

■ fakultät für informatik

Bachelor-Arbeit

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

Lars Naumann

10. Januar 2026

Gutachter:

PD Dr. Frank Weichert

Prof. Dr. Heinrich Müller

Lehrstuhl Informatik VII
Computergraphik
TU Dortmund

Inhaltsverzeichnis

	e
Mathematische Notation	1
1. Einleitung	1
1.1. Motivation und Hintergrund	1
1.2. Aufbau der Arbeit	2
2. Topologieaspekte in der Additive Fertigung	3
2.1. 3d-Druck	3
2.2. Überhang-Problem	4
2.3. Fertigungs-Geschwindigkeit	5
2.4. 5d-Druck	6
2.5. Stand der Technik	7
3. Flächensuche in Bauteilen	9
3.1. Kriterien für geeignete Flächen	9
3.2. Clustering nach Normalvektoren in Bauteilen	11
3.2.1. Normalvektoren	11
3.2.2. Hierarchisches Clustering	12
3.3. Separation von parallelen Flächen	14
4. Punktwolkenverarbeitung zur Bauteilausrichtung	19
4.1. Punktwolken Transformation	19
4.1.1. Merkmalsextraktion aus Punktwolke	22
4.1.2. Regression auf Merkmalsvektor	24
4.2. Konstruktion der Ausrichtung	25
5. Evaluation	31
5.1. Versuchsaufbau	31
5.2. Ausrichtung der Bauteile	31
5.3. Verlässlichkeit der Ausrichtung	33
5.3.1. Effekt von Rauschen auf den Daten	33

5.3.2. Effekt von schlechter Flächensuche	34
5.4. Generalisierbarkeit	37
6. Zusammenfassung und Ausblick	41
6.1. Zusammenfassung	41
6.2. Ausblick	42
A. Weitere Informationen	45
Abbildungsverzeichnis	47
Algorithmenverzeichnis	49
Quellcodeverzeichnis	51
Literatur	53

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 [Fas12]. Allerdings gibt es noch Probleme, zum Beispiel mit der strukturellen Integrität von Bauteilen oder der Materialverschwendungen durch Stützstrukturen. Der 5d-Druck biete eine Möglichkeit beide Probleme zu adressieren, indem das Bauteil während des Drucks gedreht wird. Dadurch ist es im 5d-Druck möglich Ausrichtungen für einzelnen Flächen eines Bauteils zu wählen, indem die Platform auf der das Bauteil platziert wird, sich in zwei Achsen kippen kann. 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. Die Abbildung 1.1 zeigt einen 5d-Drucker. Es lässt sich erkennen, dass die Platform gekippt ist.

Neurale Netze sind gut geeignet, Muster in Daten zu erkennen. Die Idee ist es die Informations-Verarbeitungsfähigkeit des menschlichen Gehirns zu auf einem Computer zu simulieren [Wan03]. Das Gebiet der künstlichen Neuralen Netze hat sich bereits in vielen Anwendungsfällen, als verlässlich erwiesen. In der Bildverarbeitung werden es zum Beispiel verwendet, um Objekte in Bildern zu erkennen [KM21].

Es ist daher naheliegend, dass sie auch verwendet werden können, um Muster in einer Menge an Raumpunkten zu erkennen. Eines 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 [Zel+25a]. Es wurde mit der PointNet Architektur [Qi+17a] gezeigt, dass Neurale Netze in der Lage seien können eine Ausrichtung für einfache geometrische Strukturen zu finden. Wenn man diese Idee weiterentwickelt könnte sie eine genutzt werden, um Ausrichtungen für die Flächen eines Bauteils im 5d-Druck zu bestimmen. Zu diesem Zweck ist es notwendig

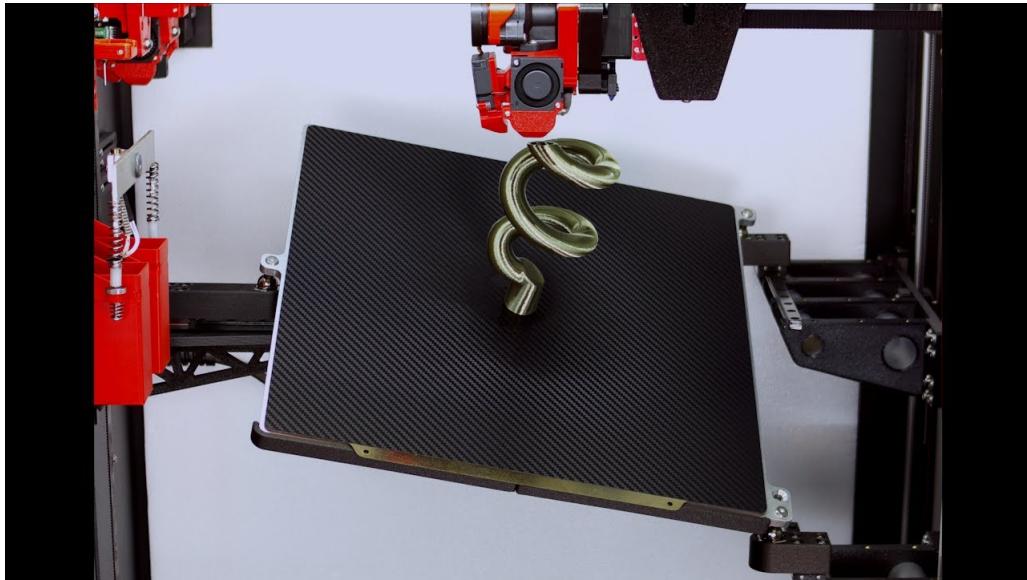


Abbildung 1.1.: Beispiel für einen 5d-Drucker
Quelle: <https://i.ytimg.com/vi/B9sdrezl6AU/maxresdefault.jpg>

lokale Merkmale zu erkennen, was die Nutzung von PointNet++ erfordert [Qi+17b]. Es ist im Gegensatz zu Pointnet in der Lage lokale Merkmale zu erkennen.

1.2. Aufbau der Arbeit

In dieser Arbeit wird versucht das Problem der Ausrichtung von Bauteilen im 5d-Druck durch die Verwendung von tiefen neuralen Netzen zu lösen. Begonnen wird mit der Aufteilung eines Bauteils in kohärente Flächen, wofür mehrere klassische Clustering verfahren 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 [Qi+17b]. 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

2.1. 3d-Druck

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 [Sta15]. 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. Es gibt mehrere Typen von 3d-Druck. Die Typen beinhalten: "binding jetting", "directed energy deposition", "material extrusion", "material jetting", "powder bed fusion", "sheet lamination and vatphotopolymerization". Alle Methoden tragen Material auf. Die am weitesten verbreitete Methode ist "material extrusion" bei der meistens Plastik erhitzt wird und dann das geschmolzene Plastik aufgetragen wird. Die Materialien die genutzt werden können sind aber mehr als Plastik und beinhalten unter anderem noch Metall, Polymere, Komposit Materialien und noch mehr. Metalle werden in der Luft und Raumfahrt verwendet, da sie sehr gute physische Eigenschaften haben. Das benutzte Metall kommt auf den genauen Anwendungsbereich an. Plastik hat den Vorteil, dass es leicht zu erhitzten und billig ist. Es wird unter anderem für die Produktion von Prototypen benutzt, kann aber auch für fertige Produkte verwendet werden. Komposit Materialien, wie Glasfaser, dagegen sind anpassungsfähig und leicht. Sie werden ebenfalls in der Luft und Raumfahrt eingesetzt. Der 3d-Druck ist eine Technologie mit vielen Anwendungsmethoden und Bereichen [SLR19].

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

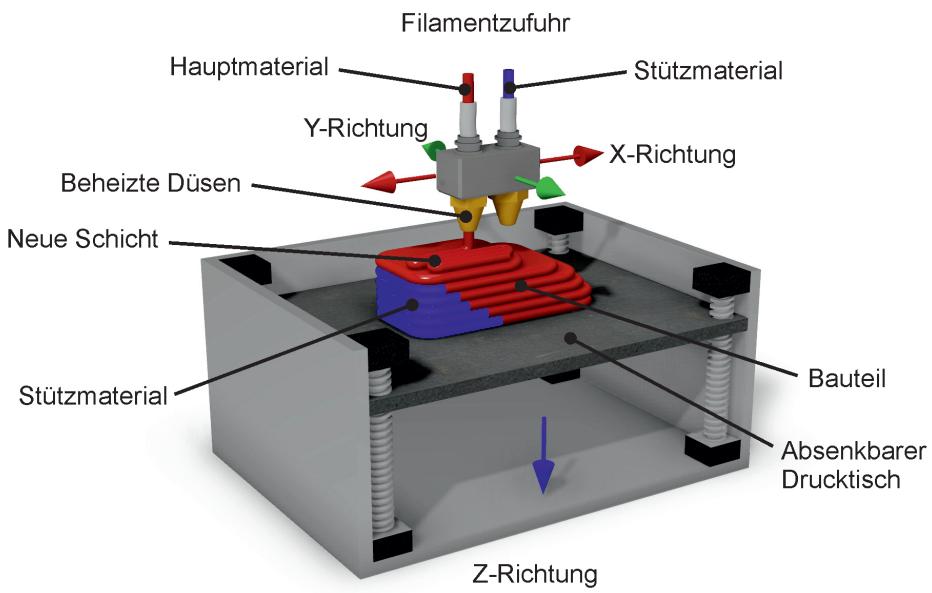


Abbildung 2.1.: Schematische Darstellung eines 3d-Druckers
 Quelle: <https://www.linux-magazin.de/wp-content/uploads/2020/11/b01.jpg>

Abschnitt 2.2 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.

Bevor ein Objekt gedruckt werden kann, muss es in Schichten aufgeteilt werden. Das wird vor dem Drucken von Software erledigt. Für jede erstellte Schicht wird im Vorhinein ein Weg berechnet, mit dem die Düse die Schicht abfährt. Diesen Vorgang nennt man „Slicing“ (de. „In Scheiben Schneiden“).

2.2. 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 eine vorherige Schicht hinausragt, ohne dass darunter Material ist. Dies wird als Überhang-Problem bezeichnet. Um diesem Problem entgegenzuwirken werden Stützstrukturen eingesetzt.

