

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

Bachelorarbeit

vorgelegt am 27. März 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

Vertraulichkeitsvermerk: Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungs- und Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung des Dualen Partners vorliegt.

Inhaltsverzeichnis

Abkürzungsverzeichnis	IV
Abbildungsverzeichnis	V
Tabellenverzeichnis	VI
1 Einleitung	1
1.1 Problemstellung	1
1.2 Zielsetzung	2
1.3 Forschungsfrage	3
1.4 Forschungsmethodik	3
1.5 Aufbau der Arbeit	4
2 Diskussion des aktuellen Stands der Forschung und Praxis	5
2.1 Aufbau der Literaturrecherche	5
2.2 Verstärkendes Lernen	6
2.2.1 Methoden des verstärkenden Lernens	8
2.2.2 Algorithmen des verstärkenden Lernens	10
2.2.3 Abgrenzung zu Multi-Agent Reinforcement Learning (MARL) Algorithmen	12
2.2.4 Limitierungen und Herausforderungen von RL	13
2.3 Simulationsumgebungen für RL	13
2.3.1 Definitionen von Simulationsumgebungen	14
2.3.2 Entwicklung von Simulationsumgebungen für RL Anwendungen	14
2.3.3 aktuelle Physik-Engines und Simulationsanwendungen	16
2.4 Simulation der Steuerungsaufgabe von Quadroptern	17
2.4.1 Flugdynamiken eines Quadropters	17
2.4.2 Existierende Simulationen von Quadroptern	19
2.5 Robustheit und Stabilität von Strategien des verstärkenden Lernens	21
2.5.1 Definitionen von Robustheit und Stabilität	21
2.5.2 Metriken der Robustheit	21
2.5.3 Experimenteller Rahmen zur Messung der Robustheit	22
2.6 Gegnerisches verstärkendes Lernen	23
2.7 Domain Randomization	25
2.7.1 Das Prinzip hinter der Randomisierung von Simulationsumgebungen . . .	25
2.7.2 Anwendung von Randomisierung der Simulationsumgebung	26
2.7.3 Errungenschaften unter Einsatz von Domain Randomization	27
3 Durchführung des Laborexperiments	28
3.1 Erläuterung der Forschungsmethodik	28
3.1.1 Hypothesen	28
3.1.2 Messverfahren	28
3.2 Anforderungen der Test- und Simulationsumgebung	29
3.2.1 Anforderungen der Simulationsumgebung	29
3.2.2 Anforderungen des Experimentverfahrens	29
3.3 Programmumsetzung der Simulationsumgebung	29
3.4 Programmumsetzung zur Experimentdurchführung	29

4	Ergebnisse des Laborexperiments	30
5	Reflexion und Forschungsausblick	31
	Anhang	32
	Literaturverzeichnis	34

Abkürzungsverzeichnis

DHBW	Duale Hochschule Baden-Württemberg
RL	Reinforcement Learning
KPI	Key Performance Indicator
MARL	Multi-Agent Reinforcement Learning
DART	Dynamic Animation and Robotics Toolkit
ODE	Open Dynamics Engine
ROS	Robot Operating System
SITL	Software-in-the-Loop
RAEARL	Robust Adaptive Ensemble Adversarial Reinforcement Learning Framework
DQL	Deep Q-Learning
TRPO	Trust Region Policy Optimization
PPO	Proximal Policy Optimization
A3C	Asynchronous Advantage Actor-Critic
A2C	Advantage Actor-Critic
DR	Domain Randomization

Abbildungsverzeichnis

1	vereinfachte Darstellung der Interaktion zwischen dem Agenten und seiner Umgebung	7
2	Klassifizierung von Algorithmen im Bereich des RL	8
3	Rotationsbewegungen eines Quadropters	18
4	Aufbau des RAEARL Frameworks	25
5	Intuition hinter dem Paradigma von DR	26
6	Einfluss der Randomisierung verschiedener Effekte auf die Fehlerrate der Objekterkennung	26

Tabellenverzeichnis

1	Konzeptmatrix für Artikel zu Simulationsumgebungen und zur Robustheit RL Algorithmen nach Webster/Watson 2002. Legende: RL (Reinforcement Learning), MARL (Multi-Agent Reinforcement Learning), ES (Entwicklung von Simulationsumgebungen), DS (Drohnen-simulation), KS (kompetitive Simulationsumgebungen), DR (Domain Randomization), RRLP (Robustheit von RL Policies), LE (Laborexperimente)	6
2	wichtigsten Kriterien zur Auswahl von Simulatoren ¹	16
3	Interne quantitative Metriken, dessen gemessenes Verhalten und ihre Häufigkeit ²	22
4	Auszug der externen quantitativen Metriken, dessen gemessenes Verhalten und ihre Häufigkeit ³	23

¹Ivaldi/Padois/Nori 2014, S. 4

²Pullum 2022, S.17

³Pullum 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 Metas 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 alle 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 ein Problem beim Einsatz von RL Algorithmen die Limitierungen sein können, Daten in der echten Welt zu sammeln und fürs Training zu verwenden, werden häufig hierfür 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 abschätzbar sind.⁹ Simulationen nehmen damit zum einen als Testumgebung eine wichtige Rolle ein in der Entwicklung von Kontrollalgorithmen.¹⁰ Zum anderen bedarf die erfolgreiche Anwendung von RL neben effizienten Algorithmen eben auch geeignete Simulationsumgebungen.¹¹ Besonders schwierig, und daher sehr wichtig zu erforschen, ist es, die Trainingsumgebung bestmöglich an die echte Welt anzupassen, sodass bspw. die Agenten für Roboter und autonome Fahrzeuge nach dem Training mit robusten Strategien in der Realität eingesetzt werden können.¹² In der Forschungsliteratur wird diese beschriebene 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 Drohnenverlust zu vermeiden.¹⁵ Drohnen tragen dabei bereits in der heutigen Zeit zur Lösung vieler komplexer Aufgaben bei, wie der Katastrophenüberwachung oder der Waldbrandbekämpfung.¹⁶ Zusätzlich

⁴Vgl. Li 2019, S. 4

⁵Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 3

⁶Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 3

⁷Vgl. Reda/Tao/van de Panne 2020, S. 1

⁸Vgl. Zhao/Queralta/Westerlund 2020, S. 737

⁹Vgl. Zhao/Queralta/Westerlund 2020, S. 738

¹⁰Vgl. Cutler/Walsh/How 2014, S. 2

¹¹Vgl. Reda/Tao/van de Panne 2020, S. 8

¹²Vgl. Slaoui u. a. 2019, S. 1

¹³Vgl. Zhao/Queralta/Westerlund 2020, S. 738

¹⁴Vgl. Deshpande/Minai/Kumar, M. 2021, S. 1

¹⁵Vgl. Sadeghi/Levine 2016, S. 4

¹⁶Vgl. Hentati u. a. 2018, S. 1495

steigen immer weiter die komplexen Einsatzanforderungen unter dem Anstieg an Anwendungsgebieten für unbemannte Luftfahrzeuge.¹⁷

Die Simulation einer möglichst realistischen Umgebung in diesem Kontext wird in der Forschung häufig mit dem Ansatz von Domain Randomization (DR) begleitet.¹⁸ 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.¹⁹ 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.²⁰ 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

Daher soll im Rahmen dieser Arbeit untersucht werden, ob die Integrierung eines RL basierten Gegenspielers in einer Simulation die Umgebung so beeinflussen kann, dass die erlernten Verhaltensmodelle, welche im Kontext von RL oftmals als Policies referenziert werden, robuster agieren unter den veränderten dynamischen Bedingungen und alternativen deterministischen Gegenspielern im Testszenario.

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 trainiert.

