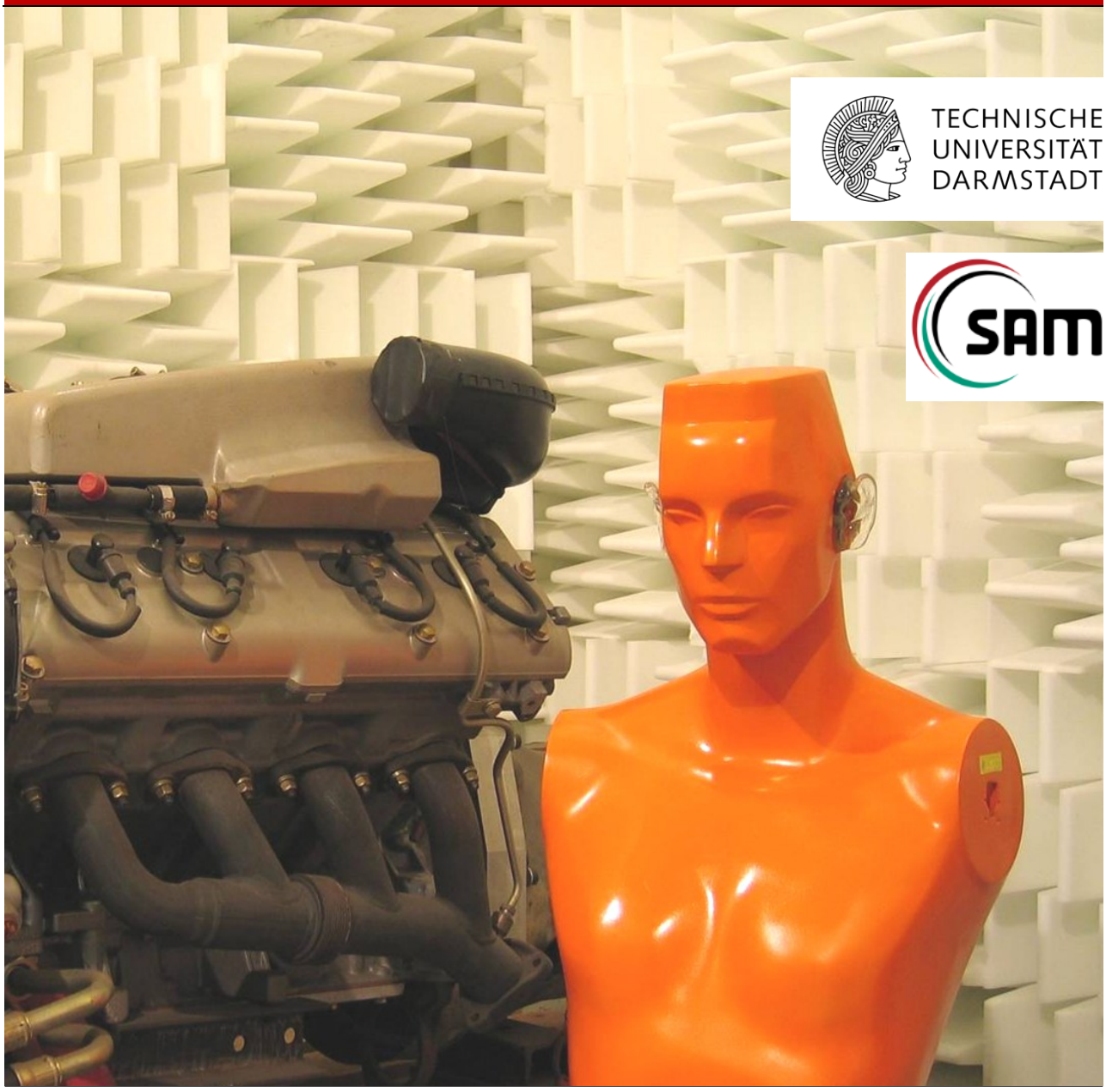
System zur automatisierten Messung der Kantenverrundung von Blechbauteilen mit Mehrachs-Kinematik und Laserscanner

System for automated measurement of edge rounding of sheet metal parts with multi-axis kinematics and laser scanner

Louis Charton | Bachelor-Thesis | 2023
Fachgebiet Systemzuverlässigkeit, Adaptronik und Maschinenakustik SAM



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Louis Charton | 2755515

Maschinenbau

Bachelor-Thesis
System zur automatisierten Messung der Kantenverrundung von Blechbauteilen mit Mehrachs-Kinematik und Laserscanner

System for automated measurement of edge rounding of sheet metal parts with multi-axis kinematics and laser scanner

Eingereicht am: 3. Juli 2023

Betreuer: Hendrik Schmidt, Valentin Mees

Prof. Dr.-Ing. Tobias Melz
Fachgebiet Systemzuverlässigkeit, Adaptronik und Maschinenakustik SAM
Fachbereich Maschinenbau
Technische Universität Darmstadt
Magdalenenstr. 4
64289 Darmstadt

Erklärung zur vorliegenden Arbeit gemäß § 22/7 und § 23/7 APB der TU Darmstadt

Hiermit versichere ich, die vorliegende Arbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Sämtliche aus fremden Quellen indirekt oder direkt übernommenen Gedanken sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und wurde noch nicht veröffentlicht. In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, 29.6.23
Ort, Datum

Name, Vorname

Thesis Statement pursuant to § 22/7 and § 23/7 of APB TU Darmstadt

I hereby declare that I have written the submitted thesis independently without any outside support except for the quoted literature and other sources mentioned in the thesis. All the thoughts acquired and employed, directly or indirectly, from any foreign source are clearly identified and listed in the thesis. This work has neither been handed in nor published in the same or similar form before. In the submitted thesis the written and electronic version are both identical.

Darmstadt, 29.6.23
Ort, Datum

Name, Vorname

Kurzfassung

Das Verrunden von Werkstückkanten ist ein wichtiger Fertigungsschritt zur Einhaltung von Normen und individuellen Spezifikationen. Scharfe Kanten und Grate entstehen in der Regel durch Trennverfahren. Derzeit werden die Kantenradien bei komplexen Bauteilgeometrien jedoch manuell vermessen, eine Automatisierung ist in der Regel nicht vorhanden.

Im Rahmen dieser Arbeit wird ein Demonstrator-System entwickelt, das die Kantenverrundung von Blechbauteilen mit einem Linienprofil-Laserscanner misst. Der Scanner ist auf einer Vierachs-Kinematik mit SPS montiert, welche Punkte in einem definierten Messbereich anfahren kann. Diese Punkte werden zuvor anhand einer CAD-Referenzdatei festgelegt. Die Erfassung des in einem Messbereich frei positionierten Blechbauteils erfolgt durch eine einzelne Kamera. Aus dem Kamerabild wird die Position des Blechbauteils ermittelt und die Bahnkoordinaten für die Kinematik/SPS berechnet. Zusätzlich wird der Demonstrator konstruiert und so gestaltet, dass er im Anschluss an die Thesis aufgebaut und genutzt werden kann.

Dazu wird ein Python-Programm entwickelt, welches die Kommunikation mit einem Laserscanner, einer SPS und einer Kamera über Ethernet implementiert. Der Schwerpunkt der praktischen Entwicklung liegt softwareseitig auf der Bildverarbeitung. In diesem Rahmen wird ein neuer Ansatz entwickelt, um perspektivisch bedingte Verzerrungen aus den Kamerabildern zu entfernen, um anschließend mittels Machine Vision Koordinaten für Messpunkte zu ermitteln. Als Ausgangspunkt dient die Open-Source-Bibliothek OpenCV. Die festgelegten Messpunkte werden über das Siemens S7 Protokoll an eine Siemens SPS exportiert.

Abschließende Tests zeigen, dass die durch das Programm berechneten und von der SPS angefahrenen Messpunktkoordinaten bei ausreichender algorithmischer Kalibrierung der Kamera um weniger als 0,5 mm von den tatsächlichen Koordinaten abweichen können.

Abstract

Rounding of workpiece edges is an important manufacturing step to meet standards and individual specifications. Sharp edges or even burrs are usually produced by cutting processes. However, currently the edge radii of complex component geometries are measured manually, and automation is not usually available.

This thesis develops a demonstrator system that measures the edge rounding of sheet metal components using a line profile laser scanner. The Scanner is mounted on a four-axis kinematic system with a PLC, which can be used to move to points in a free measuring area. These points are defined in advance using a CAD reference file. The sheet metal part, freely positioned in a measurement area, is captured by a single camera. The camera image is used to determine the position of the sheet metal part and to calculate the path coordinates for the kinematics/PLC. In addition, the demonstrator will be constructed and designed so that it can be set up and used in perspective.

A Python program will be written to communicate with a laser scanner, PLC, and camera via Ethernet. On the software side, the practical development focuses on image processing. Within this framework, a novel approach will be developed to remove perspective distortions from the camera images to subsequently determine the coordinates for measuring points using machine vision. The open-source library OpenCV is used as a starting point. These points are exported to a Siemens PLC using the Siemens S7 protocol (snap7).

Final Tests show that coordinates of the measuring points calculated by the program and approached by the PLC can deviate less than 0.5 mm from the actual coordinates if the algorithmic calibration of the camera is sufficient.

Inhaltsverzeichnis

Erklärung zur vorliegenden Arbeit gemäß § 22/7 und § 23/7 APB der TU Darmstadt III

Kurzfassung IV

Inhaltsverzeichnis VI

1 Einleitung 1

1.1 Motivation und Ziele der Arbeit ... 1
1.2 Methodisches Vorgehen / Problemstellungen & Lösungsansätze / Gliederung ... 2

2 Grundlagen und Stand der Technik 3

2.1 Definition Blech ... 3
2.2 Entgraten als industrieller Fertigungsschritt ... 3
2.3 Grundlagen für die Entwicklung und Konstruktion ... 5
2.3.1 Kinematik & Steuerung ... 5
2.3.2 Machine Vision & Kamera ... 8
2.4 Stand der Technik ... 11
2.4.1 Relevante wissenschaftliche Arbeiten ... 11
2.4.2 Durchführung in der Praxis/ Normative Anforderungen ... 13
2.4.3 Folgerungen für die vorliegende Arbeit ... 14

3 Anforderungen an das Gesamtsystem und theoretische Entwicklung 16

3.1 Methodik der Entwicklung und Randbedingungen ... 16
3.2 Randbedingungen der Entwicklung ... 18
3.3 Funktionale Anforderungen an das Gesamtsystem ... 18
3.4 Programmablaufplan ... 20
3.5 Auslegung und Auswahl der Komponenten ... 21
3.5.1 Laser-Scanner ... 21
3.5.2 Kinematik ... 23
3.5.3 Steuerung der Kinematik (SPS) ... 27
3.5.4 Bildgebung / Objekterkennung ... 28
3.6 Zusammenfassung der theoretischen Entwicklung ... 33
3.6.1 Liste verwendeter Komponenten ... 33
3.6.2 CAD-Modell des Prototyps ... 34

4 Implementierung der Subsysteme / Integration des Gesamtsystems 36

4.1 Rahmenbedingungen der Entwicklung & Software-Struktur ... 37
4.2 Software-Programmablauf / Funktionsweise der Subsysteme ... 39
4.2.1 Umgang mit Meta-Daten und Referenz-Daten ... 41
4.2.2 Bildgebung ... 41
4.2.3 Bildbearbeitung und Objekterkennung ... 45
4.2.4 Berechnung von Punktkoordinaten ... 57
4.2.5 Steuerung des Portalroboters ... 63
4.2.6 Laser-Scanner ... 67
4.2.7 Erstellen und Verarbeitung von Punktmessungen ... 68
4.2.8 Konstruktive Vorkehrungen ... 71
4.3 Zusammenfassung der praktischen Implementierung ... 73

4.4 Demonstration anhand eines Beispiels 75

5 Ergebnisse der Entwicklung, Perspektive & Ausblick **78**

5.1 Zusammenfassung der Entwicklung, Bedienung des Demonstrators 78
5.2 Möglichkeiten zur Weiterentwicklung, industrieller Nutzen 79

A Inhaltsverzeichnis des digitalen Anhangs **VIII**

Abbildungsverzeichnis **IX**

Tabellenverzeichnis **XI**

Abkürzungsverzeichnis **XII**

Literaturverzeichnis **XIII**

1 Einleitung

Diese Arbeit befasst sich mit der Entwicklung eines Systems und der Konstruktion eines entsprechenden Demonstrators mit der Aufgabe bei industriell gefertigten Blechbauteilen automatisch eine repräsentative Messung der Kantenradien vorzunehmen. Im Fokus steht hierbei der Entwicklungs- und Konstruktionsprozess des Demonstrators, sowie die Einordnung der Technologie im Kontext der gegenwärtigen Methoden der Qualitätssicherung in der blechverarbeitenden Industrie.

1.1 Motivation und Ziele der Arbeit

Die Idee für das Thema dieser Arbeit entstand im Rahmen einer Tätigkeit am Fraunhofer LBF. Über das Förderprogramn ZIM (Zentrales Innovationsprogramm Mittelstand) wurde die Zusammenarbeit mit dem Werkzeughersteller Boeck GmbH ermöglicht. In diesem Rahmen wurde der Bedarf für automatische Messverfahren in der Qualitätssicherung im Bereich der Blech-Entgratung geäußert.

Das Entgraten ist ein Fertigungsschritt, bei welchem Grate (siehe Abbildung 1.1) von einem Blechbauteil entfernt werden und gleichzeitig die Bauteilkanten verrundet werden können. Die Gründe hierfür können von funktioneller-, ergonomischer-, oder ästhetischer Natur sein [1]. Für nachfolgende Fertigungsschritte, sowie im Bereich der Arbeitssicherheit und in der Prozessoptimierung ist ein erwartungsgemäß entgratetes Blechbauteil von großer Wichtigkeit.

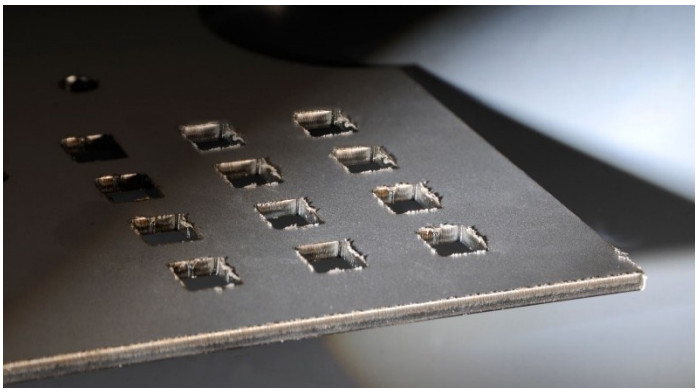


Abbildung 1.1: Ausgeprägter Grat bei Löchern in einem Dünnblech [2]

Zur Feststellung und Bewertung der Qualität für die Einhaltung von Richtlinien und Pflichtenheften kann nach der Entgratung ein Messschritt vorgesehen werden. In diesem wird die Entfernung aller Grate, sowie eine korrekte Kantenbearbeitung überprüft und bewertet. Wenn dieses Verfahren unpräzise oder ineffizient ausgeführt wird, ist die Qualität der gefertigten Blechbauteile nicht sichergestellt. Für eine effiziente und robuste Fertigungsstraße gilt es also, ein solches Mess- und Bewertungsverfahren so schnell und präzise wie möglich zu gestalten. Dies kann durch ein präzise steuerbares, automatisches Messwerkzeug sichergestellt werden.

Das Ziel der Arbeit ist die Entwicklung und die Konstruktion eines Demonstrator-Systems, welches in der Lage ist, bei Blechbauteilen an festgelegten Punkten automatisch die Qualität der Kantenverrundung zu überprüfen. Ferner soll das System in der Lage sein, Blechbauteile bekannter Geometrien auf einer Messfläche zu erkennen, und ohne vorgegebene Positionierung des Blechbauteils eine Messung vorzunehmen. Somit kann neben dem Entgrat-Prozess auch der anschließende Messprozess automatisiert werden. Perspektivisch soll sich diese Technologie in eine vollautomatisierte Fertigungsstraße einfügen lassen.

1.2 Methodisches Vorgehen / Problemstellungen & Lösungsansätze / Gliederung

Auf Basis der Analyse des Standes der Technik im Bereich der industriellen Qualitätssicherung bei Blechbauteilen werden Anforderungen an das Gesamtsystem formuliert. Daran anschließend werden Komponenten zur Konstruktion des Demonstrators ausgewählt. Die Teilsysteme der Entwicklung liefern den Rahmen der theoretischen Grundlagen. Aspekte der Entwicklung mit Bedarf einer problemspezifischen Lösung sind für diese Arbeit:

- Die Nutzung von maschinellem Sehen (im Folgenden engl. Machine Vision) zur Erfassung von freiliegenden Blechbauteilen auf einer bekannten Messfläche mit gegebenen Abmessungen auf Basis der Bildgebung durch eine Kamera, sowie der Entwicklung eines Algorithmus zur Entzerrung und Aufbereitung der Bilddaten
- Der Vergleich dieser Daten mit Referenzdaten der Blechbauteile in Form von CAD-Daten, welche Informationen über vorgegebene Messpunkte auf deren Kanten enthalten
- Die Steuerung & Regelung einer Mehrachs-Kinematik zum zeitlich optimalen Anfahren dieser Messpunkte auf den Blechbauteilen
- Die Implementierung eines Laser-Scanners als Messwerkzeug zum Vermessen der Kanten mit geeignetem Messalgorithmus
- Das Schaffen der korrekten Rahmenbedingungen für Kinematik, Machine Vision & Laser-Scanner in Form der prototypischen konstruktiven Gestaltung des Demonstrators
- Die Gesamtintegration dieser Einzelsysteme durch Software in Python, sowie die Bereitstellung einer Schnittstelle für einen Computer zur Datenerfassung- und Auswertung.

Auf Basis der definierten Liste an Anforderungen werden lösungsorientierte Technologien gesucht und anschließend, unter Bedingung der Kompatibilität, spezifische Komponenten ausgewählt. Die notwendigen Technologien werden zunächst isoliert betrachtet und ausgelegt und im letzten Schritt der Systemauslegung zu einem Gesamtsystem zusammengeführt. Abschließend werden die Ergebnisse demonstriert, die Funktionalität beurteilt, die Entwicklung in den aktuellen Innovationsstand eingeordnet, sowie die Perspektive der Technologie analysiert.

2 Grundlagen und Stand der Technik

Im folgenden Kapitel wird der Stand der Technik bei Methoden zur Qualitätssicherung in der blechverarbeitenden Industrie aufgeführt. Zu Beginn werden zunächst die Begriffe des Blechbauteils, sowie des Entgratens in der Industrie definiert. So erfolgt eine Eingrenzung des Nutzungsrahmens dieser Technologie. Mit Hinblick auf die theoretische und praktische Konstruktion des Demonstrators werden anschließend die notwendigen theoretischen Grundlagen, für die im Rahmen dieser Arbeit verwendeten Technologien bezüglich Hard- und Software erörtert. Mit Hinblick zur Entwicklungsphase in Kapitel 3 und Kapitel 4 wird der Stand der Technik im Bereich der Qualitätssicherung in der blechverarbeitenden Industrie analysiert und auf relevante wissenschaftliche Arbeiten eingegangen.

2.1 Definition Blech

Nach DIN EN 10079 ist ein Blech ein:

> *„[warm oder kalt]gewalztes Flacherzeugnis mit nicht festgelegter Verformung der Kanten, das in Tafeln meist quadratischer oder rechteckiger Form, aber auch mit jeder anderen Form, z.B. rund oder nach Zeichnung […] geliefert wird“* [3].

Für Bleche ist dementsprechend die Güte der Ober- und Unterseite relevant für eine Messung, wobei die Seiten nicht beachtet werden müssen. Bauteile, welche aus Blech hergestellt werden, sind dementsprechend *Blecherzeugnisse*, werden aber im Folgenden als Blechteil, oder Blechbauteil bezeichnet. Zwar können Blechteile ohne Verlust der Definition eines Bleches auch vorgebogen vorliegen [3], dieser Fall wird im Rahmen dieser Arbeit allerdings nicht betrachtet. Im Rahmen dieser Arbeit werden demnach Blechteile betrachtet, welche eine flache, zweidimensionale Fläche als Ober- und Unterseite aufweisen.

Relevant für diese Arbeit sind auch bereits teilverarbeitete Blechbauteile. Die verwendeten Blechteile sollen so vorliegen, dass sie allein durch eine Oberflächengeometrie und einer über der Bauteilfläche konstanten Dicke beschreibbar sind. Diese Definition ist nahezu deckungsgleich mit der Definition der DIN zuzüglich der Forderung von Flachheit. Fertigungs- oder verarbeitungsbedingt entstehen an den Kanten des Blechteils Grate, welche durch ein Entgrat-Verfahren entfernt werden müssen.

2.2 Entgraten als industrieller Fertigungsschritt

Das Entgraten ist in der metallverarbeitenden Industrie ein Fertigungsschritt, bei welchem Grate aus vorgelagerten Fertigungsschritten entfernt werden sollen. Ein Grat ist hier ein Teil des Werkstoffes, welcher an prozessbedingt an Bauteilecken- oder Kanten entsteht und nicht Teil der Soll-Geometrie

des Bauteils ist. Die meisten konventionellen Verfahren zum Entfernen eines ungewollten Grats fallen nach DIN 8587 bis DIN 8590 in den Bereich der zerteilenden, spanenden oder abtragenden Verfahren [4]. Im Rahmen dieser Arbeit werden Blechbauteile betrachtet, welche mit Tellerbürsten[1] geschliffen werden. Durch die Konstruktion der Tellerbürsten wird der Anpressdruck an den Bauteilkanten maximiert, wodurch eine Kantenverrundung erzielt werden kann.

Die Kantenverrundung wird im Allgemeinen über die Angabe des Kantenradius quantifiziert. Dieser ist der gemittelte Radius des Kantenprofils. Um diesen zu berechnen, wird zunächst ein Kantenbereich ermittelt (siehe Abbildung 2.1). Dieser beginnt an jenen Punkten, an welchen die Oberflächentangenten (grün) sich vom tatsächlichen Kantenprofil (blau) trennen. Für diesen Kantenbereich wird mittels Gauß-Fit ein Kreisprofil (gelb) angelegt, dessen Radius dem numerischen Kantenradius entspricht [6]. Darüber hinaus bestimmt der Schnittwinkel der beiden Oberflächentangenten den Kantenwinkel der gemessenen Kante. Für die Bestimmung der Qualität des Entgrat-Prozesses ist zusätzlich zum Kantenradius die Symmetrie des Kantenprofils entscheidend. Die Abstände der beiden Ablösepunkte von Oberflächentangenten und realem Profil zum Tangentenschnittpunkt sollten im Idealfall identisch sein. Das Verhältnis dieser Abstände wird in diesem Kontext K-Faktor genannt [7] und es sollte dementsprechend $K \approx 1$ gelten[2].

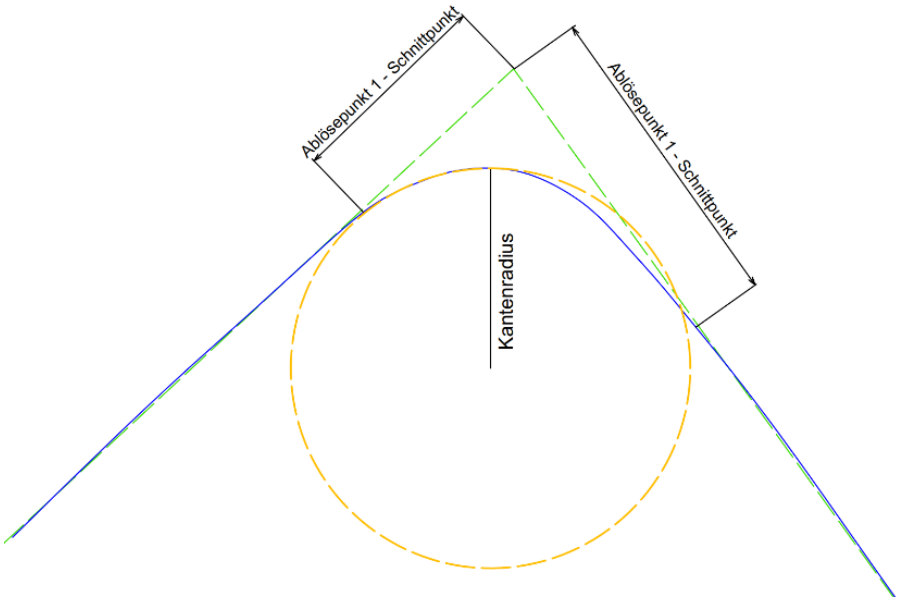


Abbildung 2.1: Schematische Bestimmung des Kantenradius aus Linien-Kantenprofil [8] (Grafik angepasst)

Der Entgrat-Prozess sollte so ausgelegt sein, dass Mindestanforderungen möglichst wenig überschritten werden. Das abzutragende Materialvolumen und damit der Werkzeugverschließ verhält sich quadratisch zum erzielten Kantenradius. Daher ist eine zu konservative Wahl der Prozessparameter

[1] Entspricht nach DIN 8589 einer Zerspanung mit geometrisch unbestimmter Schneide (Stirnplanschleifen) [5].

[2] In dieser Arbeit erfolgen alle Kantenprofil-Messungen durch einen optischen Laser-Scanner, welcher durch seine Messungen Profilansichten gemäß Abbildung 2.1 erzeugt, siehe weiterführend 3.5.1

nicht wirtschaftlich. Ein Messschritt ermöglicht dementsprechend auch das optimale Einstellen der flexiblen Prozessparameter für den Entgrat-Prozess. Die Menge der Systemvariablen eines Entgrat-Prozesses macht eine sinnvolle heuristische Aussage zur Qualität des Entgrat-Prozesses direkt aus den Prozessparametern praktisch nicht möglich. Daher wird in der Praxis überwiegend ein sensorische Messschritt vorgesehen, bei welchem direkt auf die Messgröße geschlossen wird.

Entgraten ist in einem Fertigungsprozess generell dann vorzusehen, wenn eine konkrete Kantenbeschaffenheit hergestellt werden muss. Diese sind oft normativ empfohlen, oder durch Pflichtenhefte genauer spezifiziert [9]. Gründe für eine normative Empfehlung sind unter anderem die Vorbereitung des Werkstückes für nachfolgende Fertigungsprozesse oder zur Steigerung der Langlebigkeit. So gibt die DIN 1090 für Stahl- und Aluminium-Tragwerke (Lasttragende Bauteile) für verschiedene Anwendungen konkrete Zahlenwerte zur Kantenverrundung vor [10,11][3]. Weitere Gründe für eine Entgratung können ästhetischer Natur sein, aber auch zur Steigerung der Arbeitssicherheit wird im industriellen Kontext gelegentlich eine Entgratung der Bauteile empfohlen [14,15]. Aufgrund der vielen Gründe für die Entgratung eines Bauteils und der hohen Anforderungen bezüglich der Prozessgüte ist es in der gegenwärtigen blechverarbeitenden Industrie unerlässlich, derartige Bearbeitungsschritte verlässlich und effizient vornehmen zu können.

2.3 Grundlagen für die Entwicklung und Konstruktion

Die im Folgenden aufgeführten Grundlagen setzen das Fundament für den wissenschaftlichen und industriellen Kontext dieser Arbeit. Die zwei großen Entwicklungsschwerpunkte sind der *Robotik* zuzuordnen. Im Rahmen der Entwicklungsphase des Demonstrators (siehe Kapitel 3 und 4) lassen sich die zu lösenden Kernprobleme weiter in die zwei Bereiche *Roboter-Kinematik* und *Machine Vision*, sowohl in Hard- als auch Software aufteilen. Die Ausrichtung und Steuerung des Roboters fallen unter ersteren Bereich, und die Bauteilerkennung unter letzteren. Der in dieser Arbeit entwickelte Demonstrator genügt der Definition eines Industrieroboters nach VDI-Richtlinie 2860, Blatt 1 [16].

2.3.1 Kinematik & Steuerung

Kinematik ist in der Mechanik die Lehre der Bewegung mechanischer Systeme ohne dynamische Einflüsse [17]. In der Robotik wird deshalb eine Maschine, welche über bestimmte Aktoren und Lagerungen eine definierte, berechenbare Bewegung erzeugt, oft über die Art und Anzahl der Bewegungsfreiheitsgrade (DOF) definiert. Unterschieden wird hierbei üblicherweise zwischen

[3] Gründe hierfür sind beispielsweise das Vermeiden von Kantenflucht bei Lackier-Prozessen am Werkstück[12]. In der DIN 8501-3 werden die gewünschten Kantenparameter spezifiziert [13]. Diese Werte werden häufig pauschal gefordert und müssen entsprechend präzise umsetzbar sein.

translatorischen und *rotatorischen* DOF[4]. Ausgehend dieser DOF können anhand der Lagerpunkte der Aktoren am Roboter iterativ über Lineartransformationen aus den Minimalkoordinaten der Roboter-Aktoren die Raumkoordinaten jedes Lagerpunktes bestimmt werden. Im umgekehrten Fall muss zur Bestimmung einer Lage der Kinematik aus gewünschten Raumkoordinaten des Roboters eine Rücktransformation vorgenommen werden, welche je nach Achsgeometrie nicht eindeutig ist [17]. Über die Lage der Lagerpunkte kann die Position jedes Punktes des Roboters immer exakt bestimmt werden. Diese Annahmen setzen vollständige Informationen über die Achsgeometrie des Roboters voraus, wobei eine Achse die Verbindung zweier Lagerpunkte ist. In modernen Industrierobotern wird unter den Achsen zwischen *Haupt-* und *Nebenachsen* (Handachsen) unterschieden [16]. Im Allgemeinen bieten die Hauptachsen genügend DOF, jeden Punkt im Arbeitsraum anzufahren, wobei die Handachsen zusätzliche Flexibilität bei der Ausrichtung liefern.

Weiterhin können Kinematiken nach serieller und paralleler Ausführung unterschieden werden. Bei parallel gestalteten Kinematiken können mehrere Achsen auf demselben Lager liegen, was im Allgemeinen zusätzliche kinematische Abhängigkeiten in das System einbringt. Bei seriell gestalteten Kinematiken sind alle unabhängig geschalteten Aktoren in Reihe geschaltet. Das Ende einer Achse ist dementsprechend der Lagerpunkt der nächsten Achse. Der Freiraum, den ein Roboter aufgrund seiner kinematischen Gestaltung für seine Bewegung benötigt, ist dessen Kollisionsbereich. Wichtig ist, dass der Bereich, in dem der Roboter Arbeit verrichten soll, im Allgemeinen nicht mit dessen Kollisionsbereich übereinstimmt. Ersterer ist der Arbeitsbereich des Roboters [19]. Der Kollisionsbereich schließt den Arbeitsbereich per Definition immer vollständig ein [20].

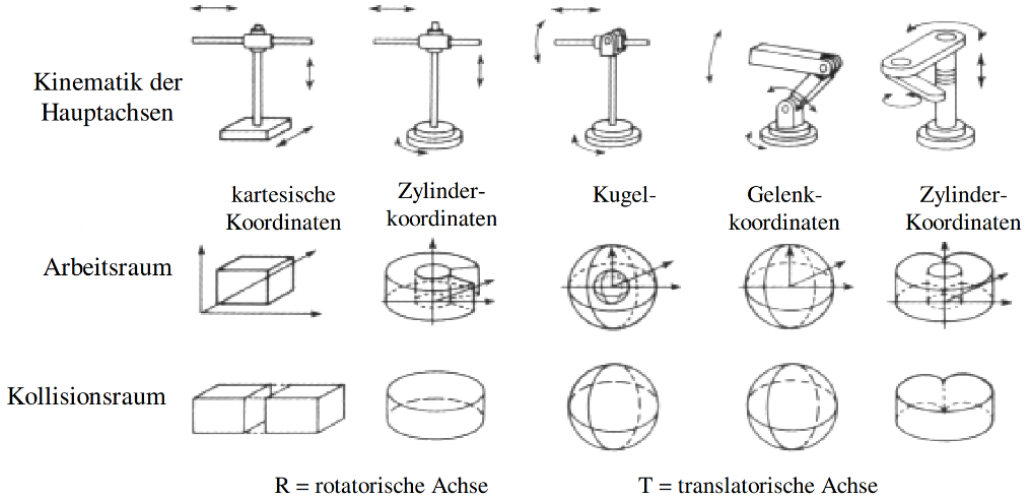


Abbildung 2.2: Roboter-Kinematiken mit: (v. o. n. u) DOF, Arbeitsraum, Koollisionsraum [16]

[4] Die Spezifikation eines Roboters erfolgt gelegentlich allein über die Angabe der Freiheitsgrade. So kann die im Rahmen dieser Arbeit verwendete Raumportalkinematik so als 4-TTTR-Kinematik beschrieben werden (T = translatorisch, R = rotatorisch) [18] (siehe 3.5.2 für Details der verwendeten Kinematik)

Die Achsen eines Roboters werden zum Antrieb mit geeigneten Aktoren ausgestattet. In den meisten normalen Anwendungen werden die DOF über Rotationsmotoren mit geeigneten Getrieben realisiert. Im Rahmen dieser Arbeit werden hierzu ausschließlich Schrittmotoren verwendet. Dieser wird über einen Pulsgeber in disktreten Schritten angetrieben. Das Wirkprinzip für dieser Motoren ist die elektromagnetische Reluktanz-Kraft [21].

SPS

Ein mechanisches System muss grundsätzlich gesteuert werden. Die Aufgabe einer Steuerungseinheit ist das Kontrollieren der Aktorik des Systems, um eine koordinierte Bewegung zu realisieren. Bei teil- bis vollautomatisierten Prozessen müssen diese Steuerungseinheiten dementsprechend programmierbar sein. Die jeweiligen Programme enthalten Instruktionen für die Abfolge von Prozeduren und logische Verknüpfungen zum Realisieren der gewünschten Bewegungsabläufe der Maschine. Mittlerweile werden für diese Zwecke überwiegend Speicherprogrammierbare Steuerungen (SPS[5]) verwendet [22]. Durch eine Mikroprozessorbasierte Rechenstruktur bieten SPS eine flexible Plattform zur einfachen Integration automatisierter industrieller Prozesse [22,23]. SPS sind derartig präsent in der Industrie, dass diese nach DIN EN IEC 61131 genormt sind. In dieser sind die Struktur einer SPS aufgeführt [24], aber ebenfalls Programmiersprachen- und Formate festgelegt, mit welchen eine SPS kompatibel sein soll [25,26]. Die SPS wird ausgangsseitig über passende Hardware-Treiber mit den verwendeten Aktoren verbunden. Über Sensoren können Stellgrößen überwacht und zur eingangsseitig SPS zurückgeführt werden. Eine SPS ist somit theoretisch ein geschlossenes Regelsystem, welches lediglich von außen überwacht werden muss, um Informationen über den Status der Maschine zu erhalten. Ein Eingriff in die internen Prozeduren durch einen externen Computer ist nicht nötig.

Die Abgrenzung zwischen einem Mikrocontroller und einer modernen SPS verschwimmt. SPS grenzen sich vor allem durch die sichergestellte Industrietauglichkeit und die normativ geregelte Programmierbarkeit ab. Zudem sind zusätzliche Module leicht in das System der SPS integrierbar. Im Allgemeinen kann mit der entsprechenden Hardware-Konfiguration mit einem Mikrocontroller dieselbe Funktionalität wie mit einer SPS hergestellt werden. Für SPS existieren in der Praxis häufig Frameworks, welche das Erstellen einer Steuerungslogik vereinfachen. Siemens entwickelt zu diesem Zweck das *Totally-Integrated-Automation-Portal*. Anders als bei herkömmlichen Mikrocontrollern erfolgt die Programmierung so auf höheren Ebenen der Abstraktion, was die Handhabung der Programmierung intuitiver gestalten soll [27].

[5] Im Englischen als Programmable Logic Controller (PLC) bezeichnet, die Bezeichnung wird mittlerweile auch im deutschen Sprachraum übernommen

Für den in dieser Arbeit verwendeten Roboter ergibt sich eine Konstruktion aus einem Computer, welcher der vorprogrammierten SPS anzufahrende Raumkoordinaten im Messbereich des Demonstrators übergibt. Mit der SPS werden Schrittmotoren über entsprechende Motortreiber verbunden (siehe Abbildung 2.3).

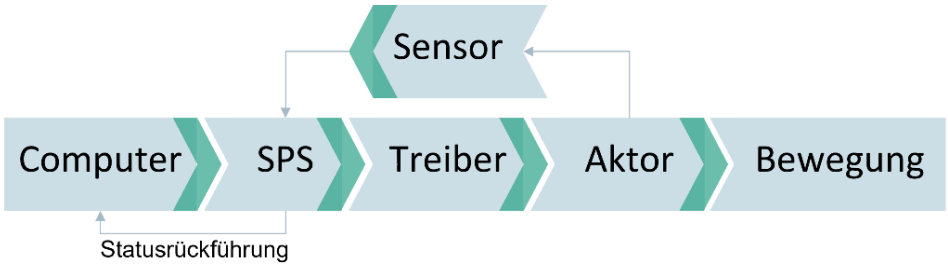


Abbildung 2.3: Aufbau eines Kinematik-Steuersystems mit SPS

2.3.2 Machine Vision & Kamera

Die Prämisse von *Machine Vision*, beziehungsweise *Computer-Vision* ist das Schaffen einer sensorischen Schnittstelle zwischen der Realität und einem Computer auf Basis von Bildern [28,29]. Aus den abstrakten Informationen der zugrundeliegenden Bilddateien werden über Bildbearbeitungsverfahren Informationen abgeleitet, welche im Endeffekt ein „Sehen“ des Computers ermöglichen. In der Praxis wird Machine Vision unter anderem für maschinelle Messungen, zur Prozessüberwachung, zur Objekterkennung und zu Steuerungszwecken als Sensor verwendet. Gegenstand von Machine Vision ist im Gegensatz zur optischen Sensorik nicht die hardwareseitige Messtechnik, sondern Algorithmen zur Messauswertung und zur Interpretation der Sensordaten [30].

Im Rahmen dieser Arbeit wird für alle Machine-Vision-Algorithmen die Open-Source-Bibliothek *OpenCV* verwendet. Da die Bilder im Allgemeinen als Pixeldaten in Tensoren vorliegen, handelt es sich bei den meisten Bildbearbeitungsoperationen mathematisch um Tensor-Multiplikationen, deren Rechenaufwand exponentiell zur Datenmenge steigt [31]. Eine numerisch effiziente Implementierung dieser Operationen ist für die meisten industriellen Computer-Vision-Anwendungen essenziell, um in kurzer Rechenzeit brauchbare Ergebnisse zu liefern. OpenCV ist hierfür eine weit verbreitete Bibliothek und bewirbt sich als tauglich für Echtzeitanwendungen [32]. Mittlerweile gehen viele Computer-Vision-Anwendung mit der Nutzung einer künstlichen Intelligenz einher. So soll der Automatisierungsgrad weiter gesteigert werden [33]

Softwareseitig sind viele Machine-Vision-Anwendungen ähnlich gestaltet. In der Praxis sind diese in der Regel intuitiv aufgeteilt in: Bilderstellung, Vorverarbeitung, Segmentierung und Anwendung der Messalgorithmen, beispielsweise einer Kantendetektion, der Informationsauswertung und einer abschließenden Entscheidungsfällung [29,34]. Segmentierung beschreibt den Prozess des Aufteilens des Bildes in zusammenhängende Bereiche, wobei Kriterien vorgegeben werden können, nach

welchen die Bereiche ermittelt werden [35]. Diese Bereiche können anschließend qualitativ untersucht werden.

