Fraunhofer IPA

HAW HAMBURG

Bachelorarbeit

Van Uoc Phung

Entwicklung eines Cobot-Sensor-Systems zur Online-Erkennung und Verschweißung von Schlitz-Zapfen-Verbindungen

Fakultät Technik und Informatik *Faculty of Engineering and Computer Science*
Department Fahrzeugtechnik und Flugzeugbau *Department of Automotive and Aeronautical Engineering*

Van Uoc Phung

Entwicklung eines Cobot-Sensor-Systems zur Online-Erkennung und Verschweißung von Schlitz-Zapfen-Verbindungen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Mechatronik*
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Thomas Frischgesell
Zweitgutachter: Christoph Martin, M. Sc.
Betreuer: Erik Schäfer, M. Sc.

Eingereicht am: 20. Januar 2025

Van Uoc Phung

Thema der Arbeit

Entwicklung eines Cobot-Sensor-Systems zur Online-Erkennung und Verschweißung von Schlitz-Zapfen-Verbindungen

Stichworte

Schweißnaht-Extraktion, Merkmalserkennung, Punktwolkenverarbeitung, Schlitz-Zapfen-Verbindungen, kollaborative Roboter (Cobots), automatisierte Schweißverfahren

Kurzzusammenfassung

Die Bachelorarbeit befasst sich mit der Entwicklung eines Cobot-Sensor-Systems zur Online-Erkennung von Schlitzen und Verschweißung von Schlitz-Zapfen-Verbindungen. Das System vereinfacht und automatisiert den Schweißprozess, wodurch der Bedarf an umfangreichen Roboterprogrammierkenntnissen entfällt und die Produktionseffizienz, insbesondere in High-Mix/Low-Volume-Produktion, gesteigert wird. Die Arbeit basiert auf dem Projekt *ProcessIt* und führt Methoden zur Verarbeitung von Punktwolken ein, darunter die Randpunkterkennung, die Generierung medialer Achsen und die Identifikation der Schlitzbreite. Diese wurden in Python mit der Open3D-Bibliothek entwickelt und in das ROS-Framework integriert. Tests zeigen eine erfolgreiche Erkennung von Schlitz-Zapfen-Merkmale, die Trajektorienplanung und die Online-Verfolgung, wobei Herausforderungen bei schmalen Schlitzen, der Verfolgung bei hohen Geschwindigkeiten und verrauschten Daten bestehen. Die Arbeit hebt Optimierungspotenziale hinsichtlich Verarbeitungszeit, Parameteranpassung und der Erweiterung auf gekrümmte Schlitzgeometrien hervor.

Van Uoc Phung

Title of Thesis

Development of a Cobot Sensor System for Online Detection and Welding of Mortise and Tenon Joints

Keywords

Weld seam extraction, feature detection, point cloud processing, mortise and tenon joints, collaborative robots (cobots), automated welding processes

Abstract

The thesis focuses on the development of a Cobot-Sensor system for online detection of slits and welding of mortise and tenon joints. The system simplifies and automates the welding process, eliminating the need for extensive robot programming skills and enhancing production efficiency, particularly for high-mix/low-volume production environments. The work leverages the project *ProcessIt* and introduces methods for point cloud processing, including boudary detection, medial axis generation, and slit width identification, implemented in Python using the Open3D library and integrated into the ROS framework. Testing demonstrates successful detection of slit and mortise-tenon features , trajectory planning, and online following, though challenges remain with narrow slits, high-speed tracking, and noisy data. The thesis highlights optimization opportunities for processing speed, parameter refinement, and extending applicability to curved slit geometries.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

3D dreidimensional.

Cobot Kollaborativer Roboter (*Collaborative robot*).

DBSCAN *Density-Based Spatial Clustering of Applications with Noise*.

Fraunhofer IPA Fraunhofer-Institut für Produktionstechnik und Automatisierung.

FRNN Nachbarn im festen Radius (*Fixed-radius near neighbors*).

k-NN k-nächste Nachbarn (*k-nearest neighbors*).

k-d-Baum k-dimensionaler Baum (*k-d Tree*).

PCA Hauptkomponentenanalyse (*Principal Component Analysis*).

RANSAC *random sample consensus*.

ROI *Region of Interest*.

ROS *Robot Operating System*.

TCP *Tool Center Point*.

1 Einleitung

1.1 Motivation

Das Schweißen als Fertigungsprozess ist ein kompliziertes Verfahren und erfordert hohe Präzision und Qualität, um die strukturelle Integrität und Zuverlässigkeit des Endprodukts zu gewährleisten. Es erfordert eine sorgfältige Kontrolle von Variablen wie Temperatur, Geschwindigkeit und Ausrichtung. Darüber hinaus ist das Handschweißen eine körperlich anstrengende und potenziell gesundheitsschädliche Arbeit, da man ständig hohen Temperaturen, ultravioletter Strahlung und potenziell schädlichen Dämpfen ausgesetzt ist.

Industrieroboter sind eine der Lösungen, die bei Schweißarbeiten eingesetzt werden. Industrieroboter bieten erhebliche Vorteile, wie hohe Präzision, langfristige Wiederholgenauigkeit und die Minimierung des menschlichen Kontakts mit gefährlichen Arbeitsbedingungen. Üblicherweise werden diese Roboter über Teach-Panel auf bestimmte Punkte vorprogrammiert (Teach-In-Verfahren oder *Teach and Playback*) und können diese mit hoher Wiederholgenauigkeit abfahren. Jedoch ist dieser Ansatz oft mit Kenntnissen der Roboterprogrammierung und erhöhten Kosten verbunden. Diese Ansätze sind für kleine und mittlere Unternehmen ungeeignet, da der Programmieraufwand bei kleinen Losgrößen zu hoch ist und kontinuierliche Anpassungen für unterschiedliche Produkte (High-Mix/Low-Volume-Produktion) erforderlich sind.

Ein anderer Ansatz zum Schweißen ist die Verwendung kollaborativer Roboter (Cobot) [42]. Cobots sind dafür entwickelt, dass sie mit dem Menschen zusammenarbeiten beziehungsweise ihn bei der Reduzierung der Aufgaben unterstützen, indem sie die Vorteile des Menschen (Flexibilität, Urteilsvermögen, Kreativität, Erfahrung, Intuition und Überblick) und die Vorteile des Roboters (Ausdauer, Präzision und Stärke) kombinieren [8]. Cobots werden häufig zusammen mit spezialisierter Software entwickelt, um den Arbeitsaufwand des Benutzers bei Schweißarbeiten zu reduzieren. Der Benutzer muss dabei nur

einfache Einrichtungsvorgänge wie die Ausrichtung der Schweißnaht durchführen. Für die Automatisierung des Schweißprozesses wird der Cobot mit der Schweißausrüstung und einem zusätzlichen Sensor ausgestattet. Mit Hilfe der Sensorik und Datenverarbeitung ist es möglich, die Schweißnaht automatisch zu erkennen und im Raum zu lokalisieren. Dies dient als Grundlage für die Ermittlung der notwendigen abzufahrenden Trajektorie des TCPs.

Während des Fertigungsprozesses müssen einige Einzelteile zusammengefügt werden. Beim Fügen werden zwei oder mehr Bauteile relativ zueinander positioniert und dann durch Schweißen dauerhaft miteinander verbunden. Eine der relativen Positionierungsmethoden ist die durchgehende Schlitz-Zapfen-Verbindung.

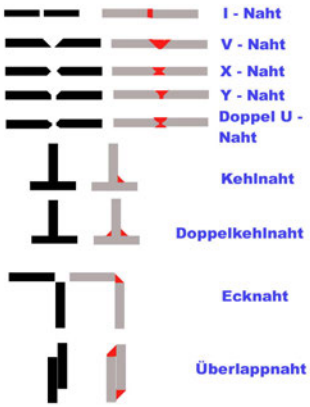


Abbildung 1.1: Schweißnahtarten [23]

Der Großteil der Forschungs- und Entwicklungsarbeiten zur Schweißnaht-Detektion ist Schweißnähten mit einfachen Geometrien wie Kehlnaht, Ecknaht, I-Naht, V-Naht oder Überlappnaht gewidmet. Allerdings existiert derzeit keine dem Autor bekannte automatisierte Schweißlösung für Schlitz-Zapfen-Verbindungen. Eine besondere Herausforderung bei der Schweißnaht von Schlitz-Zapfen-Verbindungen besteht darin, dass der Schlitz durch zwei Breitseiten begrenzt wird. Dies erfordert spezifische Ansätze zur präzisen Bestimmung der Start- und Endpunkte der Schweißnaht, die bislang noch nicht umfassend untersucht wurden.

Dementsprechend besteht das Ziel dieser Arbeit darin, eine Lösung für Cobots mit Sensor-Add-On zu entwickeln, um Schlitze zu erkennen und Bewegungen entlang der Länge der Schlitze zu planen, um Schlitz-Zapfen-Verbindungen zu verschweißen. Durch diese Entwicklung wird das Schweißen von Schlitz-Zapfen-Verbindungen zwischen Bauteilen

automatisiert und vereinfacht, sodass umfangreiche Kenntnisse in der Roboterprogrammierung nicht zwingend erforderlich sind. Gleichzeitig wird das Fixieren der Bauteile durch das Schweißen der Schlitz-Zapfen-Verbindung erleichtert, wodurch das Vorheften und der Einsatz von Fixiervorrichtungen eingespart werden können. Dies führt zu einer Steigerung der Produktionsgeschwindigkeit in Fertigungsprozessen, insbesondere für High-Mix/Low-Volume-Produktionen in kleinen und mittleren Unternehmen.

Diese Arbeit ist eine Weiterentwicklung des Projekts ***ProcessIt*** der Gruppe „Roboterprozesse und Kinematiken“ am Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA.

1.2 Das Fraunhofer-Institut für Produktionstechnik und Automatisierung

Das **Fraunhofer-Institut für Produktionstechnik und Automatisierung (Fraunhofer IPA)** [17] ist eines der führenden Institute für angewandte Forschung im Bereich Produktionstechnik, Automatisierung und Robotik und hat seinen Sitz in Stuttgart. Es wurde 1959 gegründet und ist Teil der **Fraunhofer-Gesellschaft** - die führende Organisation für angewandte Forschung in Europa. Das Fraunhofer IPA arbeitet eng mit Industriepartnern zusammen, um innovative Lösungen für die industrielle Produktion zu entwickeln. Dabei liegt der Fokus auf der Optimierung von Prozessen, der Einführung neuer Technologien und der Steigerung der Effizienz in der Fertigung. Zu den Forschungsschwerpunkten gehören unter anderem Robotik, Automotive, Maschinen- und Anlagenbau, Elektronik und Mikrosystemtechnik, Energie, Medizin- und Biotechnik sowie Prozessindustrie.

Die Abteilung **Roboter- und Assistenzsysteme** [18] am Fraunhofer IPA konzentriert sich auf die Entwicklung innovativer Technologien und Lösungen im Bereich der Robotik und Automatisierung für industrielle Anwendungen und für den Dienstleistungsbereich. Die Expertise der Abteilung umfasst verschiedene Aspekte der Robotik, darunter die Entwicklung von Industrierobotern für die Automatisierung von Fertigungsprozessen, die Gestaltung von Mensch-Roboter-Kollaborationssystemen, die Integration von Sensorik und künstlicher Intelligenz für intelligente Robotiksysteme sowie die Entwicklung von Assistenzsystemen zur Unterstützung von Menschen in verschiedenen Arbeitsumgebungen.

Die Gruppe „**Roboterprozesse und Kinematiken**“ [16] ist eine Entwicklungsgruppe innerhalb der Abteilung Roboter- und Assistenzsysteme. Die Gruppe beschäftigt sich mit der Entwicklung von Softwarefunktionen, die speziell für Schweißroboter mit Laserlinien-Sensor-Add-On ausgelegt sind. Das Hauptziel dieses Projekts ist es, eine Automatisierungslösung namens ***ProcessIt*** zu entwickeln, die Robotern, insbesondere Cobots, dabei hilft, beim Schweißen effizienter zu arbeiten. *ProcessIt* ermöglicht eine Bahnplanung basierend auf den Daten des Laserlinien-Sensors. Dadurch können adaptive Nahtdetektion und Schweißen in einem Arbeitsschritt durchgeführt werden, ohne lange Wartezeiten. Benutzer können über das Teach-Panel Parameter schnell und intuitiv einstellen oder programmieren.

1.3 Aufgabenstellung

Das Ziel dieser Arbeit ist die Entwicklung einer Lösung für Cobots mit einem Sensor-Add-On, die es ermöglicht, Schlitze online zu erkennen und den Roboter entlang ihrer Länge zu bewegen, um Schlitz-Zapfen-Verbindungen zu schweißen. Die Lösung soll eine automatische Erkennung des Schlitzes und der beiden Breitseiten gewährleisten. Zudem wird angestrebt, den Schlitz während des Schweißprozesses online zu verfolgen und sowohl die Trajektorie als auch die Schweißaktivität des Cobots dynamisch entlang der Schlitzlänge anzupassen. Das übergeordnete Ziel ist eine benutzerfreundliche Automatisierungslösung, die keine vertieften Kenntnisse in der Roboterprogrammierung erfordert.

„Online“ bedeutet in diesem Zusammenhang, dass der Sensor kontinuierlich Daten sammelt, um Schlitze zu erkennen und die Bahn sowie die Schweißaktivität des Cobots in einem Schritt anzupassen. Im Gegensatz dazu steht die „Offline“-Methode, bei der die gesamte Punktwolke der Verbindungsoberfläche vorab erfasst und verarbeitet wird. Die Bahnplanung erfolgt anschließend auf Basis dieser Daten, bevor der Schweißprozess beginnt.

Der Schwerpunkt dieser Bachelorarbeit liegt auf der Softwareentwicklung, insbesondere auf der Verarbeitung von Punktwolken zur präzisen Schlitzerkennung und der anschließenden Generierung geeigneter Trajektorienpunkte für den Endeffektor. Da es sich um eine Erweiterung handelt, baut die Lösung auf der bestehenden System- und Softwarearchitektur des Projekts *ProcessIt* auf. Der bisherige Funktionsumfang des Systems umfasst ausschließlich die Bearbeitung von Schweißnähten in Form von Kehl- und Ecknähten, während eine automatische Lösung zur Erkennung von Schlitzen oder I-Nähten

derzeit noch fehlt. Aspekte wie die Konvertierung von 2D-Sensordaten in 3D-Daten, Hand-Auge-Kalibrierung oder spezifische Herausforderungen beim Schweißen fallen nicht in den Umfang dieser Arbeit.

1.4 Methodisches Vorgehen

Die Vorgehensweise dieser Arbeit basiert auf dem V-Modell. Die Vorgehensweise umfasst:

1. **Anforderungsdefinition**: Zu Beginn werden die Ziele und technischen Anforderungen an die Entwicklung ermittelt. Dabei werden die notwendigen und optionalen Anforderungen an die Funktionalität, Geometrie der Schlitz-Zapfen-Verbindung, Performanz oder weitere relevante Kriterien definiert. Basierend auf diesen Kriterien wird eine detaillierte Anforderungsliste erstellt, die als Grundlage für die weitere Entwicklung dient.

2. **Einarbeitung in das bestehende System**: Es erfolgt eine intensive Einarbeitung in die bestehende System- und Softwarearchitektur des Projekts ProcessIt sowie in die Entwicklungsumgebung, einschließlich ROS und des *Motion Planning Frameworks* MoveIt. Auf Basis dieser Einarbeitung wird die Funktionsstruktur für die Entwicklung definiert und der Grundstein für die Integration der neuen Lösung in das bestehende System gelegt.

3. **Konzeption**: Im nächsten Schritt werden mögliche Lösungen recherchiert, insbesondere bestehende Algorithmen zur Punktwolkenverarbeitung, Merkmalserkennung und Methoden zur automatischen Schweißnahtextraktion. Diese werden hinsichtlich ihrer Stärken und Schwächen im spezifischen Kontext analysiert und hinsichtlich ihrer Eignung für die Teilfunktionen sowie deren Kombination bewertet. Basierend auf dieser Analyse wird ein geeignetes Konzept für die Schlitzerkennung ausgewählt.

4. **Implementierung**: In dieser Phase wird das zuvor entwickelte Konzept in einzelne Module unterteilt und schrittweise umgesetzt. Dies umfasst die Auswahl geeigneter Bibliotheken sowie die Programmierung neuer Klassen, Methoden und Funktionen. Jedes Modul wird in der Simulationsumgebung getestet, um seine Funktionalität und Präzision sicherzustellen. Die Simulation ermöglicht es, potenzielle Probleme

frühzeitig zu erkennen und zu beheben, bevor die Module in das Gesamtsystem integriert werden.

5. **Integration und Test am realen Roboter**: Nach Abschluss der Implementierung wird die Lösung basierend auf der zuvor definierten Funktionsstruktur in das bestehende System integriert. Die Funktionalität der Lösung sowie des gesamten Systems wird anschließend mit realen Sensordaten getestet. Dabei wird überprüft, ob die Erkennung und die TCP-Bewegung für verschiedene Schlitz-Zapfen-Verbindungen und Schlitze durchgeführt werden können. Abschließend wird die Lösung anhand der Anforderungen validiert.

1.5 Anforderungen

Auf Grundlage der oben genannten Aufgabenstellung wurde eine Liste von Anforderungen für die Entwicklung und Implementierung erstellt. Die Liste ist im Einzelnen in der Tabelle 1.1 gelistet. Diese Liste dient der Festlegung der Entwicklungsrichtung und definiert klar die Anforderungen an das System, wobei zwischen zwingenden Anforderungen (*Forderung*) und optionalen Anforderungen (*Wunsch*) unterschieden wird. Forderungen sind essenziell, um die grundlegende Funktionalität und Kompatibilität des Systems sicherzustellen. Wünsche hingegen sind nicht zwingend erforderlich, tragen jedoch zur Verbesserung der Funktionalität und Effizienz des Systems bei. Sie werden je nach ihrer Relevanz weiter unterteilt in *sehr wichtig*, *wichtig*, *wenn möglich* und *nicht wichtig*, um die Priorität innerhalb des Entwicklungsprozesses zu setzen.

Die Anforderungen sind in verschiedene Kategorien unterteilt, die unterschiedliche Aspekte des Systems abdecken, darunter allgemeine Rahmenbedingungen, Performanz, Geometrie, Erkennung und Trajektorienplanung:

- **Allgemein (A)**: Diese Kategorie umfasst grundlegende Anforderungen an die Nutzung bestehender System- und Softwarestrukturen von *ProcessIt*. Die bevorzugte Programmiersprache ist Python aufgrund der schnellen Ideenumsetzung und der verfügbaren Unterstützungsbibliotheken wie Numpy und Open3D. Darüber hinaus sind die Wiederverwendung vorhandener Parameter und die zukünftige Erweiterbarkeit gewünschte Anforderungen.

- **Performanz (P)**: Anforderungen in dieser Kategorie betreffen die Prozessabläufe und deren Effizienz. Es wird zwischen Offline- und Online-Verarbeitung unterschieden, wobei der Fokus auf der parallelen Bearbeitung von Scannen, Erkennen und Generierung der Trajektorie liegt.
- **Geometrie des Schlitzes (G)**: Hier werden die geometrischen Eigenschaften der Schlitze begrenzt, einschließlich Form, Breite, Tiefe und Länge. Dabei ist die Hauptform des Schlitzes ein Rechteck mit linearen Kanten.
- **Erkennen (E)**: Diese Kategorie bezieht sich auf die Erfassung und Identifizierung von Schlitzkonturen und deren relevanten Maßen. Eine hohe Erkennungsgenauigkeit sowie die Unempfindlichkeit gegenüber Störeinflüssen wie Lichtbögen und geometrischem Rauschen sind hierbei entscheidend.
- **Trajektorienplanung (T)**: Anforderungen an die Planung und Generierung der Trajektorien für die Schweißnaht sind in dieser Kategorie zusammengefasst. Dabei ist es wichtig, dass die Trajektorie in der Mitte der Breite und entlang der Länge des Schlitzes erzeugt wird.

Bei dieser Liste ist zu beachten, dass die Entwicklung von Online-Lösungen zwar angestrebt wird, jedoch auch Offline-Lösungen parallel entwickelt werden sollten. Online-Lösungen sind effektiv für längere Schlitze, da sie Variationen und Abweichungen entlang der gesamten Schlitzlänge in Echtzeit erfassen und berücksichtigen können. Bei kurzen Schlitzen hingegen ist die Datenerfassung begrenzt, und die Variationen entlang der Schlitzlänge sind minimal. Die kontinuierliche Datenverarbeitung und Anpassung, wie sie bei Online-Lösungen erforderlich ist, würde hier lediglich zu unnötigem Rechenaufwand führen. Offline-Lösungen hingegen bieten die Möglichkeit, eine Analyse und Trajektorienplanung vor dem Schweißprozess durchzuführen, was sie für kurze Schlitze zu einer effizienteren und genaueren Alternative macht.

Nr.	Anforderungen	F/W
A	**Allgemein**	
A.1	Nutzung der bestehenden Systemstruktur von *ProcessIt*	Forderung
A.2	Nutzung der bestehenden Softwarearchitektur von *ProcessIt*	Forderung
A.3	Programmierung in Python	Wunsch: wichtig
A.4	Beibehaltung der Parameter von *ProcessIt*	Wunsch: wenn möglich
A.5	Modulare und erweiterbare Programmierung	Wunsch: wenn möglich
P	**Performanz**	
P.1	Offline: Erst Scannen, Erkennen und dann Generierung der Trajektorie	Forderung
P.2	Online: Paralleles Scannen, Erkennen und Generierung der Trajektorie	Forderung
P.3	Verarbeitung nur eines Schlitzes pro Ablauf	Wunsch: wichtig
P.4	Anpassung der Schweiß- und Erkennungsgeschwindigkeit	Wunsch: wenn möglich
G	**Geometrie des Schlitzes**	
G.1	Schlitzform: Rechteck mit linearem Verlauf	Forderung
G.2	Erweiterbarkeit für abgerundete Rechtecke oder Spline-Verläufe	Wunsch: nicht wichtig
G.3	Schlitzbreite: 2 – 10 mm	Wunsch: wichtig
G.4	Schlitztiefe: 1 – 20 mm	Wunsch: wichtig
G.5	Schlitzlänge: 20 – 100 mm	Wunsch: wichtig
G.6	Benutzerdefinierte Nahtlänge als optimierte Variable	Wunsch: wenn möglich

Nr.	Anforderungen	F/W
E	**Erkennen**	
E.1	Erkennung des Schlitzes und der Schlitzkontur	Forderung
E.2	Automatische Erkennung des Schlitzanfangs und Schlitzendes	Forderung
E.3	Erkennung der Schlitzrichtung	Wunsch: sehr wichtig
E.4	Erkennung von Schlitzbreite und -länge	Wunsch: sehr wichtig
E.5	Erkennung der Schlitztiefe	Wunsch: wenn möglich
E.6	Keine signifikante Beeinträchtigung der Schlitzerkennung durch geometrisches Rauschen	Wunsch: sehr wichtig
E.7	Keine Beeinträchtigung durch Lichtbogen	Wunsch: wenn möglich
E.8	Funktionsfähigkeit der Online-Erkennung im Bereich relevanter Schweißparameter	Wunsch: wenn möglich
T	**Trajektorienplanung**	
T.1	Offline-Generierung der Trajektorie der Schweißnaht	Forderung
T.2	Online-Generierung der Trajektorie der Schweißnaht	Forderung
T.3	Die Trajektorie ist in der Mitte der Breite und entlang der Länge des Schlitzes	Forderung
T.4	Offset am Schlitzanfang und -ende	Wunsch: wenn möglich

Tabelle 1.1: Anforderungsliste

1.6 Struktur der Arbeit

Im nächsten Kapitel 2 werden die Grundlagen zur Schlitz-Zapfen-Verbindung, Roboterkinematik, dem Einsatz von Laserlinien-Scannern sowie *Robot Operating System* (ROS) erläutert. Zudem wird das Projekt *ProcessIt* vorgestellt, einschließlich seiner System- und Softwarearchitektur sowie des Ablaufs des Schweißprozesses. Kapitel 3 gibt einen Überblick über relevante Ansätze aus der Literatur zur Punktwolkenverarbeitung und Schweißnahterkennung. Die Konzeption der Lösung wird in Kapitel 4 beschrieben, während Kapitel 5 die Implementierung der einzelnen Module detailliert darlegt. Kapitel 6 präsentiert die Testergebnisse und validiert die entwickelte Lösung. Abschließend fasst Kapitel 7 die Ergebnisse zusammen und gibt einen Ausblick auf mögliche Erweiterungen und Optimierungsmöglichkeiten des Systems.

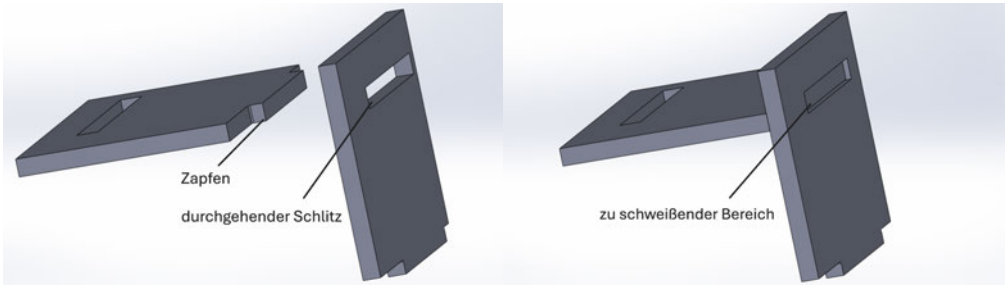
2 Grundlagen und Überblick über das Projekt *ProcessIt*

2.1 Grundlagen

Im Folgenden sind die wesentlichen theoretischen Grundlagen aufgeführt, die in dieser Arbeit berücksichtigt werden.

2.1.1 Schlitz-Zapfen-Verbindung

Die **Schlitz-Zapfen-Verbindung** ist eine stabile und häufig verwendete Methode zum Verbinden von Bauteilen. Insbesondere bei Schweißkonstruktionen bietet sie erhebliche Vorteile. Der Schlitz dient als natürliche Führung, wodurch die präzise Positionierung der zu fügenden Bauteile erleichtert wird und diese automatisch in die korrekte Ausrichtung gebracht werden.



(a) (b)

Abbildung 2.1: Schlitz-Zapfen-Verbindung

Die in dieser Bachelorarbeit untersuchte Verbindung ist eine Schlitz-Zapfen-Verbindung, wie in der Abbildung 2.1 dargestellt. Die beiden Bauteile sind aus metallischen Werkstoffen gefertigt und können daher geschweißt werden. Diese Verbindung besteht aus einem

Zapfen mit rechteckigem Querschnitt, der in ein entsprechendes durchgehendes Schlitz bzw. Zapfenloch passt. Der Zapfen geht jedoch nicht vollständig durch den Schlitz, sondern endet in einem Abstand vor der gegenüberliegenden Ebene. Dieser Raum des Schlitzes wird absichtlich zum Schweißen freigehalten.

2.1.2 Roboterkinematik

Eine wesentliche Aufgabe bei der Roboterprogrammierung von Industrierobotern ist die Beschreibung der Pose (Position und Orientierung) des Endeffektors bzw. *Tool Center Point* (TCP). Die Beschreibung der Pose kann mithilfe mehrerer Koordinatensysteme definiert werden, typischerweise im Operationsraum oder Konfigurationsraum.

Im Operationsraum wird die Pose des TCP x durch kartesische Koordinaten für die Position und Drehwinkel für die Orientierung gegenüber dem Weltkoordinatensystem beschrieben. Die Position $x_{Position}$ wird durch drei Freiheitsgrade (x, y, z) definiert, während die Orientierung $x_{Orientierung}$ üblicherweise in der Euler-Winkel-Form mit ebenfalls drei Freiheitsgraden (α, β, γ) dargestellt wird.

Alternativ kann die Orientierung auch in der Quaternion-Darstellung ausgedrückt werden, die im Projekt *ProcessIt* häufig verwendet wird:

$$\hat{q} = \hat{q}_0 + \hat{q}_1 i + \hat{q}_2 j + \hat{q}_3 k$$

Dabei sind $\hat{q}_0, \hat{q}_1, \hat{q}_2, \hat{q}_3$ reelle Zahlen, und i, j, k sind Basis-Elemente, die den folgenden Regeln genügen:

$$\begin{aligned} i^2 = j^2 = k^2 = -1 \\ i \cdot j = k, j \cdot i = -k \\ j \cdot k = i, k \cdot j = -i \\ k \cdot i = j, i \cdot k = -j \end{aligned} \tag{2.1}$$

Quaternionen bieten den Vorteil, Singularitäten in der Berechnung zu vermeiden und stets Orthogonalität sicherzustellen [19].

Im Konfigurationsraum werden die Gelenkvariablen des Industrieroboters bzw. Manipulators, also die Winkel oder Wege, in einem Vektor q zusammengefasst. Die Dimension

bzw. die Anzahl der Komponenten des Vektors entspricht der Anzahl unabhängiger Gelenke n und damit der Anzahl der Freiheitsgrade der mechanischen Struktur.

Die Kinematik stellt einen geometrischen und mathematischen Zusammenhang zwischen Operations- und Konfigurationsraum dar, wobei zwischen direkter und indirekter Kinematik unterschieden wird. Mit Hilfe der direkten Kinematik kann die Pose des Endeffektors x im Operationsraum ausgehend von den bekannten Gelenkvariablen q im Konfigurationsraum berechnet werden:

$$x = \begin{bmatrix} x_{Position} \\ x_{Orientierung} \end{bmatrix} = f(q)$$

Im Gegensatz zur direkten Kinematik beschreibt die indirekte Kinematik die Berechnung der Gelenkvariablen q in Abhängigkeit von der gewünschten Pose x:

$$q = f^{-1}(x)$$

2.1.3 Laserlinien-Scanner

Ein Laserlinien-Scanner dient zur Erfassung von Objektoberflächen und besteht typischerweise aus einer Laserkomponente, die eine dünne Laserlinie auf die Oberfläche des Objekts projiziert, einem Sensor zur Erfassung der Reflexion und einem integrierten Controller zur Berechnung der Daten. Der Scanner arbeitet nach dem Lichtschnittverfahren, das auf dem Triangulationsprinzip basiert. Dabei wird für jeden Punkt der Linie die Abstandsinformation (Z-Achse) anhand des Projektionswinkels α, des Einfallswinkels β und des Abstands zwischen Laser und Sensor b berechnet:

$$z = b\frac{tan\alpha \cdot tan\beta}{tan\alpha + tan\beta}$$

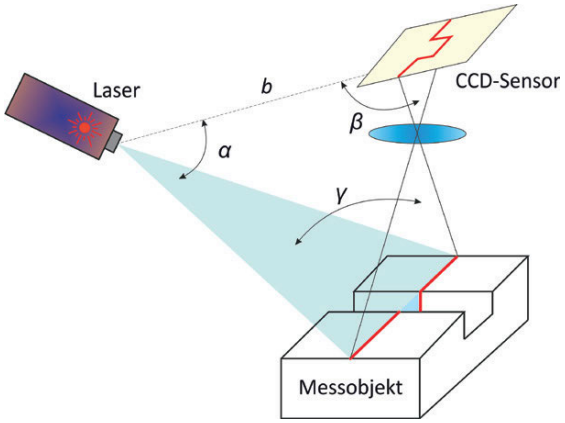


Abbildung 2.2: Prinzip des Laserlinien-Scanners [43]

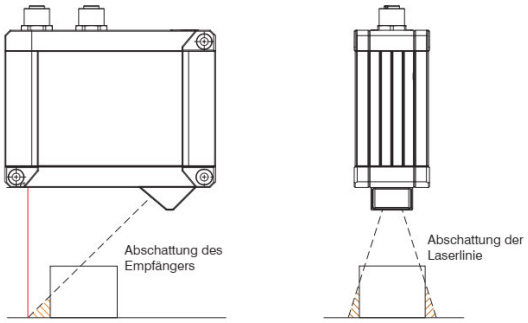


Abbildung 2.3: Abschattungen [27]

Neben der Abstandsinformation wird die exakte Position jedes Punktes (X-Achse) auch erfasst und vom Controller ausgegeben. Der Scanner erzeugt dann eine 2D-Punktwolke im Koordinatensystem des Sensors, die danach nachgearbeitet werden muss.

Der Laserlinien-Scanner bietet die Vorteile wie hohe Auflösung und geringe Größe. Darüber hinaus wird er kaum von der Umwelt beeinflusst. Er kann sowohl kleine als auch große Messobjekte mit beliebiger Geometrie erfassen. Daher wird der Laserlinien-Scanner häufig in bahnbasierten Prozessen eingesetzt, die hohe Präzision erfordern, wie beim Schweißen, Lackieren oder in der industriellen Messtechnik [33].

Neben den Vorteilen sind jedoch die Einschränkungen des Laserlinien-Scanners zu beachten. Um die gesamte Oberfläche des Messobjekts zu erfassen, muss sich der Scanner um das Objekt herum bewegen und die Einzelaufnahmen verknüpfen. Und bei Schweißarbeiten kann das Licht des Schweißlichtbogens die Genauigkeit des Sensors beim Reflektieren

von Erträgen beeinträchtigen. Darüber hinaus hat die Oberfläche oder die Beschaffenheit des Objekts großen Einfluss auf die Reflexion der Laserlinie. Abschattung, wie in Abbildung 2.3 dargestellt, kann auftreten, wenn die Laserlinie hinter steilen Kanten ganz oder teilweise blockiert wird. Sie muss bei der Entwicklung der Schlitzerkennung basierend auf Daten des Laserlinien-Scanners berücksichtigt werden.

