DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Pepper VR – Teleoperation eines humanoiden Roboter auf Basis der Analyse menschlicher Bewegung

Studienarbeit

für die Prüfung zum

Bachelor of Science

des Studienganges Informatik / Angewandte Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Matthias Schuhmacher und Marlene Rieder

Abgabedatum 20. Mai 2024

Bearbeitungszeitraum	300 Stunden
Matrikelnummer	4128647 und 8261867
Kurs	tinf21b3 und tinf21b5
Gutachter der Studienakademie	Prof. Dr. Marcus Strand

Erklärung

Wir versicheren hiermit, dass wir unsere Studienarbeitmit dem Thema: »Pepper VR – Teleoperation eines humanoiden Roboter auf Basis der Analyse menschlicher Bewegung« selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versicheren zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort Datum

Unterschrift

Zusammenfassung

Das Ziel der vorliegenden Studienarbeit ist es, eine mögliche Verbindung zwischen einer Virtual-Reality Brille und dem humanoiden Roboter Pepper herzustellen. Das Kamerabild des Roboters soll auf der Brille angezeigt werden, ebenfalls soll es möglich sein, den Roboter mit Hilfe der Controller der Brille zu steuern.

Inhaltsverzeichnis

Abkürzungsverzeichnis

Kapitel 1

Einleitung

Im Umfeld der Duale Hochschule Baden-Württemberg (DHBW) Karlsruhe werden immer wieder Projekte im Bereich der Robotik an Studierende im Rahmen verschiedener Arbeiten vergeben. Ebenso dieses Projekt, welches im Rahmen der Studienarbeit an die beiden Studierenden Matthias Schuhmacher und Marlene Rieder übergeben wurde. =======
Im Umfeld der DHBW Karlsruhe werden immer wieder Projekte im Bereich der Robotik an Studierende im Rahmen verschiedener Arbeiten vergeben. Ebenso dieses Projekt, welches im Rahmen der Studienarbeit an die beiden Studierenden Matthias Schuhmacher und Marlene Rieder übergeben wurde.

In unserer immer digitaler werdenden Welt spiele sowohl Robotik als auch Virtual Reality (VR) eine immer größer werdende Rolle, diese zwei Welten sollen durch das, im folgenden beschriebene Projekt verbunden werden.

Ziel dieses Projektes ist es den humanoiden Roboter Pepper mit Hilfe der VR Brille Meta Quest 3 zu steuern. Dies soll ermöglicht werden, indem das Kamerabild des Roboters an die Brille übertragen wird und sich die Bewegungen des Roboters durch die Controller der Brille steuern lassen. Der Nutzer soll also sehen, was sich vor dem Roboter befindet und ihn dann steuern können.

Kapitel 2

Grundlagen

2.1 Pepper

Pepper ist ein humanoider Roboter, der entwickelt wurde, um die Gefühle und Gesten von Menschen zu analysieren und basierend auf diesen, darauf zu reagieren. Das Projekt entstand durch eine Zusammenarbeit des französischen Unternehmens Aldebaran Robotics Société par actions simplifiée (SAS) und des japanischen Telekommunikations- und Medienkonzerns SoftBank Mobile Corp. Ziel diese Projektes war es, einen humanuiden "Roboter-Gefährten" oder einen "persönlichen Roboter-Freund" zu schaffen, der zunächst im Gewerbesektor in Verkaufsräumen, an Empfangstischen oder in Bildungs- und Gesundheitseinrichtungen eingesetzt werden sollte. Die Produktion wurde jedoch aufgrund geringer Nachfrage bis auf Weiteres pausiert.

Das Konzept von Pepper distanziert sich von herkömmlichen Industrierobotern und reinen Spielzeugrobotern, indem er als informativer und kommunikativer Begleiter konzipiert wurde. Sein Aussehen, das im etwa an die Größe eines Kindes angelehnt ist, sowie ein freundliches Gesicht und eine kindliche Stimme sind im ästhetischen Konzept von "kawaii" (japanisch für "niedlich" oder auch "liebenswert") gehalten.

Pepper wurde im Rahmen einer Präsentation am 5. Juni 2014 als der "erste persönliche Roboter der Welt mit Emotionen" vorgestellt. Die Vermarktung begann damit, dass SoftBank Pepper-Geräte in ihren Verkaufsräumen einsetzte, um Kunden zu unterhalten und zu informieren. Die Roboter sollte dabei den Umgang mit Kunden erlernen, um zukünftige Anwendungsmöglichkeiten zu erforschen. Verkauft wurde offiziell ab dem 3. Juli 2015 zu einem Preis von 198.000 Yen pro Einheit, zuzüglich monatlicher Gebühren für Zusatzleistungen. Im Laufe der Zeit wurde Pepper auch für den Einsatz in weiteren Unternehmen und Einrichtungen verfügbar gemacht.

Pepper wird mit einer Grundausstattung an Anwendungen geliefert, jedoch sind für spezifische Anwendungen, individuell entwickelte Softwarelösungen erforderlich wie auch zum Beispiel in diesem. SoftBank ermöglichte unabhängigen Entwicklern durch die Veröffentlichung der Schnittstellen den Zugang zu einem Interface für Applikationsprogramme, um zusätzliche Anwendungen für Pepper zu erstellen. Das NAOqi-Framework welches für diesen Nutzen bereitgestellt wurde, beinhaltet eine Application Programming In-

terface (API), eine Software Development Kit (SDK) und weitere Tools, welche in den Sprachen Python und C++ uneingeschränkten Zugriff auf die Komponenten, Sensoren und Aktoren des Roboters bieten, dazu später Ausführlicheres in Abschnitt 2.2. Mit Hilfe diese Interfaces haben verschiedene Unternehmen integrierte Lösungen entwickelt, die Pepper beispielsweise bei der Kundenberatung unterstützen können.

Das Design von Pepper ist dem Menschen ähnlich und umfasst einen Kopf mit integrierten Mikrofonen und Kameras sowie einen Torso mit weiteren Sensoren für Stabilität und Sicherheit. Der Roboter verfügt über verschiedene Mechaniken, die es ihm ermöglichen, sich flüssig zu bewegen und mit Personen zu interagieren. Durch die Verwendung von Kameras und bereitgestellter Software ist Pepper in der Lage, Emotionen bei seinen Gesprächspartnern zu erkennen und darauf zu reagieren, obwohl er selbst keine Mimik besitzt. Sicherheitsvorkehrungen wie Abstandssensoren und Stabilisatoren gewährleisten einen sicheren Einsatz von Pepper in verschiedenen Umgebungen. Diese können jedoch bedingt durch den Entwickler deaktiviert werden, um den Roboter in komplexeren oder laborähnlichen Umgebungen zu betreiben.

2.2 NAOqi

2.2.1 Definition

NAOqi ist die Bezeichnung für die Hauptsoftware, die auf dem Pepper Roboter ausgeführt wird und ihn intern steuert. Das NAOqi Framework ist das Programmiergerüst, welches zur Programmierung von NAO und Pepper Robotern verwendet wird. Es implementiert alle allgemeinen Anforderungen der Robotik, einschließlich: Parallelität, Ressourcen-Management, Synchronisation und Ereignisse. Dieses Framework ermöglicht eine homogene Kommunikation zwischen verschiedenen Modulen wie etwa die Bewegung, Audio oder Video sowie eine homogene Programmierung und einen homogenen Informationsaustausch. Das Framework ist:

- plattformübergreifend, d.h. es ist möglich, damit auf Windows, Linux oder Mac zu entwickeln. Genaueres dazu im Abschnitt 2.2.1.
- sprachübergreifend, mit einer identischen API für C++ und Python. Weitere Details dazu sind in Abschnitt 2.2.1 aufgeführt.
- bereit für Introspektion, was bedeutet, dass das Framework weiß, welche Funktionen in den verschiedenen Modulen verfügbar sind und wo. Für Details diesbezüglich siehe Abschnitt 2.2.1.

