

# Aufgabenorientierte Konfiguration intelligenter Agenten in der Montage

Marcus Röhler

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines  
Doktors der Ingenieurwissenschaften (Dr.-Ing.)  
genehmigten Dissertation.

Vorsitz: Prof. Dr.-Ing. Rüdiger Daub

Prüfende der Dissertation:

1. Prof. Dr.-Ing. Gunther Reinhart
2. Prof. Dr. rer. nat. Tim C. Lüth

Die Dissertation wurde am 18.04.2024 bei der Technischen Universität München eingereicht  
und durch die TUM School of Engineering and Design am 24.12.2024 angenommen.



## Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Gießerei-, Composite- und Verarbeitungstechnik IGCV in Augsburg sowie am Institut für Werkzeugmaschinen und Betriebswissenschaften (*iwb*) der Technischen Universität München (TUM).

Mein besonderer Dank gilt zunächst meinem Doktorvater Herrn Prof. Dr.-Ing. Gunther Reinhart für die wohlwollende Unterstützung dieser Arbeit sowie für die kontinuierliche Förderung meiner Tätigkeit am Institut. Herrn Prof. Dr. rer. nat. Tim C. Lüth, dem Leiter des Lehrstuhls für Mikrotechnik und Medizingerätetechnik an der Technischen Universität München, danke ich für die Übernahme des Korreferats und Herrn Prof. Dr.-Ing. Rüdiger Daub, dem Leiter des Lehrstuhls für Produktionstechnik und Energiespeichersysteme ebenfalls an der Technischen Universität München, danke ich für die Übernahme des Prüfungsvorsitzes.

Darüber hinaus bedanke ich mich bei allen Kolleginnen und Kollegen, welche mich durch ihre kreativen Ideen, zahlreiche konstruktive Diskussionen und eine kollegiale Atmosphäre stets unterstützt haben. Insbesondere möchte ich mich bei Dr.-Ing. Martin Rösch und Dr.-Ing. Julia Berger bedanken, welche mich durch ihren fachlichen Rat in den Gebieten Reinforcement Learning und aufgabenorientierter Programmierung von der Forschungsidee bis zur Durchsicht meiner Arbeit begleitet haben. Außerdem möchte ich mich bei allen Studenten bedanken, die mich bei der Erstellung meiner Arbeit unterstützt haben.

Mein größter Dank gilt meiner Familie und meinen Freunden, die mich in jeder Phase des Promotionsvorhabens gestärkt und motiviert haben. Ohne sie wäre diese Arbeit sicherlich nicht möglich gewesen.

In dieser Arbeit wird nur die männliche Sprachform verwendet. Dies dient lediglich der Beibehaltung eines hohen Maßes an Lesbarkeit. Sämtliche Bezeichnungen gelten jedoch ausdrücklich gleichermaßen für alle Geschlechter.

München, im Februar 2025

Marcus Röhler





# Inhalt

<b>Abkürzungsverzeichnis</b>	<b>VII</b>
<b>Formelzeichenverzeichnis</b>	<b>IX</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Bedeutung der Automation im 21. Jahrhundert . . . . .	1
1.2 Deep Reinforcement Learning in der Montage . . . . .	2
1.3 Zielsetzung der Arbeit . . . . .	3
1.4 Aufbau und wissenschaftliches Vorgehen . . . . .	5
<b>2 Grundlagen intelligenter Montagesysteme</b>	<b>9</b>
2.1 Roboterbasierte Montagesysteme . . . . .	9
2.1.1 Funktionen und Aufbau von Montagesystemen . . . . .	9
2.1.2 Ausprägungen roboterbasierter Montagesysteme . . . . .	11
2.1.3 Sensorik . . . . .	12
2.1.4 Programmierung roboterbasierter Montagesysteme . . . . .	13
2.1.5 Begriffe im Kontext von Varianz . . . . .	15
2.2 Intelligente Robotersysteme . . . . .	16
2.2.1 Intelligente Agenten . . . . .	16
2.2.2 Maschinelles Lernen in der Robotik . . . . .	19
2.3 Deep Reinforcement Learning . . . . .	22
2.3.1 Reinforcement Learning . . . . .	22
2.3.2 Deep Learning . . . . .	25
2.3.3 Lernbare Funktionen in Deep Reinforcement Learning . . . . .	27
2.3.4 Strategiebasierte Lernalgorithmen . . . . .	29
2.4 Zusammenfassung . . . . .	30

<b>3</b>	<b>Stand der Wissenschaft und Technik</b>	<b>33</b>
3.1	Rechnergestützte Umsetzung roboterbasierter Montagesysteme . . . . .	33
3.1.1	Herleitung von Skills in der Montage . . . . .	33
3.1.2	Ansätze zur Vereinfachung der Programmierung von Roboter- systemen . . . . .	35
3.2	Aktuelle Ansätze zur Umsetzung des Agenten . . . . .	38
3.2.1	Proximal Policy Optimization . . . . .	38
3.2.2	Werkzeuge zur Implementierung des Agenten . . . . .	41
3.3	Umsetzung der Lernumgebung . . . . .	42
3.3.1	Ausgewählte montagerelevante Belohnungsfunktionen . . . . .	42
3.3.2	Simulative Lernumgebungen . . . . .	44
3.4	Automatisiertes maschinelles Lernen in Deep Reinforcement Learning .	46
3.4.1	Empirische Studien zur Konfiguration von Hyperparametern . .	46
3.4.2	Automatisierte Hyperparameter-Optimierung . . . . .	48
3.4.3	Meta Learning . . . . .	50
3.5	Zusammenfassung und Handlungsbedarf . . . . .	52
<b>4</b>	<b>Einschränkung des Betrachtungsbereichs und Lösungsansatz</b>	<b>57</b>
4.1	Analyse der Rahmenbedingungen zur Umsetzung intelligenter Systeme in der Industrie . . . . .	57
4.1.1	Studie zum Kompetenzbedarf in produzierenden Unternehmen	57
4.1.2	Vorgehen in der Umsetzung auf RL basierender Systeme . . . .	59
4.1.3	Nutzergruppen von Programmiersystemen in der Montage . . .	62
4.2	Einschränkung des Betrachtungsbereichs . . . . .	64
4.2.1	Handlungsbedarf in der Industrie . . . . .	64
4.2.2	Notwendige Funktionalitäten der aufgabenorientierten Konfi- guration intelligenter Agenten . . . . .	67
4.2.3	Betrachtete Montagefunktionen . . . . .	69
4.2.4	Anforderungen . . . . .	70
4.3	Lösungsansatz . . . . .	71
4.3.1	Auswahl einer Grundarchitektur der aufgabenorientierten Pro- grammierung . . . . .	71
4.3.2	Auswahl eines Lösungsansatzes zur Umsetzung der aufgaben- orientierten Konfiguration intelligenter Agenten . . . . .	74

---

<b>5</b>	<b>Erweiterung des Informationsmodells automatisierter Montagesysteme</b>	<b>77</b>
5.1	Überblick zum Informationsmodell . . . . .	77
5.1.1	Einordnung des Informationsmodells in die Gesamtarchitektur . . . . .	77
5.1.2	Vorgehen zur Modellierung des Informationsmodells . . . . .	78
5.1.3	Elemente des Informationsmodells . . . . .	80
5.2	Abstraktion der Belohnungsfunktionen über Agentenzielle . . . . .	82
5.2.1	Agentenzielle . . . . .	82
5.2.2	Teilziele . . . . .	83
5.2.3	Arten elementarer Ziele . . . . .	86
5.2.4	Belohnungsfunktionen . . . . .	87
5.3	Agenten-Skills . . . . .	90
5.3.1	Zuordnung von Agenten-Skills zu Agentenziellen . . . . .	90
5.3.2	Unterscheidung von Agenten-Skills zu Ressourcen-Skills . . . . .	91
5.4	Schnittstellen des Agenten mit der Umgebung . . . . .	93
5.4.1	Aktionsraum . . . . .	93
5.4.2	Wahrnehmung . . . . .	94
5.5	Modellierung der Varianz . . . . .	96
5.5.1	Ausprägungen variierender Merkmale . . . . .	96
5.5.2	Benutzerdefinierte Merkmale . . . . .	97
5.5.3	Abhängige Merkmale . . . . .	98
5.5.4	Varianz im Informationsmodell . . . . .	99
5.6	Rahmenbedingungen . . . . .	100
5.7	Zusammenfassung . . . . .	101
 <b>6</b>	 <b>Wissensbasierte Entscheidungsunterstützung</b>	 <b>103</b>
6.1	Modellierung der Datenbasis . . . . .	103
6.1.1	Abhängigkeiten der Planungselemente des Aufgabenmodells . . . . .	103
6.1.2	Repräsentation der Aufgabenstellungen . . . . .	105
6.2	Wiederverwendung bestehender Lösungselemente innerhalb der Datenbasis . . . . .	109
6.2.1	Kriterien der Metrik zur Berechnung der Ähnlichkeit . . . . .	109
6.2.2	Abrufen von Lösungselementen aus der Datenbasis . . . . .	110
6.3	Anpassung der Lösungselemente . . . . .	111
6.4	Zusammenfassung . . . . .	113

<b>7</b>	<b>Automatische Konfiguration des Agenten</b>	<b>115</b>
7.1	Übersicht des Ansatzes . . . . .	115
7.2	Ähnlichkeit der Lösungen . . . . .	116
7.2.1	Modellierung der Lösungen . . . . .	116
7.2.2	Bewertung der Ähnlichkeit von Hyperparameter-Konfigurationen	118
7.3	Lernen der Ähnlichkeit von Agentenaufgaben . . . . .	120
7.3.1	Meta-Features . . . . .	120
7.3.2	Abbildung der Semantik über Embeddings . . . . .	122
7.3.3	Ansatz zum Lernen der Ähnlichkeit . . . . .	123
7.3.4	Wahl der Architektur des Ansatzes . . . . .	125
7.3.5	Analyse der Interpretierbarkeit der Case Embeddings . . . . .	127
7.3.6	Analyse des Informationsgehalts der Case Embeddings . . . . .	130
7.4	Module des fallbasierten Schließens . . . . .	132
7.4.1	Retrieve-Phase . . . . .	132
7.4.2	Reuse- und Revise-Phase . . . . .	134
7.4.3	Retain-Phase . . . . .	135
7.5	Zusammenfassung . . . . .	137
<b>8</b>	<b>Umsetzung und Erprobung</b>	<b>139</b>
8.1	Definition des Referenzszenarios . . . . .	139
8.2	Softwaretechnische Umsetzung . . . . .	142
8.2.1	Umsetzung der simulativen Lernumgebung . . . . .	142
8.2.2	Umsetzung der automatischen Konfiguration des Agenten . . . . .	144
8.3	Anwendung im Referenzszenario . . . . .	146
8.3.1	Automatische Konfiguration des Agenten beim kollisionsfreien Anfahren . . . . .	146
8.3.2	Automatische Konfiguration des Agenten beim kamerabasierten Weitergeben . . . . .	148
8.3.3	Bewertung von Lösungsalternativen beim Folgen der Schweiß- bahn . . . . .	149
<b>9</b>	<b>Technische und wirtschaftliche Bewertung</b>	<b>153</b>
9.1	Anforderungserfüllung und technische Grenzen . . . . .	153
9.2	Wirtschaftliche Bewertung . . . . .	156
<b>10</b>	<b>Zusammenfassung und Ausblick</b>	<b>163</b>

<b>Literatur</b>	<b>167</b>
<b>A Anhang</b>	<b>209</b>
A.1 Interview-Fragebogen . . . . .	209
A.2 Zum Aufbau der Fallbasis verwendete Literatur . . . . .	211
A.3 Informationsmodell . . . . .	214
A.4 Versuchsergebnisse . . . . .	215
<b>Verzeichnis betreuter Studienarbeiten</b>	<b>217</b>



## Abkürzungsverzeichnis

---

Abkürzung	Bedeutung
API	Application Programming Interface
AutomationML	Automation Markup Language
AutoML	Automated Machine Learning
CAD	Computer Aided Design
CAEX	Computer Aided Engineering Exchange
CBR	Case-based Reasoning
CIRP	Collège International pour la Recherche en Productique
CNC	Computerized Numerical Control
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
DIN	Deutsches Institut für Normung
DRL	Deep Reinforcement Learning
DSM	Design Structure Matrix
FPGA	Field Programmable Gate Array
GAE	Generalized Advantage Estimation
GAN	Generative Adversarial Network
GPU	Graphical Processing Unit
ID	Identifikator
IEC	International Electrotechnical Commission
IFR	International Federation of Robotics
JSON	JavaScript Object Notation
KI	Künstliche Intelligenz
KMU	Kleine und mittelständische Unternehmen
KNN	Künstliches Neuronales Netz

## Abkürzungsverzeichnis

---

LSTM	Long Short-Term Memory
MAG	Metall-Aktivgas
MCD	Monte Carlo Dropout
MDM	Multiple Domain Matrix
MDP	Markov Decision Process
MIG	Metall-Inertgas
ML	Maschinelles Lernen
MLP	Multilay Perception
MuJoCo	Multi-Joint Dynamics with Contact
NLL	Negative Log-Likelihood
ODE	Open Dynamics Engine
ONNX	Open Neural Network Exchange
OWL	Web Ontology Language
PCA	Principal Component Analysis
POMDP	Partially Observable Markov Decision Process
PPO	Proximal Policy Optimization
PPR	Produkt-Prozess-Ressource
RGB	Rot-Grün-Blau-Farbraum
RL	Reinforcement Learning
RNN	Recurrent Neural Network
ROI	Return-on-Investment
ROS	Robot Operating System
SWRL	Semantic Web Rule Language
UCB	Upper Confidence Bound
VDE	Verband der Elektrotechnik Elektronik Informationstechnik
VDI	Verein Deutscher Ingenieure
VDMA	Verband Deutscher Maschinen- und Anlagenbau
VI	Variational Inference
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language (rekursives Akronym)



## Formelzeichenverzeichnis

---

Formelzeichen	Bedeutung
A	Aktionsraum
a	Aktion
$A_{GAE}$	Advantage
d	Distanzmaß
h	Case Embedding
J	Zielfunktion beim maschinellen Lernen
$L_1$	Manhattan-Distanz
o	Wahrnehmung (engl. <i>observation</i> )
P	Transitionswahrscheinlichkeit
p	p-Wert (Signifikanzniveau)
$Q^\pi$	Wert eines Aktion-Zustand-Paars
R	Diskontierte zukünftige Belohnung (engl. <i>return</i> )
r	Belohnungssignal (engl. <i>reward</i> )
s	Zustand
$V^\pi$	Wert eines Zustands
$\alpha$	Lernrate (Hyperparameter in KNN)
$\gamma$	Discount-Faktor
$\epsilon$	Clipping-Parameter (Hyperparameter in PPO)
$\theta$	Lernparameter
$\lambda$	GAE-Lambda (Hyperparameter in PPO)
$\pi$	Strategie
$\rho_0$	Startzustand



# 1 Einleitung

## 1.1 Bedeutung der Automation im 21. Jahrhundert

Der durch die industriellen Revolutionen gewonnene Vorsprung hat es derzeitigen Hochlohnländern in Europa lange Zeit ermöglicht, eine drei Mal höhere Produktivität als der weltweite Durchschnitt zu erzielen (PIKETTY 2018). Das wirtschaftliche Wachstum wird in Zukunft maßgeblich von der technologischen Entwicklung intelligenter Systeme und der Umsetzung daraus resultierender Automationslösungen zur Steigerung der Produktivität abhängig sein (MCKINSEY et al. 2017).

Doch wirtschaftliches Wachstum trägt nicht zwingend zur gesellschaftlichen Verbesserung bei. Insbesondere die bisher höhere Automatisierbarkeit von Tätigkeiten im Niedriglohn-Sektor und deren Einfluss auf die Beschäftigung stellen eine Frage in den Vordergrund: Automation um jeden Preis? (TILLEY 2017) So ist durch Automation stets ein kurzfristiger Einbruch der Beschäftigung entstanden. Doch wie in der Vergangenheit erwarten Experten auch im Rahmen intelligenter Systeme eine langfristige Umstrukturierung der derzeitigen Berufe zu höherqualifizierten und besser bezahlten Berufsprofilen (AGRAWAL et al. 2017, IFR 2018b).

Zudem haben die Erfahrungen hinsichtlich intelligenter Systeme in den vergangenen Jahren aufgezeigt, dass ein großes Potenzial für die Automation einzelner menschlicher Tätigkeiten besteht. Allerdings sind viele Aufgaben, welche dem Menschen einfach erscheinen, weitaus schwerer automatisierbar, als ursprünglich erwartet wurde (MITCHELL 2021).

Für die Integration intelligenter Systeme in bestehende Unternehmensprozesse ist daher eine vorherige Machbarkeitsanalyse notwendig. Möglichkeiten hierfür stellen Werkzeuge zur Vereinfachung der prototypischen Umsetzung und der Einsatz von Simulationen dar. Doch auch in der Realisierung und im Betrieb sind Werkzeuge notwendig, welche eine Interaktion des Menschen mit den intelligenten Systemen ermöglichen (VDMA 2018).

### 1.2 Deep Reinforcement Learning in der Montage

Die zunehmende Variantenvielfalt sowie verkürzte Produktlebenszyklen stellen eine Herausforderung für derzeitige Produktionssysteme dar (REINHART 2017). Der Montage vorgelagerte Fertigungsprozesse ermöglichen durch Technologien wie Computerized Numerical Control (CNC) und additive Fertigungsverfahren eine zunehmende Individualisierung von Produkten sowie eine Produktion bis hin zur Losgröße Eins (TILLEY 2017). Diese Entwicklung erfordert in nachgelagerten Montageprozessen ein höheres Maß an Flexibilität, welches bei der Umsetzung von Automationslösungen berücksichtigt werden muss (SHAH et al. 2008).

Zudem sind die Umsetzung von kundenindividuellen Produkten, die schnelle Reaktionsfähigkeit und der Einsatz einer Vielzahl an spezifischen Technologien maßgebliche Wettbewerbsvorteile kleiner und mittelständischer Unternehmen (KMU) in Europa (PERZYLO et al. 2019). Neben einem hohen Maß an Flexibilität erfordern die oftmals evolutionär gewachsenen Produktionsstrukturen und die damit verbundenen manuellen Teilprozesse eine Anpassungsfähigkeit der Montagesysteme an Varianz (z. B. variable Ablagepositionen durch manuelle Logistikprozesse).

Diese erhöhten Anforderungen an Montageprozesse erschweren eine Integration von Automationslösungen in KMU (PERZYLO et al. 2019). Doch auch die notwendigen finanziellen Mittel und das notwendige Expertenwissen zur Integration und zum Betrieb stellen Gründe dar, warum ein hoher Automationsgrad bisher vor allem Großunternehmen vorbehalten ist (IFR 2018b).

Während die Kosten der Hardware von Automationssystemen sinken (IFR 2018b), ermöglichen neue Ansätze intelligenter Robotersysteme ein steigendes Maß an Flexibilität und Anpassungsfähigkeit (TILLEY 2017). Diese sind zunehmend in der Lage, bestehende manuelle Tätigkeiten durchführen zu können (MCKINSEY et al. 2019). Einen dieser Ansätze stellt Reinforcement Learning dar.

Im Gegensatz zu alternativen Ansätzen, welche unter anderem auf Planungs- oder Optimierungsalgorithmen basieren, wird in Reinforcement Learning eine direkte Zuordnung der Aktionssignale (z. B. Steuerungssignale von Roboterachsen) zum wahrgenommenen Zustand der Umgebung innerhalb einer softwaretechnischen Einheit, dem sogenannten Agenten, abgebildet (LEONETTI et al. 2016).

Hierdurch ist es möglich, ohne zusätzliche Rechenzeiten auf auftretende Veränderungen der Umgebung zu reagieren (H. ZHU et al. 2019b). Neben der Flexibilität kann hierdurch auch die Anpassungsfähigkeit von Montagesystemen erhöht werden.

Die Kombination mit Deep Learning (Deep Reinforcement Learning, DRL) ermöglicht es, auch hochdimensionale sensorische Daten wie Kamerabilder als Zustände der Umgebung in den Ansatz von Reinforcement Learning mit aufzunehmen (SUTTON & BARTO 2018, IBARZ et al. 2021). Somit ist keine aufgabenspezifische Implementierung weiterer Module zur Datenvorverarbeitung notwendig (z.B. zur Ableitung von Positionen durch Bildverarbeitungssysteme). Hierdurch wird eine breite Anwendbarkeit auf DRL basierender Agenten in unterschiedlichen Aufgabenstellungen ermöglicht (SICILIANO & KHATIB 2016).

Durch DRL konnten im Forschungsumfeld bereits Automationslösungen mit einem hohen Maß an Flexibilität und Robustheit im Bereich der Montage, wie Handhabungs- und Fügeprozessen, umgesetzt werden (OPENAI et al. 2019, BREYER et al. 2018, SCHOETTLER et al. 2019, HAARNOJA et al. 2018). Während DRL im Forschungsumfeld stark an Bedeutung gewonnen hat, haben auch bereits hierauf aufbauende Anwendungen Einzug in die Industrie gehalten (WINDER 2020).

### 1.3 Zielsetzung der Arbeit

Während in der Industrie ein hoher Nutzen von intelligenten Systemen erwartet wird, fehlt in den meisten Unternehmen noch die Expertise, um diese Systeme in bestehende Prozesse integrieren zu können (DHAWAN et al. 2018). Insbesondere KMU besitzen nicht die Möglichkeit, entsprechende Experten beschäftigen zu können (VDMA 2018). Ein wesentlicher Befähiger für die Integration von Ansätzen wie DRL in Unternehmen liegt daher in der Vereinfachung der Implementierung hierauf aufbauender Systeme durch entsprechende Werkzeuge (IFR 2020, NG 2021).

Ein im Rahmen bestehender Automationslösungen existierendes Werkzeug zur Vereinfachung der Implementierung stellt die aufgabenorientierte Programmierung dar (BACKHAUS 2016, PERZYLO et al. 2019). In der aufgabenorientierten Programmierung erfolgt eine automatische Generierung von Steuerungs- und Roboterprogrammen basierend auf einer abstrahierten Beschreibung der Aufgabenstellung. Hierdurch wird eine Reduktion der notwendigen Kompetenzen sowie der manuellen Aufwände ermöglicht (BACKHAUS 2016, CSISZAR et al. 2017, JACOBSSON et al. 2016).

# 1 Einleitung

Um die Umsetzung und Nutzung auf DRL basierender Montagesysteme in KMU zu ermöglichen, verfolgt diese Arbeit das Ziel, die bestehenden Ansätze der aufgabenorientierten Programmierung um die Domäne von DRL zu erweitern. Abbildung 1.1 gibt einen Überblick der in dieser Arbeit verfolgten Teilziele.

**TZ** : Teilziel dieser Arbeit

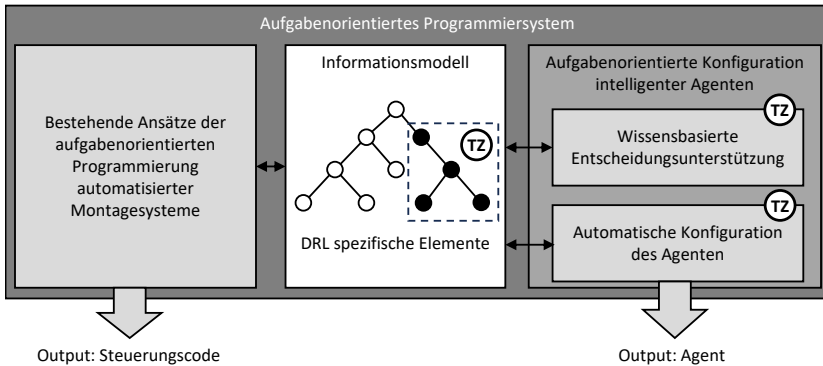


Abbildung 1.1: Teilziele zur Integration von DRL in die aufgabenorientierte Programmierung von Montagesystemen

Diese Ansätze basieren auf Informationsmodellen, welche eine digitale Beschreibung der Aufgabenstellung (sog. Aufgabenmodell) und ein digitales Modell des Montagesystems (sog. Umweltmodell) beinhalten. Daher besteht ein Teilziel dieser Arbeit darin, diese bestehenden *Informationsmodelle der aufgabenorientierten Programmierung um Elemente von DRL zu erweitern*.

Ein wesentlicher Unterschied zu bestehenden Ansätzen der aufgabenorientierten Programmierung ist, dass das Verhalten des Agenten in DRL innerhalb einer Trainingsphase selbstständig gelernt und durch ein künstliches neuronales Netz (KNN) abgebildet wird. Vom Nutzer müssen vor der Trainingsphase die Bewertung des Agentenverhaltens und dessen Interaktion mit der Umgebung definiert werden. Zur Reduktion der hierzu nötigen Kompetenzen ist die Konzeptionierung eines Teilsystems zur *wissensbasierten Entscheidungsunterstützung* notwendig.

Vor der Trainingsphase des Agenten müssen zudem die Architektur des KNN und die Parameter des verwendeten Lernalgorithmus (sog. Hyperparameter) definiert werden. Diese aufgabenspezifischen Hyperparameter erfordern im Bereich des maschinellen Lernens viel Erfahrung und können selbst durch Experten nicht a priori bestimmt werden (ANAND et al. 2020). Während der Trainingsprozess stark von der Wahl dieser Hyperparameter abhängt (MAHMOOD et al. 2018), existiert ein geringes Wissen darüber, welche Auswirkungen einzelne Hyperparameter auf den Lernprozess des Agenten haben (ENGSTROM et al. 2020). Daher wird das derzeitige auf Versuch-und-Irrtum basierende Vorgehen teilweise mit Alchemie verglichen (MITCHELL 2021). In der Praxis wird in Deep Learning auf Hyperparameter-Konfigurationen ähnlicher Fälle zurückgegriffen und in weiteren Iterationen eine Optimierung des Agenten vorgenommen (BENGIO et al. 2009, GRAESSER & KENG 2019).

Diese Konfiguration der Hyperparameter erfordert kein durch den Nutzer vorhandenes Prozesswissen. Sie stellt durch die notwendigen Kompetenzen im Bereich von DRL allerdings eine wesentliche Hürde in der Umsetzung dar. Daher wird in dieser Arbeit ein Teilsystem für die *automatische Konfiguration des Agenten* konzeptioniert, welches diesen Schritt ohne manuelle Anweisungen des Nutzers durchführt. Im Gegensatz zur aufgabenorientierten Programmierung erfolgt daher keine Generierung von Code, sondern die Konfiguration der Hyperparameter des Agenten. Aus diesem Grund wird in dieser Arbeit der Begriff der *aufgabenorientierten Konfiguration* für den Einsatz der beiden Teilsysteme verwendet.

Ein weiterer Unterschied zur aufgabenorientierten Programmierung liegt in der Generierung der Teillösungen auf Basis von Wissenselementen. Während aufgabenorientierte Programmiersysteme sowohl Planungs- und Optimierungsalgorithmen als auch die Wiederverwendung bestehender Teillösungen einsetzen (BACKHAUS 2016), lassen sich die in dieser Arbeit konzeptionierten Teilsysteme dem Bereich der *wissensbasierten Systeme* zuordnen.

### 1.4 Aufbau und wissenschaftliches Vorgehen

Da das Ziel dieser Arbeit die Konzeptionierung eines Systems zur aufgabenorientierten Konfiguration intelligenter Agenten in der Montage darstellt, lässt sich diese Arbeit nach ULRICH & HILL (1976) den angewandten Wissenschaften zuordnen, wobei die Umset-

# 1 Einleitung

zung auf DRL basierender Montagesysteme durch Ansätze der aufgabenorientierten Programmierung industriell anwendbar gemacht werden soll.

KUBICEK (1977) empfiehlt im Rahmen des forschungsmethodischen Vorgehens die Entwicklung eines heuristischen Bezugsrahmens, in welchem eine aktive Auseinandersetzung mit den Problemstellungen in einem definierten Forschungskontext erfolgt. Abbildung 1.2 zeigt die in dieser Arbeit als relevant erachteten Elemente des heuristischen Bezugsrahmens aus den Disziplinen der Produktionstechnik und Informatik. Das bestehende Wissen zu den einzelnen Elementen und deren Beziehungen wird über die referenzierten Abschnitte zu den Grundlagen und dem Stand der Wissenschaft und Technik dieser Arbeit abgebildet.

So kann in diesem Forschungsvorhaben auf fundiertes Wissen aus dem Einsatz der aufgabenorientierten Programmierung von Montagesystemen zurückgegriffen werden. Der wesentliche Erkenntnisgewinn wird daher durch die Auseinandersetzung mit Problemstellungen in der Integration des bestehenden Wissens aus der Anwendung intelligenter Agenten in der Robotik sowie Ansätzen zur automatischen Konfiguration dieser intelligenten Agenten angestrebt.

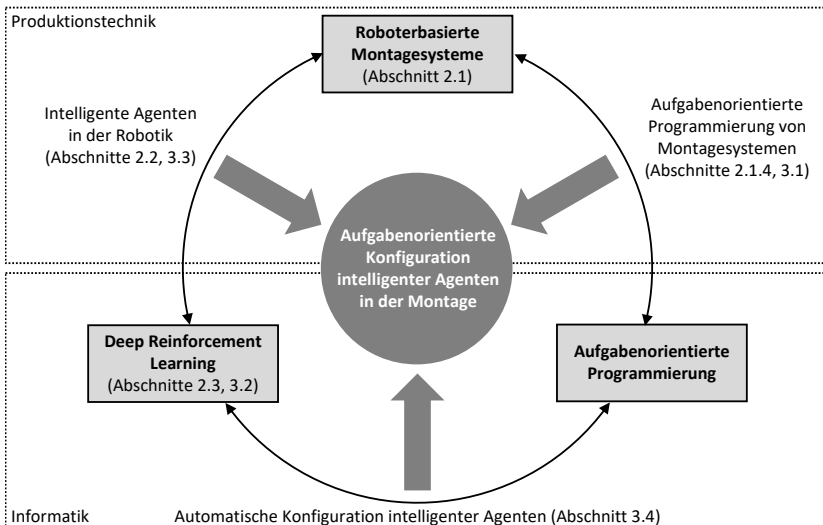


Abbildung 1.2: Heuristischer Bezugsrahmen dieser Arbeit



## 1.4 Aufbau und wissenschaftliches Vorgehen

Zur Erreichung der im vergangenen Abschnitt definierten Teilziele orientiert sich das wissenschaftliche Vorgehen dieser Arbeit an der *Design Research Methodology* (DRM) nach BLESSING & CHAKRABARTI (2009). Die Methode lässt sich unter anderem bei der Konzeptionierung von Softwaresystemen anwenden und besteht aus den Schritten *Research Clarification*, *Descriptive Study I&II* und *Prescriptive Study*. Die Schritte und die korrespondierenden Kapitel sind in Abbildung 1.3 dargestellt.

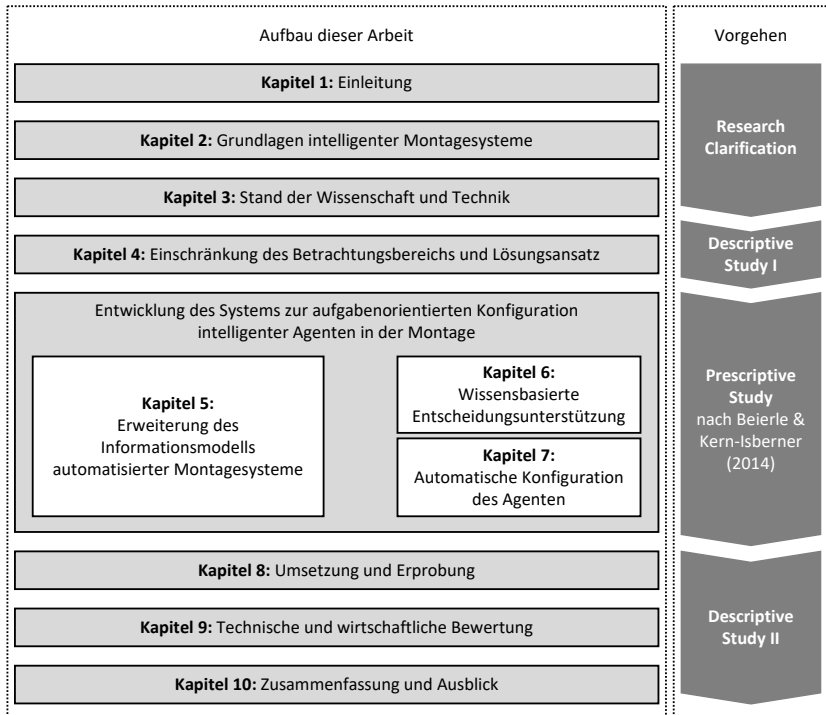


Abbildung 1.3: Aufbau und wissenschaftliches Vorgehen der Arbeit im Rahmen der Design Research Methodology

Nach einem Überblick über die Grundlagen zu Montagesystemen und Deep Reinforcement Learning in Kapitel 2 werden in Kapitel 3 die Ergebnisse der *Research Clarification* aufgezeigt.

## 1 Einleitung

---

Eine Literaturrecherche bestehender Ansätze in der Umsetzung auf Deep Reinforcement Learning basierender Systeme sowie von Ansätzen zur aufgabenorientierten Programmierung und semantischen Beschreibung von Montagesystemen bildet die Grundlage, um Überschneidungen und zu schließende Lücken zwischen den Disziplinen abzuleiten.

Im Schritt der *Descriptive Study I* erfolgt eine Analyse der Rahmenbedingungen, um eine Detaillierung der Lösungselemente der Arbeit zu ermöglichen. Die Ergebnisse dieses Schritts sind in Kapitel 4 zusammengefasst. Neben einer Analyse der Rahmenbedingungen des angestrebten Systems erfolgt hier eine Definition des Betrachtungsbereichs sowie eine Detaillierung der notwendigen Lösungsbausteine.

Das darauffolgende Kapitel befasst sich mit dem Schritt der *Prescriptive Study*, in welchem eine methodische Entwicklung des Systems erfolgt. Nach BLESSING & CHAKRABARTI (2009) können hier Ansätze der Softwareentwicklung angewandt werden. So definieren BEIERLE & KERN-ISBERNER (2014, S. 19) ein iteratives Vorgehen zur Entwicklung wissensbasierter (Software-)Systeme, welches in diesem Schritt der Arbeit adaptiert wird.

Im Anschluss erfolgen nach BEIERLE & KERN-ISBERNER (2014) eine Auswahl relevanter Wissensquellen sowie das Design der Wissensbasis, deren Elemente durch das in Kapitel 5 vorgestellte Informationsmodell abgebildet werden. Darauf folgen eine Konzeptionierung und Implementierung von wissensbasierten Inferenzmechanismen in Kapitel 6 und 7. Da diese Inferenzmechanismen von der Wissensbasis abhängen, wird nach BEIERLE & KERN-ISBERNER (2014) eine iterative Verfeinerung und Generalisierung vorgesehen, bei der das Informationsmodell angepasst wird, um fehlende Elemente zur Umsetzung der Inferenzmechanismen zu ergänzen.

Sowohl im Schritt der *Descriptive Study II* nach BLESSING & CHAKRABARTI (2009) als auch im Vorgehen nach BEIERLE & KERN-ISBERNER (2014) werden letztendlich eine prototypische Umsetzung und die Erprobung des Systems durchgeführt. Kapitel 8 zeigt die Ergebnisse der Anwendung des Systems in einem Referenzszenario aus der Montage.

Hierauf aufbauend erfolgt in Kapitel 9 eine technische und wirtschaftliche Bewertung des Ansatzes dieser Arbeit. Zudem wird in Kapitel 10 eine Zusammenfassung der Ergebnisse und ein Ausblick gegeben.

## 2 Grundlagen intelligenter Montagesysteme

Diese Arbeit verfolgt das Ziel der Integration von Deep Reinforcement Learning in bestehende Ansätze zur Umsetzung roboterbasierter Montagesysteme. Um die Schnittstellen dieser Disziplinen aufzuzeigen, wird daher in Abschnitt 2.1 zunächst ein Überblick zum Aufbau und zur Programmierung roboterbasierter Montagesysteme gegeben. In Abschnitt 2.2 erfolgt eine Einführung in intelligente Robotersysteme, welche einen Ansatz zum Umgang mit Varianz in Montageprozessen darstellen. Anschließend wird in Abschnitt 2.3 der in dieser Arbeit fokussierte Ansatz des Deep Reinforcement Learnings näher erläutert.

### 2.1 Roboterbasierte Montagesysteme

#### 2.1.1 Funktionen und Aufbau von Montagesystemen

Die Montage beschreibt die Gesamtheit aller Vorgänge zum Zusammenbau von Einzelteilen oder Baugruppen zu höherwertigen Baugruppen oder Endprodukten (WARNECKE 1995). Diese Vorgänge lassen sich einteilen in Fügen, Handhaben, Kontrollieren, Justieren und Sonderoperationen (siehe Abbildung 2.1). In der Produktion liegt die Montage zwischen der Fertigung und der Auslieferung des Produkts (FELDMANN 2004).

Eine Montagestation ist die kleinste Einheit eines Montagesystems, welche zur Erfüllung einer einzelnen Montagefunktion dient (CIRP 2020). Wenn zusätzlich zur Mechanisierung auch die Steuerung des Prozesses durch ein technisches System erfolgt, handelt es sich um ein *automatisiertes Montagesystem* (CIRP 2020).

Eine Montagestation lässt sich noch weiter unterteilen in das *Funktionsmodul*, welches die eigentliche Funktionsausübung beinhaltet, und das *Versorgungsmodul*, welches die Teilbereitstellung der angrenzenden Systeme einschließt.

## 2 Grundlagen intelligenter Montagesysteme

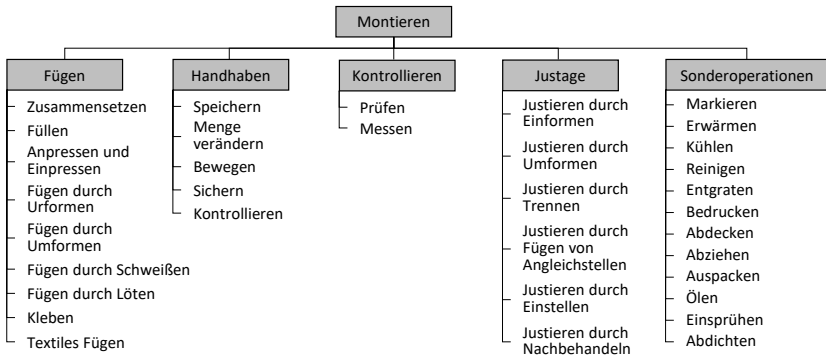


Abbildung 2.1: Funktionen der Montage nach LOTTER & WIENDAHL (2012)

Die in Abbildung 2.2 aufgeführten Arten der Operatoren in einer Montagestation lassen sich nach FELDMANN (2004) einteilen in

- Basisoperatoren: Grundlage der Montagetätigkeit ohne aktive Funktion (z. B. Arbeitstisch, Maschinengestell)
- Positionsoperatoren: Fixierung von Objekten in einer definierten Lage (z. B. Werkstückträger, Formnest)
- Funktionsoperatoren: Durchführung der eigentlichen Montagefunktion (z. B. Schweißgerät, Lötanlage, Sensoren bei Kontrollfunktionen)
- Handhabungsoperatoren: Bewegen von Positionsoperatoren oder Funktionsoperatoren (z. B. Industrieroboter, Förderband).

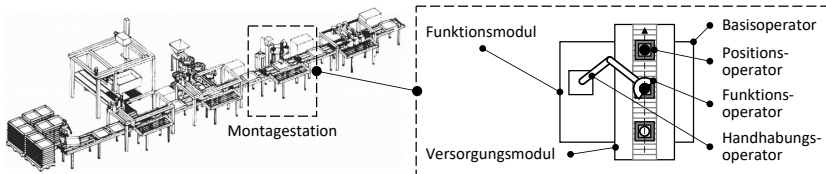


Abbildung 2.2: Aufbau einer Montagestation innerhalb einer Montagelinie nach FELDMANN (2004) und CIRP (2020)

### 2.1.2 Ausprägungen roboterbasierter Montagesysteme

In automatisierten Montagesystemen werden als universell einsetzbare Handhabungsoperatoren vorwiegend die in Abbildung 2.3 dargestellten Industrieroboter in Form von SCARA- und Knickarmrobotern eingesetzt (CIRP 2020). Weiterhin zählen Portalroboter zu dieser Klasse der seriellen Roboter, welche 95 % der Industrieroboter ausmachen. Parallele Roboter lassen sich in Nischenanwendungen wiederfinden, in welchen durch parallel angeordnete Antriebe höhere Steifigkeiten und Bewegungsgeschwindigkeiten ermöglicht werden (POTT & DIETZ 2019).

Neben industriell eingesetzten Industrierobotern existieren weitere Roboterkinematiken, welche eine Anwendung in veränderlichen Umgebungen ermöglichen (SICILIANO & KHATIB 2016). Während diese Roboterkinematiken in Forschungsprototypen bereits physisch umgesetzt werden, stellt die Komplexität der softwaretechnischen Konfiguration dieser Systeme eine wesentliche Hürde für ihren Einsatz in der Industrie dar (NEE 2015, SICILIANO & KHATIB 2016).

Ein Beispiel hierfür stellen redundante Roboter dar, deren hohe Anzahl an Achsen eine Erhöhung der Flexibilität ermöglicht. Durch die redundanten Posen zur Erreichung einzelner Zielkoordinaten ergibt sich allerdings keine eindeutige Lösung der inversen Kinematik, wodurch die Steuerung erschwert wird.

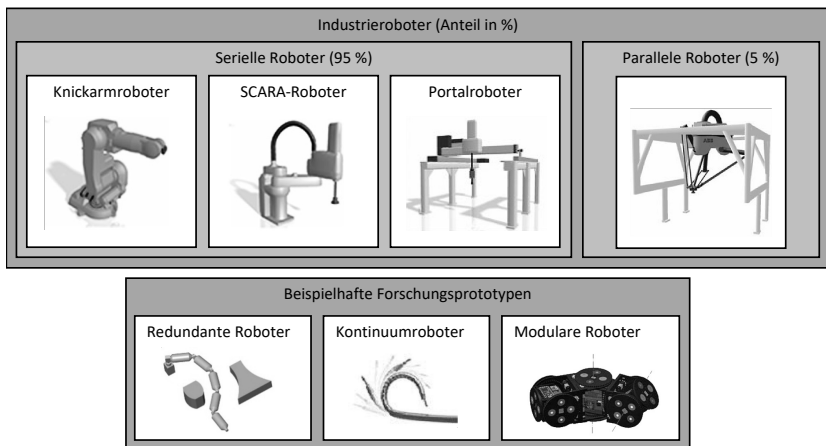


Abbildung 2.3: Überblick verschiedener Roboterkinematiken nach NEE (2015), SICILIANO & KHATIB (2016) und POTT & DIETZ (2019)

## 2 Grundlagen intelligenter Montagesysteme

---

Neben der Bewegungseinrichtung des Roboters werden in der Montage zur Durchführung der Montageaufgabe Funktionsoperatoren eingesetzt, welche sich abhängig von der auszuführenden Montagefunktion in Greifer, Werkzeuge und Mess-/Prüfmittel einteilen lassen. Gegebenenfalls werden weitere Elemente wie Zubehör am Endeffektor oder Peripherie zur Ausübung der Montagefunktion benötigt (POTT & DIETZ 2019). In Tabelle 2.1 werden die zusätzlichen Elemente eines Robotersystems aufgeführt.

Tabelle 2.1: Funktionsoperatoren und weitere Elemente eines Robotersystems nach NEE (2015), POTT & DIETZ (2019) und HAMMERSTINGL (2020)

Element	Ausführung	Beispiele
Funktionsoperator	Greifer	Klemm-, Saug-, Magnet-, Ultraschall-, Bernoulli-, Nadelgreifer
	Werkzeug Mess-/Prüfmittel	Schweißbrenner, Klebepistole Kamerasystem (z. B. zur Qualitätskontrolle)
Zubehör		Wechselsystem, Auslenkungssystem bei Überlast, nachgiebige Elemente, Kraftmomentsensor, Versorgungsleitung für Druckluft
Peripherie		Wechselstation, Kabelschutz

### 2.1.3 Sensorik

Sensoren beschreiben die Gesamtheit aller Funktionseinheiten, welche physikalische Größen aus einem Prozess aufnehmen und der Informationsverarbeitung zugänglich machen (MCCARTHY 2007, CIRP 2020). Sie stellen notwendige Elemente für die Anpassung an Veränderungen der Umgebung dar.

Sensoren ermöglichen die Wahrnehmung des Zustands des Roboters selbst (sog. interne Sensoren) oder die Wahrnehmung der Umgebung des Roboters (sog. externe Sensoren) (HAUN 2013). In Abbildung 2.4 wird ein Überblick über Ausprägungen von Sensoren in der Robotik gegeben.

Sowohl interne als auch externe Sensoren bieten die Möglichkeit, den Zustand der Umgebung den softwaretechnischen Einheiten zur Verfügung zu stellen. Zur Anpassung des Verhaltens an Veränderungen der Umgebung sind daher beide Sensor-Klassen notwendig.

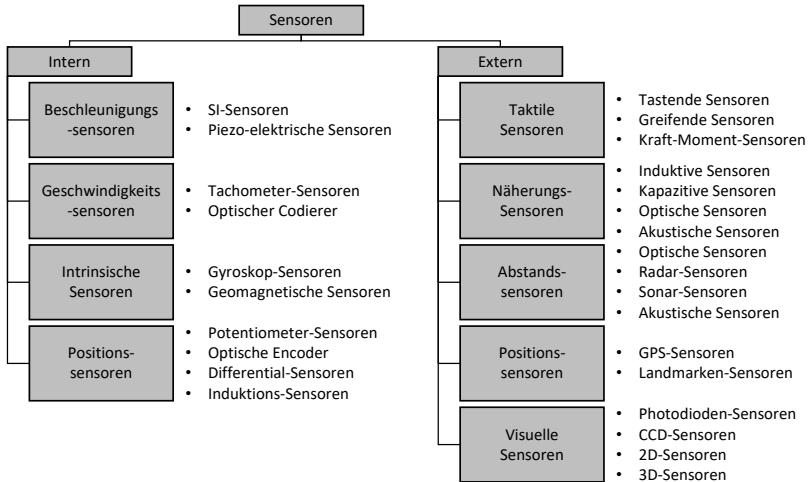


Abbildung 2.4: Sensoren in Robotersystemen nach HAUN (2013)

### 2.1.4 Programmierung roboterbasierter Montagesysteme

Die Programmierung der Steuerungslogik kann auf unterschiedlichen Abstraktionsebenen der Aufgabenbeschreibung erfolgen (HAUN 2013). Die niedrigste Abstraktionsstufe bildet die Vorgabe von Stellgrößen wie Achswinkeln, welche über ein Regelungssystem an der entsprechenden Aktorik (z. B. Elektromotor einer Roboterachse) eingestellt werden (SICILIANO & KHATIB 2016). Durch den Nutzer wird die Programmierung unter anderem durch manuelles Anfahren und Speichern einzelner Bahnpunkte (sog. Teach-In) vorgenommen (HAUN 2013).

Eine höhere Abstraktionsstufe bilden Roboterprogrammiersprachen, welche die Vorgabe des Bewegungsablaufs des Endeffektors in einem kartesischen Koordinatensystem ermöglichen (HAUN 2013). Durch Algorithmen für die Interpolation von Bahnpunkten (z. B. lineare Bewegungen, LIN, oder Point-to-Point-Bewegungen, PTP) und die Rückführung der Zielkoordinaten des Endeffektors in den Gelenkwinkelraum des Roboters (sog. Inverse Kinematik) lässt sich die Sequenz der Steuerungswerte einer Ressource ableiten (SICILIANO & KHATIB 2016).

## 2 Grundlagen intelligenter Montagesysteme

Die aufgabenorientierte Programmierung ermöglicht die höchsten Abstraktionsstufen der Aufgabenbeschreibung, in welchen die Aufgabe durch Zielzustände von Produkten und Ressourcen vorgegeben wird und auf Basis eines digitalen Modells des Montagesystems (Umweltmodell) eine automatisierte Generierung des Roboter- bzw. Steuerungs codes in den jeweiligen Programmiersprachen der Ressourcen erfolgt (WECK & BRECHER 2006, HAUN 2013). Die höchste Abstraktionsebene der Aufgabenbeschreibung stellt das Produkt selbst dar, aus welcher sich die Abfolge und Ausprägung der Montageprozesse ableiten lassen (SICILIANO & KHATIB 2016, BACKHAUS 2016). Die erforderliche Funktionalität des Programmiersystems und somit die Anzahl der notwendigen Softwaremodule nimmt mit steigender Abstraktion der Aufgabenbeschreibung zu (siehe Abbildung 2.5).

Der Output der einzelnen Softwaremodule bildet den Input für weitere Softwaremodule innerhalb der Aufgabenebene oder für die Softwaremodule der nächsten Aufgabenebene. Ein auf Deep Reinforcement Learning basierender Agent lässt sich im Aufgabenmodell der Ebene der Programme und Steuerung zuordnen, da die Aktionssignale eines Agenten sowohl Steuerungsgrößen als auch Regelgrößen der Hardware darstellen können.

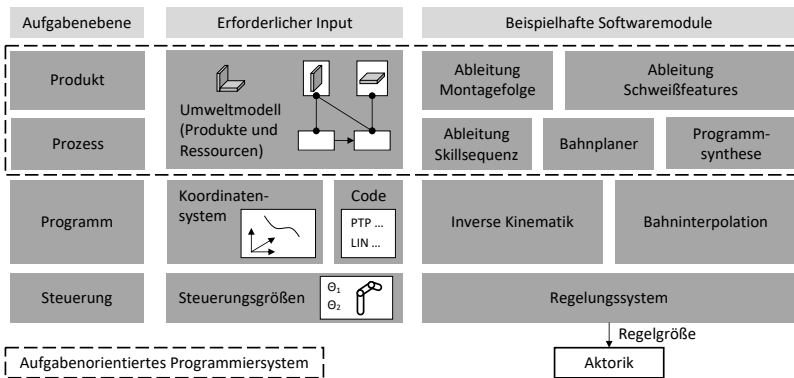


Abbildung 2.5: Einordnung der aufgabenorientierten Programmierung nach WECK & BRECHER (2006), HAUN (2013), BACKHAUS (2016), KUSS et al. (2016) und SICILIANO & KHATIB (2016)



### 2.1.5 Begriffe im Kontext von Varianz

Wie einleitend erläutert, stellt der Umgang mit Varianz eine maßgebliche Herausforderung in der Automatisierung in produzierenden Unternehmen dar. Die Eigenschaft, die hierzu notwendigen Anpassungen von Hardware und Software innerhalb der Produktion wirtschaftlich durchführen zu können, wird als *Veränderungsfähigkeit* bezeichnet (WIENDAHL et al. 2007). Bezogen auf Montagesysteme werden zudem die Begriffe Rekonfigurierbarkeit und Flexibilität verwendet (MICHNIEWICZ 2018).

*Rekonfigurierbarkeit* beschreibt die Eigenschaft zur Anpassung eines Systems an Veränderungen durch Ersetzen, Hinzufügen oder Entfernen von Modulen hinsichtlich ihrer Struktur, Technologie, Kapazität oder Funktionalität (LORENZER 2011, HEES 2017). Die *Rekonfiguration* beschreibt den Vorgang zur Änderung einer Konfiguration bzw. wird durch die Durchführung einer *Konfiguration* eine Konfiguration initial realisiert. ELMARAGHY (2005) unterscheidet zudem zwischen einer weichen bzw. softwaretechnischen und einer harten bzw. physischen Rekonfiguration. Durch die Rekonfiguration wird somit eine Änderung des Systems ermöglicht, welche zur Gewährleistung von Flexibilität notwendig ist (SCHMIDT 1992, GEHLHOFF et al. 2019).

Die *Unterteilung der Flexibilität von Produktionssystemen* erfolgt nach der Ausprägung der Veränderung (WIENDAHL et al. 2014). Bezogen auf Veränderungen des Produkts (Produktflexibilität) findet eine weitere Unterteilung der Flexibilität in Abhängigkeit variierender Merkmale wie Ausgangsmaterial, Format, Arbeitsfolge oder Produktionsvolumen statt (ELMARAGHY 2009, WIENDAHL et al. 2014, GÖTZ 2017, CIRP 2020).

Weiterhin kann auf Begriffsdefinitionen aus dem Bereich des Systems Engineerings zurückgegriffen werden, dessen Systembetrachtung Montagesysteme mit einschließt (HABERFELLNER et al. 2019). Hier erfolgt eine Abgrenzung von Flexibilität zu den Begriffen *Anpassungsfähigkeit* und *Robustheit* (RYAN et al. 2013).

Die *Flexibilität von Systemen* stellt sowohl in der Produktion als auch im Systems Engineering den am häufigsten diskutierten Begriff dar (RYAN et al. 2013, WIENDAHL et al. 2014). RYAN et al. (2013) leiten auf Basis einer Literaturrecherche im Bereich des Systems Engineerings folgende konsensfähige Definition ab: "the measure of how easily a system's capabilities can be modified in response to external change" (RYAN et al. 2013). Bezogen auf Produktionssysteme wird der Aufwand zur Änderung eines

## 2 Grundlagen intelligenter Montagesysteme

---

Systems durch die Zeit und die Kosten bei Durchführung der Änderung bestimmt (ZÄH et al. 2005).

Sowohl im Systems Engineering als auch in der Produktionstechnik wird bei einem flexiblen System davon ausgegangen, dass die Änderung des betrachteten Systems von einer externen Quelle durchgeführt wird (RYAN et al. 2013, VOGEL-HEUSER et al. 2014). In beiden Disziplinen schließt Flexibilität zudem eine Änderung des Systems selbst mit ein (ROSS et al. 2008, SHAH et al. 2008, KALUZA et al. 2005, WESTKÄMPER et al. 2000). Im Systems Engineering wird zudem zwischen Flexibilität innerhalb der Planungs- bzw. Entwicklungsphase und innerhalb der Betriebsphase unterschieden (RYAN et al. 2013).

Im Kontext autonomer Produktionssysteme wird der Begriff der *Anpassungsfähigkeit* verwendet (SCHOLZ-REITER & FREITAG 2007, BRECHER & ÖZDEMIR 2017), aber keine Abgrenzung zur Flexibilität vorgenommen. Nach RYAN et al. (2013) herrscht in Systems Engineering jedoch ein Konsens darüber, dass Anpassungsfähigkeit im Gegensatz zur Flexibilität keine Änderung des Systems durch eine externe Quelle benötigt, sondern sich das System selbst an Veränderungen anpasst.

*Robustheit* findet als Begriff in der Produktionstechnik unter anderem in dessen Schnittmenge zur Regelungstechnik Anwendung. Hier werden durch eine robuste Regelung die Stabilität und die Regelgüte bei Veränderungen in der Umgebung in zuvor definierten Bereichen gewährleistet (UNBEHAUEN & LEY 2014). Der Begriff der Robustheit wird auch bezogen auf Systeme konsistent verwendet und lässt sich dadurch definieren, dass das betrachtete System seine Funktionalität bei Veränderung der Umgebung auch ohne Änderung des Systems selbst aufrechterhält (RYAN et al. 2013).

## 2.2 Intelligente Robotersysteme

### 2.2.1 Intelligente Agenten

Die Vielzahl unterschiedlicher Perspektiven und Meinungen erschweren eine präzise Definition des Begriffs der *künstlichen Intelligenz* (KI) (DIN 2020). Nach ISO/CD 22989 kann sich der Begriff auf die Fähigkeit eines Systems oder auf eine technische Disziplin, welche sich mit der Entwicklung und Erforschung dieser Systeme beschäftigt, beziehen. PORETSCHKIN (2019) unterscheidet weiterhin zwischen KI-Anwendungen

und KI-Komponenten, welche zusammen mit weiteren Softwaremodulen (Vor- und Nachbearbeitungsprozeduren) in eine KI-Anwendung eingebettet sind.

In der Literatur verwendete Definitionen von künstlicher Intelligenz differenzieren sich dahingehend, ob die Leistungsfähigkeit eines Systems in der Durchführung von Aufgaben betrachtet oder ein direkter Bezug zum menschlichen Denkvermögen vorgenommen wird (RUSSELL & NORVIG 2010). In letzterem Kontext wird unterschieden zwischen einer *schwachen künstlichen Intelligenz*, welche nicht mit menschlicher Intelligenz vergleichbar ist und nur spezifische Aufgaben durchführen kann, und einer *starken künstlichen Intelligenz*, welche das Niveau der menschlichen Intelligenz besitzt (FJELLAND 2020, KRAMER 2009, IEC 2021, DIN 2020). Die Expertengruppe für künstliche Intelligenz der europäischen Kommission schlägt bezogen auf die Leistungsfähigkeit eines Systems folgende Definition vor:

"Künstliche Intelligenz beschreibt Systeme, die intelligentes Verhalten dadurch zeigen, dass sie – mit einem gewissen Grad an Autonomie – ihre Umgebung analysieren und entsprechend agieren, um spezifische Ziele zu erreichen." (EUROPÄISCHE KOMMISSION 2019, DIN 2020)

Zur weiteren Abgrenzung verweist die EUROPÄISCHE KOMMISSION (2021) zudem auf drei grundsätzliche Ansätze, auf welchen intelligente Systeme basieren:

1. Maschinelles Lernen (überwachtes Lernen, unüberwachtes Lernen, Reinforcement Learning),
2. Logik- und wissensbasierte Ansätze (u. a. Expertensysteme, Inferenzmaschinen, logisches Schließen, symbolisches Schlussfolgern),
3. Statistische Ansätze (u. a. Such- und Optimierungsmethoden).

Vergleichbar mit der physischen Fitness von Athleten, lässt sich *Intelligenz* nicht auf einen einzelnen messbaren Faktor reduzieren, sondern muss im Kontext der Erfüllung von Aufgaben betrachtet werden (CHOLLET 2019). LEGG & M. HUTTER (2007) analysieren in ihrer Studie 70 Quellen zur Definition von Intelligenz und leiten folgende konsensfähige Definition ab:

"[I]ntelligence measures an agent's ability to achieve goals in a wide range of environments" (LEGG & M. HUTTER 2007)

## 2 Grundlagen intelligenter Montagesysteme

---

Diese nach der sogenannten Legg-Hutter-Intelligenz gemessene Fähigkeit zur Zielerreichung unter veränderlichen Umgebungsbedingungen korreliert mit den Fähigkeiten, welche im Bereich der Automatisierungstechnik relevant sind (EVERITT et al. 2018).

Ein *Agent* wird in diesem Kontext als ein System definiert, welches basierend auf seiner derzeitigen Beobachtung oder basierend auf seinen bisherigen Erfahrungen (bestehend aus bisherigen Beobachtungen) eine Aktion auswählt und auf diese Weise mit der Umgebung interagiert (RUSSELL 2016, SILVER et al. 2021). Im Kontext der Automatisierungstechnik wird ein Agent nach VDI-Richtlinie 2653 zudem definiert als "(...) eine abgrenzbare (Hardware- und/oder Software-) Einheit mit definierten Zielen. Ein Agent ist bestrebt, diese Ziele durch selbständiges Verhalten zu erreichen und interagiert dabei mit seiner Umgebung und anderen Agenten."

Der Begriff des *intelligenten Agenten* stellt somit ein System dar, welches seine Ziele auch unter variierenden Umgebungsbedingungen selbstständig erreicht. In dieser Arbeit wird ein intelligenter Agent zudem als eine Software-Einheit definiert, welche nur über Datenaustausch mit dem physischen System interagiert.

Der im Kontext der Automatisierungstechnik ebenfalls auftretende Begriff der *Autonomie* entzieht sich analog zu künstlicher Intelligenz einer objektiven Definition (DIN 2020). Nach SCHOLZ-REITER & FREITAG (2007) bedeutet Autonomie die Unabhängigkeit der Entscheidungsfindung eines Systems hinsichtlich externer Instruktionen.

Dies ermöglicht es dem System seine Aufgaben über einen längeren Zeitraum ohne manuelles Eingreifen durchführen zu können (DEWEY 2014). Da Automaten wie eine speicherprogrammierbare Steuerung (SPS) ebenfalls eine selbstständige Entscheidungsfindung auf Basis von einer vorgegebenen Logik verfolgen (CIRP 2020), wird in der Literatur eine klare Abgrenzung von Automation und Autonomie bemängelt (IFR 2018a, MÜLLER et al. 2021).

Nach RUSSELL & NORVIG (2010) stellt die Unabhängigkeit eines autonomen Systems vom durch den Entwickler implementierten Vorwissen und die damit verbundene Notwendigkeit zum Lernvermögen in Anbetracht partiellen oder inkorrekten Vorwissens ein wesentliches Merkmal dar. Bei autonomen Systemen ist daher die Anpassungsfähigkeit des Systems und die damit verbundene Fähigkeit, auf Veränderungen reagieren zu können, ein zentrales Element (MÜLLER et al. 2021). Diese Anpassungsfähigkeit kann durch Ansätze des maschinellen Lernens oder auch andere Ansätze umgesetzt werden (JESCHKE et al. 2017, MÜLLER et al. 2021).

### 2.2.2 Maschinelles Lernen in der Robotik

Um auf Veränderungen in der Umgebung reagieren zu können, benötigt die Software eines intelligenten Robotersystems drei wesentliche Fähigkeiten: Wahrnehmen, Planen und Ausführen (siehe Abbildung 2.6). Zur Wahrnehmung des Zustands der Umgebung müssen Sensorsignale zunächst vorverarbeitet werden. Meta-Sensoren nehmen rohe Sensorsignale der Hardware als Input und leiten höherwertige Informationen wie Positionen von Objekten in Bilddaten ab (OTTE 2009). Eine geeignete Repräsentation der Wahrnehmung beinhaltet die Aggregation unterschiedlicher und vergangener Sensorsignale.

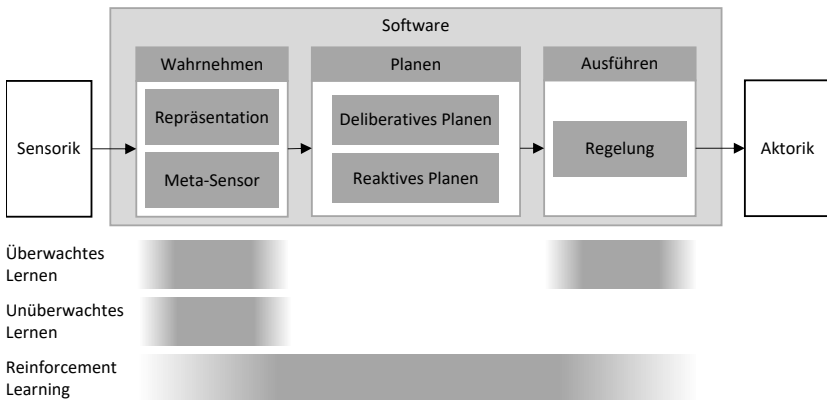


Abbildung 2.6: Typische Anwendungsbereiche maschineller Lernverfahren in der Umsetzung intelligenter Robotersysteme nach BRUNTON & KUTZ (2019), WINDER (2020), HAUN (2013), OTTE (2009) und KRAMER (2009)

Subsysteme zum Planen entscheiden darüber, welche Aktionen auf Basis der wahrgenommenen Zustandsinformationen der Umgebung abgeleitet werden. Zwei wesentliche Ausprägungen dieser Systeme stellen das deliberative und das reaktive Planen dar (SICILIANO & KHATIB 2016). Deliberative Planungssysteme leiten die zur Zielerreichung notwendigen Aktionen basierend auf einem Modell der Umgebung ab, welches durch Informationen der Wahrnehmung aktuell gehalten wird und eine Vorhersage der Auswirkungen von Aktionen ermöglicht (SICILIANO & KHATIB 2016). Diese Pläne können kollisionsfreie Trajektorien basierend auf Optimierungsalgorithmen oder eine Abfolge einzelner Prozesse basierend auf Ansätzen wie dem symbolischen Schließen sein (OTTE 2009, SICILIANO & KHATIB 2016).

## 2 Grundlagen intelligenter Montagesysteme

---

Diese modellbasierten Ansätze erfordern daher eine genaue Modellierung des dynamischen Verhaltens (OTTE 2009). Da die Sensorik teilweise nicht alle Informationen zur Umgebung auf einmal aufnehmen kann und insbesondere in dynamischen Umgebungen eine mehrmalige Umplanung erfolgen muss, sind diese zeitaufwändigen Planungsansätze oftmals nicht umsetzbar (OTTE 2009, SICILIANO & KHATIB 2016). Dies schränkt die Anwendbarkeit zur Umsetzung intelligenter Agenten ein.

Bei reaktiven Planungen wird dagegen kein Modell der Umwelt benötigt, sondern basierend auf vordefinierten Regeln mit einer geringen Rechenzeit auf Veränderungen der Umgebung reagiert (SICILIANO & KHATIB 2016). Dieser Ansatz ermöglicht daher den effizienten Einsatz in dynamischen Umgebungen (u. a. Sicherheitssysteme), lässt allerdings keine selbstständige Optimierung des Verhaltens zu (HAUN 2013).

