

■ fakultät für informatik

Bachelor-Arbeit

Deep Learning-gestützte
Ausrichtung von räumlichen
Entitäten für den 5D-Druck

Lars Naumann

31. Dezember 2025

Gutachter:

PD Dr. Frank Weichert

Prof. Dr. Heinrich Müller

Lehrstuhl Informatik VII
Computergraphik
TU Dortmund

Inhaltsverzeichnis

Mathematische Notation	e
1. Einleitung	1
1.1. Motivation und Hintergrund	1
1.2. Aufbau der Arbeit	1
2. Topologieaspekte in der Additive Fertigung	3
2.1. Überhang-Problem	4
2.2. Fertigungs-Geschwindigkeit	5
2.3. 5d-Druck	6
3. Flächensuche in Bauteilen	9
3.1. Clustering nach Normalvektoren in Bauteilen	9
3.2. Separation von parallelen Flächen	12
4. Punktwolkenverarbeitung zur Bauteilausrichtung	15
4.1. Punktwolken Transformation	15
4.1.1. Merkmalsextraktion aus Punktwolke	17
4.1.2. Regression auf Merkmalsvektor	19
4.2. Konstruktion der Ausrichtung	20
5. Evaluation	25
5.1. Ausrichtung der Bauteile	26
5.2. Minimierung der Bewegung	26
5.3. Verlässlichkeit der Ausrichtung	26
5.3.1. Effekt von Rauschen auf den Daten	26
5.3.2. Effekt von schlechter Flächensuche	27
5.4. Generalisierbarkeit	28
6. Fazit	31
6.1. Fazit	31

6.2. Ausblick	32
A. Weitere Informationen	35
Abbildungsverzeichnis	37
Algorithmenverzeichnis	39
Quellcodeverzeichnis	41

Mathematische Notation

Notation	Bedeutung
\mathbb{N}	Menge der natürlichen Zahlen $1, 2, 3, \dots$
\mathbb{R}	Menge der reellen Zahlen
\mathbb{R}^d	d -dimensionaler Raum
$\mathcal{M} = \{m_1, \dots, m_N\}$	ungeordnete Menge \mathcal{M} von N Elementen m_i
$\mathcal{M} = \langle m_1, \dots, m_N \rangle$	geordnete Menge \mathcal{M} von N Elementen m_i
\mathbf{v}	Vektor $\mathbf{v} = (v_1, \dots, v_n)^T$ mit N Elementen v_i
$v_i^{(j)}$	i -tes Element des j -ten Vektors
\mathbf{A}	Matrix \mathbf{A} mit Einträgen $a_{i,j}$
$G = (V, E)$	Graph G mit Knotenmenge V und Kantenmenge E

1. Einleitung

1.1. Motivation und Hintergrund

Die additive Fertigung ist ein Feld mit viel Potenzial. Es ist eine Technologie, die in der Fertigung immer wichtiger wird. Schnelle Produktion von Prototypen und Ersatzteilen sind nur einige der Anwendungsbereiche von dem 3d-Druck. Allerdings gibt es noch Probleme, zum Beispiel mit der strukturellen Integrität von Bauteilen oder der Materialverschwendungen durch Stützstrukturen. Der 5d-Druck bietet eine Möglichkeit beide Probleme zu adressieren, indem das Bauteil während des Drucks gedreht wird. Dadurch ist es im 3d-Druck möglich Ausrichtungen für einzelne Flächen eines Bauteils zu wählen. Um eine optimale Ausrichtung zu finden, gibt es bis jetzt noch keine weit anerkannte automatische Lösung. Eine Möglichkeit ist die Verwendung von neuronalen Netzen, um eine optimale Ausrichtung zu schätzen.

Neurale Netze sind gut geeignet, Muster in Daten zu erkennen. Das Gebiet des maschinellen Lernens hat sich bereits in vielen Anwendungsfällen als verlässliche Lösung erwiesen. In der Bildverarbeitung werden sie zum Beispiel verwendet, um Objekte in Bildern zu erkennen. Es ist daher naheliegend, dass sie auch verwendet werden können, um Muster in einer Menge an Raumpunkten zu erkennen. Ein Neurales Netz kann gut über Daten abstrahieren und auch für unbekannte komplexe Daten eine gute Lösung finden, wo klassische Algorithmen versagen könnten. Auch für die Ausrichtung von Bauteilen haben sich neurale Netze bereits als gute Lösung erwiesen. In dem Artikel [test] wurde ein tiefes neurales Netz genutzt, um geometrische Objekte anhand von vorgegebenen künstlichen Zielen auszurichten. Eine Erweiterung dieser Vorgehensweise könnte für die Ausrichtung von Bauteilen im 5d-Druck genutzt werden.

1.2. Aufbau der Arbeit

In dieser Arbeit wird versucht das Problem der Ausrichtung von Bauteilen im 5d-Druck durch die Verwendung von überwachtem Lernen zu lösen. Begonnen wird

mit der Aufteilung eines Bauteils in kohärente Flächen, wofür mehrere klassische Clusteringverfahren genutzt werden. Anschließend wird ein Neurales Netz trainiert, dass die Flächen eines Bauteils auf Rotationen abbildet, die das Bauteil optimal für den 5d-Druck ausrichten. Für das Neurale Netz wird eine angepasste Version von einem Bereits existierenden Neuralem Netz für die Merkmalsextraktion verwendet [test]. Zum Schluss wird die Leistung des neuronalen Netzes evaluiert, anhand der Ausrichtung auf unterschiedlichen Objekten und der Widerstandsfähigkeit gegen Störungen.

2. Topologieaspekte in der Additive Fertigung

Der 3d-Druck ist eine Methode der Fertigung. Es wird auch additive Fertigung genannt, da das Bauteil durch das Hinzufügen von Material Schicht für Schicht erstellt wird. Um eine Bauteil 3d-Drucken zu können, wird das Bauteil im Computer in mehrere Schichten aufgeteilt. Diese Schichten werden dann nacheinander gedruckt, bis das Bauteil fertig ist. Die Schichten werden mit einer Drüse aufgetragen und gestehen meistens aus Kunststoff. Es ist aber auch möglich, andere Materialien zu verwenden, wie zum Beispiel Metall oder Beton. Das Material muss flüssig sein, wenn es aufgetragen wird, festigt sich dann aber schnell wieder.

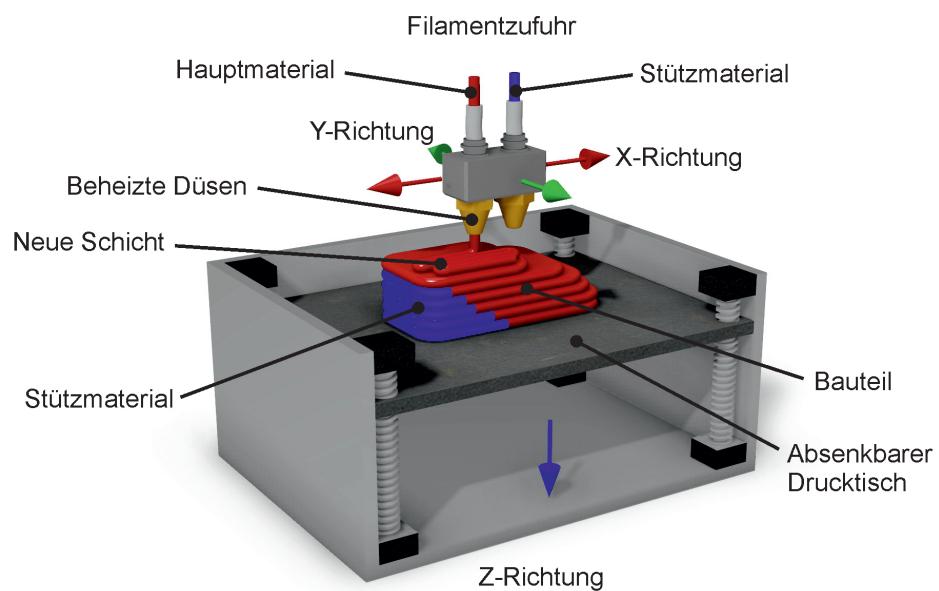


Abbildung 2.1.: Schematische Darstellung eines 3d-Druckers
Quelle: <https://www.linux-magazin.de/wp-content/uploads/2020/11/b01.jpg>

In der Abbildung 2.1 ist eine schematische Darstellung eines 3d-Druckers zu sehen. Dabei ist oben die Düse, welche das Material erst schmilzt und dann aufträgt. Die Düse ist lässt sich in x und y Richtung bewegen und kann so jede beliebige Position

auf der Plattform erreichen. Unten kann man die Platform sehen, auf der das Bauteil gedruckt wird. Sie kann sich nach unten bewegen, damit Platz für die nächste Schicht geschaffen wird. Die Funktion des Stützmaterials und der Stützschichten wird in Abschnitt 2.1 erklärt. Der in Abbildung 2.1 zusehen 3d Drucker ist ein Beispiel für den Aufbau eines 3d-Druckers. Es gibt 3d drucker, die anders funktionieren, aber das Grundprinzip ist bei allen gleich. Alle haben eine Düse, die das Material ausgibt und diese Düse bewegt sich, um Schichten abzuarbeiten.

2.1. Überhang-Problem

Im 3d-Druck werden Bauteile traditionell Schicht für Schicht gebaut. Das Bauteil wird vor dem Druck in dünne horizontale Schichten zerlegt. Jede Schicht wird dann nacheinander gedruckt. Dabei kann es zu Problemen kommen, wenn eine Schicht über einer vorherigen Schicht hinausragt, ohne dass darunter Material ist. Dies wird als Überhang-Problem bezeichnet. Um diesem Problem entgegenzuwirken werden Stützstrukturen eingesetzt.

