Freie Universität Berlin

Fraunhofer IPK
INSTITUT PRODUKTIONSANLAGEN UND KONSTRUKTIONSTECHNIK

Eine Werkzeugkette für das Reverse Engineering von Konstruktionsteilen aus 3D-Scans

Abschlussarbeit
zur Erlangung des akademischen Grades
Master of Science
im Studiengang Informatik

Vorgelegt am 9. März 2016 von

Robert Schlenz
Matrikelnummer: 4370001

Erstgutachter:	Prof. Dr. Ina Schieferdecker
Zweitgutachter:	Prof. Dr.-Ing. Rainer Stark
Betreuer:	Dipl.-Ing. Hendrik Grosser

Institut für Informatik, Freie Universitüat Berlin, Deutschland

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, sind als solche gekennzeichnet. Die Zeichnungen oder Abbildungen sind von mir selbst erstellt worden oder mit entsprechenden Quellennachweisen versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner Prüfungsbehörde eingereicht worden.

Berlin, den 9. März 2016

(Robert Schlenz)

Zusammenfassung

Trotz der zunehmenden Digitalisierung in vielen Bereichen der Industrie ist die Instandhaltung langlebiger Produktionsgüter teilweise noch stark von manuellen Abläufen geprägt. Dazu gehören beispielsweise die Demontage und die Bestimmung der korrekten Bauteile. Der Einsatz der fortschreitenden Technologien der 3D-Scanner und der Computer Vision bietet jedoch neue Möglichkeiten.

Diese Arbeit stellt eine Werkzeugkette vor, die es ermöglicht, einen Scan einer Baugruppe zu segmentieren, die einzelnen Bauteile zu identifizieren und die Referenzen anhand der Segmente auszurichten. Dafür werden verschiedene Verfahren für die einzelnen Teilprozesse bewertet und innerhalb einer Anwendung samt graphischer Benutzeroberfläche unter Verwendung einer freien Bibliothek zur Verarbeitung dreidimensionaler Daten implementiert.

Es wird gezeigt, dass die Rekonstruktion zweier verschiedener Baugruppen basierend auf einen dreidimensionalen Scan und der umgesetzten Werkzeugkette möglich ist.

Abstract

Despite of growing digitalisation in many fields of industry the maintenance of durable investment goods is heavily shaped by manual processes. For example, this includes disassembly and determination the correct components . The usage of advancing technologies of 3D scanners and computer vision offers new possibilities.

This thesis introduced a toolchain which enables to segment scans of an assembly, to identify separate components and to align references to these separate components. Therefor different approaches are validated and an application including graphical user interface for processing of three-dimensional data is implemented with the use of a free library.

It is shown that the reconstruction of two distinct assemblies is possible by using their scans and the implemented toolchain.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Ausgangssituation . . . 1
1.2 Vorarbeiten am Fraunhofer IPK . . . 2
1.3 Problemstellung . . . 4
1.4 Lösungsansatz . . . 4
1.5 Übersicht . . . 6

2 Stand der Technik **7**
2.1 Erzeugung von 3D-Scans . . . 7
2.1.1 Optische Digitalisierungsverfahren . . . 8
2.1.2 Computertomographie . . . 8
2.2 Nachbearbeitung . . . 9
2.3 Datenformate . . . 9
2.4 Verwandte Arbeiten . . . 12

3 Entwurf **17**
3.1 Analyse der bestehenden Identifikation . . . 17
3.1.1 Konzept . . . 17
3.1.2 Bestimmung der *Keypoints* . . . 18
3.1.3 Bestimmung der *Features* . . . 19
3.1.4 Bestimmung der Korrespondenzen . . . 22
3.2 Analyse der bestehenden Registrierung . . . 24
3.2.1 Konzept . . . 24
3.2.2 Bestimmung der Korrespondenzen . . . 24
3.2.3 Bestimmung der Ausrichtung . . . 25
3.3 Zusammenfassung der Teilprozesse . . . 26
3.4 Segmentierungsalgorithmen . . . 27
3.5 Entwurf der Werkzeugkette . . . 31

4 Implementierung **37**
4.1 Entwicklungsumgebung . . . 37
4.2 Strukturmuster . . . 38
4.3 Implementierung der Werkzeugkette . . . 39
4.3.1 Segmentierung . . . 39
4.3.2 Identifikation . . . 41
4.3.3 Registrierung . . . 43
4.4 Implementierung der Anwendung . . . 48

4.4.1 Umsetzung des *MVVM*-Musters 48
4.4.2 Auslagerung von Prozessen 50

5 Durchführung und Auswertung der Fallbeispiele 52
5.1 Aufbau des Testdatensatzes 52
5.2 Fallbeispiel 1: PKW-Lichtmaschine 52
5.3 Fallbeispiel 2: Turbinenstufe 63

6 Zusammenfassung und Ausblick 72

7 Anhang I
7.1 Abbildungen der Referenzteile I
7.2 Versuchsreihen zur Segmentierung der PKW-Lichtmaschine III
7.3 Versuchsreihen zur Segmentierung der Turbinenstufe IX
7.4 Versuchsreihen zur Identifizierung der Segmente der PKW-Lichtmaschine . . XI
7.5 Versuchsreihen zur Identifizierung der Segmente der Turbinenstufe XVII

Literatur XXII

Codeverzeichnis XXV

Abbildungsverzeichnis XXV

Tabellenverzeichnis XXVI

1 Einleitung

1.1 Ausgangssituation

Der Begriff *Industrie 4.0* ist in der Fachpresse derzeit allgegenwärtig. Er fasst die zunehmende Digitalisierung und Vernetzung für eine flexiblere und bessere Steuerung der Wertschöpfungskette in der Industrie zusammen [1]. Ziel der *Industrie 4.0* ist es, den Lebenszyklus eines Produktes von seiner Konstruktion und Produktion, über dessen Einsatzes, bis zum Recycling desselben durch digitale Informationen und Vernetzung zu optimieren. Bei dem Entwurf neuer Produkte oder der Konstruktion neuer Maschinen herrscht bereits ein sehr hoher Grad an Digitalisierung vor. Durch die virtuelle Produktentstehung und die digitale Vernetzung können die Konstruktion und Produktion viel effizienter miteinander verbunden werden. Dabei entstehen heutzutage während der Konstruktion digitale Modelle oder ganze Baugruppen und die anschließende Produktion wird durch digitale Konstruktionspläne vereinfacht und somit beschleunigt.

Aber nicht in allen Bereichen der Industrie hält die Digitalisierung gleichermaßen Einzug. In der industriellen Instandhaltung und Wartung (Maintenance, Repair and Overhaul, MRO) kann sich die Digitalisierung der Prozesse als sehr schwierig gestalten. Es existieren viele hochkomplexe und langlebige Investitionsgüter wie Dampf- oder Gasturbinen, die eine Laufzeit von mehreren Jahrzehnten besitzen können. Die Instandhaltung solcher Güter ist der Schlüssel zu ihrer Langlebigkeit. Allerdings ist der genaue Produktzustand durch fehlende Daten und Dokumente, dem Wechsel von Instandhaltungsfirmen, unvollständige Dokumentation von vorgenommenen Veränderungen oder durch unbekannte Ersatzteile meist ungenau oder sogar unbekannt, wodurch die Instandhaltung zusätzlich erschwert wird. Auch haben die Hersteller das Interesse, ihr Wissen (Intellectual Property, IP) zu schützen und enthalten daher vorhandene Informationen zu Maschinen oder Produkten, wie beispielsweise Konstruktionspläne, Schaltpläne oder Modelle, vor. Daher ist in der industriellen Instandhaltung stets ein hohes Maß an manueller Arbeit vonnöten, um den störungsfreien Betrieb gewährleisten zu können. Für die Instandhaltung kann zunächst die komplette Demontage für das Identifizieren von Bauteilen und Defekten notwendig sein, um anschließend die defekten Teile auszutauschen oder zu reparieren. Unbekannte Bauteile müssen mittels ihrer geometrischen Eigenschaften oder vorhandenen Seriennummern identifiziert werden. Dafür werden Bauteilkataloge oder zunehmend auch PDM-Systeme (Product Data Management) genutzt. Dennoch ist ein großes Fachwissen zur korrekten Klassifizierung unbekannter Konstruktionsteile erforderlich. Erst nach der Klassifizierung kann der Zustand des Konstruktionsteiles bewertet werden, der für die weitere Planung der Instandhaltung unverzichtbar ist. Der große Anteil manueller Arbeit insbesondere durch Demontage und Bauteilidentifikation führt dazu, dass die MRO-Prozesse sehr zeitaufwendig und damit sehr kostenintensiv sind.

Allerdings bietet die fortschreitende Technik neue Möglichkeiten, um auch hier den Anteil manueller Arbeit zu verringern und einen gewissen Grad an Automatisierung zu erreichen. Aktuelle Forschungen am Fraunhofer Institut für Produktionsanlagen und Konstruktionstechnik (IPK) untersuchen, wie die MRO-Prozesse durch die Nutzung von verschiedenen 3D-Scanverfahren unterstützt werden können. Es wird untersucht, inwieweit einzelne Bauteile extrahiert und identifiziert werden können und welche weiteren Informationen sich aus dem 3D-Scan einer kompletten Baugruppe gewinnen lassen, ohne das eine vorherige Demontage erfolgen muss.

1.2 Vorarbeiten am Fraunhofer IPK

Zu den Forschungsprojekten am Fraunhofer IPK gehörte unter anderem das Projekt „CO2-Strahlreinigung und zustandsbasierte Digitalisierung für MRO-Prozesse“ (COSDIMRO) [2]. Innerhalb des Projektes wurde untersucht, ob mithilfe generierter 3D-Modelle die Produktstruktur des Testobjektes abgeleitet werden kann, um dieses in einem Produktdatenmanagement-System zur Dokumentation abzulegen [3].

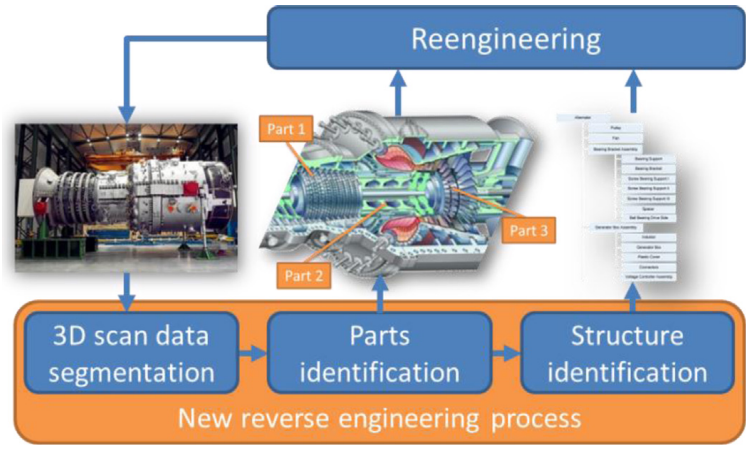


Abbildung 1: Neuer Prozess des Projektes für das Reverse Engineering [3]

Als Teilprozesse wurden dabei die Segmentierung der 3D-Scans, die Identifikation der segmentierten Einzelteile und die darauffolgende Ableitung der Produktstruktur, wie in Abbildung 1 dargestellt, bestimmt. Es zeigte sich, dass 3D-Scans von einzelnen Bauteilen mittels des Abgleichs mit einer kommerziellen Datenbank korrekt identifiziert und zugeordnet werden konnten. Jedoch war die Identifizierung der Einzelteile, die durch Clustering-Algorithmen aus einem kompletten 3D-Scan der Baugruppe extrahiert wurden, nicht erfolgreich. Durch die Extraktion entstanden in den resultierenden Einzelteilen Löcher oder Artefakte, da während der Anwendungen der Clustering-Algorithmen Punkte aus den Modellen

entfernt wurden. Durch diesen Umstand war die anschließende Identifikation sehr fehlerbehaftet. Eine zuverlässige Erkennung konnte nur durch nachfolgende manuelle Aufbereitung der einzelnen Modelle erreicht werden. Für die Ableitung der geometrischen Produktstruktur wurde ein Ansatz entwickelt, der basierend auf der Analyse der vorhandenen Kontaktflächen der CAD-Modelle (Computer-aided Design) einer Baugruppe die Produktstruktur herleitete.

Das Verbundprojekt „Integrierte Prozesskette für die Instandhaltung elektronischer Komponenten“ (INPIKO) [4] beschäftigte sich mit einer ähnlichen Problemstellung. Da auch langlebige Investitionsgüter zunehmend mit elektronischen Komponenten (Printed Circuit Board, PCB) ausgerüstet wurden, müssen defekte PCBs ebenfalls untersucht und repariert werden. Allerdings ist die Reparatur elektronischer Komponenten ohne Schaltungspläne und Stücklisten sehr kompliziert. Auch hier bildeten 3D-Modelle, die mittels 3D-Scanverfahren und Computertomographie erstellt wurden, die Ausgangslage. Ähnlich zu dem COSDIMRO-Projekt wurden die Segmentierung und Identifikation der einzelnen elektronischen Komponenten eines PCBs sowie die Ableitung von Netzplänen untersucht [5]. Ziel war es, die elektronische Analyse durch optische Verfahren für eine stärker automatisierte und verbesserte Rekonstruktion von PCBs zu ergänzen.

Die festgestellte Problematik bei der Identifikation der Einzelteile führte zu einer weiteren Projektarbeit. Dabei wurde untersucht, ob einzelne Objekte innerhalb des Scans einer gesamten Szene zu identifizieren sind [6]. Es wurden verschiedene Verfahren zur Erkennung evaluiert und die Identifizierung basierend auf lokalen Merkmalen innerhalb der geometrischen Struktur als sehr vielversprechend befunden. Dieser Ansatz zeigte sich robuster gegenüber unvollständigen Bauteilen, da mehrere lokale Bereiche miteinander verglichen werden und trotz Unvollständigkeit Übereinstimmungen gefunden werden können.

Das Ziel einer weiteren Projektarbeit war die korrekte Ausrichtung von segmentierten Einzelteilen mit den dazugehörigen Referenzteilen [7]. Digital modellierte Objekte besitzen ein eigenes Koordinatensystem, welches in Abhängigkeit der geometrischen Form des Objektes meist innerhalb des Objektes oder am Objekt selbst liegt. Durch das Scannen erhält das gescannte Objekt ebenfalls ein eigenes Koordinatensystem. Dessen Ursprung liegt meist in der Nähe zum Objekt, ist aber nicht an vorhandenen Achsen oder Flächen ausgerichtet. Es wurde also ein Verfahren entwickelt, dass die Herleitung der Transformation, bestehend aus Translation und Rotation, ermöglicht, so dass die Objekte möglichst kongruent zueinander ausgerichtet werden können. Auch für diese Problemstellung wurden lokale Merkmale innerhalb der Objekte bestimmt, um diese zu vergleichen und um die nötige Transformation zu bestimmen.

1.3 Problemstellung

Die Digitalisierung und Automatisierung der MRO-Prozesse soll durch die Verwendung von 3D-Scanverfahren und digitalen Modellen vorangetrieben werden. Es existieren verschiedene kommerzielle Datenbanken, die den geometrischen Abgleich von Modellen unterstützen und zur Identifizierung genutzt werden können. Allerdings sind dafür vollständige Modelle notwendig, die entweder nur durch vorherige Demontage einer Baugruppe und anschließendem Scannen der einzelnen Bauteile oder durch zeitaufwendige manuelle Nachbearbeitung der segmentierten Teile erstellt werden können. Wenn die verschiedenen Konstruktionsteile identifiziert sind, müssen diese noch ausgerichtet werden, um die Baugruppe korrekt zu beschreiben und um die Produktstruktur der Gruppe in späteren Prozessen herleiten zu können.

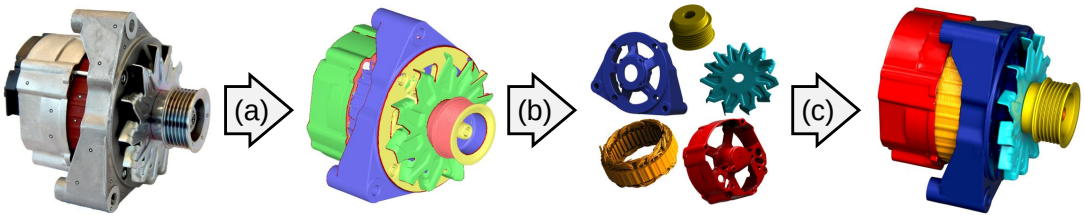


Abbildung 2: Prozesskette des Reverse Engineerings, bestehend aus Segmentierung (a), Identifikation (b) und Registrierung (c)

Der Kern der Problematik liegt darin, dass die vorherigen Projekte stets Teillösungen adressierten, wie beispielsweise die Identifikation [6] oder die Registrierung [7]. Diese Teilprozesse wurden untersucht und optimiert. Eine komplette prototypische Implementierung der gesamten Prozesskette, wie sie in Abbildung 2 dargestellt ist, erfolgte bisher noch nicht. Daher gilt es unter Betrachtung der Vorarbeiten und deren erlangten Erkenntnisse, eine vollständige Werkzeugkette mit einem höchstmöglichen Grad an Automatisierung zu implementieren und zu evaluieren. Darüber hinaus muss für die gesamte Werkzeugkette ein ausgewogenes Maß zwischen Komplexität und Benutzbarkeit hergestellt werden. Einerseits sollen die genutzten Algorithmen und Verfahren möglichst schnell sein, um die aufkommenden Rechenzeiten gering zu halten und so die Benutzbarkeit zu erhöhen. Anderseits bedürfen die verwendeten Verfahren einer gewissen Komplexität, um auch zufriedenstellende Ergebnisse liefern zu können.

1.4 Lösungsansatz

In der Arbeit wird die komplette Prozesskette durchgängig implementiert. Die Werkzeugkette muss Verfahren für das Segmentieren, das Identifizieren und das Ausrichten zur Verfügung

stellen.

1. **Segmentierung**:
   Als Eingabe wird ein 3D-Modell übergeben. Es sollen verschiedene Verfahren angewandt werden können, um die Eingabe in separate Segmente nach Ermessen des Benutzers zu überführen.
2. **Identifikation**:
   Die erstellten Segmente aus dem vorhergehenden Schritt werden zur Identifizierung mit einer vorgegebenen Menge von Referenzobjekten verglichen. Referenzen, bei denen geometrische Gemeinsamkeiten identifiziert werden können, dienen zur Auswahl des korrekten Referenzobjekts.
3. **Registrierung**:
   Alle Paare aus Segmenten und Referenzobjekten müssen zur Rekonstruktion der Baugruppe aneinander ausgerichtet werden. Hierfür muss die nötige Translation und Rotation anhand der korrespondierenden Bereiche bestimmt werden, um das Referenzobjekt hin zur Position des Segments zu transformieren.
4. **Export**:
   Alle gesammelten Informationen, die während der Ausführung der Werkzeugkette anfallen, sollen gebündelt und in Form einer XML-Datei ausgegeben werden. Diese Datei und alle erstellten oder hergeleiteten 3D-Modelle werden abgespeichert und können als Eingabe für zusätzliche Prozesse dienen.

Während der gesamten Werkzeugkette muss die Möglichkeit bestehen, vollzogene Prozesse mit nicht zufriedenstellenden Ergebnissen zu wiederholen oder manuell zu korrigieren.

Für die Umsetzung lässt sich die Arbeit in die folgenden Abschnitte unterteilen und durchführen. Zuerst werden die einzelnen Lösungen der Teilprozesse analysiert und deren wesentliche Bestandteile extrahiert. Ähnliche Zwischenschritte müssen miteinander verglichen werden, um Vor- und Nachteile der verwendeten Verfahren zu bestimmen. Es erfolgt die Kombination der Teillösung für eine möglichst effiziente und komplette Werkzeugkette. Für die Vervollständigung der Prozesskette müssen noch weitere Verfahren zur Verarbeitung von 3D-Scans eingeführt und implementiert werden. Eines der vorhandenen Frameworks wird mit samt seiner graphischen Benutzeroberfläche überarbeitet, um einen Prototypen mit vollständig nutzbarer Werkzeugkette zu erhalten. Abschließend wird diese mit zwei verschiedene Fallbeispiele durchlaufen und die Ergebnisse ausgewertet. Dabei ist vor allem der Zusammenhang von der Genauigkeit des Ergebnisses, dem Grad der Automatisierung und dem manuellen Aufwand von Interesse.

1.5 Übersicht

Der weitere Verlauf der Arbeit ist wie folgt strukturiert: In Kapitel 2 werden zunächst verschiedene Aspekte bezüglich der dreidimensionalen Digitalisierung, wozu konkrete Digitalisierungsverfahren und verbreitete Datenformate gehören, beleuchtet. Es folgt eine Vorstellung verwandter Arbeiten, die sich mit Problematiken einzelner Teilprozesse beschäftigten und dafür verschiedene Lösungsansätze entwickelten. Im darauffolgenden Kapitel 3 wird der Entwurf der Werkzeugkette unter Betrachtung der bestehenden Teillösungen erarbeitet und Kapitel 4 stellt die konkrete Umsetzung der entworfenen Werkzeugkette mit Auszügen aus der Implementierung vor. Dabei wird nicht nur auf die einzelnen Teilprozesse, sondern auch auf wichtige Bestandteile der Anwendung selbst eingegangen. In Kapitel 5 wird die Durchführung der Werkzeugkette für zwei unterschiedliche Fallbeispiele präsentiert. Durch die Analyse der erzielten Ergebnisse folgt eine Bewertung der Werkzeugkette. In Kapitel 6 werden die gewonnenen Erkenntnisse resümiert, Schlussfolgerungen daraus gezogen sowie ein kurzer Ausblick gegeben.

2 Stand der Technik

In dem folgenden Kapitel wird zuerst die Erzeugung von dreidimensionalen Modellen eingeführt. Dafür werden einige Verfahren zur Digitalisierung von Objekten vorgestellt. Anschließend wird die Notwendigkeit der Nachbearbeitung von erstellten dreidimensionalen Scans erläutert. Es werden verschiedene Datenformate für dreidimensionale Modelle präsentiert und die für die Werkzeugkette interessanten Formate näher erläutert. Als Letztes werden interessante Arbeiten vorgestellt, die sich mit ähnlichen Problematiken beschäftigen.

2.1 Erzeugung von 3D-Scans

Die dreidimensionale Erfassung von Objekten wurde in den letzten Jahren stetig vorangetrieben, da diese für eine Vielzahl von Bereichen relevant ist. So ist die Erfassung von Objekten neben der Instandhaltung beispielsweise auch für die Steuerung von Robotern oder fahrerlosen Systemen, wie auch für die Erkennung von Gesichtern oder Gesten wesentlich. Zudem wird die weitere Entwicklung durch höhere Verfügbarkeit von Geräten in verschiedenen Preis- und Qualitätsklassen beschleunigt. Daher existieren viele unterschiedliche Verfahren zur dreidimensionalen Erfassung von Objekten, die sich grundlegend in aktiv und passiv unterteilen lassen [8]. Bei den passiven Verfahren werden lediglich Kameras genutzt und durch Auswertung der entstandenen Bilder dreidimensionale Koordinaten ermittelt. Im Gegensatz zu den passiven Verfahren werden bei den aktiven Verfahren zusätzliche Informationsquellen geschaffen, um die Erfassung des Objektes zu vereinfachen. Optische Verfahren sind nicht taktile Verfahren. Dabei wird das zu erfassende Objekt berührungslos und zerstörungsfrei digitalisiert. Problematisch sind spiegelnde oder durchlässige Flächen, welche die korrekte Erkennung verfälschen können. Darüber hinaus können Erschütterungen ebenfalls zu falschen Messergebnissen führen. In der Industrie existieren aber auch taktile Verfahren, bei denen Objekte durch Abtasten ihrer Oberfläche dreidimensional erfasst werden. Durch die Verwendung einer automatisierten Anlage zum Abtasten kann eine hohe Präzision erreicht werden. Taktiles Erfassen ist allerdings nicht bei leicht biegsamen Objekten möglich. Da optische und taktile Verfahren aber lediglich eine Erfassung der äußeren Oberfläche zulassen, wird auch die Computertomographie zur dreidimensionalen Erfassung eingesetzt. Diese ermöglicht den Einblick in das Innere eines Objektes, ohne dieses zerlegen oder zerstören zu müssen.

Für die Instandhaltung werden sowohl taktile und optische Verfahren als auch Computertomographie genutzt, da alle Digitalisierungsverfahren ihre Vor- und Nachteile haben. Für die vorgestellten Projekte wurden vor allem die optischen Messverfahren und die Computertomographie verwendet [2,4]. Erstere können ein Objekt schnell und ohne Einwirkungen digitalisieren und letzteres ermöglicht die durchgängige Erfassung des Objektes.

2.1.1 Optische Digitalisierungsverfahren

Die optischen Digitalisierungsverfahren nutzen Licht, um Muster auf dem Objekt abzubilden und zu identifizieren. Anhand der bekannten geometrischen Struktur des abgebildeten Musters und der erfassten Form bzw. Position auf dem Objekt lassen sich dreidimensionale Koordinaten ableiten [8].

- **Laserscanning:**
  Beim Laserscanning wird mithilfe eines Lasers eine Linie auf die Oberfläche des Objektes projiziert, welche aus dem Blickwinkel des Lasers eine Gerade bildet. Aus einem anderen Winkel lässt sich die Struktur der Oberfläche anhand der Deformation der Linie erkennen. Der hohe Kontrast zwischen dem Licht des Lasers und der Objektoberfläche ermöglicht eine schnelle und genaue Identifizierung der entsprechenden Pixel im aufgenommenen Bild und deren dreidimensionale Koordinaten können bestimmt werden.
- **Streifenlicht-Verfahren:**
  Bei dem Streifenlicht-Verfahren werden gleichzeitig mehrere Streifen auf das Objekt projiziert. Dadurch entsteht ein Muster aus abwechselnden hellen und dunklen Streifen auf der Oberfläche. Die Grenze zwischen einem hellen und einem dunklen Streifen bildet eine Linie, welche sich wie beim Laserscanning aufgrund des starken Kontrastes schnell erfassen lässt. Der Vorteil liegt darin, dass das Muster mehrere dieser Linien zur selben Zeit auf dem Objekt erzeugt und somit die Erfassung beschleunigt. Neben dem Streifenlicht können auch Punkte auf das Objekt projiziert und erfasst werden.

Als Basis für die Bestimmung der Koordinaten kann die Triangulation dienen. Da der Aufbau der Bestandsteile des Scansystems durch dessen Kalibrierung exakt bekannt ist, kann anhand der Winkel und Abstände die Tiefe der Messpunkte mittels trigonometrischer Funktionen errechnet werden.

2.1.2 Computertomographie

Der Computertomographie (CT) hat ihren Ursprung im medizischen Bereich mit der Entdeckung der Röntgenstrahlung und der Entwicklung von Röntgengeräten. Die heutigen Einsatzgebiete von CT-Scans erstrecken sich weit über den medizinischen Bereich hinaus. Der Vorteil des Erfassungsverfahrens liegt in der durchgängigen und zerstörungsfreien Erfassung von Objekten. Somit wird die Computertomograpie auch als „allgemein-diagnostisches Werkzeug zur zerstörungsfreien Materialprüfung und dreidimensionalen Visualisierung“ [9, S.2] genutzt. Bei einem CT-Scan enstehen axial angeordnete, überlagerungsfreie Schnittbilder, welche durch die Positionierung des Objektes, der Schichtdicke, der Strahlungsenergie und -intensität bestimmt sind [9]. Dabei werden für den industriellen Einsatz vor allem

Fächerstahl- und Kegelstrahl-CT verwendet [10]. Während für die dreidimensionale Rekonstruktion eines Objektes bei der Fächerstrahl-CT eine Bewegung des Objektes oder des Gerätes nötig ist, kann bei der Kegelstrahl-CT das Objekt in Abhängigkeit des Winkels vom Kegel sofort dreidimensional erfasst werden. Auch bei diesem Verfahren können Fehler auftreten. Es kann zu Strahlaufhärtungs- und Objektstreustrahlungsartefakten kommen. Strahlaufhärtungsartefakte entstehen durch die verwendete Strahlung selbst und bei Objektstreustrahlungsartefakten handelt es sich um am Objekt gestreute Photonen, die auf dem Detektor an verschiedenen Positionen auftreffen.

2.2 Nachbearbeitung

Zur Erfassung eines kompletten Objektes und zur Generierung möglichst fehlerfreier Daten ist einige Nachbearbeitung nötig [6]. Hierfür existieren kommerzielle sowie frei zugängliche Werkzeuge, die diese Bearbeitung halb- oder vollautomatisch ermöglichen.

- **Vereinigung:**
  Es können mehrere Messungen zur kompletten Erfassung notwendig sein. Die resultierenden Teilscans müssen für die Rekonstruktion des Objektes aneinander ausgerichtet werden. Zudem muss die Überlappung so nachbearbeitet werden, dass die gleichmäßige Auflösung des Scans gewahrt bleibt.
- **Entfernung überschüssiger Flächen:**
  Während der Messung können Flächen der Umgebung oder hinter transparenten Flächen zusätzlich erfasst werden. Da diese Flächen nicht zum erfassenden Objekt gehören, müssen diese anschließend entfernt werden, um die Korrektheit des Modells zu gewährleisten.
- **Reparatur:**
  Bei der Erfassung können geometrische Fehler entstehen. Dazu gehören Löcher, falsch ausgerichtete Normalen oder auch ungenaue Kanten. Für die Korrektheit des Scans müssen die Löcher gefüllt, Kanten begradigt und Normalen berichtigt werden.

2.3 Datenformate

Es existieren viele verschiedene Datenformate für die Speicherung von 3D-Modellen. Zu den gängigsten Formaten gehören STL, OBJ, PLY, DXF, VRML und 3DS [11]. Das STL-Format wurde für die Stereolithografie von der Firma 3DSystems entwickelt. Die Stereolithografie ist das Prinzip des Materialisierens eines Objektes im freien Raum (3D-Druck). STL kann als Industriestandard bezeichnet werden und wird sowohl von Maschinen-, als auch von Softwareherstellern häufig genutzt. Das OBJ-Format und das PLY-Format wurden als freie

Formate entwickelt. VRML ist als Erweiterung von HTML eine Modellierungssprache für die Beschreibung dreidimensionaler Modelle. DXF und 3DS sind von der Firma Autodesk entwickelte Formate, welche von dessen Grafikprogrammen genutzt werden. Da die spätere Werkzeugkette das STL- und PLY-Format unterstützen soll, werden diese näher erläutert.

- **PLY-Datenformat**:
  Das PLY-Datenformat ist als *Polygon File Format* oder auch als *Standford Triangle Format* bekannt [11, 12]. 3D-Modelle werden bei diesem Format durch Polygone dargestellt. Die Grundlage des Formats bilden eine Liste von Punkten und eine Liste von Flächen. Eine PLY-Datei besteht aus einem Kopf und einem Körper. Innerhalb des Kopfes, der durch die Schlüsselwörter `ply` und `end_header` begrenzt ist, wird die Struktur des Körpers und die Anzahl der vorhandenen Elemente beschrieben. Der Körper kann entweder in leserlicher ASCII-Codierung oder in speichereffizienter Binär-Codierung geschrieben werden.

```
ply
format ascii 1.0
element vertex 8
property float x
property float y
property float z
element face 6
property list uchar int vertex_index
end_header
0 0 0
0 0 1
0 1 1
0 1 0
1 0 0
1 0 1
1 1 1
1 1 0
4 2 3 1 0
4 7 6 5 4
4 0 4 6 2
4 5 1 3 7
4 3 2 6 7
4 5 4 0 1
```

Code 1: Einheitswürfel im PLY-Format (ASCII) nach [12]

In Code 1 ist ein Einheitswürfel im PLY-Format zu sehen, welcher folglich aus acht

Knoten und sechs Flächen besteht. Jeder Knoten enthält wiederum drei Koordinaten vom Datentyp `float`. Die einzelnen Flächen bilden sich aus vier Referenzen der entsprechenden Eckpunkte. Aufgrund der Nutzung von Referenzen erfolgt keine redundante Speicherung der Knoten. Die ausschließliche Verwendung von Polygonen führt allerdings zu einem Annäherungsproblem, da Rundungen nicht exakt in diesem Format abgebildet werden können. Neben Knoten und Flächen ermöglicht das PLY-Format auch das Speichern weiterer Attribute wie beispielsweise die Farbe der Punkte. Die Kompaktheit des Formates führt dazu, dass bei Änderungen des Modells die Datei neu erstellt werden muss und nicht nur bestimmte Abschnitte innerhalb der Datei angepasst werden können.

