STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

Felix Spenrath

»Heuristisches Suchverfahren für die effiziente Planung zum Greifen ungeordnet gelagerter Werkstücke mit Industrierobotern«

Fraunhofer IPA

Universität Stuttgart

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG BAND 137

Felix Spenrath

»Heuristisches Suchverfahren für die effiziente Planung zum Greifen ungeordnet gelagerter Werkstücke mit Industrierobotern«

Herausgeber

Univ.-Prof. Dr.-Ing. Thomas Bauernhansl[1,2]
Univ.-Prof. Dr.-Ing. Dipl.-Kfm. Alexander Sauer[1,3]
Univ.-Prof. Dr.-Ing. Kai Peter Birke[4]
Univ.-Prof. Dr.-Ing. Marco Huber[1,2]

[1]Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
[2]Institut für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart
[3]Institut für Energieeffizienz in der Produktion (EEP) der Universität Stuttgart
[4]Institut für Photovoltaik (*ipv*) der Universität Stuttgart

Kontaktadresse:
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA
Nobelstr. 12
70569 Stuttgart
Telefon 0711 970-1101
info@ipa.fraunhofer.de
www.ipa.fraunhofer.de

Bibliographische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliografische Daten sind im Internet über http://dnb.de abrufbar.

Zugl.: Stuttgart, Univ., Diss., 2022

D 93

2022

Druck und Weiterverarbeitung:
Fraunhofer Verlag, Mediendienstleistungen, Stuttgart, 2022
Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

Heuristisches Suchverfahren für die effiziente Planung zum Greifen ungeordnet gelagerter Werkstücke mit Industrierobotern

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik
der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

Vorgelegt von
Felix Spenrath
aus Berlin

Hauptberichter:	Prof. Dr.-Ing. Andreas Pott
Mitberichter:	Univ.-Prof. Dr.-Ing. Marco Huber
Tag der mündlichen Prüfung:	5. April 2022

Institut für Steuerungstechnik der Werkzeugmaschinen und
Fertigungseinrichtungen (ISW) der Universität Stuttgart

2022

Vorwort des Autors

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Produktionstechnik und Automatisierung in Stuttgart.

Besonderer Dank gilt Prof. Dr.-Ing. Andreas Pott für die Unterstützung und Förderung meiner wissenschaftlichen Arbeit und die Übernahme des Hauptberichts. Univ.-Prof. Dr.-Ing. Marco Huber danke ich für die Übernahme des Mitberichts.

Weiterhin möchte ich mich bei Dr.-Ing. Werner Kraus und Dipl.-Ing. Richard Bormann, M. Sc. für ihr inhaltliches Feedback zu meiner Dissertation und für die gute Arbeitsatmosphäre bedanken, zu der sie in ihrer Funktion als Abteilungs- bzw. Gruppenleiter maßgeblich beigetragen haben. Dr.-Ing. Alexander Kuss danke ich für die ausführliche Durchsicht meines Manuskripts, die inhaltlichen Anregungen sowie die motivierenden Gespräche. Bei meinen Kollegen und Freunden Manuel Mönnig, M. Sc. und Marius Moosmann, M. Sc. bedanke ich mich für die inhaltlichen Diskussionen und das hilfreiche Feedback zur Arbeit sowie für die gemeinsame Zeit am Fraunhofer IPA. Meinen ehemaligen Kollegen Dipl.-Math. Marc Teschner, Dr.-Ing. Alexander Spiller und Dr.-Ing. Matthias Palzkill danke ich für die kollegiale und freundschaftliche Unterstützung sowie für die großartige Zusammenarbeit. Bei allen weiteren Kollegen, die mich während meiner Zeit in der Gruppe 321 begleitet haben, möchte ich mich ebenfalls für die hervorragende Zusammenarbeit bedanken.

Luzia Schuhmacher, M. A. danke ich herzlich für die sorgfältige und sicherlich zeitaufwendige Durchsicht des Manuskripts. Für die hervorragende Unterstützung bei allen organisatorischen Fragen bedanke ich mich bei Heide Kreuzburg. Schließlich danke ich meiner Familie, die mir ein Studium ermöglicht und mich jederzeit uneingeschränkt gefördert hat.

Stuttgart, im September 2022 Felix Spenrath

Kurzinhalt

Im Rahmen dieser Arbeit wird eine effiziente Greifplanung für die modellbasierte Entnahme von Werkstücken durch einen Industrieroboter entwickelt, die insbesondere beim sogenannten Griff-in-die-Kiste zum Einsatz kommen kann. Das Verfahren basiert auf einem Suchbaum, der so konstruiert wird, dass alle relevanten Aspekte der Greifplanung enthalten sind, und der alle potenziell möglichen Griffe enthält. Kriterien, die einen erfolgreichen Griff verhindern können, werden beim Expandieren des Suchbaums überprüft, so dass jeder Zielknoten einen durchführbaren Entnahmevorgang repräsentiert. Diese Prüfungen beinhalten beispielsweise Kollisionstests basierend auf einer Punktwolke der aktuellen Szene, um einen kollisionsfreien Griff gewährleisten zu können.

Um trotz des großen Suchraums eine kurze Rechenzeit zu erreichen, kommt bei der Expansion des Suchbaums ein heuristisches Suchverfahren zum Einsatz. Die für das Suchverfahren verwendete Heuristikfunktion wird dabei so definiert, dass sie auf Basis diverser Eigenschaften eines Knotens die Wahrscheinlichkeit abschätzt, unter den Nachfolgern dieses Knotens eine Greiflösung zu finden. Um eine Aussage darüber zu treffen, von welchen Eigenschaften die Wahrscheinlichkeit einer geeigneten Greiflösung abhängt, werden in dieser Arbeit einige Eigenschaften verschiedener Komponenten analysiert. Zudem wird in dieser Arbeit die Verwendung von künstlichen neuronalen Netzen in der Heuristikfunktion untersucht. Die neuronalen Netze lernen dabei, welche Knoten des Suchbaums expandiert werden sollten, um möglichst schnell eine Lösung zu finden. Die in der Heuristikfunktion enthaltenen Gewichtungsfaktoren werden durch Optimierungsverfahren bestimmt.

Zum Abschluss wird die in dieser Arbeit entwickelte Greifplanung anhand zweier Versuchsaufbauten experimentell untersucht. Zudem wird durch die Auswertung von über 1,5 Mio. Griffen einer industriellen Anwendung gezeigt, dass die entwickelte Greifplanung für die Praxis geeignet ist.

Short Summary

In this dissertation, a method for efficient grasp planning for the model-based extraction of workpieces by a robot is presented, which can be used particularly for the so-called bin picking process. The method is based on a search tree, which is constructed in a way that all relevant aspects of grasp planning are included and which contains all potentially possible grasps. Criteria, which can prevent a successful grasp, are checked during the expansion of the search tree so that each destination node represents a feasible grasp. These checks include for example collision tests based on a point cloud of the current scene to guarantee a collision-free grasp.

To achieve a short calculation time despite the large search space, a heuristic search algorithm is used during the expansion of the search tree. The heuristic function used for this algorithm is defined so that the probability to find a destination node underneath a node is estimated based on several properties of this node. To get information about the properties, which the probability of a feasible grasp depend on, several properties of different components are analyzed in this work. Additionally, the use of artificial neural networks within the heuristic function is analyzed. These neural networks learn which nodes of the search tree should be expanded to find a solution as quickly as possible. The weighting factors used in the heuristic function are determined by optimization algorithms.

Finally, the grasp planning method is examined in experiments and, by analyzing more than 1.5 million grasps from an industrial application, it is shown that the grasp planning method is suitable for practice.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AABB	Axis-Aligned Bounding Box
BO	Bayessche Optimierung
CAD	Computer Aided Design
FANN	Fast Artificial Neural Network
ID	Identifikationsnummer
KLT	Kleinladungsträger
LED	Light-emitting diode
LIN	Linearbewegung
NN	(Künstliches) Neuronales Netz
ODE	Open Dynamics Engine
PTP	Punkt-zu-Punkt-Bewegung (Point-to-point)
SPS	Speicherprogrammierbare Steuerung
STEP	Standard for the exchange of product model data
STL	Standard Triangulation Language
TCP/IP	Transmission Control Protocol / Internet Protocol
TCP	Tool Center Point / Tool Center Pose
TPE	Tree-structured Parzen Estimator
VRML	Virtual Reality Modeling Language

Symbolverzeichnis

Lateinische Symbole

Symbol	Beschreibung
A	Ebene
$\boldsymbol{a}$	Achse eines rotatorischen oder translatorischen Freiheitsgrades
$\boldsymbol{b}$	Vektor, der die bevorzugte Orientierung von Komponenten beschreibt
b_g	Bewertungsheuristik eines Knotens basierend auf den gültigen und ungültigen Nachfolgern
b_n	Bewertungsheuristik eines Knotens basierend auf der erwarteten Anzahl zu generierender Nachfolger zur Erreichung einer Greiflösung
b_s	Bewertungsheuristik eines Knotens basierend auf der statistischen Wahrscheinlichkeit zur Erreichung einer Greiflösung
b_z	Bewertungsheuristik eines Knotens basierend auf der Anzahl der Zielknoten in den Nachfolgern
c_d	Dynamische Kosten eines Knotens im Suchbaum
$c_{\tilde{d}}$	Monoton steigende, dynamische Kosten eines Knotens im Suchbaum
c_{dof}	Durch den übergeordneten Freiheitsgrad verursachte Kosten
c_{exp}	Heuristik für mehrfach expandierte Knoten
c_l	Durch den Konfidenzwert der Objektlageerkennung verursachte Kosten
c_n	Kosten der Nachfolger eines Knoten im Suchbaum
c_{nn}	Kosten des neuronalen Netzwerks
c_{ori}	Durch die Orientierung einer Komponente verursachte Kosten
c_{prio}	Durch die Priorität der Greifposen verursachte Kosten
c_{xy}	Durch die horizontale Werkstückposition verursachte Kosten
c_{xy}^{ecke}	Kosten für den horizontalen Abstand zur nächstgelegenen Kistenecke
c_{xy}^{mitte}	Kosten für den horizontalen Abstand zur Kistenmitte
c_{xy}^{wand}	Kosten für den horizontalen Abstand zur nächstgelegenen Kistenwand
c_z	Durch die vertikale Werkstückposition verursachte Kosten
c_z^L	Durch die vertikale Werkstückposition verursachte Kosten, berechnet auf Basis der Höhe des Ladungsträgers

Symbol	Beschreibung
$c_z^{max_1}$	Durch die vertikale Werkstückposition verursachte Kosten, berechnet auf Basis des höchsten erkannten Werkstücks (Variante 1)
$c_z^{max_2}$	Durch die vertikale Werkstückposition verursachte Kosten, berechnet auf Basis des höchsten erkannten Werkstücks (Variante 2)
c_z^W	Durch die vertikale Werkstückposition verursachte Kosten, berechnet auf Basis des höchsten und des niedrigsten erkannten Werkstücks
D	Dreieck
d_K^{wand}	Horizontaler Abstand einer Komponente zur nächstgelegenen Kistenwand
d_{max}	Maximal erwarteter Abstand eines Knotens zur Wurzel des Suchbaums
d_W	Abstand eines Knotens zur Wurzel des Suchbaums
d_W^{ecke}	Horizontaler Abstand eines Werkstücks zur nächstgelegenen Kistenecke
d_W^{mitte}	Horizontaler Abstand eines Werkstücks zur Kistenmitte
d_W^{wand}	Horizontaler Abstand eines Werkstücks zur nächstgelegenen Kistenwand
d_x^L	Länge des Ladungsträgers
d_y^L	Breite des Ladungsträgers
d_Z	Abstand eines Knotens zum nächstgelegenen Zielknoten unter seinen Nachfolgern
d_z^L	Höhe des Ladungsträgers
E	Fehler beim Trainieren eines neuronalen Netzes
F	Flächenmodell
f	Fitnessfunktion des evolutionären Algorithmus
f_L	Füllgrad des Ladungsträgers
G	Greifpose
g_A	Faktor der Unterabtastung beim Kollisionstest
h	Durch die Heuristikfunktion berechnete Kosten eines Knotens im Suchbaum
h_F	Statischer Teil der Heuristikfunktion eines Knotens im Suchbaum
h_K^G	Spezifischer Teil der Heuristikfunktion für Greifposen
h_K	Spezifischer Teil der Heuristikfunktion eines Knotens im Suchbaum
h_K^{dof}	Spezifischer Teil der Heuristikfunktion für Freiheitsgrade
h_K^T	Spezifischer Teil der Heuristikfunktion für TCPs
h_K^V	Spezifischer Teil der Heuristikfunktion für Verbindungsposen
h_K^W	Spezifischer Teil der Heuristikfunktion für Werkstücke
i	Laufvariable

Symbol	**Beschreibung**
j	Laufvariable
K	Knoten des Suchbaums
$\mathcal{K}_K$	Koordinatensystem der einem Suchbaumknoten zugehörigen Komponenteninstanz
$\mathcal{K}_0$	Weltkoordinatensystem
$\mathcal{K}_E$	Koordinatensystem eines Greiferelements
$\mathcal{K}_F$	Koordinatensystem des Handflansches des Roboters
$\mathcal{K}_G$	Koordinatensystem einer Greifpose
$\mathcal{K}_L$	Koordinatensystem des Ladungsträgers
$\mathcal{K}_S$	Koordinatensystem des Sensors
$\mathcal{K}_T$	Koordinatensystem eines Tool Center Points
$\mathcal{K}_V$	Koordinatensystem einer Verbindungspose
$\mathcal{K}_W$	Koordinatensystem des Werkstücks
k	Laufvariable
N_a	Anzahl zusätzlicher Greiferachsen
N_A	Anzahl der Bahnposen der Anfahrbahn
N_{aus}	Anzahl an Ausgabewerten eines neuronalen Netzes
N_C	Anzahl an Kollisionspunkten in der Punktwolke
N_C^{max}	Maximale Anzahl an zulässigen Kollisionspunkten
N_{col}	Anzahl an Kollisionstests
N_d^{min}	Mindestanzahl an Nachfolgern zur Berechnung der dynamischen Kosten eines Knotens
N_{dof}	Anzahl an Schritten für einen diskreten Freiheitsgrad
N_{dof}^{max}	Maximale Anzahl an Schritten bei der Diskretisierung eines kontinuierlichen Freiheitsgrades
N_E	Anzahl der Bahnposen der Entnahmebahn
N_{eval}	Anzahl evaluierter Knoten zur Erreichung eines Zielknotens
N_{exp}	Anzahl expandierter Knoten zur Erreichung eines Zielknotens
N_{exp}^{dof}	Anzahl der bereits durchgeführten Expansionen eines Freiheitsgradknotens
N_{exp}^{max}	Maximale Anzahl an Expansionen eines Knotens
N_F	Anzahl an definierten Freiheitsgraden
N_G	Anzahl an definierten Greifposen an einem Werkstück
N_k	Anzahl aller Kinder eines Knotens im Suchbaum
N_L	Anzahl an Greiflösungen
N_n	Anzahl aller Nachfolger eines Knotens im Suchbaum

Symbol	**Beschreibung**
N_n^b	Anzahl aller Blätter unter den Nachfolgern eines Knotens im Suchbaum
N_n^g	Anzahl gültiger Nachfolger eines Knotens im Suchbaum
N_n^z	Anzahl an Zielknoten unter den Nachfolgern eines Knotens im Suchbaum
N_P	Anzahl an Punkten in einer Punktwolke
N_S	Anzahl an untersuchten Situationen
$N_{\tilde{S}}$	Anzahl an Situationen, bei denen mindestens eine Lösung existiert
N_S^A	Anzahl an Schritten einer gegebenen zusätzlichen Greiferachse
N_{sek}	Anzahl an Sektionen je Koordinatenachse
N_S^F	Anzahl an Schritten eines Freiheitsgrades einer Greifpose
N_T	Anzahl an parallel arbeitenden Threads
N_W	Anzahl an erkannten Werkstücklagen
N_W^L	Anzahl an Werkstücken im Ladungsträger
$\boldsymbol{n}_K$	Negative Richtung der z-Achse einer Komponente
$\boldsymbol{n}_T$	Negative Richtung der z-Achse des TCPs
$\widehat{\boldsymbol{n}}_T$	Projizierte negative Richtung der z-Achse des TCPs
P	Punktwolke
P_1^A	Erste Pose der Anfahrbahn
P_1^E	Erste Pose der Entnahmebahn
$P_{N_A}^A$	Letzte Pose der Anfahrbahn
$P_{N_E}^E$	Letzte Pose der Entnahmebahn
p_i	Punkt
Q	Vorrangwarteschlange
q	Anteil der Situationen, in denen eine Lösung existiert
$\boldsymbol{R}$	Matrix, welche eine Rotation im dreidimensionalen Raum beschreibt
R_l	Konfidenzwert der Objektlageerkennung
r_{exp}	Faktor in der Fitnessfunktion des evolutionären Algorithmus
S	Sektion der in mehrere Sektionen zerteilten Punktwolke P
sig	Sigmoid-Funktion
$\boldsymbol{T}$	Homogene Transformationsmatrix
$\boldsymbol{t}$	Vektor, welcher eine Translation im dreidimensionalen Raum beschreibt
t_{max}	Maximale Rechenzeit für die Greifplanung
$\boldsymbol{t}_{xy}$	Horizontale Komponente des Vektors vom Koordinatensystem $\mathcal{K}_L$ des Ladungsträgers zum TCP-Koordinatensystem $\mathcal{K}_T$

Symbol	Beschreibung
u	In den Trainingsdaten enthaltener gewünschter Ausgabewert eines neuronalen Netzes
v	Berechneter Ausgabewert eines neuronalen Netzes
w	Gewichtungsfaktor
w_d	Gewichtungsfaktor für die dynamischen Kosten eines Knotens im Suchbaum
w_{dof}	Gewichtungsfaktor für die durch den übergeordneten Freiheitsgrad verursachten Kosten
w_{exp}	Gewichtungsfaktor für die komponentenspezifische Heuristik für expandierte Freiheitsgrade
w_G	Gewichtungsfaktor für die komponentenspezifische Heuristik für Greifposen
w_l	Gewichtungsfaktor für die durch den Konfidenzwert der Objektlageerkennung verursachten Kosten
w_{nn}^G	Gewichtungsfaktor für das neuronalen Netz für Greifposen
w_{nn}^T	Gewichtungsfaktor für das neuronalen Netz für TCPs
w_{nn}^V	Gewichtungsfaktor für das neuronalen Netz für Verbindungsposen des Greifers
w_{nn}^W	Gewichtungsfaktor für das neuronalen Netz für Werkstücke
w_T	Gewichtungsfaktor für die komponentenspezifische Heuristik für TCPs
w_V	Gewichtungsfaktor für die komponentenspezifische Heuristik für Verbindungsposen
w_W	Gewichtungsfaktor für die komponentenspezifische Heuristik für Werkstücke
w_z	Gewichtungsfaktor für die durch die vertikale Werkstückposition verursachten Kosten
x	Wert auf oder Translation entlang der x-Achse
y	Wert auf oder Translation entlang der y-Achse
z	Wert auf oder Translation entlang der z-Achse
z_Δ^{exp}	Maximal erwartete Höhendifferenz zwischen zwei erkannten Werkstücklagen
$\boldsymbol{z}_K$	z-Achse einer Komponente
$\boldsymbol{z}_L$	z-Achse des Ladungsträgers
${}^L\bar{z}_p$	Mittlere Höhe aller Punkte der Punktwolke innerhalb des Ladungsträgers
${}^L z_W$	Höhe eines Werkstücks im Koordinatensystem $\mathcal{K}_L$ des Ladungsträgers

Symbol	Beschreibung
${}^{L}z_{W}^{max}$	Höhe des höchsten erkannten Werkstücks im Koordinatensystem $\mathcal{K}_L$ des Ladungsträgers
${}^{L}z_{W}^{min}$	Höhe des niedrigsten erkannten Werkstücks im Koordinatensystem $\mathcal{K}_L$ des Ladungsträgers
$\boldsymbol{z}_T$	z-Achse eines TCPs

Griechische Symbole

Symbol	Beschreibung
α	Konkreter Wert eines Freiheitsgrades
α_{max}	Maximal zulässiger Wert eines Freiheitsgrades
α_{min}	Minimal zulässiger Wert eines Freiheitsgrades
β	Winkel zwischen der bevorzugten Orientierung und der z-Achse des Ladungsträgers
β_{max}	Maximaler Winkel zwischen der bevorzugten Orientierung und der z-Achse des Ladungsträgers
γ	Abweichung von der bevorzugten Orientierung
γ_{max}	Maximal erlaubte Abweichung von der bevorzugten Orientierung
δ	Abstand für die Bahninterpolation
δ_{max}	Maximaler Abstand für die Bahninterpolation
δ_{min}	Minimaler Abstand für die Bahninterpolation
ϵ	Geforderte Genauigkeit bei der Diskretisierung eines kontinuierlichen Freiheitsgrades
ϑ	Winkel der Abweichung eines TCPs von der Vertikalen
$\widehat{\vartheta}$	Winkel der projizierten Abweichung eines TCPs von der Vertikalen
ξ	Winkel einer Roboterachse
ξ_i^{max}	Maximal zulässiger Winkel der Roboterachse i
ξ_i^{min}	Minimal zulässiger Winkel der Roboterachse i
ξ_{singu}	Grenzwert für die Achsen 3 und 5 des Roboters zur Erkennung von Singularitäten
ψ	Rotation um die x-Achse
σ_R	Standardabweichung des Rauschens zur Erzeugung virtueller Sensordaten
θ	Rotation um die y-Achse
ϕ	Rotation um die z-Achse

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Dieses Kapitel führt in die Thematik der industriellen Werkstückzuführung und insbesondere in die Greifplanung für dafür verwendete Roboteranlagen ein. Dazu wird zunächst die derzeitige Ausgangssituation in der Industrie vorgestellt und daran die Motivation der Arbeit erläutert. Im Anschluss bietet Abschnitt 1.2 eine Übersicht über den Stand der Forschung verschiedener Ansätze für die Greifplanung. In Abschnitt 1.3 erfolgt die Präsentation der zu erreichenden Ziele sowie eine Abgrenzung von den mit der Greifplanung verwandten Themen. Zum Abschluss des Kapitels wird der in dieser Arbeit entwickelte Lösungsansatz kurz vorgestellt und damit ein Überblick über die weitere Gliederung der Arbeit gegeben.

1.1 Ausgangssituation und Motivation

Da die meisten Fertigungsprozesse eine definierte Lage der Werkstücke voraussetzen, wird in automatisierten Produktionsanlagen häufig versucht, diese Lage auch zwischen den einzelnen Prozessen zu erhalten, beispielsweise durch das Ablegen von Werkstücken in definierte Formnester. Während dieses Ziel bei verketteten Maschinen meist ohne großen Aufwand erreicht werden kann, sind dafür sowohl in der Logistik als auch im innerbetrieblichen Materialfluss Sonderladungsträger oder Kisten mit speziellen Einlagen (Blistern) notwendig. Der Transport und die Bereitstellung dieser Sonderladungsträger bedeuten, insbesondere bei kleinen Losgrößen, einen erheblichen Mehraufwand, weshalb Werkstücke heutzutage in der industriellen Fertigung nach wie vor meist ungeordnet in Standardladungsträgern, wie beispielsweise Kisten oder Gitterboxen, gelagert und transportiert werden. Um die Werkstücke im nächsten Fertigungsschritt weiter bearbeiten zu können, müssen diese wieder vereinzelt und lagerichtig zugeführt werden. Für kleine Werkstücke werden hierfür oft mechanische Vereinzelungslösungen, wie zum Beispiel Stufenförderer oder Vibrationswendelförderer eingesetzt. Eine Übersicht über verschiedene mechanische Vereinzelungslösungen findet sich in Boothroyd (2005). Diese müssen allerdings meist aufwändig auf ein bestimmtes Werkstück eingestellt werden, was lange Rüstzeiten zur Folge hat. Solche mechanischen Vereinzelungslösungen sind weder flexibel noch wandlungsfähig. Bei großen und schweren Werkstücken sind mechanische Lösungen zudem sehr laut und verschleißen schnell. Hinzu kommt, dass die Ladungsträger dazu vorab ausgekippt werden müssen, was

insbesondere bei großen und metallischen Werkstücken eine enorme Geräuschbelastung darstellt. Bei empfindlichen Werkstücken kann es dadurch auch zu Beschädigungen kommen. Daher werden große und schwere Werkstücke häufig einzeln von Mitarbeitern aus den Ladungsträgern entnommen. Je nach Masse der Werkstücke und geforderter Taktzeit stellt dies eine monotone, anstrengende und teilweise auch gesundheitsschädliche Tätigkeit dar. Bei sehr schweren Werkstücken stehen den Mitarbeitern deshalb teilweise Hebehilfen zur Verfügung, was die Arbeit jedoch häufig verlangsamt. Somit gibt es sowohl wirtschaftliche Gründe als auch Gründe bezüglich der Arbeitssicherheit, die Tätigkeit der Werkstückzuführung zu automatisieren.

Industrieroboter könnten aufgrund ihrer Geschwindigkeit und Flexibilität eine Möglichkeit darstellen, diese Probleme zu lösen. Der überwiegende Anteil an Industrierobotern wird heute jedoch für repetitive Aufgaben eingesetzt, bei denen wiederholt dieselben Bewegungen ausgeführt werden müssen. Die Roboterposen für solche Anwendungen können bei der Inbetriebnahme manuell von Mitarbeitern eingelernt werden. Für viele herkömmliche Anwendungen, wie zum Beispiel Schweißen, Lackieren oder Greifen von Werkstücken von festen Positionen, ist dies ausreichend. Um ungeordnete Werkstücke aus einer Kiste zu greifen, muss jedoch die Annahme einer bekannten Werkstückposition aufgegeben werden, was eine individuelle Greif- und Bahnplanung nötig macht. Sensorik und Bildverarbeitung bieten die Möglichkeit, die Lage der einzelnen Werkstücke zu bestimmen, so dass diese gezielt aus der Kiste gegriffen werden können. Die Vereinzelung und Entnahme von Werkstücken aus einer Kiste durch einen Roboter wird häufig als *Griff-in-die-Kiste* oder *(Random) Bin Picking* bezeichnet (Siciliano et al. 2008). Der Griff-in-die-Kiste hat gegenüber den zuvor genannten Methoden zur Werkstückvereinzelung folgende Vorteile:

- **Höhere Flexibilität und Wandlungsfähigkeit:** Eine roboterbasierte Lösung lässt sich schneller auf andere Werkstückvarianten umrüsten als rein mechanische Systeme, bei denen häufig diverse Schikanen aufwändig eingestellt und getestet werden müssen. Falls sich der Greifer für verschiedene Varianten einsetzen lässt oder automatisch gewechselt werden kann, ist eine solche Umrüstung häufig softwaretechnisch möglich, so dass verschiedene Varianten im Mischbetrieb produziert werden können. Dies führt insbesondere zu einer deutlichen Kostenreduzierung bei kleinen Losgrößen, bei denen sich die Herstellung von Komponenten für eine rein mechanische Vereinzelungslösung nicht rentiert.

- **Schonende Handhabung:** Durch das gezielte Greifen einzelner Werkstücke werden diese deutlich weniger mechanischen Belastungen ausgesetzt, als bei rein mechanischen Vereinzelungslösungen.

- **Geringere Geräuschbelastung:** Das gezielte Greifen einzelner Werkstücke durch einen Roboter führt bei schweren Werkstücken zu einer erheblichen Reduzierung des Lärms im

Vergleich zu einer rein mechanischen Vereinzelungslösung, bei der die Werkstücke meist vibrieren oder wiederholt von Stufen bzw. Schikanen herunterfallen.

- **Entlastung von Mitarbeitern:** Es existiert eine Vielzahl an Robotern mit Traglasten im Bereich von mehreren 100 kg. Das Gewicht der Werkstücke stellt daher keine Hürde für die Automatisierung dar. Somit können auch schwere Werkstücke ohne körperliche Belastung der Mitarbeiter mit einer geringen Taktzeit vereinzelt werden.

Der Vollständigkeit halber seien an dieser Stelle noch Lösungen genannt, die das Greifen mittels eines Roboters mit mechanischer Vorvereinzelung kombinieren. Hierbei werden die Werkstücke beispielsweise mittels Förderbändern oder einer vibrierenden Ebene weitgehend vereinzelt und in eine greifbare Lage gebracht. Da somit nur wenige Werkstücke zeitgleich für das Greifen durch den Roboter bereitgestellt werden und Werkstücke, die nicht erkannt oder gegriffen werden können, mechanisch in eine andere Lage gebracht werden können, genügt hier eine Bildverarbeitungslösung auf Basis einer 2D-Kamera. Solche Lösungen bieten ebenfalls einige der oben genannten Vorteile, werden aufgrund des Platzbedarfs und der mechanischen Beanspruchung der Werkstücke jedoch überwiegend für Kleinteile eingesetzt.

Die Anzahl der Industrieroboter, die weltweit im Bereich der Handhabung und Maschinenbeladung eingesetzt werden, ist in den letzten Jahren stetig gestiegen und lag im Jahr 2019 bei 1 215 303, was 45 % der operativen Industrieroboter entsprach (Müller et al. 2020). Trotz des hohen Anteils an Industrierobotern, die für Handhabung und Maschinenbeladung eingesetzt werden, findet der Griff-in-die-Kiste bisher kaum Verbreitung. Dies liegt daran, dass die Vereinzelung von ungeordnet gelagerten Werkstücken nach wie vor eine der anspruchsvollsten Aufgaben für Industrieroboter darstellt, an der seit über 35 Jahren geforscht wird (Ikeuchi et al. 1983; Horn et al. 1983; Horn et al. 1984). Dabei ist zu beachten, dass die Problemstellung lange Zeit etwas vernachlässigt wurde, da für die Produktion von Großserien werkstückspezifische Ladungsträger durchaus wirtschaftlich sein können. Aufgrund der sinkenden Losgrößen in den letzten Jahren, steigt das Interesse an dieser Technologie jedoch stark an.

Ein Ausschlusskriterium für den Griff-in-die-Kiste in industriellen Anwendungen stellt häufig die geforderte Taktzeit dar. Bichler et al. (2005) definieren die Taktzeit als die *„zulässige Bearbeitungsdauer an einer Arbeitsstation im Rahmen einer Taktproduktion“*. Bezogen auf die roboterbasierte Werkstückzuführung stellt dies somit die Zeit dar, die die Anlage benötigen darf, um nach der Zuführung eines Werkstücks das nächste Werkstück bereitzustellen. Abbildung 1.1 zeigt eine vom Autor erhobene Statistik der erforderlichen Taktzeit aus 70 Kundenanfragen. Die Abbildung zeigt anschaulich, welcher Anteil der Kundenanfragen sich mit einer bestimmten Taktzeit erfüllen ließen. Hierbei ist zu beachten, dass diese Abbildung lediglich die Taktzeit berücksichtigt und andere Faktoren vernachlässigt, welche ebenfalls die Machbarkeit beeinflussen

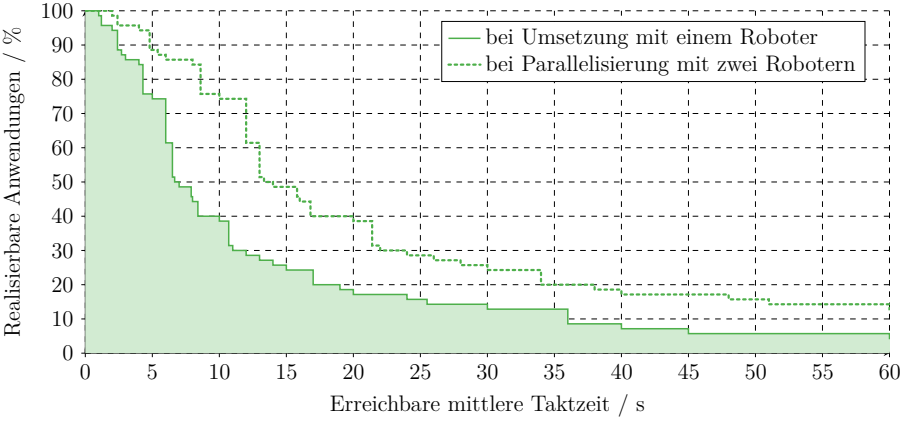


Abbildung 1.1: Anteil der realisierbaren Anwendungen in Abhängigkeit von der mittleren Taktzeit, die eine potenzielle Vereinzelungslösung erreichen kann (eigene Erhebung basierend auf 70 Kundenanfragen aus den Jahren 2014 - 2021)

können. Die Parallelisierung mittels mehrerer Roboter stellt eine Möglichkeit dar, Anwendungen mit einer zu kurzen Taktzeit realisieren zu können. Aufgrund der deutlich höheren Kosten und des höheren Platzbedarfs ist dies allerdings häufig keine wirtschaftliche Option. Zu beachten ist ebenfalls, dass die für einen einzelnen erfolgreichen Entnahmevorgang erreichbare Taktzeit der Anlage geringer sein muss als die notwendige Taktzeit für den Folgeprozess, damit eventuelle Fehlgriffe ausgeglichen werden können. Dafür muss ein entsprechender Teilepuffer im Anlagenkonzept berücksichtigt werden. Wie Abbildung 1.1 zeigt, lassen sich mit einer Lösung, die eine Taktzeit von 10 Sekunden erreicht, nur maximal 40 % der angefragten Anwendungen umsetzen. Beträgt die mittlere Taktzeit dieser Lösung 20 Sekunden, so lassen sich gerade noch ca. 19 % realisieren. Dies zeigt deutlich, wie wichtig eine möglichst kurze Taktzeit ist, um den Griff-in-die-Kiste flächendeckend einsetzen zu können. Ein weiterer Grund, der den Einsatz von Robotern für die Werkstückvereinzelung verhindert, ist eine mangelnde Zuverlässigkeit des Gesamtsystems. Kommt es zu Kollisionen oder ähnlichen Problemen, die zu einem Stillstand der Anlage führen, so verringert dies die Verfügbarkeit. Häufig wird zudem ein hoher Entleerungsgrad der Kiste gefordert, der bei einer roboterbasierten Werkstückvereinzelung nicht immer sichergestellt werden kann. Die Ursache dafür liegt darin, dass die Werkstücke ungeordnet gelagert sind und somit schwierige Situationen auftreten können, die der Roboter nicht auflösen kann. Insbesondere Werkstücke, die in den Ecken der Kiste liegen, stellen eine Herausforderung dar. Diese sind durch die Seitenwände der Kiste nur schwer zugänglich und können dadurch, abhängig von ihrer Orientierung und der Flexibilität des Greifers, unter Umständen nicht entnommen werden. Werkstücke können sich, in der gesamten Kiste, aber auch gegenseitig blockieren und so eine

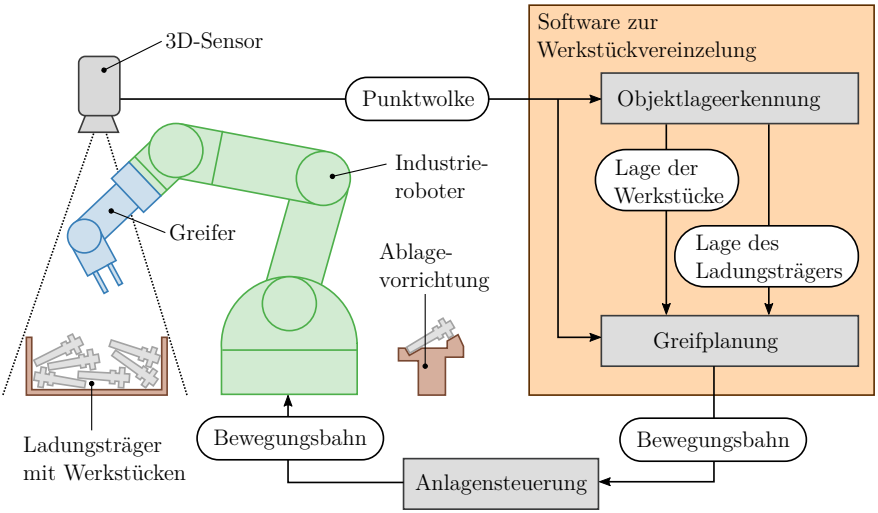


Abbildung 1.2: Schematischer Aufbau einer Roboteranlage für den Griff-in-die-Kiste mit modellbasiertem Ansatz

Entnahme verhindern. Die Komplexität des Gesamtsystems kann ebenfalls eine Hürde darstellen, da sie eine Fehlersuche erschwert. Zudem wird es für den Anlagenbetreiber dadurch schwieriger, selbst neue Werkstücke einzulernen. Bei der Beschreibung der Zielsetzung in Abschnitt 1.3 wird näher auf diese Herausforderungen eingegangen.

Für den Griff-in-die-Kiste existieren diverse Lösungsansätze. Diese lassen sich in Ansätze, die ohne ein (CAD-)Modell des Werkstücks auskommen (modellfreie Ansätze), und Ansätze, die ein Modell des Werkstücks benötigen (modellbasierte Ansätze), unterteilen. Bei modellfreien Ansätzen wird basierend auf den Sensordaten ein geeigneter Griff berechnet, ohne eine Werkstücklage zu bestimmen. Bei modellbasierten Ansätzen wird hingegen zunächst die Werkstücklage ermittelt. Modellbasierte Ansätze lassen sich somit in zwei Teilprobleme unterteilen: Die Bestimmung der Lage einzelner Werkstücke innerhalb des Ladungsträgers und die Planung eines Greifvorgangs inklusive einer kollisionsfreien Bewegungsbahn für den Roboter. Abbildung 1.2 gibt einen schematischen Überblick über den Aufbau einer solchen Roboteranlage mit modellbasiertem Ansatz. Dabei wird der Ladungsträger mit den Werkstücken zunächst mit einem 3D-Sensor erfasst und eine Punktwolke aller sichtbaren Oberflächen generiert. Je nach Anwendung stehen verschiedene Messprinzipien zur Auswahl, die in Abschnitt 2.4.1 kurz vorgestellt werden. Bei modellbasierten Ansätzen werden in dieser Punktwolke zunächst die einzelnen Werkstücke erkannt und deren Lage bestimmt. Dieser Teilschritt wird als *Objektlageerkennung* bezeichnet. Die Greifplanung berechnet, basierend auf der Lage der erkannten Objekte und meist weiterer Informationen,

wie der Lage des Ladungsträgers und der Punktwolke der aktuellen Szene, eine geeignete Greifpose sowie kollisionsfreie Bewegungsbahnen. Diese Bewegungsbahnen führen den Roboter zur berechneten Greifpose und, nach dem Greifen des Werkstücks, wieder aus dem Ladungsträger hinaus. Üblicherweise werden die Objektlageerkennung und die Greifplanung auf demselben Industrie-PC ausgeführt. In industriellen Roboteranlagen übernimmt eine speicherprogrammierbare Steuerung (SPS) die Koordination des Gesamtablaufs. In diesem Fall wird die Anfrage nach einem zu greifenden Werkstück und damit die Erfassung der Szene durch den 3D-Sensor von dieser Steuerung ausgelöst. Die berechnete Bewegungsbahn kann im Anschluss gegebenenfalls von der Steuerung auf Plausibilität geprüft und an den Industrieroboter weitergeleitet werden. Alternativ kann die Anforderung nach einem zu greifenden Werkstück direkt vom Industrieroboter ausgehen. In dem Fall wird die generierte Bewegungsbahn direkt an der Roboter zurück gesandt. Die Kommunikation zwischen der Griff-in-die-Kiste-Software und dem Industrieroboter beziehungsweise der SPS erfolgt meist per TCP/IP. Als Handhabungskinematik kommt bei den hier beschriebenen Vereinzelungsanlagen ein Industrieroboter zum Einsatz. Dieser ist dafür zuständig, den Greifer so zu positionieren, dass ein Werkstück gegriffen und entnommen werden kann. In Abschnitt 2.4.4 wird näher auf die Handhabungskinematik eingegangen. Das physikalische Greifen des Werkstücks erfolgt durch einen anwendungsspezifischen Greifer. Es existieren verschiedene Greifprinzipien, auf die in Abschnitt 2.4.3 näher eingegangen wird. Das gegriffene Werkstück wird nach dem Entnahmevorgang üblicherweise in einer definierten Lage abgelegt. Diese kann sich beispielsweise direkt in einer Bearbeitungsmaschine oder in einer separaten Ablagevorrichtung befinden. Das Werkstück kann von dieser Ablagevorrichtung erneut gegriffen werden, um beispielsweise ein Umgreifen zu realisieren, gegebenenfalls mit Unterstützung durch eine Wendefunktion in der Ablagevorrichtung.

Der Aufbau einer Roboteranlage für den Griff-in-die-Kiste mit einem modellfreien Ansatz sieht weitgehend ähnlich aus. Aufgrund des fehlenden Werkstückmodells werden hier jedoch keine Werkstücklagen bestimmt. Stattdessen wird in den Sensordaten, abhängig vom verwendeten Greifer, nach greifbaren Bereichen gesucht. Dies können beispielsweise glatte Flächen für einen Sauggreifer oder geeignete Geometrien für einen Klemmgreifer sein. Da somit nicht bekannt ist, wie ein Werkstück gegriffen wurde, ist hier ein Ablegen in einer definierten Lage nicht ohne Weiteres möglich.

Bisher haben sich die meisten Forschungsarbeiten im Bereich des Griff-in-die-Kiste auf die Bestimmung der Werkstücklagen beziehungsweise der greifbaren Bereiche konzentriert. Da in industriellen Anwendungen eine hohe Verfügbarkeit von zentraler Bedeutung ist, sollten jedoch jegliche Probleme, die beim Greifen auftreten können, vorab geprüft und vermieden werden. Dies betrifft insbesondere Fehler, die zu einem Stillstand der Anlage führen können, wie zum Beispiel Kollisionen des Roboters oder des Greifers mit dem Ladungsträger oder mit anderen

Werkstücken. Daher ist die Planung eines geeigneten Greifvorgangs in der Praxis von hoher Bedeutung, um eine robuste Anwendung zu realisieren. Die geringe Anzahl an Roboteranlagen für den Griff-in-die-Kiste in der industriellen Praxis deutet darauf hin, dass hier noch ein hoher Forschungsbedarf besteht, weshalb sich diese Arbeit mit der Bestimmung von geeigneten Greiflösungen für die roboterbasierte Werkstückvereinzelung befasst.

1.2 Stand der Forschung

Obwohl bereits seit über 35 Jahren an der Vereinzelung von ungeordnet gelagerten Werkstücken gearbeitet wird, ist das Thema nach wie vor Gegenstand der aktuellen Forschung. Zur Bestimmung von geeigneten Griffen basierend auf Sensordaten der jeweiligen Situation existieren unterschiedliche Verfahren. Dieser Abschnitt gibt eine Übersicht, über den Stand der Forschung bei modellfreien und modellbasierten Ansätzen. Dabei liegt der Fokus insbesondere auf der Planung eines kollisionsfreien Greifvorgangs.

1.2.1 Modellfreie Ansätze

Modellfreie Ansätze kommen ohne ein Modell des Werkstücks und somit ohne eine Bestimmung der Werkstücklage aus und lassen sich grob in folgende Kategorien einordnen.

1.2.1.1 Geometrische Ansätze

Geometrische Ansätze suchen in den Sensordaten nach festgelegten, geometrischen Merkmalen, die einen Griff zulassen. Eine Variante solcher geometrischer Merkmale sind ebene Flächen. Diesen Ansatz verfolgen beispielsweise Kaiser et al. (2009) zur automatischen Entnahme von quaderförmigen Objekten aus Containern (siehe auch Kaiser (2010) und Tauro et al. (2010)). Berger et al. (2000) verwenden die Suche nach Flächen, um Werkstücke mittels eines Sauggreifers aus Kisten zu greifen und zunächst auf einer Fläche abzulegen. Dabei wird jeweils die größte und am höchsten gelegene Fläche für den aktuellen Griff ausgewählt. In einem zweiten Schritt wird mittels einer 2D-Kamera die Objektlage bestimmt. Trobina et al. (1994) detektieren ebene Flächen in aus zwei Richtungen aufgenommenen Tiefenbildern und segmentieren diese zu einzelnen konvexen Objekten. Durch die Suche nach gegenüberliegenden, weitgehend parallelen Flächen wird der Griff für einen Klemmgreifer bestimmt (vgl. auch Trobina et al. (1995), Trobina (1995) und Ade et al. (1995)). Dessimoz et al. (1984) verwenden Optimalfilter, um lokale Strukturen in 2D-Bildern zu finden, die sich zum Greifen mit einem Saug- oder Klemmgreifer

eignen. Fan et al. (2018) verwenden einen iterativen Ansatz, um die Geometrie bekannter Greiferfinger bestmöglich in die 3D-Sensordaten der Szene einzupassen. Ein ähnliches Konzept zur Bestimmung von greifbaren Bereichen verwenden Domae et al. (2014). Hier werden jedoch zweidimensionale Masken der Greiferfinger verwendet, um in einer Tiefenkarte nach möglichen Griffen zu suchen. Weiterhin gibt es Konzepte für konfigurierbare Greifer, wie zum Beispiel für Sauggreifer mit relativ zueinander verschiebbaren Saugnäpfen (You et al. 2018). Somit existiert eine Vielzahl an unterschiedlichen Ansätzen, die jeweils auf bestimmte Anwendungsfälle und Greifer zugeschnitten sind.

1.2.1.2 Ansätze auf Basis von Lernverfahren

Eine Alternative zu den geometrischen Ansätzen ist das Lernen möglicher Griffe auf Basis von realen Greifversuchen oder Simulationen. Pauli (1998) erreicht dies durch ein manuelles Bewegen des Greifers zum zu greifenden Werkstück. Dort werden einige Referenzbilder, in denen das zu greifende Werkstück und die Greiferfinger sichtbar sind, aufgenommen, um neuronale Netze zu trainieren. Während des Entnahmeprozesses bewegt sich der Greifer, nach jeweiliger Aufnahme eines Kamerabildes, inkrementell zu einem stabilen Griff. Alternativ können mögliche Griffe auch ohne Roboter, beispielsweise manuell in RGB-D-Daten gelabelt werden (Jiang et al. 2011) und neuronale Netze auf diesen Daten trainiert werden (Jiang et al. 2011; Redmon et al. 2015). Dieser Ansatz ist auf bestimmte Greifer und Greifbewegungen beschränkt. Ein weiterer Ansatz stellt das automatische Training durch den Roboter dar. Hierbei führt ein Roboter im Vorfeld selbstständig eine Vielzahl an Greifversuchen durch und lernt, welche Griffe bei welchen Kameradaten erfolgreich sind. Ein inkrementelles Herantasten an das Werkstück ist dabei meist nicht notwendig. Insbesondere in den letzten Jahren kommen dabei sogenannte *Deep Convolutional Neural Networks* zum Einsatz (Pinto et al. 2016; Finn et al. 2016; Levine et al. 2018; Berscheid et al. 2019; Shao et al. 2019). Mahler et al. (2017) und Mahler et al. (2018) präsentieren weitere Ansätze zum Lernen geeigneter Griffe auf Basis von Punktwolken, hier allerdings durch das Trainieren von Deep Convolutional Neural Networks auf Basis von automatisch erzeugten Griffen mit Zweifinger- bzw. Sauggreifern an virtuellen Werkstückmodellen. Auch Simulationen können für das Training dieser neuronalen Netze zum Einsatz kommen (Johns et al. 2016). Zusammengefasst gibt es eine große Anzahl verschiedener Ansätze auf Basis von maschinellen Lernverfahren, die auf unterschiedliche Weise trainiert werden. Durch den Verzicht auf ein Modell des Werkstücks ist jedoch, wie auch bei den geometrischen Ansätzen, die Lage des Werkstücks im Greifer nicht bekannt, so dass ein definiertes Ablegen nicht ohne Weiteres möglich ist.

1.2.2 Modellbasierte Ansätze

Bei modellbasierten Ansätzen wird im ersten Schritt die Lage eines oder mehrerer Objekte bestimmt und im zweiten Schritt basierend auf diesen Objektlagen ein Griff berechnet. Die meisten modellbasierten Arbeiten fokussieren sich auf die Bestimmung der Objektlage (Rahardja et al. 1996; Ohba et al. 1996; Takenouchi et al. 1998; Hema et al. 2006; Boehnke 2007; Oh et al. 2010; Park et al. 2010; Breitenreicher et al. 2010; Ledermann 2012; Palzkill et al. 2012; Effenberger et al. 2013; Kuo et al. 2014; Kleeberger et al. 2020). Zur effizienten Planung von Griffen bei ungeordneten Werkstücken gibt es hingegen nur wenig Ansätze. Die existierenden Ansätze lassen sich grob in drei Kategorien einteilen, die im Folgenden beschrieben werden. Als Unterscheidungsmerkmal wird dafür zunächst der Begriff *Greifpose* eingeführt und in Abschnitt 3.1.1 näher beschrieben.

> *Eine Greifpose definiert eine von mehreren Möglichkeiten, ein Werkstück mit einem vorgegebenen Greifer zu greifen und umfasst dazu ein relativ zum Werkstück definiertes Koordinatensystem sowie alle weiteren Informationen, die für den Greifprozess oder die Auswahl von Greifposen notwendig sind.*

1.2.2.1 Ansätze ohne vorab definierte Greifposen

Modellbasierte Ansätze ohne vorab definierte Greifposen kommen in der Literatur nur selten vor. Dennoch seien an dieser Stelle zwei Ansätze erwähnt. Bei dem Ansatz von Yanagihara et al. (1991) wird davon ausgegangen, dass die Lage aller Werkstücke vor der Berechnung eines Griffs durch ein geeignetes Verfahren zur Objektlageerkennung bestimmt werden kann. Basierend darauf wird für jedes Werkstück ein stabiler Griff bestimmt, der Kollisionen mit dem Untergrund und anderen erkannten Werkstücken vermeidet. Im Anschluss wird ein möglichst isoliert liegendes Werkstück ausgewählt. Al-Hujazi et al. (1990) segmentieren die Punktwolke bereits für die Objektlageerkennung in einzelne Oberflächen und berechnen mögliche Greifposen für einen Sauggreifer auf den jeweils höchsten Oberflächen eines Objekts. Die Auswahl der zu verwendenden Greifpose erfolgt im Anschluss auf Basis der Eigenschaften der Oberfläche sowie ihrer Lage relativ zum Schwerpunkt des Objekts.

1.2.2.2 Ansätze auf Basis vorab manuell definierter Greifposen

Wenn die Lage der Werkstücke im Entnahmeprozess bestimmt wird, ergibt sich die Möglichkeit, die zu verwendenden Greifposen am Werkstück vorab manuell festzulegen. Dabei definiert ein Anlagenbediener oder Einrichter, wo die Werkstücke gegriffen werden können oder gegriffen

werden sollen. Diese Greifposen können im laufenden Prozess insbesondere auf Kollisionen geprüft oder sogar modifiziert werden. Es existieren allerdings auch einfache Ansätze, bei denen auf diese Prüfung verzichtet wird, wodurch diese Ansätze ein Kollisionsrisiko in Kauf nehmen. So schlagen Papazov et al. (2012) vor, das höchste erkannte Werkstück mit einem möglichst vertikalen Griff zu greifen und Kollisionen allein mit dem Roboter zu detektieren. Auch wenn vorgeschlagen wird, Griffe, bei denen der Greifer stark geneigt ist, auszuschließen, ist dieser Ansatz nicht robust für den Einsatz in industriellen Vereinzelungssystemen. Ohne eine deutliche Einschränkung der Geschwindigkeit des Roboters besteht die Gefahr, dass Kollisionen nicht rechtzeitig erkannt werden können, um den Roboter kontrolliert zu stoppen. Chen et al. (2018) vernachlässigen ebenfalls die Prüfung auf Kollisionen und gehen davon aus, dass sich das höchstgelegene Werkstück kollisionsfrei greifen lässt. Horn et al. (1983) verwenden ebenfalls nur einfache Heuristiken, wie den Grad der Überdeckung des Werkstücks und seine Orientierung.

Eine naheliegende Möglichkeit, eine kollisionsfreie Anwendung mit manuell definierten Greifposen zu realisieren, ist hingegen, alle vorab festgelegten Greifposen im laufenden Prozess zu überprüfen und den *besten* Griff auszuwählen oder die Suche zu stoppen, sobald ein geeigneter Griff gefunden wurde (Schraft et al. 2003; Spenrath et al. 2012; Buchholz et al. 2010; Buchholz et al. 2013). Diese erschöpfende Suche ist allerdings langsam oder benötigt zusätzliche, aufwändig zu konfigurierende Strategien, um die Suche zu beschleunigen. Teilweise wird die Berechnung beschleunigt, indem die Punktwolke nur für eingeschränkte Kollisionsprüfungen verwendet wird. So verwenden beispielsweise Martinez et al. (2015) lediglich eine Maske der vorderen Spitzen der Greiferfinger, um zu prüfen, ob die Positionen frei liegen, an denen die Fingerspitzen zugreifen. Auf eine Kollisionsprüfung des gesamten Greifers mit der Punktwolke wird hingegen verzichtet. Andere Lösungen prüfen während des Prozesses lediglich auf Kollisionen mit *Bounding Boxes* von anderen erkannten Werkstücken (Takenouchi et al. 1998), um die Suche zu beschleunigen und sind daher nicht robust gegenüber nicht erkannten Werkstücken oder anderen Hindernissen.

1.2.2.3 Ansätze auf Basis automatisch generierter Greifposen

Auch wenn bei den im vorherigen Absatz genannten Ansätzen davon ausgegangen wird, dass die Greifposen durch einen Einrichter manuell definiert werden, so besteht ebenso die Möglichkeit, diesen Schritt zu automatisieren. Für die Bestimmung von möglichen Greifposen an Werkstücken stehen diverse Bibliotheken oder Ansätze zur Verfügung (Miller et al. 2000; Miller et al. 2004; Diankov et al. 2008; Jørgensen et al. 2008; Dupuis et al. 2008; Khalid et al. 2021; Kleeberger et al. 2021), teilweise auch auf Basis von neuronalen Netzen (Mahler et al. 2016). Weiterhin existieren Ansätze, die auf bestimmte Werkstückgeometrien beschränkt sind und deren Greifposen dadurch trivial festzulegen sind. Als Beispiel sei der Ansatz von Roy et al.

(2016) genannt, der sich auf zylindrische Werkstücke beschränkt. Die Kollisionsprüfung dieses Ansatzes berücksichtigt lediglich Kollisionen mit dem Ladungsträger, wobei der Greifer durch einen Zylinder approximierbar sein muss.

Andere Ansätze für die Greifplanung beim Griff-in-die-Kiste basieren direkt auf automatisch generierten Greifposen. Hier sei der Ansatz von Wang et al. (1994) zur Entnahme von gebogenen Rohren genannt. Dabei werden die Greifposen direkt beim Einlernen eines Werkstücks generiert. Das Einlernen erfolgt mittels optischer Erfassung der Werkstücke in allen stabilen Lagen, wobei die Greifpose jeweils am höchsten Punkt einer gegebenen Objektlage generiert wird. Weitere Ansätze prüfen den Anfahrvektor bereits bei der Generierung der Greifposen auf Kollisionen mit dem Werkstück und während des Entnahmeprozesses auf Erreichbarkeit und auf Kollisionen mit der Punktwolke (Holz et al. 2014; Nieuwenhuisen et al. 2012; Nieuwenhuisen et al. 2013), was allerdings in den veröffentlichten Ergebnissen meist mehrere Sekunden dauert.

1.2.3 Zusammenfassung und Fazit

In der Produktion müssen Werkstücke häufig lagerichtig abgelegt werden, um sie weiter verarbeiten zu können. Dies ist mit modellfreien Ansätzen, bei denen die Lage des Werkstücks unbekannt bleibt, nicht möglich, so dass diese kaum für einen Einsatz in Produktionsumgebungen geeignet sind. Mögliche Einsatzbereiche für diese Ansätze sind eher im Warenlager zu finden, wo eine Vielzahl unterschiedlicher Produkte gegriffen werden muss und eine vollständig definierte Ablage meist nicht notwendig ist. Zudem ist es bei modellfreien Ansätzen im Allgemeinen nicht möglich, Stellen am Werkstück zu definieren, an denen das Greifen nicht gewünscht ist. Da diese Verfahren lediglich nach greifbaren Bereichen in den Sensordaten suchen, kann es außerdem vorkommen, dass fälschlicherweise zwei Werkstücke auf einmal gegriffen werden. Aktuelle modellfreie Ansätze berücksichtigen zudem meist keine Kollisionen, sondern verwenden häufig Schalen mit nach außen geneigten Wänden anstatt realistischer industrieller Ladungsträger mit paarweise parallelen Seitenwänden. Die geometrischen Ansätze sind, bedingt durch das Funktionsprinzip, zusätzlich auf bestimmte Werkstück- und Greifergeometrien beschränkt.

In vielen industriellen Anwendungen wird eine manuelle Definition der erlaubten Greifposen am Werkstück bevorzugt, um einen stabilen Griff sicherzustellen und um Schäden an empfindlichen Stellen des Werkstücks zu vermeiden. Sofern die Greifposen vorab festgelegt werden, können automatisch generierte Greifposen den Einrichtvorgang jedoch unterstützen und somit zu einer geringeren Inbetriebnahmezeit führen. Insbesondere sollte es dabei eine Möglichkeit geben, Greifposen, die nicht sicher oder aus anderen Gründen unerwünscht sind, auszuschließen. Bei

Ansätzen, in denen die Greifpose erst im laufenden Betrieb bestimmt wird, gibt es diese Möglichkeit nicht. Je nach Schwerpunkt, Masse, Oberfläche und Geometrie der Werkstücke kann dies somit dazu führen, dass ein Werkstück nicht sicher gegriffen wird und bei der Entnahme herunter fällt.

Alle Ansätze, die auf bestimmte Werkstückgeometrien oder Werkstückeigenschaften angewiesen sind, lassen sich nur in wenigen Szenarien und daher nicht universell einsetzen. Die meisten existierenden modellbasierten Ansätze für beliebige Werkstücke erfordern jedoch eine hohe Rechenzeit, was die Möglichkeiten eines Einsatzes im industriellen Umfeld stark einschränkt. Zudem mangelt es vielen Ansätzen an der notwendigen Flexibilität. Einige Anwendungen erfordern beispielsweise eine oder mehrere zusätzliche Bewegungsachsen im Greifer, um Werkstücke in allen Lagen greifen zu können. Dies wird nur von den wenigsten Ansätzen unmittelbar unterstützt. Ansätze, bei denen keine Kollisionsprüfung mit der Punktwolke stattfindet, sind für industrielle Vereinzelungsaufgaben im Allgemeinen ungeeignet, da Kollisionen mit nicht erkannten Werkstücken, mit einer verbogenen Kiste oder mit Fremdkörpern nicht erkannt werden können. Dies führt zu einer erhöhten Kollisionswahrscheinlichkeit und senkt dadurch die Verfügbarkeit der Anlage.

Bei Betrachtung der Literatur fällt auf, dass viele der modellbasierten Ansätze einfache Heuristiken verwenden, um die Suche nach einer geeigneten Greiflösung zu beschleunigen. Einige Ansätze berücksichtigen dazu den Abstand eines Werkstücks zu anderen Werkstücken oder die Größe der Fläche, an der gegriffen werden soll. Am häufigsten wird jedoch davon ausgegangen, dass höher im Ladungsträger liegende Werkstücke besser zu greifen sind. Daher werden diese höher liegenden Werkstücke beziehungsweise höher liegenden Greifposen in vielen Ansätzen bevorzugt verwendet.

1.3 Zielsetzung und Abgrenzung

Ziel dieser Arbeit ist die Entwicklung und Untersuchung eines Verfahrens zur Greifplanung für die Entnahme ungeordnet gelagerter Werkstücke aus Kisten oder anderen Ladungsträgern. Diese Greifplanung soll es einem Roboter ermöglichen, Werkstücke, die ungeordnet in einem Ladungsträger liegen, zu entnehmen und einer Maschine zuzuführen oder anderweitig definiert abzulegen. Dabei wird davon ausgegangen, dass der Greifplanung die Punktwolke, die Lage der einzelnen Werkstücke und die Lage des Ladungsträgers zur Verfügung stehen. Die Bestimmung der Lagen der Werkstücke und des Ladungsträgers kann vorab durch ein Verfahren zur Objektlageerkennung erfolgen und ist nicht Bestandteil dieser Arbeit. Für die experimentellen Untersuchungen wird stattdessen auf ein Verfahren aus dem Stand der Technik zurückgegriffen. Weiterhin wird

davon ausgegangen, dass die möglichen Greifposen am Werkstück vorab bekannt sind. Diese können entweder manuell durch einen Inbetriebnehmer definiert oder automatisch generiert werden. Die automatische Bestimmung von Greifposen an einem gegebenen Werkstück ist jedoch nicht Bestandteil dieser Arbeit. Für industrielle Anwendungen wird, wie in Abschnitt 1.2.3 erwähnt, meist ohnehin gewünscht, die Greifposen manuell zu definieren. Falls dennoch eine automatische Bestimmung von möglichen Greifposen an einem Werkstück notwendig ist, so kann dies im Vorfeld mittels existierender Ansätze erfolgen (siehe Abschnitt 1.2). Danach lässt sich auch in solchen Anwendungen, die hier entwickelte Greifplanung verwenden. Es wird davon ausgegangen, dass alle definierten Greifposen ein anschließendes Ablegen in der Ablagevorrichtung zulassen. Dies lässt sich sicherstellen, indem alle Greifposen vorab auf die Möglichkeit eines Ablegens geprüft werden, beziehungsweise eine entsprechende Ablageplanung für alle Greifposen vorab durchgeführt wird. Da eine Greifpose die Lage des TCP zum Werkstück beschreibt, kann somit auf eine aufwändige Prüfung vor jeder Entnahme eines Werkstücks verzichtet werden. Das Ablegen der Werkstücke soll in dieser Arbeit nicht näher betrachtet werden. Es sei jedoch erwähnt, dass das in dieser Arbeit entwickelte Verfahren mit geringfügigen Änderungen auch zur Planung einer kollisionsfreien Bewegungsbahn für die Ablage verwendet werden kann und in diesem Zusammenhang bereits industriell eingesetzt wird. Das Ergebnis der Greifplanung muss geeignet sein, einen Roboter so zu steuern, dass der Greifer kollisionsfrei zum Werkstück bewegt und das Werkstück gegriffen und kollisionsfrei entnommen werden kann. Weiterhin muss die Greifplanung alle Informationen über die gewählte Greifpose bereitstellen, die für den Greifvorgang notwendig sind.

Durch eine Vielzahl an Kundenanfragen hat sich gezeigt, dass die folgenden Anforderungen für eine universell einsetzbare, industrielle Vereinzelungs- und Zuführlösung besonders wichtig sind. Die Greifplanung soll daher in der Lage sein, diese Anforderungen zu erfüllen.