2.1.4 *Robot Operating System* (ROS)

***Robot Operating System* (ROS)** [40], das ursprünglich im Jahr 2007 von dem *Stanford Artificial Intelligence Laboratory* der Stanford Universität entwickelt wurde [38], ist eine kostenlose Open-Source-Middleware, die für die Entwicklung von Robotern sowie Roboteranwendungen verwendet wird. Es bietet eine Sammlung von Tools und Bibliotheken, mit denen Entwickler die Komponenten eines Robotersystems (Sensoren, Aktoren und Controller) einfach verbinden und die Entwicklung von Roboteranwendungen beschleunigen können. Das Grundkonzept von ROS besteht darin, den Nachrichtenaustausch zur Übertragung und Synchronisierung von Daten zwischen verschiedenen Komponenten zu nutzen. ROS ist modular strukturiert und setzt sich aus mehreren unabhängigen, aber miteinander kommunizierenden Komponenten zusammen:

1. **Knoten** (*Nodes*): Unabhängige Softwaremodule, die spezifische Aufgaben ausführen.
2. **Themen** (*Topics*): Kommunikationskanäle zwischen Knoten.
3. **Nachrichten** (*Messages*): Strukturierte Daten für den Informationsaustausch.
4. **Dienste** (*Services*): Anfragen und Antworten zwischen Knoten.
5. **Parameter Server** (*Parameters*): Zentrale Konfigurationsparameter.
6. **Aktionen** (*Actions*): Lang laufende Aufgaben.
7. **Launch-Dateien**: Starten und Konfigurieren mehrerer Knoten.
8. **Bibliotheken und Tools**: Für Entwicklung, Simulation und Visualisierung.

MoveIt [28] ist ein flexibles Framework für Bewegungsplanung und Manipulation (*Motion Planning Framework*) in ROS. Es integriert Löser der inversen bzw. indirekten Kinematik, Algorithmen zur Trajektorienplanung und Kollisionserkennung in eine einheitliche

ROS-Schnittstelle wie z. B. *Open Motion Planning Library* (OMPL), *Pilz Industrial Motion Planner* oder *Covariant Hamiltonian Optimization for Motion Planning* (CHOMP) [36]. MoveIt läuft auf ROS und baut auf den ROS-Nachrichten- und Build-Systemen auf. Es nutzt einige der gängigen ROS-Tools wie ROS-Visualizer (RViz), Gazebo und *Universal Robot Description Format* (URDF).

2.2 Überblick über das Projekt *ProcessIt*

In diesem Abschnitt wird ein Überblick über das Sensor-Cobot-System zur automatisierten Schweißlösung für kleine und mittelgroße Bauteile im Rahmen des Projekts *ProcessIt* gegeben. Die Abbildung 2.4 zeigt den realen Schweißroboter im Einsatz am Fraunhofer IPA.

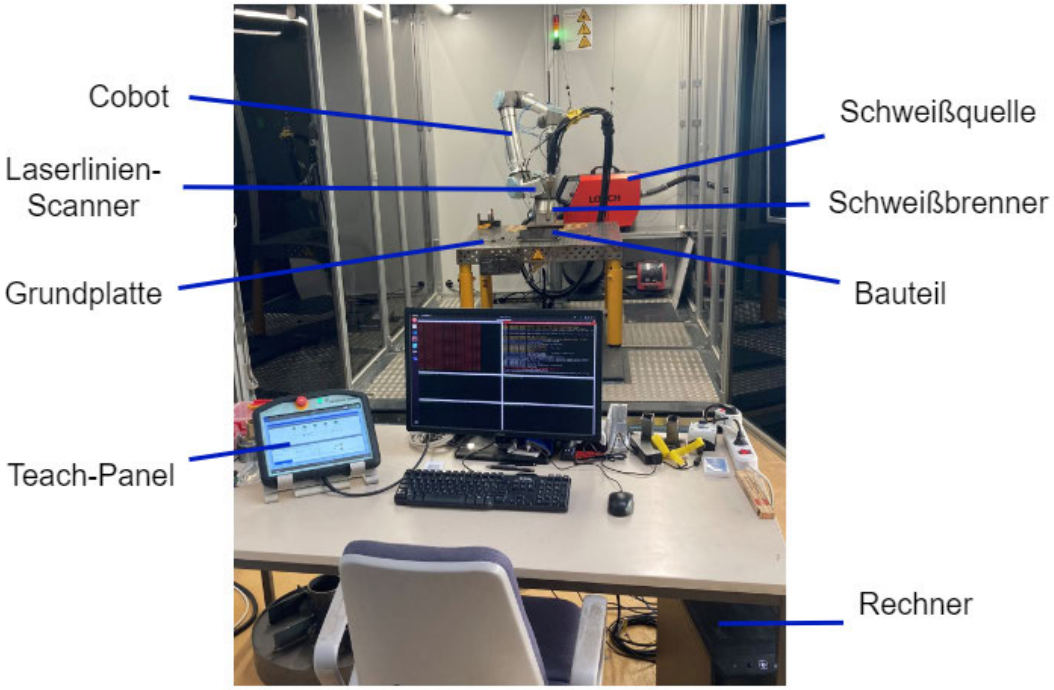


Abbildung 2.4: Der verwendete Schweißroboter [24]

2.2.1 Systemarchitektur

Die Abbildung 2.5 zeigt die Systemarchitektur der aktuellen Automatisierungslösung, die sich in drei Hauptteile gliedert: **Cobot**, **Sensor-Add-On** und **Schweißausrüstung**.

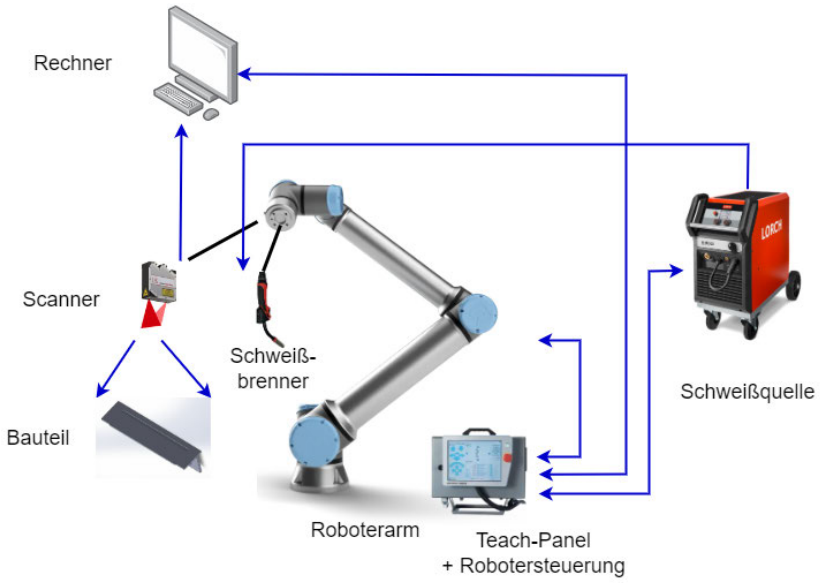


Abbildung 2.5: Systemarchitektur

Industrieroboter:

In diesem Projekt wird der Industrieroboter *UR10e* von *Universal Robots* eingesetzt. Er ist der zentrale Teil des Systems und umfasst einen 6-achsigen Roboterarm, eine Robotersteuerung sowie ein Teach-Panel. Die Programmierung und Steuerung des Roboterarms erfolgen über das Teach-Panel. Im Rahmen dieses Projekts wird das Teach-Panel durch eine Schnittstelle namens *URCap* erweitert. Über *URCap* kann der Benutzer die automatisierte Schweißlösung von *ProcessIt* aufrufen und Parameter wie Schweißgeschwindigkeit, Nahtlänge, Schweißwinkel und die Länge der Drahtelektrode (engl. *stick out*) einstellen. Zudem ermöglicht *URCap* die Anzeige des Systemstatus und informiert den Benutzer über Fehler, die während des Schweißprozesses auftreten.

Sensor-Add-On:

Das Sensor-Add-On bildet das Herzstück zur Steigerung der Intelligenz des Roboters und besteht aus einem Laserlinien-Scanner sowie einem Computer. Der am Roboterarm mon-

tierte *Micro-Epsilon scanCONTROL 3000-100* Laserlinien-Scanner ist für die Erfassung der Oberflächeninformationen des Bauteils über eine rote Laserlinie (x-Achse) zuständig. Die grüne Laserlinie stellt die y-Achse des Koordinatensystems des Scanners dar (siehe Abbildung 2.4). Er kann bis zu 2048 Datenpunkte pro Profil oder mit einer maximalen Profilfrequenz von 10.000 Hz erfassen [27]. Die gesammelten Daten werden zur weiteren Verarbeitung an den Rechner übertragen. Dieser Rechner übernimmt die zentrale Steuerungsfunktion, analysiert die vom Scanner empfangenen Informationen und berechnet die optimale Schweißbahn.

Schweißausrüstung:

Die Schweißausrüstung umfasst einen direkt am Roboterarm montierten Schweißbrenner, eine Schweißmaschine sowie eine Schlauchmontage und zusätzliche Schutzausrüstung. Zum Einsatz kommt das Metall-Aktivgas-Schweißen (MAG). Die Koordination des Schweißprozesses erfolgt dabei in Echtzeit und wird direkt vom Rechner gesteuert.

2.2.2 Softwarearchitektur

Die Abbildung 2.7 zeigt den Kern der Softwarearchitektur von *ProcessIt*, die auf **ROS-Noetic** und **MoveIt** basiert. Ergänzend dazu liefert die Abbildung 2.6 die Legende, die die verschiedenen Komponenten und deren Kommunikation innerhalb der Softwarearchitektur erläutert.

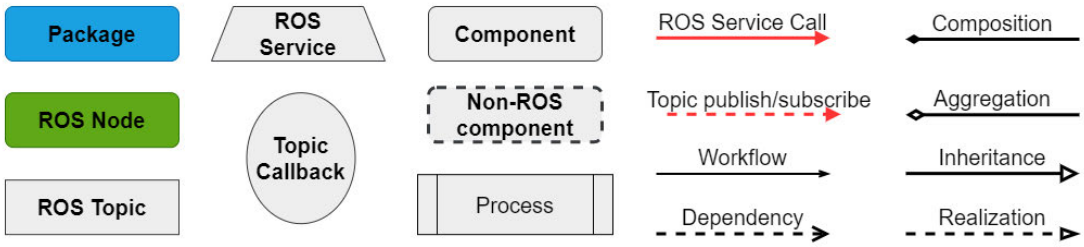


Abbildung 2.6: Legende der Softwarearchitektur

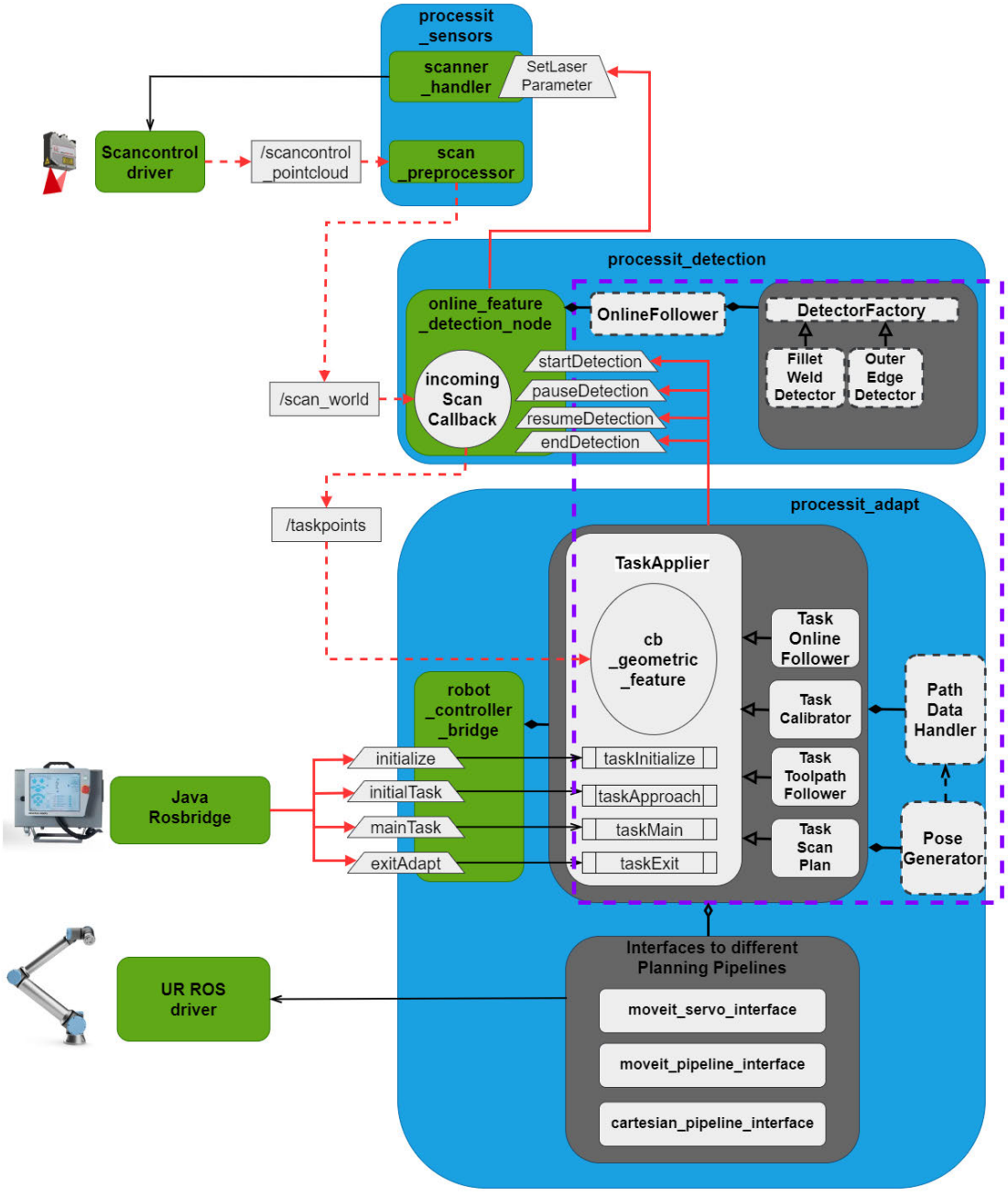


Abbildung 2.7: Softwarearchitektur

Aus einer übergeordneten Perspektive gibt es 3 Kernpakete: `processit_sensors`, `processit_detection` und `processit_adapt`.

2.2.2.a `processit_sensors`

Dieses Paket übernimmt die Erfassung und Vorverarbeitung von Rohdaten (`scan_preprocessor`) sowie die Steuerung (`scanner_handler`) des Scanners.

- `scan_preprocessor`: Die Rohdaten des Scanners werden von diesem Knoten verarbeitet, wobei Rauschen entfernt und die aktuelle Sensorpose (Position und Orientierung) hinzugefügt wird. Anschließend werden die bereinigten und transformierten Daten auf dem Thema `/scan_world` veröffentlicht.
- `scanner_handler`: Dieser Knoten stellt einen Dienst bereit, über den die Parameter, insbesondere die Leistung des Lasers vom Scanner angepasst werden können.

2.2.2.b `processit_detection`

Die Funktion dieses Pakets besteht darin, die Merkmale der Schweißnaht zu erkennen und anschließend die erforderlichen Zielpunkte (*task or target points*) für die Schweißbahn zu generieren. Diese Funktion wird über den Knoten `online_feature_detection_node` realisiert.

Der Knoten verfügt über die Callback-Methode `incomingScanCallback`, die immer dann ausgeführt wird, wenn eine neue Nachricht zum Thema `/scan_world` veröffentlicht wird. Der Ausführungsstatus dieser Methode wird durch vier Dienste gesteuert: `startDetection`, `pauseDetection`, `resumeDetection` und `endDetection`. Durch den Aufruf des Dienstes `startDetection` wird die Geometrie der zu erkennenden Schweißnaht (z.B. Kehlnaht oder Ecknaht) spezifiziert und mitgesendet. Daraufhin wird eine entsprechende Version vom `Detektor` initialisiert.

Der `Detektor` übernimmt die Erkennung von Merkmalen und die Generierung von Zielpunkten auf Basis der vom `OnlineFollower` bereitgestellten Daten. Das `OnlineFollower`-Objekt ist für das Sammeln jeder einzelnen Scandatenzeile verantwortlich. Sobald ausreichend Datenzeilen erfasst werden, aktiviert das Objekt die Erkennungsmethode vom `Detektor`, speichert die Ergebnisse und veröffentlicht die Zielpunkte über das Thema `/taskpoints`.

2.2.2.c `processit_adapt`

Dieses Paket kümmert sich um die Steuerung des Schweißprozesses, der am Knoten `robot_controller_bridge` durchgeführt wird, einschließlich: Kommunikation mit Teach-Panel; Kontrolle des Erkennungsstatus; Sammeln der Zielpunkte und Planen der Trajektorie für TCP. Dieser Knoten ist das Zentrum der Softwarearchitektur und die Brücke zwischen Teach-Panel, Merkmalserkennung und Robotersteuerung.

Über *Java Rosbridge* kann Teach-Panel mit `robot_controller_bridge` über 4 Dienste kommunizieren. Diese 4 Dienste entsprechen wiederum 4 Stufen oder 4 Aufgaben des Schweißprozesses. Jede Aufgabe wird spezifisch in einer Instanz der bestimmten `TaskApplier`-Spezialisierung ausgeführt. Basierend auf einem Factory-Method-ähnlichen Design-Pattern wird durch die Angaben der Geometrie der Schweißnaht vom Teach-Panel die spezialisierte `TaskApplier`-Instanz aus der Basisklasse initialisiert. `TaskApplier` ist für den Aufruf der Dienste von `online_feature_detection_node` in jeder Aufgabe verantwortlich.

Ähnlich wie der Knoten `online_feature_detection_node` verfügt der Knoten `robot_controller_bridge` über eine Callback-Methode `cb_geometric_feature` in `TaskApplier`, die Nachrichten aus dem Thema `/taskpoints` verarbeitet. Die empfangenen Zielpunkte werden bei `PathDataHandler` gesammelt, geglättet und interpoliert. Basierend auf den Daten von `PathDataHandler` berechnet `PoseGenerator` die nächsten Zielposen für TCP. Von dort aus wird die Planung der Trajektorie durch die spezielle Pipeline von **MoveIt** ausgeführt, bevor sie an den *UR ROS-Treiber* gesendet wird, um den Roboter in der realen Umgebung zu bewegen.

2.2.3 Ablauf des Schweißprozesses

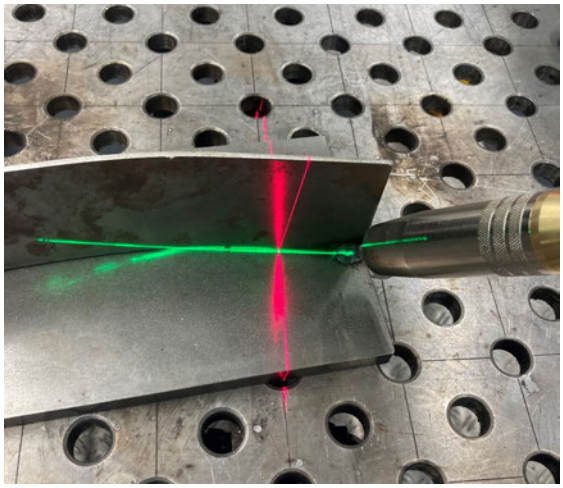


Abbildung 2.8: Positionierung des TCPs [24]

Am Anfang sollte der TCP des Roboters manuell an der Startposition der Schweißnaht positioniert werden. Dabei wird die rote Laserlinie des Scanners senkrecht zur Schweißnaht ausgerichtet, während die grüne Laserlinie entlang der Schweißnaht verläuft, wie in Abbildung 2.8 dargestellt. Die rote Laserlinie dient dabei zur Erfassung der Oberfläche und die grüne Laserlinie wird zur optischen Ausrichtung der Bewegung des TCP genutzt. Nach der Festlegung der Parameter über das Teach-Panel erfolgt die automatische Erkennung und Überwachung der Schweißnaht. Der automatisierte Schweißprozess des Projektes *ProcessIt* ist in 4 Aufgaben unterteilt, nämlich:

1. `taskInitialize` (`initialize`)
    - Der spezifische `TaskApplier`-Instanz wird initialisiert.
    - Die vom Benutzer festgelegte Parameter für den Schweißprozess werden initialisiert, konvertiert und angepasst.
2. `taskApproach` (`initalTask`): initiales Anfahren
    - Der Endeffektor (TCP) und der Scanner bewegen sich entlang der y-Achse des Scanner-Koordinatensystems (die grüne Laserlinie) über eine kurze Strecke von **5 cm**.
    - Die erfassten Daten werden ausgewertet, um die Schweißnaht zu erkennen.

- Die ersten Zielpunkte werden identifiziert und erstellt.
- Der TCP bewegt sich zum ersten Zielpunkt.

3. `taskMain` (`mainTask`): Online-Verfolgung
   - Diese Aufgaben werden in einer Schleife ausgeführt.
   - Der Schweißvorgang wird durchgeführt.
   - Der TCP des Schweißbrenners bewegt sich zu zuvor erstellten Zielpunkten.
   - Parallel dazu sammelt der Scanner kontinuierlich weitere Daten vom Bauteil, um die Schweißnaht zu verfolgen und neue Schweißpunkte zu generieren.
   - Die Aufgaben endet, wenn entweder das Ende der Schweißnaht erreicht ist oder die erforderliche Länge geschweißt wird.
4. `taskExit` (`exitAdapt`)
   - Das Schweißen sowie die Datenerfassung werden beendet.
   - Die Daten werden bereinigt.

Basierend auf der Systemarchitektur, der Softwarearchitektur und dem Ablauf des Schweißprozesses, die oben beschrieben werden, wird in dieser Arbeit eine neue Lösung für die Schlitzerkennung und das Schweißen der Schlitz-Zapfen-Verbindung entwickelt. Daher konzentriert sich diese Arbeit auf die Entwicklung spezialisierter Versionen von `TaskApplier` und `Detektor` für die Schlitz-Zapfen-Verbindung, wie in der violetten gestrichelten Box in Abbildung 2.7 dargestellt.

3 Stand der Technik

Dieses Kapitel widmet sich der Untersuchung der bestehenden Algorithmen zur Verarbeitung von Punktwolken, mit dem Ziel, Lösungen zur Erkennung von Schlitzen in Punktwolken zu finden.

3.1 Punktwolkenverarbeitung

Eine **Punktwolke** (*point cloud*) ist eine Sammlung von Datenpunkten, die Informationen über die räumliche oder physische Oberfläche eines Objekts liefern. Diese Punkte werden von Sensoren, wie LiDAR (*Light Detection and Ranging*), oder Scannern, wie dem in Abschnitt 2.1.3 erwähnten Laserlinien-Scanner, oder anderen Technologien, erfasst. In einem 3D-Koordinatensystem hat jeder Punkt einen eindeutigen X-, Y- und Z-Koordinatenwert. Der Abstand zwischen den einzelnen Punkten in der Wolke oder die Dichte hängt von der verwendeten Erfassungstechnologie ab. Punktwolken werden zunehmend in Bereichen eingesetzt, in denen die Erfassung und Verarbeitung räumlicher Informationen erforderlich ist, z. B. in der Automobilindustrie, Geologie, Architektur und insbesondere in der Robotik.

Punktwolkenverarbeitung umfasst eine Reihe von Methoden und Algorithmen, um rohe Punktwolken zu analysieren, zu bereinigen und zu verarbeiten. Ziel ist es, nützliche Informationen aus der rohen Punktwolke zu extrahieren, z.B. Objekterkennung, Merkmalserkennung usw., die für die Wahrnehmung, Navigation oder Qualitätskontrolle verwendet werden können. Bibliotheken wie *Point Cloud Library* [41] oder *Open3D* [60] bieten Verarbeitungsfunktionen für Punktwolken.

Diese Arbeit konzentriert sich darauf, die Schlitzerkennung auf der Grundlage eines kleinen Satzes lokaler Datenpunkte aus jeder Einzelaufnahme durchzuführen, anstatt die Punktwolke aus dem vollständigen Scan des Bauteils zu verarbeiten. Aus diesem Grund

werden die Algorithmen zur Registrierung von Punktwolken in dieser Arbeit nicht behandelt.

3.1.1 Down-Sampling

Rohe Punktwolken haben oft eine große Anzahl von Datenpunkten mit hoher Dichte und enthalten eine Menge redundanter Informationen. Dies beeinträchtigt die Geschwindigkeit und erhöht die Kosten der Berechnung, wenn die Punktwolke direkt verarbeitet wird. Daher sollte bei der Vorverarbeitung ein Down-Sampling durchgeführt werden, um die Anzahl der Punkte zu verringern sowie das Rauschen zu reduzieren. Dabei müssen jedoch die wichtigen Informationen über die Struktur und die Merkmale erhalten bleiben.

Voxel-Down-Sampling ist eine verbreitete Methode, bei der nahe beieinander liegende Datenpunkte in kleine Voxel (Volumen-Pixel oder 3D-Würfel) unterteilt werden. In der Regel wird der berechnete Mittelpunkt in jedem Voxel beibehalten, während die anderen Punkte darin entfernt werden [60] [41]. Durch die Wahl einer bestimmten Größe der einzelnen Voxel wird die Punktdichte in der Punktwolke gleichmäßig verteilt, wobei die Informationen der Objektoberfläche erhalten bleiben. Jedoch wird die Methode Voxel-Down-Sampling leicht durch Ausreißer in den Randbereichen der Punktwolke gestört, was die Genauigkeit der Konstruktion von der Oberfläche bis zu einem gewissen Grad verringert [37].

3.1.2 Clustering

Unter **Clustering** versteht man ein Verfahren zur Klassifizierung oder Gruppierung von Datenpunkten in verschiedene Gruppen bzw. Clustern, so dass Punkte in derselben Gruppe ähnlich sind und die Gruppen nach der Klassifizierung unterschiedliche Merkmale aufweisen. In der Robotik bzw. bei der Verarbeitung von Punktwolken beinhaltet das Clustering die Gruppierung von Punkten, die von Sensoren oder Scannern erfasst werden, in Cluster basierend auf ihrer räumlichen Nähe oder Ähnlichkeit. Dieses Verfahren ist wichtig, um Punkte zu identifizieren, die zu derselben Komponente oder demselben Bauteil gehören. In dieser Arbeit ist das Clustering ein notwendiger Schritt, um Punkte zu entfernen, die nicht zu dem zu schweißenden Bauteil gehören, wie z.B. Punkte auf der Grundplatte (*ground plane*) oder der Vorrichtung.

Euklidisches Clustering

Euklidisches Clustering (*Euclidean Clustering*) ist eine einfache Clustering-Methode für Punktwolken. Auf der Grundlage des euklidischen Abstands zwischen Punkten werden bei dieser Methode Datenpunkte innerhalb eines bestimmten Radius (Schwellenwert) gruppiert. Die Methode beginnt mit der Auswahl eines Startpunkts und fügt iterativ benachbarte Punkte hinzu, die das Kriterium des Abstands erfüllen, und bildet einen Cluster. Sobald der Cluster vollständig ist, wird der Prozess für nicht besuchte Punkte wiederholt, bis alle Punkte verarbeitet sind. In [45, 58, 57] wird das euklidische Clustering verwendet, um Punkte zu gruppieren, die zu verschiedenen Oberflächen gehören. Basierend auf diesen Clustern kann das Merkmal oder die Schweißnaht extrahiert werden.

Das euklidische Clustering ist eine einfache, intuitive Methode mit geringem Rechenaufwand. Diese Methode eignet sich sehr gut für Punktwolken mit klaren Abständen zwischen Clustern. Weist die Punkte jedoch eine komplexere Struktur auf, ist es schwierig, den Abstandschwellenwert zu bestimmen. Außerdem wird die Methode während des Clustering-Prozesses leicht durch Rauschen beeinträchtigt. Diese Methode kann jedoch für das Clustering mit einfachen Punktmengen wie Linien- oder Flächensegmenten verwendet werden.

DBSCAN

DBSCAN-Clustering (*Density-Based Spatial Clustering of Applications with Noise*) ist eine Clustering-Methode, die von M. Ester, H.-P. Kriegel, J. Sander und X. Xu in [13] veröffentlicht wurde. Wie im Namen steht, arbeitet diese Methode auf der Grundlage der Dichte der Wolke, wobei die Grundidee darin besteht, dass jeder Punkt eines Clusters in einem bestimmten Radius mindestens eine Mindestanzahl von Nachbarpunkten haben muss. Es gibt also zwei Parameter, die vordefiniert werden:

1. *eps*: Maximalentfernung, innerhalb derer zwei Punkte zum selben Cluster gehören
2. *min_samples*: Minimale Anzahl der Nachbarpunkte

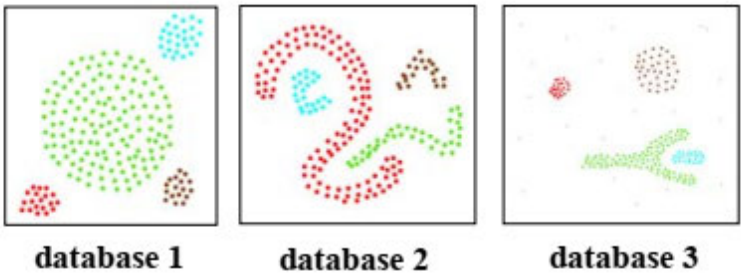


Abbildung 3.1: Durch DBSCAN entdeckte Cluster [13]

Diese Methode schlägt auch eine Lösung für die Punkte am Rand vor, bei denen die lokale Punktdichte niedriger ist als im Inneren des Clusters. Deshalb kann diese Methode Cluster beliebiger Form erkennen, wie in Abbildung 3.1 dargestellt. Durch die Auswahl geeigneter Parameter (*eps* und *min_ samples*) können Rauschpunkte, die sich nicht für die Bildung von Clustern eignen, erkannt und entfernt werden. Allerdings ist die Auswahl dieser Parameter anspruchsvoll. Außerdem erfordert diese Methode bei großen Punktwolken einen relativ hohen Rechenaufwand.

K-Means-Clustering

K-Means-Clustering ist eine klassische Methode, die weit untersucht und in der Praxis verwendet wird. Die Grundidee dieser partitionierenden Methode besteht darin, die Anzahl der aufzuteilenden Cluster (K) im Voraus festzulegen. Iterativ wird jeder Punkt dem nächstgelegenen Mittelpunkt des Clusters zugeordnet und der Mittelpunkt wird dann basierend auf dem Mittelwert (*Means*) der zugewiesenen Punkte aktualisiert. Diese Methode wird in [6] zusammen mit Normalenvektoren verwendet, um Kantenpunkte in der Punktwolke zu bestimmen.

Obwohl das K-Means-Clustering einfach und leicht zu implementieren ist und nur einen geringen Rechenaufwand erfordert, hat diese Methode bei der Verarbeitung räumlicher Punktwolken viele Einschränkungen. Bei dieser Methode muss vorab eine bestimmte Anzahl von Clustern vordefiniert werden, und sie hängt von anfänglichen Mittelpunkten ab, wodurch die Ergebnisse von Clustern instabil werden können [49]. Darüber hinaus ist diese Methode auch empfindlich gegenüber Rauschen und Ausreißern. Bei Punktwolken mit komplexen Strukturen, wie z.B. Clustern mit ungleicher Anzahl von Punkten, Clustern mit unkonvexen (konkaven) Formen, ist diese Methode wirklich unwirksam.

3.1.3 Anpassung von Geraden und Ebenen (*Line and Plane Fitting*)

Im 3D-Raum sind Linien und Ebenen die grundlegenden Elemente, die zur Beschreibung räumlicher Beziehungen und Strukturen verwendet werden.
In kartesischen Koordinaten kann eine Linie in parametrischer Form dargestellt werden:

$$x = x_0 + at; y = y_0 + bt; z = z_0 + ct$$

wobei (x_0, y_0, z_0) ein Punkt auf der Linie, (a, b, c) der Richtungsvektor und t der skalare Parameter ist.

Eine Ebene hingegen wird durch die Gleichung beschrieben:

$$ax + by + cz + d = 0$$

wobei (a,b,c) der Normalenvektor der Ebene und d der senkrechte Abstand vom Ursprung ist.

Die Anpassung von Modell, insbesondere Linien bzw. Ebenen, in 3D-Punktwolken ist eine grundlegende Aufgabe in der Robotik und der Computer-Vision, da sie für die Modellierung und die Analyse der geometrischen Struktur der Umgebung wesentlich ist. Durch die Anpassung von Linien können lineare Strukturen wie gerade Kanten oder Ränder extrahiert werden, insbesondere bei der Verarbeitung von Datenpunkten aus dem Laserlinien-Scanner. Daneben ermöglicht die Anpassung von Ebenen die Extraktion von flachen Oberflächen wie Wänden, Böden oder Tischplatten.

Methode der kleinsten Quadrate

Die Methode der kleinsten Quadrate (*method of least squares* oder lediglich *least squares*) ist eine verbreitete und einfache Methode zur Anpassung von Modellen an Datenpunkte. Diese Methode basiert auf der Minimierung der Summe der quadratischen Abstände der Datenpunkte zur gewünschten geometrischen Einheit. Daher ist diese Methode nicht nur auf gerade Linien oder Ebenen, sondern auch auf Kurven oder gekrümmte Flächen anwendbar.

Diese Methode bietet den Vorteil, dass sie einfach zu implementieren ist und eine hohe Rechengeschwindigkeit aufweist, was insbesondere bei einer großen Punktanzahl von Bedeutung ist. Allerdings besteht ein wesentlicher Nachteil in ihrer hohen Empfindlichkeit gegenüber Ausreißern, die häufig in der Punktwolke auftauchen. Der Grund dafür ist,

dass sie an allen Punkten, auch an den Ausreißern, optimal angepasst ist. Schon wenige Ausreißer können dazu führen, dass die resultierende Gleichung erheblich vom erwarteten Ergebnis abweicht. Zudem ist die Methode auch unzuverlässig, wenn die Daten ungleichmäßig verteilt sind.

Random-Sample-Consensus-Methode

Die RANSAC-Methode (*random sample consensus*) [15] ist eine der verbreitetsten Methoden zur Anpassung von Modellen an Daten mit Behandlung von Ausreißern (engl. *outliers*). Die Grundidee von RANSAC ist folgende: Zunächst wird eine minimale zufällige Teilmenge aus der Punktwolke gezogen und daraus ein geeignetes Modell (Linie oder Ebene) geschätzt. Anschließend werden die Abstände zwischen den verbleibenden Punkten in der Punktwolke und dem Modell berechnet. Diese Abstände werden mit einem Schwellenwert verglichen. Punkte, die innerhalb des Schwellenwerts liegen, werden als Inliers klassifiziert, während Punkte außerhalb dieses Bereichs als Outliers betrachtet werden. Diese beiden Schritte werden in einer bestimmten Anzahl von Iterationen wiederholt, um das am besten angepasste Modell mit der größten Anzahl von Inliers zu erhalten.

