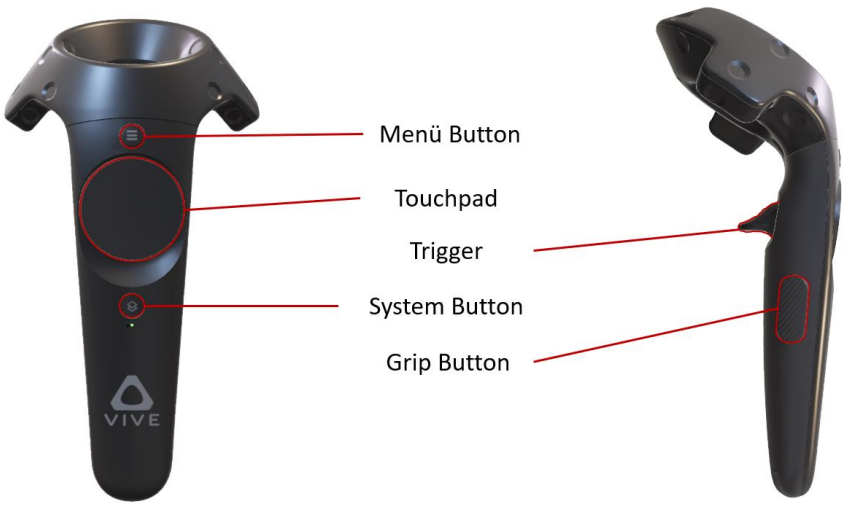

Abb. 2–2: *HTC VIVE* Motion-Controller [29]

Die zentrale Datenverarbeitung erfolgt auf einem extern angeschlossenen PC, der mit Hilfe eines Programms die für die Anwendung notwendige Berechnungen durchführt sowie die Bildinformation generiert. Die Datenverarbeitung der *VIVE* übernimmt die von *Valve Corp.* erstellte Software *SteamVR* und sendet die Ausgabedaten kabelgebunden an das **HMD-Headset** weiter, der daraufhin die notwendigen Daten für die Informationsausgabe an den beiden Controller weiterleitet.

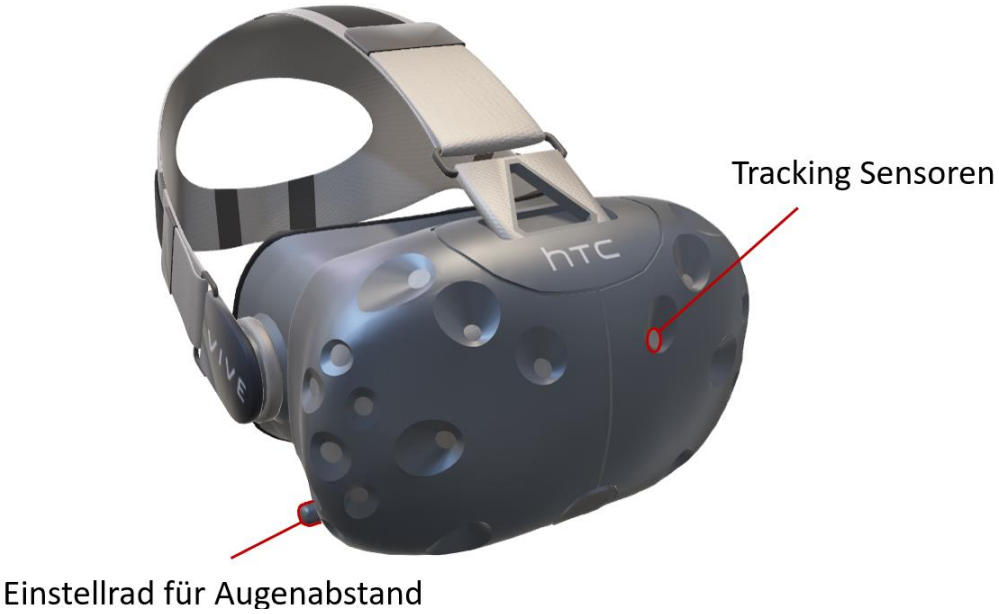

Abb. 2–3: *HTC VIVE* Headset [30]

Die visuelle Ausgabe erfolgt über das in Abbildung 2–3 dargestelltes HMD-Headset mittels zwei hinter der VR-Optik verbauten Displays. Sie haben jeweils eine Auflösung von 1080 x 1200 Pixeln und bieten jeweils ein Field of View (FoV) von 110° [25]. Dies ist zwar immer noch weniger als das FoV eines durchschnittlichen Menschen, ist jedoch für die Anwendung in VR völlig ausreichend. Durch das Drehen des auf der rechten Seite des HMD vorhandene Einstellregler wird der Abstand zwischen den beiden Augen eingestellt. Dies ist erforderlich, weil der Augenabstand von Nutzer zu Nutzer unterschiedlich ist und die Immersion und Präsenz nur fehlerfrei erschafft werden kann, wenn die Augen in der Mitte der kreisrunden VR-Optik sich befinden. Darüber hinaus wird sowohl die Position, als auch die Orientierung des HMDs durch das zuvor beschriebene Trackingsystem verfolgt und in die VR-Interaktion miteingebunden. Das bedeutet, dass die Blickrichtung in die Simulation weitergegeben wird und der Nutzer sie mit Hilfe der Orientierung des Kopfes steuern kann.

3 Anforderungen

Der wichtigsten Punkte bei der Entwicklung und Implementierung einer Nutzerschnittstelle sind die Selbstverständlichkeit und Intuitivität. Idealerweise soll erreicht werden, dass der Benutzer ohne jegliche Hilfe, bspw. durch das Nachschlagen der Dokumentation, im Stande ist, durch die UI mit der Maschine zu interagieren, um die gewünschten Funktionen zu verwenden. Um sich an das Optimum anzunähern, sind eine gewisse Anzahl an Anforderungen zu erfüllen. Sie lassen sich in Bezug auf die Integration einer Steuerungskomponente in zwei verschiedenen Kategorien einteilen. Diese sind die Anforderungen für das Steuerungskonzept und die erweiterten Anforderungen für das Nutzerinterface in VR.

3.1 Anforderungen an das Steuerungskonzept

Für den Entwurf des Steuerungskonzepts sind in der Umgebung der virtuellen Realität besondere Anforderungen erforderlich. Diese werden in der Tabelle 3–1 zusammenfassend aufgelistet und im Anschluss erläutert.

Tab. 3–1: Anforderungen an das Steuerungskonzept

Index	Anforderung	Spezifikation
RFC 1	Nutzerfreundlichkeit	
RFC 2	Präzision	
RFC 3	Portabilität	vollständige Ortsunabhängigkeit

Nutzerfreundlichkeit

Der wichtigste Punkt für die Entwicklung des Steuerungskonzepts ist die Nutzerfreundlichkeit. Dabei steht die Selbstverständlichkeit im Vordergrund und das Konzept soll sich an die natürlichen Bewegungsabläufen der Menschen orientieren. Vor allem bei der Anwendung in VR ist es vorteilhaft, mit dem Hintergrund die Eingabe so intuitiv wie möglich zu gestalten.

Präzision

Für die unmittelbare Steuerung eines Satelliten ist eine hohe Präzision erforderlich. Das bedeutet, dass die Eingabe, wenn überhaupt, nur mit sehr geringer Abweichung vom Sollwert unterscheiden darf. Eine zahlenmäßige Definition wird hierbei nicht festgelegt. Es soll ermöglicht werden, dass die präzise Ansteuerung der einzelnen Koordinatenachsen in der Translation und der Rotation von der Maschine zweifelsfrei erkannt wird. Zusätzlich ist es gefordert, eine einachsige Eingabe von einer zwei- oder dreiachsigen zu unterscheiden und die Fähigkeit eine mehrdimensionale Eingaben zu verarbeiten.

Portabilität

Die VR-Option des RACOON-Labs ist die Erschaffung einer vollständigen virtuellen Umgebung, die über die stationäre bzw. Schreibtischanwendung der VR hinausgeht. Folglich muss die Steuerungskomponente so konzipiert und implementiert werden, dass sie keine Ortsgebundenheit aufweist. Das bedeutet, dass die Eingabe überall im virtuellen Raum unabhängig von der Position und Orientierung des Steuerungsgeräts ermöglicht werden soll.

3.2 Anforderungen an das Steuerungs- und Visualisierungsinterface

Für die Erweiterung der bereits implementierte VR-Schnittstelle ist ebenfalls eine Überarbeitung der bereits implementierten VR-UI notwendig. Das aktuell vorhandene Interaktionsinterface besitzt nur die Kapazität für eine geringfügige Erweiterung und kann den zusätzlichen Funktionsumfang der Steuerungskomponente nicht vollständig aufnehmen. Darüber hinaus werden für die Steuerungskomponente Funktionen und Ansichtsperspektiven benötigt, die über die des aktuellen Implementierungsstandes hinausgehen. In Tabelle 3–2 werden die Anforderungen bezüglich der UI zusammengefasst und anschließend erläutert.

Tab. 3–2: Anforderungen an die gesamte UI

Index	Anforderung	Spezifikation
RUI 1	Nutzerfreundlichkeit	nach Prinzipien von J. Nielsen
RUI 2	Modularität	
RUI 3	Erweiterbarkeit	
RUI 4	Multimodalität	min. zwei
RUI 5	Stabilität	

Nutzerfreundlichkeit

Die Nutzerfreundlichkeit ist das wichtigste Kriterium bei der Gestaltung einer UI. Es ist nämlich die Aufgabe der Entwickler, die Schnittstelle an den Benutzer anzupassen und nicht die Aufgabe der Benutzer, sich an die technischen Konventionen der Maschinen anzunähern. Es dienen wie in der vorangehenden Bachelorarbeit [9] die Heuristiken für UI-Design von Jakob Nielsen [31] als Orientierung für die Gestaltung der Interaktionsschnittstelle. Für die Entwicklung der UI werden die für die Anwendung relevanten Prinzipien herausgenommen und der Anwendung entsprechend angepasst.

Die **Sichtbarkeit des Systemstatus** ist wichtig für die Kommunikation zwischen Mensch und Maschine und hat den Zweck, die Transparenz aufrechtzuerhalten [32]. Parallel mit dem steigenden Systemüberblick ist es wahrscheinlicher, die richtige Entscheidung zu treffen [32]. Für die Nutzerinteraktion ist es folglich wichtig, dass der Benutzer bei der Interaktion Feedback bekommt und sich im Klaren ist, in welchem Zustand sich das System gerade befindet und ob die gewünschte Steuerungseingabe auch von der Maschine erkannt und ausgeführt wird.

Um die Selbstverständlichkeit des Systems zu erreichen, ist die **Übereinstimmung der Systemfunktionen mit der Realität** eine wichtige Anforderung. Das bedeutet, dass die Nutzer-Maschinen-Interaktion bspw. im Bereich der linguistischen Wiedergabe nicht auf einem Fachjargon aufbauen soll, sondern sich an den zwischenmenschlichen Interaktionsweisen bedienen soll [33]. Besonders in der Umgebung der virtuellen Realität ist es ratsam, die Orientierung der Interaktionsmöglichkeiten an die natürlichen Bewegungsmuster des Menschen anzupassen und sie auch zu nutzen.

Die **Nutzerfreiheit und -einschränkung** ist ein viel diskutiertes Thema. Im Allgemeinen soll die Balance zwischen der Möglichkeit des Nutzers die GUI individuell nach persönlichen Präferenzen anpassen zu können und die systemseitige Einschränkungen der Individualität gefunden werden.

Für eine bessere Übersichtlichkeit ist es notwendig, das Design der GUI an gewisse **Standards** zu knüpfen. Für die gleichen Funktionen sollen keine unterschiedlichen Bezeichnungen oder Darstellungen verwendet werden und der Nutzer soll nicht in einer Schleife der Hinterfragung hineingezogen werden. Diese dienen zum Vorbeugen von Irritation des Nutzers und fördern die Erschaffung einer Einheit innerhalb der UI. Darüber hinaus ist bei der Entwicklung darauf zu achten, das Prinzip der **Minimalismus** einzuhalten und die gesamte Nutzeroberfläche nicht zu überladen.

Innerhalb des Interfaces werden insgesamt zwei Hauptfunktionen, die Steuerung des Satelliten und der Visualisierung, untergebracht. Um eine eindeutige Unterscheidung der beiden Systeme zu gewährleisten, gehört die **Eindeutigkeit** ebenfalls zu den geforderten Anforderungen

Um die Anforderungen zum Erreichen der Nutzerfreundlichkeit zu vervollständigen, ist eine **Fehlertoleranz** des Systems notwendig, da es sowohl durch menschliche, als auch technische Ungenauigkeiten, zu einer unbeabsichtigte Fehleingabe kommen kann. Zur Minimierung dieser Ereignisse muss das System daher eine gewisse Fehlerverzeihlichkeit aufweisen, um sie bei Bedarf zu identifizieren und zu unterdrücken.

Modularität

Eine modulare Gestaltung des Interfaces bedeutet, dass die einzelnen Funktionskomponenten eine in sich geschlossene Einheit bilden. Damit soll ermöglicht werden, einzelne Komponenten unter festdefinierten Rahmen- bzw. Schnittstellenbedingungen untereinander austauschbar zu gestalten. Zusätzlich ist es erwünscht, dass das Basisinterface vollfunktionsfähig bleibt, auch wenn eine Funktionskomponente nicht vorhanden ist oder einen Fehler meldet.

Erweiterbarkeit

Für die weitere Integration von Zusatzfunktionen ist es gefordert, dass das Interface über eine ausreichend große Restkapazität verfügt. Das bedeutet, dass eine Erweiterung ohne großen Aufwand und ohne die Umgestaltung des gesamten Interfaces realisierbar sein muss.

Multimodalität

Multimodalität in der virtuellen Realität bedeutet, dass das System dem Benutzer nicht nur auf eine Art und Weise Feedback gibt, sondern gleichzeitig mehrere Sinne stimuliert werden. Für den Aufbau eines multimodales Feedbacksystems ist es notwendig, mindestens zwei Varianten der menschliche Wahrnehmungsmöglichkeit einzusetzen.

Stabilität

Diese Anforderung bezieht sich darauf, dass die Erweiterungen, die im Rahmen dieser Arbeit in die bereits vorhandenen Programme integriert werden, keinen negativen Einfluss auf die Systemleistung haben darf. In Bezug auf die virtuelle Realität ist es essentiell, weil mit einem steigenden Arbeitsumfang und die daraus folgende Latenz die Immersion gebrochen werden kann. Die Erhaltung der Immersion ist essentiell für die Darstellung der virtuellen Umgebung, da sonst die virtuelle Umgebung für den Nutzer nicht aufgebaut werden kann und die VR-Anwendung ihren Sinn verliert.

4 Konzeptionierung der Steuerungserweiterung

In dem vorherigen Kapitel wurde herausgearbeitet, welche Anforderungen die Steuerungserweiterung erfüllen muss. Im folgenden Abschnitt werden die auf die Rahmenbedingungen zugeschnittenen Inputmethoden und Unterstützungssysteme vorgestellt und erläutert. Es ist zu beachten, dass dies nur die grundlegende Funktionsweise der einzelnen Systeme beinhaltet und die Anwendung und die Beschreibung der Anwendung und Implementierung der Simulation erst in Kapitel 5 stattfindet.

4.1 Vorstellung der Eingabekonzepte

Der Einsatz der VR-Technologie erweitert die herkömmlichen Eingabekonzepte mit vielfältigen neuen Möglichkeiten und einer Vielzahl neuer Eingabemethoden. Um die RFC 4 bezüglich der Portabilität zu gewährleisten, ist das Hinzuziehen von stationären Eingabegeräten wie zum Beispiel einer 3D-Maus, mit der eine äußerst akkurate Eingabe erfolgen kann, nicht möglich. Die Entwicklung des Eingabekonzepts beschränkt sich folglich nur auf die Interaktionshardware der *HTC VIVE*. In der Tabelle 4–1 werden die im Rahmen der Arbeit erörterten Eingabekonzepte aufgelistet.