*Maschinelles Lernen* kann zur Realisierung der einzelnen Softwaremodule eines intelligenten Robotersystems eingesetzt werden. Beim maschinellen Lernen wird Wissen aus Erfahrungen generiert, indem Lernalgorithmen basierend auf Beispielen eine Modellierung der Zusammenhänge entwickeln, welche anschließend auf neue Daten angewandt werden kann (DÖBEL et al. 2018). Diese Lernfähigkeit ermöglicht hierauf aufbauenden Systemen zukünftige Aufgaben der selben Art effizienter oder effektiver auszuführen und sich während der Trainingsphase in der Entwicklungsphase oder während des Betriebs selbstständig anzupassen (VDI/VDE 3550). Maschinelle Lernverfahren finden daher in Situationen Einsatz, in welchen die manuelle Implementierung des Systems zu aufwändig oder unmöglich ist und veränderliche Umgebungen eine Neuanpassung des Systems erfordern (GÉRON 2018).

Maschinelle Lernverfahren lassen sich in drei grundlegende Ansätze einteilen: überwachtes Lernen, unüberwachtes Lernen und Reinforcement Learning (GÉRON 2018). In der Industrie kommt derzeit vornehmlich überwachtes Lernen zum Einsatz (MIN et al. 2017, NG 2018), wobei dem lernenden System zu jedem Datenpunkt die entsprechenden Antworten mitgegeben werden und durch die erlernten Zusammenhänge eine Vorhersage der Antworten bei neuen Eingangsdaten möglich ist (RASCHKA & MIRJALILI 2018). In der Robotik wird dieser Ansatz typischerweise im Bereich der Wahrnehmung zur Ableitung planungsrelevanter Informationen (wie Positionen von Objekten aus Bilddaten) eingesetzt (OTTE 2009). Auch im Bereich modellbasierter Regelungssysteme wird überwachtes Lernen zur Systemidentifikation bei nichtlinearen Systemen angewendet, wodurch nichtlineare Systeme wie beispielsweise Schweißprozesse geregelt werden können (BRUNTON & KUTZ 2019, BACHMANN et al. 2018).

Beim unüberwachten Lernen werden dagegen keine Antworten mitgegeben, sondern die Struktur der Daten selbst erkundet (RASCHKA & MIRJALILI 2018). Hierdurch können u. a. bei hochdimensionalen Daten wie Bilddaten vereinfachte Repräsentationen abgeleitet werden (HAUN 2013, BRUNTON & KUTZ 2019).

In Reinforcement Learning (dt. bestärkendes Lernen) lernt das System, durch Interaktion mit einer Umgebung und der Bewertung der ausgeführten Aktionen anhand eines Feedbacksignals (sog. Belohnungssignal), seine Leistung zu verbessern (RASCHKA & MIRJALILI 2018). Gelernt werden hier der Zusammenhang zwischen den wahrgenommenen Zuständen und den für die langfristige Zielerreichung optimalen Aktionen zu jedem Zeitpunkt.

Hierdurch sind während des Betriebs keine zeitaufwändigen Rechenprozesse wie Such- oder Optimierungsalgorithmen bei einer Veränderung der Umgebung notwendig (Geffner 2018). Während sich frühere Ansätze zum Einsatz von Reinforcement Learning in der Robotik auf die Umsetzung des Planungssystems konzentriert haben (OTTE 2009, HAUN 2013), hat die Kombination mit Deep Learning (sog. Deep Reinforcement Learning) in den vergangenen Jahren enorme Fortschritte hinsichtlich des Ende-zu-Ende-Lernens komplexer Aufgaben im Bereich der Robotik ermöglicht (LILLICRAP et al. 2016, OPENAI et al. 2019, OBANDO-CERÓN & CASTRO 2021).

In Deep Reinforcement Learning werden unvorverarbeitete Sensorsignale wie Bilddaten in den Lernprozess mit eingeschlossen, wodurch sich der gesamte Datenverarbeitungsprozess von der sensorischen Wahrnehmung bis hin zu den Stellgrößen der Aktorik ganzheitlich anpassen und optimieren lässt (SINGH et al. 2019). In den vergangenen Jahren wurden erste auf Deep Reinforcement Learning basierende Roboteranwendungen im Bereich der Handhabung durch Start-ups wie OSARO und Covariant im industriellen Kontext umgesetzt (SATARIANO & METZ 2020, HAO 2021, RÖHLER et al. 2021).

### 2.3 Deep Reinforcement Learning

#### 2.3.1 Reinforcement Learning

Eine wesentliche Aufgabenstellung intelligenter Systeme besteht im Durchführen sequentieller Entscheidungen in stochastischen Umgebungen, welche formal durch das Markov-Entscheidungsproblem (engl. Markov decision process, MDP) beschrieben werden (GUO 2017). Ein MDP ist grundlegend definiert durch

- $s$ : Zustände der Umgebung (engl. *states*)
- $a$ : Ausführbare Aktionen (engl. *actions*)
- $P(s'|s, a)$ : Transitionswahrscheinlichkeit bei der Ausführung von  $a$  in den Folgezustand  $s'$  zu gelangen und
- $r(s, a)$ : Belohnungssignal (engl. *reward*) (SUTTON & BARTO 2018).

Das dynamische Verhalten der Umgebung wird formal durch  $P(s'|s, a)$  beschrieben und ist dem Agenten, welcher die Entscheidungen über die auszuführende Aktion  $a$  trifft, nicht bekannt. Während des Lernprozesses wird  $P(s'|s, a)$  durch eine Simulation oder ein reales System abgebildet. Der Agent erhält als einzige Information den nächsten Zustand  $s'$  der Umgebung und ein skalares Belohnungssignal  $r$  (siehe Abbildung 2.7a). Das Belohnungssignal wird durch eine Belohnungsfunktion  $r(s, a)$  definiert, welche den Zustand der Umgebung und die ausgeführte Aktion zu jedem Zeitpunkt bewertet. Durch  $r(s, a)$  wird somit die Aufgabenstellung des Agenten beschrieben, wobei gewünschte Zustände durch eine Erhöhung des Belohnungssignals und ungewünschte Zustände durch eine Verringerung des Belohnungssignals bewertet werden. Eine Herausforderung in Reinforcement Learning besteht darin, dass die ausgewählte Aktion nicht nur einen Einfluss auf das direkt folgende Belohnungssignal besitzt, sondern auch die Erreichung späterer Zustände und Belohnungssignale beeinflusst. Das Ziel bei der Auswahl einer Aktion besteht daher darin, die Summe aller folgenden Belohnungssignale zu maximieren. Dies wird durch die *diskontierte zukünftige Belohnung*  $R$  bei einem Zeitpunkt  $t$  im Lernprozess berücksichtigt. Diese ist definiert als

$$R_t := r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^T r_T = r_{t+1} + \gamma R_{t+1}. \quad (2.1)$$



$T$  gibt den Zeitpunkt des terminalen Zustands bei episodischen Aufgabenstellungen an und ist bei kontinuierlichen Aufgabenstellungen unendlich groß. In diesem Fall ist die Summe aller zukünftigen Belohnungssignale unendlich groß, weshalb der Wert von  $R$  über den Discount-Faktor  $\gamma$  ( $0 \leq \gamma \leq 1$ ) begrenzt werden kann (SUTTON & BARTO 2018). Episodische Aufgabenstellungen besitzen dagegen eine vordefinierte maximale Anzahl an Zeitschritten und gegebenenfalls Abbruchkriterien (z. B. bei Erreichung eines Zielzustands), wodurch  $T$  begrenzt wird. Wie das Beispiel in Abbildung 2.7b zeigt, kann  $\gamma$  auch bei einer Einschränkung von  $T$  zur Definition der Aufgabe genutzt werden. Im gezeigten Beispiel wird durch eine geringere Anzahl an Zeitschritten ein höherer Wert von  $R$  erreicht.

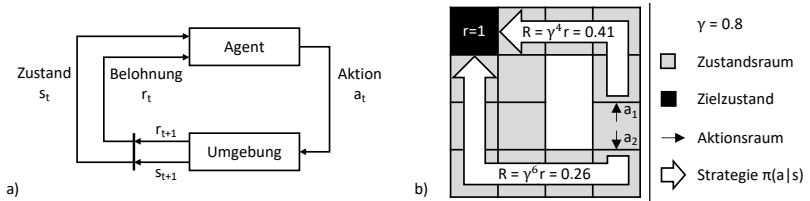


Abbildung 2.7: a) Agent-Umgebung-Interaktion nach SUTTON & BARTO (2018)  
 b) Beispielhafte Bewertung von zwei Strategien in der Trainingsphase, nach der eine kürzere Schrittzahl zu einem höheren Wert von  $R$  führt.

Die Zielstellung eines Agenten besteht innerhalb der Trainingsphase darin, eine Strategie  $\pi(a|s)$  zu erlernen, welche zu jedem wahrgenommenen Zustand eine Aktion auswählt, sodass  $R$  maximiert wird. Der Agent erhält keine Information darüber, welche Aktion durchgeführt werden soll, sondern lernt durch Versuch und Irrtum, welche Aktionen zu den jeweiligen Zuständen zu einer Maximierung von  $R$  führen. Während des Lernprozesses muss daher stets ein Kompromiss aus Exploration der Umgebung durch zufällige Aktionen und die Nutzung des Wissens aus bisherigen Erfahrungen erfolgen, um ein optimales Verhalten effizient zu erlernen.

Reinforcement Learning basiert auf der Annahme, dass sich jede Aufgabenstellung durch ein skalares Belohnungssignal ausdrücken lässt (sog. Reward Hypothesis) (SUTTON & BARTO 2018). In diversen Domänen, wie beispielsweise in der Definition von Montageaufgaben, lässt sich die Aufgabe am natürlichsten über die Definition eines positiven Belohnungssignals bei der Erreichung eines Zielzustands abbilden (BELOUSOV et al. 2021).

## 2 Grundlagen intelligenter Montagesysteme

Bis ein solcher Zustand erreicht wird, erhält ein Agent noch kein Feedback, weshalb dieser bei komplexeren Aufgabenstellungen eine oftmals untragbare Anzahl an Trainingsschritten benötigen würde, um das gewünschte Verhalten zu lernen (BELOUSOV et al. 2021). Daher wird das lernende System über zusätzliche Belohnungssignale im Lernprozess unterstützt (sog. Reward Shaping), indem Vorwissen zur Aufgabenausführung (z. B. Belohnung der Näherung an eine Zielposition) in die Belohnungsfunktion integriert wird (LAUD 2004, DEVLIN & KUDENKO 2016). Die geeignete Definition dieser Belohnungsfunktion ist daher nicht trivial und kann bei einer falschen Ausprägung zu einem ungewollten Verhalten des Agenten führen (LINDNER et al. 2021, WIEWIORA 2017).

Im Falle einfacher Aufgabenstellungen lässt sich  $\pi(a|s)$  auf Basis einer einfachen Modellierung der Zusammenhänge wie Tabellen und lineare Modelle abbilden (GATTI 2015). In vielen Anwendungen, in welchen Reinforcement Learning zum Einsatz kommen soll, sind die Zustände der Umgebung nicht direkt wahrnehmbar. Zudem lässt ein zu großer Zustandsraum (z. B. die Anzahl aller möglichen Werte der Pixel in Kamerabildern) eine solche Modellierung nicht zu (SUTTON & BARTO 2018, BOTVINICK et al. 2019). Ein wesentlicher Bestandteil der Anwendung von Reinforcement Learning liegt daher in der Approximation der Funktionen, welche zur Abbildung einer Strategie verwendet werden. Ein Ansatz hierfür stellen künstliche neuronale Netze bzw. Deep Learning dar, weshalb die hierauf basierenden Lernalgorithmen unter dem Begriff Deep Reinforcement Learning zusammengefasst werden (GRAESSER & KENG 2019).

In Abbildung 2.8 sind die grundsätzlichen Bestandteile eines auf Reinforcement Learning basierenden Agenten dargestellt, welche es vor der Trainingsphase zu konfigurieren gilt. Neben der Modellierung von  $\pi(a|s)$  und dem Lernalgorithmus, benötigt ein Agent während der Trainingsphase einen sogenannten *Replay Buffer*. In Diesem werden die bisherigen Erfahrungen in Form von  $(s', a, s, r)$ -Tupeln abgebildet, auf Basis derer eine Anpassung von  $\pi(a|s)$  erfolgt (FRANÇOIS-LAVET et al. 2018).

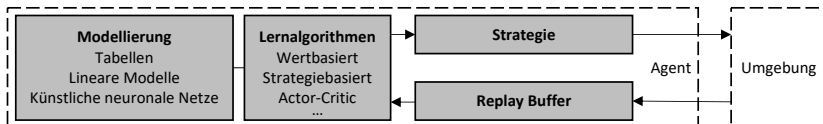


Abbildung 2.8: Grundsätzliche Bestandteile von Reinforcement-Learning-Ansätzen nach GATTI (2015), FRANÇOIS-LAVET et al. (2018)

### 2.3.2 Deep Learning

Auf algorithmischer Ebene besteht die zentrale Problemstellung bei maschinellen Lernverfahren darin, zu lernen, wie Daten sinnvoll transformiert werden können, um Repräsentationen der Daten zu erhalten, welche der Zielstellung näher kommen (CHOLLET 2021). Bei klassischen maschinellen Lernverfahren geschieht dies durch eine manuelle Implementierung von Mechanismen zur Ableitung von informationshaltigeren Repräsentationen der Rohdaten (sog. Feature Engineering) und klassischen Lernalgorithmen wie der linearen Regression, welche eine weitere Transformation der Daten erlernen (WANG et al. 2018).

Deep Learning basiert dagegen auf der Komposition mehrerer Schichten (siehe Abbildung 2.9), in welchen der Backpropagation-Algorithmus verwendet wird. Backpropagation beschreibt den Ansatz, durch den die durch eine Funktion  $J(y)$  durchgeführte Bewertung der Ausgänge  $y$  aus einer Schicht zur Anpassung der Gewichte in der vorherigen Schicht verwendet werden.  $J(y)$  kann als eine Verlustfunktion definiert werden (z. B. basierend auf dem Vorhersagefehler in überwachtem Lernen), welche es zu minimieren gilt, oder eine Zielfunktion, welche es zu maximieren gilt (CHOLLET 2021).

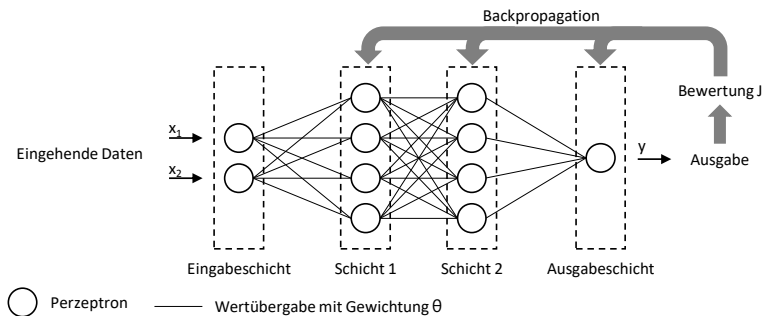


Abbildung 2.9: KNN mit zwei Eingängen, zwei Schichten mit je 4 Perzeptronen (verdeckte Schichten) und einem Ausgang zur Approximation der Funktion  $y(x_1, x_2)$  nach LECUN et al. (2015) und BURKOV (2019)

Angelehnt an die Neuronen im menschlichen Gehirn, werden diese Schichten vornehmlich durch künstliche neuronale Netze (KNN) umgesetzt, allerdings werden in Deep Learning auch Ansätze betrachtet, welche nicht biologischen Neuronen nachempfunden sind (GOODFELLOW et al. 2018, CHOLLET 2021). Die am generischsten einsetzbare

## 2 Grundlagen intelligenter Montagesysteme

---

Art der KNN stellen mehrschichtige Perzeptronen (engl. *Multilayer Perceptron*, MLP) dar (GRAESSER & KENG 2019). Ein Perzeptron ist eine Einheit, in welcher die eingehenden Werte mit einem jeweiligen Gewicht multipliziert werden und die Summe der gewichteten Werte in eine nichtlineare Funktion (sog. Aktivierungsfunktion) überführt wird (siehe Abbildung 2.10a).

Diese Gewichte werden während des Lernprozesses durch ein Gradientenverfahren angepasst und stellen daher Lernparameter  $\theta$  dar. Theoretisch ließe sich durch eine beliebige Anzahl an Perzeptronen in einer Schicht jede beliebige mathematische Funktion approximieren. Durch den Einsatz von mehreren Schichten lassen sich komplexere Funktionen mit einer geringeren Anzahl an Lernparametern effizienter erlernen.

Eine weitere Klasse der KNN stellen faltende neuronale Netze (engl. *Convolutional Neural Networks*, CNN) dar, welche unter anderem bei Bilddaten eingesetzt werden (vgl. Abbildung 2.10b). Hierbei wird die lokale Abhängigkeit der Pixel in der Modellierung berücksichtigt, indem auf einzelne Raster umliegender Pixel Faltungen (engl. *convolutions*) durchgeführt werden und die resultierenden Werte für die folgende Schichten weiterverarbeitet werden (z. B. durch Extraktion des höchsten Werts beim sog. *Max-Pooling*). Die Einheiten zur Durchführung der Faltungen, werden als *Kernel* bezeichnet und besitzen lernbare Gewichte, welche die Ausprägung der Faltung bestimmen. (RASCHKA & MIRJALILI 2018)

Rekurrente neuronale Netze (engl. *Recurrent Neural Networks*, RNN) sind eine weitere Klasse, welche unter anderem für das Lernen auf Basis sequentiell eingehender Daten wie Zeitreihen eingesetzt werden (vgl. Abbildung 2.10c). In *Reinforcement Learning* kann die optimale Aktion oftmals nicht nur auf Basis des aktuellen Zustands abgeleitet werden, da hierfür Informationen aus vergangenen Zeitschritten notwendig sind. Ein Beispiel hierfür kann die Position von Objekten sein, welche nicht durchgängig durch die Kamera erfasst wird. Durch RNN lässt sich ein Gedächtnis in KNN modellieren, indem eine Repräsentation der Sequenz an Erfahrungen (sog. *Hidden State*) modelliert und diese Repräsentation auf Basis neuer Erfahrungen angepasst wird. (GRAESSER & KENG 2019)

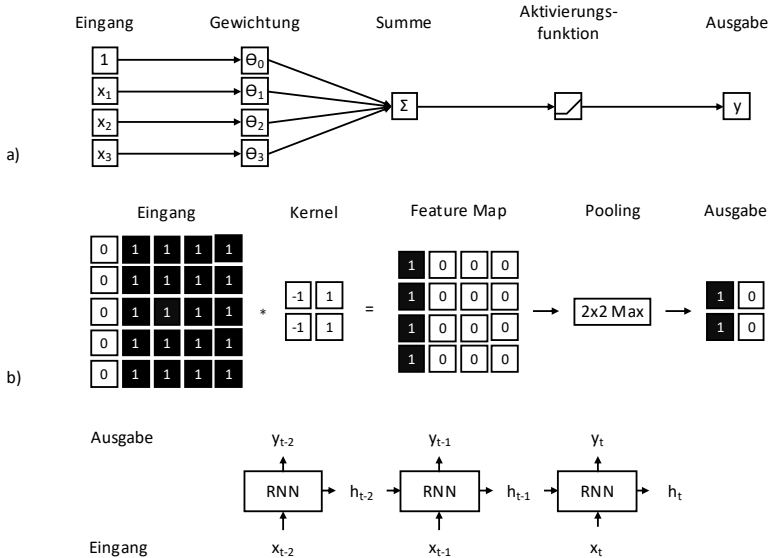


Abbildung 2.10: Schematische Darstellung möglicher Umsetzungen der Schichten in einem KNN: a) Perzeptron mit 3 Eingangswerten b) CNN mit beispielhafter Kanten-Erkennung c) RNN mit Weitergabe des Hidden State h über mehrere Zeitschritte

### 2.3.3 Lernbare Funktionen in Deep Reinforcement Learning

Deep Reinforcement Learning bezieht sich nicht auf einen konkreten Algorithmus, sondern auf eine Klasse an Lernalgorithmen. Diese unterscheiden sich hinsichtlich der Funktionen, welche durch Interaktion mit einer Umgebung gelernt werden und in Deep Reinforcement Learning durch KNN approximiert werden. So können, um eine Strategie innerhalb eines in Abschnitt 2.3.1 beschriebenen Markow-Entscheidungsproblems zu erlernen, unterschiedliche Funktionen innerhalb der Trainingsphase gelernt werden:

- $V^\pi(s)$ : Wertefunktion eines Zustands,
- $Q^\pi(a, s)$ : Wertefunktion von Aktion-Zustand-Paaren,
- $\pi(a|s)$ : Direktes Lernen der Strategie. (SUTTON & BARTO 2018)

## 2 Grundlagen intelligenter Montagesysteme

Wertefunktionen bilden den Erwartungswert von  $R$  auf Basis der jeweiligen Zustände bzw. Aktionen ab. In deterministischen Umgebungen ergibt sich die Strategie durch die Auswahl der Aktion, welche zu einem folgenden Zustand mit maximalen  $V^\pi(s) \sim \mathbb{E}[R_t | s_t = s]$  führt.

In stochastischen Umgebungen können die gleichen Aktionen zu unterschiedlichen Folgezuständen führen. Zudem ist die Dynamik der Umgebung  $P(s' | s, a)$  und somit die zur Erreichung eines Zustands notwendige Aktion oftmals nicht bekannt. Daher wird in diesen Fällen die Wertefunktion von Aktion-Zustand-Paaren  $Q^\pi(a, s)$  gelernt. Die Strategie ergibt sich durch die Auswahl der Aktion mit dem maximalen Wert bei einem definierten Zustand  $Q^\pi(a, s) \sim \mathbb{E}[R_t | s_t = s, a_t = a]$ .

In strategiebasierten Lernalgorithmen wird die Auswahl einer Aktion zu einem wahrgenommenen Zustand  $a \sim \pi(a|s)$  direkt gelernt. Die Lernparameter der Repräsentation von  $\pi(a|s)$  werden in diesem Ansatz durch die Bewertung der ausgeführten Aktionen hinsichtlich der Maximierung von  $R$  während der Trainingsphase angepasst. Während wertebasierte Lernalgorithmen im Allgemeinen effizienter sind, beschränkt sich deren Einsatzbereich zumeist auf diskrete Aktionsräume (GRAESSER & KENG 2019).

Durch das direkte Lernen der auszuführenden Aktionen über  $\pi(a|s)$  eignen sich strategiebasierte Ansätze insbesondere für kontinuierliche Aktionsräume, weshalb im Bereich der Robotik vornehmlich hierauf aufbauende Lernalgorithmen Einsatz finden (SICILIANO & KHATIB 2016). Der Grundgedanke strategiebasierter Lernalgorithmen wird durch den Ansatz REINFORCE abgebildet (siehe Abschnitt 2.3.4).

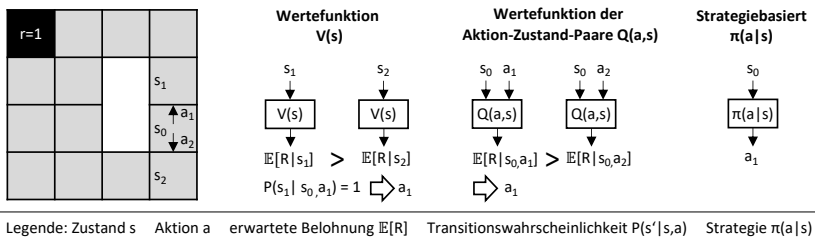


Abbildung 2.11: Umsetzung der Strategie eines Agenten basierend auf den drei lernbaren Funktionen  $V(s)$ ,  $Q(a,s)$  und  $\pi(a|s)$  im beispielhaften Markow-Entscheidungsproblem aus Abbildung 2.7

### 2.3.4 Strategiebasierte Lernalgorithmen

WILLIAMS (1992) stellt den strategiebasierten Ansatz REINFORCE zum Erlernen der Strategie  $\pi_\theta$  durch Anpassung der Lernparameter  $\theta$  basierend auf einer Zielfunktion  $J(\pi_\theta)$  vor. Hierbei wird das Ziel verfolgt, den Erwartungswert der Belohnung  $R$  entlang der sich durch  $\pi_\theta$  ergebenden Trajektorie  $\tau$  zu maximieren:

$$\max_{\theta} J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)]. \quad (2.2)$$

Die Lernparameter werden basierend auf dem Gradienten der Zielfunktion  $\nabla_{\theta} J(\pi_\theta)$  (sog. *Policy Gradient*) und einem vordefinierten skalaren Wert  $\alpha$  (Lernrate) aktualisiert (sog. *Gradient Ascent*):

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\pi_\theta). \quad (2.3)$$

Da sich die Belohnung  $R$  nicht als Funktion von  $\theta$  abbilden lässt, lässt sich auch  $\nabla_{\theta} J(\pi_\theta)$  nicht direkt berechnen. Allerdings lässt sich  $\nabla_{\theta} J(\pi_\theta)$  durch die Ausführung mehrerer Trajektorien (sog. *Monte Carlo Sampling*) in der Form

$$\nabla_{\theta} J(\pi_\theta) \approx \sum_{t=0}^T R_t(\tau) \nabla_{\theta} \log \pi_\theta(a_t | s_t) \quad (2.4)$$

auf Basis von Interaktionen mit der Umgebung approximieren (siehe Abbildung 2.12). Der Ausdruck  $\nabla_{\theta} \log \pi_\theta(a_t | s_t)$  lässt sich aus dem KNN, welches die Strategie abbildet (sog. *Policy Network*), berechnen. Es gibt den Gradienten der Wahrscheinlichkeit  $p(\tau | \theta)$  wieder, dass eine Trajektorie basierend auf  $\theta$  ausgeführt wird:

$$\nabla_{\theta} \log \pi_\theta(a_t | s_t) = \nabla_{\theta} \log p(\tau | \theta) = \frac{\nabla_{\theta} p(\tau | \theta)}{p(\tau | \theta)}. \quad (2.5)$$

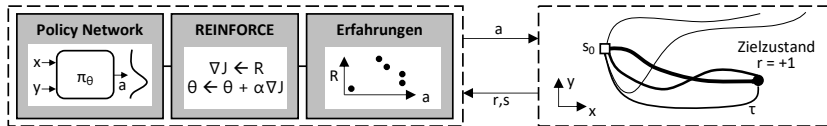


Abbildung 2.12: Anpassung der Lernparameter der Strategie (z. B. Auswahl des Fahrtwinkels bei den wahrgenommenen Positionen  $x, y$ ) basierend auf Erfahrungen aus der Umgebung in REINFORCE

### 2.4 Zusammenfassung

Zur Verfügung stehende Aktoren wie Knickarmroboter ermöglichen flexible Montagesysteme, welche durch softwareseitige Rekonfigurationen an neue Produkte angepasst werden können. Durch den Ansatz der aufgabenorientierten Programmierung können der Aufwand und die notwendigen Kompetenzen zur Konfiguration bzw. Rekonfiguration reduziert werden.

Allerdings ist die Fähigkeit automatisierter Montagesysteme, sich selbstständig an Störgrößen im Prozess anzupassen, beschränkt. Hierdurch können die Rahmenbedingungen von produzierenden Unternehmen und insbesondere von KMU nicht erfüllt werden.

Die zur Steigerung des Automatisierungsgrads in der Industrie notwendige Flexibilität und Anpassungsfähigkeit von Montagesystemen erfordert eine Robustheit der Softwaremodule gegenüber variierender Merkmale der Produkte (z. B. Geometrien) und Prozessbedingungen (z. B. Ablagepositionen von Bauteilen). Analog zu robusten Regelungssystemen ist hierzu eine Aufnahme und Verarbeitung dieser Varianz notwendig, um das Verhalten des Montagesystems anpassen zu können.

Sensoren ermöglichen die Wahrnehmung variierender Merkmale der Umgebung, welche in intelligenten Robotersystemen durch entsprechende Vorverarbeitung der Sensordaten und Planungsalgorithmen eine Anpassungsfähigkeit der Montagesysteme gewährleisten. Während die Hardware zur Umsetzung intelligenter Montagesysteme günstiger geworden ist, stellt die Komplexität der Software eine wesentliche Hürde dar, um solch anpassungsfähige Montagesysteme in KMU wirtschaftlich umzusetzen.

Deep Reinforcement Learning stellt einen Ansatz zur Umsetzung intelligenter Montagesysteme dar, in welchem die Software in Form eines intelligenten Agenten eine direkte Ableitung der notwendigen Steuerungssignale aus Sensorsignalen ermöglicht. Durch Interaktion mit der Umgebung wird zudem ein Umgang mit variierenden Merkmalen der Umgebung gelernt, wodurch in der Betriebsphase eine echtzeitfähige Reaktion auf Veränderungen ermöglicht wird.

Da nicht der gesamte Prozess eine Reaktion auf variierende Umgebungsbedingungen erfordert, können Teilprozesse durch statische Steuerungsprogramme umgesetzt werden, wodurch sich die Aufgabe eines intelligenten Agenten einschränken lässt. Allerdings erfordern die Definition der Teilprozesse und die Umsetzung des Agenten Expertenwissen, welches in KMU nicht vorhanden ist.



Daher wird in dieser Arbeit eine Integration von DRL in die aufgabenorientierte Programmierung von Montagesystemen angestrebt. Wie in Abbildung 2.13 dargestellt, ermöglicht die aufgabenorientierte Konfiguration intelligenter Agenten eine Reduktion des Aufwands entlang des gesamten Lebenszyklus eines Montagesystems.

In der Planungsphase ist eine Implementierung der intelligenten Agenten notwendig, um Planungsalternativen anhand eines physischen oder simulierten Montagesystems validieren und bewerten zu können. In der Realisierungsphase erfolgt die Umsetzung des physischen Montagesystems durch Implementierung der Software auf der entsprechenden Hardware wie eingebettete GPU. Unvorhersehbare Änderungen der Varianz (z. B. neue Produktvarianten) erfordern zudem eine Rekonfiguration der Software.

Nach einer Darstellung des Stands der Wissenschaft und Technik in Kapitel 3 erfolgt in Kapitel 4 daher eine Einschränkung des Betrachtungsbereichs.

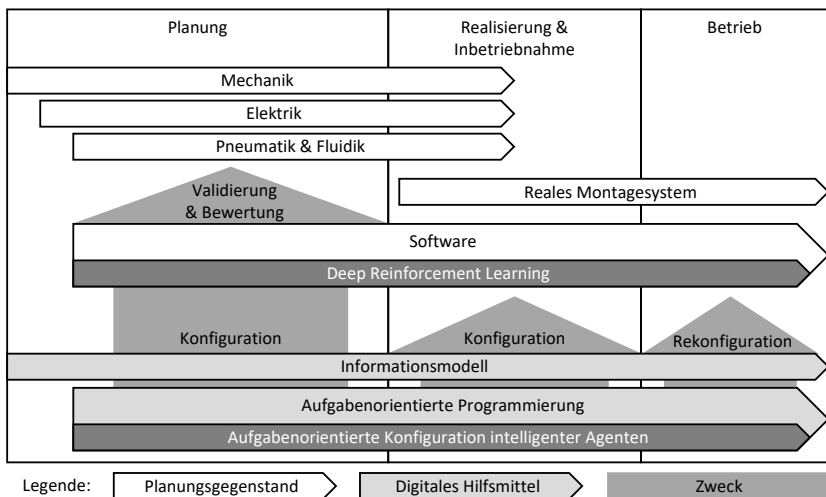


Abbildung 2.13: Einordnung der aufgabenorientierten Konfiguration intelligenter Agenten in den Lebenszyklus roboterbasierter Montagesysteme



## **3 Stand der Wissenschaft und Technik**

Zur Identifikation von Möglichkeiten zur Integration von Deep Reinforcement Learning in die aufgabenorientierte Programmierung werden in diesem Kapitel zunächst aktuelle Ansätze zur rechnergestützten Umsetzung roboterbasierter Montagesysteme aufgezeigt. Da Deep Reinforcement Learning selbst ein Forschungsfeld darstellt, wird in Abschnitt 3.2 eine Übersicht aktueller Ansätze zur Umsetzung eines Agenten und der simulativen Lernumgebung gegeben. Zudem werden in Abschnitt 3.4 Forschungsergebnisse in der Automatisierung der Konfiguration maschineller Lernverfahren aufgezeigt, auf deren Ansätzen im Rahmen der aufgabenorientierten Konfiguration des Agenten aufgebaut werden kann.

### **3.1 Rechnergestützte Umsetzung roboterbasierter Montagesysteme**

#### **3.1.1 Herleitung von Skills in der Montage**

Skills werden in der Entwicklung von Montagesystemen eingesetzt, um die Fähigkeiten einzelner Ressourcen abzubilden und Montageaufgaben in die notwendigen Teilaufgaben aufzuteilen. Sie stellen somit semantische Elemente zur Modellierung von Ressourcen und Montageaufgaben dar, welche in rechnergestützten Werkzeugen zur Planung und Umsetzung von Montagesystemen eingesetzt werden.

SMALÉ (2011) stellt einen Ansatz zur Modellierung von Skills zur Ressourcenauswahl in der Mikromontage vor. Die Skills werden den sechs übergeordneten Klassen "Bewegen", "Fügen", "Sichern", "Messen", "Zuführen" und anderweitigen Skills (z. B. Bearbeiten) zugeordnet. Für die Mikromontage werden diesen abstrakten Klassen Rahmenbedingungen zugeteilt (z. B. "Reinraum") und diese den Anforderungen seitens Produkt und Prozess zugeordnet.

### 3 Stand der Wissenschaft und Technik

MICHNIEWICZ (2018) setzt Skills zur automatischen Arbeitsplanung in der Montage ein. Die Skills werden auf zwei Ebenen definiert: Funktionsprimitiva und Tasks. Die Klassifizierung der Funktionsprimitiva orientiert sich an der Taxonomie der VDI 2860 und wird in "Bewegen", "Verbinden", "Kontrollieren", "Speichern" und "Sonderoperationen" mit jeweiligen Unterklassen unterteilt. Zur Ressourcenauswahl werden diese Funktionsprimitiva in Anforderungs-Funktionsprimitiva, welche dem Produkt zugeordnet werden, und in Fähigkeits-Funktionsprimitiva, welche den Ressourcen zugeteilt werden, unterteilt. Durch die Zusammenführung von Funktionsprimitiva werden wiederkehrende Sequenzen in Tasks wie "Zusammensetzen zweier Bauteile" zusammengefasst.

HAMMERSTINGL (2020) leitet Skills zur automatisierten Ressourcenauswahl her. In diesem Ansatz wird zwischen elementaren und zusammengesetzten Skills, welche die Skills einzelner Ressourcen zu höherwertigen abstrakten Funktionalitäten zusammenführen, unterschieden (vgl. Abbildung 3.1). Die elementaren Skills werden in "Handhaben", "Verbinden", "Kontrollieren", "Datenaustausch", "Mensch-Maschine-Interaktion" und "Sonderfunktionen" unterteilt. Ein Fokus wird auf die Wahrnehmung physikalischer Größen wie "geometrische Größen" (z. B. Position) und "mechanische Größen" (z. B. Kraft) über Sensorik gelegt. Auch wird der Datenaustausch wie das Lesen und Aktualisieren von Daten als eigenständige Skills der Ressourcen mit aufgenommen. Die zusammengesetzten Skills bestehen aus einer Sequenz von Skills.

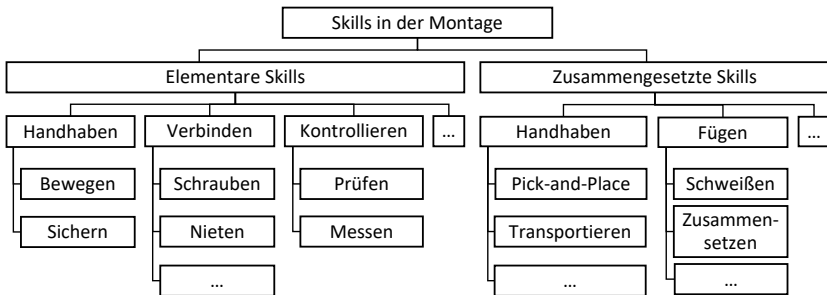


Abbildung 3.1: Auszug der Skill-Taxonomie nach HAMMERSTINGL (2020)

JÄRVENPÄÄ et al. (2019) leiten eine Ontologie für Skills von Produktionssystemen zur Ressourcenauswahl in der Web Ontology Language (OWL) her. Zum Abgleich produkt-seitiger Anforderungen und ressourcenseitiger Skills wird eine Prozesstaxonomie verwendet.

### 3.1 Rechnergestützte Umsetzung roboterbasierter Montagesysteme

---

Hierbei werden zusammengesetzte Skills wie "Transportieren" und "Nehmen" durch die Zuordnung von einfachen Skills oder anderen kombinierten Skills über "UND" bzw. "ODER" Beziehungen definiert (z. B. "Nehmen" = ("Fingergreifen" ODER "Vakuumgreifen") UND "Bewegen").

GONNERMANN et al. (2020) definieren Skills von Sensoren für den Anwendungsfall des Prozessmonitorings, welche auf Vorarbeiten von HAMMERSTINGL et al. (2015) aufbauen. In der Arbeit wurden basierend auf einer Literaturrecherche 73 sensorische Skills identifiziert. Durch die Kombination von elementaren Skills durch mathematische Beziehungen wie der Berechnung der elektrischen Leistung über Spannung und Strom, werden kombinierte Skills über die Semantic Web Rule Language (SWRL) in der Taxonomie berücksichtigt.

#### 3.1.2 Ansätze zur Vereinfachung der Programmierung von Robotersystemen

Die Programmierung roboterbasierter Montagesysteme benötigt analog zur Umsetzung autonomer Montagesysteme entsprechende Aufwände und Expertenwissen. Durch die Vereinfachung der Programmierung soll somit die wirtschaftliche Umsetzung von Automationslösungen ermöglicht werden.

Die Vereinfachung der Programmierung von roboterbasierten Montagesystemen durch Abstraktion der Aufgabenstellung wurde bereits in diversen frühzeitigen Arbeiten verfolgt (HUMBURGER 1998, KUGELMANN 1999, WECK & BRECHER 2006, EHRMANN 2007). Im Folgenden wird der Fokus auf Arbeiten gelegt, welche auf diesen Vorarbeiten aufbauen, und daher auf ältere Ansätze nicht näher eingegangen.

BACKHAUS (2016) stellt ein System zur aufgabenorientierten Programmierung automatisierter Montagesysteme vor, welches auf einer skillbasierten Ablaufbeschreibung aufbaut (vgl. Abbildung 3.2). In der Arbeit werden Skills (z. B. "Führen Linear", "Sichern") zur Durchführung von Montageprozesse wie Handhaben und Schweißen definiert, welche sich einzelnen Ressourcen zuordnen lassen. Das Planungsmodul basiert auf einer Blackboard Architektur, in welcher einzelne Wissensquellen für einzelne Lösungsschritte, wie der simulationsbasierten Planung des Schweißprozesses, eingesetzt werden.

### 3 Stand der Wissenschaft und Technik

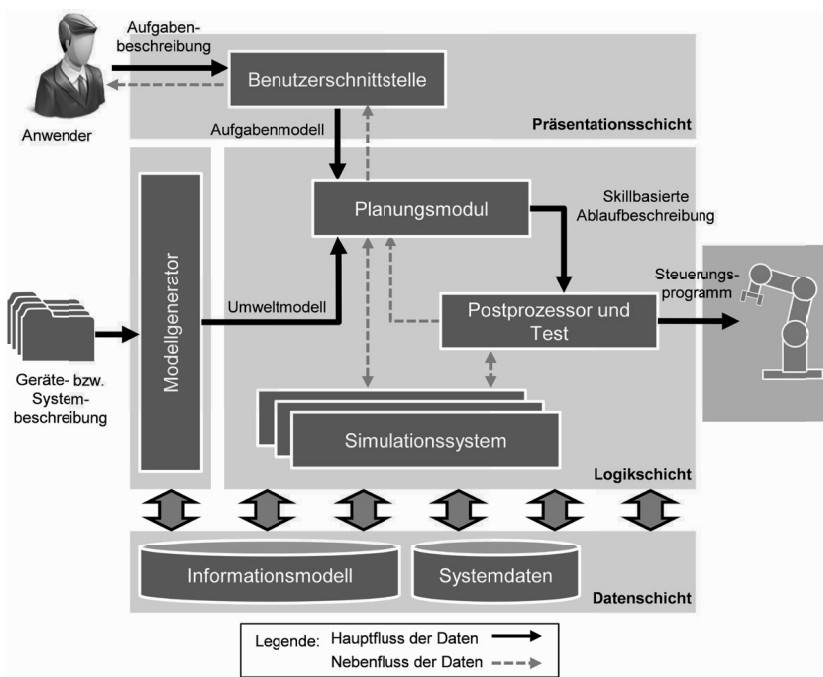


Abbildung 3.2: Konzept der Aufgabenorientierten Programmierung von Montagesystemen (BACKHAUS 2016)

Die Modellierung der Aufgabe (Aufgabenmodell) und der Hardware-Konfiguration des Montagesystems (Umweltmodell) erfolgt im Standard AutomationML (Automation Markup Language). Die Basis von AutomationML liegt in der Beschreibung von Automationssystemen über das XML-basierte Datenformat CAEX (Computer Aided Engineering Exchange), welches über die Verknüpfung zusätzlicher Daten wie Geometrien über den Standard Collada und Sequenzen über den Standard PLCopen XML abgebildet werden können (DRATH 2010).

HUA et al. (2016) automatisieren die Code-Generierung von auf ROS (Robot Operating System) basierenden Robotersystemen. Die Modellierung des Robotersystems erfolgt über AutomationML. Die Beschreibung der Systemkomponenten wird in CAEX um Taxonomien für roboterspezifische Hardwarekomponenten und Kommunikationsschnittstellen sowie ROS erweitert.

### 3.1 Rechnergestützte Umsetzung roboterbasierter Montagesysteme

---

Die Inhalte des CAEX-Dokuments werden in die semantische Beschreibungssprache RDF (Resource Description Framework) in Form von RDF-Graphen transformiert. Auf Basis dieser RDF-Graphen und dem von BUBECK et al. vorgestellten Tool BRIDE erfolgt eine automatische Generierung von ROS-Code.

HALT et al. (2018) definieren zwei roboterbasierte Prozesse zum Zusammensetzen von Bauteilen auf Basis der in VDI 2860 definierten Teilfunktionen. In der Arbeit werden sechs Teilfunktionen und deren Merkmale identifiziert (u. a. Bewegung mit definierten Geschwindigkeiten durch Geschwindigkeitsvektoren), auf deren Basis eine Vielzahl an Montageprozessen beschrieben werden kann. Die Teilfunktionen werden über Finite State Machines und ROS beschrieben. In weiteren Arbeiten durch NÄGELE (2021) wird ein Skill-Modell für kraftgeregelte Montageprozesse entwickelt, wobei die Übertragbarkeit der Skills auf unterschiedliche Ressourcen im Fokus steht.

TRAPANI (2019) stellt in seiner Arbeit ein System zur aufgabenorientierten Programmierung von Robotersystemen mit mehreren Robotern vor. Die Fokus der Arbeit liegt auf der Ableitung möglicher Abfolgen von Aufgaben und deren Zuordnung zu Ressourcen zur Erfüllung einer übergeordneten Aufgabe, welche durch mehrere Roboter in einer Zelle flexibel ausgeführt werden können. Die Ausführung der Aufgaben wird auf Basis einer Simulation des Robotersystems geplant.

BERG (2020) stellt ein System zur aufgabenorientierten Programmierung für die Mensch-Roboter-Kooperation vor. Die aus den CAD-Daten des Produkts abgeleitete Montagereihenfolge und Aufgaben werden durch eine Erweiterung der Skillbeschreibung nach HAMMERSTINGL & REINHART 2017 um zusätzliche Kriterien dem Roboter und dem Menschen zugeteilt. Neben der Planung und Programmierung des Roboterprogramms sowie der Überprüfung in einer Simulation, wird zusätzlich eine Tätigkeitserkennung des Menschen implementiert, welche in der Betriebsphase eine Anpassung des Roboters an den Menschen ermöglicht.

KUSS (2020) beschreibt ein System zur aufgabenorientierten Programmierung von Schweißrobotern, wobei mögliche Geometrieabweichungen in der Betriebsphase berücksichtigt werden. Zusätzlich zur Modellierung von Produkten, Prozessen und Ressourcen in AutomationML wird ein Technologiemodell eingeführt, welches Wissen zu einer bestimmten Fertigungstechnologie repräsentiert. Auf Basis der geometrischen und materialbezogenen Daten des Produkts, erfolgt eine automatische Ableitung der Werkzeugpositionierung, der Roboterbahn und der Schweißprozessparameter.

### 3 Stand der Wissenschaft und Technik

---

Durch eine 3D-Sensormessung können Abweichungen hinsichtlich der Lage und Form des Bauteils in der Betriebsphase erkannt werden und über das System eine Anpassung des Roboterprogramms erfolgen.

JOHANNSMEIERS et al. (2018) nutzen Skill-Formalismen zur Reduktion des Lösungsraums bei der Optimierung eines auf Kraftregelung basierenden Robotersystems beim Zusammensetzen von Objekten in Führungen (engl. *Peg-in-Hole*). Der Fügeprozess wird durch fünf sequentielle Skills zum Annähern bis zum Einsetzen unterteilt. Durch die Reduktion der Freiheitsgrade im Rahmen der Regelung basierend auf den jeweiligen Skills kann der Optimierungsprozess beschleunigt werden. LÄMMLE et al. (2020) bauen auf dieser Arbeit auf und setzen die Skill-Sequenz beim Erlernen der Aufgabe durch Deep Reinforcement Learning in einer Simulation um.

Im Projekt SMErobotics wurden Skills im Rahmen der Montage zur Handhabung und Verschraubung von Bauteilen eingesetzt, um eine Rekonfiguration des Robotersystems zu vereinfachen. Die Skills wurden als einzelne Funktionsblöcke implementiert, welche Module zur Wahrnehmung und Planung beinhalten (NOTTENSTEINER et al. 2016). Im Anschlussprojekt DEAL wurde der Ansatz für autonome Montagesysteme erweitert und einzelne Skills zur Wahrnehmung und Planung bei kamerabasierten Robotersystemen in einer Peg-in-Hole-Aufgabe konzeptioniert (NOTTENSTEINER et al. 2021).

WIESE et al. (2022) fokussieren die Mensch-Maschine-Schnittstelle in der Skill-basierten Programmierung von Robotersystemen. Das größte Potential zur Reduktion der manuellen Aufwände wird in Speicherung bereits definierter Skillsequenzen für abstraktere Aufgaben (z.B. "Platziere X auf Y") gesehen.

## 3.2 Aktuelle Ansätze zur Umsetzung des Agenten

### 3.2.1 Proximal Policy Optimization

Die algorithmischen Herausforderungen in der Trainingsphase eines Agenten liegen in der Effizienz, aus den gesammelten Erfahrungen zu lernen, in einer geeigneten Exploration unterschiedlicher Strategien und in der Fähigkeit, Auswirkungen einzelner Aktionen auf die spätere Zielerreichung bewerten zu können (OSBAND et al. 2019). Durch Fortschritte im Umgang mit diesen Herausforderungen werden in aktuellen Forschungsarbeiten somit RL-Algorithmen entwickelt, durch welche zunehmend komplexe Aufgabenstellungen mit geringen Trainingszeiten gelernt werden können.



### 3.2 Aktuelle Ansätze zur Umsetzung des Agenten

Ein Ansatz besteht in der Kombination von wertbasierten und strategiebasierten Ansätzen (siehe Abbildung 3.3). Die Auswahl der Aktion erfolgt analog zu strategiebasierten Ansätzen durch das direkte Erlernen der Strategie  $\pi(a|s)$  im Policy Network.

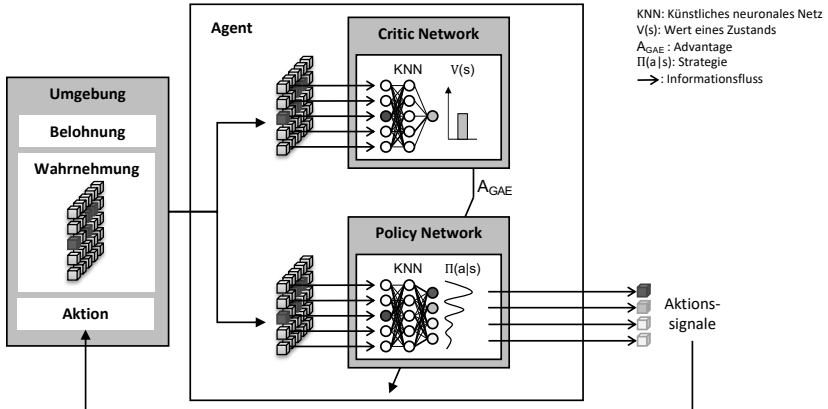


Abbildung 3.3: Informationsfluss im RL-Algorithmus PPO basierend auf einem Critic Network und einem Policy Network

Eine Herausforderung beim Erlernen von  $\pi(a|s)$  basierend auf der Belohnung ist, dass in Abhängigkeit der Belohnungsfunktion und der Dynamik der Umgebung unterschiedliche Ausprägungen der Belohnungssignale möglich sind.

Dies erschwert eine Bewertung unterschiedlicher Strategien in neuen Umgebungen, da der absolute Wert der Belohnung noch keinen Informationsgehalt zum Grad der Zielerreichung beinhaltet. Um einen effizienteren Lernprozess zu ermöglichen, wird mit dem Ansatz des *Advantage Actor Critic* (Mnih et al. 2016) daher zusätzlich der Zusammenhang erlernt (sog. *Critic Network*), in dem die erwartete Belohnung in einem Zustand  $V^\pi(s)$  erfasst wird. Durch diesen Referenzwert ergibt sich der Vorteil einer neu ausgeführten Aktion (sog. *Advantage*) zu

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s). \quad (3.1)$$

Mit  $A^\pi(s, a)$  wird somit ein Wert berechnet, welcher aussagt, wie weit sich die erwartete Belohnung durch eine ausgewählte Aktion  $Q^\pi(s, a)$  im Vergleich zum bisherigen Erwartungswert  $V(s)$  erhöht. Durch den höheren Informationsgehalt dieser Bewertung wird ein effizienterer Lernprozess von  $\pi(a|s)$  ermöglicht.

### 3 Stand der Wissenschaft und Technik

---

SCHULMAN et al. (2016) führen hierauf aufbauend den Ansatz der *Generalized Advantage Estimation* (GAE) ein. Der Wert einer Aktion wird in GAE nicht durch eine zusätzliche Funktion  $Q^\pi(s, a)$  approximiert, sondern durch das tatsächlich erhaltene Belohnungssignal und den Wert des Folgezustands bei der Ausführung einer Aktion über  $r_t + \gamma V^\pi(s_{t+1})$  abgebildet. Hierdurch reduziert sich die Berechnung des Werts einer Aktion auf die Approximation von  $V^\pi(s)$  im Ausdruck

$$\delta_t = r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t). \quad (3.2)$$

Da die Belohnungssignale einer hohen Varianz unterliegen können, wird in GAE der gewichtete Durchschnitt des Advantage über mehrere Zeitschritte mit

$$A_{GAE}^\pi(s_t, a_t) = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + (\gamma\lambda)^{T-t-1}\delta_{T-1} \quad (3.3)$$

berechnet.

Neben dem Discount-Faktor  $\gamma$  benötigt GAE somit die zusätzlichen Hyperparameter  $\lambda$  im Wertebereich  $[0,1]$  und die Anzahl an Zeitschritten  $T$  (sog. *Horizont*), nach denen die in einer Umgebung gesammelten Erfahrungen zur Bewertung der durch das Policy Network ausgewählten Aktion herangezogen werden.

In dieser Arbeit wird der von SCHULMAN et al. (2017) vorgestellte RL-Algorithmus *Proximal Policy Optimization* (PPO) verwendet, welcher zum Zeitpunkt der Erstellung dieser Arbeit den Stand der Technik im Bereich der Robotik darstellt (ECOFFET et al. 2021). Auch in anderen Anwendungsbereichen wie dem Chatbot ChatGPT (OPENAI 2023) stellt PPO einen aktuellen Ansatz zur Umsetzung intelligenter Systeme dar.

Zusätzlich zur Bewertung der Strategie  $\pi(a|s)$  über  $A_{GAE}^\pi$  wird in PPO eine erweiterte Zielfunktion  $J$  eingesetzt, in welcher die Vorhersagegüte von  $V(s)$  und der Explorationsgrad mit aufgenommen werden. Für deren Erläuterungen wird an dieser Stelle auf GRAESSER & KENG (2019) verwiesen.

Im PPO-Ansatz nach SCHULMAN et al. (2017) besitzen die KNN des Critic Network und des Policy Network zudem dieselbe Netzarchitektur und teilen sich die lernbaren Gewichte  $\Theta$ , um die jeweiligen Werte von  $V(s)$  und die Strategie zu lernen.

### 3.2.2 Werkzeuge zur Implementierung des Agenten

Für die Implementierung von Agenten werden Programmiersprachen verwendet, in welchen durch Bibliotheken, welche aufrufbare Unterprogramme darstellen, die Softwareentwicklung vereinfacht wird. Bibliotheken zur Implementierung von Agenten ermöglichen eine Konfiguration der Parameter des RL-Algorithmus und der Architektur der KNN, auf deren Basis ein lernfähiger Agent generiert wird.

Als Schnittstelle zu Umgebungen hat sich die Programmierschnittstelle *Gym API* (Gym Application Programming Interface) etabliert (BROCKMAN et al. 2016). Über diese Schnittstelle erfolgt ein Austausch von Aktionssignalen seitens des Agenten und ein Austausch von Umgebungsinformationen hinsichtlich der momentanen Wahrnehmung, des Belohnungssignals sowie eines booleschen Wertes, ob eine Episode beendet wurde.

Die Grundlage zur Implementierung eines auf DRL basierenden Agenten stellen Bibliotheken dar, welche eine Umsetzung von KNN ermöglichen (z. B. Tensorflow, PyTorch). Diese werden zur Konfiguration der Architektur des KNN in Bibliotheken genutzt, welche RL-Algorithmen wie PPO mit Vorgabe der entsprechenden Parameter umsetzen (z. B. Stable Baseline, Keras-RL, TF Agents).

Es existieren weitere Bibliotheken, welche sich auf spezifische Anforderungen fokussieren. So stellt Rllib (E. LIANG et al. 2017) eine Bibliothek zum verteilten Rechnen von RL-Algorithmen auf Rechenclustern dar. Mit Horizon (GAUCI et al. 2018) wird seitens der Firma Facebook ein Werkzeug zur Anwendung von RL in Online-Services vorgestellt. In dieser Arbeit werden Bibliotheken fokussiert, welche eine automatische Implementierung von Agenten ohne die Notwendigkeit der Programmierung ermöglichen (siehe Abbildung 3.4).

So erfolgt die Implementierung des Agenten in der Python-Bibliothek *Unity ML Agents* (Unity Machine Learning Agents) basierend auf einer Definition der Hyperparameter des RL-Algorithmus und der Hyperparameter wie Lernrate und Anzahl der verdeckten Schichten des KNN in einer auf dem Datenformat YAML (YAML Ain't Markup Language) basierenden Konfigurationsdatei (JULIANI et al. 2018).

*SLM Lab* stellt ebenfalls eine auf Python basierende Bibliothek zur automatischen Implementierung von Agenten auf Grundlage auf einer Konfigurationsdatei im Datenformat JSON (JavaScript Object Notation) dar. Hier wird der Fokus auf eine hohe Flexibilität in der Konfiguration des Agenten und die Reproduzierbarkeit gelegt (KENG et al. 2019).

### 3 Stand der Wissenschaft und Technik

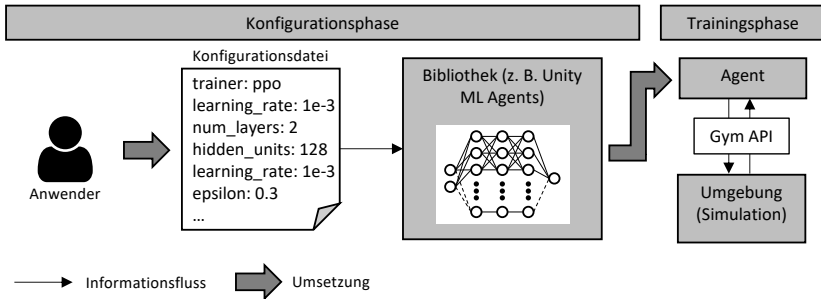


Abbildung 3.4: Schematischer Informationsfluss in Unity ML Agents und SLM Lab

Es existieren zudem kommerzielle Softwarewerkzeuge, welche ebenfalls eine Implementierung des Agenten ohne Programmierkenntnisse ermöglichen. Mit dem Dienst *Bonsai* wird in der Cloud-Umgebung Azure von Microsoft ein Werkzeug zur vereinfachten Umsetzung von Agenten angeboten. Das kommerzielle Softwarewerkzeug *Matlab*, welches unter anderem für die Umsetzung von Regelungssystemen und Simulationen eingesetzt wird, besitzt mit der Reinforcement Learning Toolbox eine Erweiterung, in welcher eine Konfiguration und Umsetzung des Agenten erfolgt.

### 3.3 Umsetzung der Lernumgebung

#### 3.3.1 Ausgewählte montagerelevante Belohnungsfunktionen

Die Aufgabenbeschreibung erfolgt in Reinforcement Learning mittels einer Belohnungsfunktion, über welche aus dem Zustand der Umgebung ein skalares Belohnungssignal berechnet wird. Die Definition einer Belohnungsfunktion ist nicht trivial, allerdings kann in der Praxis auf bestehende Belohnungsfunktionen zurückgegriffen werden (WINDER 2020). Im Folgenden werden daher bestehende Gestaltungen der Belohnungsfunktion in für Montageprozesse relevanten Aufgabenstellungen aufgezeigt.

LOPEZ et al. (2019) definieren Handhabungsaufgaben basierend auf dem Abstand des Handhabungsobjekts und des Zielzustands hinsichtlich der Distanz  $x$  und dem Abstandswinkel  $y$  im kartesischen Raum. Zum *Positionieren* wird die Belohnung über die Distanz durch

$$r_D = \frac{e^{-ax} - e^{-a} + 10 \left( e^{-\frac{a}{b}x} - e^{-a} \right)}{1 - e^{-a}} \quad (3.4)$$

definiert. Die Variablen  $a$  und  $b$  stellen einstellbare Parameter dar und bestimmen die Skalierung der Belohnung bei Näherung ( $a$ ) bzw. die zusätzliche Skalierung der Belohnung im nahen Bereich ( $b$ ). Das Belohnungssignal zum *Orientieren* ergibt sich über den Abstandswinkel  $y$  zu

$$r_O = \frac{1 + c - \left( \frac{y}{\pi} \right)^d}{1 + c}. \quad (3.5)$$

Die Parameter  $c$  und  $d$  stellen einstellbare Parameter dar. Durch Kombination der Belohnungssignale zu Distanz und Abstandswinkel ergibt sich ein Belohnungssignal zum *Erreichen einer Pose* unter Berücksichtigung einer Kollision zu

$$r_P = \begin{cases} r_D r_O - 1 - e^{(r_D - 1)^f}, & \text{mit Kollision} \\ r_D r_O - 1, & \text{ohne Kollision} \end{cases} \quad (3.6)$$

mit den einstellbaren Parametern  $e$  und  $f$ .

JAMES & JOHNS (2016) definieren die Belohnungsfunktion zum *Greifen* bzw. *Heben* eines Objekts durch ein positives Belohnungssignal bei der Reduktion der Distanz  $x$  zum Objekt und bei Erhöhung des Abstands des Objekts vom Tisch  $z$  über

$$r_G = \begin{cases} 100, & z = z_{terminal} \\ 1 + z, & \text{Objekt gegriffen} \\ e^{-ax}, & \text{sonst.} \end{cases} \quad (3.7)$$

Der einstellbare Parameter  $a$  skaliert analog zu Gleichung 3.4 die Belohnung der Näherung zum Objekt. Die Episode wird bei Erreichung eines vordefinierten Abstands zum Tisch  $z_{terminal}$  mit einem zusätzlichen Belohnungssignal terminiert.

SCHOETTLER et al. (2019) untersuchen für die Aufgabe zum *Zusammensetzen* von Verbindungssteckern in entsprechende Buchsen verschiedene Arten, die Belohnungsfunktion zu gestalten. Die geringste Fehlerquote des Fügeprozesses wird durch die Integration der Näherung an den Zielzustand sowie der aufgebrauchten Kraft im phy-

### 3 Stand der Wissenschaft und Technik

---

sischen Versuchsaufbau in der Belohnungsfunktion erzielt. NUGROHO et al. (2023) teilen die Aufgabe für den RL-Agenten in die Teilaufgaben "Erreichen", "Pick-and-Place" und "Push" auf. "Erreichen" wird hierbei durch ein negatives Belohnungssignal von -1 definiert, wenn der Arbeitsbereich noch nicht erreicht wurde. Die anschließende "Pick-and-Place"-Aufgabe berücksichtigt zudem die Erreichung der Position der Buchse. Anschließend erfolgt das Zusammensetzen durch die "Push"-Aufgabe, in welcher Zielposition und -winkel des Handhabungsobjekts als Zielzustand in der Belohnungsfunktion definiert werden.

MEYES et al. (2017) und MARX et al. (2020) betrachten die Aufgabe einer Bahn zu folgen, um Anwendungsfälle wie *Schweiß- und Klebprozesse* abbilden zu können. Die Belohnung erfolgt durch ein positives Belohnungssignal bei Erreichung der Bahn bzw. durch Sanktionierung bei Abweichung von der Bahn.

#### 3.3.2 Simulative Lernumgebungen

Die Trainingsphase des Agenten kann auf Basis der Interaktion mit einer physischen Umgebung (z. B. Roboterzelle) erfolgen (SCHOETTLER et al. 2019). Allerdings unterliegt die Verwendung einer physischen Lernumgebung Restriktionen, welche in einer Simulation nicht vorliegen. Nach IBARZ et al. (2021) ergeben sich durch das Lernen in einer Simulation vier Vorteile:

- Die Dateneffizienz von KNN erfordert zeitliche Aufwände und einen hieraus resultierenden Hardware-seitigen Verschleiß zur Datengenerierung in der physischen Welt. Die Dauer der Trainingsphase kann durch beschleunigte Simulationen reduziert werden.
- Die zufällige Auswahl an Aktionen erschwert die Einhaltung von Sicherheitsaspekten, welche in einer Simulation nicht relevant sind.
- Die Lernumgebung muss nach jeder Trainings-Episode auf einen Ausgangspunkt zurückgesetzt werden. In einer Simulation ist dies ohne manuelle Eingriffe möglich.
- Das Belohnungssignal erfordert eine Messbarkeit der Zustände der Umgebung. In einer Simulation lassen sich die Zustände der Simulationsobjekte stets direkt auslesen.

Ein Ansatz zur Umsetzung eines Agenten besteht darin, diesen zunächst in einer *realistischen Simulation* zu trainieren und den trainierten Agenten anschließend an eine physische Umgebung zu übertragen (ZHAO et al. 2020). Die Qualität von physikbasierten Simulationen von Produktionssystemen ist zur Bewertung des kinematischen Verhaltens ausreichend (STICH 2017).

Dennoch können Abweichungen der Aktor-Dynamik und Latenzzeiten der Steuerungssignale eine direkte Übertragbarkeit des Agenten verhindern, weshalb die Simulation in bestehenden Arbeiten anhand gemessener Werte des physischen Systems kalibriert wird (HWANGBO et al. 2019, IBARZ et al. 2021). Um Abweichungen hinsichtlich der sensorischen Wahrnehmung entgegenzuwirken, werden zudem Sensoren wie Tiefenbildkameras und Laserscanner genutzt, welche sich über eine Simulation realitätsgetreu abbilden lassen (BREYER et al. 2018, SURMANN et al. 2020).

Es kann nicht davon ausgegangen werden, dass sich eine reale Umgebung stets ausreichend genau über eine Simulation abbilden lässt. Die zwei wesentlichen Ansätze zum Schließen dieser sogenannten *Realitätslücke* bestehen in Domain Adaptation und Domain Randomization (KLEEBERGER et al. 2020, ZHAO et al. 2020). Abbildung 3.5 zeigt beispielhafte Umsetzungen dieser Ansätze.

In *Domain Adaptation* werden Daten aus der physischen Umgebung genutzt, um eine einheitliche Repräsentation der simulierten und realen Wahrnehmung zu erhalten. So verwenden JAMES et al. (2018) Generative Adversarial Networks (GAN), um das Kamerabild bei einer Greif-Aufgabe für reale und simulierte Kamerabilder anzugleichen.

*Domain Randomization* verfolgt dagegen das Ziel, einen gegenüber den Abweichungen der Simulation robusten Agenten zu trainieren. So ist ein Übertragen des erlernten Agenten bei unkalibrierten Simulationsmodellen möglich, indem Simulationsparameter wie Reibungskoeffizienten und Massen innerhalb der Trainingsphase variiert werden (PENG et al. 2017). Auch kann eine Robustheit gegenüber Abweichungen in der Wahrnehmung in den Simulations-basierten Lernprozess eingebracht werden, indem Belichtungen und Texturen innerhalb der Simulation variiert werden (PINTO et al. 2017). In OPENAI et al. (2019) wird auf diese Art eine komplexe Handhabungsaufgabe in einer Simulation erlernt und der trainierte Agent auf die Steuerungshardware des realen Systems übertragen.

