1 Einleitung

1.1 Motivation und Aufgabenstellung

Mit der rasanten Entwicklung der optischen Messtechnik werden dreidimensionale (3D)-Modelle von realen Objekten mit unterschiedlichen Ansichten zur Datenerfassung für viele Anwendungsszenarien eingesetzt [1, 11]. Als grundlegende Technik in der 3D-Datenverarbeitung ist die Registrierungsmethode hauptsächlich für die Bestimmung der Korrespondenz und Transformation von 3D-Daten mit mehreren Ansichten verantwortlich, was die Implementierung vieler Reverse-Engineering-Anwendungen erleichtert, wie z. B. die Schädelregistrierung für die Rekonstruktion im medizinischen Bereich [30], die quellen-übergreifende Punktwolkenregistrierung in großen gescannten Daten für Straßenansichten im Freien [12] und die 3D-Gesichtsregistrierung für die Gesichtserkennung [31] usw. Daher ist die Entwicklung effizienter Registrierungsmethoden zu einem wichtigeren Forschungsthema geworden. Als der bekannteste Registrierungsalgorithmus führt der Iterative Closest Point (ICP)-Algorithmus einen iterativen Prozess durch, um die relativ ähnlichen 3D-Punktwolken unter den Randbedingungen einer gleichmäßigen Dichte, einer hohen Überlappungsrate und einer guten Ausgangsposition gut auszurichten, und neigt dazu, in lokale Optimalitätsprobleme zu geraten [4, 6].

Ziel dieser Arbeit ist, ein ICP-Algorithmus weiter zu entwickeln und optimieren, sodass das Algorithmus verschiedene 2D- und 3D-Modelle mit unterschiedlichen Abständen, Verschiebungen und Rotationen zwischen einander registrieren, einlesen, in einer 3-dimensionalen Umgebung visualisieren und effizient bearbeiten kann. Dazu soll eine Statistik durchgeführt werden, die diesen Algorithmus ergebnisorientiert und in Abhängigkeit von beeinflussenden Parameter bewerten kann.

1.2 Aufbau dieser Arbeit

Die Arbeit ist folgendermaßen aufgebaut:

Kapitel 2: Dieses Kapitel beginnt mit der Definition von Punktwolken und Registrierung. Anschließend wird der ICP-Algorithmus vorgestellt und erläuter. Schließlich wird auf die ICP-Varianten kurz eingegangen.

Kapitel 3: Dieses Kapitel befasst sich hauptsächlich mit der Implementierung des optimierten ICP-Algorithmus. Zunächst werden die verwendete Tools und Bibliotheken kurz vorgestellt, mit denen dieses Projekt umgesetzt wird. Dann folgt die Beschreibung der wichtigsten Funktionen des ICP-Algorithmus. Schließlich wird auf die optimierte Variante eingegangen und erklärt.

Kapitel 4: Die Ergebnisse werden in diesem Kapitel vorgestellt und ausgewertet.

Kapitel 5: In diesem Kapitel wird noch einmal auf alle Kapitel der Arbeit zurückgeblickt und ein Fazit gezogen.

2 Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen dieser Arbeit vorgestellt. Zunächst gibt es Informationen über Punktwolken und wie sie verarbeitet werden können. . Anschließend wird auf der sogenannten Registrierung kurz eingegangen. Schließlich wird der Iterative Closest Point (ICP)-Algorithmus vorgestellt.

2.1 Punktwolken

Eine Punktwolke ist eine digitale 3D-Darstellung eines physischen Objekts oder Raums. Sie besteht aus Millionen von einzelnen Messpunkten, von denen jeder eine x-, y- und z-Koordinate hat. Die Abbildung 1 zeigt Punktwolken von zwei verschiedenen Objekte.

Je nach der zur Erfassung der Wolke verwendeten Methode - und den beteiligten Sensoren - kann jeder Punkt auch RGB-Farbdaten oder sogar Intensitätsinformationen enthalten, die die Rückstrahlstärke des Laserpulses widerspiegeln, der den Punkt erzeugt hat.

Punktwolken, die von einem Sensor im realen Leben erfasst werden, enthalten immer Rauschen und eine große Menge an Daten enthalten, die mit Downsampling- und Algorithmen zur Entfernung von Ausreißern behandelt werden können. Einige Punktwolkenabgleichs-Methoden, wie z. B. der ICP-Algorithmus (Abschnitt 2.3), erfordern Informationen über die Oberfläche, die die Punktwolke repräsentiert. Diese Informationen können durch die Schätzung von Oberflächennormalen gewonnen werden.

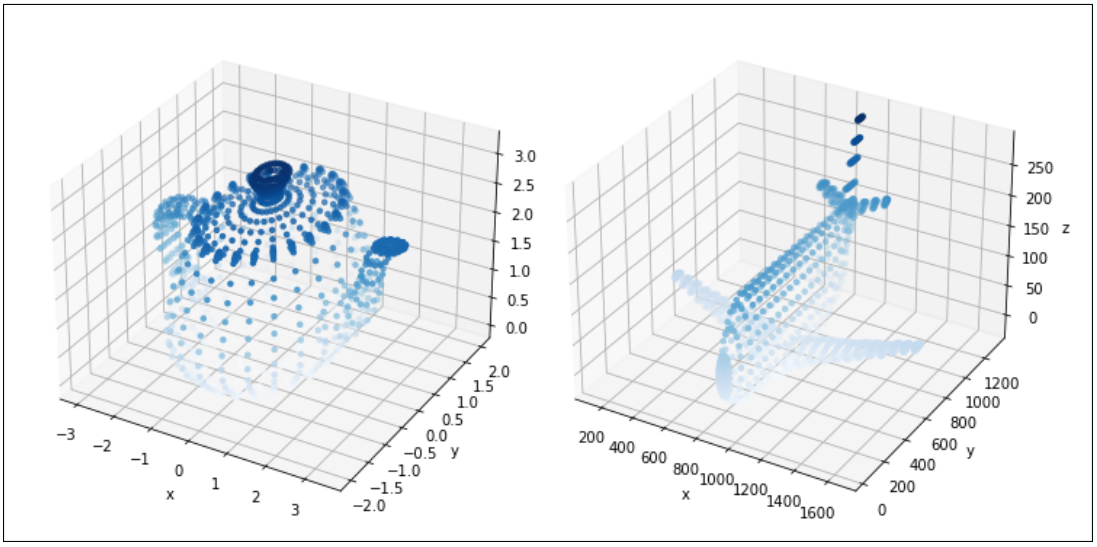

Abbildung 1: 3D-Darstellung von zwei Punktwolken .

2.2 Registrierung

Die Aufgabe, mehrere Teilpunktwolken eines größeren Modells auszurichten, wird gewöhnlich als Registrierung bezeichnet. Andere verwandte Schlüsselwörter sind Posenschätzung, Oberflächenabgleich, Ausrichtung und Bewegungsabschätzung.

Die Registrierung von Oberflächennetzen ist eine grundlegende Aufgabe mit zahlreichen Anwendungen in verschiedenen Bereichen wie Computer Vision und medizinische Bildgebung. Die allgemeine Registrierung ist definiert durch die Berechnung einer Transformation zum Ausrichten von zwei oder mehr Oberflächen.

Die Registrierung von Punktwolken, d.h. die Aufgabe, die Starrkörpertransformation zwischen zwei Punktwolken zu finden, ist ein integraler Bestandteil der geometrischen Inferenz in vielen modernen Wahrnehmungssystemen. Eine umfassende Überblick starrer und nicht starrer Registrierung wird in [24, 5] gegeben.

2.3 Iterative Closest Point Algorithmus

Der Iterative Closest Point (ICP)-Algorithmus [4, 6] ist eine gängige Registrierungsmethode, welche darauf abzielt, die Transformation zwischen einer Punktwolke und einer Referenzfläche (oder einer anderen Punktwolke) zu finden, indem die quadratischen Fehler zwischen den entsprechenden Einheiten minimiert werden. Der iterative Charakter von ICP ergibt sich aus der Tatsache, dass die Korrespondenzen neu betrachtet werden, wenn sich die Lösung dem lokalen Fehlerminimum nähert. Wie jede Methode des Gradientenabstiegs ist der ICP anwendbar, wenn wir im Voraus einen relativ guten Startpunkt haben. D.h. wenn die Ausgangspositionen nahe genug sind. Andernfalls wird es im ersten lokalen Minimum gefangen und die Lösung ist unbrauchbar.

Der ICP-Algorithmus lässt sich wie folgt beschrieben werden. Gegeben sei eine Punktwolke $Q = \{q_j, q_j \in \mathbb{R}^3,\ j = 1,\ldots,N_Q\}$, die als Source-Modell genannt ist. Für eine Punktwolke $P = \{p_i, p_i \in \mathbb{R}^3,\ i = 1,\ldots,N_P\}$, als Destination-Modell genannt, ist eine Transformation bestehend aus einer Rotation R und einer Translation t zu finden, die auf P angewendet wird, um Q so gut wie möglich anzupassen, wie die Abbildung 2 veranschaulicht. Mathematisch gesehen wird eine Rotation durch eine Matrix (Rotationsmatrix genannt) beschrieben, während eine Translation durch einen Vektor (Translationsvektor genannt) definiert ist.

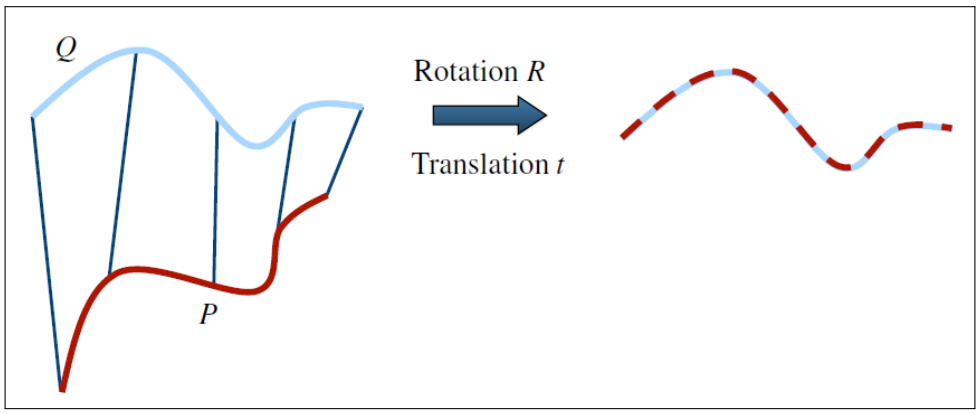

Abbildung 2: Idee hinter dem iterativen Closest Point-Algorithmus in $\mathbb{R}^2$.

In dieser Arbeit stellt das Source-Modell bzw. Destination-Modell ein 3D-Polygon-Modell dar. Eine graphische Darstellung der verwendeten Modelle ist im Abschnitt **??** gegeben.
Dabei muss das Minimum der Fehlerfunktion E in Abhängigkeit von R und t

$$E(R,t) = \sum_{i=1}^{N_Q} \sum_{j=1}^{N_P} ||q_i - (Rp_j + \mathbf{t})||^2 \tag{1}$$

berechnet werden, wobei $||\cdot||$ die euklidische Norm (auch euklidische Metrik genannt) darstellt. Es können jedoch auch andere Normen verwendet werden [4, 15].
Um die Rotation R und die Translation t zu finden, durchläuft der ICP-Algorithmus folgende Iterationsschritte:

1. Zentrieren der Punktwolken P und Q durch Subtraktion des Mittelwerts: zuerst werden die Mittelwerte von P und Q berechnet:

$$\bar{p} = \frac{1}{N_P} \sum_{i=1}^{N_P} p_i$$
$$\bar{q} = \frac{1}{N_Q} \sum_{j=1}^{N_Q} q_j$$

   Dann werden die zentrierte Punktwolken P' und Q' bestimmt, wobei

$$P' = \{p_i - \bar{p},\ i = 1,\ldots,N_P\}$$
$$Q' = \{q_j - \bar{q},\ j = 1,\ldots,N_Q\}$$

2. Ermittlung der Korrespondenzmenge $C = \{(p'_i, q'_j)\}$ aus P' und Q', wie die Abbildung 3 darstellt. Dabei werden die entsprechende Punkte aus Q' für jeden Punkt p'_i aus P' bestimmt. Die geschieht mittels Berechnung der euklidischen Norm $||p'_i - q'_j||$ für jedes i mit $j = 1,\ldots,N_Q$. Falls für ein i_0 ein j_0 existiert mit

$$||p'_{i_0} - q'_{j_0}|| = \min_{j=1,\ldots,N_Q} ||p'_{i_0} - q'_j||$$

Dann bildet das Paar (p'_{i_0}, q'_{j_0}) einen Punkt aus der Korrespondenzmenge C. Damit haben die entsprechende Punkte präsentieren die gleichen Teile von jeder Punktwolke, den am Ende der Bearbeitung aufeinander liegen müssen. Bezeichne mit $|C|$ die Kardinalität (oder auch Mächtigkeit) der Menge C und setze $|C| = N$. Danach ordnen wir den Punktwolken neu anhand der Korrespondenzen (p'_i, q'_j) mit Hilfe eines Indexes n. Damit erhalten wir

$$C = \{(p'_n, q'_n),\ n = 1, \ldots, N\}. \tag{2}$$

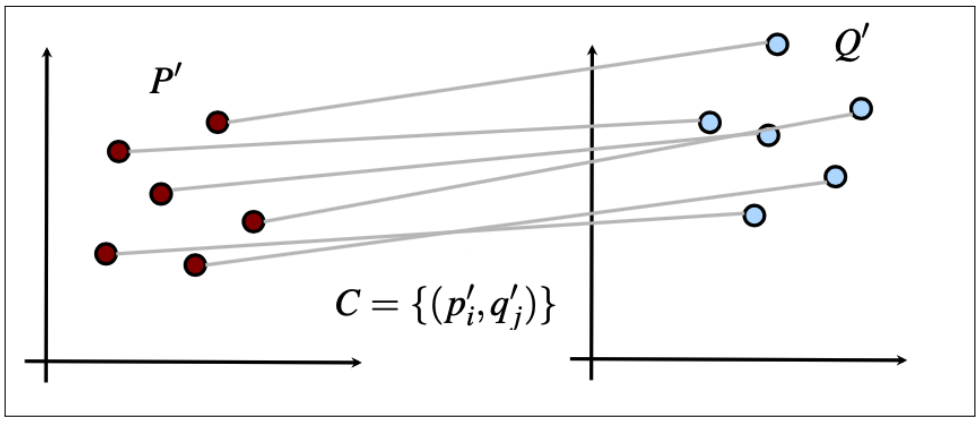

Abbildung 3: Korrespondenzmenge $C = \{(p'_i, q'_j)\}$ aus P' und Q'.

3. Erstellung der Kreuzkovarianzmatrix: Aus der Korrespondenzmenge C in Gleichung (2) wird die Kreuzkovarianzmatrix K wie folgt berechnet:

$$K = \frac{1}{N} \sum_{i=1}^{N} q'_i p'^T_i,$$

wobei x^T der transponierte Vektor von x darstellt. Jeder Punkt (p'_i, q'_j) hat drei Dimensionen, d.h. Somit hat die Kreuzkovarianz die Form von (der Einfachheit halber lassen wir die Indizes weg):

$$K = \begin{bmatrix} cov(p'_x, q'_x) & cov(p'_x, q'_y) & cov(p'_x, q'_z) \\ cov(p'_y, q'_x) & cov(p'_y, q'_y) & cov(p'_y, q'_z) \\ cov(p'_z, q'_x) & cov(p'_z, q'_y) & cov(p'_z, q'_z) \end{bmatrix}$$

Intuitiv sagt uns die Kreuzkovarianz, wie sich die Koordinate eines Punktes q'_i mit der Änderung der Koordinate p'_i ändert, d.h. $cov(p'_x, q'_x)$ sagt uns, wie sich die x Koordinate von q'_i mit der Änderung der x Koordinate von p'_i ändert, wenn die Punkte übereinstimmen.

4. Ermittelung der Rotation R und der Translation t: Die Singulärwertzerlegung (Singular Value Decomposition (SVD)) wird auf der Kreuzkovarianzmatrix K angewendet um diese als Produkt dreier $\mathbb{R}^3 \times \mathbb{R}^3$ Matrizen U, D und V darzustellen. Damit bekommen wir

$$K = UDV^T$$

Dabei sind die Spalten von U und V orthonormal und die Matrix D ist diagonal bestehend aus den Singulärwerte σ_1, σ_2 und σ_3 von K mit $\sigma_1 \geq \sigma_2 \geq \sigma_3$. D.h.

$$K = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

Somit ergibt sich die gesuchte Rotation R als

$$R = UV^T$$

und die optimale Translation t ist durch

$$t = \bar{q} - R\bar{p},$$

definiert, wobei $\bar{q}$ bzw. $\bar{p}$ den Mittelwert aller Punkte der Q bzw. Q beschreibt. Mittels der berechneten Rotation R und des Translation t erzeugt man eine erste Schätzung der Punktwolke Q, dazu wendet man R und t auf alle Punkte der Punktwolke P an.

