Experiment zur Verbesserung der Robustheit von Reinforcement Learning Modellen in Drohnensimulationen anhand trainiertem Gegenspieler

Bachelorarbeit

vorgelegt am 5. Mai 2023

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs WWI2020F

von

Leon Henne

Betreuerin in der Ausbildungsstätte:	DHBW Stuttgart:
IBM Deutschland GmbH Sophie Lang Senior Data Scientist	Prof. Dr. Kai Holzweißig Studiendekan Wirtschaftsinformatik
Unterschrift der Betreuerin	

Inhaltsverzeichnis

Abkürzungsverzeichnis

A2C Advantage Actor-Critic

A3C Asynchronous Advantage Actor-Critic

CSV kommaseparierte Wertedatei

DART Dynamic Animation and Robotics Toolkit

DHBW Duale Hochschule Baden-Württemberg

DQL Deep Q-Learning

DR Domain Randomization

IL Imitation Learning

KPI Key Performance Indicator

MARL Multi-Agent Reinforcement Learning

ODE Open Dynamics Engine

PPO Proximal Policy Optimization

RAEARL Robust Adaptive Ensemble Adversarial Reinforcement Learning Framework

RL Reinforcement Learning

ROS Robot Operating System

SITL Software-in-the-Loop

TRPO Trust Region Policy Optimization

Abbildungsverzeichnis

Tabellenverzeichnis

[1] Ivaldi/Padois/Nori 2014, S. 4
[2] Pullum 2022, S.17
[3] Pullum 2022, S.19

1 Einleitung

1.1 Problemstellung

Reinforcement Learning (RL) findet heutzutage bereits Anwendung in vielerlei Forschungsprojekten wie Deepmind AlphaStar oder OpenAI Five, aber auch in Produkten und Dienstleistungen wie AWSDeepRacer oder Meta's Horizon open-source RL-Plattform.[4] RL ist im Bereich des maschinellen Lernens eine Herangehensweise zur Lösung von Entscheidungsproblemen.[5] Ein Software-Agent leitet dabei durchzuführende Aktionen aus seiner Umgebung ab, mit dem Ziel die kumulierte erhaltene Belohnung zu maximieren, währenddessen sich seine Umgebung durch diverse Aktionen verändert.[6] Die Umgebungen beinhalten in ihrer einfachsten Form eine simulierte Welt, welche zu jedem Zeitschritt eine Aktion entgegennimmt, und den eigenen nächsten Zustand sowie einen Belohnungswert zurückgibt.[7] Da das Sammeln von Daten in der echten Welt für den Einsatz von RL Algorithmen eine Limitierung darstellen kann, werden häufig fürs Training Simulationsumgebungen eingesetzt.[8] Eine Limitierung können bspw. Sicherheitsaspekte sein, welche beim Training von Roboterarmen, oder sich autonom bewegenden Systemen auftreten, da die einzelnen physischen Bewegungen nicht vorhersehbar abzuschätzen sind.[9] Simulationen nehmen damit zum einen als Testumgebung eine wichtige Rolle in der Entwicklung von Kontrollalgorithmen ein.[10] Zum anderen bedarf die erfolgreiche Anwendung von RL neben effizienten Algorithmen auch geeignete Simulationsumgebungen.[11] Besonders schwierig, und daher sehr wichtig zu erforschen, ist die Anpassung der Trainingsumgebung an die reale Welt, sodass bspw. Agenten für Roboter und autonome Fahrzeuge nach dem Training mit robusten Strategien in der Realität eingesetzt werden können.[12] In der Forschungsliteratur wird diese Problematik als „Sim to real"-Transfer beschrieben.[13]

Ein Forschungsgebiet, bei dem die Lösung des Sim to Real Transfers betrachtet wird, ist die autonome Steuerung von unbemannten Luftfahrzeugen bzw. Drohnen.[14] Das Transferproblem entsteht bspw. dabei, dass zur automatisierten Kollisionsvermeidung die Kollisionsbeispiele einer Simulationsumgebung entnommen werden, um physischen Schaden oder Drohnen Verlust zu vermeiden.[15] Drohnen tragen dabei bereits in der heutigen Zeit zur Lösung vieler komplexer Aufgaben, wie der Katastrophenüberwachung oder der Waldbrandbekämpfung bei.[16] Ein weiterer

[4] Vgl. Li 2019, S. 4
[5] Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 3
[6] Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 3
[7] Vgl. Reda/Tao, T./van de Panne 2020, S. 1
[8] Vgl. Zhao/Queralta/Westerlund 2020, S. 737
[9] Vgl. Zhao/Queralta/Westerlund 2020, S. 738
[10] Vgl. Cutler/Walsh/How 2014, S. 2
[11] Vgl. Reda/Tao, T./van de Panne 2020, S. 8
[12] Vgl. Slaoui u. a. 2019, S. 1
[13] Vgl. Zhao/Queralta/Westerlund 2020, S. 738
[14] Vgl. Deshpande/Minai/Kumar, M. 2021, S. 1
[15] Vgl. Sadeghi/Levine 2016, S. 4
[16] Vgl. Hentati u. a. 2018, S. 1495

wichtiger Bereich stellt die Anwendung von Drohnensimulationen in militärischen und sicherheitsbezogenen Kontexten dar.[17] Zusätzlich steigen die komplexen Einsatzanforderungen immer weiter, unter dem Anstieg an Anwendungsgebieten für unbemannte Luftfahrzeuge.[18]

Die Simulation einer möglichst realistischen Umgebung in diesem Kontext wird in der Forschung häufig mit dem Ansatz von Domain Randomization (DR) begleitet.[19] Unter dem Themenfeld der DR wird die Idee erforscht, anstelle der akkuraten Modellierung realistischer Dynamiken, diese so stark zu randomisieren, dass reale Dynamikeffekte abgedeckt sind.[20] Neben den dynamischen Bedingungen unterliegt die Realität jedoch häufig auch dem Einfluss mehrerer Parteien. Diese tragen teilweise kooperierend aber auch teilweise konkurrierend zum eigenen Erfolg bei, wie z.B. im Rahmen eines dem Wettbewerb unterliegenden Marktes.[21] Stellt man sich ein Szenario im Kontext kooperativer oder konkurrierender Drohnen vor, ist es naheliegend, dass auch jene Einflüsse möglichst präzise in die Simulationsumgebung integriert sein müssen, um ein robustes Modell erlernen zu können. Während bereits in Produkten wie PowerTAC von Collins/ Ketter 2022 die Simulation von Märkten entwickelt wurde, scheint der Einfluss des Gegenspielers in kompetitiven Drohnensimulationen auf die Robustheit von RL Algorithmen, und demnach auf die Lösung des „Sim to real“-Transfers, unerforscht.

1.2 Zielsetzung

Im Rahmen dieser Arbeit soll eine Simulationsumgebung entwickelt werden, in welcher sich RL basierter Gegenspieler und deterministische Gegenspieler integrieren lassen. Anschließend ist zu untersuchen, wie sich die Wahl des Trainingsgegenspielers auf das Leistungsverhalten der RL Modelle Strategien, im Kontext von RL oftmals als Policies referenziert, unter verändertem Testszenario auswirkt.

Dazu soll eine kompetitive Simulationsumgebung entwickelt werden, in welcher sich zwei konkurrierender Spieler in Form von Flugobjekten spielerisch gegenseitig bekämpfen. In der Simulation werden folgende Policies in drei verschiedenen Szenarien optimiert.

- Training mit regelbasiertem Gegenspieler unter gleichbleibenden Dynamikparametern
- Training mit RL basiertem Gegenspieler unter gleichbleibenden Dynamikparametern
- Training mit regelbasiertem Gegenspieler unter sich verändernden Dynamikparametern

Anschließend werden alle trainierten Policies in einem Testszenario untersucht. Das Testszenario verfügt dabei über festgelegte, sich vom Training unterscheidende Eigenschaften, wie Flugbahnen oder Start- und Zielpositionen. Außerdem wird ein deterministischer Gegenspieler unter im

[17] Vgl. Fitwi u. a. 2019, S. 1
[18] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 1
[19] Vgl. Sadeghi/Levine 2016, S. 1
[20] Vgl. Zhao/Queralta/Westerlund 2020, S. 4f.
[21] Vgl. Collins/Ketter 2022, S. 2

Gegensatz zum Training, veränderten Handlungspräferenzen eingesetzt. Bei der Untersuchung werden jeweils die folgenden Variablen als Key Performance Indicator (KPI) betrachtet.

- kumulierte erzielte Belohnung
- Varianz der Belohnungen
- Anzahl an Misserfolgen

Durch die Auswertung des Testszenarios kann der Effekt des RL basierten Gegenspielers auf die Robustheit evaluiert werden, indem die Leistungsdiskrepanz im Testszenario zwischen den Strategien im Training mit RL- und regelbasierten Gegenspielers- und der Domain Randomization verglichen wird.

1.3 Forschungsfrage

Aus der beschriebenen Problemstellung und der für den Rahmen dieser Arbeit festgelegten Zielsetzung ergibt sich folgende Forschungsfrage:

Inwiefern kann durch den Einsatz eines mittels RL trainierten Gegenspielers die Robustheit einer optimierten Policy verbessert werden?

Zur Beantwortung der Forschungsfrage werden folgende Hypothesen aufgestellt und im Rahmen der Arbeit untersucht:

Hypothese 1: *Die im Testszenario erzielte kumulierte Belohnung ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig höher, als die Policy aus dem Training mit regelbasiertem Gegenspieler.*

Hypothese 2: *Die Varianz der im Testszenario erzielten kumulierten Belohnung ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig geringer, als die Policy aus dem Training mit regelbasiertem Gegenspieler.*

Hypothese 3: *Die im Testszenario erreichte Anzahl von Misserfolgen ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig geringer, als die Policy aus dem Training mit regelbasiertem Gegenspieler.*

1.4 Forschungsmethodik

Als Forschungsmethodik soll im Rahmen dieser Arbeit ein quantitatives Laborexperiment nach Recker 2021 durchgeführt werden. Hierbei wird häufig nach dem hypothetisch-deduktiven Modell vorgegangen, in welchem Hypothesen formuliert, empirische Studien entwickelt, Daten gesammelt, Hypothesen anhand dieser evaluiert und gewonnene Erkenntnisse berichtet werden.[22] Damit

[22] Vgl. Recker 2021, S. S.89f.

stellt das Laborexperiment eine Möglichkeit für die Untersuchung der Ursache- und Wirkungsbeziehung dar.[23] Dabei wird die kontrollierte Umgebung der Simulation erschaffen, deren Aufbau die unabhängige Variable des Laborexperiments darstellt. Die Metriken, auf deren Grundlage die Leistung und die Robustheit der trainierten Policies gemessen werden, bilden im Experiment die abhängigen Variablen, zu dessen Grundlage die Forschungsfrage beantwortet wird.

1.5 Aufbau der Arbeit

Die Arbeit beginnt mit einem einleitenden Kapitel, in welchem Motivation, Problemstellung, Zielsetzung und Forschungsmethodik erläutert sind. Anschließend wird im zweiten Kapitel der aktuelle Stand der Forschung zu den relevanten Konzepten beschrieben. Im dritten Kapitel wird die Forschungsmethodik dargestellt, indem die Simulationsumgebung als Messinstrument entwickelt wird, verschiedene Messszenarien erläutert und entsprechende Daten gesammelt werden. Daraufhin erfolgt im vierten Kapitel die Auswertung der Messdaten sowie die Beantwortung der Forschungsfrage durch Annahme oder Ablehnung der eigens aufgestellten Hypothesen. Im Zuge dessen kann ebenso die Forschungsfrage anhand der Annahme oder Ablehnung der Hypothesen beantwortet werden. Abschließend wird im letzten Kapitel ein Fazit gezogen, die Forschungsmethodik kritisch reflektiert und ein Ausblick auf nachfolgende Forschungsansätze gegeben.

[23] Vgl. Recker 2021, S. 106

2 Diskussion des aktuellen Stands der Forschung und Praxis

2.1 Aufbau der Literaturrecherche

In Anlehnung an die Literaturrecherche nach Webster/Watson 2002 wurden alle voraussichtlich benötigten Konzepte für die Durchführung der beschriebenen Forschungsmethodik in Tabelle 1 festgehalten. Alle angeführten Konzepte wurden mittels verschiedener Suchbegriffe in Suchmaschinen, Datenbanken und Bibliotheken wie *Google Scholar* 2/28/2023, *IEEE Xplore* 2/28/2023 oder die digitale Bibliothek der Association for Computing Machinery (ACM) ACM Digital Library 2/28/2023 recherchiert. In der akquirierten Literatur wurden zitierte Werke nach den beschriebenen Konzepten durchsucht und insgesamt jede Literaturquelle in Tabelle 1 den in ihnen enthaltenen Konzepten zugeordnet.

Artikel	Konzepte							
	RL	MARL	IL	ES	DS	KS	DR	RRLP
Sutton/Barto 2018	X							
Li 2019	X							
Zhao/Queralta/Westerlund 2020	X			X			X	
Wang/Hong 2020	X							
Zhang/Wu/Pineau 2018	X			X				X
Cutler/Walsh/How 2014	X			X				
Canese u. a. 2021	X	X						
Reda/Tao, T./van de Panne 2020	X			X			X	
Ningombam 2022	X							
Arulkumaran u. a. 2017	X							
Huang u. a. 2017	X							
Mnih/Kavukcuoglu u. a. 2013	X							
Wong u. a. 2022	X	X						
Hussein u. a. 2017	X		X					
Attia/Dayan 2018			X					
Gao u. a. 2014			X					
Fang u. a. 2019			X					
Balakrishna u. a. 2020			X					
Schuderer/Bromuri/van Eekelen 2021	X	X		X				
Körber u. a. 2021				X				
Bharadhwaj u. a. 2019				X			X	
Foronda 2021				X				
Anu 1997				X				

Artikel	Konzepte							
Brockman u. a. 2016	X			X				
Yan Duan u. a. 2016	X			X				X
Ivaldi/Padois/Nori 2014				X				
Ayala u. a. 2020				X				
Todorov/Erez/Tassa 2012				X				
Koch u. a. 2018	X				X			
Deshpande/Kumar, R. u. a. 2020	X				X			
Deshpande/Minai/Kumar, M. 2021					X		X	X
Hentati u. a. 2018					X			
Molchanov u. a. 2019					X		X	X
Furrer u. a. 2016					X			
Silano/Iannelli 2019					X			
Shah u. a. 2017					X			
Panerati u. a. 2021				X	X			
Moos u. a. 2022								X
Pullum 2022								X
Liu u. a. 2023								X
Yan Duan u. a. 2016								X
Schott/Hajri/Lamprier 2022	X					X		
Pinto u. a. 2017	X					X		X
Pan u. a. 2021						X		
Zhai u. a. 2022						X		X
Tobin u. a. 2017							X	
Chen u. a. 2021							X	
Hsu u. a. 2023							X	
Alghonaim/Johns 2021							X	
Sadeghi/Levine 2016							X	

Tab. 1: Konzeptmatrix für Artikel zu Simulationsumgebungen und zur Robustheit RL Algorithmen nach Webster/Watson 2002. Legende: RL (Reinforcement Learning), MARL (Multi-Agent Reinforcement Learning), IL (Imitation Learning), ES (Entwicklung von Simulationsumgebungen), DS (Drohnensimulation), KS (kompetitive Simulationsumgebungen), DR (Domain Randomization), RRLP (Robustheit von RL Policies)

2.2 Verstärkendes Lernen

Verstärkendes Lernen, oder auch als RL in der Fachsprache bezeichnet, definiert einen konzeptionellen Ansatz, zielorientiertes Lernen von Entscheidungen zu verstehen und zu automatisieren.[24]

[24] Vgl. Sutton/Barto 2018, S. 13

Dabei liegt der Fokus darauf, dass ein Modell, welches of als Agent bezeichnet wird, aus der direkten Interaktion mit seiner Umgebung lernt, ohne dass explizite Überwachung notwendig ist.[25] Der Agent lernt über die Zeit eine optimale Strategie zur Lösung des Entscheidungsproblems durch Ausprobieren und Scheitern die gewünschte Veränderung in seiner Umwelt herzustellen.[26] Dafür ist es notwendig, dass der Agent den Zustand seiner Umgebung wahrnehmen und durch entsprechende Aktionen beeinflussen kann, um das Erreichen des Zielzustandes zu ermöglichen.[27] Dazu muss der Agent diejenigen Aktionen finden, die ihm die größtmögliche kumulierte Belohnung liefern, wobei Aktionen nicht nur die unmittelbaren, sondern auch zukünftige Belohnungen beeinflussen.[28] Zusammengefasst lässt sich die beschriebene Interaktion des Agenten mit seiner Umgebung wie in Abbildung 1 darstellen.

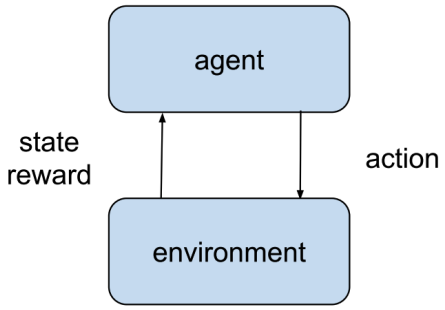


Abb. 1: vereinfachte Darstellung der Interaktion zwischen dem Agenten und seiner Umgebung[29]

Ein Standardaufbau einer Aufgabe für verstärkendes Lernen kann demnach als sequentielles Entscheidungsproblem verstanden werden, zu dessen Lösung ein Agent zu jedem diskreten Zeitschritt eine Aktion ausführt, welche den Zustand der Umgebung verändert.[30] Betrachtet man die technische Umsetzung einer solchen Interaktion zwischen dem Agenten und dessen Umgebung, wird häufig zur Modellierung ein Markov Entscheidungsprozess verwendet. Im Kontext von RL ist der Entscheidungsprozess definiert als ein Tupel aus folgenden Elementen:[31]

- Zustandsraum S
- Aktionsraum A
- initiale Zustandsverteilung $p_0(S)$
- Übergangswahrscheinlichkeit $T(S_{t+1}|S_t, A_t)$
- Belohnungswahrscheinlichkeit $R(r_{t+1}|S_t, A_t)$

[25] Vgl. Sutton/Barto 2018, S. 13
[26] Vgl. Li 2019, S. 4
[27] Vgl. Sutton/Barto 2018, S. 2
[28] Vgl. Sutton/Barto 2018, S. 1
[29] Enthalten in: Li 2019, S. 5
[30] Vgl. Zhao/Queralta/Westerlund 2020, S. 2
[31] Vgl. Zhang/Wu/Pineau 2018, S. 2

Zum Finden der optimalen Strategie existieren modellbasierende und modellfreie Algorithmen des verstärkenden Lernens.[32] Bei modellbasierenden Algorithmen wird das Umgebungsverhalten, also die Übergangs- und Belohnungswahrscheinlichkeiten, als bekannt vorausgesetzt.[33] Als modellbasierender Algorithmus wird z. B. die dynamische Programmierung eingesetzt, um mittels Strategieevaluation und Strategieiteration eine optimale Strategie zu finden.[34] Unter modellfreien Algorithmen wird zwischen einem wertbasierten-, einen strategiebasierten und einem Akteur-Kritik-Ansatz unterschieden.[35] Der Agent im Kontext von modellfreien RL Methoden kennt dabei nur die Zustände S und die Aktionen A, jedoch nicht das Umgebungsverhalten T oder die Belohnungswahrscheinlichkeit R.[36] Abbildung 2 stellt eine Klassifizierung der Algorithmen und Methoden von RL dar.

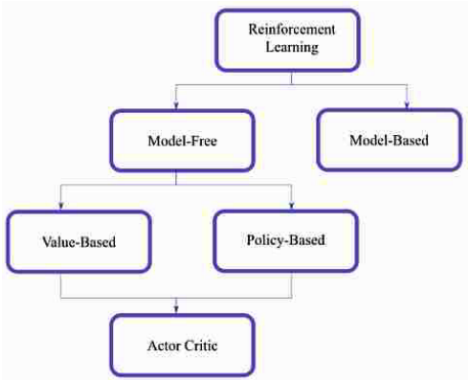


Abb. 2: Klassifizierung von Algorithmen im Bereich des RL[37]

2.2.1 Methoden des verstärkenden Lernens

Der Agent sucht im Kontext von **wertbasierenden Methoden** die optimale Strategie π^*, welche zu allen Zuständen S die jeweilige Aktion $A(S)$ zuordnet, sodass die kummulierte Belohnungswahrscheinlichkeit $R(r_{t+1}|S_t, A_t)$ über alle Zeitschritte t maximal ist.[38] Neben der unmittelbaren Belohnung aus einem Zustands-Aktions Paar müssen auch die zukünftigen Belohnungen der möglichen Folgezustände betrachtet werden, wofür das Konzept der Wertigkeit eingeführt wird.[39] Über eine Zustands- oder Aktionswertigkeitsfunktion, oftmals als Q-Funktion referenziert, wird eine Vorhersage über die zu erwartende abgezinste Belohnung berechnet.[40] Durch den Abzinsungsfaktor $\gamma \in [0, 1)$ wird der Einfluss zukünftiger Belohnungen nach ihrer zeitlichen Reihenfolge priorisiert.[41] Mit der Wertigkeitsfunktion kann evaluiert werden, welche Strategie langfristig am erfolgreichsten ist, da manche Aktionen trotz geringer anfänglicher Belohnung

[32] Vgl. Wang/Hong 2020, S. 3
[33] Vgl. Wang/Hong 2020, S. 3
[34] Vgl. Li 2019, S. 5
[35] Vgl. Li 2019, S. 5
[36] Vgl. Cutler/Walsh/How 2014, S. 2
[37] Enthalten in: Canese u. a. 2021, S. 6
[38] Vgl. Reda/Tao, T./van de Panne 2020, S. 2
[39] Vgl. Wang/Hong 2020, S. 3
[40] Vgl. Li 2019, S. 5
[41] Vgl Li 2019, S. 5

einen hohen Wert aufweisen können, wenn aus einem zukünftigen Zustand eine hohe Belohnung zu erwarten ist.[42] Die Wertigkeitsfunktion und die daraus berechneten Wertigkeiten von Aktionen oder Zuständen werden über alle Zeitschritte neu geschätzt und stellen mit die wichtigsten Komponenten in Algorithmen des verstärkenden Lernens dar.[43] Methoden basierend auf diesem Wertigkeitswert lernen eine Schätzfunktion der Wertigkeit für alle Zustände ($V_\pi(s) \forall S$) und alle Zustandsaktions-Paare ($Q_\pi(s_t, a_t) \forall s, a \in (S, A)$) durch das Aktualisieren der folgenden Funktionen:[44]

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (1)$$

$$V(s_t) = \max_a Q(s_t, a|\omega) \quad (2)$$

Aus der geschätzten Wertigkeit jedes Zustands-Aktions Paares kann die optimale Strategie $\pi^*(s)$ durch $\arg\max_a Q(s, a)$ bestimmt werden.[45]

Methoden, welche die Strategie durch direkte Parametrisierung, anstelle einer Bewertung aller Handlungsalternativen mittels Wertigkeitsfunktion optimieren, werden als **strategiebasierend** bezeichnet.[46] Diese Methodik kann beim Trainieren deterministischer Strategien zu unerwarteten Aktionen führen, weshalb häufig das Optimieren einer Wahrscheinlichkeitsverteilung für alle Aktionen bevorzugt wird.[47] Als Subklasse der RL Methoden wird der statistische Gradientenabstieg verwendet, um die parametrisierte Strategie π_θ hinsichtlich der maximalen langfristigen kumulierten Belohnung zu optimieren.[48] Unter dem Parametervektor θ beschreibt die Strategie π_θ oder auch $\pi(a|s, \theta)$ die Wahrscheinlichkeit, Aktion a im Zustand s auszuwählen.[49] Zur Optimierung der Strategie wird die Funktion der kumulierten Belohnungen J nach dem Parameter der Gewichte wie folgt in Formel drei abgeleitet und der optimierte Parametervektor anhand Formel vier aktualisiert.[50]

$$\nabla_\theta J(\theta) \approx \tfrac{1}{N} \sum_{i=1}^{N} \left[(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t))(\sum_{t=1}^{T} r(s_t, a_t)) \right] \quad (3)$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \quad (4)$$

Zusammengefasst können die Formeln vier und fünf dabei so interpretiert werden, dass die logarithmierte Wahrscheinlichkeit Aktion a_t im Zustand s_t auszuwählen erhöht wird, wenn a_t in einer höheren kumulierten Belohnung resultiert.[51]

[42] Vgl. Sutton/Barto 2018, S. 6
[43] Vgl. Sutton/Barto 2018, S. 6f.
[44] Vgl. Zhang/Wu/Pineau 2018, S. 2
[45] Vgl. Zhang/Wu/Pineau 2018, S. 2
[46] Vgl. Zhang/Wu/Pineau 2018, S. 2
[47] Vgl. Ningombam 2022, S. 3
[48] Vgl. Ningombam 2022, S. 3
[49] Vgl. Sutton/Barto 2018, S. 321
[50] Vgl. Wang/Hong 2020, S. 6
[51] Vgl. Wang/Hong 2020, S. 6

Unter **Akteur-Kritiker Methoden** werden hybride wertebasierende und strategiebasierende Methoden verstanden, welche zugleich die Parameter der Strategie optimieren und eine Wertefunktion approximieren.[52] Die strategiebasierende Methodik mit der lernenden Strategie agiert dabei als Akteur, wohingegen die Wertefunktion, welche jeder Aktion und jedem Zustand einen Behlohnungswert zuweist, als Kritiker handelt.[53] Der Akteur wählt somit aus seiner Wahrscheinlichkeitsverteilung die auszuführende Aktionen aus, während der Kritiker diese anhand seiner Wertigkeit evaluiert.[54] Betrachtet man den Trainingsprozess von Akteur-Kritiker basierten Methoden, ist dieser wie folgt aufgebaut:[55]

1. Aktueller Zustand der Umgebung als Eingabe dem Akteur und Kritiker übergeben
2. Akteur liefert eine auszuführende Aktion basierend auf dem Umgebungszustand
3. Der Kritiker bekommt die Aktion als Eingabe und berechnet deren Wertigkeit mittels Q-Funktion
4. Durch die Wertigkeit einer Aktion kann der Akteur seine Strategie anpassen
5. Mit der neuen Strategie führt der Akteur die nächste Aktion im Folgezustand aus
6. Die Q-Funktion des Kritikers wird mit den neuen Informationen aus der erhaltenen Belohnung angepasst

2.2.2 Algorithmen des verstärkenden Lernens

Im vorherigen Kapitel sind die Methoden des verstärkenden Lernens klassifiziert und deren Unterschiede beschrieben worden. Anschließend wird in diesem Abschnitt auf eine Auswahl konkreter Algorithmen und Implementierungen, die verschiedenen Methoden sowie deren Funktionsweise eingegangen. Die Auswahl an Algorithmen beinhaltet die fundamentalen Algorithmen des verstärkenden Lernens Deep Q-Learning (DQL), Trust Region Policy Optimization (TRPO) und Asynchronous Advantage Actor-Critic (A3C).[56]