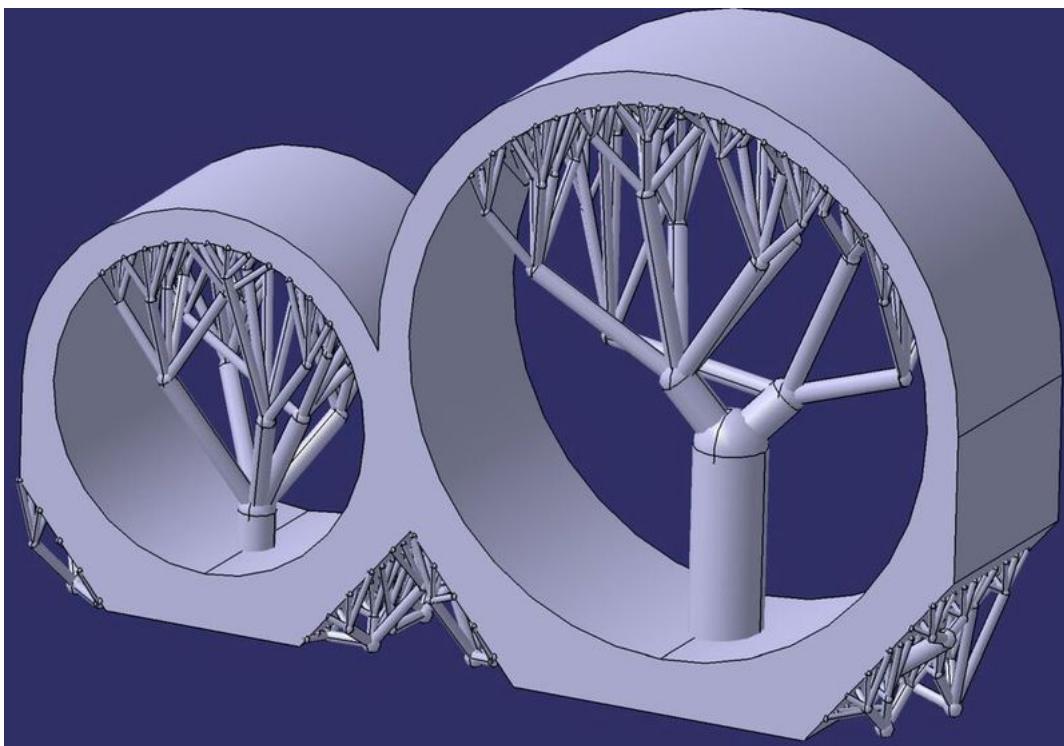


Abbildung 2.2.: Vorschau eines Bauteils mit Stützstrukturen
Quelle: <https://i.stack.imgur.com/cIDB3.png>

In der Abbildung 2.2 ist ein Bauteil zu sehen, das Stützstrukturen benötigt. Stützstrukturen erhöhen den Materialaufwand, wie man in Abbildung 2.2 sehen kann.

Die Strukturen werden nach dem Druck entfernt, was das zusätzlich benötigte Material zu Müll macht. Durch das Hinzufügen von Stützstrukturen verlängert sich auch die Druckzeit. Dieses Problem kann durch die richtige Ausrichtung des Bauteils minimiert werden, aber bei komplexen Bauteilen ist es unwahrscheinlich, dass eine Ausrichtung gefunden wird, die keine oder wenige Stützstrukturen benötigt.

2.2. Fertigungs-Geschwindigkeit

Wie bei jeder Art der Fertigung ist die Produktionsgeschwindigkeit ein wichtiger Faktor. In der additiven Fertigung wird die Produktionsgeschwindigkeit durch verschiedene Faktoren beeinflusst. Eine dieser Faktoren wird in Abschnitt 2.1 beschrieben. Das Hinzufügen von Stützstrukturen erhöht den Druckzeit, da mehr Material gedruckt werden muss. Das ist ein weiterer Grund warum es wichtig ist Stützstrukturen zu minimieren. Dementsprechend ist die Ausrichtung eines Bauteils wichtig, da eine gute Ausrichtung die benötigten Stützstrukturen minimieren kann.

Der offensichtlichste Faktor ist die Größe des Bauteils und die Druckgeschwindigkeit des Druckers. Ein größeres Bauteil benötigt mehr Zeit, wenn es mit der gleichen Geschwindigkeit gedruckt wird. Die Druckgeschwindigkeit ist abhängig vom Drucker. Unabhängig von dem Drucker gibt es physikalische Grenzen für die Druckgeschwindigkeit. Wenn eine bestimmte Geschwindigkeit überschritten wird, wird es zu Problemen mit der Genauigkeit und Qualität des Drucks kommen. Die Geschwindigkeit mit der eine Düse Material ausgeben kann ist ebenfalls ein limitierender Faktor.

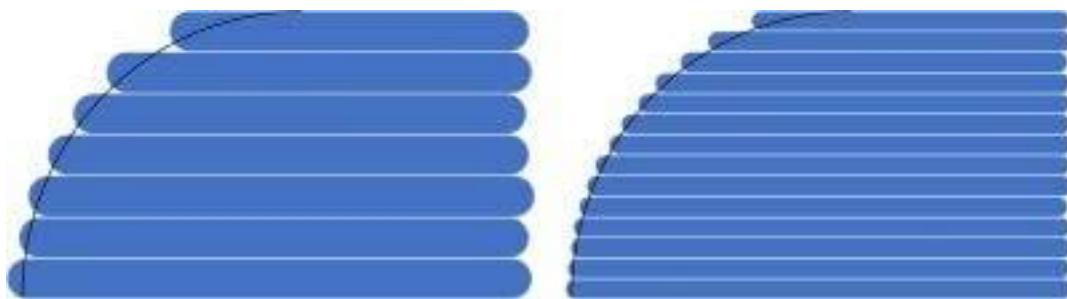


Abbildung 2.3.: Darstellung der Unterschiede im Bauteil abhängig von der Höhe der Schichten

Quelle: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS6yH0dj1g06mhqwt_JJ22BE7VaI-0V4U4SWg&s

Ein weiterer Faktor ist die Höhe der einzelnen Schichten. In der Abbildung 2.3 ist der Unterschied, den die Höhe der Schichten auf die Qualität des Bauteils hat, zu sehen. Umso dünner die Schicht umso genauer wird das Bauteil gedruckt. Wenn die Dicke

der Schichten verringert wird, werden mehr Schichten benötigt, was die Strecke, die die Düse abarbeiten muss, erhöht. Dadurch verlängert sich die Druckzeit.

2.3. 5d-Druck

Der 5d-Druck ist eine Erweiterung des 3d-Drucks. Ein 3d-Drucker kann die Düse in der x und y Richtung und die Platform in der z Richtung bewegen. Ein 5d-Drucker kann zusätzlich die Platform in zwei Dimensionen kippen.

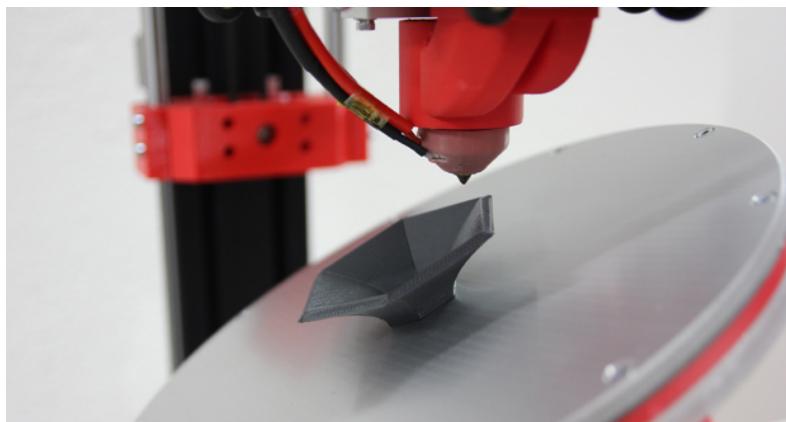


Abbildung 2.4.: Darstellung der Beweglichen Platform eines 5d-Druckers
Quelle: https://www.3dnatives.com/de/wp-content/uploads/sites/3/150217_Zurich1.jpg

In der Abbildung 2.4 ist eine Düse und eine Platform zu sehen. Die Platform ist sichlich gekippt. Die Fähigkeit die Platform zu kippen ermöglicht es Bauteile zu drucken, ohne Stützstrukturen zu benötigen, da wenn eine Schicht am Überhängen ist, das Bauteil gedreht werden kann, sodass die nächste Schicht nicht mehr über die aktuelle Schicht überhängt. Durch das Kippen der Platform ist es ebefalls möglich Schichten, die nicht parallel zu der Platform sind zu drucken. Die besere Anpassung an die Geometrie der Bauteile verbessert die Festigkeit des Bauteils.

Diese Fähigkeit Schichten zu erstellen, die nicht parallel zu der Platform sind, hat den Nachteil, dass die Berechnung der Schichten komplexer wird. Die meisten Programme zu der Erstellung von Schichten unterstützen keinen 5d-Druck. Herausforderung bei der Erstellung von Schichten für den 5d-Druck sind unter anderem die Kolisionsvermeidung und die Berechnung der optimalen Kippwinkel für jede Schicht. Die Kolisionsvermeidung ist ein Problem, das durch den 5d-Druck entsteht. Wenn das Bauteil sich während des Druckes drehen kann, besteht die Möglichkeit, dass die Düse mit dem Bauteil kollidiert. Bei dem 3d-Druck ist das nicht möglich, da die Düse immer über dem Druckbereich ist.

Optimale Kippwinkel für jede Schicht zu bestimmen benötigt die Berechnung des optimalen Winkels für jede Schicht. Dieser Prozess wird schwerer gemacht, da berücksichtigt werden muss, dass eine Fläche unter Umständen nicht von einem bestimmten Winkel aus erreichbar sein könnte, weil dort bereits etwas anderes gedruckt wurde, was mit der Düse kollidieren würde. Es muss also auch bei der Ausrichtung des Bauteils Kolisionsvermeidung berücksichtigt werden.

3. Flächensuche in Bauteilen

Bevor ein Bauteil ausgerichtet werden kann, müssen zuerst kohärente Flächen im Bauteil identifiziert werden. Die Verarbeitung dieser Flächen wird in Kapitel 4 behandelt.

Dieses Kapitel erklärt, wie in einem Bauteil kohärente Flächen gefunden werden. Dafür wird ein Bauteil durch zufällig auf der Oberfläche verteilte Punkte $p \in \mathbb{R}^3$ dargestellt. Eine Menge \mathbb{R}^{3xN} dieser Punkte wird Punktfolge genannt. Es werden mehrere tausend Punkte benutzt, um ein Bauteil mit einer Punktfolge darzustellen. Danach wird die Punktfolge in mehrere kleinere Gruppen aufgeteilt.

Eine Aufteilung von Datenpunkten in Gruppen wird Clustering genannt. Die Aufteilung in Gruppen basiert auf den Werten der Datenpunkte, Daten mit ähnlichen Werten kommen in dieselbe Gruppe. Die Gruppen werden auch Cluster genannt. Was Ähnlichkeit bedeutet kommt auf die Implementierung an, aber ein häufiges Ähnlichkeitsmaß ist die räumliche Distanz.

In der Abbildung 3.1 sind zwei zweidimensionale Punktfolgen. Die linke ist nicht geclustert, die rechte ist geclustert, beide stellen dieselbe Punktfolge dar. Die rechte Punktfolge ist mit drei Farben markiert, Punkte derselben Farbe sind im selben Cluster. In der Abbildung 3.1 ist das Ergebnis von einem Clusteringverfahren dargestellt, alle Punkte in einer Gruppe sind raumlich gesehen ähnlich zueinander.