Sprachübergreifend

Software kann in C++ und Python entwickelt werden. Eine Übersicht über die Sprachen selbst in den Abschnitten Unterabschnitt 2.4.1 und Unterabschnitt 2.4.2. In allen Fällen sind die Programmiermethoden genau die gleichen, alle vorhandenen APIs können unabhängig von den unterstützten Sprachen aufgerufen werden:

- Wird ein neues C++-Modul erstellt, können die C++-API-Funktionen von überall aus aufgerufen werden,
- Sind sie richtig definiert, können auch die API-Funktionen eines Python-Moduls von überall aus aufgerufen werden.

In der Regel werden die Verhaltensweisen in Python und Ihre Dienste in C++ entwickelt.

Introspektion

Die Introspektion ist die Grundlage der Roboter-API, der Fähigkeiten, der Überwachung und der Maßnahmen bei überwachten Funktionen. Der Roboter selbst kennt alle verfügbaren API-Funktionen. Wird eine Bibliothek entladen, werden die entsprechenden API-Funktionen automatisch ebenfalls entfernt. Eine in einem Modul definierte Funktion kann der API mit einem `BIND_METHOD` hinzugefügt werden.

Wird eine Funktion gebunden, werder automatisch folgende Funktionen ausgeführt:

- Funktionsaufruf in C++ und Python, wie in Abschnitt 2.2.1 beschrieben
- Erkennen der Funktion, wenn sie gerade ausgeführt wird
- Funktion lokal oder aus der Ferne, z.B. von einem Computer oder einem anderen Roboter, ausführen weiter im Detail beschrieben in Abschnitt 2.2.1
- Generierung und Aufruf von `wait`, `stop`, `isRunning` in Funktionen

Die API wird im Webbrowser angezeigt wenn auf das Gerät per Uniform Resource Locator (URL) oder Internet Protocol (IP)-Addresse auf dem Port 9559 zugegriffen wird. In dieser Übersicht, zeigt der Roboter seine Modulliste, Methodenliste, Methodenparameter, Beschreibungen und Beispiele an. Der Browser zeigt auch parallele Methoden an, die überwacht, zum Warten veranlasst und gestoppt werden können.

Die Introspektion und derer Implementation im NAOqi-Framework, ist also ein mächtiges Werkzeug, welches es ermöglicht, die Roboter-API zu verstehen und zu verwenden aber auch zu überwachenund zu steuern.

Verteilter Baum und Kommunikation

Eine Echtzeitanwendung kann aus einer einzelnen ausführbaren Datei oder einem Baum von mehreren Systemen wie etwa Robotern, Prozessen oder Modulen bestehen. Unabhängig davon sind die Aufrufmethoden immer dieselben. Eine ausführbare Datei kann durch eine Verbindung mit einem anderen Roboter mit IP-Adresse und Port verbunden werden, sodass alle API-Methoden von anderen ausführbaren Dateien sind auf die gleiche Weise verfügbar sind, genau wie bei einer lokalen Methode. NAOqi trifft dabei selbst die Wahl zwischen schnellem Direktaufruf Local Procedure Call (LPC) und Fernaufruf Remote Procedure Call (RPC).

Unterstützte Betriebssysteme

2.2.2 NAOqi Vorgehensweise

Die NAOqi Software, welche auf dem Roboter läuft, ist ein Broker. Wenn dieser startet, lädt er eine Voreinstellungsdatei in den Speicher, in der festgelegt ist, welche Bibliotheken in dieser Konfiguration geladen werden sollen. Jede Bibliothek enthält ein oder mehrere Module, die den Broker benutzen, um ihre Methoden bereitzustellen.

Der Broker selbst bietet Nachschlagdienste an, so dass jedes Modul im Baum oder im Netzwerk jede Methode finden kann, die an dem Broker bekannt gegeben wurde.

Das Laden von Modulen bildet dann einen Baum von Methoden, die mit Modulen verbunden sind, und von Modulen, die mit dem Broker verbunden sind.

2.2.3 Broker

2.3 Robot Operating System

2.4 Programming Languages

2.4.1 C++

2.4.2 Python

2.5 Virtual Reality

Virtual Reality (VR) hat sich in den letzten Jahrzehnten von einem visionären Konzept zu einer greifbaren Technologie entwickelt, die in zahlreichen Bereichen unseres Lebens Anwendung findet. VR bezeichnet computergenerierte, dreidimensionale Umgebungen, die durch spezielle Hardware wie VR-Brillen und Bewegungssensoren eine immersive Erfahrung ermöglichen. Benutzer können in diese virtuellen Welten eintauchen und mit ihnen interagieren, was das Gefühl vermittelt, tatsächlich dort anwesend zu sein.

Die Wurzeln der VR-Technologie reichen bis in die 1960er Jahre zurück. Einer der frühesten Vorläufer moderner VR-Systeme war das Sensorama, das 1962 von Morton Heilig entwickelt wurde. Das Sensorama bot multisensorische Erlebnisse und kann als eines der ersten Systeme betrachtet werden, das den Nutzer vollständig in eine künstliche Umgebung eintauchen ließ [Heilig 1962]. Ein weiterer bedeutender Meilenstein war das SSword of Damocles", das erste echte VR-Headset, das 1968 von Ivan Sutherland entwickelt wurde. Es war ein klobiges Gerät, das an der Decke montiert werden musste, und bot eine rudimentäre Form der virtuellen Realität [Sutherland 1968].

In den 1980er und 1990er Jahren erlebte VR einen weiteren Aufschwung, insbesondere durch die Arbeit von Jaron Lanier, der den Begriff „Virtual Reality“ populär machte und die Firma VPL Research gründete, die einige der ersten kommerziellen VR-Produkte

entwickelte [Lanier 1992]. Trotz dieser Fortschritte blieb VR lange Zeit eine Nischentechnologie, hauptsächlich aufgrund der hohen Kosten und technischen Einschränkungen.

Erst in den 2010er Jahren, mit der Einführung moderner, kostengünstigerer VR-Headsets wie der Oculus Rift, entwickelte sich VR zu einer breiter zugänglichen Technologie. Die Oculus Rift, ursprünglich 2012 auf Kickstarter finanziert, revolutionierte den Markt und führte zu einer neuen Welle von Innovationen in der VR-Technologie [Luckey 2012]. Kurz darauf folgten andere bedeutende Systeme wie die HTC Vive und PlayStation VR, die die VR-Erfahrung weiter verbesserten und breitere Zielgruppen erreichten.

Die heutige VR-Technologie zeichnet sich durch hochauflösende Displays, präzises Tracking und eine Vielzahl von Eingabemethoden aus, die eine immersive und interaktive Erfahrung ermöglichen. Meta Quest 3 (ehemals Oculus Quest 3) ist ein Beispiel für ein modernes, eigenständiges VR-Headset, das keine Verbindung zu einem leistungsstarken PC benötigt und dennoch eine beeindruckende Leistung bietet [Meta 2023].