Mit der Entwicklung von DQL konnte erstmals ein Algorithmus entwickelt werden, welcher eine Reihe von Atari 2600 Spielen auf der Fähigkeitsebene eines professionellen Videospieltesters spielen konnte.[57] Als wertbasierte Methode wird für jeden Zustand die Wertigkeit berechnet, was unter DQL mittels eines neuronalen Netzes erreicht wird, welches die Q-Funktion approximiert.[58] Durch Auswählen der Aktion mit der höchsten Wertigkeit in jedem Zustand, kann die deterministische Strategie abgeleitet werden.[59] DQL adressiert das Instabilitätsproblem von

[52] Vgl. Zhang/Wu/Pineau 2018, S. 2f.
[53] Vgl. Sutton/Barto 2018, S. 321
[54] Vgl. Ningombam 2022, S. 3
[55] Vgl. Ningombam 2022, S. 4
[56] Vgl. Arulkumaran u. a. 2017, S. 1
[57] Vgl. Arulkumaran u. a. 2017, S. 6
[58] Vgl. Huang u. a. 2017, S. 4
[59] Vgl. Huang u. a. 2017, S. 4

Funktionsapproximation unter dem Einsatz von Erfahrungswiederholung und Zielnetzwerken.[60] Erfahrungswiederholung beschreibt die Technik, die Beobachtungen des Agenten, also in welchem Zustand welche Aktion zu welcher Belohnung und welchem Folgezustand führt, in jedem Zeitschritt zu speichern und zufällig anhand Übergangserfahrungen aus anderen Trainingsläufen die Gewichte der Wertigkeitsfunktion anzupassen.[61] Mittels Erfahrungswiederholung wird eine bessere Dateneffizienz zum einen durch die Wiederholung, und zum anderen durch das Auflösen von Korrelationen zwischen aufeinanderfolgenden Zuständen erzielt.[62] Das Zielnetzwerk beinhaltet die zunächst fixen Gewichte der Strategie, welche lediglich nach einer festen Anzahl an Schritten angepasst werden, um die Fluktuation der approximierten Q-Funktion auszugleichen.[63] Die Stärke von DQL liegt in der kompakten Repräsentation der Q-Funktion und der hochgradig dimensionierten Zustandsbeobachtungen durch neuronale Netze.[64]

Trust Region Policy Optimization (TRPO) ist ein strategiebasierender Gradientenalgorithmus zur effektiven Optimierung großer nicht linearer Strategien wie z. B. neuronale Netze.[65] Der Algorithmus weist dabei die zwei Varianten single-path, welche im modellfreien Kontext angewendet werden kann, und *vine* auf, welche sich nur in Simulationen eignet, da das System zu bestimmten Zuständen gespeichert wird.[66] Die Varianten unterscheiden sich im Schätzverfahren des Gradienten, wobei die *single-path* Methodik diesen anhand einer Aktions-Zustandskette der initialen Verteilung bestimmt, während unter *vine* eine Teilmenge verschiedener Aktion-Zustandspaare verwendet wird.[67] Die daraus entstehenden Anpassungen der Strategieparameter θ zwischen der alten und neuen Strategie werden mittels Kullback-Leibler-Divergenz bemessen und kontrolliert.[68] Eine Weiterentwicklung des TRPO Algorithmus ist Proximal Policy Optimization (PPO), durch dessen Verwendung die Implementierung erleichtert und die Datenprobenkomplexität verbessert wird.[69] PPO passt die Beschränkung der KL-Divergenz an, indem die Wahrscheinlichkeitsmasse der Optimierung außerhalb der gesetzten Hyperparametergrenzen bearbeitet oder die Größe der KL-Divergenz bestraft wird.[70] Durch diese Anpassung der KL-Divergenz Beschränkung ermöglicht der PPO Algorithmus die mehrfache Berechnungen des Gradienten anhand eines Datenobjektes.[71]

Zusätzlich zu den bisherigen Algorithmen kann durch jeweilige asynchrone Varianten, welche den Trainingsprozess des Agenten parallelisieren, die Stabilität des Trainings verbessert werden.[72] Einer dieser asynchronen Algorithmen im Bereich der Akteur-Kritiker Methodik stellt der asynchronous advantage actor-critic (A3C) Algorithmus dar, welcher anstelle von Erfahrungswiederholung

[60] Vgl. Arulkumaran u. a. 2017, S. 7
[61] Vgl. Mnih/Kavukcuoglu u. a. 2013, S. 4
[62] Vgl. Mnih/Kavukcuoglu u. a. 2013, S. 4f.
[63] Vgl. Arulkumaran u. a. 2017, S. 7
[64] Vgl. Arulkumaran u. a. 2017, S. 7
[65] Vgl. Schulman/Levine u. a. 2015, S. 1
[66] Vgl. Schulman/Levine u. a. 2015, S. 1
[67] Vgl. Schulman/Levine u. a. 2015, S. 4
[68] Vgl. Huang u. a. 2017, S. 4
[69] Vgl. Schulman/Wolski u. a. 2017, S. 1
[70] Vgl. Schulman/Wolski u. a. 2017, S. 3f.
[71] Vgl. Schulman/Wolski u. a. 2017, S. 4
[72] Vgl. Mnih/Badia u. a. 2016, S. 1

mehrere Agenten parallel in unterschiedlichen Umgebungsinstanzen trainiert.[73] A3C kombiniert die Berechnung der relativen Wertigkeit einer Aktion, anstelle der absoluten Wertigkeit durch die Q-Funktion, mit der Akteur-Kritiker Struktur und lässt sich auf einzelnen sowie verteilten Systemen einsetzen.[74] Wird mit dem Algorithmus lediglich ein Agent trainiert, wird dies häufig auch als advantage actor-critic (A2C) referenziert.[75] Neben A3C ist der soft actor-critic Algorithmus ein weiterer Algorithmus der Akteur-Kritiker Methodik, welcher die kumulierte Belohnung, aber auch die Entropie maximiert.[76] Durch dieses Konzept der Maximierung der Belohnung und der Entropie ergeben sich die Vorteile eines stärker erkundenden Algorithmus, welcher gleichzeitig mehrere nahezu optimale Lösungen erfassen kann.[77] Zur Evaluation wird die auf der Belohnung und Entropie basierte Wertigkeit iterativ anhand des Bellman Operators nach Bellman 1966 bestimmt und anschließend die Strategie hinsichtlich der exponentiellen Q-Funktion angepasst.[78] Durch die abwechselnde Approximation der Q-Funktion des Kritikers sowie der Strategie des Akteurs, anstelle der Berechnung dieser bis zur Konvergenz, kann der soft actor-critic Algorithmus besonders mit großen kontinuierlichen Handlungsbereichen umgehen.[79]

2.2.3 Abgrenzung zu Multi-Agent Reinforcement Learning (MARL) Algorithmen

Innerhalb dieses Unterkapitels soll der beschriebene Aufbau von RL Algorithmen und deren Optimierungsproblem zu den von MARL Systemen abgegrenzt werden. Bei MARL Systemen wird anstatt eines Agenten eine Menge von Agenten eingesetzt, welche alle mit ihrer Umgebung interagieren, um den Weg der Zielerreichung zu lernen.[80] Dieser Ansatz dient dazu, die Vielzahl an Problemstellungen in der echten Welt, welche nicht vollständig durch einen einzelnen Agenten lösbar sind, zu bearbeiten.[81] Einsatzgebiete von MARL sind dabei unter anderem das Routing von Netzwerkpaketen, Wirtschaftsmodellierung oder zusammenhängende Robotersysteme.[82] Je nach Ziel und der demnach definierten Belohnungsfunktion können die Agenten auf die drei unterschiedlichen Arten vollständig kooperativ, vollständig kompetitiv und der Mischung aus beiden miteinander interagieren.[83] Aus den einzelnen Interaktion jedes Agenten mit derselben Umgebung ergibt sich der Unterschied, dass die Umgebungsdynamik aus der Kombination aller Aktionen der Agenten beeinflusst wird anstatt aus der Aktion des einzelnen Agenten.[84] Da dieser Effekt die Annahme der Stationarität von Markov Entscheidungsprozessen verletzt, bedarf die Umgebung einer anderen Representation.[85] Ein Konzept, welches hierfür häufig verwendet

[73] Vgl. Mnih/Badia u. a. 2016, S. 1
[74] Vgl. Arulkumaran u. a. 2017, S. 9
[75] Vgl. Arulkumaran u. a. 2017, S. 9
[76] Vgl. Haarnoja u. a. 2018, S. 1
[77] Vgl. Haarnoja u. a. 2018, S. 3
[78] Vgl. Haarnoja u. a. 2018, S. 4
[79] Vgl. Haarnoja u. a. 2018, S. 4
[80] Vgl. Wong u. a. 2022, S. 6
[81] Vgl. Canese u. a. 2021, S. 1
[82] Vgl. Canese u. a. 2021, S. 1
[83] Vgl. Canese u. a. 2021, S. 8f.
[84] Vgl. Wong u. a. 2022, S. 2
[85] Vgl. Wong u. a. 2022, S. 6

wird, ist das Markov Spiel, welches sich anders als der Entscheidungsprozess durch einen mehrdimensionalen Aktions- und Belohnungsraum aus der Kombination aller N Agenten auszeichnet.[86] Betrachtet man die Limitierungen von MARL, erkennt man aus den beschriebenen Punkten die Herausforderungen der nicht vorhandenen Stationarität und der Skalierbarkeit, welchen sich die Herausforderung der teilweisen Beobachtbarkeit der Umgebung anschließt.[87]

2.2.4 Limitierungen und Herausforderungen von RL

Trotz signifikanter Errungenschaften birgt der Einsatz der besprochenen RL Algorithmen weiterhin Limitierungen und Risiken für ungewolltes Verhalten.[88]

Eine der Herausforderungen zeigt sich bei der Representation der Agentenumwelt, da RL stark auf diesem Konzept basiert.[89] Daraus ergibt sich die Aufgabe, die Umwelt und deren Verhalten sowie die Wahrnehmung durch den Agenten realitätsgetreu und präzise zu gestalten.[90] Neben der Definition und Wahrnehmung des Umweltverhaltens, ist die Spezifikation des Ziels des Agenten ein ebenso kritischer Teil, da unerwartete Intentionen aus der Zielstellung abgeleitet werden könnten.[91] Zusätzlich teilen RL Algorithmen auch Herausforderungen aus anderen Gebieten des maschinellen Lernens wie Genauigkeit, Interpretierbarkeit und die im Rahmen dieser Arbeit untersuchte Robustheit von Modellen.[92]

Eine weitere Limitierung stellt der große Suchraum an Aktionen und das unbekannte Verhalten der Umgebung dar. Dies sorgt dafür, dass die Effizienz einzelner Daten häufig sehr gering ist und die Abwägung zwischen der Exploration neuer Strategie und der Optimierung bekannter Verhaltensmuster ein wichtiger Bestandteil ist.[93] Aufgrund der geringen Effizienz der Daten, aber des hohen Bedarfs an bewerteter Agentenerfahrung wird häufig auf simulierte Daten zurückgegriffen.[94] Simulierte Daten werden dabei häufig von möglichst hochqualitativen Simulationsumgebungen bereitgestellt, da zu dem hohen Bedarf der Methodik häufig Limitierungen in der Sammlung von Daten in der echten Welt bestehen.[95]

Aufgrund der Bedeutung der Simulationsumgebungen und dessen Handlungsräume für RL Algorithmen und deren Transfer in die echte Welt, werden in den beiden nachfolgenden Kapiteln imitierendes Lernen und Simulationen genauer betrachtet.

[86] Vgl. Canese u. a. 2021, S. 4
[87] Vgl. Canese u. a. 2021, S. 9ff.
[88] Vgl. Li 2019, S. 7
[89] Vgl. Sutton/Barto 2018, S. 8
[90] Vgl. Sutton/Barto 2018, S. 7
[91] Vgl. Li 2019, S. 7
[92] Vgl. Li 2019, S. 7
[93] Vgl. Li 2019, S. 7
[94] Vgl. Zhao/Queralta/Westerlund 2020, S. 7
[95] Vgl. Li 2019, S. 8

2.3 Imitierendes Lernen

Für Algorithmen des maschinellen Lernens wie z. B. RL, erweist die Skalierung zu höherdimensionalen Handlungsräumen ein großes Optimierungsproblem auf.[96] Hinzu kommen Schwächen in der Anwendung von verstärkendem Lernen wie Handlungsstrategien in unerfahrenen Umgebungszuständen.[97] Imitierendes Lernen (IL) kann hier als ein Vorschritt zur Optimierung von RL Modellen dienen, womit signifikante Belohnungsverbesserungen erzielt werden können.[98] Ein Anwendungsgebiet des imitierenden Lernens ist die Robotik, in welchem ein chirurgischer Roboter durch Teleoperation bereits erfolgreich chirurgische Eingriffe demonstrieren konnte.[99] Allgemeiner wird IL häufig eingesetzt, um menschliches Verhalten nachzuahmen und zu deren Aktionen zu beschleunigen.[100] Das Ziel von IL ist hierbei eine Zuordnung zu erlernen zwischen den Beobachtungen und Aktionen von Expertendemonstrationen.[101] Dadurch soll ein Roboter das ihm demonstrierte Verhalten reproduzieren und auch für unbekannte Szenarien generalisieren können.[102] Problemstellungen können somit auf das Demonstrieren und Anlernen der Fähigkeit beschränkt werden, ohne dass explizite Anweisungen oder Belohnungsfunktionen programmiert werden müssen.[103] IL stellt dabei einen passiven Ansatz dar, die Zielstrategie durch Beobachten der vollständigen Ausführungsabläufe zu erlernen.[104] Dabei wird eine Ersatz-Verlustfunktion optimiert, welche die Diskrepanz zwischen der gewählten Aktion der parametrisierten Strategie π_θ und der des Experten ψ misst.[105]

Der typische Einsatzprozess von IL beinhaltet das Akquirieren von Demonstrationsbeispielen, deren Kodierung als Zustands-Aktions Paare und das Optimieren einer Strategie.[106] Einzelne Prozessschritte können sich dabei je nach Anwendungsgebiet unterscheiden oder Nachfolgeprozesse, wie die Feinabstimmung mittels RL enthalten, was an Abbildung 3 deutlich wird.

Algorithmen des imitierenden Lernens unterscheiden sich in der Literatur nach den Begriffen *on-policy* und *off-policy*.[108] Dabei werden zum einen erlernte Strategien in der Umgebung ausgeführt und mittels Experten evaluiert, zum anderen werden lediglich die Expertendemonstrationen verwendet.[109] DAgger stellt ein *on-policy* Algorithmus dar, welcher zu jeder Iteration die Optimierung aller bisher beobachteten Zustands-Aktions Paare vornimmt.[110] Behavioral Cloning als

[96] Vgl. Hussein u. a. 2017, S. 3
[97] Vgl. Attia/Dayan 2018, S. 1
[98] Vgl. Hussein u. a. 2017, S. 4
[99] Vgl. Gao u. a. 2014
[100] Vgl. Attia/Dayan 2018, S. 1
[101] Vgl. Hussein u. a. 2017, S. 1
[102] Vgl. Fang u. a. 2019, S. 365
[103] Vgl. Hussein u. a. 2017, S. 1
[104] Vgl. Attia/Dayan 2018, S. 2
[105] Vgl. Balakrishna u. a. 2020, S. 2f.
[106] Vgl. Hussein u. a. 2017, S. 3
[107] Enthalten in: Hussein u. a. 2017, S. 4
[108] Vgl. Balakrishna u. a. 2020, S. 3
[109] Vgl. Balakrishna u. a. 2020, S. 3
[110] Vgl. Attia/Dayan 2018, S. 5

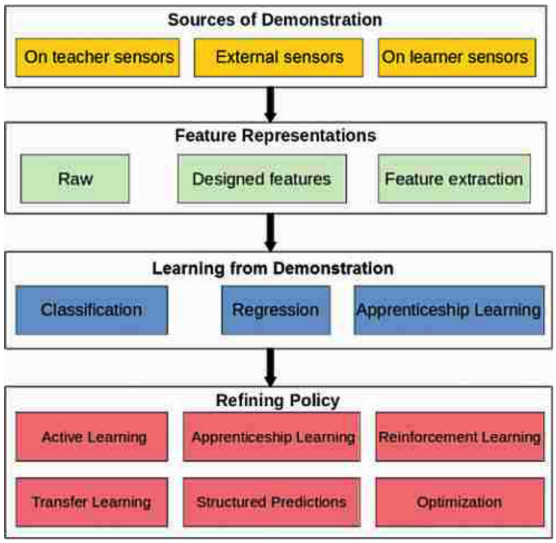


Abb. 3: Ablaufdiagramm des imitierenden Lernens[107]

off-policy Algorithmus erlernt hingegen die direkte Übersetzung der Zustands- und Umgebungsinformationen zu ihren Aktionen, ähnlich einem Label unter überwachtem Lernen.[111]

Insgesamt kann der Einsatz von IL unter anderem folgende Vorteile verzeichnen:[112]

- Verbesserung der Anpassungsfähigkeit hinsichtlich neuer Umgebungen
- Erhöhung der Lerneffizienz aufgrund schnellerer Übertragung von Wissen
- Kompatibilität mit anderen Algorithmen des maschinellen Lernens wie z. B. RL

Aufgrund der interdisziplinären Natur von IL ergeben sich jedoch wie folgt auch eine Reihe von Herausforderungen:[113]

- Der IL Prozess ist sowohl während der Sammlung von Demonstrationsdaten sowie während des Trainingsverlaufs anfällig für Rauschen und Fehler der Sensorik.
- Fähigkeiten, Aufbau und Freiheitsgrade des lernenden Modells und des Experten müssen bestmöglich übereinstimmen.
- Anwendungen lassen sich aufgrund ihres Echtzeitbezugs und Limitierungen der Rechenkapazitäten nur schwierig, hinsichtlich hoher Freiheitsgrade skalieren.

[111] Vgl. Fang u. a. 2019, S. 4
[112] Vgl. Fang u. a. 2019, S. 1
[113] Vgl. Hussein u. a. 2017, S. 4f.

2.4 Simulationsumgebungen für RL

Anders als im klassischen Bereich des maschinellen Lernens wie dem überwachten- und unüberwachten Lernen, werden beim verstärkenden Lernen viele der Testdatensätze nicht aus der echten Welt akquiriert.[114] Um entsprechend realistische Daten für das Training bereitzustellen, werden Simulationsumgebungen in Abhängigkeit von ihrer RL Anwendung ausgewählt.[115] Dennoch bleibt nahezu immer eine gewisse Diskrepanz zwischen der Dynamik in der Simulation und der Dynamik in der echten Welt.[116] Möglichkeiten, diese Diskrepanz zu minimieren, sind zum einen das Fehlverhalten von Sensoren einzubinden oder ein reales Signal mit der virtuellen Umgebung zu verknüpfen.[117] Dennoch lässt sich kaum garantieren, dass erlernte Strategien der Agenten sich auf nur leicht veränderte Umgebungen übertragen lassen.[118] Anders als in der Simulation von Flüssigkeiten und deren Dynamik, bedarf RL eine reaktive Umgebung, dessen Verhältnis der Simulationszeit zur echten Zeit mindestens eins oder darüber liegt.[119] RL kann von einem erhöhten Echtzeitfaktor profitieren, trotz der damit einhergehenden verringerten Präzision.[120]

2.4.1 Definitionen von Simulationsumgebungen

Ausgehend von der Literaturrecherche zeigt sich, dass in der Forschungsliteratur die allgemeine Definition von Simulationen kaum aufgegriffen wird. Eine mögliche Definition nach Anu 1997 wird wie folgt dargelegt:

Eine Simulation eines existierenden Systems stellt die Anwendung eines Modells dar, welches konfigurierbar zu experimentellen Zwecken das eigentliche System vertritt, um wirtschaftliche oder systematische Herausforderungen des existierenden Systems zu umgehen. Das Model wird in diesem Kontext definiert als Repräsentation des Aufbaus und der Verhaltensweise des existierenden Systems.

Innerhalb bestimmter Anwendungsgebiete, wie der Medizin und der Pflege, werden zusätzlich virtuelle Simulationen wie nachstehend definiert.

Unter virtuellen Simulationen versteht man eine digitale Lernumgebung, welche durch teilweiser Immersion eine wahrnehmbare Erfahrung bereitstellt.[121]

[114] Vgl. Zhang/Wu/Pineau 2018, S. 1
[115] Vgl. Körber u. a. 2021, S. 7
[116] Vgl. Bharadhwaj u. a. 2019, S. 1
[117] Vgl. Zhang/Wu/Pineau 2018, S. 1
[118] Vgl. Bharadhwaj u. a. 2019, S. 1
[119] Vgl. Körber u. a. 2021, S. 3
[120] Vgl. Körber u. a. 2021, S. 3
[121] Vgl. Foronda 2021, S. 1

2.4.2 Entwicklung von Simulationsumgebungen für RL Anwendungen

Im weiteren Teil dieses Kapitels wird aufbauend auf den zuvor angeführten Definitionen, die Entwicklung von Simulationen betrachtet. Allgemein lässt sich dieser Entwicklungsprozess in die folgenden Teilschritte gliedern:[122]

1. Identifikation der Herausforderungen im existierenden System und Ableitung von Anforderungen für die Simulation.
2. Zielgruppe, Funktionsrahmen und quantitative Bewertungskriterien der Simulation definieren.
3. Analyse des zu simulierenden Systemverhaltens durch Sammeln und Verarbeiten von realen Daten des existierenden Systems.
4. Entwicklung einer schematischen Darstellung des Modells und dessen Überführung in nutzbare Software.
5. Validierung des Modells durch bspw. den Vergleich mit dem existierenden System.
6. Dokumentierung des Modells, dessen Variablen, Metriken und getroffene Annahmen.

Die Entwicklung von Simulationen wurde in der Forschungsliteratur besonders durch den Fortschritt im Bereich des verstärkenden Lernens vorangetrieben, da der Vergleich von RL-Algorithmen zuverlässige Benchmarks in Form von Simulationsumgebungen benötigt.[123] Aus dieser Motivation wurde 2016 durch die *OpenAI* der *OpenAI Gym* Werkzeugkasten entwickelt, welcher eine Sammlung an Benchmarksimulationen mit einer einheitlichen Schnittstelle für RL Algorithmen enthält.[124] Seither wurde diese definierte Schnittstelle vielfach verwendet, um RL Umgebungen mit dem Ziel zu entwickeln, diese zu publizieren und dessen Wiederverwendung zu ermöglichen.[125] Die Entwicklung dieses Softwareframeworks wurde nach der Version 0.26.0 im Jahr 2022 durch ein neues Team unter dem Namen Gymnasium weitergeführt.[126] Die Schnittstelle ist definiert als Python Klasse *gym.Env*, von welcher weitere Klassen erben und die vorgeschriebenen Funktionen zum Zeitschritt und zum Zurücksetzen der Simulation implementieren.[127] Der Werkzeugkasten von OpenAI fokussiert sich auf einen „Episoden ähnlichen Rahmen“, in welchem der Agent durch zunächst zufälliges Auswählen von Interaktionen lernt.[128] Weitere Entwicklungsentscheidungen des OpenAI Gym Werkzeugkastens umfassen z. B. die bewusst fehlende Schnittstelle des Agenten, die strikte Versionierung der Umgebung oder die standardmäßige Simulationsüberwachung.[129]

[122] Vgl. Anu 1997, S. 8f.
[123] Vgl. Brockman u. a. 2016, S. 1
[124] Vgl. Brockman u. a. 2016, S. 1
[125] Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 4
[126] Vgl. GitHub 4/4/2023
[127] Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 4
[128] Vgl. Brockman u. a. 2016, S. 1
[129] Vgl. Brockman u. a. 2016, S. 2f.

Werden Lernumgebungen nach der Gym Schnittstelle oder nach eigener Definition für RL Anwendungen eingesetzt, kann sich deren Gestaltung unterschiedlich auf die Leistung der Anwendung auswirken.[130] Eine enge initiale Wahrscheinlichkeitsverteilung des Umgebungszustandes kann die Lerneffizienz erhöhen, wohingegen eine weite Wahrscheinlichkeitsverteilung positiv die Robustheit der erlernten Strategie beeinflusst.[131] Die Robustheit kann zusätzlich durch die Einbindung von Fehlverhalten in der Wahrnehmung der Umgebung beeinflusst werden, da auch in realen Szenarien ein Risiko für Fehlverhalten besteht.[132] Im Bereich der Robotik bzw. in der Simulation von Bewegungen, kann auch durch die Gestaltung des Aktionsraumes, basierend auf elektrischer Regelungstechnik mittels PID-Regler, anstatt basierend auf Drehmomenten ein effizienterer Lernprozess stattfinden.[133]

Neben den beschriebenen Eigenschaften von Umgebungen für verstärkendes Lernen unterliegen auch die verwendeten Simulationen bestimmten Merkmalen, welche in der Entwicklung zu berücksichtigen sind. Laut einer Umfrage nach Ivaldi/Padois/Nori 2014 sind diese wichtigsten Eigenschaften die Stabilität, Geschwindigkeit, Präzision, Genauigkeit, Bedienbarkeit und der Ressourcenverbrauch. Die Entwicklung des Modells, welches das existierende System ersetzt, sollte sich demnach möglichst positiv auf die beschriebenen Eigenschaften auswirken. Neben den beschriebenen leistungsbezogenen Merkmalen, sind die folgenden weiteren Kriterien mitunter die wichtigsten zur Auswahl einer Simulation:

Rank	Most important criteria
1	Simulation very close to reality
2	Open-source
3	Same code for both real and simulated robot
4	Light and fast
5	Customization
6	No interpenetration between bodies

Tab. 2: wichtigste Kriterien zur Auswahl von Simulatoren[134]

Aus Tabelle 2 lässt sich entnehmen, dass besonders die Nähe zur Realität ein wichtiges Auswahlkriterium ist. Im Kontext von verstärkendem Lernen im Robotik Bereich ist ein wichtiger Baustein die Physik-Engine zur Modellierung von Dynamiken.[135]

2.4.3 Aktuelle Physik-Engines und Simulationsanwendungen

Innerhalb dieses Abschnittes wird aufgrund der Bedeutung der Physik-Engine für den Grad der Simulationsrealität, eine Auswahl der aktuellen Physik-Engines und deren Simulationsanwendung betrachtet.