Jetzt können wir unsere ursprünglichen Punkte wie folgt umformulieren:

$$p_i = p'_i + \bar{p}$$
$$q_i = q'_i + \bar{q}$$

Diese werden in unsere ursprüngliche Fehlerfunktion $E(R,t)$ aus der Gleichung (1) eingesetzt:

$$E(R,t) = \sum_{i=1}^{N} ||q'_i + \bar{q} - R(p'_i + \bar{p}) - \mathbf{t}||^2 \tag{3}$$
$$= \sum_{i=1}^{N} ||Rp'_i - q'_i + (\bar{q} - R\bar{p} - t)||^2 \tag{4}$$

Also wenn $t = \bar{q} - R\bar{p}$, dann wird unser ursprüngliches Optimierungsproblem über 2 Variablen (R,t) zu einem Optimierungsproblem über nur eine einzige Variable R, was weitaus überschaubarer ist und sich leichter in geschlossener Form lösen lässt:

$$E(R) = \sum_{i=1}^{N} ||Rp'_i - q'_i||^2 \tag{5}$$

In [2] lässt sich beweisen, dass zur Minimierung von Gleichung (3) (und damit auch von Gleichung (5)) die Rotationsmatrix wie folgt definiert werden sollte:

$$R = UV^T$$

5. Wiederholung des Prozesses: Nach dem Schritt 4 werden die Rotationsmatrix und der Translationsvektors auf das gedrehte Modell (Destination Modell) angewendet, um das Destination Modell auf das Source Modell anzupassen. Aber dafür ist ein iterativer Vorgang nötig, weil der Algorithmus meistens erstes mal nicht das bestmögliche Ergebnis realisieren kann.

   Der Algorithmus kann in Abhängigkeit von den folgenden Parametren wenige oder mehrere Iterationen brauchen, um das bestmögliche Match-Ergebnis zu bekommen:

   - Rotationswinkel von dem Destination Modell
   - Die Anzahl von den Punkten der Modelle
   - Geometrische Komplexität

   In diesem Schritt werden die Punkte des Destination Modells auf die entsprechenden Punkte des Source-Modells umgeschrieben, und der ICP-Algorithmus beginnt wieder von Schritt 1. Das bedeutet, die o.g. Schritte werden iterativ wiederholt. Dies geschieht so lange bis die Verbesserung des Match-Ergebnisses nicht mehr möglich ist, oder mit anderen Worten, bis die Roationsmatrix und Translationsvektor nicht mehr geändert werden können.

   Die Parameter, die den Output von dem ICP-Algorithmus beschreiben sind:

   - Rotationsmatrix R
   - Translationsvektor t
   - Match-Ergebnis (visuell und quantitativ bzw. prozentual)
   - Anzahl der nötigen Iterationen für das bestmögliche Match-Ergebnis (mithilfe von einem Abbruchkriterium)

   In dem nachfolgenden Kapitel werden der Abbruchkriterium und das Match-Ergebnis detailliert erklärt.

2.4 Variante des ICP

Nach der Einführung des grundlegenden ICP-Algorithmus wurde die Forschung zur Verbesserung der Geschwindigkeit und Effizienz des ICP fortgesetzt. Es wurden viele verschiedene Ansätze verwendet, um unterschiedliche Effekte zu erzielen und damit den ICP-Algorithmus zu verbessern. Einige Ansätze konzentrieren sich auf die Erhöhung der ICP-Iterationsgeschwindigkeit, die Korrektheit der Korrespondenzsuche und die Geschwindigkeit der Korrespondenzsuche. Eine ausführliche Übersicht findet sich z.B. in [15]. In diesem Abschnitt werden einige der möglichen Wege zur Optimierung des ICP-Algorithmus aufgezeigt.

Korrespondierende Punkte

In der ursprünglichen Arbeit [4] wird es mit allen Punkten in beiden Wolkenpunkten gearbeitet. Bestimmte Verbesserungen wurden bei der Verwendung einer gleichmäßigen Unterauswahl (sub-sampling) von gegebenen Punkten und bei der Zufallsauswahl (random sampling) vorgenommen [17, 18]. Das Problem tritt bei Rauschen oder stark gekrümmten Wolkenpunkten auf, bei denen diese Methoden nicht genügend Referenzpunkte auswählen (Abbildung 4, links). Die Lösung besteht darin, die Normalen für die Stichprobenziehung zu verwenden [9]. Die Erkennung der korrespondierenden Punkte mit Hilfe von Normalen ist in Abbildung 4 rechts dargestellt.

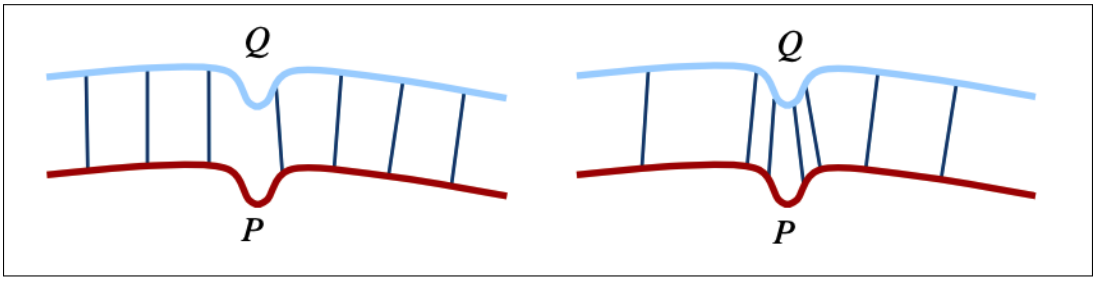

Abbildung 4: Zufallsauswahl (links) und Stichproben mit Normalen (rechts) [15].

k-d tree

k-d tree [3] sind ein Spezialfall von binären Raumaufteilungsbäumen, die in vielen Anwendungen als geeignete Struktur verwendet werden, z. B. bei der Suche mit einem mehrdimensionalen Suchschlüssel (z. B. Bereichssuche und Suche nach dem nächsten Nachbarn). Der ICP-Algorithmus kann durch ihre Verwendung effizient beschleunigt werden [19]. Allerdings wurde in [16] erklärt, wie die Verwendung des k-d tree die Geschwindigkeit der Registrierung um das 10-fache verbessert werden kann. In der Implementierung von ICP-Algorithmus (Abschnitt 3) wird die k-d tree-Technik auch verwendet.

Punktvergleich

Die Berechnung der euklidischen Metrik zwischen Punkten in [4] führt jedoch zu einem erheblichen Rechenaufwand. Dies kann durch die Verwendung der Punktsortierung im kd-

tree führt reduziert werden.

Es gibt eine Reihe von Methoden zur Beschleunigung der Abstandsberechnung. Die meisten Möglichkeiten der Punktvergleichsmethoden werden in [27] abgedeckt. Die normale Methode [6] basiert auf der Minimierung des Abstands nicht nur zwischen zwei Punkten, sondern auch des approximierten Abstands des Punkts von der Ebene. Diese Berechnung reduziert die Anzahl der Iterationen, aber die Methode bleibt nicht so robust.

Andere Methoden basieren auf der Projektion von Punkten aus einer Punktwolke auf eine andere. Das Zentrum der Projektion kann als die Position einer Kamera, die scannt oder allgemein gegeben werden. Dann kann man einen Schnittpunkt auswählen oder mit zwei Punkten arbeiten, die von der Entfernung oder der Intensität [29] oder der Farbe [20] abhängen. Es ist auch möglich, die sogenannte Hausdorff-Metrik [1] zu verwenden, wie in [8], in dem einige theoretische Ergebnisse zur Verwendung der Hausdorff-Metrik in ICP untersucht werden.

[1] https://de.wikipedia.org/wiki/Hausdorff-Metrik

3 Implementierung

Dieser Abschnitt befasst sich mit der Implementierung des im Abschnitt 2.3 beschriebenen Algorithmus. Zunächst werden die verwendete Bibliotheken und Modulen kurz vorgestellt. Anschließend wird es auf der Beschreibung der wichtigsten Funktionen in der Implementierung eingegangen. Schließlich wird eine Optimierung kurz präsentiert und erläutert.

Bei der Umsetzung des Codes werden die Parameter zum Einstellen genauer definiert und erklärt, somit sollte es dem Benutzer ein Einfaches sein das Programm richtig zu verwenden. Es gibt 3 verschiedene Optionen wie man den Code bedienen kann: (1) Man verschiebt das Destination Modell um eine fixe Rotation und eine fixe Translation, (2) Man macht unterschiedliche, äquidistanten Drehungen, wobei man den Start und Endwert und die Schrittweite angibt, (3) Man dreht das Modell in allen 3 Dimensionen in einer vordefinierten Weise.
Die im diesem Abschnitt vorgestellten Implementierung basiert auf [7] mit weiteren Optimierungen um beste Match-Ergebnisse zu erzielen. Details zu den Optimierungen sind im Abschnitt 3.3 gegeben.

3.1 Verwendete Tools und Bibliotheken

Python

Der in Abschnitt 2.3 vorgestellte ICP-Algorithmus wurde mit Python implementiert. Python [25] ist die führende Programmiersprache im TIOBE[2] und PYPL Index[3]. Es ist eine interpretierte, objektorientierte High-Level-Programmiersprache mit dynamischer Semantik. Seine hochentwickelten, integrierten Datenstrukturen, kombiniert mit dynamischer Typisierung und dynamischer Bindung, machen es sehr attraktiv für die schnelle Anwendungsentwicklung sowie für die Verwendung als Skript- oder Klebesprache, um bestehende Komponenten miteinander zu verbinden. Python ist eine einfache, leicht zu erlernende Syntax, die die Lesbarkeit betont und somit die Kosten für die Programmpflege reduziert. Python unterstützt Module und Pakete, was die Modularität des Programms und die Wiederverwendung von Code fördert.

Jupyter Notebook

Ein Jupyter Notebook [14] ist eine Open-Source-Webanwendung, mit der Datenwissenschaftler Dokumente mit Live-Code, Gleichungen und anderen Multimedia-Ressourcen erstellen und gemeinsam nutzen können.

Jupyter-Notebooks werden für alle Arten von datenwissenschaftlichen Aufgaben verwendet, wie z. B. explorative Datenanalyse (EDA), Datenbereinigung und -umwandlung, Datenvisua-

[2] https://www.tiobe.com/tiobe-index/
[3] https://pypl.github.io/PYPL.html

lisierung, statistische Modellierung, maschinelles Lernen und Deep Learning.

Jupyter-Notebooks können auch über die Weboberfläche in eine Reihe von Standardausgabeformaten (HTML, Powerpoint, LaTeX, PDF, ReStructuredText, Markdown, Python) konvertiert werden. Diese Flexibilität macht es Datenwissenschaftlern leicht, ihre Arbeit mit anderen zu teilen.

PyCharm

PyCharm[4] ist eine dedizierte integrierte Entwicklungsumgebung (IDE) für Python, die eine breite Palette an wichtigen Werkzeugen für Python-Entwickler bietet, die eng integriert sind, um eine komfortable Umgebung für die produktive Entwicklung von Python, Web und Data Science zu schaffen.

Zur Implementierung des ICP-Algorithmus werden nachfolgende Python-Bibliotheken bzw. -Modulen verwendet. Weitere Einzelheiten zu jeder Bibliothek bzw. zu jedem Modul sind in der entsprechenden Referenz gegeben.

NumPy

NumPy (auch Numeric Python) [10] ist eine Bibliothek für wissenschaftliche Berechnungen in Python. Sie bietet mathematische Funktionen auf hohem Niveau und eine mehrdimensionale Struktur (bekannt als ndarray) für die Bearbeitung großer Datensätze.

Open3D

Open3D [32] ist eine Open-Source-Bibliothek, die die schnelle Entwicklung von Software für den Umgang mit 3D-Daten unterstützt. Das Open3D-Frontend stellt eine Reihe sorgfältig ausgewählter Datenstrukturen und Algorithmen in Python zur Verfügung. Das Backend ist hoch optimiert und für die Parallelisierung eingerichtet.

PyVista

PyVista [23] ist eine Hilfsbibliothek für das Visualization Toolkit (VTK), die einen anderen Ansatz für die Schnittstelle zu VTK über NumPy und direkten Array-Zugriff verfolgt. Dieses Paket bietet eine gut dokumentierte Python-Oberfläche, die das leistungsstarke Visualisierungs-Backend von VTK offenlegt, um schnelles Prototyping, Analyse und visuelle Integration von räumlich referenzierten Datensätzen zu erleichtern.
Dieses Modul kann für wissenschaftliche Plots für Präsentationen und Forschungsarbeiten sowie als unterstützendes Modul für andere netzabhängige Python-Module verwendet werden.

[4] https://www.jetbrains.com/pycharm/

Scipy

SciPy (auch Scientific Python) [26][5] ist eine Bibliothek für wissenschaftliche Berechnungen, die NumPy als Grundlage verwendet. In der Implementierung werden zwei wichtige Klassen aus dem Modul scipy.spatial benötigt: Scipy Kdtree und Scipy Rotation. Die Klasse Scipy Kdtree ist ein Teil des Moduls scipy.spatial und kann für die schnelle Suche nach den nächsten Nachbarn verwendet werden . Diese Klasse bietet einen Index in einer Menge von k-dimensionalen Punkten, der verwendet werden kann, um schnell die nächsten Nachbarn eines beliebigen Punktes zu finden. Die Klasse Scipy Rotation ist in dem Modul scipy.spatial.transform und bietet eine Schnittstelle zur Initialisierung und Darstellung von Rotationen mit Quaternionen, Rotationsmatrizen, Rotationsvektoren, Modifizierte Rodrigues-Parameter und Euler-Winkel.

Matplotlib

Matplotlib [13] ist die bekannteste Bibliothek für die Datenvisualisierung mit Python. Es bietet sowohl eine schnelle Möglichkeit zur Visualisierung von Daten aus Python als auch Abbildungen in Publikationsqualität in vielen Formaten.

Json

JSON steht für JavaScript Object Notation und ist ein einfaches Format zum Speichern und Übertragen von Daten. JSON wird häufig verwendet, wenn Daten von einem Server an eine Webseite gesendet werden. Python verfügt über das integrierte Modul json[6], mit dem die JSON-Daten bearbeitet werden können.

3.2 Beschreibung der wichtigsten Funktionen

Der implementierte Python-Skript des ICP-Algorithmus (Quellcode **??**) besteht aus den folgenden Abschnitte und befindet sich im Anhang (Abschnitt 6).

- **Import-Abschnitt**: Hier sind die Module und Bibliotheken zu sehen, die für die Bearbeitungen und Visualisierungen in der Arbeit verwendet. In diesem Teil muss man zum Benutzen des Programms nichts ändern.
- **Konfiguration-Abschnitt**: Bei Visualisierung von den Inputs und Outputs und Berechnungen in dem ICP-Algorithmus gibt es verschiedene Parameter, die je nach Wünsch und Bedarf des Nutzers und in Abhängigkeit von den Polygon Modellen unterschiedlich definiert sein müssen, um die richtigen Outputs zu realisieren. Diese Sektion ist dafür erstellt, um ein modulares Skript zu haben, wobei die Nutzer die nötigen Parameter in Abhängigkeit von Ihren Aufgaben und Bedarfen richtig und entsprechend eingeben können. Alle diese Parameter sind in dem Python-Skript als Variablen definiert. Im

[5] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html
[6] https://docs.python.org/3/library/json.html

nachfolgenden werden alle 22 einstellbaren Parameter erklärt.

Einstellbare Parameter

1. option_include_every_nth_point_source: Dieser Parameter ein positiver Integer-Wert, der dafür definiert, um beim Bedarf die Punkte des Source Modells, teilweise zu löschen und nimmt nur den nten Punkte des Modells für die Bearbeitung. Manche Polygon Modelle bestehen aus extrem vielen Punkten und nicht allen Punkten beschreiben die Oberfläche der Modelle. In diesem Fall braucht der ICP-Algorithmus mehr Zeit für die Bearbeitung der Modelle und Visualisierung der Outputs. Um die Matchergebnisse für die großen Modelle schneller und bequem zu berechnen zu lassen, kann man bei der oben genannten Option eine höhere Anzahl eingeben. Somit ist diese Option dafür gedacht den Code zu beschleunigen, besonders relevant bei Modellen mit sehr vielen Punkten. Der Parameterwert 1 bedeutet, dass alle Punkte des Modells angenomen werden und 100 bedeutet, dass 1/100 der Punkte des Modells angenommen werden.
2. option_include_every_nth_point_destination: Dieser Paremeter macht die gleiche Funktion für das Destination Modell wie der vorherige Parameter. Es ist empfohlen, dass dieser Parameter den gleichen Wert (gleich 1) wie bei dem Ersten aufnimmt.
3. option_opacity_sphere : Diese Variable kann die Transparenz von der Oberfläche der Kugel erhöhen oder reduzieren und liegt im Wertbereich zwischen 0 und 1. Der Wert 0 bedeutet komplett transparent und 1 bedeutet komplett intransparent. Aus der bisherigen Erfahrung ist ein Wert von 0.45 zu empfehlen.
4. option_size_point_on_sphere: Die perfekten Match-Ergebnisse (> 98%) und die schlechten Match-Ergebnisse (< 98%) werden am Ende der Bearbeitung je mit grünen und roten Punkten auf der Oberfläche der Kugel visualisiert. Die Rotationswinkel von diesen Punkten zeigen die Rotationswinkel von dem Destination Modell vor dem Start des ICP-Algorithmus. Dieser Parameter ist ein positiver Integer-Wert und beeinflusst die Größe von den o.g. Punkten. Umso größer man den Wert wählt, umso größer erscheint der Punkt auf der Kugel, damit es übersichtlich aber trotzdem ersichtlich bleibt ist ein Wert von 10 in den meisten Fällen empfehlenswert
5. option_color_matched_points: Es ist ein String, welcher die Farbe der Punkte definiert, die die perfekten Match-Ergebnisse (Match-Ergebnisse >98%) zeigen. Der Wertbereich umfasst nur vordefinierte Werte, wie z.B. „blue“, „green“, „red“.
6. option_color_unmatched_points: Analog zum obigen Parameter nur für die Punkte welche die schlechten Match Ergebnisse darstellen. Der Wertbereich umfasst auch nur vordefinierte Werte, wie z.B. „blue“, „green“, „red“.
7. option_color_histogram: Mit diesem Parameter stellt man die Farbe der Spalten des Histogramms ein. Der Wertbereich enthält nur vordefinierte Werte, wie z.B. „blue“, „green“, „red“, "maroon".