Die Anwendungsbereiche von VR sind vielfältig und umfassen nicht nur Unterhaltung und Spiele, sondern auch Bildung, medizinische Therapie, Training und Simulationen, Architektur und Design sowie viele andere Felder. Zum Beispiel wird VR in der Medizin zur Behandlung von Phobien, in der Schmerztherapie und in der Rehabilitation eingesetzt [Rizzo und Koenig 2017]. In der Ausbildung ermöglicht VR realitätsnahe Trainingsumgebungen, die sicher und kontrolliert sind, was insbesondere in der Luftfahrt und der Medizin von großem Nutzen ist [Huang und Liaw 2018].

Die rapide Weiterentwicklung von Hardware und Software sowie die steigende Akzeptanz und Integration von VR in verschiedenen Lebensbereichen zeigen, dass diese Technologie das Potenzial hat, unsere Interaktion mit digitalen Inhalten und unsere Wahrnehmung von Realität grundlegend zu verändern. Die folgenden Abschnitte dieser Arbeit werden die technologischen Grundlagen von VR, verschiedene Anwendungsbereiche sowie die aktuellen Herausforderungen und Zukunftsaussichten der Technologie detailliert untersuchen.

2.5.1 C#

C# (ausgesprochen "C-Sharp") ist eine moderne, objektorientierte Programmiersprache, die von Microsoft im Jahr 2000 als Teil seiner .NET-Initiative entwickelt wurde. Die Sprache wurde unter der Leitung von Anders Hejlsberg entwickelt und ist stark von anderen populären Sprachen wie C, C++ und Java inspiriert. C# ist für seine Vielseitigkeit, Robustheit und Benutzerfreundlichkeit bekannt und wird in einer Vielzahl von Anwendungen eingesetzt, von Desktop- und Web-Anwendungen bis hin zu mobilen Apps und Spielen, insbesondere in der Entwicklung mit der Unity-Engine.

2.5.2 Syntax und Grundstruktur

Die Syntax von C# ist einfach und klar, was es für Entwickler leicht macht, die Sprache zu erlernen und zu verwenden.

2.5.3 Objektorientierte Programmierung (OOP)

C# unterstützt die vier Hauptprinzipien der objektorientierten Programmierung:

1. **Abstraktion**: Das Verbergen komplexer Implementierungsdetails und das Zeigen nur der notwendigen Eigenschaften eines Objekts.
2. **Kapselung**: Das Zusammenfassen von Daten und Methoden, die auf diese Daten zugreifen, innerhalb einer Klasse und das Verbergen der Details der Implementierung vor anderen Klassen.
3. **Vererbung**: Die Fähigkeit einer Klasse, die Eigenschaften und Methoden einer anderen Klasse zu erben, was Code-Wiederverwendung und Hierarchien ermöglicht.
4. **Polymorphismus**: Die Fähigkeit, eine Methode auf verschiedene Weise zu implementieren oder zu überschreiben, was eine flexible und dynamische Nutzung von Methoden ermöglicht.

2.5.4 Vorteile von C#

C# bietet mehrere Vorteile, die es zu einer beliebten Wahl für Entwickler machen:

- **Einfach zu erlernen**: Die klare Syntax und die umfangreiche Dokumentation machen C# zu einer anfängerfreundlichen Sprache.
- **Leistungsfähig und flexibel**: C# ist leistungsfähig genug für komplexe Anwendungen und flexibel genug für schnelle Entwicklung.
- **Große Community und Ressourcen**: Die große Entwicklergemeinschaft und die Fülle an Online-Ressourcen, Tutorials und Foren machen es einfach, Unterstützung zu finden und Wissen auszutauschen.
- **Integrierte Entwicklungsumgebungen (IDEs)**: Tools wie Visual Studio und Visual Studio Code bieten leistungsstarke Entwicklungsumgebungen mit Debugging- und Code-Analyse-Funktionen.

2.6 Virtual Reality

Virtual Reality (VR) hat sich in den letzten Jahrzehnten von einem visionären Konzept zu einer greifbaren Technologie entwickelt, die in zahlreichen Bereichen unseres Lebens Anwendung findet. VR bezeichnet computergenerierte, dreidimensionale Umgebungen, die durch spezielle Hardware wie VR-Brillen und Bewegungssensoren eine immersive Erfahrung ermöglichen. Benutzer können in diese virtuellen Welten eintauchen und mit ihnen interagieren, was das Gefühl vermittelt, tatsächlich dort anwesend zu sein.

Die Wurzeln der VR-Technologie reichen bis in die 1960er Jahre zurück. Einer der frühesten Vorläufer moderner VR-Systeme war das Sensorama, das 1962 von Morton Heilig entwickelt wurde. Das Sensorama bot multisensorische Erlebnisse und kann als eines der ersten Systeme betrachtet werden, das den Nutzer vollständig in eine künstliche Umgebung eintauchen ließ [Heilig 1962]. Ein weiterer bedeutender Meilenstein war das SSword of Damocles", das erste echte VR-Headset, das 1968 von Ivan Sutherland entwickelt wurde. Es war ein klobiges Gerät, das an der Decke montiert werden musste, und bot eine rudimentäre Form der virtuellen Realität [Sutherland 1968].

In den 1980er und 1990er Jahren erlebte VR einen weiteren Aufschwung, insbesondere durch die Arbeit von Jaron Lanier, der den Begriff „Virtual Reality“ populär machte und die Firma VPL Research gründete, die einige der ersten kommerziellen VR-Produkte entwickelte [Lanier 1992]. Trotz dieser Fortschritte blieb VR lange Zeit eine Nischentechnologie, hauptsächlich aufgrund der hohen Kosten und technischen Einschränkungen.

Erst in den 2010er Jahren, mit der Einführung moderner, kostengünstigerer VR-Headsets wie der Oculus Rift, entwickelte sich VR zu einer breiter zugänglichen Technologie. Die Oculus Rift, ursprünglich 2012 auf Kickstarter finanziert, revolutionierte den Markt und führte zu einer neuen Welle von Innovationen in der VR-Technologie [Luckey 2012]. Kurz darauf folgten andere bedeutende Systeme wie die HTC Vive und PlayStation VR, die die VR-Erfahrung weiter verbesserten und breitere Zielgruppen erreichten.

Die heutige VR-Technologie zeichnet sich durch hochauflösende Displays, präzises Tracking und eine Vielzahl von Eingabemethoden aus, die eine immersive und interaktive Erfahrung ermöglichen. Meta Quest 3 (ehemals Oculus Quest 3) ist ein Beispiel für ein modernes, eigenständiges VR-Headset, das keine Verbindung zu einem leistungsstarken PC benötigt und dennoch eine beeindruckende Leistung bietet [Meta 2023].

Die Anwendungsbereiche von VR sind vielfältig und umfassen nicht nur Unterhaltung und Spiele, sondern auch Bildung, medizinische Therapie, Training und Simulationen, Architektur und Design sowie viele andere Felder. Zum Beispiel wird VR in der Medizin zur Behandlung von Phobien, in der Schmerztherapie und in der Rehabilitation eingesetzt [Rizzo und Koenig 2017]. In der Ausbildung ermöglicht VR realitätsnahe Trainingsumgebungen, die sicher und kontrolliert sind, was insbesondere in der Luftfahrt und der Medizin von großem Nutzen ist [Huang und Liaw 2018].