- 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 einer Reihe von Testszenarien untersucht. Jedes Testszenario verfügt dabei über festgelegte sich vom Training unterscheidende Dynamikparameter und jeweils leicht unterschiedliche Handlungspräferenzen des deterministischen Gegenspielers. Bei der Untersuchung werden jeweils die folgenden Variablen als Key Performance Indicator (KPI) betrachtet.

¹⁷Vgl. Deshpande/Kumar, R. u. a. 2020, S. 1

¹⁸Vgl. Sadeghi/Levine 2016, S. 1

¹⁹Vgl. Zhao/Queralta/Westerlund 2020, S. 4f.

²⁰Vgl. Collins/Ketter 2022, S. 2

- durchschnittlich erzielte Belohnung
- Varianz der Belohnungen
- Anzahl an unbeabsichtigten Abstürzen

Aus der Auswertung der Testszenarien kann der Effekt des RL basierten Gegenspielers auf die Robustheit mittels des Vergleichs mit dem regelbasierten Gegenspieler und der Domain Randomization evaluiert werden.

1.3 Forschungsfrage

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

Kann durch den Einsatz eines mittels RL trainierten Gegenspielers die Robustheit der gelernten Policy verbessert werden?

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

Hypothese 1: *Die in den Testszenarien durchschnittlich erzielte 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 in den Testszenarien 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.*

Hypothese 3: *Die in den Testszenarien erreichte Anzahl von unbeabsichtigten Abstürzen 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.²¹ Damit stellt das Laborexperiment eine Möglichkeit der Untersuchung der Ursache- und Wirkungsbeziehung dar.²² Dabei wird die kontrollierte Umgebung der Simulation erschaffen, deren

²¹Vgl. Recker 2021, S. S.89f.

²²Vgl. Recker 2021, S. 106

Aufbau die unabhängige Variable darstellt. Die Metriken, anhand welcher die Performance und die Robustheit der trainierten Policies gemessen werden, bilden im Experiment die abhängigen Variablen.

1.5 Aufbau der Arbeit

Insgesamt gliedert sich die Arbeit nach einem Schema von Holzweißig 2022. 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 der Problemstellung wiedergegeben. Im dritten Kapitel wird die Forschungsmethodik dargestellt, indem die Simulationsumgebung als Messinstrument entwickelt wird sowie verschiedene Messszenarien erläutert und entsprechende Daten gesammelt werden. Daraufhin sind im folgenden vierten Kapitel die Messdaten auszuwerten und aufgestellte Hypothesen zu überprüfen. Im Zuge dessen kann ebenso die Forschungsfrage anhand der Annahme oder Ablehnung der Hypothesen beantwortet werden. Abschließend wird im letzten Kapitel ein Fazit zu den erzielten Forschungsergebnissen dargelegt und ein Ausblick auf weitere Forschung gegeben.

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 daraus gefunden Literatur wurden zitierte Werke ebenfalls nach den beschriebenen Konzepten durchsucht und insgesamt jede Literaturquelle in Tabelle 1 den in ihnen enthaltenen Konzepten zugeordnet.

Artikel	Konzepte							
	RL	MARL	ES	DS	KS	DR	RRLP	LE
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/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						
Schuderer/Bromuri/van Eekelen 2021	X	X	X					
Körber u. a. 2021			X					
Bharadhwaj u. a. 2019			X			X		
Foronda 2021			X					
Maria 1997			X					
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					

Artikel	Konzepte							
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), ES (Entwicklung von Simulationsumgebungen), DS (Drohnsimulation), KS (kompetitive Simulationsumgebungen), DR (Domain Randomization), RRLP (Robustheit von RL Policies), LE (Laborexperimente)

2.2 Verstärkendes Lernen

Verstärkendes Lernen, oder auch als Reinforcement Learning (RL) in der Fachsprache bezeichnet, definiert einen konzeptionellen Ansatz zielorientiertes Lernen von Entscheidungen zu verstehen und zu automatisieren.²³ Dabei besteht der Fokus darauf, dass ein Agent aus der direkten Interaktion mit seiner Umgebung lernt, ohne dass explizite Überwachung notwendig ist.²⁴ Der Agent lernt über die Zeit eine optimale Strategie zur Lösung des Entscheidungsproblems aus dem Ausprobieren und Scheitern mittels verschiedener Aktionen die gewünschte Veränderung

²³Vgl. Sutton/Barto 2018, S. 13

²⁴Vgl. Sutton/Barto 2018, S. 13

in seiner Umwelt herzustellen.²⁵ Notwendig dabei ist es, dass der Agent den Zustand seiner Umgebung wahrnehmen und auch durch entsprechende Aktionen beeinflussen kann, sodass die Erreichung des Zielzustandes möglich ist.²⁶ Zur Erreichung dieses Zielzustandes muss der Agent diejenigen Aktionen entdecken, welche ihm die größtmögliche kumulierte Belohnung liefern, wobei Aktionen nicht nur die unmittelbare sondern auch zukünftige Belohnungen beeinflussen.²⁷ Zusammengefasst lässt sich die beschriebene Interaktion des Agenten mit seiner Umgebung wie folgt in Abbildung 1 darstellen.

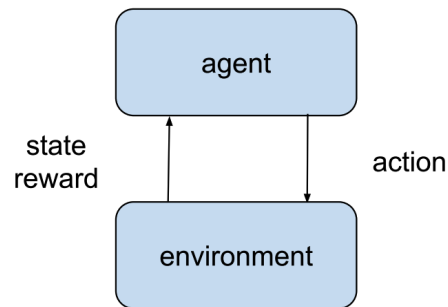


Abb. 1: vereinfachte Darstellung der Interaktion zwischen dem Agenten und seiner Umgebung²⁸

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.²⁹ 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 nach einem Tupel aus folgenden Elementen:³⁰

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

Zum Finden der optimalen Strategie existieren modellbasierende und modellfreie Algorithmen des verstärkenden Lernens.³¹ Bei modellbasierenden Algorithmen wird das Umgebungsverhalten, also die Übergangs- und Belohnungswahrscheinlichkeiten, als bekannt vorausgesetzt.³² Als modellbasierender Algorithmus wird z.B. dynamische Programmierung eingesetzt, um mittels

²⁵Vgl. Li 2019, S. 4

²⁶Vgl. Sutton/Barto 2018, S. 2

²⁷Vgl. Sutton/Barto 2018, S. 1

²⁸Enthalten in: Li 2019, S. 5

²⁹Vgl. Zhao/Queralta/Westerlund 2020, S. 2

³⁰Vgl. Zhang/Wu/Pineau 2018, S. 2

³¹Vgl. Wang/Hong 2020, S. 3

³²Vgl. Wang/Hong 2020, S. 3

Strategieevaluation und Strategieiteration die optimale Strategie zu finden.³³ Unter modellfreien Algorithmen werden die drei verschiedenen Ansätze Wertebasierend, Strategiebasierend und Akteur-Kritiker basierend unterschieden³⁴ Der Agent im Kontext von modellfreien RL Methoden kennt nur die Zustände S und die Aktionen A , jedoch nicht das Umgebungsverhalten T und die Belohnungswahrscheinlichkeit R .³⁵ Fasst man die Klassifizierung der Algorithmen und Methoden von RL zusammen, lässt sie sich wie folgt darstellen:

