

# 1 Verfahren

## 1.1 Datenerfassung

Ziel ist ein digitales Abbild von einem realen Objekt zu erstellen. Als Grundlage für dieses Abbild arbeite ich in diesem Fall mit Pointclouds die mithilfe eines Laserscanners aufgenommen wurden. Auch andere Möglichkeiten Pointclouds zu erhalten sind denkbar und sollten mit dem Verfahren kompatibel sein.

## 1.2 Scanner

Der von mir eingesetzte Scanner kann in einer Linie mit einer maximalen Länge die Abstände zu einem Objekt messen. Um das ganze Objekt zu erfassen muss der Scanner also linear über das Objekt bewegt werden. Hierfür wurde eine CNC-Fräse benutzt, beziehungsweise die Verschiebungseigenschaften von einer. Hier ist es wichtig die Vorlaufgeschwindigkeit der Fräse in Abhängigkeit von der Frequenz des Scanners zu wählen. Ist die Geschwindigkeit zu gering nimmt der Scanner mehrere Aufnahmen von der selben Linie auf, ist sie zu hoch werden Koordinaten auf dem Objekt übersprungen und die Auflösung der resultierenden Pointcloud sinkt.

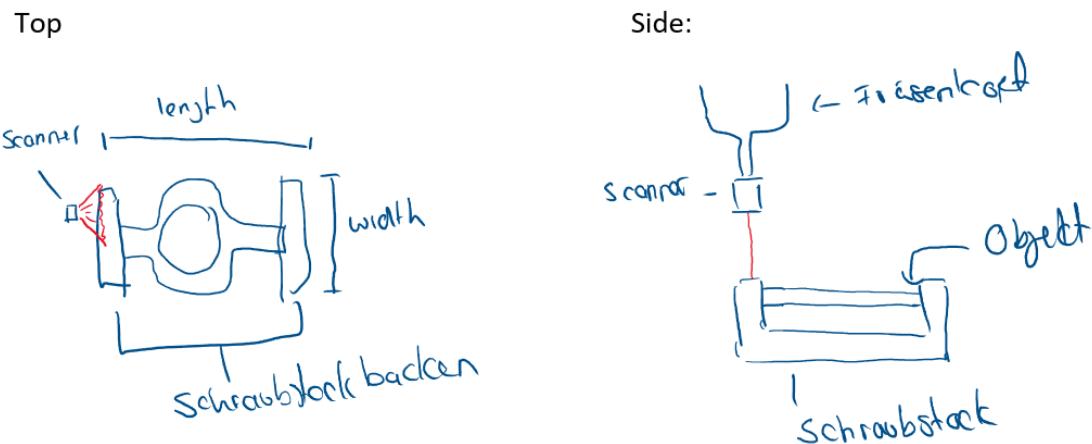


Abbildung 1: Versuchsaufbau

TODO: Berechnung für die Geschwindigkeit einfügen

In 1 ist der Versuchsaufbau skizziert. Der Fräsenkopf wird in Richtung der Länge und Breite des Objekts verschoben. Die gesamte Länge des Objekts kann also erfasst werden in dem Scanner beziehungsweise Fräsenkopf in X-Richtung bewegt wird. In dieser Bewegung ist die Limitierung einmal die maximale Länge des Fräsenachse, und außerdem der verfügbare Speicher auf dem Computer der die Pointcloud aufnimmt. In Y-Richtung sind wir durch die maximale Scannerbreite begrenzt. Diese ist, je nach Scanner, relativ klein. Um auch Objekte Scannen zu können die breiter sind als die Scannerbreite werden mehrere Scans erstellt, zwischen denen der Scanner immer in Y-Richtung verschoben wird. Die Länge der Verschiebung sollte kleiner als die Scannerbreite sein, damit eine Überlappung entsteht die genutzt werden kann um die Pointclouds wieder zusammenzufügen. So können Pointclouds aufgenommen werden die dann im 2. Schritt wieder zu einem digitalen Abbild zusammengefügt werden.

