Universität Passau

– Masterarbeit –

3D - Oberflächen - Rekonstruktion

Von der Punktewolke zum Oberflächennetz

Florian Himmelsbach

2022

Betreuer:
Prof. Dr. Tomas Sauer*

*Lehrstuhl für Mathematik mit Schwerpunkt Digitale Bildverarbeitung

Zusammenfassung

Diese Arbeit befasst sich insgesamt mit der Zusammenstellung einzelner Algorithmen im Bezug zur dreidimensionalen Oberflächen Rekonstruktion. Neben einer allgemeinen Einführung in das Thema der Rekonstruktion, werden Algorithmen hinsichtlich ihrer Strategie in Klassen eingeteilt. Stellvertretend für die einzelnen Ideen zur Rekonstruktion werden die Algorithmen *Marching Cubes*, *Poisson Oberflächen Rekonstruktion*, *Ball Pivoting Algorithmus*, *Scale Space Meshing* und der *Power Crust Algorithmus* genauer betrachtet. Die einzelnen Varianten werden erläutert und folglich hinsichtlich deren Eigenschaften mit eigenen Anwendungsbeispielen verglichen. Hierbei wird auf den Glättungsfaktor der Rekonstruktionen eingegangen. Ebenso wird untersucht, ob die Algorithmen potentiell Löcher in der Oberfläche aufweisen können oder diese schließen. Der Umgang mit Eingangsdaten, welche nicht der Abtastrate entsprechen oder verrauscht sind, wird ebenfalls untersucht. Schließlich wird auch auf die primitiven Flächen der Oberflächenrepräsentationen eingegangen. Das Oberflächennetz kann hierbei ein allgemeines Polygonnetz sein, wie auch speziell eine Triangulierung, welche nur noch aus Dreiecken besteht. Die Größe dieser primitiven Fläche und die verbundene Speicherkapazität ist ebenfalls ein Thema.

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Seit einigen Jahren beschäftigt sich die Wissenschaft mit der Thematik Objekte und Oberflächen aus zuvor gescannten Punktewolken zu rekonstruieren. Abgesehen aus Gründen mathematischer und geometrischer Modellierung, war eines der ursprüngliche Aufgabengebiete die Visualisierung von medizinischen Datensätzen. Verschiedenste Anwendungen, wie zum Beispiel MRT oder CT, liefern dreidimensionale Daten, welche verschiedenste Oberflächen repräsentieren, die aus Behandlungsgründen ausgewertet werden sollen. Neben der Medizin hat die Industrie in den letzten Jahren ebenfalls Anwendungsgebiete für dreidimensionale Rekonstruktionen gefunden. Neben Reverse Engineering, um bestehende Bauteile wieder rekonstruieren zu können, ist ein großes Anwendungsgebiet das automatische Erstellen von CAD Modellen. Diese spielen aufgrund von Visualisierung, Modellierung, Architektur, Ingenieurswesen, wie zur Vorverarbeitung für die Fertigung eine wichtige Rolle. Speziell die Technologie rund um den 3D-Druck ist eine Fertigungsvariante, die in diesem Zusammenhang für Industrie, wie aktuell auch für privat Personen, ein großes Interesse zeigt. Der Vorteil ein reales Objekt scannen zu können, ein Modell automatisch generieren zu lassen und dieses dann wiederum real zu rekonstruieren spricht für sich selbst. Weitere Anwendungsgebiete sind Archäologie und Kunst um Modelle von bestehenden Ausgrabungen oder auch Kunstwerken anzufertigen, ohne das Objekt selbst zu berühren. Dies geschieht aus Sicherheitsgründen um das reale Objekt nicht zu zerstören. Neuere Forschungen in Verbindung mit LIDAR Scannern befassen sich ebenfalls mit Erdoberflächen-Rekonstruktion, wie auch Umgebungsrekonstruktion. Die Umgebungsrekonstruktion spielt eine wichtige Rolle für autonomes Fahren, wie auch Sicherheitsaspekte, wie zum Beispiel Objekterkennung. Insgesamt gibt es somit eine Vielzahl an Anwendungsbereichen, in denen anhand einer gescannten Punktewolke ein Objekt bzw. eine Oberfläche rekonstruiert werden soll. Da hierfür verschiedenste Techniken erforscht wurden, ist der Zweck der Ar-

beit ein Überblickswissen über die dreidimensionale Oberflächen-Rekonstruktion zu liefern. In diesem Zusammenhang werden einige Algorithmen vorgestellt und anhand eigener Versuche verglichen.

1.2 Inhaltsübersicht

Diese Arbeit befasst sich insgesamt mit der Zusammenstellung einzelner Algorithmen im Bezug zur dreidimensionalen Oberflächen Rekonstruktion.

Um in die Aufgabenstellung einer Rekonstruktion allgemein eingeführt zu werden, beinhalten die Kapitel *Problematik & Schwierigkeit* und *Allgemeine Verarbeitungsschritte* einige triviale Probleme der Messtechnik, wie auch der Verarbeitung von Messwerten. Anhand dieser beiden Kapitel sollte einerseits verstanden werden, warum es verschiedene Strategien für eine Rekonstruktion gibt. Andererseits wird eine Verarbeitungspipeline vorgestellt, anhand welcher der Zeitpunkt der Rekonstruktionsalgorithmen geklärt wird. Zudem werden Prozesse zur Unterstützung dieser angeschnitten.

Folglich schließt das Kapitel *Rekonstruktionsklassen* an. In diesem Kapitel werden allgemein verschiedene Strategien vorgestellt. Anhand dieser Klassen wurden jeweils Vertreter ausgewählt, welche in den folgenden Kapiteln genauer erläutert werden. Die speziell vorgestellten Algorithmen sind: *Marching Cubes*, *Poisson Oberflächen Rekonstruktion*, *Ball Pivoting Algorithmus*, *Scale Space Meshing* und der *Power Crust Algorithmus*.

Nachdem die genannten Verfahren genauer erläutert wurden, werden im Kapitel *Anwendungen & Ergebnisse* meine eigenen Versuche hinsichtlich dreidimensionaler Oberflächen Rekonstruktion vorgestellt. Hierzu wird zunächst geklärt, wo meine Eingangsdaten ihren Ursprung haben. Anschließend werden Ergebnisse meiner Anwendungen vorgestellt und hinsichtlich deren Eigenschaften eingeteilt.

Schließlich beinhaltet meine Arbeit noch Abbildungshinweise, eine Übersicht der erwähnten Datensätze, ein Literaturverzeichnis und ein Abbildungsverzeichnis.

2 Problematik & Schwierigkeit

Seitdem sich Wissenschaftler mit dem Thema Rekonstruktion beschäftigen, müssen Sie sich auch mit verschiedenen Problemen befassen. Die einzelnen Probleme und Schwierigkeiten haben unterschiedlichen Ursprung. Speziell im Bereich der dreidimensionalen Oberflächen Rekonstruktion gibt es nun eine Vielzahl verschiedener Strategien eine gewünschte Oberfläche zu erhalten. Der Grund für diese Anzahl an Algorithmen ist, dass diese für bestimmte Endergebnisse bzw. auch Eigenschaften des Inputs individuell geeignet sind. Zunächst einmal die Formulierung des Hauptproblems bzw. des zu lösenden Vorgangs.

2.1 Problem Deklaration

Der zu lösende Vorgang führt über einen bestimmten Input zu einem gewünschten Output. Der Eingang für eine Rekonstruktion ist eine abgetastete Punktewolke P, die anhand einer Oberfläche S aufgenommen wurde. Hierbei werden bereits wissenschaftliche Anforderungen an die Abtastung gestellt. Das reale Objekt sollte möglichst genau dargestellt werden.

Der Übergang zwischen physikalischer Messung, kontinuierlichem Wert und einer schließlich digitalen Darstellung formuliert die ersten Probleme der Rekonstruktion. Abgesehen von technischen Fehlern müssen zudem reale Welt Probleme im Umgang mit einem Scann betrachtet werden. Um das Problem schließlich fertig zu deklarieren, wird anhand der Punktewolke P versucht eine neue Oberfläche M anzunähern. Diese sollte der ursprünglichen Oberfläche S zu einem hohen Grad ähneln.

Im Folgenden werden zunächst einige reale Probleme hinsichtlich der Messwerte selbst geschildert. Anschließend wird die Deklaration des Problems so erläutert, dass man verstehen kann, warum man nicht explizit „die“ Oberfläche, sondern „eine“ Oberfläche rekonstruiert wird.

2.2 Probleme mit Messwerten

Abb. 2.1: Problematik unterschiedliche Qualität der Messwerte

Dass in der Realität Messwerte immer zu einem gewissen Grad fehleranfällig sind ist bekannt. Um die einzelnen Probleme eines dreidimensionalen Scann genauer zu verstehen, werden einiger dieser explizit in Abbildung 2.1 im Zweidimensionalen veranschaulicht. Für die Visualisierung wird eine zweidimensionale Kurve anhand der nachfolgenden, unterschiedlichen Messwerte dargestellt.

- Die erste Menge an Punkten zeigt eine nahezu ideale Abtastung der Kurve. Das Abtasttheorem wurde eingehalten. Alle Frequenzen können rekonstruiert werden. Die Abtastdichte ist konstant.
- Die zweite Punktemenge erfüllt ebenfalls das Abtasttheorem. Zusätzlich hohe Frequenzen in den Messwerten repräsentieren weißes Rauschen.
- Die dritte Menge erfüllt ebenfalls noch das Abtasttheorem. Der Fokus ist hierbei auf eine Vielzahl an Ausreißern, welche die ursprüngliche Kurve garnicht beschreiben.
- Die Punkte der vierten Variante zeigen eine variable Abtastdichte. Insgesamt ist das Abtasttheorem jedoch noch erfüllt.
- Die vorletzte Menge weist im Unterschied zur vorherigen Menge in den meisten Bereichen eine konstante Abtastdichte auf. Jedoch sind in zwei Positionen große Löcher. Diese Bereiche spiegeln fehlende Messwerte wieder.
- Zuletzt wird noch veranschaulicht, dass verrauschte Daten nicht zwangsweise überall gleich verrauscht sein müssen. Das Rauschen kann in unterschiedlichen Bereichen variieren.

Abgesehen von der idealen Abtastung kann man die Behauptung aufstellen, dass die verschiedenen Ausgangslagen besondere Anforderungen an die spätere Verarbeitung stellen. Wird das Abtasttheorem generell nicht eingehalten, kann überhaupt nicht garantiert werden, dass die Rekonstruktion ein Erfolg ist. Die Behauptung aufzustellen, dass ein Scann eines realen Objektes eine ideale Repräsentation dessen ist, würde aus wissenschaftlicher Sicht keinen Sinn machen. Speziell eine Oberfläche kann aufgrund der Abtastdichte immer nur Frequenzen, welche die Hälfte ihrer selbst entsprechen repräsentieren. Somit muss bereits vor einem Scann klar sein, welche Features der Oberfläche wie genau dargestellt werden möchten.

Zusätzlich kommt in einem realen Umfeld hinzu, dass aufgrund technischer Möglichkeiten, wie auch der Diskretisierung der Messwerte die Punktgenauigkeit leidet. Technische Möglichkeiten beziehen sich hierbei auf den verwendeten Scanner, wie auch die Beschaffenheit der zu scannenden Oberfläche.

Allgemein kann man davon ausgehen, dass die Punktewolke jede Art der vorgestellten Problematiken aufweist. Die Frage ist eher zu welchem Ausmaß dies passiert ist. Aufgrund dessen sind eben unterschiedliche Strategien zur Rekonstruktion entstanden, welche die verschiedenen Ausgangslagen unterschiedlich tolerieren.

2.3 Kandidaten für Rekonstruktion

Nachdem zuvor erläutert wurde, dass die Messwerte spezielle Anforderungen darstellen, wird nun erklärt, warum eine bestimmte Punktewolke unterschiedliche Oberflächen erzeugen kann. Die Abbildung 2.2 zeigt eine zweidimensionale Punktemenge, welche einen Pfeil darstellen sollte.