Die rapide Weiterentwicklung von Hardware und Software sowie die steigende Akzeptanz und Integration von VR in verschiedenen Lebensbereichen zeigen, dass diese Technologie das Potenzial hat, unsere Interaktion mit digitalen Inhalten und unsere Wahrnehmung von Realität grundlegend zu verändern. Die folgenden Abschnitte dieser Arbeit werden die technologischen Grundlagen von VR, verschiedene Anwendungsbereiche sowie die aktuellen Herausforderungen und Zukunftsaussichten der Technologie detailliert untersuchen.

2.7 Entwicklung für Virtual Reality

Die Entwicklungsumgebungen für VR sind Softwareplattformen, welche den Entwicklern die Werkzeuge und Funktionen zur Erstellung von Anwendungen für VR-Brillen bieten. Die zwei bekanntesten und am häufigsten verwendeten sind Unity und Unreal Engine. Diese beiden bieten eine umfassende Unterstützung für die VR-Entwicklung. Sie werden in vielen Anwendungen und Spielen und auch bei industriellen Lösungen verwendet. Im folgenden werden beide Entwicklungsplattformen vorgestellt, in Kapitel 3 wird dann erläutert für welche der beiden Plattformen sich für die Entwicklung entschieden wurde.

2.7.1 Unity

Unity ist eine weit verbreitete, benutzerfreundliche Entwicklungsplattform, die sich durch ihre Vielseitigkeit und umfangreiche Toolsets auszeichnet. Unity bietet integrierte Unterstützung für VR und wird häufig aufgrund seiner Benutzerfreundlichkeit und der breiten Community bevorzugt.

Benutzerfreundlichkeit Unity hat eine intuitive Benutzeroberfläche, für welche die Plattform auch bekannt ist. Ebenfalls bekannt ist es für die leichte Erlernbarkeit, weshalb es sowohl Einsteigern als auch erfahrenen Entwicklern zusagt [Technologies 2021c]. Unterstützte Plattformen: Unity unterstützt eine Vielzahl von VR-Plattformen, darunter Oculus Rift, HTC Vive, PlayStation VR und verschiedene mobile VR-Headsets wie Google Cardboard und Samsung Gear VR [Technologies 2021c].

Asset Store Der Unity Asset Store bietet eine große Auswahl an vorgefertigten Assets, Plugins und Tools, die die Entwicklung von VR-Anwendungen erleichtern und beschleunigen [Technologies 2021b].

Scripting Unity verwendet C# als Hauptprogrammiersprache, was Entwicklern ermöglicht, komplexe Interaktionen und Animationen zu erstellen [Technologies 2021c].

2.7.2 Unreal Engine

Unreal Engine wurde von Epic Games entwickelt und ist ebenfalls eine führende Plattform für die VR Entwicklung. Diese Entwicklungsumgebung ist vor allem für ihre leistungsstarke Grafik und Rendering-Fähigkeiten bekannt .

Grafikqualität Die Grafikqualität und realistischen visuellen Effekte, welche besonders hochwertige Spiele wichtig sind, sind bei Unreal Engine besonders ausgeprägt [Games 2021b]

Blueprint System Unreal Engine bietet das Blueprints Visual Scripting System, das es Entwicklern ermöglicht, ohne tiefgreifende Programmierkenntnisse komplexe Interaktionen zu erstellen [Games 2021a].

Unterstützte Plattformen Wie Unity unterstützt auch Unreal Engine eine breite Palette von VR-Plattformen, einschließlich Oculus Rift, HTC Vive und PlayStation VR [Games 2021b].

Open Source Ein großer Vorteil der Unreal Engine ist ihr Open-Source-Charakter, der Entwicklern vollständigen Zugang zum Quellcode der Engine bietet, was tiefergehende Anpassungen und Optimierungen ermöglicht [Games 2021b].

2.8 Unity-ROS TCP Controller

Der Unity-ROS TCP Controller ist ein leistungsstarkes Tool, das es ermöglicht, Robot Operating System (ROS) mit der Unity-Engine zu verbinden. Diese Integration bietet Entwicklern die Möglichkeit, VR- und AR-Anwendungen zu erstellen, die mit realen Robotern interagieren können, indem sie Daten zwischen ROS und Unity austauschen. Diese Funktionalität ist besonders nützlich in Bereichen wie Robotik, Simulationen und erweiterter Realität, wo die Interaktion zwischen virtuellen und physischen Welten von zentraler Bedeutung ist [*ROS-TCP-Endpoint Documentation* o. D.; *Unity-ROS-TCP-Connector Documentation* o. D.]

2.8.1 Grundlagen und Architektur

Der Unity-ROS TCP Controller funktioniert durch die Verwendung eines TCP-Protokolls zur Kommunikation zwischen ROS und Unity. Dies ermöglicht eine zuverlässige und bidirektionale Datenübertragung, die für Anwendungen erforderlich ist, die auf Echtzeitdaten angewiesen sind. Die grundlegende Architektur besteht aus zwei Hauptkomponenten:

1. **ROS TCP Endpoint**: Dies ist der Server, der auf der ROS-Seite läuft und auf eingehende Verbindungen von Unity wartet. Er empfängt Nachrichten von Unity und sendet Nachrichten an Unity [*ROS-TCP-Endpoint Documentation* o. D.]

2. **Unity TCP Connector**: Dies ist der Client, der in Unity läuft und eine Verbindung zum ROS TCP Endpoint herstellt. Er sendet Nachrichten an ROS und empfängt Nachrichten von ROS [*Unity-ROS-TCP-Connector Documentation* o. D.]

2.8.2 Einrichtung und Verwendung

Die Einrichtung des Unity-ROS TCP Controllers umfasst mehrere Schritte, sowohl auf der ROS- als auch auf der Unity-Seite. Hier ist eine allgemeine Anleitung zur Einrichtung:

ROS-Seite

- Installiere die erforderlichen ROS-Pakete, z.B. `ros_tcp_endpoint` [*ROS-TCP-Endpoint Documentation* o. D.]
- Starte den ROS TCP Endpoint:

```
roslaunch ros_tcp_endpoint endpoint.launch
```

Unity-Seite

- Importiere das ROS-TCP-Connector-Paket in dein Unity-Projekt [*Unity-ROS-TCP-Connector Documentation* o. D.]
- Füge das `RosConnector`-Skript zu einem GameObject in deiner Szene hinzu und konfiguriere die IP-Adresse und den Port des ROS-TCP-Endpunkts.
- Erstelle Nachrichten-Typen in Unity, die mit den ROS-Nachrichten übereinstimmen, die du senden und empfangen möchtest.
- Verwende das `RosConnector`-Skript, um Nachrichten an ROS zu senden und Nachrichten von ROS zu empfangen.

2.8.3 Beispiel für Unity-Skript

Hier ist ein Beispiel für ein Unity-Skript, das den Unity-ROS TCP Controller verwendet, um Positionsdaten an ROS zu senden:

```
using UnityEngine;
using Unity.Robotics.ROSTCPConnector;
using RosMessageTypes.Geometry;

public class PositionPublisher : MonoBehaviour
{
    ROSConnection ros;
    public string topicName = "/unity/position";
    public float publishRate = 0.5f;

    private float timeElapsed;

    void Start()
    {
        ros = ROSConnection.instance;
        ros.RegisterPublisher<PointMsg>(topicName);
    }
```

```
    void Update()
    {
        timeElapsed += Time.deltaTime;

        if (timeElapsed > publishRate)
        {
            PointMsg positionMessage = new PointMsg(
                transform.position.x,
                transform.position.y,
                transform.position.z
            );

            ros.Publish(topicName, positionMessage);

            timeElapsed = 0;
        }
    }
}
```

In diesem Beispiel wird die Position eines GameObjects in Unity periodisch an ein ROS-Thema gesendet. Das Skript:

- Registriert einen Publisher für das angegebene ROS-Thema.
- Sendet die Position des GameObjects als `PointMsg`-Nachricht an ROS [*Unity-ROS-TCP-Connector Documentation* o. D.]