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 Ma-

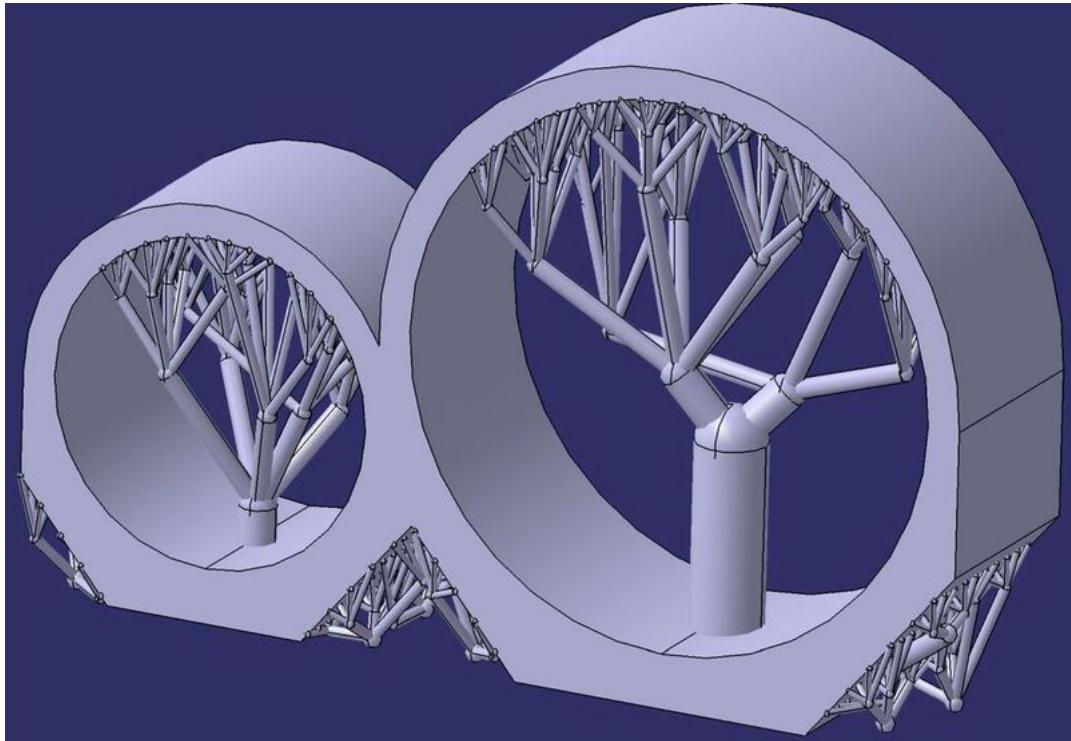


Abbildung 2.2.: Vorschau eines Bauteils mit Stützstrukturen

Quelle: <https://i.stack.imgur.com/cIDB3.png>

terial zu Müll mach. 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. Stützstrukturen sind nicht der einzige Weg das Übergangsproblem zu bekämpfen. Eine Anpassung der Druckgeschwindigkeit und der Materialtemperatur können die Notwendigkeit von Stützstrukturen reduzieren. Dabei ist ein Überhang von 40 Grad möglich ohne Stützstrukturen oder Verlust an Qualität möglich [Jia+18].

2.3. 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 wurde in Abschnitt 2.2 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.

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. Der Pfad, den die Düse über eine Schicht nimmt, kann ebenfalls optimiert werden. Ein Düsenpfad (engl. Toolpath") kann die Qualität und Geschwindigkeit stark beeinflussen [Zhu+24]. Ein längerer Pfad, den die Düse abarbeiten muss, resultiert in einer längeren Druckzeit.

2.4. 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 Plattform in der z Richtung bewegen. Ein 5d-Drucker kann zusätzlich die Plattform in zwei Dimensionen kippen.

In der Abbildung 2.4 ist eine Düse und eine Plattform zu sehen. Die Plattform ist sichtlich gekippt. Die Fähigkeit die Plattform 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 Plattform ist es ebenfalls möglich Schichten, die nicht parallel zu der Plattform sind zu drucken. Die bessere Anpassung an die Geometrie der Bauteile verbessert die Festigkeit des Bauteils.

Diese Fähigkeit Schichten zu erstellen, die nicht parallel zu der Plattform sind, hat den Nachteil, dass die Berechnung der Schichten komplexer wird. Die meisten Programme zu der Erstellung von Schichten unterstützen keinen 5d-Druck. Herausforderungen bei der Erstellung von Schichten für den 5d-Druck sind unter anderem die Kollisionsvermeidung und die Berechnung der optimalen Kippwinkel für jede Schicht.

Die Kollisionsvermeidung ist ein Problem, das durch den 5d-Druck entsteht. Wenn

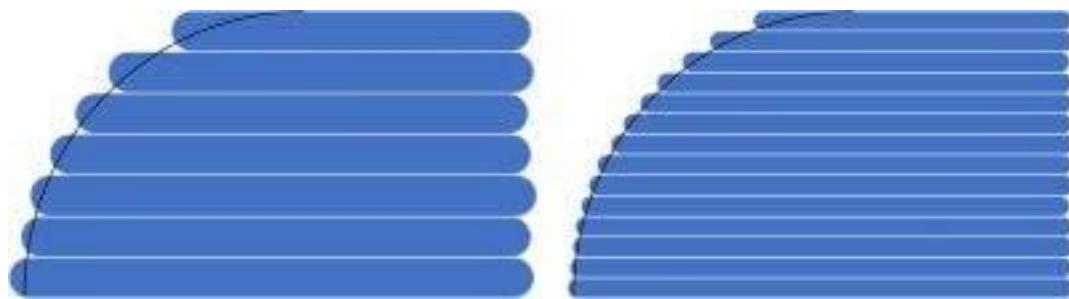


Abbildung 2.3.: Darstellung der Unterschiede im Bauteil abhängig von der Höhe der Schichten

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

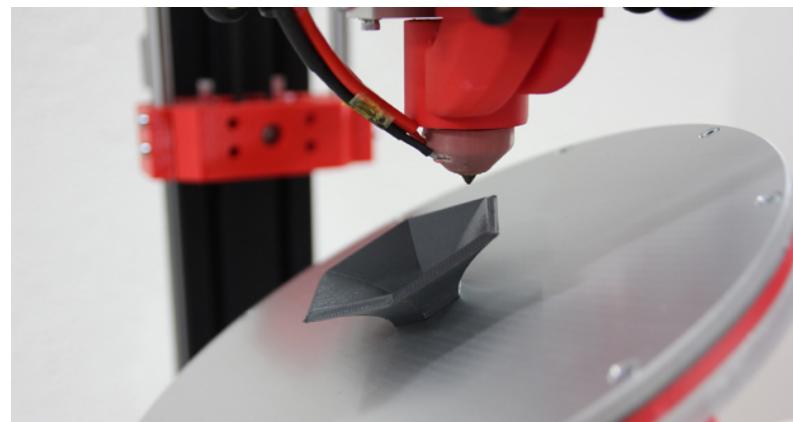


Abbildung 2.4.: Darstellung der beweglichen Plattform eines 5d-Druckers
Quelle: https://www.3dnatives.com/de/wp-content/uploads/sites/3/150217_Zurich1.jpg

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 Kollisionsvermeidung berücksichtigt werden.

2.5. Stand der Technik

3. Flächensuche in Bauteilen

3.1. Kriterien für geeignete Flächen

Bevor ein Bauteil ausgerichtet werden kann, müssen zuerst kohärente zusammenhängende Flächen im Bauteil identifiziert werden. Die Verarbeitung dieser Flächen wird in Kapitel 4 behandelt. Eine kohärente Fläche ist eine stetige Fläche. Sie hat weder Sprünge noch Lücken oder Risse. Diese Eigenschaft ist hilfreich, da eine Fläche die nicht kohärent ist definitionsgemäß an mindestens einer Stelle einen Sprung oder Lücke haben muss. Es ist einfacher, wenn diese Fläche als zwei oder mehr Flächen weiterverarbeitet wird, anstelle von einer Fläche die an mindestens einer Stelle getrennt ist. Eine zusammenhängende Fläche ist in dieser Arbeit eine Fläche, die entweder keine oder eine geringe Krümmung aufweist. Eine Krümmung kann nur bis zu einem bestimmten Punkt akzeptiert und ein Knick innerhalb einer Fläche sollte immer vermieden werden. Wie der Punkt bis, zu dem eine Krümmung akzeptabel ist festgelegt wird, in dem folgenden Abschnitt 3.2 erklärt. Eine Fläche mit einer Krümmung kann eine gute Ausrichtung unmöglich machen. Das Ziel der Ausrichtungen ist es eine Fläche senkrecht in Relation zu der Druckerplatte zu Orientieren. Bei einer gekrümmten Fläche ist dies unmöglich, da wenn man zwei Tangenten an zwei zufälligen Punkten einer gekrümmten Fläche nimmt, werden sie nicht parallel sein. Die Tangenten haben die Steigung der Fläche an den Punkten. Die Abbildung 3.1 zeigt eine gekrümmte Fläche in Blau und auf der Fläche zwei Tangenten, t_1 und t_2 , die in Magenta markiert sind. Wenn die y-Achse die Grundfläche eines Druckers ist, dann lässt es sich leicht erkennen, dass die Fläche unmöglich Senkrecht zu der Druckerfläche seien kann. Sobald in einer Fläche zwei Tangenten, die nicht parallel zueinander sind, existieren, ist eine perfekt senkrechte Ausrichtung unmöglich. Eine Krümmung sollte dementsprechend so gering wie möglich sein.

Dieses Kapitel erklärt, wie in einem Bauteil kohärente und zusammenhängende 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 Punkt-wolke genannt. Es werden in dieser Arbeit mehrere tausend Punkte benutzt, um ein

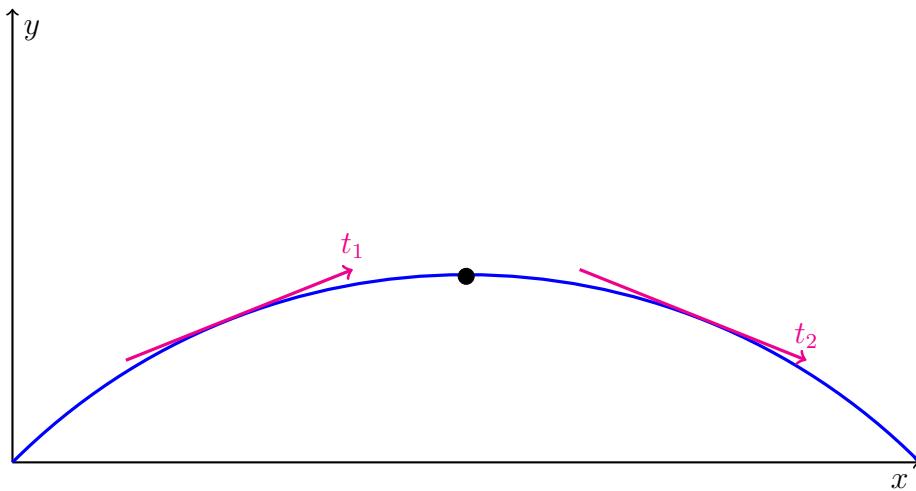


Abbildung 3.1.: Darstellung von Tangenten in einer gekrümmten Fläche

Bauteil mit einer Punktwolke darzustellen. Danach wird die Punktwolke in mehrere kleinere Gruppen aufgeteilt.