- **STL-Datenformat**:
  Das STL-Datenformat wurde als Schnittstelle für die Stereolithographie (abgekürzt STL), welches das Grundprinzip zum Beispiel des 3D-Drucks ist, entwickelt [11]. Im Laufe der Zeit kamen *Standard Transformation Language*, *Surface Tessellation Language* oder auch *Standard Triangulation Language* als Namen hinzu [13]. Anders als bei dem PLY-Datenformat gibt es keinen Kopf, welcher die Beschreibung der darauffolgenden Daten beinhaltet. Eine STL-Datei beginnt mit dem Schlüsselwort `solid` und endet mit `endsolid`. Dazwischen werden ausschließlich Dreiecke durch ihre Eckpunkte und ihrer Normalen beschrieben. Darüber hinaus müssen sich benachbarte Dreiecke stets zwei Punkte teilen und infolgedessen besitzen diese eine vollständig gemeinsame Kante. Auch bei diesem Datenformat können Rundungen nur durch Annäherung durch eine Vielzahl von Dreiecken beschrieben werden, wodurch eine gewisse Ungenauigkeit entsteht. Diese Annäherung und die vorhandene Redundanz führt schnell zu einer sehr großen Datenmenge, was das STL-Format viel speicherintensiver als das PLY-Format macht. Es besteht aber ebenfalls die Möglichkeit, eine STL-Datei in ASCII- oder Binär-Codierung zu speichern.
  In Code 2 ist ein Auszug des selben Einheitswürfels zu sehen. Dieser wurde durch die Konvertierung der PLY-Datei erzeugt. Es sind die ersten beiden der zwölf Dreiecke zu sehen, welche den Würfel bilden. Es fällt auf, dass das STL-Datenformat nicht frei von Redundanz ist. Die Eckpunkte werden durchgehend für jedes Dreieck vollständig angegeben und somit werden die Punkte vielfach abgespeichert. Der Vorteil der redundanten Speicherung ist, dass zum Beispiel eine lokale Änderung durchgeführt werden kann, ohne dass die restlichen Dreiecke überarbeitet werden müssen. Diese Möglichkeit macht dieses Format so geeignet für Verfahren nach dem Prinzip der Stereolithographie, da ein Modell problemlos in jeder Höhe durchschnitten und in Schichten aufgeteilt werden kann.

```
    solid
        facet normal 1.0 0.0 0.0
            outer loop
                vertex  0.0 1.0 0.0
                vertex  0.0 1.0 1.0
                vertex  0.0 0.0 1.0
            endloop
        endfacet
        facet normal 1.0 0.0 0.0
            outer loop
                vertex  0.0 1.0 0.0
                vertex  0.0 0.0 1.0
                vertex  0.0 0.0 0.0
            endloop
        endfacet
        ...
    endsolid
```

Code 2: Auszug des Einheitswürfels im STL-Format (ASCII) abgeleitet von Code 1

2.4 Verwandte Arbeiten

Für die Identifikation von Bauteilen gibt es verschiedene Ansätze, die einen Vergleich von dreidimensionalen Modellen ermöglichen. So haben Mehalawi und Miller [14, 15] eine Anwendung entworfen, in der mechanische Bauteile mittels Graphen repräsentiert, in einer Datenbank abgelegt und miteinander verglichen werden können. Ziel der Anwendung ist es, dass bereits vorhandene Konstruktionen mit ähnlichen Aufbau identifiziert werden sollen. Die Bauteile liegen als CAD-Modelle im STEP-Format vor. Durch Analyse der STEP-Datei wird der Graph eines Modells abgeleitet. Die Knoten repräsentieren die Flächen des Modells und sind mit einer Kante verknüpft, wenn es sich um benachbarte Flächen mit einer gemeinsamen Kante handelt. Knoten und Kanten sind jeweils mit mehreren Attributen versehen. Die Knoten beinhalten noch die Art und Ausrichtung der Oberfläche. Eine Kante beinhaltet neben den Knoten, die sie verbindet, noch Art, Länge und Richtung. In Abbildung 3 sind zwei Modelle mit ähnlichem Aufbau zu sehen. Beide Modelle werden durch den abgebildeten Graphen abstrahiert. Für die Indexierung dienen die Anzahl der verschiedenen Flächen. Dabei werden die Anzahl aller Flächen ohne die ringförmigen Flächen, die Anzahl ebener Flächen und die Anzahl von kegelförmigen und zylindrischen Flächen zusammen zum Vergleich herangezogen. Es wird die Distanz dieser Mengen zwischen zweier Graphen berechnet, wobei diese sich durch die Division des Betrages der Differenz durch das Maximum der jeweiligen Größe der Mengen ergibt. Insofern die Summe aller Distan-

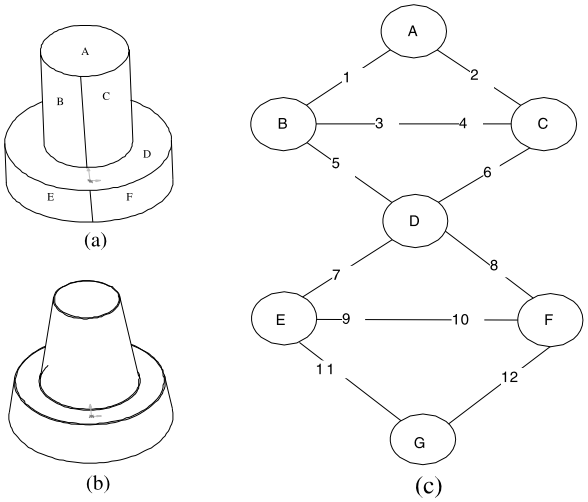


Abbildung 3: Einfache Modelle (a, b) und deren Graph (c) [15]

zen für diese Mengen einen benutzerdefinierten Grenzwert nicht überschreitet, werden die gefundenen Graphen anhand dieser Summe aufsteigend sortiert und können zur Ähnlichkeitsbestimmung herangezogen werden. Für diese Bestimmung werden die Knoten wieder in Mengen von ebenen Flächen, von kegelförmigen und zylindrischen Flächen, von kugelförmigen Flächen und von begrenzten Flächen unterteilt. Anschließend werden alle Knoten einer Menge der beiden Graphen untereinander gepaart. Für jedes Paar wird die Distanz basierend auf der Anzahl der vorhandenen Kanten bestimmt und nach dieser Distanz ansteigend sortiert. Dann werden die Attribute der Kanten des Paares von Knoten mit der geringsten Distanz verglichen. Wenn die Art und Ausrichtung sowie der Typ des Knotens, der durch diese Kante verbunden ist, identisch sind bzw. unter dem Schwellwert liegen, dann werden diese Knoten als kompatibel angesehen. Alle verbleibende Knotenpaare, die einen dieser kompatiblen Knoten enthält, werden entfernt. Dieser Vorgang wird mit allen verbleibenden Paaren wiederholt, bis alle Paare aus der Menge entfernt wurden. Dieser Prozess wird für die verschiedenen Mengen durchgeführt. Das Maß der Ähnlichkeit zweier Graphen wird aus der Anzahl gefundener kompatibler Knotenpaare, kompatibler Kanten und der Gesamtanzahl von Knoten der Graphen bestimmt. Die Autoren testeten ihren Ansatz mit einer Vielzahl einfacher zylindrischer und kastenförmiger Modelle. Dabei wurden auch für ähnliche Modelle die höchsten Ähnlichkeitsmaße erzielt. Modelle, die sich sehr ähnlich waren und sich lediglich durch zusätzliche Bohrungen unterschieden, wurden bei diesem Ansatz und dem geringen Grenzwert aufgrund der fehlenden zylindrischen Flächen aussortiert und deswegen nicht verglichen. Die zylindrischen Modelle aus Abbildung 3 erzielten eine Ähnlichkeit von 100 %. Abbildung 4 zeigt die komplexesten Modelle, für die das Maß der Ähnlichkeit bestimmt wurde. Dabei wurde eine Ähnlichkeit von 86 % errechnet. Die durchgeführten Testreihen

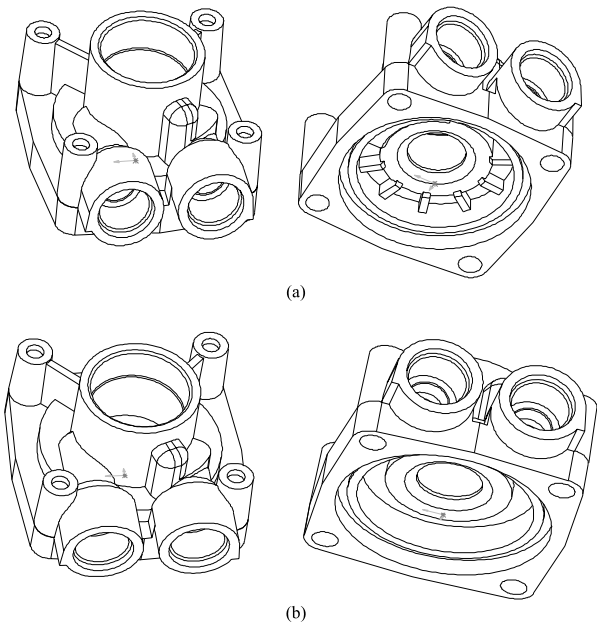


Abbildung 4: Testmodelle mit komplexer Struktur mit zusätzlichen Merkmalen auf der Innenfläche in (a) gegenüber (b) [15]

zeigten, dass der Ansatz funktioniert und auch mit komplexeren Modellen umgehen kann. Neben der Aussortierung ähnlicher Objekte wegen fehlender Bohrungen wurde auch noch ein weiteres Problem identifiziert. Die Position von Merkmalen wie Bohrungen wurden nicht ausreichend mit einbezogen und führt daher zu falschen Zuordnungen bei denen Bohrungen beispielsweise auf einer anderen Seite des Modells zu finden waren.

Während der Ansatz von Mehalawi und Miller [14, 15] dem Vergleich von CAD-Modellen miteinander dient, entwickelten Ip und Gupta [16] einen Ansatz zum Vergleich von CAD-Modellen mit 3D-Scans. Dieser Ansatz basiert ebenfalls auf eine vergleichende Analyse von Flächen. Bei dem 3D-Scan handelt es sich allerdings um kein vollständig digitalisiertes Teil, sondern lediglich um einen Scan des Teils aus einem einzigen Blinkwinkel. Dadurch soll die notwendige Vorarbeit zur Durchführung des Ansatzes so gering wie möglich gehalten werden. Zunächst werden der 3D-Scan und das CAD-Modell auf einzelne Flächen reduziert. Die Segmentierung der Flächen erfolgt durch die Analyse der Krümmungen des 3D-Scan oder des Modells. Innerhalb des 3D-Scans werden die Normalenvektoren für die einzelnen Punkte mithilfe ihrer Nachbarschaft bestimmt. Bei dem CAD-Modell werden die Normalen anhand der Vernetzung hergeleitet. Die so erhaltenen separaten Flächen vom Scan und vom Modell sollen einander zugeordnet werden. Die Zuordnung geschieht anhand des Flächeninhalts und der Krümmung. Nur wenn diese gleichermaßen ausgeprägt sind und die Fläche des Scans vollständig von der dazugehörigen Fläche des Modells bedeckt ist, wird diese Fläche für die Ausrichtung des Modells genutzt. Mit der Hauptkomponentenanalyse (Principle Com-

ponent Analysis, PCA) wird eine Transformation zur Ausrichtung der Flächen ermittelt, wobei die Eigenwerte von Kovarianzmatrizen genutzt werden. Die Kovarianzmatrix wird für die Flächen des 3D-Scans aus der summierten Distanz der Punkte zu ihrem Schwerpunkt und für die Flächen des CAD-Modells aus der summierten Distanz der Mittelpunkte der Dreiecke zum Mittelpunkt der Fläche hergeleitet. Wenn zwei korrekte Flächen zueinander ausgerichtet sind, dann ist die Distanz der Punkte der beiden Flächen sehr gering. Die blau

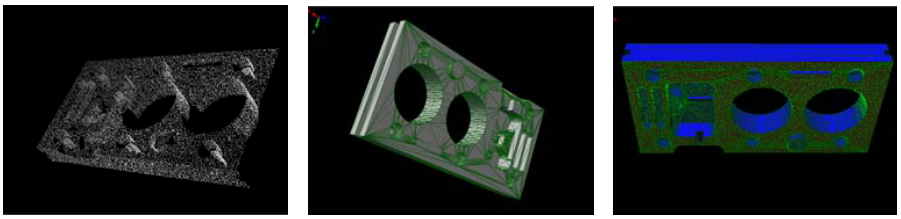
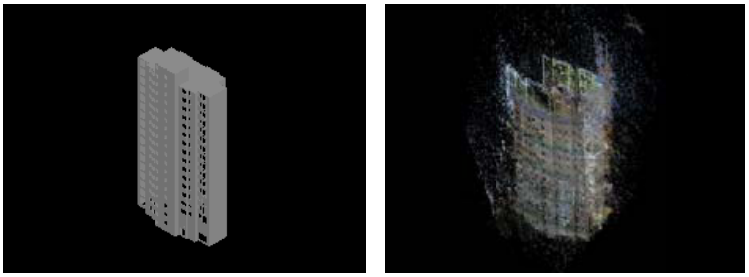


Abbildung 5: 3D-Scan, CAD-Modell und ihre Ausrichtung anhand der markierten Flächen [16]

hervorgehobenen Flächen in Abbildung 5 waren die größten zu identifizierenden Flächen. Diese waren somit die Eingabe für die Durchführung des beschriebenen Ansatzes. Dabei wurde das korrekte CAD-Modell bestimmt und mit einer durchschnittlichen Abweichung von 0.39 mm ausgerichtet. Es wurden noch weitere Modelle erfolgreich getestet. Damit wurde gezeigt, dass mithilfe dieses Ansatzes 3D-Scans mit CAD-Modellen anhand ihrer Flächen direkt verglichen und ausgerichtet werden können. Allerdings wurde auch angemerkt, dass für komplexere Freiformen eine Segmentierung basierend auf der Krümmung möglicherweise nicht ausreichend ist. Des Weiteren könnte die Einbindung von Deskriptoren als weiteres Beschreibungsmerkmal der Oberflächen oder die Einbindung des Anwenders, der die erkannten Flächen priorisiert, den Ansatz optimieren.

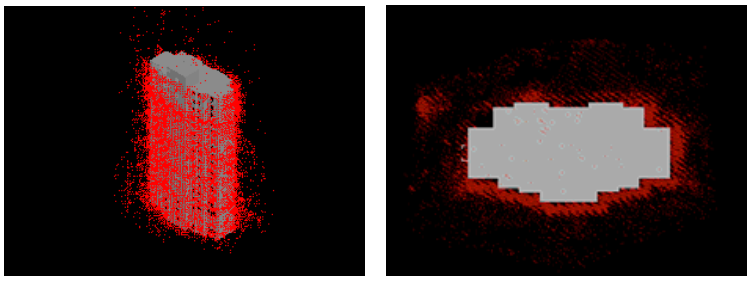
Während die Ansätze von Mehalawi/Miller [14, 15] und Ip/Gupta [16] die Problematik der Identifizierung adressieren, beschäftigten sich Kim, Lee, Cho und Kim [17] mit der genauen Ausrichtung eines 3D-Scans und eines CAD-Modells. Die Ausrichtung wird in zwei Schritten durchgeführt. Zuerst wird eine grobe Ausrichtung bestimmt. Die Bestimmung der groben Ausrichtung erfolgte ebenfalls mittels Hauptkomponentenanalyse. Für die feine Ausrichtung wurde der optimierte *Iterative Closest Point* (ICP) Algorithmus von Levenberg und Marquardt genutzt. Während der normale ICP-Algorithmus lediglich versucht, die Distanz zwischen allen Korrespondenzen zu minimieren, wird bei dem optimierten Verfahren eine Gewichtung und ein Schwellenwert eingeführt, um den Einfluss schlechter Korrespondenzen, die durch verrauschte Daten verursacht werden, zu minimieren. Durch die Ausrichtung eines dreidimensionalen Modells mit einem 3D-Scan eines Gebäudes (siehe Abbildung 6) während dessen Konstruktion sollte der Ansatz getestet werden. Das CAD-Modell wurde aus vorhandenen zweidimensionalen Zeichnungen abgeleitet und der 3D-Scan mit einem kommerziellen



(a) (b)

Abbildung 6: CAD-Modell (a) und 3D-Scan (b) des Gebäudes [17]

3D-Scanner erzeugt. Während das CAD-Modell aus 125.957 Punkten bestand, enthielt der 3D-Scan 921.782 Punkte. Um vergleichbare Daten zu erhalten, wurden beide Modelle so reduziert, dass sie eine ähnliche Auflösung vorwiesen. Dadurch bestand das Modell nur noch aus 22.522 Punkten und der 3D-Scan aus 19.015 Punkten.



(a) (b)

Abbildung 7: Ausrichtung in der Vollansicht (a) und Draufsicht (b) [17]

Abbildung 7 zeigt das erzielte Ergebnis des Verfahrens. Der Ansatz wurde als effizient, vollautomatisiert und robust gegenüber verrauschten Scandaten bewertet. Weitere Testläufe in praktischen Anwendungsbereichen wie der Überwachung der Gebäudekonstruktion sollten die Qualität des Ansatzes weiter überprüfen.

3 Entwurf

In diesem Kapitel wird der Entwurf der zu implementierenden Werkzeugkette erarbeitet und beschrieben. Als Grundlage dafür dienen die bestehenden Vorarbeiten zur Identifizierung von Politz [6] und zur Registrierung von Mertens [7] für dreidimensionale Objekte. Dafür wird zunächst eine genauere Analyse der vorhandenen Teillösungen vorgenommen. Durch diese Analyse sollen die wesentlichen Bestandteile der jeweiligen Prozesse, deren Vor- und Nachteile sowie die ebenfalls für die Werkzeugkette notwendigen Verfahren identifiziert und extrahiert werden. Auch werden verschiedene Algorithmen zur Segmentierung vorgestellt, da diese ebenfalls innerhalb der Werkzeugkette notwendig sein werden. Abschließend wird die komplette Werkzeugkette, wie sie zur Umsetzung der Problemstellung implementiert werden soll, konkret konzipiert.

3.1 Analyse der bestehenden Identifikation

3.1.1 Konzept

Das Identifizieren von Objekten lässt sich im Allgemeinen in die folgenden Schritte unterteilen [18]:

1. **Extrahieren signifikanter Punkte:**
   Die Bestimmung von signifikanten Punkten, sogenannten *Keypoints*, ist aus zweierlei Gründen wichtig. Einerseits wird die zu bearbeitende Datenmenge für die darauffolgenden Berechnungen durch die Reduzierung auf eine Teilmenge von Punkten deutlich kleiner und die Berechnungszeit damit kürzer. Anderseits sind nicht alle Punkte innerhalb eines Objektes gleichermaßen für die Identifizierung geeignet. Die wichtigste Eigenschaft eines *Keypoint*s ist eine hohe Unverwechselbarkeit. Je besonderer ein Punkt ist, desto wahrscheinlicher ist eine Ähnlichkeit zwischen zwei Objekten, wenn sich eine Vielzahl solch signifikanter Punkte wiederfinden lässt.

2. **Beschreibung von Merkmalen:**
   Um die bestimmten *Keypoints* miteinander vergleichen zu können, müssen diese vergleichbare Merkmale erhalten. Dafür werden Merkmalsvektoren, sogenannte *Features*, für alle *Keypoints* bestimmt. Diese *Features* werden mithilfe der umliegenden Nachbarschaft des jeweiligen *Keypoints* oder für ganze Objekte berechnet. Durch diese abstrahierte Beschreibung der betrachteten Oberfläche oder Objekte als mathematisches Element kann die Ähnlichkeit rechnerisch ermittelt werden.

3. **Berechnung von Korrespondenzen:**
   Wenn die *Features* bestimmt sind, müssen Korrespondenzen zwischen den verschie-

denen Mengen gebildet werden. Bei der Beschreibung der *Features* durch Vektoren kann der in Gleichung 1 beschriebene euklidische Abstand als Maß für die Ähnlichkeit genutzt werden.

$$d(x, y) = ||x - y||_2 = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2} \tag{1}$$

Dabei wird für jedes *Feature* eine Korrespondenz mit minimalem Abstand gesucht. Jedoch muss der minimale Abstand nicht zwangsläufig eine gute Korrespondenz ausmachen. Daher ist nach Bildung aller Korrespondenzen eine Auswertung zur Erhaltung gültiger und zur Verwerfung ungültiger Korrespondenzen notwendig.

3.1.2 Bestimmung der *Keypoints*

Die Auswahl der *Keypoints* kann entsprechend der verwendeten Verfahren und bestimmter Eigenschaften variieren. So können für manche Verfahren eher Punkte auf Kanten oder auf besonders stark gekrümmten Flächen von Vorteil sein. Dafür wäre allerdings eine vorhergehende Analyse sämtlicher Punkte notwendig. Die Identifizierung von Politz [6] wurde unter zeitkritischen Umständen betrachtet. Aus diesem Grund sollte der Rechenaufwand möglichst gering gehalten und eine leichtgewichtige Bestimmung der *Keypoints* umgesetzt werden. Infolgedessen erfolgte diese durch das Rastern der Modelle mittels Voxeln. Dabei wurde angenommen, dass durch das Rastern zwar schlechte *Keypoints* mit geringer Aussagekraft entstehen werden, aber auch ausreichend viele gute *Keypoints*, anhand derer eine zuverlässige Identifikation möglich ist. *Keypoints* mit geringer Aussagekraft sind beispielsweise Punkte auf Ebenen, die sich von keinem Punkt auf einer beliebigen, anderen Ebene unterscheiden und daher nicht eindeutig zuzuordnen wären. Demzufolge sind *Keypoints*, die eine sehr signifikant geformte Umgebung besitzen, sehr aussagekräftig. Ein Voxel ist ein dreidimensionaler Pixel mit meist würfelförmigem Aufbau. Die Auflösung eines Voxelmodells wird durch die Kantenlänge der einzelnen Voxeln bestimmt. Durch das Stapeln dieser entsteht ein vollständiges gefülltes Modell. Die Bestimmung der *Keypoints* ist somit eigentlich eine Reduktion der vorhandenen Punktmenge. Dabei wird die Punktwolke in ein Voxelgitter gelegt und anschließend werden alle Punkte, die innerhalb eines Voxels liegen, durch ihren Schwerpunkt approximiert. So kann eine Punktwolke in Abhängigkeit der Voxelgröße reduziert und die resultierenden Punkte als *Keypoints* verwendet werden. Dabei wurden verschiedene Kantenlängen für das Voxelgitter und dessen Auswirkungen auf das Ergebnis der Identifikation untersucht. Eine Kantenlänge von $2,0$ und $4,0$ mm (Maßeinheit wird durch das Modell bestimmt) ermöglichten eine ausreichende Generierung von *Keypoints*, um die Objekte zuverlässig zu identifizieren.

3.1.3 Bestimmung der *Features*

Die Auswahl des geeigneten *Features* erfolgte anhand der folgenden Bewertungskriterien. Das verwendete Verfahren sollte für Punktwolken geeignet sein, da die Objekte in dieser Form als Eingabe übergeben werden. Das Verfahren sollte unabhängig von der Textur der Objekte sein. Das bedeutet, dass das *Feature* auch ohne Farbinformationen aussagekräftige und zuverlässige Ergebnisse liefern soll. Generell sollte das *Feature* ausreichend robust sein, da durch das Erfassen mittels eines 3D-Scanners immer mit Ungenauigkeiten gerechnet werden muss. Des Weiteren können die Objekte selbst durch Abnutzungen oder Defekte zu verfälschten digitalen Modellen führen. Daher sollte das genutzte *Feature* bestenfalls mit lokalen Abweichungen umgehen können und dennoch eine Zuordnung anhand der Geometrie des Objektes ermöglichen. Das letzte Kriterium war die Unabhängigkeit von globalen Koordinatensystemen. Das während der Erfassung entstehende Koordinatensystem des Modells darf nicht von Bedeutung sein, da das Objekt je nach Blickwinkel oder Nachbearbeitung verschieden in dessen Koordinatensystem liegt. Würde das Verfahren dieses Koordinatensystem als Bezugspunkt nutzen, wäre eine erfolgreiche Identifikation nur dann möglich, wenn die zu vergleichenden Objekte die gleiche Position im Raum besitzen würden. Andernfalls müssten für alle Objekte mehrere Aufnahmen aus verschiedenen Blickwinkeln vorliegen und all diese während der Identifikation für den Vergleich herangezogen werden.

Bei der Identifikation wurden verschiedene Szenarien untersucht. Dabei wurde in die Identifizierung von eindeutigen, unvollständigen Objekten unterteilt und die Identifikation von unvollständigen Objekten mit einzelnen Referenzobjekten und innerhalb einer Szene vorgenommen. Konkret wurden *Ensemble of Shape Functions* [19] (*ESF Feature*) und *Signature of Histograms of Orientations* [20] (*SHOT Feature*) dafür genutzt. Das *ESF Feature* ist dabei als globales und das *SHOT Feature* als lokales Verfahren einzuordnen. Während bei einem globalen Verfahren alle verfügbaren Punkte für die Berechnung hinzugezogen werden, wird bei lokalen Verfahren nur eine lokal begrenzte Menge genutzt und eine Vielzahl von *Features* bestimmt.

- ***ESF Feature*:**
  Bei diesem Verfahren werden verschiedene Funktionen angewendet und in Histogrammen abgebildet, um eine mathematische Beschreibung des Objektes zu erhalten. Die erste Funktion $D2$ ist eine Distanzfunktion. Hierfür werden zwei zufällige Punkte gewählt, verbunden und die Distanz zwischen diesen Punkten berechnet. Die ermittelte Distanz wird in einem Histogramm abgebildet. Dabei existieren drei Klassen von Histogrammen für Linien die innerhalb des Modells (IN), außerhalb des Modells (OUT) oder dazwischen (MIXED) liegen. Zusätzlich existiert ein viertes Histogramm, indem das Verhältnis jeder Linie, die innerhalb und außerhalb des Modells liegt, abgebildet

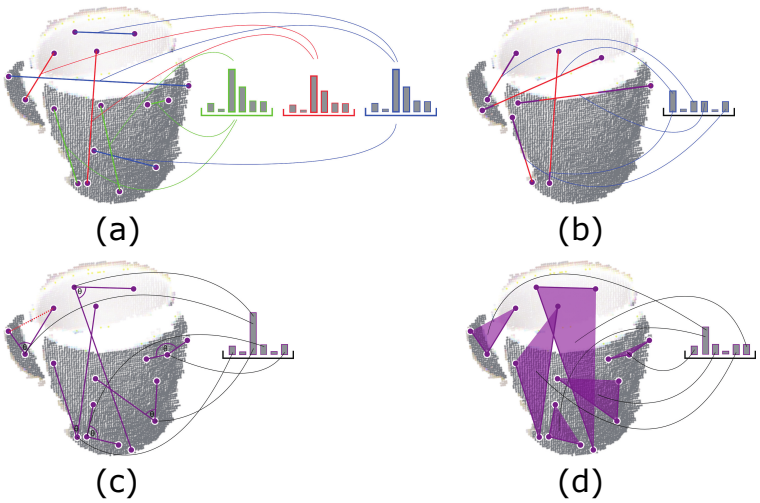


Abbildung 8: *ESF Feature* und deren Funktionen, bestehend aus Funktion $D2$ (a), Verhältnis der Linien zum Modell (b), Funktion $A3$ (c) und Funktion $D3$ (d) [19]

wird. Die zweite Funktion $A3$ ist eine Winkelfunktion für drei zufällig gewählte Punkte. Diese Funktion bestimmt den Winkel zwischen zwei Linien. Dieser Winkel wird ebenfalls in ein Histogramm der Klassen IN, OUT und MIXED abgebildet. Allerdings bestimmt die dem Winkel gegenüberliegende Linie die Zuordnung zu den Histogrammen. Die letzte Funktion $D3$ ist eine Flächenfunktion, bei der die Fläche des Dreiecks, dass durch die zufällig gewählten Punkte aufgespannt wird, errechnet wird und in drei weitere verschiedene Histogramme der bekannten Klassen abgebildet wird. So entsteht bei diesem Verfahren eine Beschreibung des Objektes durch zehn verschiedene Histogramme, welche alle jeweils in 64 Unterklassen aufgeteilt sind. In Abbildung 8 sind die Funktionen an einem Beispiel dargestellt, wobei lediglich für die Funktion $D2$ alle drei Histogramme der Klassen IN, OUT und MIXED visualisiert sind. Um einen zuverlässigen Vergleich der Histogramme zu ermöglichen, werden diese normalisiert. Als Maß wird die in Gleichung 2 gezeigte $L1$ Distanz verwendet, welche als Summe der absoluten Differenzen von Vektoren oder Matrizen beschrieben werden kann.

$$d(x, y) = ||x - y||_1 = \sum |x_i - y_i| \tag{2}$$

Die Vorteile dieses Ansatzes gegenüber anderen *Features* hinsichtlich der Rechenzeit ist die konstante Anzahl von Rechenschritten. Die Funktionen wählen stets eine feste Anzahl von Punkten aus und somit ist der Ansatz nahezu unabhängig von der Größe und Auflösung des Modells. Der Nachteil dieses Ansatzes ist die zufällige Wahl der Punkte für die Berechnung des *Features*. Dieser Nichtdeterminismus kann selbst bei dem selben Objekt zu nicht vollständig identischen *Features* führen. Dennoch entstehen durch eine genügend große Anzahl an gewählten Punkten ähnliche Approximationen

für ähnliche Objekte, wodurch eine Zuordnung möglich wird.

- ***SHOT Feature*:**
  Bei diesem Verfahren werden nicht nur durch Histogramme repräsentierte Häufigkeiten topologischer Eigenschaften genutzt, sondern auch konkrete topologische Informationen hergeleitet und zum Aufbau des *Features* verwendet. Die konkrete Topologie wird durch lokale und reproduzierbare Koordinatensysteme beschrieben, welche für die einzelnen *Keypoints* erzeugt werden. Für die Erzeugung der lokalen Koordinatensysteme dient die jeweilige Nachbarschaft. Für diese Nachbarschaft wird eine Kovarianzmatrix mithilfe einer gewichteten Linearkombination aufgebaut und anhand derer Eigenvektoren ein Koordinatensystem abgeleitet. Die Richtung wird anhand des mehrheitlichen Vorzeichens des Skalarproduktes vom Eigenvektor mit der Differenz aus *Keypoint* und den jeweiligen Punkten gewählt. So werden die X-Achse und Z-Achse bestimmt und die Y-Achse als Kreuzprodukt abgeleitet. Dieses Verfahren zur Erzeugung lokaler Koordinatensysteme ist unabhängig gegenüber Translation und Rotation und kann so zuverlässig für jede Lage im Raum reproduziert werden. Zur Berechnung des *Features* wird anhand des lokalen Koordinatensystems und des angegebenen Radius der Nachbarschaft ein kugelförmiges Bezugssystem aufgebaut und in mehrere Sektoren unterteilt, wie in Abbildung 9 abgebildet. Für jeden Sektor wird ein Histogramm mit einer Größe

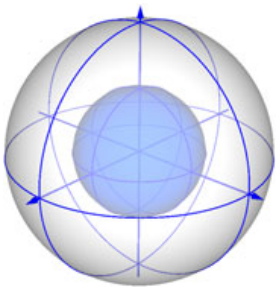


Abbildung 9: Kugelförmiges Bezugssystem für das *SHOT Feature* [20]

von 32 Klassen erstellt, welche die Häufigkeit des Kosinus der Schnittwinkel der Normalen vom *Keypoint* und allen Punkten innerhalb des Sektors abbilden. Abschließend werden die Histogramme zusammengefasst und normalisiert. So wird der Vergleich trotz verschiedener Auslösungen innerhalb des Bezugssystems ermöglicht. Der Vorteil dieses Verfahrens ist die hohe Robustheit gegenüber unterschiedlichen Auflösungen und verschiedenen Lagen im Raum. Diese Robustheit wird allerdings erst durch viele Vorberechnungen erreicht. So müssen für alle Punkte die Normalen und für alle *Keypoints* die lokalen Bezugssysteme bestimmt werden, um das *Feature* berechnen zu können.