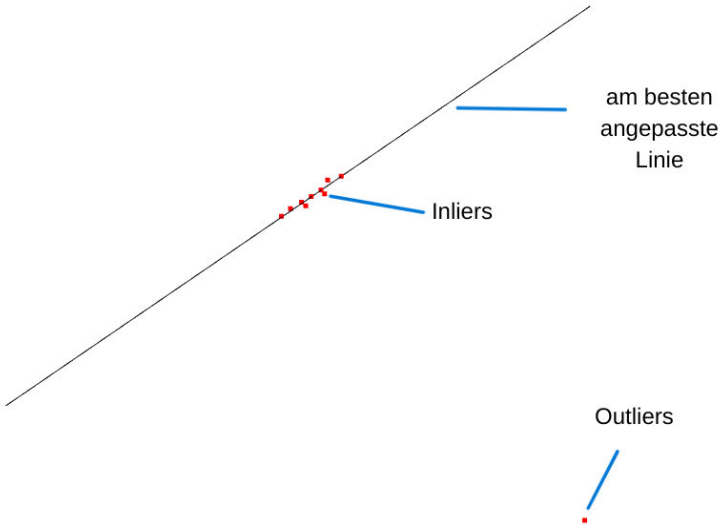


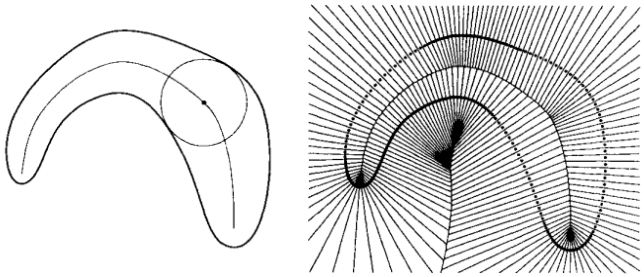
Abbildung 3.2: Anpassung einer Linie mit RANSAC

Die Funktionsweise von RANSAC zeigt, dass diese Methode bei der Behandlung von Ausreißern bzw. Rauschen sehr effektiv ist. RANSAC kann ein Modell mit hoher Genauigkeit und hoher Flexibilität finden, wenn die Punktwolke eine erhebliche Anzahl von Ausreißern aufweist und komplex ist. Die Einschränkung von RANSAC besteht jedoch darin, dass geeignete Parameter ausgewählt werden müssen und es schwierig ist, sie für jede Punktwolke anzupassen. Außerdem basiert diese Methode auf einer Zufallsauswahl, die viele Iterationen erfordern kann, um Stabilität zu erreichen, was zu einer langen Berechnungszeit führen kann. Wenn jedoch die Anzahl der Iterationen begrenzt ist, kann diese Methode möglicherweise nicht das optimale Modell ermitteln.

3.1.4 Mediale Achse

Das Hauptziel dieser Bachelorarbeit ist die Erkennung des Schlitzes aus der Punktwolke und die Generierung der Trajektorie für den TCP entlang der Mittellinie des Schlitzes. Dabei kann die Mittellinie entlang der Länge des Schlitzes als Symmetrieachse oder **Mediale Achse** betrachtet werden.

Die Mittelachse wurde ursprünglich von Blum [3] beschrieben. Die mediale Achse einer ebenen Form ist der Ort (engl. *locus*) der Mittelpunkte einer Reihe von Scheiben (2D) oder Kugeln (3D), die maximal in die Form passen, wie in Abbildung 3.3a dargestellt. Und die mediale Achsen-Transformation ist die mediale Achse zusammen mit den entsprechenden Radiuswerten. Punkte auf der medialen Achse, d. h. die Mittelpunkte solcher Scheiben/Kugeln, werden als die medialen Punkte (oder Punkte der medialen Achse) der Form bezeichnet [7].



(a) (b)

Abbildung 3.3: Extraktion der medialen Achse [44]
a) aus einer Scheibe, der die Ränder der Form an zwei oder mehr Punkten berührt
b) anhand des Voronoi-Diagramms

Mediale Achse aus Voronoi-Diagramm

Eine Methode zur Erstellung der medialen Achse ist die Nutzung von **Voronoi-Diagrammen**, die sich dual zur Delaunay-Triangulierung verhalten [44, 52], wie in Abbildung 3.3b dargestellt. Hierbei wird um jeden Punkt der Punktwolken eine Voronoi-Zelle erzeugt. Voronoi-Kanten, die durch weit auseinanderliegende Randpunkte gebildet werden, bleiben erhalten, da diese wahrscheinlich tief in der Form liegen und einen Teil der medialen Achse bilden. Diese Methode bietet eine präzise Darstellung der medialen Achse, ist jedoch rechenintensiv, da die Berechnung von 3D-Voronoi-Diagrammen komplex und speicherintensiv ist.

Mediale Achse durch Reduzierung des Kugelradius

Ma et al. schlagen eine iterative Methode zur Approximation der medialen Achse in [26] vor. Diese Methode basiert auf den nächsten Nachbarn und den Normalen der Randpunkte. Für jeden Randpunkt wird eine große Tangentialkugel konstruiert, deren Zentrum entlang der Normale des Randpunkts liegt und deren Radius auf einem ausreichend großen Anfangswert basiert. Die Kugel mit dem Radius wird iterativ verkleinert, während das Zentrum immer entlang der Normale liegt. Sobald die Kugel einen weiteren Randpunkt berührt und keine anderen Punkte innerhalb der Kugel liegen, wird das Zentrum als ein Punkt der medialen Achse markiert. Diese Methode ist nicht nur präzise und rechnerisch effizient, sondern gilt auch als die einfachste und schnellste Methode zur Oberflächenskelettierung [51].

3.2 Merkmalserkennung in der Punktwolke

Die Erkennung von Merkmalen stellt einen wesentlichen Schritt in der Analyse von Punktwolken dar, mit dem Ziel, geometrische Strukturen und prägnante Merkmale der Umgebung zu identifizieren. Hierbei werden in der Regel Daten zu Rändern, Kanten, Ecken, Linien oder Flächen aus der aufgenommenen Punktwolke extrahiert. Diese Merkmale dienen als Basis für die Objekterkennung, das Verständnis von Szenen und die Bahnplanung des Roboters.

3.2.1 Nächste-Nachbarn-Analyse

Einer der verbreiteten Ansätze zur Erkennung von Merkmalen in Punktwolken ist die Analyse der nächsten Nachbarn, d. h. die Analyse der Informationen über die nächsten Nachbarn für jeden Abfragepunkt. Diese Informationen können die Anzahl der Nachbarn in der Umgebung, große Winkellücke oder Informationen über die Dichte der Punktverteilung sein. Anhand dieser Informationen können die geometrischen Merkmale des Abfragepunkts bestimmt werden.

3.2.1.a k-d-Baum (*k-d tree*)

Bei einer großen Anzahl von Punkten ist die naive Suche nach nächsten Nachbarn durch den Vergleich des euklidischen Abstands zu jedem Abfragepunkt rechenintensiv und ineffizient. Der k-d-Baum (k-dimensionaler Baum, wobei k die Anzahl der Dimensionen ist) gehört zu den verbreitetsten Methoden, um die Anzahl der notwendigen Abstandsberechnungen in niedrig-dimensionalen Räumen ($k < 20$) zu minimieren. Durch die rekursive Partitionierung des Raums entlang abwechselnder achsenparalleler Hyperebenen erzeugt der k-d-Baum eine hierarchische Struktur, die schnelle und effiziente räumliche Abfragen ermöglicht.

Die Konstruktion eines k-d-Baums erfolgt zügig, da die Partitionierung ausschließlich entlang der Datenachsen stattfindet und keine vollständigen k-dimensionalen Abstandsberechnungen erforderlich sind. Nach der Erstellung kann ein der nächsten Nachbarn eines Abfragepunkts mit lediglich $O(logN)$ Abstandsberechnungen gefunden werden [34]. Die Raumaufteilung erlaubt es dem Algorithmus, große Bereiche des Suchbaums, die den nächsten Nachbarn nicht enthalten können, rasch auszuschließen, wodurch der Rechneraufwand erheblich reduziert wird. Während des Suchprozesses durchläuft der Algorithmus den Baum und trifft an jedem Knoten binäre Entscheidungen auf der Grundlage der Teilungsdimension und des Wertes, wodurch der Suchraum logarithmisch eingegrenzt wird, anstatt jeden Punkt des Datensatzes zu untersuchen.

Nach der Erstellung des k-d-Baums kann die Suche nach den nächsten Nachbarn als k-nächste Nachbarn (*k-nearest neighbors* - k-NN) oder als Nachbarn im festen Radius (*Fixed-radius near neighbors* - FRNN) durchgeführt werden. Bei der k-NN-Suche wird für jeden Abfragepunkt eine feste Anzahl von k Nachbarn bestimmt, unabhängig von der Entfernung. Dieser Ansatz ist besonders nützlich für Fälle, die eine konstante Anzahl

von Referenzpunkten erfordern, wie etwa die Interpolation oder Dichteschätzungen. Im Gegensatz dazu sucht der FRNN-Absatz nach allen Punkten innerhalb eines festgelegten Radius um den Abfragepunkt. Dies ist vorteilhaft für Fälle, in denen lokale Dichte oder die Erkennung von Strukturen innerhalb bestimmter Entfernungsgrenzen relevant ist. Die Wahl von k oder des Radius für die Analyse der Nachbarschaft muss sorgfältig festgelegt werden. Wenn k oder der Radius zu klein gewählt werden, werden die Informationen über die lokale Region nicht genau berechnet. Ist der Parameter jedoch zu groß, beeinträchtigt dies die Geschwindigkeit der Berechnungen.

3.2.1.b Punktdichte

In [57] schlagen Zhang et al. eine einfache Methode zur Extraktion von Kanten- bzw. Randpunkten des Werkstücks auf der Grundlage der Punktdichte vor. Nach der Erfassung der Punktwolke durch die Kamera werden die zur Ebene gehörenden Punkte durch Filterung auf Grundlage der minimalen Anzahl der nächsten Nachbarn identifiziert. Anschließend wird der Hintergrund mithilfe des euklidischen Clustering entfernt. Die Erkennung der Kanten- bzw. Randpunkte erfolgt durch den Vergleich der tatsächlichen Anzahl von Nachbarpunkten innerhalb eines festen Radius (FRNN) mit der theoretisch erwarteten Anzahl.

Die von Zhang et al. vorgeschlagene Methode zur Randpunktextraktion zeichnet sich durch die Einfachheit und Geschwindigkeit aus. Die Methode erfordert nur einen geringen Rechenaufwand. Allerdings ist die Methode stark von der Wahl des Radius und des Schwellenwerts (der theoretisch erwarteten Anzahl) abhängig. In Punktwolken mit ungleichmäßiger Verteilung erreicht die Methode ihre Grenzen, da Schwankungen in der Dichte potenziell zu ungenauen Ergebnissen führen können. Zudem neigt die Methode dazu, nicht nur die tatsächlichen Randpunkte zu identifizieren, sondern auch benachbarte Punkte fälschlicherweise als Randpunkte zu klassifizieren, was die Präzision der Extraktion beeinträchtigen kann.

3.2.1.c Entfernung von Punkt zum Schwerpunkt

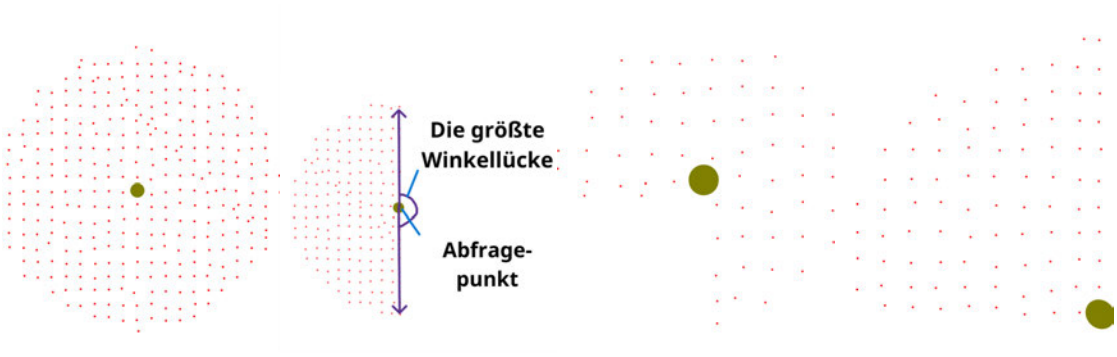
Eine Methode zur Erkennung von Randpunkten basiert auf der Analyse des Abstands zwischen einem Abfragepunkt und dem Schwerpunkt seiner benachbarten Punkte [1, 10, 32, 48]. Der Kern dieser Methode liegt in der Beobachtung, dass die Nachbarpunkte eines Randpunkts überwiegend auf einer Seite der Randlinie liegen, wodurch der Schwerpunkt

merklich vom Abfragepunkt abweicht. Im Gegensatz dazu sind die Nachbarn eines Punkts innerhalb einer Ebene gleichmäßig um ihn verteilt, sodass der Schwerpunkt nahe am Abfragepunkt liegt.

Die oben beschriebene Methode zur Erkennung von Randpunkten ist vorteilhaft durch die Einfachheit und die hohe Geschwindigkeit der Berechnung, da lediglich der Schwerpunkt der benachbarten Punkte berechnet wird. Dies macht die Methode rechenleicht und effizient. Allerdings weist sie Einschränkungen auf, da die Genauigkeit stark von der Wahl der Nachbarschaft (die Anzahl k oder Radius) abhängt. Die Festlegung eines geeigneten Schwellenwerts zur Unterscheidung zwischen Randpunkten und Punkten der Ebene gestaltet sich schwierig. Dies kann dazu führen, dass die benachbarten Punkte fälschlicherweise als Randpunkte erkannt werden. Zudem ist die Methode bei der Erkennung von konkaven Ecken weniger effektiv, da die Nachbarpunkte auch um den Punkt verteilt sind.

3.2.1.d Die größte Winkellücke

Die in [6, 31, 32, 49] vorgestellte Methode zur Erkennung von Randpunkten basiert auf der Analyse der größten Winkellücke zwischen einem Abfragepunkt und seinen Nachbarn. Zunächst werden die Nachbarn des Abfragepunkts mithilfe der k-NN oder FRNN bestimmt. Anschließend wird eine lokale Ebene und die Nachbarn angepasst, auf die sowohl der Abfragepunkt als auch dessen Nachbarn projiziert werden. Zur Bestimmung der Randpunkte wird die größte Winkellücke zwischen den Vektoren vom projizierten Abfragepunkt zu den projizierten Nachbarpunkten berechnet (Abbildung 3.4). Diese Winkellücke wird schließlich mit einem festgelegten Schwellenwert verglichen, um zu entscheiden, ob der Abfragepunkt als Randpunkt klassifiziert wird.



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

Abbildung 3.4: Verteilung der nächsten Nachbarn von
a) eines inneren Punkts der Ebene
b) eines Randpunkts
c) eines konkaven Eckpunkts
d) eines konvexen Eckpunkts

Diese Methode bietet den Vorteil einer hohen Genauigkeit bei der Extraktion von Randpunkten, ohne dass benachbarte Punkte fälschlicherweise als Randpunkte erkannt werden. Zudem ermöglicht die Wahl eines geeigneten Schwellenwerts die zuverlässige Erkennung konkaver Ecken. Die Berechnung ist effizient und erfordert nur geringen Rechenaufwand.

Dennoch weist die Methode einige Einschränkungen auf. Sie ist empfindlich gegenüber Rauschen, das die Größe der Winkellücke verzerren und die Präzision der Erkennung beeinträchtigen kann. Darüber hinaus hat sie auch eine Schwäche mit anderen Methoden, die auf der Analyse der Nachbarschaft basieren: Die Wahl eines geeigneten k-Werts oder Suchradius ist entscheidend und hängt von der Punktdichte der Wolke ab.

3.2.1.e Abschätzung von Normalen

In [21, 9, 10, 20, 35, 50] wird eine Methode zur Merkmalserkennung vorgestellt, die auf der Abschätzung von Normalen basiert. Für jeden Abfragepunkt werden benachbarte Punkte - häufig mithilfe eines k-d-Baums - bestimmt. Anschließend wird eine Ebene an die lokale Region der benachbarten Punkte angepasst. Der Normalenvektor der angepassten Ebene wird als Normale des Abfragepunkts übernommen. Kantenpunkte werden durch den Vergleich der Ähnlichkeit der Normalen oder durch ein regionsbasiertes Wachstum

(*region growing*) identifiziert. Dabei sind Kantenpunkte die Schnittpunkte, an denen zwei unterschiedliche Regionen oder Ebenen aufeinandertreffen.

Die Methode der Abschätzung von Normalen bietet eine einfache und effektive Möglichkeit zur Erkennung von scharfen Kanten. Sie ist jedoch ungeeignet zur Extraktion von Randpunkten, wenn auf einer Seite keine Nachbarn vorhanden sind. In solchen Fällen entspricht die Normale des Randpunkts der Normale des inneren Punkts.
Rauschen in der Punktwolke kann die Abschätzung von Normalen verfälschen, was die Genauigkeit der Merkmalserkennung und Segmentierung beeinträchtigt. Die Qualität der Normalenabschätzung hängt stark von der Wahl der Nachbarschaft (k oder Radius) ab. Eine zu kleine Nachbarschaft führt zu instabilen Normalen, während eine zu große Nachbarschaft feine Details verwischen kann.

3.2.1.f Hauptkomponentenanalyse (*Principal Component Analysis*)

Hauptkomponentenanalyse (*Principal Component Analysis* - PCA) ist eine weit verbreitete Methode zur Erkennung von geometrischen Merkmalen in Punktwolken, wie in [21, 1, 2, 22, 39, 59] beschrieben. Sie wird verwendet, um lokale Strukturen zu analysieren, indem sie die Punktverteilung in einer Nachbarschaft untersucht und die Hauptachsen der Punktverteilung bestimmt.

Für jeden Punkt wird eine lokale Nachbarschaft definiert, möglicherweise durch einen k-d-Baum. Aus den benachbarten Punkten wird der Schwerpunkt bestimmt. Die Kovarianzmatrix (oder Korrelationsmatrix) wird berechnet, indem die Abweichungen der benachbarten Punkte vom Schwerpunkt in jeder Dimension multipliziert und gemittelt werden:

$$C = \frac{1}{N} \sum_{i=1}^{N} (p_i - \bar{p})(p_i - \bar{p})^T$$

Dabei ist p_i ein Punkt in der Nachbarschaft und $\bar{p}$ der Schwerpunkt der Nachbarn.

Aus der Kovarianzmatrix (oder Korrelationsmatrix) werden die Eigenwerte und Eigenvektoren ermittelt. Die Eigenvektoren geben die Hauptachsen der Punktverteilung an, während die Eigenwerte die Ausdehnung entlang dieser Achsen beschreiben. Basierend auf diesen Eigenwerten können lokale geometrische Werte wie Normale, Linearität, Planarität, Krümmung usw. berechnet werden. Daraus können Kanten, Falten, Flächenübergänge und Ecken extrahiert werden.

Ein Vorteil dieser Methode ist die Vielseitigkeit, lokale geometrische Merkmale effektiv zu erkennen. Darüber hinaus hat diese Methode die Robustheit gegenüber Rauschen und hängt nicht vollständig von der Punktdichte ab. Allerdings ist die Wahl der Schwellenwerte sowie die Auswahl der Nachbarschaft eine echte Herausforderung.

3.2.2 Bildverarbeitung

Ein indirekter Ansatz zur Merkmalsextraktion aus 3D-Punktwolken nutzt die Algorithmen der 2D-Bildverarbeitung [30, 47, 53]. Dabei wird die Punktwolke zunächst in ein 2D-Bild transformiert, in der Regel durch die Suche nach einer Ebene, die zur Punktwolke passt. Die Punkte aus der Punktwolke werden auf diese Ebene projiziert, wobei der Grauwert eines Pixels der Distanz des entsprechenden Punkts zur Ebene entspricht. Anschließend werden Bildverarbeitungsalgorithmen wie Sobel-, Canny- oder Laplace-Filter, gradientenbasierte Verfahren, morphologische Operationen oder Fourier-Transformationen eingesetzt, um geometrische Merkmale wie Kanten, Ecken und Konturen zu extrahieren. Im letzten Schritt werden die erkannten Konturen und Merkmale in den 3D-Raum zurückprojiziert, um die Strukturen in der Punktwolke wiederherzustellen.

Dieser indirekte Ansatz ist besonders vorteilhaft für einfache und flache Punktwolken. Ein großer Vorteil liegt in der Effizienz und Rechenleistung, da 2D-Bildverarbeitungsalgorithmen durch umfangreiche Forschung und zahlreiche optimierte Bibliotheken weit verbreitet und leicht zugänglich sind. Darüber hinaus ist die Verarbeitung in 2D rechenintensiv deutlich geringer als direkt in 3D.

Allerdings hat dieser Ansatz auch Einschränkungen. Während der Projektion zwischen 3D- und 2D-Räumen können feine Details und kleine geometrische Merkmale verloren gehen. Die Qualität der Ergebnisse hängt stark von der Wahl der Projektionsebene ab, die bei komplexen Geometrien wie gekrümmten oder sich überlagernden Strukturen schwer zu bestimmen ist. Zusätzlich beeinflusst die Auflösung des erzeugten 2D-Bildes die Präzision der extrahierten Merkmale. Auch ist die Methode empfindlich gegenüber Rauschen, das die Projektion verzerren und zu falschen Ergebnissen führen kann.

3.2.3 *Alpha-Shape*

Alpha-Shape ist eine Methode zum Generieren von Hüllen in Punktwolken, die erstmals von Edelsbrunner et al. in [11] für den 2D-Raum und in [12] für den 3D-Raum vorgestellt

wurde. Um ein Alpha-Shape zu erstellen, wird eine Scheibe (2D) oder Kugel (3D) mit einem Radius von $1/\alpha$ über die Punktwolke gerollt. Die Form entsteht durch Bereiche, die von dieser Scheibe oder Kugel nicht erreicht werden können - sowohl von außen als auch innerhalb der Punktwolke. Punkte, die von der rollenden Scheibe/Kugel berührt werden, ohne dass dabei andere Punkte der Punktwolke im Inneren der Scheibe/Kugel liegen, definieren die Grenze des Alpha-Shapes.

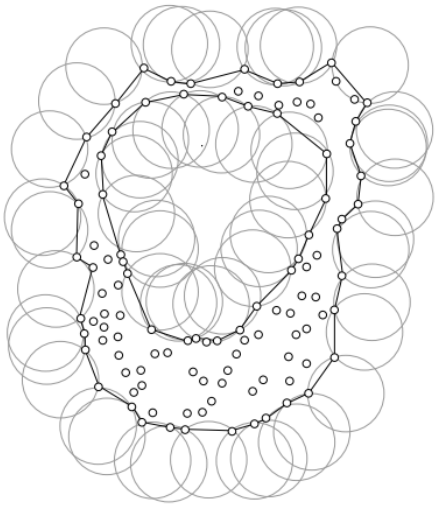


Abbildung 3.5: Alpha-Shape [14]

Der Parameter α definiert den Grad der Detaillierung: Ein kleiner Wert von α (größerer Radius) führt zu einer gröberen Annäherung, die möglicherweise zu einer konvexen Hülle wird. Ein großer Wert von α (kleinerer Radius) ermöglicht die Erkennung feinerer Details und lokaler Strukturen, bis hin zur Darstellung einzelner Punkte.

Die Konstruktion des Alpha-Shapes kann aus der Delaunay-Triangulation erfolgen, indem Dreiecke (2D) oder Tetraeder (3D) entfernt werden, deren Umkreise oder Kugeln einen Radius größer als $1/\alpha$ haben. Dadurch wird die Erstellung des Alpha-Shapes schneller und einfacher berechnet. Darüber hinaus bietet das Alpha-Shape eine detaillierte Beschreibung der Punktwolke, da es die Darstellung von Löchern und konkaven Strukturen ermöglicht. Daher wird die Methode zur Erkennung von Randpunkten in [5, 4, 46, 51] eingesetzt.

Allerdings weist diese Methode eine hohe Empfindlichkeit gegenüber Rauschen auf. Selbst kleine Störungen oder Ausreißer können die Form des Alpha-Shapes erheblich beeinflus-

sen. Die Auswahl eines geeigneten Wertes von α ist herausfordernd, da selbst geringfügige Änderungen die Detaillierung der Form stark verändern können. Dies kann die Erkennung feiner Merkmale wie Ecken oder Löcher in der Punktwolke erschweren. Zudem erfordert die Berechnung bei großer Anzahl der Punkte aufgrund der Delaunay-Triangulation einen erheblichen Zeitaufwand, was die Effizienz der Methode bei sehr dichten Punktwolken einschränkt. Eine weitere Einschränkung dieser Methode ist die Anwendbarkeit auf 3D-Punkte, die auf einer Ebene liegen. In solchen Fällen fällt es schwer, Alpha-Shapes direkt zur Erkennung von Randpunkten zu nutzen, da die Methode primär auf Volumen- oder Flächenrekonstruktion ausgelegt ist.

3.3 Nahterkennung für Schweißroboter

Derzeit existiert eine Vielzahl an Studien zur Merkmalserkennung in Punktwolken und zur Extraktion von Schweißnähten für Schweißroboter, die im Folgenden erläutert werden. Die meisten dieser Studien konzentrieren sich auf die Erkennung und Extraktion von Schweißnähten einfacher Geometrien wie Kehlnaht, Ecknaht, I-Naht, V-Naht oder Überlappnaht. Jedoch existiert derzeit keine spezifische Untersuchung zur automatisierten Schlitzerkennung für Schweißroboter. Die I-Naht (engl. *butt joint*) weist Ähnlichkeiten zur Geometrie eines Schlitzes auf, unterscheidet sich jedoch durch ihre Formgebung. Während eine I-Naht durch zwei aufeinanderstoßende Flächen von zwei Bauteilen gebildet wird, ist ein Schlitz in der Regel auf ein Rechteck beschränkt.

Ein verbreiteter und einfacher Ansatz zur Erkennung und Extraktion von Schweißnähten besteht darin, die Hauptebene der Punktwolke zu bestimmen und darauf basierend relevante Punkte zu extrahieren. Yang und Liu [54] wenden für die Erkennung von V-Schweißnähten den RANSAC-Algorithmus an, um die Hauptebene des Werkstücks zu bestimmen. Anschließend werden die Punkte, die zur Schweißfuge gehören, durch die Analyse der Abstände zwischen den Punkten der Punktwolke und der ermittelten Ebene extrahiert. Die Schweißnahtpunkte werden durch statistische Auswertung der maximalen Abstände erfasst, wobei sich die Punkte der Schweißfuge typischerweise durch größere **Abstände zur Hauptebene** auszeichnen.

Ein ähnlicher Ansatz wird bei anderen Schweißnahtarten wie Kehl-, Eck- oder Überlappnähten verfolgt, insbesondere bei Schweißnähten, bei denen sich zwei Ebenen überschneiden. In diesen Fällen erfolgt die Erkennung durch die **Anpassung mehrerer Ebenen**. Zhang et al. [56] und Takubo et al. [45] verwenden den RANSAC-Algorithmus, um die

Ebenen der Bauteile zu identifizieren. Die Kehlnaht wird anschließend als Schnittlinie der beiden Ebenen extrahiert. Yang et al. [55] verwenden vergleichbare Methoden zur Erkennung von Kehlnähten und erweitern diesen Ansatz über Überlappnähte. Dabei wird die Schweißnaht durch die Analyse der Abstände von Punkten, die zu einer Ebene gehören, zur anderen ermittelt. In [25] verfolgen Kusumoto et al. denselben Ansatz, um Schweißnähte an zylindrischen Bauteilen zu erkennen. In diesem Fall wird die Schweißnaht als Schnittkreis zwischen der Zylinderoberfläche und der Hauptebene extrahiert.

Zhang et al. [57] präsentieren einen Ansatz zur Merkmalsextraktion für ebene Schweißnähte, der auf der **Punktdichte** innerhalb der Punktwolke basiert. Dieser Ansatz wird für V-, I- und Überlappnähte auf ebenen Bauteilen angewendet. Die Punktdichte dient zur Identifizierung der Punktwolken, die den Ebenen der Bauteile entsprechen. Nach der Entfernung des Hintergrunds durch euklidisches Clustering werden die Randpunkte der Ebenen ebenfalls anhand der Punktdichte bestimmt (siehe Unterabschnitt 3.2.1.b). Zur Erkennung der Randlinien wird eine lineare Anpassung mithilfe des RANSAC-Algorithmus durchgeführt. Die Schwerpunkte jeder Ebene werden berechnet, um die relevanten Randlinien zu bestimmen, aus denen schließlich die Schweißlinie extrahiert wird.

Sowohl Peng et al. [35] als auch Gao et al. [20] verwenden die **Abschätzung der Normalen** als den Kern der Methoden zur Extraktion von Schweißnähten. Peng et al. konzentrieren sich auf V-Schweißnähte und verwenden die Methode der kleinsten Quadrate, um die Normalen für jeden Punkt in seiner Nachbarschaft zu schätzen. Durch die Analyse der Winkel zwischen den lokalen Normalen und einer globalen Referenznormale werden die Punkte identifiziert, die zur Schweißfuge gehören, und bilden so die Grundlage für die Generierung der Schweißnahttrajektorie. Gao et al. hingegen zielen auf D-Schweißnähte auf komplexen 3D-Oberflächen wie Pipelines und Druckbehältern ab. Ihre Methode berechnet die Normale für jeden Punkt anhand der Kovarianzmatrix der Nachbarpunkte. Die benachbarten Punkte werden dann auf die angepasste Ebene projiziert, und die Randpunkte werden durch Ermittlung des größten Winkels zwischen zwei Projektionsvektoren bestimmt. Die Sammlung dieser Randpunkte bildet die Schweißnaht.

In [59] stellen Zhou et al. eine Methode zur Identifizierung von Schweißnähten aus 3D-Punktwolken vor, die auf der **Analyse von Kantenintensität und Gradienten** basiert. Die Kantenintensität (engl. *edge intensity*) jedes Punkts wird berechnet, indem die normalisierte Abweichung zwischen dem Punkt und dem lokalen geometrischen Schwerpunkt seiner Nachbarschaft innerhalb eines festen Radius gemessen wird. Deshalb kann diese Kantenintensität wie die **Entfernung von Punkt zum Schwerpunkt** im Un-

terabschnitt 3.2.1.c erwähnt betrachtet werden. Punkte in der Nähe von Kanten oder Rändern weisen typischerweise höhere Intensitätswerte auf. Zur weiteren Verfeinerung der Schweißnahtextraktion wird eine symmetrische Matrix verwendet, die auf einem 3D-gradientenbasierten Ansatz beruht. Dieser Ansatz schätzt die Variationen der Kantenintensität entlang der drei Raumachsen. Die Methode nutzt eine Eigenwertanalyse, um signifikante Änderungen im Gradienten zu erkennen, die auf die Präsenz von Kanten und Rändern hinweisen.

Die oben beschriebenen Ansätze zur Erkennung von Schweißnähten für Schweißroboter verarbeiten Punktwolken als Ganzes, die vom gesamten Werkstück erfasst werden. Zhou et al. [58] verfolgen hingegen einen lokalen Ansatz, bei dem Schweißpunkte schrittweise aus jeder Zeile der geordneten Punktwolke extrahiert und zu einer kontinuierlichen Schweißpunktfolge zusammengefügt werden. Die zentrale Idee besteht darin, den RANSAC-Algorithmus zur **Linienanpassung für jede Zeile** der Punktwolke einzusetzen.
Bei I-Nähten wird für jede Zeile eine Linie angepasst, wobei der Schweißpunkt durch die Identifizierung von Lücken zwischen den Inliers der Linie bestimmt wird. Für V-Schweißnähte erfolgt die RANSAC-Linienanpassung iterativ, um zunächst die Hauptoberfläche und anschließend die beiden Kanten der V-Naht zu extrahieren. Der Schnittpunkt der beiden Kanten wird als Schweißpunkt für die jeweilige Zeile betrachtet. Bei Kehlnähten wird der RANSAC-Algorithmus nacheinander zur Erkennung der Linien der beiden angrenzenden Oberflächen verwendet. Der Schweißpunkt ergibt sich aus dem Schnittpunkt dieser Linien. Für Rohrverbindungen werden die Schweißpunkte durch die Ermittlung der Schnittlinie zweier zylindrischer Flächen innerhalb der Punktwolke bestimmt.

4 Konzeption

In diesem Kapitel werden die Suche nach der Konzeption und Lösungen zur Schlitzerkennung und damit zur Generierung der Zielpunkte für das Schweißen der Schlitz-Zapfen-Verbindung beschrieben.
Aus der Softwarearchitektur und der Beschreibung vom Ablauf des Schweißprozesses in Abschnitt 2.2 ergeben sich zwei wichtige Entwicklungspunkte der Schlitzerkennung, die entsprechend den beiden wichtigsten Aufgaben im Schweißprozess untersucht werden:

- `taskApproach` - initiales Anfahren: Entwicklung einer Lösung zur Erkennung des Schlitzes und zur Generierung des ersten Zielpunkts beim initialen Scan
- `taskMain` - Online-Verfolgung: Entwicklung einer Lösung zur kontinuierlichen Schlitzerkennung, einschließlich der Erkennung des Schlitzendes und der Erstellung von Zielpunkten während der Online-Verfolgung

4.1 `taskApproach` - initiales Anfahren

Bei dieser Aufgabe bewegt sich der TCP des Roboters eine kurze Strecke von etwa 5 cm entlang der grünen Laserlinie des Scanners. Ziel dieser Bewegung ist es, eine breite Punktwolke zu sammeln, sich einen Überblick über die Arbeitsumgebung zu verschaffen und von dort aus das Bauteil zu lokalisieren sowie die Position des Schlitzes zu bestimmen. Daher basiert die Schlitzerkennung in dieser Aufgabe auf der Analyse dieser Punktwolke. Darüber hinaus stellt die Konzeption einer solchen Punktwolke auch die Grundlage für die Offline-Erkennung dar. Eine erhaltene Punktwolke des initialen Scans mit der Breite von 5 cm wird in Abbildung 4.1 dargestellt.