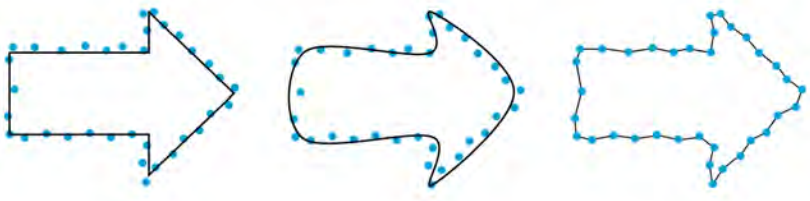

Abb. 2.2: Mögliche Kandidaten für Rekonstruktionsproblem

Die reale Form zur Punktemenge ist nicht bekannt. Zudem könnte man auch davon ausgehen, dass man gegebenenfalls nicht einmal weis, dass die Punkte insgesamt einen Pfeil repräsentieren. Anhand der Messwerte in Abbildung 2.2 ist nicht feststellbar, welche Rekonstruktion am wahrscheinlichsten die ursprüngliche Form darstellt.

Fragen

- Sollten bestimmte Bereiche mithilfe einer Regressionskurve erzeugt werden um einen Trend festzustellen?
- Sollten die Messwerte explizit als zur Kurve gehörend definiert und einfach linear verbunden werden?
- Sollte diese Interpolation mit einem höheren Grad definiert werden?
- Muss die Punktemenge zunächst geglättet werden?
- Sollte die Punktewolke anhand der Krümmung verändert werden?
- Muss man bereits vor der Rekonstruktion Ausreißer entfernen?
- Kann man überhaupt vor der Rekonstruktion bereits Ausreißer entfernen?
- ...

Mit eben solchen Fragen beschäftigten sich die Autoren, deren Rekonstruktionen in folgenden Kapiteln erläutert wird. Eine richtige Antwort gibt es in vielen Fällen nicht. Der Grad an Ausreißern oder Rauschen ist schwierig allgemein festzustellen, speziell wenn die ursprüngliche, zu rekonstruierende Oberfläche nicht bekannt ist.

Um so mehr Informationen über das Ursprungsobjekt bzw. den verwendeten Laser oder der gescannten Oberfläche bekannt ist, um so leichter kann man schließlich abschätzen, ob das Ergebnis das Objekt ausreichen darstellt. Ebenso kann aufgrund des erhofften Ergebnisses entschieden werden, welche Art der Rekonstruktion verwendet wird. Zu diesem Aspekt werden in folgenden Kapiteln die Eigenschaften der Algorithmen genauer vorgestellt.

Insgesamt kann man somit behaupten, dass die Aufgabenstellung nicht ist „die“ Oberfläche, sondern „eine“ Oberfläche zu rekonstruieren, welche anhand der Messwerte eine gute Annäherung ergibt. Im Englischen wird in diesem Zusammenhang das Problem gern als „ill-posed“ beschrieben, da man grundsätzlich immer darüber spricht die genaue Oberfläche des gescannten Objektes zu rekonstruieren. Diese Aussage spiegelt eine ideelle Aufgabenstellung wieder, welche unter Berücksichtigung realer Bedingungen nicht erreicht werden kann.

Im Folgenden wird zunächst die allgemeine Verarbeitung einer Rekonstruktion erläutert. Zudem wird auf einige Vorverarbeitungsmöglichkeiten einer Punktewolke hingewiesen.

3 Allgemeine Verarbeitungsschritte

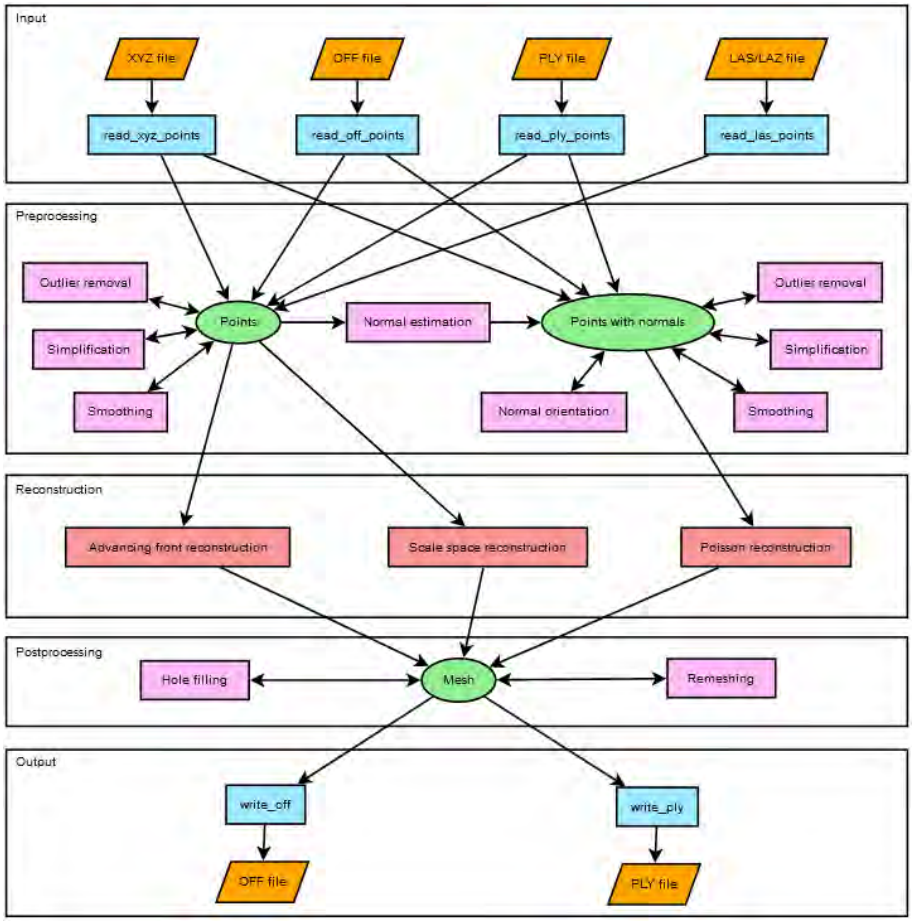

Abb. 3.1: Pipeline Overview (CGAL)[30]

Die Rekonstruktion einer Oberfläche durchläuft, unabhängig der Rekonstruktionsmethode, immer wieder die selben Schritte. In der Dokumentation der Computational Geometry Algorithms Library [30], kurz CGAL, wird beschrieben, wie die Verarbeitungsschritte klassifiziert werden können und einer Pipeline zugewiesen. Neben dem Input, Output und der eigentlichen Rekonstruktion, gibt es noch die Schritte der Vorverarbeitung und Nachbearbeitung. Welche Anwendungen in diesen beiden Abschnitten durchlaufen wird, hängt einerseits am erwarteten Ergebnis ab, andererseits an der gewählten Rekonstruktionsvariante. Somit sind manche Vorverarbeitungsschritte gegebenenfalls nicht optional, sondern zwingend erforderlich.

Im Folgenden wird kurz auf die einzelnen Verarbeitungsschritte eingegangen. Ein Überblick über die Pipeline wird dargelegt. Ebenfalls wird der Unterschied zwischen optionalen und notwendigen Schritten erklärt. Die einzelnen Methoden werden meist in Verbindung mit den bereitgestellten Funktionen von CGAL [30] erwähnt.

3.1 Input

Der erste Verarbeitungsschritt ist das Einlesen der Eingabewerte. Eine unorientierte Punktewolke im dreidimensionalen Raum besteht minimalistisch aus Punkten mit jeweils drei Koordinatenwerten x, y und z. Falls zusätzlich zu den Eingabewerten die jeweiligen Normalen pro Punkt angegeben sind, können die Werte nx, ny und nz bereits genutzt werden und auf eine spätere Vorverarbeitung zum Abschätzen der Normalenvektoren kann verzichtet werden. Eine weitere Art von Punktewolken sind Mengen an Punkten, die jeweils einen Intensitätswert (Iso-Wert) mitliefern. Diese sind ursprünglich aus medizinischen Anwendungen, wie MRT oder CT bekannt. Solche Daten können mithilfe eines Iso-Oberflächen-Algorithmus ausgewertet werden. Ein sehr bekannter ist hier der Marching Cubes Algorithmus [7]. Ein bereits bestehendes Oberflächennetz kann ebenfalls als Eingang zählen. In diesem Fall kann man das Netz erneut rekonstruieren, die Knotenpunkte als Punktewolke nutzen oder die bestehende Oberfläche neu abtasten.

Für die Speicherung von Punktewolken, wie auch Oberflächennetzen gibt es eine Vielzahl an Dateiformaten. Bekannte Formate sind XYZ, OFF, PLY und LAS [1]. Aus der bekannten Point Cloud Library (PCL) [27] ist auch noch das Format

PCD bekannt. Die einzelnen Formate unterscheiden sich im Layout und darin, welche zusätzlichen Informationen gespeichert werden können. Das LAS (Lidar Format) [32] kann standardmäßig keinen Normalenvektor speichern, wobei das XYZ Format dies zur Verfügung stellt [1]. Dateiformate, wie OFF [26] und PLY [24] können sowohl Punktewolken, wie auch entstehende Oberflächennetze speichern. Zusätzlich zu Intensitätswerten können noch optionale Eigenschaften, wie zum Beispiel Farbwerte gespeichert sein. Zur reinen Rekonstruktion der Oberfläche ist dies jedoch im Rahmen dieser Arbeit nicht notwendig.

Für die Pipeline ist in diesem Schritt wichtig, dass anhand des Eingangsformates die Daten richtig interpretiert und eingelesen werden.

Beispiel PLY Format

Als Beispiel wird kurz der Aufbau eines einfachen Ply Formates erläutert.

```
1  ply
2  format ascii 1.0
3  comment generated by ply_writer
4  element vertex 22998
5  property float x
6  property float y
7  property float z
8  element face 47794
9  property list uchar int vertex_indices
10 end_header
11 0.032643 0.0561475 -0.0499583
12 0.0325168 0.0580135 -0.050051
13 0.030805 0.0559782 -0.0499123
14 0.0312899 0.0574837 -0.0497482
15 0.0343362 0.0580692 -0.0499246
16 0.0346007 0.0599148 -0.0498238
17 0.0359961 0.0588523 -0.049794
```

Abb. 3.2: Ply Format Beispiel

In der ersten Zeile des Headers befindet sich der Name des Dateiformates. In diesem Fall *ply*. Die nächste Zeile definiert, ob die anschließenden Datensätze nach dem Header im *ascii* oder *binär* Format vorliegen. Für das Beispiel in Abbildung 3.2 wurde *ascii* gewählt, um folgende X,Y und Z Koordinaten der Knotenpunkte zu erkennen. Daraufhin kann mit dem Schlüsselwort comment eine beliebige Anzahl an Zeilen für mehrere Kommentare zum Datensatz geschrieben werden.

Nach den Kommentaren werden die Elemente des Oberflächennetzes definiert. Beginnend mit der Anzahl der gespeicherten Knotenpunkte. Darauffolgend werden die Eigenschaften der einzelnen Messwerte definiert. Haben diese weitere Werte, wie Intensität, Farbe oder Ähnlichem wird dies ebenfalls als Eigenschaft aufgelistet. Die somit angegebenen Eigenschaften definieren, wie viele Einträge ein Messwert pro Zeile aufweist. Die Reihenfolge der Daten pro Zeile gleicht der Folge der Definition der Eigenschaften. Für eine Punktewolke könnte man nun das Ende des Headers definieren und sogleich mit den einzelnen Datensätzen beginnen. Ein Oberflächennetz muss nun, wie auch im Beispiel erkennbar, noch seine Anzahl an Flächenelemente definieren, bevor die Knotenpunkte pro Zeile aufgezählt werden. Da ein Oberflächennetz aus Polygonen mit beliebiger Anzahl an Eckpunkten bestehen kann, wird für jede primitive Fläche zunächst die Anzahl der Eckpunkte definiert und im folgenden der Zeile die Indexe der zugehörigen Knotenpunkte. Im Anschluss des Headers kommen somit zeilenweise die Einträge der Knotenpunkte und folglich die Definitionen der Flächen anhand deren Eckpunktindexen.

