

■ fakultät für informatik

Endbericht

Projektgruppe 575: DynOLog

3. Juni 2014

Teilnehmende:

Ella Albrecht,
Timo Bleuel,
Felix Finkeldey,
Sarah Gillet,
Markus Kloß,
Tim Krämer,
Jan Eric Lenssen,
Samir Mahmalat,
David Reher,
Daniel Schaefers,
Romano Schulz,
Florian Will

Betreuer:

Dr. Frank Weichert,
Dipl.-Inform. Pascal Libuschewski,
Dipl.-Inform. Adrian Schyja,
Dr.-Ing. Andreas Kamagaew,
Dipl.-Inform. Bartholomäus Rudak,
Dipl.-Logist. Christian Prasse

Lehrstuhl Informatik VII
(Graphische Systeme)
TU Dortmund

Lehrstuhl Informatik XII
(Eingebettete Systeme)
TU Dortmund

Fraunhofer IML
Verpackungs- und
Handelslogistik

Institut für
Produktionssysteme
TU Dortmund

Inhaltsverzeichnis

1. Einleitung	1
1.1. Motivation	1
1.2. Aufgabe der Projektgruppe	1
1.3. Struktur des Endberichtes	2
2. Logistischer Kontext	5
2.1. Grundlagen der Logistik	5
2.2. Fahrerlose Transportsysteme	8
2.3. Logistik im Anwendungsfall	12
3. Detektion dynamischer Positionsdaten	13
3.1. Aufnahmeverfahren	13
3.1.1. Projektive Geometrie	13
3.1.2. Lochkameramodell und perspektivische Projektion	17
3.1.3. Kamerakalibrierung	20
3.2. Bildverarbeitung	21
3.2.1. Definition	21
3.2.2. Bildverbesserung im Ortsraum	21
3.2.3. Kantendetektion	23
3.3. Objekterkennung	25
3.3.1. Segmentierung	26
3.3.2. Template-Matching	29
4. 3D-Lokalisierung und Orientierungsbestimmung von Objekten	31
4.1. Bildaufnahme	31
4.2. Punktwolken und Verarbeitung	36
4.2.1. Punktwolken	36
4.2.2. Repräsentation von Punktwolken	37
4.2.3. Entrauschen und Reduktion	38
4.2.4. Bilderzeugung	40
4.3. Objekterkennung	43
4.3.1. Filterung lokaler Attribute	44

4.3.2.	RANSAC	44
5.	Handhabung von Industrierobotern	47
5.1.	Problemstellung	47
5.2.	Industrieroboter	48
5.2.1.	Freiheitsgrade	48
5.2.2.	Arbeitsbereich	49
5.2.3.	Koordinatensysteme	50
5.2.4.	Transformation	51
5.3.	Bahnplanung	52
5.3.1.	Bewegungsformen	53
5.3.2.	Zellbasierte Methoden	55
5.3.3.	Potentialfelder	58
6.	Prädiktion von Positionsdaten	61
6.1.	Problemstellung	61
6.2.	Stochastische Tracking-Verfahren	61
6.2.1.	Kalman-Filter	62
6.2.2.	Partikel-Filter	64
6.2.3.	Condensation-Algorithmus	65
6.3.	Alternative Ansätze	67
7.	Automatisierung von Entladungsvorgängen	69
7.1.	Ablauf des Entladevorgangs	69
7.2.	Fahrzeu-erkennung und Positionsbestimmung	71
7.2.1.	Bildakquisition	71
7.2.2.	Detektion der FTF-Bildposition	76
7.2.3.	Berechnung der FTF-Position im Raum	80
7.3.	Paketerkennung	83
7.3.1.	Bildakquisition	84
7.3.2.	Detektion der Paketposition	87
7.3.3.	Berechnung der relativen Paketposition	92
7.4.	Prädiktion der Paketposition	94
7.4.1.	Approximation der Fahrzeugbahn	94
7.4.2.	Anwendung des Condensation-Algorithmus	97
7.4.3.	Prädiktion über größere Zeitintervalle	102
7.5.	Positionierung des Roboter-Greifarms	102
7.5.1.	Planung der Roboterbewegung	102

7.5.2.	Ansteuerung des Roboters	108
7.5.3.	Bestimmung der TCP-Position	111
7.6.	Zusammenwirken der Module zur Paketentnahme	115
8.	Evaluation	117
8.1.	Allgemeiner Versuchsaufbau	117
8.2.	Fahrzeu-erkennung und Positionsbestimmung	118
8.2.1.	Versuchsaufbau	118
8.2.2.	Evaluationskriterien	119
8.2.3.	Versuchsergebnisse und -bewertung	120
8.3.	Paketerkennung	121
8.3.1.	Versuchsaufbau	121
8.3.2.	Evaluationskriterien	122
8.3.3.	Versuchsergebnisse und -bewertung	122
8.4.	Prädiktion der Paketposition	123
8.4.1.	Versuchsaufbau	124
8.4.2.	Evaluationskriterien	124
8.4.3.	Versuchsergebnisse und -bewertung	124
8.5.	Robotersteuerung	125
8.5.1.	Versuchsaufbau	126
8.5.2.	Evaluationskriterien	126
8.5.3.	Versuchsergebnisse und -bewertung	126
8.6.	Vollständiger Entladevorgang	128
8.6.1.	Versuchsaufbau	128
8.6.2.	Evaluationskriterien	128
8.6.3.	Versuchsergebnisse und -bewertung	128
9.	Zusammenfassung und Ausblick	131
9.1.	Zusammenfassung	131
9.2.	Ausblick	132
	Literaturverzeichnis	135
	Abbildungsverzeichnis	139
	Tabellenverzeichnis	143
	A. Pflichtenheft	145

Mathematische Notation

Notation	Bedeutung
\mathbb{N}	Natürliche Zahlen ohne Null
\mathbb{Z}	Ganze Zahlen
\mathbb{R}	Reelle Zahlen
\mathbb{R}^+	Positive reelle Zahlen
\mathbb{P}^d	d -dimensionaler projektiver Raum für $d \in \mathbb{N}$
\mathbf{x}	Ein n -dimensionaler Vektor mit den Elementen x_1, \dots, x_n
x	Eine eindimensionale, skalare Variable
\mathcal{P}	Eine Menge von Vektoren
$ \mathcal{P} $	Mächtigkeit der Menge \mathcal{P}
\mathbf{A}	Eine Matrix mit Einträgen a_{ij}
$(\mathbf{x}_k)_{k \in \mathbb{N}}$	Eine Folge von Vektoren $\mathbf{x}_k \in \mathbb{R}^n$
$p(X)$	Wahrscheinlichkeitsverteilung einer Zufallsvariable X
$p(A B)$	Bedingte Wahrscheinlichkeit von A , unter der Bedingung, dass B eingetreten ist
$x^{\mathbf{a}}$	Polynom x mit Parametern des Vektors \mathbf{a} und des Grades $(n - 1)$ bei Länge n des Vektors \mathbf{a}

1. Einleitung

Der Begriff *DynOLog*, welcher als Bezeichnung der Projektgruppe fungiert, bedeutet dynamische Detektion von Objekten im logistischen Kontext. Im Folgenden wird der Kontext und die Aufgabenstellung des während der Projektgruppe entwickelten Systems motiviert (Kapitel 1.1). Um die Aufgabenstellung zu definieren, wird in Kapitel 1.2 beschrieben welche gewünschten Funktionen das fertige System zur Verfügung stellen soll. Im Anschluss wird in Kapitel 1.3 die Struktur des Endberichts erläutert.

1.1. Motivation

Für den Materialtransport in Logistikzentren oder Lagereinrichtungen wird der Einsatz eines innerbetrieblichen, flurgebundenen Fördersystems zum Standard [1]. Ein Fördersystem dieser Art wird Fahrerloses Transportsystem (FTS) genannt. Die Navigation der automatisch gesteuerten Fördersysteme erfolgt dabei völlig autonom. Um einen adäquaten Durchsatz beziehungsweise eine hohe Leistung zu erreichen, sind in der Regel möglichst viele autonom agierende, flurgebundene Fördermittel innerhalb eines solchen logistischen Systems zeitgleich aktiv. Ein flurgebundenes Fördermittel wird als Fahrerloses Transportfahrzeug (FTF) bezeichnet. Während die automatisierte Entnahme von Stückgut zur Beladung von Transportsystemen Stand der Technik ist [1], stellt eine automatisierte Lösung der Entladung der FTF an Übergabestationen, die bisher überwiegend manuell erfolgt, eine Herausforderung dar, insbesondere dann, wenn sich die Fahrzeuge in Bewegung befinden. Die hier anvisierte Lösung zur automatisierten Entladung der Transportsysteme sieht den Einsatz eines Industrieroboters und geeigneter Kameras vor. Aus Gründen der Energieeffizienz und der Durchsatzleistung sollen die vorbeifahrenden FTF ihre Geschwindigkeit nicht verringern, sondern während der Fahrt mithilfe eines Roboters, entladen werden. Dafür müssen sowohl die FTF als auch die jeweilige Beladung dynamisch erkannt werden, damit daraufhin die Bewegung des Industrieroboters geeignet angepasst werden kann.

1.2. Aufgabe der Projektgruppe

Die Zielsetzung dieser Projektgruppe ist der Entwurf und die Implementierung eines Systems zur Lokalisierung und Klassifizierung von sich in Bewegung befindenden FTF und deren Ladungsträgern sowie die dynamische Bewegungsplanung eines Roboters, der zur Entladung des FTF verwendet wird.

Hierzu soll eine exakte Lokalisation der FTF und der transportierten Ladungsträgern mit Paketen über eine bildbasierte Sensorik erfolgen, während sich das FTF der Entladestation

nähert. Vorgesehen ist dabei sowohl ein Einsatz von optischen 2D- und insbesondere 3D-Sensoren, wie PMD-Kameras (Photonic Mixer Device), die zu den 3D-Time-of-Flight-Kameras zählen, sowie Kinect- beziehungsweise Xtion-3D-Kameras (Prinzip des strukturierten Lichts). Zur weiteren Steigerung der Messgüte ist zudem die Fusion von 2D- und 3D-Sensoren zu realisieren. Im Hinblick auf die echtzeitfähige Bildverarbeitung soll die Umsetzung dabei optional unter Einsatz der Grafikkarte erfolgen.

Für den nachgeschalteten Prozess des Entladens, der thematisch dem Bereich des Bin-Picking zugeordnet wird, ist eine exakte Analyse der erfassten und vorverarbeiteten optischen Sensordaten notwendig. Dies ist die Grundlage für den nachfolgend umzusetzenden Schritt der Generierung kollisionsfreier Roboterbahnen. Nur durch eine adäquate Roboterbewegung ist es möglich, das Stückgut sicher aus dem sich bewegenden Ladungsträger zu greifen. Hierzu ermittelt das Bildverarbeitungssystem Informationen über Geschwindigkeit und Position aus den akquirierten Kameradaten. Die spezifizierten Abgrenzungs-, Muss- und Wunschkriterien befinden sich im Anhang A.

1.3. Struktur des Endberichtes

Nachdem in diesem Kapitel die Aufgabe der Projektgruppe DynOLog einleitend motiviert, die Problemstellung definiert und die Aufgabenbereiche abgesteckt wurden, erfolgt im Anschluss (Kapitel 2) zuerst eine Darstellung des logistischen Kontextes der Projektgruppe. Hierbei werden grundlegende Begriffe der Logistik angeführt, logistische Grundstrukturen erörtert sowie auf, die im Rahmen der Projektgruppe fokussierten, FTF eingegangen.

In den darauffolgenden vier Kapiteln werden die einzelnen Teilaspekte der zu bearbeitenden Aufgabe betrachtet und zu jedem grundlegende Definitionen getroffen. Zudem werden allgemeingültige Methoden und Herangehensweisen genannt und näher beschrieben, um bei der anschließend erfolgenden Umsetzung möglichst optimale Verfahren auswählen zu können. In Kapitel 3 werden dabei die Grundlagen für eine Erkennung der FTF mithilfe einer CCD-Kamera gelegt. Kapitel 4 beschreibt im Anschluss daran verschiedene Techniken zur Aufnahme von 3D-Informationen mittels einer Tiefenkamera. Da neben der Erkennung der Pakete auf den FTF auch der Transfer der Pakete aus diesen hin zu einem entsprechenden Lagerplatz betrachtet werden muss, wird in Kapitel 5 der Aufbau und die Funktionsweise eines Industrieroboters beschrieben, welcher diesen Transfer realisieren wird. Abschließend werden erforderliche Techniken zur Fusion der Sensordaten von den zuvor beschriebenen Bereichen sowie der Prädiktion der Paketposition in Kapitel 6 erörtert.

Nachdem die Grundlagen dargelegt wurden, erfolgt in Kapitel 7 eine Beschreibung des gesamten automatisierten Entladevorgangs. Hierzu werden die in den einzelnen Teilaspekten verwendeten Methoden genannt und die tatsächlich erfolgte Umsetzung detailliert beschrieben.

Das Kapitel 8 umfasst daran anschließend die Evaluation der Teilaspekte sowie des Gesamtsystems auf Basis eines allgemeinen Versuchsaufbaus. Beispielsweise werden hierbei die Erfolgsrate der FTF-Erkennung sowie der Paketentnahme betrachtet.

Abschließend folgt eine Zusammenfassung (Kapitel 9) und ein Ausblick auf zukünftige, noch mögliche Entwicklungen im Bereich der automatisierten Entladung. Im Anhang A ist zudem

1. Einleitung

das vorab definierte Pflichtenheft zu finden, welches diverse Kriterien beinhaltet und unter anderem auch den Produkteinsatz sowie die technische Produktumgebung beschreibt.

2. Logistischer Kontext

Da die Logistik den zentralen Kontext der Projektgruppe *DynOLog* darstellt, befasst sich dieses Kapitel einleitend mit dieser Thematik. Zur besseren Übersichtlichkeit wurde es in zwei zentrale Abschnitte unterteilt.

Der erste Abschnitt befasst sich mit Logistik. Um eine generelle Grundidee zu bekommen, was Logistik ist, werden mehrere Definitionen der Logistik aufgeführt und erläutert. Zudem werden wichtige logistische Grundstrukturen behandelt, die bei der logistischen Handhabung von Gütern eine entscheidende Rolle spielen. Es wird außerdem auf die wichtigsten Eigenschaften von Ladungsträgern und Mehrwegsystemen eingegangen, welche bei der Projektgruppe zum Einsatz kommen.

Im Anschluss wird der Begriff des Materialflusses erläutert, um in den zweiten Abschnitt, welcher sich mit den Thematiken FTF und FTS befasst, überzuleiten. Hierbei werden die Begriffe des FTS und des FTF definiert und es wird speziell auf verschiedene Methoden der Führung solcher Fahrzeuge eingegangen. Abschließend wird die Idee des Schwarmverhaltens von Robotern anhand von *Boids Modell nach Craig Reynolds* erklärt [2].

2.1. Grundlagen der Logistik

In diesem Abschnitt wird beschrieben, was Logistik grundlegend leistet und welche Kernaspekte dabei von Bedeutung sind. Die einleitend angeführten Definitionen sind dabei dem Buch *Logistiksysteme* von Hans-Christian Pfohl entnommen [3].

Definitionen

Wie viele andere wirtschaftliche und technologische Errungenschaften entstammt der Begriff der *Logistik* dem Militär. Dort umfasst die Logistik alle organisatorischen Tätigkeiten, die zur Unterstützung der Truppen gedacht sind. Diese beschränken sich nicht nur auf militärische Güter, sondern auch den Transport, die Quartierung und die Versorgung von Streitkräften. Auch in den Wirtschaftswissenschaften wird der Begriff Logistik verwendet, bezieht sich jedoch nur auf stoffliche Güter.

Eine flußorientierte Definition der Logistik beschreibt die Versorgung eines Empfangspunktes und lässt sich an dieser Stelle anhand der *Sechs R der Logistik* beschreiben.

Die Logistik sorgt dafür, dass ein Empfangspunkt mit dem richtigen Produkt, welches im richtigen Zustand ist, zum richtigen Zeitpunkt, am richtigen Ort, in der richtigen Menge und zu den richtigen Kosten versorgt wird.

Eine weitere flußorientierte Definition der Logistik macht sich den Begriff des Materialflusses zunutze. Dort heißt es, dass die Logistik die Organisation, Planung, Kontrolle und Durchführung

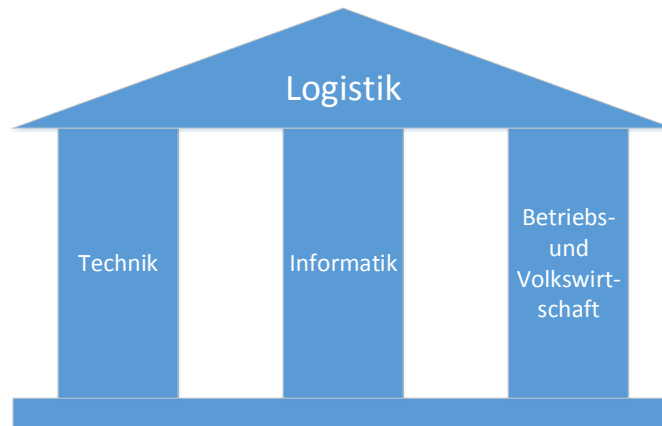


Abbildung 2.1.: Die drei Säulen der Logistik [4]

eines Materialflusses ist. Der dabei verwendete Begriff des Materialflusses ist dabei wie folgt definiert. Ein Materialfluß ist die Verknüpfung von Gewinn, Be- und Verarbeitung und Verteilung stofflicher Güter innerhalb bestimmter Produktionsbereiche.

Aus den dargelegten Definitionen lässt sich schließen, dass sich ein Großteil der Logistik mit der Beförderung und Lagerung von Gütern befasst. Um dies möglichst effizient und kostengünstig zu gestalten, baut die Logistik auf Informatik, Betriebs- und Volkswirtschaft und Technik auf. Diese bilden, wie in Abbildung 2.1 zu sehen, die *drei Säulen der Logistik*.

Grundstrukturen von Logistikprozessen

Ein Logistikprozess beschreibt den Transport von Gütern von Lieferpunkten zu Empfangspunkten. Um dies zu realisieren, gibt es drei zentrale Vorgehensweisen. Die einfachste Vorgehensweise ist ein direkter Güterfluß von genau einem Lieferpunkt zu einem Empfangspunkt. Dies wird *einstufiges System* genannt und ist in Abbildung 2.2a illustriert.

Falls mehrere Liefer- oder Empfangspunkte vorhanden sind, bietet sich dem Logistiker ein indirekter Güterfluß an. Dieser wird, wie in Abbildung 2.2b gezeigt, durch Auflösungs- oder Konzentrationspunkte realisiert und wird *mehrstufiges System* genannt.

Die Ideen des indirekten und direkten Güterflusses lassen sich kombinieren. Diese Systeme heißen *kombinierte Systeme*. Wie in Abbildung 2.2c zu beobachten, sehen diese Systeme den mehrstufigen Systemen sehr ähnlich. Sie besitzen jedoch zusätzliche *Shortcut-Verbindungen* vom Lieferpunkt zum Empfangspunkt. Der dynamische Aufbau dieser Systeme gibt dem Logistiker mehr Möglichkeiten den Güterfluß zu planen [3].

Ladungsträger

Um Güter sicher und unversehrt zu befördern, gibt es viele verschiedene Arten von Ladungsträgern. Daher sind Ladungsträger ein wichtiger Bestandteil der Lieferkette. Als Beispiele können folgende Begriffe genannt werden:

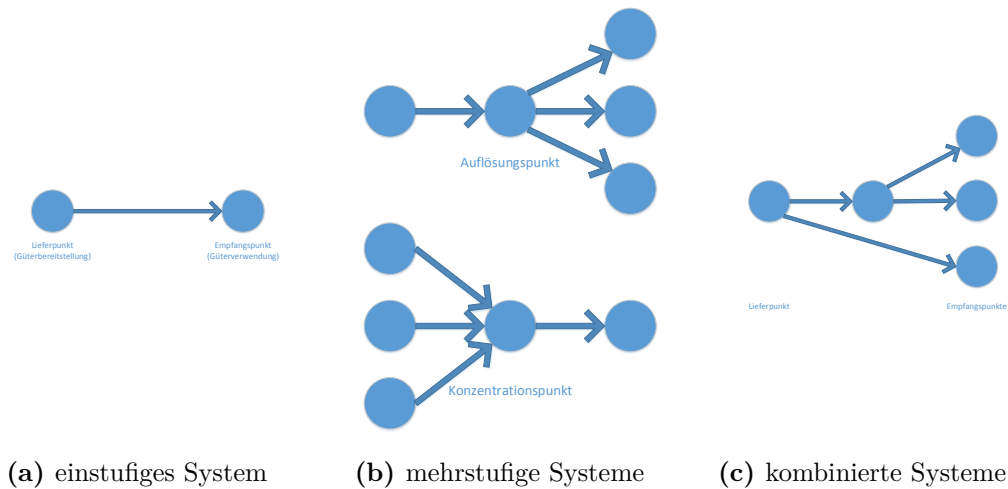


Abbildung 2.2.: Verschiedene Systeme von Logistikprozessen [3]

- Palette,
- Container,
- Karton,
- Box,
- Kiste.

Jeder Ladungsträger eignet sich für bestimmte Anwendungsgebiete. Dabei gibt es Herausforderungen, die im Bezug zu Ladungsträgern auftreten. Die Bewältigung dieser Herausforderungen bei minimalen Kosten wird *Ladungsträgermanagement* genannt und setzt sich zusammen aus Beschädigungen, Qualitätsmängel, Logistikkosten, Verfügbarkeit, Variantenvielfalt und die Schwierigkeit, den Bedarf und die Bestände zu verfolgen.

Das Ladungsträgermanagement bietet einige Lösungsansätze, wie sich diese Herausforderungen bewältigen lassen. Die Variantenvielfalt der Ladungsträger kann zum Beispiel reduziert werden, indem Standards eingesetzt werden, die in den verschiedenen Aufgabenbereichen von Ladungsträgern am sinnvollsten sind.

Eine nützliche Unterkategorie von Ladungsträgern stellen Mehrwegsysteme dar. Dies sind Ladungsträger, die nach dem Gebrauch erneut verwendet werden können und finden Erwähnung, da sie im Kontext der Projektgruppe angewandt werden.

Materialflußsystem

Der folgende Abschnitt stützt sich auf *Materialflußsysteme* von R. Jünemann und T. Schmidt [4]. Nach einer kurzen Exkursion zu Ladungsträgern befasst sich der folgende Unterabschnitt mit der Fortbewegung der Güter und geht auf die verschiedenen Fördermittel eines Materialflusssystems ein.

Ein Materialflußsystem beschreibt eine intralogistische Vorgangsstrategie zum Lagern, Transportieren, Zusammenführen und Verteilen von Waren. Diese Strategie lässt sich in folgende

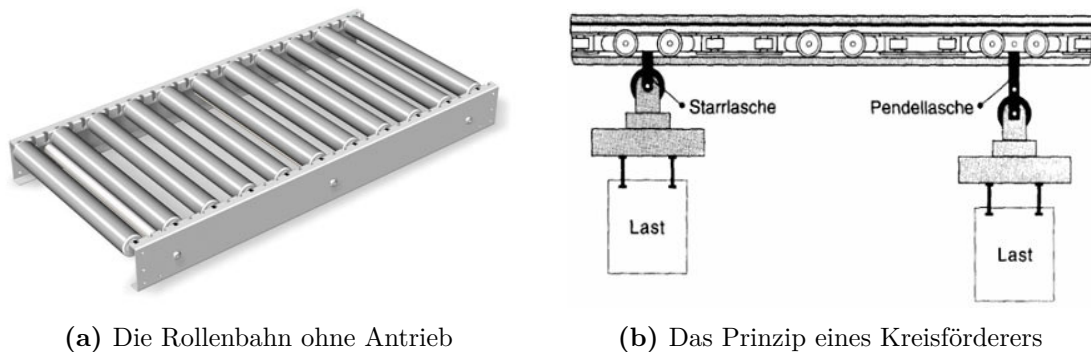


Abbildung 2.3.: Beispiele verschiedener Stetigförderer¹

Teiloperationen aufteilen: Fortbewegung, Bereitstellung, Entnahme, Abgabe und Rücktransport der Güter.

Fördermittel lassen sich in *Stetigförderer* und *Unstetigförderer* aufteilen. Die Stetigförderer fördern die Güter kontinuierlich und stoppen nicht. Die Unstetigförderer hingegen fördern in verschiedenen Intervallen, in denen die Förderung benötigt wird. Alle Fördermittel beider Gruppen werden weiterhin noch in drei Untergruppen aufgeteilt: flurgebunden, aufgeständert und flurfrei.

Ein anschauliches Beispiel für einen aufgeständerten Stetigförderer ist die Rollenbahn (siehe Abbildung 2.3a). Diese gibt es in zwei Ausführungen: angetrieben oder ohne Antrieb. Bei einer Rollenbahn, die ohne Antrieb fördert, muss der Neigungswinkel des Förderers groß genug sein, um die Güter mit Hilfe der Erdgravitation zu transportieren. Der Achsenabstand der Rollen muss kleiner als die Hälfte der Länge der zu transportierenden Ware sein, um eine fehlerfreie Förderung zu gewährleisten.

Ein Beispiel für einen flurfreien Stetigförderer stellt der Kreisförderer dar (siehe Abbildung 2.3b). Dabei sind die Güter an einem Gehänge befestigt, das typischerweise über Rollenlaufwerke befördert wird. Oft werden Kreisförderer an einer Deckenkonstruktion befestigt, wo der Förderer nicht im Konflikt mit aufgeständerten Fördermitteln steht.

Das im Kontext der Projektgruppe interessanteste Fördermittel lässt sich unter den flurgebundenen Unstetigförderern finden und heißt *fahrerloses Transportfahrzeug*. Der anschließende Abschnitt befasst sich mit diesem Fördermittel und durchleuchtet die wichtigsten Aspekte, die im Bezug auf die Projektgruppe von Bedeutung sind.

2.2. Fahrerlose Transportsysteme

Um den Begriff der FTS greifbar zu machen, wird zuerst erklärt, was *Cyber-Physical-Systems* sind und welche Verbindungen mit der Robotik existieren. FTS sind Systeme, in denen FTF, welche Roboter in der Logistik darstellen, miteinander interagieren und verschiedene logistische Aufgaben bewältigen.

¹TU Dortmund: <https://ews.tu-dortmund.de> und Industrial Packaging: <http://www.industrial-packaging.de>

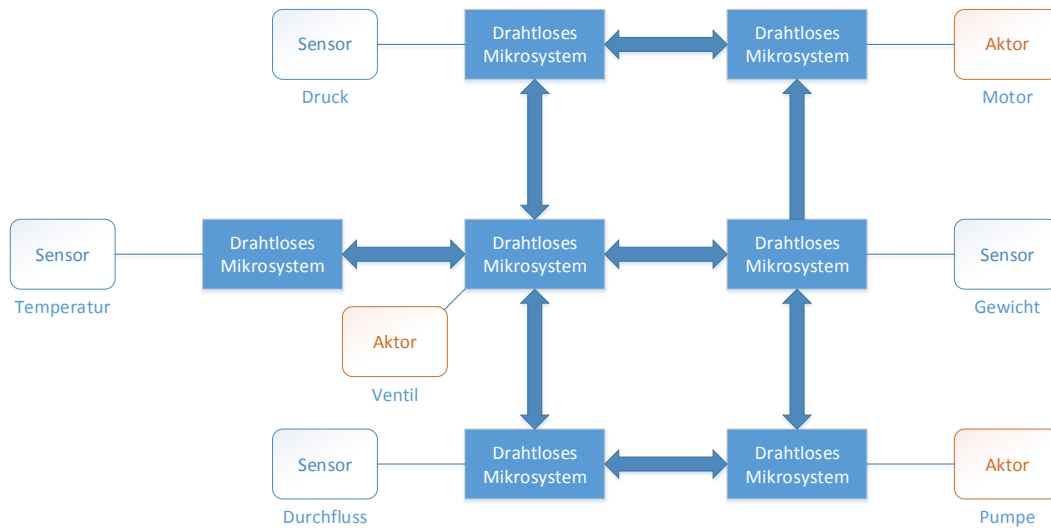


Abbildung 2.4.: Dieses Beispiel eines Sensor-Aktor-Netzwerkes zeigt die Interaktion zwischen Sensoren und Aktoren mit Hilfe von drahtloser Kommunikation ²

Definitionen

Eingebettete Systeme, die einen starken Bezug zur physikalischen Umgebung aufweisen, werden *Cyber-Physical-Systems* genannt. Die Konstruktionen und Implementierungen solcher Systeme legen besonderen Wert auf die Integration von physikalischen Faktoren wie Laufzeit, Energie und Speicherplatz.[5]

Roboter sind das typische Beispiel für Cyber-Physical-Systems. Sie stellen eine Kombination von elektrischen und mechanischen Systemen dar und interagieren mit Hilfe von Sensorik und Aktorik mit ihrer Umwelt. Ein Beispiel eines solchen *Sensor-Aktor-Netzwerkes* lässt sich aus Abbildung 2.4 entnehmen. Die Reaktionen auf den Verlauf der Umwelt müssen Anforderungen an Echtzeitschranken einhalten, bei deren Nichteinhalten ein Verlust von Qualität auftreten kann oder sogar Personen zu Schaden kommen können. Eine Echtzeitschranke, deren Nichteinhalten in einer Katastrophe enden kann, heißt *harte Echtzeitschranke* [5]. Die Implementierung solcher Systeme muss platzsparend und der Energieverbrauch effizient sein.

FTF sind automatisch geführte, flurgebundene Roboter, die Materialtransportaufgaben übernehmen. Die Fahrzeugführung ist entweder durch eine reale oder eine virtuelle Leitlinie realisiert. Mehrere FTF werden, durch einen übergeordneten Rechner gesteuert, oft zu einem FTS kombiniert.

Um ein FTS zu realisieren, müssen verschiedene Teilaspekte beachten werden, damit es sinnvoll betrieben werden kann. Diese Teilaspekte werden im Folgenden aufgeführt [6].

- Die Kommunikation und Synchronisation der einzelnen FTF untereinander muss realisiert werden.
- Die Fahrzeugführung, die anhand einer realen oder virtuellen Leitlinie vollzogen wird, wird im anschließenden Unterabschnitt behandelt.

²IT Wissen: <http://www.itwissen.info>

- Schnelle Reaktionszeit auf Geschwindigkeits- und Richtungsänderungen ist unabdingbar und wird mit Hilfe von geeigneten Algorithmen und Datenstrukturen umgesetzt.
- Die Erkennung von Hindernissen ist ein wichtiger Teilaspekt zur Kollisionsvermeidung.

Fahrzeugführung

Die Fahrzeugführung kann entweder durch eine reale oder durch eine virtuelle Leitlinie realisiert.

Reale Leitlinie

Es gibt mehrere Verfahren die Fahrzeugführung anhand einer realen Leitlinie umzusetzen. In diesem Abschnitt wird der Leitdraht und die Transpondertechnik vorgestellt.

Der Leitdraht wird im Boden verbaut und führt einen sinusförmigen Strom. Das FTF besitzt einen Sensor, der das durch den sinusförmigen Strom induzierte elektrische Feld abtastet und somit die Querabweichung berechnet. Diese Abweichung wird anschließend korrigiert. Diese Technik ist sehr genau jedoch auch extrem unflexibel, da sich das FTF nur entlang der physikalisch vorhandenen Leitlinie bewegen kann. Falls eine neue Route festgelegt werden soll, muss auch der Leitdraht, der im Boden eingebettet ist, neu verlegt werden.

Eine weitere Methode, die in die Kategorie der realen Leitlinie fällt, ist die Transpondertechnik. Dabei werden Transponder in einem Abstand von etwa $3m$ im Boden implementiert. Die Transponder selbst besitzen keine eigene Energieversorgung, da das FTF selbst ein Lesegerät besitzt, das ein magnetisches Feld zur Energieversorgung des Transponders abstrahlt. Je näher das FTF einem Transponder kommt, desto schneller ist der entsprechende Transponder mit genügend Energie versorgt. Sobald der Transponder ausreichend viel Energie besitzt, wird der auszuführende Code, der auf dem Transponder gespeichert ist, zum FTF gesendet [6].

Virtuelle Leitlinie

Die virtuelle Leitlinie bedient sich dem Prinzip der Umweltabbildung mit Hilfe verschiedener Sensoren. Das FTF enthält somit Informationen über seinen Aufenthaltsort und kann seine Fortbewegung der Zielposition entsprechend anpassen. Er kann außerdem Hindernisse leichter erkennen und ihnen ausweichen. Das Prinzip der virtuellen Leitlinie besitzt einen hohen Grad an Dynamik. Das FTF ist nicht auf physikalisch vorhandene Installationen angewiesen und kann sich frei im Raum bewegen und verschiedene Routen autonom bewältigen.

Ein Ultraschallsensor kann zum Beispiel für die Umweltabbildung genutzt werden. Dieser Sensor besteht aus einem Ultraschall-Generator und einem Ultraschall-Empfänger. Der Ultraschall-Generator sendet einen Impuls mit einer Frequenz ab $16kHz$. Der Ultraschall-Empfänger empfängt das Echo des Sendeimpulses, das entsteht, sobald der Impuls auf ein Hindernis stößt. Die Zeit zwischen dem Senden und dem Empfangen, mit deren Hilfe die Distanz zum Hindernis berechnen werden kann, wird gemessen. Das Prinzip ist in Abbildung 2.5 illustriert.

Andere Möglichkeiten für die Abbildung der Umwelt, wie das Prinzip des *strukturierten Lichtes*, der *Stereokamera* und des *PMD-Sensors*, werden in Kapitel 4.1 erläutert.

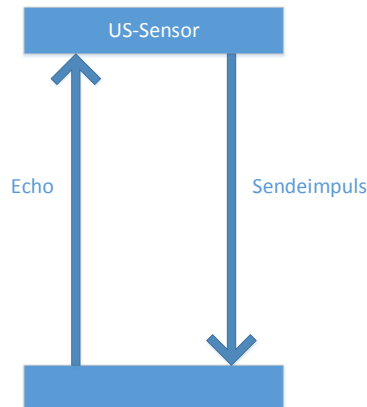


Abbildung 2.5.: Prinzip eines Ultraschallsensors ³

Eine Kombination zwischen virtueller und realer Leitlinie stellt die Odometrie dar. Dabei wird bei einer bekannten Startposition die aktuelle Position des Roboters ermittelt, indem die vollbrachten Bewegungen, wie zum Beispiel die Anzahl der Radumdrehungen, gemessen werden. Diese Methode ist jedoch sehr fehleranfällig, da viele Störeinflüsse existieren, die die Genauigkeit der Berechnung beeinträchtigt. Beispiele für solche Einflüsse sind die Beschaffenheit der Räder, die Beschaffenheit des Untergrundes, das Fahrzeuggewicht oder andere von außen einwirkende Kräfte. Aus dem Grunde der hohen Fehleranfälligkeit wird Odometrie nur in Verbindung mit den zuvor genannten Methoden verwendet. [4]

Schwarmverhalten

In diesem Unterabschnitt wird kurz auf das Prinzip des Schwarmverhaltens eingegangen, das bei FTS häufig eingesetzt wird, da viele FTF miteinander kommunizieren und agieren müssen. Schwarmverhalten ist ein regelbasiertes Verhalten, das die Wechselwirkung zwischen Individuen beschreibt und in einem dezentralen System eingesetzt werden kann. Schwärme sind oft im Tierreich zu beobachten und werden anhand dieses Vorbildes auch bei Robotern umgesetzt. Im Folgenden wird *Boids Modell nach Craig Reynolds* [2] vorgestellt, das eine Methode beschreibt, wie Schwarmverhalten bei anorganischen Individuen realisiert werden kann.

Die einzelnen Individuen werden *Boids* genannt, welche jeweils eine Position und eine Geschwindigkeit als Attribute besitzen. Jeder der Boids reagiert auf die Position und Geschwindigkeit seiner unmittelbaren Nachbarn und passt seine eigene Position und Geschwindigkeit dementsprechend an. Dies wird durch drei Unterkonzepte realisiert: *Seperation*, *Alignment* und *Cohesion*.

Das Prinzip der *Seperation* stellt sicher, dass jeder Boid den Abstand zu seinen unmittelbaren Nachbarn hält, dabei jedoch die Bewegungsbahn anderer Boids berücksichtigt, um Kollisionen zu vermeiden.

Alignment beinhaltet die Anpassung des Verhaltens eines Boids an das Gruppenverhalten. Dabei wird der Richtungsvektor jedes Boids mit Hilfe des durchschnittlichen Richtungsvektors

³Wikimedia: www.wikimedia.org

aller Boids in einer bestimmten Umgebung korrigiert, sodass sich alle Boids in etwa in die gleiche Richtung bewegen.

Das Prinzip der Cohesion ermöglicht es einem Boid auf die durchschnittliche Zielposition aller Boids in der unmittelbaren Umgebung zuzusteuern. Dieses Prinzip funktioniert nur, wenn auch die Separation durchgeführt wird, da es sonst zu Kollisionen kommen würde.

Zusätzlich zu den soeben vorgestellten drei Prinzipien, welche ein Schwarmverhalten realisieren, existiert das Konzept des *Flockings*. Dieses Konzept ermöglicht es einzelnen Individuen sich der Gruppe anzuschließen und den Schwarm somit zu vergrößern. [2]

2.3. Logistik im Anwendungsfall

Logistik bildet die Grundlage des Anwendungsfalles, da das entwickelte System in einem logistischen Kontext, Entladung und Transport von Paketen, eingesetzt werden soll. Hierbei wird zum Beispiel ein FTS eingesetzt. Weitere Details über die eingesetzten FTFs und den Versuchsaufbau finden sich in Anhang A.

3. Detektion dynamischer Positionsdaten

Das Entladen eines sich bewegenden fahrerlosen Transportfahrzeuges mittels eines Roboters erfordert die Detektion der Positionsdaten des Fahrzeuges. Die automatische Erkennung eines fahrerlosen Transportfahrzeuges (FTF) kann optisch erfolgen, womit zunächst einmal die optische Aufnahme einer Szene zur späteren Verarbeitung bewerkstelligt werden muss. Dazu werden zunächst Grundlagen von Aufnahmeverfahren und die Kalibrierung von Kameras in Abschnitt 3.1 vorgestellt. Zur besseren Verarbeitung der Daten werden in Abschnitt 3.2 einige Verfahren zur Bildverarbeitung erläutert. Abschließend werden in Abschnitt 3.3 Verfahren zur Segmentierung von Bildern und zum Erkennen von Objekten vorgestellt.

3.1. Aufnahmeverfahren

Zur Be- und Entladung eines fahrerlosen Transportfahrzeuges durch einen Roboter bedarf es zunächst der Erkennung des Fahrzeuges sowie der Bestimmung seiner Position relativ zu einem Bezugskordinatensystem mittels geeigneter technischer Aufnahmegeräte. Eine Möglichkeit ist die Bestimmung der Position und Rekonstruktion der FTFs durch optische 3D-Sensoren, wie zum Beispiel Laserscanner (siehe Kapitel 4.1). Eine weitere Möglichkeit ist die Rekonstruktion durch zweidimensionale Aufnahmen, zum Beispiel gewöhnlichen Fotos.

Abbildung 3.1 zeigt ein Foto der Logistik-Halle des IML. Die 3D-Rekonstruktion von Objekten aus einer solchen Aufnahme birgt einige Teilprobleme. In der Abbildung liegt eine perspektivische Aufnahme vor. Um diese Szene geometrisch zu beschreiben, wird die im Folgenden vorgestellte *projektive Geometrie* verwendet. Ein weiteres Problem stellt die offensichtliche Verzerrung der Aufnahme dar. Für eine korrekte Rekonstruktion muss diese Verzerrung zurückgerechnet werden, welche durch eine *Kamerakalibrierung* ermittelt werden kann.

Im Folgenden wird zunächst die projektive Geometrie vorgestellt. Anschließend werden Kameramodelle und die perspektivische Projektion erläutert. Abschließend wird die Kamerakalibrierung beschrieben.

3.1.1. Projektive Geometrie

Die *projektive Geometrie* (im Folgenden angelehnt an die Ausführungen in [7]) ist eine Generalisierung der affinen Geometrie, welche wiederum eine Verallgemeinerung der euklidischen Geometrie ist. Je allgemeiner eine Geometrie ist, also je weniger Einschränkungen sie bezüglich der invarianten Eigenschaften ihrer Objekte zulässt, desto stärker können ihre Objekte durch entsprechende Transformationen verändert werden und desto weniger geometrische Eigenschaften behalten sie nach der Transformation. Eine Transformation eines bestimmten Objektes A



Abbildung 3.1.: Perspektivische Aufnahme der ZFT-Halle des Fraunhofer-Institut für Materialfluss und Logistik (IML) mit Verzerrung¹

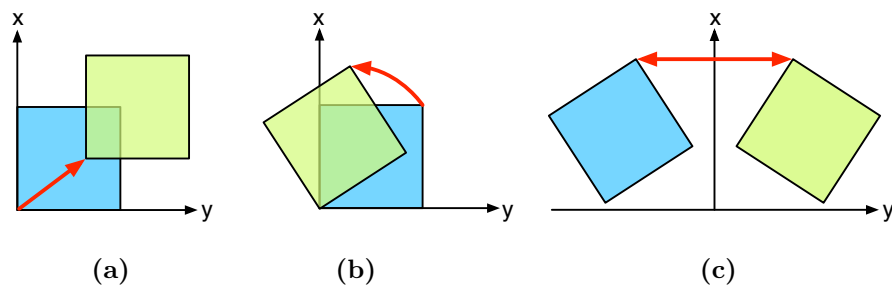


Abbildung 3.2.: Illustration euklidischer Transformationen: (a) Translation (b) Rotation und (c) Spiegelung an beliebiger Achse

in einer bestimmten Geometrie G bringt immer dasselbe Objekt A hervor, mit denselben, in der jeweiligen Geometrie G invarianten Eigenschaften.

Euklidische Geometrie

Die euklidische Geometrie ist durch Abbildung 3.2 beispielhaft illustriert. In der euklidischen Geometrie können Objekte durch Translation, Rotation und Spiegelung transformiert werden. Invariante Eigenschaften unter euklidischen Transformationen sind Längen, Winkel, Parallelen, Teilverhältnisse und Doppelverhältnisse.

Ein Quadrat A , das rotiert und gespiegelt wird, stellt nach der Transformation immer noch dasselbe Quadrat A mit den selben Eigenschaften dar. Zwei Punkte des Quadrates haben also denselben Abstand wie zuvor und die Verbindungslinien zweier Punkte denselben Winkel. Das Objekt A nach der Transformation ist also dasselbe Objekt A wie vor der Durchführung der Transformation. Würde beispielsweise eine affine Transformation (z.B. eine Scherung) durchgeführt, würde das transformierte Objekt B nicht mehr das ursprüngliche Objekt B sein,

¹Fraunhofer-Institut für Materialfluss und Logistik (IML): <http://iml.fraunhofer.de>

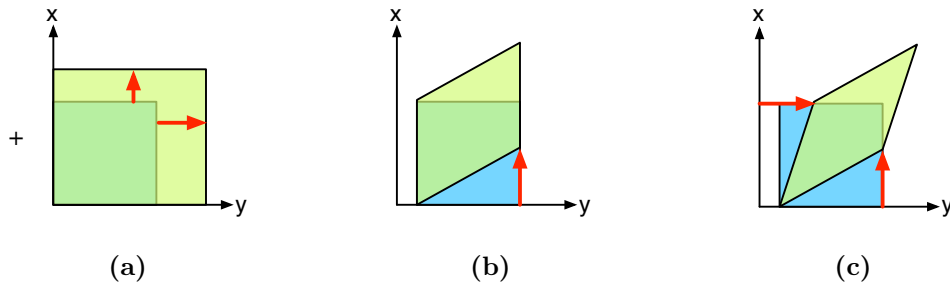


Abbildung 3.3.: Illustration affiner Transformationen: (a) Skalierung (b) Asymmetrische Scherung und (c) Symmetrische Scherung

denn es gingen zum Beispiel Winkel verloren, da diese unter affinen Transformationen nicht invariant sind.

Affine Geometrie

Die *affine Geometrie* ist eine Verallgemeinerung der euklidischen Geometrie. In der affinen Geometrie gibt es sowohl die euklidischen Transformationen, als auch zusätzliche affine Transformationen wie die Skalierung sowie die symmetrische sowie asymmetrische Scherung. Durch diese beiden zusätzlichen Transformationen gehen in der affinen Geometrie die Invarianten Länge und Winkel verloren. Das bedeutet, dass nach der Transformation eines Objektes Längen und Winkel unterschiedlich sein können. Das Objekt ist in der affinen Geometrie dennoch dasselbe (die selbe Instanz des Objektes) wie vor der Transformation. Abbildung 3.3 veranschaulicht Transformationen der affinen Geometrie.

Projektive Geometrie

Die *projektive Geometrie* ist eine Verallgemeinerung der affinen Geometrie. Sie enthält neben allen euklidischen und affinen Transformationen zusätzlich eine für die Projektionsgruppe besonders wichtige Transformation: die Perspektivprojektion, welche in Abbildung 3.4 illustriert ist. Wie der Name schon andeutet, können Objekte mit der Perspektivprojektion betrachtungswinkelabhängig verzerrt werden. Somit ist es möglich, Objekte so zu projizieren, wie es beispielsweise auch beim menschlichen Auge oder einer gewöhnlichen Fotokamera der Fall ist. Die Perspektivprojektion hat zur Folge, dass neben Teilverhältnissen insbesondere auch parallele Geraden nicht erhalten werden. Die einzige invariante Eigenschaft ist das Doppelverhältnis.

Das Foto der Halle des IML aus Abbildung 3.1 veranschaulicht die Perspektivprojektion. Am linken Rand ist ein Zaun zu sehen. Die horizontalen Streben des Zaunes verlaufen in der Realität parallel. Die perspektivische Aufnahme durch die Kamera stellt jedoch eine betrachtungswinkelabhängige Darstellung dar und entspricht somit einer Perspektivprojektion. Es ist deutlich zu sehen, dass die horizontalen Streben im Bild nicht mehr parallel verlaufen, sondern scheinbar in der Ferne zusammenlaufen.

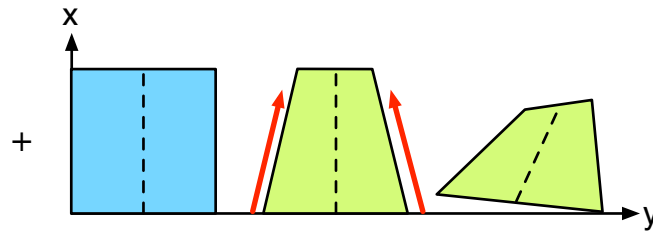


Abbildung 3.4.: Illustration projektiver Transformationen: Perspektivprojektion

Grundlagen der projektiven Geometrie

Die projektive Geometrie wird für die perspektivische Darstellung dreidimensionaler Objekte im Zweidimensionalen verwendet [7]. Wie im vorherigen Abschnitt gezeigt wurde, werden bei Perspektivprojektionen keine parallelen Geraden erhalten. Präziser gesagt scheinen sich parallele Geraden in perspektivischer Darstellung im Unendlichen zu schneiden. Um diese Eigenschaft mathematisch zu beschreiben, werden in der projektiven Geometrie sogenannte *Fernpunkte* (auch *unendliche* oder *unendlich ferne Punkte* genannt) definiert. Diese werden dazu genutzt, um den Schnitt paralleler Geraden im Unendlichen mathematisch zu beschreiben. Allgemein werden Punkte in der projektiven Geometrie durch *homogene Koordinaten* angegeben, bei denen eine dritte Koordinatenkomponente $w \neq 0$ hinzugefügt wird: die sogenannte *homogene Koordinate*. Ein Vektor \mathbf{x} des \mathbb{R}^2 wird durch homogene Koordinaten mit

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2 \implies \mathbf{x}' = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim k \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \in \mathbb{P}^2 \text{ für } k \neq 0 \quad (3.1)$$

für $w = 1$ auf einen Vektor \mathbf{x}' in der projektiven Ebene \mathbb{P}^2 abgebildet. Projektive Koordinaten sind also bis auf einen Skalierungsfaktor k gleich. Diese Eigenschaft wird bei der Abbildung eines Punktes im Raum auf eine Bildebene durch ein perspektivisches Kameramodell genutzt. Durch diese Eigenschaft ist die Umkehrung der Abbildung im Allgemeinen nicht eindeutig:

$$\mathbf{x}' = \begin{pmatrix} x \\ y \\ w \end{pmatrix} \in \mathbb{P}^2 = w \cdot \begin{pmatrix} x/w \\ y/w \\ 1 \end{pmatrix} \implies \mathbf{x} = \begin{pmatrix} x/w \\ y/w \end{pmatrix} \in \mathbb{R}^2. \quad (3.2)$$

Fernpunkte werden ebenfalls mit homogenen Koordinaten dargestellt, wobei die homogene Koordinate hier mit $w = 0$ belegt wird und somit einen Fernpunkt bzw. einen Punkt im Unendlichen

$$\mathbf{x}'_{\infty} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \in \mathbb{P}^2 \quad (3.3)$$

definiert.

Der zweidimensionale projektive Raum (die projektive Ebene) \mathbb{P}^2 lässt sich als eine in den \mathbb{R}^3 eingebettete Ebene vorstellen:

$$\mathbb{P}^2 = \mathbb{R}^3 \setminus \{(0, 0, 0)^{\top}\}. \quad (3.4)$$

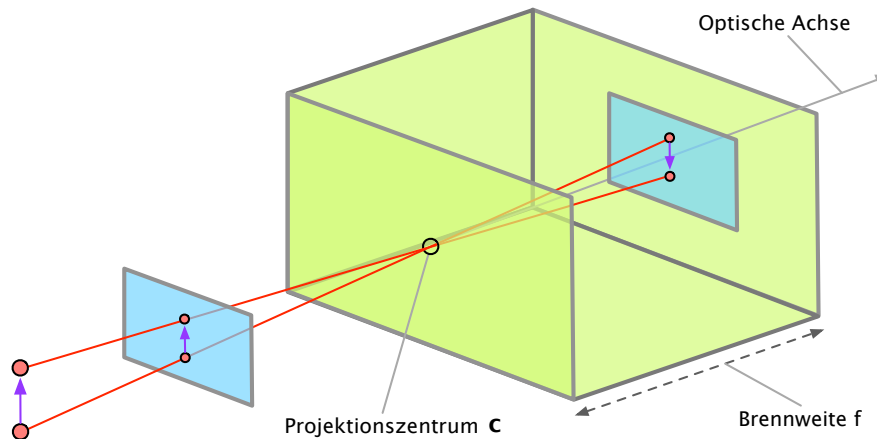


Abbildung 3.5.: Schematische Darstellung des Lochkameramodells

Der Punkt $(0, 0, 0)^T$ existiert in der projektiven Geometrie nicht. Die projektive Ebene kann prinzipiell an jeder beliebigen Position im \mathbb{R}^3 liegen. Es wird jedoch definiert, dass sie bei $z = 1$ liegt. Daher ist die homogene Koordinate w bei der Abbildung eines Vektors in den projektiven Raum auch mit $w = 1$ als Repräsentant gewählt. Wie Gleichung 3.1 verdeutlicht, ist ein projektiver (bzw. homogener) Vektor bis auf einen Skalierungsfaktor k äquivalent. Dieser Skalierungsfaktor repräsentiert in dieser Vorstellung die z -Position der projektiven Ebene und kann daher beliebig gewählt werden. Zur Abbildung eines Punktes im Raum auf die projektive Ebene lassen sich Strahlen vom Ursprung des \mathbb{R}^3 aus mit der projektiven Ebene schneiden und damit Punkte in der projektiven Ebene definieren. Ein Punkt $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ im \mathbb{P}^2 entspricht also einem *Strahl* $k \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ im \mathbb{R}^3 .

3.1.2. Lochkameramodell und perspektivische Projektion

Die perspektivische Projektion lässt sich mathematisch mit Hilfe von Kameramodellen modellieren. Im Folgenden werden zunächst ein ideales Kameramodell und nicht-ideale Kameramodelle vorgestellt. Anschließend wird die Kameraverzeichnung bei nicht-idealen Kameramodellen gesondert erläutert.

Ideales Kameramodell

Mit den im letzten Abschnitt beschriebenen Grundlagen lässt sich die perspektivische Projektion einer Szene im Raum auf eine Bildebene mathematisch definieren [8]. Dazu wird vom sogenannten *idealen Lochkameramodell* ausgegangen, welches eine Abstraktion der realen *Camera Obscura* darstellt. In Abbildung 3.5 ist das Lochkameramodell schematisch dargestellt. Das Lochkameramodell führt eine sogenannte *Zentralprojektion* aus. Punkte im dreidimensio-

nenen Raum werden mittels Strahlen durch das Projektionszentrum auf eine Bildebene mittels projektiver Geometrie abgebildet. Mathematisch wird ein dreidimensionaler Punkt \mathbf{x} mittels

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 \implies \mathbf{x}' = \begin{pmatrix} f \cdot x/z \\ f \cdot y/z \end{pmatrix} \in \mathbb{R}^2 \quad (3.5)$$

auf die Bildebene projiziert. Bei Betrachtung der projektiven Geometrie, wird \mathbf{x} zunächst auf die projektive Ebene (jene mit Abstand f zum Projektionszentrum) projiziert und anschließend lediglich die homogene dritte Koordinate (f) entfernt, um einen Punkt in der zweidimensionalen reellen Ebene darzustellen:

$$\mathbf{x}' = \begin{pmatrix} f \cdot x/z \\ f \cdot y/z \\ f \end{pmatrix} \in \mathbb{P}^2 \implies \mathbf{x}' = \begin{pmatrix} f \cdot x/z \\ f \cdot y/z \end{pmatrix} \in \mathbb{R}^2. \quad (3.6)$$

Mit Gleichung 3.5 lässt sich nun eine homogene Transformationsmatrix

$$\mathbf{K} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.7)$$

aufstellen, welche dreidimensionale Punkte auf eine zweidimensionale Bildebene abbildet. Diese Matrix wird *intrinsische Matrix* genannt, da sie die intrinsischen, also inneren, Parameter der Kamera enthält. Die Zentralprojektion stellt jedoch lediglich den intrinsischen Teil der perspektivischen Projektion dar. Um die Projektion mathematisch beschreiben zu können, müssen vorher die Koordinaten eines dreidimensionalen Punktes im Raum eindeutig in einem Weltkoordinatensystem definiert werden. Hierzu muss die Kamera, die die Projektion durchführt, eindeutig in diesem Weltkoordinaten bestimmt sein. Die Position der Kamera wird lediglich durch eine Rotation und Translation vom Ursprung des Weltkoordinatensystems mittels einer homogenen Transformationsmatrix

$$\mathbf{D} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \quad (3.8)$$

definiert, wobei \mathbf{R} eine 3×3 -Rotationsmatrix und \mathbf{t} ein 3×1 -Translationsvektor ist. Die Matrix \mathbf{D} wird *extrinsische Matrix* genannt, da sie die extrinsischen Parameter der Projektion beinhaltet.

Abbildung 3.6 zeigt schematisch den gesamten Aufbau der perspektivischen Projektion. Zunächst wird der zu projizierende Punkt \mathbf{x} in homogenen Weltkoordinaten mit der extrinsischen Matrix \mathbf{D} ins Kamerakoordinatensystem transformiert. Anschließend wird er mit der intrinsischen Matrix \mathbf{K} auf die Bildebene projiziert, wobei die Tiefeninformation verloren geht. Die perspektivische Projektion lässt sich damit durch eine Projektionsmatrix \mathbf{P} mit

$$\mathbf{x}' = \mathbf{K}\mathbf{D}\mathbf{x} = \mathbf{P}\mathbf{x} \quad (3.9)$$

zusammenfassen.

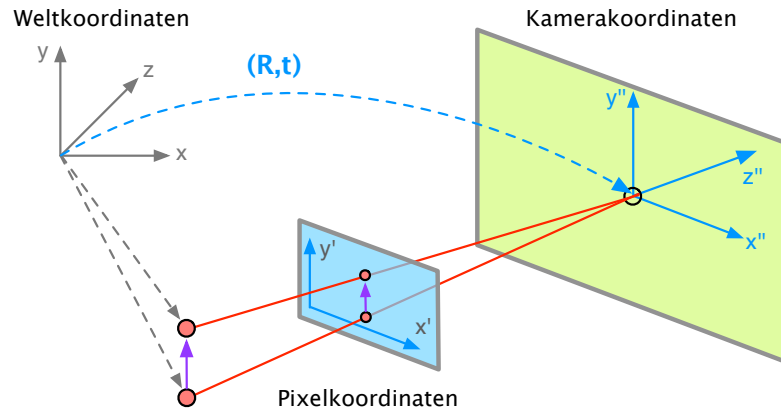


Abbildung 3.6.: Schematische Darstellung der perspektivischen Projektion

Nicht-ideale Kameramodelle

Das Lochkameramodell stellt ein ideales Kameramodell dar. In der Realität ist es jedoch nicht möglich, eine Kamera mit idealen Eigenschaften herzustellen. Aus diesem Grund weisen reale Kameras immer Eigenschaften auf, welche zur korrekten mathematischen Beschreibung modelliert werden müssen. Alle diese Eigenschaften ergeben sich aus der inneren Geometrie der Kamera, also den intrinsischen Kamera-Parametern, welche im Folgenden erläutert werden. Die Sensorelemente einer Kamera sind bei realen Kameras nicht infinitesimal klein, sondern besitzen eine gewisse Ausdehnung $\mathbf{m} = \begin{pmatrix} m_x \\ m_y \end{pmatrix}$ in Breite und Höhe. Sie beeinflusst, wieviel „Raum“ auf ein einziges Pixel im Bild abgebildet wird. Des Weiteren sind die Sensorelemente nicht immer quadratisch, sondern besitzen eine gewisse Schiefe s , die berücksichtigt werden muss. Für die Abbildung in Bildkoordinaten muss der Ursprung der Kamerakoordinatensystemes, der Kamera-Hauptpunkt $\mathbf{c} = \begin{pmatrix} c_x \\ c_y \end{pmatrix}$, bekannt sein. Für die Modellierung einer realen Kamera können diese Parameter in der intrinsischen Matrix

$$\mathbf{K} = \begin{pmatrix} f \cdot m_x & s & c_x \\ 0 & f \cdot m_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (3.10)$$

berücksichtigt werden.

Kameraverzeichnung

Über die im vorherigen Abschnitt vorgestellten intrinsischen Parameter der Kamera gibt es noch weitere Größen, die bei der Modellierung berücksichtigt werden müssen. Dabei handelt es sich um die *Kameraverzeichnung* oder auch *Linienverzeichnung* [9]. Die Kameraverzeichnung ist objektivbedingt und verzerrt das Bild in verschiedenen Formen. Üblicherweise handelt es sich dabei um radialsymmetrische Verzerrungen, bei denen die Verzerrung mit Abstand zum Verzerrungszentrum zunimmt, oder um tangentielle Verzerrungen, bei denen die Verzerrung entlang einer Tangente am Verzerrungszentrum entsteht. Im Folgenden wird nur die

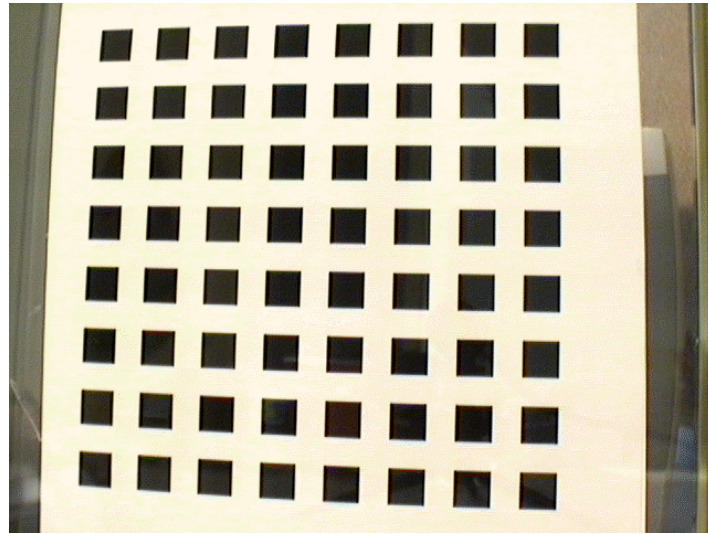


Abbildung 3.7.: Planares Kalibriermuster [10]

radialsymmetrische Verzerrung betrachtet, da sie den größten Anteil der Kameraverzeichnung ausmacht.