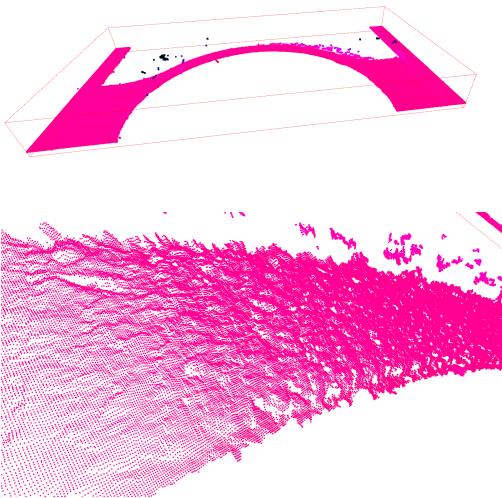


Abbildung 2: Pointcloud Komplett

Bevor ich mich aber mit dem 2. Schritt des Verfahrens beschäftige möchte ich auf die Datenstruktur von Pointclouds eingehen. Wie der Name schon sagt bestehende Pointclouds aus Punkten. In unserem Fall ca. 3 Millionen für eine einzelne Bewegung in X-Richtung. In jedem Punkt sind mindestens 3 Koordinaten gespeichert, Eine X und Y Koordinate relativ zu Ursprung der Pointcloud und eine Z Koordinate in der die Höhe gespeichert wird. Der Ursprung der Pointcloud liegt in der Mitte der X-Achse, also genau da wo der Scanner verläuft. Zusätzlich ist für jeden Punkt auch noch eine Farbe als RGB-Farbwert (3 Werte von 0 bis 255) gespeichert. Die Farbe wird relativ zu der Höhe berechnet. Punkte bekommen einen Farbwert zugewiesen je nachdem wie nah sie an dem Minimum oder Maximum der Z-Koordinate sind. In 2 ist die komplette Pointcloud eines Demonstratorbauteils zu sehen, zusammen mit einer BoundingBox, durch diese werden die Randpunkte der Pointcloud visuell dargestellt. Man sieht außerdem das unten liegende Punkte hell und weiter oben liegende Punkte dunkler dargestellt werden. Unterhalb ist eine Nahaufnahme des Mit-

telstück des gleichen Bauteils zu sehen. Hier sind die einzelnen Punkte sichtbar.

Die Koordinaten sind als Float gespeichert haben keinerlei bezug zueinander. Das heißt die X und Y Koordinaten sind nicht zum Beispiel anhand eines Grids angelegt, sondern können einen beliebigen Abstand zwischeneinander haben. Das werden wir auch später in der weiteren Bearbeitung der Pointclouds sehen.

Wenn nun also alle Pointclouds für ein Objekt erfasst wurden kann zu dem 2. Schritt des Verfahrens übergegangen werden. Hier müssen die Daten zusammengefügt werden. Hierfür existieren in der Literatur schon verschiedene Verfahren, ein besonders beliebtes ist der ICP-Algorithmus. Dieser Algorithmus existiert schon seit dem Beginn der 90ziger Jahre und ist der klassische Methode wenn es um die Registrierung von Pointclouds und anderen Punkt-Sets geht. [?, ] Der Algorithmus errechnet eine lokale, optimale Transformation die ein Datenset dem anderen annähern kann. [?] Um diese Transformation zu bestimmen werden zuerst die Distanzen von allen Punkten in Datenset A zu dem jeweils nähsten Punkt in Datenset B aufsummiert werden. Dann wird eins der Datensets verschoben und rotiert und wieder die Distanzen gebildet. Dies wird solange gemacht bis die Änderung der Distanzen konvergiert. Die entstehende Transformation ist dann optimal. Für identische Datensets die sich nur in einer Transformation und Rotation unterscheiden, funktioniert dieser Algorithmus sehr gut. Bei Datensätzen die Messfehler oder Überlappungen beeinhalten kann häufig keine Optimale Transformation bestimmt werden. Deswegen wurden seit der ersten Vorstellung des Algorithmus viele Varianten entwickelt die mit diesem Schwächen umgehen. Zum Beispiel der 'Sparse Iterative Closest Point' Algorithmus von [?] oder die 'Anderson-accelerated' Version die besser mit Außreißern und nur partiell überlappenden Daten umgehen kann und eine gleichwertige oder bessere Transformation errechnen kann. [?]