Die Bilddaten werden von Foto- oder Video-Kameras erzeugt. Da die Kameras in der Regel direkt mit einem Rechner kommunizieren und vor allem bei Echtzeitkommunikation hohe Datenmengen übertragen, existieren verschiedene Schnittstellen- und Übertragungsstandards. Hierzu zählen unter anderem *GigE Vision*, basierend auf TCP-IP, *USB3 Vision* auf Basis von USB3 und *MIPI CSI-2* auf Basis von digitalen Rohdaten des CMOS-Chips. McCURRACH [36] liefert einen Überblick zu den Vorteilen der unterschiedlichen Protokolle und Standards. Über Software-Development-Kits (SDKs) können diese Standards für verschiedene Anwendungen genutzt werden. Die Standards sind im Allgemeinen nicht Open-Source und werden von privat geführten Organisationen entwickelt und lizensiert [36].

Kamera-Fehler & lineare Optik

Eine moderne Digitalkamera besteht in der einfachsten Ausführung aus einem Sensor und einem Objektiv. Dies entspricht der Ausführung, welche im Rahmen dieser Arbeit verwendet wird. Bei der Bildgebung durch eine Digitalkamera entstehen verschiedene Fehler, welche es zu minimieren gilt. Die Ursachen liegen in den optischen Charakteristiken des Objektivs und des Kamera-Sensors. Für die Beschreibung dieser Fehler wird häufig das *Lochkamera-Modell* aus der linearen Optik als Ansatzpunkt genutzt [37–39].

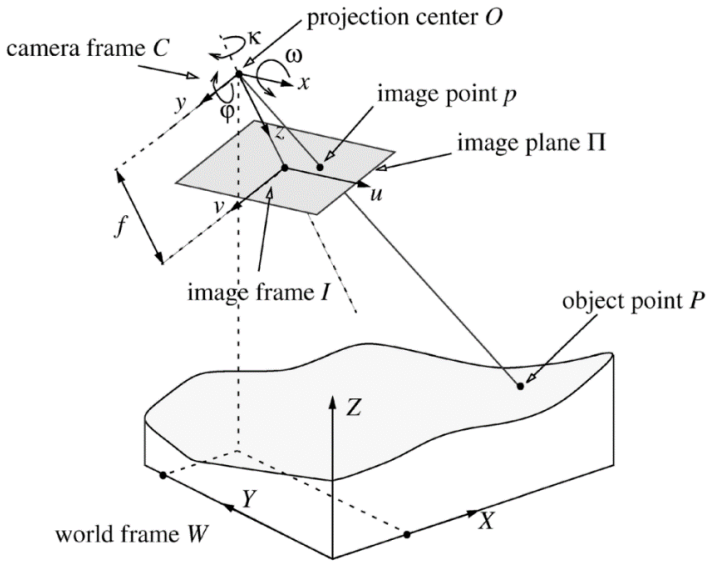


Abbildung 2.4: Lochkamera-Modell mit beschreibenden Parametern (engl.) [38]

In diesem Modell werden Objektpunkte räumlich unverzerrt, auf die Bildfläche projiziert. Die Abbildungen sind linear, was die mathematische Handhabung dieses Modells vereinfacht. Durch die Verwendung von Linsenobjektiven entstehen allerdings Verzeichnungen, welche entweder durch die

Konstruktion des Objektivs oder durch eine Kalibriermaßnahme an den geometrischen Parametern der Kamera-Abbildung ausgeglichen werden können [37].

Abbildung 2.5: Verzeichnungstypen bei Linsenobjektiven (v.l.n.r. ohne, Kissen- und Tonnen-Verzeichnung) [40]

Perspektivische Verzerrungen sind eine weitere Fehlerursache bei Machine-Vision-Anwendungen. Für ein Kamerabild werden Objekte im dreidimensionalen Raum auf eine zweidimensionale Fläche projiziert. Bei einer Draufsicht auf ein dreidimensionales Objekt werden im Allgemeinen auch Flächen auf den Kamerasensor projiziert, welche nicht direkt der Bildebene zugewandt sind. In der Praxis bedeutet dies, dass auch Flächen dargestellt werden, die nicht parallel zur Bildebene liegen. Bei einem Blechteil im Sinne dieser Arbeit sind das z.B. die Seitenflächen oder Innenflächen von Löchern im Bauteil. Diese sind abhängig von den Oberflächen und der Belichtung optisch nicht unterscheidbar von der abzubildenden Oberseite und produzieren so einen nicht vernachlässigbaren Abbildungsfehler.

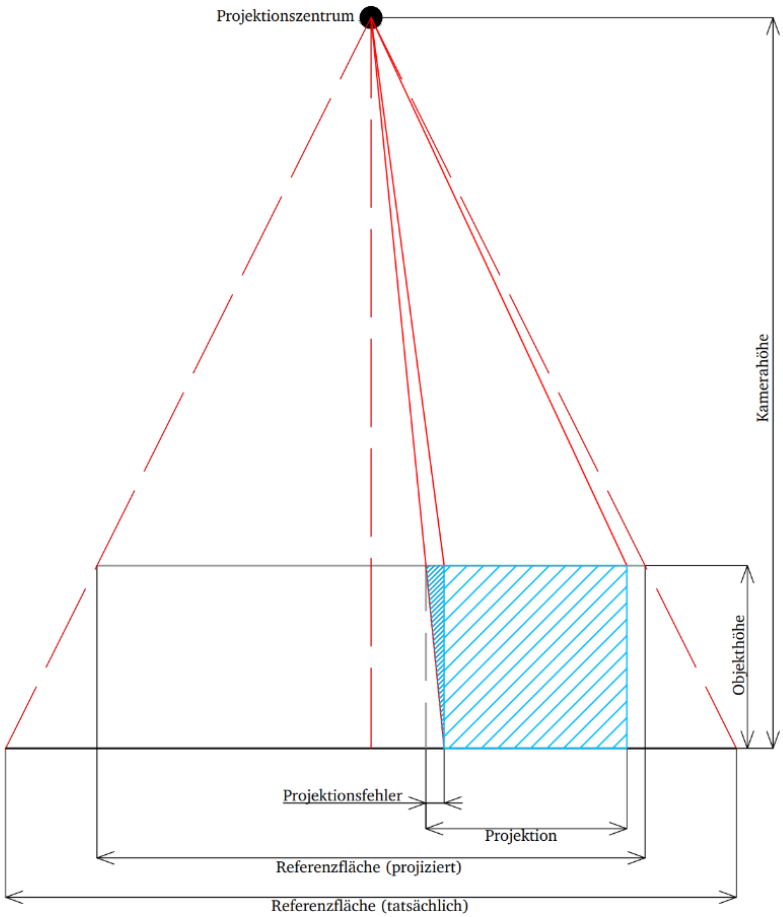


Abbildung 2.6: Projektionsfehler bei einem quaderförmigen Objekt (blau) im Lochkamera-Modell

Aus Abbildung 2.6 gehen drei Modi der perspektivischen Verzerrung hervor. Für die Bemaßung von Objekten auf Bildern ist es sinnvoll, ein Referenzmaß vorzusehen, welches als Vergleich zum Messobjekt genutzt werden kann. Zunächst wird das Objekt durch seine Höhe und den nicht-parallelen Projektionsachsen vergrößert auf der Bildebene projiziert. Der Faktor dieser Vergrößerung entspricht dem Verhältnis der tatsächlichen Referenzlänge und der projizierten Fläche entlang der Projektionsachsen (rot) auf Höhe der Objektoberseite. Zusätzlich wird die Kantenfläche (blau, fein schraffiert) auf die Bildebene projiziert. Diese durch die Perspektive hervorgerufenen Abbildungsfehler sind immer der optischen Achse (rote Mittellinie) der Kamera zugewandt. Aufgrund des gekrümmten Blickfeldes erscheint das Objekt außerdem radial weiter von der optischen Achse entfernt zu sein. Geometrisch wird dies anhand des Verhältnisses der Abstände von Ober- und Unterseiten des Objekts (blau) von den äußeren Projektionsachsen ersichtlich.

Diese Fehlerquellen werden im Rahmen der Software-Entwicklung in Kapitel 4.2.3 algorithmisch bearbeitet, sodass ein unverzerrtes Bild für die Erkennung und Lagebestimmung der Blechbauteile entstehen kann. Die verwendeten Algorithmen basieren auf den zuvor beschriebenen Paradigmen von Machine Vision.

2.4 Stand der Technik

Die Sparte der dieser Arbeit zugrunde liegenden Entwicklung ist die industrielle Qualitätssicherung in der blechverarbeitenden Industrie in Technologie und Methodik. Ausgehend von vergangenen und gegenwärtigen wissenschaftlichen Entwicklungen wird die industrielle Praxis plausibilisiert. Mit Hinblick auf die Konstruktion des Demonstrators wird der Inhalt dieser Arbeit in die aktuelle Innovationslandschaft eingeordnet.

2.4.1 Relevante wissenschaftliche Arbeiten

Technologien zur automatisierten Qualitätskontrolle sind heutzutage in der Industrie allgegenwärtig. Die Erhöhung des Automatisierungsgrades in der Industrie ist in der Regel durch eine niedrigere Fehleranfälligkeit und durch genauere Prozesstoleranzen motiviert.

Optische Messungen, meist basierend auf Laser-Technologien sind heutzutage etabliert und weit verbreitet. Optische Technologien sind seit Beginn des 20. Jahrhunderts ein Treiber technischer Innovationen [41]. Sie bieten einen berührungslosen Weg, mit hoher Genauigkeit und niedriger Messunsicherheit in ein bis drei Raumdimensionen geometrische Messungen vorzunehmen [42,43]. HERING liefert eine umfassende Beschreibung der Messverfahren in der optischen Sensorik [43]. Machine Vision ist in vielen Anwendungen nach Definition ebenso in optische Messverfahren einzuordnen, da das primäre Messwerkzeug eine Kamera ist.

RABABAAH et al. entwickelten ein automatisches System zur maschinellen Inaugenscheinnahme von Blechwerkstücken [34]. Hier werden die untersuchten Teile auf Konturfehler untersucht. Zweck dieses Verfahrens soll keine detaillierte Messung der Werkstücke sein, sondern ein schneller Prozess zum Aussortieren grober Fehlproduktionen, wobei quantitative Vergleichswerte geliefert werden. Der Algorithmus wurde an synthetischen Bilddaten getestet, die durch die Projektion von Lochkonturen auf Metalltexturen erzeugt wurden. Dadurch wurde eine unverzerrte Draufsicht simuliert. Mit den herkömmlichen Methoden der Bildverarbeitung für Machine Vision entsprechend 2.3.2 werden die Konturen der Bauteile qualitativ mit Referenzdaten verglichen und ein prozentualer Gleichheitswert ermittelt. So kann zur Entscheidungsfällung ein Mindest-Prozentsatz zur Akzeptanz eines Werkstücks vorgegeben werden, wodurch eine vollautomatische Messung ermöglicht ist.

SHAO et al. entwickelten ein System zur Überprüfung eines Schrittes in der Fertigung von Blechbauteilen mit komplexer Oberflächengeometrie [44]. Dazu wird ein CNC-Koordinatenmessgerät[6] genutzt, um bei großen Blechbauteilen direkt auf der Arbeitsfläche der Bearbeitungsmaschine einen 3D-Oberflächenscan des Werkstücks vorzunehmen. Dieser Scan wird mit CAD-Daten verglichen, indem algorithmisch diskrete Prüfpunkte ermittelt und die beiden Modelle optimal zueinander ausgerichtet werden. Nach dieser Optimierung werden die Abweichungen der Datensätze an den Prüfpunkten ermittelt und so die Formabweichung der Oberflächenprofile zwischen Messung und CAD-Referenzmodell berechnet. Der Fokus dieses Verfahrens liegt auf der einfachen Integrierbarkeit in bestehende Prozesse, sowie einer numerisch effizienten Implementierung.

LIN et al. entwickelten einen Algorithmus, welcher die Laufzeit von 3D-Prüfverfahren maßgeblich optimieren sollte [45]. Um dies zu erreichen, wurde zunächst eine 2D-Aufnahme des zu vermessenden Werkstücks erstellt. Daraus werden über ein Segmentierungsverfahren entsprechend 2.3.2 zunächst Messpunkte extrahiert. Extrahiert werden somit nur spezielle Punkte, welche explizit vermessen werden sollen. Diese Punkte werden anschließend über herkömmliche 3D-Scan-Verfahren als Punktwolke mit anschließender Tessellation (Parkettierung) zu einem selektiven 3D-Modell zusammengefügt, an welchem softwareseitig weitere Messdaten erzeugt werden können. Durch die selektiven Scans kann eine substanzielle Laufzeitverbesserung des Messalgorithmus erreicht werden. Ebenso wird darauf verwiesen, dass eine industrielle *Online*-Implementierung dieser Technologie [46] perspektivisch realistisch ist.

Zusammenfassend gibt es wenig erkennbare chronologische Zusammenhänge in den wissenschaftlichen Innovationen auf dem Gebiet der automatisierten Qualitätssicherung in der Industrie. Auffällig ist, dass zeitgenössische Trends der Methodik besonderen Einzug in die Forschungsschwerpunkte halten. So sind aktuell viele Ansätze mit KI-gestützten Systemen zu finden, wobei gerade die

[6] Entspricht hier hardwareseitig einer 3-TTT Kinematik mit Abstands-Messgerät (siehe 2.3.1

verwendete Hardware innovationstechnisch zu stagnieren scheint [47]. Die aktuelle Innovationslandschaft scheint mit Priorität Nutzen aus der steigenden Rechenleistung der verfügbaren Computer ziehen zu wollen. Bemerkenswert ist jedoch, dass gerade Machine-Vision-Anwendungen bis vor einigen Jahren tendenziell prototypische Technologien waren, die in ihrer Implementierung aber wenig praxistauglich waren [34,44]. In der näheren Vergangenheit entstehen mehr Konzepte für eine Echtzeit-Implementierung in der Industrie [33,45].

2.4.2 Durchführung in der Praxis/ Normative Anforderungen

Es existieren viele frei verfügbare Systeme, welche zum Prüfen von Werkstückeigenschaften, so auch der Kantenradien, konzipiert sind. Nach wie vor repräsentiert die manuelle Messtechnik nach eingängiger Recherche hier die größte Sparte der kaufbaren Technologien. Handgeführte Werkzeuge, vorprogrammierbare Kinematiken, und starr montierte Messsysteme überwiegen quantitativ den Angeboten für vollautomatisch operierende Systeme. Auffällig ist auch ein Mangel an vorkonfektionierten Systemen aus Messgerät und Kinematik.

In der Praxis sind industrielle Messverfahren zur Qualitätssicherung meist normativ geregelt. Diese geben eine Liste an Anforderungen vor, welche die qualitätssichernden Systeme zu erfüllen haben, wobei die genaue Durchführung im Allgemeinen nicht in letzter Konkretisierung vorgeschrieben ist.

Die ISO 9001 ist die wichtigste internationale Norm für die Definition der Anforderungen an Qualitätsmanagementsysteme. Sie gibt allgemeine Empfehlungen für die Struktur und die Entwicklungszyklen von Wertschöpfungsprozessen zur Qualitätsbestimmung- und Sicherung in jeder wirtschaftlichen Sparte vor [48]. Entsprechend der Norm muss das Verfahren zur Qualitätssicherung klar definiert, die zu bewertenden Größen eindeutig messbar, der Prozess ständig überwacht und angemessen bewertbar sein. Technische Prüfstellen wie der TÜV bieten beispielsweise in Deutschland Zertifizierungen im Sinne dieser Norm an [49]. Unter anderem schreibt die Norm auch vor, dass Prüfmittel in regelmäßigen Abständen zu kalibrieren sind, insofern die Messergebnisse hiervon abhängig sind. Für technische Entwicklungen im Bereich der industriellen Qualitätssicherung ist es aufgrund der weiten Anwendung der ISO 9001 attraktiv, deren Anforderungen gerecht zu werden.

Die DIN EN ISO 10012 baut auf ISO 9001 auf und definiert konkrete Anforderungen an Messmittel und Messprozesse [50]. Ein besonderer Fokus der Norm ist die Verständlichkeit und Dokumentierung der Messprozesse. So soll Messsoftware zu jedem Zeitpunkt strukturell durchsichtig, und auf Fehlerfreiheit geprüft sein. Die Verwendung ist nur zulässig, wenn letzteres absolut sichergestellt ist. Weiterhin soll es möglich sein, eine korrekte Kalibrierung des Systems einfach im Betrieb zu verifizieren. Messparameter wie Messunsicherheit, Stabilität, maximale Messabweichung, Wiederholbarkeit, sowie Fähigkeiten des Bedieners müssen bestimmt und quantifiziert werden, um den

Messprozess zu bewerten. Auffällig ist, dass für ein automatisches System Einflüsse des Bedieners weitgehend hinfällig wären.

Im Sinne dieser Arbeit sind besonders die normativen Anforderungen an den Entgrat-Prozess relevant, welche von dem hier entwickelten Demonstrator überprüft werden sollen[7]. Hieraus lassen sich die Messtoleranzen ableiten, welche die verwendbare Messtechnik definieren. Für die Ausführung von Stahl- und Aluminiumtragwerken als funktionskritische Bauteile vor allem im Bausektor[8] ist die DIN EN 1090 eine in der Praxis häufig zitierte Richtlinie [10,11,52–54]. Diese gibt unter anderem Anforderungen für notwendige Fertigungsprozesse von Stahl- und Aluminiumtragwerken vor. Für den Korrosionsschutz müssen diese beschichtet werden, wofür die DIN EN ISO 8501-3 konkrete Werte zur Kanten- und Oberflächenbehandlung vorgibt [13].

2.4.3 Folgerungen für die vorliegende Arbeit

Es scheint keine marktgängigen kompakten, leicht skalierbaren und fehlertoleranten Lösungen zum automatisierten Prüfen der Kantenverrundung von Blechteilen zu geben. Während Konzepte zur automatisierten Überprüfung der Oberfläche durch Machine Vision existieren, fällt eine große Lücke im Bereich der automatisierten Qualitätssicherung der Kantenbearbeitung auf. Hier korrekt durchgeführte Prozesse zu gewährleisten, ist allerdings gerade in Bezug auf einzuhaltende Normen, Richtlinien und Pflichtenhefte von großer Bedeutung. Während in der Wissenschaft ein Trend in Richtung KI-unterstützter vollautonomer Prüfsysteme erkennbar ist [29,33], sind momentan wenige praktische Anwendungen dieser Technologien erkennbar. Für Detailmessungen, wie beispielsweise der Kantenverrundung, sind rein kamerabasierte Technologien zudem in der Regel unbrauchbar. Das Verhältnis aus Pixelanzahl und zu vermessender Bauteilfläche macht Detailmessungen zu einer Herausforderung seitens der aktuell verfügbaren Kameratechnik.

Generell scheint es wenige praktische Anwendungen der Kombination von Bildgebung, Objekterkennung durch 2D-Machine Vision und einem anschließenden Messschritt über genauere optische Sensorik (hier Laser-Scanner) zu geben. Diese Kombination ermöglicht eine flexible und schnelle Bildgebung über Machine-Vision-Technologien und realisiert gleichzeitig eine genaue Messung an spezifizierten Messpunkten. Arbeiten wie von LIN et al. [45] (siehe oben) zeigen das Potential einer selektiven Messung.

Da viele in der Industrie verwendeten Bauteile und Komponenten unter die hiesige Definition eines Blechbauteils nach 2.1 fallen, sind komplexe 3D-Scans des Bauteils häufig schlicht nicht notwendig.

---

7 Für Normverweise im Kontext des industriellen Entgratens siehe auch Abschnitt 2.2

8 Relevanteste Abnehmerbranche für Stahl- und Stahlerzeugnisse in Deutschland mit 35% Anteil am gesamten Stahlbedarf in Deutschland im Jahre 2020 [51].

Wenn dieser Schritt durch einen einfachen 2D-Scan in Form eines Bildes ersetzt wird, kann die Messung wie im Folgenden beschrieben in schnell und mit großer Fehlertoleranz realisiert werden. Prozessanalytisch ist abhängig von der Dauer des Messprozesses mindestens eine *Online*-Messung möglich [46]. Durch die Struktur und geplante Funktionsweise des hier entwickelten Systems wird eine robuste, gut skalierbare Möglichkeit geschaffen, die Kantenverrundung von Blechbauteilen entsprechend geltender Normen, Richtlinien und etwaiger Pflichtenhefte zu prüfen.

3 Anforderungen an das Gesamtsystem und theoretische Entwicklung

Ausgehend von den skizzierten Anforderungen wird in den folgenden Kapiteln ein System zur automatisierten Kantenmessung von Blechbauteilen entwickelt. Vor der theoretischen Ausarbeitung werden zunächst die allgemeinen Methoden der Entwicklung aufgeführt und der nachfolgende Entwicklungsprozess in der Theorie erörtert. Anschließend werden ausgehend von den vordefinierten Systemanforderungen konkrete Komponenten ausgewählt, welche sich zu einem kompatiblen Gesamtsystem zusammenfügen lassen. Im Rahmen dieses Kapitels werden die Entscheidungen durch den theoretischen Entwicklungsprozess begründet.

3.1 Methodik der Entwicklung und Randbedingungen

Das V-Modell nach Abbildung 3.1 bildet den Grundsatz und den konzeptionellen Leitfaden der gesamten Entwicklung. Durch die vielen einzelnen Subsysteme, welche im Rahmen dieser Arbeit entwickelt und schlussendlich zusammengefügt werden, ist die Entwicklung nach dem V-Modell sinnvoll. Die folgenden Kapitel behandeln jeweils einzelne Phasen der Entwicklung, welche sich ebenfalls klar durch das V-Modell abgrenzen lassen.

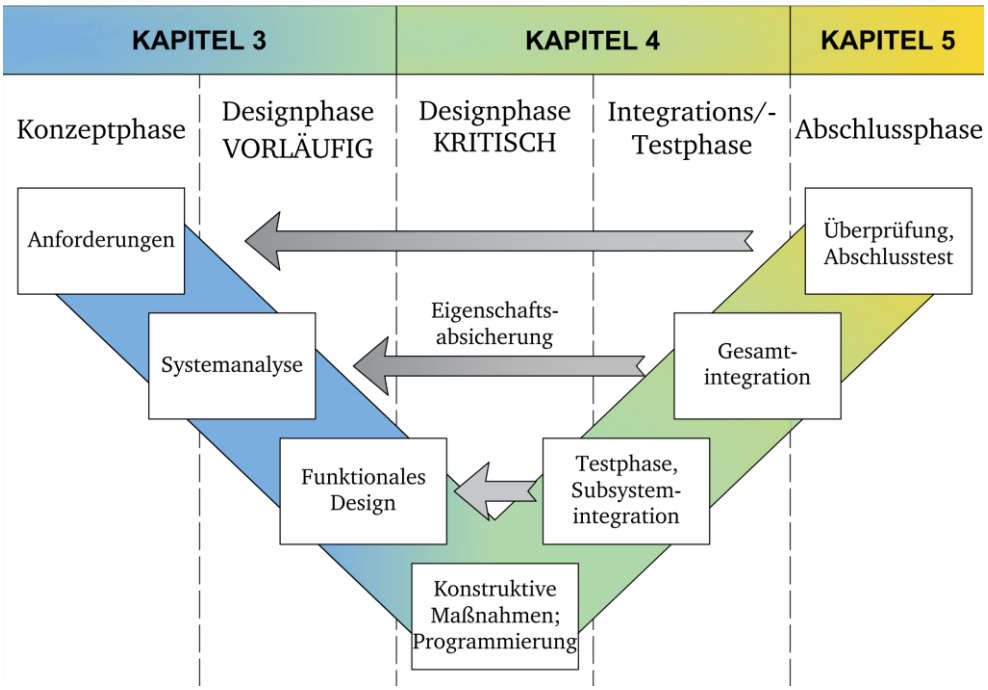


Abbildung 3.1: V-Modell als generische Entwicklungsmethodik

Zunächst werden in der *Konzeptphase* qualitative Anforderungen an das Gesamtsystem formuliert. Dieser Schritt erfolgt auf Basis der vorgesehenen Funktionsweise und Randbedingungen an die Entwicklung. Eine Randbedingung ist hier ein Aspekt der Entwicklung, welcher keinen direkten Einfluss auf die vorgegebene Funktionsweise des Gesamtsystems hat, aber dennoch zu erfüllen ist. Derartige Anforderungen sind klassischerweise in spezifischen Pflichtenheften vermerkt.

Aus der Gesamtheit dieser Anforderungen kann im Rahmen der *Systemanalyse* eine Liste an Ansätzen zur Problemlösung abgeleitet werden. Ansätze sind in diesem Falle Technologien, die zur Problemlösung beitragen, aber zunächst keinen Anspruch auf Kompatibilität untereinander haben. In diesem Schritt werden also Möglichkeiten gelistet, den einzelnen Funktionen des Gesamtsystems gerecht zu werden.

Aus den einzelnen Technologien kann schematisch ein prototypisches *funktionales Design* mit Anspruch auf Kompatibilität der einzelnen Subsysteme konstruiert werden. Die Auswahl der Komponenten folgt der Methode nach Abbildung 3.2, welche auf einem morphologischen Kasten basiert [55]. Die Auswahl der konkreten Komponenten richtet sich sinnvollerweise nach jenen Komponenten, welche wenig oder keine Auswahl zulassen. Diese bieten wenig Spielraum bezüglich der Erfüllung anderer Anforderungen und werden demnach mit Priorität genau festgelegt. Unter Berücksichtigung aller internen Abhängigkeiten können so iterativ konkrete Komponenten ausgewählt werden.

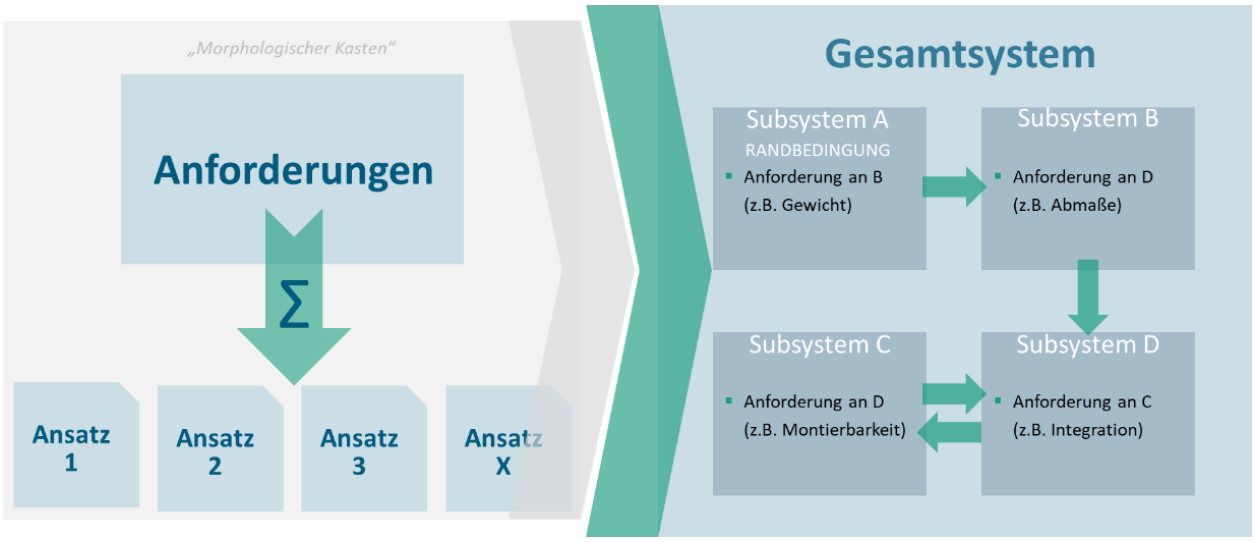


Abbildung 3.2: Vertiefende Methodik der Entwicklung

Aus den Ergebnissen der Systemanalyse und der Erstellung des ersten funktionalen Designs kann ein prinzipieller Programmablaufplan (PAP) erstellt werden. Dieser bietet gerade für die Entwicklung der Software ein Schema zur Steuerung der systeminternen Prozesse.

Am Ende dieser Phase existiert der Entwurf für das Gesamtsystem in der Theorie, entsprechend der Systementwurfsphase nach dem V-Modell nach Abbildung 3.1. Der genaue praktische Entwurf der Subsysteme, sowie die Integration des Gesamtsystems folgt in der anschließenden Konstruktionsphase in Kapitel 4. Dort werden die Subsysteme praktisch umgesetzt, sowie notwendige Maßnahmen für eine Integration des Gesamtsystems getroffen. In Kapitel 5 folgt die Überprüfung der Anforderungen, sowie die Meta-Analyse der entwickelten Technologie.

3.2 Randbedingungen der Entwicklung

Für die Versuche steht eine automatische Entgrat-Maschine des Modells *F200* der Firma *Q-Fin* zur Verfügung. Diese ist in der Lage, Blechbauteile mit geometrischen Eigenschaften nach Tabelle 3-1 zu verarbeiten. Die Maximalwerte der Maschine werden unverändert als geometrische Anforderungen für den Demonstrator übernommen.

Als maximale Länge für kompatible Blechbauteile wird pauschal 300 mm festgelegt. Diese Begrenzung wird gewählt, da im Rahmen dieser Arbeit zunächst die prinzipielle Funktionalität der Technologie gezeigt werden soll. Anzustreben ist hierbei dennoch eine leichte Skalierbarkeit des Aufbaus, um eine etwaige Erweiterung Größenbegrenzung in der Praxis zu ermöglichen. So sind flexible Ausführungen der Technologie in der Praxis möglich, welche unterschiedlichen Anforderungen gerecht werden. Dies erhöht den theoretischen Einsatzbereich der Technologie direkt im Rahmen dieses einmaligen Entwicklungsprozesses.

Tabelle 3-1: Spezifikation Q-Fin F200 [56]

Max. Blech-Abmaße (Breite, Höhe)	200mm, 100mm
Min. Blech-Abmaße (Breite x Länge)	20x20 mm
Förderbandgeschwindigkeit	$0{,}3 - 5\ \frac{m}{min}$
Werkstoffe	Al, Stahl, Edelstahl

Ein massives Blechbauteil der maximalen Abmessungen 200 x 300 x 100 mm (b, l, h) wäre somit der obere Grenzfall bezüglich des Bauteilgewichts. Das Gesamtgewicht eines solchen Teils aus gewöhnlichem Baustahl mit Dichte von $\varrho_{Stahl} \approx 7850\ \mathrm{kg\,m^{-3}}$ beläuft sich auf 47,1 kg. Der Prüfstand muss also in der Lage sein, dieses Gewicht robust mit ausreichender Sicherheit gegen Versagen zu stützen. Um die freie Positionierbarkeit demonstrieren zu können, wird der zulässige Messbereich in der Breite, sowie der Länge mindestens 100 mm größer vorgesehen als das größtmögliche Blechbauteil, also 300 x 400 mm (b, l).

3.3 Funktionale Anforderungen an das Gesamtsystem

Die Hauptfunktion des Demonstrators entsprechend des Entwicklungskonzepts ist das automatische Messen der Kantenverrundung von Blechbauteilen mittels Laser-Scanner und Mehrachs-Kinematik. Zusätzlich soll diese Funktion robust gegenüber Fehlern aus vorherigen Schritten abgesichert sein. Dies wird erreicht, indem die Blechbauteile lediglich innerhalb einer Messfläche positioniert werden müssen, wobei die genaue Ausrichtung der Teile unerheblich sein soll. Im Rahmen dieser Arbeit werden die Blechbauteile starr positioniert und manuell in den Demonstrator eingelegt. Weiterhin sollen für die Objekterkennung und die anschließende Vermessung die vorliegenden Blechbauteile

jeweils mit einem Referenzdatensatz verglichen werden, woraus Metadaten wie Abmaße und Messpunktkoordinaten abgeleitet werden können. Dies soll mit gängigen CAD-Dateiformaten wie DXF[9] möglich sein, um eine maximal breite Softwarekompatibilität zu ermöglichen.

Zusätzlich zu den vordefinierten Messpunkten auf den Messobjekten soll es möglich sein, freie Messpunkte auf diesen zu wählen, wobei stets überprüft werden muss, ob es sich bei der Auswahl um einen messbaren Punkt auf einer Bauteilkante handelt. Alle Messpunkte müssen sich auf der Oberseite des Messobjekts auf konstanter Höhe befinden und die verwendeten Messobjekte müssen mindestens der Definition eines Blechbauteils entsprechend 2.1 genügen. Alle Koordinaten sollen zwecks Intuition und wegen des rechteckigen Messbereichs in kartesischen Koordinaten vorliegen und verarbeitbar sein.

Nach Erfassen des Objektes und dem Ableiten von Raumkoordinaten für die einzelnen Messpunkte sollen diese zeitlich optimiert nacheinander angefahren werden, wobei der Laser-Scanner an jedem dieser Punkte eine Messung des Bauteil-Kantenprofils vornehmen soll. Hierbei soll die Abweichung der Soll- und Ist-Lage der Kinematik maximal *1 mm* betragen, allerdings soll in jedem Falle bei der korrekten Auswahl eines Kantenpunktes eine valide Messung entstehen. Dies soll unabhängig von allen Ungenauigkeiten in der Manövrierfähigkeit mit hoher Robustheit gewährleistet sein.
Optimal wäre die Gewährleistung einer Messdauer kürzer als der Prozessdauer eines Entgrat-Prozesses bei beliebigem Förderbandvorschub der Entgrat-Maschine, sowie beliebiger Blechgröße. Die theoretisch optimale Messdauer, bei welcher ein pausenloser Betrieb der Entgrat-Maschine möglich wäre, ergibt sich nach Gleichung 3.1.

$$t_{\mathrm{Mess,opt}} = \frac{l_{\mathrm{Blech}}}{v_{\mathrm{Band}}} \tag{3.1}$$

l_{Blech} entspricht der Länge des Blechbauteils in Richtung des Förderbandes und v_{Band} der Vorschubgeschwindigkeit des Förderbandes der Entgrat-Maschine. Ein Ziel ist die Erfüllung dieser Bedingung zumindest für einige Kombinationen aus Vorschub und Blechlänge gemäß Tabelle 3-1. Diese Forderung soll der Präzision des Gesamtsystems nachgestellt werden.

Das Ergebnis dieser Messungen soll verarbeitet werden und dem Nutzer in einer intuitiv verständlichen Art und Weise angezeigt werden. Es soll eindeutig sein, ob ein Blechbauteil korrekt oder fehlerbehaftet bearbeitet wurde, wobei Fehler spezifisch gekennzeichnet werden sollen. Die Kommunikation zwischen den einzelnen Systemen soll über ein Lokalnetzwerk erfolgen, was die Anbindung eines Computers zum Ausführen des Programms und der Datenverwaltung vereinfacht. Kommunikation über Netzwerkprotokolle sind Stand der Technik im industriellen Bereich.

[9] DXF ist ein open-source Dateiformat und wird als Industriestandard von den meisten CAD-Programmen und CNC-Anwendungen zum Austausch von 2D- und 3D-CAD-Daten unterstützt. [57].

3.4 Programmablaufplan

Aus den Anforderungen an das Gesamtsystem wird ein PAP erstellt, welcher qualitativ die Vorgänge des Gesamtsystems beschreibt. Jeder Block ist hier zunächst als Blackbox zu verstehen. Deren Inhalte und Funktionsweisen werden im Rahmen der Konstruktion und der Entwicklung der Software, sowie der Integration des Gesamtsystems in Kapitel 4 erörtert. Die Software wird so strukturiert, dass sie den vollständigen Messprozess des Demonstrators darstellt, wobei auch die hardwareseitigen Prozesse abstrahiert werden sollen.

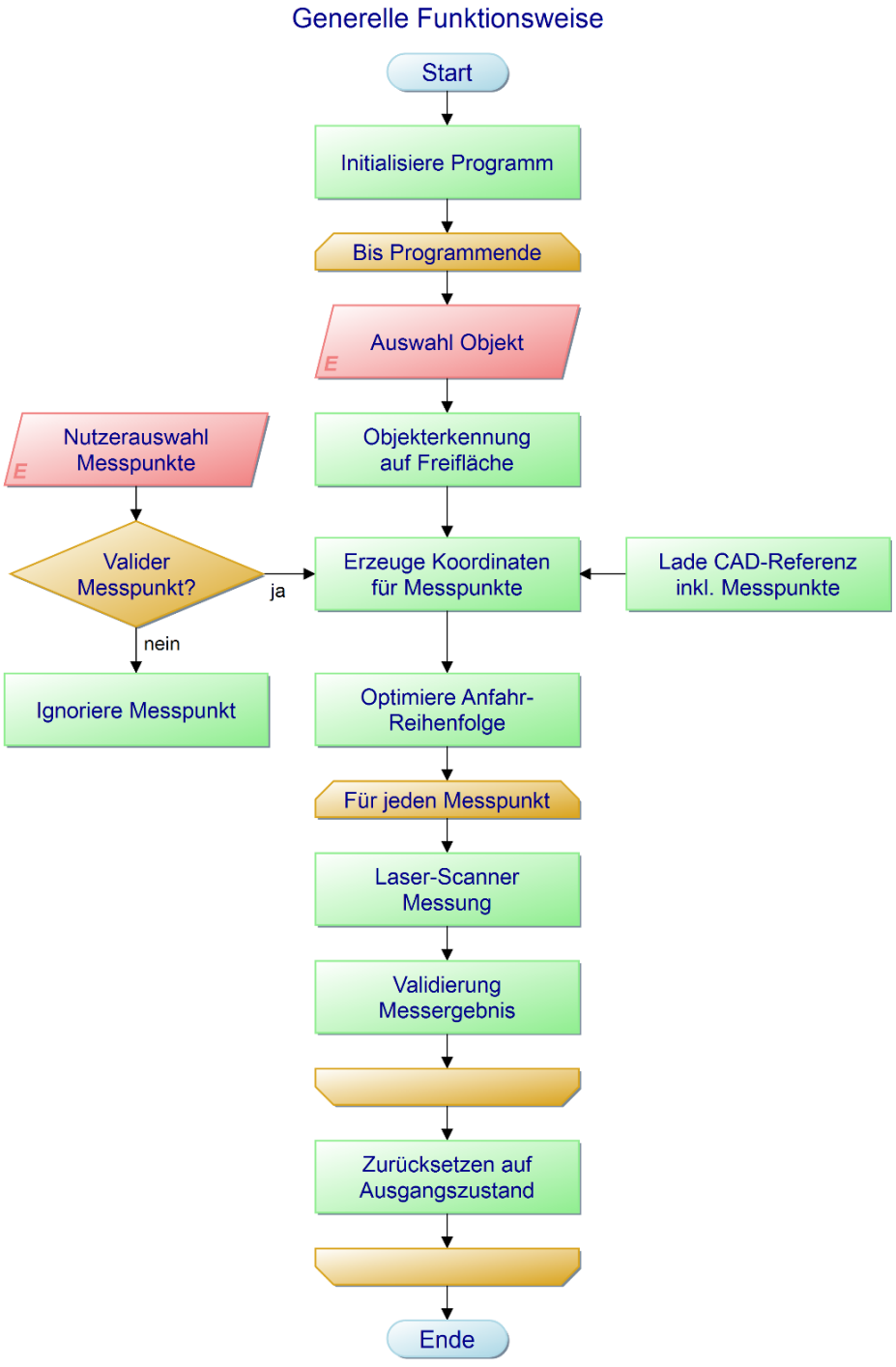


Abbildung 3.3: Schematischer PAP des Gesamtsystems auf Basis der Anforderungen

3.5 Auslegung und Auswahl der Komponenten

Auf Basis des PAP nach Abbildung 3.3 und der funktionalen Anforderungen an das Gesamtsystem werden im Folgenden Technologien und Komponenten ausgewählt, welche insgesamt alle funktionalen Anforderungen an das Gesamtsystem erfüllen können. Der domänenspezifische Entwurf entsprechend dem V-Modell (Abbildung 3.1) erfolgt im Folgenden in der Theorie.