Eine Aufteilung von Datenpunkten in Gruppen wird Clustering genannt, eine Gruppe ist ein Cluster. Die Aufteilung in Gruppen basiert auf den Werten der Datenpunkte, Daten mit ähnlichen Werten kommen generell in dieselbe Gruppe. Die Gruppen werden auch Cluster genannt. Was Ähnlichkeit bedeutet ist unterschiedlich, aber ein häufiges Ähnlichkeitsmaß und das Ähnlichkeitsmaß, welches in dieser Arbeit benutzt wird, ist die euklidische Distanz. Euklidische Distanz ist die Distanz zweier Punkt in einer Ebene oder in einem Raum. Die Distanz, die sie beschreibt, ist die Länge der minimalen Strecke zwischen zwei Punkten. Wenn $p = (p_1, p_2, \dots, p_i)$ und $q = ((q_1, q_2, \dots, q_i))$ Koordinaten in einem i-Dimensionalen Raum sind, dann ist die euklidische Distanz gleich $\sqrt{\sum_{i=1}^n (q_i - p_i)^2}$.

In der Abbildung 3.2 sind zwei zweidimensionale Punktwolken. Die linke ist nicht geclustert, die rechte ist geclustert, beide stellen dieselbe Punktwolke dar. Die rechte Punktwolke ist mit drei Farben markiert, Punkte derselben Farbe sind im selben Cluster. In der rechten Punktwolke ist das Ergebnis von einem Clustering verfahren dargestellt. Clustering ist ein Verfahren, mit dem ein Satz an Daten in Gruppen eingeteilt wird. Die Elemente einer Gruppe sind sich innerhalb ähnlicher als Elemente aus anderen Gruppen. Es lässt sich auf der rechten Seite der Abbildung 3.2 erkennen, dass alle Punkte einer Farbe nahe beieinander sind. Außerdem kann man eine klare Trennung zwischen den Farben erkennen. Es gibt keine Ausnahmen, wo ein Punkt der einen Farbe in dem Bereich einer anderen Farbe ist. Es lässt keine Ausnahmen zu. Clustering wird benutzt, um Konzentrationen von ähnlichen Punkten zu finden.

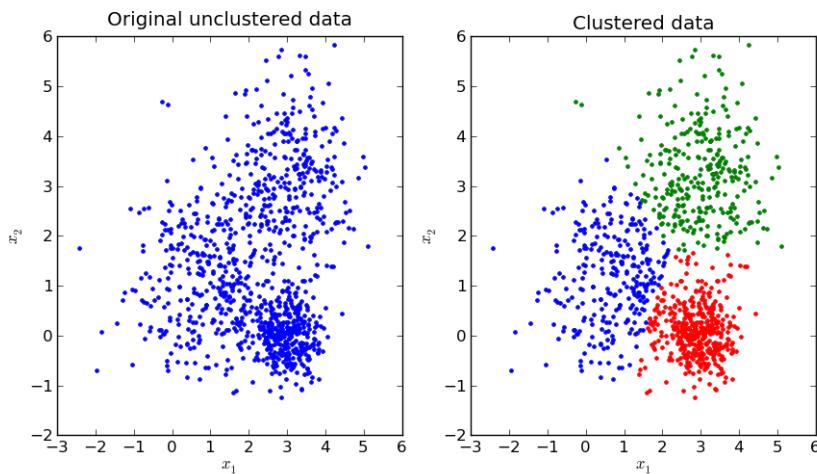


Abbildung 3.2.: Clustering von verstreuten zweidimensionalen Punkten
Quelle: <https://i.stack.imgur.com/cIDB3.png>

3.2. Clustering nach Normalvektoren in Bauteilen

3.2.1. Normalvektoren

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, wenn man eine Ebene, die genau die Steigung des Punktes hat und so tangential zu ihm ist nimmt, wäre die Ausrichtung des Punktes senkrecht zu dieser Ebene. Solche Vektoren werden Normalvektor genannt. Es sind nur die Normalvektoren relevant, die einen dazugehörigen Punkt in der Punktwolke des Bauteils haben.

In der Abbildung 3.3 ist die Bauteiloberfläche 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 Normalvektoren in der Abbildung 3.3 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 gib ihre Ähnlichkeit an. Punkte mit ähnlichem Normalvektor haben eine ähnliche Ausrichtung und sollte teil derselben Fläche sein. Die Darstellung als dreidimensionale Punkte erlaubt funktioniert nur, wenn alle Normalvektoren normiert sind. Ein normierter Vektor hat eine Länge von 1.

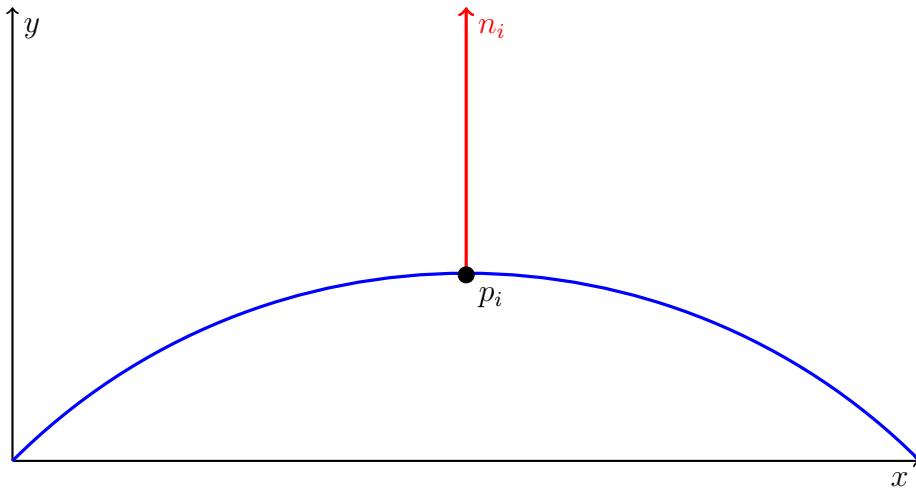


Abbildung 3.3.: Darstellung von Normalvektoren an einem Punkt der Bauteiloberfläche

3.2.2. Hierarchisches Clustering

Viele Methoden des Clusterings benötigen eine vorgegebene Anzahl an Gruppen, in die sie die Punkte unterteilen sollen. Es muss davon ausgegangen werden, dass bei einem Bauteil die Anzahl der Flächen die gefunden werden können unbekannt ist. Das macht Methoden, die eine Anzahl an Gruppen benötigen ungeeignet. Stattdessen wird eine Methode genutzt, die mit allen Datenpunkten zum Start Ihre eigene Gruppe haben, sodass es für jeden Datenpunkt eine Gruppe gibt. Die ähnlichsten zwei Gruppen werden dann zusammengefügt. Auf diesen Gruppen wird der Vorgang wiederholt, bis das Zusammenfügen von Gruppen immer einen vor der Ausführung festgelegten Wert an Differenz überschreitet. Es passt so die Anzahl der Gruppen dem Bauteil an. Dieses Vorgehen wird Hierarchie Clustering genannt.

Die Abbildung 3.4 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 3.4 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 zusammengeführte 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 3.4

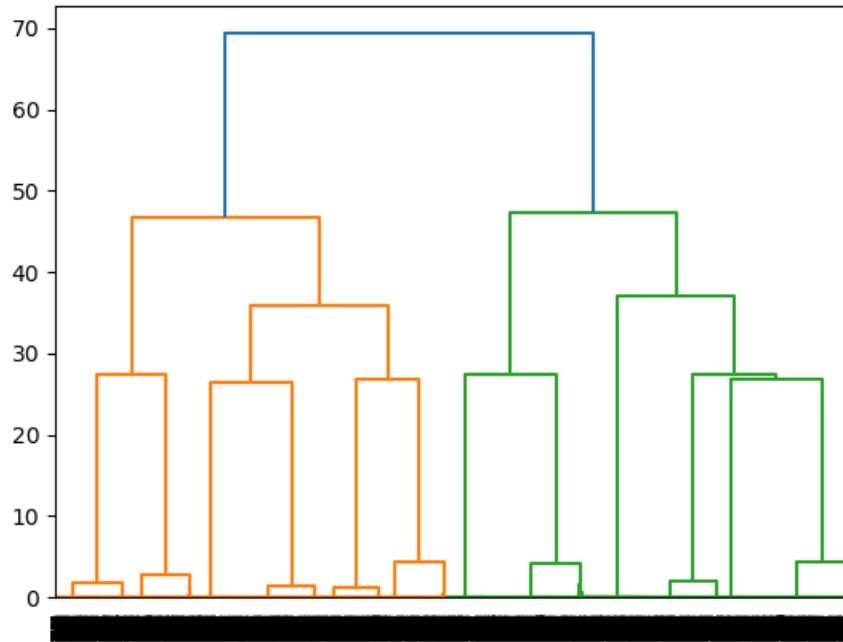


Abbildung 3.4.: Beispiel von Hierarchie Clustering als Dendrogramm

stoppt der Algorithmus nicht, nachdem alle Cluster einen vorbestimmten Distanzwert überschreiten, sondern fügt so lange Cluster zusammen, bis nur noch eins übrig ist.

Das Zusammenfügen von Gruppen basiert auf Ähnlichkeit der Gruppen. In jedem Schritt werden die ähnlichsten Gruppen zusammengefasst. Die Ähnlichkeit zwischen Gruppen wird über die Ward Methode bestimmt [War63]. Diese Methode vereinigt die zwei Gruppen, mit dem geringsten Anstieg an Varianz. Dafür wird für jede Gruppe G die Varianz mittels $\sum_{i \in G} (\|\bar{x} - x_i\|^2)$ bestimmt. \bar{x} ist dabei der Durchschnitt aller Punkte in G . Der Veränderung der Varianz bei dem Zusammenfügen zweier Gruppen A, B kann durch $\sum_{i \in A \cup B} (\|\bar{x} - x_i\|^2) - \sum_{i \in A} (\|\bar{x} - x_i\|^2) - \sum_{i \in B} (\|\bar{x} - x_i\|^2)$ bestimmt werden. Von der Varianz der zusammengefügten Gruppen wird die Varianz der beiden einzelnen Gruppen abgezogen, das Ergebnis ist die Erhöhung der Varianz durch das Zusammenfügen von A und B . Die Gruppen zwischen denen diese Erhöhung minimal ist, werden zusammengefasst.