Aufgrund der Versprechen dieses Varianten und weil dieser Algorithmus in vielen Open-Source Bibliotheken schon implementiert ist war das auch mein erster Ansatz und ich habe mit Hoffnungen gemacht das, das zusammenfügen der Pointclouds damit schnell abgehakt ist. Dafür habe ich die Methode der Global Registrierung aus der Open3D Bibliothek genutzt die auf dem Paper von Qian-Yi Zhou, Jaesik Park und Vladlen Koltun basiert. [?] Das Globale Registrierung Verfahren wird vor dem ICP Algorithmus benutzt um eine erste, initiale Transformation zu erstellen, die der ICP Algorithmus im ersten Schritt braucht.

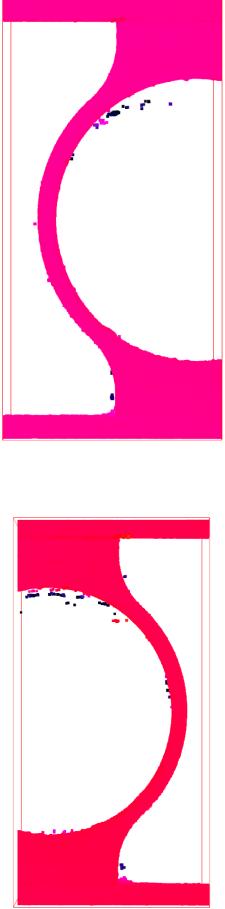


Abbildung 3: Demonstratorbauteil

Das Problem bei dieser Methodik und meinem Datensatz ist, das bei den hälften des Demonstratorbauteils symmetrisch sind. In dem Verfahren wird nicht nur eine Transformation anhand der X,Y und Z Achsen eingesetzt sondern auch eine Rotation der Pointcloud. Wie man in 3 sehen kann überlappen die beiden Pointclouds schon sehr gut wenn ein Datensatz um 180 Grad gedreht wird. Auf dieses Ergebnis ist auch das Global Registrierungsverfahren gekommen. Für unseren Fall ist also ein Verfahren nötig was ausschließlich eine Transformation in X und Y Richtung benutzt. Das sind auch die einzigen Achsen in der sich der Scanner bewegt hat.

### **Wahl des Demonstratorbauteils**

Man könnte sich jetzt hier die Frage stellen warum das Demonstratorbauteil dann so gewählt und entworfen wurde das es symmetrisch ist. In dem ersten Schritt des zusammenfügen ist die Bauteilgeometrie zwar hinderlich, dafür profitiert die Spannkraftdeformationserkennung von dem initial runden innenkreis des Bauteils. Zusätzlich soll das hier zu entwickelnde Verfahren auf viele Bauteilgeometrien anwendbar sein und somit nicht durch eine Symmetrie beschränkt werden.