[130] Vgl. Reda/Tao, T./van de Panne 2020, S. 1
[131] Vgl. Reda/Tao, T./van de Panne 2020, S. 3
[132] Vgl. Yan Duan u. a. 2016, S. 2
[133] Vgl. Reda/Tao, T./van de Panne 2020, S. 7
[134] Enthalten in: Ivaldi/Padois/Nori 2014, s. 4
[135] Vgl. Ayala u. a. 2020, S. 2

Gazebo

Gazebo ist eine durch die Open Source Robotics Foundation entwickelte Simulationsanwendung, welche mehrere Physik-Engines unterstützt.[136] Mittels Gazebo lassen sich Interaktionen zwischen Robotern in Innen- und Außenbereichen unter realistischer Sensorik simulieren.[137] Die unterstützten Physik-Engines umfassen Bullet, Dynamic Animation and Robotics Toolkit (DART), Open Dynamics Engine (ODE) and Simbody.[138]

MuJoCo

MuJoCo stellt eine Physik-Engine für modellbasierte Steuerung dar, dessen Objekte durch C++ oder XML definiert und Gelenkzustände im Koordinatensystem beschriebenen werden.[139] Diese beschriebene Eigenschaft lässt sich auch aus dem Namen als Abkürzung für **Mu**lti-**Jo**int dynamics with **Co**ntact ableiten.[140] Die MuJoCo Anwendung ist lizensiert, was den Besitz einer Lizenz für die Installation oder die Virtualisierung innerhalb eines Containers voraussetzt.[141]

PyBullet

PyBullet basiert als Simulationssoftware auf der Bullet-Engine und fokussiert sich funktional auf die Anwendung von RL im Robotikbereich.[142] Die Bullet-Engine ist hingegen eine offene Softwarebibliothek, welche neben verstärkenden Lernen auch bei Computeranimationen angewendet werden kann.[143] Bie Handhabung von PyBullet profitiert von der ausgiebigen Dokumentation, der großen Entwicklergemeinschaft und der Unterstützung von verschiedenen Dateiformaten wie SDA, URDF und MJCF zur Einbindung von Objekten.[144]

2.5 Simulation der Steuerungsaufgabe von Quadrokoptern

Die Popularität von unbemannten Flugzeugen im letzten Jahrzehnt nahm besonders im Bereich der Quadrokopter zu, sodass durch die sinkenden Kosten von Sensorik und Minicomputern, zahlreiche zukunftsträchtige Ergebnisse und Anwendungen erforscht worden sind.[145] Daher befasst sich dieses Kapitel der theoretischen Betrachtung, dessen Eigenschaften und deren Simulation. Anwendungsgebiete beinhalten z. B. die Landwirtschaft, den Pakettransport oder die Überwachung von großflächiger Infrastruktur wie Stromnetze.[146] Eine Simulation von unbemannten

[136] Vgl. Ivaldi/Padois/Nori 2014, S. 7
[137] Vgl. Ayala u. a. 2020, S. 4
[138] Vgl. Körber u. a. 2021, S. 3
[139] Vgl. Todorov/Erez/Tassa 2012, S. 1
[140] Vgl. Todorov/Erez/Tassa 2012, S. 2
[141] Vgl. Körber u. a. 2021, S. 3
[142] Vgl. Körber u. a. 2021, S. 3
[143] Vgl. Ivaldi/Padois/Nori 2014, S. 7
[144] Vgl. Körber u. a. 2021, S. 6
[145] Vgl. Koch u. a. 2018, S. 1
[146] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 1

Flugzeugen stellt eine Flugumgebung und vielseitige Sensorik bereit und kann je nach Anwendung Effekte wie Wind, Wolken und Niederschlag einbeziehen.[147]

2.5.1 Flugdynamiken eines Quadrokopters

Die Simulation des Quadrokopters stellt eine Simulation eines Flugkörpers mit drei Rotations- und drei Translationsbewegungen und demnach insgesamt sechs verschiedenen Freiheitsgraden dar.[148] Ein Quadrokopter ist ein fester Körper mit vier befestigten Rotoren, welche sich ausschließlich in eine Richtung drehen und positiven Schub in die Z-Achse des Körpers ausüben können.[149] Die vier Rotoren werden als + oder X Konfiguration entweder direkt in Richtung der X- und Y-Achse (+) oder um 45° gedreht (X) an der Drohne befestigt.[150] Jede Bewegung der sechs verschiedenen Arten wird durch die unterschiedliche Ansteuerung, der im Fall des Quadrokopters, vier Rotoren getätigt. Aus den verschieden starken auftrieben resultieren die drei Rotationsbewegungen Rollen, Nicken und Gieren.[151]

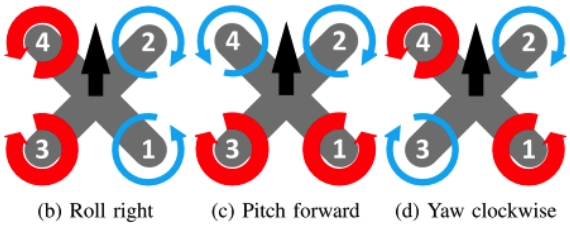


Abb. 4: Rotationsbewegungen eines Quadrokopters[152]

Rollen wird, wie aus Abbildung 4 hervorgeht, durch unterschiedlichen Auftrieb der zwei linken und rechten Rotatoren, das Nicken durch Unterschiede der vorderen und hinteren Rotatoren hervorgerufen. Das Gieren bzw. die Rotation um die Z-Achse wird durch die stärkere Rotation der sich im oder gegen den Uhrzeiger drehenden Rotoren bewerkstelligt.

Diese Rotationsbewegungen werden mathematisch als Matrix der Eulerschen Winkel repräsentiert, welche die Folge der einzelnen Drehungen entlang der X-, Y-, und Z-Achse enthält.[153] Das Zusammenspiel dieser Rotationsbewegungen, dargestellt anhand Formel vier, mit der durch die Rotatoren entlang der Z-Achse der Drone erzeugte Schubkraft und den Rollraten, beschrieben in Formel fünf, sorgt für die Bewegung der Drohne in Richtung der zukünftigen Koordinaten nach Formel sechs.[154]

[147] Vgl. Hentati u. a. 2018, S. 1496
[148] Vgl. Koch u. a. 2018, S. 2
[149] Vgl. Molchanov u. a. 2019, S. 3
[150] Vgl. Koch u. a. 2018, S. 2
[151] Vgl. Koch u. a. 2018, S. 2
[152] Enthalten in: Koch u. a. 2018, S. 2
[153] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 3
[154] Vgl. Deshpande/Minai/Kumar, M. 2021, S. 2

$$R = \begin{pmatrix} C_\psi C_p & S_\xi S_p C_\psi - S_\psi C_\xi & S_\xi S_\psi + S_p C_\xi C_\psi \\ S_\psi C_p & S_\xi S_\psi S_p + C_\xi C_\psi & -S_\xi C_\psi + S_\psi S_p C_\xi \\ -S_p & S_\xi C_p & C_\xi C_p \end{pmatrix} \quad (4)$$

In Formel vier wird der Rotationszustand der Drohne durch die Sinus- und Cosinuswinkel S_a und C_a repräsentiert, wobei für a die Art der Rotation also Rollen (ξ), Nicken (p) und Gieren (ψ) eingesetzt wird.[155]

$$I \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} l(F_1 + F_2 - F_3 - F_4) \\ l(-F_1 + F_2 + F_3 - F_4) \\ -M_1 + M_2 - M_3 + M_4 \end{pmatrix} - \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times I \begin{pmatrix} p \\ q \\ r \end{pmatrix} \quad (5)$$

Formel fünf inkludiert in der Matrix I die Trägheitsmomente um die X-, Y- und Z-Achsen in Abhängigkeit der Rollrate p, Nickrate q und Gierrate r.[156] Der Schub der in der Länge l vom Schwerpunkt entfernten Rotatoren wird mittels F_i und das Drehmoment mit $M_i, \forall i \in \{1, 2, 3, 4\}$ gekennzeichnet.[157]

$$m \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} + R \begin{pmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} F_i \end{pmatrix} \quad (6)$$

Weiterhin ist die Beschleunigung $\ddot{x}$, $\ddot{y}$ und $\ddot{z}$ in Richtung der drei Achsen durch die Masse m und die Gravitation g beeinflusst.[158]

2.5.2 Existierende Simulationen von Quadrokoptern

RotorS

Die Simulationsumgebung RotorS wurde auf Basis des Robot Operating Systems (ROS) entwickelt, um Programmtestzeiten zu verkürzen, Fehlersuche zu vereinfachen und Unfälle mit echten Mikroflugzeugen zu vermindern.[159] Dabei wurde die Simulation modular entworfen, sodass Komponenten wie Steuerung oder Zustandsschätzung austauschbar sind und das Hinzufügen von neuen Drohnen erleichtert wird.[160] Die Komponenten der Drohne stellen dabei Plug-ins der verwendeten Gazebo Physik-Engine dar, wodurch ein Mikroflugzeug aus den Teilen des Körpers, der Anzahl der Rotoren und Sensorik an fixen Position zusammengesetzt wird.[161] Mittels der standardmäßigen Sensorik können Informationen über die direkte und visuell gemessene Trägheit sowie über die Wegbestimmung erzielt werden.[162] Anstelle des Wegbestimmungssensors kann

[155] Vgl. Deshpande/Minai/Kumar, M. 2021, S. 2
[156] Vgl. Deshpande/Minai/Kumar, M. 2021, S. 2
[157] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 3
[158] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 3
[159] Vgl. Furrer u. a. 2016, S. 596
[160] Vgl. Furrer u. a. 2016, S. 595
[161] Vgl. Furrer u. a. 2016, S. 597
[162] Vgl. Furrer u. a. 2016, S. 597

auch eine Komponente zur Zustandsschätzung für hochfrequente Abfragen implementiert werden.[163] Die Steuerungskomponente wird durch eine einfache Schnittstelle einer geometrischen Steuerung bedient, welche Aktionen in Form von Rotationswinkeln, Höhen und Positionen entgegennimmt.[164]

CrazyS

CrazyS stellt eine Erweiterung der Simulation RotorS auf Basis desselben ROS, um die Modellierung des Nano-Quadrokopters Crazyflie samt ihrer Dynamik, ihres Kontrollsystems und ihrer Sensorik dar.[165] Mit der Modellierung der Nanodrohne wurde gleichzeitig ein Konzept zur Erweiterung der RotorS Fähigkeiten dargelegt sowie die Entwicklung von Software-in-the-Loop (SITL) als nahezu Echtzeitüberwachung vorangetrieben.[166]

AirSim

AirSim ist eine open-source Simulationsplattform, mit der das Ziel verfolgt wird, durch eine detailreiche Simulation die Entwicklung von RL und anderen Methoden des maschinellen Lernens voranzutreiben[167] Zur Modellierung der Simulationsphysik wird die Unreal Engine 4 aufgrund ihres hohen Grades an physikalischer und visueller Realität eingesetzt.[168] Der Aufbau der Simulation folgt einem modularen Entwurf, welcher unter anderem die einzelnen Komponenten Fahrzeug, Umgebung, Physik-Engine, Sensorik und Darstellungsschnittstelle beinhaltet.[169] Die Schnittstelle des Fahrzeugs erlaubt eine Steuerung über viele Betätigungselemente und deren Eigenschaftsparameter wie Masse, Trägheit, Widerstand oder Reibung.[170] Ein Fahrzeug ist dabei durch die Umgebungskomponente beeinflusst, welche physikalische Effekte wie Gravitation, Luftwiderstand, Luftdruck und magnetische Felder simuliert.[171] Die Umgebung wird für das Fahrzeug wahrnehmbar durch die Modellierung von Sensoren wie GPS, Beschleunigungsmesser, Gyroskop, Barometer und Magnetometer.[172]

gym-pybullet-drones

Eine weitere, nach der Gym Schnittstelle definierte Simulationsumgebung, ist gym-pybullet-drones.[173] Basierend auf der Bullet Physik-Engine ermöglicht die Simulation unter anderem das visuell basierte Training von mehreren Agenten mittels RL unter realistischer Modellierung von Kollisionen und aerodynamischen Effekten.[174] Die Wahl der Physik-Engine wurde aufgrund des CPU und GPU basierenden Renderings, des Kollisionsmanagements und der Kompatibilität mit

[163] Vgl. Furrer u. a. 2016, S. 598
[164] Vgl. Furrer u. a. 2016, S. 598
[165] Vgl. Silano/Iannelli 2019, S. 81
[166] Vgl. Silano/Iannelli 2019, S. 82
[167] Vgl. Shah u. a. 2017, S. 2
[168] Vgl. Shah u. a. 2017, S. 1
[169] Vgl. Shah u. a. 2017, S. 3
[170] Vgl. Shah u. a. 2017, S. 5
[171] Vgl. Shah u. a. 2017, S. 6
[172] Vgl. Shah u. a. 2017, S. 9
[173] Vgl. Panerati u. a. 2021, S. 1
[174] Vgl. Panerati u. a. 2021, S. 1

dem Unified Robot Description Format (URDF) getroffen.[175] Durch die Kompabilität mit dem URDF Format kann die standardmäßige Simulation des Drohnenmodells Bitcraze Crazyflie 2.x um weitere Nanoquadrokopter erweitert werden.[176]

2.6 Robustheit und Stabilität von Strategien des verstärkenden Lernens

Anders als in Simulationen lassen sich in der echten Welt häufig Unsicherheiten, Störeinflüsse und grundlegende Veränderungen der Umgebung wahrnehmen, für welche die Methoden des RL standardmäßig nicht robust genug sind.[177] Im nachfolgenden Kapitel wird daher genauer dargestellt, was unter der Robustheit von Algorithmen des verstärkenden Lernens verstanden wird, was Kenngrößen sind und in welchem Kontext diese erfasst werden können.

2.6.1 Definitionen von Robustheit und Stabilität

In der aktuellen Forschungsliteratur finden sich nur wenige Gemeinsamkeiten innerhalb der unterschiedlichen Definitionen von Stabilität und Robustheit.[178] Die Definition der Robustheit im Kontext von verstärkendem Lernen wird verschieden interpretiert, wie z. B. als Robustheit gegen Störeinflüsse, Beeinflussung der Belohnung, oder Umgebungsunterschiede.[179] Pullum 2022 definiert Stabilität und Robustheit im Kontext der Literaturanalyse wie folgt:

Stabilität ist eine Eigenschaft des lernenden Algorithmus, die sich auf dessen Leistungsvarianz bezieht und bei geringer Varianz auf ein stabiles Modell hinweist.[180]

Robustheit im Kontext von Software, referenziert eine Eigenschaft eines Systems, welches nicht nur ausschließlich unter normalen, sondern auch unter außergewöhnlichen Bedingungen, die die Annahmen des Entwicklers übersteigen, gut funktioniert.[181]

Moos u. a. 2022 beschreibt die Robustheit in seiner Literaturanalyse als Fähigkeit mit Variationen und Unsicherheiten in der Umgebung umgehen zu können, wobei Unsicherheiten häufig variierende physische Parameter darstellen.[182]

[175] Vgl. Panerati u. a. 2021, S. 3
[176] Vgl. Panerati u. a. 2021, S. 3
[177] Vgl. Moos u. a. 2022, S. 1
[178] Vgl. Pullum 2022, S. 5
[179] Vgl. Liu u. a. 2023, S. 2
[180] Vgl. Pullum 2022, S. 5
[181] Vgl. Pullum 2022, S. 5
[182] Vgl. Moos u. a. 2022, S. 1

2.6.2 Metriken der Robustheit

Werden Algorithmen und der Erfolg von Veränderungen dieser Experimente betrachtet, kommt es oftmals zu einem Vergleich zwischen Leistung und Stabilität, womit die Belohnung die einzige Kenngröße bildet.[183] Wird diese Metrik im Kontext unterschiedlicher Umgebungen überprüft, kann allerdings so auch die Robustheit von RL Algorithmen betrachtet werden.[184] Neben der Betrachtung der Belohnung lassen sich auch weitere quantitative sowie qualitative Kenngrößen untersuchen, welche die Robustheit und Stabilität eines Algorithmus innerhalb der Umgebung widerspiegeln.[185] In der Literaturanalyse nach Pullum 2022 werden quantitative Metriken zusätzlich nach internen Kenngrößen, welche den Trainingsprozess beschreiben und externen Kenngrößen, welche die Modellqualität repräsentieren, klassifiziert sowie folgende Tabellen drei und vier zu dessen Metriken angeführt.[186]

Internal Quantitative Metric	**Behavior**	**Total citations**
Reward or Score – magnitude, mean/ variance, variation in average reward, time to threshold, episode duration	Stability, Robustness	75
Policy entropy	Stability	2
Variations in control strategy approximation weights	Stability, Robustness	2
Convergence rate	Stability	2
Lyapunov stability criteria calculated	Stability	1
Policy weight	Robustness	1
Regret	Robustness	1
Wasserstein function bounds calculated	Robustness	1
		85

Tab. 3: Interne quantitative Metriken, dessen gemessenes Verhalten und ihre Häufigkeit[187]

2.6.3 Experimenteller Rahmen zur Messung der Robustheit

Die Tabellen drei und vier zeigen auf, dass trotz der Existenz weiterer Metriken, die Belohnung, deren Durchschnitt, Varianz und Entwicklung am häufigsten eingesetzt werden, um die Robustheit von RL Algorithmen zu messen. Dabei werden Experimente unter festgelegten oder optimierten Hyperparametern, in mehreren Simulationsumgebungen durchgeführt, um aus dem Vergleich derselben Strategie in unterschiedlichen Umfeldern Rückschlüsse auf die Robustheit zu

[183] Vgl. Yan Duan u. a. 2016, S. 6
[184] Vgl. Pinto u. a. 2017, S. 6
[185] Vgl. Pullum 2022, S. 15
[186] Vgl. Pullum 2022, S. 16
[187] Ähnlich enthalten in: Pullum 2022, S.17
[188] Ähnlich enthalten in: Pullum 2022, S.19

External Quantitative Metric	Behavior	Total citations
Deviations/variation in other (than precision, accuracy and recall) performance-related metrics	Stability, Robustness, Resilience	39
Error and failure rates/success rate	Stability, Robustness	28
Performance of tracking/trajectories estimation error; mean absolute deviation, mean square error, mean absolute percentage error, margins and magnitude of correlation coefficient	Stability, Robustness	23
Network-related timing/delay, path and link metrics, connectivity, delivery ratio, routing loops, path optimality, visitation distribution, structural Hamming distance, Small base station-serving ratio, sum-rate and 5th percentile rate	Stability, Robustness	15
Mean/average and variation inaccuracy, precision and recall, area under the receiver operating characteristic (ROC) curve (AUC)	Stability, Robustness, Resilience	12
Variance of the estimation of loss, regret	Robustness	5
		122

Tab. 4: Auszug der externen quantitativen Metriken, dessen gemessenes Verhalten und ihre Häufigkeit[188]

ziehen.[189] Unterschiede in den Umgebungen können z.B. durch voneinander abweichende Reibwerte während des Trainingsprozesses und der Testphase realisiert werden.[190] Ein weiterer Ansatz kann die Sim-to-Sim Verifikation sein, bei der die optimierten Strategien in einer nicht während des Trainings verwendeten Simulationen untersucht werden.[191]

2.7 Gegnerisches verstärkendes Lernen

Methoden des gegnerischen verstärkenden Lernens verfolgen das Ziel, Strategien zu lernen, die robuster gegenüber Risiken wie beeinflusste Wahrnehmung, unbekannte Situationen oder ansteigende Umgebungskomplexität agieren.[192] Die Robustheit gegenüber fehlerhafter Umgebungsbetrachtung kann durch Störung des Wahrnehmungszustands des trainierenden Agenten erzielt

[189] Vgl. Pinto u. a. 2017, S. 5
[190] Vgl. Pinto u. a. 2017, S. 6
[191] Vgl. Molchanov u. a. 2019, S. 5
[192] Vgl. Schott/Hajri/Lamprier 2022, S. 2

werden.[193] Das Ziel ist es, durch gegnerische Aktionen eine veränderte Umgebungswahrnehmnung herzustellen, auf deren Basis sich der lernende Agent verbessert.[194]

Um die lernende Strategie besser auf unbekannte Situationen und steigende Komplexität vorbereiten zu können, werden destabilisierende Kräfte in der Dynamik eingeführt.[195] Anders als beim ersten Ansatz werden dafür nicht nur die Wahrnehmung des lernenden Agenten beeinflusst, sondern direkte Einflüsse auf die Umgebung ausgeübt.[196] Hierbei wird der gegnerische Agent dafür belohnt, mittels Kräfteeinfluss die Umgebungsdynamik zu verändern, sodass der lernende Agent an seiner Aufgabe scheitert.[197] Dazu kann ein zusätzlicher Agent mit z. B. gleichem Aktionsraum den gemeinsamen Umgebungszustand beeinträchtigen.[198] Pan u. a. 2021 zeigt eine solche Anwendung im Rahmen der Kontrolle eines Stromnetzes, bei welcher ein gegnerischer Agent für das Trennen von Verbindungen im Netz belohnt wird.[199] Zum Generieren von Störeinflüssen auf die Umgebungsdynamik kann das *Robust Adversarial Reinforcement Learning* (RARL) Framework verwendet werden.[200] Dabei wird der gegnerische Agent selbst durch RL dahingehend optimiert, die möglichst effektivsten Destabilisierungsmaßnahmen zu finden.[201] Formal dargestellt folgt dieses gegnerische Spiel der in Formel sieben angeführten Minmax Optimierung.[202]

$$R^{1*} = \min_{\nu}\max_{\mu} E_{s_0\sim p, a^1\sim\mu(s), a^2\sim\nu(s)}[\sum_{t=0}^{T-1} r^1(s, a^1, a^2)] \quad (7)$$

Zu jedem Zeitschritt t in der gegnerischen Simulation wird von beiden Spielern eine Aktion $a_t^N \sim \mu(s_t) \forall N \in \{1, 2\}$ nach der Wahrnehmung des Umgebungszustands s ausgeübt, was zum Erhalt der Belohnung $r_t^1 = r_t$ und $r_t^2 = -r_t$ führt.[203] Das Training erfolgt aus der abwechselnden Optimierung einer der beiden Strategien bis zu deren Konvergenz, während die jeweils andere nicht verändert wird.[204]

Im Kontext der Steuerung von Quadrokoptern greifen Zhai u. a. 2022 das *Robust Adaptive Ensemble Adversarial Reinforcement Learning Framework (RAEARL)* auf. Unter dessen Einsatz wird der Trainingsprozess des normalen und gegnerischen Agenten zu Gunsten der Kontinuität und Stabilität getrennt, und das System mit einem PID-Regler erweitert, um die Stärke des Gegners über den Trainingsverlauf anzupassen.[205] Das getrennte Training des normalen und gegnerischen Agenten verwendet kopierte Agenten des jeweiligen Gegenparts, welche zwar dieselben neuronalen Netzstrukturen und initialen Parameter besitzen, jedoch schwächer sind, da deren Lernfortschritt um einen Zeitschritt verschoben ist.[206]

[193] Vgl. Schott/Hajri/Lamprier 2022, S. 2
[194] Vgl. Schott/Hajri/Lamprier 2022, S. 3
[195] Vgl. Pinto u. a. 2017, S. 1
[196] Vgl. Schott/Hajri/Lamprier 2022, S. 2
[197] Vgl. Pinto u. a. 2017, S. 2
[198] Vgl. Pinto u. a. 2017, S. 2
[199] Vgl. Pan u. a. 2021, S. 2
[200] Vgl. Schott/Hajri/Lamprier 2022, S. 2
[201] Vgl. Pinto u. a. 2017, S. 1
[202] Vgl. Pinto u. a. 2017, S. 3
[203] Vgl. Pinto u. a. 2017, S. 3f.
[204] Vgl. Pinto u. a. 2017, S. 4
[205] Vgl. Zhai u. a. 2022, S. 2
[206] Vgl. Zhai u. a. 2022, S. 2f.
[207] Enthalten in: Zhai u. a. 2022, S. 3

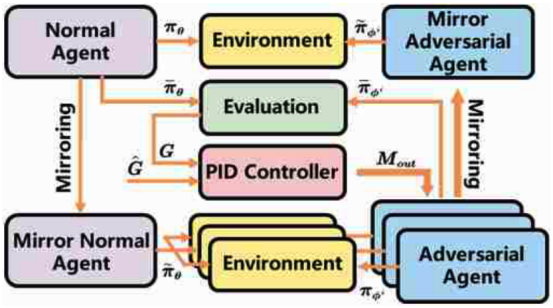


Abb. 5: Aufbau des RAEARL Frameworks[207]

Abbildung 5 zeigt den getrennten Aufbau des RAEARL Frameworks, in dem die Strategien der originalen Agenten mit π_θ bzw. π_{ϕ^i} und jene kopierte Strategien mit $\widetilde{\pi}_\theta$ bzw. $\widetilde{\pi}_{\phi^i}$ notiert sind.[208] Zur Evaluation jeder Epoche und der Bestimmung der kumulierten Belohnung G werden die deterministischen Strategien $\bar{\pi}_\theta$ bzw. $\bar{\pi}_{\phi^i}$ einbezogen.[209]

2.8 Domain Randomization

Domain Randomization (DR) ist eine weitere Methodik, die Realitätslücke zwischen der Simulation und der echten Welt zu schließen, indem die Simulation während des Trainingsprozesses vielfach variiert wird.[210] Im Gegensatz zur Systemidentifikation als Prozess, der versucht die Parameter bestmöglich an ein physisches System anzupassen, ermöglicht DR einen geringeren Zeitaufwand und geringere Fehleranfälligkeit.[211] Teilweise kann durch diesen Ansatz die Diskrepanz so weit geschlossen werden, dass eine Datensammlung in der echten Welt nicht erforderlich ist.[212] Anhand von Komponenten der Simulationsumgebung lässt sich DR in visuelle und dynamische Randomisierung unterteilen.[213] Weitere Methoden die Realitätslücke zu verkleinern sind progressive Netzwerke, inverse Dynamikmodelle und bayessche Methoden.[214]

2.8.1 Das Prinzip hinter der Randomisierung von Simulationsumgebungen

DR verfolgt den Ansatz eine hohe Anzahl an Simulationsumgebungen mit randomisierten Eigenschaften zu erzeugen, sodass die Strategie dahingehend optimiert werden, in allen Umgebungen ihr Ziel bestmöglich zu erfüllen ohne sich einer Umgebung überanzupassen.[215] Durch das Aussetzen des Agenten gegenüber einer Vielzahl von Umgebungen, wird unter der Erwartung einer

[208] Vgl. Zhai u. a. 2022, S. 3
[209] Vgl. Zhai u. a. 2022, S.3
[210] Vgl. Bharadhwaj u. a. 2019, S. 3
[211] Vgl. Tobin u. a. 2017, S. 1
[212] Vgl. Molchanov u. a. 2019, S. 2
[213] Vgl. Zhao/Queralta/Westerlund 2020, S. 5
[214] Vgl. Chen u. a. 2021, S. 2
[215] Vgl. Hsu u. a. 2023, S. 1

guten Leistung das Ziel verfolgt, den Agenten direkt in die physische Welt übertragen zu können.[216] Diese Erwartung resultiert aus der Hypothese, dass unter genug Variabilität die echte Welt lediglich eine bereits trainierte Variation darstellt.[217] Dieses Prinzip kann auch der Abbildung 6 entnommen werden, worin das physische System als Teil der Randomisierung der Simulation gezeigt wird.