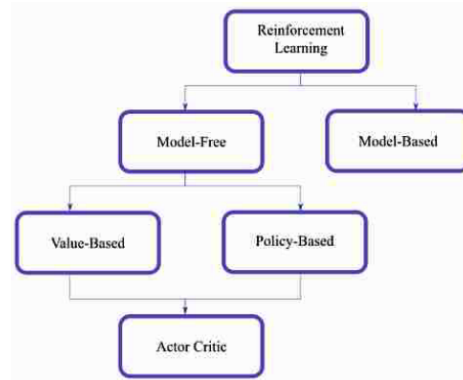


Abb. 2: Klassifizierung von Algorithmen im Bereich des RL³⁶

2.2.1 Methoden des verstärkenden Lernens

Der Agent sucht im Kontext von wertbasierenden Methoden die optimale Strategie π^* , welche 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.³⁷ Neben dieser kurzfristigen direkten Belohnung müssen auch die langfristigen zukünftigen Belohnungen aus den neuen Zuständen betrachtet werden, wofür das Konzept der Wertigkeit eingeführt wird.³⁸ Über eine Zustands- oder Aktionswertigkeitsfunktion, oftmals als Q-Funktion referenziert, wird eine Vorhersage über die zu erwartende kumulierte abgezinste zukünftige Belohnung berechnet.³⁹ Durch den Abzinsungsfaktor $\gamma \in [0, 1)$ wird der Einfluss zukünftiger Belohnungen nach ihrer zeitlichen Reihenfolge priorisiert.⁴⁰ Mit der Wertigkeitsfunktion kann evaluiert werden, welche Strategie langfristig am erfolgreichsten ist, da bspw. manche Aktionen trotz geringer sofortiger Belohnung einen hohen Wert aufweisen können, wenn aus dem zukünftigen Zustand eine hohe Belohnung zu erwarten ist.⁴¹ 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.⁴² Methoden basierend auf diesem

³³Vgl. Li 2019, S. 5

³⁴Vgl. Li 2019, S. 5

³⁵Vgl. Cutler/Walsh/How 2014, S. 2

³⁶Enthalten in: Canese u. a. 2021, S. 6

³⁷Vgl. Reda/Tao/van de Panne 2020, S. 2

³⁸Vgl. Wang/Hong 2020, S. 3

³⁹Vgl. Li 2019, S. 5

⁴⁰Vgl. Li 2019, S. 5

⁴¹Vgl. Sutton/Barto 2018, S. 6

⁴²Vgl. Sutton/Barto 2018, S. 6f.

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 aktualisieren der folgenden Funktionen eins und zwei:⁴³

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

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

Aus den geschätzten Wertigkeit jedes Zustandsaktions Paares kann die optimale Strategie $\pi^*(s)$ durch $\arg \max_a Q(s, a)$ bestimmt werden.⁴⁴

Methoden, welche die Strategie durch deren direkte Parametrisierung anstelle einer Bewertung aller Handlungsalternativen mittels Wertigkeitsfunktion optimieren, werden als strategiebasierend bezeichnet.⁴⁵ Diese Methodik kann beim Trainieren deterministischer Strategien zu unerwarteten Aktionen führen, weshalb häufig das Optimieren einer Wahrscheinlichkeitsverteilung für alle Aktionen bevorzugt wird.⁴⁶ Als Subklasse der RL Methoden wird der statistische Gradientenabstieg verwendet um die parametrisierte Strategie π_θ hinsichtlich der maximalen langfristigen kumulierten Belohnung zu optimieren.⁴⁷ Die Strategie π_θ oder auch $\pi(a|s, \theta)$ beschreibt dabei die Wahrscheinlichkeit Aktion a im Zustand s auszuwählen unter dem Parametervektor θ .⁴⁸ Zur Optimierung der Strategie wird die Funktion der kumulierten Belohnungen J nach dem Parameter der Gewichte θ wie folgt in Formel drei abgeleitet und der optimierte Parametervektor anhand Formel vier aktualisiert.⁴⁹

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

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

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.⁵⁰

Unter Akteur-Kritiker Methoden werden hybride wertebasierende und strategiebasierende Methoden verstanden, welche zugleich die Strategie optimieren und eine Wertefunktion approximieren.⁵¹ Die strategiebasierende Methodik mit der lernenden Strategie agiert dabei als Akteur, wohingegen die Wertefunktion, welche jeder Aktion und jedem Zustand einen Belohnungswert zuweist, als Kritiker handelt.⁵² Der Akteur wählt somit aus seiner Wahrscheinlichkeitsverteilung

⁴³Vgl. Zhang/Wu/Pineau 2018, S. 2

⁴⁴Vgl. Zhang/Wu/Pineau 2018, S. 2

⁴⁵Vgl. Zhang/Wu/Pineau 2018, S. 2

⁴⁶Vgl. Ningombam 2022, S. 3

⁴⁷Vgl. Ningombam 2022, S. 3

⁴⁸Vgl. Sutton/Barto 2018, S. 321

⁴⁹Vgl. Wang/Hong 2020, S. 6

⁵⁰Vgl. Wang/Hong 2020, S. 6

⁵¹Vgl. Zhang/Wu/Pineau 2018, S. 2f.

⁵²Vgl. Sutton/Barto 2018, S. 321

die auszuführende Aktionen aus, während der Kritiker diese anhand seiner Wertigkeit bewertet.⁵³ Betrachtet man den Trainingsprozess von Akteur-Kritiker basierten Methoden ist dieser wie folgt aufgebaut:⁵⁴

- Aktueller Zustand der Umgebung als Eingabe dem Akteur und Kritiker übergeben
- Akteur liefert eine auszuführende Aktion basierend auf dem Umgebungszustand
- Der Kritiker bekommt die Aktion als Eingabe und berechnet deren Wertigkeit mittels Q-Funktion
- Durch die Wertigkeit seiner Aktion kann der Akteur seine Strategie anpassen
- Mit der neuen Strategie führt der Akteur die nächste Aktion im folgenden Zustand aus
- 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 daran wird in diesem Abschnitt der Arbeit 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).⁵⁵

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.⁵⁶ Als wertbasierte Methode wird für jeden Zustand die Wertigkeit berechnet, was unter DQL mittels eines neuronalen Netzes erreicht wird, welches die Q-Funktion approximiert.⁵⁷ Durch Auswählen der Aktion mit der höchsten Wertigkeit in jedem Zustand, kann die deterministische Strategie abgeleitet werden.⁵⁸ DQL adressiert das Instabilitätsproblem von Funktionsapproximation unter dem Einsatz von Erfahrungswiederholung und Zielnetzwerken.⁵⁹ Erfahrungswiederholung beschreibt die Technik die Erfahrung 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.⁶⁰ Durch die Erfahrungswiederholung wird eine

⁵³Vgl. Ningombam 2022, S. 3

⁵⁴Vgl. Ningombam 2022, S. 4

⁵⁵Vgl. Arulkumaran u. a. 2017, S. 1

⁵⁶Vgl. Arulkumaran u. a. 2017, S. 6

⁵⁷Vgl. Huang u. a. 2017, S. 4

⁵⁸Vgl. Huang u. a. 2017, S. 4

⁵⁹Vgl. Arulkumaran u. a. 2017, S. 7

⁶⁰Vgl. Mnih/Kavukcuoglu u. a. 2013, S. 4

bessere Dateneffizienz zum einen durch die Wiederholung erzielt, und zum anderen durch das Auflösen von Korrelationen zwischen aufeinanderfolgenden Zuständen.⁶¹ 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.⁶² Die Stärke von DQL liegt in der kompakten Repräsentation der Q-Funktion und der hochgradig dimensionierten Zustandsbeobachtungen durch neuronale Netze.⁶³

Trust Region Policy Optimization (TRPO) ist ein strategiebasierender Gradientenalgorithmus zur effektiven Optimierung großer nicht linearer Strategien wie z.B. neuronale Netze.⁶⁴ 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.⁶⁵ Die Varianten unterscheiden sich im Schätzverfahren des Gradienten, wobei die *single-path* Methodik diesen anhand einer Aktions-Zustandskette der initialen Verteilung bestimmt, und unter *vine* eine Teilmenge mehrerer verschiedener Aktion-Zustandspaare verwendet werden.⁶⁶ Die daraus entstehenden Anpassungen der Strategieparameter θ zwischen der alten und neuen Strategie werden mittels Kullback-Leibler-Divergenz bemessen und kontrolliert.⁶⁷ Eine Weiterentwicklung des TRPO Algorithmus ist Proximal Policy Optimization (PPO), durch dessen Verwendung die Implementierung erleichtert und die Datenprobenkomplexität verbessert wird.⁶⁸ 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.⁶⁹ Durch diese Anpassung der KL-Divergenz Beschränkung ermöglicht der PPO Algorithmus mehrfache Berechnungen des Gradienten anhand eines Datenobjektes.⁷⁰

Zusätzlich zu den bisherigen Algorithmen kann durch jeweilige asynchrone Varianten, welche den Trainingsprozess des Agenten parallelisieren, die Stabilität des Trainings verbessert werden.⁷¹ Einer dieser asynchronen Algorithmen im Bereich der Akteur-Kritiker Methodik stellt der *asynchronous advantage actor-critic* (A3C) Algorithmus dar, welcher anstelle von Erfahrungswiederholung mehrere Agenten parallel in unterschiedlichen Umgebungsinstanzen trainiert.⁷² 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.⁷³ Wird mit dem Algorithmus lediglich ein Agent trainiert, wird dies häufig auch

⁶¹Vgl. Mnih/Kavukcuoglu u. a. 2013, S. 4f.