Anforderung: Geringe mittlere Taktzeit
Wie in Abschnitt 1.1 gezeigt wurde, ist die Taktzeit eine der wichtigsten Anforderungen für eine industrielle Vereinzelungslösung. Für die Greifplanung bedeutet dies, dass die Berechnung in möglichst kurzer Zeit abgeschlossen sein sollte. Dies ist besonders wichtig, wenn die notwendige Sensorik am Roboter montiert ist und die Szene erst direkt vor dem Greifen erfasst wird, da in diesem Fall die Szenenerfassung nicht hauptzeitparallel erfolgen kann. In der Praxis zeigt sich, dass die mittlere Rechenzeit der Greifplanung in vielen Anwendungen unter einer Sekunde liegen sollte, was somit für diese Arbeit als zu erreichendes Ziel gelten soll.

Anforderung: Hohe Verfügbarkeit
Der Verein Deutscher Ingenieure (VDI) (VDI 3423) definiert die technische Verfügbarkeit einer Anlage als „den prozentualen Anteil der Belegungszeit (...), für den die Maschine/Anlage ohne technischen Mangel der Produktion zur Verfügung steht“. Sie ist eines der wichtigsten Merkmale

einer industriellen Roboteranlage, da Störungen an der Anlage zu Produktionsausfällen und damit zu hohen Kosten führen. Dies bedeutet, dass störungsbedingte Stillstandszeiten der Anlage gering gehalten werden müssen. Somit muss sichergestellt werden, dass die geplanten Griffe nicht zu Störungen führen. Insbesondere müssen Kollisionen des Roboters oder des Greifers ausgeschlossen werden. Weiterhin sollten auch mögliche Probleme für die Roboterkinematik, wodurch einzelne Greifposen nicht erreichbar sein können, vorab erkannt werden. Eine industriell einsetzbare Greifplanung muss somit in der Lage sein, verschiedene potenzielle Probleme vorab zu prüfen und zu vermeiden. Insbesondere sollte das Verfahren so flexibel sein, dass weitere Prüfungen leicht integriert werden können.

Anforderung: Hoher Entleerungsgrad
Abhängig von den zu vereinzelnden Werkstücken und dem verwendeten Greifer kann beim Griff-in-die-Kiste nur selten eine vollständige Entleerung garantiert werden. Um dennoch einen möglichst hohen Entleerungsgrad zu erreichen, sollte die Greifplanung in der Lage sein, in möglichst vielen Situationen eine geeignete Lösung zur Entnahme eines Werkstücks zu finden.

Anforderung: Hohe Flexibilität
Aufgrund der hohen Varianz an Werkstücken und Ladungsträgern in der Industrie müssen für eine roboterbasierte Vereinzelung meist speziell für den Anwendungsfall konstruierte Greifer sowie ausgewählte Sensoren und Verfahren zur Objektlageerkennung zum Einsatz kommen. Aus diesem Grund sollten Verfahren für die industrielle Werkstückvereinzelung flexibel sein, um möglichst breit eingesetzt werden zu können. Insbesondere soll die in dieser Arbeit entwickelte Greifplanung daher folgenden Ansprüchen genügen:

- Die Greifplanung soll für eine möglichst hohe Anzahl unterschiedlicher Werkstücke eingesetzt werden können. Der Fokus liegt dabei insbesondere auf Werkstücken, die üblicherweise in Kisten gelagert werden. Dabei handelt es sich meistens um formstabile und unempfindliche Werkstücke, wie zum Beispiel Guss- oder Schmiedeteile. Da biegeschlaffe Werkstücke nur einen kleinen Teil der in Kisten gelagerten Werkstücke darstellen, nach der Vereinzelung meist von Hand weiterverarbeitet werden müssen und für die Objektlageerkennung ohnehin eine große Herausforderung darstellen, beschränkt sich diese Arbeit auf formstabile Werkstücke.

- Die Greifplanung soll für eine möglichst hohe Anzahl an unterschiedlichen Ladungsträgern eingesetzt werden können. Dies beinhaltet insbesondere Kleinladungsträger (KLT), wie zum Beispiel Kunststoffkisten, Großladungsträger, wie zum Beispiel Metallkisten oder Gitterboxen, jedoch auch Paletten (mit oder ohne Aufsatzrahmen). Abbildung 1.3 zeigt einige Beispiele häufig vorkommender Ladungsträger. Nicht formstabile Ladungsträger, wie zum Beispiel Säcke, finden in der industriellen Praxis hingegen kaum Anwendung und

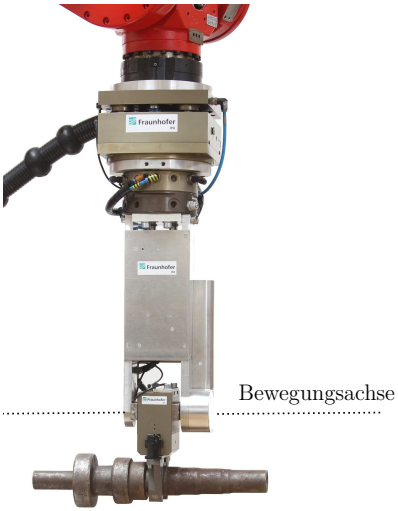
(a) (b) (c) (d)

Abbildung 1.3: Gängige Ladungsträger: Europalette mit Aufsatzrahmen mit den Abmessungen 1200 mm × 800 mm × 744 mm (a), Gitterbox mit den Abmessungen 1200 mm × 800 mm × 1000 mm (b), Metallkiste mit den Abmessungen 1200 mm × 800 mm × 600 mm (c), Kleinladungsträger mit den Abmessungen 600 mm × 400 mm × 240 mm (d)

sollen in dieser Arbeit daher nicht betrachtet werden. Teilweise werden die Werkstücke innerhalb des Ladungsträgers zusätzlich in Folie verpackt, um sie vor Umwelteinflüssen zu schützen. Solche Folien können sowohl eine optische Erfassung der Werkstücke erschweren, als auch einen störenden Einfluss auf die Entnahme der Werkstücke haben. In dieser Arbeit wird davon ausgegangen, dass keine Folie verwendet wird oder diese vorab vollständig an den Kistenrand angelegt wurde und daher ignoriert werden kann.

- Da Greifer für den Griff-in-die-Kiste meist speziell auf die jeweils zu vereinzelnden Werkstücke zugeschnitten sein müssen, muss die Greifplanung mit einer großen Anzahl unterschiedlicher Greifer und Greifprinzipien verwendet werden können. Teilweise kommen hier auch Kombinationen aus verschiedenen Greifprinzipien, wie Saug- und Klemmgreifern, oder Systeme für einen automatischen Greiferwechsel zum Einsatz, um die Anzahl der möglichen Greifposen an einem Werkstück zu erhöhen. Die Greifplanung sollte diese Möglichkeiten unterstützen. In der Praxis werden für den Griff-in-die-Kiste teilweise auch zusätzliche, meist rotatorische, Bewegungsachsen in die Greifer integriert, um eine bessere Zugänglichkeit in der Kiste zu erreichen (siehe beispielsweise Abbildung 1.4). Obwohl die in den Greifer integrierte Bewegungsachse als zusätzliche Achse in der gesamten Roboterkinematik betrachtet werden könnte, wird diese in der Praxis meist separat durch die Anlagensteuerung angesteuert. Die Verwendung einer beliebigen Anzahl solcher zusätzlicher rotatorischer und translatorischer Bewegungsachsen im Greifer soll durch die Greifplanung ebenfalls unterstützt werden.

- Da das verwendete Robotermodell von mehreren Randbedingungen, wie dem Werkstückgewicht und dem Kundenwunsch, abhängt, muss eine flexibel einsetzbare Greifplanung mit allen gängigen Industrierobotern verwendet werden können. Eine universelle Möglichkeit, die Roboterbewegung zu beschreiben, besteht in zwei Listen von Posen (eine für die



(a) Greifer mit einer Bewegungsachse (Quelle des Originalfotos: Fraunhofer IPA)

(b) Greifer mit zwei orthogonal zueinander angeordneten Bewegungsachsen (Quelle: Liebherr 2013)

Abbildung 1.4: Beispiele für Greifer mit zusätzlichen Bewegungsachsen

Anfahrbahn und eine für die Entnahmebahn), die der Roboter der Reihe nach linear abfährt.

- Die Greifplanung soll mit beliebigen Algorithmen zur Objektlageerkennung eingesetzt werden können, die eine Liste der Lagen aller erkannten Werkstücke zur Verfügung stellen. Zusätzlich wird davon ausgegangen, dass für jede erkannte Werkstücklage ein Konfidenzwert $R_l \in [0, 1]$ zur Verfügung steht, der die Wahrscheinlichkeit der korrekten Lagebestimmung angibt.

Zusammengefasst soll in dieser Arbeit folgende Frage beantwortet werden:

> *Wie können innerhalb kurzer Rechenzeit ein greifbares Werkstück und eine entsprechende Greifpose an diesem ausgewählt, sowie ein geeigneter Greif- und Entnahmevorgang geplant werden, um es einem Roboter zu ermöglichen, erkannte Werkstücke mit gegebener, starrer Geometrie und vordefinierten Greifposen kollisionsfrei aus einem ungeordnet gefüllten Ladungsträger zu entnehmen?*

1.4 Lösungsansatz und Gliederung der Arbeit

In dieser Arbeit wird eine effiziente Greifplanung für die roboterbasierte Werkstückvereinzelung entwickelt, die insbesondere für den Griff-in-die-Kiste eingesetzt werden kann.

In *Kapitel 2* werden zunächst die wichtigsten Grundkonzepte und Konventionen eingeführt, die in der Arbeit verwendet werden. Im Anschluss erfolgt eine Einführung in das Thema der roboterbasierten Werkstückvereinzelung und die dafür notwendigen Komponenten, wie zum Beispiel Sensoren, Greifer und Roboter. Weiterhin werden in diesem Kapitel die relevanten Grundlagen im Bereich der Suchverfahren und der künstlichen neuronalen Netze vorgestellt, die für das Verständnis dieser Arbeit benötigt werden.

In *Kapitel 3* wird das heuristische Suchverfahren für die Planung der Griffe entwickelt und dessen Aufbau sowie die Funktionsweise vorgestellt. Um geeignete Greifvorgänge planen zu können, muss eine Vielzahl an beteiligten Komponenten, wie zum Beispiel Werkstücke, Greifer und Ladungsträger, berücksichtigt werden. Daher werden diese vorab in einer Wissensbasis modelliert. Basierend auf dieser Wissensbasis wird für jeden Entnahmevorgang ein Suchbaum erstellt. Der Suchbaum beginnt mit den einzelnen erkannten Werkstücken als Kindknoten der Wurzel und beinhaltet alle beteiligten Komponenten. Nach Abschluss der Suche kann die gefundene Greiflösung aus dem Suchbaum ausgelesen werden. Um später möglichst schnell eine geeignete Greiflösung zu finden, kommt die Bestensuche als heuristisches Suchverfahren zum Einsatz. In diesem Kapitel wird die Effizienz des Ansatzes zunächst nicht betrachtet, sondern angenommen, dass eine geeignete Heuristikfunktion existiert. Während der Suche werden mehrere Prüfungen, wie Kollisionstests und die Überprüfung der Roboterkinematik, durchgeführt. Zudem wird bei der iterativen Erstellung des Suchbaums die Bewegungsbahn für den Roboter generiert. Durch die Berücksichtigung aller relevanten Komponenten im Suchbaum soll gewährleistet werden, dass sich die berechnete Greiflösung problemlos durch den Roboter ausführen lässt. Zum Abschluss des Kapitels wird das entwickelte Verfahren auf Basis der gesetzten Ziele bewertet und die Komplexität des Suchbaums abgeschätzt. Darüber hinaus wird das Verfahren anhand von simulierten Daten validiert und untersucht.

In *Kapitel 4* wird schließlich eine geeignete Heuristikfunktion entwickelt, um eine möglichst geringe Rechenzeit für das in Kapitel 3 vorgestellte Verfahren zu erreichen. In einer solchen Heuristikfunktion können diverse Einflussfaktoren, wie die Höhe von Werkstücken, deren Abstand vom Kistenrand oder die Orientierung von Greifposen, berücksichtigt werden. Ziel dieser Heuristikfunktion ist es, abzuschätzen, welche Knoten im Suchbaum schnell zu einer geeigneten Greiflösung führen und daher bevorzugt expandiert werden sollten. Eine Alternative zur manuellen Definition geeigneter Einflussfaktoren für die Heuristikfunktion lässt sich mittels maschinellem Lernen realisieren. Dafür kommen in dieser Arbeit künstliche neuronale Netze zum Einsatz,

die selbstständig lernen, welche Einflussfaktoren die Wahrscheinlichkeit erhöhen, eine geeignete Greiflösung zu finden. Ein solcher Ansatz wird in Abschnitt 4.4 vorgestellt. Zum Abschluss des Kapitels werden die vorgestellten Einflussfaktoren und der Effekt der neuronalen Netze anhand von Simulationsdaten untersucht. Dabei kommen verschiedene Optimierungsverfahren zum Einsatz, um die jeweils besten Parameter für die Heuristikfunktion zu bestimmen.

In *Kapitel 5* werden schließlich die vorgestellten Verfahren anhand zweier realer Versuchsaufbauten analysiert und dadurch die simulierten Ergebnisse bestätigt. Zudem wird untersucht, inwieweit sich auf simulierten Daten trainierte neuronale Netze auf reale Roboteranlagen übertragen lassen. Das in dieser Arbeit vorgestellte Verfahren befindet sich bereits mehrfach im industriellen Einsatz. Daher werden zum Abschluss die Ergebnisse dieser Arbeit mit den Daten aus einer dieser industriellen Realisierungen validiert. Dadurch wird gezeigt, dass die in dieser Arbeit vorgestellte Greifplanung nicht nur in Versuchsaufbauten funktioniert, sondern auch für die industrielle Praxis geeignet ist.

Die Arbeit schließt in Kapitel 6 mit einer Zusammenfassung sowie einem Ausblick.

2 Grundlagen

In diesem Kapitel werden die benötigten Grundlagen der in der Arbeit verwendeten Themengebiete vorgestellt. Dabei werden zunächst einige mathematische Grundlagen vorgestellt und die im weiteren Verlauf verwendeten Notationen eingeführt. Im Anschluss wird auf die wichtigsten Aspekte von Anlagen eingegangen, in denen Industrieroboter für die Vereinzelung ungeordnet gelagerter Werkstücke eingesetzt werden. Dabei wird näher auf die darin verwendeten wesentlichen Komponenten, wie Industrieroboter, Greifer und Sensoren eingegangen. Abschnitt 2.2 führt in die Thematik der Suchverfahren ein und widmet sich insbesondere heuristischen Suchverfahren mittels Suchbäumen. Zum Abschluss des Kapitels beschreibt Abschnitt 2.3 die Grundlagen der *künstlichen neuronalen Netze* und zeigt, wie diese in heuristischen Suchverfahren verwendet werden können.

2.1 Mathematische Grundlagen und Notationen

2.1.1 Koordinatensysteme und homogene Transformationen

In der Robotik ist es notwendig, die *Lage* von Objekten im dreidimensionalen Raum zu beschreiben. Eine solche Lage (auch *Pose* genannt) besitzt einen Translationsanteil, der durch einen Ortsvektor

$$\boldsymbol{t} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.1}$$

beschrieben werden kann, sowie einen Rotationsanteil, der durch eine Rotationsmatrix

$$\boldsymbol{R} = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{pmatrix} \tag{2.2}$$

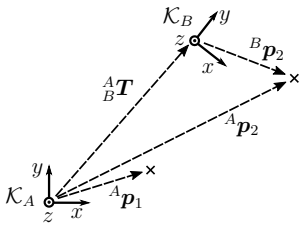


Abbildung 2.1: Koordinatensysteme und Transformationen

ausgedrückt werden kann. Eine *homogene Transformation* aus der speziellen euklidischen Gruppe SE(3) bietet die Möglichkeit, Translation und Rotation durch eine einzelne Matrix auszudrücken:

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & x \\ r_{2,1} & r_{2,2} & r_{2,3} & y \\ r_{3,1} & r_{3,2} & r_{3,3} & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

Eine nähere Erläuterung zu dieser Einordnung findet sich in Blanco (2010). Zur Beschreibung einer Lage eines Objektes wird stets ein Referenzkoordinatensystem benötigt, auf das sich die beschreibende homogene Transformation bezieht. In dieser Arbeit werden verschiedene Koordinatensysteme verwendet. Diese werden durch ein kalligraphisches $\mathcal{K}$ mit einem entsprechenden Index bezeichnet, der beschreibt, was das jeweilige Koordinatensystem repräsentiert. Beispielsweise steht $\mathcal{K}_W$ für das Koordinatensystem eines Werkstücks. Nach Craig (2005) existieren mehrere Interpretationen einer homogenen Transformationsmatrix:

- zur Beschreibung eines Koordinatensystems. Beispielsweise beschreibt die Transformation ${}^{A}_{B}\boldsymbol{T}$ in Abbildung 2.1 das Koordinatensystem $\mathcal{K}_B$ relativ zum Koordinatensystem $\mathcal{K}_A$.

- zur Umrechnung eines festen Punkts (bzw. Koordinatensystems) in ein anderes Koordinatensystem. Dies wird auch als Basiswechsel bezeichnet. So lässt sich beispielsweise der im Koordinatensystem $\mathcal{K}_B$ definierte Punkt ${}^{B}\boldsymbol{p_2}$ folgendermaßen in das Koordinatensystem $\mathcal{K}_A$ umrechnen: ${}^{A}\boldsymbol{p}_2 = {}^{A}_{B}\boldsymbol{T} \cdot {}^{B}\boldsymbol{p}_2$.

- zur Transformation eines Punkts (bzw. Koordinatensystems) auf Basis der Transformation zwischen zwei Koordinatensystemen. Sei beispielsweise ${}^{A}\boldsymbol{p}_1 = {}^{B}\boldsymbol{p}_2$, so gilt ${}^{A}\boldsymbol{p}_2 = {}^{A}_{B}\boldsymbol{T} \cdot {}^{A}\boldsymbol{p}_1$. Dies lässt sich insbesondere dazu verwenden, bei einer Transformation (beispielsweise einer Translation oder einer Rotation) eines Koordinatensystems auch die darin definierten Punkte bzw. Koordinatensysteme zu transformieren.

In dieser Arbeit kommen insbesondere die erste und die dritte Interpretation mehrfach zur Anwendung. Dabei wird im weiteren Verlauf der Arbeit eine Transformation von einem Koordinatensystem $\mathcal{K}_B$ in das Koordinatensystem $\mathcal{K}_A$ ohne explizite Definition als ${}^{A}_{B}\boldsymbol{T}$ bezeichnet. Um Koordinatensysteme informationstechnisch zu speichern, können (nach der ersten Interpretation) ebenfalls homogene Transformationen verwendet werden. Dabei ist es sinnvoll, soweit möglich, alle Koordinatensysteme in einem gemeinsamen Referenzkoordinatensystem zu beschreiben. In der Robotik wird dafür ein Weltkoordinatensystem $\mathcal{K}_0$ verwendet, das in dieser Arbeit dem Basiskoordinatensystem des Roboters entspricht. Alle weiteren Koordinatensysteme werden daher in Abhängigkeit von diesem Koordinatensystem definiert. Somit kann beispielsweise das Werkstückkoordinatensystem $\mathcal{K}_W$ durch die Transformationsmatrix ${}^{0}_{W}\boldsymbol{T}$ relativ zum Weltkoordinatensystem $\mathcal{K}_0$ beschrieben werden.

Die Transformation $\boldsymbol{T}$ lässt sich als sechsdimensionaler Vektor $(x, y, z, \psi, \theta, \phi)^T$ parametrisieren, der die Translation entlang und die Rotation um die Koordinatenachsen des Referenzkoordinatensystems beschreibt. Bei Verwendung dieser Repräsentation ist jedoch zu beachten, dass es verschiedene Konventionen zur Darstellung der Rotation gibt (Siciliano et al. 2008). Wenn in dieser Arbeit auf diese Repräsentation Bezug genommen wird, so beziehen sich die Rotationen der Reihe nach auf die feststehenden x-, y- und z-Achsen. Die Rotationsmatrix $\boldsymbol{R}$ berechnet sich somit folgendermaßen aus den Winkeln ψ, θ und ϕ:

$$
\begin{aligned}
\boldsymbol{R}(\psi,\theta,\phi) &= \boldsymbol{R}_z(\phi)\cdot\boldsymbol{R}_y(\theta)\cdot\boldsymbol{R}_x(\psi) \\
&= \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \\
&= \begin{pmatrix} \cos\phi\cos\theta & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \sin\phi\cos\theta & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi \\ -\sin\theta & \cos\theta\sin\psi & \cos\theta\cos\psi \end{pmatrix}
\end{aligned}
\tag{2.4}
$$

Zur Repräsentation der Rotation sind auch *Einheitsquaternionen* und die *Angle-Axis*-Darstellung üblich (Siciliano et al. 2008). Für die Umrechnung einer homogenen Transformation in die verschiedenen Repräsentationen sei ebenfalls auf Siciliano et al. (2008) verwiesen. Um numerische Instabilitäten bei der Berechnung von Einheitsquaternionen zu vermeiden, wird in dieser Arbeit das Verfahren von Shepperd (1978) verwendet.

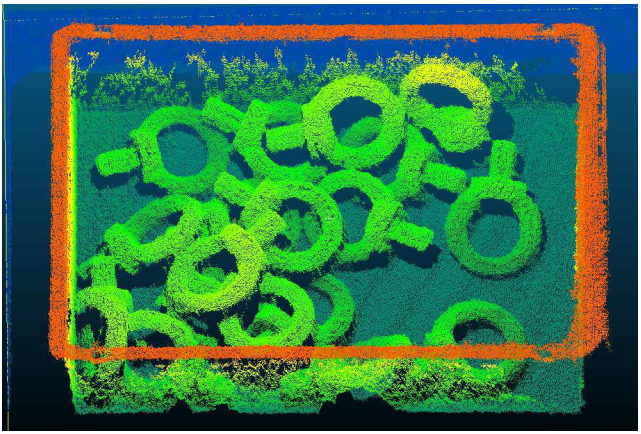


Abbildung 2.2: Grafische Darstellung einer Punktwolke eines mit Ringschrauben gefüllten Kleinladungsträgers. Die Farbe eines Punkts hängt in dieser Darstellung von seiner z-Koordinate ab.

2.1.2 Punktwolken und Flächenmodelle

Die für den Griff-in-die-Kiste eingesetzten 3D-Sensoren erfassen die Oberflächen der sich im Messbereich befindlichen Objekte und generieren Punktwolken bzw. Tiefenkarten, die diese approximieren. Bei modellbasierten Ansätzen bietet diese Punktwolke bzw. Tiefenkarte die Grundlage für die Objektlageerkennung. Um alle potenziellen Hindernisse berücksichtigen zu können, greift auch die in dieser Arbeit entwickelte Greifplanung für die Kollisionsprüfung auf diese Punktwolke zurück. Eine solche Punktwolke ist durch eine Menge $P = \{p_i | p_i \in \mathbb{R}^3\}$ definiert, deren Punkte die Oberflächen der im Sichtbereich des Sensors befindlichen Objekte repräsentieren. Die Anzahl der Punkte in einer Punktwolke wird im Folgenden mit N_P bezeichnet. Abbildung 2.2 zeigt die grafische Darstellung einer beispielhaften Punktwolke. Abschnitt 2.4.1 gibt einen kurzen Überblick über die gängigsten Messprinzipien, die zur Erzeugung einer Punktwolke verwendet werden können.

Bei der Konstruktion von Maschinen und Anlagen ist die Verwendung von CAD-Programmen gängige Praxis. Dies bedeutet, dass sowohl für Roboter, Greifer und andere Komponenten innerhalb der Anlage als auch für die zu vereinzelnden Werkstücke üblicherweise CAD-Daten zur Verfügung stehen. Das Format dieser CAD-Daten ist aufgrund vieler unterstützter Funktionen sehr komplex und hängt stark von der verwendeten CAD-Software ab, allerdings können alle gängigen CAD-Programme die Konstruktionen auch in Standard-Formaten, wie z.B. *STEP (Standard for the exchange of product model data)*, aber auch in triangulierter Form als Flächen-

modell exportieren. Ein solches Flächenmodell lässt sich durch eine Liste von Punkten aus $\mathbb{R}^3$ sowie eine Menge von Dreiecken definieren, die zwischen diesen Punkte aufgespannt werden. Dabei wird jedes Dreieck durch ein 3-Tupel beschrieben, das die Indizes der Punkte beinhaltet, die dieses Dreieck aufspannen. Im Weiteren wird davon ausgegangen, dass dieses Dreiecksnetz geschlossen ist, das heißt, dass es ein Volumen einschließt, das durch das Dreiecksnetz vollständig vom umgebenden Raum getrennt ist. Die im Folgenden verwendeten Flächenmodelle dürfen aus mehreren solchen, von einander getrennten Dreiecksnetzen bestehen. Der Vorteil eines solchen Modells gegenüber einem komplexeren Format, wie beispielsweise STEP, besteht in der einfachen Verarbeitung, da alle Operationen zur Verarbeitung dieser Modelle auf Punkten beziehungsweise auf Dreiecken ausgeführt werden können. Für die Speicherung und den Austausch solcher Dreiecksnetze stehen verschiedene Dateiformate zur Verfügung (Heckner et al. 2014). Dazu gehört beispielsweise das Format *VRML (Virtual Reality Modeling Language)*, das für die Implementierungen, die im Rahmen dieser Arbeit entstanden sind, verwendet wurde. Das ebenfalls weit verbreitete Format *STL (Standard Triangulation Language)* weicht insofern von der obigen Definition ab, als dass die Punkte direkt in den Dreiecken und damit mehrfach gespeichert werden. Dennoch lässt sich ein Modell, das in diesem Format vorliegt, sehr einfach in das redundanzfreie Format VRML konvertieren.

Bei der Generierung von triangulierten Flächenmodellen können viele geometrische Formen nur näherungsweise beschrieben werden. Daher spielt die Anzahl der Dreiecke und damit auch die Anzahl der Punkte, die für das Modell verwendet werden, eine entscheidende Rolle. Werden nur sehr wenig Punkte verwendet, so wird das Flächenmodell ungenau, wohingegen eine hohe Dreiecksanzahl die Rechenzeit darauf arbeitender Algorithmen erhöht. Um Rechenzeiten zu verkürzen, empfiehlt es sich, die Dreiecksanzahl der Flächenmodelle algorithmisch zu reduzieren oder ein neues Modell aus einfachen geometrischen Formen (beispielsweise Quadern) zu konstruieren. Abbildung 2.3 zeigt den Unterschied zwischen dem konstruierten Modell einer Ringschraube und den daraus generierten triangulierten Flächenmodellen mit unterschiedlicher Anzahl an Punkten und Dreiecken. Zur Reduzierung der Anzahl der Dreiecke wurde an dieser Stelle der Algorithmus von Melax (1998) verwendet.

Wie in Abschnitt 2.1.1 gezeigt, können Punkte durch eine Transformationsmatrix im Raum transformiert werden. Da sowohl Punktwolken als auch die hier beschriebenen Flächenmodelle auf einer Menge von Punkten basieren, können beide mit den Verfahren für Punkte im Raum transformiert werden, wovon im weiteren Verlauf der Arbeit mehrfach Gebrauch gemacht wird. Hier zeigt sich auch der Vorteil, in jedem Dreieck nur die Indizes der drei verwendeten Punkte zu speichern. Auf diese Weise muss jeder Punkt des Flächenmodells nur einmal gespeichert und damit auch nur einmal transformiert werden, was die Rechenzeit zur Transformation der Flächenmodelle kurz hält.

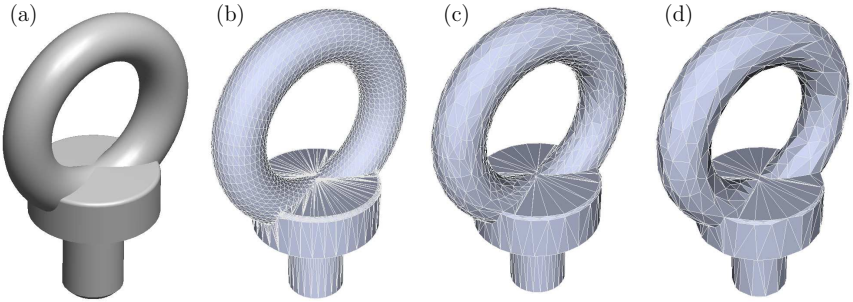


Abbildung 2.3: CAD-Modell (a) und Flächenmodelle einer M30-Ringschraube. Aus dem triangulierten Flächenmodell bestehend aus 2509 Punkten und 5018 Dreiecken (b) wurden reduzierte Flächenmodelle mit 1000 Punkten und 2000 Dreiecken (c) beziehungsweise mit 500 Punkten und 1000 Dreiecken (d) erstellt.

Um Berechnungen mit Flächenmodellen oder einzelnen Dreiecken zu beschleunigen, kommen im späteren Verlauf der Arbeit quaderförmige Hüllkörper zum Einsatz, die an den Achsen des Weltkoordinatensystems ausgerichtet sind. Diese sogenannten *Axis-Aligned Bounding Boxes (AABB)* lassen sich effizient berechnen, indem die minimalen und maximalen Koordinaten aller enthaltenen Punkte bestimmt werden.

2.2 Suchverfahren

Die in dieser Arbeit entwickelte Greifplanung basiert auf der Suche einer geeigneten Lösung innerhalb eines großen Suchraums. Zur Strukturierung des Suchraums kommt ein graphentheoretischer Baum zum Einsatz. Für ein besseres Verständnis der Arbeit wird in diesem Abschnitt daher zunächst auf einige Grundlagen der Graphentheorie eingegangen. Im Anschluss werden wesentliche Grundbegriffe zur Einordnung von Suchverfahren eingeführt sowie verschiedene konkrete Suchverfahren vorgestellt.

2.2.1 Graphen und Bäume

Ein Graph besteht aus Knoten, die mit Kanten verbunden sind. Falls diese Kanten eine Richtung besitzen, spricht man von einem *gerichteten* Graphen, andernfalls von einem *ungerichteten* Graphen. Die Kanten können zusätzlich mit einem Gewicht versehen sein. In solch einem Fall spricht man von einem *kantengewichteten* Graphen. Bei einem gerichteten Graphen bezeichnet der *Eingangsgrad* eines Knotens die Anzahl der Kanten, die zu diesen Knoten führen, und der

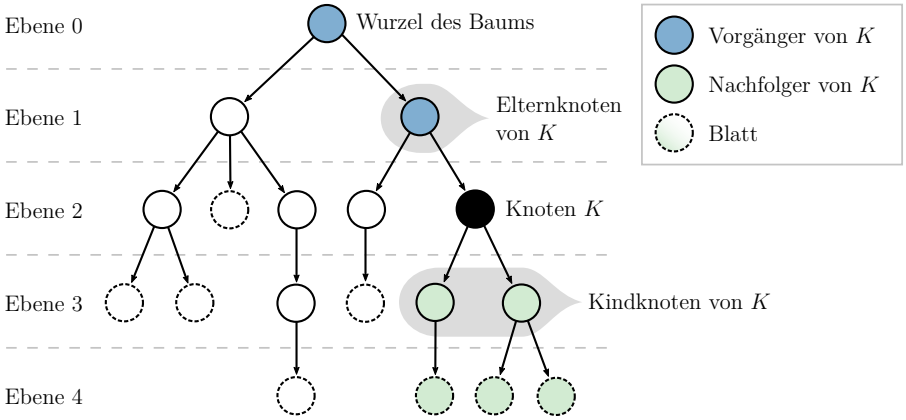


Abbildung 2.4: Beispiel eines Baums im Sinne der Graphentheorie

Ausgangsgrad die Anzahl der Kanten, die von diesem Knoten ausgehen. Eine Folge von Knoten, die jeweils paarweise mit einer Kante verbunden sind, wird als *Pfad* bezeichnet.

Ein gerichteter Graph, in dem jeder Knoten durch genau einen Pfad von einem *Wurzelknoten* (auch *Wurzel* genannt) aus erreichbar ist, wird als *gewurzelter Baum* bezeichnet. In dieser Arbeit wird dieser, der Einfachheit halber, als *Baum* bezeichnet, auch wenn es in der Graphentheorie andere (ungerichtete) Bäume gibt. In einem gewurzelten Baum haben alle Knoten, außer der Wurzel, den Eingangsgrad 1. Abbildung 2.4 zeigt ein Beispiel eines solchen Baums und einige relevante Begriffe. Jeder Knoten eines Baums kann, basierend auf dem Abstand zur Wurzel, genau einer Ebene zugeordnet werden. Die Ebene, zu der ein Knoten K gehört, entspricht der Anzahl der Kanten des Pfades von der Wurzel zu K. Alle Knoten, von denen ein Knoten K über einen Pfad erreichbar ist, werden als *Vorgänger* oder *Vorgängerknoten* von K bezeichnet. Alle Knoten, die von einem Knoten K über einen Pfad erreichbar sind, werden als *Nachfolger* oder *Nachfolgerknoten* von K bezeichnet. Der direkte Vorgänger eines Knotens K heißt *Elternknoten* von K. Direkte Nachfolger eines Knotens K heißen *Kinder* bzw. *Kindknoten* von K. Ein Knoten mit Ausgangsgrad 0 wird als *Blatt* bezeichnet.

2.2.2 Einführung in Suchverfahren

Eine der einfachsten Möglichkeiten, in einer gegebenen Menge von Daten nach gewünschten Lösungen zu suchen, ist die sogenannte erschöpfende Suche. Dabei wird der gesamte Suchraum vollständig durchsucht, um die beste Lösung auszuwählen. In vielen realen Anwendungen ist der Suchraum jedoch sehr groß. Daher wird häufig versucht, die Daten so zu strukturieren, dass

sich ein *Suchbaum* erstellen lässt, durch den die Daten gezielt durchsucht werden können. Meist wird ein solcher Suchbaum nicht vorab explizit generiert, sondern nur bei Bedarf aufgebaut. Bei einem solchen implizit generierten Suchbaum werden die Kindknoten eines Knotens erst in den Suchbaum eingefügt, sobald dieser Knoten betrachtet wird. Das Einfügen der Kindknoten eines Knotens in einen Baum wird als *Expandieren* bezeichnet. Knoten, die noch nicht expandiert wurden, nennt man *offen* und Knoten, die bereits expandiert wurden, *geschlossen* (Winston 1984). Die Wurzel eines Suchbaums wird auch *Startknoten* oder *Wurzelknoten* genannt, die gesuchten Knoten nennt man *Zielknoten*. Suchverfahren lassen sich aufgrund folgender Eigenschaften bewerten (Burkhard 2006):

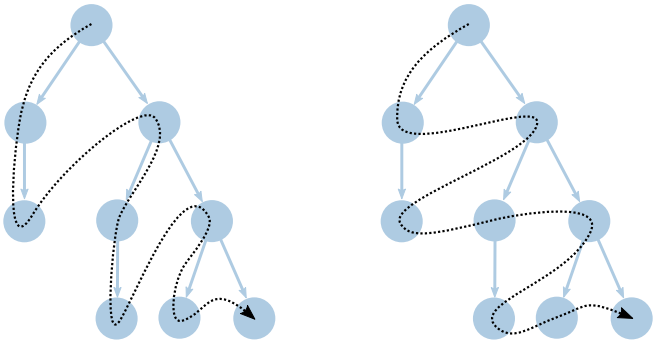
- Ein Suchverfahren wird als *korrekt* bezeichnet, wenn alle gefundenen Lösungen korrekt sind.
- Ein Suchverfahren wird als *vollständig* bezeichnet, wenn garantiert ist, dass es in einer endlichen Zeit eine Lösung findet, sofern diese existiert.
- Ein Suchverfahren wird als *optimal* bezeichnet, wenn garantiert ist, dass das Suchverfahren die optimale Lösung findet.

An dieser Stelle sei weiterhin auf zwei Eigenschaften von Algorithmen hingewiesen, die auch bei Suchverfahren relevant sind (Claus et al. 2001):

- Ein Algorithmus ist *determiniert*, wenn er bei gleicher Eingabe und Startbedingungen stets das selbe Ergebnis liefert.
- Ein Algorithmus ist *deterministisch*, wenn zu jedem Zeitpunkt der Folgeschritt eindeutig bestimmt ist, also alle Zustände reproduzierbar sind. Ein deterministischer Algorithmus ist stets determiniert.

Die meisten Suchverfahren sind sowohl determiniert als auch deterministisch.

Die Verfahren zur Suche in Bäumen lassen sich in *uninformierte* und *informierte* Suchverfahren unterteilen. *Uninformierte* Suchverfahren durchsuchen den Suchraum ohne Kenntnis von problemspezifischen Informationen, wohingegen *informierte Suchverfahren* diese Informationen nutzen, um die Suche zu steuern (Rothlauf 2011, S. 62). Die wichtigsten Beispiele für solche Suchverfahren werden in den folgenden Abschnitten vorgestellt.



(a) Tiefensuche (b) Breitensuche

Abbildung 2.5: Beispiele uninformierter Suchverfahren

2.2.3 Uninformierte Suchverfahren

Die bekanntesten uninformierten Suchverfahren sind die *Tiefensuche* und die *Breitensuche*, die in Abbildung 2.5 beispielhaft dargestellt sind. Bei der Tiefensuche werden jeweils die Kinder eines Knotens vollständig expandiert, bevor seine Nachbarknoten expandiert werden. Hingegen werden bei der Breitensuche zunächst alle Knoten einer Ebene expandiert, bevor die Kinder der Knoten dieser Ebene expandiert werden. Die Reihenfolge der Expansion der Kinder eines Knotens ist bei beiden Suchverfahren fest (zum Beispiel von links nach rechts). Die Wahl des geeigneteren uninformierten Suchverfahrens hängt von der entsprechenden Anwendung ab. Falls sich beispielsweise alle Zielknoten in einer ähnlichen Tiefe des Suchbaumes befinden, ist die Breitensuche nicht effizient, da zunächst alle übergeordneten Knoten durchsucht werden, bevor der erste mögliche Zielknoten erreicht wird.

2.2.4 Informierte Suchverfahren

Bei den heuristischen Suchverfahren, auch informierte Suchverfahren genannt, werden Informationen über die zu durchsuchenden Daten genutzt, um die Suche zu steuern und damit zu beschleunigen. Dazu wird eine sogenannte *Heuristikfunktion* $h(K)$ (auch *Kostenfunktion* oder *evaluation function*) definiert, die die Kosten abschätzt, einen Zielknoten von einem gegebenen Knoten K aus zu erreichen (Rothlauf 2011, S. 64). Basierend auf dieser Heuristikfunktion kann entschieden werden, welche Knoten expandiert werden sollten, um möglichst schnell einen Zielknoten zu finden. Die Wahl einer geeigneten Heuristikfunktion ist eine Abwägung zwischen der Genauigkeit der Funktion und dem Aufwand, das Ergebnis der Funktion zu berechnen.

Eine sehr genaue Heuristikfunktion kann die Anzahl der benötigten Schritte während der Suche reduzieren. Wenn die Berechnung dieser Funktion allerdings sehr lange dauert, kann sich die Suchzeit dadurch erhöhen.

- Eine Heuristik ist *zulässig*, wenn sie die tatsächlichen Kosten niemals überschätzt, das heißt, wenn $h(K)$ stets kleiner oder gleich den tatsächlichen Kosten ist.

- Eine Heuristik ist *monoton*, wenn für alle Knoten K und für alle direkten Nachfolger K' von K gilt
$$h(K) \leq h(K') + c(K, K')\,, \tag{2.5}$$
wobei $c(K, K')$ die tatsächlichen Kosten angibt, um von K nach K' zu kommen. Jede monotone Heuristik ist auch zulässig (Lunze 2016).

2.2.4.1 Beam Search

Ein einfaches Beispiel eines informierten Suchverfahrens stellt das sogenannte *Beam Search* dar. Dieses Suchverfahren ähnelt der Breitensuche, jedoch werden nur die n besten Knoten einer Ebene des Suchbaums weiter untersucht. Dadurch verringert sich der Aufwand zum Durchsuchen des Suchbaums signifikant, allerdings kann es vorkommen, dass Lösungen nicht gefunden werden. Dieses Suchverfahren ist daher nicht *vollständig*.

2.2.4.2 Hill Climbing

Hill Climbing, auch *Bergsteigeralgorithmus* genannt, erweitert hingegen das Konzept der Tiefensuche um eine Heuristikfunktion. Im Gegensatz zur Tiefensuche werden jedoch die direkten Nachfolger eines Knotens nicht in einer festen Reihenfolge expandiert, wie zum Beispiel von links nach rechts, sondern auf Basis ihrer Heuristikfunktion. Dabei werden alle Kinder eines Knotens auf Basis ihrer Kosten sortiert und anschließend vom Knoten mit den geringsten Kosten beginnend expandiert. Dieses Suchverfahren ist relativ einfach, hat allerdings in der Praxis einige Probleme. Das für den in dieser Arbeit untersuchten Anwendungsfall relevanteste Problem ist, dass lokale Minima der Heuristikfunktion zu einer ausgiebigen Suche eines Teilbaums führen können, obwohl die Fortführung der Suche an einer anderen Stelle im Baum schneller zum Erfolg führen könnte.

2.2.4.3 Bestensuche

Die *Bestensuche*, auch *Best-First Search* genannt, ähnelt dem Hill-Climbing-Verfahren, jedoch werden hier nicht nur die direkten Nachfolger eines Knotens unter sich sortiert, sondern alle *offenen* Knoten des Suchbaums. Dafür wird eine Vorrangwarteschlange verwendet, in der sich zu Beginn lediglich der Wurzelknoten befindet. Während der Suche wird immer das erste Element der Warteschlange entnommen und expandiert. Die beim Expandieren erstellten Knoten werden entsprechend ihrer Kosten in diese Warteschlange einsortiert. Dieses Vorgehen wird so lange wiederholt, bis ein Zielknoten erreicht wurde.

2.2.4.4 Optimale Suchverfahren

Die bisher genannten Suchverfahren sind darauf ausgelegt, eine beliebige Lösung innerhalb des Suchraums zu finden. Für die Suche nach der optimalen Lösung, d.h. der Lösung mit den geringsten Kosten, stehen andere Suchverfahren zur Verfügung. An dieser Stelle sei hier insbesondere $A*$ genannt (Hart et al. 1968). Die A*-Suche ähnelt der Bestensuche, wobei jedoch in jedem Schritt nicht der Knoten mit der niedrigsten Heuristikfunktion $h(x)$ expandiert wird, sondern der Knoten mit dem niedrigsten $h(x) + c(x)$, wobei $c(x)$ die tatsächlichen Kosten angibt, um vom Startknoten zum Knoten x zu kommen. Um das Finden der optimalen Lösung sicherzustellen, darf die Heuristikfunktion dabei die Kosten nie überschätzen. Weiterhin wird bei diesem Verfahren verhindert, dass verschiedene Pfade zum selben Knoten separat weiter expandiert werden, was allerdings bei der Suche in Bäumen nicht relevant ist, da jeder Knoten nur über einen Pfad vom Wurzelknoten aus erreichbar ist (vgl. Abschnitt 2.2.1).

In Anwendungen, bei denen nicht die optimale, sondern lediglich eine beliebige Lösung benötigt wird, ist die Bestensuche jedoch besser geeignet als die A*-Suche. Durch die Verwendung der A*-Suche kann sich die Dauer der Suche verlängern, da zu jeder Zeit sichergestellt werden muss, dass es keine bessere Lösung gibt und somit gegebenenfalls weitere Knoten expandiert werden müssen.

2.3 Künstliche neuronale Netze

Dieser Abschnitt gibt eine kurze Einführung in *künstliche neuronale Netze* (McCulloch et al. 1943; Rosenblatt 1961), meist nur *neuronale Netze (NN)* genannt. Die Beschreibung lehnt sich an die Ausführungen von Kriesel (2007) und Yang (2018) an. Im Anschluss wird kurz auf die Kombination von künstlichen neuronalen Netzen mit heuristischen Suchverfahren eingegangen.

2.3.1 Einführung in künstliche neuronale Netze

Künstliche neuronale Netze sind dem biologischen Vorbild nachempfundene Netze aus künstlichen Neuronen. Die Topologie neuronaler Netze variiert je nach Aufgabenstellung. Im Allgemeinen sind neuronale Netze mehrschichtig und bestehen aus einer *Eingabeschicht*, einer oder mehrerer *verdeckter Schichten* und einer *Ausgabeschicht*. Jede Schicht besteht aus einem oder mehreren künstlichen Neuronen, wobei die Neuronen der einzelnen Schichten miteinander verbunden sind. Jede dieser Verbindungen ist dabei mit einem Verbindungsgewicht $w_{i,j}$ versehen, so dass die Verbindungen entweder verstärkend oder hemmend wirken können. Abbildung 2.6(a) zeigt den Aufbau eines beispielhaften neuronalen Netzes.

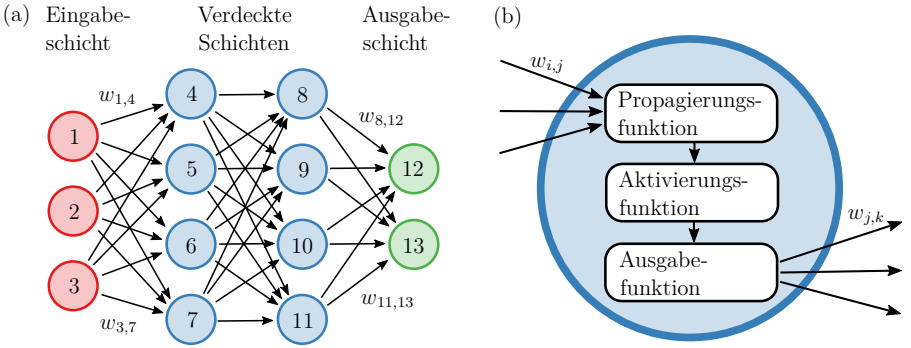


Abbildung 2.6: Aufbau eines künstlichen neuronalen Netzes (a) und Funktionsweise eines einzelnen Neurons (b) (angelehnt an Kriesel (2007)).

Jedes einzelne Neuron besteht aus einer *Propagierungsfunktion*, einer *Aktivierungsfunktion* und einer *Ausgabefunktion*. Die Propagierungsfunktion vereint die Ausgabewerte der vorangehenden Neuronen, meist durch die Berechnung der gewichteten Summe dieser Werte. Die Aktivierungsfunktion berechnet den aktuellen Aktivierungszustand des Neurons auf Basis der Eingabe und des vorherigen Aktivierungszustands. Hierbei wird meist eine Aktivierungsfunktion gewählt, die sicherstellt, dass der Wert immer in einem beschränkten Wertebereich (meist $(0, 1)$ oder $(-1, 1)$) liegt. Eine gängige Aktivierungsfunktion ist beispielsweise die Sigmoid-Funktion (Mitchell 1997). Die Ausgabefunktion berechnet schließlich den Ausgabewert des Neurons und wird häufig als Identität definiert. Abbildung 2.6(b) visualisiert diese Funktionsweise eines einzelnen Neurons. Ein solches neuronales Netz kann somit dazu genutzt werden, für gegebene Eingabewerte die zugehörigen Ausgabewerte zu berechnen. Diese Berechnung wird als *Vorwärtspropagierung* bezeichnet.

2.3.1.1 Trainieren von neuronalen Netzen

Künstliche neuronale Netze werden häufig im Bereich des maschinellen Lernens eingesetzt. Dabei wird das Netz auf Basis einer repräsentativen Menge an Daten, der sogenannten *Trainingsmenge*, trainiert. Die dafür verwendeten Lernverfahren lassen sich in verschiedene Kategorien einordnen, wobei hier lediglich *überwachte Lernverfahren* betrachtet werden sollen. Bei überwachten Lernverfahren enthält die Trainingsmenge sowohl die Eingabe des neuronalen Netzes als auch die gewünschten Ausgabewerte zu jeder Eingabe. Während dieses Trainingsprozesses kann das neuronale Netz auf verschiedene Weise modifiziert werden. In der Praxis werden meist lediglich die Verbindungsgewichte $w_{i,j}$ verändert. Das Ziel des Lernverfahrens ist es somit, die Gewichte des neuronalen Netzes so anzupassen, dass der Fehler E des neuronalen Netzes minimal wird. Dazu wird die Ausgabe des neuronalen Netzes für jede Eingabe der Trainingsmenge berechnet, der Fehlervektor zwischen der berechneten Ausgabe und der gewünschten Ausgabe bestimmt und das Netz auf Basis dieses Fehlervektors verbessert. Der (quadratische) Fehler E berechnet sich aus der Differenz zwischen den gewünschten Ausgabewerten u_k der Trainingsdaten und den tatsächlichen Ausgabewerten v_k des neuronalen Netzes mittels

$$E = \frac{1}{2} \cdot \sum_{k=1}^{N_{aus}} (u_k - v_k)^2 \,, \tag{2.6}$$

wobei N_{aus} die Anzahl der Ausgabewerte bezeichnet. Zur Optimierung der Gewichte kann beispielsweise *Backpropagation* (auch *Backpropagation of Error* genannt) (Werbos 1982; Rumelhart et al. 1985; Rumelhart et al. 1986a; Rumelhart et al. 1986b) zum Einsatz kommen, wobei die Gewichte des neuronalen Netzes auf Basis des Gradientenabstiegs optimiert werden. Dieses gilt als eines der einfachsten und häufig genutzten Lernverfahren (Yang 2018). Zu beachten ist, dass der Wertebereich der Ausgabe eines neuronalen Netzes insbesondere durch die Aktivierungsfunktion beschränkt ist. Das bedeutet, dass die Ausgabewerte so gewählt bzw. normalisiert werden sollten, dass sie innerhalb dieses Wertebereichs liegen, da das neuronale Netz andernfalls selbst für die Trainingsdaten nicht die gewünschten Ausgabewerte berechnen kann.

Das Lernen kann einerseits *online* erfolgen, was bedeutet, dass das Netz nach jedem Trainingsbeispiel angepasst wird. Meist werden neuronale Netze jedoch *offline* trainiert. Dabei wird das neuronale Netz mit einer Menge von Trainingsdaten (üblicherweise der gesamten Trainingsmenge) auf einmal trainiert und daraufhin die Gewichte angepasst. Viele Lernverfahren führen diese Berechnung der Ausgabewerte für die gesamte Trainingsmenge und die Anpassung der Gewichte mehrfach aus, um so iterativ das neuronale Netz zu verbessern. Ein einzelner solcher Durchlauf wird dabei *Epoche* genannt.

2.3.2 Künstliche neuronale Netze und heuristische Suchverfahren

In der Literatur existieren verschiedene Ansätze, künstliche neuronale Netze in Kombination mit heuristischen Suchverfahren zu verwenden (Greenberg 1990). Beispielsweise können heuristische Suchverfahren genutzt werden, um neuronale Netze zu trainieren (Jordanov et al. 2007) oder anzupassen (Opitz et al. 1995). Eine weitere Möglichkeit besteht darin, mittels maschinellem Lernen die jeweils beste Heuristikfunktion auszuwählen. Solche Ansätze bezeichnet man als *Hyperheuristiken*, bei denen versucht wird, das notwendige domänenspezifische Wissen zu reduzieren. Ein Überblick über diverse Ansätze von Hyperheuristiken lässt sich Burke et al. (2013) entnehmen. Anstatt zu lernen, welche Heuristikfunktion jeweils verwendet werden soll, kann auch die Ausgabe eines künstlichen neuronalen Netzes direkt in der Heuristikfunktion eines Suchverfahrens verwendet werden. Chen et al. (2011) zeigen beispielsweise, wie sich der A*-Algorithmus diesbezüglich modifizieren lässt. Die Verwendung von neuronalen Netzen in heuristischen Suchverfahren lässt sich für unterschiedliche Anwendungsfälle verwenden, beispielsweise für die effiziente Bereitstellung von Frachtcontainern (Hottung et al. 2020). Als weiteres Beispiel sei die Software *AlphaGo* (Silver et al. 2016) genannt, die mehrere neuronale Netze in Kombination mit *Monte Carlo tree search (MCTS)* für das Spielen von Go verwendet.

Die Verwendung eines neuronalen Netzes hat das Potenzial, die Notwendigkeit von anwendungsspezifischem Expertenwissen in der Heuristikfunktion eines Suchverfahrens zu reduzieren. In dieser Arbeit wird untersucht, ob sich dies auch für die Suche nach einer Greiflösung und damit für den Anwendungsfall der Greifplanung beim Griff-in-die-Kiste eignet.

2.4 Werkstückvereinzelung durch Roboter

Für das Verständnis der Arbeit wird in diesem Abschnitt in einige Themen eingeführt, die für die industrielle Werkstückvereinzelung relevant sind. Dies beinhaltet insbesondere die Themen 3D-Sensorik, Kalibrierung, Greiftechnik und Roboterkinematik.

2.4.1 Generierung von 3D-Sensordaten

Für die optische Erfassung von dreidimensionalen Objekten und Strukturen steht eine Vielzahl unterschiedlicher Sensoren zur Verfügung. Diese wurden in den letzten Jahren stetig weiterentwickelt, was die Qualität der Sensordaten signifikant verbessert hat. Die Anbindung der Sensoren an einen Industrie-PC, der die Sensordaten verarbeitet, erfolgt heutzutage überwiegend per Ethernet. Das Resultat des 3D-Sensors stellt eine dreidimensionale Punktwolke dar, die auf

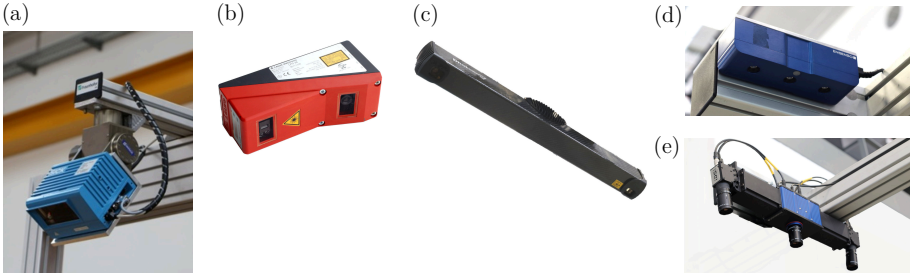


Abbildung 2.7: Verschiedene Sensoren zur Erzeugung von Punktwolken: Laserlaufzeitsensor Sick LMS-400 an einer Schwenkneigeeinheit PW-70 der Firma Schunk (a), Lasertriangulationssensor Leuze LPS-36 (b), Streifenprojektionssensor Photoneo PhoXi 3D Scanner L (c), Stereokamera Ensenso N20-1202-16-BL (d) und Stereokamera Ensenso X-36 (e)

das Koordinatensystem des Sensors bezogen ist, das im Folgenden mit $\mathcal{K}_S$ bezeichnet wird. Neben Sensoren, die selbstständig eine dreidimensionale Punktwolke generieren, besteht zudem die Möglichkeit, einen 2D-Sensor (beispielsweise einen Laserlinienscanner) an einer Schwenk- oder Linearachse zu montieren. Durch die Synchronisation der Achsdaten mit den zugehörigen Sensordaten lässt sich, unter der Annahme einer statischen Szene, ebenfalls eine 3D-Punktwolke generieren. Für beide Varianten existieren verschiedene Messprinzipien, die im Folgenden kurz vorgestellt werden. Abbildung 2.7 zeigt einige Beispielsensoren für die verschiedenen Messprinzipien.

- **Laufzeitmessung und Phasenlagemessung:** Bei der Laufzeitmessung wird Licht (meist Laser oder Infrarot) ausgesandt und die Zeit gemessen, bis das von der Szene reflektierte Licht am Sensor empfangen wird. Teilweise wird diese Dauer durch einen Vergleich der Phasenlage des ausgesandten Lichts und des empfangenen Lichts bestimmt. Dieses Messprinzip verfolgen einerseits *Time-of-Flight-Kameras*, wie zum Beispiel das Modell Helios2 von LUCID Vision Labs, die direkt Punktwolken generieren können, und andererseits Laserlinienscanner, wie zum Beispiel das Modell LMS400 von SICK.

- **Lasertriangulation:** Ein Lasertriangulationssensor projiziert eine Laserlinie (oder auch nur einen einzelnen Punkt) auf die Szene und nimmt diese mit einer oder mehreren zur Austrittsöffnung des Lasers versetzten Kameras auf. Aus der Position der einzelnen Punkte der Laserlinie im Kamerabild kann daraus mittels Triangulation die Entfernung dieser Punkte zum Sensor bestimmt werden. Auch hier existieren 3D-Modelle, wie zum Beispiel das Modell ScanningRuler von SICK, die die Laserlinie zusätzlich mittels eines Spiegels ablenken und somit direkt eine dreidimensionale Punktwolke ausgeben, sowie 2D-Modelle,

die zusätzlich bewegt werden müssen, um eine Punktwolke zu generieren, wie zum Beispiel das Modell LPS 36 von Leuze.

- **Streifenprojektion:** Bei der Streifenprojektion, oft auch Streifenlichtprojektion genannt, werden unterschiedliche Streifenmuster auf die Szene projiziert und von einer oder mehreren Kameras aufgenommen. Die Funktionsweise zur Bestimmung der Entfernung ähnelt der von Lasertriangulationssensoren (Zanuttigh et al. 2016). Diese Projektion der Streifenmuster erfolgt dabei meist mittels eines LED- oder Laserprojektors, wie zum Beispiel beim Modell PhoXi 3D Scanner L des Herstellers Photoneo.

- **Stereokameras:** Eine Stereokamera besteht im Wesentlichen aus zwei zueinander seitlich versetzten Graustufen- oder Farbkameras. Zur Tiefenbestimmung wird in den Einzelbildern dieser Kameras nach korrespondierenden Merkmalen gesucht. Aufgrund des Versatzes der beiden Kameras befinden sich solche Merkmale an unterschiedlichen Positionen in den beiden Einzelbildern. Die Differenz der Position dieser Merkmale entlang des Kameraversatzes wird als *Disparität* bezeichnet und kann durch Kenntnis des Abstands der beiden Kameras, des Vergenzwinkels und der Objektiveigenschaften zur Bestimmung der Tiefe verwendet werden. Einige Stereokameras, wie zum Beispiel die Modelle von Ensenso verwenden zur Unterstützung einen Projektor, der ein Muster auf die Szene projiziert, um mehr Merkmale in der Szene zu generieren und sicherzustellen, dass auch in andernfalls monotonen Bereichen ausreichend Merkmale für eine Tiefenbestimmung zu finden sind.

Die verschiedenen Sensoren unterscheiden sich stark in ihren Eigenschaften. Insbesondere ist dabei der Messbereich zu nennen, der lediglich Kleinladungsträger abdecken oder sogar für große Gitterboxen geeignet sein kann. Je nach Messprinzip und Sensormodell sind die Punktwolken mit systematischen und statistischen Fehlern behaftet, weshalb die erzeugte Punktwolke verrauscht ist und die gemessenen Oberflächen lediglich approximiert. Der statistische Fehler des Messabstands wird von den Herstellern spezifiziert und über die Standardabweichung angegeben. Bei einigen Sensoren ist diese Angabe abhängig vom Messabstand. Im Allgemeinen ist der statistische Fehler bei Sensoren mit großem Messbereich größer als bei Sensoren mit kleinem Messbereich. Eine weitere wichtige Eigenschaft von 3D-Sensoren ist die Anzahl der Messpunkte, die bei Time-of-Flight-Kameras meist geringer ist als bei anderen Sensoren. Bei der Erfassung der Werkstücke ist zudem zu berücksichtigen, dass es Abschattungen beispielsweise durch den Kistenrand geben kann. Dieses Problem ist bei der Lasertriangulation, bei der Streifenprojektion und bei Stereokameras besonders zu beachten, da hier eine freie Sichtverbindung zu zwei Punkten am Sensor bestehen muss (entweder zu zwei Kameras oder zu einer Kamera und einer Projektionseinheit). Schließlich muss bei der Auswahl des optimalen Sensors auch das Reflexionsverhalten der Werkstückoberflächen und mögliches Fremdlicht (vor allem Sonneneinstrahlung) berücksichtigt werden. Im Umgang mit diesen Problemen unterscheiden sich

die verschiedenen Sensoren deutlich. Zusammengefasst lässt sich festhalten, dass es keinen universellen, optimalen Sensor gibt, sondern dieser speziell für den jeweiligen Anwendungsfall ausgewählt werden muss.

2.4.2 Hand-Auge-Kalibrierung

Da die Sensordaten zunächst im Koordinatensystem $\mathcal{K}_S$ des Sensors erfasst werden, der Roboter sich jedoch basierend auf diesen Daten bewegen soll, ist es notwendig, eine Beziehung zwischen dem Weltkoordinatensystem $\mathcal{K}_0$ und $\mathcal{K}_S$ herzustellen. Für die Verwendung in der Greifplanung ist es naheliegend, die Sensordaten in $\mathcal{K}_0$ zu transformieren, um somit alle weiteren Berechnungen basierend auf $\mathcal{K}_0$ durchführen zu können. Dazu werden alle Punkte der Punktwolke mittels

$$ {}^{0}\boldsymbol{p}_i = {}^{0}_{S}\boldsymbol{T} \cdot {}^{S}\boldsymbol{p}_i \tag{2.7} $$

vom Koordinatensystem $\mathcal{K}_S$ des Sensors in das Weltkoordinatensystem $\mathcal{K}_0$ transformiert. Die Transformation ${}^{0}_{S}\boldsymbol{T}$ lässt sich durch eine sogenannte Hand-Auge-Kalibrierung bestimmen. In dieser Arbeit erfolgt die Hand-Auge-Kalibrierung, wie auch in Spenrath et al. (2013), indem die Lage eines Kalibrierkörpers einerseits mit dem Roboter vermessen wird und andererseits von der 3D-Sensorik erfasst und lokalisiert wird. Auf Basis dieser beiden Lagen lässt sich ${}^{0}_{S}\boldsymbol{T}$ mittels

$$ {}^{0}_{S}\boldsymbol{T} = {}^{0}_{K}\boldsymbol{T} \cdot \left({}^{S}_{K}\boldsymbol{T}\right)^{-1} \tag{2.8} $$

berechnen. Dabei beschreibt die Transformation ${}^{S}_{K}\boldsymbol{T}$ die Lage des Kalibrierkörpers im Koordinatensystem $\mathcal{K}_S$ des Sensors und ${}^{0}_{K}\boldsymbol{T}$ die durch den Roboter eingemessene Lage des Kalibrierkörpers im Weltkoordinatensystem $\mathcal{K}_0$.