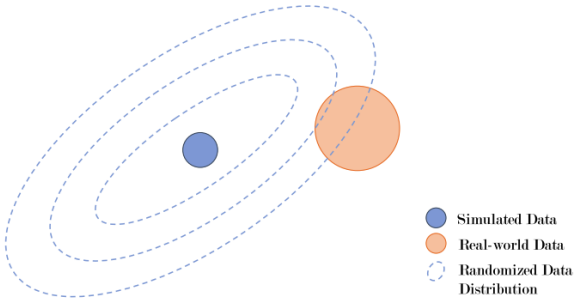


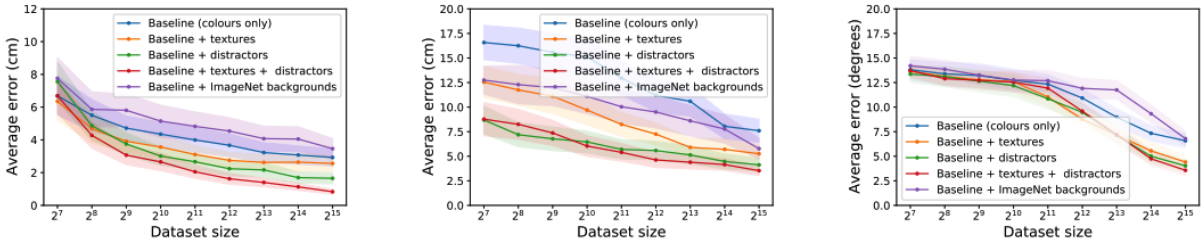
Abb. 6: Intuition hinter dem Paradigma von DR[218]

2.8.2 Anwendung von Randomisierung der Simulationsumgebung

Die Anwendung von DR beginnt nach der Erstellung einer möglichst realen Simulation, um sicherzustellen, dass spätere Variation die wirkliche Welt abbilden können.[219] Anschließend kann die Randomisierung verschiedener Aspekte den Agenten darin unterstützen, die Strategie, welche z. B. als rekurrentes neuronales Netz dargestellt werden kann, so zu optimieren, dass diese für die Wirklichkeit gut genug generalisiert.[220] Aspekte, welche im Rahmen von dynamischer DR randomisiert werden können, sind z. B. die Masse von Objekten, die Dauer eines Zeitschrittes oder die Schubkoeffizienten von Rotatoren.[221] Verwendet man visuelle Randomisierung, so sind bspw. Position, Textur, Ausrichtung und Lichteinwirkung jedes Objektes sowie Kamerawinkel und Zufallsrauschen Eigenschaften, die während des Trainingsprozesses verändert werden können.[222] Alghonaim/Johns 2021 untersuchten dazu detailliert den Einfluss der Randomisierung von Hintergrundfarben und -texturen sowie Ablenkungsobjekte auf die Modellleistung und erhielten folgende Ergebnisse.

Aus der Untersuchung in Abbildung 7 wird deutlich, dass die Fehleranfälligkeit der Erkennung der X-, Y-Position, der Z-Position und der Orientierung, durch das Hinzufügen von Texturen und Ablenkungsobjekten mitunter am besten verringert werden kann.

[216] Vgl. Chen u. a. 2021, S. 2
[217] Vgl. Tobin u. a. 2017, S. 1
[218] Enthalten in: Zhao/Queralta/Westerlund 2020, S. 6
[219] Vgl. Chen u. a. 2021, S. 4
[220] Vgl. Chen u. a. 2021, S. 4
[221] Vgl. Molchanov u. a. 2019, S. 4
[222] Vgl. Tobin u. a. 2017, S. 3
[223] Enthalten in: Alghonaim/Johns 2021, S. 6



(a) xy-position (b) z-position (c) Orientation

Abb. 7: Einfluss der Randomisierung verschiedener Effekte auf die Fehlerrate der Objekterkennung[223]

2.8.3 Errungenschaften unter Einsatz von Domain Randomization

Mit dem Einsatz von DR ist es bereits nach aktuellem Stand der Forschung und Praxis ermöglicht worden, z. B. Strategien ohne reale Bilder in Simulationen ausschließlich mittelmäßiger Qualität zu trainieren und für reale Drohnenflüge im Innenbereich einzusetzen.[224] Weiterhin konnte erreicht werden, ausschließlich aus dem Training in Simulationen eine Objekterkennung und Lokalisation zu einer Genauigkeit von 1,5 cm zu realisieren.[225] Neben der Anwendung von DR wird in der aktuellen Literatur sehr stark thematisiert, welche Randomisierung positive Effekte erzielen kann, jedoch wird wenig zu deren Erklärungen und Funktionsweise detailliert betrachtet.[226] Dies erschwert die Entwicklung effizienter Simulationen und die Bestimmung der Zufallsverteilungen zur Randomisierung.[227]

[224] Vgl. Sadeghi/Levine 2016, S. 1
[225] Vgl. Tobin u. a. 2017, S. 1
[226] Vgl. Zhao/Queralta/Westerlund 2020, S. 6
[227] Vgl. Zhao/Queralta/Westerlund 2020, S. 6

3 Durchführung des Laborexperiments

Anschließend an die Diskussion des aktuellen Stands der Forschung und der Praxis wird im Rahmen dieses Kapitels die Forschungsmethodik erläutert, um die folgende Forschungsfrage zu beantworten:

Inwiefern kann durch den Einsatz eines mittels RL trainierten Gegenspielers die Robustheit der gelernten Strategie verbessert werden?

In der ersten Sektion dieses Kapitels wird auf den Aufbau des Laborexperiments eingegangen. Dabei werden grundlegende Trainingsdaten sowie zu untersuchende Hypothesen, ausgeübte Einflüsse und erfasste Metriken beschrieben. Anschließend werden in der nächsten Sektion daraus resultierende Anforderungen für die Entwicklung der Simulation und der Testumgebung abgeleitet. Die dritte Sektion beinhaltet die Umsetzung der beschriebenen Anforderungen und erläutert die endgültige Implementierung der Simulation, der Testdatenerhebung und deren Auswertung.

3.1 Erläuterung der Forschungsmethodik

Innerhalb des Laborexperiments soll die Beziehung zwischen dem Trainingsszenario als Ursache und der Leistungsrobustheit als Wirkung betrachtet werden. Dabei sind zunächst Strategien durch verstärkendes Lernen unter unterschiedlichen Szenarien zu optimieren. Anschließend werden die trainierten Policies in einer veränderten Simulation als Testszenario ausgeübt, und währenddessen verschiedene Metriken der Strategieleistung betrachtet. Ziel ist es, die Leistungsdiskrepanzen zwischen den Strategien des Trainings mit deterministischen und mit optimiertem Gegenspieler zu betrachten. Kein Teil der umzusetzenden Forschungsmethodik ist die Anwendung der simulationsbasierten Strategien zur Steuerung von Quadrokoptern in der echten Welt. Weiterhin wird nicht die Ergebnisabhängigkeit zu Faktoren wie der Wahl des Simulationsframeworks, der Physik-Engine oder des Abstraktionsniveaus untersucht. Die Auswahl jener Aspekte wird unter den Gesichtspunkten der Softwarearchitektur getroffen, deren Anforderungen und Implementierung in nachfolgenden Sektionen behandelt wird.

3.1.1 Beschreibung der Simulationsumgebung

Beginnend mit dem Training der Policies durch verstärkendes Lernen, wird hierfür anders als bei dem überwachtem und unüberwachtem Lernen kein unmittelbar vorliegender Datensatz benötigt. Anstelle dessen basiert das Training auf der vollständigen oder teilweisen Wahrnehmung einer Lernumgebung, welche den lernenden Agenten für ausgeführte Aktionen positiv oder negativ belohnt. Wie auch in der Einleitung erwähnt, wird im Kontext dieser Arbeit die Simulation von Quadrokoptern dafür verwendet, die Lernumgebung und damit die Trainingsdaten für die

Algorithmen des RL zur Verfügung zu stellen. Die Simulation von Quadrokoptern stellt ein hochdynamisches Anwendungsgebiet dar, bei dem von einer hohen Diskrepanz zwischen der echten und simulierten Welt ausgegangen werden kann. Die Simulationsumgebung stellt grundsätzlich ein Szenario dar, in welchem zwei verschiedene Drohnen kompetitiv gegeneinander agieren. Das Ziel einer Drohne ist es, zu einem festgelegten Punkt hinter der zweiten Drohne zu gelangen, währenddessen die zweite Drohne versucht die angreifende Drohne auf deren Weg abzufangen. Um dies zu erreichen geht die verteidigende Drohne eine bewusste Kollision mit der anzugreifenden Drohne ein. Beide Drohnen werden unter der Einschränkung ihrer Nähe zum Zielpunkt zufällig in einem drei dimensionalem Raum initialisiert, welcher nur horizontal einseitig beschränkt wird. Durch die einseitige Beschränkung des Flugraums wird der natürliche Untergrund dargestellt. Zu Beginn soll die verteidigende Drohne näher am Zielpunkt als die angreifende Drohne starten, sodass stets ein Abfangen möglich ist. Während der Simulation wird die angreifende Drohne sich entweder regelbasiert auf einer geraden Linie oder sich durch optimierte Aktionen zum Zielpunkt bewegen. Die Bewegungsaktionen der verteidigenden Drohne werden durch ein sich optimierendes neuronales Netz bestimmt. Ist das Ziel einer der beiden Drohnen erreicht oder besteht Kontakt mit dem Untergrund, so wird die Simulation beendet.

3.1.2 Erläuterung der Trainingsszenarien

Durch die erläuterte Simulation, werden für das Laborexperiment Strategien in vier unterschiedlichen Szenarien mit einem Algorithmus des verstärkenden Lernens optimiert. Jede der Strategien wird dabei in einem der nachfolgenden Simulationsszenarien optimiert, dessen Auswahl die unabhängige Variable des Laborexperiments darstellt.

1. Das erste Szenario beinhaltet das Training der zu verteidigenden Drohne gegen eine angreifende Drohne, die eine deterministische regelbasierte Strategie ausführt.

2. Im zweiten Szenario wird eine deterministische Strategie für die anzugreifende Drohne mittels RL optimiert, während die verteidigende Drohne anhand einer regelbasierten Strategie agiert.

3. Anschließend enthält das dritte Szenario das Training einer Verteidigungsstrategie im kompetitiven Spiel mit einer anzugreifenden Drohne, die entsprechend der zuvor optimierten Strategie agiert.

4. Zum Abschluss wird das erste Szenario mit regelbasiertem Gegenspieler unter DR, also unter der Randomisierung dynamischer Parameter, als viertes Szenario wiederholt. Zur Randomisierung der Trainingsdomäne wird ein konstanter Windeffekt simuliert, dessen Richtung und Stärke zu jeder Trainingsepisode variiert.

3.1.3 Erläuterung des Testszenarios

Nach den Trainingsphasen werden einzelne Policies in mehreren Episoden, eines vom Training abweichenden Testszenarios, ausgeführt und deren Leistungsverhalten gemessen. Eine Episode entspricht dabei einer Simulation des Testszenarios bis zu ihrem erfolgreichen oder nicht erfolgreichem Ende durch Drohnen-, Ziel- oder Bodenkontakt. Zusätzlich zu den Trainingsszenarien ist neben den initialen Startpositionen der Drohnen auch der von der Angreiferdrohne zu erreichende Zielpunkt zufällig zu verändern. Neben der zusätzlichen Randomisierung des Zielpunktes wird ein im Vergleich zu den Trainingszenarien 1 und 4 verbesserter regelbasierter Gegenspieler eingesetzt.

3.1.4 Messung der Robustheit von RL Policies

Die in den Testszenarien betrachteten abhängigen Variablen sind die erzielte kumulierte Belohnung, deren Varianz sowie die Anzahl an Misserfolgen. Die kumulierte Belohnung und dessen Varianz stellen wie bereits beschrieben in der Forschung wichtige Kenngrößen dar, um die Robustheit von RL Policies zu bestimmen. Als Robustheit wird im Rahmen dieser Arbeit die Signifikanz einer Leistungsdiskrepanz von Modellen zwischen Trainings- und Testszenarien und die demnach fehlende Stabilität definiert. Die Metrik der Anzahl von unbeabsichtigten Misserfolgen spiegelt im behandelten Anwendungsfall das Fehlschlagen der trainierten Strategie wider die gegnerische Drohne nicht abfangen zu können. Mit der Auswahl der Metriken werden die Strategieeigenschaften der maximalen Strategieleistung, der Strategiestabilität über mehrere Episoden und der Strategieerfolgszuverlässigkeit deutlich. Aus der Abweichung zwischen dem Leistungsverhalten während des Trainings und während des Tests soll so die Robustheit von RL Policies erkennbar und messbar gestaltet werden. Der Fokus liegt dabei auf der Messung der Robustheit von den optimierten Strategien aus Szenario eins, drei und vier. Aus dem Vergleich des Leistungsverhaltens zwischen Strategie eins und drei kann eine mögliche Verbesserung der Robustheit durch das Training mit einem Gegenspieler, der entsprechend einer ebenfalls mit RL trainierten Strategie agiert, abgeleitet werden. Anschließend kann der erzielte Effekt mit der Abweichung zwischen eins und vier verglichen, und so im Vergleich zu aktuell verwendeten Methoden zur Erhöhung der Robustheit bewertet werden.

3.1.5 Auswertung mittels statistischer Tests

Durch den Vergleich einzelner Strategien anhand dieser Vorgehensweise werden im Laborexperiment die zu Beginn der Arbeit aufgestellten Hypothesen auf ihre Gültigkeit untersucht. Die nachfolgenden, bereits zu Beginn angeführten Hypothesen beinhalten, wie im vorherigen Kapitel diskutiert, verschiedene Aspekte der Robustheit von RL Strategien, welche in späteren Abschnitten einzeln ausgewertet werden.

1. *Die im Testszenario erzielte kumulierte Belohnung ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig höher als die Policy aus dem Training mit regelbasiertem Gegenspieler.*
2. *Die Varianz der im Testszenario erzielten Belohnung ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig geringer als die Policy aus dem Training mit regelbasiertem Gegenspieler.*
3. *Die im Testszenario erreichte Anzahl von unbeabsichtigten Misserfolgen ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig geringer als die Policy aus dem Training mit regelbasiertem Gegenspieler.*

Jede der drei oben genannten Behauptungen wird in zwei statistischen Tests ausgewertet. Dabei wird zum einen ein Signifikanztest zur Gleichheit der Verteilungen und zum anderen ein Signifikanztest zur Verschlechterung der Metrik durch RL basierten Gegenspieler durchgeführt. Die Thesen zur gleichen oder schlechteren Verteilung der Metrik durch trainierten Gegenspieler werden in den Signifikanztests als H0 Hypothese eingesetzt. Die entsprechenden Gegenhypothesen formulieren jeweils die gegensätzliche Annahme einer Ungleichheit der Verteilungen oder einer Verbesserung der Metrik. Fasst man diesen Aufbau der statistischen Signifikanztests zusammen, können folgende Testhypothesen festgelegt werden.

Gleichheit des ersten Messwerts

- **Hypothese H0:** *Die erzielte kumulierte Belohnung im dritten Testszenario ist signifikant gleich zu der des ersten Testszenarios.*
- **Hypothese H1:** *Die erzielte kumulierte Belohnung im dritten Testszenario ist signifikant unterschiedlich zu der des ersten Testszenarios.*

Verschlechterung des ersten Messwerts

- **Hypothese H0:** *Die erzielte kumulierte Belohnung im dritten Testszenario ist signifikant geringer zu der des ersten Testszenarios.*
- **Hypothese H1:** *Die erzielte kumulierte Belohnung im dritten Testszenario ist signifikant höher zu der des ersten Testszenarios.*

Gleichheit des zweiten Messwerts

- **Hypothese H0:** *Die Varianz der erzielten Belohnung im dritten Testszenario ist signifikant gleich zu der des ersten Testszenarios.*
- **Hypothese H1:** *Die Varianz der erzielten Belohnung im dritten Testszenario ist signifikant unterschiedlich zu der des ersten Testszenarios.*

Verschlechterung des zweiten Messwerts

- **Hypothese H0:** *Die Varianz der erzielten Belohnung im dritten Testszenario ist signifikant höher zu der des ersten Testszenarios.*
- **Hypothese H1:** *Die Varianz der erzielten Belohnung im dritten Testszenario ist signifikant geringer zu der des ersten Testszenarios.*

Gleichheit des dritten Messwerts

- **Hypothese H0:** *Die erreichte Anzahl von unbeabsichtigten Misserfolgen ist im dritten Testszenario ist signifikant gleich zu der des ersten Testszenarios.*
- **Hypothese H1:** *Die erreichte Anzahl von unbeabsichtigten Misserfolgen ist im dritten Testszenario ist signifikant unterschiedlich zu der des ersten Testszenarios.*

Verschlechterung des dritten Messwerts

- **Hypothese H0:** *Die erreichte Anzahl von unbeabsichtigten Misserfolgen ist im dritten Testszenario ist signifikant höher zu der des ersten Testszenarios.*
- **Hypothese H1:** *Die erreichte Anzahl von unbeabsichtigten Misserfolgen ist im dritten Testszenario ist signifikant geringer zu der des ersten Testszenarios.*

Die Auswahl des Signifikanztests wird anhand der Wahrscheinlichkeit einer vorliegenden Normalverteilung vorgenommen. Kann nach einem Kolmogorov- oder Shapiro- Test zu einem Signifikanzniveau von 10 % eine Normalverteilung angenommen werden, lässt sich zur Überprüfung der Hypothesen ein T-Test einsetzten. Ist keine Normalverteilung gegeben, wird ein Mann-Whitney U Test verwendet. Liegt schlussendlich der P-Wert des T- oder Mann-Whitney U Signifikanztests zur Prüfung eines Unterschieds unter 10 %, so wird die H0 Hypothese abgelehnt und die Abweichung in den Robustheitsdaten als signifikant betrachtet. Weist der anschließende statistische Signifikanztest einen P-Wert unter dem Signifikanzniveau, wird H0 abgelehnt und die Verbesserung mittels RL Gegenspieler als signifikant wahrgenommen. Die Beantwortung der Forschungsfrage erfolgt abschließend mit der Betrachtung der angenommenen oder abgelehnten Forschungshypothesen. So werden die verschiedenen Merkmale der RL Policies untersucht und es kann festgestellt werden, welche Effekte auf Robustheit erzielt worden sind.

3.2 Programmanforderungen

Für die Durchführung des Laborexperiments wird ein Softwareprogramm implementiert, welches in der Lage ist, die zuvor beschriebene Methodik umzusetzen. In diesem Kapitel werden die dafür bestehenden technischen und funktionalen Anforderungen genauer thematisiert.

Eine allgemeine Anforderung liegt in der Wahl einer Entwicklungssprache, welche die Entwicklung der Simulation anhand von Softwareframeworks und zugleich die Integration mit Algorithmen des verstärkenden Lernens erlaubt. Die Entwicklungssprache und verwendete Softwarebibliotheken sollten einer möglichst aktuellen Version entsprechen, und deren Einsatz mit ihrer Version

dokumentiert sein. Durch die Dokumentierung kann ein gleiches Abbild der Softwareumgebung auf anderen Instanzen eines Betriebssystems eingerichtet werden.

3.2.1 Anforderungen der Simulationsumgebung

Die übergeordnete Hauptaufgabe der Simulation ist die Generierung von Trainingsdaten für die Optimierung von Strategien mittels RL. Die Qualität der Simulation und besonders der Grad des Realismus spielen dabei besonders für die spätere Auswertung der Robustheit einzelner Policies eine wichtige Rolle. In diesem Abschnitt wird die Hauptaufgabe in mehrere Anforderungen unterteilt, welche im Entwicklungsprozess umzusetzen sind.

Aus der Betrachtung des aktuellen Stands der Forschung und der Praxis für die Entwicklung von Simulationen geht hervor, dass besonders das OpenAI Gym Framework bzw. dessen Nachfolger Gymnasium in diesem Kontext ein wichtiger Bestandteil ist. Das Gymnasium Framework definiert die wichtigsten Funktionsschnittstellen, welche zur Entwicklung der Simulation im Rahmen dieser Arbeit zu implementieren sind.

Eine der Funktionen beschreibt die **Initialisierung** der eigenen Umgebungsklasse, welche von der Umgebungsklasse der Gymnasium-Bibliothek erbt. Initial sind darin für die Simulation die Startpositionen der verteidigenden Drohne und der angreifenden Drohne sowie der anzugreifende Zielpunkt anzugeben. Die Simulation soll eine zufällige Platzierung der Drohnen und des Zielpunktes im Raum ermöglichen, jedoch ausschließlich unter der Bedingung, dass die direkte Strecke zum Zielpunkt für die verteidigende Drohne kürzer ist, als für die angreifende Drohne. Zur Erfüllung dieser Anforderung muss eine Wahrscheinlichkeitsfunktion für die Zielpunktkoordinaten sowie je Drohne, für die drei Startkoordinaten X, Y und Z bestimmt werden. Die Randomisierung dieser Koordinaten soll über eine Variable an- und abgeschaltet werden können, sodass einfach zwischen Trainings- und Testszenario gewechselt werden kann. Als Spielgebiet wird eine flache feste Ebene auf der Höhe Null eingesetzt, über dessen der drei dimensionale Raum unbegrenzt in X-, Y- und Z-Richtung zur Verfügung steht.

Eine weitere Funktion beinhaltet die Definition des **Beobachtungsraums**, also welche Informationen für den Agenten wahrnehmbar sind. In der beschriebenen Simulation soll der Agent die aktuelle Position und Richtungsgeschwindigkeit der Drohne wahrnehmen. Jede dieser Eigenschaften wird von der verteidigenden Drohne und zusätzlich von der angreifenden Drohne wahrgenommen. Insgesamt stellt der Beobachtungsraum somit ein Datenobjekt dar, welches alle Informationen wie Koordinaten, Geschwindigkeiten und Drehzahlen beinhalte.

Neben dem Beobachtungsraum muss auch ein **Handlungsraum** festgelegt werden, welcher die Steuerung der verteidigenden Drohne umfassen soll. Hierbei soll durch den Agenten die Geschwindigkeit und ihre Richtung in Form einer vier elementigen Liste vorgegeben werden. Auch die regelbasiert gesteuerten Drohnen müssen damit eine dieser Form entsprechende Aktion erzeugen. Anschließend besteht die Anforderung, diese Aktionen in einen konkreten Einfluss auf den Zustand des Quadrokopters zu übersetzen.

Führt der Agent eine Aktion seines Handlungsraums aus, wird die Implementierung eines **Zeitschritts** der Simulation benötigt. In dieser muss deklariert sein, wie sich die gewählte Aktion auf den Zustand der Drohnen auswirkt. Während jedes Zeitschritts der Simulation sind dabei die Kriterien zum Zurücksetzen der Simulation zu prüfen.

Im Anwendungsbereich dieser Arbeit ist die Simulation zurückzusetzen, sobald das Abfangen der angreifenden Drohne erfolgt ist, oder eine der Drohnen in Kontakt mit dem Untergrund kommt. Das Gymnasium Framework definiert die **Rücksetzfunktion**, welche den initialen Simulationszustand herstellt, sobald ein Kriterium erfüllt ist. Durch die Schrittfunktion ist abschließend die neue Wahrnehmung der Simulation, die Erfüllung des Rücksetzkriteriums und der Belohnungswert zurückzugeben.

Zur Berechnung einer entsprechenden Belohnung, für die Ausführung der gewählten Aktion, ist eine **Belohnungsfunktion** einzusetzen. Diese muss anhand des neuen Simulationszustandes Kennzahlen ermitteln und gewichten und dadurch eine numerische Bewertung der neuen Situation vornehmen. Hinsichtlich der Maximierung des kumulierten Belohnungswertes optimiert der Agent mittels des RL Algorithmus seine Aktion in Abhängigkeit der Simulation. Die Anforderungen dieser Optimierung der Aktionen werden in der nachfolgenden Sektion detailliert betrachtet.

3.2.2 Anforderungen des Optimierungsverfahrens

Das Optimierungsverfahren trägt die Aufgabe zur Bestimmung der bestmöglichen Aktion zu jedem Zeitschritt der Simulation. Dabei ist die Strategie zur Steuerung des Quadrokopters mittels eines RL-Algorithmus zu optimieren. Da die eigene Implementierung des RL-Algorithmus nicht im Fokus dieser Arbeit liegt, werden frei verfügbare Softwarebibliotheken verwendet. Eine Bibliothek, welche eine Vielzahl an RL-Algorithmen beinhaltet, ist Stable-Baselines3 nach Raffin u. a. 2021. Die Softwarebibliothek integriert weitere Bibliotheken wie Tensorboard zum Messen von Trainings- und Evalutationskennzahlen, was die Durchführung der Forschungsmethodik unterstützt. Weiterhin ist eine umfangreiche Dokumentation zur Anwendung von RL-Algorithmen in Verbindung mit dem Gymnasium Framework vorhanden, was die Umsetzung des Trainingsprozesses in Verbindung mit der Simulationsumgebung erleichtert. Aus der Softwarebibliothek Stable-Baselines3 soll der PPO-Algorithmus zum Optimieren je einer Policy in jedem Trainingsszenario eingesetzt werden.

Der Trainingsprozess beinhaltet im ersten Schritt die Bestimmung der unabhängigen Variablen. Im Kontext dieser Arbeit sind die die Eigenschaften der Simulation nach der Wahl des Trainingsszenarios in der Simulation einzustellen. Alle Eigenschaften der Szenarios sollen über die Veränderung weniger Parameter als zentrale Schnittstelle im Programm auswählbar sein. Anschließend sind diejenigen Metriken anzugeben, welche im Trainingsprozess erfasst werden sollen. Diese beinhalten die durchschnittliche Episodenlänge, die kumulierte Belohnung sowie die Erfolgsrate des Zusammenstoßes beider Drohnen, über alle im Trainingsprozess evaluierten Strategien. Zur

Erfassung dieser Trainingsmetriken gilt es, die Integration mit der Tensorboard Bibliothek zu nutzen. Daraufhin ist das Training über die dafür vorgesehene Funktion des RL-Modells zu starten und nach dessen Ablauf das optimierte Modell zu speichern. Das Speicherformat ist so zu wählen, dass gespeicherte Modelle zur Auswertung in das Testszenario geladen werden können.

