Hochschule Karlsruhe
University of
Applied Sciences

Moritz Arntz
Sersbachstraße 30
76596 Forbach

armo1014@h-ka.de
Matrikelnummer: 73280
Abgabedatum: TT. Monat JJJJ

Konzeption und Implementierung eines kameragestützten Robotersystems zur vollautomatisierten Aufhängung von Strukturbauteilen für kathodische Tauchlackierungen

Design and implementation of a camera-supported robot system for the fully automated suspension of structural components for cathodic dip painting

Masterarbeit

vorgelegt zur Erlangung des Mastergrades
der Hochschule Karlsruhe – University of Applied Sciences

Fakultät für Mechatronik und Maschinenbau

Robotik und künstliche Intelligenz in der Produktion

Betreuer: Prof. Dr.-Ing. Christian Wurll

Zweitgutachter: Prof. Dr.-Ing. Habil. Björn Hein

Aus Gründen der besseren Lesbarkeit wird auf eine durchgängige Verwendung der weiblichen Form verzichtet. Die geschlechterbezogenen Bezeichnungen gelten sowohl in der weiblichen als auch in der männlichen Form.

Kurzfassung

Konzeption und Implementierung eines kameragestützten Robotersystems zur vollautomatisierten Aufhängung von Strukturbauteilen für kathodische Tauchlackierungen

Deutsche Zusammenfassung. (max. zehnzeilig)

Schlüsselwörter:
Schlüsselwörter, meiner, Thesis, Schlüsselwörter, meiner, Thesis, Schlüsselwörter, meiner, Thesis (max. 10)

Abstract

Design and implementation of a camera-supported robot system for the fully automated suspension of structural components for cathodic dip painting

English abstract.

Keywords:
Keywords, of, my, thesis, Keywords, of, my, thesis, Keywords, of, my, thesis (max. 10)

Danksagung

Optional

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

iRAS	Forschungsgruppe Robotik und Autonome Systeme
KI	Künstliche Intelligenz
KNN	Künstliche Neuronale Netze
MRK	Mensch-Roboter-Kollaboration/-Kooperation/-Koexistenz

Listingverzeichnis

1 Einleitung

1.1 Kathodische Tauchlackierung

1.2 Problemstellung

1.3 Zielsetzung

1.4 Aufbau der Arbeit

2 Theoretische Grundlagen

2.1 Robotik

2.1.1 Übersicht über Bewegungsarten (PTP, LIN, Spline)

2.1.2 Mensch-Roboter-Kollaboration

2.2 Computer Vision

2.2.1 Klassische Bildverarbeitung (Filter, Konturenerkennung, etc.)

2.3 Neuronale Netze

2.3.1 Klassifizierung mit neuronalen Netzen

2.3.2 Regressionsaufgaben

2.3.3 Faltungsnetze

2.3.4 Segmentierung

2.3.5 Größere Architekturen

3 Stand der Technik

3.1 Robotik und Mensch-Roboter-Kollaboration

3.1.1 Präzisionsanwendungen

3.2 Visuelle Erkennung und Verarbeitung von Objekten

3.3 Kameratechnik und Bildverarbeitung

3.4 Steuerung bzw. Regelung des Roboters

3.4.1 Positionsbasierte Bewegung

3.4.1.1 Spline-Kurven

3.4.2 Regelungsbasierte Bewegung und Visual Servoing

3.4.2.1 Image Based Visual Servoing

3.4.2.2 Position Based Visual Servoing

4 Stand der Forschung

4.1 Anforderungen an Präzision

4.2 Visual Servoing

4.3 Rekonstruktion von Tiefeninformation aus 2D-Bildern

5 Methoden

5.1 Herangehensweise

5.1.1 Konzept Bildverarbeitung – Erkennung der Haken und Finden von Parametern

5.1.2 Konzept Robotersteuerung – Einfädelbewegung

5.2 Evaluierung

5.2.1 Bildverarbeitung

5.2.1.1 Auswertung klassische Bildverarbeitung

5.2.1.2 Arbeit mit neronalen Netzwerken (Training, Loss-Kurven, Test-Datenset)

5.2.2 Robotik

5.2.2.1 Ziel – Finden von Bewegungsmethoden für die verschiedenen Prozessteile

5.2.2.2 Methoden zur Auswertung – Messung der Genauigkeit

6 Konzept

6.1 Hardware

6.1.1 Kassow Robots KR1205

6.1.2 Robot Operating System 2

6.1.3 Pytorch und Ultralytics

6.2 Bildverarbeitung auf 2D-Pixelebene

6.2.1 Einführung

- *Welche Eigenschaften / Merkmale haben die Haken?*
- *4 Hakenmodelle*
    - *Gemeinsamkeiten / Unterschiede*
    - *Welche Herausforderungen bringen die jeweiligen Modelle mit sich?*
- *Welche Informationen aus der Bildverarbeitung sind zur Lösung der Aufgabe notwendig?*
- *Welche Methoden stehen zur Auswahl?*