8. option_width_histogram = 0.4: Damit gibt man die Breite der Spalten des Histogramms an. Der Wertebereich ist die postive Floats. In den meisten Fällen hat sich ein Wert von 0.4 bewährt.
9. option_default_file_path_source: In diesem Parameter wird der Path des Source Modells als ein String-Wert eingeschrieben.
10. option_default_file_name_source: Damit definiert man die Name der Datei des Source Modells als ein String-Wert.
11. option_default_file_path_destination: In diesem Parameter wird der Path des Destination Modells als ein String-Wert eingeschrieben.
12. option_default_file_name_destination : Dieser Parameter definiert die Name der Datei des Destination Modells als ein String-Wert.
13. option_shift_parameter_for_shifting_destination_points: Dieser Parameter nimmt Arrays bestehend aus 3 Floatswerte auf und definiert die Werte zur Verschieben des Modells in Richtung x-, y- und z-Achse. Diese Einstellung kann nur für den Fall auswählen, das man das Modell um einen fixen Werte verschieben möchte.
14. option_switch_covariance_or_rotation_translation_comparison: Das ist ein Boolean-Parameter (Wertebereich = $\{True,\ False\}$). In diesem ICP-Algorithmus sind zwei Abbruchkriterien implementiert. Man kann diesem Parameter „True“ oder „False“ setzten, um das Abbruchkriterium durch die Covariance Matrix oder das Abbruchkriterium durch Rotation und Translation zu verwenden.
15. option_ICP_iterations: Dies definiert die maximale Anzahl an erlaubter Iterationen des ICP Algorithmus. In der Praxis hat sich ein Wert von 500 bewährt.
16. option_show_estimate_every: Damit beschreibt man nach wie vielen Iterationen der Nutzer die Zeit sehen möchte, die für die Berechnungen nötig sind.
17. option_radius_closest_points_correspondences: Es beschreibt den maximalen zulässigen Radius als ein positiver Float-Wert, in dem die corresponding points gesucht werden, diese bedeutet umso kleiner umso schneller die Berechnung jedoch kann es vorkommen, dass keine Punkte im Radius sind.
18. option_distance_threshold_for_matching_points: In diesem Parameter stellt man den maximalen zulässigen Distanz zwischen 2 Punkten ein als ein positiver Float-Wert, damit diese als gematcht gelten, somit entspricht dieser Parameter genau dem Parameter th_{match}. Ein Wert von 50 hat sich in der Praxis gut bewährt.
19. option_example_point_sphere_rotation: Es definiert die Drehung der Sphäre um die erfolgreichen und nicht erfolgreichen Matchings zu zeigen. Der angegebene Punkt in dieser Option beschreibt den am besten sichtbaren Punkt der Sphäre, somit sollte er so gewählt werden das dieser von besonderem Interesse ist. Dieser Parameter hat rein einen Einfluss auf die Visualisierung der Ergebnisse und nicht auf die Berechnung an sich. Als Wertbereich nimmt dieser Parameter 3-dimensionales Float-Arrays auf.

20. option_choose_show_labels_for_matches: Es definiert, wie die Labels für verschiedene Punkte der Match-Ergebnisse angezeigt werden sollen. Der Wertebereich sind Integers mit den folgenden vordefinierten Werten: 0=Ohne Labels, 1= Labels für alle Punkte, 2=Labels nur für die grünen Punkte, 3=Labels nur für die roten Punkte.
21. option_label_sphere_point_size: Die Labels haben bestimmt Punkte, die dahinter liegen. Diese Punkte sind nicht nötig. Es ist somit empfohlen, diese kleiner als die Punkte auf der Sphäre des Matchings zu wählen, damit diese unsichtbar bleiben. Als Wertbereich nimmt dieser Parameter Float-Werte auf.
22. option_label_sphere_font_size: In den Labels der Punkte der Match-Ergebnisse sind die Rotationswinkel des Destination Modells vor dem Start des ICP-Algorithmus geschrieben. Diese Einstellung beschreibt die Schriftgröße der Labels. Als Wertbereich nimmt dieser Parameter Float-Werte auf.

- **Abschnitt zur Funktionsdefinition**: In diesem Abschnitt sind die Funktionen definiert, die Entwicklungs- und Verbesserungsprozesse Schritt für Schritt definieren. Im Anhang (Abschnitt 6) befindet sich der vollständigen Code des Skripts und die wichtigsten Funktionen werden hier beschrieben.
  1. Pyvista Aufruf für die Visualisierung von den Polygon Modellen in 3 dimensionaler Umgebung
  2. Die Schritte von ICP: Jeder Schritt des ICP-Algorithmus ist als eine Funktion definiert und dann werden alle Funktionen mit gleicher Ordnung in der Funktion ComputeICP wieder aufgerufen.
  3. Um den Punkten der Match-Ergebnisse auf der Oberfläche der Kugel die gleiche Rotationswinkel über alle Achsen zuweisen zu können, ist unter der Funktion „rotation-matrix“ eine Euler-Matrix definiert.
  4. Unter der Funktion „angle_to_xyz“ konvertiert die Rotationswinkel, sodass die auf der Oberfläche der Kugel projiziert werden.
  5. Die Funktionen „draw_sphere“, „labels_sphere“ und „add_labels“ sind je für die Visualisierung von Kugel, Labels der Punkte von den Match-Ergebnissen zuständig.
  6. Die Funktion „histogram_plotting“ definiert die Parametre der x- und y-Achse des Histogramms.

- **Abschnitt zum Lesen und Ausführen**: Hier werden die Dateien (Polygon Modelle) eingelesen und kann sie durch den ICP-Algorithmus in drei verschiedenen Formen laufen. Der o.g. Abschnitt empfehlt dem Nutzer die folgenden Möglichkeiten.

  M1 : Man kann die Rotationswinkel des „Destination Modells“ beliebig als Grad definieren und in dem ICP-Algorithmus ein mal bearbeiten und laufen lassen.

  M2 : Man kann beliebig unterschiedliche Anfangs- und Endwerte für die Rotationswinkel des „Destination Modells“ definieren und einen Schritt-Wert (Step

Value) zwischen den Anfangs- und Endwerten eingeben. Das Python-Skript nimm die eingegebenen Rotationswerte an und lässt den ICP-Algorithmus für alle angenommene Rotationswerte laufen.

M3 : Mann kann keine neue Rotationswerte für das „Destination Modell“ definieren. Bei dieser Option wird das Skript den ICP-Algorithmus mit bestimmen Rotationswerten können über verschiedene Achsen als variable oder konstant definiert sein.

3.3 Optimierte Variante

Dieser Abschnitt zeigt, wie der bereits vorgestellte ICP-Algorithmus optimiert werden kann. Es wurde die Richtigkeit des ICP-Algorithmus und die Visualisierung der erwarteten Outputs erfolgreich geprüft. Die Polygon-Dateien können mithilfe von dem Open3d eingelesen und dann mit Pyvista in einer 3 dimensionalen Umgebung visualisiert werden. Match-Ergebnisse der Modelle am Ende der Bearbeitung werden visuell und quantitativ geprüft. Um die nutzlose Iterationen nach den bestmöglichen Match-Ergebnissen zu vermeiden, ist ein Abbruchkriterium erstellt. Kdtree ist für die Beschleunigung der Bearbeitungen erfolgreich integriert geworden. Die 3 dimensionale Polygon Modelle können manchmal sehr groß sein. Die Anzahl von den Punkten die eine Fläche oder ein 3 dimensionales Modell präsentieren, können subjektiv und unterschiedlich sein. Außerdem Kdtree ist in dieser Arbeit eine zusätzliche Möglichkeit erstellt, die die Bearbeitung bei dem Bedarf weiter beschleunigen kann, indem die Punkte der Polygon Modelle ja dem Wünsch und dem Bedarf von dem Nutzer teilweise löscht. Modularität von dieser Arbeit ist geprüft und das Python Skript kann unterschiedliche Polygon Modelle bearbeiten. Man kann beim Bedarf Rotationswinkel für das Destination Modell unterschiedlich definieren und um verschiedene Achsen unterschiedliche variable oder konstante Werte eingeben. Dann kann man die Richtigkeit von den Match-Ergebnissen auf der Oberfläche von einer Kugel mit grünen und roten Punkten visualisieren. Die Quantitativen Match-Ergebnisse kann man mit Histogramm visualisieren.

Match-Ergebnis

Ein wichtiger Parameter des Outputs von dem ICP-Algorithmus ist das Match-Ergebnis. Match-Ergebnis zeigt die Anzahl der Punkte, die die gleichen Teile von beiden Modellen präsentieren und am Ende der Bearbeitung übereinander liegen. Die quantitative Berechnung des Match-Ergebnisses match% ist wie folgt definiert:

$$match\% = \frac{100}{N} \sum_{i=1}^{N} N_i, \tag{6}$$

wobei

$$N_i = \begin{cases} 1 & \text{falls } ||p_i - q_j|| < th_{max} \text{ mit } i = 1, \cdots, N \\ 0 & \text{sonst} \end{cases}$$

Die Idee dieser Formel ist, dass nur die Punkte als gematcht gezählt werden, welche bzgl. Ihrer Position einen geringeren Abstand als eine vorgegebene Schranke th_{max} haben.
Der ICP-Algorithmus kann nicht immer am Ende der Bearbeitung ein perfektes Match-Ergebnis zeigen. Das Match-Ergebnis hängt von den folgenden Parametern ab:

- Rotationswinkel und Verschiebung des Destination Modells
- Geometrische Komplexität
- Vordefinierte Genauigkeit (th_{max})

Aus bisheriger Erfahrung ist es zu empfehlen, um den Algorithmus erst mit einem geringen Wert der Rotation des Destination Modells (z.B. 20 Grad) laufen zu lassen. Mit geringen Werten der Rotationswinkel des Destination Modells kann man ein perfektes Match-Ergebnis bekommen, in dem die Punkte der Modelle, die die gleichen Teile der Modelle präsentieren, übereinander stimmen oder sehr nah zu einander sind. In diesem Fall muss die quantitative Match-Ergebniss größer als 98% sein. Hier kann man beim Bedarf den Wert th_{max} ergebnisorientiert ändern, damit das visuelle Match-Ergebnis auf dem Bild des Outputs und der prozentuale Wert des Match-Ergebnisses bei einem perfekten Match einander passen und entsprechen.

Abbruchkriterium

Um eine Endlosschleife zu vermeiden, wird der ICP-Algorithmus nach einer, einstellbaren, maximalen Anzahl an Iterationen das Programm, unabhängig vom Ergebnis, abgebrochen und beendet. Dieser Wert ist aber ein subjektiver Wert und kann kleiner oder größer als der nötiger Wert sein. Unter dem Begriff „nötiger Wert für die Iterationen“ versteht man den genauen Wert der Iterationen, nach dem der ICP-Algorithmus das Match-Ergebnis nicht mehr verbessern kann, sobald der ICP konvergiert ist. Mit anderen Worten, nach einer bestimmten Iteration, auch wenn der ICP-Algorithmus weiter läuft und die wiederholten Schritte iteriert, kann die mittlere quadratische Abweichung (auch Mean Squared Error (MSE) genannt) nicht weiter optimiert werden. Dass bedeutet, dass die Kreuzkovarianzmatrix und die Rotationsmatrix werden sich ab einer bestimmten Iteration nicht mehr ändern und deswegen wird das Match-Ergebnis auch nicht verbessert. Diese Iteration ist die letzte hilfreiche Iteration und die weiteren Iterationen sind nicht sinnvoll. Die oben erklärte Eigenschaft kann dafür verwendet werden, um mithilfe der Kreuzkovarianzmatrix oder der Rotationsmatrix ein Abbruchkriterium für den ICP-Algorithmus zu erstellen, sodass der Algorithmus die letzte hilfreiche Iteration anerkennt und nach dieser Iteration die Bearbeitung und das Prozess automatisch stoppt.

Kovarianzvergleich: In jedem Iterationsschritt wird die Veränderung der Kovarianzmatrix mitbetrachtet, sobald diese zu gering ist, wird die Iteration abgebrochen. Im Detail beschreibt man das wie folgt: Die vorhergehende Kreuzkovarianzmatrix wird abgespeichert als $M_{CrossCovprevious}$ und es wird die Matrixnorm von $(M_{CrossCovprevious} - M_{CrossCov})$ berechnet, sobald diese den Wert von 10^{-9} unterschreitet ist das Abbruchkriterium erfüllt und die ICP

Iterationen stoppen. Der Wert von 10^{-9} wurde gewählt, da er eine sehr hohe Genauigkeit beschreibt, man sich jedoch Problem auf Grund von Rundungsfehler der Operationen nicht beeinflusst wird und man somit eine hohe Anzahl an unnötigen Iterationen vermeidet. Wenn man die Schranke kleiner wählt, werden mehr Iterationen berechnet.

Rotations-Translation Vergleich: Dieses Abbruchkriterium ist analog zum obigen Kovarianzvergleich, man nimmt anstelle der Kreuzkovarianzmatrix jedoch die Rotationsmatrix und den Translationsvektor, beide müssen ihr jeweiliges Abbruchkriterium erfüllen damit die ICP Iteration wirklich gestoppt wird. Man kann die Anzahl von den hilfreichen Iterationen in dem Output von dem Python-Skript sehen.

4 Ergebnisse und Bewertung

Bevor die verschiedenen Ergebnisse vorgestellt werden, werden die in dieser Arbeit verwendeten Daten kurz erläutert.

4.1 Ergebnisse eines 2-dimensionalen Modells

4.1.1 Match-Ergebnis zu M1

Dieses Abschnitt präsentiert die Match-Ergebnisse im Fall der die Möglichkeit M1 aus dem Abschnitt 3.2.
Als erstes Beispiel wird in dieser Arbeit ein simples 2 dimensionales Polygon-Modell bearbeitet. Das o.g. Modell namens „Human_noise_moddell“, hat das Format ASCII und besteht aus 120 Punkten, die mit mit x-, y-, und z-Koordinaten präsentiert sind, wobei alle z-Koordinaten gleich Null sind. Die Abbildung 5 stellt diesen Datensatz graphisch dar.

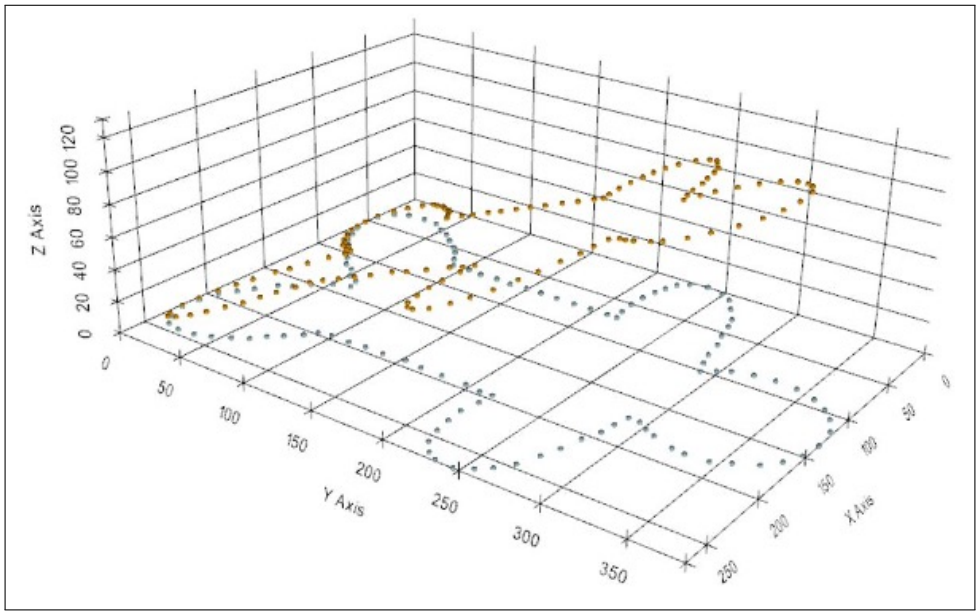

Abbildung 5: Graphische Darstellung des Human_Noise Modells vor der ICP-Anwendung.