Aufgrund des hohen dreidimensionalen Aktionsraums, welcher zu jedem Zeitschritt zu optimieren ist, soll vor dem beschriebenen Trainingsprozess ein Training durch IL eingesetzt werden. Im IL Prozess ist ein Modell dahingehend zu optimieren, eine vorgeschriebene regel-basierte Strategie zur Steuerung des Quadrokopters nachzuahmen. Die Policies der später mittels RL zu optimierenden Quadrokopter werden demnach zunächst regelbasiert als Experte definiert. Daraufhin ist es das Ziel, die Gewichte eines neuronalen Netzes mithilfe eines IL-Algorithmus wie z.B. Behavioral Cloning so anzupassen, dass diese bestmöglich die regelbasierte Strategie widerspiegeln. Anschließend sind die in diesem Prozess erlernten Gewichte in ein RL-Modell Softwareobjekt zu transferieren und mittels des gewählten RL-Algorithmus feinzujustieren. Durch diesen Transfer soll das Training der RL-Modelle beschleunigt werden, da die Startgewichte bereits näher der optimalen Strategie liegen als zufällig gewählte Gewichte.

3.2.3 Anforderungen des Laborexperiments

Mithilfe der programatischen Implementierung des Laborexperiments sollen die notwendigen Messdaten erhoben, die Forschungshypothesen evaluiert, und die Forschungsfrage beantwortet werden. Dazu ist es notwendig, die beschriebenen Metriken, wie die kumulierte Belohnung deren Varianz und die Anzahl an Misserfolgen, für alle Policies im Testszenario zu erfassen. Das Testszenario wird durch eine Instanz der Simulationsumgebung repräsentiert, die zum Training veränderte Parameter beinhaltet.

Die Abweichung der Domäne zu den Trainingsszenarien wird durch die zusätzliche Randomisierung des Zielpunktes und durch die Verbesserung des regelbasierten Gegenspielers erzielt. Die Verteilung des Zielpunktes soll dabei ähnlich zu den Startkoordinaten so gewählt sein, dass sich die verteidigende Drohne näher an ihr befindet. Das Abfangen eines direkten Anflugs der anzugreifenden Drohne bleibt dadurch möglich. Im Testszenario soll die Verbesserung der anzugreifenden Drohne aus einer parabelförmigen anstatt geraden Fluglinie zum Zielpunkt bestehen. Damit die Validität der Messdaten gegeben ist, muss sichergestellt sein, dass alle Strategien den gleichen Testbedingungen und demnach denselben zufälligen Start- und Zielpunkten unterliegen. Hierfür ist ein Zufallsstartwert zu übergeben, worunter bei jedem Programmstart die gleichen zufälligen Testbedingungen hergestellt werden. Anschließend ist jede Policy in mehrfachen verschiedenen Episoden auszuführen und deren jeweilige Leistungsverhalten zu messen.

Innerhalb eines Testszenarios sind zu jedem Zeitschritt die aktuelle Belohnung zu speichern und zu messen, ob ein Misserfolg der Drohne vorliegt. Beide Variablen folgen dem Format einer Zeitreihe über den Verlauf der Testsimulation. Das Programm zur Auswertung muss nach

100-facher Ausführung der Policy im Testszenario die Kennzahlen der durchschnittlichen Belohnung und dessen Varianz berechnen. Zur Dokumentation und für die weitere Auswertung mittels statistischer Tests sind die Messdaten zu speichern.

Durch das Laden der Messdaten aus Dateien in entsprechende Datenobjekte liegen alle Metriken zur Bestimmung des statistischen Tests vor. Anschließend sind die Daten durch das Auswertungsprogramm, mittels der in der Forschungsmethodik beschriebenen Signifikanztests auf ihre Verteilung hin zu untersuchen. In Abhängigkeit der vorliegenden oder nicht vorliegenden Normalverteilung ist entsprechend der in der Forschungsmethodik beschriebene Test umzusetzen.

Die genaue Implementierung dieser Tests soll durch zusätzliche Softwarebibliotheken bestimmt sein. Anhand der Testergebnisse sind durch das Programm die H0 und H1 Hypothesen zu evaluieren und darauf aufbauend die Forschungshypothesen zu bestätigen oder zu verwerfen. Zusätzlich zu den Messdaten der Testszenarien sind die Ergebnisse der Signifikanztests zu speichern. Schlussendlich sollen in einer Tabelle zu jeder Forschungshypothese die Ergebnisse der statistischen Tests angeführt werden. Durch die gesamtheitliche Betrachtung der Robustheitsmerkmale kann final die Forschungsfrage beantwortet werden.

3.3 Implementierung der Simulation und des Experiments

Fortfolgend an die in der letzten Sektion beschriebenen Anforderungen, werden in dieser Sektion die finalen Implementierungen der einzelnen Programmabschnitte erläutert. In Anlehnung an den Aufbau der letzten Sektion, wird die Umsetzung der Anforderungen zu den Programmabschnitten Simulationsumgebung, Trainingsverfahrens und Laborexperiment dargelegt.

Allgemeiner Natur ist die Wahl der Entwicklungssprache Python. Die Entwicklungssprache ermöglicht eine hohe Kompatibilität mit bekannten Simulationsframeworks, RL-Softwarebibliotheken und Bibliotheken zur statischen Auswertung von Messdaten. Im Rahmen dieser Arbeit wird die Python Version 3.10 eingesetzt, um von den Möglichkeiten neuer Funktionen zu profitieren und zugleich eine möglichst hohe Kompatibilität zu existieren frei verfügbaren Bibliotheken aufzuweisen. Alle in dieser Arbeit verwendeten Bibliotheken sowie deren Abhängigkeiten sind in einer virtuellen Umgebung installiert, sodass Störeinflüsse bestmöglich durch bereits installierte Pakete vermieden werden. Ebenso sind alle eingesetzten Softwarepakete in einer Textdatei inklusive deren Version dokumentiert.

3.3.1 Programmumsetzung der Simulationsumgebung

Im Kern der Implementierung der Simulationsumgebung liegt das Gymnasium-Programmiergerüst. Die entwickelte Simulation baut dabei auf bestehenden Programmbibliotheken auf. Dies fördert vor allem die im Zuge dieser Arbeit umsetzbare Qualität und ermöglicht einen hohen Grad an Realismus, welcher wie in den Anforderungen beschrieben unmittelbar das Sim2Real Problem

und damit die Robustheit beeinflusst. Weiterhin würde eine Neuentwicklung einer Simulationsumgebung keinen Entwicklungsaufwand darstellen, welcher einen Einfluss auf die auszuwertende Forschungsfrage ausüben würde. Wird die im zweiten Kapitel angeführte Auswahl an in der Literatur beschriebenen Simulationsumgebungen betrachtet, kann aus diesen eine Basis zur Entwicklung der eigenen Simulationen gewählt werden.

Zur Auswahl einer Basissimulation, in welcher die eigenen Trainings- und Testszenarien abgebildet werden können, sind unterschiedliche Kriterien zu beachten. Ein Kriterium ist die Kompatibilität der Basissimulation mit der gewählten Entwicklungssprache. Hierbei sollte entweder die Simulation selbst in Python, oder eine entsprechende Schnittstelle vorhanden sein. Die gewählte Simulation sollte eine Physik-Engine einsetzen, welche zum einen einen möglichst hohen Grad an Realismus erlaubt, um die Forschungsfrage möglichst valide zu beantworten. Zum Anderen sollte die Simulation einen beherrschbaren Rechenleistungsaufwand verursachen, da die Zielsetzung dieser Arbeit die Optimierung mehrerer Policies voraussetzt. Zur Erfüllung der zuvor beschriebenen Anforderungen muss die Basissimulation weiterhin mit dem Framework Gym oder dessen Nachfolger Gymnasium kompatibel sein. Um die Entwicklung des Optimierungsverfahrens zu unterstützen, wird eine RL Integration oder zumindest eine umfangreiche Dokumentation zu dessen Einsatz vorausgesetzt. Die nachfolgende Tabelle stellt die in Betracht gezogenen Drohnensimulationen, den für die Auswahl getroffenen Kriterien gegenüber.

	RotorS	CrazyS	AirSim	gym-pybullet-drones
Python API	X	X	X	X
Integration des Gymnasium Frameworks	X	X	X	X
höchster Realismusgrad			X	
kontrollierbarer Rechenleistungsaufwand	X	X		X
umfangreiche RL Integration und Dokumentation			X	X
Simulation mehrerer Quadrokopter enthalten				X

Tab. 5: Gegenüberstellung von Auswahlkriterien und bekannten Drohnensimulationen

Aus der Gegenüberstellung von Auswahlkriterien und aktuellen Simulation geht hervor, dass zunächst alle Simulationen eine Schnittstelle für die Entwicklungssprache bereitstellen. Dabei wurden native Schnittstellen sowie Schnittstellen aus zusätzlichen Softwarebibliotheken miteinbezogen. Auch die Integration des Gym oder Gymnasium Frameworks wird durch alle Simulationen eigens oder durch zusätzliche Bibliotheken sichergestellt. Bei der Betrachtung der Realitätsnähe zeigt die AirSim Umgebung aufgrund der verwendeten Unreal Physik-Engine den höchsten Grad

[227] Eigene Darstellung

an Realismus auf. Im Gegenzug werden durch diese Simulation Rechenkapazitäten erwartet, welche im Rahmen dieser Arbeit nicht zur Verfügung stehen. Eine umfangreiche Dokumentation und bereits vorhandene Integration von RL wird durch AirSim und gym-pybullet-drones gewährleistet. Die Simulationen RotorS und CrazyS bieten eine Integration mit RL nur auf Basis der wenig dokumentierten zusätzlichen Softwarebibliotheken GymFC und gym_multirotor. Wird das letzte Kriterium der Simulation mehrerer Drohnen betrachtet, tritt dies nur in Beispielen der gym-pybullet-drones Simulation auf. Wird die Erfüllung der Auswahlkriterien zusammengefasst, so wird erkenntlich, dass gym-pybullet-drones als einzige Drohnensimulationen die zusätzlichen Anforderungen der RL Integration und der Dokumentation erfüllt. Demzufolge wird für die nachfolgende Entwicklung der für diese Arbeit verwendeten Simulation auf der gym-pybullet-drones Bibliothek aufgebaut.

Die gym-pybullet-drones Simulation ist in einzelne Simulationszenarien gegliedert, welche je eine Python Datei umfassen. Als Aviaries gekennzeichnet, bilden sie je eine Trainingsumgebung nach der Schnittstellendefinition des Gymnasium-Frameworks ab. In der Struktur der Simulationsumgebung wird das Programmierkonzept der Vererbung verwendet, um ähnliche Simulationen auf denselben übergeordneten Klassen zu basieren. Jede erbende Klasse enthält damit alle Funktionen und Variablen der übergeordneten Elternklasse. Dies ermöglicht die verschieden starke Abstraktion der Simulation auf den unterschiedlichen Vererbungsebenen. Weiterhin ermöglicht die Bibliothek verschiedene Steuerungsarten der Quadrokopter. Eine Option ist die Steuerung über die direkte Vorgabe der Drehzahlen aller Rotatoren. Zusätzlich kann auch ein PID-Regler eingesetzt werden um die Steuerungssignale entgegenzunehmen. Zur Entwicklung der für diese Arbeit verwendeten Simulation wird die Steuerung über einen Richtungsvektor und einen Geschwindigkeitswert gewählt. Unabhängig des gewählten Aktionstyps werden alle Steuerungssignale durch Reglerklassen in konkrete Drehzahlen übersetzt. Dessen genaue Übersetzung wird im Rahmen dieses Kapitels nicht erläutert und ist in der entsprechenden Dokumentation nach Panerati u. a. 2021 nachzulesen. Auch der Beobachtungsraum kann zum einen visuell und zum anderen kinetisch erfolgen, wobei Letzteres im Rahmen dieser Arbeit verwendet wird. Der kinetische Beobachtungsraum stellt eine Reihe von wahrnehmbaren Eigenschaften wie Position, Ausrichtung, Geschwindigkeit und Drehzahl dar. Unabhängig der Überwachungsart kann durch die pybullet Physik-Engine ein grafisches Modell der Simulation erzeugt werden, welches in Abbildung 8 dargestellt wird.

Die Umsetzung, der zu Beginn dieses Kapitels beschriebenen Simulationsumgebung, basiert auf der Verwendung der Basissimulation und der Steuerungsklasse. Zusätzlich sind teilweise Funktionen aus ähnlichen, bereits nativ vorhandenen Szenarien mit Änderungen übernommen und im Programm entsprechend gekennzeichnet. Die beschriebene Simulation ist innerhalb einer Klasse nach dem Gymnasium Framework aufgebaut. Zusätzlich wird die Simulation durch eine erbende Klasse umhüllt, um für den Algorithmus eine verarbeitbare Abstraktionsschicht zu erzeugen. In dieser Schicht wird die Steuerung der angreifenden Drohne abstrahiert, sodass die Trainingsumgebung lediglich die Bestimmung einer Aktion für die verteidigende Drohne erwartet. Nachfolgend

[228] Eigene Darstellung

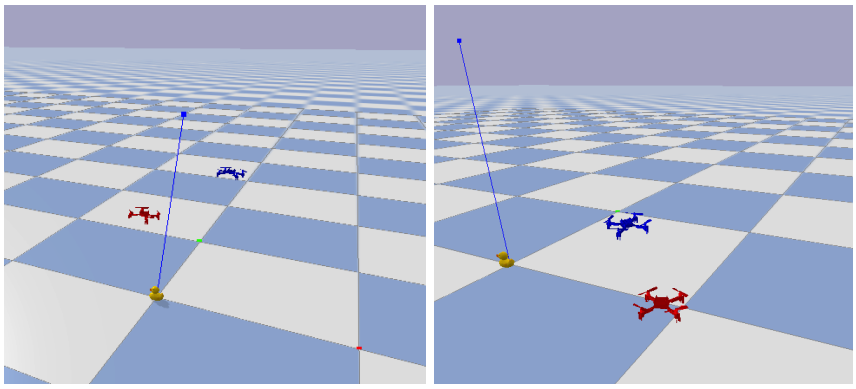


Abb. 8: grafisches Modell der Simulation[228]

wird die Umsetzung dieser beiden Klassen, nach den einzelnen Funktionen des Gymnasium-Frameworks erläutert.

Die Simulation beider Quadrokopter ist in der Klasse **DualDroneAviary** definiert. Zur Initialisierung einer Simulationsinstanz wird die **Init-Funktion** aufgerufen, welche die Übergabe aller simulationsrelevanten Parameter erwartet. Die Parameter umfassen unter anderem Informationen zur Anzahl und Art und Position der Drohnen, gewählte Handlungs- und Beobachtungsarten sowie die unabhängigen Variablen der Trainingsszenarien. Außerdem wird die Gewichtung der Belohnungsfunktion als Parameter entgegengenommen. Kernaufgabe der Funktion ist die Umsetzung der Eigenschaften eines Trainings- und Testszenarios. Dies beinhaltet u. A. die Erzeugung und Übermittlung der zufälligen Start- und Zielpunkte und des zufälligen Windeffektes mittels unterstützender Funktionen aus der eigenen Hilfsklasse. Die verteidigende Drohne wird zufällig innerhalb von 1,5 Meter in X- und Y-Richtung und 0.1 bis 3 Meter in Z-Richtung zum Ursprung initialisiert. Hingegen wird die angreifende Drohne im Rahmen von zwei bis sechs Metern uniform um den Ursprung herum auf einer möglichen Höhe von 0.1 bis fünf Meter platziert. Sind die Eigenschaften bestimmt, können diese zur Erzeugung der darunter liegenden Abstraktionsschicht der BaseAviary Instanz genutzt werden.

Der Beobachtungsraum in dieser Simulation wird durch einen Informationsvektor in der **ObservationSpac Funktion** definiert. Der für die Experimente eingesetzte Informationsvektor je Drohne enthält die Position, Fortbewegungsrichtung und Geschwindigkeit. Die Position wird durch dreidimensionale Koordinaten, die Ausrichtung durch Quaternion und Drehwinkel kodiert. Positionen können dabei von $-\inf$ bis $+\inf$ in X- und Y-Richtung und von 0 bis $+\inf$ in Z-Richtung betrachtet werden. Fortbewegungsrichtung und Geschwindigkeit werden als Geschwindigkeitsvektor von minus bis plus unendlich zu den Koordinatenachsen definiert. Insgesamt ist für jede Drohne ein Beobachtungsvektor mit sechs Elementen innerhalb eines Wörterbuchobjektes definiert.

Die Definition des Aktionsraums ist in der **ActionSpace-Funktion** bestimmt. Eine Aktion des ausgewählten Aktionstyps wird als vierdimensionaler Vektorbereich definiert. Jeder Wert des Vektors besitzt eine untere und obere Grenze, aus dessen Bereich die endgültige Aktion gewählt wird. Zur Steuerung des Quadrokopters wird im Rahmen dieser Arbeit der Aktions-

typ auf Basis der Geschwindigkeit verwendet. Der Wertebereich des Richtungsvektors ist dabei von $[-1, -1, -1]$ bis $[1, 1, 1]$ definiert. Das vierte Element spiegelt den Anteil der maximalen Geschwindigkeit von null bis eins wider.

Ist eine Aktion regelbasiert oder durch ein RL-Modell bestimmt, wird diese als Parameter an die **Step-Funktion** weitergegeben. Auf der Ebene der DualDroneAviary Klasse kann die übergebene Aktion in Abhängigkeit vom anfangs initialisierten Wind abgewandelt werden. Der Windvektor wird in Abhängigkeit des Winkels zwischen Bewegungsrichtung der Drohne und des Windes zur Aktion hinzugefügt. Der Drohnenbewegungsrichtung wird die Windrichtung addiert sowie die Geschwindigkeit in Abhängigkeit des Eintreffwinkels vollständig positiv, negativ oder gar nicht beeinflusst. Anschließend wird unter veränderter oder unveränderter Aktion die Schrittfunktion der Elternklasse aufgerufen. Die Schrittfunktion der Elternklasse übersetzt die Aktion aus dem Aktionsraum in konkrete Rotordrehzahlen und übt diese unter Einfluss der simulierten Physik aus. Zur Erhöhung des Realismusgrades wird zusätzlich zu den Effekten der pybullet Physik-Engine ein Boden- und Trägheitseffekt eingesetzt. Anschließend wird die neue Zustandsbeobachtung, die entstandene Belohnung, das Rücksetzkriterium und ein Datenobjekt für Zusatzinformationen zurückgegeben.

Die Bestimmung der Belohnung ist unter der **reward-Funktion** der DualDroneAviary Klasse implementiert. Innerhalb dieser Funktion werden aus dem aktuellen Zustand der Simulation Kennzahlen zu dessen Evaluation berechnet und gewichtet. Durch die Gewichtung und die Optimierung dieser kann das Strategieziel formuliert und der Einfluss der einzelnen Kennzahlen angepasst werden. Die Aufsummierung der Produkte aus den folgenden Kennzahlen und deren Gewichtung spiegelt die Belohnung zu jedem Zeitschritt wider:

- Distanz zwischen der angreifenden und verteidigenden Drohne
- Belohnung bei erfolgreichem Abfangen der angreifenden Drohne
- Negative Belohnung, sofern die angreifende Drohne den Zielpunkt erreicht hat
- Belohnung für eine möglichst direkte Flugrichtung zur gegnerischen Drohne
- Lineare Belohnung der Geschwindigkeit

Eine Optimierung der Gewichte wurde durch ausgiebige Evaluation unterschiedlicher Gewichtskombinationen und Messen der entsprechenden Erfolgsrate erzielt. Zu jedem Zeitschritt eintretende Belohnungen wurden aufgrund ihrer Häufigkeit in Fünfer-Schritten, außerordentliche Belohnungen in 10000er-Schritten untersucht. Insgesamt sind daraus unter anderem folgende in Abbildung 9 dargestellte Log-Dateien entstanden, welche den Optimierungsprozess widerspiegeln.

Die Optimierung der Belohnungsfunktion erzielte $[-10, 50000, 50000, 25, 20]$ als optimalen Gewichtsvektor, da hierunter mehrere Algorithmen wie PPO und A2C, eine positive Erfolgsrate verzeichneten.

[229] Diagramme durch die Tensorboard Bibliothek erstellt

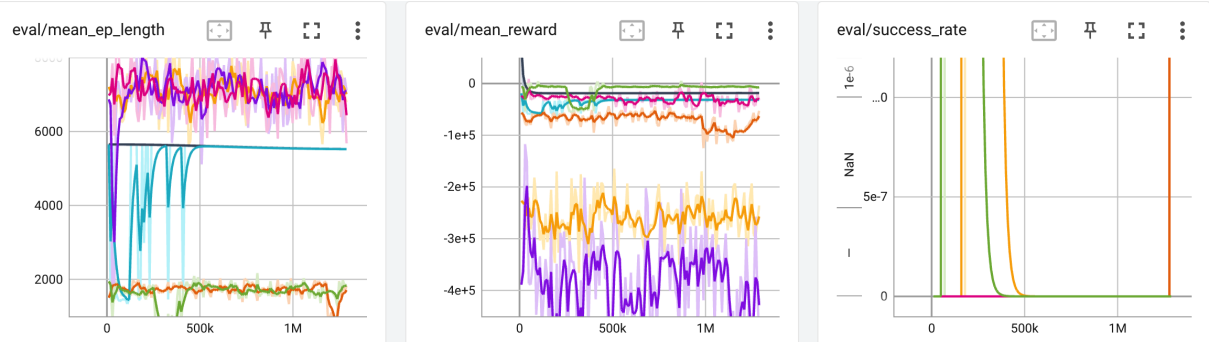


Abb. 9: Episodenlänge, Belohnung und Erfolgsrate unter verschiedenen Gewichtungen[229]

Ist das Rücksetzkriterium erfüllt, wird die Simulationsumgebung mittels **reset-Funktion** neu gestartet. Dabei werden neue Zufallspositionen der Drohnen und des Ziels sowie im vierten Trainingsszenario neue Windeigenschaften generiert. Sind diese Variablen verändert, kann die Rücksetzfunktion der darunterliegenden Abstraktionsschicht aufgerufen werden. Die Funktion der BaseAviary-Klasse setzt die Zeitschritte und die Drohneninformationen zurück und baut das grafische Modell neu auf. Über den Zugriff auf den Klienten der pybullet Physik-Engine werden die Drohnenobjekte sowie das Untergrundobjekt aus den URDF-Dateien geladen und an entsprechenden Stellen platziert. Zur Unterscheidung des verteidigenden Quadrokopters von der angreifenden Drohne werden zwei nahezu identische URDF-Dateien verwendet, welche sich nur in der Farbgebung zwischen rot und blau unterscheiden. Am Ende der Funktion wird der erste Beobachtungsvektor der neuen Episode sowie ein Infoobjekt zurückgegeben.

Die zuvor beschriebene Klasse vererbt ihre Variablen und Funktionen auf die höhergelegenen Abstraktionsschichten der **DefenderDroneWrapper** und **AttackerDroneWrapper** Klasse. Auf dieser Ebene bietet die Simulation ihre Schnittstelle für die Verwendung durch den IL- und RL-Algorithmus. Dafür ist es notwendig, den Beobachtungsraum sowie den Aktionsraum neuzudefinieren. Zur **Initialisierung** werden zum einen alle Parameter zur Instanziierung des DualDroneAviary Objekt erwartet. Zum anderen sind zusätzlich die Strategie und dessen Rauschverhalten der nicht durch RL oder IL gesteuerten Drohne anzugeben, sodass diese in Klassenvariablen gespeichert werden. Auf Ebene der Wrapper-Klassen wird der **Beobachtungsraum** mit allen Informationen beider Quadrokopter zu einem zwölf elementigen Vektor zusammengefasst. Der **Handlungsraum** hingegen umfasst auf dieser Ebene lediglich einen vier-elementigen Vektor mit dem bekannten Wertebereich. Zur Ausführung eines Simulationsschrittes werden in der **Step-Funtkion** die Aktion der jeweils gegnerischen Drohne und die durch das Modell übergebene Aktion in ein gemeinsames Datenobjekt überführt. Anschließend kann mittels dieses Datenobjektes die Schrittfunktion der Elternklasse aufgerufen, und deren Rückgabewerte an den zu optimierenden Algorithmus weitergereicht werden. Als **Belohnungswert** wird auf die im Trainingsszenario relevante Komponente des Belohnungsobjektes zugegriffen.

Werden die beschriebene Umsetzung der Simulationsumgebung zusammengefasst, kann folgendes Klassendiagramm und deren Funktionsabhängigkeiten in Abbildung 10 betrachtet werden.

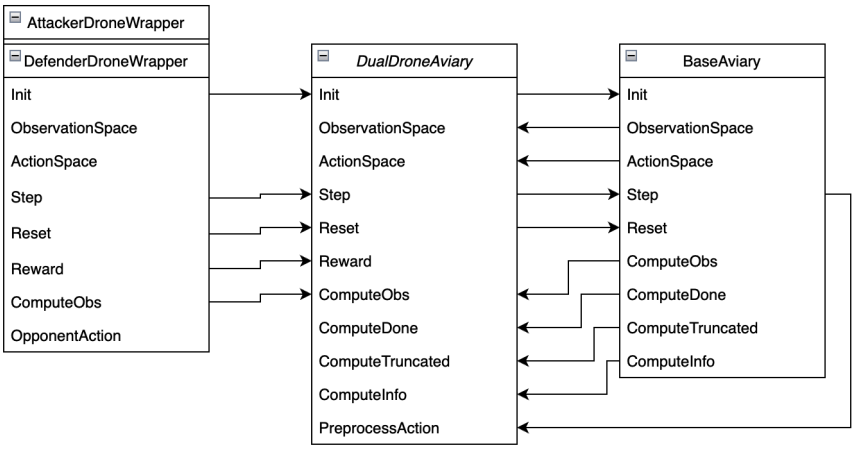


Abb. 10: Klassendiagramm mit Funktionsabhängigkeiten[230]

3.3.2 Programmumsetzung des Optimierungsverfahrens

Mit der Entwicklung der Simulationsumgebung stehen Trainingsdaten zur Verfügung, um darauf aufbauend Strategien des verstärkenden Lernens zu optimieren. Der gesamte Optimierungsprozess unterteilt sich in den Einsatz von imitierendem Lernen zu Beginn und folgender Optimierung mittels RL-Algorithmen. Die Umsetzung des Optimierungsverfahrens wird innerhalb der Python Dateien *imitation_train.py* und *train.py* beschrieben.