Die bisherigen Ausführungen gehen davon aus, dass der Sensor stationär über dem Ladungsträger montiert ist. Der Sensor kann jedoch auch in den Greifer integriert und somit durch den Roboter geführt werden. Durch die Kenntnis der Lage des Roboterflanschs während des Kalibriervorgangs und während der Szenenerfassung für den Entnahmeprozess kann die Kalibrierung dabei im Wesentlichen mit den gleichen Verfahren durchgeführt werden. Dazu muss während der Kalibrierung lediglich einmalig die Lage des Sensors relativ zum Roboterflansch ermittelt werden. Im Anschluss kann die Lage des Sensors relativ zum Weltkoordinatensystem für jede Pose des Roboters bestimmt werden. Die Umrechnung zwischen der Lage des Sensors im Weltkoordinatensystem und der durch ${}^{F}_{S}\boldsymbol{T}$ beschriebenen Lage des Sensors relativ zum Roboterflansch kann durch die Gleichungen

$$ {}^{F}_{S}\boldsymbol{T} = \left({}^{0}_{F}\boldsymbol{T}\right)^{-1} \cdot {}^{0}_{S}\boldsymbol{T} \tag{2.9} $$

und

$$ {}^{0}_{S}\boldsymbol{T} = {}^{0}_{F}\boldsymbol{T} \cdot {}^{F}_{S}\boldsymbol{T} \tag{2.10} $$

erfolgen, wobei ${}^{0}_{F}\boldsymbol{T}$ die bekannte Lage des Roboterflanschs relativ zum Weltkoordinatensystem beschreibt und durch die Roboterbewegung veränderlich ist.

2.4.3 Greifprinzipien

Trotz langjähriger Entwicklungen im Bereich der Greiftechnik existiert nach wie vor kein universeller Greifer, der beliebige Werkstücke prozesssicher greifen und entnehmen kann. Daher wird bei Roboteranlagen üblicherweise ein speziell auf die zu handhabenden Werkstücke zugeschnittener Greifer konstruiert und gefertigt. Zur Umsetzung eines konkreten Greifers stehen diverse Greifprinzipien zur Verfügung, von denen für eine konkrete Anwendung eines oder mehrere ausgewählt werden müssen. Das Greifprinzip ist bei der Greifplanung zu berücksichtigen, daher werden die in der Robotik am häufigsten verwendeten Prinzipien im Folgenden beschrieben. Abbildung 2.8 zeigt beispielhaft einen konkreten Greifer für jedes dieser Prinzipien.

- **Klemmgreifer** eignen sich nur für Werkstücke, die eine ausreichende Anzahl geeigneter Merkmale aufweisen, an denen das Werkstück gegriffen werden kann. Ob ein Werkstück an einer bestimmten Stelle gegriffen werden kann, ist von der Geometrie und Bewegung der Greiferfinger und der Geometrie der Greifstelle abhängig. Insbesondere parallele Oberflächen und zylindrische Abschnitte eignen sich für einen Griff mit einem Klemmgreifer. Das Greifen erfolgt dabei entweder durch das Öffnen oder durch das Schließen des Greifers. Es existieren Varianten mit zwei, drei und selten auch mehr Greiferfingern, die entweder pneumatisch oder elektrisch angetrieben werden. Die Greiferfinger von Klemmgreifern werden üblicherweise speziell für einen Anwendungsfall konstruiert und gefertigt. Beim Greifen von ungeordnet gelagerten Werkstücken tritt das Problem auf, dass ein Werkstück von verschiedenen Seiten greifbar sein muss und daher die Greiferfinger für unterschiedliche Griffe geeignet sein müssen. Ein elektrischer Greifer hat dabei den Vorteil, dass die Greiferfinger beliebige Positionen anfahren können und somit mehr Flexibilität beim Hub und damit bei der Auswahl möglicher Greifstellen besteht. *Formschlüssige* Griffe, bei denen sich das Werkstück selbst im Greifer zentriert, sind dabei *kraftschlüssigen* Griffen vorzuziehen, weil sie eine definierte Ablage erleichtern können, da nach dem Greifvorgang die relative Lage von Greifer und Werkstück wohldefiniert ist.

- **Sauggreifer** (auch Vakuumgreifer genannt) greifen ein Werkstück durch einen oder mehrere Saugnäpfe mittels Unterdruck. Sie bieten im Gegensatz zum Klemmgreifer den Vorteil, dass das Werkstück nur an einer Fläche zugänglich sein muss, was beispielsweise

(a) (b) (c)

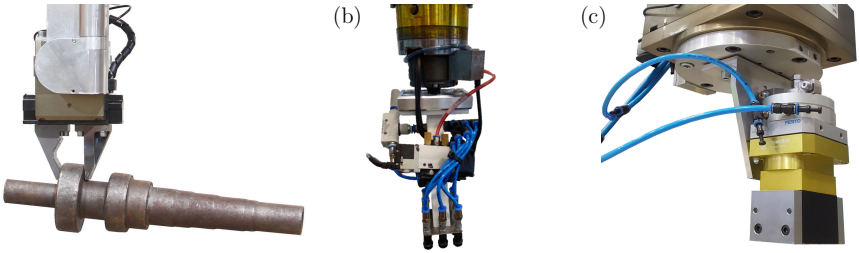


Abbildung 2.8: Beispielgreifer der gängigen Greifprinzipien: Zweifinger-Klemmgreifer mit Werkstück (Quelle des Originalfotos: Fraunhofer IPA) (a), Sauggreifer mit mehreren Saugnäpfen (b) und pneumatisch geschalteter Permanentmagnetgreifer (c)

das Greifen eng aneinander liegender Werkstücke vereinfachen kann. Allerdings lässt sich physikalisch meist nicht garantieren, dass ein Werkstück stets wie gewünscht gegriffen wird. Oft haftet ein Werkstück auch dann, wenn die gewünschte Greifpose verfehlt wurde, wodurch das Werkstück zwar gegriffen, aber nicht genau abgelegt werden kann. Die einzelnen Saugnäpfe können mit einem Faltenbalg versehen sein. Dies erlaubt einerseits mehr Toleranz beim Greifen, hat andererseits aber ebenfalls den Nachteil, dass das Werkstück durch die Verformung der Saugnäpfe nicht so genau abgelegt werden kann. Ein weiterer Nachteil ist, dass die Haltekraft nur ausreicht, um vollständig frei liegende Werkstücke sicher zu greifen. Werkstücke, die von anderen Werkstücken teilweise verdeckt sind, lassen sich somit meist nicht greifen und entnehmen. Die Erzeugung des Unterdrucks zum Ansaugen des Werkstücks erfolgt über einen Ejektor, eine Vakuumpumpe oder ein Gebläse.

- **Magnetgreifer** ermöglichen eine hohe Greifkraft und werden daher häufig für schwere Guss- oder Schmiedeteile eingesetzt. Sie eignen sich nur für magnetische Werkstücke, bieten jedoch ebenfalls den Vorteil, dass das Werkstück nur an einer Fläche (beziehungsweise einer zum Magnet passenden Form) zugänglich sein muss. Durch eine geeignete Form des Magnetgreifers kann teilweise physikalisch sichergestellt werden, dass ein Werkstück nur an einer bestimmten Stelle gegriffen werden kann. Meist ist dies allerdings nicht möglich, sodass ein Werkstück häufig auch gegriffen wird, wenn der Greifer die gewünschte Greifpose verfehlt. Bei Magnetgreifern besteht zudem die Gefahr, dass ungewollt mehrere Werkstücke gleichzeitig gegriffen werden. Beide Effekte führen dazu, dass ein gegriffenes Werkstück nicht sicher abgelegt werden kann. Ein weiterer Nachteil ist, dass die Werkstücke durch das Greifen magnetisiert werden können, was in weiteren Prozessschritten Probleme verursachen kann. Für dieses Greifprinzip existieren Ausführungen mit Permanentmagneten, die meist pneumatisch umgeschaltet werden können, und Ausführungen mit Elektromagneten.

Bei der Arbeit mit Greifern und anderen Werkzeugen wird in der Robotik ein sogenannter *Werkzeugmittelpunkt (engl.: Tool Center Point (TCP))* verwendet. Trotz des Namens stellt dieser Werkzeugmittelpunkt eine Pose dar und wird auch in dieser Arbeit in diesem Sinne verwendet. Der TCP definiert die Lage eines Werkzeugkoordinatensystems $\mathcal{K}_T$ relativ zum Flanschkoordinatensystem $\mathcal{K}_F$ des Industrieroboters. Die Bewegung des Roboters kann somit auf Basis dieses TCP programmiert werden. Einzelne Posen des Roboters können durch die Lage des TCP-Koordinatensystems $\mathcal{K}_T$ relativ zum Weltkoordinatensystem $\mathcal{K}_0$ definiert werden. Industrieroboter besitzen Funktionen, um den TCP für ein montiertes Werkzeug einzumessen. Bei einem Klemmgreifer wird der TCP meist mittig zwischen den Greiferfingern definiert, bei Magnet- und Sauggreifern an deren Unterseite, mit der das Werkstück gegriffen wird. Dadurch lässt sich eine Greifpose an einem Werkstück festlegen, indem der TCP relativ zum zu greifenden Werkstück positioniert wird. Durch diese Formalisierung lassen sich grundsätzlich alle Greifprinzipien modellieren, bei denen der Greifprozess bei einer konstanten Position des Roboters stattfindet. Da für ungeordnet gelagerte Werkstücke normalerweise Klemm-, Saug- oder Magnetgreifer zum Einsatz kommen, beschränkt sich die weitere Betrachtung in dieser Arbeit jedoch auf diese Prinzipien.

2.4.4 Roboterkinematik und Arbeitsraum

Nach der Norm ISO 8373 ist ein Industrieroboter ein automatisch gesteuerter, programmierbarer und vielseitig einsetzbarer Manipulator mit mindestens drei Achsen. Während für schnelle *Pick-and-Place-Aufgaben* bei kleinen Werkstücken oft parallele Roboterkinematiken, wie zum Beispiel *Delta-Roboter* zum Einsatz kommen, werden beim Vereinzeln von Werkstücken aus Kisten und Gitterboxen meist Knickarmroboter, und somit serielle Roboterkinematiken, eingesetzt. Diese haben gegenüber den Delta-Robotern einen deutlich größeren Arbeitsraum und haben, durch ihre serielle Bauweise, eine geringere Störkontur und somit weniger Einschränkungen durch Kollisionen. Obwohl sich die Inhalte dieser Arbeit weitgehend auf andere Roboterkinematiken übertragen lassen, beschränkt sich die Betrachtung im Folgenden auf serielle Roboterkinematiken mit sechs Achsen.

Die Berechnung der Lage des Endeffektors eines Industrieroboters auf Basis der Positionen seiner einzelnen Roboterachsen bezeichnet man als *Vorwärtskinematik*. Die Berechnung der Vorwärtskinematik ist, im Gegensatz zu parallelen Roboterkinematiken, bei seriellen Roboterkinematiken trivial und eindeutig lösbar. Analog versteht man unter der *inversen Kinematik* (auch *Rückwärtskinematik* genannt) die Berechnung der Positionen der einzelnen Roboterachsen auf Basis der Lage des Endeffektors. Obwohl die Berechnung der inversen Kinematik bei seriellen Roboterkinematiken komplexer ist, existieren diverse Lösungsverfahren, meist auf

Basis der *Denavit-Hartenberg-Konvention* (Denavit et al. 1955; Hartenberg et al. 1964). Für Knickarmroboter mit sechs Achsen, bei denen sich die Achsen 4 bis 6 in einem gemeinsamen Punkt, dem sogenannten *Handwurzelpunkt*, schneiden, lässt sich die inverse Kinematik deutlich einfacher und effizienter berechnen (vgl. Paul et al. (1986) oder Abschnitt 2.12.2 in Siciliano et al. (2009)). In diesem Fall bestimmen die ersten drei Achsen des Industrieroboters die Position des Handwurzelpunkts und die letzten drei Achsen die Orientierung des Endeffektors und können daher unabhängig voneinander berechnet werden. Bei der Berechnung der inversen Kinematik ist außerdem zu beachten, dass es mehrere Lösungen gibt, da ein Roboter eine bestimmte Lage im Raum in mehreren Konfigurationen erreichen kann. Sofern nicht anders vorgegeben, wird dabei die Konfiguration verwendet, bei der sich die Positionen der einzelnen Roboterachsen von denen der aktuellen Konfiguration am wenigsten unterscheiden.

Zur Bewegung eines Roboters kommen überwiegend die folgenden Bewegungsarten zum Einsatz:

- Bei der **Point-to-point-Bewegung (PTP)** (oft auch *Joint-Bewegung* genannt) bewegen sich alle Roboterachsen von ihrer Ausgangsposition synchron zu ihrer Zielposition. Diese Bewegung ist, sofern sich die notwendigen Achswerte innerhalb der Achslimits befinden, immer möglich, allerdings lässt sich die Bewegung des TCP hierbei nur schwer vorhersagen. Da sich jede Roboterachse auf direktem Weg zur Zielposition bewegt und die langsamste Achse die Geschwindigkeit vorgibt, lässt sich mit dieser Bewegungsart die maximale Geschwindigkeit erreichen. Aus diesem Grund wird bei der Programmierung von Roboterbewegungen zwischen festen Posen, insbesondere beim Überbrücken großer Entfernungen meist die PTP-Bewegung verwendet.
- Bei der **Linearbewegung (LIN)** bewegt sich der Roboter so, dass sich der TCP im kartesischen Raum linear von der Ausgangslage zur Ziellage bewegt. In engen Arbeitsräumen ist dies zu bevorzugen, da sich die Bewegung des Endeffektors genau kontrollieren lässt. Allerdings ist die dadurch erreichte Geschwindigkeit geringer als bei PTP-Bewegungen.

Bei der Bewegung von Knickarmrobotern können sogenannte *Singularitäten* auftreten. Dies sind Roboterstellungen, bei denen die Freiheitsgrade des TCP eingeschränkt sind. Bei klassischen 6-Achs-Knickarmrobotern treten die in der Praxis wichtigen Singularitäten bei einer Strecklage von Achse 3 (und damit am Rand des Arbeitsraums) sowie bei einer Strecklage von Achse 5 (wodurch die Achsen 4 und 6 koaxial werden) auf (Pott et al. 2019). Eine weitere Singularität liegt vor, wenn die Achsen 1 und 6 koaxial sind, aber diese kann bei geeigneter Positionierung des Roboters relativ zum Ladungsträger vernachlässigt werden. Bei einer Linearbewegung durch die Singularitäten im Arbeitsraum muss die Geschwindigkeit des TCP meist stark verlangsamt und die Rotationsgeschwindigkeit der einzelnen Roboterachsen stark erhöht werden. In vielen

Fällen stoppt die Robotersteuerung den Roboter an Singularitäten, weshalb diese Stellungen in der Praxis vermieden werden müssen. Für weitere Details zu industriellen Robotersystemen sei auf Pott et al. (2019) verwiesen.

3 Entwicklung eines Suchverfahrens zur Greifplanung

In diesem Kapitel wird ein Verfahren entwickelt, um Greifvorgänge zu planen, die prozesssicher und insbesondere kollisionsfrei ausgeführt werden können. Dieses Verfahren soll in Roboteranlagen zur Vereinzelung und Zuführung von ungeordnet gelagerten Werkstücken zum Einsatz kommen und somit auf deren Anforderungen zugeschnitten sein. In Abschnitt 3.1 werden zunächst die dabei verwendeten Daten und deren geeignete Repräsentation für die Suche vorgestellt. Da der Suchraum durch einen Baum repräsentiert werden kann, wird für die Suche nach einer Greiflösung ein Baumsuchverfahren eingesetzt. Abbildung 3.1 stellt beispielhaft einen Ausschnitt eines solchen Suchbaums dar. In einem solchen Baumsuchverfahren wird jeder Knoten einerseits evaluiert und andererseits expandiert. Die dafür verwendeten Funktionen werden in Abschnitt 3.2 definiert. Um sicherzustellen, dass die berechneten Lösungen zu prozesssicheren Greifvorgängen führen, werden die Knoten, im Gegensatz zu klassischen Suchverfahren, beim Evaluieren auch auf mögliche Probleme überprüft. Diese Prüfungen werden in Abschnitt 3.3 beschrieben. Für eine effiziente heuristische Suche müssen beim Evaluieren eines Knotens die Kosten zum Erreichen eines Zielknotens geschätzt werden. Auf die Beschreibung einer entsprechenden Heuristikfunk-

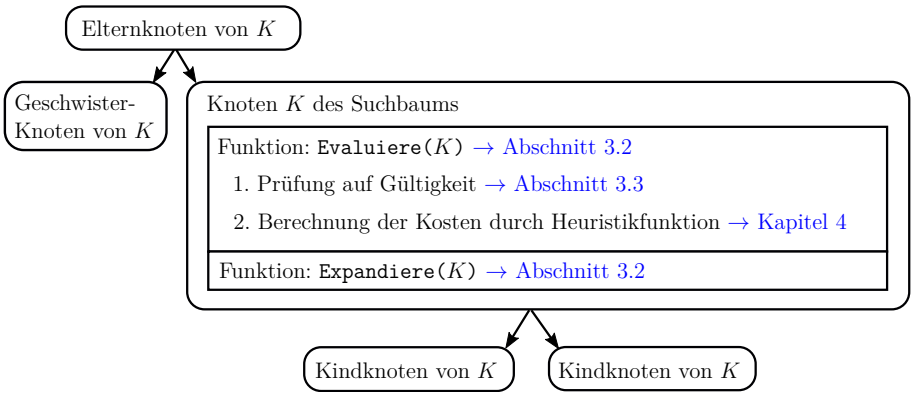


Abbildung 3.1: Schematischer Ausschnitt eines Suchbaums mit Darstellung der Grundfunktionen eines Knotens

tion wird in diesem Kapitel zunächst verzichtet, sondern angenommen, dass eine geeignete Heuristikfunktion existiert. Die Entwicklung einer Heuristikfunktion für eine effiziente Suche wird in Kapitel 4 behandelt. In Abschnitt 3.4 erfolgen erste Untersuchungen basierend auf simulierten Daten sowie eine Einordnung und Bewertung des beschriebenen Suchverfahrens. Einige Ansätze dieses Kapitels orientieren sich an Spenrath et al. (2017a).

3.1 Repräsentation der benötigten Daten

Um Griffe planen zu können, werden einige Daten über den Anwendungsfall benötigt. Diese beinhalten insbesondere die zu vereinzelnden Werkstücke, die Greifer, Ladungsträger und Robotermodelle sowie die in Abschnitt 1.2 eingeführten Greifposen. Es wird angenommen, dass diese Daten durch die Inbetriebnahme bereitstehen und in einer *Wissensbasis* abgelegt sind. Der Inhalt der Wissensbasis wird in Abschnitt 3.1.1 detailliert beschrieben. Zusätzlich zur Wissensbasis, die für eine gegebene Anwendung konstant ist, benötigt die Greifplanung dynamische Daten über den aktuellen Zustand im Ladungsträger, wie beispielsweise die Sensordaten oder die erkannten Werkstücke. Diese Informationen werden in Abschnitt 3.1.2 detailliert beschrieben. Basierend auf diesen Daten, berechnet das Suchverfahren unter Verwendung eines Suchbaums eine Greiflösung. Abbildung 3.2 zeigt eine Übersicht der verwendeten Daten.

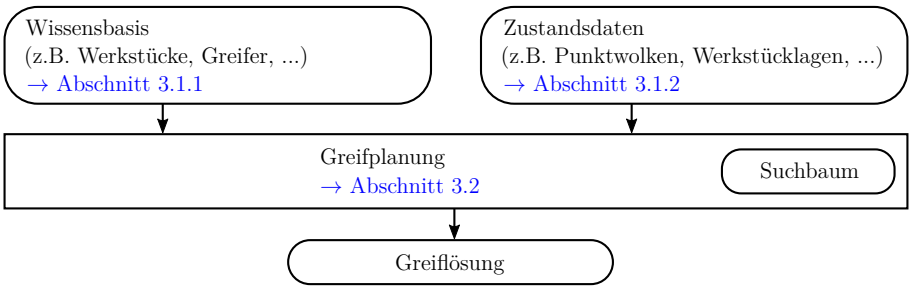


Abbildung 3.2: Übersicht der in der Greifplanung verwendeten Daten

3.1.1 Komponenten der Wissensbasis

Um die Zusammenhänge zwischen den einzelnen Komponenten der Wissensbasis abzubilden, wird die Mehrzahl dieser Komponenten als Knoten in einem gerichteten Graphen (vgl. Abschnitt 2.2.1) abgespeichert. Da die heuristische Suche bei den Werkstücken beginnt, sind dies

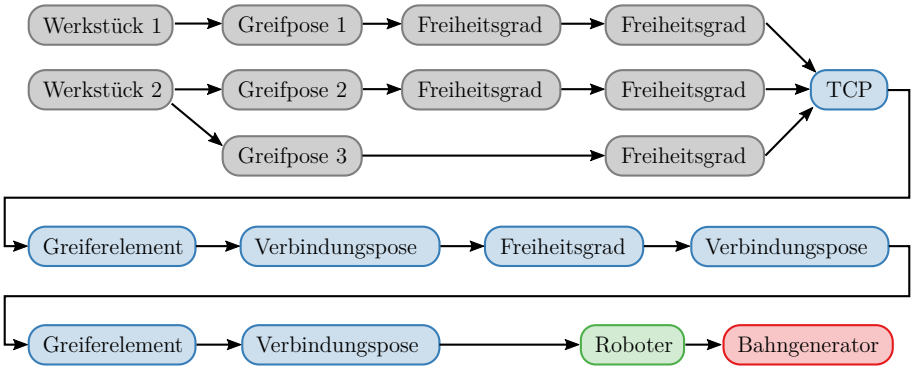


Abbildung 3.3: Beispielhafter gerichteter Graph der Wissensbasis mit den darin enthaltenen Komponenten: zwei Werkstücke mit Greifposen und Freiheitsgraden (in grau dargestellt) und ein Greifer (in blau dargestellt)

die einzigen Knoten innerhalb des gerichteten Graphen, die keinen Vorgänger besitzen. Alle anderen Komponenten sind entsprechend ihrer geometrischen Anordnung sowie der Reihenfolge in der späteren Baumsuche angeordnet. Abbildung 3.3 zeigt ein Beispiel eines solchen Graphen der Wissensbasis. In dieser Wissensbasis sind zwei unterschiedliche Werkstücke enthalten, bei denen die Suche jeweils beginnen kann. Diese Werkstücke verfügen über eine oder mehrere Greifposen mit jeweils einem oder mehreren Freiheitsgraden. Für beide Greifposen wird derselbe TCP desselben Greifers verwendet. Der Greifer besteht aus zwei Greiferelementen mit einer Bewegungsachse zwischen diesen Elementen. Zuletzt sind sowohl ein Roboter als auch ein Bahngenerator in der Wissensbasis enthalten. Durch diese Speicherung können während der Erstellung des Suchbaums sehr schnell die jeweils benötigten Komponenten ausgelesen werden. Der genaue Ablauf wird bei der Beschreibung des Suchverfahrens in Abschnitt 3.2 näher erläutert. Im Folgenden werden die einzelnen Komponenten des gerichteten Graphens beschrieben.

- **Werkstücke:** Die verschiedenen Werkstücktypen, die eine Roboteranlage handhaben muss, werden im Graphen als Werkstückknoten gespeichert. Diese Werkstückknoten beinhalten neben einer Werkstück-ID ein Flächenmodell, welches die Geometrie der Werkstücke beschreibt, sowie die Information, ob das Werkstück Symmetrien besitzt. Um die Lage eines Werkstücks eindeutig beschreiben zu können, wird ein Werkstückkoordinatensystem $\mathcal{K}_W$ definiert. Dieses kann an einer beliebigen Stelle des Werkstücks definiert sein und wird meist in die Nähe des Mittelpunkts oder auf eine Symmetrieachse gelegt. Alle Koordinaten des Flächenmodells beziehen sich ebenfalls auf $\mathcal{K}_W$.

- **Greifposen:** Um zu beschreiben, wie ein Werkstück gegriffen werden kann, werden, wie in Abschnitt 1.2 bereits eingeführt, Greifposen verwendet. Diese werden als Kindknoten des entsprechenden Werkstücks im Graphen der Wissensbasis modelliert. Jedes Werkstück kann theoretisch beliebig viele Greifposen besitzen. Entsprechend der Definition in Abschnitt 1.2 legen die Greifposen fest, wo ein Werkstück gegriffen werden kann. Das Koordinatensystem einer solchen Greifpose wird im Folgenden mit $\mathcal{K}_G$ bezeichnet. Dadurch kann die Lage der Greifpose mittels der Transformation ${}^{W}_{G}\boldsymbol{T}$ in der Greifpose gespeichert werden. Zusätzlich lassen sich in jeder Greifpose eine Priorität zur Festlegung bevorzugter Greifposen sowie weitere für den Roboter notwendige Informationen über den Greifprozess, wie die Art des Griffs oder die gewünschte Greifkraft hinterlegen.

- **Greiferelemente:** In Abschnitt 1.3 wurde festgelegt, dass der in dieser Arbeit entwickelte Ansatz auch Greifer unterstützen soll, die über zusätzliche Bewegungsachsen verfügen. Aus diesem Grund wird der Greifer innerhalb der Wissensbasis durch ein oder mehrere Greiferelemente repräsentiert. Jedes Greiferelement verfügt seinerseits über ein oder mehrere Flächenmodelle, die die Störkontur des entsprechenden Elements darstellen. Das Koordinatensystem eines solchen Greiferelements wird im Folgenden mit $\mathcal{K}_E$ bezeichnet.

- **TCPs:** Um festzulegen, mit welchem Teil des Greifers ein Werkstück gegriffen werden soll, sind an einem Greifer ein oder mehrere TCPs definiert. Diese werden mittels der Transformation ${}^{E}_{T}\boldsymbol{T}$ relativ zu einem festgelegten Greiferelement beschrieben. Die Möglichkeit, mehrere TCPs zu definieren, erlaubt es beispielsweise verschiedene Greifprinzipien an einem Greifer umzusetzen, indem für jedes Greifprinzip ein eigener TCP verwendet wird. Das Koordinatensystem eines TCPs wird im Folgenden mit $\mathcal{K}_T$ bezeichnet.

- **Verbindungsposen:** Wenn ein Greifer aus mehreren Greiferelementen besteht, so muss festgelegt werden, wie diese Greiferelemente miteinander verbunden sind. Dazu können an den einzelnen Greiferelementen entsprechende Verbindungsposen definiert werden. Die Verbindungsposen erlauben es, Verbindungen zwischen zwei Greiferelementen genauso zu behandeln wie beispielsweise zwischen Greifposen und TCPs. Durch die Verwendung der Verbindungsposen kann zudem das Koordinatensystem $\mathcal{K}_E$ eines Greiferelements beliebig relativ zum Flächenmodell, das die Geometrie des Greiferelements beschreibt, gewählt werden und muss beispielsweise nicht am Flansch liegen. Dies ermöglicht einen großen Spielraum bei der praktischen Modellierung eines Greifers. Falls ein Greiferelement über eine Bewegungsachse mit einem anderen Greiferelement verbunden ist, so befindet sich die entsprechende Verbindungspose auf dieser Bewegungsachse. Der Flansch des Greifers, mit dem dieser am Roboter montiert wird, wird ebenfalls als Verbindungspose modelliert. Das Koordinatensystem einer solchen Verbindungspose wird im Folgenden mit $\mathcal{K}_V$ bezeichnet.

- **Freiheitsgrade:** Für Freiheitsgrade gibt es in dem in dieser Arbeit entwickelten Ansatz zwei Anwendungsfälle. Sie können einerseits verwendet werden, um physikalische Freiheitsgrade an einer Greifpose zu repräsentieren. Diese können sich beispielsweise aus einer Rotationssymmetrie des Werkstücks oder einer Eigenschaft des Greifers (als Beispiel sei ein rotatorischer Freiheitsgrad an einem kreisförmigen Sauggreifer genannt) ergeben. Auch ein translatorischer Freiheitsgrad entlang einer greifbaren Kante oder Fläche lässt sich dadurch beschreiben. Dabei können mehrere Freiheitsgrade pro Greifpose oder TCP definiert werden. Den zweiten Anwendungsfall von Freiheitsgraden stellen die in Abschnitt 1.3 erwähnten zusätzlichen Bewegungsachsen im Greifer dar. Bei beiden Anwendungsfällen sind sowohl translatorische als auch rotatorische und diskrete sowie kontinuierliche Freiheitsgrade möglich. Jeder Freiheitsgrad ist durch seine Bewegungsrichtung (bzw. Achse) $\boldsymbol{a} \in \mathbb{R}^3$ sowie die minimal und maximal zulässigen Werte (bzw. Winkel) $\alpha_{min} \in \mathbb{R}$ bzw. $\alpha_{max} \in \mathbb{R}$ mit $\alpha_{min} < \alpha_{max}$ definiert. Bei diskreten Freiheitsgraden können zudem die konkreten Werte oder alternativ die Anzahl $N_{dof} \in \mathbb{N}$ der Schritte hinterlegt werden. Eine Beschreibung der Berechnung der jeweiligen Transformationsmatrix basierend auf den verschiedenen Freiheitsgraden erfolgt in Abschnitt 3.2.4.

- **Roboter:** Die Eigenschaften des zu verwendenden Industrieroboters werden ebenfalls in der Wissensbasis hinterlegt. Dazu gehörigen insbesondere die Flächenmodelle, die die Geometrie der einzelnen Elemente des Roboters beschreiben, sowie die Parameter, die die Roboterkinematik definieren. In dieser Arbeit wird ohne Beschränkung der Allgemeinheit davon ausgegangen, dass der Roboter stets im Ursprung des Weltkoordinatensystems $\mathcal{K}_0$ steht. Die Wissensbasis kann prinzipiell beliebig viele Roboter beinhalten, wobei für eine konkrete Anwendung, üblicherweise einer dieser Roboter ausgewählt und dadurch mit dem letzten Greiferelement verbunden wird. Geometrisch beschreibt diese Komponente die Lage des Handflansches des Roboters während des Greifvorgangs, dessen Koordinatensystem im Folgenden mit $\mathcal{K}_F$ bezeichnet wird.

- **Bahngenerator:** Um die Anwendung auf unterschiedliche Randbedingungen anpassen zu können, wird in der Praxis oft gewünscht, die Bewegungsbahn des Roboters durch Parameter beeinflussen zu können. Diese Parameter, die die zu generierende Bewegungsbahn definieren, werden innerhalb eines Bahngeneratorknotens gespeichert. Der Bahngeneratorknoten ist im gerichteten Graphen Nachfolger des Roboterknotens. Die einzelnen Stützstellen der Bewegungsbahn sind nicht Bestandteil dieses Graphen, sondern werden als Ergebnis generiert. Die Anzahl dieser Stützstellen ist dadurch variabel.

Die meisten der oben genannten Komponenten besitzen eine Transformation, die beschreibt, wie die jeweilige Komponenten bezüglich einer anderen Komponente räumlich positioniert ist. Durch

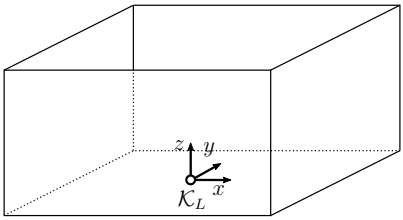


Abbildung 3.4: Visualisierung des Koordinatensystems $\mathcal{K}_L$ des Ladungsträgers

die einheitliche Darstellung in einem gerichteten Graphen lassen sich die Transformationen verallgemeinern, indem, abgesehen von den Werkstücken, jeder Knoten K_i eine Transformation ${}^{K_{i-1}}_{K_i}\boldsymbol{T}$ enthält, die seine Lage relativ zu seinem Vorgänger K_{i-1} beschreibt. In den Ausnahmefällen, in denen ein Knoten des Graphen mehrere Vorgänger besitzt, wird die Transformationsmatrix ${}^{K_{i-1}}_{K_i}\boldsymbol{T}$ als Einheitsmatrix festgelegt. Für eine Beschreibung der Berechnung der Lagen der einzelnen Komponenten während der Suche sei auf Abschnitt 3.2.2 verwiesen. Für die weitere Arbeit wird davon ausgegangen, dass Greifposen und TCPs so definiert werden, dass sich ein Werkstück greifen lässt, indem der Greifer so bewegt wird, dass das TCP-Koordinatensystem des Greifers deckungsgleich mit dem Koordinatensystem der am Werkstück definierten Greifpose ist. Zum Zeitpunkt des Greifens gilt somit

$$\mathcal{K}_T = \mathcal{K}_G\,. \tag{3.1}$$

Zusätzlich zu den oben genannten Komponenten innerhalb des gerichteten Graphen sind die verschiedenen Typen von Ladungsträgern in der Wissensbasis hinterlegt. Diese beinhalten die Abmessungen der Ladungsträger sowie einen optionalen Sicherheitsabstand zu den Seitenwänden, der während des Entnahmevorgangs eingehalten werden soll. Ohne Beschränkung der Allgemeinheit wird im Folgenden davon ausgegangen, dass das Koordinatensystem des Ladungsträgers, das in dieser Arbeit mit $\mathcal{K}_L$ bezeichnet wird, in der Mitte des Bodens des Ladungsträgers definiert ist. Abbildung 3.4 visualisiert das Koordinatensystem des Ladungsträgers beispielhaft. Die Ladungsträger befinden sich jedoch nicht innerhalb des gerichteten Graphen, da sie geometrisch nicht mit den darin enthaltenen Komponenten verbunden sind.

3.1.2 Zustandsdaten

Neben den konfigurierten Daten der Wissensbasis werden einige Informationen über den aktuellen Zustand innerhalb des Ladungsträgers benötigt, um einen geeigneten Greifvorgang planen

zu können. Diese ändern sich im Allgemeinen für jeden Entnahmevorgang und werden im Folgenden *Zustandsdaten* genannt. Die in den Zustandsdaten enthaltenen Informationen werden im Folgenden aufgeführt und beschrieben:

- **Punktwolke:** Die Punktwolke besteht aus diskreten Messpunkten auf Oberflächen der Werkstücke und der Kiste. Sie repräsentiert die aktuelle Situation innerhalb des Ladungsträgers und wird durch den Sensor im Sensorkoordinatensystem $\mathcal{K}_S$ erzeugt. Durch eine Hand-Auge-Kalibrierung (siehe Abschnitt 2.4.2) wird die Punktwolke in das Weltkoordinatensystem $\mathcal{K}_0$ transformiert, wobei der Ursprung des Sensorkoordinatensystems $\mathcal{K}_S$ und damit die Position des Sensors bezogen auf das Weltkoordinatensystem für eine spätere Verwendung vorgehalten wird.

- **Erkannte Werkstücke:** Ein Algorithmus zur Objektlageerkennung stellt Informationen über die erkannten Werkstücke für die Greifplanung bereit. Diese bestehen aus einer Transformation ${}^{0}_{W}\boldsymbol{T}$, welche die Lage des jeweiligen Werkstückkoordinatensystems $\mathcal{K}_W$ im Weltkoordinatensystem angibt, sowie dem Werkstücktyp (beschrieben durch eine eindeutige ID). Zusätzlich wird davon ausgegangen, dass für jede erkannte Werkstücklage ein Konfidenzwert $R_l \in [0, 1]$ zur Verfügung steht, der die Wahrscheinlichkeit der korrekten Lagebestimmung angibt.

- **Erkannter Ladungsträger:** Die Lage des Ladungsträgers im Weltkoordinatensystem wird ebenfalls durch den Algorithmus zur Objektlageerkennung bereitgestellt und wird durch die Transformation ${}^{0}_{L}\boldsymbol{T}$ repräsentiert. Weiterhin ist der Typ des Ladungsträgers durch eine eindeutige ID bekannt.

In diesem Abschnitt wurden die in der Wissensbasis enthaltenen Komponenten beschrieben, die vorab definiert werden, wie zum Beispiel die zu vereinzelnden Werkstücke oder die zu verwendenden Greifer. Weiterhin wurden die Zustandsdaten beschrieben, die für jeden Entnahmevorgang erfasst beziehungsweise bestimmt werden. Von besonderer Bedeutung sind dabei die Punktwolke sowie die erkannten Werkstücklagen. Beide Kategorien von Daten stehen für die Greifplanung zur Verfügung. Im folgenden Abschnitt wird ein heuristisches Suchverfahren entwickelt, das in der Lage ist, basierend auf diesen Daten eine geeignete Greiflösung zu bestimmen.

3.2 Heuristisches Suchverfahren zur Greifplanung

Die in dieser Arbeit entwickelte Greifplanung basiert auf einem heuristischen Suchverfahren. Das Suchverfahren wird in diesem Abschnitt beschrieben. Dabei wird zunächst der grundlegende

Ablauf beschrieben und im Anschluss auf verschiedene Details des Suchverfahrens, wie die Umsetzung der Freiheitsgrade oder die Integration der Bahnplanung, eingegangen.

3.2.1 Konzept des Suchverfahrens

Um einen möglichst robusten Entnahmeprozess zu erreichen, soll das Suchverfahren so ausgelegt werden, dass jede gefundene Lösung geeignet ist, ein Werkstück aus dem Ladungsträger zu entnehmen. Auf die Suche nach einer „optimalen“ Lösung, beispielsweise einer Lösung mit möglichst kurzer Roboterbahn, soll aufgrund des großen Suchraums und der Anforderung der kurzen Rechenzeit in dieser Arbeit verzichtet werden. Daher kommt für diese Problemstellung, anstatt eines optimalen Suchverfahrens, die Bestensuche (siehe Abschnitt 2.2.4) zum Einsatz. Diese basiert auf einem Suchbaum, der inkrementell aufgebaut und dabei durchsucht wird. Bei der konventionellen Baumsuche werden die Kosten den Kanten des Suchbaums zugeordnet. Es handelt sich daher um einen kantengewichteten Graphen (vgl. Abschnitt 2.2.1). Um die folgenden Ausführungen zu vereinfachen, werden die Kosten hier hingegen dem jeweils nachfolgenden Knoten zugeordnet. Da in einem gerichteten Baum jeder Knoten (abgesehen vom Wurzelknoten) nur eine eingehende Kante besitzt, besteht hier eine eindeutige Zuordnung, so dass die Theorie der Baumsuchverfahren weiterhin zutrifft.

Der Suchbaum beinhaltet konkrete Instanzen der in der Wissensbasis hinterlegten Komponenten und beginnt, unterhalb seines Wurzelknotens, mit den Instanzen der Werkstücke. Diese Werkstückinstanzen werden auf Basis der erkannten Werkstücklagen beim Expandieren des Wurzelknotens generiert. Der in der Wissensbasis hinterlegte gerichtete Graph ermöglicht es nun, diesen Suchbaum schrittweise zu expandieren, indem jeweils die Nachfolger der in diesem Graph gespeicherten Komponenten hinzugefügt werden. Obwohl alle Knoten des Suchbaums (mit Ausnahme des Wurzelknotens) Instanzen der Komponenten der Wissensbasis sind, wird im Folgenden meist auf den Begriff der Instanz verzichtet und beispielsweise die Knoten der Werkstückinstanzen der Einfachheit halber als Werkstückknoten bezeichnet. Der grundlegende Ablauf des Suchverfahrens mit den Abhängigkeiten auf die Wissensbasis ist in Abbildung 3.5 beispielhaft dargestellt.

Für den Ablauf der Bestensuche wird eine Vorrangwarteschlange Q verwendet, in der die noch nicht expandierten Knoten des Suchbaums gespeichert sind. Diese Vorrangwarteschlange ist als sortierte Liste realisiert, wobei die durch die Heuristikfunktion abgeschätzten Kosten zur Erreichung eines Zielknotens als Kriterium für die Sortierung verwendet werden. Das Element mit den geringsten geschätzten Kosten befindet sich stets an erster Stelle der Vorrangwarteschlange und wird demnach in einem nächsten Schritt der Suche zuerst entnommen. Der grundlegende

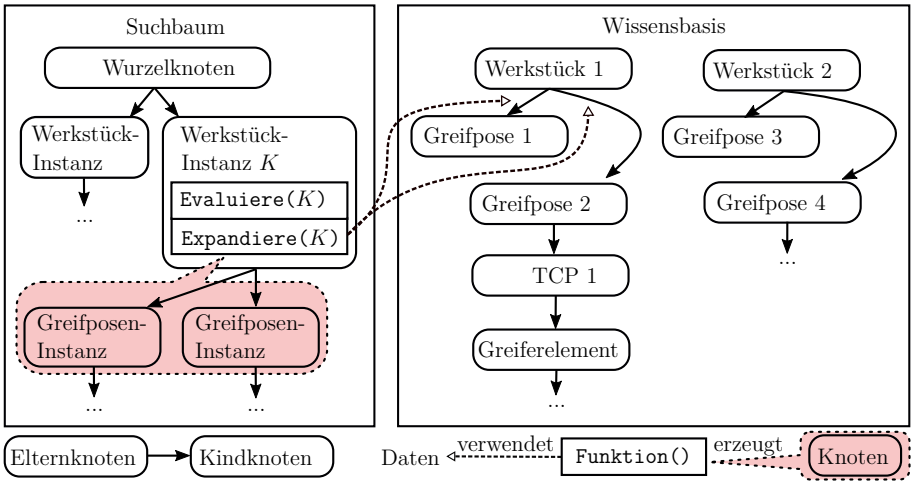


Abbildung 3.5: Vereinfachtes schematisches Beispiel eines Suchbaums und der Wissensbasis. Dabei ist hervorgehoben, welche Daten die Funktion `Expandiere` verwendet und welche Knoten dadurch erzeugt werden.

Ablauf des Suchverfahrens ist in Abbildung 3.6 mittels Pseudo-Code dargestellt und läuft folgendermaßen ab:

Zu Beginn der Suche wird der Wurzelknoten in die Vorrangwarteschlage eingefügt (Zeile 1 in Abbildung 3.6). Im Anschluss wird wiederholt der erste Knoten K_i der Vorrangwarteschlange Q entnommen und expandiert. Dieser Ablauf wird solange wiederholt, bis ein Zielknoten gefunden wurde, der eine gültige Greiflösung repräsentiert oder eine maximale Rechenzeit von t_{max} verstrichen ist. Der Abbruch der Suche nach einer definierten maximalen Rechenzeit ist sinnvoll, um in Situationen, in denen keine Lösung gefunden werden kann, nicht zu viel Zeit durch eine aussichtslose Suche zu verlieren. Stattdessen kann beispielsweise versucht werden, durch die Objektlageerkennung weitere Werkstücke zu erkennen oder es kann eine erneute Szenenerfassung durch die Sensorik durchgeführt werden.

Das Expandieren der in der Vorrangwarteschlange enthaltenen Knoten erfolgt durch die Funktion `Expandiere`(K_i), die ab Zeile 6 in Abbildung 3.6 mittels Pseudocode dargestellt ist. Die Funktion erzeugt in Zeile 7 zunächst auf Basis der Wissensbasis alle Kindknoten von K_i. Da die Zusammenhänge zwischen den verschiedenen Komponenten durch den in der Wissensbasis hinterlegten Graphen definiert sind, kann die Erzeugung der Kindknoten generisch erfolgen, ohne dass der Typ der jeweiligen Komponente berücksichtigt werden muss. Einige Komponenten besitzen jedoch ein spezielles Verhalten beim Expandieren. Dies ist einerseits der Wurzelknoten.

```
   Daten : Vorrangwarteschlange Q
   Ergebnis : Gültige Greiflösung
1  Füge Wurzelknoten in Vorrangwarteschlange Q ein;
2  solange keine Lösung gefunden und max. Rechenzeit t_max nicht erreicht wiederhole
3  |  K_i ← Entnehme das erste Element von Q;
4  |  Expandiere(K_i);
5  Ende

6  Funktion Expandiere(K_i):
7  |  Erzeuge alle Kindknoten von K_i;
8  |  für jeden Kindknoten K_j führe aus
9  |  |  Berechne ⁰_{K_j}T;
10 |  |  Setze den Vorgängerzeiger von K_j auf K_i;
11 |  |  wenn K_j ist ein Zielknoten dann
12 |  |  |  Brich ab und liefere Ergebnis K_j;
13 |  |  sonst
14 |  |  |  h ← Evaluiere(K_j);
15 |  |  |  wenn K_j gültig dann
16 |  |  |  |  Sortiere den Knoten K_j in Q ein (mit h als Priorität);
17 |  |  |  Ende
18 |  |  Ende
19 |  Ende
20 zurück
```

Abbildung 3.6: Ablauf des Suchverfahrens mittels Bestensuche

Beim Expandieren dieses Knotens wird für jede in den Zustandsdaten enthaltene Werkstücklage ein Werkstückknoten in den Suchbaum eingefügt. Dabei entspricht die Lage der erzeugten Werkstückknoten der jeweiligen erkannten Werkstücklage. Andererseits sind dies Knoten, die für Freiheitsgrade und für die Bahnplanung verwendet werden. Diese werden in den Abschnitten 3.2.4 bzw. 3.2.5 näher beschrieben. Nach dem Erzeugen der Kindknoten wird in Zeile 9 für jeden erzeugten Knoten K_j die Transformation ${}^{0}_{K_j}\boldsymbol{T}$ berechnet, welche die Lage von K_j relativ zum Weltkoordinatensystem $\mathcal{K}_0$ beschreibt. Dies erfolgt unter Verwendung der bekannten Transformation ${}^{0}_{K_i}\boldsymbol{T}$ sowie der in der Wissensbasis enthaltenen Transformation ${}^{K_i}_{K_j}\boldsymbol{T}$ und wird in Abschnitt 3.2.2 im Detail beschrieben. Weiterhin wird K_j in den Suchbaum eingefügt und der entsprechende Vorgängerzeiger von K_j auf K_i gesetzt. Falls es sich bei K_j um einen Zielknoten handelt, muss dieser nicht weiter expandiert werden, da er bereits eine gesuchte Greiflösung darstellt. In diesem Fall kann die Suche beendet werden. Andernfalls werden die geschätzten Kosten zur Erreichung eines Zielknotens berechnet.

Die Berechnung der geschätzten Kosten eines Knotens K_j erfolgt in Zeile 14 durch die Funktion `Evaluiere(`K_j`)`. Diese Funktion berechnet die Kosten h des Knotens K_j (die wie zuvor erwähnt,

auch der vorhergehenden Kante zugeordnet werden können) mittels einer Heuristikfunktion (vgl. Abschnitt 2.2.2). In diesem Kapitel soll zunächst lediglich angenommen werden, dass eine solche geeignete Heuristikfunktion existiert. Dabei wird die Annahme getroffen, dass die durch diese Heuristikfunktion für einen Knoten K_j geschätzten Kosten mit dem Abstand des Knotens K_j zu einem Zielknoten korrelieren. Da es sich nicht um ein optimales Suchverfahren handelt, darf die Heuristikfunktion die Kosten überschätzen. Wie eine solche Heuristikfunktion realisiert werden kann, wird in Kapitel 4 im Detail betrachtet. Darüber hinaus dient die Funktion `Evaluiere(`K_j`)` dazu, mögliche Probleme beim Greifen zu vermeiden, indem sie bestimmte Eigenschaften des Knotens K_j prüft und diesen dadurch für ungültig erklären kann. In Abschnitt 3.3 wird näher darauf eingegangen, wie diese Prüfungen für verschiedene Komponenten umgesetzt werden können. Sofern der evaluierte Knoten gültig ist, wird er entsprechend seiner geschätzten Kosten h in die Vorrangwarteschlange Q einsortiert (Zeile 16).

3.2.2 Berechnung der Lagen der einzelnen Komponenten

Da jeder Knoten der Wissensbasis eine Transformation ${}^{K_{i-1}}_{K_i}\boldsymbol{T}$ enthält, die seine Lage relativ zur Lage seines Vorgängerknotens angibt, lässt sich die Lage eines neu in den Suchbaum einzufügenden Knotens K_i im Weltkoordinatensystem $\mathcal{K}_0$ mittels

$$ {}^{0}_{K_i}\boldsymbol{T} = {}^{0}_{K_{i-1}}\boldsymbol{T} \cdot {}^{K_{i-1}}_{K_i}\boldsymbol{T} \tag{3.2} $$

berechnen. Diese Berechnung kann bei der Erzeugung eines neuen Knotens unabhängig davon erfolgen, um was für eine Komponente es sich handelt. Abbildung 3.7 visualisiert den daraus resultierenden Zusammenhang zwischen den einzelnen Koordinatensystemen und Transformationen am Beispiel eines Werkstücks mit einer Greifpose und einem Greifer mit einer zusätzlichen Bewegungsachse. Die Transformation ${}^{0}_{W}\boldsymbol{T}$, die die Lage des Werkstückkoordinatensystems $\mathcal{K}_W$ relativ zum Weltkoordinatensystem $\mathcal{K}_0$ beschreibt, wird durch ein Verfahren zur Objektlageerkennung bestimmt. Die Transformation ${}^{W}_{G}\boldsymbol{T}$, die die Lage der Greifpose relativ zu $\mathcal{K}_W$ definiert, ist in der Wissensbasis hinterlegt. Ebenso sind alle Transformationen der Greiferelemente und der TCPs in der Wissensbasis gespeichert. Um ein Werkstück greifen zu können, wird das Koordinatensystem $\mathcal{K}_T$ des am Greifer definierten TCPs mit dem Koordinatensystem $\mathcal{K}_G$ der am Werkstück definierten Greifpose gleichgesetzt. Auf die Darstellung der Koordinatensysteme $\mathcal{K}_V$ der Verbindungsposen wird in Abbildung 3.7 aufgrund der Übersichtlichkeit verzichtet.

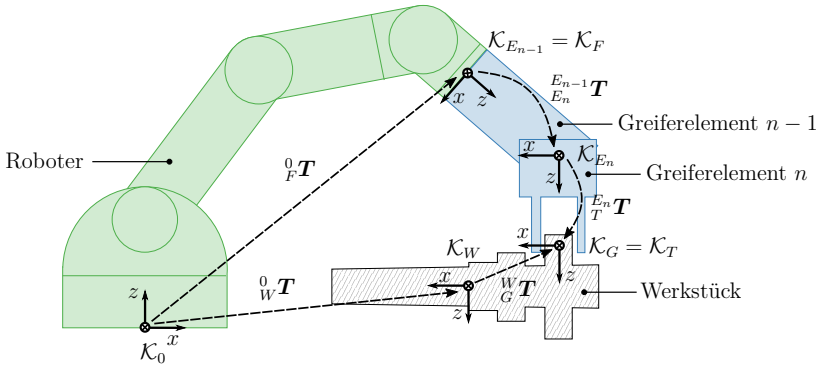


Abbildung 3.7: Übersicht der wichtigsten Koordinatensysteme $\mathcal{K}_i$ und Transformationen $\boldsymbol{T}$

3.2.3 Bevorzugte Orientierung von Komponenten

Bei einigen Komponenten existieren physikalische Einschränkungen bezüglich ihrer Orientierung. Beispielsweise ist es naheliegend, dass ein Greifer ein Werkstück nicht von unten und damit durch den Kistenboden greifen kann. Solche Orientierungen können somit in der Suche frühzeitig ausgeschlossen werden. Zudem können bestimmte Orientierungen einzelner Komponenten mit höherer Wahrscheinlichkeit zu einer zulässigen Greiflösung führen. Indem diese Orientierungen bevorzugt überprüft werden, kann die Suche somit zielgerichteter durchgeführt werden. Zur Definition dieser bevorzugten Orientierungen seien, ohne Beschränkung der Allgemeinheit, alle Koordinatensysteme des Greifers so definiert, dass die z-Achse aus dem Handflansch des Roboters heraus zeigt (vergleiche auch die Orientierungen der Koordinatensysteme in Abbildung 3.7). Weiterhin sei der Vektor

$$\boldsymbol{n}_K = -\boldsymbol{z}_K \tag{3.3}$$

definiert, der in die entgegengesetzte Richtung der z-Achse $\boldsymbol{z}_K$ einer Komponente zeigt. Da der Roboter im Wesentlichen von oben in die Kiste greift, ist somit zu erwarten, dass dieser Vektor $\boldsymbol{n}_K$ bei den zulässigen Griffen ungefähr in die Richtung der z-Achse des Koordinatensystems $\mathcal{K}_L$ des Ladungsträgers zeigt. Um mit dem Greifer oder dem Roboter nicht gegen die Wände des Ladungsträgers zu stoßen, ist es zudem vorteilhaft, wenn der obere Teil des Greifers leicht in Richtung der Mitte des Ladungsträgers geneigt ist. Es sei daher eine bevorzugte Orientierung durch den Vektor $\boldsymbol{b}$ definiert, der vom Vektor $(0, 0, 1)^{\mathrm{T}}$ in $\mathcal{K}_L$ ausgehend um den Winkel β in Richtung der Mitte des Ladungsträgers geneigt ist. Der Vektor $\boldsymbol{b}$ kann nun verwendet werden, um $\boldsymbol{n}_K$ und damit auch die z-Achse des Koordinatensystems einer Komponente auszurichten. Abbildung 3.8 visualisiert dieses Konzept. Um die Komponenten im mittleren Bereich der Kiste nicht so stark zu neigen und dadurch den Abstand zur gegenüberliegenden Seite des

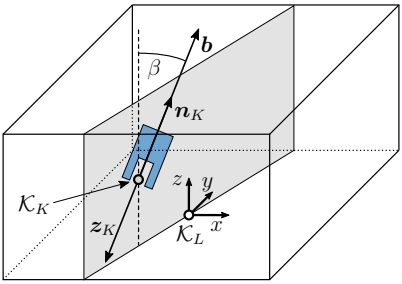


Abbildung 3.8: Bevorzugte Orientierung von Komponenten

Ladungsträgers zu stark zu verringern, wird der Winkel β vom Abstand der betrachteten Komponente zur nächstgelegenen Seitenwand des Ladungsträgers abhängig gemacht. Dafür sei ${}^{L}x_K$ die x-Koordinate und ${}^{L}y_K$ die y-Koordinate des Koordinatensystems der Komponente im Bezug auf das Ladungsträgerkoordinatensystem $\mathcal{K}_L$ und weiterhin d_x^L die Länge des Ladungsträgers in x-Richtung und d_y^L die Breite des Ladungsträgers in y-Richtung. Dann sei der Winkel

$$\beta = \beta_{max} \cdot \left(1 - \frac{2 \cdot d_K^{wand}}{\min\left(d_x^L, d_y^L\right)}\right) \tag{3.4}$$

mit dem Abstand

$$d_K^{wand} = \min\left(\frac{d_x^L}{2} - {}^{L}x_K, \frac{d_x^L}{2} + {}^{L}x_K, \frac{d_y^L}{2} - {}^{L}y_K, \frac{d_y^L}{2} + {}^{L}y_K\right) \tag{3.5}$$

der betrachteten Komponenten von der nächstgelegenen Seitenwand des Ladungsträgers definiert. Diese bevorzugte Orientierung kann somit verwendet werden, um Komponenten geeignet auszurichten. Um später große Abweichungen von dieser bevorzugten Orientierung $\boldsymbol{b}$ ausschließen zu können, sei weiterhin der Winkel

$$\gamma = \arccos\left(\frac{\langle \boldsymbol{b}, \boldsymbol{n}_K \rangle}{\|\boldsymbol{b}\| \cdot \|\boldsymbol{n}_K\|}\right) \in [0, \pi] \tag{3.6}$$

zwischen $\boldsymbol{b}$ und $\boldsymbol{n}_K$ definiert.

3.2.4 Freiheitsgrade

Beim Expandieren eines Baumknotens, der einen Freiheitsgrad repräsentiert, werden mehrere Kindknoten im Suchbaum erstellt. Diese Kindknoten repräsentieren Instanzen der Komponente,

die im gerichteten Graphen der Wissensbasis dem Freiheitsgrad folgt. Die Lage dieser neu erstellten Kindknoten wird dabei ausgehend von der Lage des Freiheitsgradknotens entsprechend der Parameter des Freiheitsgrades modifiziert. Dazu wird die Transformationsmatrix ${}^{K_i}_{K_{i+1}}\boldsymbol{T}$, die die Lage des neu erstellten Knotens relativ zu seinem Elternknoten beschreibt, im Gegensatz zu den meisten anderen Komponenten hier zur Laufzeit berechnet. Dabei können sowohl translatorische als auch rotatorische Freiheitsgrade zum Einsatz kommen, die beide von einem einzelnen Skalar α abhängen:

- Bei einem translatorischen Freiheitsgrad mit einem translatorischen Versatz α entlang des Einheitsvektors $\boldsymbol{a} = (\boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{a}_z)^{\mathrm{T}}$ lässt sich die Transformationsmatrix ${}^{K_i}_{K_{i+1}}\boldsymbol{T}$ mittels

$$ {}^{K_i}_{K_{i+1}}\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & \alpha\boldsymbol{a}_x \\ 0 & 1 & 0 & \alpha\boldsymbol{a}_y \\ 0 & 0 & 1 & \alpha\boldsymbol{a}_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} I_3 & \alpha\boldsymbol{a} \\ 0 & 1 \end{pmatrix} \tag{3.7} $$

  berechnen.

- Bei einem rotatorischen Freiheitsgrad mit Winkel α um die durch den Einheitsvektor $\boldsymbol{a} = (\boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{a}_z)^{\mathrm{T}}$ beschriebene Achse lässt sich die Transformationsmatrix ${}^{K_i}_{K_{i+1}}\boldsymbol{T}$ mittels

$$ {}^{K_i}_{K_{i+1}}\boldsymbol{T} = \begin{pmatrix} \boldsymbol{a}_x^2\Lambda + \cos\alpha & \boldsymbol{a}_x\boldsymbol{a}_y\Lambda - \boldsymbol{a}_z\sin\alpha & \boldsymbol{a}_x\boldsymbol{a}_z\Lambda + \boldsymbol{a}_y\sin\alpha & 0 \\ \boldsymbol{a}_y\boldsymbol{a}_x\Lambda + \boldsymbol{a}_z\sin\alpha & \boldsymbol{a}_y^2\Lambda + \cos\alpha & \boldsymbol{a}_y\boldsymbol{a}_z\Lambda - \boldsymbol{a}_x\sin\alpha & 0 \\ \boldsymbol{a}_z\boldsymbol{a}_x\Lambda - \boldsymbol{a}_y\sin\alpha & \boldsymbol{a}_z\boldsymbol{a}_y\Lambda + \boldsymbol{a}_x\sin\alpha & \boldsymbol{a}_z^2\Lambda + \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8} $$

  mit $\Lambda = (1 - \cos\alpha)$ berechnen.

In dieser Arbeit wird zwischen diskreten und kontinuierlichen Freiheitsgraden unterschieden, die ein unterschiedliches Verhalten beim Expandieren aufweisen:

- Bei diskreten Freiheitsgraden ist die Anzahl N_{dof} an gewünschten Schritten in der Wissensbasis hinterlegt. Somit werden bei der Expansion diese N_{dof} Schritte des Freiheitsgrades zwischen den minimal und maximal erlaubten Werten α_{min} bzw. α_{max} berechnet und die entsprechenden Knoten erzeugt.

- Bei kontinuierlichen Freiheitsgraden wird mehrstufig vorgegangen, um nicht unnötigerweise eine große Anzahl an Knoten im Suchbaum zu erzeugen. Im ersten Schritt wird zunächst nur der Wert für den Freiheitsgrad untersucht, der innerhalb der minimal und maximal erlaubten Werte α_{min} bzw. α_{max} liegt und der dazu führt, dass die Orientierung des nachfolgenden Elements möglichst nah an seiner bevorzugten Orientierung (siehe Abschnitt 3.2.3)

liegt. Im Anschluss wird der Bereich zwischen α_{min} und α_{max} zunächst mit einer festen Anzahl N_{dof} an Schritten diskretisiert und in mehreren Stufen immer weiter verfeinert. Dennoch kann es Situationen geben, in denen keine geeigneten Greiflösungen möglich sind und in denen nach einer ausreichenden Verfeinerung abgebrochen werden soll. Daher wird für diese Diskretisierung eine maximale Anzahl N_{dof}^{max} an Schritten definiert, nach der keine weiteren Expansionen mehr stattfinden. Alternativ kann eine Genauigkeit ϵ definiert werden, die den maximal erlaubten Abstand zwischen zwei benachbarten Schritten festlegt. In diesem Fall lässt sich N_{dof}^{max} mittels

$$N_{dof}^{max} = \left\lceil \frac{\alpha_{max} - \alpha_{min}}{\epsilon} \right\rceil \tag{3.9}$$

berechnen. Wie sich dieses Expandieren eines Freiheitsgrades in mehreren Schritten im Rahmen der Baumsuche umsetzen lässt, wird im Folgenden im Detail beschrieben.

3.2.4.1 Mehrfaches Expandieren von Freiheitsgraden

Wie zuvor beschrieben, ist es insbesondere bei kontinuierlichen Freiheitsgraden sinnvoll, Knoten mehrfach zu expandieren. Dies kann im Rahmen der Suchstrategie, wie in Abbildung 3.9(a) gezeigt und im Folgenden beschrieben, umgesetzt werden. Bei jeder Expansion des Freiheitsgradknotens (abgesehen von der letzten) wird, zusätzlich zu den erzeugten Kindknoten, der Freiheitsgradknoten selbst ebenfalls wieder in die Vorrangwarteschlange eingefügt. Sofern die Heuristikfunktion sicherstellt, dass die Kosten der neu erzeugten Kindknoten geringer sind, als die des Freiheitsgradknotens, so werden diese untersucht, bevor der Freiheitsgradknoten erneut expandiert wird. Weiterhin sollte die Anzahl der Expansionen des mehrfach zu expandierenden Knotens in der Heuristikfunktion berücksichtigt werden, so dass die Kosten dieses Knotens steigen, je öfter er expandiert wird, da die Wahrscheinlichkeit sinkt, unter den Kindern dieses Freiheitsgrades noch eine gültige Lösung zu finden. Dies kann beispielsweise durch einen Kostenwert

$$c_{exp} = \frac{N_{exp}^{dof}}{N_{exp}^{max}} \tag{3.10}$$

realisiert werden, wobei N_{exp}^{dof} die Anzahl der bereits durchgeführten Expansionen und N_{exp}^{max} die maximale Anzahl an möglichen Expansionen eines Freiheitsgrades beschreiben. Eine alternative Möglichkeit, dieses Verhalten zu realisieren, ist das Einfügen einer zusätzlichen Ebene im Suchbaum, wie in Abbildung 3.9(b) dargestelt. Dadurch wird beispielsweise ein Knoten, der einen Freiheitsgrad beschreibt, beim Expandieren mehrere Knoten erzeugen, die verschiedene Diskretisierungsstufen darstellen und die die tatsächlichen Nachfolgerknoten (laut gerichtetem Graphen der Wissensbasis) erzeugen. Über die Berücksichtigung eines Kostenwertes c_{exp} in deren

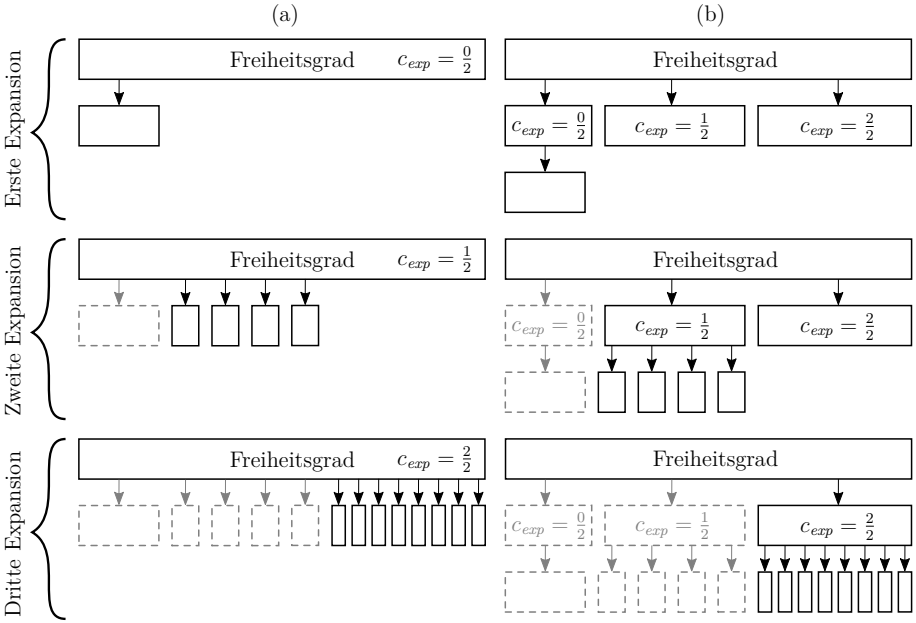


Abbildung 3.9: Konzept des mehrfachen Expandierens von Knoten (a) und alternative Repräsentation mittels einer Zwischenebene (b)

Heuristikfunktion könnte hier ebenfalls die gewünschte Expansionsreihenfolge sichergestellt werden. Um den Suchbaum einfach zu halten, wurde in dieser Arbeit allerdings das mehrfache Expandieren von Knoten bevorzugt. Neben den Freiheitsgraden lässt sich auch die Bahnplanung durch den Suchbaum abbilden. Wie dies realisiert werden kann, wird im folgenden Abschnitt beschrieben.