### 3 Stand der Wissenschaft und Technik

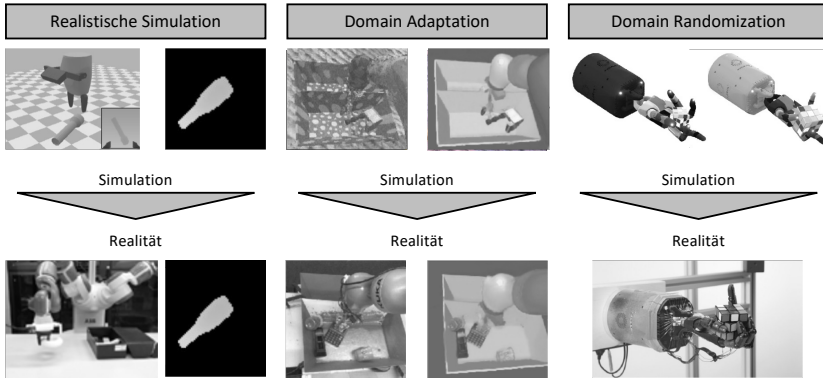


Abbildung 3.5: Ansätze zur Übertragung von Agenten aus einer Simulation auf reale Systeme durch realistische Simulationen (BREYER et al. 2018), Domain Adaptation (JAMES et al. 2018) und Domain Randomization (OPENAI et al. 2019)

## 3.4 Automatisiertes maschinelles Lernen in Deep Reinforcement Learning

### 3.4.1 Empirische Studien zur Konfiguration von Hyperparametern

Eine Herausforderung beim Einsatz von Deep Learning liegt im geringen theoretischen Wissen darüber, welche Einflüsse einzelne Hyperparameter auf den Lernprozess haben (HUTSON 2018, JORDAN 2019). Auch in Deep Reinforcement Learning ist kein explizites Wissen darüber vorhanden, wie die Hyperparameter der RL-Algorithmen den Lernprozess des Agenten beeinflussen (ENGSTROM et al. 2020).

GATTI (2015) stellt daher einen statistischen Versuchsplan auf, um den Einfluss verschiedener Hyperparameter in Deep Reinforcement Learning auf die Leistungsfähigkeit des Agenten nach dem Lernprozess in verschiedenen Lernumgebungen zu untersuchen. Die Ergebnisse können aufgrund der Stochastizität des Lernprozesses, welche unter anderem durch die zufällige Initialisierung der lernbaren Gewichte im KNN hervorgerufen wird, variieren. Es zeigt sich zudem, dass der Einfluss einzelner Hyperparameter von der jeweiligen Einstellung anderer Hyperparameter abhängt. So lassen sich bei dem verwendeten RL-Algorithmus TD( $\lambda$ ) einzelne Subregionen im Hyperparameter-Raum



### 3.4 Automatisiertes maschinelles Lernen in Deep Reinforcement Learning

ausmachen, in denen der Agent zu einer Lösung konvergiert bzw. die Leistung des Agenten höher als in anderen Hyperparameter-Bereichen ist (siehe Abbildung 3.6).

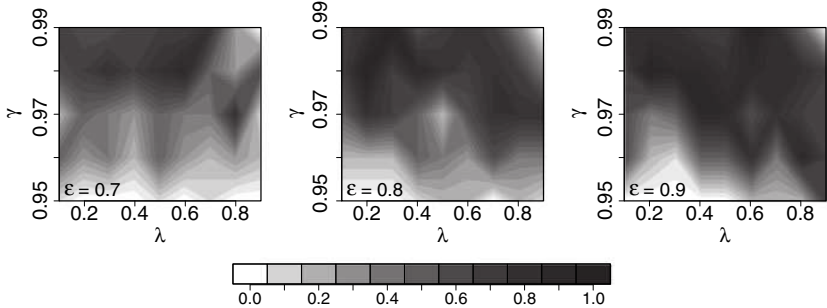


Abbildung 3.6: Wahrscheinlichkeit auf eine Lösung hin zu konvergieren, basierend auf den drei einstellbaren Hyperparametern  $\lambda$ ,  $\gamma$  und  $\epsilon$  (GATTI 2015)

MAHMOOD et al. (2018) untersuchen den Einfluss von neun Hyperparametern in PPO (u. a. Parameter des RL-Algorithmus wie  $\lambda$  und Parameter des KNN wie die Anzahl verdeckter Schichten und die Lernrate) bei der Aufgabe einen Koordinationenpunkt mit dem Endeffektor eines Roboters zu erreichen. In den Experimenten zeigt sich, dass die Ergebnisse bei der Verwendung gleicher Hyperparameter-Konfigurationen bei unterschiedlichen Roboterkinematiken korrelieren (siehe Abbildung 3.7). Allerdings ist zur Erreichung der maximalen Leistung stets eine Evaluation verschiedener Hyperparameter-Konfigurationen bzw. ein weiteres Anpassen der Hyperparameter für eine spezifische Aufgabe notwendig.

ANDRYCHOWICZ et al. (2020) führen eine empirische Studie zum Einfluss von über 50 möglichen algorithmischen Entscheidungen in PPO an fünf simulierten Roboteraufgaben durch. Auf Basis der 250.000 durchgeführten Lernprozesse lassen sich Empfehlungen zur Konfiguration des Lernalgorithmus geben (z. B.  $\lambda = 0.9$ ). Zudem wird die hohe Sensitivität des Lernprozesses auf einzelne Hyperparameter wie der Lernrate, der Batch-Größe sowie der Architektur des KNN gezeigt.

Die empirische Studie zeigt, dass die erfolgreiche Umsetzung eines Agenten maßgeblich von einer geeigneten Wahl der Hyperparameter abhängt. Basierend auf Erfahrungen aus bestehenden Aufgabenstellungen lassen sich Empfehlungen für Wertebereiche dieser Hyperparameter ableiten. Allerdings sind die optimalen Hyperparameter-

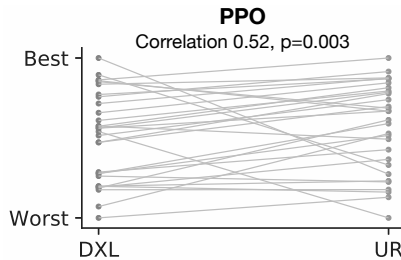


Abbildung 3.7: Wiederverwendung von Hyperparameter-Konfigurationen bei zwei ähnlichen Roboteraufgaben nach MAHMOOD et al. (2018) (Wahrscheinlichkeit, dass kein Zusammenhang besteht < 0.3 % bei p-Wert von 0.003)

Konfigurationen für neue Aufgabenstellungen nicht a-priori bestimmbar. Aus diesem Grund ist eine Adaption der Hyperparameter-Konfigurationen durch Evaluation und Optimierung innerhalb einer neuen Lernumgebung notwendig.

#### 3.4.2 Automatisierte Hyperparameter-Optimierung

Die Definition der Hyperparameter stellt somit ein wesentliches Element bei der Konfiguration der auf Deep Reinforcement Learning basierenden Systeme dar. Da Hyperparameter-Konfigurationen ähnlicher Aufgabenstellungen korrelieren, können erfahrene Experten auf geeignete Hyperparameter-Kandidaten schließen, doch selbst Experten müssen mittels Versuch-und-Irrtum eine aufgabenspezifische Optimierung durchführen (ANAND et al. 2020). Die notwendige Expertise und der zeitliche Aufwand behindert den industriellen Einsatz, weshalb sich automatisiertes maschinelles Lernen (AutoML) mit der Automatisierung dieses Prozesses beschäftigt (F. HUTTER et al. 2019). Da die Trainingsphase eines Agenten bei komplexen Aufgabenstellungen auch in einer simulativen Lernumgebung mehrere Tage dauern kann (SHACKLETT et al. 2021), ist in Deep Reinforcement Learning eine effiziente Suche geeigneter Hyperparameter-Konfigurationen von Bedeutung.

Die Suche nach geeigneten Hyperparameter-Konfigurationen kann als ein Blackbox-Optimierungsproblem betrachtet werden. Im Bereich von Deep Learning hat sich der Einsatz der *Bayesschen Optimierung* in der Suche nach geeigneten Netz-Architekturen (z. B. Anzahl der Perzeptronen pro Schicht) sowie weiterer Hyperparameter wie der Lernrate als effizient erwiesen (BERGSTRA et al. 2011, SNOEK et al. 2012).

### 3.4 Automatisiertes maschinelles Lernen in Deep Reinforcement Learning

Die Bayessche Optimierung hat das Ziel, den Lösungsraum mit möglichst wenigen Versuchen zu durchsuchen, indem auf Basis vergangener Versuche ein probabilistisches Modell (z. B. ein Gauß-Prozess) für den unbekanntem Zusammenhang zwischen den Hyperparametern und dem Wert der Zielfunktion generiert wird (siehe Abbildung 3.8). Hieraus lässt sich ableiten, in welchen Bereichen die höchste Wahrscheinlichkeit zur Verbesserung vorliegt, welche über die Akquisitionsfunktion approximiert wird.

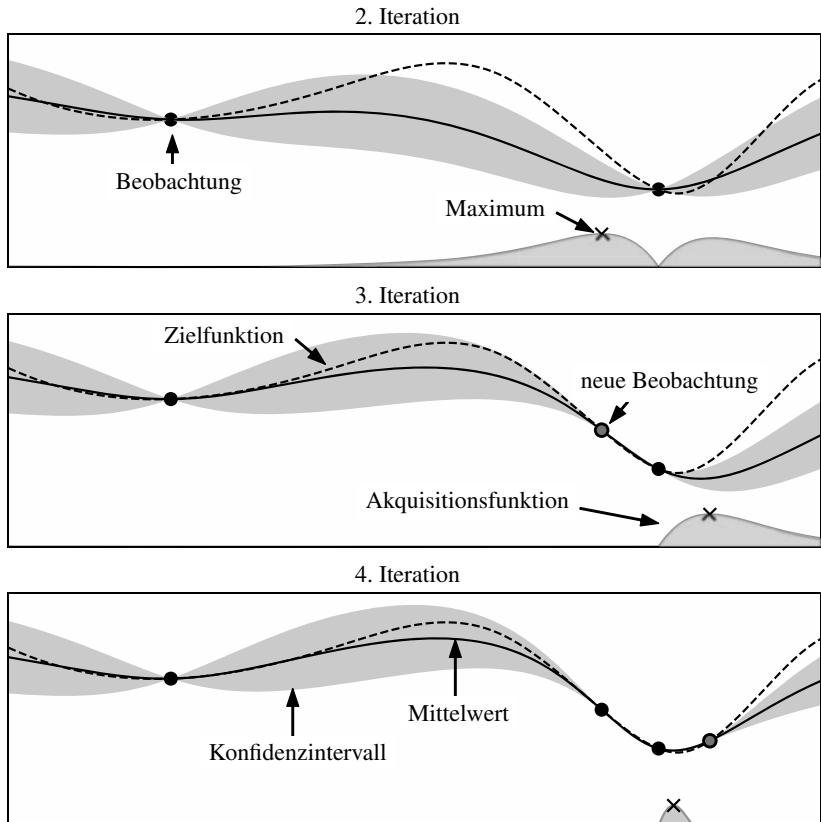


Abbildung 3.8: Ansatz der Bayesschen Optimierung zur Minimierung einer unbekanntem Zielfunktion nach F. HUTTER et al. (2019)

FALKNER et al. (2018) setzen einen auf der Bayesschen Optimierung basierenden Ansatz zur Optimierung der Hyperparameter in PPO ein. Zur weiteren Steigerung der Effizienz werden mehrere Agenten gleichzeitig trainiert und die Lernprozesse nach einem Abbruchkriterium beendet (sog. Successive Halving). Es werden hierbei sieben Hyperparameter von PPO in einer simulierten Aufgabenstellung optimiert. Hyperparameter wie die Lernrate (Bereich  $10^{-7}$  bis  $10^{-1}$ ), welche mehrere Größenordnungen annehmen können, werden über eine logarithmische Funktion skaliert. Mit dem Ansatz kann innerhalb einer Minute eine Hyperparameter-Konfiguration gefunden werden, welche gegen eine Lösung konvergiert, wohingegen bei einer Zufallssuche über zehn Minuten notwendig sind.

YOUNG et al. (2020) setzen die Bayessche Optimierung beim Trainieren eines PPO Agenten in einem visuellen Computerspiel (Atari Qbert) ein. Um eine Parallelisierung der Suche zu ermöglichen, wird der Suchraum der Hyperparameter in einzelne Subräume aufgeteilt. Es werden 8 Hyperparameter (u. a. Lernrate  $\alpha$ ,  $\lambda$ , Größe der Kernel im CNN) über 20 Optimierungssiterationen optimiert, wobei eine deutliche Steigerung der Leistung des Agenten erzielt wird.

Neben den Hyperparametern des KNN und des RL-Algorithmus wird in anderen Arbeiten zudem eine automatische Anpassung der Lernumgebung wie die Parametrisierung der Belohnungsfunktion selbst als ein zusätzliches Optimierungsproblem betrachtet (FAUST et al. 2019, FRANKE et al. 2020, HU et al. 2020). Da die Leistung des Agenten somit nicht über einen Belohnungswert gemessen werden kann, wird in diesen Ansätzen eine Bilevel-Optimierung eingesetzt. Hierbei wird in einer übergeordneten Optimierungsschleife die Parametrisierung der Belohnungsfunktion in Bezug auf ein übergeordnetes Optimierungskriterium durchgeführt und in einer inneren Schleife die Optimierung der weiteren Hyperparameter zur Maximierung der Belohnung vorgenommen.

#### 3.4.3 Meta Learning

Bei der manuellen Konfiguration der Hyperparameter in Deep Learning bzw. Deep Reinforcement Learning wird üblicherweise zunächst auf Hyperparameter-Konfigurationen aus ähnlichen Aufgabenstellungen zurückgegriffen und von hier eine weitere Optimierung durchgeführt (FEURER et al. 2015, F. HUTTER et al. 2019, GRAESSER & KENG 2019). Diese Wiederverwendung der Erfahrungen aus vergangenen Aufgaben

### 3.4 Automatisiertes maschinelles Lernen in Deep Reinforcement Learning

wird im Rahmen von AutoML als *Meta Learning* bezeichnet, wobei eine Modellierung der Aufgabenstellung über sogenannte *Meta-Features* erfolgt, um auf deren Basis Hyperparameter-Konfigurationen ähnlicher Aufgaben wiederzuverwenden (F. HUTTER et al. 2019). Abbildung 3.9 zeigt den Ansatz zum Abrufen bestehender Hyperparameter-Konfigurationen auf Basis von Meta-Features.

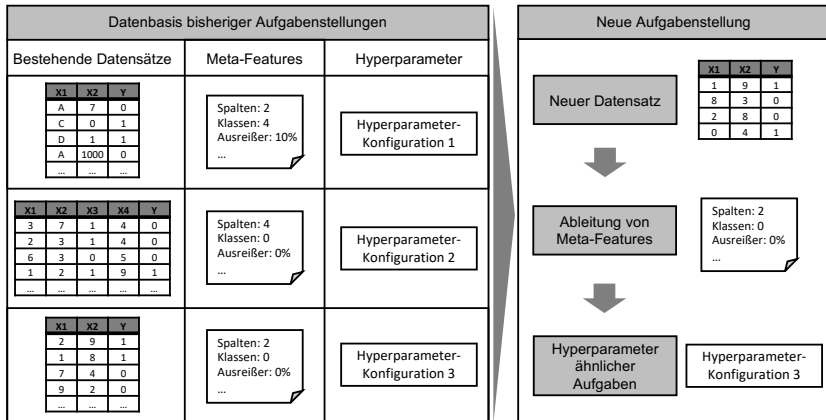


Abbildung 3.9: Ansatz des Meta Learnings beim überwachten Lernen

J. KIM et al. (2017) beschleunigen die Optimierung von KNN im Rahmen der Bilderkennung durch das Erlernen der Ähnlichkeit von Datensätzen mit Bilddaten und Wiederverwendung der Hyperparameter in der Bayesschen Optimierung. Die Ähnlichkeit von Datensätzen wird durch die erreichte Vorhersagegüte der Vorhersagemodelle bei Verwendung gleicher Hyperparameter-Konfigurationen gemessen. Diese Ähnlichkeit wird durch die  $L_1$ -Distanz der Vektoren der jeweiligen Meta-Features eines Datensatzes abgebildet, wobei die Meta-Features eines Datensatzes über ein KNN selbstständig auf Basis der bestehenden Datensätze erlernt werden. Hierdurch können bei einer neuen Aufgabenstellung Hyperparameter-Konfigurationen von ähnlichen Datensätzen zur Initialisierung der Bayesschen Optimierung wiederverwendet werden. Der Ansatz wird auf 8 Test-Datensätzen validiert, wobei bereits durch Wiederverwendung bestehender Hyperparameter-Konfigurationen Vorhersagegüten erreicht werden können, welche ohne Initialisierung 20 oder mehr Optimierungsschleifen benötigen würden.

### 3 Stand der Wissenschaft und Technik

---

FEURER et al. (2020) verwenden als Meta-Features einfache statistische Eigenschaften tabellarischer Datensätze im Rahmen der Klassifikation (Anzahl der Datenpunkte, Eingangsparameter und Klassen), um die Ähnlichkeit von Datensätzen zu quantifizieren. Aus historischen Hyperparameter-Konfigurationen wird ein Portfolio an Lösungen gebildet und diese Hyperparameter bei der Bayesschen Optimierung bei neuen Datensätzen wiederverwendet.

Die Ergebnisse des Wettbewerbs zur Hyperparameter-Optimierung der NeurIPS-Konferenz 2020 zeigen, dass durch diese Form der Initialisierung der Optimierungsansätze eine starke Verbesserung erzielt werden kann (TURNER et al. 2021). Auch im Rahmen von Deep Reinforcement Learning kann in CO-REYES et al. (2021) durch die Initialisierung der Optimierung basierend auf bestehenden Lösungen bereits eine starke Beschleunigung des Optimierungsprozesses nachgewiesen werden, allerdings erfolgt keine automatisierte Auswahl der Lösungen.

### 3.5 Zusammenfassung und Handlungsbedarf

Diese Arbeit hat das Ziel, Deep Reinforcement Learning in die aufgabenorientierte Programmierung von Montagestationen zu integrieren. Zur Bewertung der im Stand der Wissenschaft und Technik vorgestellten Ansätze werden diese hinsichtlich der folgenden Kriterien untersucht:

#### **Modellierung automatisierter bzw. autonomer Montagesysteme**

1. *Ganzheitliche Betrachtung von Montagesystemen*: Mit diesem Kriterium wird die Anwendbarkeit der Ansätze auf unterschiedliche Montageaufgaben geprüft.
2. *Aufgabenorientierte Programmierung*: Verwendung von Skills zur Vereinfachung der softwareseitigen Konfiguration von Montagesystemen.
3. *Simulation*: Anwendung einer Simulation zur Generierung von Lösungen oder zur Validierung verschiedener Lösungsansätze.
4. *Herleitung von Skills*: Skills stellen ein Kernelement der aufgabenorientierten Programmierung dar. Mit diesem Kriterium wird geprüft, ob in den betrachteten Arbeiten ein methodisches Vorgehen zur Herleitung von Skills verwendet wird.

5. *Anwendungsbereich der Skills*: Neben der Beschreibung der Fähigkeiten von Aktoren ist in autonomen Systemen eine Modellierung von Fähigkeiten hinsichtlich der Wahrnehmung und des Datenaustauschs notwendig.

#### **Automatische Konfiguration auf Deep Reinforcement Learning basierender Agenten**

6. *Reinforcement Learning*: Anwendung des betrachteten Ansatzes im Kontext von Reinforcement Learning
7. *Modellierung von Agenten*: Modellierung wiederverwendbarer Elemente im Kontext von Reinforcement Learning
8. *Effizientes AutoML ohne Expertenwissen*: Automatisiertes Maschinelles Lernen durch Abstrahierung der Aufgabenstellung über Meta-Features (Meta Learning).

Die im Stand der Wissenschaft und Technik aufgeführten Ansätze werden in Abbildung 3.10 entsprechend ihrer Zielsetzung in "Vereinfachung der Programmierung von Montagesystemen", "Herleitung von Skills in der Montageplanung" und "Automatisiertes Maschinelles Lernen" zusammengefasst.

Die Ansätze zur Vereinfachung der Programmierung und Planung von Montagesystemen unterscheiden sich hinsichtlich der betrachteten Prozesse. Teilweise werden spezifische Prozesse (KUSS 2020, LÄMMLÉ et al. 2020) oder Ressourcen wie Robotersysteme (HUA et al. 2017, TRAPANI 2019) fokussiert. Andere Arbeiten nehmen eine ganzheitliche Betrachtung von Montagesystemen vor (BACKHAUS 2016, MICHNIEWICZ 2018, HAMMERSTINGL 2020) und führen eine Validierung des Ansatzes anhand ausgewählter Prozesse und Ressourcen durch.

Während die lösungsneutrale Beschreibung von Aufgaben und Ressourcen über Skills die Grundlage der meisten betrachteten Ansätze ausmacht, beschränken sich diese Ansätze vornehmlich auf aktorische Skills der Ressourcen. In HAMMERSTINGL (2020) und GONNERMANN et al. (2020) werden zudem sensorische Skills betrachtet, welche zur Wahrnehmung der Umgebung der in dieser Arbeit untersuchten Systeme ebenfalls behandelt werden müssen.

### 3 Stand der Wissenschaft und Technik

	Vereinfachung der Programmierung von Montagesystemen										Herleitung von Skills in der Montageplanung			Automatisiertes Maschinelles Lernen							
<ul style="list-style-type: none"> <li>— Keine Bewertung möglich</li> <li>○ Kriterium nicht erfüllt</li> <li>◐ Kriterium kaum erfüllt</li> <li>◑ Kriterium teilweise erfüllt</li> <li>● Kriterium weitestgehend erfüllt</li> <li>● Kriterium erfüllt</li> </ul>	BACKHAUS 2016	HUA ET AL. 2016	HALT ET AL. 2018	TRAPANI 2019	BERG 2020	KUSS 2020	LÄMMLE ET AL. 2020	NOTTENSTEINER ET AL. 2021	SMALE 2011	MICHNIEWICZ 2018	HAMMERSTINGL 2020	JÄRVENPÄÄ ET AL. 2019	GONNEMANN ET AL. 2020	FALKNER ET AL. 2018	YOUNG ET AL. 2020	FAUST ET AL. 2019	FRANKE ET AL. 2020	HU ET AL. 2020	CO-REYES ET AL. 2021	KIM ET AL. 2017	FEUER ET AL. 2020
Ganzheitliche Betrachtung von Montagesystemen	●	○	◐	○	○	◐	◐	◐	○	●	●	●	○	—	—	—	—	—	—	—	—
Aufgabenorientierte Programmierung	●	◐	◐	●	●	●	◐	◐	○	●	●	○	○	—	—	—	—	—	—	—	—
Simulation	●	○	○	●	◐	●	●	●	○	●	●	○	○	●	●	●	●	●	○	○	○
Herleitung von Skills	●	○	●	◐	●	○	○	◐	●	●	●	●	●	—	—	—	—	—	—	—	—
Anwendungsbereich der Skills	●	○	◐	○	◐	◐	◐	◐	◐	●	●	◐	●	—	—	—	—	—	—	—	—
Reinforcement Learning	○	○	○	○	○	○	●	○	○	○	○	○	○	●	●	●	●	○	○	○	○
Modellierung von Agenten	○	○	○	○	○	○	◐	○	○	○	○	○	○	◐	◐	◐	◐	○	○	○	○
Effizientes AutoML ohne Expertenwissen	—	—	—	—	—	—	—	—	—	—	—	—	—	◐	◐	◐	◐	●	●	●	●

Abbildung 3.10: Bewertung der Literatur hinsichtlich ihrer Anwendbarkeit zur aufgabenorientierten Konfiguration autonomer Montagesysteme

LÄMMLE et al. (2020) verwenden einen Skill-basierten Ansatz zur Konfiguration eines auf Deep Reinforcement Learning basierenden Systems zum kraftgeregelten Zusammensetzen von Bauteilen. Der Ansatz beschränkt sich auf den betrachteten Prozess und es wird keine Herleitung der Skills durchgeführt.

In den bisherigen Arbeiten lässt sich somit noch keine allgemeine Modellierung der zur Konfiguration von auf Deep Reinforcement Learning basierenden Systemen notwendigen Elemente finden. Des weiteren existiert keine Integration von Deep Reinforcement Learning in die aufgabenorientierte Programmierung und Planung von Montagesystemen.

Automatisiertes Maschinelles Lernen lässt sich als das Pendant zur automatischen Programmengenerierung in der aufgabenorientierten Programmierung betrachten. Im Gegensatz zur Erstellung des Codes, welches zur Ausführung der Aufgabe auf entsprechende



Ressourcen übertragen wird, erfolgt die Parametrisierung und Implementierung des Codes, welcher den Trainingsprozess des lernenden Systems ermöglicht. Somit erfolgt die Programmgenerierung, welche durch die Strategie  $\pi(a|s)$  des Agenten abgebildet wird, durch das lernende System selbst.

Diverse Arbeiten haben sich mit automatisiertem maschinellen Lernen im Rahmen von Deep Reinforcement Learning beschäftigt, mit dem Fokus, eine effiziente Optimierung der Hyperparameter in einem vordefinierten Raum vorzunehmen. Neben der Erreichung einer größtmöglichen Leistung des Systems, welche in Reinforcement Learning durch die Belohnung abgebildet wird, liegt der Fokus auf der effizienten Durchführung des Optimierungsprozesses, wobei die Effizienz über die Anzahl an Iterationen, der Nutzung von Rechenkapazitäten oder der Dauer der Optimierungsprozesses gemessen wird (FALKNER et al. 2018, YOUNG et al. 2020, FRANKE et al. 2020).

Diese Ansätze benötigen einen hohen zeitlichen Aufwand und entsprechende Rechenressourcen, welche nur durch die manuelle Einschränkung des Lösungsraums durch einen Experten reduziert werden können. Meta Learning stellt einen Ansatz zur effizienten Optimierung dar, wobei der Optimierungsprozess von bestehenden ähnlichen Lösungen aus gestartet wird und somit kein Expertenwissen benötigt wird (J. KIM et al. 2017, FEURER et al. 2020). Die Ähnlichkeit der Aufgabenstellungen wird hier durch Meta-Features der jeweiligen Datensätze im Bereich des überwachten Lernens ermittelt.

Wie in Abbildung 3.10 dargestellt, wird die Abstraktion der Aufgabenstellung zum Einsatz von Meta-Learning in RL bisher noch nicht eingesetzt. Die semantische Beschreibung der Aufgabenstellung, wie sie in bisherigen Arbeiten zur aufgabenorientierten Programmierung eingesetzt wird, bietet die Möglichkeit, den Ansatz des Meta-Learnings auf Reinforcement Learning zu übertragen.



## **4 Einschränkung des Betrachtungsbereichs und Lösungsansatz**

Zur Eingrenzung des Betrachtungsbereichs und zur Ableitung eines Lösungsansatzes erfolgt zunächst eine Analyse der Rahmenbedingungen und Anforderungen seitens der Industrie sowie eine Analyse des Vorgehens in der Umsetzung auf RL basierender Systeme in Abschnitt 4.1. Durch den Vergleich der verfügbaren und notwendigen Kompetenzen erfolgt anschließend eine Eingrenzung des Betrachtungsbereichs und eine Ableitung der notwendigen Funktionalitäten des Systems in Abschnitt 4.2. Abschnitt 4.3 gibt einen Überblick über den Lösungsansatz dieser Arbeit, um die benötigten Funktionalitäten in einem System umzusetzen.

### **4.1 Analyse der Rahmenbedingungen zur Umsetzung intelligenter Systeme in der Industrie**

#### **4.1.1 Studie zum Kompetenzbedarf in produzierenden Unternehmen**

Zur Analyse des derzeitigen Stands in der Einführung intelligenter Systeme wurden strukturierte Interviews (siehe Anhang A.1) mit einzelnen Digitalisierungsbeauftragten von 22 produzierenden Unternehmen aus Süddeutschland durchgeführt (RÖHLER 2020). Die befragten Unternehmen setzen sich zu gleichen Teilen aus kleinen und mittelständischen Unternehmen (Mitarbeiterzahl < 500) sowie Großunternehmen zusammen. Aufgrund der weitgreifenden Definition intelligenter Systeme bzw. Künstlicher Intelligenz, wurden in der Studie Systeme betrachtet, welche auf maschinellen Lernverfahren basieren.

Bei 55 % der Unternehmen ist bereits eine Umsetzung auf ML basierender Systeme, wie eine automatische optische Qualitätssicherung oder eine vorausschauende Wartung, erfolgt oder eine Umsetzung wird zum Zeitpunkt der Befragung bereits geplant oder durchgeführt. Als Motivation zur Einführung intelligenter Systeme wurden Qualitäts-

## 4 Einschränkung des Betrachtungsbereichs und Lösungsansatz

---

steigerung (68 %), Erhöhung des Automationsgrads (55 %), Effizienzsteigerung (41 %) und das Anbieten neuer Produkte und Dienstleistungen (41 %) angegeben.

Für diese Arbeit sind die vorhandenen Kompetenzen in der Umsetzung und dem Betrieb von ML-Systemen in produzierenden Unternehmen relevant, welche die Unternehmen intern aufbauen wollen oder sie externe Unterstützung durch Dienstleister oder Softwarewerkzeuge heranziehen. In der Studie werden die folgenden Kompetenzen betrachtet:

- *ML Expertise*: Umsetzung spezifischer ML-Komponenten basierend auf Methoden des maschinellen Lernens.
- *Datenarchitektur*: Implementierung der IT-Infrastruktur für die Datenspeicherung und Bereitstellung der Daten.
- *MLOps*: Kombination aus den Begriffen DevOps (Development Operations) und maschinelles Lernen. Verantwortlich für die Überwachung eines ML-Systems während des Betriebs.
- *Projektmanagement*: Organisation und Planung eines ML-Projekts.
- *Strategie*: Identifikation relevanter Anwendungsfälle.

Bei Betrachtung der gesuchten externen Unterstützung zeigt sich eine deutliche Diskrepanz zwischen KMU und Großunternehmen (siehe Abbildung 4.1). In Großunternehmen wird ein interner Aufbau zur Schaffung der notwendigen Strukturen für den eines maschinellen Lernverfahrens angestrebt, allerdings wird hier gerade in neuen ML-Ansätzen und -Anwendungsfällen vornehmlich nach externer Expertise gesucht. KMU besitzen dagegen nicht die Ressourcen, um die notwendigen Kompetenzen intern aufzubauen und suchen daher nach vollumfänglichen Lösungen. Unabhängig der Unternehmensgrößen kann daher mangelnde RL-Expertise als ein wesentliches Hindernis in der weiten Nutzung des Ansatzes gesehen werden.

Die Unternehmen wurden zudem gefragt, welche Kriterien externe Partner erfüllen müssen, um gemeinsam intelligente Systeme in die Unternehmen einzuführen. Die drei wesentlichen Kriterien sind:

*Umsetzungsfähigkeit*: 73 % der befragten Unternehmen geben an, dass die Umsetzungsfähigkeit ein zentrales Kriterium darstellt. Diese Fähigkeit setzt einerseits die Umsetzung ähnlicher Anwendungsfälle des externen Partners voraus, andererseits muss

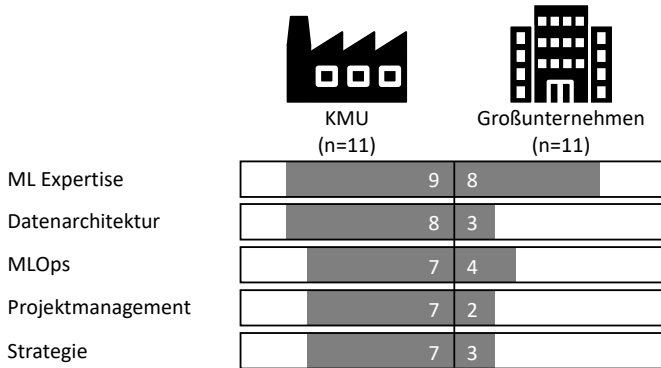


Abbildung 4.1: Anzahl der Unternehmen, welche eine externe Unterstützung hinsichtlich einzelner Kompetenzen ohne internen Kompetenzaufbau anstreben, nach RÖHLER (2020)

auch eine Integration des Anwendungsfalls in die bestehende IT-Infrastruktur (z. B. Steuerungssysteme, Cloud-Umgebung) möglich sein.

*Domänenwissen:* 36 % der befragten Unternehmen geben an, dass eine Nähe des Domänenwissens der externen Partner zu den eigenen Geschäftsprozessen gegeben sein sollte, um eine effiziente Adaption von ML-Anwendungen zu ermöglichen. Dieses Kriterium überschneidet sich teilweise mit dem Kriterium der Umsetzungsfähigkeit, da auch hier die Ähnlichkeit bestehender Anwendungsfälle zu den Fertigungs- und Montageprozessen der produzierenden Unternehmen zur Erfüllung des Kriteriums beiträgt.

*ML-Expertise:* 27 % der befragten Unternehmen geben an, dass ein tiefes theoretisches Wissen zu ML-Methoden von externen Partnern erwartet wird. So sollen diese auch neueste algorithmische Entwicklungen umsetzen können, allerdings steht dies weniger im Vordergrund als die zuverlässige Umsetzung der Anwendung.

#### 4.1.2 Vorgehen in der Umsetzung auf RL basierender Systeme

Seit Reinforcement Learning an Relevanz in Forschung und Industrie gewonnen hat, konnten mittlerweile Erfahrungen in der Umsetzung hierauf basierender Systeme gesammelt werden, welche zu Beginn der Arbeiten im Bereich von Deep Reinforcement

## 4 Einschränkung des Betrachtungsbereichs und Lösungsansatz

---

Learning in 2015 noch nicht vorhanden waren (OBANDO-CERÓN & CASTRO 2021, WINDER 2020).

In diesem Abschnitt werden daher die bisherigen Erkenntnisse zum Vorgehen in der Umsetzung auf RL basierender Systeme analysiert und die vom Anwendungsfall unabhängigen Projektphasen konsolidiert.

Im Forschungsprojekt InPuls wurde ein Vorgehen zur Umsetzung in der Produktionstechnik entwickelt (PABICH et al. 2019). In ENNEN et al. (2019) erfolgt eine detaillierte Darstellung des entwickelten Vorgehens, welches in die *Planungsphase* und die *Realisierungsphase* aufgeteilt ist. Im Folgenden werden die von ENNEN et al. (2019) beschriebenen Schritte mit weiteren Quellen verglichen.

Sowohl ENNEN et al. (2019) als auch WINDER (2020) und DONG et al. (2020) stellen zunächst die *Identifikation von Prozessen*, welche einen Einsatz von Reinforcement Learning erlauben, als ersten Schritt zur Einführung des Ansatzes in einem Unternehmen dar. Neben der Evaluation alternativer Ansätze muss hier auch die Modellierbarkeit der Umgebung betrachtet werden, um eine simulationsbasierte Umsetzung und Evaluation des Agenten zu ermöglichen (WINDER 2020).

Anschließend erfolgt eine *Analyse des Prozesses*. Die Spezifikation der Aufgabenstellung und der verwendeten Ressourcen wie Aktoren und Sensoren bildet die Basis für die folgenden Schritte, wie der Abbildung der Aufgabenstellung über eine Belohnungsfunktion (SHAO et al. 2018, ENNEN et al. 2019).

SHAO et al. (2018) und WINDER (2020) stellen der Realisierungsphase zudem eine *iterative Umsetzung der Lernumgebung und des Agenten* in einer Simulation voran, welche in ENNEN et al. (2019) nicht näher betrachtet wird. In dieser Phase erfolgt zunächst eine Implementierung der simulativen Lernumgebung und des Agenten mit entsprechenden Hyperparametern. Die Lernumgebung (u. a. Wahrnehmung und Aktionsraum) und die Hyperparameter des Agenten werden iterativ angepasst, bis das gewünschte Systemverhalten in der Simulation erreicht wird, und erst anschließend erfolgt eine Umsetzung des realen Systems.

Die durch die Lernumgebung abgebildete Aufgabenstellung kann in dieser Phase zudem beginnend vom einfachsten Fall schrittweise hinsichtlich ihrer Komplexität gesteigert werden (DONG et al. 2020, WINDER 2020). Neben statistischen Größen wie den erreichten Belohnungswerten und Erfolgsquoten kann eine Visualisierung des Systemverhaltens zur Bewertung des Agentenverhaltens genutzt werden (WINDER 2020).

## 4.1 Analyse der Rahmenbedingungen zur Umsetzung intelligenter Systeme in der Industrie

Die Realisierung des Systems an einer realen Anlage kann über Sim-to-real Ansätze (vgl. Abschnitt 3.3.2) erfolgen, wobei ein *Trainieren des Agenten in einer realitätsnahen Simulation* erfolgt. Ein anderer Ansatz besteht im *Trainieren des Agenten in einer realen Umgebung*, wobei ein effizientes Trainieren des Agenten durch Demonstrationen der Bewegungsabläufe durch einen Menschen in der realen Umgebung unterstützt werden kann (VECERIK et al. 2017, HESTER et al. 2017, Z. ZHU et al. 2020). Derzeit sind diese als konkurrierende Ansätze anzusehen (WIGGERS 2021).

Weiterhin muss eine *Implementierung* der Schnittstellen zwischen den Ressourcen des Montagesystems und der Steuerungshardware, auf welcher der Agent softwareseitig implementiert ist, erfolgen, wobei zusätzlich Softwaretechniker und Steuerungstechniker hinzugezogen werden (ENNEN et al. 2019).

In Abbildung 4.2 ist die Zugehörigkeit der Planungsgegenstände, welche RL-Expertise benötigen, zu den beschriebenen Phasen abgebildet.

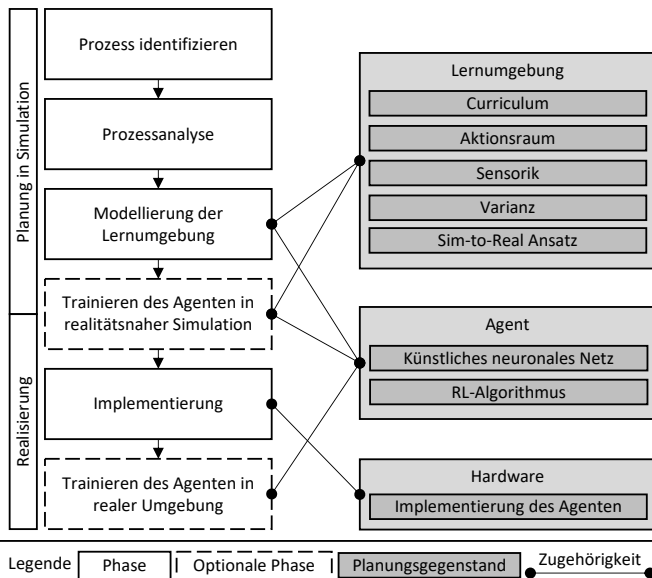


Abbildung 4.2: Projektphasen und Prozesse in der Umsetzung eines auf RL basierenden Systems

## 4 Einschränkung des Betrachtungsbereichs und Lösungsansatz

---

Aufgrund des iterativen Umsetzungsprozesses müssen die Prozesse über die Projektphasen hinweg teilweise mehrmals durchgeführt werden. Im Folgenden werden die einzelnen Planungsgegenstände näher erläutert:

- *Curriculum*: Definition der Aufgabe des Agenten durch Bewertung des Agentenverhaltens über eine Belohnungsfunktion. Zur Definition des Curriculums bzw. Lehrplans sind zudem die Definition der maximalen Anzahl an Schritten für einen Versuch der Aufgabendurchführung (sog. Episodenlänge) und die Gesamtzahl der durchzuführenden Trainingsschritte notwendig.
- *Aktionsraum*: Aktionssignale in Form von Steuerungssignalen der Aktorik, welche durch den Agenten genutzt werden.
- *Wahrnehmung*: Sensorsignale, über welche der Agent den Zustand der Umgebung wahrnimmt.
- *Varianz*: Variierende Merkmale der Umgebung und deren Wertebereiche innerhalb der Simulation. Zur Vereinfachung der Aufgabe kann die Varianz in der Planungsphase schrittweise gesteigert werden.
- *Sim-to-real Ansatz*: Implementierung von Methoden innerhalb der Simulation, welche eine direkte Übertragung des trainierten Agenten auf reale Systeme ermöglichen.
- *Künstliches neuronales Netz*: Hyperparameter zur Definition der Architektur des künstlichen neuronalen Netzes.
- *RL-Algorithmus*: Hyperparameter des RL-Algorithmus.
- *Implementierung*: Umsetzung des Agenten und dessen Schnittstellen auf der Hardware.

### 4.1.3 Nutzergruppen von Programmiersystemen in der Montage

Ein wesentliches Element in der Gestaltung von Programmiersystemen stellt die Betrachtung der Nutzergruppen dar (EHRMANN 2007). So wird auch bei der Konzeptionierung aufgabenorientierter Programmiersysteme auf Nutzerrollen zurückgegriffen, um die notwendigen Funktionalitäten des Programmiersystems abzuleiten. Neben den Nutzern erfolgt zudem eine Interaktion von Experten mit dem Programmiersystem, welche dieses um neue Funktionalitäten erweitern und Fehler beseitigen (BACKHAUS 2016). In dieser Arbeit werden RL-Experten als solche Experten betrachtet, welche das System



erweitern, aber nicht als Nutzer des Systems betrachtet werden. Als Nutzergruppen werden in dieser Arbeit die in EHRMANN (2007) und BACKHAUS (2016) analysierten Gruppen *Bediener*, *Anwender* und *Prozessexperte* betrachtet.

BACKHAUS (2016) erweitert die von EHRMANN (2007) definierten Nutzergruppen um deren Kompetenz hinsichtlich der Erfahrung im Umgang mit Simulationen, welche einen wesentlichen Bestandteil der aufgabenorientierten Programmierung und auch in der Umsetzung von RL ausmachen. Bei der Konzeptionierung aufgabenorientierter Programmiersysteme für neue Technologien, zu denen noch keine weitreichenden Kenntnisse in der Industrie vorhanden sind, muss zudem die hierbei notwendige Kompetenz in den jeweiligen Nutzergruppen betrachtet werden (BERG 2020). Im in dieser Arbeit angestrebten System ist an dieser Stelle die RL-Expertise von Relevanz. Demnach werden für die Analyse der in der Industrie möglichen Kompetenzen die folgenden Analyse Kriterien betrachtet:

- *Funktionswissen*: Wissen über Aufbau und Funktionsweise einzelner Steuerungen und Geräte eines Montagesystems.
- *Prozesswissen*: Wissen über den Ablauf und die Realisierung von Montageprozessen.
- *Erfahrung Simulationssysteme*: Erfahrung in der Anwendung von Simulationssystemen zur Planung und Validierung von Montagesystemen bzw. -prozessen.
- *Programmierkenntnisse*: Kenntnisse über steuerungsspezifische Programmiersprachen und Hochsprachen, welche u. a. aus Schulungen und dem täglichen Umgang resultieren.
- *RL-Expertise*: Expertenwissen in der Theorie und Umsetzung von RL-Algorithmen sowie von hierauf aufbauenden Anwendungsfällen.

Der Bediener beschäftigt sich im täglichen Betrieb mit der Überwachung und Aufrechterhaltung der Prozesse und verfügt kaum über Programmiererfahrung (EHRMANN 2007). Bei ihm wird von keiner Erfahrung in Simulation und RL ausgegangen.

Anwender führen die Bedienung und Programmierung von Montagesystemen durch und verfügen über ein breites Gerätewissen, allerdings nur über grundlegende Programmierkenntnisse (EHRMANN 2007). Sie lassen sich vornehmlich in KMU finden und verfügen zudem über grundlegendes Wissen in der Anwendung von Simulationssystemen. Umfangreiche Schulungen sind für sie oftmals nicht möglich, weshalb hier von keiner Weiterbildung im Bereich RL ausgegangen wird.

## 4 Einschränkung des Betrachtungsbereichs und Lösungsansatz

Der Prozessexperte verfügt über umfassende Kenntnisse zu Montageprozessen und Simulationssystemen und ist ebenfalls in KMU anzutreffen (BACKHAUS 2016). BACKHAUS (2016) geht hier im Gegensatz zu EHRMANN (2007) von keinen tiefergehenden Programmierkenntnissen aus. Bei Systemintegratoren und größeren Unternehmen kann allerdings vom Vorhandensein von *Automatisierungsexperten* ausgegangen werden, welche über umfassendes Funktionswissen und tiefergehende Programmierkenntnisse verfügen. In Hinblick auf die Ergebnisse der Studie (Abschnitt 4.1.1) werden sowohl beim Prozessexperten als auch beim Automatisierungsexperten keine oder kaum Erfahrung im Bereich RL angenommen.

Kenntnisbereich \ Nutzergruppe	Funktionswissen	Prozesswissen	Erfahrung Simulationssysteme	Programmierkenntnisse	RL-Expertise
Bediener	☉	☉	○	☉	○
Anwender	☉	☉	☉	☉	○
Prozessexperte	☉	●	☉	☉	☉
Automatisierungsexperte	●	☉	☉	☉	☉

○ kaum vorhanden, ☉ gering, ☉ durchschnittlich, ● umfangreich, ● vollständig vorhanden

Abbildung 4.3: Betrachtete Nutzergruppen eines Systems zur aufgabenorientierten Konfiguration intelligenter Agenten

## 4.2 Einschränkung des Betrachtungsbereichs

### 4.2.1 Handlungsbedarf in der Industrie

Durch die in dieser Arbeit konzeptionierte aufgabenorientierte Konfiguration intelligenter Agenten soll die notwendige Kompetenz zur Umsetzung auf RL basierender Montagesysteme reduziert werden. Zur Analyse der notwendigen Funktionalitäten wird der Ansatz der Design Structure Matrix (DSM) verwendet, in welcher die Einflüsse verschiedener Elemente abgebildet werden (LINDEMANN 2005). Eine Multiple Domain Matrix (MDM) ermöglicht als Erweiterung der DSM zudem eine Analyse der Abhängigkeiten einzelner Elemente verschiedener Domänen (KREIMEYER & LINDEMANN 2011).

## 4.2 Einschränkung des Betrachtungsbereichs

Zur Analyse der Anwendbarkeit des Systems für die in Abschnitt 4.1.3 definierten Nutzergruppen erfolgt zunächst eine Analyse der wechselseitigen Abhängigkeit der in Abschnitt 4.1.2 definierten Planungsgegenstände in der Umsetzung auf RL basierender Systeme. Da die Architektur des KNN und die Hyperparameter des RL-Algorithmus gemeinsam definiert werden, werden diese als "Agent" zusammengefasst.

Abbildung 4.4 zeigt die Abhängigkeiten dieser Planungsgegenstände unterteilt in "beeinflusst" (z. B. beeinflusst die Wahrnehmung die Architektur des künstlichen neuronalen Netzes des Agenten), "definiert" (z. B. definiert der Aktionsraum die notwendigen Schnittstellen in der Implementierung) und "validiert" (z. B. validiert das umgesetzte Agentenverhalten den Aktionsraum und die Wahrnehmung). Die Aktivsumme ergibt sich als Summe der Einflüsse eines Planungsgegenstandes auf andere Planungsgegenstände und die Passivsumme ergibt sich durch die Abhängigkeit von anderen Planungsgegenständen. Aufgrund der hohen Aktiv- und Passivsumme ist die Konfiguration des Agenten als kritischster Planungsgegenstand anzusehen.

	Aktionsraum	Wahrnehmung	Varianz	Curriculum	Agent	Sim-to-real	Implementierung	Aktivsumme
Aktionsraum				beeinflusst	beeinflusst	beeinflusst	definiert	4
Wahrnehmung				beeinflusst	beeinflusst	beeinflusst	definiert	4
Varianz	beeinflusst	beeinflusst		beeinflusst	beeinflusst	beeinflusst		5
Curriculum					beeinflusst			1
Agent	validiert	validiert	validiert	validiert		validiert	beeinflusst	6
Sim-to-real				beeinflusst	beeinflusst			2
Implementierung					validiert	validiert		2
Passivsumme	2	2	1	5	6	5	3	

Abbildung 4.4: Abhängigkeiten der Prozesse

Als weitere Domänen werden die Phasen *Planung*, *Realisierung* und *Rekonfiguration* entlang des Lebenszyklus eines Montagesystems und die in Abschnitt 4.1.3 definierten Kompetenzen in die in Abbildung 4.5 dargestellte MDM aufgenommen. In der Planungsphase erfolgen die Prozessanalyse und die Umsetzung des Agenten zur Validierung der hardwareseitigen Konfiguration des Systems und der Umsetzbarkeit des intelligenten

#### 4 Einschränkung des Betrachtungsbereichs und Lösungsansatz

Montagesystems. In der Realisierung erfolgt die Implementierung des Agenten auf der Hardware und das Training des Agenten in der realen Umgebung bzw. auf Basis eines Sim-to-real-Ansatzes. Zudem wird die Rekonfiguration des Montagesystems in der Betriebsphase betrachtet. In der Industrie existieren Anwendungsfälle wie das Greifen von Objekten, in welchen die Varianz der Geometrien in der initialen Umsetzung des Agenten groß genug ist, sodass auch bei neuen Objekten kein erneutes Training notwendig ist (RÖHLER et al. 2021). Für den Sondermaschinenbau und die spezifischen Montageprozesse von KMU muss davon ausgegangen werden, dass Varianz, welche in der Umsetzung des Montagesystems nicht berücksichtigt wurde, nicht vom Agenten gehandhabt werden kann. Bei Änderung der Varianz (z. B. durch neue Produktvarianten) muss daher ein erneutes Trainieren des Agenten sowie eine Evaluation des Agentenverhaltens erfolgen. Da die Schnittstellen und die übergeordnete Steuerungsarchitektur bereits vorliegt, muss keine erneute Implementierung vorgenommen werden.

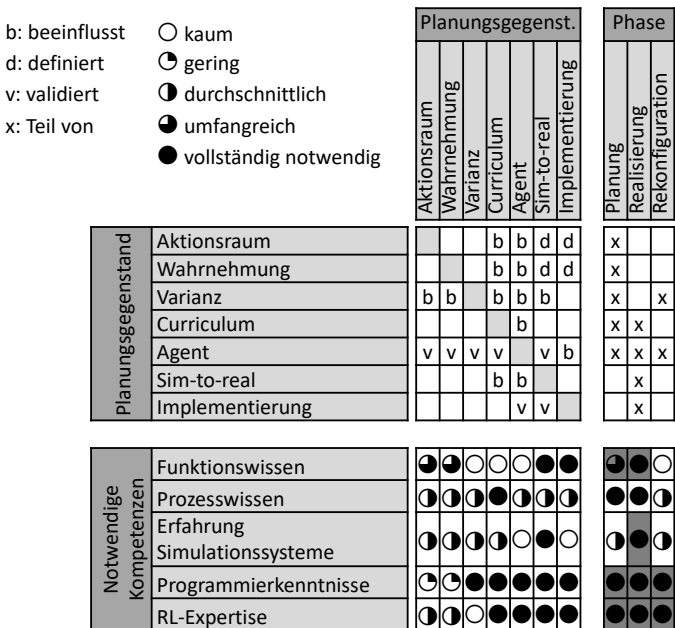


Abbildung 4.5: Ist-Situation der notwendigen Kompetenzen entlang des Lebenszyklus. Die schattierten Felder zeigen den Handlungsbedarf entlang des Lebenszyklus bei KMU (vgl. Abbildung 4.3).

Die notwendigen Kompetenzen für die einzelnen Planungsgegenstände ergeben sich durch die Betrachtung des Planungsgegenstands selbst sowie dessen Abhängigkeit von anderen Planungsgegenständen. So erfordert die Auswahl der Aktorik und Sensorik entsprechendes Funktions- und Prozesswissen, allerdings sind bei der Auswahl eines geeigneten Aktionsraums (z. B. Steuerung des Achsmoments oder der Achsgeschwindigkeiten) und der Definition der Wahrnehmung (z. B. Art der Bildvorverarbeitung wie die Reduktion der Auflösung) Erfahrungswissen in der Umsetzung von RL notwendig.

Zur weiteren Analyse der notwendigen Kompetenzen werden die minimal notwendigen Kompetenzen durch Betrachtung der in den einzelnen Phasen zu definierenden Planungsgegenstände analysiert. In allen Phasen des Lebenszyklus sind Kompetenzen erforderlich, welche durch die derzeitigen Nutzergruppen in KMU nicht abgebildet werden können. Doch auch in größeren Unternehmen, welche Automatisierungsexperten beschäftigen, ist nicht die erforderliche RL-Expertise vorhanden.

### 4.2.2 Notwendige Funktionalitäten der aufgabenorientierten Konfiguration intelligenter Agenten

Um die notwendigen Kompetenzen mittels einer aufgabenorientierten Konfiguration intelligenter Agenten zu reduzieren, ist neben der Automatisierung der Konfiguration des Agenten daher auch eine wissensbasierte Entscheidungsunterstützung zur Definition weiterer Planungsgegenstände sowie die Modellierung dieser Planungsgegenstände im Informationsmodell notwendig.

Da die Definition der Montageprozesse unabhängig von den verwendeten Steuerungsansätzen ist, muss eine Unterstützung des Nutzers erst bei der *Ableitung der zur Umsetzung der Prozesse notwendigen Skill-Sequenzen* erfolgen. Skills, welche unabhängig von variierenden Umgebungsbedingungen sind, lassen sich über bestehende Ansätze der aufgabenorientierten Programmierung umsetzen. Auf dieser Ebene muss daher bereits eine *Zuordnung von Agentenskills* erfolgen, welche im Ansatz dieser Arbeit fokussiert wird.

Zum Erlernen eines Agentenskills ist zunächst die *Ableitung der Lernumgebung* notwendig, welche den MDP-Prozess abbildet. Es existieren Ansätze zur Automatisierung der Konfiguration des Curriculum, allerdings werden bisher nur einzelne Aspekte wie die Parametrisierung der Belohnungsfunktion betrachtet (PORTELAS et al. 2020).

## 4 Einschränkung des Betrachtungsbereichs und Lösungsansatz

Durch eine geeignete *Unterstützung des Nutzers hinsichtlich des Curriculums* kann das notwendige RL-Expertenwissen reduziert werden. Auch ist eine *Unterstützung des Nutzers hinsichtlich des Aktionsraums und der Wahrnehmung* notwendig.

Die aufgabenspezifische Varianz und die damit verbundene Implementierung einer Lernumgebung mit entsprechender Variation der Startzustände stellen hinsichtlich des Programmieraufwands eine weitere Hürde dar. Eine geeignete *Modellierung der Varianz* soll eine automatisierte Implementierung innerhalb des Initialisierungsprozesses der Simulation ermöglichen.

Zur automatischen Konfiguration des Agenten existieren ausgereifte Ansätze, auf welchen in dieser Arbeit aufgebaut werden kann. Wie im vergangenen Abschnitt gezeigt (vgl. Abbildung 4.4), stellt dies zudem den kritischsten Planungsgegenstand dar. In dieser Arbeit soll daher die *Auswahl der Hyperparameter des Agenten automatisiert* werden. Zudem muss eine für den Nutzer interpretierbare *Evaluation des Agentenverhaltens* erfolgen, um die Validierung weiterer Planungsgegenstände zu ermöglichen.

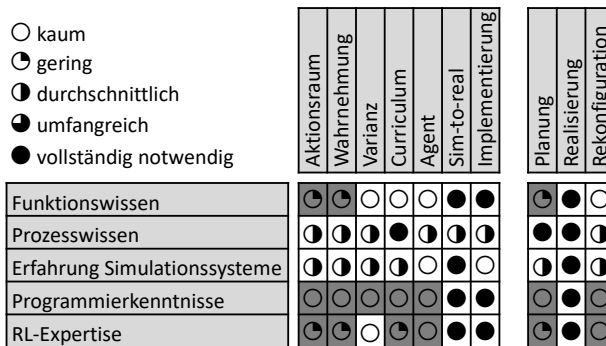


Abbildung 4.6: Angestrebte Reduzierung der notwendigen Kompetenzen in den einzelnen Phasen. Durch die Änderungen der Kompezentbedarfe (schattierte Felder) werden eine Planung durch Prozessexperten und eine Rekonfiguration durch Nutzer ermöglicht.

Wie in Abbildung 4.6 dargestellt, ist über die abgeleiteten Funktionalitäten eine eigenständige Planung des Montagesystems durch einen Prozessexperten möglich. Die Funktionalitäten ermöglichen zudem eine Rekonfiguration durch Nutzer, da in dieser Phase lediglich eine Anpassung der Varianz innerhalb der Simulation notwendig ist.

Allerdings erfordert eine vollumfängliche Automatisierung innerhalb der Realisierungsphase sowohl die Implementierung des Agenten auf der Hardware als auch die Planung weiterer Aspekte wie Sicherheitsmaßnahmen, welche in der Literatur noch nicht betrachtet worden sind. Wie in Abschnitt 3.3.2 beschrieben, stellen zudem Sim-to-real-Ansätze wie Domain Adaptation und Domain Randomization noch eigene Forschungsschwerpunkte dar. Aufgrund des geringen Reifegrades wird eine Automatisierung dieser Ansätze in dieser Arbeit nicht betrachtet. Der Fokus wird daher auf Anwendungsfälle gelegt, welche eine realitätsnahe Simulation des Montageprozesses ermöglichen.

### 4.2.3 Betrachtete Montagefunktionen

Eine wesentliche Beschränkung stellt die Modellierung und Simulation der Montageprozesse dar. Mit einer physikbasierten Simulation kann das dynamische Verhalten von Festkörpern und mechatronischen Systemen effizient simuliert werden. Derzeit verfügbare Physik-Engines wie PhysX, Bullet, ODE (Open Dynamics Engine) und MuJoCo (Multi-Joint dynamics with Contact) ermöglichen zudem die Simulation von Sensoren wie Tastsensoren oder Tiefenbild-Kameras.

Teilweise wird die physikbasierte Simulation auf weitere physikalische Phänomene wie Fluidodynamik (KROTIL 2017) und Wärmetransportphänomene (ZHOU et al. 2014) ausgeweitet. Allerdings stellen viele Prozesse noch eine Herausforderung hinsichtlich der Modellierung des physikalischen Verhaltens und der Recheneffizienz dar (FUJIMOTO et al. 2017). Eine Einschränkung dieser Arbeit besteht daher auf Montageprozesse und sensorischen Wahrnehmungen, welche sich über bestehende Physik-Engines abbilden lassen.

Wie in der aufgabenorientierten Programmierung im Kontext von Schweißaufgaben gezeigt wird (BACKHAUS 2016), kann das Verhalten der Handhabungsoperatoren bei Fügeprozessen (z. B. zur Handhabung der Schweißpistole) simuliert werden. Neben vollumfänglich simulierbaren Montagefunktionen wie Handhabungsprozessen und Zusammensetzen wird in dieser Arbeit das Fügen durch Schweißen und Kleben daher mit aufgenommen (siehe Abbildung 4.7).

Als Referenzszenario wird im Rahmen der Erprobung des Systems ein roboterbasierter Schweißprozess abgebildet. Der Aufbau des Referenzszenarios wird in Abschnitt 8.1 detaillierter erläutert.

## 4 Einschränkung des Betrachtungsbereichs und Lösungsansatz

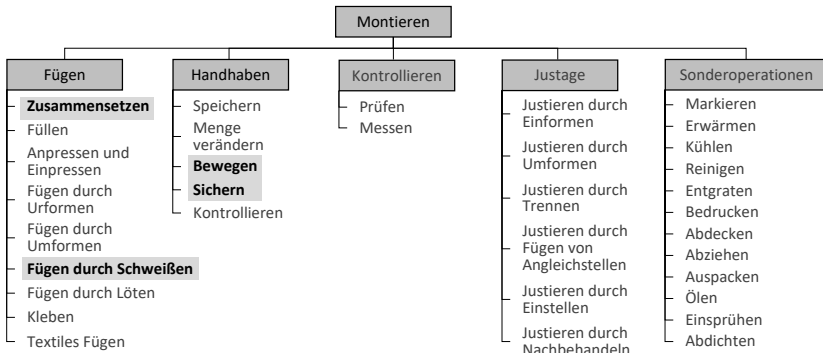


Abbildung 4.7: In dieser Arbeit betrachtete Montagefunktionen.

### 4.2.4 Anforderungen

Die Anforderungen an das System werden abgeleitet aus Anforderungen an aufgabenorientierte Programmiersysteme, welche durch KRUG (2012) und BACKHAUS (2016) definiert werden, und den Kriterien produzierender Unternehmen an externe Partner, welche in der Studie in Abschnitt 4.1.1 vorgestellt werden.

*Effizienz:* Die Aufwände zum Einsatz der aufgabenorientierten Programmierung bzw. Konfiguration dürfen den Nutzen der Anwendung nicht überschreiten. In dieser Arbeit wird die Planungsphase fokussiert, in welcher eine schnelle Umsetzung des Agenten notwendig ist, um lange Zykluszeiten zur Validierung von Lösungsalternativen zu vermeiden.

*Universalität und Flexibilität:* Das System soll eine Adaption auf die spezifischen Anwendungsfälle der Unternehmen zulassen. Hierzu muss sowohl eine Flexibilität hinsichtlich verschiedener Ressourcen und deren Schnittstellen vorliegen, als auch unternehmensspezifische Rahmenbedingungen der Prozesse berücksichtigt werden können.

*Transparenz:* Die Integration neuer Systeme in bestehende Unternehmensprozesse erfordert ein Vertrauen in die Entscheidungen, welche durch das System gefällt werden. Um zudem eine Integration des Nutzers in den Planungsprozess der aufgabenorientierten Konfiguration zu ermöglichen, müssen diese Entscheidungsprozesse nachvollziehbar und gegebenenfalls durch den Nutzer anpassbar sein.



Zusätzlich zu diesen drei Anforderungen, welche sich ebenfalls auf aufgabenorientierte Programmiersysteme beziehen, werden für die Domäne des Reinforcement Learnings die folgenden Anforderungen an das System gestellt.

*Einschätzbarkeit der Umsetzbarkeit:* Die Schätzung der Umsetzbarkeit auf DRL basierender Montageprozesse ist gerade für Anwender ohne RL-Expertise nicht trivial. Das System soll dem Nutzer daher signalisieren können, wie weit der betrachtete Montageprozess mit den gegebenen Rahmenbedingungen umsetzbar ist.

*Geringes notwendiges Expertenwissen:* Da in Unternehmen von kaum vorhandener Expertise im Bereich RL ausgegangen wird, soll auch zur Nutzung des Systems keine tiefgreifende RL-Expertise vorausgesetzt werden. Um der Anforderung der Universalität dennoch nachkommen zu können, sollen dem Nutzer in Aufgabenstellungen, welche nicht durch das System selbstständig gelöst werden können, Vorschläge bzw. Teillösungen durch das System abgeleitet werden.

*Adaptierbarkeit:* Da RL als Forschungsbereich noch einer großen Dynamik unterliegt und die Leistungsfähigkeit durch neue Algorithmen und Netzarchitekturen gesteigert wird, muss der Ansatz dieser Arbeit auch auf zukünftige Entwicklungen anwendbar sein. Um eine langfristige Anwendbarkeit des System zu gewährleisten, dürfen sich die Planungsprozesse des Systems nicht auf einen spezifischen Algorithmus (z. B. PPO) beschränken.

## 4.3 Lösungsansatz

### 4.3.1 Auswahl einer Grundarchitektur der aufgabenorientierten Programmierung

Um RL als eine zusätzliche Möglichkeit zur Umsetzung von Montagesystemen in die aufgabenorientierte Programmierung zu integrieren, wird auf bestehenden Ansätzen aufgebaut. Zum einen wird dadurch die Umsetzung in einem rechnergestützten Werkzeug ermöglicht, in welchem auch die Teilprozesse des Montagesystems abgebildet werden, die wiederum mit herkömmlichen Steuerungssystemen umgesetzt werden können. Zum anderen kann auf bestehende Informationsmodelle und Planungsmodule aufgebaut und diese um Charakteristiken von DRL erweitert werden.

## 4 Einschränkung des Betrachtungsbereichs und Lösungsansatz

---

KUSS (2020) erarbeitet ein aufgabenorientiertes Programmiersystem für einen sensorbasierten Schweißprozess. Allerdings wird der Fokus nicht auf die Herleitung montagerelevanter Skills gelegt, sondern ein System zur Ableitung von Prozessgrößen aus dem Produktmodell entwickelt. TRAPANI (2019) und BERG (2020) setzen ebenfalls den Ansatz der aufgabenorientierten Programmierung ein, allerdings konzentriert sich die Anwendung der Skills auf die Verteilung der Montageaufgaben auf mehrere Roboter bzw. auf die Verteilung auf Menschen und Roboter. Diese Anwendungsfälle gehen daher über den Betrachtungsbereich dieser Arbeit hinaus.

Von BACKHAUS (2016) wird dagegen ein generischer Ansatz zur Programmierung von SPS und Robotersystemen in der Montage erarbeitet, welcher eine ganzheitliche Betrachtung automatisierter Montagesysteme ermöglicht. Der Ansatz eignet sich daher für die Erweiterung um den Aspekt von DRL, welches ebenfalls einen generischen Ansatz darstellt.

Die Flexibilität des Einsatzbereichs des Programmiersystems wird in BACKHAUS (2016) durch die Abstraktion der Aufgabenstellung auf verschiedenen Ebenen und die Modularisierung der Funktionselemente des Programmiersystems ermöglicht.

Der aktuelle Planungsstand wird über ein Umweltmodell und ein Aufgabenmodell abgebildet. Das Umweltmodell beschreibt das Montagesystem und enthält die notwendigen Informationen, um ein Simulationsmodell zu generieren. Das Aufgabenmodell stellt eine hierarchische Beschreibung des Planungsprozesses im Programmiersystem dar. Abbildung 4.8 zeigt das für RL adaptierte Aufgabenmodell.