⁶²Vgl. Arulkumaran u. a. 2017, S. 7

⁶³Vgl. Arulkumaran u. a. 2017, S. 7

⁶⁴Vgl. Schulman/Levine u. a. 2015, S. 1

⁶⁵Vgl. Schulman/Levine u. a. 2015, S. 1

⁶⁶Vgl. Schulman/Levine u. a. 2015, S. 4

⁶⁷Vgl. Huang u. a. 2017, S. 4

⁶⁸Vgl. Schulman/Wolski u. a. 2017, S. 1

⁶⁹Vgl. Schulman/Wolski u. a. 2017, S. 3f.

⁷⁰Vgl. Schulman/Wolski u. a. 2017, S. 4

⁷¹Vgl. Mnih/Badia u. a. 2016, S. 1

⁷²Vgl. Mnih/Badia u. a. 2016, S. 1

⁷³Vgl. Arulkumaran u. a. 2017, S. 9

als advantage actor-critic (A2C) referenziert.⁷⁴ Neben A3C ist der soft actor-critic Algorithmus ein weiterer Algorithmus der Akteur-Kritiker Methodik, welcher die kumulierte Belohnung, aber auch die Entropie maximiert.⁷⁵ 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.⁷⁶ 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 exponentialen Q-Funktion angepasst.⁷⁷ Durch die abwechselnde Approximation der Q-Funktion des Kritikers und der Strategie des Akteurs, anstelle der Berechnung dieser bis zur Konvergenz, kann der soft actor-critic Algorithmus besonders mit großen kontinuierlichen Handlungsbereichen umgehen.⁷⁸

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 einem Agenten eine Menge von Agenten eingesetzt, welche alle mit ihrer Umgebung interagieren um den Weg der Zielerreichung zu lernen.⁷⁹ Dieser Ansatz dient dazu die Vielzahl an Problemstellungen in der echten Welt, welche nicht vollständig durch einen Agenten lösbar sind, zu bearbeiten.⁸⁰ Einsatzgebiete von MARL sind dabei unter anderem das Routing von Netzwerkpaketen, Wirtschaftsmodellierung oder zusammenhängende Robotersysteme.⁸¹ 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.⁸² Aus den einzelnen Interaktion jedes Agenten mit der selben Umgebung ergibt sich der Unterschied, dass die Umgebungsdynamik aus der Kombination aller Aktionen der Agenten beeinflusst wird anstatt aus der Aktion des einzelnen Agenten.⁸³ Da dieser Effekt die Annahme der Stationarität von Markov Entscheidungsprozessen verletzt, bedarf die Umgebung einer anderen Representation.⁸⁴ Ein Konzept, das dafür häufig verwendet wird, ist das Markov Spiel, welches sich anders als der Entscheidungsprozess durch einen mehrdimensionalen Aktions- und Belohnungsraum aus der Kombination aller N Agenten auszeichnet.⁸⁵ 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.⁸⁶ Zieht man Rückschlüsse zum in der Ein-

⁷⁴Vgl. Arulkumaran u. a. 2017, S. 9

⁷⁵Vgl. Haarnoja u. a. 2018, S. 1

⁷⁶Vgl. Haarnoja u. a. 2018, S. 3

⁷⁷Vgl. Haarnoja u. a. 2018, S. 4

⁷⁸Vgl. Haarnoja u. a. 2018, S. 4

⁷⁹Vgl. Wong u. a. 2022, S. 6

⁸⁰Vgl. Canese u. a. 2021, S. 1

⁸¹Vgl. Canese u. a. 2021, S. 1

⁸²Vgl. Canese u. a. 2021, S. 8f.

⁸³Vgl. Wong u. a. 2022, S. 2

⁸⁴Vgl. Wong u. a. 2022, S. 6

⁸⁵Vgl. Canese u. a. 2021, S. 4

⁸⁶Vgl. Canese u. a. 2021, S. 9ff.

leitung beschriebenen Szenario, wird deutlich, dass in jenem Szenario kompetitives Multi-Agent RL zum Einsatz kommt.

2.2.4 Limitierungen und Herausforderungen von RL

Trotz signifikanter Errungenschaften birgt der Einsatz von den besprochenen RL Algorithmen weiterhin Limitierungen und Risiken für ungewolltes Verhalten.⁸⁷

Eine der Herausforderungen zeigt sich bei der Representation der Agentenumwelt, da RL stark auf diesem Konzept basiert.⁸⁸ Daraus ergibt sich die Aufgabe, die Umwelt und deren Verhalten sowie die Wahrnehmung durch den Agenten realitätsgetreu und präzise zu gestalten.⁸⁹ 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.⁹⁰ 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.⁹¹

Eine weitere Limitierung stellt der große Suchraum an Aktionen und das unbekannte Verhalten der Umgebung dar. Dies sorgt dafür, dass häufig die Effizienz einzelner Daten sehr gering ist und die Abwägung zwischen der Exploration neuer Strategie und der Optimierung bekannter Verhaltensmuster ein wichtiger Bestandteil ist.⁹² Aufgrund der geringen Effizienz der Daten aber des dennoch hohen Bedarfs an bewerteter Agentenerfahrung wird häufig auf simulierte Daten zurückgegriffen.⁹³ Simulierte Daten werden dabei häufig von möglichst hoch qualitativen Simulationsumgebungen bereitgestellt, da zu dem hohen Bedarf der Methodik häufig Limitierungen in der Sammlung von Daten in der echten Welt bestehen.⁹⁴

Aufgrund dieser Bedeutung der Simulationsumgebung für RL Algorithmen und deren Transfer in die echte Welt werden im nachfolgenden Kapitel die Merkmale und Entwicklungen von Simulationen genauer betrachtet.

