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

Bachelorarbeit

vorgelegt am 18. März 2023

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs WWI2020F

von

Leon Henne

Betreuerin in der Ausbildungsstätte:

IBM Deutschland GmbH
Sophie Lang
Senior Data Scientist

Unterschrift der Betreuerin

DHBW Stuttgart:

Prof. Dr. Kai Holzweißig
Studiendekan Wirtschaftsinformatik

Inhaltsverzeichnis

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

EARL 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

Abbildungsverzeichnis

Tabellenverzeichnis

[1] Ivaldi/Padois/Nori 2014, S. 4

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.[2] RL ist im Bereich des maschinellen Lernens eine Herangehensweise zur Lösung von Entscheidungsproblemen.[3] 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.[4] 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.[5] 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.[6] 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.[7] Simulationen nehmen damit zum einen als Testumgebung eine wichtige Rolle ein in der Entwicklung von Kontrollalgorithmen.[8] Zum anderen bedarf die erfolgreiche Anwendung von RL neben effizienten Algorithmen eben auch geeignete Simulationsumgebungen.[9] 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 generalisierten Policies in der Realität eingesetzt werden können.[10] In der Forschungsliteratur wird diese beschriebene Problematik als „Sim to real"-Transfer beschrieben.[11]

Ein Forschungsgebiet, bei dessen die Lösung des Sim to Real Transfers betrachtet wird, ist die autonome Steuerung von unbemannten Luftfahrzeugen bzw. Drohnen.[12] Das Transferproblem entsteht bspw. dabei, dass zur automatisierten Kollisionsvermeidung die Kollisionsbeispiele einer Simulationsumgebung entnommen werden, um physischen Schaden oder Drohnenverlust zu vermeiden.[13] Drohnen tragen dabei bereits in der heutigen Zeit zur Lösung vieler komplexer Aufgaben bei, wie der Katastrophenüberwachung oder der Waldbrandbekämpfung.[14] Zusätzlich

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

steigen immer weiter die komplexen Einsatzanforderungen unter dem Anstieg an Anwendungsgebieten für unbemannte Luftfahrzeuge.[15]

Die Simulation einer möglichst realistischen Umgebung in diesem Kontext wird in der Forschung häufig mit dem Ansatz von Domain Randomization (DR) begleitet.[16] 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.[17] 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 Markt.[18] 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 generalisierendes 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 folgend 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.

[15] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 1
[16] Vgl. Sadeghi/Levine 2016, S. 1
[17] Vgl. Zhao/Queralta/Westerlund 2020, S. 4f.
[18] Vgl. 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 under 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 under 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 under 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-deduktives Modell vorgegangen, in welchem Hypothesen formuliert, empirische Studien entwickelt, Daten gesammelt, Hypothesen anhand dessen evaluiert und gewonnene Erkenntnisse berichtet werden.[19] Eine Möglichkeit der Untersuchung der Ursache- und Wirkungsbeziehung stellt das Laborexperiment

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

dar.[20] Dabei wird die kontrollierte Umgebung der Simulation erschaffen, deren 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.

[20] Vgl. Recker 2021, S. 106

2 Diskussion des aktuellen Stands der Forschung und Praxis

2.1 Aufbau der Literaturrecherche

In Anlehnung der 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					
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		
Furrer u. a. 2016				X				
Silano/Iannelli 2019				X				
Shah u. a. 2017				X				
Panerati u. a. 2021			X	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	

Tab. 1: Konzept Matrix 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 (Drohnensimulation), KS (kompetitive Simulationsumgebungen), DR (Domain Randomization), RRLP (Robustheit von RL Policies), LE (Laborexperimente)

2.2 Verstärkendes Lernen

Verstärkendes Lernen oder als RL in der Fachsprache bezeichnet, definiert einen konzeptionellen Ansatz zielorientiertes Lernen von Entscheidungen zu verstehen und zu automatisieren.[21] Dabei besteht der Fokus darauf, dass ein Agent aus der direkten Interaktion mit seiner Umgebung lernt, ohne das explizite Überwachung notwendig ist.[22] Der Agent lernt über die Zeit eine optimale Strategie zur Lösung des Entscheidungsproblems aus dem Ausprobieren und Scheitern mittels verschiedener Aktionen die gewünschte Veränderung in seiner Umwelt herzustellen.[23] 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.[24] Zur Erreichung dieses Zielzustandes muss der Agent alle Aktionen entdecken, welche ihm die größtmögliche kumulierte Belohnung liefern, wobei Aktionen nicht nur die unmittelbare sondern auch zukünftige Belohnungen beeinflussen.[25] Zusammengefasst lässt sich die beschriebene Interaktion des Agenten mit seiner Umgebung wie folgt in Abbildung 1 darstellen.