3.2.5 Integration der Bahnplanung

Bei der Vereinzelung von Werkstücken lässt sich die Roboterbewegung in eine Anfahrbahn und eine Entnahmebahn unterteilen. Beide Bewegungsbahnen werden über eine diskrete Anzahl an Posen des Roboterflanschs als Stützstellen definiert, die im Folgenden als *Bahnposen* bezeichnet werden. Die Anzahl der Bahnposen der Anfahrbahn wird mit N_A und die Anzahl der Bahnposen für die Entnahmebahn mit N_E bezeichnet. Die Anfahrbahn, die somit aus den Bahnposen $P_1^A, \ldots, P_{N_A}^A$ besteht, führt den Greifer zum Werkstück. Die letzte Bahnpose $P_{N_A}^A$ liegt kurz vor der Lage, an der der TCP mit der am Werkstück definierten Greifpose übereinstimmt und

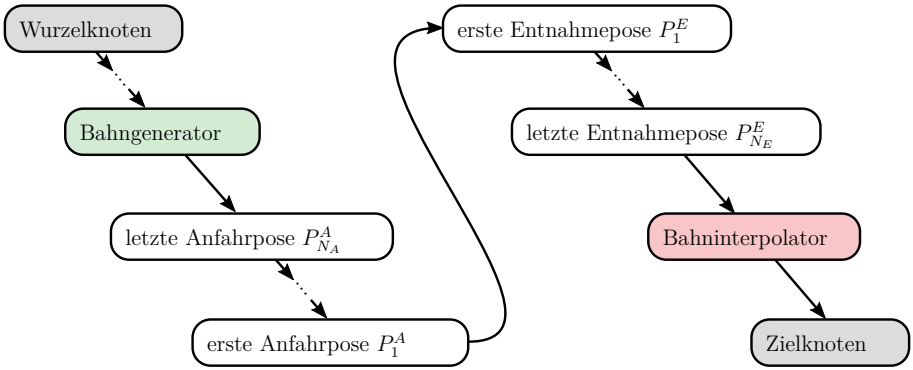


Abbildung 3.10: Reihenfolge der Bahnposen im Suchbaum

das Werkstück somit gegriffen werden kann. Nachdem das Werkstück gegriffen wurde, wird der Greifer auf der aus den Bahnposen $P^E_1, \ldots, P^E_{N_E}$ bestehenden Entnahmebahn aus der Kiste heraus bewegt. Beide Bewegungsbahnen lassen sich meist mit einfachen Regeln definieren, die dazu führen, dass der Greifer sich dem Werkstück von oben und leicht aus Richtung der Mitte des Ladungsträgers nähert. Die Orientierung des Greifers wird in dieser Arbeit während der gesamten Bewegung konstant gehalten. Obwohl diese einfache Regeln üblicherweise ausreichend sind, sollen die Bewegungsbahnen in der in diesem Kapitel beschriebenen Suchstrategie ebenfalls betrachtet werden. Dazu wird ein Bahngenerator definiert, der diese Regeln zur Generierung von Bahnposen beinhaltet. Dieser folgt sowohl im gerichteten Graphen der Wissensbasis als auch im Suchbaum direkt auf den Roboterknoten. Beim Expandieren eines solchen Bahngeneratorknotens wird zunächst die letzte Bahnpose $P^A_{N_A}$ der Anfahrbahn generiert und in den Suchbaum eingefügt. Beim Expandieren eines Bahnposenknotens wird unter Verwendung der Funktionen des Bahngenerators die jeweils nächste Bahnpose erzeugt und ebenfalls in den Suchbaum eingefügt. Im Suchbaum ist die Anfahrbahn in umgekehrter Reihenfolge enthalten, so dass der Kindknoten des Bahngenerators die letzte Bahnpose der Anfahrbahn ist. Auf die erste Bahnpose P^A_1 der Anfahrbahn folgt im Suchbaum zunächst die erste Pose P^E_1 der Entnahmebahn. Die gesamte Bahn endet im Suchbaum mit der letzten Pose $P^E_{N_E}$ der Entnahmebahn. Abbildung 3.10 visualisiert die Reihenfolge der Bahnposen im Suchbaum. Der Grund für diese Repräsentation ist, dass somit beide Bewegungsbahnen vom Werkstück ausgehend im Suchbaum vertreten sind. Dies ermöglicht es, aufgrund der Baumstruktur, bei eventuell ungültigen Bahnposen auf eine alternative Bahnpose zurückzugreifen. Dazu könnten die Bahnposen verschiedene Kindknoten erzeugen, unter denen der bestmögliche ausgewählt werden kann. Da der Fokus dieser Arbeit jedoch nicht auf der Bahnplanung liegt und mögliche Entnahmen nur selten aufgrund einer

nicht optimalen Bewegungsbahn verhindert werden, wird in dieser Arbeit auf die Verwendung alternativer Bahnposen verzichtet.

Die letzte Bahnpose $P^E_{N_E}$ erzeugt beim Expandieren einen Bahninterpolator. Dieser ist dafür zuständig, die zuvor nur durch wenige Bahnposen gestützte Bewegungsbahn zu interpolieren, um die Bahn somit auch zwischen den Stützstellen überprüfen zu können. Dabei wird davon ausgegangen, dass der Roboter sich zwischen den einzelnen Bahnposen linear bewegt (vgl. Abschnitt 2.4.4). Es werden so viele Bahnposen hinzugefügt, dass zwei benachbarte Bahnposen nie weiter als δ voneinander entfernt sind. Um in der Nähe des Werkstücks eine hohe Genauigkeit zu erreichen und dennoch in anderen Bereichen der Bewegungsbahn nicht unnötig viele Bahnposen zu generieren, ist δ entlang der Bewegungsbahn nicht konstant. Stattdessen wird δ, aufgrund der Position entlang der Bewegungsbahn, linear zwischen zwei vorab definierten Parametern δ_{min} und δ_{max} interpoliert. Dabei wird δ_{min} für die Pose beim Greifen des Werkstücks und δ_{max} an der höchsten Pose der Bewegungsbahn verwendet. Dadurch kann erreicht werden, dass die Genauigkeit für die Prüfung der Bahnposen in der Nähe des Werkstücks am höchsten ist und über der Kiste, wo mit weniger Hindernissen zu rechnen ist, geringer. Dies führt zu einer geringen Rechenzeit bei gleichzeitig hoher Genauigkeit in der Nähe des Werkstücks. Beim Expandieren eines Bahninterpolators wird schließlich ein Zielknoten erzeugt, der eine gültige Greiflösung repräsentiert.

3.2.6 Paralleler Ablauf des Suchverfahrens

Im bisherigen Verlauf des Kapitels wurde der Ablauf des vorgestellten Verfahrens als rein sequentiell beschrieben. Um die Ausführungsgeschwindigkeit insbesondere für industrielle Realisierungen zu steigern, lässt sich das Verfahren jedoch auf mehreren Prozessorkernen parallel ausführen. Dies kann durch eine feste Anzahl N_T an Threads realisiert werden, die die Suche parallel durchführen. Dabei entnimmt jeder Thread, der seine vorherige Aufgabe abgeschlossen hat, den ersten Eintrag der Vorrangwarteschlange, evaluiert diesen und fügt gegebenenfalls neu erzeugte Knoten in den Suchbaum und in die Vorrangwarteschlange ein. Die einzelnen Threads können dadurch weitgehend unabhängig voneinander arbeiten. Lediglich die notwendigen Zugriffe auf gemeinsam genutzte Daten, wie die Vorrangwarteschlange, müssen synchronisiert werden. Falls die Threads voneinander unabhängig auf gemeinsam genutzten Daten arbeiten können, diese jedoch jeweils verändern, können die Daten alternativ mehrfach vorgehalten werden, um Engpässe durch die Synchronisation zu vermeiden. Auf diese Weise lassen sich die Transformation und Kollisionsprüfung von Flächenmodellen realisieren, ohne dass die Threads sich gegenseitig blockieren. Sobald einer der Threads einen Zielknoten erreicht hat, können alle anderen Threads ebenfalls beendet werden. Die Parallelisierung wird in dieser Arbeit nicht näher

untersucht. In der Praxis lässt sich die Rechenzeit für die Suche auf einem handelsüblichen Computer damit jedoch in etwa halbieren.

3.2.7 Extraktion der berechneten Greiflösung

Nach dem Abschluss der Suche können alle Informationen, die für die Ausführung des Greif- und Entnahmevorgangs benötigt werden, aus dem Suchbaum extrahiert werden. Dazu genügt es, den Pfad vom gefundenen Zielknoten aus rückwärts zur Wurzel des Baumes zu durchlaufen und dabei die notwendigen Informationen zu speichern. Dies sind insbesondere die einzelnen Bahnposen, die der Roboter benötigt, um die Bewegungsbahnen abfahren zu können. Falls der Greifer über zusätzliche Bewegungsachsen verfügt, so kann die Stellung dieser Achsen aus den zugehörigen Freiheitsgraden im Suchbaum ausgelesen werden. Darüber hinaus kann die konkret zu verwendende Greifpose von Interesse sein, da diese möglicherweise Informationen enthält, die für den eigentlichen Greifprozess relevant sind. Als Beispiel seien hier die Greifkraft oder die Information, ob sich der Greifer zum Greifen öffnen oder schließen soll, genannt. Je nach Anwendungsfall können weitere Daten relevant sein und ebenfalls aus dem Suchbaum ausgelesen werden, wie beispielsweise die Lage des zu greifenden Werkstücks.

Der bis zum Zeitpunkt einer Lösungsfindung expandierte Suchbaum und die Vorrangwarteschlange können nach der Suche weiter vorgehalten werden. Dies ermöglicht es, schnell eine alternative Lösung zu berechnen, falls der Roboter die zuerst berechnete Lösung nicht akzeptiert und eine andere Lösung anfragt. Durch das Vorhalten des Suchbaums und der Vorrangwarteschlange kann die Suche direkt an der bis dahin berechneten Stelle fortgeführt werden. Wenn solche Fälle häufiger auftreten, wird die Suche nach dem Finden der ersten Lösung auf Verdacht fortgeführt, so dass eine Alternativlösung bereits zur Verfügung steht.

In diesem Abschnitt wurde beschrieben, wie ein heuristisches Suchverfahren zur Greifplanung in der roboterbasierten Werkstückvereinzelung eingesetzt werden kann. Dazu wurde ein Verfahren auf Basis der Bestensuche entwickelt und im Anschluss wurden einige Aspekte im Detail beschrieben. Das Suchverfahren soll in der Lage sein, nur Greiflösungen zu finden, die in der Praxis ausführbar sind, ohne dass beispielsweise Kollisionen auftreten. Im folgenden Abschnitt wird beschrieben, wie das Verfahren prüfen kann, ob die einzelnen Knoten gültig sind, um sicherzustellen, dass die gefundene Greiflösung geeignet ist, ein Werkstück zu entnehmen.

3.3 Sicherstellung valider Griffe

Neben der Berechnung der Kosten dient das Evaluieren eines Knotens auch dazu, mögliche Probleme, die bei der Ausführung des Griffes auftreten können, zu erkennen und dadurch zu vermeiden. Dazu werden je nach Typ der Komponente, die der Knoten beschreibt, verschiedene Prüfungen durchgeführt und der Knoten als ungültig deklariert, falls eine dieser Prüfungen ein Problem aufdeckt. Dieser Abschnitt beschreibt die verschiedenen Prüfungen, die durchgeführt werden, und führt auf, bei welchen Knoten sie jeweils eingesetzt werden. Die Gesamtheit aller Prüfungen stellt sicher, dass die berechneten Griffe valide sind und vom Roboter ausgeführt werden können, ohne dass dabei Probleme auftreten.

3.3.1 Kollisionen mit der Punktwolke

Um einen robusten Entnahmeprozess sicherzustellen, müssen mögliche Kollisionen vorab erkannt und vermieden werden. Dazu werden einerseits der Greifer und andererseits die einzelnen Elemente des Roboters auf Kollisionen mit der Punktwolke geprüft. Durch diese Kollisionsprüfung mit der Punktwolke können potenzielle Kollisionen mit beliebigen Hindernissen detektiert werden. Dies beinhaltet insbesondere sowohl die erkannten als auch die nicht erkannten Werkstücke. Auch verbogene Kistenränder und unbekannte Hindernisse, die sich im Sichtbereich des Sensors befinden, werden dadurch berücksichtigt. Es wird davon ausgegangen, dass mögliche Hindernisse ausreichend groß und bezüglich ihrer Oberflächeneigenschaften geeignet sind, um vom Sensor erfasst zu werden.

Das in dieser Arbeit verwendete Verfahren zur Kollisionsprüfung entspricht im Wesentlichen dem Verfahren von Spenrath et al. (2012) und wird im Folgenden kurz vorgestellt. Dabei wird das Flächenmodell der zu prüfenden Komponente (Greifer- oder Roboterelement) zunächst in die zu prüfende Lage im Weltkoordinatensystem transformiert. Im Anschluss kann der Kollisionstest mit der Punktwolke durchgeführt werden, der auf dem sogenannten *Raytracing-Verfahren* (Glassner 1989) basiert. Dabei wird von einem Sensor ausgegangen, der den Abstand zur Szene strahlenförmig von einem Punkt aus abtastet, wie in Abbildung 3.11 dargestellt. Dies ist zwar lediglich bei der Laufzeit- und Phasenlagemessung der Fall, andere Sensoren, wie zum Beispiel die in Abschnitt 2.4.1 vorgestellten Lasertriangulationssensoren oder Stereokameras, können dadurch jedoch in ausreichendem Maße approximiert werden. Betrachtet man nun einen solchen Strahl vom 3D-Sensor zu einem beliebigen Messpunkt der 3D-Punktwolke, so lässt sich dieser Strahl in zwei Bereiche aufteilen, die durch die Messpunkte der Punktwolke voneinander abgegrenzt sind. Der Bereich zwischen dem Messpunkt p_i und dem Sensor wird als *sicherer Bereich* bezeichnet, da sich dort kein Hindernis befinden kann. Der Bereich hinter dem

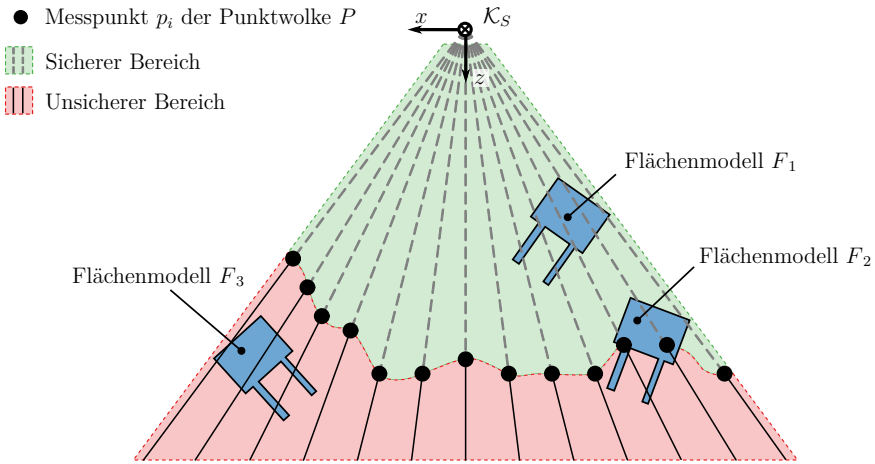


Abbildung 3.11: Kollisionstest mit der Punktwolke. Das Flächenmodell F_1 befindet sich vollständig im sicheren Bereich, Flächenmodell F_2 kollidiert direkt mit Messpunkten und Flächenmodell F_3 befindet sich vollständig im unsicheren Bereich.

Messpunkt wird als *unsicherer Bereich* bezeichnet, da dieser für den Sensor nicht sichtbar ist und daher keine Informationen über diesen Bereich vorliegen. Erweitert man diese Betrachtungsweise von einem Punkt auf die gesamte Punktwolke, so lässt sich der gesamte Sichtbereich des Sensors in die vorgenannten Bereiche aufteilen. Um eine Kollision mit Hindernissen sicher ausschließen zu können, sollten sich die Flächenmodelle des Greifers und des Roboters jederzeit im sicheren Bereich befinden. Aufgrund dieser Betrachtungsweise ist es nicht notwendig, zu unterscheiden, ob ein Flächenmodell mit einem Messpunkt kollidiert oder vollständig im unsicheren Bereich liegt. Dies erleichtert die Kollisionsprüfung, da nicht geprüft werden muss, ob einer der Messpunkte p_i der Punktwolke innerhalb des Flächenmodells des Greifers liegt. Stattdessen genügt es, von p_i ausgehend, eine Halbgerade in die dem Ursprung des Sensorkoordinatensystems $\mathcal{K}_S$ entgegengesetzte Richtung zu konstruieren. Schneidet diese Halbgerade das Flächenmodell, so liegt eine Kollision bzw. eine unsichere Situation vor. Da ein solches Flächenmodell in dieser Arbeit aus einzelnen Dreiecken besteht (vgl. Abschnitt 2.1.2), muss daher lediglich geprüft werden, ob die Halbgerade eines oder mehrere dieser Dreiecke schneidet.

Um die Kollisionsprüfung weiter beschleunigen zu können, wird die gesamte Punktwolke zu Beginn der Greifplanung einmalig in Quader (im folgenden *Sektionen* genannt) unterteilt. Bei einer Anzahl von N_{sek} Sektionen pro Koordinatenachse ergibt sich eine Gesamtanzahl von ${N_{sek}}^3$ Sektionen. Bei der Kollisionsprüfung kann die Berechnung somit beschleunigt werden, indem zunächst bestimmt wird, welche Sektionen für die *Axis-Aligned Bounding Boxes (AABB)*

Eingabe : Flächenmodell F, Anzahl N_C^{max}, Faktor g_A
Ausgabe : Vorliegen einer Kollision
Daten : Anzahl detektierter Kollisionen N_C

```
1  N_C ← 0;
2  für jedes Dreieck D ∈ F führe aus
3  |  Berechne AABB von D;
4  |  Bestimme relevante Sektionen der Punktwolke;
5  |  für jede Sektion S der relevanten Sektionen führe aus
6  |  |  für jeden g_A-ten Punkt p_i ∈ S führe aus
7  |  |  |  wenn Halbgerade von p_i entgegen der Richtung zu K_S schneidet D dann
8  |  |  |  |  N_C ← N_C + 1;
9  |  |  |  Ende
10 |  |  Ende
11 |  |  wenn N_C · g_A > N_C^max dann
12 |  |  |  Brich ab und liefere Ergebnis Kollision;
13 |  |  Ende
14 |  Ende
15 Ende
16 Brich ab und liefere Ergebnis keine Kollision;
```

Abbildung 3.12: Algorithmus zur Prüfung auf Kollisionen eines Flächenmodells mit der Punktwolke

(siehe Abschnitt 2.1.2) des jeweiligen Dreiecks relevant sind, so dass nur die in diesen Sektionen enthaltenen Punkte tatsächlich auf eine Kollision mit dem Dreieck geprüft werden. Abbildung 3.12 zeigt die Umsetzung des Verfahrens in einem Algorithmus.

Daten von optischen Sensoren sind immer mit einem Rauschen behaftet. Daher lassen sich Kollisionen mit einzelnen Punkten der Punktwolke in vielen Situationen nicht vollständig vermeiden. Wenn eine einzige Kollision eines Messpunkts mit einem Flächenmodell eine Greiflösung verhindern würde, könnten nur in den wenigsten Situationen Werkstücke gegriffen werden. Aus diesem Grund wird für jedes Greiferelement und jedes Roboterelement eine feste Obergrenze N_C^{max} an erlaubten Kollisionen definiert. Sei N_C die Anzahl der Punkte p_i, die in einem Kollisionstest eines Flächenmodells zu einer Kollision führen, so wird der zugehörige Knoten im Suchbaum nur als ungültig deklariert, falls

$$N_C > N_C^{max}\,. \tag{3.11}$$

Je nach Auflösung des Sensors können die erzeugten Punktwolken sehr viele Messpunkte enthalten. Um dennoch eine geringe Rechenzeit für die Kollisionsprüfung zu erreichen, kann vor der Kollisionsprüfung eine Unterabtastung der Punktwolke erfolgen. Dazu wird die Punktwolke mit einer Schrittweite von g_A abgetastet und es werden nur diese Punkte für die Kollisionsprüfung verwendet. Dies reduziert die Anzahl der Punkte um den Faktor g_A. Da die Punkte in der

Punktwolke normalerweise in der selben Reihenfolge gespeichert sind, in der sie durch den Sensor erfasst oder berechnet wurden, sind die verbleibenden Punkte dadurch relativ gleichmäßig über die gesamte Punktwolke verteilt. Für den Vergleich mit der maximal erlaubten Punktanzahl N_C^{max} muss dieser Faktor g_A ebenfalls berücksichtigt werden.

3.3.2 Kollisionen mit bekannten Objekten

Zusätzlich zu dem Kollisionstest der einzelnen Elemente des Greifers und des Roboters mit der Punktwolke werden diese Komponenten auf Kollisionen mit anderen Objekten geprüft. Zu diesen Objekten gehören einerseits fest im Raum installierte Hindernisse, wie Maschinen oder Schutzzäune, deren Lage bekannt ist. Die Geometrie dieser Objekte lässt sich durch ein Flächenmodell beschreiben. Da es beim Ladungsträger vorkommen kann, dass einzelne Seitenwände, beispielsweise aufgrund von Reflexionen, kaum Sensordaten in der Punktwolke generieren, soll die Robustheit des Verfahrens erhöht werden, indem auch der Ladungsträger bei der Kollisionsprüfung berücksichtigt wird. Hierbei kommt ein einfaches Flächenmodell zum Einsatz, das aus wenigen Dreiecken besteht, die die Seitenwände und den Boden des Ladungsträgers beschreiben. Um einen zusätzlichen Sicherheitsabstand zum Ladungsträger einzuhalten, kann dieses Modell geringfügig von der Geometrie des tatsächlichen Ladungsträgers abweichend gestaltet werden. In dieser Arbeit wird davon ausgegangen, dass der Ladungsträger bereits durch einen Algorithmus zur Objektlageerkennung erkannt wurde und die Lage des Ladungsträgerkoordinatensystems $\mathcal{K}_L$ relativ zum Weltkoordinatensystem daher bekannt ist. Sowohl bei den fest installierten Hindernissen als auch bei dem Ladungsträger kann somit ein Algorithmus zum Einsatz kommen, der auf Kollisionen zwischen zwei Flächenmodellen prüft. Der in dieser Arbeit verwendete Algorithmus ist in Abbildung 3.13 dargestellt. Prinzipiell wird dazu jedes Dreieck des einen Flächenmodells mit jedem Dreieck des anderen Flächenmodells auf Kollisionen getestet. Dabei wird jedoch vorab in mehreren Schritten auf Basis der *AABB* geprüft, ob eine Kollision möglich wäre. Im ersten Schritt (Zeile 1 in Abbildung 3.13) wird dabei geprüft, ob sich die *AABB* der beiden Flächenmodelle überlappen. Falls dies der Fall ist, wird im zweiten Schritt für jedes Dreieck des Flächenmodells eines Hindernisses geprüft, ob seine *AABB* die *AABB* des Flächenmodells des Greiferelements überlappt (Zeile 3). Diese Reihenfolge berücksichtigt, dass das Flächenmodell des Greiferelements meist kleiner als das Flächenmodell eines Hindernisses ist und häufig sogar vollständig innerhalb der *AABB* desselben liegt, wie beispielsweise im Falle des Ladungsträgers. Nur falls eine Kollision auf diese Weise noch nicht ausgeschlossen werden konnte, werden alle Dreiecke des Flächenmodells des Greifers einzeln überprüft – auch hier zunächst auf Basis ihrer *AABB* (ab Zeile 4). Da diesem Kollisionstest keine mit Rauschen behaftete Punktwolke zugrunde liegt, genügt hier schon eine einzelne detektierte Kollision, um den Knoten im Suchbaum für ungültig zu deklarieren. Falls in einer

```
   Eingabe : Flächenmodell F_G des Greifers und F_H eines Hindernisses
   Ausgabe : Vorliegen einer Kollision
 1 wenn AABB von F_G und F_H sich schneiden dann
 2 |  für jedes Dreieck D_H ∈ F_H tue
 3 |  |  wenn AABB von D_H und F_G sich schneiden dann
 4 |  |  |  für jedes Dreieck D_G ∈ F_G tue
 5 |  |  |  |  wenn AABB von D_H und D_G sich schneiden dann
 6 |  |  |  |  |  wenn D_H und D_G sich schneiden dann
 7 |  |  |  |  |  |  Brich ab und liefere Ergebnis Kollision;
 8 |  |  |  |  |  Ende
 9 |  |  |  |  Ende
10 |  |  |  Ende
11 |  |  Ende
12 |  Ende
13 Ende
14 Brich ab und liefere Ergebnis keine Kollision;
```

Abbildung 3.13: Algorithmus zur Prüfung auf Kollisionen mit bekannten Objekten

Anwendung sehr viele und komplexe Flächenmodelle zum Einsatz kommen, könnten alternativ auch effizientere, beispielsweise hierarchische Kollisionstests verwendet werden, wie zum Beispiel in Leonard et al. (2007) beschrieben. Dies ist jedoch nicht im Fokus dieser Arbeit.

Prinzipiell wäre es möglich, auch die Flächenmodelle aller im Ladungsträger erkannten Werkstücke auf Kollision mit dem Greifer und dem Roboter zu prüfen. Da jedoch nicht davon ausgegangen werden kann, dass sich jedes Werkstück erfolgreich erkennen lässt, muss die Kollisionsprüfung mit der Punktwolke dennoch in der Lage sein, Kollisionen mit Werkstücken sicher zu erkennen. Aus diesem Grund ist eine separate Prüfung mit den Flächenmodellen der Werkstücke, die zusätzlichen Rechenaufwand bedeuten würde, wenig zielführend.

3.3.3 Roboterkinematik und Erreichbarkeit

Neben Kollisionen stellen Probleme bei der Erreichbarkeit und insbesondere Singularitäten des Industrieroboters eine weitere Herausforderung für die Verfügbarkeit einer Roboteranlage in der industriellen Praxis dar. Aus diesem Grund wird beim Evaluieren eines Roboterknotens im Suchbaum die inverse Kinematik des Industrieroboters berechnet. In Abschnitt 2.4.4 wurde bereits erwähnt, dass sich die Betrachtung in dieser Arbeit auf Sechs-Achs-Knickarmroboter beschränkt, da in Roboteranlagen zur Vereinzelung von Werkstücken überwiegend diese Roboter eingesetzt werden. Da sich die inverse Kinematik jedoch auch für andere Roboter berechnen lässt, lässt sich dennoch das in Abschnitt 1.3 definierte Ziel erfüllen, dass beliebige Industrieroboter

einsetzbar sein sollen. Der Einfachheit halber wird weiterhin davon ausgegangen, dass sich wie in Abschnitt 2.4.4 erwähnt die Achsen 4 bis 6 des Industrieroboters in einem gemeinsamen Handwurzelpunkt schneiden. Da dies bei den meisten, in der Praxis eingesetzten, Industrieroboter der Fall ist, wird dies auch bei der dieser Arbeit zugrundeliegenden Implementierung vorausgesetzt. Für andere Industrieroboter ließe sich diese Implementierung jedoch durch eine universellere Implementierung ersetzen, so dass sich das Ziel der Flexibilität hinsichtlich des Roboters dennoch erreichen lässt.

Durch die Berechnung der inversen Kinematik werden die Winkel der einzelnen Roboterachsen bestimmt. Dadurch kann geprüft werden, ob sich diese innerhalb der für den Roboter zulässigen Achsgrenzen befinden. Durch die Kenntnis der einzelnen Achswerte kann darüber hinaus geprüft werden, ob eine Singularität (siehe Abschnitt 2.4.4) vorliegt, indem untersucht wird, ob sich Achse 3 oder Achse 5 in (oder nahe an) einer Strecklage befinden. Dies ist der Fall, wenn der Wert von Achse 3 bzw. Achse 5 im Intervall $(-\xi_{singu}, \xi_{singu})$ liegt. Seien ξ_i die Winkel der einzelnen Roboterachsen und ξ_i^{min} bzw. ξ_i^{max} die minimal bzw. maximal zulässigen Werte der Roboterachsen, so wird der entsprechende Knoten als ungültig deklariert, falls

$$\left(\exists i : \xi_i < \xi_i^{min} \vee \xi_i > \xi_i^{max}\right) \vee (|\xi_3| < \xi_{singu}) \vee (|\xi_5| < \xi_{singu}) \ . \tag{3.12}$$

An dieser Stelle sei erwähnt, dass die Berechnung der inversen Kinematik auch notwendig ist, um eine Kollisionsprüfung für die einzelnen Elemente des Roboters durchführen zu können.

3.3.4 Einschränkung der Orientierung

Für einige Komponenten sind gewisse Orientierungen physikalisch ausgeschlossen. Daher ist es sinnvoll, die zulässige Orientierung für diese Komponenten einzuschränken und damit die Suche zu beschleunigen. Dazu wird zunächst die in Abschnitt 3.2.3 definierte Abweichung γ zur bevorzugten Orientierung der Komponente berechnet. Weiterhin wird eine maximal erlaubte Winkelabweichung $\gamma_{max} \in [0, \pi]$ definiert und der Knoten für ungültig erklärt, falls

$$\gamma > \gamma_{max} \, . \tag{3.13}$$

Diese Einschränkung dient im Allgemeinen nicht der Vermeidung von potenziell auftretenden Problemen, kann jedoch stark geneigte Griffe verhindern, die vom Endanwender meist nicht gewünscht sind und die Akzeptanz verringern würden. Zudem kann der Aufwand für die Suche erheblich reduziert werden, indem viele unmögliche Griffe schon durch die Betrachtung der Orientierung vermieden werden. Andernfalls könnte dies erst bei einem Kollisionstest mit der Punktwolke oder bei der Prüfung der Roboterkinematik festgestellt werden.

Tabelle 3.1: Übersicht der durchgeführten Prüfungen bei der Evaluierung einzelner Knoten

Knoten	**Prüfungen und Reihenfolge**
TCP	Einschränkung der Orientierung des TCP relativ zum Ladungsträgerkoordinatensystem $\mathcal{K}_L$
Greiferelement	Folgende Prüfungen für den Zeitpunkt des tatsächlichen Greifvorgangs: 1. Kollisionsprüfung des Greiferelements mit dem Ladungsträger 2. Kollisionsprüfung des Greiferelements mit festen Hindernissen 3. Kollisionsprüfung des Greiferelements mit der Punktwolke
Verbindungspose	Einschränkung der Orientierung der Verbindungspose relativ zum Ladungsträgerkoordinatensystem $\mathcal{K}_L$
Roboter	Folgende Prüfungen für den Zeitpunkt des tatsächlichen Greifvorgangs: 1. Prüfung der Roboterkinematik, Achsbegrenzungen und Singularitäten 2. Kollisionsprüfung aller Roboterelemente mit dem Ladungsträger 3. Kollisionsprüfung aller Roboterelemente mit festen Hindernissen 4. Kollisionsprüfung aller Roboterelemente mit der Punktwolke
Bahnpose und Bahninterpolator	Folgende Prüfungen an der entsprechenden Bahnpose bzw. im Falle des Bahninterpolators an jeder durch die Interpolation hinzugefügten Bahnpose: 1. Kollisionsprüfung aller Greiferelemente mit dem Ladungsträger 2. Kollisionsprüfung aller Greiferelemente mit festen Hindernissen 3. Kollisionsprüfung aller Greiferelemente mit der Punktwolke 4. Prüfung der Roboterkinematik, Achsgrenzen und Singularitäten 5. Kollisionsprüfung aller Roboterelemente mit dem Ladungsträger 6. Kollisionsprüfung aller Roboterelemente mit festen Hindernissen 7. Kollisionsprüfung aller Roboterelemente mit der Punktwolke

3.3.5 Übersicht der in den Komponenten durchgeführten Prüfungen

In den vorherigen Abschnitten wurden diverse Prüfungen beschrieben, um sicherstellen zu können, dass ein Entnahmevorgang erfolgreich ausgeführt werden kann. Diese Prüfungen können einzelne Knoten des Suchbaums als ungültig deklarieren und werden teilweise in mehreren Knoten und für verschiedene Komponenten durchgeführt. Häufig werden in einzelnen Knoten auch mehrere dieser Prüfungen nacheinander durchgeführt. Für eine effiziente Berechnung ist es dabei wichtig, dass schnell durchzuführende Prüfungen und Prüfungen, die mit höherer Wahrscheinlichkeit negativ ausfallen, zuerst durchgeführt werden, da somit die Anzahl durchzuführender und insbesondere rechenaufwändiger Prüfungen reduziert werden kann. Tabelle 3.1 gibt eine Übersicht, in welchen Knoten des Suchbaums die einzelnen Prüfungen in welcher Reihenfolge durchgeführt werden.

Zusammengefasst lässt sich festhalten, dass die Kollisionsprüfung für die einzelnen Elemente des Greifers, einerseits für den Zeitpunkt des tatsächlichen Greifvorgangs und andererseits an den einzelnen Bahnposen der Anfahr- und Entnahmebahn durchgeführt wird. Selbiges gilt für die Prüfung der Roboterkinematik sowie für die Kollisionsprüfung der einzelnen Elemente des

Roboters. Das gegriffene Werkstücke wird dabei nicht berücksichtigt und auf der Entnahmebahn nicht auf Kollisionen geprüft. Dadurch kann es zwar vorkommen, dass Werkstücke auf der Entnahmebahn gegen andere Werkstücke stoßen, jedoch stellt dies bei den meisten als Schüttgut gelagerten Werkstücken kein Problem dar. Andererseits können dadurch auch Werkstücke entnommen werden, die nicht vollständig frei liegen, sondern von anderen Werkstücke überlappt sind. In der Praxis ist dies häufig notwendig, um in allen Situationen ein Werkstück entnehmen zu können, auch wenn nur wenige Werkstücke erkannt werden konnten. Diese Strategie wird daher auch von den meisten Ansätzen im Stand der Technik verfolgt. Für Anwendungsfälle, in denen jegliche Kollision des gegriffenen Werkstücks verhindert werden muss, könnte das Werkstück auf der Entnahmebahn als Teil des Flächenmodells des Greifers betrachtet und lediglich Kollisionen mit den durch das Werkstück selbst erzeugten Sensordaten ignoriert werden. Dies soll jedoch nicht Gegenstand der vorliegenden Arbeit sein.

3.4 Analyse und Bewertung

Zur Untersuchung verschiedener Aspekte von Roboteranlagen zur Vereinzelung von ungeordnet gelagerten Werkstücken ist es möglich, einzelne Komponenten oder sogar die gesamte Anwendung zu simulieren (Schyja et al. 2012; Schyja et al. 2014; Schyja et al. 2015; Wnuk et al. 2017; Fur et al. 2018). Dabei kann eine Physiksimulation zum Einsatz kommen, um Ladungsträger virtuell mit Werkstücken zu füllen. Durch virtuelle Sensoren können Punktwolken dieser virtuellen Szene generiert werden. Ein wesentlicher Vorteil solcher Computersimulationen liegt darin, dass dadurch eine Vielzahl an Durchläufen in kurzer Zeit virtuell durchgeführt werden kann, die andernfalls ein Vielfaches länger dauern würden und mit deutlich höherem Aufwand verbunden wären. Durch die Verwendung einer Simulation ist zudem die Lage aller Werkstücke bekannt. Eine solche Simulation kann daher zur Analyse und Evaluierung der in dieser Arbeit vorgestellten Greifplanung eingesetzt werden, ohne dass ein Verfahren zur Objektlageerkennung verwendet werden muss. Dieses Kapitel stellt zunächst die Simulationsumgebung vor, die für die nachfolgenden Untersuchungen verwendet wird. Im Anschluss wird das in dieser Arbeit entwickelte Verfahren zur Greifplanung anhand dieser Simulation untersucht und damit simulativ validiert. In den weiteren Abschnitten wird das Verfahren schließlich theoretisch eingeordnet und bewertet sowie die Komplexität theoretisch abgeschätzt.

3.4.1 Beschreibung der Simulations- und Testumgebung

Zur Evaluation der in dieser Arbeit entwickelten Methoden kommt eine Computersimulation zum Einsatz, die auf der Simulationsumgebung V-REP (Freese et al. 2010; Rohmer et al. 2013)

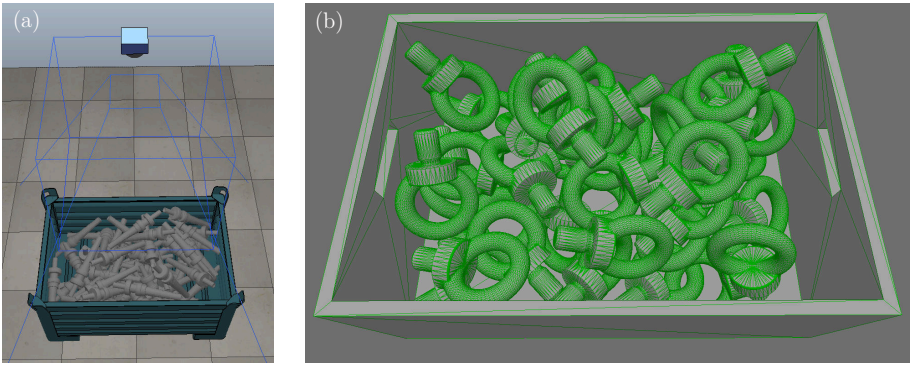


Abbildung 3.14: Verwendete Simulationsumgebung mit Ladungsträger, Werkstücken und 3D-Sensor für das Szenario der Getriebewellen (a) sowie eine Detailansicht der Werkstücke im Ladungsträger für das Szenario der M30-Ringschrauben (b).

basiert und bereits von Kleeberger et al. (2019) für die Generierung eines Datensatzes für den Griff-in-die-Kiste verwendet wurde. Innerhalb dieser Simulationsumgebung werden virtuelle Ladungsträger mit virtuellen Werkstücken gefüllt, indem die Werkstücke im Abstand von einer Sekunde erzeugt und von einer Position mittig über dem Ladungsträger fallen gelassen werden. Die Physiksimulation der Werkstücke erfolgt dabei mittels der Open Dynamics Engine (ODE) (Smith et al. 2007). Die erzeugten Szenen können, sobald die Werkstücke zur Ruhe gekommen sind, mittels eines virtuellen 3D-Sensors erfasst werden. Die Erzeugung der Sensordaten erfolgt dabei durch eine perspektivische Projektion mittels OpenGL. Eine Einführung in OpenGL findet sich beispielsweise in Woo et al. (1999). Im Rahmen dieser Arbeit wird eine solche Simulation für zwei verschiedene Szenarien durchgeführt. Das erste Szenario verwendet Getriebewellen als Werkstücke und das zweite Ringschrauben der Größe M30. Dies sind die selben Werkstücke, die auch in Kleeberger et al. (2019) zum Einsatz kommen. Abbildung 3.14 zeigt eine 3D-Darstellung der Simulationsumgebung für beide Szenarien. Bei der Generierung der Sensordaten für beide Szenarien gibt es folgende Unterschiede:

- **Getriebewellen:** Für das Szenario der Getriebewellen befindet sich der Sensor in einem Abstand von 2333 mm mittig über dem Boden des Ladungsträgers. Die mittlere Dichte der Punktwolke, d.h. die Anzahl der Punkte pro Flächeneinheit, entspricht dabei in etwa der eines Laserscanners vom Typ Sick LMS-400 bei vergleichbarem Messabstand. Da reale Sensordaten üblicherweise mit einem gewissen Rauschen behaftet sind, wird die virtuell gemessene Entfernung eines Messpunkts zum Sensor ebenfalls mit einem Rauschen belegt. Die für dieses gaußsche Rauschen verwendete konstante Standard-Abweichung $\sigma_R = 9\,\text{mm}$ entspricht in etwa dem typischen Rauschen des Sick LMS-400 Laserscanners unter den zu

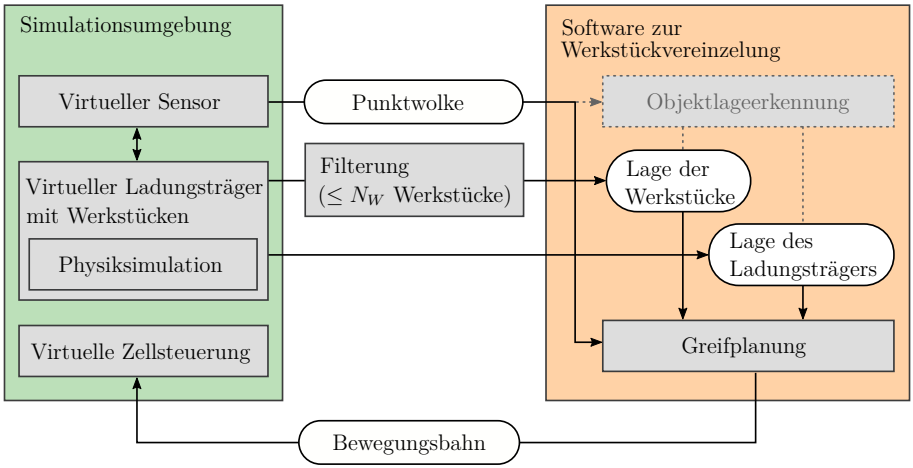
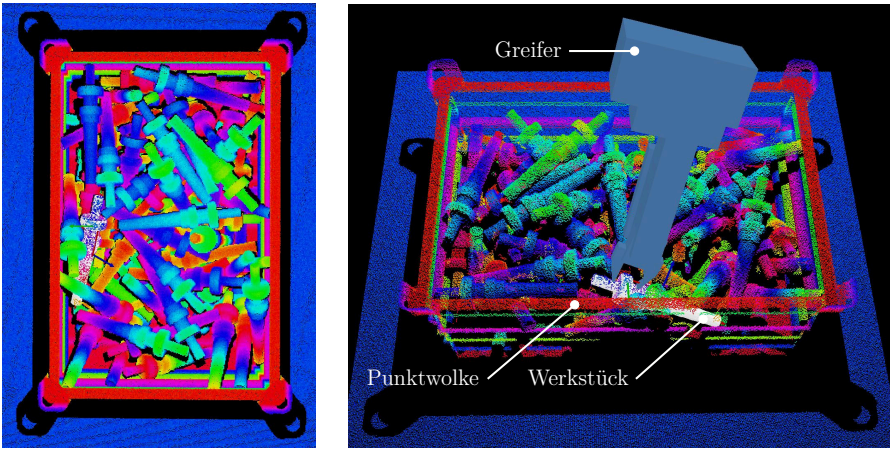


Abbildung 3.15: Architektur der Versuche mit Simulationsumgebung. Alle Elemente der realen Roboteranlage werden dabei durch eine Simulationsumgebung ersetzt.

erwartenden Einsatzbedingungen bei dunklen Werkstücken mit einer Remission von 10 % (SICK 2017).

- **Ringschrauben:** Für das Szenario der Ringschrauben befindet sich der Sensor in einem Abstand von 1665 mm mittig über dem Boden des Ladungsträgers. Die mittlere Dichte der Punktwolke entspricht hier in etwa der einer Stereokamera vom Typ Ensenso N20-1202-16-BL. Das Rauschen des Sensors wird hier durch ein lineares Rauschmodell in der Richtung der z-Achse des Sensorkoordinatensystems $\mathcal{K}_S$ realisiert, das die in der Spezifikation (IDS 2018) angegebenen Genauigkeiten im relevanten Messabstand zwischen 1425 mm und 1665 mm approximiert. Abbildung A.1 in Anhang A.1 zeigt die Standardabweichung des verwendeten Modells im Vergleich zu den in der Spezifikation angegebenen Genauigkeiten. Um realistischere Sensordaten zu erzeugen, könnte auch ein fortgeschritteneres Sensormodell verwendet werden, wie beispielsweise von Fur et al. (2018) vorgeschlagen. In dieser Arbeit soll darauf jedoch verzichtet werden.

Die Simulationsumgebung kann nun, wie in Abbildung 3.15 dargestellt, mit der Greifplanung verbunden werden. Die Simulation ersetzt dabei alle realen Komponenten. Durch eine Physiksimulation wird die Kiste mit virtuellen Werkstücken gefüllt und durch einen virtuellen Sensor erfasst. Die somit erzeugte Punktwolke wird für die Kollisionsprüfung innerhalb der Greifplanung verwendet. Auch wenn eine Objektlageerkennung, basierend auf den virtuellen Sensordaten, weiterhin verwendet werden könnte, so lässt sich bei Verwendung der Simulation



(a) Ansicht einer simulierten Punktwolke von oben

(b) Schräge Ansicht einer berechneten Greiflösung mit eingeblendetem Flächenmodell des Greifers sowie dem zu greifenden Werkstück

Abbildung 3.16: Ansichten einer simulierten Punktwolke und einer berechneten Greiflösung

auf diese Komponente verzichten, da die Lagen aller Werkstücke und des Ladungsträgers durch die Simulation vollständig bekannt sind. Dies stellt einen großen Vorteil dar, da die Greifplanung somit unabhängig von den Eigenschaften und Ungenauigkeiten eines bestimmten Algorithmus zur Objektlageerkennung erprobt und validiert werden kann. Zu beachten ist jedoch, dass ein realer Algorithmus zur Objektlageerkennung meist nicht in der Lage ist, alle Werkstücke zu erkennen. Häufig werden nur die oben liegenden Werkstücke erkannt. Werkstücke, die größtenteils von anderen Werkstücken verdeckt sind, werden in der Praxis nicht erkennbar sein. Aus diesem Grund werden die Werkstücke in der Simulation anhand des z-Werts ihrer Position relativ zum Ladungsträgerkoordinatensystem sortiert. Es wird angenommen, dass nur die N_W höchstgelegenen Werkstücke erkannt werden können und es werden daher nur diese für die Greifplanung berücksichtigt.

Abbildung 3.16 zeigt eine durch die Simulation erzeugte Punktwolke einer Kiste mit Getriebewellen sowie einen auf diesen Daten geplanten Griff. Als Greifer kommt bei diesem Szenario der abgebildete Zweifinger-Klemmgreifer mit einer zusätzlichen Bewegungsachse zum Einsatz. Das zugehörige Flächenmodell für die Kollisionsprüfung kann ebenfalls Abbildung 3.16 entnommen werden. Auf ein Robotermodell wird bei den beiden Simulationsszenarien verzichtet, um zu vermeiden, dass die Ergebnisse von Position und Kinematik des Roboters abhängen. Abbildung 3.17 zeigt die beiden Greifposen, die für das Szenario mit den Getriebewellen verwendet werden. Dabei

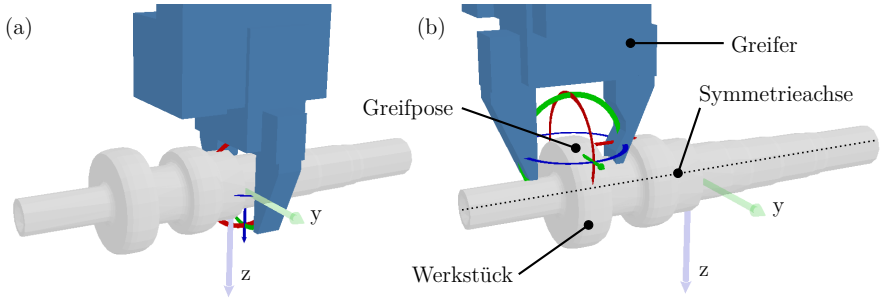


Abbildung 3.17: Festgelegte Greifposen für das Simulationsszenario mit Getriebewellen

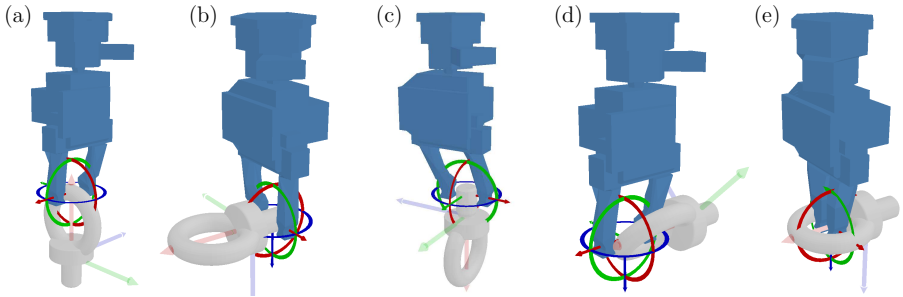


Abbildung 3.18: Beispielhafte Greifposen für das Simulationsszenario mit Ringschrauben. Bei (a) bis (d) wird das Werkstück durch ein Schließen der Greiferfinger gegriffen, bei (e) durch ein Öffnen der Greiferfinger.

verfügen beide Greifposen über zwei rotatorische Freiheitsgrade. Der erste erlaubt eine Rotation um 180° um die z-Achse, so dass die beiden Finger vertauscht sind. Der zweite Freiheitsgrad erlaubt eine beliebige Drehung um die Symmetrieachse des rotationssymmetrischen Werkstücks. Dieser Freiheitsgrad ist bei rotationssymmetrischen Werkstücken immer notwendig, da dort die Orientierung des Werkstückkoordinatensystems $\mathcal{K}_W$ um die Symmetrieachse durch das Verfahren zur Objektlageerkennung nicht eindeutig bestimmt werden kann. Bei den Ringschrauben wird auf eine zusätzliche Bewegungsachse im Greifer verzichtet. Um ein Werkstück dennoch in möglichst vielen Lagen greifen zu können, werden stattdessen insgesamt 14 Greifposen verwendet (teilweise mit mehreren Freiheitsgraden). Abbildung 3.18 zeigt beispielhaft einige dieser Greifposen.

3.4.2 Beispiele für Suchbäume auf Simulationsdaten

Die Abbildungen 3.19 und 3.20 zeigen beispielhaft, wie die Suchbäume für das Szenario der Getriebewellen in einer vollständigen Umsetzung aussehen. In beiden Abbildungen lässt sich das Szenario der Getriebewellen, mit den zwei Freiheitsgraden und der zusätzlichen Greiferachse wiedererkennen. In Abbildung 3.19 ist ein kleiner Suchbaum dargestellt, in dem drei Werkstücke berücksichtigt wurden und relativ schnell eine Lösung gefunden wurde. Hier wurden nur wenige Knoten durch die Funktion `Evaluiere` für ungültig befunden, alle aufgrund von Kollisionen einzelner Greiferelemente. Abbildung 3.20 zeigt hingegen einen größeren Suchbaum, in dem, aufgrund von erkannten Kollisionen, mehr Knoten untersucht werden mussten, bis eine Lösung gefunden wurde. Bei einigen potenziellen Greiflösungen war hier der Griff selbst zwar zugänglich, jedoch wurden Probleme an einer Pose der Bewegungsbahn festgestellt und die Suche deshalb dort nicht fortgesetzt. Nach dieser beispielhaften Betrachtung zweier vollständiger Suchbäume folgen nun einige quantitative Ergebnisse auf Basis der vorgestellten Simulationen.

3.4.3 Ergebnisse auf Basis von Simulationsdaten

In diesem Abschnitt wird untersucht, wie viele Werkstücke erkannt werden müssen, um eine geeignete Greiflösung zu finden und wie die Anzahl aller möglichen Greiflösungen von der Anzahl der erkannten Werkstücke abhängt. Dazu werden, wie in Abschnitt 3.4.1 beschrieben, Ladungsträger simulativ mit Werkstücken gefüllt. Anschließend wird durch die in diesem Kapitel vorgestellte Suchstrategie versucht, geeignete Greiflösungen zu finden. Im Gegensatz zu einer realen Anwendung wird allerdings nicht nach dem Erreichen des ersten Zielknotens gestoppt, sondern alle Suchbäume vollständig expandiert. Um verschiedene Füllgrade des Ladungsträgers zu betrachten, werden in der folgenden Untersuchung die Ladungsträger mit $N_W^L = 25$, $N_W^L = 50$ und $N_W^L = 75$ Getriebewellen bzw. mit $N_W^L = 10$, $N_W^L = 20$, $N_W^L = 30$ und $N_W^L = 40$ Ringschrauben gefüllt. Für jede dieser Varianten werden $N_S = 200$ Ladungsträger gefüllt, um verschiedene Situationen analysieren zu können. An dieser Stelle sei hervorgehoben, dass der Umfang einer solchen statistischen Untersuchung ohne die Verwendung von Simulationen nur mit extremem Aufwand umsetzbar wäre. Die detaillierten Parameter für den Versuch mit Getriebewellen können dem Parametersatz 1.1 und für den Versuch mit Ringschrauben dem Parametersatz 1.2 in Tabelle A.1 in Anhang A.2 entnommen werden.

Abbildung 3.21 zeigt, wie viele Greiflösungen in Abhängigkeit von der Anzahl berücksichtigter Getriebewellen gefunden werden. Dabei ist zu sehen, dass die Anzahl der gefundenen Lösungen zunächst nahezu linear mit der Anzahl N_W der berücksichtigten Werkstücke steigt. Dieser Anstieg lässt mit steigender Werkstückanzahl nach, was sich dadurch erklären lässt, dass an niedrigeren

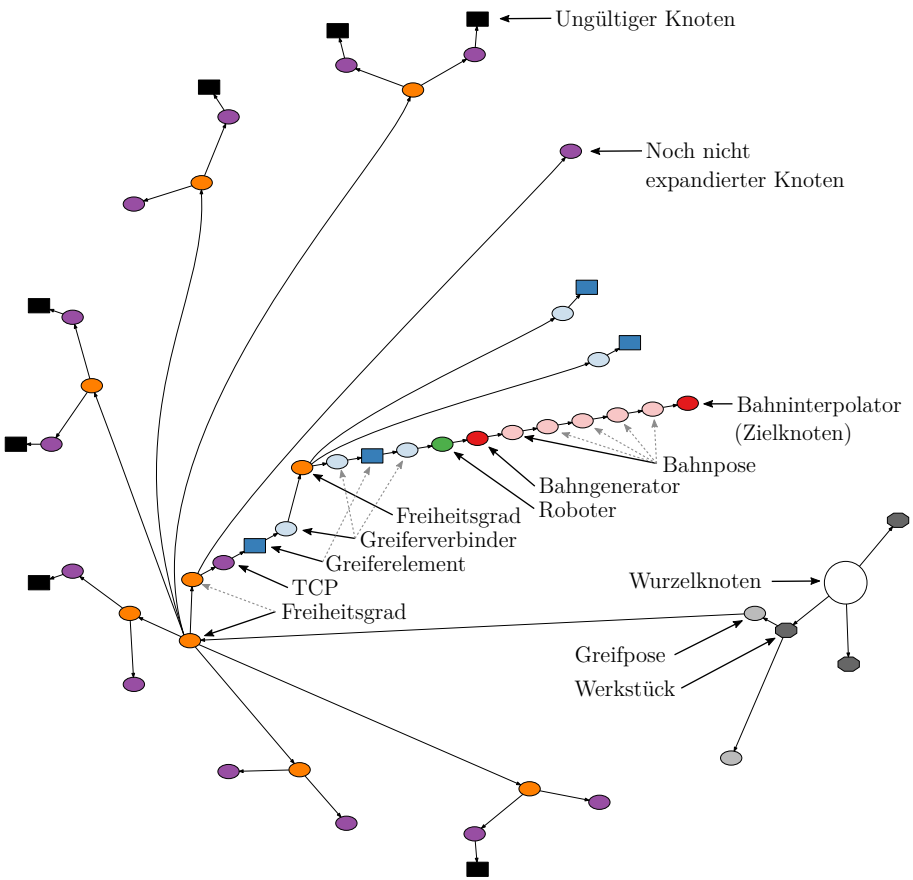


Abbildung 3.19: Beispiel eines kleinen Suchbaums für das Szenario der Getriebewellen, in dem drei Werkstücke berücksichtigt wurden und relativ schnell eine Lösung gefunden wurde. Die einzelnen Knoten sind anhand ihres Typs eingefärbt und vom Wurzelknoten bis zum Zielknoten zusätzlich beschriftet. Um den Suchbaum besser darstellen zu können, sind die Knoten nicht ausgehend von der Wurzel von oben nach unten, sondern weitgehend kreisförmig angeordnet. Die schwarz dargestellten Knoten wurden durch die Funktion `Evaluiere` für ungültig befunden. Im Suchbaum sind deutlich die beiden Freiheitsgrade an der Greifpose zu erkennen, von denen der zweite nur zwei diskrete Schritte zulässt und somit zwei TCPs als Kindknoten besitzt.

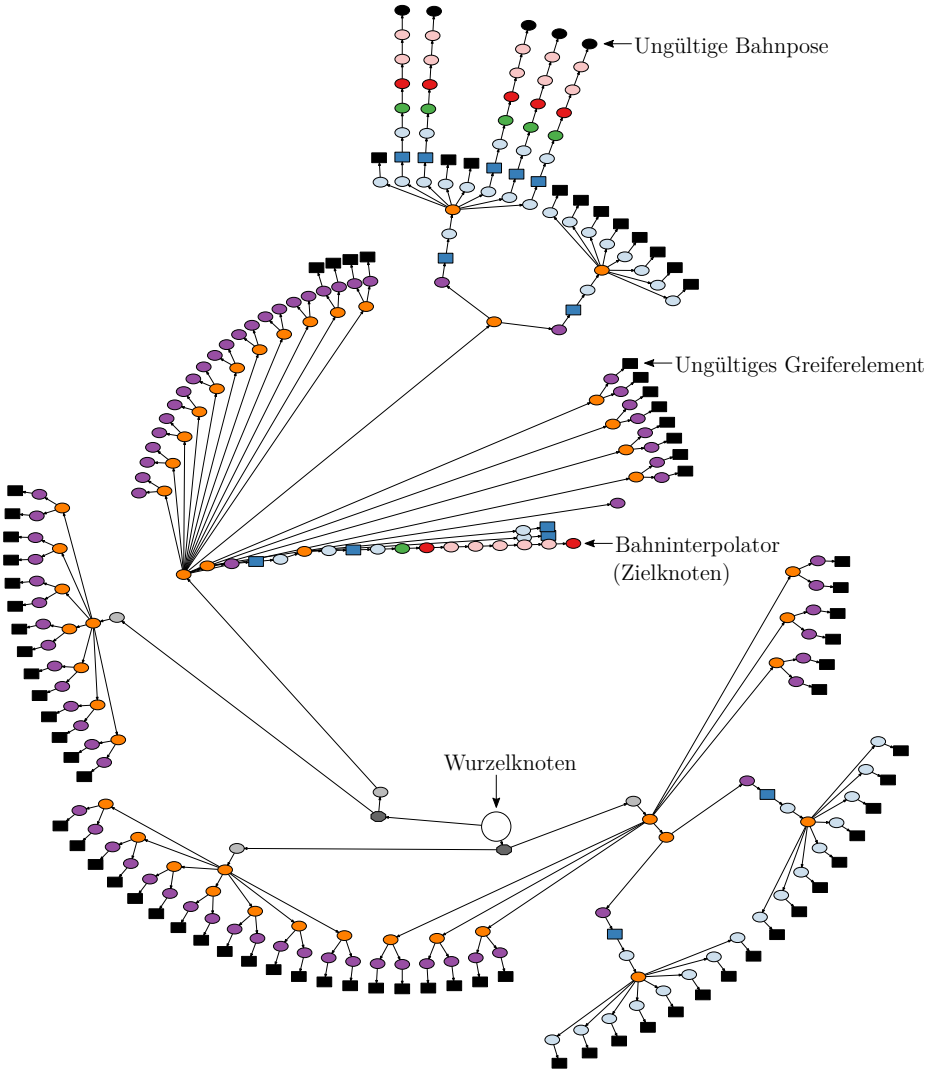


Abbildung 3.20: Beispiel eines größeren Suchbaums für das Szenario der Getriebewellen, in dem nur zwei Werkstücke berücksichtigt wurden und die Suche nach einer Lösung relativ lange gedauert hat. Die Farbgebung der Knoten entspricht der aus Abbildung 3.19.

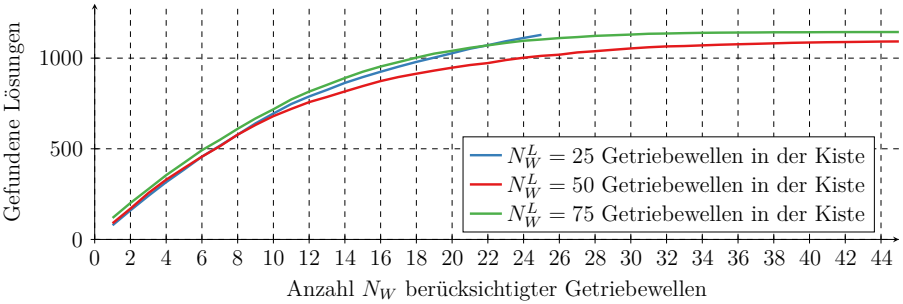


Abbildung 3.21: Anzahl gefundener Greiflösungen beim Szenario der Getriebewellen in Abhängigkeit von der Anzahl berücksichtigter Werkstücke. Angegeben ist der Mittelwert von $N_S = 200$ gefüllten Ladungsträgern.