6.2.2 Einflüsse der Umgebung

- *Welche Art Hintergrund ist sinnvoll?*
- *Ausleuchtung des Kamerabilds*
    - *Möglicherweise Simulation von verschiedenen Lichtarten*
    - *Vergleich von Lichtquellen und Ausleuchtung*
- *Finden sinnvoller Beleuchtungen im Kontext der Aufgabenstellung*
- *Regelung der Beleuchtungshelligkeit*

6.2.3 Klassische Bildverarbeitung

- *Sinnvolles Merkmal ist definitiv die Form des Hakens*
- *Farbe eher nutzlos*
    - *Schwarz und glänzt*
- *Hinführung zu Ideen (Template, Keypoints, …)*

6.2.3.1 Template Matching

- *Einfache Herangehensweise*
- *Findet jedoch nur sehr ähnliche Haken*
- *Bietet dann auch nur eine allgemeine Information, wo sich der Haken im Bild befindet*
    - *Keine Informationen über die exakte Form oder die Koordinaten von Spitze bzw. Senke*

6.2.3.2 Keypoints und Deskriptoren

- *Idee: Finden von Keypoints, die eindeutig einem Haken zugeordnet werden können (unabhängig von der Form)*
- *Problematisch, weil schwarz und glänzend -> schlechte Eigenschaften für Keypoints*
    - *Keypoints brauchen Strukturen, Kontraste -> glatte Oberflächen schwierig*

6.2.3.3 Handling von Instanzen

- *Es gibt mehrere Haken, die über Leiste oben verbunden sind*
- *Kantenerkennung (Canny, Sobel) finden zwar zuverlässig Kanten, betrachten aber alles als EIN zusammenhängendes Objekt -> keine Trennung von Instanzen*
- *Feste Suchbereiche*
    - *Setzt voraus, dass man die Leisten vorher kennt, Kameraparameter und Distanzen immer gleich sein müssen*
    - *schlechte Voraussetzungen für Kollaboration -> generell schlecht, wenn sich ein Zielobjekt ändern kann und trotzdem immer verarbeitet werden soll*

6.2.3.4 Bedeutung für die Aufgabenstellung

- *Klassische Bildverarbeitung funktioniert vor allem gut, wenn Objekt immer gleich*
    - *Für diese Aufgabe hier schlecht*
- *Wegen Abweichungen der Haken kann nicht auf feste Muster detektiert werden*
    - *Man bräuchte unendlich viele Templates, die alle möglichen Fälle abdecken*
    - *Widerspricht der eigentlichen Idee der Aufgabe*
- *Tieferes Verständnis eines Hakens benötigt -> künstliche Intelligenz*

6.2.4 Einsatz von neuronalen Netzen

- *Vorteil von Machine Learning in der Bildverarbeitung im Kontext der Aufgabe*
- *Erhofftes Ziel durch den Einsatz von ML in der Bildverarbeitung*

6.2.4.1 Instanzsegmentierung -> Zuordnung der Haken

- *Warum ist Segmentierung hilfreich -> Instanzen*

- *Berücksichtigung der Hakenform über Maskierungen (pixelweise)*
- *Instanzsegmentierung vs. Segmentierung*
- *Für die vorliegende Aufgabe Instanzsegmentierung -> einzelne Adressierung der Haken-Instanzen notwendig*

6.2.4.2 Datenset

- *Wie wurde gelabelt?*
    - *Was ist wichtig bei der Erstellung der Masken?*
- *Wie viele Bilder bzw. Hakensamples?*
- *Anteile der verschiedenen Hakenmodelle im Datenset*
- *Möglichkeiten zur Erhöhung der Varianz im Datenset -> Augmentation, Spiegelung, Rotation*

6.2.5 Faster-RCNN

- *Erster Ansatz*
- *Beschreibung des Outputs (anschließend NMS notwendig)*

6.2.5.1 Architektur

- *Was ist die Grundidee eines Faster-RCNN?*
- *Wie setzt sich die Architektur zusammen?*
- *Welchen Vorteil bieten die Bestandteile der Architektur?*