Tab. 4–1: Übersicht der Steuerungskonzepte

Index	Konzept	Eingabemedium
UIS 1	Analogsteuerung	Physische/virtuelle Tasten
UIS 2	Pseudo-Gestensteuerung	6-Freiheitsgraden Joystick

4.1.1 Eingabe mittels Analogsteuerung

Das erste Konzept befasst sich mit der Möglichkeit, den Satelliten auf einer herkömmliche Art und Weise mit Hilfe von Tasten zu steuern. Dabei interagiert der Benutzer mit dem Computer, indem er die Steuerungsbefehle über ein vorimplementiertes Tastenfeld tätigt. Dabei ist es möglich, entweder einen festgelegten Impuls bei der Betätigung der Taste an den Satelliten weiterzugeben, oder die Eingabe variabel zu gestalten. Die variable Gestaltung bietet bei der Steuerung die Möglichkeit, die Kraftimpulse, die durch jeden Translations- und Rotationsbefehl erzeugt werden, zu unterteilen und ihr den Charakter eines Joysticks zu geben. Die grundsätzliche Idee ist das Transferieren der in der Weltraumsimulation vorhandenen alternativen Tastatursteuerung auf die Nutzerinteraktionshardware der *HTC VIVE*. Mit Hilfe der dreidimensionalen Formgebung und Anordnung des Motion-Controllers und den auf ihm vorhandenen Tasten ist es möglich, von dem quasizweidimensionalen Universum der Analogeingabe abzukommen und die Selbstverständlichkeit zu steigern. Das Touchpad auf dem VIVE-Controller bietet die ideale Hardware für die Umsetzung einer variablen Steuerung. Die kreisförmige Scheibe wird mit Hilfe zweier Koordinatenachsen charakterisiert und bestimmt die Position, an der eine Berührung stattfindet.

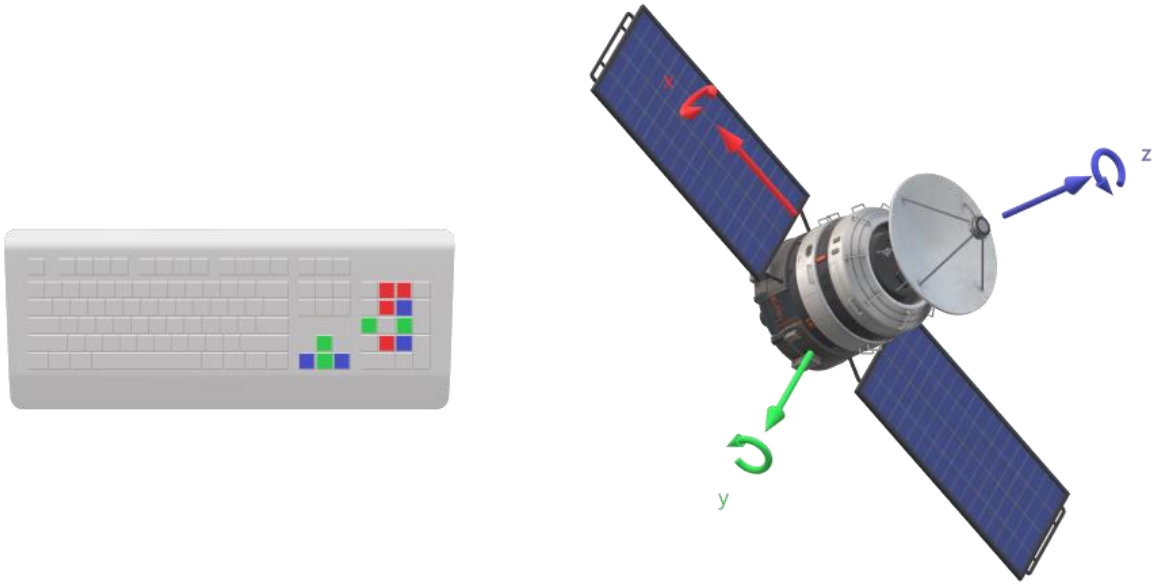

Abb. 4–1: Steuerungsmethode mit Hilfe von Tasten

4.1.2 Eingabe durch Pseudo-Gestensteuerung

Die herkömmlichen Eingabegeräte, wie beispielsweise der Drei-Freifreiheitsgradenjoystick, sind aufgrund der Mechanik lediglich auf zwei Raumachsen und eine Rotationachse beschränkt. Die restlichen Koordinaten- und Rotationsrichtungen werden durch die Betätigung verschiedener Tasten angesteuert. Dadurch ist es nicht möglich, alle sechs Freiheitsgrade eines dreidimensionalen Raums auf eine intuitive Art und Weise anzusteuern. Die *HTC VIVE* Controller verzichtet in diesem Zusammenhang komplett auf eine mechanische Anbindung und die Positionsdaten werden mit Hilfe von zwei *Lighthouses* bestimmt. Somit erschließt sich die Möglichkeit, die Befehlseingabe in den sechs Raumrichtungen und sechs Rotationsrichtungen komplett ohne das Betätigen von Tasten oder die Veränderung der Gesamtkonfiguration des Controllers auszuführen. Die Translation des Satelliten wird dementsprechend mit Hilfe einer Bewegung des Controllers in die gewünschte Richtung und die Rotation mit Hilfe der Drehung des Controllers um die vorher festgelegten Koordinatenachsen eingeleitet. Das Koordinatensystem lässt sich prinzipiell variable im Raum festlegen und bietet somit einen Nullpunkt, der sich je nach Bedarf beliebig an die ergonomischen Bedürfnisse des Operators anpassen lässt. Mit diesem Steuerungskonzept wird die herkömmliche, zweidimensionale Eingabeschnittstelle um eine weitere Dimension erweitert und erlaubt es dem Anwender, natürliche Bewegungsmustern in die Nutzerschnittstelle zu integrieren. Dies fördert die Intuitivität des gesamten Eingabesystems und beugt in diesem Zusammenhang eine Fehleingabe vor. Ferner bietet das System aufgrund seiner Einfachheit und Selbstverständlichkeit auch für einen weniger geübten Operator einen schnellen und zielgerichteten Eingriff in Situationen, für die eine augenblickliche Reaktion verlangt wird. Im Allgemeinen ist die Steuerung mittels Positionsbestimmung eine Abwandlung der Gestensteuerung. Die Eingaben werden in diesem Fall nicht unmittelbar durch die Hand, sondern mit Hilfe eines handgeführten Controllers verrichtet.

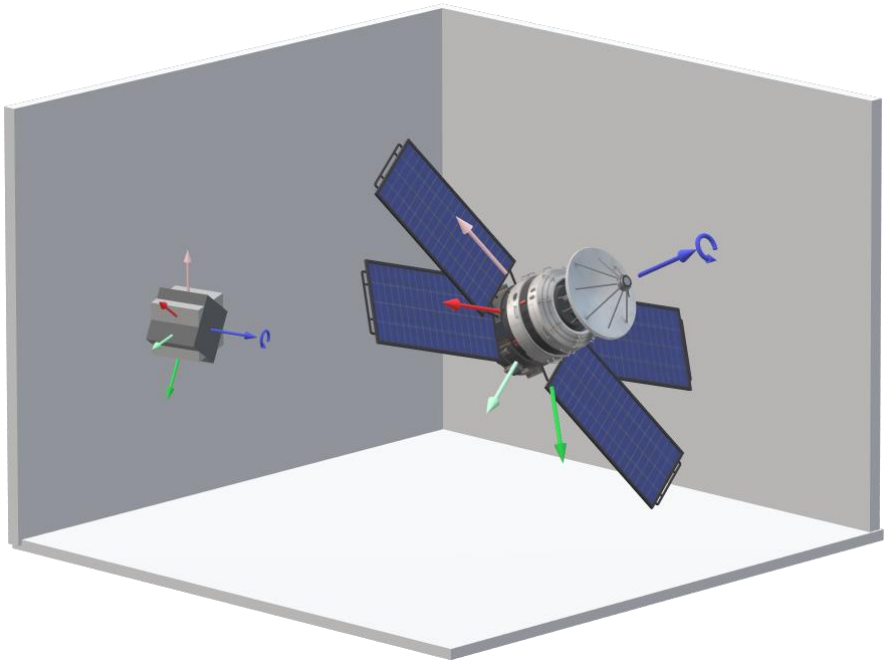

Abb. 4–2: Steuerungsmethode mit Hilfe der Positionsbestimmung

4.2 Vorstellung der Hilfssysteme

Abseits der eigentlichen Steuerungslogik ist es für den Erfolg einer Mission ausschlaggebend, den Operator so gut wie möglich mit relevanten Informationen zu versorgen. Die Herausforderung dabei ist es, die wichtigsten Daten herauszusuchen und sie situationsabhängig an den Piloten weiterzugeben. Damit wird vorgebeugt, den Operator mit unnötigen Komplexitäten zu überfrachten und zu überfordern. Für dieses Vorhaben ist es sinnvoll, möglichst viele Aspekte der multisensorischen Wahrnehmung des Benutzers anzusprechen.

Die zwei nachfolgenden Unterkapitel teilen die Assistenzsysteme in einen optischen und einen haptischen Bereich ein und geben einen Überblick über alle bereits vorhandenen und im Rahmen der Bachelorarbeit entwickelten Ausgabeelemente.

Das akustische Feedback wurde 2016 am RACOON-Lab im Rahmen eines Papers für eine Konferenz des Institute of Electrical and Electronics Engineers (IEEE) untersucht. Das Ergebnis ist, dass es sich nur sehr gering auf den Missionsverlauf auswirkt [22]. Deswegen erfährt die akustische Unterstützung, wie bereits bei der Abgrenzung angekündigt, in diesem Kapitel keine weitere Beachtung und sie wird in die Nutzerschnittstelle nicht integriert.

4.2.1 Optische Unterstützung

Bei der ersten Kategorie der Hilfssysteme wird der visuelle Sinn der Menschen in Anspruch genommen. Er ist im Kontext der heutigen VR-Technologie auch der am meisten angeregte Wahrnehmungsbereich und nimmt somit eine sehr große Bedeutung ein. Die Tabelle 4–2 listet überblicksmäßig alle im Rahmen der Arbeit integrierten optischen Hilfssysteme auf.

Tab. 4–2: Überblick über die optischen Hilfssysteme

Index	Bezeichnung	Anmerkung
OFS 1	Graphisches HUD für Info. -Darstellung	Entwurf von [9]
OFS 2	Schneekugel-Überblicksansicht	
OFS 3	Neutralpunktanzeige	
OFS 4	Basic Deviation Indicator (BDI)	Erweiterung von OFS 3
OFS 5	Advanced Deviation Indicator (ADI)	Erweiterung von OFS 3

Graphisches HUD für Informationsdarstellung

Das graphische Head Up Display (HUD) für die Informationsdarstellung fasst alle bereits aus der Bachelorarbeit von Thomas Landauer vorhandenen graphischen Hilfssysteme zusammen. Das von ihm implementierte HUD orientierte sich an das experimentelle HUD für das RACOON-Lab und es lässt sich in drei Teilbereichen einteilen. In Abbildung 4–3 ist das HUD, wie sie auch im VR-Nutzerinterface implementiert wurde, dargestellt.

Eine ausführliche Erklärung der hierbei verwendeten Koordinatensysteme ist am Anfang des Kapitel 5 dargelegt.

Die zwei Grafiken auf der linken Seite repräsentieren die Position des Chasers gegenüber dem Target. Dabei ist die linke der beiden Graphiken die Projektion auf die x-z-Ebene, die für die Vorstellung die Frontalansicht abbildet, und die rechte auf die x-y-Ebene, die wiederum die Draufsicht von oben repräsentiert. Eine weitere Grafik auf der rechten Seite stellt die relative Rotation der beiden Satelliten in der Outside-In Variante dar. Das bedeutet, dass das Koordinatensystem des Chasers seine Orientierung nicht ändert, während sich der Repräsentant für das Target je nach momentaner Rotation um die z-Achse verschieden schnell um den Ursprung dreht [9].

Neben der graphischen Anzeige werden dem Operator die relative Position und Rotation in absoluten Werten zur Verfügung gestellt.

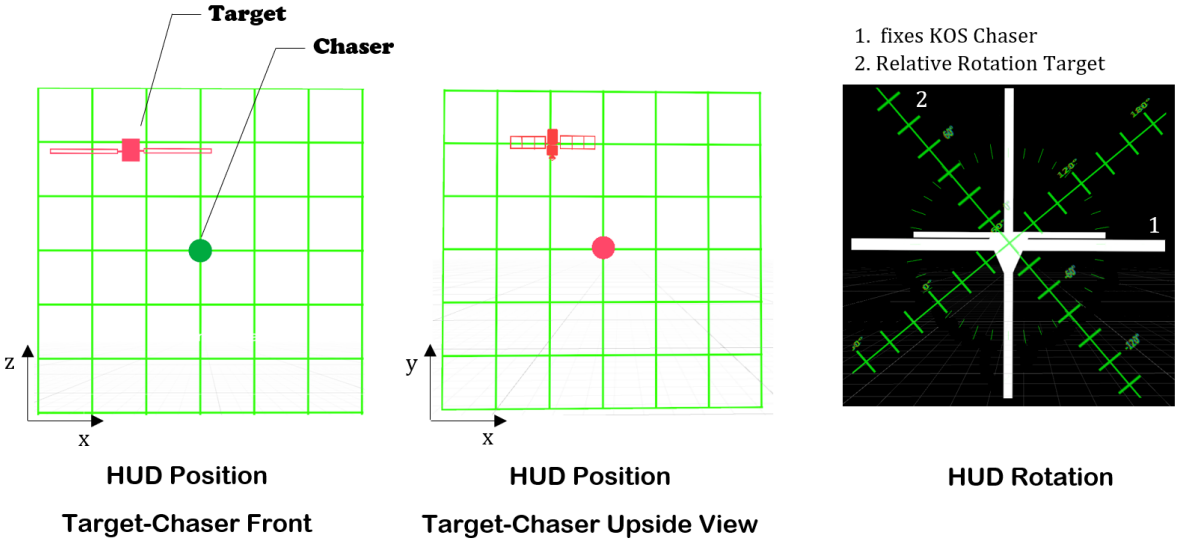

Abb. 4–3: Head Up Display und Telemetriedaten [9]

Schneekugel-Überblicksansicht

Den Überblick über die ganze Situation zu haben/behalten, gehört zu den Sachen, die für die Arbeit, sehr wichtig ist. Dies wird erreicht, indem die Gesamtsituation in einer sphärischen Kugelansicht zusammengefasst wird und dem Nutzer unabhängig von der momentanen Blickrichtung die notwendige Information über die momentane Situation gegeben wird. Dabei werden die Position und Orientierung des Chaser- und Targetsatelliten sowohl räumlich als vereinfachte Abbildungen, als auch absolut mit Zahlenwerten angegeben. Zusammenfassend ist die Schneekugelansicht eine dreidimensionale Variante des zuvor vorgestellten zweidimensionalen HUDs.

Neutralpunktanzeige

Die Neutralpunktanzeige hat die Aufgabe, das Koordinatensystem, auf das die Steuerungseingaben basieren, zu visualisieren. Dabei besitzt sie je nach der ausgewählten Steuerungsmethode unterschiedliche Funktionen. Bei der Eingabevariante mit Hilfe eines 6-Freiheitsgraden Joysticks ist sie für die Orientierung des Nutzers bei der Eingabe zuständig. Der Neutralpunkt ist innerhalb dieser Methode die Stelle im Raum, bei der alle Eingabeparametern null sind. Ihre graphische Darstellung im Raum unterstützt den Operator bei der Eingabe von Steuerungsbefehlen und stellt eine Referenz in Zusammenhang mit dem der Position

und Orientierung des Controllers dar. Für die analoge Steuerung ist hingegen kein Bezugspunkt im Raum notwendig, da hierbei die räumliche Position und Orientierung für die Steuerungskomponente nebensächlich ist. Die Neutralpunktanzeige wird für diese Anwendung abgewandelt und wird im Blickfeld des Benutzers fixiert. Die analogen Eingaben des Benutzers werden in eine Ansicht um, bei der Kommandos ausgehend von einem Neutralpunkt dreidimensional angezeigt werden.

Zusätzlich zu den aktiven Funktionen ist diese Anzeige ein Indikator dafür, dass die Steuerungsapparatur aktiviert und freigegeben ist. Allgemein betrachtet bildet die Neutralpunktanzeige die Basis, bestehend aus der Visualisierung der Koordinatenachsen, für den BDI und ADI.

Basic Deviation Indicator

Das BDI bildet auf Basis der Darstellung der Koordinatenachsen die Abbildung eines Pfeils, der seinen Anfang am Ursprung des Koordinatensystems hat und die Länge und Richtung der tatsächlichen Auslenkung besitzt. Somit zeigt der Pfeil die Richtung und Stärke der Steuerungsauslenkung an. Diese Funktion verbessert das Situationsbewusstsein bei der Manipulation der Chaser-Position und hat die Aufgabe, den Nutzer jederzeit über die erfolgte Steuerung aufzuklären und eine Fehleingabe auf dem schnellsten Weg zu identifizieren. Für die Rotation des Objekts um ihre eigenen Achsen wurden ebenfalls Hinweisanzeigen entwickelt. Sie werden mit Hilfe von Kreislaufsymbole in die jeweiligen Richtungen visuell sichtbar gemacht.