2.8.4 Vorteile und Anwendungsbereiche

Der Unity-ROS TCP Controller bietet zahlreiche Vorteile:

- **Echtzeit-Interaktion**: Ermöglicht die Echtzeit-Interaktion zwischen virtuellen und physischen Robotern.
- **Flexibilität**: Unterstützt verschiedene ROS-Nachrichtentypen und ermöglicht die Anpassung an spezifische Anwendungsanforderungen.
- **Erweiterbarkeit**: Kann in verschiedene VR/AR-Projekte integriert werden, um erweiterte Robotik-Simulationen und Visualisierungen zu erstellen.

Anwendungsbereiche umfassen:

- **Robotik-Forschung**: Simulation und Testen von Robotern in virtuellen Umgebungen.

- **Bildung**: Interaktive Lernumgebungen für die Robotik-Ausbildung.
- **Industrie**: Visualisierung und Steuerung von Industrierobotern in einer virtuellen Umgebung.

2.8.5 Fazit

Der Unity-ROS TCP Controller ist ein mächtiges Werkzeug, das die Lücke zwischen virtuellen Simulationen in Unity und realen Robotersystemen, die auf ROS basieren, schließt. Durch die Nutzung von TCP für die Kommunikation bietet es eine robuste und flexible Lösung für eine Vielzahl von Anwendungen in der Robotik und darüber hinaus.

2.9 Unreal Engine ROS Integration

2.9.1 Einführung

Die Unreal Engine ROS Integration ermöglicht die Verbindung und Interaktion zwischen dem Robot Operating System (ROS) und der Unreal Engine. Diese Integration bietet ähnliche Funktionen wie der Unity-ROS TCP Controller und ist nützlich für die Erstellung von Simulationen, Visualisierungen und interaktiven Anwendungen, die mit physischen Robotern kommunizieren [ROSIntegration Contributors o. D.]

2.9.2 Grundlagen und Architektur

Das ROSIntegration Plugin für die Unreal Engine besteht aus mehreren Komponenten, die zusammenarbeiten, um eine nahtlose Kommunikation zwischen ROS und der Unreal Engine zu gewährleisten. Die grundlegende Architektur umfasst:

1. **ROS Nodes**: Diese werden in ROS definiert und dienen zur Kommunikation und Datenverarbeitung.
2. **Unreal Engine ROS Nodes**: Diese Nodes werden in der Unreal Engine erstellt und fungieren als Schnittstelle zu den ROS Nodes.

2.9.3 Einrichtung und Verwendung

Die Einrichtung des ROSIntegration Plugins in der Unreal Engine umfasst mehrere Schritte:

Installation des Plugins

- Lade das ROSIntegration Plugin von der offiziellen GitHub-Seite herunter: `https://github.com/code-iai/ROSIntegration`.
- Füge das Plugin zu deinem Unreal Engine Projekt hinzu und aktiviere es in den Projekteinstellungen.

Konfiguration des Plugins

- Konfiguriere die IP-Adresse und den Port des ROS Masters, mit dem die Unreal Engine kommunizieren soll.
- Stelle sicher, dass das ROS Master läuft und erreichbar ist.

Erstellen von ROS-Komponenten in Unreal

- Erstelle ROS-spezifische Komponenten wie Publisher und Subscriber in der Unreal Engine, um Daten zu senden und zu empfangen.

2.9.4 Beispiel für Unreal Engine Skript

Hier ist ein Beispiel für die Verwendung des ROSIntegration Plugins in der Unreal Engine, um Positionsdaten zu veröffentlichen:

```
#include "ROSIntegrationGameMode.h"
#include "ROSIntegration/Public/ROSIntegrationGameInstance.h"
#include "ROSIntegration/Public/Publisher.h"
#include "ROSIntegration/Public/std_msgs/String.h"

void AROSIntegrationGameMode::BeginPlay()
{
    Super::BeginPlay();

    UROSIntegrationGameInstance* ROSInst =
        Cast<UROSIntegrationGameInstance>(GetGameInstance());
    if (ROSInst)
    {
        ROSInst->Init();

        TSharedPtr<ROSPublisher> Publisher =
            MakeShareable(new ROSPublisher(ROSInst->ROSIntegrationCore,
                                           TEXT("/unity/position"),
                                           TEXT("geometry_msgs/Point")));
        Publisher->Advertise();

        // Publish a message
        FROSTime now = ROSInst->ROSIntegrationCore->ROSTimeNow();
        TSharedPtr<ROSMessages::geometry_msgs::Point> PointMessage =
            MakeShareable(new ROSMessages::geometry_msgs::Point());
        PointMessage->x = GetActorLocation().X;
        PointMessage->y = GetActorLocation().Y;
        PointMessage->z = GetActorLocation().Z;
```

```
        Publisher->Publish(PointMessage);
    }
}
```

In diesem Beispiel wird die Position eines Akteurs (Actors) in der Unreal Engine an ein ROS-Thema gesendet. Das Skript:

- Initialisiert das ROSIntegration Plugin.
- Erstellt einen Publisher für das angegebene ROS-Thema.
- Sendet die Position des Akteurs als `geometry_msgs/Point` Nachricht an ROS [ROSIntegration Contributors o. D.]

2.9.5 Vorteile und Anwendungsbereiche

Die Integration von ROS in die Unreal Engine bietet zahlreiche Vorteile:

- **Echtzeit-Interaktion**: Ermöglicht die Echtzeit-Interaktion zwischen virtuellen und physischen Robotern.
- **Visuelle Genauigkeit**: Die Unreal Engine bietet hochwertige Grafiken und realistische Visualisierungen, die für Simulationen und Präsentationen nützlich sind.
- **Flexibilität**: Unterstützt verschiedene ROS-Nachrichtentypen und ermöglicht die Anpassung an spezifische Anwendungsanforderungen.

Anwendungsbereiche umfassen:

- **Robotik-Forschung**: Simulation und Testen von Robotern in virtuellen Umgebungen.
- **Bildung**: Interaktive Lernumgebungen für die Robotik-Ausbildung.
- **Industrie**: Visualisierung und Steuerung von Industrierobotern in einer virtuellen Umgebung.

2.9.6 Fazit

Die Unreal Engine ROS Integration ist ein mächtiges Werkzeug, das die Lücke zwischen virtuellen Simulationen in der Unreal Engine und realen Robotersystemen, die auf ROS basieren, schließt. Durch die Nutzung dieses Plugins können Entwickler hochwertige Simulationen und Anwendungen erstellen, die eine nahtlose Integration von ROS-Daten in die Unreal Engine ermöglichen.

Kapitel 3

Technologieauswahl

3.1 VR-Brillen

Die Auswahl der Technologie für die Entwicklung von Virtual-Reality-Anwendungen ist von entscheidender Bedeutung für den Erfolg eines Projekts. Bei der DHBW wurde die Entscheidung getroffen, die Metaquest 3 für das VR-Projekt zu verwenden. Diese Entscheidung wurde aufgrund mehrerer Faktoren getroffen, die im Folgenden erläutert werden.