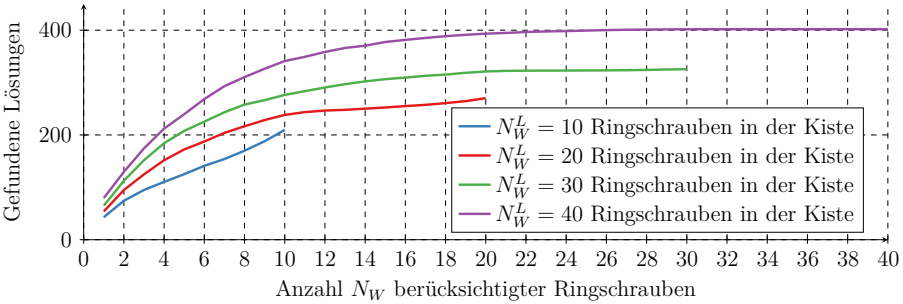


Abbildung 3.22: Anzahl gefundener Greiflösungen beim Szenario der Ringschrauben in Abhängigkeit von der Anzahl berücksichtigter Werkstücke. Angegeben ist der Mittelwert von $N_S = 200$ gefüllten Ladungsträgern.

Werkstücken weniger geeignete Griffe gefunden werden können. Daraus lässt sich ableiten, dass die Wahrscheinlichkeit, einen geeigneten Griff zu finden, an höher liegenden Werkstücken größer ist. Zudem kommen im Bereich von ca. 30 - 40 berücksichtigten Werkstücken kaum noch neue Greiflösungen hinzu, da die zusätzlich berücksichtigten Werkstücke größtenteils verdeckt sind. Weiterhin fällt auf, dass der Füllgrad des Ladungsträgers bei diesem Szenario kaum einen Einfluss auf die Anzahl der gefundenen Lösungen hat. Abbildung 3.22 zeigt die Ergebnisse im Falle der Ringschrauben. Hier lässt sich, insbesondere bei den Versuchen mit volleren Ladungsträgern, ein vergleichbares Resultat beobachten. Jedoch führen bei den Ringschrauben weniger gefüllte Ladungsträger zu einer geringeren mittleren Anzahl an Lösungen. Dies lässt sich dadurch erklären, dass der Greifer in diesem Szenario nicht über eine zusätzliche Achse verfügt. Somit sind die Werkstücke im unteren Teil der Kiste schwieriger zu greifen, ohne dass der Greifer mit den Seitenwänden der Kiste kollidiert.

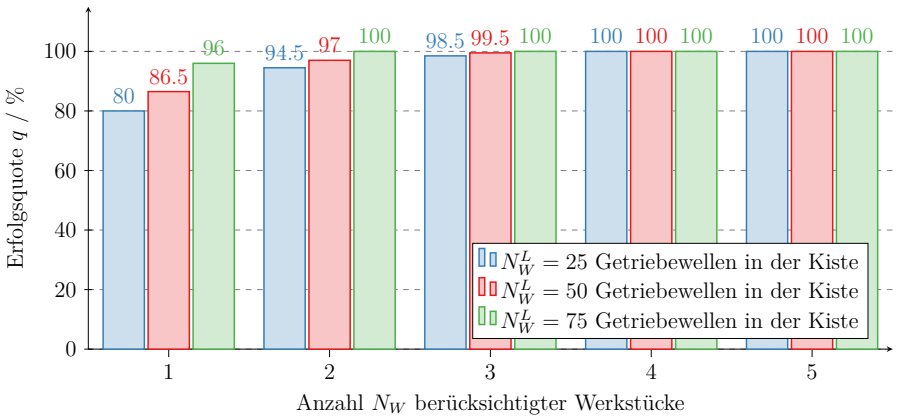


Abbildung 3.23: Erfolgsquote q in Abhängigkeit von der Anzahl der berücksichtigten Werkstücke

Die Abbildungen 3.21 und 3.22 zeigen, wie die Anzahl der Lösungen mit der Anzahl N_W der berücksichtigten Werkstücke zusammenhängt. Da es in realen Anwendungen jedoch ausreicht, für ein gegebenes Szenario eine einzige geeignete Lösung zu finden, sei eine *Erfolgsquote* q definiert als

$$q = \frac{N_{\tilde{S}}}{N_S} \tag{3.14}$$

mit der Anzahl N_S an untersuchten Situationen und der Anzahl $N_{\tilde{S}}$ an Situationen, in denen mindestens eine Lösung existiert. Abbildung 3.23 zeigt die Erfolgsquote q in Abhängigkeit von der Anzahl N_W der in der Suche berücksichtigten Werkstücke für das Szenario der Getriebewellen. Es ist deutlich zu sehen, dass es in weniger gefüllten Ladungsträgern prinzipiell schwieriger ist, einen geeigneten Griff zu finden. In vielen Situationen existiert keine geeignete Lösung, wenn nur das höchste Werkstück in der Greifplanung berücksichtigt wird. In diesem Fall beträgt q bei einer kaum gefüllten Kiste mit 25 Werkstücken sogar nur 80 %. Ab vier berücksichtigten Werkstücken lässt sich jedoch in allen untersuchten Situationen ein geeigneter Griff finden. Für das Szenario der Ringschrauben ist eine ähnliche Tendenz erkennbar, jedoch deutlich geringer, da q in den damit durchgeführten Versuchen nie unterhalb 97,5 % lag. Diese Ergebnisse zeigen auch, dass der in diesem Kapitel vorgestellte Ansatz geeignet ist, Greiflösungen zu finden, sofern ausreichend Werkstücke durch das Verfahren zur Objektlageerkennung erkannt werden. Auf eine Untersuchung der Rechenzeit wird an dieser Stelle noch verzichtet, da bisher lediglich angenommen wurde, dass eine geeignete Heuristikfunktion existiert. Ohne eine solche Heuristikfunktion ist eine Betrachtung der Rechenzeit oder der Anzahl der evaluierten Knoten des Suchbaums jedoch wenig aussagekräftig.

3.4.4 Theoretische Einordnung des Suchverfahrens

Sofern die beschriebenen Prüfungen der einzelnen Komponenten so parametriert sind, dass alle eventuell auftretenden Probleme durch diese erkannt werden, stellen alle Zielknoten geeignete Lösungen zur Entnahme von Werkstücken dar. Damit kann das Verfahren zu den *korrekten* Suchverfahren (siehe Abschnitt 2.2.2) gezählt werden. Falls die maximale Rechenzeit t_{max} nicht überschritten wird, ist das Verfahren zudem insofern *vollständig*, als dass es alle im Baum enthaltenen Lösungen finden kann. Es ist jedoch zu beachten, dass die Tatsache, dass das Suchverfahren keine Lösung findet, nicht zwingend bedeutet, dass physikalisch keine Entnahme möglich ist. Einerseits kann es durch die Diskretisierung der Freiheitsgrade Lösungen geben, die nicht im Suchbaum enthalten sind. Andererseits wurde beispielsweise in Abschnitt 3.3.1 gezeigt, dass die Kollisionsprüfung Griffe, bei denen Greiferelemente in unsicheren Bereichen liegen, nicht zulässt, um Kollisionen in jedem Fall zu vermeiden. Dennoch könnte in diesen Bereichen ein kollisionsfreier Griff existieren. Da die Bestensuche nicht zu den *optimalen* Suchverfahren zählt, gehört auch das in diesem Kapitel beschriebene Verfahren nicht zu den optimalen Suchverfahren. Eine optimale Lösung zu finden ist bei der hier beschriebenen Greifplanung nicht notwendig. Wie in Abschnitt 1.3 beschrieben, ist das Ziel vielmehr, in möglichst kurzer Zeit eine geeignete Lösung zur Entnahme eines Werkstücks zu finden.

Sofern das entwickelte Suchverfahren nur auf einem Prozessorkern ausgeführt wird, ist es deterministisch, da bei gleicher Eingabe stets die gleiche Ausgabe erzeugt wird und dabei immer die selben Zustände durchlaufen werden (vgl. Abschnitt 2.2.2). Damit ist es auch determiniert. Wird das Verfahren hingegen auf mehreren Prozessorkernen ausgeführt, so kann es vorkommen, dass die Knoten in unterschiedlicher Reihenfolge evaluiert und expandiert werden und somit unterschiedliche Ausgaben berechnet werden. Somit ist das entwickelte Verfahren bei der parallelen Ausführung auf mehreren Prozessorkernen weder deterministisch noch determiniert. Wie im vorherigen Absatz beschrieben, ist dies jedoch akzeptabel, da lediglich eine geeignete Lösung gefunden werden soll.

3.4.5 Bewertung der Greifplanung

An dieser Stelle soll theoretisch bewertet werden, inwieweit die in Abschnitt 1.3 aufgeführten Anforderungen durch das in diesem Kapitel beschriebene Suchverfahren erfüllt werden können.

Eine theoretische Aussage über die *Verfügbarkeit* ist nur schwer möglich. Die verschiedenen Prüfungen während des Suchverfahrens können zwar viele Probleme vermeiden und somit zu einer hohen Verfügbarkeit beitragen, ob die Verfügbarkeit für einen Einsatz in industriellen Vereinzelungsanlagen ausreicht, lässt sich jedoch nur durch eine Evaluierung in der Praxis

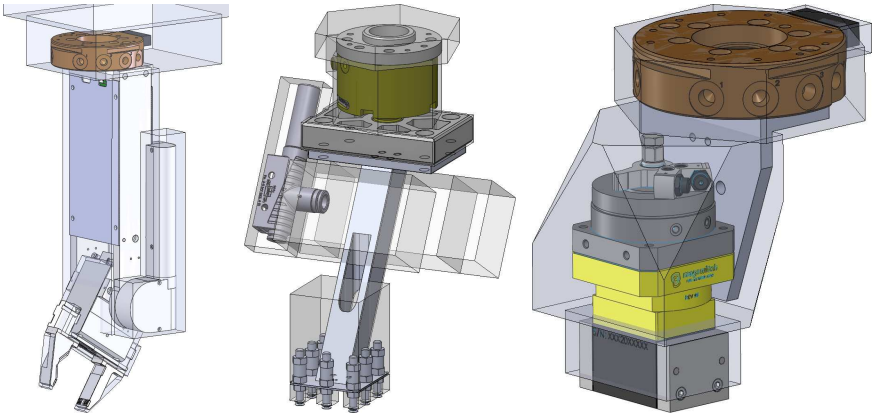
nachweisen. An dieser Stelle sei daher auf Abschnitt 5.2 verwiesen, in dem das Verfahren in einem industriellen Einsatz validiert wird.

Der erreichbare *Entleerungsgrad* ist in erster Linie von einem geeigneten Algorithmus zur Objektlageerkennung abhängig, der alle Werkstücke zuverlässig erkennen kann. Ebenso wichtig ist hierbei jedoch auch ein geeigneter Greifer für die zu entnehmenden Werkstücke. Sofern diese beiden Punkte erfüllt sind, lässt sich durch die Vollständigkeit des Suchverfahrens jedoch gewährleisten, dass (bei geeigneter Konfiguration) jederzeit ein möglicher Griff bestimmt und der Ladungsträger somit vollständig entleert werden kann. Zusätzlich wurde in Abschnitt 3.4.3 simulativ gezeigt, dass die Greifplanung bei einer ausreichenden Anzahl erkannter Werkstücke stets in der Lage ist, eine Greiflösung zu bestimmen.

Die Anforderung einer *hohen Flexibilität* beinhaltet, unter anderem, beliebige formstabile Werkstücke mit der Lösung handhaben zu können. Im Gegensatz zu Ansätzen, die nach bestimmten Geometrien, wie zum Beispiel Flächen, am Werkstück suchen, um diese dort greifen zu können, ist es durch die Repräsentation der Werkstückgeometrie als Flächenmodell und die Definition der Greifposen relativ zu diesem Modell möglich, beliebige formstabile Werkstücke zu handhaben. Eine weitere Anforderung besteht in der Unterstützung unterschiedlicher Ladungsträger. Dies ist dadurch gewährleistet, dass neben der Punktwolke, auch die vordefinierte Geometrie des Ladungsträgers bei der Kollisionsprüfung berücksichtigt wird. Damit lassen sich Kollisionen beispielsweise auch bei Gitterboxen ausschließen, bei denen die Seitenwände nur zu wenigen Sensordaten in der Punktwolke führen. Auch verschiedene Greifer und Greifprinzipien lassen sich mit der in dieser Arbeit vorgestellten Greifplanung abbilden. Durch die Definition der Greifposen mittels Transformationen relativ zu den Werkstücken und Repräsentation der Greiferelemente als Flächenmodelle spielt das verwendete Greifprinzip keine Rolle für die Greifplanung. Dabei ist jedoch zu beachten, dass die folgenden Anforderungen an die Flächenmodelle erfüllt sein müssen:

- Da die Sensordaten in der Praxis verrauscht sind, sollten alle Flächenmodelle, die für Kollisionsprüfungen verwendet werden, etwas größer gestaltet sein, als die tatsächlich vorhandene Störkontur.

- Bei Klemmgreifern muss das Flächenmodell des Greifers in ungegriffenem Zustand verwendet werden, da der Greifer in dieser Konfiguration in den Ladungsträger hinein bewegt wird. Zudem würde eine Kollision mit dem Werkstück detektiert werden, falls das Flächenmodell des Greifers in gegriffenem Zustand verwendet werden würde. Dass sich die Störkontur des Greifers in gegriffenem Zustand geringfügig unterscheidet, wird auf der Entnahmebahn vernachlässigt, da die Störkontur nach außen dabei ohnehin meist kleiner wird.



(a) Klemmgreifer (b) Sauggreifer (c) Magnetgreifer

Abbildung 3.24: Beispielhafte CAD-Modelle für verschiedene Greifprinzipien mit eingeblendeten Flächenmodellen für die Kollisionsprüfung

- Bei Saug- und Magnetgreifern muss das für die Kollisionsprüfung verwendete Flächenmodell an der Kontaktfläche etwas kleiner sein als die tatsächliche Störkontur. Da der Greifer das Werkstück an dieser Fläche berühren muss, würde andernfalls aufgrund der verrauschten Punktwolke eine Kollision festgestellt. Besonders stark lässt sich dieser Effekt bei elastischen Sauggreifern beobachten, da diese durch das Ansaugen „eingedrückt“ werden.

- Verformbare Teile des Greifers, wie zum Beispiel Kabel und Schläuche, müssen durch die Flächenmodelle nicht abgedeckt werden, solange keine Schäden durch Berührung dieser Teile mit der Kiste oder mit Werkstücken entstehen können und insbesondere keine Gefahr des Verhakens besteht.

Abbildung 3.24 zeigt beispielhafte CAD-Modelle für die drei wichtigsten Greifprinzipien samt der jeweils verwendeten Flächenmodelle für die Kollisionsprüfung. Die dargestellten Modelle entsprechen den physikalischen Greifern aus Abbildung 2.8. Hier ist zu sehen, dass in allen Fällen versucht wurde, die Störgeometrie durch wenige geometrische Körper abzubilden, die jeweils durch eine geringe Anzahl an Dreiecken repräsentiert werden können, und damit die Rechenzeit für die Kollisionsprüfung gering zu halten. Weiterhin ist zu erkennen, dass das Flächenmodell für die Kollisionsprüfung beim Sauggreifer und beim Klemmgreifer an der Kontaktfläche des Greifers etwas kleiner ist als der reale Greifer. Durch die Aufteilung in verschiedene Greiferelemente, die über Verbindungsposen mit möglichen Freiheitsgraden verbunden sind, lassen sich auch die in Abschnitt 1.3 erwähnten zusätzlichen Bewegungsachsen im Greifer in der Greifplanung berück-

sichtigen. Dies lässt sich beispielsweise auch am dargestellten Klemmgreifer in Abbildung 3.24(a) erkennen.

Die Anforderung, dass die entwickelte Greifplanung mit *beliebigen Algorithmen zur Objektlageerkennung* verwendbar sein soll, ist ebenfalls erfüllt. Dies wird dadurch sichergestellt, dass der Algorithmus lediglich eine Liste der Lagen aller erkannten Werkstücke voraussetzt. Eine zusätzliche Bestimmung der Lage des Ladungsträgers kann das Verfahren an einigen Stellen unterstützen, ist jedoch nicht zwingend notwendig. Auch wenn die meisten Objektlageerkennungsverfahren für Roboteranlagen zur Werkstückvereinzelung von sortenreinen Kisten ausgehen, kann die entwickelte Greifplanung auch eingesetzt werden, wenn eine Kiste unterschiedliche Werkstücke enthält, sofern deren Lagen bestimmt werden können. An dieser Stelle sei angemerkt, dass das Verfahren prinzipiell auch für modellfreie Ansätze verwendet werden kann. In diesem Fall würde der Suchbaum nicht bei denen einzelnen Werkstücken beginnen, sondern bei den anhand der Punktwolke bestimmten Greifposen. Die weiteren Komponenten des Verfahrens, wie die Greiferelemente, Greiferachsen oder Bahnposen bleiben hierbei unverändert.

Das Ziel der möglichst geringen *Taktzeit* wird im folgenden Abschnitt theoretisch betrachtet und in den Kapiteln 4 und 5 im Detail untersucht.

3.4.6 Abschätzung der Komplexität des Suchverfahrens

In diesem Abschnitt erfolgt eine Abschätzung der Größe des Suchbaums und damit der Komplexität der Suche. Dafür seien zunächst N_W als die Anzahl der erkannten Werkstücke, N_G als die Anzahl der an diesem Werkstücktyp definierten Greifposen, $(N_F)_i$ als die Anzahl der an der Greifpose G_i definierten Freiheitsgrade und $(N_S^F)_{i,j}$ als die Anzahl der diskreten Schritte der jeweiligen Freiheitsgrade definiert. Sei weiterhin N_a die Anzahl der in den Greifer integrierten zusätzlichen Bewegungsachsen und $(N_S^A)_k$ die Anzahl der Schritte der einzelnen Greiferachsen, so ergibt sich eine Anzahl an potenziellen Greiflösungen von

$$N_L = N_W \cdot \sum_{i=1}^{N_G} \left(\prod_{j=1}^{(N_F)_i} (N_S^F)_{i,j} \right) \cdot \prod_{k=1}^{N_a} (N_S^A)_k \,. \tag{3.15}$$

Für industrielle Anwendungen, mit einer großen Anzahl an Greifposen und Freiheitsgraden, kann diese Anzahl sehr groß werden. Als Beispiel sei an dieser Stelle eine industrielle Anwendung genannt, die in Kapitel 5 vorgestellt wird. Die darin zu handhabenden Werkstücke verfügen über $N_G = 20$ Greifposen mit jeweils zwei bis vier Freiheitsgraden mit unterschiedlicher Anzahl an diskreten Schritten. Der Greifer besitzt zwei zusätzliche Bewegungsachsen mit jeweils vier

diskreten Schritten. Werden nun in einer Entnahmesituation beispielsweise $N_W = 5$ Werkstücke erkannt, so ergibt sich daraus eine Gesamtzahl N_L von

$$\begin{aligned} N_L = 5 \cdot \Big(& 2 \cdot 11 + 2 \cdot 11 + 2 \cdot 11 + 2 \cdot 11 + 2 \cdot 2 \cdot 2 + 5 \cdot 2 \cdot 2 + 5 \cdot 2 \cdot 2 \\ & + 19 \cdot 2 \cdot 2 + 2 \cdot 16 + 2 \cdot 27 + 2 \cdot 11 \cdot 5 \cdot 3 + 2 \cdot 11 \cdot 5 \cdot 3 + 2 \cdot 11 \cdot 5 \cdot 3 \\ & + 2 \cdot 11 \cdot 5 \cdot 3 + 2 \cdot 2 + 2 \cdot 2 + 5 \cdot 2 + 5 \cdot 2 + 5 \cdot 2 + 5 \cdot 2 \Big) \cdot (4 \cdot 4) = 133\,280 \end{aligned} \tag{3.16}$$

möglichen Greiflösungen. Falls eine feinere Diskretisierung verwendet wird, erhöht sich diese Anzahl deutlich.

Betrachtet man die Anzahl der benötigten Kollisionstests während der Suche, so wird das Problem der Rechenzeit noch deutlicher. In einem relativ simplen Szenario bestehe die Anfahrbahn aus 10 Bahnposen und die Entnahmebahn ebenfalls aus 10 Bahnposen. Weiterhin sei angenommen, dass jedes der beiden Greiferelemente und jedes der sechs beweglichen Elemente eines Knickarmroboters (da die Roboterbasis fest steht, muss diese nicht geprüft werden) jeweils nur aus einem Flächenmodell bestehen. Würde man nun den gesamten Suchbaum vollständig durchsuchen, so müssen insgesamt

$$N_{col} = N_L \cdot \Big((1 + 10 + 10) \cdot (2 + 6) \Big) = 133\,280 \cdot 168 = 22\,391\,040 \tag{3.17}$$

Kollisionstests zwischen einem Flächenmodell und der Punktwolke sowie den Ladungsträger- und Hindernismodellen durchgeführt werden. Um die Rechenzeit dennoch gering zu halten, werden folgende Aspekte berücksichtigt:

- Ungültige Greiflösungen sollten nur soweit untersucht werden, bis festgestellt werden kann, dass sie ungültig sind. Zudem sollten Instanzen von Komponenten, die in mehreren Greiflösungen vorkommen, nicht mehrfach geprüft werden. Beide Punkte werden durch die Baumstruktur der Suche erreicht, da hier beispielsweise ein Kollisionstest des untersten Greiferelements mehrere potenzielle Greiflösungen auf einmal ausschließen kann.

- Weiterhin sollte eine Heuristik verwendet werden, die geeignet ist, um möglichst schnell eine gültige Lösung zu finden. Mit der Entwicklung und Analyse einer solchen Heuristikfunktion befasst sich Kapitel 4. Erst durch Verwendung einer geeigneten Heuristikfunktion, deren Existenz in diesem Kapitel lediglich angenommen wurde, kann eine Aussage über die tatsächliche Rechenzeit getroffen werden.

3.5 Zusammenfassung

In diesem Kapitel wurde ein heuristisches Suchverfahren entwickelt, mit dem Greifvorgänge für einen Industrieroboter geplant werden können, um ungeordnet gelagerte Werkstücke zu vereinzeln. Dazu wurde gezeigt, wie alle beteiligten Komponenten der Roboteranlage in einer Wissensbasis repräsentiert werden können. Basierend auf dieser Wissensbasis wurde ein heuristisches Suchverfahren entwickelt, das basierend auf der Bestensuche, für jeden Entnahmevorgang einen Suchbaum erstellt und diesen nach einer geeigneten Lösung durchsucht. Während der Suche wird direkt auf potenzielle Probleme geprüft, die beim Entnahmevorgang auftreten können. Somit werden beispielsweise Kollisionen mit Hindernissen oder singuläre Stellungen des Roboters vermieden. Schließlich wurde das Verfahren anhand von Simulationsdaten evaluiert und theoretisch eingeordnet. Weiterhin wurde gezeigt, dass das Suchverfahren geeignet ist, eine hohe Verfügbarkeit zu erreichen sowie die Kiste weitgehend zu entleeren und dass das Suchverfahren flexibel für verschiedene Anwendungen eingesetzt werden kann. Damit werden die gesetzten Ziele, abgesehen von der Taktzeit, erfüllt. Es wurde gezeigt, dass die Suchbäume, je nach Anzahl der erkannten Werkstücke und der konfigurierten Greifposen und Freiheitsgrade sehr groß werden können. Um dennoch eine kurze Rechenzeit und damit eine kurze Taktzeit zu erreichen, setzt das Verfahren eine geeignete Heuristikfunktion voraus, auf deren Beschreibung in diesem Kapitel zunächst verzichtet wurde. Mit der Definition und Untersuchung einer solchen Heuristikfunktion zur Beschleunigung der Suche beschäftigt sich das folgende Kapitel.

4 Entwicklung und Untersuchung einer Heuristik zur effizienten Suche

Im vorherigen Kapitel wurde ein heuristisches Suchverfahren zur Planung von Greifvorgängen entwickelt. Dabei wurde davon ausgegangen, dass eine geeignete Heuristikfunktion existiert. In diesem Kapitel wird eine solche Heuristikfunktion entwickelt und untersucht. Dafür werden zunächst verschiedene Bewertungsheuristiken eingeführt. Diese Bewertungsheuristiken basieren auf einer vollständigen Kenntnis des Suchbaums und ermöglichen daher eine objektive Bewertung von Knoten. Im Anschluss werden verschiedene Einflussfaktoren, wie die Position von Werkstücken oder die Orientierung von TCPs, für die einzelnen Arten von Komponenten beschrieben und mittels der Bewertungsheuristiken untersucht. Weiterhin wird ein Ansatz vorgestellt, die manuell definierten Einflussfaktoren um künstliche neuronale Netze zu ergänzen bzw. zu ersetzen. Die Einflussfaktoren und die daraus zusammengesetzte Heuristikfunktion werden auf Basis von Simulationsdaten untersucht. Zudem werden bayessche Optimierungsverfahren und evolutionäre Algorithmen verwendet, um optimale Werte für die Gewichtungsfaktoren der Heuristikfunktion zu finden.

4.1 Bewertung und Vergleich von Heuristikfunktionen und Einflussfaktoren

In Abschnitt 3.4.5 wurde bereits gezeigt, dass das Suchverfahren die meisten der in Abschnitt 1.3 definierten Ziele erfüllt. Die Taktzeit, die unter anderem durch die Rechenzeit der Greifplanung beeinflusst wird, wurde dabei zunächst nicht näher betrachtet. Die Rechenzeit des Suchverfahrens wird maßgeblich von der Wahl einer geeigneten Heuristikfunktion beeinflusst. Das Ziel der Heuristikfunktion ist es somit, die Suche möglichst schnell zu einem Zielknoten zu lenken und somit eine geringe Rechenzeit der Suche zu erreichen. Eine solche Heuristikfunktion wird ab Abschnitt 4.2 entwickelt und untersucht. Um verschiedene Heuristikfunktionen miteinander vergleichen zu können, wird jedoch zunächst diskutiert, wie verschiedene Heuristikfunktionen und deren Einflussfaktoren miteinander verglichen werden können. Dazu werden in diesem

Abschnitt Kriterien eingeführt, um sowohl verschiedene Heuristikfunktionen als auch einzelne Knoten des Suchbaums zu bewerten.

4.1.1 Bewertung von Heuristikfunktionen

Zur Bewertung von Suchverfahren in Bäumen lässt sich neben der Rechenzeit die Anzahl der Knoten heranziehen, die zum Auffinden einer Lösung expandiert bzw. evaluiert werden müssen. Die Anzahl der expandierten Knoten während einer Suche wird im Folgenden als N_{exp} bezeichnet und die Anzahl der Knoten die während einer Suche erzeugt und damit auch direkt evaluiert werden als N_{eval}. Je geringer die Werte von N_{exp} und N_{eval} sind, desto schneller findet das Suchverfahren im Allgemeinen eine Lösung, da der Baum kleiner bleibt und weniger Knoten expandiert bzw. evaluiert werden müssen. Um unabhängig von der verwendeten Hardware zu sein, werden in diesem Kapitel zunächst diese Werte anstelle der Rechenzeit betrachtet.

Zu beachten ist, dass zur Bestimmung der beiden Werte für eine gegebene Heuristikfunktion ein Suchdurchlauf durchgeführt werden muss. Für die Untersuchung einer großen Anzahl unterschiedlicher Einflussfaktoren und deren Kombinationen bedeutet dies somit einen hohen Rechenaufwand. Daher werden im folgenden Abschnitt Bewertungskriterien für einzelne Knoten des Suchbaums eingeführt.

4.1.2 Bewertung von Knoten eines vollständig expandierten Suchbaums

Um im weiteren Verlauf der Arbeit die einzelnen Einflussfaktoren effizient zu untersuchen, ist es vorteilhaft, einzelne Knoten des Suchbaums objektiv bewerten zu können. Dazu können Bewertungskriterien dienen, welche im Gegensatz zu der für die Suche verwendeten Heuristikfunktion auf Basis eines vollständig expandierten Suchbaums berechnet werden. Diese Kriterien sollen – unter Kenntnis der Zielknoten – eine Aussage darüber treffen, wie erfolgversprechend es ist, einen bestimmten Knoten zu expandieren und dadurch als objektive Referenz für die Einflussfaktoren der späteren Heuristikfunktion dienen. Die Bewertung eines Knotens K kann somit auf Basis des bekannten Teilbaums unterhalb von K erfolgen. Zur Bestimmung dieser Bewertungskriterien wird daher vorab der gesamte Suchbaum vollständig expandiert, unabhängig davon, ob bereits Zielknoten gefunden wurden. Dadurch können, im Gegensatz zu einem normalen Suchdurchlauf, auch mehrere Zielknoten gefunden werden. Basierend auf diesem vollständig expandierten Suchbaum können die einzelnen Knoten, beginnend bei den Blättern, bewertet werden. Da diese Bewertungskriterien das Ergebnis einer idealen Heuristikfunktion darstellen, die Kenntnis über

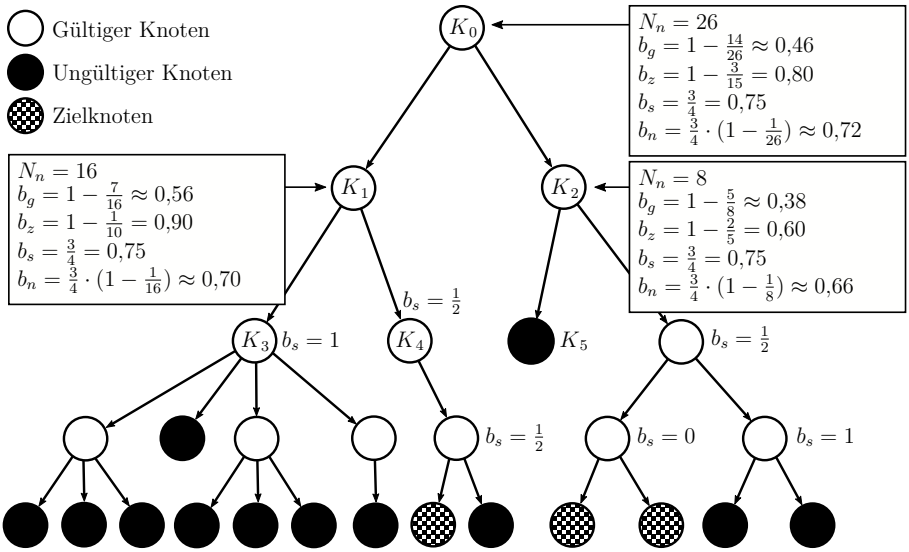


Abbildung 4.1: Visualisierung der Berechnung der Bewertungsheuristiken

den gesamten Suchbaum hat, werden sie im weiteren Verlauf der Arbeit *Bewertungsheuristiken* genannt. Abbildung 4.1 visualisiert die im Folgenden vorgestellten Bewertungsheuristiken für die Knoten K_0, K_1 und K_2 an einem beispielhaften Suchbaum.

Seien $N_n^g(K)$ die Anzahl der gültigen Nachfolger eines Knotens K im Suchbaum und $N_n(K)$ die Anzahl aller Nachfolger von K, so lässt sich die Bewertungsheuristik

$$b_g(K) = 1 - \frac{N_n^g(K)}{N_n(K)} \tag{4.1}$$

definieren. Die Definition wurde so gewählt, dass ein Wert von 0 den erfolgversprechendsten Knoten repräsentiert und ist damit analog zu den Kosten, die eine Heuristikfunktion berechnen würde. Diese Bewertungsheuristik berücksichtigt jedoch nur die Anzahl der gültigen Knoten und vernachlässigt die Anzahl der gefundenen Zielknoten. Sei $N_n^z(K)$ die Anzahl der Zielknoten unter den Nachfolgern des Knotens K und N_n^b die Anzahl aller Blätter unterhalb des Knotens K, so lässt sich daher eine alternative Bewertungsheuristik definieren durch

$$b_z(K) = 1 - \frac{N_n^z(K)}{N_n^b(K)}\,. \tag{4.2}$$

Die beiden bisher eingeführten Bewertungsheuristiken berücksichtigen lediglich die Anzahl der jeweiligen Knoten im Suchbaum, ignorieren jedoch deren Verteilung im Suchbaum. An dieser Stelle seien daher zwei weitere Bewertungsheuristiken genannt, die im Rahmen dieser Arbeit in Betracht gezogen werden. Eine dieser Bewertungsheuristiken lässt sich über die Betrachtung der Wahrscheinlichkeit definieren, einen Zielknoten zu erreichen. Unter der Annahme, dass bei der Suche in jedem Knoten zufällig und mit gleicher Wahrscheinlichkeit einer der Nachfolgerknoten gewählt wird, lässt sich für jeden Knoten K eine Wahrscheinlichkeit berechnen, mit der bei Expandierung dieses Knotens ein Zielknoten erreicht wird. Damit hier wiederum ein Wert von 0 den erfolgversprechendsten Knoten repräsentiert, wird stattdessen jedoch die Wahrscheinlichkeit $b_s(K)$ verwendet, mit der ein ungültiger Blattknoten erreicht wird. Seien $K_1, K_2, ..., K_{N_k}$ die Kinder von K, so lässt sich diese Wahrscheinlichkeit mittels

$$b_s(K) = \begin{cases} 0 & \text{falls } K \text{ Zielknoten} \\ 1 & \text{falls } K \text{ ungültiger Blattknoten} \\ \sum_{i=1}^{N_k} \frac{b_s(K_i)}{N_k} & \text{sonst} \end{cases} \tag{4.3}$$

rekursiv berechnen. Abbildung 4.1 zeigt die Berechnung dieser Wahrscheinlichkeit an einem Beispiel.

Bei der Bewertungsheuristik b_s ist jedoch zu beachten, dass sie nicht berücksichtigt, wie lange die Suche in den jeweiligen Teilbäumen dauert. Betrachtet man beispielsweise die Knoten K_1 und K_2 in Abbildung 4.1, so fällt auf, dass diese beiden Knoten den selben Wert

$$b_s(K_1) = b_s(K_2) = \frac{3}{4} \tag{4.4}$$

besitzen, obwohl die Suche im Teilbaum von K_1 unter Umständen deutlich länger dauern kann. Um dies zu kompensieren, lässt sich beispielsweise mit der Bewertungsheuristik

$$b_n(K) = b_s(K) \cdot \left(1 - \frac{1}{\max\left(1, \ln(N_n(K))\right)}\right) \tag{4.5}$$

auch die Anzahl der Nachfolger und damit die erwartete Anzahl an zu generierenden Knoten zur Erreichung eines Zielknotens berücksichtigen. Wie bei den anderen Bewertungsheuristiken gibt es jedoch auch Konstellationen im Suchbaum, für die diese Definition nicht vorteilhaft ist. Betrachtet man in Abbildung 4.1 beispielsweise die beiden Knoten K_3 und K_5, die beide nicht zu einem Zielknoten führen, so würde der Knoten K_3 aufgrund der größeren Anzahl an Nachfolgern hier schlechter bewertet als K_5. Dies ist einerseits sinnvoll, da in K_5 schneller erkannt werden kann, dass keine Lösung existiert als in K_3 und daher weniger Zeit bei der Suche

aufgewendet werden muss. Für die Verwendung in einer Heuristikfunktion ist dies jedoch nicht wünschenswert, da K_3 im Gegensatz zu K_5 gültig ist und daher ohne Kenntnis der Nachfolger eine größere Wahrscheinlichkeit bietet, einen Zielknoten zu erreichen. In Abschnitt 4.6.1 wird die Eignung der verschiedenen Bewertungsheuristiken untersucht und verglichen.

4.2 Allgemeine Suchstrategien und Einflussfaktoren

Dieser Abschnitt beschreibt zunächst allgemeine Suchstrategien, die nicht vom Typ der jeweiligen Komponente abhängen. Damit lassen sich diese für alle Knoten auf dieselbe Weise realisieren, unabhängig davon, ob ein Knoten beispielsweise ein Werkstück oder eine Greifpose repräsentiert. Da in dieser Arbeit ein heuristisches Suchverfahren entwickelt wird, werden diese Suchstrategien durch entsprechende Einflussfaktoren für die Heuristikfunktion umgesetzt und in den folgenden Abschnitten beschrieben.

4.2.1 Bevorzugung von Knoten mit geringem Abstand zu einem Zielknoten

Bei einem heuristischen Suchverfahren soll die Heuristikfunktion die zu erwartenden Kosten zur Erreichung eines Zielknotens abschätzen. Da die Nachfolger des betrachteten Knotens jedoch noch nicht erzeugt wurden, ist über diese zunächst nichts bekannt. Im Gegensatz zu vielen theoretischen Beispielen in der Literatur für heuristische Suchverfahren existiert hier auch keine einfache, beispielsweise geometrische Beziehung zwischen den einzelnen Knoten, die für die Heuristikfunktion herangezogen werden könnte. Daher ist der Abstand d_Z des Knotens zu einem potenziellen Zielknoten unter seinen Nachfolgern ein wichtiges Kriterium für die Wahrscheinlichkeit, unterhalb des betrachteten Knotens in möglichst wenig Schritten einen Zielknoten zu erreichen. Da die Bäume in der hier untersuchten Anwendung überwiegend eine konstante Tiefe besitzen, kann ein linearer Zusammenhang zwischen d_Z und dem Abstand d_W des Knotens zur Wurzel des Suchbaums angenommen werden. Sei d_{max} die maximal zu erwartende Tiefe des Suchbaums, so lässt sich dieser Zusammenhang durch

$$d_Z = d_{max} - d_W \tag{4.6}$$

ausdrücken. Der Abstand d_W zur Wurzel des Suchbaums kann sehr einfach und effizient bei der Generierung des Suchbaums mitgezählt und in den einzelnen Knoten gespeichert werden. Der

Wert für d_{max} lässt sich anhand der Wissensbasis bestimmen. Die Kosten zur Erreichung eines Zielknotens lassen sich somit durch

$$c_n = \frac{d_Z}{d_{max}} = 1 - \frac{d_W}{d_{max}} \tag{4.7}$$

abschätzen.

4.2.2 Vermeidung einer Konzentration der Suche auf einzelne Knoten

Eine mögliche Ursache für Ineffizienz des Suchverfahrens ist, wenn sich die Suche fälschlicherweise auf einen vielversprechend erscheinenden Knoten konzentriert, wie auf den Knoten K_3 in Abbildung 4.1. Dies kann vorkommen, wenn die Heuristikfunktion geringe Kosten abschätzt, sich jedoch kein Zielknoten unter den Nachfolgern des Knotens befindet. In solch einem Fall kann schneller eine Lösung gefunden werden, wenn die Suche an einer anderen Stelle im Suchbaum, beispielsweise an einem anderen Werkstück, fortgesetzt wird. Eine Möglichkeit, dieses Problem zu vermeiden, ist es, die geschätzten Kosten in der Heuristikfunktion eines Knotens zu erhöhen, je mehr ungültige Knoten unter seinen Nachfolgern evaluiert wurden, ohne dass ein Zielknoten gefunden wurde. Dies lässt sich durch zusätzliche dynamische Kosten c_d erreichen, die in der Heuristikfunktion berücksichtigt werden. Diese dynamischen Kosten werden dabei mittels der in Abschnitt 4.1 beschriebenen Bewertungsheuristik b_g definiert als

$$c_d = b_g\,. \tag{4.8}$$

Die anderen in Abschnitt 4.1 vorgestellten Bewertungsheuristiken lassen sich an dieser Stelle nicht verwenden, da diese die Kenntnis der Zielknoten voraussetzen. Bei der Greifplanung stehen während der Suche jedoch noch keine Zielknoten zur Verfügung, da die Suche beim Erreichen des ersten Zielknotens sofort abbricht. Weiterhin ist zu beachten, dass die Erfolgswahrscheinlichkeit im Gegensatz zu Abschnitt 4.1 nur auf einem unvollständig erstellten Suchbaum berechnet wird und daher jeweils nur den aktuellen Zustand der Suche beschreibt.

Da der Baum im Laufe der Suche weiter wächst und neue Knoten generiert werden, müssen die dynamischen Kosten aller übergeordneten Knoten bei jeder Erstellung eines neuen Knotens aktualisiert (*reevaluiert*) werden. Zu beachten ist, dass die Knoten dadurch ebenfalls erneut in die Vorrangwarteschlange einsortiert werden müssen, was einen zusätzlichen Rechenaufwand darstellt. Wird die Berechnung der dynamischen Kosten für den Knoten K im Suchschritt i jedoch mittels

$$c_{\tilde{d}}(K, i) = \max\left(c_d(K, i), c_{\tilde{d}}(K, i-1)\right) \tag{4.9}$$

definiert, so entstehen dynamische Kosten für jeden Knoten, die im Verlauf der Suche monoton steigen. Es genügt nun, bei der Entnahme des ersten Knotens der Vorrangwarteschlange zu prüfen, ob sich der Wert der Heuristikfunktion seit der Einsortierung in die Vorrangwarteschlange geändert hat. Wenn dies nicht der Fall ist, handelt es sich noch immer um den Knoten mit den geringsten Kosten, da sich die Kosten aller anderen Knoten lediglich erhöht haben können. Falls sich der Wert der Heuristikfunktion geändert hat, muss der Knoten erneut in die Vorrangwarteschlange einsortiert werden und es wird der nächste Knoten der Vorrangwarteschlange betrachtet. Dadurch können Knoten, deren Kosten sich nicht erhöht haben, mit jeder erneuten Einsortierung eines anderen Knotens vorrücken.

Durch die Verwendung der dynamischen Kosten $c_{\tilde{d}}$ muss ein Knoten somit nur neu einsortiert werden, wenn er an der Spitze der Vorrangwarteschlange angekommen ist und nicht direkt bei jeder Evaluierung eines seiner Nachfolger. Im Folgenden werden daher die dynamischen Kosten $c_{\tilde{d}}$ verwendet. Zu beachten ist, dass dieser Wert nur für Knoten berechnet werden kann, die mindestens einen Nachfolger besitzen. Um starke Variationen aufgrund einer sehr geringen Anzahl N_n an Nachfolgern eines Knotens K zu vermeiden, wird $c_{\tilde{d}}$ in der Heuristikfunktion von K nur berücksichtigt, falls

$$N_n \geq N_d^{min}, \tag{4.10}$$

wobei N_d^{min} einen festen Schwellwert darstellt.

4.2.3 Vermeidung der Abweichung von der Ideallage bei Freiheitsgraden

In Abschnitt 3.1.1 wurde das Konzept von Freiheitsgraden eingeführt. In vielen Fällen ist es nicht relevant, welcher konkrete Wert für einen Freiheitsgrad verwendet wird und wie das Werkstück somit gegriffen wird. Bei rotationssymmetrischen Werkstücken sind diese Griffe beispielsweise nicht einmal unterscheidbar. Es gibt jedoch auch Greifposen, bei denen ein gewisser Freiheitsgrad zwar gegeben ist, ein Griff möglichst nahe an der definierten Greifpose jedoch bevorzugt wird, beispielsweise da die Stabilität des Griffs dort am höchsten ist. Auch bei Zusatzachsen im Greifer möchte ein Anlagenbediener möglicherweise starke Abweichungen von der Ausgangslage vermeiden, um die Zeit für die Positionierung der Greiferachse zu minimieren. Um diese Anforderungen zu erfüllen, erhält jeder Knoten im Suchbaum einen dafür vorgesehenen Kostenwert c_{dof}, der in der Heuristikfunktion berücksichtigt werden kann. Seien α_{min} und α_{max} die festgelegten minimal bzw. maximal erlaubten Werte des Freiheitsgrads, dann wird beim Expandieren eines

Freiheitsgrads im Suchbaum dieser Kostenwert in den erzeugten Nachfolgerknoten entsprechend gesetzt und ergibt sich aus

$$c_{dof} = \frac{|\alpha|}{\max\left(|\alpha_{min}|, |\alpha_{max}|\right)}, \tag{4.11}$$

mit dem für den entsprechenden Knoten verwendeten Wert α des Freiheitsgrads.

In diesem Abschnitt wurden die Einflussfaktoren für die Heuristikfunktion vorgestellt, die bei allen Knoten des Suchbaums gleichermaßen berücksichtigt werden können. Insbesondere der geschätzte Abstand zu einem Zielknoten ist für eine schnelle Suche von zentraler Bedeutung. Um die Suche auch durch die Eigenschaften bestimmter Komponenten zu beschleunigen, werden im folgenden Abschnitt Einflussfaktoren vorgestellt, die nur bei Knoten eines bestimmten Typs zum Einsatz kommen.

4.3 Spezifische Einflussfaktoren für die Komponenten der Wissensbasis

Neben den allgemeinen Einflussfaktoren, die im vorherigen Abschnitt vorgestellt wurden, gibt es einige Einflussfaktoren, die nur bei Knoten eines bestimmten Typs berücksichtigt werden. Diese können die Suche durch spezifische Eigenschaften der Komponenten beschleunigen, wie zum Beispiel durch die Berücksichtigung der Position von Werkstücken innerhalb des Ladungsträgers. Einige Ansätze dieses Abschnitts wurden in Spenrath et al. (2012) und Spenrath et al. (2017b) diskutiert.

4.3.1 Konfidenzwert der Objektlageerkennung

Die meisten Algorithmen für die Bestimmung von Objektlagen ermitteln für jede Lage einen Konfidenzwert, der angibt, wie sicher ein bestimmtes Werkstück erkannt wurde. Ein niedriger Konfidenzwert der Objektlageerkennung kann verschiedene Ursachen haben. Einerseits kann dies darauf hindeuten, dass das erkannte Werkstück durch ein anderes Werkstück teilweise verdeckt wird. Dieser Fall ist im Allgemeinen unproblematisch, kann jedoch dazu führen, dass bei der Entnahme andere Werkstücke bewegt werden. Eine weitere mögliche Ursache ist, dass sich an der geschätzten Lage kein Werkstück befindet, sondern dieses fälschlicherweise erkannt wurde. Ein Versuch, dieses falsch erkannte Werkstück zu greifen, würde somit in einem Fehlgriff resultieren. Schließlich kann auch eine ungenaue Lagebestimmung einen niedrigen Konfidenzwert verursachen. Auch in diesem Fall besteht eine erhöhte Wahrscheinlichkeit, dass der Griff fehlschlägt. Um Fehlgriffe zu vermeiden, ist es daher empfehlenswert, bevorzugt Werkstücke zu greifen, die einen

hohen Konfidenzwert besitzen. Sei $R_l \in [0, 1]$ ein solcher Konfidenzwert, wobei ein höherer Wert eine zuverlässigere Erkennung repräsentiert, dann lässt sich ein Kostenwert als

$$c_l = 1 - R_l \tag{4.12}$$

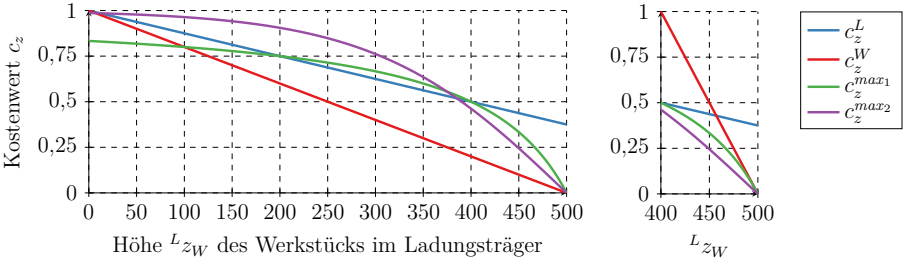
definieren und somit in der Heuristikfunktion verwenden. Der dadurch geringere Anteil an Fehlgriffen führt zu einer geringeren mittleren Taktzeit und einer höheren Verfügbarkeit.

Eine Abweichung zwischen der ermittelten Werkstücklage und der Punktwolke führt dazu, dass die Kollisionsprüfung für Greiferelemente an einer versetzten Position durchgeführt wird. Dadurch kann es vorkommen, dass eine Kollision zwischen diesen Greiferelementen (beispielsweise den Greiferfingern oder eines Saugnapfs) und den Sensordaten des Werkstücks selbst detektiert wird und Griffe dadurch richtigerweise als ungültig deklariert werden. Aus diesem Grund kann die Suche beschleunigt werden, wenn solche ungenau erkannten Werkstücke, an denen häufig kein valider Griff existiert, bei der Suche vermieden werden. Die Berücksichtigung des Konfidenzwerts in der Heuristikfunktion kann somit, neben einer verringerten Fehlgriffrate und der daraus resultierenden Verkürzung der mittleren Taktzeit, auch zu einer schnelleren Suche führen. Eine schnellere Suche trägt, wie in Abschnitt 1.3 bereits erwähnt, ebenfalls zu einer kürzeren Taktzeit bei.

4.3.2 Vertikale Position des Werkstücks im Ladungsträger

Ein naheliegender Indikator für die Wahrscheinlichkeit, an einem Werkstück einen geeigneten Griff zu finden, ist die vertikale Position des Werkstücks im Ladungsträger. Ein tief im Ladungsträger liegendes Werkstück wird häufig von anderen Werkstücken verdeckt und besitzt eine, durch höher liegende Werkstücke, erschwerte Zugänglichkeit. Daher ist es unwahrscheinlicher, dort einen geeigneten Griff zu finden als an höher liegenden Werkstücken. Wie in Abschnitt 1.2 gezeigt wurde, wird diese Tatsache auch in anderen Ansätzen aus dem Stand der Forschung genutzt, um einen geeigneten Griff zu bestimmen. Das Greifen von Werkstücken, die von anderen Werkstücken überlagert sind, führt zudem zu höheren Belastungen am Werkstück und am Greifer und erhöht somit die Gefahr von Beschädigungen. Zudem kann durch das Greifen von tief liegenden Werkstücken ein ungleicher Füllstand im Ladungsträger entstehen, was die Werkstücksuche und Greifplanung für die folgenden Entnahmen erschweren kann. Aus diesen Gründen wird die vertikale Position des Werkstücks (im folgenden auch Werkstückhöhe genannt) bei der Suche berücksichtigt.

Um einen Kostenwert für die Werkstückhöhe zu berechnen, gibt es verschiedene Möglichkeiten. Eine Variante ist die Verwendung der Position des Werkstücks im Ladungsträgerkoordinatensys-



(a) ${}^{L}z_W^{min} = 0$ (b) ${}^{L}z_W^{min} = 400$

Abbildung 4.2: Vergleich der Funktionen zur Berechnung des Kostenwerts für die Werkstückhöhe mit ${}^{L}z_W^{max} = 500$, $d_z^L = 800$ und $z_\Delta^{exp} = 200$. Dargestellt ist die Situation für zwei verschiedene Werte für die Werkstückhöhe ${}^{L}z_W^{min}$ des niedrigsten erkannten Werkstücks.

tem relativ zur Höhe des Ladungsträgers. Seien ${}^{L}z_W$ der z-Wert der Position des betrachteten Werkstücks im Ladungsträgerkoordinatensystem und d_z^L die Höhe des Ladungsträgers, so lässt sich ein Kostenwert als

$$c_z^L = \begin{cases} 0 & \text{für } {}^{L}z_W > d_z^L \\ 1 - \dfrac{{}^{L}z_W}{d_z^L} & \text{sonst} \end{cases} \tag{4.13}$$

definieren. Diese Definition hat allerdings den Nachteil, dass die Kosten in einem leerer werdenden Ladungsträger tendenziell steigen. Das führt dazu, dass in einem fast leeren Ladungsträger, alle verbleibenden Werkstücke sehr hohe Kosten besitzen, obwohl kein höher liegendes Werkstück mit niedrigeren Kosten zur Verfügung steht, das die Entnahme der unten liegenden Werkstücke erschwert. Ein weiterer Nachteil dieser Definition ist, dass der Unterschied zwischen zwei erkannten Werkstücken bezogen auf die gesamte Höhe des Ladungsträgers teilweise sehr gering ist. Diese Nachteile sind in Abbildung 4.2 zu erkennen. Die Abbildung zeigt für ein theoretisches Beispielszenario, welche Kosten, mit den in diesem Abschnitt vorgestellten Varianten der Kostenfunktion, für verschiedene Werkstückhöhen ${}^{L}z_W$ berechnet werden.

Eine Alternative zu der Definition in Gleichung 4.13 ist die Berücksichtigung anderer erkannter Werkstücke. Sei ${}^{L}z_W^{min}$ die Werkstückhöhe des niedrigsten erkannten Werkstücks und ${}^{L}z_W^{max}$ die Werkstückhöhe des höchsten erkannten Werkstücks, so lässt sich der Kostenwert

$$c_z^W = \begin{cases} 0 & \text{für } {}^{L}z_W^{min} = {}^{L}z_W^{max} \\ 1 - \dfrac{{}^{L}z_W - {}^{L}z_W^{min}}{{}^{L}z_W^{max} - {}^{L}z_W^{min}} & \text{sonst} \end{cases} \tag{4.14}$$

berechnen. Bei dieser Definition werden geringe Unterschiede in der Werkstückhöhe deutlich stärker berücksichtigt. Dieser starke Bezug auf die anderen Werkstücke kann jedoch auch nachteilig wirken. Werden beispielsweise nur zwei Werkstücke detektiert, die sich in ihrer Werkstückhöhe kaum unterscheiden, so bekommt das höher liegende Werkstück einen Wert von $c_z^W = 0$ und das niedrigere Werkstück einen Wert von $c_z^W = 1$, obwohl es kaum eine Rolle spielt, welches Werkstück gegriffen wird. Diese Eigenschaft, die in Abbildung 4.2(b) zu erkennen ist, macht die Kombination in der Heuristikfunktion mit anderen Einflussfaktoren gegebenenfalls problematisch.

Aufgrund der Nachteile der beiden vorgestellten Definitionen, sei im Folgenden eine dritte Definition genannt, die auf der maximal zu erwartenden Höhendifferenz z_Δ^{exp} zwischen zwei Werkstücken basiert. Diese Höhendifferenz ist abhängig vom zu vereinzelnden Werkstück. Im weiteren Verlauf der Arbeit wird hierfür ein Viertel der Höhe des Ladungsträgers angenommen. Dies trifft in etwa auf die beiden simulierten Szenarien zu und stellt auch für viele weitere Anwendungen eine geeignete Abschätzung dar. Es sei daher die Kostenfunktion

$$c_z^{max_1} = 1 - \frac{1}{\frac{{}^L z_W^{max} - {}^L z_W}{0{,}5 \cdot z_\Delta^{exp}} + 1} \tag{4.15}$$

mit

$$z_\Delta^{exp} = 0{,}25 \cdot d_z^L \tag{4.16}$$

definiert. Durch diese Funktion hat das höchste erkannte Werkstück einen Kostenwert von 0 und ein Werkstück, das um die halbe erwartete maximale Höhendifferenz niedriger im Ladungsträger liegt einen Kostenwert von 0,5. Weiterhin können beliebig tief liegende Werkstücke durch die Funktion abgebildet werden. Eine Alternative auf Basis der Sigmoid-Funktion (Mitchell 1997) lässt sich definieren als

$$c_z^{max_2} = 2 \cdot \mathrm{sig}\left(\frac{{}^L z_W^{max} - {}^L z_W}{0{,}5 \cdot z_\Delta^{exp}}\right) - 1 \tag{4.17}$$

mit der Sigmoid-Funktion

$$\mathrm{sig}(t) = \frac{1}{1 + e^{-t}}\,. \tag{4.18}$$

Der Zusammenhang zwischen der Werkstückhöhe und den beiden Kostenwerten $c_z^{max_1}$ und $c_z^{max_2}$ ist ebenfalls in Abbildung 4.2 dargestellt.

4.3.3 Horizontale Position des Werkstücks im Ladungsträger

Neben der vertikalen Position des Werkstücks im Ladungsträger ist auch die horizontale Position ein naheliegender Indikator für die Erfolgswahrscheinlichkeit, einen geeigneten Griff an diesem

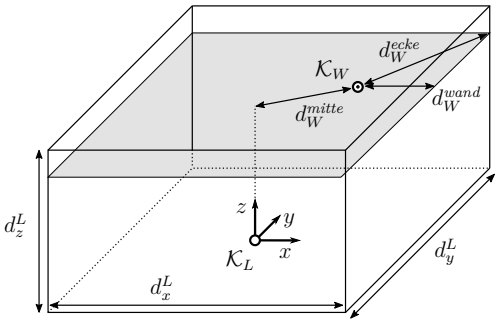


Abbildung 4.3: Verschiedene Ansätze zur Berücksichtigung der horizontalen Werkstückposition

Werkstück zu finden. Aufgrund der größeren Störkontur ist es in der Nähe der Seitenwände des Ladungsträgers deutlich schwieriger, einen kollisionsfreien Griff zu bestimmen. Auch hier gibt es verschiedene Möglichkeiten, eine geeignete Kostenfunktion festzulegen. Sei ${}^{L}x_W$ die x-Koordinate und ${}^{L}y_W$ die y-Koordinate des Werkstückkoordinatensystems $\mathcal{K}_W$ im Bezug auf das Ladungsträgerkoordinatensystem $\mathcal{K}_L$. Sei weiterhin d_x^L die Länge des Ladungsträgers in x-Richtung und d_y^L die Breite des Ladungsträgers in y-Richtung. Im Folgenden werden damit drei verschiedene Kostenfunktionen definiert und in Abbildung 4.3 schematisch visualisiert.

- Basierend auf dem horizontalen euklidischen Abstand des Werkstücks zur Mitte des Ladungsträgers:

$$c_{xy}^{mitte} = \frac{d_W^{mitte}}{0{,}5 \cdot \sqrt{{d_x^L}^2 + {d_y^L}^2}} \tag{4.19}$$

  mit

$$d_W^{mitte} = \sqrt{{{}^{L}x_W}^2 + {{}^{L}y_W}^2} \tag{4.20}$$

- Basierend auf dem horizontalen Abstand zur nächstgelegenen Seitenwand des Ladungsträgers:

$$c_{xy}^{wand} = 1 - \frac{d_W^{wand}}{0{,}5 \cdot \min\left(d_x^L, d_y^L\right)} \tag{4.21}$$

  mit

$$d_W^{wand} = \min\left(\frac{d_x^L}{2} - {}^{L}x_W, \frac{d_x^L}{2} + {}^{L}x_W, \frac{d_y^L}{2} - {}^{L}y_W, \frac{d_y^L}{2} + {}^{L}y_W\right) \tag{4.22}$$

  analog zu Abschnitt 3.2.3.

- Basierend auf dem horizontalen euklidischen Abstand zur nächstgelegenen Ecke des Ladungsträgers:

$$c_{xy}^{ecke} = 1 - \frac{d_W^{ecke}}{0{,}5 \cdot \sqrt{{d_x^L}^2 + {d_y^L}^2}} \tag{4.23}$$

mit

$$d_W^{ecke} = \min\left(\sqrt{\left(\frac{d_x^L}{2} - {}^L x_W\right)^2 + \left(\frac{d_y^L}{2} - {}^L y_W\right)^2}, \sqrt{\left(\frac{d_x^L}{2} - {}^L x_W\right)^2 + \left(\frac{d_y^L}{2} + {}^L y_W\right)^2}, \sqrt{\left(\frac{d_x^L}{2} + {}^L x_W\right)^2 + \left(\frac{d_y^L}{2} - {}^L y_W\right)^2}, \sqrt{\left(\frac{d_x^L}{2} + {}^L x_W\right)^2 + \left(\frac{d_y^L}{2} + {}^L y_W\right)^2}\right) \tag{4.24}$$

Diese Kosten können nun in der Heuristikfunktion der Werkstückknoten verwendet werden und werden in Abschnitt 4.6.3 untersucht.

4.3.4 Orientierung von Komponenten

Wie in Abschnitt 3.2.3 beschrieben, lässt sich für einige Komponenten der Suche, insbesondere für TCPs oder Verbindungsposen des Greifers, eine bevorzugte Orientierung relativ zum Ladungsträgerkoordinatensystem $\mathcal{K}_L$ festlegen. Unter der Annahme, dass diese Orientierung geeignet definiert wurde, lässt sich ein Kostenwert mittels

$$c_{ori} = \begin{cases} 1 & \text{für } \gamma_{max} = 0 \\ \dfrac{\gamma}{\gamma_{max}} & \text{sonst} \end{cases} \tag{4.25}$$

definieren, mit der berechneten Abweichung $\gamma \in [0, \pi]$ von der bevorzugten Orientierung und der maximal erlaubten Abweichung $\gamma_{max} \in [0, \pi]$. Dabei ist zu beachten, dass c_{ori} nie größer als 1 sein kann, da der Knoten für ungültig erklärt würde, falls $\gamma > \gamma_{max}$ (vergleiche Abschnitt 3.3.4).

4.3.5 Kostenwert für mehrfach expandierte Knoten

Wie in Abschnitt 3.2.4 beschrieben, sollte die Heuristikfunktion bei mehrfach zu expandierenden Freiheitsgraden sicherstellen, dass die Kosten des Freiheitsgradknotens mit jeder Expansion ansteigen. Dadurch soll vermieden werden, dass ein Freiheitsgradknoten immer weiter expandiert wird, obwohl es zielführender wäre, die Suche zunächst an einer anderen Stelle im Suchbaum

fortzuführen. Zur Verbesserung des Verständnisses wurde in Abschnitt 3.2.4 bereits ein entsprechender Kostenwert c_{exp} eingeführt, der somit in der Heuristikfunktion verwendet werden kann.

4.3.6 Anwendungsspezifische statische Prioritäten für Greifposen

In industriellen Anwendungen gibt es Gründe, bestimmte Greifposen an Werkstücken gegenüber anderen Greifposen zu bevorzugen, beispielsweise weil der entsprechende Griff stabiler ist, mehr Toleranz beim Greifen zulässt oder sich das Werkstück mit diesem Griff zuverlässiger ablegen lässt. Eine solche Priorisierung der Greifposen sollte somit auch durch die Greifplanung berücksichtigt werden können. Diese Prioritäten lassen sich nicht universell festlegen, sondern sind anwendungsspezifisch zu definieren. Dafür wird für jede Greifpose ein Kostenwert $c_{prio} \in [0, 1]$ eingeführt, der beispielsweise manuell vom Inbetriebnehmer festgelegt werden kann.

Neben einer manuellen Definition auf Basis von Expertenwissen oder Erfahrungen der Bediener könnten diese Prioritäten der Greifposen auch durch Greifversuche bestimmt werden. Diese automatische Bestimmung der Prioritäten ließe sich beispielsweise durch reale Greifversuche erreichen, bei denen nach einem Greifversuch mittels Sensoren erfasst wird, ob ein Griff erfolgreich war und wie groß die Lageabweichung von der theoretischen Greifpose war. Alternativ kann dies auch effizienter auf Basis von Computersimulationen erfolgen. Unter Verwendung dieser Trainingsdaten ließe sich daraufhin die optimale Priorität für jede Greifpose bestimmen, beispielsweise mittels überwachter Lernverfahren. Schließlich ist es durch die Priorität ebenso möglich, das Suchverfahren selbst zu beschleunigen, indem Greifposen, die statistisch gesehen öfters zu erfolgreichen Griffen führen, höher priorisiert werden.

In diesem Abschnitt wurden verschiedene Einflussfaktoren vorgestellt, die bei Knoten eines bestimmten Typs zum Einsatz kommen. Diese wurden auf Basis von Erfahrungswerten definiert und werden in Abschnitt 4.6 untersucht. Im folgenden Abschnitt wird zunächst ein alternativer Ansatz entwickelt, bei dem diese Einflussfaktoren maschinell gelernt werden.