3.1. Clustering nach Normalvektoren in Bauteilen

Um zu messen, ob zwei Punkte Teil derselben Fläche sein sollen, muss die Ausrichtung eines Punktes gemessen werden. Punkte mit einer ähnlichen Ausrichtung gehören zu derselben Fläche. Mit der Ausrichtung eines Punktes ist ein Vektor, der von der Oberfläche des Bauteils an, genau der Position eines Punktes, senkrecht nach oben geht gemeint. Solche Vektoren werden Normalvektor genannt.

In der Abbildung 3.2 ist die Bauteilloberfläche blau markiert. Der Punkt p_i ist ein beliebiger Punkt auf der Bauteiloberfläche. Der Normalvektor n_i geht von dem Punkt p_i aus senkrecht nach oben. Er ist im 90 Grad Winkel zu der Bauteiloberfläche. Die

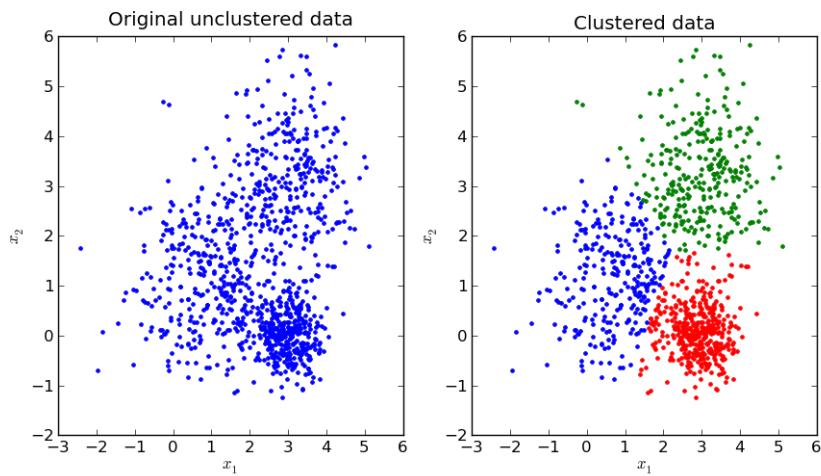


Abbildung 3.1.: Clustering von versteckten zweidimensionalen Punkten
Quelle: <https://i.stack.imgur.com/cIDB3.png>

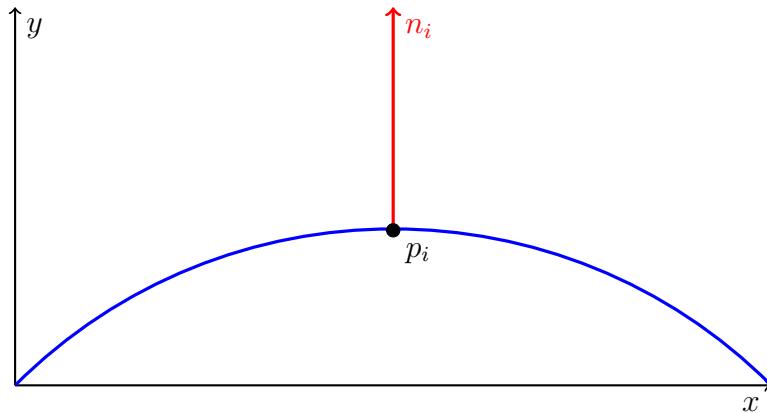


Abbildung 3.2.: Darstellung von Normalvektoren an einem Punkt der Bauteilloberfläche

Normalvektoren in der Abbildung 3.2 sind zweidimensionale, im Gegensatz dazu sind die Normalvektoren auf den Punkten im Bauteil dreidimensional. Die Normalvektoren werden für jeden Punkt anhand der Oberfläche des Bauteils bestimmt.

Eine Aufteilung der Normalvektoren in Cluster ist notwendig, um Flächen zu erkennen. Dafür werden die Normalvektoren als dreidimensionale Punkte dargestellt, die euklidische Distanz zwischen den Normalvektoren gibt ihre Ähnlichkeit an. Für das Aufteilen in Cluster wird ein Algorithmus benutzt, der mit einem Cluster startet, dass alle Normalvektoren beinhaltet und über mehrere Iterationen die Gruppe immer wieder aufteilt, bis jede Gruppe eine voreingestellte Ähnlichkeit aufweist.

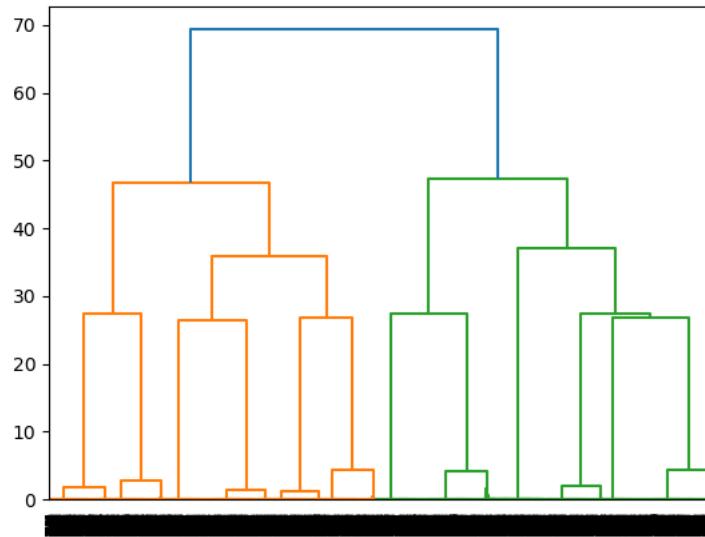


Abbildung 3.3.: Beispiel von Hierachie Clustering als Dendrogram

Die Abbildung 3.3 stellt das Vorgehen von Hierarchie Clustering dar. Dabei ist die y-Achse der Distanzwert und auf der x-Achse sind die Cluster markiert. Das Clustering fängt unten, bei $y = 0$ an. Dort ist jeder Punkt ein Cluster. Danach wird die Distanz zwischen allen Clustern gemessen und die zwei ähnlichen Cluster werden zusammen genommen zu einem Cluster. Dieses Vorgehen wird so lange wiederholt, bis ein bestimmter Distanzwert zwischen allen Clustern überschritten wurde. In der Abbildung 3.3 kann man das Zusammenfügen von Clustern sehen, wenn zwei über eine senkrechte Linie verbunden werden. Aus der Linie, welche die beiden Cluster verbindet, kommt eine weitere Linie, die das neue zusammengefügte Cluster symbolisiert. Das hat den Vorteil, dass die Anzahl der Cluster in einem Bauteil von dem Bauteil selber abhängt. Es ist nicht nötig eine feste Anzahl an Clustern festzulegen, sondern nur eine Ähnlichkeit, welche die Cluster haben sollen. In der Abbildung 3.3 stoppt der Algorithmus nicht, nachdem alle Cluster einen vorbestimmten Distanzwert überschreiten, sondern fügt so lange Cluster zusammen, bis nur noch eins übrig ist.

Auf diese Art und Weise werden die Cluster zwischen Normalvektoren gesucht. Alle Normalvektoren, die in einem Cluster sind, sind so ähnlich, dass das hierarchisches Clustering sie nicht weiter zusammengefügt hat. Der Distanzwert auf der y-Achse in 3.3 dient dabei als Fehlermaß für die Flächen. Wenn das Fehlermaß null ist, werden nur Flächen gefunden, die absolut flach sind, aber bei komplexen Bauteilen ist es notwendig ein höherer Fehler zu zulassen, um gebogene Flächen zu finden.

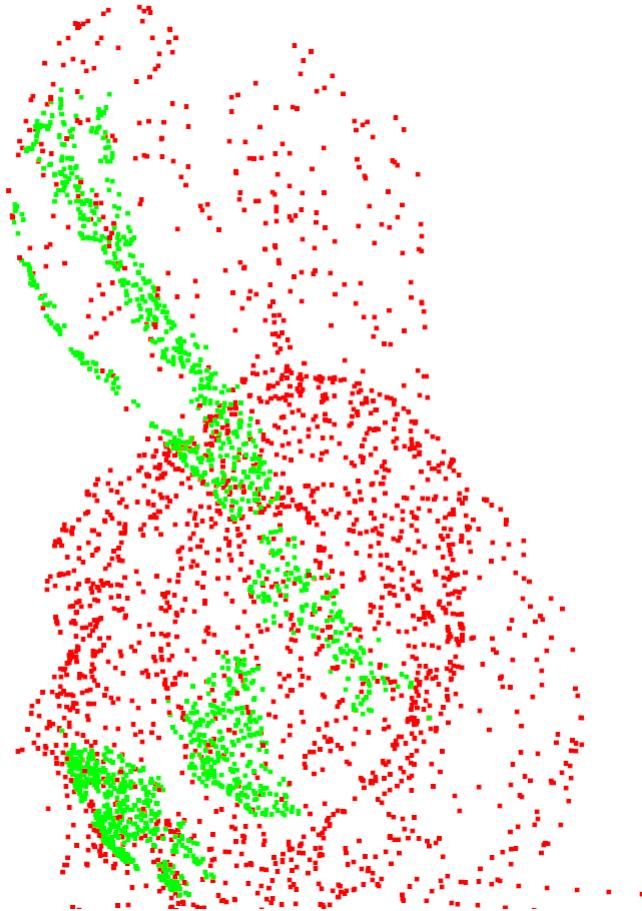


Abbildung 3.4.: Hierarchisches Clustreing eines Bauteils nach Normalvector

Das Clustreing nach Normalvectoren ist nur geeignet um Flächen zu finden, die, innerhalb des Distanzwertes als Fehlermaß, Flach sind. Das Clustering gibt keine Garantie, dass die gefundenen Flächen kohärent sind. In Abbildung 3.4 ist sichtbar, dass die in grün markierten Flächen, parallel zueinander, aber nicht kohärent miteinander sind. Die Flächen die in Grün markiert sind, sind Teil desselben Clusters, aber sollten es nicht sein.

Der nächste Abschnitt 3.2 beschäftigt sich damit wie die in den hierarchischen Clustering gefundenen Flächen getrennt werden.

3.2. Separation von parallelen Flächen

Alle Flächen in einem Cluster des Normalvektorclusterings aus Abschnitt 3.1 sind in etwa parallel, aber räumlich voneinander getrennt. Das macht sie ungeeignet für die Bildung einer Ausrichtung in Kapitel 4.

Die räumliche Trennung zwischen den Flächen in einem Cluster kann genutzt werden, um sie zu trennen. Dafür wird ein weiterer Clustering-Algorithmus genutzt. Der Algorithmus benötigt einen Radius und eine Mindestanzahl an Nachbarn. Der Radius wird esp und die Mindestanzahl an Nachbarn wird minPts in Abbildung 3.5 genannt.