3.1.4 Bestimmung der Korrespondenzen

Für das lokale *SHOT Feature* ist es notwendig, eine Vielzahl von Korrespondenzen zwischen den zu vergleichenden Objekten zu bilden. Dafür werden mithilfe von Bäumen nächste Nachbarn für die einzelnen *Features* gesucht. Im Speziellen handelt es sich bei diesen Bäumen um k-d-Bäume [21]. Hierbei handelt es sich um mehrdimensionale Binärbäume, die eine effizi-

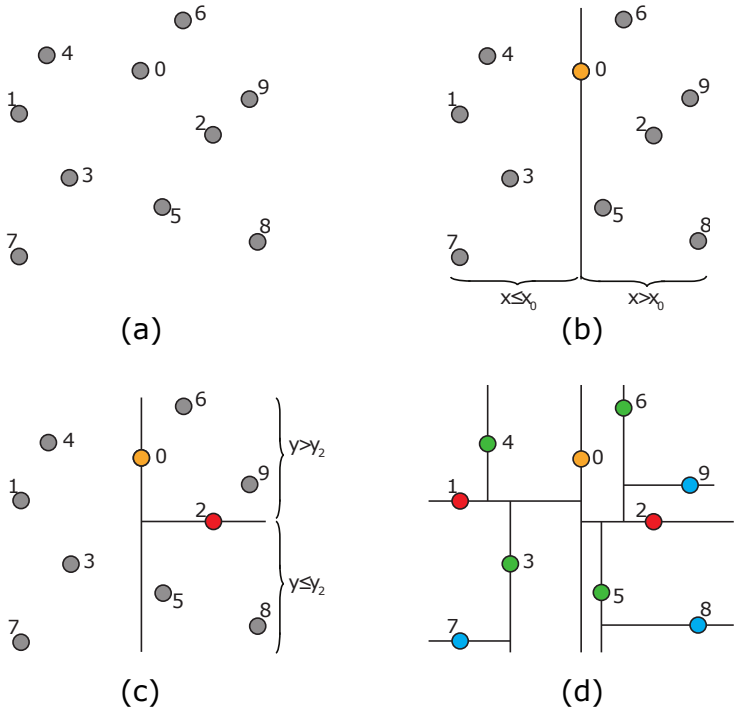


Abbildung 10: Aufbau eines k-d-Baumes [22]

ente Suche des nächsten Nachbarn ermöglichen. Bei dem Aufbau eines k-d-Baumes werden die Daten in jeder Tiefenebene anhand einer Dimension aufgeteilt. In Abbildung 10 ist dieser Vorgang für zweidimensionale Daten (a) dargestellt. In der ersten Iteration wird der Median zur Auswahl des Startknotens genutzt. Der ausgewählte Knoten wird der Wurzelknoten und alle Punkte mit einer kleineren oder gleichen x-Komponente bilden den linken Teilbaum und alle Punkte mit einer größeren x-Komponente bilden den rechten Teilbaum (b). In der zweiten Iteration werden die Teilbäume entsprechend der nächsten Komponente gebildet (c). Dieser Prozess wird wiederholt, bis alle Punkte dem Baum hinzugefügt wurden (d). Die so entstandenen Zellen teilen den Raum entsprechend des Baumes auf. Um nun den nächsten Nachbarn eines beliebigen Punktes zu finden, wird der Baum zunächst, wie in Abbildung 11 verbildlicht, entsprechend der Werte des übergebenen Punktes traversiert [23]. Das gefundene Blatt ist der aktuell nächste Nachbar und die Distanz zwischen dem übergebenen Punkt und diesem Nachbarn bestimmt die Nachbarschaft, in der mögli-

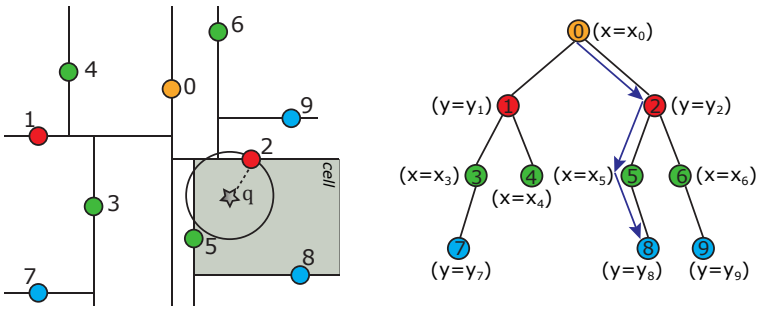


Abbildung 11: Suche des nächsten Nachbarn innerhalb des k-d-Baumes [22]

cherweise noch dichtere Punkte liegen könnten. Um die Nachbarschaft zu überprüfen, wird entlang des traversierten Pfades wieder im Baum in Richtung des Wurzelknotens gegangen. Bei jedem Knoten müssen zwei Bedingungen geprüft werden. Zuerst wird die Distanz zu diesem Knoten bestimmt und falls diese kleiner als die Distanz zum aktuellen Nachbarn ist, wird dieser aktualisiert. Zweitens wird mit dem *Bounds Overlap Ball* Test überprüft, ob die Nachbarschaft des Punktes, die Zelle des anderen Teilbaums schneidet. Wenn dieser Test eine Überschneidung feststellt, muss der andere Teilbaum des aktuellen Knotens iteriert werden. Andernfalls kann der Teilbaum ignoriert werden und es wird wieder ein Knoten entlang des Pfades zurückgegangen. Wenn der Algorithmus wieder am Wurzelknoten angekommen ist, ist der nächste Nachbar des übergebenen Punktes gefunden. Alle *Features* bilden mit dem jeweiligen nächsten Nachbarn eine Korrespondenz.

Zur Auswahl gültiger Korrespondenzen wird das *Geometric Consistency Clustering* Verfahren [18] genutzt. Ziel dieses Clusteringalgorithmus ist es, Korrespondenzen mit ähnlichen geometrischen Eigenschaften zu gruppieren. Dabei werden zunächst alle vorhandenen Korrespondenzen aufsteigend nach ihrer Distanz sortiert. Die erste Korrespondenz $c_i = \{p_i^r, p_i^s\}$, welche die mit minimalen Abstand verknüpfte *Features* besitzt, dient als Ausgangskorrespondenz für das aktuelle Cluster. Anschließend werden alle nicht gruppierten Korrespondenzen mit dieser Korrespondenz verglichen.

$$|||p_i^r - p_j^r||_2 - ||p_i^s - p_j^s||_2| < \epsilon \tag{3}$$

Wenn die aktuelle Korrespondenz $c_j = \{p_j^r, p_j^s\}$ die Bedingung in 3 mit der Ausgangskorrespondenz c_i für den benutzerdefinierten Grenzwert ϵ erfüllt, wird diese Korrespondenz zum Cluster hinzugefügt. Neben dem benutzerdefinierten Grenzwert kann noch die minimale Größe eines Cluster festgelegt werden. Nach der Iteration durch alle Korrespondenzen, wird das Verfahren mit den verbliebenen, nicht gruppierten Korrespondenzen wiederholt, bis alle Korrespondenzen einem Cluster zugewiesen oder verworfen wurden. Als Ergebnis produziert das Verfahren eine Liste von Mengen aus Korrespondenzen. Für die Identifikation wird das größte Cluster ausgewählt und die darin enthaltenen Korrespondenzen als gültige

Korrespondenzen angesehen. Die Größe dieser Menge wird als Maß für die Identifikation angegeben.

3.2 Analyse der bestehenden Registrierung

3.2.1 Konzept

Konzeptionell kann die Registrierung als weiterführender Prozess der Identifikation angesehen werden. Wenn ein Modell identifiziert wurde, ist oft die Berechnung einer Transformation interessant, um das Objekt korrekt im Raum zu positionieren. Demzufolge sind die in Abschnitt 3.1.1 dargestellten Teilschritte auch hier notwendig. Wenn die Korrespondenzen zwischen den *Features* der *Keypoints* gefunden sind, kann eine Ausrichtung der Modelle vorgenommen werden. Die Ausrichtung wird dabei oft in zwei Teilschritte durchgeführt. Zunächst erfolgt eine grobe Vorausrichtung und abschließend eine feine Ausrichtung für ein optimales Ergebnis.

Wie für die Identifikation wurde hier eine leichtgewichtige Bestimmung der *Keypoints* gewählt und erneut mit dem Voxel basierenden Verfahren gerastert. Im Gegensatz zur Teillösung der Identifikation wurden keine verschiedenen *Features* mehr evaluiert, sondern lediglich das *SHOT Feature* genutzt.

3.2.2 Bestimmung der Korrespondenzen

Zwar wurden die Korrespondenzen ebenfalls mit einem k-d-Baum und dessen Suche nach dem nächsten Nachbarn bestimmt, aber für die Auswahl gültiger Korrespondenzen wurde ein anderer Ansatz genutzt, um den Einfluss schlechter Korrespondenzen zu minimieren. Anstatt die Korrespondenzen zu gruppieren, wurden diese gewichtet und anschließend anhand verschiedener Kriterien verworfen. Die Gewichtung ω für eine Korrespondenz mit dem Abstand x zwischen ihren verbundenen *Features* erfolgt mit der gauß'schen Gewichtung aus Gleichung 4, wobei d_{min} und d_{max} für den minimalen bzw. maximalen Abstand einer Korrespondenz innerhalb einer Menge von Korrespondenzen stehen.

$$\begin{aligned} \omega &= \frac{3}{4(m \cdot x + n)^2 + 2} - \frac{1}{2}, \text{ mit} \\ m &= \frac{1}{d_{\max} - d_{\min}} \\ n &= -\frac{d_{\min}}{d_{\max} - d_{\min}} \end{aligned} \tag{4}$$

Die darauffolgende Verwerfung schlechter Korrespondenzen erfolgt in drei Schritten. Zuerst werden mehrere Korrespondenzen gesucht, die gleiche Start- oder Endpunkte besitzen. Wenn

für zwei Korrespondenzen $c_i = \{p_i^r, p_i^s\}$ und $c_j = \{p_j^r, p_j^s\}$ entweder $p_i^r = p_j^r$ oder p_i^s, p_j^s gilt, dann wird die Korrespondenz mit der größeren Distanz verworfen. Dieser Schritt wird wiederholt bis alle verbliebenen Korrespondenzen keine identischen Endpunkte mehr besitzen. Für den nächsten Schritt wird die Standardabweichung berechnet und alle Korrespondenzen, deren Distanz größer als die Standardabweichung ist, verworfen.

$$\sigma_d = \sqrt{\sum (d_i - \mu_d)^2 \cdot p(d_i)} \tag{5}$$

Gleichung 5 zeigt die Standardabweichung, wobei μ_d der Durchschnitt der Distanzen aller zu betrachtender Korrespondenzen und $p(d_i)$ die Wahrscheinlichkeit sowie d_i die Distanz der aktuellen Korrespondenz ist. Im letzten Schritt werden die noch verbliebenen Korrespondenzen reduziert. Dafür werden alle Korrespondenzen aufsteigend nach ihrem Gewicht sortiert und ein vom Benutzer definierter Prozentsatz verworfen. Die restlichen Korrespondenzen wurden für die Registrierung verwendet.

3.2.3 Bestimmung der Ausrichtung

Die Registrierung wurde in eine Grob- und Feinausrichtung unterteilt. Das Ergebnis dieser Ausrichtungen ist jeweils eine Transformationsmatrix, die die Drehung und die Verschiebung beinhalten, um ein Modell möglichst kongruent an der Position des anderen Modells zu bewegen.

Für die grobe Ausrichtung wurde das *Sample Consensus Initial Alignment* (*SAC-IA*) Verfahren [24] genutzt. Bei diesem Verfahren werden zunächst eine Menge von Punkten, sogenannte *Samples*, zufällig ausgewählt. Für jeden dieser *Samples* werden eine feste Anzahl ähnlicher *Features* innerhalb des anderen Modells gesucht und ein *Feature* davon zufällig als Korrespondenz ausgewählt. Nun kann für alle Samples und deren Korrespondenzen die Transformation berechnet werden und die Transformation mit dem geringsten Fehler wird gespeichert. Als Fehlermaß dient der Huber-k-Schätzer aus Gleichung 6. Dabei handelt es sich um eine Funktion, die sich für kleine Eingaben e_i quadratisch und für große Eingaben ab der Tuning-Konstante t_e nur noch linear verhält. Damit soll der Einfluss großer Werte, die als Ausreißer angesehen werden können, minimiert werden. Dieser Vorgang wird einige Male wiederholt und die so randomisiert ermittelte Transformationsmatrix ist zur groben Ausrichtung geeignet.

$$L_h(e_i) = \begin{cases} \frac{1}{2} \cdot e_i^2, & ||e_i|| \leq t_e \\ \frac{1}{2} \cdot t_e \cdot (2 \cdot ||e_i|| - t_e), & ||e_i|| > t_e \end{cases} \tag{6}$$

Für die abschließende Feinausrichtung wird der *Iterative Closest Point* Algorithmus [25] verwendet. Dieses Verfahren benötigt eine grobe Vorausrichtung, da es die Distanzen zwi-

schen den Punkten der Modelle zu minimieren versucht. Die Grundlage des Verfahrens bildet die Bestimmung des dichtesten Punktes innerhalb des anderen Objektes. Der Algorithmus kann die minimale Distanz zwischen einem Punkt und einem Punkt, einer Linie oder eines Dreiecks ermitteln. Für die Registrierung wurde die Variante von Punkt zu Punkt genutzt. Zunächst werden für alle Punkte die dichtesten Punkte innerhalb des anderen Objektes gefunden. Mithilfe der Schwerpunkte der Modelle und der Kreuzkovarianzmatrix wird die Transformation bestimmt, wobei die maximalen Eigenwerte dieser Matrix zur Rotation und die Translation der Schwerpunkte zur Verschiebung genutzt werden. Anschließend wird die bestimmte Transformation angewendet und die Summe der Fehlerquadrate bestimmt. Wenn die Summe aller Fehler unterhalb des gegebenen Grenzwertes liegt oder die maximale Anzahl der vorgegebenen Iterationen erreicht ist, wird der Algorithmus beendet und die resultierende Transformationsmatrix zurückgegeben.

3.3 Zusammenfassung der Teilprozesse

In Tabelle 1 sind alle Gemeinsamkeiten und Unterschiede aus [6] und [7] zusammenfassend dargestellt. Die fest definierten Werten wurden durch die Autoren der Vorarbeiten empirisch bestimmt. Die restlichen Werte werden durch die angegebenen Variablen in Abhängigkeit der Eigenschaften der Objekte bestimmt. Aufgrund der konzeptionellen Gemeinsamkeiten und der gleichen verwendeten Verfahren lassen sich diese beiden Teilprozesse stark miteinander verknüpfen, was von Vorteil für die angestrebte Prozesskette ist.

Lediglich bei den unterschiedlichen Verfahren und genutzten Werten für gewisse Prozessschritte muss eine Auswahl erfolgen. Beide Ansätze verwenden sowohl für die Bestimmung der *Keypoints*, als auch für die Bestimmung der *Features*, die selben Verfahren. Lediglich bei der Belegung der Parameter werden unterschiedliche Werte genutzt. Zwar werden die Korrespondenzen mit der selben Datenstruktur ermittelt, dennoch ist der größte Unterschied bei der Auswertung der Korrespondenzen zu finden.

Um die Auswirkungen der unterschiedlichen Verfahren und Parameter zu visualisieren, wurden die Korrespondenzen für verschiedene Scans desselben Bauteils bestimmt. Abbildung 12 zeigt die erhaltenen Korrespondenzen. Bei den Modellen handelt es sich um ein Bauteil aus einem Einzelscan (unten) und dem selben Bauteil, welches aus einem Gesamtscan der dazugehörigen Baugruppen extrahiert wurde (oben). Für Abbildung 12a ist der Ansatz aus der Identifikation und für 12b der Ansatz aus der Registrierung verwendet worden. Des Weiteren wurden alle Parameter, die für den jeweiligen Ansatz von Bedeutung sind, mit den vorgestellten Standardwerten belegt. Die Abbildung zeigt deutlich, dass die Korrespondenzen der Identifikation wesentlich besser zueinander liegen. Zwar ist die resultierende Anzahl

Tabelle 1: Gemeinsamkeiten und Unterschiede der Teilprozesse

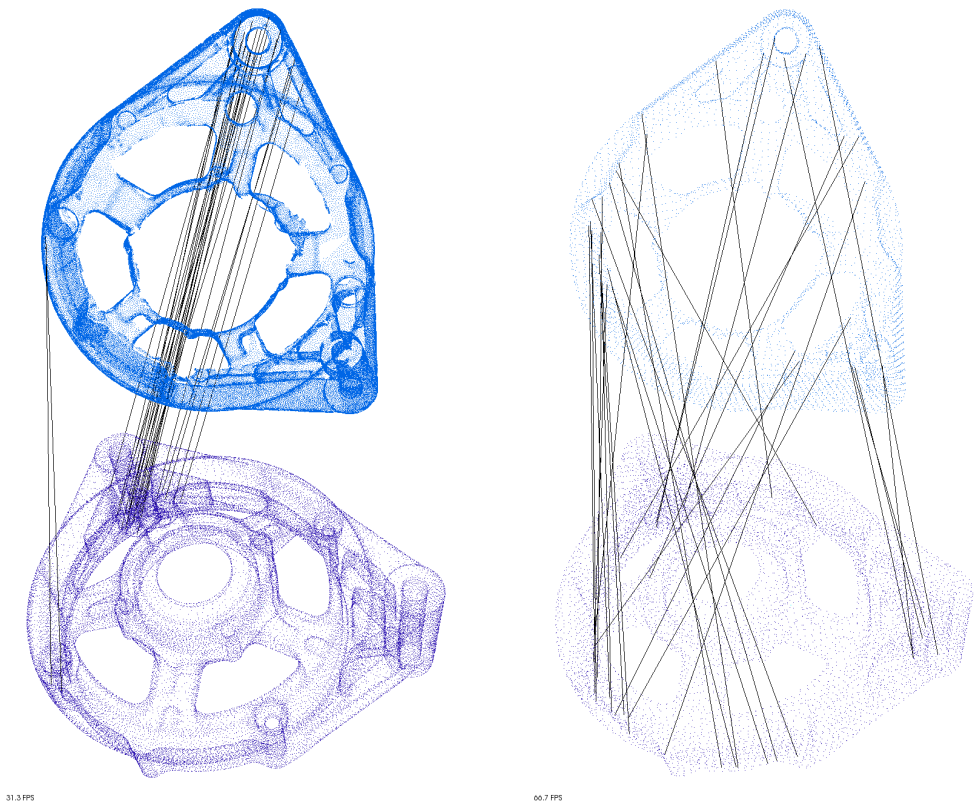
	Identifikation		**Registrierung**	
	Verfahren	Parameter	Verfahren	Parameter
Keypoints	Voxelgitter	$r_v = 4,0$	Voxelgitter	$r_v = \frac{s}{12}$
Features	*SHOT*	$r_d = \frac{s}{2}$, $n_n = 10$	*SHOT*	$r_d = \frac{s}{3}$, $r_n = \frac{s}{6}$
	ESF			
Korrespondenzen	k-d-Baum	$d_f = 0,2$	k-d-Baum	
	Geometric Consisteny Grouping	$s_{gc} = 5 \cdot r_m$, $t_{gc} = \frac{n_p}{400}$		
			Gewichtung und Verwerfung (Gauß, doppelte Korrespondenzen, Standardabweichung, Prozentsatz)	$p = 0,95$
Ausrichtung			*SAC-IA*	$d_s = 0,5$
			ICP	x

r_v : Kantenlänge des Voxelgitters
s : kürzeste Seite des quaderförmigen Hüllkörpers
r_d : Radius des *Features*
n_n : Anzahl der nächsten Nachbarn für Normalenberechnung
r_n : Radius für Normalenberechnung
d_f : maximale Distanz zwischen *Features*
s_{gc} : minimale Clustergröße
r_m : Auflösung des Modells
t_{gc} : Grenzwert für Clusterbildung
p : Prozentsatz zu verwerfender Korrespondenzen
d_s : minimale Distanz der *Samples*

der Korrespondenzen zufälligerweise mit 30 zu 31 beinahe identisch, aber die gemittelte Distanz ist mit $0,171679$ zu $0,326945$ ebenfalls deutlich besser, da die verbundenen *Features* ähnlicher sind, je kleiner deren Distanz ist. Allerdings unterscheiden sich die Ansätze auch in den vorhergehenden Prozessschritten, wodurch das Ergebnis keine eindeutigen Rückschlüsse zulässt. Aus diesem Grund ist für den konkreten Entwurf der Werkzeugkette für den Prozess der Bestimmung zuverlässiger Korrespondenzen eine weitere Evaluation notwendig.

3.4 Segmentierungsalgorithmen

Es existieren eine Vielzahl verschiedener Verfahren zur Segmentierung von Punktwolken. Der *Min-Cut Based Segmentation* Algorithmus dient der Extraktion von Objekten innerhalb ei-



(a) Korrespondenzen für Identifikation (3.1.4)

(b) Korrespondenzen für Registrierung (3.2.2)

Abbildung 12: Vergleich der resultierenden Korrespondenzen der vorgestellten Ansätze

nes Scans, der dafür in Vordergrund und Hintergrund unterteilt wird [26]. Mithilfe eines Nächste-Nachbarn-Graphen, einer übergebenen Startposition und eines Radius soll ein Objekt als Vordergrund identifiziert werden. Bei dem Aufbau des Graphen werden die Kanten abhängig von der Distanz der verbundenen Knoten gewichtet. Wenn dabei unabhängige Zusammenhangskomponenten entstehen, werden diese durch eine Kante zwischen den beiden dichtesten Punkten wieder verbunden. Für die Segmentierung wird eine Hintergrundbedingung genutzt, welche die Distanz mit dem angegebenen Radius normiert vergleicht. Diese Bedingung besteht wiederum aus zwei Komponenten, die sich gegensätzlich verhalten und in Abbildung 13 dargestellt sind. Die erste Komponente (a) ist eine früh, aber langsam ansteigende Funktion, die schlecht mit dem Vordergrund verbundene Punkte als Hintergrund identifizieren soll. Die zweite Komponente (b) ist eine sehr späte und dann stark ansteigende Funktion. Der späte Anstieg soll dicht zu dem angegebenen Radius liegen und alle Punkte nahe dem Radius ebenfalls als Hintergrund markieren. Durch den aufgebauten Graph und der konstruierten Hintergrundbedingung kann eine Aufteilung in Vordergrund und Hintergrund durchgeführt werden. Der Radius ist dabei entscheidend für eine korrekte Aufteilung.

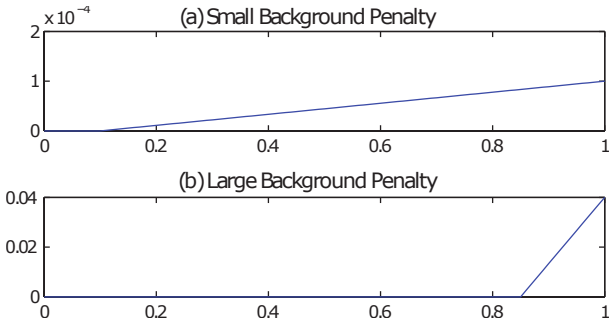


Abbildung 13: Darstellung der Hintergrundbedingung mit ihren einzelnen Komponenten (a) und (b) [26]

Der *Euclidean Cluster Extraction* [27] Algorithmus unterteilt eine Punktewolke in verschiedene Cluster basierend auf dem Abstand der Punkte. Dabei wird eine kugelförmige Umgebung für den aktuellen Punkt bestimmt und alle Punkte innerhalb dieser Umgebung werden zu dessen Cluster hinzugefügt. Der euklidische Abstand dieser Punkte ist somit kleiner oder gleich dem angegebenen Radius. Für diese Punkte wird wiederum die Umgebung bestimmt und dem Cluster weitere Punkte hinzugefügt, insofern diese nicht bereits gefunden wurden. Wenn keine neuen Punkte mehr erreicht werden, wird das Cluster in einer Liste gespeichert. Anschließend wird einer der verbliebenen Punkte ausgewählt und mit der Bildung eines neuen Clusters begonnen. Dieser Prozess wird wiederholt, bis alle Punkte verarbeitet sind. Obwohl dieser Segmentierungsalgorithmus recht einfach scheint, kann er durchaus geeignet sein, um Einzelteile innerhalb des Scans zu segmentieren, insofern ein Teil der Punktwolke einen größeren Abstand zu den restlichen Punkten des Modells besitzt.

Bei dem *Difference of Normals Based Segmentation* Algorithmus [28] spielt der Abstand nur eine untergeordnete Rolle. Zwar müssen für dieses Verfahren zwei unterschiedlich große Radien angegeben werden, diese sind aber nur für die Vorberechnungen und nicht direkt für die eigentliche Segmentierung nötig. Diese Radien definieren für einen Punkt zwei verschieden große Nachbarschaften und für jede dieser Nachbarschaft wird ein Normalenvektor bestimmt. Dabei sollten die Radien ausreichend verschieden sein, um die Auswirkung einer Änderung der Oberfläche auf den Normalenvektor zu übertragen. Anschließend kann die Differenz dieser beiden Vektoren bestimmt werden. Abbildung 14 verbildlicht diese Vorbe-

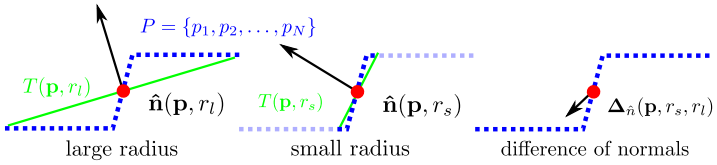


Abbildung 14: Darstellung der Normalen $\hat{n}$ eines Punktes p für verschiedene Radien r_l, r_s und der resultierende Differenzvektor $\Delta_{\hat{n}}$ [28]

rechnung des Verfahrens. Anhand dieses Differenzvektors wird die eigentliche Segmentierung durchgeführt. Die Punkte können basierend auf den Betrag oder einzelner Komponenten dieses Differenzvektors segmentiert werden. Der Vorteil gegenüber dem *Euclidean Cluster Extraction* Algorithmus liegt darin, dass die durch die Punkte abgebildete Oberfläche durch den Normalenvektor approximiert wird. Es erfolgt keine einfache Unterteilung anhand von Punkteigenschaften. Allerdings muss für jeden Punkt eine zweifache Normalenbestimmung erfolgen, wobei die Betrachtung einer Vielzahl von Punkten je nach Auflösung der Punktwolke und der verwendeten Radien dafür notwendig sein kann.

Ein weiterer Algorithmus, der für die Segmentierung die Oberfläche durch Normalen approximiert, ist der *Region Growing* Algorithmus [29]. Dieser Algorithmus hat das Ziel, lokal verbundene Segmente mit gleichmäßigen Oberflächen zu finden. Dafür sind benachbarte Punkte von immenser Bedeutung, die bestenfalls sehr ähnliche Normalenvektoren besitzen sollten. Die Bestimmung der nächsten Nachbarn kann entweder mit einer festen Anzahl an Nachbarn oder einem festgelegten Radius durchgeführt werden. Für den Algorithmus müssen zwei Schwellwerte festgelegt werden. Zum einen gibt es einen Grenzwert für die maximale Krümmung und zum anderen einen Grenzwert für die maximale Abweichung des Schnittwinkels zwischen zwei Normalen. Als Maß für die Krümmung eines Punktes dient die Abweichung aller Punkte seiner Nachbarschaft zu einer geraden Fläche, welche orthogonal zu dessen Normalen verläuft. Der Algorithmus startet in dem Punkt mit minimaler Krümmung und betrachtet alle Punkte seiner Nachbarschaft. Wenn der Schnittwinkel der Normalen des Startpunktes und des ausgewählten Punktes unterhalb des Grenzwertes ist, dann gehört dieser Punkt zu der aktuellen Region. Wenn die Krümmung des betrachteten Punktes zusätzlich noch unterhalb des Grenzwertes für die maximale Krümmung liegt, wird dieser Punkt als potenzieller Startpunkt für die nächste Iteration markiert. Solang die Liste möglicher Startpunkte nicht leer ist, wird dieser Vorgang wiederholt und die Region wächst. Erst wenn die Liste keine weiteren Punkte mehr enthält, wird die Region in die Liste der Segmente aufgenommen und der Punkt mit der minimalen Krümmung als Startpunkt für eine neue Region gewählt, insofern noch nicht verarbeitete Punkte vorhanden sind. Die Liste der Segmente wird ausgegeben, sobald alle Punkte abgearbeitet sind. Die Schwellwerte für den Winkel und die Krümmung sind entscheidend für die resultierende Anzahl der Segmente. Wenn diese Grenzwerte sehr klein sind, dann werden viele kleine Segmente mit wenigen Punkten entstehen. Andersherum werden große Grenzwerte dazu führen, dass weniger Segmente mit sehr vielen Punkten entstehen werden. Diese entscheidet, ob eher eine Über- oder Untersegmentierung der Punktwolke auftreten wird.

Jedoch sind nicht alle Segmentierungsalgorithmen, welche zusammenfassend in Tabelle 2 aufgeführt sind, für die Prozesskette geeignet. Der *Min-Cut Based Segmentation* Algorithmus ist für die Prozesskette als Segmentiersalgorithmus nicht geeignet, da der Scan nur die

Tabelle 2: Zusammenfassung der Segmentierungsalgorithmen

	Eigenschaften
Min-Cut Based Segmentation	Radius und Startposition als Eingabe Aufbau eines Nächster-Nachbar-Graphen Aufteilung in Vorder- und Hintergrund anhand Hintergrundbedingung
Euclidean Cluster Extraction	Radius als Eingabe Segmentierung anhand des Punktabstandes
Difference of Normals Based Segmentation	Radien und Grenzwerte als Eingabe doppelte Normalenbestimmung notwendig Nachbarschaft durch Radien definiert verschiedene Vergleichsoptionen für Differenzverktor möglich
Region Growing	Raduis/Anzahl und Grenzwerte als Eingabe Normalbestimmung notwendig Nachbarschaft durch Radius/Anzahl definiert Approximation der Oberflächenkrümmung Segmentierung anhand der Normalenschnittwinkel und des Krümmungswertes

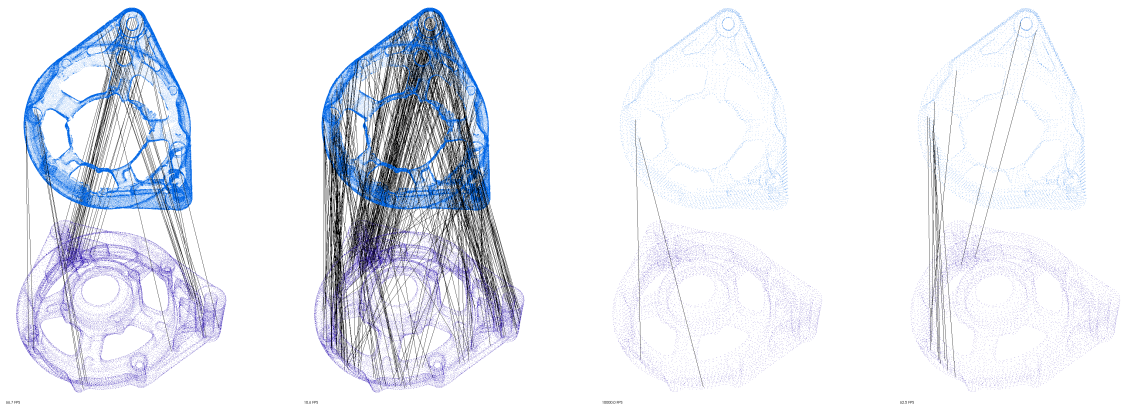
Baugruppe selbst beinhalten wird. Diese Baugruppe wird sich nicht in einer Szene befinden, aus der sie zunächst extrahiert werden soll. Die anderen Algorithmen können prinzipiell zur Segmentierung der Einzelteile eingesetzt werden.

3.5 Entwurf der Werkzeugkette

Basierend auf der Analyse der in den Vorarbeiten verwendeten Verfahren und weiterer Algorithmen zur Segmentierung soll nun eine konkrete Werkzeugkette entworfen werden. Diese soll die komplette Rekonstruktion einer Baugruppe aus einem dreidimensionalen Scan ermöglichen. Dabei können als Eingabe für die Prozessketten dreidimensionale Punktwolken von einzelnen Bauteilen oder einer gesamten Baugruppe dienen. Wird zunächst eine gesamte Baugruppe übergeben, muss als erster Schritt die Segmentierung erfolgen, um die separaten Bauteile zu extrahieren und anschließend zu identifizieren. Für die Segmentierung können drei der vorgestellten Segmentierungsalgorithmen prinzipiell verwendetet werden. Dabei fällt die Wahl für die Werkzeugkette auf den *Euclidean Cluster Extraction* und den *Region Growing* Algorithmus. Der *Euclidean Cluster Extraction* Algorithmus ist ein sehr einfacher Algorithmus und daher für eine erste Segmentierung empfehlenswert. Die abstandsbasierte Segmentierung ermöglicht eine Aufteilung einer Punktwolke in mehrere Segmente, wenn sich diese durch einen größeren Abstand voneinander auszeichnen. Für eine feinere Segmentierung soll anschließend der *Region Growing* Algorithmus dienen. Dabei erfolgt während der Segmentierung eine Suche einer angegebenen Anzahl von nächsten Nachbarn unabhän-

gig von deren Distanz. Die Betrachtung von Normalen und Krümmung sollte eine feine Segmentierung ermöglichen. Des Weiteren soll eine manuelle Korrektur möglich sein. Das bedeutet, dass stark fehlerbehaftete Segmentierungen rückgängig gemacht und mit anderen Parametern wiederholt werden können. Bei beiden Algorithmen sollen die entstehenden Segmente auch in der Größe limitiert werden können. Dabei ist vor allem eine untere Schranke wichtig, um so Ausreißer zu entfernen. Zusätzlich soll die Möglichkeit bestehen, dass mehrere Punktwolken miteinander vereint werden können. So ist eine vom Benutzer gestützte Segmentierung möglich und die entstandenen Segmente werden als Eingabe für die Identifikation genutzt.

Da sowohl für die Identifikation als auch für die Registrierung der Bauteile das lokale *SHOT-Feature* zufriedenstellende Resultate liefert, soll dieses auch für die Werkzeugkette genutzt werden. Durch die gute Reproduzierbarkeit und Robustheit dieses Verfahrens sind vergleichbare Merkmale möglich. Die *Keypoints* für die Bestimmung der *Features* sollen weiterhin durch das Rastern mithilfe eines Voxelgitters bestimmt werden, da dieser Ansatz schnell und einfach ist. Die *Keypoints*, die dann schwache *Features* und später schlechte Korrespondenzen liefern, sind durch die folgende Bewertung der Korrespondenzen nicht problematisch. In Abschnitt 3.3 wurde bereits angedeutet, dass die Teillösungen deutlich unterschiedliche Korrespondenzen produzieren. Da die Verfahren allerdings auch verschiedene Vorverarbeitungsschritte wie das *Downsampling* und die unterschiedlichen Parameterbestimmungen durchführen, muss für eine bessere Bewertung eine vergleichbare Ausgangssituation für die Qualität der Korrespondenzen geschaffen werden. Dafür wird die Korrespondenzbewertung innerhalb der Verfahren zur Identifikation nach [6] und zur Registrierung nach [7] ausgetauscht, um einen zuverlässigen Rückschluss auf die Qualität der Korrespondenzen zu ermöglichen und mit den Korrespondenzen aus Abbildung 12 zu vergleichen.