Die radialsymmetrische Verzerrung kommt in zwei Formen vor: die *kissenförmige* Verzerrung verzerrt das Bild ins Verzerrungszentrum hinein, die *tonnenförmige* Verzerrung verzerrt das Bild vom Verzerrungszentrum weg. Beiden ist gemein, dass sie gerade Linien am Rand des Bildes auf Kurven abbilden, was ein unerwünschter Effekt ist. Ein reales Beispiel zeigt Abbildung 3.1, welche eine leichte tonnenförmige Verzerrung aufweist.

Mathematisch wird die Linsenverzerrung durch eine nicht-lineare Funktion mit radialen Verzerrungskoeffizienten κ modelliert. Üblicherweise genügt es, lediglich einen einzigen Verzerrungskoeffizienten für die Modellierung zu verwenden.

3.1.3. Kamerakalibrierung

Der Zweck der Kamerakalibrierung ist es, die intrinsischen und extrinsischen Parameter einer realen Kamera zu ermitteln. Eine verbreitete Möglichkeit ist die Kamerakalibrierung nach Zhang [10]. Die Kalibrierung nach Zhang erfordert keine kostspieligen Kalibrierobjekte oder Kameraaufbauten. Es genügt, ein ebenes Kalibriermuster mit einer Kamera aus verschiedenen Ansichten aufnehmen zu können.

Die einzelnen Schritte der Zhang-Kalibrierung sind folgende [10]:

- Ein Kalibriermuster (siehe Abbildung 3.7) ausdrucken und auf einer ebenen Fläche anbringen,
- Aufnahmen des Kalibrierusters aus verschiedenen Ansichten erstellen,
- Merkmalspunkte in den Ansichten detektieren. Dazu werden kantenbasiert die Eckpunkte der Quadrate in jeder Ansicht gesucht.
- Intrinsische und extrinsische Parameter in geschlossener Form bestimmen,
- Koeffizienten der radialen Verzerrung mit Methode der kleinsten Quadrate schätzen,

- Parameter mit der Maximum-Likelihood-Methode optimieren.

Die Zhang-Kalibrierung wird in Abschnitt 7.2.1 detailliert beschrieben.

3.2. Bildverarbeitung

Je nach Aufnahmegesetz und äußeren Bedingungen, wie zum Beispiel Lichtverhältnisse, können Bilder verrauscht, überbelichtet oder unterbelichtet. Deshalb kann in einem Vorverarbeitungsschritt das Bild zunächst verbessert werden (siehe Abschnitt 3.2.2). Auf dem verbesserten Bild können dann Kanten bestimmt (siehe Abschnitt 3.2.3) oder andere Merkmale extrahiert werden, die für die Objekterkennung (siehe Abschnitt 3.3) benötigt werden. Folglich muss im Rahmen der Projektgruppe das Bild soweit verarbeitet werden, dass die für die Erkennung des Fahrzeugs relevanten Merkmale, wie beispielsweise die Farbe und Form, in dem Bild leichter zu identifizieren sind.

3.2.1. Definition

Bei Rastergrafiken wird das Bild durch eine Matrix von Bildpunkten, auch Pixel (vom englischen *picture elements* abgeleitet) genannt, repräsentiert. Die Größe der Matrix wird durch die Faktoren Auflösung und Komponentenanzahl pro Pixel bestimmt. Die Auflösung bestimmt die Anzahl an Spalten und Zeilen in einer Matrix.

Die Anzahl d der Kanäle definiert die Dimension der Matrix. Bekannte Beispiele sind unter anderem Schwarz-Weiß-Bilder, Grauwertbilder und Farbbilder. Schwarz-Weiß-Bilder enthalten nur einen ($d = 1$) Kanal mit einer Breite von einem Bit. Es wird auch von 1 Bit pro Pixel (BPP) gesprochen. Die Werte geben an, ob der entsprechende Pixel schwarz oder weiß ist. Grauwertbilder enthalten einen ($d = 1$) Kanal mit einer Breite von 8 Bit, der in Grauwertabstufungen angibt, wie hell ein Pixel ist. Dabei steht 255 für weiß und 0 für schwarz.[11]

Weiterhin gibt es Farbbilder, welche meist 3 Kanäle ($d = 3$), mit je 8 Bit enthalten. Jedes Pixel wird folglich durch 24 Bit repräsentiert. Welche Bedeutung die einzelnen Kanäle haben, hängt vom *Farbraum* ab. Ein häufig verwendeter Farbraum ist der RGB Farbraum. In ihm steht je ein Kanal für die roten, grünen und blauen Farbanteile im Bild. Damit sind etwas mehr als 16,7 Millionen Farben darstellbar. Zusätzlich gibt es noch andere Farbräume, wie zum Beispiel HSV (Farbwert *Hue*, Farbsättigung *Saturation*, Hellwert *Value*) oder YUV (Luminanz (*Yellow*) und Chrominanz (*Under* und *Violet*); im analogen Fernsehen verwendet). Vor allem in der Bildverarbeitung werden manchmal auch Kanäle mit einer Breite von 10 oder 12 Bit verwendet.

3.2.2. Bildverbesserung im Ortsraum

Die einfachste Art der Bildverbesserung basiert auf Punktoperatoren. Die Helligkeit (Intensität) eines Pixels im Ausgabebild ist dabei nur abhängig von der Intensität im Eingabebild.

Abbildung 3.8.: Bilder mit Histogrammen²

Bei den ortsunabhängigen Intensitätstransformationen werden zwei Pixel gleicher Helligkeit identisch transformiert, unabhängig von ihrer Lage im Bild. Häufig wird hierfür das *Histogramm* herangezogen.

Histogramm

Das Histogramm stellt die absoluten oder relativen Häufigkeiten der Helligkeitswerte in einem Bild dar. Dabei werden auf der x-Achse die möglichen Helligkeitswerte aufgetragen, die y-Achse stellt die absoluten oder relativen Häufigkeiten der jeweiligen Helligkeitswerte dar.

Anhand der Verteilung der Helligkeitswerte im Histogramm kann schnell ermittelt werden, ob ein Bild unterbelichtet (Häufung am linken Rand), überbelichtet (Häufung am rechten Rand) oder gut belichtet ist (gleichmäßige Verteilung). In Abbildung 3.8 sind drei unterschiedlich belichtete Bilder mit Histogrammen gezeigt, die dies verdeutlichen.

Ausblenden

Beim *Ausblenden* wird nur ein bestimmter Helligkeitsbereich des Bildes extrahiert. Es werden zwei Helligkeitswerte als Parameter erwartet. Der eine Parameter definiert die untere Grenze des Histogrammbereichs, der verwendet werden soll, der andere definiert die obere Grenze des Histogrammbereichs. Die Helligkeitswerte, die nicht in diesem Bereich liegen, werden aus dem Bild entfernt [12].

Histogrammglättung

Bei diesem Verfahren wird eine Glättung der kumulierten Helligkeitswerte im Histogramm durchgeführt. Dafür werden die Helligkeitswerte gleichmäßig über den gesamten Histogrammbereich verteilt [12]. Abbildung 3.9 zeigt exemplarisch die Anwendung einer Histogrammglättung.

²M. Groer. Histogramm und Belichtungssteuerung: <http://www.kleine-fotoschule.de>

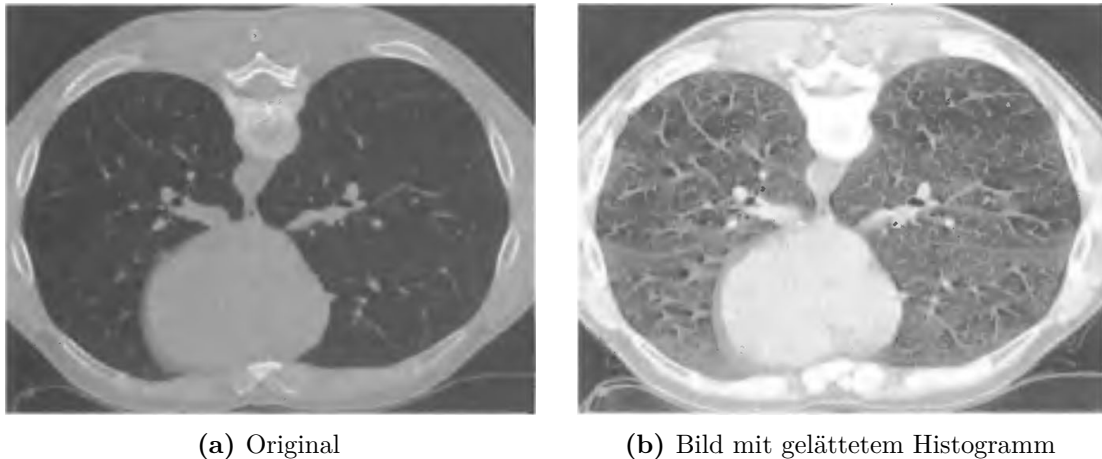


Abbildung 3.9.: Histogrammglättung [12]

3.2.3. Kantendetektion

Kanten sind dadurch gekennzeichnet, dass sich der Grauwert der Bildpunkte beim Überschreiten einer Kante signifikant verändert. Daher können Kanten über Maximal- oder Minimalwerte der Ableitung der Grauwerte detektiert werden. Es existieren verschiedene Filter – z.B. dem Laplace-Filter, welcher im Folgenden erklärt wird –, die über eine diskrete Faltung auf ein Bild angewendet werden können und deren Resultat ein Gradientenbild ist, auf dem Kanten einen unterschiedlichen Grauwert besitzen als flächige Bereiche des Originalbilds.

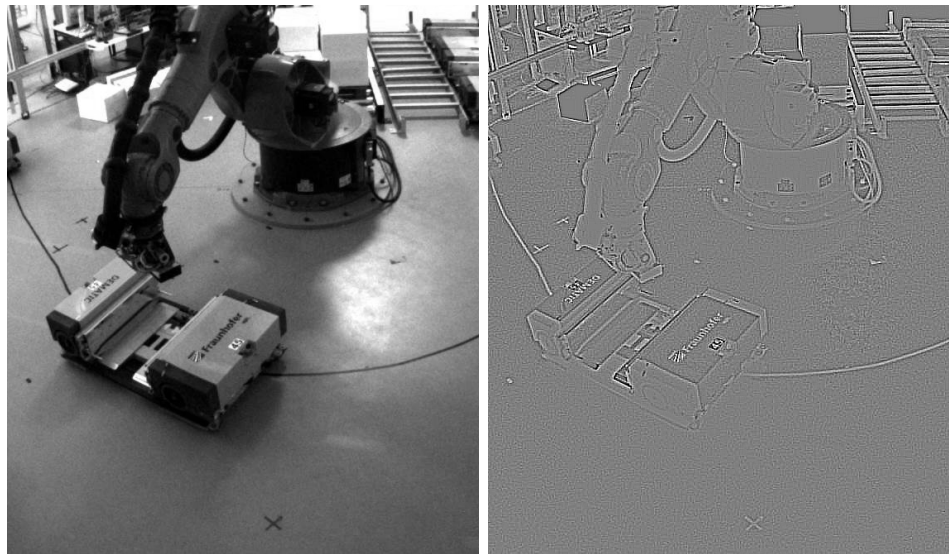
Laplace-Filter

Der Laplace-Filter [13] berechnet nicht die Ableitungen erster Ordnung für eine Kantenerkennung, sondern erzeugt ein Gradientenbild der Ableitungen *zweiter* Ordnung. Kanten befinden sich dementsprechend nicht an Maximalstellen, sondern an Stellen mit einem Vorzeichenwechsel des Gradientenbilds. Abbildung 3.10 zeigt das Ergebnis einer Anwendung dieses Filters. Mittleres Grau entspricht dabei einem Filterergebnis von null, Pixel mit negativen Werten sind schwarz markiert, während weiße Pixel positive Werte darstellen.

Im diskreten Fall kann das Ergebnis über verschiedene Faltungsmasken berechnet werden, wobei eine vergleichsweise reduzierte Anisotropie – also eine gleichmäßige Erkennung von Kanten unabhängig von dem Winkel, in dem sie verlaufen – durch eine Faltung mit der Maske

$$\frac{1}{4} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (3.11)$$

gewährleistet ist [13]. Durch eine Faltungsmaske wird die Nachbarschaft, also die umliegenden Bildpunkte, eines bestimmten Bildpunktes erfasst und zur Berechnung eines neuen Wertes mit entsprechender Gewichtung gemäß der Einträge in der Faltungsmaske durchgeführt.



(a) Originalbild

(b) Anwendung des Laplace-Filters

Abbildung 3.10.: Vergleich eines Bildes vor und nach Anwendung des Laplace-Filters

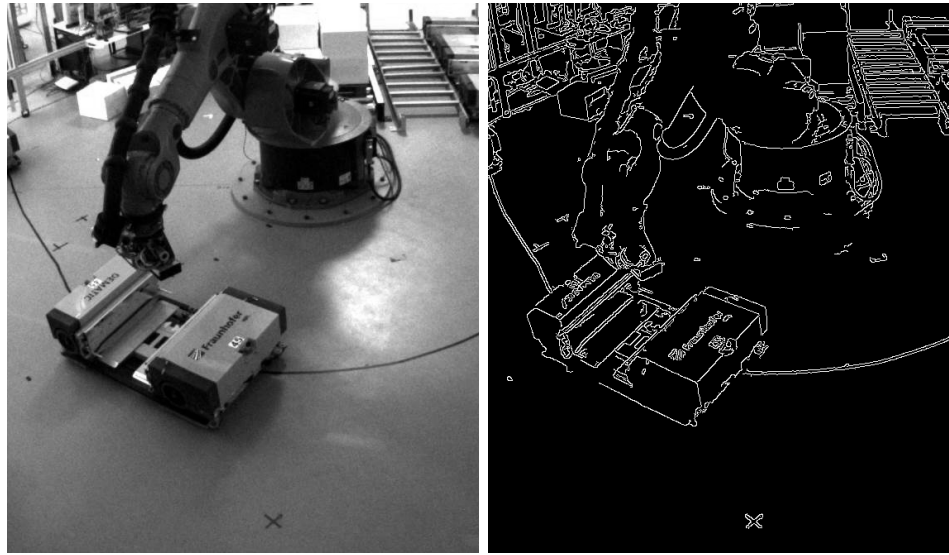
Canny-Kantendetektion

Wie Abbildung 3.10 zeigt, kann aus einem mit einem Ableitungsfiler berechneten Bild nicht auf algorithmisch einfache Weise entschieden werden, ob in einem bestimmten Pixel eine Kante verläuft oder nicht. Oft sind Kanten mehrere Pixel breit und das Bildrauschen oder andere Störungen führen zu uneindeutigen Ergebnissen. Wünschenswert ist daher eine Binärmaske, die genau angibt, welche Pixel zu einer Kante gehören.

Der Canny-Algorithmus [14] erzeugt über mehrere Schritte ein solches binäres Kantenbild, in dem jede detektierte Kante mit einer Breite von einem Pixel repräsentiert wird. Er wurde hinsichtlich dreier Kriterien optimiert:

- Es sollen alle im Bild vorhandenen Kanten erkannt werden. Gleichzeitig soll die Erkennung unechter Kanten, die durch Störungen oder Farbunterschiede der Flächen verursacht werden, vermieden werden.
- Die Position, an der die Kante im Kantenbild dargestellt wird, soll mit der Position der Kante im Originalbild übereinstimmen.
- Jede Kante soll nur „eine Antwort“ liefern, also beispielsweise nicht im Kantenbild zu zwei parallel verlaufenden Kanten führen.

Abbildung 3.11b zeigt das bereits aus Abbildung 3.10 bekannte Bild nach einer Anwendung des Canny-Algorithmus. Der Algorithmus glättet zunächst das Bild mit Hilfe eines Gauß-Filters. Im nächsten Schritt wird durch Anwendung des *Sobel*-Filters, einzeln in horizontaler und vertikaler Richtung, die Änderung der Grauwerte in x - und y -Richtung an jeder Stelle des Bildes bestimmt. Aus diesen beiden Werten lässt sich nun der Gradientenvektor mit Betrag und Richtung für jedes Pixel berechnen. Die Richtung des Gradienten wird dabei auf 45° gerundet, so dass Kanten entweder als waagrecht, senkrecht oder diagonal verlaufend gesehen werden.



(a) Originalbild

(b) Anwendung des Canny-Algorithmus

Abbildung 3.11.: Vergleich eines Bildes vor und nach Anwendung des Canny-Algorithmus

Im Anschluss werden alle Pixel entfernt, bei denen mindestens eines der in Gradientenrichtung – also senkrecht zur Kante – benachbarten Pixel einen höheren Gradientenbetrag besitzt. Dadurch werden Kanten, die eine Breite von mehreren Pixeln besitzen, auf eine Breite von genau einem Pixel reduziert. Um endgültig binär festzulegen, welche Pixel zu einer Kante gehören, werden zwei Schwellwerte festgelegt. Übersteigt der Gradientenbetrag eines Pixels den größeren Schwellwert, so handelt es sich um ein Kantenpixel. Unterschreitet er den kleineren Schwellwert, so handelt es sich nicht um ein Kantenpixel. Pixel, deren Gradientenbeträge zwischen beiden Schwellwerten liegen, werden nur zu den Kantenpixeln hinzugefügt, wenn sie zu einem bereits festgelegten Kantenpixel benachbart sind. Der letzte Schritt wird wiederholt, bis alle unsicheren Pixel, die an Kantenpixel angrenzen, in die Kantenpixelmenge aufgenommen worden sind. Verbleibende unsichere Pixel gehören nicht zu einer Kante.

Im Kontext der Projektgruppe wurde aufgrund der einfach Form des Fahrzeugs zunächst ein kantenbasiertes Verfahren in Erwägung gezogen. Es konnte jedoch keine zuverlässige Methode gefunden werden, um die zum Fahrzeug gehörenden Kanten im Gradientenbild zu finden. Außerdem wurden bei schlechten Lichtverhältnissen manche Kanten nicht als solche erkannt. Einige Flächen des Fahrzeugs waren dadurch nicht abgeschlossen und konnten deshalb nicht zu den einzelnen Seiten des Fahrzeugs zugeordnet werden.

3.3. Objekterkennung

Zur Bestimmung der Position des Fahrzeuges im Raum muss zunächst die Position des Fahrzeuges im Bild bestimmt werden. Dazu muss das Fahrzeug als solches im Bild gefunden

und identifiziert werden. In diesem Abschnitt werden verschiedene Verfahren zur Erkennung eines Objektes im Bild vorgestellt.

3.3.1. Segmentierung

Die Segmentierung eines Bildes verfolgt das Ziel, mehrere Teilbilder zu identifizieren, die jeweils bedeutungsvolle Bereiche im Bild beschreiben [12]. Für die Projektgruppe ist dieses Thema von Bedeutung, da für die Bestimmung der aktuellen Position des zu entladenden FTF Kamerabilder ausgewertet werden müssen. Bevor einzelne Objekte im Bild z.B. als Teil des Roboterarms oder Transportfahrzeug klassifiziert werden können, müssen zunächst durch Segmentierung die Regionen des Bildes herausgestellt werden, in denen ein Objekt sichtbar ist.

Als *vollständige Segmentierung* (complete segmentation) [12] eines als Menge von Bildpunkten repräsentierten Bildes R wird eine endliche Menge von *Regionen* R_1, R_2, \dots, R_S mit den Eigenschaften

$$R = \bigcup_{i=1}^S R_i \quad (3.12)$$

$$R_i \cap R_j = \emptyset, \quad i \neq j \quad (3.13)$$

bezeichnet. Es muss also jeder Bildpunkt von R genau einer der segmentierten Regionen zugeordnet sein, das heißt die Regionen überschneiden sich nicht (siehe Gleichung 3.13), decken jedoch gemeinsam R komplett ab (siehe Gleichung 3.12).

Eine *partielle Segmentierung* (partial segmentation) [12] bestimmt Regionen so, dass diese homogen bezüglich einer festzulegenden Eigenschaft sind. Mögliche Eigenschaften sind unter anderem die Helligkeit, Farbe oder Textur der Bildpunkte in einer Region. Eine solche partielle Segmentierung darf Regionen aufweisen, die sich überlappen.

Segmentierungsverfahren lassen sich in verschiedene Kategorien einordnen. Oft wird unterschieden nach *kantenorientierten* Methoden, die ein Kantenbild (siehe Abschnitt 3.2.3) als Grundlage nutzen, und *regionenorientierten* Methoden, die versuchen, homogene Flächen im Bild zu identifizieren. Es existieren außerdem Verfahren, die auf *globalen Informationen* (global knowledge) über das Bild basieren [12]. Es können außerdem neben kanten- und regionenorientierten auch einige Methoden als *pixelorientiert* oder *modellorientiert* kategorisiert werden [13]. Zum Beispiel kann eine Einordnung alleine aufgrund der Pixelhelligkeit getroffen werden. Ein Vorgehen wird hingegen als modellorientiert beschrieben, wenn dafür Informationen über die „exakte Form der im Bild enthaltenen Objekte“ benötigt werden.

Schwelwertverfahren

Im simpelsten Fall erfolgt die Segmentierung in zwei Teilbilder. Dabei wird jeder Bildpunkt \mathbf{b} einzeln betrachtet und der zugehörige Grauwert mit einem zuvor festgelegten, globalen *Schwelwert* c verglichen. Je nachdem, ob der Grauwert eines Pixels größer oder kleiner als c ist, erfolgt die Zuordnung von \mathbf{b} in das „Objekt“-Segment oder in das „Hintergrund“-Segment [12]. Das Vorgehen kann als pixelorientiert [13] beschrieben werden.

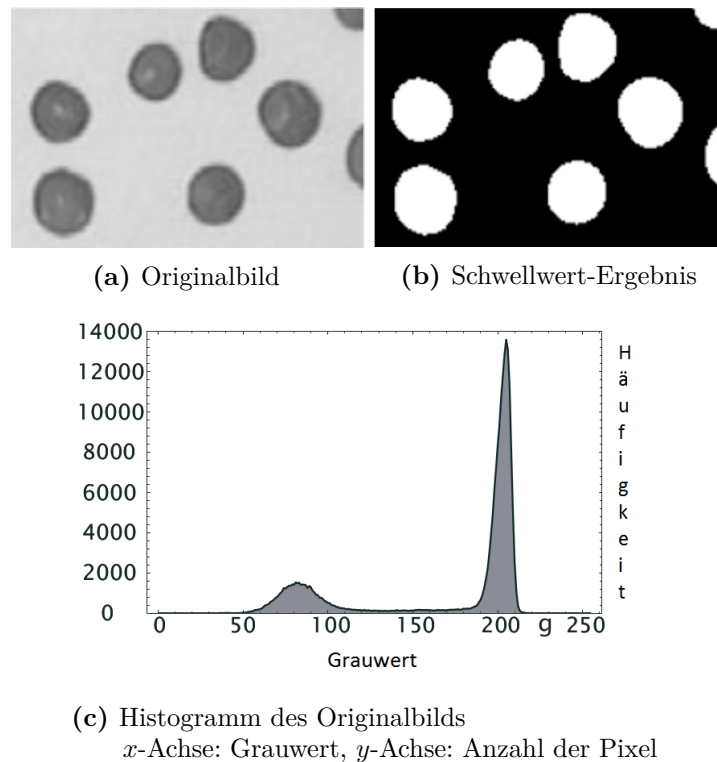


Abbildung 3.12.: Anwendung eines Schwellwertverfahrens mit globalem Schwellwert von 147 [13]

Ein guter globaler Schwellwert lässt sich finden, wenn das Histogramm der Grauwerte des Bildes bimodal verteilt ist [13] – wenn also nahezu alle Grauwerte der herauszustellenden Objekte grundsätzlich größer bzw. kleiner als nahezu alle Grauwerte der Hintergrundpixel sind. Lediglich an den Objekträndern ist eine gewisse Vermischung der Grauwerte des Vorder- und Hintergrunds weniger problematisch und auch bei Bildaufnahmen schwer zu vermeiden. Abbildung 3.12 zeigt eine beispielhafte Anwendung des Verfahrens. Aufgrund der Bimodalität des Histogramms (siehe Abbildung 3.12c) ist mit einem sinnvoll in der Mitte beider Histogramm-Maxima gewählten Schwellwert von 147 ein gutes Ergebnis gewährleistet.

Falls hingegen Grauwerte des Hintergrunds auch im Bereich eines Objekts angenommen werden, kann kein geeigneter Schwellwert festgelegt werden. Dies kann unter realistischen Bedingungen beispielsweise durch inhomogene Hintergründe oder ungleichmäßige Beleuchtung des Bildes auftreten. Ebenso kann das Verfahren fehlschlagen, wenn Objekte mit unterschiedlichen Grauwerten im Bild sichtbar sind [13].

Zur Festlegung eines globalen Schwellwerts kann das *p-tile thresholding* [12] angewendet werden. Dabei wird das $\frac{1}{p}$ -Quantil ($p \in \mathbb{N}$) der Grauwerte des Bildes als Schwelle ausgewählt. Dies ist beispielsweise bei digitalisierten Dokumenten geeignet, wenn der Anteil der bedruckten Fläche grob bekannt ist. Eine andere Möglichkeit besteht darin, ein lokales Minimum der Histogrammwerte als Schwellwert zu wählen.

Erweiterte Schwellwertverfahren

Zu diesem einfachen Verfahren können Erweiterungen vorgenommen werden [11]. Statt des bisher beschriebenen globalen Schwellwerts kann auch ein als *lokal* bezeichneter Schwellwert genutzt werden. Hierbei wird die Zuordnung zu Vorder- oder Hintergrundsegment nicht nur anhand des Grauwerts eines Pixels getroffen, sondern es kann zusätzlich eine lokale Eigenschaft berücksichtigt werden, wie z.B. der mittlere Grauwert der Umgebung eines Pixels. Desweiteren kann der Schwellwert *adaptiv* gewählt werden, wodurch er in Abhängigkeit von den Koordinaten des gerade untersuchten Pixels festgelegt wird.

Regionenorientierte Verfahren

Die in diesem Abschnitt vorgestellten Verfahren verfolgen auf unterschiedliche Arten das Ziel, eine Segmentierung in Regionen vorzunehmen, bei der die Pixel jeder segmentierten Region bezüglich einer festgelegten Eigenschaft homogen sind. Es handelt sich daher um regionenorientierte Segmentierungstechniken.

Region Growing

Bei der Anwendung eines *Region Growing*-Verfahrens [11] wird zunächst eine Anzahl von Startpixeln (seed points) festgelegt, um anschließend Regionen mit einer gemeinsamen Eigenschaft um diese Startpixel herum „wachsen“ zu lassen.

Das Verfahren zur Auswahl der Startpixel ist dabei je nach Anwendung unterschiedlich. So kann beispielsweise bekannt sein, dass jede Region, die durch ein Segment repräsentiert werden soll, mindestens ein Pixel mit einem Grauwert von 255 besitzt [11]. In diesem Fall können alle Pixel des Bilds, die diesen Grauwert besitzen, als Startpixel ausgewählt werden. Falls dabei mehrere Startpixel direkt nebeneinander liegen, werden sie zu einer Startregion (seed region) zusammengefasst. Eine vergleichbare Situation tritt auf, wenn die Projektgruppe ein FTF in einem erfassten Bild erkennen möchte. Da die Fahrzeuge gelb sind, könnten Startpixel oder -regionen durch eine Betrachtung des Farbwerts der Pixel festgelegt werden.

Auch bei der „Wachstumsphase“ der Regionen können unterschiedliche Techniken zum Einsatz kommen. Beispielsweise können statische Kriterien wie die maximale Intensitätsdifferenz zum Startpixel [11] für die Entscheidung, ob ein angrenzendes Pixel in eine Region aufgenommen werden soll, berücksichtigt werden. Der Wachstumsprozess endet, sobald keines der zu einer Region benachbarten Pixel mehr das Kriterium erfüllt. Die resultierende Segmentierung ist also nicht unbedingt vollständig.

Region Splitting und Merging

Eine weitere Möglichkeit, eine regionenorientierte und stets vollständige Segmentierung zu erhalten, ergibt sich durch die Anwendung von *Split*- und *Merge*-Schritten [12] auf das zu segmentierende Bild.

Es wird zunächst von nur einer Region ausgegangen, die das komplette Bild umfasst. Durch einen Split-Schritt wird eine Region ausgewählt, deren Pixel nicht die geforderte Homoge-

nitätseigenschaft erfüllen, und in vier disjunkte, quadratische Unterregionen zerteilt. Um eine Übersegmentierung zu vermeiden, werden durch einen Merge-Schritt zwei benachbarte Regionen, deren Pixel zusammengenommen das Homogenitätskriterium erfüllen, zu einer gemeinsamen Region verschmolzen. Die Segmentierung ist abgeschlossen, wenn keiner der beiden Schritte mehr angewendet werden kann.

3.3.2. Template-Matching

Das Template-Matching versucht die Aufgabe zu lösen, ein vorgegebenes *Muster* (Template) in einem Bild zu finden. Die Schwierigkeit besteht darin, dass das Muster durch Rauschen, einen anderen Blickwinkel, eine andere Beleuchtung oder auch einen anderen Sensor, der für die Bilderzeugung genutzt wird, im Bild anders aussehen kann [15].

Die algorithmisch einfachste Lösung besteht darin, jede mögliche Position im Bild auf die Abweichung der Eigenschaften der dort vorhandenen Pixel zum Muster zu überprüfen. Das Vorgehen bietet jedoch weder große Effizienz, noch werden Abweichungen des vorgegebenen Musters von der Darstellung im Bild berücksichtigt.

Fortgeschrittenere Lösungsansätze nutzen *Pyramiden* für eine effizientere Lösung des Problems. Eine Bild-Pyramide besteht aus dem Bild selbst und mehreren Kopien des Bilds in einer jeweils kleineren Auflösung. Werden die Bilder gedanklich der Größe nach sortiert übereinander angeordnet, so entsteht eine Pyramide mit rechteckiger Grundfläche. Ein allgemeiner Ansatz [16] für die Suche in Pyramiden ergibt sich durch rekursiven Abstieg. Dabei wird zunächst in einer „hohen“ – also gering aufgelösten – Ebene der Pyramide nach dem Muster gesucht. Viersprechende Regionen dieser Ebene werden dann auf der darunter liegenden Pyramidenebene mit feinerer Auflösung genauer untersucht. Dieser Vorgang wird rekursiv bis zur Ebene mit höchster Auflösung fortgesetzt.

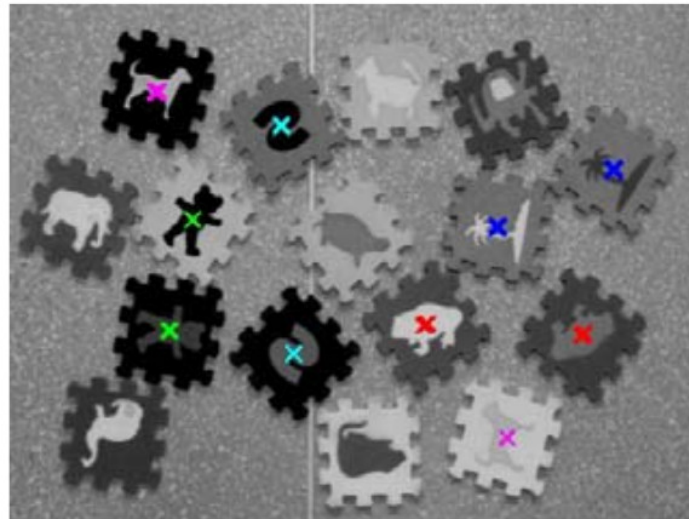
Ein komplexer Algorithmus für das Template-Matching-Problem ist der Ciratefi-Algorithmus [17]. Er erkennt das Template auch dann, wenn es im Bild rotiert, skaliert oder translatiert, mit anderer Helligkeit oder einem anderen Kontrast vorhanden ist. Ein Ergebnis dieser Matching-Methode ist in Abbildung 3.13 dargestellt. Hier wurden die gefundenen Übereinstimmungen der fünf vorgegebenen Templates durch farbige Kreuze markiert. Dieser Algorithmus besteht aus drei Filtern, die nacheinander angewendet werden und jeweils Pixel, die für eine Übereinstimmung mit dem Muster nicht mehr in Frage kommen, aussortieren:

Cifi Der *Circular Sampling*-Filter „Cifi“ berechnet den mittleren Grauwert der Pixel, die auf verschiedenen Kreisen mit vorher festgelegten Radien um den Mittelpunkt des Musters liegen und nutzt diese mittleren Grauwerte als Sample. Es werden alle Pixel des Bildes ausgefiltert, bei denen ein Circular Sampling mit verschiedenen Skalierungsfaktoren signifikante Unterschiede im Vergleich mit dem Sampling des Musters ergeben. Für nicht ausgefilterte Pixel wird die passende Skalierung des Musters vermerkt.

Rafi Der *Radial Sampling*-Filter „Rafi“ nutzt radial vom Mittelpunkt des Musters ausgehende sinusförmige Linien, die bis zum Verlauf des größten Radius des Cifi-Filters reichen. Der mittlere Grauwert der Pixel auf diesen Linien wird wieder genutzt, um durch einen Vergleich der Grauwerte mit verschiedenen Rotationen des Bilds weitere Pixel auszufiltern. Dabei wird die passende Rotation des Musters für jedes noch verbleibende Pixel vermerkt.



(a) Vorgegebene Templates



(b) Gefundene Positionen der Templates im Bild

Abbildung 3.13.: Resultat des Ciratefi-Algorithmus [17]

Tefi Der *Template Matching*-Filter „Tefi“ untersucht die nach Rafi verbliebenen Pixel mit Hilfe der ermittelten Rotation und Skalierung auf tatsächliche Übereinstimmung mit dem Muster. Dafür wird ein helligkeits- und kontrastinvarianter Matchingalgorithmus verwendet.

Im Kontext der Projektgruppe hat sich dieses Verfahren als ungeeignet erwiesen, da die Objekte im Bild der Kamera räumlich verzerrt abgebildet werden.

4. 3D-Lokalisierung und Orientierungsbestimmung von Objekten

Die 3D-Detektion von Objekten teilt sich in unterschiedliche Phasen auf. Im ersten Schritt muss eine 3D-Aufnahme der Szenerie mit Hilfe einer Tiefenkamera aufgenommen werden. Die dabei entstehenden Daten werden dann mit unterschiedlichen Verfahren vorverarbeitet, damit die anschließend folgende Objekterkennung effizienter durchgeführt werden kann. Ist die Lokalisierung des Objektes abgeschlossen, so kann dann die Orientierung des Objektes bestimmt werden.

Im ersten Abschnitt dieses Kapitels (siehe Kapitel 4.1) werden unterschiedliche Techniken zur Aufnahme von 3D-Informationen vorgestellt. Anschließend werden Verfahren zur effizienten Speicherung, Reduktion und Bilderkennung von Punktwolken als Repräsentant von 3D-Informationen erläutert (siehe Kapitel 4.2.4). Abschließend wird näher auf die Objekterkennung mittels 3D-Informationen eingegangen (siehe Kapitel 4.3.2).

4.1. Bildaufnahme

Die Ansätze für das Ermitteln von 3D-Informationen eines Objekts lassen sich in drei Bereiche einteilen. Zum einen können bildbasierte Verfahren verwendet werden, welche aus 2D-Bildern (es werden mindestens zwei Bilder benötigt) die bei der Aufnahme verlorengegangenen 3D-Informationen rekonstruieren. Ein Beispiel hier für ist die Stereo-Vision (siehe Kapitel 4.1). Hierbei wird mit Hilfe der Epipolargeometrie aus zwei unterschiedlichen Ansichten einer Szene die Tiefeninformation rekonstruiert. Die zweite Möglichkeit bilden Verfahren, die die Zeit die ein Lichtstrahl zu einem Objekt und zurück benötigt messen. Diese lichtwegbasierten Verfahren benötigten spezielle Hardware, auf die in Kapitel 4.1 genauer eingegangen wird. Als letztes gibt es die Möglichkeit strukturiertes Licht zur Bestimmung der 3D-Informationen zu verwenden (siehe Kapitel 4.1). Dabei wird die Szene mit einem Lichtmuster beleuchtet, um aus der Verzerrung dieses Muster 3D-Informationen bestimmen zu können. In diesem Abschnitt werden Beispiele der verschiedenen Verfahren zur Ermittlung von 3D-Informationen erläutert und ihre Einsetzbarkeit in der Projektgruppe betrachtet

Stereo-Vision

Die Stereo-Vision bietet eine Möglichkeit zur Ermittlung von 3D-Informationen aus 2D-Bildinformationen. Bei der Projektion einer Szene auf eine Bildebene gehen Tiefeninformation verloren. Daher ist es nicht möglich, aus einer einzigen Aufnahme allein eine 3D-Rekonstruktion durchzuführen. Die Stereo-Rekonstruktion bedient sich deshalb mehrerer Aufnahmen einer

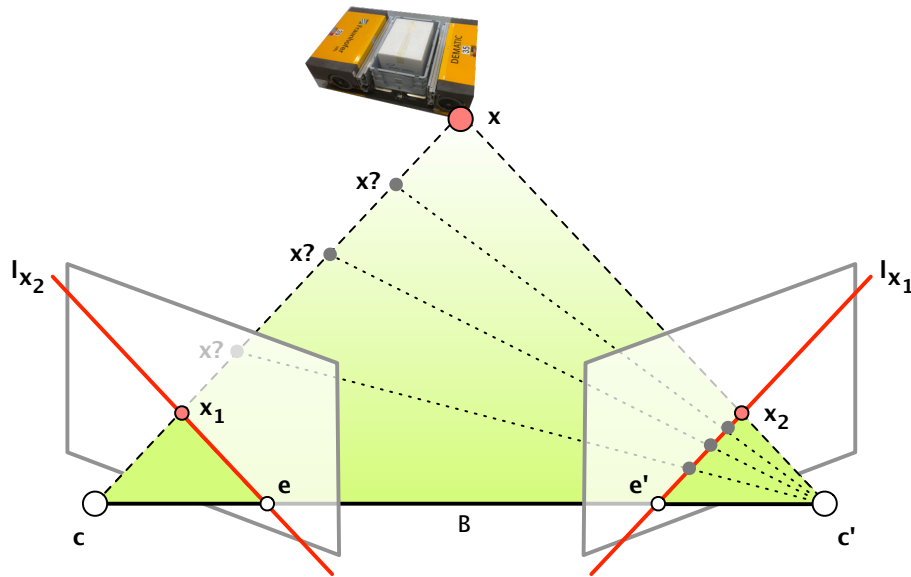


Abbildung 4.1.: Schematische Darstellung der Epipolargeometrie

Szene aus unterschiedlichen Perspektiven und nutzt diese, um die Tiefeninformationen wiederherzustellen.

Ausgehend von den beiden Aufnahmen einer Szene, werden zunächst jeweils zwei Punkte in den beiden Aufnahmen ermittelt, welche vom selben Punkt im Raum auf die Bildebene projiziert wurden. Solche Punktpaare werden *korrespondierende Punkte* genannt und die Ermittlung solcher nennt sich *Korrespondenzanalyse*. Die Korrespondenzanalyse besteht typischerweise aus den folgenden drei Schritten [7]:

- Auffinden interessanter Punkte, in beiden Ansichten getrennt (Punktmerkmale; Gradientenverfahren etc.)
- Bewertung potentieller Korrespondenzpaare (z.B.: Block Matching-Maße; Scale-Invariant Feature Transform (SIFT)-Deskriptoren)
- Ermittlung der Korrespondenzpaare (z.B.: Cross-Validierung)

Zur Vereinfachung der Korrespondenzanalyse kann die *Epipolargeometrie* [7] genutzt werden, welche den Suchbereich zur Ermittlung korrespondierender Punkte erheblich einschränkt. Abbildung 4.1 zeigt schematisch die Epipolargeometrie zweier Ansichten einer Szene. Dargestellt sind zwei Bildebenen mit den optischen Zentren c und c' , welche dieselbe Szene darstellen (FTF im Hintergrund). Die beiden optischen Zentren sind durch die Basislinie B verbunden, welche beide Bildebenen in den *Epipolen* e und e' schneidet. Die *Epipolareinschränkung* besagt nun, dass sich ein zum Punkt x_1 in der ersten Bildebene korrespondierender Punkt x_2 auf einer bestimmten Linie l_{x_1} in der zweiten Bildebene befinden muss, der *Epipolarlinie* von x_1 . Damit ist der Suchraum für den korrespondierenden Punkt zu x_1 auf eine einzige definierte Linie beschränkt. Dieselbe Beziehung gilt analog auch für die Suche des korrespondierenden Punktes x_1 zum Punkt x_2 . Mathematisch gesehen wird diese Beziehung für alle Punkte der beiden

Ansichten durch die *Fundamentalmatrix* \mathbf{F} vollständig beschrieben. Für korrespondierende Punktpaare \mathbf{x} und \mathbf{x}' gilt die Beziehung

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0, \quad (4.1)$$

wobei $\mathbf{F} \mathbf{x}$ die Epipolarlinie von \mathbf{x} beschreibt.

Da die Fundamentalmatrix \mathbf{F} genau sieben Freiheitsgrade besitzt, kann sie mit mindestens sieben korrespondierenden Punktpaaren exakt berechnet werden. Die mindestens sieben benötigten Punktkorrespondenzen müssen ohne Suchraumeinschränkung vorher ermittelt werden. Ein sehr umfangreiches Review und Details über die verschiedenen Möglichkeiten zur Berechnung der Fundamentalmatrix findet sich in [18].

Nach Ermittlung korrespondierender Punktpaare kann eine *Stereotriangulierung* [19] durchgeführt werden, um aus den korrespondierenden Punkten die Tiefeninformation des entsprechenden Punktes im Raum zu ermitteln. Die Stereotriangulierung basiert auf der Rückprojektion zweier korrespondierender Punkte \mathbf{x}_1 und \mathbf{x}_2 in den Raum. Die Rückprojektion kann mit den Projektionsmatrizen der Kameras durchgeführt werden, welche aus der Fundamentalmatrix extrahiert werden können. Punkte auf einer Bildebene entsprechen Strahlen im Raum. Dies lässt sich gut anhand der bereits vorgestellten Abbildung 4.1 einsehen. Der Bildpunkt \mathbf{x}_1 könnte potentiell von jedem Punkt auf dem Strahl, der durch das Kamerazentrum C und dem Punkt \mathbf{x} induziert wird, projiziert worden sein. Ziel ist es nun, den genauen Punkt auf dem Strahl ausfindig zu machen, der dem projizierten Punkt \mathbf{x}_1 entspricht. Aus einer Aufnahme allein ist dies nicht möglich. Hier kommt der korrespondierende Punkt \mathbf{x}_2 zu \mathbf{x}_1 ins Spiel. Da beide Punkte Projektionen des 3D-Punktes \mathbf{x} sind, sollten sich ihre rückprojizierten Strahlen in genau dem Punkt \mathbf{x} schneiden. Theoretisch genügt es also, die beiden korrespondierenden Punkte auf Strahlen rückzuprojizieren und die Strahlen zu schneiden, um die Tiefeninformationen zu rekonstruieren.

In der Realität schneiden sich die beiden Strahlen auf Grund von Ungenauigkeiten jedoch nicht unbedingt, da die Bildinformationen nicht ideal aufgelöst sind oder die Schätzung der korrespondierenden Punktpaare ungenau sein kann. Eine Lösung kann daher approximativ berechnet werden, indem zum Beispiel der 3D-Punkt mit dem minimalsten Abstand zu beiden Strahlen als Schnittpunkt verwendet wird. Ein weiteres Problem bei den Verfahren stellt die projektive Verzerrung durch die Kameramatrizen dar, denn die Projektion unterliegt einer projektiven Transformation, in welcher Distanzen, Orthogonalitäten etc. nicht invariant sind. Zur Verfolgung der Paketposition könnten zwar zwei Kameras am Sauggreifer angebracht werden, die oben genannten Ungenauigkeiten bilden jedoch den Hauptgrund, warum dieses Verfahren nicht in der gegebenen Umgebung von der Projektgruppe eingesetzt wird (die Verfahren werden in Kapitel 7.3 genauer erläutert). Der vom Sauggreifer aus sichtbare Bildausschnitt besteht zu einem großen Teil aus dem einheitlichen Boden auf dem das Transportsystem fährt (die Position der Kamera am Sauggreifer ist auf Abbildung 7.8 zu erkennen), was eine Korrespondenzanalyse stark erschwert. Da der Sauggreifer selbst keinen Drucksensor beinhaltet, kann eine fehlerhafte Abstandsangabe im besten Fall dazu führen, dass der Sauggreifer das Paket nicht greifen kann, wenn die Position höher eingeschätzt wurde als sie tatsächlich ist. Im schlechtesten Fall könnte die Position tiefer geschätzt werden, so dass der Greifer das Paket beschädigen könnte. Da jedoch die Unversehrtheit des Pakets gewährleistet sein muss, wurde der Ansatz der Stereo-Vision in der Projektgruppe nicht weiter verfolgt.

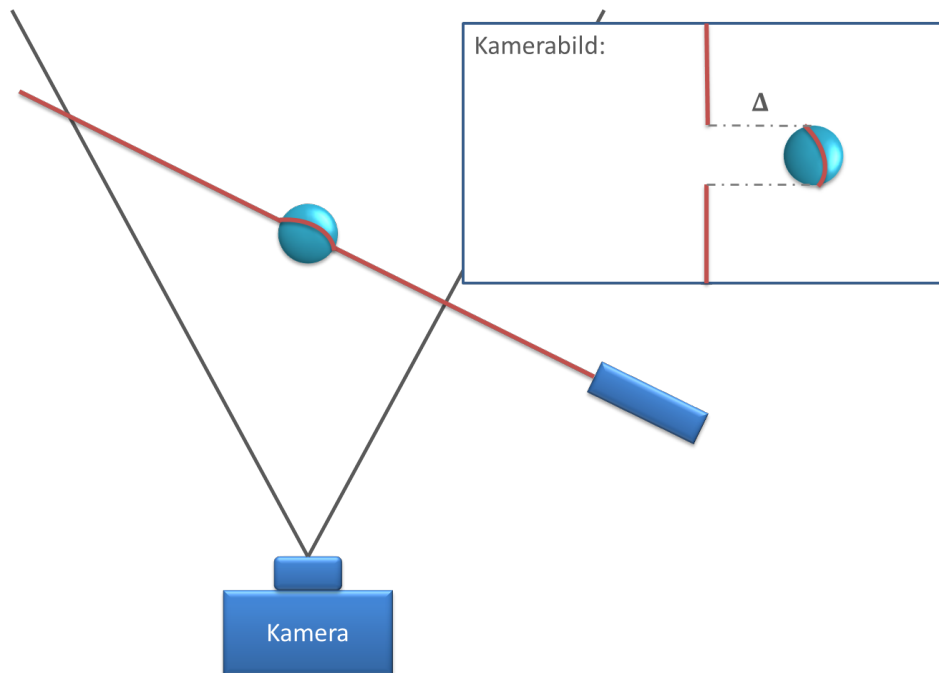


Abbildung 4.2.: Durch die Position des Lichtstreifens im Kamerabild kann die Entfernung des Objektes zur Kamera berechnet werden. Aus dem Abstand Δ ergibt sich der Abstand zwischen Objekt und Hintergrund.

Strukturiertes Licht

Ähnlich wie die Stereovision basiert auch die Verwendung von strukturiertem Licht zur Gewinnung von Tiefeninformationen auf Triangulation. Allerdings wird hierbei nur eine Kamera verwendet. Die Rolle der zweiten Kamera übernimmt eine Lichtquelle, welche ein Lichtmuster auf die Szene projiziert. Die Verzerrung bzw. die Position dieses Muster im Kamerabild bildet dann die Grundlage für die Triangulation [19], aus der Tiefeninformationen entnommen werden. Abbildung 4.2 zeigt schematisch wie ein einzelner Lichtstreifen zur Bestimmung von Tiefeninformationen eines Objektes verwendet werden kann. Bei bekannter Position der Lichtquelle lässt sich aus der Position des Streifens im Bild die genaue Tiefe des Objektes ermitteln. Um das ganze Objekt zu erfassen, reicht jedoch ein Lichtstreifen nicht aus. Daher verwenden Tiefenkameras die mit strukturiertem Licht arbeiten Lichtmuster, welche sich über das gesamte Bild erstrecken. Abbildung 4.3 zeigt ein solches Muster. Hier wird ein Streifenmuster verwendet, so dass durch die Breite der Streifen sowie die Farbe ein einzelner Streifen identifiziert werden kann. Die horizontale Position dieses Streifen gibt dann die Tiefe dieses Bildpunktes an.

Um strukturelle Fehler durch die Verwendung eines regelmäßigen Lichtmusters zu vermeiden, verwenden Kameras wie z.B. die Microsoft Kinect¹ und die ASUS Xtion (siehe Kapitel 7.3.1) ein pseudo zufälliges Punktmuster, welches in die Szene zu projizieren. Um einzelne Punkte identifizieren zu können, werden kleine Bildausschnitte betrachtet und mit dem Ausgangsmuster verglichen. Damit die Punkte nicht kleiner werden, je weiter sie von der Kamera entfernt sind,

¹<http://www.microsoft.com/en-us/kinectforwindows/>, Zuletzt aufgerufen: 15.3.2014

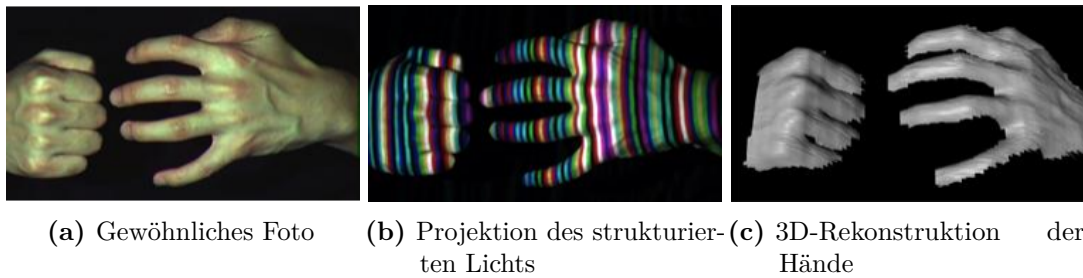


Abbildung 4.3.: Projektion eines strukturierten Streifenmusters auf Hände ²

wird ein Lichtkegel verwendet, der im gleichen Maße größer wird, indem das Sichtfeld eines Pixels sich ausdehnt. Dadurch ändert sich nur die Position des Lichtpunktes, jedoch nicht die Größe. Die Tiefe des Bildpunktes ergibt sich dann wiederum durch die horizontale Koordinate des Punktes.

Obwohl Kameras, die mit strukturiertem Licht arbeiten in der Regel günstiger sind, als z.B. PMD-Kameras (siehe Kapitel 4.1), sind sie auf Grund ihrer geringen Größe mobil einsetzbar und liefern bei Objekten, welche groß im Vergleich zum Durchschnittlichen Abstand zweier benachbarter Punkte im Punktmuster genaue Ergebnisse (siehe Evaluation in Kapitel 8.3). Allerdings ist diese Technologie nicht dafür geeignet sehr kleine bzw. dünne Objekte zu erkennen, da diese durch das Punktmuster nicht erfasst werden können. Im Rahmen der Projektgruppe spielt dies jedoch keine Rolle, da die Größe der Pakete ausreicht, so dass auf Grund der Genauigkeit, des geringen Preises und der Mobilität im Rahmen der Projektgruppe eine Tiefenkamera verwendet wird, welche mit strukturiertem Licht arbeitet.

PMD-Kameras

Eine weitere Möglichkeit ist der Einsatz von PMD-Kameras (*Photonic Mixing Device*-Kamera) [20]. Sie bilden eine Weiterentwicklung der ToF-Kameras (*Time of Flight*-Kamera). Bei ToF-Kameras wird die Zeit, die ausgesandtes Licht von der Kamera bis zum Objekt und wieder zurück benötigt, gemessen, um so Rückschlüsse über den Abstand des Objektes zur Kamera ziehen zu können. Da dies unter anderem sehr fremdlichtanfällig ist, verwendet eine PMD-Kamera zur Gewinnung der Abstandsinformationen die Phasenverschiebung des wiedereintreffenden Lichtes. Grundsätzlich bestehen PMD-Kameras aus den Hauptkomponenten *Emitter*, *Detektor* und *Steuerungselektronik*. Abbildung 4.4 zeigt, wie diese Komponenten miteinander zusammenhängen. Der Emitter leuchtet die Szenerie durch das Aussenden von moduliertem, zumeist infrarotem Licht aus. Nachdem diese Lichtwellen vom zu vermessenden Objekt reflektiert wurden, erreichen sie den Detektor. Dabei verschiebt sich die Phase des eintreffenden Lichtes je nach Abstand des Objektes. Aus dieser Phasenverschiebung können dann Rückschlüsse über den Abstand des Objektes zur Kamera gewonnen werden. Ein Vorteil dieses Verfahrens liegt darin, dass Fremdlicht direkt herausgefiltert werden kann. Die Korrelation des aus- und einfallenden Lichtes reicht aus, um Fremdlicht fast komplett zu unterdrücken, so dass die Kamera bei voller Sonneneinstrahlung betrieben werden kann. Allerdings führt das Ausnutzen der Phasenverschiebung dazu, dass nur ein beschränkter Abstandsbereich korrekt gemessen

²University of Utah: <http://www.sci.utah.edu/gerig/CS6320-S2012/Materials/CS6320-CV-S2012-StructuredLight.pdf>, Zuletzt aufgerufen: 15.3.2014

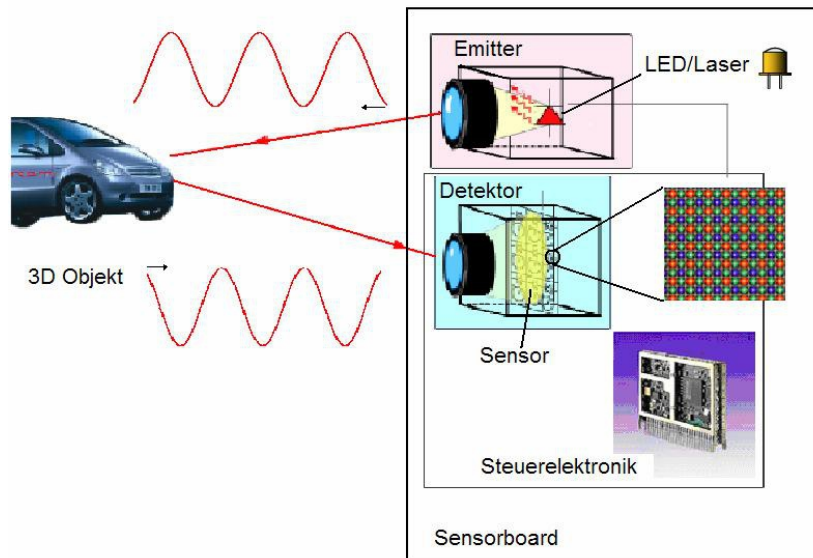


Abbildung 4.4.: Komponenten einer PMD-Kamera [20]

werden kann, da darüberhinaus Mehrdeutigkeiten entstehen können. Eine mögliche Lösung dieses Mehrdeutigkeitsproblems ist die Verwendung mehrerer Wellenlängen oder bestimmter Codewörter, die einen größeren Eindeutigkeitsbereich ermöglichen.

Für den praktischen Einsatz in der Projektgruppe eignen sich jedoch auch diese Kameras nicht. Eine geringe Auflösung der Kamera lässt zwar eine genaue Abstandsmessung zu, macht jedoch eine exakte Bestimmung der Orientierung des Paketes problematisch. Weiterhin sind die Kameras deutlich teurer als z.B. 3D-Sensoren die mit strukturiertem Licht arbeiten (siehe Abschnitt 4.1).

4.2. Punktwolken und Verarbeitung

Für die Erkennung von Objekten sind so genannte Punktwolken von zentraler Bedeutung. Diese durch eine Tiefenkamera erzeugten Daten können zum Erkennen von Objekten im Fokus der Kamera genutzt werden. Um eine Erkennung dieser Objekte zu vereinfachen, existieren Ansätze um die Rohdaten der Kamera vorzeitig zu verarbeiten und effizient zu speichern. In diesem Abschnitt werden unterschiedliche Ansätze zur Reduzierung, Repräsentation und Triangulierung erläutert, die sich zur graphischen Datenverarbeitung von Punktwolken eignen.

4.2.1. Punktwolken

Wie bereits im vorherigen Abschnitt erläutert, werden die Messungen als so genannte *Punktwolken* repräsentiert. Eine Punktwolke $\mathcal{P} \subseteq \mathbb{R}^k$ eine Menge von Punkten in einem k -dimensionalen Raum. In diesem Bericht wird nur der Fall $k = 3$ betrachtet, da nur dieser einen praktischen Nutzen für die Projektgruppe enthält.

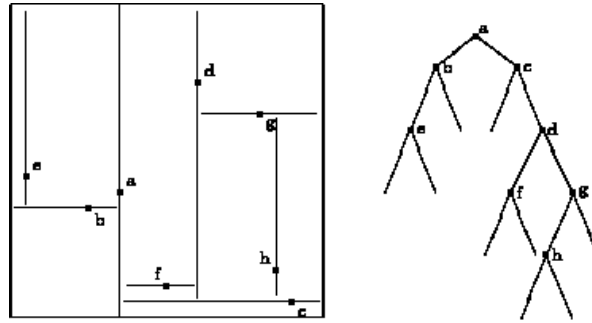


Abbildung 4.5.: kd -Baum mit einer Illustration der Punktmengenaufteilung³

Zusätzlich zu den Positionsinformationen gibt es Implementierungen von Punktwolken, die zu jedem Punkt auch Farbinformationen speichern. Diese werden in einem solchen Fall entsprechend dem RGB-Standard angegeben und können für jeden Punkt ausgelesen werden. Eine Punktwolke mit Farbinformationen ist also im drei-dimensionalen Fall eine Menge von sechs-dimensionalen Vektoren. Mit Hilfe dieser Informationen ist zum Beispiel die Implementierung eines Farbfilters möglich (siehe Kapitel 4.3.2).

Zum einfacheren Verständnis werden die folgenden Konzepte zunächst für zweidimensionale Punktmengen betrachtet. Anschließend wird für jedes Konzept erläutert, wie es sich auch auf einer dreidimensionalen Menge (also auf Punktwolken) anwenden lässt.

4.2.2. Repräsentation von Punktwolken

Punktwolken können über unterschiedliche Datenstrukturen verwaltet werden. Eine häufig genutzte Datenstruktur sind die so genannten kd -Bäume, die sowohl eine Punkt- als auch eine Bereichssuche ermöglichen, um so eine Datenreduktion möglichst effizient durchzuführen [21].