2.3 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.⁹⁵ Um entsprechend realistische Daten für das Training bereitzustellen, werden

⁸⁷Vgl. Li 2019, S. 7

⁸⁸Vgl. Sutton/Barto 2018, S. 8

⁸⁹Vgl. Sutton/Barto 2018, S. 7

⁹⁰Vgl. Li 2019, S. 7

⁹¹Vgl. Li 2019, S. 7

⁹²Vgl. Li 2019, S. 7

⁹³Vgl. Zhao/Queralta/Westerlund 2020, S. 7

⁹⁴Vgl. Li 2019, S. 8

⁹⁵Vgl. Zhang/Wu/Pineau 2018, S. 1

Simulationsumgebungen in Abhängigkeit von ihrer RL Anwendung ausgewählt.⁹⁶ Dennoch bleibt nahezu immer eine gewisse Diskrepanz zwischen der Dynamik in der Simulation und der Dynamik in der echten Welt.⁹⁷ Möglichkeiten, diese Diskrepanz zu verkleinern, sind zum einen das Fehlverhalten von Sensoren einzubinden oder ein reales Signal mit der virtuellen Umgebung zu verknüpfen.⁹⁸ Trotzdem ist es kaum garantiert, dass erlernte Strategien der Agenten sich auch auf nur leicht veränderte Umgebungen übertragen lassen.⁹⁹ 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.¹⁰⁰ RL kann von einem erhöhten Echtzeitfaktor profitieren, trotz der damit einhergehenden verringerten Präzision.¹⁰¹

2.3.1 Definitionen von Simulationsumgebungen

Ausgehend von der Literaturrecherche zeigte sich, dass in der Forschungsliteratur die allgemeine Definition von Simulationen kaum aufgegriffen wird. Eine mögliche Definition nach Maria 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.¹⁰²

2.3.2 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:¹⁰³

1. Identifikation der Herausforderungen im existierenden System und Ableitung von Anforderungen für die Simulation.

⁹⁶Vgl. Körber u. a. 2021, S. 7

⁹⁷Vgl. Bharadhwaj u. a. 2019, S. 1

⁹⁸Vgl. Zhang/Wu/Pineau 2018, S. 1

⁹⁹Vgl. Bharadhwaj u. a. 2019, S. 1

¹⁰⁰Vgl. Körber u. a. 2021, S. 3

¹⁰¹Vgl. Körber u. a. 2021, S. 3

¹⁰²Vgl. Foronda 2021, S. 1

¹⁰³Vgl. Maria 1997, S. 8f.

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.¹⁰⁴ Aus dieser Motivation wurde 2016 durch die OpenAI der OpenAI Gym Werkzeugkasten entwickelt, welcher eine Sammlung an Benchmarksimulationen mit einheitlicher Schnittstelle für RL Algorithmen enthält.¹⁰⁵ Seither wurde diese definierte Schnittstelle vielfach verwendet, um RL Umgebungen mit dem Ziel zu entwickeln, diese zu publizieren und dessen Wiederverwendung zu ermöglichen.¹⁰⁶ 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.¹⁰⁷ Der Werkzeugkasten von OpenAI fokussiert sich auf einen Episoden ähnlichen Rahmen, in welchem der Agent durch zunächst zufälliges Auswählen von Interaktionen lernt.¹⁰⁸ Weitere Entwicklungsentscheidungen des OpenAI Gym Werkzeugkastens umfassen z.B. die bewusst fehlende Schnittstelle des Agenten, die strikte Versionierung der Umgebung oder die standardmäßige Simulationsüberwachung.¹⁰⁹

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.¹¹⁰ Eine enge initiale Wahrscheinlichkeitsverteilung des Umgebungszustandes kann die Lerneffizienz erhöhen, wohingegen eine weite Wahrscheinlichkeitsverteilung positiv die Robustheit der erlernten Strategie beeinflusst.¹¹¹ 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.¹¹² Im Bereich der Robotik bzw. in der Simulation von Bewegungen, kann auch durch die Gestaltung des Aktionsraumes, basierend auf

¹⁰⁴Vgl. Brockman u. a. 2016, S. 1

¹⁰⁵Vgl. Brockman u. a. 2016, S. 1

¹⁰⁶Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 4

¹⁰⁷Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 4

¹⁰⁸Vgl. Brockman u. a. 2016, S. 1

¹⁰⁹Vgl. Brockman u. a. 2016, S. 2f.

¹¹⁰Vgl. Reda/Tao/van de Panne 2020, S. 1

¹¹¹Vgl. Reda/Tao/van de Panne 2020, S. 3

¹¹²Vgl. Yan Duan u. a. 2016, S. 2

elektrischer Regelungstechnik mittels PID-Regler, anstatt basierend auf Drehmomenten ein effizienterer Lernprozess stattfinden.¹¹³

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: wichtigsten Kriterien zur Auswahl von Simulatoren¹¹⁴

Aus Tabelle zwei 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.¹¹⁵

2.3.3 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.

Gazebo

Gazebo ist eine durch die Open Source Robotics Foundation entwickelte Simulationsanwendung, welche mehrere Physik-Engines unterstützt.¹¹⁶ Mittels Gazebo lassen sich Interaktionen zwischen Robotern in Innen- und Außenbereichen unter realistischer Sensorik simulieren.¹¹⁷ Die unterstützten Physik-Engines umfassen Bullet, Dynamic Animation and Robotics Toolkit (DART), Open Dynamics Engine (ODE) und Simbody.¹¹⁸

¹¹³Vgl. Reda/Tao/van de Panne 2020, S. 7

¹¹⁴Enthalten in: Ivaldi/Padois/Nori 2014, s. 4

¹¹⁵Vgl. Ayala u. a. 2020, S. 2

¹¹⁶Vgl. Ivaldi/Padois/Nori 2014, S. 7

¹¹⁷Vgl. Ayala u. a. 2020, S. 4

¹¹⁸Vgl. Körber u. a. 2021, S. 3

MuJoCo

MuJoCo stellt eine Physik-Engine für modellbasierte Steuerung dar, dessen Objekte durch C++ oder XML definiert und Gelenkzustände im Koordinatensystem beschrieben werden.¹¹⁹ Diese beschriebene Eigenschaft lässt sich auch aus dem Namen als Abkürzung für **M**ulti-**J**oint dynamics with **C**ontact ableiten.¹²⁰ Die MuJoCo Anwendung ist lizenziert, was den Besitz einer Lizenz für die Installation oder die Virtualisierung innerhalb eines Containers voraussetzt.¹²¹

PyBullet

PyBullet basiert als Simulationssoftware auf der Bullet-Engine und fokussiert sich funktional auf die Anwendung von RL im Robotikbereich.¹²² Die Bullet-Engine ist hingegen eine offene Softwarebibliothek, welche neben verstärkenden Lernen auch bei Computeranimationen angewendet wird.¹²³ Die 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.¹²⁴

2.4 Simulation der Steuerungsaufgabe von Quadroptern

Die Popularität von unbemannten Flugzeugen im letzten Jahrzehnt nahm besonders im Bereich der Quadropten zu, sodass durch die sinkenden Kosten von Sensorik und Minicomputern, zahlreiche zukunftssträchtige Ergebnisse und Anwendungen erforscht worden sind.¹²⁵ Anwendungsgebiete beinhalten z.B. die Landwirtschaft, den Pakettransport oder die Überwachung von großflächiger Infrastruktur wie Stromnetze.¹²⁶ Eine Simulation von unbemannten Flugzeugen stellt eine Flugumgebung und vielseitige Sensorik bereit und kann je nach Anwendung Effekte wie Wind, Wolken und Niederschlag einbeziehen.¹²⁷

2.4.1 Flugdynamiken eines Quadropters

Die Simulation des Quadropters stellt eine Simulation eines Flugkörpers mit drei Rotations- und drei Translationsbewegungen und demnach insgesamt sechs verschiedenen Freiheitsgraden dar.¹²⁸ Ein Quadropter 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.¹²⁹ Die vier Rotoren werden als + oder X Konfiguration entweder direkt in Richtung der

¹¹⁹Vgl. Todorov/Erez/Tassa 2012, S. 1

¹²⁰Vgl. Todorov/Erez/Tassa 2012, S. 2

¹²¹Vgl. Körber u. a. 2021, S. 3

¹²²Vgl. Körber u. a. 2021, S. 3

¹²³Vgl. Ivaldi/Padois/Nori 2014, S. 7

¹²⁴Vgl. Körber u. a. 2021, S. 6

¹²⁵Vgl. Koch u. a. 2018, S. 1

¹²⁶Vgl. Deshpande/Kumar, R. u. a. 2020, S. 1

¹²⁷Vgl. Hentati u. a. 2018, S. 1496

¹²⁸Vgl. Koch u. a. 2018, S. 2

¹²⁹Vgl. Molchanov u. a. 2019, S. 3

X- und Y-Achse (+), oder um 45° gedreht (X) an der Drohne befestigt.¹³⁰ Jede Bewegung der sechs verschiedenen Arten wird durch die unterschiedliche Steuerung, der im Fall des Quadropters, vier Rotoren getätigt. Eine unterschiedliche Ansteuerung der Rotoren und den demnach verschieden starken Auftrieben resultiert in den drei Rotationsbewegungen Rollen, Nicken und Gieren.¹³¹