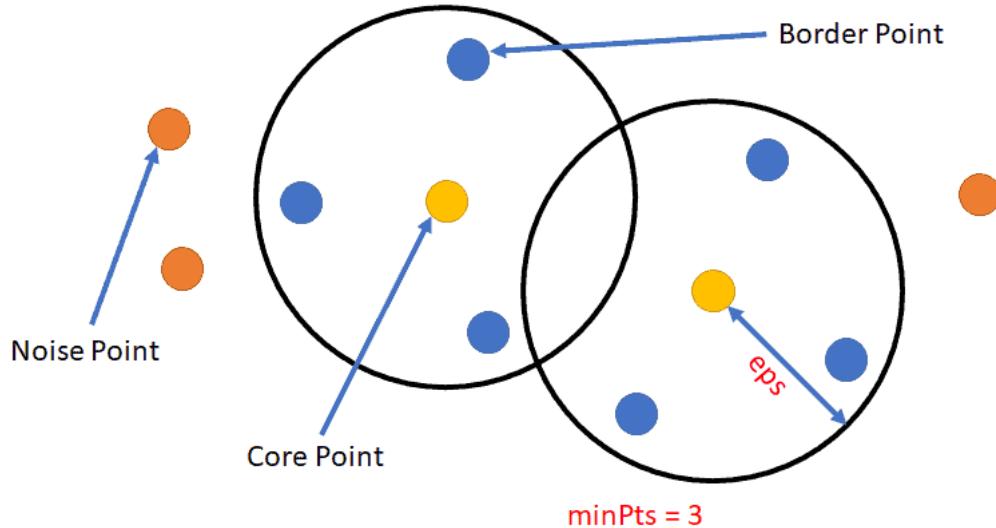


Abbildung 3.5.: Beispielhafte Darstellung von DB-Scan Algorithmus
Quelle: <https://machinelearninggeek.com/wp-content/uploads/2020/10/image-58.png>

Der Algorithmus nimmt einen Punkt, der in keinem Cluster ist und überprüft, wie viel Punkte in einem Radius esp um ihn herum sind. Wenn innerhalb des Radiuses mindestens minPts viele Punkte sind, dann ist der Punkt ein Kernpunkt (Core Point) eines neuen Clusters. Alle Punkte, die in dem Radius des Punktes sind, werden Teil des Clusters und genauso überprüft, wie der Startknoten des Clusters. Dieses Vorgehen wird fortgeführt, bis keine Punkte mehr hinzugefügt werden können. Wenn ein Punkt Teil des Clusters ist, aber nicht mindestens minPts viele Punkte in seiner Nachbarschaft hat, dann ist dieser Punkt ein Randpunkt (Borderpoint), aber immer noch Teil des Clusters. Wenn weniger als minPts innerhalb des Radius sind, dann gilt der Punkt als Noise. Das bedeutet aber nicht unbedingt, dass der Punkt noise bleibt. Er kann auch zu einem Randpunkt eines noch nicht existierenden Clusters werden. Es werden so lange neue Punkte außerhalb von Clustern gewählt, bis alle Punkte Teil eines Clusters oder Noise sind.

Alle Flächen innerhalb eines Clusters der Normalvectoren sind räumlich getrennt. Wenn der Algorithmus auf die vorher gefundenen Cluster angewandt wird, findet er kohärente Flächen als Cluster: In diesen Flächen liegen die Punkte sehr nahe beieinander, wodurch die Punkte in der Fläche meistens die minPts Bedingung erfüllen.

Andere Flächen werden wahrscheinlich nicht innerhalb von *esp* der Randpunkte einer Fläche liegen. Das bedeutet, es gibt keine Garantie, dass alle Flächen getrennt werden, aber in den meisten Fällen wird es passieren.

Zuletzt werden von den finalen Clustern alle aussortiert, die eine bestimmte Punktzahl unterschreiten. Flächen mit zu wenigen Punkten würden schwer zu verarbeiten sein, daher ist es einfacher sie nicht zu betrachten.

4. Punktwolkenverarbeitung zur Bauteilausrichtung

Um ein Bauteil korrekt auszurichten, muss eine Fläche des Bauteils auf eine Ausrichtung abgebildet werden. Die Flächen in den Bauteilen wurden in Kapitel 3 identifiziert und als Punktwolken gespeichert. Punktwolken wurden im Kapitel 3 bereits beschieben. Eine Punktwolke ist eine Menge an Punkten im 3D Raum, die zusammen eine Fläche oder ein Objekt darstellen.

In diesem Kapitel wird dargestellt, wie man eine Punktwolke, die eine Fläche in einem Bauteil darstellt, zu einer Rotation transformieren kann. Für die Transformation wird ein Neurales Netz eingesetzt, welches im Folgenden genauer erklärt wird. Dieses Neurale Netz wird eine Punktwolke beliebiger Größe zu der Z-Invarianz transformieren. Die Z-Invarianz wird in einem folgenden Abschnitt 4.2 erklärt. Aus dieser Ausgabe lässt sich dann eine vollständige Drehung erstellen.

4.1. Punktwolken Transformation

Die Transformation der Punktwolken erfolgt über ein Neurales Netz. Ein Neurales Netz kann Daten verarbeiten, indem es sie durch viele Schichten an künstlichen Neuronen durchleitet. Ein künstliches Neuron ist eine Funktion, die als Eingabe andere künstliche Neuronen oder echte Daten nimmt. Die Eingabe des Neurons besteht aus den einzelnen Eingaben (x_i) multipliziert mit den Gewichten für die einzelnen Eingaben (w_i).

Diese Funktion bildet die gewichteten Eingaben dann auf eine Ausgabe ab, die entweder an ein weiteres Neuron übergeben wird, oder die Ausgabe des Netzes darstellt. Die Abbildung in der Aktivierungsfunktion kann jede Funktion sein, die einen Wert x auf einen anderen Wert y abbildet, allerdings sind nicht lineare Funktionen wie zum Beispiel: $y = \max(0, x)$ (ReLU) generell besser geeignet, da man nur mit dieser Art von Aktivierungsfunktionen nicht lineare Zusammenhänge darstellen kann.

Ein typisches Neuron ist in Abbildung 4.1 dargestellt. Durch das Anpassen von

$$\sum_{i=1}^n x_i * w_i = \text{Netzeingabe}$$

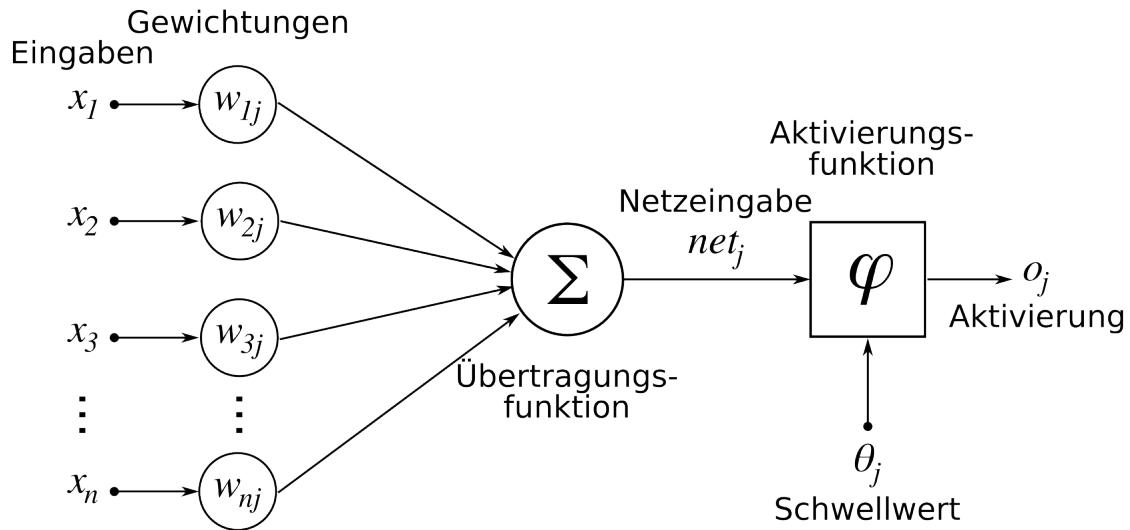


Abbildung 4.1.: Funktionsweise eines künstlichen Neurons

Quelle: https://upload.wikimedia.org/wikipedia/commons/7/7f/ArtificialNeuronModel_deutsch.png

Gewichten, w_i in der Abbildung 4.1, kann ein Neuron den Einfluss von Eingabedaten verändern. Die optimale Belegung von Gewichten wird von dem Netz über mehrere Versuche hinweg erarbeitet. Dafür muss definiert werden, was das Neurale Netz als optimales Ergebnis ansieht. Für die Anpassung der Gewichte in jedem Neuron wird ein Datensatz genutzt, den das Neurale Netz mehrmals durchläuft. Diese Daten werden Trainingsdaten genannt. Bei jeden Durchlauf der Trainingsdaten überprüft das Neurale Netz, wie gut seine Ausgaben im Vergleich zu dem vordefinierten optimalen Ergebnis ist. Dis Trainingsdaten brauchen für jeden Datenpunkt ein richtiges Ergebnis, damit das Neurale Netz sein Ergebniss mit dem vorher berechnet oder festgelegtem Optimum vergleichen kann. In dem Falle ist das Optimum der Normalvector zu der jeweiligen Punktfolge.

Die Gewichte in den Neuronen werden vor der Anpassung zufällig festgelegt. In jeder Iteration werden die Gewichte an den Neuronen angepasst, falls dies zu einem besseren Ergebnis führt. Dafür wird über gradienten berechnet, wie sehr jedes gewicht zu dem Fehler beigetragen hat. Über viele iterationen versucht das Netz dadurch den Fehler (engl. Loss) zu minimieren.

In vereinfachte darstellung der Schichten eines Neurales Netzes ist in Abbildung 4.2. Eine Schicht an künstlichen Neuronen ist so definiert, dass man die Neuronen der Schicht immer mit der gleichen Anzahl an Kantenübergägen erreicht. Jedes Neuron der ersten verborgenen Schicht kann zum Beispiel mit nur genau einem Kantenüber-

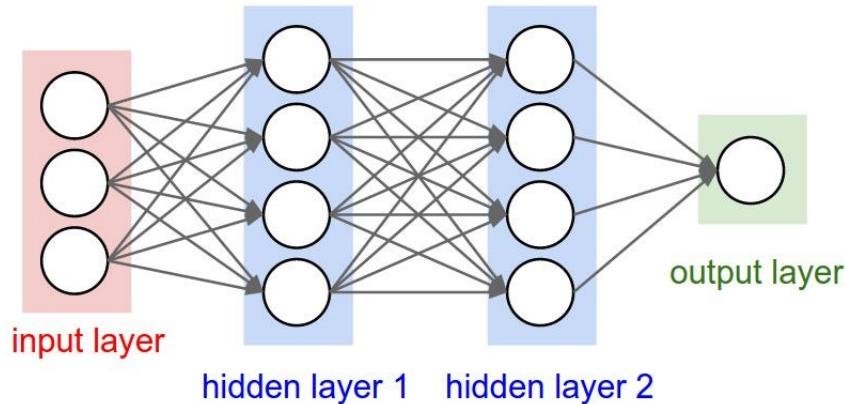


Abbildung 4.2.: Schematische Darstellung von Regressionsnetz
Quelle: https://miro.medium.com/1*LaEgAU-vdsR_pC1McgbikQ.jpeg

gang erreicht werden. Für eine vollständig verbundene Schicht muss jedes künstliche Neuron mit jedem künstlichen Neuron der vorherigen und drarauffolgenden Schicht verbunden sein. Dabei gibt es drei Arten von Schichten, wie in der Abbildung 4.1.2 zu sehen. Zuerst gib es die Eingabeschicht. Diese Schicht repräsentiert die Eingaben und ist in Abbildung 4.2 rot markiert. Jeder Knoten der Eingabeschicht ist ein Datenpunkt, oder in diesem Fall ein Merkmal. Danach gibt es die versteckten Schichten, in Blau markiert. Ihre Aufgabe ist es die Eingabe zu transformieren. Sie erlauben es dem Neuralen Netz eine Eingabe über mehrere Schichten zu verarbeiten. Durch kann das Neurale Netz nicht lineare Zusammenhänge darstellen. Zuletzt gibt es die Ausgabeschicht. Von dieser Schicht wird die Z-Invarianz abgelesen.