[21] Vgl. Sutton/Barto 2018, S. 13
[22] Vgl. Sutton/Barto 2018, S. 13
[23] Vgl. Li 2019, S. 4
[24] Vgl. Sutton/Barto 2018, S. 2
[25] Vgl. Sutton/Barto 2018, S. 1
[26] Enthalten in: Li 2019, S. 5

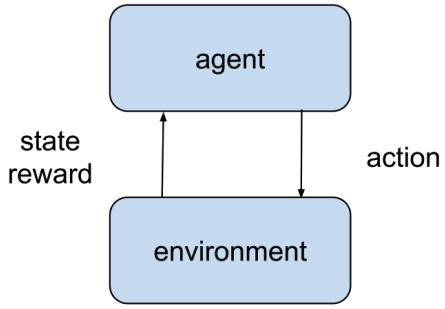

Abb. 1: vereinfachte Darstellung der Interaktion zwischen dem Agenten und seiner Umgebung[26]

Ein Standardaufbau einer Aufgabe für verstärkendes Lernen kann demnach verstanden werden, als sequentielles Entscheidungsproblem zu dessen Lösung ein Agent zu jedem diskreten Zeitschritt eine Aktion ausführt, welche den Zustand der Umgebung verändert.[27] 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:[28]

- Alle Zustände S
- Alle Aktionen A
- initiale Zustandsverteilung $p0(S)$
- 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.[29] Bei modellbasierenden Algorithmen wird das Umgebungsverhalten, also die Übergangs- und Belohnungswahrscheinlichkeiten als bekannt vorausgesetzt.[30] Unter modellbasierenden Algorithmen wird dynamische Programmierung eingesetzt, um mittels Strategieevaluation und Strategieiteration die optimale Strategie zu finden.[31] Unter modellfreien Algorithmen werden die drei verschiedenen Ansätze Wertebasierend, Strategiebasierend und Akteur-Kritiker basierend unterschieden[32] Der Agent im Kontext von modellfreien RL Methoden kennt nur die Zustände S und die Aktionen A, jedoch nicht die Umgebungsverhalten T und die Belohnungswahrscheinlichkeit R.[33] Fasst man die Klassifizierung der Algorithmen und Methoden von RL zusammen, lässt sie sich wie folgt darstellen:

[27] Vgl. Zhao/Queralta/Westerlund 2020, S. 2
[28] Vgl. Zhang/Wu/Pineau 2018, S. 2
[29] Vgl. Wang/Hong 2020, S. 3
[30] Vgl. Wang/Hong 2020, S. 3
[31] Vgl. Li 2019, S. 5
[32] Vgl. Li 2019, S. 5
[33] Vgl. Cutler/Walsh/How 2014, S. 2
[34] Enthalten in: Canese u. a. 2021, S. 6

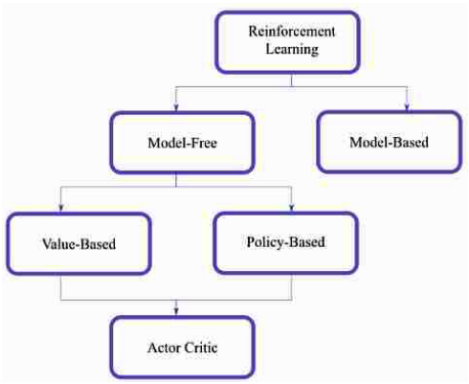

Abb. 2: Klassifizierung von Algorithmen im Bereich des RL[34]

2.2.1 Methoden des verstärkenden Lernens

Der Agent sucht in diesem 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.[35] Neben dieser kurzfristigen direkten Belohnung müssen auch die langfristigen zukünftigen Belohnungen aus den neuen Zuständen betrachtet werden, wofür das Konzept der Wertigkeit eingeführt wird.[36] Über eine Zustands- oder Aktionswertigkeitsfunktion, oftmals als Q-Funktion referenziert, wird eine Vorhersage über die zu erwartende kumulierte abgezinste zukünftige Belohnung berechnet.[37] Durch den Abzinsungsfaktor $\gamma \in [0, 1)$ wird der Einfluss zukünftiger Belohnungen nach ihrer zeitlichen Reihenfolge priorisiert.[38] 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.[39] Die Wertigkeitsfunktion und die daraus berechneten Wertigkeiten von Aktionen oder Zuständen werden über alle Zeitschritte neu geschätzt und stellen mit die wichtigste Komponenten in Algorithmen des verstärkenden Lernens dar.[40] Methoden basierend auf diesem Wertigkeitswert lernen eine Schätzfunktion der Wertigkeit für alle Zustände $(V_\pi(s) \forall S)$ und alle Zustandsaktions-Paare $(Q_\pi(s_t, a_t) \forall s, a \in (S, A))$ der optimalen Strategie π^* durch aktualisieren der folgenden Funktionen eins und zwei:[41]