Um allen Anforderungen gerecht zu werden, müssen sich die Einzelkomponenten ohne Einschränkungen zusammenfügen lassen. Dies benötigt die Abstimmung der Spezifikationen der Einzelkomponenten untereinander, um Kompatibilitätsprobleme von vornherein zu vermeiden. Wichtige Dimensionen, welche die Flexibilität in der Auswahl beeinflussen sind an dieser Stelle folgende:

- Geringe Anschaffungskosten für eine möglichst wirtschaftliche Ausführung des Gesamtsystems
- Bereits verfügbare Teile, welche aufgrund vorhandener Erfahrung mit der jeweiligen Entwicklungsumgebung übernommen werden

Im Folgenden werden anhand der Anforderungsliste Abhängigkeiten der Einzelsysteme untereinander bestimmt, sowie konkrete Produkte ausgewählt. Ausgehend von den Randbedingungen werden nacheinander alle Komponenten so gewählt, dass diese den Gesamtanforderungen sowie den Spezifikationen der anderen Komponenten gerecht werden. Die für die Auslegung aller weiteren Komponenten wichtigste Randbedingung ist die Auswahl des Laser-Scanners.

3.5.1 Laser-Scanner

Als Laser-Scanner wird das Modell *Gocator 2510* des Herstellers *LMI Technologies* verwendet. Dieses Modell ist eine bekannte Umgebung bezüglich Hard- und Software, was den Entwicklungs- und Integrationsprozess vereinfacht. Dieser Laser-Scanner besitzt eine räumliche Auflösung von 8 μm in der Bildgebung, was eine ausreichend präzise Messung ermöglicht. Weiterhin ist das Gewicht des Laser-Scanners von 0,65 kg im Vergleich gering, was die Auslegung und die Auswahl der Kinematik vereinfacht. Insgesamt wird der Laser-Scanner als Randbedingung für diesen Entwicklungsprozess genutzt, da von diesem direkt die Genauigkeit der finalen Messung abhängig ist. An dieser Stelle sind Kompromisse in der Auswahl zugunsten der Kompatibilität nicht sinnvoll.

Der *Gocator 2510* ist ein Linienprofilsensor basierend auf der Reflexion eines Linienlasers auf einer Messoberfläche. In einem trapezförmigen, zweidimensionalen Messbereich gemäß Abbildung 3.5 kann ein zweidimensionaler Ausschnitt einer Oberfläche erfasst werden, welcher zunächst als Wolke diskreter Punkte mit Lagedaten für jeden Punkt verarbeitet wird. Diese Lagedaten sind für jeden zweidimensionalen Ausschnitt des gemessenen Profils in Form von Koordinaten auf die Lage im Messfeld des Laser-Scanners bezogen. Anschließend können die Werte dieser Punktwolke numerisch

analysiert werden. So kann beispielsweise der Kantenradius eines Blechbauteils direkt über den Laser-Scanner berechnet und anschließend gesendet werden.

Der Laser-Scanner liefert eine webbrowserbasierte GUI und bietet die Möglichkeit, mit bereits integrierter Software direkt vorverarbeitete Messdaten über unterschiedliche Netzwerkprotokolle zu senden, sowie Befehle von einem Computer entgegenzunehmen. Im Rahmen dieser Anwendung basiert die Kommunikation zwischen Computer und Laser-Scanner auf dem TCP/IP-Protokoll, womit über ein Lokalnetzwerk unverschlüsselter ASCII-Code gesendet wird. Dies macht die Kommunikation mit dem Laser-Scanner leicht implementierbar, sowie effizient, wobei die unverschlüsselte Kommunikation hier kein Sicherheitsrisiko darstellt. Im laufenden Betrieb überblickt der Laser-Scanner ein Linienprofil entsprechend Abbildung 3.4 und Abbildung 3.5.

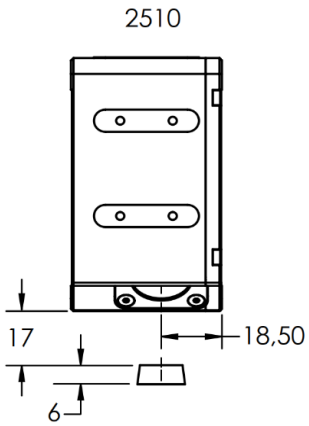


Abbildung 3.4: Scanbereich Gocator 2510 [58]

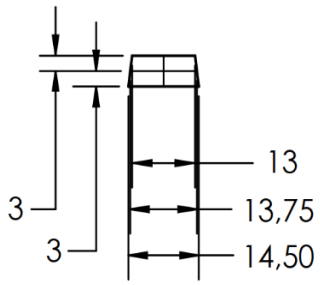


Abbildung 3.5: Scanbereich, Detail [58]

Für einen optimalen Scan der Kante benötigt der Laser-Scanner idealerweise eine Anstellung von 45 Grad zur Achse der Blechkante. Somit können Blechseite- und Oberfläche gleichermaßen abgebildet werden und eine symmetrische Draufsicht auf das Kantenprofil kann gewährleistet werden. Um diese Positionierung zu realisieren, muss ein Adapter zwischen der Kinematik und dem Laser-Scanner konstruiert werden. Das erfasste Bild des Laser-Scanners entspricht dem Kantenausschnitt im Sichtbereich des Laser-Scanners entsprechend Abbildung 3.6.

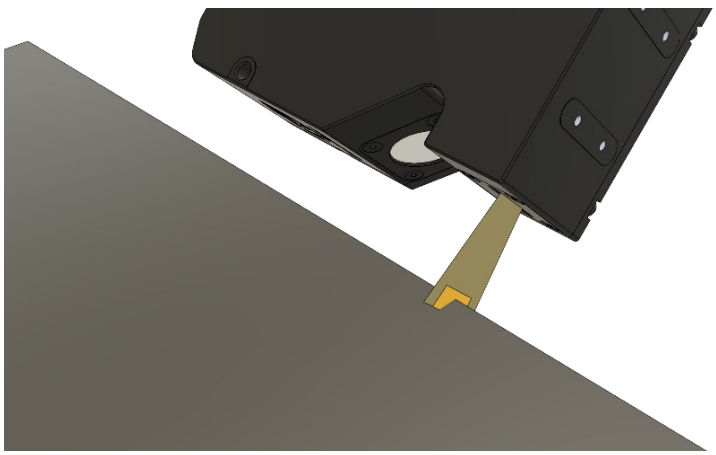


Abbildung 3.6: Anstellung Laser-Scanner an der Bauteilkante mit Sichtbereich (gelb)

3.5.2 Kinematik

Die Aufgabe der Kinematik ist das Manövrieren des Laser-Scanners an beliebige Raumpunkte innerhalb des Messbereichs, sowie der korrekten Ausrichtung des Laserscanners, sodass dieser die Punktmessung normal zur Blechkante vornehmen kann. Dabei sind die Anforderungen, die durch den Laser-Scanner entstehen bei der Produktauswahl zu beachten.

Auswahl des Kinematik-Prinzips

Für die Abdeckung des gesamten dreidimensionalen Messbereichs werden mindestens drei Bewegungsfreiheitsgrade (im folgenden DOF engl. „Degrees of Freedom“) benötigt. Dies kann durch eine beliebige Kombination an translatorischen oder rotatorischen DOF realisiert werden, wobei die geometrischen Anforderungen an die jeweiligen Glieder je nach Ausführung variieren. Zusätzlich muss eine Drehachse normal zur Messebene vorgesehen werden, um die Ausrichtung des Scanners normal zur Bauteil-Kante zu ermöglichen. Insgesamt werden also mindestens 4 DOF benötigt, um den Arbeitsbereich mit allen zusätzlichen Ausrichtungsmöglichkeiten anfahren zu können.

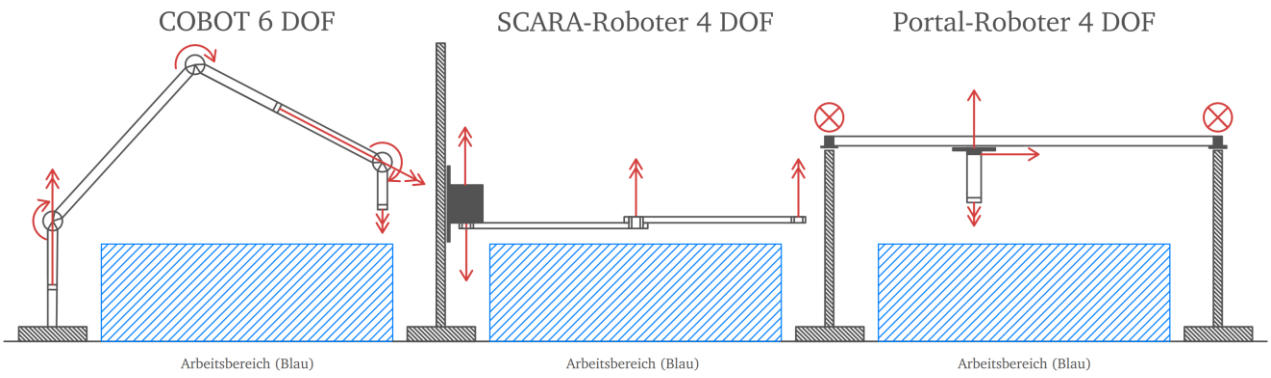


Abbildung 3.7: Mögliche Kinematik-Konzepte 4-6 DOF

Wichtig ist, dass die Kinematik nicht hängend montiert wird, um den Aufbau des Gesamtsystems so niedrig und stabil wie möglich zu gestalten. Somit werden Delta-Kinematiken trotz guter Voraussetzungen bezüglich Schnelligkeit [59] und gut skalierbarem möglichen Arbeitsbereich grundsätzlich ausgeschlossen.

Es existieren viele vorkonfektionierte Kinematik-Konzepte, welche die benötigten vier DOF bereitstellen können und aus diesem Grund theoretisch der Mindestanforderung an die Kinematik gerecht werden. Die Auswahl fokussiert sich auf COBOTs (engl. Collaborative Robot), SCARA-Roboter (engl. Selective Compliance Assembly Robot Arm) und Raumportal-Roboter, welche die benötigten DOF mit unterschiedlichen Prinzipien realisieren (siehe Abbildung 3.7). Für die Auswahl des finalen Kinematik-Prinzips sind die folgenden, nach Priorität geordneten Kriterien relevant, wobei das sinnvollste Kinematik-Prinzip zur individuellen Erfüllung jedes Kriteriums in Tabelle 3-2 erörtert wird.

Tabelle 3-2: Auswahlkriterien & Bewertung der kompatiblen Kinematik-Konzepte

1	Skalierbarkeit der Größe des Arbeitsraums	**Portal-Roboter** – theoretisch beliebig skalierbar → größeres Gerüst vorsehen, bei Arm-Konzepten tendenziell mehr Antriebsleistung in den Achsen nötig
2	Verhältnis aus Bauraum/Manövrierraum gegenüber tatsächlichem Arbeitsraum	**Portal-Roboter** – Rechteckiger/ Quaderförmiger Manövrierraum, gut auf Arbeitsraum anpassbar, Nachteil ist das dauerhafte Gerüst der Kinematik, welches bei den Arm-Konzepten nicht existiert
3	Schnelligkeit u. Präzision der Kinematik	In allen Fällen durch Anpassung der Aktorik fast beliebig skalierbar, für max. DOF **COBOT**
4	Systemintegration u. Komplexität der Lageberechnung / Orientierungsbestimmung	**Portal-Roboter** - Aufgrund kartesischer Koordinatendarstellung ist der Aufbau mit vielen translatorischen DOF sinnvoll, Koordinatentransformation unaufwändig bzw. direkt übertragbar

Hinsichtlich dieser Bewertung wird ein Kinematik-System auf Basis eines Portal-Roboters für den Demonstrator verwendet. Die Skalierbarkeit, sowie die Robustheit und die intuitive Steuerung der Achsen ergeben hier die bestpassende Lösung.

Auslegung und Auswahl des Portal-Roboters

Die Kinematik muss in der Lage sein, das Gewicht des Laser-Scanners inklusive eines Adapters für die Montage des Laser-Scanners zu manövrieren. Zu beachten ist hierbei auch der Einfluss der Anschlusskabel des Scanners. Bei der Auslegung der Maximallast der Kinematik muss dementsprechend eine angemessene Toleranz vorgesehen werden. Der *LMI Gocator 2510* besitzt ein Nettogewicht von 0,65 kg. Die Kinematik soll außerdem eine Wiederholgenauigkeit von 1 mm besitzen, um die Systemanforderungen nach 3.3 zu erfüllen und komfortabel innerhalb der Positionierungs-Toleranz des Laser-Scanners zu liegen.

Wichtig ist weiterhin die Abdeckung des vollständigen Messbereichs von 300 x 400 x 100 mm (b, l, h) den die Kinematik zwangsläufig in Form ihres Arbeitsbereichs abdecken muss. Mit Hinblick auf alle formulierten Anforderungen wird der Raumportal-Roboter *DLE-RG-0001* des Herstellers *IGUS* als Basis für die Kinematik ausgewählt.

Tabelle 3-3: Auszug aus Datenblatt IGUS DLE-RG-0001 [60]

Max. Nutzlast	2,5 kg
Arbeitsraum	500 x 500 x 100 mm (b, l, h)
Positioniergenauigkeit	$\pm$ 0,5 mm
Max. Geschwindigkeit; Beschleunigung	1 $m\,s^{-1}$; 1 $m\,s^{-2}$
Achsen / DOF	3-TTT (3 translatorisch)

Tabelle 3-3 zeigt, dass diese Kinematik beinahe allen Anforderungen genügt, aber nicht über alle notwendigen Bewegungsfreiheitsgrade verfügt. Zusätzlich muss eine rotatorische Achse zwischen Laser-Scanner und Kinematik eingebracht werden, um die benötigten DOF zu erhalten. Zusätzlich sind der Verbindungsadapter von Scanner und Kinematik, sowie der Einfluss der Kabel noch nicht berücksichtigt. Die resultierende konstruktive Anforderung an Adapter, Rotationsachse und Laser-Scanner sieht ein gesamtes Maximalgewicht von 2,5 kg, abzüglich angemessener Toleranz vor.

Implementierung der zusätzlichen Drehachse

Der Raumportal-Roboter *DLE-RG-0001* ist mit einem Schrittmotor pro Achse (insgesamt drei) vorkonfektioniert. Für eine einheitliche Implementierung wird für die zusätzliche rotatorische Achse ebenso ein Schrittmotor vorgesehen. Diese Achse wird entsprechend Abbildung 3.7 in den bestehenden Portalroboter integriert. Der verwendete Schrittmotor soll so ausgelegt sein, dass eine volle Umdrehung der rotatorischen Achse mit hinreichender Sicherheit innerhalb maximal einer Sekunde sichergestellt ist. Eine hinreichende Dynamik des Systems ist für ein zeitoptimiertes Arbeiten des Demonstrators essenziell.

Wichtig für die Auswahl des Motors ist, dass dessen Gewicht inklusive dem Laser-Scanner und einer Adapterkonstruktion unter der Maximalbelastung des Portalroboters liegt. Das qualitative Ziel der Adapterkonstruktion zwischen Motor und Scanner ist das Ausrichten des Laser-Scanners, sodass sich der anzufahrende Kantenpunkt im Zentrum der Drehachse befindet. So wird vermieden, dass der Winkel der Bauteil-Kante auf die Einstellung der translatorischen Achsen des Roboters Einfluss nimmt.

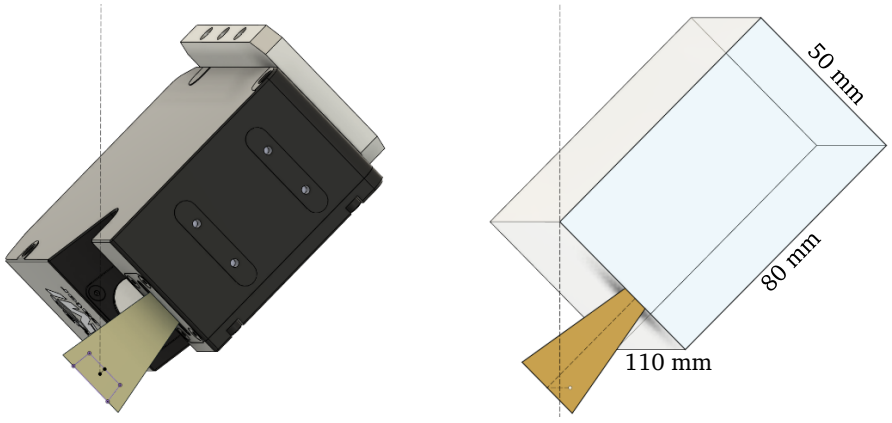


Abbildung 3.8: Ausrichtung Laser-Scanner mit Drehachse, sowie dessen vereinfachte Darstellung

Für das Massenträgheitsmoment der Konstruktion wird vereinfachend nur der Laser-Scanner in dessen Montageposition betrachtet. Zusätzlich wird dieser als idealer Quader mit den jeweiligen maximalen Maßen bezüglich Höhe, Breite und Länge gemäß [58] betrachtet (siehe Abbildung 3.8). Dies führt zu einer konservativeren Bewertung, da effektiv mit einem ausgedehnteren Körper gerechnet wird, und wird daher zur Berechnung verwendet. Dämpfung wird zunächst vernachlässigt.

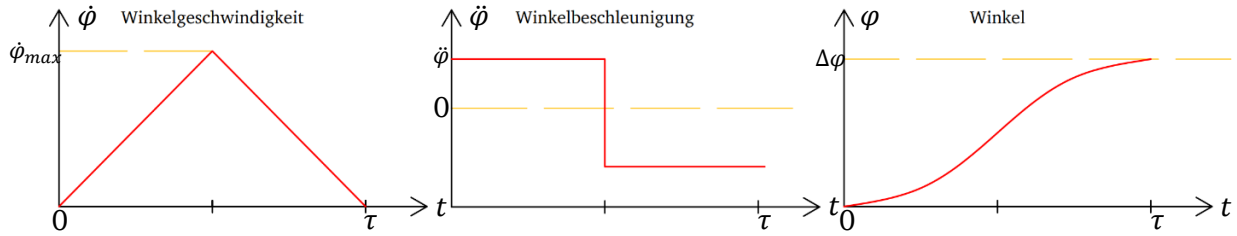


Abbildung 3.9: Beispielhafte Verläufe der Motordynamik (v.l.n.r. Geschwindigkeit, Beschleunigung, Winkel)

Für die Berechnung der Stellzeiten des Motors, sowie dessen Mindestdrehmoment wird von einem dreiecksförmigen Verlauf der Winkelgeschwindigkeit ausgegangen, um die Motorbeschleunigung konstant zu halten (siehe Abbildung 3.9). Für die Betrachtung der Dynamik werden nicht-konservative Kräfte vernachlässigt, wodurch sich für die Systemdynamik Gleichung 3.2 ergibt.

$$M_{\text{Motor}} = \theta_{\text{ges}}\,\ddot{\varphi} \tag{3.2}$$

Hier ist θ_{ges} das vollständige am Motor anliegende Trägheitsmoment in Richtung der Drehachse. Durch einfaches Integrieren (Abbildung 3.9 Rechts), sowie einfaches Differenzieren (Abbildung 3.9 Mitte) ergibt sich aus dem Verlauf der Winkelgeschwindigkeit ein Zusammenhang zwischen zurückgelegtem Winkel und der konstanten Winkelbeschleunigung, sowie der Zeit zum Verstellen der Rotation um jenen Winkel (siehe Gleichung 3.3).

$$\Delta\varphi = \frac{1}{2}\,\dot{\varphi}_{\text{max}}\,\tau \;\text{ für }\; \dot{\varphi}(0) = 0$$

$$\Delta\varphi \;=\; \ddot{\varphi}\,\frac{\tau^2}{4} \;=\; \frac{M_{\text{Motor}}}{\theta_{\text{ges}}}\,\frac{\tau^2}{4} \;\Rightarrow\; \tau \;=\; 2\cdot\sqrt{\frac{\Delta\varphi\cdot\theta_{\text{ges}}}{M_{\text{Motor}}}} \tag{3.3}$$

Zur Einhaltung der Anforderung der Umstellung des Drehwinkels $\Delta\varphi = 2\pi$ bei $\tau \leq 1\,s$ ergibt sich für θ_{ges} Ungleichung 3.4, nach welcher sich das notwendige Antriebsmoment des Motors proportional zum gesamten am Motor anliegenden Trägheitsmoment verhält.

$$\theta_{\text{ges}} \leq \frac{M_{\text{Motor}}}{8\pi}\cdot(1\,s)^2 \tag{3.4}$$

Für die Masse des Laser-Scanners von 0,65 kg und mit der Annahme einer homogenen Masseverteilung ergibt sich für die vereinfachte Scanner-Geometrie und Drehachse entsprechend Abbildung 3.8 $\theta_{\text{ges}} = 4{,}25e-3\,\text{kg}\,\text{m}^2$ [10]. Nach Gleichung 3.4 muss daher der Motor mindestens ein Drehmoment von 10,7 N cm besitzen, um exakt der Mindestanforderung gerecht zu werden. Auf Basis dieser Anforderungen wird als Motor der zusätzlichen Drehachse der Kinematik der Schrittmotor *17HS15-1504-ME1K*, sowie das Planetengetriebe EG17-G5 von *Stepperonline* ausgewählt. Das Getriebe wird vorgesehen, um die Motorwelle axial zu entlasten, sowie um die Drehzahl zu reduzieren. Weiterhin erhöht sich so das Antriebsmoment und die Präzision der Schritte. Tabelle 3-4 beschreibt die kritischen Parameter des Gesamtsystems aus Getriebe und Motor.

[10] Werte geliefert durch 3D-CAD-Programm *Autodesk Fusion 360*

Tabelle 3-4: Auszug aus Datenblatt 17HS15-1684S-PG5 Motor inkl. EG17-G5 Getriebe [61–63]

Gewicht	0,59 kg (0,3 + 0,29 kg)
Schrittweite	0.36 Grad
Antriebsmoment	200 N cm (60 rpm)
Haltemoment	240 N cm
Max. axiale Belastung	100 N
Bauform	Closed-Loop (mit Drehgeber)

Dieser Motor ist aufgrund seines niedrigen Gewichts und seines verhältnismäßig hohen Drehmoments für diese Anwendung gut geeignet. Der integrierte Drehgeber ermöglicht eine genaue Kontrolle über die Motorposition. Eine volle Umdrehung wäre bei maximaler Motorleistung entsprechend Gleichung 3.3 und Tabelle 3-4 theoretisch innerhalb von 0,24 s möglich. Somit ist der Motor auch entsprechend robust gegenüber einem signifikanten Mehrgewicht über den Laser-Scanner hinaus. Bei niedrigen Drehzahlen kann so ein Überschreiten des Haltemoments[11] mit großer Sicherheit vermieden werden. Ebenso ist die Genauigkeit von 0,36 Grad der Schritte in jedem Falle ausreichend präzise, da der Messalgorithmus des Laser-Scanners robust gegenüber kleinen Ausrichtungsfehlern ist. Zu Beginn des Betriebs des Demonstrators muss der Motor manuell kalibriert werden, sodass ein fester Startpunkt für den Drehgeber erzeugt wird. Durch die Closed-Loop-Konstruktion dieses Schrittmotors lassen sich alle vier Achs-Aktoren der Kinematik auf dieselbe Art und Weise steuern und regeln.

3.5.3 Steuerung der Kinematik (SPS)

Zur Steuerung und Regelung der Bewegungsabläufe der Kinematik wird eine Speicherprogrammierbare Steuerung (im Folgenden SPS) verwendet. Die technischen Anforderungen an diese werden durch die Freiheitsgrade der Kinematik bestimmt. Die SPS muss dementsprechend vier Achsen der Kinematik synchron steuern können, welche alle als Schrittmotoren ausgeführt sind. Weiterhin muss die SPS die Daten der Messpunkte über Netzwerkprotokolle empfangen können, sowie Statusmeldungen an den zentralen Computer übermitteln, um die Verbindung aller Komponenten über ein lokales Netzwerk zu ermöglichen.

Um alle Anforderungen zu realisieren, wird ein SPS-System zusammengestellt, welches sich aus verschiedenen Modulen zusammensetzt. Die Zentralbaugruppe des SPS-Systems ist eine CPU, welche Schnittstellen verwaltet und das Steuerungs-Programm ausführt. Um die Drehgeber der Schrittmotoren der Kinematik anzubinden, wird ein Encoder-Auswertungs-Modul vorgesehen. Für weitere

[11] Dies würde das Überspringen eines Motorschrittes bedeuten. Durch den Drehgeber können diese Fehler zwar ausgeregelt werden, Springen sollte dennoch vermieden werden.

flexible Eingangs- und Ausgangsanbindungen werden zusätzlich digitale IO-Module eingesetzt. Für die Schrittmotoren der Kinematik wird jeweils ein Steuerungsmodul vorgesehen, welche über ein Master-Steuerungsmodul an die CPU angebunden werden können. Für die Module des SPS-Systems werden die Komponenten entsprechend Tabelle 3-5 ausgewählt.

Tabelle 3-5: Liste der Komponenten des SPS-Systems

Zentralbaugruppe / CPU	Siemens SIMATIC S7-1500T
Schrittmotor-Mastersteuerung	Siemens PTO-4
Encoder-Auswertung	Siemens TM PosInput 2 (x2)
Digital IO	Siemens DI 16xDC 24V & DQ 16xDC 24V
Schrittmotor-Achssteuerung	Igus drylin D8 (x2) für horizontale Achsen Igus drylin D7 für vertikale Achse Stepperonline DM542T (V4.0) für rotatorische Achse

Über die Python-Bibliothek *Snap7* kann ein Computer mit der Siemens 1511T SPS kommunizieren. Die Kommunikation basiert auf TCP/IP über Ethernet. In dieser Konfiguration ist es möglich, Messpunkt-Koordinaten von einem Computer auf die Kinematik zu übertragen. Weiterhin kann durch einen festgelegten Schrittverlauf direkt das benötigte Antriebsmoment beeinflusst werden, um den Belastungsgrenzen der Schrittmotoren der Kinematik entsprechend 3.5.2 gerecht zu werden. Ebenso können von der verwendeten SPS die Achsen parallel gesteuert werden.

Die einzelnen Module der SPS werden auf Hutschienen in einem Schaltschrank montiert. Innerhalb dieses Schaltschrankes wird außerdem ein Netzwerkswitch untergebracht, an welchem der Computer mit der SPS, dem Laser-Scanner und dem Kamera-System verbunden wird.

3.5.4 Bildgebung / Objekterkennung

Die Aufgabe des Systems zu Objekterkennung ist das Bereitstellen einer Rohaufnahme des Messbereichs zur Feststellung der Lage eines freiliegenden Blechbauteils. Diese Rohdaten sollen eine entsprechend hohe Qualität aufweisen, um Fehler durch Ungenauigkeiten in diesem Schritt zu minimieren. Dies muss gewährleistet sein, da die Genauigkeit aller nachfolgenden Prozesse entsprechend des PAP nach 3.4 von der initialen Bildgebung abhängig ist.

Zusätzlich sollte der Prozess gemäß den Gesamtanforderungen leicht skalierbar sein und in der Praxis möglichst wenig Zeit in Anspruch nehmen. Weiterhin soll das System zur Bildgebung nicht die Kinematik beanspruchen. Dies ist aus den folgenden Gründen anzustreben:

- Einsparung der Manövrierzeit der Kinematik zwischen Prozess zur Bildgebung
- Eliminierung der Ungenauigkeit in der Positionierung der Kinematik
- Kinematik mit niedrigerer Leistung/ maximaler Nutzlast ist verwendbar (hier 2,5 kg)

Die Objekterkennung erfolgt dementsprechend durch einen stillstehenden Sensor. Aufgrund des niedrigen Preises, der flexiblen Ausführung und der guten Skalierbarkeit wird eine CMOS-Kamera für die Bildgebung genutzt. Diese Methode ist sehr gut skalierbar, da die Auflösung der CMOS-Sensoren flexibel anpassbar und auf den jeweiligen Einsatz anpassbar ist. Alternativ lassen sich mit niedrigem Aufwand über Bildverarbeitung die Bildfläche, sowie die Auflösung nahezu beliebig hochskalieren, wenn ein Array aus Kameras verwendet wird. Der geforderte Arbeitsbereich von 300 mm x 400 mm (b, l) des Demonstrators wird bei diesem Demonstrator mit einer einzelnen Kamera abgedeckt. Diese soll zentriert über dem Arbeitsbereich angebracht werden.

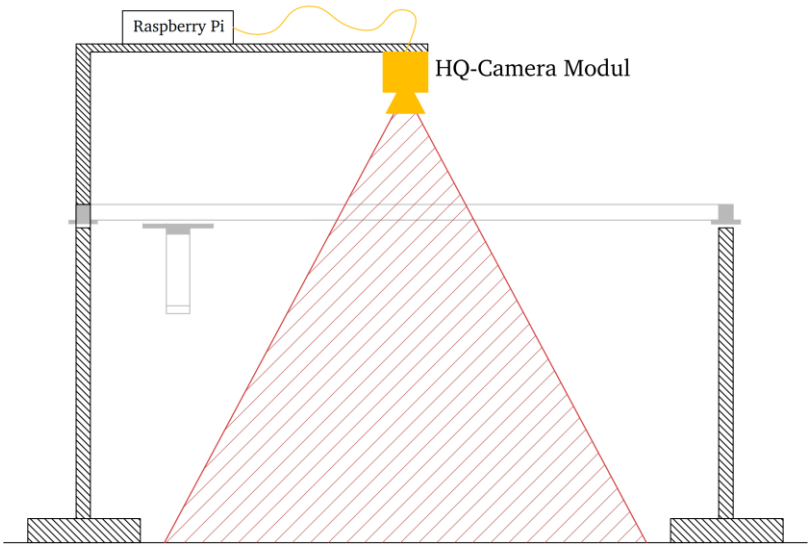


Abbildung 3.10: Aufbau des Kamerasystems mit Blickfeld (rot) u. Kinematik

Auslegung und Auswahl des Kamera-Moduls

Im Rahmen der Bildverarbeitung wird ausgehend von der Lage des Blechbauteils auf der Messfläche das CAD-Referenzmodell an die richtige Stelle des Bildes projiziert. Da dieses später als Vektorgrafik vorliegt, lassen sich Punktkoordinaten nach der Objekterfassung verlustfrei auf die erfasste Geometrie projizieren. Essenziell ist, dass die Lage des zu vermessenden Blechbauteils hinreichend genau erfasst werden kann. Dies erfordert eine vernachlässigbar kleine Ungenauigkeit bei der Bildgebung, um den Gesamtfehler des gesamten Prozesses so wenig wie möglich zu beeinflussen[12]. Anzustreben ist hier eine Ungenauigkeit, welche sich maximal in derselben Größenordnung der Ungenauigkeit der Kinematik ($\pm 0{,}5\ mm$) befindet. Die Genauigkeit des Bildgebungsprozesses lässt sich aus der Pixeldichte des Kamerachips errechnen. In der Praxis wird die Annahme getroffen, dass sich die einzelnen Pixel ideal diskret verhalten, sodass eine proportionale Pixelauflösung nach Gleichung 3.5 berechnet werden kann. Für das Maß der Ungenauigkeit in mm wird im Allgemeinen eine Toleranz

[12] Diese Annahme basiert auf der Fehlerfortpflanzung der Bildgebungsfehler, sowie der Positionierungsfehler der Kinematik. Aufgrund der Unabhängigkeit der Fehlergrößen können diese getrennt voneinander skaliert werden. [64].

von ±5 Pixel empfohlen [65]. Die gesamte Ungenauigkeit des Verfahrens bestimmt sich nach Gleichung 3.6.

$$\frac{\text{Bildabmaß [mm]}}{\text{Pixelanzahl [Pixel]}} = A\text{uflösung}\left[\frac{\text{mm}}{\text{Pixel}}\right] \tag{3.5}$$

$$\text{Auflösung}\left[\frac{\text{mm}}{\text{Pixel}}\right] \cdot 5\ \text{Px} = \text{Maßtoleranz [mm]} \tag{3.6}$$

Für die Umsetzung der Objekterkennung wird ein sichtbarer Rahmen um den Arbeitsbereich vorgesehen. Daher wird das Sichtfeld der Kamera auf mindestens 10 mm über jede Seite des Arbeitsbereiches festgelegt. Insgesamt beläuft sich dies auf einen Bildbereich von mindestens 320 x 420 mm (h, v). Mit diesen Abmaßen und der maximalen Maßtoleranz von ±0,5 mm ergeben sich für die Größenordnungen der anzustrebenden Pixelanzahlen des Bildbereichs nach den Gleichungen 3.3 und 3.4 3600 x 4200 Pixel (h, v).

Für die Auswahl der Komponenten des Kamera-Systems des Demonstrators ist demnach die maximale Auflösung des Systems der wichtigste Entscheidungsparameter. Wichtig für die weitere Entwicklung dieser Technologie ist, dass bei jeglichen Ungenauigkeiten, welche aus dieser beispielhaften Implementierung entstehen, ein Kamera-System mit höherer Auflösung Abhilfe schaffen kann.

Auf Basis dieser Anforderungen wird das Kameramodul *High Quality Camera* des Herstellers *Raspberry Pi* als Sensor für die Objekterkennung ausgewählt. Als Schnittstelle zwischen dem Computer und dem Kameramodul wird zudem ein *Raspberry Pi 4 B* als Schnittstelle vorgesehen[13]. Damit wird effektiv eine herkömmliche Industriekamera mit Netzwerkanbindung ersetzt. Nicht öffentlich zugängliche Standards wie *GigE Vision* [66] müssen bei der Verwendung des *Raspberry Pi*-basierten Systems ebenfalls nicht genutzt werden, die Implementierung ist Open-Source. Das Kameramodul sendet hier rohe RGB-Bilddaten mit einem Dynamikumfang von bis zu 12 Bit an den *Raspberry Pi*, welcher dieses Bild zunächst erzeugt und dann dem Computer per Netzwerkkommunikation zur Verfügung stellt.

Tabelle 3-6: Auszug aus Datenblatt Raspberry Pi HQ Camera, inkl. Ergänzung [67]

Auflösung	12,3 Megapixel (4056 x 3040 Pixel)
Sensor	Sony IMX477R
Anbindung Objektiv	M12-/ S-Mount
Chipgröße	7,9 mm (1/2,3 “) Diagonale (6,287 x 4,711 mm)
Pixel-Größe	1,55 x 1,55 µm
Anbindung Raspberry Pi – Kamera	RAW 12/10/8, COMP 8 (CSI-2 Anschluss)
Anbindung Raspberry Pi – PC	Beliebiges Netzwerkprotokoll (siehe 3.3)

[13] Im Allgemeinen verwendbar sind Einplatinencomputer mit MIPI/ CSI-2 Schnittstelle zur Anbindung von Kamerachips, welche Rohdaten entsprechend Tabelle 3-6 senden.

Die beinahe exakte Auslegung bezüglich der Pixelanzahl, sowie die open-source Implementierung der Software führen zur Wahl dieses Kamera-Moduls. Für die Bestimmung der Bildgebungsungenauigkeit werden alle Annahmen konservativ zugunsten der Allgemeingültigkeit getroffen [65]. Das fertige Bild wird allerdings einen hohen Kontrast besitzen, keine Fehler durch Bewegungsunschärfe besitzen und benötigt beinahe keine Tiefenschärfe. Daher wird die leichte Unterauslegung der gewählten Kamera bezüglich der Auflösung bewusst toleriert.

Auslegung und Auswahl des Objektivs

Zuzüglich des Kamera-Moduls muss ein passendes Objektiv ausgewählt werden. Dieses muss im M12- oder S-Mount Format ausgeführt sein (siehe Tabelle 3-6). Das Objektiv sollte weiterhin die Auflösung des Kamerachips unterstützen, sowie auf die Chipgröße angepasst sein. So werden sowohl Auflösung des Objektivs als auch des Chips optimal ausgenutzt. Perspektivische Verzerrungen und Verzeichnungen[14] in der Abbildung des Blechbauteils auf der Messoberfläche sollen im Prozess der Bildgebung minimiert werden. Dies wird im Idealfall durch ein beidseitig telezentrisches[15] Objektiv erreicht. Dieses wäre in der Lage, die Bilder der Messfläche ohne perspektivische Verzerrung oder Verzeichnungen abzubilden. Dazu ist aufgrund der Funktionsweise dieser Objektive allerdings eine Eingangslinse der Größe des abzubildenden Bereichs notwendig. Dies stellt aufgrund der Größe des Bildbereichs von mindestens 320 x 420 mm ein offensichtliches Ausschlusskriterium dar, da das Kameramodul starr montiert sein soll.

Ein ideal telezentrisches Objektiv stellt den Grenzwert Null bezüglich des Bildwinkels[16] dar. Perspektivische Verzerrungen können durch kleine Bildwinkel allerdings zumindest verringert werden. Der Bildwinkel eines Kamera-Systems aus Bild-Sensor und nicht-telezentrischem Objektiv errechnet sich nach Gleichung 3.7 [69].

$$\alpha_0 := 2\, arctan\left(\frac{s_{\text{sensor}}}{2\, f}\right) \tag{3.7}$$

Hier ist s_{sensor} das entsprechende Sensorabmaß (Horizontale, Vertikale o. Diagonale) und f die Brennweite des Objektivs. Zur Montage der Kamera außerhalb des Arbeits- und Manövrierraumes der Kinematik sollte ein Abstand des Kamera-Moduls von mindestens 400 mm zur der Messfläche

[14] Siehe 2.3.2 für konkrete Beschreibungen der Fehlertypen und derer Ursachen

[15] Objektivaufbau, bei welchem das perspektivische Zentrum (Fluchtpunkt) im Unendlichen liegt, da alle in das Objektiv einfallenden Lichtstrahlen parallel verlaufen (Bildwinkel 0°). Dadurch entfallen theoretisch alle perspektivischen Fehler. Der *telezentrischen Perspektive* entgegen steht die *entozentrische Perspektive* [68].