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

Abbildung 15: Weiterführender Vergleich der Korrespondenzen

Die Abbildungen 15a und 15b zeigen die Vorverarbeitung, wie sie bei der Identifikation vor-

gestellt wurde, mit den ausgewählten Korrespondenzen, wie sie durch die Gewichtung und Auswertung der Registrierung erzeugt werden. Bei den Abbildungen 15c und 15d ist genau das gegenteilige Prozedere zu sehen. Die jeweils erste Abbildung für beide Vorgehensweisen wurde mit den Standardwerten durchgeführt. Aufgrund der wenigen Korrespondenzen in Abbildung 15c wurde der Vorgang mit weniger stark limitierenden Parametern wiederholt und das Ergebnis dieser Durchläufe ist jeweils in Abbildung 15b und 15d zu sehen. Dabei wurden die maximale Distanz der *Features* einer Korrespondenz, die für das *Geometric Consistency Clustering* Verfahren genutzt werden dürfen, von $0,2$ auf $0,35$ angehoben und den prozentualen Anteil zu verwerfender Korrespondenzen des zweiten Ansatzes von 95% auf 70% reduziert. So wird die Anzahl der resultierenden Korrespondenzen gesteigert. In

Tabelle 3: Zusammenfassung der durchgeführten Korrespondenzbestimmungen

Abbildung	**Vorbereitung nach**	**Korrespondenz-bestimmung**	**Parameter**	**Anzahl**	**gemittelte Distanz**
12a	Identifikation	*Geometric Consistency Clusterung*	$d_f = 0,2$	30	$0,171679$
15a	Identifikation	gauß'sche Gewichtung	$p = 0,9$	48	$0,173488$
15b	Identifikation	gauß'sche Gewichtung	$p = 0,7$	287	$0,28272$
12b	Registrierung	gauß'sche Gewichtung	$p = 0,9$	31	$0,326945$
15c	Registrierung	*Geometric Consistency Clusterung*	$d_f = 0,2$	2	$0,178679$
15d	Registrierung	*Geometric Consistency Clusterung*	$d_f = 0,35$	13	$0,283385$

Tabelle 3 sind die Eigenschaften der erhaltenen Korrespondenzen zusammengefasst. Zudem unterstreicht Abbildung 15 den Eindruck von Abbildung 12. Auch wenn sich die gemittelte Distanz durch den Austausch der Korrespondenzbewertung nicht verschlechterte, so liegen die Korrespondenzen, die mit dem *Geometric Consistency Clustering* Verfahren heraus gefiltert wurden, wesentlich besser zueinander. Lediglich die Anzahl der resultierenden Korrespondenzen wird durch die Anwendung des Verfahrens reduziert. Dies ist damit zu begründen, dass nicht allein der Abstand das ausschlaggebende Kriterium zur Korrespondenzbewertung ist, sondern auch die Lage zwischen zwei Korrespondenzen betrachtet wird. Selbst die Anwendung des Algorithmus auf das stark reduzierte Modell in Abbildung 15d zeigt einige gut zueinander liegende Korrespondenzen. Daher empfiehlt sich das Gruppieren

der Korrespondenzen mit diesem Verfahren innerhalb der Werkzeugkette. Dieser Prozess bestehend aus Korrespondenzbildung und -bewertung muss für ein Segment mit allen vorhandenen Referenzbauteilen durchgeführt werden. Dabei wird die Anzahl der erhaltenen Korrespondenzen als Maß für die Ähnlichkeit zweier Bauteile genutzt. Wie bei der Segmentierung muss auch dieser Teilprozess im Falle eines Fehlers korrigiert werden können. Die Korrektur soll auch hier durch einen erneuten Durchlauf mit veränderten Parametern oder durch eine manuelle Auswahl des Referenzteils erfolgen.

Ist die Identifikation eines Bauteils erfolgt, können die bestimmten Korrespondenzen direkt für die Ausrichtung mithilfe der vorgestellten Algorithmen erfolgen. Für die korrekte Positionierung der gefundenen Referenzteile wird ebenfalls eine zweistufige Ausrichtung mit dem *Sample Consensus Initial Alignment* Verfahren und dem *Iterative Closest Point* Verfahren durchgeführt. Zur Bewertung der Lage des Referenzteils zum segmentierten Bauteil soll ein anderer Ansatz genutzt werden. In [7] wurde der Abstand von verschiedenen Korrespondenzen zur Feststellung der korrekten Lage ins Verhältnis gesetzt. Jedoch wird darauf hingewiesen, dass fehlerhafte Korrespondenzen diesen Wert verzerren können. Somit kann eine gute Positionierung nicht als solche identifiziert werden. Daher sollen das Referenzteil und das Bauteil auf alle Koordinatenebenen abgebildet und zur Bewertung der Lage zueinander verglichen werden. Dabei wird die prozentuale Überlagerung der Teile für jede Ebene bestimmt und das Mittel aus allen Ansichten als Maß für die Ausrichtung genutzt. Während in [7] lediglich eine manuelle Rotation zur Korrektur der Lage zur Verfügung stand, soll nun auch eine Verschiebung des Bauteils im Raum möglich sein. Somit kann eine fehlerhafte Ausrichtung, die sich durch erneute Ausführung der vorgestellten Verfahren nicht weiter verbessern lässt, vollständig manuell korrigiert werden.

Als letzter Schritt muss das Ergebnis der Werkzeugkette exportiert werden. Dafür sollen alle erzeugten Segmente, alle identifizierten Referenzteile sowie alle gewonnenen Informationen ausgegeben werden. Die Modelle aller Bauteile werden als Punktwolken im angegebenen Speicherort abgelegt. Die Informationen, wie die Zugehörigkeit von Segment und Referenz oder die ermittelte Transformationsmatrix, werden in Form einer XML-Datei abgelegt.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<bom>
    <assembly name="Baugruppe">
        <pcpath>C:\...\Baugruppe.ply</pcpath>
        <part id="0">
            <name>Name Referenz</name>
            <pcpath>C:\...\Segment.ply</pcpath>
            <refpath>C:\...\Referenz.ply</refpath>
            <transformMatrix type_id="opencv-matrix">
                <rows>4</rows>
                <cols>4</cols>
                <data>1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1</data>
                <dt>float</dt>
            </transformMatrix>
        </part>
    </assembly>
</bom>
```

Code 3: Aufbau der XML-Struktur

Code 3 zeigt die genutzte Struktur für den Export der Informationen. Innerhalb der XML-Datei wird die Eingabe mit Namen und den Pfaden aller geöffneten Modelle hinterlegt. Dann wird für jedes Teil, das innerhalb des Durchlaufs identifiziert wird, der Name, der Pfad zum Segment, der Pfad zu dessen Referenzteils und die Transformationsmatrix gespeichert. Diese Exportmöglichkeit soll für eventuell spätere Prozesse mit externen Werkzeugen als Eingabe dienen können. In Abbildung 16 ist das Flussdiagramm der entworfenen Werkzeugkette zu sehen. Dabei ist die Kette der Teilprozesse von Segmentierung, Identifikation und Registrierung und die dafür genutzten Verfahren klar zu erkennen.

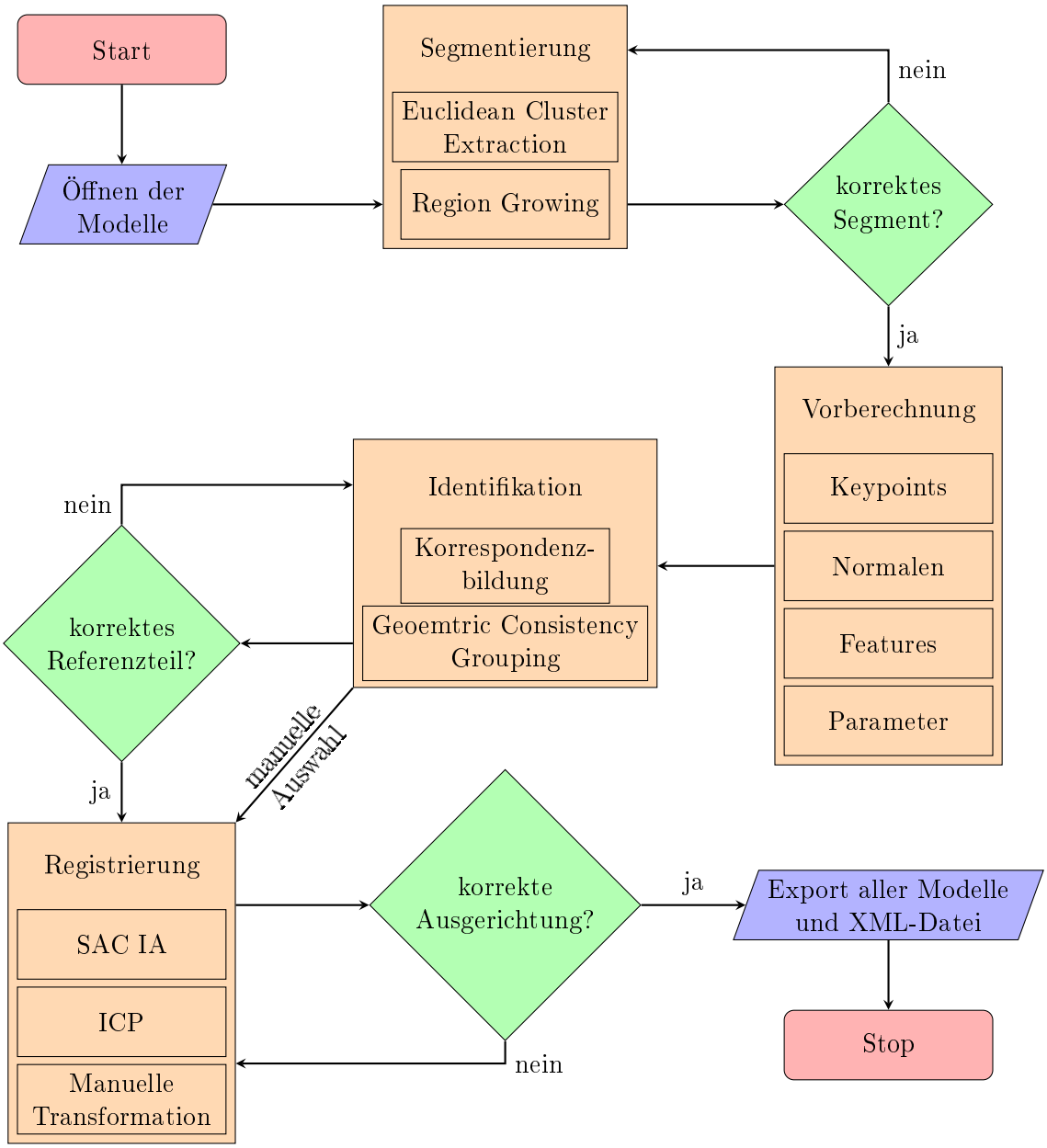


Abbildung 16: Flussdiagramm der entworfenen Werkzeugkette

4 Implementierung

Im folgenden Kapitel wird die Implementierung der Werkzeugkette, wie sie im vorherigen Kapitel entworfen wurde, beschrieben. Zunächst wird die genutzte Entwicklungsumgebung samt externer Bibliotheken und das verwendete Strukturmuster vorgestellt. Anschließend wird auf die wichtigsten Bestandteile der Implementierung der einzelnen Teilprozesse eingegangen. Zum Abschluss wird die Umsetzung des ausgewählten Strukturmusters sowie der Auslagerung von Prozessen für die Anwendung erläutert.

4.1 Entwicklungsumgebung

Für die Entwicklungsumgebung der Werkzeugkette bestehen die selben Kriterien wie für die vorhandenen Vorarbeiten. Zu den Kriterien gehören eine Ein- und Ausgabeschnittstelle für dreidimensionale Daten im PLY- oder STL-Datenformat, verschiedene Funktionalitäten für die Verarbeitung der Modelle innerhalb der Teilprozesse und eine Visualisierungsschnittstelle zur Darstellung der Modelle für den Benutzer. Nach wie vor ist die *Point Cloud Library* (*PCL*) [30] eine sehr geeignete Bibliothek für die Verarbeitung von dreidimensionalen Daten, weshalb diese auch für die Realisierung der vorhandenen Teillösungen [6, 7] genutzt wurde. Diese Bibliothek bietet mit seinen Modulen verschiedenste Verfahren für die Segmentierung, Identifizierung, Ausrichtung und für weitere Problemstellungen. Die wichtigste Eigenschaft der Bibliothek ist, dass diese der BSD-Lizenz unterliegt und somit frei verwendet werden kann. Des Weiteren unterstützen viele namhafte Organisationen, Unternehmen und Universitäten die Entwicklung dieser Bibliothek. Für die Verwendung von *PCL* in der Version 1.7 muss die Bibliothek selbstständig kompiliert werden und die vorherige Installation weiterer externer Bibliotheken ist notwendig. Dazu gehören die Bibliotheken *Boost* (1.50), *Eigen* (3.0.5), *FLANN* (1.7.1), *VTK* (5.8) und *Qt* (4.8). *Boost* dient der effizienten Speicherung von Objekten. Mithilfe sogenannter *Shared Pointers* werden die Daten zwischen Modulen und Algorithmen übergeben. *Eigen* ist eine freie Bibliothek für Lineare Algebra und die mathematischen Berechnungen von *PCL* basieren auf dieser Grundlage. *FLANN* (*Fast Library for Approximate Nearest Neighbors*) stellt einen schnellen und effizienten Ansatz für Identifizierung von nächsten Nachbarn bereit, welcher in vielen Verfahren und Algorithmen verwendet wird. *VTK* (*Visualization ToolKit*) dient als Grundlage für die Visualisierung von *PCL*. Diese Bibliothek ermöglicht die Darstellung von dreidimensionalen Punktwolken oder Netze, Texturen und weiterer Visualisierungen. Für die Entwicklung der graphischen Benutzeroberfläche wird Bibliothek *Qt* genutzt. Die Gesamtheit der vorgestellten Bibliotheken ermöglicht die Umsetzung der Werkzeugkette innerhalb einer Anwendung mit graphischer Benutzeroberfläche.

4.2 Strukturmuster

Für die Entwicklung einer Anwendung mit einer graphischen Benutzeroberfläche muss ein geeignetes Muster genutzt werden. Das genutzte Muster sollte eine klare Trennung der Programmierung der oberflächlichen Darstellung und der darunterliegenden Programmlogik ermöglichen. So sollte eine Modifikation der Oberfläche ohne Kenntnis der damit verbundenen Logik möglich sein und andersherum. Klassischerweise wurde dafür das *Model-View-Controller* (*MVC*) Muster entwickelt. Dabei wurde die Programmlogik im *Model* und die Benutzeroberflächen in den *Views* bereitgestellt. Der *Controller* verarbeitet die Anfragen von Benutzern unter Verwendung des *Models*, gibt die Resultate an die *Views* zurück und koordiniert somit alle Prozesse der Anwendung. Das *Model-View-ViewModel* (*MVVM*) Muster [31] ist dafür ebenfalls sehr geeignet.

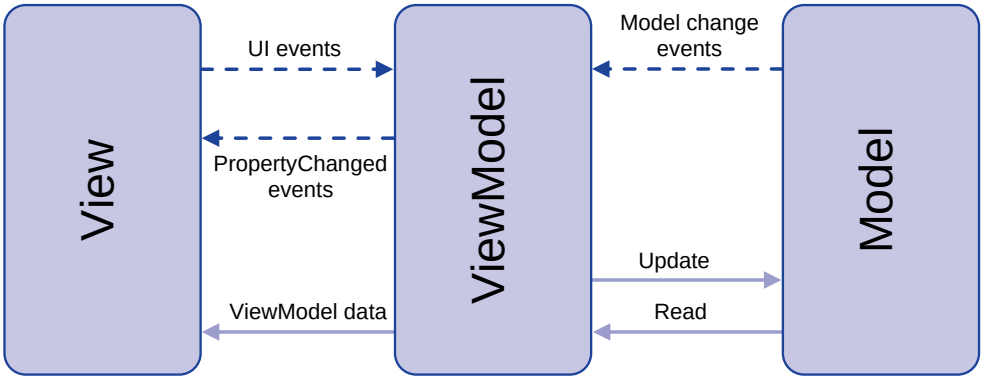


Abbildung 17: Darstellung des *MVVM* Musters [31]

Es ist eine Variante des *Model-View-Controller* Musters und unterteilt ebenfalls die Darstellung in *Views* und die Programmlogik im *Model*, wobei das *ViewModel* als Bindeglied zwischen diesen Komponenten dient. Anders als bei dem *Controller* leitet das *ViewModel* entsprechende Prozesse ein und verarbeitet sie nicht selbst. Abbildung 17 zeigt die Bestandteile dieses Konzeptes und deren Zusammenhänge. Die konkreten Funktionen der einzelnen Teile sind die folgenden:

- *Model*:
  Es beinhaltet die darstellungsunabhängige Repräsentation der Daten. Daher richtet sich diese Repräsentation ausschließlich nach der Programmlogik.
- *View*:
  Sie beinhalten die Benutzeroberflächen und werden für die Darstellung von Informationen für den Benutzer genutzt. Zusätzlich geben diese die Aktionen des Benutzers ohne Wissen über deren Verarbeitung weiter.

- *ViewModel*:
  Es dient als Brücke zwischen den vorherigen Komponenten. Es empfängt Benutzeraktionen von den *Views* und leitet die entsprechenden Prozesse ein, die für die Verarbeitung dieser Eingaben ausgeführt werden müssen. Auch wird hier die Manipulation der Daten des *Models* eingeleitet und anschließend aktualisiert. Alle Resultate, die eine Modifikation der Darstellung mit sich ziehen, werden hier ohne Kenntnisse der notwendigen Veränderungen an die *Views* weitergereicht.

Qt [32] eignet sich hervorragend für die Umsetzung des *MVVM* Musters. Diese Bibliothek stellt sogenannte *Signals* und *Slots* als Funktionen für die Kommunikation bereit. Somit können die verschiedenen Bestandteile des Musters implementiert und Ereignisse oder Änderungen mittels dieser Funktionen kommuniziert werden.

4.3 Implementierung der Werkzeugkette

Um eine Baugruppe verarbeiten zu können, werden die entworfenen Teilprozesse unter der Verwendung von *PCL* umgesetzt. So kann der Scan einer Baugruppe importiert und segmentiert werden. Nach der Identifizierung ihrer Einzelteile erfolgt die Ausrichtung der Referenzteile an den Segmenten. Abschließend kann das Ergebnis exportiert werden.

4.3.1 Segmentierung

Für die Segmentierung ist keine große Vorverarbeitung notwendig. Vielmehr bildet die Segmentierung die Vorverarbeitung für die darauffolgenden Teilprozesse. Durch die Benutzeroberfläche können die Parameter für die ausgewählten Verfahren übergeben und das zu segmentierende Objekt ausgewählt werden. Sowohl der *Euclidean Cluster Extraction* als auch der *Region Growing* Algorithmus sind innerhalb von *PCL* implementiert. Der erste Algorithmus bedarf für seine Ausführung eine Punktwolke, eine minimale Clustergröße, eine maximale Clustergröße und eine Toleranz zur Clusterbildung. Das Ergebnis des Algorithmus ist eine Liste von Menge, die die Indizes der Punkte eines jeweiligen Clusters enthalten. Durch das Iterieren über diesen Indizes und dem Kopieren der Punkte kann die übergebene Punktwolke aufgeteilt und eine Liste von Segmenten zurückgeben werden. Code 4 zeigt die beschriebene Segmentierung mithilfe des *Euclidean Cluster Extraction* Verfahrens.

```
1    // create kd tree representation of cloud
2    pcl::search::KdTree<Point>::Ptr kdTree (new pcl::search::KdTree<Point>);
3    kdTree->setInputCloud(cloud);
4
5    // euclidean cluster extraction with given attributes
6    vector<pcl::PointIndices> clusterIndices;
7    pcl::EuclideanClusterExtraction<pcl::PointXYZ> cluster;
```

```
8     cluster.setClusterTolerance(clusTol);
9     cluster.setMinClusterSize(minSize);
10    cluster.setMaxClusterSize(maxSize);
11    cluster.setSearchMethod(kdTree);
12    cluster.setInputCloud(cloud);
13    cluster.extract(clusterIndices);
14
15    // iterate clusters to construct point cloud
16    for(vector<pcl::PointIndices>::const_iterator cIter = clusterIndices.
          begin(); cIter != clusterIndices.end(); ++cIter) {
17        PointCloudPtr cloudCluster (new PointCloud());
18        // iterate points of cluster
19        for (vector<int>::const_iterator pIter = cIter->indices.begin();
              pIter != cIter->indices.end(); ++pIter)
20            cloudCluster->push_back(cloud->points[*pIter]);
21
22        cloudCluster->width = cloudCluster->points.size();
23        cloudCluster->height = 1;
24        cloudCluster->is_dense = true;
25
26        segmentedClouds.push_back(cloudCluster);
27    }
```

Code 4: *Euclidean Cluster Extraction* Algorithmus

Neben der oberen und unteren Schranke für die Größe eines Cluters benötigt der zweite Segmentierungsalgorithmus noch eine Anzahl der nächsten Nachbarn. Diese ist sowohl für die Bestimmung der Normalen zur Approximation der Oberfläche, als auch für das spätere Wachstum des Clusters notwendig. Für die Bestimmung der Normalen werden die nächsten Nachbarn in der angegebenen Anzahl gesucht und durch die Hauptkomponentenanalyse ein Normalenvektor ermittelt. Zusätzlich werden die zwei Grenzwerte übergeben, die das Wachstum beschränken. Code 5 zeigt die Bestimmung der Normalen und die Ausführung des *Region Growing* Verfahrens. Auch hier wird eine Liste von Indexmengen zurückgeben, welche mit dem selben Ansatz in separate Punktwolken umgewandelt werden.

```
1     // create kd tree representation of cloud
2     pcl::search::KdTree<Point>::Ptr kdTree (new pcl::search::KdTree<Point>);
3     NormalsPtr normals (new Normals());
4
5     // define normals
6     pcl::NormalEstimationOMP<Point, Normal> nor;
7     nor.setNumberOfThreads(3);
8     //pcl::NormalEstimation<Point, Normal> nor;
9     nor.setSearchMethod(kdTree);
10    nor.setInputCloud(cloud);
11    nor.setKSearch(numberNeighbours);
12    nor.compute(*normals);
13
14    // define segmentation attributes
15    pcl::RegionGrowing<Point, Normal> reg;
16    reg.setMinClusterSize(minSize);
17    reg.setMaxClusterSize(maxSize);
```

```
18      reg.setSearchMethod(kdTree);
19      reg.setNumberOfNeighbours(numberNeighbours);
20      reg.setInputCloud(cloud);
21      reg.setInputNormals(normals);
22      // angle in rad (for range of normals deviation)
23      reg.setSmoothnessThreshold(angle / 180.0 * M_PI);
24      // small normals deviation -> disparity of curvatures tested
25      reg.setCurvatureThreshold(curvature);
```

Code 5: *Region Growing* Algorithmus

Außerdem wurde ein Segmentierungsverlauf implementiert, um getätigte Segmentierung rückgängig zu machen oder um rückgängig gemachte Segmentierung ohne erneuten Rechenaufwand wiederherzustellen. Dabei handelt es sich um eine Liste von Listen, welche die vorhandenen Segmente jener Durchführung enthalten. Sollten mehrere Segmentierungsvorgänge rückgängig gemacht und anschließend eine neue Segmentierung gestartet werden, so werden alle voraus liegenden Schritte verworfen und die neu entstandene Segmentenliste hinzugefügt.

4.3.2 Identifikation

Zunächst müssen für das Identifizieren der Bauteile die *Keypoints* und *Features* bestimmt werden. Im Speziellen ist für die Berechnung des ausgewählten *SHOT-Features* die Bestimmung der Normalen notwendig. Die Vorverarbeitung für die Identifikation setzt sich demnach aus der Bestimmung der *Keypoints*, der Normalen für alle Punkte, der *Features* für die ausgewählten *Keypoints* und der Berechnung der Parameter für die spätere Identifizierung zusammen.

```
1       PointCloudPtr cloud = qo.getCloud();
2       PointCloudPtr keypoints (new PointCloud());
3       NormalsPtr normals (new Normals());
4       SHOTFeaturesPtr shotFeatures (new SHOTFeatures);
5
6       // sample keypoints
7       doUniformSampling(cloud, 4.0f, keypoints);
8       qo.setKeypoints(keypoints);
9
10      // determine normals
11      estimateNormals(cloud, 10, normals);
12      qo.setNormals(normals);
13
14      // estimate identification parameters
15      double resolution = computeResolution(cloud);
16      double smallestBBSide = computeSmallestBBSide(cloud);
17      double descriptorRadius = 0.5f * smallestBBSide;
18      qo.setGlobalDistance(resolution);
19      qo.setDescriptorRadius(descriptorRadius);
20      qo.setGCSize(5.0f * resolution);
21      qo.setGCThreshold(keypoints->points.size () / 400.0f);
```

```
22
23      // get SHOT features
24      computeSHOTFeatures(cloud, keypoints, normals, descriptorRadius,
            shotFeatures);
25      qo.setSHOTFeatures(shotFeatures);
```

Code 6: Vorverarbeitung für Identifikation

Code 6 zeigt diese Vorverarbeitung, wobei für die einzelnen Operationen die bereits vorgestellten Verfahren und deren in *PCL* bereitgestellten Implementierungen genutzt werden. Alle Parameter werden gemäß der von Politz [6] entwickelten Konvention gewählt oder berechnet. Die Auflösung einer Punktwolke wird durch den Abstand eines jeden Punktes zu dessen nächsten Nachbarn bestimmt. Es werden allen Abstände aufsummiert und durch die Anzahl der Punkte geteilt und das Fünffache des Wertes als Auflösung innerhalb eines Clusters genutzt. Die minimale Größe eines Clusters wird als vier Hundertstel der Anzahl der *Keypoints* gewählt. Die schmalste Seite eines Bauteils wird anhand des orientierten Hüllkörpers, auch *Bounding Box* genannt, bestimmt. Durch den Schwerpunkt der Punktwolken und eines lokalen Koordinatensystems anhand der Eigenvektoren kann ein Quader um das Objekt platziert werden. Dessen Dimensionen werden durch den minimalen und maximalen Punkt innerhalb der Punktwolke beschränkt. Die Hälfte dieser kürzesten Seite dient als Radius für die Berechnung der *Features*. Für die Identifizierung müssen sowohl das erzeugte Segment als auch das geladene Referenzteil entsprechend vorbereitet werden. Der entwickelte Suchprototyp von Politz [6] wurde so implementiert, dass die Referenzteile innerhalb des Scans einer gesamten Baugruppe detektiert werden sollte. Durch die nun vorhergehenden Segmentierung wird eine Baugruppe in seine Bestandteile aufgeteilt und die Identifizierung innerhalb der gesamten Baugruppe ist nicht länger vorgesehen. Des Weiteren war diese Vorgehensweise mit einem Mehraufwand verbunden. Da die einzelnen Referenzteile unterschiedliche Größen besitzen und der Radius des *SHOT-Features* anhand dieser Größe berechnet wurde, war für jedes Referenzteil eine erneute Bestimmung des *SHOT-Features* für die *Keypoints* der Baugruppe notwendig, um eine Vergleichbarkeit der *Features* zu gewähren. Somit erfolgt die Identifizierung nun anhand des Vergleiches der *SHOT-Features* von Segment und Referenzteil, die entsprechend ihrer Größen erzeugt werden. Eine erneute Berechnung ist für keinerlei Bauteil, unabhängig ob es sich um ein Segment oder Referenzteil handelt, nötig. Mittels der Nächsten-Nachbarn-Suche innerhalb eines k-d-Baums werden die Korrespondenzen gebildet, die dann durch den vorgestellten Algorithmus geclustert werden. Code 7 zeigt den wesentlichen Inhalt des Vergleiches zweier Bauteile, der der tatsächlichen Identifikation dient.

```
1       // search correspondences
2       pcl::KdTreeFLANN<SHOT> correspondencesSearchTree;
3       correspondencesSearchTree.setInputCloud(referencesFeatures);
4
```

```
5     for (size_t i = 0; i < segmentsFeatures->size (); ++i) {
6         int foundNeighs = correspondencesSearchTree.nearestKSearch (
              segmentsFeatures->at(i), 1, neighIndices, neighSqrDists);
7         if(foundNeighs == 1 && neighSqrDists[0] < distance) {
8             pcl::Correspondence corr (neighIndices[0], static_cast<int> (i),
                  neighSqrDists[0]);
9             correspondences->push_back (corr);
10        }
11    }
12    // Start the clustering process with geometric consistency
13    pcl::GeometricConsistencyGrouping<Point, Point> clusterer;
14    clusterer.setGCSize(reference.getGCSize());
15    clusterer.setGCThreshold(reference.getGCThreshold());
16    clusterer.setInputCloud(reference.getKeypoints());
17    clusterer.setSceneCloud(segment.getKeypoints());
18    clusterer.setModelSceneCorrespondences(correspondences);
19    clusterer.recognize(instances, clusteredCorrs);
20
21    // Find the correspondence set with the most votes
22    int maxVotes = 0;
23    int index = -1;
24    for(int i = 0; i < instances.size(); ++i){
25        if(clusteredCorrs[i].size() > maxVotes) {
26            maxVotes = clusteredCorrs[i].size();
27            index = i;
28        }
29    }
30    if(maxVotes==0)
31        result = std::make_tuple(maxVotes, *correspondences, TransMatrix::
              Identity());
32    else
33        result = std::make_tuple(maxVotes, *correspondences, instances.at(
              index));
```

Code 7: Vergleich von Segment und Referenzteil zur Identifikation

Ein weiterer Unterschied liegt in dem maximalen Abstand innerhalb einer Korrespondenz, der als zulässig betrachtet wird. Durch die Normalisierung bei der Bestimmung der *SHOT-Features* liegt der Abstand zweier *Features* einer Korrespondenz im Intervall $[0, 1]$. In der Implementierung des Suchprototyps von Politz [6] wurden nur Korrespondenzen mit einem maximalen Abstand von $0,2$ zugelassen. Dieser Grenzwert soll nun vom Benutzer definiert werden, um bei einem nicht erfolgreichen Durchlauf mithilfe eines schwächeren Grenzwertes noch ein Ergebnis erzielen zu können. Falls dennoch kein passendes Referenzteil identifiziert werden kann, besteht die Möglichkeit einer manuellen Auswahl des gewünschten Referenzteils.

4.3.3 Registrierung

Der bei der Identifizierung genutzte *Geometric Consistency Grouping* Algorithmus produziert neben der Menge von Korrespondenzenclusters auch noch eine Menge von Transforma-

tionsmatrizen. Für jedes Cluster wird eine Transformationsmatrix hergeleitet, die es erlaubt die verbundenen Punkte aller Korrespondenzen bestmöglich übereinander zu legen. Dabei bestimmt die Qualität der im Cluster vorhandenen Korrespondenzen das Resultat dieser Transformation. Aus diesem Grund wird diese Transformationsmatrix zwar genutzt, um das Segment und das Referenzteil einander anzunähern, aber es werden trotzdem die vorgestellten Verfahren zur weiteren groben und feineren Ausrichtung angewandt. Code 8 zeigt, dass für die Anwendung des *SAC-IA*-Verfahrens die aneinander auszurichtenden Punktwolken auf die in den Korrespondenzen enthaltene Punkte reduziert werden müssen, da die Implementierung eine gleiche Anzahl von *Features* erfordert.

```
1     // reduce clouds
2     for(int i = 0; i < correspondences->size() ; i++){
3         corresRefCloud->points.push_back(Point(refKeypoints->points[
              correspondences->at(i).index_query]));
4         corresRefFeatures->points.push_back(refFeatures->points[
              correspondences->at(i).index_query]);
5
6         corresSegCloud->points.push_back(Point(segKeypoints->points[
              correspondences->at(i).index_match]));
7         corresSegFeatures->points.push_back(segFeatures->points[
              correspondences->at(i).index_match]);
8     }
9     //Perform initial alingment on keypoints
10    PointCloud alignedCloud;
11    pcl::SampleConsensusInitialAlignment<Point, Point, SHOT> aligner;
12    aligner.setMinSampleDistance(0.5f);
13    aligner.setInputCloud(corresRefCloud);
14    aligner.setSourceFeatures(corresRefFeatures);
15    aligner.setInputTarget(corresSegCloud);
16    aligner.setTargetFeatures(corresSegFeatures);
17    aligner.align(alignedCloud);
18    return aligner.getFinalTransformation();
```

Code 8: *SAC-IA*-Verfahren

Der Implementierung des *ICP*-Verfahrens muss dagegen lediglich beide Punktwolken übergeben werden. Anschließend wird versucht die Ausrichtung des Referenzteils zum Segment zu verbessern. Code 9 zeigt die nötigen Befehle für die Anwendung des Verfahrens und den Erhalt der resultierenden Transformationsmatrix.