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

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

Aus den geschätzten Wertigkeit jedes Zustandsaktions Paares kann die optimale Strategie $\pi^*(s)$

[35] Vgl. Reda/Tao/van de Panne 2020, S. 2
[36] Vgl. Wang/Hong 2020, S. 3
[37] Vgl. Li 2019, S. 5
[38] Vgl Li 2019, S. 5
[39] Vgl. Sutton/Barto 2018, S. 6
[40] Vgl. Sutton/Barto 2018, S. 6f.
[41] Vgl. Zhang/Wu/Pineau 2018, S. 2

durch $\arg\max_a Q(s,a)$ bestimmt werden.[42]

Methoden, welche die Strategie durch deren direkte Parametrisierung anstelle einer Bewertung aller Handlungsalternativen mittels Wertigkeitsfunktion optimieren, werden als strategiebasierend bezeichnet.[43] Diese Methodik kann beim Trainieren deterministischer Strategien zu unerwarteten Aktionen führen, weshalb häufig das Optimieren einer Wahrscheinlichkeitsverteilung für alle Aktionen bevorzugt wird.[44] Als Subklasse der RL Methoden wird der statistische Gradientenabstieg verwendet um die parametrisierte Strategie π_θ hinsichtlich der maximalen langfristigen kumulierten Belohnung zu optimieren.[45] Die Strategie π_θ oder auch $\pi(a|s,\theta)$ beschreibt dabei die Wahrscheinlichkeit Aktion a im Zustand s auszuwählen unter dem Parametervektor θ.[46] 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.[47]

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

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

Zusammengefasst kann Formel vier und fünf dabei so interpretiert werden, dass die logarithmierte Wahrscheinlichkeit Aktion a_t im Zustand s_t auszuwählen erhöht werden soll, wenn a_t in einer höheren kumulierten Belohnung resultiert.[48]

Unter Akteur-Kritiker Methoden werden hybride wertebasierende und strategiebasierende Methoden verstanden, welche zugleich die Strategie optimieren und eine Wertefunktion approximieren.[49] 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.[50] Der Akteur wählt somit aus seiner Wahrscheinlichkeitsverteilung die auszuführende Aktionen aus, während der Kritiker diese anhand seiner Wertigkeit bewertet.[51] Betrachtet man den Trainingsprozess von Akteur-Kritiker basierten Methoden ist dieser wie folgt aufgebaut:[52]

- 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 dessen Wertigkeit mittels Q-Funktion

[42] Vgl. Zhang/Wu/Pineau 2018, S. 2
[43] Vgl. Zhang/Wu/Pineau 2018, S. 2
[44] Vgl. Ningombam 2022, S. 3
[45] Vgl. Ningombam 2022, S. 3
[46] Vgl. Sutton/Barto 2018, S. 321
[47] Vgl. Wang/Hong 2020, S. 6
[48] Vgl. Wang/Hong 2020, S. 6
[49] Vgl. Zhang/Wu/Pineau 2018, S. 2f.
[50] Vgl. Sutton/Barto 2018, S. 321
[51] Vgl. Ningombam 2022, S. 3
[52] Vgl. Ningombam 2022, S. 4

- 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, der 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).[53]

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.[54] Als wertbasierte Methode wird für jeden Zustand die Wertigkeit berechnet, was unter DQL mittels eines neuronalen Netzes erreicht wird, welches die Q-Funktion darstellt.[55] Durch Auswählen der Aktion mit der höchsten Wertigkeit in jedem Zustand, kann die deterministische Strategie abgeleitet werden.[56] DQL adressiert das Instabilitätsproblem von Funktionsapproximation unter dem Einsatz von Erfahrungswiederholung und Zielnetzwerke.[57] 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 neuer Erfahrung anzupassen.[58] Durch die Erfahrungswiederholung wird eine bessere Dateneffizienz zum einen durch die Wiederholung erzielt, und zum anderen durch das Auflösen von Korrelationen zwischen aufeinanderfolgenden Zuständen.[59] 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.[60] Die Stärke von DQL liegt in der kompakten Repräsentation der Q-Funktion und der hochgradig dimensionierten Zustandsbeobachtungen durch neuronale Netze.[61] Trust Region Policy Optimization (TRPO) ist ein strategiebasierender Gradientenalgorithmus zur effektiven Optimierung großer nicht linearer Strategien wie z.B. neuronale Netze.[62] Der Algorithmus weist dabei die zwei Varianten single-path, welcher im modellfreien Kontext angewendet werden kann, und *vine* auf, welcher