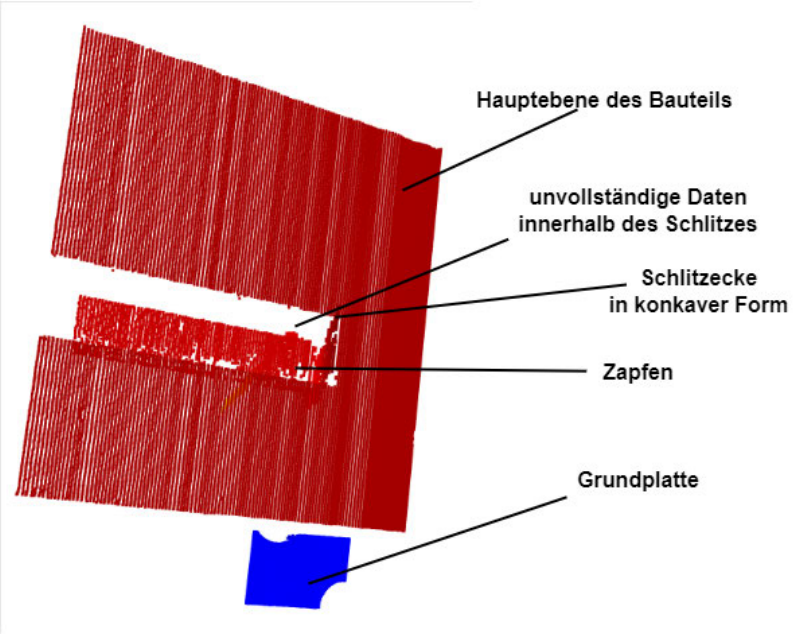


Abbildung 4.1: Die erfasste Punktwolke beim initialen Scan

Teilfunktionen		**Lösungsvarianten**				
Vorverarbeitung	**Clustering**	Euklidisches Clustering	DBSCAN-Clustering	K-Means-Clustering		
	Anpassung der Ebene	Methode der kleinsten Quadrate	RANSAC-Methode			
Verarbeitung	**Merkmals-erkennung**	Punktdichte	Entfernung von Punkt zum Schwerpunkt	Die größte Winkellücke	Abschätzung von Normalen	Haupt-komponenten-analyse (PCA)
		Bildverarbeitung	Alpha-Shape	Abstände zur Hauptebene	Linien-anpassung für jede Zeile	
	Extraktion der Naht	Anpassung mehrerer Linien	Mediale Achse aus Voronoi-Diagramm	Mediale Achse durch Reduzierung des Kugelradius		

Tabelle 4.1: Sammlung der Lösungsvarianten zur Schlitzerkennung

Die Vorgehensweise zur Schlitzerkennung in dieser Aufgabe gliedert sich in zwei zentrale Schritte: die **Vorverarbeitung** und die **Verarbeitung**. Jeder dieser Schritte ist wei-

ter in spezifische Teilfunktionen unterteilt, um die verschiedenen Aspekte der Analyse und Lösung systematisch anzugehen. Basierend auf dem aktuellen Stand der Technik im Kapitel 3 sind die möglichen Lösungsvarianten für jede Teilfunktion in Tabelle 4.1 übersichtlich dargestellt. Die detaillierte Beschreibung, Bewertung und Auswahl der jeweils geeigneten Lösung für jede Teilfunktion werden in den folgenden Unterabschnitten näher erläutert.

4.1.1 Vorverarbeitung

Der Zweck der Vorverarbeitung besteht darin, Rauschen und redundante Punkte, wie beispielsweise jene des Schweißtisches (Grundplatte) oder anderer nicht relevanter Komponenten, zu entfernen. Ziel ist es, ausschließlich die Punktwolke zu extrahieren, die zu dem zu schweißenden Bauteil gehört. Dadurch wird die Grundlage für eine effiziente und präzise Verarbeitung geschaffen.

4.1.1.a Clustering

Beim Clustering wird die Punktwolke in einzelne Cluster segmentiert, um redundante Punkte von der Grundplatte (Schweißtisch) oder der Vorrichtung zu identifizieren und zu isolieren. Gleichzeitig dient die Clusterbildung der effektiven Entfernung von Rauschen. Die Bewertungskriterien für die Wahl der Clustering-Methode umfassen die Effizienz bei der Gruppierung der Punktwolke; die Fähigkeit, Rauschen effektiv zu entfernen; die Robustheit gegenüber komplexen Strukturen und die Unabhängigkeit von der Clusteranzahl.

Es ist wichtig zu berücksichtigen, dass die von Scannern erfasste Punktwolke häufig komplex strukturiert ist und verrauschte Punkte enthält. Dies sind die im Unterabschnitt 3.1.2 genannten Einschränkungen der euklidischen Clustering-Methode. Daher erweist sich das euklidische Clustering für diesen Schritt als ungeeignet. Zudem garantiert die vom Scanner erfasste Punktwolke keine konstante Anzahl von Clustern, da Faktoren wie der Projektionswinkel des Scanners oder die Anzahl der Oberflächen bei jeder Anwendung variieren können. Diese Einschränkungen machen auch das K-Means-Clustering für die Vorverarbeitung ungeeignet.

Aufgrund ihrer Vorteile, insbesondere der Effizienz bei der Clusterbildung und der robusten Erkennung von Rauschen, ist die **DBSCAN-Methode** (*Density-Based Spatial*

Clustering of Applications with Noise) ideal für die Anwendung in dieser Arbeit geeignet. Diese Methode ermöglicht die zuverlässige Gruppierung der Punktwolke, selbst bei unterschiedlicher Dichte und komplexen Strukturen.

4.1.1.b Anpassung der Ebene

Nach dem Clustering der Punktwolke besteht der nächste Schritt darin, das Bauteil-Cluster aus den gefundenen Clustern zu identifizieren, das die Schlitzmerkmale enthält. Da der Scanner bzw. Sensor zur Datenerfassung stets auf die ebene Oberfläche des Bauteils projiziert, kann jede Ebene jedes Clusters als Vergleichsobjekt herangezogen werden. Der entscheidende Faktor ist, dass der Abstand vom Sensor zur Ebene des Bauteils entlang der Sensorprojektion (z-Achse im Sensor-Koordinatensystem) immer der kürzeste ist, wie Abbildung 4.2 dargestellt. Diese spezifische Ebene des Bauteils wird als Hauptebene bezeichnet und bildet die Grundlage für die weitere Analyse und Schlitzerkennung.

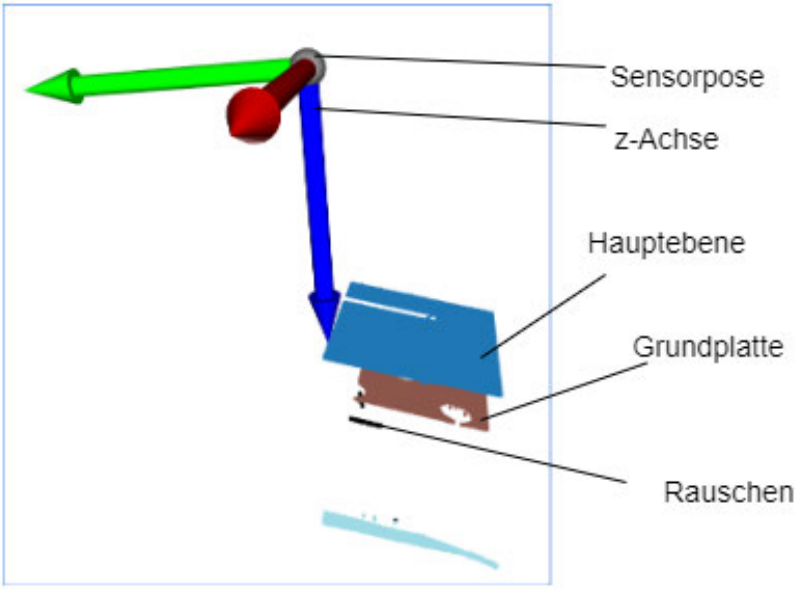


Abbildung 4.2: Bestimmen der Hauptebene

Für jeden Cluster wird eine Ebene angepasst, die als Objekt zur Suche nach der Hauptebene dienen kann. Die Bewertungskriterien für die Wahl der Methode zur Anpassung der Ebene umfassen die Genauigkeit bei der Identifikation der Hauptebene und die Robustheit gegenüber Rauschen sowie Ausreißern.

Zur Anpassung der Ebene für jeden Cluster wird die **RANSAC**-Methode verwendet, da RANSAC robust gegenüber Ausreißern und Rauschen in der Punktwolke ist. RANSAC ermöglicht es, stabile Ebenen auch bei unvollständigen oder verrauschten Daten zu bestimmen. Die Wahl von RANSAC gewährleistet, dass nur die Punkte berücksichtigt werden, die am besten zur Hauptebene passen, wodurch die Genauigkeit der Schlitzerkennung weiter verbessert wird. Im Gegensatz dazu wird die Methode der kleinsten Quadrate nicht eingesetzt, da diese Methode empfindlicher auf Ausreißer reagiert und die resultierende Ebene durch wenige fehlerhafte Punkte stark beeinflusst werden könnte.

4.1.2 Verarbeitung

Der Zweck der Verarbeitung besteht darin, die relevanten Merkmale zu extrahieren, um den Schlitz zu identifizieren und daraufhin die Mittellinie des Schlitzes zu bestimmen. Diese Mittellinie wird als Schweißnaht betrachtet und dient als Grundlage für die Generierung der ersten Zielpunkte, die für die Trajektorienplanung des TCPs verwendet werden.

4.1.2.a Merkmalserkennung

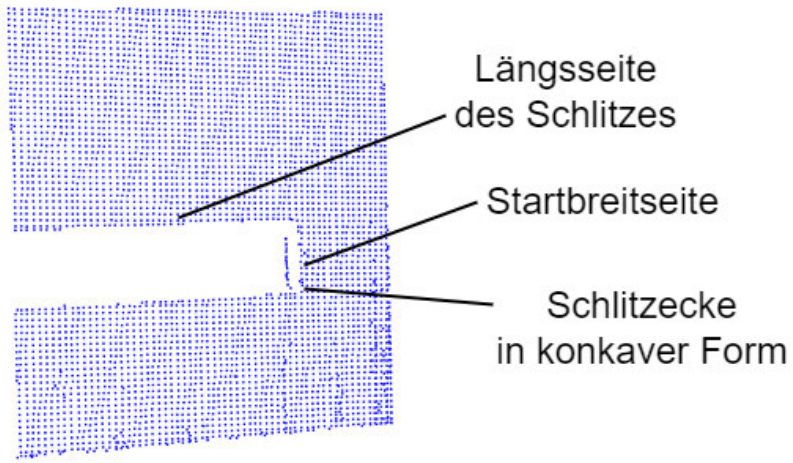


Abbildung 4.3: Hauptebene des Bauteils

Die Erkennung von Merkmalen aus der Punktwolke des Bauteils ist ein wesentlicher Schritt für die präzise Schlitzerkennung und ein zentrales Element des automatisierten Schweißprozesses. Praktische Experimente haben gezeigt, dass bei der Merkmalserkennung aus der Punktwolke des Bauteils folgende Herausforderungen auftreten:

- Unvollständige Datenerfassung innerhalb des Schlitzes oder am Zapfen (siehe Abbildung 4.1):
  Die Erfassung von Daten innerhalb des Schlitzes oder am Zapfen kann aufgrund von Abschattungen oder ungünstigen Projektionswinkeln unvollständig sein, wie in Abschnitt 2.1.3 beschrieben. Daher ist eine Lösung der Merkmalserkennung erforderlich, die unabhängig von diesen spezifischen Datenpunkten arbeitet.

- Ungleichmäßige Punktdichte und Lücke zwischen den Datenzeilen:
  Die ungleichmäßige Punktdichte resultiert aus der Datenübertragung vom Scanner und dem Empfang der Nachrichten über das Thema `/scan_world`. Dadurch kann es vorkommen, dass die empfangene Datenzeile einen erheblichen Abstand zur vorherigen aufweist. Dieses Problem tritt besonders in Simulationsumgebungen auf, wo die Verarbeitungsgeschwindigkeiten vom Rechnerprozessor bzw. Grafikprozessor begrenzt sind. Zur Behandlung dieses Problems ist bei der Datensammlung eine Vorbehandlung für jede Datenzeile erforderlich. Die Vorbehandlung kann auf der Entfernung einzelner Datenzeilen basieren.

- Erkennung von Schlitzecken:
  Das Erkennen der Schlitzecken ist entscheidend, da es für die Bestimmung der Startbreitseite des Schlitzes erforderlich ist. Erkennung von Schlitzecken in konkaver Form (siehe Abbildung 4.3) stellt eine Herausforderung dar, da diese schwer zu erfassen und zu interpretieren sind. Die oben gesammelten Methoden befassen sich nicht speziell mit der Erkennung dieser Ecken, sondern konzentrieren sich eher auf die Erkennung von Kanten, Ränder oder Ecken in konvexer Form.

Die Bewertungskriterien für die Wahl der Methode zur Merkmalserkennung umfassen daher die Fähigkeit, Randpunkte präzise zu identifizieren; die Robustheit gegenüber unvollständigen oder verrauschten Daten; die Recheneffizienz bei der Verarbeitung großer Punktwolken; die Handhabung ungleichmäßiger Punktdichten sowie die Eignung zur Erkennung von Schlitzecken.

Die Methode, die auf der **Punktdichte** basiert, bietet keine ausreichende Genauigkeit, da sie stark von der Wahl des Schwellenwerts sowie der Punktdichte der Punktwolke ab-

hängt. Zudem führt die häufige Fehlklassifizierung benachbarter Punkte als Randpunkte dazu, dass diese Methode für die Anforderungen dieser Arbeit nicht bevorzugt wird.

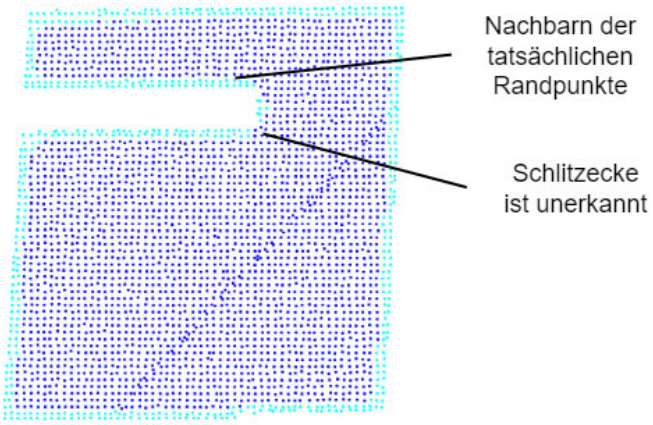


Abbildung 4.4: Methode der Entfernung von Punkt zum Schwerpunkt

Mit ähnlichen Einschränkungen wie bei der Methode der Punktdichte wird die Methode, die auf der **Entfernung von Punkt zum Schwerpunkt** basiert, in dieser Arbeit ebenfalls nicht bevorzugt. Darüber hinaus zeigt sich, dass diese Methode insbesondere bei der Erkennung von Schlitzecken ineffektiv ist, wie in Abbildung 4.4 verdeutlicht wird.

Die auf der **größten Winkellücke** basierende Methode überwindet die Einschränkungen der zuvor genannten Ansätze und bietet eine hohe Genauigkeit bei der Erkennung von Randpunkten. Mit der Auswahl eines geeigneten Schwellenwinkels können zudem die Ecken des Schlitzes präzise identifiziert werden. Diese Methode zeichnet sich durch ihre Einfachheit und hohe Rechengeschwindigkeit aus. Allerdings erfordert diese Methode besondere Betrachtung bei der Auswahl der Nachbarn sowie beim Umgang mit Rauschen, um zuverlässige Ergebnisse zu gewährleisten.

Die Methode der **Abschätzung von Normalen** ist eine effiziente Technik zur Erkennung von Kanten als Schnittlinien zwischen zwei Ebenen, wie im Unterabschnitt 3.2.1.e beschrieben. Die Abbildung 4.5 zeigt den Bereich zwischen der Hauptebene und dem Zapfen, wobei die Datenpunkte (blau) auf der Oberfläche durch Normalen (schwarze Linien) dargestellt werden. Die Normalen zeigen jedoch im dargestellten Bereich keine signifikanten Änderungen in der Richtung. Dies erschwert die Erkennung von Randpunkten und Übergängen zwischen der Hauptebene und dem Zapfen. Zusätzlich beeinträchtigt das Problem unvollständiger Daten innerhalb des Schlitzes die Funktionalität dieser

Methode erheblich. Aus diesen Gründen wird diese Methode in dieser Arbeit nicht implementiert.

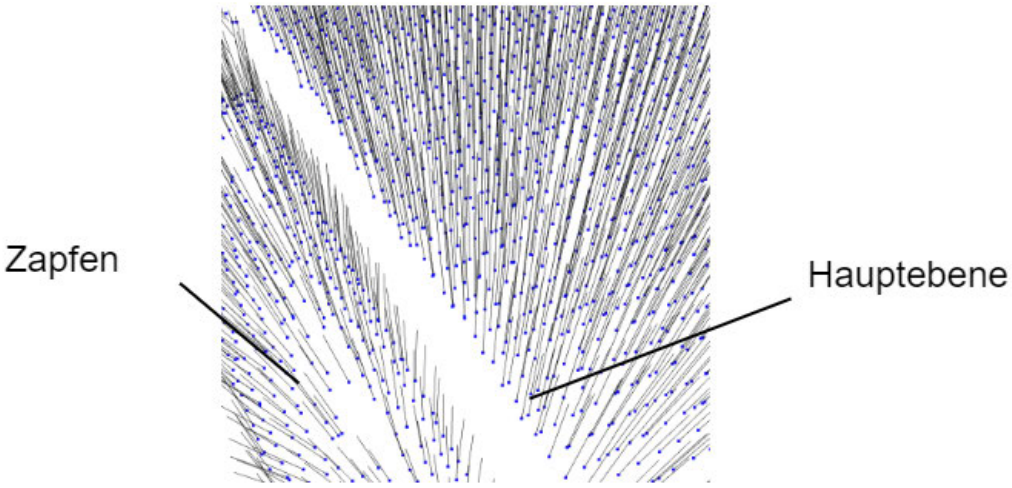


Abbildung 4.5: Abschätzung von Normalen

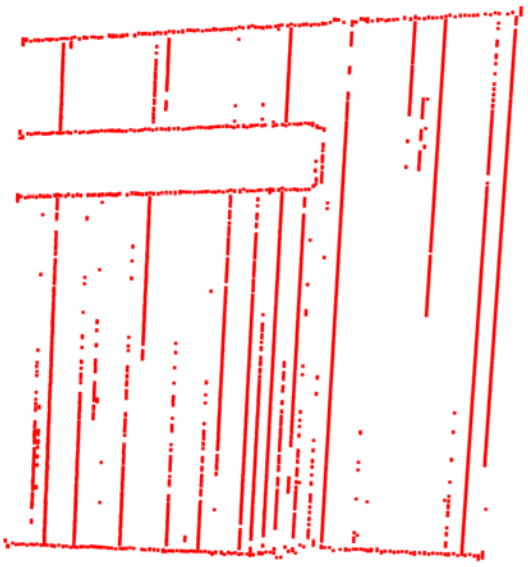


Abbildung 4.6: Versuch zur Schlitzerkennung mit der Hauptkomponentenanalyse

Die **Hauptkomponentenanalyse (PCA)** weist mehrere Nachteile auf, die ihre Eignung für die Schlitzerkennung in dieser Arbeit einschränken und ihre Umsetzung aus-

schließen. Insbesondere ist die PCA für die Identifizierung von Randpunkten ungeeignet, da diese Punkte häufig nicht eindeutig durch die Hauptkomponenten repräsentiert werden. Dies wird besonders problematisch, wenn zwischen den Datenzeilen kleine Lücken bestehen, wie in Abbildung 4.6 dargestellt. Zudem ist die Berechnung der Kovarianzmatrix und ihrer Eigenwerte rechenintensiv, was die Effizienz bei großen Punktwolken erheblich beeinträchtigt. Ein weiterer Nachteil besteht in der Herausforderung, geeignete Bewertungsfunktionen und Schwellenwerte festzulegen, was die Anwendung der Methode zusätzlich erschwert. Außerdem wird bei dieser Methode die Erkennung von konkaven Ecken nicht erwähnt. Aufgrund dieser Einschränkungen wird die PCA im Folgenden nicht umgesetzt.

Wie bereits oben erwähnt, wird aufgrund der Unvollständigkeit der Datenerfassung innerhalb des Schlitzes oder am Zapfen in dieser Arbeit die Methode, die auf **Abständen zur Hauptebene** basiert, nicht verwendet. Aus demselben Grund sowie wegen der Nachteile durch Informationsverluste bei der Projektion werden auch **Bildverarbeitungsmethoden** in dieser Arbeit nicht berücksichtigt.

Die **Alpha-Shape**-Methode eignet sich gut zur Erkennung von Randpunkten im 2D-Raum, weist jedoch die Einschränkung auf, dass sie nicht direkt auf die Erkennung von Randpunkten in Ebenen eines 3D-Koordinatensystems angewendet werden kann. Bei der Umwandlung der Punkte der Hauptebene in ein 2D-Koordinatensystem gehen kleine geometrische Informationen verloren. Darüber hinaus ist die Methode komplex und erfordert intensive Berechnungen, insbesondere bei großen Punktwolken. Die Wahl eines geeigneten Alpha-Wertes gestaltet sich ebenfalls schwierig, da Lücken zwischen den Datenzeilen die Optimierung erschweren. Aufgrund dieser Einschränkungen wird die Alpha-Shape-Methode nicht für die Erkennung dieses spezifischen Merkmals eingesetzt.

Die Methode, die auf der **Linienanpassung für jede einzelne Zeile** basiert, wird im taskApproach nicht bevorzugt. Die zeilenweise Verarbeitung einer großen Anzahl von Datenzeilen ist rechnerisch ineffizient und bietet keinen umfassenden Überblick über die gesamte Arbeitsumgebung. Allerdings könnte diese Methode im taskMain betrachtet werden, sobald die Lage des Schlitzes bestimmt wird, um die fortlaufende Verfolgung und Punktgenerierung effizient zu unterstützen.

Basierend auf den Experimenten und den oben beschriebenen Analysen wird deutlich, dass die Methode zur Erkennung der Randpunkte, die auf der **größten Winkellücke** basiert, optimal ist und für die Verwendung in dieser Arbeit ausgewählt wird. Die Methode kann geringfügig angepasst werden, indem anstelle einer separaten Ebene für die

Nachbarschaft jedes Abfragepunkts die Hauptebene für alle Punkte dieser Ebene verwendet wird. Diese Anpassung beschleunigt den Prozess erheblich. Ein Winkelwert von **75 Grad** ($\frac{5\pi}{12}$) kann gewählt werden, um sowohl Randpunkte als auch konvexe und konkave Ecken zuverlässig zu erkennen. In Abbildung 4.7 sind die erkannten Randpunkte der Hauptebene in Zyan dargestellt.

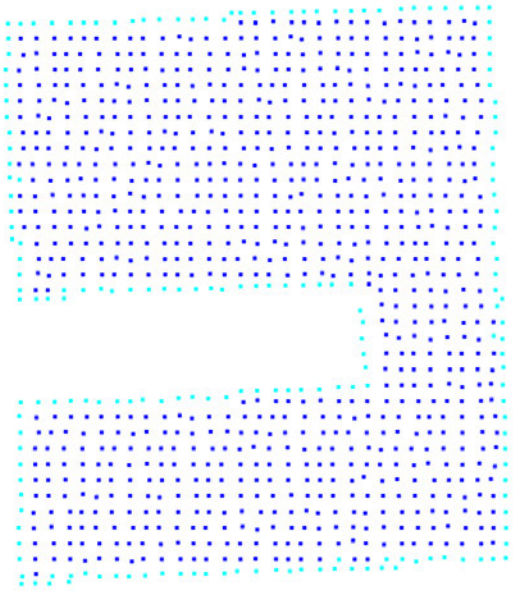


Abbildung 4.7: Erkennung der Randpunkte via Methode der größten Winkellücke

4.1.2.b Extraktion der Naht

Nach dem Extrahieren der Randpunkte der Hauptebene besteht der nächste Schritt darin, die Mittellinie des Schlitzes zu finden, die als Grundlage für die Generierung der ersten Zielpunkte dient. Die Bewertungskriterien für die Methode zur Nahtextraktion umfassen die Genauigkeit bei der Bestimmung der Mittellinie trotz unvollständiger oder verrauschter Randpunkte; die Recheneffizienz; die direkte Anwendung im 3D-Koordinatensystem und die Fähigkeit bei gekrümmten Schlitzverläufen.

Ein Ansatz zur Nahtextraktion ist die Methode der **Anpassung mehrerer Linien**. Diese Methode besteht aus drei Schritten: Zunächst werden alle geraden Linien identifiziert, die durch die Randpunkte der Hauptebene gebildet werden. Anschließend werden mögliche Linien ausgewählt, die den Seiten des Schlitzes entsprechen. Im letzten Schritt wird

die Mittellinie des Schlitzes berechnet, indem der Durchschnitt der ausgewählten Linien berechnet wird oder ein geometrisches Modell verwendet wird.

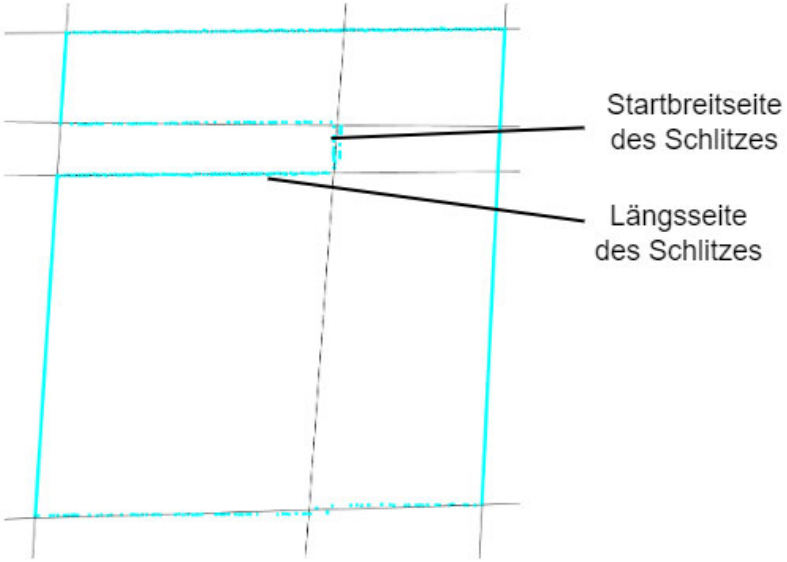


Abbildung 4.8: Anpassung mehrerer Linien

Diese Methode weist einige Nachteile auf. Unvollständige oder verrauschte Randpunkte erschweren die Bestimmung der Startbreitseite des Schlitzes. Die Anpassung der Linie dieser Seite ist aufgrund der geringen Länge oft schwierig, da nur wenige Punkte erkannt werden. Bei Schlitzen, deren Verlauf entlang der Länge nicht geradlinig, sondern gekrümmt ist, kann die Zuordnung der Linien zu den Schlitzseiten unzuverlässig sein. Zudem erfordert das Finden und Analysieren aller möglichen Linien einen hohen Rechenaufwand. Aufgrund dieser Einschränkungen ist die Methode für diesen Schritt nicht bevorzugt.

Die Methode zur Erstellung der **Mittelachse aus dem Voronoi-Diagramm** wird in diesem Schritt nicht angewendet. Voronoi-Diagramme setzen voraus, dass die Punkte über den gesamten 3D-Raum verteilt sind, um volumetrische Partitionen zu erstellen. Da die gefundenen Randpunkte jedoch alle auf einer einzigen Ebene im 3D-Koordinatensystem liegen, ist die Erstellung eines Voronoi-Diagramms in diesem Kontext nicht möglich. Zudem ist eine Umwandlung in ein 2D-Koordinatensystem nicht empfehlenswert, da hierbei kleine geometrische Informationen verloren gehen können.

Die von Ma et al. [26] vorgeschlagene Methode zur Erstellung **medialer Achsen durch iterative Reduzierung des Kugelradius** erweist sich in diesem Fall als effektive Lösung. Diese Methode kann direkt auf Randpunkte in einem 3D-Koordinatensystem angewendet werden und ist somit flexibel einsetzbar. Darüber hinaus gewährleistet die Methode sowohl Effizienz als auch Genauigkeit bei der Berechnung medialer Punkte, selbst für Schlitze mit unterschiedlichen Verläufen, sei es geradlinig oder gekrümmt entlang ihrer Länge. In Abbildung 4.9 wird die mediale Achse des Schlitzes in Rot angezeigt.

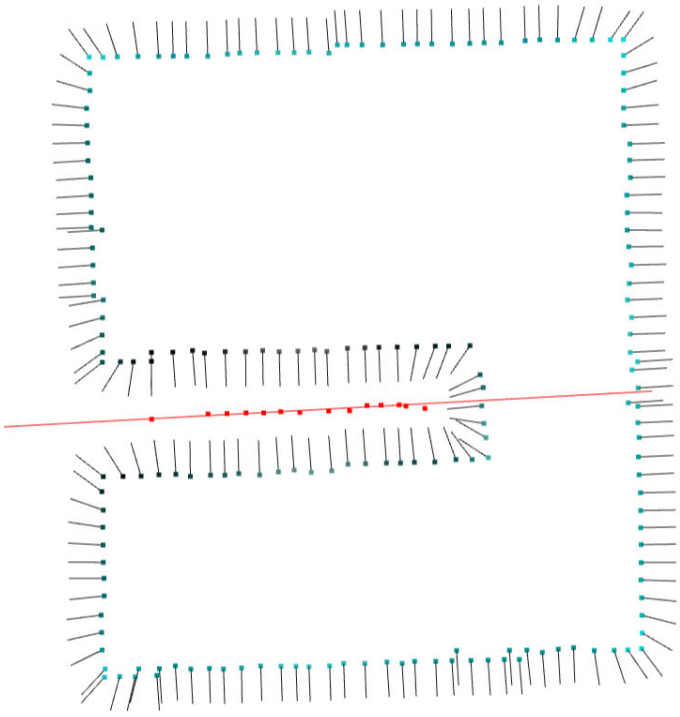


Abbildung 4.9: Die medial Achse des Schlitzes

Teilfunktionen		Lösung
Vorverarbeitung	**Clustering**	DBSCAN-Clustering
	Anpassung der Ebene	RANSAC-Methode
Verarbeitung	**Merkmals-erkennung**	Die größte Winkellücke
	Extraktion der Naht	Mediale Achse durch Reduzierung des Kugelradius

Tabelle 4.2: Konzept der Schlitzerkennung in `taskApproach`

Basierend auf den oben genannten Bewertungen und Analysen wird das Konzept zur Schlitzerkennung im `taskApproach` entwickelt. Die Umsetzung dieses Konzepts wird als `SearchingSlitDetector` bezeichnet und ausführlich in Abschnitt 5.2 beschrieben.

4.2 `taskMain` - Online-Verfolgung

Nachdem der Schlitz lokalisiert und die ersten Zielpunkte bestimmt worden sind, konzentriert sich diese Aufgabe darauf, der Schweißnaht kontinuierlich zu folgen und neue Schweißpunkte zu erzeugen, bis das Ende des Schlitzes erkannt wird. Die Erkennung des Schlitzes in dieser Aufgabe erfolgt daher in kurzer Zeit, um die Kontinuität des Prozesses zu wahren. Daher wird die zu verarbeitende Punktwolke schmaler und lokaler sein. Eine erhaltene Punktwolke zur Online-Verfolgung mit der Breite von 3 mm wird in Abbildung 4.10 dargestellt.

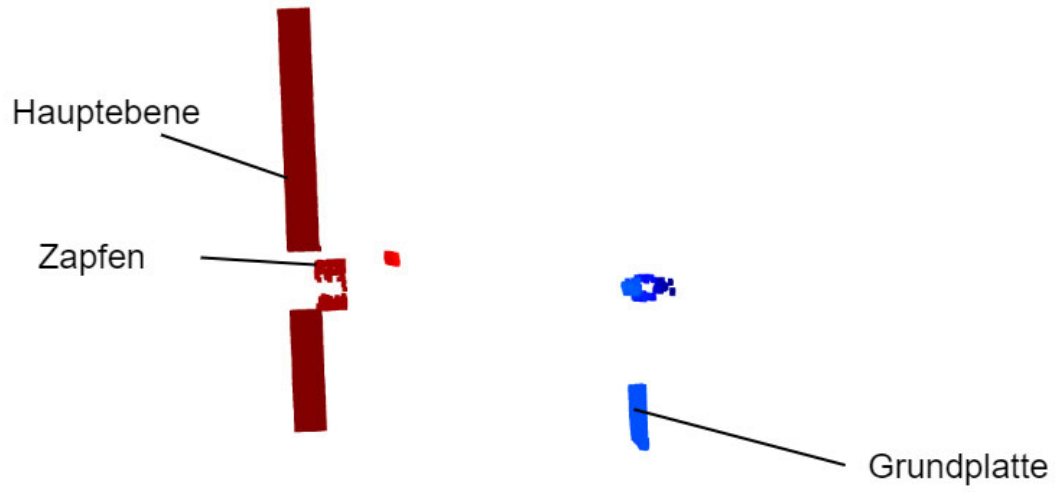


Abbildung 4.10: Eine erfasste Punktwolke zur Online-Verfolgung

Bei einer so kleinen Punktwolke ist die Anwendung des oben entwickelten Konzepts in `taskApproach` nicht effektiv.
In der Vorverarbeitung ist die Verwendung von Clustering und Anpassung der Ebene überflüssig, da die Hauptebene bereits in `taskApproach` identifiziert wird. Punkte auf der Hauptebene des Bauteils können direkt durch den **Abstand zu dieser Ebene** extrahiert werden.

Für die Merkmalserkennung bei der Verarbeitung ist die Erkennung aller Randpunkte der Hauptebene nicht sehr sinnvoll, da die Lage des Schlitzes bereits bestimmt wird. Außerdem ist die Breite dieser Punktwolke zu gering, um Methoden zur Merkmalserkennung auf der Grundlage der Nächste-Nachbarn-Analyse anzuwenden. Methoden wie Bildverarbeitung oder Alpha-Shape sind aus den oben genannten Gründen ebenfalls nicht anwendbar.