3.2 Preprocessing

Vorverarbeitung ist generell dann notwendig, wenn die Rekonstruktionsalgorithmen spezielle Anforderungen an die eingehenden Punktwolken stellen [30]. Diesen Anspruch werden viele Inputdaten nicht gerecht. Andererseits kann das Bestreben an ein gutes Endresultat, die Vorverarbeitung der Punktewolke erzwingen, da diese zu einem gewissen Grad unvollkommen und fehlerhaft sein kann. Wie in Abbildung 3.1 zu sehen, sind die üblichsten Anwendungen der Vorverarbeitung folgende:

3.2.1 Average Spacing

Der Vorverarbeitungsschritt Average Spacing ist eine Analysemethode, die vor der Verarbeitung der Punktewolke durchgeführt wird. Diese Analyse versucht anhand einer definierten Anzahl an Nachbarn die durchschnittliche Entfernung der Eingangspunkte festzustellen. In der Dokumentation von CGAL zum Thema Point Set Processing [1] wird die Funktion "compute_average_spacing()"beschrieben. Diese berechnet den durchschnittlichen Abstand aller Eingabepunkte zu ihren k nächsten Nachbarpunkten, wobei k vom Benutzer angegeben werden kann [1]. In der Dokumentation wird ebenfalls darauf hingewiesen, dass dieses Verfahren

nützlich sein kann für die spätere Rekonstruktion einer Oberfläche mithilfe der Punktewolke.
Dieser Prozess ist somit optional. Der Vorteil jedoch ist die automatische Abschätzung einiger Parameter für die folgende Rekonstruktion. In meiner Implementierung des Poisson Rekonstruktionsverfahrens [17] mithilfe von CGAL [30] verwende ich diese Funktion um den Abstand der Punkte nicht manuell wählen zu müssen.

3.2.2 Outlier Removal

Nachdem bekannterweise echte Messwerte zumeist nicht vollständig korrekt sind, ist es üblich abweichende Messpunkte und verrauschte Daten vor der Verwendung anzupassen. In diesem Schritt der Vorverarbeitung werden speziell die Ausreißer der Punktewolke entfernt. Diese würden bei einigen Rekonstruktionen zu fehlerhaften Oberflächen führen.
Die Bibliothek CGAL [30] bietet hier ebenfalls eine Funktion "remove_outliers()", die mithilfe einiger Parameter die Eingabewerte ausdünnt. Die Funktion wird unter [1] beschrieben und kann mit einem Grenzwert für das maximale Entfernen von Punkten belegt werden. Die Ausdünnung wird hierbei entweder mithilfe einer definierten Anzahl an Nachbarn berechnet oder mit einem festen Radius. Die Punkte mit den höchsten Werten, anhand ihrer quadrierten Entfernung zu den Nachbarn, wird entfernt.
Dieser Verarbeitungsschritt wird bei stark fehlerhaften Daten empfohlen. Insgesamt dient er jedoch nur zur frühzeitigen Ergebnisoptimierung und ist prinzipiell optional. Einige Rekonstruktionsalgorithmen können sogar mit einem gewissen Grad an Ausreißern umgehen und sind somit sehr stabil.

3.2.3 Simplification

Wie der Name Simplification schon vermuten lässt, wird in diesem Schritt die Punktewolke vereinfacht. Dies ist hinsichtlich verschiedener Eigenschaften durchführbar. In der Dokumentation der Punktewolkenverarbeitung [1] werden 4 verschiedene Ansätze beschrieben. Die zwei simpleren Verfahren vereinfachen die Punktwolke einerseits zu einem Benutzer gewählten Grad zufällig und andererseits nach einem definierten Gitter, in dessen Zellen die Punktewolke auf jeweils einen repräsentativen Punkt vereinfacht wird. Die beiden Funktionen sind als

random_simplify_point_set() und grid_simplify_point_set() deklariert. Das zufällige Vereinfachen ist hinsichtlich Performance die schnellere Variante.[1] Eine weitere Methode ist gegeben als hierarchy_simplify_point_set(), welche im Großteil auf dem Verfahren beschrieben in [25] beruht. Diese adaptive Vereinfachung nutzt lokales Clustering, wobei die Größe der Cluster vom Nutzer spezifiziert werden kann oder anhand der lokalen Variation der Punktewolke angepasst wird. Schließlich wird die Funktion wlop_simplify_and_regularize_point_set() bereitgestellt. Dies ist eine Implementierung des Weighted Locally Optimal Projection (WLOP)-Algorithmus [16], der die Punktewolke nicht nur vereinfacht, sondern auch reguliert [1].

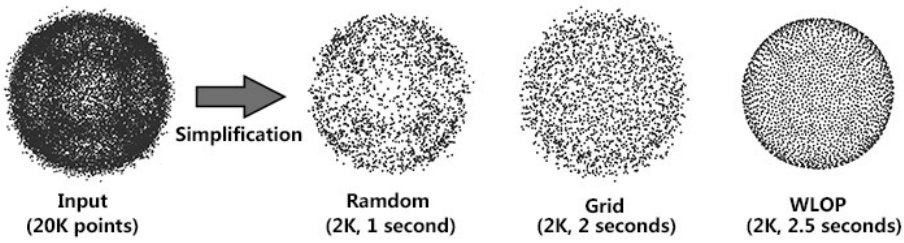

Abb. 3.3: Vereinfachungsmethoden: Zufällige Vereinfachung, Gittervereinfachung, WLOP-Vereinfachung [1]

Die Vereinfachung der Eingangswerte ist grundsätzlich ebenfalls optional. Wie in Abbildung 3.3 veranschaulicht, sieht man das die verschiedenen Methoden ein jeweils unterschiedliches Ergebnis liefern. Die Verarbeitungszeit ist ebenfalls unterschiedlich. Der Schritt der Vereinfachung wird generell genutzt um die Menge des Inputs zu reduzieren und somit die folgende Rekonstruktion zu beschleunigen. Der Punktabstand im Resultat der Vereinfachung kann speziell bei expliziten Rekonstruktionen eine Rolle spielen. Diese benutzen oft alle Punkte der Punktewolke als Knotenpunkte der Oberfläche. Falls dieser optionale Schritt durchgeführt wird, kann somit eine kleinere Menge an Knotenpunkten des resultierenden Oberflächennetzes erreicht werden.

3.2.4 Smoothing

Das Glätten ist ein Verfahren, welches allgemein speziell aus der Messtechnik und Bildverarbeitung bekannt ist. Einige Rekonstruktionen setzen bereits in ihrer Idee der Oberflächen-Wiederherstellung einen glättenden Filter um. Ein Beispiel

hierfür wäre die Poisson Rekonstruktion [17]. Die glättende Wirkung kann für die Nachbildung der Oberfläche positiv, wie auch negativ sein. Der Hauptgrund absichtlich einen glättenden Filter umzusetzen ist verrauschte Eingangsdaten zu optimieren. Zu Beachten ist, das dadurch hochfrequente Informationen verloren gehen können. Hinsichtlich Oberflächenrekonstruktion ist hierbei auf echte Kanten und raue Oberflächen zu achten, da diese gegebenenfalls im Ergebnis verloren gehen können. Die glättende Wirkung auf die erstellte Oberfläche ist im Kapitel Anwendungen & Ergebnisse zu betrachten.

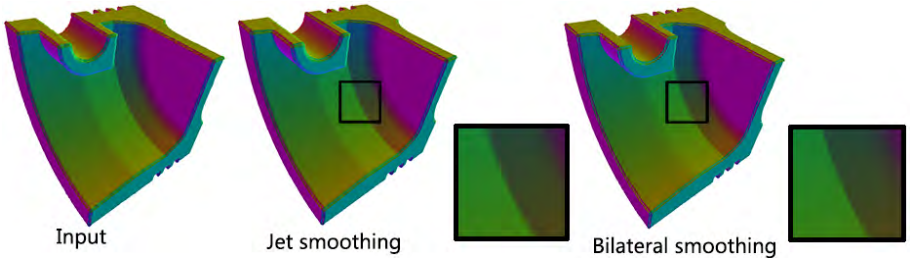

Abb. 3.4: Glättungsmethoden: Input 250k, Jet Smoothing, Bilateral Smoothing [1]

Der Vorverarbeitungsschritt der Glättung ist optional und wird bei einem gewissen Vorwissen über den verrauschten Input empfohlen. Dieser Schritt kann auch zusätzlich zu einer glättenden Rekonstruktion verwendet werden, um die Glättung zu verstärken bzw. manuell zu kontrollieren. Die Beschreibung der glättenden Funktionen jet_smooth_point_set() und bilateral_smooth_point_set() kann unter [1] gefunden werden. Für nachfolgende Experimente wurde teilweise die erste Methode verwendet. In Abbildung 3.4 können die jeweiligen Resultate der Vorverarbeitung eines Beispiels betrachtet werden.

3.2.5 Normal Estimation

Das Abschätzen der Normale für die Punkte der Punktewolke kann potentiell notwendig sein. Dies kommt auf den verwendeten Rekonstruktionsansatz an. Speziell implizite Verfahren, wie die Hoppe Methode [15] oder auch die Poisson Rekonstruktion [17] verwenden den Normalenvektor der Punkte um ihre Distanzfunktion bzw. Indikatorfunktion aufzustellen. Explizite Verfahren nutzen die Normalen der Punktwolke meistens nicht und berechnen folglich nur die Normalenvektoren der entstehenden Flächen und Knotenpunkte der Oberfläche.

Für das Abschätzen stellt die CGAL Bibliothek [1] drei verschiedene Funktionen zur Verfügung. Im Ansatz von Hoppe [15], wie auch ähnlichen Methoden, wird eine Regressionsebene einer definierten lokalen Nachbarschaft verwendet um den Normalenvektor für einen Messwert abzuschätzen.
Falls eine unorientierte Punktewolke als Input gegeben ist, ist dieser Schritt und der folgende dringend notwendig um eine orientierte Punktewolke zu erhalten.

3.2.6 Normal Orientation

Nachdem man die Normalenvektoren abgeschätzt hat, muss ein Schritt der Orientierung durchgeführt werden. In der CGAL Bibliothek [1] wird hier die Funktion mst_orient_normals() vorgestellt. Diese beruht hauptsächlich auf dem Algorithmus von Hueges Hoppe et. al [15], der insgesamt eine implizite Rekonstruktion anstrebt. In meiner Implementierung der Poisson Rekonstruktion [17] mithilfe der CGAL Bibliothek [30] verwende ich die Funktionen pca_estimate_normals() zum Abschätzen der Normalenvektoren und anschließend mst_orient_normals() zur Orientierung. Die unorientierbaren Punkte werden im Anschluss vor der Rekonstruktion entfernt um Fehler zu vermeiden.

Die einheitliche Orientierung der Messwerte ist aus zwei Gründen notwendig. Einerseits werden unorientierbare Punkte aus der Rekonstruktion ausgeschlossen. Andererseits kann ein Vektor, welcher eine Ebene, in diesem Fall die Regressionsebene beschreibt in zwei entgegengesetzte Richtungen zeigen. Im Schritt der einheitlichen Orientierung wird versucht die Normalenvektoren entweder gesamt ins innere des Objektes zeigen zu lassen oder alle Weg von der Oberfläche, nach außen des Objektes.

Wie bereits erwähnt kann ein solcher Schritt zwingend erforderlich sein, falls für die Rekonstruktion eine orientierte Punktewolke vorausgesetzt wird.

3.3 Rekonstruktion

Bei der eigentlichen Rekonstruktion kann aus einer Vielzahl bestehender Algorithmen gewählt werden. Manche erzwingen bestimmte Eigenschaften der Punktewolke, weshalb die Vorverarbeitung eine Rolle spielen kann. Viele der Algorithmen besitzen die selbe Grundidee, haben jedoch eine unterschiedliche Vorgehensweise.

Um nicht zu allgemein nur zwischen impliziter und expliziter Rekonstruktion unterscheiden zu müssen, wird im Kapitel Rekonstruktionsklassen versucht die Verfahren einzuteilen. Die Klassen repräsentieren jeweils eine bestimmte Grundidee. Da in den nächsten Kapiteln die einzelnen Rekonstruktionsalgorithmen vorgestellt werden, wird hier auf eine weitere Ausformulierung verzichtet. Die Resultate der verschiedenen Methoden sind im Kapitel Anwendungen & Ergebnisse zu betrachten.