[53] Vgl. Arulkumaran u. a. 2017, S. 1
[54] Vgl. Arulkumaran u. a. 2017, S. 6
[55] Vgl. Huang u. a. 2017, S. 4
[56] Vgl. Huang u. a. 2017, S. 4
[57] Vgl. Arulkumaran u. a. 2017, S. 7
[58] Vgl. Mnih/Kavukcuoglu u. a. 2013, S. 4
[59] Vgl. Mnih/Kavukcuoglu u. a. 2013, S. 4f.
[60] Vgl. Arulkumaran u. a. 2017, S. 7
[61] Vgl. Arulkumaran u. a. 2017, S. 7
[62] Vgl. Schulman/Levine u. a. 2015, S. 1

sich nur in Simulationen eignet, da das System zu bestimmten Zuständen gespeichert wird.[63] 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.[64] Die daraus entstehenden Anpassungen der Strategieparameter θ zwischen der alten und neuen Strategie werden mittels Kullback-Leibler-Divergenz bemessen und kontrolliert.[65] Eine Weiterentwicklung des TRPO Algorithmus ist Proximal Policy Optimization (PPO), durch dessen Verwendung die Implementierung erleichtert und die Datenprobenkomplexität verbessert wird.[66] 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.[67] Durch diese Anpassung der KL-Divergenz Beschränkung ermöglicht der PPO Algorithmus mehrfache Berechnungen des Gradienten anhand eines Datenobjektes.[68]

Zusätzlich zu den bisherigen Algorithmen kann durch jeweilige asynchrone Varianten, welche den Trainingsprozess des Agenten parallelisieren, die Stabilität des Trainings verbessert werden.[69] 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.[70] 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.[71] Wird der mit dem Algorithmus lediglich ein Agent trainiert, wird dies häufig auch als advantage actor-critic (A2C) referenziert.[72] Neben A3C ist der soft actor-critic Algorithmus ein weiterer Algorithmus der Akteur-Kritiker Methodik, welcher die kumulierte Belohnung, aber auch die Entropie maximiert.[73] 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.[74] Zur Evaluation wird die auf der Belohnung und Entropie basierten Wertigkeit iterativ anhand des Bellman Operators nach Bellman 1966 bestimmt, und anschließend die Strategie hinsichtlich der exponentialen Q-Funktion angepasst.[75] 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.[76]

[63] Vgl. Schulman/Levine u. a. 2015, S. 1
[64] Vgl. Schulman/Levine u. a. 2015, S. 4
[65] Vgl. Huang u. a. 2017, S. 4
[66] Vgl. Schulman/Wolski u. a. 2017, S. 1
[67] Vgl. Schulman/Wolski u. a. 2017, S. 3f.
[68] Vgl. Schulman/Wolski u. a. 2017, S. 4
[69] Vgl. Mnih/Badia u. a. 2016, S. 1
[70] Vgl. Mnih/Badia u. a. 2016, S. 1
[71] Vgl. Arulkumaran u. a. 2017, S. 9
[72] Vgl. Arulkumaran u. a. 2017, S. 9
[73] Vgl. Haarnoja u. a. 2018, S. 1
[74] Vgl. Haarnoja u. a. 2018, S. 3
[75] Vgl. Haarnoja u. a. 2018, S. 4
[76] Vgl. Haarnoja u. a. 2018, S. 4

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.[77] Dieser Ansatz dient dazu Problemstellungen welche nicht vollständig durch einen Agenten lösbar sind zu bearbeiten.[78] Einsatzgebiete von MARL sind dabei unter anderem das Routing von Netzwerkpaketen, Wirtschaftsmodellierung oder zusammenhängende Robotersysteme.[79] Je nach Ziel und der demnach definierter Belohnungsfunktion können die Agenten auf die drei unterschiedlichen Arten vollständig kooperativ, vollständig kompetitiv und der Mischung aus beiden miteinander interagieren.[80] Aus der unterschiedlichen 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.[81] Da dieser Effekt auch die Annahme der Stationarität von Markov Entscheidungsprozessen verletzt, bedarf dieUmgebung auch einer anderen Representation.[82] Ein Konzept was dafür häufig verwendet ist das Markov Spiel, welches sich anders als der Entscheidungsprozess durch einen mehrdimensionalen Aktions- und Belohnungsraum aus der Kombination aller N Agenten auszeichnet.[83] Betrachtet man die Limitierungen von MARL erkennt man aus den beschriebenen Punkten die Herausforderungen der nicht vorhandenen Stationarität und der Skalierbarkeit, welcher sich die Herausforderung der teilweisen Beobachtbarkeit der Umgebung anschließt.[84]

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.[85]

Eine der Herausforderungen zeigt sich bei der Representation der Agentenumwelt, da RL stark auf diesem Konzept basiert.[86] Daraus ergibt sich die Aufgabe, die Umwelt und dessen Verhalten sowie die Wahrnehmung durch den Agenten realitätsgetreu und präzise zu gestalten.[87] 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.[88] Zusätzlich teilen RL Algorithmen auch Herausforderungen aus anderen Gebieten des