Die Metaquest 3 wurde aufgrund ihrer vielseitigen Anwendungsmöglichkeiten und ihrer Benutzerfreundlichkeit ausgewählt. Die DHBW legt großen Wert darauf, den Studierenden eine moderne und zugängliche Lernumgebung zu bieten. Die Metaquest 3 erfüllt diese Anforderungen durch ihre intuitive Bedienung und ihre Fähigkeit, komplexe VR-Erlebnisse bereitzustellen, ohne dass zusätzliche Hardware wie externe Sensoren erforderlich sind.

Ein weiterer wichtiger Faktor bei der Auswahl der Metaquest 3 war die Integration von Oculus in die bestehende IT-Infrastruktur der DHBW. Die Unterstützung und Zusammenarbeit mit Oculus ermöglichte es der DHBW, Schulungen und Support für die Verwendung der Metaquest 3 bereitzustellen. Dies erleichterte die Einführung der VR-Technologie in den Lehrplan und sorgte für eine nahtlose Integration in bestehende Lehr- und Lernaktivitäten.

Darüber hinaus bietet die Metaquest 3 eine breite Palette von Anwendungen und Inhalten über den Oculus Store, einschließlich Bildungs- und Trainingsanwendungen, die für den Einsatz in der Hochschulbildung geeignet sind. Die Verfügbarkeit von hochwertigen Bildungsressourcen spielte eine wichtige Rolle bei der Entscheidung für die Metaquest 3, da sie den Lehrern und Studierenden Zugang zu einer Vielzahl von Lernmaterialien und Simulationen bietet, die den Lernprozess unterstützen und verbessern können.

Insgesamt wurde die Metaquest 3 aufgrund ihrer Benutzerfreundlichkeit, Integration in die bestehende Infrastruktur der DHBW und der Verfügbarkeit von Bildungsressourcen als ideale Wahl für das VR-Projekt der DHBW angesehen. Diese Auswahl bietet nicht nur eine solide Grundlage für die Entwicklung von VR-Anwendungen, sondern ermöglicht es auch, die VR-Technologie effektiv in den Lehrplan zu integrieren und den Lernerfolg zu

steigern.

3.2 Entwicklung für VR-Brillen

Die Entscheidung, Unity als Entwicklungsplattform für das VR-Projekt "Pepper VR – Teleoperation eines humanoiden Roboters auf Basis der Analyse menschlicher Bewegungßu wählen, wurde nach sorgfältiger Abwägung verschiedener Faktoren getroffen, wobei insbesondere auch die Unreal Engine in Betracht gezogen wurde.

Unity

Im folgenden werden die Vor- und Nachteile von Unity beleuchtet.

Vorteile:

1. **Branchenübliche Plattform**: Unity ist eine der führenden Entwicklungsplattformen für VR-Anwendungen und wird von einer großen Community von Entwicklern und Unternehmen weltweit genutzt. Diese weitverbreitete Akzeptanz macht Unity zu einer branchenüblichen Wahl für die Entwicklung von VR-Inhalten und bietet Zugang zu einer Vielzahl von Ressourcen, Tutorials und Support, die für die erfolgreiche Umsetzung des Projekts entscheidend sind.

2. **Umfangreiche Funktionalitäten**: Unity bietet eine umfangreiche Auswahl an Funktionen und Werkzeugen, die speziell für die Entwicklung von VR-Anwendungen konzipiert sind. Die Integration von VR-Technologien wie Oculus Rift, HTC Vive und Metaquest in Unity ermöglicht es den Entwicklern, immersive VR-Erlebnisse mit hoher Qualität zu erstellen. Darüber hinaus bietet Unity eine benutzerfreundliche Oberfläche und eine intuitive Entwicklungsumgebung, die auch für Anfänger leicht zugänglich ist.

3. **Plattformübergreifende Unterstützung**: Unity ermöglicht die Entwicklung von VR-Anwendungen, die auf einer Vielzahl von Plattformen ausgeführt werden können, einschließlich PC, Konsolen, Mobilgeräten und Webbrowsern. Diese Flexibilität eröffnet die Möglichkeit, das VR-Projekt auf verschiedenen Geräten und Betriebssystemen zu testen und bereitzustellen, um eine maximale Reichweite und Zugänglichkeit zu gewährleisten.

4. **Erweiterbarkeit und Anpassbarkeit**: Unity zeichnet sich durch seine Erweiterbarkeit und Anpassbarkeit aus. Durch den Einsatz von Plugins und Assets aus dem Unity Asset Store können Entwickler zusätzliche Funktionen und Ressourcen in ihre VR-Anwendungen integrieren, was die Entwicklung beschleunigt und die Produktivität erhöht. Darüber hinaus bietet Unity die Möglichkeit, eigene Tools und Skripte zu erstellen, um die spezifischen Anforderungen des Projekts zu erfüllen und maßgeschneiderte Lösungen zu entwickeln.

Nachteile:

1. **Grafische Qualität**: Obwohl Unity in Bezug auf die Grafikqualität fortschrittliche Techniken bietet, erreicht es möglicherweise nicht das gleiche grafische Niveau wie die Unreal Engine, insbesondere bei fotorealistischen Rendering-Anforderungen.

2. **Lernkurve**: Unity kann eine steilere Lernkurve haben als die Unreal Engine, insbesondere für Anfänger oder Personen ohne Programmiererfahrung. Die Vielzahl von Funktionen und Optionen kann anfangs überwältigend sein.

Unreal Engine

Nun werden die Vor- und Nachteile von Unreal Engine betrachtet.

Vorteile:

1. **Grafische Qualität**: Die Unreal Engine ist bekannt für ihre beeindruckende Grafikqualität und ihre Fähigkeit, fotorealistische Umgebungen zu erstellen. Sie bietet fortschrittliche Rendering-Techniken wie Raytracing und hochwertige Materialien, die für VR-Anwendungen mit hohen grafischen Anforderungen von Vorteil sind.

2. **Visuelle Skripting-Tools**: Die Unreal Engine bietet visuelle Skripting-Tools wie den Blueprint-Editor, die es auch Personen ohne umfangreiche Programmierkenntnisse ermöglichen, komplexe Logik und Interaktionen zu erstellen. Dies kann die Entwicklungszeit verkürzen und die Kreativität fördern.

3. **Leistung**: Die Unreal Engine ist für ihre hohe Leistung und Stabilität bekannt, insbesondere bei großen Projekten mit komplexen Szenen und großen Datenmengen.

Nachteile:

1. **Einschränktere Plattformunterstützung**: Im Vergleich zu Unity bietet die Unreal Engine möglicherweise eine eingeschränktere Plattformunterstützung für die Entwicklung von VR-Anwendungen. Die Unterstützung für bestimmte VR-Geräte oder Plattformen kann begrenzt sein.

2. **Komplexität**: Die Unreal Engine kann aufgrund ihrer fortschrittlichen Funktionen und der visuellen Komplexität ihrer Benutzeroberfläche für Anfänger schwieriger zu erlernen sein. Die Blueprint-Logik kann zwar visuell sein, erfordert jedoch immer noch ein Verständnis von Konzepten wie Variablen und Logik.