Auf diese Art und Weise werde die Cluster zwischen Normalvektoren gesucht. Alle Normalvektoren, die in einem Cluster sind, sind so ähnlich, dass das hierarchische Clustering sie nicht weiter zusammengefügt hat. Der Distanzwert auf der y-Achse in 3.4 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 zuzulassen, um gebogene Flächen zu finden. Bei komplexen Bauteilen würde die Suche nach ausschließlich gerade Flächen in sehr viele extrem kleinen Flächen resultieren, da viele Bauteile keine gerade Flächen haben werden. Eine Abweichung zuzulassen ist also sinnvoll.

Das Clustering nach Normalvektoren 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 3.5 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. Die roten Punkte markieren das Bauteil, in dem die Flächensuche durchgeführt wurde.

Der nächste Abschnitt 3.3 beschäftigt sich damit wie die in den hierarchischen Clustering gefundenen Flächen getrennt werden.

3.3. Separation von parallelen Flächen

Alle Flächen in einem Cluster des Normalvektorclusterings aus Abschnitt 3.2 sind in etwa parallel, aber räumlich voneinander getrennt. Das macht sie ungeeignet für die Bildung einer Ausrichtung in Kapitel 4. In Abbildung 3.5 lässt sich erkennen, dass die Punkte, die kohärente Flächen bilden immer sehr dicht beieinander sind. Zwischen den erkennbaren Flächen sind große Lücken ohne Punkte. Die Dichte der Punkte kann somit genutzt werden, um die einzelnen Flächen voneinander zu trennen.

DB-Scan ist eine Dichte basierender Clustering Methode. Es braucht wenig Informationen über die zu gruppierenden Daten, funktioniert auf Clustern jeglicher Form und ist auch auf großen Datensätzen effektiv [Est+96].

Die räumliche Trennung zwischen den Flächen in einem Cluster kann genutzt werden, um sie zu trennen. Dafür wird der DB-Scan Algorithmus verwendet. DB-Scan benötigt einen Radius und eine Mindestanzahl an Nachbarn. Der Radius wird esp und die Mindestanzahl an Nachbarn wird $minPts$ in Abbildung 3.6 genannt. Die Eingabe muss relativ zu der Dichte der Flächen bestimmt werden.

Die Abbildung 3.6 zeigt ein Beispiel für die Funktion von dem DB-Scan Algorithmus, der im Folgenden erklärt wird. 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 (engl. Core Point) eines neuen Clusters. Alle Punkte, die in

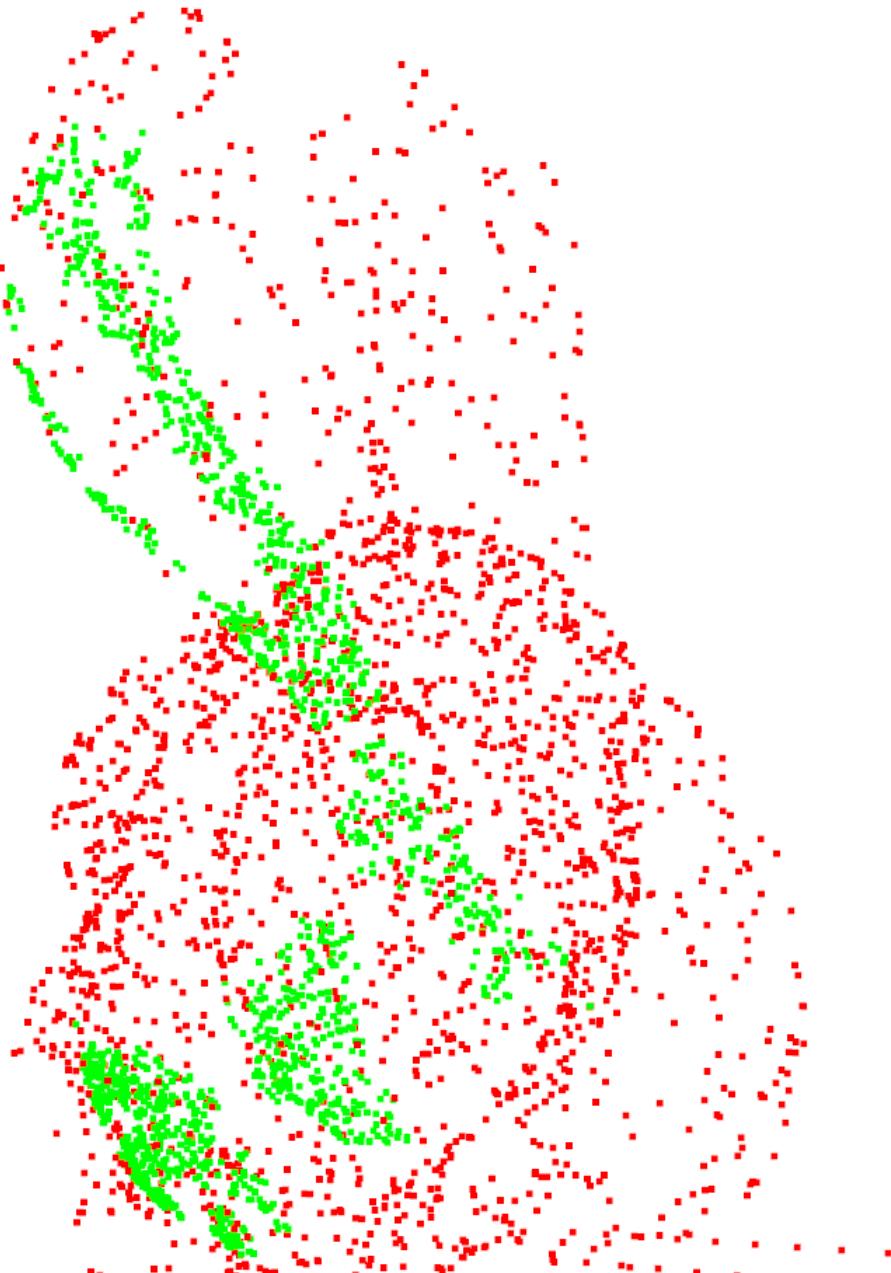


Abbildung 3.5.: Hierarchisches Clustering eines Bauteils nach Normalvektor

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 (engl. Borderpoint), aber immer noch teil des Clusters. Wenn weniger als $minPts$ innerhalb des Radiusses sind, dann gilt der Punkt als Noise. Das bedeutet aber nicht unbedingt, dass der Punkt Noise bleibt. Er kann auch zu ei-

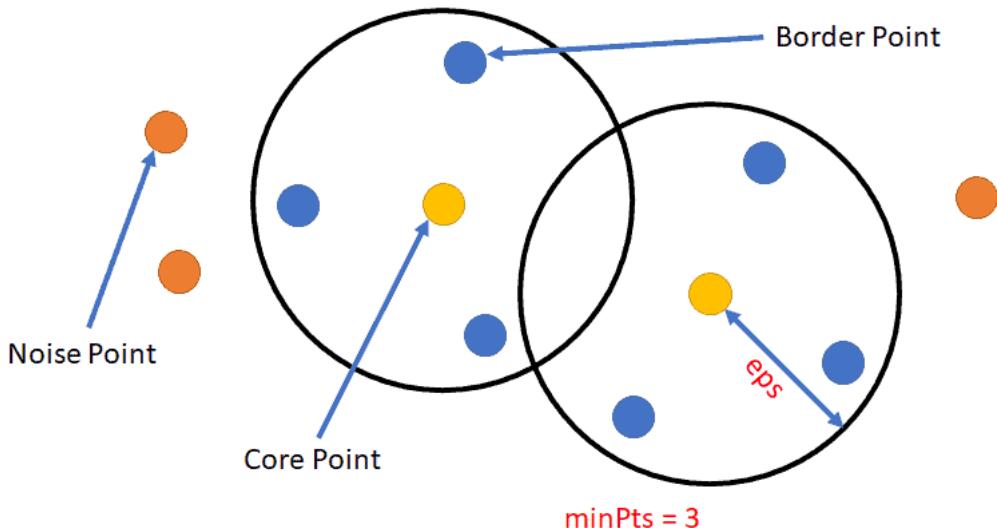


Abbildung 3.6.: Beispielhafte Darstellung von DB-Scan Algorithmus

Quelle: <https://machinelearninggeek.com/wp-content/uploads/2020/10/image-58.png>

nem 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 Cluster der Normalvektoren sind räumlich getrennt. Wenn der Algorithmus auf die vorher gefunden 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. Es gibt keine Garantie, dass alle Flächen getrennt werden, aber in den meisten Fällen wird es passieren. Ein kleinerer Radius verringert die Chance die Trennung von zwei Flächen zu übersehen, aber benötigt auch mehr Punkte, um verlässlich zu funktionieren. Mit zu wenigen Punkten könnte eine ununterbrochene Fläche fehlerhaft getrennt werden. Die Erhöhung der Punkte führt außerdem zu einer erhöhten Rechenzeit. DB-Scan ist auch auf großen Datensätzen effizient, mit einer Zeit Komplexität Relativ zu der Eingabelänge von $O(n \log n)$ [GD13]. Davor muss das hierarchische Clustering aus Abschnitt 3.2.2 ausgeführt werden. Es hat eine Zeitkomplexität von $O(n^2)$ [DE84]. Mit einer quadratischen Zeitkomplexität ist das Clustern auf den Normalvektoren der Problempunkt, durch den eine die gesamte Flächensuche eine Zeitkomplexität von $O(n^2)$ hat.

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. Der Verlust an Punkten

dadurch wird nur gering sein. Für eine echte Teilausrichtung dürften allerdings keine Punkte verloren gehen. Jeder Punkt muss gedruckt werden. Das Problem ließe sich im Realfall lösen, indem das Aussortieren weggelassen wird.

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 beschrieben. Eine Punktwolke ist eine Menge an Punkten im 3D Raum, die zusammen eine Fläche oder ein Objekt darstellen. Die Ausrichtung die berechnet wird, wird Z-Invarianz genannt. Die Z-Invarianz ist ein R^3 Vektor, der die Rotation eines Bauteils beschreibt. Dieser Vektor sollte im optimalen Fall im 90 Grad Winkel zu dem Normalvektor einer Fläche liegen, damit die Fläche möglichst parallel zu der Achse der Druckerdüse ist. Wie aus der Z-Invarianz eine Rotationsmatrix erstellt wird, wird in Abschnitt 4.2 erklärt.