[77] Vgl. Wong u. a. 2022, S. 6
[78] Vgl. Canese u. a. 2021, S. 1
[79] Vgl. Canese u. a. 2021, S. 1
[80] Vgl. Canese u. a. 2021, S. 8f.
[81] Vgl. Wong u. a. 2022, S. 2
[82] Vgl. Wong u. a. 2022, S. 6
[83] Vgl. Canese u. a. 2021, S. 4
[84] Vgl. Canese u. a. 2021, S. 9ff.
[85] Vgl. Li 2019, S. 7
[86] Vgl. Sutton/Barto 2018, S. 8
[87] Vgl. Sutton/Barto 2018, S. 7
[88] Vgl. Li 2019, S. 7

maschinellen Lernens wie Genauigkeit, Interpretierbarkeit und die im Rahmen dieser Arbeit untersuchte Robustheit von Modellen.[89]

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 Exploration neuer Strategie und der Optimierung bekannter Verhaltensmuster ein wichtiger Bestandteil ist.[90] Aufgrund der geringen Effizienz der Daten aber des dennoch hohen Bedarfs an bewerteter Agentenerfahrung wird häufig auf simulierte Daten zurückgegriffen.[91] 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.[92]

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 überwachtes- und unüberwachtes Lernen, werden beim verstärkenden Lernen viele der Testdatensätze nicht aus der echten Welt akquiriert.[93] Um entsprechend realistische Daten für das Training bereitzustellen, werden Simulationsumgebungen in Abhängigkeit von ihrer RL Anwendung ausgewählt.[94] Dennoch bleibt nahezu immer eine gewisse Diskrepanz zwischen der Dynamik in der Simulation und der Dynamik in der echten Welt.[95] Daher ist es kaum garantiert, dass erlernte Strategien der Agenten sich auch auf nur leicht veränderte Umgebungen übertragen lassen.[96]

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

[89] Vgl. Li 2019, S. 7
[90] Vgl. Li 2019, S. 7
[91] Vgl. Zhao/Queralta/Westerlund 2020, S. 7
[92] Vgl. Li 2019, S. 8
[93] Vgl. Zhang/Wu/Pineau 2018, S. 1
[94] Vgl. Körber u. a. 2021, S. 7
[95] Vgl. Bharadhwaj u. a. 2019, S. 1
[96] Vgl. Bharadhwaj u. a. 2019, S. 1

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.[97]

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:[98]

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.[99] 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.[100] Seither wurde diese definierte Schnittstelle vielfach verwendet, um RL Umgebungen mit dem Ziel zu entwickeln, diese zu publizieren und dessen Wiederverwendung zu ermöglichen.[101] 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.[102] Der Werkzeugkasten von OpenAI fokussiert sich auf einen Episoden ähnlichen Rahmen,

[97] Vgl. Foronda 2021, S. 1
[98] Vgl. Maria 1997, S. 8f.
[99] Vgl. Brockman u. a. 2016, S. 1
[100] Vgl. Brockman u. a. 2016, S. 1
[101] Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 4
[102] Vgl. Schuderer/Bromuri/van Eekelen 2021, S. 4

in welchem der Agent durch zunächst zufälliges Auswählen von Interaktionen lernt.[103] Weitere Entwicklungsentscheidungen des OpenAI Gym Werkzeugkastens umfassen z.B. die bewusst fehlende Schnittstelle des Agenten, die strikte Versionierung der Umgebung oder die standardmäßige Simulationsüberwachung.[104]

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.[105] Eine enge initiale Wahrscheinlichkeitsverteilung des Umgebungszustandes kann die Lerneffizienz erhöhen, wohingegen eine weite Wahrscheinlichkeitsverteilung positiv die Robustheit der erlernten Strategie beeinflusst.[106] 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.[107] 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.[108]

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 Simulatoren[109]

Aus Tabelle zwei lässt sich entnehmen, das 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.[110]

[103] Vgl. Brockman u. a. 2016, S. 1
[104] Vgl. Brockman u. a. 2016, S. 2f.
[105] Vgl. Reda/Tao/van de Panne 2020, S. 1
[106] Vgl. Reda/Tao/van de Panne 2020, S. 3
[107] Vgl. Yan Duan u. a. 2016, S. 2
[108] Vgl. Reda/Tao/van de Panne 2020, S. 7
[109] Enthalten in: Ivaldi/Padois/Nori 2014, s. 4
[110] Vgl. Ayala u. a. 2020, S. 2