Sei eine zwei-dimensionale Punktmenge $\mathcal{P} \subset \mathbb{R}^2$ gegeben. Ein kd -Baum $T = (\mathcal{P}, E)$ ist ein binärer Baum mit folgenden Eigenschaften:

- Für jeden Knoten $\mathbf{u} \in \mathcal{P}$ mit gerader Höhe gilt: Die x -Koordinate jedes Knoten im linken Teilbaum von \mathbf{u} ist kleiner oder gleich und die eines jeden Knotens im rechten Teilbaum größer als die x -Koordinate von \mathbf{u} ;
- Für jeden Knoten $\mathbf{u}' \in \mathcal{P}$ mit ungerader Höhe gilt: Die y -Koordinate jedes Knoten im linken Teilbaum von \mathbf{u}' ist kleiner oder gleich und die eines jeden Knotens im rechten Teilbaum größer als die y -Koordinate von \mathbf{u}' ;

kd -Bäume sind eine Verallgemeinerung von binären Suchbäumen, in denen die schnelle Suche von eindimensionalen Schlüsseln ermöglicht wird. Die gegebene, zweidimensionale Punktmenge $\mathcal{P} \subset \mathbb{R}^2$ an einem Punkt $\mathbf{p} \in \mathcal{P}$ in zwei Mengen aufgeteilt. Da es sich aber um eine zweidimensionale Punktmenge handelt, wird in der Wurzel zunächst nach der x -Koordinate, und in jeder neuen Ebene jeweils abwechselnd nach y - und dann wieder nach x -Koordinate geteilt. Dieses rekursive Vorgehen endet dann, wenn die aufzuteilende Teilmenge nur noch aus einem Punkt besteht. Dieser wird dann die Wurzel eines Teilbaumes mit zwei leeren Kindern.

³TU München: <http://www.in.tum.de>

Um einen möglichst ausgeglichenen Baum zu erhalten, erfolgt die Wahl des Punktes \mathbf{p} , an dem geteilt wird, oft nicht willkürlich, sondern es wird der Median, also das $(\lceil \frac{n}{2} \rceil)$ -ste Element in der Ordnung der aufzuteilenden Menge M mit $|M| = n$ gewählt, die je nach Höhe des Baumes nach x - oder y -Koordinate sortiert ist. Somit ist sichergestellt, dass jeder Teilbaum eine ungefähr gleiche Anzahl an Elementen enthält.

Abbildung 4.5 zeigt ein Beispiel für einen zweidimensionalen kd -Baum für eine Punktmenge sowie eine Illustration der Mengenteilung der Punktmenge. Hier ist klar zu sehen, dass die Wurzel willkürlich gewählt wurde und dementsprechend ein Baum mit nicht optimaler Tiefe entsteht. In den Ebenen wird abwechselnd nach x - und y -Koordinate aufgeteilt.

Das Suchen eines Punktes entspricht nun genau der Suche in einem Binärbaum, mit der Ausnahme, dass bei einem kd -Baum von Ebene zu Ebene gespeichert werden muss, nach welcher Koordinate die Punktmenge aufgeteilt wurde. Damit ist die Suche eines Punktes in einem kd -Baum in $O(\log \mathcal{P})$ möglich.

Aber auch eine Bereichssuche ist in einem kd -Baum möglich. In einer zweidimensionalen Punktmenge kann eine solche Bereichssuche in $O(|\mathcal{P}|^{\frac{1}{2}} \cdot \mathcal{F})$ durchgeführt werden, wobei $\mathcal{F} \subseteq \mathcal{P}$ die Menge der ausgegebenen Punkte ist.

Das Konzept von kd -Bäumen ist auch auf Räume mit drei Dimensionen übertragbar. Der einzige Unterschied liegt in der Alternierung der Dimensionen in den Baumebenen. In einem dreidimensionalen kd -Baum wird also wiederholend zunächst nach x -, dann nach y - und schließlich nach z -Koordinate aufgeteilt. Mit einem kd -Baum lassen sich also effizient Punkte und sogar Bereiche in Punktwolken finden [22].

Während der Entwicklungsphase im Projekt hat sich gezeigt, dass die Objekterkennung über eine punkt-basierte Objekterkennung (dem RANSAC-Verfahren, siehe Kapitel 4.3.2) eine effiziente Methode zum Erkennen von Paketen darstellt. Mit dem RANSAC-Verfahren kann die Oberseite von Paketen, die im Rahmen des Projektes von Interesse ist, über das Erkennen von Schnittebenen lokalisiert werden. Da es sich bei diesem Verfahren um einen randomisierten Ansatz ohne den Gebrauch einer Bereichssuche handelt, ist eine Erkennung des Paketes somit auch ohne eine Suche in der Punktmenge effizient möglich. Die Verwendung eines kd -Baumes als Datenrepräsentation hat sich für das Projekt also als nicht sinnvoll erwiesen und wird deshalb nicht genutzt.

4.2.3. Entrauschen und Reduktion

Bei einer Punktwolkenaufnahme einer Szene kann ein sogenanntes *Rauschen* entstehen. Im Falle einer Punktwolke besteht ein Rauschen aus so genannten *Fragmenten*, das heißt Punkte der Punktwolke, die in einer Tiefenbildaufnahme zum Beispiel durch Messfehler entstanden sind und nicht zur eigentlichen Szene gehören oder das Abbild der Szene verzerren. Sei $\mathcal{P}^* \subset \mathbb{R}$ die Punktwolke, die die aufgenommene Szene vollständig und rauschfrei darstellt. Rauschpunkte $\mathbf{p} \in \mathcal{P}$ weichen also von dem korrespondierendem Punkt \mathbf{p}^* in der rauschfreien Punktwolke ab: $\mathbf{p}^* - \mathbf{p} = \mathbf{0}$. Das Ziel der *Rauschentfernung* ist es nun, solche Fragmente zu finden und diese schließlich zu entfernen.

Zusätzlich zum Entfernen des Rauschens ist es wichtig, eine Punktwolke zu *reduzieren*. Punktwolken enthalten im Allgemeinen viele Punkte, deren Koordinaten im Raum einen sehr geringen Abstand besitzen, um einen möglichst hohen Detailgrad der Szene darzustellen. Solche Details

werden für weitere Verarbeitungen allerdings nicht unbedingt benötigt und sind sogar hinderlich, da sie durch die große Menge an Punkten die Berechnungszeit von verarbeitenden Verfahren vergrößern. So ist es bei vielen Verfahren (zum Beispiel bei der Konstruktion von kd -Bäumen) nötig über die gesamte Punktmenge zu iterieren. Eine größere Punktmenge wirkt sich also negativ auf die Laufzeit der Verfahren aus. Kleinere Punktmenge verzerren das Ergebnis solcher Verfahren dagegen nicht. Es bietet sich also an die Punktmenge in einer Form zu reduzieren, so dass die grundlegenden Informationen immer noch genutzt werden können. Dann führt die Reduktion einer Punktwolke zu einer deutlichen Verbesserung der Laufzeit von Verfahren, die zur weiteren Verarbeitung der Punktmenge genutzt werden.

Sei eine endliche Punktmenge $\mathcal{P} \subset \mathbb{R}^2$ gegeben. Um in dieser nun Rauschpunkte zu ermitteln, kann eine einfache Heuristik genutzt werden. Hierbei sind Rauschpunkte zumeist in weniger dicht besiedelten Gebieten zu finden. Viele dieser Punkte lassen sich also durch eine k -Nächste-Nachbarn-Heuristik $H : \mathbb{R}^2 \rightarrow \mathbb{R}$ ermitteln, in der für jeden Punkt der mittlere Abstand zu seinen k nächsten Nachbarn ermittelt wird. Sei $N_k(\mathbf{p})$ die Menge der nächsten k Nachbarn des Punktes $\mathbf{p} \in \mathcal{P}$. Dann kann die Heuristik wie folgt definiert werden [21]:

$$H(\mathbf{p}) = \frac{1}{k} \cdot \sum_{\mathbf{p}' \in N_k(\mathbf{p})} (\mathbf{p} - \mathbf{p}'). \quad (4.2)$$

Für jeden Punkt \mathbf{p} wird also der mittlere Abstand zu seinen k nächsten Nachbarn in \mathcal{P} ermittelt. Um mit diesem Wert nun Rauschpunkte zu entfernen, können verschiedene Wege genutzt werden. Ein einfacher Weg ist es einen Schwellwert Θ zu definieren und alle Punkte $\mathbf{p} \in \mathcal{P}$ mit $H(\mathbf{p}) > \Theta$ zu entfernen. Eine weitere Variante ist möglich, wenn angenommen wird, dass die Punkte über der Heuristik normalverteilt sind. Mit der Normalverteilung $N(x, \mu, \sigma)$ mit dem Erwartungswert μ und der Varianz σ lässt sich dann ein Intervall definieren, so dass alle Punkte, die nicht in diesem Intervall liegen, entfernt werden.

Diese Methode hat den Vorteil, dass sie auch Punkte aus der Punktmenge entfernt, die sehr nahe an ihren Nachbarn liegen. Da das Entfernen dieser Punkte die Wirksamkeit anderer Algorithmen nicht beeinflusst, kann auf diese Weise neben der Entfernung des Rauschens die Datenmenge noch weiter reduziert werden. Eine dreidimensionale Punktmenge kann auf dieselbe Weise entrauscht werden [21].

Im Falle der Projektgruppe findet keine Entrauschung der Punktwolken statt. Die Anwendung des RANSAC-Verfahrens hat sich selbst ohne eine Entrauschung der Punktwolken als effizient erwiesen (siehe Kapitel 8.3), weshalb die Entrauschung der Punktwolken nicht weiter berücksichtigt wurde. Zur Reduktion der Punktwolken hat sich hingegen die Anwendung eines so genannten *Voxelgrid-Filters* als effizient erwiesen. Im Projekt wird die Reduktion deshalb mit einem solchen durchgeführt.

Voxelgrid-Filter

Auch Mithilfe eines Voxelgrid-Filters ist die zur Reduktion einer Punktwolke möglich [21]. In diesem wird der Raum der Punktmenge \mathcal{P} mit *Voxeln* - kubische Objekte einer gleichen, festen Länge k - überlappt. Jeder Punkt $\mathbf{p} \in \mathcal{P}$ ist dann genau einem solcher Voxel zugeordnet. Jeder Voxel enthält dann eine bestimmte Menge von Punkten $\mathcal{V}_i \subseteq \mathcal{P}$ der ursprünglichen Punktmenge. Um diese nun zu reduzieren, wird als Repräsentant eines jeden Voxels der so

genannte *Zentroid* über eine fest gewählte Metrik berechnet. Ein Zentroid entspricht dann dem Mittelpunkt der im Voxel liegenden Punkte.

Die berechneten Zentroiden ersetzen schließlich die Punkte im entsprechen Voxel in der resultierenden Punktwolke. Das bedeutet, dass für jeden Voxel höchstens ein Punkt in der resultierenden Punktwolke enthalten bleibt. Die Anzahl der übrig bleibenden Punkte in der Punktwolke hänge also maßgeblich von der gewählten Größe der Voxel ab [21].

Im Projekt der Projektgruppe wird ein Voxelgrid-Filter zur Reduktion von Punktwolken verwendet. Weitere Details zur Anwendung und der Wahl der Parameter eines solchen Filters werden in den folgenden Kapiteln erläutert.

4.2.4. Bilderzeugung

Durch die Darstellung einer Punktwolke in einem Bild ist es bereits möglich Szenarien und darin dargestellte Objekte zu erkennen. Für eine detaillierte Darstellung dreidimensionaler Objekte ist eine Punktwolke dagegen nicht ausreichend, da die darzustellenden Elemente bisher kein zusammenhängendes Objekt sind. Im Falle einer dreidimensionalen Bilderzeugung ist es notwendig die Oberfläche der darzustellenden Objekte zu berechnen. Für die Bilderzeugung sind also weitere Verarbeitungsschritte auf den Punktwolken notwendig. Hier wird im Allgemeinen zwischen zwei Arten von Verarbeitungen unterschieden. Die *netzbasierte* Bilderzeugung generiert aus Punktmengen Netze, in denen die Punkte als Knoten fungieren. Die auf diese Weise entstehenden Flächen im Bild erzeugen ein zusammenhängendes Objekt. Es handelt sich damit um einer *Interpolation* der Punktmenge, da die gesamte Punktmenge selbst Teil des Bildes bleibt. Für dreidimensionale Punktmengen werden dagegen Netze der Punkte der konvexen Hülle gebildet, so dass das berechnete Netz als Oberfläche des Objektes fungiert.

In der *punktbasierter* Bilderzeugung hingegen werden die Punkte durch Kurven und Flächen *approximiert*, sie gehören also im Allgemeinen nicht zu den resultierenden Flächen. Auch in den Verfahren dieser Kategorie können die entstehenden Flächen zum Formen einer Oberfläche und Darstellung von Objekten genutzt werden.

Delaunay-Triangulierung

Ein klassisches Verfahren zur netzbasierten Bilderzeugung von Punktmengen ist die so genannte *Triangulierung* [22]. Aus der gegebenen Punktmenge \mathcal{P} wird ein Dreiecksnetz generiert, das heißt, jeder Kreis im Netz mit einer Länge größer oder gleich 4 enthält eine Sehne. Die Knoten des Dreiecksnetzes entsprechen dabei den Punkten der Punktwolke. Durch die Generierung eines Dreiecksnetzes für die Punktwolken entsteht eine zusammenhängende Darstellung der Szenerie. Triangulierungen von Punktmengen finden deshalb unter anderem in der dreidimensionalen graphischen Darstellung von Objekten Anwendung (siehe Abbildung 4.6).

Sei $\mathcal{P} \subset \mathbb{R}^2$ eine zweidimensionale Punktmenge gegeben. Eine spezielle Art der Triangulierung ist die so genannte *Delaunay-Triangulierung*, in der die Dreiecke im Netz über eine weitere Eigenschaft, die *Delaunay-Eigenschaft* verfügen: Sei \mathcal{U}_i der Kreis, der durch das Dreieck $D_i = (\mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i)$ mit den Eckpunkten $\mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i \in \mathcal{P}$ aufgespannt wird. \mathcal{U}_i ist für jedes Dreieck eindeutig definiert. Wird für jedes Dreieck D_i im Netz der entsprechende Kreis \mathcal{U}_i betrachtet, so liegt kein anderer Punkt $\mathbf{p} \in \mathcal{P} \setminus \{\mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i\}$ innerhalb dieses Kreises: $\mathbf{p} \notin \mathcal{U}_i$ In Abbildung



Abbildung 4.6.: Triangulierung einer Punktwolke [21]

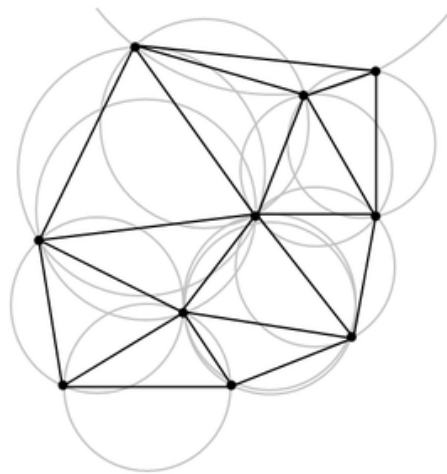


Abbildung 4.7.: Delaunay-Triangulierung einer Punktmenge; der zu jedem Dreieck eindeutig aufgespannte Kreis enthält keinen weiteren Punkt des Netzes⁴.

4.7 ist eine Delaunay-Triangulierung einer Punktmenge gemeinsam mit den entsprechenden Kreisen der Dreiecke abgebildet. Da hier die Delaunay-Eigenschaft erfüllt ist, liegen in keinem der Kreise Punkte, die nicht zum entsprechenden Dreieck gehören.

Um eine solche Delaunay-Triangulierung zu berechnen, existieren unterschiedliche Ansätze. So lässt sich eine Delaunay-Triangulierung leicht aus dem Voronoi-Diagramm der Punktmenge berechnen. In einem Voronoi-Diagramm besitzt jeder Punkt $\mathbf{p} \in \mathcal{P}$ eine *Fläche*, in der jeder Punkt liegt, dessen nächster Nachbar aus \mathcal{P} der Punkt \mathbf{p} ist. Auf diese Weise entstehen Nachbarsbeziehungen zwischen den Punkten in \mathcal{P} durch die jeweils benachbarten Flächen im Diagramm. Werden zwei Punkte $\mathbf{p}, \mathbf{p}' \in \mathcal{P}$ nun genau dann miteinander verbunden, wenn ihre Flächen benachbart sind, so erfüllt das entstehende Netz die Delaunay-Eigenschaft.

Einen anderen Ansatz zur Berechnung einer Delaunay-Triangulierung verfolgt der Flip-Algorithmus. Hier ist es das Ziel, eine beliebige Triangulierung der Punktmenge so anzupassen, dass sie die Delaunay-Eigenschaft erfüllt. Sei also eine beliebige Triangulierung von \mathcal{P} gegeben. Nun wird angenommen, dass die Triangulierung die Delaunay-Eigenschaft an mindestens einer Stelle verletzt. An einer solchen Stelle finden sich dann zwei Dreiecke, die sich eine Seite teilen und in der der Punkt, der jeweils nicht zum Dreieck gehört, in dem Umkreis des Dreiecks

⁴http://en.wikipedia.org/wiki/File:Delaunay_circumcircles.png



(a) Dreiecke vor dem Flip



(b) Dreiecke nach dem Flip

Abbildung 4.8.: Dreiecke vor und nach dem Flip⁵

liegt (siehe Abbildung 4.8 (a)). Mit diesen zwei Dreiecken wird dann ein *Flip* durchgeführt: Die bestehende Innenkante wird entfernt und durch die einzig andere innere Kante ersetzt. Auf diese Weise entstehen zwei neue Dreiecke. Diese erfüllen nun die Delaunay-Eigenschaft miteinander (siehe Abbildung 4.8 (b)).

Der Flip-Algorithmus sucht in der gegebenen Triangulierung also nach Dreiecken, die die Delaunay-Eigenschaft verletzen, und führt einen Flip auf diesen durch. Dieses Vorgehen wird solange wiederholt, bis das Dreiecksnetz die Delaunay-Eigenschaft erfüllt. Die durch einen Flip entstandenen Dreiecke verletzen möglicherweise an einer anderen Stelle die Delaunay-Eigenschaft. Deshalb besitzt der Flip-Algorithmus insgesamt eine Laufzeit von $O(|\mathcal{P}|^2)$.

Im Falle einer dreidimensionalen Punktmenge erhöht sich die Komplexität der Verarbeitung. Für eine dreidimensionale graphische Darstellung ist es nötig die Oberfläche des zu erzeugenden Objektes zu modellieren und aus dieser dann ein Dreiecksnetz zu berechnen. Für eine Punktwolke entspricht dies der Berechnung eines Dreiecksnetzes für die konvexe Hülle der Punktmenge. Dies ist auf eine einfache Weise möglich, indem zunächst die dritte Dimension ignoriert wird. Durch eine geeignete Projektion lässt sich dann die entsprechende Position der Dreiecksnetze im Raum berechnen [22].

Da die Verwendung des RANSAC-Verfahrens zur Objekterkennung sich während der Entwicklungsphase als sehr effizient erwiesen hat und diese auf einer punktbasierten Bilderzeugung aufbaut, werden netzbasierte Objekterzeugungsverfahren wie die Delaunay-Triangulierung für das Projekt nicht benötigt. Da auch die Darstellung von Objekten kein gefordertes Kriterium im Zuge der Projektgruppe war, wurde im Projekt von einer Triangulierung der Daten abgesehen.

⁵http://en.wikipedia.org/wiki/File:Delaunay_before_flip.png und
http://en.wikipedia.org/wiki/File:Delaunay_after_flip.png

Moving least squares-Verfahren

Ein klassisches Beispiel für ein Verfahren einer punktbasierten Bilderzeugung ist das so genannte *Moving least squares*-Verfahren (MLS). Ziel dieser Methode ist es für eine Punktmenge eine Ausgleichsgerade beziehungsweise eine Ausgleichsfläche (MLS-Fläche) zu berechnen. Im Gegensatz zu netzbasierten Verfahren handelt es sich hier also um einer *Approximation* der Punktmenge, das heißt die Punkte sind im Allgemeinen nicht Teil der im Ergebnis berechneten Gerade oder Fläche [23].

Für eine globale Berechnung des Ausgleichspolynoms $x^{\mathbf{a}}$ mit einem Vektor von Koeffizienten \mathbf{a} wird die Methode der kleinsten Quadrate verwendet. Die Parameter von $x^{\mathbf{a}}$ lassen sich über eine Minimierung der Summe des quadratischen Abstands der Punkte $(y_i)_{1 \leq i \leq N}$ vom Polynom bestimmen:

$$\sum_{i=1}^N (x_i^{\mathbf{a}} - y_i)^2. \quad (4.3)$$

Sofern es sich um ein Polynom vom Grad kleiner oder gleich N handelt, gibt es genau ein Polynom, das diese Gleichung minimiert. Für eine Ausgleichsgerade sind also mindestens zwei und für eine Ausgleichsfläche mindestens drei Punkte notwendig.

Es handelt sich hierbei allerdings immer um ein global approximiertes Polynom, das heißt alle Punkte der Punktmenge beeinflussen die Berechnung des Ausgleichspolynoms. Oftmals ist es allerdings gewünscht, dass das Polynom nur einer lokalen Approximation entspricht. Dies ist über das MLS-Verfahren möglich. Um eine globale Funktion $x^{\mathbf{b}}$ nur lokal zu approximieren, wird um einen festen Punkt \hat{y} eine Kugel (oder im zweidimensionalen ein Kreis) berechnet, in dem die zu approximierende Funktion $x^{\mathbf{b}(\hat{y})} = x^{\mathbf{b}}$ definiert ist. Außerhalb dieses Bereiches gilt stets $x^{\mathbf{b}(\hat{y})} = 0$. So wird sichergestellt, dass die Approximation nur im Bereich der Kugel stattfindet.

Die lokale Funktion wird dann durch eine (frei wählbare) Basisfunktion $b(x)$ mit entsprechenden Koeffizienten approximiert. Dabei entsteht ein Fehler $x^{\mathbf{r}(\hat{y})}$, der schließlich über die Methode der kleinsten Quadrate minimiert werden kann. Durch die Minimierung des Fehlers wird dann ein möglichst gutes Ausgleichspolynom $x^{\mathbf{b}(\hat{y})}$ berechnet [23].

Die Berechnung der Ausgleichspolynome ist essentiell für das RANSAC-Verfahren ([24]) und wird wie dieses im Projekt zur Erkennung der Pakete verwendet.

4.3. Objekterkennung

Nachdem aus den Sensordaten ein 3D-Bild erstellt wurde (siehe Kapitel 4.1), müssen die Daten interpretiert werden. Im Kontext der Projektgruppe bedeutet dies, dass das Paket auf den Bildern erkannt und Orientierung sowie genaue örtliche Position relativ zum Sauggreifer ermittelt werden müssen. Dafür muss das Objekt – hier also das Paket – im Tiefenbild mittels Klassifikation identifiziert werden. Um die Einschränkungen und das Vorwissen, wie z.B. die bekannten Größen-Maße des Paketes, optimal mit in den Entscheidungsprozess mit einfließen

zu lassen, wurden im Rahmen der Projektgruppe Klassifikationsverfahren untersucht, welche die Geometrie des Pakets direkt berücksichtigen.

4.3.1. Filterung lokaler Attribute

Im ersten Schritt ist es für die Erkennung von Objekten sinnvoll, lokale Attribute sinnvoll zu filtern. Mit der Filterung von Objekten, die bestimmte lokale Attribute erfüllen oder nicht zu weit von Optimalwerten abweichen, lassen sich bereits vor der Klassifikation bestimmte Objekte ausschließen, die als zu erkennendes Objekt nicht in Frage kommen.

Sei Ω die Menge aller möglichen Objekte. Ein lokales Attribut ist eine Funktion $c : \Omega \rightarrow \mathbb{R}$, das heißt, sie ordnet jedem Objekt einen Wert zu. Ein im Rahmen der Projektgruppe relevantes Beispiel hierzu ist eine Menge Ω von Paketen beziehungsweise paketähnlichen Objekten und einer Funktion c , die jedem Objekt die Länge angibt. Ist ein für ein gesuchtes Objekt bekannter Attributwert $c^* \in \mathbb{R}$ (wie die bekannte Paketlänge) gegeben, so lassen sich Objekte durch einen Vergleich der entsprechenden Attributwerte filtern. Sei mit Θ dazu ein Schwellwert für die Abweichung vom Optimalwert c^* gegeben. Weicht der Wert $c(\omega)$ für ein Objekt $\omega \in \Omega$ zu weit vom Optimalwert ab, so muss ω nicht mehr als Kandidat für die Klassifikation beachtet werden.

Da unterschiedlichste lokale Attribute in Objekten existieren, lässt sich hier allgemein nicht viel zum Erfolg einer spezifischen Filterung aussagen. Eine solche Filterung hängt also stark von den zu erkennenden Objekten sowie deren prägnanten Attributen ab. Zwei Filtermethoden, die eine effizientere Klassifikation der Pakete im Rahmen der Projektgruppe ermöglicht haben, werden in Kapitel 7.3.2 erläutert.

4.3.2. RANSAC

Im Zuge der Projektgruppe ist für das Erkennen des Paketes die Bestimmung der oberen Paketfläche ausreichend, da nur diese für das Greifen des Paketes von Relevanz ist. Ziel der Erkennung ist es also, die Oberseite eines Quaders, also einem Rechteck zu klassifizieren. In diesem Fall ist es möglich eine solche mit Hilfe einer Ausgleichsfläche eines lokalen Bereiches zu bestimmen.

Im Kapitel 4.2.4 ist mit MLS bereits ein Verfahren zur lokalen Berechnung eines Ausgleichspolynoms vorgestellt worden. In dieser Methodik muss allerdings ein fester Punkt gegeben sein, so dass nur Punkte in dessen Umgebung bei der Bestimmung des Ausgleichspolynoms betrachtet werden. Da im Allgemeinen kein Punkt des zu erkennenden Objektes im Vorfeld bekannt ist, eignet sich dieses Verfahren in der vorgestellten Form zunächst nicht zur Objekterkennung.

Um den wahrscheinlichsten Ort einer zu erkennenden Fläche zu bestimmen, ist ein randomisierter Ansatz möglich. Wird die Punktmenge der Oberseite eines Quaders betrachtet, so liegen die Punkte der entsprechenden Menge der Geometrie auf einer Ebene. Wird nur zur Punktmenge eine Ausgleichsfläche berechnet, so ist der minimale quadratische Fehler sehr gering. Dies rührt daher, dass eine solche Punktmenge wenige so genannte *Ausreißer* besitzt, die weit von der approximierten Fläche entfernt liegen und das Gesamtbild somit verzerren. Eine Umgebung mit wenigen Ausreißern kann also ein aussichtsreicher Kandidat für das Finden einer Fläche sein.

Das Finden von lokalen Punktmengen mit möglichst wenigen Ausreißern (und damit der wahrscheinlichste Ort für die zu erkennende Fläche) lässt sich über das *RANSAC-Verfahren* (*Random sample consensus*) realisieren [24]. Um Ausreißer zu bestimmen, muss dabei zunächst ein Schwellwert für den Abstand zur Ausgleichsfläche festgelegt werden. Sei mit $\mathcal{P} \subset \mathbb{R}^3$ die Punktmenge und mit Ω der angesprochene Schwellwert gegeben. Das RANSAC-Verfahren ist iterativ und wählt zunächst eine minimale Menge von zufälligen Punkten, die zum Aufspannen des entsprechenden Ausgleichspolynoms benötigt werden. Im Falle einer Ausgleichsfläche sind dies drei Punkte. Seien $\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \mathbf{p}_{i,3} \in \mathcal{P}$ die Punkte, die in der i -ten Iteration zufällig gewählt werden. Über die Methode der kleinsten Quadrate kann dann das Ausgleichspolynom x_i^a dieser drei Punkte bestimmt werden.

Es handelt sich dabei um ein iteratives Verfahren, das zunächst eine minimale Anzahl von zufälligen Punkten wählt, die zum Aufspannen des entsprechenden Ausgleichspolynoms nötig sind, im Falle einer Fläche also drei Punkte. Um zu überprüfen, ob es sich bei diesen Punkten um Ausreißer handelt, wird der mittlere quadratische Abstand aller anderen Punkte zum Ausgleichspolynom berechnet. Liegt dieser für einen Punkt unter einem vorher festgelegten Schwellwert, so *unterstützt* dieser die das Ausgleichspolynom und wird der so genannten *Consensus-Menge* hinzugefügt.

Dieses Verfahren wird iterativ wiederholt und unabhängig voneinander mehrfach durchgeführt. Abschließend wird die größte Consensus-Menge gewählt und auf dieser eine Ausgleichsfläche (zum Beispiel über die Methode der kleinsten Quadrate) berechnet. Alternativ kann auch die Consensus-Menge als erkanntes Objekt ausgegeben werden [24].

5. Handhabung von Industrierobotern

Zur Entladung eines sich bewegenden FTF ist neben der Detektion der Positionsdaten des Fahrzeuges und somit des zu entladenden Paketes auch die adäquate Steuerung eines Roboters erforderlich. Zur Realisierung dieser, werden in diesem Kapitel, nach einer einleitenden Problemstellung in Abschnitt 5.1, Aufbau und Funktionsweise von Industrierobotern in Abschnitt 5.2 sowie die wesentlichen Grundlagen der Bahnplanung in Abschnitt 5.3 vorgestellt.

5.1. Problemstellung

Die fahrerlosen Transportfahrzeuge befördern die Pakete innerhalb des Logistikzentrums. Der Industrieroboter soll einen autonomen Transfer der Pakete zwischen Fahrzeug und Europalette realisieren (siehe Abbildung 5.1). Hierzu sind die Positionen, Orientierungen und Geschwindigkeiten der Fahrzeuge beziehungsweise der Pakete erforderlich. Diese werden mithilfe von optischen Sensoren bestimmt. Die Funktionsweise dieser Sensoren wurde bereits in Kapitel 3, bzw. in Kapitel 4 vorgestellt. Weiterhin ist eine Prädiktion dieser Sensordaten nötig, welche im anschließenden Kapitel 6 beschrieben wird.

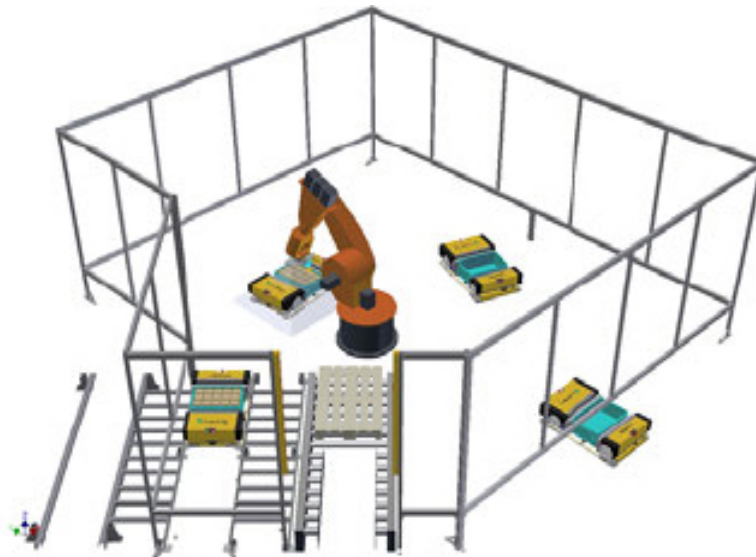


Abbildung 5.1.: Modell der Roboterzelle ¹

¹TU Dortmund. PG-Präsentation: <http://ls7-www.cs.uni-dortmund.de/projectgroups/dynolog>



Abbildung 5.2.: Aufnahme des Industrieroboters Kuka 125/3 beim Entladen der fahrerlosen Transportfahrzeuge ²

5.2. Industrieroboter

Roboter haben einen bedeutenden Anteil an der zunehmenden Automatisierung in vielen Industriezweigen. Die wichtigsten Grundlagen werden in den folgenden Abschnitten erläutert.

Industrieroboter werden von dem Verein Deutscher Ingenieure durch die Richtlinie 2860 definiert:

„Industrieroboter sind universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolge und Wegen bzw. Winkeln frei (d.h. ohne mechanischen Eingriff) programmierbar und gegebenenfalls sensorgeführt sind. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar und können Handhabungs- und/oder Fertigungsaufgaben ausführen“ [25].

Abbildung 5.2 zeigt den Industrieroboter Kuka 125/3, der in der Projektgruppe verwendet wird, beim Entladen eines Paketes. Der folgende Abschnitt erläutert den Aufbau und die grundsätzliche Funktionsweise der wichtigsten Komponenten.

5.2.1. Freiheitsgrade

Der Freiheitsgrad gibt die Anzahl der möglichen, voneinander unabhängigen Translationen und Rotationen an. Somit sind im dreidimensionalen Raum beliebige Positionen und Orientierungen

²TU Dortmund. PG-Präsentation: <http://ls7-www.cs.uni-dortmund.de/projectgroups/dynolog>

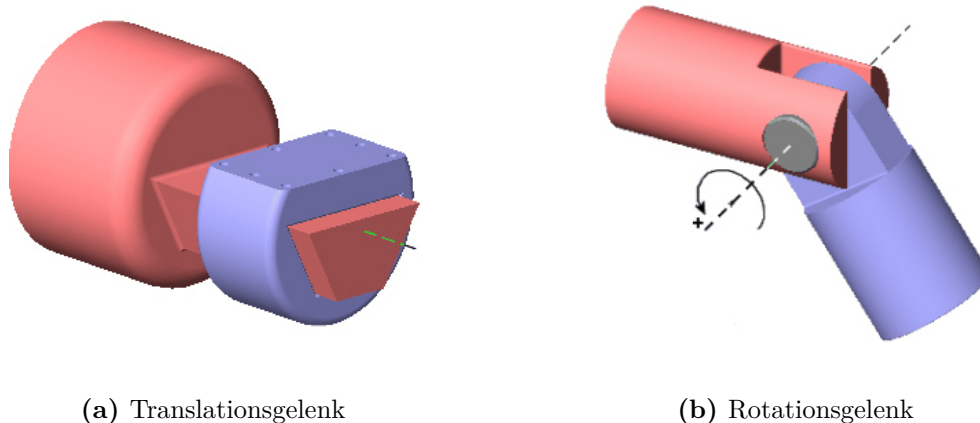


Abbildung 5.3.: Gelenkmodelle ³

mit Freiheitsgrad sechs realisierbar.

Ein Translationsgelenk (siehe Abbildung 5.3a) hat genau wie ein Rotationsgelenk (siehe Abbildung 5.3b) den Freiheitsgrad eins, da jeweils nur die Verschiebung bzw. die Rotation um eine Achse möglich ist. Ein Schraubgelenk realisiert dagegen eine Verschiebung und gleichzeitig eine Rotation. Da diese Bewegungen aber voneinander abhängig sind, hat dieses Gelenk trotzdem den Freiheitsgrad eins. Ein Kugelgelenk kann alle beliebige Rotationen, aber keine Translationen realisieren und hat deshalb den Freiheitsgrad drei.

Ein Industrieroboter mit sechs Achsen hat den Freiheitsgrad sechs, wenn seine Gelenke alle voneinander unabhängige Bewegungen ausführen. Singularitäten führen zum Verlust von Freiheitsgraden. Sie treten auf, wenn mehrere Achsen die gleiche Bewegung durchführen und somit nicht mehr unabhängig sind (siehe Abbildung 5.5b).

5.2.2. Arbeitsbereich

Die Bewegungsmöglichkeiten des Roboters sind abhängig von dessen kinematischer Kette, die sich aus den Armgliedern und Gelenken zusammensetzt. Der daraus hervorgehende Arbeitsraum vom Kuka Roboter 125/3, der beim Transport der Pakete verwendet wird, ist in Abbildung 5.4 zweidimensional dargestellt. Im dreidimensionalen Raum ist der Arbeitsbereich annähernd kugelförmig.

Neben der Erreichbarkeit einer Position sind auch die jeweils möglichen Orientierungen relevant. Deshalb wird beim Arbeitsbereich zwischen dem Raum, der vom Tool Center Point (TCP) erreicht werden kann (reachable workspace), und einem Unterraum, in dem beliebige Orientierungen realisiert werden können (dexterous workspace), unterschieden [26]. Der Tool Center Point ist ein charakteristischer Punkt des Effektors, für den Zielpunkte des Roboters definiert werden können. Der (End-)Effektor ist dabei das Werkzeug, welches vom Roboter gerade eingesetzt wird.

³MathWorks: <http://www.mathworks.de/>

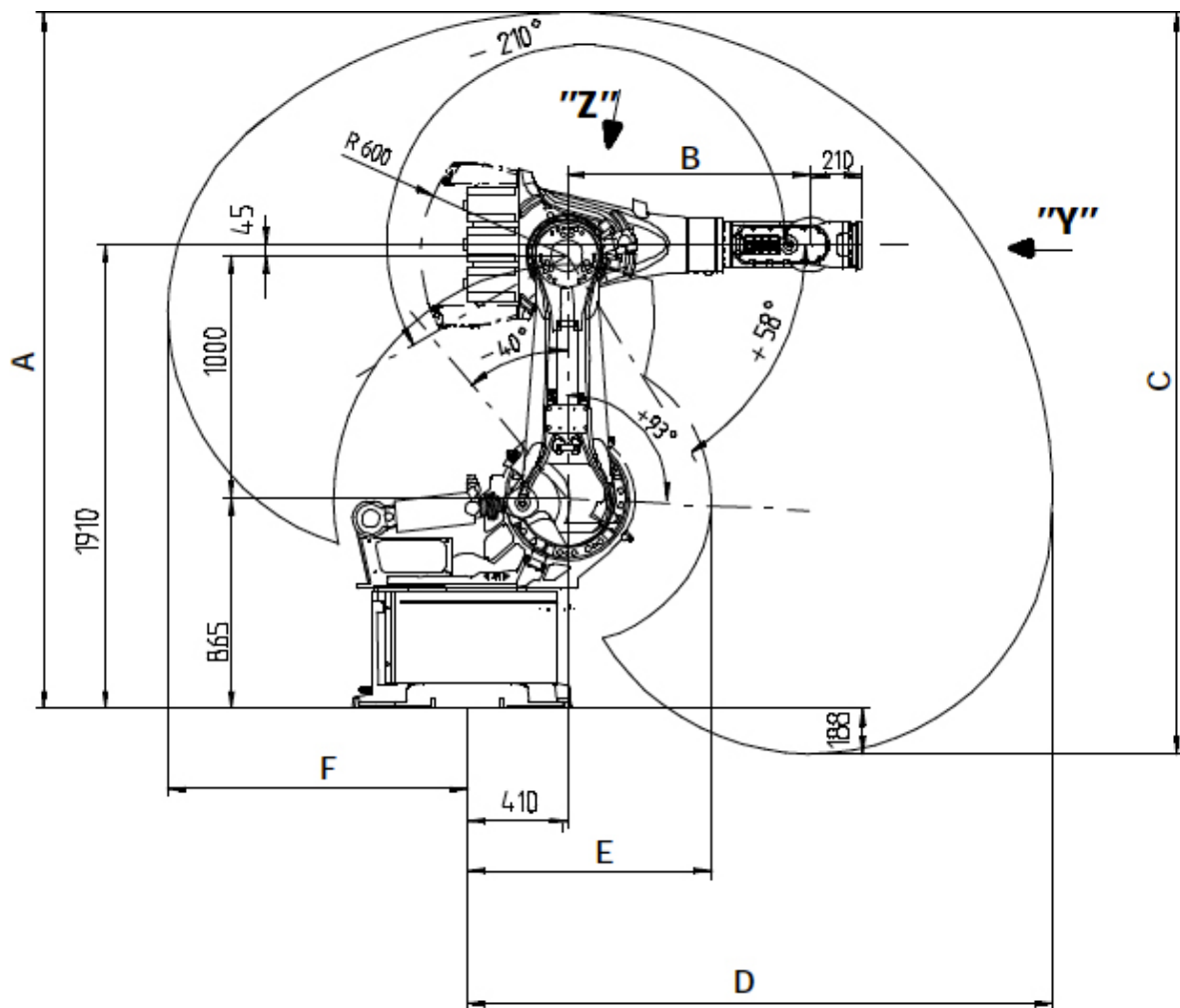


Abbildung 5.4.: Seitenansicht vom Arbeitsbereich des Kuka Roboter 125/3 [27]

5.2.3. Koordinatensysteme

Koordinatensysteme sind in der Robotik essenziell für die eindeutige Beschreibung der Position und Orientierung eines Objektes. Sie ermöglichen die Angabe von globalen aber auch relativen Werten. So kann der Zielpunkt für den TCP sowohl im Weltkoordinatensystem als auch abhängig vom Koordinatensystem und somit der Position und Orientierung des Werkobjektes festgelegt werden.

Die wichtigsten Koordinatensysteme werden im Folgenden vorgestellt, die exakten Bezeichnungen können in der Literatur variieren:

- *World*: Das Weltkoordinatensystem definiert einen globalen Bezugspunkt. Alle weiteren Koordinatensysteme werden relativ zu diesem System angegeben. Es kann mit dem Basiskoordinatensystem des Roboters übereinstimmen oder an einem anderen sinnvollen Punkt platziert werden.
- *Base*: Das Basiskoordinatensystem beschreibt den Ursprung des Roboters.

- *Workobject*: Für Objekte, mit denen der Roboter interagiert, können weitere Koordinatensysteme spezifiziert werden. So kann beispielsweise ein Paket abhängig von der Position und Orientierung einer Palette bewegt werden, ohne die Position in Weltkoordinaten explizit zu bestimmen.
- *TCP*: Der Tool Center Point ist ein charakteristischer Punkt des Effektors. Bei der Programmierung eines Roboters werden meist Zielpunkte für den TCP angegeben, die dann von der Robotersteuerung in Gelenkvariablen umgerechnet werden.

5.2.4. Transformation

Im folgenden werden die Transformationen vorgestellt, die benötigt werden, um aus den Gelenkvariablen des Roboters in das TCP Koordinatensystem und wieder zurück zu transformieren.

Die Vorwärtstransformation wird benötigt, um aus den Gelenkvariablen des Roboters das TCP Koordinatensystem und somit Position und Orientierung des Effektors zu bestimmen. Die Rückwärtstransformation erlaubt das Berechnen möglicher Gelenkvariablen aus einer gegebenen Position und Orientierung des TCP.

Zunächst muss der Roboter in ein mathematisches Modell überführt werden. Nach der Denavit-Hartenberg-Konvention wird in jedem Robotergelenk nach bestimmten Regeln ein Koordinatensystem platziert. Die Transformation zwischen zwei aufeinanderfolgenden Koordinatensystemen kann dabei eindeutig durch vier Parameter angegeben werden. Eine Transformationsmatrix mit diesen Werten beschreibt den Übergang zwischen den Koordinatensystemen. Die Vorwärtstransformation beschreibt somit die Beziehung zwischen dem Basiskoordinatensystem des Roboters und dem TCP durch eine Reihe von Matrixmultiplikationen

$$\mathbf{T}_n^0 = \mathbf{T}_1^0 \cdot \mathbf{T}_2^1 \cdot \mathbf{T}_3^2 \cdot \dots \cdot \mathbf{T}_n^{n-1}. \quad (5.1)$$

Die Rückwärtstransformation ist aufwendiger. Damit die Gelenkvariablen ermittelt werden können, muss die Gleichung 5.1 schrittweise invertiert werden.

$$\begin{aligned} (\mathbf{T}_1^0)^{-1} \cdot \mathbf{T}_n^0 &= \mathbf{T}_2^1 \cdot \mathbf{T}_3^2 \cdot \dots \cdot \mathbf{T}_n^{n-1} \\ (\mathbf{T}_2^1)^{-1} \cdot (\mathbf{T}_1^0)^{-1} \cdot \mathbf{T}_n^0 &= \mathbf{T}_3^2 \cdot \dots \cdot \mathbf{T}_n^{n-1} \\ &\dots \end{aligned} \quad (5.2)$$

Gleichung 5.2 zeigt die Vorgehensweise. Bei jedem Schritt werden die Matrizen elementweise gleichgesetzt, um so die Gelenkvariablen zu bestimmen [28].

Bei der Lösung können Probleme durch den begrenzten Arbeitsraum, Mehrdeutigkeiten der Armstellung (siehe Abbildung 5.5a) und Singularitäten der Gelenkachsen (siehe Abbildung 5.5b) auftreten.

Wenn die Gelenkvariablen für einen Zielpunkt außerhalb des Arbeitsbereiches oder eine nicht realisierbare Orientierung bestimmt werden sollen, ist die inverse Kinematik nicht lösbar. Bei Mehrdeutigkeiten und Singularitäten ergeben sich mehrere beziehungsweise beliebig viele mögliche Gelenkvariablen.

Die inverse Kinematik kann alternativ auch durch ein numerisches Verfahren berechnet werden,

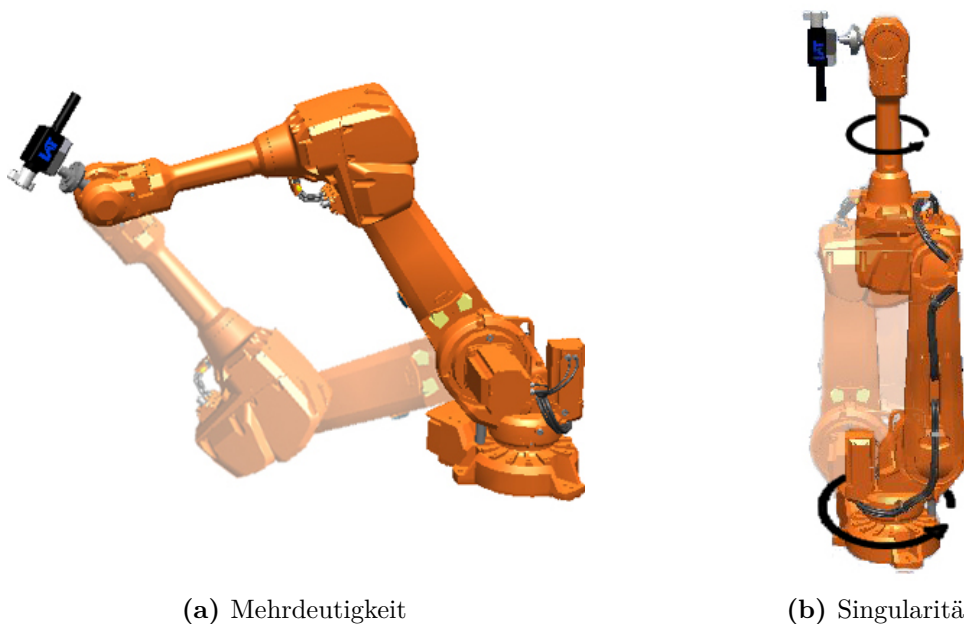


Abbildung 5.5.: Darstellung von Problemen der Rückwärtstransformation [29]

das allerdings nur eine der möglichen Lösungen bestimmt. Die so bestimmten Werte sind aber von den festgelegten Startwerten abhängig. Damit Mehrdeutigkeiten und Singularitäten auf diese Weise erkannt werden können, muss dieses Verfahren mit verschiedenen sinnvoll gewählten Startwerten durchgeführt werden. Für eine Echtzeitanwendung ist die numerische Berechnung meist zu langsam.

5.3. Bahnplanung

Mittels der Bahnplanung kann eine Trajektorie zwischen einem Start- und einem Endpunkt innerhalb des Arbeitsraumes des Roboters bestimmt werden [30]. Die Herausforderung der Bahnplanung besteht dabei darin, die gewünschte Bewegung des Roboters zu definieren und diese hinsichtlich mehrerer Kriterien zu optimieren, wobei eine kollisionsfreie Bahn selbstverständlich die Grundvoraussetzung ist. Gütekriterien können eine möglichst geringe Zeitspanne für die Bewegung, eine energieeffiziente Roboterbahn und ein geringer Verschleiß des Roboters sein. In der Projektgruppe ist die Zeit, die zur Berechnung und Durchführung der Roboterbewegung benötigt wird, eine besonders kritische Größe, da sich das FTF nur eine begrenzte Zeit im Arbeitsbereich des Roboters aufhält.

Das Bewegungsverhalten, das der Industrieroboter beschreiben soll, kann mittels zweier grundlegender Verfahren in der Robotersteuerung einprogrammiert werden. Eines dieser Verfahren ist die Online-Programmierung. Bei dieser erfolgt die Programmierung durch direkte Interaktion des Bedieners mit dem Roboter. Hierbei werden die Gelenke des Roboters durch ein Steuergerät im sogenannten Langsamfahrbetrieb in die gewünschte Zielstellung bewegt und abgespeichert [31]. Diese Vorgehensweise wird auch als Teach-In-Modus bezeichnet. Eine Reihe von definierten Zielpunkten mit zusätzlichen Parametern, wie beispielsweise

Interpolationsverfahren, Positionierung und Geschwindigkeit, bilden zusammen mit Befehlen für entsprechend angebrachte Effektoren das Programm. Der Vorteil dieses Verfahrens ist, dass der Bediener sich das Verhalten des Roboters direkt im Anschluss an die Erstellung des Programmes anschauen und gegebenenfalls notwendige Änderungen umgehend einarbeiten kann. Allerdings muss bei dieser Art der Bewegungsprogrammierung in der Praxis ein Teil des Systems (beispielsweise einer Fertigungsanlage) stillgelegt werden, bis das Programm voll funktionstüchtig ist [31].

Aus dem Grund der Wirtschaftlichkeit kommt häufig bei Systemen, die sich bereits im Betrieb befinden, die sogenannte Offline-Programmierung zum Einsatz. Hierbei ist kein Zugriff auf den Roboter erforderlich, da die Programmierung an einem Computer durchgeführt wird. Die programmierten Bewegungen und Aktionen werden dabei meist durch eine Simulation evaluiert, bevor das erstellte Programm auf den Roboter überspielt wird. Zur Erstellung des Programms existieren einige kommerzielle Simulations- und Offlineprogrammierungstools. Die Programme der Roboterhersteller wie KUKA.SIM und ABB Robot Studio eignen sich, um beliebige Aktionen äquivalent zur Robotersteuerung im Programm zu definieren und realitätsnah zu simulieren. Sie beinhalten zusätzliche Mechanismen wie beispielsweise Bahnoptimierung oder Kollisionserkennung [32]. Anwendungsspezifische Spezialprogramme bieten dagegen die Möglichkeit, die Programmierung zumindest teilweise automatisch durchzuführen.

5.3.1. Bewegungsformen

Die Bewegungsformen des Roboters lassen sich in zwei wesentliche Kategorien einteilen. Es gibt die Point to Point (PTP) Bewegung, bei der nur die genaue Position und Orientierung für den Start- und Zielpunkt definiert ist und die Continuous Path (CP) Bewegung, welche einen genau definierten Roboterpfad beschreibt, bei dem alle Zwischenpunkte exakt bestimmbar sind [26].

Bei der PTP-Bewegung folgt der Roboter keinem spezifizierten Pfad, sondern bewegt alle Gelenke separat in die Zielstellung, die dem definierten Zielpunkt im Weltkoordinatensystem entspricht. Die Bahnbewegung ergibt sich dabei aus der Trajektorie, welche zwischen den Achsstellungen des Start- und des Zielpunktes berechnet wird. Bei der PTP-Bewegung sind somit nur diese beiden Punkte detailliert festgelegt. Es kann zwischen dem asynchronen PTP und dem synchronen PTP unterschieden werden [33]. Im asynchronen Fall bewegen sich alle Gelenke mit ihrer maximal zulässigen Geschwindigkeit und erreichen die Zielstellung meist nicht gleichzeitig. Bei diesem Verfahren treten hohe Beschleunigungen auf, die oft keinerlei Vorteil bewirken, aber den Verschleiß der Motoren erhöhen [33]. Deshalb ermöglicht die synchrone Variante alle Geschwindigkeiten an die langsamste Achse anzupassen, sodass alle Gelenke die Zielposition zeitgleich erreichen (siehe Abbildung 5.6). Hierzu wird von der Steuerung des Roboters die Achse bestimmt, welche den größten Weg bzw. die maximale Bewegungszeit benötigt [33].

Die CP-Bewegung kann hingegen als Interpolation zwischen dem Startpunkt und dem Zielpunkt aufgefasst werden. Hierbei werden definierte geometrische Bahnen, beispielsweise lineare oder kreisförmige Pfade, im Weltkoordinatensystem mittels des Effektors abgefahren [33]. Neben den Positionen werden auch die Orientierungen interpoliert, sodass jeder Zwischenpunkt theoretisch eindeutig nachvollziehbar ist. Zur Definition einer Kreisbahn ist neben dem Start- und Zielpunkt noch zusätzlich ein weiterer Punkt erforderlich.

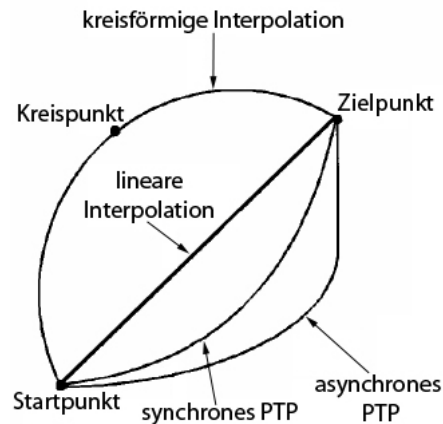


Abbildung 5.6.: Darstellung des Roboterpfades für verschiedene Bewegungsformen (mod. nach [26])

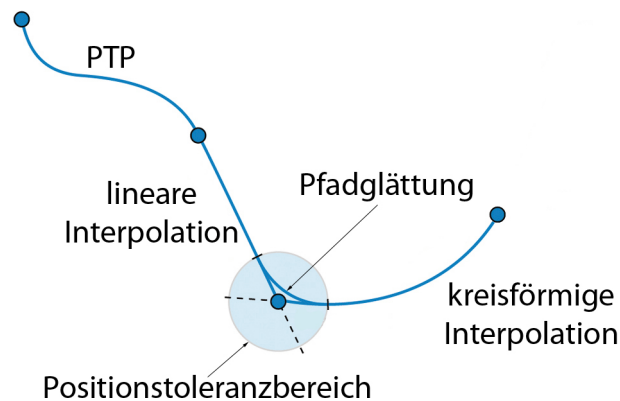


Abbildung 5.7.: Schematische Darstellung der Pfadglättung ⁴

Die PTP-Bewegung ist zwar schneller, aber kann ausschließlich dann verwendet werden, wenn die tatsächliche Bahn nicht genau definiert werden muss. Dies kann beispielsweise der Fall sein, wenn der Abstand der PTP-Bahn zu sämtlichen Hindernissen ausreichend groß ist. Um die PTP-Bewegung auch bei störenden Hindernissen nutzen zu können, ist es möglich, den Gesamtpfad des Roboters durch eine Vielzahl von Zielpunkten zu definieren. Diese stellen, bis auf den letzten Punkt, Zwischenpunkte dar, welche auch als Fly-by-Punkte verstanden werden können. Fly-by-Punkte müssen nicht exakt angefahren werden, sondern dienen beispielsweise dazu Hindernisse zu umfahren. Die exakte Anfahrt eines Zwischenpunktes hätte zur Folge, dass der Roboter in der Regel seine Geschwindigkeit stark reduzieren muss, um eine mögliche Richtungsänderung für den nächsten Bahnabschnitt vom aktuellen Zwischenpunkt zum nachfolgenden realisieren zu können. Es kann folglich einiges an Zeit eingespart werden, wenn der Roboter die Zwischenpunkte lediglich näherungsweise anfahren muss. Aus diesem Grund werden für Zwischenpunkte, wenn möglich, Toleranzbereiche definiert [34]. Diese Bereiche werden auch als Überschleifbereiche bezeichnet. Sie werden in Form einer Kugel um den betrachteten Punkt definiert und genutzt um die Position, Geschwindigkeit oder Orientierung zwischen den angrenzenden Bahnabschnitten sanft anzugleichen (siehe Abbildung 5.7). Auf

⁴SIGMATEK GmbH: <http://www.sigmatek-automation.com/>

diese Weise wird eine gleichmäßigere Bewegung ermöglicht, die meist schneller ist und zudem weniger Verschleiß hervorruft, da starke Beschleunigungen vermieden werden können.

Im Folgenden werden zwei verschiedene Ansätze beschrieben, welche bei der Bahnplanung zur Bestimmung des Weges zwischen Start- und Zielpunkt zum Einsatz kommen könnten.

5.3.2. Zellbasierte Methoden

Die Hindernisvermeidung spielt eine zentrale Rolle bei der Bahnplanung. Dabei wird das Umfeld, welches der Roboter passiert, durch eine Abbildung modelliert.

Algorithmen, die auf einer kartenbasierten Umfeldmodellierung basieren, arbeiten auf einer Unterteilung des Raumes in Zellen. Die einzelnen Zellen besitzen zwei Zustände. Sie sind entweder frei, falls der beschriebene Umfeldbereich keine Hindernisse enthält, oder belegt. Die Gesamtheit aller Zellen stellen den Zustandsraum der Karte dar. Die Unterteilung geschieht typischerweise zweidimensional oder dreidimensional. Im Nachfolgenden wird die Annahme getroffen, dass eine zweidimensionale kartenbasierte Umfeldmodellierung angestrebt wird. Um diese Modellierung zu berechnen, werden zwei verschiedene Ansätze vorgestellt.

Binärer Bayes Filter mit statistischem Zustandsvektor

Der binäre Bayes Filter mit statistischem Zustandsvektor bedient sich der Methoden der klassischen Wahrscheinlichkeitstheorie und liefert eine probabilistische Beschreibung, die einen Kartenzustand bestimmt, der mit den gegebenen Messdaten übereinstimmt.

Eine Karte wird durch ein zweidimensionales $M \times N$ -Gitter beschrieben. Eine Zelle dieses Gitters wird durch $z(i, j)$ indexiert und ist folgendermaßen definiert:

$$z(i, j) = \begin{cases} 0 & \text{wenn frei} \\ 1 & \text{wenn belegt.} \end{cases} \quad (5.3)$$

Gesucht ist die a-posteriori Wahrscheinlichkeitsverteilung bei Messdaten Y zum Zeitpunkt k :

$$P(Z | Y_k) = P(z(1, 1), \dots, z(M, N) | Y_k). \quad (5.4)$$

Aus der berechneten Verteilung wird anschließend die Karte gewählt, die die größte Verbundwahrscheinlichkeitsdichte besitzt.

Um den Rechenaufwand zu minimieren, wird die Annahme getroffen, dass die Zellen statistisch unabhängig sind, da jede Zelle einen individuellen Umfeldbereich behandelt und der Zustand einer Zelle somit nicht vom Zustand ihrer Nachbarn abhängt.