3.4 Postprocessing

Eine Nachverarbeitung des Oberflächennetzes ist dann eine Option, falls an das Ergebnis gewisse Anforderungen gestellt werden. Mithilfe einer expliziten Rekonstruktion kann es passieren, dass Löcher entstehen. Diese fehlenden Bereiche im Netz sind für geschlossene Objekte nicht wünschenswert. Neben der Möglichkeit ein Remeshing durchzuführen, kann es auch interessant sein bestimmte Eigenschaften, wie zum Beispiel die Umkreisbedingung einer Delaunay - Triangulierung zu erhalten. Solche Ergebnisse werden mithilfe iterativer Maßnahmen erreicht. Ebenfalls interessant können Verfahren zum Ausdünnen der Menge an primitiven Flächen des Oberflächennetzes sein. Diese versuchen zu lokalisieren, ob benachbarte Flächen in einer Ebene liegen. Falls dies zutrifft, kann die Triangulierung relativ verlustfrei ausgedünnt werden. Die verschiedenen Möglichkeiten zur Nachverarbeitung werden im Rahmen dieser Arbeit nicht genauer erläutert.

3.5 Output

Das Ergebnis der Pipeline ist das gewünschte Oberflächennetz. Dies kann, wie bereits beschrieben, in verschiedenen Formaten gespeichert werden. In meinem Fall verwendete ich das OFF [26] und das PLY [24] Dateiformat. Die meisten Rekonstruktionen, die im Folgenden vorgestellt werden, liefern als Ergebnis eine Triangulierung. Alle simplen Flächen dieses Netzes bestehen aus Dreiecken. Um auch eine andere Repräsentation der Oberfläche zu zeigen, wird der Power Crust Algorithmus [23] in seiner ursprünglichen Implementierung verwendet, welcher ein Polygonnetz liefert. Die Flächen eines Polygonnetzes basieren auf Polygonen mit unterschiedlicher Anzahl an Eckpunkten und somit auch Kanten. Je nach Wunsch kann ein Polygonnetz durch Einfügen weiterer Kanten zu einer Triangu-

lierung werden. Die unterschiedlichen Oberflächenresultate der Rekonstruktionen können im Kapitel Anwendungen & Ergebnisse betrachtet werden.

4 Rekonstruktionsklassen

In diesem Kapitel wird eine Übersicht über verschiedene Rekonstruktionsklassen vorgestellt. Insgesamt gibt es eine Vielzahl an Algorithmen, die versuchen eine Oberfläche aus Punktewolken zu generieren. Die einzelnen Algorithmen basieren oft auf einer ähnlichen Grundidee. Hier werden speziell einige dieser Strategien fokussiert und in folgenden Kapiteln werden dann einzelne Algorithmen stellvertretend für ihre Klasse noch einmal genauer erläutert. Im Kapitel Anwendungen & Ergebnisse werden die Resultate dieser Verfahren anhand eigener Versuche verglichen.

Die Iso-Oberflächen Auswertung wird als Erstes vorgestellt, da sie im Folgenden für die impliziten Rekonstruktionen verwendet wird. Das Konzept der Delaunay Triangulierung wird speziell aufgrund ihrer nützlichen Eigenschaften genauer erläutert. Aus Gründen einer besseren Veranschaulichung werden hierfür zweidimensionale Beispiele gewählt. Die Dualität zum Voronoi-Diagramm wird ebenfalls aufgrund dessen Eigenschaften erläutert. Alle folgenden Rekonstruktionen verwenden einen Teil der Konzepte einer Delaunay-Triangulierung, eines Voronoi-Diagramms, wie auch einer Iso-Oberflächen Auswertung. Die meisten Rekonstruktionen werden entweder als explizit oder implizit bezeichnet. Im Anschluss werden hierfür zwei explizite und eine implizite Variante vorgestellt. Eine besondere Rekonstruktion, welche die mediale Achsen-Transformation (MAT) nutzt, wird zuletzt vorgestellt. Im Bereich Rekonstruktion gibt es noch weitere spezielle Verfahren, welche diese Strategien nutzen oder mit gewisser Ähnlichkeit versuchen eine Oberfläche zu rekonstruieren. Im Rahmen dieser Arbeit werden die Ausformulierungen auf folgende Strategien begrenzt.

4.1 Iso-Oberflächen Auswertung

Bereits 1987 wurde von William E. Lorensen und Harvey E. Cline ein neuer Algorithmus, genannt "Marching Cubes"[7], zum Auswerten von medizinischen

Dichtedaten vorgestellt. Diese Daten stammten von CT, MRT und SPECT Anwendungen [7]. Das Ziel war die Auswertung und Konstruktion einer Oberfläche um die dreidimensionalen Ergebnisse zu visualisieren.

Ein weiterer Algorithmus zur Iso-Oberflächen Auswertung ist der Marching Tetrahedra. Er wurde entwickelt, da er "das Patent, das den Marching Cubes-Algorithmus abdeckte, das inzwischen abgelaufen ist"[9], vermeidete. Im folgenden Verlauf wurden diese Algorithmen stets angepasst und leicht abgeändert. Viele Varianten finden ihren Einsatz bei impliziten Rekonstruktionen, wie der Poisson Oberflächen Rekonstruktion [17]. Das Ziel dieser Marching Cubes ähnlichen Methoden ist die Auswertung der Iso-Oberfläche anhand einer vorher abgeschätzten Funktion dieser. Genaueres hierzu im Abschnitt zur impliziten Rekonstruktion.

Die Grundidee einer Iso-Oberflächen Auswertung ist die Eingangsdaten in ein Voxelgitter einzuteilen und mithilfe eines Würfels bzw. eines Tetrahedras über diese Struktur zu iterieren. Die Eckpunkte werden anhand einer trilinearen Interpolation ausgewertet und liefern entweder einen Wert 1 oder 0. Mithilfe einer Tabelle kann nachgeschlagen werden, welche Flächen, Kanten und Knotenpunkte der zu erstellenden Oberfläche in diesem Bereich hinzugefügt werden.

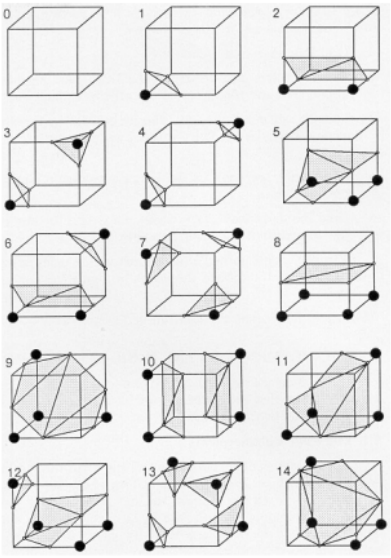

Abb. 4.1: Triangulierte Würfel [7]

Im Falle des Marching Cubes gibt es 2^8, also 256 verschiedene Indexvariationen. Diese bestehen aus 15 unterschiedlichen Würfeln, die jeweils gedreht und gespiegelt werden können. In der Abbildung 4.1 kann man diese Muster sehen. Der Index selbst kann in drei Klassen eingeteilt werden [7] :

- index = 0: Der Würfel ist komplett außerhalb der Oberfläche.
- 0 < index < 255: Der Würfel befindet sich auf der Oberfläche.
- index = 255: Der Würfel ist komplett innerhalb der Oberfläche.

Für jeden Iterationsschritt braucht es nun noch einige Feinheiten. Die Flächen und Knotenpunkte werden nicht immer anhand der Größe der betrachteten Form(z.B. Würfel) hinzugefügt. Wie in [7] beschrieben, werden die Positionen der Knotenpunkte auf den betrachteten Kanten der Form linear interpoliert. Dies geschieht mithilfe der Intensitätswerte der Eckpunkte. In einem folgenden Schritt müssen schließlich noch die Normalenvektoren der Knotenpunkte und Flächen abgeschätzt werden. Somit ist die Iso-Oberflächen Auswertung komplett. Die Menge an Knotenpunkten, Kanten und Flächen ergeben insgesamt die gewünschte Repräsentation der Oberfläche.
Das Resultat des Marching Tetrahedras ist im Gegensatz zum Marching Cubes Algorithmus ein Polygonnetz und keine Triangulierung. In der Abbildung 4.2 sieht man die primitiven Flächen der Indexauswertung, die nicht immer ein Dreieck sind.
Stellvertretend für diese Klasse wird im Anschluss der Marching Cubes Algorithmus noch einmal Schritt für Schritt erklärt.

4.2 Delaunay-Triangulierungen

"Delaunay-Triangulation[...] ist ein gebräuchliches Verfahren, um aus einer Punktemenge ein Dreiecksnetz zu erstellen." [10] Die Punktewolken können hierbei auf einer Ebene liegen, also im Zwei-Dimensionalen, oder im Drei-Dimensionalen Raum. Zur Anwendung kommen Delaunay-Triangulierungen einerseits in der Erstellung von Oberflächennetzen, wie auch andererseits in der Optimierung von bestehenden Triangulierungen. Der Grund dafür ist, dass Delaunay Triangulierungen eine bestimmte Bedingung erfüllen. Dieses Prinzip wird in [10] und [18] erläutert und beschreibt die sogenannte Umkreisbedingung wie folgt.

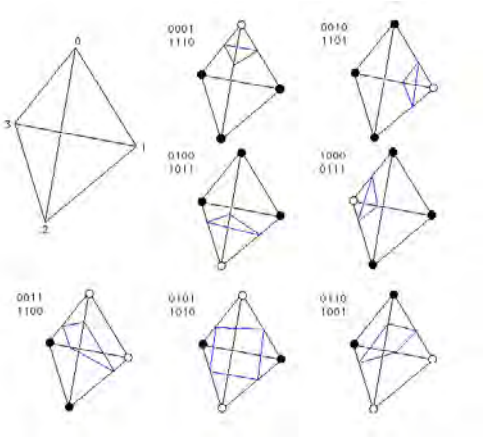

Abb. 4.2: Tetrahedra Möglichkeiten [5]

Umkreisbedingung

Jeder Umkreis eines Dreieckes der bestehenden Triangulierung darf keinen anderen Knotenpunkt des Oberflächennetzes enthalten. Diese Eigenschaft führt lokal gesehen zu Dreiecken mit möglichst großem Innenwinkel und global betrachtet zu einer Maximierung des kleinsten Innenwinkels. [10] [18] Im Dreidimensionalen wird oft auch von einer analogen Umkugelbedingung gesprochen. [10]

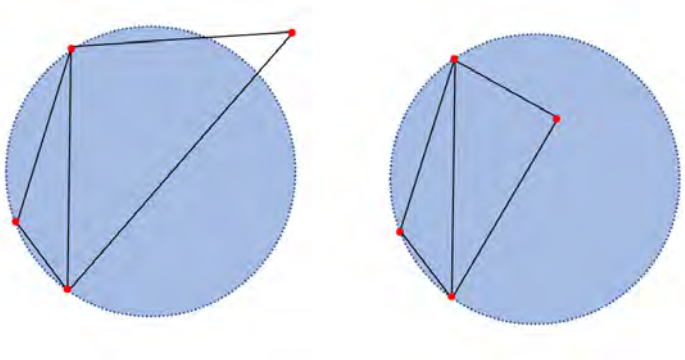

Abb. 4.3: Umkreisbedingung - links: erfüllt, rechts: verletzt

Diese Bedingung erklärt, warum eine Delaunay Triangulierung einerseits zur

Rekonstruktion eines Oberflächennetzes verwendet wird, wie auch andererseits als Optimierung eines bestehenden Netzes. Die Bedingung kann bereits bei verschiedenen Algorithmen in der Entstehung beachtet werden oder die Bedingung wird durch iterative Verfahren auf ein Netz angewandt, welches die Umkreisbedingung zunächst nicht erfüllt. Die Umkreisbedingung muss nicht zwangsläufig eindeutig sein, da sich mehrere Messpunkte auf einem gemeinsamen Umkreis befinden können. In diesem Fall kann sich der Anwender beliebig drei Punkte für das nächste Dreieck aussuchen. [10]

In [10] wird beschrieben das Netze, welche diese Bedingung erfüllen, speziell in der Computergrafik erwünscht sind, da sie Rundungsfehler minimieren. Als weiteres Anwendungsfeld wird die Optimierung von Berechnungsnetzen für Finite-Elemente-Methode genannt. Ein gutes Beispiel, warum eine solche Bedingung wichtig sein kann, präsentiert Prof. Dr. Philipp Kindermann in seinem Kurs über algorithmische Geometrie [18]. Die Aufgabenstellung im Beispiel beruht auf einer Interpolation von Höhenprofilen. In Abbildung 4.4 wird der Sachzusammenhang veranschaulicht. Die Grafik zeigt eine Projektion von Messpunkten einer Erdoberflächen-Abtastung. Die einzelnen Zahlen geben den Höhenwert zum zugehörigen Messpunkt auf der Ebene wieder. Für eine feinere Auflösung sollten nun weitere Knotenpunkte zwischen den bestehenden Punkten interpoliert werden. Die Fragestellung ist nun, welches Oberflächennetz wahrscheinlicher den wahren Wert der Höhe widerspiegelt.