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.[111] Mittels Gazebo lassen sich Interaktionen zwischen Robotern in Innen-und Außenbereichen unter realistischer Sensorik simulieren.[112] Die unterstützten Physik-Engines umfassen Bullet, Dynamic Animation and Robotics Toolkit (DART), Open Dynamics Engine (ODE) and Simbody.[113]

MuJoCo

MuJoCo stellt eine Physik-Engine für modellbasierte Steuerung dar, dessen Objekte durch C++ oder XML definiert und Gelenkzustände im Koordinatensystem beschriebenen werden.[114] Diese beschriebene Eigenschaft lässt sich auch aus dem Namen als Abkürzung für **Mu**lti-**Jo**int dynamics with **Co**ntact ableiten.[115] Die MuJoCo Anwendung ist lizensiert, was diesen Besitz einer Lizenz für die Installation oder die Virtualisierung innerhalb eines Containers voraussetzt.[116]

PyBullet

PyBullet basiert als Simulationssoftware auf der Bullet-Engine und fokussiert funktional auf die Anwendung von RL im Robotikbereich.[117] Die Bullet-Engine ist hingegen eine offene Softwarebibliothek welche neben verstärkenden Lernen auch bei Computeranimationen angewendet wird.[118] 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.[119]

2.4 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.[120] Anwendungsge-

[111] Vgl. Ivaldi/Padois/Nori 2014, S. 7
[112] Vgl. Ayala u. a. 2020, S. 4
[113] Vgl. Körber u. a. 2021, S. 3
[114] Vgl. Todorov/Erez/Tassa 2012, S. 1
[115] Vgl. Todorov/Erez/Tassa 2012, S. 2
[116] Vgl. Körber u. a. 2021, S. 3
[117] Vgl. Körber u. a. 2021, S. 3
[118] Vgl. Ivaldi/Padois/Nori 2014, S. 7
[119] Vgl. Körber u. a. 2021, S. 6
[120] Vgl. Koch u. a. 2018, S. 1

biete beinhalten z.B. die Landwirtschaft, den Pakettransport oder die Überwachung von großflächiger Infrastruktur wie Stromnetze.[121] Eine Simulation von unbemannten Flugzeugen stellt eine Flugumgebung und vielseite Sensorik bereit, und kann je nach Anwendung Effekte wie Wind, Wolken und Niederschlag einbeziehen.[122]

2.4.1 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.[123] 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.[124] 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.[125] Jede Bewegung der sechs verschiedenen Arten wird durch die unterschiedliche Steuerung, der im Fall des Quadrokopters, vier Rotoren getätigt. Eine unterschiedliche Ansteuerung der Rotoren und den demnach verschieden starken Auftrieben resultiert in den drei Rotationsbewegungen Rollen, Nicken und Gieren.[126]

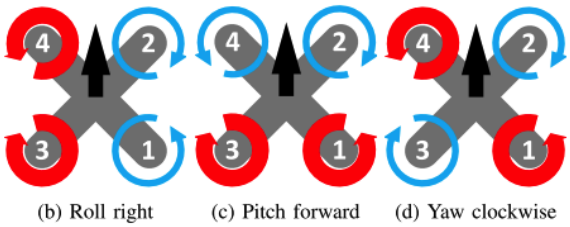

Abb. 3: Rotationsbewegungen eines Quadrokopters[127]

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 repräsentiert als Matrix der Eulerschen Winkel, welche die Folge der einzelnen Drehungen entlang der X-, Y-, und Z-Achse enthält.[128] 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

[121] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 1
[122] Vgl. Hentati u. a. 2018, S. 1496
[123] Vgl. Koch u. a. 2018, S. 2
[124] Vgl. Molchanov u. a. 2019, S. 3
[125] Vgl. Koch u. a. 2018, S. 2
[126] Vgl. Koch u. a. 2018, S. 2
[127] Enthalten in: Koch u. a. 2018, S. 2
[128] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 3

Formel fünf, sorgt für die Bewegung der Drohne in Richtung der zukünftigen Koordinaten nach Formel sechs.[129]

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

In Formel vier wird der Rotationszustand der Drohne zur 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.[130]

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

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.[131] Der Schub, der in der Länge l vom Schwerpunkt entfernten Rotatoren, wird mittels F_i, das Drehmoment mit $M_i, \forall i \in \{1, 2, 3, 4\}$ gekennzeichnet.[132]

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

Weiterhin ist die Beschleunigung $\ddot{x}$, $\ddot{y}$ und $\ddot{z}$ in Richtung der drei Achsen durch die die Masse m und die Gravitation g beeinflusst.[133]

2.4.2 Quadrokopter im Kontext von RL

2.4.3 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.[134] Dabei wurde die Simulation modular entworfen, sodass Komponenten wie Steuerung oder Zustandsschätzung austauschbar sind und das Hinzufügen von neuen Drohnen erleichtert wird.[135] Die Komponenten der Drohne stellen dabei Plug-ins der verwendeten Gazebo Physik-Engine dar, wodurch ein Mikroflugzeug aus den Teilen des Körper, der