### **Weitere Verfahren**

Nachdem die Globale Registrierung und der ICP-Algorithmus auf den Pointclouds nicht anwendbar waren musste ein anderes Verfahren her. Aufgrund weniger schon existieren Verfahren die eine reine Bewegung auf nur 2 Achsen verwenden habe ich eine Idee entworfen, die auf dem ICP-Algorithmus basiert, um 2 Pointclouds zu vergleichen und zu verschieben. Hierfür habe ich die Pointclouds in ein Zweidimensionales Array mit den Z-Werten als Inhalt konvertiert. Nun kann die euklidische Distanz der Z-Werte und jedem X und Y Wert aufsummiert werden. In einem Naiven Ansatz kann dann eine Pointcloud entlang der Länge und Breite der anderen Pointcloud bewegt werden und für jede Transformation die Summe der Distanzen gespeichert werden. Die Idee war das die Transformation mit der kleinsten Summe die beste Überlappungen der beiden Pointclouds ist. Das hat aber nicht funktioniert. Durch die tatsächliche relativ kleine Überlappung ist die Summe der Distanzen nicht bei der korrekten Transformation minimal sondern wenn die Pointclouds maximal übereinander geschoben sind. Zusätzlich war die Verschiebung und Berechnung der Zweidimensionalen Arrays rechenintensiv und hat zu langen Laufzeiten geführt. Also habe ich dieses Verfahren wieder aufgeben mit einer neuen Idee:

### **Pointcloud in Bild konvertieren**

Um Rechenzeit zu sparen und auf viele Funktionen von schon bestehenden Bilderkennungsbibliotheken zurückgreifen zu können habe ich die Pointclouds in ein Bild konvertiert. Hierfür wird zuerst in leeres Bild mit den gleichen Maßen einer Pointcloud erstellt. Dann wird über alle Punkte der Pointcloud iteriert und jeweils der Pixel an der X und Y Koordinate des Punktes auf einen Helligkeitswert gesetzt. Um Rechenzeit und Speicherkapazitäten zu schonen, und weil es für die Berechnung ausreichend ist, habe ich mich für 8 Bit single-channel Bilder die nur Helligkeitswerte abbilden entschieden. Hier kann also jede Pixel einen Wert zwischen 0 und 255 annehmen. Der entsprechende Wert kann wie folgt berechnet werden:

Der resultierende Wert ist die Helligkeit die dem Pixel zugewiesen wird.  $Z$  ist die Z-Koordinate des Punktes in der Pointcloud.  $min_y$  und  $max_y$  sind die Grenzen der Z-Koordinate, diese werden gebraucht um die Helligkeit relativ zu der Höhe zu berechnen.  $min_b$  und  $max_b$  sind die gewünschten Grenzen der Helligkeit. In unserem Fall sind  $min_b = 0$  und  $max_b = 255$  da ein 8 Bit Bild verwendet wird.

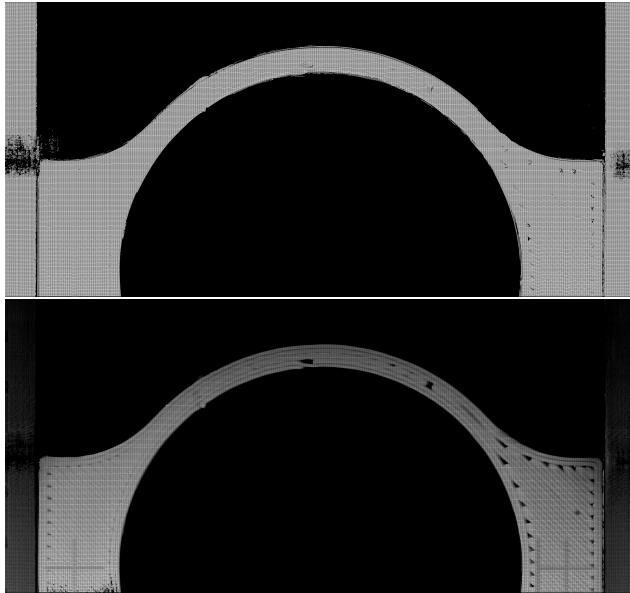


Abbildung 5: Konvertierung Pointcloud zu Bild

den n-ten Prozentsatz können Außreißer entfernt werden. Wenn nur die häufigsten 10 Prozent übernommen werden erhält man untere Bild in Abbildung 1.2

Features auf dem Bauteil können jetzt deutlich besser erkannt werden. Auch zu sehen sind jetzt die Markierungen auf der linken und Rechten Seite die bei der Registrierung helfen sollen. Auch schön zu sehen sind die Spuren und Lücken die durch den FDM Herstellungsprozess entstehen.