Gleichung 5.4 vereinfacht sich folgendermaßen:

$$P(z(1, 1), \dots, z(M, N) | Y_k) = \prod_{i,j=1}^{M,N} P(z(i, j) | Y_k). \quad (5.5)$$

Somit können die Wahrscheinlichkeiten für die Zellen isoliert berechnet werden. Gesucht ist also die a-posteriori Belegwahrscheinlichkeit einer Zelle zum Zeitpunkt $k + 1$, die folgendermaßen beschrieben werden kann:

$$P(Z_{k+1} = 1 | Y_{k+1}) = \frac{P(Y_{k+1} | z_{k+1} = i, Y_k) \cdot P(z_{k+1} = 1 | Y_k)}{P(y_{k+1} | Y_k)}. \quad (5.6)$$

Weiterhin wird die Vereinfachung getroffen, dass der Zellinhalt statisch ist. Er kann sich somit nicht durch Eigendynamik verändern. Unter Verwendung der Markov-Eigenschaft ergibt sich:

$$P(Z_{k+1} = 1 | Y_{k+1}) = \frac{\overbrace{P(Y_{k+1} | z_{k+1} = i)}^{\text{Messwertaktualisierung}} \cdot \overbrace{P(z_k = i | Y_k)}^{\text{Prädiktion}}}{\underbrace{\sum_{l=0}^1 P(Y_{k+1} | z_{k+1} = l) \cdot P(z_k = l | Y_k)}_{\text{Normierung}}}. \quad (5.7)$$

Für die Anwendung des Verfahrens ist eine logarithmische Darstellung von Vorteil. Damit ergibt sich ein rekursiver Filteralgorithmus für die Aktualisierung jeder Zelle:

$$l(z_{k+1}) = \underbrace{\log \frac{P(z_{k+1} | Y_{k+1})}{1 - P(z_{k+1} | Y_{k+1})}}_{\text{Messwertaktualisierung}} + \underbrace{\log \frac{P(\bar{z}_{k+1})}{1 - P(\bar{z}_{k+1})}}_{\text{A-priori Wissen}} + \underbrace{l(z_k)}_{\text{letzter Wert}}. \quad (5.8)$$

Um die klassische Wahrscheinlichkeitstheorie zu erweitern, wird im Folgenden die Dempster-Shafer Theorie vorgestellt. Diese bietet die Möglichkeit unsichere und unvollständige Messdaten modellieren zu können. Außerdem kann zwischen unsicherem Wissen und fehlerhaftem Wissen unterschieden werden.

Dempster-Shafer Theorie

Die Dempster-Shafer Theorie arbeitet mit Evidenzen als Beobachtung für oder gegen eine Teilmenge des Ereignisraums Ω . Ω besteht, wie bei der klassischen Wahrscheinlichkeitstheorie, aus sich gegenseitig ausschließenden Elementarereignissen. Jedoch bekommt auch jede mehrelementige Teilmenge von Ω eine Basiswahrscheinlichkeit $m : P[X] \rightarrow [0, 1]$. Somit lassen sich Unsicherheiten modellieren. Durch Dempsters Kombinationsgesetz lässt sich aus den Basiswahrscheinlichkeiten ein gemeinsamer Konsens bilden:

$$m_{1,2}(R) = m_1(R) \oplus m_2(R) = \frac{\sum_{S_1, S_2: S_1 \cap S_2 = R \neq \emptyset} m_1(S_1) \cdot m_2(S_2)}{1 - \sum_{S_1, S_2: S_1 \cap S_2 = \emptyset} m_1(S_1) \cdot m_2(S_2)}, R \in \Omega. \quad (5.9)$$

Falls für jedes einzelne Element in Ω eine Wahrscheinlichkeit angegeben werden kann, degeneriert die Dempster-Shafer Theorie zur klassischen Wahrscheinlichkeitstheorie.

Auf der Basis der soeben vorgestellten Dempster-Shafer Theorie lassen sich die Belegungskarten auch mit Unsicherheiten modellieren. Da weiterhin von einem binären Zustand einer Zelle ausgegangen wird, wird die folgende Potenzmenge erhalten:

$$2^\Omega = \begin{cases} \emptyset & \text{(leere Menge)} \\ B & \text{(belegt)} \\ F & \text{(frei)} \\ \{B, F\} = U & \text{(unbekannt/keine Aussage)}. \end{cases} \quad (5.10)$$

Die Unsicherheit wird hierbei durch die Teilmenge $\{B, F\}$ modelliert.

Wie bei der Bayes'schen Belegungskarte wird von statistischer Unabhängigkeit und statischen Zellen ausgegangen. Die Berechnung wird also für jede Zelle separat durchgeführt. Aus Dempsters Kombinationsgesetz ergibt sich folgende Verteilung:

$$m_{k+1}(B) = \frac{m_k(B) \cdot m_{Y_{k+1}}(B) + m_k(U) \cdot m_{Y_{k+1}}(B) + m_k(B) \cdot m_{Y_{k+1}}(U)}{1 - m_k(B) \cdot m_{Y_{k+1}}(F) - m_k(F) \cdot m_{Y_{k+1}}(B)} \quad (5.11)$$

$$m_{k+1}(F) = \frac{m_k(F) \cdot m_{Y_{k+1}}(F) + m_k(U) \cdot m_{Y_{k+1}}(F) + m_k(F) \cdot m_{Y_{k+1}}(U)}{1 - m_k(B) \cdot m_{Y_{k+1}}(F) - m_k(F) \cdot m_{Y_{k+1}}(B)} \quad (5.12)$$

$$m_{k+1}(U) = 1 - m_{k+1}(B) - m_{k+1}(F). \quad (5.13)$$

Wavefront-Algorithmus

Der Wavefront-Algorithmus ist ein möglicher Bahnplanungsansatz, der einen Pfad zwischen einem Start- und Zielpunkt berechnet. Auf der Suche nach dem kürzesten Weg werden Hindernisse berücksichtigt [35].

Der betrachtete Raum wird durch ein reguläres Gitter modelliert, welches mit Hilfe der zuvor beschriebenen kartenbasierten Umfeldmodellierungsverfahren erhalten wird. Für jede Zelle wird die Distanz zum Startpunkt berechnet, indem eine modifizierte Adaption des Dijkstra-Algorithmus ausgeführt wird. Dabei werden alle Kantengewichte zwischen zwei benachbarten Zellen als Eins angenommen. Somit können sich einmal gesetzte Werte nicht mehr ändern, sondern nur unbetrachtete Elemente, wodurch eine deutliche Vereinfachung erzielt wird. Wenn der Zielpunkt erreicht wurde, muss die Berechnung nicht fortgesetzt werden, weil keine weitere Verbesserung der Gewichte möglich ist. Ausgehend vom Zielpunkt wird rückwärts der optimale Pfad gesucht, indem jeweils der Weg zur benachbarten Zelle mit dem geringsten Wert gewählt wird (siehe Abbildung 5.8). Bei Mehrdeutigkeiten müssen Heuristiken angewandt werden.

Der so bestimmte Roboterpfad ist nicht zwangsläufig optimal im Hinblick auf Zeit oder Energieeffizienz. Außerdem ist der Pfad aufgrund der kartenbasierten Modellierung starken Richtungsänderungen ausgesetzt, was für einen Roboter ungünstig ist. Daher ist eine Nachbearbeitung notwendig. Es können beispielsweise für Zwischenpunkte, abhängig von den Distanzen zu den nächstgelegenen Hindernissen, Glättungsbereiche definiert werden, die es der Robotersteuerung ermöglicht einen für die Kinematik günstigeren Pfad zu wählen.

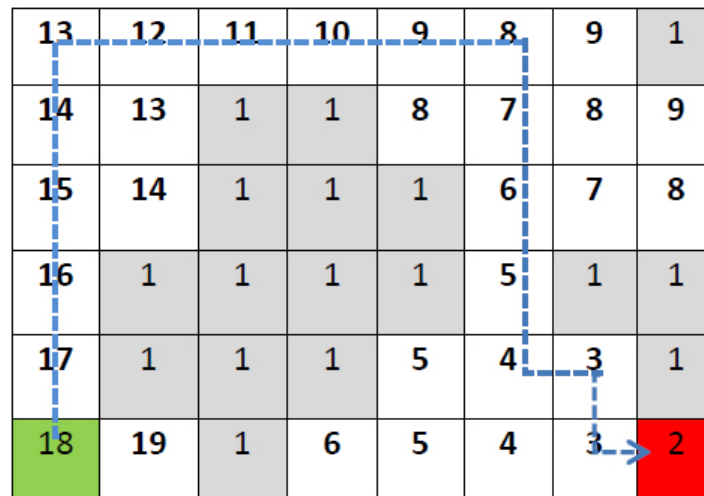


Abbildung 5.8.: Zweidimensionale Visualisierung des Wavefront-Algorithmus ⁵

Neben diesem Ansatz existieren weitere Algorithmen, wie das Verwenden von Potenzialfeldern, welche für die Bahnplanung eingesetzt werden können und sich im Bezug zur Projektgruppe eher eignen.

5.3.3. Potentialfelder

Potenzialfelder ermöglichen bei der Bahnplanung, im Gegensatz zu zellbasierten Methoden, den Verlauf der resultierenden Bahn detailliert zu beeinflussen. Dabei wird nicht zwangsläufig der kürzeste Weg, der Startpunkt und Zielpunkt verbindet, sowie Hindernisse umgeht, gesucht. Stattdessen kann der exakte Verlauf durch Potenzialfunktionen beeinflusst werden, die nicht nur Hindernisse vermeiden, sondern auch beispielsweise besonders günstige Bereiche bevorzugen können.

Ein Potenzialfeld ist aus einer beliebigen Anzahl von sich überlagernden Funktionen aufgebaut. Diese können anziehend oder abstoßend wirken, um die benötigten Eigenschaften zu erzielen. Radialsymmetrische Funktionen sind eine elegante Möglichkeit, um punktförmige Ziele zu definieren. Sie werden allerdings auch häufig für kleine Hindernisse eingesetzt. Eine klassische anziehende Funktion ist

$$U_{att}(q) = \frac{1}{2}\xi\rho_g^2. \quad (5.14)$$

Dabei beschreibt $\rho_g = \|q - q_g\|$ den euklidischen Abstand zwischen Roboter und Zielpunkt und ξ ist ein positiver Skalierungsfaktor für die Größe des Einflussbereichs der Funktion [36].

⁵Ashesi University: <http://www.ashesi.edu.gh/>

Der betrachtete Punkt des Potenzialfeldes ist q . Eine klassische abstoßende Funktion wird durch Gleichung (5.15) angegeben:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 & \text{wenn } \rho(q) \leq \rho_0 \\ 0 & \text{wenn } \rho(q) > \rho_0. \end{cases} \quad (5.15)$$

Die minimale Distanz zwischen Roboter und Hindernis wird durch $\rho(q)$ festgelegt, η ist eine positive skalierende Konstante und ρ_0 beschreibt den Einflussbereich der Funktion [36]. Die beiden Funktionen (5.14) und (5.15) sind typische Beispiele, darüber hinaus können aber im Prinzip beliebige Funktionen verwendet werden.

Die Bahn wird ausgehend von einem beliebigen Startpunkt berechnet, indem sie schrittweise zum geringsten angrenzenden Potenzial verläuft. Dieses Verfahren kann als Gradientenabstieg aufgefasst werden. Damit das Bahnplanungsproblem eindeutig lösbar ist, muss sichergestellt sein, dass der Zielpunkt das geringste Potenzial des gesamten Feldes aufweist. Außerdem dürfen keine lokalen Minima existieren. Diese können von der Bahn nicht durchlaufen werden, da die umliegenden Potenziale größere Werte haben und somit kein absteigender Gradient für den nächsten Schritt vorhanden ist.

In der Projektgruppe DynOLog wird ein Potenzialfeld zur Bahnplanung eingesetzt. Die Bahngeschwindigkeit des Roboters variiert in den verschiedenen Bereichen des Arbeitsraumes. Daher ist es zweckmäßig, die Gebiete bei der Bahnplanung zu bevorzugen, die eine höhere Maximalgeschwindigkeit ermöglichen. So wird nicht die Kürzeste, sondern die schnellste Bahn angestrebt, da der gesamte Be- und Entladevorgang zeitkritisch ist. Dies ist mit zellbasierten Methoden nicht ohne umfangreiche Adaptionen möglich. Potenzialfelder erlauben es hingegen, alle Bereiche des Feldes beliebig zu gewichten und so eine Priorisierung bestimmter Gebiete umzusetzen. Außerdem ist die Berechnung der Bahn relativ effizient.

Die projektgruppenspezifische Realisierung wird in Abschnitt 7.5.1 ausführlich erläutert.

6. Prädiktion von Positionsdaten

Im folgenden Kapitel sollen Techniken vorgestellt werden, die die Vereinigung der erfassten Sensordaten durch Verfahren, wie in Kapitel 3 und 4 erläutert, realisieren. Gleichzeitig soll eine Prädiktion von Position und Orientierung des Paketes ermöglicht werden. Zunächst wird in Kapitel 6.1 die Problemstellung erläutert. Im Folgenden beschäftigt sich Kapitel 6.2 mit stochastischen Trackingverfahren, die für die Anwendung im Kontext geeignet wären. Abschließend werden in Kapitel 6.3 noch einige alternativen Ansätze aufgeführt.

6.1. Problemstellung

Um die Gesamtaufgabe (siehe Kapitel 1) zu lösen, wurden in Kapitel 3 und 4 Verfahren zur Ermittlung der Position des Paketes/Fahrzeuges erläutert. Da diese Informationen vereinigt werden müssen und das Gesamtsystem auch mit verrauschten Sensordaten umgehen können soll, wird ein Tracking benötigt, welches eben genau diese beiden Teilprobleme lösen kann: die Vereinigung der Daten und die Verminderung der Fehler durch Sensorrauschen.

Um die Auswertung der 3D-Kameradaten vornehmen zu können, muss sich die Kamera/der Sauggreifer immer direkt über dem Paket befinden. Dies ist allein durch eine Positionsverfolgung nicht zu erreichen, da der Roboter eine nicht zu vernachlässigende Zeit benötigt, um eine gegebene Position anzufahren. Mit einer Positionsverfolgung würde er dann nie direkt über dem Paket fahren, sondern immer etwas versetzt dahinter. Daher muss hier eine Prädiktion durchgeführt werden. Auch im finalen Greifprozess, indem die Sensoren aufgrund von Verdeckungen durch den Roboter lediglich noch eingeschränkt Messdaten liefern können, wird die Prädiktion benötigt, um ein sicheres und zuverlässiges Greifen zu erreichen.

Da die Bewegung des Roboters simultan zu der des Fahrzeuges erfolgen muss, ist die Echtzeitfähigkeit des Tracking- und Prädiktionsalgorithmus eine wichtige Eigenschaft.

Im Folgenden werden einige Verfahren aufgeführt, die Lösungen für diese Probleme bereitstellen. Dabei wird der Schwerpunkt auf stochastische Verfahren gelegt, die eine Prädiktion leisten können.

6.2. Stochastische Tracking-Verfahren

Viele modellbasierte Trackingsysteme bedienen sich stochastischen Verfahren, da diese häufig sehr effizient sind und somit die Echtzeitfähigkeit des Systems garantieren [37]. Im Allgemeinen lassen sich diese Verfahren in zwei Schritte unterteilen, die in jedem Zeitintervall wiederholt werden: Die Prädiktion und die Korrektur.

Während der Prädiktionsphase werden die zu trackenden Eigenschaftswerte für den nächsten

Zeitpunkt vorausgesagt. Dabei wird das vorhandene Modell mit Bewegungsparametern aus vorherigen Schritten angewendet. So wird beispielsweise die Position des Fahrzeugs nach dem nächsten Zeitintervall geschätzt, indem das Bewegungsmodell mit Richtung und Geschwindigkeit der bisherigen Bewegung des Objektes ausgewertet wird. Nach der Prädiktion folgt im nächsten Zeitintervall dann die Korrektur. Hier werden die tatsächlichen Eigenschaften des Fahrzeugs gemessen und mit Hilfe dieser Werte die Voraussage der Prädiktionsphase korrigiert. Diese Messungen können und sollten die Prädiktion als Hypothese verwenden, um effizienter berechnet werden zu können. Im genannten Beispiel wird jetzt die tatsächliche Position des Fahrzeugs gemessen und mit dieser Position die vorausgesagte Position korrigiert. Bei der Suche des Fahrzeugs im Bild kann die Position der Prädiktion verwendet werden, um den Suchbereich einzuschränken und somit die Effizienz zu verbessern.

In den folgenden Unterkapiteln wird näher auf gängige Verfahren dieser Kategorie eingegangen. Zunächst wird in Abschnitt 6.2.1 mit dem Kalman-Filter ein klassisches Verfahren vorgestellt. Des Weiteren werden in Abschnitt 6.2.2 Partikel-Filter behandelt. Dabei wird insbesondere auf den Condensation-Algorithmus eingegangen.

6.2.1. Kalman-Filter

Der Kalman-Filter stellt ein klassisches stochastisches Verfahren dar, welches zum Tracken von Objekten verwendet werden kann [38]. Dem Verfahren liegt ein modifiziertes Hidden Markov Modell zugrunde, welches Prozesszustände in Form von einer Menge mathematischer Gleichungen repräsentiert. Im Kontext des Trackings modelliert so ein Prozess eine Folge von Eigenschaftswerten des zu trackenden Objektes. Damit kann beispielsweise eine Folge von Objektpositionen abhängig von einem Zeitpunkt t repräsentiert werden. Mithilfe des Modells können dann im Folgenden zukünftige Zustände geschätzt werden.

Das Modell wird durch zwei Gleichungen repräsentiert: Die Zustands- und die Messgleichung [38]. In der ersten werden die Abhängigkeiten der Prozesszustände \mathbf{x}_t , die im Anwendungsfall die zu trackende Objekteigenschaft modellieren, durch die Gleichung

$$\mathbf{x}_t = \mathbf{A}_{t-1} \cdot \mathbf{x}_{t-1} + \mathbf{B}_{t-1} \cdot \mathbf{u}_{t-1} + \mathbf{w}_{t-1} \quad (6.1)$$

definiert. Die Matrix \mathbf{A}_{t-1} repräsentiert hierbei die lineare Transformation, die auf die Objekteigenschaft während des Zeitintervalls zwischen Zeitpunkt $t - 1$ und t wirkt, während der zweite und der dritte Summand äußere Einwirkungen auf das System modellieren. \mathbf{u}_{t-1} ist hierbei eine deterministische Störung, die durch die Matrix \mathbf{B}_{t-1} linear transformiert werden kann. Der Summand \mathbf{w}_{t-1} repräsentiert zufälliges Rauschen im Zustandsübergang. Dabei wird angenommen, dass \mathbf{w}_{t-1} normalverteilt ist mit Erwartungswert 0 und einer Kovarianzmatrix \mathbf{Q} . Als Folge dessen sind auch die Zustände \mathbf{x}_t normalverteilt jeweils um den Erwartungswert $\mathbf{A}_{t-1} \cdot \mathbf{x}_{t-1} + \mathbf{B}_{t-1} \cdot \mathbf{u}_{t-1}$. Des Weiteren ist anzumerken, dass ein Zustand \mathbf{x}_t lediglich vom letzten Zustand \mathbf{x}_{t-1} abhängig ist und die Beziehung der aufeinander folgenden Zustände auf lineare Zusammenhänge beschränkt ist. Die zweite Gleichung des Modells ist die Messgleichung. Hierbei wird die Folge von Messungen \mathbf{z}_t durch die Gleichung

$$\mathbf{z}_t = \mathbf{H}_t \cdot \mathbf{x}_t + \mathbf{v}_t \quad (6.2)$$

definiert. Eine Messung \mathbf{z}_t zum Zeitpunkt t ist von der zu messenden Eigenschaft \mathbf{x}_t abhängig, die gegebenenfalls durch die Matrix \mathbf{H}_t linear transformiert werden kann. Außerdem wird hier durch \mathbf{v}_t ein normalverteiltes Messrauschen mit Kovarianzmatrix \mathbf{R} modelliert.

Das Modell bestehend aus oben genannten Gleichungen kann im Folgenden verwendet werden, um die zukünftige Objekteigenschaft \mathbf{x}_t , gegeben der vorher berechneten Zustände \mathbf{x}_1 bis \mathbf{x}_{t-1} , im Prädiktionsschritt zu schätzen und anschließend durch die Messung \mathbf{z}_t zu korrigieren [38]. Sei der geschätzte, unkorrigierte Zustand zum Zeitpunkt t definiert als $\bar{\mathbf{x}}'_t$ und der geschätzte, korrigierte Zustand nach dem Verwenden der Messung zur Korrektur definiert als $\bar{\mathbf{x}}_t$. Außerdem sei \mathbf{P}'_t die Fehler-Kovarianzmatrix des Prädiktionsfehlers $\mathbf{x}_t - \bar{\mathbf{x}}'_t$ und \mathbf{P}_t die Fehler-Kovarianzmatrix des Fehlers nach Korrektur $\mathbf{x}_t - \bar{\mathbf{x}}_t$. Dann lässt sich die Prädiktion durch die Formeln

$$\bar{\mathbf{x}}'_t = \mathbf{A}_{t-1} \cdot \bar{\mathbf{x}}_{t-1} + \mathbf{B}_{t-1} \cdot \mathbf{u}_{t-1} \quad \text{und} \quad (6.3)$$

$$\mathbf{P}'_t = \mathbf{A}_{t-1} \cdot \mathbf{P}_{t-1} \cdot \mathbf{A}_{t-1}^T + \mathbf{Q} \quad (6.4)$$

leisten. Mit Hilfe der ersten Gleichung wird dabei anhand des Modells die Schätzung der zukünftigen Objekteigenschaft berechnet. Die zweite Gleichung berechnet die Kovarianzmatrix des Fehlers, der bei der Schätzung von $\bar{\mathbf{x}}'_t$ gemacht wurde [38].

In der darauffolgenden Korrekturphase wird zunächst der *Kalman-Gain* \mathbf{K} berechnet. Diese Matrix wird so gewählt, dass sie die in der nachfolgenden Korrektur entstehende Fehlervarianz \mathbf{P}_t minimiert [38]. Dies kann durch die Formel

$$\mathbf{K}_t = \mathbf{P}'_t \cdot \mathbf{H}_t^T \cdot (\mathbf{H}_t \cdot \mathbf{P}'_t \cdot \mathbf{H}_t^T + \mathbf{R})^{-1} \quad (6.5)$$

geschehen. Im Folgenden kann die Korrektur der Zustandsschätzung mit Hilfe der Formel

$$\bar{\mathbf{x}}_t = \bar{\mathbf{x}}'_t + \mathbf{K}_t \cdot (\mathbf{z}_t - \mathbf{H}_t \cdot \bar{\mathbf{x}}'_t) \quad (6.6)$$

stattfinden. Der Faktor $(\mathbf{z}_t - \mathbf{H}_t \cdot \bar{\mathbf{x}}'_t)$ repräsentiert dabei den Wert der *Innovation*, welcher die Abweichung einer aus dem geschätzten Zustand berechneten Messung sowie der realen Messung angibt. Als letzter Schritt wird noch die Fehler-Kovarianzmatrix \mathbf{P}_t des Fehlers nach der Korrektur mit Hilfe der Formel

$$\mathbf{P}_t = (\mathbf{E} - \mathbf{K}_t \cdot \mathbf{H}_t) \cdot \mathbf{P}'_t \quad (6.7)$$

berechnet. Dabei steht \mathbf{E} für die Einheitsmatrix. Die Fehler-Kovarianzmatrix liefert Informationen über die Güte des geschätzten und korrigierten Zustands und wird in der nächsten Iteration des Verfahrens weiter verwendet (siehe Gleichung 6.4) [38].

Ein Beispiel für die Anwendung des Kalman-Filters zeigt Abbildung 6.1 .

Es wird die zweidimensionale Position eines Objektes getrackt. Die grünen Kreuze symbolisieren die tatsächlichen Positionen des Objekts, während die roten kleinen Kreise die Prädiktion repräsentieren. Die blauen Kreuze sind die Messungen, mit Hilfe derer die korrigierten Positio-

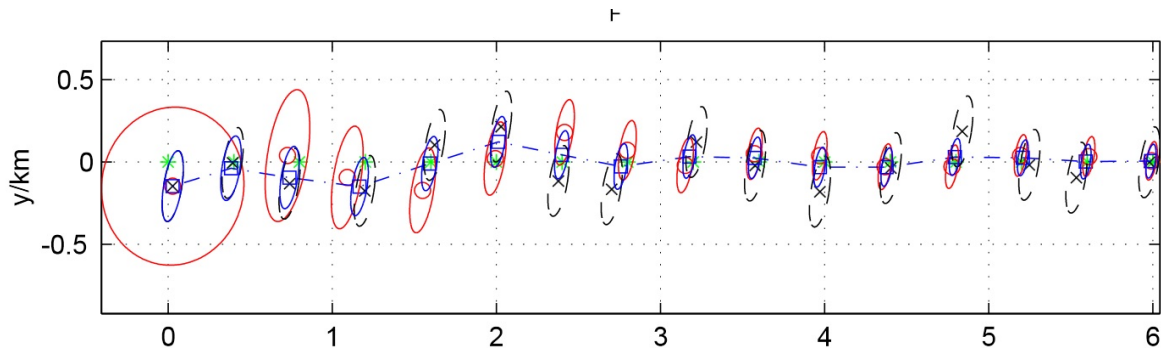
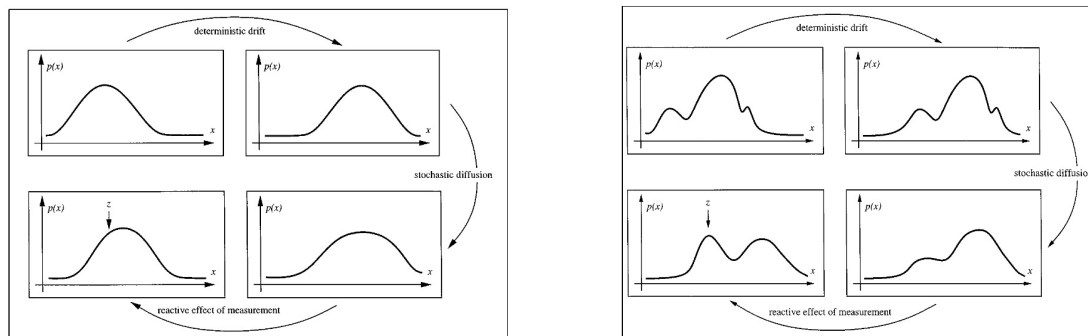


Abbildung 6.1.: Tracking einer zweidimensionalen Objektposition mit Hilfe des Kalman-Filters. Die grünen Kreuze symbolisieren die tatsächlichen Positionen des Objekts, während die roten kleinen Kreise die Prädiktion repräsentieren. Die blauen Kreuze sind die Messungen, mit Hilfe derer die korrigierten Positionen, die durch blaue Quadrate gezeigt werden, berechnet werden. Die Ellipsen geben jeweils die Unsicherheit der Schätzungen nach Prädiktion und Korrektur an [39].



(a) Änderung der Normalverteilung während einer Tracking-Iteration (b) Änderung einer beliebigen Verteilung während einer Tracking-Iteration

Abbildung 6.2.: Unterschiede bei einer Tracking-Iteration mit verschiedenen Verteilungen [37]

nen, die durch blaue Quadrate gezeigt werden, berechnet werden. Die Ellipsen geben jeweils die Unsicherheit der Schätzungen nach Prädiktion und Korrektur an.

6.2.2. Partikel-Filter

Bei Analyse des Kalman-Filters fällt auf, dass das verwendete Modell einige Einschränkungen macht. So können lediglich lineare Bewegungsmodelle dargestellt werden und die Verteilung der Zustände ist immer normalverteilt [38]. Beide Einschränkungen könnten in gegebenen Kontext Nachteile darstellen [37]. Während sich die Positionen, Rotationen und Geschwindigkeiten auch nicht linear ändern könnten, ist die zweite Einschränkung problematisch bei Überdeckungen, abrupten Bewegungen des Objektes und bei mehreren ähnlichen Objekten in der Szene. Abbildung 6.2 stellt die Unterschiede von Normalverteilungen und beliebigen Verteilungen grafisch dar.

Während Transformationen der Objekteigenschaften und abweichende Messungen bei Normalverteilungen lediglich den Erwartungswert verschieben, können in beliebigen Verteilungen

weitere Maxima entstehen [37]. Dementsprechend könnte das Tracking mit beliebigen Verteilungen genauer und robuster sein. Gegeben ein Szenario, in dem mehrere ähnliche Objekte nah bei einander liegen, können die einzelnen Objekte damit differenziert betrachtet werden. Um beliebige Transformationen und beliebige Verteilungen zu ermöglichen, muss das verwendete Modell verallgemeinert werden. Eine Möglichkeit, um dies zu erreichen, ist die Verwendung von folgenden Gleichungen für Zustandsprozess und Messungen:

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}) + \mathbf{w}_{t-1}, \quad (6.8)$$

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v}_t. \quad (6.9)$$

Die Funktionen g und h sind hier nicht weiter eingeschränkt und die Sequenzen \mathbf{w}_t und \mathbf{v}_t sind beliebig zufällig verteilt. Zur Schätzung beliebiger Verteilungen werden unter anderem Partikel-Filter verwendet. Hierbei handelt es sich um Monte-Carlo-Ansätze, die die gesuchten Verteilungen approximieren. Ein Beispiel hierfür ist der Condensation-Algorithmus [37]. Dabei handelt es sich um eine *Monte-Carlo-Approximation* mit *Importance Sampling*. Im folgenden Unterkapitel wird der Algorithmus näher erläutert.

6.2.3. Condensation-Algorithmus

Der Name Condensation steht für *Conditional Density Propagation* und beschreibt den Kern des Algorithmus [37]. Gegeben die Wahrscheinlichkeitsverteilung $p(\mathbf{x}_{t-1})$ des Zustands \mathbf{x}_{t-1} aus der vorherigen Tracking-Iteration soll mit Hilfe der Messung und deterministischen Änderung des Zustands die Verteilung $p(\mathbf{x})$ des nächsten Zustands propagiert werden. Es wird vorausgesetzt, dass es sich bei der Folge $(\mathbf{x}_k)_{k \in \mathbb{N}}$ um eine Markov-Kette handelt. Der Zustand \mathbf{x}_t ist also lediglich von seinem Vorgänger \mathbf{x}_{t-1} abhängig:

$$p(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}) = p(\mathbf{x}_t | \mathbf{x}_{t-1}). \quad (6.10)$$

Zur Schätzung der Verteilungsfunktion wird eine Monte-Carlo-Approximation verwendet. Anstatt die gesamte Funktion zu schätzen, wird sie lediglich an N verschiedenen, zufälligen Punkten bestimmt. Diese Punkte dienen dann als Eingabe für die nächste Iteration. Verwendet werden sie für ein *Importance Sampling*. Dabei werden die neuen Punkte nicht nur zufällig bestimmt, sondern mit Hilfe einer bereits vorhandenen Wahrscheinlichkeitsverteilung. Dadurch wird die Wahrscheinlichkeit Punkte zu wählen, die einen hohen Informationsgehalt über die zu approximierende Funktion liefert, erhöht, wenn die für das *Importance Sampling* verwendete Funktion der zu approximierenden ähnelt. Da sich die Zustände pro Iteration lediglich im geringen Maße ändern, ist diese Voraussetzung erfüllt. Eine Iteration des Algorithmus lässt sich in drei Schritte unterteilen [37]:

1. Sampling: Auswahl von N Punkten $\bar{\mathbf{s}}_t^i$, $i \in \{1, \dots, n\}$ mit Hilfe der Punkte \mathbf{s}_{t-1}^i aus der letzten Iteration.
2. Prädiktion: Anwenden des Modells auf die N ausgewählten Punkte $\bar{\mathbf{s}}_t^i$, um Schätzungen \mathbf{s}_t^i zu berechnen.

3. Korrektur: Gewichten der Punkte \mathbf{s}_t^i anhand der Wahrscheinlichkeit, dass die gemachte Messung \mathbf{z}_t aus der tatsächlichen Eigenschaft \mathbf{s}_t^i resultieren würde.

Zum Durchführen des Samplings kann beispielsweise das Intervall $[0, 1]$ anhand der normierten, kumulativen Gewichte der Punkte aus vorheriger Iteration in Teilintervalle aufgeteilt werden. Im Folgenden wird N -mal eine Zahl r gleichverteilt aus dem Intervall $[0, 1]$ gezogen und der Punkt \mathbf{s}_{t-1}^i , dessen Teilintervall getroffen wurde, als jeweils neuer Punkt verwendet. Die neuen Punkte $\bar{\mathbf{s}}_t^i$ werden also aus den alten Punkten \mathbf{s}_{t-1}^i ihrer Gewichtung nach gewählt. Die Prädiktion besteht aus der Anwendung des Modells auf die gewählten Punkte. In dem oben genannten allgemeinen Modell würden Punkte \mathbf{s}_t^i gemäß der Formel

$$\mathbf{s}_t^i = g(\bar{\mathbf{s}}_t^i) + \mathbf{w}_{t-1} \quad (6.11)$$

berechnet werden. Für die Korrektur anhand der Messung muss zunächst ein Messungsmodell definiert werden. Dies sollte in Form einer Wahrscheinlichkeitsverteilung der möglichen Messungen gegeben sein. Eine solche Verteilung lässt sich aus Gleichung 6.9 bestimmen, wenn die Parameter bekannt sind [37]. Ist die Verteilung vorhanden, lassen sich die Gewichte π_t^i der Punkte \mathbf{s}_t^i durch Anwenden dieser mit

$$\pi_t^i = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^i) \quad (6.12)$$

bestimmen. \mathbf{z}_t ist dabei die tatsächlich gemachte Beobachtung. Es wird angenommen, dass \mathbf{s}_t^i die tatsächliche Objekteigenschaft ist, und die Wahrscheinlichkeit, dass die gemachte Messung unter dieser Bedingung auftritt, wird berechnet. Diese Wahrscheinlichkeit wird als Gewicht π_t^i gespeichert. Anschließend werden alle Gewichte normiert, damit sie sich zu 1 aufsummieren. Im Folgenden ist es möglich, die Gewichte direkt kumulativ zu speichern. Es ergibt sich eine Aufteilung des Intervalls $[0, 1]$, die in der nächsten Iteration für das Sampling verwendet werden kann. Abbildung 6.3 stellt den Ablauf des Condensation-Algorithmus grafisch dar. Untereinander werden die drei Schritte Sampling, Prädiktion und Korrektur gezeigt.

Nach Durchführung der drei Schritte kann in jeder Iteration ein Wert der Objekteigenschaft auf verschiedene Weisen berechnet werden [37]. So ist es möglich, den als höchsten gewichteten Punkt als tatsächliche Objekteigenschaft anzunehmen. Auch ein gewichtetes Mittel aus allen Punkten wäre denkbar. Dies würde eine Normalverteilung modellieren. Des Weiteren ist es möglich, alle Maxima der Verteilung über einem bestimmten Schwellwert zu bestimmen. Mit dem Resultat ließen sich mehrere mögliche Eigenschaftswerte bestimmen. Als Beispiel könnten diese Werte beim Positions-Tracking von mehreren Objekten genutzt werden, um die verschiedenen Positionen zu repräsentieren.

Unter den betrachteten Verfahren zeichnete sich der Condensation-Algorithmus als die beste Lösung für die gegebene Problemstellung ab. Dabei spielen die Möglichkeit zur Umsetzung von nicht linearen Bewegungsmodellen sowie die Robustheit des Verfahrens gegenüber falschen Messungen entscheidende Rollen. Außerdem ist eine einfache, effiziente Implementierung der einzelnen Schritte möglich, da sie für sich betrachtet übersichtlich und parallelisierbar sind. Im Lösungsansatz des Problems wird deshalb für die Prädiktion der Objekteigenschaften der Condensation-Algorithmus verwendet. Realisierungsdetails werden in Kapitel 7.4 näher beschrieben. Dabei werden unter anderem die deterministische und nicht-deterministische Transformation sowie die Berechnung der Beobachtungsfunktion näher ausgeführt.

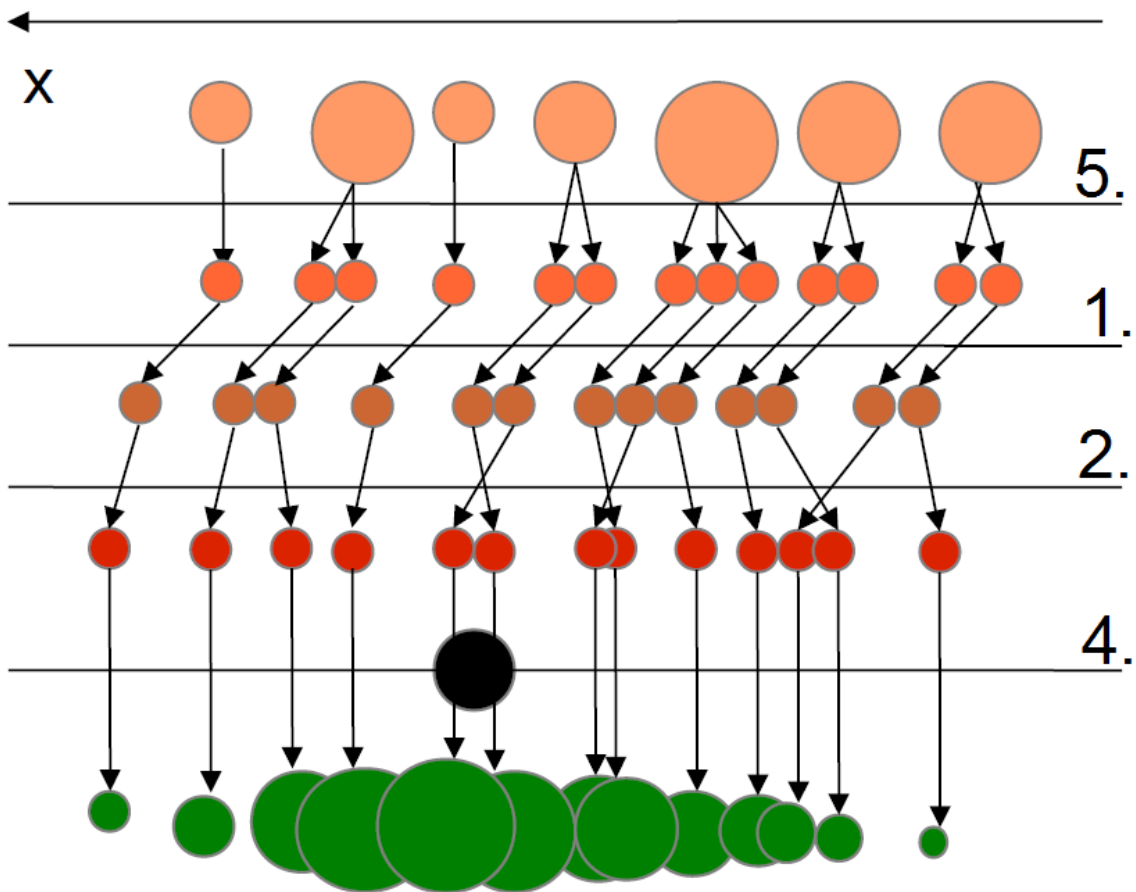


Abbildung 6.3.: Grafische Darstellung des Condensation-Algorithmus. Untereinander werden die drei Schritte Sampling, Prädiktion und Korrektur gezeigt. Schritt 5 ist das Sampling der Partikel, während Schritt 1 die deterministische Transformation zeigt. In Schritt 2 werden die Partikel nicht-deterministisch verschoben, bevor in Schritt 4 anhand der Beobachtungsgleichung die Samples gewichtet werden.

6.3. Alternative Ansätze

Neben den vorgestellten stochastischen Verfahren zum Tracken von Objekten existieren einige deterministische Verfahren, um Objekte in einer Bildsequenz zu erkennen und zu verfolgen. Zu nennen ist hier das Tracking mit *Aktiven Konturen*. Dabei werden die Kanten eines Objektes im Bild erkannt und verfolgt. Intuitiv wird eine bewegbare Kontur definiert, die *Snake*, die sich um das zu trackende Objekt herum an die Kanten legt und sich mit dem Objekt über eine Bildsequenz hinweg bewegt [40]. Die Positionierung der Snake erfolgt über die Minimierung einer Energiefunktion. Diese kann verschiedene Teilformeln enthalten, um das Verhalten der Snake zu beeinflussen. Unterscheiden lassen sich diese Teilformeln zwischen Teilformeln der internen und der externen Energie:

- **Interne Energie:** Bewertet die Struktur der Snake unabhängig vom Bild. Hier können Informationen über das zu trackende Objekt eingebracht werden.

- Externe Energie: Liefert Kriterien für die Bewertung von Bildpunkten. Punkte, an denen sich die Snake befinden soll, werden niedrig bewertet.

Wird die Snake durch eine Parametrisierung $v(s) = (x(s), y(s))$, $s \in [0, 1]$ repräsentiert, lässt sich die Energie der Snake formal durch

$$E_{snake} = \int_0^1 E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s)) ds \quad (6.13)$$

beschreiben [40]. Dabei repräsentiert E_{int} die interne Energie, während E_{image} und E_{con} die externe bilden. E_{image} bewertet die oben beschriebenen Bildpunkteigenschaften. Als mögliches Beispiel kann hier die Gradientenfunktion des Bildes verwendet werden:

$$E_{image}(v(s)) = -|\nabla I(v(s))|^2 . \quad (6.14)$$

Die Minimierung der Energiefunktion liefert dann eine optimale Snake für ein gegebenes Bild. Eine weitere Möglichkeit zur Verfolgung von Objekten in einer Bildsequenz bietet das Tracking mittels *Support Vector Machines* [41]. In einem Bild wird das zu trackende Objekt mittels einer Reihe von SVN-Klassifikationen gesucht und verfolgt. Beim SVN-Tracking stellt der erste Schritt das Lernen eines SVN-Modells mit Bildausschnitten dar. Dabei sollten sowohl Teilbilder gelernt werden, die das zu trackende Objekt enthalten, als auch Bilder mit anderen Objekten oder ohne Objekte. Alle gelernten Objekte sollten im Vorhinein manuell klassifiziert worden sein.

Beim Durchführen des Trackings mittels des gelernten Modells wird vorausgesetzt, dass die Position des zu trackenden Objekts durch eine Initialisierung zu Beginn bekannt ist. Dies kann auf verschiedene Arten durch ein Initialisierungsmodul geleistet werden [41]. Liegt eine Hypothese für den Zeitpunkt t_0 fest, kann aufbauend auf dieser die Position zum nächsten Zeitpunkt geschehen. Dabei werden Bildausschnitte in der lokalen Umgebung der letzten Hypothese mittels des SVN-Modells klassifiziert. Der Bildausschnitt mit dem besten Klassifikationsergebnis wird als neue Hypothese verwendet.

Beide vorgestellten deterministischen Verfahren liefern Methoden zur Lokalisierung der Objekte in einem Bild, unter Berücksichtigung vorhergegangener Positionen in der Bildsequenz. Eine Prädiktion der Objekteigenschaften findet nicht statt. Sie eignen sich aus diesem Grund nur bedingt für die gegebene Problemstellung. Sie müssten bei Anwendung durch eine solche Prädiktion gemäß der stochastischen Verfahren erweitert werden. Es hat sich außerdem herausgestellt, dass die Objektposition mithilfe der Daten aus CCD- und Tiefenkamera auch ohne solche Verfahren effizient erkannt werden kann. Eine Miteinbeziehung der vorherigen Objektposition ist nicht notwendig, wodurch die vorgestellten deterministischen Verfahren zur Objektverfolgung im Anwendungsfall nicht benötigt werden.

7. Automatisierung von Entladungsvorgängen

Dieses Kapitel ist in sechs große Abschnitte eingeteilt und erläutert den gesamten automatisierten Be- und Entladungsvorgang im Detail. Der erste Abschnitt dient der Übersicht über den Ablauf im Gesamtsystem. Die darauf folgenden vier Abschnitte umfassen die einzelnen Module, welche zur Realisierung des Gesamtsystems erforderlich sind. Diese vier Module werden im weiteren Verlauf als *CCD-Modul*, *Tiefenkamera-Modul*, *Tracking-Modul* und *Roboter-Modul* bezeichnet. Zur Illustration dieser sei die Abbildung 7.1 gegeben, welche den gesamten Be- und Entladungsvorgang auf logischer Ebene als eine Art Verarbeitungspipeline darstellt.

Zunächst wird in Abschnitt 7.2 die initiale Fahrzeugerkennung durch das CCD-Modul und deren Positionsbestimmung mittels zweidimensionaler optischer Methoden erläutert, welche die Teilschritte „CCD-Kamerabild“, „Detektion der FTF-Position“ sowie die beiden entsprechenden zugehörigen Teilschritte „Kalibrierung“ und „Koordinatentransformation“ beinhaltet. Abschnitt 7.3 behandelt anschließend die durch das Tiefenkamera-Modul erfolgende, exakte Erkennung des Paketes auf dem FTF mittels dreidimensionaler optischer Methoden und erläutert dazu die Teilschritte „3D-Kamerabild“, „Detektion der Paketposition“ und die entsprechenden zugehörigen Teilschritte „Kalibrierung“ sowie „Koordinatentransformation“. Der darauffolgende Abschnitt 7.4 behandelt die Prädiktion der Paketposition zu späteren Zeitpunkten mit Hilfe bisher gesammelter Sensordaten durch das Tracking-Modul und erläutert die Teilschritte „Sensordatenfusion“, „Prädiktion der Paketposition“ sowie „Fahrzeugbahn“. Abschnitt 7.5 behandelt die Ansteuerung des Roboters zur Entnahme des Paketes aus dem fahrenden FTF durch das Roboter-Modul und konkretisiert die Teilschritte „Bahnplanung“, „Ansteuerung des Roboters“ sowie „Position des TCP“. Im abschließenden Abschnitt 7.6 wird dann das Zusammenwirken der zuvor beschriebenen Module für die Realisierung der Paketentnahme durch den Roboter genauer beschrieben.

7.1. Ablauf des Entladevorgangs

Zur Realisierung des gesamten automatisierten Entladevorgangs von Paketen müssen die vier genannten Module in folgenden Schritten ablaufen:

- Durch das CCD-Modul werden fortlaufend Bilder analysiert und ermittelte Positionen über ein Signal an das Tracking-Modul gesendet. Wird im Fahrzeug-Einfahrts-Bereich in mehreren aufeinander folgenden Bildern ein Fahrzeug erkannt, so wird durch das CCD-Modul ein Signal für den Start des Vorgangs emittiert, wodurch das Tracking-Modul mit der Ausführung des Condensation-Algorithmus beginnt und das Roboter-Modul regelmäßig über ein Signal eine Prädiktion der Fahrzeugposition für die nahe Zukunft beim Tracking-Modul anfordert.

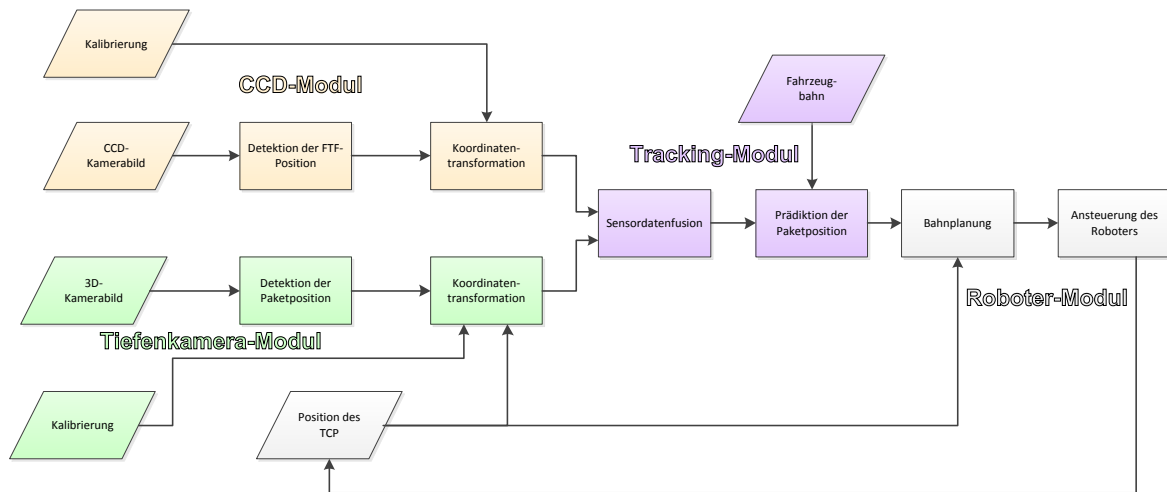


Abbildung 7.1.: Illustration der Einzelschritte des automatisierten Entladungsvorgangs

- Das Tracking-Modul berechnet nun fortlaufend die Schätzung der aktuellen Fahrzeugposition und antwortet über ein weiteres Signal auf die regelmäßigen Prädiktionsanfragen des Roboter-Moduls, welches den Roboter daraufhin eine Stelle über der prognostizierten Position anfahren lässt.
- Das Tiefenkamera-Modul führt ebenfalls fortlaufend die Erkennung durch. Befindet sich der Roboterarm oberhalb des Fahrzeugs, so wird ein Paket erkannt. Für die Umrechnung der Position in das Weltkoordinatensystem wird die aktuelle Position des Roboterarms benötigt und daher ein Signal ausgelöst, das mit einem Slot des Roboter-Moduls verknüpft ist. Dieses emittiert dann seinerseits ein Signal mit der Roboterarmposition zum Zeitpunkt der Tiefenkamera-Bildaufnahme, welche durch das Tiefenkamera-Modul für die Berechnung der Weltposition genutzt wird. Die berechnete Position wird über ein Signal an das Tracking-Modul weitergegeben. Sobald dem Tracking-Modul Eingaben des Tiefenkamera-Moduls vorliegen, werden diese für die aktuelle Schätzung und für Prädiktionen berücksichtigt.
- Das Tracking-Modul vergleicht fortlaufend Prädiktionen, die zu einem früheren Zeitpunkt erstellt wurden und sich auf den jetzigen Zeitpunkt beziehen, mit der aktuellen Schätzung. Die Abweichung zwischen beiden Werten wird über ein Signal an das Roboter-Modul gemeldet.
- Das Roboter-Modul glättet die Abweichungen, die durch das Tracking-Modul übermittelt wurden, exponentiell. Wenn dieser Wert eine Schwelle unterschreitet und eine Mindestzeit seit dem Start des Trackings verstrichen ist, wird der Greifvorgang eingeleitet. Dazu werden zwei weitere Prädiktionen für Zeitpunkte mit kurzem zeitlichen Abstand voneinander beim Tracking-Modul angefordert und diese zu den entsprechenden Zeitpunkten auf der Höhe des Pakets angefahren. An der ersten Position wird der Sauggreifer aktiviert. Im Anschluss erfolgt eine Fahrt des Roboterarms zur Paket-Ablageposition. Dort wird der Sauggreifer deaktiviert und das System in den Anfangszustand zurückgesetzt.



Abbildung 7.2.: Anbringung der CCD-Kamera im Roboterkäfig

7.2. Fahrzeugerkennung und Positionsbestimmung

Zur automatischen Be- und Entladung muss das FTF zunächst im Roboterbewegungsbereich optisch erkannt und dessen dreidimensionale Position im Roboterkoordinatensystem bestimmt werden. Nach Bestimmung der Fahrzeugposition kann der Roboterarm das FTF nun anfahren und durch die dreidimensionale Paketerkennung eine genauere Position des Paketes auf dem FTF ermitteln. Abschnitt 7.2.1 behandelt zunächst die zur optischen Erkennung verwendete Hardware, sowie die Kalibrierung und Entzerrung der entstandenen Bilder. In Abschnitt 7.2.2 wird die zweidimensionale Approximation der Fahrzeugposition im aufgenommenen Bild dargestellt und letztlich wird in Abschnitt 7.2.3 die Transformation der FTF-Bildposition ins 3D-Roboterkoordinatensystem erläutert.

7.2.1. Bildakquisition

Im Folgenden wird zunächst in Abschnitt 7.2.1 die verwendete Hardware vorgestellt. Dazu werden technische Daten genannt sowie die Anbringung und Positionierung im Roboterbewegungsbereich dargestellt und erklärt. Abschnitt 7.2.1 erläutert die Kalibrierung der Kamera mittels des Verfahrens nach *Zhang*, sowie die anschließende Entzerrung der aufgenommenen Bilder.

Verwendete Hardware

Zur optischen Erkennung des einfahrenden FTFs wird eine CCD-Kamera (Charge-Coupled Device-Kamera) der Firma *The Imaging Source Europe GmbH*¹ mit der Modellbezeichnung *DFK 31BU03.H* verwendet. Diese nimmt Farbbilder mit einer Auflösung von 1024×768 Pixeln bei max. 30 Bildern in der Sekunde auf. Tests haben gezeigt, dass eine solche Auflösung ausreichend für die benötigte Präzision der Positionsbestimmung ist (siehe dazu Kapitel 8.2). Weiterhin bieten 30 Bilder pro Sekunde die Möglichkeit der Fahrzeugerkennung in Echtzeit.

Zur Aufnahme des gesamten befahrbaren Bereiches durch das FTF wurde die Kamera an der, dem Roboterarm gegenüberliegenden Ecke, mit optischer Achse in Richtung Roboterarm, angebracht. Die Kamera ist auf einer Höhe von etwa 2.5 Metern befestigt, um sowohl den Eingang als auch den Ausgang des befahrbaren Bereiches durch das FTF aufnehmen zu können. Abbildung 7.2 zeigt die Anbringung der CCD-Kamera im Roboterkäfig (rote Ellipse).

Kalibrierung und Entzerrung

Zur Kalibrierung der verwendeten Kamera wird das Verfahren nach *Zhang* verwendet, welches in Kapitel 3.1.3 als einfaches und kostengünstiges Verfahren zur Kalibrierung vorgestellt wurde. Als Kalibrierkörper wurde dabei ein Schachbrettmuster (wie in Kapitel 3.7 gezeigt) verwendet, dass auf eine planare Fläche angebracht wurde.

Das Kalibrierungsverfahren nimmt für den Zusammenhang eines Punktes im Raum und seinem entsprechenden Bildpunkt das Lochkameramodell, welches bereits in Kapitel 3.1.2 vorgestellt wurde, an. Bei diesem wird ein Raumpunkt \mathbf{x} mittels einer Projektionsmatrix \mathbf{P} auf einen Bildpunkt \mathbf{x}' mittels

$$\lambda \mathbf{x}' = \mathbf{P} \mathbf{x} = \mathbf{K} \mathbf{D} \mathbf{x} \quad (7.1)$$

abgebildet, welche bis auf eine Skalierung λ eindeutig ist. Die Projektionsmatrix \mathbf{P} besteht aus der Matrix der intrinsischen Parameter

$$\mathbf{K} = \begin{pmatrix} f \cdot m_x & s & c_x \\ 0 & f \cdot m_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (7.2)$$

mit Brennweite f , Pixelgröße $\begin{pmatrix} m_x \\ m_y \end{pmatrix}$, Kamera-Hauptpunkt $\begin{pmatrix} c_x \\ c_y \end{pmatrix}$ und Pixel-Scherung s sowie der Matrix der extrinsischen Parameter

$$\mathbf{D} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \quad (7.3)$$

mit der Rotationsmatrix \mathbf{R} und dem Translationsvektor \mathbf{t} . Zur Entzerrung eines aufgenommenen Bildes werden neben den intrinsischen Parametern noch weitere Parameter benötigt, welche die Verzerrung eines aufgenommenen Bildes (siehe Abschnitt 3.1.2) beschreiben. Diese werden im weiteren Sinne ebenfalls zu den intrinsischen Parametern einer Kamera gezählt, da sie unabhängig von der aufgenommenen Szene sind und nur von der internen

¹<http://www.theimagingsource.com>

Geometrie der Kamera abhängen. Zur Modellierung der radialen Verzeichnung werden die radialen Verzerrungskoeffizienten κ_1 und κ_2 benötigt, wobei Koeffizienten höherer Ordnung aufgrund ihres zu vernachlässigen Einflusses ignoriert werden.

Die Berechnung der intrinsischen Parameter inklusive der Verzerrungskoeffizienten wird in drei Schritten durchgeführt. Zunächst wird eine Homographie \mathbf{H} mit Hilfe des oben beschriebenen Lochkameramodells und mindestens vier bekannten Paaren von Punkten im Weltkoordinatensystem und Bildkoordinatensystem geschätzt. Nach Schätzung der Homographie können die intrinsischen sowie extrinsischen Parameter aus der Homographie extrahiert werden. Abschließend werden die Verzerrungskoeffizienten geschätzt und alle Parameter gemeinsam mit Hilfe eines Optimierungsverfahrens optimiert. Im Folgenden werden alle drei Schritte näher beschrieben.

1. Schätzung der Homographie \mathbf{H} :

Zur Schätzung der Homographie \mathbf{H} wird angenommen, dass der Kalibrierkörper in der $z = 0$ -Ebene liegt und die Berechnung somit vereinfacht, ihre Gültigkeit dennoch behält. Mit dem vorgestellten Lochkameramodell stellt sich die Beziehung zwischen einem Punkt $\mathbf{x} = (x \ y \ 0 \ 1)^\top$ im Weltkoordinatensystem und einem Punkt im Bildkoordinatensystem $\mathbf{x}' = (x' \ y' \ 1)^\top$ als

$$\begin{aligned} \lambda \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} &= \mathbf{K} [\mathbf{R} \ \mathbf{t}] \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} \\ &= \mathbf{K} [\mathbf{r}_1 \mathbf{r}_2 \mathbf{r}_3 \ \mathbf{t}] \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} \\ &= \mathbf{K} [\mathbf{r}_1 \mathbf{r}_2 \ \mathbf{t}] \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \end{aligned} \tag{7.4}$$

dar, wobei $\mathbf{r}_1, \mathbf{r}_2$ und \mathbf{r}_3 jeweils Spaltenvektoren der Rotationsmatrix \mathbf{R} darstellen. Der Term $\mathbf{K} [\mathbf{r}_1 \mathbf{r}_2 \ \mathbf{t}]$ lässt sich somit als Homographie \mathbf{H} mit

$$\mathbf{H} = \mathbf{K} [\mathbf{r}_1 \mathbf{r}_2 \ \mathbf{t}] \tag{7.5}$$

darstellen. Die Beziehung zwischen einem Raumpunkt \mathbf{x} und einem Bildpunkt \mathbf{x}' wird somit durch die Formel

$$\lambda \mathbf{x}' = \mathbf{H} \mathbf{x} \tag{7.6}$$

dargestellt. Zur Schätzung der Homographie \mathbf{H} , welche sowohl intrinsische Parameter als auch extrinsische Parameter enthält, werden mindestens vier bekannte korrespondierende Punktpaare aus Raum und Bild benötigt. Zur Verbesserung der Stabilität und Genauigkeit der Kalibrierungsergebnisse wurden in der Praxis zehn verschiedene Aufnahmen aus unterschiedlichen Winkeln verwendet. Aufgrund der nicht exakt ermittelbaren Koordinaten der

Punktpaare, wird anschließend eine Optimierung des mittleren quadratischen Fehlers zwischen den gemessenen Punktpaaren durchgeführt und die Homographie somit optimiert. [10]

Extraktion der intrinsischen und extrinsischen Parameter:

Mit einer gegebenen Homographie ist es möglich, eine Matrix \mathbf{B} mit

$$\begin{aligned} \mathbf{B} &= \lambda \mathbf{K}^{-\top} \mathbf{K}^{-1} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{\alpha^2} & -\frac{s}{\alpha^2\beta} & \frac{c_y s - c_x \beta}{\alpha^2\beta} \\ -\frac{s}{\alpha^2\beta} & \frac{s^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{s(c_y s - c_x \beta)}{\alpha^2\beta^2} - \frac{c_y}{\beta^2} \\ \frac{c_y s - c_x \beta}{\alpha^2\beta} & -\frac{s(c_y s - c_x \beta)}{\alpha^2\beta^2} - \frac{c_y}{\beta^2} & \frac{(c_y s - c_x \beta)^2}{\alpha^2\beta^2} + \frac{c_y^2}{\beta^2} + 1 \end{pmatrix} \end{aligned} \quad (7.7)$$

zu konstruieren und zu berechnen. Die Variablen α und β stellen jeweils die mit der Brennweite f skalierten Pixelgrößen m_x bzw. m_y dar [10]. Diese Matrix enthält alle intrinsischen Parameter so, dass sie nachfolgend in geschlossener Form berechnet werden können. Zur Extraktion können folgende Formeln verwendet werden:

$$\begin{aligned} c_y &= (B_{12}B_{13} - B_{11}B_{23}) / (B_{11}B_{22} - B_{12}^2) \\ \lambda &= B_{33} - [B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23})] / B_{11} \\ \alpha &= \sqrt{\lambda / B_{11}} \\ \beta &= \sqrt{\lambda B_{11} / (B_{11}B_{22} - B_{12}^2)} \\ s &= -B_{12}\alpha^2\beta / \lambda \\ c_x &= s c_y / \alpha - B_{13}\alpha^2 / \lambda. \end{aligned} \quad (7.8)$$

Nach Ermittlung der intrinsischen Parameter und somit bekannter Matrix \mathbf{K} , können die extrinsischen Parameter mit Hilfe der Gleichung 7.5 berechnet werden. Dazu kann diese als

$$\mathbf{H} = [\mathbf{h}_1 \ \mathbf{h}_2 \ \mathbf{h}_3] = \lambda \mathbf{K} [\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{t}] \quad (7.9)$$

dargestellt werden, welche auch schon zur Berechnung der Matrix \mathbf{B} verwendet wurde. Zur Extraktion der extrinsischen Parameter können dann die Formeln

$$\begin{aligned} \mathbf{r}_1 &= \lambda \mathbf{K}^{-1} \mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda \mathbf{K}^{-1} \mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda \mathbf{K}^{-1} \mathbf{h}_3 \end{aligned} \quad (7.10)$$

verwendet werden.

