



UNIVERSITY
OF APPLIED SCIENCES
UPPER AUSTRIA

BACHELORSTUDIENGANG

Automatisierungstechnik

Erstellen von Applikationen für humanoide Robotik basierend auf Imitation Learning Methoden

ALS BACHELORARBEIT EINGEREICHT

zur Erlangung des akademischen Grades

Bachelor of Science in Engineering (BSc)

von

Jasmin Melanie Wiesinger

Wels, September 2026

Betreuung der Arbeit durch

Thomas Schichl BSc MSc

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Die vorliegende Arbeit ist mit dem elektronisch übermittelten Textdokument identisch.

.....

Jasmin Melanie Wiesinger

Grieskirchen, September 2026

KURZFASSUNG

Humanoide Roboter gewinnen zunehmend an Bedeutung. Die klassische Programmierung stößt jedoch bei komplexen Bewegungsabläufen schnell an ihre Grenzen. Einen möglichen Lösungsansatz stellt das Imitation Learning dar, bei dem komplexe Aufgaben anhand menschlicher Demonstrationen erlernt werden. In dieser Arbeit wurde untersucht, wie der humanoide Roboter Unitree G1 mithilfe von Imitation Learning einfache Manipulationsaufgaben erlernen und autonom ausführen kann.

Ziel dieser Arbeit war die Umsetzung eines Imitation-Learning-Workflows auf dem Unitree G1, damit der Roboter einfache Pick-and-Place-Aufgaben selbstständig ausführen kann. Hierzu wurden zunächst die Softwareumgebungen auf dem Desktop-PC sowie auf dem Roboter-PC eingerichtet und angepasst. Anschließend wurden mithilfe einer VR-Brille Demonstrationsdaten aufgezeichnet und mit dem LeRobot Framework aufbereitet. Auf Grundlage dieses Trainingsdatensatzes wurde eine ACT-Policy trainiert, auf das reale Robotersystem übertragen und hinsichtlich ihrer Robustheit evaluiert.

Die Ergebnisse dieser Arbeit zeigen, dass der Roboter bereits anhand weniger Demonstrationen einfache Pick-and-Place-Aufgaben ausführen kann. Mit steigender Aufgabenkomplexität sowie einer größeren Vielfalt möglicher Ausgangssituationen erhöht sich die Anzahl der benötigten Demonstrationen jedoch deutlich. Das System reagiert robust auf geringfügige Änderungen des Hintergrundbildes. Befindet sich das zu manipulierende Objekt jedoch an einer Position, die in den Trainingsdaten nicht enthalten war, kann die Policy nicht zuverlässig generalisieren und die Aufgabe nicht erfolgreich abschließen. Darüber hinaus zeigte sich, dass der Unitree G1 aufgrund seiner begrenzten Genauigkeit und Steifigkeit für präzise Manipulationsaufgaben nur eingeschränkt geeignet ist, weshalb teilweise mehrere Greifversuche erforderlich sind.

Zur Verbesserung der Generalisierungsfähigkeit sind eine größere Anzahl an Demonstrationen sowie eine höhere Variabilität der Trainingsdaten erforderlich. Das mittels Imitation Learning erlernte Modell kann anschließend auch durch nachgelagerte Reinforcement-Learning-Verfahren optimiert werden. Während Imitation Learning eine effiziente Grundlage für das Erlernen der Aufgabe schafft, kann Reinforcement Learning die Robustheit gegenüber unbekannten Situationen verbessern.

ABSTRACT

Humanoid robots are becoming increasingly important. However, traditional robot programming quickly reaches its limits when dealing with complex motion sequences. One possible solution is imitation learning, in which complex tasks are learned through human demonstrations. This thesis investigated how the Unitree G1 humanoid robot could learn and autonomously perform simple manipulation tasks using imitation learning.

The objective of this thesis was to implement an imitation learning workflow on the Unitree G1, enabling the robot to perform simple pick-and-place tasks autonomously. To achieve this, the software environments on both the industrial PC and the robot PC were first set up and configured. Subsequently, demonstration data were recorded using a virtual reality headset and processed with the LeRobot Framework. Based on this training dataset, an ACT policy was trained, transferred to the real robotic system, and evaluated with regard to its robustness.

The results show that the robot can perform simple pick-and-place tasks based on only a few demonstrations. However, as task complexity and the variety of possible initial configurations increase, the number of required demonstrations also increases significantly. The system is robust against minor changes in the background image. However, if the object to be manipulated is located at a position that was not included in the training data, the policy cannot generalize reliably and is unable to complete the task successfully. Furthermore, it was found that the Unitree G1 is only partially suitable for precise manipulation tasks due to its limited accuracy and stiffness, making multiple grasping attempts necessary in some cases.

To improve the generalization capability, a larger number of demonstrations and greater variability in the training data are required. The model learned through imitation learning can subsequently be further optimized using reinforcement learning methods. While imitation learning provides an efficient foundation for learning the task, reinforcement learning can further improve robustness against previously unseen situations.

INHALTSVERZEICHNIS

Kurzfassung	III
Abstract.....	IV
1 Einleitung.....	1
2 Stand der Technik	3
2.1 Humanoide Robotik.....	3
2.1.1 Definition.....	3
2.1.2 Einsatzgebiete und Motivation	4
2.1.3 Aktuelle humanoide Robotersysteme	4
2.1.4 Technische Herausforderungen humanoider Robotersysteme	10
2.2 Methoden der Programmierung humanoider Robotersysteme	11
2.2.1 Reinforcement Learning	12
2.2.1.1 Grundprinzip.....	12
2.2.1.2 Lernprozess.....	14
2.2.2 Imitation Learning	15
2.2.2.1 Behavioral Cloning.....	16
2.2.2.2 Inverse Reinforcement Learning	21
2.2.3 Vision-Language-Action-Modelle	22
2.2.3.1 Grundprinzip.....	23
2.2.3.2 Funktionsweise	25
2.2.4 Vergleich der Programmiermethoden	26
3 Methodik des Imitation Learning	28
3.1 Demonstrationsdatenerfassung.....	28
3.2 Aufbereitung des Trainingsdatensatzes	31
3.3 Modelltraining	32
3.4 Deployment am Robotersystem	33
4 Implementierung und Evaluierung	37
4.1 Entwicklungsumgebung	37
4.1.1 Hardware	37
4.1.2 Software.....	38
4.1.3 Versuchsaufbau	39
4.2 Demonstrationsdatenerfassung.....	41

4.3	Aufbereitung des Trainingsdatensatzes	43
4.4	Modelltraining	44
4.5	Deployment am Robotersystem	45
4.6	Evaluierung der Ergebnisse	48
5	Zusammenfassung/Schlussfolgerung	52
6	Abkürzungsverzeichnis	54
7	Abbildungsverzeichnis	55
8	Tabellenverzeichnis	56
9	Literaturverzeichnis	57
10	Anhang	62

1 EINLEITUNG

Die Robotik hat sich in den vergangenen Jahren stark weiterentwickelt. Neben den bereits etablierten Industrierobotern werden zunehmend Robotersysteme entwickelt, die auch bei variierenden Aufgaben und Umgebungsbedingungen eingesetzt werden können. Eine besondere Entwicklung stellen dabei humanoide Robotersysteme dar. Durch ihren menschenähnlichen Aufbau können sie grundsätzlich in Arbeitsumgebungen eingesetzt werden, die für den Menschen ausgelegt sind.

Mit zunehmender Aufgabenkomplexität steigt jedoch auch der Aufwand für die Programmierung der erforderlichen Bewegungsabläufe. Insbesondere wenn unterschiedliche Ausgangssituationen berücksichtigt werden müssen, ist es aufwendig, für jede mögliche Situation geeignete Bewegungsabläufe manuell festzulegen. Lernbasierte Verfahren bieten hierfür einen alternativen Ansatz. Anstatt die Bewegungen für jede Situation einzeln vorzugeben, wird das gewünschte Verhalten anhand von Daten erlernt. Zu den möglichen Verfahren zählen unter anderem Imitation Learning und Reinforcement Learning.

Beim Imitation Learning wird eine Aufgabe anhand menschlicher Demonstrationen erlernt. Expertinnen und Experten führen die gewünschte Aufgabe vor, während die dabei entstehenden Daten aufgezeichnet werden. Auf Grundlage dieser Demonstrationen wird anschließend eine Policy trainiert, die das erlernte Verhalten auf dem Robotersystem ausführen kann.

Im Rahmen dieser Bachelorarbeit wird dieser Ansatz am humanoiden Roboter Unitree G1 untersucht. Dabei steht folgende Forschungsfrage im Mittelpunkt:

Wie kann der humanoide Roboter Unitree G1 mittels Imitation Learning eine Aufgabe erlernen und autonom ausführen?

Zur Beantwortung dieser Frage wird eine Pick-and-Place-Aufgabe ausgewählt, die der Roboter nach dem Training selbstständig ausführen soll. Hierfür wird ein vollständiger IL-Workflow von der Erfassung der Demonstrationen über die Aufbereitung der Trainingsdaten und das Modelltraining bis zur Ausführung auf dem realen Robotersystem

umgesetzt. Um die für das Training benötigte Anzahl an Demonstrationen zu reduzieren, werden die möglichen Ausgangssituationen bewusst begrenzt.

Für die praktische Umsetzung wird zunächst die Entwicklungsumgebung eingerichtet und entsprechend den Anforderungen des verwendeten Robotersystems angepasst. Dazu zählen insbesondere die Teleoperation über ein VR-System sowie die Integration der FTP Inspire-Hand als Endeffektor. Anschließend werden Demonstrationsdaten aufgezeichnet, zu einem Trainingsdatensatz aufbereitet und für das Training einer ACT-Policy verwendet. Die trainierte Policy wird anschließend auf dem realen Robotersystem deployed und unter unterschiedlichen Bedingungen getestet.

Zu Beginn der Arbeit werden die technischen Grundlagen der humanoiden Robotik sowie verschiedene Ansätze zur Programmierung solcher Robotersysteme dargestellt. Anschließend wird Imitation Learning als zentraler Ansatz dieser Arbeit detaillierter erläutert und der verwendete Workflow beschrieben. Darauf aufbauend folgt die praktische Umsetzung dieses Workflows am Unitree G1, von der Entwicklungsumgebung und Demonstrationsdatenerfassung über die Aufbereitung der Trainingsdaten und das Modelltraining bis zum Deployment. Abschließend werden die Ergebnisse der autonomen Ausführung evaluiert und die dabei gewonnenen Erkenntnisse sowie mögliche Ansätze für eine Weiterentwicklung des verwendeten Verfahrens diskutiert.

2 STAND DER TECHNIK

2.1 HUMANOIDE ROBOTIK

2.1.1 DEFINITION

Die humanoide Robotik ist ein Teilgebiet der Robotik, das sich mit der Entwicklung robotischer Systeme beschäftigt, die in ihrer Optik und Statur dem Menschen nachempfunden sind. Ein humanoider Roboter ist durch eine charakteristische Körpersegmentierung gekennzeichnet, die eine Ähnlichkeit zur menschlichen Anatomie aufweist. Die Mehrheit der humanoiden Roboter ist mit einem Kopf, einem Rumpf sowie zwei Armen und zwei Beinen ausgestattet. Es ist jedoch anzumerken, dass die genaue Übereinstimmung mit dem menschlichen Erscheinungsbild nicht zwingend erforderlich ist. Vielmehr sind auch Variationen in Form von Rädern anstelle von Füßen oder unterschiedlichen Händen mit diversen Greifvarianten möglich. Die Handkonfiguration kann beispielsweise von einfachen Greifern bis hin zur Fünf-Finger-Hand mit taktilen Sensoren für feinmotorische Aufgaben reichen [vgl. 1, S. 405-412].

Um eine Interaktion mit der Umgebung einschließlich einer Feedbackschleife zu ermöglichen, können humanoide Robotersysteme zudem mit unterschiedlichen Sensoren ausgestattet werden. Je nach Modell reicht die Sensorik von einer einfachen Head-Only-Kamera bis hin zu zusätzlichen LiDAR-Sensoren und Tiefenkameras [vgl. 1, S. 412].

Neben der menschenähnlichen physischen Gestalt werden humanoide Roboter in der Literatur häufig auch durch ihre funktionalen Eigenschaften charakterisiert. Hierzu zählen insbesondere die Fähigkeit zur Fortbewegung in menschenorientierten Umgebungen (Locomotion), die geschickte Manipulation von Objekten (Dexterity) sowie die Wahrnehmung und autonome Interaktion mit der Umgebung (Intelligence). Diese Eigenschaften ermöglichen es humanoiden Robotern, Aufgaben in für Menschen konzipierten Umgebungen auszuführen [vgl. 2, S. 2].

2.1.2 EINSATZGEBIETE UND MOTIVATION

Die Entwicklung humanoider Roboter zielt primär darauf ab, die Fähigkeit zur Adaption an variable Umgebungsbedingungen sowie die zunehmende Autonomie in der Aufgabenausführung sicherzustellen. Im Gegensatz zu den bisherigen, auf spezifische Aufgaben optimierten Industrie-Robotern sind humanoide Roboter für den Einsatz in bestehenden, für den Menschen ausgelegten Umgebungen konzipiert. Aufgrund ihrer menschenähnlichen Bauweise sind sie in der Lage, sich an unterschiedliche Umgebungen anzupassen, was eine vielseitige Anwendung in diversen Bereichen wie Industrie, Gesundheitswesen, Bildung, Rettungseinsätzen oder im privaten Umfeld ermöglicht, ohne eine starke Anpassung des Umfelds an den Roboter vorauszusetzen [vgl. 3, S. 301-302].

Ein weiterer Vorteil der menschenähnlichen Körperstruktur besteht darin, dass humanoide Roboter Treppen und Leitern überwinden können. Dadurch ist ihr Einsatz nicht auf ebene Arbeitsumgebungen beschränkt. Aufgrund ihrer anthropomorphen Bauweise und ihrer Manipulationsfähigkeiten können sie zudem für den Menschen ausgelegte Werkzeuge verwenden sowie in bestehende Arbeitsumgebungen eingesetzt werden. Dies ermöglicht die Nutzung bestehender Infrastrukturen, ohne dass umfangreiche Anpassungen oder Umgestaltungen erforderlich sind [vgl. 4, S. 2].

Die beschriebenen Eigenschaften und Einsatzmöglichkeiten spiegeln sich in der Entwicklung zahlreicher humanoider Robotersysteme wider. Diese unterscheiden sich hinsichtlich Konstruktion, Funktionsumfang und Anwendungsbereich. Im folgenden Abschnitt werden einige aktuelle Systeme vorgestellt und miteinander verglichen.

2.1.3 AKTUELLE HUMANOIDE ROBOTERSYSTEME

In den vergangenen Jahren hat die Entwicklung humanoider Robotersysteme erheblich an Dynamik gewonnen [vgl. 3, S. 313]. Zahlreiche Unternehmen verfolgen unterschiedliche Konzepte und Zielsetzungen, wodurch sich die entwickelten Systeme sowohl hinsichtlich ihrer technischen Auslegung als auch ihrer vorgesehenen Einsatzbereiche unterscheiden. In Tabelle 1 wird ein Überblick über ausgewählte aktuelle humanoide Robotersysteme gegeben.

Tabelle 1: Aktuelle humanoide Robotersysteme im Vergleich (eigene Darstellung)

Robotersystem	Hersteller	Größe [cm]	Gewicht [kg]	Primärer Einsatzbereich	Quelle
Figure 03	Figure AI	172	61	Haushalt	[vgl. 5]
Walker S2	UBTECH Robotics	176	70	Industrielle Fertigung	[vgl. 6]
Digit	Agility Robotics	175	65	Logistik	[vgl. 7]
GR-3	FOURIER-Robotics	165	71	Pflege	[vgl. 8]
Unitree G1	Unitree Robotics	130	35	Forschung	[vgl. 9]

Der Vergleich zeigt, dass sich aktuelle humanoide Robotersysteme hinsichtlich ihrer Baugröße, ihres Gewichts sowie ihres vorgesehenen Einsatzbereichs unterscheiden. Während Systeme wie Walker S2 oder Digit primär für industrielle Anwendungen entwickelt wurden, ist der Figure 03 insbesondere für die Übernahme von Haushaltsaufgaben vorgesehen. Der Fourier GR-3 ist für Anwendungen im Pflegebereich konzipiert. Der Unitree G1 verfolgt dagegen einen forschungsorientierten Ansatz [vgl. 5], [vgl. 6], [vgl. 7], [vgl. 8], [vgl. 9]. Abb. 1 veranschaulicht die unterschiedlichen Bauformen und Größenverhältnisse der in Tabelle 1 aufgeführten humanoiden Robotersysteme.

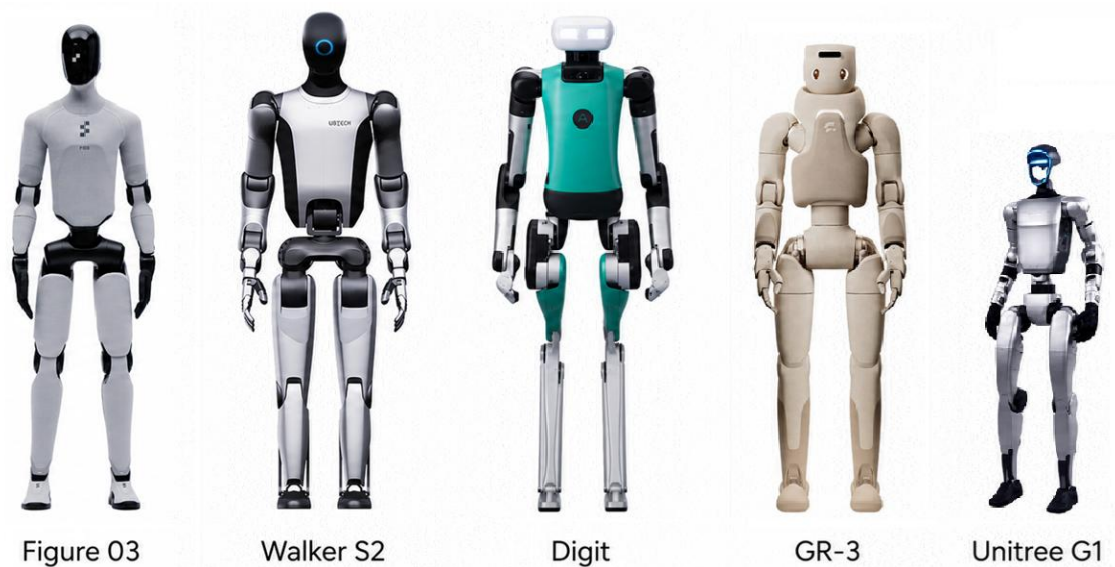


Abb. 1: Vergleich aktueller humanoider Robotersysteme (eigene Darstellung nach [vgl. 10], [vgl. 11], [vgl. 12], [vgl. 13], [vgl. 14])

Im Folgenden werden die in Tabelle 1 aufgeführten humanoiden Robotersysteme näher vorgestellt und ihre wesentlichen technischen Merkmale beschrieben.