Das imitierende Training beginnt mit dem Herstellen des Trainingsszenarios zum Akquirieren der Demonstrationen des Experten. Für die Erhebung der Trainingsdaten wird zunächst die Simulation unter den Trainingsgegebenheiten in 20000 Episoden ausgeführt. Für das imitierende Lernen wurde der Parameter AGGR_PHY_STEPS der BaseAviary Klasse auf 50 gesetzt, was bedeutet, dass die Schrittfunktion nur alle 50 Zeitschritte aufgerufen wird und so je Episode deutlich weniger Aktionen erfolgen und Transitionen aufgezeichnet werden. Damit verteilen sich die daraus entstehenden Transition-Objekt der verwendeten Softwarebibliothek Imitation, auf eine höhere Anzahl von verschiedenen Start- und Zielkoordinaten. Das Transition-Objekt beschreibt einen durch die Aktion entstehenden Übergang vom bekannten zu einem neuen beobachteten Zustand sowie den Erhalt der Belohnung. Hieraus ergeben sich eine Menge von aufeinanderfolgenden oder zeitversetzten Transition-Objekten. Die Übergänge in den Trainingsdaten werden gemischt, so dass sie nicht entsprechend ihrer Entstehung geordnet liegen, um die Datenvielfalt innerhalb einzelner Batches zu erhöhen und korrelierende Effekte zu mindern. Basierend auf diesen Transaktionsdaten werden in 4000 Epochen die Gewichte der zu optimierenden Strategie angepasst. Die Anpassung der Gewichte einer Akteur-Kritiker-Strategie wird durch den Behavioral Cloning Algorithmus vorgenommen. Dessen Ziel ist es, mittels eines neuronalen Netzes eine Funktion von einem beobachteten Zustand zu der vom Experten gewählten Aktion zu approximieren.

Als Experte wird eine regelbasierte Version der jeweils zu optimierenden Strategie eingesetzt. Alle regelbasierten Strategien sind in der *rule-based.py* Datei deklariert und umfassen eine regel-

[230] Eigene Darstellung

basierte Strategie für die verteidigende Drohne sowie die Trainings- und Testversion der anzugreifenden Drohne. Die verteidigende Drohne verfolgt regelbasiert die direkte Flugbahn zu einem Vorhaltepunkt in Bewegungsrichtung der Angreiferdrohne. Der Vorhaltepunkt rückt mit geringer werdender Distanz zur generischen Drohne immer näher zu dessen aktueller Position. Die angreifende Drohne wird im ersten Trainingsszenario regelbasiert gesteuert, indem sie sich von ihrem Startpunkt in gerader Fluglinie zum Zielpunkt begibt. Dessen Erweiterung zur parabelförmigen Fluglinie wird in der nachfolgenden Sektion der Umsetzung des Laborexperiments behandelt.

Wurde ein Akteur-Kritiker Modell durch imitierendes Lernen hin zum regelbasiertem Verhalten optimiert, wird dieses als Basis zur weiteren Optimierung mittels RL verwendet. Im RL-Prozess ist zu beachten, dass anders als beim imitierenden Lernen zu jedem Zeitschritt eine neue Aktion gewählt wird, da der Parameter AGGR_PHY_STEPS hier standardmäßig bei 1 liegt. Mit dem Einsatz des *train.py* Programms lässt sich das in einer Pickle-Datei gespeicherte Modell laden. Dazu wird das grundlegende neuronale Netz des PPO-Objekts mit derselben Architektur und den trainierten gewichten des Modells aud dem imitierenden Lernen initialisiert. Anschließend werden die Gewichte und Vorurteile in ein Objekt des PPO-Algorithmus von Stable-Baselines3 übertragen. Mittels der learn-Funktion kann die erzeugte Strategie im gegebenen Trainingsszenario über eine Anzahl von Zeitschritten weiter optimiert werden. Die während des RL Prozesses verwendeten Trainingszeitschritte lagen bei 2,592,000, welche sich aus der Trainingszeit von 180 Minuten und der Simulationsfrequenz von 240 Schritten pro Sekunde zusammensetzen. Durch die Tensorboard Bibliothek werden während des Trainingsprozesses gemittelte Metriken über alle Episoden berechnet sowie alle 10000 Zeitschritte eine Evaluation durchgeführt. In der Evaluationsperiode wird die aktuelle Strategie bemessen und deren Episodenlänge, erzielte Belohnung und Erfolg aufgezeichnet. Als erfolgreich wird die Simulation gekennzeichnet, sobald die Strategie der Verteidigerdrohne die angreifende Drohne durch Kollision abfangen konnte. Nach dem Abschluss der Optimierung werden die neuen Modelle in Pickle Dateien gespeichert.

3.3.3 Programmumsetzung des Laborexperiments

Als letzter Teil der Umsetzung wird die Durchführung des Experiments betrachtet. Kernaufgabe dieses Teils ist die Erhebung und Auswertung der Messdaten entsprechend des in der Einleitung des Kapitels verfassten Experimentaufbaus. Die Erhebung der Testdaten erfolgt durch das *test.py*, die Auswertung durch das *experiment_ evaluation.py* Skript.

Mit der Ausführung des ersten Skripts wird eine Testklasse instanziiert und die Simulation nach den Eigenschaften des Testszenarios erzeugt. Das Testszenario implementiert die Anforderung der Randomisierung des Zielpunktes. Zur zufälligen Bestimmung des Zielpunkts wird, ebenso wie zur Initialisierung der Verteidigerdrohne, eine uniforme Verteilung bis zu 1,5 Meter um den Ursprung verwendet. Eine uniforme Verteilung fördert dabei die Varianz der Testszenarien und erhöht so die Validität des Experiments. Anders als bei der Initialisierung der Drohnen wird die Höhe des Zielpunktes stets auf 0,5 Meter festgelegt, um so Bodenkontakt und Simulationserfolg

Episode	Episodenlänge	kumulierte Belohnung	Belohnungsabweichung zum Mittelwert	Episodenmisserfolg
1	1931	13512	3192	0
2	1579	8413	1873	1
3	1431	16920	6308	0
4	1944	6741	3701	1
5	1508	10173	89	0

Tab. 6: Beispieltabelle der erhobenen aggregierten Messdaten

abzugrenzen. Zusätzlich zur Randomisierung des Zielpunktes wird im Testszenario der regelbasierte angreifende Quadrokopter im Aspekt seiner parabelförmigen Anflugsstrategie verändert. Ausgehend von den Startkoordinaten des Angreifers werden dazu insgesamt fünf Wegpunkte entlang einer unten geöffneten Parabel berechnet, welche nacheinander angeflogen werden. Die Parameter der Parabel berechnen sich wie in den Formeln 8 bis 10 dargestellt.

$$a = -1/(2 * (\text{Distanz zum Zielpunkt}/6)) \quad (8)$$

$$b = 0 \quad (9)$$

$$c = \text{Z-Koordinate des Startpunktes} \quad (10)$$

Alle Messdaten werden aus der hundertfachen Durchführung der Testsimulation erhoben. Zu jedem Zeitschritt jeder Episode wird durch das RL-Modell eine bestimmte Aktion ausgeübt und eine Teilmenge des daraus resultierenden Zustands gemessen. Die Teilmenge umfasst die durch die Aktion unmittelbar ausgelöste Belohnung und einen Wahrheitswert, ob mit Aktion ein Misserfolg der Simulation eingegangen wurde. Beide Werte werden gekennzeichnet mit der jeweiligen Episode und dem Zeitschritt, in einer Tabelle als Datensatz hinzugefügt. Nach dem erfolgreichen Durchlaufen aller Zeitschritte der 100 Episoden wird die Tabelle als Komma separierte Wertedatei (CSV) gespeichert.

Die erstellte Datei wird anknüpfend mittels *experiment_ evaluation.py* ausgewertet, indem die Messdaten geladen, die Signifikanztests durchgeführt und jeweils ein Wahrheitswert für die Annahme/Ablehnung der Hypothesen bestimmt wird. Für jeden dieser drei Schritte wurde eine Funktion entwickelt, welche im Skript aufgerufen wird. Zum Laden der Messdaten sind die Speicherpfade zweier Wertedateien anzugeben, woraufhin der Verlauf der Kennzahlen zu einzelnen Episoden aggregiert werden. Dabei wird zunächst die Episodenlänge, die kumulierte Belohnung und der final vorliegende Erfolgszustand ermittelt. Basierend darauf kann die quadrierte absolute Abweichung der Belohnung je Episode zu dem Mittel aller kumulierten Belohnungen im Testprozess berechnet werden. Nach dem Laden und Vorverarbeiten entspricht der Aufbau der Messdaten zweier Tabellen ähnlich der Tabelle Sechs.

Im nächsten Schritt ist mittels statistischer Signifikanztests die Ähnlichkeit zu einer Normalverteilung sowie die ungleiche Verteilung und mögliche Verbesserung aller Metriken zu untersuchen.

[230] Eigene Darstellung

Die Implementierung der Signifikanztests wird im Rahmen dieser Arbeit durch die Softwarebibliothek SciPy bereitgestellt. Iterativ werden für alle Metriken beider Messtabellen die Ähnlichkeit zur Normalverteilung überprüft, um zwischen einem T-Test und einem Mann-Whitney U Test auszuwählen. Anschließend wird je Metrik der entsprechende Test zur Ungleichheit und Verbesserung der Verteilung ausgeführt, woraus je Test eine Teststatistik und ein P-Wert gespeichert wird. Liegt der P-Wert eines Tests unter dem Signifikanzniveau von 10 % wird die H0 Hypothese abgelehnt und daraus schlussfolgernd die Ungleichheit oder Verbesserung der Metrik angenommen. Insgesamt wird diese Auswertung der Leistungsdaten zweier Strategien zweimal durchgeführt. Dabei werden die erzielten Messdaten zwischen den Strategien aus Trainingsszenario eins und drei sowie aus Szenario eins und vier verglichen. Der vollständige Prozess des Experiments kann auch als BPMN Modell wie in Abbildung 11 dargestellt werden.

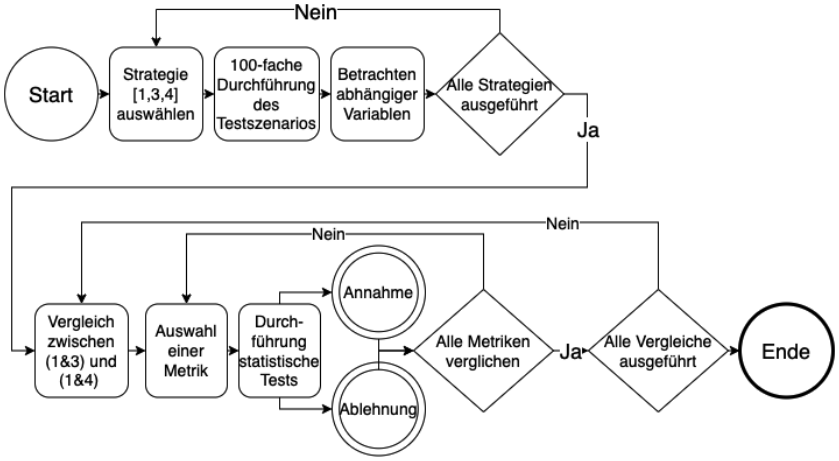


Abb. 11: BPMN Prozess des Laborexperiments[231]

[231] Eigene Darstellung

4 Ergebnisse des Laborexperiments

Anschließend an die Durchführung des Laborexperiments werden innerhalb dieses Kapitels die erzeugten Messdaten betrachtet und die zur Beantwortung der Forschungsfrage aufgestellten Hypothesen ausgewertet. Es gilt dabei einen möglichen Effekt des trainierten Gegenspielers auf die Robustheit des RL-Modells zu untersuchen. Dazu werden die Leistungen der Strategien aus den unterschiedlichen Trainingsszenarien im abweichenden Testszenario verglichen. Eine Leistungsmessung im abweichenden Testszenario zeigt, welche Lernumgebung eine stabilere und robustere Strategie hervorbringt, welche auch unter unbekannten Effekten performanter ist.

Die auszuwertenden Leistungsmetriken aggregiert je Testepisode können im Anhang gefunden werden. Im ersten Vergleich werden die Strategien aus dem Training mit regelbasiertem und mittels RL optimierten Gegenspieler untersucht. Weiterhin wird ein Vergleich der Testmetriken zwischen dem Training ohne und mit randomisierter Umgebung bei regelbasiertem Gegenspieler durchgeführt. Dazu werden mehrfach dieselben statistischen Signifikanztests durchgeführt, um zum einen den Effekt der DR zu ermitteln. Zum anderen werden die Einflüsse der im Rahmen dieser Arbeit entwickelten Methoden wie der DR und dem RL basierten Gegenspieler gegenübergestellt. Das Ende dieses Kapitels bildet die Beantwortung der Forschungsfrage anhand der ausgewerteten Test- und Forschungshypothesen.

4.1 Vergleich der Robustheit der Strategien aus dem Training mit regelbasierten und RL Gegenspieler

Metrik	Gleichheit P-Wert	Verschlechterung P-Wert	Ungleichheit	Verbesserung
Erzielte kumulierte Belohnung	0.6276673409642575	0.6870324210336459	False	False
Abweichung vom Belohnungs-Mittelwert	1.2200477617083057e-22	1.0	True	False
Häufigkeit von Misserfolg	0.1814596593937995	0.09072982969689974	False	True

Tab. 7: Ergebnisse der statistischen Tests aus dem Leistungsvergleich der Modelle aus den Trainingsszenarien 1 und 3

Anhand Tabelle 7 lassen sich die Ergebnisse der statistischen Signifikanztests ablesen. Die Ergebnisse basieren dabei auf dem Leistungsvergleich der Modelle aus den Trainingsszenarien 1 und 3. Den Tabellenspalten werden zum einen die P-Werte der Tests zur Gleichheit und Verschlechterung der Verteilung zugeordnet. Zum anderen wird darauf aufbauend die Erfüllung der H1-Hypothese anhand des gewählten Signifikanzniveaus von 10 % evaluiert. Den Zeilen der Tabelle können die Ergebnisdaten zu je einer der drei untersuchten Metriken entnommen werden.

4.1.1 Betrachtung der kumulierten Belohnung

Hypothese 1: *Die im Testszenario erzielte kumulierte Belohnung ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig höher, als die Policy aus dem Training mit regelbasiertem Gegenspieler.*

Die erste in der Einleitung aufgestellte Forschungshypothese untersucht den Trainingseinfluss auf die in einer Testepisode erzielte kumulierte Belohnung. Mit der Hypothese 1 wird die Behauptung aufgestellt, dass durch die Wahl eines Trainingsszenarios mit RL-optimierten Gegenspieler zuverlässig eine bessere kumulierte Belohnung erzielt werden kann, als mittels eines Trainings mit regelbasierten Gegenspieler. Die Auswertung der Hypothese erfolgt durch statistische Signifikanztests zur Verteilung der kumulierten Belohnungen je Testepisode. Zur Bestätigung dieser Forschungshypothese müssen die P-Werte zur Gleichheit und Verbesserung der Belohnungsverteilung jeweils geringer als das Signifikanzniveau sein.

Der P-Wert des Signifikanztests zur Gleichheit der Verteilungen der kumulierten Belohnungen liegt in etwa bei 0.6277. Daraus kann festgestellt werden, dass dieser größer als das gewählte Signifikanzniveau von 10 % bzw. 0.1 ist. Die im Test untersuchte H0 Hypothese einer gleichen unterliegenden Verteilung ist damit anzunehmen. Eine Ungleichheit der unterliegenden Verteilung der Belohnungen, wie in Hypothese H1 behauptet, ist folglich abzulehnen. Da demnach keine Unterschiede in der Verteilung der im Testszenario erzielten Belohnungen vorliegen, ist festzuhalten, dass die Forschungshypothese 1 abgelehnt wird. Im Laborexperiment konnte somit kein Nachweis erbracht werden, dass ein Optimieren eines RL-Modells, unter Verwendung eines selbst mittels RL optimierten Gegenspielers, zuverlässig zu einer höheren kumulierten Belohnung führt.

Schließt man von diesem Ergebnis auf dessen Bedeutung für die Robustheit der Strategien aus verstärkendem Lernen zeigt sich folgende Erkenntnis. Betrachtet man lediglich den Aspekt der kumulierten Belohnung einer Episode, kann eine Lernumgebung mit RL-basiertem Gegenspieler keinen positiveren Effekt auf die Leistungsfähigkeit unter unbekanntem Umgebungsverhalten ausüben. Ebenso ist auch festzuhalten, dass keine Verschlechterung der Leistungsfähigkeit im Testszenario vorliegt. Ein Einfluss eines RL-basierten Gegenspielers auf die Robustheit der kumulierten Belohnung unter unbekannten Umgebungsverhalten kann demnach als nicht gegeben betrachtet werden.

4.1.2 Betrachtung der Belohnungsstabilität

Hypothese 2: *Die Varianz der im Testszenario erzielten kumulierten Belohnung ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig geringer, als die Policy aus dem Training mit regelbasiertem Gegenspieler.*

Mittels der Betrachtung der zweiten Forschungshypothese wird die Stabilität der im Testszenario erzielten kumulierten Belohnung untersucht. Dazu wird die Abweichung einer Belohnung aus

einer Testepisode zum Mittelwert der Belohnungen über aller 100 Testepisoden berechnet. Forschungshypothese 2 stellt dabei die Behauptung auf, dass eine geringere Varianz der Belohnungen im Testszenario vorliegen unter Auswahl der Strategie aus einem Trainingsszenario mit RL basiertem Gegenspieler. Zur Bestätigung dieser Hypothese gilt es zum einen, eine Ungleichheit der unterliegenden Verteilung der Abweichungen festzustellen. Zum anderen ist es notwendig, dass die Verteilung der Abweichungen zur Basis des RL basiertem Gegenspielers signifikant geringer ist.

Zur Evaluation der Forschungshypothese 2 unterstützen die Ergebnisse der Tabelle 7. Aus der Tabelle wird erkenntlich, dass der P-Wert des Signifikanztests zur Gleichheit bei in etwa $1.2200e-22$ liegt. Da sich dieser Wert unter dem gewählten Signifikanzniveau befindet, kann abgeleitet werden, dass nur sehr unwahrscheinlich eine Gleichheit der unterliegenden Verteilung vorherrscht. Die aufgestellte H0 Hypothese wird daher verworfen, und die gegensätzliche H1 Hypothese einer Ungleichheit der beiden Verteilungen wird angenommen. Wird hinzu der P-Wert des Signifikanztests zur Verschlechterung der Belohnungsstabilität von 1.0 betrachtet, kann die folgende Erkenntnis festgestellt werden. Aufgrund des über dem Signifikanzniveau befindlichen P-Wertes, ist die H0 Hypothese zu bestätigen und von einer geringeren Varianz der erzielten Belohnungen auszugehen unter dem Training mit regelbasiertem Gegenspieler. Eine Verringerung der Belohnungsvarianz durch Optimierung unter RL basierten Gegenspieler, wie in der H1 Hypothese behauptet, ist demnach auszuschließen.

Wird diese Erkenntnis im Kontext der Messung der Robustheit betrachtet, werden zwei Effekte deutlich. Zum einen zeigte sich mit der Durchführung des Laborexperiments, dass die Gestaltung des Gegenspielers in der Trainingsumgebung einen Effekt auf die Belohnungsstabilität ausübt. Zum anderen wird deutlich, dass der Einsatz eines optimierten Gegenspielers zu einer Verschlechterung der Belohnungsstabilität führt. Eine RL-Policy, welche unter einem durch verstärkendes Lernen optimierten Gegenspieler trainiert wurde, erzielt demnach geringere Robustheit gemessen an der höheren Varianz der Belohnung.

4.1.3 Betrachtung der Anzahl der Misserfolge

Hypothese 3: *Die im Testszenario erreichte Anzahl von Misserfolgen ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig geringer, als die Policy aus dem Training mit regelbasiertem Gegenspieler.*

Als dritte Forschungshypothese zur Beantwortung der Forschungsfrage wurde die Behauptung aufgestellt, dass die erreichte Anzahl von Misserfolgen im Testszenario durch den Einsatz des RL-basierten Gegenspielers signifikant geringer ist. Zur Untersuchung dessen wird die Verteilung der erfolgreichen und erfolglosen Testepisoden mittels statistischer Signifikanztests ausgewertet. Weisen die Ergebnisse des Tests zur Gleichheit und Verschlechterung der Metrik je einen P-Wert aus, welcher unter dem Signifikanzniveau liegt, so kann die Forschungshypothese bestätigt und von einem positiven Effekt ausgegangen werden.

Betrachtet man zunächst den P-Wert zur Gleichheit der Verteilungen von erfolgreichen und erfolglosen Episoden, liegt dieser entsprechend Tabelle 7 bei in etwa 0.1815. Daher, dass dieser Wert geringer als das Signifikanzniveau ist, wird die H0 Hypothese einer Gleichheit der Verteilungen angenommen. Entsprechend wird die H1 Hypothese einer ungleichen Verteilung der Misserfolge abgelehnt, sodass kein Unterschied in den Erfolgsverteilungen der RL-Policies festzustellen war. Mit der Bestätigung keiner unterschiedlichen Verteilung des Misserfolgs bei einem Training mit RL basiertem Gegenspieler ist die Forschungshypothese 3 abzulehnen.

Das Ablehnen der dritten Forschungshypothese bestätigt, dass kein Effekt durch die Wahl des Trainingsszenarios auf die Robustheit entstanden ist. Wird lediglich die Metrik des Misserfolgs betrachtet, konnte festgestellt werden, dass unter unbekanntem Umgebungsverhalten ähnlich häufig Misserfolge mittels der Strategie erzielt worden sind, welche den RL basierten Gegenspieler im Trainingsszenario verwendete. Insgesamt kann dadurch festgehalten werden, dass keine Verbesserung der Robustheit mit der im Rahmen dieser Arbeit angewendeten Methodik erzielt wurde.

4.2 Vergleich der Robustheit der Strategien aus dem Training mit regelbasierten Gegenspieler und Domain Randomization

Metrik	Gleichheit P-Wert	Verschlechterung P-Wert	Ungleichheit	Verbesserung
Erzielte kumulierte Belohnung	0.48390799726891964	0.7588082897727775	False	False
Abweichung vom Belohnungs-Mittelwert	2.2302103039562726e-05	0.9999889697144758	True	False
Häufigkeit von Misserfolg	0.005412977104604817	0.0027064885523024086	True	True

Tab. 8: Ergebnisse der statistischen Tests aus dem Leistungsvergleich der Modelle aus den Trainingsszenarien 1 und 4

Tabelle 8 weist die Ergebnisse der statistischen Signifikanztests zum Vergleich der Strategieleistungen aus dem Training mit regelbasiertem Gegenspieler und DR aus. Dabei werden in gleicher Struktur zur Tabelle 7 die P-Werte zur Gleichheit und Verschlechterung der Verteilung sowie die Annahme der H1 Hypothesen anhand jeder Metrik dargestellt. Forschungshypothesen und Testhypothesen sind entsprechend dem Vergleich der Policies aus dem Training mit DR und ohne DR anzupassen.

4.2.1 Betrachtung der kumulierten Belohnung

Hypothese 1: *Die im Testszenario erzielte kumulierte Belohnung ist unter Verwendung der Policy aus dem Training mit DR signifikant und zuverlässig höher, als die Policy aus dem Training*

ohne Randomisierung der Umgebung.

Für die Ermittlung des Effekts der DR wird die Forschungshypothese 1 für den Vergleich der Strategieleistungen aus dem Training mit und ohne DR adaptiert. Die Hypothese stellt damit die Behauptung auf, dass die kumulierte Belohnung signifikant höher ist, sofern ein Training mit DR vorliegt. Die Auswertung erfolgt über eine ähnliche Adaptierung der H0 und H1 Hypothesen der Tests zur Gleichheit und Verschlechterung der Ergebnisdaten. Um die adaptierte Hypothese zu bestätigen, gilt es eine Ungleichheit im ersten Signifikanztest und eine Verbesserung im zweiten Test festzustellen.

Werden zur Auswertung die P-Werte aus der Tabelle 8 betrachtet, liegt unter anderem aus dem Test der Gleichheit ein P-Wert von in etwa 0.4839 vor. Da dieser Wert über dem Signifikanzniveau liegt, ist die entsprechende Nullhypothese anzunehmen. Daraus ist festzustellen, dass die H1 Hypothese verworfen wird und folglich von einer gleichen Verteilung der Belohnungen in Abhängigkeit des Trainingsszenarios auszugehen ist.

Für das beschriebene Testszenario konnte der Einsatz von DR im Trainingsszenario demnach keine Verbesserung der Belohnungsverteilung hervorrufen. Die im Rahmen dieser Arbeit verwendete Form der Domänenrandomisierung zeigte somit auch keinen Effekt auf die Robustheit der Strategien gegenüber des unbekannten Testszenarios, wird lediglich die kumulierte Belohnung je Episode betrachtet.

4.2.2 Betrachtung der Belohnungsstabilität

Hypothese 2: *Die Varianz der im Testszenario erzielten kumulierten Belohnung ist unter Verwendung der Policy aus dem Training mit DR signifikant und zuverlässig geringer, als die Policy aus dem Training ohne Randomisierung der Umgebung.*

Mittels der adaptierten zweiten Forschungshypothese wird die Verteilung der quadrierten absoluten Abweichungen vom Mittelwert der Belohnungen über alle 100 Testszenarien untersucht. Dazu wird im ersten Test die Gleichheit der Verteilungen, aus der Verwendung der beiden Strategien, mit und ohne DR, als H0 Hypothese vorausgesetzt. Im zweiten Test wird die Verringerung der Mittelwerts der Abweichungsverteilung aus der Verwendung der Strategie mit DR angenommen.

Tabelle 8 zeigt zum ersten Test der Gleichheit einen P-Wert von in etwa $2.2302e-05$. Dies bedeutet, dass nur zu einer sehr geringen Wahrscheinlichkeit per Zufall die Daten aus einer selben unterliegenden Verteilung gezogen worden sein können. Eine gleiche unterliegende Verteilung wird demnach ausgeschlossen und die H0 Hypothese abgelehnt sowie die H1 Hypothese angenommen. Bezogen auf den zweiten Test liegt ein P-Wert von in etwa 1.0 vor. Folgend kann abgeleitet werden, dass die H0 Hypothese eines höheren Mittelwerts der Verteilung der Abweichungen anzunehmen ist. Entsprechend wird die H1 Hypothese sowie die Hypothese 2 abgelehnt, da die angewendete Form der DR zu einem negativen Effekt auf die Varianz der Belohnungen führt.

Für die Messung der Robustheit ist somit festzuhalten, dass DR im Kontext dieser Arbeit als Teil des Trainingsszenarios, keine Verbesserung der Robustheit erzielte. Dies resultiert daraus, dass keine Verbesserung im Sinne einer stabileren Belohnung erreicht worden ist.