[16] Der Bildwinkel wird beschrieben durch den Winkel zwischen zwei Lichtstrahlen, welche durch den Linsenmittelpunkt des Objektivs auf den Kamera-Sensor treffen. Dieser wird bezüglich des Sensors horizontal, vertikal oder diagonal angegeben [69].

vorgesehen werden. Die Mindestmontagehöhe wird aus dem Bildwinkel und der mindesten Bildfläche gemäß Gleichung 3.8 bestimmt.

$$h_{\min} = \frac{s_{\text{mess}}}{2\,tan\left(\frac{\alpha_0}{2}\right)} \equiv \frac{s_{\text{mess}}\,f}{s_{\text{sensor}}} \Rightarrow f_{\min} = \frac{h_{\min}\,s_{\text{sensor}}}{s_{\text{mess}}} \tag{3.8}$$

Hier ist s_{mess} das jeweilige Abmaß des Messbereichs und α_0 der Bildwinkel entsprechend Gleichung 3.7. Für die Sensorgröße des Kamera-Sensors nach Tabelle 3-6, des geforderten Bildbereichs von 320 x 420 mm (h, v) und der Mindestmontagehöhe der Kamera von 400 mm ergibt sich eine Mindestbrennweite für das Objektiv, entsprechend der horizontalen und vertikalen Sensorabmaßen. Diese belaufen sich auf $f_{\text{min,horizontal}} = 5{,}89\,mm$; $f_{\text{min,vertikal}} = 5{,}99\,mm$. Das verwendete Objektiv sollte dementsprechend mindestens eine Brennweite von 6 mm besitzen.

Tabelle 3-7: Auszug aus Technischen Daten Raspberry Pi 8mm Objektiv M12-Mount [70]

Brennweite	8 mm
Befestigung	M12-/ S-Mount
Auflösung	12 Megapixel
Blende	F2
Abbildungsgröße	1/2,3“

Auf Basis dieser Forderungen wird das *8mm Objektiv, M12-Mount* des Herstellers *Raspberry Pi* ausgewählt. Es liefert eine entozentrische Perspektive [68] und ist gemäß allen Parametern ideal auf den Kamera-Sensor abgestimmt.

Mit der Brennweite des Objektivs nach Tabelle 3-7 und Gleichung 3.7 ergeben sich für den horizontalen und vertikalen Bildwinkel jeweils $\alpha_{0,\text{horizontal}} = 42{,}9\,°$; $\alpha_{0,\text{vertikal}} = 32{,}8\,°$. Für die minimale Montagehöhe ergeben sich nach Gleichung 3.8 und den horizontalen beziehungsweise vertikalen Bildmaßen $h_{\min} = 543{,}4$ mm beziehungsweise $h_{\min} = 534{,}4$ mm. Das Kamerasystem wird, wie in Abbildung 3.10 skizziert, mit minimaler Toleranz auf 550 mm Höhe zentriert über der Messoberfläche montiert. Die Ausrichtung erfolgt an der Eingangslinse des Kamera-Objektivs.

3.6 Zusammenfassung der theoretischen Entwicklung

Im Folgenden wird auf Basis der vorangegangenen Komponentenauslegung ein digitaler Prototyp des Gesamtsystems konstruiert. Im Rahmen der Komponentenauslegung und -auswahl in 3.5 wurde bereits die Kompatibilität der Komponenten sichergestellt. Für den CAD-Prototyp werden primär die Bauräume der einzelnen Komponenten betrachtet und ein Demonstrator-Gerüst gestaltet, auf welchem alle Komponenten montiert werden können.

3.6.1 Liste verwendeter Komponenten

In Tabelle 3-8 sind die im Rahmen dieses Kapitels vorausgelegten und ausgewählten Komponenten aufgeführt. Dies ist keine vollständige Stückliste und dient zur Veranschaulichung, sowie zum Zusammenfassen der Ergebnisse dieses Kapitels.

Tabelle 3-8: Stückliste der vorausgelegten Komponenten

Kap.	Baugruppe	Produkt	Hersteller	Verwendung
3.3.1	Laser-Scanner	Gocator 2510	LMI-Technologies	Laser-Scanner
3.3.2	Kinematik	DLE-RG-0001	igus	Raumportal-Roboter (3-TTT)
		17HS15-1504-ME1K	Stepperonline	Schrittmotor (Drehachse)
		EG17-G5		Getriebe (Drehachse)
3.3.3	SPS	1511T	Siemens	CPU (Mehrachsensteuerung)
		PTO-4		Motor-Mastersteuerung
		TM PosInput 2		Encoder-Auswertung (Achsen)
		DI 16xDC 24V & DQ 16xDC 24V		Digital I/O
		drylin D8	igus	Treiber für horizontale Achsen (x2)
		drylin D7		Treiber für vertikale Achse
		DM542T (V4.0)	Stepperonline	Treiber für rotatorische Achse
3.3.4	Bildgebung	HQ Camera Modul	Raspberry Pi	Kamera-Chip (CMOS)
		8mm M12-Mount		Objektiv

3.6.2 CAD-Modell des Prototyps

Für das Gerüst des Demonstrators wird eine Konstruktion aus Aluminium-Profilstangen mit quadratischem Querschnitt in der Dimension 40x40 mm, sowie eine schwarze Kunststoffplatte mit ausgewiesenem Messbereich (Referenzrahmen) als Messoberfläche vorgesehen. Die Stangen werden mit Schnellspannverschlüssen montiert, sodass keine Montagewinkel verwendet werden müssen. Nicht konkret funktionsrelevant und daher nicht genau spezifiziert ist außerdem der Schaltschrank in Abbildung 3.11. Aus dem CAD-Modell muss keine finale Konstruktion abgeleitet werden, es dient hier zur Visualisierung des Aufbaus und der Bauraumgestaltung des Demonstrators. Ebenso vernachlässigt werden zur Übersichtlichkeit hier sämtliche Kabel und Kabelführungen, sowie die Komponenten der SPS, welche im Schaltschrank montiert werden.

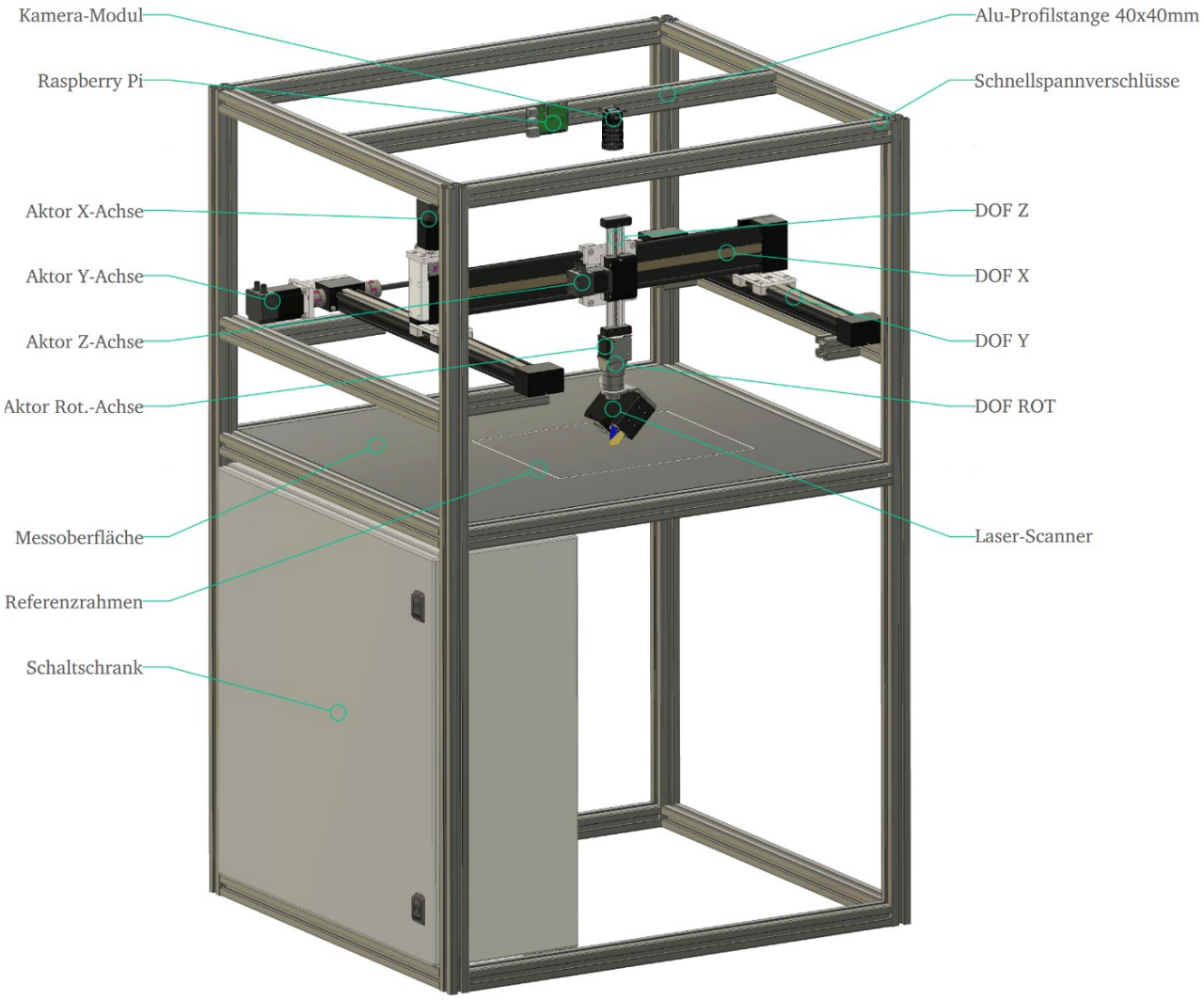


Abbildung 3.11: Gesamtansicht CAD-Prototyp mit ausgewiesenen Komponenten

Der Demonstrator ist vollständig durch Aluminium-Profilstangen eingerahmt. So kann dieser beispielsweise mit Blechen oder Acrylglas verkleidet werden, ohne nachträgliche Konstruktionen vorzunehmen. Bis zur Höhe der Messfläche soll der Demonstrator perspektivisch mit Blech verkleidet werden, darüber soll Acrylglas einen Einblick in den Mess-Prozess ermöglichen. An der Vorderseite soll eine Tür montiert werden, welche das manuelle Einlegen der Blechteile ermöglicht. Abbildung 3.12 zeigt ein Rendering des ersten Prototyps[17].

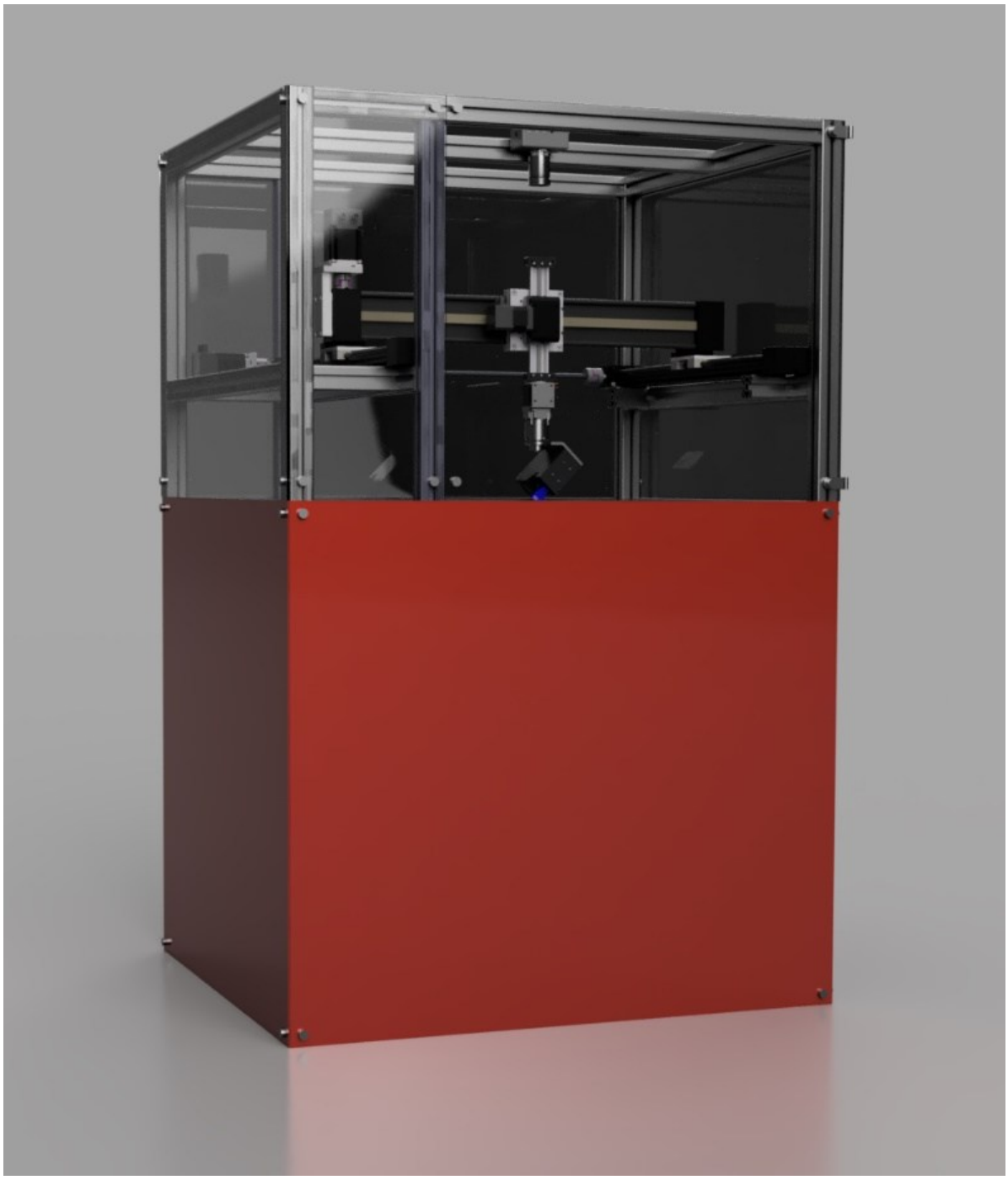


Abbildung 3.12: 3D-Render CAD-Prototyp mit Verkleidung (Tür geschlossen

[17] Die CAD-Modelle des Demonstrators sind im Anhang enthalten

4 Implementierung der Subsysteme / Integration des Gesamtsystems

In diesem Kapitel werden die zuvor ausgelegten Komponenten zu einem Gesamtsystem zusammengefügt. Analog zu 3.5 erfolgt hier der domänenspezifische Entwurf der Einzelsysteme nach dem V-Modell (Abbildung 3.1) in der Praxis. Die Reihenfolge der Beschreibung des Entwicklungsprozesses erfolgt in der Reihenfolge, nach welcher diese im System-PAP nach Abbildung 3.3 relevant werden. Abschließend erfolgt die Integration des Gesamtsystems, wobei der finale Demonstrator konstruiert wird. In Abbildung 4.1 ist der Messprozess in Bezug auf die Subsysteme als Blockschaltbild dargestellt. Die rot markierten Subsysteme werden im Rahmen dieser Arbeit nicht vollständig praktisch implementiert. Hier wird die Kontinuität des Prozesses über Simulationen seitens der SPS überprüft. Dunkel dargestellt ist die Schnittstelle mit Ein- und Ausgängen des Gesamtsystems.

Der Schwerpunkt ist die Entwicklung der Software, da die Funktionalität der einzelnen Komponenten seitens der Hardware hier nicht nachgewiesen werden muss. Die mechanische Funktionalität der Komponenten wurde in der Systemauslegung im Sinne dieser Arbeit hinreichend sichergestellt.

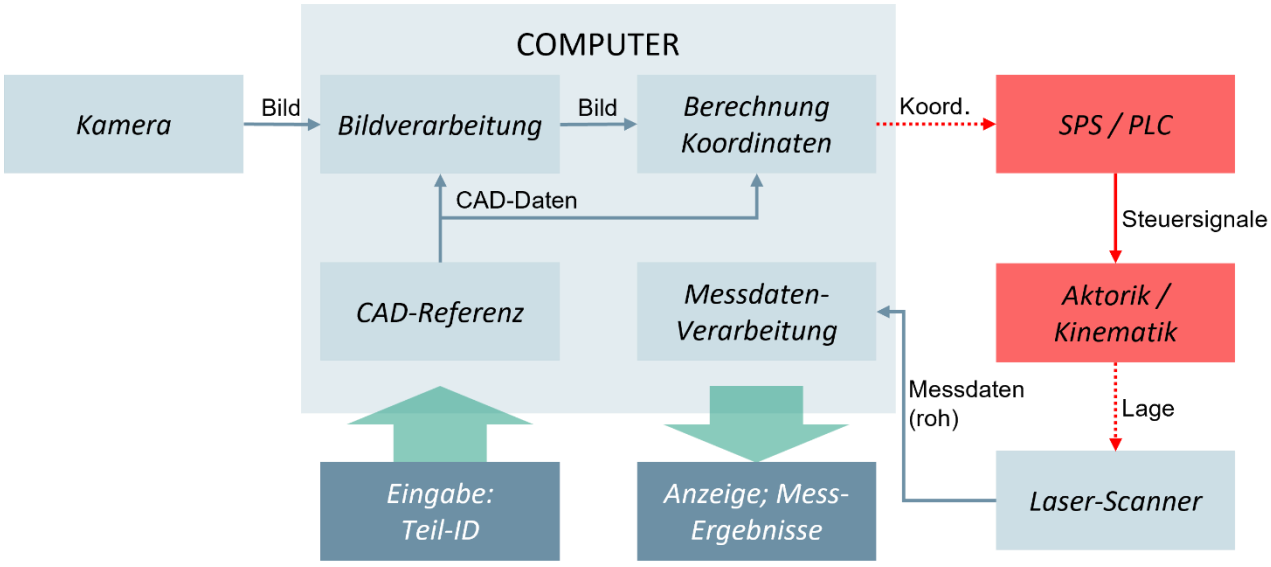


Abbildung 4.1: Blockschaltbild des Gesamtsystems

Aus Abbildung 4.1 gehen die vier aktiven Komponenten hervor, welche im Rahmen des Messprozesses zwecks Datentransfer kommunizieren. Die hierzu verwendeten Protokolle sind in Abbildung 4.2 gelistet, wobei die Kommunikation immer bidirektional zwischen dem jeweiligen Subsystem und dem Host-Computer stattfindet. Die Kommunikation findet entsprechend der Anforderungen über Ethernet auf Basis von geeigneten Kommunikationsprotokollen statt.

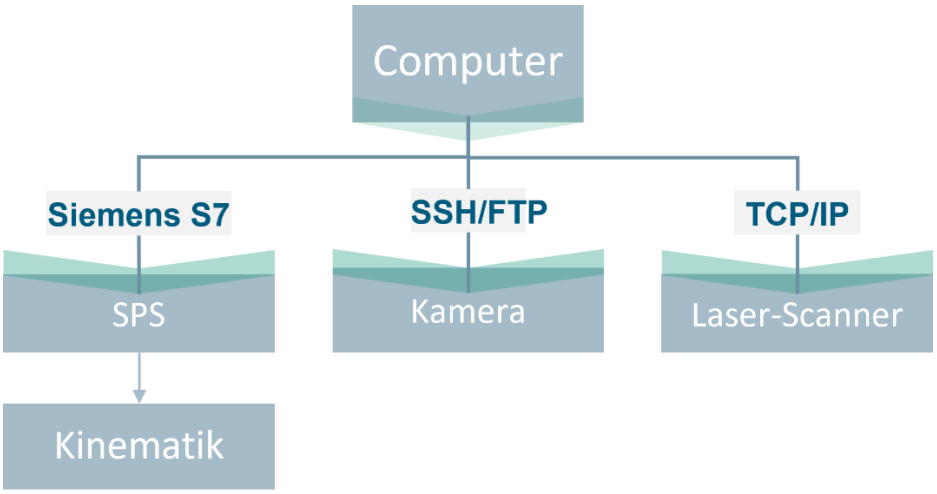


Abbildung 4.2: Implementierung der Netzwerkkommunikation (Übersicht)

Zur Übersicht über den gesamten Ablauf eines Messprozesses am Demonstrator wird zunächst die Struktur der Software beleuchtet. In die Struktur des fertigen Prozessskriptes fließt der System-PAP nach Abbildung 3.3, das System-Blockschaltbild nach Abbildung 4.1, sowie die Netzwerktopologie entsprechend Abbildung 4.2 ein. Dadurch wird ein verständlicher Programmcode realisiert.

4.1 Rahmenbedingungen der Entwicklung & Software-Struktur

Die Software des Demonstrators wird mit Unterstützung verfügbarer Bibliotheken und APIs in Python entwickelt. Hierbei werden die 64-Bit Versionen von Python 3.10 und 3.11 verwendet. Alle verwendeten Computer operieren auf Basis von 64-Bit Microsoft Windows 10 oder Windows 11. Für alle verwendeten Systeme wird im Rahmen dieser Arbeit eine erwartungsgemäße Funktionsweise sichergestellt. Zusätzlich zu Python müssen externe Module (siehe Tabelle 4-1) über den *Python Package Installer* (PIP) installiert werden. Zusätzlich werden Module aus der Standard-Bibliothek in Python verwendet, welche nicht explizit installiert werden müssen und nicht gelistet werden.

Tabelle 4-1: Modulliste für die Software-Implementierung in Python 3.10 mit Version

OpenCV	4.7.0.68	Alle Computer-Vision Anwendungen (C-API)
Imutils	0.5.4	Grundlegende Bild-Bearbeitung
Ezdxf	1.0.2	Import von DXF-Referenzdateien
Matplotlib	3.7.1	Erstellen von Bild-Matrix aus DXF-Referenzdatei
Numpy	1.22.3	Effiziente Rechenoperationen (C-API)
Paramiko	3.1.0	SSH-Client mit SFTP-Kompatibilität
Snap7	1.3.0	Native Kommunikation mit Siemens S7 SPS

Die Software-Programmierung erfolgt paradigmatisch objektorientiert. Die Software ist entsprechend des Klassendiagramms nach Abbildung 4.3 organisiert. Die Klassen vererben in Pfeilrichtung nach unten.

Im Falle einer Mehrfachvererbung dient die Struktur der Aufteilung einzelner Methoden in intuitivere Klassen. Bei einer Programm-Implementierung in einer Sprache, welche dies nicht zulässt, müsste an dieser Stelle eine andere Struktur implementiert werden. Fett gedruckte Klassen werden im Programm explizit instanziiert. Diese abstrahieren unter anderem die Subsysteme des Demonstrators (*Camera, LaserScanner, PLC*), sowie wichtige Programmschritte (*CameraImage, MeasurementPoints, MetaDataParser*). Nicht fett sind Klassen, welche übergeordnete Funktionen vererben und automatisch instanziiert werden.

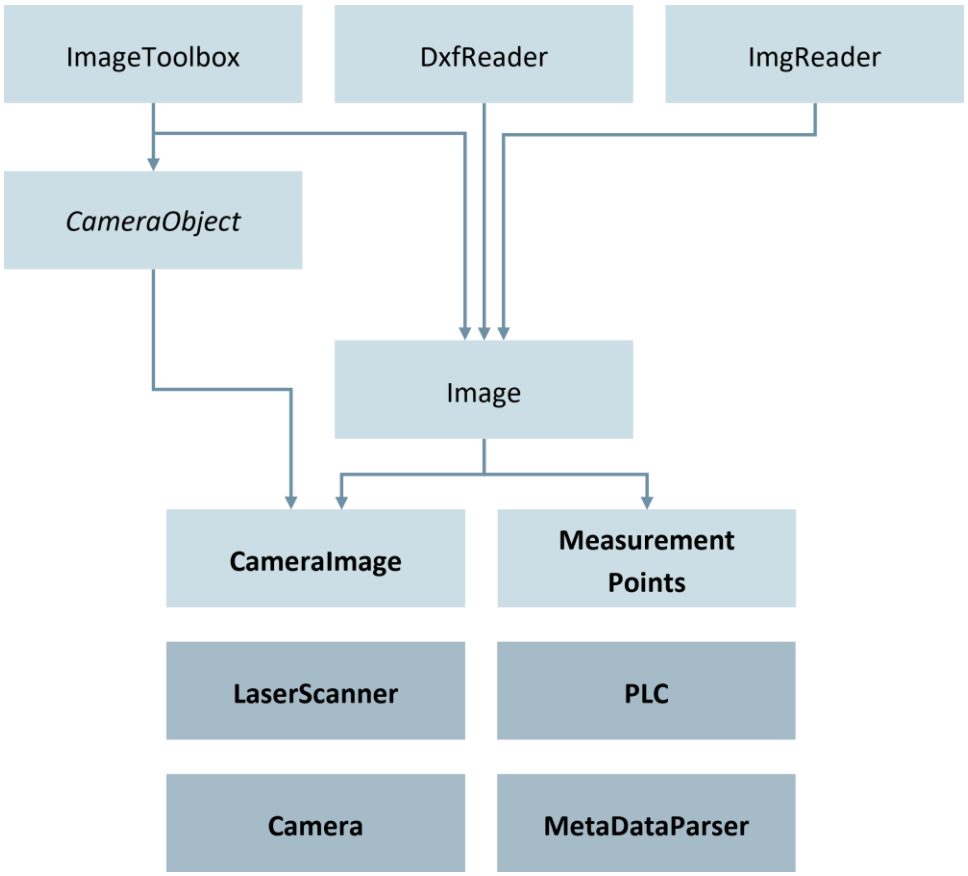


Abbildung 4.3: Klassendiagramm der Softwareumgebung

Die dunkel hinterlegten Klassen funktionieren eigenständig. Diese abstrahieren jeweils die Hardwarekomponenten, beziehungsweise die Metadaten des Systems und ermöglichen die Kommunikation mit dem Computer. Die grundsätzlichen Funktionen der Klassen werden in Tabelle 4-2 beschrieben. Auch die Klassenstruktur selbst vereinfacht eine Abstrahierung der Softwareprozesse im Sinne des System-PAP aus Abbildung 3.3.

Tabelle 4-2: Kurzbeschreibung der Klassen

CameraImage	Klasse für Bild der Kamera als Objekt und stellt alle Funktionen zu dessen Bearbeitung (Import, Vorverarbeitung)
Image	Klasse für Bilder im Allgemeinen, sowohl für Pixelgrafiken (.png) als auch Vektorgrafiken (.dxf) verwendbar
MeasurementPoints	Klasse für Messungs-Objekt, erstellen- und berechnen von Messpunktlage, Berechnung der Fahrtroute der Kinematik, Datenexport etc.
MetaDataParser	Klasse für Objekt zum Einlesen von Metadaten und Systemabhängigkeiten, wird an erster Stelle instanziiert.
Camera	Klasse zur Kommunikation mit dem Raspberry Pi zum Erstellen und Import von Bilddaten über SSH und FTP
LaserScanner	Klasse zur Kommunikation mit dem Laser-Scanner zum Durchführen von Kantenmessungen an Messpunkten über TCP/IP
PLC	Klasse zur Kommunikation mit der SPS, übertragen von Koordinaten und Empfangen von Statusmeldungen
ImageToolbox	Meta-Klasse für grundlegende Bildbearbeitung, anzeigen, speichern, zuschneiden etc. → wird grundsätzlich instanziiert
CameraObject	Meta-Klasse mit Methoden zur Objekt-Erkennung auf einem Bild → wird mit jedem Bild-Objekt instanziiert
ImgReader	Meta-Klasse zum Import von Pixelgrafiken (.png)
DxfReader	Meta-Klasse zum Import von Vektorgrafiken (.dxf)

Metadaten wie Randbedingungen des Messprozesses oder Kalibrierungsdaten werden in strukturierten Dateien gespeichert. Alle Metadaten, welche nicht den Anforderungen entsprechend in DXF-Referenzmodellen enthalten sind, werden nach Bauteil in einer JSON-Datei geordnet. So können wichtige Programmpfade, Namenskonventionen, vorgegebene Messpunkte und geometrische Randbedingungen des Messprozesses abgespeichert und im Messprozess ausgelesen werden (siehe 4.2.1).

Als Proof of Concept wird die Software außerdem in einen Docker-Container ausgelagert. Hierzu wird ein Python 3 Image als Grundlage genutzt, wobei die notwendigen Pakete mit spezifizierten Versionen aus Tabelle 4-1 über PIP installiert werden. Bei dem Betrieb der Software auf dem Container ist die Anzeige von Bild-Dateien in den von OpenCV implementierten Popup-Fenstern nicht mehr möglich. Hierzu wird ein Port spezifiziert, welcher zur Kommunikation über einen Webserver mit dem Container eingesetzt werden kann. Hierzu könnte perspektivisch ein Jupyter-Notebook in einem beliebigen Webbrowser verwendet werden.

4.2 Software-Programmablauf / Funktionsweise der Subsysteme

Ausgehend von der zuvor beschriebenen Software-Architektur und den Ergebnissen der theoretischen Systemanalyse im PAP (siehe Abbildung 3.3) werden nun die Subsysteme in Soft- und/oder Hardware entwickelt. Die Main-Funktion bietet hier einen Überblick über den Programmablauf und ist eine praktische Umsetzung des System-PAP.

Die einzige Eingangsvariable des Programms ist der Name der Referenzdatei des zu vermessenden Bauteils, welche zum Berechnen von Messpunkt-Koordinaten genutzt werden soll. Dies kann entweder als Nutzereingabe erfolgen, oder bei Bedarf durch mehrfaches Aufrufen des Programms mit vorgegebenem Input automatisiert werden. Die Main-Funktion ist in Blöcke aufgeteilt, welche die Kommunikation mit der Kamera, der SPS und dem Laser-Scanner, sowie der Bildverarbeitung, abstrahieren. Aus den Namen der Methoden geht die Funktion im Kontext des Programmes hervor. In den nachfolgenden Abschnitten werden die einzelnen Programmblöcke funktional erörtert. Am Ende jedes Blockes befindet sich eine Log-Methode zum Protokollieren der Messdaten. Diese werden bei Bedarf in der Konsole des ausführenden Terminals, aber bei Aufruf immer in einer fortlaufenden Log-Datei gespeichert. Zur Identifikation einer Messung wird ein Zeitstempel genutzt.

```
def main() -> None:
    ### main program setup-object instantiation
    SETUP:mv.MetaDataParser = mv.MetaDataParser(FILE_NAME_REF="reference25.dxf")

    ### take / import image
    try:
        CAMERA = cm.Camera()
        CAMERA.take_send_img(local_file=SETUP.file_name)
        CAMERA._end_session()
    except ConnectionError:
        print("\033[31;1mCamera not working properly […] \033[0m")

    ### edit image
    IMG = mv.CameraImage(setup_obj=SETUP)
    IMG._allign_to_frame()
    IMG._get_object_data()
    IMG._log_data(verbose=True)

    ### Get / Set Measurement points
    MEASUREMENT = mv.MeasurementPoints(setup_obj=SETUP)
    MEASUREMENT._insert_static_points()
    MEASUREMENT._enter_custom_points()
    MEASUREMENT._export_points(source=IMG)
    MEASUREMENT._log_data(verbose=True)

    ## PLC / Scanner communication & measurement
    try:
        PLC = cm.PLC(csv_path=SETUP.data_export_path)
        SCANNER = cm.LaserScanner(log_path=SETUP.log_path)
        cm.exec_measurement(PLC, SCANNER)
        SCANNER._log_results(verbose=True)
        SCANNER._end_session()
        PLC._end_session()
    except ConnectionError:
        print("\033[31;1mLaser-Scanner or PLC not working properly\033[0m")
```

4.2.1 Umgang mit Meta-Daten und Referenz-Daten

Zu Beginn wird ein Setup-Objekt instanziiert, welches die Metadaten, wie vorgegebene Messpunkte, Dateinamenskonventionen oder Speicherpfade ausliest und in das Programm einführt. Dies wird in der ersten Zeile der Main-Funktion ausgeführt:

```
SETUP = mv.MetaDataParser(FILE_NAME_REF="reference<ID>.dxf")
```

Diese Daten sind strukturiert in der JSON-Referenz-Datei gespeichert. Da die Metadaten abhängig von dem zu vermessenden Blechbauteil sind, wird als Argument zur Bestimmung des zugehörigen Metadatensatzes der Dateiname der Referenzdatei genutzt. Diese liegt, im Sinne der Systemanforderungen nach 3.3 als CAD-Datei im DXF-Format vor.

Da das Referenzbild im Verlauf des Programmes mehrmals geladen werden muss, wird aus der DXF-Datei in diesem Schritt eine PNG-Grafik abgeleitet. Außerdem werden alle wichtigen Maße aus der DXF-Datei abgeleitet. Dies ist sinnvoll, da sich das Laden einer DXF-Datei in der Praxis als wesentlich zeit- und rechenintensiver erweist. Im Anschluss wird zum Laden einer Referenzabbildung nur noch die abgeleitete PNG-Grafik verwendet.

4.2.2 Bildgebung

Im ersten physischen Schritt des Demonstrator-Programms muss ein Bild der Messfläche erstellt werden. Hierzu wird das in 3.5.4 ausgelegte System zur Bildgebung genutzt. Für die Kommunikation zwischen dem angeschlossenen Computer und dem Raspberry Pi wird zu Beginn des Programmes eine Secure Shell (SSH)-Sitzung über Ethernet gestartet, wodurch sich der Raspberry Pi über das Lokalnetzwerk steuern lässt. Zur Verbindung werden lediglich die IP-Adresse des Raspberry Pi benötigt, sowie ein Nutzername inklusive Passwort[18].

Der Datentransfer basiert auf dem Secure File Transfer Protocol (SFTP). Dabei sendet der Computer einen Befehl zur Erstellung eines Bildes über SSH an den Raspberry Pi, und kopiert die erstellte Bilddatei über SFTP in das eigene Verzeichnis. Am Ende des Programms wird die SSH-Sitzung beendet. In der Praxis wird der Bildgebungsprozess durch die folgenden Befehle im Programmcode realisiert:

```
CAMERA = cm.Camera()
CAMERA.take_send_img()
CAMERA.end_session()
```

Zunächst wird das Kamera-Objekt instanziiert, anschließend ein Bild erstellt und in das lokale Verzeichnis importiert. Am Schluss folgt das Beenden der Sitzung. Durch die Python-API *Paramiko* (siehe

[18] In dieser Anwendung → IP: *imagepi.local*, port: 22 (Standard für SSH)

Tabelle 4-1) gestaltet sich die praktische Implementierung intuitiv [71]. Das System aus Kamera und Raspberry Pi wird softwareseitig durch ein Objekt dargestellt, welches im Rahmen der Instanziierung eine Verbindung zwischen Computer und Raspberry Pi herstellt. Eine weitere Methode abstrahiert den Bildgebungs- und Importprozess. Als Betriebssystem für den Raspberry Pi wird *Raspberry OS 32-Bit Lite* genutzt. Für die Implementierung der Kamera wird eine Version von Debian Buster genutzt. Die Implementierung im neueren Debian Bullseye weist zum Zeitpunkt der Entwicklung noch Probleme auf. Eine Desktop-Umgebung ist für den hiesigen Verwendungszweck nicht notwendig. Zusätzlich ist das Raspberry Pi HQ-Kamera-Modul direkt systemkompatibel. Dementsprechend werden keine weiteren Gerätetreiber oder weitere Software für den Betrieb benötigt. In der Systemkonfiguration des Raspberry Pi muss die Kamera-Schnittstelle manuell aktiviert werden. Die Systemkonfiguration kann *headless* ohne VNC[19] durchgeführt werden, rein über den Windows SSH-Client.

Montage und konstruktive Annahmen

Bei der Montage des Kamera-Systems können für die nachfolgenden Bildbearbeitungsschritte bereits einige Fehler minimiert oder beseitigt werden. Die Kamera wird mittig über der Messfläche montiert. In 3.5.4 wurde die Montagehöhe so ausgewählt, dass die geforderte Messfläche von 300 mm x 400 mm (h, b) entsprechend 3.2 inklusive Referenzrahmen mit Außenkante gerade vollständig abgebildet werden kann. So wird die Pixeldichte auf der zu messenden Fläche maximiert. Die optische Achse der Kamera wird normal zur Messfläche ausgerichtet, um perspektivischen Verzerrungen vorzubeugen. In dieser Konfiguration ist, abgesehen von einer etwaigen Linsenverzeichnung, eine direkte parallele Projektion in der horizontalen und vertikalen Bilddimension gegeben. Die restlichen Verzerrungen bleiben im Bild bestehen und müssen im Rahmen der Bildverarbeitung beseitigt werden. Für den Aufbau am Demonstrator bedeutet dies, dass die Kamera mit maximaler Genauigkeit so positioniert wird, dass der Kamerachip parallel zur Messfläche liegt. Der Schnittpunkt zwischen der optischen Achse und der Messfläche soll der Mittelpunkt des Referenzrahmens sein. Die Montagehöhe von 550 mm folgt aus der Systemauslegung in 3.5.4 Nach der Befestigung erfolgt die das manuelle Justieren des Kameraobjektivs. Der optimale Fokus wird iterativ durch Prüfen der Bildschärfe bei einer gegebenen Objektivposition ermittelt. Als Prüfgröße wird auf einem Bild eines Schachbrettmusters geprüft, wie breit der Übergang von rein schwarzen zu rein weißen Pixeln ist. Die Fokuseinstellung mit dem schmalsten Übergang von schwarz zu weiß ist das Optimum und wird nach einstelligen Iterationen gewählt.

Das vollständige Kamerasystem aus Raspberry Pi und Kamerachip wird in einem 3D-gedruckten Gehäuse montiert, welches seitlich an Alu-Profilstangen im Bosch-Typ mit 10 mm Nutbreite montiert

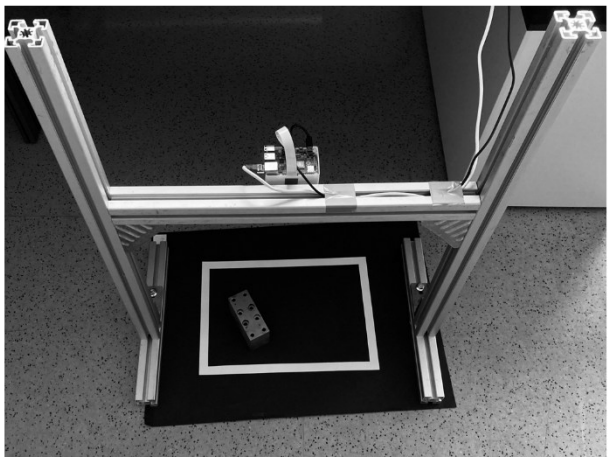
[19] Headless: Betrieb ohne direkte Peripherie; VNC (Virtual Network Computing): entspricht Remote-Desktop Verbindung

werden kann. Zwar ist eine Kommunikation mit dem Raspberry Pi über WLAN möglich, aber in der Praxis erweist sich eine Ethernet-Verbindung als schneller und robuster, mit deutlich geringerem Aufwand in der Implementierung. Die Kamera wird über USB mit Strom versorgt. Ein Raspberry Pi Modul für *Power-over-Ethernet* (PoE) basierte Stromversorgung existiert [72] und wäre perspektivisch eine Möglichkeit zur Optimierung des Systems.