```
1     pcl::IterativeClosestPoint<Point, Point> aligner;
2     aligner.setInputCloud(refKeypoints);
3     aligner.setInputTarget(segKeypoints);
4     aligner.align(alignedCloud);
5     return aligner.getFinalTransformation();
```

Code 9: *ICP*-Verfahren

Das Aktualisieren der Lage der Punktwolken und der Transformationsmatrix eines ausgerichteten Referenzteils nach einer durchgeführten Transformationen ist sehr einfach. Durch

die genutzte Matrizenform, wie in Gleichung 7 dargestellt, kann die gespeicherte Transformationsmatrix mit einer weiteren Matrix durch einfache mathematische Operationen zur Aktualisierung verrechnet werden. Innerhalb der Matrix sind Rotation und Translation zu finden. Die Rotation wird durch die obere linke 3×3-Matrix und die Translation durch die obere rechte 3×1-Matrix definiert. Die letzte Zeile muss mit den gezeigten Werten belegt sein.

$$T = \begin{pmatrix} r_{11} & r_{12} & r_{31} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

Zwei Transformationsmatrizen in dieser Form können miteinander multipliziert werden und die resultierende Matrix beinhaltet die kombinierten Transformationen. Allerdings ist diese Multiplikation nicht kommutativ. Wenn zuerst eine Transformationsmatrix T_1 und anschließend eine weitere Transformationsmatrix T_2 zur Änderung der Lage eines Objektes angewandt werden, dann beschreibt die Matrix $T = T_2 \cdot T_1$ dieselbe Transformation. Zusätzlich bleibt die Form aus Gleichung 7 bei der resultierenden Matrix erhalten.

Mertens [7] unterschied bei der Ausrichtung zwei verschiedene Modi. Es existierte ein automatischer und ein manueller Modus. Bei dem automatischen Modus wurde einmalig das *SAC-IA*-Verfahren zur groben Ausrichtung und darauffolgend mehrmalig der *ICP*-Algorithmus zur feinen Ausrichtung angewandt. Dieser wurde solang wiederholt, bis eine fixe Anzahl von Iterationen erreicht oder die Ausrichtung für gut befunden wurde. Wenn dieser Modus nicht erfolgreich war oder von Anfang an der manuelle Modus ausgewählt wurde, konnte das auszurichtende Teil vom Benutzer rotiert und abschließend mit dem *ICP*-Algorithmus ausgerichtet und gespeichert werden. Zwar ist diese Unterteilung in automatischer und manueller Ausrichtung durchaus sinnvoll, dennoch sollte diese nicht explizit durch verschiedene Modi getrennt werden. Für die Ausrichtung innerhalb der Werkzeugkette besteht die automatische Ausrichtung ebenfalls aus der Anwendung von *SAC-IA*- und *ICP*-Verfahrens. Die manuelle Ausrichtung kann jederzeit vor und nach einer automatischen Ausrichtung genutzt werden. Auch ist der Verzicht auf eine automatische Ausrichtung möglich, um nur manuell durch den Benutzer zu erfolgen. Das Verfahren zur Bewertung der Ausrichtung wurde entsprechend des Entwurfs ersetzt. Dafür werden zunächst das Segment und das Referenzteil temporär verschoben. Das Referenzteil wird so verschoben, dass dessen orientierte *Bounding Box* parallel und mit leichtem Versatz zu den Koordinatenachsen im ersten Oktant liegt. Diese Transformationsmatrix wird ebenfalls zum Transformieren des Segments genutzt. Somit bleibt die Lage der beiden Objekte zueinander erhalten und können auf die positiven Ebenen des Koordinatensystems abgebildet werden. Die Umsetzung der Ausrichtungsbewertung ist in Auszügen in Code 10 zu sehen. Für die Abbildung wird jeweils eine zweidimensionale Matrix je Ebene aufgebaut. Da das Segment als Teil eines

Scans und das Referenzteil nie exakt die gleichen Punkte besitzen können, werden die Koordinaten der Punkte gerundet. Es werden alle Punkte des Referenzteils in den Matrizen markiert. Fallen mehrere Punkte des Modells auf einen Punkt in der Ebene, so werden diese nicht zusätzlich gezählt oder ähnliches. Abschließend wird über alle Punkte des Segments iteriert und geprüft, ob es eine Übereinstimmung mit dem Referenzteil in den Matrizen gibt. Die Anzahl der gemeinsamen Punkte wird mit der Anzahl aller Punkte des Segments normalisiert und der Durchschnitt aus allen Ebenen gebildet. Dieser Wert wird als Maß der Ausrichtung genutzt und ab einem Wert von $0,9$, der einer 90 prozentigen Übereinstimmung entspricht, wird die Ausrichtung als ausreichend gut bewertet.

```
1     // check aligment for each plane
2     int counterXY = 0;
3     int counterXZ = 0;
4     int counterYZ = 0;
5     int m = (int) (1.1f * box.width() + 0.5f);
6     int n = (int) (1.1f * box.height() + 0.5f);
7     int o = (int) (1.1f * box.depth() + 0.5f);
8     Eigen::MatrixXi xy = Eigen::MatrixXi::Zero(m, n);
9     Eigen::MatrixXi xz = Eigen::MatrixXi::Zero(m, o);
10    Eigen::MatrixXi yz = Eigen::MatrixXi::Zero(n, o);
11
12    // set reference's points in matrix
13    for(int i=0; i<alignedReference->points.size(); ++i) {
14        int pointX = (int) (alignedReference->points.at(i).x + 0.5f);
15        ...
16        xy(pointX, pointY) = 1;
17        ...
18    }
19    for(int i=0; i<alignedSegment->points.size(); ++i) {
20        int pointX = (int) (alignedSegment->points.at(i).x + 0.5f);
21        ...
22        // check for outliners
23        if(pointX >= 0 && pointX < m && pointY >= 0 && pointY < n && xy(
              pointX, pointY))
24            ++counterXY;
25        ...
26    }
27    // estimate matching
28    double matchXY = (double) (counterXY) / (double) (alignedSegment->points.
          size());
29    ...
30    double percentage = (matchXY / 3.) + (matchXZ / 3.) + (matchYZ / 3.);
```

Code 10: Überprüfung der Ausrichtung

Die manuelle Ausrichtung sollte zu jedem Zeitpunkt durchgeführt werden können, um die Positionierung zu verbessern oder die korrekte Endposition zu erhalten, falls dies notwendig ist. Dafür soll das Referenzteil um alle Achsen des Koordinatensystems rotiert, sowie in Richtung dieser verschoben werden können. Damit die Rotation um die Achsen an Ort und Stelle durchgeführt werden kann, besteht die Transformation nicht nur aus einer Rotation.

Diese beinhaltet zusätzlich eine vorübergehende Translation. Da die Objekte nicht zwangsläufig um den Koordinatenursprung liegen müssen, würde bei einer Rotation um die Achsen eine starke Verschiebung im Raum erfolgen, je weiter das Bauteil vom Ursprung entfernt ist. Daher setzt sich die Rotation des Bauteils an dessen Position aus der Verschiebung, so dass dessen Schwerpunkt im Ursprung liegt, der eigentlichen Rotation um die Koordinatenachse und der inversen Verschiebung, so dass der Schwerpunkt wieder an der Ausgangsposition liegt, zusammen. Für die Translation muss dagegen lediglich das Maß der Verschiebung für die entsprechende Richtung übergeben werden und die Transformationsmatrix kann sofort erzeugt werden. Code 11 zeigt, wie die Transformationsmatrizen für eine manuelle Rotation bzw. Translation erzeugt werden können. Beide Transformationen werden separat für die einzelnen Achsen durchgeführt.

```
1  // initialize matrix for rotation
2      if (axis == "x")
3          rotationMatrix.rotate (Eigen::AngleAxisf (rotation * (M_PI/180),
               Eigen::Vector3f::UnitX()));
4      else if (axis== "y")
5          rotationMatrix.rotate (Eigen::AngleAxisf (rotation * (M_PI/180),
               Eigen::Vector3f::UnitY()));
6      else if (axis== "z")
7          rotationMatrix.rotate (Eigen::AngleAxisf (rotation * (M_PI/180),
               Eigen::Vector3f::UnitZ()));
8
9      // transformation matrix for rotation around the object's centroid
10     Eigen::Vector4f centroid;
11     pcl::compute3DCentroid(*(*it).second.getCloud(), centroid);
12     translationMatrix.translation() << centroid.x(), centroid.y(), centroid.z
           ();
13     inverseTranslationMatrix.translation() << (-1)*centroid.x(), (-1)*
           centroid.y(),(-1)* centroid.z();
14
15     // update transformation matrix
16     TransMatrix matrix = (translationMatrix*rotationMatrix*
           inverseTranslationMatrix).matrix();
17     (*it).second = this->updateTransformationMatrix((*it).second, matrix);
18
19     ...
20
21     // initialize matrix for translation
22     Eigen::Affine3f translationMatrix = Eigen::Affine3f::Identity();
23     if (axis == "x")
24         translationMatrix.translation() << translation, 0, 0;
25     else if (axis == "y")
26         translationMatrix.translation() << 0, translation, 0;
27     else if (axis == "z")
28         translationMatrix.translation() << 0, 0, translation;
29
30     // update transformation matrix
31     TransMatrix matrix = translationMatrix.matrix();
32     (*it).second = this->updateTransformationMatrix((*it).second, matrix);
```

Code 11: Manuelle Transformation

4.4 Implementierung der Anwendung

Für die benutzerfreundliche Verwendung der Werkzeugkette muss diese in einer Anwendung mit graphischer Benutzeroberfläche eingebettet werden. Dabei sollen die bereitgestellten Werkzeuge selbsterklärend zu nutzen und sinnvoll miteinander verknüpft sein.

4.4.1 Umsetzung des *MVVM*-Musters

Da die Anwendung entsprechend des ausgewählten Musters umgesetzt wurde, lassen sich die entstandenen Klassen dessen Bestandteile (*Views*, *ViewModel* und *Model*) zuordnen. Es existieren eine Vielzahl von *Widgets*, die die graphischen Oberflächen und die Benutzerschnittstelle bilden. Diese entsprechen damit den *Views* innerhalb des Musters. Abbildung 18 zeigt exemplarisch das *MainWidget*, welches das Grundgerüst der Benutzeroberfläche bildet.

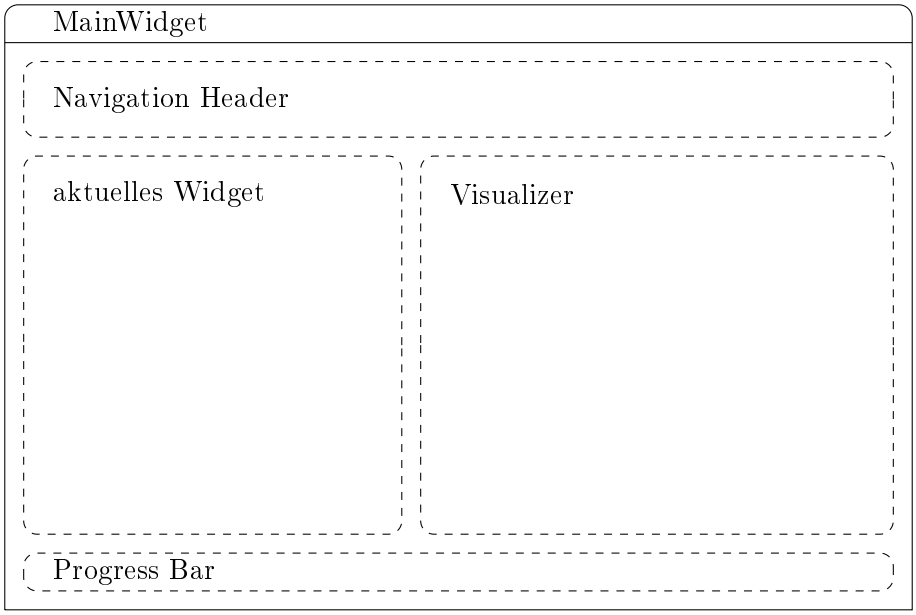


Abbildung 18: Aufbau des *MainWidgets*

Innerhalb des *MainWidgets* werden andere *Widgets* angezeigt oder verborgen, um die Oberfläche entsprechend der benötigten Darstellung für die verschiedenen Teilprozesse zu modifizieren. Das *HeaderWidget* dient dabei der Navigation zwischen den Teilprozessen der Werkzeugkette und das *ProgressWidget* dient der Darstellung von Prozessfortschritten und Statusmeldungen. In dem *VisualizationWidget* werden die dreidimensionalen Punktwolken dargestellt. Zusätzlich existieren für das Öffnen neuer Modelle, das Segmentieren, das Identifizieren, das Ausrichten und für den Export der Ergebnisse weitere *Widgets*, die als interaktive Schnittstelle für den Benutzer fungieren. Die Klasse `QueryObject` ist als Repräsentation

der Daten als *Model* einzustufen. Instanzen dieser Klasse beinhalten alle Daten und Objekte, die im Laufe der Werkzeugkette nötig sind oder währenddessen entstehen. So besitzen diese Objekte eine Punktwolke, Normalen, *Keypoints*, *SHOT-Features* und weitere nötige Werte der jeweiligen Prozesse. Entsprechend des Musters dient die `ViewModel`-Klasse als Bindeglied zwischen den *Widgets* und der `QueryObject`-Klasse. Die Instanz der `ViewModel`-Klasse empfängt die Aktionen, die durch den Benutzer ausgelöst werden, und leitet deren Verarbeitung ein. Dort werden die Prozesse der Werkzeugkette initiiert und anschließend die Objekte des *Models* anhand der Resultate aktualisiert.

Für die Kommunikation zwischen den einzelnen Komponenten werden *Signals* and *Slots* der Bibliothek *Qt* genutzt. Die *Signals* können durch verschiedene Ereignisse wie beispielsweise dem Druck auf einen Button oder durch explizites Emittieren ausgesendet werden [33]. Es existieren demnach eine Vielzahl von standardmäßigen *Signals*. Ein *Signal* kann mit einer Vielzahl von *Slots* verbunden werden. Dabei legt die Anordnung bei der Verbindung von *Signals* und *Slots* die Ausführungsreihenfolge fest. Erfolgt das Aussenden eines *Signals*, werden die damit verbundenen *Slots* beinahe unverzüglich wie bei einem normalen Funktionsaufruf ausgeführt. Des Weiteren sind *Slots* normale Memberfunktionen und können auch normal aufgerufen werden. Eine Besonderheit ist dabei, dass die Ausführung unabhängig von verschiedene Zugriffsniveaus eingeleitet werden kann, insofern eine Verbindung zwischen einem *Signal* und einem *Slot* auf verschiedenen Zugriffsniveaus existieren sollte. *Signals* und *Slots* müssen bei der Verbindung eine identische Signatur besitzen. Die Signaturen von *Slots* können jedoch kürzer sein, was das Ignorieren der zusätzlichen Parameter des *Signals* zur Folge hätte. Ansonsten kann die Anzahl der Parameter jeglichen Datentyps beliebig groß sein. Das innerhalb einer Klasse *Signals* und *Slots* verwendet werden können, muss diese direkt von der `QObject`-Klasse oder einer Unterklasse erben. Der Vorteil dieses Kommunikationsmechanismus liegt in der hohen Flexibilität gegenüber normaler Rückruffunktionen. Für diese Flexibilität ist der Mechanismus etwas langsamer, was innerhalb realer Anwendungen letztendlich unerheblich ist. Auch ist er unabhängig von jeglichen Ereignisschleifen der graphischen Benutzerschnittstellen. *Signals* und *Slots* werden durch das `SIGNAL()`- und das `SLOT()`-Makro definiert und mit der `connect`-Funktion, die von der `QObject`-Klasse vererbt wird, verbunden. Code 12 zeigt als Beispiel die Verbindung zum Öffnen von neuen Dateien. Dabei wird eine Liste von Dateipfaden vom *OpenFilesWidget* zum *ViewModel* durch diese Verbindung weitergereicht, wo der eigentliche Import der Datei in die Anwendung erfolgt. Der `connect`-Funktion wird dabei der Sender, das *Signal*, der Empfänger sowie der *Slot* übergeben.

```
1    this->connect(
2        this,                                // sender
3        SIGNAL(openFiles(QStringList)),      // signal
4        vm,                                  // receiver
5        SLOT(openPointClouds(QStringList))   // slot
```

```
6     );
```

Code 12: Verbindung von Signal und Slot zum Öffnen von Punktwolken

4.4.2 Auslagerung von Prozessen

Unabdingbar ist die Auslagerung von Prozessen mittels Multithreading für diese Anwendung. Einerseits können Prozesse durch die parallele Ausführung beschleunigt werden und andererseits darf die Benutzerschnittstelle bei rechen- und zeitintensiven Operationen nicht blockieren. Anders als bei der üblichen Verwendung von Threads wird nicht von einer `Thread`-Klasse geerbt und die `run`-Methode zur Ausführung der auszulagernden Prozesse überschrieben. Bei der Klasse `QThread`, die von *Qt* bereitgestellt wird, handelt es um eine Schnittstelle [34] zu Objekte der `Thread`-Klasse, die die korrekte Verwendung sicher stellt und die dafür nötige Architektur bereit stellt. Auch wird durch diese Klasse die Kommunikation der Anwendung mit den Threads durch *Signals* und *Slots* gesichert. Für die Ausführung eines Threads wird eine Instanz der Klasse `QThread` und ein Objekt, dass die auszuführenden Prozesse bzw. Operationen enthält, erzeugt. Dieses Objekt wird mithilfe der erzeugten Instanz auf den Thread ausgelagert. Anschließend werden beide mittels entsprechender *Signals* und *Slots* miteinander verknüpft. Durch den Start des Threads wird die Ausführung angestoßen. Wenn die Ausführung beendet ist, signalisiert das Objekt dies der Instanz der `QThread`-Klasse, die den Thread anschließend beendet und zusammen mit dem Objekt löscht. Code 13 zeigt die Erzeugung der nötigen Objekte und den Aufbau ihrer Verbindungen.

```
1     QThread* thread = new QThread;
2     Worker* worker = new Worker();
3     worker->moveToThread(thread);
4     connect(worker, SIGNAL(error(QString)),
5             this, SLOT(errorString(QString)));
6     connect(thread, SIGNAL(started()), worker, SLOT(process()));
7     connect(worker, SIGNAL(finished()), thread, SLOT(quit()));
8     connect(worker, SIGNAL(finished()), worker, SLOT(deleteLater()));
9     connect(thread, SIGNAL(finished()), thread, SLOT(deleteLater()));
10    thread->start();
```

Code 13: Erzeugung und Verbindung eines Threads nach [34]

Durch den Aufruf der `start`-Methode wird der Thread gestartet und durch ein *Signal* an das auszuführende Objekt weitergereicht. Das `finished`-*Signal* des Objekts beendet den laufenden Thread und leitet das Löschen von Thread und Objekt ein. Die Verwendung von *Signals* und *Slots* ist threadsicher und ermöglicht somit eine einfache und gesicherte Verarbeitung der erzielten Ergebnisse, indem diese direkt weitergeleitet werden können. Auch können diese genutzt werden, um Status- oder Fehlermeldung an den Hauptprozess

zu übergeben.

Für die Auslagerung und Parallelisierung von Prozessen der Werkzeugkette wurde die Klasse `Worker` implementiert. Objekte dieser Klasse werden mittels ihrer Konstruktoren alle nötigen Objekte und die auszuführende Aufgabe übergeben. Diese Aufgabe wird in einer Konsolenbefehl ähnlichen Art übergeben und zur Erhaltung der auszuführenden Operation und der nötigen Parameter geparst. Je nach Befehl wird die entsprechende Operation mit den übergebenen Objekten ausgeführt. Nach der Ausführung wird mittels der entsprechenden *Signals* das Resultat an das *ViewModel* weitergegeben und die Abschaltung und Löschung des Threads angestoßen. Das *ViewModel* kümmert sich um die weitere Verarbeitung des Ergebnisses oder aktualisiert damit das *Model*.

5 Durchführung und Auswertung der Fallbeispiele

In diesem Kapitel wird die Evaluierung der Werkzeugkette beschrieben. Zunächst erfolgt eine Erklärung des Aufbaus des Testdatensatzes, die als Referenzteile innerhalb Werkzeugkette genutzt werden. Das Testsystem besteht aus einem MacBook Pro mit einem Intel i5-4278U (2,6 GHz) Prozessor, 8 GB Arbeitspeicher und Windows 7 (SP1) in der 64-Bit Version, welches mittels Apple Boot Camp installiert wurde. Auf diesem System wird für zwei unterschiedliche Fallbeispiele die Werkzeugkette durchlaufen, um ihre Teilprozesse bewerten zu können. Als Fallbeispiele dienen einerseits eine PKW-Lichtmaschine sowie ein Teil einer Turbinenstufe.

5.1 Aufbau des Testdatensatzes

Für die Lichtmaschine existieren zwei verschiedene Datensätze, die als korrekte Referenzteile anzusehen sind. Einerseits wurde diese Lichtmaschine komplett demontiert und die separaten Bauteile jeweils einzeln digitalisiert. Anderseits wurde zusätzlich eine Nachmodellierung der Baugruppe durchgeführt, die als Ergebnis einen CAD-Datensatz lieferte. Um auch diese Modelle für die Werkzeugkette als Referenzteile nutzen zu können, wurde über diese Modelle ein gleichmäßiges Punktenetz gelegt und anschließend im PLY-Format abgespeichert. Für die Turbinenstufe existieren lediglich Referenzteile aus den Digitalisierungen der Einzelteile. Des Weiteren werden zusätzliche Teile, die nicht mit den Baugruppen der Fallbeispiele in Verbindung stehen, als Referenzteile aufgeführt. Diese dienen als Kontrollgruppe, um die Korrektheit der Identifikation besser bewerten zu können. Dabei wurden Modelle ausgewählt, die durchaus wegen ihrer geometrischen Eigenschaften innerhalb der Identifikation als Referenz bewertet werden könnten. Im Anhang sind in Abbildung 26 alle Referenzteile abgebildet.

5.2 Fallbeispiel 1: PKW-Lichtmaschine

Die in Abbildung 19 gezeigte Lichtmaschine besteht aus 665.304 Punkten bei einer Auflösung von $0,267059$ mm und dient als erstes Fallbeispiel. Die Dimensionen dieser Baugruppe betragen etwa $18,5 \times 17,9 \times 14,4$ cm. Ziel dieses Fallbeispiels soll es sein, dass die fünf größten Bauteile dieser Gruppe erfolgreich segmentiert, identifiziert und ausgerichtet werden. Bei den gewünschten Bauteilen handelt es sich um die Riemenscheibe, das Lüfterrad, den Stator und das obere sowie untere Gehäuseteil. Dabei sollte bestenfalls die Rekonstruktion mit den beiden vorhandenen Referenzbaugruppen durchgeführt werden können.

Als erster Teilprozess wird die Segmentierung mit dem *Euclidean Cluster Extraction* und

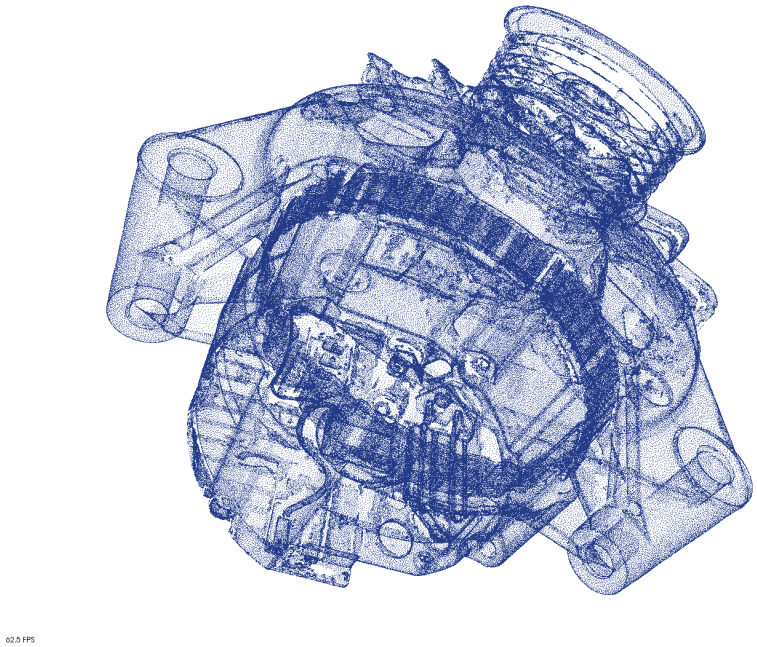


Abbildung 19: Punktwolke der gescannten PKW-Lichtmaschine

dem *Region Growing* Verfahren evaluiert. Zuerst wir das *ECE*-Verfahren auf die Punktwolke aus Abbildung 19 angewandt. Als Startwert für die Toleranz des Algorithmus wird die Auflösung der Punktwolke, die der gemittelten Distanz aller Punkte zu ihrem nächsten Nachbarn entspricht, gewählt. Die anderen Durchläufe wurden dann ab einer Toleranz t von $0,3$ in $0,1$ Schritten ausgeführt. Als minimale Clustergröße gilt für alle Durchläufe $0,1$ Prozent der Anzahl aller Punkte des Scans, was für das konkrete Fallbeispiel $s_{min} = 665$ Punkte entspricht. Die obere Schranke der Clustergröße wurde ausreichend groß gewählt, so dass kein resultierendes Cluster diese Größe überschreiben kann.

Tabelle 4: Resultate der Segmentierung mit dem *Euclidean Cluster Extraction* Verfahren

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 665$, $t = 0,267059$	1544	35	große Mehrheit der Baugruppe verworfen
$s_{min} = 665$, $t = 1,0$	2792	20	Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe stark übersegmentiert
$s_{min} = 665$, $t = 1,7$	4867	8	Stator, Gehäuseteile und Lüfterrad ein Segment, Riemenscheibe leicht übersegmentiert

In Tabelle 4 sind die Durchläufe mit gravierender Änderung des Segmentierungsergebnisses zusammengefasst. Alle durchgeführten Segmentierungen mit diesem Verfahren sind im Anhang in Tabelle 13 aufgeführt. Von der Auslösung der Punktwolken $(0,267059)$ bis zu einer Toleranz von $0,9$ führt die Anwendung dieses Algorithmus zu einer Verwerfung großer Teile der Baugruppe. Mit steigender Toleranz kommen zunächst Kanten und später Flächen wieder hinzu. Zwischen einer Toleranz von $1,0$ und $1,6$ bilden die Gehäuseteile und der Stator ein Segment. Das Lüfterrad zusammen mit der Mantelfläche der Riemenscheibe, die senkrecht auf dem Lüfterrad steht, und der Rest der Riemenscheibe sind jeweils ein Segment. Ab einer Toleranz von $1,7$ sind die Gehäuseteile, der Stator und das Lüfterrad innerhalb eines Segmentes. Damit kann der Algorithmus für eine grobe Aufteilung der Baugruppe in Gehäuse mit Stator, Lüfterrad und Riemenscheibe verwendet werden.

Als nächster Schritt wird zum Vergleich der *Region Growing* Algorithmus zur Segmentierung der kompletten Lichtmaschine genutzt. Da dieses Verfahren drei verschiedene Parameter besitzt, die für das Segmentierungsergebnis entscheidend sind, werden als erstes die Standardwerte der Implementierung von *PCL* für die erste Ausführung genutzt. Dabei werden die 30 nächsten Nachbarn (n) der jeweiligen Punkte betrachtet, eine maximale Abweichung der Normalen (d_a) von 30 Grad toleriert und ein Krümmungsgrenzwert (c) von $0,05$ genutzt. Als erstes wird der Einfluss der jeweiligen Parameter auf die Segmentierung untersucht. Dafür wird zunächst die Größe der Nachbarschaft, dann der maximale Winkel zwischen den Normalen und als letztes der Krümmungsgrenzwert variiert. Bei den Versuchen ließ sich feststellen, dass die Anzahl der zu betrachtenden Nachbarn eher einen geringen Einfluss auf das Segmentierungsergebnis hat. Bei den Ausführungen mit 25 bis 40 Nachbarn waren sich deren Resultate sehr ähnlich. Bei schwachen Grenzwerten führt eine zu große Nachbarschaft zu einer sehr starken Untersegmentierung, bei der mehrere Einzelteile vollständig innerhalb eines Segmentes zusammengefasst werden. Bei dem Grenzwert für den Winkel der Normalen ließ sich feststellen, dass ein zu großer Wert zu Untersegmentierung und ein zu kleiner Wert zu Übersegmentierung führt. Kleine Änderungen des Winkels können zu erheblichen Veränderungen des Resultats führen. Je kleiner der maximale Winkel zwischen zwei Normalen sein darf, desto mehr Segmente entstehen und die benötigte Zeit für die Ausführung steigt an. Für den Krümmungsgrenzwert ließ sich das gleiche Verhalten erkennen, allerdings in einem viel kleineren Maße als bei der Variation des Winkels.

In Tabelle 5 ist die Segmentierung mit den Standardwerten, die jeweils beste Segmentierung bei Variation eines Parameters und ein Segmentierung mit frei gewählten Werten für alle Parameter aufgeführt. Die ausführliche Auflistung aller durchgeführten Versuchsreihen befinden sich im Anhang in Abschnitt 7.2. Bei der Variation der Parameter ließ sich feststellen, dass der maximale Winkel zwischen den Normalen für dieses Beispiel zwischen 5 und 10 Grad liegen muss. Während bei 10 Grad alle gewünschten Segmente voneinander getrennt

Tabelle 5: Resultate der Segmentierung mit dem *Region Growing* Verfahren

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 665$, $d_a = 30$, $c = 0,05$, $n = 30$	8.544	25	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil (stark untersegmentiert) jeweils mit Teile des Stators verbunden
$s_{min} = 665$, $d_a = 30$, $c = 0,05$, $n = 25$	9.026	32	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil sowie Großteil des Stators ein Segment
$s_{min} = 665$, $d_a = 15$, $c = 0,05$, $n = 30$	16.396	36	Riemenscheibe stark übersegmentiert, Lüfterrad und obere Gehäuseteil sehr gut segmentiert, untere Gehäuseteil stark untersegmentiert und mit einem Teile des Stators verbunden
$s_{min} = 665$, $d_a = 30$, $c = 0,40$, $n = 30$	5.341	15	Riemenscheibe weniger stark übersegmentiert, obere und untere Gehäuseteil sowie Lüfterrad und Stator ein Segment
$s_{min} = 665$, $d_a = 8$, $c = 0,45$, $n = 25$	25.553	52	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil leicht übersegmentiert, untere Gehäuseteil gut segmentiert, Stator übersegmentiert

waren, auch wenn dabei teilweise Über- und Untersegmentierung auftrat, so war die Übersegmentierung bei 5 Grad bereits so stark, dass sehr kleine Segmente entstanden, die durch ihre Verwerfung zur Zerstörung der Riemenscheibe führten. Da der Krümmungsgrenzwert standardmäßig bereits sehr klein ist, konnte mit dem Erhöhen des Grenzwertes nur gezeigt werden, dass dessen Einfluss auf das Resultat der Segmentierung geringer ausfällt. Somit ist der entscheidende Parameter der Winkel zwischen den Normalen und der Krümmungsgrenzwert kann zur feinen Korrektur genutzt werden. Die beste Segmentierung ließ sich mit einem Winkel von 8 Grad, einen Krümmungsgrenzwert von $0,45$ und einer Nachbarschaftsgröße von 25 Nachbarn erzielen, da die auftretende Übersegmentierung leicht durch manuelles Zusammenführen einzelner Punktwolken korrigiert werden kann und keine wiederholte Segmentierung benötigt wird. Bei der Anwendung dieser Parameter wurden die Bauteile konkret wie in Tabelle 6 segmentiert.

Zur Bestimmung der Anzahl der Punkte und der Auflösung wurden alle Segmente eines

Tabelle 6: Resultierende Segmente der Bauteile

Bauteil	**Anzahl der Segmente**	**Anzahl der Punkte**	**Auflösung**
Riemenscheibe	17	55.761	$0,203528$
Lüfterrad	1	67.034	$0,248497$
obere Gehäuseteil	4	177.129	$0,343911$
Stator	4	106.242	$0,165073$
untere Gehäuseteil	2	136.035	$0,322756$

Bauteils zusammengeführt und anschließend diese Eigenschaften erfasst. Durch die Kombination von dem *ECE*- und *RG*-Verfahren kann das Segment der Riemenscheibe verbessert werden. Durch die Anwendung vom *ECE*-Verfahren ($t = 1,2$) und die anschließende Anwendung vom *RG*-Verfahren ($d_a = 8, c = 0,45, n = 25$) auf das Segment des Lüfterrades konnte die Punktanzahl auf 69.394 bei einer Auflösung von $0,199526$ gesteigert werden. Die Segmente der anderen Bauteile unterliegen dabei nur minimalen Abweichungen von maximal 250 Punkten gegenüber den dargestellten Werten. Die benötigte Zeit für die einzelnen Segmentierungsschritte summiert sich dabei auf $13,011$ Sekunden und für die manuelle Zusammenführung der übersegmentierten Bauteile waren etwas mehr als 3 Minuten nötig. Das Ergebnis der Segmentierung ist in Abbildung 20 dargestellt.

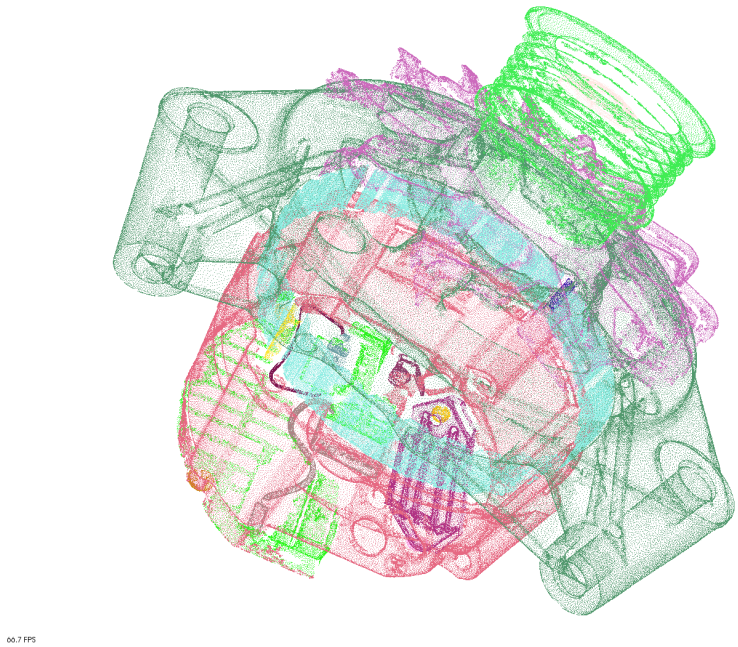


Abbildung 20: Resultierende Segmente der PKW-Lichtmaschine

Nachdem die Segmentierung der PKW-Lichtmaschine erfolgreich durchgeführt wurde, müs-

sen nun die ausgewählten Segmente identifiziert werden. Dafür werden die Segmente einzeln mit den vorhandenen Referenzteilen verglichen und eine Rangfolge anhand der gefundenen und gruppierten Korrespondenzen ihrer *SHOT-Features* erstellt. Dieser Vorgang wird mit unterschiedlichen, maximal zulässigen Abständen innerhalb einer Korrespondenz wiederholt, um den Einfluss dieses Abstandes auf die Korrektheit und der benötigten Zeit abzuschätzen. Für das erste Segment, welches die Riemenscheibe beinhaltet, wird die Identifikation mit maximalen Distanzen von $0,1$ bis $1,0$ in Schritten der Größe $0,1$ wiederholt. Die Versuchsreihe besteht somit aus zehn Identifikationsversuchen, welche die Referenzteile *Pulley* bzw. *Pulley CAD* für eine korrekte Identifizierung des Bauteils als Referenz identifizieren sollten.

Dabei ist festzustellen, dass bis zu einer Distanz von $0,2$ keines der Referenzteile als Referenz ausgegeben wurde. Bei einer Distanz von $0,3$ wurden sowohl das Referenzteil *Pulley* als auch *Pulley CAD* als Referenz identifiziert, während keines der restlichen Referenzteile vorgeschlagen wurde. Somit ist dieses Resultat als vollständig korrekt zu bewerten. Bei den darauffolgenden Versuchen kommen immer weitere Referenzteile fälschlicherweise zum Ergebnis hinzu. Bei der Distanz von $0,4$ steigen die *Votes* der korrekten Referenzen zwar leicht an, dennoch wird bereits ein falsches Referenzteil besser bewertet. Bis zur Distanz von $1,0$ lässt sich erkennen, dass eine großzügig gewählte Distanz keine Verbesserung für die korrekten Referenzen mit sich bringt. Es kommen lediglich immer mehr falsche Referenzteile hinzu, welche bei der Distanz von $1,0$ bis zu dreimal besser bewertet werden. Somit kann dieser Versuchsreihe entnommen werden, dass eine gute Distanz für zulässige Korrespondenzen wohl zwischen $0,3$ und $0,4$ liegen muss. Daher wurde für die restlichen Segmente der PKW-Lichtmaschine die Identifikation nur mit den Distanzen von $0,2$ bis $0,5$ durchgeführt, um diesen Eindruck zu bestätigen. Auch das Lüfterrad konnte bei der kleinen Distanz von $0,2$ nicht identifiziert werden. Jedoch werden bei diesem Segment schon bei dieser Distanz zwei falsche Referenzen geliefert. Bei der Distanz von $0,3$ wird das Referenzteil *Fan CAD* und bei $0,4$ auch das Referenzteil *Fan* korrekt identifiziert. Die Anzahl der falschen Referenzen steigt sehr schnell und daher wurden im letzten Versuch bereits zehn falsche Korrespondenzen geliefert. Auch bei diesem Bauteil erhalten falsche Referenzteile bessere Bewertungen als die korrekten Referenzen, aber nicht im gleichen Maße wie bei der Riemenscheibe. Dagegen fallen die Identifizierungen der beiden Gehäuseteile wesentlich besser aus. Für beide Segmente konnten bereits bei einer Distanz von $0,2$ die korrekten Referenzen gefunden werden. Zwar kommen auch hier mit größerer Distanz viele falsche Referenzen hinzu, aber diese werden bei keinem Versuch besser bewertet als die dazugehörigen Referenzteile. Diese Referenzteile erhalten teilweise eine viermal bessere Bewertung als falsche Referenzen und sind somit eindeutig als Referenz zu erkennen. Die Identifikation des Stators ist dagegen gänzlich als Fehlversuch zu bezeichnen. In keinem Versuch wurde auch nur eines der beiden korrekten Referenzteile zurückgegeben. Allerdings ist bei diesem Bauteil auch anzumerken, dass lediglich die äußere Mantelfläche des Bauteils während des Scans erfasst werden konnte. Somit

besteht das Segment lediglich aus ringförmig angeordneten Punkten und es wurden mit größerer Distanz immer mehr Referenzteile angezeigt, die gewölbte oder ringförmige Bereiche beinhalten. Die Referenzteile *Stator* und *Stator CAD* dagegen wurden wahrscheinlich aufgrund des inneren Aufbaus, der bei der Bestimmung des *SHOT-Features* auch bei *Keypoints* auf der Mantelfläche mit einfloss, verworfen. Es lässt sich zusammenfassen, dass die Identifikation der Bauteile der PKW-Lichtmaschine größtenteils erfolgreich durchgeführt werden kann, wobei allerdings stets falsche Referenzteile mit zurückgegeben werden. Das Lüfterrad braucht für das zweite korrekte Referenzteil zwar eine Distanz von $0,4$, aber das andere konnte bereits mit der kleineren Distanz identifiziert werden. Die Segmente des Stators und teilweise des Lüfterrades zeigen, dass die Identifizierung problematisch sein kann. Zeitlich ist die Identifikation hauptsächlich durch die Anzahl der *Keypoints* des Segments sowie des Referenzteils und nicht durch die veränderten Distanzen bedingt. Während das kleine Segment der Riemenscheibe bei einer Distanz von $0,3$ im Durchschnitt für den Vergleich mit einem Referenzteil etwa $4,3$ Sekunden benötigte, dauerte die Identifikation des oberen Gehäuseteils bei der gleichen Distanz durchschnittlich $15,1$ Sekunden. Durch die feste Kantenlänge des Voxelgitters für die Bestimmung der *Keypoints* generieren großräumigere Segmente automatisch mehr *Keypoints* gegenüber kleineren Segmenten. Nichtsdestotrotz sind die Zeiten für den Abgleich mit den Referenzen für das Fallbeispiel akzeptabel. Alle durchgeführten Versuchsreihen sind tabellarisch samt der genutzten Distanzen, den erzielten *Votes* und der benötigten Zeiten im Anhang 7.4 aufgeführt.

Nachdem den Segmenten die korrekten Referenzteile zugeordnet sind, müssen diese aneinander ausgerichtet werden. Hierfür wird zunächst die automatische Ausrichtung bestehend aus *SAS-IA*- und *ICP*-Verfahren angewendet. Nach der groben Ausrichtung mit dem *SAS-IA*-Verfahren wird das *ICP-Verfahren* maximal zehnmal zur Verbesserung der Lage zueinander wiederholt. Zwischen jeder Anwendung wird die aktuelle Ausrichtung anhand der Ebenenabbildungen bewertet und bei einer Übereinstimmung von 90 Prozent als korrekt abgeschlossen. Wenn der übergebene Prozentsatz nicht erzielt wird, dann wird der Prozess beendet und nach einer manuellen Ausrichtung verlangt. Die Tabellen 7 und 8 zeigen die Bewertungen der automatischen Ausrichtungen sowie die manuellen Korrekturen für beide Referenzdatensätze.

Das Referenzteil der Riemenscheibe, des oberen Gehäuseteils und des Stators erzielten dabei eine Bewertung von über 90 Prozent. Alle Ausrichtungen waren auch sehr gut und nur bei der Riemenscheibe muss das Referenzteil minimal in Bezug zum Segment nach unten geschoben werden. Das Lüfterrad erhielt mit etwa $76,4$ Prozent eine schlechte Bewertung. Das Problem bei dem Lüfterrad besteht darin, dass das Segment und das Referenzteil verkehrt übereinander lagen. Der Grund dafür liegt in der Lage der Korrespondenzen zueinander. Zwar erfüllen diese die Bedingung des Clusteringverfahrens, dennoch zeigen Korresponden-

Tabelle 7: Resultate der Ausrichtung der Einzelscan-Referenzen

Bauteil	**Bewertung**	**Zeit in *ms***	**Kommentar**	**Korrektur**
Riemenscheibe	92,4941 %	125	leichter vertikaler Versatz	Verschiebung Z -2
Lüfterrad	76,4197 %	390	verkehrt herum	Rotation Y 180, Z 11, *ICP*
obere Gehäuseteil	93,5745 %	265	gute Ausrichtung	keine
Stator	99,0374 %	1748	gute Ausrichtung	keine
untere Gehäuseteil	74,3355 %	889	vertikaler Versatz und schief zueinander	Rotation X 7, Y 2 sowie Verschiebung Z 12

zen von Innen nach Außen und andersherum. Da sich die grobe Ausrichtung nur nach den Korrespondenzen richtet, werden die Bauteile verkehrt zueinander grob ausgerichtet. Da das *ICP*-Verfahren nur auf die Reduzierung von nächsten Punkten basiert, kann diese Fehllage nicht durch den Algorithmus korrigiert werden. Durch das Umdrehen des Bauteils, einer leichten Rotation zur Annäherung der Schaufeln und der Anwendung des *ICP*-Verfahrens konnte eine perfekte Ausrichtung erreicht werden. Das untere Gehäuseteil erzielte mit etwa $74,4$ Prozent ebenfalls eine schlechte Bewertung. Das Referenzteil lag verdreht, verkantet und etwas unterhalb des Segments. Durch leichte Rotation und Verschiebung konnte die schlechte Lage aber einfach korrigiert werden. Die Dauer der automatischen Ausrichtung reicht dabei von etwa $0,1$ bis maximal $1,7$ Sekunden. Wenn die Lage bereits nach wenigen Iterationen als ausreichend gut bewertet wird, dann terminiert der Prozess der automatischen Ausrichtung früher. Falls dennoch alle Iterationen durchgeführt werden müssen, da keine gute Ausrichtung erreicht wird, geschieht dies auch in sehr kurzer Zeit.

Abbildung 21 zeigt die Rekonstruktion des Scans aus allen zusammengefügten Einzelscan-Referenzteilen. Im Vergleich zur Punktwolke der PKW-Lichtmaschine der Abbildung 19, die als Eingabe der Werkzeugkette diente, liegen alle Bauteile an den gleichen Positionen.

Bei der automatischen Ausrichtung der Referenzteile aus den CAD-Modellen erzielten nur die Riemenscheibe und der Stator eine Bewertung von über 90 Prozent. Die Riemenscheibe bedurfte wieder einer minimalen vertikalen Verschiebung und der Stator keinerlei manueller Nachbearbeitung. Das obere Gehäuseteil lag mit $88,2$ Prozent knapp unter dem festgelegten Schwellwert. Dennoch war keine manuelle Korrektur der Ausrichtung für dieses Bauteil notwendig. Das Verfehlen des Schwellwertes ist mit leichten Unterschieden im geometrischen Aufbau des Bauteils und des Referenzteils zu begründen, da diese dadurch nie eine voll-

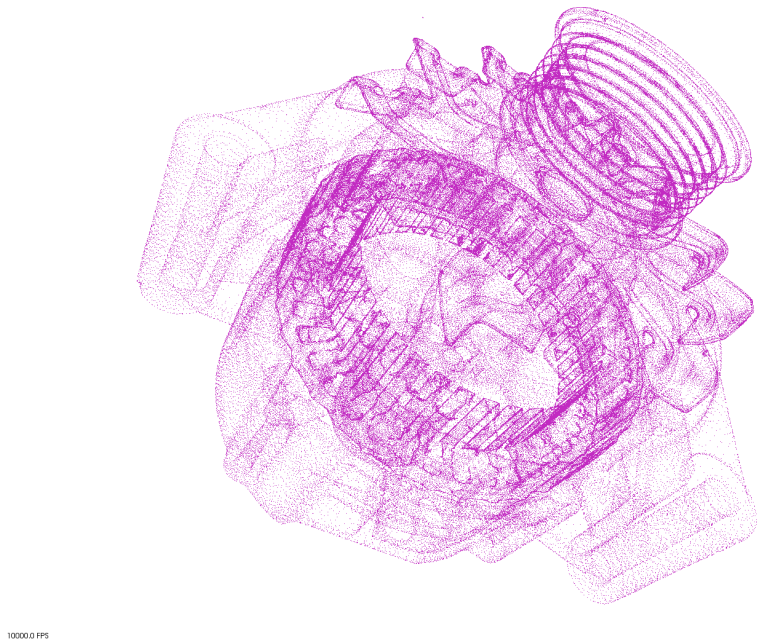


Abbildung 21: Rekonstruktion der PKW-Lichtmaschine mit den Referenzteilen aus den Einzelscans

Tabelle 8: Resultate der Ausrichtung der CAD-Referenzen

Bauteil	**Bewertung**	**Zeit in ms**	**Kommentar**	**Korrektur**
Riemenscheibe	93,5715 %	156	leichter vertikaler Versatz	Verschiebung Z -1
Lüfterrad	86,6948 %	344	leicht horizontal verschoben	Verschiebung X -6, Y -3
obere Gehäuseteil	88,1527 %	1154	gute Ausrichtung	keine
Stator	97,0526 %	218	gute Ausrichtung	keine
untere Gehäuseteil	67,8887 %	904	vertikaler Versatz, schief und verdreht zueinander	Rotation X 4, Y 5, Z 91 sowie Verschiebung Z 13

ständige Übereinstimmung erzielen könnten. Das Lüfterrad mit $86,7$ Prozent ist ebenfalls knapp unterhalb des Grenzwertes. Da dieses Lüfterrad ebenfalls nicht die selbige geometrische Struktur hat, erlauben die Abweichung keine vollständig korrekte Ausrichtung. Dennoch liegen beide Lüfterräder gut zueinander und lediglich ihre Mittelpunkte stimmen nicht ganz überein. Das untere Gehäuseteil mit $67,9$ Prozent ist bei dieser automatischen Ausrichtung am schlechtesten bewertet. Es ist ähnlich wie in der ersten Versuchsreihe mit dem anderen Referenzdatensatz ausgerichtet. Die Differenz in den Bewertungen zwischen den beiden korrekten Referenzen ist auch mit der abweichenden geometrischen Struktur zu begründen. Die

aufzubringende Zeit für die automatische Ausrichtung bewegt sich im gleichen Rahmen wie bei der vorhergehenden Ausrichtung der anderen Referenzteile.

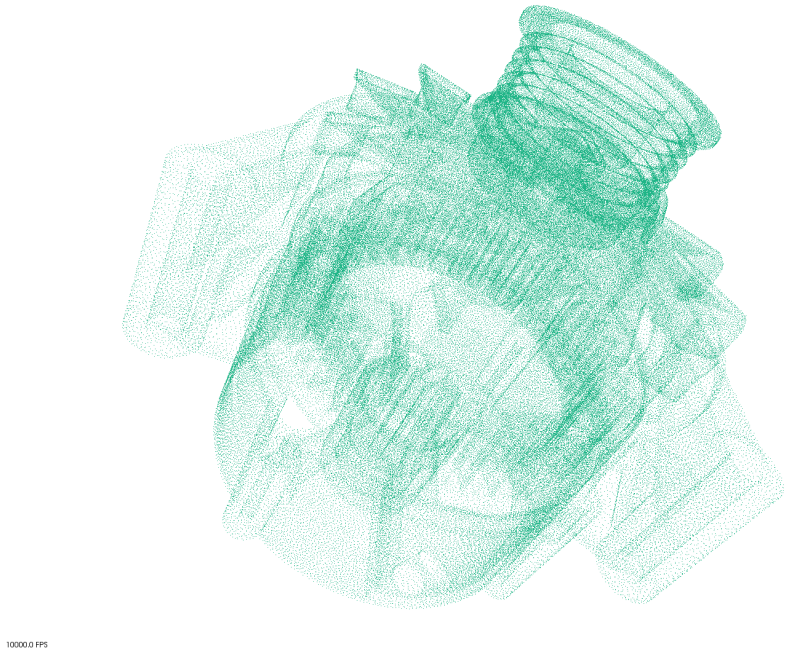


Abbildung 22: Rekonstruktion der PKW-Lichtmaschine mit den Referenzteilen aus den CAD-Modellen

Abbildung 22 zeigt ebenfalls die Rekonstruktion des Scans, wobei in diesem Falle die CAD-Referenzteile genutzt wurden. Die Positionierung der Bauteile ist im Vergleich zur Ausgangslage aus Abbildung 19 ebenfalls identisch.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<bom>
    <assembly name="Lichtmaschine">
        <pcpath>C:\Users\Robert\Desktop\Lichtmaschine.ply</pcpath>
        <part id="0">
            <name>Pulley CAD Reference</name>
            <pcpath>C:\Users\Robert\Desktop\2016-02-19_11-19-50_RE-Cockpit\
                Segments\Segment0.ply</pcpath>
            <refpath>C:\Users\Robert\Desktop\2016-02-19_11-19-50_RE-Cockpit\
                References\Pulley CAD Reference.ply</refpath>
            <transformMatrix type_id="opencv-matrix">
                <rows>4</rows>
                <cols>4</cols>
                <data>0.729646 -0.683135 -0.0307628 3.28247 -0.683822
                    -0.728704 -0.0371773 8.48186 0.00298017 0.0481625
                    -0.998837 91.7031 0 0 0 1</data>
                <dt>float</dt>
            </transformMatrix>
        </part>
        ...
```

```
17     </assembly>
18 </bom>
```