In diesem Kapitel wird dargestellt, wie man eine Punktwolke, die eine Fläche in einem Bauteil darstellt, zu einer Rotationsmatrix verarbeiten kann. Für die Verarbeitung 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).

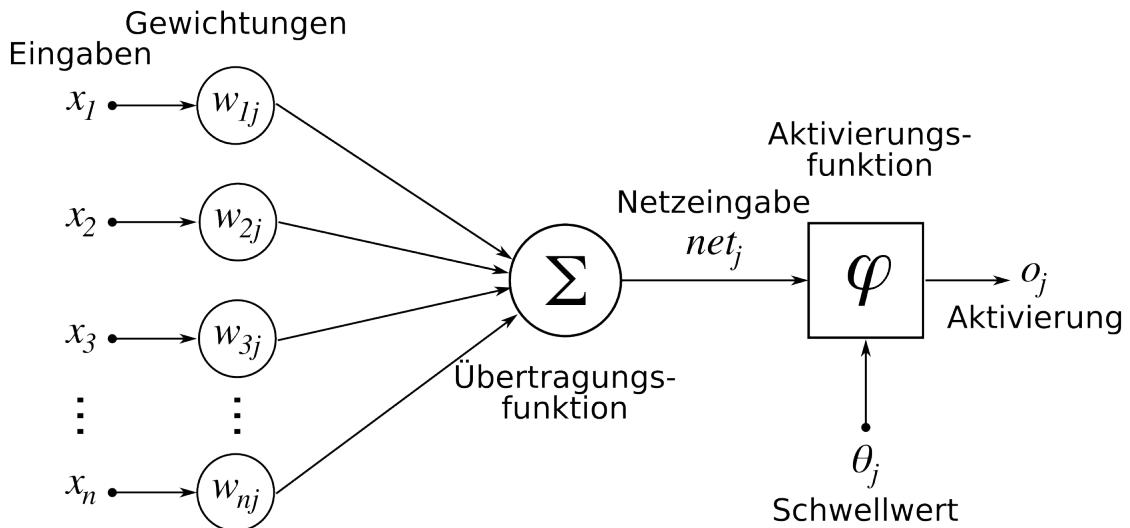
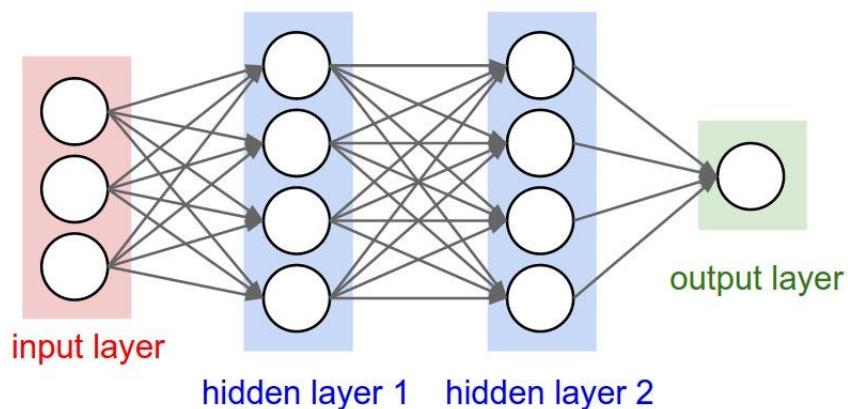
$$\sum_{i=1}^n x_i * w_i = o_j$$

Diese Funktion bildet die gewichteten Eingaben dann auf eine Ausgabe o_j mittels Aktivierungsfunktion 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 Gewichten, w_i in der Abbildung 4.1, kann ein Neuron den Einfluss von Eingabedaten (x_i) verändern. Die optimale Belegung von Gewichten wird von dem Netz über mehrere Versuche hinweg erarbeitet. Dafür muss definiert werden, was das Neurale Netz als korrektes 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 korrekten Ergebnis ist. Die Trainingsdaten brauchen für jeden Datenpunkt ein richtiges Ergebnis, damit das Neurale Netz sein Ergebnis mit dem vorher berechnet oder festgelegtem Optimum vergleichen kann. In dem Falle ist das richtige Ergebnis der Normalvektor zu der jeweiligen Punktwolke und das Netz nimmt die Standardabweichung von der Orthogonalität zu dem Normalvektor der Fläche als Fehler. Umso mehr die Z-Invarianz in einem 90 Grad Winkel zu dem Normalvektor der Fläche steht umso geringer ist der Fehler. Das Netz sieht jede Bewegung des Bauteils ebenfalls als Fehler. Umso größer die Bewegung umso der Fehler. Die beiden Fehler werden zusammenaddiert und ergeben dann den Fehler, den das Netz zu verringern versucht, indem es Gewichte anpasst. Der Fehler den die Bewegung macht, wird aber vor der Addition künstlich verringert, um die Ausrichtung der Fläche als wichtigsten Aspekt des Netzes zu behalten.

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 Kantenübergängen erreicht. Jedes Neuron der ersten verborgenen Schicht kann zum Beispiel mit nur genau einem

**Abbildung 4.1.:** Funktionsweise eines künstlichen NeuronsQuelle: https://upload.wikimedia.org/wikipedia/commons/7/7f/ArtificialNeuronModel_deutsch.png**Abbildung 4.2.:** Schematische Darstellung von RegressionsnetzQuelle: https://miro.medium.com/1*LaEgAU-vdsR_pC1McgbikQ.jpeg

Kantenübergang erreicht werden. Für eine vollständig verbundene Schicht muss jedes künstliche Neuron mit jedem künstlichen Neuron der vorherigen und darauffolgenden 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.

Nicht alle neurale Netze funktionieren über mehrere voll verbundene Schichten, aber die Idee eine Eingabe schichtweise durch das Anpassen von Gewichten zu verarbeiten kann in fast allen gefunden werden. Das für diese Arbeit genutzte neurale Netz reduziert erst die Eingabe auf Merkmale und nutzt dann mehrere voll verbundene Schichten um aus den Merkmalen eine Ausgabe zu generieren. Merkmale sind Darstellungen der Eingabedaten, die das neurale Netz verstehen kann. Merkmale sind, wenn sie von einem neuralen Netz gefunden wurden, für gewöhnlich extrem abstrakt und ohne viel Arbeit nicht für Menschen verständlich.

4.1.1. Merkmalsextraktion aus Punkt wolke

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 Extraktion der Merkmale wird in diesem Abschnitt erklärt. In der Merkmalsextraktion wird eine Punkt wolke auf eine Anzahl an Merkmalen reduziert. Die Merkmale, auf welche die Punkt wolke reduziert wird, werden in einem flachen Vektor $(v_0, v_1, v_2, \dots, v_n)$ mit $v_i \in \mathbb{R}$ der Größe n repräsentiert. Dieser Vektor wird Merkmalsvektor genannt und ist eine abstrakte Darstellung der Punkt wolken. Der Merkmalsvektor ist als Zwischenergebnis wichtig, um die Eingabedaten in für ein Regressionsnetz besser verarbeitbare Form zu bringen. Für Menschen ist der Inhalt eines Merkmalsvektors meistens unverständlich. 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: [Qi+17b]

Die generelle Struktur der Merkmalsextraktion ist eine wiederholte Folge an Reduzierungen auf den konkreten Daten der Punkt wolke, 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 Punkt wolke ausgewählt. Dafür wird ein Algorithmus eingesetzt, der eine Teilmenge der Ausgangspunkte auswählt, um eine Möglichst gute Abdeckung über die Ausgangspunkte zu erreicht (Farthest Point Sampling). Die Methode Farthest Point Sampling“ wurde in dem Artikel [Eld+97] erstellt.

Dafür geht der Algorithmus schrittweise vor. In jedem Schritt wählt er den Punkt der Punkt wolke aus, der am weitesten von allen bisher ausgewählten Punkten ent-

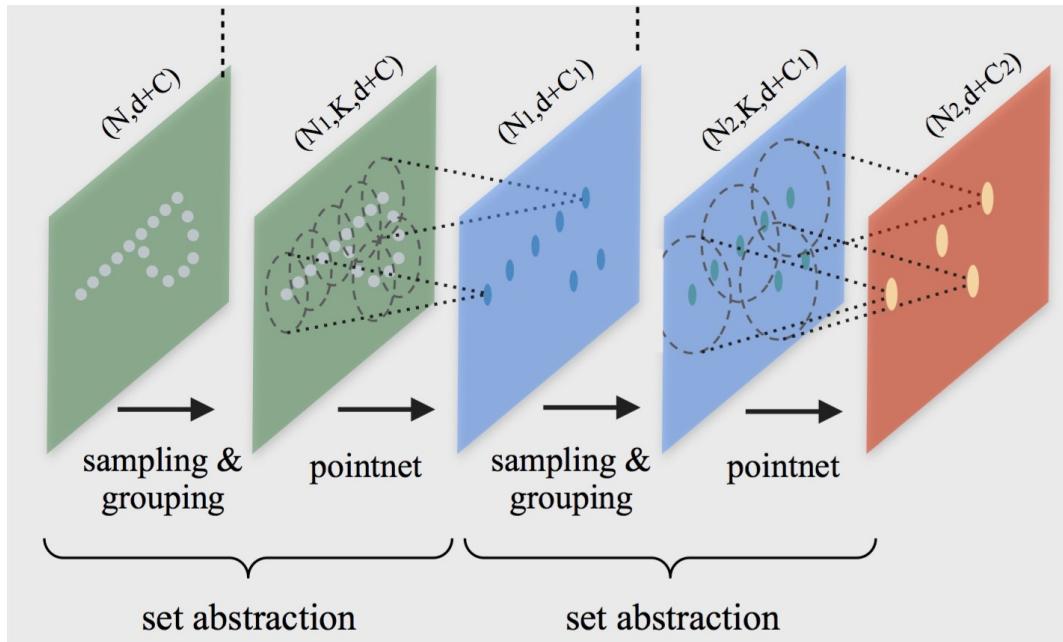


Abbildung 4.3.: Schematische Darstellung von Merkmalsextraktionschichten
Quelle: <https://stanford.edu/~rqi/pointnet2/images/pnpp.jpg>

fernt 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 minimiert, 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 wir 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. Wenn die Auswahl der Punkte A ist und die gesamte Punkt wolke P ist, dann gilt:

$$A \subseteq P, (E, V) = G, \forall \vec{a} \in A, \forall \vec{p} \in P : \|\vec{a} - \vec{p}\|_2 \leq r \implies (p, a) \in E$$

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 seinen sollen. Dafür nutzt der MLP die relative Position von ausgewählten Punkten zu ihren Nachbarn. Ein MLP funktioniert wie ein voll

verbundenes Netz, was in Abschnitt 4.1 erklärt wird. Ein ausgewählter Punkt verändert seine Merkmale anhand der Summe von Merkmalen der Punkte in seiner Nachbarschaft, die nach ihrer relativen Position mit dem MLP gewichtet werden. Die Anzahl an Punkten verringert sich, aber die Anzahl der Merkmale pro Punkt erhöht sich mit jeder Schicht.