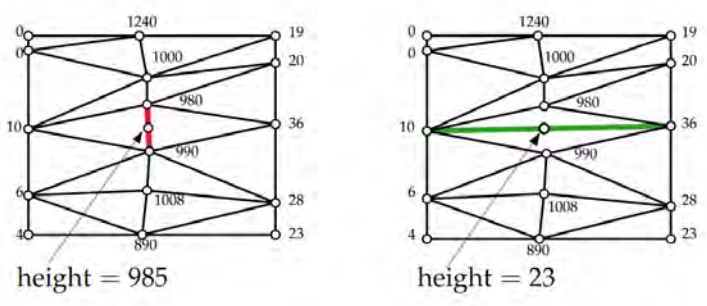

Abb. 4.4: Höhenprofil - Interpolation [18]

Mit dem Wissen, dass es der Scann einer Bergkette ist, würde man die linke Variante bevorzugen. Diese Entscheidung würde aufgrund einer Delaunay - Triangulierung ebenfalls gewählt werden. In [18] wird zusätzlich die Aussage getroffen,

schmale Dreiecke zu vermeiden oder anders gesagt Dreiecke mit größeren Innenwinkeln zu priorisieren. Dies passt zur Umkreisbedingung, die global den kleinsten Innenwinkel maximiert.

Für die Erstellung einer Delaunay - Triangulierung gibt es mehrere verschiedene Methoden. Im Folgenden werden einige zweidimensionale Ansätze vorgestellt und der Zusammenhang mit einem Voronoi Diagramm erklärt.

Flip - Algorithmus

Das Edge Flipping kann entweder auf eine bestehende Zweidimensionale Triangulierung angewandt werden oder es muss zunächst ein Zweidimensionales Oberflächennetz erzeugt werden. Dieses muss jedoch die Bedingung erfüllen, dass es keine schneidenden Kanten besitzt.[10] Die Idee besteht darin für alle benachbarten Dreiecke die Umkreisbedingung zu erfüllen. Edelsbrunner [13] erläutert, dass eine Triangulierung als Delaunay Triangulierung zählt, sobald sie an jeder lokalen Stelle die Bedingung erfüllt.
Aufgrund der Tatsache, dass als Ausgangspunkt ein Oberflächennetz aus Dreiecken besteht, haben benachbarte Flächen jeweils eine Kante gemeinsam. Eine Kante kann hierbei entweder zu zwei Dreiecken gehören oder das Oberflächennetz abschließen und somit nur zu einem Dreieck gehören. Interessant sind somit alle Kanten, die zu zwei Flächen gehören. Diese werden vom Algorithmus abgearbeitet und es wird pro Kante getestet, ob die beiden zugehörigen Dreiecke die Umkreisbedingung erfüllen. Falls nicht muss gehandelt werden.

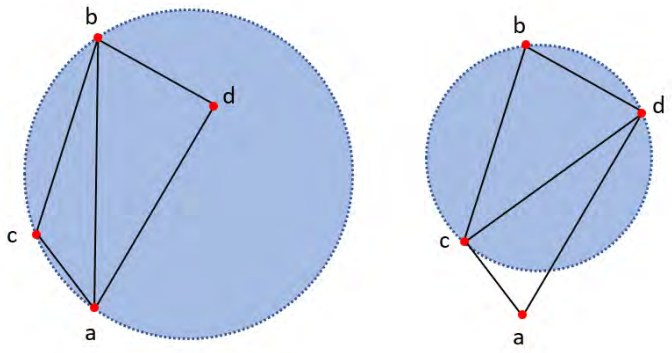

Abb. 4.5: Edge Flipping

Wie in Abbildung 4.5 zu sehen, wird die ungültige Kante “geflippt”, das heißt sie wird entfernt und die jeweils andere Diagonale des entstehenden Vierecks wird eingefügt. Im Beispiel würden somit der Triangulierung die Dreiecke abc und adb, sowie die Kante ab entfernt werden. Zusätzlich werden jedoch die Dreiecke adc und bcd, sowie die Kante cd hinzugefügt. Der Algorithmus hat insgesamt einen Rechenaufwand von $\mathcal{O}(n^2)$. [10] Nachdem also der Algorithmus lokal alle Kanten überprüft hat und gegeben falls die benachbarten Kanten angepasst hat, ist die zweidimensionale Triangulierung Delaunay konform.

Inkrementelle Konstruktion

Die inkrementelle Konstruktion ist eine Verallgemeinerung des Flip - Algorithmus für höhere Dimensionen.[10] Wie der Name schon vermuten lässt, wird hierbei der neuen Triangulierung jeweils ein neues Dreieck schrittweise hinzugefügt. Für jedes neue Dreieck wird geprüft, ob in dessen Umfeld die Umkreisbedingung bzw. Umkugelbedingung erfüllt bleibt. Falls nicht werden in der bestehenden Struktur die betroffenen Kanten und Dreiecke angepasst.
Der Vorteil dieser simplen Methode ist, dass die Messpunkte nicht unbedingt zu Beginn des Algorithmus bekannt sein müssen. In der Praxis wird jedoch bei vielen dieser Varianten eine Starttriangulierung (in 2D auch - Super Triangle) außerhalb aller erwarteten Eingangspunkte gewählt.[29] Diese wird nach Beendigung der bestehenden Triangulierung wieder entfernt.
Die Konstruktion beruht grundsätzlich auf zwei Phasen. Einer Such-Phase, in welcher das bestehende Dreieck, in welchem der neue Punkt ist, aufgespürt wird und einer Update-Phase, welche das neue Dreieck erzeugt und alle nötigen Flips durchführt. Mit angepasster Datenstruktur für die Suche kann eine Laufzeit von $\mathcal{O}(n\ log\ n)$ erreicht werden.[29]

Divide and Conquer - Ansatz

Es liegt nahe einen bekannten Ansatz in der Informatik zu nutzen, nämlich einen Teile-und-Herrsche Ansatz. Die Punktewolke oder bestehende Triangulierung wird in bestimmte Bereiche eingeteilt und einzeln durch inkrementelle Konstruktion zu einer Delaunay Triangulierung umgewandelt. Im Anschluss werden die Schnittstellen der angrenzenden Bereiche verknüpft. In [10] und [29] werden auf Ansätze hingewiesen, welche eine Laufzeit von $\mathcal{O}(n\ log\ n)$ aufweisen. Ebenfalls wird in [29] erwähnt, dass Dwyer [12] durch eine einfache Modifikation der

Algorithmen-Ansätze eine Laufzeit von $\mathcal{O}(n\ log\ log\ n)$ erzielt.

Sweepline - Algorithmus

In [10] und [29] wird ebenfalls der Sweepline - Algorithmus vorgestellt. Der Unterschied zum inkrementellen Vorgehen ist, dass bei diesem nicht jeweils um einen zufälligen Punkt bzw. Dreieck erweitert wird. Der Sweepline - Ansatz arbeitet indem er Punkte aus einer angrenzenden Region als nächsten Schritt wählt. Hierzu wird in [29] erläutert, dass dafür ein bestimmtes Grenzgebiet gespeichert wird und für die übrigen Punkte eine Warteschlange mit Priorisierung erstellt wird. Der Sweepline - Algorithmus hat ebenfalls eine Laufzeit von $\mathcal{O}(n\ log\ n)$. [10][29]

Dualität zum Voronoi Diagramm

Schließlich wird noch der Zusammenhang zwischen einem Voronoi Diagramm und einer Delaunay Triangulierung geklärt. Anhand deren Zusammenhang gibt es viele verschiedene Ansätze ein Voronoi Diagramm zu erhalten und somit auch eine Delaunay Triangulierung. Die wichtigste Eigenschaft eines Voronoi Diagrams ist die Dualität zur Delaunay Triangulierung.

Zunächst werden kurz die Eigenschaften eines Voronoi Diagramms geschildert.

Das Voronoi Diagram besteht aus Regionen, Kanten, Knotenpunkten und Zentren. Die Zentren sind eine vorgegebene Menge an Punkten. Diese wären in unserem Falle die gegebenen Messpunkte. Ziel ist es nun die Voronoi Regionen einzuteilen. Eine Region hat die Eigenschaft, dass jeder Punkt in ihr den kürzesten Abstand zum eigenen Zentrum aufweist, im Gegensatz zu den anderen Zentren. Prof. Dr. Kindermann erläutert in seinem Kurs [19] den Zusammenhang zum bekannten Post-Office Problem, indem genau diese Umstände versucht werden zu erhalten. Die Kanten unterteilen den Raum in ihre einzelnen Regionen. Diese erhält man, indem man für alle benachbarten Zentren eine Mittelsenkrechte oder Mittellotebene berechnet. Alle Punkte auf einer Kante weisen zu den Zentren benachbarter Regionen den selben Abstand auf. Die Knotenpunkte ergeben sich beim Schneiden der Kanten.
Prof. Dr. Kindermann stellt hierzu in seinem Kurs [19] eine formelle Ausformulierung dieser Eigenschaften für ein zweidimensionales Beispiel dar. Das Voronoi

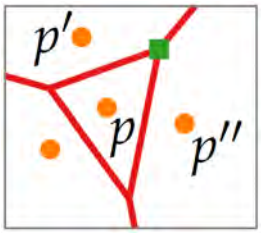

Abb. 4.6: Voronoi Diagramm Beispiel [19]

Diagramm ist die Menge aller Regionen. Für eine Punktewolke P werden benachbarte Punkte $p, p', p'' \in$ P betrachtet. Diese sind in der Abbildung 4.6 zu sehen. Eine Voronoi Zelle/Region wird definiert als :

$$V(\{p\}) = V(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \; \forall \; q \in P \setminus \{p\}\} \tag{4.1}$$

Eine Kante ist definiert als:

$$V(\{p, p'\}) = \{x : \; |xp| = |xp'| \; and \; |xp| < |xq| \; \forall q \neq p, p'\} \tag{4.2}$$

Und schließlich werden die Schnittstellen der Kanten, also die Knotenpunkte definiert als:

$$V(\{p, p'.p''\}) = \{x : \; |xp| = |xp'| = |xp''| \; and \; |xp| \leq |xq| \;\; \forall q\} \tag{4.3}$$

In [18] wird bewiesen, dass eine Dualität zwischen einem Voronoi Diagramm und einem Delaunay Graphen besteht. Konstruiert wird dieser Graph indem Zentren benachbarter Regionen mit einer neuen Kante verbunden werden. Der entstehende Graph ist noch nicht zwangsweise eine Triangulierung, da die Flächen nicht unbedingt Dreiecke sind. Durch einen weiteren Schritt, der höhere Polygone in Dreiecke einteilt, kann eine Delaunay Triangulierung erhalten werden. Dieser Zusammenhang ist in Abbildung 4.7 zu erkennen. Die blauen Kanten gehören zum Voronoi Diagramm und die schwarzen Kanten sind Teil des entstehenden Oberflächennetzes.
Ein simpler Ansatz zur Konstruktion eines Voronoi Diagramms ist ein inkrementelles Vorgehen, welches jeweils einen neuen Punkt dem bestehenden Diagramm hinzufügt. Die umliegenden Regionen müssen somit angepasst werden und die Struktur der Kanten und Zellen verändert sich. Ein weiteres bekanntes Vorgehen is ein Sweep-Algorithmus mit einer "Beachline", welches in [19] erläutert wird.