Figure 03

Das US-amerikanische Unternehmen Figure AI entwickelt mit dem Figure 03 einen humanoiden Roboter für den Einsatz im häuslichen Umfeld. Ziel ist die autonome Ausführung alltäglicher Haushaltsaufgaben wie Wäsche waschen, Putzen oder Geschirrspülen [vgl. 5].

Das humanoide Robotersystem Figure 03 weist eine Tragfähigkeit von 20 kg sowie eine Gehgeschwindigkeit von 1,2 m/s auf. Die vom Hersteller angegebene Betriebsdauer liegt bei bis zu 5 h [vgl. 5].

Für den Einsatz im häuslichen Umfeld kommt die KI Helix zum Einsatz. Diese ermöglicht es dem Figure 03, aus alltäglichen Situationen zu lernen, sich an Veränderungen im Alltag anzupassen und in dynamischen Wohnumgebungen zu navigieren [vgl. 5], [vgl. 15].

Die Entwicklung des humanoiden Figure 03 zielt spezifisch auf die Fähigkeit zur selbstständigen Fortbewegung in Wohnräumen ab. Hierzu zählen unter anderem das

Bewältigen enger Kurven sowie das Navigieren in wechselnden Raumaufteilungen [vgl. 5].

Walker S2

Der Walker S2 ist ein humanoider Roboter des chinesischen Unternehmens UBTECH Robotics, welches sich neben humanoiden Robotern auch auf Service-Roboter konzentriert. Im Jahr 2025 führte das Unternehmen zudem ein kollaboratives Praxis-Trainingsprogramm für humanoide Robotersysteme durch, bei dem mehrere Roboter in industriellen Umgebungen gemeinsam unterschiedliche Aufgaben bearbeiteten [vgl. 16].

Der Walker S2 ist ein Modell, das über 52 Freiheitsgrade (engl. Degrees of Freedom, DoF) verfügt und eine Gehgeschwindigkeit von 2 m/s erreicht. Darüber hinaus ist der humanoide Roboter mit einem dynamischen Gleichgewichtsalgorithmus ausgestattet, welcher es ihm ermöglicht, Lasten von bis zu 15 kg innerhalb eines Arbeitsbereichs von 0 bis 1,8 m zu handhaben [vgl. 6].

Abhängig von der Aufgabenpriorität besteht für den Walker S2 die Möglichkeit, mittels Integration des Echtzeit-Batteriemanagements in die Batteriewechselstationen autonome Entscheidungen zwischen Batteriewechsel oder Lademodus zu treffen. Der Batteriewechsel kann dabei selbstständig vom Walker S2 durchgeführt werden, was dem Roboter einen kontinuierlichen Betrieb mit nur einem dreiminütigen Batteriewechsel ermöglicht [vgl. 6].

Zur Integration des Walker S2 in industrielle Fertigungslinien setzt der Hersteller auf ein Dual-Loop-KI-System. Dieses besteht aus der aufgabenorientierten Schwarmintelligenz BrainNet 2.0 sowie den sogenannten Co-Agenten. Durch diese Funktionen ist der Walker S2 insbesondere für den Einsatz in industriellen Fertigungsumgebungen konzipiert [vgl. 6].

Digit

Ein weiteres Beispiel für einen humanoiden Roboter, dessen Einsatzgebiet in der industriellen Umgebung liegt, ist der Digit. Dieser humanoide Roboter gehört zum US-amerikanischen Unternehmen Agility Robotics [vgl. 7].

Der Digit ist ein Roboter, der über 16 DoF verfügt, welche sich auf jeweils 5 DoF pro Fuß und 3 DoF pro Hand verteilen. Seine maximale Tragfähigkeit liegt bei 16 kg [vgl. 17].

Hinsichtlich der sensorischen Ausstattung ist zunächst ein LiDAR-Sensor zu nennen, ergänzt durch vier Intel RealSense-Tiefenkameras sowie Absolut- und Inkrementalgeber für die Propriozeption [vgl. 17].

Die Einbindung des Digit in bestehende Lager- und Automatisierungssysteme erfolgt über die cloudbasierte Plattform Arc. Diese ermöglicht unter anderem die Integration autonomer mobiler Roboter (AMRs) sowie bestehender Verwaltungs- und Ausführungssysteme. Der Digit wurde in einer menschenzentrierten Bauweise entwickelt und ist insbesondere für repetitive, gefährliche und körperlich belastende Tätigkeiten vorgesehen, bei denen Menschen Unterstützung benötigen [vgl. 7].

GR-3

Der GR-3 des chinesischen Unternehmens Fourier Robotics unterscheidet sich in Bezug auf sein äußeres Erscheinungsbild signifikant von den bereits genannten Robotersystemen. Anstelle einer metallischen Oberfläche wurde beim GR-3 ein Soft-Shell-Design gewählt. Die Außenhülle des Roboters kombiniert eine hochwertige Polsterung in Automobilqualität mit einer umweltfreundlichen Schaumstoffpolsterung. Diese Gestaltung zielt darauf ab, den Roboter sowohl hochfunktional zu gestalten als auch eine einladende Wirkung zu erzielen [vgl. 18].

Der GR-3 ist mit bis zu 55 DoF ausgestattet und verfügt über eine Batterielaufzeit von 3 h. Durch den Einsatz von zwei Akkus, die im laufenden Betrieb gewechselt werden können, wird jedoch ein unterbrechungsarmer Dauerbetrieb ermöglicht [vgl. 8].

Zudem ist der humanoide Roboter mit 31 verteilten Drucksensoren ausgestattet, welche es ihm erlauben, auf Berührungen zu reagieren. Das RGB- und Strukturlicht-Kameramodul ermöglicht eine zuverlässige Gesichtserkennung sowie die dynamische Verfolgung von Personen, sodass sich diese kontinuierlich im Sichtfeld des Roboters befinden. Der Einsatz eines Arrays aus vier Mikrofonen ermöglicht eine omnidirektionale Sprachaktivierung, eine Echounterdrückung sowie eine präzise Lokalisierung der

Schallquelle. Infolgedessen richtet sich der GR-3 durch synchronisierten Blickkontakt auf die sprechende Person aus, was zu einer Förderung der natürlichen sprachlichen Interaktion führt [vgl. 18].

Der GR-3 wurde entwickelt, um Menschen in unterschiedlichen Anwendungsbereichen zu unterstützen und gemeinsam mit ihnen zu interagieren. Dabei soll sich das Robotersystem an veränderte Anforderungen sowie komplexe Umgebungen anpassen können [vgl. 8].

Unitree G1

Der Unitree G1 ist ein humanoides Robotersystem des chinesischen Herstellers Unitree Robotics. Neben der Entwicklung von humanoiden und vierbeinigen Robotersystemen entwickelt das Unternehmen auch wesentliche Hardwarekomponenten sowie Algorithmen zur Bewegungssteuerung und Umgebungswahrnehmung [vgl. 19].

Grundsätzlich bietet der Hersteller zwei verschiedene Modelle des G1 an: den G1 und den G1 EDU. Dabei unterscheiden sich diese beiden Modelle in einigen technischen Aspekten, beispielsweise hinsichtlich der Leistung der Motoren sowie der möglichen Anzahl an DoF. Der primäre Unterschied besteht darin, dass es sich bei der Basic-Version des G1 um ein geschlossenes System handelt. Folglich ermöglicht diese Variante nur vorinstallierte Aktionen zur Fortbewegung sowie einige Handbewegungen wie Winken oder Klatschen [vgl. 9].

In der vorliegenden Arbeit wird das offene System G1 EDU untersucht. Die nachfolgenden Daten beziehen sich daher auf dieses Modell. Der humanoide Roboter verfügt je nach Konfiguration über 23 bis 43 DoF und eine Batterielaufzeit von etwa 2 h. Die maximale Traglast pro Hand ist mit 3 kg angegeben, wobei diese Angabe stark von der Position der Hände abhängt [vgl. 9].

Des Weiteren ist der Roboter mit dem 3D-LiDAR-Sensor LIVOX-MID360 sowie der Tiefenkamera Intel RealSense D435i ausgestattet [vgl. 9].

Der Unitree G1 unterstützt Verfahren des maschinellen Lernens (engl. Machine Learning, ML), insbesondere des Imitationslernens (engl. Imitation Learning, IL) sowie des

verstärkenden Lernens (engl. Reinforcement Learning, RL). Diese Verfahren bilden die Grundlage für zahlreiche Anwendungen und Entwicklungsansätze auf dem Robotersystem [vgl. 9]. In den später behandelten Kapiteln 2.2.1 Reinforcement Learning sowie 2.2.2 Imitation Learning wird näher auf diese Ansätze des ML eingegangen.

2.1.4 TECHNISCHE HERAUSFORDERUNGEN HUMANOIDER ROBOTERSYSTEME

Trotz der erheblichen Fortschritte in der Entwicklung humanoider Robotersysteme bestehen weiterhin zahlreiche technische Herausforderungen. Es besteht nach wie vor eine erhebliche Diskrepanz zwischen den aktuellen Fähigkeiten humanoider Roboter und den Fähigkeiten des Menschen [vgl. 3, S. 301].

Eine wesentliche Herausforderung besteht in der hohen technologischen Komplexität humanoider Robotersysteme. Die Realisierung natürlicher Bewegungen humanoider Robotersysteme erfordert den Einsatz fortschrittlicher Steuerungs- und Sensortechnologien. Insbesondere die stabile bipedale Fortbewegung sowie die menschenähnliche Interaktion stellen nach wie vor erhebliche technische Herausforderungen dar. Gleichzeitig stellen der hohe Energiebedarf sowie die begrenzte Kapazität der Energiespeicher Einschränkungen für längere Einsatzzeiten dar [vgl. 3, S. 314].

Neben den Herausforderungen im Bereich der Hardware gewinnt insbesondere die Entwicklung intelligenter Software zunehmend an Bedeutung. Zukünftige humanoide Robotersysteme sollen in der Lage sein, neue Fähigkeiten selbstständig zu erlernen, sich an veränderte Umgebungen anzupassen und Aufgaben mit einem höheren Grad an Autonomie auszuführen. In diesem Zusammenhang werden Verfahren der künstlichen Intelligenz (KI) und des ML als wesentliche Entwicklungsrichtungen betrachtet [vgl. 3, S. 316]. Die verschiedenen Ansätze zur Programmierung humanoider Robotersysteme werden im folgenden Kapitel näher erläutert.

2.2 METHODEN DER PROGRAMMIERUNG HUMANOIDER ROBOTERSYSTEME

Mit der zunehmenden Verlagerung humanoider Robotersysteme aus strukturierten Industrieumgebungen in komplexe, dynamische und unvorhersehbare Einsatzbereiche steigen die Anforderungen an deren Steuerung und Entscheidungsfindung. Roboter agieren in klassischen Fertigungsstraßen überwiegend unter konstanten Bedingungen. Demgegenüber werden humanoide Robotersysteme zunehmend in Umgebungen eingesetzt, die durch wechselnde Situationen, unbekannte Objekte sowie Interaktionen mit Menschen geprägt sind. Infolgedessen gewinnen Verfahren an Bedeutung, die eine flexible Anpassung an sich verändernde Umgebungsbedingungen ermöglichen [vgl. 20, S. 1].

Über einen signifikanten Zeitraum hinweg bildeten klassische regelungsbasierte Verfahren die Grundlage robotischer Steuerung. Diese basieren auf einem möglichst genauen mathematischen Modell des Roboters sowie seiner Umgebung und ermöglichen eine stabile Regelung, eine Optimierung der Systemleistung sowie ein vorhersehbares Verhalten. Insbesondere im Hinblick auf sicherheitskritische Anwendungen erweisen sich diese Eigenschaften als von erheblichem Vorteil. Die Leistungsfähigkeit dieser Verfahren hängt jedoch erheblich von der Genauigkeit des zugrunde liegenden Modells ab. Weichen reale Bedingungen, beispielsweise durch Reibung, äußere Störungen oder unvorhersehbare Umwelteinflüsse, von den modellierten Annahmen ab, kann dies zu einer deutlichen Verschlechterung der Regelgüte führen [vgl. 20, S. 1-2].

Infolge der signifikanten Fortschritte in den Bereichen KI und ML haben sich datengetriebene Verfahren zunehmend in Kombination mit klassischen Regelungsansätzen etabliert. Der aktuelle Stand der Forschung wird dabei insbesondere durch die synergetische Verbindung beider Ansätze geprägt. Im Gegensatz zu modellbasierten Verfahren lernen diese Ansätze Verhaltensweisen direkt aus Daten oder durch Interaktion mit ihrer Umgebung und können dadurch komplexe, nichtlineare Zusammenhänge erfassen, ohne dass hierfür ein vollständiges mathematisches Modell erforderlich ist. Lernbasierte Verfahren sind jedoch auch mit Nachteilen verbunden. Aufgrund ihrer hohen Komplexität sind die zugrunde liegenden Modelle häufig nur eingeschränkt interpretierbar und werden daher oftmals als sogenannte Black-Box-

Modelle bezeichnet. Darüber hinaus erfordern sie in der Regel große Datenmengen und können im Gegensatz zu klassischen Regelungsverfahren die Stabilität und Sicherheit eines Systems nicht gleichermaßen gewährleisten. Insbesondere in sicherheitskritischen Anwendungen stellt dies eine wesentliche Herausforderung dar [vgl. 20, S. 2].

In den folgenden Kapiteln werden RL, IL sowie Vision-Language-Action-Modelle (VLA-Modelle) als Ansätze zur Programmierung humanoider Robotersysteme vorgestellt und hinsichtlich ihrer Funktionsweise näher erläutert.

2.2.1 REINFORCEMENT LEARNING

RL ist ein Verfahren des maschinellen Lernens, das zur Lösung sequenzieller Entscheidungsprobleme in Umgebungen mit unbekannter Dynamik eingesetzt wird [vgl. 20, S. 11]. Die für den Lernprozess benötigten Daten werden dabei nicht im Voraus bereitgestellt. Stattdessen sammelt der Agent diese eigenständig durch die Interaktion mit seiner Umgebung [vgl. 21, S. 17].

Während dieser Interaktion nimmt der Agent den aktuellen Zustand der Umgebung wahr und führt darauf basierend Aktionen aus. Das Ziel besteht dabei darin, eine geeignete Steuerungs- beziehungsweise Entscheidungsstrategie (engl. Policy) zu erlernen [vgl. 22, S. 500], [vgl. 23, S. 4]. Die Bewertung der ausgeführten Aktionen erfolgt durch Rückmeldungen in Form von Belohnungen (engl. Rewards). Anhand dieser Rückmeldungen wird die Policy schrittweise angepasst, um bei zukünftigen Entscheidungen einen möglichst hohen Reward zu erzielen [vgl. 21, S. 17], [vgl. 23, S. 4].

Der Lernprozess erfolgt iterativ. Nach jeder Interaktion werden die während der Zustandsübergänge gesammelten Informationen genutzt, um die Policy kontinuierlich zu verbessern. Dieser Vorgang wird wiederholt, bis eine möglichst gute beziehungsweise optimale Strategie für die jeweilige Aufgabe erlernt wurde [vgl. 24, S. 4], [vgl. 25, S. 3].

2.2.1.1 GRUNDPRINZIP

Das Grundprinzip des Reinforcement Learning basiert auf der Interaktion zwischen einem Agenten und seiner Umgebung. Diese Interaktion wird durch mehrere zentrale Komponenten beschrieben. Zu diesen zählen insbesondere der Agent, die Umgebung

(engl. Environment), der Zustand (engl. State), die Aktion (engl. Action), der Reward sowie die Policy [vgl. 26, S. 47-48]. Das Zusammenspiel dieser Komponenten ist in Abb. 2 dargestellt. Die Bedeutung der einzelnen Komponenten wird im Folgenden näher erläutert.

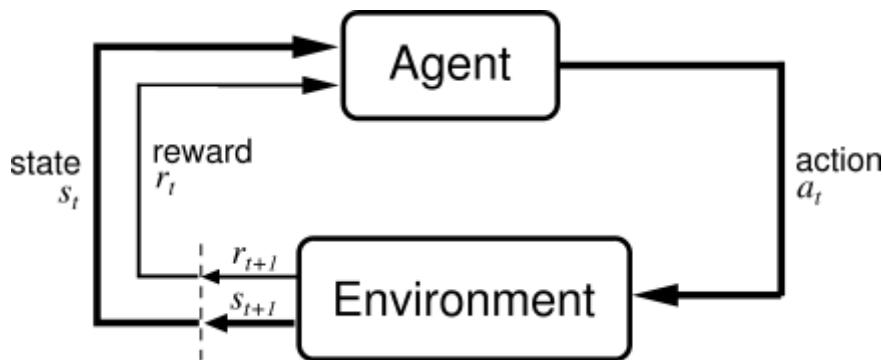


Abb. 2: Agent-Environment Interaction im RL [26, S. 48]

Agent

Als Agent wird ein System bezeichnet, das mit seiner Umgebung interagiert, um spezifische Aufgaben zu erfüllen. Der Agent nimmt dabei seine Umgebung wahr und wirkt auch auf sie ein. Dabei besteht seine Aufgabe darin, aus den ihm zur Verfügung stehenden Aktionen für die jeweilige Situation möglichst geeignete Entscheidungen zu treffen. Diese sollen im Hinblick auf die zu erfüllende Aufgabe effizient und effektiv sein. Jede ausgeführte Aktion beeinflusst die weitere Entwicklung der Umgebung und führt zu einer neuen Situation, auf deren Grundlage der Agent erneut Entscheidungen treffen muss [vgl. 23, S. 14].