Die blauen Punkte präsentieren das Source Modell und die Orangen Punkte präsentieren das Destination Modell. Das Destination Modell hat vor dem Start des ICP-Algorithmus bestimmte Rotationswinkel über verschiedenen Achse und die Outputs des ICP-Algorithmus können in Abhängigkeit von den vordefinierten Rotationswinkeln unterschiedlich sein. Diese Outputs sind:

- Rotationsmatrix
- Translationvektor
- Covariance Matrix

- Anzahl der Iterationen
- Macth-Ergebniss (visuel und quantitativ)

Im ersten Versuch des ICP-Algorithmus ist das zweite Polygon Modell (Destination Modell) vor dem Start des ICP-Algorithmus um 20 Grad über X-Achse rotiert. Dann lässt der Algorithmus beide Modelle laufen, um die bestmögliche Rotationsmatrix und Translationsvektor zu finden und ein Match-Ergebnis zu visualisieren. Die Abbildung 6 stellt graphisch das Endergebnis nach der ICP-Anwendung dar. Das Output des ICP-Algorithmus sieht wie folgt

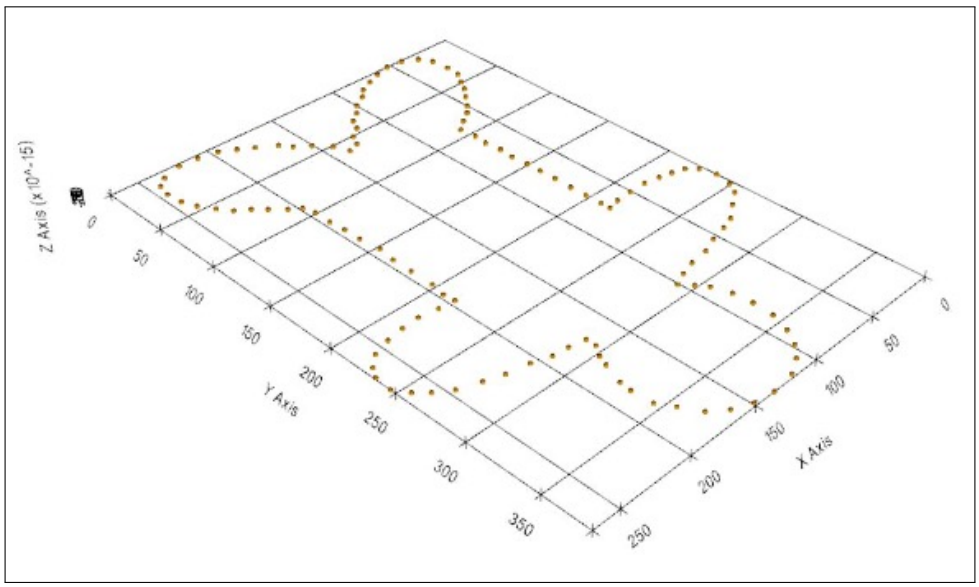

Abbildung 6: Graphische Darstellung des Human_Noise Modells nach der ICP-Anwendung.

aus:

```
Rotation & Transform Matrix for,  [20.  0.  0.] degree
Rotation :
[[ 9.99999860e-01  9.20242462e-05 -5.21896042e-04]
[ 9.20242462e-05  9.39692761e-01  3.42019745e-01]
[ 5.21896042e-04 -3.42019745e-01  9.39692621e-01]]
transform :
[[ 3.08925662e-04]
[ 4.00763182e-04]
[-1.55196934e-15]]
ICP stoped after  3  iterations
ICP Matched  100 %
```

Wie man feststellen kann, für einen geringeren Rotationswert wie 20 Grad kann der ICP-Algorithmus ein perfektes Match hinkriegen und die Punkte der Modelle werden am Ende des Prozesses übereinander liegen. Um das Macth-Ergebnis quantitativ am Ende von dem ICP-Algorithmus berechnen zu können, ist die Gleichung (6) verwendet.

Dafür sind nur drei Iterationen nötig. Der gleiche Prozess kann auch mit unterschiedlichen Werten für die Rotationsmatrix über die anderen Achsen verwendet werden. In Abbildung 7 wird das Ergebnis der Anwendung einer Rotation vom 20 Grad um Y- und Z-Achse vor bzw. nach der ICP-Durchführung visualisiert.

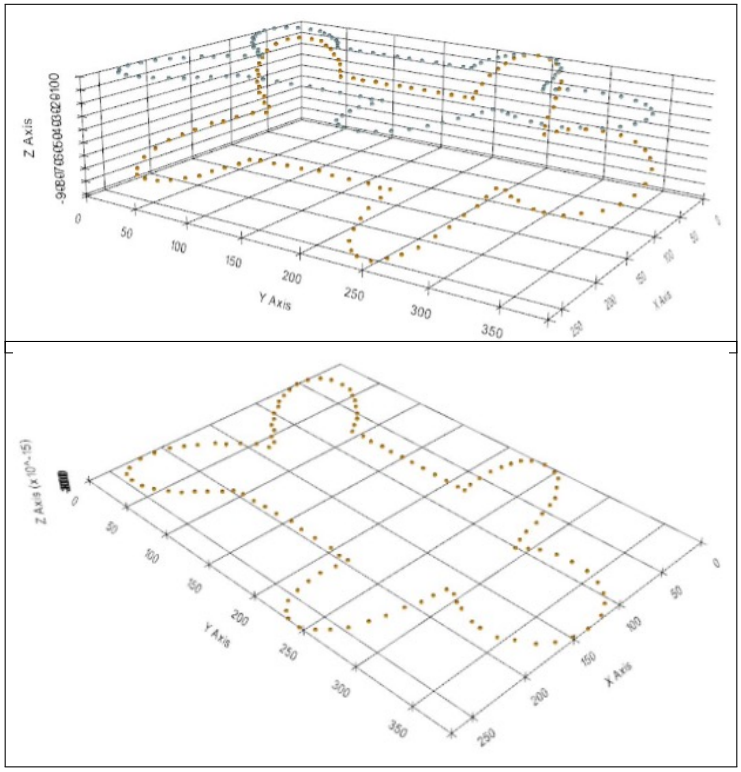

Abbildung 7: Graphische Darstellung des Human_Noise Modells vor (oben) und nach (unten) der ICP-Anwendung unter Verwendung einer Rotation vom 20 Grad um Y- und Z-Achse..

Analog zeigt die Abbildung 8 das Ergebnis der Anwendung einer Rotation vom 20 Grad um Z-Achse vor bzw. nach der ICP-Durchführung .
Der ICP-Algorithmus kann nicht für alle Rotationswerte ein perfektes Match-Ergebnis realisieren. Um das Problem besser zu visualisieren, muss der ICP-Algorithmus größere Werte von Rotationswinkel verwenden. Gleichzeitig mit Rotation über eine Achse oder mehrere Achsen, kann auch die Verschiebung in die Richtung von einer Achse oder mehrere Achsen angewendet werden, die die Aufgaben des ICP-Algorithmus anspruchsvoller machen, dafür werden auch mehrere Iterationen nötig sein. Im Folgenden sind einige visualisierte Beispiele (Abbildung 9) für einen Rotationswinkel vom 120 Grad um X-Achse gegeben:
Aus Abbildung 9 kann man feststellen, dass der ICP-Algorithmus mit einer Rotation vom 120 Grad um Z-Achse nicht das zu erwartende Ergebnis liefert, da das Destination-Modell mit dem Source-Modell nicht übereinstimmt. Daher kann man versuchen auch Verschiebungen dazu zu verwenden. D.h. Das Destination-Modell in dem ICP-Algorithmus kann nicht nur mit den Rotationswinkeln über verschiedene Achsen rotiert werden, sondern

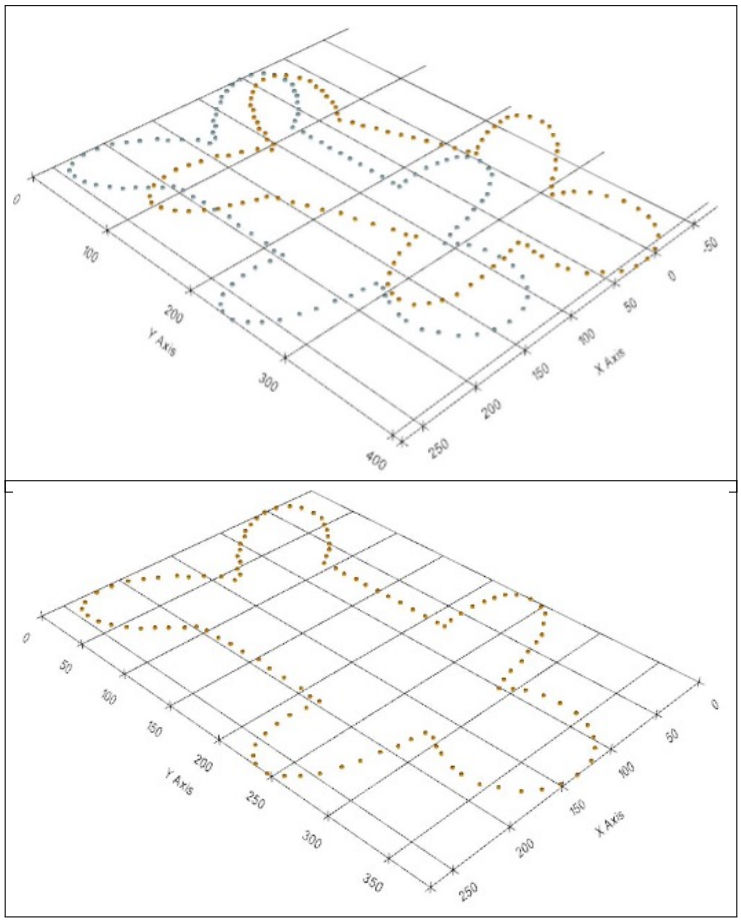

Abbildung 8: Graphische Darstellung des Human_Noise Modells vor (oben) und nach (unten) der ICP-Anwendung unter Verwendung einer Rotation vom 20 Grad um Z-Achse.

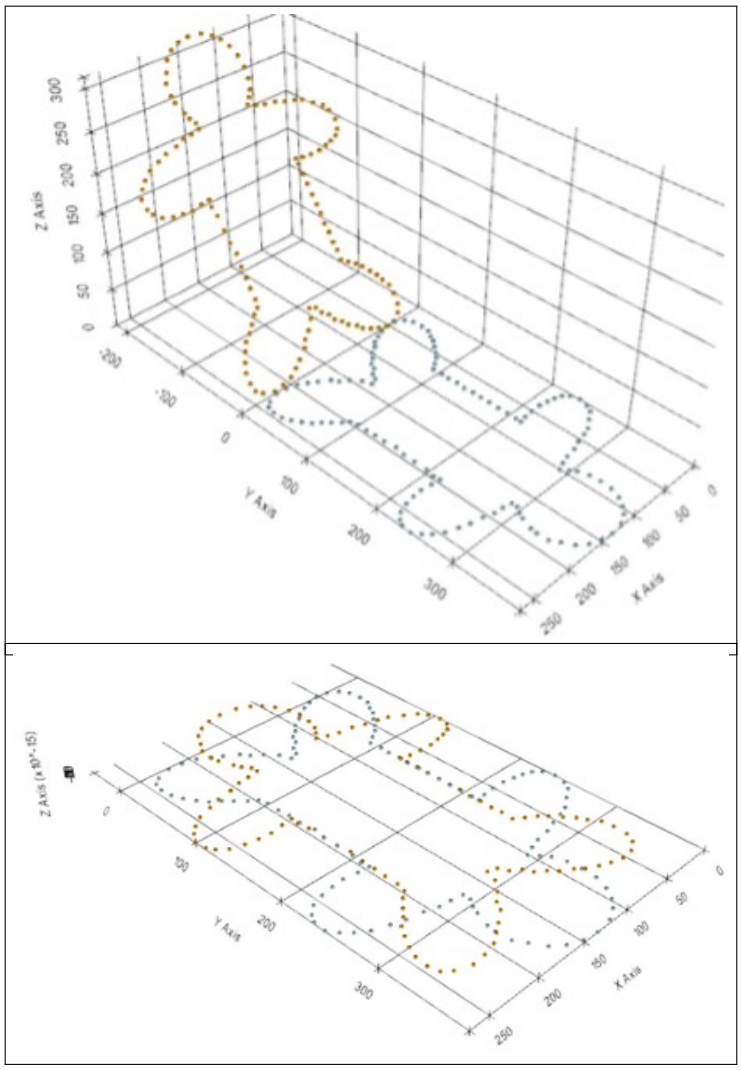

Abbildung 9: Graphische Darstellung des Human_Noise Modells vor (oben) und nach (unten) der ICP-Anwendung unter Verwendung einer Rotation vom 120 Grad um Z-Achse.

auch mit unterschiedlichen Verschiebungen in die Richtung von jede Achse verschoben sein. Man kann die Rotationswinkel und Verschiebungen gleichzeitig bevor dem Start von dem ICP-Algorithmus eingeben und visualisieren. Nächste zwei Beispiele (Abbildung 10) veranschaulichen die Ergebnisse bei der Anwendung der Rotation zusammen einer Translation vor und nach der ICP-Durchführung.

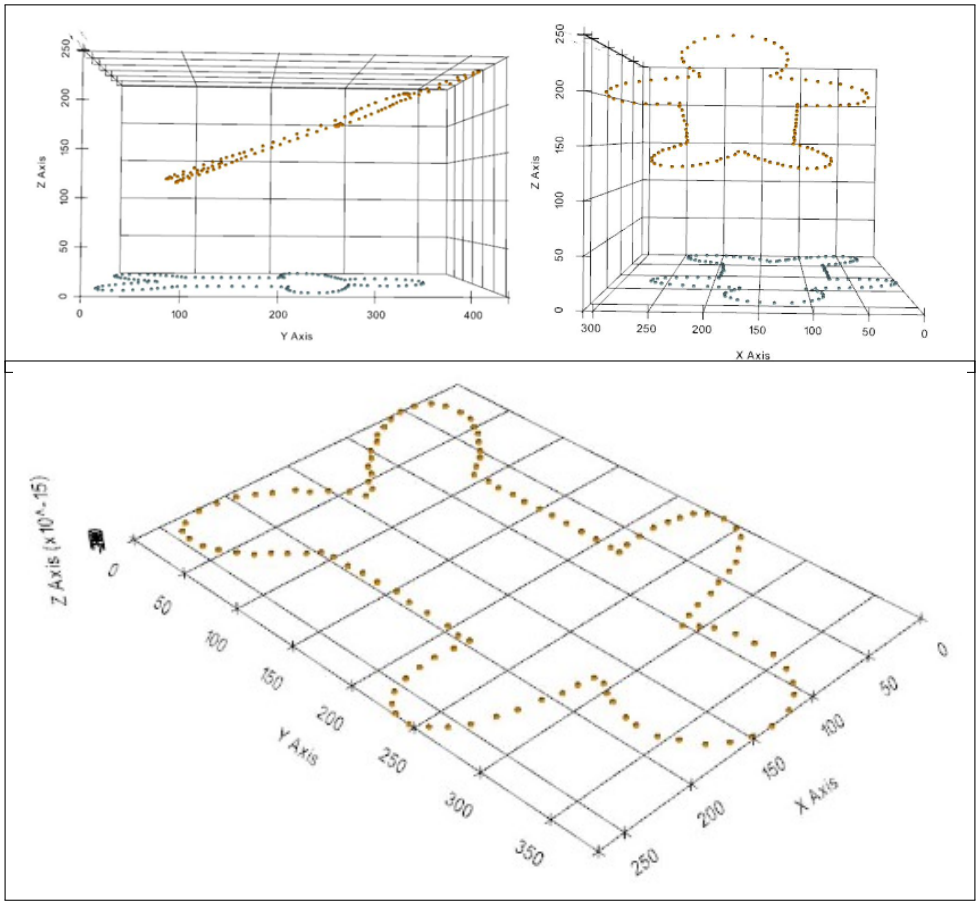

Abbildung 10: Darstellung des Human_Noise Modells vor (oben) bzw. nach (unten) der ICP-Anwendung unter Verwendung einer Rotation vom 20 Grad um X-Achse und einer Verschiebung um den Vektor (40,80,20).

Die Rotationswinkel für das Destination Modell bevor dem Start von dem ICP-Algorithmus ist [20, 0, 0] und die Verschiebungen sind [40, 80, 120]. Die ersten, zweiten und dritten Werte der Verschiebung stehen je für die Lände der Verschiebung von dem Destination Modell in die Richtung X-, Y-, und Z-Achse bevor dem Start von dem ICP-Algorithmus. In deisem Fall kann der ICP-Algorithmus ein perfektes Match-Ergebnis in dem Output bekommen.