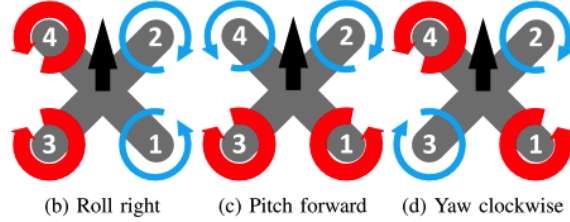


Abb. 3: Rotationsbewegungen eines Quadropters¹³²

Rollen wird wie aus Abbildung drei 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 Uhrzeiger drehenden oder der sich 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.¹³³ 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.¹³⁴

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

In Formel vier wird der Rotationszustand der Drohne zu den Weltachsen 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.¹³⁵

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

¹³⁰Vgl. Koch u. a. 2018, S. 2

¹³¹Vgl. Koch u. a. 2018, S. 2

¹³²Enthalten in: Koch u. a. 2018, S. 2

¹³³Vgl. Deshpande/Kumar, R. u. a. 2020, S. 3

¹³⁴Vgl. Deshpande/Minai/Kumar, M. 2021, S. 2

¹³⁵Vgl. Deshpande/Minai/Kumar, M. 2021, S. 2

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 .¹³⁶ 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.¹³⁷

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

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.¹³⁸

2.4.2 Existierende Simulationen von Quadroptern

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.¹³⁹ Dabei wurde die Simulation modular entworfen, sodass Komponenten wie Steuerung oder Zustandsschätzung austauschbar sind und das Hinzufügen von neuen Drohnen erleichtert wird.¹⁴⁰ Die Komponenten der Drohne stellen dabei Plug-ins der verwendeten Gazebo Physik-Engine dar, wodurch ein Mikroflugzeug aus den Teilen des Körper, der Anzahl der Rotoren und Sensorik an fixen Position zusammengesetzt wird.¹⁴¹ Mittels der standardmäßigen Sensorik können Informationen über die direkte und visuell gemessene Trägheit sowie über die Wegbestimmung erzielt werden.¹⁴² Anstelle des Wegbestimmungssensors kann auch eine Komponente zur Zustandsschätzung für hochfrequente Abfragen implementiert werden.¹⁴³ Die Steuerungskomponente wird durch eine einfache Schnittstelle einer geometrischen Steuerung bedient, welche Aktionen in Form von Rotationswinkeln, Höhen oder Positionen entgegen nimmt.¹⁴⁴

CrazyS

CrazyS stellt eine Erweiterung der Simulation RotorS auf Basis desselben ROS, um die Modellierung des Nano-Quadropters Crazyflie samt ihrer Dynamik ihres Kontrollsystems und ihrer

¹³⁶Vgl. Deshpande/Minai/Kumar, M. 2021, S. 2

¹³⁷Vgl. Deshpande/Kumar, R. u. a. 2020, S. 3

¹³⁸Vgl. Deshpande/Kumar, R. u. a. 2020, S. 3

¹³⁹Vgl. Furrer u. a. 2016, S. 596

¹⁴⁰Vgl. Furrer u. a. 2016, S. 595

¹⁴¹Vgl. Furrer u. a. 2016, S. 597

¹⁴²Vgl. Furrer u. a. 2016, S. 597

¹⁴³Vgl. Furrer u. a. 2016, S. 598

¹⁴⁴Vgl. Furrer u. a. 2016, S. 598

Sensorik dar.¹⁴⁵ 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.¹⁴⁶

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.¹⁴⁷ Zur Modellierung der Simulationsphysik wird die Unreal Engine 4 aufgrund ihres hohen Grades an physikalischer und visueller Realität eingesetzt.¹⁴⁸ Der Aufbau der Simulation folgt einem modularen Entwurf, welcher unter anderem die einzelnen Komponenten Fahrzeug, Umgebung, Physik-Engine, Sensorik und Darstellungsschnittstelle beinhaltet.¹⁴⁹ Die Schnittstelle des Fahrzeugs erlaubt eine Steuerung über viele Betätigungselemente und deren Eigenschaftsparameter wie Masse, Trägheit, Widerstand oder Reibung.¹⁵⁰ Ein Fahrzeug ist dabei durch die Umgebungskomponente beeinflusst, welche physikalische Effekte wie Gravitation, Luftwiderstand, Luftdruck und magnetische Felder simuliert.¹⁵¹ Die Umgebung wird für das Fahrzeug wahrnehmbar durch die Modellierung von Sensoren wie GPS, Beschleunigungsmesser, Gyroskop, Barometer und Magnetometer.¹⁵²

gym-pybullet-drones

Eine weitere nach der Gym Schnittstelle definierte Simulationsumgebung ist gym-pybullet-drones.¹⁵³ 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.¹⁵⁴ Die Wahl der Physik-Engine wurde aufgrund des CPU und GPU basierenden Renderings, des Kollisionsmanagements und der Kompatibilität mit dem Unified Robot Description Format (URDF) getroffen.¹⁵⁵ Durch die Kompatibilität mit dem URDF Format kann die standardmäßige Simulation des Drohnenmodells Bitcraze Crazyflie 2.x um weitere Nanoquadroptere erweitert werden.¹⁵⁶

¹⁴⁵Vgl. Silano/Iannelli 2019, S. 81

¹⁴⁶Vgl. Silano/Iannelli 2019, S. 82

¹⁴⁷Vgl. Shah u. a. 2017, S. 2

¹⁴⁸Vgl. Shah u. a. 2017, S. 1

¹⁴⁹Vgl. Shah u. a. 2017, S. 3

¹⁵⁰Vgl. Shah u. a. 2017, S. 5

¹⁵¹Vgl. Shah u. a. 2017, S. 6

¹⁵²Vgl. Shah u. a. 2017, S. 9

¹⁵³Vgl. Panerati u. a. 2021, S. 1

¹⁵⁴Vgl. Panerati u. a. 2021, S. 1

¹⁵⁵Vgl. Panerati u. a. 2021, S. 3

¹⁵⁶Vgl. Panerati u. a. 2021, S. 3

2.5 Robustheit und Stabilität von Strategien des verstärkenden Lernens

Anders als innerhalb von 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.¹⁵⁷ 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 sich diese messen lassen.

2.5.1 Definitionen von Robustheit und Stabilität

In der aktuellen Forschungsliteratur findet sich nur eine geringe Gemeinsamkeit innerhalb der unterschiedlichen Definitionen von Stabilität und Robustheit.¹⁵⁸ Die Definition der Robustheit im Kontext von verstärkendem Lernen wird verschieden interpretiert, wie z.B. als Robustheit gegen Störeinflüsse, Beeinflussung der Belohnung, oder Umgebungsunterschiede.¹⁵⁹ 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.*¹⁶⁰

*Robustheit im Kontext von Software, referenziert eine Eigenschaft eines System, welches nicht nur ausschließlich unter normalen, sondern auch unter außergewöhnlichen Bedingungen, die die Annahmen des Entwicklers übersteigen, gut funktioniert.*¹⁶¹

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.¹⁶²

2.5.2 Metriken der Robustheit

Werden Algorithmen und der Erfolg von Veränderungen dieser Experimente betrachtet, kommt es oftmals dazu, dass nur die Leistung und Stabilität verglichen wird, und so die Belohnung die einzige Kenngröße bildet.¹⁶³ Wird diese Metrik im Kontext unterschiedlicher Umgebungen überprüft, kann allerdings so auch die Robustheit von RL Algorithmen betrachtet werden.¹⁶⁴