Environment

Als Environment wird jenes System bezeichnet, mit dem der Agent interagiert. Das Environment beschreibt den Bereich, in dem der Agent agiert und mit seiner Umgebung wechselwirkt. Jede ausgeführte Action führt zu einem neuen State des Environment. Der Agent erhält zudem eine Rückmeldung in Form eines Rewards, der die Qualität der zuvor getroffenen Entscheidung bewertet. Diese Rückmeldung bildet die Grundlage für die schrittweise Anpassung der Policy [vgl. 27, S. 33].

State

Der State beschreibt den aktuellen Zustand des Environment und umfasst sämtliche Informationen, die diesen charakterisieren [vgl. 27, S. 33]. In realen Anwendungen kann der vollständige State jedoch häufig nicht direkt erfasst werden [vgl. 24, S. 11]. Stattdessen basiert die Entscheidungsfindung des Agenten auf Beobachtungen (engl. Observations), anhand derer der aktuelle State abgeschätzt wird. Je nach Anwendungsgebiet kann ein State unterschiedliche Informationen enthalten, beispielsweise Gelenkpositionen, Geschwindigkeiten, Drehmomente oder Bilddaten [vgl. 27, S. 33].

Action

Als Action wird eine vom Agenten ausgeführte Handlung bezeichnet, mit der dieser auf das Environment einwirkt. Zu jedem Zeitpunkt wählt der Agent auf Grundlage des aktuellen State eine Action aus [vgl. 26, S. 47-48]. Für jeden State steht dem Agenten eine Menge möglicher Aktionen zur Verfügung, was auch als Action Space bezeichnet wird. Dieser kann je nach Anwendungsfall diskrete oder kontinuierliche Aktionen umfassen [vgl. 24, S. 5]. Die gewählte Action beeinflusst den weiteren Verlauf des Lernprozesses, da sie zu einem neuen State sowie einem Reward führt [vgl. 26, S. 47-48].

Reward

Der Reward repräsentiert ein vom Environment an den Agenten übermitteltes Signal, welches das Ziel des Lernprozesses beschreibt. Nach jeder ausgeführten Action erhält der Agent einen Reward, der die Qualität der zuvor ausgeführten Handlung bewertet. Das Ziel des Agenten besteht jedoch nicht darin, ausschließlich den unmittelbar erhaltenen Reward zu maximieren, sondern die Summe der langfristig erzielten Rewards. Der Reward definiert das Lernziel des Agenten, gibt jedoch nicht vor, auf welche Weise dieses Ziel erreicht werden soll [vgl. 26, S. 51-52].

2.2.1.2 LERNPROZESS

Basierend auf den im vorherigen Abschnitt erläuterten Komponenten des RL wird im Folgenden der zugrunde liegende Lernprozess beschrieben. Dieser Ansatz basiert auf

einer fortlaufenden Interaktion zwischen Agent und Environment, bei der die Policy kontinuierlich angepasst wird [vgl. 26, S. 48].

Zu Beginn eines jeden Lernschrittes erhält der Agent Informationen über den aktuellen State des Environment. Auf Grundlage seiner aktuellen Policy wählt der Agent anschließend eine Action und führt diese im Environment aus. Als Reaktion auf die ausgeführte Action verändert sich der State des Environment und der Agent erhält einen Reward. Dieser beschreibt die Qualität der zuvor ausgeführten Action und bildet die Grundlage für den weiteren Lernprozess [vgl. 26, S. 47-48].

Die während der Interaktion gewonnenen Erfahrungen werden zur schrittweisen Anpassung der Policy genutzt. Diese Vorgehensweise ermöglicht es dem Agenten, seine Entscheidungsstrategie kontinuierlich zu optimieren. Dadurch lernt der Agent, in unterschiedlichen Situationen zunehmend geeignete Actions auszuwählen [vgl. 26, S. 47-48].

Das Ziel des RL besteht demnach darin, eine Policy zu erlernen, mit der die langfristig kumulierten Rewards maximiert werden [vgl. 26, S. 48].

Zusammenfassend lässt sich der Lernprozess als fortlaufender Zyklus zwischen Agent und Environment beschreiben. Der Agent beobachtet zunächst den aktuellen State des Environment. Anschließend wählt er auf Grundlage seiner Policy eine Action aus und führt diese im Environment aus. Als Reaktion darauf erhält der Agent einen Reward und das Environment geht in einen neuen State über. Diese Rückmeldung wird genutzt, um die Policy kontinuierlich anzupassen. Durch die wiederholte Ausführung dieses Zyklus verbessert sich die Entscheidungsstrategie des Agenten schrittweise.

2.2.2 IMITATION LEARNING

Mit der zunehmenden Verlagerung robotischer Systeme in komplexe und unstrukturierte Umgebungen steigen die Anforderungen an deren Programmierung. Insbesondere die manuelle Programmierung komplexer Aufgaben sowie die Definition geeigneter Reward-Funktionen, wie sie beim RL erforderlich sind, gestalten sich in solchen Einsatzbereichen zunehmend schwierig. Für Aufgaben, die von Menschen intuitiv ausgeführt werden können, deren algorithmische Beschreibung jedoch komplex ist, bietet das IL einen

geeigneten Lösungsansatz. Im Gegensatz zum RL, bei dem der Agent geeignete Verhaltensweisen durch Trial-and-Error und die Maximierung von Rewards erlernt, basiert das IL auf Demonstrationen einer Expertin bzw. eines Experten [vgl. 28, S. 1], [vgl. 29, S. 7173], [vgl. 26, S. 7].

Beim IL lernt der Agent das gewünschte Verhalten anhand von Demonstrationen einer Expertin bzw. eines Experten. Die Demonstrationen dienen dabei als Trainingsgrundlage, aus der eine Policy erlernt wird. Die Expertin bzw. der Experte muss hierfür nicht in der Lage sein, das gewünschte Verhalten selbst zu programmieren. Stattdessen lernt der Agent, das demonstrierte Verhalten anhand der bereitgestellten Demonstrationen nachzuahmen [vgl. 28, S. 1], [vgl. 29, S. 7173].

Das IL umfasst verschiedene Verfahren, die sich hinsichtlich ihrer Herangehensweise an den Lernprozess unterscheiden. Eine grundlegende Methode stellt das Verhaltensklonen (engl. Behavioral Cloning, BC) dar. Dabei wird die Policy unmittelbar aus den Demonstrationen einer Expertin bzw. eines Experten erlernt [vgl. 28, S. 1], [vgl. 29, S. 7173]. Das Grundprinzip dieses Verfahrens wird im folgenden Abschnitt näher erläutert.

2.2.2.1 BEHAVIORAL CLONING

BC stellt einen grundlegenden Ansatz des IL dar, bei dem eine Policy anhand von Demonstrationen einer Expertin bzw. eines Experten erlernt wird. Ziel des Verfahrens ist es, das demonstrierte Verhalten möglichst genau nachzubilden, sodass der Agent in vergleichbaren Situationen geeignete Actions auswählen kann. Hierzu werden während der Demonstrationen aufgezeichnete Zustands-Aktions-Paare verwendet, anhand derer die Policy schrittweise erlernt wird [vgl. 30, S. 105-107].

Im Gegensatz zum RL erfolgt der Lernprozess dabei nicht durch die Interaktion mit dem Environment und die Maximierung eines Rewards. Stattdessen basiert BC auf einem Ansatz des überwachten Lernens (engl. Supervised Learning), bei dem die Demonstrationen der Expertin bzw. des Experten als Grundlage für das Training dienen. Während des Trainings werden die Parameter der Policy schrittweise angepasst, um die Abweichung zwischen den vom Agenten vorhergesagten Actions und den durch die Expertin bzw. den Experten demonstrierten Actions zu minimieren. Auf diese Weise wird

eine Policy erlernt, die das Expertenverhalten möglichst genau approximiert [vgl. 30, S. 105-108].

Da BC den Lernprozess als Problem des Supervised Learning formuliert, werden im folgenden Abschnitt die grundlegenden Eigenschaften dieses Lernverfahrens näher erläutert.

Supervised Learning

Supervised Learning ist ein Verfahren des maschinellen Lernens, bei dem anhand gelabelter Trainingsdaten eine Beziehung zwischen Eingabe- und Ausgabedaten erlernt wird. Jeder Trainingsdatensatz besteht aus einem Eingabewert sowie dem dazugehörigen Zielwert, welcher als Label bezeichnet wird. Ziel des Lernprozesses ist es, ein Modell zu entwickeln, das den Zusammenhang zwischen Eingaben und Ausgaben erlernt und diesen anschließend zur Vorhersage von Ausgaben für neue Eingabedaten nutzen kann [vgl. 31].

Während des Trainings werden die Eingabedaten dem Lernsystem übergeben, welches daraus zunächst eine Vorhersage erzeugt. Anschließend wird diese Vorhersage mit dem bekannten Zielwert verglichen. Die dabei ermittelte Abweichung dient als Fehlersignal und wird verwendet, um die Modellparameter schrittweise anzupassen. Dieser Vorgang wird für alle Trainingsdaten wiederholt, bis die Gesamtabweichung möglichst gering ist [vgl. 31].

Ein geringerer Fehler während des Trainings führt jedoch nicht zwangsläufig zu einer besseren Leistung bei unbekannten Daten. Wird das Modell zu stark an die Trainingsdaten angepasst, kann es zu einem sogenannten Overfitting kommen. Daher besteht das Ziel des Trainings nicht nur darin, den Fehler auf den Trainingsdaten zu minimieren, sondern gleichzeitig eine gute Generalisierungsfähigkeit für bisher unbekannte Eingaben zu erreichen [vgl. 31].

Im Rahmen des BC entsprechen die beobachteten States den Eingabedaten, während die von der Expertin bzw. dem Experten ausgeführten Actions die zugehörigen Zielwerte darstellen. Dadurch kann das Erlernen einer Policy als Problem des Supervised Learning formuliert werden.

Herausforderungen des Behavioral Cloning

BC weist jedoch auch gewisse Einschränkungen auf. Die größte Herausforderung stellt dabei das sogenannte Covariate-Shift-Problem dar. Dieses beschreibt den Unterschied zwischen der Verteilung der während des Trainings beobachteten States und jener States, welche der Agent während der späteren Ausführung selbst erzeugt. Während die Demonstrationen der Expertin bzw. des Experten die Trainingsdaten bilden, entstehen während der Ausführung neue Zustände durch die Entscheidungen der erlernten Policy. Da Demonstrationen in der Regel nur einen begrenzten Teil aller möglichen Situationen einer Aufgabe abdecken, trifft der Agent im späteren Einsatz häufig auf States, die in den Trainingsdaten nicht enthalten sind [vgl. 28, S. 11].

Die Ursachen für das Covariate-Shift-Problem sind vielfältig. Zum einen basiert das Training lediglich auf einer endlichen Anzahl an Demonstrationen, wodurch nicht sämtliche möglichen Situationen einer Aufgabe erfasst werden können. Darüber hinaus kann die erlernte Policy das Verhalten der Expertin bzw. des Experten aufgrund einer begrenzten Modellkapazität nicht vollständig reproduzieren. Auch stochastische Umgebungsbedingungen, Sensorrauschen, unvollständige Beobachtungen oder Ungenauigkeiten bei der Ausführung von Actions können dazu führen, dass der Agent bereits früh von den demonstrierten Trajektorien abweicht. Zusätzlich sind Expertendemonstrationen selbst nicht zwangsläufig fehlerfrei, sodass auch suboptimale oder inkonsistente Verhaltensweisen erlernt werden können [vgl. 28, S. 11], [vgl. 30, S. 110-111].

Bereits kleine Abweichungen können dazu führen, dass der Agent States erreicht, welche in den Demonstrationen nicht oder nur unzureichend vertreten sind. Für diese Situationen besitzt die erlernte Policy keine ausreichende Entscheidungsgrundlage, wodurch die Wahrscheinlichkeit weiterer Fehlentscheidungen steigt. Infolgedessen entfernt sich der Agent zunehmend von der ursprünglich demonstrierten Trajektorie, wodurch sich einzelne Fehler im weiteren Verlauf der Ausführung fortlaufend akkumulieren und die Leistungsfähigkeit der Policy schrittweise verschlechtern [vgl. 30, S. 110-111]. Die Auswirkungen dieses Verhaltens sind beispielsweise in Abb. 3 dargestellt.

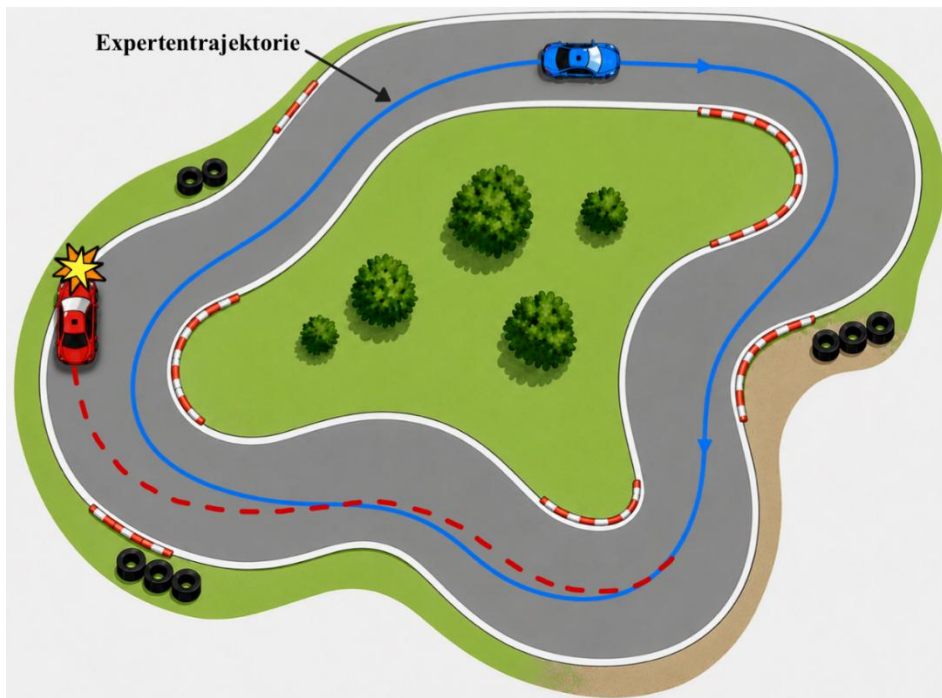


Abb. 3: Auswirkungen des Covariate-Shift-Problems (eigene Darstellung, erstellt mit ChatGPT GPT-5.5 nach [vgl. 32])

In Abb. 3 folgt das blaue Fahrzeug der durch Demonstrationen vorgegebenen Expertentrajektorie. Das rote Fahrzeug repräsentiert den Agenten, welcher versucht, dieses Fahrverhalten nachzubilden. Bereits eine geringe Abweichung von der Expertentrajektorie führt dazu, dass der Agent den durch Demonstrationen abgedeckten Bereich verlässt. Da für diese Situation keine geeigneten Demonstrationen vorliegen, kann der Agent seine Trajektorie nicht mehr zur Expertentrajektorie zurückführen. Dadurch vergrößert sich die Abweichung kontinuierlich, bis die Aufgabe schließlich nicht mehr erfolgreich ausgeführt werden kann.

Das Covariate-Shift-Problem stellt somit eine der größten Herausforderungen des klassischen BC dar. Um die Robustheit der erlernten Policy gegenüber bislang unbekannten Situationen zu erhöhen und die Akkumulation von Fehlern zu reduzieren, wurden verschiedene Erweiterungen des BC entwickelt. Eine der bekanntesten Methoden stellt Dataset Aggregation (DAGger) dar, dessen Grundprinzip im folgenden Abschnitt näher erläutert wird [vgl. 30, S. 110-111].

Dataset Aggregation

DAGger stellt eine der bekanntesten Erweiterungen des klassischen BC dar. Das Verfahren erweitert den BC-Ansatz um einen interaktiven Trainingsprozess, der auf der aktuellen Policy basiert und mit dem die Auswirkungen des Covariate-Shift-Problems reduziert werden sollen. Im Gegensatz zum klassischen BC wird die Policy nicht ausschließlich anhand eines festen Datensatzes aus Expertendemonstrationen trainiert. Stattdessen interagiert der Agent während des Trainings wiederholt mit dem Environment unter Verwendung seiner aktuellen Policy und erzeugt dadurch eigene Trajektorien. Für die dabei besuchten States stellt die Expertin bzw. der Experte jeweils die korrekte Action bereit. Diese zusätzlichen Zustands-Aktions-Paare werden anschließend gemeinsam mit den ursprünglichen Demonstrationen in den Trainingsdatensatz aufgenommen und für das erneute Training der Policy verwendet [vgl. 29, S. 7174-7175], [vgl. 30, S. 111-115].

Durch die schrittweise Erweiterung des Trainingsdatensatzes lernt der Agent auch jene States kennen, die während der autonomen Ausführung durch seine eigenen Entscheidungen entstehen. Dadurch wird die Verteilung der Trainingsdaten zunehmend an jene States angepasst, die der Agent später tatsächlich besucht. Dies verbessert die Robustheit der erlernten Policy gegenüber unbekannten Situationen und reduziert die Auswirkungen des Covariate-Shift-Problems [vgl. 29, S. 7174-7175], [vgl. 30, S. 111-115].