4.4 Künstliche neuronale Netze für die Heuristikfunktion

Aktuell wird in vielen Bereichen an unterschiedlichen Ansätzen mit maschinellem Lernen, insbesondere mit künstlichen neuronalen Netzen, geforscht. Nicht alle Ansätze schaffen den Sprung in eine industrielle Anwendung. Ein möglicher Grund dafür ist, dass diese Verfahren

häufig nicht einfach nachvollziehbar sind und das Verhalten der Anlage daher nicht immer vorhersehbar und erklärbar ist. Für die Werkstückvereinzelung bedeutet das beispielsweise, dass Werkstücke unter Umständen an unerwarteten und ungewollten Stellen gegriffen werden könnten oder dass andere unerwünschte Eigenschaften gelernt werden. Aus diesem Grund soll die Berechenbarkeit und die Robustheit des in dieser Arbeit vorgestellten Ansatzes auch mit dem Einsatz der neuronalen Netze beibehalten werden. Es sollen nach wie vor die in Kapitel 3 vorgestellten Prüfungen durchgeführt werden, so dass der Ansatz mit neuronalen Netzen keine Greiflösungen generiert, die ohne Einsatz dieser Netze nicht entstanden wären. Vielmehr sollen die neuronalen Netze in dieser Arbeit dazu dienen, die Geschwindigkeit der Suche zu steigern.

Die bisher vorgestellten Einflussfaktoren beruhen auf manuell festgelegten Einschätzungen und Erfahrungen, welche Eigenschaften einer Komponente die Wahrscheinlichkeit beeinflussen, unterhalb des zugehörigen Knotens einen geeigneten Griff zu finden. Es können allerdings weitere Eigenschaften der einzelnen Komponenten existieren, deren Einfluss auf die Wahrscheinlichkeit, eine gültige Lösung zu finden, nicht ersichtlich ist. Um diese Eigenschaften ebenfalls in der Heuristikfunktion berücksichtigen zu können, wird in diesem Abschnitt ein Ansatz entwickelt, deren Einfluss mittels künstlicher neuronaler Netze (vgl. Abschnitt 2.3) zu lernen. Einige Ansätze dieses Abschnitts orientieren sich an Spenrath et al. (2018).

4.4.1 Architektur und Grundprinzip mit überwachtem Lernen

Um die spezifischen Eigenschaften einzelner Komponenten berücksichtigen zu können, wird für jede Komponente der Wissensbasis, beispielsweise für jedes unterschiedliche Werkstück, ein eigenes neuronales Netz trainiert. Die neuronalen Netze für die einzelnen Komponenten werden in den jeweiligen Knoten im Graphen der Wissensbasis vorgehalten. Abbildung 4.4 visualisiert die Speicherung und Verwendung der neuronalen Netze während des Suchverfahrens. Die Nutzung eines eigenen neuronalen Netzes für jede Komponente hat den Vorteil, dass die neuronalen Netze spezifische Eigenschaften einzelner Komponenten lernen können. Dadurch werden beispielsweise die Besonderheiten verschiedener Werkstücke oder Greiferelemente berücksichtigt. Das vorgestellte Konzept ist unabhängig von der konkreten Architektur der neuronalen Netze. Erst bei der Analyse des Ansatzes in Abschnitt 4.6.7 wird die dafür verwendete Architektur vorgestellt.

Das Training der neuronalen Netze erfolgt, wie schon für die Berechnung der Bewertungsheuristiken in Abschnitt 4.1 beschrieben, durch eine vollständige Expansion der Suchbäume. Da in jedem Suchbaum mehrere Werkstückknoten und auch eine Vielzahl an Knoten enthalten sind, die beispielsweise Greifposen oder TCPs entsprechen, lässt sich dadurch auch mit wenigen

Situationen eine umfangreiche Menge an Trainingsdaten generieren. Der Ablauf für das Training der neuronalen Netze erfolgt daher mittels folgender Schritte:

1. Für eine Situation wird ein vollständiger Suchbaum berechnet.
2. Für jeden Knoten des Suchbaums wird eine Bewertungsheuristik bestimmt, die auf Basis der Nachfolger des Knotens angibt, wie erfolgversprechend eine Expansion des Knotens tatsächlich ist und im Knoten gespeichert.
3. Diese Bewertungsheuristik, die den gewünschten Ausgabewert des neuronalen Netzes darstellt, wird samt der jeweiligen Eingabedaten des Knotens als Trainingsdatensatz gespeichert.
4. Die Schritte 1 - 3 werden für alle Situationen im Trainingsdatensatz wiederholt.
5. Zum Abschluss werden die neuronalen Netze mit den gesammelten Trainingsdaten trainiert.

Es handelt sich somit um ein überwachtes Lernverfahren (vgl. Abschnitt 2.3.1). Bei der Suche nach einer geeigneten Greiflösung können die neuronalen Netze nun berücksichtigt werden. Wie der Ausgabewert der neuronalen Netze, der im Folgenden mit c_{nn} bezeichnet wird, in die Berechnung der Heuristikfunktion einbezogen wird, wird in Abschnitt 4.5 beschrieben.

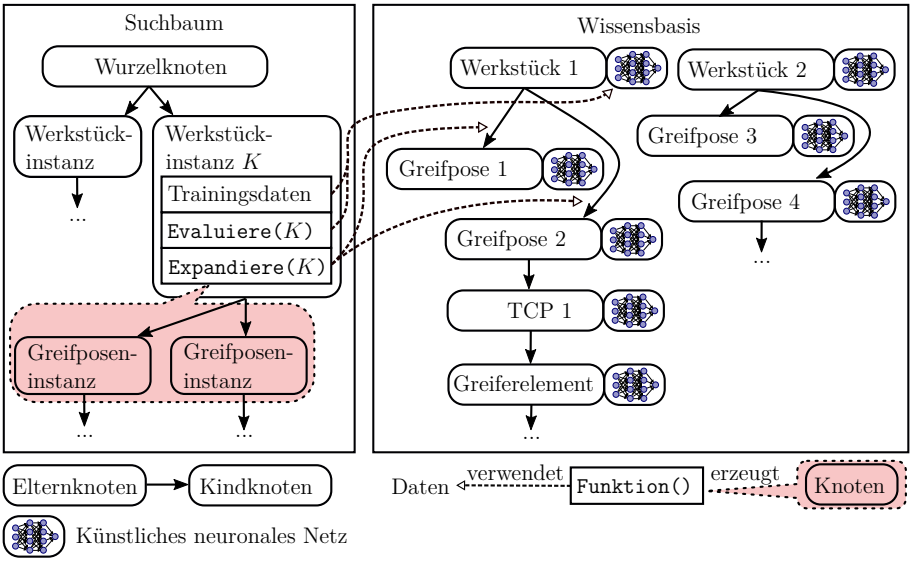


Abbildung 4.4: Konzept des Suchbaums mit neuronalen Netzen

Tabelle 4.1: Eingabewerte für die neuronalen Netze der verschiedenen Komponenten

Komponente	**Eingabewerte für die jeweiligen neuronalen Netze**
Werkstück	• Die Lage des Werkstücks relativ zum Ladungsträgerkoordinatensystem $\mathcal{K}_L$ • Der z-Wert der Position des höchsten erkannten Werkstücks im Ladungsträgerkoordinatensystem $\mathcal{K}_L$ • Der z-Wert der Position des niedrigsten erkannten Werkstücks im Ladungsträgerkoordinatensystem $\mathcal{K}_L$
Greifpose	• Die Lage der Greifpose relativ zum Ladungsträgerkoordinatensystem $\mathcal{K}_L$
Verbindungspose	• Die Lage der Verbindungspose relativ zum Ladungsträgerkoordinatensystem $\mathcal{K}_L$
TCP	• Die Lage des TCPs relativ zum Ladungsträgerkoordinatensystem $\mathcal{K}_L$

4.4.2 Eingabewerte der neuronalen Netze

Da die neuronalen Netze für verschiedene Arten von Komponenten (beispielsweise Werkstücke oder Greifposen) trainiert werden, variieren die Eingabewerte dieser Netze. Tabelle 4.1 führt alle Eingabewerte für die neuronalen Netze der verschiedenen Komponenten auf.

Um die Lagen der verschiedenen Komponenten in den neuronalen Netzen berücksichtigen zu können, müssen diese durch eine endliche Anzahl skalarer Werte ausgedrückt werden. Die Position einer Komponente ist durch die drei Koordinaten x, y und z in einem festgelegten Koordinatensystem eindeutig bestimmt. Für die Rotation existieren, wie in Abschnitt 2.1.1 und in Siciliano et al. (2008) beschrieben, hingegen verschiedene gebräuchliche Repräsentationen. Im weiteren Verlauf des Kapitels werden die in Tabelle 4.2 beschriebenen Repräsentationen für die Berücksichtigung der Rotation von Lagen in neuronalen Netzen verwendet und untersucht.

4.4.3 Ausgabewerte der neuronalen Netze

Um neuronale Netze zu trainieren, werden neben den Eingabewerten auch die gewünschten Ausgabewerte für alle Trainingsdaten benötigt. Da die neuronalen Netze lernen sollen, welche Einflussfaktoren geeignet sind, schnell eine Greiflösung zu finden, können dafür die in Abschnitt 4.1 eingeführten Bewertungsheuristiken als Ausgabewerte verwendet werden. Hierzu sei der Kostenwert c_{nn} definiert, der einer der eingeführten Bewertungsheuristiken entspricht. Dieser Wert kann für alle Knoten im vollständig expandierten Suchbaum berechnet und als Ausgabewert der neuronalen Netze beim Lernen verwendet werden.

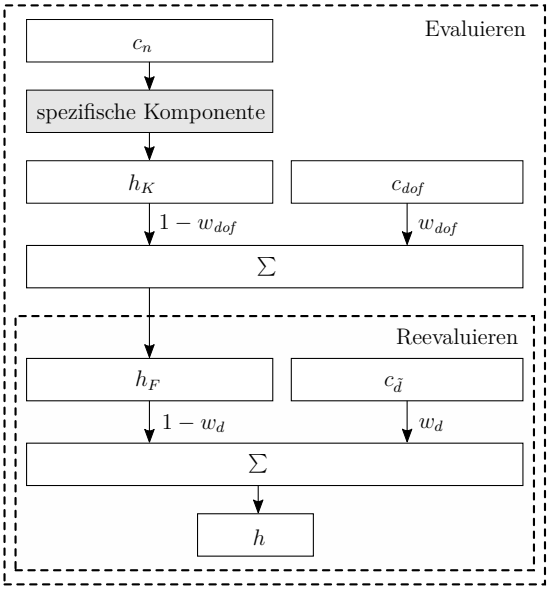
Tabelle 4.2: Auflistung der verwendeten Repräsentationen für Rotationen

Bezeichnung	**Dim.**	**Beschreibung**
Feste Winkel	3	Repräsentation durch die Winkel ψ, θ, ϕ, die die Drehungen um die feststehenden x-, y- und z-Achsen beschreiben. Andere Reihenfolgen und Euler-Winkel sind möglich, sollen in dieser Arbeit jedoch nicht untersucht werden.
Quaternionen	4	Repräsentation durch Einheitsquaternionen
6D	6	Repräsentation durch die beiden ersten Spalten der Rotationsmatrix. Diese wurde in Zhou et al. (2019) als kontinuierliche Repräsentation von Rotationen im dreidimensionalen Raum für die Verwendung in neuronalen Netzen vorgeschlagen. Die Darstellung lässt sich, wie in Zhou et al. (2019) beschrieben, um eine Dimension reduzieren. Dies soll in der vorliegenden Arbeit jedoch nicht untersucht werden.

Die Bewertungsheuristik gibt eine Aussage über die Wahrscheinlichkeit an, schnell zu einer beliebigen Greiflösung zu kommen. Dabei werden alle Greiflösungen als gleichwertig betrachtet. Durch eine geeignete Definition des Ausgabewerts der neuronalen Netze wäre es allerdings auch möglich, Greiflösungen mit gewünschten Eigenschaften zu bevorzugen. Anwendungsfälle für eine solche Vorgehensweise sind beispielsweise die Priorisierung von Greifposen oder die Vermeidung von extremen Winkeln der zusätzlichen Greiferachsen. Auch Eigenschaften der Bewegungsbahn des Roboters könnten dabei berücksichtigt werden, wie beispielsweise die erwartete Dauer der Bewegung zur Minimierung der Taktzeit. Um die Ergebnisse neutral bewerten zu können, beschränkt sich diese Arbeit jedoch auf gleichwertige Greiflösungen.

4.5 Definition und Optimierung der Heuristikfunktion

In den bisherigen Abschnitten dieses Kapitels wurden verschiedene Kosten bzw. Teilheuristiken eingeführt. Für die Heuristikfunktion eines heuristischen Suchverfahrens müssen diese jedoch in einem einzelnen sortierbaren Wert vereint sein. Eine Möglichkeit, alle Teilheuristiken zu berücksichtigen, ist eine Linearkombination der einzelnen Teilheuristiken, wie beispielsweise von Burke et al. (2012) verwendet. Alternativ können die Teilheuristiken priorisiert und der Reihe nach ausgewertet werden, so dass nachfolgende Teilheuristiken nur berücksichtigt werden, falls durch die vorherigen Heuristiken nicht bereits eine Entscheidung getroffen werden konnte (Carrington et al. 2007). Dieser Ansatz lässt sich durch die Verwendung unscharfer Regeln mittels Fuzzylogik erweitern (Asmuni et al. 2005b; Asmuni et al. 2005a). Da die letztgenannten Ansätze entweder zu eingeschränkt oder andererseits relativ komplex sind, werden die Teilheuristiken in



Legende:

c_n	Kosten der Nachfolger des Knotens
h_K	Spezifischer Teil der Heuristikfunktion
c_{dof}	Kosten durch übergeordneten Freiheitsgrad
w_{dof}	Gewichtungsfaktor für c_{dof}
h_F	Statischer Teil der Heuristikfunktion
$c_{\tilde{d}}$	Dynamische Kosten des Knotens
w_d	Gewichtungsfaktor für $c_{\tilde{d}}$
h	Ergebnis der Heuristikfunktion
$\sum$	Summe

Abbildung 4.5: Allgemeine Struktur der Heuristikfunktion

dieser Arbeit durch eine Linearkombination vereint. Aufgrund der unterschiedlichen Arten von Knoten und der Reevaluierung von Knoten geschieht dies mehrstufig.

4.5.1 Allgemeine Struktur der Heuristikfunktion

Dieser Abschnitt beschreibt zunächst die allgemeine Struktur der Heuristikfunktion und stellt zudem in Abbildung 4.5 visuell dar, wie die einzelnen Einflussfaktoren in die Heuristikfunktion einfließen. Einer der wichtigsten Einflussfaktoren für die Heuristikfunktion stellt der geschätzte Abstand zu einem Zielknoten und damit die geschätzten Kosten c_n der Nachfolger dar. Um die spezifischen Einflussfaktoren unterschiedlich stark gewichten zu können, wird c_n zunächst abhängig von der jeweiligen Komponente mit deren Einflussfaktoren zu einem einzelnen Wert h_K kombiniert, der den entsprechenden Knoten bewertet. Wie dies in Abhängigkeit von der jeweiligen Komponente erfolgt, wird in Abschnitt 4.6 beschrieben. Basierend auf h_K und den Kosten c_{dof} für einen eventuellen übergeordneten Freiheitsgrad wird bei der Evaluierung eines Knotens der Wert

$$h_F = h_K \cdot (1 - w_{dof}) + c_{dof} \cdot w_{dof} \tag{4.26}$$

mit $w_{dof} \in [0, 1]$ berechnet, der für einen gegebenen Knoten konstant bleibt. Wenn der Knoten reevaluiert wird, muss dieser Wert daher nicht erneut berechnet werden. Um den finalen Wert der Heuristikfunktion zu bestimmen, fließen schließlich die dynamischen Kosten $c_{\tilde{d}}$ mit ein (siehe Abschnitt 4.2.2). Für die Heuristikfunktion gilt daher

$$h = h_F \cdot (1 - w_d) + c_{\tilde{d}} \cdot w_d \tag{4.27}$$

mit $w_d \in [0, 1]$.

Falls sich die dynamischen Kosten $c_{\tilde{d}}$ ändern, muss somit lediglich die letztgenannte Gleichung erneut berechnet werden. Die konkreten Werte der in den Versuchen dieser Arbeit verwendeten Gewichtungsfaktoren finden sich in Anhang A.2.

4.5.2 Spezifischer Anteil der Heuristikfunktion aller Komponenten der Wissensbasis

Wie in Abschnitt 4.5.1 beschrieben, besitzt die Heuristikfunktion einen spezifischen Teil, der davon abhängt, um welchen Typ von Komponente es sich bei dem jeweiligen Knoten handelt. Dieser Teil der Heuristikfunktion berechnet, basierend auf c_n und auf spezifischen Einflussfaktoren, einen Wert h_K, der den entsprechenden Knoten bewertet. Abhängig vom Typ der jeweiligen Komponenten in der Wissensbasis ist h_K definiert als

$$h_K = \begin{cases} h_K^W & \text{für Werkstücke} \\ h_K^G & \text{für Greifposen} \\ h_K^T & \text{für TCPs} \\ h_K^V & \text{für Verbindungsposen} \\ h_K^{dof} & \text{für Freiheitsgrade} \\ c_n & \text{sonst.} \end{cases} \tag{4.28}$$

Der spezifische Anteil wird in diesem Abschnitt für die verschiedenen Komponententypen mathematisch beschrieben und in Abbildung 4.6 visualisiert. Dabei wird durch die jeweilige Definition sichergestellt, dass der Wert für alle Komponententypen zwischen 0 und 1 liegt und somit $h_K \in [0, 1]$ gilt.

Bei **Werkstücken** wird, wie in den Abschnitten 4.3.2 und 4.3.3 beschrieben, angenommen, dass ihre vertikale und horizontale Position im Ladungsträger einen Einfluss auf die Wahrscheinlichkeit haben, einen gültigen Griff zu finden. Daher werden die dort eingeführten Kosten c_z und c_{xy} im spezifischen Teil der Heuristikfunktion verwendet. Als Alternative zu diesen Werten wird

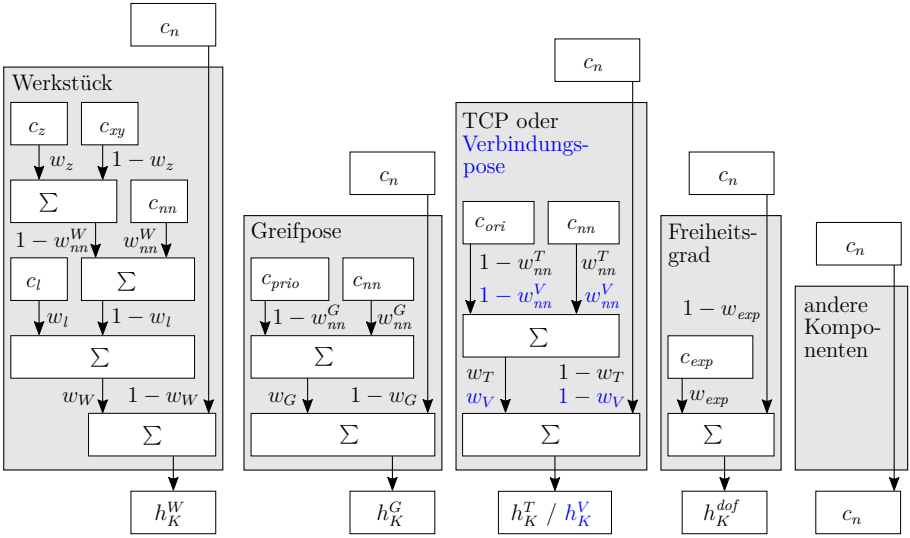


Abbildung 4.6: Komponentenspezifische Teile der Heuristikfunktion

die in Abschnitt 4.4.3 eingeführte Ausgabe eines neuronalen Netzes verwendet. Schließlich werden die auf der Bewertung der Objektlageerkennung basierenden Kosten c_l und die vom erwarteten Abstand zu einem Zielknoten abhängenden Kosten c_n berücksichtigt. Um die einzelnen Einflussfaktoren unabhängig voneinander bewerten und gewichten zu können, werden diese wie in Abbildung 4.6 dargestellt linear interpoliert. Damit ergibt sich folgende Gleichung zur Berechnung des komponentenspezifischen Teils der Heuristikfunktion für Werkstücke:

$$\begin{aligned} h^W_K = \Big(\Big(\Big(\Big(c_z \cdot w_z + c_{xy} \cdot (1 - w_z)\Big) \cdot (1 - w^W_{nn}) + c_{nn} \cdot w^W_{nn}\Big) \\ \cdot (1 - w_l) + c_l \cdot w_l\Big) \cdot w_W + c_n \cdot (1 - w_W)\Big) \end{aligned} \tag{4.29}$$

Bei **Greifposen** fließen einerseits die Priorität mittels c_{prio} sowie andererseits die Ausgabe c_{nn} des neuronalen Netzes in das Ergebnis ein. Die einzelnen Einflussfaktoren werden mittels

$$h^G_K = \Big(c_{prio} \cdot (1 - w^G_{nn}) + c_{nn} \cdot w^G_{nn}\Big) \cdot w_G + c_n \cdot (1 - w_G) \tag{4.30}$$

ebenfalls linear interpoliert.

Bei **TCPs** oder **Verbindungsposen** wird neben der Ausgabe c_{nn} des neuronalen Netzes, die Orientierung mittels c_{ori} in der Heuristikfunktion berücksichtigt. Dadurch berechnet sich h_K für TCPs mittels

$$h_K^T = \left(c_{ori} \cdot (1 - w_{nn}^T) + c_{nn} \cdot w_{nn}^T\right) \cdot w_T + c_n \cdot (1 - w_T) \tag{4.31}$$

und für Verbindungsposen mittels

$$h_K^V = \left(c_{ori} \cdot (1 - w_{nn}^V) + c_{nn} \cdot w_{nn}^V\right) \cdot w_V + c_n \cdot (1 - w_V) \ . \tag{4.32}$$

Bei **Freiheitsgraden** sollte die Anzahl der bereits durchgeführten Expansionen mittels des bereits in Abschnitt 3.2.4 eingeführten Wertes c_{exp} berücksichtigt werden, so dass sich h_K^{dof} folgendermaßen berechnet:

$$h_K^{dof} = c_{exp} \cdot w_{exp} + c_n \cdot (1 - w_{exp}) \ . \tag{4.33}$$

Für alle **anderen Komponenten** gibt es keine spezifischen Einflussfaktoren, so dass die Heuristikfunktion hier lediglich die Kosten der Nachfolger berücksichtigt und somit

$$h_K = c_n \tag{4.34}$$

gilt, wie bereits in Gleichung (4.28) definiert.

4.5.3 Optimierung der Heuristikfunktion

Das Ziel der Heuristikfunktion ist es, auf Basis der Einflussfaktoren abzuschätzen, welche Knoten expandiert werden sollten, sodass schnellstmöglich eine Lösung gefunden wird. Die Vielzahl an Gewichtungsfaktoren in der Heuristikfunktion macht es schwierig, optimale (oder nahezu optimale) Werte für sie zu bestimmen, um dieses Ziel zu erreichen. Um die geeigneten Werte für die Gewichtungsfaktoren nicht manuell, beispielsweise auf Basis von Expertenwissen und Erfahrungen, suchen zu müssen, können Optimierungsverfahren eingesetzt werden. Dabei ist zu beachten, dass zur Bewertung von konkreten Werten für diese Gewichtungsfaktoren, jeweils eine große Menge an unterschiedlichen Situationen ausgewertet werden muss, um ein belastbares Ergebnis zu erhalten. Daher fokussiert sich die Auswahl an Optimierungsverfahren auf Verfahren, die für sehr aufwändige Zielfunktionen geeignet sind und häufig beispielsweise auch zur Optimierung von Hyperparametern in neuronalen Netzen eingesetzt werden. In dieser Arbeit werden folgende Verfahren untersucht:

- **Bayessche Optimierung (BO)** (Mockus 1989): Dieses Optimierungsverfahren erstellt basierend auf einzelnen Messungen ein probabilistisches Modell und verwendet dieses, um die besten Gewichtungsfaktoren für die jeweils nächste Messung zu schätzen. Nach jeder Messung wird das Modell aktualisiert. Es existieren verschiedene Ansätze der bayesschen Optimierung, die insbesondere verschiedene probabilistische Modelle verwenden. In dieser Arbeit werden die folgenden Modelle verwendet:

  - **Gauß-Prozess** (Rasmussen et al. 2006): Ein Gauß-Prozess ist ein stochastischer Prozess, bei dem die einzelnen stochastischen Größen normalverteilt sind. Er kann verwendet werden, um eine Funktion zu definieren, bei der die Zufallsvariablen die Funktionswerte repräsentieren, und somit als Modell für die BO dienen. Für die Implementierung wird das Python-Paket *GPyOpt* (GPyOpt 2018) verwendet.

  - **Tree-structured Parzen Estimator (TPE)** (Bergstra et al. 2011): Dieses Modell verwendet Thresholds, um die einzelnen Messungen für jeden Gewichtungsfaktor in zwei Wahrscheinlichkeitsverteilungen zu unterteilen. Aus diesen wird der nächste zu testende Gewichtungsfaktor bestimmt. Die einzelnen Gewichtungsfaktoren werden baumartig strukturiert. Für die Implementierung des TPE wird das Python-Paket *hyperopt* (Bergstra et al. 2013; Bergstra et al. 2015) verwendet.

- **Evolutionäre Algorithmen**: Bei diesem Optimierungsverfahren werden alle Gewichtungsfaktoren als sogenannte *Gene* eines *Individuums* repräsentiert. Ein Individuum stellt einen vollständigen Parametersatz dar. Der Ablauf des Algorithmus entspricht im Wesentlichen einem *genetischen Algorithmus* (Mitchell 1998), jedoch werden die Gewichtungsfaktoren reellwertig kodiert, anstatt binäre Zeichenketten zu verwenden. Der in dieser Arbeit vorgestellte Algorithmus ist in Abbildung 4.7 dargestellt und wird im Folgenden beschrieben.

Zu Beginn des evolutionären Algorithmus werden zehn zufällige Individuen erstellt und im Anschluss die folgenden Schritte so lange wiederholt, bis eine vorgegebene Anzahl an Generationen erreicht ist. Eine Fitnessfunktion weist jedem Individuum einen Fitnesswert zu. Diese Fitnessfunktion basiert auf der mittleren Anzahl N_{exp} expandierter Knoten und ist wie folgt definiert:

$$f_k = 1 - \mathrm{sig}\left(\frac{N_{exp}(k)}{\min_i(N_{exp}(i))} \cdot r_{exp}\right) \tag{4.35}$$

mit der Sigmoid-Funktion

$$\mathrm{sig}(t) = \frac{1}{1+e^{-t}} \tag{4.36}$$

Durch den Wert r_{exp} kann festgelegt werden, wie stark der Einfluss von N_{exp} auf die Unterschiede zwischen den einzelnen Fitnesswerten ist. Ein hoher Wert von r_{exp} führt dazu, dass auch

Daten : Maximale Anzahl N_G an Generationen
Ergebnis : Bester Parametersatz (Individuum)
1 $i = 0$;
2 Zufällige Generierung von 10 Individuen;
3 Bestimmung der Fitness für alle Individuen;
4 **solange** $i < N_G$ **wiederhole**
5 **Selektion und Rekombination:** Fitnessproportionale Selektion von jeweils zwei Individuen und paarweise Rekombination zu 10 Nachkommen;
6 **Mutation:** Zufällige Veränderung einzelner Gene in den Nachkommen;
7 Bestimmung der Fitness für alle Nachkommen;
8 Zusammenstellung der neuen Generation aus den alten und neuen Individuen;
9 **Aussterben:** Entfernung der Individuen mit der geringsten Fitness, so dass 10 Individuen übrig bleiben;
10 $i = i + 1$;
11 **Ende**

Abbildung 4.7: Evolutionärer Algorithmus zur Optimierung der Parameter für das Suchverfahren

Individuen, die lediglich zu einer geringfügig kleineren Anzahl an expandierten Knoten als andere Individuen führen, eine deutlich bessere Fitness erhalten. Im Anschluss werden aus diesen zehn Individuen, zehnmal zwei Individuen ausgewählt und jeweils zu einem neuen Individuum (*Nachkommen*) rekombiniert. Die Auswahl der Individuen für die Reproduktion erfolgt mittels fitnessproportionaler Selektion (Weicker 2007). Dabei werden die Individuen zufällig ausgewählt, wobei die Wahrscheinlichkeit der Wahl eines Individuums seiner Fitness relativ zur Summe der Fitness aller Individuen entspricht. Die Rekombination zu einem neuen Individuum erfolgt mittels *Uniform-Crossover* (Weicker 2007; Umbarkar et al. 2015). Die somit erzeugten Nachkommen werden im Anschluss mutiert. Dabei kommt die *Gauß-Mutation* (Kramer 2009) zum Einsatz, bei der auf jedes Gen eine normalverteilte Zufallsvariable addiert wird. Für die folgenden Versuche wurde dafür eine Standardabweichung von $\sigma = 0{,}1$ gewählt. Nach der Rekombination wird jedem Nachkommen ebenfalls ein Fitnesswert zugewiesen. Der evolutionäre Algorithmus wurde so implementiert, dass für die nachfolgende Generation sowohl die Nachkommen als auch die Eltern zur Verfügung stehen (*Plusselektion*) (Kramer 2009). Daher werden aus allen Individuen (sowohl Eltern als auch Nachkommen) die zehn Individuen mit der geringsten Fitness eliminiert, so dass in der nächsten Generation ebenfalls wieder zehn Individuen zur Verfügung stehen. Der beschriebene evolutionäre Algorithmus wird, wie die beiden vorherigen Optimierungsverfahren, ebenfalls in Python umgesetzt.

4.6 Analyse auf Basis von Simulationsdaten

In diesem Abschnitt werden die zuvor eingeführten Einflussfaktoren für die Heuristikfunktion auf Basis von Simulationsdaten analysiert. Dabei kommt die in Abschnitt 3.4.1 eingeführte Simulationsumgebung zum Einsatz. Sofern nicht anders angegeben, werden in diesem Kapitel die folgenden zwei Datensätze verwendet:

- 3750 verschiedene simulierte Situationen für das in Abschnitt 3.4.1 vorgestellte Szenario der Getriebewellen. Diese Situationen entsprechen 50 vollständigen Entleerungen von Ladungsträgern, die initial mit 75 Werkstücken gefüllt sind.
- 4000 verschiedene simulierte Situationen für das in Abschnitt 3.4.1 vorgestellte Szenario der Ringschrauben. Diese Situationen entsprechen 100 vollständigen Entleerungen von Ladungsträgern, die initial mit 40 Werkstücken gefüllt sind.

Da ein Algorithmus zur Objektlageerkennung, insbesondere aufgrund von Verdeckungen, im Allgemeinen nicht alle Werkstücke erkennt, soll für die Untersuchungen in diesem Kapitel jeweils nur eine Teilmenge aller Werkstücke im Ladungsträger berücksichtigt werden. Dafür werden in den Untersuchungen lediglich die höchsten N_W Werkstücke berücksichtigt. Es wird davon ausgegangen, dass mit einem Wert von $N_W = 25$ bei den Getriebewellen nahezu alle Werkstücke enthalten sind, die von einem Algorithmus zur Objektlageerkennung erkannt werden können. Da die Grundfläche des Ladungsträgers bei den Ringschrauben im Verhältnis zur Werkstückgröße kleiner ist und die Werkstücke sich dadurch stärker verdecken, genügt es, in diesem Szenario $N_W = 20$ Werkstücke zu berücksichtigen.

4.6.1 Untersuchung der Bewertungsheuristiken

In Abschnitt 4.1 wurden verschiedene Bewertungsheuristiken zur Bewertung einzelner Knoten des Suchbaums eingeführt. Um diese Bewertungsheuristiken für die Bewertung anderer Einflussfaktoren und für das Trainieren neuronaler Netze zu verwenden, wird zunächst untersucht, welche der Bewertungsheuristiken für diese Zwecke am besten geeignet ist. Eine ideale Bewertungsheuristik sollte entscheiden können, welche Knoten jeweils im nächsten Schritt expandiert werden sollen, um mit möglichst wenigen expandierten bzw. evaluierten Knoten eine Greiflösung zu finden. Um zu bewerten, wie gut eine Bewertungsheuristik dafür geeignet ist, wird zunächst ein vollständiger Suchbaum generiert, indem alle Knoten expandiert werden, unabhängig davon, ob schon ein Zielknoten gefunden wurde. Auf diesem vollständig expandierten Suchbaum wird für alle Knoten die entsprechende Bewertungsheuristik berechnet. Im Anschluss wird die heuristische Suche verwendet, um einen der Zielknoten zu finden. Dabei verwendet die Heuristikfunktion hier jedoch

Tabelle 4.3: Vergleich der Bewertungsheuristiken für alle Knoten. Dargestellt ist jeweils die mittlere Anzahl expandierter und evaluierter Knoten unter Verwendung der Bewertungsheuristiken b_g, b_z, b_s und b_n. Für diese Untersuchung wurden die Situationen von 20 Kistenentleerungen der Getriebewellen und 25 Kistenentleerungen der Ringschrauben analysiert.

	b_g	b_z	b_s	b_n
Getriebewellen:				
Anzahl N_{exp} expandierter Knoten	14,8	14,7	14,4	15,8
Anzahl N_{eval} evaluierter Knoten	56,0	56,3	52,6	53,7
Ringschrauben:				
Anzahl N_{exp} expandierter Knoten	102,6	102,7	104,4	108,3
Anzahl N_{eval} evaluierter Knoten	145,1	145,3	142,7	146,3

nicht die in Abschnitt 4.5 beschriebenen Parameter, sondern beruht auf der entsprechenden Bewertungsheuristik. Somit kann untersucht werden, wie viele Expansionen und Evaluierungen von Knoten benötigt werden, um unter Verwendung der verschiedenen Bewertungsheuristiken eine geeignete Greiflösung zu finden. Da die Bewertungsheuristik vorwiegend dazu dienen soll, Knoten der selben Ebene des Baums gegeneinander zu gewichten, wird an dieser Stelle anstatt der Bestensuche das Hill Climbing (vgl. Abschnitt 2.2.4) verwendet. Hill Climbing ähnelt der Tiefensuche, wobei die Kinder eines Knotens nicht in einer beliebigen Reihenfolge, sondern anhand der Bewertungsheuristik expandiert werden. Alle Parameter des Versuches können Parametersatz 2.1 für Getriebewellen und Parametersatz 2.2 für Ringschrauben in Tabelle A.2 in Anhang A.2 entnommen werden. Tabelle 4.3 zeigt die Anzahl der Expansionen und Evaluierungen unter Verwendung der verschiedenen Bewertungsheuristiken für die beiden simulierten Szenarien der Getriebewellen und der Ringschrauben. Dabei ist in beiden Szenarien zu sehen, dass es für die Anzahl der expandierten und evaluierten Knoten kaum einen Unterschied macht, welche Bewertungsheuristik in der Heuristikfunktion verwendet wird. Alle Bewertungsheuristiken führen dazu, dass die Lösung nahezu auf direktem Weg gefunden wird.

Da eine reale Heuristik, im Gegensatz zur Bewertungsheuristik, keine Kenntnis über den gesamten Suchbaum hat, wird die Suche in der folgenden Untersuchung erschwert. Dabei wird die Bewertungsheuristik lediglich in den Werkstückknoten verwendet. Bei allen weiteren Knoten werden die Kinder in zufälliger Reihenfolge expandiert. Dadurch soll untersucht werden, wie gut die Bewertungsheuristik geeignet ist, die Suche zu steuern, wenn die nachfolgenden Knoten keine Kenntnis über den gesamten Suchbaum besitzen. Tabelle 4.4 zeigt die Ergebnisse dieser Untersuchung. Dabei ist zu sehen, dass die Anzahl der expandierten und evaluierten Knoten deutlich höher ist, wenn die Bewertungsheuristik nur für Werkstückknoten verwendet wird und andere Knoten in zufälliger Reihenfolge expandiert werden. Auch hier unterscheiden sich die

Tabelle 4.4: Vergleich der Bewertungsheuristiken für Werkstückknoten. Dargestellt ist jeweils die mittlere Anzahl expandierter und evaluierter Knoten unter Verwendung der Bewertungsheuristiken b_g, b_z, b_s und b_n. Für diese Untersuchung wurden die Situationen von 20 Kistenentleerungen der Getriebewellen und 25 Kistenentleerungen der Ringschrauben analysiert.

	b_g	b_z	b_s	b_n
Getriebewellen:				
Anzahl N_{exp} expandierter Knoten	90,6	89,7	89,8	106,0
Anzahl N_{eval} evaluierter Knoten	188,3	186,8	188,1	214,0
Ringschrauben:				
Anzahl N_{exp} expandierter Knoten	949,5	842,7	855,8	973,6
Anzahl N_{eval} evaluierter Knoten	1673,2	1490,9	1514,1	1729,5

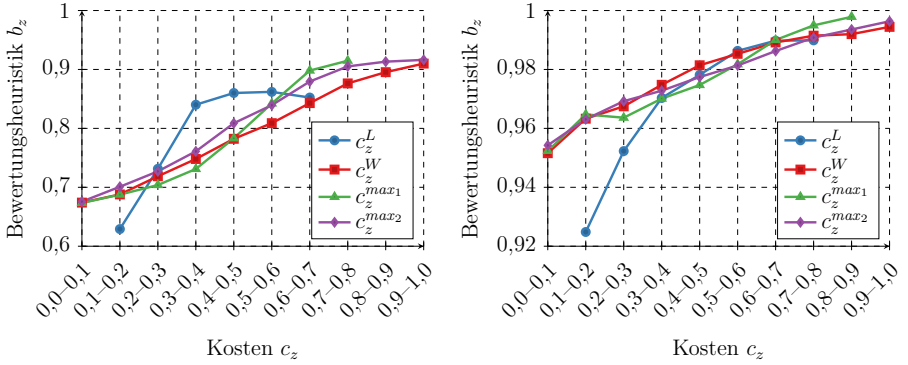
Ergebnisse der einzelnen Bewertungsheuristiken nur geringfügig, jedoch lassen sich, insbesondere bei den Ringschrauben, durch die Verwendung von b_z und b_s etwas bessere Ergebnisse erzielen. Da b_z in beiden Szenarien gut geeignet ist, um die Suche zu steuern und zudem einfach zu berechnen ist, wird diese Bewertungsheuristik im Folgenden verwendet, um die verschiedenen Einflussfaktoren der Heuristikfunktion zu bewerten. Insbesondere wird dieser Wert mittels

$$c_{nn} = b_z \tag{4.37}$$

in allen weiteren Versuchen für das Trainieren der in Abschnitt 4.4 eingeführten neuronalen Netze verwendet.

4.6.2 Bewertung des Einflusses der vertikalen Werkstückposition

Wie in Abschnitt 4.3.2 beschrieben, ist die vertikale Position eines Werkstücks ein naheliegender Indikator für die Wahrscheinlichkeit, einen geeigneten Griff zu finden. Daher wurden verschiedene Kostenwerte eingeführt, die von der vertikalen Position eines Werkstücks abhängen. In diesem Abschnitt wird untersucht, wie diese Kosten die Wahrscheinlichkeit, eine geeignete Lösung zu finden, beeinflussen. Dabei kommt die in Abschnitt 4.1 eingeführte Bewertungsheuristik b_z zum Einsatz. Die detaillierten Parameter für diesen Versuch können Parametersatz 3.1 und 3.2 in Tabelle A.3 in Anhang A.2 entnommen werden. Im Gegensatz zu einer realen Anwendung wird hier wiederum nicht nach dem Erreichen des ersten Zielknotens gestoppt, sondern alle Suchbäume werden vollständig expandiert. Da in den Suchbäumen von nahezu allen Situationen mehrere Werkstückinstanzen enthalten sind, umfasst diese Untersuchung die Eigenschaften von insgesamt 78 750 Knoten für das vorgestellte Szenario der Getriebewellen und 61 000 Knoten für das Szenario der Ringschrauben. Abbildung 4.8 zeigt, wie die Bewertungsheuristik b_z von den verschiedenen



(a) Getriebewellen (b) Ringschrauben

Abbildung 4.8: Bewertungsheuristik b_z in Abhängigkeit von der vertikalen Werkstückposition. Die Abbildung vergleicht die verschiedenen Kostenwerte c_z^L, c_z^W, $c_z^{max_1}$ und $c_z^{max_2}$ für das Szenario der Getriebewellen (a) und der Ringschrauben (b). Dargestellt ist jeweils der Mittelwert der Bewertungsheuristik b_z aller Knoten, deren Kostenwerte im selben Bereich liegen.

Kostenwerten für die vertikale Position des Werkstücks abhängt. Dabei unterscheiden sich die Verläufe der meisten Kostenwerte nur geringfügig. Lediglich der Kostenwert c_z^L, der auf der Position der Werkstücke relativ zum Ladungsträger basiert, weist einen abweichenden Verlauf auf. Ziel der Heuristikfunktion ist es, eine möglichst gute Abschätzung zu geben, welche Knoten expandiert werden sollten, um möglichst schnell einen Zielknoten zu erreichen. Nach demselben Kriterium wurden die Bewertungsheuristiken evaluiert und ausgewählt. Daher sollte eine möglichst starke Korrelation zwischen den Einflussfaktoren der Heuristikfunktion und der Bewertungsheuristik bestehen. Tabelle 4.5 zeigt deshalb den Korrelationskoeffizienten nach Pearson (1896) zwischen den verschiedenen Kostenwerten und der Bewertungsheuristik b_z. Dabei ist ebenfalls zu erkennen, dass der Kostenwert c_z^L nur eine schwache Korrelation aufweist. Dies lässt sich durch die in Abschnitt 4.3.2 genannten Nachteile dieses Kostenwerts erklären. Die

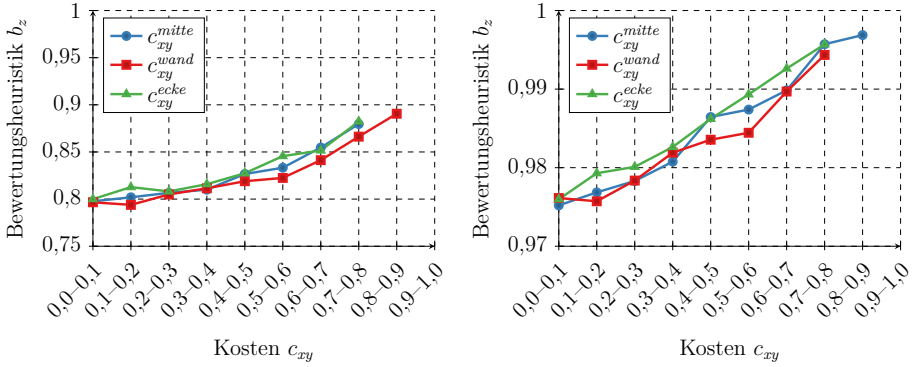
Tabelle 4.5: Vergleich verschiedener Kostenwerte für die vertikale Werkstückposition. Dargestellt sind die Korrelationskoeffizienten zwischen den verschiedenen, durch die Werkstückhöhe verursachten, Kosten und der Bewertungsheuristik b_z für die Szenarien der Getriebewellen und der Ringschrauben.

	c_z^L	c_z^W	$c_z^{max_1}$	$c_z^{max_2}$
Getriebewellen	0,320	0,517	0,500	0,512
Ringschrauben	0,356	0,478	0,508	0,506

Korrelationen für die anderen Kostenwerte sind jedoch deutlich stärker und liegen, insbesondere für das Szenario der Getriebewellen, in vergleichbaren Bereichen. Diese starke Korrelation zeigt, dass die auf der vertikalen Position des Werkstücks basierenden Kostenwerte c_z^W, $c_z^{max_1}$ und $c_z^{max_2}$ als Einflussfaktoren für die Heuristikfunktion geeignet sind. Die stärkste Korrelation liegt für das Szenario der Getriebewellen bei c_z^W und für das Szenario der Ringschrauben bei $c_z^{max_1}$ vor. Da $c_z^{max_1}$ im Mittel zu einer höheren Korrelation führt und das Szenario der Ringschrauben aufgrund der tieferen Kisten relevanter für die vertikale Position der Werkstücke ist, wird dieser Kostenwert für die weiteren Versuche verwendet.

4.6.3 Bewertung des Einflusses der horizontalen Werkstückposition

Neben Kostenwerten für die vertikale Position des Werkstücks wurden Kostenwerte eingeführt, die von der horizontalen Position eines Werkstücks innerhalb des Ladungsträgers abhängen (vgl. Abschnitt 4.3.3). In diesem Abschnitt werden daher auch die Auswirkungen der horizontalen Position der Werkstücke auf die Wahrscheinlichkeit untersucht, eine geeignete Lösung zu finden. Dabei kommt wiederum die in Abschnitt 4.1 eingeführte Bewertungsheuristik b_z zum Einsatz. Für die Untersuchung werden die selben 3750 bzw. 4000 simulierten Situationen und damit auch dieselbe Anzahl an Werkstückknoten analysiert, wie im vorherigen Abschnitt. Die detaillierten Parameter für diesen Versuch entsprechen ebenfalls Parametersatz 3.1 und 3.2 in Tabelle A.3 in Anhang A.2. Abbildung 4.9 zeigt, wie die Bewertungsheuristik b_z von den verschiedenen Kostenwerten für die horizontale Werkstückposition abhängt. Auch hier ist eine deutliche Korrelation zwischen der Bewertungsheuristik b_z und allen von der horizontalen Werkstückposition abhängigen Kosten zu erkennen. Beim Szenario der Getriebewellen ist dieser Zusammenhang, im Gegensatz zum Szenario der Ringschrauben, jedoch deutlich schwächer, je näher ein Werkstück an der Mitte des Ladungsträgers liegt. Dies kann durch den größeren Ladungsträger, bei dem ein relativ großer Bereich in der Mitte des Ladungsträgers kaum durch die Seitenwände eingeschränkt ist, erklärt werden. Tabelle 4.6 zeigt schließlich die Korrelationskoeffizienten zwischen den verschiedenen Kostenwerten und der eingeführten Bewertungsheuristik b_z. Bei dem Vergleich der einzelnen Kostenwerte fällt auf, dass c_{xy}^{wand} für den Abstand zur nächstgelegenen Seitenwand des Ladungsträgers in beiden Szenarien die höchste Korrelation besitzt und damit am besten für die Heuristikfunktion geeignet ist. Die zweithöchste Korrelation besitzt der Kostenwert c_{xy}^{mitte}, der sich aus dem Abstand zur Mitte des Ladungsträgers berechnet. Diese Korrelation zeigt, dass die auf der horizontalen Werkstückposition basierenden Kostenwerte als Einflussfaktoren für die Heuristikfunktion geeignet sind. Aufgrund der höchsten Korrelation wird im Folgenden der vom Abstand zur nächstgelegenen Seitenwand des Ladungsträgers abhängige Kostenwert c_{xy}^{wand} verwendet.



(a) Getriebewellen (b) Ringschrauben

Abbildung 4.9: Bewertungsheuristik in Abhängigkeit von der horizontalen Werkstückposition. Die Abbildung vergleicht die verschiedenen Kostenwerte c_{xy}^{mitte}, c_{xy}^{wand} und c_{xy}^{ecke} für das Szenario der Getriebewellen (a) und der Ringschrauben (b). Dargestellt ist jeweils der Mittelwert der Bewertungsheuristik b_z aller Knoten, deren Kostenwerte im selben Bereich liegen.

Tabelle 4.6: Vergleich verschiedener Kostenwerte für die horizontale Werkstückposition. Dargestellt sind die Korrelationskoeffizienten zwischen den verschiedenen, durch die horizontale Werkstückposition verursachten, Kosten und der Bewertungsheuristik b_z für die Szenarien der Getriebewellen und der Ringschrauben.

	c_{xy}^{mitte}	c_{xy}^{wand}	c_{xy}^{ecke}
Getriebewellen	0,121	0,148	0,101
Ringschrauben	0,206	0,224	0,190

4.6.4 Gewichtung zwischen vertikaler und horizontaler Position eines Werkstücks

In Abschnitt 4.5.2 wurde der Gewichtungsfaktor w_z eingeführt, der bestimmt, wie die aus der vertikalen Position eines Werkstücks resultierenden Kosten c_z gegenüber den aus der horizontalen Position des Werkstücks resultierenden Kosten c_{xy} gewichtet werden. Die Auswirkungen dieses Gewichtungsfaktors werden in diesem Abschnitt analysiert. Dazu wurden wiederum die beschriebenen Situationen für Getriebewellen und Ringschrauben untersucht und jeweils die Korrelation zwischen der Bewertungsheuristik b_z und $c_z \cdot w_z + c_{xy} \cdot (1 - w_z)$ für verschiedene Werte des Gewichtungsfaktors w_z berechnet. Abbildung 4.10 zeigt die Ergebnisse für die vorgestellten Szenarien der Getriebewellen und der Ringschrauben. Dabei ist zu sehen, dass das Optimum für

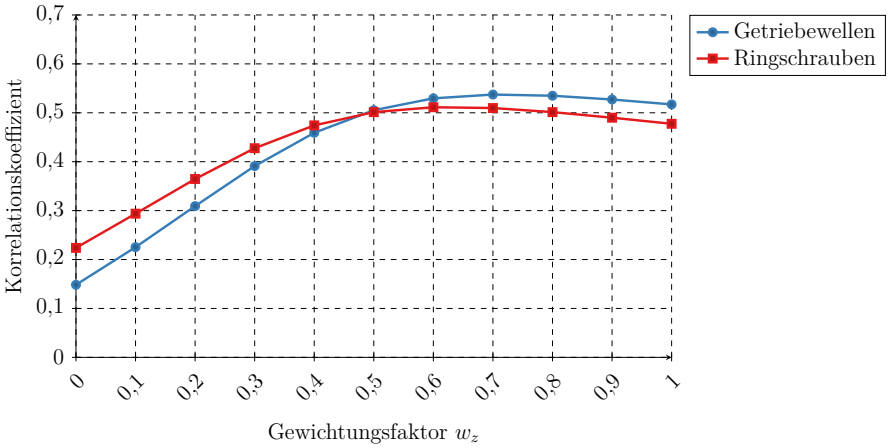


Abbildung 4.10: Korrelation zwischen der Bewertungsheuristik b_z und $c_z \cdot w_z + c_{xy} \cdot (1 - w_z)$ in Abhängigkeit von verschiedenen Werten für den Gewichtungsfaktor w_z zwischen den Kosten der vertikalen und horizontalen Position eines Werkstücks für die Szenarien der Getriebewellen und der Ringschrauben

die Getriebewellen bei 0,7 und für die Ringschrauben bei 0,6 liegt. In beiden Szenarien nimmt die Korrelation deutlich ab, wenn die vertikale Position des Werkstücks nicht berücksichtigt wird ($w_z = 0$). Dies zeigt, dass die vertikale Position des Werkstücks einen wichtigen Einfluss auf die Erfolgswahrscheinlichkeit hat. Auch für $w_z = 1$, was bedeutet, dass die horizontale Werkstückposition nicht berücksichtigt wird, nimmt die Korrelation leicht ab. Für eine möglichst zielgerichtete Suche sollten somit die Kosten der vertikalen und horizontalen Position eines Werkstücks in die Heuristikfunktion einfließen.

4.6.5 Bewertung des Einflusses der TCP-Orientierung

In den Abschnitten 3.2.3 und 4.3.4 wurden Ansätze vorgeschlagen, um die Orientierung von Komponenten in der heuristischen Suche zu berücksichtigen. In diesem Abschnitt wird untersucht, inwieweit die Orientierung von TCPs die Wahrscheinlichkeit beeinflusst, eine geeignete Greiflösung zu finden. Dafür werden, wie in den vorherigen Abschnitten, die simulierten Daten aus 3750 bzw. 4000 verschiedenen Situationen analysiert. Die detaillierten Parameter für diesen Versuch entsprechen weiterhin Parametersatz 3.1 bzw. 3.2 in Tabelle A.3 in Anhang A.2. Da jeder Werkstückknoten aufgrund mehrerer möglicher Greifposen am Werkstück und aufgrund der Freiheitsgrade mehrere TCP-Knoten als Nachfolger besitzt, ist die Anzahl der TCP-Knoten

in den Suchbäumen deutlich höher als die der Werkstückknoten. In dieser Untersuchung werden daher die Eigenschaften von 5 355 000 TCP-Knoten für das Szenario der Getriebewellen und 56 778 037 TCP-Knoten für das Szenario der Ringschrauben analysiert.

Zunächst werden die Abweichungen zwischen den Orientierungen der TCPs und der Orientierung des Ladungsträgers betrachtet. Es sei $\boldsymbol{z}_L$ die z-Achse des Ladungsträgerkoordinatensystems $\mathcal{K}_L$ und $\boldsymbol{z}_T$ die z-Achse des TCP-Koordinatensystems $\mathcal{K}_T$. Sei weiterhin, wie in Abschnitt 3.2.3,

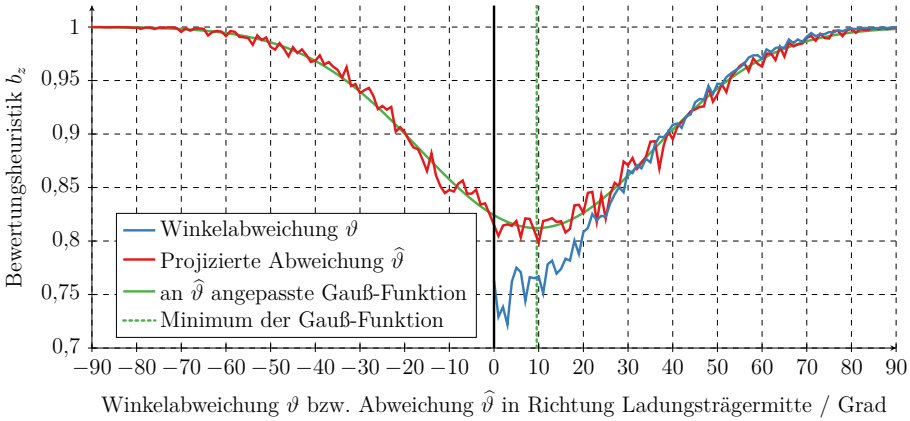
$$\boldsymbol{n}_T = -\boldsymbol{z}_T \tag{4.38}$$

ein Vektor, der in die entgegengesetzte Richtung der z-Achse von $\mathcal{K}_T$ zeigt, so lässt sich der Winkel der Abweichung von der z-Achse des Ladungsträgerkoordinatensystems $\mathcal{K}_L$ mittels

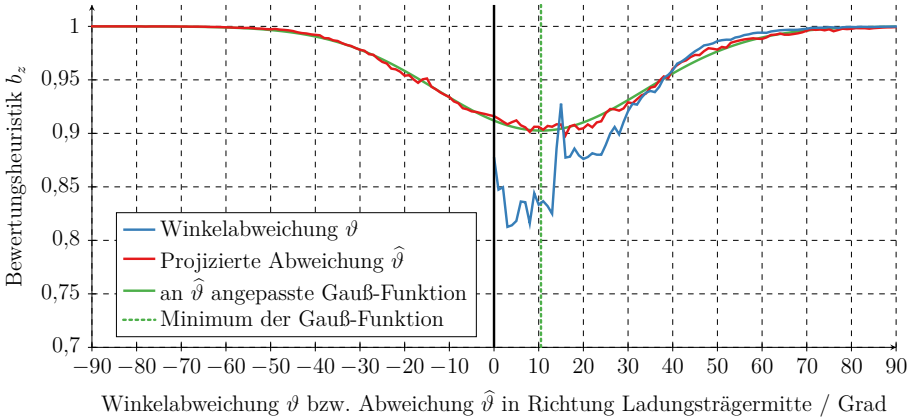
$$\vartheta = \arccos\left(\frac{\langle \boldsymbol{z}_L, \boldsymbol{n}_T \rangle}{\|\boldsymbol{z}_L\| \cdot \|\boldsymbol{n}_T\|}\right) \tag{4.39}$$

berechnen. Die blaue Kurve in Abbildung 4.11 zeigt, inwiefern die Bewertungsheuristik b_z von diesem Winkel abhängt (die weiteren Elemente der Abbildung können zunächst vernachlässigt werden). Dabei fällt auf, dass ein Griff, bei dem die z-Achse des TCP-Koordinatensystems $\mathcal{K}_T$ relativ zum Koordinatensystem $\mathcal{K}_L$ des Ladungsträgers weitgehend nach unten (in die negative Richtung der z-Achse) zeigt, die höchste Erfolgswahrscheinlichkeit besitzt. Wie in Abbildung 4.11 zu sehen ist, nimmt diese Erfolgswahrscheinlichkeit mit steigender Abweichung von dieser Orientierung stark ab. Daraus lässt sich ableiten, dass diese Abweichung der Orientierung geeignet ist, zur Beschleunigung der Suche in der Heuristikfunktion berücksichtigt zu werden.

Die vorige Untersuchung berücksichtigt lediglich den Betrag der Winkelabweichung. In welche Richtung die Abweichung jeweils vorliegt, wird hingegen ignoriert. In Abschnitt 3.2.3 wurde angenommen, dass es vorteilhaft ist, wenn der Greifer bzw. der Vektor $\boldsymbol{n}_T$ leicht in Richtung der Mitte des Ladungsträgers geneigt ist. Diese Annahme soll in der folgenden Untersuchung überprüft werden. Dazu wird im Gegensatz zur vorigen Untersuchung unterschieden, ob eine Abweichung in Richtung der Mitte des Ladungsträgers oder in die entgegengesetzte Richtung vorliegt. Um dies zu bestimmen, wird eine Ebene A definiert, die die z-Achse des Ladungsträgerkoordinatensystems $\mathcal{K}_L$ und den Ursprung des TCP-Koordinatensystems $\mathcal{K}_T$ enthält. Abbildung 4.12 zeigt die Berechnung grafisch an einem Beispiel. Der bereits in der vorigen Untersuchung definierte Vektor $\boldsymbol{n}_T$ wird nun auf A projiziert, um den Vektor $\hat{\boldsymbol{n}}_T$ zu erhalten. Dieser Vektor kann verwendet werden, um eine vergleichbare Analyse wie zuvor durchzuführen. Allerdings ist es nun möglich, zwischen Vektoren, die in Richtung der Mitte des Ladungsträgers



(a) Getriebewellen



(b) Ringschrauben

Abbildung 4.11: Bewertungsheuristik in Abhängigkeit von der Orientierung des TCPs. Dargestellt ist der Mittelwert von b_z über alle untersuchten Situationen für die Szenarien der Getriebewellen (a) und der Ringschrauben (b).

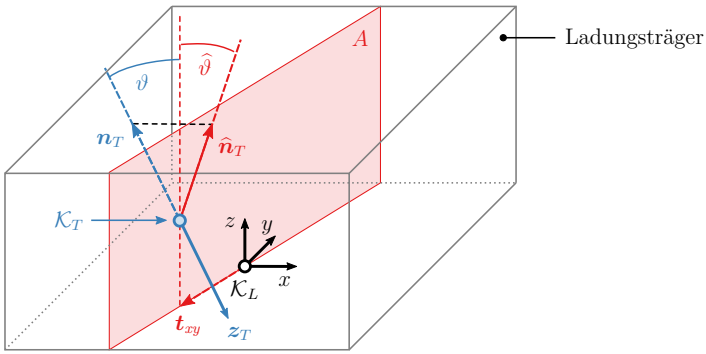


Abbildung 4.12: Berechnung der Neigung von $\hat{\boldsymbol{n}}_T$ in Richtung der Mitte des Ladungsträgers

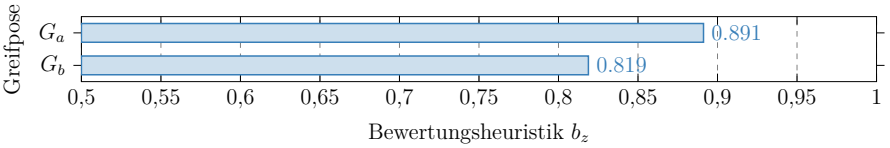
geneigt sind, und Vektoren, die in die entgegengesetzte Richtung geneigt sind, zu unterscheiden. Es sei daher der Winkel

$$\hat{\vartheta} = \arccos\left(\frac{\langle \boldsymbol{z}_L, \hat{\boldsymbol{n}}_T \rangle}{\|\boldsymbol{z}_L\| \cdot \|\hat{\boldsymbol{n}}_T\|}\right) \cdot \operatorname{sgn}(\langle \hat{\boldsymbol{n}}_T, \boldsymbol{t}_{xy} \rangle) \tag{4.40}$$

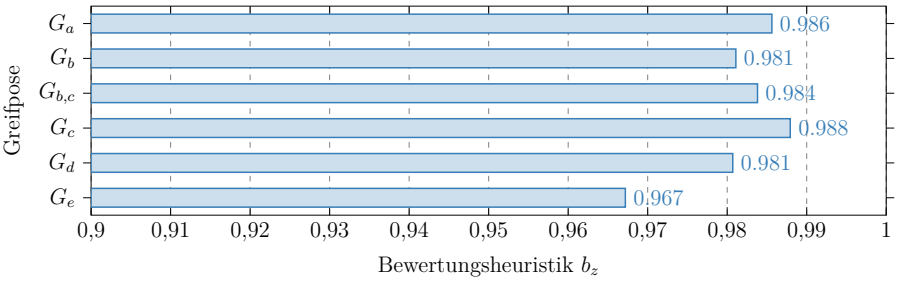
definiert, wobei sich das Vorzeichen aus dem Skalarprodukt von $\hat{\boldsymbol{n}}_T$ und der horizontalen Komponente $\boldsymbol{t}_{xy}$ des Vektors vom Ladungsträgerkoordinatensystem $\mathcal{K}_L$ zum TCP-Koordinatensystem $\mathcal{K}_T$ berechnet. Der Zusammenhang zwischen diesem Winkel $\hat{\vartheta}$ und der Bewertungsheuristik b_z ist ebenfalls in Abbildung 4.11 dargestellt. Dabei ist zu sehen, dass die optimale Orientierung nicht senkrecht ist, sondern, wie in Abschnitt 3.2.3 angenommen, um einige Grad in Richtung der Mitte des Ladungsträgers davon abweicht. Eine mittels nichtlinearer Regression an diese Daten angepasste Gauß-Funktion ist ebenfalls in Abbildung 4.11 dargestellt. Das Minimum dieser Gauß-Funktion liegt für das Szenario der Getriebewellen bei ca. 9,56° und für das Szenario der Ringschrauben bei ca. 10,51°. Dies bedeutet, dass am schnellsten eine Lösung gefunden wird, wenn das Greiferelement, an dem der jeweilige TCP definiert ist, um diesen Winkel in Richtung der Mitte des Ladungsträgers geneigt ist.