Code 14: Auszug aus entstandener XML-Datei für die PKW-Lichtmaschine

Code 14 zeigt einen Auszug der entstandenen XML-Datei, der alle Informationen zum Bauteil der Riemenscheibe beinhaltet. Für die anderen Bauteile existiert jeweils ein weiterer Eintrag vom selben Aufbau und mit deren spezifischen Informationen. Sowohl das Segment als auch die Referenz wurde an den angegebenen Pfaden abgelegt und mittels Anwendung der gezeigten Transformationsmatrix kann die Referenz in die Lage des Segments überführt werden. Auch während der ersten Durchführung der Werkzeugkette wurde eine solche XML-Datei mit den Referenzen aus den Einzelscans erzeugt.

Dieses Fallbeispiel hat gezeigt, welche Möglichkeiten die Werkzeugkette bietet und welche Probleme bestehen. Bei der Segmentierung wurde deutlich, dass die einmalige Anwendung eines Segmentierungsverfahrens nicht ausreichend ist. Innerhalb einer Baugruppe können die unterschiedlichsten geometrischen Bauteile zusammenkommen und diese können auf die verschiedensten Arten aneinander befestigt sein. So können Bauteile ineinander eingelassen werden, beinahe perfekt aufeinander liegen oder durch verschieden große Spaltmaße getrennt sein. Kommen all diese Varianten innerhalb einer Baugruppe zusammen, ist es schwer einen Algorithmus und die richtige Wahl der Parameter zu finden, bei der alle Bauteile perfekt voneinander getrennt werden. Dennoch haben die Versuchsreihen gezeigt, dass die Kombination der eingesetzten Verfahren unter Auftreten von Übersegmentierung und Anwendung manuellen Zusammenführens eine gute und ausreichend schnelle Segmentierung zulässt. Übersegmentierung ist durch verschiedene Umstände begründet. Beispielsweise können Bauteile während der Digitalisierung nicht bis ins letzte Detail erfasst werden oder sind teilweise durch andere Teile verdeckt. Bei dieser Baugruppe war die Riemenscheibe beispielsweise relativ schlecht erfasst worden und wurde deshalb stark übersegmentiert, während die anderen Bauteile nur sehr leicht übersegmentiert waren. Der Stator wurde auch stets übersegmentiert, was auf die Befestigungsschrauben, die die beiden Gehäuseteile miteinander verbinden, zurückzuführen ist. Diese teilen den Stator optisch in vier Abschnitte auf und so wurde dieser auch stets segmentiert. Die Versuchsreihen zur Identifizierung der Bauteile waren größtenteils erfolgreich. Das Digitalisieren einer Baugruppe im montierten Zustand birgt immer die Gefahr, dass Bauteile nicht vollständig erfasst werden können. Dagegen sind die Referenzteile natürlich vollständig und somit kann es zu gravierenden Unterschieden in den Modellen kommen, die keine Identifizierung zulassen. Sind die einzelnen Bauteile aber ausreichend erfasst, kann mithilfe von lokalen *Features* eine Identifizierung dieser erfolgen. Der Stator war auch von diesem Umstand betroffen, da sein innerer Aufbau, der in den Referenzteilen vorhanden ist, nicht erfasst werden konnte und daher fehlt. Aus diesem Grund konnte dieses Bauteil nicht korrekt identifiziert werden, da bei der Bestimmung der *Features* die innere Struktur mit einfließen hätte müssen. Es wurde auch gezeigt, dass bei

der Identifikation schnell falsche Referenzteile ausgegeben werden. Lokale Bereiche können in den verschiedensten Bauteilen identisch oder sehr ähnlich sein. Doch der ganzheitliche Vergleich von Bau- und Referenzteil kann infolge der bereits erwähnten unvollständigen Erfassung ebenfalls problematisch sein. Nachdem die Bauteile mittels Segmentierung und deren Referenzen mittels Identifikation hervorgebracht wurden, musste eine Ausrichtung zur Rekonstruktion der Baugruppe aus den Referenzen erfolgen. Die Versuchsreihen der automatischen Ausrichtung der jeweiligen Bauteile zeigten sowohl Erfolge als auch Fehlschläge. Da die grobe Ausrichtung auf die gefundenen Korrespondenzen der *Features* zwischen den Bauteilen basiert, ist deren Qualität entscheidend für das Resultat der Ausrichtung, welches durch die feine Ausrichtung weiter verbessert werden kann. Wenn Korrespondenzen zwischen markanten geometrischen Merkmalen gebildet wurden, ist die Ausrichtung wesentlich besser, als wenn nur Korrespondenzen beispielsweise auf Flächen zur Verfügung stehen. Die Gehäuseteile haben dieses Problem sehr gut verdeutlicht. Während das obere Gehäuseteil mit seinen charakteristischen Befestigungspunkten sehr gut ausgerichtet wird, schlägt die Ausrichtung bei dem unteren Gehäuseteil fehl, da dieses Bauteil rundum gleichartig aufgebaut ist und Korrespondenzen nicht immer die korrekten Seiten oder Kanten des Bauteils verbinden müssen. Das Lüfterrad hat auch gezeigt, dass das nicht nur zu einer Schieflage führen muss, sondern es auch zu einer extremen Fehlausrichtung kommen kann und die Bauteile verkehrt herum aneinander ausgerichtet sind. Doch all die entstandenen Fehllagen konnten schnell durch manuelles Rotieren und Verschieben korrigiert werden, so dass alle Referenzen richtig positioniert werden konnten.

5.3 Fallbeispiel 2: Turbinenstufe

Das in Abbildung 23 gezeigte Teil einer Turbinenstufe besteht aus 1.405.759 Punkten bei einer Auflösung von $0,0751498$ mm und ist das zweite Fallbeispiel. Die Turbinenstufe besitzt einen Radius von etwa $7,2$ cm und ist $1,5$ cm stark. Die Dimensionen der Turbinenschaufeln betragen etwa $2,7 \times 3 \times 0,8$ cm. Damit im weiteren Verlauf eine eindeutige Zuordnung zur entsprechenden Turbinenschaufel möglich ist, werden diese von links nach rechts durchnummeriert und im Folgenden Turbinenschaufel 1, Turbinenschaufel 2 usw. genannt. Ziel dieses Fallbeispiels soll es sein, dass die Turbinenstufe und die darauf angeordneten Schaufeln erfolgreich segmentiert, identifiziert und ausgerichtet werden.

Auch bei diesem Fallbeispiel wird zuerst die Segmentierung dieser Baugruppe durchgeführt und analog zum ersten Fallbeispiel zunächst das *Euclidean Cluster Extraction* Verfahren dafür eingesetzt. Als Toleranz für den Algorithmus wird im ersten Durchlauf ebenfalls wieder die Auflösung der Punktwolke genutzt und in den darauffolgenden Durchläufen schrittweise erhöht. Aufgrund der sehr kleinen Auflösung wird die Toleranz ab $0,1$ nur um $0,05$ für den

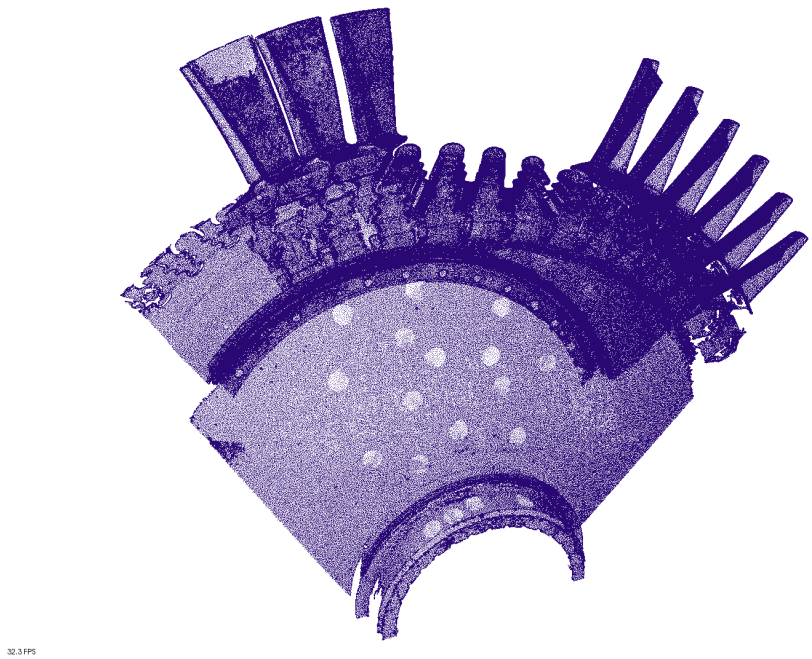


Abbildung 23: Punktwolke der gescannten Turbinenstufe

nächsten Durchlauf erhöht. Für die minimale Clustergröße gilt auch hier $0,1$ Prozent der Anzahl aller Punkte des Scans, was einer Größe von etwa 1400 Punkten entspricht. Die obere Schranke wird wieder vernachlässigt, um die resultierenden Cluster nicht nach oben zu beschränken.

Tabelle 9: Ergebnisse der Segmentierung der Turbinenstufe mit dem *Euclidean Cluster Extraction* Verfahren

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 1.400$, $t = 0,0751498$	3.089	31	große Mehrheit der Baugruppe verworfen
$s_{min} = 1.400$, $t = 0,1$	3.182	60	große Mehrheit der Baugruppe verworfen, einige Kanten teilweise erhalten
$s_{min} = 1.400$, $t = 0,15$	3.806	30	Schaufeln und Rand der Stufe ein Segment, Großteil der Flächen verworfen
$s_{min} = 1.400$, $t = 0,2$	4.555	4	Schaufeln und Großteil der Stufe ein Segment

Tabelle 9 zeigt, dass durch die Anwendung des Algorithmus keine Trennung der Schaufeln und der Stufe möglich ist. Bei einer Toleranz von bis zu $0,1$ ist das selbe Verhalten, wie bei dem ersten Fallbeispiel zu erkennen. Der Großteil der Punktwolke wird dabei verworfen und nur einige kleine Bereiche bleiben erhalten. Bei einer Toleranz von $0,15$ werden Flächen

größtenteils immer noch verworfen und dennoch bilden die Schaufeln und der Rand der Stufe bereits ein Segment. Somit können die gewünschten Bauteile nicht ansatzweise mit dem *ECE*-Verfahren voneinander getrennt werden.

Da bereits im ersten Fallbeispiel festgestellt werden konnte, dass die Standardwerte des *Region Growing* Verfahrens von *PCL* weit von einer optimalen Segmentierung entfernt liegen, wird bei diesem Fallbeispiel bei den besten Parametern des vorherigen Beispiels angesetzt. Somit bilden ein Winkel von 8 Grad, ein Krümmungsgrenzwert von $0,45$ und eine Nachbarschaftsgröße von 25 Nachbarn die Ausgangslage für die Segmentierung mit diesem Verfahren.

Tabelle 10: Ergebnisse der Segmentierung der Turbinenstufe mit dem *Region Growing* Verfahren

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 1.400$, $d_a = 8$, $c = 0,45$, $n = 25$	171.439	8	vorderer Rand ein Segment, hinterer Rand zwei Segmente, Schaufel und Stufe ein Segment, einzelne Flächen ein Segment
$s_{min} = 1.400$, $d_a = 5$, $c = 0,45$, $n = 25$	446.735	57	Stufe stark übersegmentiert, Schaufeln teilweise zusammen und alle ohne Fuß
$s_{min} = 1.400$, $d_a = 2$, $c = 0,45$, $n = 25$	1.572.434	67	Stufe stark übersegmentiert und teilweise verworfen, Schaufeln übersegmentiert und Fuß fast immer verworfen

In Tabelle 10 sind die Durchläufe mit wesentlichen Unterschieden im Segmentierungsergebnis dargestellt. Im Anhang in Abschnitt 7.3 sind alle durchgeführten Versuchsreihen des *RG*-Verfahrens für dieses Fallbeispiel aufgelistet. Bei dem ersten Durchlauf mit den Parametern des ersten Fallbeispiels konnten keine Trennung von Schaufeln und Stufe erreicht werden. Daher wurden die anschließenden Versuche mit kleineren Winkeln durchgeführt. Bei einem Winkel von 5 Grad wurden die Schaufeln von der Stufe getrennt. Allerdings bilden dabei mehrere Schaufeln ein Segment. Die Stufe wird dabei bereits sehr stark übersegmentiert. Nicht nur der vordere und hintere Ansatz wurde von der Stufe getrennt, auch die Vertiefungen, in denen keine Schaufeln eingeschoben waren, wurden in viele Segmente aufgeteilt. Bei einem Winkel von 3 Grad konnten die Schaufeln getrennt werden. Dabei tritt ebenfalls eine starke Übersegmentierung einzelner Schaufeln auf. Bei der Stufe ist die Übersegmentierung bei diesem Winkel derart stark, dass Teile davon bereits verworfen wurden.

Um die gewünschten Segmente zu erhalten, müssen mehrere Segmentierungsdurchläufe ausgeführt werden. Zunächst wird die komplette Punktwolke mit dem *ECE*-Verfahren mit einer Toleranz von $0,2$ segmentiert. Dadurch werden die Ansätze von der Stufe getrennt. Dies beschleunigt einerseits die anschließende Anwendung des zweiten Verfahrens. Anderseits wird dadurch eine zu starke Übersegmentierung dieser Bereiche verhindert. Darauf folgt das *RG*-

Verfahren mit einem Winkel von 5 Grad, einem Krümmungsgrenzwert von $0,45$ und 25 Nachbarn. Nun sind die Schaufeln von der Stufe getrennt und alle Segmente der Stufe können wieder zusammengeführt werden. Ein entstandenes Segment besteht dabei aus Turbinenschaufel 2 sowie Turbinenschaufel 3 und ein weiteres aus den Turbinenschaufeln 4 bis 9. Diese Segmente werden nochmals zur Trennung der einzelnen Schaufeln mit strengeren Parametern segmentiert. Für das erste Segment der Schaufeln wird das RG-Verfahren mit $2,5$ Grad, einem Krümmungsgrenzwert von $0,45$ und 25 Nachbarn genutzt. Bei dem anderen Segment beträgt der Winkel 2 Grad und es wurden jeweils nur 20 Nachbarn betrachtet.

Tabelle 11: Resultierende Segmente der Bauteile

Bauteil	**Anzahl der Segmente**	**Anzahl der Punkte**	**Auflösung**
Turbinenstufe	34	714.692	$0,0782754$
Turbinenschaufel 1	2	48.917	$0,0788893$
Turbinenschaufel 2	7	41.923	$0,0889923$
Turbinenschaufel 3	4	39.203	$0,0883537$
Turbinenschaufel 4	4	31.819	$0,10785$
Turbinenschaufel 5	5	33.577	$0,100756$
Turbinenschaufel 6	5	37.201	$0,0984344$
Turbinenschaufel 7	4	35.174	$0,0995092$
Turbinenschaufel 8	6	36.177	$0,0946942$
Turbinenschaufel 9	6	34.852	$0,0935051$

Abbildung 24 zeigt die resultierende Segmentierung nach dem Zusammenführen der übersegmentierten Teile der einzelnen Bauteile. Die Anwendung der Segmentierungsverfahren benötigte dabei eine Zeit von $349,462$ Sekunden. Die manuelle Korrektur der Übersegmentierung beanspruchte etwa 8 Minuten und 40 Sekunden.