Ein Nachteil von DAGger besteht jedoch darin, dass während des gesamten Trainingsprozesses eine Expertin bzw. ein Experte verfügbar sein muss, um die durch den Agenten besuchten States fortlaufend mit den korrekten Actions zu annotieren. Diese kontinuierliche Einbindung stellt insbesondere bei menschlichen Expertinnen bzw. Experten eine erhebliche Belastung dar und ist in vielen praktischen Anwendungen nur eingeschränkt realisierbar. Zudem können häufige Expertenabfragen zu verzögerten oder ungenauen Rückmeldungen führen und dadurch den Trainingsprozess beeinträchtigen [vgl. 29, S. 7174-7175], [vgl. 30, S. 111-115].

Obwohl DAGger die Auswirkungen des Covariate-Shift-Problems reduzieren kann, basiert das Verfahren weiterhin auf der direkten Nachbildung des Expertenverhaltens. Einen grundlegend anderen Ansatz verfolgt das Inverse Reinforcement Learning (IRL). Dabei wird nicht die Policy selbst, sondern zunächst die zugrunde liegende Reward-

Funktion aus den Demonstrationen erlernt [vgl. 32]. Das Grundprinzip dieses Ansatzes wird im folgenden Abschnitt erläutert.

2.2.2.2 INVERSE REINFORCEMENT LEARNING

Im Gegensatz zum BC wird beim IRL nicht die Policy der Expertin bzw. des Experten direkt erlernt. Stattdessen versucht der Agent, aus den Demonstrationen der Expertin bzw. des Experten die zugrunde liegende Reward-Funktion zu rekonstruieren. Dabei wird angenommen, dass das beobachtete Verhalten durch die Optimierung dieser unbekannten Reward-Funktion entstanden ist. Anschließend dient die rekonstruierte Reward-Funktion als Grundlage, um mithilfe von RL eine geeignete Policy zu erlernen [vgl. 29, S. 7176-7178], [vgl. 30, S. 116-118].

Das Grundprinzip des IRL besteht demnach aus zwei aufeinanderfolgenden Lernschritten. Zunächst werden die Demonstrationen der Expertin bzw. des Experten genutzt, um eine Reward-Funktion zu bestimmen, welche das beobachtete Verhalten möglichst gut beschreibt. Anschließend wird diese Reward-Funktion innerhalb eines RL-Verfahrens verwendet, um eine Policy zu erlernen, die langfristig einen möglichst hohen kumulativen Reward erzielt. Da Reward-Funktion und Policy iterativ optimiert werden, wechseln sich Reward-Schätzung und Policy-Training während des Lernprozesses mehrfach ab [vgl. 30, S. 116-118].

Während BC die demonstrierte Policy unmittelbar anhand von Zustands-Aktions-Paaren nachbildet, versucht IRL zunächst, die zugrunde liegende Zielsetzung der Expertin bzw. des Experten in Form einer Reward-Funktion zu bestimmen. Dadurch soll nicht ausschließlich das beobachtete Verhalten reproduziert werden, sondern die den Entscheidungen zugrunde liegende Zielsetzung erfasst werden [vgl. 29, S. 7176-7178], [vgl. 30, S. 116-118].

Ein wesentlicher Vorteil des IRL besteht darin, dass der Agent durch das nachgelagerte RL eigenständig mit der Umgebung interagieren und dadurch Erfahrungen sammeln kann. Dadurch gilt IRL als robuster gegenüber dem Covariate-Shift-Problem und kann besser auf bislang unbekannte Situationen reagieren als BC [vgl. 29, S. 7176-7178].

Demgegenüber ist IRL mit einem deutlich höheren Rechenaufwand verbunden. Die wiederholte Schätzung der Reward-Funktion sowie das anschließende Policy-Training führen zu einer geringen Sample-Effizienz und erfordern zahlreiche Interaktionen mit der Umgebung. Darüber hinaus ist die Rekonstruktion der Reward-Funktion grundsätzlich nicht eindeutig, da dieselbe optimale Policy durch unterschiedliche Reward-Funktionen beschrieben werden kann [vgl. 29, S. 7176-7178], [vgl. 30, S. 116-118].

BC und IRL stellen zwei grundlegende Ansätze des IL dar, welche beide das Lernen aus Demonstrationen verfolgen [vgl. 29, S. 7174-7178]. Mit der zunehmenden Entwicklung leistungsfähiger Sprach- und Bildverarbeitungsmodelle wurden diese klassischen Lernverfahren um multimodale Ansätze erweitert. VLA-Modelle kombinieren die Verarbeitung visueller Informationen und natürlichsprachlicher Anweisungen mit der Generierung von Roboteraktionen und eröffnen dadurch neue Möglichkeiten für die Generalisierung auf unterschiedliche Aufgaben und Umgebungen. Das Grundprinzip dieser Modelle wird im folgenden Abschnitt erläutert [vgl. 33].

2.2.3 VISION-LANGUAGE-ACTION-MODELLE

Die rasante Entwicklung großer Sprachmodelle (engl. Large Language Models, LLMs) sowie Vision-Language-Modelle (VLMs) hat in den vergangenen Jahren erhebliche Fortschritte in der Verarbeitung natürlicher Sprache und visueller Informationen ermöglicht. Diese Entwicklungen wurden zunehmend auf robotische Systeme übertragen, wodurch Roboter Aufgaben nicht mehr ausschließlich anhand fester Bewegungsabläufe ausführen, sondern ihre Umgebung wahrnehmen, Anweisungen verstehen und situationsabhängig darauf reagieren können. Frühere Ansätze trennten die Verarbeitung visueller und sprachlicher Informationen jedoch häufig von der eigentlichen Aktionsgenerierung. Dadurch war ihre Generalisierungsfähigkeit auf neue Aufgaben begrenzt. Zur Überwindung dieser Einschränkungen wurden VLA-Modelle entwickelt [vgl. 33].

VLA-Modelle sind multimodale Systeme, die visuelle Beobachtungen und natürlichsprachliche Anweisungen als Eingaben verarbeiten und daraus unmittelbar Roboteraktionen in Form von Steuerbefehlen erzeugen. Neben Bild- und Sprachinformationen können bei Bedarf weitere Sensordaten in das Modell integriert

werden. Ziel ist es, generalisierbare Policies zu erlernen, die unterschiedliche Aufgaben, Objekte und Einsatzumgebungen mit einem gemeinsamen Modell bearbeiten können [vgl. 33].

2.2.3.1 GRUNDPRINZIP

Das Grundprinzip von Vision-Language-Action-Modellen basiert auf der gemeinsamen Verarbeitung der drei Modalitäten Vision, Language und Action. Während die visuelle Modalität Informationen über die Umgebung bereitstellt, beschreibt die Sprachmodalität die auszuführende Aufgabe. Die Aktionsmodalität dient schließlich dazu, aus den verarbeiteten Informationen konkrete Roboteraktionen abzuleiten. Die gleichzeitige Verarbeitung dieser Modalitäten ermöglicht es, sprachliche Anweisungen mit den wahrgenommenen Umweltinformationen zu verknüpfen und daraus direkt ausführbare Steuerbefehle für den Roboter zu erzeugen. Im Gegensatz zu klassischen Vision-Language-Modellen endet die Verarbeitung somit nicht bei einer semantischen Interpretation der Eingaben, sondern führt unmittelbar zur Erzeugung ausführbarer Aktionen [vgl. 33], [vgl. 34, S. 1].

Vision

Die Vision-Komponente dient der visuellen Wahrnehmung der Umgebung und stellt dem VLA-Modell Informationen über den aktuellen Zustand der Szene bereit. Für Manipulationsaufgaben umfasst dies insbesondere Informationen über die Position, Form und Anordnung von Objekten. Diese visuellen Informationen bilden die Grundlage für die weitere Verarbeitung innerhalb des Modells und ermöglichen es, die aktuelle Umgebung bei der Planung und Ausführung von Roboteraktionen zu berücksichtigen [vgl. 35, S. 13].

Zur Verarbeitung dieser visuellen Informationen setzen aktuelle VLA-Modelle überwiegend vortrainierte Bildencoder ein. Häufig verwendete Modelle sind ResNet und Vision Transformer (ViT). Darüber hinaus werden neben klassischen Bildencodern zunehmend VLM wie CLIP oder SigLIP eingesetzt. Diese Modelle lernen gemeinsame Repräsentationen für Bild- und Sprachinformationen und erleichtern dadurch die spätere Verknüpfung beider Modalitäten. Ergänzend werden selbstüberwacht trainierte Modelle

wie DINOv2 genutzt, welche ohne gepaarte Bild-Text-Daten aussagekräftige visuelle Merkmale lernen können [vgl. 33].

Da die ausschließliche Nutzung visueller Informationen die Fähigkeit eines Roboters zur Interpretation komplexer Aufgaben begrenzt, ergänzen VLA-Modelle die visuelle Modalität um natürlichsprachliche Anweisungen [vgl. 35, S. 13].

Language

Die Language-Komponente verarbeitet natürlichsprachliche Anweisungen, welche die vom Roboter auszuführende Aufgabe beschreiben. Die Spracheingaben werden zunächst in einzelne Token zerlegt und anschließend in Vektorrepräsentationen überführt. Diese Repräsentationen dienen dazu, die semantische Bedeutung der Instruktion für die weitere Verarbeitung innerhalb des Modells bereitzustellen. Moderne VLA-Modelle übernehmen hierfür häufig die Sprachverarbeitung etablierter LLMs [vgl. 33], [vgl. 34, S. 1-2].

In sprachgesteuerten Robotikaufgaben ermöglicht die Language-Komponente dem Modell, natürlichsprachliche Anweisungen zu verstehen und gemeinsam mit den visuellen Informationen zu verarbeiten. Dadurch lassen sich Aufgaben in natürlicher Sprache beschreiben und unter Berücksichtigung der wahrgenommenen Umgebung in geeignete Roboteraktionen überführen. Die gemeinsame Verarbeitung von Sprach- und Bildinformationen stellt somit eine wesentliche Grundlage von VLA-Modellen dar [vgl. 34, S. 1-2].

Die Kombination aus visuellen Informationen und natürlichsprachlichen Anweisungen ermöglicht dem Modell zwar das Verständnis einer Aufgabe, führt jedoch noch nicht zur Ausführung einer Handlung. Erst die Action-Komponente übersetzt diese Informationen in konkrete Roboteraktionen.

Action

Die Action-Komponente unterscheidet VLA-Modelle grundlegend von klassischen VLMs. Während diese in der Regel lediglich sprachliche oder visuelle Ausgaben erzeugen, generieren VLA-Modelle unmittelbar Roboteraktionen beziehungsweise Steuerbefehle. Hierzu werden die gemeinsam verarbeiteten Informationen aus Vision und

Language genutzt, um Aktionen für den aktuellen Zustand des Roboters abzuleiten [vgl. 33], [vgl. 34, S. 1-2].

Für die Repräsentation von Roboteraktionen existieren unterschiedliche Ansätze. Eine häufig verwendete Methode besteht darin, Aktionen zunächst als diskrete Token zu repräsentieren. Diese werden anschließend mithilfe eines Decoders, beispielsweise eines Multi-Layer-Perceptrons (MLP), wieder in kontinuierliche Roboteraktionen überführt. Darüber hinaus existieren nicht-autoregressive Ansätze, welche mehrere Aktions-Token zu einer gemeinsamen Aktionsrepräsentation zusammenfassen und auch mehrschrittige Aktionsfolgen vorhersagen können [vgl. 33].

Neuere VLA-Modelle verwenden zunehmend Diffusionsmodelle oder Flow-Matching-Verfahren zur Generierung kontinuierlicher Roboteraktionen. Während Diffusionsmodelle schrittweise Aktionen erzeugen, verfolgt Flow Matching das Ziel, diesen Prozess effizienter zu gestalten und dadurch insbesondere den Einsatz in Echtzeitanwendungen zu erleichtern [vgl. 33].

Eine Herausforderung besteht darin, dass Roboter aufgrund unterschiedlicher Bauformen verschiedene Aktionsräume und Sensorinformationen besitzen. Dadurch sind die gemeinsame Nutzung von Trainingsdaten sowie die Übertragung gelernter Fähigkeiten zwischen unterschiedlichen Robotersystemen erschwert. Aktuelle Forschungsarbeiten begegnen diesem Problem beispielsweise durch die Entwicklung einheitlicher Aktionsräume oder den Einsatz roboterspezifischer Aktionsdecoder. Ziel ist es, die gemeinsame Nutzung von Daten verschiedener Robotersysteme zu verbessern und die Wiederverwendung bereits erlernten Wissens über unterschiedliche Robotersysteme hinweg zu ermöglichen [vgl. 33].

2.2.3.2 FUNKTIONSWEISE

Die Funktionsweise eines VLA-Modells beginnt mit der Erfassung der Eingaben. Hierzu zählen insbesondere Kamerabilder der aktuellen Umgebung sowie eine natürlichsprachliche Aufgabenbeschreibung. Die Bilddaten werden durch einen Vision Encoder verarbeitet, während die Sprachinstruktion in eine für das Modell verarbeitbare Repräsentation überführt wird. Abhängig vom Modell können zusätzlich weitere

Informationen, beispielsweise Propriozeption oder andere Sensordaten, berücksichtigt werden [vgl. 33], [vgl. 34, S. 1-2].

Die verarbeiteten Eingaben dienen anschließend als Grundlage für die Aktionsgenerierung. Hierzu werden die Informationen aus den verschiedenen Modalitäten gemeinsam verarbeitet, sodass die Sprachinstruktion unter Berücksichtigung der wahrgenommenen Umgebung in eine geeignete Handlung umgesetzt werden kann. Je nach Modell erfolgt diese Verarbeitung beispielsweise durch einen Transformer oder unter Verwendung eines vortrainierten VLM [vgl. 33].

Auf Grundlage dieser gemeinsamen Repräsentation wird schließlich eine Aktionsrepräsentation erzeugt und durch einen Action Decoder in ausführbare Roboteraktionen überführt. Die konkrete Umsetzung hängt von der jeweiligen Modellarchitektur ab. So können Aktionen beispielsweise über diskrete Aktions-Token, Diffusionsmodelle oder Flow-Matching-Verfahren erzeugt werden [vgl. 33].

2.2.4 VERGLEICH DER PROGRAMMIERMETHODEN

RL, IL und VLA-Modelle verfolgen unterschiedliche Ansätze zur Erzeugung robotischer Bewegungsstrategien. Während RL und IL eigenständige Lernparadigmen darstellen, kombinieren VLA-Modelle visuelle Informationen, Sprachinstruktionen und Aktionen innerhalb eines gemeinsamen Modells zur Generierung von Roboteraktionen [vgl. 26, S. 1-7], [vgl. 29, S. 7173-7174], [vgl. 33]. Zur besseren Einordnung werden die drei Ansätze in Tabelle 2 anhand zentraler Eigenschaften gegenübergestellt.

Tabelle 2: Vergleich von RL, IL und VLA (eigene Darstellung nach [vgl. 25, S. 1-10], [vgl. 26, S. 1-6], [vgl. 28, S. 3-5], [vgl. 29, S. 7173-7174], [vgl. 33], [vgl. 34, S. 1-3])

Kriterien	RL	IL	VLA
Lernprinzip	Lernen durch Belohnung	Lernen aus Demonstrationen	Lernen aus Vision-, Sprach- und Aktionsdaten
Trainingsinformation	Belohnungssignal	Expertendaten	Bild- und Sprachinformationen

Kriterien	RL	IL	VLA
Datenquelle	Interaktion mit der Umgebung	Demonstrationen einer Expertin bzw. eines Experten	Multimodale Robotikdaten (Vision, Sprache, Aktion)
Verarbeitung natürlicher Sprache	Nein	Nein	Ja
Ziel des Lernens	Optimale Strategie (Policy) erlernen	Expertenverhalten nachahmen	Roboteraktionen aus Bild- und Spracheingaben erzeugen
Typische Anwendung	Entscheidungs- und Steuerungsaufgaben	Nachahmung menschlicher Demonstrationen	Sprachgesteuerte Robotikaufgaben

Der Vergleich zeigt zunächst die grundlegenden Unterschiede der klassischen Lernverfahren. Während RL durch Interaktion mit der Umgebung und ein Belohnungssignal lernt, basiert IL auf der Nachahmung menschlicher Demonstrationen. Dadurch entfällt die Definition einer expliziten Belohnungsfunktion und das Verhalten kann direkt aus Demonstrationen gelernt werden [vgl. 26, S. 1-6], [vgl. 28, S. 3-5].

VLA-Modelle unterscheiden sich durch die Erweiterung des Ansatzes um die gemeinsame Verarbeitung visueller und sprachlicher Informationen zur Aktionsgenerierung [vgl. 34, S. 1-2].

3 METHODIK DES IMITATION LEARNING

In Kapitel 2.2.2 Imitation Learning wurde IL bereits als mögliche Methode zur Programmierung humanoider Roboter vorgestellt. Dabei wurden insbesondere verschiedene Ansätze des IL näher erläutert. In den nachfolgenden Kapiteln wird die Methodik des IL näher beschrieben. Dabei werden die einzelnen Schritte von der Demonstrationsdatenerfassung über die Aufbereitung des Trainingsdatensatzes bis hin zum Modelltraining und dem anschließenden Deployment des Modells am Roboter erläutert.

3.1 DEMONSTRATIONSDATENERFASSUNG

Die Demonstrationsdatenerfassung stellt den ersten Schritt des IL-Workflows dar. Bevor ein Modell trainiert werden kann, muss zunächst ein Demonstrationsdatensatz erstellt werden, der das gewünschte Verhalten für die zu erlernende Aufgabe beschreibt. Hierzu werden Demonstrationen einer Expertin bzw. eines Experten aufgezeichnet und in einem Datensatz zusammengeführt. Die Qualität der Demonstrationen sowie die Wahl einer geeigneten Demonstrationstechnik bilden die Grundlage des späteren Lernprozesses und beeinflussen maßgeblich die Qualität der erlernten Policy. Aus diesem Grund stellt die Demonstrationsdatenerfassung einen zentralen Bestandteil des gesamten Imitation-Learning-Prozesses dar [vgl. 36, S. 1-2].