In den ersten vier Ebenen der Primärprozess-, Montagesequenz-, Sekundärprozess- und Skillebene erfolgt eine Planung und schrittweise Detaillierung der zur Montage eines Produkts notwendigen Skillsequenz. Auf diesen Ebenen findet noch eine abstrakte Beschreibung der Aufgabe statt, welche durch den Nutzer angepasst werden kann. In den weiteren Ebenen erfolgt in BACKHAUS (2016) eine Parametrisierung der Skills und die Generierung von herstellerepezifischem Code. Im Gegensatz hierzu vollzieht diese Arbeit zunächst eine Modellierung des MDP des Agenten, welcher die notwendigen Elemente der Lernumgebung des Agenten beinhaltet.

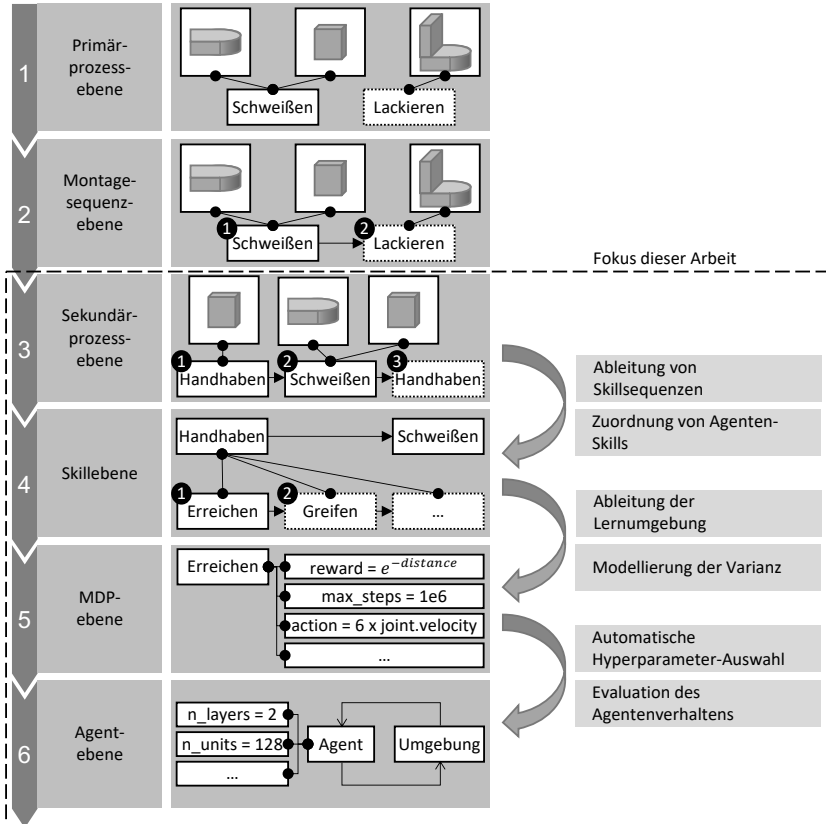


Abbildung 4.8: Für Reinforcement Learning adaptierte Aufgabenebenen und Funktionalitäten des aufgabenorientierten Programmiersystems nach BACKHAUS (2016)

Zur Umsetzung des Agenten erfolgt anschließend eine Konfiguration der Hyperparameter und das Erlernen des Agentenskills in einer simulierten Lernumgebung. Da auf dieser Ebene kein Prozesswissen notwendig ist, wird die Auswahl der Hyperparameter des Agenten mittels Meta-Learning (vgl. Abschnitt 3.4.3) automatisiert umgesetzt.

### 4.3.2 Auswahl eines Lösungsansatzes zur Umsetzung der aufgabenorientierten Konfiguration intelligenter Agenten

In BACKHAUS (2016) erfolgt das Ableiten von Lösungselementen wie Skill-Sequenzen und Programmcode über den Abruf von Wissensselementen, welche einzelne Planungsschritte über deterministische Ansätze wie logisches Schließen oder Planungsalgorithmen automatisieren. Im aufgabenorientierten Programmiersystem wird darüber hinaus die Möglichkeit zur Wiederverwendung bestehender Lösungen vorgesehen, allerdings werden die hierzu notwendigen Inferenzmechanismen nicht fokussiert.

Aufgrund des geringen Wissensstands zur Umsetzung auf RL basierender Montagesysteme und des fortschreitenden Ausbaus an Anwendungsfällen und Algorithmen werden deterministische Ansätze in dieser Arbeit nicht betrachtet. Diese Arbeit konzentriert sich daher auf Ansätze zur Wiederverwendung bestehender Lösungen. Um dennoch eine Flexibilität des Systems zu ermöglichen, erfolgt die Umsetzung dieses Ansatzes auf mehreren Abstraktionsebenen. Durch die Anpassung der Lösungselemente einer Aufgabenebene wird somit die Generierung einer neuen Gesamtlösung basierend auf Teillösungen bestehender Umsetzungen ermöglicht.

Dieser Lösungsansatz basierend auf der Wiederverwendung bestehender Lösungen wird im Kontext der wissensbasierten Systeme als *fallbasiertes Schließen* (engl. *case-based reasoning*, CBR) bezeichnet (BEIERLE & KERN-ISBERNER 2019). Im fallbasierten Schließen erfolgt der Lösungsprozess in vier Phasen:

- *Retrieve*: Selektierung ähnlicher Fälle
- *Reuse*: Wiederverwendung der Lösungen ähnlicher Fälle
- *Revise*: Überprüfung bzw. Anpassung der Lösung
- *Retain*: Aufnahme des neuen Falls mit zugehöriger Lösung in eine Fallbasis (BEIERLE & KERN-ISBERNER 2019)

In Abbildung 4.9 ist der Ansatz des fallbasierten Schließens für den Anwendungsfall dieser Arbeit dargestellt. Fälle und Lösungen basieren in dieser Arbeit auf einem Aufgabenmodell, welches das Informationsmodell der aufgabenorientierten Programmierung um RL-Elemente erweitert (Kapitel 5). Basierend auf der abstrakten Beschreibung eines Falls (z. B. ein Prozess innerhalb der Sekundärprozess-Ebene) kann eine Selektierung ähnlicher Fälle erfolgen.

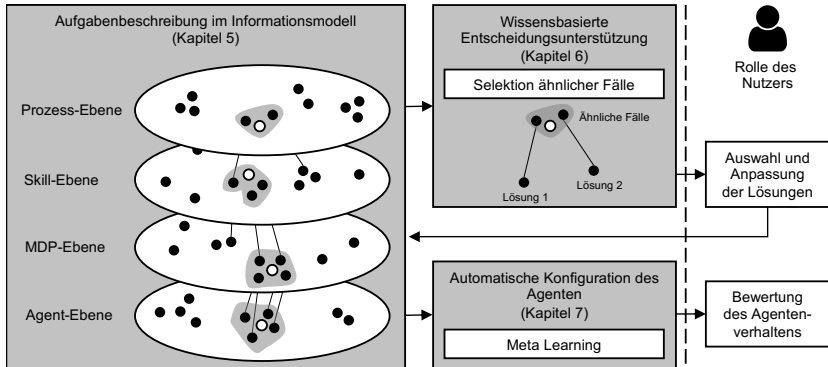


Abbildung 4.9: Ansatz des fallbasierten Schließens in der aufgabenorientierten Konfiguration intelligenter Agenten in der Montage

Für die höheren Aufgabenebenen können so bereits validierte Lösungselemente vom Nutzer wiederverwendet bzw. angepasst werden (Kapitel 6). Wie in der Zielplanung (Abschnitt 4.2) hergeleitet, stellt die Automatisierung der Hyperparameterauswahl eine Grundlage für die industrielle Anwendbarkeit des Systems dar. Der in Abschnitt 3.4.3 vorgestellte Ansatz des Meta Learnings zeigt einen Sonderfall des fallbasierten Schließens im Kontext des maschinellen Lernens, welcher im Rahmen dieser Arbeit eingesetzt wird (Kapitel 7).



## **5 Erweiterung des Informationsmodells automatisierter Montagesysteme**

In diesem Kapitel erfolgt eine Darstellung des Informationsmodells, welches die Grundlage für die wissensbasierte Entscheidungsunterstützung (Kapitel 6) und die automatische Konfiguration des Agenten (Kapitel 7) bildet. Abschnitt 5.1 ordnet das Informationsmodell daher zunächst in die Gesamtarchitektur ein und gibt einen Überblick über das Vorgehen und die Elemente des Informationsmodells. Ein wesentliches Element stellt die Beschreibung der Belohnungsfunktion zur Umsetzung von Skills dar. Die verwendeten Abstraktionsebenen werden in den Abschnitten 5.2 und 5.3 näher erläutert. Die zur Umsetzung eines Skills notwendigen Schnittstellen des Agenten zu dessen Umgebung werden in Abschnitt 5.4 betrachtet. Abschließend wird mit der Modellierung der Varianz in Abschnitt 5.5 und der Verwendung von Rahmenbedingungen (Abschnitt 5.6) eine ganzheitliche Beschreibung für die Modellierung auf Deep Reinforcement Learning basierender Montagesysteme gegeben.

### **5.1 Überblick zum Informationsmodell**

#### **5.1.1 Einordnung des Informationsmodells in die Gesamtarchitektur**

In der aufgabenorientierten Programmierung bildet das Informationsmodell den aktuellen Planungsstand über das Aufgabenmodell ab (BACKHAUS 2016). Zudem beinhaltet es das Umweltmodell, welches eine digitale Beschreibung der Ressourcen des Montagesystems enthält. In der Logikschicht werden die Inhalte des Informationsmodells zur Ableitung von Lösungselementen des Aufgabenmodells verwendet. Wie in Abbildung 5.1 gezeigt, erfolgt die Initialisierung des Informationsmodells durch die Angabe der Aufgabenbeschreibung und die Geräteinformationen der Ressourcen.

In der wissensbasierten Entscheidungsunterstützung erfolgt eine Ableitung der Elemente der Skill- und MDP-Ebene durch Wiederverwendung der Lösungselemente ähnlicher Aufgabenstellungen innerhalb der Datenbasis. In der automatischen Konfiguration des

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

Agenten geschieht eine automatische Auswahl der Hyperparameter des Agenten. In diesem Modul erfolgt zudem das Trainieren des Agenten sowie die Bewertung des Agentenverhaltens mittels einer simulativen Lernumgebung. Neben dem digitalen Modell des Montagesystems im Umweltmodell ist zum Aufbau der simulativen Lernumgebung die Beschreibung des MDP (u. a. Bewertung des Agenten mittels einer Belohnungsfunktion) im Aufgabenmodell notwendig.

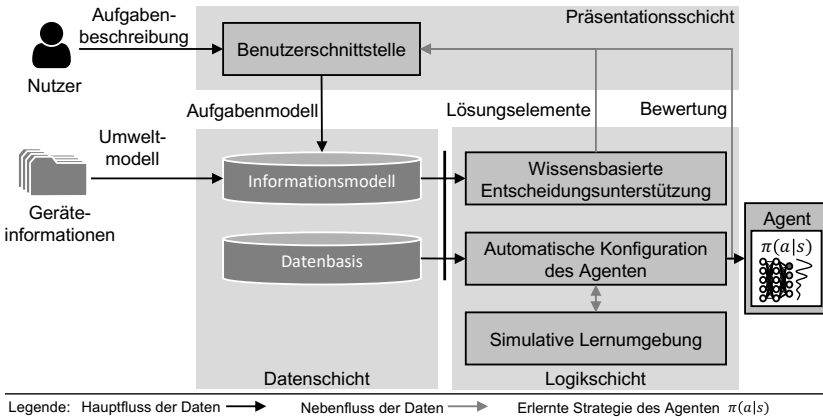


Abbildung 5.1: Gesamtarchitektur des Systems zur aufgabenorientierten Konfiguration intelligenter Agenten

### 5.1.2 Vorgehen zur Modellierung des Informationsmodells

Zur Erweiterung des Informationsmodells der aufgabenorientierten Programmierung ist eine Analyse der in Deep Reinforcement Learning notwendigen Elemente zur Beschreibung der Aufgabenstellungen und Lösungselemente einzelner Ebenen des Aufgabenmodells notwendig. Das in Abbildung 5.2 dargestellte Vorgehen zur Erweiterung des PPR-Konzepts basiert daher auf dem von BEIERLE & KERN-ISBERNER (2014) beschriebenen Vorgehen zum Aufbau einer Fallbasis für den Einsatz des fallbasierten Schließens.

Die Basis zur Sammlung einer repräsentativen Menge bestehender Fälle stellt eine strukturierte Literaturrecherche zu umgesetzten RL-Anwendungen aus der Robotik und Montage dar.



1	Sammlung einer repräsentativen Menge bestehender Umsetzungen
2	Identifikation der Besonderheiten einzelner Umsetzungen
3	Erweiterung des bestehenden PPR-Konzepts
4	Umsetzung von Testszenarien
5	Verfeinerung und Generalisierung des Informationsmodells

Abbildung 5.2: Vorgehen in der Konzeptionierung des Informationsmodells

Innerhalb der Literaturrecherche wurden 79 Umsetzungen aus 64 wissenschaftlichen Veröffentlichungen identifiziert, welche zum Aufbau der Datenbasis herangezogen werden können. Die Liste der verwendeten Literatur befindet sich im Anhang A.2.

Zudem ermöglichen beispielhafte Umsetzungen bestehender Open Source Bibliotheken wie OpenAI Gym (BROCKMAN et al. 2016), Deepmind Control Suite (TASSA et al. 2018) und Unity ML Agent (JULIANI et al. 2018) eine Analyse von 22 weiteren Umsetzungen, wodurch zur Modellierung des Informationsmodells auf insgesamt 101 bestehende Fälle zurückgegriffen werden kann.

Zur Verfeinerung und Generalisierung des Informationsmodells erfolgt eine Umsetzung von zwei Testszenarien und die Integration fehlender Elemente zur Beschreibung der Ähnlichkeit von Aufgabenstellungen (Kapitel 6 & 7). Abbildung 5.3 zeigt die beiden Testszenarien in der Simulationsumgebung.

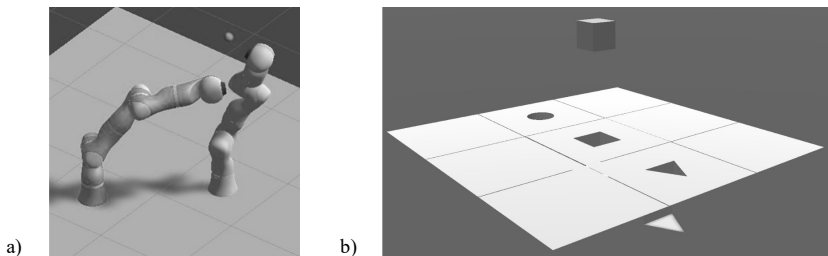


Abbildung 5.3: In der Modellierung des Informationsmodells verwendete Testszenarien.  
a) Szenario A: Kollisionsfreies Folgen einer Bahn durch einen Roboter  
b) Szenario B: Einsetzen eines Bauteils (Quader) mit Produktvarianz (ORTIZ 2019)

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

In Szenario A (Abbildung 5.3a) besteht die Aufgabe des Agenten im Folgen einer Bahn unter Berücksichtigung der Kollisionsfreiheit mit einem sich bewegenden zweiten Roboter. Die Startposition des virtuellen Bahnpunkts und die Startpose des zweiten Roboters stellen variable Startzustände dar. Szenario B (Abbildung 5.3b) zeigt das Einsetzen eines quaderförmigen Bauteils. Neben der relativen Position der beiden Bauteile erfolgt eine Variation der Bauteilform des zweiten Bauteils, welche durch unterschiedliche Anordnung der Taschen in einem Raster simuliert wird.

### 5.1.3 Elemente des Informationsmodells

Das in dieser Arbeit verwendete Informationsmodell erweitert das von BACKHAUS (2016) in der aufgabenorientierten Programmierung eingesetzte Informationsmodell. Ein Agent besteht aus einer Modellierung des MDP als Grundlage der simulativen Lernumgebung und der Definition der Hyperparameter des Lernalgorithmus (siehe Abbildung 5.4). Die simulative Lernumgebung und der Lernalgorithmus werden zum Erlernen eines Skills innerhalb der Trainingsphase genutzt. Der erlernte Skill wird durch die Strategie abgebildet.

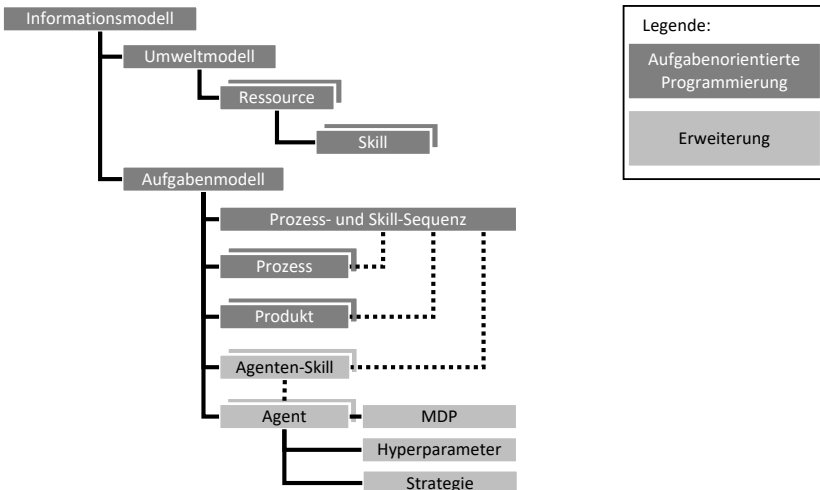


Abbildung 5.4: In das Informationsmodell der aufgabenorientierten Programmierung integrierte Elemente zur Konfiguration auf Deep Reinforcement Learning basierender intelligenter Agenten

Wie in Abschnitt 2.3.1 beschrieben, erfolgt die Aufgabenbeschreibung eines Agenten über einen MDP. Abbildung 5.5 gibt einen Überblick über die in diesem Kapitel näher erläuterten Elemente des MDP. Die formal als  $P(s'|s,a)$  definierte Transitionswahrscheinlichkeit wird über die im Simulationsmodell definierte Dynamik der Umgebung abgebildet. Die Umgebung des Agenten stellt nicht zwingend das gesamte Montagesystem dar, weshalb im Informationsmodell eine Referenzierung einzelner Elemente des Umweltmodells vorgesehen sind. Zur Definition der zu detektierenden Kollisionen innerhalb einer Simulation werden die Produkte und Ressourcen einzelnen Kollisionsebenen zugewiesen.

Eine in dieser Arbeit eingesetzte Erweiterung des MDP stellt die Definition variabler Startzustände  $\rho_0$  der Umgebung dar (MAHLER & GOLDBERG 2017). Die Modellierung der Varianz wird in Abschnitt 5.5 vorgestellt. Sowohl die Umgebung als auch die auftretende Varianz werden manuell durch den Nutzer definiert.

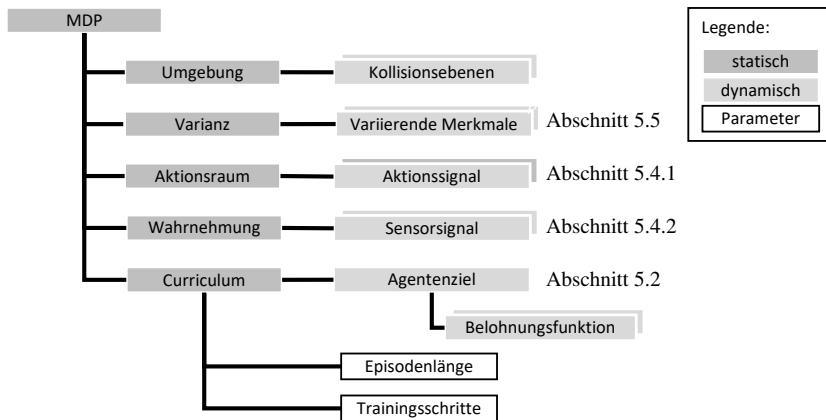


Abbildung 5.5: Elemente des zum Erlernen eines Agentenskills zu definierende Markov-Entscheidungsproblem (MDP)

In der aufgabenorientierten Programmierung erfolgt die Umsetzung eines Montageprozesses durch eine Sequenz von Skills, welche sich einzelnen Ressourcen zuordnen lassen. Im Kontext von DRL sind dagegen mehrere Ressourcen zur Umsetzung eines Skills notwendig, auf welche über Schnittstellen für Aktionssignale und Sensorsignale zugegriffen werden.

## **5 Erweiterung des Informationsmodells automatisierter Montagesysteme**

---

Die Bewertung der Aufgabenerfüllung durch eine Belohnungsfunktion erfolgt in dieser Arbeit durch die Abstrahierung der Belohnungsfunktion über Agentenziele, welche in Abschnitt 5.2 eingeführt werden. Neben der Belohnungsfunktion muss die maximale Anzahl an Trainingsschritten und somit die Summe der durch Interaktion mit der Umgebung definierten Erfahrungen vorgegeben werden. Zudem wird durch die Episodenlänge eine maximale Anzahl an Zeitschritten angegeben, nach denen eine Episode (d. h. ein Versuch zur Durchführung der Aufgabe in der Umgebung) abgebrochen wird und eine neue Initialisierung der Simulation erfolgt.

### **5.2 Abstraktion der Belohnungsfunktionen über Agentenziele**

#### **5.2.1 Agentenziele**

Die Definition der Aufgabenstellung eines Agenten erfolgt über Belohnungsfunktionen (siehe Abschnitt 3.3.1), deren Gestaltung und Parametrierung nicht deterministisch abgeleitet werden kann. Allerdings können bestehende Belohnungsfunktionen für ähnliche Aufgabenstellungen wiederverwendet werden. Auch können bestehende Belohnungsfunktionen zu neuen Aufgabenstellungen aggregiert werden. Im Informationsmodell wird daher eine Abstraktion der Belohnungsfunktion über drei Abstraktionsebenen (Agentenziel, Teilziel und elementares Ziel) vorgesehen, deren höchste Abstraktionsebene das Agentenziel darstellt.

Ein Agentenziel setzt sich aus Teilzielen zusammen. Auf der Abstraktionsebene der Teilziele wird im Informationsmodell davon ausgegangen, dass elementare Ziele auch über komplexere Belohnungsfunktionen zusammengeführt werden können. Auf der Abstraktionsebene der Agentenziele wird dagegen nur die Aggregation einzelner Teilziele betrachtet, welche hier durch unterschiedliche Gewichtung in die Belohnungsfunktion eines Agentenziels einfließen.

Neben der Gestaltung der Belohnungsfunktion durch das geeignete Hinführen des Agenten zu Zielzuständen, kann der Lernprozess durch eine Dekomposition des Agentenziels in sequenzielle Zielzustände vereinfacht werden (BRYN et al. 2017). Die Dekomposition des Agentenziels in sequenzielle Zielzustände ermöglicht weiterhin ein Steuern des zu erlernenden Agentenverhaltens, welches in komplexeren Aufgabenstellungen durch eine einfache Definition des Zielzustands von den Vorstellungen des Menschen abweichen kann.

## 5.2 Abstraktion der Belohnungsfunktionen über Agentenziele

So stellen POPOV et al. (2017) bei der Definition einer Pick-and-Place-Aufgabe durch Vorgabe der Zielposition des Handhabungsobjekts fest, dass der Agent ein Schieben des Objekts zur Zielposition erlernt, ohne das Objekt zuvor zu greifen. In der Umsetzung wird der Prozess in POPOV et al. (2017) daher in Zwischenziele zum Erreichen des Objekts durch den Greifer, dem Heben des Objekts und das Erreichen der Zielkoordinaten aufgeteilt.

Abbildung 5.6 gibt einen Überblick der Abstraktionsebenen am Beispiel der Umsetzung einer Pick-and-Place-Aufgabe.

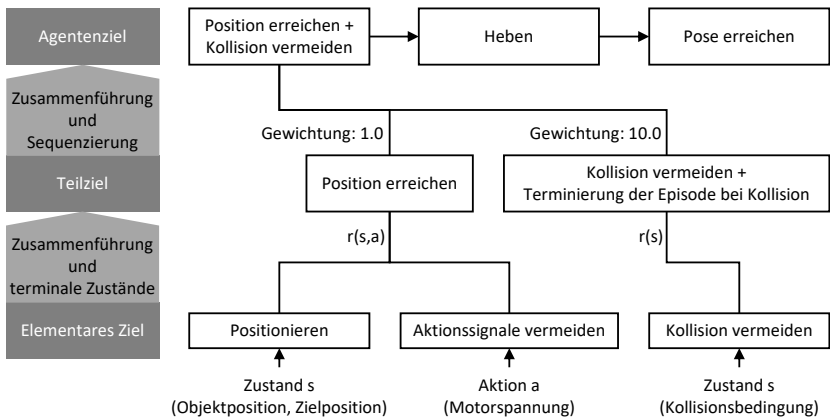


Abbildung 5.6: Beispielhafte Zusammenführung von elementaren Zielen und Teilzielen zum Agentenziel "Pick-and-Place"

### 5.2.2 Teilziele

Elementare Ziele wie "Positionieren" können bei einfachen Aufgabenstellungen für sich eine vollumfängliche Bewertung der Aufgabenerfüllung des Agenten darstellen. Allerdings erfordern komplexere Zielstellungen eine Aggregation mehrerer elementarer Ziele. Als zusätzliche Abstraktionsschicht wird im Informationsmodell daher die Definition von Teilzielen vorgesehen, welche einen spezifischen Aspekt der Aufgabenerfüllung bewerten.

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

---

Teilziele können funktionale und nicht-funktionale Bedeutungen hinsichtlich der Aufgabenstellung besitzen. Als nicht-funktionale Ziele wird die Erfüllung von Rahmenbedingungen definiert, welche zusätzlich zur Zielstellung des Agenten eingehalten werden müssen (z. B. Kollisionsfreiheit). Elementare Ziele können hinsichtlich ihrer Zuordnung zu funktionalen Zielen und nicht-funktionalen Zielen abweichen. So wird die Sanktionierung hoher Aktionssignale teilweise als Nebenbindung zur energiesparenden Ausführung der Aufgabe eingesetzt. In anderen Arbeiten hat sich die Sanktionierung hoher Aktionssignale (z. B. Drehmomente einzelner Roboterachsen) als Hilfsmittel zur Vermeidung großer Auslenkungen und somit zum vereinfachten Lernen funktionaler Zielstellungen erwiesen. Die Abstraktionsschicht der Teilziele definiert somit die Funktion eines elementaren Ziels in der Aufgabenerfüllung des Agenten.

Die Erreichung einer geringen Taktzeit wird im Informationsmodell nicht explizit als Teilziel abgebildet. Da ein Agent mit jedem Zeitschritt eine maximale Erhöhung der Belohnung und somit eine Näherung zum Zielzustand anstrebt, wird diese Rahmenbedingung implizit erfüllt.

Funktionale Ziele unterscheiden sich hinsichtlich der Art, wie ein Zielzustand herbeigeführt werden soll. Die Gestaltung der Belohnungsfunktion ist davon abhängig, ob ein Zielzustand erreicht und gehalten werden oder schnellstmöglich angefahren werden soll. So hat die Terminierung einer Episode bei Erreichung eines Zustands innerhalb einer Belohnungsfunktion einen wesentlichen Einfluss auf das gelernte Agentenverhalten (DONG et al. 2020).

Abbildung 5.7 zeigt zwei mögliche Ausführungen der Belohnungsfunktion für die Teilziele "Position erreichen" und "Position anfahren". Das Erreichen und Halten einer Position innerhalb eines Toleranzbereichs wird hier durch eine exponentielle Erhöhung der Belohnung bei kleineren Abständen bewirkt. Zusätzlich wird innerhalb des Toleranzbereichs, angelehnt an LOPEZ et al. (2019), eine weitere Erhöhung des Belohnungssignals für kleine Abstandswerte durchgeführt. Ohne Terminierung der Episode im Toleranzbereich erlernt der Agent somit nicht nur den Zielwert zu erreichen, sondern auch bei Annäherung abzubremsen, um im Toleranzbereich zu bleiben und somit auch zukünftige Belohnungen maximal zu halten. Bei der Terminierung der Episode mit einem hohen Belohnungssignal im Toleranzbereich erlernt der Agent dagegen ein Anfahren der Zielposition mit hoher Geschwindigkeit.

## 5.2 Abstraktion der Belohnungsfunktionen über Agentenziele

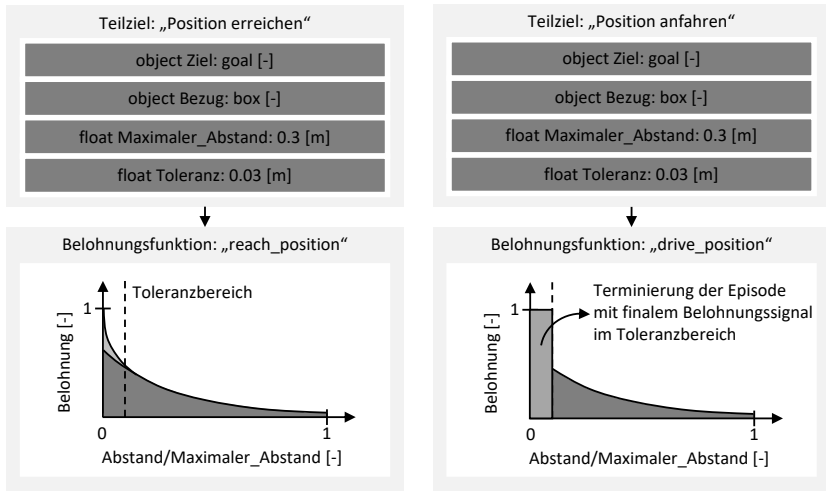


Abbildung 5.7: Unterschiedliche Umsetzung der Teilziele "Position erreichen" und "Position anfahren"

Die Zusammenführung mehrerer elementarer Ziele zu einem Teilziel kann hinsichtlich ihrer Integration in die Belohnungsfunktion neben einer Addition der jeweiligen Belohnungssignale auch über andere Funktionen wie der Multiplikation der Belohnungssignale erfolgen (siehe Abschnitt 3.3.1). Durch die Verfolgung mehrerer elementarer Ziele kann sich für den Agenten zudem ein multikriterielles Entscheidungsproblem ergeben (BRYN et al. 2017), wenn diese Ziele miteinander konkurrieren. Abhängig von der Aufgabenstellung muss daher eine Gewichtung einzelner Ziele erfolgen, um das gewünschte Agentenverhalten zu erzielen.

Im Informationsmodell wird daher für Teilziele eine Referenzierung auf eigene Belohnungsfunktionen vorgenommen, welche in ihrer Implementierung komplexere Formen annehmen können. Abbildung 5.8 zeigt die Zusammenführung der elementaren Ziele "Positionieren" und "Orientieren" zu einem Teilziel "Pose erreichen".

Die beiden elementaren Ziele stellen bezogen auf die Zustände des Handhabungsobjekts im kartesischen Raum keine konkurrierenden Zielstellungen dar. In Arbeiten zur Erreichung einer Zielpose wird dagegen eine Gewichtung der elementaren Ziele vorgenommen, wenn sich der Aktionsraum auf die rotatorischen Gelenke des Roboters bezieht.

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

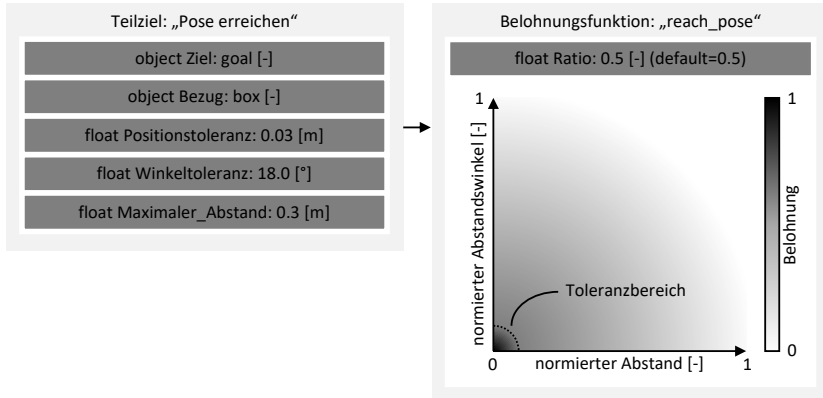


Abbildung 5.8: Zusammenführung der elementaren Ziele "Positionieren" und "Orientieren" zum Teilziel "Pose erreichen". Die Belohnungsfunktion kann eigene Parameter besitzen, welche deren Gestaltung definieren (hier beispielhaft die Ratio der Belohnungssignale hinsichtlich Position und Abstand).

Eine ungeeignete Gewichtung der beiden Ziele kann dazu führen, dass der Lernprozess des Agenten erschwert oder dieser zu einem lokalen Optimum geführt wird.

So hat die zu hohe Gewichtung des Winkelabstands in der Umsetzung eigener Testszenarien zu einem Agentenverhalten geführt, in dem die Erreichung einer Roboterpose mit der Zielwinkelstellung des Objekts unter Vernachlässigung der Position einherging. Durch Adaption einer höheren Gewichtung des Abstands zur Position analog zu Vorarbeiten (GUDIMELLA et al. 2017, MATAS et al. 2018, LOPEZ et al. 2019) konnte die übergeordnete Zielstellung erreicht werden.

Die Ausprägung der Aktionssignale (z. B. Bewegung des Endeffektors in kartesischen Koordinaten bzw. Ansteuerung einzelner rotatorischer Achsen) hat somit einen Einfluss auf die Parametrisierung der Belohnungsfunktion.

### 5.2.3 Arten elementarer Ziele

Elementare Ziele stellen im Informationsmodell die semantischen Einheiten und Funktionen dar, auf denen eine Belohnungsfunktion aufgebaut werden kann. Diese elementaren Ziele können sowohl die Form einer diskreten Bewertung eines Zustandsbereichs



## 5.2 Abstraktion der Belohnungsfunktionen über Agentenziele

---

als auch die Form einer kontinuierlichen Bewertung von Werten annehmen. Die Bewertung kann sich zudem sowohl auf gewünschtes als auch ungewünschtes Verhalten beziehen.

Wie im vergangenen Abschnitt anhand der Teilziele zum "Position erreichen" bzw. "Position anfahren" erläutert (vgl. Abbildung 5.7), wird durch die Terminierung einer Episode beim Erreichen des Zielzustands das Agentenverhalten und somit die Art der Zielerfüllung beeinflusst. Neben der Art des zu erreichenden Zielzustands wird daher auch die Art der Zielerfüllung durch das zusätzliche elementare Ziel "Zielzustand anfahren" berücksichtigt.

Die elementaren Ziele lassen sich daher in vier Kategorien einteilen:

- *Zielzustand erreichen*: Bewertung der Näherung eines Werts an einen Zielwert (z. B. Positionieren, Orientieren)
- *Zielzustand anfahren*: Diskrete Bewertung eines gewünschten Zustandsbereichs mit Terminierung (z. B. Erreichung einer Zielposition im Toleranzbereich)
- *Werte vermeiden*: Kontinuierliche Bewertung ungewünschter Werte (z. B. Höhe des Aktionssignals)
- *Zustände vermeiden*: Diskrete Bewertung eines ungewünschten Zustandsbereichs bzw. Terminierung der Episode (z. B. Kollision, Fallenlassen eines Handhabungsobjekts)

### 5.2.4 Belohnungsfunktionen

In der Literatur lassen sich für gleiche Aufgabenstellungen verschiedene Definitionen der Belohnungsfunktion finden. Da selbst auf der Ebene der elementaren Ziele kein Konsens zur optimalen Gestaltung der Belohnungsfunktion herrscht, wird im System die Möglichkeit der Referenzierung unterschiedlicher Belohnungsfunktionen vorgesehen.

Als allgemeine Kriterien zur Gestaltung von Belohnungsfunktionen nennt WINDER (2020) die schnelle Berechenbarkeit der Belohnungsfunktion, da diese im Trainingsprozess in jedem Schritt ausgeführt werden muss, sowie die Verwendung von glatten Funktionen, welche sich in einem kontinuierlichen Trainingsverlauf des Agenten manifestieren.

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

---

Die Höhe des Belohnungssignals kann zudem unterschiedliche Ausmaße annehmen. Da in Gradientenverfahren, wie sie in Deep Learning eingesetzt werden, die Skalierung der Eingangs- und Zielgrößen eines Modells ähnliche Skalierungen haben sollten (GOOD-FELLOW et al. 2018), wird auch in Deep Reinforcement Learning eine Normalisierung des Signals als sinnvoll erachtet (DONG et al. 2020). Dieses Kriterium der Normalisierung der Belohnungssignale wird in der Literatur nicht stringent eingehalten und daher auch im Informationsmodell nicht berücksichtigt.

SUTTON & BARTO (2018) schlagen ein positives Belohnungssignal für gewünschtes Verhalten und ein negatives Belohnungssignal für ungewünschtes Verhalten vor. Da die subjektive Bewertung einzelner Werte je nach Betrachtungsweise der Aufgabenstellung variieren kann, lässt dies keine direkte Zuordnung von Belohnungssignalen zu Zielstellungen zu. So kann die Erreichung einer Zielposition durch die Sanktionierung hoher Abstände zu den Zielkoordinaten über

$$\text{Belohnung}^{\text{Variante 1}} = -\alpha_1 \times \text{Abstand} \quad (5.1)$$

mit dem einstellbaren Parameter  $\alpha_1$  erfolgen. Da der Agent die Maximierung des Belohnungssignals anstrebt, stellt dies eine geeignete Formulierung der Aufgabenstellung über eine Belohnungsfunktion dar. Mit einer anderen Betrachtungsweise kann auch die Erhöhung des Belohnungssignals bei Verringerung des Abstands über die Funktion

$$\text{Belohnung}^{\text{Variante 2}} = e^{-\alpha_2 \times \text{Abstand}} \quad (5.2)$$

mit dem einstellbaren Parameter  $\alpha_2$  eine valide Bewertung der Zielerfüllung des Agenten sein. Im Informationsmodell dient die Abstraktion der Belohnungsfunktionen über Ziele daher zur Einhaltung einer einheitlichen Semantik hinsichtlich des gewünschten Agentenverhaltens.

Der Einsatz mathematischer Funktionen wie der Exponentialfunktion in Gleichung 5.2 ermöglicht eine Normalisierung des Belohnungssignals, allerdings hängt die Höhe des Belohnungssignals vom möglichen Wertebereich der Abstände ab. In Arbeiten, die den Abstand zur Formulierung der Belohnungsfunktion nutzen, wird daher ein Wert für den maximalen Abstand definiert und dieser zur Berechnung eines normalisierten Abstands eingesetzt.

## 5.2 Abstraktion der Belohnungsfunktionen über Agentenziele

---

Andere Parameter der Belohnungsfunktion, wie die in Gleichung 5.2 verwendete Gewichtung im Exponenten ( $\alpha$ ), lassen sich nicht über Prozesswissen deterministisch ableiten, sondern müssen basierend auf dem erlernten Agentenverhalten evaluiert werden.

Wie in Abschnitt 3.4.2 aufgeführt, stellt die simultane Optimierung solcher Parameter der Belohnungsfunktion und den Hyperparametern des Agenten einen derzeitigen Forschungsansatz dar.

Im Informationsmodell wird daher zwischen Parametern zur aufgabenbezogenen Skalierung der Belohnungsfunktion (Skalierungsparameter) und Parametern zur Gestaltung der Belohnungsfunktion (Gestaltungsparameter) unterschieden. Skalierungsparameter werden in unterschiedlichen Ausführungen der Belohnungsfunktion genutzt und können auf der abstrakteren Ebene der Ziele über das Prozesswissen des Nutzers definiert werden.

Für die Initialisierung einer Belohnungsfunktion ist zudem ein Bezug zu entsprechenden Elementen des Montagesystems im Umweltmodell notwendig. In bestehenden Arbeiten sind diese Bezüge auf einzelne Objekte des Montagesystems (z. B. Handhabungsobjekte, Endeffektoren) oder konstante Werte (z. B. Zielkoordinaten) gerichtet. Um einen generischen Ansatz im Informationsmodell zu ermöglichen, richten sich die Bezüge im Informationsmodell stets auf Objekte im Umweltmodell, welche eine Komponente des realen Montagesystems darstellen oder die Ausprägung als virtuelle Objekte zur Aufgabenbeschreibung (z. B. virtueller Zielpunkt) haben können.

Ein Agentenziel beinhaltet somit alle Informationen, welche zur Wiederverwendung einer Belohnungsfunktion notwendig sind und mit Prozesswissen hergeleitet werden können. Abbildung 5.9 zeigt das elementare Ziel "Positionieren", in welchem die Zielstellung durch Bezug zu Objekten im Umweltmodell und dem Skalierungsparameter "Maximaler\_Abstand" definiert wird. Neben der Definition der Ausprägung eines Parameters (u. a. als Ganzzahl, Gleitkommazahl oder als Bezug zu Objekten) sind im Informationsmodell Standardwerte für Gestaltungsparameter der Belohnungsfunktion sowie eine Beschreibung der Auswirkung des Gestaltungsparameters auf die Bewertungen des Agentenverhaltens vorgesehen.

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

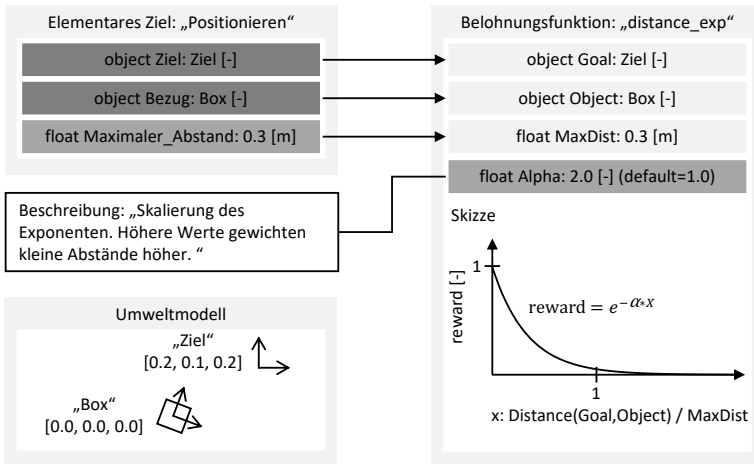


Abbildung 5.9: Definition des elementaren Ziels "Abstand minimieren" durch exponentielle Erhöhung der Belohnung bei Reduktion des Abstands

### 5.3 Agenten-Skills

#### 5.3.1 Zuordnung von Agenten-Skills zu Agentenzielen

Zur Umsetzung von Primär- und Sekundärprozessen in der Montage werden Agenten-Skills eingeführt. Agentenzielen dienen als Abstraktionsschicht bestehender Gestaltungen einer Belohnungsfunktion, welche für gleiche Aufgabenstellungen wiederverwendet werden können. Agenten-Skills nehmen Bezug auf Agentenzielen und abstrahieren die Aufgabenstellung durch Referenzierung der zieldefinierenden Objekte eines Agentenzielen zu aufgabenspezifischen Objekten im Umweltmodell. Neben Agentenzielen können sich Agenten-Skills zudem Simulationsmodulen bedienen, welche eine Manipulation virtueller Zielobjekte in der Simulation vornehmen können.

Die mittels der Belohnungsfunktion eines Agentenzielen zu erlernenden Agenten-Skills unterscheiden sich durch unterschiedliche Bezüge von Zielobjekten der Agentenzielen zu Objekten im Umweltmodell. Abbildung 5.10 veranschaulicht die Wiederverwendung des Agentenzielen "Pose erreichen" in den Agenten-Skills "Führen" und "Weitergeben".

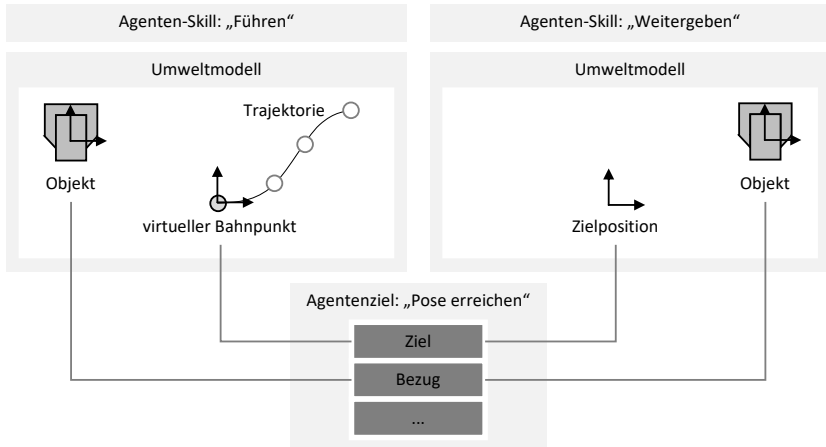


Abbildung 5.10: Einsatz eines Agentenziels für unterschiedliche Agenten-Skills

Das Weitergeben eines Handhabungsobjekts äußert sich durch die Vorgabe fixer Zielpunkte im Umweltmodell, welche durch den Nutzer definiert werden. Das Verfolgen einer Bahn wird dagegen als das Erreichen und Halten der Pose eines virtuellen Bahnpunkts definiert, welcher sich entlang einer vorgegebenen Trajektorie in einer definierten Geschwindigkeit bewegt.

Die Trajektorie kann durch den Nutzer im Umweltmodell definiert werden oder eine Teilkomponente bestehender Objekte innerhalb des Montagesystems darstellen. Im Falle eines Schweißprozesses bezieht sich Trajektorie etwa auf die Schweißbahn. Bei einem Klebeprozess bezieht sich die Trajektorie auf den Verlauf der Klebedüse beim Auftragen des Klebstoffs.

### 5.3.2 Unterscheidung von Agenten-Skills zu Ressourcen-Skills

Die Taxonomie von Agenten-Skills orientiert sich in dieser Arbeit an bestehenden Definitionen von Ressourcen-Skills in der aufgabenorientierten Programmierung, welche sich auf die Fähigkeit einzelner Ressourcen beziehen und zur Umsetzung von Montageprozessen über Skillsequenzen genutzt werden (siehe Abbildung A.3 im Anhang). Agenten-Skills setzen eine bewertbare Zustandsänderung im Montagesystem voraus, weshalb deren Taxonomie von der Taxonomie der Ressourcen-Skills teilweise

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

abweichen muss. So stellt "Greifen" einen Ressourcen-Skill dar, welcher sich einer Ressource wie einem Klemmgreifer zuordnen lässt und in der automatisierten Montage beispielweise durch ein binäres Stellsignal umgesetzt werden kann (BACKHAUS 2016).

Die Bewertbarkeit des Greifens ist bei komplexen Geometrien des Handhabungsobjekts nicht trivial und stellt ein derzeitiges Forschungsthema dar (MAHLER et al. 2019). Ein erfolgreiches Greifen lässt sich auch durch das Erreichen einer Zielhöhe durch das Handhabungsobjekt nach dem Greifvorgang definieren. Die Zielerreichung lässt sich somit über eine Belohnungsfunktion bewerten, allerdings ändert sich der korrespondierende Skill von "Greifen" zu "Heben" (siehe Abbildung 5.11).

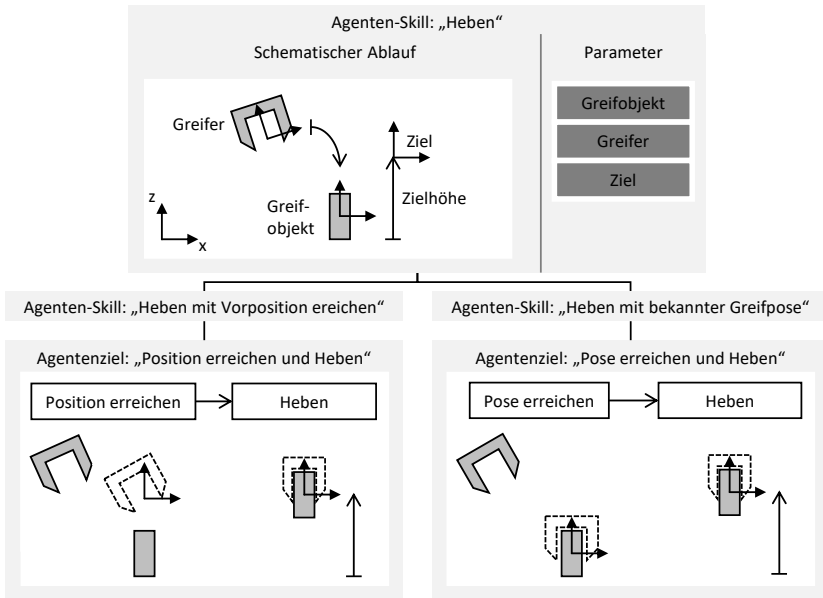


Abbildung 5.11: Unterschiedliche Varianten des Agenten-Skills "Heben" durch zusätzliche Definition von Zwischenzielen

Weiterhin sieht das Informationsmodell eine geringere Abstraktion von Agenten-Skills vor, um die mögliche Vereinfachung einer Aufgabenstellung durch Vorgaben des Nutzers zu berücksichtigen. Abbildung 5.11 zeigt den Agenten-Skill "Heben" mit den zwei weiteren Ausprägungen "Heben mit Vorposition erreichen" und "Heben mit bekannter

Greifpose". So lässt sich der Agenten-Skill "Heben" durch die Vorgabe einer bekannten Vorposition in der Nähe des Handhabungsobjekts vereinfachen oder im Fall einfacher Geometrien durch Vorgabe der Greifpose in Bezug auf das Handhabungsobjekt durch den Nutzer definieren.

### 5.4 Schnittstellen des Agenten mit der Umgebung

#### 5.4.1 Aktionsraum

Der Aktionsraum definiert die Aktionssignale, welche durch einen Agenten an die Umgebung gesendet werden können, um eine Änderung innerhalb der Umgebung zu bewirken. In den Arbeiten zu DRL beziehen sich diese Signale vornehmlich auf Steuergrößen, welche durch eine nachgelagerte Regelung in das reale System eingebracht werden. Durch diese Kaskadierung wird die Aufgabenstellung für den Agenten vereinfacht, da dieser nicht die Stabilität der Regelkreises mitberücksichtigen muss (KOBBER et al. 2013).

Die minimale Inferenzzeit eines Agenten ist in realen System zudem durch die höhere Rechenzeit von Deep-Learning-Ansätzen begrenzt, weshalb in bestehenden Arbeiten für Agenten in realen Systemen Taktzeiten von 10 bis 100 Hz angesetzt werden, welche die Steuergrößen eines Regelungssystems mit Taktzeiten im kHz-Bereich vorgeben (POPOV et al. 2017, PHAM et al. 2018, THOMAS et al. 2018, SCHOETTLER et al. 2019).

Aktionen können eine diskrete oder kontinuierliche Ausprägung besitzen (GRAESSER & KENG 2019). Zur Verringerung der Komplexität kann eine Teilauswahl der möglichen Aktionen gewählt werden, wenn einzelne Aktionen nicht zur Zielerreichung benötigt werden (LÄMMLE et al. 2020). Auch kann durch die Diskretisierung kontinuierlicher Aktionen und durch die Transformation von absoluten Werten wie Positionen in relative Werte wie Relativ-Positionen eine Verringerung der Komplexität herbeigeführt werden (GRAESSER & KENG 2019).

Neben der Art des Aktionssignals und der Definition seines Bezugs zu einer Ressource ist bei kontinuierlichen Signalen eine Angabe des Wertebereichs notwendig, welcher dem Agenten zur Verfügung stehen. Da in Abhängigkeit der Aufgabenstellung auch langsamere Bewegungsausführungen gewählt werden können, werden diese nicht im Modell der Ressource, sondern in der Modellierung des Agenten definiert.

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

Wie die Darstellung eines Aktionsraums in Abbildung 5.12 zeigt, ermöglicht die Art des Signals nicht zwingend eine Ressourcen-Zuordnung. So wird die von POPOV et al. (2017) realisierte Handhabungsaufgabe durch Ansteuerung von sechs Achsgeschwindigkeiten des Roboters und drei Achsgeschwindigkeiten der Greiferfinger umgesetzt. Um bei der Wiederverwendung von Aktionsräumen einen Bezug zum Umweltmodell der neuen Aufgabe herstellen zu können, werden in dieser Arbeit daher jedem Aktionssignal die entsprechenden Ressourcen-Skills zugeordnet.

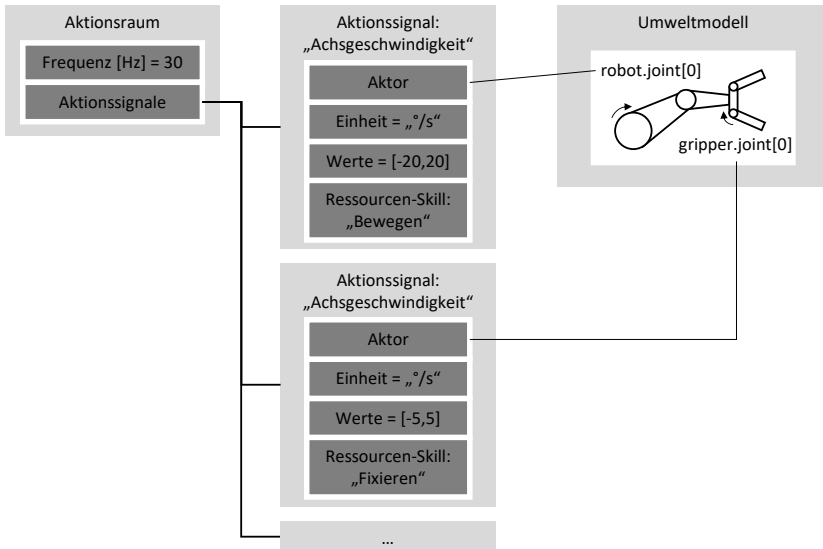


Abbildung 5.12: Zuordnung gleicher Ausprägungen der Aktionssignale zu unterschiedlichen Aktoren im Umweltmodell basierend auf Ressourcen-Skills in Anlehnung an den Anwendungsfall von POPOV et al. (2017)

### 5.4.2 Wahrnehmung

Die Wahrnehmung des Agenten gibt den aktuellen Zustand der Umgebung wieder, auf Basis dessen durch die erlernte Strategie entsprechende Aktionssignale ausgewählt werden. Wie in Abschnitt 3.3.2 erläutert, kann die Wahrnehmung des Agenten in der Trainingsphase über simulierte Sensorsignale erfolgen. Da sich diese Arbeit auf die Planungsphase konzentriert, werden derzeitige Forschungsansätze zur Übertragung des Agenten auf reale Systeme nicht betrachtet.



Analog zu bestehenden Arbeiten zur Modellierung sensorischer Elemente in Produktionssystemen (HAMMERSTINGL 2020, GONNERMANN et al. 2020, KIEFER 2020) ist eine Zuordnung der Art eines Sensorsignals zu der Art der Zustände, welche durch dieses Sensorsignal wahrgenommen werden, möglich. Im Informationsmodell dieser Arbeit wird diese Zuordnung nicht vorgesehen, da im Planungsprozess zusätzliche Aufwände durch eine explizite Definition der notwendigen Zustandsinformationen der Umgebung notwendig sind.

Unabhängig von der Art der Sensorsignale in der Wahrnehmung unterscheidet sich die Beobachtbarkeit der Umgebung nach GRAESSER & KENG (2019) in drei Arten:

- *Vollständig beobachtbar*: Alle zur Auswahl der Aktionssignale notwendigen Zustände können zu jedem Zeitschritt wahrgenommen werden (Abbildung 5.13a).
- *Teilweise beobachtbar basierend auf der Historie*: Die notwendigen Zustandsinformationen können nicht über die Wahrnehmung am einem einzelnen Zeitschritt abgeleitet werden, aber über die Wahrnehmung über mehrere Zeitschritte hinweg (z. B. lässt sich die Geschwindigkeit eines Objekts auf Basis der Position des Objekts zu mehreren Zeitschritten ableiten wie in Abbildung 5.13b).
- *Nicht vollständig beobachtbar*: Die notwendigen Zustandsinformationen sind nicht zu jedem Zeitpunkt durch die derzeitige Wahrnehmung oder eine bestimmte Historie vergangener Zeitpunkte zugänglich (z. B. bei zeitweiser Bewegung eines Objekts außerhalb des Sichtfelds der Kamera wie in Abbildung 5.13c).

Teilweise Beobachtbarkeit wird im Informationsmodell durch die Angabe der Anzahl von Sensorsignalen angegeben, welche an den Agenten übermittelt werden (Historie  $> 1$ ). Nicht vollständige Wahrnehmbarkeit wird im Informationsmodell dagegen als Rahmenbedingung des Wahrnehmungsraums betrachtet.

Die Ausprägung eines Signals der Wahrnehmung unterscheidet sich hinsichtlich seiner Dimensionalität, von welcher neben der Komplexität der Aufgabenstellung auch die Auswahl und Implementierung der Architektur des künstlichen neuronalen Netzes abhängen. Zudem erfolgt eine unterschiedliche Vorverarbeitung der Signalwerte. Skalare Sensorwerte wie die Rotationsgeschwindigkeit eines Gelenks in Abbildung 5.13a können in ihrem Wertebereich eingeschränkt werden, um das Signal zu normalisieren, bevor dieses in einem künstlichen neuronalen Netz weiterverarbeitet wird.

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

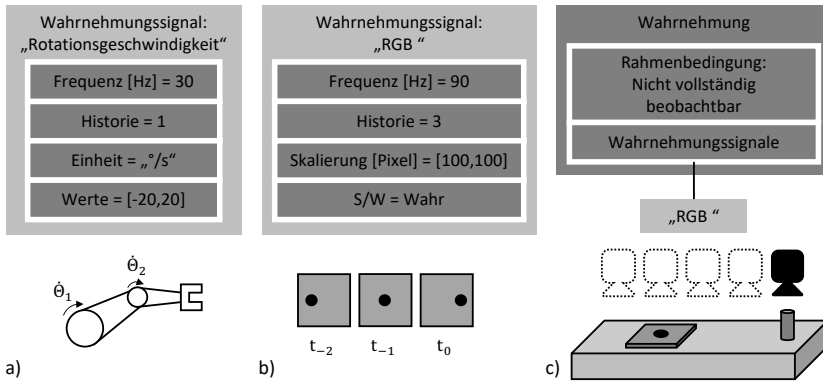


Abbildung 5.13: Arten der Beobachtbarkeit a) Geschwindigkeit eines Endeffektors über Achsgeschwindigkeiten b) Wahrnehmung der Geschwindigkeit eines Objekts über mehrere Kamerabilder c) Notwendigkeit eines Gedächtnisses des Agenten bei nicht vollständiger Beobachtbarkeit

Kamerabilder werden zur Interpretation der Umgebung durch den Agenten zudem nicht zwingend in der vollständigen Auflösung benötigt, wie diese vom Kamerasystem zur Verfügung gestellt wird. Durch Verringerung der Auflösung werden die notwendige Datentransformation und somit die Komplexität der Aufgabe für den Agenten reduziert. So wird das Zusammensetzen von Bauteilen in [SCHOETTLER et al. \(2019\)](#) mit einer reduzierten Auflösung von  $32 \times 32$  Pixeln umgesetzt. Weiterhin werden in der Arbeit die Farbwerte der RGB-Kamera in Graustufen umgewandelt, da die Farbcodierung der Pixel keine notwendigen Informationen zur Aufgabe enthält und sich die Aufgabenstellung durch den Agenten weiter vereinfacht.

### 5.5 Modellierung der Varianz

#### 5.5.1 Ausprägungen variierender Merkmale

Eine wesentliche Anforderung des Agenten besteht in der Robustheit hinsichtlich der Varianz im Montageprozess, welche sich durch variierende Merkmale der Objekte der Umgebung manifestiert. Merkmale lassen sich unterteilen in nominale, ordinale und metrische Merkmale ([POLASEK 1994](#)), deren Definitionen der Wertebereiche auf unterschiedliche Art erfolgen. Metrische und ordinale Merkmale beziehen sich

auf quantitative Merkmale, wobei die Beschreibung über ordinale Merkmale eine diskrete Aufzählung von Werten mit unterschiedlichen Abständen zwischen den Werten ermöglicht. Die Aufgabe eines Agenten lässt sich in solchen Fällen vereinfachen, da sie nicht für den gesamten Wertebereich gelernt werden muss. Nominale Merkmale beziehen sich dagegen auf qualitative Merkmale, welche sich durch keinen Zahlenwert ausdrücken lassen, wie beispielsweise die Zuordnung unterschiedlicher Produktmodelle zu einem Handhabungsobjekt.

Abbildung 5.14 zeigt am Beispiel der Formatflexibilität, wie die Beschreibung der Varianz über unterschiedliche Merkmalsklassen abgebildet werden kann.



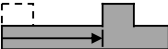
Ausprägung	Qualitativ	Quantitativ
Aufzählung	<b>Nominale Merkmale</b> Beispiel: Geometrie 	<b>Ordinale Merkmale</b> Beispiel: Diskrete Position 
Intervall		<b>Metrische Merkmale</b> Beispiel: Kontinuierliche Position  $x \in [0.0, 2.0]$

Abbildung 5.14: Beispielhafte Ausprägungen der Varianz zur Definition von Formatflexibilität

### 5.5.2 Benutzerdefinierte Merkmale

Neben der Implementierung der Simulation dient die Modellierung der Varianz im Informationsmodell der Evaluation eines erlernten Agentenverhaltens. Durch die Analyse des Agentenverhaltens in Abhängigkeit einzelner Merkmale lassen sich durch den Nutzer Entscheidungen zur Anpassung des Montagesystems oder produktspezifischer Merkmale fällen.

Um eine Interpretierbarkeit des Agentenverhaltens zu ermöglichen, muss im Informationsmodell daher die Definition interpretierbarer Merkmale möglich sein. Abbildung 5.15 zeigt die Modellierung der Varianz eines Produkts durch interpretierbare Merkmale wie die Position einer Tasche im Produkt des Testszenarios (vgl. Abbildung 5.3b). Die Definition der Produktvarianz erfolgt über zwei ordinale Merkmale (Quadrat.Pos.X bzw. Quadrat.Pos.Y) zur Beschreibung der möglichen Positionen der quadratischen

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

Tasche zum Einsetzen des quaderförmigen Bauteils und zwei nominalen Merkmalen zur Positionierung der zwei weiteren Taschen (Dreieck- bzw. Kreis-Tasche) im Bauteil. Als weiteres Element der Varianz wird eine variierende relative Position der Bauteile zu Beginn des Prozesses definiert (Quader.Pos.X bzw. Quader.Pos.Y).

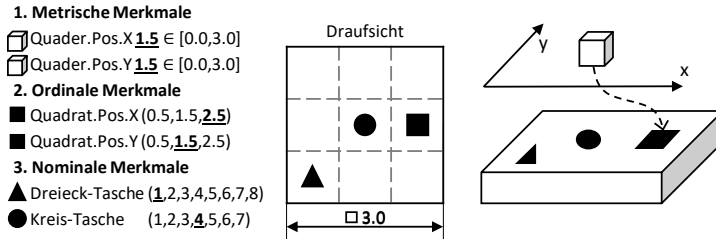


Abbildung 5.15: Definition der Varianz im Testzenario zum Zusammensetzen mit Formatflexibilität nach RÖHLER et al. (2020). Die Anzahl der möglichen Produktvarianten liegt bei  $3 \times 3 \times 8 \times 7 = 504$  (ordinale und nominale Merkmale).

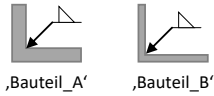
In diesem Beispiel hoher Varianz lässt sich das Agentenverhalten auf Basis einzelner Produktmerkmale interpretieren (z. B. Taktzeit, Fehlerquote bei unterschiedlichen Positionen der quadratischen Tasche). Im Testzenario wird zudem die Zuordnung und Umsetzung von 504 Produktmodellen vermieden. Allerdings erfordert dies eine benutzerdefinierte Implementierung entsprechender Module innerhalb der Simulation. Im Informationsmodell wird daher die Definition benutzerdefinierter Merkmale vorgesehen.

### 5.5.3 Abhängige Merkmale

Neben der Definition interpretierbarer Merkmale stellt der Umgang mit abhängigen Merkmalen eine Herausforderung in der datenbasierten Entscheidungsfindung dar. Abhängigkeiten zwischen den Merkmalen können in der statistischen Auswertung von Daten zu Fehlinterpretationen führen, wenn die kausalen Zusammenhänge nicht berücksichtigt werden (PEARL & MACKENZIE 2018). Die Interpretation von Daten mit abhängigen Merkmalen setzt daher den Einsatz von Methoden der kausalen Inferenz voraus, deren Anwendbarkeit nicht immer gegeben ist und ein derzeitiges Forschungsfeld darstellt (PETERS et al. 2017, SCHÖLKOPF et al. 2021).

Um Entscheidungen auf Basis einer deskriptiven Analyse der Daten zu ermöglichen, wird in der Modellierung der Varianz eine Unabhängigkeit der variierenden Merkmale vorausgesetzt. Da sich unter dieser Voraussetzung nicht alle Montageaufgaben modellieren lassen, werden in dieser Arbeit abhängige Merkmale zu Merkmalsgruppen zusammengefasst.

Abbildung 5.16 zeigt die Kombination der Geometrie des Bauteils und des von der Geometrie abhängigen Prozessparameters "Schweißgeschwindigkeit". Bei der Analyse des Agentenverhaltens lässt sich hier keine direkte Zuordnung einer fehlerhaften Ausführung der Aufgabe durch den Agenten einem spezifischen Merkmal zuordnen, da das Agentenverhalten sowohl von der Geometrie als auch von der Schweißgeschwindigkeit abhängen kann.



Index	Produktmodell [object]	Schweißgeschwindigkeit [m/s]
1	,Bauteil_A'	0.0015
2	,Bauteil_B'	0.0030
...	...	...