6.2.6 Yolo-Netzwerke

- *Zweiter Ansatz zur Steigerung der Geschwindigkeit*
- *Beschreibung des Outputs (keine NMS mehr notwendig)*

6.2.6.1 Architektur

- *Siehe oben*
- *Was macht diese Netze so schnell?*

6.2.7 Kurzes Fazit und Gegenüberstellung von Faster-RCNN und Yolo

- *Warum wird aller Voraussicht nach Yolo verwendet?*
- *Welche Netzgröße von Yolo wird verwendet bzw. zur Evaluation herangezogen?*
- *Ausblick -> Wie wird der Netzoutput für die Anwendung weiterverwednet?*

6.3 Rekonstruktion der Tiefe aus der Pixelebene

6.3.1 Einführung, Problematik, Idee

- *Bisheriger Output nur auf Pixelebene*
    - *Könnte für Image Based Servoing ausreichen*
    - *Problem – alle 6 Dimensionen müssen über 2D-Bildmerkmale regelbar sein*
        - *Besondere Herausforderung -> Tiefeninformation*
        - *Kann typischerweise gut mit Größen/Abstands-Verhältnissen realisiert werden*
            - *Funktioniert hier nicht, da die Haken vorher nicht bekannt sind*
            - *Haken haben unterschiedliche Dimensionen und Dicken*
        - *Somit wird jede Art der Robotersteuerung/Regelung zumindest einen Richtwert für die Tiefe oder einen Bezugspunkt benötigen*

6.3.2 Messung der Tiefe mit Tiefenkameras

- *Bezug auf Einleitungskapitel (Grundlagen)*
- *In diesem Kapitel werden 2 Methoden getestet -> Time-Of-Flight und Triangulation*

6.3.2.1 Test von Time-of-Flight

- *Intel RealSense D435*
- *Einsatzgebiete (Bin Picking, Aufgaben mit weniger Präzisionsanforderungen)*
- *Messung der Tiefe über Laufzeitmessung -> einfaches Prinzip*
- *Test mit verschiedenen Abständen -> Ergebnis -> Haken im Tiefenbild meistens gar nicht sichtbar*

6.3.2.2 Test von Stereo-Triangulation

- *Roboception rc_visard 65*
- *Tests Roboception mit verschiedenen Abständen*
- *Ergebnis*
    - *Eigenartige Verzerrungen in den Tiefenbildern (vllt durch Reflektion)*
    - *Keypointdetektion funktioniert bei schwarz glänzenden Oberflächen ziemlich schlecht*

6.3.3 Tiefenrekonstruktion über Stereo-Triangulation

6.3.3.1 Idee (Punkte aus NN Output)

- *Neuer Ansatz -> anstatt Keypoints wird der NN-Output für die Berechnung der Triangulation verwendet*
    - *Betrachtung des Haken aus zwei verschiedenen Perspektiven*
    - *Finden der Pixelkoordinaten von Spitze und Senke*
    - *Berechnung von Triangulation*
        - *Wo kommen die benötigten Rechengrößen her?*
            - *Baseline aus Roboterkoordinaten*
            - *Disparität aus Bildern*
            - *Intrinsische Kameraparameter aus Objektiv*

6.3.3.2 Berechnung der Triangulationspunkte aus NN-Output-Masken

- *NN gibt lediglich BBox-Koordinaten und die Masken aus*
- *Für Triangulation sind Spitze und Senke entscheidend*
    - *Müssen zunächst berechnet werden*
        - *Ansatz 1 – Mittelpunkt der BBox*
            - *Einfach zu berechnen*
            - *Liegt je nach Verzerrungen der Maske nicht in der Mitte der Maske*
            - *Manchmal nichtmal innerhalb der Maske*
        - *Ansatz 2 – Durchschnittspunkt der Maske*
            - *Vergleichbar mit Schwerpunkt der Maske*
            - *Erklärung (Formeln)*
            - *Findet den tatsächlichen Mittelpunkt der Maske*
            - *Verzerrungen sind hier weniger problematisch*
        - *Ansatz 3 – Kmeans*
            - *Bekanntes ML-Verfahren -> kurze Erklärung*
            - *Generiert Clusterzentrum auf Basis der Verteilung der Masken-Pixel*
            - *Gleiches Ergebnis wie Ansatz 2 – dauert aber länger*
    - *Vergleich der drei Ansätze mit Hilfe von Bildbeispielen (Jupyter Notebook)*