Eine Demonstration beschreibt die Ausführung einer Aufgabe durch eine Expertin bzw. einen Experten. Dabei kann es sich sowohl um einen Menschen als auch um einen Roboter handeln. Ziel einer Demonstration ist es, den vollständigen Lösungsweg einer Aufgabe aufzuzeichnen und nicht lediglich das erzielte Endergebnis festzuhalten. Dadurch wird erfasst, welche Aktionen die Expertin bzw. der Experte in den jeweiligen Situationen ausführt, sodass der zugrunde liegende Handlungsansatz vom Lernalgorithmus nachvollzogen und auf ähnliche Situationen übertragen werden kann. In robotischen Anwendungen wird eine Demonstration häufig als Trajektorie beschrieben, die die zeitlich geordnete Folge der während der Aufgabenbearbeitung aufgezeichneten Zustände und Aktionen umfasst. Mehrere Demonstrationen werden anschließend zu einem Demonstrationsdatensatz zusammengefasst, der als Grundlage für das spätere Modelltraining dient [vgl. 36, S. 1-3].

Der Inhalt einer Demonstration hängt von der jeweiligen Anwendung sowie der eingesetzten Robotikplattform ab. Im Allgemeinen besteht eine Demonstration aus einer Folge von Zuständen und den zugehörigen Aktionen der Expertin bzw. des Experten. In robotischen Anwendungen werden darüber hinaus häufig weitere Informationen aufgezeichnet, um den Bewegungsablauf möglichst vollständig zu beschreiben. Hierzu zählen beispielsweise Positionen und Orientierungen des Endeffektors, dessen Trajektorie, Gelenkwinkel, Geschwindigkeiten, Beschleunigungen, Drehmomente oder Motorströme der einzelnen Gelenke. Je nach Aufgabe können zusätzlich Kraft- und Steifigkeitsinformationen sowie weitere Sensordaten Bestandteil einer Demonstration sein. Die Gesamtheit dieser Informationen bildet die Grundlage für den späteren Demonstrationsdatensatz [vgl. 36, S. 1-2], [vgl. 37, S. 1-3].

Zur Erfassung dieser Demonstrationen stehen verschiedene Demonstrationsverfahren zur Verfügung. Dabei wird zunächst zwischen direkten und indirekten Demonstrationsmethoden unterschieden. Bei direkten Verfahren führt der Agent die Demonstration selbst aus, sodass die aufgezeichneten Daten unmittelbar dem späteren Zustands- und Aktionsraum des Roboters entsprechen. Zu den direkten Demonstrationstechniken zählen unter anderem die Teleoperation, die kinästhetische Demonstration sowie Shadowing. Indirekte Verfahren hingegen erfassen zunächst das Verhalten eines externen Demonstrators, beispielsweise eines Menschen oder eines weiteren Roboters, der nicht als Agent agiert. Die Demonstrationen werden dabei beispielsweise durch Beobachtung oder mithilfe von Sensoren am Demonstrator aufgezeichnet und anschließend in eine für den Lernagenten nutzbare Darstellung überführt [vgl. 36, S. 2-6].

Zu den am häufigsten eingesetzten Demonstrationsverfahren zählen die kinästhetische Demonstration, die Beobachtungsdemonstration (engl. Observation) sowie die Teleoperation. Die Wahl des geeigneten Verfahrens hängt insbesondere von der jeweiligen Anwendung, der verfügbaren Sensorik sowie den Anforderungen an Genauigkeit, Bedienbarkeit und Datenerfassung ab [vgl. 36, S. 2-6], [vgl. 37, S. 3-4].

Kinästhetische Demonstration

Bei der kinästhetischen Demonstration führt der Demonstrator den Roboter direkt an seinen Gelenken oder am Endeffektor durch die gewünschte Bewegung. Die dabei entstehenden Roboterzustände werden unmittelbar über die internen Sensoren aufgezeichnet. Da die Demonstration direkt am Zielroboter erfolgt, sind keine zusätzlichen Koordinatentransformationen erforderlich. Einschränkungen ergeben sich jedoch bei großen oder schweren Robotern sowie bei Anwendungen, in denen eine direkte Interaktion mit dem Roboter nicht möglich oder nicht sicher ist. Auch die Qualität der Demonstrationen ist stark von den Fähigkeiten der Expertin bzw. des Experten abhängig. Oftmals müssen Demonstrationen trotz des Einsatzes von Expertinnen bzw. Experten noch überarbeitet und verfeinert werden [vgl. 36, S. 5], [vgl. 28, S. 3-4].

Beobachtungsdemonstration

Bei der Beobachtungsdemonstration wird die Aufgabe zunächst von einem externen Demonstrator ausgeführt. Dessen Bewegungen werden mithilfe externer Sensoren, beispielsweise Kameras, Motion-Capture-Systemen oder am Körper getragenen Sensoren, erfasst und anschließend auf den Roboter übertragen. Dieses Verfahren erlaubt eine natürliche Demonstration, ohne dass der Demonstrator den Roboter direkt bedienen muss. Allerdings können beispielsweise auch kamerabedingte Probleme zu Genauigkeitsverlusten oder Abbildungsfehlern führen, wodurch eine Nachbearbeitung erforderlich ist [vgl. 36, S. 6], [vgl. 37, S. 3-4].

Teleoperation

Bei der Teleoperation steuert der Demonstrator den Roboter über ein geeignetes Eingabegerät, beispielsweise einen Joystick oder ein Virtual-Reality-System (VR-System). Dadurch werden die Demonstrationen unmittelbar auf dem Roboter erzeugt und gleichzeitig aufgezeichnet. Im Vergleich zur Beobachtungsdemonstration entfallen zusätzliche Übertragungsschritte zwischen Demonstrator und Roboter, wodurch präzisere Demonstrationen erzeugt werden können. Ein weiterer Vorteil besteht in der höheren Flexibilität, die durch die räumliche Trennung des Arbeitsplatzes von Mensch und Roboter ermöglicht wird. Demgegenüber stehen höhere Anforderungen an die

eingesetzten Eingabegeräte sowie an die Erfahrung des Demonstrators im Umgang mit der Teleoperationsschnittstelle [vgl. 36, S. 5], [vgl. 37, S. 3-4].

3.2 AUFBEREITUNG DES TRAININGSDATENSATZES

Nach der Erfassung der Demonstrationen liegen die Daten zunächst als einzelne Demonstrationen beziehungsweise Trajektorien vor. Für das anschließende Modelltraining müssen diese zu einem Demonstrationsdatensatz zusammengeführt werden. Damit ein Lernverfahren die aufgezeichneten Demonstrationen verarbeiten kann, müssen die enthaltenen Informationen in einer geeigneten Datenstruktur vorliegen. Die Wahl der Datenrepräsentation beeinflusst dabei sowohl die verwendbaren Lernalgorithmen als auch die Qualität des späteren Lernprozesses [vgl. 36, S. 6-7].

Ein Demonstrationsdatensatz besteht aus einer Sammlung mehrerer Demonstrationen, die jeweils eine vollständige Ausführung einer Aufgabe beschreiben. Eine einzelne Demonstration wird dabei üblicherweise als Trajektorie dargestellt und beschreibt eine zeitlich geordnete Folge von Daten während der Ausführung einer Aufgabe. Da reale Umgebungen häufig eine hohe Dimensionalität aufweisen und nicht alle verfügbaren Informationen für die Aufgabenlösung relevant sind, müssen geeignete Informationen ausgewählt werden. Das Ziel besteht darin, ausreichend aufgabenrelevante Informationen bereitzustellen, ohne redundante oder irrelevante Daten in das Lernverfahren einzubringen. Diese Daten können beispielsweise den Zustand des Roboters, Sensordaten oder weitere aufgabenspezifische Informationen umfassen. Zusätzlich werden häufig Kontextinformationen gespeichert, die die Randbedingungen der jeweiligen Demonstration beschreiben. Für bestimmte Lernverfahren können darüber hinaus auch Belohnungssignale Bestandteil des Datensatzes sein [vgl. 26, S. 18-19], [vgl. 36, S. 6], [vgl. 37, S. 4-5].

Für die Repräsentation der Demonstrationen können unterschiedliche Arten von Merkmalen verwendet werden. Rohmerkmale (engl. Raw Features) bestehen aus direkt aufgezeichneten Sensordaten und können ohne weitere Verarbeitung verwendet werden, sofern sie bereits alle für die Aufgabe notwendigen Informationen enthalten. Häufig werden die Sensordaten jedoch vorverarbeitet oder in eine kompaktere Darstellung überführt, um irrelevante oder redundante Informationen zu entfernen. Dies kann

entweder durch manuell definierte Merkmale erfolgen, die auf Expertenwissen basieren, oder durch automatisch gelernte Merkmalsextraktion, beispielsweise mittels neuronaler Netze. Die automatische Merkmalsextraktion reduziert den Bedarf an aufgabenspezifischem Expertenwissen und ermöglicht eine allgemeinere Verarbeitung unterschiedlicher Demonstrationen [vgl. 36, S. 6-7].

Vor dem eigentlichen Modelltraining können Demonstrationsdatensätze zusätzlich optimiert werden. Ziel ist dabei die Verbesserung der Qualität der Trainingsdaten um dadurch den Lernprozess zu unterstützen. Ursachen für eine notwendige Optimierung können eine zu geringe oder zu große Anzahl an Demonstrationen sowie fehlerhafte Demonstrationen sein. Zu den beschriebenen Maßnahmen zählen unter anderem die Erweiterung des Datensatzes durch Data Augmentation, die Reduktion redundanter Demonstrationen sowie Verfahren zur Korrektur fehlerhafter Demonstrationen. Darüber hinaus können komplexe Demonstrationen in mehrere Teilaufgaben segmentiert werden. Dadurch entstehen Subtrajektorien mit ähnlichen geometrischen oder funktionalen Eigenschaften, die anschließend separat gelernt werden können und häufig eine robustere Modellierung ermöglichen [vgl. 37, S. 10].

3.3 MODELLTRAINING

Nach der Aufbereitung des Demonstrationsdatensatzes erfolgt das Modelltraining. In dieser Phase werden die Demonstrationen verwendet, um ein Lernmodell zu trainieren, das die gewünschte Aufgabe aus den gezeigten Expertenausführungen erlernt. Abhängig vom gewählten Lernverfahren wird anschließend beispielsweise eine Policy, eine Belohnungsfunktion oder ein Dynamikmodell erlernt. Der allgemeine Ablauf des Lernens aus Demonstrationen (engl. Learning from Demonstration, LfD) umfasst somit die Erstellung eines Demonstrationsdatensatzes, die Auswahl eines geeigneten Lernverfahrens sowie das anschließende Training des entsprechenden Modells [vgl. 36, S. 2-3].

Für das Modelltraining können unterschiedliche Verfahren des IL eingesetzt werden. Zu den am häufigsten verwendeten Ansätzen zählen BC sowie IRL. Während BC die Abbildung zwischen Zuständen und Aktionen direkt aus den Demonstrationen erlernt, rekonstruiert IRL zunächst eine Belohnungsfunktion, auf deren Grundlage anschließend

eine Policy bestimmt wird [vgl. 37, S. 15-16]. Eine detaillierte Beschreibung der beiden Verfahren erfolgt in Kapitel 2.2.2.1 für BC beziehungsweise in Kapitel 2.2.2.2 für IRL.

Das eigentliche Lernverfahren wird durch ein geeignetes Lernmodell umgesetzt, das die in den Demonstrationen enthaltenen Zusammenhänge nutzt, um eine Abbildung zwischen Beobachtungen beziehungsweise Zuständen und den zugehörigen Aktionen zu erlernen. Hierfür kommen je nach Anwendungsfall unterschiedliche Modellklassen zum Einsatz. In der Literatur werden unter anderem Hidden Markov Models (HMM), Dynamic Movement Primitives (DMP), Gaussian Mixture Models (GMM/GMR) sowie Artificial Neural Networks (ANN) als Lernmodelle für LfD beschrieben [vgl. 36, S. 1], [vgl. 37, S. 2].

Eine aktuelle Policy-Architektur für das Imitation Learning stellt Action Chunking with Transformers (ACT) dar. ACT wurde entwickelt, um die bei feinmotorischen Manipulationsaufgaben auftretenden Herausforderungen des Imitation Learnings, insbesondere die Akkumulation von Vorhersagefehlern (engl. compounding errors), zu reduzieren. Hierzu sagt das Modell nicht nur die unmittelbar nächste Aktion voraus, sondern erzeugt für jede Beobachtung eine Sequenz zukünftiger Aktionen (Action Chunks). Dadurch wird der effektive Planungshorizont verkürzt und die Robustheit gegenüber kleinen Vorhersagefehlern verbessert. Zusätzlich nutzt ACT einen Transformer zur Verarbeitung visueller Beobachtungen und Gelenkzustände sowie eine Conditional Variational Autoencoder (CVAE)-Struktur, um die Variabilität menschlicher Demonstrationen abzubilden [vgl. 38].

Nach Abschluss des Trainings wird die trainierte Policy für die Inferenz verwendet und erzeugt auf Grundlage der aktuellen Beobachtungen die vom Robotersystem auszuführenden Aktionen [vgl. 38]. Der Übergang vom trainierten Modell zum Einsatz auf dem Robotersystem wird im folgenden Kapitel im Rahmen des Deployments näher beschrieben.

3.4 DEPLOYMENT AM ROBOTERSYSTEM

Nach Abschluss des Trainings wird die gelernte Policy auf das Zielrobotersystem übertragen und dort zur Lösung der vorgesehenen Aufgabe eingesetzt. Im Rahmen des Deployments übernimmt die trainierte Policy die Steuerung des Roboters während der

Aufgabenbearbeitung. Hierzu verarbeitet sie die vom Robotersystem bereitgestellten Beobachtungen und erzeugt daraus die auszuführenden Steuerbefehle. Diese werden auf der physischen Hardware ausgeführt und bestimmen das Verhalten des Roboters. Im Gegensatz zu den vorangegangenen Trainingsschritten findet die Ausführung der Policy nun unter den Bedingungen der realen Einsatzumgebung statt. Dabei kann sich zeigen, ob die im Training erlernte Policy auch außerhalb der Trainingsumgebung zuverlässig arbeitet. Unterschiede zwischen Trainings- und Einsatzbedingungen können sich unmittelbar auf das Verhalten der Policy auswirken und die Leistungsfähigkeit des Systems beeinflussen [vgl. 39, S. 1730-1731], [vgl. 38], [vgl. 40, S. 8-9]. In den nachfolgenden Abschnitten werden einige Herausforderungen beim Deployment von IL-Policies näher beschrieben.

Covariate Shift

Eine zentrale Herausforderung besteht darin, dass sich die Bedingungen während der Ausführung von den Bedingungen unterscheiden können, unter denen die Demonstrationen aufgezeichnet oder die Policy trainiert wurde. Bereits geringe Änderungen der Umgebungsbedingungen, beispielsweise abweichende Beleuchtungsverhältnisse, veränderte Kameraperspektiven, Unterschiede in den physikalischen Eigenschaften von Objekten oder Sensorrauschen, können dazu führen, dass die aktuellen Beobachtungen nicht mehr der während des Trainings erlernten Datenverteilung entsprechen. Dadurch kann die Policy mit Situationen konfrontiert werden, die im Demonstrationsdatensatz nicht oder nur unzureichend enthalten waren. Dieses Phänomen wird als Covariate Shift bezeichnet und zählt zu den grundlegenden Herausforderungen des Imitation Learning, da sich Vorhersagefehler während der Ausführung akkumulieren und den Roboter zunehmend in Zustände führen können, für die keine geeigneten Demonstrationen vorliegen. Dadurch kann sich die Leistungsfähigkeit der Policy schrittweise verschlechtern und letztlich zum Fehlschlagen der Aufgabe führen [vgl. 41, S. 5-7], [vgl. 42]. Eine genauere Beschreibung des Covariate Shift sowie mögliche Lösungsansätze wurden bereits in Kapitel 2.2.2.1 Behavioral Cloning erläutert.

Reality Gap

Werden Demonstrationen oder das Training vollständig oder teilweise in einer Simulationsumgebung durchgeführt, entsteht zusätzlich die sogenannte Reality Gap. Simulationen können das Verhalten realer Robotersysteme und ihrer Umgebung nur näherungsweise abbilden. Unterschiede in Kontaktkräften, Reibungskoeffizienten, Materialeigenschaften, Sensormodellen oder der visuellen Darstellung führen dazu, dass eine in der Simulation gut funktionierende Policy ihre Leistungsfähigkeit nicht zwangsläufig auf das reale Robotersystem übertragen kann. Ziel aktueller Sim-to-Real-Verfahren ist daher, diese Diskrepanz möglichst zu reduzieren und eine robuste Generalisierung auf reale Einsatzbedingungen zu ermöglichen [vgl. 40, S. 1], [vgl. 41, S. 5], [vgl. 43, S. 403-419].

Fehlende Vorhersagbarkeit und Black-Box-Charakter

Ein weiteres Problem moderner Imitation-Learning-Ansätze besteht in der eingeschränkten Interpretierbarkeit der verwendeten neuronalen Netzwerke. Die Entscheidungsfindung erfolgt auf Grundlage einer großen Anzahl gelernter Parameter, deren Einfluss auf die erzeugte Aktion für den Menschen in der Regel nicht unmittelbar nachvollziehbar ist. Dadurch lässt sich häufig weder zuverlässig vorhersagen, wie sich die Policy in bislang unbekannten Situationen verhalten wird, noch warum eine bestimmte Aktion ausgewählt wurde. Insbesondere bei sicherheitskritischen Anwendungen erschwert diese eingeschränkte Nachvollziehbarkeit die Analyse von Fehlverhalten sowie die Validierung des Systems vor dem produktiven Einsatz [vgl. 39, S. 1727-1728], [vgl. 41, S. 11].