Entscheidung

Trotz der Vorteile der Unreal Engine in Bezug auf Grafikqualität und Leistung wurde Unity als die bevorzugte Entwicklungsplattform für das VR-Projekt "Pepper VR – Teleoperation eines humanoiden Roboters auf Basis der Analyse menschlicher Bewegung"gewählt. Die

breite Unterstützung, die umfangreichen Funktionalitäten, die plattformübergreifende Unterstützung und die Erweiterbarkeit von Unity waren entscheidend für diese Wahl. Unity bietet eine solide Grundlage für die Entwicklung hochwertiger VR-Anwendungen, die den Anforderungen des Projekts gerecht werden und eine erfolgreiche Integration von VR-Technologie in den Lehrplan ermöglichen.

3.2.1 Verbindungstechnologie

Die Wahl von Unity als Entwicklungsplattform für das VR-Projekt "Pepper VR – Teleoperation eines humanoiden Roboters auf Basis der Analyse menschlicher Bewegung"führte zur Notwendigkeit, eine Methode zur Kommunikation zwischen der Unity-Anwendung und dem humanoiden Roboter Pepper zu implementieren. Die ROS TCP-Verbindung wurde aufgrund ihrer Kompatibilität mit Unity und der Robotersteuerung über das Robot Operating System (ROS) als geeignete Lösung identifiziert.

Kompatibilität mit Unity

ROS TCP (Transmission Control Protocol) bietet eine zuverlässige Methode zur Datenübertragung zwischen ROS und Unity. Unity unterstützt die Kommunikation über TCP/IP-Sockets, was es ermöglicht, Daten zwischen der Unity-Anwendung und externen Geräten wie Robotern über das Netzwerk auszutauschen. Durch die Implementierung einer ROS TCP-Verbindung kann die Unity-Anwendung Befehle an den Roboter senden und Daten von ihm empfangen, was eine nahtlose Integration in das VR-Erlebnis ermöglicht.

Robotersteuerung über ROS

Pepper, der humanoide Roboter, wird über das Robot Operating System (ROS) gesteuert, das eine Standardplattform für die Entwicklung von Robotersoftware ist. ROS bietet eine Vielzahl von Funktionen zur Robotersteuerung, einschließlich der Unterstützung für verschiedene Sensoren, Aktuatoren und Navigationssysteme. Indem die ROS TCP-Verbindung verwendet wird, kann die Unity-Anwendung mit den ROS-Nodes kommunizieren, die für die Steuerung von Pepper zuständig sind. Dies ermöglicht es der Unity-Anwendung, Pepper-Bewegungen zu steuern und Sensordaten von Pepper zu empfangen, um ein interaktives VR-Erlebnis zu schaffen.

Implementierung in Unity

Die Implementierung der ROS TCP-Verbindung in Unity erfolgt mithilfe von Plugins oder eigenen Skripten, die die TCP/IP-Kommunikation ermöglichen. Durch die Verwendung von vorhandenen ROS-Bibliotheken oder der Erstellung benutzerdefinierter ROS-Nodes kann die Unity-Anwendung ROS-Nachrichten senden und empfangen, um mit Pepper zu interagieren. Dies ermöglicht es, die Bewegungen von Pepper in Echtzeit zu steuern und Feedbackdaten von Pepper in die Unity-Anwendung zu integrieren.

Insgesamt wurde die ROS TCP-Verbindung aufgrund ihrer Kompatibilität mit Unity und der Robotersteuerung über ROS als ideale Lösung für die Kommunikation zwischen der Unity-Anwendung und dem humanoiden Roboter Pepper ausgewählt. Diese Entscheidung ermöglicht es, eine immersive und interaktive VR-Erfahrung zu schaffen, bei der die Benutzer direkt mit dem Roboter interagieren können.

Kapitel 4

Umsetzung

4.1 Pepper

4.2 MetaQuest3

4.3 Verbindung

Kapitel 5

Anwendungsgebiete

Das entstandene Produkt kann in verschiedenen Fällen eingesetzt werden.

5.1 Pflege

In Pflegeeinrichtungen können Beispielsweise Bewohner, die selbst nicht mehr so gut zu Fuß sind sich gegenseitig Besuchen und kommunizieren. Ebenfalls können Routinebesuche bei den Besuchern durch Pflegekräfte von einer zentralen Stelle aus getätigt werden, was die Pflegekräfte entlasten würde.

Kapitel 6

Fazit

Die Umsetzung des Projekts *Pepper VR – Teleoperation eines humanoiden Roboters auf Basis der Analyse menschlicher Bewegung* stellte eine anspruchsvolle Herausforderung dar, die tiefgreifende Kenntnisse in verschiedenen Bereichen der Informatik und Robotik erforderte. Trotz der ambitionierten Ziele, die Verbindung zwischen der MetaQuest 3 VR-Brille und dem humanoiden Roboter Pepper herzustellen und die Steuerung von Pepper durch die VR-Controller zu ermöglichen, stießen wir auf mehrere technische und organisatorische Hürden. Welche die Umsetzung leider schlussendlich nicht möglich machten.

Die wichtigsten Erkenntnissen und Schlussfolgerungen sind:

6.1 Technische Herausforderungen

Die Implementierung der Steuerung eines humanoiden Roboters über eine VR-Brille wie die MetaQuest 3 erfordert eine präzise und latenzfreie Datenübertragung. Probleme mit der Netzwerkverbindung, Synchronisation und Verzögerungen führten dazu, dass die Steuerung von Pepper nicht in der gewünschten Präzision umgesetzt werden konnte.

6.2 Komplexität der Integration

Die Integration von Unity und ROS über den ROS-TCP-Connector stellte sich als komplexer als erwartet heraus. Obwohl Unity eine benutzerfreundliche Entwicklungsumgebung bietet, waren die Anforderungen an die ROS-Integration höher als erwartet, insbesondere im Hinblick auf Echtzeitfähigkeit und die nahtlose Zusammenarbeit der verschiedenen Softwarekomponenten.

6.3 Wahl von Unity

Die Entscheidung für Unity als Entwicklungsplattform wurde aufgrund ihrer weiten Verbreitung, der umfangreichen Dokumentation und der großen Entwicklergemeinschaft

getroffen. Unity bietet viele vorgefertigte Lösungen und Plugins, die die Entwicklung beschleunigen können. Dennoch stellte sich heraus, dass für diese spezielle Aufgabe tiefergehende Kenntnisse im Umgang mit der Unity-ROS-Integration erfordert, als in der gegebenen Zeit erlernbar war.

6.4 Potenzial für zukünftige Arbeiten

Trotz der Herausforderungen zeigt das Projekt deutlich das Potenzial, das in der Kombination von VR und humanoider Robotik liegt. Künftige Arbeiten könnten von den gemachten Erfahrungen profitieren und die entwickelten Ansätze weiter verfolgen. Welche Schritte zur Weiterführung notwendig sind, werden auch im nachfolgenden Kapitel genauer erklärt.

Insgesamt war das Projekt eine wertvolle Lernerfahrung, die Einblicke in die komplexe Welt der Robotik und Virtual Reality ermöglichte. Die gesammelten Erfahrungen und Erkenntnisse werden als Grundlage für zukünftige Entwicklungen dienen, und wir sind zuversichtlich, dass die Vision einer nahtlosen Teleoperation von humanoiden Robotern über VR-Systeme mit weiterem Engagement und Forschung realisiert werden kann.