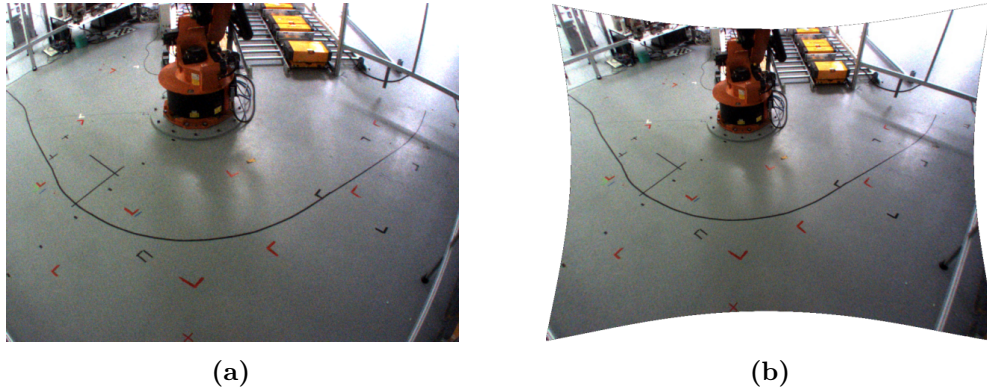


Abbildung 7.3.: Vergleich eines verzerrt aufgenommenen Bildes (a) und eines mittels berechneter Verzerrungskoeffizienten entzerrten Bildes (b)

Schätzung der Verzerrungskoeffizienten und Optimierung:

Nachdem die intrinsischen und extrinsischen Parameter ermittelt wurden, können die Verzerrungskoeffizienten berechnet werden, die es ermöglichen, ein verzerrtes Bild zu entzerren. Zur Modellierung der Verzerrung von Bildkoordinaten werden nicht-lineare Polynome zu den verzerrten Bildkoordinaten in der Form

$$\begin{aligned} x' &= x_d + (x_d - c_x)\delta_x \\ y' &= y_d + (y_d - c_y)\delta_y, \end{aligned} \quad (7.11)$$

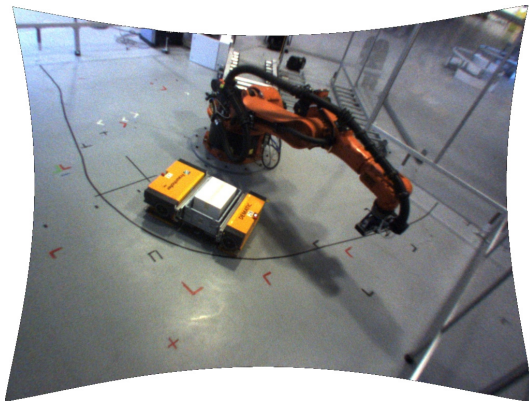
addiert, wobei x', y' die wahren, entzerrten Koordinaten, x_d, y_d die verzerrten Koordinaten und δ_x, δ_y die nicht-linearen Polynome zur radialen Verzerrung darstellen. Die Verzerrungen δ_x und δ_y hängen von der radialen Distanz zum Kamera-Hauptpunkt $(\frac{c_x}{c_y})$ in der Form $r = \sqrt{(x_d^2 + y_d^2)}$ ab und werden durch die Polynome

$$\begin{aligned} \delta_x &= x_d + (\kappa_1 r^2 + \kappa_2 r^4) = x_d + \left[\kappa_1 (x_d^2 + y_d^2) + \kappa_2 (x_d^2 + y_d^2)^2 \right] \\ \delta_y &= y_d + (\kappa_1 r^2 + \kappa_2 r^4) = y_d + \left[\kappa_1 (x_d^2 + y_d^2) + \kappa_2 (x_d^2 + y_d^2)^2 \right] \end{aligned} \quad (7.12)$$

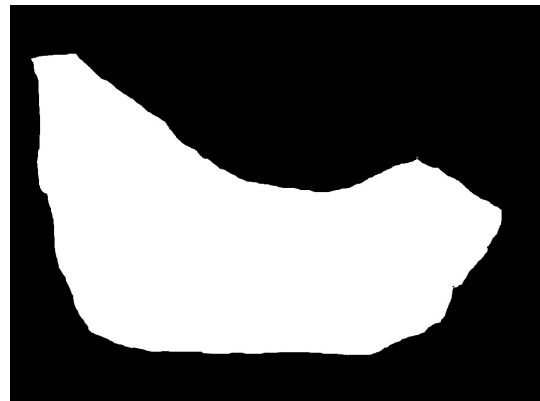
modelliert, wobei höherwertige Terme ignoriert werden und κ_1, κ_2 die radialen Verzerrungskoeffizienten sind. Zur Berechnung der Verzerrungskoeffizienten sowie zur Optimierung dieser und aller intrinsischen und extrinsischen Parameter werden diese gemeinsam mit Hilfe des Levenberg-Marquard-Optimierungsverfahrens in der Form

$$\min_{\mathbf{K}, \mathbf{R}, \mathbf{t}, \kappa_1, \kappa_2} \sum_i \|\mathbf{x}'_i - \mathbf{x}'_i(\mathbf{K}, \mathbf{R}, \mathbf{t}, \kappa_1, \kappa_2)\|^2 \quad (7.13)$$

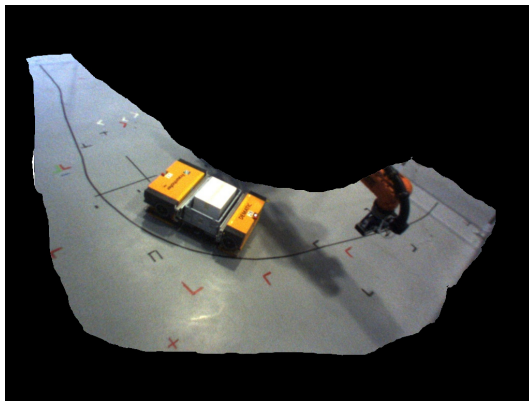
optimiert, wobei $\mathbf{x}'_i(\mathbf{K}, \mathbf{R}, \mathbf{t}, \kappa_1, \kappa_2)$ die Projektion des Weltkoordinatenpunktes \mathbf{x} nach Gleichung 7.1 mit Modellierung der Verzerrung nach Gleichung 7.11 darstellt. Dazu werden für \mathbf{K}, \mathbf{R} und \mathbf{t} die zuvor geschätzten Werte verwendet und die Verzerrungskoeffizienten κ_1 und κ_2 zunächst auf 0 gesetzt.



(a) Entzerrtes Ausgangsbild



(b) Region of Interest



(c) Ergebnis der Suchraumeinschränkung



(d) Ergebnis des Schwellwertverfahrens

Abbildung 7.4.: Darstellung der Suchraumeinschränkung und Segmentierung durch ein Schwellwertverfahren im HSV-Farbraum

Nach Berechnung aller intrinsischen Parameter inklusive der Verzerrungskoeffizienten kann ein verzerrt aufgenommenes Bild entzerrt werden. Abbildung 7.3 zeigt einen Vergleich zwischen einem ursprünglich verzerrt aufgenommenen Bild und einem mittels beschriebener Kalibrierung entzerrtem Bild.

Nachdem alle Parameter zur Projektion eines Punktes im Raum auf Punkte im aufgenommenen und entzerrten Bild bekannt sind, kann die Detektion und Positionsapproximation des FTFs im Roboterbewegungsbereich durchgeführt werden. Diese wird im folgenden Abschnitt detailliert beschrieben.

7.2.2. Detektion der FTF-Bildposition

Nachdem ein Bild wie im vorangegangenen Abschnitt beschrieben aufgenommen und entzerrt wurde, erfolgt die Erkennung des fahrerlosen Transportfahrzeugs im Bild. Die dafür nötigen Schritte sind in den folgenden Unterabschnitten beschrieben.

Suchraumeinschränkung

Nur ein Teilbereich des aufgenommenen Kamerabilds ist für die weitere Verarbeitung interessant. Um zu verhindern, dass beispielsweise außerhalb der Roboterzelle fahrende Fahrzeuge die Fahrzeugdetektion stören, wird zunächst eine Suchraumeinschränkung vorgenommen. Dafür wird eine Binärmaske, die *Region of Interest*, genutzt um den Bereich des Bilds, in dem das Fahrzeug nicht detektiert werden soll, auszublenden.

Abbildung 7.4 stellt diesen Vorgang bildlich dar. In Abbildung 7.4a ist zunächst das entzerrte Bild zu sehen, auf dem noch alle uninteressanten Bildbereiche vorhanden sind. Die Maske ist in Abbildung 7.4b abgebildet und das Ergebnis nach Anwendung der Suchraumeinschränkung ist in Abbildung 7.4c dargestellt.

Segmentierung im HSV-Raum

Die Detektion des Fahrzeugs wird dadurch erleichtert, dass es sich durch seine Farbgebung stark von der Umgebung abhebt. Daher findet zunächst eine Segmentierung auf der Basis der Pixelfarbe statt. Um einfach feststellen zu können, welche Pixel des Bildes einen gelben Farbton besitzen, erfolgt eine Transformation aus dem BGR²- in den HSV-Farbraum. Durch einen festen Schwellwert für die Farbsättigung S und einen festen zulässigen Bereich für den Farbwert H können alle Pixel des Bildes bestimmt werden, die eine gelbe Farbe besitzen.

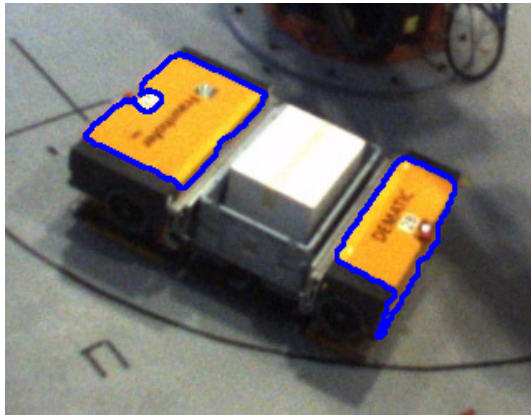
Da die Gelbtöne des Fahrzeugs sich deutlich von der Umgebung unterscheiden, genügt es, den Schwellwert bzw. den zulässigen Farbwertbereich manuell unter Berücksichtigung des visuellen Feedbacks manuell festzulegen. Die Pixel, die durch diese Segmentierung als zu den Fahrzeugflächen gehörend eingestuft wurden, sind in Abbildung 7.4d noch sichtbar, alle anderen Pixel wurden schwarz gefärbt. Das Ergebnis der Segmentierung ist eine binäre Maske, die für jeden Pixel angibt, ob er gemäß den festgelegten Schwellwerten Teil einer Fahrzeugfläche ist oder nicht.

Konturenextraktion und Bestimmung der Eckpunkte

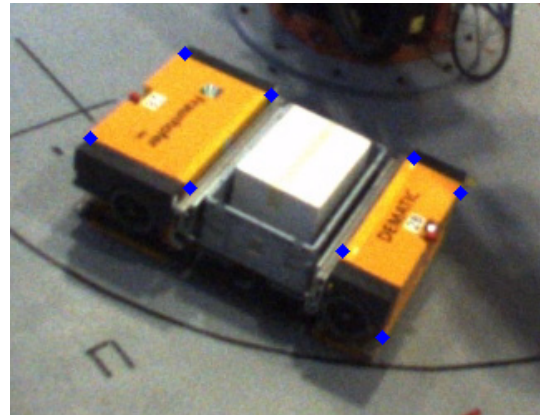
In diesem Schritt soll zunächst die Kontur der beiden Fahrzeugflächen als Folge von Punkten bestimmt werden, um darauf aufbauend die Eckpunkte der Flächen zu ermitteln. Als Vorverarbeitungsschritt wird ein *Closing* [12] als morphologische Operation mit einem ellipsenförmigen Strukturelement durchgeführt, um kleine Löcher in der Maske, die durch die Segmentierung erzeugt wurde, auszufüllen.

Die Extraktion der Konturen erfolgt nach einem Border-Following-Ansatz [42]. Dieser Algorithmus ermöglicht die Erstellung einer Hierarchie der Konturen von in einem Binärbild enthaltenen Flächen sowie deren Löcher. Damit werden mehr Informationen berechnet als für diesen Anwendungsfall eigentlich benötigt werden, denn für das weitere Vorgehen genügt die Extraktion der äußeren Konturen der im Bild vorhandenen Flächen. Löcher oder weitere in den Löchern enthaltene Flächen sind für die Projektgruppe nicht interessant und treten in der Regel auch nicht auf. Die für Bildverarbeitungsoperationen eingesetzte Programmbibliothek

²Das BGR-Format, in dem das Bild zunächst vorliegt, entspricht dem RGB-Format in umgekehrter Reihenfolge. Es werden also getrennt die Anteile von Rot, Grün und Blau abgespeichert.



(a) Extrahierte Konturpolygone (blau)



(b) Ecken nach Anwendung des Douglas-Peucker-Algorithmus

Abbildung 7.5.: Ergebnis nach der Konturenextraktion sowie verbleibende Ecken nach der Bildung der konvexen Hülle und Anwendung des Douglas-Peucker-Algorithmus zur Vereinfachung der Polygone. Grundlage ist die in Abbildung 7.4d dargestellte Segmentierung, der relevante Bildausschnitt wurde für die Abbildung vergrößert.

*OpenCV*³ enthält jedoch eine Implementierung dieses Algorithmus, so dass er trotz der nicht benötigten zusätzlichen Funktionalität genutzt wurde. In Abbildung 7.5a sind die aus der in Abbildung 7.4d gezeigten Segmentierung extrahierten Konturen eingezeichnet. Zur besseren Sichtbarkeit ist der Bildausschnitt vergrößert dargestellt.

Für die Projektgruppe hätte auch nur der Kern dieses Algorithmus ausgereicht, klassisches Border-Following [43, Algorithm 2]. Ein Pixel der Maske wird dabei als 1-Pixel angesehen, wenn er durch die Segmentierung als Bestandteil der Fahrzeugfläche eingestuft wurde, und als 0-Pixel, wenn er nicht als Fahrzeugfläche eingestuft wurde. Das Bild wird von links nach rechts und von oben nach unten durchsucht, bis ein 1-Pixel als Startpixel gefunden wurde. Die Randverfolgung durchläuft nun von einem benachbarten 0-Pixel beginnend gegen den Uhrzeigersinn alle benachbarten Pixel der Startposition, bis ein 1-Pixel und damit eine Fortsetzung des Rands gefunden wird. Die Randverfolgung wird bei diesem 1-Pixel fortgesetzt und dessen benachbarte Pixel gegen den Uhrzeigersinn durchlaufen, wobei die Suche bei dem Pixel startet, von dem aus zuvor gesucht wurde.

Auf diese Weise wird für jede im Bild vorhandene zusammenhängende Komponente eine Kontur als Folge von direkt benachbarten Pixelkoordinaten und somit als Polygon mit sehr vielen Punkten bestimmt. Da das Fahrzeug aus zwei gelben Flächen besteht, entstehen im Idealfall zwei Polygone. In einigen Fällen können abhängig von der Roboterposition und Beleuchtungssituation kleine Teile des Roboters ebenfalls durch die Segmentierung als Teil der Fahrzeugfläche eingestuft werden. Aus diesem Grund wird für jedes der extrahierten Polygone der Flächeninhalt bestimmt und zu kleine Polygone verworfen. Dafür wird eine leicht veränderte Form der Gaußschen Trapezformel [44, Formel 2] (siehe Formel 7.14) verwendet.

³<http://opencv.org/>

In der Formel steht n für die Anzahl der Punkte $(x_i, y_i)^T$ des Polygons. Es existiert zusätzlich $(x_{n+1}, y_{n+1})^T = (x_0, y_0)^T$. Der Wert des Flächeninhalts A ergibt sich durch

$$A = \frac{1}{2} \cdot \sum_{i=1}^n (y_i + y_{i+1}) \cdot (x_i - x_{i+1}). \quad (7.14)$$

Der Vorgang wird nur fortgesetzt wenn exakt zwei Polygone, die den vorgegebenen Mindest-Flächeninhalt besitzen, ermittelt wurden. Falls mehr als zwei den Kriterien entsprechende Polygone im Bild vorhanden sind, so ist davon auszugehen, dass sich zwei Fahrzeuge im Bild befinden – dieses Szenario wird durch die Detektion nicht unterstützt, könnte jedoch einfach ergänzt werden. Ist nur ein Polygon sichtbar, so ist die Sicht auf das Fahrzeug zu sehr eingeschränkt, um eine zuverlässige Position zu liefern.

Für die Bestimmung der Fahrzeugposition im Bild sollen im nächsten Schritt die Ecken der extrahierten Konturen ermittelt werden. Zwar liegen die Konturen bereits als Polygon vor, jedoch ist hier jeder Pixel der Kontur als ein Punkt des Polygons repräsentiert, so dass die Polygone sehr viele Ecken besitzen. Eigentlich interessant ist, welche Positionen die auf dem Kamerabild sichtbaren 90° -Winkel der Fahrzeugflächen besitzen. Daher wird zunächst die konvexe Hülle jedes Polygons gebildet, wodurch bereits einige Ecken entfernt werden.

Eine Reduzierung der Eckenzahl auf die eigentlich relevanten Ecken, die auch mit denen der Plastik-Fahrzeugabdeckungen im Bild übereinstimmen, erfolgt durch den Douglas-Peucker-Algorithmus [45]. Dafür werden zunächst zwei Eckpunkte des Polygons mit großem Abstand zueinander und das durch diese beiden Ecken beschriebene vereinfachte Polygon (eine Linie) betrachtet. Es wird nun die Ecke des ursprünglichen Polygons ermittelt, die den größten Abstand zum vereinfachten Polygon besitzt. Ist dieser Abstand größer als eine festgelegte Schwelle ϵ , so wird die Ecke in das vereinfachte Polygon eingefügt und das Verfahren erneut angewendet, ansonsten terminiert der Algorithmus und gibt das vereinfachte Polygon aus. Der Algorithmus lässt sich nach dem Teile-und-Herrsche-Prinzip effizient rekursiv implementieren. Das vereinfachte Polygon enthält wie nun nur noch Ecken an Bildpositionen, an denen sich auch eine tatsächliche Ecke der durch die Segmentierung bestimmten Maske der Fahrzeugflächen befindet. Die Ecken des so ermittelten Polygons sind in Abbildung 7.5b farblich markiert.

Positionsapproximation

Nachdem für jede der beiden Fahrzeugflächen die Eckpunkte bestimmt wurden, erfolgt die Approximation der Position des Fahrzeugs im Bild. Dafür werden aus jedem der beiden Polygone die zwei Eckpunkte ermittelt, deren Entfernung zur anderen Fläche am geringsten ist. In diesem einfachen Fall erfolgt die Sortierung dabei über die Summe der Entfernungen eines Eckpunkts zu allen Eckpunkten des jeweils anderen Polygons. In Abbildung 7.6 sind die vier so ermittelten Ecken rot hervorgehoben. Der Zentroid dieser vier innenliegenden Eckpunkte wird als Approximation der Fahrzeug-Bildposition genutzt und ist in Abbildung 7.6 als schwarzer Punkt markiert. Es ist bei der weiteren Verarbeitung zu berücksichtigen, dass der Punkt im Raum betrachtet nicht auf der Bodenebene liegt, sondern in 35 Zentimetern Höhe auf der Ebene, auf der sich auch die gelben Fahrzeugflächen befinden.

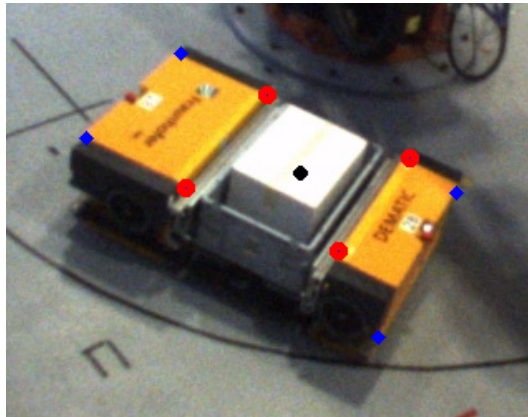


Abbildung 7.6.: Innenliegende Ecken und approximierte Fahrzeugposition

7.2.3. Berechnung der FTF-Position im Raum

Im vorhergehenden Abschnitt wurde erklärt, wie die FTF-Position im Bild bestimmt wird. Die Position liegt zu diesem Zeitpunkt in Bildkoordinaten vor. Da nicht bekannt ist, an welcher Position im Raum sich das FTF befindet, muss die Position ins Weltkoordinatensystem transformiert werden. In Kapitel 3.1.2 wurde bereits erläutert, dass sich zu einer homogenen Weltkoordinate \mathbf{x} die entsprechende Bildkoordinate

$$\mathbf{x}' = \mathbf{K}\mathbf{D}\mathbf{x} = \mathbf{P}\mathbf{x} \quad (7.15)$$

mithilfe der intrinsischen Kameramatrix \mathbf{K} und der extrinsischen Kameramatrix \mathbf{D} berechnen lässt. In Abschnitt 7.2.1 wurde beschrieben, wie sich die intrinsischen Kameraparameter bestimmen lassen. Im nachfolgenden Abschnitt wird geschildert, wie sich die extrinsischen Kameraparameter bestimmen lassen. Anschließend wird dargelegt, wie sich damit die FTF-Position in Weltkoordinaten berechnen lässt.

Bestimmung der extrinsischen Kameraparameter

Die extrinsische Matrix $\mathbf{D} = (\mathbf{R} \ \mathbf{t})$ setzt sich aus einer 3×3 -Rotationsmatrix \mathbf{R} und einem 3×1 -Translationsvektor \mathbf{t} zusammen.

Der Translationsvektor besteht aus den Koordinaten t_x, t_y und t_z , die die Verschiebung des Kamerakoordinatensystems zum Ursprung des Weltkoordinatensystems beschreiben:

$$\mathbf{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}. \quad (7.16)$$

Mithilfe der Rodrigues-Formel [46] ergibt sich die Rotationsmatrix

$$\mathbf{R} = \begin{pmatrix} \cos\theta + x^2(1 - \cos\theta) & xy(1 - \cos\theta) - z\sin\theta & xz(1 - \cos\theta) + y\sin\theta \\ xy(1 - \cos\theta) + z\sin\theta & \cos\theta + y^2(1 - \cos\theta) & yz(1 - \cos\theta) - x\sin\theta \\ xz(1 - \cos\theta) - y\sin\theta & yz(1 - \cos\theta) + x\sin\theta & \cos\theta + z^2(1 - \cos\theta) \end{pmatrix}, \quad (7.17)$$

die eine Rotation um die Rotationsachse $\mathbf{a}_R = (x \ y \ z)^T$ um den Winkel $\theta = |\mathbf{a}_R|$ beschreibt. Die Translation und Rotation lassen sich in einem Parametrisierungsvektor

$$\mathbf{a} = \begin{pmatrix} \mathbf{t} \\ \mathbf{a}_R \end{pmatrix} \quad (7.18)$$

zusammenfassen. Die extrinsischen Kameraparameter lassen sich dann in Abhängigkeit von \mathbf{a} als

$$\mathbf{D}(\mathbf{a}) = \begin{pmatrix} \cos\theta + x^2(1 - \cos\theta) & xy(1 - \cos\theta) - z\sin\theta & xz(1 - \cos\theta) + y\sin\theta & t_x \\ xy(1 - \cos\theta) + z\sin\theta & \cos\theta + y^2(1 - \cos\theta) & yz(1 - \cos\theta) - x\sin\theta & t_y \\ xz(1 - \cos\theta) - y\sin\theta & yz(1 - \cos\theta) + x\sin\theta & \cos\theta + z^2(1 - \cos\theta) & t_z \end{pmatrix} \quad (7.19)$$

darstellen. Um diese Parameter bestimmen zu können, werden Korrespondenzen zwischen Welt- und Bildkoordinaten benötigt. Eine Korrespondenz zwischen einer Weltkoordinate \mathbf{x} und einer Bildkoordinate \mathbf{x}' wird als Tupel $k : (\mathbf{x}, \mathbf{x}')$ dargestellt.

Zur Bestimmung der Korrespondenzen werden mehrere Punkte im Roboterkäfig markiert (siehe Abbildung 7.7) und manuell mit dem Roboterarm angefahren, um damit die Positionen im Weltkoordinatensystem zu bestimmen. Anschließend werden die entsprechenden Punkte im Bild bestimmt, sodass eine Menge von Punktkorrespondenzen

$$\mathcal{M}_P = \{k_1, k_2, \dots, k_n\}$$

mit den Elementen $k_j : (\mathbf{x}_j, \mathbf{x}'_j), j = 1, \dots, n$ erhalten wird. Gesucht ist der Parametrisierungsvektor \mathbf{a} , für den im optimalen Fall für alle j gilt:

$$\mathbf{x}'_j = \mathbf{K}\mathbf{D}(\mathbf{a})\mathbf{x}_j. \quad (7.20)$$

Um \mathbf{a} eindeutig bestimmen zu können, werden mindestens sechs Korrespondenzpaare benötigt [47]. Aufgrund von möglichen Messungenauigkeiten empfiehlt es sich jedoch, mehr Korrespondenzen zu verwenden. Es entsteht also ein nicht-lineares Gleichungssystem. Nicht-lineare Gleichungssysteme lassen sich nicht direkt lösen und werden häufig mittels numerischer Verfahren angenähert. Ziel der Verfahren ist es, den quadratischen Fehler e zu minimieren, d.h.

$$e = \min \sum_{j=1}^n (\mathbf{x}'_j - \mathbf{K}\mathbf{D}(\mathbf{a})\mathbf{x}_j)^2. \quad (7.21)$$

Das Problem, einen Satz von Parametern für eine Funktion zu bestimmen, sodass die Summe der quadratischen Fehler minimal wird, wird auch als *Ausgleichsproblem* bezeichnet. Ein Verfahren zur Lösung des Ausgleichsproblems ist der *Levenberg-Marquaedt-Algorithmus* [48].

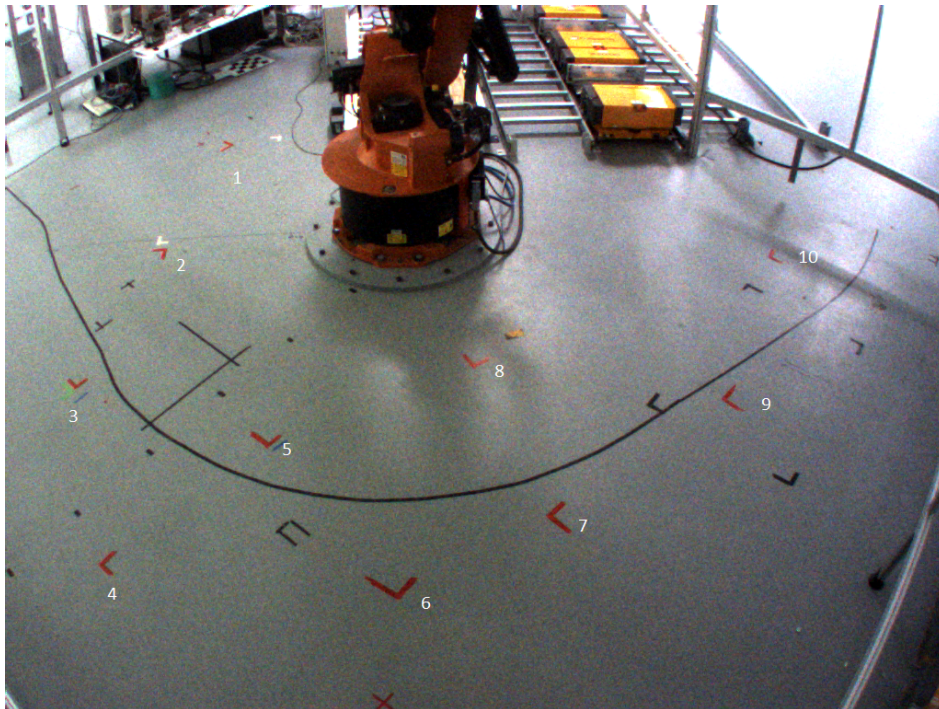


Abbildung 7.7.: Markierung der Punkte zur Bestimmung der Korrespondenzen zwischen Welt- und Bildkoordinaten

Der Levenberg-Marquardt-Algorithmus ist ein iterativer Lösungsalgorithmus und eine Mischung zwischen dem Gauß-Newton-Verfahren [49] und dem Gradientenabstiegsverfahren. Dabei wird sich in jedem Iterationsschritt dem Minimum angenähert, bis sich der Minimumwert nicht mehr signifikant ändert. Aus den durch das Levenberg-Marquardt-Algorithmus geschätzten Parametern \mathbf{a} lässt sich am Ende die extrinsische Matrix \mathbf{D} aufstellen.

Transformation der FTF-Bildposition

Im vorherigen Abschnitt wurde erläutert, wie die extrinsischen Parameter einer Kamera näherungsweise bestimmt werden können. Da nun sowohl intrinsische als auch extrinsische Parameter bekannt sind, ist es jetzt einfach möglich, zu einem beliebigen Punkt im Weltkoordinatensystem den entsprechenden Punkt im Bildkoordinatensystem zu bestimmen. Da im Rahmen der Projektgruppe jedoch die FTF-Position in Weltkoordinaten von Interesse ist, diese aber zunächst nur in Bildkoordinaten vorliegt (siehe Abschnitt 7.2.2), muss die Transformation invertiert werden.

Gegeben ist ein Punkt \mathbf{x}' in homogenen Bildkoordinaten. Durch

$$\mathbf{x} = \mathbf{D}^{-1}\mathbf{K}^{-1}\mathbf{x}' \quad (7.22)$$

ergibt sich ein möglicher Punkt \mathbf{x} im Weltkoordinatensystem, der auf \mathbf{x}' im Bildkoordinatensystem abgebildet werden würde. Da \mathbf{x} in homogenen Koordinaten vorliegt, muss jede Komponente des Vektors zunächst durch die letzte Vektorkomponente geteilt werden, um die normalen Weltkoordinaten zu erhalten:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \in \mathbb{P}^3 \implies \mathbf{x}^* = \begin{pmatrix} x_1/x_4 \\ x_2/x_4 \\ x_3/x_4 \end{pmatrix} = \begin{pmatrix} x_1^* \\ x_2^* \\ x_3^* \end{pmatrix} \in \mathbb{R}^3. \quad (7.23)$$

Da bei der Projektion aus dem Weltkoordinatensystem ins Bildkoordinatensystem jedoch die Tiefeninformation verloren geht, kann auch genauso gut jeder andere Punkt, der auf der Geraden, die durch \mathbf{x} und den Ursprung des Kamerakoordinatensystems geht, auf \mathbf{x}' abgebildet worden sein. Da angenommen werden kann, dass der bestimmte Punkt auf der Höhe h des FTFs liegt, lässt sich damit eine eindeutige Weltkoordinate bestimmen. Die Position der Kamera \mathbf{c} lässt sich aus der Rotationsmatrix \mathbf{R} und dem Translationsvektor \mathbf{t} bestimmen:

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \mathbf{R}^{-1}\mathbf{t}. \quad (7.24)$$

Daraus lässt sich die Geradengleichung

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ h \end{pmatrix} = \begin{pmatrix} x_1^* \\ x_2^* \\ x_3^* \end{pmatrix} - \lambda \begin{pmatrix} x_1^* - c_1 \\ x_2^* - c_2 \\ x_3^* - c_3 \end{pmatrix} \quad (7.25)$$

aufstellen und das passende λ bestimmen durch:

$$\lambda = \frac{x_3^* - h}{x_3^* - c_3}. \quad (7.26)$$

Durch Einsetzen des berechneten λ in die Geradengleichung wird die Position des FTF im Raum bestimmt.

7.3. Paketerkennung

Die Paketerkennung befasst sich mit der Auswertung der Tiefenkameradaten, um durch diese die genaue Paketposition in Weltkoordinaten, d.h. in Koordinaten relativ zum Robotorursprung, zu ermitteln. Der grüne Zweig in Abbildung 7.1 verdeutlicht das schematische Vorgehen. Aus den Daten, welche die Tiefenkamera liefert (siehe Kapitel 7.3.1), werden mit Hilfe einer Filterpipeline (siehe Kapitel 7.3.2) nicht relevante Datenpunkte verworfen, um zuletzt eine Identifizierung der Paketfläche zu ermöglichen. Zum Schluss wird die, durch eine Hand-Auge-Kalibrierung auf TCP-Koordinaten zurückgeführte Paketposition, durch eine Koordinatentransformation in Weltkoordinaten überführt (siehe Kapitel 7.3.3).

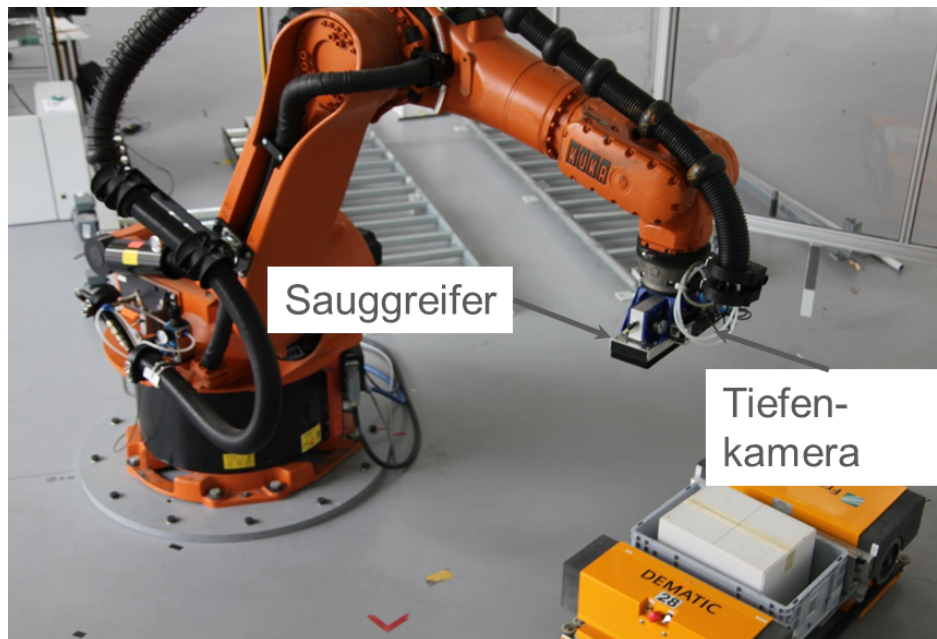


Abbildung 7.8.: Die Tiefenkamera ist parallel zum Sauggreifer direkt am Ende des Greifarms justiert.

7.3.1. Bildakquisition

Zur Ermittlung der Tiefeninformationen befindet sich am Sauggreifer des Roboters eine Tiefenkamera. Abbildung 7.8 zeigt die am Ende des Greifarms parallel zum Sauggreifer angebrachte Tiefenkamera. Durch diese Positionierung kann das Tiefenkameramodul, sobald sich der Sauggreifer und damit die Tiefenkamera einmal über dem Fahrzeug befindet, die aktuelle Paketposition ermitteln und diese an das Tracking weiterleiten (siehe Kapitel 7.4). So wird die aktuelle Paketposition in die Bahnplanung mit einbezogen, um zu gewährleisten, dass der Sauggreifer das Paket verfolgt und letztendlich greift.

Allerdings folgen aus der Positionierung direkt am Sauggreifer auch Probleme. Zum einen ist das Sichtfeld der Kamera sehr eingeschränkt. Sowohl durch den Sauggreifer selbst, als auch durch die Ausrichtung des Sauggreifers eingeschränkt, ist der Ausschnitt des für die Tiefenkamera sichtbaren Bereichs klein im Vergleich zur Fahrzeugbahn, wodurch eine initiale Erkennung des Paketes erforderlich wird. Um den Greifarm zu Beginn über das Fahrzeug zu lenken, wird daher nur die CCD-Kamera (siehe Kapitel 7.2) verwendet. Erst wenn der Greifarm über dem Fahrzeug und damit das Paket im Blickfeld der Tiefenkamera ist, werden auch die Daten der Tiefenkamera für das Tracking verwendet. Weiterhin liefert die Tiefenkamera Tiefendaten relativ zum Kameraursprung. Der Roboter liefert jedoch nur Positionsdaten des TCP, welche später für die Umrechnung in Weltkoordinaten verwendet werden. Es ist also eine Transformation der Kameradaten in TCP-Koordinaten notwendig. Dies wird in Abschnitt 7.3.1 genauer erklärt. Zuvor wird jedoch im nächsten Abschnitt die verwendete Hardware genauer beschrieben.

Verwendete Hardware

Zum Ermitteln der Tiefendaten und damit zum Ermitteln des Paketmittelpunktes wird, wie bereits in Kapitel 4.1 erwähnt, eine Tiefenkamera verwendet, die mit strukturiertem Licht arbeitet. Gründe für diese Entscheidung liegen sowohl in der Messgenauigkeit, als auch Portabilität und Kompaktheit der Hardware, sowie dem niedrigen Preis.

Es wird eine *ASUS⁴ Xtion Pro mit PrimeSense 3D-Sensor⁵* verwendet. Diese ist mit einem Messbereich von 80cm bis 350cm beziffert, die Evaluation (siehe Kapitel 8.3) hat jedoch ergeben, dass auch bei kleinerer Distanz zur Kamera noch Tiefeninformationen aufgenommen werden konnten. Die Auflösung des Tiefenbildes beträgt maximal 640×480 Pixel bei 30fps . Bei niedrigerer Auflösung können bis zu 60fps erzielt werden. Weiterhin ist eine CCD-Kamera enthalten, welche zusätzlich zu den Tiefen- auch Farbinformationen liefert. Diese werden in der Filterpipeline (siehe Abschnitt 7.3.2) zum Ausschneiden von Fahrzeugpunkten verwendet. Im Programm selbst kann die Kamera über das OpenNI-Framework⁶ angesprochen werden. Das Tiefenkameramodul verwendet dabei die Point Cloud Library⁷ (PCL), ein Framework zur Verwaltung und Verarbeitung von Punktwolken, das zur Akquisition auf das OpenNI-Framework zurückgreift.

Kamera-Kalibrierung

Mit der Aufnahme und Auswertung der Tiefenbilder werden Tiefeninformationen relativ zur Tiefenkamera erlangt. Letztendlich werden Koordinaten im Roboterkoordinatensystem benötigt. Die Position des Greifarmes wird über die TCP-Koordinaten bereit gestellt. Auf Abbildung 7.9 ist sowohl das Roboterkoordinatensystem (gelb) als auch das TCP-Koordinatensystem (grün) zu sehen. Um, wie in Kapitel 7.3.3 beschrieben, eine Koordinatentransformation von TCP-Koordinaten in Roboter-Koordinaten durchführen zu können, müssen also die von der Tiefenkamera berechneten Punkte in das TCP-Koordinatensystem überführt werden. Dies geschieht ähnlich zu der Transformation in Kapitel 7.3.3 durch eine affine Transformation der Form

$$f(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} + \mathbf{t}. \quad (7.27)$$

Dabei ist \mathbf{A} eine 3×3 affine Transformationsmatrix und $\mathbf{t} \in \mathbb{R}^3$ ein Translationsvektor, die zusammen einem Punkt $\mathbf{x} \in \mathbb{R}^3$ in Kamerakoordinaten einen Punkt $f(\mathbf{x}) \in \mathbb{R}^3$ in TCP-Koordinaten zuordnen. Wird davon ausgegangen, dass die Tiefenkamera intrinsisch kalibriert ist, so stellt die Matrix \mathbf{A} eine Rotation des Kamerakoordinatensystems im TCP-Koordinatensystem dar.

Um diesen Versatz der Tiefenkamera im TCP-Koordinatensystem automatisiert berechnen zu können, kann eine Hand-Auge-Kalibrierung durchgeführt werden [50]. Dabei werden Kalibrierungsbilder aufgenommen, in denen Punkte sowohl in Kamera, als auch in TCP-Koordinaten bekannt sind. Dafür eignet sich z.B. ein Schachbrett bekannter Gitterweite. Um die Koor-

⁴<http://www.asus.com/>, Stand 15.3.2014

⁵<http://www.primesense.com/>, Stand 15.3.2014

⁶<http://www.openni.org/>, Stand 15.3.2014

⁷<http://pointclouds.org/>, Stand 15.3.2014

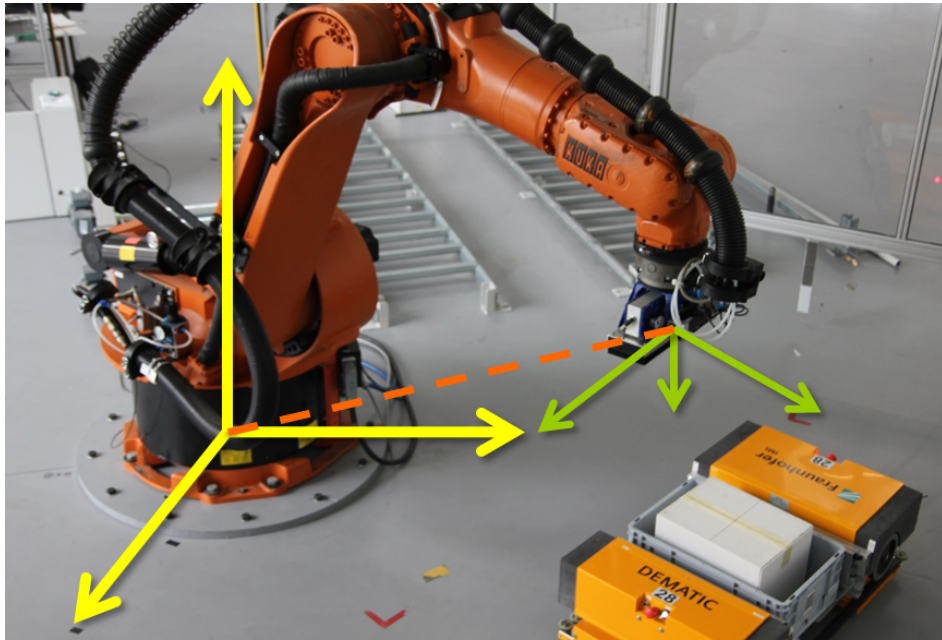


Abbildung 7.9.: Das Koordinatensystem des Roboters (gelb) liegt im Ursprung des Roboters. Die TCP-Koordinaten des Greifarms werden vom Roboter in diesem Koordinatensystem angegeben. Relevant für die Kalibrierung ist das TCP-Koordinatensystem (grün), d.h. die Position von Objekten relativ zum TCP.

diaten in TCP-Koordinaten zu ermitteln, muss das Schachbrett genau vermessen werden. Nachdem die TCP-Koordinaten bekannt sind, können über extrinsische Kalibrierungsverfahren wie z.B. das Verfahren von Zhang [51], der Ort des Schachbretts in Kamerakoordinaten ermittelt werden.

Sei I die Indexmenge der aufgenommenen Testbilder. Bekannt sind dann für jedes Bild $i \in I$ die Koordinaten des Schachbrettes $\mathbf{s}_i^{(K)} \in \mathbb{R}^3$ sowohl in Kamerakoordinaten $\mathbf{s}_i^{(K)} \in \mathbb{R}^3$, als auch in TCP-Koordinaten $\mathbf{s}_i^{(TCP)} \in \mathbb{R}^3$. Gesucht ist die Transformationsmatrix $\mathbf{A} \in \text{Mat}_{3 \times 3}$ und der Translationsvektor $\mathbf{t} \in \mathbb{R}^3$, so dass

$$\mathbf{s}_i^{(TCP)} = \mathbf{A} \cdot \mathbf{s}_i^{(K)} + \mathbf{t}. \quad (7.28)$$

Da jedoch die Koordinateninformationen z.B. durch Beschränkungen der Auflösung der Kamera nicht exakt bestimmt werden können, ist es nicht trivial, diese Transformation zu bestimmen. Bekannte Verfahren bauen dafür in der Regel lineare Gleichungssysteme für jedes Kalibrierungsbild auf und ermitteln die Transformationsmatrix, die über alle Bilder einen Fehler minimiert. Detaillierte Informationen über die Vorgehensweise können dem Paper von Tsai [50] entnommen werden.

7.3.2. Detektion der Paketposition

Die Tiefenkamera, am Ende des Roboterarms nahe des Sauggreifers angebracht, dient im finalen Projektssystem der Detektion der Paketposition im Weltkoordinatensystem der Roboterumgebung. Hat sich der Roboterarm über das Fahrzeug bewegt, hat die Tiefenkamera den Laderaum des Fahrzeugs im Blick. Um die Position des Paketes im Weltkoordinatensystem zu bestimmen, detektiert das Tiefenkameramodul zunächst die Paketposition im Koordinatensystem der Tiefenkamera. Die Transformation der Koordinaten in das entsprechende Weltkoordinatensystem ist in Kapitel 7.3.3 beschrieben. Zusätzlich wird die Orientierung des Paketes, das heißt die Ausrichtung des Paketes im Fahrzeug bestimmt, um die Ausrichtung des Sauggreifers beim Greifen des Paketes an die des Paketes anzupassen.

Im Folgenden werden die Verfahren beschrieben, die das Tiefenkameramodul zur Detektion der Paketposition und -orientierung verwendet. Der Aufbau dieses Unterkapitels orientiert sich dabei an der Pipeline des Moduls: Gegeben eine Punktwolke der Szene (in der sich das Fahrzeug befindet), werden zunächst Filter auf die Punktwolke zur Reduktion der Daten angewandt. Mit dem RANSAC-Verfahren wird anschließend die Oberseite des Paketes erkannt. Mit Hilfe der konvexen Hülle der Paketoberseite wird schließlich die Position und die Orientierung des Paketes bestimmt.

Filterpipeline

Im ersten Schritt der Detektion werden unterschiedliche Filterverfahren auf die gegebene Punktwolke der Umgebung des Paketes, die aus der Tiefenkamera extrahiert wird, angewandt. Die Filterverfahren dienen alle der Reduktion der Punktwolkendaten in einer Form, so dass die weitergehende Detektion effizienter durchgeführt werden kann. Die Filterverfahren werden in der Reihenfolge des Auftretens in diesem Unterabschnitt verwendet.

Z-Filter

Sei $\mathcal{P} \subset \mathbb{R}^3$ die aus der Tiefenkamera extrahierte, drei-dimensionale Punktwolke. Auf diese wird zunächst ein so genannter *Z-Filter* angewandt. Wie bereits in Kapitel 4.3.2 beschrieben, kann es sinnvoll sein alle Punkte zu verwerfen, die eine bestimmte Höhengrenze unterschreiten. So kann beispielsweise die minimale Höhenposition eines zu detektierenden Objektes in der Szene bekannt sein. Dadurch werden die Punkte redundant, die unterhalb dieser Position (in der z -Koordinate) liegen, und können somit verworfen werden.

Im Rahmen der Projektgruppe wird eine minimale Höhenposition des Paketes im Koordinatensystem berechnet. Punkte aus \mathcal{P} , die unter dieser Höhenposition liegen, können also nicht Teil des Paketes sein und somit aus der Punktwolke entfernt werden.

Sei Ω die minimale Höhenposition des Paketes gegeben. Dann kann Ω als Schwellwert für die Höhe der Punkte, also der z -Koordinate, verwendet werden:

$$\mathcal{P}_z = \{(p_x, p_y, p_z)^T \in \mathcal{P} \mid p_z < \Omega\}. \quad (7.29)$$

Die entstehende Punktwolke \mathcal{P}_z enthält dann alle Punkte, die zum Paket gehören können. Punkte, die zum Boden oder zum unteren Teil des Fahrzeuges gehören, sind dagegen nicht mehr in \mathcal{P}_z enthalten [21].

Voxelgrid-Filter

Nach dem Z-Filter wird ein *Voxelgrid*-Filter auf die Punktwolke angewandt. Punktwolken enthalten im Allgemeinen sehr viele, nah aneinander liegende Punkte, die zwar einen größeren Detailgrad der Szene ermöglichen, allerdings für die Erkennung von Objekten in der Szene nicht alle benötigt werden. Für die Effizienz weitergehender Anwendungen ist es vielmehr von Vorteil den Detailgrad zu verringern, ohne dass die grundlegenden Merkmale der Szene verzerrt werden. Ziel des Voxelgrid-Filters ist eine solche Reduktion der Punktwolke.

Sei mit $\mathcal{P}_z \subset \mathbb{R}^3$ diejenige Punktwolke gegeben, die bereits durch den Z-Filter reduziert worden ist. Die Idee des Voxelgrid-Filters basiert, wie bereits in Kapitel 4.3.2 erläutert, auf der Überdeckung der Punktwolke mit n so genannten *Voxeln* $(\mathcal{V}_j)_{1 \leq j \leq n}$. Ein Voxel $\mathcal{V}_j = ([v_{j,x}, v_{j,x} + l] \times [v_{j,y}, v_{j,y} + l] \times [v_{j,z}, v_{j,z} + l]) \cap \mathcal{P}_z$ ist ein kubisches Objekt im Raum der Punktwolke mit der festen Größe l , das durch einen Punkt $(v_{j,x}, v_{j,y}, v_{j,z}) \in \mathbb{R}^3$ definiert ist. Voxel besitzen alle die gleiche, feste Größe l , überlappen sich nur an den Rändern und decken den gesamten Bereich der Punktwolke im Raum ab. Somit ist jeder Punkt der Punktwolke in mindestens einem Voxel enthalten, das heißt für jedes $\mathbf{p} \in \mathcal{P}_z$ existiert ein j ($1 \leq j \leq n$), so dass $\mathbf{p} \in \mathcal{V}_j$ gilt.

Zu jedem Voxel nicht leeren $\mathcal{V}_j \neq \emptyset$ wird nun der so genannte *Zentroid* \mathbf{z}_j , also der Mittelpunkt der im Voxel enthaltene Punktmenge, berechnet:

$$\mathbf{z}_j = \frac{\sum_{\mathbf{p} \in \mathcal{V}_j} \mathbf{p}}{|\mathcal{V}_j|}. \quad (7.30)$$

Die Zentroiden dienen als Repräsentanten der Voxel und der in ihnen enthaltenen Punkte. Um die Punktwolke zu reduzieren, werden nur noch Zentroiden in die Punktwolke aufgenommen. Damit ergibt sich die Punktwolke wie folgt:

$$\mathcal{P}_v = \bigcup_{j=1}^n \{\mathbf{z}_j\}. \quad (7.31)$$

Damit wird für jeden Voxel genau ein Punkt in die resultierende Punktwolke \mathcal{P}_v aufgenommen. Die Größe von \mathcal{P}_v hängt also von der Anzahl der zur Überdeckung der Punktwolke benötigten Voxel und ebenfalls von der festen Größe l der Voxel ab. Über die Wahl von l kann demnach der Detailgrad der resultierenden Punktwolke bestimmt werden [21].

Farbfilter

Zum Abschluss der Filterpipeline wird ein *Farbfilter* auf die Punktwolke angewandt. In diesem Verfahren werden die zusätzlichen RGB-Farbinformationen, die es zu jedem Punkt in der Punktwolke gibt (siehe Kapitel 4.2.4), genutzt.



Abbildung 7.10.: Die FTF des Fraunhofer Instituts besitzen einen sehr markanten Orangeton, deren zugehörige Punkt看ke durch einen Farbfiler aus der Punkt看ke extrahiert werden können.

Sind die Farbinformationen des zu erkennenden Objektes bekannt, so können andere Objekte, die sonst ebenfalls als Paket erkannt werden könnten, sich allerdings deutlich in der Farbe unterscheiden, aus der Punkt看ke entfernt werden. Im Rahmen der Projektgruppe haben die Fahrzeugober- und -unterseite eine dem Paket ähnliche Form. Diese unterscheiden sich allerdings durch ihren markanten, orangenen Farbton von den grauen Paketen. Somit können sie mit Hilfe eines Farbfilters, der Punkte aus der Punkt看ke löscht, die eben jenen orangenen (oder einen ähnlichen) Farbton (siehe Abbildung 7.10) besitzen, entfernt werden.

Sei \mathcal{P}_v die aktuelle Punkt看ke der Filterpipeline, das heißt auf \mathcal{P}_v wurde bereits sowohl der Z- als auch der Voxelgridfilter angewandt. Seien mit $\mathcal{F}_v \subseteq [0, 255]^3$ außerdem die RGB-Farbinformationen der Punkte aus \mathcal{P}_v gegeben, so dass für alle Koordinateninformationen $\mathbf{p}_i \in \mathcal{P}_v$ entsprechende Farbinformationen $\mathbf{f}_i \in \mathcal{F}_v$ existieren. Dann können für jeden Punkt \mathbf{p}_i der farbliche Unterschied zu einem festen Farbvektor $\mathbf{f}' \in [0, 255]^3$ durch den geometrischen Abstand im RGB-Farbraum bestimmt werden:

$$r_i = \|\mathbf{f}_i - \mathbf{f}'\|_2. \quad (7.32)$$

Sei Ω nun ein Schwellwert für den farblichen Abstand zu \mathbf{f}' im RGB-Farbraum. Der Farbfiler entfernt dann alle Punkte \mathbf{p}_i , deren farblicher Abstand r_i zu \mathbf{f}' den Schwellwert nicht überschreitet:

$$\mathcal{P}_f = \{\mathbf{p}_i \in \mathcal{P}_v \mid r_i > \Omega\}. \quad (7.33)$$

In \mathcal{P}_f sind also keine Punkte mehr enthalten, die einen *ähnlichen* Farbton wie \mathbf{f}' besitzen. Der Grad der Ähnlichkeit kann derweil mit der Angabe des Schwellwertes Ω festgelegt werden. Im Rahmen des Tiefenkameramoduls wird $\mathbf{f}' = (255, 165, 0)$ auf einen orangenen Farbton gesetzt, der dem orangenen Farbton der Fahrzeugteile ähnelt.

RANSAC

Um die Detektion der Paketposition zu erleichtern, wird im Tiefenkameramodul zunächst die Oberseite des Paketes erkannt. Dies ist für das Projekt ausreichend, da nur diese für das Greifen des Paketes von Relevanz ist. Die Oberseite des Paketes kann mit Hilfe einer Ausgleichsfläche eines lokalen Bereiches bestimmt werden.

In Kapitel 4.2.4 wurden bereits die Methode der kleinsten Quadrate (Ausgleichsfläche über alle bekannten Punkte) und die Bestimmung von MLS-Flächen vorgestellt. In letzterer Methode werden nur Daten berücksichtigt, die sich in einem lokalen Bereich eines festen, gegebenen Punktes befinden. Da im Rahmen der Projektgruppe allerdings im Allgemeinen keine ungefähre Position des Paketes bekannt ist, eignet sich diese Methode alleine nicht zur Bestimmung einer Ausgleichsfläche und somit zur Erkennung der Paketoberseite.

Im Tiefenkameramodul wird für die Detektion der Paketoberseite das so genannte *RANSAC-Verfahren* (*Random sample consensus*) genutzt. Bei diesem handelt es sich um ein randomisiertes Verfahren, das den *wahrscheinlichen* Ort des Paketes durch Erkennungsversuche schätzt. Dabei verfolgt das Verfahren der folgenden Grundidee: Sind nur Punkte der Paketoberseite gegeben und es wird die Ausgleichsfläche dieser Punkte bestimmt (zum Beispiel mittels der Methode der kleinsten Quadrate), so ist die Abweichung der Fläche von den Punkten sehr gering, da sich die Punkte der Paketoberseite bereits auf einer Ebene befinden. In der gegebenen Punktmenge existieren also nur wenige so genannte *Ausreißer*, die weit von der approximierten Fläche entfernt liegen und das Gesamtbild somit verzerren. Ausreißer sind also ein Indiz zur Erkennung der Paketoberseite: Existieren in der analysierten Punktmenge viele Ausreißer, so ist die Wahrscheinlichkeit, dass es sich um die Paketoberseite handelt, gering.

Um den wahrscheinlichen Ort der Paketoberseite zu erkennen, wird im RANSAC-Verfahren also eine Punktmenge gesucht, die zu möglichst wenig Ausreißern bei der Erstellung der Ausgleichsfläche führt. Um Ausreißer zu bestimmen, muss dabei zunächst ein Schwellwert für den Abstand zur Ausgleichsfläche festgelegt werden. Sei mit $\mathcal{P}_f \subset \mathbb{R}^3$ die bereits gefilterte Punktmenge und mit Ω der angesprochene Schwellwert gegeben. Das RANSAC-Verfahren ist iterativ und wählt zunächst eine minimale Menge von zufälligen Punkten, die zum Aufspannen des entsprechenden Ausgleichspolynoms benötigt werden. Im Falle einer Ausgleichsfläche sind dies drei Punkte. Seien $\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \mathbf{p}_{i,3} \in \mathcal{P}_f$ die Punkte, die in der i -ten Iteration zufällig gewählt werden. Über die Methode der kleinsten Quadrate kann dann das Ausgleichspolynom x_i^a dieser drei Punkte bestimmt werden.

Um nun zu bestimmen, ob es sich bei diesen Punkten um Ausreißer handelt, wird der Abstand aller anderen Punkte $\mathcal{P}_j \setminus \{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \mathbf{p}_{i,3}\}$ zu x_i^a bestimmt. Punkte, deren Abstand zu x_i^a den Schwellwert nicht unterschreiten, *unterstützen* das Ausgleichspolynom und werden der so genannten *Consensus-Menge* \mathcal{C}_i hinzugefügt:

$$\mathcal{C}_i = \{\mathbf{p}_j \in \mathcal{P}_f \mid \|x_i^a - \mathbf{p}_j\|_2 < \Omega\}. \quad (7.34)$$

Abbildung 7.11 zeigt eine mit dem RANSAC-Verfahren berechnete Ausgleichsgerade: Blaue Punkte unterstützen die Gerade und gehören damit zur Consensus-Menge, rote Punkte werden als Ausreißer klassifiziert. Dieses Verfahren wird mehrfach unabhängig voneinander durchgeführt. Daraus ergibt sich eine Folge $(\mathcal{C}_i)_{1 \leq i \leq n}$ von Consensus-Mengen, wobei n die Anzahl der Iterationen beschreibt. Je größer eine Consensus-Menge ist, desto mehr wird das zufällige Ausgleichspolynom unterstützt und desto weniger Ausreißer gibt es zu diesem Polynom. Deshalb wird die größte Consensus-Menge als Ergebnis des Verfahrens ausgegeben:

$$\mathcal{C}^* = \max_{1 \leq i \leq n} \{\mathcal{C}_i\}. \quad (7.35)$$

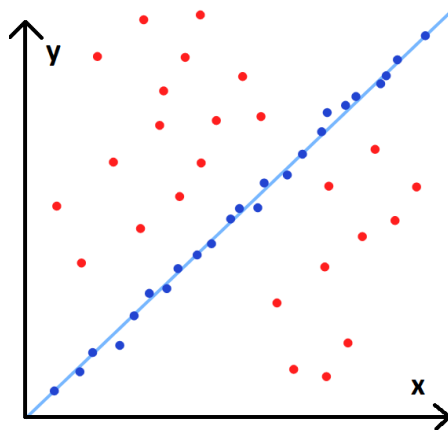


Abbildung 7.11.: Ausgleichsgerade, berechnet mit dem RANSAC-Verfahren; Ausreißer sind mit roter Farbe markiert, blaue Punkte gehören zur Consensus-Menge.