Zusammenfassend stellt das Deployment den Übergang von der Trainingsphase zur praktischen Anwendung einer Policy auf dem realen Robotersystem dar. Während dieses Schrittes zeigen sich die Grenzen der im Training erlernten Policy besonders deutlich, da Unterschiede zwischen Trainings- und Einsatzbedingungen sowie die eingeschränkte Nachvollziehbarkeit neuronaler Netzwerke die Zuverlässigkeit des Systems beeinflussen können. Die Berücksichtigung dieser Herausforderungen ist daher eine wesentliche Voraussetzung für den erfolgreichen Einsatz von Imitation-Learning-Verfahren in realen Robotikanwendungen.

Insgesamt folgt die Methodik des Imitation Learning einem aufeinander aufbauenden Workflow. Dieser beginnt mit der Demonstration einer Aufgabe und der Erfassung der dabei entstehenden Demonstrationsdaten, umfasst deren Aufbereitung sowie das Erlernen einer Policy und endet mit deren Deployment auf dem Zielrobotersystem. Während die ersten Schritte die Grundlage für das Erlernen des gewünschten Verhaltens schaffen, zeigt sich erst während des Deployments, ob die Policy die erlernten Fähigkeiten auch unter realen Einsatzbedingungen zuverlässig anwenden kann.

4 IMPLEMENTIERUNG UND EVALUIERUNG

4.1 ENTWICKLUNGSUMGEBUNG

4.1.1 HARDWARE

Für die praktische Umsetzung des Imitation-Learning-Workflows wurde eine Entwicklungsumgebung eingerichtet, die sowohl das Training der verwendeten Policy als auch die Demonstrationsdatenerfassung, die Simulation und die Ausführung auf dem realen Robotersystem ermöglicht. Da das Training neuronaler Netze eine hohe Rechenleistung erfordert, wurde ein leistungsfähiges Entwicklungssystem eingesetzt.

Die Entwicklung erfolgte auf einem Desktop-PC mit dem Betriebssystem Ubuntu 22.04.5 LTS. Der Desktop-PC verfügt über einen Intel® Core™ Ultra 9 285K Prozessor, eine NVIDIA RTX 5000 Ada Generation Grafikkarte sowie 128 GB Arbeitsspeicher. Als Massenspeicher wurde eine 2 TB SSD verwendet. Eine genaue Übersicht der verwendeten Hardwarekomponenten ist in Tabelle 3 dargestellt.

Für die praktische Aufzeichnung der Demonstrationen und die anschließende autonome Ausführung kam der humanoide Roboter Unitree G1 in der Version EDU mit 29 DoF zum Einsatz. Für die Manipulationsaufgaben wurden am Roboter Fünf-Finger-Hände montiert, um das Greifen der Objekte zu ermöglichen. Als Endeffektor wurde dabei die RH56DFTP Dexterous Hand von Inspire Robots verwendet.

Tabelle 3: Verwendete Hardware für Demonstrationsdatenaufzeichnung und Training

Komponente	Spezifikation
Betriebssystem	Ubuntu 22.04.5 LTS
Prozessor	Intel® Core™ Ultra 9 285K (24 Kerne)
Grafikkarte	NVIDIA RTX 5000 Ada Generation
Arbeitsspeicher	128 GB
Massenspeicher	2 TB SSD

Komponente	Spezifikation
Robotersystem	Unitree G1 EDU
Endeffektor	RH56DFTP Dexterous Hand

4.1.2 SOFTWARE

Die Softwareumgebung setzt sich aus mehreren Komponenten zusammen, die gemeinsam den vollständigen Imitation-Learning-Workflow unterstützen. Die einzelnen Werkzeuge übernehmen dabei unterschiedliche Aufgaben innerhalb der Demonstrationsdatenerfassung, des Modelltrainings sowie des Deployments auf dem Robotersystem.

Die Grundlage bildet das LeRobot-Framework (Version 0.4.1), welches für die Erstellung des Demonstrationsdatensatzes, das Training der ACT-Policy sowie deren Deployment eingesetzt wurde. Für die Teleoperation des Unitree G1 während der Demonstrationsdatenerfassung wurde das von Unitree bereitgestellte Repository *xr_teleoperate* verwendet.

Zur Simulation und Validierung der Roboterumgebung kam NVIDIA Isaac Sim (Version 5.0.0) in Verbindung mit Isaac Lab (Version 0.44.9) zum Einsatz. Für die Integration der Inspire-Hand mussten zusätzlich der DFX Inspire Hand Service sowie das Inspire Hand SDK eingebunden werden. Darüber hinaus waren kleinere Anpassungen auf Grundlage eines veröffentlichten GitHub-Issue erforderlich, um die FTP Inspire-Hand vollständig in die bestehende Entwicklungsumgebung zu integrieren.

Da Isaac Sim und das LeRobot-Framework unterschiedliche Softwareabhängigkeiten sowie Versionskonflikte aufweisen, wurden getrennte Python-Umgebungen erstellt. Das Training der ACT-Policy erfolgte mit Python 3.10.19 und PyTorch 2.7.1, während für Isaac Sim und Isaac Lab eine separate Umgebung mit Python 3.11.14, PyTorch 2.7.0 und CUDA 12.8 eingerichtet wurde. Eine Übersicht der eingesetzten Softwarekomponenten und Versionen ist in Tabelle 4 zusammengefasst.

Tabelle 4: Verwendete Software- und Framework-Versionen sowie Unitree Repository

Komponente	Version / Beschreibung
Ubuntu	22.04.5 LTS
LeRobot	0.4.1
Isaac Sim	5.0.0
Isaac Lab	0.44.9
Python (Training)	3.10.19
Python (Simulation)	3.11.14
PyTorch (Training)	2.7.1
PyTorch (Simulation)	2.7.0 + CUDA 12.8
xr_teleoperate	Unitree Repository
unitree_lerobot	Unitree Repository
DFX Inspire Hand Service	Unitree Repository
Inspire Hand SDK	Unitree Repository

4.1.3 VERSUCHSAUFBAU

Für die Durchführung der Demonstrationsdatenerfassung sowie der späteren Evaluierung wurde ein einheitlicher Versuchsaufbau verwendet, um möglichst konstante Umgebungsbedingungen während sämtlicher Aufzeichnungen sicherzustellen. Der humanoide Roboter Unitree G1 wurde auf einem Stuhl positioniert und zur Gewährleistung einer stabilen Arbeitsposition an einem Kran befestigt. Vor dem Roboter befand sich ein höhenverstellbarer Tisch, dessen Höhe an die Sitzposition des Roboters angepasst wurde.

Als Demonstrationsaufgabe wurde das Sortieren von drei farbigen, 3D-gedruckten Würfeln gewählt, die auf ihrer Oberseite mit den Buchstaben B, R und P beschriftet waren. Die Beschriftung orientiert sich an der Bezeichnung des Projektpartners BRP-

Rotax GmbH & Co KG und dient im Rahmen dieser Arbeit ausschließlich der eindeutigen Identifikation der einzelnen Würfel. Zu Beginn jeder Demonstration befanden sich die Würfel in einer fest definierten Ausgangsanordnung auf dem Tisch. Auf eine Variation der Ausgangsreihenfolge wurde bewusst verzichtet, da sich dadurch die Anzahl der erforderlichen Demonstrationen erheblich erhöht hätte. Um jedoch eine bessere Robustheit gegenüber leichten Abweichungen zu erreichen, wurde die genaue Position sowie die Orientierung der drei Würfel jeweils leicht verändert. Des Weiteren wurden Markierungen auf der Arbeitsfläche angebracht, welche sowohl den Start- als auch den Zielbereich der Würfel kennzeichneten und dadurch eine reproduzierbare Positionierung ermöglichten. Die Würfel lagen zu Beginn der Aufgabe in der Reihenfolge Orange (R), Gelb (B) und Grau (P) vor. Ziel der Demonstrationen war es, die Würfel entsprechend der vorgegebenen Zielanordnung B-R-P umzusortieren.

Zur Veranschaulichung der Zielkonfiguration wurden rechts neben dem Arbeitsbereich drei kleinere Würfel in der gewünschten Zielreihenfolge auf dem Tisch platziert. Diese dienten lediglich als visuelle Referenz für das Sortierergebnis.

Um möglichst konstante Bildbedingungen für die kamerabasierte Wahrnehmung sicherzustellen, wurde der Arbeitsbereich zusätzlich durch eine frontal ausgerichtete Tischleuchte beleuchtet. Ergänzend blieb die allgemeine Beleuchtung des Labors während sämtlicher Demonstrationen unverändert eingeschaltet. Der vollständige Versuchsaufbau ist in Abb. 4 dargestellt.

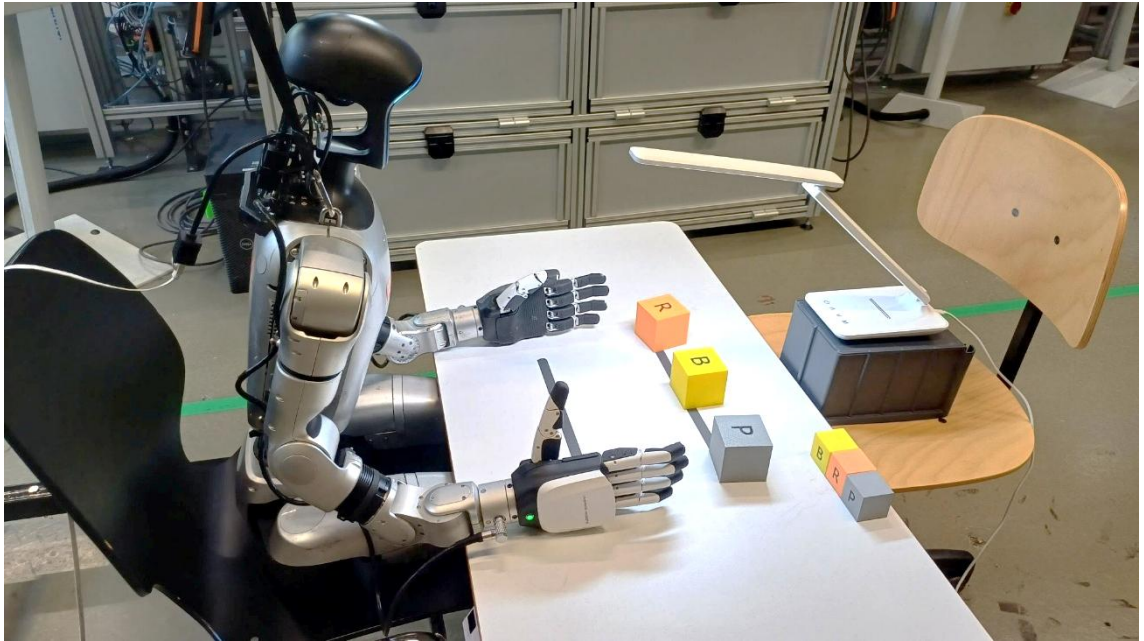


Abb. 4: Versuchsaufbau zur Durchführung der Sortieraufgabe

4.2 DEMONSTRATIONS DATENERFASSUNG

Zur Erstellung des Demonstrationsdatensatzes wurden sämtliche Demonstrationen durch Teleoperation des realen Robotersystems aufgezeichnet. Dabei wurde die Sortieraufgabe von der Demonstratorin vorgeführt. Eine weitere Person startete dabei über eine Terminaleingabe die Aufzeichnung. Somit wurden unter anderem RGB-Bilder mit einer Auflösung von 640 x 480 bei 30 FPS sowie Zustands- und Aktionsdaten für die Arme und Endeffektoren aufgezeichnet. Auf diese Weise entstand ein Demonstrationsdatensatz, der die vollständigen Trajektorien der Aufgabe enthält und anschließend als Grundlage für das Training der Policy verwendet wurde.

Die Teleoperation erfolgte mit einer Pico 4 Ultra VR-Brille in Verbindung mit dem Repository `xr_teleoperate`. Während der Demonstration wurden die Bewegungen der Hände der Demonstratorin mittels Handtracking erfasst und auf die Arme sowie die Inspire-Hand des Roboters übertragen. Zur visuellen Rückmeldung wurde das Kamerabild des Roboters über eine WebRTC-Verbindung auf die VR-Brille gestreamt, sodass die Demonstratorin die Aufgabe aus der Perspektive des Roboters ausführen konnte.

Für die Demonstrationsdatenerfassung wurde die im Roboterkopf verbaute Tiefenkamera verwendet. Zusätzliche Kameras wie zum Beispiel an den Armgelenken wurden nicht verwendet.

Vor Beginn jeder Demonstration wurden der Roboter sowie die drei Würfel in ihre jeweiligen Ausgangspositionen gebracht. Anschließend wurde zunächst das Programm *Headless_driver_double.py* am Roboter-PC (PC2) für die Kommunikation zu den Inspire-Händen gestartet. Damit wurde sichergestellt, dass auch die Finger des Roboters gesteuert werden können. Der Roboter selbst ist dabei mit dem internen WLAN sowie über ein Lan-Kabel mit dem Desktop-PC verbunden. Der Desktop-PC befindet sich ebenfalls im internen WLAN.

Als nächstes wurde das Programm *teleop_hand_and_arm.py* zur Teleoperation am Desktop-PC gestartet. Um das Programm starten zu können, mussten beim Programmaufruf weitere Parameter definiert werden. Als Endeffektor wurde die FTP Inspire-Hand angegeben. Des Weiteren wurde die IP-Adresse des PC2 im internen WLAN für die visuelle Kommunikation und die Konfiguration der WebRTC-Verbindung definiert. Das angegebene Netzwerk Interface bezieht sich wiederum auf den Desktop-PC. Die Ergänzung *--record* aktiviert die Möglichkeit zur Aufzeichnung der Demonstrationen. In Terminalaufruf 1 ist der verwendete Programmaufruf einschließlich der Parameterdefinition abgebildet. Dieser Aufruf wurde in der virtuellen Umgebung des zugehörigen Repository *xr_teleoperate* ausgeführt.

```
python teleop_hand_and_arm.py --ee=inspire_ftp --img-server-ip=<PC2_IP> --  
network-interface=<NETWORK_INTERFACE> --record
```

Terminalaufruf 1: Start des Teleoperationsprogramms

Im nächsten Schritt musste die Verbindung zur VR-Brille hergestellt werden. Als Browser wurde dabei Google Chrome verwendet, da dieser unter anderem als WebXR-unterstützender Browser empfohlen wird. Zunächst wurde auf der VR-Brille ebenfalls eine Verbindung zum internen WLAN hergestellt. Anschließend wurde im Browser die Seite https://<DESKTOP-PC_IP>:8012/?ws=wss://<DESKTOP-PC_IP>:8012

aufgerufen. Dabei steht `<DESKTOP-PC_IP>` als Platzhalter für die IP-Adresse des Desktop-PCs.

Anschließend wurde das Handtracking sowie die Datenaufzeichnung über eine Eingabe im Terminal des laufenden Teleoperationsprogramms gestartet und die Würfel entsprechend der vorgegebenen Aufgabe von der Demonstratorin sortiert. Nach erfolgreichem Abschluss der Sortieraufgabe wurde die Aufzeichnung beendet und die Demonstration lokal gespeichert. Nach etwa 20 aufgezeichneten Episoden trat ein mechanischer Defekt an der linken Hand des Roboters auf, wodurch der Daumen nicht mehr vollständig angesteuert werden konnte. Die weiteren Demonstrationen wurden daher unter dieser Einschränkung aufgezeichnet.

Für den gesamten Datensatz wurden bei dieser Aufgabe knapp 60 Episoden aufgezeichnet. Dabei wurde darauf geachtet, nur Aufzeichnungen für das spätere Modelltraining zu verwenden, in denen die Sortieraufgabe erfolgreich abgeschlossen wurde. Zusätzlich wurden auch Durchgänge aufgezeichnet, bei denen die Würfel leicht variierende Positionen und Rotationen aufwiesen. Auch das Korrekturverhalten bei möglichen Fehlgriffen wurde in einigen der Aufzeichnungen dargestellt, um dem späteren Modell das Erlernen geeigneter Korrekturstrategien zu ermöglichen.

Die Dauer einer einzelnen Demonstration betrug jeweils weniger als fünf Minuten. Der gesamte Datensatz konnte daher mit etwas Übung im Umgang mit der Teleoperation des Roboters innerhalb eines Tages aufgezeichnet werden. Ungeplante Störungen wie beispielsweise Kommunikationsprobleme mit der VR-Brille oder dem Roboter verlängerten jedoch häufig die für die Aufzeichnungen benötigte Zeit.

4.3 AUFBEREITUNG DES TRAININGSDATENSATZES

Nach der Aufzeichnung der Demonstrationen sowie der Aussortierung aller nicht erfolgreich abgeschlossenen Durchgänge wurden die einzelnen Episoden zunächst sortiert und einheitlich umbenannt. Hierfür kam das Skript `sort_and_rename_folders.py` zum Einsatz. Dieses stellte sicher, dass die Nummerierung der Demonstrationen mit `episode_0` beginnt und alle weiteren Episoden fortlaufend in einem einheitlichen Format benannt werden.

Anschließend wurde der Datensatz, der zunächst im Unitree-JSON-Format vorlag, in das für das Modelltraining benötigte LeRobot-Format konvertiert. Hierzu wurde das Skript `convert_unitree_json_to_lerobot.py` ausgeführt. Zusätzlich zur Ausführung des Python-Skripts mussten mehrere Parameter definiert werden. Dazu gehörten unter anderem der Dateipfad zum Datensatz im JSON-Format, die Repository-ID sowie der verwendete Robotertyp. Da das Modelltraining lokal ausgeführt werden sollte, wurde auf den Befehl `--push_to_hub` verzichtet. Dadurch wurde der konvertierte Datensatz nicht auf den Hugging Face Hub hochgeladen, sondern lediglich lokal im standardmäßigen Hugging Face Cache gespeichert. Der verwendete Programmaufruf einschließlich der Parameterdefinition ist in Terminalaufruf 2 abgebildet.

```
python      unitree_lerobot/utils/convert_unitree_json_to_lerobot.py      -raw-  
dir=<JSON_DATASET_PFAD>              --repo-id=<REPOSITORY-ID>              --  
robot_type=Unitree_G1_Inspire_HeadOnly
```

Terminalaufruf 2: Konvertierung des Demonstrationsdatensatzes

Die Konvertierung des Datensatzes kann je nach Größe des Demonstrationsdatensatzes mehrere Minuten in Anspruch nehmen.