¹⁵⁷Vgl. Moos u. a. 2022, S. 1

¹⁵⁸Vgl. Pullum 2022, S. 5

¹⁵⁹Vgl. Liu u. a. 2023, S. 2

¹⁶⁰Vgl. Pullum 2022, S. 5

¹⁶¹Vgl. Pullum 2022, S. 5

¹⁶²Vgl. Moos u. a. 2022, S. 1

¹⁶³Vgl. Yan Duan u. a. 2016, S. 6

¹⁶⁴Vgl. Pinto u. a. 2017, S. 6

Neben Betrachtung der Belohnung lassen sich auch weitere quantitative sowie qualitative Kenngrößen untersuchen, welche die Robustheit und Stabilität eines Algorithmus innerhalb der Umgebung widerspiegeln.¹⁶⁵ 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.¹⁶⁶

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 Häufigkeit¹⁶⁷

2.5.3 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 ziehen.¹⁶⁹ Unterschiede in den Umgebungen können bspw. durch fixe Dynamiken wie z.B. Reibwerte während des Trainingsprozesses und unterschiedlicher Reibwerte während der Testphase realisiert werden.¹⁷⁰ 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.¹⁷¹

¹⁶⁵Vgl. Pullum 2022, S. 15

¹⁶⁶Vgl. Pullum 2022, S. 16

¹⁶⁷Ähnlich enthalten in: Pullum 2022, S.17

¹⁶⁸Ähnlich enthalten in: Pullum 2022, S.19

¹⁶⁹Vgl. Pinto u. a. 2017, S. 5

¹⁷⁰Vgl. Pinto u. a. 2017, S. 6

¹⁷¹Vgl. Molchanov u. a. 2019, S. 5

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 Häufigkeit¹⁶⁸

2.6 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.¹⁷² Die Robustheit gegenüber fehlerhafter Umgebungsbeurteilung kann durch Störung des Wahrnehmungszustands des trainierenden Agenten erzielt werden.¹⁷³ Das Ziel ist es dabei, durch die gegnerische Aktion eine veränderte Umgebungswahrnehmung herzustellen, auf deren Basis sich der lernende Agent verbessert.¹⁷⁴

Um die lernende Strategie besser auf unbekannte Situationen und steigende Komplexität vorbereiten zu können, werden destabilisierende Kräfte in der Dynamik eingeführt.¹⁷⁵ Anders als beim ersten Ansatz werden dafür nicht nur lediglich die Wahrnehmung des lernenden Agenten

¹⁷²Vgl. Schott/Hajri/Lamprier 2022, S. 2

¹⁷³Vgl. Schott/Hajri/Lamprier 2022, S. 2

¹⁷⁴Vgl. Schott/Hajri/Lamprier 2022, S. 3

¹⁷⁵Vgl. Pinto u. a. 2017, S. 1

beeinflusst, sondern direkte Einflüsse auf die Umgebung ausgeübt.¹⁷⁶ Hierbei wird der gegnerische Agent dafür belohnt, mittels Kräfteinfluss die Umgebungsdynamik zu verändern, so dass der lernende Agent an seiner Aufgabe scheitert.¹⁷⁷ Dazu kann ein zusätzlicher Agent mit z.B. gleichem Aktionsraum den gemeinsamen Umgebungszustand beeinträchtigen.¹⁷⁸ 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.¹⁷⁹ Zum Generieren von Störeinflüssen auf die Umgebungsdynamik kann das *Robust Adversarial Reinforcement Learning* (RARL) Framework verwendet werden.¹⁸⁰ Dabei wird der gegnerische Agent selbst durch RL daran angelernt, die möglichst effektivsten Destabilisierungsmaßnahmen zu finden.¹⁸¹ Formal dargestellt folgt dieses gegnerische Spiel der in Formel sieben angeführten Minimax Optimierung.¹⁸²

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

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.¹⁸³ Das Training erfolgt aus der abwechselnden Optimierung einer der beiden Strategien bis zu deren Konvergenz, während die jeweils andere nicht verändert wird.¹⁸⁴

Im Kontext der Steuerung von Quadroptern 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.¹⁸⁵ Das getrennte Training des normalen und gegnerischen Agenten verwendet kopierte Agenten des jeweiligen Gegenparts, welche zwar die selben neuronalen Netzstrukturen und initialen Parameter besitzen, jedoch schwächer sind, da deren Lernfortschritt ein Zeitschritt verschoben ist.¹⁸⁶

Abbildung vier zeigt den beschriebenen getrennten Aufbau des RAEARL Frameworks, in dem die Strategien der originalen Agenten mit π_{θ} bzw. π_{ϕ^i} und jene kopierte Strategien mit $\tilde{\pi}_{\theta}$ bzw. $\tilde{\pi}_{\phi^i}$ notiert sind.¹⁸⁸ Zur Evaluation jeder Epoche und der Bestimmung der kumulierten Belohnung G werden die deterministischen Strategien $\bar{\pi}_{\theta}$ bzw. $\bar{\pi}_{\phi^i}$ einbezogen.¹⁸⁹

¹⁷⁶Vgl. Schott/Hajri/Lamprier 2022, S. 2

¹⁷⁷Vgl. Pinto u. a. 2017, S. 2

¹⁷⁸Vgl. Pinto u. a. 2017, S. 2

¹⁷⁹Vgl. Pan u. a. 2021, S. 2

¹⁸⁰Vgl. Schott/Hajri/Lamprier 2022, S. 2

¹⁸¹Vgl. Pinto u. a. 2017, S. 1

¹⁸²Vgl. Pinto u. a. 2017, S. 3

¹⁸³Vgl. Pinto u. a. 2017, S. 3f.

¹⁸⁴Vgl. Pinto u. a. 2017, S. 4

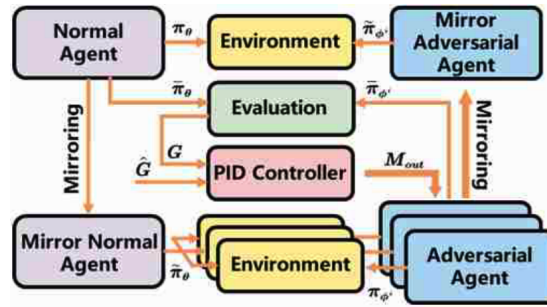
¹⁸⁵Vgl. Zhai u. a. 2022, S. 2

¹⁸⁶Vgl. Zhai u. a. 2022, S. 2f.

¹⁸⁷Enthalten in: Zhai u. a. 2022, S. 3

¹⁸⁸Vgl. Zhai u. a. 2022, S. 3

¹⁸⁹Vgl. Zhai u. a. 2022, S.3


Abb. 4: Aufbau des RAEARL Frameworks¹⁸⁷

2.7 Domain Randomization

Domain Randomization (DR) ist eine weitere Methodik die Realitätslücke zwischen dem Simulator und der echten Welt zu schließen, in dem die Simulation vielfach variiert wird während des Trainingsprozesses.¹⁹⁰ Im Gegensatz zur Systemidentifikation als Prozess die Parameter bestmöglich an ein physisches System anzupassen, ermöglicht DR einen geringeren Zeitaufwand und geringere Fehleranfälligkeit.¹⁹¹ Teilweise kann durch diesen Ansatz die Diskrepanz soweit geschlossen werden, dass es eine Datensammlung in der echten Welt nicht weiter bedarf.¹⁹² Anhand von Komponenten der Simulationsumgebung lässt sich DR in visuelle und dynamische Randomisierung unterteilen.¹⁹³ Weitere Methoden die Realitätslücke zu verkleinern sind progressive Netzwerke, inverse Dynamikmodelle und bayessche Methoden.¹⁹⁴

2.7.1 Das Prinzip hinter der Randomisierung von Simulationsumgebungen

DR verfolgt den Ansatz eine hohe Anzahl an Simulationsumgebungen mit randomisierten Eigenschaften zu erzeugen, in welchen die Strategie dahin optimiert wird, in allen Umgebungen ihr Ziel bestmöglich zu erfüllen.¹⁹⁵ Durch das Aussetzen des Agenten gegenüber einer Vielzahl von Umgebungen, wird das Ziel verfolgt den Agenten direkt in die physische Welt übertragen zu können, unter der gleichzeitigen Erwartung einer guten Leistung.¹⁹⁶ Diese Erwartung resultiert aus der Hypothese, dass unter genug Variabilität die echte Welt lediglich eine bereits trainierte Variation darstellt.¹⁹⁷ Dieses Prinzip kann auch der Abbildung fünf entnommen werden, in welcher das physische System im Bereich der Randomisierung der Simulation liegt.