Nachdem die Baugruppe erfolgreich in die gewünschten Segmente unterteilt wurde, kann nun die Identifikation der Bauteile erfolgen. Auch bei der Identifikation werden wie bei der Segmentierung die gewonnenen Erkenntnisse des ersten Fallbeispiels genutzt. Folglich wird die Identifizierung wieder mit den Distanzen von $0,2$ bis $0,5$ für zulässige Korrespondenzen durchgeführt. Bei den ersten beiden Versuchen zur Identifizierung der Turbinenstufe wird zunächst eine und anschließend fünf falsche Referenzen ausgegeben. Das Problem bei diesem Bauteil ist, dass der charakteristische Kranz durch die eingeschobenen Turbinenschaufeln größtenteils fehlt, da dieser während des Scannens nicht erfasst werden kann. Die fälschlicherweise angegebenen Referenzen besitzen alle eine Gemeinsamkeit. Alle Referenzteile zeichnen sich teilweise durch gebogene Flächen aus. Diese bilden Korrespondenzen mit der gebogenen Seitenfläche der Stufe. Erst bei einer Distanz von $0,4$ konnte die Turbinenstufe identifiziert

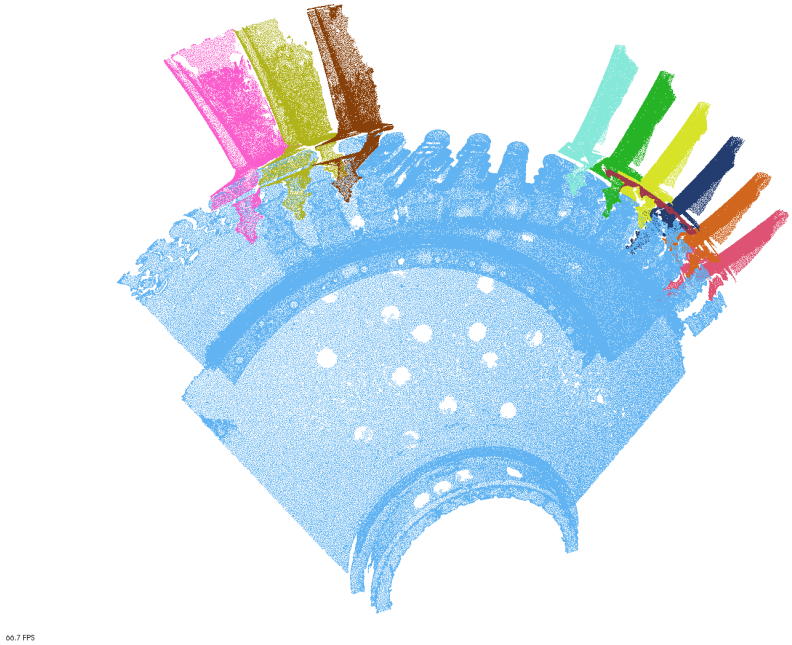


Abbildung 24: Resultierende Segmente der Turbinenstufe

werden. Die Zahl der falschen Referenzteile steigt auf zehn Stück. Dennoch erhält das richtige Referenzteil ähnlich viele *Votes* wie die falschen Referenzteile. Lediglich eine andere Referenz wird mit doppelt so vielen *Votes* bewertet. Bei der Distanz von $0,5$ ist auch bei diesem Fallbeispiel zu erkennen, dass eine groß gewählte Distanz keine Verbesserung für das korrekte Referenzteil impliziert. Während die *Votes* der falschen Referenzen weiter steigen, verbessert sich die Bewertung des gewünschten Referenzteils nicht. Bei der Identifikation der Turbinenschaufel 1 wurde bereits bei der Distanz von $0,2$ das korrekte Referenzteil zurückgegeben. Die weiteren Durchläufe mit größerer Distanz verdeutlichen ebenfalls, dass schnell falsche Referenzteile zur Ausgabemenge hinzugefügt werden und diese schneller mehr *Votes* erhalten. In den einzelnen Durchläufen ist die beste Bewertung eines falschen Referenzteils maximal doppelt so gut wie die der korrekten Referenz. Für die Turbinenschaufel 2 zeigt sich ein ähnliches Resultat. Die richtige Referenz wird bereits ab der Distanz von $0,2$ ausgegeben und in den darauffolgenden Durchläufen steigt die Menge falscher Referenzteile und deren Anzahl an *Votes*. Ein kleiner Unterschied liegt in der zurückgegebenen Menge der falschen Referenzteile. Dabei ist ein Zusammenhang mit der Qualität der Punktwolke des Segments zu vermuten. Die Punktwolke der Turbinenschaufel 1 hat im Vergleich zur Turbinenschaufel 2 etwa 14 Prozent mehr Punkte und ist daher besser erhalten. Generell lässt sich für die Turbinenschaufeln feststellen, dass das korrekte Referenzteil stets mit ausgegeben wurde. Für die Turbinenschaufeln 3 bis 9 wurde die Identifikation nur noch mit einer Distanz von $0,3$ durchgeführt, da bereits bei dieser Distanz die korrekte Referenz und mehrere falsche Referenzteile zurückgegeben werden. Es lässt sich zusammenfassen, dass das Spektrum der

Bewertung des korrekten Referenzteils von am besten bewertet, bis nur halb so gut bewertet wie ein falsches Referenzteil reicht. Je nach Beschaffenheit der Punktwolke der einzelnen Segmente variiert die Menge und die darin enthaltenen falschen Referenzteile. Aber für jedes falsche Referenzteil lassen sich immer mindestens zwei Turbinenschaufeln finden, die diese Referenz bei einer Distanz von $0,3$ erhalten. Nichtsdestotrotz konnten alle Bauteile dieser Baugruppe automatisch identifiziert werden. Zeitlich lässt sich die Identifizierung der Turbinenstufe mit den kleineren Teilen der PKW-Lichtmaschine vergleichen, da der Vergleich mit einer Referenz durchschnittlich $4,7$ Sekunden bei der genutzten Distanz von $0,3$ benötigte. Für die Turbinenschaufeln mit ihren kleinen Dimensionen konnte die Identifikation wesentlich schneller durchgeführt werden. Im Durchschnitt betrug die Zeit zum Vergleich mit einem Referenzteil bei selbiger Distanz nur $0,73$ Sekunden. Alle durchgeführten Identifizierungen für dieses Fallbeispiel sind ebenfalls tabellarisch im Anhang in Abschnitt 7.5 zu finden.

Nachdem für alle Segmente die Referenzen bestimmt worden sind, müssen diese wieder ausgerichtet werden. Zuerst erfolgt für alle Paare von Segmenten und Referenzen die automatische Ausrichtung. Auch für dieses Fallbeispiel gilt ein Schwellwert von 90 Prozent bei der Übereinstimmung der Ebenenabbildungen zur Bewertung der Lage als ausreichend gute Ausrichtung.

Tabelle 12 fasst die Ergebnisse der automatischen Ausrichtung zusammen. Dabei erhält die Ausrichtung der Turbinenstufe eine Bewertung von etwa $71,5$ Prozent und erfordert demnach eine manuelle Korrektur. Auch bei diesem Bauteil liegen das Segment und die Referenz nach der groben Ausrichtung verkehrt ineinander. Durch die nachträgliche feinere Ausrichtung mit dem *ICP*-Verfahren kann diese Fehllage nicht mehr korrigiert werden. Da es sich sowohl bei dem Segment als auch bei der Referenz nur um einen Ausschnitt einer kompletten Stufe handelt, kann es zu diesem Fehler kommen. Wenn beide Punktwolken das jeweils vollständige Bauteil enthalten würden, dann würde die automatische Ausrichtung ein besseres Resultat erzielen. Die Ausrichtungen der Referenzteile der Riemenscheibe des ersten Fallbeispiels wären dafür vergleichbare Beispiele, die bereits gezeigt haben, dass die automatische Ausrichtung von Rotationskörpern erfolgreich sein kann. Durch das Umdrehen der Referenz und leichte Korrekturen kann schnell eine richtige Ausrichtung erzielt werden.

Zwar erhalten alle Turbinenschaufeln im Gegensatz zur Stufe eine bessere Bewertung und einige überschreiten selbst den vorgegebenen Schwellwert, dennoch war die erzielte Ausrichtung für keine Turbinenschaufel annähernd akzeptabel. Bei allen Turbinenschaufeln war die Referenz stets verkehrt herum, verdreht oder verschoben. Die Problematik dieser Bauteile liegt an dem Aufbau der Turbinenstufe und den Schaufeln. Da der Fuß der Schaufel in der Stufe eingelassen ist, kann dieser bei der Digitalisierung nicht erfasst werden. Somit fehlt dieser nach der Segmentierung beinahe vollständig und nur dessen Vorder- oder Rückseite

Tabelle 12: Resultate der Ausrichtung der Referenzen

Bauteil	**Bewertung**	**Zeit in *ms***	**Kommentar**	**Korrektur**
Turbinenstufe	69,966 %	922	verkehrt herum	Rotation X 166, Y 2, Z 8 sowie Verschiebung X -7, Y 32
Turbinenschaufel 1	82,025 %	266	verkehrt herum und verdreht	Rotation X 180, ICP, Rotation Y -3, Z 2 sowie Verschiebung X -1, Y -3, Z -2
Turbinenschaufel 2	82,9266 %	390	verdreht und vertikaler Versatz	Rotation Y 180, ICP, Rotation X 9, Y 8, Z -9 sowie Verschiebung X 1, Y -4, Z -1
Turbinenschaufel 3	94,5599 %	96	verdreht und vertikaler Versatz	Rotation Y 180, ICP, Rotation X -2, Y 3, Z -2 sowie Verschiebung X -2, Y -1, Z -1
Turbinenschaufel 4	89,0558 %	275	verdreht und vertikaler Versatz	Rotation Y 180, ICP, Rotation X -6, Y 9 sowie Verschiebung X 1, Y -5, Z 2
Turbinenschaufel 5	89,323 %	256	verdreht und vertikaler Versatz	Rotation Y 180, ICP, Rotation X 2, Z 6 sowie Verschiebung X -1, Y -5, Z 1
Turbinenschaufel 6	87,6724 %	309	leicht verdreht und vertikaler Versatz	Rotation X 7, Y 25, Z -5 sowie Verschiebung X 2, Y -6, Z 3
Turbinenschaufel 7	94,7224 %	93	verdreht und umgekippt	Rotation Y 90, Z 180, ICP, Rotation X 10, Y 4, Z -1 sowie Verschiebung Y -5, Z 1
Turbinenschaufel 8	88,1905 %	253	umgekippt	Rotation X -90, Z 90, ICP, Rotation X -1, Y 9, Z -1 sowie Verschiebung X 1, Y -4, Z 3
Turbinenschaufel 9	90,4797 %	118	verdreht und vertikaler Versatz	Rotation X-90, Y -180, ICP, Rotation X -25, Y 2, Z 10 sowie Verschiebung X 1, Y -4, Z 3

bleiben teilweise erhalten. Dadurch sind die Korrespondenzen zwischen dem Segment und der Referenz lediglich zwischen den Flächen des jeweiligen Schaufelblattes. Bei der Bildung der Korrespondenzen werden *Keypoints* auf den unterschiedlichen Seiten des Blattes miteinander verbunden, da sich diese Flächen sehr ähnlich sind. Die grobe Ausrichtung verringert anschließend den Abstand dieser Flächen und durch die Anwendung des *ICP*-Verfahrens werden diese Flächen letztendlich übereinander gelegt. Da es sich aber um die gegenüber liegenden Seiten handelt, entstehen die beobachteten Fehllagen. Durch den fehlenden Fuß und durch die großen Flächen des Blattes, die verdreht ineinander liegen, entsteht eine große Überschneidung bei den Ebenenabbildungen und impliziert fälschlicherweise eine gute Ausrichtung. Daher müssen alle Turbinenschaufeln manuell richtig ausgerichtet werden. Die manuelle Ausrichtung ist durch ihre radiale Anordnung und ihrer unabhängigen Lage von einer oder mehrerer Koordinatenachsen etwas schwieriger. Im Durchschnitt war für die Ausrichtung einer Turbinenschaufel mehrere Minuten notwendig und diese war nicht immer

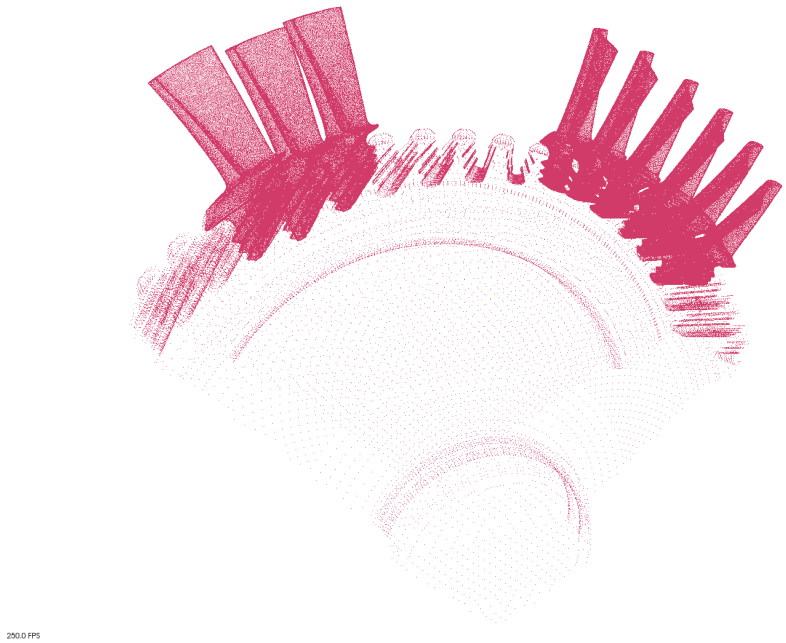


Abbildung 25: Rekonstruktion der Turbinenstufe mit den Referenzteilen

perfekt präzise möglich.