Abbildung 5.16: Kombination abhängiger Merkmale am Beispiel des von Produktgeometrie abhängigen Prozessparameters "Schweißgeschwindigkeit"

### 5.5.4 Varianz im Informationsmodell

Wie am Beispiel der Produktvarianz in Abbildung 5.15 dargestellt, lässt die Vielfalt möglicher Modellierungen der Varianz keine allgemeingültige Taxonomie wiederverwendbarer Elemente zu. In dieser Arbeit wird die Möglichkeit der Definition komplexerer Merkmalskombinationen zur Initialisierung der Simulation durch Implementierung entsprechender Simulations-Module durch den Nutzer in der Simulationsumgebung berücksichtigt.

Zur Vereinfachung der Anwendung werden darüber hinaus wiederverwendbare Simulations-Module für Standardmerkmale berücksichtigt. Standardmerkmale unterscheiden sich zu benutzerdefinierten Merkmalen dahingehend, dass diese eine automatische Implementierung innerhalb der simulativen Lernumgebung ermöglichen. Innerhalb der Simulation wird zum Start jeder Episode eine zufällige Kombination der vordefinierten Merkmalsausprägungen gewählt und die Simulationsumgebung entsprechend initialisiert.

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

Neben der Definition einzelner Merkmale, welche zu Beginn einer Episode variieren können, werden in bestehenden Arbeiten teilweise auch komplexere Initialisierungen der Simulationsumgebung vorgenommen. So lässt sich eine ungeordnete Lagerung von Objekten nicht direkt über einzelne Merkmale wie Koordinaten definieren, da Überschneidungen der einzelnen Objekte ausgeschlossen sein müssen (MAHLER et al. 2019). Solch komplexere Modellierungen der Varianz lassen sich für eine Simulationsumgebung allgemeingültig definieren und im Informationsmodell abbilden. Abbildung 5.17 zeigt einen Überblick der Modellierungsmöglichkeiten von Varianz.

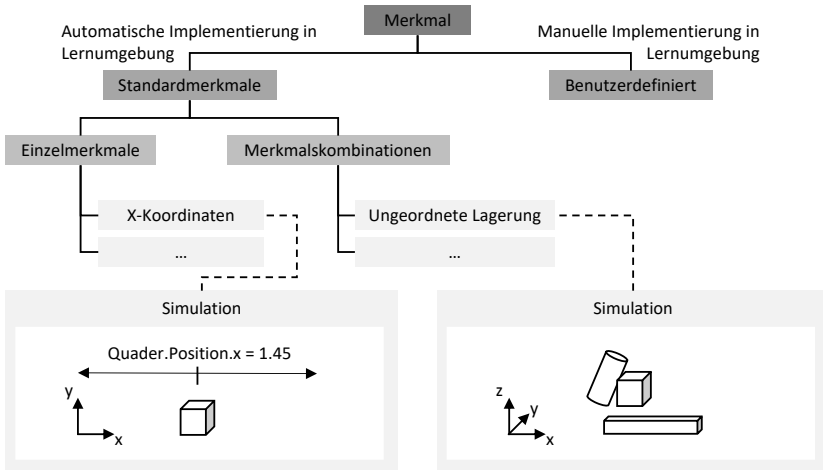


Abbildung 5.17: Modellierung der Varianz über Einzelmerkmale, Merkmalskombinationen und benutzerdefinierte Merkmale

### 5.6 Rahmenbedingungen

Zur Ermittlung der Ähnlichkeit der Aufgaben auf Prozess-Ebene, Skill-Ebene und MDP-Ebene werden Rahmenbedingungen verwendet, um eine spezifischere Definition und Bewertung der Ähnlichkeit von Aufgabenstellungen zu ermöglichen. Die im vergangenen Abschnitt beschriebene Rahmenbedingung "Nicht vollständig wahrnehmbar" stellt ein Beispiel im Bereich Wahrnehmung auf MDP-Ebene dar, welches in jeder Ausprägung einer Montageaufgabe vorkommen kann und für den weiteren Lösungsprozess relevant ist.

Rahmenbedingungen müssen auf jeder Ebene des Aufgabenmodells im Informationsmodell definiert werden. Darüber hinaus besitzen einzelne Elemente des Informationsmodells vordefinierte Rahmenbedingungen. So führt der Agenten-Skill "Führen" durch die Bewegung des Zielpunkts zu einer erhöhten Komplexität des Agentenziels "Pose erreichen", welche durch die Rahmenbedingung "Bewegendes Ziel" im Informationsmodell berücksichtigt wird.

Auf Prozess-Ebene sind zudem Rahmenbedingungen hinsichtlich der Aufgabenstellung relevant, um im Informationsmodell abbilden zukönnen, ob die Notwendigkeit des Einsatzes eines intelligenten Agenten bestehen. So erfordert die Rahmenbedingung "variierende Objektpose" die Wahrnehmung der Umgebung und die Anpassung des Verhaltens im Beispiel eines Handhabungsprozesses.

Analog zu Forschungsergebnissen zur aufgabenorientierten Programmierung im Kontext der Mensch-Roboter-Kooperation von BERG 2020, bei der Aufgaben auf Menschen und Roboter aufgeteilt werden, dienen Rahmenbedingungen somit bereits auf Prozess-Ebene als Grundlage für die frühzeitige Entscheidungsfindung zur Eignung von Agenten-Skills bei der Umsetzung eines Prozesses.

## 5.7 Zusammenfassung

Die in diesem Kapitel beschriebene Erweiterung des Informationsmodells der aufgabenorientierten Programmierung umfasst alle in der Umsetzung eines auf DRL basierenden intelligenten Montagesystems notwendigen Elemente. Zudem beinhaltet das Informationsmodell eine Beschreibung der Aufgabenstellung auf der Prozess-, Skill- und MDP-Ebene, welche zum Abrufen von ähnlichen Aufgabenstellungen und deren Lösungen aus der Datenbasis notwendig sind. Hierzu werden Rahmenbedingungen als semantische Elemente eingesetzt, welche zusätzliche Elemente zur Beschreibung der Aufgabenstellung darstellen.

Durch die Integration dieser Elemente in ein bestehendes Informationsmodell der aufgabenorientierten Programmierung wird eine ganzheitliche Abbildung des Planungsstandes eines Montagesystems gewährleistet. Zudem wird hierdurch die Verwendung der aufgabenorientierten Programmierung für Prozesse bzw. Skills, deren Rahmenbedingungen keine Umsetzung durch einen intelligenten Agenten benötigen, ermöglicht.

## 5 Erweiterung des Informationsmodells automatisierter Montagesysteme

---

Die auf MDP-Ebene definierte Umgebung des Agenten beinhaltet alle Informationen, um eine simulative Lernumgebung implementieren zu können. Hierzu werden die zum Aufbau der Simulation notwendigen Ressourcen und Produkte des Umweltmodells referenziert sowie die zur Wahrnehmung der Umgebung und zur Ausführung von Aktionen notwendigen Schnittstellen des Agenten zu den vorhandenen Ressourcen definiert. Zudem wird die Varianz der Umgebung durch Standardmerkmale modelliert, welche sich in der Simulation durch entsprechende Simulationsmodule umsetzen lassen.

Agentenziele stellen eine Abstraktion von Belohnungsfunktionen im Curriculum dar, welche für unterschiedliche Ausprägungen von Agenten-Skills wiederverwendet werden können. Durch die Parametrisierung der Agentenziele und die Referenzierung der für die Berechnung des Belohnungssignals relevanten Objekte der Umgebung wird die automatisierte Implementierung der Belohnungsfunktion innerhalb der simulativen Lernumgebung ermöglicht.

Auf Agenten-Ebene erfolgt eine Beschreibung der Hyperparameter-Konfiguration des Agenten, die vor der Trainingsphase innerhalb der simulativen Lernumgebung definiert werden muss. Das Ergebnis der Trainingsphase stellt die Strategie des Agenten dar, in welcher die Auswahl der Aktionen zu den wahrgenommenen Zuständen der Umgebung mittels eines künstlichen neuronalen Netzes erfolgt.

Die Unterstützung des Nutzers in der Definition der Skill- und MDP-Ebene und die automatische Konfiguration des Agenten auf der Agenten-Ebene wird in den folgenden Kapiteln erläutert.



## **6 Wissensbasierte Entscheidungsunterstützung**

Die wissensbasierte Entscheidungsunterstützung beruht in dieser Arbeit auf dem Ansatz des fallbasierten Schließens, weshalb im folgenden Abschnitt zunächst die Modellierung der Datenbasis aufgezeigt wird. Anschließend wird der Ansatz zum Abrufen von Lösungen ähnlicher Aufgabenstellungen in Abschnitt 6.2 vorgestellt. Da die Lösungen bestehender Umsetzungen eine Anpassung benötigen können, wird in Abschnitt 6.3 auf die notwendigen manuellen Anpassungsschritte des Nutzers eingegangen.

### **6.1 Modellierung der Datenbasis**

#### **6.1.1 Abhängigkeiten der Planungselemente des Aufgabenmodells**

Die wissensbasierte Entscheidungsunterstützung ausgehend von dem Ansatz des fallbasierten Schließens erfordert die Berechenbarkeit der Ähnlichkeit von Aufgabenstellungen, um geeignete Lösungselemente aus der Datenbasis wiederverwenden zu können. Zunächst erfolgt daher eine Analyse der betrachteten Lösungselemente und deren Abhängigkeiten zu anderen Planungselementen im Aufgabenmodell.

Die Abhängigkeiten unterscheiden sich hinsichtlich harter und weicher Kriterien. Harte Kriterien werden als Ausschlusskriterien zur Vorauswahl relevanter Lösungen genutzt. In den folgenden Tabellen werden diese Abhängigkeiten mit einem "x" gekennzeichnet. Weiche Kriterien stellen dagegen keine Ausschlusskriterien dar, sondern bieten die Möglichkeit, die Ähnlichkeit von Aufgabenstellungen ableiten zu können. Diese Abhängigkeiten werden in den folgenden Tabellen mit einem "(x)" gekennzeichnet.

Ausgangspunkt des Systems ist die Sekundärprozess-Ebene (vgl. Abbildung 4.8), in welcher die Montagesequenz über die erforderlichen Sekundärprozesse definiert ist. Eine Skillsequenz muss den jeweiligen Prozess abbilden, weshalb dies ein hartes Kriterium darstellt.

## 6 Wissensbasierte Entscheidungsunterstützung

Rahmenbedingungen wie die aus der Formflexibilität des Produkts resultierende Rahmenbedingung "variierende Objektgeometrie" stellt ein weiches Kriterium dar (vgl. Tabelle 6.1). So kann die Objektgeometrie beim Prozess "Weitergeben" eine Anpassung der Trajektorie beim Greifen aber auch beim Ablegen des Objekts erfordern. Diese Abstraktion der Aufgabenstellung besitzt daher nicht genügend Informationen, um eine geeignete Skillsequenz und die hiermit verbundene Zuordnung von Agenten-Skills deterministisch ableiten zu können.

Tabelle 6.1: Abhängigkeiten der Skillsequenz von der Prozess-Ebene

Lösung	Prozess-Ebene	
	Prozess	Rahmenbedingungen
Skillsequenz	x	(x)

Die Elemente der MDP-Ebene hängen vom umzusetzenden Agenten-Skill und dessen Rahmenbedingungen ab (vgl. Tabelle 6.2). Die in der Umgebung des Agenten zugeordneten Ressourcen müssen zudem die notwendigen Schnittstellen enthalten, welche in der Wahrnehmung und im Aktionsraum vergangener Fälle verwendet werden.

Tabelle 6.2: Abhängigkeiten von Wahrnehmung und Aktionsraum von der Skill-Ebene und anderen Planungselementen der MDP-Ebene

Lösung	Skill-Ebene		MDP-Ebene	
	Agenten-Skill	Rahmenbedingungen	Aktor-Schnittstellen	Sensor-Schnittstellen
Wahrnehmung	x	(x)		(x)
Aktionsraum	x	(x)	(x)	

Neben der über den Agenten-Skill abstrahierten Aufgabenstellung des Agenten kann das Agentenziel von den eingesetzten Aktionssignalen abhängen. Auch beeinflussen unterschiedliche Arten der Wahrnehmung die Komplexität der Aufgabenstellung und somit die im Curriculum definierte maximale Anzahl an Trainingsschritten. Tabelle 6.3 gibt einen Überblick über die Abhängigkeiten des Curriculums.

Tabelle 6.3: Abhängigkeiten des Curriculums von der Skill-Ebene und anderen Planungselementen der MDP-Ebene

	Skill-Ebene		MDP-Ebene	
Lösung	Agenten-Skill	Rahmenbedingungen	Wahrnehmung	Aktionsraum
Curriculum	x	(x)	(x)	(x)

Die Konfiguration der Hyperparameter des Agenten erfordert kein Prozesswissen des Nutzers, weshalb dieser Schritt in dieser Arbeit automatisiert erfolgt. In Kapitel 7 wird der in dieser Arbeit verwendete Ansatz zur Ableitung der Hyperparameter-Konfiguration anhand der im Aufgabenmodell enthaltenen Informationen vorgestellt. Tabelle 6.4 zeigt die Elemente der MDP-Ebene, welche in der automatischen Konfiguration des Agenten verwendet werden.

Tabelle 6.4: Abhängigkeiten der Hyperparameter auf Agenten-Ebene von MDP-Ebene und Skill-Ebene

	MDP-Ebene			
Lösung	Elementare Ziele	Rahmenbedingungen	Aktions-signale	Sensor-signale
Hyperparameter	(x)	(x)	(x)	(x)

### 6.1.2 Repräsentation der Aufgabenstellungen

Die Repräsentation von Aufgabenstellungen erfolgt nicht durch Auflistung aller Merkmale, welche im Informationsmodell zur Verfügung stehen. Zum Einsatz des fallbasierten Schließens muss die Aufgabenstellung das notwendige Wissen im jeweiligen Kontext repräsentieren und die Informationen zum Ableiten geeigneter Lösungen beinhalten (BEIERLE & KERN-ISBERNER 2019).

Nach RICHTER & WEBER (2013) lassen sich Aufgabenstellungen auf folgende Arten abbilden:

1. *Attribut-Wert-Paare*: Repräsentation der Aufgabenstellung über domänenspezifische Attribute und deren Werte (z. B. Zahlenwerte, Enumerationen)

## 6 Wissensbasierte Entscheidungsunterstützung

2. *Graphen*: Abbildung von Strukturen über ein Set an Knoten und ein Set an Kanten, welche die Relationen zwischen Knoten repräsentieren
3. *Objektorientiert*: Zusammenführung von Attributen in einzelnen Objektklassen und Abbildung der Klassenstruktur über Beziehungen (z. B. Vererbung, Aggregation, Assoziation)

Attribut-Wert-Paare stellen die einfachste Form der Repräsentation dar (vgl. Abbildung 6.1). Sie ermöglichen eine Abbildung von Semantik, indem symbolische Elemente als Attribute (z. B. binäre Angabe vorhandener Elemente, Enumerationen) aufgenommen werden können.

Graphen stellen die komplexeste Repräsentation dar, haben allerdings das größte Potential Wissen abzuspeichern. So können die Knoten unterschiedlichen Klassen zugeordnet werden und eigene Attribute besitzen. Auch Kanten können über eigene Attribute (z. B. Gewichtung) verfügen. Eine vereinfachte Form der Graphen stellen Taxonomien dar, in welchen der Graph eine Baumstruktur besitzt.

Die objektorientierte Modellierung kann wie Taxonomien als vereinfachte Form eines Graphen angesehen werden. Klassen (engl. *Class*) vererben hierbei mögliche Attribute an hiervon abgeleitete Klassen. Instanzen (engl. *Instance*) von Klassen stellen konkrete Objekte einer Klasse dar.

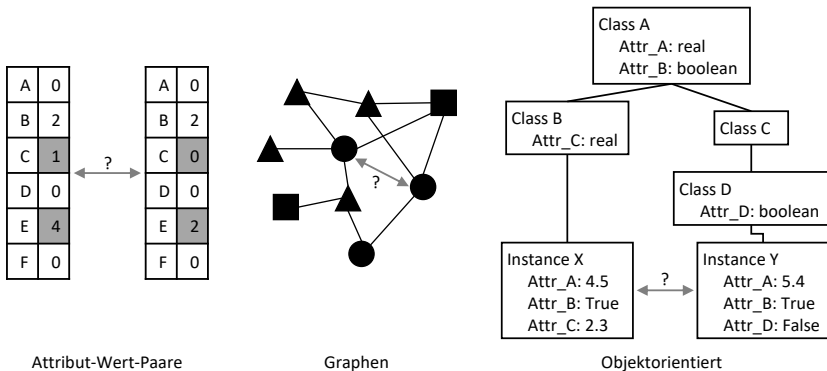


Abbildung 6.1: Betrachtete Arten der Repräsentation von Fällen

Zur Auswahl einer Modellierung der Aufgabenstellungen erfolgt eine Analyse der Methoden-Klassen, welche bei den jeweiligen Repräsentations-Arten zur Berechnung eines Ähnlichkeitsmaßes verwendet werden. Tabelle 6.5 zeigt die Bewertung der Methoden-Klassen, welche unterschiedliche Ausprägungen haben können.

Tabelle 6.5: Ausschlusskriterien der Repräsentationen und Methoden nach der Abbildbarkeit von Semantik (S) und Kardinalität (K)

Repräsentation	Methoden-Klasse	Beispielhafte Ausprägungen	S	K
Attribut-Wert-Paare	Binärer Vergleich	Hamming, Tversky	x	-
	Metrisch	Manhattan, Euklidisch	x	x
Graphen	Maschinelles Lernen	Siamesische Netze	x	x
	Struktur-basiert	Largest Common Subgraph, Graph Edit Distance	x	x
	Graph Embeddings	node2Vec, GraphSAGE	x	x
	Maschinelles Lernen	Graph Neural Networks	x	x
	Taxonomisch	Shortest-Path, Deepest Common Predecessor	x	-
Objekt-orientiert	Kombination aus Attribut-Wert-Paaren + Taxonomisch		x	-

Die im vergangenen Abschnitt betrachteten Abhängigkeiten erfordern neben der Abbildung von *Semantik* auch eine Berücksichtigung von *Kardinalität*, da die Anzahl einzelner semantischer Elemente ebenfalls in die Berechnung der Ähnlichkeit einfließen muss. So spielt beispielsweise die Anzahl der Aktor-Signale eine entscheidende Rolle hinsichtlich der Komplexität der Aufgabenstellung eines Agenten und somit der benötigten Trainingsschritte.

Zur Bewertung der einsatzfähigen Repräsentations-Methoden-Paare werden diese nach allgemeinen Kriterien der Methoden und der Anwendbarkeit der Methoden in der Entscheidungsunterstützung und Hyperparameter-Auswahl analysiert. Abbildung 6.2 gibt einen Überblick über die betrachteten Kriterien und die Bewertung der einzelnen Repräsentationen und Methoden-Klassen.

Um die Unterbrechungen des Planungsprozesses durch die Suche nach ähnlichen Fällen möglichst gering zu halten, stellt die *Effizienz* der Methode ein Kriterium dar. Graph-basierte Methoden erfordern hier teilweise komplexere Algorithmen, welche eine entsprechende Berechnungszeit in Anspruch nehmen.

## 6 Wissensbasierte Entscheidungsunterstützung

Repräsentation	Methoden-Klasse	Kriterien der Methoden				Entscheidungsunterstützung		Hyperparameter	
		EFF	AUS	INT	EXP	ANW	BEW	ANW	BEW
Attribut-Wert-Paare	Metrisch	●	●	●	●	●	<b>0.93</b>	○	0.55
	Maschinelles Lernen	●	●	○	●	○	0.50	●	<b>0.88</b>
Graphen	Struktur-basiert	○	●	●	○	○	0.25	○	0.25
	Graph Embeddings	○	●	●	○	○	0.28	○	0.28
	Maschinelles Lernen	●	●	○	●	○	0.40	●	0.65
	Gewichtung	0.1	0.1	0.1	0.2	0.5		0.5	

Erfüllungsgrad: ○ kaum (0) ○ gering (0.25) ● teilweise (0.5) ● umfangreich (0.75) ● vollständig (1)

Abbildung 6.2: Bewertung (BEW) der Repräsentations-Methoden-Paare nach Effizienz (EFF), Aussagekraft (AUS), Interpretierbarkeit (INT), Expertenwissen (EXP) und Anwendbarkeit (ANW) für die Entscheidungsunterstützung bzw. Hyperparameter-Auswahl

Auch hängt die *Aussagekraft* des Ähnlichkeitsmaßes davon ab, welche Informationen in den Repräsentationen enthalten sein können und wie weit diese Informationen in den Methoden genutzt werden. Graphen bieten im Vergleich zu Attribut-Wert-Paaren mehr Möglichkeiten, um semantische Beziehungen abzubilden.

Maschinelle Lernverfahren bieten allerdings die Möglichkeit, solche Beziehungen selbstständig zu lernen, wenn die entsprechenden Daten vorhanden sind.

Sowohl in der Entscheidungsunterstützung als auch in der Hyperparameter-Auswahl setzt der Einsatz des Systems eine *Interpretierbarkeit* der Ergebnisse der Ähnlichkeits-Berechnung von Fällen voraus. Dies erhöht zum einen die Akzeptanz der Ergebnisse durch den Nutzer und ermöglicht zum anderen eine Anpassung der Aufgabenstellung in einem iterativen Planungsprozess, falls der derzeitige Planungsstand keine geeigneten Lösungen zulässt.

Ein weiteres Kriterium stellt die Unabhängigkeit der Repräsentation vom *Expertenwissen* dar. Gerade in der Domäne von Reinforcement Learning ist geringes Expertenwissen darüber vorhanden, welche Zusammenhänge und semantischen Elemente für die Lösung entscheidend sind. Während Graphen mehr Möglichkeiten zur Abbildung von Expertenwissen und somit eine größere Aussagekraft ermöglichen, hängt deren Anwendbarkeit auch vom vorhandenen Expertenwissen ab.

## **6.2 Wiederverwendung bestehender Lösungselemente innerhalb der Datenbasis**

---

Das wesentliche Kriterium ist die *Anwendbarkeit* der Methoden, weshalb die Bewertung für die Entscheidungsunterstützung und die Hyperparameter-Auswahl getrennt vorgenommen werden. Eine wesentliche Unterscheidung liegt hierbei in der Bewertbarkeit der Ähnlichkeitsmaße zwischen Fällen, welche bei der Hyperparameter-Auswahl durch eine Bewertung des gelernten Agentenverhaltens validiert werden kann.

Da in der Beschreibung der Aufgabenstellungen über Attribut-Wert-Paare am wenigsten Expertenwissen in der Modellierung der Datenbasis notwendig ist, wird in dieser Arbeit diese Art der Repräsentation gewählt.

### **6.2 Wiederverwendung bestehender Lösungselemente innerhalb der Datenbasis**

#### **6.2.1 Kriterien der Metrik zur Berechnung der Ähnlichkeit**

In der Retrieve-Phase des Ansatzes des fallbasierten Schließens erfolgt die Bestimmung der Ähnlichkeit von Aufgabenstellungen in der Datenbasis, um die Lösungen dieser Fälle wiederverwenden zu können. Basierend auf den Attribut-Wert-Paaren der Aufgabenstellungen gilt es daher eine Metrik zu wählen, welche die Ähnlichkeit von Fällen abbildet.

Die Auswahl einer Metrik zur Berechnung der Ähnlichkeit von Aufgabenstellungen erfolgt auf Basis von Domäne-spezifischen Kriterien (RICHTER & WEBER 2013). In dieser Arbeit wird ein Distanzmaß  $d(x,y)$  zur Bestimmung der Ähnlichkeit der Aufgabenstellungen  $x$  und  $y$  gewählt, welches die Kriterien der Reflexivität und Symmetrie erfüllt.

*Reflexivität* setzt voraus, dass die Distanz eines Falls zu sich selbst stets Null ist ( $d(x,x)=0$ ). Dieses Kriterium wird gewählt, um exakte Übereinstimmungen bestehender Fälle zur derzeitigen Aufgabenstellung identifizieren zu können (z. B. Verwendung von gleichen Ressourcen und Agenten-Skills).

*Symmetrie* setzt voraus, dass die Distanz von zwei Fällen unabhängig von deren Integration im Distanzmaß ist ( $d(x,y)=d(y,x)$ ). Somit wird gewährleistet, dass Lösungen komplexerer Aufgabenstellungen (z. B. Prozesse mit mehr Rahmenbedingungen) nicht für einfachere Aufgabenstellungen wiederverwendet werden.

Das Kriterium eines eingeschränkten Wertebereichs (z. B.  $d(x,y) \rightarrow [0,1]$ ) wird in dieser Arbeit nicht aufgestellt, um starke Abweichungen neuer Fälle zur Datenbasis abbilden zu können und somit eine Evaluation der Umsetzbarkeit eines neuen Falls zu ermöglichen. So deutet ein hohes Distanzmaß auf eine signifikante Abweichung der Aufgabenstellung von den vorhandenen Fällen in der Datenbasis hin.

### 6.2.2 Abrufen von Lösungselementen aus der Datenbasis

Zur Auswahl eines Distanzmaßes werden die Kriterien der Metrik und die Kriterien der Methoden (vgl. Abbildung 6.2) herangezogen. In dieser Arbeit betrachtete Metriken, welche die Kriterien der Metriken erfüllen, sind die  $L_1$ -Norm (bzw. Manhattan-Abstand) der Werte von einer Anzahl an  $i$  Attributen der zwei Aufgabenstellungen  $x$  und  $y$

$$d(x,y) = \sum |x(i) - y(i)| \quad (6.1)$$

und die  $L_2$ -Norm (bzw. euklidischer Abstand)

$$d(x,y) = \sqrt{\sum (x(i) - y(i))^2}. \quad (6.2)$$

Die  $L_2$ -Norm besitzt eine höhere Sensibilität bezogen auf hohe Abstände einzelner Attribute. Da der Einfluss jedes Attributs (z. B. Rahmenbedingungen) auf die Ähnlichkeit der Lösungen nicht bekannt ist, kann durch den Einsatz der  $L_2$ -Norm eine Übergewichtung weniger relevanter Attribute erfolgen. Zur Berechnung der Ähnlichkeit von Aufgabenstellungen wird in dieser Arbeit daher die  $L_1$ -Norm verwendet.

In Abschnitt 6.1.1 sind zudem Ausschlusskriterien definiert, welche bei der Auswahl bestehender Aufgabenstellungen zum Vorfiltern der Ergebnisse verwendet werden. Wie in Abbildung 6.3 aufgezeigt, werden die zur Umsetzung eines Prozesses möglichen Skillsequenzen nach ihrer Übereinstimmung hinsichtlich der Rahmenbedingungen mit dem neuen Fall mit der  $L_1$ -Norm bewertet.



## 6.3 Anpassung der Lösungselemente

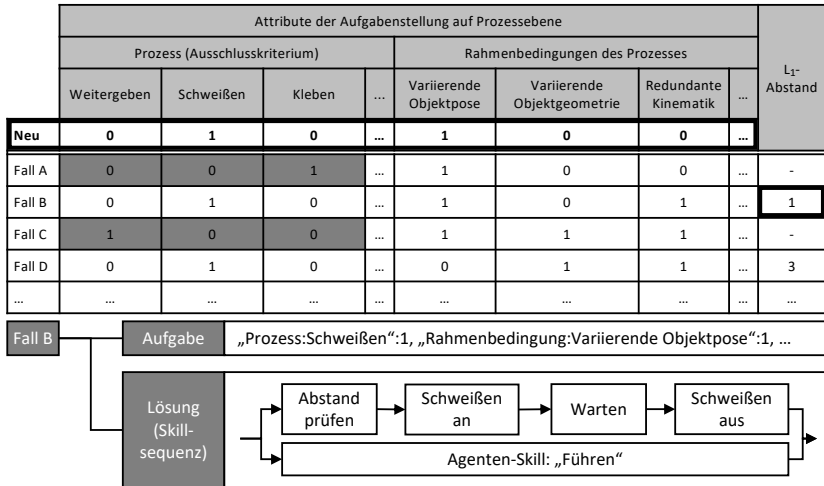


Abbildung 6.3: Ableitung einer Skillsequenz durch Wiederverwendung der Lösungen einer Aufgabenstellung mit ähnlichen Attribut-Wert-Paaren aus der Datenbasis

## 6.3 Anpassung der Lösungselemente

Die Lösungselemente ähnlicher Aufgabenstellungen müssen für die betrachtete Aufgabenstellung adaptiert werden. Abbildung 6.4 zeigt die in den jeweiligen Aufgabenebenen notwendigen manuellen Entscheidungen des Nutzers.

Nach dem Abrufen von Lösungselementen aus der Datenbasis wird im System ein Prüfschritt durch den Nutzer vorgesehen. Erst auf der Abstraktionsebene der Skills kann durch den Nutzer evaluiert werden, welche Schritte eine Anpassung des Verhaltens an die Umgebung und somit einen intelligenten Agenten erfordern. Zudem können die jeweiligen Skills unterschiedliche Rahmenbedingungen besitzen.

So kann sich eine variierende Objektgeometrie nur auf den Greifprozess auswirken und ist daher nicht zwingend als Rahmenbedingung für nachfolgende Skills zu sehen.

Ein Agenten-Skill erfordert die Definition des MDP, welcher eine formale Beschreibung der Lernumgebung darstellt. Welche Objekte des Umweltmodells für die Agentenaufgabe relevant sind, wird durch den Nutzer bestimmt. Auch erfolgt durch den Nutzer die Definition der Varianz.

## 6 Wissensbasierte Entscheidungsunterstützung

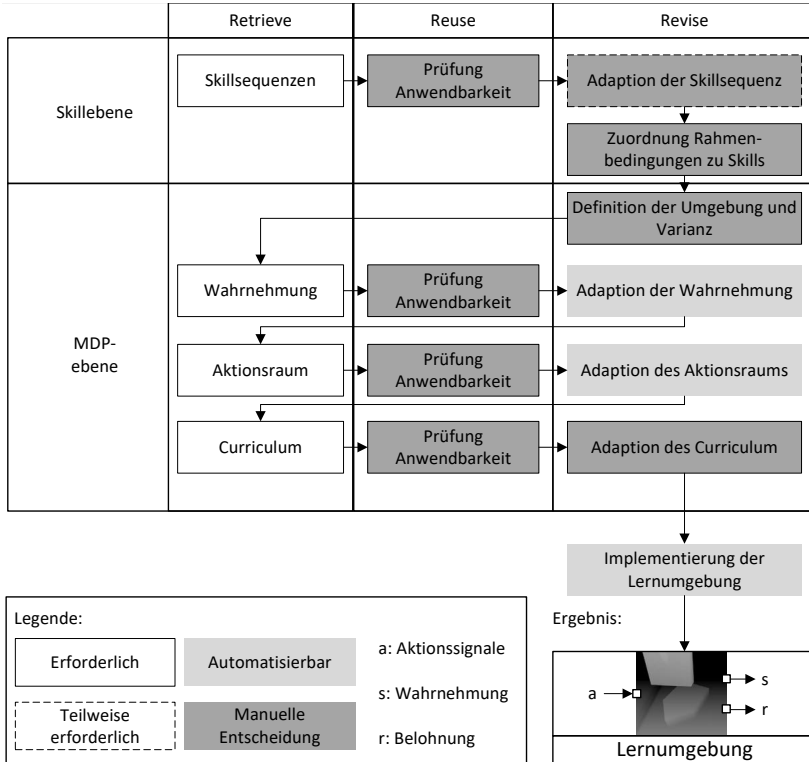


Abbildung 6.4: Manuelle Entscheidungen des Nutzers in der aufgabenorientierten Konfiguration intelligenter Agenten

Die Abhängigkeiten der in der Wahrnehmung und im Aktionsraum definierten Schnittstellen von den Ressourcen der Umgebung werden als weiches Kriterium angesehen. So können bestehende Fälle zur Umsetzung eines Agenten-Skills Aufschluss darüber geben, ob die erforderlichen Ressourcen verfügbar sind. Im Falle geeigneter Agenten-Schnittstellen ist eine automatisierte Zuordnung zu Schnittstellen der Ressourcen möglich.

Da die Optimierung des Curriculum noch ein Forschungsfeld darstellt (PORTELAS et al. 2020) wird eine weitere Anpassung des Curriculum in dieser Arbeit als manuelle Entscheidung des Nutzers im Planungsprozess mit aufgenommen. Um das hierzu notwendige Expertenwissen zu reduzieren, erfolgt durch die Abstraktion der Belohnungsfunktionen über Agentenziele bzw. Teilziele eine Trennung von Prozesswissen (z. B. Definition von maximalem Abstand beim Agentenziel "Positionieren") und DRL-spezifischem Wissen (z. B. Gestaltung der Belohnungsfunktion).

### 6.4 Zusammenfassung

Zur Abbildung einer Aufgabenstellung werden die Elemente des Informationsmodells über Attribut-Wert-Paare abstrahiert. Die Bewertung der Ähnlichkeit durch den  $L_1$ - bzw. Manhattan-Abstand dieser Attribut-Wert-Paare wird verwendet, um Lösungen ähnlicher Aufgabenstellungen abzurufen und die Elemente der Skill-Ebene und MDP-Ebene zu definieren.

Durch den hohen Abstraktionsgrad der Aufgabenbeschreibungen sind nicht alle Informationen zur Auswahl und Anpassung bestehender Lösungen vorhanden. Daher werden manuelle Anpassungen durch den Nutzer benötigt. Die Diskrepanz einzelner Elemente der Aufgabenstellung zu bestehenden Aufgabenstellungen der Datenbasis kann zudem zur Anpassung übergeordneter Ebenen des Aufgabenmodells genutzt werden (z. B. zur Anpassung der verwendeten Sensoren).

Die zur Ableitung der Hyperparameter-Konfiguration auf Agenten-Ebene notwendige Beschreibung der Aufgabenstellung auf MDP-Ebene erfolgt ebenfalls auf Basis von Attribut-Wert-Paaren. Die automatische Konfiguration des Agenten wird in Kapitel 7 beschrieben.



## **7 Automatische Konfiguration des Agenten**

In diesem Kapitel wird das Modul zur automatischen Konfiguration vorgestellt, dessen Aufbau im folgenden Abschnitt erläutert wird. Basierend auf bestehenden Umsetzungen aus der Literatur wird das Erlernen von Ähnlichkeiten hinsichtlich der Lösungen und der Aufgabenstellung des Agenten ermöglicht. Die Lösungen stellen Hyperparameter-Konfigurationen dar, auf welche in Abschnitt 7.2 näher eingegangen wird. Abschnitt 7.3 beschreibt anschließend einen Ansatz zum Erlernen der Ähnlichkeit von Aufgabenstellungen basierend auf Elementen des Informationsmodells. Die hierauf aufbauende Automatisierung der Hyperparameter-Konfiguration und das Erlernen eines Agenten-Skills werden in Abschnitt 7.4 erläutert.

### **7.1 Übersicht des Ansatzes**

Im Gegensatz zur wissensbasierten Entscheidungsunterstützung erfordert die Hyperparameter-Auswahl kein weiteres Prozesswissen über den Montageprozess, weshalb dieses Teilsystem ohne Interaktion mit dem Nutzer umgesetzt wird. Abbildung 7.1 zeigt die Integration des Teilsystems in das Gesamtsystem.

Die notwendigen Eingangsdaten beziehen sich auf die Beschreibung der Agentenaufgabe im Informationsmodell und das simulative Umgebungsmodell, welches ein Teilmodell des Umweltmodells des Montagesystems darstellt.

Im Teilsystem erfolgt eine Optimierung der Hyperparameter-Auswahl basierend auf bestehenden ähnlichen Fällen, welche durch das Informationsmodell auf Agentenebene beschrieben werden. Nach dem Optimierungsprozess muss das Agentenverhalten vom Nutzer validiert werden. Wenn ein gewünschtes Agentenverhalten erreicht wurde, besteht die Möglichkeit, die derzeitige Aufgabe als neuen Fall in die Datenbasis aufzunehmen.

Zum Aufbau einer initialen Datenbasis werden in dieser Arbeit 42 bestehende Fälle aus der Literatur verwendet. Im Gegensatz zur Konzeptionierung des Informationsmodells

## 7 Automatische Konfiguration des Agenten

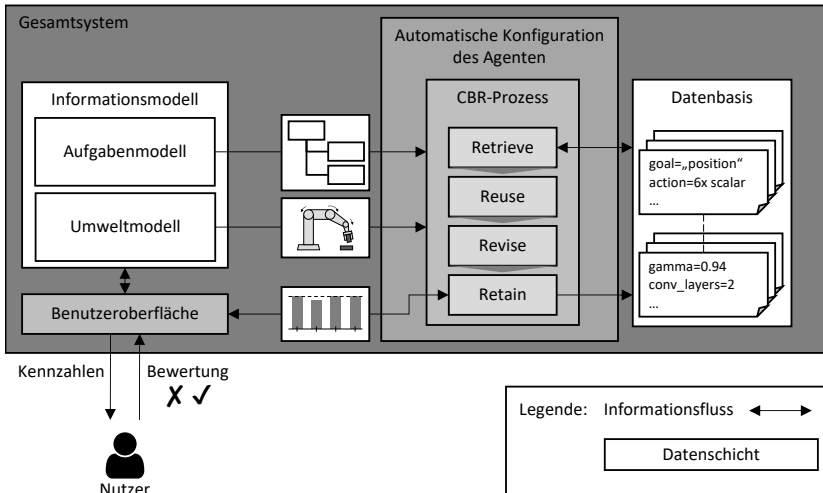


Abbildung 7.1: Integration der automatisierten Hyperparameter-Konfiguration in das Gesamtsystem

aus Kapitel 5 müssen die Fälle in der automatischen Konfiguration eine umfassende Beschreibung der verwendeten Hyperparameter besitzen, welche insbesondere in den Literaturquellen nicht immer gegeben ist. Zudem beschränkt sich diese Arbeit auf den RL-Ansatz PPO, weshalb von den 101 betrachteten Fällen lediglich 42 Fälle zum Aufbau der initialen Datenbasis genutzt werden können.

## 7.2 Ähnlichkeit der Lösungen

### 7.2.1 Modellierung der Lösungen

Auf Agent-Ebene stellt die Hyperparameter-Konfiguration des Agenten zur Parametrierung des Lernalgorithmus und zur Definition der Architektur des künstlichen neuronalen Netzes das Lösungselement dar. Diese Hyperparameter müssen definiert werden, bevor der Agent die Strategie  $\pi(a|s)$  zur Durchführung einer Aufgabe innerhalb der simulativen Lernumgebung erlernen kann (GRAESSER & KENG 2019).

In dieser Arbeit wird der Lernalgorithmus PPO verwendet, weshalb keine Auswahl verschiedener Algorithmen betrachtet wird. Auch reduziert sich der Lösungsraum für

die Parametrierung des Lernalgorithmus auf die Liste der PPO-spezifischen Hyperparameter. Allerdings wird in der Konzeptionierung des Systems kein PPO-spezifisches Wissen in den Lösungsprozess integriert, weshalb von einer Anwendbarkeit auf andere Lernalgorithmen ausgegangen werden kann.

Neben dem Lernalgorithmus stellt die Architektur des künstlichen neuronalen Netzes eine weitere wichtige Komponente des Agenten dar, welche in bestehenden Arbeiten zu Reinforcement Learning oftmals vernachlässigt wird (SINHA et al. 2021). Die Gestaltung der Netzarchitektur lässt sich nicht durch eine Liste von Parametern allgemeingültig abbilden, da unterschiedliche Architekturen eine unterschiedliche Anzahl an Parametern erfordern. So kann die Architektur aus CNN- und RNN-Schichten bestehen, welche jeweils unterschiedliche Parameter besitzen.

Um eine effiziente Optimierung der Architekturen zu ermöglichen, wird der Lösungsraum der Architekturen in bestehenden Arbeiten daher auf Gestaltungsräume reduziert (RADOSAVOVIC et al. 2020). Diese Gestaltungsräume lassen sich über eine finite Anzahl an Hyperparametern modellieren, aus denen sich über vordefinierte Regeln die jeweilige Netzarchitektur ableiten lässt. Durch eine geeignete Wahl des Gestaltungsraums lässt sich der Lösungsraum in der Optimierung daher nicht nur einschränken, sondern auch auf bereits validierte Architektur-Typen reduzieren.

Abbildung 7.2 zeigt die in dieser Arbeit betrachteten Hyperparameter, welche zum Vergleich der Ähnlichkeit der Lösungen in der Hyperparameter-Ebene herangezogen werden. Während die von KENG et al. (2019) konzeptionierte Hyperparameter-Konfiguration im JSON-Format ein hohes Maß an Flexibilität bietet, wird in dieser Arbeit der Lösungsraum auf 11 wesentliche Hyperparameter reduziert.

Die Hyperparameter sind nach dem Kriterium gewählt, dass die wesentlichen Unterscheidungsmerkmale der Hyperparameter-Konfigurationen durch diese Abstraktion abgebildet werden kann. Zudem lassen sich diese Hyperparameter zum Trainieren mit der Bibliothek Unity ML Agents einstellen, welche in dieser Arbeit zur Umsetzung der Anwendungsszenarien eingesetzt wird.

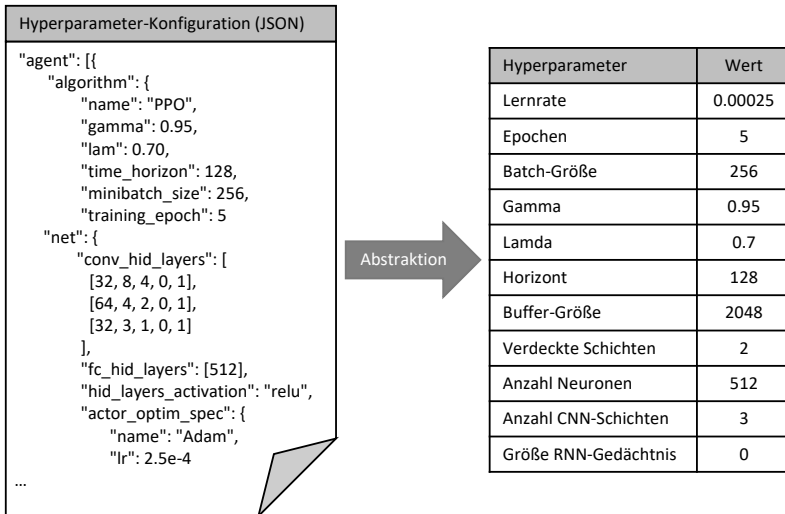


Abbildung 7.2: Abstraktion der möglichen Ausprägungen der Hyperparameter-Konfiguration in Attribut-Wert-Paare

### 7.2.2 Bewertung der Ähnlichkeit von Hyperparameter-Konfigurationen

Die Signifikanz einzelner semantischer Elemente der Aufgabenbeschreibung hinsichtlich der Ähnlichkeit der Hyperparameter-Konfigurationen verschiedener Aufgabenstellungen ist nicht bekannt. Daher wird im verwendeten Meta-Learning-Ansatz die Ähnlichkeit mittels überwachten Lernens basierend auf bestehenden Fällen abgebildet. Der Einsatz überwachten Lernens setzt eine Bewertbarkeit der Vorhersagen des gelernten Modells voraus, welche in dieser Arbeit über die numerische Ähnlichkeit der Hyperparameter-Konfigurationen berechnet wird.

Bestimmte Hyperparameter, wie beispielsweise die Lernrate und die Batch-Größe, können Werte in mehreren Größenordnungen annehmen, weshalb diese in der automatischen Hyperparameter-Konfiguration bestehender Arbeiten über logarithmische Funktionen (z. B. Logarithmieren der Lernrate mit  $\log(0.00025) = -3.6$ ) in den Optimierungsprozess einfließen.



Wie in Abbildung 7.3 beispielhaft dargestellt, treten die Hyperparameter in unterschiedlichen Wertebereichen auf. Um eine einheitliche Bewertung der Ähnlichkeit von Hyperparametern zu ermöglichen, werden die berechneten parametrischen Unterschiede in dieser Arbeit normalisiert.

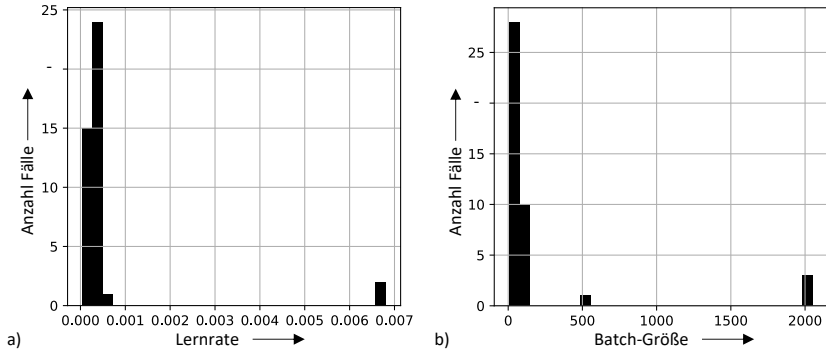


Abbildung 7.3: Beispielhafte Verteilung der Hyperparameter-Werte in der Datenbasis

Die Ähnlichkeit von Hyperparameter-Werten lässt sich über den Abstand einzelner Hyperparameter abbilden. Abbildung 7.4 zeigt die Verteilung der Differenzen für die Lernrate und die Batch-Größe, in welcher die Hyperparameter-Werte logarithmisiert werden und die Differenzen der Hyperparameter-Werte normalisiert sind. Da alle 42 Fälle miteinander verglichen werden, ergeben sich

$$\sum_{i=1}^{42} i = 903 \text{ Fall-Paare}, \quad (7.1)$$

welche zum Trainieren eines ML-Modells eingesetzt werden.

Es besteht eine Abhängigkeit zwischen den Hyperparametern, weshalb die Wiederverwendung einzelner Hyperparameter nicht von der ganzheitlichen Hyperparameter-Konfiguration entkoppelt werden kann. Es existiert ein geringes theoretisches Verständnis hinsichtlich der gegenseitigen Abhängigkeit der Hyperparameter. Zudem können die semantischen Elemente der Aufgabenbeschreibung unterschiedliche Einflüsse auf einzelne Hyperparameter haben.

## 7 Automatische Konfiguration des Agenten

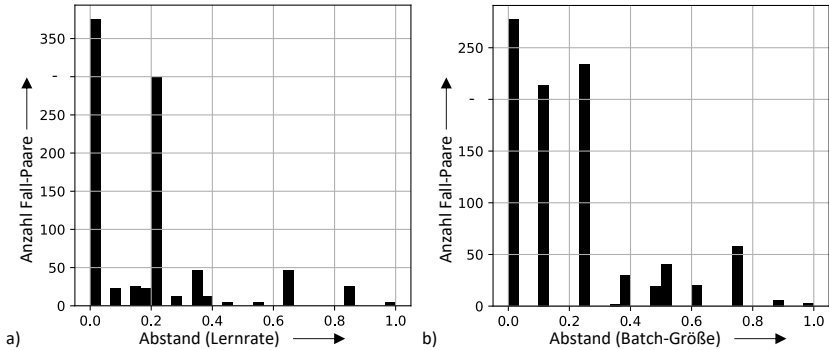


Abbildung 7.4: Differenz der Hyperparameter zwischen verschiedenen Fällen nach Logarithmieren und Normalisierung

Zur Vorhersage der Ähnlichkeit von zwei Fällen wird daher kein einzelnes Bewertungsmaß für die Ähnlichkeit von Hyperparameter-Konfigurationen herangezogen. Stattdessen ist die Zielstellung des maschinellen Lernverfahrens die Vorhersage der Ähnlichkeiten einzelner Hyperparameter der Fälle. Abbildung 7.5 zeigt die Zielstellung des ML-Modells.

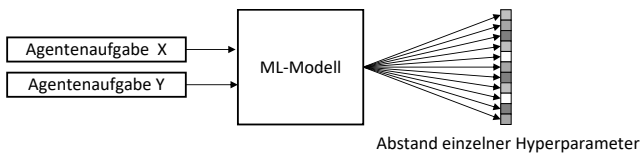


Abbildung 7.5: Zielstellung des ML-Modells zur Vorhersage der Ähnlichkeit von zwei Agentenaufgaben hinsichtlich der verwendeten Hyperparameter

### 7.3 Lernen der Ähnlichkeit von Agentenaufgaben

#### 7.3.1 Meta-Features

Meta-Features bilden die Basis zur Bewertung der Ähnlichkeit von Aufgabenstellungen in Meta-Learning. Während Meta-Features in bestehenden Ansätzen im Kontext des überwachten Lernens durch eine statistische Beschreibung der vorliegenden Datensätze

### 7.3 Lernen der Ähnlichkeit von Agentenaufgaben

abgeleitet werden (vgl. Abschnitt 3.4.3), erfolgt in dieser Arbeit die Bewertung der Ähnlichkeit durch die auf MDP-Ebene definierte Aufgabenstellung des Agenten.

Um die Ähnlichkeit der Aufgabenstellungen bewerten zu können, muss eine geeignete Abstraktion gewählt werden. Da sich Agentenziele aus einer Menge elementarer Ziele zusammensetzen lassen, ermöglichen diese eine geeignete Abstraktion zur Beschreibung der Komplexität der Zielstellung des Agenten.

Allerdings vereinfacht die Terminierung einer Episode bei Erreichung des Zielzustands durch das elementare Ziel "Zielzustand anfahren" die Aufgabenstellung des Agenten, weshalb bei der Ableitung von Meta-Features an dieser Stelle vom Informationsmodell abgewichen wird. Daher wird die Rahmenbedingung "Zielzustand halten" in Abhängigkeit des elementaren Ziels "Zielzustand anfahren" eingeführt ("WENN Anfahren = Falsch DANN Halten = Wahr").

Die Wahrnehmung eines Agenten lässt sich durch die Kardinalität und Art (Skalar bzw. 2D) der Sensorsignale abbilden. Zudem geben die Kardinalität und Art (diskret bzw. kontinuierlich) der Aktionssignale zusätzliche Informationen zur Komplexität der Aufgabenstellung wieder.

Tabelle 7.1 zeigt die in dieser Arbeit verwendeten Meta-Features. Die in der Literatur betrachteten elementaren Ziele *Fixieren*, *Stoßen*, *Schieben* und *Weitere* werden zusätzlich als Meta-Feature mit aufgenommen, um die bestehenden Umsetzungen aus der Literatur abbilden zu können, die keine montagerelevanten Prozesse abbilden.

Tabelle 7.1: Meta-Features der Agentenaufgabe aus der MDP-Ebene

Kategorie	Art	Meta-Feature
Agentenziel	Elementares Ziel	Erreichen
	Elementares Ziel	Orientieren
	(Elementares Ziel)	(Fixieren)
	(Elementares Ziel)	(Stoßen)
	(Elementares Ziel)	(Schieben)
	(Elementares Ziel)	(Weitere)
	Rahmenbedingung	Kollisionsfreiheit
Rahmenbedingung	Zielzustand halten	
Rahmenbedingung	Bewegendes Ziel	
Wahrnehmung	Wahrnehmung	Skalar
	Wahrnehmung	2D bzw. Kamera
	Rahmenbedingung	Nicht vollständig beobachtbar
Aktionsraum	Aktion	Diskret
	Aktion	Kontinuierlich

### 7.3.2 Abbildung der Semantik über Embeddings

Die Beschreibung der Agentenaufgabe enthält die Informationen zur Auswahl geeigneter Hyperparameter. Allerdings ist die Bedeutung einzelner Elemente der Agentenaufgabe hinsichtlich der Hyperparameter-Konfiguration nicht bekannt. Teilweise ist implizites Wissen zum Einfluss einzelner Aufgabenelemente auf die Hyperparameter-Konfiguration vorhanden. Etwa kommen bei kamerabasierten Aufgabenstellungen oftmals CNN-Schichten in der Architektur des künstlichen neuronalen Netzes vor oder es werden RNN-Schichten in dem Fall eingesetzt, dass der Zustand des Systems nicht zu jedem Zeitpunkt vollständig wahrnehmbar ist (BATTAGLIA et al. 2018).

Einzelne Aufgabenelemente können daher einen großen Einfluss auf einzelne Hyperparameter haben. Allerdings ist die Relevanz einzelner Aufgabenelemente nicht bekannt.

Eine Möglichkeit besteht darin, die semantischen Elemente in Form von Attribut-Wert-Paaren als Eingangsgrößen eines ML-Modells zu verwenden, welches die Ähnlichkeit zwischen zwei Aufgaben erlernt. Vergleichbar mit Ansätzen zum maschinellen Lernen basierend auf Text-Daten, lassen sich die semantischen Elemente der Aufgabenstellung auch als sog. *Word Embeddings* abbilden. Ein *Word Embedding* stellt einen Vektor an Fließkommazahlen dar, welcher die semantische Bedeutung eines Wortes abbildet. In Deep-Learning-Ansätzen zum Umgang mit natürlicher Sprache, werden diese *Word Embeddings* dazu genutzt, um die semantische Bedeutung von Wörtern und Sätzen numerisch abzubilden. Doch auch bei tabellarischen Datensätzen mit semantischen Elementen (in diesem Kontext auch "kategorische Variablen" genannt) bieten *Word Embeddings* die Möglichkeit, höhere Vorhersagegüten als eine direkte Verwendung der Attribut-Wert-Paare zu erzielen sowie die Interpretierbarkeit einzelner semantischer Elemente zu ermöglichen (HOWARD et al. 2020).

Im Gegensatz zur natürlichen Sprache besitzen *Word Embeddings* der semantischen Elemente der Agentenaufgabe keine Reihenfolge. Zudem besitzen Wort-Kategorien, die häufig in Texten vorkommen (z. B. Artikel), im Bereich der natürlichen Sprache wenig Informationsgehalt, um Texte voneinander unterscheiden zu können. Bei Agentenaufgaben stellt die Kardinalität einzelner semantischer Elemente dagegen ein wichtiges Unterscheidungskriterium zwischen Fällen dar.

Aufgrund der unterschiedlichen Aufgabenstellungen des Lernverfahrens werden in dieser Arbeit Word Embeddings verwendet, allerdings in der weiteren Konzeptionierung des ML-Modells nicht auf Ergebnisse aus dem Bereich der natürlichen Sprache zurückgegriffen.

Da die Elemente der Agentenaufgabe nur eine Approximation der tatsächlichen Aufgabe darstellen, kann zudem nicht davon ausgegangen werden, dass die Hyperparameter-Konfigurationen ähnlicher Aufgabenstellungen ohne weitere Optimierungsschritte wiederverwendet werden können. Auch basiert die initiale Datenbasis auf Fällen unterschiedlicher Quellen und stellt nicht zwingend eine Sammlung optimaler Lösungen dar.

Diese Unsicherheiten hinsichtlich der Ähnlichkeit von Hyperparameter-Konfigurationen lassen sich durch die Angabe von Konfidenzintervallen in der Vorhersage abbilden. Hierdurch ist eine Rückmeldung an den Nutzer möglich, wenn zu wenig Erfahrung hinsichtlich einer neuen Aufgabenstellung vorhanden sind.

### 7.3.3 Ansatz zum Lernen der Ähnlichkeit

Architekturen künstlicher neuronaler Netze, die Beziehungen wie Ähnlichkeiten zwischen Daten erlernen, indem eine parallele Vorverarbeitung der Daten erfolgt, werden siamesische Netze genannt (RANASINGHE et al. 2019). Die Vorverarbeitung der zu vergleichenden Daten erfolgt auf parallel implementierten Teil-Architekturen, welche ihre Lernparameter teilen, wodurch ein Lernen mit geringeren Datenmengen möglich ist (RANASINGHE et al. 2019).

Das Ergebnis der Vorverarbeitung sind Embeddings, welche die Daten beschreiben (z. B. Embeddings ganzer Sätze durch die Vorverarbeitung der Word Embeddings). Durch eine Metrik zur Berechnung des Abstands mehrdimensionaler Vektoren (z. B. Manhattan Abstand, Euklidischer Abstand, Cosinus) wird anschließend die Ähnlichkeit dieser Embeddings und somit der Eingangsdaten verglichen.

Um die Kardinalität einzelner semantischer Elemente der Agentenaufgabe zu berücksichtigen, werden in dieser Arbeit zunächst Word Embeddings gelernt, welche die Bedeutung einzelner semantischer Elemente der Agentenaufgabe modellieren. Anschließend werden diese Word Embeddings aufsummiert.

Die aus der Addition der Word Embeddings resultierenden Case Embeddings ermöglichen somit eine numerische Repräsentation der Semantik einer Agentenaufgabe, in welcher die Kardinalität berücksichtigt wird.

Da sich die Aspekte der Agentenaufgabe unterschiedlich auf die Wahl der Hyperparameter auswirken können, wird die Ähnlichkeit der Case Embeddings zweier Agentenaufgaben nicht durch ein Abstandsmaß abgebildet. Stattdessen werden die Abstände hinsichtlich einzelner Dimensionen der Case Embeddings berechnet. Für die beispielhaften Case Embeddings mit der Dimension  $m$

$$h_1 = \begin{bmatrix} h_1(1) \\ h_1(2) \\ \vdots \\ h_1(m) \end{bmatrix} = \begin{bmatrix} 0.3 \\ 1.0 \\ \vdots \\ 0.5 \end{bmatrix}, h_2 = \begin{bmatrix} h_2(1) \\ h_2(2) \\ \vdots \\ h_2(m) \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.0 \\ \vdots \\ 0.5 \end{bmatrix} \quad (7.2)$$

ergibt sich ein Vektor

$$|h_1(i) - h_2(i)| = \begin{bmatrix} |h_1(1) - h_2(1)| \\ |h_1(2) - h_2(2)| \\ \vdots \\ |h_1(m) - h_2(m)| \end{bmatrix} = \begin{bmatrix} 0.1 \\ 1.0 \\ \vdots \\ 0.0 \end{bmatrix}, \quad (7.3)$$

welcher die semantische Differenz zwischen Agentenaufgaben auf Basis verschiedener Betrachtungsweisen abbildet (z. B. Unterscheidung hinsichtlich der Wahrnehmung oder Komplexität des Agentenziels).

In einer weiteren Netz-Architektur wird der Zusammenhang zwischen den semantischen Unterschieden der Agentenaufgaben und den Distanzen der einzelnen Hyperparameter gelernt. Hierbei wird ein probabilistischer Ansatz gewählt, welcher neben der Vorhersage der Hyperparameter-Distanzen auch die Unsicherheit der Vorhersage in Form der Standardabweichung der Vorhersagen abbildet. Abbildung 7.6 zeigt den Ansatz zum Erlernen der Ähnlichkeit von Hyperparameter-Konfigurationen durch Vorverarbeitung der semantischen Elemente der Agentenaufgabe mittels lernbarer Embeddings.

### 7.3 Lernen der Ähnlichkeit von Agentenaufgaben

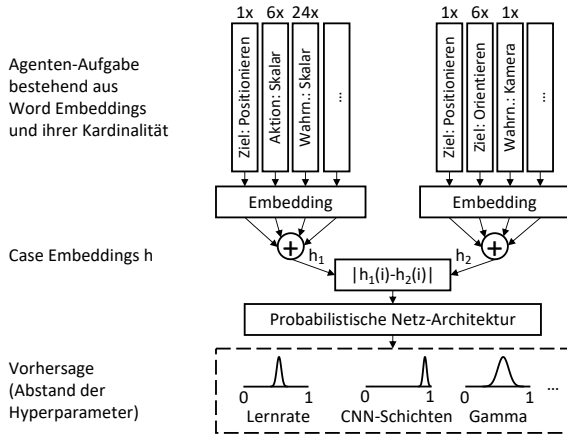


Abbildung 7.6: Zum Lernen von Case Embeddings und Hyperparameter-Ähnlichkeiten verwendete Architektur nach RÖHLER & SCHILP (2022)

#### 7.3.4 Wahl der Architektur des Ansatzes

Zum Lernen der Case Embeddings werden die Attribut-Wert-Paare der Agentenaufgaben der 903 Fall-Paare verwendet. Der Lernalgorithmus zum Erlernen der Case Embeddings besitzt ebenfalls Hyperparameter, welche basierend auf der Datenbasis optimiert werden. Die Fall-Paare werden aufgeteilt in 767 Fall-Paare (85 % der Datenbasis) zum Trainieren des Modells und 136 Fall-Paare (15 % der Datenbasis) zur Validierung des gelernten Modells.

Zur Umsetzung der probabilistischen Netz-Architektur wird *Monte Carlo Dropout* (MCD) eingesetzt. In MCD wird ein Ensemble an Modellen trainiert, aus deren einzelnen Vorhersagen ein Durchschnittswert und eine Standardabweichung abgeleitet werden können. Dies erfolgt durch das zufällige Ausschalten einzelner Neuronen in den verdeckten Schichten des künstlichen neuronalen Netzes. Diese Wahrscheinlichkeit zum Ausschalten eines Neuronen wird als Dropout bezeichnet.

Die Vorhersagen der probabilistischen Modelle werden mittels einer Kostenfunktion basierend auf *Negative Log-Likelihood* (NLL) bewertet. Hierbei wird für jede Vorhersage der Ähnlichkeit eines Hyperparameters bei einem Fall-Paar die Wahrscheinlichkeit der tatsächlichen Ähnlichkeit basierend auf der Wahrscheinlichkeitsverteilung der Vorhersage berechnet.

## 7 Automatische Konfiguration des Agenten

---

Der negative Logarithmus dieser Wahrscheinlichkeit besitzt einen Wert von 0 bei einer Wahrscheinlichkeit von 1 bzw. 100 % ( $-\log(1)=0$ ) und geht gegen  $\infty$ , je mehr sich die Wahrscheinlichkeit der richtigen Vorhersage dem Wert 0 nähert.

Die Dimension der Embeddings stellt ein Maß für die unterschiedlichen semantischen Ausprägungen dar, welche ein Embedding annehmen kann. Im Bereich der natürlichen Sprache können diverse Ausprägungen vorkommen (z. B. positiv bzw. negativ, weiblich bzw. männlich), weshalb Embeddings hier hunderte bis tausende Dimensionen haben.

Bei der im Vergleich zur natürlichen Sprache geringen Komplexität der Agentenaufgaben kann von einer weitaus geringeren Anzahl an Dimensionen ausgegangen werden. Die Dimension lässt sich in Abhängigkeit der 14 betrachteten semantischen Elemente (vgl. Tabelle 7.1) mit der Heuristik nach LAKSHMANAN et al. (2020) zu

$$1.6\sqrt{m} = 1.6\sqrt{14} \approx 6 \text{ Dimensionen} \quad (7.4)$$

abschätzen. In der Optimierung werden daher Dimensionen in dieser Größenordnung betrachtet. Vor dem Start des Trainings müssen zudem die Werte der Embeddings in vordefinierten Wertebereichen initialisiert werden. Diese minimalen und maximalen Werte werden ebenfalls in der Optimierung berücksichtigt.

Vergleichbar mit den Hyperparametern in Reinforcement Learning, wird auch hier die Architektur des künstlichen neuronalen Netzes durch die Anzahl der verdeckten Schichten und die Anzahl der Neuronen pro Schicht bestimmt, während die Umsetzung des Gradientenverfahrens über die Lernrate definiert wird.

Zum Finden der Hyperparameter-Konfiguration mit der höchsten Vorhersagegüte (d. h. mit kleinstem NLL-Wert im Validierungs-Datensatz) werden 100 ML-Modelle basierend auf einer zufälligen Auswahl der Hyperparameter trainiert und die Hyperparameter-Konfiguration mit dem geringsten NLL-Wert ausgewählt (sog. *Random Search*). Tabelle 7.2 zeigt die verwendeten Hyperparameter und deren Wertebereiche. Jedes Modell wurde mit einer Anzahl von 500 Epochen und einer Batch-Größe von 32 trainiert.

Mit einer Dimension der Embeddings von 10 wurde die höchste Vorhersagegüte mit einem NLL-Wert von 0.85 ermittelt. Bei den 11 betrachteten Hyperparametern bedeutet dies, dass die Vorhersage einzelner Hyperparameter-Abstände im Durchschnitt bei  $10^{-0.85/11} \simeq 83,7\%$  der Erwartung des probabilistischen Modells liegt.



## 7.3 Lernen der Ähnlichkeit von Agentenaufgaben

Tabelle 7.2: In der Optimierung betrachtete Wertebereiche der Hyperparameter

Hyperparameter	Wertebereich	Ergebnis
Dimension der Embeddings	[4, 6, 8, 10, 12, 14, 16]	10
Min. Initialisierung der Embeddings	[0.00001, 0.0001, 0.001]	0.0001
Max. Initialisierung der Embeddings	[0.001, 0.01, 0.1, 1]	0.01
Anzahl verdeckter Schichten	[1, 2, 3, 4, 5]	2
Anzahl Einheiten pro Schicht	[8, 16, 32, 64, 128, 256, 512]	256, 512
Lernrate	[0.0001, 0.001, 0.01]	0.001
Dropout	[0.1, 0.2, 0.3, 0.5, 0.8, 0.9]	0.9, 0.2
NLL (Validierung)		<b>0.85</b>

### 7.3.5 Analyse der Interpretierbarkeit der Case Embeddings

Die gelernten Case Embeddings sind aufgrund ihrer Dimensionalität von 10 nicht visualisierbar. Die Hauptkomponentenanalyse stellt ein Verfahren zur Dimensionsreduktion dar, bei dem eine lineare Transformation mehrdimensionaler Daten gesucht wird, in welcher die Varianz der Daten maximal gehalten wird. Hierdurch lassen sich die Case Embeddings auf eine zweidimensionale Darstellung projizieren, in welcher die Abstände aus dem 10-dimensionalen Raum übertragen werden. Abbildung 7.7 zeigt das Ergebnis der Hauptkomponentenanalyse der 42 Fälle der Datenbasis. Die beiden Hauptkomponenten repräsentieren die Koordinaten der 10-dimensionalen Vektoren nach der Transformation im zweidimensionalen Raum.