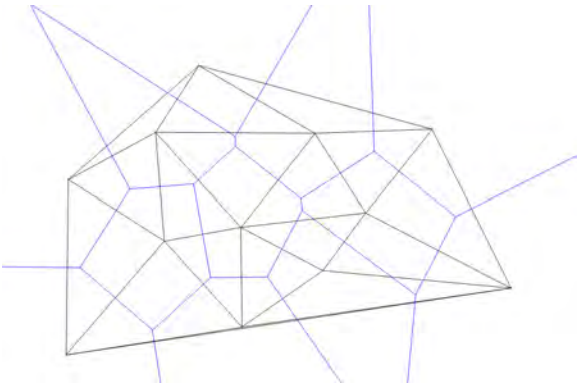

Abb. 4.7: Voronoi Diagramm und Delaunay Graph

Für die Erstellung eines Oberflächennetzes ist in diesem Zusammenhang am wichtigsten die Dualität zwischen Voronoi Diagramm und Delaunay Triangulierung. Die Dualität wird für Rekonstruktionen im Zweidimensionalen, wie auch im Dreidimensionalen ausgenutzt. Anwendungen dieses Zusammenhangs werden in den später vorgestellten Strategien zur Rekonstruktion einer Oberfläche klar.

4.3 Implizite Rekonstruktion

Implizite Rekonstruktionen basieren grob auf zwei Grundschritten. Zunächst wird im ersten Basisschritt eine implizite Funktion gesucht, welche die Oberfläche des gesuchten Objektes repräsentieren sollte. Im anschließenden Schritt wird die Oberfläche der erzeugten Funktion ausgewertet. Dies geschieht mit einem angepassten Iso-Oberflächen Auswertalgorithmus, welche großteils auf dem Marching Cubes Verfahren beruhen.

Das Aufstellen der impliziten Funktion wird von den verschiedenen Methoden jeweils etwas unterschiedlich angegangen. Grundsätzlich kann man jedoch von einer Distanzfunktion sprechen, welche den inneren Raum, die Oberfläche und den äußeren Raum definiert. Speziell der Bereich der Oberfläche ist hier interessant, wodurch in vielen wissenschaftlichen Arbeiten von einer Indikatorfunktion gesprochen wird, welche nur noch 2 Zustände kennt. Zum Auswerten der Iso-Oberfläche reicht die Unterscheidung, ob ein Bereich zur Oberfläche gehört oder

nicht.

Einer der neueren Ansätze von 2006, die eine Indikatorfunktion anstreben, ist die Poisson Surface Rekonstruktion. Diese wird ebenfalls stellvertretend für diese Klasse im Anschluss vorgestellt. Sie strebt eine globale Lösung an und verbindet Ansätze vorheriger Verfahren. [17] Der beteiligte Autor Hueges Hoppe hat hierzu bereits 1992 ein früheres Verfahren [15] vorgestellt, welches die Distanzfunktion lokal durch eine definierte Nachbarschaft abschätzt. Die Technik beschrieben in [15] wurde bereits im Zusammenhang mit der Normalen Abschätzung und Orientierung erwähnt. Solche Vorverarbeitungsschritte basieren oft auf dem selben lokalen Ansatz.

Die Schwierigkeit dabei ist im dreidimensionalen Raum die Nachbarschaft eines Punktes zu definieren. Triviale Verfahren sind die Definition über einen bestimmten Abstandsradius oder einer bestimmten Menge an am nächsten liegenden Punkte, die dann als Nachbarn gewertet werden. Eine andere Methode verfolgen Jean-Daniel Boissonat und Frédéric Cazals in ihrer Veröffentlichung [4] einer impliziten Rekonstruktion, indem sie die Nachbarschaft eines Punktes über die angrenzenden Voronoi Regionen definieren. Anders gesagt zählen, aufgrund der Dualität zur Delaunay Triangulierung, diese Knotenpunkte als Nachbarn, welche eine gemeinsame Kante im Delaunay Graphen mit dem betrachteten Punkt aufweisen würden.

Weitere Verfahren streben eher eine globale Lösung an und beziehen sich nicht lokal gesehen auf eine bestimmte Nachbarschaft. Zum Beispiel werden [22], [6] und [31] in [17] erwähnt als implizite Rekonstruktionen. Diese streben eine globale Lösung an, welche die implizite Funktion durch Aufsummieren von Radialen Basisfunktionen (RBFs) erreichen will. Die Poisson Rekonstruktion versucht die lokalen und globalen Ansätze zu verbinden. [17] Einzelheiten hierzu im Kapitel Poisson Rekonstruktion.

Aufgrund der Auswertung der Iso-Oberfläche im zweiten Abschnitt der Rekonstruktion haben die Knotenpunkte des entstandenen Oberflächennetzes nicht mehr direkt Bezug zu den eigentlichen Messwerten. Je nachdem in welcher voxelbasierten Größe die Auswertung stattfindet, wird das Ergebnis geglättet. Dadurch weisen implizite Verfahren immer einen gewissen Grad an Glättung der Oberfläche auf. Dies kann positive, wie auch negative Auswirkungen haben. Hierzu mehr im

Kapitel Anwendungen & Ergebnisse.

4.4 Advancing Front Algorithmen

Advancing Front Algorithmen versuchen, im Gegensatz zu impliziten Methoden, möglichst alle Messwerte als Knotenpunkte für das entstehende Oberflächennetz zu verwenden. Wie der Name schon sagt verfolgen diese Rekonstruktionen die Idee eine gewisse voranschreitende Menge an Dreiecken hinsichtlich der gegebenen Messpunkte zu erweitern. Zwei bekannte Algorithmen sind der Ball-Pivoting Algorithmus (BPA) [3] und der Greedy Delaunay-based Oberflächen Rekonstruktions Algorithmus [8]. Diese unterscheiden sich hinsichtlich der Wahl ihrer Startkomponenten und der Art der Erweiterung dieser. Allgemein jedoch verfolgen sie den selben Ansatz.

Zu Beginn eines Advancing Front Algorithmus müssen Startdreiecke generiert werden. Diese werden entweder zufällig oder nach bestimmten Kriterien ausgewählt. Somit hat man nach dem ersten Schritt eine Menge an Dreiecken bzw. deren Kanten, an denen man die gesuchte Oberfläche aufbaut. Wie die wachsende Region nun aufgebaut wird, hängt am gewählten Verfahren ab.

Der BPA nutzt beispielsweise die Kanten der Seed-Dreiecke und speichert diese vorübergehend als Grenzkanten. An diesen wird jeweils ein Ball mit definiertem Radius r gedreht bis ein neuer Eingangspunkt getroffen wird. Im Falle eines erfolgreichen Treffers wird ein neues Dreieck zwischen der bestehenden Kante und dem neuen Messwert der Triangulierung hinzugefügt. Die hinzukommenden Kanten werden im Anschluss als neue Grenzkanten gesehen, sodass an ihnen im folgenden Verlauf der selbe Schritt wiederholt werden kann. Sobald keine frischen Kanten mehr hinzugefügt werden können, also durch das Ball-Pivoting kein Messwert mehr getroffen wird, ist der Algorithmus fertig. Die voranschreitende Menge an Kanten bzw. Dreiecken ist die entstandene Triangulierung, das Ergebnis des Verfahrens.

Der Greedy Delaunay-based Oberflächen Rekonstruktions Algorithmus [8] von David Cohen-Steiner und Frank Da ist, wie bereits erwähnt, ebenfalls ein Advancing Front Algorithmus. Dieser generiert zunächst eine Delaunay Triangulierung um im späteren Verlauf seine Startdreiecke zu wählen. Ebenfalls wird nach be-

stimmten Kriterien eine Priorisierung von potentiellen neuen Dreiecken anhand der Delaunay Triangulierung erzeugt. Die Delaunay Dreiecke mit den kleinsten Radien werden hier als Startdreiecke gewählt. Die Kanten dieser werden, wie beim BPA, als wachsende Front gesehen und hinsichtlich der potentiellen neuen Dreiecke erweitert. Die Erweiterung findet nach 4 verschiedenen Arten definiert in [8] statt. Nachdem die Menge an Dreiecken ebenfalls soweit gewachsen ist, dass keine neuen Dreiecke aufgrund der Eingangspunkte hinzugefügt werden können, ist der Algorithmus beendet.

Im Folgenden wird stellvertretend für diese Klasse noch einmal genauer der Ball-Pivoting Algorithmus erläutert, da dieser ebenfalls Bestandteil der nächsten Rekonstruktionsstrategie ist.

4.5 Scale Space Meshing

Das Scale Space Meshing ist eine besondere Art eines Advancing Front Algorithmus. Aufgrund der abweichenden Strategie würde ich dieses Verfahren jedoch extra behandeln. Der Algorithmus wurde 2011 von Julie Digne et al. [11] veröffentlicht. Das Vorgehen unterscheidet sich indem, dass die Punktewolke zunächst aufgrund ihrer Krümmung analysiert und vereinfacht wird. Ein sogenannter *mean curvature motion* Ansatz (MCM) wird direkt auf die Punktewolke angewandt und diese wird somit geglättet. Nach einer definierbaren Anzahl an Glättungsschritten wird ein Advancing Front Algorithmus genutzt um daraus ein Oberflächennetz aufgrund der entrauschten Punktewolke zu erstellen. Die ursprünglichen Messwerte werden jedoch nicht vergessen und es findet zusätzlich eine Projektion von der angepassten Punktewolke zur Ursprünglichen statt. Somit besteht das Resultat nicht nur aus fast allen Messwerten, sondern auch aus den Knotenpunkten der originalen Position dieser.

Im Folgenden wird auf den Ansatz von Julie Digne et al.[11] noch einmal genauer eingegangen und die einzelnen Schritte erläutert.

4.6 Rekonstruktion mithilfe der medialen Achsen-Transformation (MAT)

Als besondere Strategie wird der Ansatz vorgestellt, welcher anhand der medialen Achsen-Transformation versucht, eine Oberfläche zu rekonstruieren.

Der Power Crust Algorithmus von Nina Amenta, Sunghee Choi und Ravi Krishna Kolluri [23] ist ein Verfahren, welches zunächst versucht die mediale Achsen-Transformation anzunähern und folglich ein Oberflächennetz zu erzeugen. Die MAT selbst wird in [23] als Skelett-Form Repräsentation beschrieben, welche in verschiedensten Anwendungen im Bereich Form-Manipulation, Form-Erkennung und Animation genutzt wird. Sie beschreibt die mediale Achse und eine Menge an maximalen Bällen komplett im inneren der Form. Der folgende Schritt der Oberflächen-Rekonstruktion wird in der Arbeit von Nina Amenta, Sunghee Choi und Ravi Krishna Kolluri [23] als inverse bezeichnet, da zunächst anhand der Messproben eine Art Skelett der Form angenähert wird und im Anschluss die Berechnung der Oberfläche anhand dieser stattfindet und nicht mehr direkt anhand der Messwerte.

Der Algorithmus verwendet zunächst die Technik eines Voronoi-Diagramms. Anhand dessen werden Voronoi-Bälle mit den einzelnen Voronoi-Knoten als Mittelpunkt definiert. Hierbei wird darauf geachtet, dass nur bestimmte Bälle, die Pole, verwendet werden. Diese beziehen sich auf maximal leere Bälle komplett außerhalb, wie auch innerhalb der Struktur. Anhand dieser Substruktur der Voronoi-Bälle wird das so definierte Power-Diagramm benannt. Dieses beinhaltet die Mittelpunkte und Radien der einzelnen Bälle. Der so definierte Abstand zur Oberfläche wird in [23] als Gewichtung bezeichnet. Mithilfe des Power-Diagramms wird versucht die Oberfläche, bzw. passend genannt in [23] die (Power) Kruste, als Grenzschicht zwischen den Scharen an äußeren und inneren Bällen anzunähern. Die Kruste selbst ist ein Polygonnetz.