Nach Abschluss der Konvertierung wurde der Datensatz mithilfe des in Terminalaufruf 3 dargestellten Befehls aus dem lokalen Hugging Face Cache an einen beliebigen Speicherort auf dem Desktop-PC – hier das Home-Verzeichnis – kopiert.

```
cd ~/.cache/huggingface/lerobot  
  
cp -r <REPOSITORY-ID> ~/
```

Terminalaufruf 3: Kopieren des Datensatzes ins Home-Verzeichnis

4.4 MODELLTRAINING

Nach der Aufbereitung der Demonstrationsdaten erfolgte das Modelltraining auf Basis des konvertierten Datensatzes. Ziel war es, am Ende des Trainings eine Policy zu erhalten, die anschließend am realen Robotersystem getestet und evaluiert werden kann.

Als Policy-Architektur für das Imitation Learning wurde ACT gewählt, da diese besonders für Manipulationsaufgaben gut geeignet ist. Des Weiteren wird ACT von LeRobot direkt unterstützt und bietet außerdem eine kurze Trainingszeit, geringe Rechenanforderungen und eine hohe Leistungsfähigkeit.

Das Modelltraining wurde mithilfe des Skripts *lerobot_train.py* in der Umgebung des LeRobot-Repositorys lokal auf dem Desktop-PC durchgeführt. In Terminalaufruf 4 ist der verwendete Programmaufruf einschließlich der Parameterdefinition dargestellt. Dabei muss unter anderem der Pfad zum Datensatz im LeRobot-Format sowie der Policy-Typ angegeben werden. Der Befehl *--policy.push_to_hub* wird aufgrund des lokalen Trainings deaktiviert. Nach dem Training wird das fertige Modell automatisch in dem durch *--output_dir* angegebenen Verzeichnis gespeichert.

```
python src/lerobot/scripts/lerobot_train.py --dataset.repo_id=<REPOSITORY-ID>
--dataset.root=<LEROBOT_DATASET_PFAD>          --policy.type=act          --
policy.push_to_hub=false  --output_dir=<POLICY_PFAD>  --job_name=XXXX  --
wandb.enable=false
```

Terminalaufruf 4: Start des Modelltrainings der ACT-Policy

Das Training der ACT-Policy erfolgte über insgesamt 100.000 Trainingsschritte mit einer Batchgröße von 8. Dabei wurden regelmäßig Checkpoints in einem Intervall von 20.000 Schritten gespeichert, sodass insgesamt mehrere Trainingsstände der Policy zur Verfügung standen.

Die Trainingsdauer des Modells lag bei etwa zweieinhalb Stunden. Je nach Anzahl der Demonstrationen und insbesondere abhängig von der verwendeten Hardware kann die Trainingsdauer jedoch variieren. Anzumerken ist allerdings, dass ein Datensatz mit 30 Demonstrationen annähernd gleich lange für das Modelltraining benötigte wie ein Datensatz mit 200 Demonstrationen.

4.5 DEPLOYMENT AM ROBOTERSYSTEM

Mit Abschluss des Modelltrainings stand die trainierte ACT-Policy für den anschließenden Einsatz auf dem realen Robotersystem zur Verfügung. Um das Modell am Roboter testen und anschließend evaluieren zu können, musste der Roboter zunächst

in den richtigen Betriebsmodus gebracht werden. Eine genauere Übersicht der verschiedenen Modi des Unitree G1 ist in Abb. 5 dargestellt.

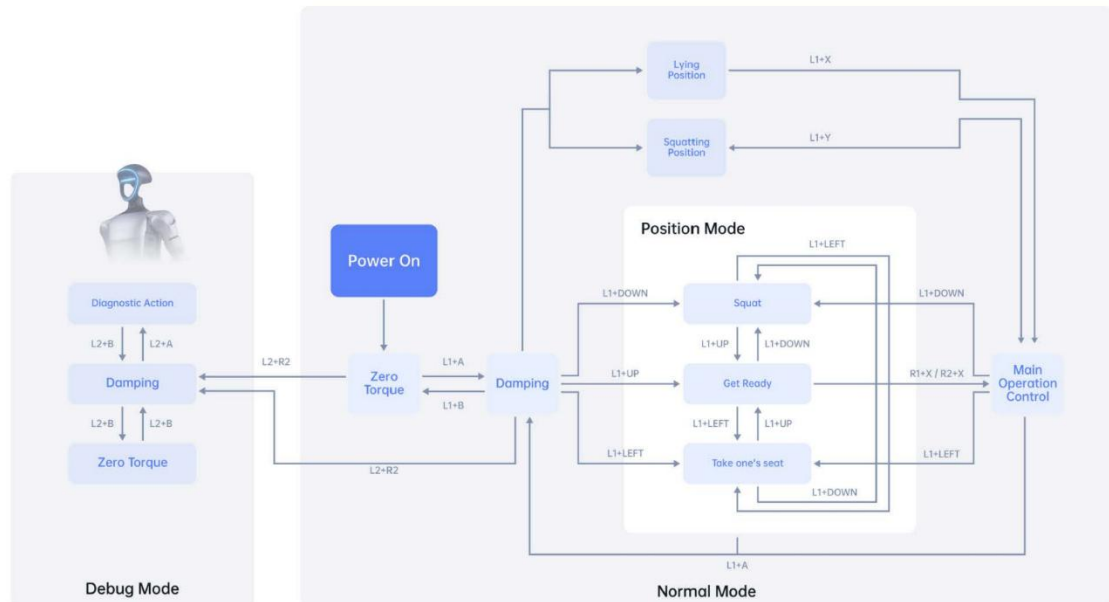


Abb. 5: Betriebsmodi des Unitree G1 [44]

Für die Ausführung der ACT-Policy wurde der Unitree G1 zunächst in den Debug Mode – auch Developer Mode genannt – gebracht. Dazu mussten die Tasten L2 und R2 am Unitree G1 Remote Controller für etwa fünf Sekunden gedrückt gehalten werden. Der Wechsel des Betriebsmodus wird visuell sowie auditiv vom Roboter bestätigt.

Nachdem der Roboter in den richtigen Modus gewechselt hatte, wurden die Programme für die Ausführung der Policy gestartet. Dabei wird erneut das Programm *Headless_driver_double.py* für die Kommunikation mit den FTP Inspire-Händen benötigt. Sobald dieses auf dem PC2 läuft, kann das Programm *eval_gl.py* am Desktop-PC gestartet werden. Der genaue Programmaufruf ist in Terminalaufruf 5 dargestellt. Dabei müssen unter anderem der Dateipfad zur trainierten Policy sowie der Dateipfad zum konvertierten Datensatz angegeben werden. Hierbei wurde der zuletzt erstellte Checkpoint für die Ausführung der Policy verwendet. Die Bewegungsfrequenz wird auf 30 Hz begrenzt. Zusätzlich müssen noch das Modell des verwendeten Roboterarms sowie der Typ des Endeffektors angegeben werden.

```
python          unitree_lerobot/eval_robot/eval_g1.py      --
policy.path=<POLICY_PFAD>/checkpoints/100000/pretrained_model  --
repo_id=<REPOSITORY-ID>  --root="XXXXX"  --episodes=0  --frequency=30  --
arm="G1_29"  --ee="inspire1"  --visualization=true
```

Terminalaufruf 5: Deployment der trainierten ACT-Policy am Unitree G1

Nachdem die Ausführung des Programms im Terminalfenster bestätigt wurde, begann der Roboter mit der autonomen Ausführung der trainierten Policy. Dabei verarbeitet das Modell fortlaufend die vom Robotersystem bereitgestellten Beobachtungen sowie Gelenkdaten und erzeugt daraus die für die jeweilige Situation erforderlichen Roboteraktionen. Die erzeugten Aktionen werden anschließend an die entsprechenden Gelenke und Endeffektoren des Roboters übertragen und unmittelbar ausgeführt. In Abb. 6 ist das Ergebnis der vom Unitree G1 autonom ausgeführten Sortieraufgabe abgebildet.

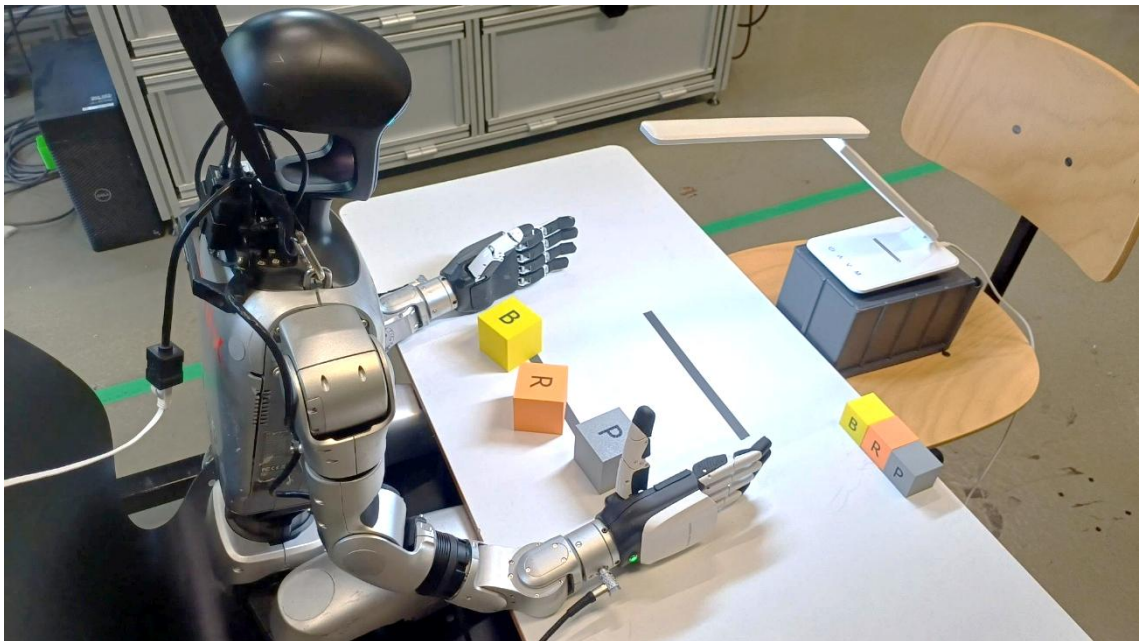


Abb. 6: Ergebnis der autonomen Sortierung durch die trainierte Policy

Die autonome Ausführung durch das Deployment am Roboter stellt somit den abschließenden Schritt des implementierten Imitation-Learning-Workflows dar. Damit kann überprüft werden, ob die aus den Demonstrationen erlernte Policy die Aufgabe auch außerhalb der Trainingsdaten selbstständig ausführen kann. Die Evaluierung der

Ausführung des Roboters sowie die dabei beobachteten Ergebnisse werden im folgenden Kapitel 4.6 Evaluierung der Ergebnisse beschrieben.

4.6 EVALUIERUNG DER ERGEBNISSE

Im Rahmen der Evaluierung wird untersucht, inwieweit die trainierte ACT-Policy die erlernte Sortieraufgabe auf dem Unitree G1 zuverlässig und autonom ausführen kann. Hierzu werden die Ausführungen hinsichtlich ihres Erfolgs sowie des dabei beobachteten Verhaltens des Roboters analysiert. Neben der erfolgreichen Ausführung der Aufgabe wird zusätzlich getestet, wie die Policy auf Abweichungen von den während des Trainings verwendeten Bedingungen reagiert.

Versuchsbedingungen

Als erfolgreich galt ein Versuch, wenn die Aufgabe vollständig abgeschlossen wurde, die Würfel also am Ende in der Reihenfolge B-R-P platziert waren und der Roboter wieder seine Ausgangsposition eingenommen hatte. Für die Würfel war lediglich ein mittels Markierung auf der Arbeitsfläche gekennzeichneteter Abstellbereich vorgegeben. Eine exakte Position war daher nicht erforderlich.

Bei der Positionierung der Würfel wurden einerseits Startpositionen gewählt, die den Positionen aus den Demonstrationsdaten ähnelten. Andererseits wurde untersucht, wie sich der Roboter bei größeren Abweichungen von den Trainingsdaten verhält. Hierzu wurden die Würfel auch in Positionen platziert, die weiter entfernt waren als in den Demonstrationen. Zusätzlich wurde auch getestet, wie sich das Modell verhält, wenn die Würfel in der Startreihenfolge B-R-P vorliegen. In den Demonstrationsdaten wurde ausschließlich die Startposition R-B-P verwendet. Somit lagen dem Modell keine Daten zu dieser Startreihenfolge vor.

Darüber hinaus wurden die Umgebungsbedingungen variiert. Hierzu wurden unterschiedliche Räume und Lichtquellen verwendet sowie Versuche an unterschiedlichen Tagen durchgeführt. Dadurch wurden sowohl der Bildhintergrund als auch die Beleuchtungsverhältnisse gegenüber den Bedingungen während der Demonstrationsdatenerfassung verändert.

Für die autonome Ausführung wurde das zuvor beschriebene Programm gestartet, woraufhin der Roboter die Sortieraufgabe in einer Endlosschleife auszuführen versuchte. Aufgrund des während der Demonstrationsdatenerfassung aufgetretenen Defekts an der linken Hand war bei einigen Ausführungen ein manueller Eingriff erforderlich. Die Würfel R und B, die vom Roboter mit der linken Hand gegriffen werden sollten, wurden dabei teilweise manuell leicht umpositioniert, damit sie trotz der eingeschränkten Ansteuerung des Daumens von der Hand gegriffen werden konnten.

Ergebnisse der autonomen Ausführung

Aufgrund der defekten linken Hand mussten die Würfel R und B oft umpositioniert werden, damit sie nach dem Greifen nicht wieder aus der Hand des Roboters fielen. Ursache hierfür war der mechanisch beschädigte Daumen, durch den die Würfel häufig nicht ausreichend fest gegriffen werden konnten. Ein zuverlässiges Weiterführen der Manipulation war nur möglich, wenn der Würfel nicht mit den Fingerspitzen, sondern mit der vollständigen Länge des Daumens gegriffen wurde. Der Würfel P, welcher mit der rechten Hand gegriffen wurde, konnte hingegen ohne vergleichbare Probleme aufgenommen und weiter manipuliert werden. In einzelnen Versuchsdurchläufen konnte der Roboter auch die Würfel R und B erfolgreich greifen, sofern sich diese in einer für die beschädigte Hand geeigneten Position befanden.

Auch bei vorherigen Pick-and-Place-Aufgaben mit dem Unitree G1 zeigte sich, dass das Greifen der Würfel mit einer unbeschädigten Hand grundsätzlich möglich war. Solange sich die Würfel innerhalb des durch die Trainingsdaten abgedeckten Bereichs befanden, konnten sie durch einen oder mehrere Greifversuche aufgenommen werden. Weichten die Startpositionen der Würfel jedoch zu stark von den in den Demonstrationen enthaltenen Positionen ab, konnte der Roboter den jeweiligen Würfel nicht greifen und blieb im Greifprozess stecken.

Die unbekannte Ausgangssituation aufgrund der Würfelreihenfolge B-R-P führte entgegen der Erwartung nicht dazu, dass der Roboter die Aufgabe unmittelbar abbrach. Stattdessen griff der Roboter zunächst den Würfel B und positionierte ihn an der vorgesehenen Zielposition. Beim anschließenden Versuch, den Würfel R zu greifen, blieb der Roboter jedoch im Greifprozess stecken. Der bereits positionierte Würfel B versperrte

den direkten Zugriff auf den nächsten Würfel. Auffällig war somit, dass der Roboter trotz der in den Demonstrationen nicht enthaltenen Startreihenfolge zunächst nach dem ersten Würfel griff und diesen anschließend an der vorgesehenen Position platzierte.

Die unterschiedlichen Räume sowie die damit einhergehenden Veränderungen des Bildhintergrunds führten zu keinen erkennbaren Veränderungen im Verhalten des Roboters. Dabei ist allerdings zu berücksichtigen, dass der Tisch einen Großteil des Kamerabildes einnahm und sich der Bildhintergrund zwischen den Versuchen daher nur geringfügig veränderte. Auch unterschiedliche Lichtquellen und deren Anzahl führten zu keinen wesentlichen Veränderungen im Verhalten des Roboters. Die durch unterschiedliche Lichtquellen entstehenden Schatten der Würfel wirkten sich ebenfalls nicht erkennbar auf die Ausführung der Aufgabe aus.

Analyse der Ergebnisse

Das beobachtete Problem beim Greifen der Würfel R und B lässt sich im Wesentlichen auf die mechanisch beschädigte linke Inspire-Hand zurückführen. Da der Würfel P mit der funktionierenden rechten Inspire-Hand ohne vergleichbare Probleme gegriffen werden konnte und mit der linken Inspire-Hand im unbeschädigten Zustand bereits zuvor Pick-and-Place-Aufgaben problemlos ausgeführt werden konnten, kann die eingeschränkte Greiffunktion der linken Inspire-Hand als wesentliche Einflussgröße betrachtet werden.

Die wiederholten Greifversuche, die oftmals notwendig waren, weil der Würfel beim ersten Versuch nicht direkt gegriffen werden konnte, können zudem auf die begrenzte Genauigkeit und die geringe Steifigkeit des Unitree G1 zurückgeführt werden. Darüber hinaus zeigt die Abhängigkeit von den in den Demonstrationen enthaltenen Startpositionen der Würfel eine begrenzte Generalisierungsfähigkeit der Policy. Durch das Erreichen von Zuständen, die deutlich von den während des Trainings beobachteten Zuständen abweichen, kann die Policy keine zuverlässige Aktion ableiten. Die Anzahl und Variabilität der durch die Demonstrationen abgedeckten Zustände haben daher einen wesentlichen Einfluss auf die Robustheit der Policy.