4.1.1. Merkmalsextraktion aus Punktwolke

Wenn eine Eingabe ohne Umwege auf die Ausgabe abgebildet werden soll, muss ein tiefes Netz eingesetzt werden. Um ein Bauteil automatisch auszurichten, ist es notwendig, dass das Neurale Netz die Eingabedaten selber verarbeitet. Der Unterschied zwischen einem tiefen und nicht tiefen Neuralen Netz ist es, dass ein tiefes Neurales Netz nicht nur die Abbildung von Merkmalen auf Ausgaben erlernt, sondern auch die Extraktion von Merkmalen aus den Eingabedaten. Die Extractrion der Merkmale wird in diesem Abschnitt erklärt. In der Merkmalsextraktion wird eine Punktwolke auf eine Anzahl an Merkmalen reduziert. Die Merkmale, auf welche die Punktwolke reduziert wird, werden in einem flachen Vektor $feat = (v_0, v_1, v_2, \dots, v_n)$ mit $v_i \in \mathbb{R}$ der Größe n repräsentiert. Dieser Vektor wird Merkmalsvektor genannt und ist eine abstrakte Darstellung der Punktwolken. Der Merkmalsvektor ist als Zwischenergebniss wichtig, um die Eingabedaten in für ein Regressionsnetz besser verarbeitbare

Form zu bringen. Das Regressionsnetz ist der zweite Teil der Struktur des Neurales Netzes und wird in dem folgenden Abschnitt 4.1.2 beschrieben.

Die Struktur der Merkmalsextraktion basiert auf der Arbeit: [test]

Die generelle Struktur der Merkmalsextraktion ist eine wiederholte Folge von Reduzierungen auf den konkreten Daten der Punktwolke, gefolgt von einem kleinen voll verbundenen Neuralen Netz, was mit den konkreten Daten dieser Schicht Merkmale über mehrere Schichten hinweg konstruiert. In dem ersten Schritt in jeder Schicht werden Punkte aus der Punktwolke ausgewählt. Dafür wird ein Algorithmus eingesetzt, der eine Teilmenge der Ausgangspunkte auswählt, um eine möglichst gute Abdeckung über die Ausgangspunkte zu erreichen (Farthest point sampling).

Dafür geht der Algorithmus schrittweise vor. In jedem Schritt wählt er den Punkt der Punktwolke aus, der am weitesten von allen bisher ausgewählten Punkten entfernt ist, solange bis die gewünschte Menge an Punkten erreicht ist. Der Algorithmus sucht nicht nach der optimalen Abdeckung, welche die Distanz von jedem nicht ausgewählten Punkt zu dem nächsten ausgewählten Punkt minimieren, sondern ist nur eine Abschätzung, dieser optimalen Lösung. Die gefundenen Punkte bilden die ausgewählten Punkte für die aktuelle Schicht. Danach werden für jeden ausgewählten Punkt jeder Punkt innerhalb eines Radius um sich selber verbunden. Diese Punkte sind die Nachbarn des ausgewählten Punktes. In der Abbildung 4.3 wird die Auswahl von Punkten und die Reduktion der ausgewählten Punkte in dem SSampling and GroupingSSchritt gezeigt. Es ist zu sehen, dass um bestimmte Punkte nach diesem Schritt ein Radius gezeichnet ist, der die Nachbarn der ausgewählten Punkte darstellt.

Auswahl ⊂ Punktwolke

Die ausgewählten Punkte werden als künstliche Neuronen genutzt. Dabei sind die Merkmale, die in den Punkten der Nachbarn gespeichert sind, die Eingabe für das künstliche Neuron. Die Gewichte für jede Eingabe werden erlernt, dafür wird ein Multi Layer Perceptron (MLP) genutzt. Der MLP lernt, wie die Gewichte für die oben bestimmten Kanten sein sollen. Dafür nutzt der MLP die relative Position von ausgewählten Punkten zu ihren Nachbarn. Ein MLP funktioniert ähnlich zu dem Regressionsnetz, was in Abschnitt 4.1.2 erklärt wird. Ein ausgewählter Punkt verändert seine Merkmale anhand der Summe von Merkmalen der Punkte in seiner Nachbarschaft, die nach ihrer relativen Position gewichtet werden. Die Anzahl an Punkten verringert sich, aber die Anzahl der Merkmale pro Punkt erhöht sich mit jeder Schicht.

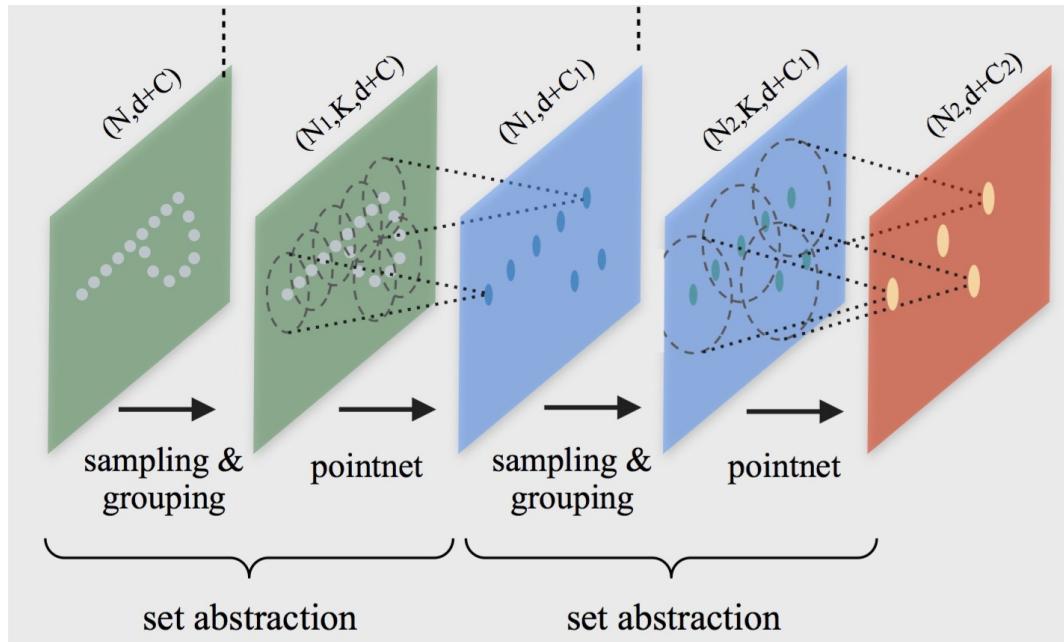


Abbildung 4.3.: Schematische Darstellung von Merkmalsextraktionschichten
Quelle: <https://stanford.edu/~rqi/pointnet2/images/pnpp.jpg>

$$\forall \vec{a} \in \text{Auswahl}, \forall \vec{p} \in \text{Punkt wolke} : \|\vec{a} - \vec{p}\|_2 \leq r \implies (p, a) \in \text{Kanten}$$

Über mehrere Schichten hinweg, werden Merkmale für die Punkte gelernt, um dann die Punkte zu reduzieren und die Merkmale an die ausgewählten Punkte weiterzugeben. Das generiert eine Menge an Merkmalen für jeden Punkt, aber für die Weiterverarbeitung wird ein Merkmalsvektor benötigt, der das gesammelte Bauteil beschreibt. Im letzten Schritt wird das größte Merkmal in jeder Dimension der Merkmale aller Punkte genommen. So wird ein flacher Merkmalsvektor erstellt. Mit dem größten Merkmal ist das numerisch Größte gemeint, nicht unbedingt das Wichtigste.

4.1.2. Regression auf Merkmalsvektor

Nach der Bestimmung von abstrakten Merkmalen in einem Merkmalsvektor im vorherigen Abschnitt 4.1.1 muss aus diesem Vektor ein konkretes Ergebnis abgeleitet werden. Dafür wird eine Abbildung von dem Vektor auf die Ergebnisse erlernt. Dieses Vorgehen wird Regression genannt. In diesem Fall wird auf die Z-Invarianz regresiert.

Eine Schicht funktioniert so, dass jedes Neuron die gewichtete Summe von allen vorhergenen Neuronen erhält und diese Summe an eine Aktivierungsfunktion übergibt. Dabei werden Methoden wie Normalisierung und zufälliges Zurücksetzen von Eingaben verwendet, um die Transformation verlässlicher zu machen.

Die Normalisierung stellt sicher, dass es keine zu großen Änderungen innerhalb einer Schicht vorkommen, da alle Werte relativ zu dem Durchschnitt normalisiert werden. Bei einer großen Änderung innerhalb einer Schicht, zwingt es darauffolgende Schichten ebenfalls Änderungen vorzunehmen. Die konstante Anpassung zu Änderungen in den vorherigen Schichten macht das Lernen instabil.

Das Zufällige Zurücksetzen von Eingaben verhindert eine zu große Abhängigkeit von nur einem Neuron oder einem Pfad. Dafür werden mit einer vorbestimmten Wahrscheinlichkeit gewichtete Summen auf null gesetzt, sodass sie keinen Einfluss auf die nächste Schicht haben.