6.3.4 Vergleich von Triangulationsverfahren

- *Ansätze aus Paper sollen getestet und verglichen werden*
- *Messungen/Versuchsreihe mit drei Methoden*

- *Erklärung des Versuchsaufbaus und des Ablaufs*

6.3.4.1 Horizontale Triangulation

- *Standard Stereo-Triangulation*
- *Vergleichbar mit Kameraanordnungen bei Stereo-Kameras*

6.3.4.2 Vertikale Triangulation

- *Baseline entlang der y-Achse*

6.3.4.3 Kombinierte Triangulation

- *Berechnung beider Triangulationsmethoden*
- *Anschließender Mittelwert*

6.3.4.4 Vergleich der drei Testergebnisse

- *Kombiniert bietet die besten Ergebnisse*
- *Dauert jedoch deutlich länger -> 4 Bewegungen für jeden Haken notwendig*
- *Möglicherweise Kompromiss sinnvoll aus Laufzeit und Genauigkeit*

6.3.5 Integration der Triangulation als Scan-Vorgang in den Prozess

- *Beschreibung des Scan-Prozesses*
- *Zwei Aufgaben -> Berechnung der realen XYZ-Koordinaten für jeden Haken und globale Zuordnung der Haken*
- *Beschreibung des Prozesses mit Ablaufdiagramm oder Grafik*

6.3.6 Filterung von Netz-Output und Roboterschwingung

- *Für Scan-Vorgang sind Bewegungsstopps erforderlich*
    - *Bringen Schwingungen des Roboterarms mit sich*
    - *Überträgt sich auf den NN-Output*
- *Ziel des Abschnitts: Filterung der Roboterschwingung und allgemeines Zittern des NN-Outputs filtern*

6.3.6.1 Schwingungen des Roboterarms bei Bewegunsstopp

- *Testmessungen bei verschiedenen Geschwindigkeiten*
- *Systemtheoretische Beschreibung des Schwingungsverhaltens*
    - *Frequenz -> abhängig von Fahrgeschwindigkeit*
    - *Verhalten -> asymptotisch*

6.3.6.2 Grundrauschen des Netz-Outputs

- *NN-Output rauscht auch bei Roboter-Stillstand Leicht*
- *Sollte ebenfalls minimiert werden*

6.3.6.3 Vergleich von verschiedenen Filtern (EMA-Filter, Lowpass-Filter, Sliding-Window-Filter)

- *Für jeden Filtertyp*
  - *Glättung einer aufgenommen Testreihe*
  - *Unterschiedliche Filterparameter*
- *Vergleich aller Filter*
- *Auswahl eines Filtertyps im Kontext der Aufgabenstellung*

6.3.6.4 Filterung von Fehldetektionen im Netz-Output

- *NN detektiert ab und zu fälschlicherweise Haken, wo es eigentlich keine gibt*
- *Sollten ebenfalls rausgefiltert werden*
- *Filter basierend auf Frames*
  - *Eine neu auftauchende Hakeninstanz muss eine bestimmte Anzahl Frames vorhanden sein, bevor sie als valide zählt.*
  - *Kurze Beschreibung des Algorithmus*
  - *Gleiches Verhalten für detektierte Haken, die für kurze Frames verschwinden*

6.4 Steuerung bzw. Regelung des Roboterarms

- *Bezugnahme zu verschiedenen Ansätzen aus Stand der Technik*
- *Aufgabe beinhaltet verschiedene Schritte, die jeweils unterschiedliche Methoden erfordern*
  - *Greifen des Bauteils (positionsbasiert)*
  - *Heranfahren an Haken (positionsbasiert)*
    - *Erweiterung um A* der um Kollision herumplant*
  - *Genaues positionieren am Haken (regelungsbasiert)*
  - *Einfädeln bzw. Einhängen (regelungsbasiert)*
  - *Zurückfahren (positionsbasiert)*
- *Vor allem interessant -> Einfädelungsvorgang (regelungsbasiert)*

6.4.1 Position-Based-Visual-Servoing-Verfahren im Kontext der Aufgabenstellung

- *Beschreibung der Idee*
- *Die Punkte XYZ kommen aus der Triangulation*
- *Für das Einfädeln und Einhängen müssen nun sinnvolle Merkmale gefunden werden*