[129] Vgl. Deshpande/Minai/Kumar, M. 2021, S. 2
[130] Vgl. Deshpande/Minai/Kumar, M. 2021, S. 2
[131] Vgl. Deshpande/Minai/Kumar, M. 2021, S. 2
[132] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 3
[133] Vgl. Deshpande/Kumar, R. u. a. 2020, S. 3
[134] Vgl. Furrer u. a. 2016, S. 596
[135] Vgl. Furrer u. a. 2016, S. 595

Anzahl der Rotoren und Sensorik an fixen Position zusammengesetzt wird.[136] Mittels der standardmäßigen Sensorik können Informationen über die direkte und visuell gemessene Trägheit sowie über die Wegbestimmung erzielt werden.[137] Anstelle des Wegbestimmungssensors kann auch eine Komponente zur Zustandsschätzung für hochfrequente Abragen implementiert werden.[138] Die Steuerungskomponente wird durch eine einfache Schnittstelle einer geometrischen Steuerung bedient, welche Aktionen in Form von Rotationswinkeln, Höhen oder Positionen entgegen nimmt.[139]

CrazyS

CrazyS stellt eine Erweiterung der Simulation RotorS auf Basis des selben ROS, um die Modellierung des Nano-Quadrokopters Crazyflie samt ihrer Dynamik ihres Kontrollsystems und ihrer Sensorik dar.[140] 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.[141]

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[142] Zur Modellierung der Simulationsphysik wird die Unreal Engine vier aufgrund ihres hohen Grades an physikalischer und visueller Realität eingesetzt.[143] Der Aufbau der Simulation folgt einem modularen Entwurf, welcher unter anderem die einzelnen Komponenten Fahrzeug, Umgebung, Physik-Engine, Sensorik und Darstellungsschnittstelle beinhaltet.[144] Die Schnittstelle des Fahrzeugs erlaubt eine Steuerung über viele Betätigungselemente und deren Eigenschaftsparameter wie Masse, Trägheit, Widerstand oder Reibung.[145] Ein Fahrzeug ist dabei durch die Umgebungskomponente beeinflusst, welche physikalische Effekte wie Gravitation, Luftwiederstand, Luftdruck und magnetische Felder simuliert.[146] Die Umgebung wird für das Fahrzeug wahrnehmbar durch die Modellierung von Sensoren wie GPS, Beschleunigungsmesser, Gyroskop, Barometer und Magnetometer.[147]

gym-pybullet-drones

Eine weitere nach der Gym Schnittstelle definierte Simulationsumgebung ist gym-pybullet-drones.[148] Basierend auf der Bullet Physik-Engine ermöglicht die Simulation unter anderem das visuell ba-

[136] Vgl. Furrer u. a. 2016, S. 597
[137] Vgl. Furrer u. a. 2016, S. 597
[138] Vgl. Furrer u. a. 2016, S. 598
[139] Vgl. Furrer u. a. 2016, S. 598
[140] Vgl. Silano/Iannelli 2019, S. 81
[141] Vgl. Silano/Iannelli 2019, S. 82
[142] Vgl. Shah u. a. 2017, S. 2
[143] Vgl. Shah u. a. 2017, S. 1
[144] Vgl. Shah u. a. 2017, S. 3
[145] Vgl. Shah u. a. 2017, S. 5
[146] Vgl. Shah u. a. 2017, S. 6
[147] Vgl. Shah u. a. 2017, S. 9
[148] Vgl. Panerati u. a. 2021, S. 1

sierte Training von mehreren Agenten mittels RL unter realistischer Modellierung von Kollisionen und aerodynamischen Effekten.[149] 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[150] Durch die Kompabilität mit dem URDF Format kann die standardmäßige Simulation des Drohnenmodells Bitcraze Crazyflie 2.x um weitere Nanoquadrokopter erweitert werden.[151]

2.5 gegnerisches verstärkendes Lernen

Methoden des gegnerischen verstärkenden Lernens verfolgen das Ziel die angelernten Strategien robuster gegenüber Risiken wie beeinflusste Wahrnehmung, unbekannte Situationen oder ansteigende Umgebungskomplexität zu trainieren.[152] Die Robustheit gegenüber fehlerhafter Umgebungsbetrachtung kann durch Störung des Wahrnehmungszustands des trainierenden Agenten erzielt werden.[153] Das Ziel ist es dabei, durch die gegnerische Aktion eine veränderte Umgebungswahrnehmnung herzustellen, zu dessen Basis sich der lernende Agent verbessert.[154]