Advanced Deviation Indicator

Ergänzend zu der einfachen Darstellung eines Differenzvektors im BDI, bietet dieses System die erweiterte Darstellung der Auslenkung. Das Koordinatensystem wird in insgesamt zwölf Richtungsebenen und Oktanten eingeteilt. Je nachdem in welche Richtung die Auslenkung erfolgt, werden die angesteuerten Ebenen bzw. die dazugehörigen Oktanten visuell hervorgehoben. Diese Art der Veranschaulichung findet ihren Einsatz nur bei der Translation und signalisiert dem Operator nochmal verstärkt die Auslenkungsrichtung.

4.2.2 Haptisches Feedback

Die Ansteuerung des Tastsinns ist der zweite menschliche Wahrnehmungsaspekt, der für die Eingabe in das VR-System aufgenommen wurde. Dabei stellen sich je nach Funktionsbereich verschiedenste Systemtypen zur Verfügung. Die im Rahmen dieser Bachelorarbeit entwickelten Feedbacksysteme basieren ausschließlich auf Vibrationstechnik. Das heißt, die Rückgabe erfolgt nur über Schwingung des Ausgabegeräts. Die Tabelle 4–3 fasst alle implementierten, haptischen Rückgabeelemente zusammen.

Tab. 4–3: Überblick über die haptischen Unterstützungen

Index	Bezeichnung	Anmerkung
HFS 1	Eingabebestätigung Funktionsaufruf	Gestaltung je nach Fkt.
HFS 2	Deviationsfeedback	Variables Feedback

Eingabebestätigung Funktionsaufruf

Bei der Betätigung der hardwareseitig vorhandenen Tastelemente wird eine Rückgabe in Form einer Vibrationssequenz an den Nutzer weitergegeben. Dabei ist es möglich, die Dauer und Intensität oder die Form der Sequenz individuell festzulegen und sie, je nach Anwendungsfall, eindeutig unterscheidbar zu machen. Die Ausprägung der Ausgabe kann von einem kurzen Klick über eine definierte Abfolge mit unterschiedlicher Intensität bis zu einer kontinuierlichen Vibration sein.

Deviationsfeedback

Das Deviationsfeedback ist eine Ausgabefunktion, die je nach Auslenkung des Steuerungselements eine Rückgabe mit variabler Intensität an dem Operator ansteuert. Dabei wird dem Anwender, optional zusammen mit den beiden optischen Assistenten, signalisiert, in welcher Größenordnung die von ihm angesteuerten Auslenkungen sind. Die Intensität erhöht bzw. verringert sich unterdessen mit zu- oder abnehmender Auslenkung.

4.3 Evaluierung und Auswahl für die Implementierung

Nachdem die möglichen Interaktionskonzepte und die Hilfssysteme vorgestellt worden sind ist es die Aufgabe dieses Kapitels die verschiedenen Konzepte zu bewerten und eine Auswahl zu treffen.

Aus den bereits vorgestellten Eingabekonzepte ist es deutlich zu erkennen, dass die zweite Variante mit dem Positionstracking der Controller die bevorzugtere Variante darstellt. Sie bildet mit der Orientierung an den natürlichen menschlichen Bewegungen und ihre Selbstverständlichkeit eine intuitive Interaktionsplattform. Die erste Variante mit dem analogen Input ist eine zusammenfassend die Transferierung der bereits vorhandene Tastatursteuerung in die Steuerungseingabegeräte in der virtuellen Realität. Sie wirkt verglichen mit Gestensteuerung altmodisch und wenig intuitiv, da sie zwar einen Zugriff auf die dreidimensionalen VR-Umgebung besitzt, sie aber nicht einsetzt und in der zweidimensionalen Hardware verharrt. Allerdings ist die analoge Steuerung bis zum heutigen Tage die Steuerungsmethode, die sich in der Industrie und unter anderem auch in der Raumfahrttechnik durchgesetzt hat. Der große Vorteil dieser Methode ist, dass sie wegen der mechanischen Anbindung und der damit verbundene Eindeutigkeit eine große Zuverlässigkeit mit sich bringt. Die oben beschriebene Gestensteuerung ist ein Produkt aus dem 21. Jahrhundert und besitzt in Ihrer aktuellen Entwicklungsstufe eine noch zu große Abweichung, um damit präzise Eingabekommandos durchführen zu können.

Für die Steuerung des Satelliten wird aus diesem Grund neben der Haupteingabe Methode mittels Gestensteuerung auch eine analoge Eingabemethode implementiert. Dies bietet eine Alternative für präzise Eingaben bei evtl. auftretenden ungenauen Positionstracking und sichert die Steuerungskomponente gegen unbeabsichtigte Fehleingaben nochmals ab. Dazu werden die für den Steuerungseinsatz notwendigen Hilfssysteme implementiert und in einem GUI auf der obersten Ebene zusammengefasst. Sie beinhaltet die OFS 3, 4 und 5 zur Visualisierung des neutralen Punktes bei der Steuerung und der Deviation. Zusätzlich zu den optischen Hilfssystemen werden beide haptischen Systeme implementiert.

Die Steuerung der Visualisierungskomponente und die Verschiebung der Kameraperspektive bedingt keine so hohe Genauigkeitsanforderungen. Somit ist für diese Anwendungsfall völlig ausreichend die Steuerung mittels Gesten als alleinige Eingabemethode festzulegen. Als zusätzliche optische Hilfssysteme werden die OFS 1, 3, 4 und 5 implementiert. Wie schon zuvor bei der Satellitensteuerung beinhaltet die Kamerasteuerung ebenfalls die beiden haptischen Ausgabesysteme.

Die Schneekugelansicht (OFS 3) wird sowohl bei der Steuerung des Satelliten, als auch bei der Steuerung der Visualisierungskomponente außenvor gelassen, weil sie eine Redundanz mit dem graphischen HUD (OFS 1) bildet. Die dreidimensionale Darstellung der Gesamtsituation ist zwar für die grobe Orientierung besser als die zweidimensionale Abbildung, sie ist jedoch für eine genauere Abschätzung der relativen Entfernung zwischen dem Chaser- und Targetsatelliten zu inakkurat, weil die dreidimensionale Ansicht nicht immer eindeutig ist und die optische Wahrnehmung sich von Perspektive zu Perspektive unterscheidet. Erst mit Hilfe der Projektion auf zwei zueinander orthogonalen Ebenen wird gewährleistet, dass die wahrgenommene Position aus allen Blickwinkeln übereinstimmt.

5 Implementierung von RacoonVR

Nachdem in Kapitel 4 die Steuerungsmethode und die Hilfssysteme vorgestellt und mit Hilfe der Evaluierung eine Auswahl getroffen wurde, erfolgt die Detailierung und Implementierung der zuvor ausgewählten Mechanismen. Die einzelnen Bausteine werden miteinander zu einem vollständigen VR-Steuerungssystem zusammengebaut, mit dem es möglich ist, sowohl die Steuerung des Chasersatellitens, als auch die Veränderung der VR-Visualisierung durchzuführen.

5.1 Überblick Koordinatensystem

Bevor mit der Implementierung begonnen werden kann, ist es essentiell, sich über alle vorkommenden Koordinatensysteme in der gesamten RACOON-Simulation bewusst zu werden. Dabei ist die Spezifikation der körperfesten Systeme am manövrierbar Chasersatelliten und dessen genaue Kenntnis bedeutend für die weitere Integration und das Verständnis der ganzen Steuerungsapparatur.

Die Koordinatensysteme des Chasers ist grundsätzlich so definiert, dass die z-Achse nach „vorne“, also in die entgegengesetzte Richtung vom Haupttriebwerk des künstlichen Himmelskörpers, zeigt. Die anderen beiden Achsen der Systeme unterliegen keinen besonderen Festlegungsregeln und wurden bei der Implementierung des RACOON-Labs festgesetzt. Die Abbildungen 5–1 und 5–2 stellen die Anordnung der Koordinatenachsen vom Chasersatelliten dar.

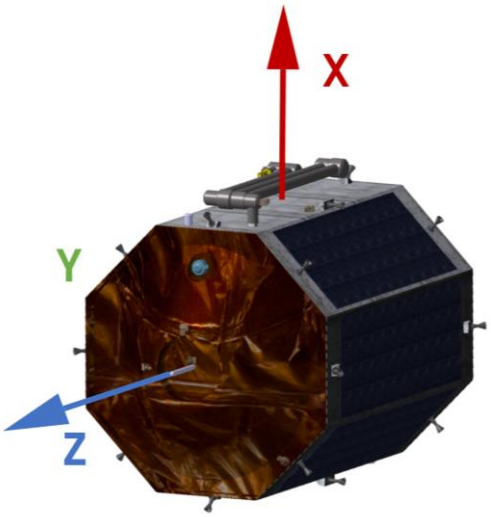

Abb. 5–1: KOSY Chasersatellit (Front) [22]

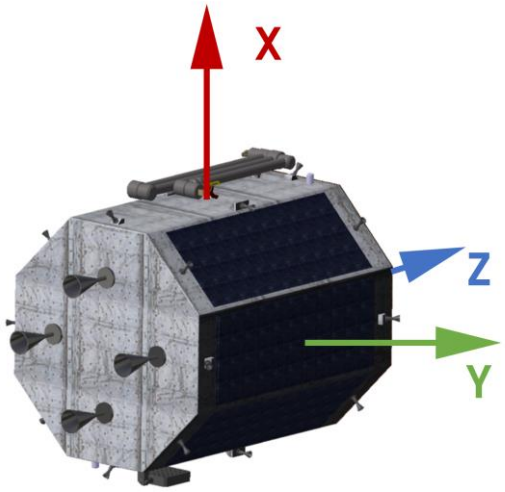

Abb. 5–2: KOSY Chasersatellit (Rear) [22]

5.2 Grundlagen der Software-/Entwicklungsumgebung

Als Basis der Erweiterung dient die im RACOON-System bereits integrierte VR-Plattform. Im Rahmen der Bachelorarbeit von Thomas Landauer wurde diese erstellt. Die Ansteuerung der VR-Brille erfolgt über das Programm *Unity*. Die zentrale Verwaltung der verwendeten Objekte findet man im *Hierarchy*-Fenster auf der linken Seite der Standard-Entwicklungsoberfläche. In diesem Bereich werden alle *GamesObjects* der aktuelle Szene angezeigt. Das bedeutet, dass jedes Element, das innerhalb der Szene verwendet wird, einen entsprechenden Eintrag bekommt und an

dieser Stelle gefunden und ausgewählt werden kann [25]. Für die Bearbeitung der ausgewählten *GamesObjects* ist der *Inspector*-Bereich verantwortlich. Er stellt einen dynamischen Bereich dar, bei der die Eigenschaften und Komponenten der jeweils ausgewählten Spielobjekte aufgelistet sind [25]. In diesem Bereich ist es möglich, die Eigenschaften des *GamesObjects* durch das Hinzufügen oder die Veränderung von Spezifikationsmodulen zu bearbeiten, sowie ihre Funktion mit Hilfe eines Codeprogramms zu erweitern. Diese werden im Zuge der Erweiterung in der Programmiersprache C# verfasst. Die Funktionserweiterung innerhalb der Entwicklungsumgebung von *Unity* ist ebenfalls mit Hilfe der Integration von vorgefertigten Komponenten möglich. Der Unterschied zu den *Scripts* liegt darin, dass die als *Assets* bezeichnenden Komponenten vollständige *GamesObjec*ts sind und durch Drag and Drop in das *Hierarchy*-Fenster hinzugefügt werden können. Zusammengefasst werden alle in der Projektmappe befindlichen Objekte im Projektbereich der *Unity*-Arbeitsumgebung, von der der Zusammenbau der Szene wiederum mit Hilfe von Drag and Drop ausgeht. Für die VR-Erweiterung des RACOON-Labs wird das *Asset SteamVR* benötigt [9]. Dieses von *Valve Corp.* entwickeltes Plugin ist eine bereits vollständig implementierte Schnittstelle zwischen dem *HTC VIVE* System und *Unity* [9]. Das VR-HMD und die beiden Controller werden als *GamesObjects* initialisiert und können, wie die anderen Spielobjekte, durch den *Inspector* und das Hinzufügen von *Scrips* verändert werden. Im Rahmen der Überarbeitung wird die *Unity*-Version von 5.6.03f der ursprünglichen Implementierung auf eine neue Version mit der Versionsnummer 2018.2.9f1 und die *SteamVR* Plugin Version auf die aktuelle Version 1.2.3 aktualisiert.

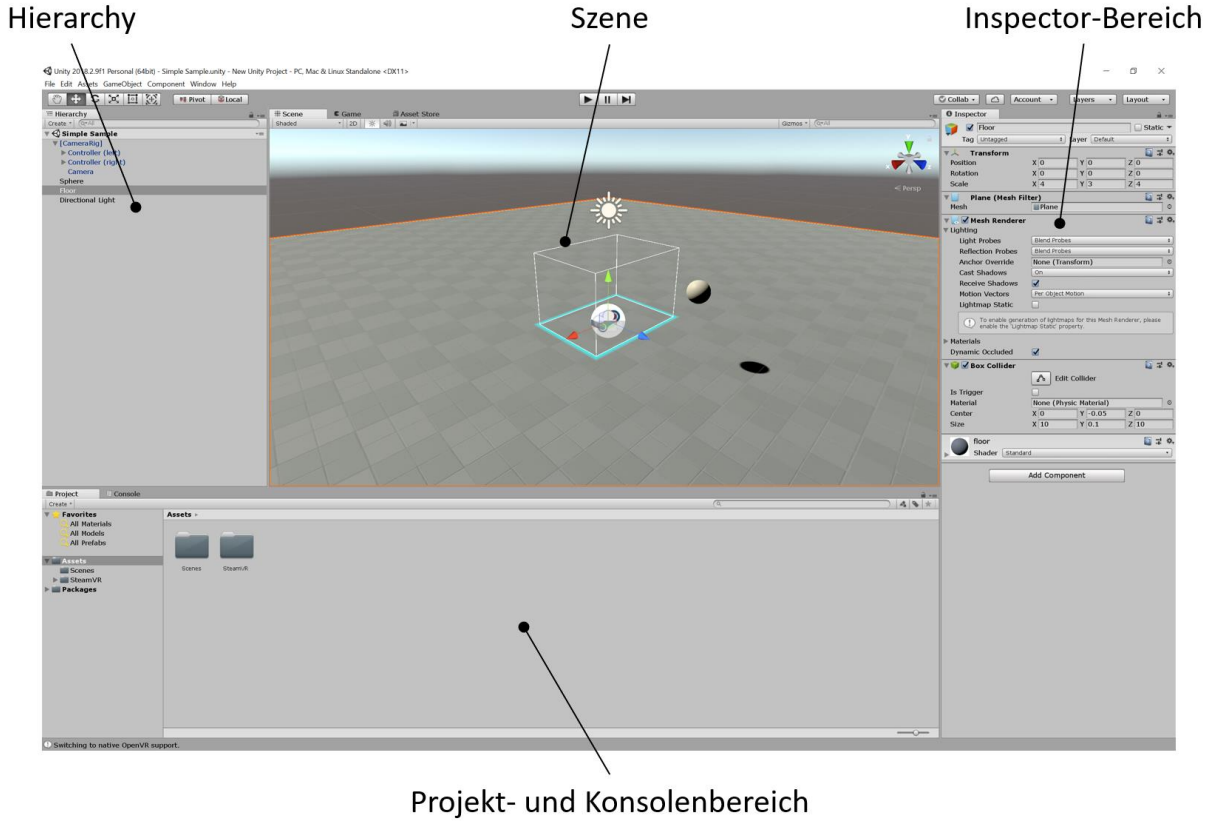

Abb. 5–3: Entwicklungsumgebung von Unity, inspiriert von [9]

5.3 Implementierung des Steuerungsinterfaces

Für die Interaktion des Nutzers mit der Simulation wurde ein Interface für die Steuerung des Satelliten erstellt. Um eine klare Trennung zu der Steuerung der Kameraperspektive und sonstigen Funktionen zu gewährleisten, sind alle Funktionen der Satellitensteuerung im rechten Controller untergebracht. Mit Hilfe der Funktionstrennung soll gewährleistet werden, dass eine Irritation des Nutzers durch die klare Aufgabenverteilung der linken und rechten Hand ausgeschlossen wird. Dazu ermöglicht die Trennung zwischen der Satelliten- und Visualisierungsteuerung eine parallele Bedienung der beiden Interaktionseinheiten.