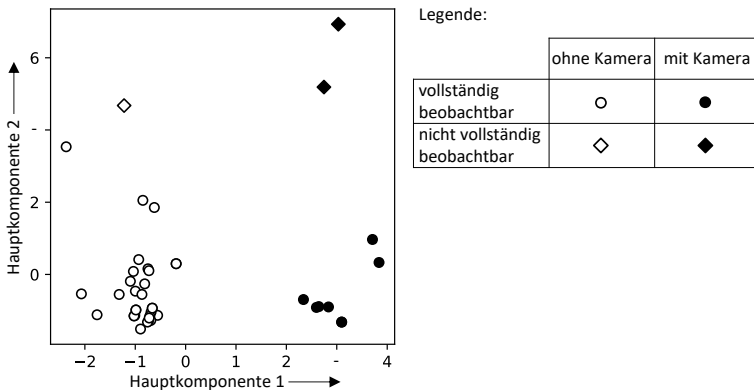


Abbildung 7.7: Ergebnis der Hauptkomponentenanalyse zur Visualisierung der erlernten 10-dimensionalen Case Embeddings. Jedes Case Embedding stellt einen der 42 Fälle der Datenbasis dar.

## 7 Automatische Konfiguration des Agenten

In der Abbildung wird deutlich, dass Agentenaufgaben mit unterschiedlichen Arten der Wahrnehmung größere Abstände bezüglich ihrer Case Embeddings aufweisen. Dieses Ergebnis lässt sich dadurch erklären, dass die Art der Wahrnehmung eine hohe Relevanz bei der Auswahl der Netzarchitektur besitzt. Zur genaueren Interpretation des Einflusses einzelner semantischer Elemente der Agentenaufgabe können die einzelnen Word Embeddings der semantischen Elemente betrachtet werden.

Da es sich bei der Hauptkomponentenanalyse um eine lineare Transformation handelt und die Zusammensetzung der Word Embeddings zu Case Embeddings durch Addition erfolgt, lassen sich auch die Hauptkomponenten der Case Embeddings als eine Addition der Hauptkomponenten der Word Embeddings darstellen. Abbildung 7.8 zeigt die einzelnen Word Embeddings projiziert auf die Hauptkomponenten der Case Embeddings.

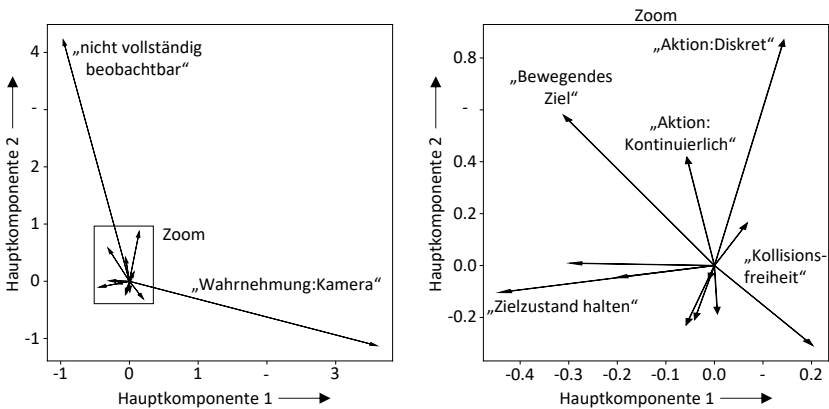


Abbildung 7.8: Darstellung der einzelnen Word Embeddings. Word Embeddings mit einem geringeren Einfluss werden auf der rechten Seite vergrößert dargestellt.

Die Word Embeddings der Wahrnehmung über Kamerabilder und der nicht vollständigen Beobachtbarkeit der Umgebung haben den größten Einfluss auf die Ausprägung der Case Embeddings, welche durch die Norm der Vektoren abgebildet wird. Da diese semantischen Elemente einen direkten Bezug zu den Hyperparametern (Anzahl der CNN-Schichten, Größe des RNN-Speichers) haben, kann erwartet werden, dass diese hohe Korrelation auch in den gelernten Word Embeddings abgebildet wird.

### 7.3 Lernen der Ähnlichkeit von Agentenaufgaben

Wie in Tabelle 7.3 aufgelistet, wird den semantischen Elementen der Wahrnehmung und des Aktionsraums des Agenten die größte semantische Bedeutung zugeordnet. Als Metrik zur Bewertung der semantischen Bedeutung wird die  $L_1$ -Norm der Word Embeddings gewählt. Da die Zusammensetzung der Case Embeddings über die Addition der Word Embeddings der semantischen Elemente erfolgt, lässt sich der maximale Einfluss ebenfalls über die Multiplikation der maximalen Kardinalität eines semantischen Elements innerhalb der Datenbasis mit der  $L_1$ -Norm eines Word Embeddings abbilden.

Tabelle 7.3: Word Embeddings sortiert nach ihrem maximalen Einfluss auf die Case Embeddings ( $L_1$ -Norm  $\times$  Kardinalität)

Kategorie	Word Embedding	$L_1$ -Norm	$L_1$ -Norm (PCA)	Max. Kardinalität	Max. Einfluss
Wahrnehmung	Kamera	3.63	3.60	4	<b>14.54</b>
Aktionsraum	Kontinuierlich	0.51	0.42	20	<b>10.18</b>
Aktionsraum	Diskret	1.00	0.87	6	<b>6.01</b>
Wahrnehmung	Nicht vollständig beobachtbar	5.08	4.22	1	<b>5.08</b>
Wahrnehmung	Skalar	0.07	0.05	70	<b>4.55</b>
Rahmenbedingung	Zielzustand halten	2.20	0.44	1	<b>2.20</b>
Rahmenbedingung	Bewegendes Ziel	1.24	0.58	1	<b>1.24</b>
Elementarziel	Fixieren	1.01	0.20	1	<b>1.01</b>
Elementarziel	Stoßen	0.88	0.23	1	<b>0.88</b>
Elementarziel	Pick&Place	0.87	0.19	1	<b>0.87</b>
Elementarziel	Schieben	0.86	0.21	1	<b>0.86</b>
Elementarziel	Orientieren	0.37	0.16	1	<b>0.37</b>
Rahmenbedingung	Kollisionsfreiheit	0.36	0.31	1	<b>0.36</b>
Elementarziel	Erreichen	0.30	0.30	1	<b>0.30</b>

Beim Vergleich der  $L_1$ -Norm der 10-dimensionalen Word Embeddings und der Norm der Hauptkomponenten " $L_1$ -Norm (PCA)" zeigt sich, dass die Hauptkomponenten die semantischen Bedeutungen einzelner Elemente nicht vollständig abbilden. Durch die Hauptkomponentenanalyse wird eine Projektion gelernt, in welcher die Abstände der Case Embeddings basierend auf der  $L_1$ -Norm im Verhältnis möglichst beibehalten werden. Allerdings führt die Dimensionsreduktion auch zu einer Reduktion des Informationsgehalts, da Dimensionen, welche stärker mit anderen Dimensionen korrelieren, weniger Einfluss zukommt.

So zeigt sich, dass der Rahmenbedingung der Kollisionfreiheit nach der Hauptkomponentenanalyse eine verhältnismäßig höhere Bedeutung zukommt als über die Case

Embeddings gelernt wurde. Abbildung A.5 zeigt das Ergebnis einer Hauptkomponentenanalyse, in welcher die unterschiedlichen Arten der Wahrnehmung nicht direkt aus den Hauptkomponenten ablesbar sind, wohingegen eine eindeutige Vorhersage der Art der Wahrnehmung basierend auf einer logistischen Regression der Elemente der Case Embeddings möglich ist.

Der Einsatz der Word Embeddings ermöglicht somit eine Aussage darüber, was die Ähnlichkeit von Agentenaufgaben ausmacht. Es ist auch möglich, Agentenaufgaben auf diese Weise anzupassen, um eine höhere Ähnlichkeit zu bestehenden Fällen zu erreichen. Eine Automatisierung dieses Vorgangs wird in dieser Arbeit nicht betrachtet. Allerdings zeigen die Ergebnisse, dass in diesem Fall nicht nur die absoluten Abstände, sondern auch die Abstände innerhalb einzelner Dimensionen zwischen zwei Case Embeddings berücksichtigt werden müssen.

### 7.3.6 Analyse des Informationsgehalts der Case Embeddings

Ob die Beschreibung der Agentenaufgabe über semantische Elemente ausreicht, um auf einzelne Hyperparameter schließen zu lassen, lässt sich anhand der Vorhersagegüte des gelernten ML-Modells beurteilen. Wie in Abbildung 7.9a dargestellt, zeigt sich am Beispiel der Lernrate, dass die Ähnlichkeiten der Lernraten mit einer höheren Unsicherheit verbunden sind als die Ähnlichkeiten der Anzahl an CNN-Schichten (Abbildung 7.9b).

Abgebildet sind die Wahrscheinlichkeiten, inwieweit sich zwei Agentenaufgaben trotz der selben semantischen Elemente ( $|h_1(i) - h_2(i)| = 0$ , vgl. Gleichung 7.3) hinsichtlich der einzelnen Hyperparameter unterscheiden können. In Abbildung 7.9 sind zudem die parametrischen Unterschiede der Hyperparameter-Konfigurationen in den Bibliotheken Unity ML Agents und SLM-Lab abgebildet, in welchen für dieselbe Aufgabe (Unity Reacher-v0) unterschiedliche Hyperparameter gewählt wurden. So wird in Unity ML Agents eine Lernrate von 0.0003 verwendet, während in SLM-Lab für selbige Lernumgebung eine Lernrate von 0.0007 eingesetzt wird. Aufgrund der Logarithmierung ergibt sich hierfür in der Datenbasis ein normierter Abstand von 0.17. Die unterschiedliche Wahl der Lernrate durch Experten führt daher auch zu einer größeren Unsicherheit hinsichtlich des Zusammenhangs zwischen Agentenaufgabe und Lernrate.

Die Wahl, CNN-Schichten zur Vorverarbeitung von Kamerabildern einzusetzen, lässt sich dagegen einfacher von der Beschreibung der Agentenaufgabe ableiten. Da die

### 7.3 Lernen der Ähnlichkeit von Agentenaufgaben

betrachtete Lernumgebung keine Wahrnehmung über Kamerabilder besitzt, werden in beiden berücksichtigten Hyperparameter-Konfigurationen keine CNN-Schichten verwendet. Da nur ganzzahlige Werte angenommen werden können, lässt sich mit einer Wahrscheinlichkeit von 76 % sagen, dass die Anzahl der CNN-Schichten gleich ist. Die Wahl unterschiedlicher CNN-Schichten in Fällen mit kamerabasierter Wahrnehmung kann dazu führen, dass dennoch eine gewisse Unsicherheit vorherrscht (Mittelwert=0.05, Standardabweichung=0.31).

Wie in Abschnitt 3.4.1 beschrieben, ist eine Streuung valider Hyperparameter-Werte in DRL gegeben, wodurch eine gewisse Unschärfe in der Vorhersage der Hyperparameter unvermeidbar ist. Um die Anwendbarkeit des Ansatzes genauer zu untersuchen, wird daher eine Evaluation in der Optimierung von Hyperparametern in drei Anwendungsszenarien in Kapitel 9 durchgeführt.

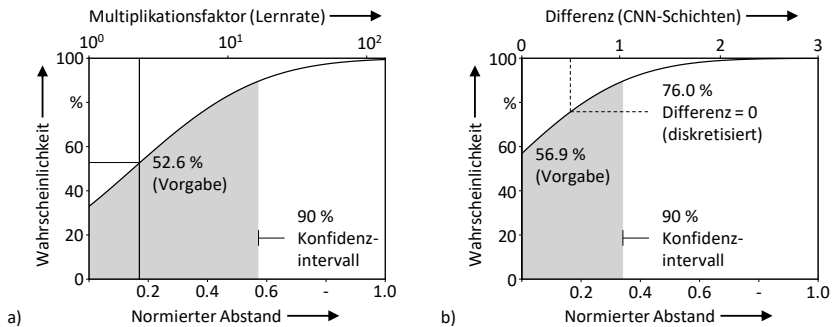


Abbildung 7.9: Streuung der Vorhersage der parametrischen Ähnlichkeit bei derselben Agentenaufgabe mit unterschiedlicher Wahl der Hyperparameter:  
a) Unterschied der Lernrate (0.17 in Datenbasis)  
b) Unterschied der CNN-Schichten (0.0 in Datenbasis)

Die Dimensionalität der Embeddings stellt ein Maß dafür dar, welche unterschiedlichen semantischen Dimensionen die Beschreibung einer Agentenaufgabe besitzen kann. Zur Untersuchung des Einflusses der Dimensionalität der Embeddings wurden jeweils 100 Versuchsdurchführungen mit Dimensionen von 1 (Beschreibung als ein skalarer Wert) bis 10 (mittels Random Search gefundene Dimensionalität, vgl. Abschnitt 7.3.4) gelernt. Ein geringerer Wert der Kostenfunktion (Negative Log-Likelihood) stellt einen höheren Informationsgehalt der Embeddings dar. Wie in Abbildung 7.10 dargestellt, nimmt der Wert der Kostenfunktion in der Vorhersage der parametrischen Ähnlichkeit der

## 7 Automatische Konfiguration des Agenten

Hyperparameter mit steigender Dimensionalität ab und der Informationsgehalt nimmt somit zu.

Für den Nutzer darstellbare Dimensionen von 1 bis 3 haben zu wenig Aussagekraft, um die Zusammenhänge innerhalb der Trainingsdaten (7.10a) abzubilden. Die Generalisierbarkeit der gelernten Zusammenhänge wird über die Kostenfunktion basierend auf den Validierungsdaten geprüft. Hier zeigt sich, dass die Kostenfunktion und somit der Informationsgehalt erst ab einer Dimension von 6 stagnieren. Die Daumenregel aus Gleichung 7.4 scheint in diesem Zusammenhang daher eine valide Regel zur Abschätzung der notwendigen Dimensionen. Bei einer Dimensionalität von 10 zeigt sich zudem, dass die Vorhersagegüte auf den Validierungsdaten stark variieren kann. Eine mögliche Ursache stellt die Überanpassung an die Trainingsdaten dar, welche durch höhere Dimensionalitäten der Embeddings begünstigt wird.

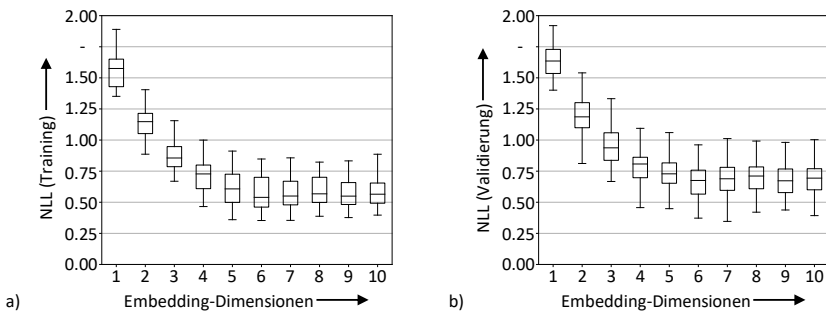


Abbildung 7.10: Verteilung des Negative Log Likelihood mit verschiedenen Dimensionen der Embeddings bei jeweils 100 Versuchsdurchführungen:  
a) Trainingsdaten b) Validierungsdaten. Die Box-Plots zeigen den Median, die Quartile (50 %-Bereich) und die Extremwerte.

## 7.4 Module des fallbasierten Schließens

### 7.4.1 Retrieve-Phase

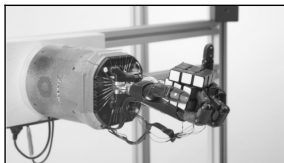
Die in Abschnitt 7.3 vorgestellte Repräsentation von Agentenaufgaben wird in der Retrieve-Phase zum Abrufen ähnlicher Agentenaufgaben und der zugehörigen Hyperparameter-Konfigurationen verwendet. Die  $L_1$ -Norm der Differenz von zwei Case Embeddings ( $L_1$ -Abstand) gibt ein Maß für die Ähnlichkeit von Agentenaufgaben an.

Agentenaufgaben mit den geringsten  $L_1$ -Abständen zur neuen Agentenaufgabe werden zur Konfiguration des Agenten weiterverwendet.

Die automatische Konfiguration von Agenten beschränkt sich in dieser Arbeit auf Agentenaufgaben, die sich durch die semantischen Elemente bestehender Agentenaufgaben abbilden lassen und nahe an existierenden Fällen liegen. Die Bewertung der Umsetzbarkeit erfolgt über das Bewertungsmaß

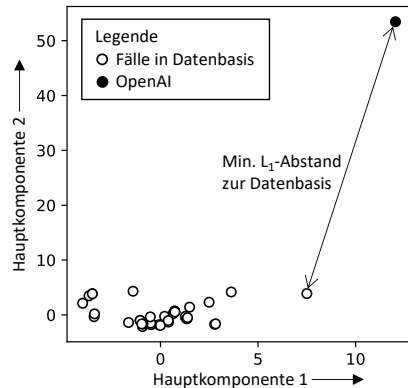
$$\text{Abstand zur Datenbasis} = \frac{\text{Minimaler } L_1\text{-Abstand zur Datenbasis}}{\text{Maximaler } L_1\text{-Abstand in Datenbasis}}, \quad (7.5)$$

welches aussagt, wie weit die neue Agentenaufgabe von den bestehenden Fällen entfernt ist. Wie in Abbildung 7.11 dargestellt, lässt sich somit ein Abbruchkriterium definieren (z. B. Abstand zur Datenbasis  $> 0.5$ ). Die in OPENAI et al. (2019) umgesetzte Agentenaufgabe besitzt aufgrund der 20 anzusteuern Achsen mit jeweils 11 möglichen Achsgeschwindigkeiten eine hohe Komplexität, welche in der bestehenden Datenbasis nicht abgebildet wird. Durch einen hohen Abstand zur Datenbasis (hier ein Wert von 5.5) kann somit eine Aussage hinsichtlich der Umsetzbarkeit der Agentenaufgabe auf Basis der bestehenden Fälle getroffen werden.



Agentenaufgabe	
Orientieren	1
Nicht vollst. beobachtbar	1
Wahrnehmung:Skalar	50
Aktion:Diskret	220

a)



b)

Abbildung 7.11: Bewertung neuer Aufgaben hinsichtlich ihrer Umsetzbarkeit im System. a) Abstraktion der von OPENAI et al. (2019) umgesetzten Handhabungsaufgabe b) Vergleich der neuen Aufgabe mit Fällen aus bestehender Datenbasis

### 7.4.2 Reuse- und Revise-Phase

Die in der Retrieve-Phase abgerufenen Hyperparameter-Konfigurationen ähnlicher Agentenaufgaben werden in der Reuse-Phase zum Trainieren des Agenten innerhalb der simulativen Lernumgebung eingesetzt. Da RL-Algorithmen eine zunehmende Robustheit hinsichtlich der Wahl der Hyperparameter besitzen (SCHULMAN et al. 2017), erhöht sich mit fortschreitenden Forschungsarbeiten im Bereich von Reinforcement Learning auch die Wahrscheinlichkeit, dass keine weitere Anpassung der Hyperparameter-Konfigurationen notwendig ist.

Von diesem Umstand kann allerdings bei neuen Fällen nicht zwingend ausgegangen werden, weshalb in der Revise-Phase eine weitere Anpassung der Hyperparameter mittels einer Bayesschen Optimierung vorgenommen wird (vgl. Abschnitt 3.4.2).

Wie in Tabelle 7.4 dargestellt lässt die bestehende Datenbasis in dem Ansatz dieser Arbeit auch bei exakter Übereinstimmung der Agentenaufgaben noch keine ausreichende Reduktion des Lösungsraums ausgehend von typischen Hyperparameter-Werten zu. Für die weitere Optimierung wird der Lösungsraum daher nicht eingeschränkt.

Tabelle 7.4: Einschränkung des Lösungsraums typischer Werte (aus Unity ML Agents übernommene Extremwerte sind fett gedruckt)

Hyperparameter	Min	Max	90 %-Konfidenzintervall im Vergleich zum gesamten Hyperparameter-Bereich
Gamma	<b>0.8</b>	0.999	-30.2%
Lambda	0.5	0.99944	35.1%
Lernrate	<b>0.00001</b>	0.0068	-8.9%
Epochen	3	30	17.3%
Horizont	3	2048	15.1%
Buffer Größe	256	<b>409600</b>	-34.1%
Batch Größe	8	<b>5120</b>	-7.5%
Anzahl Neuronen	20	512	4.9%
Größe RNN-Speicher	0	<b>5120</b>	-44.4%
Verdeckte Schichten	1	3	-4.0%
Anzahl CNN-Schichten	0	3	-7.8%

Aus der Initialisierung der Bayesschen Optimierung durch wiederverwendete Hyperparameter-Konfigurationen resultiert allerdings ein Gauß-Prozess, in welchem naheliegende Hyperparameter-Konfigurationen während der Optimierung höher priorisiert werden (siehe Abbildung 7.12).



Da einzelne Iterationen in Abhängigkeit der Komplexität der Aufgabenstellung Stunden bzw. Tage dauern können (im Fall von OPENAI et al. (2019) mehrere Wochen), wird im System die Definition von Abbruchkriterien hinsichtlich der erreichten Belohnung während der Trainingsphase ermöglicht. Der Trainingsverlauf wiederverwendeter Hyperparameter-Konfigurationen kann dem Nutzer als Anhaltspunkt zur Definition von Abbruchkriterien dienen. Eine automatische Definition der optionalen Abbruchkriterien wird im System nicht vorgesehen. Wie das Beispiel des Testszenarios zum Zusammenetzen unter Produktvarianz zeigt (siehe Abbildung A.4), steigt das Belohnungssignal während der Trainingsphase nicht zwingend kontinuierlich, weshalb eine Definition trivialer Abbruchkriterien (z. B. Belohnung kleiner 50 % der maximalen Belohnung bei 50 % der Trainingszeit) nicht anwendbar ist.

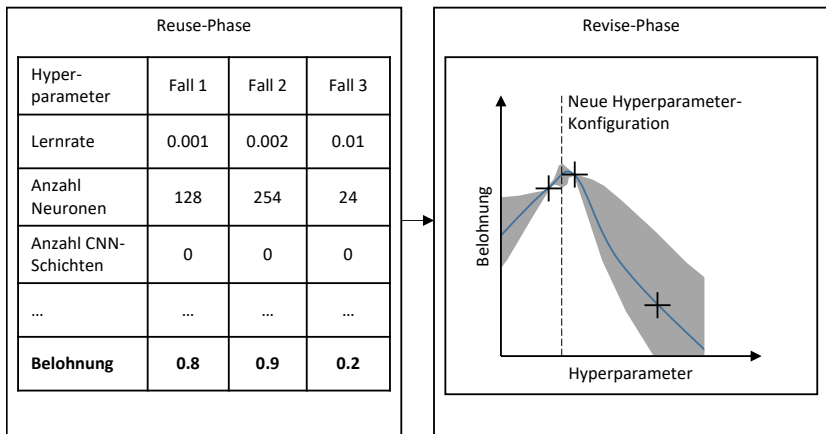


Abbildung 7.12: Wiederverwendung naheliegender Hyperparameter-Konfigurationen und Initialisierung der Bayesschen Optimierung basierend auf den resultierenden Belohnungen

### 7.4.3 Retain-Phase

Das Belohnungssignal stellt lediglich eine Approximation der realen Aufgabenstellung dar, die zum Trainieren eines Agenten verwendet wird. Darüber hinaus kann die Agentenaufgabe über unterschiedliche Belohnungsfunktionen abgebildet werden, wodurch eine Bewertung unterschiedlicher Agentenverhalten nicht auf Basis der erreichten Belohnungswerte möglich ist. Für die innerhalb der Retain-Phase stattfindende Entscheidung,

## 7 Automatische Konfiguration des Agenten

---

ob die gefundene Lösung verwendet und in die Datenbasis mit aufgenommen werden kann, ist daher eine zusätzliche Validierung des gelernten Agentenverhaltens innerhalb der Simulation notwendig.

Durch die Varianz in der Aufgabenstellung (z. B. unterschiedliche Abstände zur Zielposition zu Beginn der Episode) kann die Bewertung des Agentenverhaltens zu unterschiedlichen Ergebnissen führen. Für die ganzheitliche Bewertung des Agentenverhaltens wird in der Retain-Phase daher eine mehrfache Simulation des Agentenverhaltens im Varianz-Bereich der Aufgabenstellung (sog. Monte-Carlo-Simulation) durchgeführt, um anschließend lokale Bewertungen einzelner Versuche und eine globale Bewertung des Agentenverhaltens aus den in der Simulation ermittelten Variablen, wie der Erfolg eines Versuchs und Taktzeiten, abzuleiten.

Die Bewertung des Agentenverhaltens erfolgt auf Basis innerhalb der Skill-Ebene definierten Agenten-Skills und deren Rahmenbedingungen wie "Kollisionsfreiheit", um eine von der Belohnungsfunktion bzw. Agentenzielen unabhängige Bewertung zu ermöglichen. So erfordert der in Abschnitt 5.3.1 dargestellte Einsatz des Agentenziels "Pose erreichen" zur Umsetzung des Agenten-Skills "Führen" zusätzliche Informationen zum Verhalten des virtuellen Bahnpunkts (z. B. Verfahren an/aus). Das Bewertungsmodul zum Agenten-Skill "Führen" muss daher berücksichtigen, ob sich Position und Orientierung während der Fahrt des virtuellen Bahnpunkts innerhalb eines vordefinierten Toleranzbereichs befinden.

Die Ermittlung der Taktzeiten erfolgt ebenfalls auf Basis des Bewertungsmoduls des Agenten-Skills, da nur auf dieser Abstraktionsebene definiert werden kann, wann ein Agenten-Skill abgeschlossen wurde. So ist im vergangenen Beispiel "Führen" ein Versuch erst beendet, wenn der Bahnpunkt das Ende der Trajektorie erreicht hat.

Eine globale Bewertung erfolgt durch Aggregation der Kennzahlen und variabler Merkmale der Umgebung. Abbildung 7.13 veranschaulicht die Bewertung des Agenten-Skills "Weitergeben" mit einer variablen Startposition unter der Rahmenbedingung der "Kollisionsfreiheit"

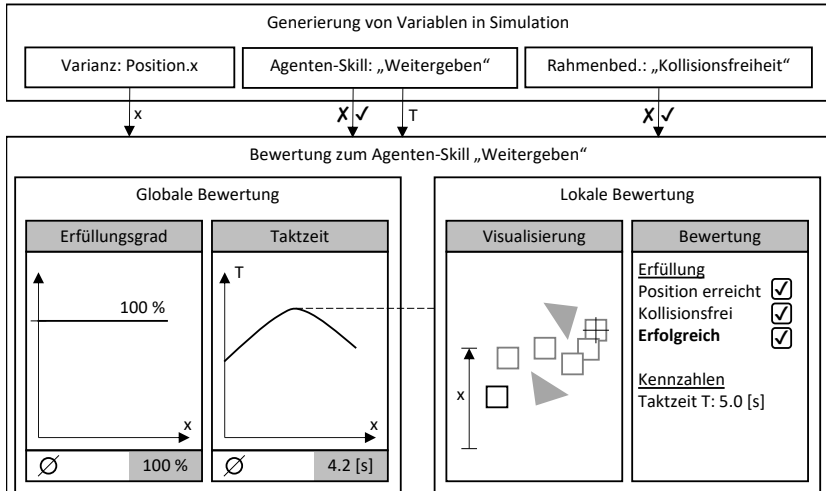


Abbildung 7.13: Gesamt-Bewertung der Umsetzung eines Agenten-Skills basierend auf der Bewertung einzelner Versuche

## 7.5 Zusammenfassung

Durch das Erlernen der Zusammenhänge zwischen einzelnen Elementen der Aufgabenbeschreibung des Agenten und der Ähnlichkeit der Hyperparameter-Konfiguration mittels eines siamesischen Netzes wird kein explizites Wissen in der Definition des Informationsmodells vorausgesetzt. Eine Analyse der resultierenden Case Embeddings zeigt, dass eine Interpretation des Einflusses einzelner Elemente der Aufgabenstellungen (Word Embeddings) hinsichtlich der Ähnlichkeit der Hyperparameter-Konfigurationen möglich ist. Somit lassen sich relevante Hyperparameter-Konfigurationen aus der Datenbasis für neuen Aufgabenstellungen finden.

Dennoch kann nicht davon ausgegangen werden, dass eine bestehende Hyperparameter-Konfiguration ohne Anpassung wiederverwendet werden kann. Zur Automatisierung der Revise-Phase wird daher eine Bayessche Optimierung eingesetzt, welcher auf bestehenden Ansätzen des automatisierten maschinellen Lernens aufbaut.

Das erlernte Agentenverhalten muss durch den Nutzer bewertet werden können. Hierzu werden weitere Simulationen durchgeführt, um Kennzahlen zum erlernten Agentenverhalten bezogen auf den Agenten-Skill zu erhalten.



## 8 Umsetzung und Erprobung

Zur Erprobung des Ansatzes wird zunächst ein Referenzszenario definiert (Abschnitt 8.1) und die softwaretechnische Umsetzung des Systems erläutert (Abschnitt 8.2). Ergebnisse der Anwendung des Systems auf die drei Teilprozesse des Referenzszenarios in Abschnitt 8.3 bilden die Grundlage für die wirtschaftliche und technische Bewertung in Kapitel 9.

### 8.1 Definition des Referenzszenarios

Aufbauend auf Vorarbeiten im Bereich der aufgabenorientierten Programmierung wird das System auf Basis einer Schweißaufgabe erprobt (siehe Abbildung 8.1). KRUG (2012), BACKHAUS (2016) und KUSS (2020) verwenden ein Referenzszenario, in welchem ein Sechssachs-Knickarmroboter zur Durchführung eines MIG/MAG (Metall-Inertgas/ Metall-Aktivgas) Schweißprozesses eingesetzt wird. Der Schweißbrenner ist mit einer automatischen Drahtzufuhr ausgestattet und wird zur Umsetzung einer geradlinigen Kehlnaht eingesetzt.

Da diese Arbeit die Integration von DRL in die aufgabenorientierte Programmierung anstrebt, werden die folgenden Kriterien zum Einsatz von DRL in der Adaption des Referenzszenarios berücksichtigt:

- *Notwendigkeit von Agenten-Skills*: Der Einsatz von Reinforcement Learning ist lediglich bei Teilprozessen notwendig, welche aufgrund der Varianz oder komplexen Roboterkinematiken nicht bzw. nicht wirtschaftlich durch statische Steuerungslogiken umgesetzt werden können.
- *Umsetzbarkeit statischer Steuerungslogiken*: Die statischen Steuerungslogiken müssen in Kombination mit Agenten-Skills ausführbar sein.
- *Simulierbarkeit*: Das Trainieren und Bewerten des Agentenverhaltens setzt in dieser Arbeit eine Simulierbarkeit der betrachteten Montageprozesse voraus.

## 8 Umsetzung und Erprobung

---

- *Beschreibbarkeit*: Die Montageaufgabe muss sich durch die semantischen Elemente bestehender Fälle beschreiben lassen.

Um die industrielle Relevanz des Ansatzes für KMU zu gewährleisten, werden im Referenzszenario typische Rahmenbedingungen der Montageprozesse integriert, welche die Steigerung des Automatisierungsgrads in KMU hindern und den Einsatz intelligenter Agenten erfordern:

- *Variierende Objektgeometrie*: Produktflexibilität hinsichtlich der Geometrie stellt eine Herausforderung von Montagesystemen dar. Sollen mehrere Produktvarianten ohne Rekonfiguration des Montagesystems verarbeitet werden, sind eine Wahrnehmung und entsprechende Anpassung des Verhaltens durch das Montagesystem notwendig.
- *Variierende Objektpose*: Es kann nicht von definierten Ordnungszuständen der Produkte bei den Eingängen des Montagesystems ausgegangen werden, da die logistischen Rahmenbedingungen hierfür nicht immer gegeben bzw. wirtschaftlich umsetzbar sind.

Der Handhabungsvorgang zum Montagesystem wird als ein manueller Prozess ausgeführt, weshalb bei der Ablage des Produkts auf dem Förderband eine Varianz hinsichtlich der Objektpose (translatorischen Koordinaten und Ablagewinkel) angenommen wird. Zudem werden zwei Varianten des "Bauteil A" im Montagesystem verarbeitet, weshalb auch variierende Objektgeometrien vorliegen. Diese Form- und Lagevarianz gilt es bei der Umsetzung einzelner Agenten-Skills zu berücksichtigen, wenn diese eine Anpassung des Agentenverhaltens erfordert.

Zudem stellen begrenzte räumliche Möglichkeiten ein Hindernis zur Integration von Automationslösungen in KMU dar, weshalb im Referenzszenario eine Überschneidung des Arbeitsraums mit einem zweiten Roboter vorgesehen wird. Dieser zweite Roboter, welcher nicht im betrachteten Montageprozess eingesetzt wird, stellt daher ein sich bewegendes Kollisionsobjekt dar. Da die Trajektorie des Roboters variieren kann, ist eine Entscheidungsfindung geeigneter Trajektorien während des Handhabungsprozesses von "Bauteil B" notwendig.

Der Schweißprozess selbst wird anschließend durch den Schweißbrenner durchgeführt, welcher von einer Portal-Kinematik gehandhabt wird. Hierbei erfolgt keine Ansteuerung von Roboterachsen, sondern eine direkte Ansteuerung der translatorischen und rotatorischen Geschwindigkeit des Endeffektors bzw. des Schweißbrenners. In der Modellierung der simulativen Lernumgebung wird daher lediglich der Schweißbrenner selbst abgebildet.

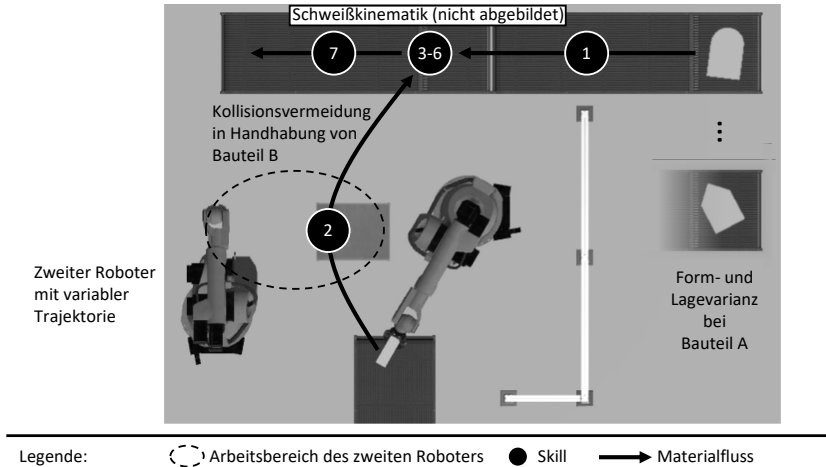


Abbildung 8.1: Referenzszenario mit Handhabungs- und Schweißprozessen sowie deren Realisierung mittels einer Skillsequenz (siehe Abbildung 8.2)

Die Primär- und Sekundärprozesse werden in Skill-Steps eingeteilt, welche sowohl durch Ressourcen-Skills als auch Agenten-Skills umgesetzt werden. Ressourcen-Skills lassen sich durch bestehende Ansätze der aufgabenorientierten Programmierung implementieren (u. a. BACKHAUS 2016, KUSS 2020) und werden in der Erprobung des Ansatzes dieser Arbeit nicht näher betrachtet. Abbildung 8.2 zeigt die Skillsequenz und den Bezug der einzelnen Skills zu den Ressourcen des Umweltmodells.

Im Referenzszenario werden das kollisionsfreie Anfahren von Bauteil B (Abschnitt 8.3.1), das kamerabasierte Positionieren von Bauteil B für einen T-Stoß mit Bauteil A (Abschnitt 8.3.2) und das Folgen der linearen Schweißbahn (Abschnitt 8.3.3) mit Hilfe des Ansatzes dieser Arbeit als Agenten-Skills umgesetzt.

Nach dem kollisionsfreien Anfahren ist ein weiteres Positionieren von Bauteil B mittels einer statischen Steuerungslogik vorgesehen, weshalb beim anschließenden kamerabasierten Positionieren von einem definierten Startzustand ausgegangen wird und lediglich die Koordinaten von Bauteil A variieren können. Während des Schweißprozesses muss die Führung des Schweißbrenners ebenfalls an die Lage der beiden Bauteile angepasst werden. Über einen Tastsensor erfolgt parallel eine Schaltung des Schweißbrenners, welche über eine statische Steuerungslogik abgebildet werden kann.

## 8 Umsetzung und Erprobung

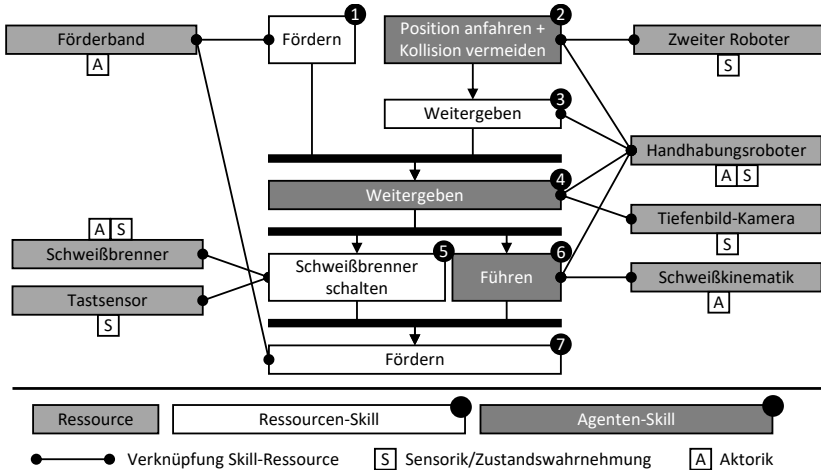


Abbildung 8.2: Skillsequenz im Referenzszenario mit dem Einsatz intelligenter Agenten zum kollisionsfreien Antransport des Bauteils (2), zum kamerabasierten Vorpositionieren des Bauteils (4) und zum Führen der Schweißpistole mittels einer Schweißkinematik (6)

## 8.2 Softwaretechnische Umsetzung

### 8.2.1 Umsetzung der simulativen Lernumgebung

Zur Implementierung der simulativen Lernumgebung wird in dieser Arbeit die Entwicklungsumgebung Unity eingesetzt, in welcher durch das Unity Machine Learning Agent Toolkit (Unity ML-Agents) die Implementierung einer Schnittstelle zu RL-Agenten ermöglicht wird.

Zur Definition der Simulation wird innerhalb der Entwicklungsumgebung die Programmiersprache C# verwendet, über welche die Schnittstelle (Belohnung, Wahrnehmung, Aktionsraum) der Lernumgebung zum Agenten definiert wird. Abbildung 8.3 zeigt eine beispielhafte Implementierung der Belohnungsfunktion. Die Parameter der Belohnungsfunktion werden über die Parametrisierung der Agentenziele im Informationsmodell abgeleitet.

Die Varianz der Lernumgebung erfolgt in den bestehenden Arbeiten durch die manuelle Programmierung eines Skripts, das bei der Initialisierung der Simulation ausgeführt wird



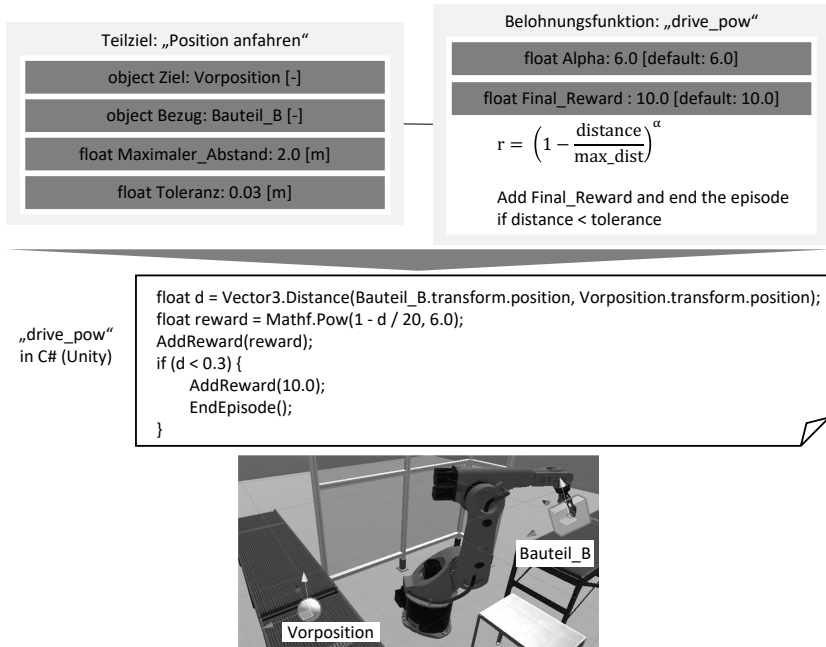


Abbildung 8.3: Beispielhafte Ausprägung des Teilziels "Position anfahren" in Unity

und die variablen Merkmale der Simulationsobjekte in einem vorgegebenen Bereich zufällig initialisiert. Um die manuellen Aufwände und notwendigen Programmierkenntnisse zu reduzieren, werden in dieser Arbeit vordefinierte Standardmerkmale und Simulationsmodule eingesetzt (vgl. Abschnitt 5.5.4).

Abbildung 8.4 zeigt die Ausprägung des Simulationsmoduls zu Form- und Lagevarianz im Unity Editor, welcher die Parametrisierung vordefinierter Skripte in der Benutzeroberfläche ermöglicht. Neben den drei variierenden Merkmalen hinsichtlich der Position und Orientierung des Produkts wird über eine Referenzierung auf mehrere Produktvarianten die Formatflexibilität des Produkts definiert und während der Trainingsphase des Agenten randomisiert.

Diese variierenden Merkmale sowie weitere zur Bewertung des Agentenverhaltens notwendige Variablen, werden in der Simulationsumgebung aufgezeichnet und im Datenformat CSV (Character-separated values) abgespeichert.

## 8 Umsetzung und Erprobung

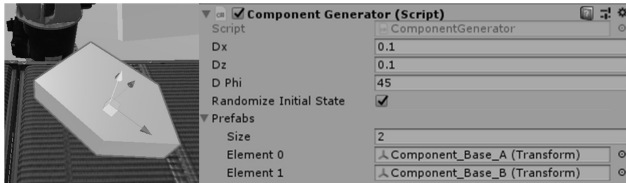
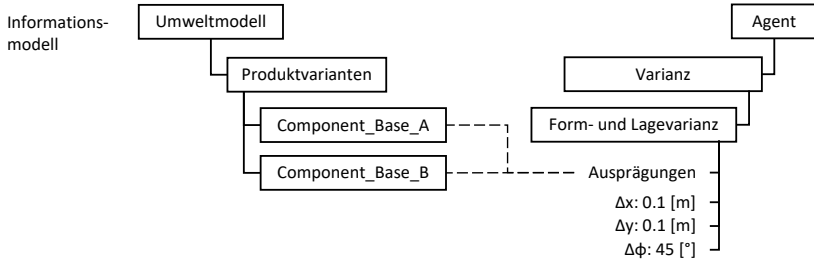


Abbildung 8.4: Ausprägung der im Informationsmodell definierten Varianz in einem vordefinierten Simulationsmodul in Unity

### 8.2.2 Umsetzung der automatischen Konfiguration des Agenten

Der in Kapitel 7 vorgestellte Ansatz zur automatischen Konfiguration des Agenten ist in der Programmiersprache Python realisiert. Die Architektur der in der Retrieve-Phase verwendeten siamesischen Netzarchitektur ist in der Deep-Learning-Bibliothek *Tensorflow* und deren Erweiterung für probabilistische Ansätze *Tensorflow Probability* umgesetzt (RÖHLER 2021).

Die Wiederverwendung bestehender Hyperparameter-Konfigurationen und das Trainieren des Agenten erfolgen auf Grundlage der Python-basierten Schnittstelle von Unity ML-Agents, welche in der in dieser Arbeit verwendeten Version 0.9.1 ebenfalls auf der Bibliothek *Tensorflow* aufbaut. Die Definition der Hyperparameter-Konfiguration des Agenten erfolgt über eine Konfigurationsdatei im YAML-Datenformat (rekursives Akronym für "YAML Ain't Markup Language"). Die Umsetzung des Agenten wird über die Implementierungslogik in Unity ML-Agents realisiert.

Für die Bayessche Optimierung der Hyperparameter-Konfigurationen in der Revise-Phase wird die Implementierung von NOGUEIRA 2014 verwendet. Die Bayessche Optimierung basiert auf einem Gauß-Prozess, der die erreichte Belohnung bei unterschiedlichen Hyperparameter-Konfigurationen approximiert, und dem Explorationsansatz des Upper Confidence Bound (UCB), welcher basierend auf dem Gauß-Prozess die Hyperparameter-Konfiguration auswählt, welche die höchste Wahrscheinlichkeit zur Erhöhung der Belohnung besitzt.

Die Bewertung des Agenten in der Retain-Phase erfolgt durch die vom Agenten-Skill und der Varianz abhängigen Variablen, welche innerhalb der Simulation generiert werden, und deren Analyse in einem dem Agenten-Skill zugeordneten Python-Skript. Abbildung 8.5 gibt einen Überblick der in der Umsetzung verwendeten Bibliotheken.

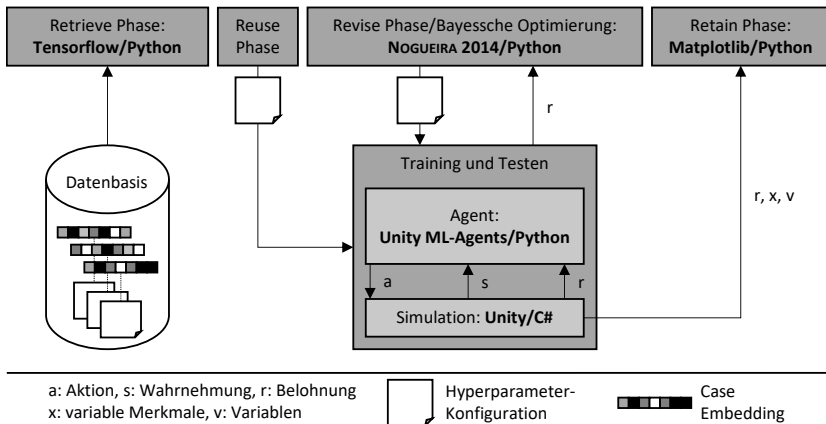


Abbildung 8.5: Softwaretechnische Umsetzung der Systemkomponenten

### 8.3 Anwendung im Referenzszenario

#### 8.3.1 Automatische Konfiguration des Agenten beim kollisionsfreien Anfahren

Mit dem Agenten-Skill "Position anfahren" mit der Rahmenbedingung "Kollisionsfreiheit" wird die Vorpositionierung des Bauteils im Arbeitsbereich des Schweißprozesses mittels Deep Reinforcement Learning erlernt. Das Markow-Entscheidungsproblem ist, angelehnt an Vorarbeiten aus der Literatur, wie folgt definiert:

- *Aktionsraum*: Ansteuerung der sechs Achsgeschwindigkeiten (begrenzt auf  $180^\circ/s$ ) des Handhabungsroboters mit 6 Hz
- *Wahrnehmung*: Die sechs eigenen Achspositionen und -geschwindigkeiten und die des zweiten Roboters (24 skalare Werte)
- *Curriculum*: Der Agenten-Skill wird umgesetzt mit dem Teilziel "Position anfahren" (Ausprägung siehe Abbildung 8.3) und dem Teilziel "Kollision vermeiden" (Abbruch der Episode bei Kollision).

Zur Erprobung des Ansatzes dieser Arbeit werden die drei ähnlichsten Hyperparameter-Konfigurationen wiederverwendet (Reuse Phase) und anschließend eine Bayessche Optimierung durchgeführt, welche basierend auf den wiederverwendeten Hyperparameter-Konfigurationen weitere Kandidaten zum Testen auswählt (Revise Phase). Zur Optimierung werden die Hyperparameter Lernrate und Batch-Größe, welche einen großen Einfluss auf das Lernverhalten des Agenten haben (ANDRYCHOWICZ et al. 2020), gewählt und maximal 10 Versuche durchgeführt.

Abbildung 8.6 zeigt den Ansatz dieser Arbeit (Reuse- bzw. Revise-Phase) im Vergleich zur Bayesschen Optimierung ohne Wiederverwendung bestehender Lösungen und der zufälligen Auswahl von Hyperparametern (sog. Random Search). Da bei Random Search kein Wissen aus vorangegangenen Experimenten eingesetzt wird, wird der Verlauf in Abbildung 8.6 in aufsteigender Reihenfolge dargestellt.

Ohne Wiederverwendung bestehender Lösungen kann durch die Bayessche Optimierung nach fünf Iterationen ein Belohnungswert erreicht werden, welcher mit wiederverwendeten Lösungen vergleichbar ist. Im Vergleich zur Bayesschen Optimierung bis zur 5. Iteration (Gesamtzeit: 5,9 Stunden) kann durch die Wiederverwendung einer Hyperparameter-Konfiguration (Gesamtzeit ca. eine Stunde) die Trainingszeit um 83.4 %

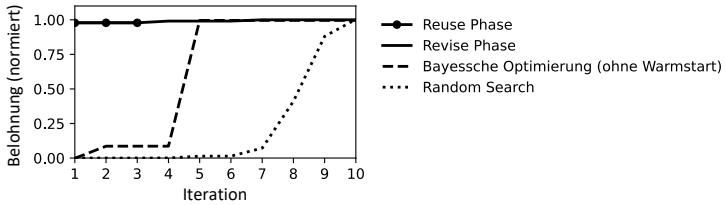


Abbildung 8.6: Experimente zur automatischen Konfiguration des Agenten beim kollisionsfreien Anfahren. Eine Iteration benötigt ca. eine Stunde.

verringert werden. Durch Zufallsauswahl (Random Search) kann bei der Beschränkung auf zwei Hyperparameter nur eine valide Konfiguration gefunden werden.

Abbildung 8.7 zeigt die erlernte Strategie des Agenten. In Abhängigkeit der Achswinkel und -geschwindigkeiten des zweiten Roboters werden die sechs Achsgeschwindigkeiten des Handhabungsroboters so gewählt, dass eine Kollision vermieden bzw. die schnellstmögliche Trajektorie verfolgt wird, wenn keine Kollisionsgefahr mit dem zweiten Roboter besteht.

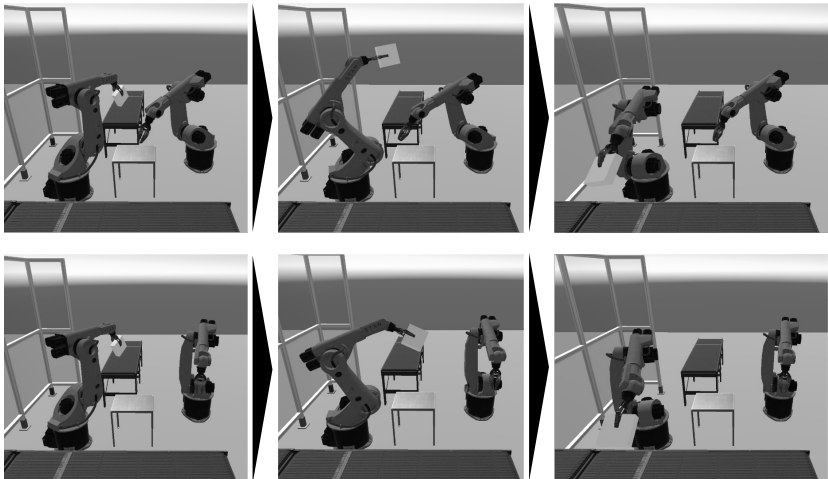


Abbildung 8.7: Erlernetes Agentenverhalten zum Anfahren einer Zielposition mit Kollisionsvermeidung. In Abhängigkeit der Entfernung des zweiten Roboters werden längere Trajektorien zur Kollisionsvermeidung (oben) oder kürzere Trajektorien bei ausreichender Entfernung des zweiten Roboters zur Prozessbeschleunigung (unten) gefahren.

### 8.3.2 Automatische Konfiguration des Agenten beim kamerabasierten Weitergeben

Mit dem Agenten-Skill "Weitergeben" wird das Bauteil B für das Schweißen in Abhängigkeit der Lage und Orientierung von Bauteil A mit der folgenden Konfiguration auf MDP-Ebene ausgerichtet:

- *Aktionsraum*: Ansteuerung der sechs Achsgeschwindigkeiten (begrenzt auf  $30^\circ/s$ ) des Handhabungsroboters mit 6 Hz
- *Wahrnehmung*: Tiefenbild-Kamera mit einer Auflösung von  $84 \times 84$  Pixel (siehe Abbildung 8.8c)
- *Curriculum*: Der Agenten-Skill wird mit zwei Teilzielen "Pose erreichen" umgesetzt (vgl. Abbildung 5.8), welche sich auf den Start und das Ende der Schweißbahn innerhalb der beiden Produktmodelle beziehen (siehe Abbildung 8.8a).

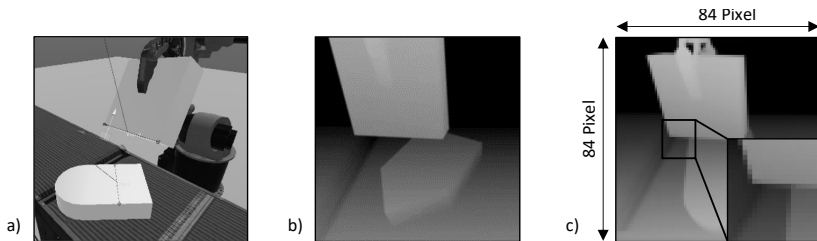


Abbildung 8.8: Simulative Lernumgebung beim kamerabasierten Positionieren: a) Startposition der Bauteile mit Schweißbahnen als Teil des Produktmodells b) Simulierte Tiefenbild-Kamera c) Wahrnehmung des Agenten

In der automatischen Konfiguration des Agenten wird durch die Wiederverwendung von zwei der drei Hyperparameter-Konfigurationen in der Reuse-Phase die maximale Belohnung erreicht (siehe Abbildung 8.9). Alle drei wiederverwendeten Netz-Architekturen basieren auf CNN-Schichten, in welchen die zweidimensionale Wahrnehmung des Agenten zunächst vorverarbeitet wird. Ohne Wiederverwendung bestehender Hyperparameter-Konfigurationen wird die maximale Belohnung in der Bayesschen Optimierung ohne Warmstart nach drei Iterationen erreicht.

Aufgrund der hohen Dauer des Lernprozesses wird an dieser Stelle auf die randomisierte Auswahl an Hyperparametern (Random Search) verzichtet. In den Experimenten zeigt sich, dass keine weitere Erhöhung der Belohnung durch weitere Optimierungsschritte wiederverwendeter Hyperparameter-Konfigurationen erfolgt. Da der Lösungsraum mit den zwei betrachteten Hyperparametern stark eingeschränkt wird, kann allerdings davon ausgegangen werden, dass ohne Wiederverwendung der restlichen Hyperparameter aus der Datenbasis mit weiteren Iterationen zu rechnen ist.

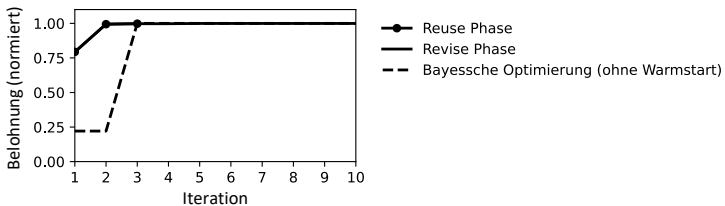


Abbildung 8.9: Experimente zur automatischen Konfiguration des Agenten beim kamerabasierten Weitergeben des Bauteils. Ein Lernprozess und somit eine Iteration des Optimierungsprozesses benötigt ca. 13 Stunden.

### 8.3.3 Bewertung von Lösungsalternativen beim Folgen der Schweißbahn

Der Schweißprozess wird im Anwendungsfall durch den Agenten-Skill "Führen linear" und einem parallelen Schalten des Schweißgeräts umgesetzt. Der Agenten-Skill wird durch das Teilziel "Pose erreichen" und einem virtuellen Bahnpunkt umgesetzt. Der Bahnpunkt verfährt auf der Schweißbahn des Produktmodells, sobald die Toleranzgrenzen eingehalten werden. Als Lösungsalternativen auf MDP-Ebene werden unterschiedliche Frequenzen im Aktionsraum betrachtet.

- *Aktionsraum*: Ansteuerung der translatorischen und rotatorischen Geschwindigkeit der Schweißkinematik (nicht dargestellt) mit unterschiedlichen Frequenzen (10 Hz bzw. 20 Hz)
- *Wahrnehmung*: Lage und Orientierung des Greifers und des Schweißbrenners (siehe Abbildung 8.10)

## 8 Umsetzung und Erprobung

---

- *Curriculum*: Der Agenten-Skill wird mit dem Teilziel "Pose erreichen" mit Bezug auf den virtuellen Bahnpunkt umgesetzt. Als Toleranzbereich wird ein maximaler Abstand von 5 mm und ein maximaler Winkelabstand von  $5^\circ$  gewählt.

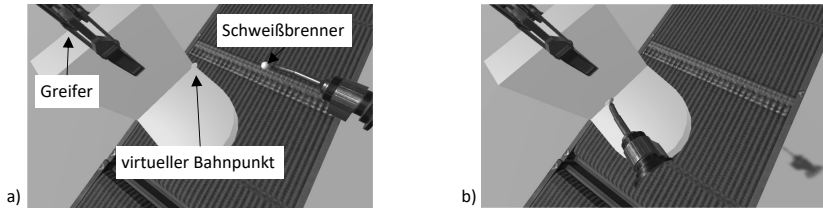


Abbildung 8.10: Simulative Lernumgebung für den Agenten-Skill "Führen linear": a) Start einer Episode mit Form- und Lagevarianz b) Verfahren des virtuellen Bahnpunkts entlang der Schweißbahn

Wie die Experimente der ersten beiden Anwendungsfälle der automatischen Konfiguration des Agenten zeigen (vgl. Abbildungen 8.6 & 8.9), lässt sich durch Wiederverwendung ähnlicher Hyperparameter-Konfiguration ein geeignetes Agentenverhalten erreichen. Im Folgenden wird der Fokus auf die Bewertung des Agentenverhaltens innerhalb der Retain-Phase gelegt.

Nach den Trainingsphasen der Agenten werden in der Retain-Phase 1.000 Versuche innerhalb der Simulation mit dem erlernten Agentenverhalten durchgeführt. Abbildung 8.11 zeigt die aufgezeichneten Variablen innerhalb der einzelnen Module der Simulation, welche für die Evaluation in einer CSV-Datei abgespeichert werden.

Als mögliche Lösungsalternativen werden zwei unterschiedliche Frequenzen der Aktionsignale des Agenten an die Schweißkinematik gewählt. Wie Tabelle 8.1 zeigt, wird durch das Evaluations-Modul eine höhere Erfolgsquote bei einer Frequenz von 20 Hz erreicht.



### 8.3 Anwendung im Referenzszenario

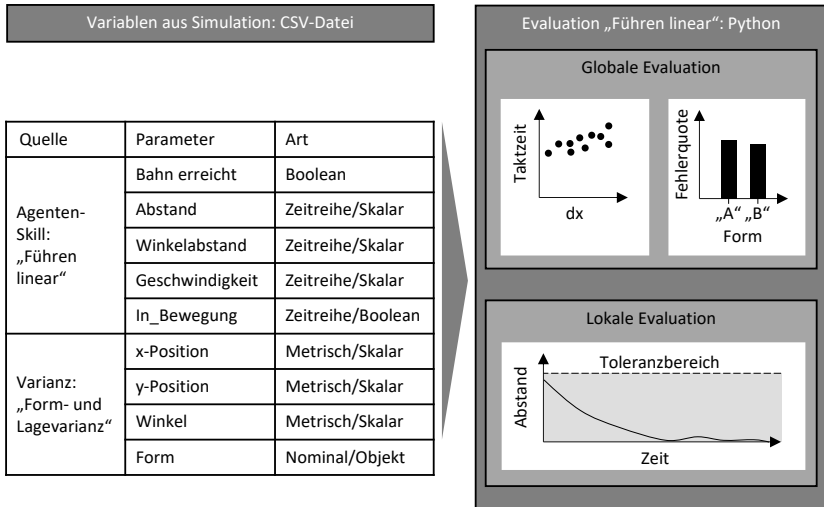


Abbildung 8.11: Evaluation eines gelernten Agentenverhaltens beim Agenten-Skill "Führen linear" in der Retain Phase

Die globale Evaluation ermöglicht zudem eine Analyse der Kenngrößen in Abhängigkeit der variierenden Merkmale. Abbildung 8.12 zeigt den Einfluss des Ablagewinkels  $\Delta\Phi$  des Bauteils auf die Kenngrößen. Abbildung 8.12a zeigt den Einfluss des Ablagewinkels auf die Fehleranzahl, die mit größeren Ablagewinkeln zunimmt (siehe Abbildung 8.13). Eine Möglichkeit zur Steigerung der Erfolgsquote auf 100 % liegt somit in der Reduktion der Winkeltoleranz, mit welcher die Bauteile durch die Mitarbeiter auf das Fließband abgelegt werden, auf  $\pm 5^\circ$  (z. B. durch den Einbau von Schikanen).

Tabelle 8.1: Globale Kennzahlen für den Agenten-Skill "Führen linear" für unterschiedliche Frequenzen der Aktions-Signale des Agenten

	Frequenz	
	10 Hz	20 Hz
Erfolgsquote	89.20 %	93.50 %
Taktzeit	4.61 s	4.63 s
∅ Abstand	1.85 mm	1.74 mm
∅ Winkelabstand	0.78°	0.69°

## 8 Umsetzung und Erprobung

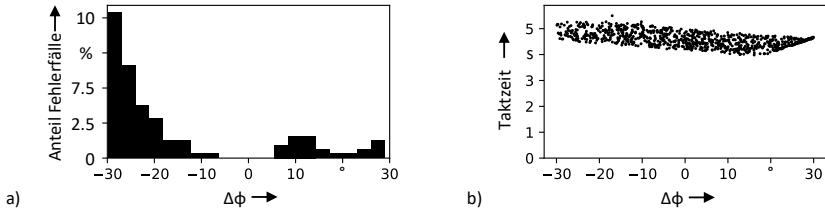


Abbildung 8.12: Globale Evaluation des Agentenverhaltens: a) Fehlerfälle in Abhängigkeit des Ablagewinkels b) Taktzeit in Abhängigkeit des Ablagewinkels

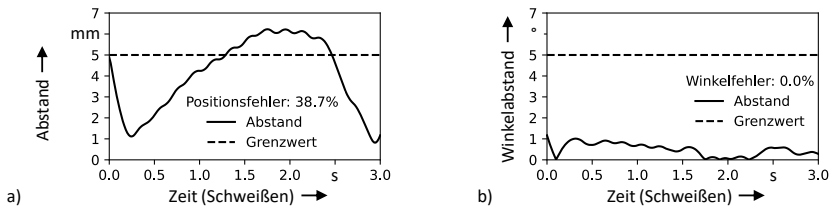


Abbildung 8.13: Lokale Evaluation eines fehlerhaften Versuchs mit einem Ablagewinkel von  $-29.9^\circ$ : a) Positionsfehler während des Schweißens b) Fehlerfreiheit hinsichtlich des Winkels

Positive Ablagewinkel führen dagegen zu einer Reduktion der Taktzeit (Abbildung 8.12b), was durch das notwendige Abbremsen der Schweißkinematik beim Anfahren des Startpunkts der Schweißbahn begründet ist, da hier ein Richtungswechsel des Schweißgeräts bei negativen Ablagewinkeln stattfinden muss, um der Schweißtrajektorie zu folgen. Die Untergrenze der Taktzeit stellt die Zeit zum Abfahren der Schweißtrajektorie dar, welche drei Sekunden dauert und zunächst durch die Schweißkinematik erreicht werden muss.

## 9 Technische und wirtschaftliche Bewertung

In diesem Kapitel werden zunächst die Erkenntnisse aus der Anwendung des Systems im Referenzszenario zusammengefasst und die technischen Grenzen erläutert. Anschließend erfolgt eine Bewertung der Wirtschaftlichkeit des Ansatzes. Hierbei wird zunächst die Konfiguration des Agenten selbst betrachtet wird. Hierauf aufbauend erfolgt eine wirtschaftliche Bewertung des Montagesystems.

### 9.1 Anforderungserfüllung und technische Grenzen

In Abschnitt 4.2.4 werden sechs Anforderungen an das System definiert, welche in diesem Abschnitt auf Basis des Referenzszenarios eingeschätzt werden. Abbildung 9.1 gibt einen Überblick zur Anforderungserfüllung hinsichtlich der in Abschnitt 4.3.1 definierten Funktionalitäten des Systems.