Kapitel 7

Fortsetzung des Projekts

Um das Projekt *Pepper VR – Teleoperation eines humanoiden Roboters auf Basis der Analyse menschlicher Bewegung* in Zukunft erfolgreich umzusetzen, sind mehrere wichtige Schritte und Verbesserungen erforderlich. Nach der Analyse der bisherigen Herausforderungen und der gewonnenen Erkenntnisse haben wir folgende Empfehlungen für die Fortsetzung des Projekts zusammengestellt:

7.1 Optimierung der technischen Infrastruktur

Eine der größten Herausforderungen war die präzise und latenzfreie Datenübertragung zwischen der MetaQuest 3 VR-Brille und Pepper. Zur Verbesserung der technischen Infrastruktur sollten folgende Maßnahmen ergriffen werden:

- **Verbesserung der Netzwerkverbindung**: Sicherstellen einer stabilen und schnellen Netzwerkverbindung, möglicherweise durch den Einsatz von dedizierten Netzwerken oder der Optimierung der bestehenden Infrastruktur.
- **Reduzierung der Latenzzeiten**: Implementierung von Echtzeit-Optimierungen sowohl auf der Hardware- als auch auf der Softwareseite, um die Verzögerungen bei der Datenübertragung zu minimieren.

7.2 Erweiterung der Softwareintegration

Die Integration von Unity und ROS über den ROS-TCP-Connector muss weiter verfeinert und genauer recherchiert werden. Hier sind die wesentlichen Schritte:

- **Tiefere ROS-Integration**: Entwicklung zusätzlicher ROS-Nodes und -Services, um eine nahtlosere Kommunikation zwischen Unity und dem Robot Operating System zu gewährleisten.
- **Fehlersuche und Debugging**: Intensive Fehlersuche und Debugging der bestehenden Implementierung, um die Ursachen für Synchronisationsprobleme zu identifizieren und zu beheben.

7.3 Testen und Validieren der VR-Steuerung

Um die Steuerung von Pepper über die MetaQuest 3 VR-Brille erfolgreich umzusetzen, sind umfangreiche Tests und Validierungen notwendig:

- **Erstellung von Testfällen**: Entwicklung spezifischer Testfälle, die die verschiedenen Aspekte der VR-Steuerung abdecken, um sicherzustellen, dass alle Funktionen wie erwartet arbeiten.
- **Benutzerstudien**: Durchführung von Benutzerstudien, um die Benutzerfreundlichkeit und Effektivität der VR-Steuerung zu bewerten und basierend auf dem Feedback Verbesserungen vorzunehmen.

7.4 Schulung und Dokumentation

Die erfolgreiche Umsetzung des Projekts erfordert gut geschulte Teammitglieder und umfassende Dokumentation:

- **Schulung der Teammitglieder**: Regelmäßige Schulungen und Workshops für das Team, um sicherzustellen, dass alle Mitglieder die notwendigen Kenntnisse und Fähigkeiten haben, um mit den verwendeten Technologien effektiv zu arbeiten.
- **Erstellung umfassender Dokumentation**: Dokumentation aller Implementierungs- und Testprozesse, um eine klare und nachvollziehbare Basis für zukünftige Arbeiten zu schaffen.

7.5 Langfristige Wartung und Weiterentwicklung

Um das Projekt langfristig erfolgreich zu halten, sollten regelmäßige Wartung und Weiterentwicklungen eingeplant werden:

- **Regelmäßige Updates**: Kontinuierliche Aktualisierung der verwendeten Software und Hardware, um mit den neuesten Entwicklungen Schritt zu halten und die Stabilität und Sicherheit zu gewährleisten.
- **Weiterentwicklung der Funktionen**: Fortlaufende Erweiterung der Funktionen und Fähigkeiten des Systems, basierend auf den Bedürfnissen und Rückmeldungen der Benutzer.

Durch die Umsetzung dieser Maßnahmen kann das Projekt *Pepper VR* erfolgreich weitergeführt werden, um das volle Potenzial der Kombination von Virtual Reality und humanoider Robotik auszuschöpfen. Die gewonnenen Erkenntnisse und Erfahrungen bilden eine solide Grundlage für zukünftige Entwicklungen und Innovationen in diesem spannenden und zukunftsträchtigen Bereich.

Literatur

Autodesk [2021a]. *3ds Max*. https://www.autodesk.com/products/3ds-max/overview.

— [2021b]. *Autodesk Maya*. https://www.autodesk.com/products/maya/overview.

Carnegie Mellon University [2024]. *NAOqi Documentation*. https://www.cs.cmu.edu/~cga/nao/doc/reference-documentation/dev/naoqi/index.html.

Dante [Jan. 2010]. *Webseite der* Deutschsprachige Anwendervereinigung TeX e.V. http://www.dante.de.

Foundation, Blender [2021]. *Blender*. https://www.blender.org/.

Games, Epic [2021a]. *Blueprints Visual Scripting*. https://docs.unrealengine.com/en-US/ProgrammingAndScripting/Blueprints/index.html [siehe S. 13].

— [2021b]. *Unreal Engine Documentation*. https://docs.unrealengine.com/ [siehe S. 12, 13].

Heilig, Morton [1962]. »Sensorama Simulator«. Pat. U.S. Patent 3,050,870 [siehe S. 8, 11].

Huang, Wen-Hao David und Shu-Sheng Liaw [2018]. »An analysis of learners' intentions toward virtual reality learning based on constructivist and technology acceptance approaches«. In: *International Review of Research in Open and Distributed Learning* [siehe S. 9, 11].

Knuth, Donald E. [1984]. *The TEXbook*. Addison-Wesley.

Lamport, Leslie [1995]. *Das LaTeX Handbuch*. Addison-Wesley.

Lanier, Jaron [1992]. *Virtual Reality: The Revolutionary Technology of Computer-Generated Artificial Worlds - and How It Promises to Transform Society*. Simon & Schuster [siehe S. 9, 11].

Luckey, Palmer [2012]. *Oculus Rift: Step into the Game*. https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game [siehe S. 9, 11].

Meta [2023]. *Meta Quest 3*. https://www.meta.com/quest/quest-3/ [siehe S. 9, 11].

Rizzo, Albert A. und Steven T. Koenig [2017]. »Is clinical virtual reality ready for primetime?« In: *Neuropsychology* [siehe S. 9, 11].

ROS-TCP-Endpoint Documentation [o. D.] `https://github.com/Unity-Technologies/ROS-TCP-Endpoint`. Accessed: 2024-05-14 [siehe S. 13, 14].

ROSIntegration Contributors [o. D.] *ROSIntegration Plugin for Unreal Engine*. `https://github.com/code-iai/ROSIntegration`. Accessed: 2024-05-14 [siehe S. 16, 18].

Sutherland, Ivan E. [1968]. »A head-mounted three-dimensional display«. In: *Proceedings of the Fall Joint Computer Conference* [siehe S. 8, 11].

Technologies, Unity [2021a]. *OpenXR in Unity*. `https://docs.unity3d.com/Manual/com.unity.xr.openxr.html`.

— [2021b]. *Unity Asset Store: Oculus Integration*. `https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022` [siehe S. 12].

— [2021c]. *Unity Documentation*. `https://docs.unity3d.com/Manual/index.html` [siehe S. 12].

Unity-ROS-TCP-Connector Documentation [o. D.] `https://github.com/Unity-Technologies/ROS-TCP-Connector`. Accessed: 2024-05-14 [siehe S. 13–15].