5.3.1 Interface für Funktionsauswahl und Menüführung

Das Hauptkonzept bei der Implementierung des Steuerungsinterfaces orientiert sich an der Natürlichkeit. Das bedeutet, dass die Steuerungselemente sich nahtlos in die von *HTC* und *Valve Corp.* erstellten Darstellungen für das *HTC VIVE* integrieren soll um insgesamt eine runde Bedienoberfläche zu bilden. Um die Übersichtlichkeit zu bewahren, werden alle Funktionen auf eine Hauptoberfläche implementiert. Das dabei verfolgte Ziel ist, dem Anwender zu jeder Zeit einen Überblick über alle Systemfunktionen zu ermöglichen. Das berührungs- und drucksensitive Touchpad auf der Vorderseite des *HTC VIVE* Controllers bietet dabei die perfekte Eingabeoberfläche für das Auswählen von Funktionen bzw. Hilfssystemen. Zusätzlich dazu ist sie die ideale Stelle für die Darstellung und Überwachung der Systemfunktionen bzw. des Systemzustands. Für diesen Zweck wird das Touchpad in verschiedenen Funktionsfelder aufgeteilt, worauf jeweils eine virtuelle Taste erstellt wird. Für die Unterteilung und Implementierung der virtuellen Tasten wurde ein externes Programm in Unity eingebaut, das von Adrian Biagioli 2017 entwickelt worden ist und seitdem als Free und Open Source Software auf der Plattform *GitHub* verfügbar ist [34]. Der *Vive Virtual Button Editor* ermöglicht eine direkte Integration von virtuellen Tasten auf dem Touchpad auf eine endanwenderfreundliche Art und Weise. Die Definition der einzelnen Felder ist in einem gesonderten Menü untergebracht und mit ihm ist es möglich, die Form und die Position der Taste je nach Anforderungen der jeweiligen Anwendungen anzupassen. In diesem Steuerungsinterface wird mit dieser Software insgesamt zehn virtuellen Tasten implementiert. Dabei wird ein rundes Layout ausgewählt, bei dem im inneren Kreis vier und im äußeren Ring sechs Felder eingeteilt sind. Für die Erweiterung ist in diese Konfiguration eine weitere Unterteilung des äußeren Rings möglich und es können auf dem Touchpad bis zu zwölf Tastenfelder integriert werden. Die Bedienung des Touchpads erfolgt mit Hilfe des Daumens und ihre aktuelle Position wird seitens des SteamVR Plugins standardmäßig durch einen gräulichen Punkt auf der runden Scheibe dargestellt. Die dabei ausgewählte Taste wird zu Signalisierung der Auswahl angehoben und der Controller gibt zusätzlich dazu einen kurzen haptischen Feedback an den Nutzer zurück. Nachdem bei der vorher beschriebenen Scroll-Funktion die gewünschte Komponente gefunden wurde, wird sie durch das Drücken des Touchpads aktiviert. Die Betätigung wird in der VR-Umgebung dargestellt, indem die ausgewählte Taste auf eine Position kurz unter ihrer Ausgangposition sinkt und dabei das Drücken einer realen Taste simuliert. Die Eingabebestätigung wird zum Schluss durch ein haptisches Feedback vervollständigt. Neben dem unscheinbaren mechanischen Klickgeräusch wird bei der Tastenbetätigung zusätzlich eine künstliche Bestätigung mit Hilfe des im Controller

vorhandenen Vibrationsmotors simuliert. Dabei ist das haptische Feedback mit einem festgelegten Zeitspanne von 2000 Mikrosekunden etwa doppelt so stark wie die vorherige für die Signalisierung der Funktionsauswahl verwendete Vibration. Um den Status der jeweiligen Funktionen anzuzeigen werden die aktivierten Systeme mit Hilfe einer Beleuchtung hervorgehoben. Die Deaktivierten werden unbeleuchtet in ihrer definierten Hintergrundfarben dargestellt. Zusammengefasst orientiert sich die auf dem Touchpad befindliche UI an einer Schalttafel, die bspw. bei der heutigen Steuerung von Raumfahrzeuge als zentrale Verwaltung der Systemfunktionen zum Einsatz kommt.

Auf Basis des innerhalb vom *SteamVR* Plugin erstelltes *Prefab* für die Controller der *HTC VIVE* sind für die Umsetzung der Steuerungsschnittstelle zwei C#-Skripte notwendig. Sie erweitern das *GamesObject* um die zuvor erläuterten Funktionalitäten, in dem das *Script „GUIInput.cs*“ für das Erstellen der graphischen Nutzerschnittstelle zuständig ist. Die zentrale Verwaltung und die Veränderung der Eigenschaften der Zustandsvariablen hinter jeder Funktion wird im ersten Teilbereich von *„FlightControl.cs*“ ausgeführt. In diesem Script ist zugleich die Kernkomponente der Steuerungslogik untergebracht und bildet mit ihrer Struktur und Funktion die oberste Ebene der Programmhierarchie für die Funktionalitäten der Satellitensteuerung.

Der gesamte Controller inklusive der implementierten UI ist in Abbildung 5–4 dargestellt. Die auf der rechten Seite abgebildete Vergrößerung der Nutzerschnittstelle veranschaulicht die Position aller Funktionen auf dem Touchpad, die im Rahmen der Steuerungserweiterung integriert wurde. Dabei befindet sich die Bedienung der Steuerungsfunktion im unteren Bereich und die Hilfssysteme und die restlichen Funktionen im oberen Bereich des Touchpads.

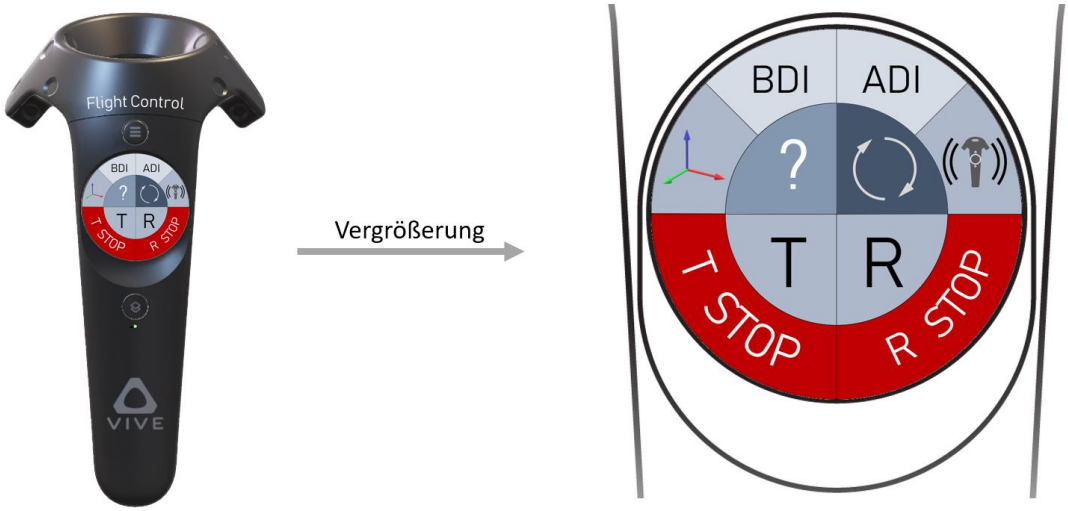

Abb. 5–4: Gesamtansicht des r. Controllers und Interaktionsoberfläche

Die implementierten Tasten werden in der Tabelle 5–1 mit der jeweiligen Position und Funktion übersichtsmäßig zusammenfasst und anschließend genau erläutert.

Tab. 5–1: Tastenbelegung Interaktionsoberfläche des r. Controllers

Hauptgruppe	Tastenposition	Funktion
Flight Control	Innenkreis unten links	Translatorische Steuerungseingabe
	Innenkreis unten rechts	Rotatorische Steuerungseingabe
	Außenring oben links	Translationsbremse
	Außenring unten rechts	Rotationsbremse
	Innenkreis oben rechts	Wechseln Steuerungsmethode
Hilfssystem	Außenring oben links, Segment 1	Neutralpunktanzeige
	Außenring oben links, Segment 2	Basic Deviation Indicator
	Außenring oben rechts, Segment 1	Advanced Deviation Indicator
	Außenring oben rechts, Segment 2	Haptic Feedback
	Innenkreis oben links	Dokumentationsmenü

Ein-/Ausschalten der Steuerungsfunktionen des Satelliten

Die zwei Tasten in der unteren Hälfte des inneren Kreises der UI ist zuständig für die Aktivierung bzw. Deaktivierung der Steuerungsfunktionen. Der linke der Beiden Schalter trägt die Abkürzung „T“ und verwaltet die Möglichkeit der translatorischen Bewegungseingabe. Der Rechte hingegen ist für die rotatorische Bewegungseingabe verantwortlich und ist mit dem Buchstabe „R“ abgekürzt. Standardmäßig sind sie bei der Initialisierung des VR-Systems inaktiv und vermeiden eine unbeabsichtigte Position- bzw. Orientierungsänderung des Satelliten. Durch das Drücken der Taste wird die entsprechende Zustandsvariable von false auf true gesetzt und ermöglicht die Abarbeitung des sich in einer if-Schleife befindlichen Programmierung der Steuerungslogik. Die beiden Funktionen schließen sich gegenseitig nicht aus und somit ist es prinzipiell möglich, Steuerungsbefehle bezogen auf Positions- und auf Orientierungsänderung gleichzeitig einzugeben.

Ein-/Ausschalten der Translations- und Rotationsbremsen

Die beiden Interaktionsflächen am äußeren Ring der Interaktionsfläche bilden die Funktion der Translations- und Rotationsbremsen. Sie sind ebenfalls standardmäßig nach der Initialisierung deaktiviert und werden bei der Betätigung der Taste aktiv. Dabei verändert sich wie schon vorher die zugewiesene Zustandsvariable und diese Information wird mit Hilfe der Schnittstelle zu der RACOON-Lab Simulation übertragen. Dort wird sie an die für die Ansteuerung der Bremsen zuständigen Module übergeben, die diese Funktion letztendlich aktiviert. Die Farbe der beiden Tasten ist, im Gegensatz zu den Sonstigen, nicht in einem blau-grauen Farbmuster gehalten, sondern verwendet gezielt die Signalfarbe rot. Die Orientierung der Farbgebung ist dabei an die gängige farbliche Darstellung für die Stopp-Funktion angelehnt, bei der in den meisten Fällen die rote Farbe verwendet wird.

Änderung der Eingabemethode

Durch das Betätigen dieser Taste wird die standardmäßige Gestensteuerung deaktiviert und die Steuerungsmethode wechselt in die im Kapitel 5.3.2.2 vorgestellten alternativen Analogsteuerung. Dabei wird die aktuelle UI für Funktionsauswahl ausgeblendet und durch eine an die alternative Steuerungsmethode angepasste graphische Darstellung des Touchpads ersetzt.

Anzeigen/Ausblenden der Neutralpunktanzeige

Das Hilfssystem auf der linken Seite des oberen äußeren Ringes ist die Neutralpunktanzeige, die eine entscheidende Rolle bei der Orientierung bei der Steuerungseingabe hat. Sie wird bei der standardmäßigen Gestensteuerung mit Hilfe von zwei dreidimensionalen Koordinatensysteme dargestellt. Das erste KOSY bildet den innerhalb der VR-Anwendung frei definierbaren Nullpunkt der Steuerung im ortsfesten System der virtuellen Umgebung ab. Für ihre Darstellung wurde das *GamesObject* „*COSY Right*“ erstellt und innerhalb des *Objects* „*CameraRig*“ auf der Ebene der beiden Controllerobjekte und das *GamesObject* des VR-HMDs untergeordnet. Das „*CameraRig*“ ist nichts anderes als die Zusammenfassung der Interaktionskomponenten der *HTC VIVE* und verfügt relativ zum Basiskoordinatensystem des virtuellen Raums über keinerlei Verschiebung und Rotation. Das Zweite ist genau wie das erste Koordinatensystem aufgebaut, wobei dieses in diesem Fall körperfest am KOSY des Controllers gebunden ist. Es hat die Aufgabe in Kombination mit dem graphisch identischen ortsfesten System dem Nutzer sowohl bei der Ausführung einer translatorischen Bewegung, als auch bei einer Drehbewegung zur Änderung der Orientierung des Satelliten einen Bezug zu geben. Wie schon das erste KOSY, erfolgt seine graphische Darstellung ebenfalls über ein *GamesObject*, welches unter dem Spielobjekt des rechten Controllers angeordnet ist.

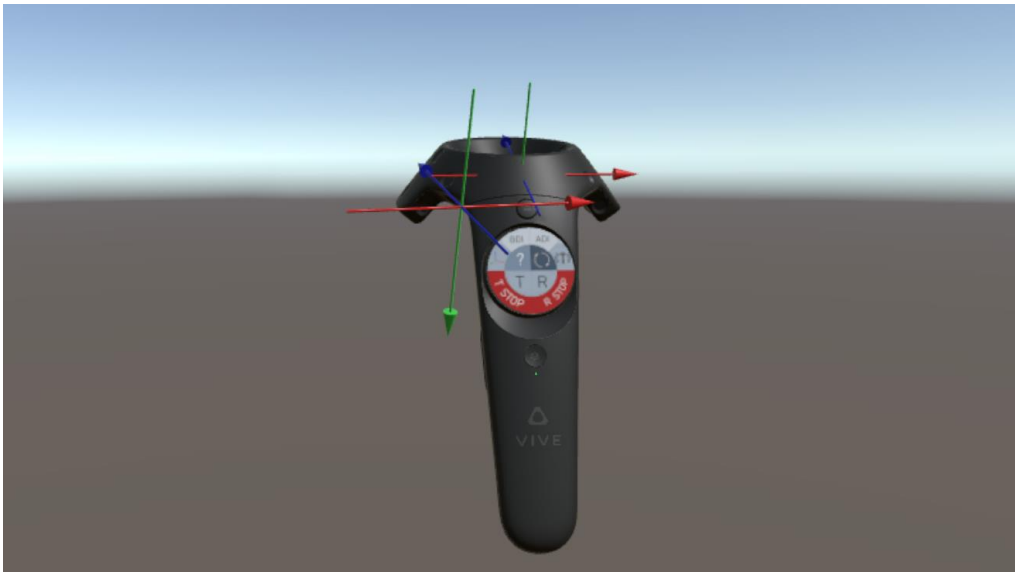

Abb. 5–5: Neutralpunktanzeige

Mit Hilfe der Neutralpunktanzeige wird insbesondere die ein- oder zweiachsige Eingabe der Translation und die einachsige Rotation vereinfacht, während parallel dazu die Genauigkeit der Ansteuerung erhöht wird. Dies geschieht dadurch, dass bei den einachsigen Operationen das raumfeste KOSY als Referenz angenommen

werden kann und die Eingabe relativ zu der festgelegten Nullkonfiguration getätigt wird. Beispielsweise bei der einachsigen Rotation ist es lediglich darauf zu achten, dass die Achse, um der die Rotation gewünscht ist, in den beiden KOSY parallel angeordnet ist oder so weit wie möglich übereinstimmt. Abbildung 5–5 stellt das zuvor erläuterte System graphisch dar. Die Neutralpunktanzeige gehört zu eine von zwei Funktionen, die standardmäßig nach der Initialisierung aktiv sind. Dazu ist sie nur sichtbar, wenn das für die standardmäßige Gestensteuerung benötigte Nullpunkt im Raum durch das Betätigen des Grip-Button definiert wurde.

Für individuelle Anpassung der Benutzerergonomie können alle Ausgangseinstellung über das *GamesObject* des rechten Controllers im *Inspector*-Fenster angepasst werden.

Anzeigen/Ausblenden der Basis-Deviationsanzeige

Basierend auf die Neutralpunktanzeige ist der BDI eine Erweiterung zur Darstellung der Auslenkung des Steuerungsgerätes bei der translatorischen und rotatorischen Bewegung im raumfesten KOSY.

Für die Darstellung der Translation berechnet die Funktion aus den verfügbaren Daten den dreidimensionalen Differenzvektor. Für dessen Implementierung wird ein Pfeil mit einer normierten Länge parallel zu der x-Achse als *GamesObject* in der Szene erstellt. In dieser Grundkonfiguration, bei der der Pfeil sich orthonormal zu der y-z-Ebene befindet und in die Richtung der positiven x-Achse zeigt, haben alle Rotationskomponenten dieses *GamesObjects* den Wert 0. Um den Differenzvektor darstellen zu können, werden insgesamt zwei Charakterisierungsparametern, die betragsmäßige Länge des Auslenkungsvektors und die relative Rotation zwischen der Pfeilrichtung und die Orientierung der tatsächlichen Auslenkung benötigt.