Man kann die Polygon Modelle unterschiedlich mit verschiedenen variablen und konstanten Rotationswinkeln über verschiedene Achse in dem ICP-Algorithmus bearbeiten lassen. Um die Match-Ergebnisse der Bearbeitungen praktisch zu visualisieren, ist in dieser Arbeit in dem Python-Skript eine Möglichkeit erstellt, um eine Statistik durchzuführen. Dies wird in dem

folgenden Kapitel detailliert erklärt.

Um alle Match-Ergebnisse praktisch zu präsentieren, ist in dieser Arbeit eine Statistik durchgeführt, wobei die Match-Ergebnisse sowohl visuell, als auch quantitativ bzw. prozentual virtualisierbar sind. Für eine 3-dimensionale visuelle statistische Präsentation von den Matchergebnisse ist eine Kugel verwendet. Die quantitative Match-Ergebnisse sind mit dem Histogrammen visualisiert geworden.

In dem nächsten Schritt der Arbeit sind die Match-Ergebnisse von verschiedenen Versuchen mit grünen und roten Punkten auf der Kugeloberfläche visualisiert. Die grüne Farbe steht für ein perfektes Match-Ergebnis (98% oder höher) und die rote Farbe ist für die Match-Ergebnisse, die nicht perfekt sind (kleiner als 98%). Damit die Punkte auf der Kugeloberfläche in 3-dimensionaler Umgebung die gleichen Werte von Rotationen des Destination Modells vor dem Start des ICP-Algorithmus haben können, müssen die Koordinaten richtig konvertiert werden. Dafür ist in dieser Arbeit eine Euler-Matrix verwendet[7]. Die Euler-Matrix wandelt einen gegebenen Winkel in xyz-Koordinaten um und das wird mit Hilfe der Euler-Darstellung berechnet. Es gibt auch andere Darstellungen, wie die Kugelkoordinaten. Das Kugelkoordinatensystem sind ein Koordinatensystem, das die Punkte auf einer Kugel über zwei Winkel und einen Radius definiert. Wohingegen die Euler-Darstellung die Drehung des Modells darstellt und definiert drei Rotationswinkel über x-, y- und z-Achse. Der ICP-Algorithmus arbeitet mit zwei Modellen und ein davon (Destination Modell) vor dem Start des Algorithmus gedreht wird, und das Ziel ist die Drehung des Modells auf der Kugeloberfläche genau darstellen zu können, wobei man dann auch das Match-Ergebnis visuell anzeigen kann. Dafür muss man die exakt gleiche Rotation des Modells verwenden und das wird durch die Euler-Matrix erreicht. Das Algorithmus in dieser Arbeit berechnet zuerst die Euler-Matrix der entsprechenden Winkel und erstellt dann einen normalisierten Punkt, der einen Beispielpunkt im Modell darstellt und auf die Kugel abgebildet wird. Mann kann diesen Punkt in den Optionen „option_example_point_sphere_rotation“ beliebig ändern. Der Punkt [1,0,0] ist ein intuitiver Beispielpunkt, der die Bewegung des gesamten Modells darstellt. Dieser Punkt wird oft verwendet, um das Prinzip der Euler-Winkel zu erklären. Deshalb ist er so intuitiv und passt am besten zum gesamten Modell. Man kann diesen Punkt jedoch ändern, da es sich nur um einen Beispielpunkt handelt. Wenn man sich dafür entscheidet, nur einen Punkt auszuwerten, dann möchte man ihn auf jeden Fall auf einen bestimmten Punkt des Modells ändern, damit er genau diese Punktbewegung im Modell auf der Kugel darstellt.

Die Koordinaten bzw. die Rotation von den Punkten sind wie die Rotationen des Destination Modells vor dem Start des ICP-Algorithmus. In dem Label der Punkte kann man die Werte der Rotationen über x- y- und z-Achse sehen. Z.B. Ein grüner Punkt mit Label (20, 0, 0) auf der Oberfläche von der Kugel bedeutet, dass das Destination Modell mit einer Rotation von 20 Grad über x-Achse und 0 Grad Rotation über y- und z-Achse vor dem Start des ICP-Algorithmus, ist am Ende der Bearbeitung auf dem Source-Modell erfolgreich aufgelegt (Match-Ergebnis ist 98% oder mehr - Abbildung 11). Durch diese Art von Visualisierung der

[7] https://programming-surgeon.com/en/euler-angle-python-en/

Match-Ergebnisse kann man die Position von den roten Punkten sehen. Mit anderen Worten, der Nutzer weißt, welche Positionen er vermeiden muss, um ein perfektes Match-Ergebnis zu bekommen.

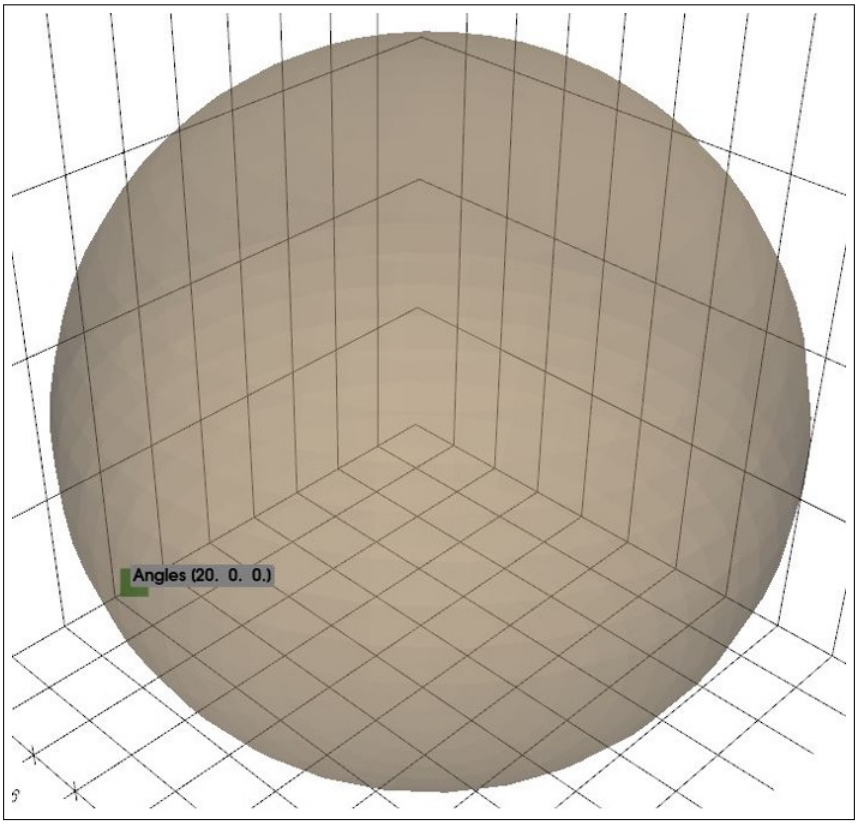

Abbildung 11: Match-Ergebnis des ICP-Algorithmus mit einer Rotation vom 20 Grad um X-Achse.

Dieser Prozess kann auch mit einem iterativen Vorgang für mehrere Rotationswinkel des Destination-Modells über eine oder mehrere Achsen durchführen. Dafür kann man die Rotationswinkel über die Achsen je nach Wunsch und Bedarf unterschiedlich als Variable oder Konstant definieren.

4.1.2 Match-Ergebnis zu M2

In diesem Teil lässt man das Python-Skript durch die Möglichkeit M2 aus dem Abschnitt 3.2 laufen. Hier kann man sowohl die Anfangs- und Endwerte über alle Achsen, als auch die step valuses zwischen den definierten Anfangs- und Endwerten definieren. In der Abbildung 12 sind die Match-Ergebnisse für das Human_Noise Modell mit folgenden Eingaben visualisiert:

- Anfansgwerte der Rotationen des „Destination Modells“ : [0, 0, 0]
- Endwerte der Rotationen des „Destination Modells“: [40, 40, 40]

- Step Value: [5, 5, 5]

Die ersten, zweiten und dritten Werte in den eckigen Klammern stehen je für x-, y-, und z-Achse.

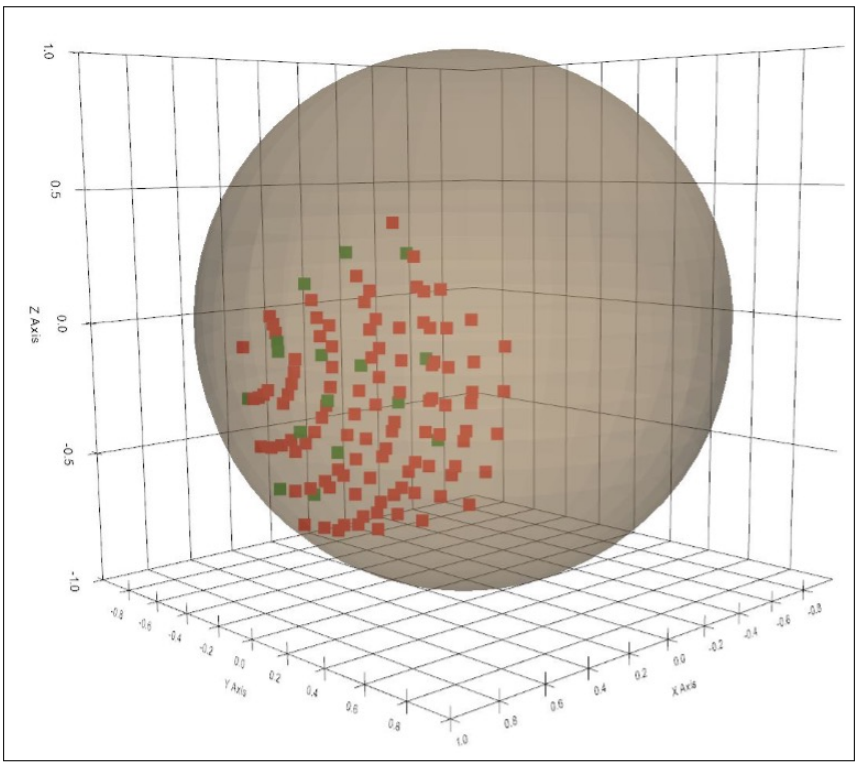

Abbildung 12: Match-Ergebnis zu Möglichkeit M2 .

Um die Anzahl von den perfekten (> 98%) und niedrigen (< 98%) Match-Ergebnissen zu zeigen und praktisch mit einem Blick zu visualisieren, sind in dieser Arbeit verschiedene Histogramme verwendet. Die Histogramme stellen dar wie die Verteilung der unterschiedlichen Versuche (jeder Versuch hat einen anderen Drehwinkel) in Bezug auf ihr Match-Ergebnis. Die x-Achse teilt die Balken von einem prozentualen Matching von 0-20, 20-40, 40-60, 60-80, 80-99, 99-100 ein, somit ist ersichtlich wie hoch die Anzahl der jeweils erfolglosen bzw. erfolgreichen Versuche ist (y-Achse). Der letzte Balken mit wurde mit 99-100% gewählt, da er das perfekte Match-Ergebnis wiederspiegelt und somit ein wichtiger Spezialfall ist. Ein Histogramm des Match-Ergebnisses ist in der Abbildung 13 gegeben.

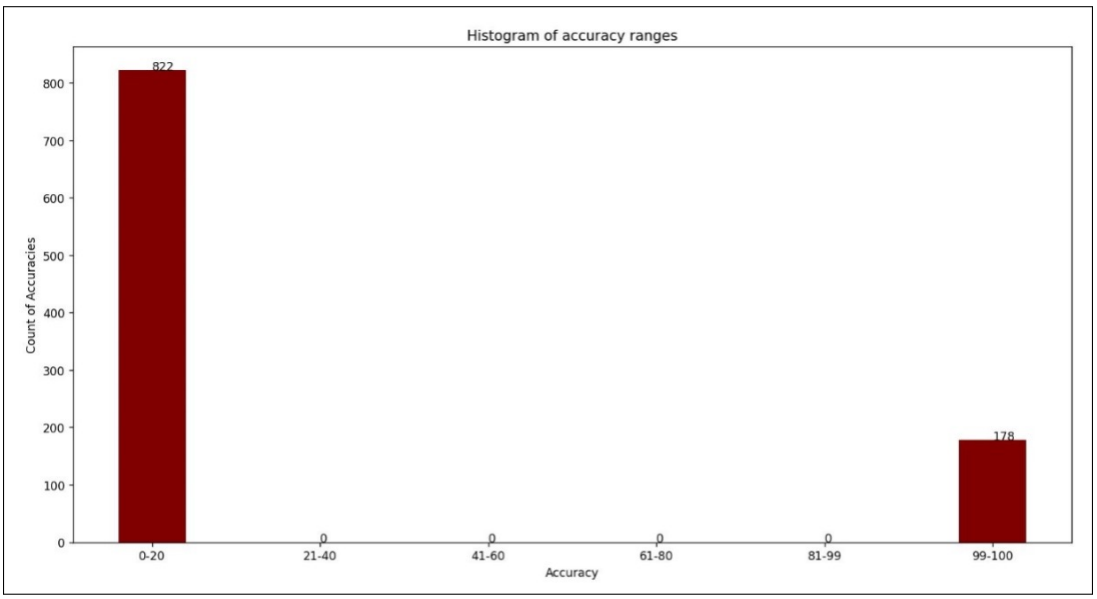

Abbildung 13: Histogramm des Match-Ergebnisses zu Möglichkeit M2 .

4.1.3 Match-Ergebnis zu M3

Im Fall der Möglichkeit M3 aus dem Abschnitt 3.2 kann man das Python Skript mit den bestimmten vordefinierten Rotationswinkeln über verschiedene Achsen laufen lassen, ohne neue Eingabe einzugeben. Die vordefinierten Rotationswinkeln in dem Skript können für verschiedene Achsen sowohl als Variable, als auch konstant definiert sein. Z. B. die Werte ([variable, 90, 0]) (wie in dem Python-Skript geschrieben ist) bedeuten, dass das Destination Modell vor dem Start des ICP-Algorithmus unterschiedliche (variable) Rotationen über x-Achse hat. Aber die Rotationen um y-und z-Achsen sind je gleich 90 und 0 und bleiben konstant. In Abhängigkeit davon, wie viele von den möglichen und virtualisierbaren Match-Ergebnisse man sehen möchte, kann man zwischen den variablen Werten einen Schritt-Wert (Step Value) definieren. In dem o.g. Fall mit den Rotationen ([variable, 90, 0]) und mit dem Step Value = 10 zwischen den Rotationen über x-Achse von 0 bis 360 Grad und den konstanten Werten von den Rotationen um y- und z-Achsen kreigt man 36 Punkte, die die Positionen des Destination-Modells vor dem Start des ICP-Algorithmus visualisieren (Abbildung 14). Die Labls der Punkte kann man je nach Bedarf mit dem Parameter option_choose_show_label_matches aktivieren. Dadurch kann man die Labls für alle Punkte, oder nur für Punkte der perfekter Macth-Ergebnisse, oder auch für niedrige Match-Ergebnisse aktivieren.

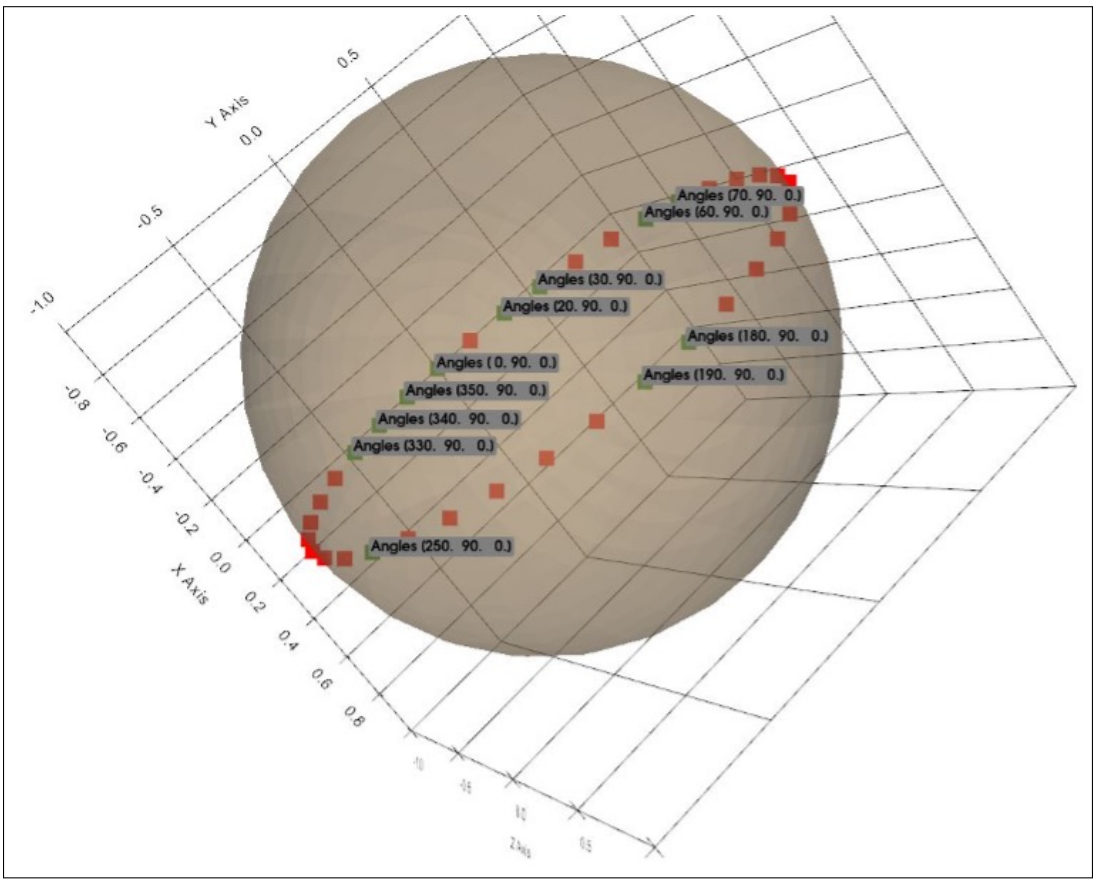

Abbildung 14: Match-Ergebnis zu Möglichkeit M3 .