		Anforderungserfüllung					
		1	2	3	4	5	6
Wissensbasierte Entscheidungsunterstützung	Ableitung von Skillsequenzen	🕒	🕒	🕒	🕒	🕒	🕒
	Zuordnung von Agentenskills	🕒	🕒	🕒	🕒	🕒	🕒
	Modellierung der Varianz	🕒	🕒	🕒	🕒	🕒	🕒
	Ableitung der Lernumgebung	🕒	🕒	🕒	🕒	🕒	🕒
Automatische Konfiguration des Agenten	Automatische Hyperparameter-Konfiguration	🕒	🕒	🕒	🕒	🕒	🕒
	Evaluation des Agentenverhaltens	🕒	🕒	🕒	🕒	🕒	🕒

🕒 wenig erfüllt  
 🕒 teilweise erfüllt  
 🕒 überwiegend erfüllt  
 🕒 voll erfüllt

Abbildung 9.1: Einschätzung der Anforderungserfüllung hinsichtlich Effizienz (1), Universalität und Flexibilität (2), Transparenz (3), Einschätzbarkeit der Umsetzbarkeit (4), geringes notwendiges Expertenwissen (5) und Adaptierbarkeit (6)

## 9 Technische und wirtschaftliche Bewertung

---

Die *Effizienz* des Ansatzes bezieht sich auf die manuellen Aufwände innerhalb des Konfigurationsprozesses. Während sich die automatische Konfiguration des Agenten weitestgehend automatisieren lässt, sind zur Definition der MDP-Ebene noch mehrere Entscheidungsprozesse durch den Nutzer notwendig. Eine nicht betrachtete Möglichkeit zur Automatisierung dieser manuellen Schritte stellt die Integration eines zusätzlichen regelbasierten Ansatzes dar. Auch ist die Definition der MDP-Ebene mit dem Aufbau einer Simulationsumgebung verbunden, welcher in dieser Arbeit nicht automatisiert umgesetzt ist.

Unterschiedliche Abstraktionsebenen gewährleisten die *Universalität* und *Flexibilität* des Ansatzes. So können Teilziele für neue Aufgabenstellungen des Agenten wiederverwendet werden. Jedoch sind die umsetzbaren Aufgabenstellungen durch die Verwendung der physikbasierten Simulation technisch begrenzt. Zur Erweiterung des Ansatzes bedarf es daher zusätzlicher Simulationen (z. B. Schweißsimulationen). Auch setzt der Ansatz des fallbasierten Schließens das Vorhandensein ähnlicher Aufgabenstellungen voraus, um Lösungen für einzelne Ebenen des Aufgabenmodells finden zu können.

Die Möglichkeit zur Identifikation einzelner Elemente (z. B. Aktor- bzw. Sensorsignale), durch die sich eine neue Aufgabenstellung von bestehenden Aufgabenstellungen unterscheidet, sorgt für *Transparenz* im Lösungsprozess. Hierdurch ist der Entscheidungsprozess innerhalb des Systems in jedem Schritt nachvollziehbar. Da derzeit noch wenige Umsetzungen von Montageaufgaben durch Reinforcement Learning existieren, ist allerdings die Bedeutung einzelner Elemente hinsichtlich der Umsetzbarkeit insbesondere in höheren Abstraktionsebenen des Aufgabenmodells (z. B. Rahmenbedingungen von Prozessen) nicht abgebildet.

Dies führt daher zu einer gesteigerten Unsicherheit in der *Einschätzung der Umsetzbarkeit*. So wird durch die aus einzelnen Elementen der Aufgabenstellung berechneten Distanzmaße eine Einschätzung der Umsetzbarkeit durchgeführt. Bei geringen Distanzen zu bestehenden Aufgabenstellungen kann nur von einer hohen Umsetzbarkeit ausgegangen werden, wenn die Gewichtung der Elemente im Distanzmaß ihre Bedeutung widerspiegelt. Da diese Gewichtung im letzten Schritt der Konfiguration des Agenten mittels Embeddings erfolgt, ist erst am Ende der Planungsphase eine Aussage zur Umsetzbarkeit möglich.

Da bisher nur wenige Umsetzungen mit hoher Varianz existieren, wird die Ausprägung der Varianz in den Distanzmaßen innerhalb dieser Arbeit nicht berücksichtigt. In der Umsetzung der Anwendungsfälle zeigt sich, dass diese Restriktion bei geringer

Varianz vernachlässigbar ist. Eine hohe Varianz (z. B. eine Vielzahl unterschiedlicher Handhabungsobjekte) besitzt allerdings einen hohen Einfluss auf die Komplexität und daher auf die Umsetzbarkeit. Das System ist daher auf ein geringes Ausmaß der Varianz beschränkt.

Durch die automatische Konfiguration des Agenten wird ein *geringes Expertenwissen* vorausgesetzt. So können durch den verwendeten Ansatz alle Anwendungsfälle umgesetzt werden. Sofern keine geeigneten Hyperparameter-Konfigurationen des Agenten gefunden werden können, ist eine weitere Umsetzung der Aufgabenstellung nur durch erhöhtes Expertenwissen in der Definition geeigneter Hyperparameter-Konfigurationen möglich.

Auch wird die Parametrierung der Belohnungsfunktion in der Optimierung nicht berücksichtigt. Da die Gestaltung der Belohnungsfunktion Einfluss auf das Agentenverhalten und somit die Leistung des resultierenden Montagesystems hat, wird Expertenwissen benötigt, um eine weitere Verbesserung des Agentenverhaltens zu ermöglichen. Für die Realisierung des Montagesystems kann eine Erweiterung des Ansatzes durch derzeitige Forschungsansätze zur Einbindung der Belohnungsfunktion in den Optimierungsprozess (vgl. Abschnitt 3.4.2) in Betracht gezogen werden.

Da in dieser Arbeit zudem der RL-Algorithmus PPO und bestehende Architekturen künstlicher neuronaler Netze fokussiert werden, erfolgt keine Validierung der Anwendbarkeit des Ansatzes für weitere RL-Algorithmen und Architekturen. So werden andere Hyperparameter-Konfigurationen bei der Verwendung eines anderen RL-Algorithmus oder weiteren Elementen der Architektur verwendet, welche eine Adaption des umgesetzten Systems erfordern.

Da die automatische Konfiguration des Agenten kein explizites Wissen (z. B. hinsichtlich PPO) verwendet, ist der Lösungsansatz selbst allerdings auf andere Ausprägungen der Hyperparameter-Konfigurationen übertragbar. Die *Adaptierbarkeit* des Ansatzes auf weitere Entwicklungen im Bereich Deep Reinforcement Learning wird somit gewährleistet.

### 9.2 Wirtschaftliche Bewertung

Aufgrund der technischen Grenzen des Systems beschränkt sich die wirtschaftliche Bewertung auf Anwendungsfälle, welche sich durch eine physikbasierte Simulation abbilden lassen. Es wird zudem die Annahme getroffen, dass das reale System durch die Simulation realitätsgetreu abgebildet werden kann und somit keine weitere Anpassung des erlernten Agentenverhaltens in der Realisierungsphase erforderlich ist. In der Rekonfiguration des Systems wird eine Teilmenge der Funktionalitäten benötigt, weshalb der Ansatz auch zur Anpassung an geänderte Aufgabenstellungen (z. B. neue Produktvarianten) im Betrieb genutzt werden kann.

Da noch kein Expertenwissen zur Konfiguration von Agenten in Unternehmen existiert, wird zur wirtschaftlichen Bewertung des Systems der Vergleich mit einer externen Dienstleistung herangezogen. Den Kosten der Dienstleistung stehen die Lizenzkosten für den Einsatz des in dieser Arbeit konzeptionierten Systems, sowie die Qualifizierung und die Stundensätze der internen Arbeitskräfte gegenüber (siehe Tabelle 9.1).

Tabelle 9.1: Getroffene Annahmen zur Umsetzung der Agenten

Annahmen zum Anwendungsfall		
Anzahl Agentenaufgaben	n	3 -
Produktänderungen / Rekonfigurationen	R	12 -
Stundensatz Mitarbeitende	KM	50 €/h
Kostenannahmen zur Dienstleistung		
Dienstleistung Umsetzung eines Agenten	KA	30.000 €
Dienstleistung Trainieren des Agenten	KT	5.000 €
Kostenannahmen zum System		
Lizenzkosten Software	KS	20.000 €
Qualifizierung Mitarbeitende	KQ	10.000 €

Bei der Annahme von 12 Produktänderungen und drei Agentenaufgaben ergibt sich der in Tabelle 9.2 abgebildete Überschuss. Der Überschuss wird über die Einsparung der Kosten eines Dienstleisters abzüglich der in Tabelle 9.1 gezeigten Kosten des Systems und der manuellen Aufwände der internen Arbeitskräfte ermittelt. Da das System analog zu einer virtuellen Inbetriebnahme eine Simulationsumgebung des Montagesystems erfordert, müssen diese Aufwände auch in der wirtschaftlichen Bewertung des Systems berücksichtigt werden.

Tabelle 9.2: Überschuss bei Einsatz des Systems

Einsparungen		
Initiale Umsetzung	$KA \cdot n$	90.000 €
Rekonfiguration	$KT \cdot R \cdot n$	180.000 €
Gesamteinsparungen	GE	270.000 €

Manuelle Aufwände		
Konfiguration der Simulationsumgebung	TU	40 h
Konfiguration der Agentenaufgabe	TG	8 h
Konfiguration des Agenten	TA	1 h
Kosten initiale Umsetzung	$KU = KM \cdot (TU + TG) \cdot n$	7.350 €
Kosten Rekonfiguration	$KR = KM \cdot TA \cdot R \cdot n$	1.800 €
Kosten manueller Aufwände	$GK = KU + KR$	9.150 €

Überschuss	$GE - (GK + KS + KQ)$	230.850 €
------------	-----------------------	-----------

Im Referenzszenario aus Abschnitt 8.3 ergibt sich somit ein Überschuss von 230.850 €. Da die Anzahl der Agenten und Rekonfigurationen einen wesentlichen Einfluss auf den Überschuss haben, wird in Abbildung 9.2 ein Vergleich unterschiedlicher Szenarien gezeigt. Während der Überschuss bei zunehmendem Einsatz des Systems zunimmt, wird lediglich bei der Annahme eines Agenten ohne Rekonfiguration kein Überschuss erzielt.

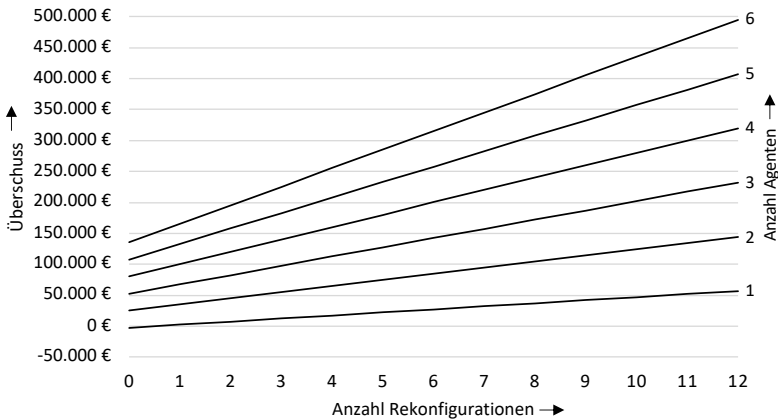


Abbildung 9.2: Vergleich der Überschüsse bei unterschiedlicher Anzahl an Agenten und Rekonfigurationen.

## 9 Technische und wirtschaftliche Bewertung

Bei der Eingrenzung der wirtschaftlichen Bewertung auf die Prozesse zur software-technischen Umsetzung des Agenten lässt sich bei mehrfacher Anwendung daher ein wirtschaftlicher Nutzen der aufgabenorientierten Konfiguration von Agenten feststellen. Für eine ganzheitliche wirtschaftliche Bewertung ist zudem eine Betrachtung der Hardwarekosten des Robotersystems sowie weitere Aufwände in der Projektierung und Programmierung des Montagesystems notwendig.

Für den Betrieb von auf Reinforcement Learning basierenden Robotersystemen kann noch auf wenige Erfahrungen zurückgegriffen werden. In einem Interview mit der Firma Obeta (RÖHLER et al. 2021), welche ein auf Reinforcement Learning basierendes Robotersystem zur Kommissionierung im Lager nutzt, kann allerdings der *manuelle Eingriff im Fehlerfall* des autonomen Systems als Kostenquelle identifiziert werden. Neben diesen Prozessfehlerkosten müssen zudem die laufenden Hardwarekosten im Betrieb berücksichtigt werden.

Die für die Realisierung und den Betrieb getroffenen Annahmen sind in Tabelle 9.3 zusammengefasst. Hinsichtlich des Einsparpotenzials des Montagesystems wird von einem 3-Schicht-Betrieb an einem Montagearbeitsplatz ausgegangen.

Tabelle 9.3: Getroffene Annahmen zum Unternehmen und zum Montagesystem

Unternehmenskennzahlen		
Abschreibungszeitraum	AB	5 Jahre
Kalk. Zinsen	Z	10 %
Stückzahl pro Jahr pro Montageplatz	SZ	100.000 -
Personalkosten Werkskraft pro Jahr	KW	50.000 €/a
Anzahl Montageplätze / Anlagenvarianten	N / NA	1 -
Anzahl Unternehmen	NU	1 -

Annahmen zum Montagesystem		
Fehlerquote (Betrieb)	FQ	0,1 %
Kosten im Fehlerfall (Betrieb)	KF	100 €/Fehler
Kosten für manuelle Programmierung (SPS)	KS	30.000 €
Kosten für Umprogrammierung (SPS)	KU	2.000 €

Für die betrachteten Rahmenbedingungen werden in Tabelle 9.4 vier Umsetzungsszenarien zur Berechnung der Kosten und Einsparungen unterschieden:

- *Automation*: Automatisiertes Montagesystem basierend auf einer statischen Steuerungslogik mit der Notwendigkeit zur Umprogrammierung bei Produktänderungen und nur teilweisen Automatisierung (50 % bestehender Montageprozesse)



## 9.2 Wirtschaftliche Bewertung

- *Dienstleistung*: Realisierung und Rekonfiguration eines autonomen Montagesystems mit Hilfe eines Dienstleisters
- *System (Agent)*: Anwendung der aufgabenorientierten Konfiguration des Agenten in der Realisierung und Rekonfiguration
- *System (voll. Funktionalität)*: Vollständige Funktionalität durch zusätzlichen Einsatz der aufgabenorientierten Programmierung

In den Szenarien des autonomen Montagesystems wird von den Ergebnissen aus Tabelle 9.2 für die Konfiguration des Agenten ausgegangen. Im Szenario des automatisierten Montagesystems wird dagegen von Kosten zur manuellen Programmierung und Umprogrammierung aus Tabelle 9.3 sowie von geringeren Hardwarekosten ausgegangen.

Tabelle 9.4: Kosten und Einsparung der Umsetzungsszenarien

		Automation	Autonom		
			Dienstleistung	System (Agent)	System (vollst. Funktionalität)
<b>Hardwarekosten</b>					
Roboter		100.000 €		100.000 €	
Sensorik		0 €		10.000 €	
Steuerung		5.000 €		10.000 €	
Weitere Ressourcen (Schutzgäune, Fließband)		15.000 €		15.000 €	
Summe Hardwarekosten	HK	120.000 €		135.000 €	
<b>Realisierungskosten</b>					
Konfiguration Agent		-	90.000 €	7.350 €	7.350 €
Aufwandsreduktion (Programmierung)	AP	-	-	-	50 %
Programmierkosten	KS*(1-AP)*NA	30.000 €	30.000 €	30.000 €	15.000 €
Projektierung und Inbetriebnahme	25% HK*N	30.000 €	33.750 €	33.750 €	33.750 €
Kosten zur Qualifizierung	KQ*NU	-	-	10.000 €	10.000 €
Summe Realisierungskosten	RK	60.000 €	153.750 €	81.100 €	66.101 €
Summe Invest pro Montageplatz	HK+RK/N	180.000 €	288.750 €	216.100 €	201.101 €
<b>Betriebskosten</b>					
Hardware (Wartung, Energie, Raum)	15% HK	18.000 €	20.250 €	20.250 €	20.250 €
Prozessfehlerkosten	(1-FQ)*SZ*KF	-	8.500 €	8.500 €	8.500 €
Rekonfiguration		24.000 €	180.000 €	1.800 €	1.800 €
Softwarelizenz		-	-	20.000 €	20.000 €
Summe Betriebskosten pro Montageplatz	BK	44.000 €	213.750 €	52.350 €	52.350 €
<b>Einsparung pro Montageplatz</b>					
Personalkosten Montage (3-Schicht)	3*KW	150.000 €	150.000 €	150.000 €	150.000 €
Automatisierungsgrad	AQ	50 %	100 %	100 %	100 %
Einsparung	AQ*(3*KW)	75.000 €	150.000 €	150.000 €	150.000 €

Zur wirtschaftlichen Bewertung wird die Kapitalwertmethode herangezogen. Hierbei wird der kalkulatorische Zins in die Bewertung mit aufgenommen und zukünftige Cashflows entsprechend angepasst, um den Return-on-Investment (ROI) zu ermitteln. Die sich für die vier Umsetzungsszenarien ergebenden Verläufe der Kapitalwerte sind

## 9 Technische und wirtschaftliche Bewertung

in Abbildung 9.3a dargestellt. Durch die häufige Inanspruchnahme eines Dienstleisters bei Produktänderungen lassen sich die laufenden Betriebskosten des Montagesystems gegenüber den manuellen Montagearbeitsplätzen nicht rechtfertigen.

Eine statische Steuerungslogik im Szenario der Automation kann durch die geringeren Realisierungs- und Betriebskosten wirtschaftlicher umgesetzt werden. Durch das geringere Potenzial zur Automatisierung und somit zur Einsparung von Personalkosten in der Montage lassen sich hierdurch allerdings auch nach fünf Jahren kein positiver ROI erzielen.

Durch die Reduktion der Aufwände mithilfe aufgabenorientierter Systeme können dagegen vergleichbare Realisierungs- und Betriebskosten eines autonomen Montagesystems erreicht werden. Da in diesen Umsetzungsszenarien zudem höhere Einsparungen von Personalkosten erzielt werden, ergeben sich unter den getroffenen Annahmen nach drei Jahren positive ROI. Die Wirtschaftlichkeit hängt aufgrund der Prozessfehlerkosten allerdings stark von der Fehlerquote des Montagesystems ab.

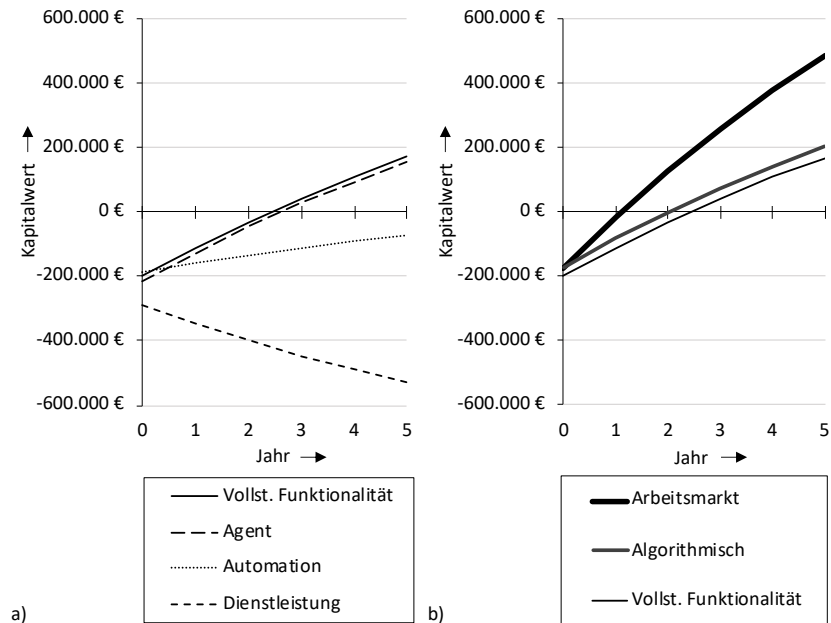


Abbildung 9.3: Verlauf der Kapitalwerte: a) Umsetzungsszenarien b) Zukunftsszenarien

So wird bei der getroffenen Kostenannahme von 100 € pro Fehler bereits ab einer Fehlerquote von 0,6 % kein positiver ROI mehr erzielt. Ab einer Fehlerquote von 1,3 % liegen die Betriebskosten des Montagesystems bereits über den Personalkosten manueller Montageplätze.

Zwei wesentliche Faktoren hinsichtlich der Wirtschaftlichkeit stellen zudem die Fähigkeiten der Agenten und die Personalkosten dar, welche als Zukunftsszenarien in Abbildung 9.3b aufgezeigt werden:

- *Algorithmisch*: Steigende Fähigkeit von Agenten mit Varianz umzugehen und damit verbundener Einsatzbereich (Annahmen: 20 Unternehmen mit je zwei Montageplätzen ohne Notwendigkeit einer Rekonfiguration des Agenten)
- *Arbeitsmarkt*: Steigende Personalkosten für Fachkräfte (Annahme: zusätzliche 50 % Personalkosten)

Die Möglichkeit, eine zunehmende Varianz an Produkten über einen Agenten abbilden zu können, bietet somit weiteres Potenzial zur Steigerung der Wirtschaftlichkeit. Allerdings wird die Wirtschaftlichkeit maßgeblich von zukünftigen Entwicklungen des Arbeitsmarkts abhängen.



## 10 Zusammenfassung und Ausblick

In den vergangenen Jahren hat sich durch die Kombination von Deep Learning und Reinforcement Learning zu Deep Reinforcement Learning ein neuer Ansatz zur Umsetzung intelligenter Montagesysteme entwickelt. Durch das selbstständige Erlernen von Fähigkeiten bzw. Skills unter Varianz wird eine Automatisierung unter variierenden Umgebungsbedingungen und somit auch in KMU ermöglicht. Allerdings setzt die Umsetzung eine noch tiefergehende Expertise in den genannten Disziplinen voraus, welche gerade in KMU meist noch nicht vorhanden ist.

Wie eine Analyse bestehender Ansätze zur rechnergestützten Programmierung von Montagesystemen und zum automatisierten maschinellen Lernen im "Stand der Wissenschaft und Technik" (Kapitel 3) zeigt, lassen sich diese beiden Forschungszweige kombinieren, um eine Umsetzung intelligenter Montagesysteme ohne tiefe Kenntnisse im Bereich von Deep Reinforcement Learning zu ermöglichen.

Hierzu werden Informationsmodelle aus dem Bereich der aufgabenorientierten Programmierung, basierend auf bestehenden Anwendungsfällen aus dem Bereich von Deep Reinforcement Learning, um entsprechende Elemente erweitert, wodurch eine lösungsneutrale Beschreibung der Aufgabenstellung intelligenter Agenten ermöglicht wird.

Mithilfe diesen Aufgabenmodells wird eine Anwendung des wissensbasierten Ansatzes des fallbasierten Schließens ermöglicht. Hierbei wird das Abrufen von Lösungen ähnlicher Aufgabenstellungen innerhalb einer Datenbasis genutzt, um den Nutzer in der Spezifikation der Aufgabenstellung des Agenten zu unterstützen. Neben den Schnittstellen des Agenten zu dessen Umgebung erfolgt hier eine Beschreibung der Agentenaufgabe mittels einer Belohnungsfunktion, welche über Agentenziele abstrahiert wird.

Zusätzlich zu dieser wissensbasierten Entscheidungsunterstützung wird zur Konfiguration des Agenten der Ansatz des Meta Learnings verwendet. Durch das Lernen der Ähnlichkeit der Hyperparameter und semantischen Elemente verschiedener Aufgabenstellungen mithilfe einer siamesischen Netzarchitektur und der weiteren Optimierung von Hyperparameter-Konfigurationen mittels Bayesscher Optimierung kann in dem betrachteten Referenzszenario einer Schweißzelle eine automatische Konfiguration der Agenten realisiert werden.

Während der Ansatz der aufgabenorientierten Konfiguration intelligenter Agenten bereits eine wirtschaftliche Umsetzung intelligenter Montagesysteme ermöglichen kann, ist die Anwendbarkeit des Ansatzes dieser Arbeit noch auf spezifische Anwendungsfälle begrenzt. So wird zum Erlernen eines Agenten-Skills die Simulierbarkeit des Montageprozesses vorausgesetzt. Forschungsansätze zum Schließen dieser sogenannten Realitätslücke (siehe Abschnitt 3.3.2) bieten die Möglichkeit die Anwendungsfälle auszuweiten und die Robustheit intelligenter Montagesysteme zu steigern. Dennoch wird eine ganzheitliche Simulation von Roboterdynamik und Montageprozessen (z. B. Schweißen, Kleben, Beschichten) benötigt. Derzeitige Ansätze zur Kombination verschiedener Simulationsmodelle wie PyBullet Industrial (BAUMGÄRTNER et al. 2023) bieten daher das Potential, mehr Skills innerhalb der Montage durch Agenten abbilden zu können.

Doch auch die Erstellung der Simulationsumgebung selbst stellt einen wesentlichen Aufwand in der Anwendung des Systems dar. Forschungsvorhaben zur automatischen Generierung von 3D-Modellen (z. B. Generierung von 3D-Welten aus Kameradaten in ROHNER et al. 2020) und Auslegung des Layouts (z. B. LEIBER 2023) bieten somit weiteres Einsparungspotential. Zudem ist die Implementierung wiederverwendbarer Simulationsmodule zur Abbildung von Varianz (z. B. Generierung ungeordneter Lagerungen in HOLST et al. 2023) notwendig, um einen wirtschaftlichen Einsatz des Systems in unterschiedlichen Anwendungsfällen zu ermöglichen.

Die verschiedenen Ausprägungen der Varianz werden in der Konfiguration des Agenten innerhalb dieser Arbeit nur teilweise berücksichtigt. Insbesondere Forschungsvorhaben zu einem höheren Grad der Varianz bieten daher die Möglichkeit, die Robustheit des Agenten hinsichtlich unterschiedlicher Aufgabenstellungen (z. B. Produktvarianten) zu steigern und somit die Flexibilität des Robotersystems zu erhöhen.

---

Zudem zeigt sich im Referenzszenario analog zu Vorarbeiten aus der Skill-basierten Programmierung (siehe Abschnitt 3.1.2), dass die Vorgabe einzelner Skills für den Einsatz nicht zwingend ausreicht, sodass weitere Arbeiten zur Definition wiederverwendbarer Skillsequenzen notwendig sind. Auch kann auf Vorarbeiten aus der Skill-basierten Programmierung zurückgegriffen werden, um eine geeignete Zuordnung und Auswahl von Hardware-Komponenten des Robotersystems (z. B. geeignete Sensorsysteme) in Planungsprozess mitabzubilden und somit weitere manuelle Planungsprozesse zu automatisieren.

Auch fokussiert diese Arbeit die Konfiguration des Agenten, da hier die notwendige Expertise im Bereich von Deep Reinforcement Learning als am höchsten angesehen werden kann. Eine Einschließung weiterer Ebenen des Aufgabenmodells (z. B. Parametrisierung der Belohnungsfunktion) ermöglicht eine weitere Reduktion des Aufwands und der erforderlichen Expertise für die Umsetzung solcher intelligenten Montagesysteme. Dieser Ansatz stellt ein aktuelles Forschungsgebiet dar (siehe Abschnitt 3.4.2).

Weitere nicht betrachtete Forschungsfelder im Bereich von DRL stellen die Anpassung bereits umgesetzter Agenten auf neue Aufgabenstellungen – sogenanntes *Transfer Learning* (Z. ZHU et al. 2023) – und die Umsetzung generisch einsetzbarer Agenten – sogenannte *Foundation Models* (YANG et al. 2023) – dar. Insbesondere in der Umsetzung des Agenten sind hierdurch starke Effizienzsteigerungen zu erwarten.

Zusammenfassend zeigt sich, dass die Umsetzung auf DRL basierender Montagesysteme mittels einer aufgabenorientierten Konfiguration des Agenten noch weitere Forschungsvorhaben erfordert, um in weiteren Bereichen der Montage eingesetzt werden zu können und die manuellen Aufwände sowie die Dauer der Umsetzung weiter zu reduzieren.

Allerdings ermöglichen die in dieser Arbeit umgesetzten Funktionalitäten bereits heute eine wirtschaftliche Umsetzung autonomer Montagesysteme. Neben dem Potenzial zukünftiger technologischer Entwicklungen zeigen auch sozioökonomische Entwicklungen wie der zunehmende Fachkräftemangel in Hochlohnländern, dass es sich in der Industrie bereits heute lohnt, sich um die Vereinfachung der Umsetzung autonomer Montagesysteme zu bemühen.





## Literatur

AGRAWAL et al. 2017

Agrawal, A.; M. Dziarski; D. Subburaj; K. West: Design for value and growth in a new world. (2017). URL: <https://www.mckinsey.com/~media/McKinsey/Industries/Consumer%20Packaged%20Goods/Our%20Insights/Design%20for%20value%20and%20growth%20in%20a%20new%20world/Design-for-value-and-growth-in-a-new-world.ashx> (besucht am 08.05.2018).

AHN et al. 2019

Ahn, M.; H. Zhu; K. Hartikainen; H. Ponte; A. Gupta; S. Levine; V. Kumar: ROBEL: RObotics BEnchmarks for Learning with low-cost robots. In: *Conference on Robot Learning (CoRL)*. 2019.

ANAND et al. 2020

Anand, K.; Z. Wang; M. Loog; J. van Gemert: Black Magic in Deep Learning: How Human Skill Impacts Network Training. In: *The 31st British Machine Vision Conference*. 2020.

ANDRYCHOWICZ et al. 2017

Andrychowicz, M.; D. Crow; A. Ray; J. Schneider; R. Fong; P. Welinder; B. McGrew; J. Tobin; P. Abbeel; W. Zaremba: Hindsight Experience Replay. In: *NIPS 2017*. 2017, S. 5055–5065. URL: <https://proceedings.neurips.cc/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf>.

ANDRYCHOWICZ et al. 2020

Andrychowicz, M.; A. Raichuk; P. Stanczyk; M. Orsini; S. Girgin; R. Marinier; L. Hussenot; M. Geist; O. Pietquin; M. Michalski; S. Gelly; O. Bachem: What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study. URL: <https://arxiv.org/pdf/2006.05990>.

## Literatur

---

BACHMANN et al. 2018

Bachmann, A.; M. Röhler; S. J. Pieczona; M. Kessler; M. F. Zaeh: Torque-based adaptive temperature control in friction stir welding: a feasibility study. *Production Engineering* 12 (2018) 3, S. 391–403. ISSN: 1863-7353. DOI: 10.1007/s11740-018-0798-z.

BACKHAUS & REINHART 2017

Backhaus, J.; G. Reinhart: Digital description of products, processes and resources for task-oriented programming of assembly systems. *Journal of Intelligent Manufacturing* 28 (2017) 8, S. 1787–1800. ISSN: 1572-8145. DOI: 10.1007/s10845-015-1063-3.

BACKHAUS 2016

Backhaus, J.: Adaptierbares aufgabenorientiertes Programmiersystem für Montagesysteme. Dissertation. München: Herbert Utz Verlag & Herbert Utz Verlag GmbH. 2016.

BATTAGLIA et al. 2018

Battaglia, P. W.; J. B. Hamrick; V. Bapst; A. Sanchez-Gonzalez; V. Zambaldi; M. Malinowski; A. Tacchetti; D. Raposo; A. Santoro; R. Faulkner; C. Gulcehre; F. Song; A. Ballard; J. Gilmer; G. Dahl; A. Vaswani; K. Allen; C. Nash; V. Langston; C. Dyer; N. Heess; D. Wierstra; P. Kohli; M. Botvinick; O. Vinyals; Y. Li; R. Pascanu: Relational inductive biases, deep learning, and graph networks. URL: <https://arxiv.org/pdf/1806.01261>.

BAUMGÄRTNER et al. 2023

Baumgärtner, J.; M. Hansjosten; D. Schönhofen; P. D.-I. J. Fleischer: PyBullet Industrial: A process-aware robot simulation. *Journal of Open Source Software* 8 (2023) 85, S. 5174. DOI: 10.21105/joss.05174. URL: <https://doi.org/10.21105/joss.05174>.

BEIERLE & KERN-ISBERNER 2014

Beierle, C.; G. Kern-Isberner: *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. 5., überarb. und erw. Aufl. 2014. Springer eBook Collection. Wiesbaden: Springer Vieweg. 2014. ISBN: 9783834823007. DOI: 10.1007/978-3-8348-2300-7.

BEIERLE & KERN-ISBERNER 2019

Beierle, C.; G. Kern-Isberner: *Methoden wissensbasierter Systeme: Grundlagen, Algo-*

---

*rithmen, Anwendungen*. 6., überarbeitete Auflage. Computational intelligence. 2019. ISBN: 9783658270834.

BEJAR & MORAN 2018

Bejar, E.; A. Moran: Deep reinforcement learning based neuro-control for a two-dimensional magnetic positioning system. In: *2018 4th International Conference on Control, Automation and Robotics*. Piscataway, NJ: IEEE Press. 2018, S. 268–273. ISBN: 978-1-5386-6338-7. DOI: 10.1109/ICCAR.2018.8384682.

BELOUSOV et al. 2021

Belousov, B.; H. Abdulsamad; P. Klink: *Reinforcement learning algorithms: Analysis and applications*. Studies in Computational Intelligence. 2021. ISBN: 9783030411879.

BENGIO 2017

Bengio, Y.: The Consciousness Prior. URL: <https://arxiv.org/pdf/1709.08568>.

BENGIO et al. 2009

Bengio, Y.; J. Louradour; R. Collobert; J. Weston: Curriculum learning. In: *Proceedings / Twenty-Sixth International Conference on Machine Learning*. Hrsg. von Bottou, L. Madison. 2009, S. 1–8. ISBN: 9781605585161. DOI: 10.1145/1553374.1553380.

BERG 2020

Berg, J. K.: System zur aufgabenorientierten Programmierung für die Mensch-Roboter-Kooperation. Dissertation. München: Technische Universität München. 2020.

BERGSTRA et al. 2011

Bergstra, J.; R. Bardenet; Y. Bengio; B. Kégl: Algorithms for Hyper-Parameter Optimization. In: *Advances in Neural Information Processing Systems*. Hrsg. von J. Shawe-Taylor; R. Zemel; P. Bartlett; F. Pereira; K. Q. Weinberger. Bd. 24. Curran Associates, Inc. 2011. URL: <https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf>.

BERSCHIED et al. 2019

Berscheid, L.; P. Meissner; T. Kroger: Robot Learning of Shifting Objects for Grasping in Cluttered Environments. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, S. 612–618. ISBN: 978-1-7281-4004-9. DOI: 10.1109/IROS40897.2019.8968042.

## Literatur

---

BHARADHWAJ et al. 2019

Bharadhwaj, H.; Z. Wang; Y. Bengio; L. Paull: A Data-Efficient Framework for Training and Sim-to-Real Transfer of Navigation Policies. In: *2019 International Conference on Robotics and Automation (ICRA)*. [Piscataway, NJ]: IEEE. 2019, S. 782–788. ISBN: 978-1-5386-6027-0. DOI: 10.1109/ICRA.2019.8794310.

BLESSING & CHAKRABARTI 2009

Blessing, L. T.; A. Chakrabarti: DRM, a Design Research Methodology. DOI: 10.1007/978-1-84882-587-1.

BOTVINICK et al. 2019

Botvinick, M.; S. Ritter; J. X. Wang; Z. Kurth-Nelson; C. Blundell; D. Hassabis: Reinforcement Learning, Fast and Slow. *Trends in Cognitive Sciences* (2019). ISSN: 13646613. DOI: 10.1016/j.tics.2019.02.006.

BOURDONNAYE et al. 2018

Bourdonnaye, F. c. d. L.; C. Teulière; J. Triesch; T. Chateau: Learning to touch objects through stage-wise deep reinforcement learning. URL: <https://hal.archives-ouvertes.fr/hal-01820043>.

BRAUN et al. 2021

Braun, J. von; M. S. Archer; G. M. Reichberg; M. Sánchez Sorondo, Hrsg. (2021): *Robotics, AI, and humanity: Science, ethics, and policy*. Cham, Switzerland: Springer Nature. 2021. ISBN: 303054172X.

BRECHER & ÖZDEMİR 2017

Brecher, C.; D. Özdemir, Hrsg. (2017): *Integrative production technology: Theory and applications*. Cham, Switzerland: Springer. 2017. ISBN: 978-3-319-47451-9. URL: <http://www.springer.com/>.

BREYER et al. 2018

Breyer, M.; F. Furrer; T. Novkovic; R. Siegwart; J. I. Nieto: Flexible Robotic Grasping with Sim-to-Real Transfer based Reinforcement Learning. *CoRR* abs/1803.04996 (2018).

BROCKMAN et al. 2016

Brockman, G.; V. Cheung; L. Pettersson; J. Schneider; J. Schulman; J. Tang; W. Zaremba: OpenAI Gym. URL: <https://arxiv.org/pdf/1606.01540>.

BRUIN et al. 2016

Bruin, T. de; J. Kober; K. Tuyls; R. Babuska: Improved deep reinforcement learning for robotics through distribution-based experience retention. In: *IROS 2016*. Piscataway, NJ: IEEE. 2016, S. 3947–3952. ISBN: 978-1-5090-3762-9. DOI: 10.1109/IROS.2016.7759581.

BRULAND et al. 2010

Bruland, T.; A. Aamodt; H. Langseth: Architectures Integrating Case-Based Reasoning and Bayesian Networks for Clinical Decision Support. In: *Intelligent Information Processing V*. Hrsg. von Shi, Z.; Vadera, S.; Aamodt, A.; Leake, D. Berlin, Heidelberg: Springer Berlin Heidelberg. 2010, S. 82–91. ISBN: 978-3-642-16327-2.

BRUNTON & KUTZ 2019

Brunton, S. L.; J. N. Kutz: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press. 2019. DOI: 10.1017/9781108380690.

BRYS et al. 2017

Brys, T.; A. Harutyunyan; P. Vrancx; A. Nowé; M. E. Taylor: Multi-objectivization and ensembles of shapings in reinforcement learning. *Neurocomputing* 263 (2017) 2, S. 48–59. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.02.096.

BUBECK et al. 2014

Bubeck, A.; F. Weisshardt; A. Verl: BRIDE - A toolchain for framework-independent development of industrial service robot applications. In: *ISR/Robotik 2014; 41st International Symposium on Robotics*. 2014, S. 1–6.

BURKOV 2019

Burkov, A.: *The hundred-page machine learning book*. 2019. ISBN: 9781999579500.

CHEBOTAR et al. 2017

Chebotar, Y.; K. Hausman; M. Zhang; G. Sukhatme; S. Schaal; S. Levine: Combining Model-Based and Model-Free Updates for Trajectory-Centric Reinforcement Learning. *ArXiv e-prints* (2017).

CHOLLET 2019

Chollet, F.: The Measure of Intelligence. URL: <http://arxiv.org/pdf/1911.01547v1>.

## Literatur

---

CHOLLET 2021

Chollet, F.: *Deep Learning with Python*. 2. Edition. Manning Publications. 2021.

CIRP 2020

CIRP: *Wörterbuch der Fertigungstechnik IV - Montage*. Berlin, Heidelberg: Springer Vieweg. 2020. ISBN: 978-3-662-53341-3.

CSISZAR et al. 2017

Csiszar, A.; M. Hoppe; S. A. Khader; A. Verl: Behavior Trees for Task-Level Programming of Industrial Robots. In: *Tagungsband des 2. Kongresses Montage Handhabung Industrieroboter*. Hrsg. von Schüppstuhl, T.; Franke, J.; Tracht, K. Berlin: Springer Vieweg. 2017, S. 175–186. ISBN: 978-3-662-54440-2. DOI: 10.1007/978-3-662-54441-9\_18.

DALAL et al. 2018

Dalal, G.; K. Dvijotham; M. Vecerik; T. Hester; C. Paduraru; Y. Tassa: Safe Exploration in Continuous Action Spaces. URL: <https://arxiv.org/pdf/1801.08757>.

DEVLIN & KUDENKO 2016

Devlin, S.; D. Kudenko: Plan-based reward shaping for multi-agent reinforcement learning. *The Knowledge Engineering Review* 31 (2016) 01, S. 44–58. ISSN: 0269-8889. DOI: 10.1017/S0269888915000181.

DEWEY 2014

Dewey, D.: Reinforcement Learning and the Reward Engineering Principle. In: *AAAI Spring Symposium*. 2014. URL: <https://www.aaai.org/ocs/index.php/SSS/SSS14/paper/view/7704/7740>.

DHAWAN et al. 2018

Dhawan, R.; B. Heid; P. Kürderli; K. Laczkowski: Disruptive forces in the industrial sectors: Global executive survey. (2018). (Besucht am 13. 04. 2018).

DIN 2020

DIN: Normungsroadmap Künstliche Intelligenz: Handlungsempfehlungen zur KI-Standardisierung.

DÖBEL et al. 2018

Döbel, I.; M. Leis; M. Molina Vogelsang; J. Welz; D. Neustroev; H. Petzka; A. Riemer;

S. Püping; A. Voss; M. Wegele: Maschinelles Lernen: Eine Analyse zu Kompetenzen, Forschung und Anwendung. URL: [https://www.bigdata.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/Fraunhofer\\_Studie\\_ML\\_201809.pdf](https://www.bigdata.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/Fraunhofer_Studie_ML_201809.pdf) (besucht am 09. 09. 2020).

DONG et al. 2020

Dong, H.; Z. Ding; S. Zhang, Hrsg. (2020): *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Singapore: Springer Singapore Pte. Limited. 2020. ISBN: 9789811540950. URL: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6241464>.

DRATH 2010

Drath, R.: *Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA*. VDI-Buch. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg. 2010. ISBN: 9783642046735. DOI: 10.1007/978-3-642-04674-2. URL: <http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10358615>.

DU et al. 2021

Du, Y.; O. Watkins; T. Darrell; P. Abbeel; D. Pathak: Auto-Tuned Sim-to-Real Transfer. URL: <https://arxiv.org/pdf/2104.07662>.

ECOFFET et al. 2021

Ecoffet, A.; J. Huizinga; J. Lehman; K. O. Stanley; J. Clune: First return, then explore. *Nature* 590 (2021) 7847, S. 580–586. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03157-9.

EHRMANN 2007

Ehrmann, M.: *Beitrag zur Effizienzsteigerung bei der Programmierung flexibler, roboterbasierter Montagezellen: Konzeption und Realisierung eines nutzergerechten Programmiersystems: Zugl.: Kaiserslautern, Techn. Univ., Diss., 2007*. Als Ms. gedr. Bd. 15. Fortschritt-Berichte pak Robotik. Kaiserslautern: Techn. Univ. 2007. ISBN: 3939432385.

ELMARAGHY 2005

ElMaraghy, H. A.: Flexible and reconfigurable manufacturing systems paradigms. *In-*

## Literatur

---

*ternational Journal of Flexible Manufacturing Systems* 17 (2005) 4, S. 261–276. ISSN: 0920-6299. DOI: 10.1007/s10696-006-9028-7.

ELMARAGHY 2009

ElMaraghy, H. A., Hrsg. (2009): *Changeable and Reconfigurable Manufacturing Systems*. London: Springer London. 2009. ISBN: 978-1-84882-067-8.

ENGSTROM et al. 2020

Engstrom, A.; A. Ilyas; S. Santurkar; D. Tsipras; F. Janoos; L. Rudolph; A. madry: Implementation Matters in Deep RL: A Case Study on PPO and TRPO. In: *International Conference International Conference on Learning Representations 2020*. 2020. URL: <https://openreview.net/forum?id=r1etN1rtPB> (besucht am 23.03.2020).

ENNEN et al. 2019

Ennen, P.; E. Pabich; R. Kupper; P. Benmoussa; R. Vossen; S. Jeschke: Leitfaden Selbstlernende Produktionsprozesse: Einführungsstrategie für Reinforcement Learning in der industriellen Praxis. Hrsg. von VDMA.

EUROPÄISCHE KOMMISSION 2019

Europäische Kommission: A Definition of AI: Main Capabilities and Dsiciplines. URL: [https://ec.europa.eu/newsroom/dae/document.cfm?doc\\_id=56341](https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=56341) (besucht am 29.11.2019).

EUROPÄISCHE KOMMISSION 2021

Europäische Kommission: Proposal for a Regulation of the European Parliament and the Council.

EVERITT et al. 2018

Everitt, T.; G. Lea; M. Hutter: AGI Safety Literature Review. URL: <http://arxiv.org/pdf/1805.01109v2>.

EVERSHEIM & SCHUH 2005

Eversheim, W.; G. Schuh: *Integrierte Produkt- und Prozessgestaltung*. VDI. Berlin: Springer. 2005. ISBN: 3540211756. URL: <http://lib.mylibrary.com/detail.asp?id=61614>.



FALKNER et al. 2018

Falkner, S.; A. Klein; F. Hutter: BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In: *Proceedings of the 35th International Conference on Machine Learning*. Hrsg. von Dy, J.; Krause, A. Bd. 80. Proceedings of Machine Learning Research. PMLR. 2018, S. 1437–1446. URL: [https://www.ki-engineering.eu/content/dam/iosb/ki-engineering/know-how/paise/PAISE\(R\)\\_Whitepaper\\_CC-KING.pdf](https://www.ki-engineering.eu/content/dam/iosb/ki-engineering/know-how/paise/PAISE(R)_Whitepaper_CC-KING.pdf).

FAUST et al. 2019

Faust, A.; A. Francis; D. Mehta: Evolving Rewards to Automate Reinforcement Learning. In: *6th ICML Workshop on Automated Machine Learning*. 2019.

FELDMANN 2004

Feldmann, K., Hrsg. (2004): *Montage strategisch ausrichten: Praxisbeispiele marktorientierter Prozesse und Strukturen*. Engineering online library. Berlin: Springer. 2004. ISBN: 3540403043.

FENSEL et al. 2020

Fensel, D.; U. Simsek; K. Angele: *Knowledge graphs: Methodology, tools and selected use cases*. 2020. ISBN: 9783030374389.

FEURER et al. 2015

Feurer, M.; J. T. Springenberg; F. Hutter: Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. AAAI Press. 2015, S. 1128–1135. ISBN: 0262511290.

FEURER et al. 2020

Feurer, M.; K. Eggenberger; S. Falkner; M. Lindauer; F. Hutter: Auto-Sklearn 2.0: The Next Generation. URL: <https://arxiv.org/pdf/2007.04074>.

FJELLAND 2020

Fjelland, R.: Why general artificial intelligence will not be realized. *Humanities and Social Sciences Communications* 7 (2020) 1. DOI: 10.1057/s41599-020-0494-4.

FLORENSA et al. 2017

Florensa, C.; D. Held; M. Wulfmeier; P. Abbeel: Reverse Curriculum Generation for Reinforcement Learning. *CoRR* abs/1707.05300 (2017).

## Literatur

---

FRANCESCHETTI et al. 2020

Franceschetti, A.; E. Tosello; N. Castaman; S. Ghidoni: Robotic Arm Control and Task Training through Deep Reinforcement Learning. URL: <https://arxiv.org/pdf/2005.02632>.

FRANÇOIS-LAVET et al. 2018

François-Lavet, V.; P. Henderson; R. Islam; M. G. Bellemare; J. Pineau: An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning* 11 (2018) 3-4, S. 219–354. ISSN: 1935-8237. DOI: 10.1561/22000000071.

FRANKE et al. 2020

Franke, J. K. H.; G. Köhler; A. Biedenkapp; F. Hutter: Sample-Efficient Automated Deep Reinforcement Learning. URL: <https://arxiv.org/pdf/2009.01555>.

FRICKE & SCHULZ 2005

Fricke, E.; A. P. Schulz: Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle. *Systems Engineering* 8 (2005) 4, S. 169. ISSN: 1098-1241. DOI: 10.1002/sys.20039.

FUJIMOTO et al. 2017

Fujimoto, R. M.; C. Bock; W. Chen; E. Page; J. H. Panchal: *Research Challenges in Modeling and Simulation for Engineering Complex Systems*. Simulation foundations, methods and applications. Cham: Springer. 2017. ISBN: 3319585444.

GARCEZ & LAMB 2020

Garcez, A. d.; L. C. Lamb: Neurosymbolic AI: The 3rd Wave. URL: [https://www.fortiss.org/fileadmin/user\\_upload/Veroeffentlichungen/Informationsmaterialien/fortiss\\_whitepaper\\_trustworthy\\_ACS\\_web.pdf](https://www.fortiss.org/fileadmin/user_upload/Veroeffentlichungen/Informationsmaterialien/fortiss_whitepaper_trustworthy_ACS_web.pdf).

GATTI 2015

Gatti, C.: *Design of Experiments for Reinforcement Learning*. Cham: Springer International Publishing, 2015. ISBN: 978-3-319-12196-3. DOI: 10.1007/978-3-319-12197-0.

GAUCI et al. 2018

Gauci, J.; E. Conti; Y. Liang; K. Virochsiri; Y. He; Z. Kaden; V. Narayanan; X. Ye;

Z. Chen: Horizon: Facebook's Open Source Applied Reinforcement Learning Platform. URL: <http://arxiv.org/pdf/1811.00260v4>.

GEFFNER 2018

Geffner, H.: Model-free, Model-based, and General Intelligence. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. Hrsg. von Rosenschein, J. S.; Lang, J. California: International Joint Conferences on Artificial Intelligence Organization. 2018, S. 10–17. ISBN: 9780999241127. DOI: 10.24963/ijcai.2018/2.

GEHLHOFF et al. 2019

Gehlhoff, F.; A. Fay; B. Vogel-Heuser; M. Seitz; D. Ryashentseva: Evaluation der Flexibilität und Rekonfigurierbarkeit von Produktionssystemen – Ein quantitativer Ansatz auf Basis des Produkt-Prozess-Ressource-Konzeptes. In: *Automation 2019*. VDI Verlag. 2019, S. 629–646. ISBN: 9783181023518. DOI: 10.51202/9783181023518-629.

GÉRON 2018

Géron, A.: *Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*. 1. Auflage. Animals. Heidelberg: O'Reilly. 2018. ISBN: 9783960101147. URL: <https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=5215589>.

GHADIRZADEH et al. 2017

Ghadirzadeh, A.; A. Maki; D. Kragic; M. Bjorkman: Deep predictive policy training using reinforcement learning. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, S. 2351–2358. ISBN: 978-1-5386-2682-5. DOI: 10.1109/IROS.2017.8206046.

GONNERMANN et al. 2020

Gonnermann, C.; J. Weth; G. Reinhart: Skill Modeling in Cyber-Physical Production Systems for Process Monitoring. *Procedia CIRP* 93 (2020), S. 1376–1381. ISSN: 22128271. DOI: 10.1016/j.procir.2020.03.095.

GOODFELLOW et al. 2018

Goodfellow, I.; A. Courville; Y. Bengio: *Deep Learning: Das umfassende handbuch : Grundlagen, aktuelle verfahren und Algorithmen, neue forschungsansätze*. 1. Auflage. Frechen: Verlags GmbH & Co. KG. 2018. ISBN: 9783958457003. URL: <http://>

## Literatur

---

search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=1924753.

GÖTZ 2017

Götz, G. A. J.: Methode zur Steigerung der Formatflexibilität von Verpackungsmaschinen. Dissertation. München: Technische Universität München. 2017.

GRAESSER & KENG 2019

Graesser, L.; L. W. Keng: *Foundations of deep reinforcement learning: Theory and practice in Python*. First edition. The Pearson Addison-Wesley data and analytics series. Addison-Wesley Professional. 2019. ISBN: 9780135172384.

GRIFFITHS 2020

Griffiths, T. L.: Understanding Human Intelligence through Human Limitations. *Trends in Cognitive Sciences* 24 (2020) 11, S. 873–883. ISSN: 13646613. DOI: 10.1016/j.tics.2020.09.001.

GU et al. 2017

Gu, S.; E. Holly; T. Lillicrap; S. Levine: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, S. 3389–3396. DOI: 10.1109/ICRA.2017.7989385.

GUDIMELLA et al. 2017

Gudimella, A.; R. Story; M. Shaker; R. Kong; M. Brown; V. Shnayder; M. Campos: Deep Reinforcement Learning for Dexterous Manipulation with Concept Networks. *CoRR abs/1709.06977* (2017).

GUO 2017

Guo, X.: Deep Learning and Reward Design for Reinforcement Learning. Diss. Michigan: University of Michigan. 2017. URL: [https://deepblue.lib.umich.edu/bitstream/handle/2027.42/136931/guoxiao\\_1.pdf?sequence=1&isAllowed=y](https://deepblue.lib.umich.edu/bitstream/handle/2027.42/136931/guoxiao_1.pdf?sequence=1&isAllowed=y) (besucht am 07.09.2018).

HA et al. 2018

Ha, S.; J. Kim; K. Yamane: Automated Deep Reinforcement Learning Environment for Hardware of a Modular Legged Robot. In: *2018 15th International Conference on*

*Ubiquitous Robots (UR)*. Piscataway, NJ: IEEE. 2018, S. 348–354. ISBN: 978-1-5386-6334-9. DOI: 10.1109/URAI.2018.8442201.

HAARNOJA et al. 2018

Haarnoja, T.; V. Pong; A. Zhou; M. Dalal; P. Abbeel; S. Levine: Composable Deep Reinforcement Learning for Robotic Manipulation. *ArXiv e-prints* (2018).

HABERFELLNER et al. 2019

Haberfellner, R.; O. de Weck; E. Fricke; S. Vössner: *Systems engineering: Fundamentals and applications*. Cham: Birkhäuser. 2019. ISBN: 3030134318.

HALT et al. 2018

Halt, L.; F. Nagele; P. Tenbrock; A. Pott: Intuitive Constraint-Based Robot Programming for Robotic Assembly Tasks\* The research leading to these results has received funding from the European Unions Seventh Framework Programme FP7/2013-2017 under grant agreement n 608604 (LIAA: Lean Intelligent Assembly Automation) and Horizon 2020 Research and Innovation Programme under grant agreement n 688642 (RAMPup). In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Hrsg. von Lynch, K. Piscataway, NJ: IEEE. 2018, S. 520–526. ISBN: 978-1-5386-3081-5. DOI: 10.1109/ICRA.2018.8462882.

HAMMERSTINGL et al. 2015

Hammerstingl, V.; L. Moule; G. Reinhart: Bildverarbeitungssysteme als cyber-physische Sensoren. In: *at Automatisierungstechnik 11/2015*. 2015.

HAMMERSTINGL 2020

Hammerstingl, V.: Steigerung der Rekonfigurationsfähigkeit von Montageanlagen durch Cyber-physische Feldgeräte. Dissertation. München: Technische Universität München. 2020.

HAMMERSTINGL & REINHART 2017

Hammerstingl, V.; G. Reinhart: Fähigkeiten in der Montage. Diss. München: Technische Universität München. 2017. DOI: 10.13140/RG.2.2.22520.75526. URL: <http://mediatum.ub.tum.de/?id=1370174>.

HANINGER et al. 2020

Haninger, K.; R. Vicente-Garcia; J. Krüger: Towards Learning Controllable Representa-

## Literatur

---

tions of Physical Systems. *CoRR* abs/2011.09906 (2020). arXiv: 2011.09906. URL: <https://arxiv.org/abs/2011.09906>.

HAO 2021

Hao, K.: A new generation of AI-powered robots is taking over warehouses. *MIT Technology Review* (2021). URL: <https://www.technologyreview.com/2021/08/06/1030802/ai-robots-take-over-warehouses/> (besucht am 09.09.2021).

HAUN 2013

Haun, M.: *Handbuch Robotik: Programmieren und Einsatz intelligenter Roboter*. 2. Auflage. VDI-Buch. Berlin & Heidelberg: Springer Vieweg, 2013. ISBN: 9783642398582. DOI: 10.1007/978-3-642-39858-2. URL: <http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=818416>.

HEES 2017

Hees, A. F.: System zur Produktionsplanung für rekonfigurierbare Produktionssysteme. Dissertation. München: Technische Universität München, 2017.

HEESS et al. 2017

Heess, N.; D. TB; S. Sriram; J. Lemmon; J. Merel; G. Wayne; Y. Tassa; T. Erez; Z. Wang; S. M. A. Eslami; M. Riedmiller; D. Silver: Emergence of Locomotion Behaviours in Rich Environments. URL: <https://arxiv.org/pdf/1707.02286>.

HENDERSON et al. 2017

Henderson, P.; R. Islam; P. Bachman; J. Pineau; D. Precup; D. Meger: Deep Reinforcement Learning that Matters. *CoRR* abs/1709.06560 (2017).

HESTER et al. 2017

Hester, T.; M. Vecerík; O. Pietquin; M. Lanctot; T. Schaul; B. Piot; A. Sendonaris; G. Dulac-Arnold; I. Osband; J. Agapiou; J. Leibo; A. Gruslys: Learning from Demonstrations for Real World Reinforcement Learning. *CoRR* abs/1704.03732 (2017).

HOLST et al. 2023

Holst, D.; D. Schoepflin; T. Schüppstuhl: Generation of Synthetic AI Training Data for Robotic Grasp-Candidate Identification and Evaluation in Intralogistics Bin-Picking Scenarios. In: *Flexible Automation and Intelligent Manufacturing: The Human-Data-*

*Technology Nexus*. Hrsg. von Kim, K.-Y.; Monplaisir, L.; Rickli, J. Cham: Springer International Publishing. 2023, S. 284–292. ISBN: 978-3-031-18326-3.

HOWARD et al. 2020

Howard, J.; S. Gugger; S. Chintala: *Deep learning for coders with fastai and PyTorch: AI applications without a PhD*. First edition. Sebastopol CA: O'Reilly. 2020. ISBN: 1492045527.

HU et al. 2020

Hu, Y.; W. Wang; H. Jia; Y. Wang; Y. Chen; J. Hao; F. Wu; C. Fan: Learning to Utilize Shaping Rewards: A New Approach of Reward Shaping. URL: <https://arxiv.org/pdf/2011.02669>.

HUA et al. 2016

Hua, Y.; S. Zander; M. Bordignon; B. Hein: From AutomationML to ROS: A model-driven approach for software engineering of industrial robotics using ontological reasoning. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2016, S. 1–8. DOI: 10.1109/ETFA.2016.7733579.

HUA et al. 2017

Hua, Y.; M. Mende; B. Hein: Modulare und Wandlungsfähige Robotersysteme: Modellbasierte Softwareentwicklung basierend auf AutomationML und ontologischer Semantik. *Industrie 4.0 Management* (2017) 6, S. 33–37. ISSN: 1434-1700.

HUMBURGER 1998

Humburger, R.: *Konzeption eines Systems zur aufgabenorientierten Roboterprogrammierung: Zugl.: Aachen, Techn. Hochsch., Diss., 1997*. Als Ms. gedr. Bd. 98,4. Berichte aus der Produktionstechnik. Aachen: Shaker. 1998. ISBN: 3826534271.

HUTSON 2018

Hutson, M.: AI researchers allege that machine learning is alchemy. *Science* (2018). ISSN: 0036-8075. DOI: 10.1126/science.aau0577.

F. HUTTER et al. 2019

Hutter, F.; L. Kotthoff; J. Vanschoren: *Automated Machine Learning*. Cham: Springer International Publishing. 2019. ISBN: 978-3-030-05317-8. DOI: 10.1007/978-3-030-05318-5.

## Literatur

---

HWANGBO et al. 2019

Hwangbo, J.; J. Lee; A. Dosovitskiy; D. Bellicoso; V. Tsounis; V. Koltun; M. Hutter: Learning agile and dynamic motor skills for legged robots. *Science Robotics* 4 (2019) 26, eaau5872. DOI: 10.1126/scirobotics.aau5872.

IBARZ et al. 2021

Ibarz, J.; J. Tan; C. Finn; M. Kalakrishnan; P. Pastor; S. Levine: How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research* (2021), S. 027836492098785. ISSN: 0278-3649. DOI: 10.1177/0278364920987859.

IEC 2021

IEC: Artificial intelligence across industries: White Paper.

IFR 2018a

IFR: Artificial Intelligence in Robotics: Media Backgrounder. Frankfurt am Main.

IFR 2018b

IFR: Robots and the Workplace of the Future: Positioning Paper. Frankfurt am Main.

IFR 2020

IFR: Next Generation Skills. Enabling today's and tomorrow's workforce to benefit from automation. Positioning Paper. Frankfurt am Main.

INOUE et al. 2017

Inoue, T.; G. de Magistris; A. Munawar; T. Yokoya; R. Tachibana: Deep reinforcement learning for high precision assembly tasks. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, S. 819–825. ISBN: 978-1-5386-2682-5. DOI: 10.1109/IROS.2017.8202244.

IRIONDO et al. 2019

Iriondo, A.; E. Lazkano; L. Susperregi; J. Urain; A. Fernandez; J. Molina: Pick and Place Operations in Logistics Using a Mobile Manipulator Controlled with Deep Reinforcement Learning. *Applied Sciences* 9 (2019) 2, S. 348. DOI: 10.3390/app9020348.

IRPAN et al. o. D.

Irpan, A.; K. Rao; K. Bousmalis; C. Harris; J. Ibarz; S. Levine: Off-Policy Evaluation via Off-Policy Classification. URL: <http://arxiv.org/pdf/1906.01624v2>.



ISO/CD 22989

ISO/CD 22989: Information technology - Artificial intelligence - Artificial intelligence concepts and terminology.

JACOBSSON et al. 2016

Jacobsson, L.; J. Malec; K. Nilsson: Modularization of skill ontologies for industrial robots. In: *Proceedings of ISR 2016: 47th International Symposium on Robotics*. 2016, S. 1–6.

JAMES & JOHNS 2016

James, S.; E. Johns: 3D Simulation for Robot Arm Control with Deep Q-Learning. *CoRR* abs/1609.03759 (2016).

JAMES et al. 2018

James, S.; P. Wohlhart; M. Kalakrishnan; D. Kalashnikov; A. Irpan; J. Ibarz; S. Levine; R. Hadsell; K. Bousmalis: Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks. URL: <http://arxiv.org/pdf/1812.07252v1>.

JÄRVENPÄÄ et al. 2019

Järvenpää, E.; N. Siltala; O. Hylli; M. Lanz: The development of an ontology for describing the capabilities of manufacturing resources. *Journal of Intelligent Manufacturing* 30 (2019) 2, S. 959–978. ISSN: 1572-8145. DOI: 10.1007/s10845-018-1427-6.

JESCHKE et al. 2017

Jeschke, S.; C. Brecher; H. Song: *Industrial Internet of Things: Cybermanufacturing Systems*. Springer Series in Wireless Technology. 2017. ISBN: 978-3-319-42559-7. URL: <http://dx.doi.org/10.1007/978-3-319-42559-7>.

JOHANNISMEIER et al. 2018

Johannismeier, L.; M. Gerchow; S. Haddadin: A Framework for Robot Manipulation: Skill Formalism, Meta Learning and Adaptive Control. URL: <https://arxiv.org/pdf/1805.08576>.

JORDAN 2019

Jordan, M. I.: Artificial Intelligence - The Revolution Hasn't Happened Yet. *Harvard Data Science Review* 1 (2019) 1. DOI: 10.1162/99608f92.f06c6e61. URL: <https://hdsr.mitpress.mit.edu/pub/wot7mkc1>.

## Literatur

---

JULIANI et al. 2018

Juliani, A.; V.-P. Berges; E. Vckay; Y. Gao; H. Henry; M. Mattar; D. Lange: Unity: A General Platform for Intelligent Agents. *ArXiv e-prints* (2018).

KAHNEMAN 2012

Kahneman, D.: *Schnelles Denken, langsames Denken*. 5. Aufl. München: Siedler. 2012. ISBN: 3886808866.

KALASHNIKOV et al. 2018

Kalashnikov, D.; A. Irpan; P. Pastor; J. Ibarz; A. Herzog; E. Jang; D. Quillen; E. Holly; M. Kalakrishnan; V. Vanhoucke; S. Levine: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. In: *Proceedings of The 2nd Conference on Robot Learning*. Hrsg. von Billard, A.; Dragan, A.; Peters, J.; Morimoto, J. Bd. 87. Proceedings of Machine Learning Research. PMLR. 2018, S. 651–673. URL: <http://proceedings.mlr.press/v87/kalashnikov18a.html>.

KALUZA et al. 2005

Kaluza, B.; T. Blecker; S. Behrens, Hrsg. (2005): *Erfolgsfaktor Flexibilität: Strategien und Konzepte für wandlungsfähige Unternehmen*. Bd. 60. Technological economics. Berlin: Schmidt. 2005. ISBN: 3503083677.

KENG et al. 2019

Keng, L. W.; L. Graesser; M. Cvitkovic: SLM Lab: A Comprehensive Benchmark and Modular Software Fmework for Reproducible Deep Reinforcement Learning. URL: <https://arxiv.org/pdf/1912.12482>.

KIEFER 2020

Kiefer, L.: Gestaltung und Planung von merkmals- und agentenbasierten Selbststeuerungssystemen für die autonome Produktion. Dissertation. München: Technische Universität München. 2020.

J. KIM et al. 2017

Kim, J.; S. Kim; S. Choi: Learning to Warm-Start Bayesian Hyperparameter Optimization. URL: <https://arxiv.org/pdf/1710.06219>.

W. KIM et al. 2017

Kim, W.; T. Kim; J. Lee; H. J. Kim: Vision-based deep reinforcement learning to control a manipulator. In: *2017 Asian Control Conference Gold Coast, Australia*. Piscataway,

NJ: IEEE. 2017, S. 1046–1050. ISBN: 978-1-5090-1573-3. DOI: 10.1109/ASCC.2017.8287315.

KINDLE et al. 2020

Kindle, J.; F. Furrer; T. Novkovic; J. J. Chung; R. Siegwart; J. Nieto: Whole-Body Control of a Mobile Manipulator using End-to-End Reinforcement Learning. URL: <https://arxiv.org/pdf/2003.02637>.