Zur Bestimmung des erstgenannten Parameters wird zu nächst der Differenzvektor zwischen dem virtuell gesetzten Nullpunkt und der Controllerposition mit Hilfe der mathematischen Formel **(5–2)** gebildet. Daraus wird im Anschluss mit Hilfe der Formel **(5–3)** die betragsmäßige Länge ermittelt. Die folgenden Formeln veranschaulichen den Berechnungsvorgang:

$$\Delta\vec{x} = \begin{pmatrix}\Delta x\\ \Delta y\\ \Delta z\end{pmatrix},\quad \vec{x}_{Controller} = \begin{pmatrix}x_{Controller}\\ y_{Controller}\\ z_{Controller}\end{pmatrix},\quad \vec{x}_{Nullpunkt} = \begin{pmatrix}x_{Nullpunkt}\\ y_{Nullpunkt}\\ z_{Nullpunkt}\end{pmatrix} \quad \textbf{(5–1)}$$

$$\Delta\vec{x} = \vec{x}_{Controller} - \vec{x}_{Nullpunkt} \quad \textbf{(5–2)}$$

$$l = |\Delta\vec{x}| = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \quad \textbf{(5–3)}$$

$\Delta\vec{x}$ entspricht den Differenzvektor, $\vec{x}_{Controller}$ ist die absolute Position des Controllers im Raum und $\vec{x}_{Nullpunkt}$ die absolute Position des vom Benutzer festgelegten Nullpunktes. Die Länge l ist die betragsmäßige Länge des Differenzvektors $\Delta\vec{x}$.

Der zweiter Parameter wird nicht wie beim Differenzvektor mit Hilfe der Anwendung analoger Formeln, sondern mit Hilfe von in *UnityEngine* bereits vorhandenen Methoden durchgeführt. Dabei erfolgt die Berechnung nicht in Eulerwinkel, sondern

durch die Darstellung der Rotation durch Quaternionen. Zwar besitzen die aus insgesamt drei Winkeln bestehenden Eulerwinkeln eine intuitivere und für die Menschen verständlicheres Format, der größte Nachteil daran ist jedoch, dass es zu einer kardanischen Blockade führen kann, bei der zwei der drei möglichen Drehachsen parallel liegen und dadurch das System einen Freiheitsgrad verliert [35, 36]. Die Quantisierung der Rotation mit Hilfe von Quaternionen ist von dem Ereignis nicht betroffen. Anderes als bei den Eulerwinkeln werden sie mit Hilfe von vier Variablenparametern x, y, z und w charakterisiert, wobei der Verständnis der auch Hamilton-Zahlen genannten Zahlenmenge ein mathematisches Spezialwissen voraussetzt und nicht in der Standardausbildung der Mathematik vorhanden ist [36]. Für die Anwendung in *Unity* im Rahmen der auszuführenden Arbeiten ist händische Veränderung der Charakterisierungsparametern nicht notwendig. Aus diesem Grund ist ein tiefgründiges Verständnis der Quaternionen für diese Art der Anwendung nicht zwingend erforderlich [35]. Weitere Informationen über die Hamilton-Zahlen und ein ausführlicher Vergleich der Vor- und Nachteile von Eulerwinkeln und Quaternionen ist in [36] zusammengefasst.

Um die Deviationsanzeige in die ausgelenkte Richtung zu zielen, wird die Rotation des Differenzvektors gegenüber der x-Achse des raumfesten KOSY des Nullpunktes ermittelt, da die x-Achse parallel zu der definierten Pfeilrichtung liegt und die gleiche Rotation besitzt. Das geschieht mit Hilfe der statischen Methode „*FromtoRotation*“ aus dem Quaternion-Struct der *UnityEngine*, bei der aus der Eingabe der beiden Vektoren die relative Rotation herausgegeben wird. Die ermittelte Drehung wird abschließend in die Rotation des Pfeilelements übernommen.

Für die Darstellung der Rotation werden Rotationssymbole an den jeweiligen Pfeilspitzer des raumfesten Koordinatensystems angebracht. Sie sind dem *GamesObject* „*COSY Right*“ untergeordnet und für jede Achse ist es möglich, eine positive oder negative Rotationsrichtung anzuzeigen. Je nachdem ob die Werte der Rotationswinkeln ein positives oder negatives Vorzeichen oder den Wert null haben, werden dementsprechend die Rotationsanzeige angepasst. Die Rotationswinkeln wurden in der im Kapitel 5.3.2.1 erläuterte Satellitensteuerungskomponente ermittelt und hier für diesen Zweck abgefangen.

Abb. 5–6: Basic Deviation Indicator

Anzeigen/Ausblenden der erweiterten Deviationsanzeige

Die Funktion rechts oben am äußeren Ring ist eine weitere Erweiterung der Auslenkungsanzeige und basiert, wie schon der BDI, auf die Neutralpunktanzeige. Für das ADI wurde insgesamt 12 quadratischen Flächen als *GamesObjects* unter dem Objekt „*Plane*“, das wiederum dem Objekt „*COSY Right*“ untergeordnet ist, implementiert, die jeweils die Quadraten der x-y-, x-z- und y-z-Ebene repräsentieren. Die Flächen sind nach der Aktivierung des Hilfssystems zunächst ausgeblendet und werden je nach Auslenkungsrichtung des Controllers eingeblendet. Für die Integration des ADI werden keine mathematischen Formeln benötigt. Die Abarbeitung erfolgt innerhalb der Programmierung mit Hilfe des zuvor in der Implementierung von BDI definierten Differenzvektors durch eine Vielzahl an if-Abfragen. Je nachdem ob die Werte der x, y und z Komponente des Differenzvektors ein positives oder negatives Vorzeichen oder den Wert null haben, werden die einzublendenden Flächen identifiziert. Das bedeutet, dass bei einer einachsigen Eingabe zwei, bei zweiachsiger Eingabe einen und bei einer dreidimensionale Eingabe drei Quadranten eingeblendet. Bei einer eindimensionalen Eingabe ist das Einblenden von zwei Flächen notwendig, da der Differenzvektor parallel zur Koordinatenachse der Eingaberichtung ist und somit sowohl den linken als auch den rechten Quadranten in der Eingaberichtung angesteuert wird.

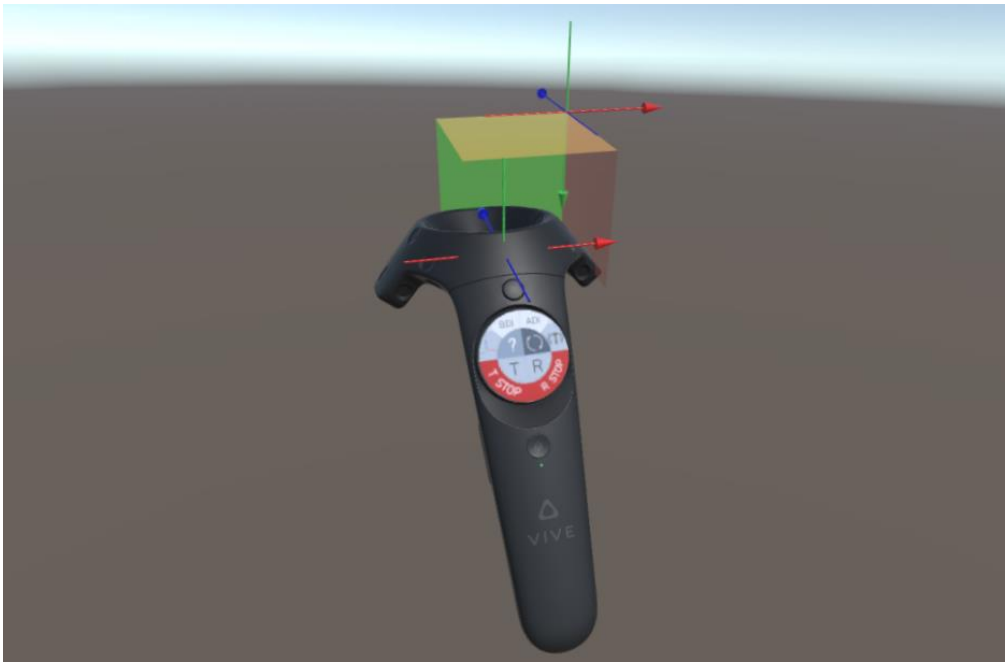

Abb. 5–7: Advanced Deviation Indicator

Ein-/Ausschalten des haptischen Feedbacks

Die Funktion auf der rechten Seite des oberen äußeren Ringes ist der zentrale Schalter für das haptisches Feedback für den rechten Controller. Durch das Betätigen dieses Knopfes lässt sich das unter anderem in den Kapiteln 5.3.2.1 und 5.3.2.2 vorgestelltes haptisches Feedback für die Steuerungseingabe aktivieren bzw. deaktivieren. Das Eingabefeedback bei der Betätigung des Touchpad-Buttons sowie die Rückgabe beim Durchscrollen der Schnittstelle für Funktionsauswahl ist unabhängig vom Zustand dieser Funktion und somit immer verfügbar. Diese Funktion ist die zweite Funktion, die standardmäßig nach der Initialisierung aktiv ist.

Anzeigen/Ausblenden des Dokumentationselements

Die eindeutige Charakterisierung erfolgt mit Hilfe geeigneter Abkürzungen und Symbole und wird innerhalb der VR-Oberfläche im Dokumentationsmenü zusammengefasst. Es lässt sich durch das Drücken der dazugehörigen Taste im inneren Kreis der Schnittstelle einblenden. Sie bietet eine kurze Erklärung der Schnittstelle für Funktionsauswahl des rechten Controllers und verwendet sowohl graphische als auch textuelle Elemente.

5.3.2 Steuerungseingabe des Satelliten

Die Hauptfunktionalität des rechten Controllers ist es den Chasersatelliten steuern zu können. Dafür ist es notwendig die Eingabe des Nutzers in VR zu erfassen und an die Steuerungsschnittstelle der RACOON-Lab Simulation weiterzuleiten. Dieses Interface befindet sich im Programm „Visualization“, die für die Erstellung der Software-Visualisierung innerhalb der RACOON-Lab Simulation verantwortlich ist. Im Rahmen der Steuerungserweiterung wurde die Steuerungskomponenten aus „Visualization“ entfernt und ein neues Modul mit dem Namen „Flight Control“ für die aktive Steuerung des Chasersatellitens erstellt. Der notwendige Datenaustausch zwischen beiden Modulen werden mit Hilfe einer duplexfähigen Kommunikationsschnittstelle mit der IPC-Methode Named Pipes sichergestellt.

Für die Erstellung einer zeitkontinuierlichen Anwendung mit Hilfe einer Software werden die im Programm implementierten Komponenten zyklisch abgearbeitet. Das bedeutet, dass die temporale Kontinuität innerhalb des Programms in zeitdiskreten Schritte aufgeteilt wird und das Programm innerhalb dieses Zeitschritts einmal vollständig abgearbeitet wird. Dadurch, dass das Durchlaufen der Skripte und die Aktualisierung der zugehörigen Parametern nur wenige Millisekunden in Anspruch nimmt, wird ein quasi-echtzeitfähiges System erschaffen.

Das Steuerungssystem beginnt in *Unity* bei der Verarbeitung der Steuerungseingaben des Nutzers. Die dabei für die Steuerung des Satelliten erforderlichen Parameter werden zur Weiterverarbeitung herausgegeben und im nächsten Schritt an das für die Verarbeitung und Anbindung der Steuerungseingaben zuständige Flight Control Modul der RACOON-Lab Simulation weitergeleitet. Dort erfolgt abschließend die Anbindung der Steuerungskommandos an die bereits für die Tastatur- und Joysticksteuerung vorhandene Datenschnittstelle. Die Rückgabe der getätigten Steuerung erfolgt visuell über das Bild der Software-Visualisierung auf dem HMD des VR-Headsets. Um das Ziel der einzelnen Schritte zu erreichen, werden die jeweils zu erfüllenden Aufgaben mit Hilfe verschiedener Unterprogramme bearbeitet. Die Verwaltung der

Unterprogramme findet sowohl in Unity, als auch in der RACOON-Lab Simulation objektorientiert in einem Masterprogramm statt. Es wird innerhalb eines Zeitschritts durchlaufen und ruft die für die Abarbeitung der Aufgaben benötigten Methoden der verschiedenen Unterprogramme auf. In den nachfolgenden Kapiteln wird der Ablauf innerhalb der verschiedenen Prozessschritte detailliert erläutert.

5.3.2.1 Standardmäßige Eingabemethode

Am Anfang der Verarbeitungskette für die Satellitensteuerung in VR muss die Eingabe des Benutzers vom Programm erfasst werden. Das Prinzip bei der standardmäßigen Eingabemethode ist es, wie bei der Konzeptionierung in Kapitel 4.1.2, mit Hilfe der Bewegung des Controllers die gewünschte translatorische und rotatorische Steuerungsbefehle des Chasersatellitens zu erreichen. Die grundsätzliche Ansteuerung des Motion-Controllers ist innerhalb vom *SteamVR* Plugin mit Hilfe des *GamesObjects* „*Controller (right)*“ realisiert. Es beinhaltet mit seinen Untermodulen alle Funktionen und Komponenten des rechten Controllers und kann durch das Hinzufügen von Eigenschaftsfelder, Scripts oder untergeordnete Spielobjekte mit zusätzlichen Funktionen erweitert werden. Für die Erkennung der Steuerungseingaben und ihre Vorbereitung für die anschließende Datenübertragung zu dem entsprechenden Modul innerhalb der RACOON-Lab Simulation wird dieses Spielobjekt durch die Integration des C# Skripts „*FlightControl.cs*“ erweitert. Dieses Codeprogramm ist, wie schon in Kapitel 5.3.1 erwähnt, das Hauptprogramm für alle implementierte Funktionen des rechten Controllers und beinhaltet unter anderem die komplette Logik für die Verarbeitung der Steuerungseingaben. Das Skript lässt sich in zwei Bereichen unterteilen. Der erste Teil weist eine Ähnlichkeit zu einer Schalttafel auf und ist zuständig für die Verwaltung der Zustandsvariablen. Dabei wird an diese Stelle, wie in Kapitel 5.3.1 erklärt, der Zustand der booleschen Parameter je nach der Eingabe des Nutzers verändert und den Wert der Zustandsvariable für die Charakterisierung des Systemzustands gespeichert. Im nachfolgenden Teilbereich des Skriptes sind die Funktionen des rechten Controllers implementiert. Je nach dem momentanen Systemzustand werden die verschiedenen aktiven Funktionen abgearbeitet.

Damit ein Steuerungsbefehl erkannt wird, ist es notwendig verschiedene Voraussetzungen zu erfüllen. Als erstes wird im Programmverlauf geprüft, ob die translatorische oder die rotatorische Steuerung aktiv ist. Diese sind standardmäßig ausgeschaltet und lässt sich auf Knopfdruck aktivieren. Die zweite Voraussetzung ist, dass eine Basis gesetzt wurde. Die Existenz der Basis ist essentiell für die gesamte Steuerung, weil mit ihr ein virtueller Nullpunkt als Referenz für die anschließenden Eingaben definiert wird. Eine Basis ist nach der Initialisierung standardmäßig nicht definiert und somit besitzt die dafür zuständige Zustandsvariable den Wert *false*. Die Festlegung lässt sich durch das einmalige Betätigen des Grip-Buttons erreichen. Dabei wird ein Referenz auf das *GamesObject* des rechten Controllers gelegt und auf die Positions- und Orientierungsdaten zugegriffen. Die drei Parameter bzgl. der Position werden direkt in einen Vektor gespeichert, während die Rotationsdaten, je nach Präferenzen des Benutzers, für eine höhere Ergonomie um die x-Achse gedreht wird. Das Koordinatensystem des Controllers ist in dem *GamesObject* so definiert, dass die z-Achse parallel zum Griff des Controllers verläuft. Aufgrund dessen, dass der Controller bei der Bedienung bspw. für eine bessere Darstellung des Touchpads in einem bestimmten Winkel gehalten wird, würde bspw. somit die z-Achse dieses Spielobjektes auf der Vorderseite schräg nach oben zeigen. Für die Anpassung wurde

das Koordinatensystem um +45° um die x-Achse gedreht und die x-, y- und z-Achse haben wieder ihre gewöhnliche Orientierung nach „rechts“, „unten“ sowie „vorne“. Diese Rotationswinkel wurde empirisch ermittelt und lässt sich im *Inspector*-Fenster anpassen. Bei Rotationen in Unity ist zu beachten, dass das KOSY standardmäßig als Linkssystem definiert ist. Das bedeutet, dass die Drehrichtungen mit Hilfe der linken Hand ermittelt werden müssen, wobei sie zu den Rotationsrichtungen der rechter Hand spiegelverkehrt sind. Bei der nochmaligen Betätigung des Grip-Buttons wird der Wert der Zustandsvariable wieder zurückgesetzt und die Basis deaktiviert. Die eigentliche Steuerung ist aktiv, sobald die Zustandsvariablen für die translatorische oder rotatorische Steuerung sowie die zur Darstellung einer festgelegten und aktiven Basis den Wert *true* besitzen.