Die schriftlichen Werte der Rotationen des "Destinnation Modellsäm Start des ICP-Algorithmus, die am Ende der Bearbeitungsphase des ICP ein perfektes Match-Ergebnis geben, sind:

```
ICP Matched:
[array([ 0., 90.,  0.]), array([20., 90.,  0.]), array([30., 90.,  0.]),
array([60., 90.,  0.]), array([70., 90.,  0.]), array([180.,  90.,   0.]),
array([190.,  90.,   0.]), array([250.,  90.,   0.]), array([330.,  90.,   0.]),
array([340.,  90.,   0.]), array([350.,  90.,   0.])] %
```

Die oben geschriebenen Punkte sind die grünen Punkten auf der Kugeloberfläche, die auch Label haben. Die schriftlichen Informationen über die Rotationen in dem Output und die Information in den Labeln der grünen Punkte sind gleich. So kann man die Match-Ergebnisse sowohl visuell, als auch schriftlich prüfen. Die Anzahl der Match-Ergebnisse in verschiedenen prozentualen Intervallen sind in der Abbildung 15 dargestellt.

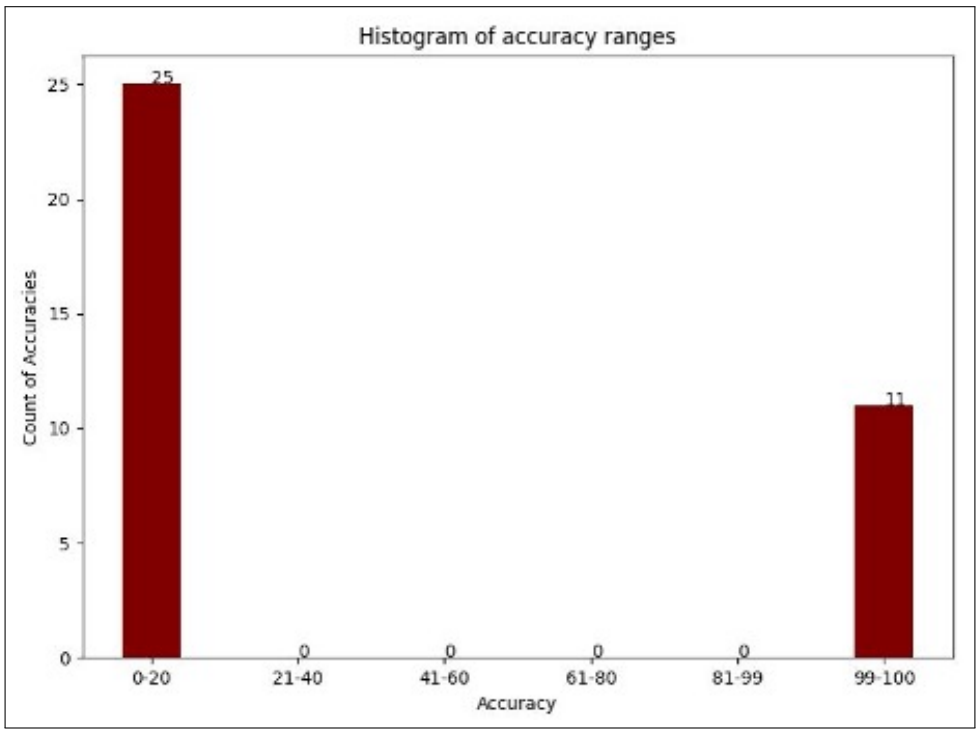

Abbildung 15: Histogramm des Match-Ergebnisses zu Möglichkeit M3 .

Mit variablen Werten des Rotationswinkels für das Destination Modell zwischen 0 bis 360 Grad um x-Achse, konstanten Werten der Rotation um die y- und z-Achsen und einem „Step Value = 10“ hat man insgesamt 36 Versuche. 11 davon haben ein perfektes Macth-Ergebniss (98% oder höher). 25 davon haben kein perfektes Macth-Ergebniss bekommen. Quantitativer Resultat von diesen 25 Match-Ergebnissen ist zwischen 0% und 25 % .

In dem Prozess, wo die Rotationswinkel des Destination Modells sich um verschiedene Achsen vor dem Start des ICP-Algorithmus iterativ ändert, kann unterschiedlich sein. In der Abbildung 16 sind die Match-Ergebnisse für die folgenden Rotationen mit grünen und roten Punkten auf der Kugeloberfläche visualisiert.

- ([variable, 90, 0]) : variable Werte der Rotation des Destination Modells vor dem Start des ICP-Algorithmus über x-Achse und konstante Werte der Rotationswinkel über y- und z-Achse je als 90 und 0
- ([0, variable, 0]): variable Werte der Rotation des Destination Modells vor dem Start des ICP-Algorithmus über y-Achse und konstante Werte der Rotationswinkel über x- und z-Achse je als 0 und 0
- ([0, 0, variable]): variable Werte der Rotation des Destination Modells vor dem Start des ICP-Algorithmus über z-Achse und konstante Werte der Rotationswinkel über x- und y-Achse je als 0 und 0

- ([45, 0, variable]): variable Werte der Rotation des Destination Modells vor dem Start des ICP-Algorithmus über z-Achse und konstante Werte der Rotationswinkel über x- und y-Achse je als 45 und 0
- ([135, 0, variable]): variable Werte der Rotation des Destination Modells vor dem Start des ICP-Algorithmus über z-Achse und konstante Werte der Rotationswinkel über x- und y-Achse je als 135 und 0

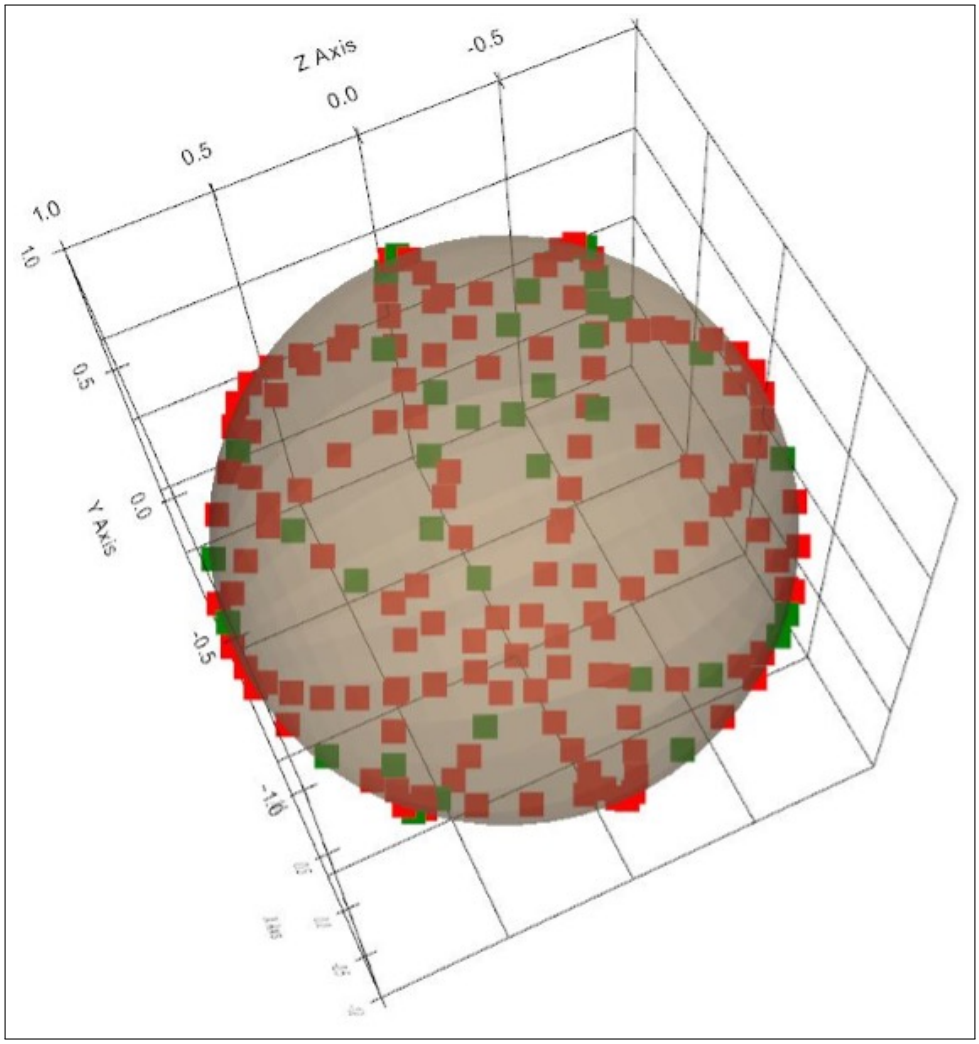

Abbildung 16: Match-Ergebnisse zu Möglichkeit M3 mit verschiedenen Rotationswinkel um mehrere Achse .

Analog sind die Match-Ergebnisse für den folgenden Werte des Rotationswinkels für das Destination Modell um verschiedene Achsen vor dem Start des ICP-Algorithmus in der Abbildung 17 als Histogramm visualisiert.

- ([variable, 90, 0]), ([0, variable, 0]), ([0, 0, variable]), ([45, 0, variable]), ([135, 0, variable])

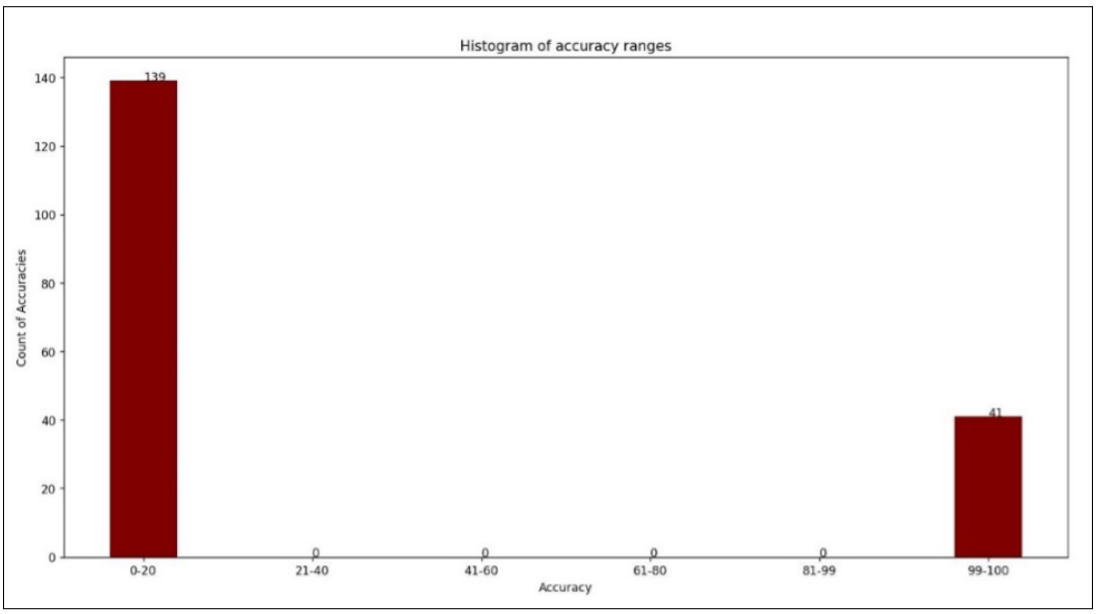

Abbildung 17: Histogramm der Match-Ergebnisse zu Möglichkeit M3 mit verschiedenen Rotationswinkel um mehrere Achse .

In dem Histogramm, 41 Versuche haben ein perfektes Match-Ergebniss (> 98%) gegeben und 139 haben niedrigere Match-Ergebnisse gegeben. Die Werte der Rotationen des Destination-Modells vor dem Start des ICP-Algorithmus von den Versuchen kann man in den „ICP Matched“ und „ICP Unmatched“ Listen in dem schriftlichen Output von dem ICP-Algorithmus sehen.

4.2 Ergebnisse eines 3-dimensionalen Modells

Analog zu dem 2-dimensionalen Modell, werden hier die Match-Ergebnisse zu jeder Möglichkeit (M1, M2 und M3) aus aus dem Abschnitt 3.2 für ein 3-dimensionales Modell, namens Mechpart-Modell, vorgestellt und bewertet. Dieses Modell, dargesellt in der Abbbildung 18, besteht aus 4100 Vertices, die jeweils als Zahlentripel aus (X,Y,Z) Floats gegeben sind und einem Objekt mit 7938 Polygon Faces.

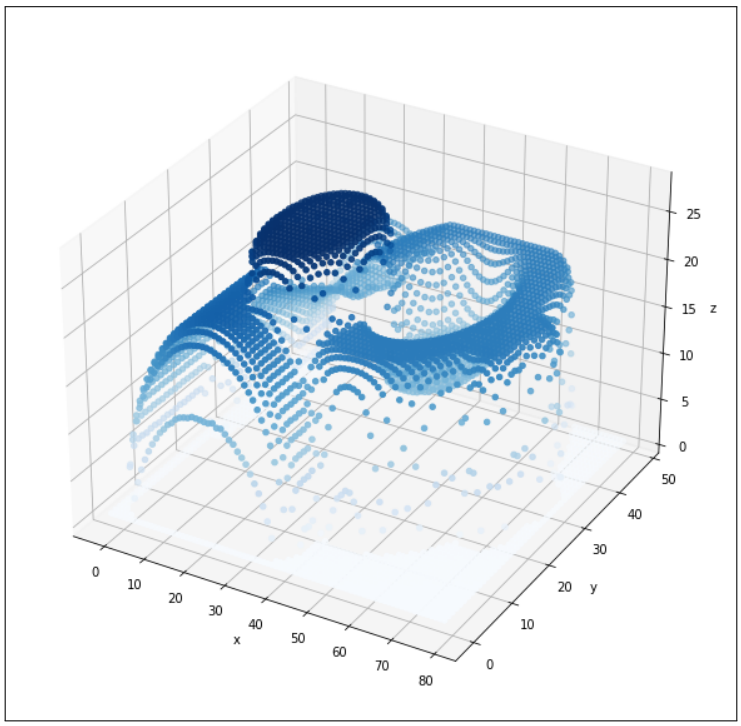

Abbildung 18: Graphische Darstellung des Mechpart-Modells vor der ICP-Anwendung .

4.2.1 Match-Ergebnis zu M1

Der ganze Prozess im Abschnitt 4.1.1 kann auch genau so für 3-dimensionales Polygon-Modell verwendet werden. Man kann im Allgemeinen den ICP-Algorithmus auch für höhere dimensionale Punktmodelle verwenden, jedoch kann man dies natürlich nicht mehr graphisch darstellen und sind nicht im Fokus dieser Arbeit.
Um die Match-Ergebnisse zu bewerten, wurde folgende Experimente (E1 bis E4) durchgeführt und graphisch dargestellt.

E1 : Hier wurde das Destination Modell um 20 Grad um x-, y- und z-Achse rotiert und die Ergebnisse sind vor bzw. nach der ICP-Anwendung in der Abbildung 19 veranschaulicht.

E2 : Dabei wurde nur eine Rotation um 125 Grad um x-Achse durchgeführt, um zu sehen, dass das Destination-Modell nach Bearbeitungsphase (nach Anwendung des ICP) nicht ganz auf das Source-Modell angepasst werden (Abbildung 20).

E3 : In diesem Experiment wurde eine Rotation zusammen mit einer Translation angewendet. Die Rotation ist um 20 Grad um x-Achse während die Translation um den Vektor (40,80,120). Das Ergebnis dazu ist in der Abbildung 21 dargestellt.

E4 : Hier wurden 3 verschiedene Rotationen gefolgt mit einer Translation durchgeführt. Die Erste um 100 Grad um x-Achse, die Zweite um 65 Grad um y-Achse und die Letzte

um 274 Grad um z-Achse, während die Translation um den Vektor (124,43,85). Das entsprechende Ergebnis ist in der Abbildung 22 dargestellt.

Das Match-Ergebnis nach der Bearbeitungsphase kann in Abhängigkeit von den Rotationswinkeln des Destination Modells höher oder niedriger als 98% sein. dies wird in den Abschnitte 4.2.2 und 4.2.3 geprüft, indem das Destination Modell unterschiedlich um mehrere Achsen rotiert wird.