6.4.1.1 Merkmale zur Repräsentation der Regelgrößen im 6D-Raum

- *Rechtwinklige Ausrichtung zur tangentialen Gerade am Haken entlang*

6.4.1.2 Berechnung der Stellgröße aus geometrischen Beziehungen

- *Wie kommt die Stellgröße zustande?*
- *Vektorrechnung*

6.4.2 Reduzierung der Regelung auf ein Abfahren der berechneten Trajektorie

- *XYZ Koordinaten sind fest gespeichert*
- *Für eine Regelung im klassischen Sinne bräuchte man konitnuierlich neu berechnete Werte aus dem Kamerabild*
- *Andere Idee: Berechnung einer festen Trajektorie, die dann abgefahren wird*

6.4.2.1 Skelettierung der Instanz-Maske auf Pixelebene

- *Idee der Skelettierung*
- *Warum ist eine Skelettierung notwendig?*

6.4.2.2 Finden eines Pfades für das Einfädeln (Gegenüberstellung von Breitensuche und A*)

- *Warum wird ein Suchverfahren benötigt?*
- *Was sind die Unterschiede zwischen BFS und A**
    - *In Bezug auf Laufzeit*
- *Wie wird der Pfad gesucht?*
- *(Ablaufdiagramm)*

6.4.2.3 Berechnung der Trajektorie über parametrisierte Spline-Interpolation

- *Verteilung von N Punkten zwischen Spitze und Senke*
- *Wie können die XY-Werte der Pfad-Punkte berechnet werden? -> Pixel_per_mm-Verhältnisse*
- *Interpolation des Tiefenwerts mit Spline-Interpolation*

- *Problem: Spline nur 1 Inputgröße -> XY-Koordinaten gehen also nicht*
- *Pearametrisierte Spline-Interpolation -> der zurückgelegte Weg auf dem Pfad ist der Input*

6.4.2.4 Beschreibung des Ablaufs zum Einhängen des Bauteils anhand der berechneten Trajektorie

- *Anfahren der Punkte*
- *Übersicht über Koordinatensysteme und erforderliche Transformationen*
- *Weiterschalten der Punkte*
- *Pseudoregelung - Positionsregelung (mit festem Feedback)*

6.4.3 Mögliche Erweiterungen für Servoing-Verfahren (Zweite Kamera, Echtzeit-Triangulation)

- *Weiterer Ansatz mit zweiter Kamera*
- *Triangulation wird dauerhaft in Echtzeit berechnet*
- *Könnte bessere Genauigkeit bringen*
- *Eliminierung von Fehlern in Triangulationsrechnung bei Scan-Vorgang*
- *Wird aufgrund des Aufwands hier nicht weiter betrachtet*

7 Implementierung

- *Kapitel zur Beschreibung der Implementierung*
- *Enthält softwaretechnische Beschreibungen*
- *Enthält Training der neuronalen Netze*

7.1 Aufbau und Training der Neuronalen Netze zur Bildverarbeitung

- *Wurden in Jupyter Notebooks programmiert*
- *Trainiert über GPU*
- *Beschreibung der vorhandenen Hardware*

7.1.1 Faster-RCNN

- *Modell von Pytorch*
- *ResNet50 Backbone*

7.1.1.1 Trainingsverlauf

- *Optimizer*
- *Loss-Funktionen*
- *Parameter – Learning Rate, Epochenanzahl*
- *Lossgrafiken etc.*
- *Dauer des Trainings*

7.1.1.2 Inferenz

7.1.1.3 Non-Maximum-Suppression

- *Warum wird NMS benötigt?*
- *Wie funktioniert NMS hier im Anschluss an den Output von Faster-RCNN?*

7.1.2 Yolo-Netzwerke

- *Verschiedene Modelle von Ultralytics*

7.1.2.1 Trainingsverlauf

- *Siehe oben*

7.1.2.2 Inferenz

7.1.3 Weiterverarbeitung des Outputs

- *Bilden der Instanzen*
- *Zuordnung von Spitzen und Senken zu den jeweilig korrekten Hakeninstanzen*
- *Sortierung in Dict*
- *Dict liefert Information über die lokal erkannten Haken im Bild*