Die Steuerungslogik ist folgendermaßen aufgebaut: Zur Absicherung gegen einer unbeabsichtigten Eingabe ist das Betätigen des Triggers bei der Auslenkung des Controllers notwendig. Dabei wird die Position des Triggers abgefragt, die den Wertebereich [0, 1] einnehmen kann. Der Wertebereich ist so definiert, dass der variable Auslenkungsweg einen Wert zwischen 0 und 0.8 einnehmen kann und die letzten Zweizehntel den Knopf am Ende des Auslenkungswegs darstellt. Bei der Implementierung hat es sich herausgestellt, dass der Wert 0, aufgrund von Staubrückstände sowie Einlagerungen während des Gebrauchs in der Mechanik des Triggers, nicht immer sicher erreicht werden kann. Aus diesem Grund wurde für die Auslenkung des Triggers einen empirisch Wert von 0.05 definiert, bei dessen Überschreitung der Trigger als ausgelenkt markiert wird. Die letztendliche Befehlseingabe geschieht mit der Veränderung der Position sowie Orientierung des Controllers gegenüber der zuvor festgelegten Basis. Die Neutralpunktanzeige ist mit ihrer Darstellung des Koordinatensystem der Basis und des Controllers eine gute Orientierungsmöglichkeit für die translatorische und rotatorische Steuerungseingabe. Um die Ungenauigkeiten der Pseudo-Gestensteuerung, wie im Kapitel 4.3 erklärt, zu minimieren, ist es notwendig eine Deadzone für die Parameter der Translation und Rotation zu definieren. Die Deadzone ist ein Wertebereich um den virtuellen Nullpunkt, bei der die Auslenkung nicht erfasst wird und die Ausgabewerte für die Satellitensteuerung auch bei aktiver Steuerung und ausgelenktem Trigger den Wert null annehmen. Das bedeutet im Falle für die translatorische Eingabe, dass der Controller sich in einer kugelförmigen Umgebung bewegen kann, in der die Auslenkungswerte vernachlässigt werden. Der festzulegende Wert der Deadzone charakterisiert dabei den Radius der Kugel und wurde mit einem empirisch ermittelten Wert von 0.02 festgesetzt. Die Längen-/ Positionsangaben werden zwar in *Unity* dimensionslos dargestellt, jedoch sind sie an das metrische System angelehnt. Somit entspricht der Wert von 0.02 einer Länge von 2 cm in der Realität und der virtuellen Umgebung. Analog dazu wurde für die Rotation einen Wert von 10° definiert, unter der einer Drehung nicht vom System erkannt wird. Dabei bezieht sich die Definition auf den einseitigen Drehwinkel einer Koordinatenachse und somit werden zusammenfassend Abweichungen von ±10° toleriert. Nach der Überwindung des zuvor erläuterten Bereichs werden die translatorischen und rotatorischen Eingabedaten in einem Vektor mit sechs Elementen zusammengefasst. Um die ruckartige Eingabe nach der Überwindung der Deadzone zu vermeiden, werden die relativen Auslenkungswerte des Controllers zum virtuellen Nullpunkt an diese Stelle auf null zurückgesetzt. Für die Ansteuerung der multisensorische Wahrnehmung wurde für die Auslenkung des Controllers ein haptisches Feedback mit variabler Stärke

implementiert. Die Intensität bezieht sich dabei auf die Daten, die im vorher erläuterten Vektor gespeichert worden sind, und steigt mit zunehmender Auslenkung. Der Vibrationsmotor des Controllers lässt sich mit Hilfe der Definition der Dauer des haptischen Impulses ansteuern und zusammen mit der zyklischen Abarbeitung des Programmes wird daraus einen fortwährenden haptischen Feedback erzeugt. Die Dauer des haptischen Impulses ist prinzipiell freiwählbar und ist in Mikrosekunden angegeben. Es wurde festgestellt, dass bei einer sequentiellen Ansteuerung ab einer Impulsdauer von 1000 Millisekunden die Positionsbestimmung des Controllers verschlechtert wird und er beginnt in der VR-Ansicht zu wackeln und im schlimmsten Fall anfängt in einer beliebigen Richtung abzudriften. Das haptische Deviationsfeedback erzeugt, je nach der Größe der Auslenkung, eine variablen Vibration mit einer maximalen Impulsdauer von 300 Millisekunden. Dies wird nach einer Auslenkung gemessen ab dem Verlassen der Deadzone bei der Translation nach etwa 10 cm und bei der Rotation nach 60° erreicht.

5.3.2.2 Alternative Eingabemethode

Für die Steuerung des Chasersatelliten wurde neben der standardmäßigen Pseudo-Gestensteuerung eine weitere Eingabemethode implementiert. Die alternative Steuerungsmethode basiert nicht auf die räumliche Position und Orientierung des Eingabegeräts, sondern realisiert die verschiedenen Steuerungsbefehle mit Hilfe der vorhandenen Tasten des *HTC VIVE* Controllers. Die Aktivierung der analogen Steuerungsmethode erfolgt durch das Betätigen der dafür vorgesehene Taste auf der Hauptinteraktionsoberfläche und lässt sich durch die zweimalige Betätigung des Grip-Buttons wieder ausschalten. Nach dem Wechsel der Steuerungsmethode wird das Hauptinteraktionsmenü ausgeblendet und durch die in Abbildung 5–8 dargestellte Graphik ersetzt. Sie ist die Projektion des Koordinatensystems auf der x-z-Ebene und hat die Aufgabe, dem Nutzer bei der Orientierung der Steuerungseingaben zu unterstützen. Analog dazu wird nach Deaktivierung der alternativen Steuerungsmöglichkeit das Hauptinteraktionsmenü wieder eingeblendet und die Standardeingabemethode wieder aktiviert.

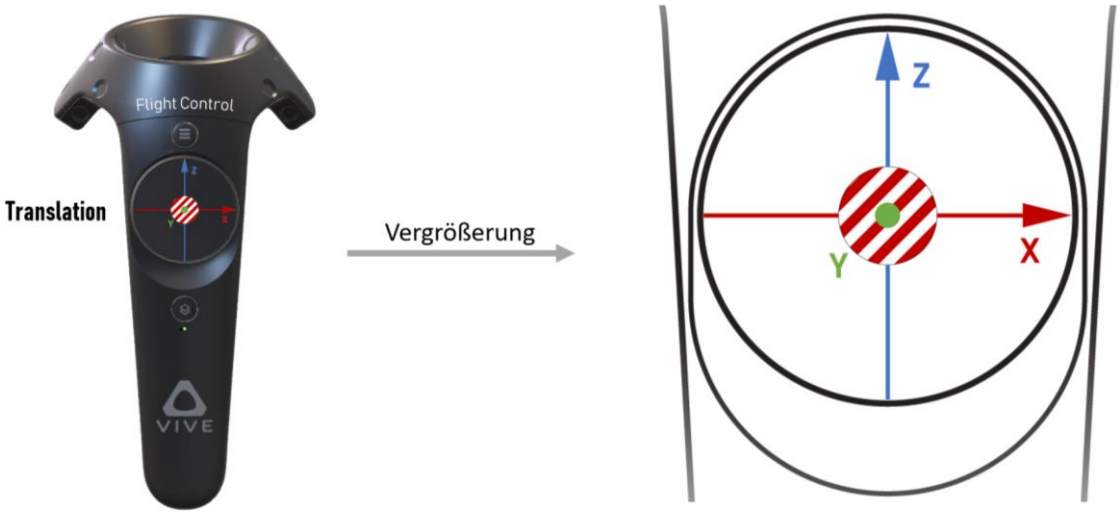

Abb. 5–8: Visuelles Nutzerinterface für die Analogsteuerung

Die analoge Steuerungsmethode ist eine vollständiges Steuerungssystem und es ist möglich, sowohl die Position, als auch die Orientierung des Chasersatelliten zu

verändern. Aufgrund dessen, dass der Controller nur über eine überschaubare Anzahl an Hardwaretasten verfügt, ist es nicht möglich gleichzeitig eine translatorische und rotatorische Steuerungseingabe zu machen. Das Umschalten zwischen der Veränderung der Position und der Orientierung geschieht durch das einmalige Drücken des Grip-Buttons. Für die visuelle Ausgabe des momentanen Steuerungszustands wird entweder bei der aktiven translatorischen Steuerung links neben dem Touchpad die Aufschrift „Translation“, oder bei der aktiven rotatorische Steuerungsmöglichkeit auf der rechten Seite neben dem Touchpad das Wort „Rotation“ eingeblendet. Die Eingabe der Steuerungsbefehle wird mit Hilfe der Kombination des Touchpads und des Triggers erfasst. Das erstgenannte Hardwareelement des Controllers ist eine berührungssensitive Fläche, bei der die Position der Kontaktstelle des Fingers mit Hilfe eines zweidimensionalen Koordinatensystems ermittelt wird. Die beiden Achsen können einen Wert im Bereich von [-1, 1] annehmen und durch das Auslesen der beiden Zahlenwerte ist es möglich, eine variable zweidimensionale Eingabe in der x-z-Ebene zu verarbeiten. Die Befehlseingabe in der dritten Dimension wird mit Hilfe des Triggers in Kombination mit der drucksensitive Komponente des Touchpads. Aufgrund dessen, dass der Trigger genau gegenüber vom Touchpad auf der anderen Seite des Controllers liegt, ist sie die beste Umsetzungsvariante für die Befehlseingabe in die y-Richtung, sowohl in der Funktionsweise, als auch in Bezug auf die Nutzerergonomie. Mit Hilfe des variablen Auslenkungswegs und das Auslesen des entsprechenden Wertes im Bereich von [0, 1] wird darüber hinaus eine variable Eingabe ermöglicht. Für einen Steuerungsinput in die negative y-Achsenrichtung muss dazu nur der Trigger ausgelenkt werden. Um eine Steuerungseingabe in die positive y-Richtung einzuleiten, ist es zusätzlich zu der Betätigung des Triggers notwendig die Touchpad-Taste gedrückt zu halten.

Die Eingabe von translatorischen Befehle ist nach dem gleichen technischen Prinzip aufgebaut, wobei hier eine gute räumliche Vorstellungskraft vorausgesetzt wird. Die Rotation um die z-Achse wird mit Hilfe der Fingerposition entlang der in der Abbildung 5–10 dargestellten x-Achse erreicht und analog dazu die Rotation um die x-Achse durch die Auslenkung entlang der z-Achse. Die sich dahinter befindliche Logik lässt sich am besten mit Hilfe eines Würfels darstellen, bei der die Betrachtung aus der Draufsichtperspektive geschieht. Um aus dieser Perspektive die linke Seitenfläche sehen zu können, ist es notwendig den Würfel um 90° nach rechts zu drehen. Die dabei entstandene Rotation erfolgt um die in der Sichtweise nach oben zeigende Raumachse, wobei die für die Drehung notwendige mechanische Krafteinwirkung nach rechts zeigt. Übertragen auf die analoge Steuerungsvariante ist die positive Rotation um die z-Achse mittels einer Krafteinwirkung bzw. Bewegung in die positive x-Richtung zu erreichen. Die Einleitung einer Rotation um die y-Achse erfolgt nicht auf eine logische Art und Weise und ihre Implementierung ist nach dem gleichen Schema wie bei der translatorischen Bewegung in die y-Achsenrichtung aufgebaut.

Wie schon bei der Standardsteuerungsmethode, ist die Definition einer Deadzone aus einer Kombination von technische Einschränkungen und der Verbesserung der Nutzerergonomie erforderlich. Daher wurde für den Trigger analog zu der standardmäßigen Steuerung einen Deadzone-Bereich von ca. 6 % des möglichen Auslenkungswegs definiert. Wie in Abbildung 5–8 ersichtlich, wurde auf dem Touchpad innerhalb des rot-weiß schraffierten Bereichs ebenfalls einen Deadzone-Bereich definiert. Dies ist notwendig, weil die Erfassung der Steuerungseingaben nicht mit Hilfe einer Taste ausgeführt ist, sondern meistens berührungssensitiv erfolgt. Mit

Hilfe dieser Maßnahme soll eine Fläche erschaffen werden, bei der die alleinige Berührung nicht zu einer Steuerungseingabe resultiert, sondern für die Untersetzung der Nutzerorientierung auf dem Touchpad genutzt werden.

Um auch hier eine multimodale Wahrnehmung zu erschaffen, wird der Nutzer nach einer erfolgten Eingabe ebenfalls mit einem haptischen Feedback versorg. Das dabei verwendete Aktivierungsprinzip ist ähnlich zu das der Pseudo-Gestensteuerung und gibt dem Operator nach der Überwindung der Deadzone eine sich in der Intensität variierende Rückgabe in Form der Vibration. Die Impulsdauer wurde hier an das Eingabemedium angepasst und auf einen maximalen Wert von 150 Millisekunden beschränkt. Der Grund für die Halbierung des Wertes ist, dass das Vibrationsmodul sich unmittelbar unter dem Touchpad befindet und durch den bei der Steuerungseingabe immerwährenden Kontakt zum Finger einen stärkeren Impuls übertragen wird.

5.3.3 Datenanbindung für IPC-Übertragung

Nachdem die Steuerungsdaten und der Zustand der Translations- und Rotationsbremsen ermittelt wurden, erfolgt an diese Stelle die Übermittlung dieser Daten an das zuständige Modul in der RACOON Simulation. Für diesen Fall wird nur eine Datenübertragung in eine Richtung benötigt. Mit Hilfe der IPC-Methode *Named Pipes* wird hierbei eine Datenverbindung zum Flight Control Modul erschaffen und übermittelt die notwendigen Daten in Form eines Bytearrays mit der Länge 32. Um alle zu übertragenden Daten in eine Variable zusammenzufassen, werden die 8 Eingabedaten im Gleitkommazahlformat zunächst in das Byteformat umgewandelt und anschließend sequentiell in das Bytearray mit einer ausreichenden Länge geschrieben. Die Datenübertragung erfolgt in diesem Fall asynchron und ist für diesen Einsatzzweck ausreichend. Die asynchrone Übertragung hat im Vergleich mit der synchrone Datenübertragung den Vorteil, dass der Empfänger und der Absender voneinander unabhängig sind und dadurch keine Wartezeit der beiden Programme bei dem Datenaustausch entsteht.

5.3.4 Eingabeschnittstelle für die Satellitensteuerung in der RACOON Simulation

Die Einbindung der von *Unity* übertragenen Steuerungsparametern und der Zustand der beiden Bremsen in die RACOON Simulation erfolgt innerhalb des Flight Control Moduls im Unterprogramm „*ChaserInput.cs*“. Nach dem Empfangen des Bytearray aus der Datenübertragung werden sie, nach der beim Zusammenfassen der insgesamt acht Variablen verwendeten Reihenfolge, wieder herausgelesen und dementsprechend abgespeichert. Dabei erfolgt die Umwandlung bei den sechs Steuerungsparametern zurück zum ursprünglichen Gleitkommaformat und bei den Parametern für die Ansteuerung der Translations- und Rotationsbremsen direkt zum booleschen Format. Aufgrund dessen, dass die Steuerung des Chasersatellitens in der RACOON Simulation bereits möglich ist, ist eine neue Implementierung für die Integration der Steuerungsbefehle in die Starrkörpersimulation nicht mehr notwendig. Für die Übertragung der Steuerungsbefehle aus der VR-Simulation werden zum Schluss die bereits vorhandene Schnittstelle herangezogen und jeweils mit den entsprechenden Steuerungsparametern verbunden werden.

5.4 Implementierung der Visualisierungserweiterung

Die Visualisierungsfunktionen innerhalb der VR-UI wurde grundlegend überarbeitet und alle Funktionen, die nicht zur Steuerung des Satelliten gehören, wurden zusammengefasst im linken Motion-Controller der *HTC VIVE* untergebracht.

5.4.1 Interface für Funktionsauswahl und Menüführung

Das Menü für Funktionsauswahl ist auf dem gleichen Prinzip der im Kapitel 5.3.1 vorgestellten Interaktionsschnittstelle auf dem Touchpad aufgebaut. Aus diesem Grund wird an dieser Stelle auf die Erklärung des Implementierungsverfahrens und der Funktionsweise verzichtet. Auf Basis des innerhalb vom *SteamVR* Plugin erstelltes *Prefab* für die Controller der *HTC VIVE* sind für die Umsetzung der Visualisierungsschnittstelle zwei C#-Skripte notwendig. Das *Script „GUIInput.cs*“ ist für das Erstellen der GUI zuständig und die zentrale Verwaltung und die Veränderung der Eigenschaften der Zustandsvariablen der Funktionen wird innerhalb von *„CameraInput.cs*“ ausgeführt

Der gesamte Controller inklusive der implementierten UI ist in Abbildung 5–9 dargestellt. Die auf der rechten Seite abgebildete Vergrößerung der Nutzerschnittstelle veranschaulicht die Position aller Funktionen auf dem Touchpad, die für die Visualisierung in der virtuellen Realität notwendig sind. Um die Funktionsmenüs der beiden Controller einheitlich zu gestalten, wurde beim Entwurf des Visualisierungsmenüs die Positionen der ebenfalls auf dem Steuerungsmenü vorhandenen Tasten übertragen. Die obere Hälfte des abgebildeten Auswahlmenüs ist weitestgehend identisch mit dem der rechten Hand. Die Taste oben rechts im inneren Kreis wurde aufgrund der fehlenden alternativen Eingabemethode nicht übernommen und ist im Rahmen des Visualisierungsmenüs mit keine Funktion belegt. Die Aktivierung der translatorischen und rotatorischen Steuerung der Kameraperspektive ist, wie schon bei der Steuerung auf dem rechten Controller, im unteren Bereich des inneren Kreises untergebracht. Verteilt im unteren äußeren Ring sind die im Rahmen der vorangehenden Bachelorarbeit Hilfssysteme und Funktionen untergebracht. Dazu gehört die Aktivierung einer zusätzlichen Lichtquelle, das graphische HUD sowie die zahlenmäßige Darstellung der Relativinformationen von Chaser und Target und die Funktion für das Pausieren der Simulation.