Test des Kamera-Systems

An einem reduzierten Aufbau soll nun die Bildgebung getestet werden. Die schwarze Messunterlage wird durch einen Bogen schwarzen Tonpapiers ersetzt, der Referenzrahmen durch einen weißen Rahmen aus Papier, welcher auf der Unterlage verklebt ist. Der Rahmen wird für diesen Versuch durch einen innen ausgeschnittenen DIN A3-Bogen ersetzt, welcher die Außenmaße von 420x297 mm (b, h) besitzt. Der Vorteil dieser Konstruktion ist die garantierte perfekte Rechteckform der Außenseite des Referenzrahmens, welche für die Bildverarbeitung vorausgesetzt wird. Die Größe weicht für die nachfolgenden Versuche leicht von der Plangröße ab, weshalb die Kamerahöhe angepasst wird. Die Kamera wird entsprechend der Anforderungen mittig über dem Referenzrahmen an einem Gerüst aus Alu-Profilstangen montiert. Der zu Testzwecken konstruierte Prüfstand ist in Abbildung 4.4 dargestellt.

Optimierte Lichtverhältnisse im Schatte, inkl. Gerüst, Kamerasystem, Messoberfläche & Messobjekt



Nahaufnahme Kamerasystem (Raspberry Pi), inkl. Netzwerkkabel & Stromversorgung

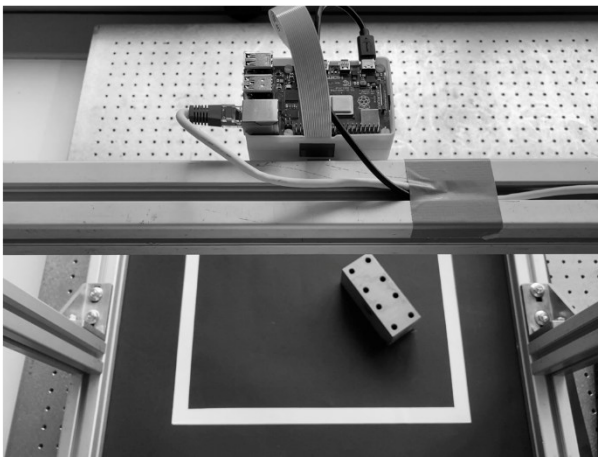


Abbildung 4.4: Prüfstand zum Test des Kamerasystems und der anschließenden Koordinatenberechnung

Ein erster Test der Kamera zeigt eine deutlich erkennbare Tonnen-Verzeichnung (vgl. Abbildung 2.5), welche entfernt werden muss. Zu Darstellungszwecken und zur Kalibrierung der Kamera wird ein Schachbrettmuster [73] in den Bildbereich gelegt. An diesem sind deutlich die durch die Verzeichnung gekrümmten Linienverläufe erkennbar (siehe Abbildung 4.5).

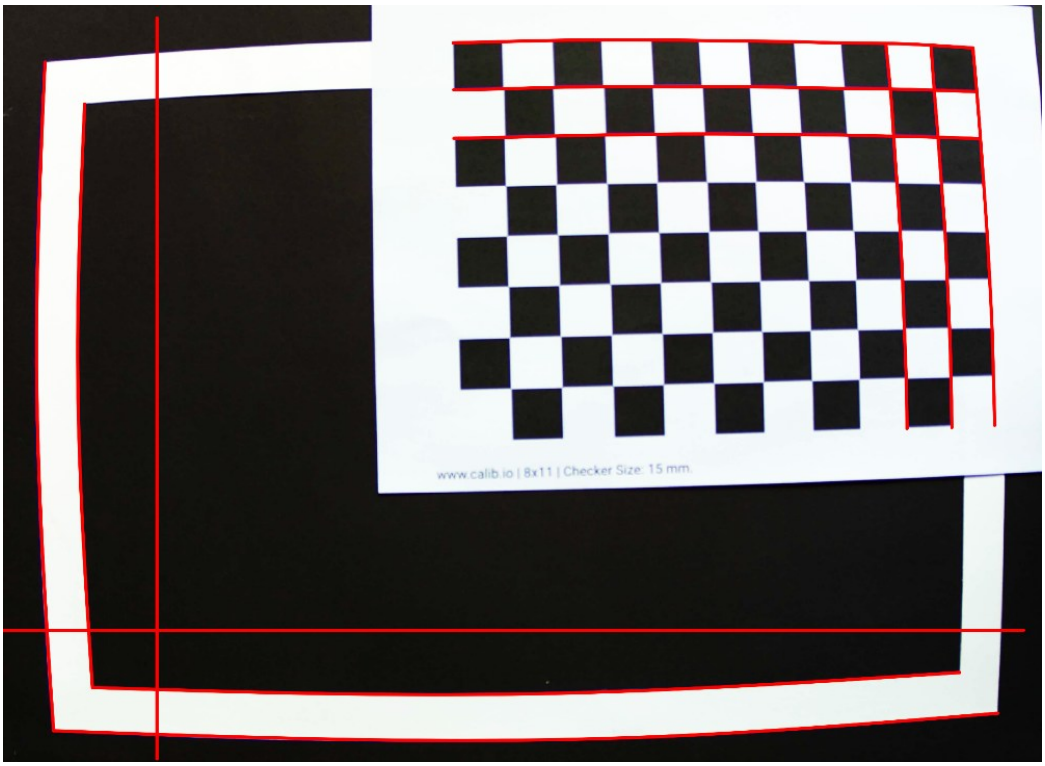


Abbildung 4.5: Deutlich erkennbare Tonnen-Verzeichnung am unbearbeiteten Kamerabild (Kontrast erhöht)

In OpenCV sind Methoden implementiert, mit welchen die Verzeichnung korrigiert wird [74]. Hierzu müssen zunächst auf Basis eines verzerrten Bildes mit Kalibrierungsmuster wie in Abbildung 4.5 die Transformationsparameter bestimmt werden. Da diese für eine gegebene Ausrichtung der Kamera nicht schwanken, ist eine einmalige Kalibrierung ausreichend. Um ein Kamerasystem zu kalibrieren, muss im fertigen Programm folgender Befehl ausgeführt werden:

```
cm.Camera.__calibrate__(calibration_file="test_image.png")
```

Der Parameter *calibration_file* verweist auf ein Kamera-Bild mit Abbildung des Messbereichs, in welchem ein Schachbrett-Target entsprechend Abbildung 4.5 eingelegt wird. Die Kalibrierungs-Parameter werden für die dauerhafte Nutzung in einer JSON-Datei gespeichert. Bei der Erstellung eines neuen Kamerabildes werden die zuletzt gespeicherten Kalibrierungsparameter zur Entzerrung des Kamerabildes genutzt. Durch eine Transformation der Bild-Projektionsmatrix werden direkt beim Importprozess die Verzeichnungen mit großer Präzision entfernt. Schachbrett-Targets, welche eine größere Fläche des Messbereiches abbilden, liefern sichtlich genauere Kalibrierungsergebnisse.

Als Endprodukt dieses Bearbeitungsschrittes liegt nun ein Kamerabild vor, welches durch das Entfernen der Verzeichnungen tauglich für die nachfolgende Objekterkennung und Koordinatenbestimmung ist. Abbildung 4.6 zeigt eine beispielhafte Abbildung eines Stahlbauteils, welches als praktisches Testobjekt genutzt wird. Im Vergleich zu Abbildung 4.5 ist die Orthogonalität des Referenzrahmens mit großer Genauigkeit gegeben.

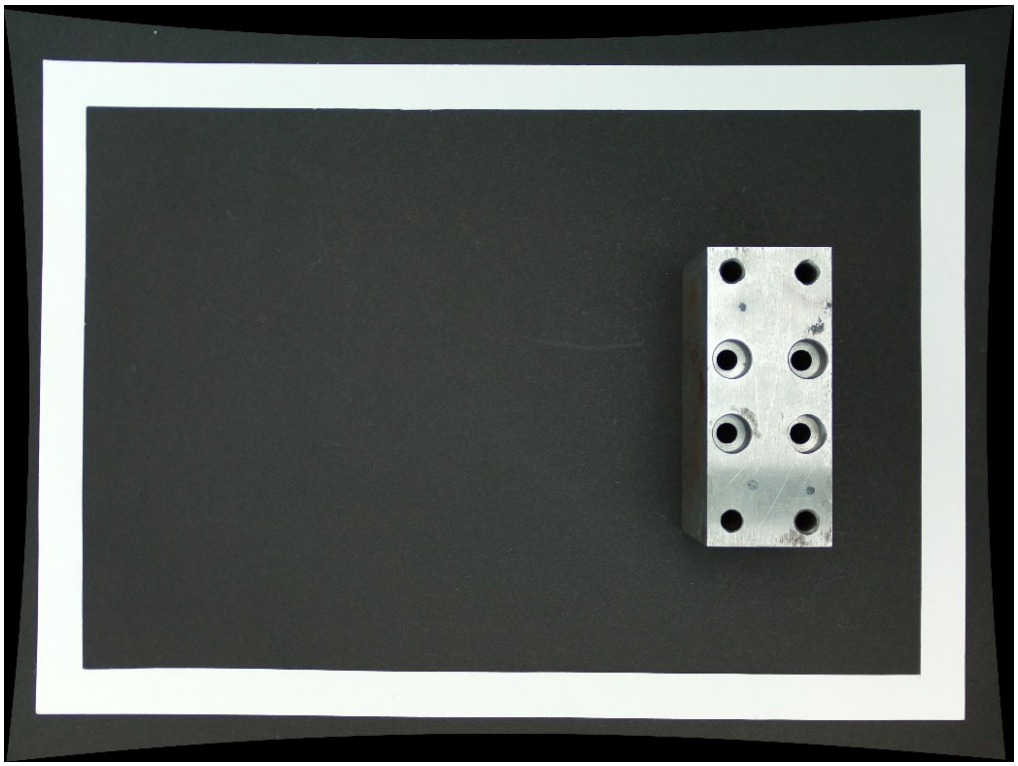


Abbildung 4.6: Entzerrtes Kamerabild. Sichtbare Einschnürung an den Seitenmitten durch die Transformation

4.2.3 Bildbearbeitung und Objekterkennung

Nach der Erstellung des Bildes muss dieses für die Berechnung der anzufahrenden Koordinaten vorbereitet werden. Die Durchführung dieser Prozessschritte erfolgt rein softwareseitig. Hierfür werden unter anderem Machine-Vision-Algorithmen aus der Python-Bibliothek *OpenCV* genutzt. Um die Maßnahmen im Rahmen dieses Abschnittes zu motivieren, wird qualitativ auch auf den nachfolgenden Abschnitt 4.2.4 vorgegriffen.

Motivation und Einführung

Da für die Berechnung der Punktkoordinaten im Wesentlichen Maße aus dem Kamerabild interpretiert werden müssen, wird der in Abbildung 3.11 dargestellte Referenzrahmen mit bekannten Maßen vorgesehen. Dieser markiert den zulässigen Bereich für die freie Positionierung eines bekannten Bauteils, und kann als Referenzmaß verwendet werden, welches auf die Position des zu vermessenden Blechbauteils bezogen werden kann. Im Rahmen der automatischen Bildbearbeitung wird zunächst das rohe Kamerabild für die Bestimmung der Koordinaten tauglich gemacht. Diese werden im Anschluss berechnet. Die Koordinaten werden der SPS gesendet, welche entsprechend die Kinematik steuert, sodass der Laser-Scanner sukzessive die Kantenprofile an den Messpunkten ermitteln kann.

Im Programmcode wird der gesamte Bildbearbeitungsprozess bis zur Ermittlung der anzufahrenden Koordinaten wie folgt abstrahiert:

```
IMG = mv.CameraImage(setup_obj=SETUP)
IMG._allign_to_frame()
IMG._get_object_data()
IMG._log_data(verbose=True)
```

Zunächst wird ein *CameraImage*-Objekt instanziiert, welches im Folgenden das zu bearbeitende Bild inklusive aller im Objekterkennungs-Prozess erhobenen Daten repräsentiert. Die gesammelten Metadaten des *MetaDataParser*- (Setup)-Objektes werden dem CameraImage-Objekt übergeben. Zusätzlich wird in der Instanziierung das RGB-Bild durch Thresholding in ein Binärbild umgewandelt[20], wobei bei Bedarf ein Rauschfilter angewendet wird. Diese Filter unterdrückt kleine Ansammlungen an Pixeln, welche durch den Threshholding-Prozess fälschlicherweise schwarz oder weiß dargestellt werden. Dadurch werden das Objekt und der Referenzrahmen weiß und der Rest schwarz dargestellt. Da durch den Grad der Filterbearbeitung proportional die Bildqualität sinkt [76], sollte das Bild grundsätzlich so wenig wie möglich filternd bearbeitet werden. Es erweist sich in der Praxis als sinnvoller, die Lichtverhältnisse des Kamerabildes im Voraus zu optimieren. Scharfe Lichtreflexionen auf der Messfläche, beispielsweise durch Sonneneinstrahlung, sollten vermieden werden, da in diesem Falle selbst die schwarze Papierunterlage zu viel Licht reflektiert. Der Kamera-Testaufbau aus Abbildung 4.4 wird in einem Innenraum mit direkter Sonneneinstrahlung im Schatten positioniert. Abbildung 4.6 zeigt die einheitliche Helligkeit der Messunterlage unter diesen Rahmenbedingungen. Hier kann beispielsweise bei einem Threshold-Wert von 95 ohne weitere Rauschfilter ein taugliches Bild zur Weiterverarbeitung erzeugt werden. Threshold-Werte und Parameter der Rauschfilter müssen jeder Situation entsprechend manuell kalibriert werden. In der Software sind Rauschfilter entsprechend Tabelle 4-3 integriert.

Tabelle 4-3: Im Programm integrierte Filter (Faltungs-Matrizen)

5x5 Mittelwert	3x3 Mittelwert	3x3 Gaus (Binomial-Filter)
$\frac{1}{25}\times\begin{pmatrix}1 & \cdots & 1\\ \vdots & \ddots & \vdots\\ 1 & \cdots & 1\end{pmatrix}$	$\frac{1}{9}\times\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}$	$\frac{1}{16}\times\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix}$

Anschließend werden jegliche noch vorhandenen perspektivischen Verzerrungen aus dem Bild entfernt. Die übrigen Verzerrungskomponenten werden in 2.3.2 beschrieben und können nacheinander algorithmisch entfernt werden. In der normalen Lage erscheint das Bauteil größer, weiter entfernt vom Mittelpunkt und besitzt in der Regel sichtbare Seitenflächen, welche im Binärbild im

[20] Beim Threshholding wird ein Mindest-Helligkeitswert (Threshold) für jedes Pixel vorgegeben, wobei jeder Wert darunter den Helligkeitswert 0 (schwarz) und darüber den Wert 255 (weiß) erhält [75]. Hierfür muss das RGB-Bild zunächst in ein Schwarz-Weiß-Bild umgewandelt werden.

Allgemeinen nicht von der Objekt-Oberseite unterscheidbar sind. Diese Fehler verzerren die Kontur des Objekts und müssen trotz kaum sichtbarer Effekte bearbeitet werden, da selbst Fehler im Sub-Millimeter Bereich die Koordinaten entsprechend verfälschen. Im Folgenden werden notwendige Konventionen und Annahmen dargelegt, welche den Bildverarbeitungsprozess veranschaulichen und plausibilisieren.

Koordinaten-Konvention

Im Programm wird die Kontur des Rahmens ermittelt. An bekannten Dimensionen (Breite, Höhe) der Rahmenkontur können nun Pixel gezählt werden, wodurch ein Verhältniswert aus Millimetern pro Pixel entsprechend Gleichung 4.1 entsteht. Wenn das Bild in der horizontalen und vertikalen Bilddimension unverzerrt ist, beziehungsweise Pixel quadratisch sind, ist dieser Wert für beide Dimensionen gleich.

$$\mathrm{X}_{\mathrm{mm,px}} = \frac{l_{\mathrm{Rahmen},i}}{px_{\mathrm{Rahmen},i}} \tag{4.1}$$

Die linke obere Ecke des Rahmens wird als Nullpunkt für die Berechnung der Koordinaten genutzt, womit jeder Punkt auf dem erfassten Bild eine eindeutig zugeordnete Koordinate erhält. Gezählt wird im Millimeterbereich positiv nach rechts für die horizontale Komponente, sowie positiv nach unten für die vertikale Komponente. Diese Koordinaten-Konvention wird verwendet, da in jeder Pixelgrafik die Pixelkoordinaten analog gezählt werden. Da der Rahmen rechteckig ist, werden die Koordinaten in kartesischer Form formuliert, wobei die beiden Koordinatenachsen sinnvollerweise entlang des Referenzrahmens verlaufen. An jedem Messpunkt wird aus den Bilddaten der notwendige Drehwinkel für den Laser-Scanner interpretiert[21] und die von der Kinematik einzustellende Höhe ergibt sich aus der konstanten Höhe des Blechbauteils, welche vorgegeben werden muss, sofern das zu vermessende Bauteil bekannt ist. So entsteht für jeden Messpunkt ein 4-Tupel aus drei translatorischen und einer rotatorischen Koordinate.

Referenzdaten-Konvention

Da für die Objekterkennung CAD-Daten als Referenzmodell vorliegen, wird zur Vereinheitlichung der Implementierung und Nutzung dieser Referenzen eine Abbildungs-Konvention gewählt. Diese ist relevant für viele Teile der in diesem Abschnitt beschriebenen Algorithmen zur Bildbearbeitung.

Die Referenzmodelle der Blech-Objekte im DXF-Format sind Vektorgrafiken mit Dimensionsangaben. Anders als Pixelgrafiken besitzen diese keine eindeutige Abbildungsgröße, die Skalierung kann frei

[21] Für die Ursachen dieser Vorgabe siehe 3.5.1

gewählt werden. Da das Auslesen dieser Dateien verhältnismäßig viel Rechenaufwand benötigt, werden im ersten Schritt die wichtigen Bemaßungen extrahiert und die Datei in eine PNG-Grafik konvertiert. Für die Erstellung von Bild-Referenzen wird eine Abbildungskonvention ausgewählt, welche gerade für die Bildbearbeitung und Positionsbestimmung des Bauteils relevant ist. Die PNG-Referenzbilder liegen so alle in der gleichen Form vor. Zunächst wird die Oberflächenkontur des zu vermessenden Blechbauteils ermittelt, und anschließend das kleinstmögliche, das Objekt vollständig umrandende Rechteck dieser Kontur ermittelt. Das Bild wird anschließend rotiert, sodass jene Umrandung der neue Rahmen des Bildes ist, sowie deren Breite größer als deren Höhe ist. Dies ist für die Ermittlung der Bauteil-Rotation relevant. Die DXF-Dateien müssen nach Konvention positioniert sein. So kann direkt aus den in der DXF-Datei enthaltenen Dimensionsangaben das Millimeter-zu-Pixel-Verhältnis nach Gleichung 4.1 ermittelt werden, was die nachfolgenden Bearbeitungs-Prozesse erheblich vereinfacht. Das Einhalten dieser Konvention ermöglicht das Vernachlässigen vieler Randbedingungen, welche so nicht beachtet werden müssen. Dies vereinfacht das Vermeiden von Fehlern und erleichtert den Umgang mit den Referenzdaten, sobald diese entsprechend vorliegen. Abbildung 4.7 und Abbildung 4.8 veranschaulichen die Darstellung eines DXF-Objektes nach Referenzdatenkonvention.

Abbildung 4.7: Als PNG-Grafik angezeigtes DXF-Objekt, in Autodesk AutoCAD erstellt, frei positioniert

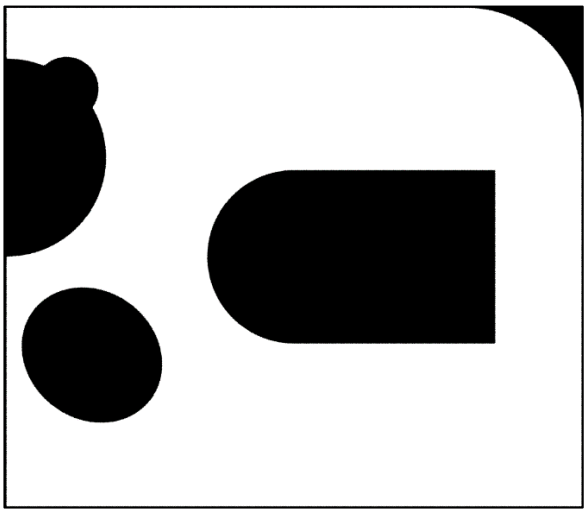


Abbildung 4.8: Positionierung der Referenzdaten nach Konvention (schwarze Umrandung zur Darstellung)

Vorbemerkungen und Annahmen zum Entzerrungs-Algorithmus

Der in dieser Arbeit entwickelte Algorithmus zur Entfernung perspektivischer Verzerrungen ist aus Überlegungen und Annahmen bezüglich der Geometrie und der optischen Fehlerquellen entstanden. Er eignet sich in der hier genutzten Form zum Beseitigen der zuvor beschriebenen perspektivischen Fehlertypen. Im Folgenden werden die relevanten Beobachtungen aufgelistet, welche die Entwicklung des Algorithmus maßgeblich beeinflusst haben.

1. Entsprechend dem Lochkamera-Modell nach 2.3.2 werden Objekte vor dem Brennpunkt der Kamera entsprechend dem Strahlensatz in jeder Raumdimension linear hochskaliert, wenn sie sich der Kamera annähern, und erscheinen im selben Zusammenhang kleiner, wenn sie sich entfernen.
2. Die der Kamera abgewandten Seitenflächen des Blechbauteils werden grundsätzlich nicht abgebildet, die der Kamera zugewandten Seitenflächen allerdings immer. Zugewandt ist eine Seitenfläche des Blechbauteils, wenn ein Vektor, welcher in der Messfläche aus der optischen Achse hinauszeigt mit dem Flächen-Normalvektor an dem bestimmten Bauteilpunkt einen Winkel $90° < \alpha < 270°$ aufspannt. Alle anderen Punkte sind der Kamera abgewandt (siehe Abbildung 4.9). Die Grenzfälle $90°;\ 270°$ werden als undefiniert angenommen.

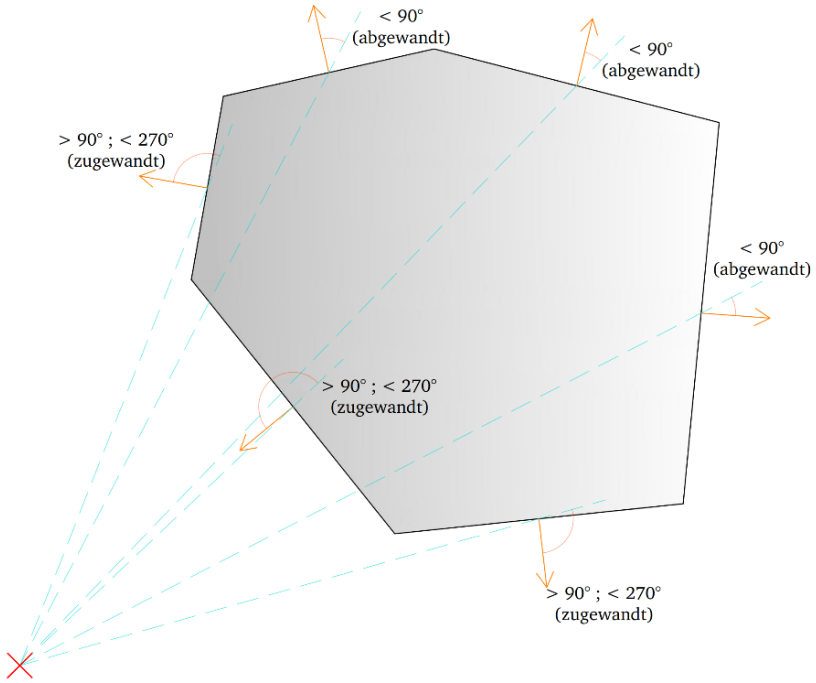


Abbildung 4.9: Veranschaulichung der Voraussetzung für die Sichtbarkeit der Seitenflächen mit optischer Achse (rot) und Blechbauteil (grauer Verlauf)

Für einen geschlossenen Körper bedeutet dies, dass maximal die Hälfte der Oberflächenkontur der Kamera zu und der Rest abgewandt ist, wenn die optische Achse das Blechbauteil nicht schneidet. Wenn dies aber der Fall ist, ist als Grenzfall **keine** Außenseite außer der Oberseite der Kamera zugewandt. Im Allgemeinen können dennoch innere Konturen, beispielsweise Bohrungen oder Ausstanzungen, auch in diesem Fall von diesem Abbildungsfehler betroffen sein.

3. Die Oberseite ist vor diesem Schritt gemäß dem Lochkamera-Modell verzerrungsfrei in dem Bild enthalten. Ausgehend von den der Kamera abgewandten Seiten ist also die Bauteiloberseite unverzerrt dargestellt.

4. Wenn ein Blechbauteil gemäß 2.1 vorliegt, entspricht die verzerrte äußerste Kontur des abgebildeten Blechbauteils im Binärbild an den der Kamera zugewandten Seiten der unverzerrten, korrekt skalierten Kontur der Objektunterseite. Dies wird in Abbildung 4.15 ersichtlich.
5. Die korrekte Kontur der Kamera-abgewandten Bauteilseiten lässt sich aus der korrekten Skalierung und Translation der gesamten verzerrten Kontur konstruieren. Die Translation und Skalierung müssen dabei so erfolgen, dass beide Abbildungen denselben optischen Fluchtpunkt besitzen. So wird effektiv eine Abbildung konstruiert, bei welchem eine Kopie des Blechteils hinter das ursprüngliche Teil gelegt wird, wobei dessen Oberseite der Unterseite der unbearbeiteten Abbildung entspricht. Wenn die Abbildungen getrennt vorliegen, ist das Resultat die logische OR-Verknüpfung beider Abbildungen. Die der Kamera abgewandten Seitenflächen des neu skalierten Bauteils sind korrekt positioniert und unverzerrt dargestellt. Mit den Informationen aus **Punkt 4** existieren nun alle Informationen zur Konstruktion der vollständigen unverzerrten Bauteilkontur. In der OR-Darstellung beider Bilder sind allerdings alle Seiten mit Verzerrungen abgebildet, da jeweils die unverzerrten Seitenflächen der einen Abbildung von verzerrt dargestellten Seitenflächen der anderen Abbildung überdeckt werden. Dies wird in Abbildung 4.11 ersichtlich.
6. Die sichtbaren Seitenflächen der skalierten und verschobenen Objekt-Kopie, sowie die zu groß dargestellte Abbildung der tatsächlichen Oberseite des Objektes lassen sich entfernen, wenn die Abbildung von ursprünglichem Objekt inklusive Kopie entsprechend **Punkt 5** mit einer passenden Bit-Maske und entsprechender logischen Operation bearbeitet werden.
7. Diese Bit-Maske ist die XOR-Verknüpfung der ursprünglichen Abbildung, sowie der skalierten und verschobenen Abbildung. Diese ist in Abbildung 4.12 grafisch dargestellt und entspricht ebenso einem Binärbild. Die Operation ist eine Subtraktion (A AND NOT B) dieser Bit-Maske von der gemeinsamen Abbildung beider Objektdarstellungen nach **Punkt 5**. Das Resultat dieser Operation entspricht dem Ausgangsbild, wobei das verzerrte Objekt durch ein Objekt mit der korrekten Außenkontur ersetzt wird. Wenn das darzustellende Objekt innere Konturen, beispielsweise Löcher besitzt, kann nun die Referenzabbildung des Objektes auf die vorhandene Kontur projiziert werden, um die inneren Konturen ebenso abzubilden.

Die Änderungen an den Bilddaten, die dieser Algorithmus verursacht sind im Allgemeinen sehr klein und gerade für dünne Blechbauteile fast nicht bei bloßer Gegenüberstellung der Bilder erkennbar. Da der Computer aber Koordinaten mit einer Genauigkeit von mindestens $\pm\, 0{,}5\ mm$ entsprechend 3.5.4 ermitteln soll, ist die Entfernung selbst kleiner perspektivischer Verzerrungen notwendig. Zusätzlich werden die im Folgenden verwendeten Abbildungen zur Veranschaulichung überzeichnete Fehler aufweisen, welche in der Realität kleiner wären.

Algorithmus zur Entfernung perspektivischer Verzerrungen

Zur Veranschaulichung des Algorithmus werden in Abbildung 4.11 bis Abbildung 4.16 die einzelnen Bearbeitungsschritte grafisch mit deutlich verstärkten Fehlern veranschaulicht. Ebenso ist in jedem Schritt das auf die korrekte Größe skalierte Referenzbild unten abgebildet. Dies kann bereits im Vorfeld geschehen, da das Verhältnis von Millimetern pro Pixel aufgrund des Referenzrahmens im Kamerabild und der DXF-Metadaten in der Referenz bekannt ist[22].

Dargestellt ist in jedem Bild ein Ausschnitt des Kamerabildes. Ein Teil des Referenzrahmens ist außen sichtbar, ebenso durch ein rotes Kreuz der Schnittpunkt von Bildachse und Bildebene. Dieses ist logischerweise der Mittelpunkt im Kamerabild und zugleich der optische Fluchtpunkt. Zu Darstellungszwecken ist der nicht durch den Algorithmus bearbeitete Hintergrundbereich schraffiert dargestellt. Das dargestellte Objekt ist ein Blechbauteil mit unregelmäßiger Außenkontur, sowie einer unregelmäßigen dreieckigen Innenkontur. Ein Ausschnitt des unbearbeiteten Bildes als Ausgangspunkt ist in Abbildung 4.10 dargestellt. Im Folgenden werden die Blechbauteil-Seitenflächen zur Übersichtlichkeit mit einem Farbverlauf dargestellt, selbst wenn diese im Binärbild nicht von der Bauteil-Oberseite unterscheidbar sind.

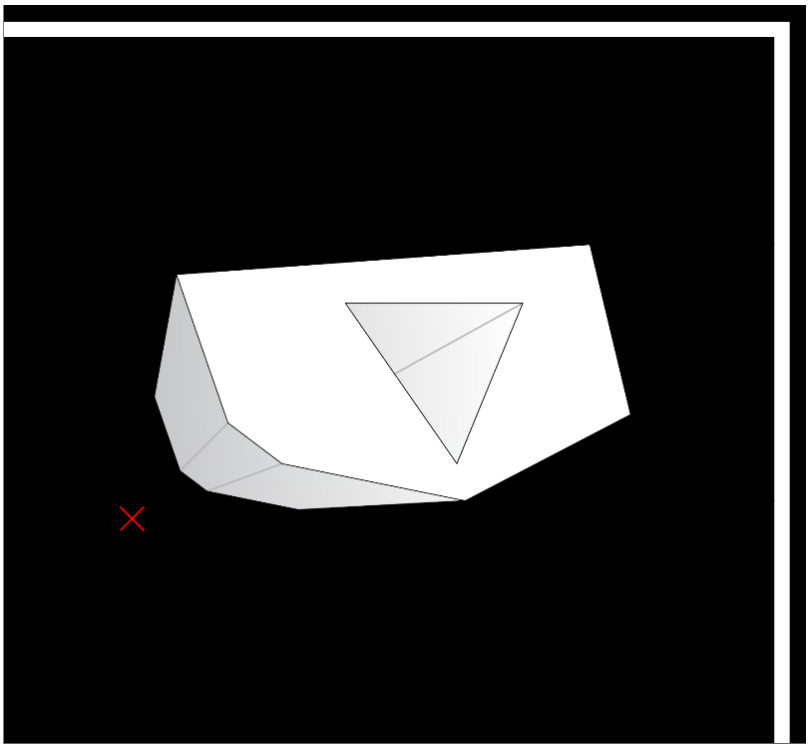


Abbildung 4.10: Bildausschnitt des unbearbeiteten, „simulierten“ Kamerabildes

[22] Siehe hierzu auch Gleichung 4.1

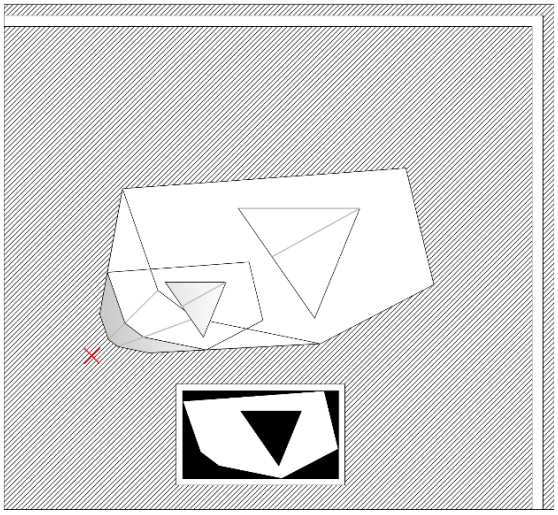


Abbildung 4.11: Entzerrungsalgorithmus Schritt 1
Skalieren und Verschieben der verzerrten Kontur an korrekte Stelle, → Objekt-Oberseite = Referenz

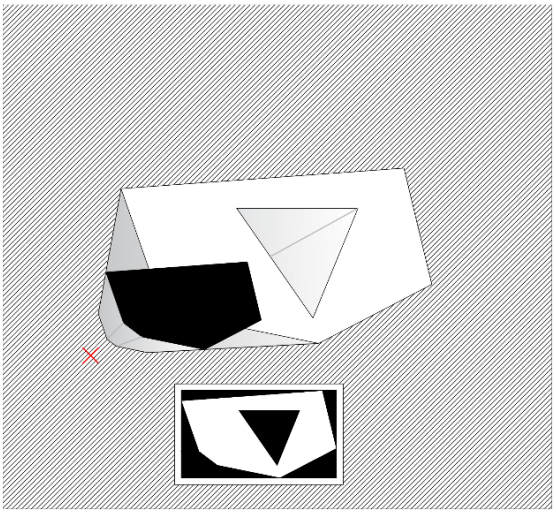


Abbildung 4.12: Entzerrungs-Algorithmus Schritt 2
Erzeugen einer Bit-Maske zur Bestimmung der unverzerrten Objektkontur

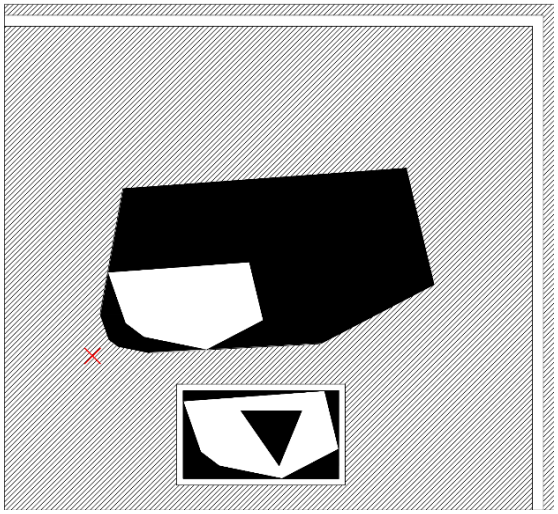


Abbildung 4.13: Entzerrungs-Algorithmus Schritt 3
Subtraktion Maske aus Schritt 2 von Bild aus Schritt 1 → weiße unverzerrte Kontur

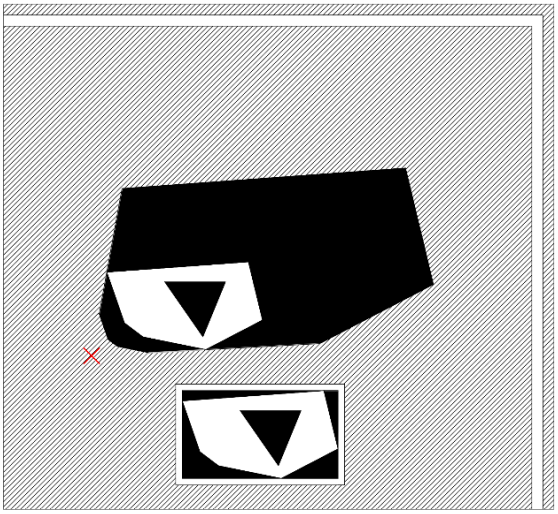


Abbildung 4.14: Entzerrungs-Algorithmus Schritt 4
Projektion der Referenz auf unverzerrte Kontur, um auch innere Konturen abzubilden

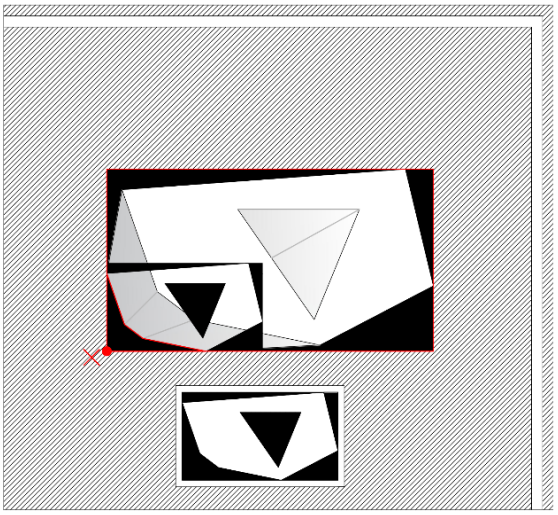


Abbildung 4.15: Entzerrungs-Algorithmus Vergleich

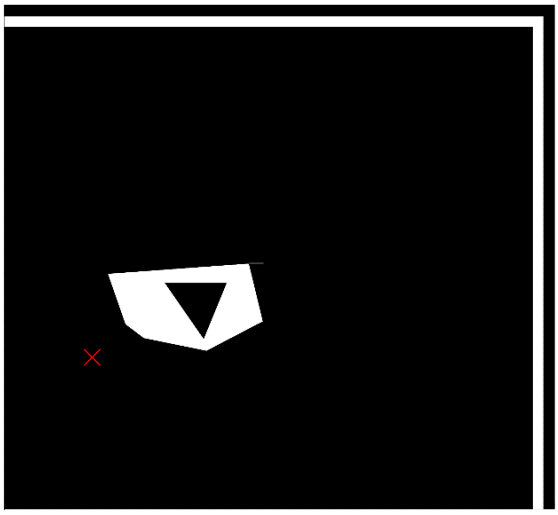


Abbildung 4.16: Entzerrungs-Algorithmus Endergebnis

Im **ersten** Schritt (siehe Abbildung 4.11) wird das Blechbauteil im binären Kamerabild erfasst. Die 2D-Projektion des Blechbauteils inklusive aller sichtbaren Verzerrungen wird kopiert, skaliert und in der Bildebene verschoben, sodass beide Abbildungen denselben optischen Fluchtpunkt besitzen. Die Verschiebung und Skalierung der verzerrten Bauteilkontur erfolgt so, dass die Oberseite dieser Abbildung der theoretischen Abbildung der Unterseite der ursprünglichen Abbildung entspricht. Dieses Vorgehen resultiert aus Punkt 5 der Vorbemerkungen.