4.2. Konstruktion der Ausrichtung

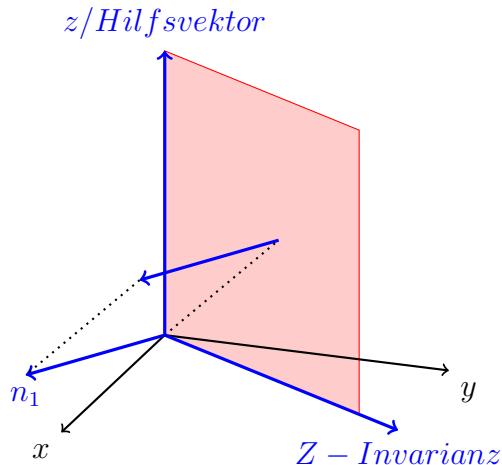
Die, in Abschnitt 4.1.2 berechnete, Z-Invarianz ist nicht genug, um eine Drehung darzustellen, um ein Bauteil drehen zu können ist eine Rotationsmatrix notwendig. Rotationsmatrizen sind, für den 3d Raum, 3 x 3 Matrizen, die durch Matrixmultiplikation eine Punkt wolke drehen können.

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} = \begin{bmatrix} x'_1 & y'_1 & z'_1 \\ x'_2 & y'_2 & z'_2 \\ \vdots & \vdots & \vdots \\ x'_n & y'_n & z'_n \end{bmatrix}$$

Dabei ist die Norm von jeder Zeile gleich Eins und jede Zeile ist, wenn man sie als Vektor betrachtet orthogonal zu den anderen Zeilen. Dasselbe gilt für Spalten.

$$\forall i \in \{1, 2, 3\} : \left\| \begin{bmatrix} R_{1i} \\ R_{2i} \\ R_{3i} \end{bmatrix} \right\|_2 = 1 \quad \text{und} \quad \forall i \in \{1, 2, 3\} : \left\| \begin{bmatrix} R_{i1} \\ R_{i2} \\ R_{i3} \end{bmatrix} \right\|_2 = 1$$

Um aus dem 3d Vektor, der die Z-invarianz darstellt, eine Rotationsmatrix zu erstellen, kann die Z-Invarianz selber als Zeile der Rotationmatrix genommen werden. Die restlichen Zeilen müssen dann aus der ersten Zeile abgeleitet werden. Die Zeilen einer Rotationmatrix müssen Orthogonal zueinander sein, daher muss für die anderen Zeilen Vektoren gefunden werden die Orthogonal zu der Z-Invarianz sind. Wenn zwei Vektoren gegeben sind, kann man aus ihnen eine Ebene bilden eine Normale dieser Ebene ist Orthogonal zu beiden Vektoren. Dadurch lassen sich orthogonale Vektoren zu der Z-Invarianz finden.

Abbildung 4.4.: Berechnung von n_1

Da die Z-Invarianz aber nur ein Vektor ist, muss für die Bestimmung anderer Zeilen im ersten Schritt ein Hilfsvektor genommen werden. Dafür bietet sich die z-Achse an, da die Rotationen relativ zu der z-Achse bewertet werden. Zwischen der Z-Invarianz und der z-Achse wird eine Ebene aufgespannt. Die Normale der Ebene ist eine Zeile in der Rotationmatrix.

Als Hilfsvektor wird immer die normierte z-Achse genommen, daher besteht die Möglichkeit, dass die Z-Invarianz sehr ähnlich zu der z-Achse ist. Das macht die Berechnung der Normalen mathematisch instabil, da durch die extreme Ähnlichkeit jetzt der Nullvektor als Ergebnis kommen könnte. Um dieses Problem zu beheben, wird überprüft, ob der Hilfsvektor zu ähnlich zu der Z-Invarianz ist. Wenn sie zu ähnlich sind, wird der Hilfsvektor durch einen anderen Vektor ersetzt. Einen anderen Hilfsvektor zu nehmen beeinflusst die Rotation, aber diese Ausnahme ist sehr selten, verschlechtert sie das durchschnittliche Ergebnis nicht sehr. Als Alternative zu der z-Achse bieten sich die x und y Achsen an, da sie beide 90 Grad von der z-Achse entfernt sind und dadurch, die Struktur der Rotation erhalten bleibt.

Durch das Berechnen einer anderen Zeile außer der Z-Invarianz ist ein Hilfsvektor nicht mehr vonnöten. Die berechnete Normale bildet die letzte Zeile, die für eine Rotationmatrix vonnöten ist. Sie ist orthogonal zu den anderen Zeilen, da sie eine Normale in der Ebene der anderen Zeilen ist. Normale-1 ist ebenfalls orthogonal zu der Normale-2, da es in der Ebene ist, dessen normale Normale-2 ist. Sie ist ebenfalls orthogonal zu der Z-Invarianz, da Normale-1 aus Ebene, die die Z-Invarianz beinhaltet gebildet wurde. Orthogonalität ist symmetrisch, daher ist auch die Z-Invarianz orthogonal zu Normale-1 und Normale-2. Das erfüllt die oben genannte Bedingung, dass alle Zeilen orthogonal zueinander seien müssen.

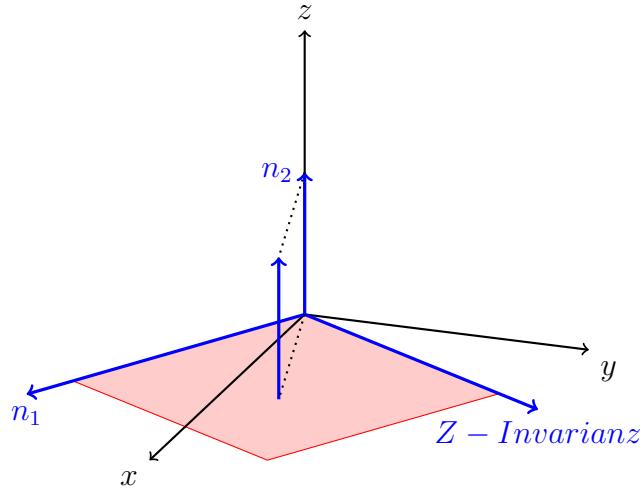


Abbildung 4.5.: Berechnung von n_2

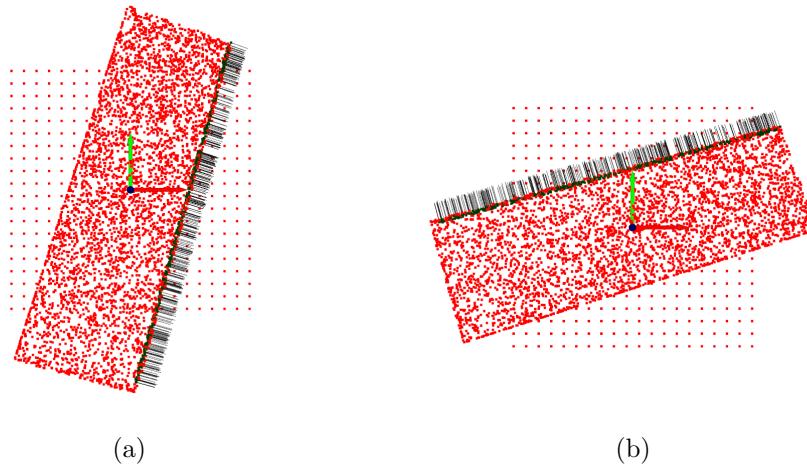


Abbildung 4.6.: Vertauschen von n_1 und n_2 in der Rotationsmatrix

Die Z-Invarianz wird als unterste Zeile genommen, die beiden berechneten Normalen können eine beliebige andere Zeile nehmen. Die Wahl der Zeilen verändert nur die Rotation des Bauteils um die z-Achse herum, wie man in der Abbildung 4.6 sehen kann. Solange also die Zeilen immer gleich angeordnet werden, wird die Gradabweichung von der z-Achse gleich bleiben.

Nachdem die Rotationmatrix erstellt wurde muss zuletzt überprüft werden, ob sie eine gültige Rotation ist. Im 5d-Druck kann sich die Platform nicht völlig frei bewegen, sie kann eine maximale Gradabweichung vom Ursprung von 90 Grad unterstützen, also insgesamt jeweils 180 Grad entlang zweier Achsen. Daher wird überprüft, ob die Rotationsmatrix das Bauteil über diese Grenzen hinaus drehen würde. Falls dem so ist, muss die Rotationsmatrix gespiegelt werden.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

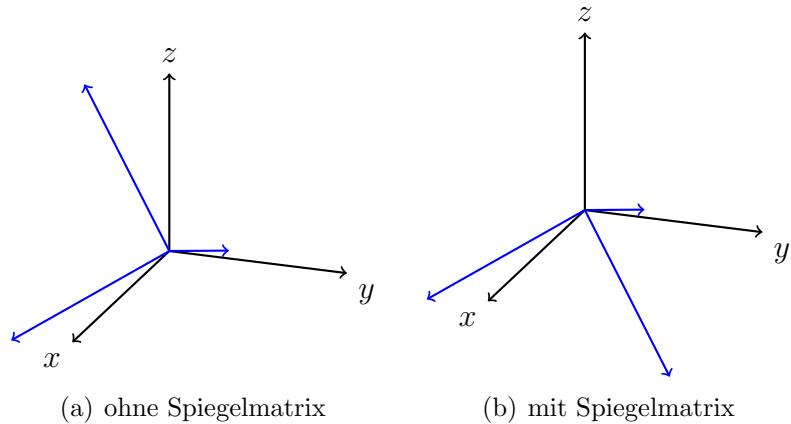


Abbildung 4.7.: Veränderung der Rotationmatrix durch die Spiegelmatrix

Durch die Matrixmultiplikation mit der Spiegelmatrix, aus 4.7, wird eine Rotationsmatrix in der xy-Ebene gespiegelt. Das hat den Effekt, dass sich das Bauteil um 180 Grad dreht. Da eine Spiegelung nötig war, galt vor der Spiegelung, dass der Betrag der Rotation mehr als 90 Grad ist. Die Rotation ist also zwischen 90 und 180 Grad. Sie kann nicht höher sein, da eine sinnvolle Rotation nur von -180 bis 180 Grad gehen kann und so einen ganzen Kreis bildet. Wenn man die Rotation spiegelt, wird sie um 180 Grad gedreht, es gilt:

Wenn sie vor der Spiegelung nicht im 180 Grad Halbkreis der erlaubten Drehungen war, dann ist sie es danach.

$$rot \in \mathbb{Q} \wedge |rot| > 90 \wedge |rot| < 180 \implies |rot| - 180 < 90$$

5. Evaluation

Die Ausrichtung eines Bauteils ist in den meisten Fällen nicht perfekt. In diesem Kapitel wird behandelt wie gut das in Kapitel 4 erstellte Neurales Netz ein Bauteil ausrichten kann. Dafür wird nicht nur die Ausrichtung selber überprüft. Es wird auch die Größe der Rotationen untersucht, da das drehen eines Bauteil Zeit kostet und die Minimierung dieser Drehung wie in 2.2 erwähnt eine Priorität ist. Außerdem wird die Fähigkeit des neuralen Netzes Fehler in den Daten zu widerstehen in Abschnitt ?? überprüft und die Fähigkeit auf komplexen Bauteilen zu arbeiten in Abschnitt 5.4 getestet.