4.2.3 Betrachtung der Anzahl der Misserfolge

Hypothese 3: *Die im Testszenario erreichte Anzahl von Misserfolgen ist unter Verwendung der Policy aus dem Training mit DR signifikant und zuverlässig geringer, als die Policy aus dem Training ohne Randomisierung der Umgebung.*

Mit der dritten adaptierten Forschungshypothese soll zuletzt der Effekt der Domänenrandomisierung auf die Anzahl der Misserfolge untersucht werden. Durch den Test der Gleichheit wird die Existenz eines Effekts überprüft. Sofern dieser Test einen vorhandenen Effekt bestätigt, wird dessen Auswirkung mittels des zweiten Tests hinsichtlich einer Verschlechterung evaluiert. Damit die dritte Forschungshypothese angenommen wird, gilt es einen verringernden Effekt auf die Häufigkeit von Misserfolgen festzustellen. Dazu sind die H0 Hypothesen beider Tests abzulehnen.

Anhand Tabelle 8 lassen sich die Ergebnisdaten der Signifikanztests ablesen. Die Tabelle zeigt einen P-Wert von in etwa 0.0054 zum Test der Gleichheit beider Verteilungen der Misserfolge auf. Aus der erfüllten Bedingung, dass dieser Wert geringer als das Signifikanzniveau ist, kann die H0 Hypothese abgelehnt werden. Folglich wird für den weiteren Testverlauf angenommen, dass die Verteilung der Misserfolge durch die Auswahl des Trainingsszenarios beeinflusst werden konnte. Wird anschließend der P-Wert zur Evaluation des entstandenen Effekts von in etwa 0.0027 betrachtet, können folgende Schlussfolgerungen gezogen werden. Die H0 Hypothese einer höheren Häufigkeit von Misserfolgen kann abgelehnt werden, da der P-Wert im Ablehnungsbereich unterhalb des Signifikanzniveaus liegt. Im Gegensatz dazu kann die H1 Hypothese und eine Verbesserung der Häufigkeit von Misserfolg angenommen werden.

Aus den Annahmen und Ablehnungen der Testhypothesen ergibt sich, dass durch die Anwendung der Domänen Randomisierung eine Verbesserung der Häufigkeit von Misserfolgen erzielen ließ. Wird nur diese Metrik betrachtet, kann entsprechend ausgesagt werden, dass die Strategie durch DR robuster auf unbekannte Umgebungen reagiert, da die Testepisoden häufiger erfolgreich abgeschlossen wurden.

4.3 Vergleich der Robustheit der Strategien aus dem Training mit RL basiertem Gegenspieler und Domain Randomization

Mit Hilfe der Tabelle 9 lassen sich die Ergebniswerte der statistischen Signifikanztests zum Vergleich der Strategien aus RL basiertem Gegenspieler und DR im Training darstellen. Dazu sind

Metrik	Gleichheit P-Wert	Verschlechterung P-Wert	Ungleichheit	Verbesserung
Erzielte kumulierte Belohnung	0.21860488490723795	0.10930244245361898	False	False
Abweichung vom Belohnungs-Mittelwert	1.1343177758713004e-13	0.9999999999999443	True	False
Häufigkeit von Misserfolg	0.09899034302740754	0.9513450789074275	True	False

Tab. 9: Ergebnisse der statistischen Tests aus dem Leistungsvergleich der Modelle aus den Trainingsszenarien 3 und 4

die P-Werte der Tests und Wahrheitswerte der Testhypothesen je untersuchter Metrik angegeben.

4.3.1 Betrachtung der kumulierten Belohnung

Hypothese 1: *Die im Testszenario erzielte kumulierte Belohnung ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig höher, als die Policy aus dem Training mit randomisierter Umgebung.*

Zuerst wird die Hypothese 1 einer signifikant höheren kumulierten Belohnung, durch den Einsatz von RL zum Optimieren eines Gegenspielers gegenüber der angewendeten Form von DR, untersucht. Die H0 Hypothese im ersten Test repräsentiert die Behauptung, dass die Daten der Belohnungen einer gemeinsamen Verteilung entstammen. Dass die Belohnungsdaten, welche durch die Strategie mit RL basiertem Gegenspieler erzeugt worden sind, einer höheren Verteilung entstammen, wird im zweiten Test als H0 Hypothese überprüft. Entsprechend gegensätzliche Annahmen werden in beiden Signifikanztests als H1 Hypothese verwendet.

Der erste P-Wert der Tabelle 9 in der Zeile zur kumulierten Belohnung liegt mit in etwa 0.2186 über dem Signifikanzniveau von 10 %. Aus diesem Grund wird die H0 Hypothese bestätigt, dass die Daten einer einheitlichen Wahrscheinlichkeitsverteilung entstammen. Entsprechend ist die H1 Hypothese abzulehnen und auch die obige adaptierte Forschungshypothese zu verwerfen. Daraus entfällt eine weitere Betrachtung des P-Wertes des zweiten Tests.

Aus der Überprüfung der Testergebnisse wird deutlich, dass RL keinen verbessernden Effekt auf die Belohnung ausüben konnte als die angewendete Form der DR. Daraus kann ableitet werden, dass die Strategie aus dem Training mit RL basiertem Gegenspieler nicht robuster agieren konnte, indem eine bessere Belohnung im abweichenden Testszenario erzielt wird.

4.3.2 Betrachtung der Belohnungsstabilität

Hypothese 2: *Die Varianz der im Testszenario erzielten kumulierten Belohnung ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig geringer, als die Policy aus dem Training mit randomisierter Umgebung.*

Mittels der nächsten Forschungshypothese soll die Leistungsstabilität, der aus dem Training mit RL basiertem Gegenspieler und mit DR entstandenen Strategien, verglichen werden. Zur Untersuchung dieser Metrik werden die Daten der Abweichungen der Strategieergebnisse betrachtet. Als erste H0 Hypothese wird die Annahme eines einheitlichen Mittelwerts, als zweite H0 Hypothese die Annahme eines größeren Mittelwerts der Verteilung eingesetzt.

Aus Tabelle 9 geht ein P-Wert von in etwa $1.1343e-13$ hervor. Folgend daraus ist die H0 Hypothese abzulehnen und festzuhalten, dass ein Einfluss des Trainingsszenarios auf die Verteilung der Abweichungen besteht. Wird dieser Einfluss mittels zweitem Test genauer untersucht, ist dabei der P-Wert von in etwa 1.0 zu betrachten. Daraus geht hervor, dass die Wahl eines RL basierten Gegenspieler anstelle der DR im Trainingsszenario einen negativen Effekt auf die Belohnungsstabilität ausübt.

Entsprechend wird die Forschungshypothese 2 in diesem Kontext verworfen, da kein Verbesserungseffekt durch einen RL basierten Gegenspieler festzustellen war. Im Gegenteil konnte gezeigt werden, dass ein RL basierter Gegenspieler im Training zu einer weniger robusten Strategie führt, als wenn DR eingesetzt würde.

4.3.3 Betrachtung der Anzahl der Misserfolge

Hypothese 3: *Die im Testszenario erreichte Anzahl von Misserfolgen ist unter Verwendung der Policy aus dem Training mit RL basiertem Gegenspieler signifikant und zuverlässig geringer, als die Policy aus dem Training mit randomisierter Umgebung.*

Anhand einer dritten Forschungshypothese soll schlussendlich die dritte Metrik der Robustheit und dessen Einfluss durch das gewählte Trainingsszenario erforscht werden. Wie in der bisherigen Vorgehensweise wird zunächst die Ähnlichkeit der beiden Datenverteilungen zum Misserfolg der Testepisoden analysiert. Anschließend ist bei einer vorliegenden Ungleichheit der Effekt des RL basierten Gegenspielers gegenüber der DR herauszustellen. Zeigt sich aus dieser Gegenüberstellung, dass eine Verbesserung durch das Training mittels RL optimiertem Gegenspieler erzielt wurde, ist die dritte Forschungshypothese anzunehmen.

Werden die Testwerte aus Tabelle 9 betrachtet, gilt es zunächst den ersten P-Wert von in etwa 0.0990 mit dem Signifikanzniveau zu vergleichen. Die Ablehnung der H0 Hypothese und Annahme einer unterschiedlichen grundlegenden Wahrscheinlichkeitsverteilung folgt aus dem geringer als 0.1 bemessenen P-Wert. In der Auswertung des zweiten Tests wird die H0 Hypothese angenommen, da dessen P-Wert mit in etwa 0.9513 über dem Signifikanzniveau liegt. Daraus

hervorgehend ist festzustellen, dass die Gültigkeit der H1 Hypothese und dementsprechend ein verringernder Effekt auf die Häufigkeit des Misserfolgs durch RL optimierten Gegenspielers auszuschließen ist.

Mit der Ablehnung der dritten Forschungshypothese lässt sich ein positiverer Effekt des Einsatzes von DR auf die Robustheit festhalten, als mittels des Einsatzes eines RL basierten Gegenspielers. Der positivere Effekt durch DR wirkt sich im Sinne einer höheren Erfolgsrate auf die Robustheit aus.

4.4 Beantwortung der Forschungsfrage

Aus der erkannten Problemstellung des Sim2Real Transfers und der Robustheit von RL-Algorithmen in besonders kritischen Bereichen wie der Robotik und dem Drohnenflug, wurde folgende Forschungsfrage gestellt.

Inwiefern kann durch den Einsatz eines mittels RL trainierten Gegenspielers die Robustheit einer optimierten Policy verbessert werden?

Im Rahmen dieser Arbeit wurde anhand von drei Forschungshypothesen der Effekt eines durch verstärkenden Lernens optimierten Gegenspieler auf die Robustheit von RL Modellen untersucht. Dazu wurde eine Simulationsumgebung entwickelt, in welcher sich eine RL Strategie als Spieler und als Gegenspieler einsetzen lässt. Mehrere Strategien aus verschiedenen Trainingsszenarien wurden in einer abgewandelten unbekannten Variation der Simulationsumgebung getestet, um so deren Robustheit zu untersuchen. Dabei wurde je Strategie der Erfolg der Testepisode, die erreichte kumulierte Belohnung und deren Abweichung zum Mittelwert aller Episoden gemessen und ausgewertet. Über diese drei Metriken wurde die Robustheit im Sinne der Effektivität, Effizienz, und Zuverlässigkeit des RL Modells im unbekannten Testszenario betrachtet.

Die Auswertung der Messdaten des Laborexperiments brachte in diesem Kapitel mehrere Schlussfolgerungen und Erkenntnisse, anhand dieser die Forschungsfrage zu beantworten ist. Dazu ist besonders der erste Vergleich der Leistungsdaten zwischen den Strategien unter dem Training mit RL basiertem und regelbasiertem Gegenspieler aussagekräftig. In Bezug auf die erste sowie dritte Metrik konnte weder eine Verbesserung noch eine Verschlechterung durch die Wahl des Trainingsszenarios nachgewiesen werden. Die zweite Metrik der Varianz konnte durch den Einsatz des RL basierten Gegenspielers lediglich verschlechtert werden. Wird das Untersuchungsergebnis zusammengefasst, kann daraus folgende Beantwortung der Forschungsfrage vorgenommen werden.

Durch den Einsatz eines mittels RL trainierten Gegenspielers kann die Robustheit einer optimierten Policy nicht im Sinne eines gütevolleren, stabileren oder erfolgreicheren Verhaltens in unbekannter Umgebung verbessert werden.

Weiterhin wurde eine Form der in der Forschungsliteratur bekannten Methodik der Domänenrandomisierung angewendet. Durch die Anwendung dieser Methodik konnte eine bekannte Art und Weise die Robustheit von RL Modellen zu erhöhen genutzt werden, um ein Benchmarking nicht nur anhand keiner vorhandenen Methodik durchzuführen. Die Anwendung von DR im Trainingsszenario in Form von Wind konnte schlussendlich gegenüber der Strategie ohne Randomisierung nicht die gleiche Belohnungsstabilität gewährleisten. Im Gegenzug konnte jedoch die Häufigkeit von Misserfolg verringert werden.

Im Kapitel 4.3 wurde schlussendlich der Effekt der DR ins Verhältnis zum Effekt des RL basierten Gegenspielers gesetzt. Dabei ist zusätzlich die Erkenntnis erzielt worden, dass die DR zumindest im Sinne Belohnungsstabilität und Erfolgsquote einen stärkeren Verbesserungseffekt als die Strategie mit RL basiertem Gegenspieler hervorrufen konnte.

5 Fazit, Reflexion und Forschungsausblick

5.1 Fazit

Künstliche Intelligenz und maschinelles Lernen zeigen heutzutage in immer mehr Lebens- und Wirtschaftsbereichen hohe Potenziale zur Übernahme und Automatisierung von Abläufen. Ein wichtiges Phänomen zur Automatisierung ist dabei der Einzug von künstlicher Intelligenz in die Robotik zur Steuerung von Maschinen und Drohnen. Deren Anwendung bleibt häufig eine Herausforderung aufgrund fehlender Datengrundlage und dem hohen manuellen Aufwand die Daten entsprechend zu kennzeichnen. Durch den Einsatz einer Simulationsumgebung und verstärkendem Lernen konnte im Rahmen dieser Arbeit gezeigt werden, dass auch ohne manuelle Kennzeichnung Modelle des maschinellen Lernens optimiert werden können. Erkannt wurde auch, dass Simulationsumgebungen immer eine gewisse Diskrepanz zur realen Welt aufweisen, weshalb es unabdingbar ist, die Robustheit von RL Modellen zu überprüfen. Daraus ergab sich für diese Arbeit die Zielstellung den Effekt eines RL basierten Gegenspielers, auf die Robustheit einer optimierten Strategie mittels eines Laborexperiments zu untersuchen. Anschließend galt es, den untersuchten Effekt mit einer aktuellen Methodik zur Erhöhung der Robustheit, wie der DR zu vergleichen.

Die Ergebnisse zeigen, dass die Anwendung des RL basierten Gegenspielers sowie der DR, zwar nicht in jeder Hinsicht die Robustheit von RL Modellen verbessern, jedoch die DR teilweise positive Effekte auf einzelne Teilaspekte wie Stabilität und Erfolgsrate ausüben können. Im weiteren Verlauf soll deshalb die im Rahmen dieser Arbeit angewendete Forschungsmethodik kritisch reflektiert, sowie ein Ausblick auf weitere Forschungsfragen gegeben werden.

5.2 Reflexion der Forschungsmethodik

Zur Untersuchung der in dieser Arbeit gestellten Forschungsfrage wurde die Methode eines Laborexperiment durchgeführt. Dabei galt es Messdaten zur Robustheit der verschiedenen RL Modelle aus unterschiedlichen Trainingsszenarien zu erheben. Da keine Trainingsumgebung und Drohnen in der echten Welt zur Verfügung standen, wurde eine Simulationsumgebung verwendet. Für das Training ist zunächst regelbasiertes Verhalten entwickelt worden, welches mittels imitierenden Lernen zur Modellstruktur des verstärkenden Lernens übertragen wurde. Nach einem zweiten Optimierungsschritt mit RL wurden die Modelle in einer Abwandlung der Simulation auf ihre Leistung hin geprüft.

Mit der Durchführung dieser Methodik bestand die Herausforderung den hohen implizierten Entwicklungsaufwand umzusetzen, da viele Komponenten, welche für die Überprüfung benötigt waren, zum Zeitpunkt kein Teil der Forschungsliteratur waren. Die Entwicklung begann mit der für den Anwendungsfall zugeschnittenen Simulationsumgebung. Zusätzlich waren in dieser

Umgebung insgesamt vier Strategien anhand von imitierendem Lernen im ersten Schritt und verstärkendem Lernen im zweiten Schritt zu optimieren. Weiterhin bedarf das Laborexperiment einem Testkonzept zur Messung und Auswertung der in der Forschung nicht eindeutig definierten Robustheit.

Aus der Anwendung eines Laborexperiments und der eigenen Entwicklung der meisten benötigten Komponenten ergeben sich vielerlei Einschränkungen und Möglichkeiten zur Methodenoptimierung. Durch die Wahl des Laborexperiments konnte zwar erfolgreich die Robustheit bemessen werden, jedoch ist dadurch auch ein beschränkter Bezug zur Realität im Vergleich zu z. B. einem Feldexperiment gegeben. Weiterhin wurde mit der Durchführung des Experiments in einer Simulation, dem Testszenario selbst, um eine nicht bemessene Diskrepanz zur Realität erweitert und damit die Messung der Robustheit beeinflusst. Im Trainingsprozess der Modelle konnte wiederum dank der niedriger frequentierten Simulation, bzw. der mehrfachen Ausführung einer gewählten Aktion, mit geringem Aufwand stark unkorrelierte Trainingsdaten erzeugt werden, was ohne Simulation zu einer großen Herausforderung führe. Dennoch kann es lohnenswert sein Optimierungspotenzial im Trainingsprozess zu suchen, da von dessen Qualität auch die im Testszenario gemessenen Leistungen der RL Modelle abhängen. Zusätzlich könnte der Prozess zur Optimierung der Modelle nur unter einer beschränkten Menge von Trainingsdaten ausgeübt werden, woraus auch ein Einfluss auf das Verhalten der Modelle bewirkt wurde. Um die gemessenen Effekte des RL basierten Gegenspielers in eine Relation, zu in der Forschungsliteratur vorhandenen Methoden zu setzen, wurde der populärste Ansatz DR angewendet. Die in dieser Arbeit entwickelte Form der DR erzeugte zwar selbst nicht in jeder Hinsicht eine Verbesserung der Robustheit, konnte aber dennoch für einen Vergleich zu bestehenden Ansätzen eingesetzt werden.

Insgesamt ließ sich die gewählte Methodik dieser Arbeit, trotz hohem Entwicklungsaufwand und demnach Einschränkungen in der allgemeinen Gültigkeit, zur Untersuchung der Robustheit von RL Algorithmen einsetzen. Betrachtet man die Quantität von in der Literatur standardisierten Konzepten, konnte mit dieser Arbeit viel beispielhafte Entwicklung geleistet werden, um die notwendige Forschung an der Robustheit von RL Algorithmen im Robotik Bereich und im Drohnenflug voranzubringen.

5.3 Ausblick für weitere Forschung

Soll innerhalb der letzten Sektion final nun ein Ausblick für weitere Forschung gegeben werden, ist ein bedeutsamer Punkt die Standardisierung von Definitionen und Messkonzepten zur Untersuchung der Robustheit von Modellen des maschinellen Lernens. Unter einem einheitlichen Verständnis der Robustheit und entwickelten Metriken könnte die Forschung zu dessen Verbesserung beschleunigt und deren Ergebnisse vergleichbarer gemacht werden. Darauf aufbauend würde es den aktuellen Stand der Forschung bereichern, ein Testszenario in der echten Welt zu implementieren, um die Simulationsdiskrepanz zur realen Welt zu verhindern. Anhand dessen ließe sich die Aussagekraft von Messungen und Ergebnisses deutlich steigern. Da die Anwendung

in der echten Welt nicht für jedes Untersuchungsszenario umsetzbar sein kann, bleibt dennoch die Entwicklung und Verbesserung von Simulation ein wichtiges Forschungsgebiet. Eine wichtige Simulationseigenschaft im Rahmen dieser Arbeit war die Simulations-, bzw. Handlungsfrequenz des Agenten, um trotz geringerer Datenpunkte einen diversen Datensatz zu erzielen. Wird diese Eigenschaft betrachtet, ergeben sich zwei neue Forschungsrichtungen, welche die Frage nach der leistungsoptimalen, also der effektivsten sowie nach der ressourcenoptimalen, also der effizientesten Simulationsfrequenz stellen.

Insgesamt bleibt es damit weiter notwendig, das Verhalten von RL Modellen in unbekannten Umgebungen zu untersuchen, um die Robustheit der Modelle zu evaluieren und eine fehlerfreie Übernahme von dynamischen Prozessen in der Robotik und dem Drohnenflug zu garantieren.

Anhang

Anhangverzeichnis

Anhang 1: Messdaten des Laborexperiments

Anhang 1/1: Messdaten des Modells aus dem ersten Trainingszenario

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
0	4973	-119446.26307111187	387329645.39367837	1
1	2135	-73579.65118847974	4296449072.243052	1
2	4315	-68451.57618280343	4995009208.997931	0
3	3021	-58856.40652218178	6443360996.138419	1
4	2200	-106828.42153534402	1043195281.3534079	1
5	13035	-219399.8086845268	6443730995.447552	1
6	1425	-99294.08419905338	1586657617.6864533	1
7	370	-20456.49584066878	14082677940.965357	1
8	2767	-66373.5605798862	5293056438.810909	1
9	394	-70932.35303882814	4650503773.3509865	1
10	1377	-77663.76882989446	3777723287.2659354	1
11	198	-12715.698783642225	15979805766.243288	1
12	6360	-217357.82021610026	6120068230.348927	1
13	776	-81468.55281955525	3324491373.038729	1
14	329	-18654.276777649084	14513666262.915075	1
15	160	-57678.73562351338	6633812483.442872	1
16	3646	-167379.6638554781	798215542.4643234	1
17	386	-24654.250630269184	13104000107.339048	1
18	936	-71691.87001738523	4547490722.911959	1
19	1868	-117854.37762945828	452522559.43645215	1
20	15220	-247291.44889176913	11699557680.638937	1
21	1854	-91688.46036564803	2250410798.709002	1
22	1949	-82100.30706836819	3252038605.015034	1
23	5635	-146728.2040420022	57778982.90688015	1
24	3915	-100710.63153067631	1475813929.5787473	1
25	27149	-288748.7499041229	22386681429.791836	1
26	17351	-773888.6594350344	402922421075.58545	1
27	18452	-266545.7001431137	16235536545.257801	1
28	4116	-128943.76280506424	103697408.76048857	1
29	4245	-146624.52291664132	56213520.62880017	1
30	693	-24905.009835843994	13046652818.56774	1
31	4204	-108705.23049853576	925481338.0186319	1
32	3839	-105973.55113709459	1099148205.5429835	1
33	13880	-102975.14464771489	1306953411.2076669	1

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
34	1575	-114665.39035433184	598368158.0815635	1
35	2368	-104427.56060357616	1204047990.15683	1
36	1223	-83398.23783177527	3105689947.2140727	1
37	33802	-735697.9326871325	355896931097.4542	1
38	11678	-280712.0211739327	20046330887.263393	1
39	4613	-103340.52131554208	1280668855.1155045	1
40	4939	-113005.67033766753	682321526.4713739	1
41	557	-34873.86419424162	10868706998.671867	1
42	2921	-111742.82137255678	749890789.2662112	1
43	1557	-73030.91534860493	4368686493.177479	1
44	1522	21140.63597449921	25685700803.03947	0
45	1954	-90232.52886299146	2390664933.586656	1
46	5999	7264.712146483675	21430520288.727715	0
47	281	-16075.567283123166	15141644085.286205	1
48	2070	12393.512751061113	22958452229.06197	0
49	19418	-473654.72629417136	111908829586.97322	1
50	15689	-276474.9519422902	18864472190.126606	1
51	5238	-165381.53396681393	689302902.5921499	1
52	6524	-105787.99825142126	1111486055.0295568	1
53	6561	-205898.10139098376	4458385954.424092	1
54	1910	-92032.12884181371	2217922676.220357	1
55	22148	-445050.6391527784	93589300360.93394	1
56	1472	-79755.73736719266	3524941515.1609735	1
57	263	-15600.021604753363	15258903340.526737	1
58	2567	-61654.41882784868	6001993902.647814	1
59	1366	-51713.97716429973	7641028741.157966	1
60	4928	-95423.07511540131	1910029140.4926941	1
61	304	-13998.380734446198	15657160165.085396	1
62	12421	-71413.81905101251	4585068816.495214	0
63	311	-14710.14499937111	15479542677.704636	1
64	2411	-90587.72499755406	2356056875.3584223	1
65	3560	-156842.56739543795	313842913.0092971	1
66	3323	-88666.1266710266	2546295222.808571	1
67	2216	-102406.93034675243	1348360230.3010497	1
68	1168	-43261.54605595228	9190176683.61268	1
69	2261	-111291.51604653947	774811676.7153043	1
70	1803	26669.781018645506	27488557763.682777	0
71	5358	-82036.28655387607	3259344454.3564596	1

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
72	6518	-35354.203289041674	10768784053.523808	0
73	100662	-485724.2106050032	120129657406.35081	1
74	1642	-112009.36921540477	735363473.381946	1
75	23079	-639349.6775980042	250222771934.0961	0
76	2505	-114534.05479872528	604810753.5479158	1
77	9151	-159660.46156868307	421624880.9532151	1
78	5556	-184610.73783700686	2068774476.6806357	1
79	2591	-113013.74205295392	681899904.4741982	1
80	1917	-48611.60139315391	8193029287.295692	1
81	1033	-65091.71325664009	5481217059.850006	1
82	3035	-151284.32248952115	147801577.73110348	1
83	6333	-248964.96189519798	12064387699.585102	1
84	143	-6270.742974679765	17650773145.23172	1
85	2408	35063.594815062024	30342347738.638252	0
86	3609	-120362.11771286349	352119128.51359135	1
87	2907	-95234.46438855604	1926550755.7024877	1
88	6909	-254854.3644192685	13392833228.600346	1
89	606	38778.579022998085	31650379131.7725	0
90	4813	-111878.77036680779	742463580.4718969	1
91	2321	-93649.80171388677	2068171495.4884348	1
92	88171	-910404.3026574675	594868746594.3689	1
93	3108	-141655.67094119178	6394402.950060642	1
94	10312	-330954.2408898612	36797707508.61226	1
95	3768	-159059.9077657962	397322595.23379785	1
96	5018	-73322.6603574858	4330205228.82047	1
97	3678	-92350.12255671938	2188072078.606148	1
98	6036	-146973.86740964398	61574030.14072773	1
99	1573	-87413.84910489814	2674245349.154552	1

Anhang 1/2: Messdaten des Modells aus dem dritten Trainingszenario

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
0	1653	-92347.12814609804	3531695.3212134503	1
1	1511	-79949.83767800787	203820479.5453756	1
2	2686	-138673.8482117061	1975574881.0166585	1
3	2076	-111671.73332763254	304339352.0974004	1