Um diese Abbildung zu realisieren, wird das erfasste Objekt ausgeschnitten und um den Objektmittelpunkt[23] skaliert. Die Skalierung leitet sich aus dem Strahlensatz her und wird in Abbildung 4.17 dargestellt. Betrachtet wird das Verhältnis aus projizierter Referenzfläche und der Objektoberseite, welche sich näher an der Kamera befindet. Es ergibt sich ein geometrischer Zusammenhang für den Skalierungsfaktor entsprechend Gleichung 4.2, wobei b, B, B' jeweils die tatsächliche Objektbreite, die tatsächliche- sowie die auf die Objekthöhe projizierte Referenzflächenbreite darstellen (graue Linie in Abbildung 4.17 auf Höhe h).

$$\frac{b}{B} = \chi \frac{b}{B'} \;\Rightarrow\; \chi = \frac{B'}{B} = \frac{H-h}{H}$$

$$\chi = \frac{H-h}{H} \tag{4.2}$$

Dass zum Erzeugen der beschriebenen Abbildung eine Verschiebung des Objektes in der Bildebene notwendig ist, lässt sich geometrisch aus dem Lochkamera-Modell herleiten und ist ebenfalls in Abbildung 4.17 dargestellt. Der Objektmittelpunkt erscheint auf der Bildebene (unten) weiter von der optischen Achse entfernt. Entsprechend Abbildung 4.17 ergibt sich in jeder Bilddimension ein geometrischer Zusammenhang aus dem Abstand eines Punktes zur optischen Achse, dessen Höhe über der Bildebene und der Höhe der Kamera zur notwendigen Translation, um diesen Fehler auszugleichen. Sie ergibt sich in jeder Koordinatenrichtung aus Gleichung 4.3.

$$\tan\varphi = \frac{\varepsilon}{H-h} = \frac{\Delta\varepsilon}{h}$$

$$\Delta\varepsilon = \frac{h}{H-h}\varepsilon \tag{4.3}$$

Die Verschiebung findet dabei immer in Richtung der optischen Achse statt.

[23] Der Objektmittelpunkt entspricht im Folgenden dem Mittelpunkt des minimal umgebenden Rechtecks der Kontur. Durch die Rechteckform können die eingerahmten Bilddaten als Tensor gespeichert werden, was das Datenmanagement, analog zur Referenzdaten-Konvention vereinfacht.

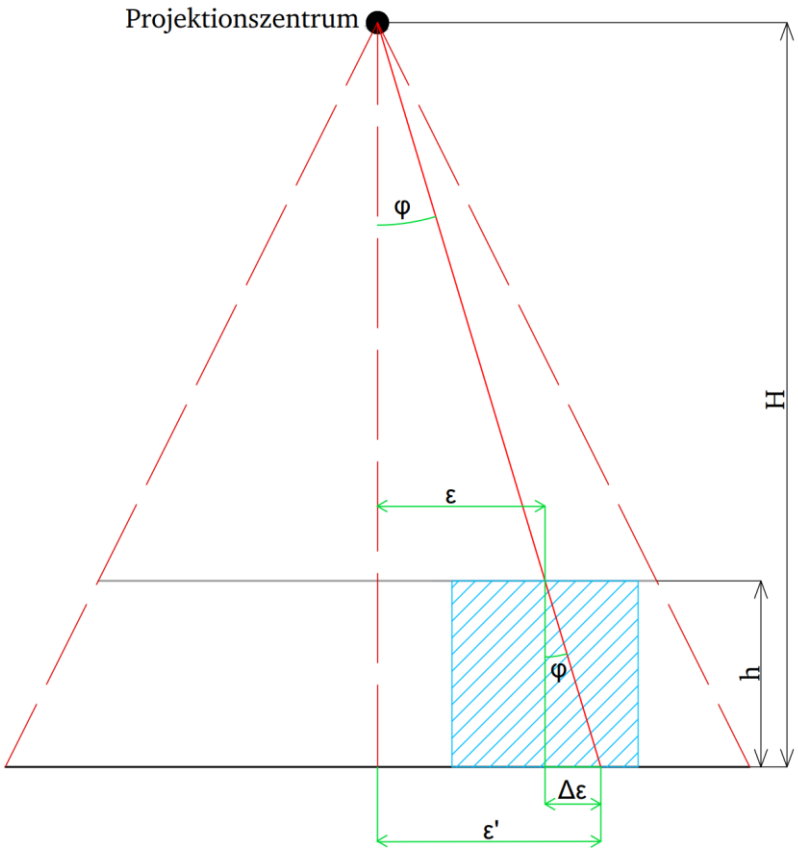


Abbildung 4.17: Scheinbar größere Exzentrizität und Größe von dreidimensionalen Objekten (blau) in der Bildebene

Nun liegen zwei Abbildungen (unbearbeitet bzw. skaliert u. verschoben) vor, welche durch die binäre Darstellung als Bitmasken fungieren können. Diese Eigenschaft wird im Folgenden genutzt, um die gewünschte unverzerrte Abbildung des Objektes auf der Referenzfläche zu generieren. Abbildung 4.11 zeigt die logische OR-Verknüpfung beider Bilder, welche sowohl die der Kamera abgewandten als auch zugewandten Objektseiten beinhaltet.

Im **zweiten** Schritt (siehe Abbildung 4.12) werden die beiden Abbildungen aus dem vorigen Schritt über eine XOR-Verknüpfung pixelweise auf Ungleichheit geprüft. So entsteht bereits die Darstellung aus Abbildung 4.12. Die Ungleichheiten beschränken sich logischerweise auf die jeweils individuellen perspektivischen Verzerrungen beider Darstellungen, sodass das Resultat dieser Verknüpfung eine Maske aller übrigen Verzerrungen darstellt. Wichtig ist auch, dass der Referenzrahmen aus der Abbildung gelöscht wird, da dieser auf beiden Abbildungen identisch ist, da er nicht bearbeitet wurde.

Im **dritten** Schritt (siehe Abbildung 4.13) wird die Bit-Maske aus dem vorherigen Schritt auf die OR-Verknüpfung beider Abbildungen angewandt. Die Bit-Maske wird subtrahiert, sodass diese effektiv alle verbleibenden perspektivischen Verzerrungen löscht, welche in der OR-Abbildung per Definition ebenso enthalten sind. Der Referenzrahmen bleibt bestehen, da dieser nicht in der Maske enthalten ist. Übrig bleibt eine Abbildung, deren Außenkontur der entzerrten Bauteil-Kontur entspricht.

Im **vierten** Schritt (siehe Abbildung 4.14) wird die Abbildung aus dem vorherigen Schritt nach Referenzdatenkonvention ausgerichtet, und das Referenzbild an die Stelle der Kontur projiziert. Durch die Abbildungs-Konvention kann die Referenz in das Bild eingefügt werden, ohne dass eine weitere algorithmische Ausrichtung der Referenzdaten nötig wäre. Durch dieses Einfügen können innere Konturen des Bauteils abgebildet werden, welche durch die Verzerrung der Seitenflächen in der Abbildung aus dem vorherigen Schritt im Allgemeinen falsch dargestellt werden.

Falls durch die Belichtungsverhältnisse die Seitenflächen des Bauteils im Rahmen des Thresholding nicht sichtbar sind, können Schritt zwei und drei des Entzerrungs-Algorithmus übersprungen werden. In der Praxis wird dies überprüft, indem die Seitenmaßverhältnisse der minimalen eingrenzenden Rechtecke verglichen werden. Wenn die Verhältnisse hinreichend ähnlich sind, werden besagte Schritte übersprungen.

Weitere Bildverarbeitungsschritte

Anschließend wird das Bild über die Methode *allign_to_frame* entlang des Referenzrahmens ausgerichtet, um etwaige leichte Rotationen zu entfernen und um die Laufzeit zu optimieren, da gegebenenfalls nicht benötigte Bilddaten abgeschnitten werden. Zusätzlich sind nun die Millimeterkoordinaten entsprechend der Koordinaten-Konvention proportional von den Pixelkoordinaten abhängig. Über die Methode *get_object_data* wird anschließend die allgemeine Lage des Blechbauteils, in Form zweier translatorischer und einer rotatorischen Komponente, auf der Messfläche ermittelt. Ausgegeben werden die geometrischen Lagedaten des minimal umgebenden Rechtecks des dargestellten Objektes. Hier gilt die Annahme, dass jedes Bauteil ein eindeutiges minimales umgebendes Rechteck besitzt. Die erfassten Daten sind:

- Der Objektmittelpunkt als 2D-Koordinate (mm- oder Pixel-System)
- Das horizontale und vertikale Maß des Rechteckes
- Die Rotation des Rechteckes

Da nach Ausführung des Entzerrungsalgorithmus das Referenzobjekt in die Bildebene projiziert wird, entspricht die Wahl des minimalen umgebenden Rechteckes der Eingrenzung des Objektes nach Referenzdatenkonvention. Die Position und Ausrichtung des tatsächlichen Bauteils lassen sich demnach indirekt aus der Position und Ausrichtung des ermittelten Rechteckes bestimmen. Ein einfacher Konturenvergleich wäre nicht möglich, da OpenCV keine sinnvolle Methode hierzu liefert, welche effizienter oder einfacher zu implementieren wäre.

Fazit zur Bildverarbeitung

Im Rahmen der Bildbearbeitung kann ohne eine komplexe Segmentierung einzelner Bildelemente die Sollgeometrie des zu vermessenden Blechbauteils auf die Abbildung des tatsächlichen Bauteils projiziert werden. Da die Bearbeitung generell auf Prinzipien der linearen Optik basiert, ist die Durchführung in der Praxis robust, da nur minimale statistische Fehlerquellen entstehen.

Ein bekanntes Problem für den Algorithmus ist der Verlust der Eindeutigkeit der Objekterkennung bei Objekten, welche mehrere Lösungen für ein minimal umgebendes Rechteck besitzen, welche nicht in 90-Grad-Intervallen zueinander liegen. In diesem Falle ist der gestellte Algorithmus der Methode *minAreaRect* in OpenCV, welche eine Basis des Algorithmus ist, nicht in der Lage alle möglichen Lösungen für dieses Problem zu berechnen. Somit können die Referenzdaten gegebenenfalls um wesentliche Winkeldifferenzen falsch projiziert werden. Dieser Fehler ließe sich bei einer für die hiesige Anwendung spezialisierten Implementierung dieser Methode beheben. Ein weiterer Lösungsansatz, welcher bei der Entwicklung in Betracht gezogen wurde, ist die Ermittlung des Pixelschwerpunktes [77]. Zwischen eventuell falsch projiziertem Referenzbild und der entzerrten Kameraabbildung kann die Winkeldifferenz bestimmt werden, was nach einer Implementierung teils rotatorische Fehler bereinigen kann. Der Algorithmus erweist sich allerdings als nicht robust und stellt im Allgemeinen eher eine Quelle von Abbildungsfehlern dar. Durch die pixeldiskrete Abbildung der Bilder und daraus resultierende Rundungsfehler können Ungenauigkeiten in der Winkelberechnung entstehen. Perspektivisch sinnvoll wäre hier eine robustere, aber weniger triviale Implementierung eines Algorithmus auf Basis der Hausdorff-Metrik [78,79].

Programmausgabe/Datendarstellung

Über die Methode *log_data* werden die Ergebnisse der Objekterkennung in der Konsole ausgegeben. Diese beschränken sich auf die innerhalb der Methode *get_object_data* erhobenen Bauteildaten. Dies ist die erste Ausgabe, welche über das Terminal ausgegeben werden kann, und dient zur Kontrolle, ob ein Objekt korrekt erkannt wurde. Die Daten werden, wie im Folgenden zu Darstellungszwecken leicht umformatiert, ausgegeben.

```
OBJECT DATA -> IMAGES/reference25.png (12/06/2023 11:32:10):
-------------------------------------------------------------------------------
|angle: 37.3104°| Coordinates x: 208.625mm| y: 145.548mm
| Frame -> width: 195.556mm | height: 97.6893mm|
-------------------------------------------------------------------------------
```

Zur Übersicht werden alle Zahlenwerte willkürlich auf sechs signifikante Stellen gerundet.

4.2.4 Berechnung von Punktkoordinaten

In den Metadaten können feste Messpunkte vorgegeben werden, welche auf der Bauteil-Oberfläche vermessen werden sollen. Zusätzlich soll die Möglichkeit gegeben sein, im laufenden Betrieb bei Bedarf weitere Messpunkte auszuwählen.

```
MEASUREMENT = mv.MeasurementPoints(setup_obj=SETUP)
MEASUREMENT._insert_static_points()
MEASUREMENT._enter_custom_points()
MEASUREMENT._export_points(source=IMG)
MEASUREMENT._log_data(verbose=True)
```

Für die Ermittlung der Koordinaten wird zunächst ein *MeasurementPoints*-Objekt instanziiert, welchem direkt die relevanten Metadaten des Setup-Objekts übergeben werden.

Über die Methode *insert_static_points* werden die vorgegebenen Koordinaten in bauteilabhängigen Koordinaten geladen. Diese Koordinaten sind entsprechend der Referenzdatenkonvention mit Ursprung in der linken oberen Ecke der Referenzabbildung gewählt. Diese können wie folgt dargestellt für jedes individuelle Bauteil in der JSON-Referenzdatei vorab notiert werden:

```
"reference23.dxf": {
    "coords_mm": [
        [66, 59],
        etc.
    ]
},
```

Vorbestimmte Messpunkte, die nicht auf einer Bauteilkante liegen, also nicht messbar sind, werden im Messprozess ignoriert. Mit der Methode *enter_custom_points* wird eine Kopie der Referenzabbildung geöffnet, in welcher per Mausklick nach Belieben weitere Messpunkte ausgewählt werden können. Die hier erzeugten Koordinaten werden im Bild als Pixelkoordinaten ermittelt und da für die DXF-Referenzdateien die Rahmendimensionen bekannt sind werden diese direkt analog zu den vorgegebenen Koordinaten in bauteilbezogene Millimeter-Koordinaten umgerechnet. In der grafischen Anzeige werden die vorgegebenen Koordinaten bereits markiert, sodass eine Doppelung einzelner Koordinaten durch den Bediener vermieden werden kann.

Mit der Methode *export_points* werden die gesammelten Koordinaten auf den Export zur SPS vorbereitet. In diesem Rahmen werden die Koordinaten auch von bauteilbezogenen Koordinaten auf referenzrahmenbezogene Koordinaten umgerechnet. Zusätzlich liegen die bauteilbezogenen Koordinaten in normierter Form von -1 bis 1 in beiden Bauteildimensionen vor. Durch die Normierung wird die Umrechnung auf messflächenbezogene Koordinaten vereinfacht. Der Objektmittelpunkt, sowie horizontale und vertikale Abmaße wurden in vorigen Schritten bereits ermittelt und können auf die

normierten Koordinaten bezogen werden. Der zugrundliegende mathematische Prozess entspricht einer nicht-linearen Koordinaten-Transformation.

Koordinaten-Transformation

Die im Rahmen der Objekterkennung ermittelten Bauteil-Lageparameter (Rotation, Höhe, Breite, Lage des Mittelpunktes) werden für die Berechnung einer Koordinatentransformation benötigt und werden durch Übergabe des *CameraImage*-Objektes der Methode verfügbar gemacht. Gedreht und skaliert wird um den Objektmittelpunkt, dessen Lage als konstanter Koordinaten-Versatz $\vec{x}_{\text{rahmen,obj,mm}}$ mit in die Transformation einbezogen wird. Die Transformationsgleichung liegt dann in der Form nach Gleichung 4.4 vor:

$$\vec{x}_{\text{rahmen,mm}} = R_{2\text{D}}(\alpha_{\text{obj}}) \begin{pmatrix} \frac{b_{\text{mm}}}{2} & 0 \\ 0 & \frac{h_{\text{mm}}}{2} \end{pmatrix} \vec{x}_{\text{obj,0}} + \vec{x}_{\text{rahmen,obj,mm}} \tag{4.4}$$

$R_{2\text{D}}(\alpha_{\text{obj}})$ ist die zweidimensionale Rotationsmatrix in Abhängigkeit des Objektrotationswinkels α_{obj}, anschließend folgt eine lineare Skalierung der Koordinatenkomponenten für die erste Umrechnung der bauteilbezogenen Einheitskoordinaten auf unverzerrte Millimeterkoordinaten. Bei der Skalierung wird die Hälfte der jeweiligen Maße verrechnet, da die Skalierung ausgehend von der Bauteilmitte erfolgt. Im Programm wird hierzu eine anonyme Funktion erstellt, mit welcher sich beliebige Einheitskoordinaten im bauteilbezogenen System auf die Rahmen-Millimeterkoordinaten umrechnen lassen. So lassen sich iterativ alle erfassten Koordinaten umrechnen. Da die Bauteilhöhe in den Metadaten vermerkt werden sollte und per Definition konstant sein muss, sind nun die drei translatorischen Punktkoordinaten-Anteile bestimmt.

Die Rotationswinkelkoordinate $\phi_{\text{Kantenpunkt}}$ wird auf Basis der bereits ermittelten Punktkoordinaten berechnet. Hierzu wird am Ort jeder Punktkoordinate in einem einstellbaren Radius ein quadratischer Bildausschnitt des Referenzobjektes betrachtet. Da das Bild als Binärbild vorliegt, und das Objekt hell sowie der Hintergrund schwarz dargestellt ist, ist an jedem Kantenpunkt bei Berechnung des Gradientenfeldes an dieser Stelle der Kantennormalen-Winkel aus dem Wert des Gradienten-Feldes am jeweiligen Kantenpunkt bestimmbar. In der Praxis wird hier im gesamten Bildausschnitt der mittlere Gradienten-Winkel ermittelt. Abbildung 4.18 zeigt den Verlauf der Gradienten-Vektoren in einem beispielhaften quadratischen Bildausschnitt, wie er im Programm verarbeitet wird.

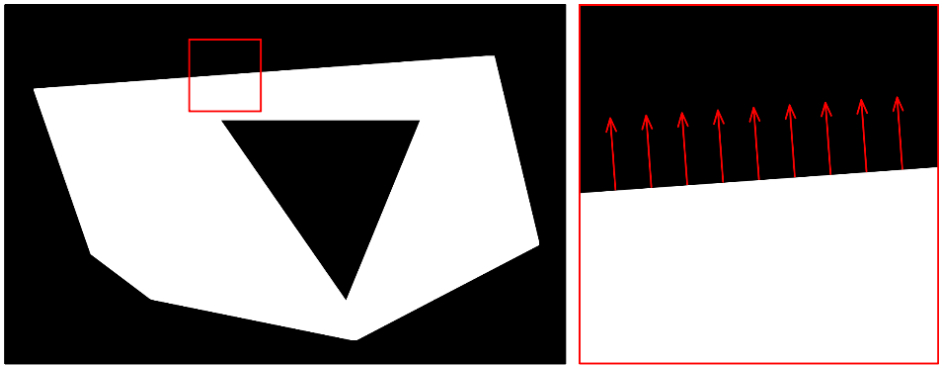


Abbildung 4.18: Gradienten-Feld (rote Pfeile) als Kanten-Normalenvektoren des Objektes im Binärbild

Hierzu werden die Gradienten-Vektoren an jedem Pixel im betrachteten Bildausschnitt zunächst addiert. Anschließend wird von diesem resultierenden Vektor der Rotationswinkel in der Bildebene bestimmt. So muss nicht exakt das Pixel auf dem Kantenübergang bei der Wahl der Messpunkte getroffen werden. Gezählt wird im Uhrzeigersinn ausgehend von der Seite, deren Normalen auf die linke Seite des Referenzrahmens im Kamerabild zeigen. Dies erfolgt in Abstimmung zu Algorithmen aus OpenCV. Der Winkel wird gemäß Gleichung 4.5 berechnet.

$$\phi_{\text{Kantenpunkt}} = \tan^{-1} \frac{\sum_{x,y} (\nabla B)_x}{\sum_{x,y} (\nabla B)_y} \tag{4.5}$$

Hier ist $(\nabla B)_i$ jeweils eine Komponente des Gradienten-Vektors der Bildmatrix B, deren Werte an jedem Pixel des Bildausschnittes aufsummiert werden. Die jeweiligen Summen der Normalenvektor-Komponenten bilden den gemittelten, nicht normierten Normalenvektor im Bildausschnitt. Über den Arkustangens wird der Winkel dieses Vektors in der Bildebene ermittelt.

Sehr große Radien geben wegen der größeren Datenmenge ein genaueres Ergebnis, sind aber anfällig, verfälschte Messungen zu provozieren, da Konturen mit vollkommen anderen Normalen-Winkeln Einfluss auf die Berechnung nehmen können. Bei sehr kleinen Radien muss der Kantenpunkt in der Punktauswahl genauer getroffen werden. Weiterhin können Schräglinien, welche bei sehr kleinen Pixelanzahlen als Treppenlinien dargestellt werden Unregelmäßigkeiten im Gradienten erzeugen. Zur Abschwächung dieses Effektes wird der Bildausschnitt zunächst mit einem 3x3 Gauss-Filter leicht unscharf gemacht, was die Filterung von Treppeneffekten im Bild bewirkt. Der Ausschnitt-Radius ist auch in den Metadaten in der JSON-Datei anpassbar. Im Rahmen der Tests haben sich Radien von rund zehn Pixeln bewährt. Nun liegen vollständige Punktkoordinaten vor, welche der SPS übermittelt werden können. Jeder Punkt wird durch die Reihenfolge seiner Auswahl eindeutig identifiziert und benannt. So können etwaige Fehler bei der Punktauswahl besser nachvollzogen werden. Nach dem Erfassen der Koordinaten werden zudem alle ausgewählten Punkte beim Anzeigen des bearbeiteten Kamerabildes inklusive deren ID mit angezeigt.

Optimierung der Messreihenfolge, Zeitkomplexität

Zunächst liegen die vollständigen Koordinaten ungeordnet in einer Liste vor. Es ist im Allgemeinen sinnvoll, eine Optimierung in der Anfahrreihenfolge vorzunehmen, um den Zeitaufwand zu reduzieren. In der Mathematik ist dieses Problem als *Travelling-Salesman-Problem* bekannt [80]. Ein optimaler Pfad wäre derjenige mit der geringsten totalen Länge. Zum Berechnen dieses Optimums kann eine Matrix mit den Abständen aller Punkte zueinander aufgestellt werden. Für jede Permutation werden alle möglichen Wege und deren Längen berechnet, wobei anschließend das Minimum dieses Datensatzes ausgewählt wird. Für N Elemente gibt es also $N!$ Permutationen, welche individuell berechnet werden müssen[24]. Dies entspräche ebenso der Zeitkomplexität des numerischen Algorithmus zur Lösung dieses Problems, diese ist $O(N!)$. Dieses Verhalten ist bereits für kleine Datenmengen praktisch zu rechenintensiv und skaliert dementsprechend schlecht für viele Messpunkte. Eine Vereinfachung des Algorithmus hätte zwar den Verlust des Minimums zur Folge, jedoch lassen sich stets Reduzierungen des Weges bereits durch einfache Maßnahmen erzeugen.

Als einfache Heuristik wird stets der Punkt ausgewählt, welcher dem aktuellen Punkt am nächsten liegt. Im Allgemeinen liefert dieses Verfahren kein Minimum. Dennoch sind grobe Fehler die Ausnahme und werden selten hervorgerufen. Da für jeden Punkt (N mal) einmal alle übrigen Punkte (N-1 mal), und einmal alle Punkte für den Startpunkt auf minimalen Abstand geprüft werden müssen (N mal) ergibt sich die Zeitkomplexität für diesen Algorithmus nach Gleichung 4.6.

$$O(N \times (N-1) \; + N) = O(N^2) \tag{4.6}$$

Die Änderung des Algorithmus hat bei zehn Messpunkten bei Vernachlässigung der konstanten Faktoren der Komplexitäten bereits eine rechnerische Zeitverbesserung um den Faktor 36288 zufolge. Aufgrund dieser Komplexitätsverbesserung wird die Verschlechterung der berechneten Wegoptimierung akzeptiert, da gegebenenfalls die verlorene Zeit durch eine fehlerhafte Optimierung sogar durch den Zeitgewinn bei der Berechnung ausgeglichen werden kann.

Bewertung der ausgewählten Messpunkte und Datenexport

Für die Bewertung, ob ein Messpunkt an einem Bauteil messbar ist, kann ebenso der Gradient der Bildausschnitte verwendet werden. Wenn keine Bauteilkante im Bild sichtbar ist, ist auch kein Übergang von Schwarz zu Weiß, oder umgekehrt, enthalten. Dementsprechend ist der Betrag des

[24] Jedem Element (N mal) muss eine geordnete Menge ohne Wiederholungen aller verbleibenden Elemente (N-1) angehangen werden. Insgesamt belaufen sich so die möglichen Permutationen auf $N \times (N-1)! = N!$

Gradienten an jeder Stelle null. Daher ist an diesen Stellen kein Winkel ermittelbar, er wird aufgrund der fehlenden Daten zu *None* gesetzt. Wenn also ein Winkel gleich *None* ist, gilt dieser als nicht messbar.

Nach dieser Bewertung werden die Messpunkte zusammengefasst und bei Bedarf in einer CSV-Datei gespeichert. Der Speicherpfad für die Datei kann in der JSON-Referenzdatei festgelegt werden. Exportiert werden nur die Punkte, welche nach der Winkelmessung als messbar gelten. Sollte kein messbarer Punkt festgestellt werden können, wird ein Fehler ausgegeben. Die SPS erhält somit nur Punkte, welche entsprechend der Objekterkennung eine plausible Messung der Kantenradien ermöglichen.

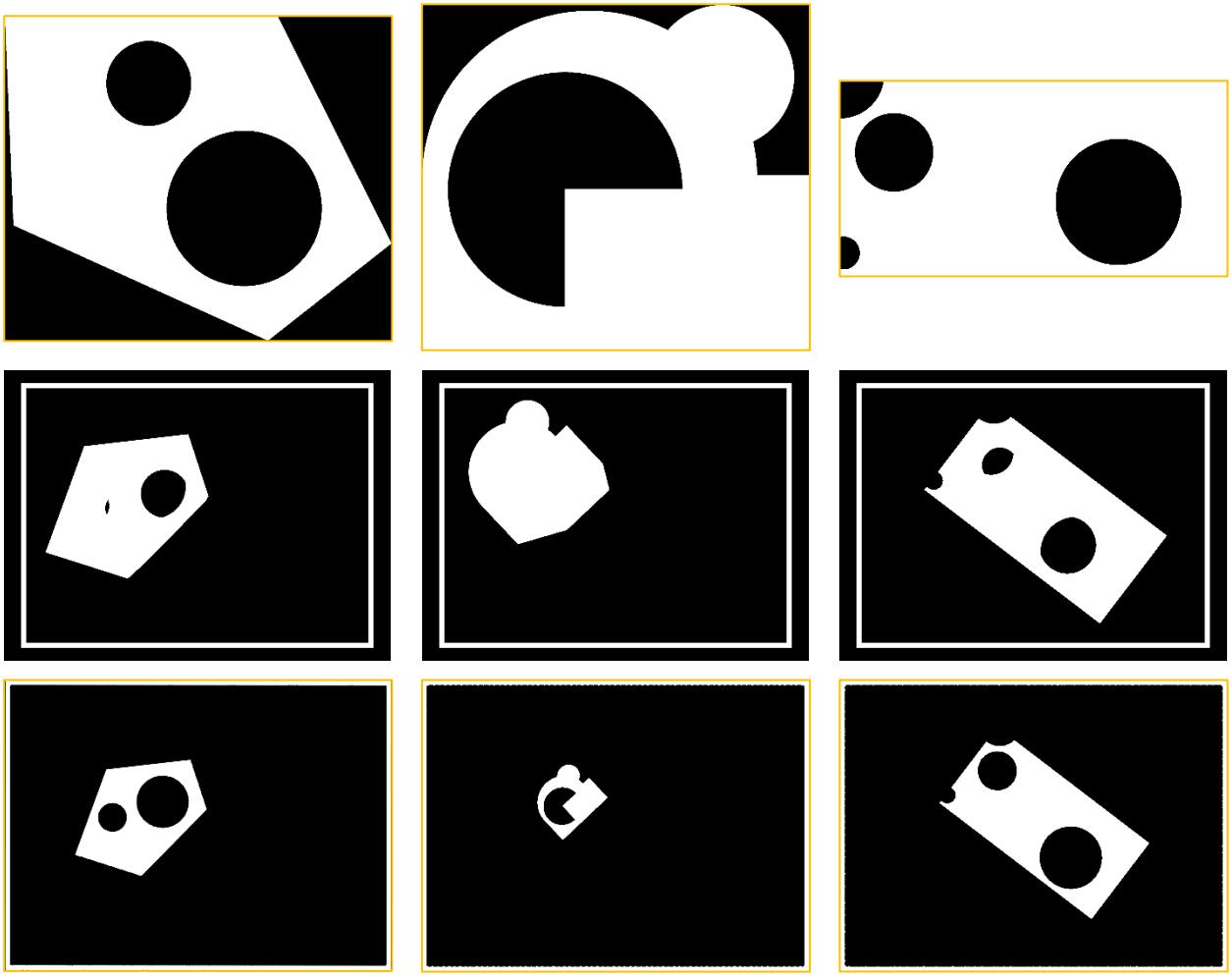
```
order,id,x,y,ang,thickness
1,2,97.11,150.34,145.7,125.0
2,1,70.37,180.96,335.1,125.0
```

Abbildung 4.19: CSV-Exportdatei, Beispiel mit zwei Messpunkten

Test der Bildverarbeitung und Koordinatenberechnung

Um mögliche Fehlerquellen aus der Bilderstellung durch die Kamera auszuschließen, werden zum Test der Bildverarbeitung und Koordinaten-Berechnung verschiedene Kamerabilder künstlich konstruiert, welche eine ideale Darstellung der perspektivischen Verzerrungen enthalten. Die konstruierten Kamerabilder werden wie Abbildung 4.10 gespeichert und stellen ein perfektes binäres Kamerabild dar, welches von Kamerafehlern unabhängig ist. Das Simulieren der perspektivischen Verzerrungen geschieht auf Basis der Erkenntnisse aus den Vorbemerkungen zum Entzerrungsalgorithmus. In Tabelle 4-4 sind drei repräsentative Bauteilgeometrien gelistet, für welche die Bildbearbeitung und Koordinatenberechnung erfolgreich durchgeführt werden konnte. Außer der bekannten Probleme bei der Bildentzerrung wird an allen konstruierten Testbeispielen keine weitere persistente Fehlerquelle festgestellt.

Tabelle 4-4: Referenzbild aus DXF-Datei nach Konvention (oben), simulierte Kamerabilder mit perspektivischer Verzerrung (mitte), Entzerrte Bilder, an Rahmen angepasst (unten)



Programmausgabe/Datendarstellung

Die Messpunktkoordinaten werden über die Methode *log_data* in der Konsole ausgegeben. Die Messpunkte sind anhand der Anfahrreihenfolge sortiert, wobei die Reihenfolge der Auswahl (ID) ebenso berücksichtigt wird, um eine eindeutige Zuordnung der Punkte zu ermöglichen. Die Datenausgabe ist wie folgt in der Konsole formatiert:

```
MEASUREMENT-POINT DATA (12/06/2023 11:32:12):
------------------------------------------------------------------------
|Order:   1 | ID:   3 | x:  163.79mm | y:   56.64mm | Scan-Angle:   None°|
|Order:   2 | ID:   1 | x:  238.10mm | y:  106.48mm | Scan-Angle:  127.3°|
|Order:   3 | ID:   4 | x:  236.06mm | y:  181.24mm | Scan-Angle:   None°|
|Order:   4 | ID:   5 | x:  289.07mm | y:  200.72mm | Scan-Angle:  217.3°|
|Order:   5 | ID:   2 | x:  178.97mm | y:  184.18mm | Scan-Angle:  307.3°|
------------------------------------------------------------------------
Total distance travelled: 505.38mm
Distance (only measurable points): 479.3mm
```

Die nicht messbaren Punkte werden über ANSI-Escape-Sequenzen, die von den meisten Terminalemulatoren unterstützt werden, farbig hervorgehoben, und sind durch den Winkelwert *None*

erkennbar. Alle Zahlenwerte werden auf vier bis fünf signifikante Stellen gerundet. Abschließend wird weiterhin die bereits minimierte Fahrtstrecke inklusive, sowie exklusive nicht messbarer Punkte ausgegeben. Diese stellt einen groben Anhaltspunkt bezüglich der restlichen Prozessdauer dar. Die CSV-Datei der Koordinaten (siehe Abbildung 4.19), welche in die SPS geladen wird, ist eine reduzierte Version dieser Nutzerausgabe. Alle ausgewählten Messpunkte werden inklusive ID im Rahmen der Punktwahl in das entzerrte Kamerabild projiziert. Dieses kann bei Bedarf angezeigt werden.

4.2.5 Steuerung des Portalroboters

Zur Steuerung des Portalroboters wird das in 3.5.3 konfigurierte SPS-System verwendet. Ausgehend vom Computer sollen die soeben berechneten Punkt-Koordinaten auf die SPS exportiert werden. Die SPS ist anschließend in der Lage, im Rahmen dieser Arbeit den kinematischen Prozess an einem digitalen Zwilling im TIA-Portal von Siemens zu simulieren [27]. Da die vier Achsen des Roboters linear unabhängig voneinander sind, ist jede mögliche Lage des Roboters eindeutig bestimmt. Eine uneindeutige Rücktransformation der Koordinaten zur Rückführung auf notwendige Aktor-Stellungen erfolgt in dieser Ausführung des Roboters also nicht. Für die Hardware-Implementierung werden alle konstruktiven Vorkehrungen zur Montage des Roboters mit Integration aller Achsen getroffen.

Kommunikation mit der SPS

Die Python-Bibliothek *Snap7* ermöglicht eine direkte Netzwerkverbindung eines Computers mit Siemens-Steuerungen der *SIMATIC-S7* Serie auf Basis des Siemens S7-Protokolls über Ethernet [81]. Die Bibliothek ermöglicht ein direktes Schreiben von Binärdaten auf Speicherblöcke der SPS. Der Inhalt dieser Speicherblöcke kann auf Basis der Programmierung der SPS interpretiert werden, so zum Beispiel auch als Punkt-Koordinaten, beziehungsweise einer Soll-Stellung der Kinematik. Für die Kommunikation zwischen SPS und Computer wird ein eigener Datenblock der SPS genutzt. Der Rechner besitzt auf allen Registern dieses Datenblockes sowohl Lese- als auch Schreibberechtigung. Im Programmcode gehen die Kommunikation mit der SPS und dem Laser-Scanner miteinander einher. Zur Plausibilisierung der SPS-Implementierung und des Programmablaufes wird auf die Implementierung des Laser-Scanners vorgegriffen[25]. Der vollständige Messprozess wird wie folgt im Programmcode abstrahiert:

[25] Für die vollständige Implementierung des Laser-Scanners siehe 4.2.6 (nachfolgend)

```
PLC = cm.PLC()
SCANNER = cm.LaserScanner()
cm.exec_measurement(PLC, SCANNER)
SCANNER._print_results()
SCANNER._end_session()
PLC._end_session()
```

Zu Beginn werden ein *PLC* (engl. SPS) und ein *LaserScanner*-Objekt instanziiert. Bei der Instanziierung werden die Netzwerkverbindungen hergestellt. Das PLC-Objekt importiert ebenso die Punktkoordinaten aus der Export-Datei entsprechend Abbildung 4.19. Die Punktkoordinaten werden für jeden Punkt als Liste gespeichert. Für jede Komponente der Koordinaten wird außerdem eine Speicheradresse im Kommunikations-Datenblock der SPS gespeichert. Die Koordinaten-Komponenten liegen in 32-Bit Fließkommazahlen vor. Daher werden für jeden Messpunkt 16 Byte (32-Bit mal 4) Speicherplatz vorgesehen.

Außerdem wird ein Byte-Register zur Statusübertragung an den Computer genutzt. Aus einem Byte ließen sich perspektivisch bis zu 256 unterschiedliche Systemstatus codieren, hier steht der Registerwert 0x00 für „geladene Koordinate wird angefahren“, 0x01 steht für „Koordinate erreicht, lade nächste Koordinate“. Über die Funktion *cm.exec_measurement(PLC, SCANNER)* wird die Punktmessung ausgeführt. Die Messpunkte werden generell nacheinander in der optimierten Reihenfolge (in der CSV-Datei v. o. n. u.) auf die SPS geladen. Zu Beginn wird ein Punkt als *Bytearray* in Python binär codiert, und auf die SPS geschrieben. Das Status-Register wird auf 0x00 gesetzt, und die SPS steuert die Kinematik zur gewünschten Koordinate. Während des Manövriervorganges fragt der Computer in einem flexiblen Intervall den Status der SPS ab. Sobald dieser beim Erreichen des Messpunktes von der SPS auf 0x01 gesetzt wird, wird eine Punktmessung am Laser-Scanner ausgelöst. Anschließend wiederholt sich der Vorgang für jeden Punkt, welcher aus der CSV-Datei importiert wurde. In diesem Prozess ist eine optionale Timeout-Bedingung wählbar, welche den Messprozess abbricht, wenn die Timeoutzeit zwischen dem Erreichen zweier Messpunkte überschritten wird. Abbildung 4.20 illustriert den beschriebenen Messprozess anhand eines PAP.