In Anlehnung an die von Zhou et al. in [58] vorgeschlagene Methode eignet sich der Algorithmus zur Schlitzerkennung in `taskMain`, der auf **Linienanpassung für jede Zeile** für I-Nähte basiert. Dieser Algorithmus bietet eine hohe Genauigkeit und eine hohe Berechnungsgeschwindigkeit, um zwei Randpunkte des Schlitzes für jede Datenzeile zu

erkennen. Von dort aus wird dann der Punkt der medialen Achse des Schlitzes erzeugt. Darüber hinaus ist der Algorithmus einfach zu implementieren.

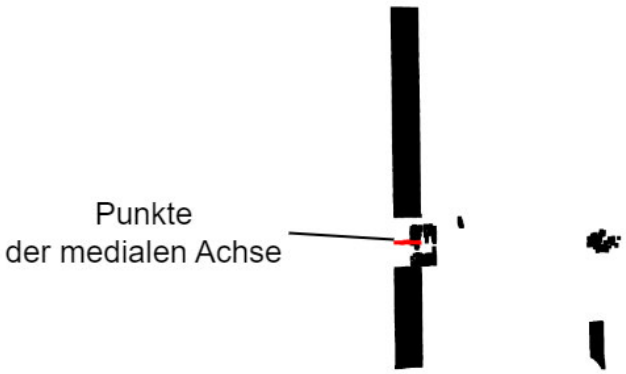


Abbildung 4.11: Punkte der medialen Achse in Online-Verfolgung in Rot

In Abbildung 4.11 werden die Punkte der medialen Achse in Online-Verfolgung in Rot dargestellt. Die Umsetzung der Schlitzerkennung in `taskMain` wird als `FollowingSlitDetector` bezeichnet und ausführlich in Abschnitt 5.3 erläutert.

5 Implementierung

Dieses Kapitel beschreibt die Implementierung des oben entwickelten Konzepts zur Schlitzerkennung und Generierung von Zielpunkten für die Trajektorienplanung. Basierend auf der in Abschnitt 2.2.2 erläuterten Softwarearchitektur liegt der Schwerpunkt der Implementierung auf der Entwicklung spezialisierter Versionen von `TaskApplier` und `Detektor`. Darüber hinaus wird `OnlineFollower` an das entwickelte Konzept angepasst und als `SlitFollower` bezeichnet. Da die Programmierung in dieser Arbeit in Python erfolgt, wird die *open3D*-Bibliothek [60] in der Version 0.13.0 genutzt, um die Verarbeitung der Punktwolken effizient zu unterstützen.

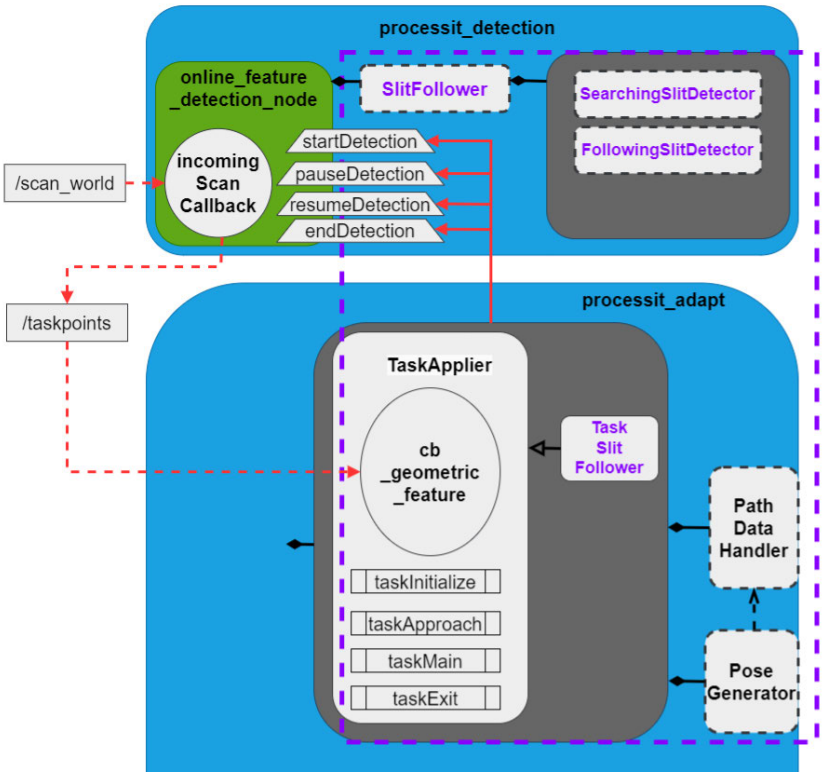


Abbildung 5.1: Die modifizierte Softwarearchitektur zur Schlitzerkennung

5.1 SlitFollower

Ähnlich wie `OnlineFollower` ist `SlitFollower` dafür verantwortlich, einzelne Datenzeilen aus dem Thema `/scan_world` in der Methode `processScanLine` zu sammeln und zu verarbeiten. Die Variable *Fenstergröße* wird verwendet, um die Breite der gesammelten Datenzeilen zu definieren. Sobald die *Fenstergröße* einen festgelegten Schwellenwert (*Fensteroffset*) erreicht, wird die Erkennungsmethode des entsprechenden `Detektors` aktiviert. Das Objekt speichert die Ergebnisse und veröffentlicht die generierten Zielpunkte über das Thema `/taskpoints`.

5.1.1 Zustandsautomat

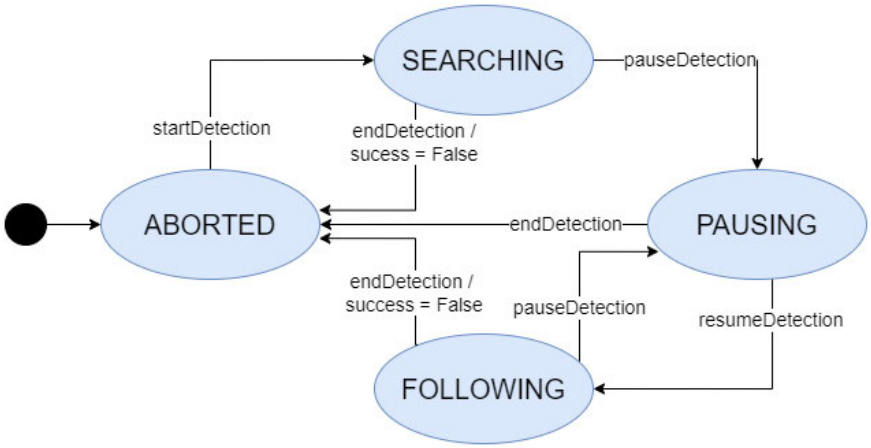


Abbildung 5.2: Zustandsdiagramm vom `SlitFollower`

Um die Sammlung und Verarbeitung von Datenzeilen entsprechend dem Fortschritt des Schweißprozesses effizient zu verwalten und zu steuern, wird ein Zustandsautomat für `SlitFollower` entwickelt, der im Zustandsdiagramm in Abbildung 5.2 dargestellt wird. Die Zustandsübergänge erfolgen entweder durch die Aktivierung der Dienste im Knoten `online_feature_detection_node` (`startDetection`, `pauseDetection`, `resumeDetection` und `endDetection`) oder basieren auf den Ergebnissen der Verarbeitung der gesammelten Datenzeilen. Jeder Zustand legt fest, ob die vom Thema `/scan_world` empfangenen Nachrichten verarbeitet werden sollen oder nicht. Zudem definiert jeder Zustand den passenden `Detektor` und den entsprechenden Wert von *Fensteroffset*.

ABORTED
Dieser Zustand ist der Freizustand des Zustandsautomaten:

- In diesem Zustand werden die empfangenen Datenzeilen ignoriert und keine weiteren Aktionen durchgeführt.
- Ein Übergang in den Zustand **SEARCHING** ist nur durch den Aufruf des Dienstes `startDetection` möglich.
- Darüber hinaus können alle anderen Zustände in den Freizustand zurückkehren, wenn der Dienst `endDetection` aufgerufen wird oder die Verarbeitung der Datenzeile fehlschlägt.

SEARCHING
Der Zustand **SEARCHING** entspricht dem `taskApproach` des `TaskAppliers`.

- In diesem Zustand werden die Datenzeilen gesammelt, wobei der *Fensteroffset* entsprechend der zu scannenden Länge angepasst wird. Der `SearchingSlitDetector` wird als `Detektor` eingesetzt, um die Schlitzerkennung und die Generierung des ersten Zielpunkts durchzuführen.
- Ein Übergang in den Zustand **PAUSING** kann über den Dienst `pauseDetection` erfolgen.

PAUSING
Dieser Zustand dient dazu, den Prozess vorübergehend anzuhalten, ohne den aktuellen Fortschritt oder die Konfiguration des Detektors zu verlieren.

- Im Zustand **PAUSING** werden keine Datenzeilen gesammelt oder verarbeitet, um eine Überlastung des Speichers zu vermeiden. Der verwendete `Detektor` bleibt in diesem Zustand unverändert.
- Auf diesen Zustand kann über den Dienst `pauseDetection` zugegriffen werden.

FOLLOWING
Der Zustand **FOLLOWING** entspricht dem `taskMain` des `TaskAppliers` und ermöglicht eine kontinuierliche Verfolgung des Schlitzverlaufs während des Schweißprozesses.

- In diesem Zustand werden kontinuierlich Datenzeilen gesammelt, wobei der *Fensteroffset* klein genug eingestellt wird, um die fortlaufende Schlitzerkennung sicherzustellen. Der verwendete `Detektor` in diesem Zustand ist der `FollowingSlitDetector`.
- Der Übergang in diesen Zustand erfolgt über den Dienst `resumeDetection`, wodurch die Schlitzerkennung nach einer Pause fortgesetzt wird.
- Ein Übergang in den Zustand **ABORTED** erfolgt, wenn ein Fehler während der Datenverarbeitung auftritt oder das Ende des Schlitzes erkannt wird.

5.1.2 `processScanLine`

Die Methode `processScanLine` ist die zentrale Funktion des `SlitFollower`. Sie wird in der Callback-Methode `incomingScanCallback` des Knotens `online_feature_detection_node` genutzt, um neue Nachrichten, die über das Thema `/scan_world` veröffentlicht werden, zu sammeln und zu verarbeiten. Der Ablauf dieser Methode wird in Abbildung 5.3 veranschaulicht und umfasst die folgenden Schritte:

Überprüfung des Zustands

Der aktuelle Zustand des Zustandsautomaten bestimmt, ob die Daten gesammelt und verarbeitet werden sollen. Die detaillierte Beschreibung des Zustandsautomaten wird im Unterabschnitt 5.1.1 erläutert.

Sammlung der Scan-Datenzeile

Die Sammlung einzelner Datenzeilen erfordert eine Vorbehandlung, um eine zuverlässige Grundlage für die Schlitzerkennung zu schaffen. Dieser Schritt umfasst die Filterung und ROI-Extraktion der Daten und wird detailliert im Unterabschnitt 5.1.3 beschrieben.

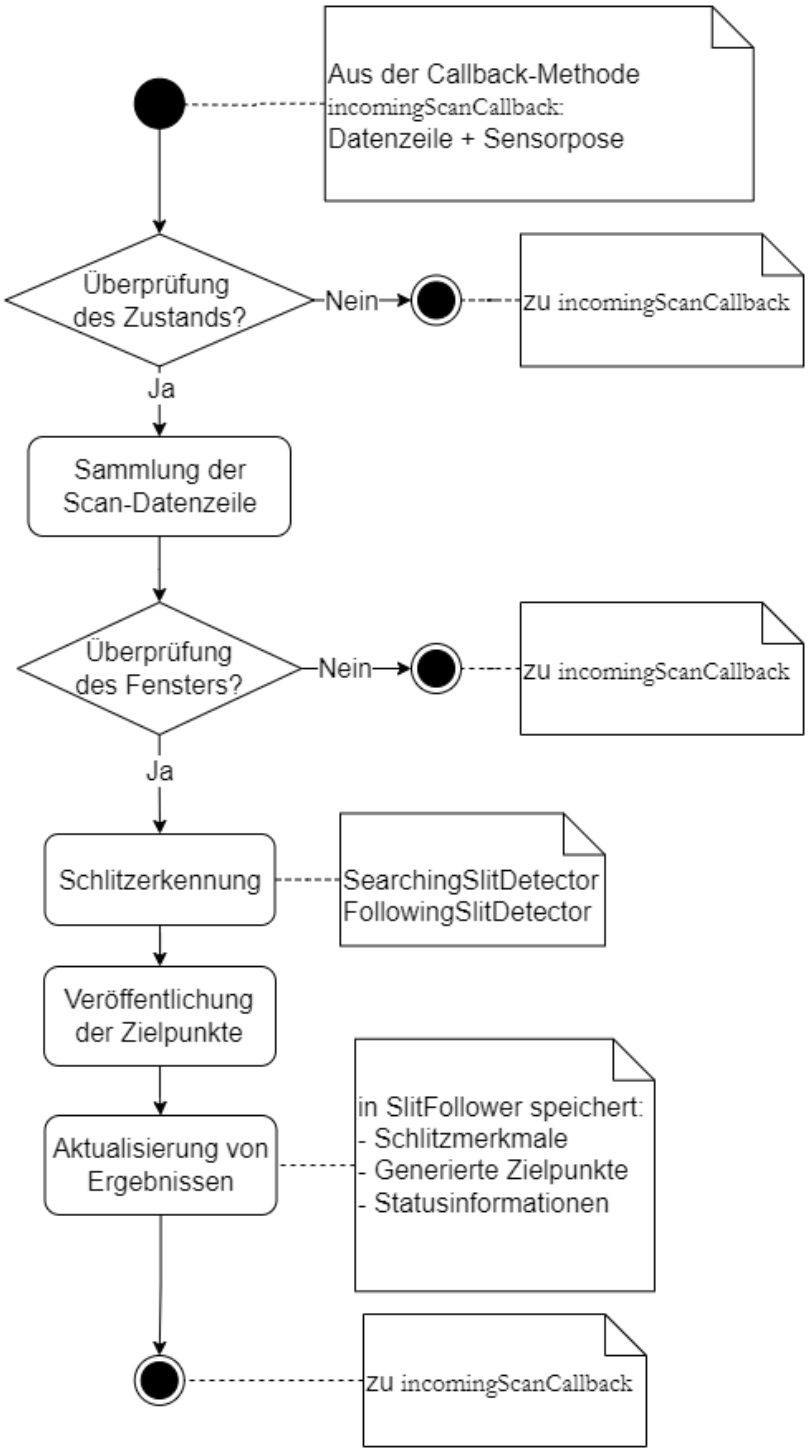


Abbildung 5.3: Ablauf von `processScanLine`

Überprüfung des Fensters

Sobald die *Fenstergröße* einen festgelegten Schwellenwert (*Fensteroffset*) erreicht, können die gesammelten Datenzeilen für die Analyse und Schlitzerkennung extrahiert werden. Der *Fensteroffset* variiert je nach Aufgabe:

- In `taskApproach` ist eine größere Anzahl an Datenzeilen erforderlich, um einen Überblick über die Arbeitsumgebung zu erhalten und den Schlitz sowie den ersten Zielpunkt zu bestimmen. Daher kann der *Fensteroffset* der Strecke von **50 mm** entsprechen, die der TCP während des initialen Scans zurücklegt.
- In `taskMain` hingegen muss der *Fensteroffset* klein gehalten werden, um eine kontinuierliche Schlitzerkennung und die fortlaufende Generierung neuer Zielpunkte sicherzustellen. Um eine kontinuierliche Online-Verfolgung der Schweißnaht während des laufenden Schweißprozesses zu gewährleisten und gleichzeitig genügend Daten für die Schlitzerkennung zur Verfügung zu haben, wird für den *Fensteroffset* ein Wert von **3 mm** eingestellt.

Schlitzerkennung

Die gesammelten Daten werden an den aktiven Detektor übergeben, um relevante Merkmale zu erkennen und Zielpunkte zu generieren. Der `SearchingSlitDetector` wird im Abschnitt 5.2 beschrieben, während der `FollowingSlitDetector` im Abschnitt 5.3 erläutert wird.

Veröffentlichung der Zielpunkte

Die generierten Zielpunkte, bestehend aus Positionen, Orientierungen und Indizes, werden über das Thema `/taskpoints` veröffentlicht. Diese Zielpunkte dienen als Grundlage für die weitere Trajektorienplanung.

Aktualisierung von Ergebnissen

Die Ergebnisse der Schlitzerkennung in `SlitFollower` werden gespeichert und aktualisiert, um als Grundlage für die Online-Verfolgung zu dienen. Dabei werden folgende wichtige Informationen erfasst:

- Schlitzmerkmale: Breite, Tiefe, Mittellinie sowie der Mittelpunkt der Startbreitseite.

- Generierte Zielpunkte: Die berechneten Punkte, die für die Trajektorienplanung verwendet werden.
- Statusinformationen:
  - `success`: Gibt an, ob der Schlitz erfolgreich erkannt wird.
  - `continue_flag`: Ist auf `True` gesetzt, wenn das Ende des Schlitzes nicht erkannt wird, sodass die Verfolgung fortgesetzt werden kann.

Nach der Aktualisierung von Ergebnissen werden die gesammelten Daten gelöscht, um Speicherplatz freizugeben und die Weiterverarbeitung sicherzustellen.

5.1.3 Sammlung der Scan-Datenzeile

Um eine übermäßige Anzahl von Datenpunkten zu vermeiden und Redundanz sowie Rauschen zu reduzieren, wird jede vom Thema `/scan_world` empfangene Datenzeile einer Vorbehandlung unterzogen. Der Ablauf der Datensammlung bei jedem Empfang einer neuen Nachricht ist in der Abbildung 5.4 dargestellt.

Abstand zur vorherigen Datenzeile berechnen

Um die Duplizierung von Datenpunkten zu vermeiden und eine gleichmäßige Punktdichte der Punktwolke sicherzustellen, muss der Abstand zwischen den einzelnen Datenzeilen überprüft werden. Da die Oberflächenstruktur variiert und die Daten in jeder Zeile nicht immer exakt entlang einer geraden Linie ausgerichtet sind, gestaltet sich die direkte Abstandsmessung zwischen den Zeilen als schwierig. Stattdessen wird der Abstand zwischen den Datenzeilen indirekt ermittelt, indem die Distanz zwischen den jeweiligen Sensorpositionen verwendet wird. Jede Sensorposition kann aus der Nachricht des Themas `/scan_world` ermittelt werden.

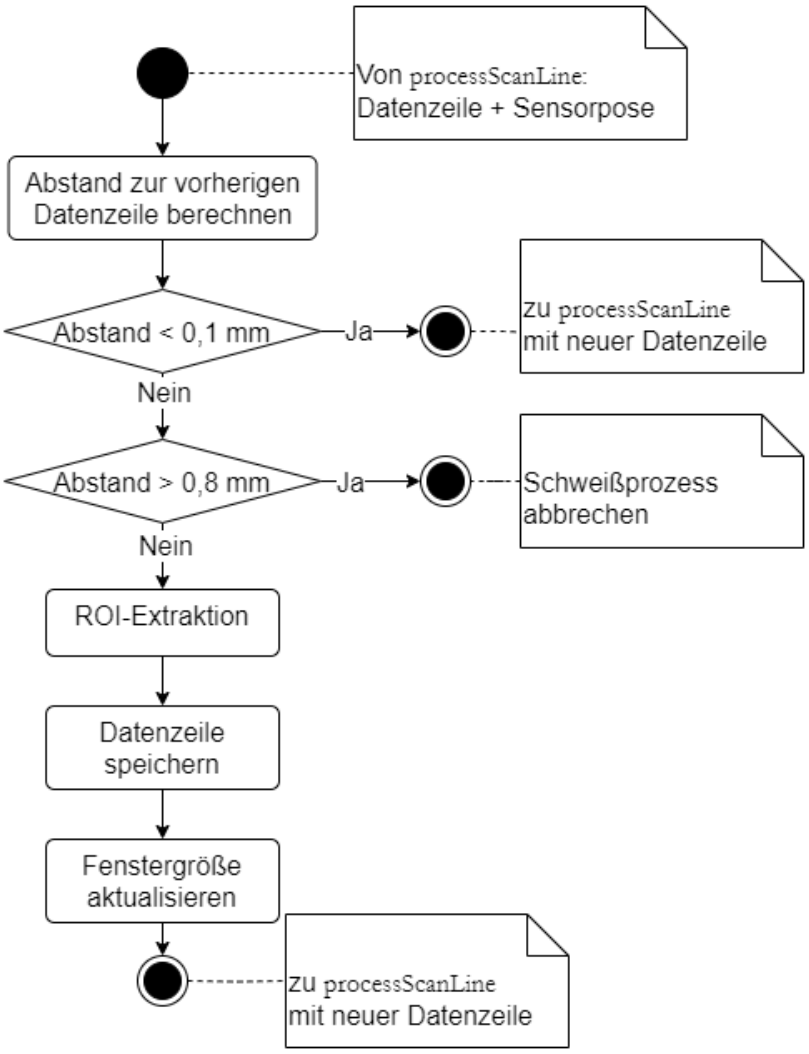


Abbildung 5.4: Ablauf der Datensammlung

Der Abstand zwischen der Sensorposition der neu empfangenen Nachricht und der Sensorposition der zuletzt gespeicherten Zeile wird berechnet. Liegt dieser Abstand unterhalb eines festgelegten Schwellenwerts von **0,1 mm**, werden die neuen Daten nicht gespeichert, da sie sich zu nahe an den bereits vorhandenen Punkten befinden und voraussichtlich keine zusätzlichen, relevanten Informationen liefern. Überschreitet der Abstand hingegen den oberen Schwellenwert von **0,8 mm**, weist dies auf einen Fehler im Scanvorgang oder bei der Datenübertragung hin. Ein solcher Fehler kann die Schlitzerkennung in den nach-

folgenden Schritten beeinträchtigen, weshalb der Schweißprozess in diesem Fall sofort abgebrochen wird.

ROI-Extraktion

Um die Analyse auf die zentrale Region zu fokussieren und redundante Informationen zu eliminieren, wird aus jeder Datenzeile die *Region of Interest* (ROI) extrahiert, wie in Abbildung 5.5 dargestellt. Die über das Thema `/scan_world` empfangenen Datenpunkte enthalten 3D-Informationen im Weltkoordinatensystem. Diese Punkte lassen sich in das Sensorkoordinatensystem des Scanners transformieren und in 2D-Daten umwandeln. Basierend auf den absoluten Werten entlang der x-Achse in diesem Koordinatensystem (entsprechend der roten Laserlinie) werden alle Punkte, die außerhalb eines definierten Schwellenwerts von **60 mm** (Standardwert) liegen, herausgefiltert und entfernt.

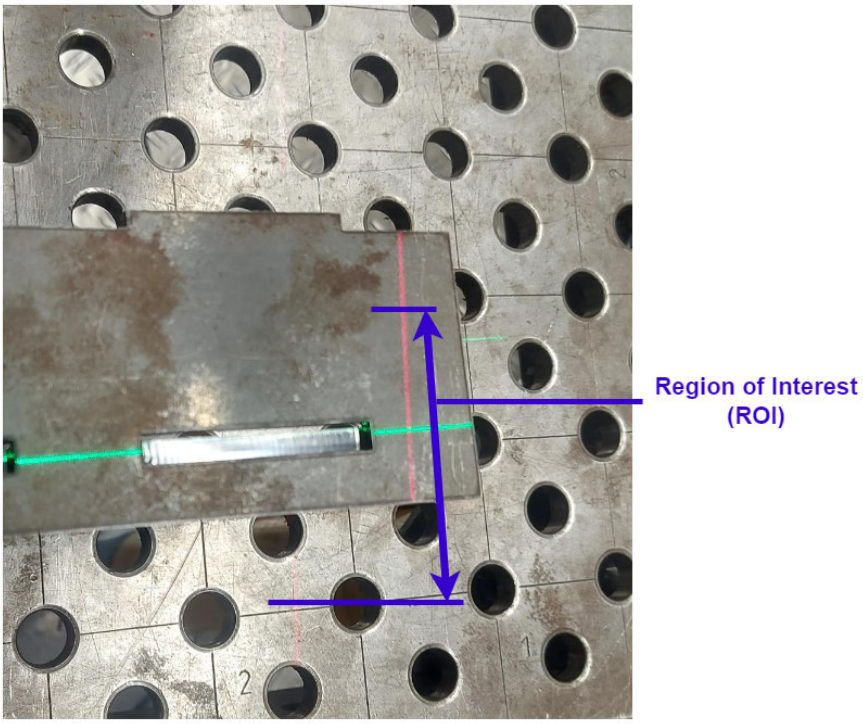


Abbildung 5.5: *Region of Interest* (ROI)

Datenzeile speichern und Fenstergröße aktualisieren

Nachdem die redundanten Datenpunkte entfernt werden, werden die verbleibenden Datenpunkte der Zeile gespeichert. Die *Fenstergröße* ist eine Variable, die die Anzahl der

gesammelten Datenzeilen bestimmt. Sie wird durch iteratives Addieren des berechneten Abstands zur vorherigen Datenzeile ermittelt.

5.2 `SearchingSlitDetector`

Basierend auf dem in Abschnitt 4.1 entwickelten Konzept wird der `SearchingSlitDetector` implementiert. `SearchingSlitDetector` ist für die Erkennung des Schlitzes und für die Generierung der ersten Zielpunkte beim initialen Scan verantwortlich. Das Diagramm 5.6 zeigt den allgemeinen Ablauf der Schlitzerkennung im `SearchingSlitDetector`. Die einzelnen Schritte lassen sich wie folgt beschreiben:

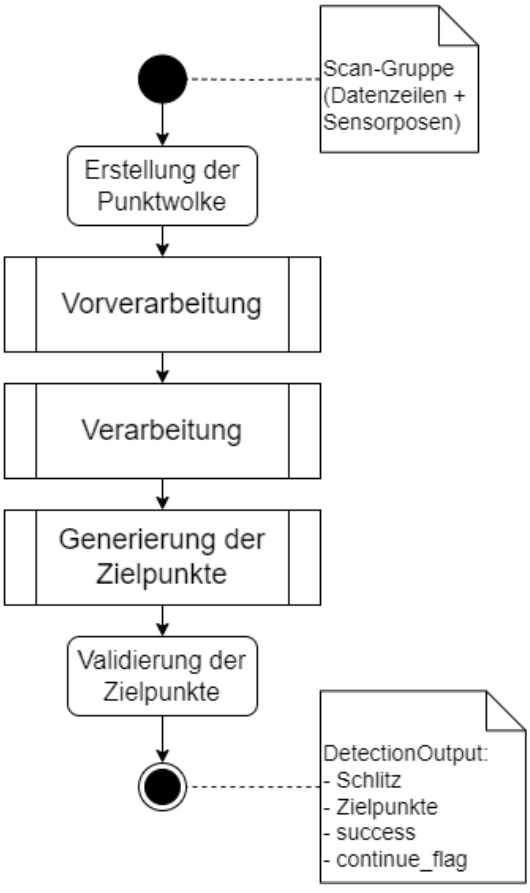


Abbildung 5.6: Der allgemeine Ablauf der Schlitzerkennung im `SearchingSlitDetector`

5.2.1 Erstellung der Punktwolke

Bevor die Datenverarbeitung beginnt, werden die Sensorposen überprüft, um sicherzustellen, dass sich der Sensor entlang einer geraden Linie bewegt, was anhand der Werte der Y-Achse validiert wird. Die von `SlitFollower` gesammelten Datenzeilen werden anschließend zu einer Punktwolke zusammengeführt, die als Open3D-Geometry-PointCloud repräsentiert wird. Diese Punktwolke dient als Grundlage für die weitere Verarbeitung und Analyse.

5.2.2 Vorverarbeitung

Vor dem Extrahieren der Schlitzmerkmale müssen redundante Datenpunkte von der Grundplatte oder anderen Bauteilen entfernt werden, um so die Hauptebene zu finden, die den Schlitz enthält. Der Ablauf der Vorverarbeitung wird in Abbildung 5.7 angezeigt.

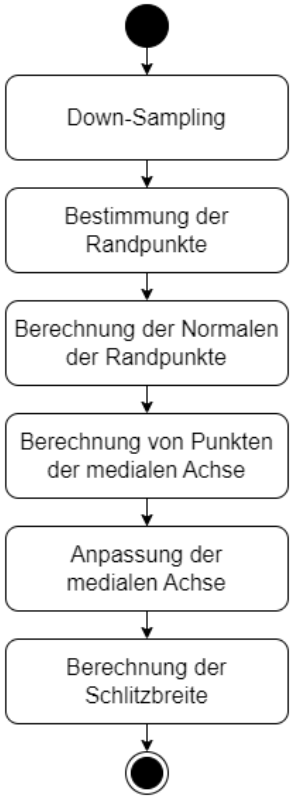


Abbildung 5.7: Ablauf der Vorverarbeitung im `SearchingSlitDetector`

Down-Sampling

Im ersten Schritt wird die Anzahl der Punkte in der Punktwolke reduziert, ohne dabei die wesentlichen Merkmale zu beeinträchtigen. Dieser Schritt dient dazu, die Verarbeitung in den nachfolgenden Schritten zu beschleunigen. Hierfür wird Voxel-Down-Sampling angewendet, wie im Unterabschnitt 3.1.1 beschrieben.

Clustering

Für das Clustering wird die Funktion `cluster_dbscan` der *open3D*-Bibliothek eingesetzt. Diese Funktion liefert Labels für die Datenpunkte zurück, wobei das Label **-1** Rauschen kennzeichnet. Punkte mit diesem Label werden als nicht zugehörig zu einem Cluster betrachtet und können in den folgenden Verarbeitungsschritten ignoriert oder entfernt werden.

Bestimmen der Hauptebene

Nach dem Clustering der Punktwolke erfolgt der nächste Schritt, die Datenpunkte zu identifizieren, die zur Hauptebene des Bauteils gehören und die Schlitzinformationen enthalten.
Für jeden Cluster wird die Funktion `segment_plane` der *open3D*-Bibliothek verwendet, um die Ebene des Clusters zu bestimmen. Diese Funktion nutzt den RANSAC-Algorithmus, um eine robuste Anpassung der Ebene vorzunehmen.
Es werden anschließend zwei Abstände berechnet: der Abstand von der Anfangsposition und der Endposition entlang der z-Achse des Sensorkoordinatensystems zur jeweiligen Ebene. Der Vergleichswert ergibt sich aus dem Durchschnitt dieser beiden Abstände. Die Ebene mit dem minimalen Vergleichswert wird als Hauptebene identifiziert. Der entsprechende Cluster, der diese Hauptebene umfasst, repräsentiert somit die relevanten Datenpunkte des Bauteils.

Berechnung der Schlitztiefe

Zur Berechnung der **Schlitztiefe** wird eine Ebene an die Datenpunkte des Zapfens angepasst. Diese Datenpunkte können aus den Ausreißern des Clusters vom Bauteil extrahiert werden. Diese Ebene ist parallel zur Hauptebene, da sie denselben Normalenvektor wie die Hauptebene besitzt. In Abbildung 5.8 sind die Punkte der Hauptebene in Blau und die Punkte am Zapfen in Grün dargestellt.

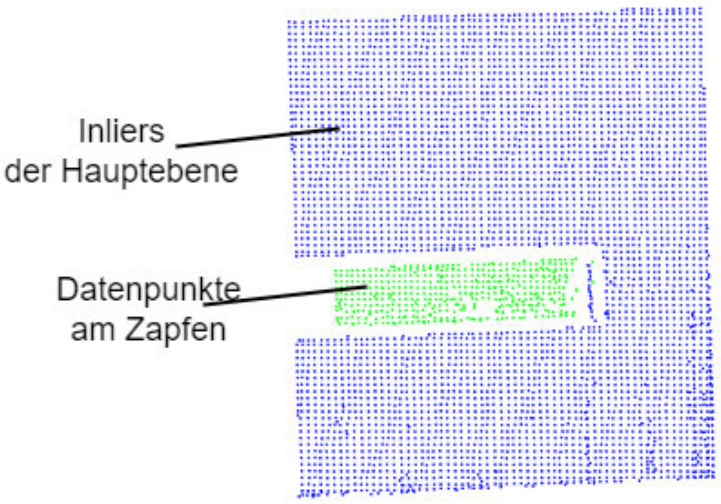


Abbildung 5.8: Berechnung der Schlitztiefe

Der Abstand zwischen einem Punkt auf der Hauptebene und der Ebene des Zapfens wird als Tiefe des Schlitzes definiert. Dieser Wert wird als ein Attribut des Schlitzes anschließend gespeichert.

5.2.3 Verarbeitung

Der Hauptzweck der Verarbeitung besteht darin, die Randpunkte der Hauptebene zu finden und dadurch die Punkte zu finden, die zur medialen Achse des Schlitzes gehören. Um die Umsetzung zu vereinfachen, konzentriert sich die Verarbeitung ausschließlich auf die Inliers der Hauptebene. Der Ablauf der Verarbeitung wird in Abbildung 5.9 dargestellt.

Down-Sampling

Die Punktwolke der Hauptebene wird mithilfe von Voxel-Down-Sampling weiter reduziert, um die Anzahl der Punkte zu verringern und die Verarbeitung zu beschleunigen, ohne dabei wesentliche Merkmale zu verlieren.

Bestimmung der Randpunkte

Sobald die Hauptebene und ihre Inliers bestimmt werden, können die Randpunkte dieser Ebene mithilfe der Methode **der größten Winkellücke** extrahiert werden.

Ein k-d-Baum für die Punktwolke der Hauptebene wird mithilfe der Klasse `KDTreeFlann` der *open3D*-Bibliothek erstellt. Diese Klasse nutzt die FLANN-Bibliothek (*Fast Library for Approximate Nearest Neighbors*) [29], um effizient k-d-Bäume zu generieren und dadurch eine schnelle Suche nach den nächsten Nachbarn in der Punktwolke zu ermöglichen.