4.6.6 Analyse der unterschiedlichen Greifposen

Abbildung 4.13 zeigt die mittlere Bewertungsheuristik b_z für die einzelnen Greifposen der Werkstücke. Dabei ist zu sehen, dass bei den Getriebewellen die Greifpose G_b, die am Bund des Werkstücks definiert ist, einen geringeren Wert für die Bewertungsheuristik besitzt und



(a) Getriebewellen



(b) Ringschrauben

Abbildung 4.13: Analyse der unterschiedlichen Greifposen für die Szenarien der Getriebewellen (a) und der Ringschrauben (b). Die Indizes der Greifposen entsprechen den Buchstaben der Abbildungen 3.17 und 3.18, wobei Greifpose $G_{b,c}$ der Greifpose G_b mit einem zusätzlichen Freiheitsgrad entspricht, sodass die Greifpose G_c darin ebenfalls beinhaltet ist.

somit erfolgversprechender ist als die Greifpose G_a, bei der der Greifer die Getriebewelle am Schaft umschließt. Dies erklärt sich durch die bessere Zugänglichkeit eines Griffs am Bund des Werkstücks im Gegensatz zu einem Griff am Schaft. Bei den Ringschrauben ist der Innengriff an der Greifpose G_e am erfolgversprechendsten, was insbesondere daran liegt, dass der Bereich in der Mitte des Rings meistens nicht durch andere Werkstücke blockiert wird. Die Zugänglichkeit von Griffen, bei denen ein Greiferfinger neben dem Werkstück positioniert werden muss, ist hingegen öfters durch danebenliegende Werkstücke eingeschränkt.

In Abschnitt 4.3.6 wurden statische Prioritäten eingeführt, um bestimmte Greifposen gegenüber anderen Greifposen zu bevorzugen. Sofern keine anderen Gründe existieren, bestimmte Greifposen zu bevorzugen, können die hier vorgestellten Ergebnisse verwendet werden, um die Suche zu beschleunigen. Dies lässt sich erreichen, indem vielversprechendere Greifposen niedrigere Kosten für die Priorität erhalten und dadurch in der heuristischen Suche bevorzugt expandiert werden. Im Folgenden werden die Prioritäten dieser Greifposen daher so festgelegt, dass ein linearer Zusammenhang zwischen c_{prio} und der Bewertungsheuristik b_z besteht. Dieser Zusammenhang

ist so gewählt, dass die erfolgversprechendste Greifpose, die den niedrigsten Wert für b_z besitzt, die Kosten $c_{prio} = 0$ erhält und die Greifpose mit der höchsten Bewertungsheuristik b_z die Kosten $c_{prio} = 1$.

4.6.7 Implementierung und Untersuchung der künstlichen neuronalen Netze

In diesem Abschnitt wird zunächst die Implementierung und Architektur der neuronalen Netze beschrieben und auf das Training der Netze eingegangen. Im Anschluss wird die Verwendung der neuronalen Netze in der Heuristikfunktion untersucht und insbesondere verschiedene Varianten für die Repräsentation von Rotationen miteinander verglichen.

4.6.7.1 Implementierung der neuronalen Netze

Für die Realisierung der neuronalen Netze kommt bei den folgenden Versuchen die frei verfügbare Bibliothek *FANN (Fast Artificial Neural Network)* (Nissen et al. 2003; Nissen 2007) zum Einsatz. Die Vorteile dieser Bibliothek liegen in einer schnellen Ausführungsgeschwindigkeit sowie einer einfachen Anwendbarkeit.

Alle in dieser Arbeit verwendeten neuronalen Netze besitzen zwei versteckte Schichten, wobei die erste aus 50 Neuronen und die zweite aus 30 Neuronen besteht. Als Aktivierungsfunktion kommt die Sigmoid-Funktion (Mitchell 1997) zum Einsatz. Die beschriebene Architektur des neuronalen Netzes hat sich in einigen vergleichenden Versuchen als geeignet herausgestellt und führt zu geringen Fehlern der Netze (mittlere quadratische Abweichung). Alternative Netzarchitekturen oder Parameter sollen in dieser Arbeit nicht untersucht werden. Das Trainieren der neuronalen Netze erfolgt mit dem in FANN implementierten Algorithmus *iRPROP* (Igel et al. 2000), der auf dem adaptiven Back Propagation Verfahren *RPROP* (Riedmiller et al. 1993) basiert. Dabei wird so lange trainiert, bis entweder der Fehler E des neuronalen Netzes kleiner oder gleich 0,001 ist oder die maximale Epochenanzahl von 500 erreicht wird.

4.6.7.2 Vergleich zwischen verschiedenen Repräsentationen von Rotationen

Die Rotationen der einzelnen Komponenten lassen sich, wie in Abschnitt 4.4.2 beschrieben, durch verschiedene Repräsentationen darstellen. Im folgenden Versuch wird der Einfluss der gewählten Repräsentation auf die Effektivität der neuronalen Netze untersucht. Die neuronalen Netze werden auf der Basis von 1500 Situationen für die Getriebewellen und 1000 Situationen

Tabelle 4.7: Vergleich verschiedener Repräsentationen von Rotationen für neuronale Netze. Dargestellt ist jeweils die mittlere Anzahl expandierter und evaluierter Knoten unter Verwendung der jeweiligen Repräsentation von Rotationen. Für diese Untersuchung wurden die Situationen von 20 Kistenentleerungen der Getriebewellen und 25 Kistenentleerungen der Ringschrauben analysiert.

	Feste Winkel	Quaternionen	6D
Getriebewellen:			
Anzahl N_{exp} expandierter Knoten	90,0	89,1	119,0
Anzahl N_{eval} evaluierter Knoten	45,1	44,2	52,2
Ringschrauben:			
Anzahl N_{exp} expandierter Knoten	1580,3	1703,4	2470,6
Anzahl N_{eval} evaluierter Knoten	914,7	997,9	1433,4

für die Ringschrauben trainiert. Die Evaluation der trainierten Netze erfolgt auf der gleichen Anzahl von Situationen, wobei die Trainingssituationen darin nicht enthalten sind. Die manuell definierten spezifischen Einflussfaktoren für die Heuristikfunktion werden für diesen Versuch mittels

$$w_{nn}^{W} = w_{nn}^{G} = w_{nn}^{T} = w_{nn}^{V} = 1 \tag{4.41}$$

nicht berücksichtigt, um das Ergebnis dieser Untersuchung nicht zu beeinflussen. Weiterhin wird der Einfluss der Kosten c_n für die Nachfolger eines Knoten zunächst mittels

$$w_W = w_G = w_T = w_V = 0{,}5 \tag{4.42}$$

für alle Komponenten auf einen festen Wert gesetzt. Die detaillierten Parameter des Versuchs lassen sich Parametersatz 4.1 und 4.2 in Tabelle A.4 in Anhang A.2 entnehmen. Tabelle 4.7 zeigt die Ergebnisse für die verschiedenen Repräsentationen von Rotationen. Dabei ist zu sehen, dass bei den Getriebewellen die Verwendung der festen Winkel und die Verwendung der Quaternionen zu ähnlichen Ergebnissen führen. Werden die beiden ersten Spalten der Rotationsmatrix (6D) verwendet, so müssen im Mittel mehr Knoten evaluiert und expandiert werden, um eine Lösung zu finden. Bei den Ringschrauben schneiden die Quaternionen hingegen etwas schlechter ab als die festen Winkel. Daher wird für die neuronalen Netze im Folgenden die Repräsentation durch die Winkel ψ, θ, ϕ verwendet, die die Drehungen um die feststehenden x-, y- und z-Achsen beschreiben.

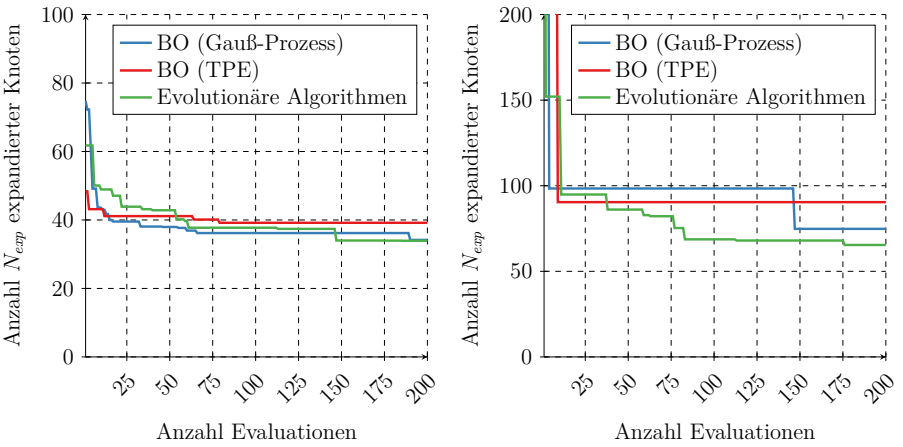
4.6.8 Untersuchung der gesamten Heuristikfunktion

Um die gesamte Heuristikfunktion zu optimieren, wurden in Abschnitt 4.5.3 folgende Optimierungsverfahren zur Bestimmung geeigneter Gewichtungsfaktoren vorgestellt:

- Bayessche Optimierung (BO) mit Gauß-Prozess
- Bayessche Optimierung (BO) mit Tree-structured Parzen Estimator (TPE)
- Evolutionäre Algorithmen

Die verschiedenen Verfahren werden in diesem Abschnitt untersucht und miteinander verglichen. In den bisherigen Versuchen wurden sowohl die Anzahl N_{exp} expandierter Knoten als auch die Anzahl N_{eval} evaluierter Knoten betrachtet. Zur Anwendung der Optimierungsverfahren muss jedoch ein einzelner Wert gewählt werden, der durch die Verfahren optimiert wird. Da als Zielsetzung dieser Arbeit eine im Mittel möglichst geringe Rechenzeit festgelegt wurde, sollte der gewählte Wert möglichst stark mit der Rechenzeit des entwickelten Verfahrens korrelieren. Bei der Berechnung der Ergebnisse für feste Winkel in Tabelle 4.7 wurde für die Anzahl N_{exp} expandierter Knoten eine Korrelation von 0,69 mit der Rechenzeit und für die Anzahl N_{eval} evaluierter Knoten eine Korrelation von 0,66 festgestellt. Für die Ringschrauben lag die Korrelation sowohl für N_{exp} als auch für N_{eval} bei 0,97. Aufgrund der geringfügig stärkeren Korrelation wird die Anzahl N_{exp} der expandierten Knoten als zu optimierender Wert verwendet.

In dieser Untersuchung werden jeweils 10 Ladungsträger vollständig entleert, was bei den Getriebewellen $N_S = 750$ Situationen und bei den Ringschrauben $N_S = 400$ Situationen entspricht. Zur Evaluation von gegebenen Parametern wird die mittlere Anzahl N_{exp} der expandierten Knoten über diese Situationen bestimmt. Die Gewichtungsfaktoren werden bei allen Verfahren so lange optimiert, bis 200 Evaluationen durchgeführt wurden. Die Populationsgröße bei den evolutionären Algorithmen beträgt 10. Somit werden bei diesem Optimierungsverfahren 20 Generationen betrachtet. Um in der Fitnessfunktion der evolutionären Algorithmen bessere Individuen ausreichend zu bevorzugen, aber geringfügige Unterschiede dennoch nicht überzubewerten, wurde für den in Abschnitt 4.5.3 eingeführten Faktor r_{exp} ein Wert von 2,5 gewählt. Dieser Wert führte auf einigen exemplarischen Testdaten zu sinnvoll erscheinenden Fitnesswerten, wird jedoch im Rahmen dieser Arbeit nicht näher untersucht. Die detaillierten Parameter des Versuchs lassen sich Parametersatz 5.1 und 5.2 in Tabelle A.5 in Anhang A.2 entnehmen. Abbildung 4.14 zeigt, wie sich das Ergebnis während des Optimierungsvorgangs bei den einzelnen Verfahren verbessert. Dabei ist zu sehen, dass sich die Ergebnisse der einzelnen Verfahren nur geringfügig unterscheiden und alle untersuchte Verfahren nach wenigen Evaluationen geeignete Gewichtungsfaktoren finden. Die evolutionären Algorithmen führen bei beiden Szenarien zur



(a) Getriebewellen (b) Ringschrauben

Abbildung 4.14: Ergebnisse der verschiedenen Optimierungsverfahren. Dargestellt ist der minimal erreichte Wert für die Anzahl N_{exp} der notwendigen expandierten Knoten in Abhängigkeit von der Anzahl an Evaluationen für verschiedene Optimierungsverfahren.

geringsten Anzahl an expandierten Knoten, weshalb diese für alle weiteren Optimierungen verwendet werden.

Im Folgenden wird schließlich untersucht, inwieweit die Verwendung der neuronalen Netze zu einer schnellen Berechnung beitragen. Dafür werden drei Varianten verglichen:

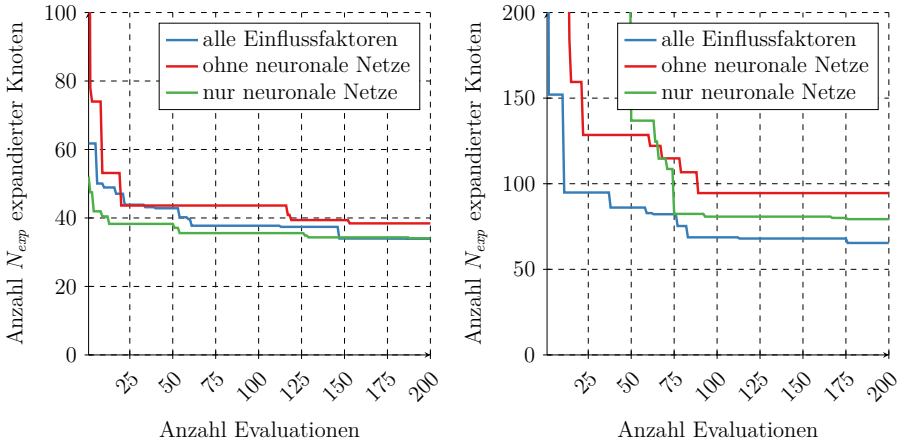
- Verwendung aller Einflussfaktoren und Optimierung der entsprechenden Gewichtungsfaktoren
- Verzicht auf die Ausgabewerte der neuronalen Netze mittels der konstanten Gewichtungsfaktoren

$$w_{nn}^{W} = w_{nn}^{G} = w_{nn}^{T} = w_{nn}^{V} = 0 \tag{4.43}$$

- Verwendung der neuronalen Netze und Verzicht auf andere spezifische Einflussfaktoren mittels der konstanten Gewichtungsfaktoren

$$w_{nn}^{W} = w_{nn}^{G} = w_{nn}^{T} = w_{nn}^{V} = 1 \tag{4.44}$$

Abbildung 4.15 zeigt die Ergebnisse dieser Varianten. Hier ist zu sehen, dass die Berücksichtigung



(a) Getriebewellen (b) Ringschrauben

Abbildung 4.15: Vergleich der Optimierung mit und ohne Verwendung der neuronalen Netze. Dargestellt ist der minimal erreichte Wert für die Anzahl N_{exp} der notwendigen expandierten Knoten in Abhängigkeit von der Anzahl an Evaluationen des Optimierungsverfahrens mit unterschiedlichem Einfluss der neuronalen Netze.

der neuronalen Netze in der Heuristikfunktion die Anzahl der benötigten expandierten Knoten reduziert. Bei dem Szenario der Getriebewellen lässt sich dadurch die mittlere Anzahl expandierter Knoten von 38,4 auf 33,9 und bei dem Szenario der Ringschrauben von 94,5 auf 65,4 reduzieren. Werden nur die Ausgabewerte der neuronalen Netze anstatt der manuell definierten Einflussfaktoren verwendet, so sinkt die Anzahl der expandierten Knoten bei dem Szenario der Getriebewellen vergleichbar auf 34,1. Hier könnte somit weitgehend auf die manuellen Einflussfaktoren verzichtet werden. Bei dem Szenario der Ringschrauben sinkt die Anzahl lediglich auf 79,3, wenn die manuell definierten Einflussfaktoren nicht berücksichtigt werden. Obwohl es Unterschiede zwischen den beiden Szenarien gibt, zeigt dies, dass die neuronalen Netze geeignet sind, die manuellen Einflussfaktoren weitgehend zu ersetzen und die Suche dadurch schneller eine Lösung findet. Das schnellste Ergebnis wird in beiden Szenarien jedoch durch eine Kombination aus manuell definierten Einflussfaktoren und neuronalen Netzen erreicht.

4.6.8.1 Rechenzeit

Abschließend soll an dieser Stelle die absolute Rechenzeit der Greifplanung auf einer konkreten Hardware betrachtet werden. Die Versuche wurden auf einem Intel Core i7-4900MQ mit 2,8

GHz und 16GB DDR3 Arbeitsspeicher durchgeführt, was eine realistische Leistungsklasse für einen Industrie-PC darstellt. Die gesamte Greifplanung wurde in C++ implementiert. Dabei lag die mittlere Rechenzeit für die gesamte Greifplanung einer Situation mit den optimalen Parametern mit bayesscher Optimierung bei 37,54 ms für das Szenario der Getriebewellen und bei 52,37 ms für das Szenario der Ringschrauben. Dies liegt deutlich unter dem in Abschnitt 1.3 festgelegten Ziel von 1000 ms. Damit wird das definierte Ziel der geringen mittleren Taktzeit durch den in dieser Arbeit entwickelten Ansatz erfüllt.

4.7 Zusammenfassung

In diesem Kapitel wurde eine geeignete Heuristikfunktion für das heuristische Suchverfahren aus Kapitel 3 vorgestellt. Dafür wurden zunächst Bewertungsheuristiken eingeführt, um einzelne Knoten im Suchbaum auf Basis ihrer nachfolgenden Knoten zu bewerten. Es wurde untersucht, wie gut diese geeignet sind, um zu bewerten, welche Knoten expandiert werden müssen, um möglichst schnell eine Lösung zu finden. Im Anschluss wurden verschiedene Einflussfaktoren wie die Position von Werkstücken oder die Orientierung von Tool Center Points (TCP) untersucht. Dabei konnte anhand von Simulationsdaten gezeigt werden, dass die Berücksichtigung der vertikalen Position von Werkstücken, des Abstands der Werkstücke zur Seitenwand des Ladungsträgers und der Orientierung von TCPs die Suche beschleunigen kann. Die Ergebnisse dieser Untersuchungen sind vergleichbar mit den Ergebnissen der in Spenrath et al. (2017b) durchgeführten Versuche auf Daten realer Roboteranlagen. Um die Einflussfaktoren nicht manuell festlegen zu müssen, wurde weiterhin gezeigt, dass es möglich ist, die relevanten Einflussfaktoren durch künstliche neuronale Netze maschinell zu erlernen, wobei die Netze mit den Daten der jeweiligen Anwendung trainiert wurden. Die zusätzliche Verwendung dieser neuronalen Netze konnte die Suche gegenüber der reinen Verwendung manuell festgelegter Einflussfaktoren zudem beschleunigen. Weiterhin wurde eine Heuristikfunktion entwickelt, mit der alle Einflussfaktoren in einen einzelnen Kostenwert vereint werden können. Damit lassen sich die verschiedenen Einflussfaktoren für ein heuristisches Suchverfahren verwenden. Um geeignete Gewichtungsfaktoren für diese Heuristikfunktion zu finden, wurden verschiedene Optimierungsverfahren angewandt und die Ergebnisse miteinander verglichen. Schließlich wurde gezeigt, dass die erreichbare mittlere Rechenzeit in den beiden untersuchten Szenarien bei 37,54 ms bzw. 52,37 ms liegt und damit das definierte Ziel einer geringen mittleren Taktzeit erfüllt wird.

5 Experimentelle Untersuchungen

In diesem Kapitel wird die in dieser Arbeit vorgestellte Greifplanung experimentell validiert. Dazu werden zunächst zwei verschiedene Versuchsaufbauten beschrieben, in denen die Greifplanung zum Einsatz kommt, sowie die Vorgehensweise bei der Versuchsdurchführung erläutert. Im Anschluss werden die Ergebnisse der mit diesen Versuchsaufbauten durchgeführten Versuche präsentiert. Da das in dieser Arbeit entwickelte Verfahren bereits mehrfach im industriellen Einsatz verwendet wird, werden in Abschnitt 5.2 schließlich einige Ergebnisse aus einer dieser Realisierungen ausgewertet. Damit kann die Validierung durch den Beweis der Industrietauglichkeit und durch statistische Aussagen von über 1,5 Mio. Griffen unterstützt werden.

5.1 Versuche an experimentellen Versuchsaufbauten

Im bisherigen Verlauf der Arbeit wurde die Greifplanung anhand von simulierten Daten untersucht und validiert. Dies hatte den Vorteil, dass die Untersuchungen unabhängig von einem konkreten Verfahren zur Objektlageerkennung erfolgen konnten. Um zu zeigen, dass der Ansatz auch praktisch eingesetzt werden kann, muss das Verfahren jedoch in realen Anwendungen experimentell evaluiert werden. In diesem Abschnitt soll gezeigt werden, dass das Verfahren auch in realen Anwendungen geringe Rechenzeiten erreicht und dass die neuronalen Netze und die Optimierung der Parameter auch dort angewendet werden können. Dazu kommen zwei Versuchsaufbauten mit unterschiedlichen Robotern, Greifern und Werkstücken zum Einsatz. Die Versuchsaufbauten entsprechen den im bisherigen Verlauf der Arbeit verwendeten Szenarien der Getriebewellen und der Ringschrauben und werden in diesem Abschnitt im Detail beschrieben.

5.1.1 Versuchsaufbau für Getriebewellen

Der erste Versuchsaufbau (Abbildung 5.1) besteht aus einem Comau NJ-130-2.7 Knickarm-Roboter und einem pneumatischen Klemmgreifer. Der Versuchsaufbau verfügt über zwei Stellplätze für Ladungsträger. Die zu vereinzelnden Werkstücke sind die bereits in der Simulationsumgebung in Abschnitt 3.4.1 vorgestellten Getriebewellen. Diese werden, wie in der Simulation,

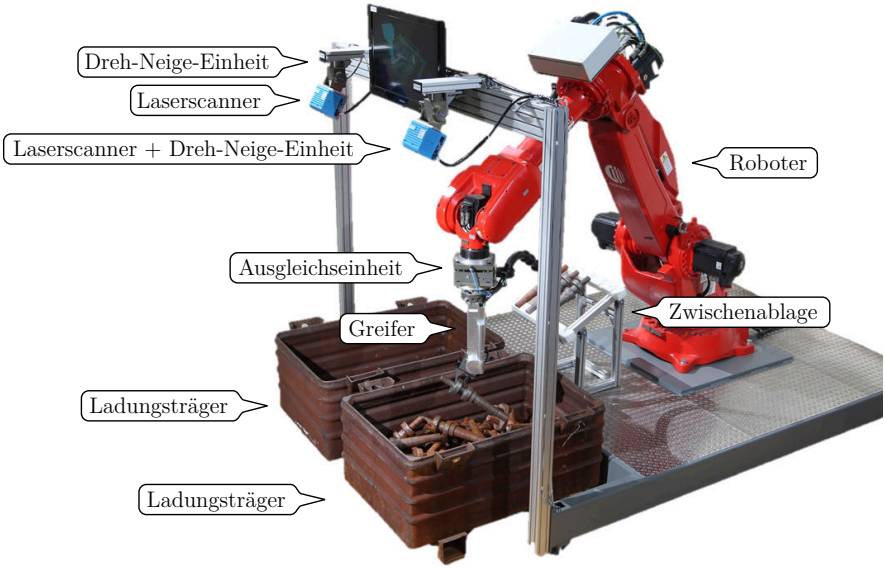


Abbildung 5.1: Versuchsaufbau zur Vereinzelung von Getriebewellen mit einem Comau NJ-130-2.7 Roboter (Quelle des Originalfotos: Fraunhofer IPA)

aus Kisten mit den Abmessungen 1200 mm × 800 mm × 600 mm entnommen. Als Sensor kommt über jedem Stellplatz ein Sick LMS-400 Laserscanner zum Einsatz, der an einer Dreh-Neige-Einheit vom Typ Schunk PW 70 montiert ist und sich ca. 2333 mm über dem jeweiligen Stellplatz befindet. Die mittlere Punktdichte und das Sensorrauschen ($\sigma_R = 9\,\text{mm}$) entsprechen dabei ungefähr den Werten aus der Simulation. Als Greifer kommt ein pneumatischer Schunk PGF 100 Zweifinger-Klemmgreifer zum Einsatz, der an einer Drehachse montiert ist, die sich von $-90°$ bis 90° drehen kann. Zusätzlich ist zwischen dem Greifer und dem Roboterflansch eine Ausgleichseinheit vom Typ Schunk AGE-S-Z-160-0 montiert. Diese ist in der Lage, eine Bewegung von 14 mm entlang der z-Achse des Koordinatensystems $\mathcal{K}_F$ des Roboterflanschs auszugleichen und zu detektieren. Dies ermöglicht es dem Roboter, eine Kollision zu erkennen, die Bewegung abzubrechen und sich in einem solchen Fall wieder in die Ausgangspose zu bewegen, so dass ein erneuter Entnahmevorgang gestartet werden kann, ohne dass ein manueller Eingriff erfolgen muss.

Der Ablauf zur Handhabung der Werkstücke sieht folgendermaßen aus. Zunächst werden Werkstücke nacheinander aus dem linken Ladungsträger entnommen und auf einer Zwischenablage abgelegt. Nachdem fünf Werkstücke auf der Zwischenablage abgelegt worden sind oder kein weiteres Werkstück aus dem linken Ladungsträger entnommen werden kann, greift der Roboter

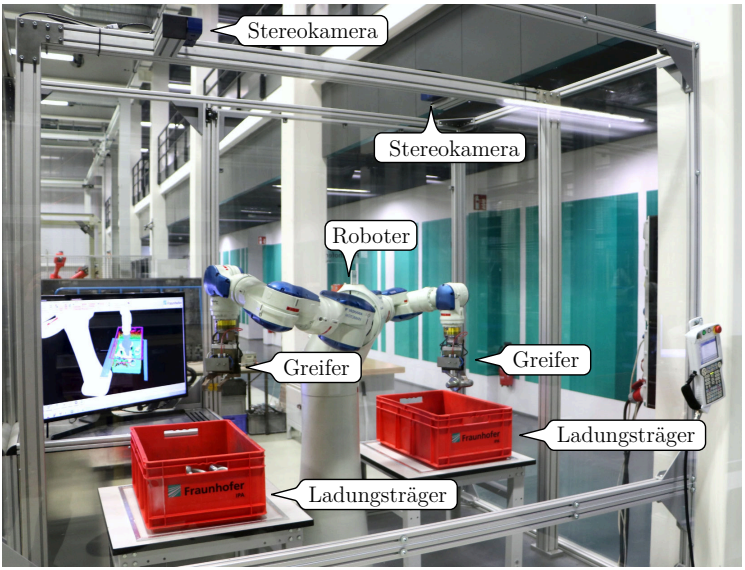


Abbildung 5.2: Versuchsaufbau zur Vereinzelung von Ringschrauben aus Kleinladungsträgern mit einem Yaskawa Motoman SDA-10F

die Werkstücke nacheinander von der Zwischenablage und lässt sie aus einer zufälligen Position in den rechten Ladungsträger fallen. Wenn die linke Kiste vollständig geleert wurde oder auch in drei Versuchen kein weiteres Werkstück entnommen werden kann, wird die Richtung des Materialflusses umgekehrt, so dass die Werkstücke aus der rechten Kiste gegriffen und in die linke Kiste fallen gelassen werden. Auf diese Weise können die Ladungsträger in den Versuchen vollständig entleert werden, so dass während dieser Versuche Situationen mit unterschiedlichen Füllgraden auftreten.

5.1.2 Versuchsaufbau für Ringschrauben

Der zweite Versuchsaufbau (Abbildung 5.2) besteht aus einem Zweiarm-Roboter Yaskawa Motoman SDA10F mit zwei pneumatischen Zweifinger-Klemmgreifern des Modells PGF80 der Firma Schunk. Auch in diesem Versuchsaufbau kommen Kollisionssensoren (hier vom Typ Schunk OPR 061) zum Einsatz, um auf Kollisionen reagieren zu können. Die Werkstücke in diesem Versuchsaufbau sind M30-Ringschrauben nach der Norm DIN 580, die aus Kleinladungsträgern (KLT) der Größe 600 mm × 400 mm × 240 mm vereinzelt werden. Der Versuchsaufbau besteht ebenfalls aus zwei Stellplätzen für Ladungsträger, über denen jeweils eine Stereokamera des

Modells N20-1202-16-BL von Ensenso montiert ist. Während der Versuche werden abwechselnd mit beiden Roboterarmen Werkstücke aus einem KLT gegriffen und in den anderen KLT fallen gelassen. Dies erfolgt wiederum so lange, bis der erste KLT leer ist oder kein weiterer geeigneter Griff gefunden werden kann. Danach wird, wie im vorherigen Versuchsaufbau, die Richtung des Materialflusses umgekehrt. Die Werkstücke werden somit aus dem zweiten KLT gegriffen und in den ersten KLT fallen gelassen.

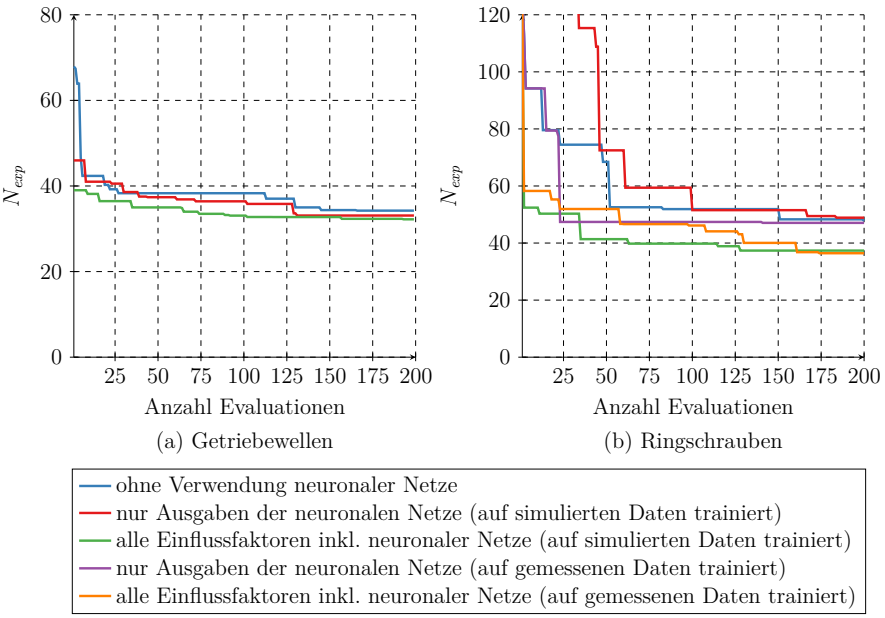
5.1.3 Objektlageerkennung und Vorgehen zur Durchführung der Versuche

Da in einer realen Roboteranlage zur Vereinzelung von ungeordnet gelagerten Werkstücken, im Gegensatz zu einer Simulation, die Objektlage der Werkstücke nicht bekannt ist, muss diese für die folgenden Versuche anhand der Sensordaten bestimmt werden. Im Umfeld der in dieser Arbeit beschriebenen Greifplanung wurde am Fraunhofer IPA ein Verfahren zur Objektlageerkennung von Flächenmodellen in Punktwolken entwickelt. Dabei handelt es sich um einen ansichtenbasierten Ansatz, bei dem aus einem Flächenmodell Tiefenbilder aus diversen Ansichten generiert und mit Projektionen der Punktwolke abgeglichen werden (Palzkill et al. 2010; Palzkill et al. 2012; Palzkill 2014). Dieses Verfahren wird in den folgenden Versuchen zur Bestimmung der Werkstücklage verwendet. Im Gegensatz zu den bisherigen Versuchen ist die Lage des Werkstücks dadurch mit einer unbekannten Toleranz behaftet, sowohl translatorisch als auch rotatorisch. Weiterhin kann nicht davon ausgegangen werden, dass alle höchstliegenden Werkstücke erkannt werden.

Das Verfahren zur Objektlageerkennung wurde in Kombination mit der in dieser Arbeit beschriebenen Greifplanung verwendet, um in den beiden Versuchsaufbauten die Werkstücke aus den jeweiligen Ladungsträgern zu entnehmen. Um eine Menge an vergleichbaren Versuchsdaten zu generieren, wurden dabei die Ladungsträger mehrfach weitgehend entleert und die jeweiligen Sensordaten samt der zugehörigen erkannten Werkstücklage abgespeichert. Diese Daten kommen in den folgenden Versuchen zum Einsatz.

5.1.4 Ergebnisse und Diskussion

In diesem Abschnitt wird die Leistung der in dieser Arbeit entwickelten Greifplanung auf den gemessenen Daten der beiden vorgestellten Versuchsaufbauten untersucht. Dabei wird insbesondere die Unterstützung der Suche durch neuronale Netze analysiert. Für die Untersuchungen werden 340 reale Situationen der Getriebewellen und 400 reale Situationen der Ringschrauben betrachtet



(a) Getriebewellen (b) Ringschrauben

Abbildung 5.3: Ergebnisse der Optimierung auf gemessenen Daten der Laboraufbauten. Dargestellt ist der minimal erreichte Wert für die Anzahl N_{exp} der notwendigen expandierten Knoten bei gemessenen Zustandsdaten in Abhängigkeit von der Anzahl an Evaluationen der Optimierungsverfahren mit unterschiedlichem Einfluss der neuronalen Netze. Für das Szenario der Ringschraube sind zusätzlich die Ergebnisse für auf gemessenen Daten trainierte neuronale Netze dargestellt.

und die Parameter, wie in Abschnitt 4.6.8, mittels evolutionären Algorithmen optimiert. Die detaillierten Parameter für diesen Versuch können Parametersatz 6.1 und 6.2 in Tabelle A.6 in Anhang A.2 entnommen werden. Abbildung 5.3 zeigt den jeweiligen Verlauf der Optimierung mit und ohne Verwendung der neuronalen Netze.

Dabei ist bei beiden Szenarien zu sehen, dass sich die Anzahl N_{exp} der notwendigen expandierten Knoten bei Verwendung der neuronalen Netze auf einen vergleichbaren Wert optimieren lässt, wie bei Verwendung der manuell definierten Einflussfaktoren. Wie auch bei den simulierten Daten lassen sich somit die manuell definierten Einflussfaktoren durch die Ausgabewerte der neuronalen Netze ersetzen. Bei beiden Szenarien führt die Verwendung aller Einflussfaktoren zur geringsten Anzahl an expandierten Knoten. Im Szenario der Getriebewellen ist der Unterschied jedoch nur gering. Für das Szenario der Ringschrauben wird zusätzlich untersucht, wie die neuronalen Netze die Suche beschleunigen können, wenn sie auf gemessenen Daten trainiert wurden (für das

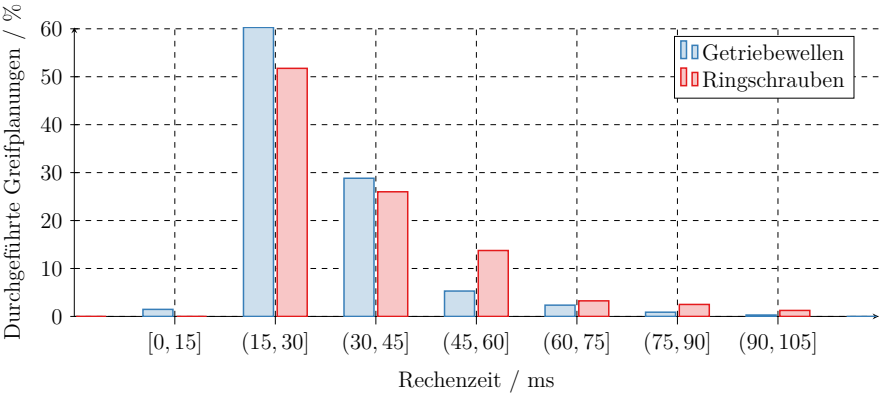


Abbildung 5.4: Rechenzeiten der Greifplanung in den Laboraufbauten. Dargestellt ist die Häufigkeitsverteilung verschiedener Rechenzeitbereiche der beiden Laboraufbauten für Getriebewellen und Ringschrauben

Szenario der Getriebewellen stand keine ausreichende Anzahl an Trainingsdaten zur Verfügung). Dieses Training basiert, wie das Training mit simulierten Daten, auf 1000 unterschiedlichen Situationen. Dabei ist zu sehen, dass auf gemessenen Daten trainierte neuronale Netze die Suche in geringem Maße weiter beschleunigen können. Trotz des höheren Aufwands bei der Datengenerierung stellt das Training mit gemessenen Daten ein industriell einsetzbares Szenario dar. In einer realen Anwendung kann die Roboteranlage in der Anfangsphase mit neuronalen Netzen betrieben werden, die auf simulierten Daten trainiert wurden. Die Situationen in dieser Anfangsphase können automatisch abgespeichert und im Anschluss die neuronalen Netze mit diesen Daten neu trainiert werden, um die Leistung der Roboteranlage zu erhöhen.

Abbildung 5.4 zeigt schließlich die Verteilung der Rechenzeiten für die Greifplanung in den beiden Laboraufbauten unter Verwendung der optimalen Parameter mit auf gemessenen Daten trainierten neuronalen Netzen. Die Versuche zur Bestimmung der Rechenzeit wurden auf einem Intel Core i7-4900MQ mit 2,8 GHz und 16GB DDR3 Arbeitsspeicher durchgeführt. In beiden Szenarien liegt die Rechenzeit meistens zwischen 15 ms (ausschließlich) und 30 ms (einschließlich). Längere Rechenzeiten kommen zunehmend seltener vor. Durch die höhere Werkstückkomplexität und die schlechtere Zugänglichkeit aufgrund der kleinen Kiste treten beim Szenario der Ringschrauben mehr längere Rechenzeiten auf als beim Szenario der Getriebewellen. Die mittlere Rechenzeit liegt für das Szenario der Getriebewellen bei 31,7 ms und für das Szenario der Ringschrauben bei 35,3 ms. Zusammengefasst zeigen diese Ergebnisse, dass die Greifplanung auf gemessenen Daten nur eine geringe Rechenzeit benötigt und die Geschwindigkeit des in dieser Arbeit vorgestellten Ansatzes damit auch für reale Roboteranlagen ausreichend ist.

5.2 Industrielle Validierung

Versuchsaufbauten im Labor haben den Vorteil, dass sich die Versuchsparameter gut kontrollieren und die Versuche umfangreich beobachten lassen. Allerdings sind solche Versuche sehr aufwändig und daher nur zeitlich begrenzt möglich. Ein Einsatz im Produktionsbetrieb ermöglicht hingegen eine langfristige Evaluierung und erhöht die Anzahl an auswertbaren Greifvorgängen um ein Vielfaches. In diesem Abschnitt werden Ergebnisse einer Realisierung bei der MAN Truck & Bus AG vorgestellt, die auf Daten von über 10 000 Betriebsstunden basieren.

5.2.1 Beschreibung des industriellen Aufbaus

Der in dieser Arbeit vorgestellte Ansatz zur Greifplanung wurde bereits in mehreren industriellen Realisierungen eingesetzt und erprobt, unter anderem in zwei Roboteranlagen bei der MAN Truck & Bus AG. Diese beiden Anlagen dienen der Vereinzelung und Zuführung von Pleuelstangen, die überwiegend ungeordnet aus Gitterboxen entnommen werden. Abbildung 5.5 zeigt eine Gitterbox mit Werkstücken in einer dieser Anlagen (Anlage 1). Anlage 2 ist gespiegelt, aber ansonsten nahezu identisch. Zum Greifen der Werkstücke kommt in beiden Anlagen ein pneumatischer Klemmgreifer mit zwei zusätzlichen Bewegungsachsen zum Einsatz, der am Industrieroboter KUKA Quantec KR150 R3100 prime montiert ist. Als Sensorik kommen jeweils zwei Lasertriangulationssensoren des Typs weCat3D MLWL2 der Firma Wenglor zum Einsatz. Für die Objektlageerkennung wurde das selbe Verfahren wie in den Laborversuchen verwendet (Palzkill et al. 2010; Palzkill et al. 2012; Palzkill 2014). Die Software für die Greifplanung wurde auf einem Intel Xeon E5-1650 v3 mit 3,5 GHz und 16GB DDR4 Arbeitsspeicher ausgeführt. Die verwendeten Parameter finden sich in Parametersatz 7.1 in Tabelle A.7 in Anhang A.2.

Die beiden Anlagen verfügen jeweils über drei Stellplätze mit Gitterboxen, aus denen abwechselnd unterschiedliche Werkstückvarianten gegriffen werden. Prinzipiell werden in beiden Roboteranlagen alle Werkstückvarianten vereinzelt, allerdings ist die Verteilung und Auslastung stark unterschiedlich. Der Ablauf für diese Anlagen sieht vor, dass zunächst der Ladungsträger, aus dem das nächste Werkstück entnommen werden soll, durch den 3D-Sensor erfasst wird. In der erzeugten Punktwolke wird zunächst die Lage des Ladungsträgers und im Anschluss der Füllgrad bestimmt. Falls der Ladungsträger leer ist, wird ein manueller Wechsel des Ladungsträgers veranlasst. Andernfalls werden die einzelnen Werkstücke erkannt und im Anschluss die in dieser Arbeit vorgestellte Greifplanung durchgeführt. Wenn ein geeigneter Griff samt Entnahmebahnen gefunden wurde, wird dieser durch den Roboter und den Greifer ausgeführt und das Werkstück anschließend abgelegt. Danach beginnt der Ablauf von vorne. Falls einer der beschriebenen Schritte fehlschlägt, wird eine erneute Erfassung der Szene durch den 3D-Sensor durchgeführt.

Abbildung 5.5: Gitterbox mit Pleuelstangen in einer industriellen Anlage zur Vereinzelung und Zuführung von Werkstücken bei der MAN Truck & Bus AG

Da die Sensordaten verrauscht sind, führt dies zu einer geringfügig unterschiedlichen Punktwolke, so dass der problematische Prozessschritt im zweiten Versuch häufig erfolgreich ist. Abbildung 5.6 visualisiert den Gesamtablauf der industriellen Realisierung. Der Füllgrad $f_L \in [0, 1]$ des Ladungsträgers berechnet sich mittels

$$f_L = \frac{{}^L\bar{z}_p}{d_z^L}. \tag{5.1}$$

Dabei beschreibt ${}^L\bar{z}_p$ den Mittelwert der z-Koordinaten aller Punkte der Punktwolke, die sich innerhalb des Ladungsträgers befinden, im Ladungsträgerkoordinatensystem $\mathcal{K}_L$. Der Nenner d_z^L ist die Höhe des Ladungsträgers. Befinden sich keine Punkte innerhalb des Ladungsträgers so gilt $f_L = 0$.

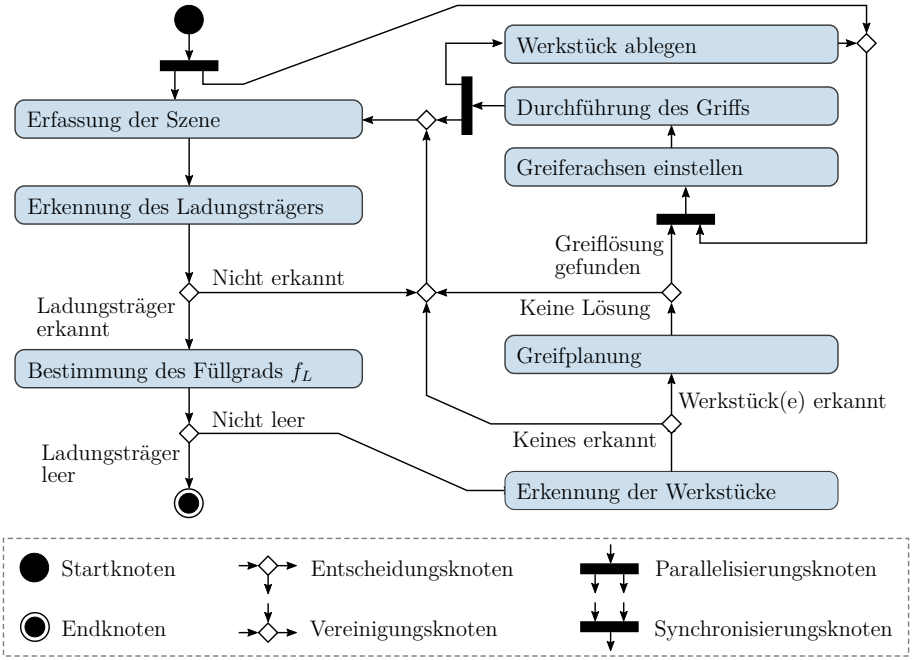


Abbildung 5.6: Ablauf der industriellen Realisierung als UML-Aktivitätsdiagramm

5.2.2 Ergebnisse und Diskussion

Aufgrund der anspruchsvollen Werkstückgeometrie und der hohen Kisten ist es in dieser Anwendung nicht immer möglich, einen geeigneten Griff zu bestimmen. Im Betrieb gelingt dies nur in 94,6 % der Situationen. Die Ursache dafür sind eine große Störkontur des Greifers, die jedoch aufgrund der durch das Werkstückgewicht auftretenden Kräfte notwendig ist, und eine schlechte Zugänglichkeit bei Werkstücken in der Nähe der Seitenwände der Ladungsträger. Für die im Folgenden präsentierten Ergebnisse werden insgesamt 1 543 215 verschiedene Situationen der beiden Anlagen untersucht. Dies entspricht mehr als 10 000 Betriebsstunden. Die mittlere Rechenzeit für die Greifplanung beträgt dabei 193 ms und liegt somit deutlich unter dem in Abschnitt 1.3 definierten Ziel von einer Sekunde. Abbildung 5.7 zeigt die Verteilung der Rechenzeiten. Dabei fällt zunächst auf, dass auf Anlage 1 deutlich mehr Werkstücke vereinzelt wurden, als auf Anlage 2. Dies liegt daran, dass Anlage 1 permanent für die Werkstücke mit hohen Stückzahlen verwendet wird und Anlage 2 seltener betrieben wird. Weiterhin ist zu sehen, dass Rechenzeiten im Bereich von 100 ms bis 150 ms am häufigsten vorkommen. Rechenzeiten unter 50 ms und Rechenzeiten über 600 ms treten mit 1,6 % bzw. 1,8 % aller Fälle nur sehr selten auf. Diese Ergebnisse zeigen, dass das Verfahren in den meisten Fällen schnell eine Lösung findet und deutet darauf hin, dass Situationen, in denen sehr viele Knoten im Suchbaum expandiert werden müssen, nur selten vorkommen. Um dies zu verifizieren, zeigt Abbildung 5.8 die Verteilung der Anzahl der Knoten, die expandiert werden müssen, um eine geeignete Greiflösung zu finden. Dabei ist zu sehen, dass 300 bis 400 der am häufigsten vorkommende Bereich für die Anzahl

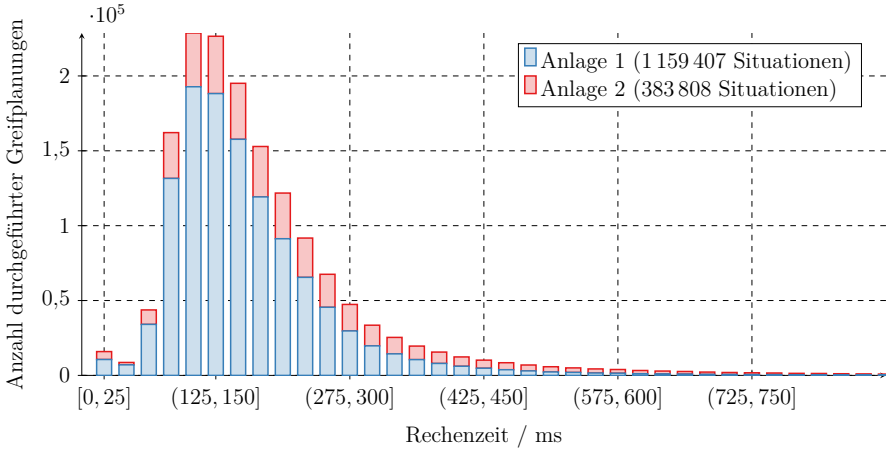


Abbildung 5.7: Rechenzeiten der Greifplanung in der industriellen Validierung. Dargestellt ist die Häufigkeitsverteilung verschiedener Rechenzeitbereiche.

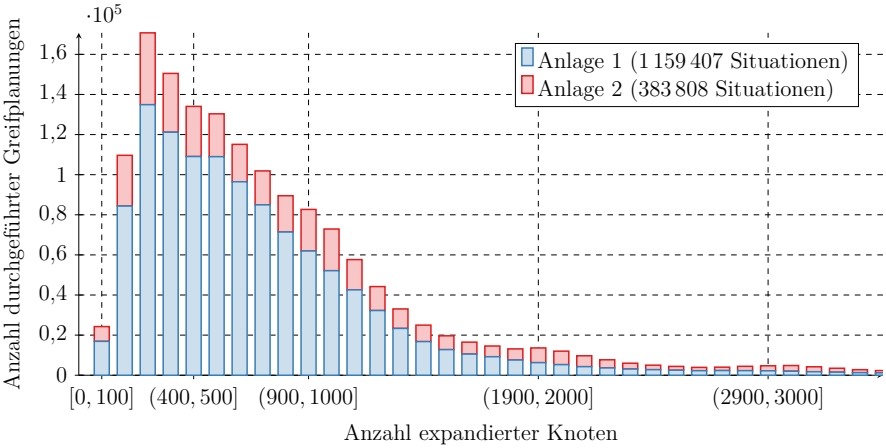


Abbildung 5.8: Anzahl der expandierten Knoten in der industriellen Validierung. Dargestellt ist die Häufigkeitsverteilung verschiedener Bereiche für die Anzahl an expandierten Knoten.

expandierter Knoten ist. Weiterhin ähnelt die Abbildung 5.8 der Verteilung der Rechenzeiten in Abbildung 5.7, wobei die Anzahl der expandierten Knoten oberhalb des Maximums etwas stärker gestreut ist. Jedoch zeigt auch die Verteilung der Anzahl der expandierten Knoten, dass das Suchverfahren in den meisten Situationen nur wenige Knoten expandieren muss, um eine Lösung zu finden. Insgesamt werden im Durchschnitt 913,6 Knoten expandiert.

Abbildung 5.9 zeigt die mittlere Rechenzeit und die Anzahl der expandierten Knoten des Suchbaums in Abhängigkeit von der Anzahl der erkannten Werkstücke für Anlage 1. Das Verfahren zur Objektlageerkennung erkennt in jeder Situation maximal 25 Werkstücke. Um eine zuverlässige Datengrundlage zu haben, sind jedoch nur Werkstückanzahlen berücksichtigt, die in mindestens 1000 Situationen auftraten. In dieser Abbildung ist zu sehen, dass die Rechenzeit für ein einzelnes Werkstück am geringsten ist. Dies ist dadurch zu erklären, dass bei nur einem Werkstück die Anzahl der potenziellen Greifmöglichkeiten stark eingeschränkt ist. Weiterhin ist erkennbar, dass die Rechenzeit bei drei Werkstücken ihr Maximum erreicht hat und danach langsam etwas abfällt. Dieser Abfall ist damit zu erklären, dass es bei entsprechend vielen Werkstücken leichter wird, eine geeignete Greiflösung zu finden. Die Anzahl der expandierten Knoten verhält sich vergleichbar. Insgesamt hat die Anzahl der erkannten Werkstücke jedoch nur einen geringen Einfluss auf die Rechenzeit und auf die Anzahl der expandierten Knoten. Dies zeigt, dass die hier vorgestellte Greifplanung ohne Modifikationen sowohl mit Objektlageerkennungsverfahren eingesetzt werden kann, die nur wenige Werkstücke erkennen, als auch mit

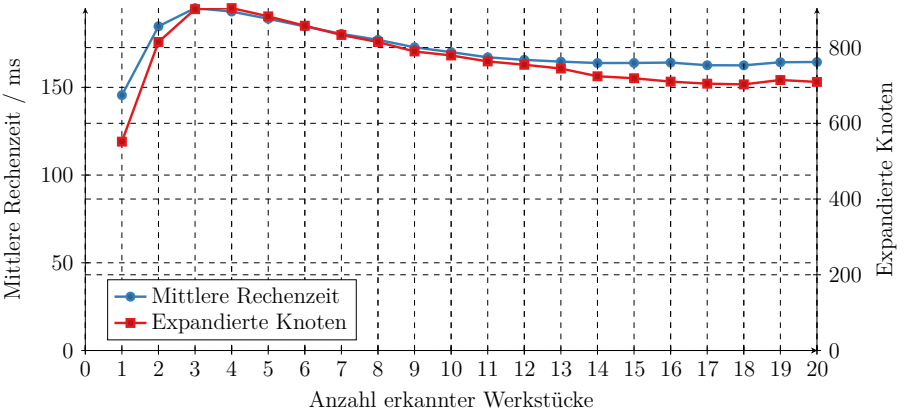


Abbildung 5.9: Rechenzeit und Anzahl expandierter Knoten in Abhängigkeit von der Anzahl an erkannten Werkstücken für Anlage 1.

Ansätzen, die in einer gegebenen Situation sehr viele Werkstücke erkennen. Dies unterstützt die Aussage, dass die entwickelte Greifplanung mit beliebigen Verfahren zur Objektlageerkennung eingesetzt werden kann, was in Abschnitt 1.3 als Anforderung an das Verfahren festgelegt wurde, auch praktisch.

Abbildung 5.10 zeigt die mittlere Rechenzeit und die Anzahl der expandierten Knoten in Abhängigkeit vom Füllgrad f_L des Ladungsträgers für Anlage 1. Eine solche Auswertung liefert nur aufgrund der sehr großen Datenbasis und der vielen Kistenentleerungen eine belastbare Aussage. An dieser Stelle sei angemerkt, dass die Kisten, die der Anlage zugeführt wurden, meist nur zu weniger als 70 % gefüllt waren. Bei dieser Auswertung ist zu sehen, dass die Rechenzeit bei einem Füllgrad von ungefähr 10 % bis 15 % am höchsten ist. Dies ist dadurch zu erklären, dass sich während der Entleerung des Ladungsträgers mit der Zeit einige problematische Werkstücksituationen bilden. Diese müssen aufgelöst werden, sobald bei einer weitgehend entleerten Kiste keine einfach greifbaren Werkstücke mehr verbleiben. In Abbildung 5.5 ist eine solche Situation zu sehen. Bei fast leereren Kisten, bei denen die problematischen Situationen bereits aufgelöst und die verbleibenden Werkstücke am Kistenboden einfach zu greifen sind, ist die Rechenzeit etwas geringer, wie in Abbildung 5.10 zu sehen. Bei volleren Kisten sind meist genügend Werkstücke greifbar, so dass die Rechenzeit auch hier geringer ist. Die Anzahl der expandierten Knoten verhält sich ähnlich, hat jedoch das Maximum bei einem Füllgrad von ungefähr 25 %. Hier ist ein leichter Anstieg bei sehr vollen Kisten zu beobachten. Eine mögliche Erklärung dafür sind Einschränkungen durch die Roboterkinematik bei hochliegenden Werkstücken.

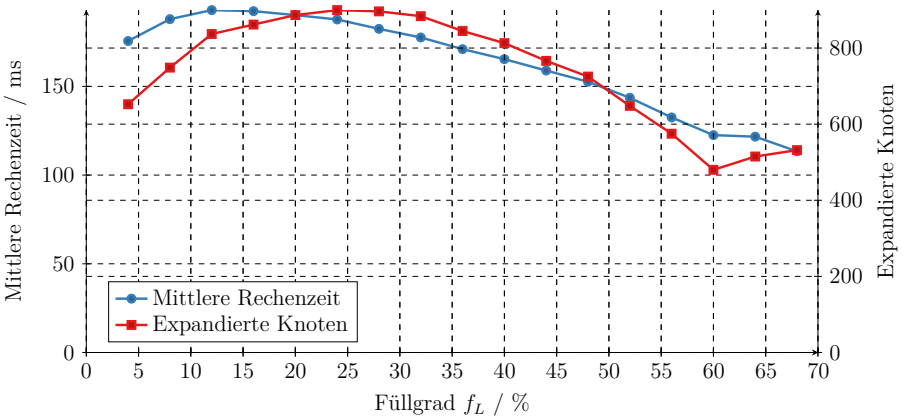


Abbildung 5.10: Rechenzeit und expandierte Knoten in Abhängigkeit vom Füllgrad des Ladungsträgers für Anlage 1. Dargestellt sind nur Füllgrade, die in mindestens 1000 Situationen auftraten.

Abbildung 5.11 zeigt schließlich die Erfolgsquote in Abhängigkeit von der Anzahl erkannter Werkstücke und in Abhängigkeit vom Füllgrad des Ladungsträgers für Anlage 1. Diese Abbildung gibt keine Auskunft darüber, wie schnell das Suchverfahren eine Lösung findet, sondern lediglich darüber, wie schwierig es ist, in bestimmten Situationen eine Lösung zu finden. In Abbildung 5.11(a) ist zu sehen, dass die Wahrscheinlichkeit, eine Lösung zu finden, wenn nur ein Werkstück erkannt wurde, lediglich bei 61,8 % liegt. Erst wenn drei Werkstücke erkannt werden, steigt die Wahrscheinlichkeit, eine Lösung zu finden, auf 95,3 %. Die geringe Wahrscheinlichkeit bei wenigen erkannten Werkstücken erklärt sich, wie zu Beginn dieses Abschnitts bereits beschrieben, durch die anspruchsvolle Werkstückgeometrie und die hohen Kisten. Den Einfluss der hohen Kistenwände verdeutlicht auch Abbildung 5.11(b). Hier ist zu erkennen, dass die Erfolgsquote bei fast leeren Kisten signifikant geringer ist, als bei Kisten mit einem hohen Füllgrad.

5.2.3 Auswertung von Fehlgriffen

In Abschnitt 4.3.6 wurde ausgeführt, dass die statischen Prioritäten der einzelnen Greifposen so gewählt werden können, dass statistisch erfolgreiche Griffe bevorzugt werden. Diese Statistik der Erfolgswahrscheinlichkeit einzelner Greifposen kann erstellt werden, indem der Roboters nach jedem Griff rückmeldet, ob die Entnahme erfolgreich war. Nicht erfolgreiche Entnahmeversuche werden im Folgenden als *Fehlgriffe* bezeichnet. In der beschriebenen industriellen

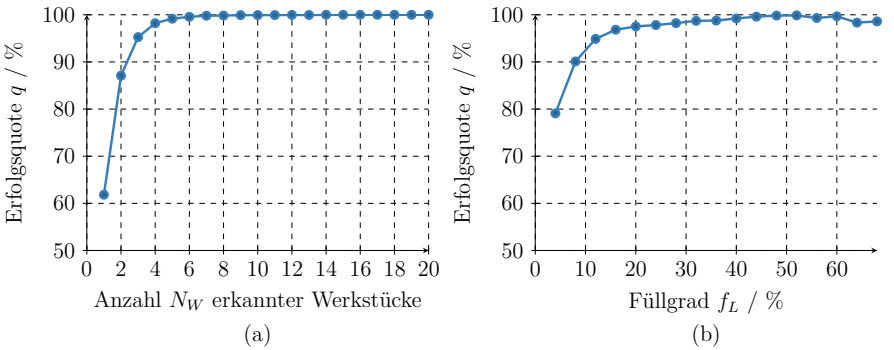


Abbildung 5.11: Erfolgsquote der Greifplanung in der industriellen Realisierung. Die Abbildung zeigt die Erfolgsquote in Abhängigkeit von der Anzahl erkannter Werkstücke und in Abhängigkeit vom Füllgrad des Ladungsträgers für Anlage 1. Dargestellt sind nur Werkstückanzahlen und Füllgrade, die in mindestens 1000 Situationen auftraten.

Realisierung wird diese Rückmeldung vom Roboter ebenfalls erfasst. Gründe für diese Fehlgriffe sind hauptsächlich falsche oder ungenaue Objektlageerkennungen sowie andere Ungenauigkeiten, beispielsweise aufgrund der Hand-Auge-Kalibrierung oder der geringen Absolutgenauigkeit des Roboters. Abbildung 5.12 zeigt die Fehlerrate in Abhängigkeit von den einzelnen verwendeten Greifposen G_1 bis G_{15}. Aufgrund der hohen Anzahl an unterschiedlichen Greifposen werden diese hier nicht näher beschrieben. In der Abbildung ist deutlich zu sehen, dass die Fehlerrate zwischen den einzelnen Griffen stark variiert. Der Korrelationskoeffizient nach Pearson (1896) zwischen der Fehlerrate und dem Anteil der jeweiligen Greifpose an der Gesamtanzahl aller Griffe liegt bei $-0{,}309$. Dies zeigt, dass Greifposen, die eine höhere Fehlerrate aufweisen, seltener durch den Roboter gegriffen werden. Somit führen die bei dieser Realisierung konfigurierten Prioritäten der Greifposen zu einer Verringerung der durchschnittlichen Gesamtfehlerrate und damit zu einer Verringerung der effektiven Taktzeit.

Abbildung 5.13 zeigt die Fehlerrate in Abhängigkeit vom jeweiligen Füllgrad des Ladungsträgers. Hier ist zu sehen, dass die Fehlerrate mit sinkendem Füllgrad zunimmt. Dies lässt sich dadurch erklären, dass die Auflösung des Sensors mit steigendem Abstand abnimmt. Somit sinkt auch die Genauigkeit der erkannten Werkstücklagen und es steigt die Gefahr von falsch erkannten Werkstücken. Beide Faktoren können zu vermehrten Fehlgriffen führen.

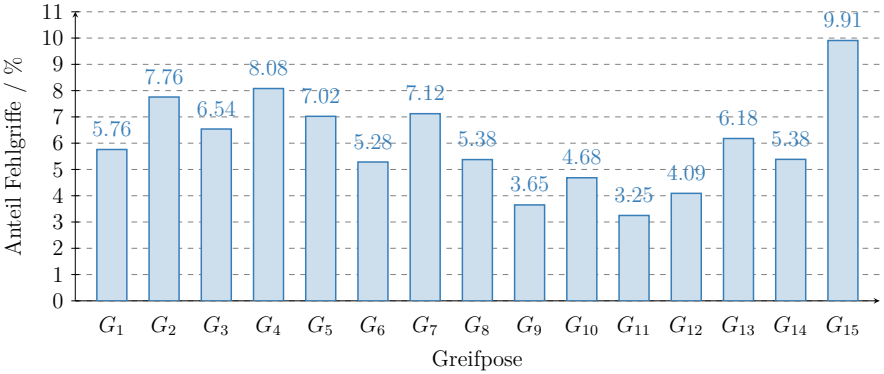


Abbildung 5.12: Anteil der Fehlgriffe für einzelne Greifposen für Anlage 1.