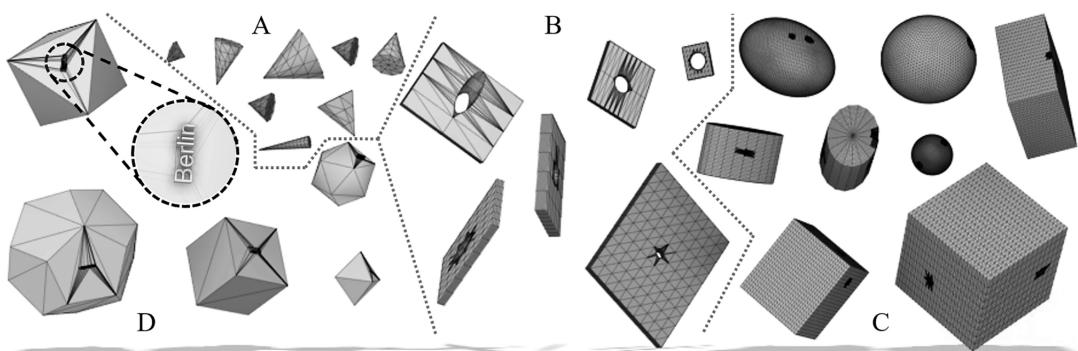


Abbildung 5.1.: Darstellung aller Typen von Bauteilen im Datensatz
Quelle: https://media.springernature.com/full/springer-static/image/art%3A10.1007%2Fs40964-025-00960-6/MediaObjects/40964_2025_960_Fig1_HTML.png

Alle Daten, die für die Auswertung benutzt werden, sind in dem Artikel: [Datensatz] erstellt worden und dann wie in Kapitel 3 beschrieben verändert worden. Die Abbildung 5.1 stellt alle Typen von Bauteilen dar, die in dem Datensatz enthalten sind. Es gibt insgesamt 4 verschiedene Typen, die alle einfache geometrische Objekte sind. Die Typen sind von A bis D durchnummeriert.

Für alle Tests werden 1000 zufällige Flächen aus dem Datensatz genommen. Diese Flächen werden wie in Kapitel 4 beschrieben gefunden. Tests haben ergeben, dass das Verwenden von einem Datensatz mit mehr als 1000 Flächen keinen Einfluss auf das Ergebniss hat.

5.1. Ausrichtung der Bauteile

Die Ausrichtung der Bauteile ist das Ziel der Arbeit. Dementsprechend ist es die wichtigste Metrik für den Erfolg des neuronalen Netzes.

5.2. Minimierung der Bewegung

Die Minimierung der Bewegung ist wichtig, da jede Drehung eines Bauteils Zeit kostet. Als zweites Ziel des Netzes wird dementsprechend versucht, die Größe der Drehung des Bauteils zu minimieren.

5.3. Verlässlichkeit der Ausrichtung

In dem bisherigen Test ist immer davon ausgegangen worden, dass die Punktwolken eine perfekte Darstellung des Bauteils sind. In der Praxis, muss aber davon ausgegangen werden, dass die Punktwolken Fehler aufweisen.

5.3.1. Effekt von Rauschen auf den Daten

Eine Fläche eines Bauteils, die Punkte beinhaltet, die nicht zu der Fläche gehören, kann das Ergebnis des neuronalen Netzes stark beeinflussen. Dieser Fehler könnte durch schlechtes Clustering entstehen, wenn zwei Flächen nicht richtig getrennt werden. Punkte die nicht zu der Fläche gehören, aber dennoch in der Punktwolke der Fläche enthalten sind, werden als Rauschen (engl. Noise) bezeichnet. Um den Einfluss von Rauschen auf das Netz zu bestimmen, werden Punkte zu sonst korrekten Flächen hinzugefügt.

Die Abbildung 5.2 zeigt die Ergebnisse des Tests. Um Noise zu simulieren, wurden für jede Fläche um das Zentrum der Fläche zufällig Punkte verteilt. Dabei wurden die Punkte normalverteilt, sodass die meisten Punkte nahe der Fläche sind. Die zufällige Verteilung bricht die kohärente Struktur der Fläche auf. In der Abbildung 5.2 wurde der Unterschied des Normalvektors der Fläche zu der z-Achse in Orange aufgezeichnet, die Standardabweichung des Normalvectors zu der z-Achse in Grün und die benötigte Größe der Drehung ist Blau. Die x-Achse der Abbildung zeigt das Rauschen. Dabei ist die Anzahl der Rauschpunkte gleich des Wertes auf der x-Achse mal 20. Die y-Achse zeigt die Ergebnisse bei den jeweiligen Rauschleveln. Abbildung 5.2 zeigt, dass das Rauschen keinen großen Einfluss auf den Unterschied des Normalvektors der Fläche zu der z-Achse oder die benötigte Größe der Drehung hat. Auf die

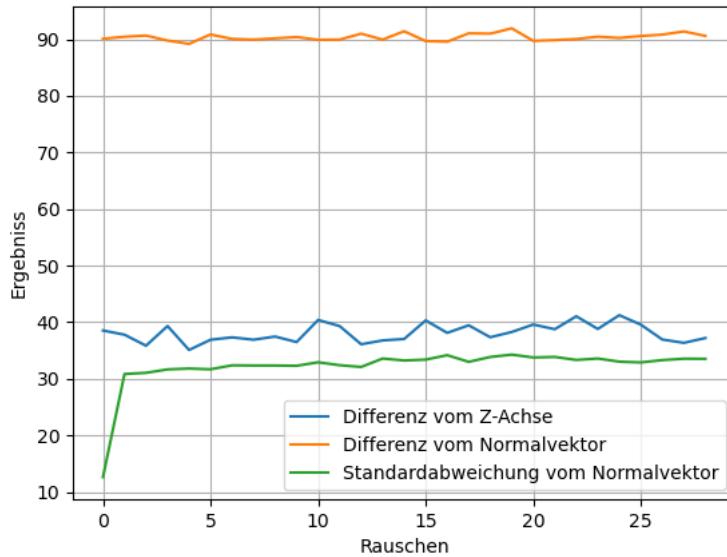


Abbildung 5.2.: Test mit Noise auf dem Neuralen Netz

Standartabweichung des Normalvektors hat das Rauschen allerdings einen extremen Einfluss. Sobald Rauschen hinzugefügt wird, steigt die Standartabweichung extrem an und bleibt dann bei dem Fehlermaß, zu dem es steigt unabhängig von weiterem Rauschen, dass hinzugefügt wird. Daraus lässt sich schließen, dass das Netz keine gute Fähigkeit hat, Rauschen zu widerstehen. Die absolute Ausrichtung in Orange bleibt zwar stabiel, aber da die Abwichung von diesem Durchschnitt extrem ansteigt, ist das Ergebnis unzuverlässig.

5.3.2. Effekt von schlechter Flächensuche

Eine Fläche, die in einem Bauteil gefunden wurde, muss nicht immer perfekt die Fläche des Bauteils repräsentieren. Häufig werden Punkte, die zu der Fläche gehören sollten, nicht gefunden, was die Punktfolge keiner macht als sie es mit einer perfekten Suche wäre. Der Einfluss von fehlenden Punkten in der Punktfolge einer Fläche wird getestet, indem über mehrere Iterationen Punkte aus den Punktfolgen entfernt werden. Die Anzahl der Punkte basiert auf einem Prozentsatz, der sich mit jeder Iteration erhöht. Durch das Verwenden von Prozentsätzen wird sichergestellt, dass die Ergebnisse unabhängig von der ursprünglichen Größe der Punktfolge sind. Die Abbildung 5.3 zeigt den Verlust an Genauigkeit des Netzes, wenn Punkte aus der Punktfolge entfernt werden. Auf der x-Achse ist der Prozentsatz der entfernten Punkte dargestellt, auf der y-Achse ist das Ergebnis der Tests dargestellt. Die

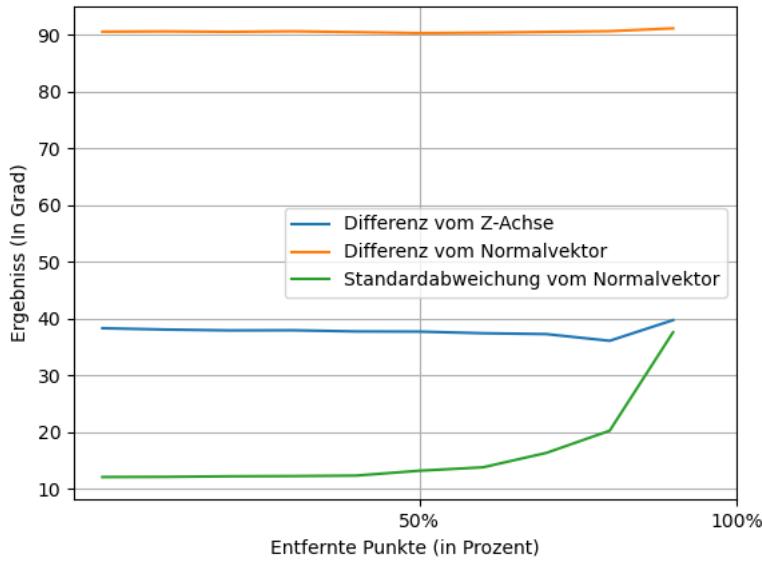


Abbildung 5.3.: Test von schlechter Flächensuche auf dem Neuralen Netz

Abweichung von dem Normalvektor der Fläche und der z-Achse sind beide nahezu unverändert, wenn Punkte entfernt werden. Die Standardabweichung des Normalvektors bleibt für die ersten 60 %, ebenfalls fast unverändert. Erst nachdem 60 % der Punkte entfernt wurden, steigt die Standardabweichung stark an. Bei 90 % entfernten Punkten ist die Standardabweichung fast bei 45 Grad angekommen, was bedeutet, dass bei einem so hohen Verlust das Ergebnis kaum besser als Zufall ist. Aber bis zu diesem Punkt zeigt das Netz eine gute Fähigkeit, fehlende Punkte in der Punktwolke zu widerstehen. Ein Verlust von über 60 % der Punkte ist extrem unwahrscheinlich in der Praxis. Das Netz zeigt also eine gute Widerstandsfähigkeit gegen schlechte Flächensuche.

5.4. Generalisierbarkeit

Alle bisherigen Auswertungen wurden auf den gleichen Typen von Daten durchgeführt, auf denen das Netz auch trainiert wurde. Um die Verlässlichkeit des Netzes zu überprüfen wird in diesem Abschnitt getestet, ob das Netz auch auf komplexeren Bauteilen arbeiten kann. Für diesen Zweck wurden mehrere komplexe Bauteile getestet, komplex bedeutet in diesem Fall, dass die Bauteile nicht aus einfachen geometrischen Formen bestehen, sondern gebogene Flächen und komplexe Strukturen beinhalten.