Im Folgenden wird auf den Power Crust Algorithmus [23] noch einmal genauer eingegangen. Die einzelnen Bestandteile, wie Pole, maximale Bälle, Voronoi-Bälle und weitere Definitionen werden zudem ebenfalls erläutert um die einzelnen Schritte des Algorithmus nachvollziehen zu können.

5 Marching Cubes

Der Marching Cubes Algorithmus wird stellvertretend für die Klasse der Iso-Oberflächen Auswertungsmethoden noch einmal genauer vorgestellt. Wie im Artikel [7] beschrieben, wird für die Rekonstruktion eine Punktewolke mit jeweiligen Intensitätswerten (Iso-Werten) bereitgestellt. Das Verfahren beinhaltet zwei primäre Schritte. Zuerst wird die Oberfläche entsprechend einem benutzerdefinierten Wert lokalisiert und dann die Dreiecke der Oberfläche darin ausgewertet. Um ein qualitativ hochwertiges Bild der Oberfläche zu gewährleisten, berechnet man folgend die Normalen zu jedem Knotenpunkt der Flächen. [7] Der Marching Cubes Algorithmus nutzt, wie bereits zur Strategie der Iso-Oberflächen Auswertungen erläutert, ein Voxelgitter, über welches er mithilfe der einzelnen Voxel/Würfel iteriert. Mithilfe einer Nachschlagetabelle werden in den Würfeln Flächen und Kanten gebildet. Sind alle Voxel durchlaufen ist der Algorithmus fertig und die Summe aller entstandenen Knotenpunkte, Kanten, Flächen und Normalen ergibt die resultierende Triangulierung.

William E. Lorensen und Harvey E. Cline [7] präsentieren in ihrer Veröffentlichung folgende Zusammenfassung des Vorgehens des Algorithmus.

Vorgehen

1. Lese zunächst 4 Schichten des Raumes in den Speicher
2. Scanne zwei Schichten und kreiere einen Würfel anhand 4 Nachbarn einer Schicht und 4 Nachbarn der nächsten Schicht
3. Berechne einen Index für den Würfel, indem Sie einen Vergleich zwischen den 8 Dichtewerte der Würfelecken und der Flächenkonstante aufstellen
4. Mithilfe des kalkulierten Indexes werden die Liste der Kanten in einer vorberechneten Tabelle nachgeschlagen

5. Mithilfe der Dichtewerte der Eckpunkte wird die Schnittkante der Oberfläche interpoliert
6. Interpoliere einen Normalenvektor für jeden Knotenpunkt
7. Gib zuletzt alle Dreiecksknotenpunkte und zugehörige Normale aus

5.1 Definition Schichten und Index

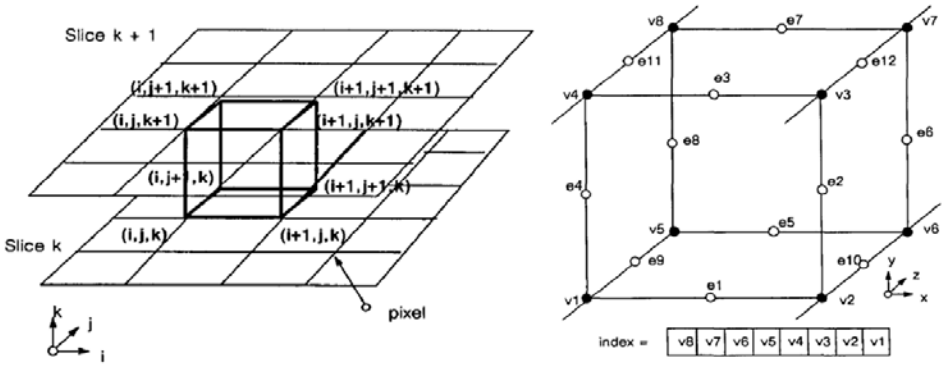

Abb. 5.1: Marching Cubes - Schichten und Indexe [7]

In der Zusammenfassung des Algorithmus werden Schichten und Indexe erwähnt. Diese sind in Abbildung 5.1 visualisiert. Die einzelnen Schichten sind parallele Querschnitte im Raum, anhand welcher die Voxelgitterstruktur aufgebaut wird. Das Raster innerhalb einer Schicht ergibt sich durch die Abstände dieser, denn die zu betrachtenden Konstrukte sind Würfel, deren Kanten alle gleich lang sind. Schritt 1 und 2 des Vorgehens beinhalten diesen Vorgang. Zunächst werden 4 Schichten eingelesen und Würfel innerhalb zweier benachbarter Querschnitte konstruiert. Der Index eines Würfels wird anhand seiner Eckpunkte berechnet. Die Werte der Eckpunkte können entweder 1 oder 0 annehmen. Dies ist in [7] als 1 für einen Pixel außerhalb des Objektes und 0 für einen Pixel innerhalb des Objektes definiert. Die Werte werden zugewiesen, indem für die gesuchte Oberfläche ein bestimmter Intensitätswert festgelegt wird. Ist der Intensitätswert folglich größer oder kleiner, werden die Werte für außerhalb und innerhalb zugewiesen.

Die gesuchte Oberfläche ist somit im Raum eines Voxels, der beide Varianten an Eckpunktwerten besitzt. Besteht der betrachtete Würfel im Gegensatz dazu nur aus den Werten 0 bzw. nur aus 1, ist der gesamte Bereich dessen entweder innerhalb oder außerhalb des Objektes. Somit wird hier kein Oberflächennetz erstellt, da der Raum nicht teil der Oberfläche anhand der Messwerte ist. Dies spiegelt die Indexe 0 und 255 wieder. Die anderen Indexe von 1 bis 254 ergeben Würfel aus der Nachschlagetabelle zum erzeugen der Triangulierung.

5.2 Nachschlagetabelle

In der Tabelle sind für jeden Index die jeweiligen zu erzeugenden Kanten gespeichert. Anhand dieser werden später die Schnittpunkte im Würfel berechnet und es ergeben sich somit Knotenpunkte, Kanten und Flächen. Wie bereits im vorherigen Kapitel beschrieben, können die 256 verschiedenen Indexvariationen auf 15 Unterschiedliche reduziert werden. Abgesehen von den leeren Behältern, also Index 0 und 255, gibt es somit 14 Basisvariationen, die je nach Index gespiegelt und rotiert werden können. Zu betrachten sind diese Muster in Abbildung 4.1. Technisch gesehen verweist ein Index auf eine Liste von Kanten, welche für die jeweilige Konstellation die benötigten, schneidenden Kanten zwischen Oberfläche und Würfel angibt. Für den genauen Schnittpunkt auf der Kante des Würfels wird in einem nächsten Schritt zwischen den Eckpunkten interpoliert.

5.3 Interpolation

Interpoliert werden müssen nun noch die genauen Schnittpunkte und deren Normalenvektor. Auf welcher Kante des betrachteten Würfel ein Schnittpunkt existiert, wird wie beschrieben durch den Index in der Nachschlagetabelle definiert. William E. Lorensen und Harvey E. Cline [7] schildern, dass sie eine lineare Interpolation anstelle von Interpolationen höherer Ordnung verwendet haben, da in ihren Experimenten hierzu der qualitative Vorteil einer komplexeren Berechnung nur minimal war. Somit wird der Schnittpunkt auf der jeweiligen Kante des Würfels aufgrund der Intensitätswerte der Eckpunkte linear interpoliert.

Schließlich wird der Normalenvektor des Schnittpunktes ebenfalls auf die selbe

weise angenähert. Um diese Interpolation durchführen zu können, muss zunächst der Gradient der Eckpunkte berechnet werden. Der Gradient kann laut [7] gebildet werden durch die Ableitung der Dichte-Funktion.

$$\vec{g}(x, y, z) = \nabla \vec{f}(x, y, z) \tag{5.1}$$

Der Gradient eines Pixels (x, y, z) wird mithilfe des Unterschieds in allen drei Koordinatenachsen berechnet. Aus diesem Grund werden auch 4 Schichten im Speicher behalten. Die Werte in Richtung der einzelnen Koordinatenachsen ergeben sich somit aus:

$$G_x(i, j, k) = \frac{D(i+1, j, k) - D(i-1, j, k)}{\Delta x} \tag{5.2}$$

$$G_y(i, j, k) = \frac{D(i, j+1, k) - D(i, j-1, k)}{\Delta y} \tag{5.3}$$

$$G_z(i, j, k) = \frac{D(i, j, k+1) - D(i, j, k-1)}{\Delta z} \tag{5.4}$$

Die Funktion $D(i, j, k)$ gibt die Dichte eines Pixels wieder, welcher durch die Parameter i, j, k festgelegt wird. Diese beziehen sich auf die vorher definierte Rasterstruktur i, j für die jeweilige Schicht k. Sind die Gradienten für der Eckpunkte des Würfels bekannt, werden die jeweiligen Kantenenden einer Kante verwendet, um die Normale des Schnittpunktes linear zu interpolieren.

5.4 Erweiterungen

Der vorherig beschriebene Ablauf wird für jede Schicht durchgeführt, bis der Algorithmus über den kompletten Bereich „marschiert“ist. Nachdem zwischen zwei Schichten das jeweilige Raster abgearbeitet wurde, kann die älteste der 4 Schichten aus dem Speicher entladen und eine Neue gelesen werden. Jeder generierte Voxel muss jedoch nicht alle Schritte durchlaufen, denn William E. Lorensen und Harvey E. Cline [7] stellen zwei zusätzliche Laufzeitoptimierungen vor. Diese werden folglich erläutert.

5.4.1 Effizienteres Vorgehen

Um ein effizienteres Vorgehen zu gewährleisten wird der Zusammenhang zwischen den benachbarten Voxeln betrachtet. Für jeden Würfel, welcher einen Vorgänger besitzt, sind gegebenenfalls schon Schnittpunkte auf dessen Kante bekannt. Dies liegt daran, dass sich benachbarte Voxel Kanten teilen. Im Falle, dass benachbarte Voxel bereits Teil der vorherigen Berechnungen waren, ist die gemeinsame Kante schon betrachtet worden. Insgesamt muss zwar jedes Mal der Index für die Nachschlagetabelle berechnet werden, jedoch nicht alle linearen Interpolationen für jede der 12 Kanten des Würfels.

Die meisten bekannten Kanten haben somit alle Würfel, welche als Bestandteil nicht die erste Schicht, die erste Linie und den ersten Pixel pro Linie beinhalten. Hat ein Voxel somit die Eigenschaft, dass sein Schichtparameter $k > 0$, sein Spaltenparameter $i > 0$ und sein Zeilenparameter $j > 0$ ist, sind bereits 9 der 12 Kanten bekannt [7]. In diesen Fällen müssen die Berechnungen nur noch für die fehlenden drei Kanten berechnet werden. Unabhängig von der Anzahl der bekannten Kanten pro Würfel, wird allgemein die Anzahl an Berechnungsschritten verkleinert.

Diese Erweiterung verbessert den Algorithmus somit hinsichtlich Rechenaufwand, wie folglich auch Laufzeit.

5.4.2 Funktionale Verbesserung

Die funktionale Verbesserung beschrieben in [7] bezieht sich auf den Fall, dass in der Medizin oft auch mehrere Oberflächen mit unterschiedlichem Intensitätswert berechnet werden wollen. Einfach vorzustellen ist, dass mithilfe einer Wahrheitstabelle entschieden werden kann, dass ein Voxel komplett außerhalb, bzw. auch innerhalb eines Objektes hinsichtlich der betrachteten Oberfläche bzw. des jeweiligen Intensitätswerts ist. Dies wäre der einfache Fall, wenn der Index 0 bzw. 255 entspricht. Im Falle von mehreren zu berechnenden Oberflächen, kann eine Wahrheitstabelle eingeführt werden, welche abwägt, ob ein Würfel mit zwei unterschiedlichen Oberflächen interagiert. In diesem Falle kann anhand der Wahrheitstabelle von [7] entschieden werden, ob in einem Würfel Dreiecke einer der beiden Oberflächen überhaupt entstehen oder nicht.