Obwohl es sich um ein randomisiertes Verfahren handelt, hat sich das RANSAC-Verfahren im Rahmen der Projektgruppe als sehr effizient erwiesen (siehe Kapitel 8.3). Abbildung 7.12 zeigt die Oberseite des zu erkennenden Paketes, die durch das RANSAC-Verfahren ermittelt wurde. Die resultierende Consensus-Menge C^* entspricht hier den Punkten der Oberseite des Paketes [24].

Mittelpunktsbestimmung

Mit den Punkten der Oberseite des Paketes kann nun sowohl der Mittelpunkt also auch die Orientierung des Paketes ermittelt werden. Dafür muss die Kontour der Oberseite extrahiert werden, d.h. die Ränder identifiziert. Diese Kontrurextraktion findet allerdings nicht mit den Eingabepunkten statt, sondern mit den reduzierten Punkten, die RANSAC übrig lässt (siehe vorheriger Abschnitt), wodurch sich die Kontourextraktion stark vereinfacht. Es reicht die Punktmenge auf eine 2D-Ebene zu projizieren und die konvexe Hülle der projizierten Punkte zu bestimmen. So erhält die Kontur des Paketes. Die Berechnung der konvexen Hülle einer Punkt Menge ist effizient durchführbar. Im Rahmen der Projektgruppe kam hierfür das Paket Qhull⁸ zum Einsatz, welches ebenfalls ein Teil der PCL ist.

Mit der konvexen Hülle lassen sich dann der Mittelpunkt und die Orientierung des Paketes ermitteln. Da die Oberfläche des Paketes rechteckig ist, stimmt der Mittelpunkt der Oberfläche mit dem Mittelpunkt eines flächenminimalen achsenparallelen Hüllrechtecks überein. Um ein solches Hüllrechteck zu finden, werden die Punkte aus der konvexen Hülle gesucht, welche die größten bzw. kleinsten Koordinatenwerte besitzen. Abbildung 7.13 zeigt das flächenminimale achsenparallele Hüllrechteck (gestrichelte Linien) einer konvexen Hülle (rote Punkte). Ist das Hüllrechteck gegeben, d.h. also die minimalen und maximalen Koordinatenwerte $x_{min}, y_{min}, x_{max}, y_{max}$ der konvexen Hülle, so ergibt sich der Mittelpunkt m (blauer Punkt) durch

⁸<http://www.qhull.org/>, Stand 15.3.2014

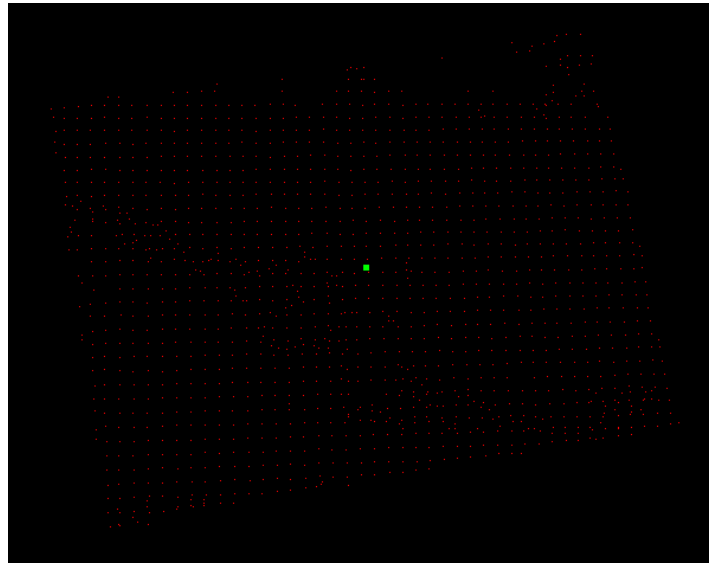


Abbildung 7.12.: Durch das RANSAC-Verfahren ermittelte Paketoberseite

$$m_x = \frac{x_{max} - x_{min}}{2}, m_y = \frac{y_{max} - y_{min}}{2} \quad (7.36)$$

und lässt sich somit sehr effizient berechnen. Für die Extremwerte $x_{min}, y_{min}, x_{max}$ und y_{max} müssen alle Punkte der konvexen Hülle einmal durchlaufen werden. Allerdings ist die Punktmenge der konvexen Hülle in der Regel sehr klein (weniger als 20 Punkte), wodurch auch diese Werte sehr schnell ermittelt werden können.

7.3.3. Berechnung der relativen Paketposition

Ist die Paketposition und -ausrichtung berechnet, sind diese Informationen nur im Koordinatensystem der Tiefenkamera bekannt. Benötigt werden die Daten allerdings für das Koordinatensystem im Arbeitsraum des Roboterarmes. Um diese Informationen zu erhalten, müssen die berechneten Koordinaten also in das Koordinatensystem des Arbeitsraumes *transformiert* werden.

Die Transformation hängt stark von der Kalibrierung der Tiefenkamera ab, die in Kapitel 7.3.1 detailliert beschrieben ist. Aus ihr ergeben sich die erforderlichen Daten für die korrekte Transformation in das Koordinatensystem des Arbeitsraumes. Da es sich bei beiden Koordinatensystemen um affine Koordinatensysteme handelt, ist für die Transformation eine *affine* Abbildung nötig. Deshalb sind neben der klassischen *Transformationsmatrix* \mathbf{A} auch ein Translationsvektor \mathbf{t} für die Transformation gegeben. Die affine Abbildung kann dann wie folgt definiert werden:

$$f(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} + \mathbf{t}. \quad (7.37)$$

Für einen beliebigen Punkt im Koordinatensystem der Tiefenkamera $\mathbf{x} \in \mathbb{R}^3$ können somit die Koordinaten im Koordinatensystem des Arbeitsraumes berechnet werden. Auf diese Weise

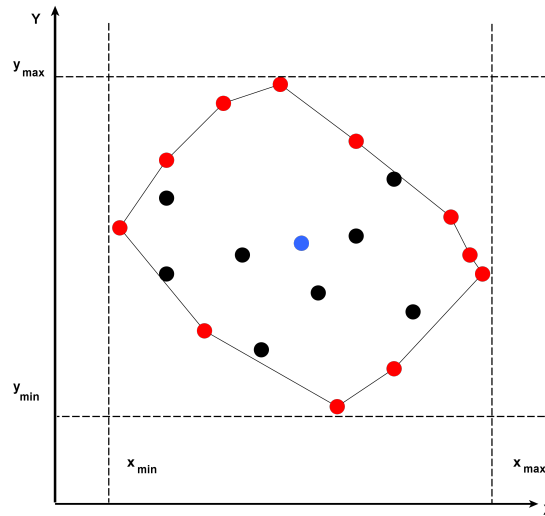


Abbildung 7.13.: Die Berechnung des Mittelpunktes (blau) findet über die Ermittlung der Extremwerte der konvexen Hülle (rot) statt.

kann sowohl die Position als auch die Ausrichtung des Paketes (in Form der Eckpunkte) in das gewünschte Koordinatensystem übertragen werden.

Um die Transformation auf die Angabe einer Transformationsmatrix zu beschränken, werden im Rahmen der Projektgruppe zur Transformation *homogene* Koordinaten verwendet. Um einen beliebigen Vektor $\mathbf{x} \in \mathbb{R}^3$ im Koordinatensystem als homogenen Koordinatenvektor anzugeben, wird lediglich eine *homogenisierende* Koordinate zusätzlich zum Vektor hinzugefügt:

$$\mathbf{x}_h = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}. \quad (7.38)$$

Mithilfe von homogener Koordinaten kann die Transformation nun durch die Verwendung einer *erweiterten Abbildungsmatrix* \mathbf{A}_{erw} durchgeführt werden, die die Transformationsmatrix \mathbf{A} mit dem Translationsvektor \mathbf{t} verbindet:

$$\mathbf{A}_{\text{erw}} = \begin{pmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}. \quad (7.39)$$

Dann lässt sich die Transformation wie folgt definieren:

$$f(\mathbf{x}) = \mathbf{A}_{\text{erw}} \cdot \mathbf{x}_h \quad (7.40)$$

Die Angabe einer erweiterten Transformationsmatrix ist für die Transformation der Koordinaten also ausreichend. Zu beachten ist allerdings, dass es sich bei dem resultierenden Vektor $f(\mathbf{x})$ des Arbeitsraumkoordinatensystems ebenfalls um einen homogenen Vektor handelt [52].

7.4. Prädiktion der Paketposition

Nachdem die Fahrzeugposition in Abschnitt 7.2 und die Paketposition in Abschnitt 7.3 bestimmt wurden, müssen diese beiden Informationen zusammengefügt werden. Weiter soll ein Rauschen auf den Positionsdaten wenig Einfluss auf die Präzision des Systems haben. Das Zusammenfügen der Daten und Glättung von Rauschen in den Daten kann zum Beispiel durch einen Tracking-Algorithmus realisiert werden. Neben diesen Anforderungen werden auch zukünftige Positionsdaten benötigt. Zum Einen, damit der Roboter sich rechtzeitig mit dem Fahrzeug bewegen kann und zum Anderen damit im Falle der Blindheit der Sensoren, wie zum Beispiel während des Greifprozesses, die Position des Paketes trotzdem bekannt ist. Für die Realisierung wurde der Condensation-Algorithmus (siehe Abschnitt 6.2.3) gewählt, da er die größte Flexibilität bietet. Die Anwendung des Condensation-Algorithmus wird in Abschnitt 7.4.2 erläutert. Um einen generellen Anhaltspunkt über den Verlauf der Positionsdaten zu erhalten, muss die Bahn, der das Fahrzeug im Optimalfall folgt, ermittelt werden. Dies wird in Abschnitt 7.4.1 näher betrachtet.

7.4.1. Approximation der Fahrzeugbahn

Die Bahn des Fahrzeuges kann einen Anhaltspunkt dafür bieten, wo sich das Fahrzeug in der Roboterzelle aufhält und wie es sich verhält. Weiter stellt die Bahn einen wichtigen Baustein im Condensation-Algorithmus dar (siehe Abschnitt 7.4.2).

Die Bahn, die dem Fahrzeug durch die Fahrzeugsteuerung vorgegeben ist, ist unbekannt⁹ und soll ermittelt werden.

Die Fahrzeugbahn soll im Folgenden als Kurve mit einem Kurvenparameter s verstanden und durch $\mathbf{track}(s)$ bezeichnet werden. Da sich das Fahrzeug in einer planaren Ebene, dem Hallenboden, bewegt, wird eine zweidimensionale Kurve angenommen. Unebenheiten des Hallenbodens werden vernachlässigt. Die Kurve wird in die beiden Raumrichtungen unabhängig betrachtet, sodass

$$\mathbf{track}(s) = \begin{pmatrix} x_{track}(s) \\ y_{track}(s) \end{pmatrix} \quad (7.41)$$

gilt.

Da es außer wenigen Zwischenpunkten im Roboterkäfig keine Anhaltspunkte über den Verlauf der Bahn gibt, werden für die Identifikation der Bahn Fahrzeugpositionen innerhalb des Roboterkäfigs benötigt, die das Fahrzeug durchfährt. Diese werden mit einem externen Messsystem¹⁰ aufgenommen. Als Ergebnis ergibt sich Matrix $\mathbf{D} \in \mathbb{R}^{3 \times n}$, die Zeit, x-Position und y-Position von n Messungen enthält. Um Geschwindigkeitsvariationen über die Zeit zu erhalten, aber trotzdem eine Parametrisierung mit Parameterwerten zwischen 0 und 1 im

⁹Der Aufbau der Steuerung ist für die PG nicht einsehbar, sodass nur Zwischenpunkte vorgegeben werden können. Die Lage dieser Punkte innerhalb des Roboterkoordinatensystems könnte durch eine Koordinatentransformation bestimmt werden. Aufgrund der Unkenntnis über das Verhalten des Fahrzeuges würde diese Transformation aber nur wenig Aufschluss bringen.

¹⁰Da die Erkennung des Fahrzeuges auf dem CCD-Bild und die anschließende Transformation der Bildposition in Weltkoordinaten genau genug sind, wurde das CCD-Modul als externes Messsystem verwendet.

interessierenden Bereich zu erhalten, wird die Zeitspalte normiert¹¹. Die Datenmatrix \mathbf{D} wird in ihre drei Komponenten zerlegt, sodass sich die Vektoren $\mathbf{t} \in \mathbb{R}^n$, $\mathbf{x}_D \in \mathbb{R}^n$ und $\mathbf{y}_D \in \mathbb{R}^n$ ergeben. Zur Veranschaulichung wurden in Abbildung 7.14a die aufgenommenen Positionsdaten planar aufgetragen. Abbildung 7.14b zeigt den ermittelten Verlauf der Raumrichtung x über die Zeit und Abbildung 7.14c den verrauschten Verlauf von y über die Zeit.

Im Folgenden soll nun mit Hilfe von \mathbf{D} $track(s)$ identifiziert werden. Um den Suchraum aller möglichen Funktionen einzuschränken werden $x_{track}(t)$ und $y_{track}(t)$ als Polynome angenommen. Der Grad r des Polynoms soll ebenfalls ermittelt werden.

Zunächst soll $x_{track}(t)$ bestimmt werden. Dazu wird $x_{track}(t)$ als Polynom r -ten Grades mit Parameter $\mathbf{a} = (a_0, \dots, a_r)$, $\mathbf{a} \in \mathbb{R}^r$ angenommen, sodass

$$x_{track}^{\mathbf{a}}(s) = a_r \cdot s^r + a_{r-1} \cdot s^{r-1} + \dots + a_1 \cdot s + a_0. \quad (7.42)$$

Mithilfe von \mathbf{t} und \mathbf{x}_D sollen die $(r + 1)$ Parameter \mathbf{a} des Polynomes identifiziert werden. Um eine Lösung zu ermitteln, würden $(r + 1)$ zufällig ausgewählte Messpunkte ausreichen. Da jedoch von rauschbehafteten Positionsdaten ausgegangen werden muss, würden die Parameter fehlerbehaftet sein.

Daher wird der Ansatz des kleinsten quadratischen Fehlers gewählt, um die Parameter des Polynoms zu bestimmen. Es soll das Polynom ermittelt werden, das „am besten“ zu den Daten passt. Die gewählte Gütefunktion G , als Kriterium für die beste Anpassung, beschreibt den quadratischen Fehler zwischen Daten und Polynom mit

$$G = \sum_{k=1}^n (x_k - x_{track}^{\mathbf{a}}(t_k))^2. \quad (7.43)$$

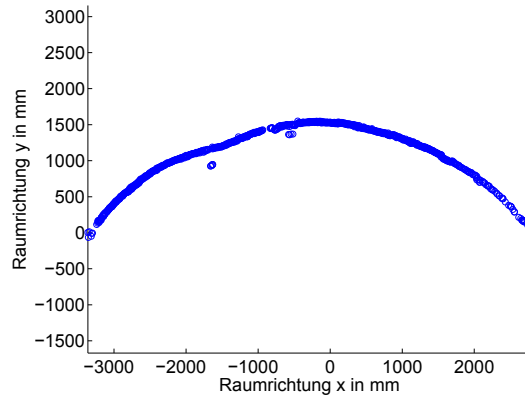
Durch Minimierung der Funktion G können die Parameter \mathbf{a} ermittelt werden. Dabei sollte gelten, dass $n \gg r$ ist.

Auf diese Weise können die Parameter eines Polynoms bestimmt werden. Um nun auch den passenden Grad des Polynoms bestimmen zu können, muss über verschiedene mögliche Polynomgrade iteriert werden. In jeder Iteration kann dabei der verbleibende quadratische Fehler berechnet werden. Für die Auswahl des passenden Grades kann entweder

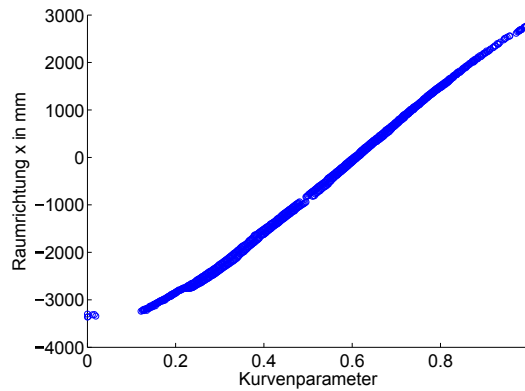
- über die Grade 0 bis $\frac{(n-1)}{2}$ iteriert und der Grad mit dem kleinsten quadratischen Fehler ausgewählt werden oder
- über die Grade 0 bis $\frac{(n-1)}{2}$ iteriert und nach der Sichtung der Ergebnisse der einzelnen Iteration das plausibelste Polynom ausgewählt werden.

Es wird $\frac{(n-1)}{2}$ als Obergrenze für die Iteration gewählt, da ein zu hoher Polynomgrad zur Überanpassung des Polynoms führt. Daher führt auch die Sichtung der Polynome oft zu einer besseren Lösung, da so die im hier betrachteten Anwendungsfall beobachtbare Bahn abgeschätzt werden kann.

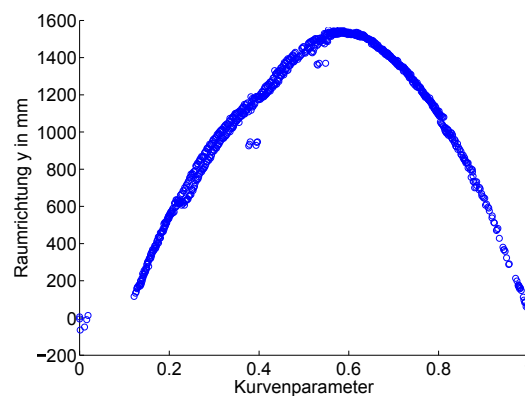
¹¹Angenommen die Zeitspalte reicht von 0 bis t_{\max} ergibt sich die Normierung durch Division des gesamten Vektors durch t_{\max} .



(a) Darstellung der gemessenen Positionsdaten in der Ebene. Die Zeitinformation wird für die Veranschaulichung vernachlässigt.



(b) Verlauf der Raumrichtung x der gemessenen Daten aufgetragen gegen die normierte Zeiteinheit



(c) Verlauf der Raumrichtung y der Daten aufgetragen gegen die normierte Zeiteinheit

Abbildung 7.14.: Um die Aufgabe der im Anwendungsfall geforderten Identifikation zu veranschaulichen wurden die Positionsdaten, aus verschiedenen Blickwinkeln betrachtet, dargestellt.

Selbige Schritte werden für die Ermittlung von $y_{track}(s)$ durchgeführt. In der konkreten Umsetzung hat sich der Grad 3 für $x_{track}(t)$ und ein Polynomgrad von 5 für $y_{track}(s)$ als plausibel herausgestellt. Abbildungen 7.15a, 7.15b und 7.15c zeigen die identifizierten Funktionen.

Innerhalb des Entwicklungsprozesses hat sich herausgestellt, dass das Auswerten der Polynome zu zeitintensiv ist, besonders, da die Funktion häufig ausgewertet wird. Daher wird vor Programmstart in einem Vorverarbeitungsschritt eine Look-up-Table angelegt.

7.4.2. Anwendung des Condensation-Algorithmus

Zum Tracken der aktuellen Position des Pakets wird der Condensation-Algorithmus verwendet. Dieser wird in Kapitel 6.2.3 im Allgemeinen beschrieben. Im Folgenden wird die explizite Realisierung des Verfahrens näher erläutert. Dabei wird zunächst ein grober Überblick gegeben, bevor näher auf die deterministische Transformation eingegangen wird. Des Weiteren wird die Berechnung der Beobachtungsverteilung beschrieben und erklärt, wie eine Prädiktion der Paketposition für spätere Zeitpunkte geleistet wird.

Das Hauptaugenmerk bei der Implementierung des Algorithmus liegt auf der Repräsentation und Veränderung der Samples. Ein einzelnes Sample besteht aus den zwei Koordinaten x und y sowie einer Geschwindigkeit v . Die Samples werden in einer einfachen Liste verwaltet. Initialisiert werden die Samples vor Beginn des Trackingvorgangs auf einem Grid im Raum der möglichen Fahrzeugpositionen mit der Geschwindigkeit $v = 0$. In jeder Iteration des Algorithmus werden folgende Schritte durchgeführt:

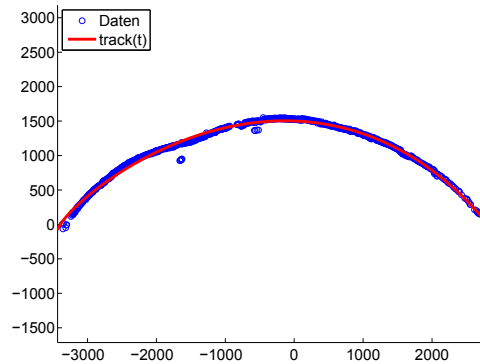
1. Anwendung der deterministischen Transformation auf alle Samples.
2. Anwendung einer normal-verteilten Streuung auf alle Samples.
3. Aktualisierung der Geschwindigkeit der Samples.
4. Bewertung der Samples anhand der Beobachtungen.
5. Wahl neuer Samples aus der Menge der alten Samples.

Abbildung 7.16 stellt Schritte 1, 2, 4 und 5 grafisch dar. Sie zeigt zunächst das Wählen der neuen Samples aus den bewerteten Samples der letzten Iteration bevor die Transformationen angewendet und die neuen Samples bewertet werden.

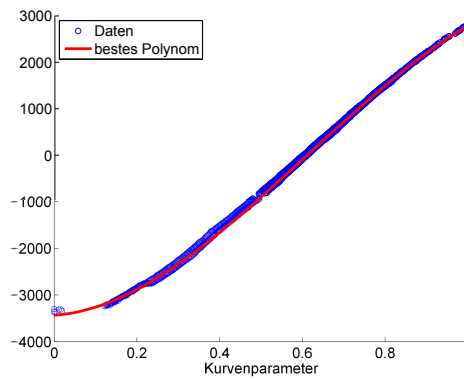
Zur Umsetzung von Schritt 2 werden anhand der Normalverteilung für jedes Sample zwei zufällige Werte erzeugt und auf die x - und y -Koordinaten dieses Samples addiert. Als Parameter werden der Erwartungswert 0 und eine vom Maßstab des Koordinatensystems abhängige Standardabweichung σ verwendet. Der Erwartungswert von 0 sorgt für eine Verteilung der Samples um ihre Ursprungsposition.

Die in Schritt 3 erwähnte Geschwindigkeit v eines Samples wird durch die Parametergeschwindigkeit des Samples auf der approximierten Kurve repräsentiert, welche in Kapitel 7.4.1 näher beschrieben wird. Sie wird nach der Transformation eines Samples neu bestimmt. Dazu werden zunächst die Punkte auf der Kurve approximiert, die dem Sample vor und nach der Transformation am nächsten liegen. Im Folgenden wird die Parameteränderung dieser Kurvenpunkte durch die verstrichene Zeit der Iteration geteilt. Nach dieser Normierung wird der Wert als Geschwindigkeit v gespeichert.

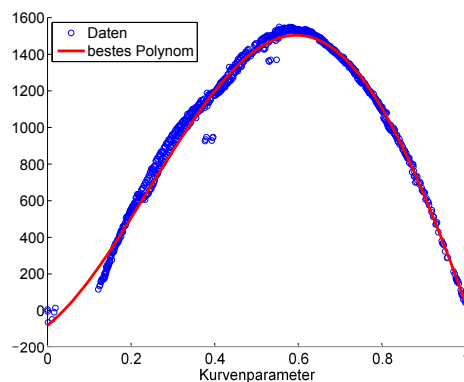
Die Wahl der neuen Samples in Schritt 5 erfolgt durch Ziehen mit Zurücklegen von Samples aus



(a) $track(s)$ und die gemessenen Positionsdaten in der Ebene aufgetragen bei Vernachlässigung des Kurvenparameters/der Zeitinformation.



(b) Raumrichtung x der Daten und des angepassten Polynoms aufgetragen gegen die normierte Zeiteinheit/den Kurvenparameter.



(c) Raumrichtung y der Daten und des angepassten Polynoms aufgetragen gegen die normierte Zeiteinheit/den Kurvenparameter

Abbildung 7.15.: Zur Veranschaulichung der Lösung die von der Identifizierung gefunden wurde, werden die Daten und die identifizierten Polynomen aus unterschiedlichen Blickwinkeln dargestellt.

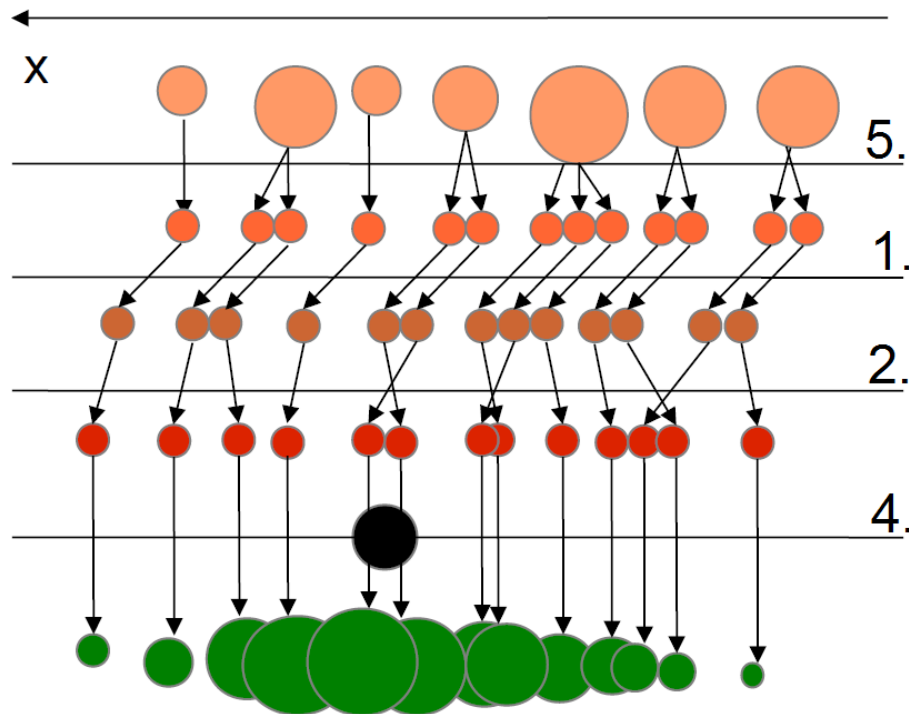


Abbildung 7.16.: Eine Iteration des Condensation Algorithmus. Zunächst werden aus den bewerteten alten Samples die neuen Samples gewählt. Im folgenden werden deterministische und nicht-deterministische Transformation auf diese angewendet. Anhand der Beobachtung können die transformierten Samples bewertet werden.

der alten Menge. Verwendet wird dabei die Verteilungsfunktion der Beobachtungsbewertung. Diese wird durch ein Array mit Wahrscheinlichkeiten repräsentiert, welche sich zu 1 addieren. Dementsprechend wird für jedes neue Samples eine gleichverteilte Zahl zwischen 0 und 1 berechnet und das neue Sample jeweils anhand der Beobachtungsbewertung ausgewählt. Zum Einordnen der gezogenen Zufallszahl in das Array der Beobachtungsbewertung wird die binäre Suche verwendet.

Abbildung 7.17 zeigt die Samples verteilt um die aktuelle Fahrzeugposition. Der orange Punkt zeigt die vermutete aktuelle Position des Fahrzeugs und der rote Punkt die Prädiktion für einen Zeitpunkt in der Zukunft (siehe Kapitel 7.4.3). Die bisher nicht näher erläuterten Schritte 1 und 4 werden im Folgenden in den Unterkapiteln detailliert beschrieben.

Umsetzung der deterministischen Transformation

Bei der deterministischen Transformation soll jedes der 971 Samples so transformiert werden, wie es aufgrund der aktuell für das Sample gespeicherten Geschwindigkeit v und Position $\mathbf{p} = (x, y)^T$ im Zeitintervall t bis zur nächsten Iteration des Condensation-Algorithmus zu erwarten ist.

Ein grundlegendes Problem besteht darin, dass Samples und auch das Fahrzeug sehr selten genau auf der approximierten Fahrzeugbahn (siehe Abschnitt 7.4.1) positioniert sind und das genaue Verhalten des Fahrzeugs bei Abweichungen von der vorgegebenen Bahn nicht bekannt

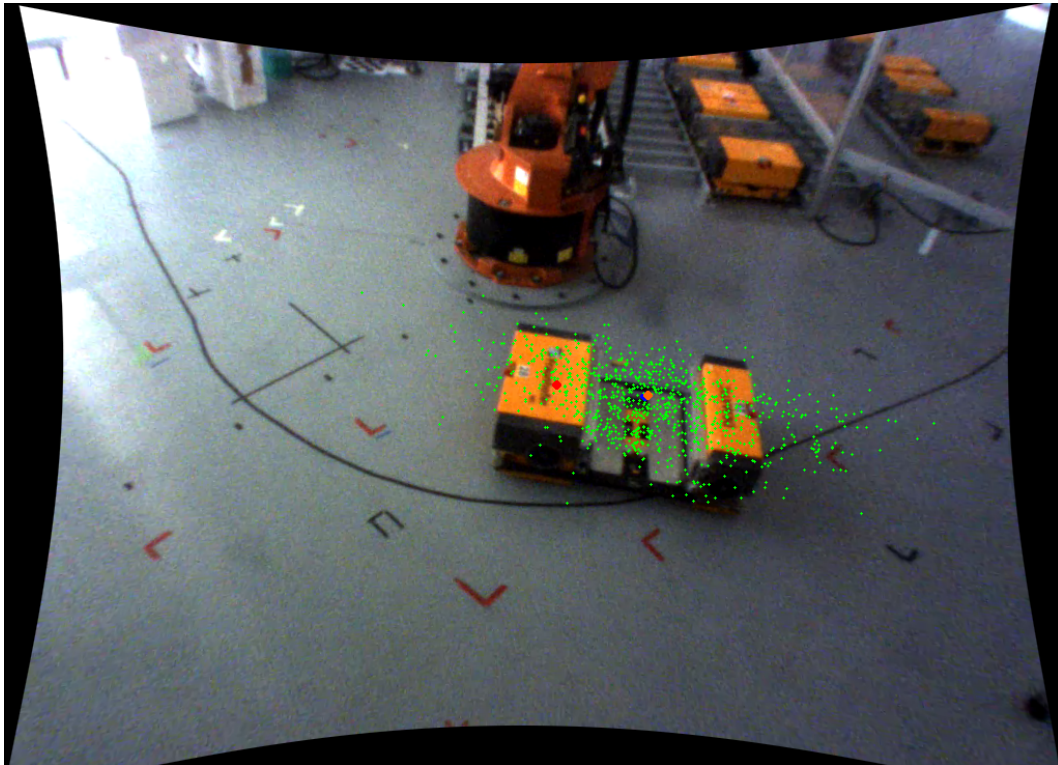


Abbildung 7.17.: Um die aktuelle Fahrzeugposition verteilte Condensation-Samples. Die Samples sind in grün eingezeichnet. Der orange Punkt zeigt die vermutete aktuelle Position des Fahrzeugs und der rote Punkt eine Prädiktion für einen Zeitpunkt 2000 ms später.

ist. Die Implementierung des Tracking-Moduls nutzt in diesem Fall die Annahme, dass die Bewegung zunächst so fortgesetzt wird, als ob keine Abweichung von der Bahn vorliegen würde. Es ist zwar zu erwarten, dass eine Korrektur stattfinden wird, sobald das Fahrzeug seine Position zuverlässig ermitteln konnte – der Zeitpunkt dafür ist jedoch nicht vorherzusehen und die Feinheiten der Fahrzeugsteuerung sind der Projektgruppe nicht zugänglich¹². Daher wird eine eventuelle Korrektur der gefahrenen Bahn durch die deterministische Transformation für den Condensation-Algorithmus nicht berücksichtigt.

Sei $(x', y')^T$ ein Punkt der Fahrzeugbahn mit kleinstem euklidischem Abstand¹³ zu \mathbf{p} . Es wird bei der deterministischen Transformation davon ausgegangen, dass die absolute Abweichung der aktuellen Position von der vorgegebenen Bahn, also der Wert $(x, y)^T - (x', y')^T$, über die Zeit konstant bleibt. Nicht berücksichtigt wird dabei ein eventuell vorhandener Fehler der aktuellen Fahrzeugausrichtung und dessen Auswirkungen auf die zukünftige Bewegung.

¹²Die Fahrzeugsteuerung wurde in einer Abschlussarbeit für das Fraunhofer-Institut für Materialfluss und Logistik entwickelt. Der Quelltext oder ausführliche Dokumentation zur Implementierung konnten der Projektgruppe nicht zur Verfügung gestellt werden. Eine genaue Aussage über die Art der Positionskorrektur konnte somit nicht als Grundlage des Trackingverfahrens dienen.

¹³Da die Fahrzeugbahn nur eine geringe Krümmung aufweist, ist dieser Punkt in der Regel eindeutig oder mehrere mögliche Punkte liegen dicht beieinander, so dass die Uneindeutigkeit keine signifikanten Auswirkungen auf das Ergebnis besitzt.

Somit erfolgt die deterministische Transformation eines Samples in drei Schritten:

- Bestimmung eines Punktes $track(s) = (x', y')^T$ auf der Fahrzeugbahn mit geringstem euklidischen Abstand zu $(x, y)^T$ sowie des dazugehörigen Kurvenparameters s ,
- dem Zeitintervall t entsprechende Bewegung des Punktes $(x', y')^T$ entlang der Fahrzeugbahn durch Berechnung des Kurvenpunkts $track(s + t \cdot v) = (x'_{neu}, y'_{neu})^T$ sowie
- Addition der Abweichung der alten Sampleposition von der Bahn zur ermittelten neuen Bahnposition ergibt die transformierte Sampleposition $(x_{neu}, y_{neu})^T = (x'_{neu}, y'_{neu})^T + (x, y)^T - (x', y')^T$.

Da eine direkte Umsetzung dieses Verfahrens sich bei der Ausführung als zu langsam¹⁴ herausgestellt hat, wurde für die Implementierung eine heuristische Optimierung vorgenommen. Stark ins Gewicht fiel die Berechnung der nächstliegenden Bahnpunkte für jedes Sample. Statt dies im zweiten Schritt für jedes Sample entlang der Bahn zu berechnen, wird nur für das Sample mit der aktuell größten Geschwindigkeit eine solche Translation entsprechend des Verlaufs der Bahnkurve berechnet. Diese Transformation wird dann auch für den zweiten Schritt jedes anderen Samples angewendet, wobei eine Skalierung entsprechend der für das jeweilige Sample gespeicherten Geschwindigkeit stattfindet. Mit der Annahme, dass der kurzfristige weitere Verlauf der Fahrzeugbahn für alle Sample-Positionen sehr ähnlich ist, hat dies nur geringe Auswirkungen auf die Korrektheit der Transformation. Da im Falle der Anwendung der Projektgruppe stets nur ein Fahrzeug vorhanden ist und somit alle Samples in direkter Nähe zueinander liegen, erscheint diese Heuristik als ein guter Kompromiss für die schnellere Berechnung der deterministischen Transformation.

Berechnung der Beobachtungsverteilung

Bei der Berechnung der Beobachtungsverteilung soll eine Wahrscheinlichkeitsverteilung über den Samples bestimmt werden. Die einem Sample durch die Verteilung zugeordnete Wahrscheinlichkeit soll dabei um so größer sein, je wahrscheinlicher es ist, dass dieses Sample die tatsächliche Position des Fahrzeugs beschreibt.

Für die Anwendung der Projektgruppe steht zunächst nur eine durch die CCD-Kamera beobachtete Position zur Verfügung. Im weiteren Verlauf kann zusätzlich eine durch die Tiefenkamera gelieferte Position hinzu kommen. Grundsätzlich wird immer nur die jeweils neueste Beobachtung einer Kamera für die Bewertung der Samples genutzt. Außerdem wird eine Beobachtung verworfen, wenn diese bereits älter als eine feste Zeitschranke ist. Stehen aktuelle Positionsmessungen beider Kameras zur Verfügung, wird ein gewichteter Mittelwert für die weitere Verwendung gebildet. Falls keine aktuellen Beobachtungswerte vorhanden sind, so wird die aktuell durch das Tracking angenommene Position als „Beobachtung“ genutzt. Somit steht für das weitere Vorgehen immer genau eine Position zur Verfügung, die als aktuelle Beobachtung angesehen wird.

Um die Berechnung der Verteilung durchzuführen, wird für jedes Sample zunächst die euklidische Distanz zwischen Sampleposition und Beobachtungsposition gebildet. Damit die Wahrscheinlichkeit mit zunehmender Entfernung von der Beobachtungsposition nicht nur

¹⁴Die Ausführungszeit auf der genutzten Hardware lag bei einigen hundert Millisekunden, während für eine zeitnahe Berücksichtigung der aktuell gemessenen Positionen eine Verarbeitungszeit von deutlich unter hundert Millisekunden wünschenswert ist.

linear abnimmt, wird die ermittelte Distanz jedes Samples anhand der Dichtefunktion einer Normalverteilung bewertet. Die Beobachtungsverteilung wird bestimmt, indem die Summe der Bewertungen aller Samples berechnet und jede Bewertung durch die Summe geteilt wird (Normalisierung), so dass für jedes Sample eine Wahrscheinlichkeit entsteht.

7.4.3. Prädiktion über größere Zeitintervalle

Eine der wichtigsten Anwendungen des implementierten Tracking-Verfahrens ist die Vorhersage einer zukünftigen Paketposition. Der Zeitpunkt, für den eine Prädiktion im Anwendungsfall geleistet werden muss, liegt bis zu 2500 *ms* in der Zukunft. Solange im Voraus wird die Position zum Greifen des Paketes geplant. Benötigt wird die Prädiktion, um den Ort zu ermitteln, an dem der Roboterarm das Paket in der Zukunft entnehmen soll. Als Grundlage für diese Prädiktion dient die regelmäßig aktualisierte aktuelle Position (x, y) und Geschwindigkeit v des Pakets, die der Condensation-Algorithmus berechnet. Die Position wird mit Hilfe der deterministischen Transformation verschoben. Als Eingabe dient dazu der Zeitpunkt für den die Prädiktion geleistet werden soll. Zunächst wird die Zeitdifferenz zwischen diesem Zeitpunkt und der aktuellen Zeit berechnet. Um, wie in Kapitel 7.4.2 beschrieben, die Änderung des Kurvenparameters zu bestimmen, wird die aktuelle Geschwindigkeit v mit dieser Zeitdifferenz multipliziert.

Um die Güte dieser berechneten Prädiktion zu überwachen, wird in jeder Iteration des Condensation-Algorithmus eine solche Prädiktion für einen 2000 *ms* in der Zukunft liegenden Zeitpunkt bestimmt und gespeichert. Nach Ablauf des Zeitintervalls wird dieser gespeicherte Wert mit der aktuellen Position des Algorithmus verglichen und die Differenz als Fehlermaß gebildet. Unter Annahme der Konvergenz des Verfahrens wird dieses Fehlermaß verwendet, um zu entscheiden, wann der Roboter das Paket greifen wird. Wenn der Fehler eine Iterationen kleiner als ein Schwellwert ist, beginnt die Greifroutine der Robotersteuerung.

7.5. Positionierung des Roboter-Greifarms

Die Positionierung des Roboter-Greifarms basiert auf mehreren Schritten. Zuerst muss die Bewegungsform bestimmt und der Verlauf der Bahn berechnet werden. Dieser Vorgang basiert auf einem Potenzialfeld und wird in Abschnitt 7.5.1 erläutert. Anschließend wird die geplante Bewegung an die Robotersteuerung übermittelt, die wiederum den Roboter kontrolliert. Abschnitt 7.5.2 veranschaulicht diese Ansteuerung des Roboters. Für die Bewegungsplanung, aber auch die Auswertung der Bilder, die durch die Tiefenkamera aufgenommen wurden, muss die Position des Roboters zu jedem beliebigen Zeitpunkt bekannt sein. Abschnitt 7.5.3 beschreibt daher die Bestimmung der TCP-Position.

7.5.1. Planung der Roboterbewegung

Aufgabe der Bahnplanung ist es, die anwendungsfallspezifische Bewegung des Roboterarms zu bestimmen. Dies kann beispielsweise das Entladen des Pakets aus dem FTF und anschließende Ablegen auf einer bereitgestellten Palette sein. Dabei müssen Kollisionen ausgeschlossen werden. Außerdem sollte die Bewegungsbahn in möglichst geringer Zeit vom Roboter realisierbar sein,

da aufgrund der FTF-Geschwindigkeit nur eine geringe Zeitspanne zum Be- und Entladen zur Verfügung steht.

Die Robotersteuerung ermöglicht es dabei dem Roboter sich von der aktuellen Ausgangspose in eine frei wählbare Zielposition und Orientierung zu bewegen, solange diese erreichbar ist. Die Bewegungsbahn zwischen diesen Posen kann über die Bewegungsformen, PTP, linear oder kreisförmig (siehe Abschnitt 5.3.1), näher spezifiziert werden. Diese Bewegungsbahn wird von der Robotersteuerung ausgeführt, solange alle Abschnitte der Bahn im Arbeitsraum liegen und somit erreichbar sind. Mögliche Hindernisse werden dabei nicht berücksichtigt, da der Roboter nicht über die notwendigen Sensoren zur Detektion verfügt. Somit hat der Programmierer die Aufgabe Kollisionen bei der Bahnplanung auszuschließen. Für statische Anwendungsfälle, wie sie häufig in der Industrie auftreten, sind Testläufe mit schrittweise steigender Geschwindigkeit ausreichend, um den kollisionsfreien Ablauf zu garantieren. Der Arbeitsraum des Roboters bleibt dabei jeweils unverändert und es sind keine dynamischen Entscheidungen notwendig, die den Prozess beeinflussen könnten.

Für das dynamische Be- und Entladen von FTF sind nur wenige Positionen statisch. Die Palette, auf der die Pakete abgelegt, beziehungsweise von der sie entnommen werden, bleibt unverändert an einem fest definierten Standort. Außerdem ist die Abmessung des Arbeitsraums bekannt. Die Bewegungsplanung des Roboters basiert deshalb hauptsächlich auf Daten, welche zuvor von Kameras erfasst und von geeigneten Algorithmen analysiert wurden (siehe Kapitel 7.1 - 7.3). Aufgrund der hohen Varianz der so ermittelten Positionen ist es nicht möglich, alle denkbaren Bewegungsbahnen zu testen. Trotzdem müssen Kollisionen mit dem Fahrzeug, der Palette und anderen Hindernissen ausgeschlossen werden, auch für den Fall, dass die ermittelte Position des Fahrzeugs fehlerhaft sein sollte.

Lineare sowie kreisförmige Bahnen sind genau definiert, sodass die Bewegung vorhersagbar und auf Kollisionen überprüfbar ist. Die PTP-Bewegungsform wird hingegen abhängig von der Robotersteuerung hinsichtlich mehrerer Kriterien optimiert, sodass die resultierende Bewegung auch aufgrund des nicht bekannten Algorithmus der Steuerungssoftware nicht exakt vorhersagbar ist. Für die Problemstellung ungünstige Bewegungsbahnen, wie beispielsweise das Kopfübergreifen des Roboterarms, dürfen nichtsdestotrotz in keinem Fall ausgeführt werden. Sie können neben anwendungsspezifischen Problemen, wie einer ungünstigen Orientierung der Tiefenkamera, auch Beschädigungen der Hardware, bis hin zur Zerstörung des Roboters, beispielsweise durch Kollision mit dem Untergrund, zur Folge haben. Deshalb müssen die Bewegungen so definiert werden, dass die Freiheit der Robotersteuerung bei der Bewegungsplanung in jedem Fall gering genug ist, um einen ausreichenden Abstand zu allen potenziellen Hindernissen einzuhalten. Dies kann durch die Wahl der Bewegungsform geschehen, die eine eindeutige Bahn beschreiben. Im Fall der PTP-Bewegung muss dies dagegen durch eine Reihe von Zwischenpunkten mit begrenztem Abstand zueinander sichergestellt werden.

Der folgende Abschnitt beschreibt den Aufbau des Potenzialfeldes, welches zur Planung sämtlicher Bahnpunkte genutzt wird und darüber hinaus auch die Erreichbarkeit jeder Pose überprüft. Anschließend wird die Berechnung der Roboterbahn veranschaulicht, die das Potenzialfeld nutzt und dabei auch den maximalen Punktabstand und die anzuwendende Bewegungsform bestimmt.

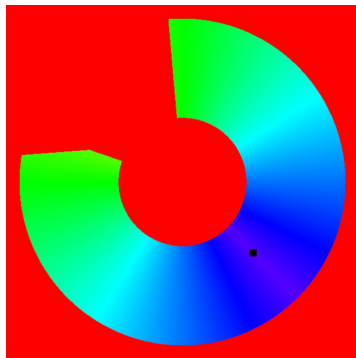
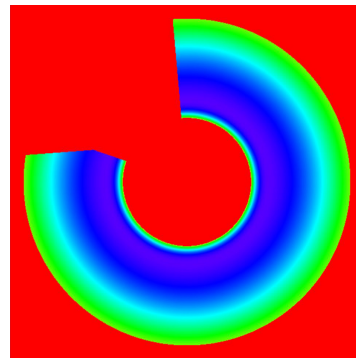
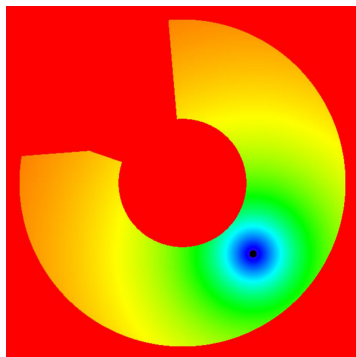
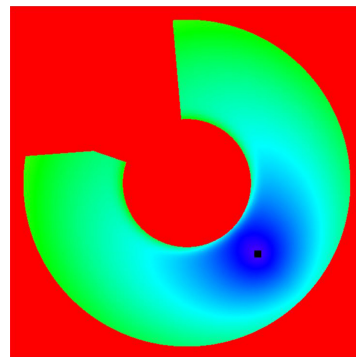
(a) Funktion $P_1(x, y)$ (siehe Gleichung (7.45))(b) Funktion $P_2(x, y)$ (siehe Gleichung (7.46))(c) Funktion $P_3(x, y)$ (siehe Gleichung (7.47))(d) Kombiniert $P(x, y)$ (siehe Gleichung (7.44))

Abbildung 7.18.: Darstellung der verschiedenen Anteile des Potenzialfeldes (siehe Abbildung 7.19 für die Farbcodierung des Potentials)

Aufbau des Potenzialfeldes

Ein Potenzialfeld wird, wie in Kapitel 5.3.3 dargelegt, durch eine beliebige Anzahl überlagerter Funktionen beschrieben. Das im Anwendungsfall der Projektgruppe für die Bahnplanung des Roboters konstruierte Potenzialfeld $P(x, y)$ besteht aus drei Anteilen, wie durch

$$P(x, y) = \sum_{i=1}^3 P_i(x, y) \quad (7.44)$$

ersichtlich wird. Die verwendeten Funktionen werden im Folgenden beschrieben.

Der Mittelpunkt des Potenzialfeldes $(0, 0)$ stimmt mit der Position des verwendeten Roboters überein. Die Abbildung 7.18d zeigt das resultierende Potenzialfeld. Der gewählte Zielpunkt $\mathbf{t} = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$ ist in Schwarz dargestellt. Dabei veranschaulicht die Farbe den Wert des Potentials. Die zugehörige Farbskala in Abbildung 7.19 nutzt den HSV-Farbraum und verläuft von lila Bereichen für niedrige Werte, bis hin zu roten Bereichen für sehr hohe Werte, welche nicht zum Arbeitsraum des Roboters gehören und somit von diesem entweder nicht erreichbar sind oder nicht durchfahren werden dürfen.



Abbildung 7.19.: Farbskala für die Visualisierung des Potenzialfeldes:
niedrige Werte: lila,
hohe Werte: rot

Die nah am Roboter und somit im Mittelpunkt des Potenzialfeldes liegenden Bereiche können genau wie die weit vom Mittelpunkt entfernten Gebiete nicht vom Sauggreifer des Roboters erreicht werden. Die genauen Abstände vom Mittelpunkt des Potenzialfeldes hängen von der Arbeitshöhe ab, wie in Abbildung 5.4 ersichtlich ist. Damit diese Höhenabhängigkeit in der Bahnplanung bedacht werden kann, wird das Potenzialfeld diskret höhenabhängig berechnet. Gleichung (7.46) modelliert dabei die Ausdehnung des Arbeitsbereichs.

Ein weiteres Gebiet des Potenzialfeldes wird vom Arbeitsraum ausgeschlossen, obwohl jeder einzelne enthaltene Punkt für sich, vom Roboter erreicht werden könnte. Dieser Bereich ist der rote Kreisausschnitt, links über dem Mittelpunkt, wie durch Abbildung 7.18 ersichtlich. Der Grund für diese Entscheidung ist eine Einschränkung des Industrieroboters, die ihn aufgrund von Kabelverbindungen daran hindern, sich beliebig häufig im Kreis zu drehen. Die imaginäre Linie vom Mittelpunkt des Potenzialfeldes nach oben kann von beiden Seiten nur um fünf Grad überschritten werden. Somit muss sich der Roboter, wenn er ein Paket von der Palette ansaugt, die sich auf der linken Seite dieser imaginären Linie befindet, und sich zum Fahrzeug auf der rechten Seite des Potenzialfeldes bewegen soll, in jedem Fall gegen den Uhrzeigersinn drehen. Auch für andere Beispiele ist der ausgeschlossene Bereich nicht vorteilhaft. Daher wurden die Ausschnitte zwischen der Palette und der nicht passierbaren Linie als unerreichbar markiert. Gleichung (7.45) beschreibt einen winkelabhängigen Anteil des Potenzialfeldes

$$P_1(x, y) = c_1 \cdot |a(x, y) - a(t_x, t_y)|. \quad (7.45)$$

Die Funktion $a(x, y)$ bestimmt den kontinuierlichen Winkel von (x, y) im Bezug zum Roboterkoordinatenursprung $(0, 0)$ in Grad. Der Faktor c_1 ist dabei eine positive skalierende Konstante. Somit wird der Betrag der Winkeldifferenz zwischen dem Zielpunkt \mathbf{t} und dem aktuell betrachteten Punkt (x, y) skaliert zum Gesamtpotenzial addiert. Dadurch wird eine kreisförmige Rampe modelliert, die zum Zielpunkt abfallend verläuft. Abbildung 7.18a stellt diesen Anteil des Potenzialfeldes dar. Die Winkelabhängigkeit motiviert sich durch den Industrieroboter, der im Koordinatenursprung platziert ist und dessen erste Achse genau diese Winkeldifferenz ausgleichen muss, um den Zielpunkt zu erreichen. Eine rein radiale Funktion ausgehend vom Ziel, wie durch Gleichung (7.47) angegeben, kann dies nicht in allen Fällen garantieren, da die so bestimmte Bahn beispielsweise in einem lokalen Minimum des Potenzialfeldes, nahe des nicht passierbaren Bereichs, enden könnte.

Die Gleichung (7.46) gibt an, welche Punkte abhängig vom Abstand zum Roboterursprung erreichbar sind:

$$P_2(x, y) = \begin{cases} c_2 \cdot \frac{2}{\exp((d_r - d_n) \cdot c_3) + 1} & \text{wenn } d_r \geq d_n \\ l & \text{wenn } d_r < d_n \end{cases} \quad (7.46)$$

$$+ \begin{cases} c_2 \cdot \frac{2}{\exp((d_f - d_r) \cdot c_4) + 1} & \text{wenn } d_r \leq d_f \\ l & \text{wenn } d_r > d_f. \end{cases}$$

Darüber hinaus werden Bereiche bevorzugt, die eine höhere Geschwindigkeit des Roboters ermöglichen und somit dazu beitragen, eine Bahn zu finden, die den Roboter für eine möglichst geringe Zeitspanne in Anspruch nimmt.

$d_r = \sqrt{x^2 + y^2}$ gibt den Abstand zwischen dem betrachteten Punkt (x, y) und dem Mittelpunkt des Potenzialfeldes an. Der minimale Abstand des Sauggreifers vom Roboterzentrum wird durch d_n angegeben und ist, genau wie der maximal realisierbare Abstand d_f , diskret von der Höhe abhängig. Die Faktoren c_2 , c_3 und c_4 sind positive skalierende Konstanten, welche heuristisch optimiert wurden, mit dem Ziel, schnelle Bereiche zu bevorzugen. Bei Versuchen hat sich gezeigt, dass der Roboter seine Geschwindigkeit besonders an den Grenzen des Arbeitsraums reduziert, wenn er beispielsweise nahezu vollständig ausgestreckt ist. Solche Bereiche werden sofern möglich bei der Bewegung nicht durchfahren. l zeigt an, dass ein Punkt nicht im Arbeitsbereich des Roboters liegt.

Gleichung (7.47) gibt eine vom Zielpunkt ausgehende, radial-symmetrische Funktion an:

$$P_3(x, y) = c_5 \cdot \left(1 - \frac{2}{\exp(d_t \cdot c_5) + 1} \right) \quad (7.47)$$

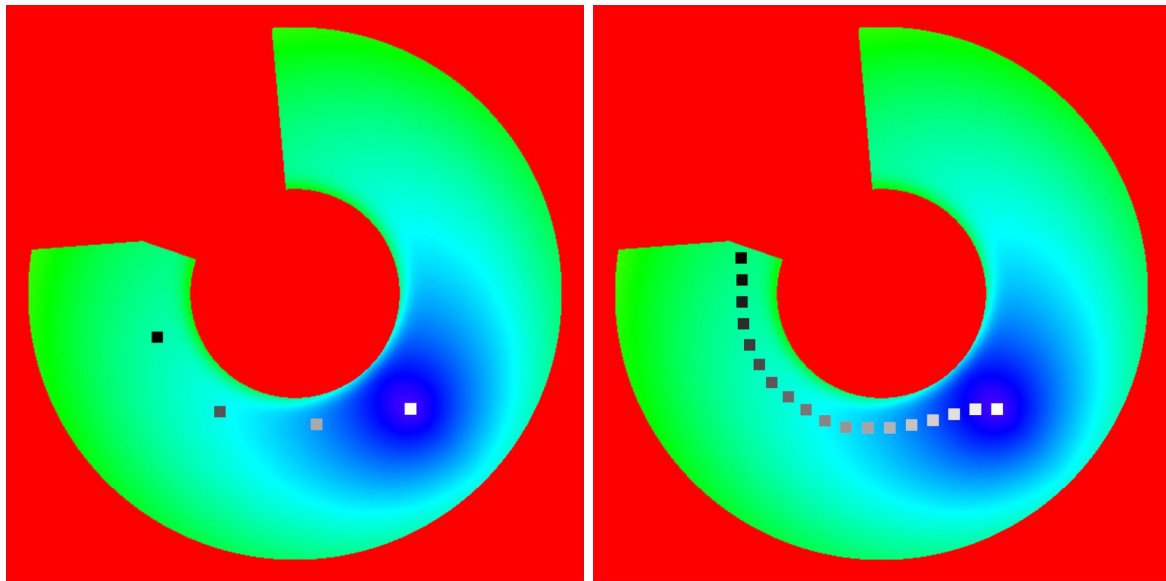
$$+ c_6 \cdot \left(1 - \frac{2}{\exp(d_t \cdot c_6) + 1} \right).$$

Dabei steht d_t für den Abstand zwischen dem betrachteten Punkt (x, y) und dem Zielpunkt t . Die Konstanten c_5 und c_6 sind positiv und skalieren den Einflussbereich der Funktion. Diese Komponente des Potenzialfeldes stellt sicher, dass der Pfad in der näheren Umgebung des Zielpunktes den direkten Weg verfolgt und das Ziel erreicht. Die anderen beiden Komponenten würden das Eintreffen am Zielpunkt alleine nicht garantieren, sondern nur den Winkel der ersten Roboterachse, und somit die Richtung vom Ursprung aus gesehen, korrigieren.

Insgesamt werden für die Berechnung des Potenzialfeldes mehrere Funktionsanteile addiert, die alle separat betrachtet, positiv sind und monoton verlaufen. Somit werden lokale Minima reduziert. Ausgeschlossen werden können sie aufgrund der Überlagerung verschiedener Funktionen, die wiederum von dem gewählten Zielpunkt und der Höhe abhängen, allerdings nicht. Dies muss bei der Berechnung der Roboterbahn zusätzlich berücksichtigt werden.

Berechnung der Roboterbahn

Die Berechnung der Roboterbahn muss auf den jeweiligen Anwendungsschritt angepasst sein. Das Verfolgen des FTFs erfordert beispielsweise andere Bewegungsmuster als das Ablegen des Pakets auf der bereitgestellten Palette.



(a) Ein Meter maximaler Punktabstand

(b) 20 Zentimeter maximaler Punktabstand

Abbildung 7.20.: Darstellung der Roboterbahn auf dem Potenzialfeld mit verschiedenen maximalen Abständen der Zwischenpunkte

Um dies zu realisieren, wird zuerst die notwendige Bewegungsform gewählt. Alle Bewegungen in unmittelbarer Nähe von Hindernissen sowie alle Bewegungen, die eine exakt berechnete Geschwindigkeit des Roboters erfordern, werden durch lineare Bahninterpolation umgesetzt. Dadurch ist sichergestellt, dass keine signifikanten Abweichungen von der vorbestimmten Bahn auftreten und somit auch keine Kollisionen durch diesen Faktor hervorgerufen werden. Dies ist besonders beim Ansaugen und Ablegen des Pakets, was in der unmittelbaren Nähe von Hindernissen, wie Boden, Fahrzeug und Palette, ausgeführt wird, der Fall. Darüber hinaus lassen sich genaue Geschwindigkeiten nur bei linear oder kreisförmig interpolierenden Bewegungen vorgeben.

Alle weiteren Bahnabschnitte, welche einen ausreichenden Abstand zu allen Hindernissen aufweisen und in möglichst kurzer Zeit überbrückt werden sollen, werden mit PTP-Bewegungen realisiert. Dabei kann die Bahn, wie in Abbildung 5.6 verdeutlicht, nicht exakt bestimmt werden und auch die Bahngeschwindigkeit lässt sich nicht absolut festlegen. Dennoch ist diese Bewegungsform meist deutlich schneller. Die PTP-Bewegung wird ausnahmslos in einer festgelegten Minimalhöhe verwendet, sodass Kollisionen mit dem Boden, der Palette und dem FTF, deren Höhen bekannt sind, ausgeschlossen werden können. Weitere Hindernisse, wie der Roboter und die Eingrenzung der Roboterzelle, können nicht beschädigt werden, da sie sich außerhalb des vom Potenzialfeld festgelegten Arbeitsraums befinden.