7.2 Prozessaufbau und Softwarearchitektur

7.2.1 Ablauf des Prozesses

7.2.1.1 Scan-Vorgang und Stereo-Triangulation

7.2.1.2 Greifen des Bauteils

7.2.1.3 Handling der Haken-Instanzen (Welche Haken werden benutzt, …)

7.2.1.4 Kollisionsfreies Anfahren des Hakens

7.2.1.5 Positionierung für Einfädeln

7.2.1.6 Einfädeln und Einhängen des Bauteils

7.2.1.7 Kollisionsfreies Zurückfahren und erneuter Griff eines neuen Bauteils

7.2.2 Aufbau der Softwarearchitektur

7.2.2.1 Schnittstelle und Interaktion mit Roboter

7.2.2.2 Ablaufdiagramme für Softwarebestandteile

8 Evaluation

8.1 Evaluation der trainierten Architekturen

8.1.1 Faster-RCNN mit ResNet50 Backbone

8.1.1.1 Testdatenset Ergebnisse

8.1.1.2 Laufzeitmessung

8.1.2 Yolo-Architekturen - Nano, Small, Medium

8.1.2.1 Testdatenset Ergebnisse

8.1.2.2 Laufzeitmessungen

8.1.3 Gegenüberstellung der Architekturen

8.1.3.1 Beste Genauigkeit

8.1.3.2 Anforderungen an die Inferenzlaufzeit

8.1.3.3 Kompromiss zwischen Genauigkeit und Inferenzlaufzeit

8.2 Evaluation der Tiefenrekonstruktion über Stereo-Triangulation

8.2.1 Variation der Triangulationsmethoden

8.2.1.1 Horizontale Triangulation

8.2.1.2 Vertikale Triangulation

8.2.1.3 Kombinierte Triangulation

8.2.2 Variation der translatorischen Geschwindigkeit

8.2.2.1 v = 5mm/s

8.2.2.2 v = 10mm/s

8.2.2.3 v = 15mm/s

8.2.2.4 v = 20mm/s

8.2.2.5 v = 25mm/s

8.2.2.6 v = 30mm/s

8.2.3 Variation der Hakenmodelle

8.2.3.1 Modell 1 – gewinkelt

8.2.3.2 Modell 2 – gerade schmal

8.2.3.3 Modell 3 – gerade breit

8.2.3.4 Modell 4 – rund

8.2.4 Variation der Kameraposition

8.2.4.1 Frontal (Test mit Hakenmodell 1-4)

8.2.4.2 30 Grad (vertikal) von links (Hakenmodell 1, 2, 4)

8.2.4.3 30 Grad (vertikal) von rechts (Hakenmodell 1, 2, 4)

8.2.4.4 30 Grad (vertikal) 20 Grad (horizontal) von links (Hakenmodell 1, 2, 4)

8.2.4.5 30 Grad (vertikal) 20 Grad (horizontal) von rechts (Hakenmodell 1, 2, 4)

8.2.5 Variation der Beleuchtungshelligkeit

8.2.5.1 Mit Helligkeitsregelung

8.2.5.2 Ohne Beleuchtung

8.2.5.3 20% Helligkeit

8.2.5.4 50% Helligkeit

8.2.5.5 80% Helligkeit

8.2.5.6 100% Helligkeit

8.3 Evaluation des Einfädel-Prozesses

8.3.1 Variation des Hakens (Anpassung auf verformte Haken des gleichen Typs)

8.3.1.1 Auswahl von 5 Haken (möglichst unterschiedlich)

8.3.2 Variation der Pfad-Interpolation

8.3.2.1 N = 0

8.3.2.2 N = 1

8.3.2.3 N = 3

8.3.2.4 N = 5

8.3.2.5 N = 10

8.3.3 Variation der Suchverfahren bei der Pfad-Suche

8.3.3.1 Breitensuche BFS

8.3.3.2 Planer A*

8.4 Taktzeitbetrachtung

8.4.1 Zeitmessung der einzelnen Prozessschritte

8.4.1.1 Scan-Prozess (alle Triangulationsmethoden)

8.4.1.2 Heranfahren und Positionieren

8.4.1.3 Einfädeln (mit N=1, N=5, N=20 – für alle Hakenmodelle 1 und 4)

8.4.2 Zeitmessung des vollständigen Prozesses

8.4.2.1 Ein Zyklus (von Scan der Leiste bis Aufnahme des zweiten Bauteils)

8.4.2.2 Vollständige Bestückung einer Leiste

9 Diskussion

10 Zusammenfassung und Ausblick

Literaturverzeichnis

Fakultät W. (2016). Richtlinien und Hinweise zur Anfertigung wissenschaftlicher Arbeiten an der Fakultät für Wirtschaftswissenschaften. S. 1-46.

Wurll, C. (2018). Durchführungshinweise für Bachelor- und Masterarbeiten. S. 1-4.

Anhang

A Anhang A

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Quellen und Hilfsmittel angefertigt habe. Alle Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegen.

Karlsruhe, den TT. Monat JJJJ

_____

Moritz Arntz