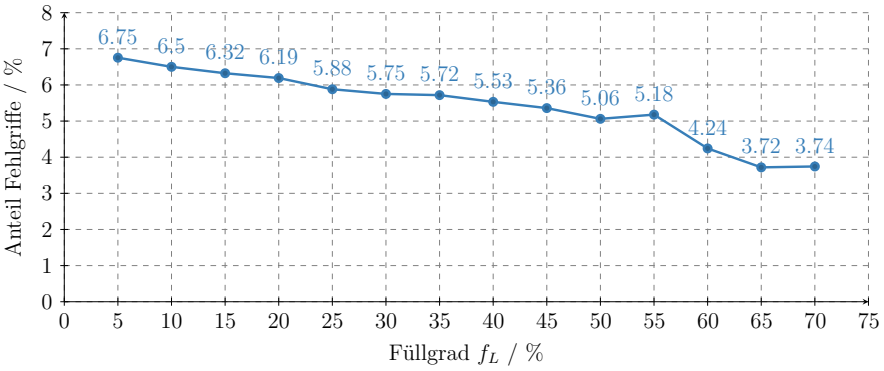


Abbildung 5.13: Anteil der Fehlgriffe in Abhängigkeit vom Füllgrad des Ladungsträgers für Anlage 1.

5.3 Zusammenfassung

In diesem Kapitel wurde die entwickelte Greifplanung an zwei Versuchsaufbauten und zwei industriellen Anlagen validiert. Dazu wurden zunächst die beiden Versuchsaufbauten vorgestellt, die den beiden, in den vorherigen Kapiteln verwendeten, simulierten Szenarien entsprechen und das Vorgehen zur Versuchsdurchführung beschrieben. Im Anschluss wurden verschiedene Kombinationen von Einflussfaktoren für die in dieser Arbeit entwickelte Greifplanung experimentell untersucht. Dabei kamen, wie bereits in Kapitel 4, bayessche Optimierungsverfahren und evolutionäre Algorithmen zum Einsatz, um jeweils die optimalen Parameter für die Heuristikfunktion zu bestimmen. Diese Untersuchung hat gezeigt, dass die Greifplanung auch für Daten realer Roboteranlagen geeignet ist. Zudem wurde gezeigt, dass die Verwendung von neuronalen Netzen eine manuelle Definition von Einflussfaktoren für die Heuristikfunktion ersetzen kann. Wurden die neuronalen Netze auf simulierten Daten trainiert, so ließen sich vergleichbare Ergebnisse erzielen. Eine Kombination aus manuell definierten Einflussfaktoren und auf gemessenen Daten trainierten neuronalen Netzen führte zur geringsten Anzahl an evaluierten Knoten während der Suche und damit zum besten Ergebnis.

Weiterhin wurden statistische Daten aus zwei industriellen Produktionsanlagen zur Vereinzelung von Pleuelstangen untersucht. Dabei wurden 1 543 215 einzelne Griffe ausgewertet. Die mittlere Rechenzeit für die Greifplanung lag bei 193 ms. Damit lässt sich das definierte Ziel einer geringen mittleren Taktzeit auch für industrielle Anwendungen erfüllen. In einer detaillierteren Auswertung wurde untersucht, wie die Rechenzeit von der Anzahl der erkannten Werkstücke und dem Füllgrad des Ladungsträgers abhängt. Dabei konnte gezeigt werden, dass die mittlere Rechenzeit der Greifplanung nur geringfügig durch die Anzahl der erkannten Werkstücke beeinflusst wird. Somit lässt sich die in dieser Arbeit entwickelte Greifplanung mit unterschiedlichen Ansätzen der Objektlageerkennung einsetzen, auch wenn diese im Mittel mehr oder weniger Werkstücke erkennen als der in diesem Kapitel verwendete Ansatz von Palzkill (2014). Weiterhin wurde analysiert, wie die Wahrscheinlichkeit, eine gültige Lösung zu finden, von der Anzahl der erkannten Werkstücke und dem Füllgrad des Ladungsträgers abhängt. Diese Untersuchung hat gezeigt, dass die Wahrscheinlichkeit, eine Lösung zu finden, wenn nur ein Werkstück erkannt wurde, lediglich bei 61,8 % liegt. Erst wenn drei Werkstücke erkannt werden, steigt die Wahrscheinlichkeit, eine Lösung zu finden, auf 95,3 %. Abschließend wurde die Anzahl der fehlgeschlagenen Greifversuche untersucht. Es zeigt sich, dass die Wahrscheinlichkeit für diese Fehlgriffe steigt, je leerer die Kiste wird. Insgesamt wurde durch die Auswertung von über 1,5 Mio. einzelnen Griffen die Tauglichkeit für den industriellen Einsatz nachgewiesen.

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung

Das Ziel dieser Arbeit ist die Entwicklung und Untersuchung eines Verfahrens zur Greifplanung für die modellbasierte Entnahme ungeordnet gelagerter Werkstücke mit gegebener, starrer Geometrie und vordefinierten Greifposen aus Kisten oder anderen Ladungsträgern. Die Greifplanung soll es einem Roboter ermöglichen, die Werkstücke aus dem Ladungsträger zu entnehmen und mit definierter Lage einer Maschine zuzuführen. Der Fokus der Entwicklung liegt auf einer kurzen Taktzeit, einer hohen Verfügbarkeit, einer vollständigen Kistenentleerung sowie einer hohen Flexibilität, um das Verfahren in möglichst vielen Anwendungsfällen einsetzen zu können.

In dieser Arbeit wurde dazu ein Ansatz entwickelt, bei dem ein Suchbaum so konstruiert wird, dass alle relevanten Aspekte der Greifplanung enthalten sind. Dieser Suchbaum repräsentiert alle potenziell möglichen Griffe für eine gegebene Situation. Kriterien, die einen erfolgreichen Griff verhindern können, werden beim Expandieren des Suchbaums überprüft, sodass jeder Zielknoten einen durchführbaren und insbesondere kollisionsfreien Entnahmevorgang repräsentiert. Durch die Integration der einzelnen Bahnposen in den Suchbaum erfolgt die Bahnplanung ebenfalls als Teil des entwickelten Verfahrens. Nach Abschluss der Suche werden alle relevanten Informationen, die für die Ausführung des Griffes notwendig sind, aus dem Suchbaum ausgelesen. Dadurch kann das Problem der Greifplanung in eine Baumsuche überführt werden.

Um trotz des großen Suchraums eine schnelle Rechenzeit zu gewährleisten und somit das Ziel der kurzen Taktzeit zu erreichen, wird die Suche in Richtung einer wahrscheinlichen Lösung gelenkt. Dies erfolgt mittels Bestensuche und einer dafür notwendigen Heuristikfunktion, die auf verschiedenen Einflussfaktoren basiert. In dieser Arbeit wurde untersucht, wie diese Einflussfaktoren mit der Wahrscheinlichkeit zusammenhängen, eine gültige Greiflösung zu finden. Dabei zeigt sich, dass die vertikale Position des Werkstücks, der Abstand zu den Wänden des Ladungsträgers sowie die Orientierung des TCP einen Einfluss auf die Wahrscheinlichkeit haben, einen kollisionsfreien Griff zu finden. Um die Greifplanung unabhängiger von manuell definierten Einflussfaktoren zu machen, wurde weiterhin gezeigt, dass sich künstliche neuronale Netze eignen, um die relevanten Eigenschaften, die zu einer schnellen Lösungsfindung führen, selbstständig

zu lernen. Experimentelle Versuche haben ergeben, dass die Suche unter Verwendung dieser neuronalen Netze weniger Knoten expandieren muss, um eine Lösung zu finden. Zudem lässt sich dadurch die Anzahl an Parametern, die konfiguriert werden müssen, reduzieren und damit den Inbetriebnahmeaufwand für den Einsatz in einem neuen Anwendungsfall verringern, da weniger Expertenwissen notwendig ist.

Das in dieser Arbeit entwickelte Verfahren zur Greifplanung wurde in Simulationen und Laborversuchen evaluiert. Im Rahmen eines realen Produktionseinsatzes zur Vereinzelung von Pleuelstangen bei der MAN Truck & Bus AG wurde zudem gezeigt, dass der in dieser Arbeit entwickelte Lösungsansatz industrietauglich ist. Innerhalb der analysierten Produktionsphase wurden in über 10 000 Betriebsstunden insgesamt 1 543 215 Werkstücke aus Gitterboxen vereinzelt. Die mittlere Rechenzeit für die Greifplanung betrug dabei 193 ms, wodurch die geforderte Gesamttaktzeit der Anlage erreicht werden konnte.

6.2 Ausblick

Im Rahmen dieser Arbeit wurden alle Greiflösungen als gleichwertig betrachtet. Um die Taktzeit weiter zu reduzieren, kann es jedoch sinnvoll sein, die erwartete Dauer für die jeweilige Bewegungsbahn zu berücksichtigen und Greiflösungen zu bevorzugen, die zu einer kürzeren Roboterbewegung führen. Ebenso könnten weitere Kriterien berücksichtigt werden, die bei der Entnahme eine Rolle spielen, beispielsweise wie groß das Risiko ist, dass sich Werkstücke bei der Entnahme verhaken oder verklemmen. Dies kann zu Problemen bei der Entnahme, dem Transport und insbesondere bei der Ablage der Werkstücke führen und sollte daher vermieden werden. Auch die Auswirkungen der Wahl des zu greifenden Werkstücks auf die nachfolgenden Entnahmen sollten näher untersucht werden. Beispielsweise könnte es sinnvoll sein, bevorzugt ein Werkstück zu greifen, das viele andere Werkstücke verdeckt, sodass sowohl die Erkennung als auch die Entnahme der anderen Werkstücke in nachfolgenden Situationen vereinfacht werden.

Weiterhin wurden in dieser Arbeit die Bewegungsbahnen des Roboters auf Basis von festen Regeln bestimmt. Diese Regeln sind so definiert, dass die Gefahr von Kollisionen möglichst gering ist und die Bewegungsbahnen nur in Ausnahmefällen nicht durchführbar sind. Durch die Prüfung auf Kollisionen und Singularitäten entlang der Bewegungsbahn ist das Verfahren für die praktische Anwendung ausreichend. Durch die Integration der einzelnen Bahnposen in den Suchbaum ist jedoch bereits eine Flexibilität hinsichtlich der Bewegungsbahn vorgesehen. Zukünftige wissenschaftliche Arbeiten könnten untersuchen, auf welche Art alternative Bahnposen generiert werden und welche Kriterien für die Bewegungsbahn bei der Entnahme ungeordneter Werkstücke relevant sind. Neben der Vermeidung, dass ein Werkstück aus dem Greifer fällt,

könnte die Bewegungsbahn beispielsweise dazu dienen, verhakte Werkstücke zu separieren oder die Unordnung innerhalb des Ladungsträgers für die folgenden Griffe zu minimieren.

In der Heuristikfunktion der in dieser Arbeit entwickelten Greifplanung wurden unter anderem neuronale Netze verwendet. Das Training dieser neuronalen Netze für einen neuen Anwendungsfall erfolgt dabei einmalig vorab. Um schnell mit der Produktion beginnen zu können und dennoch eine kontinuierliche Verbesserung zu erzeugen, könnten während der Produktion weitere Daten gesammelt und die neuronalen Netze regelmäßig auf Basis dieses erweiterten Datensatzes erneut trainiert werden. Damit ließen sich die neuronalen Netze mit jedem Entnahmevorgang verbessern. Neben der Optimierung der Rechenzeit für eine Greiflösung können hier auch Daten über den Erfolg oder die Genauigkeit eines Greifvorgangs berücksichtigt werden. Dies kann beispielsweise durch Sensoren im Greifer, in der Ablagevorrichtung oder über eine separate Kamera festgestellt werden. Somit lernt die Greifplanung, welche Eigenschaften vermehrt zu erfolgreichen Griffen oder zu Ungenauigkeiten führen und ist in der Lage, diese bei der Planung zukünftiger Greifvorgänge zu berücksichtigen. Auch roboterspezifische Ungenauigkeiten oder Verschleiß könnten dadurch möglicherweise kompensiert werden.

Literatur

Ade et al. 1995 Ade, Frank; Rutishauser, Martin; Trobina, Marjan, 1995. Grasping unknown objects. In: Bunke, Horst; Kanade, Takeo; Noltemeier, Hartmut (Hrsg.): *Modelling and Planning for Sensor Based Intelligent Robot Systems*. Singapur, New Jersey, London, Hong Kong: World Scientific, S. 445–459. ISBN 978-981-02-2238-3

Al-Hujazi et al. 1990 Al-Hujazi, Ezzet; Sood, Arun, 1990. Range Image Segmentation with Applications to Robot Bin-Picking Using Vacuum Gripper. *IEEE Transactions on Systems, Man, and Cybernetics*, **20** (6), S. 1313–1325. DOI: 10.1109/21.61203

Asmuni et al. 2005a Asmuni, Hishammuddin; Burke, Edmund K.; Garibaldi, Jonathan M., 2005. Fuzzy Multiple Heuristic Ordering for Course Timetabling. In: Mirkin, Boris; Magoulas, George (Hrsg.): *5th United Kingdom Workshop on Computational Intelligence (UKCI)*. London, 05.09.2005–07.09.2005, S. 302–309

Asmuni et al. 2005b Asmuni, Hishammuddin; Burke, Edmund K.; Garibaldi, Jonathan M.; McCollum, Barry, 2005. Fuzzy Multiple Heuristic Ordering for Examination Timetabling. In: Burke, Edmund K.; Trick, Michael (Hrsg.): *5th International Conference on the Practice and Theory of Automated Timetabling (PATAT)*. Pittsburgh, 18.08.2004–20.08.2004, S. 334–353. Lecture Notes in Computer Science (LNCS)

Berger et al. 2000 Berger, Martin; Bachler, Gernot; Scherer, Stefan, 2000. Vision Guided Bin Picking and Mounting in a Flexible Assembly Cell. In: Logananthara, Rasiah; Palm, Günther; Ali, Moonis (Hrsg.): *Intelligent Problem Solving. Methodologies and Approaches*. New Orleans, 19.06.2000–22.06.2000, S. 109–117

Bergstra et al. 2011 Bergstra, James; Bardenet, Rémi; Bengio, Yoshua; Kégl, Balázs, 2011. Algorithms for Hyper-Parameter Optimization. In: *24th International Conference on Neural Information Processing Systems (NIPS)*. Granada, 12.12.2011–25.12.2011, S. 2546–2554

Bergstra et al. 2013 Bergstra, James; Yamins, Dan; Cox, David D., 2013. Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms. In: Van der Walt, Stéfan; Millman, Jarrod; Huff, Katy (Hrsg.): *12th Python in science conference (SciPy)*. Austin, 24.06.2013–29.06.2013, S. 13–20

Bergstra et al. 2015 Bergstra, James; Komer, Brent; Eliasmith, Chris; Yamins, Dan; Cox, David D., 2015. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, **8** (1), S. 014008. DOI: 10.1088/1749-4699/8/1/014008

Berscheid et al. 2019 Berscheid, Lars; Rühr, Thomas; Kröger, Torsten, 2019. Improving Data Efficiency of Self-supervised Learning for Robotic Grasping. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Montreal, 20.05.2019–24.05.2019, S. 2125–2131. DOI: 10.1109/ICRA.2019.8793952

Bichler et al. 2005 Bichler, Klaus; Krohn, Ralf; Philippi, Peter (Hrsg.), 2005. *Gabler Kompakt-Lexikon Logistik*. Wiesbaden: Gabler Verlag. ISBN 978-3-409-12534-5

Blanco 2010 Blanco, Jose-Luis, 2010. *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*. Universität Malaga, Technischer Bericht. DOI: 10.48550/arXiv.2103.15980

Boehnke 2007 Boehnke, Kay, 2007. Object localization in range data for robotic bin picking. In: *3rd Annual IEEE Conference on Automation Science and Engineering (CASE)*. Scottsdale, 22.09.2007–25.09.2007, S. 572–577. DOI: 10.1109/COASE.2007.4341695

Boothroyd 2005 Boothroyd, Geoffrey, 2005. *Assembly Automation and Product Design*. 2. Aufl. Boca Raton: CRC Press. ISBN 978-1-57444-643-2

Breitenreicher et al. 2010 Breitenreicher, Dirk; Schnörr, Christoph, 2010. Robust 3D Object Registration Without Explicit Correspondence Using Geometric Integration. *Machine Vision and Applications*, **21** (5), S. 601–611. DOI: 10.1007/s00138-009-0227-6

Buchholz et al. 2010 Buchholz, Dirk; Winkelbach, Simon; Wahl, Friedrich M., 2010. RANSAM for Industrial Bin-Picking. In: *41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK)*. München, 07.06.2010–09.06.2010, S. 1–6

Buchholz et al. 2013 Buchholz, Dirk; Futterlieb, Marcus; Winkelbach, Simon; Wahl, Friedrich M., 2013. Efficient Bin-Picking and Grasp Planning Based on Depth Data. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, 06.05.2013–10.05.2013, S. 3245–3250. DOI: 10.1109/ICRA.2013.6631029

Burke et al. 2012 Burke, Edmund K.; Pham, Nam; Qu, Rong; Yellen, Jay, 2012. Linear combinations of heuristics for examination timetabling. *Annals of Operations Research*, **194** (1), S. 89–109. DOI: 10.1007/s10479-011-0854-y

Burke et al. 2013 Burke, Edmund K.; Gendreau, Michel; Hyde, Matthew; Kendall, Graham; Ochoa, Gabriela; Özcan, Ender; Qu, Rong, 2013. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, **64** (12), S. 1695–1724. DOI: 10.1057/jors.2013.71

Burkhard 2006 Burkhard, Hans-Dieter, 2006. *Einführung in die KI*. Vorlesungsskript, Winter-Semester 2006/07. Humboldt-Universität zu Berlin

Carrington et al. 2007 Carrington, Julie R.; Pham, Nam; Qu, Rong; Yellen, Jay, 2007. An Enhanced Weighted Graph Model for Examination/Course Timetabling. In: *26th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG)*. Prag, 17.12.2007–18.12.2007, S. 9–15

Chen et al. 2011 Chen, Hung-Che; Wei, Jyh-Da, 2011. Using Neural Networks for Evaluation in Heuristic Search Algorithm. In: *25th Conference on Artificial Intelligence (AAAI)*. San Francisco, 07.08.2011–11.08.2011, S. 1768–1769

Chen et al. 2018 Chen, Yu-Kai; Sun, Guo-Jhen; Lin, Huei-Yung; Chen, Shyh-Leh, 2018. Random Bin Picking with Multi-View Image Acquisition and CAD-Based Pose Estimation. In: *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Miyazaki, 07.10.2018–10.10.2018, S. 2218–2223. DOI: 10.1109/SMC.2018.00381

Claus et al. 2001 Claus, Volker; Schwill, Andreas, 2001. *Duden Informatik*. 3. Aufl. Mannheim, Leipzig, Wien, Zürich: Dudenverlag. ISBN 3-411-05233-3

Craig 2005 Craig, John J., 2005. *Introduction to Robotics: Mechanics and Control*. 3. Aufl. London: Pearson Higher Education. ISBN 978-0-13-123629-5

Denavit et al. 1955 Denavit, Jacques; Hartenberg, Richard Scheunemann, 1955. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Journal of Applied Mechanics, Transactions ASME*, **22** (2), S. 215–221

Dessimoz et al. 1984 Dessimoz, Jean-Daniel; Birk, John R.; Kelley, Robert B.; Martins, Henrique A. S.; Lin, Chi, 1984. Matched Filters for Bin Picking. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, **6** (6), S. 686–697. DOI: 10.1109/TPAMI.1984.4767593

Diankov et al. 2008 Diankov, Rosen; Kuffner, James, 2008. *OpenRAVE: A Planning Architecture for Autonomous Robotics*. Technischer Bericht Nr. CMU-RI-TR-08-34. Carnegie Mellon University, Robotics Institute. Verfügbar unter: http://ri.cmu.edu/pub_files/pub4/diankov_rosen_2008_2/diankov_rosen_2008_2.pdf. Zugriff am: 21.09.2021

Domae et al. 2014 Domae, Yukiyasu; Okuda, Haruhisa; Taguchi, Yuichi; Sumi, Kazuhiko; Hirai, Takashi, 2014. Fast Graspability Evaluation on Single Depth Maps for Bin Picking with General Grippers. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, 31.05.2014–05.06.2014, S. 1997–2004. DOI: 10.1109/ICRA.2014.6907124

Dupuis et al. 2008 Dupuis, Donna C.; Léonard, Simon; Baumann, Matthew A.; Croft, Elizabeth A.; Little, James J., 2008. Two-Fingered Grasp Planning for Randomized Bin-Picking. In: *Robotics: Science and Systems 2008 Robot Manipulation: Intelligence in Human Environments Workshop*. Zürich, 25.06.2008–28.06.2008

Effenberger et al. 2013 Effenberger, Ira; Kühnle, Jens; Verl, Alexander, 2013. Fast and flexible 3D object recognition solutions for machine vision applications. In: Bingham, Philip R.; Lam, Edmund Y. (Hrsg.): *Image Processing: Machine Vision Applications VI*. San Francisco, 05.02.2013–06.02.2013, S. 191–200. DOI: 10.1117/12.2006990

Fan et al. 2018 Fan, Yongxiang; Lin, Hsien-Chung; Tang, Te; Tomizuka, Masayoshi, 2018. Grasp Planning for Customized Grippers by Iterative Surface Fitting. In: *IEEE 14th International Conference on Automation Science and Engineering (CASE)*. München, 20.08.2018–24.08.2018, S. 28–34. DOI: 10.1109/COASE.2018.8560361

Finn et al. 2016 Finn, Chelsea; Goodfellow, Ian; Levine, Sergey, 2016. Unsupervised Learning for Physical Interaction through Video Prediction. In: Lee, Daniel D.; Sugiyama, Masashi; von Luxburg, Ulrike; Guyon, Isabelle; Garnett, Roman (Hrsg.): *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*. Barcelona, 05.12.2016–10.12.2016, S. 64–72

Freese et al. 2010 Freese, Marc; Singh, Surya; Ozaki, Fumio; Matsuhira, Nobuto, 2010. Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator. In: Ando, Noriaki; Balakirsky, Stephen; Hemker, Thomas; Reggiani, Monica; von Stryk, Oskar (Hrsg.): *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. Darmstadt, 15.11.2010–18.11.2010, S. 51–62

Fur et al. 2018 Fur, Shan; Verl, Alexander; Pott, Andreas, 2018. Simulation of Industrial Bin Picking: An Application of Laser Range Finder Simulation. In: *IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. Ottawa, 12.06.2018–13.06.2018. DOI: 10.1109/CIVEMSA.2018.8439959

Glassner 1989 Glassner, Andrew S. (Hrsg.), 1989. *An Introduction to Ray Tracing*. 1. Aufl. London: Academic Press Ltd. ISBN 978-0-12-286160-4

GPyOpt 2018 University of Sheffield, Machine Learning Group, 2018. *GPyOpt: A Bayesian Optimization framework in Python*. GitHub. Verfügbar unter: http://github.com/SheffieldML/GPyOpt. Zugriff am: 21.09.2021

Greenberg 1990 Greenberg, Harvey J., 1990. Neural networks and heuristic search. *Annals of Mathematics and Artificial Intelligence*, **1** (1), S. 75–95. DOI: 10.1007/BF01531071

Hartenberg et al. 1964 Hartenberg, Richard Scheunemann; Denavit, Jacques, 1964. *Kinematic Synthesis of Linkages*. New York: McGraw-Hill. ISBN 978-0-07-026910-1

Hart et al. 1968 Hart, Peter E.; Nilsson, Nils J.; Raphael, Bertram, 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, **4** (2), S. 100–107. DOI: 10.1109/TSSC.1968.300136

Heckner et al. 2014 Heckner, Heiko; Wirth, Marco, 2014. *Vergleich von Dateiformaten für 3D-Modelle*. Center for Digital Fabrication (CEDIFA), Lehrstuhl für Wirtschaftsinformatik und Systementwicklung, Universität Würzburg, Arbeitsbericht. Verfügbar unter: http://docplayer.org/2631312-Vergleich-von-dateiformaten-fuer-3d-modelle.html. Zugriff am: 21.09.2021

Hema et al. 2006 Hema, Chengalvarayan Radhakrishnamurthy; Paulraj, Murugesa Pandiyan; Nagarajan, Raghul; Yaacob, Sazali, 2006. Object Localization using Stereo Sensors for Adept SCARA Robot. In: *IEEE Conference on Robotics, Automation and Mechatronics*. Bangkok, 07.06.2006–09.06.2006, S. 1–5. DOI: 10.1109/RAMECH.2006.252694

Holz et al. 2014 Holz, Dirk; Nieuwenhuisen, Matthias; Droeschel, David; Stückler, Jörg; Berner, Alexander; Li, Jun; Klein, Reinhard; Behnke, Sven, 2014. Active Recognition and Manipulation for Mobile Robot Bin Picking. In: Röhrbein, Florian; Veiga, Germano; Natale, Ciro (Hrsg.): *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe*, S. 133–153. Springer Tracts in Advanced Robotics, Nr. 94. ISBN 978-3-319-03838-4

Horn et al. 1983 Horn, Berthold K. P.; Ikeuchi, Katsushi, 1983. *Picking Parts Out of a Bin*. AI Memos, Nr. 746. Massachusetts Institute of Technology, Technischer Bericht. Verfügbar unter: https://dspace.mit.edu/handle/1721.1/5642. Zugriff am: 21.09.2021

Horn et al. 1984 Horn, Berthold K. P.; Ikeuchi, Katsushi, 1984. The Mechanical Manipulation of Randomly Oriented Parts. *Scientific American*, **251** (2), S. 100–113

Hottung et al. 2020 Hottung, André; Tanaka, Shunji; Tierney, Kevin, 2020. Deep Learning Assisted Heuristic Tree Search for the Container Pre-marshalling Problem. *Computers & Operations Research*, **113**. DOI: 10.1016/j.cor.2019.104781

IDS 2018 IDS, Imaging Development Systems GmbH, 2018. *Spezifikation N20 Serie (Optik N20-1202-16-BL)*. Verfügbar unter: https://de.ids-imaging.com/ensensoprodukt.html?modelName=N20-1202-16-BL. Zugriff am: 21.09.2021

Igel et al. 2000 Igel, Christian; Hüsken, Michael, 2000. Improving the Rprop Learning Algorithm. In: *2nd International ICSC Symposium on Neural Computation (NC)*. Berlin, 23.05.2000–26.05.2000, S. 115–121

Ikeuchi et al. 1983 Ikeuchi, Katsushi; Horn, Berthold K. P.; Nagata, Shigemi; Callahan, Tom; Feingold, Oded, 1983. *Picking up an object from a pile of objects*. AI Memos, Nr. 726. Massachusetts Institute of Technology, Technischer Bericht. Verfügbar unter: `https://dspace.mit.edu/handle/1721.1/5651`. Zugriff am: 21.09.2021

Jiang et al. 2011 Jiang, Yun; Moseson, Stephen; Saxena, Ashutosh, 2011. Efficient Grasping from RGBD Images: Learning using a new Rectangle Representation. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, 09.05.2011–13.05.2011, S. 3304–3311. DOI: `10.1109/ICRA.2011.5980145`

Johns et al. 2016 Johns, Edward; Leutenegger, Stefan; Davison, Andrew J., 2016. Deep Learning a Grasp Function for Grasping under Gripper Pose Uncertainty. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, 09.10.2016–14.10.2016, S. 4461–4468. DOI: `10.1109/IROS.2016.7759657`

Jordanov et al. 2007 Jordanov, Ivan; Georgieva, Antoniya, 2007. Neural Network Learning With Global Heuristic Search. *IEEE Transactions on Neural Networks*, **18** (3), S. 937–942. DOI: `10.1109/TNN.2007.891633`

Jørgensen et al. 2008 Jørgensen, Jimmy Alison; Petersen, Henrik Gordon, 2008. Usage of simulations to plan stable grasping of unknown objects with a 3-fingered Schunk hand. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nizza, 22.09.2008–26.09.2008

Kaiser et al. 2009 Kaiser, Benedikt; Tauro, Ricardo A.; Wörn, Heinz, 2009. Automatic and Semi-automatic Unloading of Containers. In: *IASTED International Conference on Modelling, Simulation, and Identification (MSI)*. Peking, 12.10.2009–14.10.2009, S. 857–863

Kaiser 2010 Kaiser, Benedikt, 2010. *Bildverarbeitung für ein generisches Entladesystem*. Karlsruhe, Techn. Univ., Diss., 2009. DOI: `10.5445/IR/1000015610`

Khalid et al. 2021 Khalid, Muhammad Usman; Spenrath, Felix; Mönnig, Manuel; Moosmann, Marius; Bormann, Richard; Kunz, Holger; Huber, Marco F., 2021. Automatic Grasp Generation for Vacuum Grippers for Random Bin Picking. In: Weißgraeber, Philipp; Heieck, Frieder; Ackermann, Clemens (Hrsg.): *Advances in Automotive Production Technology – Theory and Application*, S. 247–255

Kleeberger et al. 2019 Kleeberger, Kilian; Landgraf, Christian; Huber, Marco F., 2019. Large-scale 6D Object Pose Estimation Dataset for Industrial Bin-Picking. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, 03.11.2019–08.11.2019, S. 2573–2578. DOI: 10.1109/IROS40897.2019.8967594

Kleeberger et al. 2020 Kleeberger, Kilian; Huber, Marco F., 2020. Single Shot 6D Object Pose Estimation. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Paris, 31.05.2020–31.08.2020, S. 6239–6245. DOI: 10.1109/ICRA40945.2020.9197207

Kleeberger et al. 2021 Kleeberger, Kilian; Roth, Florian; Bormann, Richard; Huber, Marco F., 2021. Automatic Grasp Pose Generation for Parallel Jaw Grippers. *CoRR*, **abs/2104.11660**. Verfügbar unter: arXiv: 2104.11660

Kramer 2009 Kramer, Oliver, 2009. *Computational Intelligence: Eine Einführung*. Berlin, Heidelberg: Springer-Verlag. Informatik im Fokus. ISBN 978-3-540-79738-8

Kriesel 2007 Kriesel, David, 2007. *Ein kleiner Überblick über Neuronale Netze* [online]. Verfügbar unter: eprint: http://www.dkriesel.com/science/neural_networks. Zugriff am: 21.09.2021

Kuo et al. 2014 Kuo, Hao-Yuan; Su, Hong-Ren; Lai, Shang-Hong; Wu, Chin-Chia, 2014. 3D Object Detection and Pose Estimation from Depth Image for Robotic Bin Picking. In: *IEEE International Conference on Automation Science and Engineering (CASE)*. Taipei, 18.08.2014–22.08.2014, S. 1264–1269. DOI: 10.1109/CoASE.2014.6899489

Ledermann 2012 Ledermann, Thomas, 2012. *Partikel-Schwarm-Optimierung zur Objektlageerkennung in Tiefendaten*. Heimsheim: Jost Jetter Verlag. IPA-IAO-Forschung und Praxis, Nr. 523. Stuttgart, Univ., Diss., 2011. ISBN 978-3-939890-97-3

Leonard et al. 2007 Leonard, Simon; Chan, Ambrose; Croft, Elizabeth; Little, James J., 2007. Robust Motion Generation for Vision-Guided Robot Bin-Picking. In: *ASME International Mechanical Engineering Congress and Exposition (IMECE)*. Seattle, 11.11.2007–15.11.2007, S. 651–658. DOI: 10.1115/IMECE2007-42606

Levine et al. 2018 Levine, Sergey; Pastor, Peter; Krizhevsky, Alex; Ibarz, Julian; Quillen, Deirdre, 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, **37** (4-5), S. 421–436. DOI: 10.1177/0278364917710318

Liebherr 2013 Liebherr-Verzahntechnik GmbH, 2013. *Intelligente Werkstückhandhabung: Liebherr-Robotersysteme für den „Griff in die Kiste“*. Verfügbar unter: https://www.liebherr.com/de/che/aktuelles/news-pressemitteilungen/detail/intelligente-werkst%C3%BCckhandhabung-liebherr-robotersysteme-f%C3%BCr-den-%E2%80%9Egriff-in-die-kiste%E2%80%9D.html. Zugriff am: 21.09.2021

Lunze 2016 Lunze, Jan, 2016. *Künstliche Intelligenz für Ingenieure: Methoden zur Lösung ingenieurtechnischer Probleme mit Hilfe von Regeln, logischen Formeln und Bayesnetzen*. 3. Aufl. Berlin: De Gruyter Oldenbourg. ISBN 978-3-11-044896-2

Mahler et al. 2016 Mahler, Jeffrey; Pokorny, Florian T.; Hou, Brian; Roderick, Melrose; Laskey, Michael; Aubry, Mathieu; Kohlhoff, Kai; Kröger, Torsten; Kuffner, James; Goldberg, Ken, 2016. A Cloud-Based Network of 3D Objects for Robust Grasp Planning Using a Multi-Armed Bandit Model with Correlated Rewards. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, 16.05.2016–21.05.2016, S. 1957–1964. DOI: 10.1109/ICRA.2016.7487342

Mahler et al. 2017 Mahler, Jeffrey; Liang, Jacky; Niyaz, Sherdil; Laskey, Michael; Doan, Richard; Liu, Xinyu; Ojea, Juan Aparicio; Goldberg, Ken, 2017. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. In: *Robotics: Science and Systems (RSS)*. Cambridge, 12.07.2017–16.07.2017

Mahler et al. 2018 Mahler, Jeffrey; Matl, Matthew; Liu, Xinyu; Li, Albert; Gealy, David; Goldberg, Ken, 2018. Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, 21.05.2018–26.05.2018, S. 1–8. DOI: 10.1109/ICRA.2018.8460887

Martinez et al. 2015 Martinez, Carlos; Boca, Remus; Zhang, Biao; Chen, Heping; Nidamarthi, Srinivas, 2015. Automated bin picking system for randomly located industrial parts. In: *IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. Boston, 11.05.2015–12.05.2015, S. 1–6. DOI: 10.1109/TePRA.2015.7219656

McCulloch et al. 1943 McCulloch, Warren S.; Pitts, Walter, 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, **5** (4), S. 115–133. DOI: 10.1007/BF02478259

Melax 1998 Melax, Stan, 1998. A Simple, Fast, and Effective Polygon Reduction Algorithm. *Game Developer*, **11**, S. 44–49

Miller et al. 2000 Miller, Andrew T.; Allen, Peter K., 2000. GraspIt!: A Versatile Simulator for Grasp Analysis. In: *ASME Dynamic Systems and Control Division*. Orlando, 05.11.2000–10.11.2000, S. 1251–1258

Miller et al. 2004 Miller, Andrew T.; Allen, Peter K., 2004. GraspIt! A Versatile Simulator for Robotic Grasping. *IEEE Robotics & Automation Magazine*, **11** (4), S. 110–122. DOI: 10.1109/MRA.2004.1371616

Mitchell 1997 Mitchell, Tom M., 1997. *Machine Learning*. 1. Aufl. New York: McGraw-Hill. ISBN 978-0-07-042807-2

Mitchell 1998 Mitchell, Melanie, 1998. *An Introduction to Genetic Algorithms*. Cambridge: MIT Press. A Bradford Book. ISBN 978-0-262-63185-3

Mockus 1989 Mockus, Jonas, 1989. *Bayesian Approach to Global Optimization: Theory and Applications*. 1. Aufl. Dordrecht: Springer. Mathematics and its Applications, Nr. 37. ISBN 978-94-010-6898-7

Müller et al. 2020 Müller, Christopher; Kutzbach, Nina, 2020. *World Robotics 2020 - Industrial Robots*. Frankfurt am Main: IFR Statistical Department, VDMA Services GmbH. ISBN 978-3-8163-0739-6

Nieuwenhuisen et al. 2012 Nieuwenhuisen, Matthias; Stückler, Jörg; Berner, Alexander; Klein, Reinhard; Behnke, Sven, 2012. Shape-Primitive Based Object Recognition and Grasping. In: *7th German Conference on Robotics (ROBOTIK)*. München, 21.05.2012–22.05.2012, S. 1–5

Nieuwenhuisen et al. 2013 Nieuwenhuisen, Matthias; Droeschel, David; Holz, Dirk; Stückler, Jörg; Berner, Alexander; Li, Jun; Klein, Reinhard; Behnke, Sven, 2013. Mobile Bin Picking with an Anthropomorphic Service Robot. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, 06.05.2013–10.05.2013, S. 2327–2334. DOI: `10.1109/ICRA.2013.6630892`

Nissen et al. 2003 Nissen, Steffen et al., 2003. Implementation of a Fast Artificial Neural Network Library (fann). *Bericht, Department of Computer Science, University of Copenhagen*, **31**, S. 29

Nissen 2007 Nissen, Steffen, 2007. Large Scale Reinforcement Learning using Q-SARSA(λ) and Cascading Neural Networks. *Masterarbeit, Department of Computer Science, University of Copenhagen*. Verfügbar unter: `http://leenissen.dk/fann/report.pdf`. Zugriff am: 21.09.2021

Norm DIN 580 DIN 580:2018-04. *Ringschrauben*

Norm ISO 8373 ISO 8373:2012-03. *Robots and robotic devices*

Oh et al. 2010 Oh, Jong-Kyu; Lee, Chan-Ho; Lee, Sang-Hun; Jung, Sung-Hyun; Kim, Dasik; Lee, Sukhan, 2010. Development of a Structured-light Sensor Based Bin-Picking System Using ICP Algorithm. In: *International Conference on Control, Automation and Systems (ICCAS)*. Gyeonggi-do, 27.10.2010–30.10.2010, S. 1673–1677. DOI: 10.1109/ICCAS.2010.5669661

Ohba et al. 1996 Ohba, Kohtaro; Ikeuchi, Katsushi, 1996. Recognition of the Multi Specularity Objects for Bin-Picking Task. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Osaka, 04.11.1996–08.11.1996, S. 1440–1447. DOI: 10.1109/IROS.1996.569004

Opitz et al. 1995 Opitz, David W.; Shavlik, Jude W., 1995. Using Heuristic Search to Expand Knowledge-Based Neural Networks. In: Petsche, Thomas; Hanson, Stephen José; Shavlik, Jude (Hrsg.): *Computational Learning Theory and Natural Learning Systems*. MIT Press, Bd. 3, S. 3–19. ISBN 978-0-262-66096-9

Palzkill et al. 2010 Palzkill, Matthias; Ledermann, Thomas; Verl, Alexander, 2010. Anticipation-Preprocessing for Object Pose Detection. In: *41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK)*. München, 07.06.2010–09.06.2010, S. 1–6

Palzkill et al. 2012 Palzkill, Matthias; Verl, Alexander, 2012. Object Pose Detection in Industrial Environment. In: *7th German Conference on Robotics (ROBOTIK)*. München, 21.05.2012–22.05.2012, S. 1–5

Palzkill 2014 Palzkill, Matthias, 2014. *Heuristisches Suchverfahren zur Objektlageerkennung aus Punktewolken für industrielle Zuführsysteme*. Stuttgart: Fraunhofer Verlag. Stuttgarter Beiträge zur Produktionsforschung, Nr. 37. Stuttgart, Univ., Diss., 2014. ISBN 978-3-8396-0784-8

Papazov et al. 2012 Papazov, Chavdar; Haddadin, Sami; Parusel, Sven; Krieger, Kai; Burschka, Darius, 2012. Rigid 3D geometry matching for grasping of known objects in cluttered scenes. *The International Journal of Robotics Research*, **31** (4), S. 538–553. DOI: 10.1177/0278364911436019

Park et al. 2010 Park, In Kyu; Germann, Marcel; Breitenstein, Michael D.; Pfister, Hanspeter, 2010. Fast and automatic object pose estimation for range images on the GPU. *Machine Vision and Applications*, **21** (5), S. 749–766. DOI: 10.1007/s00138-009-0209-8

Paul et al. 1986 Paul, Richard P.; Zhang, Hong, 1986. Computationally Efficient Kinematics for Manipulators with Spherical Wrists Based on the Homogeneous Transformation Representation. *The International Journal of Robotics Research*, **5** (2), S. 32–44. DOI: 10.1177/027836498600500204

Pauli 1998 Pauli, Josef, 1998. Learning to Recognize and Grasp Objects. *Autonomous Robots*, **5** (3), S. 407–420. DOI: 10.1023/A:1008874725911

Pearson 1896 Pearson, Karl, 1896. Mathematical Contributions to the Theory of Evolution. – III. Regression, Heredity, and Panmixia. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, **187**, S. 253–318. DOI: 10.1098/rsta.1896.0007

Pinto et al. 2016 Pinto, Lerrel; Gupta, Abhinav, 2016. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, 16.05.2016–21.05.2016, S. 3406–3413. DOI: 10.1109/ICRA.2016.7487517

Pott et al. 2019 Pott, Andreas; Dietz, Thomas, 2019. *Industrielle Robotersysteme*. 1. Aufl. Wiesbaden: Springer Vieweg. ISBN 978-3-658-25344-8

Rahardja et al. 1996 Rahardja, Krisnawan; Kosaka, Akio, 1996. Vision-Based Bin-Picking: Recognition and Localization of Multiple Complex Objects Using Simple Visual Cues. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Osaka, 04.11.1996–08.11.1996, S. 1448–1457. DOI: 10.1109/IROS.1996.569005

Rasmussen et al. 2006 Rasmussen, Carl Edward; Williams, Christopher K. I., 2006. *Gaussian Processes for Machine Learning*. Cambridge: MIT Press. Adaptive computation and machine learning. ISBN 978-0-262-18253-9

Redmon et al. 2015 Redmon, Joseph; Angelova, Anelia, 2015. Real-Time Grasp Detection Using Convolutional Neural Networks. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, 26.05.2015–30.05.2015, S. 1316–1322. DOI: 10.1109/ICRA.2015.7139361

Riedmiller et al. 1993 Riedmiller, Martin; Braun, Heinrich, 1993. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: *IEEE International Conference on Neural Networks*. Nagoya, 25.10.1993–29.10.1993, S. 586–591

Rohmer et al. 2013 Rohmer, Eric; Singh, Surya P. N.; Freese, Marc, 2013. V-REP: a Versatile and Scalable Robot Simulation Framework. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, 03.11.2013–07.11.2013, S. 1321–1326. DOI: 10.1109/IROS.2013.6696520

Rosenblatt 1961 Rosenblatt, Frank, 1961. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Technischer Bericht Nr. 1196-G-8. Cornell Aeronautical Laboratory, Cornell University, Washington DC

Rothlauf 2011 Rothlauf, Franz, 2011. *Design of Modern Heuristics: Principles and Application*. 1. Aufl. Berlin, Heidelberg: Springer-Verlag. Natural Computing Series. ISBN 978-3-540-72961-7

Roy et al. 2016 Roy, Mayank; Boby, Riby Abraham; Chaudhary, Shraddha; Chaudhury, Santanu; Roy, Sumantra Dutta; Saha, Subir Kumar, 2016. Pose Estimation of Texture-less Cylindrical Objects in Bin Picking using Sensor Fusion. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, 09.10.2016–14.10.2016, S. 2279–2284. DOI: 10.1109/IROS.2016.7759356

Rumelhart et al. 1985 Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J., 1985. *Learning Internal Representations by Error Propagation*. ICS report 8506. Institute for Cognitive Science, University of California, San Diego

Rumelhart et al. 1986a Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J., 1986. Learning Internal Representations by Error Propagation. In: Rumelhart, David E.; McClelland, James L. (Hrsg.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. Cambridge: MIT Press, Bd. 1, S. 318–362. A Bradford Book. ISBN 978-0-262-18120-4

Rumelhart et al. 1986b Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J., 1986. Learning representations by back-propagating errors. *Nature*, **323** (6088), S. 533. DOI: 10.1038/323533a0

Schraft et al. 2003 Schraft, Rolf Dieter; Ledermann, Thomas, 2003. Intelligent picking of chaotically stored objects. *Assembly Automation*, **23** (1), S. 38–42. DOI: 10.1108/01445150310460079

Schyja et al. 2012 Schyja, Adrian; Hypki, Alfred; Kuhlenkötter, Bernd, 2012. A Modular and Extensible Framework for Real and Virtual Bin-Picking Environments. In: *IEEE International Conference on Robotics and Automation (ICRA)*. St. Paul, 14.05.2012–18.05.2012, S. 5246–5251. DOI: 10.1109/ICRA.2012.6224875

Schyja et al. 2014 Schyja, Adrian; Kuhlenkötter, Bernd, 2014. Virtual Bin Picking - A Generic Framework to Overcome the Bin Picking Complexity by the Use of a Virtual Environment. In: *4th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*. Wien, 28.08.2014–30.08.2014, S. 133–140. DOI: 10.5220/0005011401330140

Schyja et al. 2015 Schyja, Adrian; Kuhlenkötter, Bernd, 2015. Realistic Simulation of Industrial Bin-Picking Systems. In: *6th International Conference on Automation, Robotics and Applications (ICARA)*. Queenstown, 17.02.2015–19.02.2015, S. 137–142. DOI: 10.1109/ICARA.2015.7081137

Shao et al. 2019 Shao, Quanquan; Hu, Jie; Wang, Weiming; Fang, Yi; Liu, Wenhai; Qi, Jin; Ma, Jin, 2019. Suction Grasp Region Prediction Using Self-supervised Learning for Object Picking in Dense Clutter. In: *IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*. Singapur, 03.05.2019–05.05.2019

Shepperd 1978 Shepperd, Stanley W., 1978. Quaternion from Rotation Matrix. *Journal of Guidance and Control*, **1** (3), S. 223–224

Siciliano et al. 2008 Siciliano, Bruno; Khatib, Oussama (Hrsg.), 2008. *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-540-23957-4

Siciliano et al. 2009 Siciliano, Bruno; Sciavicco, Lorenzo; Villani, Luigi; Oriolo, Giuseppe, 2009. *Robotics - Modelling, Planning and Control*. 1. Aufl. London: Springer-Verlag. ISBN 978-1-84628-641-4

SICK 2017 SICK AG, 2017. *Betriebsanleitung LMS400 Lasermesssensoren* [online]. Verfügbar unter: `https://cdn.sick.com/media/docs/7/97/697/operating_instructions_lms400_laser_measurement_sensors_de_im0010697.pdf`. Zugriff am: 21.09.2021

Silver et al. 2016 Silver, David; Huang, Aja; Maddison, Christopher; Guez, Arthur; Sifre, Laurent; van den Driessche, George; Schrittwieser, Julian; Antonoglou, Ioannis; Panneershelvam, Veda; Lanctot, Marc; Dieleman, Sander; Grewe, Dominik; Nham, John; Kalchbrenner, Nal; Sutskever, Ilya; Lillicrap, Timothy; Leach, Madeleine; Kavukcuoglu, Koray; Graepel, Thore; Hassabis, Demis, 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, **529**, S. 484–489. DOI: `10.1038/nature16961`

Smith et al. 2007 Smith, Russell et al., 2007. *Open Dynamics Engine*. Verfügbar unter: `https://bitbucket.org/odedevs/ode`. Zugriff am: 21.09.2021

Spenrath et al. 2012 Spenrath, Felix; Spiller, Alexander; Verl, Alexander, 2012. Gripping Point Determination and Collision Prevention in a Bin-Picking application. In: *IEEE 7th German Conference on Robotics (ROBOTIK 2012)*. München, 21.05.2012–22.05.2012, S. 1–6

Spenrath et al. 2013 Spenrath, Felix; Palzkill, Matthias; Pott, Andreas; Verl, Alexander, 2013. Object Recognition: Bin-Picking For Industrial Use. In: *IEEE 44th International Symposium on Robotics (ISR)*. Seoul, 24.10.2013–26.10.2013, S. 1–3. DOI: 10.1109/ISR.2013.6695743

Spenrath et al. 2017a Spenrath, Felix; Pott, Andreas, 2017. Gripping point determination for bin picking using heuristic search. *Procedia CIRP*, **62**, S. 606–611. DOI: 10.1016/j.procir.2016.06.015

Spenrath et al. 2017b Spenrath, Felix; Pott, Andreas, 2017. Statistical Analysis of Influencing Factors for Heuristic Grip Determination in Random Bin Picking. In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. München, 03.07.2017–07.07.2017, S. 868–873. DOI: 10.1109/AIM.2017.8014127

Spenrath et al. 2018 Spenrath, Felix; Pott, Andreas, 2018. Using Neural Networks for Heuristic Grasp Planning in Random Bin Picking. In: *IEEE 14th International Conference on Automation Science and Engineering (CASE)*. München, 20.08.2018–24.08.2018, S. 258–263. DOI: 10.1109/COASE.2018.8560458

Takenouchi et al. 1998 Takenouchi, Ayako; Kanamaru, Naoyoshi; Mizukawa, Makoto, 1998. Hough-space-based Object Recognition Tightly Coupled with Path Planning for Robust and Fast Bin-picking. In: *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*. Victoria, 13.10.1998–17.10.1998, 1222–1229 vol.2. DOI: 10.1109/IROS.1998.727466

Tauro et al. 2010 Tauro, Ricardo A.; Kaiser, Benedikt; Wörn, Heinz, 2010. Path Planning Process Optimization for a Bin Picking System. In: *41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK)*. München, 07.06.2010–07.06.2010, S. 1–7

Trobina et al. 1994 Trobina, Marjan; Leonardis, Aleš; Ade, Frank, 1994. Grasping Arbitrarily Shaped Objects. In: Kropatsch, Walter G.; Bischof, Horst (Hrsg.): *Mustererkennung 1994, Erkennen und Lernen, 16. DAGM-Symposium und 18. Workshop der ÖAGM*. Wien, 21.09.1994–23.09.1994, S. 126–134

Trobina et al. 1995 Trobina, Marjan; Leonardis, Aleš, 1995. Grasping Arbitrarily Shaped 3-D Objects from a Pile. In: *IEEE International Conference on Robotics and Automation*. Nagoya, 21.05.1995–27.05.1995, S. 241–246

Trobina 1995 Trobina, Marjan, 1995. *From Planar Patches to Grasps: A 3-D Robot Vision System Handling Unmodeled Objects*. Konstanz: Hartung-Gorre-Verlag. Zürich, Eidgenöss. Techn. Hochsch., Diss., 1995. ISBN 3-89191-973-5

Umbarkar et al. 2015 Umbarkar, Anantkumar J.; Sheth, Pranali D., 2015. Crossover Operators in Genetic Algorithms: A Review. *ICTACT Journal on Soft Computing*, **6** (1). DOI: 10.21917/ijsc.2015.0150

VDI 3423 2011 Verein Deutscher Ingenieure (VDI), 2011. *Verfügbarkeit von Maschinen und Anlagen*. VDI-Richtlinie 3423

Wang et al. 1994 Wang, Sarah; Cromwell, Robert; Kak, Avi; Kimura, Ichiro; Osada, Michiharu, 1994. Model-Based Vision for Robotic Manipulation of Twisted Tubular Parts: Using Affine Transforms and Heuristic Search. In: *IEEE International Conference on Robotics and Automation (ICRA)*. San Diego, 08.05.1994–13.05.1994, S. 208–215. DOI: 10.1109/ROBOT.1994.350986

Weicker 2007 Weicker, Karsten, 2007. *Evolutionäre Algorithmen*. 2. Aufl. Wiesbaden: Vieweg+Teubner Verlag. ISBN 978-3-8351-0219-4

Werbos 1982 Werbos, Paul J., 1982. Applications of Advances in Nonlinear Sensitivity Analysis. In: Drenick, R. F.; Kozin, F. (Hrsg.): *10th IFIP Conference on System Modeling and Optimization*. New York, 31.08.1981–04.09.1981, S. 762–770

Winston 1984 Winston, Patrick Henry, 1984. *Artificial Intelligence*. 2. Aufl. Reading, Menlo Park, London, Amsterdam, Don Mills, Sydney: Addison-Wesley. Addison-Wesley series in computer science. ISBN 978-0-201-08259-3

Wnuk et al. 2017 Wnuk, Markus; Pott, Andreas; Xu, Weilang; Lechler, Armin; Verl, Alexander, 2017. Concept for a Simulation-based Approach Towards Automated Handling of Deformable Objects — A Bin Picking Scenario. In: *24th IEEE International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. Auckland, 21.11.2017–23.11.2017, S. 1–6

Woo et al. 1999 Woo, Mason; Neider, Jackie; Davis, Tom; Shreiner, Dave, 1999. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. 3. Aufl. Boston: Addison-Wesley. ISBN 978-0-201-60458-0

Yanagihara et al. 1991 Yanagihara, Yoshimasa; Kita, Toshiro, 1991. Parts-picking in Disordered Environment. In: *IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS)*. Osaka, 03.11.1991–05.11.1991, S. 517–522. DOI: 10.1109/IROS.1991.174524

Yang 2018 Yang, Xin-She, 2018. *Optimization Techniques and Applications with Examples*. Hoboken: John Wiley & Sons. ISBN 978-1-119-49054-8

You et al. 2018 You, Fang; Mende, Michael; Štogl, Denis; Hein, Björn; Kröger, Torsten, 2018. Model-Free Grasp Planning for Configurable Vacuum Grippers. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, 01.10.2018–05.10.2018, S. 4554–4561

Zanuttigh et al. 2016 Zanuttigh, Pietro; Marin, Giulio; Dal Mutto, Carlo; Dominio, Fabio; Minto, Ludovico; Cortelazzo, Guido Maria, 2016. *Time-of-Flight and Structured Light Depth Cameras: Technology and Applications*. Schweiz: Springer. ISBN 978-3-319-30973-6

Zhou et al. 2019 Zhou, Yi; Barnes, Connelly; Lu, Jingwan; Yang, Jimei; Li, Hao, 2019. On the Continuity of Rotation Representations in Neural Networks. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, 15.06.2019–20.06.2019, S. 5745–5753

A Anhang

A.1 Modell des Sensorrauschens für die Simulation

Abbildung A.1 vergleicht die Standardabweichung des in der Simulation verwendeten, linearen Rauschmodells mit den in der Spezifikation des Ensenso N20-1202-16-BL angegebenen Genauigkeiten. Die Genauigkeit eines Messpunkts ist abhängig von seiner Entfernung zum Sensor, wobei die Standardabweichung mit zunehmendem Abstand größer wird. Die einzelnen in der Spezifikation des Sensors angegeben Z-Genauigkeiten für verschiedene Abstände sind in der Abbildung durch Punkte dargestellt, das verwendete Modell des Sensorrauschens durch eine rote Linie. Dabei ist zu sehen, dass das Modell die angegebenen Genauigkeiten sehr genau approximiert.

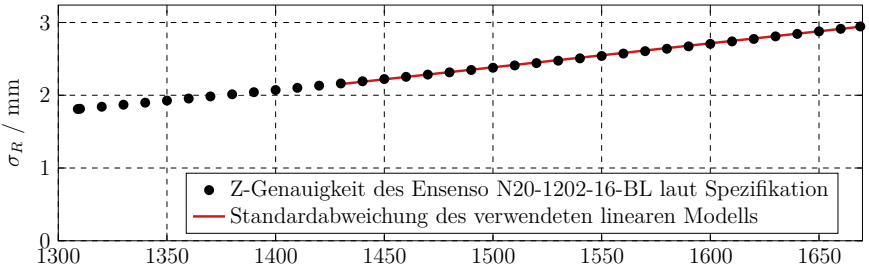


Abbildung A.1: Modell des Sensorrauschens für die Simulation mit Ringschrauben.

A.2 Versuchsparameter

Die Tabellen A.1 bis A.3 zeigen die Parameter für die Versuche, bei denen keine Heuristikfunktion verwendet wird. Im Tabellenabschnitt *Situation* ist der Typ des Werkstücks spezifiziert, sowie die Anzahl N_S der untersuchten Situationen und die Anzahl N_W^L der im Ladungsträger enthaltenen Werkstücke. Der Tabellenabschnitt *Suchbaum* beinhaltet die Parameter, die bei der Generierung des Suchbaums relevant sind. In den Tabellen A.4 bis A.7 werden im Tabellenabschnitt *Heuristik* zusätzlich die Parameter der Heuristikfunktion spezifiziert. Zur Erklärung der einzelnen Parameter sei auf den jeweiligen Abschnitt des Versuchs bzw. das Symbolverzeichnis verwiesen.

Tabelle A.1: Versuchsparameter aus Abschnitt 3.4.3

Parametersatz	1.1	1.2	Einheit
Situation:			
Werkstück	Getriebewelle	Ringschraube	
N_S	200	200	
N_W^L	25/50/75	10/20/30/40	
Suchbaum:			
N_W	variabel	variabel	
N_{dof}^{max}	12	12	
β_{max}	5	5	deg
γ_{max}	45	45	deg
ξ_{singu}	–	–	deg
δ_{min}	100	100	mm
δ_{max}	100	100	mm
g_A	1	1	
N_T	7	7	
t_{max}	–	–	s
Voll expandiert	Ja	Ja	

Tabelle A.2: Versuchsparameter aus Abschnitt 4.6.1

Parametersatz	2.1	2.2	Einheit
Situation:			
Werkstück	Getriebewelle	Ringschraube	
N_S	1875	1000	
N_W^L	1 - 75	1 - 40	
Suchbaum:			
N_W	25	20	
N_{dof}^{max}	12	12	
β_{max}	5	5	deg
γ_{max}	45	45	deg
ξ_{singu}	–	–	deg
δ_{min}	100	100	mm
δ_{max}	100	100	mm
g_A	1	1	
N_T	1	1	
t_{max}	–	–	s
Voll expandiert	im ersten Durchlauf	im ersten Durchlauf	

Tabelle A.3: Versuchsparameter aus Abschnitt 4.6.2

Parametersatz	3.1	3.2	Einheit
Situation:			
Werkstück	Getriebewelle	Ringschraube	
N_S	3750	4000	
N_W^L	1 - 75	1 - 40	
Suchbaum:			
N_W	25	20	
N_{dof}^{max}	12	12	
β_{max}	5	5	deg
γ_{max}	45	45	deg
ξ_{singu}	–	–	deg
δ_{min}	100	100	mm
δ_{max}	100	100	mm
g_A	1	1	
N_T	7	7	
t_{max}	–	–	s
Voll expandiert	Ja	Ja	

Tabelle A.4: Versuchsparameter aus Abschnitt 4.6.7

Parametersatz	4.1	4.2	Einheit
Situation:			
Werkstück	Getriebewelle	Ringschraube	
N_S	1500	1000	
N_W^L	1 - 75	1 - 40	
Suchbaum:			
N_W	25	25	
N_{dof}^{max}	12	12	
β_{max}	5	5	deg
γ_{max}	45	45	deg
ξ_{singu}	–	–	deg
δ_{min}	100	100	mm
δ_{max}	100	100	mm
g_A	5	5	
N_T	1	1	
t_{max}	–	–	s
Voll expandiert	für Training	für Training	
Heuristik:			
d_{max}	20	20	
w_d	0	0	
N_d^{min}	10	10	
w_W	0,5	0,5	
w_G	0,5	0,5	
w_T	0,5	0,5	
w_V	0,5	0,5	
w_{exp}	0	0	
w_{nn}^W	1	1	
w_{nn}^G	1	1	
w_{nn}^T	1	1	
w_{nn}^V	1	1	
w_{dof}	0	0	
w_l	0	0	
w_z	0	0	
c_z	$c_z^{max_1}$	$c_z^{max_1}$	
c_{xy}	c_{xy}^{wand}	c_{xy}^{wand}	

Tabelle A.5: Versuchsparameter aus Abschnitt 4.6.8

Parametersatz	5.1	5.2	Einheit
Situation:			
Werkstück	Getriebewelle	Ringschraube	
N_S	750	400	
N_W^L	1 - 75	1 - 40	
Suchbaum:			
N_W	25	25	
N_{dof}^{max}	12	12	
β_{max}	5	5	deg
γ_{max}	45	45	deg
ξ_{singu}	–	–	deg
δ_{min}	100	100	mm
δ_{max}	100	100	mm
g_A	5	5	
N_T	1	1	
t_{max}	–	–	s
Voll expandiert	für Training	für Training	
Heuristik:			
d_{max}	20	20	
w_d	variabel	variabel	
N_d^{min}	10	10	
w_W	variabel	variabel	
w_G	variabel	variabel	
w_T	variabel	variabel	
w_V	variabel	variabel	
w_{exp}	0	0	
w_{nn}^W	0 / 1 / variabel	0 / 1 / variabel	
w_{nn}^G	0 / 1 / variabel	0 / 1 / variabel	
w_{nn}^T	0 / 1 / variabel	0 / 1 / variabel	
w_{nn}^V	0 / 1 / variabel	0 / 1 / variabel	
w_{dof}	variabel	variabel	
w_l	0	0	
w_z	variabel	variabel	
c_z	$c_z^{max_1}$	$c_z^{max_1}$	
c_{xy}	c_{xy}^{wand}	c_{xy}^{wand}	

Tabelle A.6: Versuchsparameter aus Abschnitt 5.1.4

Parametersatz	6.1	6.2	Einheit
Situation:			
Werkstück	Getriebewelle	Ringschraube	
N_S	340	400	
N_W^L	variabel	variabel	
Suchbaum:			
N_W	25	25	
N_{dof}^{max}	12	12	
β_{max}	5	5	deg
γ_{max}	45	45	deg
ξ_{singu}	–	–	deg
δ_{min}	100	100	mm
δ_{max}	100	100	mm
g_A	5	5	
N_T	1	1	
t_{max}	–	–	s
Voll expandiert	für Training	für Training	
Heuristik:			
d_{max}	20	20	
w_d	variabel	variabel	
N_d^{min}	10	10	
w_W	variabel	variabel	
w_G	variabel	variabel	
w_T	variabel	variabel	
w_V	variabel	variabel	
w_{exp}	0	0	
w_{nn}^W	0 / 1 / variabel	0 / 1 / variabel	
w_{nn}^G	0 / 1 / variabel	0 / 1 / variabel	
w_{nn}^T	0 / 1 / variabel	0 / 1 / variabel	
w_{nn}^V	0 / 1 / variabel	0 / 1 / variabel	
w_{dof}	variabel	variabel	
w_l	0	0	
w_z	variabel	variabel	
c_z	$c_z^{max_1}$	$c_z^{max_1}$	
c_{xy}	c_{xy}^{wand}	c_{xy}^{wand}	

Tabelle A.7: Versuchsparameter aus Abschnitt 5.2

Parametersatz	7.1	Einheit
Situation:		
Werkstück	Pleuelstangen	
N_S	variabel	
N_W^L	variabel	
Suchbaum:		
N_W	–	
N_{dof}^{max}	12	
β_{max}	5	deg
γ_{max}	45	deg
ξ_{singu}	10	deg
δ_{min}	20	mm
δ_{max}	100	mm
g_A	3	
N_T	11	
t_{max}	10	s
Voll expandiert	Nein	
Heuristik:		
d_{max}	20	
w_d	0,0	
N_d^{min}	–	
w_W	0,192	
w_G	0,192	
w_T	0,192	
w_V	0,192	
w_{exp}	0,192	
w_{nn}^W	0,0	
w_{nn}^G	0,0	
w_{nn}^T	0,0	
w_{nn}^V	0,0	
w_{dof}	0,01	
w_l	0,5	
w_z	0,5	
c_z	$c_z^{max_1}$	
c_{xy}	c_{xy}^{wand}