Durch das Filtern der Höheninformationen sind Oberflächenstrukturen nicht nur besser erkennbar, auch die Ränder treten genauer hervor. Das ist sehr wichtig für das korrekte zusammenfügen.

Doch wo soll die Grenze gezogen werden um die Oberfläche möglichst genau zu erkennen, aber nicht zu viele Höheninformation zu verlieren.

$$value_p = \frac{Z - min_y}{max_y - min_y} \cdot (max_b - min_b) + min_b$$

Abbildung 4: Berechnung Pixelwert

Wie man in Abbildung 1.2 sehen kann, sind kaum Helligkeitsveränderungen im Bild sichtbar. Das liegt an der selben Problematik an der der ICP-Algorithmus häufig scheitert. Reale Datensets wie wir es vorliegen haben sind nicht perfekt sondern beeinhalten Messfehler und Streuungen.

Abbildung 1.3 zeigt die Häufigkeit der gleichen Höhenwerte einer Pointcloud von dem Demonstratorbauteil als FDM-Druck. Auf der Y-Achse die relative Häufigkeit. Die meisten Höhenwerte sind über 80, sie gehören zu den Punkten die auf dem Demonstratorbauteil liegen, es treten allerdings auch Werte darunter auf. Die in 1.2 vorgestellte Formel benutzt allerdings die absoluten Minimum und Maximum Werte, die Punkte auf dem Bauteil werden also entsprechend wenig berücksichtigt. Dem kann Abhilfe geschaffen werden indem Werte die weniger häufig auftreten entfernt werden. Sortiert man alle Höhenwerte nach der Häufigkeit ihres auftretens in der Pointcloud und entfernt

### 1.3 Pointcloud filtern

Ziel ist das das entwickelte Verfahren bei allen gängigen Additiven Fertigungsmethoden angewendet werden kann, das muss bei der Filterung der Daten berücksichtigt werden da die Datenverteilung von FDM Bauteilen anders aussieht als bei Metallenen Werkstoffen. Wie man in Abbildung 1.3 sehen kann streuen nicht alle Pointclouds gleich, abhängig von dem Werkstoff des Bauteils werden die Laserstrahlen unterschiedlich reflektiert und mehr oder weniger Ausreißer sind zu sehen. In dem oberen Histogramm sind die Häufigkeiten der Höhenwerte zu sehen, alle Punkte sind in 500 Teile gruppiert. Unterhalb ist das Histogramm mit dem gleichen Datensatz, aber mit der Y-Achse logarithmus skaliert um kleine Prozente deutlich zu machen die im oberen Diagramm nur schwer oder gar nicht sichtbar sind. Man sieht das Metallteil deutlich mehr in beide Richtungen streut, während das FDM gedruckte Bau teil weniger nach oben, aber mehr nach unten streut. Es muss also eine Filtermethode gewählt werden die für alle Fertigungsverfahren anwendbar ist und nicht bei einer Methode besser funktioniert wie bei einer anderen. Werden zum Beispiel die 10 Prozent häufigst auftretenden Höhenwerte bei einem Metallteil benutzt kommt folgendes Bild heraus.