Bauteil	Gradabweichung vom Optimum	Bewegung des Bauteils
Stanford Hase	10.76	17.29
Stanford Drache	11.61	25.02
Utah Teapot	10.84	26.72

Tabelle 5.1.: Testergebnisse auf komplexen Bauteilen

Alle Angaben in der Tabelle 5.1 sind Durchschnittswerte über alle Flächen eines Bauteils. Die Ergebnisse sind immer in Grad und für jedes Bauteil einzeln angegeben. Es lässt sich erkennen, dass die Abweichung vom Optimum bei komplexen Bauteilen höher ist, als bei den einfachen Bauteilen, die in Abschnitt 5.1 getestet wurden. Die Abweichung vom Optimum liegt bei allen komplexen Bauteilen bei etwa 11 Grad. Der geringe Unterschied zwischen den Ergebnissen der komplexen Bauteile lässt darauf schließen, dass die Auswahl der komplexen Bauteile eine gute Repräsentation für komplexe Bauteile im Allgemeinen ist. Allerdings ist die Bewegung des Bauteils bei dem Stanford Hasen deutlich geringer als bei den anderen beiden Bauteilen. Der Unterschied könnte daran liegen, dass der Stanford Hase nur eine Grundfläche hat. Grundflächen werden immer für hohe Bewegung sorgen, da die optimale Ausrichtung meistens im 90 Grad Winkel zu der Grundfläche liegt. Der Stanford Drache und die Utah Teapot haben beide mehrere Grundflächen, was die Bewegung des Bauteils erhöhen könnte.

6. Fazit

6.1. Fazit

In dieser Arbeit wurde zunächst die Notwendigkeit der Ausrichtung von Bauteilen in der additiven Fertigung erläutert. Dabei wurde ganz besonders auf die zusätzlichen Herausforderungen bei dem 5D Druck eingegangen, in dem die Ausrichtung eine größere Rolle spielt, da die Ausrichtung eines Bauteils mehrmals während des Druckens geändert werden kann.

Anschließend wurde ein neuronales Netz als Lösungsansatz vorgestellt, welches in der Lage ist, die Ausrichtung eines Bauteils für Flächen in dem Bauteil zu bestimmen. Dafür wurde ebenfalls ein Verfahren zur Extraktion von Flächen aus Bauteilen beschrieben.

Schließlich wurden Test auf dem beschieben neuronalen Netz durchgeführt und ihre Ergebnisse wurden ausgewertet. Daraus ergab sich, dass das Netz in der Lage ist, Bauteile mit einer akzeptablen Genauigkeit auszurichten. Eine Abweichung von unter 10° ist in den meisten Fällen erreicht worden. Eine solche Genauigkeit ist nicht optimal, aber dennoch ausreichend, um die Vorteile der Ausrichtung in der additiven Fertigung zu nutzen. Auch auf komplexeren Bauteilen konnte das Netz verlässlich akzeptable Ergebnisse erzielen, was auf eine gute Generalisierbarkeit des Netzes hinweist.

Die minimierung der Bewegung des Bauteils während des Ausrichtens hatte eine kleine, aber sichtbaren verkleinerung der benötigten Drehung ergeben. Der durchschnittliche Winkel, um den ein Teil gedreht werden muss, lag bei unter 40° . Ohne die Minimierung würde der durchschnittliche Drehwinkel bei über 45° liegen, also zufällig zwischen 0 und 90 Grad. Auch wenn die Verbesserung durch die Minimierung der Bewegung nicht sehr groß ist, so ist sie dennoch relevant, da jede eingesparte Drehung die Druckzeit verringert. Es ist kein anderes Verhalten bei komplexen Bauteilen zu sehen gewesen.

Das Finden von Flächen in Bauteilen hat in den meisten Fällen sehr gut funktioniert. Allerdings besonders bei komplexen Bauteilen, die über viele gebogene Flächen

verfügen, hat die Suche häufig Flächen mit Löchern oder extrem kleine Flächen ergeben. Die Löcher in den Flächen sind in sich selber kein Problem, die Suche hat ein Loch in der Fläche gelassen, weil dort die Punkte einen zu unterschiedlichen Normalvektor hatten. So sollte die Suche auf funktionieren, aber ein Loch in einer Fläche könnte das Nutzen von Stützstrukturen wieder nötig machen, was die Vorteile der Ausrichtung wieder verringert. Durch den erfolg bei einfachen Bauteilen, lässt sich schlussfolgern, dass die Flächensuche besonders gut darin ist Flächen die nicht gebogen sind zu finden und die sich durch einen Knick in der Geometrie von anderen Flächen abgrenzen lassen.

Es lässt sich sagen, dass das beschriebene neurale Netz in der Lage ist, Bauteile für die additive Fertigung auszurichten. Eine geringere Abweichung von dem Optimum wäre wünschenswert, aber die erzielten Ergebnisse sind dennoch ausreichend, um die Vorteile der Ausrichtung zu nutzen. Die Flächensuche funktioniert in den meisten Fällen gut, aber es gibt noch Raum für Verbesserungen, besonders bei komplexen Bauteilen mit gebogenen Flächen. Die Minimierung der Bewegung des Bauteils ist funktional, aber die Verbesserung ist nur gering. Insgesamt ist das beschriebene Verfahren ein akzeptabler Ansatz, um Bauteile in der additiven Fertigung auszurichten, wenn auch noch einige Verbesserungen möglich wären. Es ist für den praktischen Einsatz noch nicht denkbar.

6.2. Ausblick

Diese Arbeit hat gezeigt, dass das Ausrichten von Bauteilen mit neuronalen Netzen möglich ist. Es gibt allerdings noch viele Bereiche in denen Verbesserungen vorgenommen werden können.

Die Genauigkeit des Netzes könnte durch eine Verbesserung der Flächensuche gesteigert werden. Die Flächensuche selber basiert auf einem fehlerbehafteten Ansatz. Für unterschiedliche Arten von Bauteile müssen die Parameter der Flächensuche angepasst werden, was dem Sinn von automatischer Anpassung widerspricht. Eine Flächensuche, die weniger Fehler macht, würde dem Neuralen Netz bessere Trainingsdaten liefern, was die Genauigkeit des Netzes steigern könnte. Ein weiteres Problem der Flächensuche ist, dass sie Flächen mit Löchern findet. Das ist ein Problem, was mit dem Clustering von Normalvektoren zusammenhängt und dementsprechend vermutlich nicht behoben werden kann, ohne den Ansatz der Flächensuche zu verändern. Eine Flächensuche, die in der Lage ist, Flächen mit Löchern zu vermeiden, würde das Problem der Stützstrukturen lösen.

Die Architektur des Netzes könnte ebenfalls verändert werden, um die Genauigkeit zu steigern. Es könnten andere Architekturen ausprobiert werden, die eventuell besser für das Problem geeignet sind. Auch eine Veränderung der Hyperparameter des Netzes könnte die Genauigkeit verbessern.

Weitere Arbeiten könnten dementsprechend sich auf bessere Flächensuche und Architektur des Netzes konzentrieren.

A. Weitere Informationen

One morning, when Gregor Samsa woke from troubled dreams, he found himself transformed in his bed into a horrible vermin. He lay on his armour-like back, and if he lifted his head a little he could see his brown belly, slightly domed and divided by arches into stiff sections. The bedding was hardly able to cover it and seemed ready to slide off any moment. His many legs, pitifully thin compared with the size of the rest of him, waved about helplessly as he looked. „What's happened to me?“ he thought. It wasn't a dream. His room, a proper human room although a little too small, lay peacefully between its four familiar walls. A collection of textile samples lay spread out on the table - Samsa was a travelling salesman - and above it there hung a picture that he had recently cut out of an illustrated magazine and housed in a nice, gilded frame. It showed a lady fitted out with a fur hat and fur boa who sat upright, raising a heavy fur muff that covered the whole of her lower arm towards the viewer. Gregor then turned to look out the window at the dull weather. Drops of rain could be heard hitting the pane, which made him feel quite sad. „How about if I sleep a little bit longer and forget all this nonsense“, he thought, but that was something he was unable to do because he was used to sleeping on his right, and in his present state couldn't get into that position. However hard he threw himself onto his right, he always rolled back to where he was. He must have tried it a hundred times, shut his eyes so that he wouldn't have to look at the floundering legs, and only stopped when he began to feel a mild, dull pain there that he had never felt before. „Oh, God, he thought, what a strenuous career it is that I've chosen!“ Travelling day in and day out.

Abbildungsverzeichnis

2.1.	Schematische Darstellung eines 3d-Druckers	3
2.2.	Vorschau eines Bauteils mit Stützstrukturen	4
2.3.	Darstellung der Unterschiede im Bauteil abhängig von der Höhe der Schichten	5
2.4.	Darstellung der Beweglichen Platform eines 5d-Druckers	6
3.1.	Clustering von versteckten zweidimensionalen Punkten	10
3.2.	Darstellung von Normalvektoren an einem Punkt der Bauteiloberfläche	10
3.3.	Beispiel von Hierarchie Clustering als Dendrogram	11
3.4.	Hierarchisches Clustering eines Bauteils nach Normalvector	12
3.5.	Beispielhafte Darstellung von DB-Scan Algorithmus	13
4.1.	Funktionsweise eines künstlichen Neurons	16
4.2.	Schematische Darstellung von Regressionsnetz	17
4.3.	Schematische Darstellung von Merkmalsextraktionschichten	19
4.4.	Berechnung von n_1	21
4.5.	Berechnung von n_2	22
4.6.	Vertauschen von n_1 und n_2 in der Rotationsmatrix	22
4.7.	Veränderung der Rotationmatrix durch die Spiegelmatrix	23
5.1.	Darstellung aller Typen von Bauteilen im Datensatz	25
5.2.	Test mit Noise auf dem Neuralen Netz	27
5.3.	Test von schlechter Flächensuche auf dem Neuralen Netz	28

Algorithmenverzeichnis

Quellcodeverzeichnis

Erklärung zur Verwendung von Hilfsmitteln und KI-gestützten Schreibwerkzeugen

Ich versichere, dass ich beim Einsatz von IT-/KI-gestützten Schreibwerkzeugen diese Werkzeuge in der nachfolgenden Übersicht der verwendeten Hilfsmittel mit ihrem Produktnamen und meiner Bezugsquelle vollständig aufgeführt und/oder die betreffenden Textstellen in der Arbeit als mit IT/KI generierter Unterstützung verfasst gekennzeichnet habe.

Mir ist bewusst, dass Täuschungen bzw. Täuschungsversuche nach der für mich geltenden Prüfungsordnung geahndet werden.

Folgende Hilfsmittel und KI-gestützten Schreibwerkzeuge habe ich für die Bearbeitung genutzt:

- Hilfsmittel, Produktnamen, Version, Bezugsquelle, [Webseite](#)
- Hilfsmittel, Produktnamen, Version, Bezugsquelle, [Webseite](#)
- Hilfsmittel, Produktnamen, Version, Bezugsquelle, [Webseite](#)