Die Implementierung der Methode der größten Winkellücke berechnet den maximalen Winkelabstand zwischen den Richtungsvektoren eines Abfragepunkts und seiner Nachbarn. Die Suche nach den nächsten Nachbarn erfolgt dabei mithilfe der Nachbarn im festen Radius (*Fixed-radius near neighbors*) (FRNN). Zur Bestimmung der Winkellücken werden die Winkel der Vektoren relativ zur x-Achse mithilfe **arctan2** berechnet und anschließend sortiert. Der größte Abstand wird ermittelt, indem die Differenzen zwischen aufeinanderfolgenden Winkeln sowie der Wrap-Around-Winkel (zwischen dem letzten und dem ersten Winkel) berücksichtigt werden. Dieser größte Abstand, bekannt als größte Winkellücke, wird verwendet, um Randpunkte in einer Punktwolke zu identifizieren.

Zur Bestimmung der Randpunkte in der Hauptebene wird ein Schwellenwert von **75 Grad ($\frac{5\pi}{12}$)** gewählt, um sowohl Randpunkte als auch konvexe und konkave Ecken zu erkennen. In Abbildung 4.7 sind die erkannten Randpunkte der Hauptebene in Zyan dargestellt.

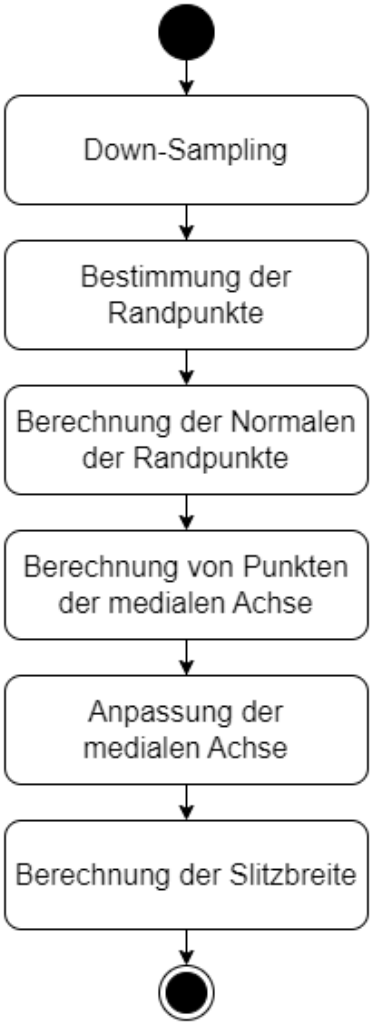


Abbildung 5.9: Ablauf der Vorverarbeitung im `SearchingSlitDetector`

Berechnung der Normalen der Randpunkte

Nachdem die Randpunkte der Hauptebene identifiziert worden sind, besteht der nächste Schritt darin, die Normalen dieser Randpunkte zu berechnen. Die Normale eines jeden Randpunktes steht senkrecht zu den benachbarten Randpunkten und liegt auf der Hauptebene. Besonders wichtig ist, dass diese Normalen in Bezug auf ihre Nachbarn auf der Hauptebene **nach außen** gerichtet sind. In Abbildung 5.10 werden die Normalen der Randpunkte dargestellt. Diese ausgerichteten Normalen bilden die Grundlage für die Berechnung von Punkten der medialen Achsen.

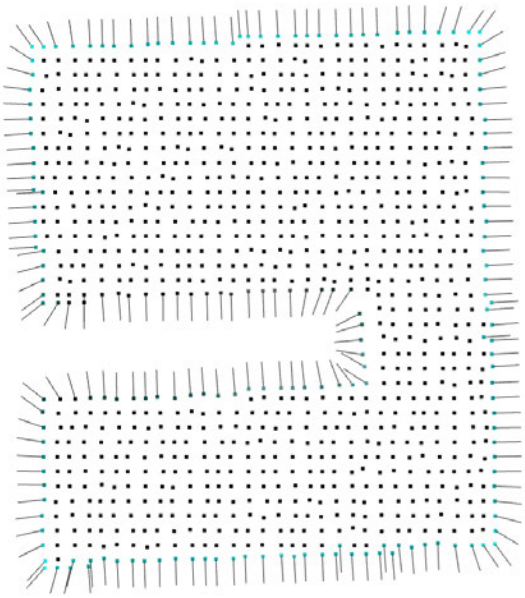


Abbildung 5.10: Normalen der Randpunkte

Berechnung von Punkten der medialen Achse

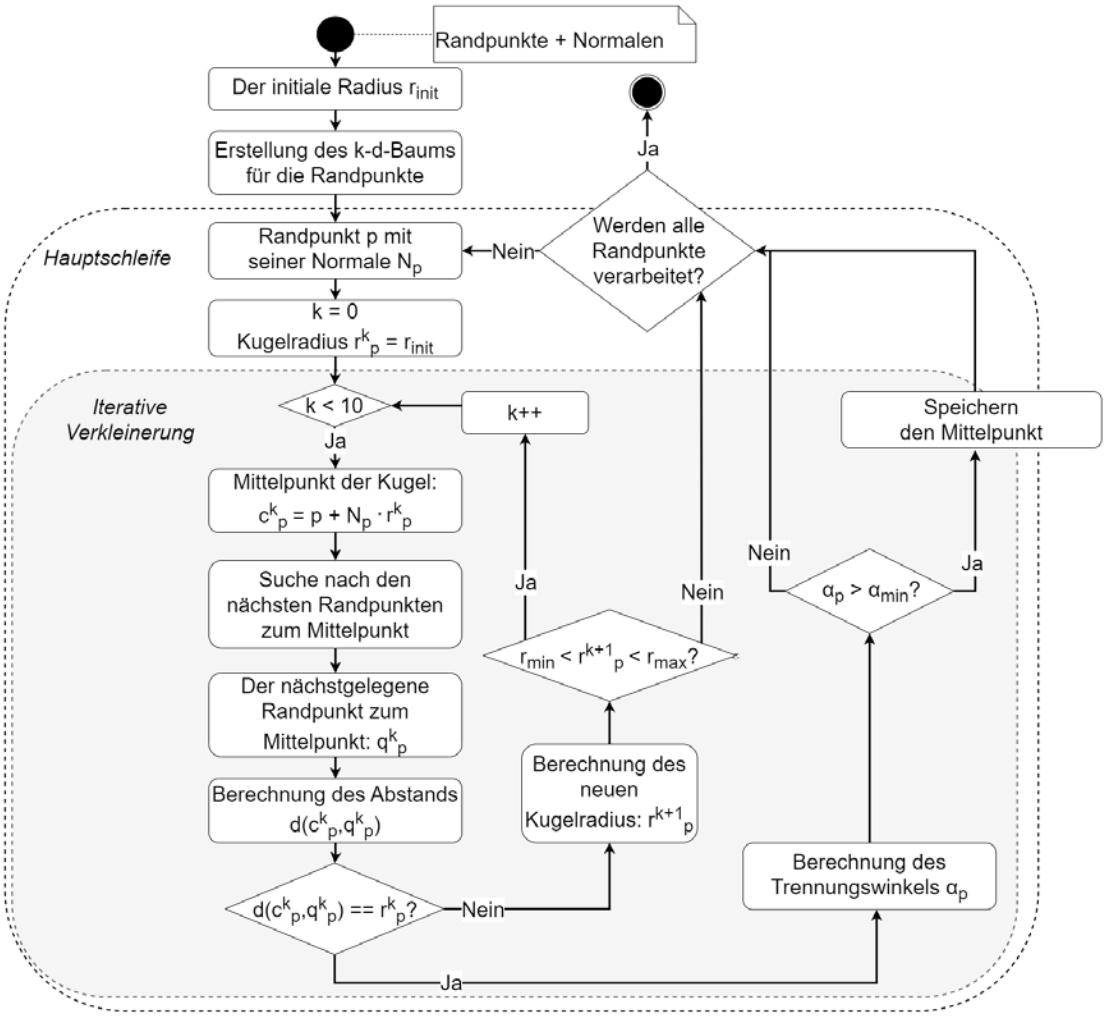


Abbildung 5.11: Ablauf der Berechnung von Punkten der medialen Achse

Inspiriert von dem Algorithmus, der von Ma et al. in [26] vorgeschlagen wird, sowie dem Ansatz von Widyaningrum et al. in [51], wird der Algorithmus zur Berechnung der Punkte der medialen Achse implementiert. Dabei werden einige kleinere Anpassungen vorgenommen, um die Berechnung an die spezifischen Anforderungen der medialen Punkte des Schlitzes in der Hauptebene anzupassen. Der Ablauf dieser Berechnung wird in der entsprechenden Abbildung 5.11 veranschaulicht und beschreibt die schrittweise Bestimmung der medialen Achse basierend auf den zuvor berechneten Randpunkten und Normalen.

Nachfolgend wird der Algorithmus zur Berechnung der Punkte der medialen Achse durch iterative Reduzierung des Kugelradius beschrieben:

a, Initialisierung

- Der initiale Radius r_{init} wird als maximale Schlitzbreite definiert, die für alle Randpunkte bestimmt wird.
- Es wird ein k-d-Baum für die Randpunkte erstellt, um die Zeit bei der Suche der Nachbarschaft zu reduzieren.

b, Hauptschleife Für jeden Randpunkt p mit seiner Normale N_p:

- Startwerte setzen: Der Kugelradius r_p wird auf r_{init} gesetzt.
- Iterative Verkleinerung (max. 10 Schritte, $k = 1, 2, \ldots, 10$):
  1. Berechnung des Kugelmittelpunkts: $c_p^k = p + N_p \cdot r_p^k$
  2. Suche nach den nächsten Randpunkten: Suche nach den nächstgelegenen Randpunkten innerhalb eines festen Radius von $1,5 \cdot r_p^k$ (FRNN) zum Mittelpunkt unter Verwendung des k-d-Baums.
  3. Der nächstgelegene Randpunkt: Bestimme den Randpunkt q_p^k, der dem Mittelpunkt am nächsten liegt und sich von p unterscheidet.
  4. Berechnung und Überprüfung des Abstands: Prüfe, ob der Abstand zwischen dem nächstgelegenen Randpunkt und dem Mittelpunkt $d(c_p^k, q_p^k)$ gleich dem Kugelradius r_p^k ist:
     - Wenn nein:
       * Berechne einen neuen Radius:

$$r_p^{k+1} = \frac{d(p, q_p^k)}{2 \cdot cos(\vartheta_p^k)}$$

Wobei: ϑ_p^k der Winkel zwischen der Normalenrichtung N_p und der Richtung des Vektors, der von q_p^k zu p zeigt.

* Überprüfe, ob der neue Radius r_p^{k+1} innerhalb der Grenzen (r_{min} als Hälfte des Minimums und r_{max} als Hälfte des Maximums der Schlitzbreite) liegt:
  - Wenn ja: fahre mit der nächsten Iteration der Verkleinerung fort.
  - Wenn nein: brich die Schleife für diesen Punkt ab und fahre mit dem nächsten Punkt der Hauptschleife fort.

– Wenn ja:

* Berechne den Trennungswinkel α_p
  (Winkel zwischen Kugelmittelpunkt c_p^k und Randpunkten q_p^k und p)
* Überprüfe, ob der Winkel α_p den Schwellenwert α_{min} von **135 Grad** überschreitet:
  - Wenn ja: Markiere den Mittelpunkt c_p^k als medialen Punkt, speichere den Radius r_p^k und fahre mit dem nächsten Randpunkt fort.
  - Wenn nein: fahre direkt mit dem nächsten Randpunkt der Hauptschleife fort.

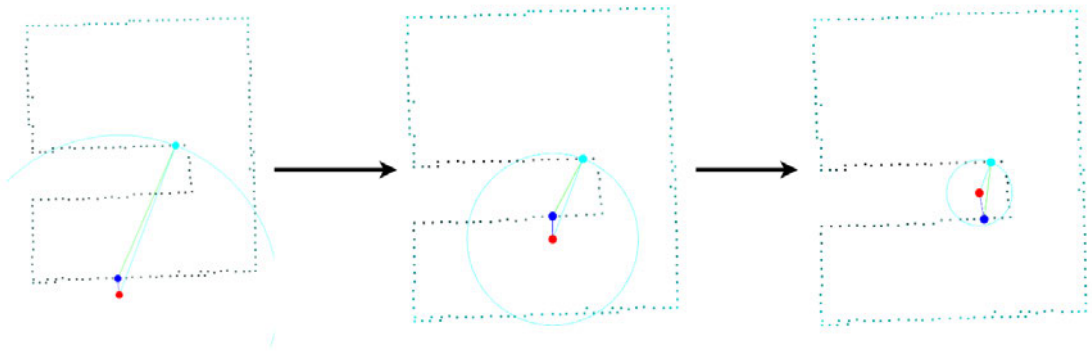


Abbildung 5.12: Visualisierung der iterativen Reduzierung des Kugelradius

Die Abbildung 5.12 zeigt eine iterative Reduzierung des Kugelradius entlang der Normalenrichtung, um einen medialen Punkt zu bestimmen. Die Mittelpunkte der Kugeln sind in Rot dargestellt, die Abfragepunkte in Zyan und die nächstgelegenen Randpunkte in Blau. In Abbildung 5.13 werden die gesamten berechneten Punkte der medialen Achse in Rot angezeigt.

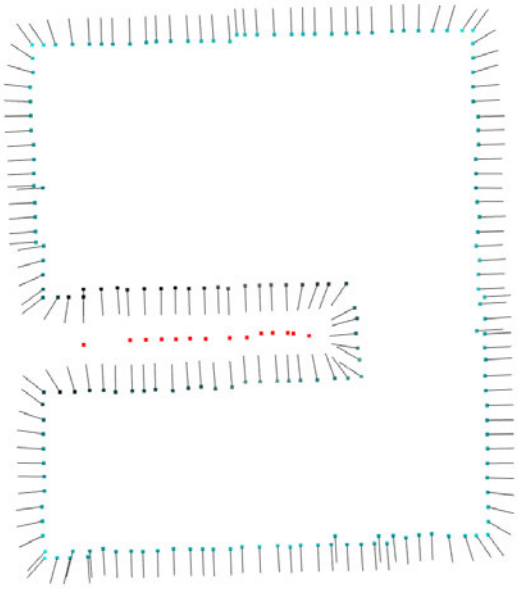


Abbildung 5.13: Die berechneten Punkte der medialen Achse

Anpassung der medialen Achse

Nach der Berechnung der Punkte der medialen Achse wird mithilfe des RANSAC-Algorithmus eine Linie an diese Punkte angepasst. Der Punkt `line_point`, der die Linie repräsentiert, wird als Durchschnitt der berechneten medialen Punkte bestimmt. Die Richtung der Linie `line_direction` wird an der Y-Achse der Sensorposen ausgerichtet, um eine konsistente Orientierung sicherzustellen. Die berechnete mediale Achse wird als Attribut des Schlitzes gespeichert und steht für die Online-Verfolgung im weiteren Prozess zur Verfügung. In Abbildung 4.9 wird die mediale Achse des Schlitzes in Rot angezeigt.

Berechnung der Schlitzbreite

Basierend auf den zuvor berechneten Punkten der medialen Achse und den zugehörigen Radien wird der Durchschnittswert dieser Radien berechnet. Der doppelte Wert des Durchschnitts wird als Breite des Schlitzes definiert und als Attribut des Schlitzes gespeichert. Dieses Attribut dient zur Beschreibung der Geometrie des Schlitzes und steht für nachfolgende Prozesse zur Verfügung.

5.2.4 Generierung der Zielpunkte

Sobald die mediale Achse des Schlitzes bestimmt ist, besteht der nächste Schritt darin, Zielpunkte entlang dieser Linie zu generieren. Der Ablauf zur Generierung der Zielpunkte wird in Abbildung 5.14 dargestellt.

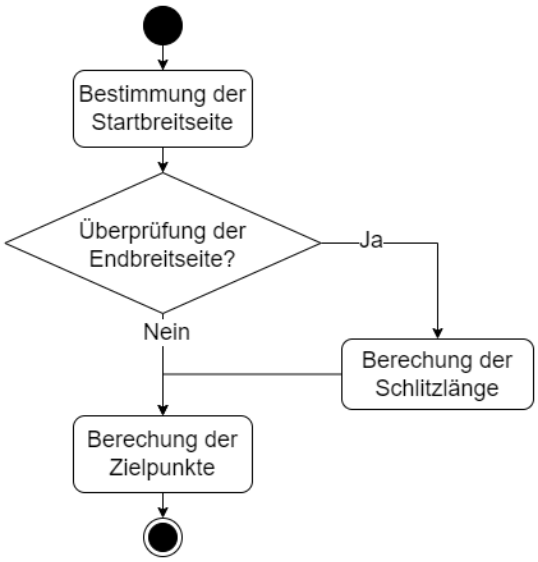


Abbildung 5.14: Ablauf der Generierung der Zielpunkte im `SearchingSlitDetector`

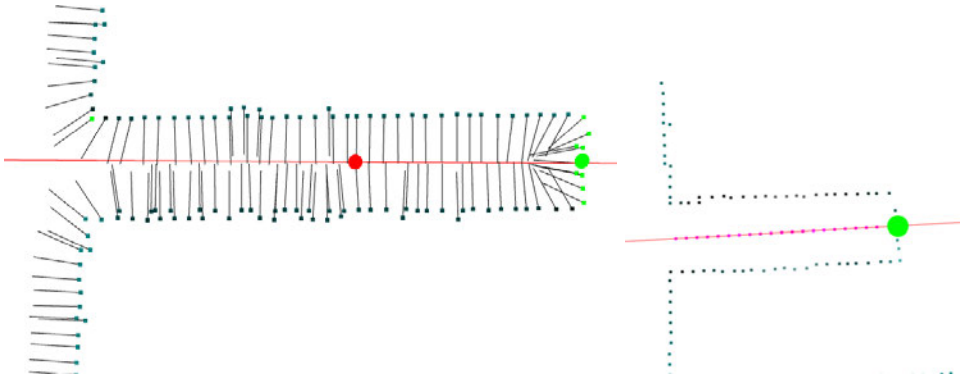
Bestimmung der Startbreitseite

Vor der Generierung von Zielpunkten müssen die Datenpunkte identifiziert werden, die zur Startbreitseite des Schlitzes gehören. Anhand der Geometrie können diese Punkte durch folgende Eigenschaften bestimmt werden:

- Der Abstand der zur Startbreitseite gehörenden Datenpunkte zur medialen Achse ist kleiner als der zuvor berechnete Radiuswert.
- Der Vektor, der vom Punkt auf der medialen Achse (`line_point`) zu einem Punkt auf der Startbreitseite zeigt, weist dieselbe Richtung wie die zugehörige Normale des Punktes auf.

Basierend auf diesen beiden Eigenschaften können die Punkte der Startbreitseite des Schlitzes extrahiert werden. In Abbildung 5.15a werden die zur Startbreitseite gehörenden Datenpunkte in Grün dargestellt.

Nachdem die Punkte der Startbreitseite identifiziert worden sind, wird eine Ebene an diese Punkte angepasst. Die Normale dieser Ebene entspricht der Richtung der medialen Achse `line_direction`. Diese Ebene ermöglicht es, den Schnittpunkt zwischen der medialen Achse und der Startbreitseite zu bestimmen. Dieser Schnittpunkt wird als **Startmittelpunkt** definiert und dient als Grundlage für die Berechnung des ersten Zielpunkts. Darüber hinaus kann der Startmittelpunkt mit dem **Endmittelpunkt** kombiniert werden, um die Gesamtlänge des Schlitzes zu berechnen. In Abbildung 5.15a wird dieser Punkt durch eine grüne Kugel visualisiert.



(a) (b)

Abbildung 5.15: Generierung der Zielpunkte
a) die Punkte der Startbreitseite in Grün
b) Die generierten Zielpunkte in Violett

Überprüfung der Endbreitseite

Ähnlich wie bei der Identifikation der Punkte, die zur Startbreitseite gehören, werden auch die Punkte der Endbreitseite des Schlitzes gesucht, ob diese Punkte in der Punktwolke vorhanden sind. Sobald diese Punkte erkannt sind, kann die Länge des Schlitzes berechnet werden.

Dieser Ansatz ist besonders nützlich, wenn die Länge der Scanstrecke so eingestellt ist, dass der gesamte Schlitz bereits im initialen Scan erfasst wird, oder wenn die Schlitzlänge so klein ist, dass keine Online-Erkennung erforderlich ist. In diesen Fällen werden die Zielpunkte direkt zwischen dem Startmittelpunkt und dem Endmittelpunkt des Schlitzes generiert, wodurch die Trajektorienplanung effizienter gestaltet wird.

Berechnung der Zielpunkte

Die Zielpunkte werden zwischen dem Start- und Endpunkt des Schlitzes berechnet. Dabei werden die Punkte durch lineare Interpolation gleichmäßig verteilt, wobei der Abstand zwischen den einzelnen Zielpunkten **1 mm** beträgt. Die Orientierung jedes Zielpunkts wird entsprechend der Normale der Hauptebene festgelegt, um sicherzustellen, dass der Schweißbrenner während des Schweißens senkrecht zur Hauptebene ausgerichtet bleibt. Zusätzlich wird jedem Zielpunkt ein eindeutiger Index zugeordnet, bevor die Zielpunkte veröffentlicht werden, um ihre Reihenfolge und Zuordnung der Trajektorie eindeutig zu definieren.

Im `SearchingSlitDetector` wird der **Startpunkt** standardmäßig mit einem Versatz vom Standard-Offset zum Startmittelpunkt festgelegt. Falls der Endmittelpunkt nicht erkannt wird, wird der **Endpunkt** durch den Schnittpunkt zwischen der medialen Achse und der X-Z-Ebene im Sensor-Koordinatensystem der letzten Sensorpose aus den gesammelten Datenzeilen bestimmt. Die generierten Zielpunkte werden in Abbildung 5.15b in Violett dargestellt.

5.2.5 Validierung der Zielpunkte

Im `SearchingSlitDetector` muss die Anzahl der Zielpunkte vor der Veröffentlichung mindestens eine minimale Zahl betragen. Dadurch wird sichergestellt, dass diese Zielpunkte mithilfe von B-Spline im `PathDataHandler` interpoliert werden können.

5.3 `FollowingSlitDetector`

Basierend auf dem entwickelten Konzept in Abschnitt 4.2 und inspiriert durch den von Zhou et al. in [58] vorgeschlagenen Algorithmus zur Extraktion von I-Schweißnähten wird der `FollowingSlitDetector` implementiert. Der `FollowingSlitDetector` ist dafür verantwortlich, während der Online-Verfolgung die Kontur des Schlitzes weiterhin zu erkennen und fortlaufend die nächsten Zielpunkte zu generieren. Die Abbildung 5.16 zeigt den allgemeinen Ablauf der Schlitz-Erkennung im `FollowingSlitDetector`.

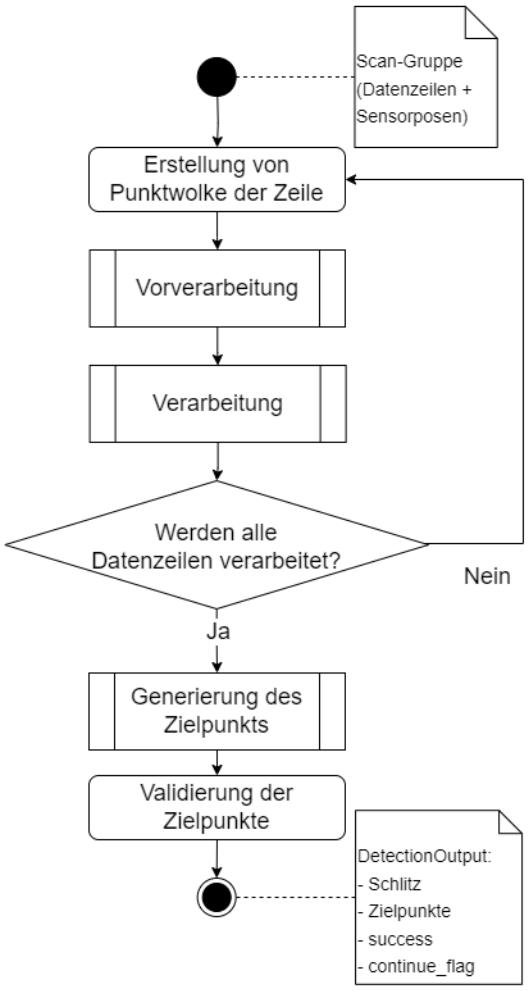


Abbildung 5.16: Der allgemeine Ablauf der Schlitzerkennung im `FollowingSlitDetector`

Vor der Datenverarbeitung wird die Konsistenz der Sensorposen überprüft, indem die Werte entlang der Y-Achse validiert werden.
Da die Punktwolke in `taskMain` schmal ist und nur eine begrenzte Anzahl von Datenzeilen enthält, erfolgt die Verarbeitung zunächst zeilenweise. Für jede Datenzeile wird der Punkt der medialen Achse berechnet.

5.3.1 Vorverarbeitung

Jede Datenzeile wird als Open3D-Geometry-PointCloud repräsentiert. Im nächsten Schritt wird Voxel-Down-Sampling angewendet, um die Anzahl der Punkte zu reduzieren, damit der Rechenaufwand in den nachfolgenden Schritten verringert wird. Der Ablauf der Vorverarbeitung jeder Zeile wird in Abbildung 5.17 dargestellt.

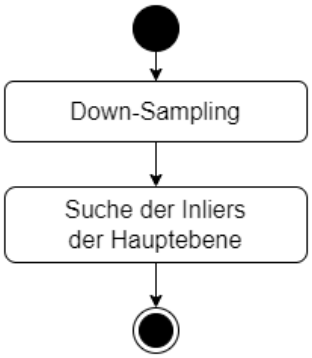


Abbildung 5.17: Ablauf der Vorverarbeitung im `FollowingSlitDetector`

Anschließend werden die Punkte der Hauptebene extrahiert. Das Modell der Hauptebene wird zuvor im `SearchingSlitDetector` während des `taskApproach` definiert. Für jeden Punkt der Datenzeile wird der Abstand zur Hauptebene berechnet und überprüft. Punkte, deren Abstand innerhalb eines festgelegten Schwellenwerts liegt, werden als Punkte der Hauptebene betrachtet und zur weiteren Verarbeitung ausgewählt. In Abbildung 5.18 werden die Punkte der Hauptebene einer Datenzeile in Blau angezeigt.

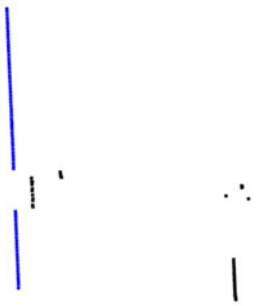


Abbildung 5.18: Punkte der Hauptebene von einer Datenzeile in Blau

5.3.2 Verarbeitung

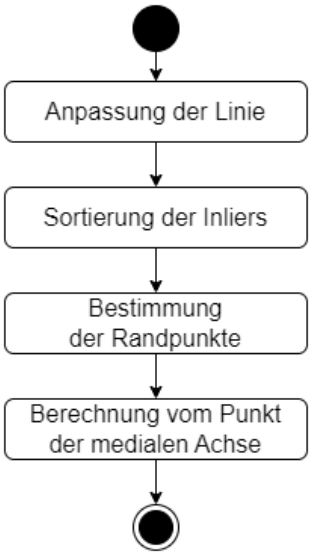


Abbildung 5.19: Ablauf der Verarbeitung im `FollowingSlitDetector`

In der Verarbeitung werden die Punkte der Hauptebene jeder Datenzeile analysiert, um den Punkt der medialen Achse zu berechnen. Der Ablauf der Verarbeitung jeder Zeile wird in Abbildung 5.19 dargestellt. Die Verarbeitung umfasst die folgenden Schritte:

Anpassung der Linie:

Die Punkte der Hauptebene werden mithilfe des RANSAC-Algorithmus an ein Linienmodell angepasst, um die Inliers dieser Linie zu extrahieren.

Sortierung der Inliers:

Die Inliers der Linie werden nach ihrer Richtung zum Mittelpunkt der Linie geordnet und anhand ihrer Distanz sortiert.

Bestimmung der Randpunkte:

Der Abstand zwischen zwei benachbarten Punkten wird mit der Schlitzbreite verglichen, die zuvor im `SearchingSlitDetector` definiert wird. Punkte, deren Abstand zueinander diesem Wert gleich ist, werden als Randpunkte identifiziert.

Berechnung des Punktes der medialen Achse:

Der Mittelpunkt der beiden identifizierten Randpunkte wird berechnet und als Punkt der medialen Achse definiert.

Dieser Punkt der medialen Achse legt die Grundlage für die Zielpunktgenerierung im weiteren Verlauf des Prozesses. Die Abbildung 5.20 zeigt die Verarbeitung einer Datenzeile zur Berechnung des Punktes der medialen Achse, wobei die Punkte der Hauptebene (blau), die angepasste Linie (blau), die identifizierten Randpunkte (zyan) und der berechnete Punkt der medialen Achse (rote Kugel) dargestellt sind.

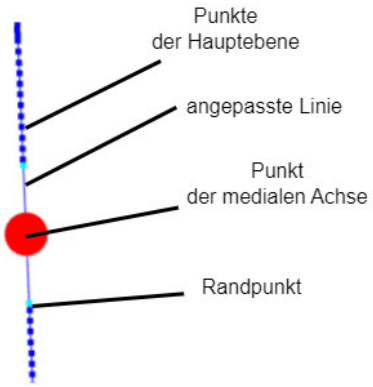


Abbildung 5.20: Berechnung des Punktes der medialen Achse einer Datenzeile

5.3.3 Generierung der Zielpunkte

Nachdem alle Datenzeilen verarbeitet und die Punkte der medialen Achse berechnet worden sind, erfolgt im nächsten Schritt die Generierung der Zielpunkte. Der Ablauf zur Generierung der Zielpunkte wird in Abbildung 5.21 dargestellt. Dieser Prozess umfasst die folgenden Schritte:

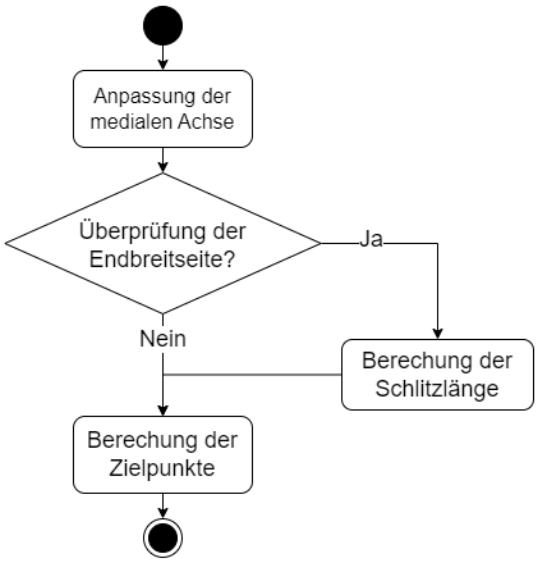


Abbildung 5.21: Ablauf zur Generierung der Zielpunkte im `FollowingSlitDetector`

Anpassung der medialen Achse:

Die mediale Achse wird mithilfe der neu berechneten Punkte der medialen Achse sowie der fünf zuletzt generierten Zielpunkte durch die Methode der kleinsten Quadrate angepasst. Diese Methode wird aufgrund ihrer hohen Berechnungsgeschwindigkeit verwendet, um eine präzise Anpassung der Kontur des Schlitzes sicherzustellen.

Überprüfung der Endbreitseite:

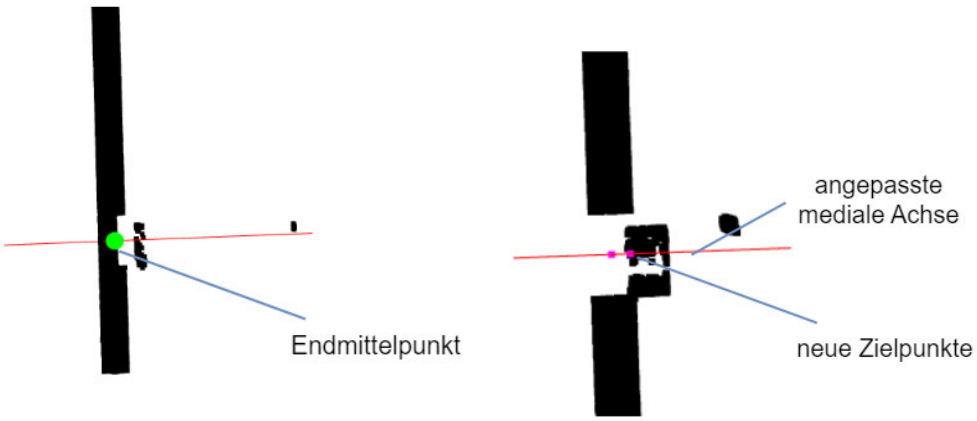
Wenn in einer Datenzeile keine Randpunkte gefunden werden, wird diese Datenzeile als die Endbreitseite interpretiert. Für die Punkte dieser Datenzeile wird eine Ebene angepasst, wobei der Normalvektor der medialen Achse verwendet wird. Der **Endmittelpunkt** wird als der Schnittpunkt zwischen der medialen Achse und dieser Ebene definiert. Die Länge des Schlitzes wird anschließend als Abstand zwischen dem Endmittelpunkt und dem Startmittelpunkt berechnet. In Abbildung 5.22a wird der Endmittelpunkt in Grün angezeigt.

Berechnung der Zielpunkte:

Die erste und letzte Zielposition werden wie folgt bestimmt:

- Erste Zielposition: Sie wird als der Schnittpunkt der medialen Achse mit der X-Z-Ebene der ersten Sensorpose berechnet.
- Letzte Zielposition:
  - Wenn der Endmittelpunkt erkannt wird, wird die letzte Zielposition als Endpunkt der medialen Achse minus einem Standard-Offset entlang der Achse definiert.
  - Falls der Endmittelpunkt nicht erkannt wird, wird stattdessen der Schnittpunkt der medialen Achse mit der X-Z-Ebene der letzten Sensorpose verwendet.

Die Berechnung der Zielpunkte zwischen der ersten und letzten Zielposition erfolgt weiterhin wie in 5.2.4 beschrieben. In Abbildung 5.22a werden die generierten Zielpunkte in Violett angezeigt.