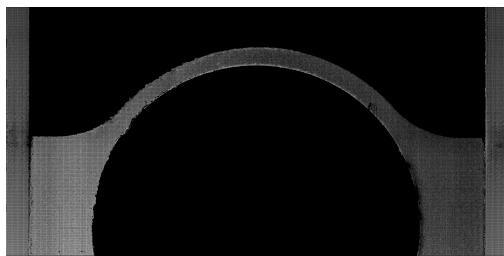


Abbildung 7: Metallteil gefiltert

Man sieht vor allem auf der rechten Seite das Ränder nicht mehr klar erkennbar sind, da sie durch die Filterung lücken aufweisen. Praxistest haben gezeigt das ein ausreichend gut funktionierender Filterwert 50 Prozent ist. Damit werden genug Messfehler aus dem Bild genommen aber trotzdem bleiben Oberflächenfeatures und Ränder sichtbar genug um ein korrektes Zusammenfügen zu gewährleisten. Dieses Filtern bezieht sich aber nur auf 2 Dimensionale Bildinformationen. Um bei dem Konvertieren noch weniger Punkte die nicht auf dem Bauteil liegen nicht in das Bild zu übernehmen kann auch noch die Pointcloud gefiltert werden. Hier kann ein einzelner Punkt relativ zu seinem Nachbarn im 3 Dimensionalen Raum betrachtet werden um so Außreißer zu erkennen. Dafür sind in der Open-Source Bibliothek 'Open3D' 2 Methoden vorhanden: Radius basiert oder auf Basis von statistischen Werten, erste Methode eignet sich gut wenn die Maße des Objekts bekannt sind. Hier wird um jeden Punkt eine Sphere gebildet und die Punkte die weniger als einen konfigurierbare Menge an Punkten in ihrer Sphere haben werden entfernt. Da das hier zu entwickelnde Verfahren sich nicht auf eine Bauteilgeometrie beschränken ist dieses Verfahren nicht geeignet. Stattdessen wird das andere benutzt. Hier werden die Punkte entfernt die weiter von ihren benachbarten Punkten entfernt sind als der durchschnittliche Abstand der Punkte in der gesamten Pointcloud. Hier kann die Menge der benachbarten Punkte die betrachtet werden sollen und ein Limit für den Abstand von der Standardabweichung. Umso mehr benachbarte Punk-

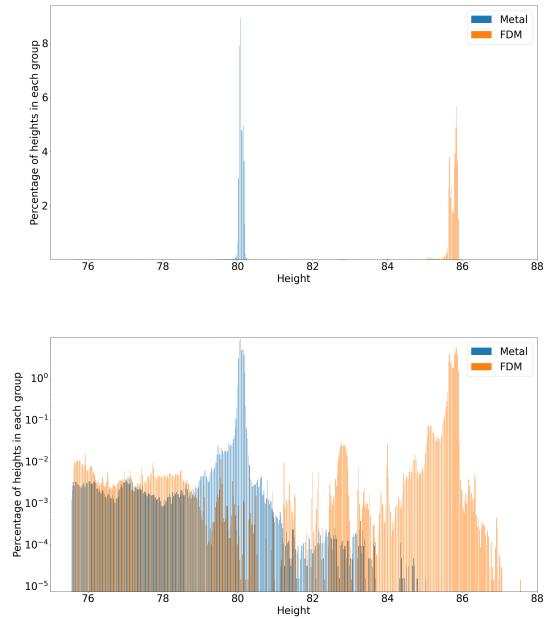


Abbildung 6: Auftreten Höhe

te betrachtet werden, umso mehr Zeit braucht die Filterung, aber die Filterung wird auch akkurate. Im Praxistest haben sich hier 50 Nachbarpunkte bewährt. Mit diesem Wert werden bei Pointclouds in unserem Datensatz jeweils ca. 2 Prozent der Punkte entfernt. So kann das resultierende Bild gut genug umgewandelt werden um eine erfolgreiche Zusammenführung von verschiedenen Bildern zu gewährleisten. Ein Nachteil bei der Filterung in Abbildung 1.2 links und rechts mittig zu sehen. Hier sind schwarze Punkte sichtbar. Diese treten auf weil der Scanner hier über dem Bau teil Punkte erkannt hat. Durch das Filtern wurden diese Punkte entfernt beziehungsweise bei der Konvertierung nicht berücksichtigt. Da diese Punkte dann fehlen bleiben sie im resultierenden Bild schwarz. Das ist zwar etwas unschön anzuschauen, beeinträchtigt das zusammenfügen aber nicht weiter.

## **Stitching**