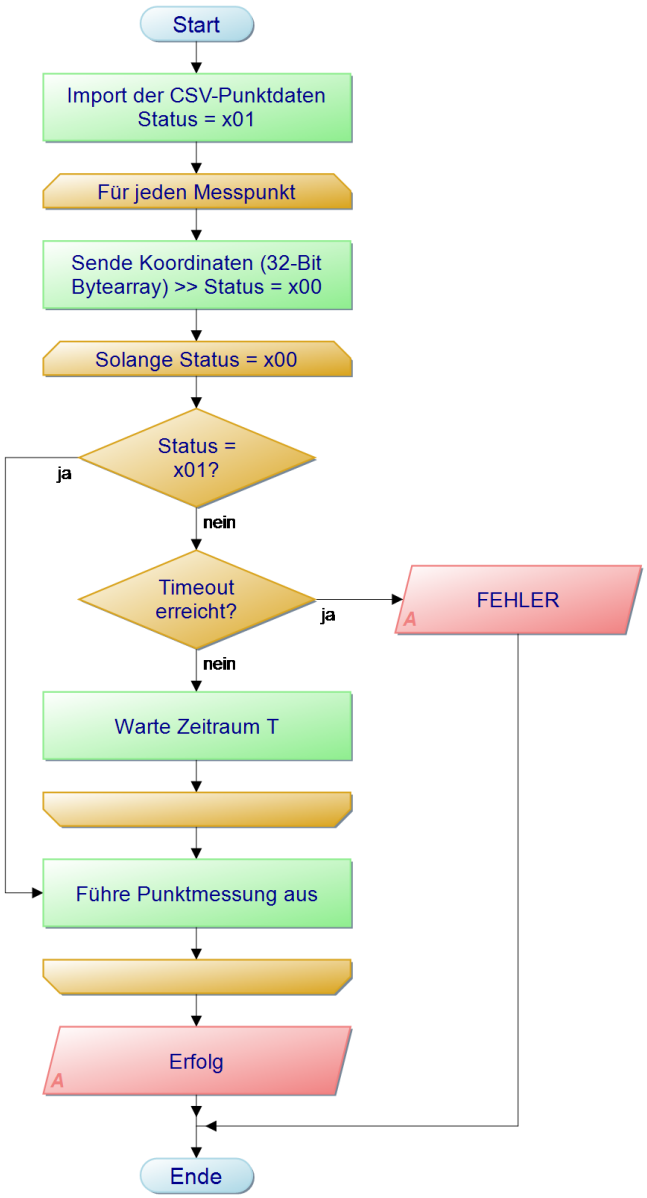


Abbildung 4.20: PAP Messprozess SPS/ Laser-Scanner seitens des Computers

Simulation der Kinematik / Programmierung der SPS

Für die Simulation wird bereits das finale Programm-Skript auf die SPS geladen. Im Siemens TIA-Portal wird dieses nach IEC 61131 (siehe 2.3.1) in Skriptform formuliert. Anschließend wird ein digitales Modell der Kinematik mit den entsprechenden geometrischen Eigenschaften (Achslänge, Manövrierbereich) erstellt, welches nun entsprechend der Steuerungslogik verfahren werden kann. Während Abbildung 4.20 den Kommunikations- und Programmablauf seitens des Computers beschreibt, ist in Abbildung 4.21 der Programmablauf der Steuerungslogik der SPS dargestellt.

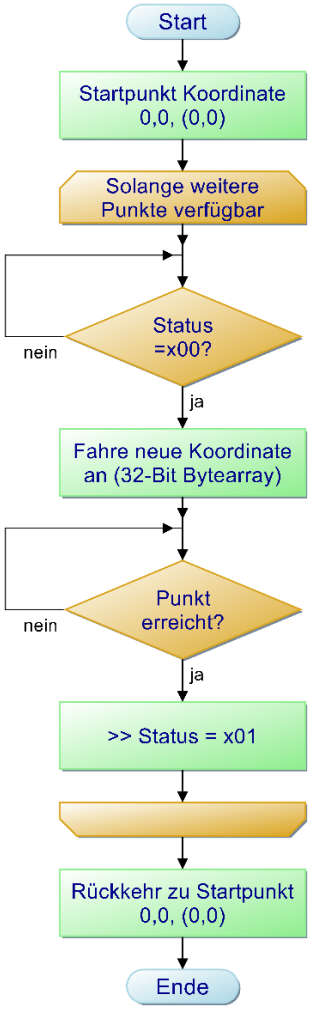


Abbildung 4.21: PAP seitens der SPS

Abbildung 4.22 zeigt die Ergebnisse einer Simulation im Siemens TIA-Portal, wobei die zu vermessenden Punkte die Eckpunkte der polygonalen Bahnlinie (rot) sind.

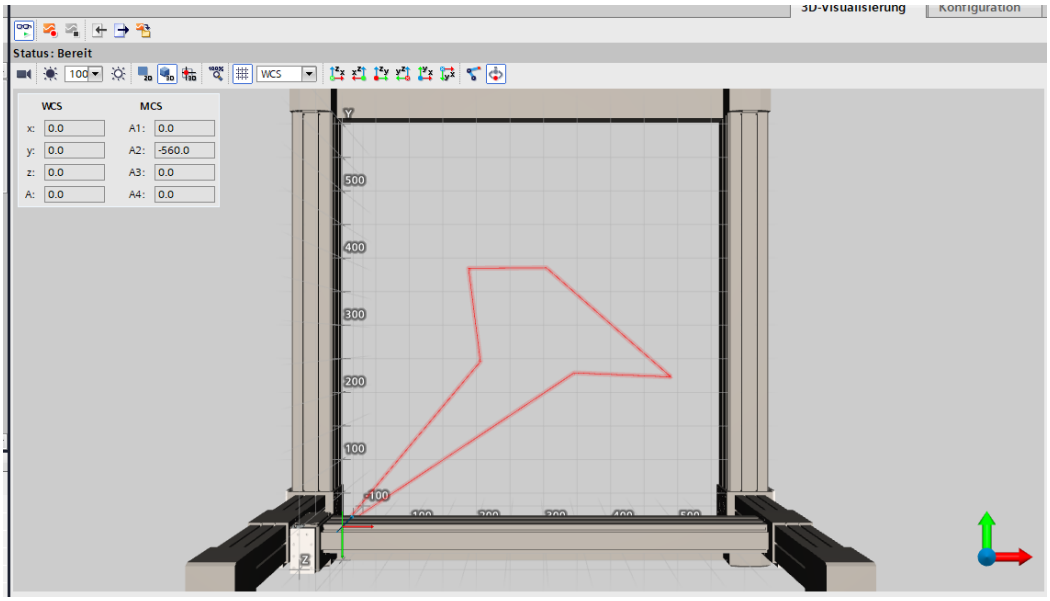


Abbildung 4.22: Ansicht der Simulation im Siemens TIA-Portal

Die Koordinaten der Messpunkte werden aus Registern der SPS geladen. Ebenso werden drei Status-Bits vorgesehen, je für die Freigabe des Prozesses, für das Anfahren des nächsten Punktes sowie für das Abschließen der Messung.

4.2.6 Laser-Scanner

Zur Durchführung der Messungen wird ein Linienprofil-Laser-Scanner entsprechend des Aufbaus nach 3.5.1 verwendet. Die Messeinstellungen können über die Webserver-basierte GUI des Laserscanners im Voraus gewählt werden. Diese Einstellungen werden auch nach einem Reboot des Laser-Scanners beibehalten. In der Anwendung können beliebige Algorithmen, welche vom Hersteller zur Verfügung gestellt werden, direkt auf dem Laser-Scanner berechnet werden, und Messergebnisse exportiert werden.

In der Praxis soll der Laser-Scanner von der Kinematik nacheinander an alle zu vermessenden Punkte auf der Oberseite des Blechbauteils manövriert werden. An jedem dieser Messpunkte soll eine Punktmessung des Kantenprofils vorgenommen werden. Die SPS kann bereits eine eindeutige Rückmeldung über die Position der Kinematik geben. Daher beschränkt sich die notwendige Kommunikation zwischen Computer und Laser-Scanner im laufenden Betrieb auf Trigger-Signale zum Erfassen des Kantenprofils und der Messwerte und Statusmeldungen des Scanners. Der Laser-Scanner liefert standardmäßig eine simple Schnittstelle für die Kommunikation über das ASCII-Protokoll. Diese basiert auf TCP/IP über Ethernet und ist im Vergleich mit den anderen verfügbaren Schnittstellen funktionsärmer[26], aber für die Anwendung ausreichend, da die anderen Protokolle häufig für die direkte Kommunikation mit einer SPS vorgesehen sind.

Kommunikation mit dem Laser-Scanner

Zum Starten der TCP/IP-Sitzung wird über die Python-Standardbibliothek *socket* eine Verbindung mit dem Laser-Scanner unter Angabe der IP-Adresse des Laser-Scanners mit spezifiziertem Port[27] hergestellt. Im Programm wird hierzu ein *LaserScanner*-Objekt instanziiert, welches die Verbindung zum Scanner bis zur expliziten Trennung dieser aufrecht erhält. Anschließend können zum Zeitpunkt der Punktmessungen nach Anfahren der Koordinaten durch die Kinematik über einen Befehl eine Profil-Messung vorgenommen und an den Rechner geschickt werden. Die Antworten des Laser-Scanners sind ebenso als ASCII-Code formatiert. Als alternatives Kommunikationsprotokoll ist Telnet

[26] Alternativen wären das herstellereigene *Gocator-Protokoll*, *Modbus*, *Ethernet/IP* und *PROFINET* über Ethernet [82].

[27] Standardmäßig für den Laser-Scanner IP: 192.168.1.10, Port: 8190

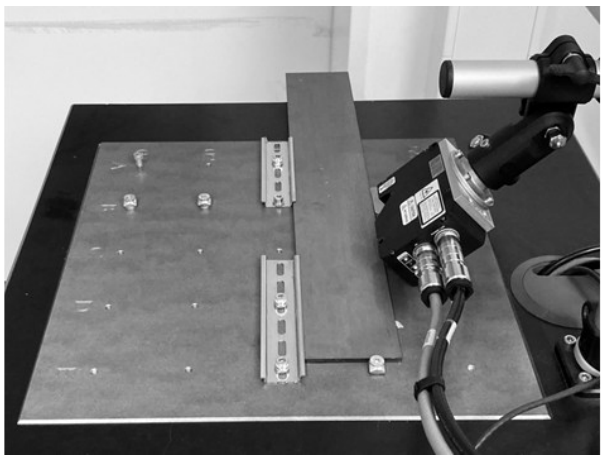
denkbar, wobei die Kommunikation auch auf diesem Weg in *PuTTy* erfolgreich verläuft. In der Praxis wird der Messprozess durch die folgenden Befehle im Programmcode realisiert:

```
SCANNER = cm.LaserScanner()
SCANNER.measure()
SCANNER.end_session()
```

Zu Beginn wird ein *LaserScanner*-Objekt instanziiert, mit welchem anschließend an jedem Punkt mit der Methode *measure* die Messungen durchgeführt werden können. Abschließend wird, analog zum Bildgebungsprozess, die TCP/IP-Sitzung beendet. Die Kommunikation läuft im praktischen Betrieb beinahe latenzfrei, wodurch keine längeren Pausen am Punkt der momentanen Messung eingeplant werden müssen. Aus dem vorigen Abschnitt, insbesondere Abbildung 4.20, ist ersichtlich, dass die Kommunikation zwischen Laser-Scanner, SPS und Computer gleichzeitig verläuft. Dies ist, wie zuvor erörtert, notwendig um abwechselnd Befehle an die Komponenten zu senden, sowie um die Netzwerk-Kommunikation immer über den Computer verlaufen zu lassen.

Zum Test des Laser-Scanners wird ein reduzierter Aufbau genutzt, bei welchem der Laser-Scanner entsprechend der Anforderungen fest in einem Anstellungswinkel von 45 Grad montiert wird. An einer Schiene können nun die Kanten von Blechbauteilen in einer optimalen Ausrichtung zum Messbereich des Laser-Scanners vermessen werden. So können Ausrichtungsfehler der Kinematik zum Test des Laser-Scanners ausgeschlossen werden. Dieser Aufbau dient zur Demonstration der verwendeten Messalgorithmen zum Ermitteln des Kantenradius.

Laser-Messstation mit starr montiertem Laser-Scanner, Messobjekt und Bauteilführung am Prüfstand



Nahaufnahme der Positionierung des Laaser-Scanners relativ zum Blechbauteil mit skizziertem Linien-Messbereich

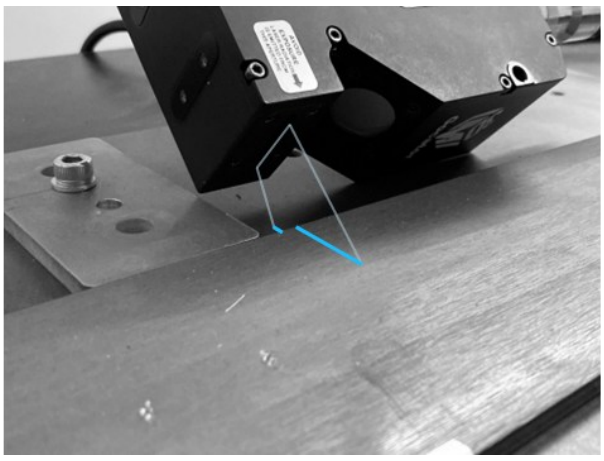


Abbildung 4.23: Prüfstand zum isolierten Testen des Laser-Scanners

4.2.7 Erstellen und Verarbeitung von Punktmessungen

Mit dem Framework zur Kommunikation mit dem Laser-Scanner ist die Wahl des Messalgorithmus, sowie die Wahl der Messgröße flexibel. Im Rahmen dieser Arbeit ist es das Ziel, eine eindeutige

Aussage über den Rundungsradius der Oberseitenkante eines Blechbauteils treffen zu können. Der Rundungsradius eines Kantenprofils kann, wie in 2.2 erörtert, mit einem Gauß-Fit eines Kreises im Kantenbereich approximiert werden. In der Praxis ist die Wahl des Messalgorithmus allerdings flexibel, je nach Bedarf kann hierzu die Software des Laser-Scanners genutzt werden.

Implementierung des Messalgorithmus

Für das Erstellen eines Gauß-Fits im korrekten Kantenbereich muss ein Bereich der Messung für das Anlegen des Kreisbogens bestimmt werden. Hierzu benötigt der Scanner einen festgelegten rechteckigen Bereich, welcher dynamisch in Abhängigkeit der Messung ermittelt werden kann. Aufgrund von Ungenauigkeiten der Positionierung, sowie des kleinen relevanten Bereiches für die Messung, kann der Bereich nicht statisch vorgegeben werden. Idealerweise wird zur Bestimmung des Kantenradius ein Teil der Kante betrachtet, an welchem kein Übergang zur flachen Bauteilfläche erkennbar ist. Abbildung 4.24 zeigt die Ansicht in der Webserver-basierten Oberfläche des Laser-Scanners.

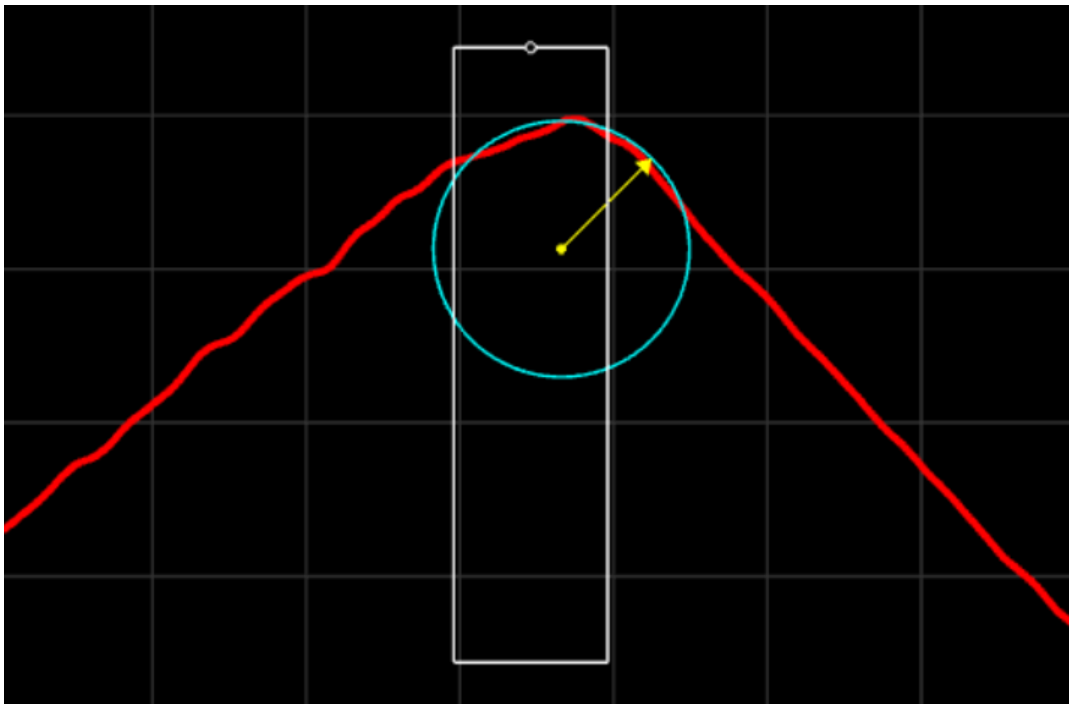


Abbildung 4.24: Laser-Scanner-Messung (rot) mit visualisierten Auswertungsalgorithmen

Um den Messbereich (grau) in Abhängigkeit der jeweiligen Punktwolke zu definieren, wird zunächst der Schnittpunkt der Oberflächentangenten des Bauteils bestimmt. Unter Annahme einer perfekt symmetrischen Kante mit Kantenwinkel von 90 Grad bildet dieser Punkt den Kantenpunkt für einen Kantenradius gleich Null. Da der Radius allerdings nach Entgratung als strikt konvex vorausgesetzt wird, ist ein Kantenradius größer Null garantiert und der Schnittpunkt der Oberflächentangenten bildet das obere Maximum des Messbereichs. Mit steigender Abrundung der Kante flüchtet diese in der gekippten Ansicht aus Abbildung 4.24 nach „unten“, sodass das untere Maximum des

Messbereichs mit hinreichender Toleranz ausgedehnt ist. Für die Ausdehnung zur Seite liefern 0,05 mm nach links und rechts die robustesten Ergebnisse. In dem vollständig definierten Messbereich kann ein Kreis durch Gauß-Fit angelegt werden (blau). Der Radius dieses Kreises (gelb) wird anschließend nach einem Trigger-Signal des Computers über das Netzwerk versendet.

Umgang mit Ungenauigkeiten

Fehler in der Auswertung können entstehen, wenn die gemessene Kontur nicht strikt konvex ist, beziehungsweise Grate besitzt. In diesem Falle liefert der Algorithmus Fehlermeldungen und das Bauteil kann als fehlerhaft identifiziert werden.

Der Algorithmus zur Kantenberechnung funktioniert ordnungsgemäß, sobald sich die zu vermessene Kante vollständig im Messbereich des Laser-Scanners befindet. Fehler in der Ausrichtung des Laser-Scanners durch die Koordinatenberechnung oder Manövrier-Ungenauigkeiten der Kinematik können in Normalrichtung zur Kante verlustfrei kompensiert werden. Die angeforderte maximale Manövrierungenauigkeit des Gesamtsystems von 1 mm nach 3.3 liegt im Toleranzfeld des Messbereichs des Laser-Scanners nach Abbildung 3.5. In Kantenrichtung kann eine fehlerhafte Ausrichtung dann mit minimalen Verlusten kompensiert werden, wenn der falsch angefahrene Punkt dieselbe Orientierung der Kante besitzt.

Da die Messergebnisse gerade bei einem Bauteil mit schlechten optischen Eigenschaften fluktuieren können, werden an jedem Messpunkt mehrere Messungen vorgenommen. Die Anzahl der Einzelmessungen ist frei wählbar und nach Abschluss aller Einzelmessungen an einem Punkt wird der Durchschnitt dieser Messungen gebildet, da die Schwankungen als normalverteilt angenommen werden.

Programmausgabe/Datendarstellung

Die Messergebnisse des Laser-Scanners werden in Form der Kantenradien an den jeweiligen Messpunkten über die Methode *log_results* in der Konsole ausgegeben. Zuzüglich der Radien wird die Reihenfolge explizit aufgeführt, um nachträgliche Rückschlüsse auf die Punktauswahl zu vereinfachen. Die Ausgabe in der Konsole ist wie folgt formatiert:

```
LASER-SCANNER RESULTS (13/06/2023 14:25:57):
--------------------------------
|Order:  1 | Radius ->   0.45mm|
|Order:  2 | Radius ->   0.63mm|
|Order:  3 | Radius ->   0.57mm|
|Order:  4 | Radius ->   0.49mm|
--------------------------------
```

4.2.8 Konstruktive Vorkehrungen

Zur Montage des Demonstrators müssen einige konstruktive Vorkehrungen getroffen werden. Wie in der Auslegung des Portalroboters beschrieben, wird der Laser-Scanner über eine zusätzliche rotatorische Achse an der Kinematik montiert. In Abbildung 4.25 ist dargestellt, wie die Komponenten gefügt werden. Kritisch sind hier die Montage des Schrittmotors der rotatorischen Achse an der Kinematik, sowie die Montage des Laser-Scanners am Schrittmotor.

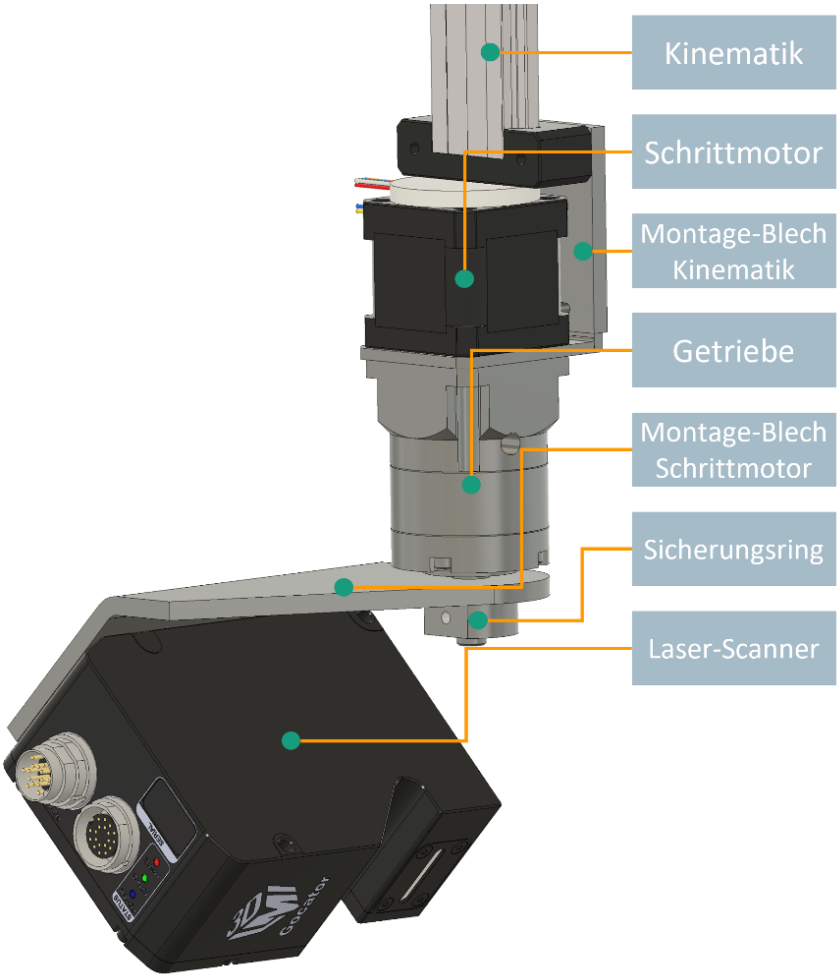


Abbildung 4.25: Detail-Ansicht Laser-Scanner Montage an der Kinematik im CAD-Modell

Da beiden Montagebleche nicht durch Normteile ersetzbar sind, werden diese aus 3 mm dickem Aluminiumblech durch Wasserstrahlschneiden gefertigt. Für die Fertigung der Teile müssen lediglich DXF-Profile der Bauteilkontur vorliegen. Nach dem Ausschneiden werden die Teile zwecks Entgratung gebürstet und die Bohrungskanten geräumt. Das Montage-Blech zwischen Motor und Laser-Scanner wird zusätzlich um 45 Grad abgekantet, um die korrekte Anstellung des Laser-Scanners zu gewährleisten. Die fertigen Teile sind in Abbildung 4.26 abgebildet[28].

[28] Die CAD-Daten der gefertigten Bauteile sind im Anhang enthalten

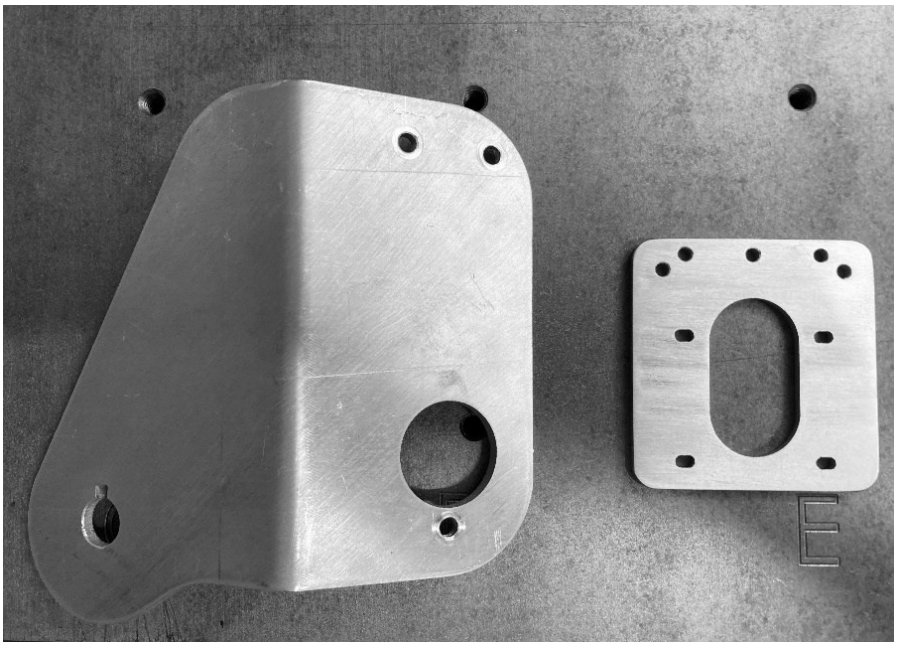


Abbildung 4.26: Montagebleche zur Montage des rotierbaren Laser-Scanners

Um mögliche hochfrequente Eigenfrequenzen zu dämpfen, wird zwischen Montageblech und dem Laser-Scanner auf der gesamten Fläche eine Gummimatte eingelegt. Eine Sichtprüfung des Schwingverhaltens durch Impulsanregung dieses Aufbaus bei fest montiertem Adapterblech zeigt kein problematisches Ausschwingverhalten. Sollten derartige Probleme in der Anwendung auftreten, muss eine Anpassung der Beschleunigung seitens der Kinematik oder eine Pause beim Erreichen des Messpunktes zum Ausschwingen des Systems vorgesehen werden.

Ein voll funktionstüchtiger Prototyp des Demonstrators, wie er in Abbildung 3.11 abgebildet wird, liegt zum Zeitpunkt dieser Arbeit noch nicht vor. Ausgehend von der prototypischen Konstruktion und den funktionalen Anforderungen wird eine überarbeitete Version des Demonstrators von der Boeck GmbH konstruiert. Hierbei werden zusätzliche Anforderungen integriert und ein Aufbau entsprechend Abbildung 4.27 entwickelt. Dieser Aufbau besitzt nebst optischer Überarbeitung eine bauraumsparende vertikale Schiebetür (vorne), welche elektrisch betrieben wird, sowie einen *Industrie-PC* (rechts), welcher einen vollintegrierten Betrieb des Demonstrators zulassen wird. Dieser Computer wird perspektivisch den extern angeschlossenen Computer ersetzen.

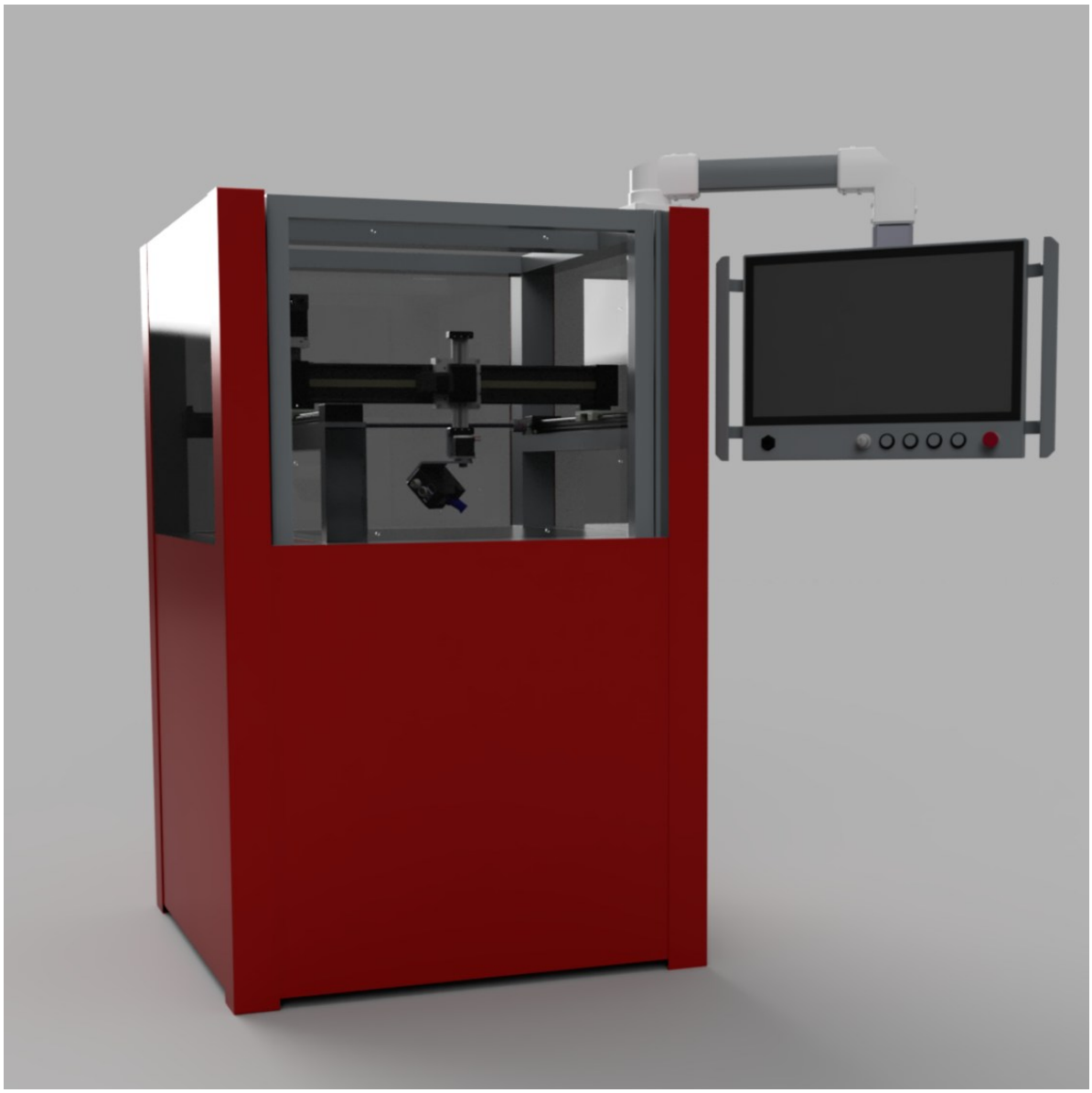


Abbildung 4.27: Finaler Stand der praktischen Ausarbeitung (3D-Render)

4.3 Zusammenfassung der praktischen Implementierung

Mit Hinblick auf den anschließenden Ausblick werden im Folgenden die Ergebnisse der praktischen Implementierung zusammengefasst und hinsichtlich der Validierung dieser Arbeit analysiert. Betrachtet werden die abgeschlossenen prototypischen Systemtests, sowie die anstehenden Schritte der Entwicklung, welche nicht im Rahmen dieser Arbeit behandelt werden.

Prototypische Implementierungen

Die Software, welche für den Betrieb des Demonstrators notwendig ist, konnte funktional fertiggestellt werden, wobei für einige Beispiele bereits erfolgreich die Objekterkennung und Koordinatenberechnung im Rahmen der Bildverarbeitung durchgeführt werden konnten. Die Kommunikation mit

den einzelnen Komponenten über Ethernet mit entsprechenden Protokollen wurde ebenfalls erfolgreich getestet. Ausführliche Modultests der Software konnten noch nicht erfolgen. Die erfolgreichen Programmtests ohne spezifische Programmanpassungen für die praktischen Tests lassen aber keine grob fehlerhafte Programmstruktur vermuten.

Im Entwicklungsprozess werden weiterhin Methoden implementiert, welche Verfahrenstests ermöglichen. Zum Test der Bildentzerrung werden digitale Kamerabilder mit und ohne Verzerrungen konstruiert. Nach dem Entzerren des zuvor verzerrten Bildes können diese Bilder auf Äquivalenz geprüft werden[29]. Die pixelweise logische XOR-Verknüpfung der Bilder zeigt die Unterschiede (XOR entspricht NOT EQU) beider Abbildungen und erweist sich als gutes Maß zur Fehlersuche beim Test der Algorithmen.

Die objektorientierte Implementierung der Software vereinfacht das Integrieren komplexer Prozeduren in den bestehenden Programmablauf. Zu Beginn der Entwicklung wurde ein paradigmatisch funktionaler Ansatz verwendet, welcher aufgrund der Vielzahl an veränderlichen Werten verworfen wurde. Das Programm besitzt eine eindeutige Anwendung, dennoch war das Ziel der Programmierung das Herstellen eines allgemeinen Frameworks, welches ebenso eine sinnvolle Basis für ähnliche Programme liefert.

Inbetriebnahme der Software

Zum Ausführen des Programms müssen alle Abhängigkeiten entsprechend 4.1 installiert sein. Das Arbeitsverzeichnis (engl. Working Directory) des Shell-Terminals, welches das Programm ausführt, muss dem der *init.py*-Datei entsprechen. Das zu vermessende Blechbauteil muss als DXF-Datei im entsprechenden Ordner[30] vorliegen, sowie in der JSON-Referenzdatei zum Vermerk der Bauteildicke enthalten sein. Wenn diese Voraussetzungen gelten, muss bei der Instanziierung des Setup-Objektes (*MetaDataParser*) der entsprechende Referenzdatei-Name im Programmcode angegeben werden. Anschließend kann das Programm auf beliebigem Wege ausgeführt werden.

Aktuell beschränkt sich die Schnittstelle zur Software auf das ausführende Terminal, sowie die grafische Ausgabe zur Anzeige von Bilddaten von OpenCV. Alle berechneten Messdaten werden mit einem Zeitstempel versehen und in einer Logdatei gespeichert. So können die Daten festgehalten werden und für spätere Nutzung aufbereitet werden. Die Programmausgabe inklusive aller angezeigten Bilder kann in einer GUI realisiert werden. Wie zu Beginn dieses Kapitels bereits erwähnt, kann

[29] Implementiert als Methode *comp_to_template* für Bild-Objekte mit jeweiligen Vergleichsdaten als Eingangsparameter

[30] Derartige lokale Abhängigkeiten lassen sich zum Großteil in der JSON-Referenzdatei spezifizieren.

diese über ein Webserver-basiertes Jupyter-Notebook realisiert werden. Dies würde weiterhin das Auslagern der Software in einen Container ermöglichen.

4.4 Demonstration anhand eines Beispiels

An einem praktischen Beispiel wird im Folgenden die Funktionalität des Demonstrators von der Erstellung des Bildes bis zur Berechnung und dem anschließenden Export der Koordinatendaten auf die SPS gezeigt. Die bisher diskret behandelten Programmschritte werden nun fortlaufend abgearbeitet, um den praktischen Betrieb des Demonstrators zu simulieren. Für diese Demonstration wird der Versuchsaufbau aus 4.2.2 beziehungsweise Abbildung 4.4 genutzt.

Zur Demonstration wird ein Stahlbauteil verwendet, welches 120 mm breit, 50 mm lang und 60 mm hoch ist. Trotz der größeren Höhe als Länge genügt dieses Bauteil der hier geltenden Definition für messbare Bauteile. Die Höhe dient hier zur Demonstration der Entzerrung für die Koordinatenberechnung. In Abbildung 4.28 sind das kalibrierte Kamerabild, sowie das vollständig bearbeitete Bild für den Beispieldurchlauf abgebildet. Auf dem rein kalibrierten Kamerabild sind die der optischen Achse zugewandten Seitenflächen deutlich erkennbar. Der Vorher-Nachher-Vergleich zeigt die Funktionalität der Entzerrung.

Kamerabild, kalibriert (erkennbare Einschnürung an den Seitenmitten)

Kamerabild, binär, auf Rahmen angepasst, entzerrt (minimale Kalibrierimperfektionen an den Rändern)

Abbildung 4.28: Kamerabilder roh (links) und vollständig bearbeitet (rechts)

Die Daten der Objekterkennung werden analog zu 4.2.3 im Terminal angezeigt:

```
OBJECT DATA -> IMAGES/reference26.png (15/06/2023 12:53:32):
-------------------------------------------------------------------------------
|angle: 71.4563deg| Coordinates x: 98.4045mm| y: 185.605mm
|Frame -> width: 118.913mm| height: 49.5639mm|
-------------------------------------------------------------------------------
```

Im Programm wurden die horizontalen Abmaße des Bauteils auf 118,9 mm (b) und 49,6 mm (l) berechnet. Dies beläuft sich auf eine Abweichung von rund 1 % der tatsächlichen Bauteilabmaße.

Die Abweichung ist durch Imperfektionen der Kamera-Kalibrierung (siehe 4.2.2), sowie durch Datenverluste durch Rauschfilter (siehe 4.2.3) zu erklären.

Im Zuge der Koordinatenberechnung werden im Voraus zwei Koordinaten in der JSON-Referenzdatei bestimmt, welche vom Laserscanner vermessen werden sollen. In Abbildung 4.29 sind diese zwei Koordinaten im vollständig bearbeiteten Kamerabild dargestellt. Hierzu erfolgt die Koordinatentransformation vom bauteilbezogenen System für die Wahl der Messpunkte zum rahmenfesten System für die Kinematik (siehe 4.2.4). Da die vorgegebenen Koordinaten entsprechend der hiesigen Koordinaten- und Referenzdatenkonvention festgelegt werden (siehe 4.2.3), lassen sich nun Parameter wie der Ausrichtungswinkel des Bauteils nach der Objekterkennung plausibilisieren[31].