Das Verhalten der Policy bei der unbekannten Startreihenfolge deutet darauf hin, dass die Policy nicht ausschließlich eine fest erlernte räumliche Reihenfolge der Würfel

abarbeitet. Obwohl die Startreihenfolge in den Demonstrationen nicht enthalten war, griff die Policy zunächst wie bei den Demonstrationen den ersten Würfel von links. In diesem Fall befand sich dort allerdings der Würfel B anstelle des Würfels R. Der gegriffene Würfel wurde jedoch nicht wie in den Demonstrationen an der für den Würfel R vorgesehenen Zielposition, sondern an der für den Würfel B vorgesehenen Zielposition platziert. Dies deutet darauf hin, dass die Policy die Platzierung der Würfel nicht ausschließlich anhand einer fest erlernten räumlichen Reihenfolge vornimmt. Bei der Auswahl der Zielposition werden vermutlich neben der räumlichen Position des Würfels auch weitere visuelle Merkmale berücksichtigt. Aufgrund der geringen Anzahl entsprechender Versuche sowie des Black-Box-Charakters des Modells kann daraus jedoch keine allgemeingültige Aussage über die Entscheidungsstrategie der Policy abgeleitet werden. Das anschließende Problem beim Greifen des nächsten Würfels könnte durch zusätzliche Demonstrationen ähnlicher Zustände reduziert werden, da dadurch eine größere Bandbreite möglicher Zustände in den Trainingsdaten abgedeckt werden könnte.

5 ZUSAMMENFASSUNG/SCHLUSSFOLGERUNG

Ziel dieser Arbeit war die Erstellung von Applikationen für humanoide Robotersysteme. Hierzu sollte ein vollständiger IL-Workflow umgesetzt und auf dem Unitree G1 getestet werden. Der humanoide Roboter sollte dabei am Ende eine Pick-and-Place-Aufgabe autonom ausführen können.

Dafür wurde zunächst die Entwicklungsumgebung auf Grundlage der von Unitree bereitgestellten Repositories aufgesetzt. Da diese insbesondere hinsichtlich der verwendeten Softwareversionen einige Konflikte aufwiesen, mussten die Umgebungen fortlaufend angepasst und korrigiert werden. Auch die Einbindung der FTP Inspire-Hand musste zusätzlich umgesetzt werden. Anschließend wurden mithilfe einer VR-Brille Demonstrationsdaten aufgezeichnet und für das Modelltraining aufbereitet. Auf Grundlage des Demonstrationsdatensatzes wurde eine ACT-Policy trainiert und anschließend auf dem realen Robotersystem deployed.

Die Ergebnisse zeigen, dass einfache Pick-and-Place-Aufgaben mittels IL bereits mit vergleichsweise geringem Programmieraufwand umgesetzt werden können. Im Gegensatz zur manuellen Programmierung von Bewegungsabläufen ermöglicht IL die Erfassung von Demonstrationsdaten durch Expertinnen und Experten, ohne dass diese über spezifische Programmierkenntnisse verfügen müssen. Dadurch können komplexe Aufgaben anhand von Demonstrationen erlernt werden, ohne deren Bewegungsabläufe vollständig manuell programmieren zu müssen.

Gleichzeitig zeigte sich, dass bereits geringfügige Änderungen der Ausgangssituation zu Problemen führen können, wenn diese nicht durch die Demonstrationsdaten abgedeckt sind. Insbesondere bei deutlich abweichenden Positionen der Würfel konnte die Policy die Aufgabe nicht zuverlässig abschließen. Die Ergebnisse bestätigen damit die bereits in den theoretischen Grundlagen beschriebenen Herausforderungen hinsichtlich der Generalisierungsfähigkeit von IL. Gelangt die Policy in Zustände, die nicht oder nur unzureichend durch die Demonstrationen abgedeckt sind, kann sie keine zuverlässige Aktion ableiten. Dieses Verhalten steht in Zusammenhang mit dem Covariate-Shift-Problem, bei dem sich die während der Ausführung auftretenden Zustände von der im Trainingsdatensatz abgebildeten Zustandsverteilung unterscheiden. Die Robustheit der

erlernten Policy hängt somit wesentlich von der Variabilität und damit auch von der Anzahl der durch die Demonstrationen abgedeckten Situationen ab.

Eine wesentliche Erkenntnis dieser Arbeit ist somit, dass IL zwar eine effiziente Grundlage für das Erlernen einfacher Manipulationsaufgaben bildet, die Generalisierungsfähigkeit gegenüber unbekannten Situationen jedoch begrenzt ist. Zur Verbesserung dieser Robustheit können ergänzend weitere Lernverfahren eingesetzt werden. Ein möglicher Ansatz besteht in der anschließenden Optimierung der erlernten Policy mithilfe von RL. Darüber hinaus könnten VLA-Modelle eine weitergehende Generalisierung ermöglichen, indem Aufgaben allgemeiner anhand visueller Informationen und natürlichsprachlicher Anweisungen beschrieben werden und dadurch nicht für jede Objektvariante separate Demonstrationen erforderlich wären.

Zusammenfassend konnte gezeigt werden, dass der Unitree G1 mithilfe von IL aus einer vergleichsweise kleinen Anzahl an Demonstrationen eine einfache Manipulationsaufgabe erlernen und unter ähnlichen Bedingungen autonom ausführen kann. Für eine zuverlässigere Ausführung unter variierenden Bedingungen sind insbesondere vielfältigere Demonstrationsdaten erforderlich. Um die Genauigkeit der Ausführung zu erhöhen, wird des Weiteren der Einsatz eines industrietauglichen humanoider Roboters mit höherer Steifigkeit empfohlen. Eine Kombination von IL mit weiterführenden Lernverfahren wie RL stellt darüber hinaus einen möglichen Ansatz für zukünftige Entwicklungen dar.

6 ABKÜRZUNGSVERZEICHNIS

Abkürzung	Erklärung
ACT	Action Chunking with Transformers
BC	Behavioral Cloning
DoF	Degrees of Freedom
IL	Imitation Learning
LfD	Learning from Demonstration
LLM	Large Language Model
ML	Machine Learning
RL	Reinforcement Learning
VLA	Vision-Language-Action
VLM	Vision-Language Model
VR	Virtual Reality

7 ABBILDUNGSVERZEICHNIS

Abb. 1: Vergleich aktueller humanoider Robotersysteme (eigene Darstellung nach [vgl. 10], [vgl. 11], [vgl. 12], [vgl. 13], [vgl. 14])	6
Abb. 2: Agent-Environment Interaction im RL [26, S. 48]	13
Abb. 3: Auswirkungen des Covariate-Shift-Problems (eigene Darstellung, erstellt mit ChatGPT GPT-5.5 nach [vgl. 32])	19
Abb. 4: Versuchsaufbau zur Durchführung der Sortieraufgabe.....	41
Abb. 5: Betriebsmodi des Unitree G1 [44]	46
Abb. 6: Ergebnis der autonomen Sortierung durch die trainierte Policy	47

8 TABELLENVERZEICHNIS

Tabelle 1: Aktuelle humanoide Robotersysteme im Vergleich (eigene Darstellung).....	5
Tabelle 2: Vergleich von RL, IL und VLA (eigene Darstellung nach [vgl. 25, S. 1-10], [vgl. 26, S. 1-6], [vgl. 28, S. 3-5], [vgl. 29, S. 7173-7174], [vgl. 33], [vgl. 34, S. 1-3])	26
Tabelle 3: Verwendete Hardware für Demonstrationsdatenaufzeichnung und Training	37
Tabelle 4: Verwendete Software- und Framework-Versionen sowie Unitree Repository	39

9 LITERATURVERZEICHNIS

- [1] A. Imran und K. Gopalakrishnan, *AI for robotics: Toward embodied and general intelligence in the physical world*. New York: Apress, 2025.
- [2] U.S.-China Economic and Security Review Commission, "Humanoid Robots," U.S.-China Economic and Security Review Commission, Okt. 2024. Zugriff am: 25. Juni 2026.
- [3] Y. Tong, H. Liu und Z. Zhang, "Advancements in Humanoid Robots: A Comprehensive Review and Future Prospects," *IEEE/CAA J. Autom. Sinica*, Jg. 11, Nr. 2, S. 301–328, 2024, doi: 10.1109/JAS.2023.124140.
- [4] T. Uthai *et al.*, "Opportunities challenges and roadmap for humanoid robots in construction," *Scientific reports*, Early Access. doi: 10.1038/s41598-025-30252-6.
- [5] Figure AI. "Figure." Zugriff am: 2. Juli 2026. [Online.] Verfügbar: <https://www.figure.ai/figure>
- [6] UBTECH Robotics. "Walker S2." Zugriff am: 3. Juli 2026. [Online.] Verfügbar: https://www.ubtrobot.com/en/humanoid/products/walker-s2?srsId=AfmBOoqAFmDs5nEH0hH7KHLT4cvunllc_WwDW4lccq1_71R9M5VxY0Jv
- [7] Fourier. "GR-3 Series: A Caring and Capable Companion." Zugriff am: 3. Juli 2026. [Online.] Verfügbar: <https://www.fftai.com/products-gr3series>
- [8] Fourier. "Fourier GR-1: Your Ideal Robot Assistant For Everything." Zugriff am: 25. Juni 2026. [Online.] Verfügbar: <https://www.fftai.com/products-gr1>
- [9] Unitree Robotics. "Unitree G1: Humanoid Agent AI Avatar." Zugriff am: 25. Juni 2026. [Online.] Verfügbar: <https://www.unitree.com/g1>
- [10] Leobotics. "Robot humanoide IA performant autonome Figure 03." Zugriff am: 6. Juli 2026. [Online.] Verfügbar: <https://es.leobotics.com/comparateur-robot/robot-humanoide-ia-performant-autonome-figure-03>

- [11] Robots Europa. "UBTECH Walker S2 Industrial Humanoid Robot + Free Delivery Anywhere in Europe." Zugriff am: 6. Juli 2026. [Online.] Verfügbar: <https://www.robotseuropa.com/UBTECH-Walker-S2.htm>
- [12] A. Kim und S. Maini. "Agility Robotics: Business Breakdown & Founding Story." Zugriff am: 6. Juli 2026. [Online.] Verfügbar: <https://research.contrary.com/company/agility-robotics>
- [13] Robots International. "Fourier GR-3 Care-Bot Humanoid Robot." Zugriff am: 6. Juli 2026. [Online.] Verfügbar: <https://www.robotsinternational.com/Fourier-GR-3-Care-Bot-Humanoid-Robot.htm>
- [14] TONEART Shop. "Unitree G1 EDU-U1 – Humanoider Roboter: 0% Leasing." Zugriff am: 30. Juli 2026. [Online.] Verfügbar: <https://www.toneart-shop.at/unitree-g1-edu-u1.html>
- [15] Figure AI. "Figure: The future of home help is here." Zugriff am: 2. Juli 2026. [Online.] Verfügbar: <https://www.figure.ai/>
- [16] UBTECH Robotics. "About Us: A leading Humanoid Robots and Smart Service Robots Company." Zugriff am: 3. Juli 2026. [Online.] Verfügbar: <https://www.ubtrobot.com/en/about/company-profile#briefIntroduction>
- [17] Agility Robotics. "Humanoid Solutions: One cohesive platform. Endless applications." Zugriff am: 2. Juli 2026. [Online.] Verfügbar: <https://www.agilityrobotics.com/solutions>
- [18] Fourier. "Fourier GR-3: Interaction-Oriented Service Humanoid Robot." Zugriff am: 3. Juli 2026. [Online.] Verfügbar: <https://www.fftai.com/products-gr3>
- [19] Unitree Robotics. "About Unitree." Zugriff am: 26. Juni 2026. [Online.] Verfügbar: <https://www.unitree.com/about>
- [20] X. Zhang, G. Li, S. Shokouhi und M.-W. L. Thein, "Modern Control Meets Machine Learning: A Review and Taxonomy of Synergistic Approaches for

- Robotics Applications," *Actuators*, Jg. 15, Nr. 5, 2026, Art. Nr. 235, doi: 10.3390/act15050235.
- [21] M. Rybczak, N. Popowniak und A. Lazarowska, "A Survey of Machine Learning Approaches for Mobile Robot Control," *Robotics*, Jg. 13, Nr. 1, 2024, Art. Nr. 12, doi: 10.3390/robotics13010012.
- [22] S. L. Brunton und J. N. Kutz, *Data Driven Science & Engineering: Machine Learning, Dynamical Systems, and Control*, 1. Aufl. Cambridge: Cambridge University Press, 2022. Zugriff am: 8. Juli 2026.
- [23] U. Lorenz, *Reinforcement learning: Aktuelle Ansätze verstehen - mit Beispielen in Java und Greenfoot*, 2. Aufl. (Moremedia). Berlin, Heidelberg: Springer Vieweg, 2024.
- [24] R. Reiter *et al.*, "Synthesis of Model Predictive Control and Reinforcement Learning: Survey and Classification," Rep. 2502.02133v1, Feb. 2025. Zugriff am: 8. Juli 2026.
- [25] J. Kober, J. A. Bagnell und J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, Jg. 32, Nr. 11, S. 1–73, 2013, doi: 10.1177/0278364913495721.
- [26] R. S. Sutton und A. G. Barto, *Reinforcement Learning: An Introduction* (A Bradford Book). Cambridge, Massachusetts: The MIT Press, 2015. Zugriff am: 9. Juli 2026.
- [27] A. Vahabpour, "Self-Organizing Generative Models for Diverse Imitation Learning," Dissertation, University of California, Los Angeles, Los Angeles, California, 2022.
- [28] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel und J. Peters, "An Algorithmic Perspective on Imitation Learning," *Foundations and Trends® in Robotics*, 2017.

- [29] M. Zare, P. M. Kebria, A. Khosravi und S. Nahavandi, "A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges," *IEEE transactions on cybernetics*, Early Access.
- [30] P. Santana, *An Introduction to Deep Reinforcement and Imitation Learning*. Lisbon, Portugal, 2025. Zugriff am: 15. Juli 2026.
- [31] Q. Liu und Y. Wu, "Supervised Learning," in *Web Data Mining*, S. 63–132. Zugriff am: 15. Juli 2026.
- [32] D. Garg. "Learning to Imitate." SAIL Blog. Zugriff am: 16. Juli 2026. [Online.] Verfügbar: <https://ai.stanford.edu/blog/learning-to-imitate/>
- [33] K. Kawaharazuka, J. Oh, J. Yamada, I. Posner und Y. Zhu, "Vision-Language-Action Models for Robotics: A Review Towards Real-World Applications," *IEEE Access*, Jg. 13, 2025.
- [34] Y. Ma, Z. Song, Y. Zhuang, J. Hao und I. King, "A Survey on Vision--Language--Action Models for Embodied AI," *IEEE transactions on neural networks and learning systems*, Jg. 37, Nr. 7, S. 1–54, 2026.
- [35] Xiaofeng Han *et al.*, "Multimodal Fusion and Vision-Language Models: A Survey for Robot Vision," Rep. 2504.02477v3, Okt. 2025. Zugriff am: 18. Juli 2026.
- [36] A. Correia und L. A. Alexandre, "A survey of demonstration learning," *Robotics and Autonomous Systems*, Jg. 182, 2024, Art. Nr. 104812.
- [37] W. Li, Y. Wang, Y. Liang und D. T. Pham, "Learning from demonstration for autonomous generation of robotic trajectory: Status quo and forward-looking overview," *Advanced Engineering Informatics*, Jg. 62, 2024, Art. Nr. 102625.
- [38] T. Z. Zhao, V. Kumar, S. Levine und C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," Rep. 2304.13705v1, 2023. Zugriff am: 24. Juli 2026.

- [39] H. Liu, S. Nasiriany, L. Zhang, Z. Bao und Y. Zhu, "Robot learning on the job: Human-in-the-loop autonomy and learning during deployment," *The International Journal of Robotics Research*, Early Access.
- [40] Akhil S Anand, Shambhuraj Sawant, Jasper Hoffmann, Dirk Reinhardt und Sebastien Gros, *Closing the Sim2Real Performance Gap in RL*, 2025. Zugriff am: 27. Juli 2026.
- [41] K. Z. Mirza und S. Singh, "Imitation learning for legged robot locomotion: a survey," *Frontiers in robotics and AI*, Early Access. doi: 10.3389/frobt.2025.1678567.
- [42] S. Ross, G. J. Gordon und J. A. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," Fort Lauderdale, Florida, USA, Rep. 15, 2011.
- [43] E. Aljalbout *et al.*, "The Reality Gap in Robotics: Challenges, Solutions, and Best Practices," *Annual Review of Control, Robotics, and Autonomous Systems*, Early Access.
- [44] Quadruped Robotics uG. "Controls (Firmware V1.0.2)." Zugriff am: 12. August 2026. [Online.] Verfügbar: https://docs.quadruped.de/projects/g1/html/operation_1.2.html

10 ANHANG

Erklärung zum Einsatz generativer KI und KI-gestützter Technologien im Schreibprozess:

Zur Verbesserung der Kohärenz und Lesbarkeit dieser Arbeit wurde das KI-Sprachmodell ChatGPT (GPT-5.5, OpenAI) unterstützend eingesetzt. Die Nutzung erfolgte ausschließlich zur sprachlichen Überarbeitung (z. B. Formulierungen, Grammatik und Stil) sowie zur Unterstützung bei der Erstellung einzelner schematischer Abbildungen. Sämtliche Textpassagen und Abbildungen wurden anschließend von der Verfasserin geprüft und bei Bedarf überarbeitet. Die Verantwortung für Inhalt, Argumentation, Quellenarbeit und Schlussfolgerungen liegt vollständig bei der Verfasserin.