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 gesamte 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. Der Merkmalsvektor wird über mehrere Schichten auf eine Ausgabe abgeleitet. Eine Schicht funktioniert so, dass jedes Neuron die gewichtete Summe von allen vorhergehenden 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. Es sind voll verbundene Schichten, die in Abschnitt 4.1 beschrieben wurden. In jeder Schicht werden Maßnahmen wie Normalisierung und zufälliges zurücksetzen eingesetzt, um die Regression stabiler 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.

In jeder Schicht des Regressionsnetzes wird dieselbe Aktivierungsfunktion genutzt. In

allen Schichten außer der letzten ist das $f(x) = \max(0, x)$ die als Rectifier oder ReLU bekannt ist. Es ist eine der bekanntesten Aktivierungsfunktionen, unter anderem weil sie einfach, aber effektiv ist. Für die letzte Aktivierung wird dagegen

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

genutzt. Diese Funktion wird tanh genannt. Sie hat den Vorteil, dass ihre Ausgabe immer zwischen -1 und 1 liegt, was genau das Intervall an Werten ist, die in einer Rotationsmatrix zugelassen sind. Die Ausgabe muss also nicht korrigiert werden, um für eine Rotationsmatrix zulässig zu sein.

Das Ergebnis der Regression ist die Z-Invarianz. Die Z-Invarianz ist aber noch keine Rotationsmatrix und muss daher zu einer gemacht werden, um ein Bauteil drehen zu können, was in Abschnitt 4.2 beschrieben wird.

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 Punktwolke drehen können. Eine Punktwolke bestehend aus mehreren Punkten (x_i, y_i, z_i) kann wie folgt durch Matrixmultiplikation mit einer Rotationsmatrix R gedreht werden.

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

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 Rotationsmatrix genommen werden.

Die restlichen Zeilen müssen dann aus der ersten Zeile abgeleitet werden. Die Zeilen einer Rotationsmatrix 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.

In der Abbildung 4.4 wird die Berechnung von der Normale-1 (n_1) dargestellt. Die aufgespannte Ebene ist rot gefärbt. 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 Rotationsmatrix.

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.

In der Abbildung 4.5 wird die Berechnung von der Normale-2 (n_2) dargestellt. Die aufgespannte Ebene ist rot gefärbt. 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 Rotationsmatrix vonnöten ist. Sie ist orthogonal zu den anderen Zeilen, da sei 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.

In der Abbildung 4.6 ist der Unterschied, den das Vertauschen von n_1 und n_2 machen kann dargestellt. Die Z-Invarianz wird als unterste Zeile genommen, die beiden berechneten Normalen können eine beliebige andere Zeile nehmen. Die Wahl der Zei-

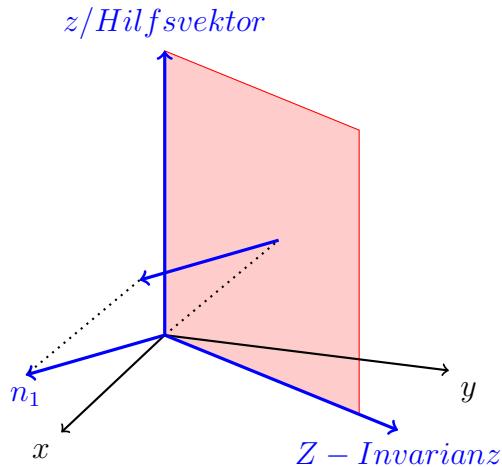


Abbildung 4.4.: Berechnung von n_1

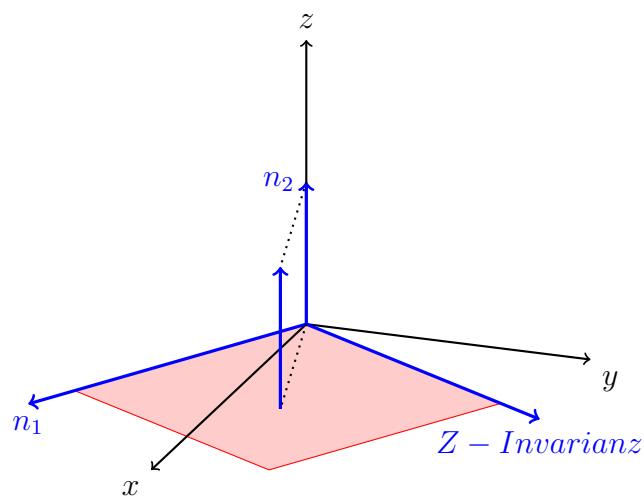


Abbildung 4.5.: Berechnung von n_2

len 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 Rotationsmatrix 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.

In der Abbildung 4.7 wird die Spiegelmatrix und die Veränderungen, die sie auf eine Rotationsmatrix wirkt dargestellt. Durch die Matrixmultiplikation mit der Spiegel-

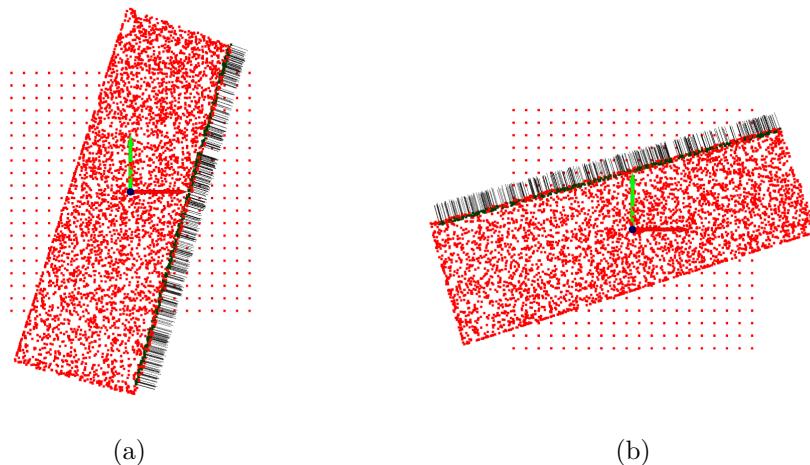


Abbildung 4.6.: Vertauschen von n_1 und n_2 in der Rotationsmatrix

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

matrix, 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.

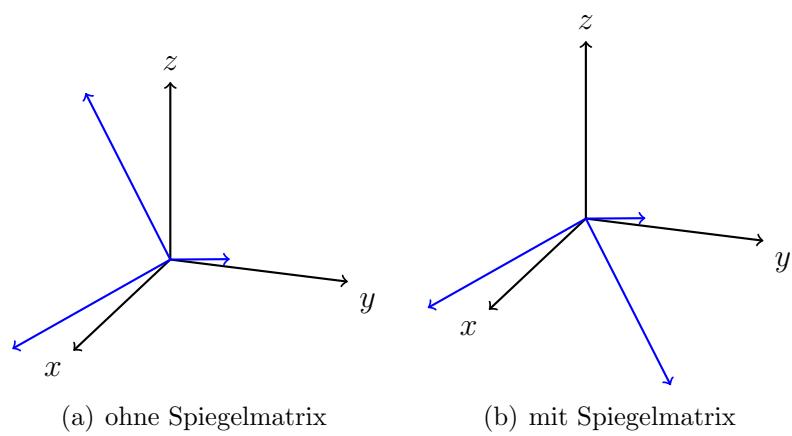


Abbildung 4.7.: Veränderung der Rotationmatrix durch die Spiegelmatrix

5. Evaluation

5.1. Versuchsaufbau

Bei 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 2.3 erwähnt eine Priorität ist. Außerdem wird die Fähigkeit des neuralen Netzes Fehler in den Daten zu widerstehen in Abschnitt 5.3 überprüft und die Fähigkeit auf komplexen Bauteilen zu arbeiten in Abschnitt 5.4 getestet.

Alle Daten, die für die Auswertung benutzt werden, sind in dem Artikel: [Zel+25b] 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. Die unterschiedlichen Typen sind: „Platten mit Löchern“, „Platonischer Körper mit Inschriften“, „Körper mit zwei Löchern und Pyramiden“.

Für alle Tests, in dem Abschnitt 5.3.1 und 5.3.2, 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 Ergebnis hat.

5.2. Ausrichtung der Bauteile

Die Ausrichtung der Bauteile ist das Ziel der Arbeit. Dementsprechend ist es die wichtigste Metrik für den Erfolg des neuronalen Netzes. Die Minimierung der Bewegung ist ebenfalls wichtig, da jede Drehung eines Bauteils Zeit kostet. Als zweites Ziel des Netzes wird dementsprechend versucht, die Größe der Drehung des Bauteils zu minimieren.

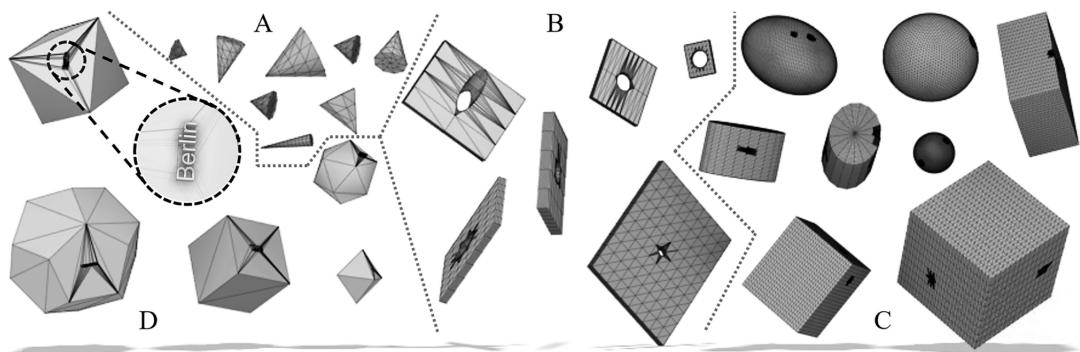


Abbildung 5.1.: Darstellung aller Typen von Bauteilen im Datensatz
Quelle: https://media.springernature.com/full/springer-static/image/art%3A10.1007%2Fs40964-025-00960-6/MediaObjects/40964_2025_960_Fig1_HTML.png

Bauteil	Gradabweichung vom Optimum	Bewegung des Bauteils
Platten mit Löchern	10.12	23.56
Platonischer Körper	9.59	26.50
Körper mit zwei Löchern	9.40	27.11
Pyramiden	9.22	24.68

Tabelle 5.1.: Testergebnisse auf einfachen Bauteilen

Die Tabelle 5.1 zeigt die Ergebnisse der Tests auf den einfachen Bauteilen des Datensatzes. Dabei ist jeder Typ von Bauteil einzeln aufgeführt. Es lässt sich erkennen, dass alle Bauteile außer die Platten mit Löchern eine Genauigkeit von unter 10 Grad Abweichung vom Optimum erreichen. Die Platten mit Löchern erreichen eine Genauigkeit von 10.12 Grad Abweichung vom Optimum, was nicht viel höher ist. Insgesamt lässt sich kein größer Unterschied zwischen den verschiedenen Typen von Bauteilen erkennen. Alle Typen erreichen eine ähnliche Genauigkeit und benötigen eine ähnliche Bewegung des Bauteils. Die Pyramiden erreichen die beste Genauigkeit mit einer Abweichung von 9.22 Grad vom Optimum und einer durchschnittlichen Bewegung von 24.68 Grad. Die Pyramiden sind die einfachsten Bauteile im Datensatz, was die gute Genauigkeit erklären könnte. Andererseits sind die Platten mit Löchern die am schlechtesten abschneidenden Bauteile, was die Gradabweichung vom Optimum angeht. Das könnte daran liegen, dass die Flächensuche Probleme mit den Löchern in den Platten hat.