Um die lernende Strategie besser gegenüber unbekannte Situationen und steigende Komplexität vorzubereiten können destabilisierende Kräfte in der Dynamik eingeführt werden.[155] Anders als beim ersten Ansatz wird dafür nicht nur lediglich die Wahrnehmung für den lernenden Agenten beeinflusst, sondern direkte Einflüsse auf die Umgebung ausgeübt.[156] Hierbei wird der gegnerische Agent dafür belohnt, mittels Kräfteinfluss die Umgebungsdynamik zu verändern, so dass der lernende Agent an seiner Aufgabe scheitert.[157] Dazu kann ein zusätzlicher Agent mit z.B. gleichem Aktionsraum den gemeinsamen Umgebungszustand beeinrächtigen.[158] Pan u. a. 2021 zeigt eine solche Anwendung im Rahmen der Kontrolle eines Stromnetzes, bei welchem ein gegnerischer Agent für das trennen von Verbindungen im Netz belohnt wird.[159] Zum Generieren von Störeinflüssen auf die Umgebungsdynamik kann das *Robust Adversarial Reinforcement Learning* (RARL) Framework verwendet werden.[160] Dabei wird der gegnerische Agent selbst durch RL daran angelernt, die möglichst effektivsten Destabilisierungsmaßnahmen zu finden.[161] Formal dargestellt folgt dieses gegnerische Spiel der in Formel sieben angeführten Minimax Optimierung.[162]

[149] Vgl. Panerati u. a. 2021, S. 1
[150] Vgl. Panerati u. a. 2021, S. 3
[151] Vgl. Panerati u. a. 2021, S. 3
[152] Vgl. Schott/Hajri/Lamprier 2022, S. 2
[153] Vgl. Schott/Hajri/Lamprier 2022, S. 2
[154] Vgl. Schott/Hajri/Lamprier 2022, S. 3
[155] Vgl. Pinto u. a. 2017, S. 1
[156] Vgl. Schott/Hajri/Lamprier 2022, S. 2
[157] Vgl. Pinto u. a. 2017, S. 2
[158] Vgl. Pinto u. a. 2017, S. 2
[159] Vgl. Pan u. a. 2021, S. 2
[160] Vgl. Schott/Hajri/Lamprier 2022, S. 2
[161] Vgl. Pinto u. a. 2017, S. 1
[162] Vgl. Pinto u. a. 2017, S. 3

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

Zu jedem Zeitschritt t in dieser 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.[163] Das Training erfolgt aus der abwechselnden Optimierung einer der beiden Strategien bis zu deren Konvergenz, während die jeweils andere nicht verändert wird.[164]

Im Kontext der Steuerung von Quadrokoptern greifen Zhai u. a. 2022 das *Robust Adaptive Ensemble Adversarial Reinforcement Learning Framework (EARL)* 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-Kontroller erweitert, um die Stärke des Gegners über den Trainingsverlauf anzupassen.[165] 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.[166]

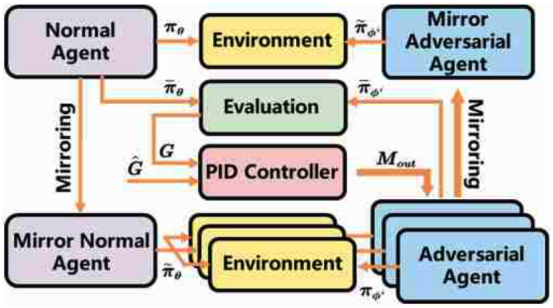

Abb. 4: Aufbau des RAEARL Frameworks[167]

Abbildung vier zeigt den beschrieben getrennten Aufbau des RAEARL Frameworks, in dessen 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.[168] Zur Evaluation jeder Epoche und der Bestimmung der kumulierten Belohnung G werden die deterministischen Strategien $\bar{\pi}_\theta$ bzw. $\bar{\pi}_{\phi^i}$ einbezogen.[169]

[163] Vgl. Pinto u. a. 2017, S. 3f.
[164] Vgl. Pinto u. a. 2017, S. 4
[165] Vgl. Zhai u. a. 2022, S. 2
[166] Vgl. Zhai u. a. 2022, S. 2f.
[167] Enthalten in: Zhai u. a. 2022, S. 3
[168] Vgl. Zhai u. a. 2022, S. 3
[169] Vgl. Zhai u. a. 2022, S.3

3 Durchführung des Laborexperiments

4 Ergebnisse des Laborexperiments

5 Reflexion und Forschungsausblick

Anhang

Anhangverzeichnis

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/.

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.

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.

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.

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.

Ningombam, D. D. (2022): Deep Reinforcement Learning Algorithms for Machine-to-Machine Communications: A Review. In: *2022 13th International Conference on Computing Commu-*

nication 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.

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.

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)