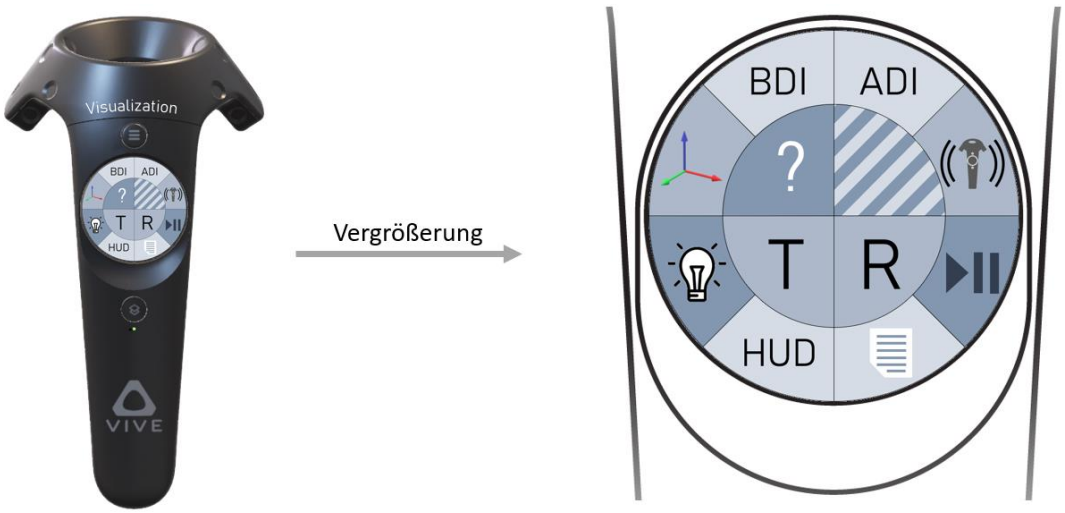

Abb. 5–9: Gesamtansicht des l. Controllers und Interaktionsoberfläche

Die implementierten Tasten werden in der Tabelle 5–2 mit der jeweiligen Position und Funktion übersichtsmäßig zusammenfasst. Sie werden insgesamt in drei Hauptgruppen geteilt und im Anschluss kurz erläutert.

Tab. 5–2: Tastenbelegung Interaktionsoberfläche des l. Controllers

Hauptgruppe	Tastenposition	Funktion
Visualisierung	Innenkreis unten links	Translatorische Kamerabewegung
	Innenkreis unten rechts	Rotatorische Kamerabewegung
Hilfssysteme Kamera-Steuerung	Außenring oben links, Segment 1	Neutralpunktanzeige
	Außenring oben links, Segment 2	Basic Deviation Indicator
	Außenring oben rechts, Segment 1	Advanced Deviation Indicator
	Außenring oben rechts, Segment 2	Haptic Feedback
	Außenring unten links, Segment 2	Graphisches HUD
	Außenring unten rechts, Segment 1	Relative Telemetriedaten
Sonstige Funktionen	Außenring unten links, Segment 1	Zusatzbeleuchtung
	Außenring unten rechts, Segment 2	Simulationsunterbrechung
	Innenkreis oben links	Dokumentationsmenü

Visualisierung

Die Hauptgruppe Visualisierung beinhaltet das Ein- und Ausschalten der Steuerungsfunktionen der Kameraperspektive. Wie schon beim rechten Controller sind die beiden Tasten in der unteren Hälfte des inneren Kreises der UI zuständig für die Aktivierung bzw. Deaktivierung der translatorischen und rotatorischen Bewegungseingaben für die Änderung der Kameraperspektive. Bei ihrer Betätigung werden die hinter den beiden Schalter befindlichen Zustandsvariablen gesetzt und die Abarbeitung der Eingabelogik ermöglicht. Durch erneutes Auswählen wird die Variable wieder zurückgesetzt und die Funktion deaktiviert.

Hilfssysteme Kamerasteuerung

In diese Hauptgruppe werden die Systeme, die dem Nutzer bei der variablen Steuerung der Kameraperspektive unterstützen sollen, zusammengefasst. Dazu gehört die Neutralpunktanzeige, die beiden Deviationsindikatoren, das haptisches Feedbacksystem und die Darstellung des graphischen HUDs sowie die Einblendung der relativen Geschwindigkeits- und Lageinformationen zwischen Chaser und Target. Die ersten vier Elemente sind dabei identisch zu den in der Satellitensteuerung implementierten Hilfssysteme und ihre jeweilige Aufgabe sowie die genaue Funktionsweise ist im Kapitel 5.3.1 bei der Erläuterung der einzelnen Komponenten der Interaktionsschnittstelle für die Satellitensteuerung erklärt. Die letzten beiden Komponenten wurden bereits innerhalb der vorausgehende Bachelorarbeit implementiert und ausgehend davon werden sie lediglich in die aktuelle Interaktionsschnittstelle integriert. Sie lassen sie folglich durch das Betätigen der entsprechend zugewiesenen Tasten ein- und ausblenden

Sonstige Funktionen

Im Rahmen der vorausgehende Arbeit wurden zu den vorher genannten Anzeigen weitere zwei Funktionen implementiert. Die erste Funktion ist auf der linken Seite des unteren äußeren Ringes mit einem Glühbirnensymbol dargestellt. Sie hat die Aufgabe die Beleuchtung des Targetsatellitens zu verbessern und lässt sich durch die Betätigung der Taste ein- und ausschalten. Mit Hilfe der zweiten Funktion auf der rechten Seite des unteren äußeren Ringes ist es möglich die RACOON Simulation anzuhalten und wieder zu starten. Die genaue Erklärung der beiden Funktionen ist aus [9] zu entnehmen. Zuletzt blendet die Funktion auf der oberen linken Seite des inneren Kreises bei Aktivierung eine kurze Erklärung der Schnittstelle für Funktionsauswahl des linken Controllers ein. Dabei verwendet sie analog zu der gleichen Funktion auf dem rechten Controller sowohl graphische, als auch textuelle Elemente.

5.4.2 Bewegung der Kameraperspektive im vollvariablen Modus

Neben den Funktionen und Hilfssysteme, die mit Hilfe des linken Controllers aktivieren lassen, ist der linke Controller für die Verschiebung der Kameraperspektive zuständig. Das Grundkonzept ist dabei, sich nicht auf verschiedenen vorherdefinierten Positionen zu beschränken, sondern die Möglichkeit sich in der Simulation frei bewegen zu können. Für die Umsetzung ist es notwendig, die Steuerungsbefehle, die mit Hilfe des Controllers eingeben werden, in das für die Verschiebung der Kameraperspektive verantwortliches Modul in der RACOON Simulation zu übertragen. Durch die Verschiebung der Kameraposition wird ein neues Bild von der neuen Perspektive generiert und anschließend für die Wiedergabe in der virtuellen Realität an das HMD weitergegeben.

Die Erfassung der Steuerungseingaben auf der Seite von *Unity* erfolgt identisch zu der im Kapitel 5.3.2.1 vorgestellten Steuerungsmethode. Dabei werden bei der Steuerung der Kameraperspektive nur die drei Parameter für die Translation ermittelt und für den Datentransfer zum Simulationsprogramm vorbereitet. Dementsprechend beinhaltet die Steuerung der Kameraperspektive lediglich die Veränderung ihrer Position. Die Datenübertragung geschieht mit Hilfe der in Kapitel 5.3.3 verwendeten IPC-Methode *Named Pipes*, bei der wieder das asynchrone Übertragungsverfahren zum Einsatz kommt. Die Weiterverarbeitung der Eingabedaten findet im Simulationsprogramm über das Unterprogramm „CameraInput.cs“ statt. Hier werden die Daten aus der *Named Pipe* herausgelesen und für die ausstehende Bestimmung der neuen Kameraposition vorbereitet. Die Ermittlung der neuen Ansichtsposition findet bei der Stereoansicht in „*VRCamera.cs*“ und bei der Monoansicht in „*VRCAMERAONEIMAGE.cs*“ statt. Dabei werden die Eingabedaten aus Unity mit den Positionsdaten der aktuellen Kameralage addiert und bildet somit die neue Position der Visualisierungsperspektive. Für die neue Generierung des Simulationsbildes aus der neuen Position, dessen Übertragung an das visuelle Ausgabegerät in VR sowie ihre Ausgabe wird die bereits in der VR-Schnittstelle vorhandenes Verfahren verwendet. Die ausführliche Beschreibung dieser Bearbeitungsprozesse ist in [9] unter den Kapiteln 5.2.4 bis 5.2.6 zu finden.

Bei der sequentiellen Abarbeitung des Simulationsprogrammes wird bei jedem Simulationsschritt die Kameraposition um die aus der Eingabe in *Unity* resultierende Verschiebung geändert und ein neues Bild ausgehend von der neuen Perspektive generiert. Zusammenfassend werden aus den einzelnen Verschiebung in der Summe den Eindruck einer fließende Bewegung des Kamerabildes erzeugt

6 Verifikation des Steuerungsprogramms

Nachdem die verschiedenen Konzepte in **Kapitel 4** vorgestellt wurden und eine Auswahl getroffen wurde, erfolgte in **Kapitel 5** die Erläuterung der gesamten Implementierung. In diesem Kapitel wird überprüft, ob die am Anfang der Arbeit definierten Anforderungen für das Steuerungskonzept sowie für die Überarbeitung der Nutzerschnittstelle in VR erfüllt wurden.

6.1 Verifikation des Steuerungskonzepts

Im Rahmen dieser Bachelorarbeit wurden zwei unterschiedliche Eingabekonzepte für die Nutzerinteraktion in VR entworfen und in die bereits vorhandene VR-Erweiterung integriert. Dabei beinhaltet das fertige Steuerungsmethode beide Eingabekonzepte, die in sich eine Symbiose bildet. In der Tabelle 6–1 sind alle Anforderungen für das Steuerungskonzept mit den jeweiligen Ergebnissen zusammengefasst.

Tab. 6–1: Ergebnis der Verifikation des Steuerungskonzeptes

Index	Anforderung	Ergebnis
RFC 1	Nutzerfreundlichkeit	Erreicht
RFC 2	Präzision	Erreicht
RFC 3	Portabilität	Erreicht

Die beiden Steuerungsmethoden sind auf Basis der technischen Rahmenbedingungen der HTC VIVE Controller entwickelt wurden. Bei der Pseudo-Gestensteuerung wird die Befehlseingabe durch die Bestimmung der momentanen Position sowie Orientierung des Controllers gegenüber des zuvor definierten Nullpunktes ermittelt. Dabei werden die Controller mit Hilfe von zwei Trackingstationen im Raum verfolgt und es ist damit möglich, von jede Stelle des überwachten Raumes die Steuerung durchzuführen. Die analoge Steuerungsmethode geht noch einen Schritt weiter. Bei dieser Steuerungsmethode ist das aktive Tracking des Controllers bei der Bedienung nicht notwendig ist und es wird für die Steuerung lediglich eine Verbindung zum VR-Headset für die Verarbeitung der Tasteneingaben benötigt. Somit erfüllen beide Eingabekonzepte die dritte Anforderung bzgl. der Portabilität.

Für die Anwendung in Bezug auf Nutzerinteraktion ist es wichtig, den Fokus bei der Entwicklung von Interaktionsmöglichkeiten auf die für Menschen natürlichen Aktionen zu legen. Die Analogsteuerung wurde basierend auf einen zweidimensionale Interaktionssystem aufgebaut und erreicht mit Hilfe der Tastenanordnung des VIVE Controllers weitestgehend die Anforderung einer intuitiven Eingabemethode. Lediglich die Rotation um die y-Achse ist aufgrund von Systemeinschränkungen nicht selbsterklärend und muss zuvor mit Hilfe einer Anleitung studiert werden. Die Pseudo-Gestensteuerung ist hingegen, wie schon der Name ankündigt, an die die natürlichen Bewegungen der Hand angeknüpft und ist auch ohne eine Einweisung schnell intuitiv erlernbar. Zusammenfassend erfüllt das implementierte Steuerungskonzept auch die erste Anforderung.

Die hauptsächliche Verwendung des Steuerungskonzeptes dient dazu die Position und Orientierung des Chasersatellitens für die Dockingmission zu manipulieren. Für diesen Zweck ist eine hohe Präzision der Eingabemethode gefordert. Wie bereits in Kapitel 4.3 erwähnt, ist die heutige VR-Technologie in Bezug auf räumliche Positionsbestimmung immer noch zu inakkurat. Mit Hilfe der definierten Deadzones ist es zumindest gewährleistet, dass es eine eindeutige Unterscheidung zwischen einer einachsigen und mehrachsigen Steuerungseingabe besteht. Die zusätzlich integrierte Analogsteuerung bietet in diesem Zusammenhang eine gute Genauigkeit. Die zweite Anforderung bzgl. Präzision ist daher in der Symbiose der beiden Steuerungsmethoden ebenfalls erfüllt.

6.2 Verifikation der Überarbeitung des VR-UI

Für die Neugestaltung der Nutzerschnittstelle in der virtuellen Realität wurden ebenfalls Anforderungen definiert. Sie sind in Tabelle 6–2 zusammen mit dem jeweiligen Ergebnis aufgelistet.

Tab. 6–2: Ergebnis der Verifikation der gesamten VR-UI

Index	Anforderung	Ergebnis
RUI 1	Nutzerfreundlichkeit	Erreicht
RUI 2	Modularität	Bedingt erreicht
RUI 3	Erweiterbarkeit	Bedingt erreicht
RUI 4	Multimodalität	Erreicht
RUI 5	Stabilität	Nicht erreicht

Die Erfüllung der ersten Anforderung der Nutzerfreundlichkeit ist an sechs zuvor definierten Unterpunkten geknüpft, die im nachfolgenden abgearbeitet werden.

In der überarbeiteten GUI wurden die Funktionstasten direkt auf dem Touchpad in Form von virtuelle Tasten implementiert. Das Touchpad ist somit vergleichbar mit einer Fernbedienung, mit der man verschiedenen Funktionen des jeweiligen Controllers ein- und ausschalten kann. Bei der Betätigung der einzelnen Tasten wird sowohl visuell eine Tastenbewegung nach unten, als auch ein virtuelles Klickgeräusch durch den Vibrationsmotor simuliert. Damit soll eine realitätsnahe Abbildung der Wahrnehmung simuliert werden, die ebenfalls bei der Betätigung einer realen Taste entstehen. Damit wird der erste Unterpunkt, die Übereinstimmung der Systemfunktionen mit der Realität, erfüllt. Mit Hilfe der zehn bzw. zwölf auf dem Touchpad befindlichen Tasten wird die Möglichkeit des Nutzers auf Selbstbestimmung erweitert. Die Funktionen und Hilfssysteme lassen sich, je nach den persönlichen Präferenzen des Nutzers, individuell anpassen. Der Vorteil von dem implementierten System ist, dass die Einstellungen bei einem Neustart von der Szene in *Unity* automatisch auf die standardmäßige Einstellung zurückgesetzt werden. Somit wird eine gute Balance zwischen Nutzerfreiheit und -einschränkung getroffen. Bei der Entwicklung und Festlegung der Tastenpositionen wurde eine strikte Konvention bzgl. der Eindeutigkeit und der Aufbau von Standards Konvention eingehalten. Bspw. besitzt die obere Hälfte linken und rechten Interaktionsfläche bis auf die Taste für das Wechsel der

Steuerungsmethode den identischen graphischen Aufbau und die gleichen Funktionen. Dazu wurde die Beschriftung der Tasten so minimalistisch gehalten wie möglich. Sie bestehen aus jeweils drei Buchstaben als Abkürzung für die jeweilige Funktion oder einer Graphik für den eindeutige Zuordnung. Die Fehlertoleranz des Systems wurde mit Hilfe der Definition verschiedener Deadzones erreicht und verringert die Wahrscheinlichkeit einer unbeabsichtigten Eingabe, weil damit z.B. der Spielraum um den Nullpunkt für die Steuerungseingabe vergrößert wurde.

Nach der Erfüllung der unterschiedlichen Unterkomponenten ist RUI 1 ebenfalls erfüllt.