Der vorherige Test wurde durchgeführt, sodass sowohl auf die Minimierung der Bewegung als auch auf die optimale Ausrichtung geachtet wird. Es ist allerdings auch relevant, wie gut das Netz eine Fläche ausrichten kann, wenn einen die Größe der Bewegungen nicht interessiert.

Bauteil	Gradabweichung vom Optimum	Bewegung des Bauteils
Platten mit Löchern	5.42	59.19
Platonischer Körper	5.67	58.28
Körper mit zwei Löchern	4.85	61.02
Pyramiden	4.26	58.79

Tabelle 5.2.: Testergebnisse auf einfachen Bauteilen ohne Berücksichtigung von der Bewegung des Bauteils

Es lassen sich starke Verbesserungen in 5.2 was die Ausrichtung angeht erkennen, wenn man es mit 5.1 vergleicht. Die Ergebnisse haben sich durch das Vernachlässigen der Bewegung des Bauteils halbiert und haben jetzt eine Abweichung von in etwa 5 Grad. Die Platten mit Löchern sind immer noch der am schlechtesten abschneidenden Typ von Bauteil und die Pyramiden sind immer noch die am besten Abschneidenden. Dafür, dass sich die Abweichung vom Optimum halbiert hat, hat sich die Bewegung des Bauteils mehr als verdoppelt. Da sie von dem Netz für diesen Test nicht beachtet wurde, war diese Veränderung zu erwarten.

5.3. Verlässlichkeit der Ausrichtung

Ein wichtiger Unterschied zwischen dem Trainingsdatensatz und der Praxis ist, dass man in der Praxis nicht davon ausgehen kann, dass die Daten einwandfrei sind. Um ein Praxisrelevantes Verfahren zu entwickeln, muss das Verfahren in der Lage sein, Fehler in den Daten zu widerstehen. In dem bisherigen Test ist immer davon ausgegangen worden, dass die Punktwolken eine perfekte Darstellung eines Bauteils ist. In der Praxis, muss aber davon ausgegangen werden, dass die Punktwolken Fehler aufweisen. Um diese Fehler zu simulieren, wird in den folgenden zwei Abschnitten der Einfluss von fehlenden Punkten und der Einfluss von falschen Punkten untersucht.

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 um das Zentrum der Fläche, 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 Normalvektors 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 Standardabweichung des Normalvektors hat das Rauschen allerdings einen extremen Einfluss. Sobald Rauschen hinzugefügt wird, steigt die Standardabweichung 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 stabil, aber da die Abweichung von diesem Durchschnitt extrem ansteigt, ist das Ergebnis unzuverlässig.

Der extreme Anstieg, der Standardabweichung in Abbildung 5.2 könnte dadurch erklärt werden, dass das Netz nur auf fehlerfreien Daten trainiert wurde. Die Rauschpunkte eliminieren die glatte Struktur der Fläche, aber die Merkmalsextraktion wird dennoch versuchen diese Punkte mit einzubeziehen und dadurch auf falsche Merkmale kommen. Das erklärt ebenfalls, warum die Standardabweichung nach dem ersten Anstieg konstant bleibt. Sobald die Merkmale falsch sind, ändert weiteres Rauschen nichts mehr an den falschen Merkmalen. Die Ergebnisse sind dennoch besser als Zufällig, was darauf hindeutet, dass das Netz trotz des Rauschens noch die Originale Fläche erkennt, aber die Rauschpunkte eine große Unsicherheit in die Ergebnisse bringen.

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 Punktwolke keiner macht als sie es mit einer perfekten Suche wäre. Der Einfluss von fehlenden Punkten in der Punktwolke einer Fläche wird getestet, indem über mehrere Iterationen Punkte aus den Punktwolken

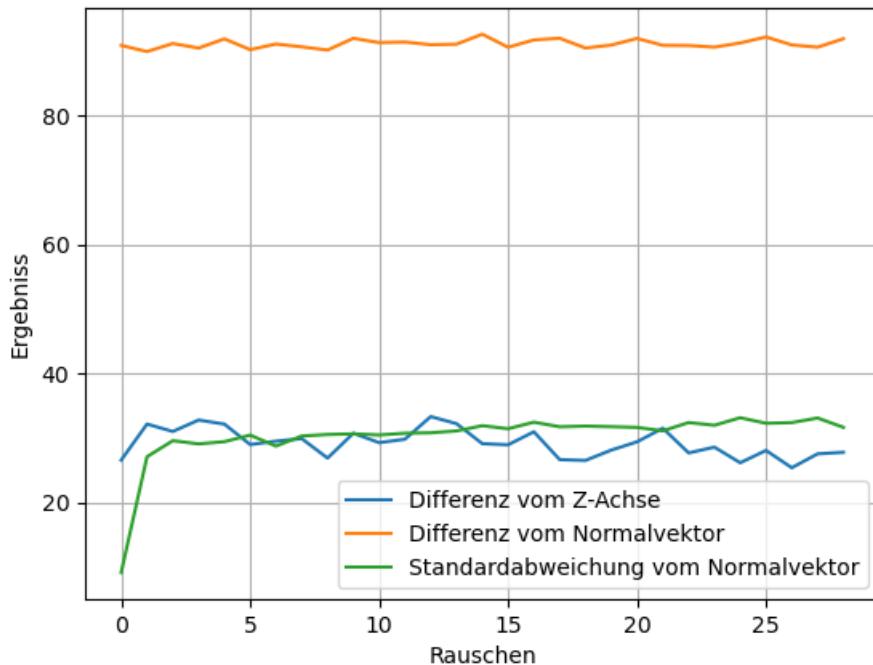


Abbildung 5.2.: Test mit Noise auf dem Neuralen Netz

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. Alle Ergebnisse sind in Grad angegeben. Auf der x-Achse ist der Prozentsatz der entfernten Punkte dargestellt, auf der y-Achse ist das Ergebnis der Tests dargestellt. Die Abweichung von dem Normalvektor der Fläche ist nahezu unverändert, wenn Punkte entfernt werden. Die Standardabweichung des Normalvektors bleibt für die ersten 80 %, ebenfalls fast unverändert, und steigt nur leicht an. Erst nachdem 80 % 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 Punktfolge zu widerstehen. Ein Verlust von über 80 % der Punkte ist extrem unwahrscheinlich in der Praxis. Das Netz zeigt also eine gute Widerstandsfähigkeit gegen schlechte Flächensuche.

Auch bei der Differenz zu der z-Achse oder der Bewegung des Bauteils ist kein großer Einfluss von fehlenden Punkten zu erkennen, bis 80 % der Punkte entfernt

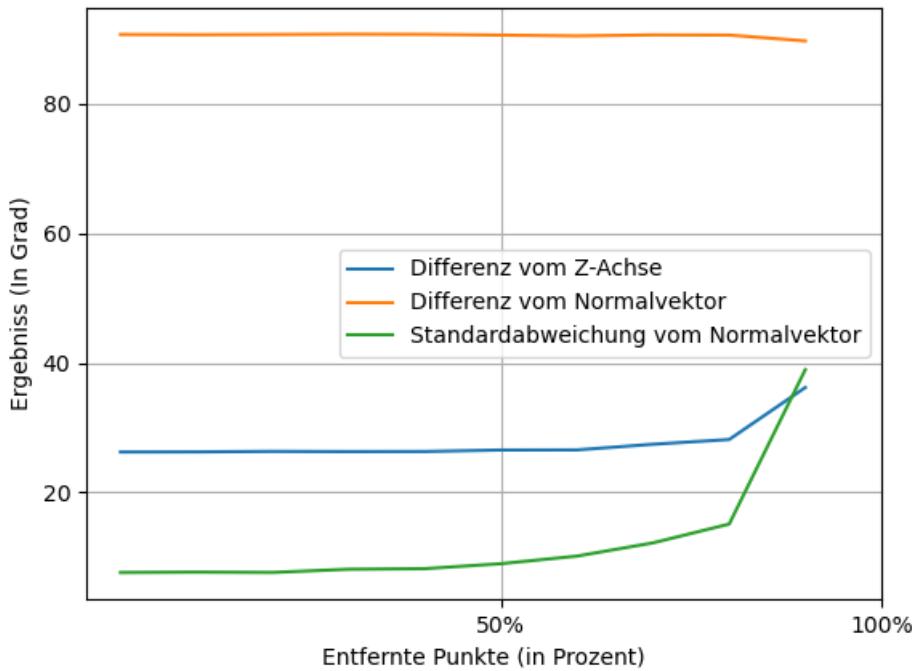


Abbildung 5.3.: Test von schlechter Flächensuche auf dem Neuralen Netz

wurden. Danach hat auch hier der Verlust an Punkten einen großen Einfluss auf die Genauigkeit des Netzes, allerdings ist der Anstieg deutlich geringer im Vergleich zu der Standardabweichung. Der Punkt, an dem die Genauigkeit stark abfällt, ist bei der Standardabweichung und der Differenz zu der z-Achse gleich. Das deutet darauf hin, dass der Abfall der Genauigkeit denselben Grund hat.