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
4	1251	-79799.73697424782	208128856.4756929	1
5	1832	-77824.71257469118	269015635.0969574	1
6	2081	-100702.72327265213	41942650.71640244	1
7	2037	-92548.82473529059	2814287.888074833	1
8	1546	-80243.20957471852	195529857.18900484	1
9	2100	-104374.48557198321	102983464.62660627	1
10	1739	-99047.36821871552	23241650.883909702	1
11	2437	-114813.9047312334	423844993.70215905	1
12	2006	-108572.06517066105	205797860.17773557	1
13	1564	-89733.93105526718	20182355.199427325	1
14	2415	-125882.56288775698	1002112099.2908063	1
15	2352	-123751.17987700357	871712114.2965164	1
16	2041	-107225.85252142692	168985540.21331608	1
17	1744	-89764.21221744212	19911197.26122899	1
18	2137	-119053.14423147225	616366798.194118	1
19	1519	-87000.82563168688	52209050.37311575	1
20	1404	-81501.43041543056	161925071.80742893	1
21	1744	-103097.75428849243	78700772.92367826	1
22	1667	-89130.51424513668	25968140.080724362	1
23	1642	-6907.798450224385	7624539692.27876	0
24	1343	-83084.15449344112	124149828.33482867	1
25	1683	-90356.76164617151	14974168.158408964	1
26	1878	-111838.56754250827	310188139.8346791	1
27	1852	-91691.71347423723	6424679.780870929	1
28	1378	5312.971207702023	9908088148.22225	0
29	1950	-91346.0914973505	8296227.06670481	1
30	1751	-14833.500644185704	6303233846.218952	0
31	2121	-122840.91833738518	818790162.991811	1
32	2027	-107247.44465804833	169547377.96857768	1
33	2222	-100644.25859708953	41188797.57156858	1
34	1879	-99075.08534082912	23509665.371303875	1
35	1330	-89172.55540681427	25541432.896430615	1
36	1735	-81005.49859622098	174792463.41600767	1
37	2410	-122139.76801712476	779155629.5412201	1
38	2002	-110744.31609087711	272841265.26849467	1
39	2110	-104030.87721611398	96127603.29594047	1
40	2368	-127586.09074176587	1112868387.3311455	1
41	1779	-88839.11463014259	29022937.168479417	1

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
42	1563	-92566.27084313083	2756057.6339688073	1
43	2262	-120846.561556859	708632539.242438	1
44	1871	-110632.23542961734	269151151.37241375	1
45	1695	-90088.13234400195	17125331.02451682	1
46	1920	-101160.33136423607	48079283.80531415	1
47	1467	-92474.14392801689	3070431.7574159317	1
48	2328	-114691.48989933486	418819549.35632735	1
49	1520	-88746.39566596689	30030542.622630715	1
50	1829	-105687.92705010656	131366403.37201113	1
51	1068	-80491.15858977714	188657095.01254028	1
52	2549	-115674.49520694178	460020415.71157074	1
53	1460	-80996.95173668266	175018530.98770744	1
54	1927	-112770.6757002326	343889839.35606617	1
55	1422	-76259.59982803596	322806220.2858289	1
56	1930	-94052.10799082342	30380.729886693778	1
57	1949	-98282.0069838972	16447877.610967359	1
58	1672	-77253.73157348904	288071768.1267542	1
59	1537	-105266.57754039323	121885328.48884536	1
60	2075	-113502.44137400572	371565436.4602714	1
61	1522	-95541.05480403255	1728294.434154625	1
62	1435	-88259.02829929876	35609628.59562785	1
63	1133	-70536.33448700237	561219615.2207029	1
64	1035	-79043.81093353551	230511274.29952642	1
65	2223	-103816.50438886588	91969935.72505565	1
66	1784	-118544.62175341861	591375487.1345174	1
67	1037	28673.595925229718	15104411131.71113	0
68	1232	14926.938427057503	11914453184.447874	0
69	2659	-151453.26593478158	3274913191.3775845	1
70	1688	-94274.73392669848	2335.329568483486	1
71	2299	-113496.06004772757	371319463.8736459	1
72	1693	-99589.95555226665	28767635.062101834	1
73	1959	-104666.89017587205	109003653.88725439	1
74	1338	-76191.43750774984	325260185.14656883	1
75	1799	-107730.14072747264	182350779.23788708	1
76	1726	-94312.72168004303	7449.934155063287	1
77	1637	-79745.42987692845	209698747.06503746	1
78	1896	-98852.78875571876	21403392.61458239	1
79	1644	-84666.40512403368	91393667.97030306	1

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
80	2133	-101018.20279461997	46128467.309442826	1
81	2230	-94581.92219002724	126389.85654766404	1
82	1691	-77884.60688015005	267054486.03175777	1
83	2122	-119640.7616723019	645889338.0699226	1
84	1324	-70624.74418543525	557038566.8651519	1
85	2216	-138310.5053513642	1943407579.4214218	1
86	2176	-95513.39386064193	1656330.857855494	1
87	1864	-88412.25104209002	33804429.02643386	1
88	1786	-92727.62776697357	2246344.2221504324	1
89	1486	-72737.9740066259	461752824.6673936	1
90	2496	-119080.5852518373	617730093.1155047	1
91	1948	-101354.26852030956	50806385.91784977	1
92	1958	-110046.45887832022	250273988.31011304	1
93	2006	-101985.01821046043	60196021.86220878	1
94	1059	-75028.5649990723	368557201.95760334	1
95	1431	-88123.49672449603	37245534.32400263	1
96	2109	-96440.1698084757	4900738.338531612	1
97	2137	-135963.8132262433	1742010938.3431518	1
98	1827	-104948.82027461141	114970110.4264485	1
99	1689	-114871.268292174	426210228.4491004	1

Anhang 1/3: Messdaten des Modells aus dem vierten Trainingszenario

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
0	1455	-139572.38780625432	744631218.091311	1
1	668	-73773.32807283304	1483106980.1097713	1
2	401	-31868.889712731754	6466663313.595069	1
3	2566	-171421.8210663553	3497227937.9511185	1
4	568	-84374.66627949841	778956366.509154	1
5	1369	-49952.79194271706	3885236353.6785493	1
6	4053	-155445.12575918084	1862843407.4631445	1
7	2124	-105875.03948737303	41080621.012135595	1
8	2559	-100702.63656066287	134138543.29590714	1
9	3690	-189124.45404525608	5904385287.848514	1
10	10912	-435051.3296926795	104178454704.00514	1
11	4692	-184106.58067775535	5158417582.247667	1

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
12	486	-60047.127430950866	2728738504.0162582	1
13	10080	-66548.61530662145	2091767130.9216878	0
14	3312	-122858.94017941065	111819714.04409674	1
15	391	-19717.436219625506	8568653165.170857	1
16	743	-83506.97471680315	828143433.4039263	1
17	1878	-103106.10932391894	84242049.1094118	1
18	1016	-85023.93260886864	743136139.7257056	1
19	1107	-104535.49176005676	60046447.97649458	1
20	1532	-42241.6360096946	4905996642.901692	1
21	3524	-117124.9053152941	23429948.83397716	1
22	203	-58907.64565455435	2849083890.813327	1
23	4397	-121775.39123662916	90077850.21502052	1
24	3493	-151743.9720871852	1557053403.802695	1
25	1468	-94328.55530234546	322414372.6089292	1
26	1667	-103489.34328833577	77354008.9553766	1
27	2227	-203299.81129311383	8283794885.218329	1
28	1663	-80731.42288443842	995593903.6152023	1
29	687	-88827.11715052626	550246749.4849268	1
30	932	-79522.16509663286	1073367710.2451876	1
31	1047	-98481.09856043328	190532678.71454552	1
32	3539	-297297.80747770553	34229940196.73022	1
33	6516	-265886.76237301354	23593668499.628105	1
34	1474	-87107.15771247805	633896352.9076811	1
35	2945	-127935.2128341831	244946187.0902154	1
36	289	-58460.85077567921	2896980487.571486	1
37	2454	-92160.30137319927	404981602.5197421	1
38	1126	-85634.7327863673	710207751.4371468	1
39	357	-74902.30855478215	1397424953.3305738	1
40	1223	-89556.25027933848	516571338.7551463	1
41	2807	-187277.46219272964	5623950966.35869	1
42	3295	-128034.55609537168	248065650.60652694	1
43	960	-68842.4879640358	1887204593.47866	1
44	370	-63979.30353180085	2333387762.3603206	1
45	1259	-97077.13882747927	231262497.9103693	1
46	247	-65317.699554064646	2205876218.231565	1
47	3634	-247874.40333470656	18384633797.501015	1
48	2321	-147716.59825833945	1255436699.5987327	1
49	1994	-154343.7654882401	1768985508.1980054	1

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
50	758	-75267.45162457463	1370258618.5890083	1
51	803	-96696.74430428469	242976758.29161587	1
52	595	-79127.62418066339	1099375501.36442	1
53	386	-20780.716701622558	8372934319.573498	1
54	928	-74235.79261964842	1447700790.8477635	1
55	763	-83532.89926826739	826652018.893257	1
56	1831	-101537.94911802994	115487411.8070363	1
57	1274	-122700.53145025088	108494626.19302548	1
58	802	-43329.848793566314	4754737865.53457	1
59	10117	-227067.8234075841	13175221415.877636	1
60	1728	-119408.46281307109	50751471.98364446	1
61	849	-99708.72749570434	158148949.33619174	1
62	1246	-82510.6863451565	886477364.0111622	1
63	288	-15740.96284696191	9320646084.949207	1
64	362	-78500.3086232856	1141368619.136103	1
65	541	-66739.89268245646	2074307254.748079	1
66	57	-51426.15702325551	3703732563.6461587	1
67	523	-69521.75031680161	1828649003.8849342	1
68	471	-63306.9524595423	2398795860.542682	1
69	1451	-57554.22148378302	2995398578.566658	1
70	677	-82659.93849905756	877612041.6923112	1
71	888	-96043.9975492387	263752493.1735646	1
72	1402	-105520.38171801696	45752701.926200815	1
73	184	-56333.05585696981	3130559186.518706	1
74	1608	-85394.50950372484	723069226.6831765	1
75	2691	-100272.7277743975	144281616.80195513	1
76	5126	-241840.43650942043	16784752065.95225	1
77	371	-29223.30313552408	6899155127.873301	1
78	168	-11817.715457175556	10093565970.657442	1
79	565	-73200.09634163367	1527587175.879315	1
80	2239	-134441.5242029784	490935667.96431106	1
81	2165	-121493.00241384219	84797325.64818572	1
82	202	-54308.91006303916	3361163935.5513124	1
83	1288	-87881.2787390615	595515064.1592721	1
84	5135	-173571.8733679027	3756147518.258564	1
85	257	-66944.76429320117	2055687650.9985905	1
86	61548	-601478.1387180209	239310459136.43716	1
87	823	-84664.817011253	762844462.0908961	1

Episode	Episodenlänge	kumulierte Belohnung	quadrierte absolute Abweichung	Episodenerfolg
88	272	-59390.91893604745	2797726271.2214975	1
89	2467	-192004.36119959655	6355263272.845969	1
90	1043	-74929.26090746139	1395410606.1120675	1
91	2149	-105531.93105444917	45596594.17253704	1
92	5500	-476829.080297996	132892783048.83955	1
93	907	-62796.06332880723	2449101016.743401	1
94	916	-96615.1085959957	245528450.4627318	1
95	984	-79178.95030215883	1095974512.5705953	1
96	45	-52242.449477733804	3605042556.3984103	1
97	424	-76972.4110787145	1246940521.9209971	1
98	2467	-50969.970664024964	3759466121.3885736	1
99	2836	-118707.5851452389	41256587.034520954	1

Literaturverzeichnis

ACM Digital Library (2/28/2023): ACM Digital Library. URL: https://dl.acm.org/.

Alghonaim, R./Johns, E. (2021): Benchmarking Domain Randomisation for Visual Sim-to-Real Transfer. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, S. 12802–12808. ISBN: 978-1-7281-9077-8. DOI: 10.1109/ICRA48506.2021.9561134.

Anu, M. (1997): Introduction to modeling and simulation. In: *Proceedings of the 29th conference on Winter simulation - WSC '97*. New York, New York, USA: ACM Press. DOI: 10.1145/268437.268440.

Arulkumaran, K./Deisenroth, M. P./Brundage, M./Bharath, A. A. (2017): Deep Reinforcement Learning: A Brief Survey. In: *IEEE Signal Processing Magazine* 34.6, S. 26–38. DOI: 10.1109/MSP.2017.2743240.

Attia, A./Dayan, S. (2018): Global overview of Imitation Learning. arXiv: 1801.06503 [stat.ML].

Ayala, A./Cruz, F./Campos, D./Rubio, R./Fernandes, B./Dazeley, R. (2020): A Comparison of Humanoid Robot Simulators: A Quantitative Approach. In: *2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, S. 1–6. DOI: 10.1109/ICDL-EpiRob48136.2020.9278116.

Balakrishna, A./Thananjeyan, B./Lee, J./Li, F./Zahed, A./Gonzalez, J. E./Goldberg, K. (2020): On-Policy Robot Imitation Learning from a Converging Supervisor. In: *Proceedings of the Conference on Robot Learning*. Hrsg. von Leslie Pack Kaelbling/Danica Kragic/Komei Sugiura. Bd. 100. Proceedings of Machine Learning Research. PMLR, S. 24–41. URL: https://proceedings.mlr.press/v100/balakrishna20a.html.

Bellman, R. (1966): Dynamic Programming. In: *Science* 153.3731, S. 34–37. DOI: 10.1126/science.153.3731.34. eprint: https://www.science.org/doi/pdf/10.1126/science.153.3731.34. URL: https://www.science.org/doi/abs/10.1126/science.153.3731.34.

Bharadhwaj, H./Wang, Z./Bengio, Y./Paull, L. (2019): A Data-Efficient Framework for Training and Sim-to-Real Transfer of Navigation Policies. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. DOI: 10.1109/icra.2019.8794310.

Brockman, G./Cheung, V./Pettersson, L./Schneider, J./Schulman, J./Tang, J./Zaremba, W. (2016): OpenAI Gym. In: *CoRR* abs/1606.01540. arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540.

Canese, L./Cardarilli, G. C./Di Nunzio, L./Fazzolari, R./Giardino, D./Re, M./Spanò, S. (2021): Multi-Agent Reinforcement Learning: A Review of Challenges and Applications. In: *Applied Sciences* 11.11, S. 4948. DOI: 10.3390/app11114948. URL: https://www.mdpi.com/2076-3417/11/11/4948.

Chen, X./Hu, J./Jin, C./Li, L./Wang, L. (2021): Understanding Domain Randomization for Sim-to-real Transfer. In: *CoRR* abs/2110.03239. arXiv: 2110.03239. URL: https://arxiv.org/abs/2110.03239.

Collins, J./Ketter, W. (2022): Power TAC: Software architecture for a competitive simulation of sustainable smart energy markets. In: *SoftwareX* 20, S. 101217. ISSN: 2352-7110. DOI: https:

//doi.org/10.1016/j.softx.2022.101217. URL: https://www.sciencedirect.com/science/article/pii/S2352711022001352.

Cutler, M./Walsh, T. J./How, J. P. (2014): Reinforcement learning with multi-fidelity simulators. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. DOI: 10.1109/icra.2014.6907423.

Deshpande, A. M./Kumar, R./Minai, A. A./Kumar, M. (2020): Developmental Reinforcement Learning of Control Policy of a Quadcopter UAV with Thrust Vectoring Rotors. URL: https://arxiv.org/pdf/2007.07793.

Deshpande, A. M./Minai, A. A./Kumar, M. (2021): Robust Deep Reinforcement Learning for Quadcopter Control. In: *IFAC-PapersOnLine* 54.20, S. 90–95. ISSN: 24058963. DOI: 10.1016/j.ifacol.2021.11.158.

Fang, B./Jia, S./Guo, D./Xu, M./Wen, S./Sun, F. (2019): Survey of imitation learning for robotic manipulation. In: *International Journal of Intelligent Robotics and Applications* 3, S. 362–369.

Fitwi, A. H./Nagothu, D./Chen, Y./Blasch, E. (2019): A Distributed Agent-Based Framework for a Constellation of Drones in a Military Operation. In: *2019 Winter Simulation Conference (WSC)*, S. 2548–2559. DOI: 10.1109/WSC40007.2019.9004907.

Foronda, C. L. (2021): What Is Virtual Simulation? In: *Clinical Simulation in Nursing* 52, S. 8. ISSN: 18761399. DOI: 10.1016/j.ecns.2020.12.004.

Furrer, F./Burri, M./Achtelik, M./Siegwart, R. (2016): RotorS—A Modular Gazebo MAV Simulator Framework. In: *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Hrsg. von Anis Koubaa. Cham: Springer International Publishing, S. 595–625. ISBN: 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_23. URL: https://doi.org/10.1007/978-3-319-26054-9_23.

Gao, Y./Vedula, S. S./Reiley, C. E./Ahmidi, N./Varadarajan, B./Lin, H. C./Tao, L./Zappella, L./Béjar, B./Yuh, D. D. u. a. (2014): Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling. In: *MICCAI workshop: M2cai*. Bd. 3. 3.

GitHub (4/4/2023): Farama-Foundation/Gymnasium: A standard API for single-agent reinforcement learning environments, with popular reference environments and related utilities (formerly Gym). URL: https://github.com/Farama-Foundation/Gymnasium.

Google Scholar (2/28/2023). URL: https://scholar.google.de/.

Haarnoja, T./Zhou, A./Abbeel, P./Levine, S. (2018): Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In: *CoRR* abs/1801.01290. arXiv: 1801.01290. URL: http://arxiv.org/abs/1801.01290.

Hentati, A. I./Krichen, L./Fourati, M./Fourati, L. C. (2018): Simulation Tools, Environments and Frameworks for UAV Systems Performance Analysis. In: *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE. DOI: 10.1109/iwcmc.2018.8450505.

Hsu, K.-C./Ren, A. Z./Nguyen, D. P./Majumdar, A./Fisac, J. F. (2023): Sim-to-Lab-to-Real: Safe reinforcement learning with shielding and generalization guarantees. In:

Artificial Intelligence 314, S. 103811. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2022.103811. URL: https://www.sciencedirect.com/science/article/pii/S0004370222001515.

Huang, S. H./Papernot, N./Goodfellow, I. J./Duan, Y./Abbeel, P. (2017): Adversarial Attacks on Neural Network Policies. In: *CoRR* abs/1702.02284. arXiv: 1702.02284. URL: http://arxiv.org/abs/1702.02284.

Hussein, A./Gaber, M. M./Elyan, E./Jayne, C. (2017): Imitation Learning: A Survey of Learning Methods. In: *ACM Comput. Surv.* 50.2. ISSN: 0360-0300. DOI: 10.1145/3054912. URL: https://doi.org/10.1145/3054912.

IEEE Xplore (2/28/2023). URL: https://ieeexplore.ieee.org/Xplore/home.jsp.

Ivaldi, S./Padois, V./Nori, F. (2014): Tools for dynamics simulation of robots: a survey based on user feedback. URL: https://arxiv.org/pdf/1402.7050.

Koch, W./Mancuso, R./West, R./Bestavros, A. (2018): Reinforcement Learning for UAV Attitude Control. In: *CoRR* abs/1804.04154. arXiv: 1804.04154. URL: http://arxiv.org/abs/1804.04154.

Körber, M./Lange, J./Rediske, S./Steinmann, S./Glück, R. (2021): Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning. URL: https://arxiv.org/pdf/2103.04616.

Li, Y. (2019): Reinforcement Learning Applications. DOI: 10.48550/ARXIV.1908.06973. URL: https://arxiv.org/abs/1908.06973.

Liu, Z./Guo, Z./Cen, Z./Zhang, H./Tan, J./Li, B./Zhao, D. (2023): On the Robustness of Safe Reinforcement Learning under Observational Perturbations. arXiv: 2205.14691 [cs.LG].

Mnih, V./Badia, A. P./Mirza, M./Graves, A./Lillicrap, T. P./Harley, T./Silver, D./Kavukcuoglu, K. (2016): Asynchronous Methods for Deep Reinforcement Learning. In: *CoRR* abs/1602.01783. arXiv: 1602.01783. URL: http://arxiv.org/abs/1602.01783.

Mnih, V./Kavukcuoglu, K./Silver, D./Graves, A./Antonoglou, I./Wierstra, D./Riedmiller, M. A. (2013): Playing Atari with Deep Reinforcement Learning. In: *CoRR* abs/1312.5602. arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

Molchanov, A./Chen, T./Hönig, W./Preiss, J. A./Ayanian, N./Sukhatme, G. S. (2019): Sim-to-(Multi)-Real: Transfer of Low-Level Robust Control Policies to Multiple Quadrotors. In: *CoRR* abs/1903.04628. arXiv: 1903.04628. URL: http://arxiv.org/abs/1903.04628.

Moos, J./Hansel, K./Abdulsamad, H./Stark, S./Clever, D./Peters, J. (2022): Robust Reinforcement Learning: A Review of Foundations and Recent Advances. In: *Machine Learning and Knowledge Extraction* 4.1, S. 276–315. ISSN: 2504-4990. DOI: 10.3390/make4010013. URL: https://www.mdpi.com/2504-4990/4/1/13.

Ningombam, D. D. (2022): Deep Reinforcement Learning Algorithms for Machine-to-Machine Communications: A Review. In: *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE. DOI: 10.1109/icccnt54827.2022.9984457.

Pan, A./Lee, Y./Zhang, H./Chen, Y./Shi, Y. (2021): Improving Robustness of Reinforcement Learning for Power System Control with Adversarial Training. In: *arXiv e-prints*, arXiv:2110.08956, arXiv:2110.08956. DOI: 10.48550/arXiv.2110.08956. arXiv: 2110.08956 [eess.SY].

Panerati, J./Zheng, H./Zhou, S./Xu, J./Prorok, A./Schoellig, A. P. (2021): Learning to Fly – a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control. URL: https://arxiv.org/pdf/2103.02142.

Pinto, L./Davidson, J./Sukthankar, R./Gupta, A. (2017): Robust Adversarial Reinforcement Learning. In: *CoRR* abs/1703.02702. arXiv: 1703.02702. URL: http://arxiv.org/abs/1703.02702.

Pullum, L. L. (2022): Review of Metrics to Measure the Stability, Robustness and Resilience of Reinforcement Learning. arXiv: 2203.12048 [cs.LG].

Raffin, A./Hill, A./Gleave, A./Kanervisto, A./Ernestus, M./Dormann, N. (2021): Stable-Baselines3: Reliable Reinforcement Learning Implementations. In: *J. Mach. Learn. Res.* 22.1. ISSN: 1532-4435.

Recker, J. (2021): Scientific research in information systems: A beginner's guide. Second Edition. Progress in IS. Cham: Springer International Publishing. ISBN: 9783030854362. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6789173.

Reda, D./Tao, T./van de Panne, M. (2020): Learning to Locomote: Understanding How Environment Design Matters for Deep Reinforcement Learning. In: *Motion, Interaction and Games*. Hrsg. von Daniele Reda/Tianxin Tao/Michiel van de Panne. New York, NY, USA: ACM, S. 1–10. DOI: 10.1145/3424636.3426907.

Sadeghi, F./Levine, S. (2016): CAD2RL: Real Single-Image Flight without a Single Real Image. In: *CoRR* abs/1611.04201. arXiv: 1611.04201. URL: http://arxiv.org/abs/1611.04201.

Schott, L./Hajri, H./Lamprier, S. (2022): Improving Robustness of Deep Reinforcement Learning Agents: Environment Attack based on the Critic Network. In: *2022 International Joint Conference on Neural Networks (IJCNN)*, S. 1–8. DOI: 10.1109/IJCNN55064.2022.9892901.

Schuderer, A./Bromuri, S./van Eekelen, M. (2021): Sim-Env: Decoupling OpenAI Gym Environments from Simulation Models. In: *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer, Cham, S. 390–393. DOI: 10.1007/978-3-030-85739-4{\textunderscore}39. URL: https://link.springer.com/chapter/10.1007/978-3-030-85739-4_39.

Schulman, J./Levine, S./Moritz, P./Jordan, M. I./Abbeel, P. (2015): Trust Region Policy Optimization. In: *CoRR* abs/1502.05477. arXiv: 1502.05477. URL: http://arxiv.org/abs/1502.05477.

Schulman, J./Wolski, F./Dhariwal, P./Radford, A./Klimov, O. (2017): Proximal Policy Optimization Algorithms. In: *CoRR* abs/1707.06347. arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

Shah, S./Dey, D./Lovett, C./Kapoor, A. (2017): AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In: *CoRR* abs/1705.05065. arXiv: 1705.05065. URL: http://arxiv.org/abs/1705.05065.

Silano, G./Iannelli, L. (2019): CrazyS: A Software-in-the-Loop Simulation Platform for the Crazyflie 2.0 Nano-Quadcopter. In: *Robot Operating System (ROS): The Complete Reference (Volume 4)*. Hrsg. von Anis Koubaa. Cham: Springer International Publishing, S. 81–115. ISBN: 978-3-030-20190-6. DOI: 10.1007/978-3-030-20190-6_4. URL: https://doi.org/10.1007/978-3-030-20190-6_4.

Slaoui, R. B./Clements, W. R./Foerster, J. N./Toth, S. (2019): Robust Domain Randomization for Reinforcement Learning. In: *CoRR* abs/1910.10537. arXiv: 1910.10537. URL: http://arxiv.org/abs/1910.10537.

Sutton, R. S./Barto, A. G. (2018): Reinforcement Learning, second edition: An Introduction. MIT Press. ISBN: 9780262352703.

Tobin, J./Fong, R./Ray, A./Schneider, J./Zaremba, W./Abbeel, P. (2017): Domain randomization for transferring deep neural networks from simulation to the real world. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, S. 23–30. DOI: 10.1109/IROS.2017.8202133.

Todorov, E./Erez, T./Tassa, Y. (2012): MuJoCo: A physics engine for model-based control. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, S. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

Wang, Z./Hong, T. (2020): Reinforcement learning for building controls: The opportunities and challenges. In: *Applied Energy* 269, S. 115036. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2020.115036.

Webster, J./Watson, R. T. (2002): Analyzing the Past to Prepare for the Future: Writing a Literature Review. In: *MIS Q.* 26.2, S. xiii–xxiii. ISSN: 0276-7783.

Wong, A./Bäck, T./Kononova, A. V./Plaat, A. (2022): Deep multiagent reinforcement learning: challenges and directions. In: *Artificial Intelligence Review*. ISSN: 0269-2821. DOI: 10.1007/s10462-022-10299-x.

Yan Duan/Xi Chen/Rein Houthooft/John Schulman/Pieter Abbeel (2016): Benchmarking Deep Reinforcement Learning for Continuous Control. In: *International Conference on Machine Learning*, S. 1329–1338. ISSN: 1938-7228. URL: https://proceedings.mlr.press/v48/duan16.html.

Zhai, P./Hou, T./Ji, X./Dong, Z./Zhang, L. (2022): Robust Adaptive Ensemble Adversary Reinforcement Learning. In: *IEEE Robotics and Automation Letters* 7.4, S. 12562–12568. DOI: 10.1109/LRA.2022.3220531.

Zhang, A./Wu, Y./Pineau, J. (2018): Natural Environment Benchmarks for Reinforcement Learning. DOI: 10.48550/ARXIV.1811.06032. URL: https://arxiv.org/abs/1811.06032.

Zhao, W./Queralta, J. P./Westerlund, T. (2020): Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. DOI: 10.1109/ssci47803.2020.9308468.

Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema: *Experiment zur Verbesserung der Robustheit von Reinforcement Learning Modellen in Drohnensimulationen anhand trainiertem Gegenspieler* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

(Ort, Datum) (Unterschrift)