Die Robotersteuerung bestimmt den genauen Verlauf der PTP-Bahn zwischen den Punkten durch interne Algorithmen. Die Bahn kann aber durch Zwischenpunkte, welche von dem Roboter passiert werden müssen, weiter verfeinert werden. Sehr viele Zwischenpunkte beschreiben die Bahn detailliert ohne viele Freiheiten für den exakten Verlauf zu lassen. Eine hohe Anzahl von Zwischenpunkten beansprucht aber auch mehr Rechenzeit und kann den Roboter im Extremfall verlangsamen, da er jeden Punkt mit einer definierbaren Mindestgenauigkeit

erreichen muss. Große Abstände zwischen den Punkten reduzieren diesen Nachteil, beschreiben die Bahn aber auch weniger genau.

Für die Berechnung der Bahn wird ein maximaler Punktabstand angegeben, der nicht überschritten werden darf. Darüber hinaus werden alle Bahnpunkte annähernd in gleichmäßigem Abstand bestimmt, sodass auch die letzten Punkte am Bahnende nicht zu nah beieinander liegen. Dadurch ist eine gleichmäßigere Geschwindigkeit des Roboters erreichbar.

Abbildung 7.20 zeigt zur Veranschaulichung des Bahnverlaufs bei verschiedenen Punktabständen beispielhaft zwei Roboterbahnen. Für beide Bahnen wurde der gleiche Start- und Zielpunkt gewählt. Die Potenzialfelder sind ebenfalls identisch. Der maximale Punktabstand wurde variiert. Der schwarz dargestellte Punkt ist der erste Bahnpunkt. Im Anschluss verläuft die Bahn durch schrittweise hellere Punkte, bis zum weißen Ziel. In Abbildung 7.20a ist die Bahn mit einem Meter maximalem Punktabstand dargestellt. Dagegen zeigt Abbildung 7.20b eine Bahn mit 20 Zentimetern maximalem Punktabstand. Durch die zahlreichen Zwischenpunkte ist diese Bahn wesentlich genauer definiert, sodass weniger Freiheiten bei der Interpretation durch die Robotersteuerung verbleiben. Ein anderes Extrembeispiel ist eine Bahn ohne Zwischenpunkte, mit undefiniertem Verlauf bis zum Zielpunkt. Praxistaugliche maximale Punktabstände werden heuristisch bestimmt.

Der Algorithmus zur Bahnbestimmung darf nicht an eventuell auftretenden lokalen Minima scheitern. Deshalb ist ein klassisches Gradientenabstiegsverfahren nicht verwendbar. Stattdessen wird das geringste Potenzial radial in dem definierten Abstand zum jeweiligen Punkt gesucht. Somit sind lokale Minima bei ausreichend großer Schrittweite kein Problem. Ein gleichmäßiger Punktabstand, auch für das letzte Intervall, wird mithilfe eines Optimierungsverfahrens erreicht, das den Punktabstand variiert. Eine geschlossene Lösung ist nicht möglich, da sich der Verlauf und somit auch die Länge der Bahn mit dem Punktabstand ändern.

Der Sauggreifer ist bei allen Bewegungen senkrecht nach unten gerichtet, da keine anderen Orientierungen für den Transport des Pakets oder für den Einsatz der Tiefenkamera sinnvoll sind.

7.5.2. Ansteuerung des Roboters

Nachdem die Bahn des Roboters geplant wurde, wird diese mithilfe eines XML(eXtensible Markup Language)-Protokolls über eine Ethernet-Verbindung an die Robotersteuerung übermittelt. Dort sorgt ein in der Kuka Robot Language (KRL) geschriebenes Programm für die korrekte Ansteuerung der Punkte. Der grundsätzliche Aufbau ist in Abbildung 7.21 dargestellt.

XML-Kommunikation

Für den Austausch zwischen Roboter und Bahnplanung werden Daten via Ethernet mithilfe von XML ausgetauscht. Das XML-Protokoll wurde von der PG definiert und unterscheidet zwischen gesendeten und empfangenen Telegrammen. Im Folgenden wird zunächst das Protokoll selbst erläutert. Danach wird beschrieben, wie das XML-Protokoll in der Robotersteuerung eingerichtet bzw. definiert werden kann.

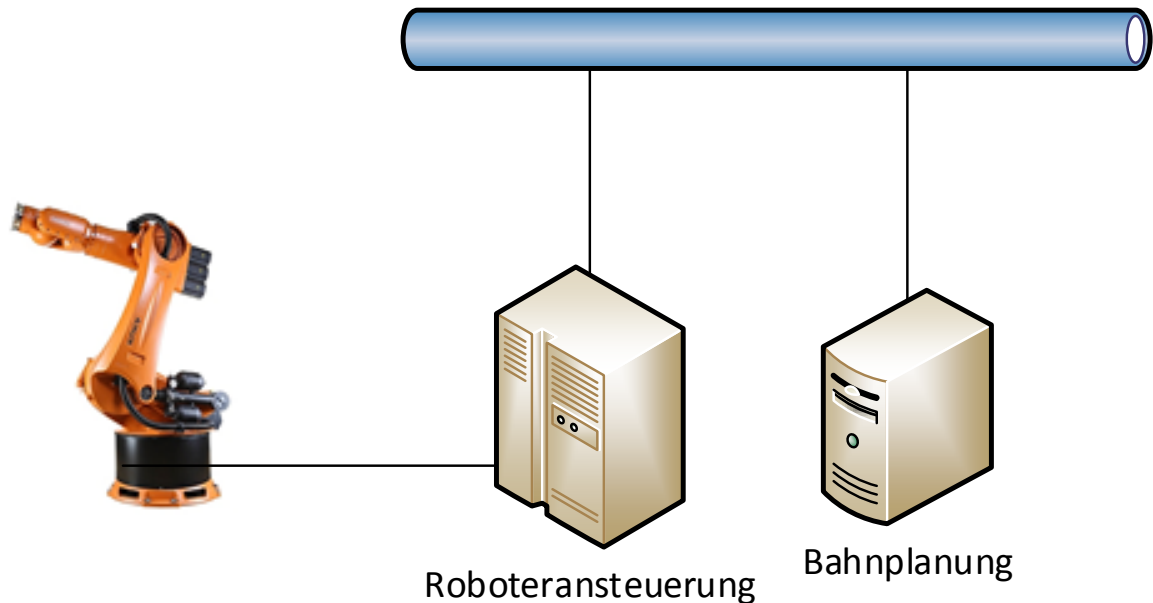


Abbildung 7.21.: Aufbau der Robotersteuerung. Der Computer, der die Kameras auswertet und die Bahn für den Roboter berechnet übermittelt die berechneten Daten über eine Ethernet-Verbindung zur Roboteransteuerung. Diese interpretiert die Daten und fährt den Roboter an die entsprechende Position.

Aufbau des XML-Protokolls

Die Telegramme, die von der Bahnplanung zum Roboter übermittelt werden, sind stark hierarchisch aufgebaut. Das Hauptelement heißt *ExternalData*. Darunter sind die Position („Position“), der Zustand des Sauggreifers („Tool“) und die Art der Bewegung („Movement“) angeordnet. Die Position besteht aus den anzufahrenden *X*-, *Y*- und *Z*-Koordinaten in Millimeter sowie der zugehörigen Rotation des Sauggreifers als *A*-, *B*- und *C*-Winkel in Grad angegeben. Alle Daten werden im Weltkoordinatensystem des Roboters angegeben, als separates Element repräsentiert und als Fließkommazahl übermittelt. Der Zustand des Sauggreifers wird als Zahl angegeben, da der Zustand des Sauggreifers auch in der KRL als Zahl übergeben werden muss. Die Zustände sind aus (0), Luft ansaugen (1) und Luft ausstoßen (2). Zur besseren Interpretation wird dieser Wert nochmal in ein Element mit dem Namen „G“ für *Gripper* als Unterkategorie geschachtelt. Die Informationen zur Art der Bewegung enthalten zwei Boolesche- und einen Fließkomma-Wert: ob die Bewegung Linear erfolgen soll (L), ob die Bewegung mit oder ohne Toleranz durchgeführt werden soll (T) und wie schnell die Bewegung ausgeführt werden soll (V). Der letzte Wert wird in Metern pro Sekunde ($\frac{m}{s}$) angegeben.

Um zum Beispiel den Punkt (−1140, 1459, 1202) mit den Winkeln (−82, 0, 180) mit aktiviertem Sauggreifer in einer linearen Bewegung ohne Toleranz und mit einer Geschwindigkeit von $1 \frac{m}{s}$ anzufahren muss das folgende Telegramm übermittelt werden:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ExternalData>
3   <Position>
```

```

4     <XPos>-1140</XPos>
5     <YPos>1459</YPos>
6     <ZPos>1202</ZPos>
7     <APos>-82</APos>
8     <BPos>0</BPos>
9     <CPos>180</CPos>
10    </Position>
11    <Tool>
12      <G>1</G>
13    </Tool>
14    <Movement>
15      <L>true</L>
16      <T>false</T>
17      <V>1</V>
18    </Movement>
19  </ExternalData>

```

Da die Interpretation auf dem Rechner der Bahnplanung einfacher ist, sind die Telegramme, die von dem Roboter zur Bahnplanung übermittelt werden, weniger strukturiert. Es gibt nur ein relevantes Element, das in einem „Robot“- und einem „Data“-Element gekapselt ist. Dabei handelt es sich um die aktuelle Roboterposition („ActPos“), als X -, Y -, und Z -Koordinaten in Millimeter, sowie Drehung in Grad. Zusätzlich wird noch ein Boolescher Wert übermittelt, der angibt, ob eine neue Position durch die Bahnplanung übermittelt werden soll oder nicht. Die Werte sind jedoch nicht als eigene Elemente ausgeführt, sondern lediglich als Attribute. Das zweite Element enthält die aktuelle Systemzeit der Robotersteuerung und wird zur Synchronisierung benötigt (siehe Abschnitt 7.5.3).

Wenn der Roboter den Punkt $(-1140, 1459, 1202)$ mit den Winkeln $(-82, 0, 180)$ erreicht hat und eine neue Position anfordert, so wird das folgende Telegramm übermittelt:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <Robot>
3   <Data>
4     <ActPos X="-1140" Y="1459" Z="1202" A="-82" B="0" C="180" NewPose="true" />
5   </Data>
6 </Robot>

```

XML-Protokoll in der Robotersteuerung definieren

Um Telegramme im XML-Format empfangen und in einem KRL-Programm verarbeiten zu können, muss in der Datei `XmlApiConfig.xml` ein neuer Kanal definiert werden. Jedem Kanal ist dabei ein Name zugeordnet. Mithilfe dieses Namens können die Daten im KRL-Programm empfangen werden. Um die Daten interpretieren zu können, muss zu jedem Kanal mindestens eine weitere XML-Datei generiert werden, die das XML-Protokoll beschreibt. Es werden der Aufbau der Telegramme und die verwendeten Datentypen definiert. Zusätzlich kann eine weitere Datei angelegt werden, die ein abweichendes Protokoll zum Versenden von Telegrammen definiert.

Im KRL-Programm muss zum Empfang ein Sensor mit dem Namen des Kanals angegeben werden. Dieser wird im Anschluss mithilfe des Befehls „EKX_Open“ geöffnet. Von nun an können Daten mit den Befehlen „EKX_Read“ und „EKX_Write“ gelesen beziehungsweise geschrieben werden.

KRL-Programmaufbau

Das KRL-Programm, welches den Bewegungsablauf des Roboters steuert, initialisiert zuerst die XML-Kommunikationsschnittstelle, welche im Abschnitt 7.5.2 detailliert beschrieben wird. Anschließend wird eine zuvor definierte „Home-Position“ angesteuert, welche als initiale Startposition der Roboterbewegung interpretiert werden kann. Erst jetzt betritt der Programmablauf eine Endlosschleife, in der die Kommunikation zwischen dem KRL-Programm und dem Hauptprogramm stattfindet.

In diesem Ablaufbereich wird dem Hauptprogramm in jeder Schleifeniteration die aktuelle TCP-Pose gesendet. Die Methode, die zum Senden der TCP-Pose konstruiert wurde, besitzt einen Booleschen Übergabeparameter mit dem die Möglichkeit besteht dem Hauptprogramm mitzuteilen, ob das KRL-Programm eine neue, vom Roboter anzusteuernde Pose benötigt.

Anschließend wird auf eine Antwort, welche die anzusteuernde Pose enthält, gewartet. Sobald diese Antwort eingetroffen ist, wird die enthaltene Pose dem Roboter gesendet. Falls die Antwort eine Pose enthält, die für die anzusteuernenden X -, Y - und Z -Koordinaten den Wert 0 besitzt, ist dies als eine Aufforderung zur Beendigung des KRL-Programms zu interpretieren.

Zusätzlich zum iterativen Austausch von anzusteuernden Posen und aktuellen TCP-Posen existiert ein interruptgesteuertes Senden der aktuellen TCP-Pose an das Hauptprogramm. Bei diesem Sendevorgang wird dem Hauptprogramm mit Hilfe des Booleschen Übergabeparameters jedoch mitgeteilt, dass keine neue anzusteuernde Pose benötigt wird. Der Aufruf der Interruptbehandlungsroutine wird durch einen Timer gesteuert, der alle $500ms$ ausgelöst wird.

Der soeben dargestellte Ablauf des KRL-Programms wurde in Abbildung 7.22 illustriert.

7.5.3. Bestimmung der TCP-Position

Sowohl für die weitere Bewegungsplanung als auch zur Auswertung der von der Tiefenkamera erhaltenen Bilder, muss die Position des TCP zu jedem Zeitpunkt bestimmt werden. Hierzu wird von Seiten des Roboters jede Pose mit dem internen Zeitpunkt, an dem der TCP diese erreicht hat, versendet. Da die interne Zeit allerdings nicht mit der Zeit des externen Systems übereinstimmt, muss eine Synchronisation erfolgen, mithilfe dieser der Offset zwischen den beiden Systemen bestimmt wird. Ist der Offset bestimmt, so kann die vom Roboter gesendete Zeit in einen äquivalenten Zeitpunkt der externen Steuerung transformiert und damit abgespeichert werden. Die zur Synchronisation verwendbaren Verfahren werden zuerst allgemein beschrieben. Im Anschluss wird das tatsächlich verwendete Verfahren erörtert.

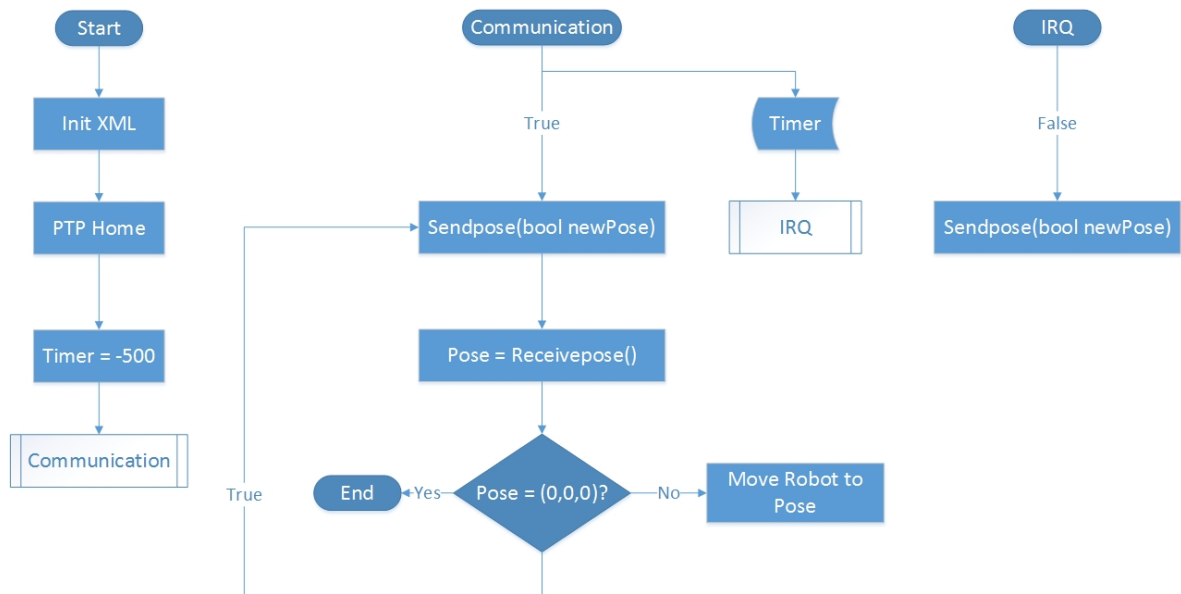


Abbildung 7.22.: Aufbau des KRL-Programms. Sequentielles Senden und Empfangen von Posen. Zusätzliches, interruptgesteuertes Senden der aktuellen TCP-Pose zur weiteren Verarbeitung im Hauptprogramm

Allgemeine Zeitsynchronisationsverfahren

Um die Daten der verschiedenen Sensoren und des Roboterarms vereinigen zu können, müssen diese Daten zeitsynchronisiert sein. Erfolgt diese Zeitsynchronisation nicht, so könnte aufgrund zeitlicher Divergenzen der Entladevorgang des Paketes ggf. nicht fehlerfrei ablaufen. Um diesen Zustand zu vermeiden, müssen die Uhren der verschiedenen Systeme (Robotersteuerung und System mit der Anwendung) entweder die exakt gleiche Zeit zeigen oder wissen, in welchem Verhältnis sich die Zeiten der Uhren zueinander befinden. Im Hinblick auf eine Fusion verschiedener Daten widmet sich dieser Abschnitt einigen zur Zeitsynchronisation bereits entwickelten und etablierten Protokollen.

Es wird im Allgemeinen zwischen interner und externer Synchronisation unterschieden [53]. Zeigen die Uhren aller teilnehmenden Systeme die Realzeit (Atomuhrzeit), die sog. „Universal Coordinated Time“, an, so handelt es sich um eine externe Zeitsynchronisation. Stellt das verteilte System allerdings ein geschlossenes System dar, so genügt die interne Zeitsynchronisation zwischen den partizipierenden Teilsystemen [53].

In traditionellen verteilten Systemen kommen zur Synchronisation von Rechneruhren verschiedene Methoden zum Einsatz. Die bekannteste ist dabei das Network Time Protocol (NTP). Dieses hat sich vor allem im Bereich des Internet durchgesetzt, da es eine hohe Robustheit und Skalierbarkeit aufweist. Dabei wird eine Baumstruktur von Netzknoten aufgebaut. Zur Synchronisation schickt dann ein Server mit einer sehr genauen Uhr, wie beispielsweise einer Atomuhr, periodisch die aktuelle Zeit an den Client [53]. Dieser kann anschließend die erhaltenen Zeitstempel weiter versenden. Da allerdings aufgrund der Latenz bei der Übertragung im Netzwerk der Zeitstempel bereits veraltet ist, wird diese Verzögerungszeit beim Empfänger mit einer geschätzten halben „round-trip-time“, der Zeit vom Sender zum Empfänger und wieder zurück, ergänzt [53].

Eine zweite mögliche Methode wäre die Synchronisation durch eine Approximation der Realzeit durch Nutzung global vorhandener Zeitinformationen, wie sie beispielsweise ein GPS-Empfänger liefert. Die Genauigkeit beträgt dabei etwa 200ns. Die Präzision, der auf diese Weise ermittelten Zeit, wäre für den Anwendungsfall der Paketentnahme ausreichend, da Positionsdaten mit Zeitstempeln verknüpft werden, welche lediglich bis zur Genauigkeit von Millisekunden reichen. Das Problem bei dieser Methode ist allerdings, dass der Empfänger eine längere Zeit zur Initialisierung benötigt und zudem Sichtkontakt zu Satelliten erfordert, welcher beispielsweise in einer Lagerhalle nicht gegeben ist [53].

Neben dieser Methode stellt auch das Prinzip der logischen Uhren, mit denen eine kausaltreue Beobachtung realisierbar ist, eine Möglichkeit zur Zeitsynchronisation dar. Mit diesen logischen Uhren lassen sich allerdings nur interne Ereignisse (zum Beispiel Nachrichtenversand und -empfang) in eine Ordnung bringen. Da keine physikalische Zeit gemessen wird, sondern lediglich zu jedem Element ein monoton steigender Wert als Stempel hinzugefügt wird [53]. Die Methode ist demnach nicht anwendbar, sobald eine Zuordnung eines Sensorwertes zu einem genauen Zeitpunkt erforderlich ist.

Die „Packet Stream Synchronization“ und die „Reference Broadcast Synchronization“ sind zwei Synchronisationsmethoden, die vor allem bei Sensornetzwerken und somit zur internen Synchronisation zum Einsatz kommen. Die „Packet Stream Synchronization“ verfolgt dabei einen ähnlichen Ansatz wie das Network Time Protocol, bei dem das Problem besteht, dass die Verzögerungszeit nicht bekannt ist. Um diese zu messen, müssten die Uhren synchron sein. Da sie dies allerdings nicht sind, geht diese Methode davon aus, dass die Verzögerung durch das Netzwerk gleich null ist. Nun wird versucht, eine untere Schranke für die Realzeit zu approximieren. Um die minimale Verzögerungszeit durch Annäherung einzustellen, dient ein aktuellerer ankommender Zeitstempel als neuer Startwert der Uhr des Empfängers. Ist nun die minimale Latenz zwischen Sender und allen Empfängern gleich, wäre die Präzision der internen Synchronisation der Teilnehmer untereinander sehr genau [53].

Bei der „Reference Broadcast Synchronization“ wird hingegen versucht, das Problem des Weges vom Sender zum Empfänger und somit der Verzögerung zu umgehen, indem der Weg durch einen Broadcast auf physischer Ebene verkürzt wird. Dies geschieht dadurch, dass bei einem Broadcast die Sendedauer, also das Konstruieren der Nachricht und Vorbereiten dieser zum Senden, sowie die Zugriffszeit auf das Netz identisch sind. Da die tatsächliche Übertragung zum Empfänger meist im Nanosekunden-Bereich liegt, kann diese vernachlässigt werden, sodass lediglich die Empfangsdauer bei den Empfängern variiert. Letztlich wird, um auch diese nicht-deterministische Komponente noch zu minimieren, bei Eintreffen des Nachrichtenansfangs ein Interrupt ausgelöst, der die aktuelle lokale Systemzeit ausliest und somit die Verzögerung des Empfangens im Anschluss mit einberechnet werden kann [53].

Unabhängig von diesen Methoden ist es im Anwendungsfall der Projektgruppe erforderlich, die aktuelle Zeit der Robotersteuerung auszulesen (s. Abschnitt 7.5.3). Dies ist beispielsweise mittels des von KUKA entwickelten Robot Sensor Interface (RSI) möglich. Das RSI stellt eine Zusatzsoftware dar, welche die Konfiguration von echtzeitfähigem Datenaustausch zwischen dem Roboter und einem externen System, im Fall der Projektgruppe das externe datenvereinigende System, ermöglicht [54]. Hierbei erfolgt die Signalverarbeitung zyklisch und wird alle 12ms parallel zur Programmausführung ausgewertet. Die Kommunikation mit dem externen System erfolgt dabei über einen Feldbus (Interbus, Profibus, DeviceNet oder ProfiNet) oder über das Ethernet. Bei Letzterem kommuniziert der Roboter mit dem externen

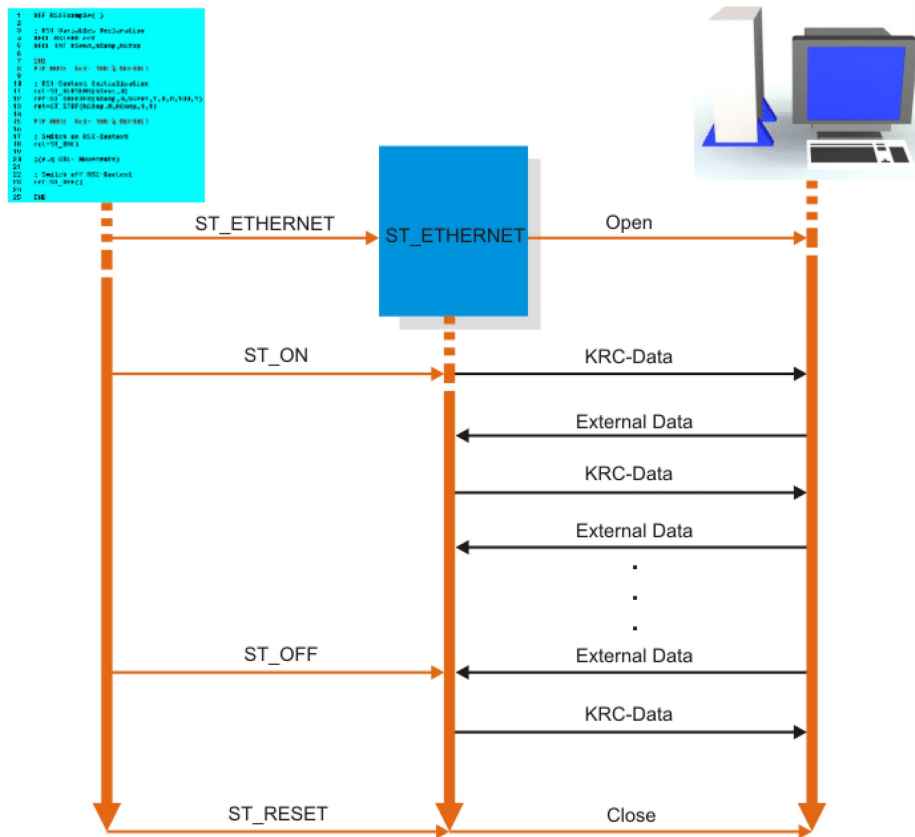


Abbildung 7.23.: RSI-Datenaustausch via Ethernet [54]

System mittels der echtzeitfähigen Netzwerkverbindung unter Zuhilfenahme von XML Strings [54]. Die detaillierte Vorgehensweise des Datenaustausches via Ethernet ist in Abbildung 7.23 zu sehen. Hierbei bildet die linke Seite die Robotersteuerung und die rechte das externe System, mit welchem die Daten ausgetauscht werden sollen.

Aufgrund eines verzögerten und langwierigen Bestellprozesses dieser Zusatzsoftware konnte es allerdings letztlich nicht bei der Synchronisation des externen Systems mit dem Roboter eingesetzt werden.

Verwendete Zeitsynchronisation

Da die soeben beschriebenen Verfahren im Anwendungsfall der Projektgruppe nicht verwendet werden konnten, wurde eine einfache Synchronisationsroutine selbst implementiert. Hierbei werden lediglich Pakete über die Ethernet-Verbindung übertragen und die Dauer der Übertragung gemessen.

Dies ist möglich, weil jede weitere Last auf der Ethernet-Verbindung ausgeschlossen werden kann, da lediglich der Roboter und die externe Steuerung angeschlossen sind. Daraus folgt, dass zu Beginn der Kommunikation lediglich die XML-Pakete der Synchronisation als Last auf der Ethernet-Verbindung liegen. Daher kann die Zeit der Übertragung eines Paketes nur minimal

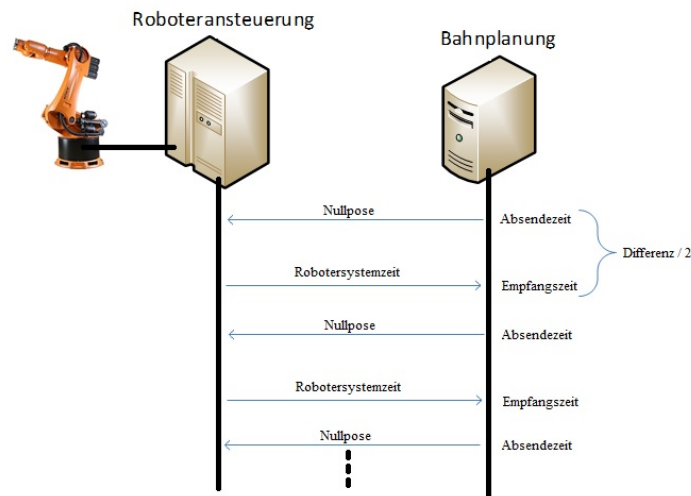


Abbildung 7.24.: Implementierte Zeitsynchronisation

schwanken. Indem nun nicht nur ein Paket zur Zeitsynchronisation versendet wird, sondern zwölf Pakete nacheinander, lässt sich ein sehr genauer Mittelwert aus den Übertragungszeiten berechnen.

Zu Beginn der Kommunikation zwischen externem System und dem Roboter wird die Synchronisationsroutine durch das externe System aufgerufen (Abbildung 7.24). Hier wird ein Synchronisationspaket als erweiterte Nullpose erstellt und vor der Absendung die Absendezeit gespeichert. Empfängt der Roboter das Paket, so wird es dort als Synchronisationspaket erkannt und mit der aktuellen Systemzeit des Roboters zurückgesendet. Bei Erhalt des Paketes im externen System wird erneut die Empfangszeit gespeichert. Nun kann die Zeitdifferenz aus den beiden gespeicherten Zeitwerten berechnet und halbiert werden. So wird die Zeit bestimmt, welche ein Paket für einen Weg, beispielsweise von der externen Steuerung an den Roboter, benötigt. Alle berechneten Zeitdifferenzen der zwölf Synchronisationspaket werden aufsummiert. Sobald das letzte Paket der Synchronisationsroutine erhalten wurde, wird die Synchronisation beendet. Um Ausreißer weniger stark zu gewichten, wird sowohl die maximale als auch die minimale Zeitdifferenz aus der Summe der Zeitdifferenzen entnommen. Aufgrund der verbleibenden zehn Zeitdifferenzen in der Summe wird diese durch zehn geteilt und so der Mittelwert errechnet. Dieser wird im späteren Betrieb verwendet, um die Zeiten der erhaltenen Pakete anzupassen und mit einer korrekten Zeit ohne den Offset der Übertragungsdauer abzuspeichern.

7.6. Zusammenwirken der Module zur Paketentnahme

In den Abschnitten 7.2 bis 7.5 wurde jeweils ein Modul beschrieben, das einen für die Entnahme von Paketen aus fahrerlosen Transportfahrzeugen wichtigen Aspekt realisiert. Dieser Abschnitt beschreibt nun, wie das Gesamtsystem auf den einzelnen Modulen aufbauend realisiert wurde. Die Module agieren bis auf wenige Kommunikationsschnittstellen unabhängig voneinander. Daher wurde bei der Implementierung für jedes Modul ein eigener Thread vorgesehen, so dass

die in Form mehrerer Rechenkerne zur Verfügung stehende Leistung durch parallele Berechnungen besser ausgenutzt werden kann. Für die Erzeugung der Threads und Bereitstellung von Kommunikationsschnittstellen wurde auf das *Qt Framework*¹⁵ zurückgegriffen, um den Implementierungsaufwand bezüglich Threadsynchronisation möglichst gering zu halten.

Die Kommunikation zwischen den Modulen und damit zwischen verschiedenen Threads kann mit Qt recht einfach über den *Signals & Slots*-Mechanismus geschehen [55].

Jedes Modul kann dabei als *Signal* bezeichnete Ereignisse definieren und eine Liste von formalen Parametern damit verknüpfen. Beispielsweise definiert das Modul, das für die Erkennung des Fahrzeugs auf CCD-Kamerabildern zuständig ist, ein Signal *vehicleDetected*, das als formale Parameter die ermittelten Weltkoordinaten und einen Zeitstempel für die Erkennung besitzt. Das Modul kann nun jederzeit die definierten Signale *emittieren* und dabei für die formalen Parameter Werte übermitteln. Im Beispiel wird das *vehicleDetected*-Signal regelmäßig nach einer erfolgreichen Erkennung gesendet und dabei die erkannte Position mit dem aktuellen Zeitstempel übermittelt.

Das Gegenstück zu Signalen sind *Slots*. Slots sind speziell als solche gekennzeichnete in C++ geschriebene Methoden. Beispielsweise besitzt das Tracking-Modul einen Slot, der als Parameter die aktuell anhand der CCD-Bilder ermittelten Weltkoordinaten sowie den dazugehörigen Zeitstempel verarbeitet, so dass diese neue Position durch den Condensation-Algorithmus im Tracking-Modul berücksichtigt werden kann.

Qt erlaubt nun das Verknüpfen von Signalen mit Slots, deren formale Parameter typgleich sind. Danach wird durch das Emittieren eines Signals jeder Slot, der mit diesem Signal verknüpft wurde, mit den bei der Emittierung übergebenen Parametern ausgeführt. Erfolgt die Verknüpfung zwischen zwei Qt-Objekten, die nicht dem gleichen Thread zugeordnet sind, so wird die Ausführung des Slots standardmäßig in einer Warteschlange zwischengespeichert und erst bei der Rückkehr des „empfangenden“ Threads in den *Eventloop* ausgeführt (Queued Connection) [56]. Sämtliche Kommunikation zwischen den Modulen erfolgt über diesen Mechanismus.

¹⁵<http://qt-project.org>

8. Evaluation

In diesem Kapitel werden die Evaluationsergebnisse der verschiedenen Module des Systems (siehe Kapitel 7) vorgestellt. Dazu werden in den folgenden Unterkapiteln die einzelnen, in der Projektgruppe entworfenen Module untersucht. Abschnitt 8.1 beschreibt vorab zuerst einmal den allgemeinen Versuchsaufbau, zu der die generelle Umgebung der Tests sowie die zum Einsatz kommenden Hardware- und Softwarekomponenten gehören.

Der Abschnitt 8.2 evaluiert anschließend die Erfolgsrate der Fahrzeugerkennung in den bei einer Fahrzeugdurchfahrt aufgenommenen CCD-Kamerabildern sowie die Abweichung der ermittelten Raumposition des Fahrzeugs von der tatsächlichen Fahrzeugposition. Danach wird für das Tiefenkamera-Modul (Abschnitt 8.3) die Genauigkeit und Robustheit der Paketerkennung unter dem Einfluss verschiedener Faktoren untersucht. Nach der Paketerkennung mittels CCD-Modul und Tiefenkamera-Modul werden die Genauigkeit der aktuellen Position und die Güte der Prädiktion für einen zukünftigen Zeitpunkt ausgewertet. Hierzu wird das implementierte Tracking-System des Tracking-Moduls evaluiert (Abschnitt 8.4). Da neben der Paketerkennung allerdings auch die Genauigkeit der Roboterposition sowie der Einfluss verschiedener Parameter auf die Bewegungsdauer des Roboters relevant sind, wird im Anschluss das Roboter-Modul evaluiert (Abschnitt 8.5). Zuletzt wird dann in Abschnitt 8.6 auf das Zusammenspiel aller zuvor erwähnten Module eingegangen.

Die in diesem Kapitel beschriebenen Evaluationstests wurden durchgeführt, um eine Aussage darüber treffen zu können, wie erfolgreich das entwickelte System die Ziele der Projektgruppe erreicht. Dazu wurden Testszenarien entwickelt, welche die einzelnen Komponenten sowie das Zusammenwirken des Systems sowohl in typischer Situation, als auch unter Extrembedingungen evaluieren. Auf diese Weise ist es möglich, Schwachstellen zu erkennen und das entwickelte System abschließend zu bewerten.

8.1. Allgemeiner Versuchsaufbau

Die Evaluation findet in der ZFT-Halle des Fraunhofer Instituts für Materialfluss und Logistik statt. Zum Entladen des Fahrzeugs wird der Industrieroboter *Kuka Roboter 125/3* verwendet, der sich aus Sicherheitsgründen in einer durch Glas abgetrennten Roboterzelle befindet (siehe Abbildung 8.1). Der Roboterarm ist mit einem Sauggreifer ausgestattet. Ein fahrerloses Transportfahrzeug (FTF) kann sich auf einer festgelegten Bahn durch die Roboterzelle bewegen. Zur Erkennung des FTFs ist eine CCD-Kamera so in einer Ecke der Roboterzelle befestigt, dass die gesamte Fahrzeugbahn von der Kamera erfasst werden kann. Die 3D-Kamera, eine *ASUS Xtion PRO* mit *PrimeSense 3D-Sensor*, ist am Roboterarm, ein Stück versetzt zum Sauggreifer angebracht (siehe Abbildung 8.2). Die in den Versuchen verwendeten Pakete haben, falls nicht anders erwähnt, eine Länge von 40cm, eine Breite von 25,5cm und eine Höhe von 30cm. Die Palette, auf die das Paket abgeladen werden soll, ist bei allen Versuchen leer.



Abbildung 8.1.: Die Roboterzelle in der ZFT-Halle. Die CCD-Kamera ist in der hinteren linken Ecke der Roboterzelle befestigt.

8.2. Fahrzeugerkennung und Positionsbestimmung

Dieser Abschnitt befasst sich mit der Evaluation jener Komponente des Systems, welche für die Erkennung des Fahrzeuges und Bestimmung dessen Position innerhalb der Roboterzelle zuständig ist.

8.2.1. Versuchsaufbau

Diese Komponente bietet die folgenden zwei Kernfunktionalitäten, welche unabhängig voneinander evaluiert werden müssen.

Fahrzeugerkennung

Die Fahrzeugerkennung erfolgt im Wesentlichen über eine CCD-Kamera (siehe Kapitel 7.2.1 und 7.2.2). Die Qualität der Erkennung hängt maßgeblich von den Lichtverhältnissen, welche in der Versuchshalle im Bereich der Roboterzelle herrschen, ab. Wie stark die Umgebungsbedingungen tatsächlich Einfluss auf die Erkennungsrate nehmen, soll mittels der Evaluation genauer untersucht werden.

Die Testszenarien werden jeweils durch Variation der folgenden Parameter gebildet:

- Licht in der Versuchshalle an/aus,
- Jalousien der großen Fensterfronten oben/unten,
- Kiste falsch auf dem Fahrzeug platziert (Jalousien unten, Licht an),
- Fahrzeug von Roboterarm überdeckt (Jalousien unten, Licht an).

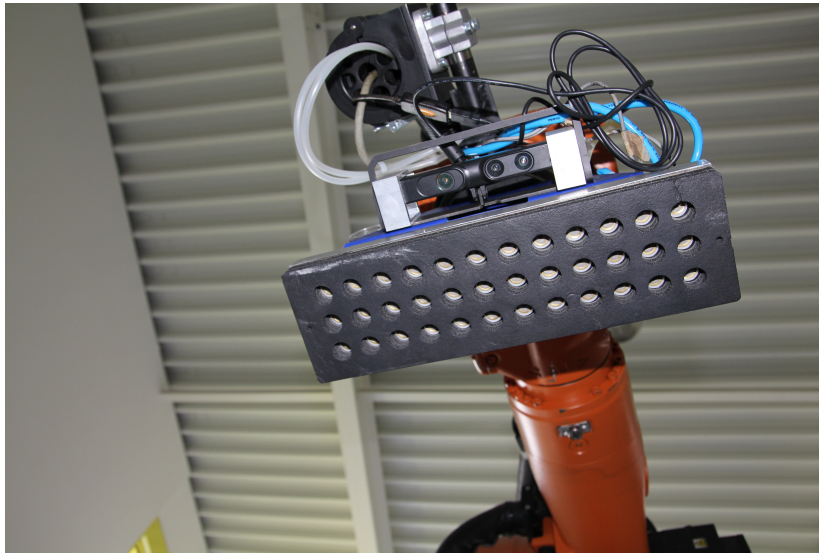


Abbildung 8.2.: Position der 3D-Kamera am Roboterarm

Die beiden zuletzt genannten Testszenarien werden durch die in Abbildung 8.3 dargestellten Fotos veranschaulicht.

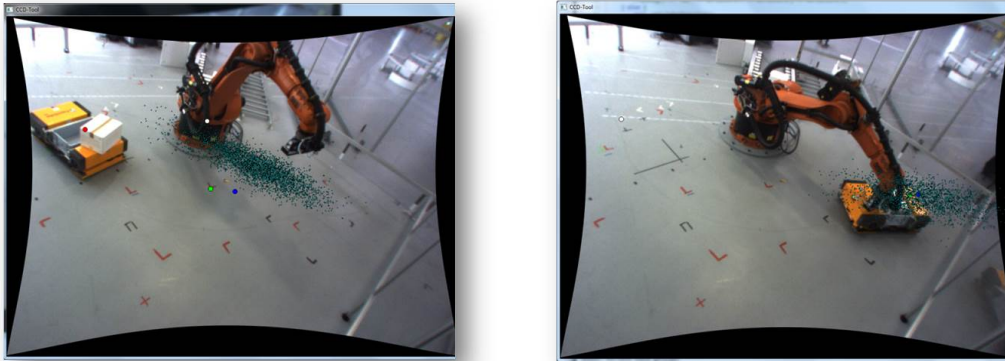
Positionsbestimmung

Nachdem mithilfe der vorgenannten Funktionalität die Position des Fahrzeuges im Bild der CCD-Kamera ermittelt wurde, müssen diese Koordinaten in das 3D-Koordinatensystem überführt werden. Wie dies technisch erfolgt, wurde in Kapitel 7.2.3 erläutert. Aufgabe der Evaluation ist in diesem Fall, die Genauigkeit dieser Transformation zu bestimmen. Da vorausgesetzt wird, dass das Fahrzeug im Bild bereits erkannt wurde, ist eine Variation der Umgebungsbedingungen in diesem Fall nicht erforderlich. Zur Ermittlung der Genauigkeit wird die Position des Fahrzeuges zu unterschiedlichen Zeitpunkten vom System ermittelt. Anschließend erfolgt eine manuelle Messung der Abweichung. Weitere Details hierzu werden im folgenden Abschnitt erläutert.

8.2.2. Evaluationskriterien

Um sowohl die Qualität der Erkennung als auch die Genauigkeit der Positionsbestimmung quantitativ bewerten zu können, müssen objektive Bewertungskriterien herangezogen werden. Im Falle der Fahrzeugerkennung wurden je drei Messungen mit einer Länge von 10 Sekunden und 30 Sekunden, beginnend bei Einfahrt des Fahrzeuges in den Roboterkäfig, durchgeführt. Hierbei wurde die Anzahl der verarbeiteten Frames $n_{erkannt}$, in denen das Fahrzeug erfolgreich erkannt wurde, gezählt und anschließend deren Anteil S_n an der Gesamtanzahl aller verarbeiteten Frames n_{gesamt} ermittelt:

$$S_n = \frac{n_{erkannt}}{n_{gesamt}}. \quad (8.1)$$



- (a) Die Kiste ist auf dem Fahrzeug falsch platziert und überdeckt teilweise eine der gelben Flächen. (b) Der Roboterarm verdeckt das Fahrzeug während der Durchfahrt zwischenzeitlich teilweise.

Abbildung 8.3.: Darstellung von zwei Testszenarien für die Evaluation der Fahrzeugerkennung

Um die Genauigkeit der Position im Roboter-Koordinatensystem zu ermitteln, wurde das FTF zunächst an drei verschiedene Positionen bewegt und mithilfe der Fahrzeugerkennung lokalisiert. Das Licht in der Halle war hierbei eingeschaltet und die Jalousien heruntergefahren. Anschließend wurde die ausgegebene Position manuell mit dem Roboterarm angefahren. Mittels einer Messung des Abstandes zwischen FTF-Mitte und dem darüber befindlichen Roboterarm wurde anschließend die Abweichung der ausgegebenen Position von den tatsächlichen Werten ermittelt.

8.2.3. Versuchsergebnisse und -bewertung

Die Ergebnisse der Messungen unter veränderten Umgebungsbedingungen werden in Abbildung 8.4 dargestellt. Es ist intuitiv verständlich, dass die Erkennungsrate bei heruntergelassenen Jalousien und gleichzeitigem Abschalten der Beleuchtung am stärksten abnimmt. Im vorliegenden Fall hat das Hochfahren der Jalousien keine nennenswerte Verschlechterung der Werte herbeigeführt. Die Erfahrung zeigt jedoch, dass bei starker Sonneneinstrahlung in die Halle das Bild der Kamera stark verfälscht wird und somit besonders im linken Bereich zu Problemen führt, wie im Folgenden erläutert wird. Die Messungen für die Evaluation wurden jedoch an einem bewölkten, winterlichen Tag durchgeführt, weshalb dieser Effekt ausblieb. Aufgrund der vorgenannten Gründe wurde die Entscheidung getroffen, im Kontext der Projektgruppe grundsätzlich mit eingeschalteter Hallenbeleuchtung und heruntergelassenen Jalousien zu arbeiten, um reproduzierbare Umgebungsbedingungen zu schaffen.

Abbildung 8.5 zeigt die Ergebnisse der beiden weiteren Messungen zur Erkennungsrate. Obwohl die falsch platzierte Kiste einen großen Teil der zur Erkennung benötigten gelben Fahrzeugfläche überdeckt, bricht die Erkennungsrate nur um circa zehn Prozent ein. Dies demonstriert, wie robust das entwickelte System zur Erkennung gegen Störeinflüsse ist. Auch die Überdeckung des Fahrzeuges mit dem Roboterarm führt lediglich zu einer ähnlich geringen Verschlechterung.

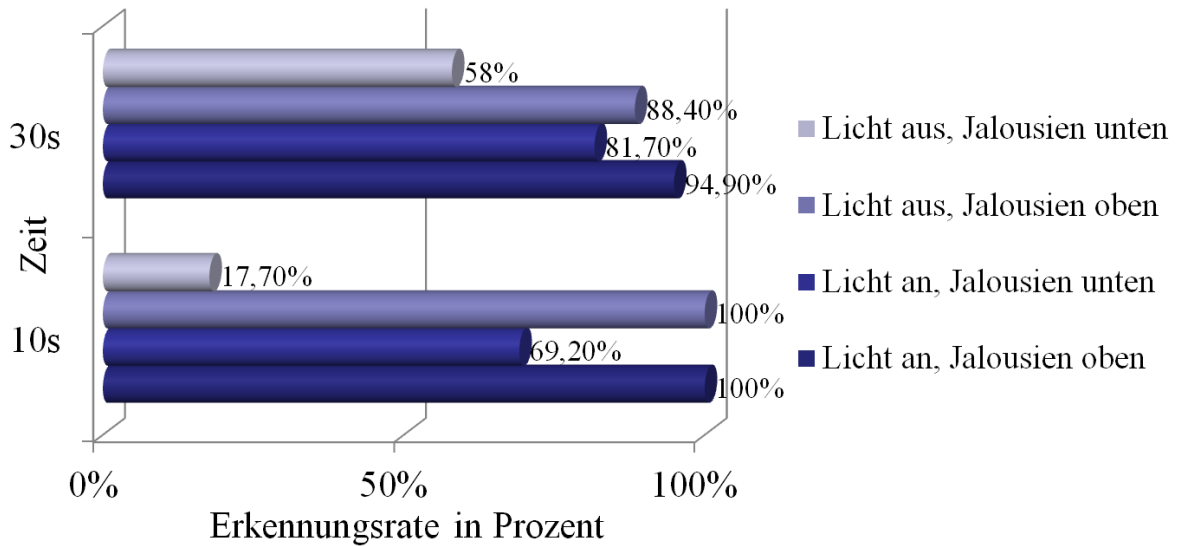


Abbildung 8.4.: Ergebnisse der Messungen unter veränderten Umgebungsbedingungen

In Abbildung 8.6 werden die Ergebnisse der Genauigkeitsmessung dargestellt. Es ist deutlich zu erkennen, dass die Abweichung nur im linken Bildbereich Werte erreicht, die eine unzureichende Genauigkeit der Ortung verursachen. Dies ist im Kontext des Systems jedoch nicht von Relevanz, da das Fahrzeug diesen Bereich erst am Ende durchfährt und das Paket zu dem Zeitpunkt, an dem die Abweichung auftritt, bereits entnommen wurde. Zudem kommen die wesentlichen Informationen zur Positionsbestimmung des Paketes zu diesem Zeitpunkt hauptsächlich vom 3D-Kameramodul.

Erklären lässt sich die relativ große Abweichung durch zwei gleichzeitig einwirkende Faktoren. Einerseits ist die Entzerrung des Kamerabildes im linken Bildbereich vermutlich nicht ausreichend exakt bzw. weniger exakt als im rechten und mittleren Bildbereich. Weiterhin treten im fraglichen Bereich teils starke Lichtreflexionen auf, die die orangenen Flächen des Fahrzeuges mitunter bis zur Unkenntlichkeit verschleiern und somit die Qualität der Positionsbestimmung maßgeblich beeinflussen.

8.3. Paketerkennung

Das folgende Kapitel beschäftigt sich mit der Evaluation des in Kapitel 7.3 vorgestellten Tiefenkameramoduls. Dabei soll insbesondere die Genauigkeit der detektierten Paketposition sowie die Robustheit getestet werden. Dafür wird zunächst in Kapitel 8.3.1 der Versuchsaufbau beschrieben. Des Weiteren werden in Kapitel 8.3.2 die Evaluationskriterien festgelegt, um im Folgenden in Kapitel 8.3.3 die Ergebnisse zu präsentieren und zu bewerten.

8.3.1. Versuchsaufbau

Getestet wird zum einen die Positionserkennung, d.h. insbesondere die Auswertung der Hand-Auge Kalibrierung, welche in Kapitel 7.3.1 beschrieben wird. Zum anderen wird die Robustheit

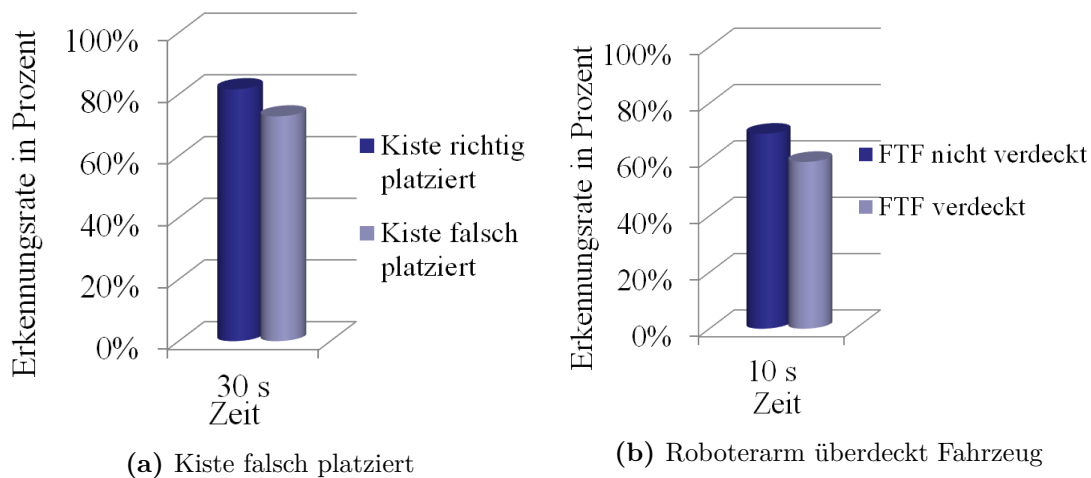


Abbildung 8.5.: Ergebnisse der Messungen bei falsch platzierter Kiste und überdeckendem Roboterarm

der Paketerkennung getestet und ermittelt, ob das Modul nur das Paket selbst oder auch andere Flächen als Paketflächen identifiziert. Dafür wird das Paket statisch an einige Position innerhalb des vom Roboter erreichbaren Bereiches gestellt und der Paketmittelpunkt manuell vermessen. Anschließend wird der Greifarm des Roboters manuell über das Paket bewegt und die errechneten Positionsdaten ausgewertet.

8.3.2. Evaluationskriterien

Betrachtet wird dann die Abweichung des errechneten Paketmittelpunktes mit dem tatsächlichen Mittelpunkt, der zuvor mit dem Messdorn des Roboters gemessen wurde. Es werde die Abweichungen in *cm* zum einen in der *X-/Y*-Ebene betrachtet und zum anderen auf der *Z*-Achse (im Roboterkoordinatensystem). Diese Unterscheidung wird getroffen, da ein Fehler in der errechneten *Z*-Koordinate weit aus gravierende Folgen haben kann, als in der *X-/Y*-Ebene. Die *Z*-Achse steht senkrecht auf dem Boden und gibt damit die Höhe des Paketes an. Ist die errechnete Höhe falsch, so kann im besten Falle nicht gegriffen werden. Im schlechtesten Falle zerstört der Sauggreifer das Paket, da, mangels Drucksensor, eine Kollision mit dem Paket nicht erkannt werden kann.

Als zweiter Punkt wird die False-Positive Rate der Erkennung ermittelt, d.h. wie oft wird eine Oberfläche fälschlich als Paket erkannt. Dies ist vor allem im Hinblick auf ein späteres Tracking relevant (siehe Kapitel 7.4). Da ein Fehlen von Daten des Tiefenmoduls in einem Frame durch das CCD-Modul ausgeglichen werden können, führt ein False-Negative beim Tracking zu geringeren Störungen als ein False-Positive. Diese können ggf. zu einem sprunghaften Verhalten der detektierten Paketposition führen.

8.3.3. Versuchsergebnisse und -bewertung

In allen durchgeführten Testszenarien wich die errechnete Paketposition auf der *X-/Y*-Ebene im Betrag maximal $1,5\text{cm}$ von dem tatsächlichen Paketmittelpunkt ab. Auf der *Z*-Achse

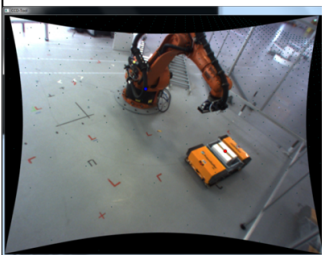
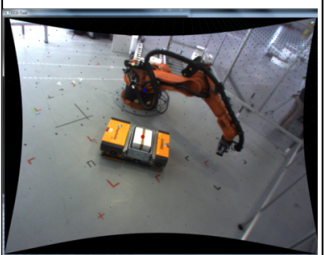
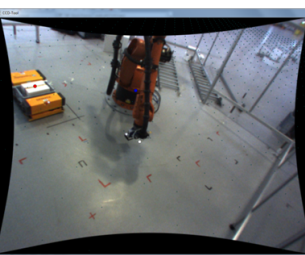
Position			
Abw. Breite in cm	3	0,6	0,3
Abw. Länge in cm	1,5	3,3	13,7

Abbildung 8.6.: Ergebnisse der Genauigkeitsmessung. Die erste Zeile stellt jeweils die Position des Fahrzeuges im Roboterkäfig dar, an der gemessen wurde, und die folgenden beiden Zeilen die jeweils maximale Abweichung in Breite und Länge.

betrug die Abweichung zwischen 0 und $+0.5\text{cm}$, d.h. dass die Paketoberfläche höher detektiert wurde, als sie tatsächlich war. Damit sind beide Abweichungen unkritisch. Bei den Ausmaßen der verwendeten Pakete ist eine Positionierung des Sauggreifers in einem Bereich von 1.5cm um den tatsächlichen Mittelpunkt ausreichend, um problemlos greifen zu können. Auch die Abweichung auf der Z -Achse ist kein Problem, da der Saugflaum am unteren Ende des Sauggreifers 2cm hoch ist, und somit ausreichend Spielraum lässt. Somit ist sicher gestellt, dass der Sauggreifer das Paket nicht zerdrückt.

Weiterhin konnte in keinem TestszENARIO ein False-Positive konstruiert werden. Allerdings muss mindestens die Hälfte des Paketes für die Kamera erkennbar sein, um eine erfolgreiche Identifizierung zu ermöglichen. Ist dies nicht der Fall, so liefert das Tiefenkameramodul keine Positionsdaten. Die Paketerkennung funktioniert von einer Höhe des Sauggreifers von 42cm über dem Paket bis 100cm über dem Paket. Dabei ist die obere Grenzen von 100cm eine in der Software gesetzte Grenze. Ohne diese ist das Paket auch aus der maximal für den Roboter erreichbaren Höhe noch erkennbar. Für das Tracking bedeutet dies, dass ggf. auf CCD-Daten zurückgegriffen werden muss, aber dass die Daten, die vom Tiefenkameramodul gesendet werden, sehr präzise sind. Weiterhin führt das Fehlen von False-Positives zu einem stabileren Tracking.

8.4. Prädiktion der Paketposition

Das folgende Kapitel beschäftigt sich mit der Evaluation des Trackingmoduls und dessen Funktionen. Dabei soll insbesondere die Genauigkeit der aktuellen Paketposition sowie die Güte der Prädiktion für die Zukunft getestet werden. Dafür wird zunächst in Kapitel 8.4.1 der Versuchsaufbau beschrieben. Des Weiteren werden in Kapitel 8.4.2 die Evaluationskriterien festgelegt, um im Folgenden in Kapitel 8.4.3 die Ergebnisse zu präsentieren und zu bewerten.

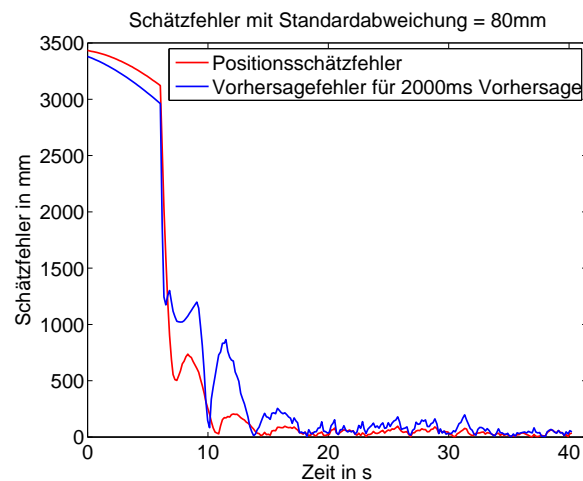


Abbildung 8.7.: Beispiel für die Entwicklung der Schätzfehler für die aktuelle Position und die vorhergesagte Position in Abhängigkeit von der verstrichenen Zeit

8.4.1. Versuchsaufbau

Getestet wird der implementierte Condensation-Algorithmus, der in den Kapiteln 6.2.3 und 7.4.2 näher beschrieben wurde. Dazu wird zunächst eine beliebige Fahrzeugbahn als Kurve modelliert, die im Anschluss mit einem Rauschen versehen wird. Als Beobachtungen erhält der Algorithmus die verrauschten Kurvenwerte als Eingabe. Der Algorithmus trackt die tatsächliche Position auf der modellierten Kurve und die Abweichung zwischen Positionsschätzung und der tatsächlichen Kurvenposition aus. Des Weiteren wird in jedem Schritt des Algorithmus eine Prädiktion für einen späteren Zeitpunkt geleistet. Auch dieser wird nach Ablauf des Zeitintervalls mit den tatsächlichen Kurvenwerten verglichen.

8.4.2. Evaluationskriterien

Evaluert wird der Fehler zwischen tatsächlichen Position auf der Kurve sowie den vom Algorithmus gelieferten Werten zur aktuellen und vorhergesagten Position. Im Zuge der Evaluation werden zwei Parameter verändert. Zum einen die Standardabweichung der Streuung, die auf die modellierte Kurve gelegt wird um einen Beobachtungsfehler zu simulieren. Der zweite variable Parameter ist die Zeitdifferenz zu dem Zeitpunkt in der Zukunft, zu dem der Algorithmus eine Prädiktion liefert. Im folgenden Unterkapitel werden die Ergebnisse der in Bezug auf die Änderung der beiden Parameter dargestellt und analysiert.

8.4.3. Versuchsergebnisse und -bewertung

Als Ergebnis jedes Testdurchlaufes ergibt sich der zeitliche Verlauf des Fehlers der Vorhersage und der aktuellen Schätzung. Abbildung 8.7 zeigt diesen Verlauf für eine Standardabweichung von 80mm und einer Vorhersagezeit von 2000ms. Der Verlauf der Fehler zeigt, dass diese immer kleiner werden, weshalb auf die Konvergenz des Tracking-Algorithmus geschlossen werden kann.