KLEEBERGER et al. 2020

Kleeberger, K.; R. Bormann; W. Kraus; M. F. Huber: A Survey on Learning-Based Robotic Grasping. *Current Robotics Reports* 1 (2020) 4, S. 239–249. DOI: 10.1007/s43154-020-00021-6.

KOBER et al. 2013

Kober, J.; D. Bagnell; J. Peters: Reinforcement Learning in Robotics: A Survey. (2013) 11, S. 1238–1274. URL: [http://www.ias.tu-darmstadt.de/uploads/Publications/Kober\\_IJRR\\_2013.pdf](http://www.ias.tu-darmstadt.de/uploads/Publications/Kober_IJRR_2013.pdf) (besucht am 02.01.2018).

KRAMER 2009

Kramer, O.: *Computational Intelligence: Eine Einführung*. Informatik im Fokus. Berlin: Springer. 2009. ISBN: 9783540797388. DOI: 10.1007/978-3-540-79739-5. URL: <http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10288881>.

KREIMEYER & LINDEMANN 2011

Kreimeyer, M.; U. Lindemann: *Complexity Metrics in Engineering Design*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2011. ISBN: 978-3-642-20962-8. DOI: 10.1007/978-3-642-20963-5.

KRIZHEVSKY et al. 2012

Krizhevsky, A.; I. Sutskever; G. E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems*. Hrsg. von F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger. Bd. 25. Curran Associates, Inc. 2012. URL: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.

KROTIL 2017

Krottil, S.: Online-Simulation von fluidischen Prozessen in der frühen Phase der

## Literatur

---

Maschinen- und Anlagenentwicklung. Diss. München: Herbert Utz Verlag & Herbert Utz Verlag GmbH. 2017.

KRUG 2012

Krug, S.: Automatische Konfiguration von Robotersystemen (Plug&Produce). Dissertation. München: Technische Universität München. 2012.

KUBICEK 1977

Kubicek, H.: Heuristischer Bezugsrahmen und heuristisch angelegte Forschungsdesigns als Elemente einer Konstruktionsstrategie empirischer Forschung. In: *Empirische und handlungstheoretische Forschungskonzeptionen in der Betriebswirtschaftslehre*. Hrsg. von Köhler, R. Stuttgart: C. E. Poeschel. 1977.

KUGELMANN 1999

Kugelmann, D.: *Aufgabenorientierte Offline-Programmierung von Industrierobotern: Zugl.: München, Techn. Univ., Diss., 1999*. Bd. 127. Forschungsberichte / IWB. München: Utz Wiss. 1999. ISBN: 389675615X.

KUSS et al. 2016

Kuss, A.; U. Schneider; T. Dietz; A. Verl: Detection of Assembly Variations for Automatic Program Adaptation in Robotic Welding Systems. In: *Proceedings of ISR 2016: 47st International Symposium on Robotics*. 2016, S. 1–6.

KUSS 2020

Kuss, A.: Aufgabenorientierte Roboterprogrammierung unter Berücksichtigung von geometrischen Bauteilabweichungen beim Lichtbogenschweißen. DOI: 10.18419/OPUS-11247.

LAKE et al. 2017

Lake, B. M.; T. D. Ullman; J. B. Tenenbaum; S. J. Gershman: Building machines that learn and think like people. *The Behavioral and brain sciences* 40 (2017), e253. DOI: 10.1017/S0140525X16001837.

LAKSHMANAN et al. 2020

Lakshmanan, V.; S. Robinson; M. Munn: *Machine learning design patterns: Solutions to common challenges in data preparation, model building, and MLOps*. First edition. Sebastopol, CA: O'Reilly. 2020. ISBN: 9781098115753. URL: <https://>

ebookcentral.proquest.com/lib/kxp/detail.action?docID=6372578.

LÄMMLE et al. 2020

Lämmle, A.; T. König; M. El-Shamouty; M. F. Huber: Skill-based Programming of Force-controlled Assembly Tasks using Deep Reinforcement Learning. *Procedia CIRP* 93 (2020), S. 1061–1066. ISSN: 22128271. DOI: 10.1016/j.procir.2020.04.153.

LAUD 2004

Laud, A.: Theory and Application of Reward Shaping in Reinforcement Learning. Dissertation. Urbana: University of Illinois at Urbana-Champaign. 2004. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.175.226&rep=rep1&type=pdf> (besucht am 09.07.2018).

LECUN et al. 2015

LeCun, Y.; Y. Bengio; G. Hinton: Deep learning. *Nature* 521 (2015) 7553, S. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539.

LEGG & M. HUTTER 2007

Legg, S.; M. Hutter: Universal Intelligence: A Definition of Machine Intelligence. *Minds and Machines* 17 (2007) 4, S. 391–444. ISSN: 0924-6495. DOI: 10.1007/s11023-007-9079-x.

LEIBER 2023

Leiber, D. F.: Automatisierte Layoutplanung von Montagelinien. de. Diss. Technische Universität München. 2023, S. 81.

LEONETTI et al. 2016

Leonetti, M.; L. Iocchi; P. Stone: A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artificial Intelligence* 241 (2016), S. 103–130. ISSN: 00043702. DOI: 10.1016/j.artint.2016.07.004.

LEVINE et al. 2016

Levine, S.; C. Finn; T. Darrell; P. Abbeel: End-to-End Training of Deep Visuomotor Policies. *J. Mach. Learn. Res.* 17 (2016) 1, S. 1334–1373. ISSN: 1532-4435.

B. H. LI et al. 2017

Li, B. H.; L. Zhang; T. Li; T. Y. Lin; J. Cui: Simulation-Based Cyber-Physical Systems and Internet-of-Things. In: *Guide to simulation-based disciplines*. Hrsg. von Mittal, S.; Durak, U.; Ören, T. Simulation foundations, methods and applications. Cham: Springer. 2017, S. 103–126. ISBN: 978-3-319-61263-8. DOI: 10.1007/978-3-319-61264-5\_5.

X. LI et al. 2017

Li, X.; C.-I. Vasile; C. Belta: Reinforcement learning with temporal logic rewards. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, S. 3834–3839. ISBN: 978-1-5386-2682-5. DOI: 10.1109/IROS.2017.8206234.

E. LIANG et al. 2017

Liang, E.; R. Liaw; R. Nishihara; P. Moritz; R. Fox; J. Gonzalez; K. Goldberg; I. Stoica: Ray RLlib: A Composable and Scalable Reinforcement Learning Library. *CoRR* abs/1712.09381 (2017).

X. LIANG et al. 2018

Liang, X.; T. Wang; L. Yang; E. Xing: CIRL: Controllable Imitative Reinforcement Learning for Vision-Based Self-driving. In: *Computer Vision - ECCV 2018*. Hrsg. von Ferrari, V. Bd. 11211. Lecture Notes in Computer Science Ser. Cham: Springer International Publishing AG. 2018, S. 604–620. ISBN: 978-3-030-01233-5. DOI: 10.1007/978-3-030-01234-2\_36.

LILLICRAP et al. 2016

Lillicrap, T. P.; J. J. Hunt; A. Pritzel; N. Heess; T. Erez; Y. Tassa; D. Silver; D. Wierstra: Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2016).

LINDEMANN 2005

Lindemann, U.: *Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden*. SpringerLink Bücher. Berlin, Heidelberg: Springer Berlin Heidelberg. 2005. ISBN: 9783540266174. DOI: 10.1007/b137764. URL: <http://swbplus.bsz-bw.de/bsz264345886cov.htm>.

LINDNER et al. 2021

Lindner, D.; R. Shah; P. Abbeel; A. Dragan: Learning What To Do by Simulating the

---

Past. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=kBVJ2NtiY->.

LOPEZ et al. 2019

Lopez, N. G.; Y. L. E. Nuin; E. B. Moral; L. U. S. Juan; A. S. Rueda; V. M. Vilches; R. Kojcev: gym-gazebo2, a toolkit for reinforcement learning using ROS 2 and Gazebo. URL: <https://arxiv.org/pdf/1903.06278>.

LORENZER 2011

Lorenzer, T.: Wandelbarkeit in der Serienfertigung durch rekonfigurierbare Werkzeugmaschinen. Diss. ETH Zurich. 2011. DOI: 10.3929/ETHZ-A-006381940.

LOTTER & WIENDAHL 2012

Lotter, B.; H.-P. Wiendahl: *Montage in der industriellen Produktion: Ein Handbuch für die Praxis*. 2. Aufl. 2013. VDI-Buch. Berlin & Heidelberg: Springer. 2012. ISBN: 9783642290619. DOI: 10.1007/978-3-642-29061-9. URL: <http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10645546>.

MAHLER & GOLDBERG 2017

Mahler, J.; K. Goldberg: Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences. In: *Proceedings of the 1st Annual Conference on Robot Learning*. Hrsg. von Sergey Levine; Vincent Vanhoucke; Ken Goldberg. Bd. 78. Proceedings of Machine Learning Research. PMLR. 2017, S. 515–524.

MAHLER et al. 2019

Mahler, J.; M. Matl; V. Satish; M. Danielczuk; B. DeRose; S. McKinley; K. Goldberg: Learning ambidextrous robot grasping policies. *Science Robotics* 4 (2019) 26, eaau4984. DOI: 10.1126/scirobotics.aau4984.

MAHMOOD et al. 2018

Mahmood, A.; D. Korenkevych; G. Vasan; W. Ma; J. Bergstra: Benchmarking Reinforcement Learning Algorithms on Real-World Robots. *ArXiv e-prints* (2018).

MALLOZZI et al. 2018

Mallozzi, P.; R. Pardo; V. Duplessis; P. Pelliccione; G. Schneider: MoVEMo: A Structured Approach for Engineering Reward Functions. In: *The Second IEEE International Conference on Robotic Computing*. Piscataway, NJ: IEEE. 2018, S. 250–257. ISBN: 978-1-5386-4652-6. DOI: 10.1109/IRC.2018.00053.

## Literatur

---

MARCUS & DAVIS 2019

Marcus, G. F.; E. Davis: *Rebooting AI: Building artificial intelligence we can trust*. 2019. ISBN: 9781524748258.

MARX et al. 2020

Marx, S.; A. Kanso; R. Müller: Unconventional path planning for a serial kinematics robot with Reinforcement Learning using the example of the wire loop game. In: Juli 2020. 2020.

MATAS et al. 2018

Matas, J.; S. James; A. J. Davison: Sim-to-Real Reinforcement Learning for Deformable Object Manipulation. URL: <http://arxiv.org/pdf/1806.07851v2>.

MCCARTHY 2007

McCarthy, J.: From here to human-level AI. *Artificial Intelligence* 171 (2007) 18, S. 1174–1182. ISSN: 00043702. DOI: 10.1016/j.artint.2007.10.009.

MCKINSEY et al. 2017

McKinsey; J. Manyika; M. Chui; M. Miremadi; J. Bughin; K. George; P. Willmott; M. Dewhurst: A Future that works: Automation, Employment, and Productivity. (2017).

MCKINSEY et al. 2019

McKinsey; A. Cam; M. Chui; B. Hall: Global AI Survey: AI proves its worth, but few scale impact. URL: <https://www.mckinsey.com/~media/McKinsey/Featured%20Insights/Artificial%20Intelligence/Global%20AI%20Survey%20AI%20proves%20its%20worth%20but%20few%20scale%20impact/Global-AI-Survey-AI-proves-its-worth-but-few-scale-impact.ashx> (besucht am 25. 11. 2019).

MEYES et al. 2017

Meyes, R.; H. Tercan; S. Roggendorf; T. Thiele; C. Büscher; M. Obdenbusch; C. Brecher; S. Jeschke; T. Meisen: Motion Planning for Industrial Robots using Reinforcement Learning. *Procedia CIRP* 63 (2017) Supplement C, S. 107–112. ISSN: 22128271. DOI: 10.1016/j.procir.2017.03.095. URL: <http://www.sciencedirect.com/science/article/pii/S221282711730241X>.



---

MICHNIEWICZ 2018

Michniewicz, J.: Automatische simulationsgestützte Arbeitsplanung in der Montage. Dissertation. München: Technische Universität München. 2018.

MIN et al. 2017

Min, A. T. W.; R. Sagarna; A. Gupta; Y.-S. Ong; C. K. Goh: Knowledge Transfer Through Machine Learning in Aircraft Design. *IEEE Computational Intelligence Magazine* 12 (2017) 4, S. 48–60. DOI: 10.1109/MCI.2017.2742781.

MITCHELL 2021

Mitchell, M.: Abstraction and Analogy-Making in Artificial Intelligence. URL: <https://arxiv.org/pdf/2102.10717>.

MNIH et al. 2016

Mnih, V.; A. P. Badia; M. Mirza; A. Graves; T. Lillicrap; T. Harley; D. Silver; K. Kavukcuoglu: Asynchronous Methods for Deep Reinforcement Learning. In: *Proceedings of The 33rd International Conference on Machine Learning*. Hrsg. von Balcan, M. F.; Weinberger, K. Q. Bd. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR. 2016, S. 1928–1937. URL: <https://proceedings.mlr.press/v48/mniha16.html>.

MÜLLER et al. 2021

Müller, M.; T. Müller; B. Ashtari Talkhestani; P. Marks; N. Jazdi; M. Weyrich: Industrial autonomous systems: a survey on definitions, characteristics and abilities. *at - Automatisierungstechnik* 69 (2021) 1, S. 3–13. ISSN: 0178-2312. DOI: 10.1515/auto-2020-0131.

MURATORE et al. 2018

Muratore, F.; F. Treede; M. Gienger; J. Peters: Domain Randomization for Simulation-Based Policy Optimization with Transferability Assessment. In: *Proceedings of The 2nd Conference on Robot Learning*. Hrsg. von Billard, A.; Dragan, A.; Peters, J.; Morimoto, J. Bd. 87. Proceedings of Machine Learning Research. PMLR. 2018, S. 700–713. URL: <https://proceedings.mlr.press/v87/muratore18a.html>.

NAGABANDI et al. 2019

Nagabandi, A.; K. Konoglie; S. Levine; V. Kumar: Deep Dynamics Models for Learning Dexterous Manipulation. URL: <http://arxiv.org/pdf/1909.11652v1>.

## Literatur

---

NÄGELE 2021

Nägele, F.: Prototypbasiertes Skill-Modell zur Programmierung von Robotern für kraft-geregelte Montageprozesse. DOI: 10.18419/OPUS-11655.

NAIR et al. 2018

Nair, A.; V. Pong; M. Dalal; S. Bahl; S. Lin; S. Levine: Visual Reinforcement Learning with Imagined Goals. *ArXiv e-prints* (2018).

NEE 2015

Nee, A. Y. C., Hrsg. (2015): *Handbook of manufacturing engineering and technology*. London: Springer Reference. 2015. ISBN: 9781447146711.

NG 2018

Ng, A.: Machine Learning Yearning: Technical Strategy for AI Engineers, in the Era of Deep Learning. URL: [https://gallery.mailchimp.com/dc3a7ef4d750c0abfc19202a3/files/5dd91615-3b3f-4f5d-bbfb-4ebd8608d330/Ng\\_MLY01\\_13.pdf](https://gallery.mailchimp.com/dc3a7ef4d750c0abfc19202a3/files/5dd91615-3b3f-4f5d-bbfb-4ebd8608d330/Ng_MLY01_13.pdf) (besucht am 04. 10. 2018).

NG 2021

Ng, A.: Die nächste Phase der Künstlichen Intelligenz beginnt gerade. *FAZ* (2021). URL: <https://www.faz.net/-ikh-a8cmi> (besucht am 11. 02. 2021).

NOGUEIRA 2014

Nogueira, F.: Bayesian Optimization: Open source constrained global optimization tool for Python. URL: <https://github.com/fmfn/BayesianOptimization> (besucht am 04. 02. 2021).

NORONHA et al. 2018

Noronha, D.; K. Gibson; B. Salehpour; S. J. Wilton: LeFlow: Automatic Compilation of TensorFlow Machine Learning Applications to FPGAs. In: *FPT 2018*. Los Alamitos, CA: IEEE Computer Society. 2018, S. 393–396. ISBN: 978-1-7281-0214-6. DOI: 10.1109/FPT.2018.00082.

NOTTENSTEINER et al. 2016

Nottensteiner, K.; T. Bodenmueller; M. Kassecker; M. A. Roa; A. Stemmer; T. Stouraitis; D. Seidel; U. Thomas: A Complete Automated Chain for Flexible Assembly using Recognition, Planning and Sensor-Based Execution. In: *Proceedings of ISR 2016: 47st International Symposium on Robotics*. 2016, S. 1–8.

NOTTENSTEINER et al. 2021

Nottensteiner, K.; A. Sachtler; A. Albu-Schäffer: Towards Autonomous Robotic Assembly: Using Combined Visual and Tactile Sensing for Adaptive Task Execution. *Journal of Intelligent & Robotic Systems* 101 (2021) 3. ISSN: 0921-0296. DOI: 10.1007/s10846-020-01303-z.

NUGROHO et al. 2023

Nugroho, K. A.; M. Amirul Haq; C.-K. Wang; S.-J. Ruan; M. Polikarpov; D. Wagstyl; J. Deuse: Towards Smart Manufacturing using Reinforcement Learning in a Sparse-Rewarded Environment for Through-Hole Technology. In: *2023 IEEE 12th Global Conference on Consumer Electronics (GCCE)*. 2023, S. 672–673. DOI: 10.1109/GCCE59613.2023.10315655.

OBANDO-CERÓN & CASTRO 2021

Obando-Cerón, J. S.; P. S. Castro: Revisiting Rainbow: More inclusive deep reinforcement learning research. In: *Proceedings of the 38th International Conference on Machine Learning*. 2021.

OPENAI 2023

OpenAI: GPT-4 Technical Report. arXiv: 2303.08774 [cs.CL].

OPENAI et al. 2018

OpenAI; M. Andrychowicz; B. Baker; M. Chociej; R. Jozefowicz; B. McGrew; J. Pachocki; A. Petron; M. Plappert; G. Powell; A. Ray; J. Schneider; S. Sidor; J. Tobin; P. Welinder; L. Weng; W. Zaremba: Learning Dexterous In-Hand Manipulation. *ArXiv e-prints* (2018).

OPENAI et al. 2019

OpenAI; Ilge Akkaya; Marcin Andrychowicz; Maciek Chociej; Mateusz Litwin; Bob McGrew; Arthur Petron; Alex Paino; Matthias Plappert; Glenn Powell; Raphael Ribas; Jonas Schneider; Nikolas Tezak; Jerry Tworek; Peter Welinder; Lilian Weng; Qiming Yuan; Wojciech Zaremba; Lei Zhang: Solving Rubik’s Cube with a Robot Hand. *arXiv preprint* (2019).

ORTIZ 2019

Ortiz, D.: Automatische Generierung von Lernumgebungen für autonome Montagesysteme. Masterarbeit. Technische Universität München. 2019.

## Literatur

---

OSBAND et al. 2019

Osband, I.; Y. Doron; M. Hessel; J. Aslanides; E. Sezener; A. Saraiva; K. McKinney; T. Lattimore; C. Szepesvari; S. Singh; B. van Roy; R. Sutton; D. Silver; H. van Hasselt: Behaviour Suite for Reinforcement Learning. URL: <http://arxiv.org/pdf/1908.03568v2>.

OTTE 2009

Otte, M. W.: A Survey of Machine Learning Approaches to Robotic Path-Planning. In: 2009.

PABICH et al. 2019

Pabich, E.; D. Janssen; F. Hees: InPulS. Intelligente und selbstlernende Produktionsprozesse. Abschlussbericht. URL: [https://industrie40.vdma.org/documents/4214230/48381082/FKM\\_InPuls\\_Abschlussbericht\\_Januar2020\\_1587989951271.pdf/bcff3666-aa49-43b6-b92d-fac71b20110f](https://industrie40.vdma.org/documents/4214230/48381082/FKM_InPuls_Abschlussbericht_Januar2020_1587989951271.pdf/bcff3666-aa49-43b6-b92d-fac71b20110f) (besucht am 21. 07. 2021).

PAPADOPOULOS et al. 2015

Papadopoulos, G.; S. Rikama; P. Alajääskö; Z. Salah-Eddine; A. Airaksinen; H. Luomaranta: Statistics on small and medium-sized enterprises - Statistics Explained. URL: [https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Statistics\\_on\\_small\\_and\\_medium-sized\\_enterprises](https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Statistics_on_small_and_medium-sized_enterprises) (besucht am 01. 04. 2021).

PEARL & MACKENZIE 2018

Pearl, J.; D. Mackenzie: *The book of why: The new science of cause and effect*. First edition. New York: Basic Books. 2018. ISBN: 9780465097616.

PENG et al. 2017

Peng, X. B.; M. Andrychowicz; W. Zaremba; P. Abbeel: Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *CoRR* abs/1710.06537 (2017).

PERZYLO et al. 2019

Perzylo, A.; M. Rickert; B. Kahl; N. Somani; C. Lehmann; A. Kuss; S. Profanter; A. B. Beck; M. Haage; M. Rath Hansen; M. T. Nibe; M. A. Roa; O. Sornmo; S. Gestegard Robertz; U. Thomas; G. Veiga; E. A. Topp; I. Kessler; M. Danzer: SMERobotics: Smart

Robots for Flexible Manufacturing. *IEEE Robotics & Automation Magazine* 26 (2019) 1, S. 78–90. ISSN: 1070-9932. DOI: 10.1109/MRA.2018.2879747.

PETERS et al. 2017

Peters, J.; D. Janzing; B. Schölkopf: *Elements of causal inference: Foundations and learning algorithms*. Adaptive computation and machine learning. Cambridge, Massachusetts & London, England: MIT Press. 2017. ISBN: 9780262037310.

PHAM et al. 2018

Pham, T.-H.; G. de Magistris; R. Tachibana: OptLayer - Practical Constrained Optimization for Deep Reinforcement Learning in the Real World. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Hrsg. von Lynch, K. Piscataway, NJ: IEEE. 2018, S. 6236–6243. ISBN: 978-1-5386-3081-5. DOI: 10.1109/ICRA.2018.8460547.

PHANITEJA et al. 2018a

Phaniteja, S.; P. Dewangan; P. Guhan; A. Sarkar; K. M. Krishna: A Deep Reinforcement Learning Approach for Dynamically Stable Inverse Kinematics of Humanoid Robots. *IEEE International Conference on Robotics and Biomimetics* (2018). URL: <https://arxiv.org/pdf/1801.10425>.

PHANITEJA et al. 2018b

Phaniteja, S.; P. Dewangan; P. Guhan; K. M. Krishna; A. Sarkar: Learning Coordinated Tasks using Reinforcement Learning in Humanoids. URL: <https://arxiv.org/pdf/1805.03584>.

PIKETTY 2018

Piketty, T.: *Das Kapital im 21. Jahrhundert*. 2. Auflage in C.H. Beck Paperback. Bd. 6236. C.H. Beck Paperback. München: C.H. Beck. 2018. ISBN: 9783406688652.

PINTO et al. 2017

Pinto, L.; M. Andrychowicz; P. Welinder; W. Zaremba; P. Abbeel: Asymmetric Actor Critic for Image-Based Robot Learning. *CoRR* abs/1710.06542 (2017).

PLATTFORM INDUSTRIE 4.0 2020

Plattform Industrie 4.0: Details of the Asset Administration Shell. Hrsg. von BMWi. (Besucht am 18.03.2021).

## Literatur

---

POLASEK 1994

Polasek, W.: *EDA Explorative Datenanalyse: Einführung in die deskriptive Statistik*. Zweite, neubearbeitete und erweiterte Auflage. Springer-Lehrbuch. Berlin & Heidelberg: Springer. 1994. ISBN: 9783540583943. DOI: 10.1007/978-3-642-57889-2.

POPOV et al. 2017

Popov, I.; N. Heess; T. P. Lillicrap; R. Hafner; G. Barth-Maron; M. Vecerik; T. Lampe; Y. Tassa; T. Erez; M. A. Riedmiller: Data-efficient Deep Reinforcement Learning for Dexterous Manipulation. *CoRR* abs/1704.03073 (2017).

PORETSCHKIN 2019

Poretschkin, M.: Vertrauenswürdiger Einsatz von Künstlicher Intelligenz: Handlungsfelder aus philosophischer, ethischer, rechtlicher und technologischer Sicht als Grundlage für eine Zertifizierung von Künstlicher Intelligenz. Sankt Augustin. URL: [https://www.iais.fraunhofer.de/content/dam/iais/KINRW/Whitepaper\\_KI-Zertifizierung.pdf](https://www.iais.fraunhofer.de/content/dam/iais/KINRW/Whitepaper_KI-Zertifizierung.pdf) (besucht am 29. 11. 2019).

PORTELAS et al. 2020

Portelas, R.; C. Colas; L. Weng; K. Hofmann; P.-Y. Oudeyer: Automatic Curriculum Learning For Deep RL: A Short Survey. URL: <https://arxiv.org/pdf/2003.04664>.

POTT & DIETZ 2019

Pott, A.; T. Dietz: *Industrielle Robotersysteme: Entscheiderwissen für die Planung und Umsetzung wirtschaftlicher Roboterlösungen*. Wiesbaden Germany: Springer Vieweg. 2019. ISBN: 9783658253448.

QIAO et al. 2008

Qiao, J.; Z. Hou; X. Ruan: Application of reinforcement learning based on neural network to dynamic obstacle avoidance. In: *International Conference on Information and Automation, 2008*. Piscataway, NJ: IEEE. 2008, S. 784–788. ISBN: 978-1-4244-2183-1. DOI: 10.1109/ICINFA.2008.4608104.

RADOSAVOVIC et al. 2020

Radosavovic, I.; R. P. Kosaraju; R. Girshick; K. He; P. Dollár: Designing Network Design Spaces. URL: <https://arxiv.org/pdf/2003.13678>.

RANASINGHE et al. 2019

Ranasinghe, T.; C. Orasan; R. Mitkov: Semantic Textual Similarity with Siamese Neural Networks. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria: INCOMA Ltd. 2019, S. 1004–1011. DOI: 10.26615/978-954-452-056-4\_116. URL: <https://www.aclweb.org/anthology/R19-1116>.

RASCHKA & MIRJALILI 2018

Raschka, S.; V. Mirjalili: *Machine Learning mit Python und Scikit-Learn und TensorFlow: Das umfassende Praxis-Handbuch für Data Science, Deep Learning und Predictive Analytics*. 2., aktualisierte und erweiterte Auflage. Frechen: MITP. 2018. ISBN: 9783958457331.

REINHART 2017

Reinhart, G.: *Handbuch Industrie 4.0: Geschäftsmodelle, Prozesse, Technik*. 2017. ISBN: 3446446427.

CO-REYES et al. 2021

Co-Reyes, J. D.; Y. Miao; D. Peng; E. Real; Q. Le; S. Levine; H. Lee; A. Faust: Evolving Reinforcement Learning Algorithms. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=0XXpJ4OtjW>.

RICHTER & WEBER 2013

Richter, M. M.; R. O. Weber: *Case-Based Reasoning: A Textbook*. Berlin/Heidelberg: Springer Berlin Heidelberg. 2013. ISBN: 9783642401664. URL: <https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=3095804>.

RÖHLER 2020

Röhler, M.: Artificial Intelligence in southern Germany: Opportunities for Dutch Businesses in the Industrial Sector. Hrsg. von Konsulat des Königreichs der Niederlande. URL: [https://www.rvo.nl/sites/default/files/2020/12/20201202\\_Market\\_Study\\_AI.pdf](https://www.rvo.nl/sites/default/files/2020/12/20201202_Market_Study_AI.pdf) (besucht am 06. 12. 2020).

RÖHLER 2021

Röhler, M.: Industrial-RL: Github Repository. URL: <https://github.com/maroehler/industrial-rl/> (besucht am 03. 08. 2021).

RÖHLER et al. 2018

Röhler, M.; V. Kumar; C. Richter; G. Reinhart: A SPH Simulation Approach using the Carreau Model for the Free Surface Flow of Adhesives. In: *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE, 2018, S. 128–132. ISBN: 978-1-5386-6786-6. DOI: 10.1109/IEEM.2018.8607468.

RÖHLER et al. 2019

Röhler, M.; J. Berg; G. Reinhart: Aufgabenorientierte Konfiguration autonomer Robotersysteme zur industriellen Anwendung von Reinforcement Learning. In: *VDI Kongress AUTOMATION 2019*. Baden Baden, 2019, S. 753–566.

RÖHLER et al. 2020

Röhler, M.; J. Berg; G. Reinhart: Automatische Konfiguration autonomer Robotersysteme. In: *at Automatisierungstechnik 05 2020*. Vulkan Verlag, 2020.

RÖHLER et al. 2021

Röhler, M.; L. Merhar; K. Sivalingam; P. Rübél; L. Theis: Reinforcement Learning - warten oder starten? URL: [https://kompetenzzentrum-augsburg-digital.de/wp-content/uploads/2021/09/Leitfaden\\_ReinforcementLearning.pdf](https://kompetenzzentrum-augsburg-digital.de/wp-content/uploads/2021/09/Leitfaden_ReinforcementLearning.pdf) (besucht am 16. 09. 2021).

RÖHLER & SCHILP 2022

Röhler, M.; J. Schilp: Knowledge-based Implementation of Deep Reinforcement Learning Agents in Assembly. *Procedia CIRP* 112 (2022). 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 14-16 July 2021, S. 459–464. ISSN: 2212-8271. DOI: <https://doi.org/10.1016/j.procir.2022.09.088>. URL: <https://www.sciencedirect.com/science/article/pii/S2212827122012513>.

ROHNER et al. 2020

Rohner, D.; M. Sand; D. Henrich: User Guidance and Automatic Completion for Generating Planar B-Rep Models. In: *Annals of Scientific Society for Assembly, Handling and Industrial Robotics*. Hrsg. von Schüppstuhl, T.; Tracht, K.; Henrich, D. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, S. 33–43. ISBN: 978-3-662-61755-7.



RÖSCH 2021

Rösch, M. W.: *System zur energieorientierten und kostenbasierten Produktionssteuerung mittels Reinforcement Learning*. de.

ROSS et al. 2008

Ross, A. M.; D. H. Rhodes; D. E. Hastings: Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. *Systems Engineering* 11 (2008) 3, S. 246–262. ISSN: 1098-1241. DOI: 10.1002/sys.20098.

RUSSELL 2016

Russell, S.: Rationality and Intelligence: A Brief Update. In: *Fundamental issues of artificial intelligence*. Hrsg. von Müller, V. C. Bd. 48. Synthese Library, Studies in Epistemology, Logic, Methodology, and Philosophy of Science. Cham: Springer. 2016, S. 7–28. ISBN: 978-3-319-26483-7. DOI: 10.1007/978-3-319-26485-1\_2.

RUSSELL & NORVIG 2010

Russell, S.; P. Norvig: *Artificial intelligence: A modern approach*. 3. ed. Prentice-Hall series in artificial intelligence. Upper Saddle River, NJ: Prentice-Hall. 2010. ISBN: 9780136042594.

RYAN et al. 2013

Ryan, E. T.; D. R. Jacques; J. M. Colombi: An ontological framework for clarifying flexibility-related terminology via literature survey. *Systems Engineering* 16 (2013) 1, S. 99–110. ISSN: 1098-1241. DOI: 10.1002/sys.21222.

SADEGHI et al. 2017

Sadeghi, F.; A. Toshev; E. Jang; S. Levine: Sim2Real View Invariant Visual Servoing by Recurrent Control. *CoRR* abs/1712.07642 (2017).

SATARIANO & METZ 2020

Satariano, A.; C. Metz: A Warehouse Robot Learns to Sort Out the Tricky Stuff. *The New York Times* (2020). ISSN: 1553-8095. URL: <https://www.nytimes.com/2020/01/29/technology/warehouse-robot.html> (besucht am 09.09.2021).

SCHMIDT 1992

Schmidt, M.: *Konzeption und Einsatzplanung flexibel automatisierter Montagesysteme*. Bd. 41. iwv Forschungsberichte, Berichte aus dem Institut für Werkzeugmaschinen und

## Literatur

---

Betriebswissenschaften der Technischen Universität München. Berlin & Heidelberg: Springer. 1992. ISBN: 9783642772177. DOI: 10.1007/978-3-642-77217-7.

SCHOETTLER et al. 2019

Schoettler, G.; A. Nair; J. Luo; S. Bahl; J. A. Ojea; E. Solowjow; S. Levine: Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards. URL: <http://arxiv.org/pdf/1906.05841v1>.

SCHÖLKOPF et al. 2021

Schölkopf, B.; F. Locatello; S. Bauer; N. R. Ke; N. Kalchbrenner; A. Goyal; Y. Bengio: Toward Causal Representation Learning. *Proceedings of the IEEE* 109 (2021) 5, S. 612–634. ISSN: 1558-2256. DOI: 10.1109/JPROC.2021.3058954.

SCHOLZ-REITER & FREITAG 2007

Scholz-Reiter, B.; M. Freitag: Autonomous Processes in Assembly Systems. *CIRP Annals* 56 (2007) 2, S. 712–729. ISSN: 00078506. DOI: 10.1016/j.cirp.2007.10.002.

SCHULMAN et al. 2016

Schulman, J.; P. Moritz; S. Levine; M. I. Jordan; P. Abbeel: High-Dimensional Continuous Control Using Generalized Advantage Estimation. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Hrsg. von Yoshua Bengio; Yann LeCun. 2016.

SCHULMAN et al. 2017

Schulman, J.; F. Wolski; P. Dhariwal; A. Radford; O. Klimov: Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017).

SHACKLETT et al. 2021

Shacklett, B.; E. Wijmans; A. Petrenko; M. Savva; D. Batra; V. Koltun; K. Fatahalian: Large Batch Simulation for Deep Reinforcement Learning. URL: <https://arxiv.org/pdf/2103.07013>.

SHAH et al. 2008

Shah, N.; J. Wilds; L. Viscito; D. Hastings: Quantifying Flexibility for Architecting Changeable Systems. (Besucht am 19.06.2018).

SHALLUE et al. 2019

Shallue, C. J.; J. Lee; J. Antognini; J. Sohl-Dickstein; R. Frostig; G. E. Dahl: Measuring the Effects of Data Parallelism on Neural Network Training. *Journal of Machine Learning Research* 20 (2019) 112, S. 1–49. URL: <http://jmlr.org/papers/v20/18-789.html>.

EL-SHAMOUTY et al. 2019

El-Shamouty, M.; K. Kleeberger; A. Lämmle; M. Huber: Simulation-driven machine learning for robotics and automation. *tm - Technisches Messen* 0 (2019) 0, S. 529. ISSN: 0171-8096. DOI: 10.1515/teme-2019-0072.

SHAO et al. 2018

Shao, S.; J. Tsai; M. Mysior; W. Luk; T. Chau; A. Warren; B. Jeppesen: Towards Hardware Accelerated Reinforcement Learning for Application-Specific Robotic Control. In: *2018 IEEE 29th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*. Piscataway, NJ: IEEE. 2018, S. 1–8. ISBN: 978-1-5386-7479-6. DOI: 10.1109/ASAP.2018.8445099.

SICILIANO & KHATIB 2016

Siciliano, B.; O. Khatib: *Springer Handbook of Robotics*. Cham: Springer International Publishing. 2016. ISBN: 9783319325521. URL: <https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4901894>.

SILVER et al. 2021

Silver, D.; S. Singh; D. Precup; R. S. Sutton: Reward Is Enough. *Artificial Intelligence* (2021), S. 103535. ISSN: 00043702. DOI: 10.1016/j.artint.2021.103535. URL: <https://www.sciencedirect.com/science/article/pii/S0004370221000862>.

SINGH et al. 2019

Singh, A.; L. Yang; K. Hartikainen; C. Finn; S. Levine: End-to-End Robotic Reinforcement Learning without Reward Engineering. *CoRR* abs/1904.07854 (2019).

SINHA et al. 2021

Sinha, S.; H. Bharadhwaj; A. Srinivas; A. Garg: D25RL6: Deep Dense Architectures in Reinforcement Learning. In: *International Conference on Learning Representations*. 2021.

SMALE 2011

Smale, D. M.: Towards an integrated framework for the configuration of modular micro assembly systems. Dissertation. University of Nottingham. 2011.

SNOEK et al. 2012

Snoek, J.; H. Larochelle; R. P. Adams: Practical Bayesian Optimization of Machine Learning Algorithms. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2. NIPS' 12*. Red Hook, NY, USA: Curran Associates Inc. 2012, S. 2951–2959.

STEHLE & HEISEL 2017

Stehle, T.; U. Heisel: Konfiguration und Rekonfiguration von Produktionssystemen. In: *Neue Entwicklungen in der Unternehmensorganisation*. Hrsg. von Spath, D.; Westkämper, E.; Bullinger, H.-J.; Warnecke, H.-J. Berlin, Heidelberg: Springer Berlin Heidelberg. 2017, S. 333–367. ISBN: 978-3-662-55426-5. DOI: 10.1007/978-3-662-55426-5\_39. URL: [https://doi.org/10.1007/978-3-662-55426-5\\_39](https://doi.org/10.1007/978-3-662-55426-5_39).

STEPHAN & WALTER 2013

Stephan, A.; S. Walter, Hrsg. (2013): *Handbuch Kognitionswissenschaft*. Stuttgart & Weimar: Verlag J.B. Metzler. 2013. ISBN: 9783476023315. DOI: 10.1007/978-3-476-05288-9.

STICH 2017

Stich, P. M. A. G.: *Interaktiver simulationsgestützter Entwurf mechatronischer Verarbeitungssysteme*. München. 2017.

SURMANN et al. 2020

Surmann, H.; C. Jestel; R. Marchel; F. Musberg; H. Elhadji; M. Ardani: Deep Reinforcement learning for real autonomous mobile robot navigation in indoor environments. URL: <https://arxiv.org/pdf/2005.13857>.

SUTTON & BARTO 2018

Sutton, R. S.; A. G. Barto: *Reinforcement Learning: An Introduction*. Second edition. The MIT Press. 2018.

TAI et al. 2017

Tai, L.; G. Paolo; M. Liu: Virtual-to-real Deep Reinforcement Learning: Continuous

---

Control of Mobile Robots for Mapless Navigation. URL: <https://arxiv.org/pdf/1703.00420>.

TAN et al. 2018

Tan, J.; T. Zhang; E. Coumans; A. Iscen; Y. Bai; D. Hafner; S. Bohez; V. Vanhoucke: Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *CoRR* abs/1804.10332 (2018).

TASSA et al. 2018

Tassa, Y.; Y. Doron; A. Muldal; T. Erez; Y. Li; D. de Las Casas; D. Budden; A. Abdolmaleki; J. Merel; A. Lefrancq; T. Lillicrap; M. Riedmiller: DeepMind Control Suite. *ArXiv e-prints* (2018).

THOMAS et al. 2018

Thomas, G.; M. Chien; A. Tamar; J. A. Ojea; P. Abbeel: Learning Robotic Assembly from CAD. URL: <https://arxiv.org/pdf/1803.07635>.

TILLEY 2017

Tilley, J.: Automation, robotics, and the factory of the future. *McKinsey* (2017).

TRAPANI 2019

Trapani, S.: Task Oriented Programming and Service Algorithms for Smart Robotic Cells. Dissertation. Politecnico di Torino. 2019.

TURNER et al. 2021

Turner, R.; D. Eriksson; M. McCourt; J. Kiili; E. Laaksonen; Z. Xu; I. Guyon: Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. URL: <https://arxiv.org/pdf/2104.10201>.

ULRICH & HILL 1976

Ulrich, P.; W. Hill: Wissenschaftstheoretische Grundlagen der Betriebswirtschaftslehre. *Wirtschaftswissenschaftliches Studium : Zeitschrift für Ausbildung und Hochschulkontakt* 5 (1976) 7+8. URL: <https://www.alexandria.unisg.ch/17331/>.

UNBEHAUEN & LEY 2014

Unbehauen, H.; F. Ley: *Das Ingenieurwissen: Regelungs- und Steuerungstechnik*. Berlin: Springer Vieweg. 2014. ISBN: 3662440253.

## Literatur

---

VDI 2860

VDI 2860: Handhabungsfunktionen, Handhabungseinrichtungen: Begriffe, Definitionen, Symbole.

VDI-Richtlinie 2653

VDI-Richtlinie 2653: Agentensysteme in der Automatisierungstechnik.

VDI/VDE 3550

VDI/VDE 3550: Computational Intelligence - Künstliche Neuronale Netze in der Automatisierungstechnik: Begriffe und Definitionen.

VDMA 2018

VDMA: Künstliche Intelligenz im Maschinenbau - Perspektiven und Handlungsempfehlungen: VDMA-Stellungnahme zu einer politischen Strategie für Künstliche Intelligenz in Deutschland und Europa. (2018).

VECERIK et al. 2017

Vecerik, M.; T. Hester; J. Scholz; F. Wang; O. Pietquin; B. Piot; N. Heess; T. Rothörl; T. Lampe; M. Riedmiller: Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. URL: <http://arxiv.org/pdf/1707.08817v2>.

VOGEL-HEUSER et al. 2014

Vogel-Heuser, B.; U. Lindemann; G. Reinhart: *Innovationsprozesse zyklenorientiert managen: Verzahnte Entwicklung von Produkt-Service Systemen*. Berlin: Springer Vieweg. 2014. ISBN: 978-3-662-44931-8. DOI: 10.1007/978-3-662-44932-5. URL: <http://dx.doi.org/10.1007/978-3-662-44932-5>.

WANG et al. 2018

Wang, J.; Y. Ma; L. Zhang; R. X. Gao; D. Wu: Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems* (2018). ISSN: 02786125. DOI: 10.1016/j.jmsy.2018.01.003.

WARNECKE 1995

Warnecke, H.-J.: *Die Montage im flexiblen Produktionsbetrieb: Technik, Organisation, Betriebswirtschaft*. Berlin & Heidelberg: Springer. 1995. ISBN: 9783642799631. DOI: 10.1007/978-3-642-79963-1.

WECK & BRECHER 2006

Weck, M.; C. Brecher: *Werkzeugmaschinen 4: Automatisierung von Maschinen und Anlagen*. 6., neu bearbeitete Auflage. VDI-Buch. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg. 2006. ISBN: 9783540453666. DOI: 10.1007/978-3-540-45366-6.

WESTKÄMPER et al. 2000

Westkämper, E.; E. Zahn; P. Balve; M. Tilebein: Ansätze zur Wandlungsfähigkeit von Produktionsunternehmen. Ein Bezugsrahmen für die Unternehmensentwicklung im turbulenten Umfeld. *wt Werkstattstechnik online* 90 (2000), S. 22–26.

WIENDAHL et al. 2007

Wiendahl, H.-P.; H. A. ElMaraghy; P. Nyhuis; M. F. Zäh; H.-H. Wiendahl; N. Duffie; M. Brieke: Changeable Manufacturing - Classification, Design and Operation. *CIRP Annals* 56 (2007) 2, S. 783–809. ISSN: 00078506. DOI: 10.1016/j.cirp.2007.10.003.

WIENDAHL et al. 2014

Wiendahl, H.-P.; J. Reichardt; P. Nyhuis: *Handbuch Fabrikplanung: Konzept, Gestaltung und Umsetzung wandlungsfähiger Produktionsstätten*. 2., überarb. und erw. Aufl., [elektronische Ressource]. München: Hanser. 2014. ISBN: 3446437029. DOI: 10.3139/9783446437029. URL: <http://www.hanser-elibrary.com/action/showBook?doi=10.3139/9783446437029>.

WIESE et al. 2022

Wiese, T.; J. Abicht; C. Friedrich; A. Hellmich; S. Ihlenfeldt: Flexible skill-based control for robot cells in manufacturing. *Frontiers in Robotics and AI* 9 (2022). ISSN: 2296-9144. DOI: 10.3389/frobt.2022.1014476. URL: <https://www.frontiersin.org/articles/10.3389/frobt.2022.1014476>.

WIEWIORA 2017

Wiewiora, E.: Reward Shaping. In: *Encyclopedia of Machine Learning and Data Mining*. Hrsg. von Sammut, C.; Webb, G. I. Boston, MA: Springer US. 2017, S. 1104–1106. ISBN: 978-1-4899-7687-1. DOI: 10.1007/978-1-4899-7687-1\_966. URL: [https://doi.org/10.1007/978-1-4899-7687-1\\_966](https://doi.org/10.1007/978-1-4899-7687-1_966).

WIGGERS 2021

Wiggers, K.: OpenAI disbands its robotics research team. *VentureBeat* (2021). URL:

## Literatur

---

<https://venturebeat.com/2021/07/16/openai-disbands-its-robotics-research-team/> (besucht am 02.08.2021).

WILLIAMS 1992

Williams, R. J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8 (1992) 3, S. 229–256. ISSN: 1573-0565. DOI: 10.1007/BF00992696. URL: <https://doi.org/10.1007/BF00992696>.

WINDER 2020

Winder, P.: *Reinforcement Learning. Industrial Applications with Intelligent Agents*. [S.l.]: O'Reilly Media, Inc. 2020. ISBN: 9781098114831.

WU et al. 2017

Wu, J.; S. Shin; C.-G. Kim; S.-D. Kim: Effective lazy training method for deep q-network in obstacle avoidance and path planning. In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Piscataway, NJ: IEEE. 2017, S. 1799–1804. ISBN: 978-1-5386-1645-1. DOI: 10.1109/SMC.2017.8122877.

XIE et al. 2018

Xie, L.; S. Wang; S. Rosa; A. Markham; N. Trigoni: Learning with Training Wheels: Speeding up Training with a Simple Controller for Deep Reinforcement Learning. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Hrsg. von Lynch, K. Piscataway, NJ: IEEE. 2018, S. 6276–6283. ISBN: 978-1-5386-3081-5. DOI: 10.1109/ICRA.2018.8461203.

YANG et al. 2023

Yang, S.; O. Nachum; Y. Du; J. Wei; P. Abbeel; D. Schuurmans: Foundation Models for Decision Making: Problems, Methods, and Opportunities. arXiv: 2303.04129 [cs.AI].

YE et al. 2018

Ye, X.; Z. Lin; H. Li; S. Zheng; Y. Yang: Active Object Perceiver: Recognition-Guided Policy Learning for Object Searching on Mobile Robots. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, S. 6857–6863. ISBN: 978-1-5386-8094-0. DOI: 10.1109/IROS.2018.8593720.

YOUNG et al. 2020

Young, M. T.; J. D. Hinkle; R. Kannan; A. Ramanathan: Distributed Bayesian optimi-



zation of deep reinforcement learning algorithms: *Journal of Parallel and Distributed Computing*, 139, 43-52. (2020). DOI: 10.1016/J.JPDC.2019.07.008.

YU et al. 2017

Yu, W.; J. Tan; C. K. Liu; G. Turk: Preparing for the Unknown: Learning a Universal Policy with Online System Identification. In: *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*. Hrsg. von Nancy M. Amato; Siddhartha S. Srinivasa; Nora Ayanian; Scott Kuindersma. 2017. DOI: 10.15607/RSS.2017.XIII.048. URL: <http://www.roboticsproceedings.org/rss13/p48.html>.

ZADOR 2019

Zador, A. M.: A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications* 10 (2019) 1, S. 1–7. ISSN: 2041-1723. DOI: 10.1038/s41467-019-11786-6. URL: <https://doi.org/10.1038/s41467-019-11786-6>.

ZÄH et al. 2005

Zäh, M.; N. Möller; W. Vogl: Symbiosis of Changeable and Virtual Production: The Emperor's New Clothes or Key Factor for Future Success? In: *1st International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2005)*. Hrsg. von Reinhart, G.; Zaeh, M. F. Utz. 2005. ISBN: 978-3-8316-0541-5. URL: <https://www.utzverlag.de/assets/pdf/40541ein.pdf> (besucht am 16.04.2019).

ZENG et al. 2018

Zeng, A.; S. Song; S. Welker; J. Lee; A. Rodriguez; T. Funkhouser: Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning. URL: <http://arxiv.org/pdf/1803.09956v3>.

ZHAO et al. 2020

Zhao, W.; J. P. Queralta; T. Westerlund: Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020, S. 737–744. DOI: 10.1109/SSCI47803.2020.9308468.

ZHOU et al. 2014

Zhou, D.; S. Chen; L. Li; H. Li; Y. Zhao: Accuracy Improvement of Smoothed Particle Hydrodynamics. *Engineering Applications of Computational Fluid Mechanics* 2 (2014) 2, S. 244–251. ISSN: 1994-2060. DOI: 10.1080/19942060.2008.11015225.

H. ZHU et al. 2019a

Zhu, H.; A. Gupta; A. Rajeswaran; S. Levine; V. Kumar: Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost. In: *2019 International Conference on Robotics and Automation (ICRA)*. [Piscataway, NJ]: IEEE. 2019, S. 3651–3657. ISBN: 978-1-5386-6027-0. DOI: 10.1109/ICRA.2019.8794102.

H. ZHU et al. 2019b

Zhu, H.; J. Yu; A. Gupta; D. Shah; K. Hartikainen; A. Singh; V. Kumar; S. Levine: The Ingredients of Real World Robotic Reinforcement Learning. In: 2019. (Besucht am 26.04.2020).

Y. ZHU et al. 2017

Zhu, Y.; R. Mottaghi; E. Kolve; J. J. Lim; A. Gupta; L. Fei-Fei; A. Farhadi: Target-driven visual navigation in indoor scenes using deep reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), S. 3357–3364.

Z. ZHU et al. 2020

Zhu, Z.; K. Lin; J. Zhou: Transfer Learning in Deep Reinforcement Learning: A Survey. URL: <https://arxiv.org/pdf/2009.07888>.

Z. ZHU et al. 2023

Zhu, Z.; K. Lin; A. K. Jain; J. Zhou: Transfer Learning in Deep Reinforcement Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2023) 11, S. 13344–13362. DOI: 10.1109/TPAMI.2023.3292075.

## **A Anhang**

### **A.1 Interview-Fragebogen**

**Frage 1: In wieweit beschäftigt sich Ihr Unternehmen bereits mit KI? Bitte ordnen Sie sich ein:**

- Nicht: Bisher wurde sich mit dem Thema KI noch nicht auseinandergesetzt
- Informativ: Es gibt ein Verständnis über die Potentiale der KI
- Anwendungsfälle: Es wurden erste Anwendungsfälle identifiziert
- Planung: Die Umsetzung wird aktuell vorbereitet
- Umsetzung: Ein oder mehrere Anwendungsfälle wurden mittels KI umgesetzt

**Frage 2: Welchen Nutzen versprechen Sie sich von KI?**

- Neue Produkte oder Dienstleistungen
- Verbesserung der Qualität
- Automation wiederkehrender Tätigkeiten
- Tieferes Verständnis interner Prozesse
- Erhöhung der Flexibilität
- Imagegewinn für die Firma
- Weiterer Nutzen (Bitte erläutern)

**Frage 3: Welche KI-Anwendungen sehen Sie als relevant oder werden bereits umgesetzt?**

**Frage 4: Welche KI-Anwendungen werden Sie in den nächsten 5 Jahren angehen?**

**Frage 5: Was sehen Sie als die größten technischen und organisatorischen Hindernisse bei der Anwendung der KI?**

- Neue Produkte oder Dienstleistungen
- Verbesserung der Qualität
- Automation wiederkehrender Tätigkeiten
- Tieferes Verständnis interner Prozesse
- Erhöhung der Flexibilität
- Imagegewinn für die Firma
- Weiterer Nutzen (Bitte erläutern)

**Frage 6: Welche Kompetenzen wollen Sie intern aufbauen, welche extern erwerben? (Make or Buy)**

- ML Expertise (Umsetzung der ML Modelle)
- MLOps (Wartung der ML Modelle)
- Datenarchitektur (Datenbereitstellung und -haltung)
- KI Projektmanagement
- KI Strategie
- Weitere Kompetenz (Bitte erläutern)

**Frage 7: Welche Kriterien sind bei der Auswahl von KI Partnern für Sie entscheidend?**

A.2 Zum Aufbau der Fallbasis verwendete Literatur

Tabelle A.1: Aufgabe des Agenten in der betrachteten Literatur

Quelle	Aufgaben	Anzahl
ANDRYCHOWICZ et al. (2017)	Erreichen, Greifen	2
BEJAR & MORAN (2018)	Erreichen	1
BERSCHIED et al. (2019)	Greifen	1
BHARADHWAJ et al. (2019)	Orientieren	1
BREYER et al. (2018)	Greifen	1
BRUIN et al. (2016)	Erreichen	1
CHEBOTAR et al. (2017)	Zusammensetzen durch Erreichen	1
DALAL et al. (2018)	Bahnfolgen	1
BOURDONNAYE et al. (2018)	Erreichen	1
DU et al. (2021)	Peg-in-Hole durch Erreichen	1
EL-SHAMOUTY et al. (2019)	Greifen	1
FLORENSA et al. (2017)	Zusammensetzen durch Erreichen	1
FRANCESCHETTI et al. (2020)	Erreichen, Greifen	2
GHADIRZADEH et al. (2017)	Greifen	1
GU et al. (2017)	Greifen durch Erreichen	1
GUDIMELLA et al. (2017)	Pick&Place durch Greifen und Pose erreichen	2
HA et al. (2018)	Lokomotion durch Erreichen	1
HAARNOJA et al. (2018)	Schieben, Zusammensetzen	2
HEESS et al. (2017)	Lokomotion	1
INOUE et al. (2017)	Peg-in-Hole durch Suchen und Einsetzen	2
IRIONDO et al. (2019)	Erreichen	1
IRPAN et al. (o. D.)	Greifen	1
JAMES et al. (2018)	Greifen	1
JAMES & JOHNS (2016)	Greifen durch Erreichen	1
KALASHNIKOV et al. (2018)	Greifen durch Erreichen	1
W. KIM et al. (2017)	Erreichen	1
KINDLE et al. (2020)	Bahnfolgen	1
LÄMMLER et al. (2020)	Peg-in-Hole durch Suchen und Einsetzen	2
LEVINE et al. (2016)	Zusammensetzen, Schrauben	2
X. LI et al. (2017)	Erreichen, Zusammensetzen durch Erreichen	2
X. LIANG et al. (2018)	Lokomotion	1
LILLICRAP et al. (2016)	Erreichen, Greifen durch Erreichen	2
KENG et al. (2019)	Erreichen	1
LOPEZ et al. (2019)	Erreichen, Orientieren, Pose erreichen	4
MAHMOOD et al. (2018)	Erreichen	1
MALLOZZI et al. (2018)	Lokomotion	1
MATAS et al. (2018)	Falten durch Greifen und Pose erreichen	1

Fortsetzung auf nächster Seite

**Tabelle A.1 – Fortsetzung von vorheriger Seite**

<b>Quelle</b>	<b>Aufgaben</b>	<b>Anzahl</b>
AHN et al. (2019)	Schrauben durch Orientieren	1
MURATORE et al. (2018)	Fixieren durch Position halten	1
NAIR et al. (2018)	Schieben, Erreichen	2
OPENAI et al. (2018)	Orientieren	1
OPENAI et al. (2019)	Orientieren	1
PENG et al. (2017)	Schieben durch Erreichen	1
PHAM et al. (2018)	Erreichen	1
PHANITEJA et al. (2018b)	Greifen durch Erreichen	1
PHANITEJA et al. (2018a)	Erreichen	1
PINTO et al. (2017)	Greifen	1
POPOV et al. (2017)	Greifen, Zusammensetzen, Pick&Place	3
QIAO et al. (2008)	Erreichen	1
SADEGHI et al. (2017)	Greifen durch Erreichen	1
SCHOETTLER et al. (2019)	Zusammensetzen	1
SHAO et al. (2018)	Greifen durch Erreichen	1
SURMANN et al. (2020)	Navigation durch Erreichen	1
TAI et al. (2017)	Navigation durch Erreichen	1
TAN et al. (2018)	Lokomotion	1
THOMAS et al. (2018)	Zusammensetzen durch Bahnfolgen	1
VECERIK et al. (2017)	Zusammensetzen durch Erreichen	1
YU et al. (2017)	Greifen durch Erreichen	1
WU et al. (2017)	Navigation durch Erreichen	1
XIE et al. (2018)	Navigation durch Erreichen	1
YE et al. (2018)	Navigation durch Erreichen	1
Y. ZHU et al. (2017)	Navigation durch Erreichen	1
ZENG et al. (2018)	Greifen	1
H. ZHU et al. (2019a)	Schrauben durch Orientieren	1

## A.2 Zum Aufbau der Fallbasis verwendete Literatur

Tabelle A.2: In der automatischen Hyperparameter-Konfiguration verwendete Fälle

Fall	Quelle	Umgebung
1	Unity ML-Agents	UnityReacher-v0
2	Unity ML-Agents	Basic
3	Unity ML-Agents	3DBall
4	Unity ML-Agents	GridWorld
5	Unity ML-Agents	Push Block
6	Unity ML-Agents	VisualPushBlock
7	Unity ML-Agents	Hallway
8	Unity ML-Agents	VisualHallway
9	Unity ML-Agents	WormStatic
10	Unity ML-Agents	WormDynamic
11	Unity ML-Agents	3DBall Hard
12	SLM-Lab	Reacher-v1
13	SLM-Lab	UnityReacher-v0
14	SLM-Lab	Unity3DBall
15	SLM-Lab	UnityPushBlock
16	SLM-Lab	Unitybasic
17	SLM-Lab	Unity3DBallHard
18	MAHMOOD et al. 2018	UR-Reacher-2
19	MAHMOOD et al. 2018	UR-Reacher-6
20	MAHMOOD et al. 2018	DXL-Reacher
21	MAHMOOD et al. 2018	DXL-Tracker
22	Open AI Baselines	Reacher-v1
23	RLzoo	Reacher-v2
24	RLzoo	FetchPickAndPlace-v1
25	RLzoo	FetchPush-v1
26	RLzoo	FetchReach-v1
27	RLzoo	FetchSlide-v1
28	RLzoo	HandReach-v0
29	RLzoo	ManipulatorBringball-v0
30	RLzoo	ReacherHard-v0
31	RLzoo	FingerTurn-v0
32	RLzoo	ReachAndDrag
33	RLzoo	OpenBox
34	RLzoo	PushButton
35	RLzoo	SlideBlockToTarget
36	LOPEZ et al. 2019	MARA
37	LOPEZ et al. 2019	MARA Orient
38	LOPEZ et al. 2019	MARA Collision
39	LOPEZ et al. 2019	MARA Collision Orient
40	IRIONDO et al. 2019	miiwa Pick and Place v1
41	IRIONDO et al. 2019	miiwa Pick and Place v2
42	KINDLE et al. 2020	RoyalPanda

A.3 Informationsmodell

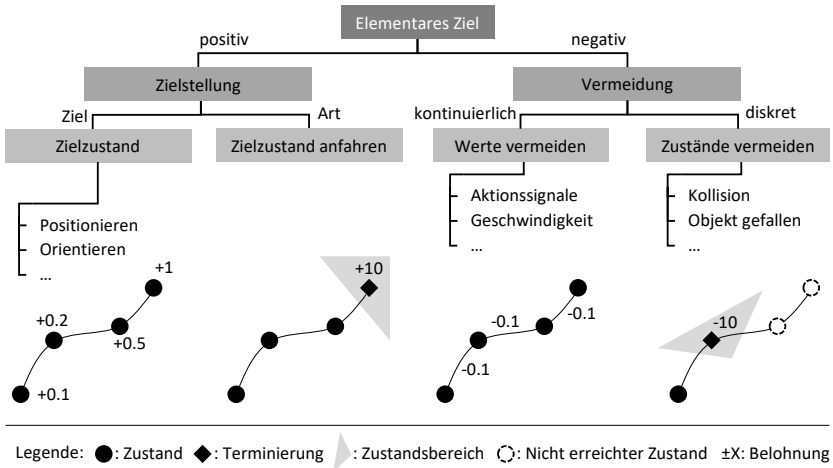


Abbildung A.1: Elementare Ziele von Belohnungsfunktionen

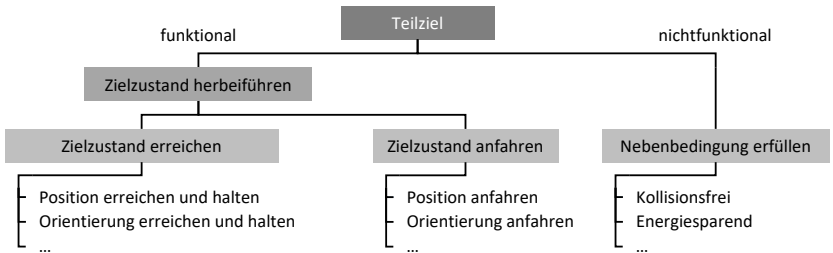


Abbildung A.2: Teilziele von Belohnungsfunktionen



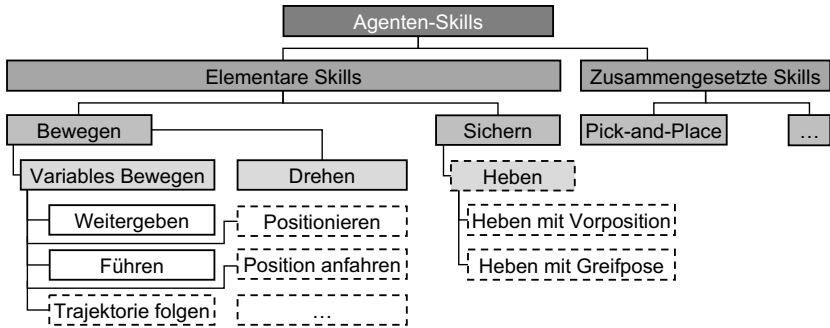


Abbildung A.3: Adaption der Skill-Taxonomie der aufgabenorientierten Programmierung (BACKHAUS & REINHART 2017, HAMMERSTINGL 2020) für Agenten-Skills

### A.4 Versuchsergebnisse

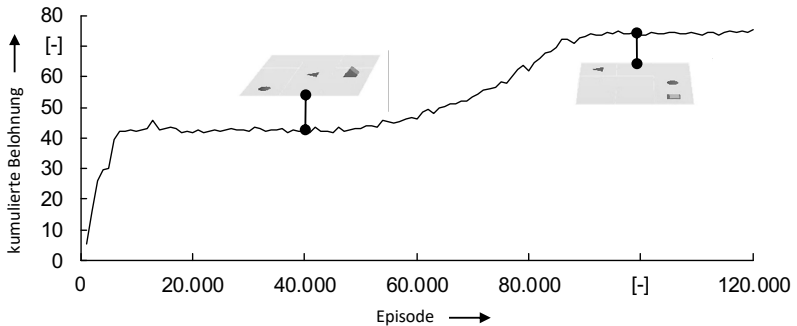


Abbildung A.4: Trainingsverlauf im Testszenario zum Zusammensetzen unter Produktvarianz.

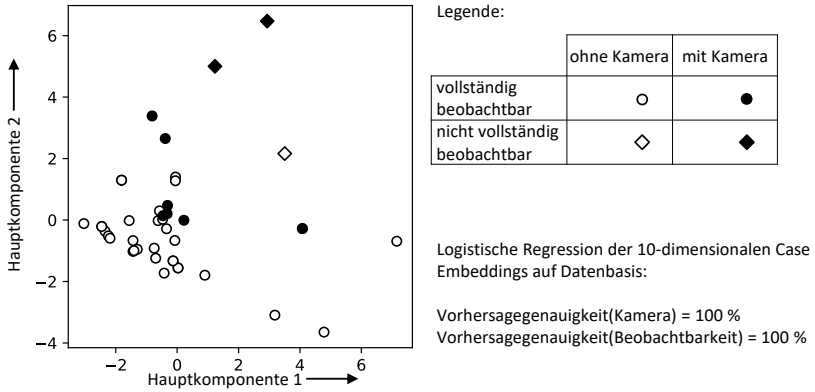


Abbildung A.5: Beispiel einer Hauptkomponentenanalyse mit wenig Aussagekraft hinsichtlich der kamerabasierten Wahrnehmung.

## Verzeichnis betreuter Studienarbeiten

Im Rahmen dieser Dissertation entstanden unter wesentlicher wissenschaftlicher, fachlicher und inhaltlicher Anleitung des Autors die im Folgenden aufgeführten studentischen Arbeiten. Deren Ergebnisse sind in Teilen in die vorliegende Arbeit eingeflossen. Der Autor dankt allen Studierenden für ihr Engagement bei der Unterstützung dieser wissenschaftlichen Arbeit. Nachfolgend sind die Studienarbeiten chronologisch aufgelistet:

*Kumar, Vakul*: Simulation and Modeling of Adhesive Flow based on Smoothed Particle Hydrodynamics. Masterarbeit. Abgabe: Mai 2018

*Lauterbach, Marco*: Anwendung von Reinforcement Learning in der Planung von Montageprozessen. Masterarbeit. Abgabe: November 2018

*Ortiz, Diego*: Automatische Generierung von Lernumgebungen für autonome Montagesysteme. Masterarbeit. Abgabe: März 2019

*Schuster, Dirk*: Methode zur Modellierung von Aufgabenstellungen im Einsatz von Reinforcement Learning in der Montage. Bachelorarbeit. Abgabe: Mai 2019

*Zhao, Min Min*: Wissensbasierte Initialisierung bayesscher Optimierung in der Entwicklung autonomer Robotersysteme. Masterarbeit. Abgabe: November 2020

*Yang, Zhihao*: Wissensbasierte Entwicklung auf Reinforcement Learning basierender Robotersysteme. Bachelorarbeit. Abgabe: Dezember 2020

*Ishaq, Fahad*: Domänenspezifische Konfiguration auf Deep Reinforcement Learning basierender Robotersysteme. Masterarbeit. Abgabe: Dezember 2020

Zudem möchte ich den Studenten Bernhard Glauber, Pavlo Mospan und Fabian Sommerfeld danken, welche als studentische Hilfskräfte an der Umsetzung der Systembausteine mitgewirkt haben.