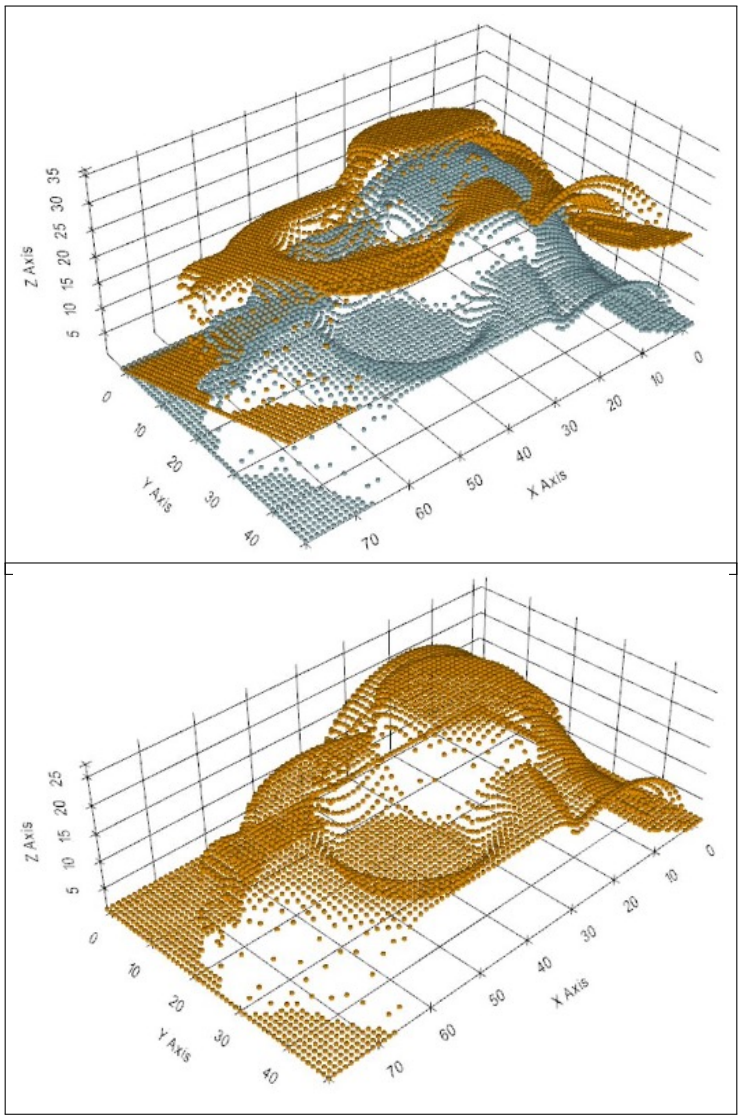

Abbildung 19: Graphische Darstellung des Mechpart-Modells vor (oben) und nach (unten) der ICP-Anwendung unter Verwendung einer Rotation vom 20 Grad um x-Achse.

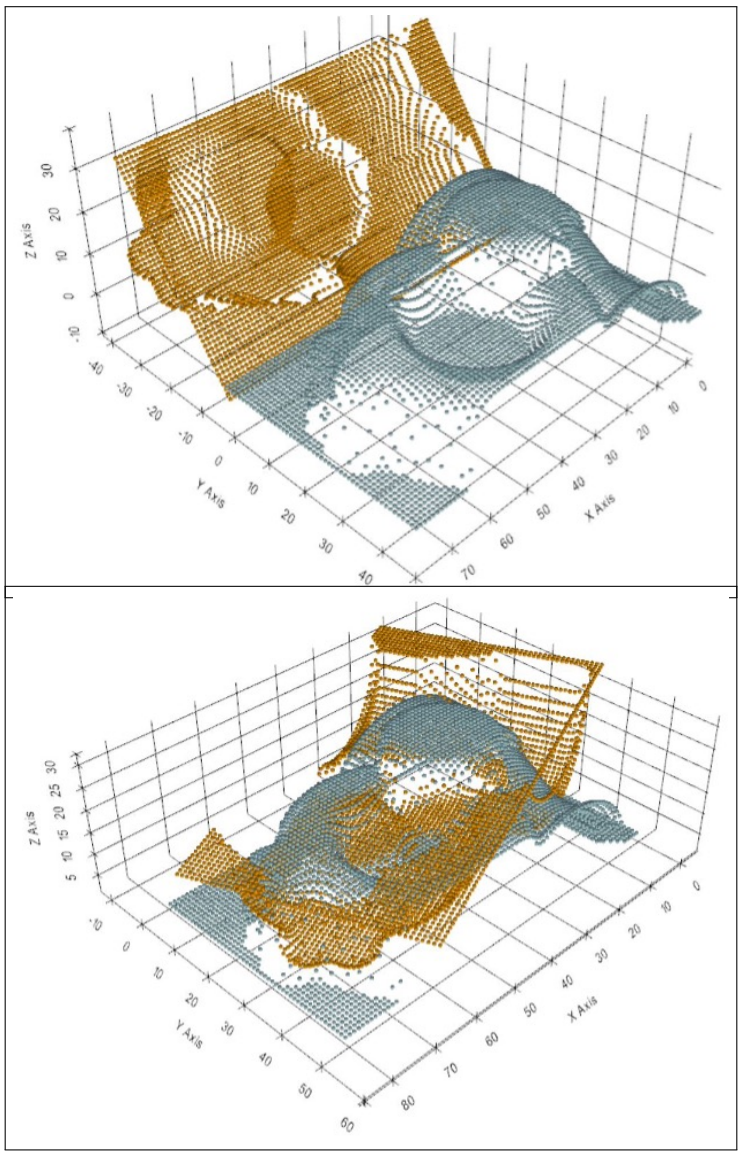

Abbildung 20: Graphische Darstellung des Mechpart-Modells vor (oben) und nach (unten) der ICP-Anwendung unter Verwendung einer Rotation vom 125 Grad um x-Achse.

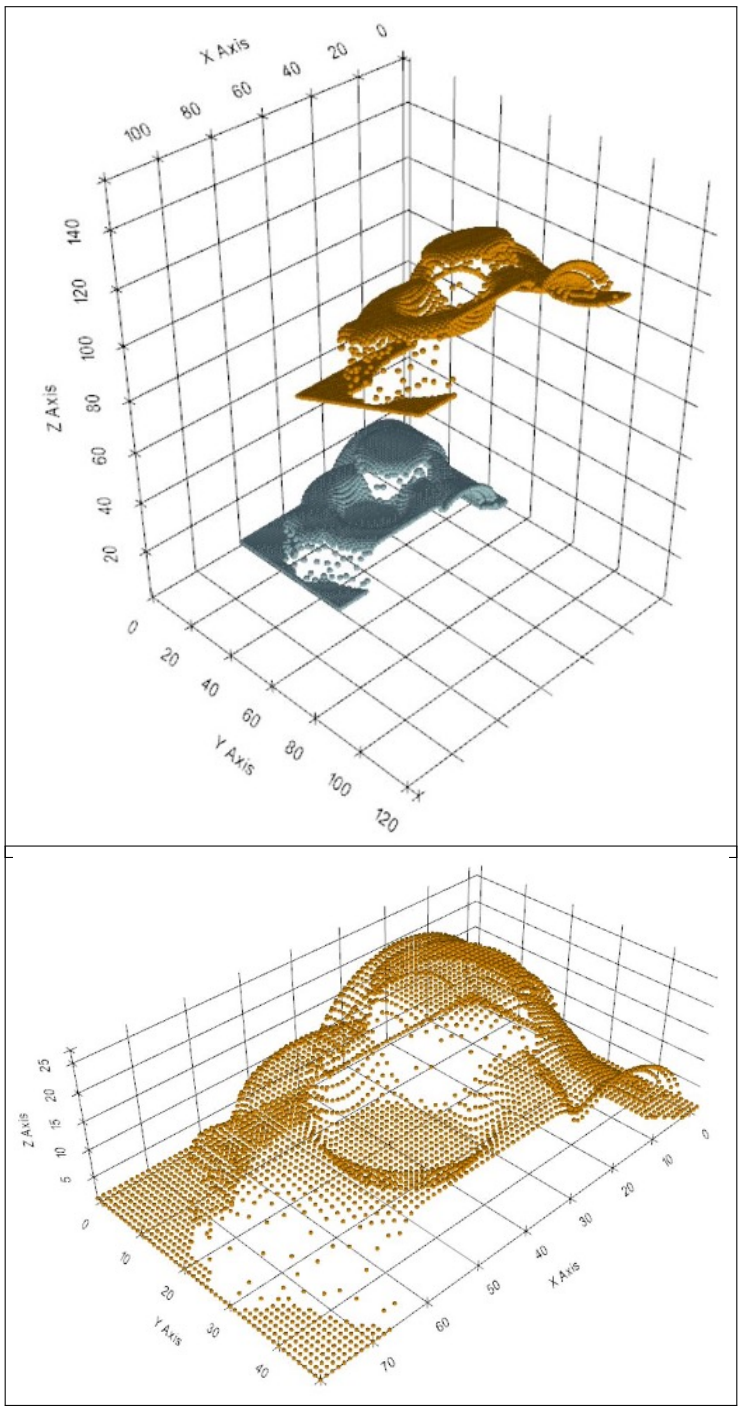

Abbildung 21: Graphische Darstellung des Mechpart-Modells vor (oben) und nach (unten) der ICP-Anwendung unter Verwendung einer Rotation vom 20 Grad um x-Achse zusammen mit einer Translation mit dem Vektor (40,80,120).

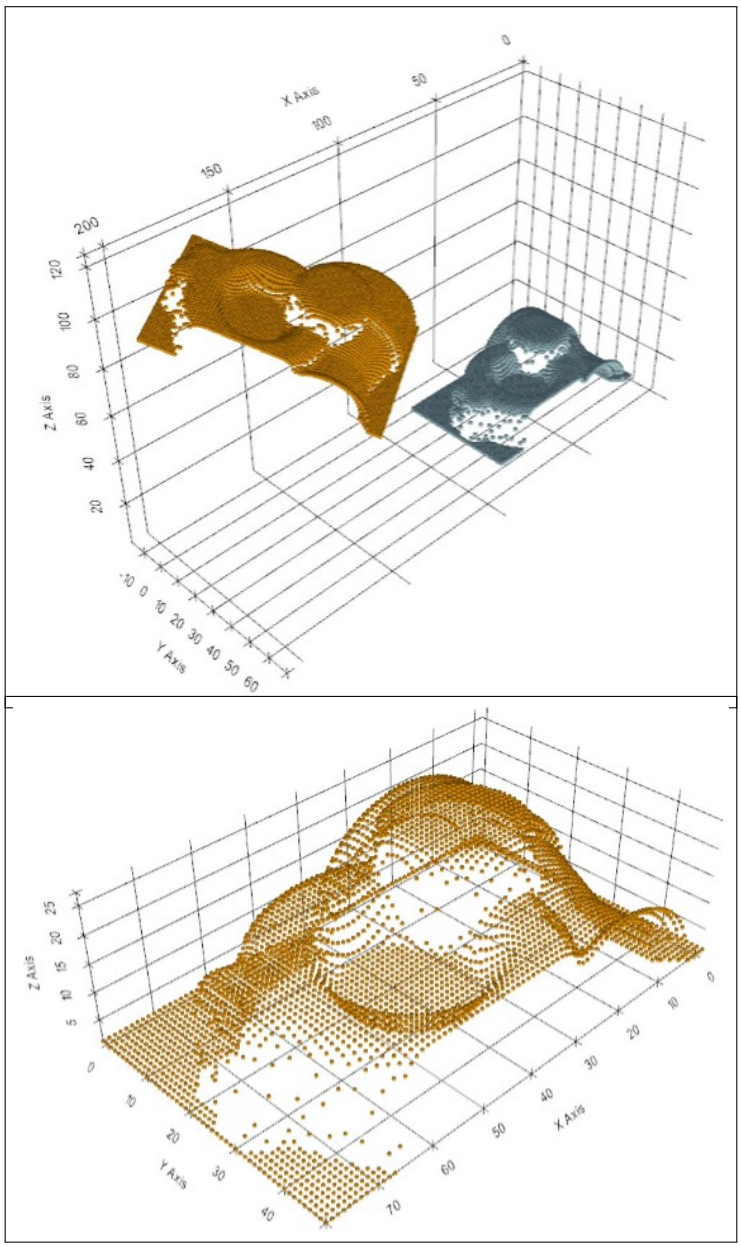

Abbildung 22: Graphische Darstellung des Mechpart-Modells vor (oben) und nach (unten) der ICP-Anwendung unter Verwendung von 3 Rotationen vom 100 um x- , 65 um y-, 274 um z-Achse zusammen mit einer Translation mit dem Vektor (124,43,85).

4.2.2 Match-Ergebnis zu M2

Analog zu Match-Ergebnis zu M2 im Fall des 2-dimensionalen Modells kann man sowohl die Anfangs- und Endwerte über alle Achsen, als auch die Step Values zwischen den definierten Anfangs- und Endwerten definieren. In der Abbildung 23 sind die Match-Ergebnisse für das Mechpart-Modell mit folgenden Eingaben visualisiert:

- Anfansgwerte der Rotationen des „Destination Modells“ : [0, 0, 0]
- Endwerte der Rotationen des „Destination Modells“: [30, 30, 30]
- Step Value: [10, 10, 10]

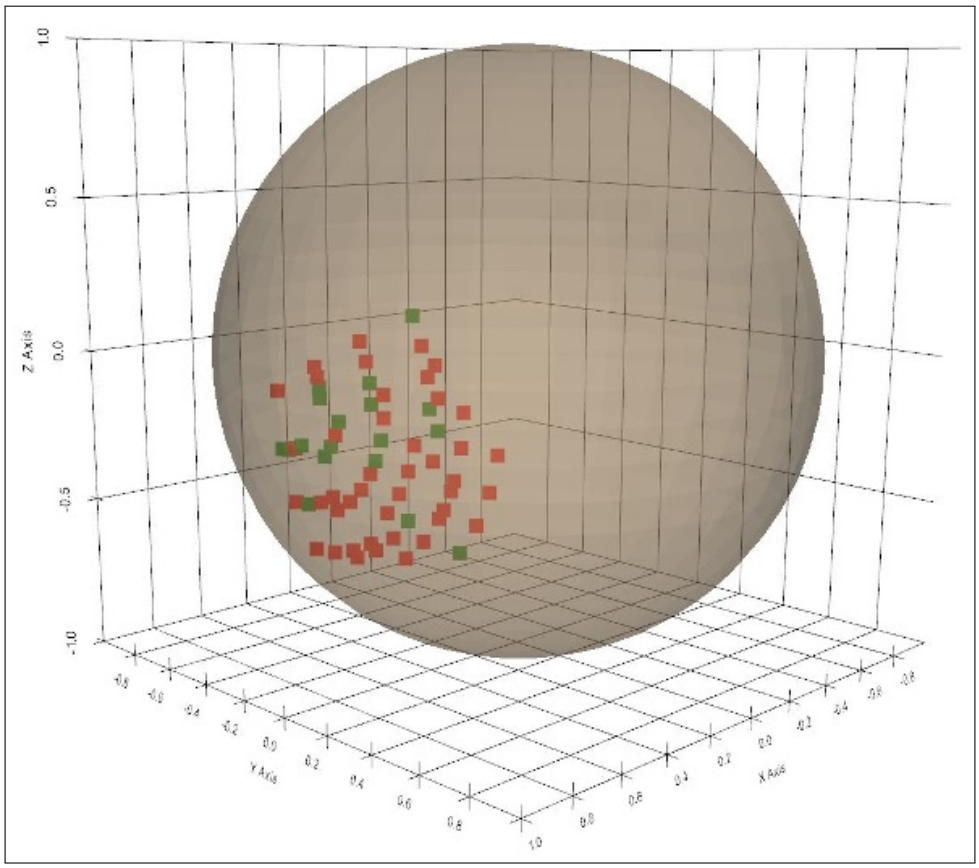

Abbildung 23: Graphische Darstellung des Mechpart-Modells für definierten Anfangs- und Endwerten der Rotation und den Step Value .

Um die Anzahl von den perfekten (> 98 %) und niedrigen (< 98 %) Match-Ergebnissen zu visualisieren sind verschiedene Histogramme plottet . 20 von den Ergebnissen sind erfolgreich und der Rest ist erfolglos wie das Histogramm in der Abbildung 24 zeigt.