In Abbildung 25 ist die Rekonstruktion der Turbinenstufe aus allen zusammengefügten Referenzteilen zu sehen. Im Vergleich zur Punktwolke des Scans aus Abbildung 23 liegen alle Turbinenschaufeln in den richtigen Vertiefungen der Stufe, allerdings gibt es kleinere Differenzen in der Höhe oder im Abstand der oberen Kante.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<bom>
    <assembly name="Turbinenstufe">
        <pcpath>C:\Users\Robert\Desktop\Turbinenstufe.stl</pcpath>
        <part id="0">
            <name>Turbinenstufe Reference</name>
            <pcpath>C:\Users\Robert\Desktop\2016-02-04_11-39-45_RE-Cockpit\
                Segments\Segment0.ply</pcpath>
            <refpath>C:\Users\Robert\Desktop\2016-02-04_11-39-45_RE-Cockpit\
                References\Turbinenstufe Reference.ply</refpath>
            <transformMatrix type_id="opencv-matrix">
                <rows>4</rows>
                <cols>4</cols>
                <data>0.0304099 -0.976447 0.213698 16.2027 0.0846863 0.215543
                     0.972826 -46.8433 -0.995953 -0.0114857 0.0892446 9.02992
                     0 0 0 1</data>
                <dt>float</dt>
            </transformMatrix>
        </part>
        ...
    </assembly>
</bom>
```

Code 15: Auszug aus entstandener XML-Datei für die Turbinenstufe

In Code 15 ist wieder ein exemplarischer Auszug der XML-Datei, die während der Durchführung der Werkzeugkette für dieses Fallbeispiel erzeugt wurde. Es beinhaltet die bekannten Informationen für das Bauteil der Turbinenstufe.

Dieses Fallbeispiel bestätigte die Erkenntnisse des ersten Fallbeispiels. Da es innerhalb dieser Baugruppe sehr wichtig ist, dass sich die Bauteile sehr dicht aneinander anreihen, wurden die Bauteile alle übersegmentiert. Bei zu schwachen Parametern waren die Schaufeln noch mit der Stufe oder miteinander verbunden und bei den Parametern, die alle Turbinenschaufeln voneinander abgrenzten, ging die Übersegmentierung mit einher. Somit war eine etwas umfangreichere, manuelle Korrektur nötig, doch das Segmentierungsergebnis war letztendlich ebenfalls zufriedenstellend. Die Versuchsreihen zur Identifizierung der Bauteile waren alle erfolgreich. Dabei war die Identifizierung der Schaufeln mit einer kleineren Korrespondenzendistanz möglich als bei der Stufe. Durch die eingelassenen Schaufeln fehlt der markante Kranz des Bauteils größtenteils und unterschied sich daher etwas stärker von dem Referenzteil. Letztendlich wurde das Bauteil bei mehr zugelassenen Korrespondenzen noch erkannt. Alle Versuchsreihen zur automatischen Ausrichtung erhielten gute Bewertungen, die teilweise den 90 prozentigen Schwellwert überstiegen, doch keine der erzielten Positionierungen stimmten mit der Ausgangslage überein. Bei dieser Baugruppe verdeckten sich die Bauteile gegenseitig, was zum Fehlen vieler geometrischer Strukturen führte. Auch war die manuelle Korrektur der Fehllage nicht so einfach wie im ersten Fallbeispiel. Während sich die Lage der Turbinenstufe noch vergleichbar schnell korrigieren ließ, war es für die einzelnen Schaufeln wesentlich schwieriger. Erstens waren diese Bauteile sehr klein und zweitens führten deren freie Anordnungen im Raum zu Problemen bei der Rotation. Die Anwendung des *ICP*-Verfahrens im Laufe der manuellen Ausrichtung war ebenfalls nicht immer hilfreich, da es stets zu einer unerwünschten Verschiebung des Referenzteils aufgrund des fehlenden Fußes der Schaufeln führte. Nach einiger Zeit gelang die manuelle Korrektur der Ausrichtung dennoch und die Baugruppe konnte rekonstruiert werden.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Werkzeugkette entworfen und umgesetzt, die der Rekonstruktion von Baugruppen dient. Dafür wird zunächst ein dreidimensionaler Scan der Baugruppe als Punktwolke übergeben und segmentiert. Anschließend werden für die erhaltenen Segmente die entsprechenden Referenzteile anhand lokaler *Features* identifiziert. Zum Schluss werden die Referenzteile vorerst automatisch ausgerichtet und bei Bedarf manuell nachjustiert. Somit kann die Rekonstruktion von Baugruppen erzielt werden. Dafür wurden die bestehenden Vorarbeiten analysiert, um die Wiederverwendung bewährter Verfahren für die Teilprozesse zu ermöglichen. Auch wurden weitere Verfahren zur Vervollständigung der Werkzeugkette betrachtet und eingesetzt. Abschließend wurden zwei verschiedene Baugruppen als Fallbeispiele, drei Referenzdatensätze sowie zehn beliebige Referenzteile genutzt, um die Güte der Teilprozesse und somit die Qualität der entstandenen Werkzeugkette zu evaluieren.

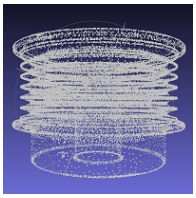
Für den ersten Teilprozess der Werkzeugkette zeigten die durchgeführten Versuchsreihen der Segmentierung der beiden Fallbeispiele, dass eine korrektere Trennung der gewünschten Bauteile möglich ist. Dabei ist die Verwendung unterschiedlicher Verfahren und mehrerer Durchläufe mit verschiedenen Parametern empfehlenswert. Es wurden das *Euclidean Cluster Extracion* und das *Region Growing* Verfahren eingesetzt. Die Wahl der richtigen Parameter ist stark vom zu segmentierenden Modell abhängig. Größe, Form und Auflösung der Eingabe sind die ausschlaggebenden Merkmale, die Einfluss auf die Segmentierung ausüben. Deswegen ließ sich aus den Versuchsreihen keine Metrik zur automatischen Bestimmung der Parameter erkennen. Dennoch konnte eine erfolgreiche Segmentierung der einzelnen Bauteile mit Tendenz zur Übersegmentierung erzielt werden. Die Übersegmentierung ließ sich schnell durch manuelles Zusammenführen der Segmente eines Bauteils korrigieren. So konnten alle gewünschten Bauteile der jeweiligen Baugruppen extrahiert und weiterverarbeitet werden. Die Identifizierung basiert auf der Detektion ähnlicher, lokaler Oberflächenregionen zwischen den Segmenten und Referenzteilen. Dabei wurden für die umliegende Nachbarschaft der ausgewählten *Keypoints* der Modelle das *SHOT-Feature* bestimmt und diese dann zur Korrespondenzbildung miteinander verglichen. Durch die Gruppierung der gefundenen Korrespondenzen mit dem *Geometric Consistency Grouping* Verfahren werden konsistente Korrespondenzen gefunden und deren Anzahl wird als Maß der Ähnlichkeit herangezogen. Beinahe alle Bauteile beider Fallbeispiele ließen sich so identifizieren. Dabei wurden identische, ähnliche aber auch falsche Referenzteile in Abhängigkeit der Vollständigkeit und Qualität der segmentierten Bauteile zurückgegeben. Es zeigte sich, dass sich die Verwendung aller gefundenen Korrespondenzen kontraproduktiv auswirkt. Bei dem Großteil der durchführten Identifizierungen wurde das korrekte Referenzteil bei der kleinsten Menge zugelassener Korrespondenzen gefunden. Durch eine etwas größere Menge an Korrespondenzen konnte die Identifizierung der restlichen Bauteile erfolgen. Lediglich ein Bauteil des ersten Fallbeispiels

konnte durch den Mangel interner Strukturen, die durch den Scan der montierten Baugruppe nicht erfasst werden konnten, nicht korrekt identifiziert werden. Das Verwenden vieler oder sogar aller Korrespondenzen förderte nicht die Bewertung des richtigen Referenzteils, sondern führte zu vielen falschen Referenzteilen mit besserer Bewertung im Vergleich zur entsprechenden Referenz. Die bereits gefundenen Korrespondenzen werden abschließend für die automatische Ausrichtung genutzt. Dabei handelt es sich um eine zweistufige Ausrichtung basierend auf dem *Sample Consensus Initial Alignment* und dem *Iterative Closest Point* Verfahren. Die Korrespondenzen wurden zur groben Ausrichtung genutzt, die anschließend durch Reduzierung der Abstände zwischen den Punkten der Modelle verfeinert wurde. Somit wurden abhängig von Anzahl und Qualität der Korrespondenzen sehr gute, aber auch sehr schlechte Ausrichtungen erzielt. Die erkannte Problematik bestand darin, dass bei wenigen Korrespondenzen ihre Lage zueinander ausschlaggebend ist. Durch verdrehte Verweise von Innen nach Außen oder von Oben nach Unten bei Bauteilen mit vielen homogenen Bereichen kann es bei der groben Ausrichtung zum Verdrehen des Bauteils kommen. Die Verfeinerung der Lage betrachtete nur nächste Punkte zwischen den Modellen und konnte solche Fehllage nicht erkennen und korrigieren. Während bei der Durchführung des ersten Fallbeispiels größtenteils gute bis sehr gute Ausrichtungen erzielt werden konnten, ließen sich die Bauteile des zweiten Fallbeispiels nicht automatisch ausrichten und mussten stets manuell korrigiert werden.

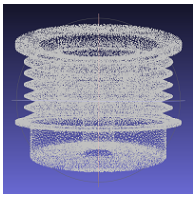
Die prototypische Werkzeugkette soll in weiteren Forschungsprojekten im Bereich des Reverse Engineerings hinsichtlich Instandhaltung und Produktlebenszyklen weiter entwickelt werden. Ziel ist es eine zuverlässige Software zu entwerfen, die es Ingenieuren ermöglichen soll, komfortabel und mit wenig manuellen Aufwand die Bauteile einer Baugruppe zu finden. Dafür ist es notwendig, dass alle Teilprozesse weiterhin optimiert und neue Aspekte untersucht werden. Die Teilprozesse könnten von der Einbindung weiterer Eigenschaften wie beispielsweise der Farbe der Oberfläche oder von der Verwendung von Polygonnetzen anstatt von Punktwolken profitieren. Andere Verfahren zur Auswahl spezieller *Keypoints* und weitere *Features* zur Merkmalsbestimmung könnten die Qualität der Korrespondenzen steigern und so die Anzahl falscher Referenzen reduzieren und so die Identifizierung sowie die automatische Ausrichtung verbessern. Mit einer stärkeren Einbindung des Benutzers könnte die Werkzeugkette als semiautomatischer Prozess eventuell vom Überblick und von den Kenntnissen des Benutzers durch verschiedene manuelle Eingaben profitieren. Das Durchlaufen der vollständigen Werkzeugkette war in einem angemessenen zeitlichen Rahmen möglich. Insofern die Eingaben aber extrem hoch aufgelöst sein sollen oder der Vergleich mit wesentlich mehr Referenzteilen durchgeführt werden soll, sollten zudem Optimierungen hinsichtlich der Laufzeit erfolgen.

7 Anhang

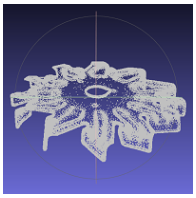
7.1 Abbildungen der Referenzteile



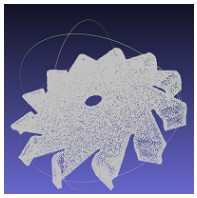
(a) Pulley



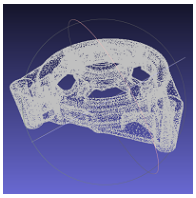
(b) Pulley CAD



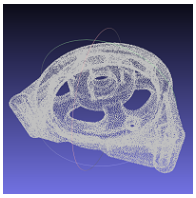
(c) Fan



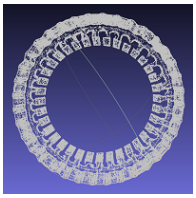
(d) Fan CAD



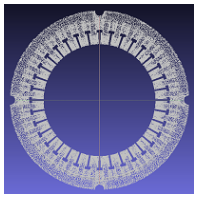
(e) Case back



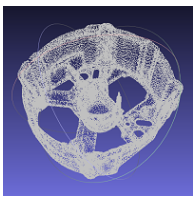
(f) Case back CAD



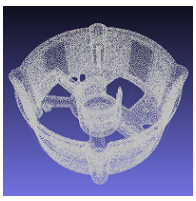
(g) Stator



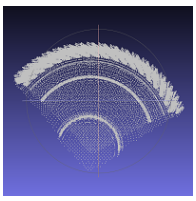
(h) Stator CAD



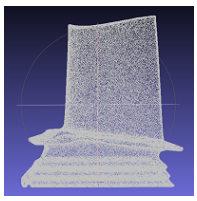
(i) Case front



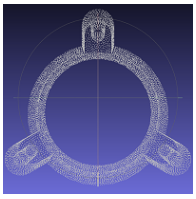
(j) Case front CAD



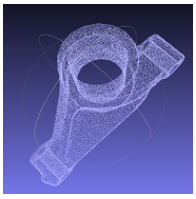
(k) Turbinen-stufe



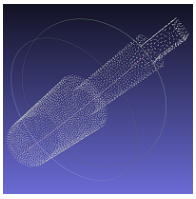
(l) Turbinen-schaufel



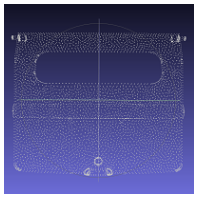
(m) Sample Part 3



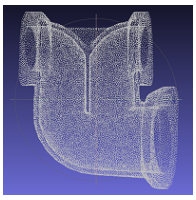
(n) Sample Part 4



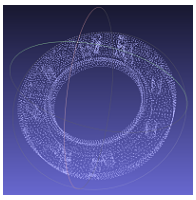
(o) Sample Part 7



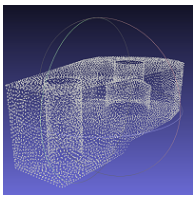
(p) Sample Part 8



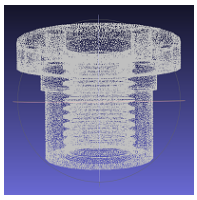
(q) Sample Part 11



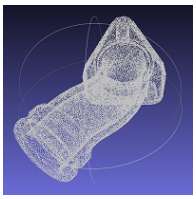
(r) Sample Part 17



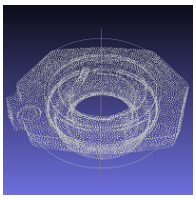
(s) Sample Part 21



(t) Sample Part 24



(u) Sample Part 39



(v) Sample Part 43

Abbildung 26: Referenzen für die Identifikation

7.2 Versuchsreihen zur Segmentierung der PKW-Lichtmaschine

Tabelle 13: Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Euclidean Cluster Extraction* Verfahren

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 665$, $t = 0.267059$	1544	35	große Mehrheit der Baugruppe verworfen
$s_{min} = 665, t = 0.3$	1607	38	große Mehrheit der Baugruppe verworfen, einige Kanten teilweise erhalten
$s_{min} = 665, t = 0.4$	1685	90	große Mehrheit der Flächen verworfen, Vielzahl der Kanten erhalten
$s_{min} = 665, t = 0.5$	1904	58	große Mehrheit der Flächen verworfen, Stator und Gehäuseteile ein Segment, Lüfterrad und Riemenscheibe stark übersegmentiert
$s_{min} = 665, t = 0.6$	2075	34	große Mehrheit ebener Flächen verworfen, Stator und Gehäuseteile ein Segment, Lüfterrad sehr gut segmentiert, Riemenscheibe stark übersegmentiert
$s_{min} = 665, t = 0.7$	2215	27	Mehrheit ebener Flächen verworfen, Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe stark übersegmentiert
$s_{min} = 665, t = 0.8$	2418	21	ebene Flächen teilweise verworfen, Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe stark übersegmentiert
$s_{min} = 665, t = 0.9$	2574	21	Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe stark übersegmentiert
$s_{min} = 665, t = 1.0$	2792	20	Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe stark übersegmentiert
$s_{min} = 665, t = 1.1$	3120	17	Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe weniger stark übersegmentiert
$s_{min} = 665, t = 1.2$	3245	11	Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe leicht übersegmentiert
$s_{min} = 665, t = 1.3$	3572	10	Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe leicht übersegmentiert
$s_{min} = 665, t = 1.4$	3884	10	Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe leicht übersegmentiert
$s_{min} = 665, t = 1.5$	4196	9	Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe leicht übersegmentiert
$s_{min} = 665, t = 1.6$	4586	9	Stator und Gehäuseteile ein Segment, Lüfterrad leicht untersegmentiert, Riemenscheibe leicht übersegmentiert
$s_{min} = 665, t = 1.7$	4867	8	Stator, Gehäuseteile und Lüfterrad ein Segment, Riemenscheibe leicht übersegmentiert

Tabelle 14: Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Region Growing* Verfahren bei unterschiedlichen Nachbarschaftsgrößen

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 665$, $d_a = 30$, $c = 0.05$, $n = 40$	9706	23	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil sowie Teile des Stators ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.05$, $n = 35$	9111	25	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil (stark untersegmentiert) jeweils mit Teile des Stators verbunden
$s_{min} = 665$, $d_a = 30$, $c = 0.05$, $n = 30$	8544	25	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil (stark untersegmentiert) jeweils mit Teile des Stators verbunden
$s_{min} = 665$, $d_a = 30$, $c = 0.05$, $n = 29$	8803	26	Riemenscheibe stark übersegmentiert, Lüfterrad etwas mehr untersegmentiert, obere und untere Gehäuseteil (stark untersegmentiert) jeweils mit Teile des Stators verbunde
$s_{min} = 665$, $d_a = 30$, $c = 0.05$, $n = 28$	8344	26	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil (stark untersegmentiert) jeweils mit Teile des Stators verbunde
$s_{min} = 665$, $d_a = 30$, $c = 0.05$, $n = 27$	8512	25	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil sowie Großteil des Stators ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.05$, $n = 26$	7998	25	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil (stark untersegmentiert) jeweils mit Teile des Stators verbunden
$s_{min} = 665$, $d_a = 30$, $c = 0.05$, $n = 25$	9026	32	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil sowie Großteil des Stators ein Segment

Tabelle 15: Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Region Growing* Verfahren bei unterschiedlichen Winkeln

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 665$, $d_a = 40$, $c = 0.05$, $n = 30$	7495	23	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil sowie Großteil des Stators ein Segment
$s_{min} = 665$, $d_a = 35$, $c = 0.05$, $n = 30$	8265	23	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil sowie Großteil des Stators ein Segment
$s_{min} = 665$, $d_a = 25$, $c = 0.05$, $n = 30$	10369	28	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil (stark untersegmentiert) jeweils mit Teile des Stators verbunden
$s_{min} = 665$, $d_a = 20$, $c = 0.05$, $n = 30$	13061	33	Riemenscheibe stark übersegmentiert, Lüfterrad und obere Gehäuseteil sehr gut segmentiert, untere Gehäuseteil stark untersegmentiert und mit Teile des Stators verbunden
$s_{min} = 665$, $d_a = 15$, $c = 0.05$, $n = 30$	16396	36	Riemenscheibe stark übersegmentiert, Lüfterrad und obere Gehäuseteil sehr gut segmentiert, untere Gehäuseteil stark untersegmentiert und mit einem Teile des Stators verbunden
$s_{min} = 665$, $d_a = 10$, $c = 0.05$, $n = 30$	26030	44	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil leicht übersegmentiert, untere Gehäuseteil leicht untersegmentiert, Stator übersegmentiert
$s_{min} = 665$, $d_a = 5$, $c = 0.05$, $n = 30$	69441	59	Riemenscheibe stark übersegmentiert und teilweise verworfen, Lüfterrad sehr gut segmentiert, obere Gehäuseteil stärker übersegmentiert und sehr kleine Flächen verworfen, untere Gehäuseteil leicht übersegmentiert, Stator übersegmentiert

Tabelle 16: Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Region Growing* Verfahren bei unterschiedlichen Krümmungsgrenzwerten

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 665$, $d_a = 30$, $c = 0.10$, $n = 30$	6263	17	Riemenscheibe stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil sowie Stator ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.15$, $n = 30$	5571	16	Riemenscheibe weniger stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil sowie Stator ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.20$, $n = 30$	5324	16	Riemenscheibe weniger stark übersegmentiert, Lüfterrad leicht untersegmentiert, obere und untere Gehäuseteil sowie Stator ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.25$, $n = 30$	5252	15	Riemenscheibe weniger stark übersegmentiert, obere und untere Gehäuseteil sowie Lüfterrad und Stator ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.30$, $n = 30$	5375	15	Riemenscheibe weniger stark übersegmentiert, obere und untere Gehäuseteil sowie Lüfterrad und Stator ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.40$, $n = 30$	5341	15	Riemenscheibe weniger stark übersegmentiert, obere und untere Gehäuseteil sowie Lüfterrad und Stator ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.50$, $n = 30$	5181	15	Riemenscheibe weniger stark übersegmentiert, obere und untere Gehäuseteil sowie Lüfterrad und Stator ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.60$, $n = 30$	5441	15	Riemenscheibe weniger stark übersegmentiert, obere und untere Gehäuseteil sowie Lüfterrad und Stator ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.75$, $n = 30$	5246	15	Riemenscheibe weniger stark übersegmentiert, obere und untere Gehäuseteil sowie Lüfterrad und Stator ein Segment
$s_{min} = 665$, $d_a = 30$, $c = 0.9$, $n = 30$	5418	15	Riemenscheibe weniger stark übersegmentiert, obere und untere Gehäuseteil sowie Lüfterrad und Stator ein Segment

Tabelle 17: Weitere Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Region Growing* Verfahren

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 665$, $d_a = 6$, $c = 0.45$, $n = 30$	41279	49	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil übersegmentiert, untere Gehäuseteil gut segmentiert, Stator übersegmentiert
$s_{min} = 665$, $d_a = 7$, $c = 0.45$, $n = 30$	29567	48	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil leicht übersegmentiert, untere Gehäuseteil gut segmentiert, Stator übersegmentiert
$s_{min} = 665$, $d_a = 7.5$, $c = 0.4$, $n = 30$	23119	44	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil leicht übersegmentiert, untere Gehäuseteil leicht untersegmentiert, Stator übersegmentiert
$s_{min} = 665$, $d_a = 7.5$, $c = 0.45$, $n = 30$	24531	44	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil leicht übersegmentiert, untere Gehäuseteil leicht untersegmentiert, Stator übersegmentiert
$s_{min} = 665$, $d_a = 7.5$, $c = 0.45$, $n = 25$	27863	53	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil etwas stärker übersegmentiert, untere Gehäuseteil etwas weniger untersegmentiert, Stator übersegmentiert
$s_{min} = 665$, $d_a = 8$, $c = 0.45$, $n = 30$	20660	42	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil leicht übersegmentiert, untere Gehäuseteil leicht untersegmentiert, Stator übersegmentiert
$s_{min} = 665$, $d_a = 8$, $c = 0.45$, $n = 25$	25553	52	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil leicht übersegmentiert, untere Gehäuseteil gut segmentiert, Stator übersegmentiert
$s_{min} = 665$, $d_a = 9$, $c = 0.45$, $n = 25$	18658	42	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil leicht übersegmentiert, untere Gehäuseteil leicht untersegmentiert und teilweise mit Stator verbunden
$s_{min} = 665$, $d_a = 9$, $c = 0.45$, $n = 30$	16578	39	Riemenscheibe stark übersegmentiert, Lüfterrad sehr gut segmentiert, obere Gehäuseteil leicht übersegmentiert, untere Gehäuseteil leicht untersegmentiert, Stator übersegmentiert

7.3 Versuchsreihen zur Segmentierung der Turbinenstufe

Tabelle 18: Resultate der Segmentierung der Turbinenstufe mit dem *Region Growing* Verfahren

Parameter	**Zeit in ms**	**Anzahl der Segmente**	**Qualität der Segmentierung**
$s_{min} = 1400$, $d_a = 8$, $c = 0.45$, $n = 25$	171439	8	vorderer Rand ein Segment, hinterer Rand zwei Segmente, Schaufel und Stufe ein Segment, einzelne Flächen ein Segment
$s_{min} = 1400$, $d_a = 5$, $c = 0.45$, $n = 25$	446735	57	Stufe stark übersegmentiert, Schaufeln teilweise zusammen und alle ohne Fuß
$s_{min} = 1400$, $d_a = 5$, $c = 0.45$, $n = 20$	539838	54	Stufe stark übersegmentiert, Schaufeln teilweise zusammen und alle ohne Fuß
$s_{min} = 1400$, $d_a = 5$, $c = 0.2$, $n = 20$	519328	54	Stufe stark übersegmentiert, Schaufeln teilweise zusammen und alle ohne Fuß
$s_{min} = 1400$, $d_a = 3$, $c = 0.45$, $n = 25$	989631	49	Stufe stark übersegmentiert, Schaufeln teilweise zusammen und alle ohne Fuß
$s_{min} = 1400$, $d_a = 3$, $c = 0.2$, $n = 20$	1072657	47	Stufe stark übersegmentiert, Schaufeln teilweise zusammen und alle ohne Fuß
$s_{min} = 1400$, $d_a = 3$, $c = 0.2$, $n = 15$	1065560	62	Stufe stark übersegmentiert, einige Schaufeln übersegmentiert, einige zusammen ein Segment
$s_{min} = 1400$, $d_a = 2,5$, $c = 0.5$, $n = 25$	1232586	53	Stufe stark übersegmentiert und teilweise verworfen, Schaufeln übersegmentiert und Fuß teilweise verworfen
$s_{min} = 1400$, $d_a = 2$, $c = 0.45$, $n = 25$	1572434	67	Stufe stark übersegmentiert und teilweise verworfen, Schaufeln übersegmentiert und Fuß fast immer verworfen
$s_{min} = 1400$, $d_a = 2$, $c = 0.2$, $n = 15$	1400706	68	Stufe stark übersegmentiert und teilweise verworfen, Schaufeln übersegmentiert und Fuß jeweils verworfen

7.4 Versuchsreihen zur Identifizierung der Segmente der PKW-Lichtmaschine

Tabelle 19: Resultate der Identifikation der Riemenscheibe bei unterschiedlicher Distanzen für zulässige Korrespondenzen

Referenz	**Distanz: 0,1**		**Distanz: 0,2**		**Distanz: 0,3**		**Distanz: 0,4**		**Distanz: 0,5**	
	Votes	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**
Pulley	0	844	0	4193	5	3501	6	2783	6	2933
Fan	0	4983	0	2316	0	4112	0	2228	4	3696
Case back	0	4729	0	4975	0	4992	0	8170	0	7063
Stator	0	6523	0	5621	0	4957	0	3958	0	3068
Case front	0	6125	0	7569	0	6688	0	7087	0	6367
Pulley CAD	0	2872	0	3502	4	3631	7	3775	7	2041
Fan CAD	0	2796	0	2648	0	2422	0	2098	0	1936
Case back CAD	0	7922	0	7445	0	7360	0	7806	0	7367
Stator CAD	0	5125	0	6063	0	6602	0	3225	0	3392
Case front CAD	0	4113	0	4992	0	7136	0	6425	0	5483
Turbinenschaufel	0	510	0	121	0	274	3	985	5	273
Turbinenstufe	0	5545	0	3669	0	3173	5	2087	5	4203
Sample Part 3	0	691	0	1367	0	377	1	584	3	782
Sample Part 4	0	3120	0	4168	0	4330	0	4722	6	2136
Sample Part 7	0	604	0	461	0	963	0	1567	0	1065
Sample Part 8	0	228	0	700	0	418	1	1280	3	300
Sample Part 11	0	8440	0	9488	0	8882	0	7941	0	9112
Sample Part 17	0	678	0	1085	0	1286	0	1752	3	767
Sample Part 21	0	2267	0	1625	0	2394	0	2021	0	992
Sample Part 24	0	7447	0	7345	0	7414	0	7619	10	6907
Sample Part 39	0	4368	0	6660	0	5547	9	3760	10	4295
Sample Part 43	0	7318	0	6505	0	7848	0	6149	0	5243
	Distanz: 0,6		**Distanz: 0,7**		**Distanz: 0,8**		**Distanz: 0,9**		**Distanz: 1,0**	
Pulley	6	3185	6	4467	6	3354	6	4371	6	2939
Fan	4	3665	6	3703	6	3438	6	1824	6	3185
Case back	0	6637	0	7650	0	6908	0	6547	0	6579
Stator	0	4996	0	4492	0	5091	0	6014	0	5309
Case front	12	7728	12	4601	12	7344	12	6572	12	6725
Pulley CAD	8	2550	8	3759	7	3827	9	1076	9	1396
Fan CAD	4	1823	5	1622	6	3885	6	3424	7	3473
Case back CAD	15	6738	14	5995	14	5030	14	7553	14	4599
Stator CAD	0	5600	9	4335	9	3241	10	6916	10	7299
Case front CAD	11	7312	12	5567	13	6813	12	5319	12	7279
Turbinenschaufel	5	537	5	679	5	116	5	152	5	407
Turbinenstufe	5	3653	5	4127	5	4092	5	3867	5	3640
Sample Part 3	7	486	10	1157	10	956	10	483	10	275
Sample Part 4	9	4324	10	5881	10	3199	11	4184	11	2584
Sample Part 7	3	182	4	1904	7	682	8	1535	9	1475
Sample Part 8	4	560	5	912	5	414	5	1537	5	1086
Sample Part 11	0	6826	23	8535	23	8984	23	8988	23	8703
Sample Part 17	4	708	5	1358	5	1094	8	1415	8	1638
Sample Part 21	7	2065	7	2083	8	1763	9	1282	10	1772
Sample Part 24	13	8214	15	7893	18	7629	18	4690	18	7220
Sample Part 39	11	5298	14	5665	14	5484	14	5042	14	5213
Sample Part 43	0	7888	18	7518	20	7358	21	7567	21	7879

Tabelle 20: Resultate der Identifikation des Lüfterrades bei unterschiedlicher Distanzen für zulässige Korrespondenzen

Referenz	**Distanz: 0,2**		**Distanz: 0,3**		**Distanz: 0,4**		**Distanz: 0,5**	
	Votes	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**
Pulley	0	2946	0	1685	0	2781	0	2547
Fan	0	3116	0	5478	4	4114	5	4488
Case back	0	8334	0	8355	0	7781	0	8590
Stator	0	4655	0	6549	0	5517	0	3791
Case front	0	6869	0	6613	0	6402	0	6611
Pulley CAD	0	3669	0	3495	5	4131	6	3174
Fan CAD	0	2859	4	1893	5	4155	5	2670
Case back CAD	0	6952	0	7596	0	5545	0	7103
Stator CAD	0	5599	0	5029	0	6172	0	6096
Case front CAD	0	6551	0	6644	0	6473	7	7679
Turbinenschaufel	1	572	3	1073	3	304	3	920
Turbinenstufe	0	3590	4	2108	4	4209	4	1748
Sample Part 3	3	421	4	1395	5	914	6	1226
Sample Part 4	0	4772	5	4006	7	4114	8	5138
Sample Part 7	0	859	1	1280	2	887	4	418
Sample Part 8	0	410	1	1004	0	822	5	352
Sample Part 11	0	10356	0	10229	0	9510	0	9342
Sample Part 17	0	770	0	840	3	651	5	220
Sample Part 21	0	2278	3	2486	5	2687	6	2444
Sample Part 24	0	7539	0	6627	0	7707	0	7433
Sample Part 39	0	6298	0	6974	0	6062	0	5558
Sample Part 43	0	7356	0	6504	0	6961	0	8206

Tabelle 21: Resultate der Identifikation des oberen Gehäuseteils bei unterschiedlicher Distanzen für zulässige Korrespondenzen

Referenz	**Distanz: 0,2**		**Distanz: 0,3**		**Distanz: 0,4**		**Distanz: 0,5**	
	Votes	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**
Pulley	0	11520	0	9821	0	12929	4	7170
Fan	0	19137	0	10520	0	12379	4	13590
Case back	24	28581	50	28681	75	25749	79	31842
Stator	0	24944	0	20101	0	18663	6	17743
Case front	0	29628	0	25463	11	28609	15	29084
Pulley CAD	0	9736	0	9080	0	10998	4	16753
Fan CAD	0	7631	0	12700	0	10128	6	11033
Case back CAD	15	27089	52	19502	69	29823	65	23856
Stator CAD	0	17375	0	17072	0	18184	9	27880
Case front CAD	0	21571	12	23018	15	15386	16	12583
Turbinenschaufel	0	411	0	1317	2	1816	3	1270
Turbinenstufe	0	12904	0	14223	0	14460	5	18339
Sample Part 3	0	1209	1	2202	4	970	4	2018
Sample Part 4	0	20401	0	17943	9	10865	9	13790
Sample Part 7	0	3008	0	3290	4	4643	6	4352
Sample Part 8	0	130	0	386	0	753	4	1378
Sample Part 11	0	35599	0	35730	0	32534	27	30687
Sample Part 17	0	1674	0	3988	0	2179	3	3564
Sample Part 21	0	6020	2	6406	4	5082	8	5387
Sample Part 24	0	25315	0	21628	0	25767	0	30322
Sample Part 39	9	16054	9	22685	15	18044	19	17283
Sample Part 43	0	30305	0	25463	0	28980	22	22918

Tabelle 22: Resultate der Identifikation des Stators bei unterschiedlicher Distanzen für zulässige Korrespondenzen

Referenz	**Distanz: 0,2**		**Distanz: 0,3**		**Distanz: 0,4**		**Distanz: 0,5**	
	Votes	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**
Pulley	0	3462	0	2359	0	3349	0	4190
Fan	0	4219	0	1576	5	4519	6	3740
Case back	0	6685	0	6699	0	5177	0	6307
Stator	0	4953	0	4969	0	5096	0	3015
Case front	0	6348	0	5516	13	6585	11	7110
Pulley CAD	0	1325	0	1812	0	2558	5	2198
Fan CAD	0	3284	0	2281	8	2581	9	1499
Case back CAD	0	6471	0	5381	0	6745	0	7268
Stator CAD	0	5361	0	5900	0	4517	0	6494
Case front CAD	0	5262	11	4439	11	6099	13	4875
Turbinenschaufel	3	1219	4	641	5	584	5	1489
Turbinenstufe	0	3300	0	2447	5	3333	5	2595
Sample Part 3	2	960	5	423	6	429	6	779
Sample Part 4	0	3631	9	2509	9	3892	10	1519
Sample Part 7	0	603	0	641	0	361	1	1508
Sample Part 8	0	1128	3	943	4	128	4	577
Sample Part 11	0	8100	0	7353	0	7547	0	7437
Sample Part 17	0	894	1	1142	3	804	8	904
Sample Part 21	0	1554	4	1685	5	2573	7	570
Sample Part 24	0	3767	0	6214	0	5558	0	5859
Sample Part 39	0	3935	0	5552	0	2649	0	3136
Sample Part 43	0	4652	0	4225	0	3206	13	4598

Tabelle 23: Resultate der Identifikation des unteren Gehäuseteils bei unterschiedlicher Distanzen für zulässige Korrespondenzen

Referenz	**Distanz: 0,2**		**Distanz: 0,3**		**Distanz: 0,4**		**Distanz: 0,5**	
	Votes	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**
Pulley	0	10814	0	5811	4	7570	4	6312
Fan	0	11513	0	11519	4	9234	5	15187
Case back	0	18919	0	21024	11	21538	13	18275
Stator	0	13850	0	12398	0	14070	6	6641
Case front	35	15856	74	20644	82	16272	86	20062
Pulley CAD	0	11854	0	10846	5	9392	6	8318
Fan CAD	0	10345	5	6894	7	10080	8	7958
Case back CAD	0	20463	0	18179	17	18618	19	16210
Stator CAD	0	13216	0	10605	0	11559	0	9217
Case front CAD	13	17449	13	18372	20	19668	27	19000
Turbinenschaufel	0	1008	1	1345	3	1227	4	1131
Turbinenstufe	0	6063	0	7848	5	10883	6	14755
Sample Part 3	0	1351	2	236	4	833	5	888
Sample Part 4	8	10792	11	8127	11	9411	12	5746
Sample Part 7	0	2513	3	3005	6	1568	8	931
Sample Part 8	0	232	0	808	4	237	6	558
Sample Part 11	0	24233	0	21588	0	23001	0	25825
Sample Part 17	0	2874	1	2724	4	4032	6	1869
Sample Part 21	0	6051	5	4220	7	4010	8	3935
Sample Part 24	0	20789	0	21787	0	21696	0	16103
Sample Part 39	0	14690	0	11158	0	13525	0	11913
Sample Part 43	0	20075	0	20478	15	18129	22	11853

7.5 Versuchsreihen zur Identifizierung der Segmente der Turbinenstufe

Tabelle 24: Resultate der Identifikation der Turbinenstufe bei unterschiedlicher Distanzen für zulässige Korrespondenzen

Referenz	**Distanz: 0,2**		**Distanz: 0,3**		**Distanz: 0,4**		**Distanz: 0,5**	
	Votes	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**
Pulley	0	1888	0	2964	5	3993	6	2121
Fan	0	3869	0	1420	0	4446	6	5803
Case back	0	7894	0	7036	0	6973	0	8860
Stator	0	5803	0	2964	0	4883	0	4851
Case front	0	8174	0	7769	0	8330	0	8049
Pulley CAD	0	2652	0	4805	4	4087	6	2199
Fan CAD	0	3604	0	4524	5	3510	7	3010
Case back CAD	0	7145	0	8689	0	7753	0	8502
Stator CAD	0	5507	0	5476	0	5491	0	6349
Case front CAD	0	6786	0	7519	0	5226	14	4726
Turbinenschaufel	0	2122	4	1498	4	1731	5	998
Turbinenstufe	0	5054	0	4555	4	2637	4	4134
Sample Part 3	1	1888	4	2527	4	1653	5	1138
Sample Part 4	0	3682	0	4680	8	2496	10	6583
Sample Part 7	0	281	0	1857	1	889	2	1107
Sample Part 8	0	203	2	2340	4	4883	5	1388
Sample Part 11	0	10577	0	10249	0	10608	0	10186
Sample Part 17	0	2012	3	1420	4	1341	7	2121
Sample Part 21	0	2340	4	702	5	1232	7	2121
Sample Part 24	0	7956	0	6053	0	6927	0	6474
Sample Part 39	0	5959	0	7051	0	6256	0	7160
Sample Part 43	0	7176	0	7395	0	7925	0	6692

Tabelle 25: Resultate der Identifikation der ersten Turbinenschaufel bei unterschiedlicher Distanzen für zulässige Korrespondenzen

Referenz	**Distanz: 0,2**		**Distanz: 0,3**		**Distanz: 0,4**		**Distanz: 0,5**	
	Votes	**Zeit in *ms***	***Votes***	**Zeit in *ms***	***Votes***	**Zeit in *ms***	***Votes***	**Zeit in *ms***
Pulley	0	156	0	687	0	156	0	686
Fan	0	718	4	686	4	765	4	1201
Case back	0	827	0	842	0	1155	0	671
Stator	0	250	0	764	0	718	0	1107
Case front	0	812	0	468	0	1170	0	639
Pulley CAD	0	827	0	842	0	811	0	1061
Fan CAD	0	281	0	125	5	234	6	920
Case back CAD	0	1170	0	983	0	1030	0	1092
Stator CAD	0	1030	0	1014	0	999	0	1046
Case front CAD	0	593	0	764	0	780	0	1014
Turbinenschaufel	1	827	2	718	3	475	3	655
Turbinenstufe	0	936	0	172	0	125	0	125
Sample Part 3	1	63	2	172	2	858	0	1092
Sample Part 4	0	936	0	874	0	172	4	686
Sample Part 7	0	110	1	562	1	889	2	1154
Sample Part 8	0	609	1	140	2	640	0	125
Sample Part 11	0	905	0	1498	0	1170	0	1326
Sample Part 17	0	1014	0	219	1	141	0	920
Sample Part 21	0	999	0	998	0	593	2	140
Sample Part 24	0	1077	0	718	0	796	0	561
Sample Part 39	0	718	0	1030	0	811	0	1107
Sample Part 43	0	921	0	515	0	1139	0	609

Tabelle 26: Resultate der Identifikation der zweiten Turbinenschaufel bei unterschiedlicher Distanzen für zulässige Korrespondenzen

Referenz	**Distanz: 0,2**		**Distanz: 0,3**		**Distanz: 0,4**		**Distanz: 0,5**	
	Votes	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**
Pulley	0	687	0	671	0	125	0	1014
Fan	0	796	0	624	0	468	0	702
Case back	0	608	0	1186	0	1326	0	764
Stator	0	842	0	858	0	187	0	1154
Case front	0	405	0	515	0	952	0	1201
Pulley CAD	0	109	0	874	0	265	4	1154
Fan CAD	0	327	0	858	0	999	0	109
Case back CAD	0	1123	0	687	0	749	0	936
Stator CAD	0	577	0	702	0	951	0	748
Case front CAD	0	858	0	889	0	889	0	1014
Turbinenschaufel	3	921	3	936	3	999	4	1154
Turbinenstufe	0	249	0	281	0	562	0	327
Sample Part 3	3	889	3	297	3	936	3	156
Sample Part 4	0	671	5	94	6	1077	6	593
Sample Part 7	0	655	1	125	1	156	2	811
Sample Part 8	0	874	2	687	2	936	2	733
Sample Part 11	0	1435	0	1233	0	967	0	1185
Sample Part 17	0	827	0	187	1	967	2	1154
Sample Part 21	0	873	2	952	2	787	3	1014
Sample Part 24	0	655	0	780	0	1092	0	795
Sample Part 39	0	530	0	671	0	484	0	358
Sample Part 43	0	795	0	889	0	858	0	967

Tabelle 27: Resultate der Identifikation der restlichen Turbinenschaufeln bei einer Distanzen von $0,3$ für zulässige Korrespondenzen

Referenz	**Turbinenschaufel 3**		**Turbinenschaufel 4**		**Turbinenschaufel 5**		**Turbinenschaufel 6**		**Turbinenschaufel 7**		**Turbinenschaufel 8**		**Turbinenschaufel 9**	
	Votes	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**	***Votes***	**Zeit in ms**
Pulley	0	499	0	686	0	234	0	640	0	905	0	749	0	483
Fan	0	250	4	827	0	530	0	125	0	796	0	296	0	1092
Case back	0	1092	0	1154	0	718	0	421	0	780	0	624	0	1123
Stator	0	1248	0	203	0	422	0	390	0	921	0	952	0	873
Case front	0	484	0	530	0	1155	0	998	0	624	0	936	0	936
Pulley CAD	0	1061	0	639	0	890	0	1092	0	640	0	827	0	671
Fan CAD	0	515	0	1014	0	125	0	1045	0	453	4	593	0	203
Case back CAD	0	1186	0	1045	0	749	0	530	0	889	0	546	0	530
Stator CAD	0	796	0	1139	0	624	0	796	0	765	0	608	0	1139
Case front CAD	0	390	0	281	0	609	0	296	0	671	0	437	0	967
Turbinenschaufel	3	531	3	998	2	765	2	983	3	475	2	718	4	218
Turbinenstufe	0	967	4	873	4	889	0	1092	0	796	0	733	0	437
Sample Part 3	3	952	4	1045	1	889	1	718	3	484	5	624	3	1139
Sample Part 4	0	1155	0	109	0	749	0	967	7	390	0	842	5	936
Sample Part 7	1	765	0	671	0	250	0	874	0	125	0	312	0	327
Sample Part 8	2	781	1	936	0	203	0	718	2	947	2	967	3	473
Sample Part 11	0	967	0	1029	0	1217	0	1186	0	1342	0	1092	0	1170
Sample Part 17	1	952	1	749	0	437	0	515	0	515	0	967	1	951
Sample Part 21	0	94	2	546	0	78	0	103	0	78	0	749	0	327
Sample Part 24	0	1201	0	1154	0	905	0	905	0	546	0	1201	0	546
Sample Part 39	0	531	0	920	0	952	0	1061	0	453	0	624	0	1108
Sample Part 43	0	1077	0	686	0	500	0	1154	0	1123	0	1076	0	1217

Literatur

[1] BITKOM, VDMA, ZVEI. Umsetzungsstrategie Industrie 4.0 - Ergebnisbericht der Plattform Industrie 4.0. https://www.bitkom.org/Bitkom/Publikationen/Umsetzungsstrategie-Industrie-40.html. [Stand: 07.03.2016].

[2] Fraunhofer IPK. Produktdigitalisierung zur Instandhaltungsplanung. https://www.ipk.fraunhofer.de/geschaeftsfelder/virtuelle-produktentstehung/informationsmaterial-publikationen-archiv/informationsmaterial/. [Stand: 07.03.2016].

[3] Rainer Stark, Hendrik Grosser, and Patrick Müller. Product analysis automation for digital MRO based on intelligent 3D data acquisition. *CIRP Annals - Manufacturing Technology*, 62(1):123 – 126, 2013.

[4] Fraunhofer IPK. Instandhaltung elektronischer Komponenten. https://www.ipk.fraunhofer.de/geschaeftsfelder/virtuelle-produktentstehung/informationsmaterial-publikationen-archiv/informationsmaterial/. [Stand: 07.03.2016].

[5] Rainer Stark, Hendrik Grosser, Boris Beckmann-Dobrev, and Simon Kind. Advanced Technologies in Life Cycle Engineering. *Procedia CIRP*, 22:3 – 14, 2014. Proceedings of the 3rd International Conference in Through-life Engineering Services.

[6] Friedrich Politz. Bauteilerkennung in 3D-Punktwolken zur Baugruppenrekonstruktion für MRO-Services. Masterarbeit, Beuth Hochschule für Technik, Berlin, Jan 2014.

[7] Josephine Mertens. Entwicklung und Implementierung eines Verfahrens zur automatisierten CAD-Baugruppenrekonstruktion. Bachelorarbeit, Freien Universität, Berlin, Apr 2015.

[8] Christian Teutsch. *Model-based Analysis and Evaluation of Point Sets from Optical 3D Laser Scanners.* Shaker Verlag GmbH, Germany, Aachen, Germany, Germany, 2007.

[9] T.M. Buzug. *Einführung in die Computertomographie: Mathematisch-physikalische Grundlagen der Bildrekonstruktion.* Springer Berlin Heidelberg, 2011.

[10] J. Hiller, C. Funk, and S. Kasperl. Vergleich von Fächerstrahl- und Kegelstrahl-Computertomographie-Messungen am Beispiel eines Zylinderkopfsegmentes. In *ZfP in Forschung, Entwicklung und Anwendung. DGZfP-Jahrestagung 2009. CD-ROM*, pages Poster 7, 6, 2009.

[11] Heckner Heiko and Marco Wirth. Vergleich von Dateiformaten für 3D-Modelle. Arbeitsbericht 7, CEDIFA, May 2014.

[12] Paul Bourke. PLY - Polygon File Format. `http://paulbourke.net/dataformats/ply/`. [Stand: 07.03.2016].

[13] Paul Bourke. STL format. `http://paulbourke.net/dataformats/stl/`. [Stand: 07.03.2016].

[14] Mohamed El-Mehalawi and R Allen Miller. A database system of mechanical components based on geometric and topological similarity. Part I: representation. *Computer-Aided Design*, 35(1):83 – 94, 2003.

[15] Mohamed El-Mehalawi and R Allen Miller. A database system of mechanical components based on geometric and topological similarity. Part II: indexing, retrieval, matching, and similarity assessment. *Computer-Aided Design*, 35(1):95 – 105, 2003.

[16] Cheuk Yiu Ip and Satyandra K. Gupta. Retrieving Matching CAD Models by Using Partial 3D Point Clouds. *Computer-Aided Design and Applications*, 4(5):629–638, 2007.

[17] Minwoo Cho Changwan Kim Changmin Kim, Joohyuk Lee. Fully Automated Registration of 3D CAD Model with Point Cloud from Construction Site. In *The 28th International Symposium on Automation and Robotics in Construction*, ISARC 2011, pages 917–922. Curran, 2011.

[18] A. Aldoma, Zoltan-Csaba Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R.B. Rusu, S. Gedikli, and M. Vincze. Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation. *Robotics Automation Magazine, IEEE*, 19(3):80–91, Sept 2012.

[19] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3D object classification. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2987–2992, Dec 2011.

[20] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique Signatures of Histograms for Local Surface Description. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6313 of *Lecture Notes in Computer Science*, pages 356–369. Springer Berlin Heidelberg, 2010.

[21] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM*, 18(9):509–517, September 1975.

[22] Lixin Li, Travis Losser, Charles Yorke, and Reinhard Piltner. Fast Inverse Distance Weighting-Based Spatiotemporal Interpolation: A Web-Based Application of Interpolating Daily Fine Particulate Matter PM2.5 in the Contiguous U.S. Using Parallel Programming and k-d Tree. *International Journal of Environmental Research and Public Health*, 11(9):9101, 2014.

[23] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for

Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 3(3):209–226, September 1977.

[24] R.B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3212–3217, May 2009.

[25] P.J. Besl and Neil D. McKay. A Method for Registration of 3-D Shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, Feb 1992.

[26] Aleksey Golovinskiy and Thomas Funkhouser. Min-Cut Based Segmentation of Point Clouds. In *IEEE Workshop on Search in 3D and Video (S3DV) at ICCV*, September 2009.

[27] Radu Bogdan Rusu. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. Dissertation, Technische Universität, München, Jul 2009.

[28] Y. Ioannou, B. Taati, R. Harrap, and M. Greenspan. Difference of Normals as a Multi-Scale Operator in Unorganized Point Clouds. *ArXiv e-prints*.

[29] T. Rabbani, F. A. van den Heuvel, and G. Vosselmann. Segmentation of point clouds using smoothness constraint. In *IEVM06*, 2006.

[30] R.B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4, May 2011.

[31] Microsoft. Implementing the Model-View-ViewModel Pattern. `https://msdn.microsoft.com/en-us/library/ff798384.aspx`. [Stand: 07.03.2016].

[32] The Qt Company. Qt. `http://www.qt.io/`. [Stand: 07.03.2016].

[33] The Qt Company. Signals & Slots. `http://doc.qt.io/qt-4.8/signalsandslots.html`. [Stand: 07.03.2016].

[34] Maya Posch. How To Really, Truly Use QThreads: The Full Explanation. `https://mayaposch.wordpress.com/2011/11/01/how-to-really-truly-use-qthreads-the-full-explanation/`. [Stand: 07.03.2016].

Codeverzeichnis

1 Einheitswürfel im PLY-Format (ASCII) nach [12] 10
2 Auszug des Einheitswürfels im STL-Format (ASCII) abgeleitet von Code 1 . 12
3 Aufbau der XML-Struktur 35
4 *Euclidean Cluster Extraction* Algorithmus 39
5 *Region Growing* Algorithmus 40
6 Vorverarbeitung für Identifikation 41
7 Vergleich von Segment und Referenzteil zur Identifikation 42
8 *SAC-IA*-Verfahren 44
9 *ICP*-Verfahren 44
10 Überprüfung der Ausrichtung 46
11 Manuelle Transformation 47
12 Verbindung von Signal und Slot zum Öffnen von Punktwolken 49
13 Erzeugung und Verbindung eines Threads nach [34] 50
14 Auszug aus entstandener XML-Datei für die PKW-Lichtmaschine 61
15 Auszug aus entstandener XML-Datei für die Turbinenstufe 70

Abbildungsverzeichnis

1 Neuer Prozess des Projektes für das Reverse Engineering [3] 2
2 Prozesskette des Reverse Engineerings 4
3 Einfache Modelle (a, b) und deren Graph (c) [15] 13
4 Testmodelle mit komplexer Struktur 14
5 3D-Scan, CAD-Modell und ihre Ausrichtung anhand der markierten Flächen [16] 15
6 CAD-Modell (a) und 3D-Scan (b) des Gebäudes [17] 16
7 Ausrichtung in der Vollansicht (a) und Draufsicht (b) [17] 16
8 *ESF Feature* und deren Funktionen [19] 20
9 Kugelförmiges Bezugssystem für das *SHOT Feature* [20] 21
10 Aufbau eines k-d-Baumes [22] 22
11 Suche des nächsten Nachbarn innerhalb des k-d-Baumes [22] 23
12 Vergleich der resultierenden Korrespondenzen 28
13 Darstellung der Hintergrundbedingung [26] 29
14 Darstellung der Normalen $\hat{n}$ eines Punktes p [28] 29
15 Weiterführender Vergleich der Korrespondenzen 32
16 Flussdiagramm der entworfenen Werkzeugkette 36
17 Darstellung des *MVVM* Musters [31] 38

18 Aufbau des *MainWidgets* 48
19 Punktwolke der gescannten PKW-Lichtmaschine 53
20 Resultierende Segmente der PKW-Lichtmaschine 56
21 Rekonstruktion der PKW-Lichtmaschine mit den Referenzteilen aus den Einzelscans 60
22 Rekonstruktion der PKW-Lichtmaschine mit den Referenzteilen aus den CAD-Modellen 61
23 Punktwolke der gescannten Turbinenstufe 64
24 Resultierende Segmente der Turbinenstufe 67
25 Rekonstruktion der Turbinenstufe mit den Referenzteilen 70
26 Referenzen für die Identifikation II

Tabellenverzeichnis

1 Gemeinsamkeiten und Unterschiede der Teilprozesse 27
2 Zusammenfassung der Segmentierungsalgorithmen 31
3 Zusammenfassung der durchgeführten Korrespondenzbestimmungen 33
4 Resultate der Segmentierung mit dem *Euclidean Cluster Extraction* Verfahren 53
5 Resultate der Segmentierung mit dem *Region Growing* Verfahren 55
6 Resultierende Segmente der Bauteile 56
7 Resultate der Ausrichtung der Einzelscan-Referenzen 59
8 Resultate der Ausrichtung der CAD-Referenzen 60
9 Ergebnisse der Segmentierung der Turbinenstufe mit dem *Euclidean Cluster Extraction* Verfahren 64
10 Ergebnisse der Segmentierung der Turbinenstufe mit dem *Region Growing* Verfahren 65
11 Resultierende Segmente der Bauteile 66
12 Resultate der Ausrichtung der Referenzen 69
13 Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Euclidean Cluster Extraction* Verfahren IV
14 Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Region Growing* Verfahren bei unterschiedlichen Nachbarschaftsgrößen V
15 Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Region Growing* Verfahren bei unterschiedlichen Winkeln VI
16 Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Region Growing* Verfahren bei unterschiedlichen Krümmungsgrenzwerten VII
17 Weitere Resultate der Segmentierung der PKW-Lichtmaschine mit dem *Region Growing* Verfahren VIII

18 Resultate der Segmentierung der Turbinenstufe mit dem *Region Growing* Verfahren X
19 Resultate der Identifikation der Riemenscheibe bei unterschiedlicher Distanzen für zulässige Korrespondenzen XII
20 Resultate der Identifikation des Lüfterrades bei unterschiedlicher Distanzen für zulässige Korrespondenzen XIII
21 Resultate der Identifikation des oberen Gehäuseteils bei unterschiedlicher Distanzen für zulässige Korrespondenzen XIV
22 Resultate der Identifikation des Stators bei unterschiedlicher Distanzen für zulässige Korrespondenzen XV
23 Resultate der Identifikation des unteren Gehäuseteils bei unterschiedlicher Distanzen für zulässige Korrespondenzen XVI
24 Resultate der Identifikation der Turbinenstufe bei unterschiedlicher Distanzen für zulässige Korrespondenzen XVIII
25 Resultate der Identifikation der ersten Turbinenschaufel bei unterschiedlicher Distanzen für zulässige Korrespondenzen XIX
26 Resultate der Identifikation der zweiten Turbinenschaufel bei unterschiedlicher Distanzen für zulässige Korrespondenzen XX
27 Resultate der Identifikation der restlichen Turbinenschaufen bei einer Distanzen von $0,3$ für zulässige Korrespondenzen XXI