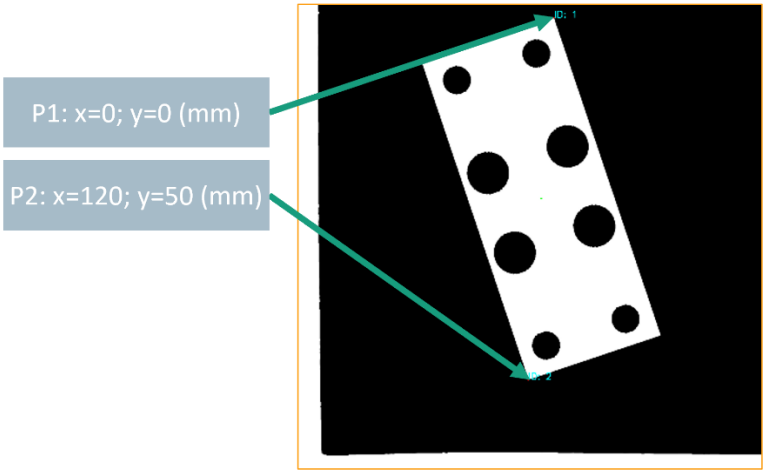


Abbildung 4.29: Messpunkte, relativ zu Referenzkoordinaten

Für diese Messpunkte werden analog zu 4.2.4 folgende Koordinaten ausgegeben:

```
MEASUREMENT-POINT DATA (15/06/2023 12:53:32):
-------------------------------------------------------------------------
|Order:   1 | ID:   1 | x:  102.95mm | y:  121.31mm | Scan-Angle:  116.5deg|
|Order:   2 | ID:   2 | x:   93.78mm | y:  249.78mm | Scan-Angle:  296.5deg|
-------------------------------------------------------------------------
Total distance travelled: 287.9mm
Distance (only measurable points): 287.9mm
```

Die Richtigkeit der berechneten rotatorischen Koordinate lässt sich direkt herleiten. Diese entspricht lediglich dem totalen Bauteilwinkel zuzüglich des Kantennormalen-Winkels in der Referenzabbildung. Dieser beträgt im Mittel an den Ecken des Rechtecks $\phi_{\text{Kantenpunkt}} = 45° + N \times 90°;\ N \in \mathbb{N}$. Daraus folgen die Messpunkt-Winkel entsprechend Gleichung 4.7.

$$\phi_{P1} = 71{,}5° + 45° = 116{,}5°;\ \phi_{P2} = 71{,}5° + 45° + 180° = 296{,}5° \tag{4.7}$$

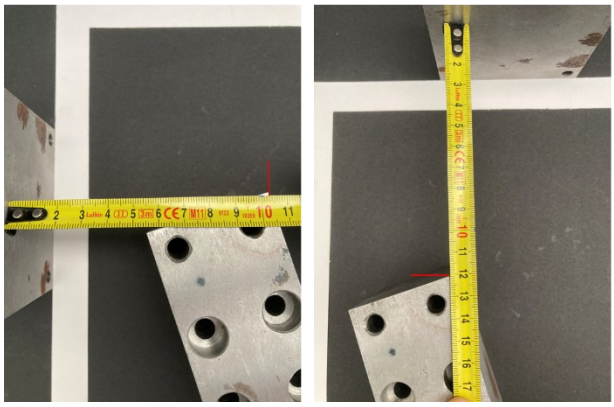
[31] Dies ist gerade für spiegel- oder rotationssymmetrische Bauteile interessant, da die Lage auf der Messfläche in diesem Falle nicht eindeutig ist.

Die tatsächlichen horizontalen Koordinaten der ausgewählten Messpunkte werden manuell geprüft und mit den algorithmisch ermittelten Koordinaten abgeglichen. Die manuellen Messergebnisse sind in Tabelle 4-5 vermerkt, sowie ergänzend in Abbildung 4.30 abgebildet.

Tabelle 4-5: Tatsächliche Koordinaten der Messpunkte (manuelle Messung)

P1	x: 102,5 mm	y: 121 mm
P2	x: 93,5 mm	y: 250 mm

Messwerte P1: x links, y rechts



Messwerte P2: x links, y rechts

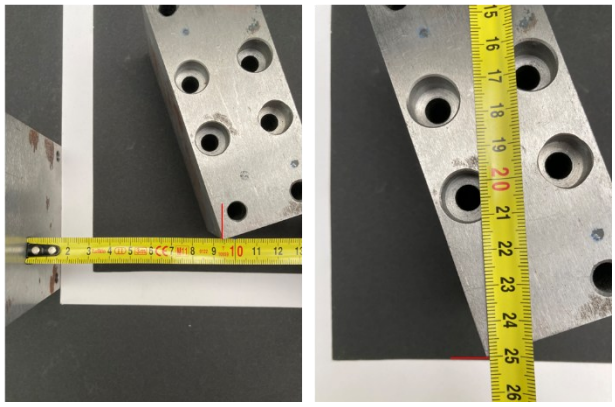


Abbildung 4.30: Tatsächliche Lage der ausgewählten Messpunkte (händische Messung)

Beurteilung der Beispielergebnisse

Im aufgezeigten Beispiel weichen die Ergebnisse von manueller Messung und tatsächlicher Messung um 0,55 mm für den ersten Messpunkt, sowie um 0,36 mm für den zweiten Messpunkt ab. Zuzüglich der korrekten rotatorischen Koordinate liefert der Algorithmus hier Koordinaten mit einer Genauigkeit, welche komfortabel innerhalb der spezifizierten Systemanforderungen liegt.

Die noch existierenden Abweichungen lassen sich auf Rundungsfehler durch die pixeldiskrete Abbildung der Bilddaten sowie auf die Ungenauigkeiten des Kamera-Systems zurückführen. Beim isolierten Testen der Bildverarbeitungsalgorithmen mit konstruierten Kamerabildern konnten Ungenauigkeiten in den Abbildungen von unter 2 Pixeln erzielt werden.

Unter Bezugnahme auf das Datenblatt der Kinematik wäre ein Manövrieren des Gesamtsystems mit Genauigkeit innerhalb der geplanten Anforderungen möglich. Die Funktionalität der Portalkinematik und der SPS in der Praxis wird an dieser Stelle vorausgesetzt.

5 Ergebnisse der Entwicklung, Perspektive & Ausblick

Die Funktionalität des Demonstrators beschränkt sich, wie in den Systemanforderungen formuliert, auf die Darstellung und Verifizierung der Technologie. Ausgehend von dem System-PAP nach Abbildung 3.3 und dem System-Blockschaltbild Abbildung 4.1 konnte bereits im Rahmen dieser Arbeit hinreichend verifiziert werden, dass die Funktionalität der entwickelten Technologie sichergestellt ist, auch wenn der vollständige Programmablauf nicht abschließend in der Praxis getestet werden konnte.

5.1 Zusammenfassung der Entwicklung, Bedienung des Demonstrators

Im Rahmen dieser Arbeit wurde ein Demonstrator entwickelt, welcher in der Lage ist, bei frei positionierten Blechbauteilen entsprechend der Definition aus 2.1 die Kantenverrundung an diskreten Punkten zu messen. Die Messpunkte sind flexibel während des Messprozesses, oder für jedes Bauteil im Voraus wählbar, wobei für jedes Bauteil ein Referenzdatensatz in Form einer DXF-Datei vorhanden sein muss. Das Grundkonzept der Entwicklung ist der Robotik zuzuordnen, in der Ausarbeitung ergeben sich jedoch Elemente von Machine Vision als thematische Schwerpunkte.

Es wurde ein Framework erstellt, welches diese Funktionen zur Verfügung stellt. Die Kommunikation zwischen Rechner und den einzelnen Komponenten (Kamera, Laser-Scanner, SPS) konnte entsprechend der Ergebnisse aus Kapitel 4 geprüft werden, aber nicht in letzter Genauigkeit auf Fehler untersucht werden. Weiterhin konnte die physische Kinematik im gegebenen Zeitrahmen noch nicht in den Gesamtprozess integriert werden. Die Funktionalität einer SPS-gesteuerten Portalkinematik ist aber durch die Systemauslegung entsprechend den verfügbaren Datenblättern hinreichend geprüft. In den nachfolgenden Entwicklungsschritten muss das Gesamtsystem montiert, und der vollständige Programmablauf getestet werden.

Das System ist softwareseitig aktuell vollständig in Python integriert, wodurch es auf jedem System ausführbar ist. Bis auf grafische Nutzerschnittstellen lässt sich die Software im aktuellen Stadium problemlos in einem Docker-Container mit Python 3 Image ausführen. Die Open-Source Machine-Vision-Bibliothek OpenCV liefert die Grundlage für die Bildverarbeitung.

Die softwareseitige Implementierung inklusive der Bildverarbeitungsalgorithmen offenbarten sich als größte Herausforderung der praktischen Implementierung. Hierzu konnte ein Algorithmus entwickelt werden, welcher auf Kamerabildern perspektivische Verzerrungen an Blechteilen der hiesigen Definition entfernen kann. Weiterhin konnten aus einem einzelnen Kamerabild alle notwendigen Lagedaten eines Bauteils extrahiert werden, sodass bauteilbezogene Messpunkte im Voraus bestimmbar sind. Diese Daten werden strukturiert in einer flexiblen Dateistruktur gespeichert.

In der Fassung, wie sie zum Zeitpunkt dieser Arbeit vorliegt wird der Computer über Ethernet mit den Komponenten des Demonstrators verbunden. Die wichtigste Nutzereingabe ist die Auswahl des zu vermessenden Bauteils, dieser Prozess kann bei einer bekannten Abfolge an Messbauteilen allerdings einfach automatisiert werden. Bei Bedarf besteht die Möglichkeit, per Mausklick weitere Messpunkte auszuwählen

Beurteilung der Lösung & Prüfen der Anforderungen

Aus der abschließenden Systemanalyse geht hervor, dass abgesehen von der Integration und Inbetriebnahme des Gesamtsystems hardwareseitig alle funktionalen Systemanforderungen aus 3.3 erfüllt werden. Über die Anforderungen bezüglich der vollständigen Prozessdauer eines Messvorganges können zum aktuellen Zeitpunkt keine quantitativen Aussagen getroffen werden.

Die Funktionalität des Bildbearbeitungsprozesses, sowie der internen Netzwerkkommunikation zwischen den Komponenten konnte für die getesteten Proben gezeigt werden, aber nicht im Allgemeinen sichergestellt werden. Für eine qualifiziertere Aussage über die Fehlerquellen und Bugs gerade in der Software müssten intensivere Tests am Gesamtsystem durchgeführt werden.

In einem praktischen Beispiel konnte die Implementierung des Demonstrator-Systems bis auf das Verfahren der physischen Kinematik demonstriert werden. Die Genauigkeit der berechneten Koordinaten lag bei deutlich erkennbaren perspektivischen Verzerrungen deutlich innerhalb der gesetzten Toleranzen (siehe 4.4).

5.2 Möglichkeiten zur Weiterentwicklung, industrieller Nutzen

Eine Weiterentwicklung des entwickelten Systems kann unter verschiedenen Ansätzen verlaufen. Zunächst wird die Skalierbarkeit des Systems in der aktuellen Form analysiert, wobei auch Spezialisierungsmöglichkeiten und alternative Lösungsansätze betrachtet werden. Bei einer entsprechenden Spezialisierung und Skalierung des Systems ist weiterhin eine industrielle Nutzung einer solchen Technologie denkbar.

Skalierbarkeit

Das Ziel der Entwicklung war die Ermöglichung einer leichten Skalierbarkeit der Technologie, sowohl horizontal als auch vertikal. Horizontale Skalierung beschreibt das Verteilen einer Aufgabe an mehrere Systeme, welche simultan arbeiten („Divide and Conquer“). Vertikale Skalierung beschreibt eine Leistungssteigerung durch Erweiterungen oder Veränderungen eines einzelnen Systems [83]. Möglichkeiten zur einfachen horizontalen Skalierung der Technologie wären unter Anderem:

- Verwendung eines Kamera-Arrays (Mehrere Chips) zur Vergrößerung des Messbereichs oder zur Steigerung der Bildauflösung
- Mehrere Kinematiken mit mehreren Laser-Scannern zur zeitlich parallelisierten Abarbeitung der Messpunkte

Die Sinnhaftigkeit der jeweiligen Skalierungen ist stark situationsabhängig und deren Implementierungen machen eine Anpassung des Systems nötig, welcher bei der einfachen Ausführung aller Subsysteme nicht entsteht. Bei mehreren Kameras müsste das Bild vorher korrekt zusammengefügt werden, und bei mehreren Kinematiken wäre die Bahnplanung deutlich komplexer.

Möglichkeiten zur vertikalen Systemskalierung wären unter anderem Folgende:

- Steigerung der Genauigkeit der einzelnen Systeme (Wiederholgenauigkeit der Kinematik, Auflösung der Kamera) mit Ziel zur Steigerung der Gesamtgenauigkeit.
- Verwendung eines größeren Roboters zum Vermessen größerer Blechbauteile.
- Montage der Kamera an der Kinematik für insgesamt bessere Bild-Auflösung auf Kosten der Prozessgeschwindigkeit wegen aufwändigerem Bildgebungsschritt.
- Steigerung der Prozessgeschwindigkeit (Leistungsstärkere Kinematik, optimierter Datentransfer der Kamera, Optimierung der Programmlaufzeit).

Da das Programm keine asynchronen Anteile besitzt, lässt sich die Gesamtlaufzeit eines Programmdurchlaufes aus den anteiligen Laufzeiten der einzelnen Unterprogramme aufaddieren. Hierdurch hat jede zeitliche Effizienzsteigerung eine Auswirkung auf die Gesamtlaufzeit. Da die Systemungenauigkeiten als unabhängig voneinander angenommen wurden, müssten für eine substanzielle Steigerung der Gesamtgenauigkeiten allerdings alle Subsysteme in einer ähnlichen Größenordnung optimiert werden[32]. Dies könnte in der Umsetzung je nach gewünschtem Grad der Verbesserung ebenfalls Herausforderungen aufwerfen.

Weiterentwicklung für industrielle Anwendungen

Für eine industrielle Verwendung ist eine Inline-Nutzung der Technologie attraktiv, um jedes Bauteil einer Qualitätskontrolle unterziehen zu können. Aufgrund der vielen Gründe für eine robuste und genaue Qualitätssicherung in Blechfertigungs- und Verarbeitungsprozessen kann eine vollautomatische Maschine zur Kantenmessung theoretisch in vielen Anwendungen sinnvoll genutzt werden.

Hierzu sollte eine Möglichkeit gegeben sein, das Messobjekt automatisch in den Messbereich der Maschine zu transportieren. Mit einem Förderband, welches durch den Messbereich verläuft und für die Messung angehalten wird, könnte der Transportprozess mit geringem Steuerungsaufwand vollautomatisiert werden. Die Maschine könnte direkt in eine Fertigungsstraße, beispielsweise hinter

[32] Annahme der Gaußschen Fehlerfortpflanzung aus 3.5.4

einer Entgrat-Maschine integriert werden, und genaue Auskünfte über jedes produzierte Teil wären möglich. Hierzu muss eine hinreichende Prozessgeschwindigkeit gewährleistet sein.

In der aktuellen Form dient die in dieser Arbeit entwickelte Technologie nur zu Demonstrationszwecken oder als alleinstehendes Messgerät. Für eine industrielle Nutzbarkeit müsste das System ausgehend vom Gerüst überarbeitet werden. Auch zur Zertifizierung nach ISO 9001 (siehe 2.4.2) müsste das System nachweislich robust eigene Prozessfehler überprüfen. Eine sicherheitstechnische Absicherung des Systems wäre ebenso vor einer Eingliederung sinnvoll bis vorgeschrieben. Bei einer entsprechenden zeitlichen Optimierung des Messprozesses wäre bei Bedarf eine industrielle Anwendung der Technologie allerdings denkbar, da die generelle Funktionalität der Technologie hinreichend demonstriert werden konnte.

Fazit

Die vorliegende Arbeit hat sich mit der Entwicklung eines Systems zur automatischen Messung der Kantenverrundung von Blechbauteilen beschäftigt. Das Ziel war die Steigerung von Präzision und Prozessgeschwindigkeit bei Qualitätssicherungsprozessen in der blechverarbeitenden Industrie. Hierzu wurde ein Demonstrator konstruiert, wobei ein präziser Laser-Scanner sowie eine Vierachs-Kinematik mit SPS als Messwerkzeug verwendet wird. Für die Objekterkennung und Bildgebung wurden zur Steigerung der Flexibilität Machine-Vision-Anwendungen mit Kamera genutzt.

Grundlage für die Entwicklung war die ausführliche Analyse der Prozessanforderungen sowie des Standes der Technik. Auf Grundlage dieser Erkenntnisse werden die Subsysteme des Demonstrators entwickelt, sowie passende Komponenten ausgewählt. Zum Betrieb des Systems wird weiterhin eine Software entwickelt.

Schwerpunkt der Entwicklung ist die Implementierung der Software. Insbesondere die nötigen Korrekturschritte bei der Bildverarbeitung, sowie die Implementierung einer Kommunikationsstruktur zwischen den einzelnen Komponenten stellten sich als wesentliche Herausforderungen heraus. Anhand eingängiger Tests konnte an einigen Beispielen das Einhalten der gesetzten Anforderungen seitens der Software und Bildgebung bereits erfolgreich gezeigt werden.

Abschließend kann festgestellt werden, dass die im Rahmen dieser Arbeit bearbeiteten Entwicklungsschritte erfolgreich und im Sinne der Anforderungen abgelaufen sind. Die physische Konstruktion des Demonstrators findet im Anschluss an diese Arbeit statt. Die Ergebnisse der Entwicklung zeigen, dass eine präzise, vollautomatische Messung der Kantenverrundung von freiliegenden Blechbauteilen mit dem hier entwickelten System möglich ist.

A Inhaltsverzeichnis des digitalen Anhangs

- 01_Python_env[33]
  - EXPORT
    - Export-CSV-Beispieldatei
  - IMAGES
    - Beispielbilder (künstlich konstruiert)
  - LOGGING
    - Beispiel Log-Datei
  - TMP
    - Temporäre Programm-Dateien (leer)
  - Python-Code
  - JSON-Dateien
- 02_cad_data
  - STEP-Dateien der Demonstrator-Modelle / Fertigungsteile
  - Technische Zeichnungen

[33] Der Anhang dieser Arbeit setzt sich aus dem Programmcode zusammen, welcher im Rahmen dieser Arbeit entstanden ist. Die Software kann unter Voraussetzung der Entwicklungs-Umgebung nach 4.1 getestet werden. Zum Testen des Programmes muss das Programm aus dem Ordner Python_env ausgeführt werden.

Abbildungsverzeichnis

Abbildung 1.1: Ausgeprägter Grat bei Löchern in einem Dünnblech [2] 1

Abbildung 2.1: Schematische Bestimmung des Kantenradius aus Linien-Kantenprofil [8] (Grafik angepasst) 4

Abbildung 2.2: Roboter-Kinematiken mit: (v. o. n. u) DOF, Arbeitsraum, Koollisionsraum [16] 6

Abbildung 2.3: Aufbau eines Kinematik-Steuersystems mit SPS 8

Abbildung 2.4: Lochkamera-Modell mit beschreibenden Parametern (engl.) [38] 9

Abbildung 2.5: Verzeichnungstypen bei Linsenobjektiven (v.l.n.r. ohne, Kissen- und Tonnen-Verzeichnung) [40] 10

Abbildung 2.6: Projektionsfehler bei einem quaderförmigen Objekt (blau) im Lochkamera-Modell 10

Abbildung 3.1: V-Modell als generische Entwicklungsmethodik 16

Abbildung 3.2: Vertiefende Methodik der Entwicklung 17

Abbildung 3.3: Schematischer PAP des Gesamtsystems auf Basis der Anforderungen 20

Abbildung 3.4: Scanbereich Gocator 2510 [58] 22

Abbildung 3.5: Scanbereich, Detail [58] 22

Abbildung 3.6: Anstellung Laser-Scanner an der Bauteilkante mit Sichtbereich (gelb) 22

Abbildung 3.7: Mögliche Kinematik-Konzepte 4-6 DOF 23

Abbildung 3.8: Ausrichtung Laser-Scanner mit Drehachse, sowie dessen vereinfachte Darstellung 25

Abbildung 3.9: Beispielhafte Verläufe der Motordynamik (v.l.n.r. Geschwindigkeit, Beschleunigung, Winkel) 26

Abbildung 3.10: Aufbau des Kamerasystems mit Blickfeld (rot) u. Kinematik 29

Abbildung 3.11: Gesamtansicht CAD-Prototyp mit ausgewiesenen Komponenten 34

Abbildung 3.12: 3D-Render CAD-Prototyp mit Verkleidung (Tür geschlossen) 35

Abbildung 4.1: Blockschaltbild des Gesamtsystems 36

Abbildung 4.2: Implementierung der Netzwerkkommunikation (Übersicht) 37

Abbildung 4.3: Klassendiagramm der Softwareumgebung 38

Abbildung 4.4: Prüfstand zum Test des Kamerasystems und der anschließenden Koordinatenberechnung 43

Abbildung 4.5: Deutlich erkennbare Tonnen-Verzeichnung am unbearbeiteten Kamerabild (Kontrast erhöht) 44

Abbildung 4.6: Entzerrtes Kamerabild. Sichtbare Einschnürung an den Seitenmitten durch die Transformation 45

Abbildung 4.7: Als PNG-Grafik angezeigtes DXF-Objekt, in Autodesk AutoCAD erstellt, frei positioniert 48

Abbildung 4.8: Positionierung der Referenzdaten nach Konvention (schwarze Umrandung zur Darstellung) 48

Abbildung 4.9: Veranschaulichung der Voraussetzung für die Sichtbarkeit der Seitenflächen mit optischer Achse (rot) und Blechbauteil (grauer Verlauf) 49

Abbildung 4.10: Bildausschnitt des unbearbeiteten, „simulierten“ Kamerabildes 51

Abbildung 4.11: Entzerrungsalgorithmus Schritt 1 52
Abbildung 4.12: Entzerrungs-Algorithmus Schritt 2 52
Abbildung 4.13: Entzerrungs-Algorithmus Schritt 3 52
Abbildung 4.14: Entzerrungs-Algorithmus Schritt 4 52
Abbildung 4.15: Entzerrungs-Algorithmus Vergleich 52
Abbildung 4.16: Entzerrungs-Algorithmus Endergebnis 52
Abbildung 4.17: Scheinbar größere Exzentrizität und Größe von dreidimensionalen Objekten (blau) in der Bildebene 54
Abbildung 4.18: Gradienten-Feld (rot) als Kanten-Normalenvektoren des Objektes im Binärbild ..59
Abbildung 4.19: CSV-Exportdatei Beispiel mit zwei Messpunkten 61
Abbildung 4.20: PAP Messprozess SPS/ Laser-Scanner seitens des Computers 65
Abbildung 4.21: PAP seitens der SPS 66
Abbildung 4.22: Ansicht der Simulation im Siemens TIA-Portal 66
Abbildung 4.23: Prüfstand zum isolierten Testen des Laser-Scanners 68
Abbildung 4.24: Laser-Scanner-Messung (rot) mit visualisierten Auswertungsalgorithmen 69
Abbildung 4.25: Detail-Ansicht Laser-Scanner Montage an der Kinematik im CAD-Modell 71
Abbildung 4.26: Montagebleche zur Montage des rotierbaren Laser-Scanners 72
Abbildung 4.27: Finaler Stand der praktischen Ausarbeitung (3D-Render) 73
Abbildung 4.28: Kamerabilder roh (links) und vollständig bearbeitet (rechts) 75
Abbildung 4.29: Messpunkte, relativ zu Referenzkoordinaten 76
Abbildung 4.30: Tatsächliche Lage der ausgewählten Messpunkte (händische Messung) 77

Tabellenverzeichnis

Tabelle 3-1: Spezifikation Q-Fin F200 [56] 18
Tabelle 3-2: Auswahlkriterien & Bewertung der kompatiblen Kinematik-Konzepte 24
Tabelle 3-3: Auszug aus Datenblatt IGUS DLE-RG-0001 [60] 24
Tabelle 3-4: Auszug aus Datenblatt 17HS15-1684S-PG5 Motor inkl. EG17-G5 Getriebe [61–63] 27
Tabelle 3-5: Liste der Komponenten des SPS-Systems 28
Tabelle 3-6: Auszug aus Datenblatt Raspberry Pi HQ Camera, inkl. Ergänzung [67] 30
Tabelle 3-7: Auszug aus Technischen Daten Raspberry Pi 8mm Objektiv M12-Mount [70] 32
Tabelle 3-8: Stückliste der vorausgelegten Komponenten 33
Tabelle 4-1: Modulliste für die Software-Implementierung in Python 3.10 mit Version 37
Tabelle 4-2: Kurzbeschreibung der Klassen 39
Tabelle 4-3: Im Programm integrierte Filter (Faltungs-Matrizen) 46
Tabelle 4-4: Referenzbild aus DXF-Datei nach Konvention (oben), simulierte Kamerabilder mit perspektivischer Verzerrung (mitte), Entzerrte Bilder, an Rahmen angepasst (unten) 62
Tabelle 4-5: Tatsächliche Koordinaten der Messpunkte (manuelle Messung) 77

Abkürzungsverzeichnis

API	Application Programming Interface
COBOT	Collaborative Robot
CPU	Central Processing Unit
DIN	Deutsches Institut für Normung
DOF	Degrees Of Freedom
DXF	Drawing Exchange Format
EN	Europäische Norm
FOV	Field Of View
GUI	Graphical User Interface
ISO	International Organisation for Standardisation
MIPI CSI-2	Mobile Industry Processor Interface - Camera Serial Interface 2
PAP	Programmablaufplan
PIP	Python Package Installer
rpm	rotations per minute
S/FTP	Secure / File Transfer Protocol
SCARA	Selective Compliance Assembly Robot Arm (dt. Gelenkarm-Roboter)
SDK	Software Development Kit
SPS	Speicherprogrammierbare Steuerung
SSH	Secure-Shell
TCP/IP	Transmission Control Protocol / Internet Protocol
VDI	Verein Deutscher Ingenieure
VNC	Virtual Network Computing

Literaturverzeichnis

[1] **Manfred Schlatter (Hrsg.) (1986).** IPA-IAO Forschung und Praxis. Entgraten durch Hochdruckwasserstrahlen. Springer.

[2] **blech-entgratung.de.** Wissen rund um die Blechentgratung. https://www.blech-entgratung.de (letzter Zugriff am 9.2.23).

[3] **DIN (2007). DIN EN 10079.** Begriffsbestimmungen für Stahlerzeugnisse; Deutsche Fassung EN 10079:2007. Beuth.

[4] **Mareile Kriwall (Ohne Datum).** Entgraten: Maßnahmen und Methoden. https://www.iph-hannover.de/de/information/umformtechnik/entgraten/.

[5] **Abele, E. (2019).** Technologie der Fertigungsverfahren II. Vorlesungsskript.

[6] **C.-F. Wyen, K. Wegener (2010).** Influence of cutting edge radius on cutting forces in machining titanium. CIRP Annals **59**/1, 93–96.

[7] **Frank, P., Otto, A. (2019).** Flakkotieren von Fräsern zum Mikro-Hartfräsen. wt (Werkstattstechnik online) **109**/11-12, 833–839.

[8] **Romano GmbH (Ohne Datum).** K-Faktor. http://romano-gmbh.de/?page_id=236 (letzter Zugriff am 19.3.2023).

[9] **ISO (2017). ISO 13715:2017.** Technical product documentation - Edges of undefined shape - Indicating and dimensioning, 3. Aufl.

[10] **DIN (2019). DIN EN 1090-3.** Ausführung von Stahltragwerken und Aluminiumtragwerken - Teil 3. Beuth.

[11] **DIN (2018). DIN EN 1090-2.** Ausführung von Stahltragwerken und Aluminiumtragwerken - Teil 2. Beuth.

[12] **ARKU Maschinenbau GmbH (2022).** Runde Sache am Blech: Fünf Gründe zum Kantenverrunden.

[13] **DIN (2007). DIN EN ISO 8501-3.** Vorbereitung von Stahloberflächen vor dem Auftragen von Beschichtungsstoffen. Beuth.

[14] **Lindörfer, M., Finus, F. (2018).** Warum ist entgraten wichtig? Und was Sie dabei beachten sollten. https://www.blechnet.com/warum-ist-entgraten-wichtig-und-was-sie-dabei-beachten-sollten-a-673125/.

[15] **DIN (2011). DIN 6930-1.** Stanzteile aus Stahl - Teil 1: Technische Lieferbedingungen. Beuth.

[16] **Hubertus Nitschke (2002).** Zur Bestimmung geometrischer Parameter von Industrierobotern. Dissertation. Bayerische Akademie der Wissenschaften, München.

[17] **Siciliano, B., Khatib, O. (2016).** Springer Handbook of Robotics. Springer.

[18] **Saheb, S. H., Babu, G., Raju, N. (2018).** Relative Kinematic Analysis of Serial and Parallel Manipulators. IOP Conf. Ser.: Mater. Sci. Eng. **455**, 12040.

[19] **Klaus Wüst (2018).** Grundlagen der Robotik. Vorlesungsskript. Technische Hochschule Mittelhessen.

[20] **Prof. Dr. Mark Ross (2018).** Robotik. Vorlesungsfolien. Hochschule Koblenz.

[21] **Krishnan, R. (2001).** Switched reluctance motor drives: modeling, simulation, analysis, design, and applications. CRC press.

[22] **Ioannides, M. G. (2004).** Design and implementation of PLC-based monitoring control system for induction motor. IEEE Transactions on Energy Conversion **19**/3, 469–476.

[23] **Dai, W., Vyatkin, V. (2012).** Redesign Distributed PLC Control Systems Using IEC 61499 Function Blocks. IEEE Transactions on Automation Science and Engineering **9**/2, 390–401.

[24] **DIN (2004). DIN EN 61131-1.** Speicherprogrammierbare Steuerungen - Teil 1: Allgemeine Informationen. Beuth.

[25] **DIN (2014). DIN EN 61131-3.** Speicherprogrammierbare Steuerungen- Teil 3: Programmiersprachen. Beuth.

[26] **DIN (2020). DIN EN IEC 61131-10.** Speicherprogrammierbare Steuerungen - XML-basiertes Austauschformat. Beuth.

[27] **Siemens AG (2019).** TIA Portal iPDF german. https://assets.new.siemens.com/siemens/assets/api/uuid:9bb07edc-ea5a-4128-99ff-7f9d4ae623db/7801-09-tia-p-ipdf-de-181029-1.pdf (letzter Zugriff am 25.5.2023).

[28] **Jain, R., Kasturi, R., Schunck, B. G. (1995).** Machine vision. McGraw-hill New York.

[29] **Hafiz, A. M., Parah, S. A., Bhat, R. A. (2021).** Reinforcement learning applied to machine vision: state of the art. Int J Multimed Info Retr **10**/2, 71–82.

[30] **Golnabi, H., Asadpour, A. (2007).** Design and application of industrial machine vision systems. Robotics and Computer-Integrated Manufacturing **23**/6, 630–637.

[31] **Pereyra, V., Scherer, G. (1973).** Efficient Computer Manipulation of Tensor Products with Applications to Multidimensional Approximation. Mathematics of Computation **27**/123, 595–605.

[32] **OpenCV.** OpenCV Website. https://opencv.org (letzter Zugriff am 1.4.2023).

[33] **Dlamini, S., Kao, C.-Y., Su, S.-L., Jeffrey Kuo, C.-F. (2022).** Development of a real-time machine vision system for functional textile fabric defect detection using a deep YOLOv4 model. Textile Research Journal **92**/5-6, 675–690.

[34] **A. R. Rababaah, Y. Demi-Ejegi (2012).** Automatic visual inspection system for stamped sheet metals (AVIS3M). In: 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), 661–665.

[35] **Leemans, V., Magein, H., Destain, M.-F. (1998).** Defects segmentation on 'Golden Delicious' apples by using colour machine vision. Computers and Electronics in Agriculture **20**/2, 117–130.

[36] **McCurrach, B. (2022).** A Quick Overview Of Industrial Camera Interfaces. Quality **61**/9, 31.

[37] **Hundelshausen (Ohne Datum).** Kamerakalibrierung nach Tsai. Verlesungsfolien.

[38] **Heikkila, J. (2000).** Geometric camera calibration using circular control points. IEEE Trans. Pattern Anal. Machine Intell. **22**/10, 1066–1077.

[39] **Dawson-Howe, K. M., Vernon, D. (1994).** Simple pinhole camera calibration. Int. J. Imaging Syst. Technol. **5**/1, 1–6.

[40] **Swarowski Optik (Ohne Datum).** Verzeichnung. https://myservice.swarovskioptik.com/s/article/Was-ist-die-Verzeichnung?language=de.

[41] **Wolfgang Osten, Norbert Kerwien (2005).** Optische Messtechnik an den Grenzen zwischen Makro und Nano. Wechselwirkungen, Jahrbuch, 51–67.

[42] **Daniel Wilhelm (2022).** Optische Messverfahren. https://3dimetik.de/glossar/optische-messverfahren/ (letzter Zugriff am 11.4.2023).

[43] **Hering, E., Martin, R. (2017).** Optische Sensoren und Messtechnik. In: Optik für Ingenieure und Naturwissenschaftler. Ekbert Hering, Rolf Martin (Hrsg.). Carl Hanser Verlag GmbH & Co. KG, 494–617.

[44] **Shao, W., Guo, J., Zhou, A. (2009).** A Framework for Measurement and Analysis of Sheet Metal Parts. In: 2009 Second International Conference on Intelligent Computation Technology and Automation. IEEE, 363–366.

[45] **Lin, X., Wang, J., Zhou, Y., Lin, C. (2018).** A Rapid 3D Vision Inspection System for Sheet Metal Parts Based on Feature Extraction and Partial Point Clouds. IOP Conf. Ser.: Mater. Sci. Eng. **452**.

[46] **Silicann Systems GmbH (2021).** Inline, Online, oder Atline - Messansätze in der Prozessanalytik. https://www.silicann.com/blog/beitrag/inline-online-atline-prozessanalytik/ (letzter Zugriff am 14.4.2023).

[47] **Huang, C. K., Wang, L. G., Tang, H. C., Tarng, Y. S. (2006).** Automatic laser inspection of outer diameter, run-out and taper of micro-drills. Journal of Materials Processing Technology **171**/2, 306–313.

[48] **DIN (2015). DIN EN ISO 9001.** Qualitätsmanagementsysteme Anforderungen (ISO 9001:2015). Beuth.

[49] **TÜV Süd (Ohne Datum).** ISO 9001 Qualitätsmanagementsystem. https://www.tuvsud.com/de-de/dienstleistungen/auditierung-und-zertifizierung/iso-9001.

[50] **DIN (2004). DIN EN ISO 10012.** Messmanagementsysteme - Anforderungen an Messpro ^ zesse und Messmittel. Beuth.

[51] **André Küster Simic, Malte Knigge, Janek Schönfeldt (2020).** Struktur, Entwicklung und Zukunft der deutschen Stahlindustrie. Eine Branchenanalyse. Nr. 187.

[52] **DIN (2011). DIN EN 1090-1:2010.** Ausführung von Stahltragwerken und Aluminiumtragwerken - Teil 1. Beuth.

[53] **DIN (2020). DIN EN 1090-4:2018.** Ausführung von Stahltragwerken und Aluminiumtragwerken - Teil 4. Beuth.

[54] **DIN (2020). DIN EN 1090-5:2017.** Ausführung von Stahltragwerken und Aluminiumtragwerken - Teil 5. Beuth.

[55] **Kaufmann, T. (2021).** Strategiewerkzeuge aus der Praxis. Analyse und Beurteilung der strategischen Ausgangslage. Springer Gabler, Berlin, Heidelberg.

[56] **Q-Fin (Ohne Datum).** Q-Fin_Productflyer_F200_DE. https://qfin-finishing.de/wp-content/uploads/2023/01/Q-Fin_Productflyer_F200_DE.pdf

[57] **Autodesk (2011).** DXF Reference. https://images.autodesk.com/adsk/files/autocad_2012_pdf_dxf-reference_enu.pdf.

[58] **LMI Technologies (2018).** Datasheet Gocator 2510/ 2520.

[59] **Dipl.-Ing. Tobias Johannes Kraus (2011).** Eine Delta-Kinematik für den nichttaktilen Einsatz in der Chirurgie. Technische Universität München.

[60] **IGUS (2022).** Technische Dokumentation Linear Robot. Raumportal. https://www.igus.de/product/20423?artNr=DLE-RG-0001 (letzter Zugriff am 14.2.23).

[61] **stepperonline (2022).** Planetary Gearbox EG17-G5. https://www.omc-stepperonline.com/de/eg-serie-planetengetriebe-uebersetzungsverhaeltnis-5-1-verdrehspiel-15-arcmin-fuer-nema-17-schrittmotor-eg17-g5 (letzter Zugriff am 20.3.2023).

[62] **stepperonline (2021).** Stepper Motor 17HS15-1504-ME1K. https://www.omc-stepperonline.com/de/nema-17-schrittmotor-mit-geschlossener-regelkreis-45ncm-64oz-in-mit-magnetischem-geber-1000ppr-4000cpr-17hs15-1504-me1k

[63] **stepperonline (Ohne Datum).** 17HS15-1504-ME1K Torque Curve.

[64] **DIN (1996).** DIN 1319-3. Grundlagen der Meßtechnik - Teil 3: Auswertung von Messungen einer einzelnen Messgröße, Messunsicherheit. Beuth.

[65] **Keyence (Ohne Datum).** Vision Basics. Dimensionsmessung. https://www.keyence.de/ss/products/vision/visionbasics/use/inspection03/ (letzter Zugriff am 19.02.23).

[66] **A3 Vision & Imaging (Ohne Datum).** GigE Vision Standard. https://www.automate.org/a3-content/vision-standards-gige-vision.

[67] **Raspberry Pi Ltd. (2023).** hq camera product brief. https://www.berrybase.de/raspberry-pi-high-quality-kamera.

[68] **Schuhmann, R., Thöniß, T. (1998).** Telezentrische Systeme für die optische Meß- und Prüftechnik. Technisches Messen **65**/4, 131–136.

[69] **Reider, G. A. (2022).** Photonik. Eine Einführung in die Grundlagen, 4. Aufl. Springer Vieweg, Berlin.

[70] **Raspberry Pi Foundation (2023).** GJ-M12-8IR_SC0949. https://thepihut.com/products/portrait-m12-lens-12mp-8mm-1-1-7 (letzter Zugriff am 8.4.2023).

[71] **Jeff Forcier (Ohne Datum).** Paramiko Documentation. https://docs.paramiko.org/en/stable/ (letzter Zugriff am 24.5.2023).

[72] **Raspberry Pi Trading Ltd. (2021).** Raspberry Pi PoE+ HAT Product Brief. https://www.berrybase.de/media/pdf/83/f1/b0/produkt_downloads-210520-Product-Brief-RPi-PoE-HAT.pdf (letzter Zugriff am 24.5.2023).

[73] **calib.io (2023).** Pattern Generator. https://calib.io/pages/camera-calibration-pattern-generator (letzter Zugriff am 24.5.2023).

[74] **OpenCV (Ohne Datum).** Camera Calibration. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (letzter Zugriff am 24.5.2023).

[75] **M. H. Chowdhury, W. D. Little (1995).** Image thresholding techniques. In: IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing. Proceedings, 585–589.

[76] **Abraham, S (Ohne Datum).** Image Manipulation. Filters and Convolutions. Vorlesungsfolien.

[77] **Bennsteiner, A. (2016).** Geometrische Roboterkalibrierung mit optischen Methoden. Masterthesis. Johannes Kepler Universität Linz, Linz.

[78] **Huttenlocher, D. (Ohne Datum).** Hausdorff-Based Image Comparison. https://www.cs.cornell.edu/~dph/hausdorff/hausdorff1.html.

[79] **Huttenlocher, D., Klanderman, G., Rucklidge, W. (1993).** Comparing images using the Hausdorff distance. Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, no. 9, 850–863.

[80] **Gavish, B., Graves, S. C. (1978).** The travelling salesman problem and related problems.

[81] **Gijs Molenaar, Stephan Preeker (2013).** python-snap7 Documentation. https://python-snap7.readthedocs.io/en/latest/ (letzter Zugriff am 25.5.2023).

[82] **LMI Technologies Inc. (Ohne Datum).** Gocator Line Profile Sensor User Manual.

[83] **Hidders, J. (2012).** Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies. ACM, New York, NY.