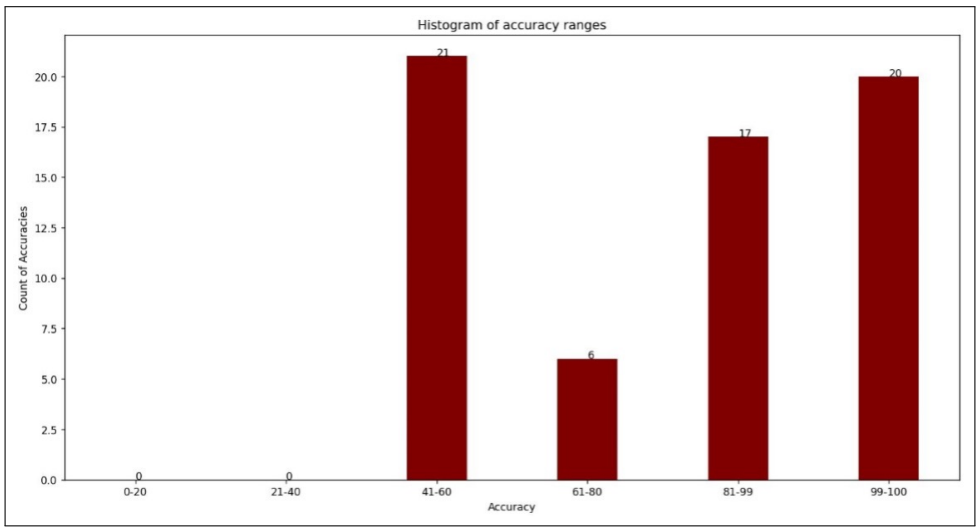

Abbildung 24: Histogramm des Mechpart-Modells für definierten Anfangs- und Endwerten der Rotation und den Step Value .

4.2.3 Match-Ergebnis zu M3

Wie in Abschnitt 4.1.3 durchgeführt , werden die Match-Ergebnisse der Versuche mit den Rotationen ([variable, 90, 0]) und mit dem Step Value = 10 zwischen den Rotationen um x-Achse von 0 bis 360 Grad und den konstanten Werten von den Rotationen über y- und z-Achsen statistisch auf der Kugeloberfläche skizziert. Die „option_include_every_nth_point“ ist in diesem Fall für beide Modelle (Source und Destination) gleich 10 gesetzt, um den Prozess zu beschleunigen, da das Mechpart-Modell ein 3 dimensionales Modell ist und aus mehreren Punkten besteht. Diese Konfiguration liefert die Ergebnisse, die in Abbildung 25 und 26 (als Histogramm) dargestellt sind.

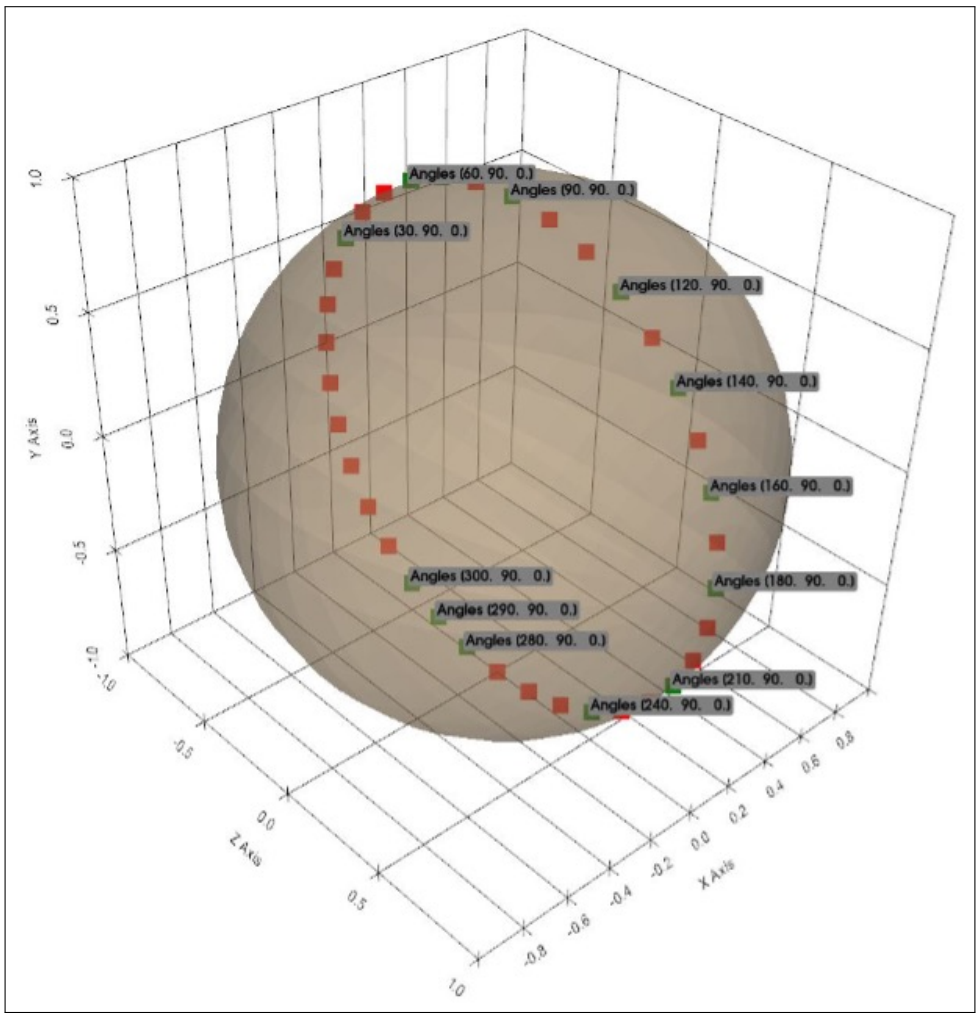

Abbildung 25: Graphische Darstellung des Mechpart-Modells für definierten Anfangs- und Endwerten der Rotation und den Step Value .

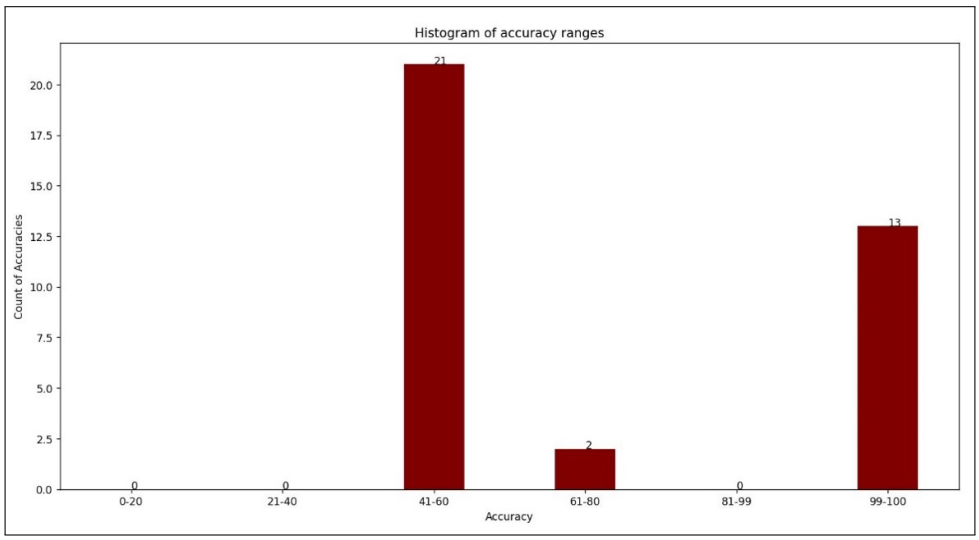

Abbildung 26: Histogramm des Mechpart-Modells für definierten Anfangs- und Endwerten der Rotation und den Step Value .

Mit dieser Konfiguration bekommt man 12 perfekte Match-Ergebnisse. Dieses Anzahl und Rotationswerte sind anders als die Anzahl und Match-Ergebnisse der gleichen Konfiguration bei dem Human-Noise-Modell.

Das Ergebnis aus der Abbildung 25 kann wie in dem Verfahren mit dem Human-Noise-Modell, unterschiedlich sein und mit verschiedenen variablen und konstanten Werten der Rotationswinkel des „Destination Modells“ um verschiedene Achsen in dem ICP-Algorithmus durchgeführt werden. Die Abbildung 27 zeigt die Match-Ergebnisse für die folgenden Rotationen mit grünen und roten Punkten auf der Kugeloberfläche.

- ([variable, 90, 0]) , ([0, variable, 0]), ([0, 0, variable]), ([45, 0, variable]), ([135, 0, variable])

Der Parameter variable gibt an, um welche Achse die Rotation durchgeführt wurde. Hier werden 5 Rotationen nacheinander angewendet.

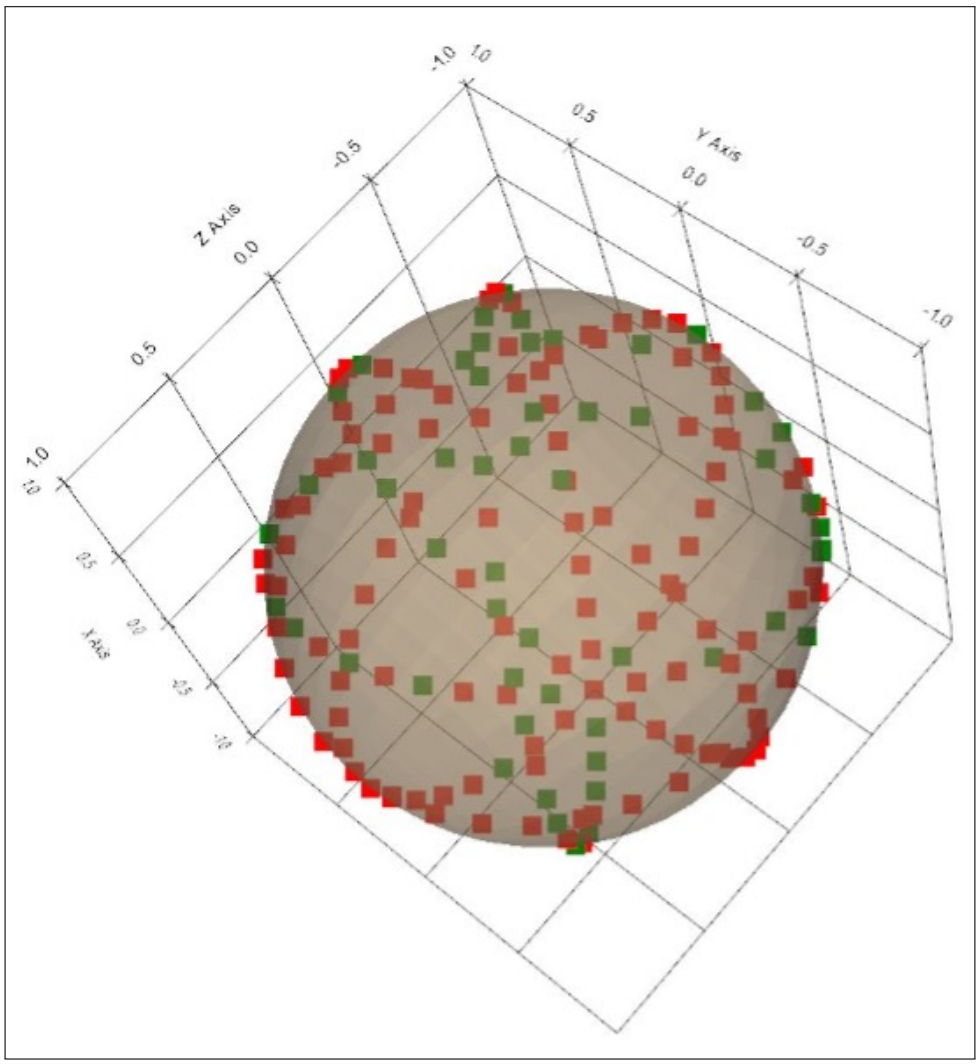

Abbildung 27: Graphische Darstellung des Mechpart-Modells für 5 variable Rotationen .

Die Labels können nach Bedarf des Nutzers mit „option_choose_show_labels_for_matches“ sowohl für alle Punkte, als auch für die matched-Punkte oder unmatched-Punkte visualisiert werden. Ein Histogram zu dieser Konfguration ist in der Abbildung 28 skizziert.

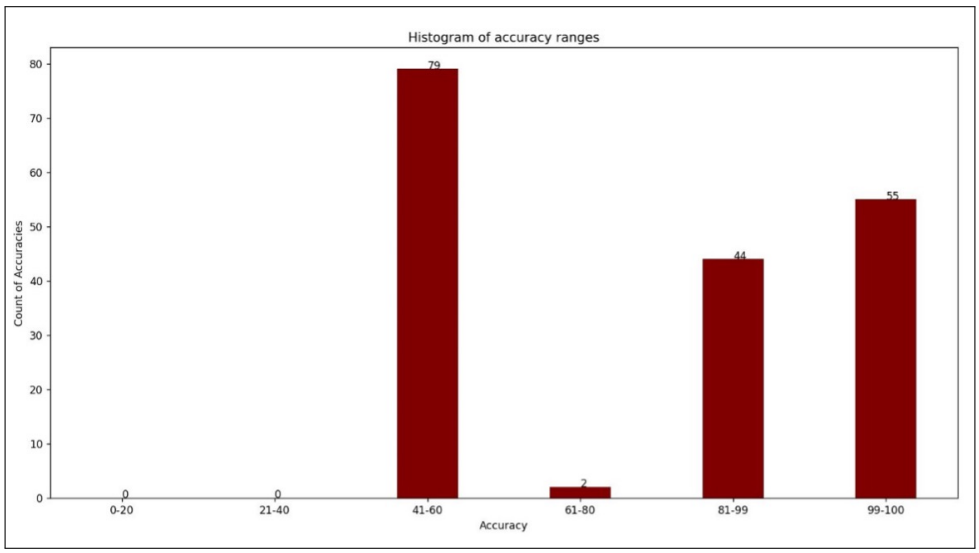

Abbildung 28: Histogramm des Mechpart-Modells für für 5 variable Rotationen.

4.3 Bewertung

Die Anzahl und Rotationswerte der perfekten Match-Ergebnisse der gleichen Verfahren (aus M1 und M2) sind für verschiedene Modelle unterschiedlich. Deswegen ist es immer sinnvoll, um die Modelle mit verschiedenen Verfahren und Konfigurationen (Parameter) zu bearbeiten. Mit den iterativen Verfahren der M2 und M3 kann man die Rotationswerte herausfinden, nach denen ein perfektes Match-Ergebnis möglich ist.

Falls das Human-Noise-Modell sich mit Rotationswerten 90 und 0 je um y- und z-Achse nur um die x-Achse zwischen 70 und 180 Grad, zwischen 190 und 250 Grad und zwischen 250 und 330 Grad rotiert, kann man kein pefektes Match-Ergebnis sehen. Dies ist für das Mechpart-Modell zwischen 240 und 280 Grad und dann wieder zwischen 300 und 30. Mit anderen Worten, die genannten Rotationsintervalle, sind die Bereiche die bei der Konfiguration vermeiden muss, um kein niedriges Match-Ergebnis zu sehen.

5 Fazit und Ausblick

Das Problem der Registrierung von Punktwolken ist ein sehr aktives Forschungsgebiet. Zur Bewältigung dieses Problems hat der ICP-Algorithmus seine Robustheit und Genauigkeit bewiesen. Mit dem ICP-Algorithmus ist es notwendig, gute Ausgangswerte zu haben und optimale Konfiguration in Abhängigkeit von einstellbaren Parametern auszusuchen, um beste Match-Ergebnisse zu erzielen.

In dieser Arbeit wurde ein ICP-Algorithmus basierend auf der Implementierung in [7] vorgestellt und optimiert. Es wurde die Richtigkeit des vorgestellten ICP-Algorithmus und die Visualisierung der erwarteten Outputs erfolgreich geprüft. Der ICP-Algorithmus wurde auf zwei verschiedenen Polygon-Modelle, ein 2D- und 3D Polygon-Modell angewendet. Das 2D-Modell, namens Human-Noise-Modell besteht aus 120 Punkten, die mit mit x-, y-, und z-Koordinaten präsentiert sind, wobei alle z-Koordinaten gleich Null sind. Das 3D-Model, namens Mechpart-Modell, besteht aus 4100 Vertices, die jeweils als (x,y,z) Floats-Zahlentripel gegeben sind und einem Objekt mit 7938 Polygon Faces.

In den Experimenten zeigen die Ergebnisse, dass dieser Algorithmus in der Lage ist, eine robuste und genaue Transformationen als Kombination von Rotationen und Translationen zu finden. Die Genauigkeit wurde mit zwei oben genannten Modelle in Abhängigkeit der ausgesuchten Konfigurationen getestet und bewertet.

Die aktuellen ICP-Algorithmen, wie beispielsweise PCRNet [22], DeepMatch [21] und Deep ClosestPoint [28], basieren auf Deep-Learning-Techniken. Sie können für verschiedene Anwendungen eingesetzt werden. Daher lohnt es sich auch zu untersuchen, wie die Genauigkeit und die Robustheit dieser Algorithmen mit unseren zwei Polygon-Modelle aussieht, um einen Vergleich mit unserem optimierten ICP-Algorithmus durchzuführen.

Im Rahmen dieser Arbeit konnten viele Erfahrungen auf dem Gebiet der Registrierung von Punktwolken mit ICP-Algorithmen erworben werden. Neben dem theoretischen Hintergrund ergibt sich daraus ein Einstiegspunkt in das Thema, der mithilfe vom angehängten Python-Skript noch festgestellt werden kann. Die Ergebnisse aus Kapitel 4 sollen nicht nur als Anreiz dienen, leistungsstarke ICP-Algorithmen zu entwickeln, sondern auch zukünftige Forschungsaktivitäten im Bereich der Registrierung von Punktwolken zu erleichtern.