Das Interaktionsmenü besitzt einen modularen Aufbauprinzip und ist mit Hilfe der Integration des Plugins für die Erstellung der unterschiedlichen Tasten einfach erweiterbar. Somit erfüllt es die Anforderung bzgl. der Modularität. Es ist jedoch auf der obersten Menüebene sehr gut ausgefüllt und besitzt dementsprechend über einer geringe Anzahl an Erweiterungsmöglichkeiten auf dieser Menüebene. Das Interface lässt sich aufgrund des menüartigen Aufbauprinzips beliebig nach unten hin erweitern. Zusammengefasst erfüllt die Nutzerschnittstelle daher nur bedingt die Anforderung der Erweiterbarkeit, da die Funktionserweiterung auf der obersten Ebene ohne Verschiebung anderer Funktionen nur begrenzt möglich ist.

Die Anforderung ist erfüllt, weil die Interaktionsschnittstelle neben dem visuellen Feedback beim Betätigen der Funktionstasten oder bei einer Auslenkung des Controllers für die Steuerungseingabe ebenfalls die haptische Wahrnehmungsfähigkeit angesteuert.

Bei der Erstellung einer VR Umgebung ist es darauf zu achten, dass die Immersion erhalten bleibt. Bei der variablen Verschiebung der Kameraposition ist es unter gewisse Umstände möglich, dass im HMD ein flimmerndes Bild angezeigt wird. Damit kann die Immersion nicht aufrechterhalten werden und somit erfüllt das Gesamtsystem nicht die Anforderung der Stabilität.

7 Diskussion

Die Verifizierung im letzten Kapitel hat gezeigt, dass die Anforderungen, die im Kapitel 3 aufgestellt wurden, größtenteils erfüllt worden sind. Die Hauptaufgabe der Bachelorarbeit war die Erweiterung der bereits vorhandenen VR-Erweiterung der RACOON-Lab Simulation mit einem Steuerungselement. Sie wurde erfolgreich implementiert und das VR-System verfügt auf dem jetzigen Stand sowohl über die Fähigkeit zur Veränderung der Kameraperspektive, als auch die direkte Steuerung des Chasersatellitens. Um alle Funktionen für die jeweiligen Operationen unterzubringen, wurde die Nutzerschnittstelle der beiden Controller überarbeitet und die Bewegungsmöglichkeit der Kameraperspektive um einen vollvariablen Modus erweitert, damit mit Hilfe der freien Positionierungsmöglichkeit die beste Visualisierung der Gesamtsituation erreicht werden kann.

Die Anforderung bzgl. der Stabilität und Systemleistung ist das einzige Ziel, das innerhalb der in Kapitel 2 definierten Liste nicht eingehalten wurde. Dabei ist diese Anforderung für die VR-Technik besonders wichtig, weil eine virtuelle Umgebung nur solange aufrechterhalten werden kann, bis die Immersion gebrochen wird. Durch die variable Steuerungsmöglichkeit der Kameraperspektive beginnen die Bilder, die im VR-Headset angezeigt werden, zu flimmern. Dieses Ereignis hat zwar keine Auswirkung auf die Funktionalität der sonstigen implementierten Funktionen und Hilfssysteme, es ist jedoch stark genug um die Qualität des gesamten Programmes zu mindern. Darüber hinaus ist das Dauerflimmern eine zusätzliche Belastung für den Nutzer und führt zu schnelleren Ermüdungserscheinungen.

Der Grund für die fehlerhafte Bildübertragung wurde bei der Beendigung der Arbeit nicht identifiziert. Sie wurde jedoch auf einen bestimmten Bereich des Bildbearbeitungsprozesses eingeschränkt. Der Bereich erstreckt sich vom Versenden der aktuellen Positionsdaten aus *Unity* über die Generierung der Bilddaten bezogen auf die aus *Unity* abgesendete Position bis zum abschließenden Übertragung der fertiggerenderten Bilder auf das HMD des VR-Headsets. Dabei ist der Fehler am wahrscheinlichsten im ersten Bereich des zuvor erläuterten Bereichs, aufgrund dessen, dass der nachfolgender Bereich im Rahmen der Arbeit nicht modifiziert wurde.

Zusätzlich wurde die Erweiterbarkeit der Bedienoberfläche nur bedingt erfüllt, weil die oberste Ebene des Interaktionsmenüs nur noch über einen bis maximal drei freie Tastenfelder für etwaige Funktionserweiterungen verfügt. Zwar lassen sich das Interaktionsmenü hierarchisch nach unten gesehen mit beliebig viele Funktionsbausteine ausstatten, jedoch ist eine Verschachtelung von Funktionselementen wegen der Verschlechterung der Übersichtlichkeit zu vermeiden. Eine mögliche Lösung ist das Entfernen redundanter Schalterfunktion auf dem linken und rechten Controller, bei der aber auf der anderen Seite die Unabhängigkeit der beiden Controller nicht mehr gewährleistet werden kann. Letztendlich ist nach der Implementierung der Steuerungskomponente das VR-System zum größten Teil fertiggestellt und es werden keine größere Menge an zusätzlichen Funktionen implementiert.

8 Zusammenfassung

Im Rahmen der Bachelorarbeit wurde die bereits vorhandene VR-Erweiterung der RACOON-Lab Simulation mit einer Steuerungskomponente erweitert. Das Ziel dabei war, für die Steuerungseingabe eine geeignete Eingabemethode in der virtuellen Realität zu entwickeln und sie anschließend in eine neue Nutzerschnittstelle zu integrieren.

Um eine Steuerungskomponente für die Nutzerinteraktion in der virtuellen Realität entwickeln zu können, ist es erforderlich die grundlegende Funktionsweise von VR zu verstehen. Dazu ist es notwendig die menschlichen Informations-verarbeitungsprozesse zu analysieren und daraus die Anforderungen an die Steuerungsmethode und die der Nutzerschnittstelle zu ermitteln. Auf Basis dieser Rahmenbedingungen wurden zwei unterschiedlichen Eingabekonzepte entwickelt. Die erste Methode basiert auf die analoge Tastensteuerung und die Eingabe erfolgt über festimplementierten Tasten auf dem *HTC VIVE* Controller. Die zweite Methode ist eine Abwandlung der Gestensteuerung und erfasst für die Befehlseingabe die relative Positionsänderung des Controllers gegenüber eines virtuellen Nullpunktes. Für die Steuerung es Chasersatelliten wurden alle beide Steuerungsmethoden implementiert. Es wurden verschiedene Hilfssysteme für die Unterstützung des Nutzers bei der Steuerungseingabe entwickelt und alle Funktionen, die für die Satellitensteuerung zuständig sind, im rechten Controller zusammengefasst. Die Verwaltung der einzelnen Funktionen erfolgt dabei direkt über die auf dem Touchpad implementierten Nutzerschnittstelle.

Zusätzlich zur Steuerung des Chasersatellitens wurden alle Komponenten für die Manipulation der Kamerakomponenten sowie andere wichtige Funktionen, die bereits in der vorherigen Nutzerschnittstelle integriert waren und dem Anwender bei der Navigation innerhalb der Simulation unterstützen soll, im linken Controller mit Hilfe einer neu entwickelten Interaktionsoberfläche zusammengefasst. Dabei bietet der linke Controller dem Nutzer die Möglichkeit, die Kameraperspektive im virtuellen Raum mit Hilfe der abgewandelten Gestensteuerung frei zu bewegen um somit den Überblick über die Gesamtsituation zu verbessern.

Das gesamte VR-System wurde zusammen mit dem RACOON-Simulationsprogramm getestet mit dem Ergebnis, dass die Anforderungen größtenteils erfüllt worden sind. Die Steuerung des Chasersatellitens ist vollständig funktionstüchtig und alle sonstigen Funktionen erfüllen ihre vorgesehenen Aufgaben. Lediglich die Stabilität des Gesamtsystems erreicht nicht ihre geforderte Spezifikation. Bei der freien Verschiebung der Kameraperspektive kann es vorkommen, dass das Bild anfängt zu flimmern. Für die Behebung dieses Problems muss das Bilderstellungsprozess im Simulationsprogramm sowie die Schnittstellen bei der Bilderzeugung zwischen den beiden Programmen nochmal überarbeitet werden.

9 Ausblick

Die VR-Erweiterung des Simulationsprogrammes des RACOON-Labs bildet nach dem Abschluss der Integration der Steuerungskomponente ein vollwertiges Steuerungselement für die Durchführung der im Rahmen des RACOON Reference Scenario definierten Weltraumteleoperationen. Dabei bietet sich die Durchführung verschiedenster Nutzerstudien an, um bspw. zu ermitteln, wie sich die Leistung des Operators mit Hilfe der VR-Steuerungskomponente im Vergleich zu der Eingabemethode durch einen Dreifreiheitsgradenjoystick verhält.

Für die Instandhaltung des VR-Programmes ist eine regelmäßige Aktualisierung der verschiedenen Assets bzw. Plugins in *Unity* notwendig. Das VR-Programm wurde mit Hilfe des SteamVR Plugins mit der Versionsnummer 1.2.3 durchgeführt, wobei die aktuelle Version beim Abschluss der Bachelorarbeit 2.0 ist. In der aktuelle Version des Plugins wurde die Programmiermethode grundlegend verändert und die Zuweisung der entsprechenden Tasten auf dem *HTC VIVE* Controller wird mit Hilfe einer graphischen Oberfläche realisiert. Dies erfordert die komplette Umstellung des Programmes auf die graphisch gehaltene Zuweisungsmethode und würde die komplette Rekonstruktion des bereits fertiggestellten Programmes bedeuten.

Neben der von der NASA bereits eingesetzte VR-Technologie im Bereich der Astronautentraining, ist mit der zunehmende Entwicklung der VR-Technologie auch denkbar eine Weltraummission damit durchzuführen. Zusammen mit ihrer Schwestertechnologie AR wäre es denkbar, komplexe Aufbauten oder Steuerungspulte virtuell nachzubauen und auf die kostspielige Hardware gänzlich zu verzichten. Dies hat den Vorteil, dass neue Technologien schneller in Anwendungen integriert werden können, da lediglich die virtuelle Abbildung als Modell in VR/AR benötigt wird.

A Literaturverzeichnis

[1] NASA, *NASA's Great Observatories.* [Online] Available: https://www.nasa.gov/audience/forstudents/postsecondary/features/F_NASA_Great_Observatories.html. Accessed on: Oct. 22 2018.

[2] NASA Goddard Space Flight Center, *SM1.* [Online] Available: https://asd.gsfc.nasa.gov/archive/hubble/missions/sm1.html. Accessed on: Oct. 22 2018.

[3] R. B. Loftin *et al.,* "Virtual Environments in Training: NASA's Hubble Space Telescope Mission," *16th Interservice/Industry Training Systems & Education Conference*, vol. 1994, 1994.

[4] D. Silverman, *NASA's virtual reality journey uses same software, hardware as gamers.* [Online] Available: https://www.houstonchronicle.com/techburger/article/NASA-s-virtual-reality-journey-uses-same-12745239.php#photo-15147154. Accessed on: Oct. 22 2018.

[5] B. Mader and E. Miralles, "ISS Onboard Virtual Reality Trainer (VRT),"

[6] K. Sloan, *How NASA Trains Astronauts with Unreal Engine.* [Online] Available: https://www.unrealengine.com/en-US/blog/how-nasa-trains-astronauts-with-unreal-engine. Accessed on: Oct. 22 2018.

[7] M. Sagardia *et al., VR-OOS: The DLR's Virtual Reality Simulator for Telerobotic On-Orbit ServicingWith Haptic Feedback*. Piscataway, NJ: IEEE, 2015.

[8] J. Harder, M. Dziura, and S. Haberl, "Future Technologies for Operating Robots in Space," *IAC*, 2017.

[9] T. Landauer, "Implementierung einer Nutzerschnittstelle zur Visualisierung der Annäherung von zwei Raumfahrzeugen in virtueller Realität," Bachelorarbeit, Lehrstuhl für Raumfahrttechnik, Technische Universität München, München, 2017.

[10] Canadian Space Agency, *Canadarm2's data sheet.* [Online] Available: http://www.asc-csa.gc.ca/eng/iss/canadarm2/data-sheet.asp. Accessed on: Oct. 26 2018.

[11] N. J. Currie and B. Peacock, "International Space Station Robotic Systems Operations - a Human Factors Perspective," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 46, no. 1, pp. 26–30, 2002.

[12] Canadian Space Agency, *Canadarm2 catches the Dragon resupply ship.* [Online] Available: http://www.asc-csa.gc.ca/eng/search/images/watch.asp?id=4566. Accessed on: Oct. 26 2018.

[13] Metecs, *Flight Software Development.* [Online] Available: http://www.metecs.com/services-software.php. Accessed on: Oct. 26 2018.

[14] NASA, *A SAFER Way to Space Walk.* [Online] Available: https://www.nasa.gov/missions/shuttle/f_saferspacewalk.html. Accessed on: Oct. 24 2018.

[15] M. Garcia, *Virtual Reality Training and Global Robotics Work Before Spacewalk.* [Online] Available: https://blogs.nasa.gov/spacestation/2018/05/10/virtual-reality-training-and-global-robotics-work-before-spacewalk/. Accessed on: Oct. 22 2018.

[16] A. Gerndt, *Virtuelle Reality für On-Orbit Servicing (VR-OOS).* [Online] Available: https://www.dlr.de/sc/de/desktopdefault.aspx/tabid-6353/10412_read-22817/. Accessed on: Oct. 23 2018.

[17] T. Hulin *et al.,* "The DLR Bimanual Haptic Device with Optimized Workspace," (eng), pp. 3441–3442, May. 2011.

[18] DLR, *SpaceJustin.* [Online] Available: https://www.dlr.de/rm/desktopdefault.aspx/tabid-8749/. Accessed on: Oct. 23 2018.

[19] J. Harder, *Scenario Visualization and Spacecraft control: Visualization Usage.* [Online] Available: https://wiki.tum.de/pages/viewpage.action?pageId=16584431. Accessed on: Oct. 24 2018.

[20] J. Harder, *RACOON Visualization Keybinding.* [Online] Available: https://wiki.tum.de/display/racoon/Visualization+Keybindings. Accessed on: Oct. 24 2018.

[21] M. Wilde, "Visual Augmentation Methods for Teleoperated Space Rendezvous," Dissertation, Lehrstuhl für Raumfahrttechnik, Technische Universität München, München, 2012.

[22] J. Harder, M. Wilde, J. Ventura, and M. Dziura, *Acoustic Telepresence for Spacecraft Proximity Operations*. Piscataway, NJ: IEEE, 2016.

[23] M. Dziura, "Racoon Reference Scenario," Missionsreferenzdokument, Lehrstuhl für Raumfahrttechnik, Technische Universität München, München, 2015.

[24] A. Hug, "Entwicklung und Implementierung einer Nutzerschnittstelle für teleoperierte Inspektionen von Satelliten in Virtual Reality," Bachelorarbeit, Lehrstuhl für Raumfahrttechnik, Technische Universität München, München, 2018.

[25] D. Korgel, *Virtual Reality Spiele entwickeln mit Unity*. München: Carl Hanser Verlag München, 2018.

[26] R. Dörner, W. Broll, P. Grimm, and B. Jung, *Virtual und Augmented Reality (VR / AR)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[27] N. van Ackern and M. Lindenberg, "Räumliches Hören," Seminararbeit, Uni Mannheim, Mannheim.

[28] HTC Corp. and Valve Corp., *Vive PRE User Guide*.

[29] J. Abaigar, *Vive Controller.* [Online] Available: https://sketchfab.com/models/9f03e4a80c5a4b31a24bb122f17cb229?ref=related . Accessed on: Oct. 24 2018.

[30] Eternal Realm, *HTC Vive.* [Online] Available: https://sketchfab.com/models/4cee0970fe60444ead77d41fbb052a33. Accessed on: Sep. 27 2018.

[31] J. Nielsen, “Enhancing the Explanatory Power of Usability Heuristics,” (eng), *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, vol. CHI '94, pp. 152–158, http://dl.acm.org/citation.cfm?id=191666, 1994.

[32] A. Harley, *Visibility of System Status.* [Online] Available: https://www.nngroup.com/articles/visibility-system-status/.

[33] A. Kaley, *Match Between the System and the Real World: The 2nd Usability Heuristic Explained.* [Online] Available: https://www.nngroup.com/articles/match-system-real-world/. Accessed on: Oct. 26 2018.

[34] A. Biagioli, *Vive-Virtual-Button-Designer*: Flafla2, 2017.

[35] Unity Technologies, *Rotation and Orientation in Unity.* [Online] Available: https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html. Accessed on: Oct. 30 2018.

[36] M. Lust, *Quarternionen - mathematischer Hintergrund und ihre Interpretation als Rotationen*, 2001.