Der Zerfall der Genauigkeit in Abbildung 5.3 könnte dadurch erklärt werden, dass die Merkmalsextraktion darauf basiert, dass eine Menge an Punkten Merkmale besitzt, die in jedem Schritt auf eine kleinere Menge an Punkten reduziert werden. Wenn zu viele Punkte fehlen, dann wird das Netz weniger Punkte für die Reduktionen haben, was den langsamem Zerfall der Standardabweichung bis zu der 80 % Grenze erklären könnte. Warum es danach so stark abfällt, ist vermutlich damit zu erklären, dass die Dichte der Punkte so stark abgenommen hat, dass die Reduktion häufig keine Punkte in der lokalen Umgebung hat, die sie für eine Reduzierung verwenden kann. Wenn die Merkmale von einem oder sehr wenigen Punkten aus reduziert werden, denn zerfällt die Idee, lokale Merkmale anhand Ihrer relativen Position zu andern Punkten zu bewerten. Das neurale Netz verliert die Fähigkeit, lokale Merkmale zu erkennen.

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.

Die Abbildung 5.4 sind die 3d-Modelle aller komplexen Bauteilen abgebildet. Die Abbildung A zeigt den Stanford Hasen (engl. Stanford Bunny). Es ist asymmetrisch und hat an den Ohren dünne gekrümmte Flächen. Der gesamte Körper ist mit Fell bedeckt, dadurch wird eine schon gebogene Fläche mit mehreren Unebenheiten überdeckt. Der Stanford Drache ist ebenfalls asymmetrisch. Seine Oberfläche ist gebogen und ist durch viele Spitzen auf der Oberfläche nicht glatt. Es hat ebenfalls mehrere Stellen, wie der Kopf oder die Mitte des Körpers, an denen der Körper keine nahegelegene Stütze vom Körper selber hat. Zuletzt ist der Utah Teepott (engl. Utah Teapot) das einzige komplexe Bauteil, was zum größten Teil symmetrisch ist. Der Teepott hat eine glatte Oberfläche, die häufig nicht gekrümmmt ist. Die Bauteile eignen sich als Beispiel für komplexe Bauteile, da sie meistens aus unebenen und gebogenen Flächen bestehen. Der Teepott ist weniger komplex, aber immer noch komplexer als die einfachen Daten in dem Abschnitt 5.2. Es ist ein Alltagsgegenstand und kann als Vergleichswert dienen.

Alle Angaben in der Tabelle 5.3 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 5.2 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.



Quelle: <https://graphics.stanford.edu/data/3Dscanrep/stanford-bunny-ceb>

(a)



Quelle: <https://graphics.stanford.edu/data/3Dscanrep/dragon.jpg>

(b)



Quelle: <https://graphics.stanford.edu/data/3Dscanrep/dragon.jpg>

(c)

Abbildung 5.4.: 3d-Modelle der komplexen Bauteile

Der Teepott hat keine Signifikat anderen Ergebnisse, im Vergleich zu den anderen zwei Bauteilen, obwohl es eine deutlich einfacherer Form hat.

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 5.3.: Testergebnisse auf komplexen Bauteilen

6. Zusammenfassung und Ausblick

6.1. Zusammenfassung

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 beschrieben neuronalen Netz durchgeführt und ihre Ergebnisse wurden ausgewertet. Daraus ergab sich, dass das Netz in der Lage ist, Bauteile mit einer akzeptablen, aber bei weitem nicht perfekten, Genauigkeit auszurichten. Eine Abweichung von unter 10 Grad ist im Durchschnitt 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. Eine durchschnittliche Abweichung von 10 Grad ist für den Praxisfall häufig zu viel. 10 Grad reichen aus, um einer Fläche einen Überhang zu geben, was dem Sinn des 5d-Drucks widersprechen würde. Das Ergebnis, wenn die Minimierung der Bewegung ignoriert wird ist in der Hinsicht vielversprechender mit 5 Grad. Es würde allerdings auch eine signifikante Erhöhung der Druckzeit mit sich bringen, da sich die Bewegung für jede Ausrichtung mehr als verdoppelt hat. Die Ausrichtung ist in beiden Fällen nicht perfekt, für eine praktische Anwendung würde man bessere Ergebnisse wollen, aber Neurale Netz sind dennoch in der Lage eine Ausrichtung mit akzeptabler Abweichung zu finden.

Die Minimierung der Bewegung des Bauteils während des Ausrichtens hatte eine kleine, aber sichtbaren Verkleinerung der benötigten Drehung ergeben. Der durch-

schnittliche Winkel, um den ein Teil gedreht werden muss, lag bei unter 40 Grad. Ohne die Minimierung würde der durchschnittliche Drehwinkel bei über 45 Grad 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. Insgesamt, ist die Verbesserung über dem Zufall nicht sehr groß, aber eine extreme Verbesserung über dem Fall, in dem die Bewegung des Bauteils ignoriert wurde.

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, die Ausrichtung von Bauteilen zu erkennen, ob die Abweichung gering genug ist um Praxisrelevanz zu sein ist fragwürdig. 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 zusammengefasst für den praktischen Einsatz unter Umständen praktikabel. Bei einer Abweichung von bis zu 40 Grad ist das Drucken immer noch möglich, wenn auch nicht sehr schnell [Jia+18].

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 vorge-

nommen 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

1.1.	Beispiel für einen 5d-Drucker	2
2.1.	Schematische Darstellung eines 3d-Druckers	4
2.2.	Vorschau eines Bauteils mit Stützstrukturen	5
2.3.	Darstellung der Unterschiede im Bauteil abhängig von der Höhe der Schichten	7
2.4.	Darstellung der beweglichen Plattform eines 5d-Druckers	7
3.1.	Darstellung von Tangenten in einer gekrümmten Fläche	10
3.2.	Clustering von verstreuten zweidimensionalen Punkten	11
3.3.	Darstellung von Normalvektoren an einem Punkt der Bauteiloberfläche	12
3.4.	Beispiel von Hierarchie Clustering als Dendrogramm	13
3.5.	Hierarchisches Clustering eines Bauteils nach Normalvektor	15
3.6.	Beispielhafte Darstellung von DB-Scan Algorithmus	16
4.1.	Funktionsweise eines künstlichen Neurons	21
4.2.	Schematische Darstellung von Regressionsnetz	21
4.3.	Schematische Darstellung von Merkmalsextraktionschichten	23
4.4.	Berechnung von n_1	27
4.5.	Berechnung von n_2	27
4.6.	Vertauschen von n_1 und n_2 in der Rotationsmatrix	28
4.7.	Veränderung der Rotationmatrix durch die Spiegelmatrix	29
5.1.	Darstellung aller Typen von Bauteilen im Datensatz	32
5.2.	Test mit Noise auf dem Neuralen Netz	35
5.3.	Test von schlechter Flächensuche auf dem Neuralen Netz	36
5.4.	3d-Modelle der komplexen Bauteile	38

Algorithmenverzeichnis

Quellcodeverzeichnis

Literatur

- [DE84] William HE Day und Herbert Edelsbrunner. „Efficient algorithms for agglomerative hierarchical clustering methods“. In: *Journal of classification* 1.1 (1984), S. 7–24.
- [Eld+97] Yuval Eldar, Michael Lindenbaum, Moshe Porat und Yehoshua Y Zeevi. „The farthest point strategy for progressive image sampling“. In: *IEEE transactions on image processing* 6.9 (1997), S. 1305–1315.
- [Est+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu u. a. „A density-based algorithm for discovering clusters in large spatial databases with noise“. In: *kdd*. Bd. 96. 34. 1996, S. 226–231.
- [Fas12] Petra Fästermann. *3d-druck/rapid prototyping: Eine zukunftstechnologie-kompakt erklärt*. Springer-Verlag, 2012.
- [GD13] Ade Gunawan und M De Berg. „A faster algorithm for DBSCAN“. In: *Master's thesis* (2013).
- [Jia+18] Jingchao Jiang, Jonathan Stringer, Xun Xu und Ray Y Zhong. „Investigation of printable threshold overhang angle in extrusion-based additive manufacturing for reducing support waste“. In: *International Journal of Computer Integrated Manufacturing* 31.10 (2018), S. 961–969.
- [KM21] Nikhil Ketkar und Jojo Moolayil. „Convolutional neural networks“. In: *Deep learning with Python: learn best practices of deep learning models with PyTorch*. Springer, 2021, S. 197–242.
- [Qi+17a] Charles R Qi, Hao Su, Kaichun Mo und Leonidas J Guibas. „Pointnet: Deep learning on point sets for 3d classification and segmentation“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 652–660.

- [Qi+17b] Charles Ruizhongtai Qi, Li Yi, Hao Su und Leonidas J Guibas. „Point-Net++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space“. In: *Advances in Neural Information Processing Systems*. Hrsg. von I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan und R. Garnett. Bd. 30. Curran Associates, Inc., 2017.
- [SLR19] Nurhalida Shahrubudin, Te Chuan Lee und RJPM Ramlan. „An overview on 3D printing technology: Technological, materials, and applications“. In: *Procedia manufacturing* 35 (2019), S. 1286–1296.
- [Sta15] IOF Standardization. „Additive manufacturing: general: principles“. In: *Terminology* 640 (2015).
- [Wan03] Sun-Chong Wang. „Artificial Neural Network“. In: *Interdisciplinary Computing in Java Programming*. Boston, MA: Springer US, 2003, S. 81–100.
- [War63] Joe H Ward Jr. „Hierarchical grouping to optimize an objective function“. In: *Journal of the American statistical association* 58.301 (1963), S. 236–244.
- [Zel+25a] Sebastian Zelder, Iryna Shevchenko, Joschka zur Jacobsmühlen, Jörg Krüger und Eckart Uhlmann. „Learning automatic part orientation for powder bed fusion of metal from part data“. In: *Progress in Additive Manufacturing* 10.3 (2025), S. 1713–1719.
- [Zel+25b] Sebastian Zelder, Iryna Shevchenko, Joschka zur Jacobsmühlen and Jörg Krüger und Eckart Uhlmann. *Learning automatic part orientation for powder bed fusion of metal from part data*. <https://doi.org/10.1007/s40964-025-00960-6>. Zugriff am: 15.12.2025. 2025.
- [Zhu+24] Zicheng Zhuang, Fengming Xu, Junhong Ye, Nan Hu, Liming Jiang und Yiwei Weng. „A comprehensive review of sustainable materials and tool-path optimization in 3D concrete printing“. In: *npj Materials Sustainability* 2.1 (2024), S. 12.

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:

- LanguangeTool, LanguangeTool, 1.0.1, Bezugsquelle, <https://languagetool.org/de>
- Hilfsmittel, Produktnamen, Version, Bezugsquelle, [Webseite](#)
- Hilfsmittel, Produktnamen, Version, Bezugsquelle, [Webseite](#)