(a) (b)

Abbildung 5.22: Visualisierung
a) Des Endmittelpunkts in Grün
b) Der generierten Zielpunkte in Violett

5.3.4 Validierung der Zielpunkte

Um sicherzustellen, dass die Trajektorie in die richtige Richtung verläuft, werden die neu generierten Zielpunkte anhand spezifischer Kriterien in Bezug auf Geometrie und Ausrichtung mit den zuvor generierten Zielpunkten verglichen. Diese Kriterien umfassen:

- **Winkel zwischen den Orientierungen:** Der Winkel zwischen der Orientierung eines neuen Zielpunkts und der des vorherigen Zielpunkts muss innerhalb eines definierten Schwellenwerts liegen.
- **Abstand zum letzten Zielpunkt:** Der Abstand zwischen dem neuen Zielpunkt und dem letzten Zielpunkt darf nicht unter oder über einem festgelegten Schwellenwert liegen, um eine gleichmäßige Punktverteilung zu gewährleisten.
- **Abstand zum initialen Zielpunkt im Vergleich zum letzten Zielpunkt:** Der neue Zielpunkt darf nicht zu stark vom Verlauf der Trajektorie abweichen, gemessen am Verhältnis seines Abstands zum initialen und zum letzten generierten Zielpunkt.

Nur Zielpunkte, die alle genannten Kriterien erfüllen, werden in die Trajektorie aufgenommen. Diese Zielpunkte erhalten fortlaufende Indizes, bevor sie veröffentlicht werden.

5.4 `TaskSlitFollower`

`TaskSlitFollower` erbt die Basisklasse `TaskApplier` entsprechend der beschriebenen Softwarearchitektur und dem Ablauf des Schweißprozesses, wie in den Unterabschnitten 2.2.2 und 2.2.3 erläutert. In `TaskSlitFollower` werden die Dienste von `online_feature_detection_node` verbunden, um die Schlitzerkennung während des Schweißprozesses zu steuern, wie im Unterabschnitt 5.1.1 beschrieben. Die Umsetzung der vier Aufgaben in `TaskSlitFollower` erfolgt wie folgt:

5.4.1 `taskInitialize`

Die Aufgabe `taskInitialize` umfasst zwei wesentliche Schritte, wie in der Abbildung 5.23 dargestellt:

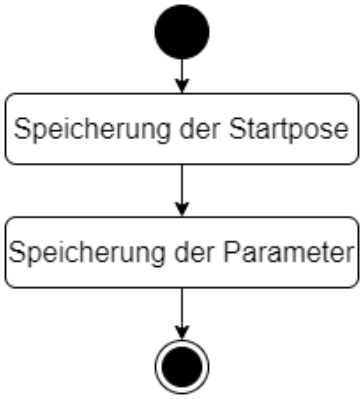


Abbildung 5.23: Ablauf von `taskInitialize`

Speicherung der Startpose:

Die aktuelle Position und Orientierung des Roboters werden als Startpose gespeichert, um als Ausgangspunkt für die nachfolgenden Aufgaben zu dienen.

Speicherung der Parameter:

Relevante Parameter, wie z. B. Schweißgeschwindigkeit, Länge der Drahtelektrode und Schweißwinkel, werden konvertiert und für den weiteren Prozess bereitgestellt.

5.4.2 `taskApproach`

Die Abbildung 5.24 zeigt die nachfolgenden Schritte der Aufgabe `taskApproach`:

`startDetection`:

Diese Aufgabe beginnt mit dem Aufruf des Dienstes `startDetection`, um die Schlitzerkennung zu initiieren.

Bewegung zum Scannen:

Der Endeffektor zusammen mit dem Scanner bewegt sich entlang der grünen Laserlinie über eine kurze Strecke von **5 cm**.

Warten auf Zielpunkte:

Der Roboter wartet in einer begrenzten Zeit, bis die ersten Zielpunkte durch den Detektor generiert werden:

- Wenn Zielpunkte generiert werden:
  Der Dienst `pauseDetection` wird aktiviert, und der TCP bewegt sich zum ersten Zielpunkt, um den Schweißprozess vorzubereiten. Der Rückgabewert und die Fortsetzungsflagge (`continue_flag`) werden auf **True** gesetzt, sodass `taskMain` anschließend fortgesetzt werden kann.
- Wenn keine Zielpunkte generiert werden:
  Der Dienst `endDetection` wird aktiviert, und der TCP kehrt zur Startpose zurück. Der Rückgabewert und die Fortsetzungsflagge (`continue_flag`) werden auf **False** gesetzt, sodass der Schweißprozess abgebrochen wird.

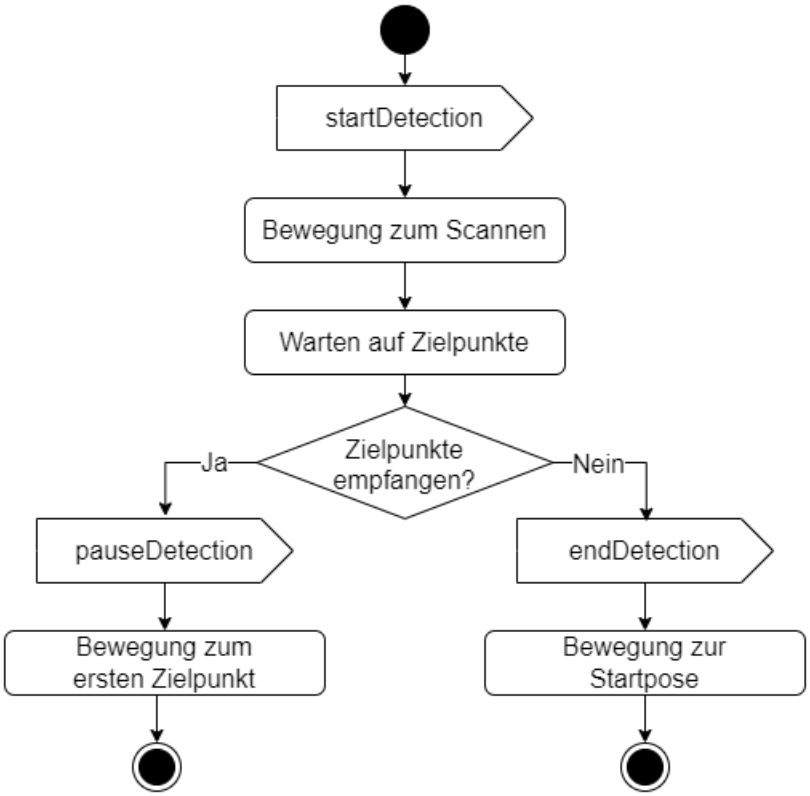


Abbildung 5.24: Ablauf von `taskApproach`

5.4.3 `taskMain`

Die Aufgabe `taskMain` beschreibt die Hauptschleife zur Online-Verfolgung des Schlitzes während des Schweißprozesses, wie in der Abbildung 5.25 dargestellt:

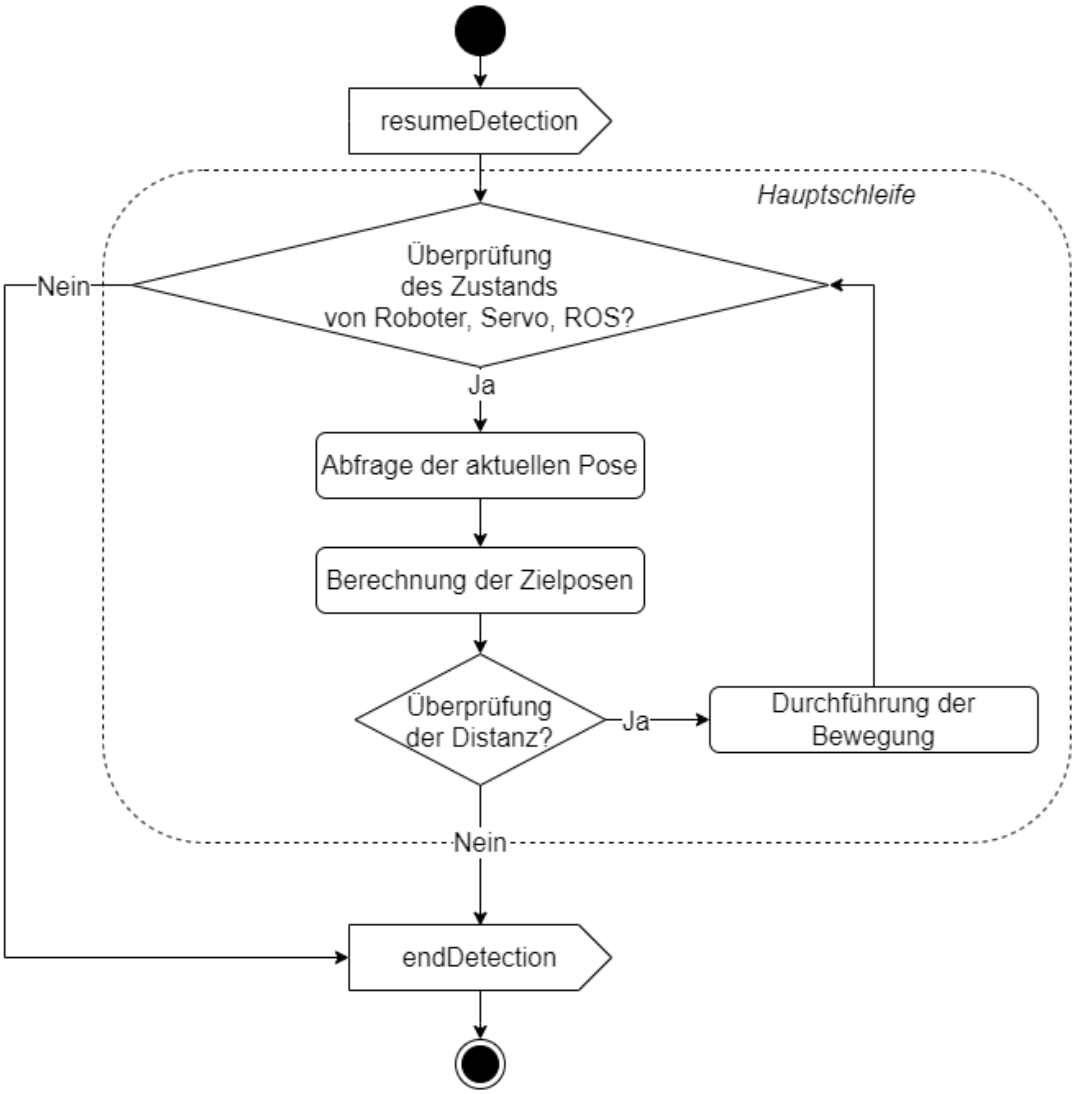


Abbildung 5.25: Ablauf von `taskMain`

`resumeDetection`

Die Aufgabe beginnt mit der Wiederaufnahme der Schlitzerkennung.

Dann werden die folgenden Schritte in einer Schleife ausgeführt:

Überprüfung des Zustands:

Der Zustand des Roboters, der Servos und des ROS-Systems wird geprüft, um sicherzustellen, dass alle Komponenten betriebsbereit sind.

Abfrage der aktuellen Pose:

Die aktuelle Pose des Roboters wird abgefragt, um die Position für die Berechnung der nächsten Zielpose zu bestimmen.

Berechnung der Zielposen:

Basierend auf der aktuellen Pose und den generierten Zielpunkten werden neue Zielposen generiert. Die Sammlung und Interpolation der Zielpunkte erfolgt kontinuierlich im `PathDataHandler`, wobei ein B-Spline verwendet wird, um eine glatte Trajektorie basierend auf den über das Thema `/taskpoints` empfangenen Zielpunkten zu erzeugen. Der `PoseGenerator` übernimmt die Verantwortung, die nächste geeignete Pose zu berechnen, basierend auf der aktuellen Position des TCPs und den interpolierten Zielpunkten aus dem `PathDataHandler`.

Überprüfung der Distanz:

Der Abstand zwischen der aktuellen Pose und der berechneten Zielpose wird überprüft. Damit soll sichergestellt werden, dass sich der TCP mit der entsprechenden Geschwindigkeit zur Endpose bewegen kann.

Bewegung:

Die Trajektorien- und Bahnplanung für TCP wird durch die spezielle Pipeline von **MoveIt** ausgeführt, bevor sie an den *UR ROS*-Treiber gesendet wird, um den Roboter in der realen Umgebung zu bewegen.

`endDetection`

Die Schleife wird beendet, sobald entweder ein Fehler im Zustand des Roboters, der Servos oder des ROS-Systems auftritt oder der TCP die letzte Zielpose erreicht hat. Der Dienst `endDetection` wird aufgerufen, um sicherzustellen, dass die Schlitzerkennung deaktiviert wird.

5.4.4 `taskExit`

Die Aufgabe `taskExit` wird implementiert, um einen sicheren und geordneten Abschluss des Prozesses zu gewährleisten. Wie in Abbildung 5.26 dargestellt, umfasst der Ablauf folgende Schritte:

Abschaltung:

Der Roboter und die entsprechenden Komponenten des Systems werden kontrolliert heruntergefahren, um den Betrieb sicher zu beenden.

`endDetection`:

Der Dienst `endDetection` wird aufgerufen, um die Schlitzerkennung ordnungsgemäß zu deaktivieren und sicherzustellen, dass keine weiteren Daten verarbeitet oder Zielpunkte generiert werden.

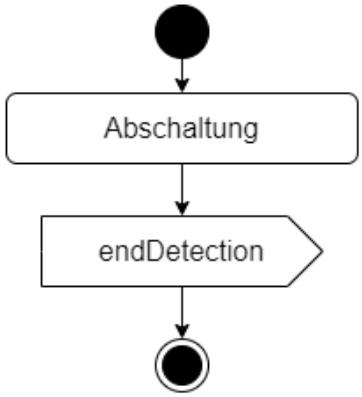


Abbildung 5.26: Ablauf von `taskExit`

Diese Aufgabe wird auch aufgerufen, wenn der Benutzer den Prozess manuell unterbrechen möchte, falls ein Problem auftritt, um die Sicherheit und Stabilität des Systems zu gewährleisten.

6 Evaluierung

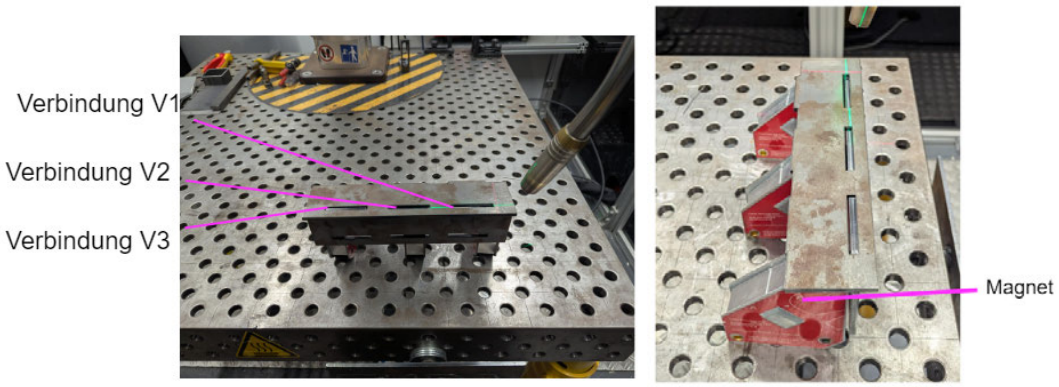
In diesem Kapitel wird das entwickelte und implementierte Konzept evaluiert. Dazu wird das Konzept in die bestehende System- und Softwarearchitektur integriert und unter verschiedenen Bedingungen getestet. Anschließend erfolgt die Validierung der Lösung anhand der zuvor definierten Anforderungen.

6.1 Funktionale Tests

Die funktionalen Tests prüfen die Fähigkeit des Systems, unter verschiedenen Bedingungen korrekte Ergebnisse zu liefern. Hierbei werden mehrere Aspekte der Schlitzerkennung und Zielpunktgenerierung evaluiert.

6.1.1 Testumgebung

Der für die Tests verwendete Rechner ist mit einem **12th Gen Intel® Core™ i7-12700K Prozessor mit 20 Kernen** ausgestattet. Der Arbeitsspeicher beträgt 31,1 GiB RAM, und als Grafikeinheit wird eine integrierte **Mesa Intel® Graphics (ADL-S GT1)** verwendet. Der Rechner verfügt über eine Speicherkapazität von **3,0 TB**. Als Betriebssystem kommt Ubuntu **20.04 LTS** zum Einsatz.



(a) (b)

Abbildung 6.1: Bauteil mit 3 Schlitz-Zapfen-Verbindungen

Für die Tests des entwickelten Konzepts werden zwei unterschiedliche Arten von Bauteilen verwendet:

- Bauteil mit 3 Schlitz-Zapfen-Verbindungen:
  Dieses Bauteil besteht aus zwei identischen Metallplatten, die drei Schlitz-Zapfen-Verbindungen mit derselben **Breite von 6,5 mm** und einer **Länge von 50 mm** bilden. Die Verbindungen unterscheiden sich in der Tiefe, die jeweils **4 mm (V1)**, **2 mm (V2)** und **1 mm (V3)** beträgt, wie in Abbildung 6.1 dargestellt.

- Metallplatte mit 5 Schlitzen:
  Dieses Testbauteil besteht aus einer Metallplatte mit einer gemeinsamen **Länge von 240 mm** und einer **Plattentiefe von 5 mm**, jedoch ohne Zapfenverbindungen. Die Platte enthält fünf Schlitze mit unterschiedlichen Breiten von **5 mm (S1)**, **4 mm (S2)**, **3 mm (S3)**, **2 mm (S4)** und **1 mm (S5)**, wie in Abbildung 6.2 dargestellt.

Diese Bauteile werden ausgewählt, um das Konzept unter verschiedenen geometrischen Bedingungen zu evaluieren und die Funktionalität der Schlitzerkennung zu prüfen.

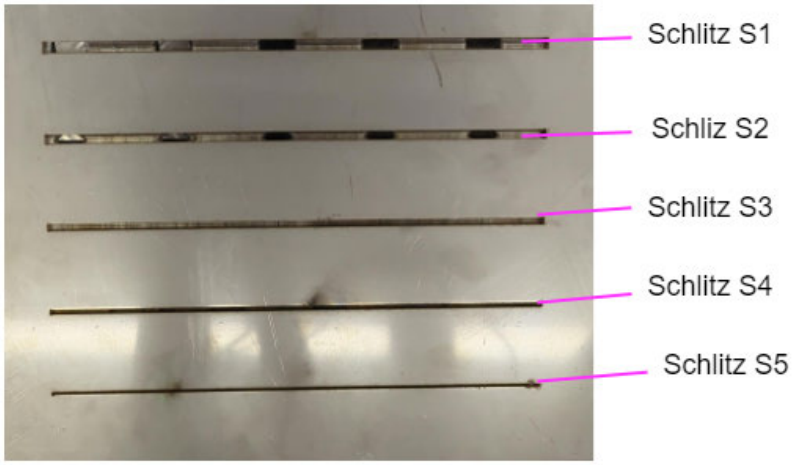


Abbildung 6.2: Metallplatte mit 5 Schlitzen

Die Abbildung 6.3 zeigt einen Überblick über die Testumgebung der Roboterzelle im Fraunhofer IPA.

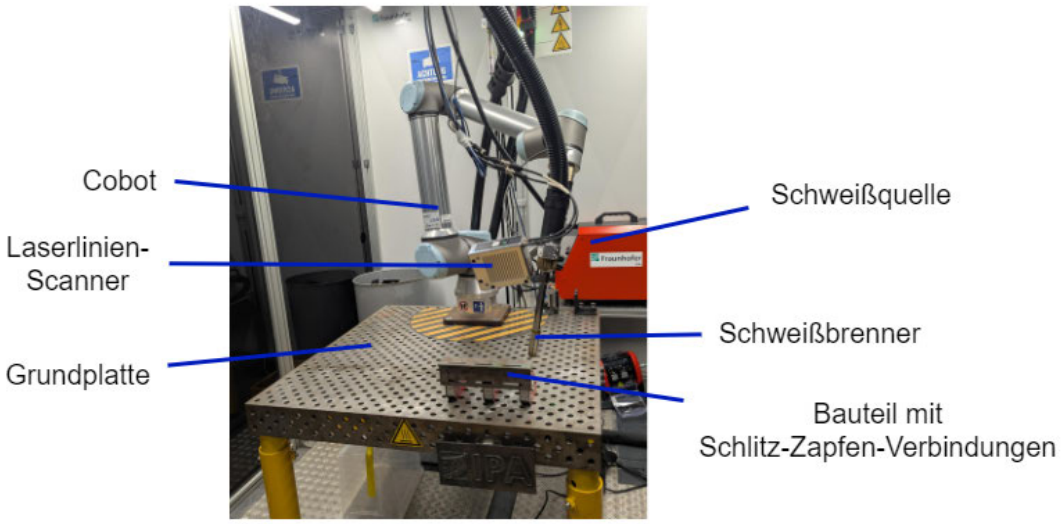


Abbildung 6.3: Testumgebung der Roboterzelle im Fraunhofer IPA

6.1.2 Test mit drei Schlitz-Zapfen-Verbindungen mit unterschiedlichen Tiefen

Um die Fähigkeit des Systems zur Erkennung von Schlitz-Zapfen-Verbindungen mit unterschiedlichen Tiefen zu überprüfen, wird ein Test mit dem **Bauteil** durchgeführt, das drei Schlitz-Zapfen-Verbindungen umfasst. Die Standard-Scangeschwindigkeit beträgt **35 mm/s**, bei einer initialen Scanstreckenlänge von **50 mm**. Darüber hinaus beträgt der Standard-Offset am Anfang und Ende des Schlitzes **2 mm**. Die Bewegungsgeschwindigkeit des Roboters während der Online-Verfolgung liegt hingegen bei **2 mm/s**. Die erfassten Ergebnisse für Tiefe, Breite, Länge, Anzahl der generierten Zielpunkte und die Bewegungsstrecke sind in Tabelle 6.1 dargestellt.

Verbindungs-nummer	**Tiefe des Schlitzes** [mm]		**Breite des Schlitzes** [mm]		**Länge des Schlitzes** [mm]		**Anzahl der Zielpunkte**	**Länge der Bewegungsstrecke** [mm]	
	Soll	*Ist*	*Soll*	*Ist*	*Soll*	*Ist*		*Soll*	*Ist*
V1	4	4,04	6,50	6,72	50	48,48	35	46	45,57
V2	3	3,03	6,50	6,79	50	48,87	35	46	45,87
V3	1	1,03	6,50	5,97	50	46,62	25	46	40,61

Tabelle 6.1: Ergebnisse der Erkennung von Schlitz-Zapfen-Verbindungen mit unterschiedlichen Tiefen

Aus Tabelle 6.1 ist Folgendes ersichtlich:

Tiefe

Die Ergebnisse zeigen, dass die Erkennung der Tiefe gut funktioniert. Die erkannten Werte (*Ist*) liegen in allen drei Verbindungen (V1, V2, V3) sehr nahe an den Sollwerten, mit einer maximalen Abweichung von 0,03 mm. Dies deutet darauf hin, dass das System zuverlässig die Tiefe der Schlitze bestimmen kann.

Breite

Die Erkennung der Breite zeigt leichte Abweichungen von den Sollwerten. Besonders bei Verbindung V3 ist die erkannte Breite mit 5,97 mm deutlich niedriger als der Sollwert von 6,50 mm. Dies weist darauf hin, dass die Breitenmessung bei bestimmten Geometrien oder Bedingungen nicht optimal ist. Die Auswertung der Breitenerkennung erfolgt weiter im Unterabschnitt 6.1.3.

Länge

Die Erkennung der Länge des Schlitzes weist in allen Verbindungen leichte Abweichungen auf. Dies könnte auf eine eingeschränkte Erfassung der Randpunkte oder eine Ungenauigkeit bei kürzeren Schlitzen hinweisen. Eine mögliche Ursache ist, dass die Daten am Zapfen oder die Lücke zwischen den Schlitz- und Zapfenseiten die Erkennung beeinflussen kann. Die Auswertung der Längenerkennung erfolgt weiter im Unterabschnitt 6.1.3 mit der Metallplatte.

Generierung der Zielpunkte

Die Generierung der Zielpunkte funktioniert gut bei Verbindungen mit Tiefe 3 mm (V2) und 4 mm (V1). Es werden ausreichend Zielpunkte in der Mitte der Breite generiert, und die Bewegungsstrecke entspricht weitgehend den Sollvorgaben mit einer leichten Abweichung.
Bei Verbindung V3 (Tiefe 1 mm) treten jedoch Probleme auf. Die Zielpunktgenerierung ist nicht stabil, und die Online-Verfolgung wird frühzeitig abgebrochen. Dies führt zu einer geringeren Anzahl von Zielpunkten (25 statt 35) und einer kürzeren Bewegungsstrecke (40,61 mm statt 46 mm). Diese Instabilität könnte durch die geringe Tiefe des Schlitzes, durch die Lücke zwischen den Schlitz- und Zapfenseiten oder eine unzureichende Punktdichte in der Erkennung verursacht werden.
Ein weiteres Problem besteht darin, dass zwischen den von `SearchingSlitDetector` und `FollowingSlitDetector` generierten Zielpunkten eine kleine Lücke auftritt, wie in Abbildung 6.4 und 6.5 dargestellt. Diese Lücke könnte durch eine Verzögerung bei der Übertragung der empfangenen Daten oder durch die Bedingung zur Validierung neuer Zielpunkte im `FollowingSlitDetector` verursacht werden. Trotz dieser Lücke bleibt die Bewegung des TCPs dank der Interpolation im `PathDataHandler` weiterhin reibungslos, wodurch die Kontinuität der Trajektorie gewährleistet ist.

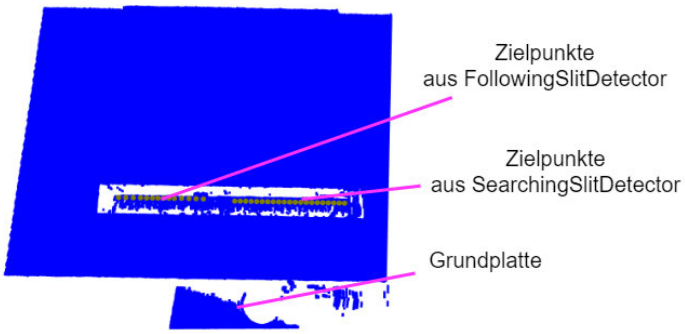
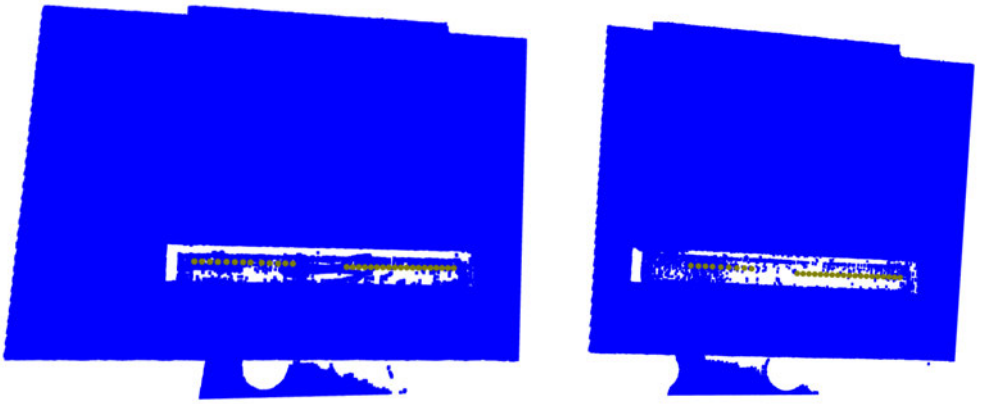


Abbildung 6.4: Die generierten Zielpunkte für Verbindung V1 mit Tiefe von **4 mm**



(a) (b)

Abbildung 6.5: Die generierten Zielpunkte für
a) Verbindung V2 mit Tiefe von **3 mm**
b) Verbindung V3 mit Tiefe von **1 mm**

6.1.3 Test mit fünf Schlitzen mit unterschiedlichen Breiten

Zur Überprüfung der Fähigkeit des Systems, Schlitze mit unterschiedlichen Breiten zu erkennen, wird ein Test mit der **Metallplatte** durchgeführt, die fünf Schlitze unterschiedlicher Breite enthält. Die Testparameter entsprechen denen aus Unterabschnitt 6.1.2, mit der Ausnahme, dass die Bewegungsgeschwindigkeit bei der Online-Verfolgung auf **1 mm/s** reduziert wird, um die Online-Verfolgung sicherzustellen. Die Ergebnisse sind in Tabelle 6.2 zusammengefasst.

Schlitz-Nummer	**Breite des Schlitzes** [mm]		**Länge des Schlitzes** [mm]		**Tiefe des Schlitzes** [mm]		**Anzahl der Zielpunkte**	**Länge der Bewegungsstrecke** [mm]	
	Soll	*Ist*	*Soll*	*Ist*	*Soll*	*Ist*		*Soll*	*Ist*
S1	5	6,10	240	238,64	5	5,15	196	236	236,36
S2	4	5,10	240	237,96	5	5,32	185	236	237,05
S3	3	Unbekannt	240	Unbekannt	5	Unbekannt	Unbekannt	236	Unbekannt
S4	2	Unbekannt	240	Unbekannt	5	Unbekannt	Unbekannt	236	Unbekannt
S5	1	Unbekannt	240	Unbekannt	5	Unbekannt	Unbekannt	236	Unbekannt

Tabelle 6.2: Ergebnisse der Erkennung von Schlitzen mit unterschiedlichen Breiten.

Die Ergebnisse des Tests zur Erkennung von Schlitzen mit unterschiedlichen Breiten (Tabelle 6.2) zeigen folgende Aspekte:

Breite

Die Ergebnisse zeigen, dass das System Schlitze mit einer Breite von 4 mm und größer grundsätzlich funktioniert, allerdings mit Abweichungen. Bei Schlitzen mit einer Breite unter 4 mm (S3 bis S5) funktioniert die Erkennung der Startbreitseite nicht. Die Gründe hierfür lassen sich wie folgt zusammenfassen:

- Voxel-Größe im Down-Sampling: Um die Punktwolke für die Verarbeitung zu reduzieren, wird ein Down-Sampling mit einer festen Voxel-Größe durchgeführt. Dies führt bei schmalen Schlitzen zu einer unzureichenden Punktdichte, da wichtige Details verloren gehen. Eine kleinere Voxel-Größe würde die Erkennung verbessern, ist jedoch mit einem erheblich höheren Rechenaufwand verbunden.
- Auswahl der Nachbarschaft für die Randpunkterkennung: Die Auswahl vom Radius für die Suche der Nachbarschaft zum Erkennen von Randpunkten entlang der Breitseite wird für größere Breiten optimiert. Bei schmaleren Schlitzen (unter 4 mm) ist die Auswahl der Nachbarschaft möglicherweise zu groß, wodurch Randpunkte übersehen oder falsch zugeordnet werden.
- Qualität der Datenerfassung vom Sensor: Besonders im Bereich der Startbreitseite, wo ein Übergang zwischen zwei Ebenen stattfindet, erzeugt der Sensor häufig Rauschen. Dieses Rauschen verschlechtert die Erkennung der Schlitzbreite erheblich und kann zu einer unvollständigen oder fehlerhaften Extraktion der Randpunkte in diesem Bereich führen (siehe Abbildung 6.6).

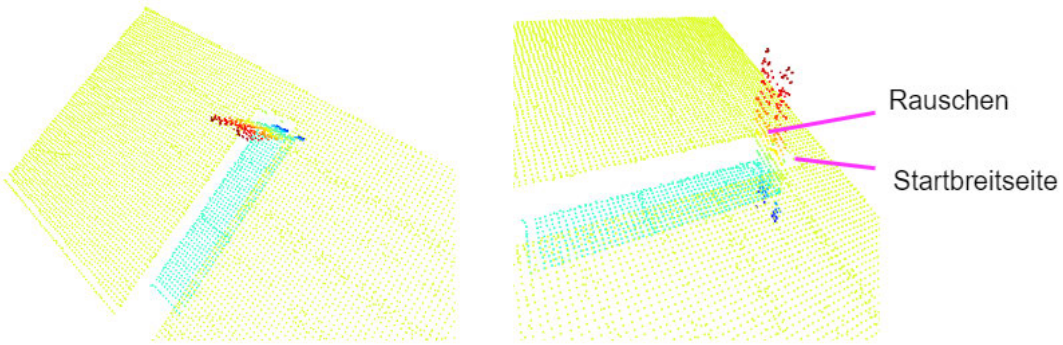


Abbildung 6.6: Rauschen im Bereich der Startbreitseite

Tiefe

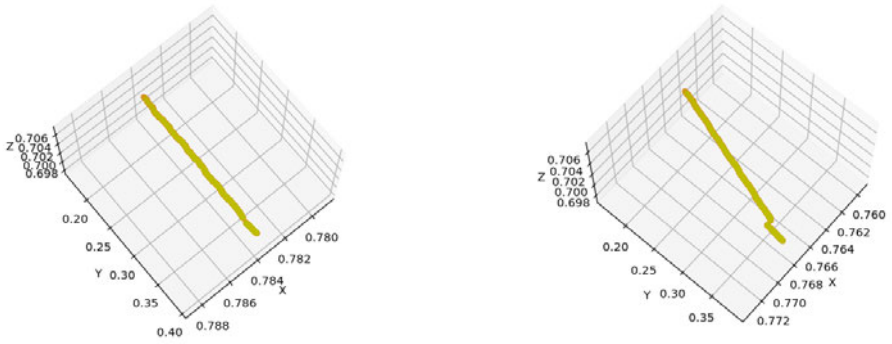
Die Tiefe wird in den meisten Fällen erkannt, allerdings mit Abweichungen. Dies deutet darauf hin, dass das System die Tiefe zuverlässig erfassen kann, unabhängig von der Breite des Schlitzes.

Länge

Die Länge des Schlitzes wird bei den breiteren Schlitzen (S1 und S2) mit geringen Abweichungen erfasst. Dies zeigt, dass die Länge gemessen werden kann, solange der Schlitz erkannt und die Online-Verfolgung durchgeführt werden kann.