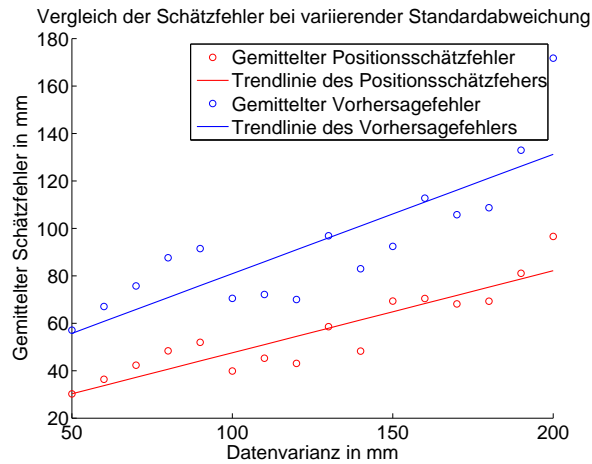


Abbildung 8.8.: Ergebnisse der Evaluation des Tracking-Moduls bei Variation der Standardabweichung zwischen 50mm und 200mm. Für den Vergleich wurde der mittlere Fehler der Vorhersage und der aktuellen Schätzung über einen festgelegten Zeitraum herangezogen.

Dieser Abschnitt stellt die Ergebnisse bei Variation der Standardabweichung vor. Um ein vergleichbares Ergebnis zu erhalten, wurde der Mittelwert über den Fehler in einem definierten Zeitraum ermittelt. Hier ergeben sich ebenfalls zwei Werte. Einer für die aktuelle Positionsschätzung und einer für die vorhergesagte Position. Getestet wurde mit Standardabweichungen zwischen 50mm und 200mm. Das Ergebnis zeigt Abbildung 8.8. Erwartungsgemäß können größere Fehler in der Prädiktion festgestellt werden, wenn die Standardabweichung größer wird. Auf das reale System übertragen bedeutet dies, dass das Tracking-Modul stark von der Güte der Module abhängig ist, die die Position des Fahrzeuges und Paketes bestimmen.

Bei der Variation der Vorhersagezeit wurde mit gleichbleibender Standardabweichung von 80mm getestet. Auch hier wurde der Mittelwert des Vorhersagefehlers über einen definierten Zeitraum gebildet. In diesem Fall ist der Fehler der aktuellen Schätzung uninteressant, da dieser maßgeblich von der Streuung der Eingabedaten abhängig ist, die hier gleichbleibend ist. Die Vorhersagezeit wurde bei den Tests zwischen 2000ms und 10000ms variiert. Abbildung 8.9 zeigt, dass die Vorhersage erwartungsgemäß einen höheren Fehler aufweist, je größer die Vorhersagezeit ist. Das hat zur Konsequenz, dass das Gesamtsystem der Erwartung nach akkurater arbeiten wird je kleiner die gewählten Vorhersagezeiten für die Robotervorsteuerung sind.

8.5. Robotersteuerung

Der folgende Abschnitt befasst sich mit der Evaluation des zur Verfügung gestellten Roboters, um unter anderem eine optimale Ansteuerung dessen durch seine Steuerungssoftware zu ermöglichen.

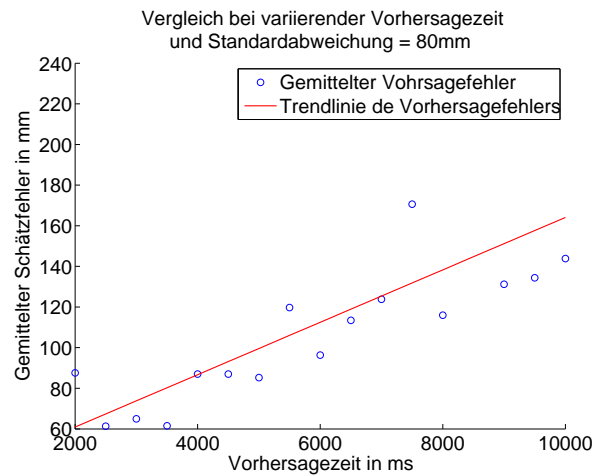


Abbildung 8.9.: Evaluationsergebnisse des Trackingsmoduls bei Variation der Vorhersagezeit zwischen 2000ms und 10000ms. Für den Vergleich wurde der mittlere Fehler der Vorhersage über einen festgelegten Zeitraum herangezogen.

8.5.1. Versuchsaufbau

Der bereits im allgemeinen Versuchsaufbau (siehe Kapitel 8.1) beschriebene Roboter wird mithilfe der entwickelten Software angesteuert. Hierbei wird in einem ersten Versuch lediglich ein Punkt an den Roboter gesendet. Auf diese Weise kann überprüft werden, wie exakt die von der Steuerungssoftware an den Roboter übertragenen Koordinaten von diesem angefahren werden können. In einem zweiten Versuch wird anschließend ein gesamter Pfad an den Roboter übergeben, den dieser in einer vorgegebenen Zeit zurücklegen soll. Dieser Versuch wird mit unterschiedlichen Einstellungen zweier Parameter durchgeführt, um die Auswirkungen auf die tatsächlich benötigte Zeit für den gleichen Pfad zu überprüfen.

8.5.2. Evaluationskriterien

Im ersten Versuch der Punktübertragung werden keinerlei Kriterien festgelegt. Hier erfolgt lediglich eine Überprüfung, ob und wie exakt die mittels der Steuerung gesendeten Koordinatenwerte auch durch den Roboter angefahren werden.

Im zweiten Versuch werden allerdings zwei Parameter verwendet. Dies sind zum Einen der Überschleifparameter \$APO.CDIS, welcher die Größe des Überschleifbereiches (in Millimetern) bei einer linearen Bewegung des Roboters beschreibt, und zum anderen der Advance-Parameter \$ADVANCE. Mittels des Advance-Parameters wird festgelegt, wie viele Programmzeilen der Vorlauf dem Hauptlauf maximal vorausseilen darf. Beide Parameter werden mit unterschiedlichen Werten belegt und anschließend die Auswirkungen betrachtet.

8.5.3. Versuchsergebnisse und -bewertung

Bei dem Versuch der Punktübertragung an den Roboter zeigen diverse durchgeführte Einstellungen der Koordinaten, dass die Punkte sehr genau an den Roboter übergeben werden. Dort

werden sie im Millimeterbereich bis auf zwei Stellen nach dem Komma gerundet. Allerdings muss beachtet werden, dass die tatsächliche Position des Robotergreifarms bzw. des Endeffektors durch die Positionsgenauigkeit des Roboters beschränkt ist und somit von den durch den Roboter angefahrenen und visualisierten Punktkoordinaten minimal abweichen kann. Diese Genauigkeit ist leider nicht in den zur Verfügung stehenden Datenblättern des Kuka-Roboters zu finden. Da die Abweichungen jedoch nicht im Zentimeterbereich liegen dürften, können sie für den in der Projektgruppe bearbeiteten Anwendungsfall vernachlässigt werden.

Der im zweiten Versuch zugrunde liegende Pfad, welchen der Roboter zurücklegen muss, ist drei Meter lang. Zudem wurde die Dauer der Bewegung auf 10 Sekunden eingestellt, innerhalb dieser der Roboter die übergebenen Zielkoordinaten erreicht haben soll. Entsprechend dieser Einstellung wird dem Roboter zusätzlich zu den anzufahrenden Punkten die Geschwindigkeit übergeben. Die Grundeinstellung der Parameter ist dabei folgende:

- $\$APO.CDIS = 200$
- $\$ADVANCE = 1$

Zuerst wird die Zeit gemessen, die der Roboter tatsächlich mit diesen Einstellungen der Parameter benötigt. Anschließend werden beide Parameter verändert und die Auswirkungen auf die Zeit und den Fluss der Bewegung betrachtet. Die Auswertung der Zeitmessung ist in Tabelle 8.1 aufgeführt.

	$\$ADVANCE = 1$	$\$ADVANCE = 2$
$\$APO.CDIS = 200$	10,1s	10,1s
$\$APO.CDIS = 0$	13,1s	13,1s
$\$APO.CDIS = 50$	10,78s	10,78s
$\$APO.CDIS = 500$	10,1s	10,05s

Tabelle 8.1.: Ergebnisse Evaluation Robotersteuerung

Hieraus wird ersichtlich, dass der Roboter tatsächlich bei Grundeinstellung der Parameter etwas länger für die Bewegung benötigt als von der Steuerungssoftware vorgegeben.

Wird nun das Überschleifen gänzlich deaktiviert (Zeile 2 in Tabelle 8.1), so benötigt der Roboter wesentlich länger für die gleiche Strecke. Dies liegt daran, dass er jeden Zwischenpunkt der Bewegung exakt anfahren muss. Hierzu bremst er ab, um anschließend nach Erreichen des Zwischenpunktes wieder zu beschleunigen. So entsteht eine sehr ungleichförmige Bewegung des Robotergreifarms. Wird das Überschleifen nicht ganz deaktiviert, sondern nur eingeschränkt (Zeile 3), so benötigt der Roboter zwar auch etwas länger als in der Grundeinstellung, allerdings kürzer, als wenn der Parameter auf Null gestellt wird. Die Bewegung ist allerdings bei dieser Einstellung ebenfalls noch nicht gleichförmig, da der Roboter die Geschwindigkeit in der Nähe der Zwischenpunkte immer noch etwas reduzieren muss. Auffällig ist, dass selbst bei einer wesentlich höheren Einstellung des Überschleifen-Parameters die Zeitspanne, anders als eigentlich erwartet, nicht weiter sinkt.

Anhand der Tabelle 8.1 lässt sich sehr gut erkennen, dass zwar die Einstellung des Überschleifen-Parameters Einfluss auf die Zeit der Bewegung hat, jedoch die Einstellung des Advance-Parameters fast keinerlei Unterschiede hervorruft.

8.6. Vollständiger Entladevorgang

Nachdem in den vorhergehenden Abschnitten die einzelnen Module des Systems unabhängig voneinander evaluiert wurden, wird in diesem Abschnitt abschließend ein Test des Gesamtsystems dokumentiert.

8.6.1. Versuchsaufbau

Bei der Evaluation der einzelnen Systemkomponenten wurde jeweils festgestellt, unter welchen Bedingungen diese optimal arbeiten. Die Evaluation des Gesamtsystems erfolgte unter idealen Umgebungsbedingungen. Diese beinhalten beispielsweise gute Lichtverhältnisse zur Maximierung der Erkennungsrate seitens des CCD-Moduls sowie eine zentrierte Positionierung der Kiste innerhalb des Fahrzeuges.

8.6.2. Evaluationskriterien

Die Bewertung der Systemleistung erfolgte im Rahmen dieser Evaluation binär, d.h. dass eine Betrachtung dahingehend durchgeführt wurde, ob die Kiste erfolgreich aus dem fahrenden Fahrzeug entladen und abgelegt wurde oder nicht. Bei einer nicht erfolgreichen Entladung wurde weiterhin dokumentiert, ob der Greifarm die Kiste *fast* entladen hat – d.h. nur um wenige Zentimeter das Ziel verfehlt hat – oder ob er weit entfernt war.

8.6.3. Versuchsergebnisse und -bewertung

Die Evaluation erfolgt sowohl unter Einbeziehung aller Module als auch unter Ausschluss des Tiefenkamera-Moduls. Die Gründe hierfür werden im folgenden Teilabschnitt erläutert.

Betrieb mit Tiefenkamera-Modul

Nachdem die Evaluationen aller Teilkomponenten – auch des Tiefenkamera-Moduls – ergaben, dass diese mit hoher Genauigkeit arbeiten, war zu erwarten, dass sie im gleichzeitigen, gemeinsamen Betrieb mindestens eine ebenbürtige Performanz aufweisen. Leider war dies nicht der Fall. Wenn die von der Tiefenkamera gelieferten Informationen bei der Paketentnahme berücksichtigt wurden, kam es in keinem von 10 Testdurchläufen zu einer erfolgreichen Entnahme. In den meisten Fällen wurde der Roboterarm vom CCD-Kameramodul korrekt zum Fahrzeug geleitet, ab dem Zeitpunkt des zusätzlichen Eintreffens von Tiefeninformationen aber deutlich vom korrekten Kurs abgebracht und hat schließlich weit daneben gegriffen.

Dieser Umstand muss nicht zwingend in einer fehlerhaften Funktionsweise des Tiefenkamera-Moduls begründet sein. Die Ursache könnte beispielsweise auch im Tracking-Modul liegen, welche für die Zusammenführung der Informationen von CCD- und Tiefenkamera verantwortlich ist.

Betrieb ohne Tiefenkamera-Modul

Da der gleichzeitige Betrieb von CCD- und Tiefenkamera nicht zu befriedigenden Ergebnissen führte und der alleinige Betrieb der Tiefenkamera kein ausreichendes Maß an Informationen liefert, wurde entschieden, ausschließlich die vom CCD-Modul kommenden Informationen zur Position des Paketes zu verwerten.

Die Evaluation des Gesamtsystems in dieser Konfiguration führt zu besseren Ergebnissen. Bei 10 Testdurchläufen konnte erfolgreich eine Entnahme durchgeführt werden. Ein wichtiger Unterschied ist aber, dass die Abweichung des Greifarms auch bei einer nicht erfolgten Entnahme lediglich wenige Zentimeter betrug. Meist war der Greifarm zwar über dem Paket, schwebte jedoch circa 0.5 bis 1 Zentimeter darüber, sodass ein Ansaugen des Paketes nicht möglich war. Es ist wahrscheinlich, dass durch weitere Optimierungen der Anteil der erfolgreichen Entnahmen in dieser Konfiguration erheblich gesteigert werden könnte. Die Tabelle 8.2 illustriert nochmals die zuvor beschriebenen Ergebnisse der Evaluation.

	Erfolgreiche Entnahmen	Abweichung bei Fehlschlag
Mit Tiefenkamera	0 von 10	50 cm bis 3 m
Ohne Tiefenkamera	1 von 10	0.5 cm bis 5 cm

Tabelle 8.2.: Ergebnisse der Evaluationen, jeweils mit und ohne eingeschaltetem Tiefenkamera-Modul.

Erhöhung der Fahrzeuggeschwindigkeit

Abschließend wurde untersucht, ob das System auch dann noch zur Entnahme des Paketes in der Lage ist, wenn das Fahrzeug mit einer erhöhten Geschwindigkeit durch den Roboterkäfig fährt. Hierzu wurden zwei Versuche durchgeführt, bei denen das Fahrzeug jeweils mit 1.5-facher bzw. 2-facher Normalgeschwindigkeit mit einer Kiste beladen durch die Zelle fuhr. Obwohl der Roboterarm das Fahrzeug auch bei hoher Geschwindigkeit mit hoher Genauigkeit verfolgen konnte, wurde das Paket in keinem der Versuche gegriffen. Dies liegt vermutlich an der fest implementierten Zeitschranke von 20 Sekunden, nach deren Ablauf frühestens nach dem Paket gegriffen wird und welche der neuen Geschwindigkeit angepasst werden müsste (siehe Kapitel 7.6).

9. Zusammenfassung und Ausblick

In den vorherigen Kapiteln wurde ein Überblick über die Grundlagen der Thematik, die verwendeten Verfahren und die Realisierung des Projektes gegeben. Zudem wurden die Ergebnisse evaluiert. Im Folgenden werden in Kapitel 9.1 die Ergebnisse zusammengefasst. Des Weiteren wird in Kapitel 9.2 ein Ausblick über mögliche Ergänzungen und Erweiterungen der Realisierung in der Zukunft gegeben.

9.1. Zusammenfassung

Ziel der Projektgruppe DynOLog war die Realisierung eines automatischen Entladevorgangs von Paketen aus einem Fahrerlosen Transportsystem im Kontext der Logistik. Verwendet werden sollten dazu Methoden der zwei- und dreidimensionalen Bildanalyse zur Erkennung der Fahrzeug- und Paketposition sowie eine Bahnplanung des Roboterarms, die mit Hilfe der Bildanalyseergebnisse den Entladevorgang realisiert. Vorgestellt wurden eine Reihe von Verfahren mit denen dieses Ziel erreicht werden konnte. Die zweidimensionale Bildanalyse verwendet Daten aus einer im Versuchsraum angebrachten CCD-Kamera, um unter Anwendung eines Segmentierungs-Verfahrens mit anschließender Konturenextraktion die Fahrzeugposition zu ermitteln. Eine am Roboterarm befestigte Tiefenkamera liefert die Sensordaten für die dreidimensionale Bildanalyse. Durch Ebenenextraktion und der Anwendung weiterer Filter wird der Mittelpunkt des zu entnehmenden Pakets ermittelt. Fusioniert werden diese Ergebnisse im Tracking-Modul. Dieses verwendet den Condensation-Algorithmus um die Paketposition zu beobachten und eine Prädiktion der Position für einen zukünftigen Zeitpunkt zu leisten. Mit Hilfe dieser Prädiktion kann eine Bahn geplant werden, um das Paket in der Zukunft an dem vorhergesagten Ort zu entnehmen.

Nach Vorstellung der verwendeten Verfahren und Methoden wurden in Kapitel 8 die Ergebnisse der Einzelmodule und des Gesamtprojekts evaluiert. Es stellte sich heraus, dass mit Hilfe des vorgestellten Verfahrens lediglich in einem von zehn Fällen eine erfolgreiche Entladung stattfand. Die Evaluation der zweidimensionalen Fahrzeugpositionsbestimmung liefert bei künstlicher, konstanter Beleuchtung eine reproduzierbare Erkennungsrate des Fahrzeugs von 100 %. Der Mittelpunkt des Pakets aus der Aufnahme der Tiefenkamera wird von dem dreidimensionalen Extraktionsverfahren mit einer maximalen Abweichung von $1,5\text{cm}$ bestimmt. Es entstehen zudem keine Lokalisierungen, denen kein realer Paketmittelpunkt zu Grunde liegt. Des Weiteren wurde gezeigt, dass der implementierte Condensation-Algorithmus konvergiert. Die geschätzte Position wird dementsprechend mit längerer Laufzeit besser und kann zum Entnehmen des Pakets verwendet werden. Als letztes wurde das Verhalten des Roboters bei verschiedenen Positionseingaben evaluiert.

Es wurde ein Verfahren erstellt, das unter gegebenen Bedingungen eine automatische Fahrzeugentladung leistet. Die Erfolgsrate des Systems entspricht noch nicht der Anforderung, das System in einer realen Umgebung zu verwenden. Weitere Optimierungen, die das entwickelte Verfahren verbessern oder erweitern könnten, werden im nächsten Kapitel näher ausgeführt.

9.2. Ausblick

Ziel weiterer Entwicklungsschritte wäre es, den Einsatz des Systems in einer realen Umgebung zu ermöglichen. Dafür müsste das System an Flexibilität und Zuverlässigkeit gewinnen. Wie diese Eigenschaften erreicht werden könnten, wird im Folgenden erläutert. Dabei wird zunächst auf Verbesserungsmöglichkeiten in den einzelnen Modulen und schon bestehenden Algorithmen eingegangen. Im Anschluss werden mögliche weitere Herausforderungen aufgeführt, für die bisher keine Lösungsansätze entwickelt wurden.

Das Modul für die Fahrzeugerkennung im 2D-Bild (siehe Kapitel 7.2) stellte sich während der Evaluation (siehe Kapitel 8.2) als empfindlich gegenüber verschiedenen Lichtverhältnissen heraus. Daher könnte eine weitere Entwicklung vorsehen, dass sich die verwendeten Filter je nach Helligkeit anpassen. Auch werden bessere Ergebnisse durch die Verwendung einer qualitativ hochwertigeren Kamera erwartet.

Auch die Erkennung des Paketes mithilfe der 3D-Kamera (siehe Kapitel 7.3) stellt sich als empfindlich gegenüber den Helligkeitsveränderungen in der Halle heraus (siehe Kapitel 8.3). Hier könnte der verwendete Farbfilter durch ein Verfahren ersetzt werden, das Helligkeitsschwankungen gegenüber unempfindlicher ist. Finale Tests des Gesamtsystems zeigten auch, dass die Ergebnisse des Moduls für die Paketerkennung nicht in Echtzeit eintreffen und daher nur selten verwendet werden können. Eine Verbesserung des Gesamtsystems könnte durch eine Beschleunigung der Erkennung erreicht werden.

Innerhalb des Tracking-Moduls (siehe Kapitel 7.4) wurde nur der Einsatz des Condensation-Algorithmus für das Tracking und dessen Konvergenz im Anwendungsfall (siehe Kapitel 8.4) untersucht. In weiteren Schritten könnten alternative Trackingverfahren eingesetzt und getestet werden. Auch hier kann eine Beschleunigung der Ausführung des Algorithmus eventuell eine Verbesserung des Gesamtsystems ermöglichen. Mithilfe einer Parallelisierung der Vorgänge könnte eine Beschleunigung erreicht werden.

Im Modul der Robotersteuerung und Bahnplanung (siehe Kapitel 7.5) könnten weitere Variationen des Potentialfeldes betrachtet werden. Als Beispiel könnte eine adaptive Änderung des Potentialfeldes untersucht werden. Auch könnte die Zeitsynchronisation weiter ausgebaut werden, indem ein spezielles Protokoll eingesetzt wird.

Die bisher aufgeführten Punkte stellen Verbesserungen am bisher bestehenden System dar. Zusätzlich könnten auch Erweiterungen sinnvoll sein, um das System in realer Umgebung einsetzen zu können.

Um die Produktivität des Systems zu erhöhen, könnte die Geschwindigkeit des FTFs erhöht werden. Das erfolgreiche Greifen des Paketes aus einem FTF mit hoher Geschwindigkeit ist im Endzustand des Systemes nicht möglich (siehe Kapitel 8.6), so dass eine Entwicklung der Algorithmen für variable Geschwindigkeiten angestrebt werden sollte. Eine weitere Produktivitätssteigerung könnte durch die Realisierung der Entnahme der Pakete aus mehreren

dicht hintereinander fahrenden Fahrzeugen geleistet werden. Auch kann die Möglichkeit zur Entnahme mehrerer Pakete aus einem FTF den Durchsatz des Systems steigern. Eine weitere bisher fest vorgegebene Eigenschaft der Pakete bezieht sich auf die Paketmaße und die Farbe der Pakete. In der weiteren Entwicklung könnte die Verarbeitung bei flexiblen Paketmaßen und störenden Adressaufklebern realisiert werden. Ein weiterer Nutzen des Systems würde durch die Realisierung einer Beladung ermöglicht werden. Mit den erläuterten Erweiterungen und Verbesserungen könnte ein für die Industrie wertvolles System entstehen.

Literaturverzeichnis

- [1] G. Ullrich. *Fahrerlose Transportsysteme*. Springer Fachmedien Wiesbaden, 2014.
- [2] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA, 1987. ACM.
- [3] H.-C. Pfohl. *Logistiksysteme. Betriebswirtschaftliche Grundlagen*. Springer Verlag, 2003.
- [4] R. Jünemann and T. Schmidt. *Materialflußsysteme. Systemtechnische Grundlagen*. Springer Verlag, 2000.
- [5] P. Marwedel. *Embedded Systems Design. Embedded Systems Foundations of Cyber-Physical Systems*. Springer, 2011.
- [6] T. Bernd. Autonomes Fahren, Seminar Mobile Systeme, 2003. Universität Koblenz-Landau.
- [7] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [8] J. Woetzel. *Projektive 3D-Rekonstruktion durch Bildfolgenanalyse monokularer Kameras mit adaptierbarer Korrespondenzsuche*. PhD thesis, University of Kiel, 2001.
- [9] T. Dang. *Kontinuierliche Selbstkalibrierung von Stereokameras*. PhD thesis, 2007.
- [10] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 22*, pages 1330–1334, 1998.
- [11] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Hall, P., 2nd edition, 2001.
- [12] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Nelson Education Limited, 2008.
- [13] B. Jähne. *Digitale Bildverarbeitung*. Springer, 2012.
- [14] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.
- [15] R. Brunelli. *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley, 2009.
- [16] S. L Tanimoto. Template matching in pyramids. *Computer Graphics and Image Processing*, 16(4):356–369, 1981.

-
- [17] H. Y. Kim and S. A. De Araújo. Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast. In *Advances in Image and Video Technology*, pages 100–113. Springer, 2007.
- [18] Z. Zhang and T. Kanade. Determining the Epipolar Geometry and its Uncertainty: A Review. *International Journal of Computer Vision*, 27:161–195, 1998.
- [19] R. I. Hartley and P. Sturm. Triangulation, 1994.
- [20] Universität Siegen. 3D-Geometrieerfassung am Beispiel einer PMD-Kamera. Seminar Technologien der Fertigungsautomatisierung.
- [21] Point Cloud Library. <http://pointclouds.org/>. Zuletzt aufgerufen: 10.11.2013.
- [22] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry. Algorithms and Application*. Springer, 1997.
- [23] B. Kunle. Entwicklung und Untersuchung von Moving Least Square Verfahren zur numerischen Simulation hydrodynamischer Gleichungen, 2001. Dissertation.
- [24] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, 1981.
- [25] VDI. Richtlinie 2860: Montage- und Handhabungstechnik; Handhabungsfunktionen, Handhabungseinrichtungen; Begriffe, Definitionen, Symbole, 1990.
- [26] W. Weber. *Industrieroboter*. Fachbuchverlag Leipzig, München, 2002.
- [27] KUKA. *Technical Data KR 125/3*.
- [28] W. E. Snyder. *Computergesteuerte Industrieroboter*. VCH, 1990.
- [29] D. Schaefers. Ein Software-Tool zur Generierung und Manipulation von Industrieroboterbahnen für extrudierte Werkstücke. Bachelor thesis, 2012.
- [30] M. Oubbati. Vorlesungsskript: Einführung in die Robotik, 2009. Universität Ulm.
- [31] B. Favre-Bulle. *Automatisierung komplexer Industrieprozesse - Systeme, Verfahren und Informationsmanagement*. Springer Verlag Wien New York, 2004.
- [32] ABB. RobotStudio für IRC5. <http://new.abb.com/products/robotics/robotstudio>. Zuletzt aufgerufen: 18.11.2013.
- [33] M. Weck. *Werkzeugmaschinen - Automatisierung von Maschinen und Anlagen*. Springer Verlag, 2001.
- [34] M. Wenz. *Automatische Konfiguration der Bewegungssteuerung von Industrierobotern*. PhD thesis, Universität Fridericiana zu Karlsruhe, 2008.
- [35] C. Y. Lee. An algorithm for path connections and its applications. *Electronic Computers, IRE Transactions on*, pages 346–365, 1961.
- [36] L. Yin, Y. Yin, and C. Lin. A new potential field method for mobile robot path planning in the dynamic environments. *Asian Journal of Control*, pages 214–225, 2009.

-
- [37] M. Isard and A. Blake. CONDENSATION - Conditional Density Propagation for Visual Tracking. In *International Journal of Computer Vision, Volume 29*, pages 5 – 28, 1998.
- [38] G. Welch and G. Bishop. An Introduction to the Kalman Filter, 2006.
- [39] D. Fränken. Dynaelectroniche Zustandsschätzung. Skript der Universität Paderborn, 2006.
- [40] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. In *International Journal of Computer Vision, Volume 1*, pages 321 – 331, 1988.
- [41] S. Avidan. Support Vector Tracking. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 26*, pages 1064–1072, 2004.
- [42] S. Suzuki. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.
- [43] S. Yokoi, J.-I. Toriwaki, and T. Fukumura. An analysis of topological properties of digitized binary pictures using local features. *Computer Graphics and Image Processing*, 4(1):63–73, 1975.
- [44] P. Bender. Eine einfache Formel für den Flächeninhalt von Polygonen. *Von Geometrie und Geschichte in der Mathematikdidaktik.*, pages 53–70, 2010.
- [45] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.
- [46] R. M. Murray, S. S. Sastry, and L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., 1st edition, 1994.
- [47] Z.Y. Hu and F.C. Wu. A note on the number of solutions of the noncoplanar p4p problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):550–555, 2002.
- [48] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.
- [49] S. Gratton, A. S. Lawless, and N. K. Nichols. Approximate gauss-newton methods for nonlinear least squares problems. *SIAM Journal on Optimization*, 18(1):106–132, 2007.
- [50] J. Tsai, R.Y. ; Thomas. *IEEE International Conference on Robotics and Automation, Proceedings*, 1:554–561, 1988.
- [51] Z. Zhang. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.
- [52] H. Grossmann. Vorlesungsskript: Algebra und Geometrie. Humboldt-Universität zu Berlin.
- [53] T. Krauer. Zeitsynchronisation in Sensornetzen. pages 1–10, 2003.
- [54] KUKA System Technology. *KUKA.RobotSensorInterface 2.3*. KUKA Roboter GmbH, 2009.

- [55] Dokumentation zu Qt 4.8: Signals & Slots.
- [56] Dokumentation zu Qt 4.8: Threads and QObjects. Abschnitt: Signals and Slots Across Threads.

Abbildungsverzeichnis

2.1.	Die drei Säulen der Logistik	6
2.2.	Systeme von Logistikprozessen	7
2.3.	Beispiele verschiedener Stetigförderer	8
2.4.	Beispiel eines Sensor-Aktor-Netzwerkes	9
2.5.	Prinzip eines Ultraschallsensors	11
3.1.	Perspektivische Aufnahme der ZFT-Halle des Fraunhofer IML	14
3.2.	Illustration euklidischer Transformationen	14
3.3.	Illustration affiner Transformationen	15
3.4.	Illustration projektiver Transformationen	16
3.5.	Schematische Darstellung des Lochkameramodells	17
3.6.	Schematische Darstellung der perspektivischen Projektion	19
3.7.	Planares Kalibriermuster	20
3.8.	Bilder mit Histogrammen	22
3.9.	Histogrammglättung	23
3.10.	Anwendung des Laplace-Filters	24
3.11.	Anwendung des Canny-Algorithmus	25
3.12.	Anwendung eines Schwellwertverfahrens	27
3.13.	Resultat des Cira-tefi-Algorithmus	30
4.1.	Epipolargeometrie	32
4.2.	Schematische Darstellung von strukturiertem Licht	34
4.3.	Strukturiertes Streifenmuster	35
4.4.	Komponenten einer PMD-Kamera	36
4.5.	<i>kd</i> -Baum mit einer Illustration der Punktmengenaufteilung	37
4.6.	Triangulierung einer Punktwolke	41
4.7.	Delaunay-Triangulierung einer Punktmenge	41
4.8.	Dreiecke vor und nach dem Flip	42
5.1.	Modell der Roboterzelle	47
5.2.	Aufnahme des Industrieroboters Kuka 125/3	48
5.3.	Gelenkmodelle	49

5.4.	Seitenansicht vom Arbeitsbereich des Kuka Roboter 125/3	50
5.5.	Darstellung von Problemen der Rückwärtstransformation	52
5.6.	Darstellung des Roboterpfades für verschiedene Bewegungsformen	54
5.7.	Schematische Darstellung der Roboterpfadglättung	54
5.8.	Zweidimensionale Visualisierung des Wavefront-Algorithmus	58
6.1.	Tracking zweidimensionaler Objektposition mittels Kalman-Filter	64
6.2.	Unterschiede bei einer Tracking-Iteration mit verschiedenen Verteilungen	64
6.3.	Grafische Darstellung des Condensation Algorithmus	67
7.1.	Automatisierungsprozess als Verarbeitungspipeline	70
7.2.	Anbringung der CCD-Kamera	71
7.3.	Vergleich verzerrtes und entzerrtes Bild	75
7.4.	Darstellung der Suchraumeinschränkung und Segmentierung	76
7.5.	Ergebnis der Konturenextraktion und Konturenvereinfachung	78
7.6.	Innenliegende Ecken und approximierte Fahrzeugposition	80
7.7.	Fixpunkte zur Bestimmung der Punktkorrespondenzen	82
7.8.	Sauggreifer und Tiefenkamera	84
7.9.	Koordinatensysteme des Roboters	86
7.10.	Ein FTF des Fraunhofer Instituts	89
7.11.	Ausgleichsgerade, berechnet mit dem RANSAC-Verfahren	91
7.12.	Durch das RANSAC-Verfahren ermittelte Paketoberseite	92
7.13.	Berechnung des Mittelpunktes aus der konvexen Hülle	93
7.14.	Darstellung der gemessenen Positionsdaten des Fahrzeuges	96
7.15.	Darstellung der approximierten Fahrzeugbahn	98
7.16.	Schritte des Condensation-Algorithmus	99
7.17.	Verteilung der Condensation-Samples	100
7.18.	Darstellung der verschiedenen Anteile des Potenzialfeldes	104
7.19.	Farbskala für die Visualisierung des Potenzialfeldes	105
7.20.	Darstellung der Roboterbahn auf dem Potenzialfeld	107
7.21.	Aufbau der Robotersteuerung	109
7.22.	Aufbau des KRL-Programms	112
7.23.	RSI-Datenaustausch via Ethernet	114
7.24.	Implementierte Zeitsynchronisation	115
8.1.	Die Roboterzelle in der ZFT-Halle	118
8.2.	Position der 3D-Kamera am Roboterarm	119
8.3.	Testszenarien für die Fahrzeugerkennung	120
8.4.	Ergebnisse der Messungen unter veränderten Umgebungsbedingungen	121

8.5.	Messergebnisse des CCD-Moduls bei Abweichungen	122
8.6.	Ergebnisse der Genauigkeitsmessung	123
8.7.	Testdurchlauf des Tracking-Moduls	124
8.8.	Performanz des Trackings bei Variation der Standardabweichung	125
8.9.	Performanz des Tracking-Moduls bei Variation der Vorhersagezeit	126
A.1.	Darstellung einer Übergabestation	149
A.2.	Moduldiagramm	151
A.3.	Überwachung der Fahrzeugbahn	152

Tabellenverzeichnis

8.1. Ergebnisse Evaluation Robotersteuerung	127
8.2. Ergebnisse Evaluation Tiefenkamera	129
A.1. Qualitätsziele	153

A. Pflichtenheft

Übersicht

In der Projektgruppe soll innerhalb eines Jahres ein System realisiert werden, welches es ermöglicht eine automatisierte Be- und Entladung von FTF mit Hilfe eines Roboterarms durchzuführen. Hierzu soll ein Paket, welches sich entweder auf einer Palette oder einem FTF befindet, über Kameras erfasst und anschließend während der Fahrt eines FTF in einem definierten Bereich um den Roboterarm von diesem versetzt werden.

Zielbestimmung

In der ersten Woche der Projektgruppenzeit wurden gemeinschaftlich Kriterien erarbeitet, die das Ziel der Projektgruppe detailliert und möglichst umfassend beschreiben. Als Grundlage dienten hierfür die Vorstellungsfolien der Projektgruppenleitung, welche vorab einen groben Funktionsrahmen für das Projekt darstellten. Die aufgestellten Kriterien werden hierbei in Muss-, Wunsch- und Abgrenzungskriterien unterteilt. Erstere spezifizieren unabdingbare Leistungen, die in jedem Fall von dem im vorgegebenen Zeitraum zu erstellenden System erfüllt werden müssen. Ohne diese Funktionen ist das System für seinen gedachten Zweck nicht einsetzbar. Die Erfüllung der Wunschkriterien ist optional und somit nicht zwingend erforderlich. Sie sollten allerdings angestrebt werden, falls zum Ende noch Kapazitäten verfügbar sind. Die Abgrenzungskriterien umfassen letztlich noch Eigenschaften und Funktionen, die das System bewusst nicht erfüllen soll.

Die drei Kriterienarten werden jeweils für die im Gesamtsystem zwingend erforderlichen Komponenten beschrieben. Hierzu zählen das Paket, welches versetzt werden soll, das Fahrerlose Transportfahrzeug, welches be- und entladen werden soll und die Palette, auf der das Paket nach dem Entladen abgestellt wird.

Musskriterien

Paket Das zu erkennende Paket muss von den Kamerasystemen erkannt werden. Hierbei sollen weiße unbedruckte geschlossene Kartons verwendet werden, die lediglich mit weißem Klebeband verschlossen sind. Um störende Reflexionen zu vermeiden, sollen auf den Paketen keine Aufkleber, wie beispielsweise Adressaufkleber, zu sehen sein. Die Pakete müssen quaderförmig sein und der EURO-Norm für Verpackungsvorgaben europäischer Versandhäuser entsprechen. Korrespondierend hiermit wird eine feste Größe für die Pakete ($L=40\text{cm}$; $B=25,5\text{cm}$; $H=30\text{cm}$) vorgegeben. Das Gewicht der Pakete darf maximal 30kg betragen.

Die FTF werden mit maximal einem Paket beladen. Hierbei ist darauf zu achten, dass das Paket sich in der dafür vorgesehenen Ladefläche befindet und nicht darüber hinausragt. Weiterhin muss das Paket auf der Ladefläche vollständig aufliegen.

Fahrerloses Transportfahrzeug Die FTF fahren zu beliebigen Zeiten in den Arbeitsbereich, in dem der Roboterarm geladene Pakete greifen könnte. Es darf sich allerdings zu jeder Zeit maximal ein FTF in diesem Bereich befinden. Die Geschwindigkeit des FTF muss so gewählt werden, dass die gesamte Durchfahrzeit durch den Greifbereich des Roboterarms etwa zwei Minuten beträgt. Das FTF muss sich dabei mit konstanter Geschwindigkeit auf einer festen vordefinierten kreisbogenförmigen Bahn in dem Bereich stetig vorwärts bewegen. Die Bahn selbst muss so angelegt worden sein, dass sie mithilfe des Roboterarms an jedem Punkt erreichbar ist. Die FTF dürfen zudem nur von einer festgelegten Seite in den Greifbereich hineinfahren.

Palette Die Palette, auf welcher die von den FTF gelieferten Pakete durch den Roboterarm abgestellt werden sollen, ist zu Beginn des Umsetzvorgangs immer leer. Mithilfe des Sauggreifers entnimmt der Roboter das Paket aus dem FTF und legt es auf der Palette ab. Anschließend muss das Paket manuell von der Palette genommen werden, da sich dort maximal ein Paket befinden darf.

Wunschkriterien

Paket Optional könnten auch braune Pakete auf den FTF transportiert und vom System als solche erkannt werden. Zudem ist es denkbar, dass die Farbe des Klebebandes variiert (beispielsweise braun oder transparent) oder diverse Aufkleber auf dem Paket angebracht werden, ohne dass dies zu Problemen bei der Erkennung der Ladung führt. Des Weiteren könnte anstelle der fest definierten Größe der Pakete diese schwanken, um so eine realistischere Gesamtsituation zu schaffen. Die möglichen Paketgrößen, die umgeladen werden könnten, müssen alle der EURO-Norm entsprechen und dürfen nicht zu klein für den Greifer des Roboterarms und nicht zu groß für den Transport auf einem FTF sind.

Neben der bisherigen Beschränkung auf ein geladenes Paket können mehrere verschieden farbige Pakete nebeneinander auf einem FTF geladen sein. Die Herausforderung besteht dabei, die entsprechende Anzahl auch tatsächlich zu detektieren und innerhalb der Fahrzeit durch den Roboter greifbereich alle vorhandenen Pakete zu entnehmen. Weiterhin ist denkbar, dass ein geladenes Paket mit einer Ecke auf einer Kante des Ladebereichs aufliegt und somit schräg im FTF transportiert wird.

Fahrerloses Transportsystem Es ist wünschenswert die konstante Geschwindigkeit der FTF bei Durchfahrt durch den Greifbereich vorab zu erhöhen, sodass der Erkennungs- sowie Be-/Entladevorgang in kürzerer Zeit erfolgen muss. Die FTF könnten zudem aus einer anderen als der bisher festgelegten Richtung in den Greifbereich des Roboterarms einfahren. Um einen flexibleren Einsatz zu ermöglichen ist es weiterhin denkbar, dass die FTF sich auf anderen definierten Bahnen, die nicht zwingend kreisbogenförmig sind, durch den Bereich bewegen. Außerdem könnte sich nicht nur ein FTF in dem Bereich aufhalten, sondern mehrere gleichzeitig, die allerdings aus derselben Richtung in den Greifbereich einfahren.

Palette Um den Umsetzvorgang von Paketen realitätsnäher darzustellen ist es möglich, dass mehrere Pakete nebeneinander auf die Palette gestellt werden. Ebenfalls können maßgleiche Pakete übereinander gestapelt werden. Der Sauggreifer des Roboterarms darf Pakete nun sowohl von den FTF zur Palette als auch in entgegengesetzte Richtung bewegen.

Abgrenzungskriterien

Pakete Es werden lediglich quaderförmige Pakete erkannt. Andere Formen der Verpackung, wie beispielsweise Versandtaschen oder Paketrollen, sind nicht vorgesehen. Weiterhin dürfen Pakete nicht geöffnet sein und ein Gewicht von 30kg nicht überschritten werden. Außerdem können sämtliche nicht in den Muss- und Wunschkriterien aufgeführten Paketmaße nicht eingesetzt werden.

Auf den FTF dürfen generell zwar mehrere Pakete liegen, allerdings ist eine Stapelung dieser auf der Ladefläche der FTF nicht vorgesehen.

Fahrerloses Transportfahrzeug Die FTF dürfen zu keiner Zeit innerhalb des Greifbereichs des Roboterarms beschleunigen bzw. abbremsen. Zudem dürfen sie die vorgegebenen Bahnen nicht verlassen und quasi freie Bahnen fahren. Außerdem können zwei FTF nicht gleichzeitig aus verschiedenen Richtungen in den Greifbereich einfahren.

Palette Eine Stapelung verschieden großer Pakete auf der Palette wird nicht unterstützt. Zudem darf eine Maximalhöhe von 60cm beim Stapelvorgang nicht überschritten werden.

Produkteinsatz

Dieser Abschnitt beschreibt den Einsatz des zu entwickelnden Systems. Hierbei werden der spätere Anwendungsbereich, die Zielgruppe sowie die Betriebsbedingungen, bei denen das System eingesetzt werden soll näher erläutert.

Anwendungsbereich

Die automatisierte Be- und Entladung von FTF soll vor allem in Logistik- und Distributionszentren eingesetzt werden. Hier sind bereits heute FTF, welche Stückgut völlig autonom von Startpunkten zu vorgegebenen Übergabestationen bringen, zu finden. Die Beladung mit diesem Stückgut erfolgt zur Zeit bereits vollautomatisiert. Die Entladung hingegen wird meist noch manuell durchgeführt, daher wird eine automatisierte Entladung entwickelt (siehe Abbildung A.1).

Zielgruppen

Die Bedienung des Systems kann nur von technischen Angestellten vorgenommen werden, die vorab in die verschiedenen Funktionalitäten eingewiesen wurden.

Betriebsbedingungen

Das System kann nur in wettergeschützten Umgebungen, wie Lagerhallen, verwendet werden. Prinzipiell sollte das System in der Lage sein an sieben Tagen in der Woche jeweils 24 Stunden zu arbeiten. Ausnahmen stellen hierbei Wartungsarbeiten oder Defekte dar, die behoben

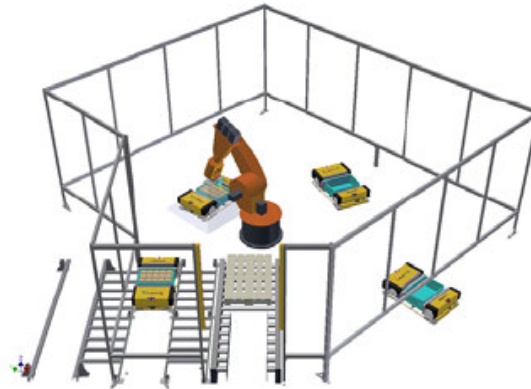


Abbildung A.1.: Darstellung einer automatisierten Übergabestation

werden müssen. Das zur Erkennung der Pakete auf den FTF mindestens benötigten Licht beträgt 150Lux, sodass künstliches Licht am besten permanent eingeschaltet bleibt.

Technische Produktumgebung

In diesem Abschnitt wird besonders auf die zur Erstellung des Systems erforderliche Hardware und Software eingegangen und diese, so weit es zum aktuellen Zeitpunkt möglich ist, spezifiziert. Des weiteren werden Sicherheitshinweise gegeben, die besonders im Umgang mit dem zum Einsatz kommenden Industrieroboter zu beachten sind.

Software

Die zu entwickelnde Software wird für ein Windows-Zielsystem (Windows XP) in C++ programmiert. Hierbei wird unter anderem auf die Point Cloud Library (PCL) zugegriffen, welche als Open-Source-Framework die Punktwolkenverarbeitung durch vorgegeben Algorithmen und Funktionen erleichtert.

Hardware

Zielsystem Es wird ein Industrie-PC verwendet, der mindestens 4GB RAM und einen leistungsstarken Prozessor beinhaltet. Das darauf befindliche Betriebssystem muss Windows XP oder höher sein. Als Grafikkarte sollte eine OpenCL unterstützende NVIDIA-Grafikkarte mit mindestens 2GB internem Speicher zum Einsatz kommen.

Kamerasysteme Zur Erkennung der Pakete werden zwei bis drei Kamerasysteme benötigt. Diese werden sowohl an einem festen hoch gelegenen Punkt in der Zelle des Roboterarms als auch am Roboterarm selbst angebracht. Eine Kamera am Roboterarm wird als 3D-Kamera von ASUS (Xtion) realisiert werden. Ihre genaue Position ist bereits getestet und ist somit vorgegeben. Entweder ein oder zwei Kameras werden den Greifbereich

des Roboters von oben abdecken, um die in den Bereich einfahrenden FTF zu erkennen. Hierbei werden CCD-Kameras eingesetzt werden.

Roboterarm Der Roboterarm, welcher die Pakete umsetzen wird, ist ein Industrieroboter (Modell KR 125/3) der Firma KUKA. Dieser wird mit einem Sauggreifer ausgestattet, womit die Pakete aus den FTF entnommen werden sollen. Der Greifer (maximale Tragekraft 40kg) hat die Maße 40cm x 12cm x 5,2cm und muss daher mit seiner rechteckigen Form passend auf den Paketen platziert werden.

Fahrerlose Transportfahrzeuge Die FTF haben ein Gewicht von 115kg und können mit einer maximalen Nutzlast von 30kg beladen werden. Diese darf die maximalen Maße von 56,5cm x 36,8cm (BxL) nicht überschreiten, da die Nutzlast in eine in den FTF stehende graue Box eingestellt werden muss. Die Beladungsbox ist nur 18cm hoch und ragt nicht aus den FTF heraus. Dies hat zur Folge, dass ggf. (sollte die Beladung weniger als 20cm hoch sein) mit dem Sauggreifer in das FTF hineingegriffen werden muss.

Sicherheitshinweise

Wichtig ist bei Arbeiten mit einem Roboter die Einhaltung der „DIN EN ISO 10218“-Norm. In dieser sind Schutzeinrichtungen und Sicherheitsmerkmale, wie beispielsweise ein Personendetektionssystem definiert, das den Roboterarm bei durchbrechen z.B. einer Lichtschranke sofort deaktiviert. So wird sichergestellt, dass während des laufenden Betriebes keiner in den Arbeitsbereich des Roboterarms eintritt. Zudem muss die Geschwindigkeit mit welcher der Roboter arbeitet, auf max. 250 mm/s reduziert werden, um hier ebenfalls Sicherheit zu gewährleisten.

Produktfunktionen

Um die zu Beginn bereits beschriebenen Aspekte der Entwicklung eines Systems zur automatisierten Be- und Entladung von fahrerlosen Transportfahrzeugen mithilfe eines Robotergreifarms umfassend zu erarbeiten, wird diese in vier Module gegliedert. Diese bekommen jeweils einen bestimmten Aufgabenbereich zugewiesen. Die verschiedenen Bereiche und deren Schnittstellen, über welche sie miteinander interagieren, sind in nachfolgendem Diagramm visualisiert (Abbildung A.2). Hieraus ist ersichtlich, dass zwei verschiedene Kamerasysteme (CCD und Tiefensensor) zur Detektierung der FTF, die Bahnplanung des Roboters sowie das Tracking der FTFs erforderlich sind. Um eine grobe Vorstellung von den Aufgaben und Lösungswegen in den einzelnen Gebieten zu erhalten, werden diese im weiteren Verlauf näher beschrieben.

Als zentrales Schnittstellen-Modul wird ein Tracking-Modul die Verbindung zwischen den weiteren drei Modulen der Entwicklung darstellen. Hierzu werden die von der CCD-Kamera sowie dem Tiefensensor beobachteten Positionen des FTF im Arbeitsbereich des Roboters zentral gesammelt und aus diesen eine Prädiktion für zukünftige Bewegung sowie Positionierung des FTF getroffen. Diese Prädiktion wird anschließend für alle weiteren Module zur Verfügung gestellt.

Mithilfe dieser berechneten Prädiktion kann sich der Greifarm des Roboters dynamisch zu den zu be- bzw. entladenden FTF bewegen. Der hierbei zurückgelegte Weg (von der Palette

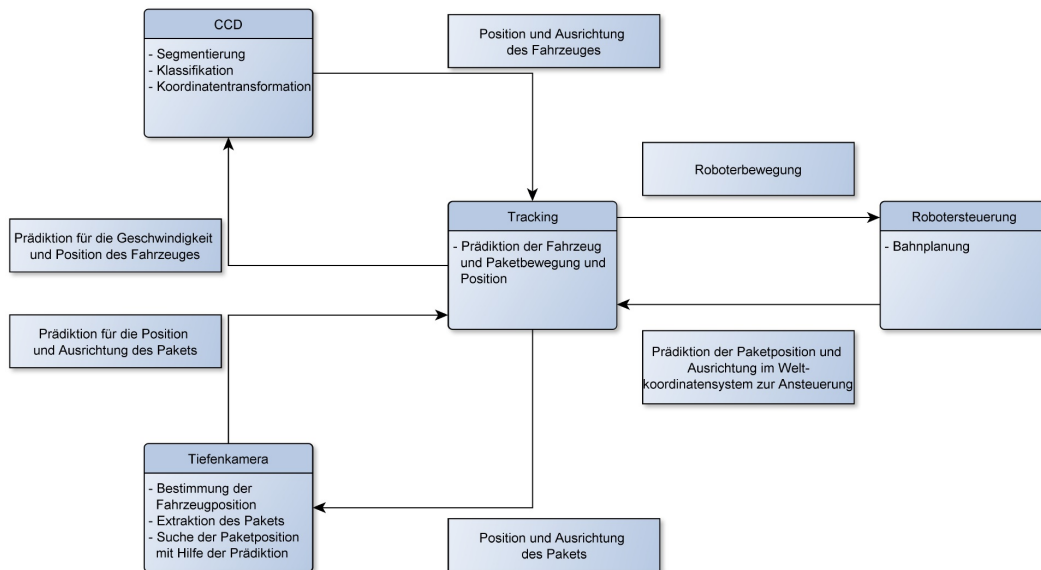


Abbildung A.2.: Diagramm der Module und Schnittstellen

als Ausgangsposition zu dem fahrenden FTF) muss allerdings vor der Durchführung der Bewegung berechnet und überprüft werden. Aus diesem Grund beschäftigt sich ein Modul mit der Planung der hierfür erforderlichen Roboterbahn. Für die zweidimensionale Planung der Bahn wird eine Potentialfeldberechnung verwendet. Die Höhe wird fest auf ein Niveau gesetzt, so dass keine Kollisionsgefahr mit den FTFs besteht.

Damit sichergestellt ist, dass keine Singularitäten auftreten oder der Roboter in den Boden fährt, wird mithilfe der inversen Kinematik die Fahrbahn des Roboters zuvor simuliert. Dazu könnten Teile des OpenRAVE-Frameworks eingesetzt werden. Dazu müssen jedoch die Denavit-Hartenberg Parameter (DH Parameter) des Roboters ermittelt werden.

Das Programm wird aus zwei Teilen bestehen: einem Client, der in KukaRobotLanguage (KRL) programmiert wird, die Ansteuerung einzelner Punkte übernimmt und das XML-Protokoll zum Nachrichtenaustausch mit dem Server vorgibt. Um eine ausreichende und vor allem kontinuierliche Geschwindigkeit zu erreichen, werden die Punkte überschiffen, bzw. miteinander verschränkt.

Der Server wird in C++ programmiert, ermittelt die Punkte, die an den Roboter gesendet werden und übernimmt die Simulation des Roboters mithilfe der inversen Kinematik. Für einen transparenten Nachrichtenaustausch wird das Serverprogramm mit dem Framework Qt programmiert. Hierüber können zukünftige Koordinaten sowie (falls erforderlich) weitere wichtige Parameter, wie beispielsweise die aktuelle Geschwindigkeit des sich im Arbeitsbereich des Roboters befindlichen FTF, vom bereits erwähnten Tracking-Modul zur Verfügung gestellt werden.

Um die Prädiktion, mit der der Roboterarm an eine bestimmte Position geführt werden kann, berechnen zu können, benötigt das Tracking-Modul aktuelle Informationen darüber, wo sich das FTF zur Zeit befindet. Eine CCD-Kamera soll hierzu so installiert werden, dass sie den

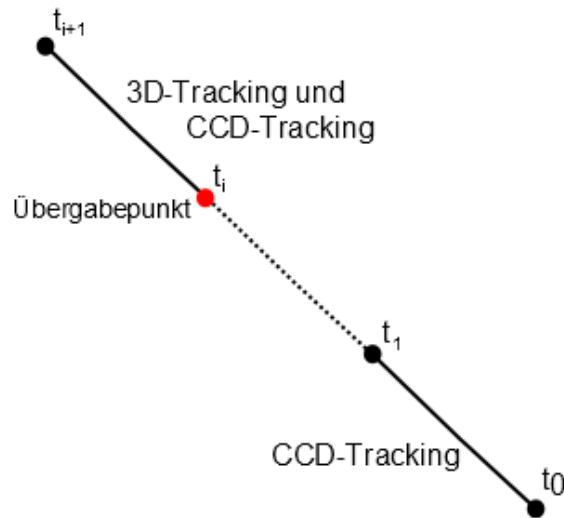


Abbildung A.3.: Überwachung der Fahrzeugbahn zu verschiedenen Zeitpunkten

Arbeitsbereich des Roboters vollständig abdeckt. Auf diese Weise wird gewährleistet, dass sie in den Arbeitsbereich einfahrende FTF in jedem Fall erkennen kann. Im Anschluss an die Installation wird eine Kamerakalibrierung vorgenommen. Ziel ist es, mithilfe der CCD-Kamera die eindeutige Bestimmung der Position des sich im Arbeitsbereich befindlichen Fahrzeuges zu ermöglichen. Dazu wird zunächst unter Nutzung kantenorientierter Segmentierungsverfahren eine Detektion des fahrerlosen Transportfahrzeugs im Kamerabild durchgeführt. Mittels projektiver Geometrie wird anschließend aus der Bildkoordinate die Koordinate des Fahrzeuges im Weltkoordinatensystem des Roboters ermittelt. Diese wird über eine Schnittstelle anderen Bereichen zur Verfügung gestellt.

Auf diese Weise kann das sich im Arbeitsbereich befindende FTF permanent verfolgt und jederzeit möglichst aktuelle Koordinaten zur Verfügung gestellt werden. Die so erhaltene grobe momentane Positionierung des FTF, welches be- bzw. entladen werden soll, wird durch eine zusätzliche Überwachung mit einem Tiefensensor präzisiert.

Mit Hilfe der Aufnahmen dieses Tiefensensors, welcher am Greifarm des Roboters befestigt ist, wird die genaue Position des zu greifenden Paketes bestimmt. Vorausgesetzt wird dabei, dass sich der Roboterarm ab einem Zeitpunkt über dem Fahrzeug befindet, sodass ein Großteil des zu greifenden Paketes durch die Tiefenkamera sichtbar ist. Dies bedeutet, dass nach erstmaliger Erkennung eines FTF im Arbeitsbereich der Roboter Greifarm nur mit Hilfe der von der CCD-Kamera gelieferten Koordinaten über das FTF geführt werden muss. Erst wenn dies erfolgt ist, beginnt der Tiefensensor mit seiner Messung (Abbildung A.3).

Das Ziel dabei ist, die obere Fläche des Paketes im Tiefenbild zu extrahieren, um die genaue Position und Ausrichtung dieser berechnen zu können und anschliessend darüber den Sauggreifer des Roboters exakt auf dem Paket zu platzieren. Hierbei wird folgendermaßen vorgegangen:

Nach Vorverarbeitung der Punktwolke durch Reduzierung der Auflösung werden zunächst die Punkte des fahrerlosen Transportsystems extrahiert. Dabei soll ein RANSAC-Verfahren angewendet werden um den Boden der Szene zu entfernen, gefolgt von einem Clustering, um das

	sehr wichtig	wichtig	unwichtig	irrelevant
Benutzerfreundlichkeit		x		
Gestaltung			x	
Effizienz	x			
Portabilität				x
Korrektheit	x			
Zuverlässigkeit	x			
Kompatibilität			x	
Lebensdauer		x		
Wartbarkeit	x			

Tabelle A.1.: Auswertung der Qualitätsziele für Software-Qualität

Fahrzeug von übrigen Objekten zu trennen. Der nächste Schritt ist das Entfernen der Punkte des fahrerlosen Transportsystems. Dazu soll ein Pattern Matching verwendet werden, da das Fahrzeug (und Aufnahmen dessen in unbeladenem Zustand) bekannt ist. Sind die Punkte des Paketes bestimmt, lässt sich durch eine Filterung anhand der Normalenvektorausrichtung die obere Fläche extrahieren, dessen Position der Mittelpunkt der Punkte ist. Die Ausrichtung wird durch das Berechnen der konvexen Hülle der Punkte angegeben.

Durch die Prädiktion, die das Tracking-Modul in jedem Zeitintervall liefert, kann die Erkennung der Fläche im Folgenden vereinfacht werden. Es muss lediglich in einer lokalen Umgebung der Prädiktion nach der Fläche des Paketes gesucht werden. Hierfür werden Pattern Matching oder ein RANSAC-Verfahren verwendet.

Produktleistungen

Da es sich bei dem Gesamtsystem um ein im realen Umfeld in Distributionszentren einsetzbares System handelt, ist die Zeit, die erforderlich ist, um die Fahrzeuge zu erkennen und diese zu be- bzw. entladen, kritisch. Daher sollte die Zeit möglichst minimiert werden, um eine möglichst effiziente Lösung für das reale Umfeld zu erzielen.

Außerdem ist es wichtig vor allem beim Greifen der Pakete, aber auch beim optionalen Stapeln dieser, mit dem Roboterarm möglichst genau zu arbeiten. Hierbei ist die Höhe des Sauggreifers beim Greifvorgang der Pakete zu berücksichtigen, da dieser nur einen Puffer von zwei Zentimeter in Form von Schaumstoff erlaubt. Zudem muss auf die Orientierung des rechteckigen Sauggreifers geachtet werden, da sonst durch bspw. fehlerhaftes Greifen Pakete unterwegs verloren gehen könnten.

Qualitätsziele

Bei der Entwicklung des beschriebenen Systems müssen die in der DIN-ISO-Norm 9126 definierten Qualitätsziele für Software-Qualität berücksichtigt werden. Diese sind in Tabelle A.1 gelistet und mit ihrer jeweiligen Relevanz für das vorliegende System versehen.

Abnahmekriterien

Zur erfolgreichen Abnahme des Projektes werden den Auftraggebern finale Dokumente zur Verfügung gestellt. Diese umfassen den hinreichend kommentierten Quellcode, in den sich ein C++ erfahrener Programmierer aufgrund seiner Verständlichkeit und Strukturierung innerhalb von zwei Arbeitstagen einarbeiten können muss. Außerdem wird ein Kurzhandbuch beigefügt, in dem die wichtigsten Schritte, welche vor allem bei Inbetriebnahme der Software beachtet werden müssen, verständlich erläutert werden. Neben den Dokumenten wird eine kurze Arbeitsunterweisung in die Inbetriebnahme und Bedienung des Gesamtsystems sowie eine Einführung in erforderliche Sicherheitsrichtlinien stattfinden.