¹⁹⁰Vgl. Bharadhwaj u. a. 2019, S. 3

¹⁹¹Vgl. Tobin u. a. 2017, S. 1

¹⁹²Vgl. Molchanov u. a. 2019, S. 2

¹⁹³Vgl. Zhao/Queralta/Westerlund 2020, S. 5

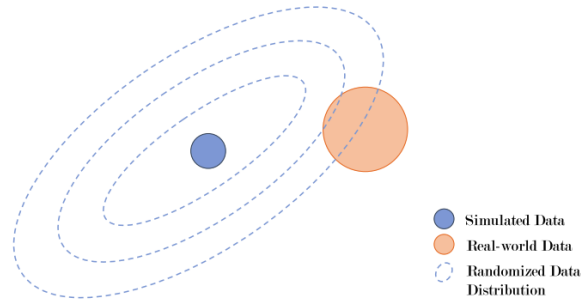
¹⁹⁴Vgl. Chen u. a. 2021, S. 2

¹⁹⁵Vgl. Hsu u. a. 2023, S. 1

¹⁹⁶Vgl. Chen u. a. 2021, S. 2

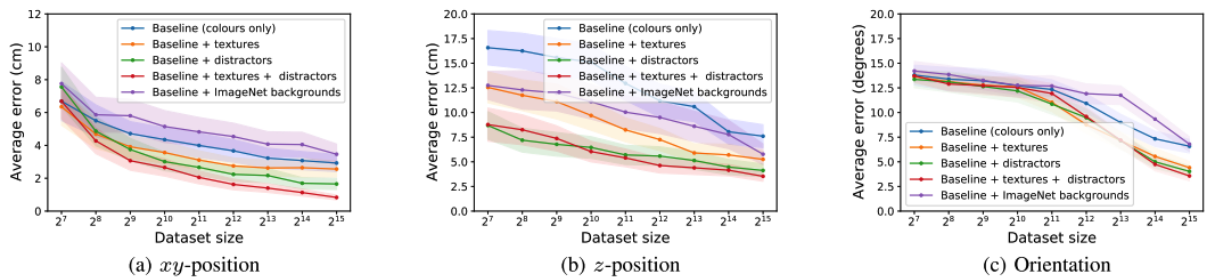
¹⁹⁷Vgl. Tobin u. a. 2017, S. 1

¹⁹⁸Enthalten in: Zhao/Queralta/Westerlund 2020, S. 6


Abb. 5: Intuition hinter dem Paradigma von DR¹⁹⁸

2.7.2 Anwendung von Randomisierung der Simulationsumgebung

Die Anwendung von DR beginnt nach der Erstellung einer möglichst realen Simulation zum Sicherstellen, dass spätere Variation die wirkliche Welt abbilden können.¹⁹⁹ 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.²⁰⁰ Aspekte welche im Rahmen von dynamischer DR randomisiert werden können, sind z.B. die Massen von Objekten, die Dauer eines Zeitschrittes oder Schubkoeffizienten von Rotatoren.²⁰¹ 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.²⁰² Alghonaim/Johns 2021 untersuchten dazu detailliert den Einfluss der Randomisierung von Hintergrundfarben und -texturen sowie Ablenkungsobjekte auf die Modellleistung und erhielten folgende Ergebnisse.


Abb. 6: Einfluss der Randomisierung verschiedener Effekte auf die Fehlerrate der Objekterkennung²⁰³

Aus der Untersuchung in Abbildung sechs 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.

¹⁹⁹Vgl. Chen u. a. 2021, S. 4

²⁰⁰Vgl. Chen u. a. 2021, S. 4

²⁰¹Vgl. Molchanov u. a. 2019, S. 4

²⁰²Vgl. Tobin u. a. 2017, S. 3

²⁰³Enthalten in: Alghonaim/Johns 2021, S. 6

2.7.3 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 ausschließlich Simulationen mittelmäßiger Qualität zu trainieren und für reale Drohnenflüge im Innenbereich einzusetzen.²⁰⁴ Weiterhin konnte erreicht werden, ausschließlich aus dem Training in Simulationen eine Objekterkennung und Lokalisation zu einer Genauigkeit von 1,5cm zu realisieren.²⁰⁵ Neben der Anwendung von DR zeigt die aktuelle Literatur zwar viel Forschung zu der Frage, welche Randomisierung positive Effekte erzielen kann, jedoch wenig Forschung zu detaillierten Erklärungen der Funktionsweise.²⁰⁶ Dies erschwert die Entwicklung effizienter Simulationen und die Bestimmung der Zufallsverteilungen zur Randomisierung.²⁰⁷

²⁰⁴Vgl. Sadeghi/Levine 2016, S. 1

²⁰⁵Vgl. Tobin u. a. 2017, S. 1

²⁰⁶Vgl. Zhao/Queralt/Westerlund 2020, S. 6

²⁰⁷Vgl. Zhao/Queralt/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 durchgeführt, um folgende Forschungsfrage zu beantworten:

Kann durch den Einsatz eines mittels RL trainierten Gegenspielers die Robustheit der gelernten Policy verbessert werden?

Dazu wird die in der Einleitung beschriebene Zielsetzung bearbeitet, ein Experiment zur Robustheit der erlernten Strategien unter deterministischem und trainiertem Gegenspieler durchzuführen. In der ersten Sektion wird auf den Aufbau des Experiments eingegangen, indem beschrieben wird, welche Metriken gemessen werden und wie der Einfluss auf die Laborumgebung ausgeübt wird. Anschließend werden daraus resultierende Anforderungen für die Entwicklung der Testumgebung abgeleitet und in der nachfolgenden Sektion dargelegt. Die dritte Sektion setzt daraufhin die beschriebenen Anforderungen um und erläutert die endgültige Implementierung des Experiments sowie dessen Auswertung.

3.1 Erläuterung der Forschungsmethodik

Wie werden die Daten

3.1.1 Hypothesen

Hypothesen aus Kapitel eins als H_0 und H_1 Hypothesen formulieren.

3.1.2 Messverfahren

Welche Daten werden gemessen ?

Welche tests werden verwendet ?

3.2 Anforderungen der Test- und Simulationsumgebung

3.2.1 Anforderungen der Simulationsumgebung

3.2.2 Anforderungen des Experimentverfahrens

3.3 Programmumsetzung der Simulationsumgebung

3.4 Programmumsetzung zur Experimentdurchführung

4 Ergebnisse des Laborexperiments

5 Reflexion und Forschungsausblick

Anhang

Anhangverzeichnis

Anhang 1	Interview Transkripte	33
Anhang 1/1	Interview Transkript: Mitarbeiter eines Unternehmens	33

Anhang 1: Interview Transkripte

Anhang 1/1: Interview Transkript: Mitarbeiter eines Unternehmens

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.
- 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.
- 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.
- 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.
- 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.
- 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.
- Holzweißig, K. (2022):** Wissenschaftliches Arbeiten. In: 6.6.
- 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>.
- 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].

- Maria, A. (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.
- 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].
- 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_{\text{underscore}}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 Policies anhand trainiertem Gegenspieler in Drohnensimulationen* 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)