Generierung der Zielpunkte

Bei schmalen Schlitzen (S3-S5) erfolgt keine Zielpunktgenerierung und Online-Verfolgung, da Schlitze im `taskApproach` nicht erkannt werden. Für breitere Schlitze (S1 und S2) werden Zielpunkte generiert und sichergestellt, dass sie die Länge des Schlitzes in der Mittellinie abdecken. Die Bewegungsstrecke zeigt eine gute Übereinstimmung mit dem Sollwert. Dies zeigt, dass die Trajektorie des TCP entlang der erkannten medialen Achse präzise umgesetzt wird. In Abbildung 6.7 sind die generierten Zielpunkte für die Schlitze S1 und S2 dargestellt.



(a) (b)

Abbildung 6.7: Die generierten Zielpunkte für
a) Schlitz S1 mit Breite von **5 mm**
b) Schlitz S2 mit Breite von **4 mm**

Aus der Abbildung 6.7 wird deutlich, dass bei Schlitzen mit einer Breite von 5 mm die Zielpunkte eine gerade Linie bilden, was darauf hinweist, dass das System bei Schlitzen dieser Breite und linearem Verlauf zuverlässig arbeitet. Bei Schlitzen mit einer Breite von 4 mm hingegen zeigt sich ein Knick in den generierten Zielpunkten, was darauf hindeutet, dass die vom `SearchingSlitDetector` erzeugten Zielpunkte nicht exakt mit der tatsächlichen Mittellinie des Schlitzes übereinstimmen. Dies könnte auf Ungenauigkeiten bei der Bestimmung der medialen Achse liegen.

6.1.4 Test mit vier unterschiedlichen Geschwindigkeiten

Um die Fähigkeit des Systems zur Online-Erkennung von Schlitzen bei unterschiedlichen Geschwindigkeiten zu testen, wird die **Schlitz-Zapfen-Verbindung V2** untersucht. Die Verbindung V2 hat eine Tiefe von **3 mm**, eine Breite von **6,5 mm** und eine Länge von **50 mm**. Die Scangeschwindigkeit für den initialen Scan ist auf den Standardwert des Systems von **35 mm/s** eingestellt, mit einer Scanstreckenlänge von **50 mm**. Die Bewegungsgeschwindigkeit der Online-Verfolgung des TCPs wird auf **1 mm/s**, **2 mm/s**, **5 mm/s** und **10 mm/s** variiert, um die Leistungsfähigkeit des Systems bei unterschiedlichen Geschwindigkeiten zu bewerten. Die Dauer für die Schlitzerkennung für `SearchingSlitDetector` und `FollowingSlitDetector` berechnet sich nach der

reinen Verarbeitungszeit der Erkennungsmethode. Die Ergebnisse der Dauer der Schlitzerkennung, Zielpunktgenerierung, Bewegungsstrecke und der Kontinuität der Trajektorien werden analysiert.

Dauer der Schlitzerkennung des `SearchingSlitDetector`

Bewegungsgeschwindigkeit beim initialen Scannen $[mm/s]$	`SearchingSlitDetector`		
	Anzahl der Punkte	**Anzahl der Zeilen**	**Dauer** $[s]$
35	86204	119	0,66
35	85458	118	0,60
35	88067	122	0,59
35	86726	120	0,65
Durchschnitt	86.614,25	119,75	**0,625**

Tabelle 6.3: Ergebnisse der Dauer der Erkennung beim initialen Scannen

Die Ergebnisse in Tabelle 6.3 zeigen eine stabile Leistung des `SearchingSlitDetector` beim initialen Scannen in `taskApproach`. Die durchschnittliche Erkennungsdauer beträgt 0,625 Sekunden bei einer durchschnittlichen Anzahl von 119,75 Datenzeilen, was einer durchschnittlichen Punktanzahl von 86.614,25 Punkten entspricht, die der `SearchingSlitDetector` von `SlitFollower` erhalten hat. Es ist jedoch zu beachten, dass Änderungen an Parametern wie der Voxel-Größe, den k-nächsten Nachbarn oder anderen Verarbeitungsparametern diese Dauer beeinflussen können.

Dauer der Schlitzerkennung des `FollowingSlitDetector`

Die Tabelle 6.4 zeigt die Ergebnisse der Schlitzerkennung während der Online-Verfolgung bei unterschiedlichen Bewegungsgeschwindigkeiten. Zusätzlich enthält die Tabelle 6.5 die Ergebnisse des gesamten Prozesses, berechnet vom Start von `taskInitialize` bis zum Ende von `taskExit`.

Bewegungs-geschwindigkeit $[mm/s]$	`FollowingSlitDetector`			
	Anzahl der Durchführungen	**Durchschn. Anzahl der Punkte**	**Durchschn. Anzahl der Zeilen**	**Durchschnitt. Dauer** $[s]$
1	8	25388,34	26,25	1,51
2	10	23640,20	24,40	1,42
5	3	22022,00	21,50	1,76
10	2	17600,00	17,00	1,22

Tabelle 6.4: Ergebnisse der Dauer der Erkennung bei der Online-Verfolgung

Bewegungs-geschwindigkeit $[mm/s]$	**Dauer des gesamten Prozesses** $[s]$	**Anzahl der Zielpunkte**	**Länge der Bewegungsstrecke** $[mm]$	
			Soll	*Ist*
1	51,00	35	46	43,25
2	30,36	36	46	45,64
5	15,06	29	46	32,75
10	11,25	26	46	28,40

Tabelle 6.5: Ergebnisse der Leistung des gesamten Prozesses

Die Ergebnisse der beiden Tabellen zeigen folgende Aspekte:

- Langsame Geschwindigkeiten (1–2 mm/s):
  Die Schlitzerkennung läuft stabil, wodurch eine kontinuierliche Zielpunktgenerierung und Trajektorienplanung gewährleistet wird.

- Mittlere und hohe Geschwindigkeiten (ab 5 mm/s):
  Die Online-Verfolgung läuft nicht mehr stabil und wird frühzeitig abgebrochen. Mögliche Gründe dafür sind:

  – Verzögerung bei der Datenübertragung:
    Die Übertragung von Sensordaten über das Thema `/scan_world` könnte mit der Geschwindigkeit der Bewegung nicht mehr Schritt halten. Dies führt zu einer verspäteten Datenbereitstellung für die Schlitzerkennung.

– Niedrige Verarbeitungsgeschwindigkeit des `FollowingSlitDetector`: Aus Tabelle 6.4 geht hervor, dass die durchschnittliche Verarbeitungszeit der Erkennungsmethode im `FollowingSlitDetector` bei **1,48 Sekunden** liegt, was vergleichsweise hoch ist. Dies deutet darauf hin, dass die Verarbeitungsgeschwindigkeit möglicherweise nicht ausreichend ist, um eine stabile Online-Verfolgung bei hohen Bewegungsgeschwindigkeiten des Schweißprozesses zu gewährleisten.

6.1.5 Test der Offline-Erkennung

Die Offline-Erkennung wird getestet, um die Fähigkeit des Systems zu bewerten, einen vollständigen Scan einer Schlitz-Zapfen-Verbindung zu analysieren und Zielpunkte zu generieren, ohne dass eine Echtzeitverarbeitung erforderlich ist. Dies dient als Vergleichsmaßstab für die Online-Erkennung und bietet eine Alternative für Fälle, in denen Online-Verfolgung nicht möglich oder erforderlich ist.
Die Verbindung V2 (Tiefe: **4 mm**, Breite: **6,5 mm**, Länge: **50 mm**) wird in diesem Test verwendet. Die Scangeschwindigkeit beträgt **35 mm/s** (Systemstandard), bei einer Scanstreckenlänge von **100 mm** und einer Bewegungsgeschwindigkeit von **2 mm/s**.

Abbildung 6.8: Die generierten Zielpunkte der Offline-Erkennung

Abbildung 6.8 zeigt die vom `SearchingSlitDetector` generierten Zielpunkte. Die Anzahl der generierten Punkte beträgt **45**. Aus der Abbildung wird deutlich, dass diese Punkte kontinuierlich und ohne Lücken generiert werden, im Gegensatz zu den Unterbrechungen, die bei der Kombination mit dem `FollowingSlitDetector` auftreten können.

Das für `taskApproach` entwickelte Konzept eignet sich als Lösung für die Offline-Erkennung. Diese Methode kann in Szenarien, in denen keine Online-Verfolgung benötigt wird, effektiv angewendet werden. Allerdings stößt die Lösung bei komplexeren Geometrien an ihre Grenzen und ist in solchen Fällen weniger geeignet.

6.2 Validierung mit der Anforderungsliste

In diesem Abschnitt wird das entwickelte System durch den Vergleich der Testergebnisse mit der in Tabelle 1.1 aufgeführten Anforderungsliste validiert. Dabei wird jede Anforderung daraufhin überprüft, ob sie vollständig **erfüllt**, **teilweise erfüllt** oder **nicht erfüllt** ist. Zusätzlich können einige Anforderungen als **ungetestet** eingestuft werden, falls Zeit- oder Ressourcenmangel die Durchführung entsprechender Tests verhindert hat.

Nr.	Anforderungen	F/W	Erfüllt?
A	**Allgemein**		
A.1	Nutzung der bestehenden Systemstruktur von *ProcessIt*	Forderung	Erfüllt
A.2	Nutzung der bestehenden Softwarearchitektur von *ProcessIt*	Forderung	Erfüllt
A.3	Programmierung in Python	Wunsch: wichtig	Erfüllt
A.4	Beibehaltung der Parameter von *ProcessIt*	Wunsch: wenn möglich	Teilweise erfüllt
A.5	Modulare und erweiterbare Programmierung	Wunsch: wenn möglich	Teilweise erfüllt
P	**Performanz**		
P.1	Offline: Erst Scannen, Erkennen und dann Generierung der Trajektorie	Forderung	Erfüllt
P.2	Online: Paralleles Scannen, Erkennen und Generierung der Trajektorie	Forderung	Erfüllt
P.3	Verarbeitung nur eines Schlitzes pro Ablauf	Wunsch: wichtig	Erfüllt
P.4	Anpassung der Schweiß- und Erkennungsgeschwindigkeit	Wunsch: wenn möglich	Nicht erfüllt

Nr.	Anforderungen	F/W	Erfüllt?
G	**Geometrie des Schlitzes**		
G.1	Schlitzform: Rechteck mit linearem Verlauf	Forderung	Erfüllt
G.2	Erweiterbarkeit für abgerundete Rechtecke oder Spline-Verläufe	Wunsch: nicht wichtig	Ungetestet
G.3	Schlitzbreite: 2 – 10 mm	Wunsch: wichtig	Teilweise erfüllt
G.4	Schlitztiefe: 1 – 20 mm	Wunsch: wichtig	Teilweise erfüllt
G.5	Schlitzlänge: 20 – 100 mm	Wunsch: wichtig	Teilweise erfüllt
G.6	Benutzerdefinierte Nahtlänge als optimierte Variable	Wunsch: wenn möglich	Ungetestet
E	**Erkennen**		
E.1	Erkennung des Schlitzes und der Schlitzkontur	Forderung	Teilweise erfüllt
E.2	Automatische Erkennung des Schlitzanfangs und Schlitzendes	Forderung	Teilweise erfüllt
E.3	Erkennung der Schlitzrichtung	Wunsch: sehr wichtig	Erfüllt
E.4	Erkennung von Schlitzbreite und -länge	Wunsch: sehr wichtig	Teilweise erfüllt
E.5	Erkennung der Schlitztiefe	Wunsch: wenn möglich	Teilweise erfüllt
E.6	Keine signifikante Beeinträchtigung der Schlitzerkennung durch geometrisches Rauschen	Wunsch: sehr wichtig	Ungetestet
E.7	Keine Beeinträchtigung durch Lichtbogen	Wunsch: wenn möglich	Ungetestet
E.8	Funktionsfähigkeit der Online-Erkennung im Bereich relevanter Schweißparameter	Wunsch: wenn möglich	Teilweise erfüllt
T	**Trajektorienplanung**		
T.1	Offline-Generierung der Trajektorie der Schweißnaht	Forderung	Erfüllt

Nr.	Anforderungen	F/W	Erfüllt?
T.2	Online-Generierung der Trajektorie der Schweißnaht	Forderung	Erfüllt
T.3	Die Trajektorie ist in der Mitte der Breite und entlang der Länge des Schlitzes	Forderung	Erfüllt
T.4	Offset am Schlitzanfang und -ende	Wunsch: wenn möglich	Teilweise erfüllt

Tabelle 6.6: Validierung der Anforderungen

7 Zusammenfassung und Ausblick

Es konnte ein Cobot-Sensor-System zur Online-Erkennung und Verschweißung von Schlitz-Zapfen-Verbindungen in rechteckiger Form entwickelt werden. Dieses System automatisiert und vereinfacht den Schweißprozess der Schlitz-Zapfen-Verbindung, reduziert den Bedarf an Roboterprogrammierkenntnissen und erleichtert das Fixieren der Bauteile. Dadurch kann das Vorheften eingespart werden und die Produktionsgeschwindigkeit, insbesondere in High-Mix/Low-Volume-Fertigungen kleiner und mittlerer Unternehmen, gesteigert werden.

Diese Arbeit basiert auf der bestehenden System- und Softwarearchitektur des Projekts *ProcessIt* am **Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA**. Im Rahmen der Arbeit werden Methoden zur Punktwolkenverarbeitung entwickelt und implementiert, darunter die Erkennung von Randpunkten, die Generierung medialer Achsen und die Identifikation der Breitseite des Schlitzes. Diese Methoden werden in Python unter Verwendung der Open3D-Bibliothek realisiert und auf dem Robot Operating System (ROS) aufgebaut. Zwei zentrale Module werden entwickelt: der `SearchingSlitDetector` für die Schlitzerkennung beim initialen Scan und der `FollowingSlitDetector` für die Online-Erkennung während des Schweißprozesses.

Die Testergebnisse zeigen, dass das entwickelte Konzept die Erkennung und Verfolgung von Schlitzen sowie Schlitz-Zapfen-Verbindungen ermöglicht. Tiefe, Breite und Länge verschiedener Schlitze werden erfasst, und Zielpunkte für die Trajektorienplanung werden erfolgreich generiert. Zudem zeigen die Ergebnisse, dass das Konzept sowohl eine Offline-Erkennung für kurze Schlitze als auch eine Online-Erkennung mit dynamischer Anpassung an den Schweißprozess unterstützt.

Allerdings weist die Lösung Einschränkungen auf. Die implementierte Lösung stößt bei der Erkennung von Schlitzen mit schmalen Breiten unter 4 mm oder geringen Tiefen unter 1 mm an ihre Grenzen. Bei TCP-Bewegungsgeschwindigkeiten von mehr als 2 mm/s ist eine zuverlässige Online-Verfolgung nicht gewährleistet, was die Durchführung des

Schweißprozesses erheblich einschränkt. Komplexere Schlitzgeometrien, wie gekrümmte Schlitze, werden nicht berücksichtigt, was die Anwendbarkeit der Lösung in solchen Szenarien einschränkt. Ebenso ist die Lösung bei komplexen Bauteilgeometrien limitiert, z. B. wenn die Hauptebene gekrümmt ist, mehrere Ebenen im Projektionsbereich des Scanners vorhanden sind oder die Ebene mit dem Schlitz nicht die dem Sensor am nächsten gelegene Ebene ist. Rauschen in der Punktwolke und Datenlücken erschweren zusätzlich das Extrahieren von Randpunkten und das Erstellen der medialen Achse. Darüber hinaus wird aus zeitlichen Gründen keine Validierung der Lösung durch tatsächliche Schweißversuche durchgeführt. Parameter wie Schweißwinkel, größere Trajektorie-Offsets am Anfang und Ende des Prozesses sowie andere spezifische Parameter von *ProcessIt* werden nicht untersucht.

Die Ergebnisse dieser Arbeit eröffnen verschiedene Ansätze, die bei der Weiterentwicklung und Optimierung des Systems berücksichtigt werden sollten:

- Behandlung von Rauschen im Bereich der Breitseite:
  Im Bereich der Breitseite, wo der Übergang zwischen zwei Ebenen stattfindet, erzeugt der Sensor häufig Rauschen. Dieses Rauschen führt zu unvollständigen oder fehlerhaften Extraktionen der Randpunkte der Startbreitseite, was wiederum die Schlitzerkennung sowie die Berechnung der Position des ersten Zielpunkts beeinträchtigt.

- Verbesserung der Erkennung schmaler Schlitze:
  Bei Schlitzen mit Breiten von weniger als 4 mm weist das aktuelle Konzept Einschränkungen auf. Mögliche Verbesserungen umfassen die Anpassung der Voxel-Größe beim Down-Sampling, wobei zu beachten ist, dass die Voxel-Größe direkt die Punktanzahl und somit die Verarbeitungsdauer beeinflusst. Für die Extraktion der Randpunkte wäre eine effizientere Auswahl der Nachbarschaft erforderlich, die flexibel an die Breite des Schlitzes angepasst werden kann [10]. Zudem werden Schlitze mit sehr großen Breiten bisher nicht getestet.

- Erhöhung der Geschwindigkeit der Online-Verfolgung:
  Um die Bewegungsgeschwindigkeit des TCPs zu erhöhen, muss die Verarbeitungszeit der Erkennungsmethode im `FollowingSlitDetector` reduziert und optimiert werden. Zudem sollte die Verzögerung bei der Datenübertragung vom Sensor analysiert und minimiert werden, um die Online-Verfolgung zu gewährleisten.

- Optimierung der Verarbeitungsparameter: Die Verarbeitungsmethoden zur Schlitzerkennung, einschließlich der Parameter wie Voxel-Größe, Schwellenwerte und Toleranzen, sollten präziser berechnet und gezielt optimiert werden, um die Genauigkeit und Effizienz der Erkennung zu verbessern.

- Erweiterung für Schlitze mit gekrümmtem Verlauf:
  Aus Zeitgründen fokussiert sich diese Arbeit ausschließlich auf die Entwicklung einer Lösung für Schlitze mit linearem Verlauf entlang ihrer Länge. Für Schlitze mit gekrümmtem Verlauf ist eine weiterführende Untersuchung und Anpassung des Systems erforderlich.

Literaturverzeichnis

[1] Ahmed, Syeda M. ; Tan, Yan Z. ; Chew, Chee M. ; Mamun, Abdullah A. ; Wong, Fook S.: Edge and Corner Detection for Unorganized 3D Point Clouds with Application to Robotic Welding. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid : IEEE, Oktober 2018, S. 7350–7355. – ISBN 978-1-5386-8094-0

[2] Bazazian, Dena ; Casas, Josep R. ; Ruiz-Hidalgo, Javier: Fast and Robust Edge Extraction in Unorganized Point Clouds. In: *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. Adelaide, Australia : IEEE, November 2015, S. 1–8. – ISBN 978-1-4673-6795-0

[3] Blum, H.: *A Transformation for Extracting New Descriptors of Shape*. M.I.T. Press, 1967

[4] Chen, Dong ; Wang, Ruisheng ; Peethambaran, Jiju: Topologically Aware Building Rooftop Reconstruction From Airborne Laser Scanning Point Clouds. In: *IEEE Transactions on Geoscience and Remote Sensing* 55 (2017), Dezember, Nr. 12, S. 7032–7052. – ISSN 0196-2892, 1558-0644

[5] Chen, Dong ; Zhang, Liqiang ; Mathiopoulos, P. T. ; Huang, Xianfeng: A Methodology for Automated Segmentation and Reconstruction of Urban 3-D Buildings from ALS Point Clouds. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7 (2014), Oktober, Nr. 10, S. 4199–4217. – ISSN 1939-1404, 2151-1535

[6] Chen, Xijiang ; Yu, Kegen: Feature Line Generation and Regularization From Point Clouds. In: *IEEE Transactions on Geoscience and Remote Sensing* 57 (2019), Dezember, Nr. 12, S. 9779–9790. – ISSN 0196-2892, 1558-0644

[7] Choi, Hyeong I. ; Han, Chang Y.: The Medial Axis Transform. In: *Handbook of Computer Aided Geometric Design*. Elsevier, 2002, S. 451–471. – ISBN 978-0-444-51104-1

[8] Automationspraxis: *Cobot und kollaborative Roboter: Wissenswertes zur MRK*. – URL https://automationspraxis.industrie.de/cobot/cobot-symbiose-von-mensch-und-roboter/. – Zugriffsdatum: 2024-11-12

[9] Demarsin, Kris ; Vanderstraeten, Denis ; Volodine, Tim ; Roose, Dirk: Detection of Closed Sharp Edges in Point Clouds Using Normal Estimation and Graph Theory. In: *Computer-Aided Design* 39 (2007), April, Nr. 4, S. 276–283. – ISSN 0010-4485

[10] Dey, Emon K. ; Tarsha Kurdi, Fayez ; Awrangjeb, Mohammad ; Stantic, Bela: Effective Selection of Variable Point Neighbourhood for Feature Point Extraction from Aerial Building Point Cloud Data. In: *Remote Sensing* 13 (2021), April, Nr. 8, S. 1520. – ISSN 2072-4292

[11] Edelsbrunner, H. ; Kirkpatrick, D. ; Seidel, R.: On the Shape of a Set of Points in the Plane. In: *IEEE Transactions on Information Theory* 29 (1983), Juli, Nr. 4, S. 551–559. – ISSN 0018-9448

[12] Edelsbrunner, Herbert ; Mücke, Ernst P.: Three-Dimensional Alpha Shapes. In: *Proceedings of the 1992 Workshop on Volume Visualization - VVS '92*. Boston, Massachusetts, United States : ACM Press, 1992, S. 75–82. – ISBN 978-0-89791-527-4

[13] Ester, Martin ; Kriegel, Hans-Peter ; Sander, Jörg ; Xu, Xiaowei: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.

[14] Fischer, Kaspar: Introduction to Alpha Shapes.

[15] Fischler, Martin A. ; Bolles, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Commun. ACM* 24 (1981), Juni, Nr. 6, S. 381–395. – URL https://doi.org/10.1145/358669.358692. – ISSN 0001-0782

[16] Fraunhofer-IPA: *Schweißen mit Cobots - Fraunhofer IPA*. – URL https://www.ipa.fraunhofer.de/de/Kompetenzen/roboter--und-assistenzsysteme/schweissen-und-bearbeiten.html. – Zugriffsdatum: 2024-11-12

[17] Fraunhofer-IPA: *Institutsprofil - Fraunhofer IPA*. – URL https://www.ipa.fraunhofer.de/de/ueber_uns/institutsprofil.html. – Zugriffsdatum: 2024-11-12

[18] Fraunhofer-IPA: *Roboter- und Assistenzsysteme.* – URL https://www.ipa.fraunhofer.de/de/Kompetenzen/roboter--und-assistenzsysteme.html. – Zugriffsdatum: 2024-11-12

[19] Frischgesell, Thomas: *Vorlesungsskript: Robotertechnik.* 2023. – Lecture notes

[20] Gao, Jie ; Li, Fang ; Zhang, Cong ; He, Wenhao ; He, Jinhu ; Chen, Xubing: A Method of D-Type Weld Seam Extraction Based on Point Clouds. In: *IEEE Access* 9 (2021), S. 65401–65410. – ISSN 2169-3536

[21] Gumhold, Stefan ; Wang, Xinlong ; MacLeod, Rob: Feature Extraction from Point Clouds. (2001), November

[22] Guo, ZhuoRan ; Lu, Jun ; Han, Jing ; Zhao, Zhuang: Weld Bead Extraction Based on Point Cloud Segmentation. In: Cen, Fengxin (Hrsg.) ; Ke, Xizheng (Hrsg.): *International Conference on Optics and Image Processing (ICOIP 2021).* Guilin, China : SPIE, September 2021, S. 29. – ISBN 978-1-5106-4695-7 978-1-5106-4696-4

[23] haiza sfera: *Schweißverbindungen: Fugformen und Schweißnahtarten.* 2024. – URL https://sfera-haiza.de/schweissen/schweissverbindungen.html. – Accessed: 2024-12-30

[24] Kapfer, Johannes: *Ein modellfreier und ein modellprädiktiver Ansatz zur Online-Bahnverfolgung eines Schweiß-Cobots.* Karlsruhe, Karlsruher Institut für Technologie, Dissertation, Mai 2023

[25] Kusumoto, Ryosuke ; Takubo, Tomohito ; Tsujioka, Tetsuo: Welding Seam Detection between Cylinder and Plane Using Point Cloud Data. In: *2022 International Symposium on Micro-NanoMehatronics and Human Science (MHS).* Nagoya, Japan : IEEE, November 2022, S. 1–5. – ISBN 978-1-66546-278-5

[26] Ma, Jaehwan ; Bae, Sang W. ; Choi, Sunghee: 3D Medial Axis Point Approximation Using Nearest Neighbors and the Normal Field. In: *The Visual Computer* 28 (2012), Januar, Nr. 1, S. 7–19. – ISSN 0178-2789, 1432-2315

[27] MICRO-EPSILON MESSTECHNIK GmbH & Co. KG: *scanCONTROL 30xx Betriebsanleitung.* Königbacher Str. 15, 94496 Ortenburg, Germany: MICRO-EPSILON MESSTECHNIK GmbH & Co. KG (Veranst.), 2016. – URL https://www.micro-epsilon.de. – User Manual for the scanCONTROL 30xx Laser Scanner Series

[28] MoveIt: *MoveIt Motion Planning Framework*. – URL https://moveit.ai/. – Zugriffsdatum: 2024-11-16

[29] Muja, Marius ; Lowe, David G.: *FLANN - Fast Library for Approximate Nearest Neighbors*. Online: , 2013. – URL https://www.cs.ubc.ca/research/flann/. – Accessed: 2024-12-30

[30] Nguyen, Van S. ; Trinh, Trong H. ; Tran, Manh H.: Hole Boundary Detection of a Surface of 3D Point Clouds. In: *2015 International Conference on Advanced Computing and Applications (ACOMP)*. Ho Chi Minh City, Vietnam : IEEE, November 2015, S. 124–129. – ISBN 978-1-4673-8234-2

[31] Ni, Huan ; Lin, Xiangguo ; Ning, Xiaogang ; Zhang, Jixian: Edge Detection and Feature Line Tracing in 3D-Point Clouds by Analyzing Geometric Properties of Neighborhoods. (2016)

[32] Nowak, Marta ; Michoński, Jakub ; Sitnik, Robert: Filling Cavities in Point Clouds Representing Human Body Surface Using Bezier Patches. In: *Multimedia Tools and Applications* 80 (2021), April, Nr. 10, S. 15093–15134. – ISSN 1380-7501, 1573-7721

[33] Paschke, Udo ; Landgraf, Christian ; Ernst, Kilian ; Stoll, Johannes T. ; Kraus, Werner: An Easy Hand-Eye Calibration Method for Laser Profile Scanners in High Precision Applications Using Optimized View Poses. In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. Mexico City, Mexico : IEEE, August 2022, S. 348–355. – ISBN 978-1-66549-042-9

[34] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: *Nearest Neighbors*. Online: scikit-learn developers (Veranst.), 2023. – URL https://scikit-learn.org/1.5/modules/neighbors.html. – Accessed: 2024-12-27

[35] Peng, Rui ; Navarro-Alarcon, David ; Wu, Victor ; Yang, Wen: *A Point Cloud-Based Method for Automatic Groove Detection and Trajectory Generation of Robotic Arc Welding Tasks*. April 2020

[36] MoveIt: *Planners | MoveIt*. – URL https://moveit.ai/documentation/planners/. – Zugriffsdatum: 2024-11-16

[37] Qu, Guangjun ; Zhang, Qing ; Liu, Guangyuan ; Zhang, Aijun: Improvement of Voxel Down-Sampling Method in Point Cloud Registration. In: *2023 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*. Chengdu, China : IEEE, November 2023, S. 919–923. – ISBN 9798350331417

[38] Quigley, Morgan ; Conley, Ken ; Gerkey, Brian P. ; Faust, Josh ; Foote, Tully ; Leibs, Jeremy ; Wheeler, Rob ; Ng, Andrew Y.: ROS: an Open-Source Robot Operating System. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* IEEE Robotics and Automation Society (Veranst.), IEEE, 2009. – URL https://api.semanticscholar.org/CorpusID:6324125. – Available online

[39] Rodriguez-Gonzalvez, Pablo ; Rodriguez-Martin, Manuel: Weld Bead Detection Based on 3D Geometric Features and Machine Learning Approaches. In: *IEEE Access* 7 (2019), S. 14714–14727. – ISSN 2169-3536

[40] ROS: *ROS: Home*. – URL https://www.ros.org/. – Zugriffsdatum: 2024-11-16

[41] Rusu, Radu B. ; Cousins, Steve: 3D is here: Point Cloud Library (PCL). In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 9-13 2011

[42] Schmitz, Thilo: *Cobot vs. Roboter – Automatisierung beim Schweißen*. 2023. – URL https://blog.binzel-abicor.com/de/cobot-vs-roboter-automatisierung-beim-schweissen. – Accessed: 2024-12-30

[43] Schuth, Michael ; Buerakov, Wassili: *Handbuch Optische Messtechnik: Praktische Anwendungen für Entwicklung, Versuch, Fertigung und Qualitätssicherung*. München : Carl Hanser Verlag GmbH & Co. KG, September 2017. – ISBN 978-3-446-43634-3 978-3-446-43661-9

[44] Sugihara, Kokichi: Voronoi Diagrams. In: *Handbook of Computer Aided Geometric Design*. Elsevier, 2002, S. 429–450. – ISBN 978-0-444-51104-1

[45] Takubo, Tomohito ; Miyake, Erika ; Ueno, Atsushi ; Kubo, Masaki: Welding Line Detection Using Point Clouds from Optimal Shooting Position. In: *Journal of Robotics and Mechatronics* 35 (2023), April, Nr. 2, S. 492–500. – ISSN 1883-8049, 0915-3942

[46] Tian, Pengju ; Hua, Xianghong ; Tao, Wuyong ; Zhang, Miao: Robust Extraction of 3D Line Segment Features from Unorganized Building Point Clouds. In: *Remote Sensing* 14 (2022), Juli, Nr. 14, S. 3279. – ISSN 2072-4292

[47] Tomek, Lukáš ; Beran, Branislav ; Erdélyi, Ján ; Honti, Richard ; Mikula, Karol: Multichannel Segmentation of Planar Point Clouds Using Evolving Curves. In: *Computational and Applied Mathematics* 42 (2023), Dezember, Nr. 8, S. 332. – ISSN 2238-3603, 1807-0302

[48] Tran, Trung-Thien ; Laurendeau, Denis ; Ali, Sarah: Automatic Sharp Feature Extraction from Point Clouds with Optimal Neighborhood Size.

[49] Wang, XiaoHui ; Chen, HuaWei ; Wu, LuShen: Feature Extraction of Point Clouds Based on Region Clustering Segmentation. In: *Multimedia Tools and Applications* 79 (2020), Mai, Nr. 17, S. 11861–11889. – ISSN 1573-7721

[50] Weber, Christopher ; Hahmann, Stefanie ; Hagen, Hans: Methods for Feature Detection in Point Clouds. In: *OASIcs, Volume 19, VLUDS 2010* 19 (2012), S. 90–99. – ISBN 9783939897293

[51] Widyaningrum, Elyta ; Peters, Ravi Y. ; Lindenbergh, Roderik C.: Building Outline Extraction from ALS Point Clouds Using Medial Axis Transform Descriptors. In: *Pattern Recognition* 106 (2020), Oktober, S. 107447. – ISSN 00313203

[52] Wipper, Joachim: Mediale Achsen und Voronoj-Diagramme in der euklidischen Ebene, Universität Stuttgart, 1997

[53] Xi, Xiaohuan ; Wan, Yiping ; Wang, Cheng: Building Boundaries Extraction from Points Cloud Using an Image Edge Detection Method. In: *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. Beijing, China : IEEE, Juli 2016, S. 1270–1273. – ISBN 978-1-5090-3332-4

[54] Yang, Lei ; Liu, Yanhong: A Novel 3D Seam Extraction Method Based on Multi-Functional Sensor for V-Type Weld Seam. In: *IEEE Access* 7 (2019), S. 182415–182424. – ISSN 2169-3536

[55] Yang, Lei ; Liu, Yanhong ; Peng, Jinzhu ; Liang, Zize: A Novel System for Off-Line 3D Seam Extraction and Path Planning Based on Point Cloud Segmentation for Arc Welding Robot. In: *Robotics and Computer-Integrated Manufacturing* 64 (2020), August, S. 101929. – ISSN 07365845

[56] Zhang, Lunzhao ; Xu, Yanling ; Du, Shaofeng ; Zhao, Wenjun ; Hou, Zhen ; Chen, Shanben: Point Cloud Based Three-Dimensional Reconstruction and Identification of Initial Welding Position. In: Chen, Shanben (Hrsg.) ; Zhang, Yuming (Hrsg.) ; Feng, Zhili (Hrsg.): *Transactions on Intelligent Welding Manufacturing*. Singapore : Springer Singapore, 2018, S. 61–77. – ISBN 978-981-10-8329-7 978-981-10-8330-3

[57] Zhang, Yuankai ; Jiang, Yong ; Tian, Xincheng ; Xu, Xiaolong ; Geng, Yusen ; Lai, Min: *A Point Cloud-Based Welding Trajectory Planning Method for Plane Welds*. August 2022

[58] Zhou, Bo ; Liu, Yirong ; Xiao, Yao ; Zhou, Rui ; Gan, Yahui ; Fang, Fang: Intelligent Guidance Programming of Welding Robot for 3D Curved Welding Seam. In: *IEEE Access* 9 (2021), S. 42345–42357. – ISSN 2169-3536

[59] Zhou, Peng ; Peng, Rui ; Xu, Maggie ; Wu, Victor ; Navarro-Alarcon, David: Path Planning With Automatic Seam Extraction Over Point Cloud Models for Robotic Arc Welding. In: *IEEE Robotics and Automation Letters* 6 (2021), Juli, Nr. 3, S. 5002–5009. – ISSN 2377-3766, 2377-3774

[60] Zhou, Qian-Yi ; Park, Jaesik ; Koltun, Vladlen: Open3D: A Modern Library for 3D Data Processing. In: *arXiv:1801.09847* (2018)

A Anhang

Erklärung zur selbständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Entwicklung eines Cobot-Sensor-Systems zur Online-Erkennung und Verschweißung von Schlitz-Zapfen-Verbindungen

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der Bachelorarbeit ist erfolgt durch:

Ort	Datum	Unterschrift im Original