Diese Erweiterung ist speziell im medizinischen Bereich interessant, da hier oft verschiedene Objekte, also auch unterschiedliche Oberflächen eines CT, MRT oder SPECT Scann wichtig sind. Für die Iso-Oberflächen Auswertung einer einzigen Oberfläche, ist dies nicht erforderlich. Dies trifft ebenfalls auf die impliziten Rekonstruktionen zu, welche den Marching Cubes Algorithmus bzw. angelehnte Verfahren für die Auswertung ihrer impliziten Funktion nutzen.

Hinweis

Alle theoretischen Ausformulierungen des Algorithmus beruhen auf den Originalen von William E. Lorensen und Harvey E. Cline [7].

6 Poisson Oberflächen Rekonstruktion

Die Poisson Oberflächenrekonstruktion [17] wird stellvertretend für die impliziten Methoden vorgestellt. Das Verfahren benötigt zunächst eine orientierte Punktewolke. Die Idee um im Anschluss auf eine angenäherte Oberfläche zu kommen, beruht auf zwei wichtigen Abschnitten. Zunächst versucht man mithilfe der Eingangspunkte eine Indikatorfunktion aufzustellen. Diese nimmt den Wert 0 außerhalb der Oberfläche an und den Wert 1 innerhalb. Der Zusammenhang zwischen Indikatorfunktion und orientierter Punktewolke wird in [17] geschildert, so dass die Eingangspunkte als Stichprobe des Gradienten der Indikatorfunktion gesehen werden können. Diese Beziehung wird ausgenutzt um die implizite Funktion zu erhalten. Sobald innerhalb und außerhalb der Oberfläche definiert wurde, wird in einem weiteren Abschnitt die Iso-Oberfläche ausgewertet. Dieser Schritt ähnelt dem bereits vorgestellten Marching Cubes Algorithmus [7], da ein angelehntes Verfahren an diesen die Oberfläche ausliest[17]. Das Resultat ist eine Triangulierung, welche bei der iterativen Auswertung der impliziten Funktion entsteht. Die einzelnen Schritte werden im Folgenden genauer untersucht.

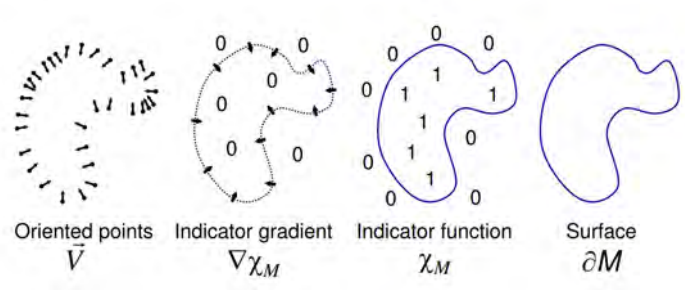

Abb. 6.1: Intuitive Illustration der Poisson Rekonstruktion in 2D [17]

Das Verfahren wird in [17] hinsichtlich zweier Modelle erläutert. Einerseits an

einem einfacheren Modell, welches annimmt, dass die Punktproben gleichmäßig und konstant verteilt sind. Andererseits an einem komplexeren Modell, welches die lokale Dichte abschätzen muss. Das Abschätzen der lokalen Dichte spiegelt hier jedoch nicht eine Auswertung der Oberfläche in einer lokalen Nachbarschaft wieder. Michael Kazhdan et al. [17] schildern, dass deren Verfahren eine stark geglättete Oberfläche liefert, welche aus einer globalen Lösung der Indikatorfunktion erreicht wird. Zunächst einmal die Ausformulierungen des einfachen Modells.

6.1 Einfaches Modell

Die Hauptherausforderung ist es die Indikatorfunktion mithilfe der Eingangspunkte zu berechnen. Um dies zu erreichen wird eine Beziehung zwischen dem Gradienten der Indikatorfunktion und einem Integral des Flächennormalenfeldes hergeleitet. Das Flächenintegral wird durch die Aufsummierung über die orientierten Punkte erreicht. Abschließend rekonstruiert man die Indikatorfunktion aus diesem Gradientenfeld als Poisson-Problem[17].

6.1.1 Datenstrukturen und Modelldefinition

Das Ziel der Rekonstruktion ist eine Repräsentation der Oberfläche. Die implizite Funktion, die diese darstellt, muss also speziell entlang der Oberfläche präzise sein. Außerhalb oder innerhalb dieser wird glücklicherweise vom Verfahren keine Genauigkeit verlangt. Dies motivierte Michael Kazhdan et al. [17] einen anpassungsfähigen Octree als Repräsentation der Funktion, wie auch zum Lösen des Problems zu verwenden.
Die Datenstruktur ist bei Informatikern bekannt und vergleichbar mit dem sehr oft genutzten Binärbaum. Die einzelnen Dimensionen werden jeweils in 2 Nachkommen unterteilt, womit man im dreidimensionalen Raum auf 2^3 also 8 Nachkommen kommt. Jeder Knoten kann entweder diese 8 Nachkommen besitzen oder keine. Die Wurzel spiegelt den gesamten Raum wieder. Jeder nachfolgende Knoten repräsentiert jeweils einen Oktanten des vorherigen Raumes. Somit kann schnell über einen dreidimensionalen Raum iteriert werden, bei welchem nur die Punkte nahe der Oberfläche interessant sind.
Für jeden Knoten o des Octrees θ wird eine Funktion F_o verbunden, sodass folgende Bedingungen erfüllt sind [17]:

1. Das Vektorfeld $\vec{V}$ kann präzise und effizient als lineare Summe der Funktion F_o dargestellt werden.

2. Die Matrixdarstellung der Poisson-Gleichung, ausgedrückt mithilfe der Funktionen F_o, kann effizient gelöst werden.

3. Eine Darstellung der Indikatorfunktion als Summe der Funktionen F_o kann präzise und effizient nahe der Oberfläche des Objektes ausgewertet werden.

6.1.2 Definition des Funktionsraumes

Der Funktionsraum wird mithilfe einer linearen Hülle umgesetzt, welche definiert wird als $F_{\theta,F} \equiv Span\{F_o\}$. Um die einzelnen Funktionen F_o zu erhalten, müssen im Vorfeld Parameter für den Octree θ gewählt werden. Dieser wird aufgrund der eingehenden Messpunkte und einer maximalen Baumtiefe D definiert. Jeder Punkt der Eingangsmenge fällt in ein Blatt des Baumes mit der vorherigen festgelegten Baumtiefe D. Somit erhalten wir einen minimalen Octree für unser Problem.

Als nächstes werden die Funktionen gebildet. "Für jeden Knoten $o \in \theta$ wird eine Funktion F_o als Einheitsintegral gesetzt, zentriert zum Knoten o und ausgedehnt durch die Größe von o "[17] :

$$F_o(q) \equiv F\Big(\frac{q - o.c}{o.w}\Big)\frac{1}{o.w^3} \tag{6.1}$$

Das Zentrum des Knotens entspricht der Variable $o.c$ und die Weite ist $o.w$. Der Funktionsraum $F_{\theta,F} \equiv Span\{F_o\}$ hat somit eine mehrfach auflösende Struktur, welche Ähnlichkeiten zur bekannten Wavelet Transformation aufweist [17]. Die erwünschten Eigenschaften sind, dass feinere Knoten mit hochfrequenten Funktionen assoziiert werden und die Darstellung der Funktion umso präziser wird, je näher an der Oberfläche.

6.1.3 Auswählen der Basisfunktion

Das Ziel eine Basisfunktion F zu wählen ist, dass das Vektorfeld $\vec{V}$ präzise und effizient als die lineare Summe der Knotenfunktionen $\{F_o\}$ repräsentiert werden kann. Dies wird in [17] erreicht, indem die Position jedes Eingangspunktes zunächst durch den Mittelpunkt seines Blattes der Baumstruktur ersetzt wird:

$$F(q) = \tilde{F}\Big(\frac{q}{2^D}\Big) \tag{6.2}$$

Der entstehende Fehler bewegt sich hier im Rahmen der halben Abtastrate, welche 2^{-D} entspricht. Um die Präzision für jeden Unterknoten zu erhöhen wird in der Praxis eine trilineare Interpolation verwendet und nicht direkt das Zentrum eines Blattknotens [17].

6.1.4 Definition des Vektorfeldes

Das Vektorfeld ist in [17] definiert als:

$$\vec{V}(q) \equiv \sum_{s \in S} \sum_{o \in Ngbr_D(s)} \alpha_{o,s} F_o(q) s.\vec{N} \tag{6.3}$$

Diese Approximation des Gradientenfeldes behilft sich einerseits der 8 am nächsten liegenden Punkte der selben Baumtiefe des betrachteten Punktes ($Ngbr_D(s)$) und andererseits der Variable $\alpha_{o,s}$, welche die jeweiligen Gewichtungen der trilinearen Interpolation widerspiegelt. Der Messpunkt wird somit auf seine 8 Nachbarn angepasst. Die Unterknotenpräzision wird somit gegenüber der Wahl des Zentrums des Blattknotens erhöht.
Da die Eingangspunkte im einfachen Modell gleichmäßig verteilt sind, wird angenommen, dass der Bereich einer bestimmten Stelle $\mathcal{P}_s$ konstant und $\vec{V}$ eine gute Approximation ist [17].

6.1.5 Lösung des Poisson-Problems

Die Poisson Gleichung ist in diesem Sinne keine exakte Lösung des Problems, sondern eine Annäherung [17]. Sie selbst ergibt sich aus der Überlegung, dass nach dem Aufstellen eines Vektorfeldes $\vec{V}$ die Funktion $\tilde{\chi}$ so gelöst werden müsste, sodass $\nabla\tilde{\chi} = \vec{V}$ gilt. Diese exakte Lösung kann jedoch nicht immer bestimmt werden, da das Vektorfeld nicht generell integrierbar ist[17]. Um nun eine Annäherung zu bestimmen wird der Divergenz-Operator angewendet und man erhält die Poisson Gleichung:

$$\Delta\tilde{\chi} = \nabla \cdot \vec{V} \tag{6.4}$$

Eine weiter Herausforderung ist nun die Tatsache, das zwar Vektorfeld $\vec{V}$ und die Gleichung $\tilde{\chi}$ im selben Raum $\mathcal{F}_{\theta,\mathcal{F}}$ sind, dies aber nicht für die Funktionen $\Delta\tilde{\chi}$ und $\nabla \cdot \vec{V}$ gelten muss. Kazhdan et al. [17] weisen darauf hin, dass für die Lösung dieses Problems, die Projektion von $\Delta\tilde{\chi}$ auf den Raum $\mathcal{F}_{\theta,\mathcal{F}}$ nahe der Projektion von $\nabla \cdot \vec{V}$ sein muss. Da zusätzlich die Funktionen F_o generell keine Orthonormalbasis formen, erschwert dies die Findung einer Lösung. Die in [17] gefundene Vereinfachung beruht auf dem Minimieren von $\tilde{\chi}$.

$$\sum_{o\in\theta}\left\|\langle\Delta\tilde{\chi} - \nabla\cdot\vec{V}, F_o\rangle\right\|^2 = \sum_{o\in\theta}\left\|\langle\Delta\tilde{\chi}, F_o\rangle - \langle\nabla\cdot\vec{V}, F_o\rangle\right\|^2 \tag{6.5}$$

Gegeben sei nun der $\|\theta\|$ - dimensionale Vektor v, für welchen gilt, dass die o-te Koordinate $v = \langle\nabla \cdot \vec{V}, F_o\rangle$ ist. Das Ziel ist nun, die Funktion $\tilde{\chi}$ so zu lösen, dass der erhaltene Vektor durch die Projektion des Laplace-Operators von $\tilde{\chi}$ auf die Funktionen F_o, so ähnlich wie möglich mit dem Vektor v ist. Um die Repräsentation als Matrixform zu gewährleisten, wird in [17] festgelegt, dass $\tilde{\chi} = \sum_o \chi_o F_o$ gilt. Anschließend wir eine Matrix L mit der Dimension $|\theta|x|\theta|$ definiert, sodass Lx das Punktprodukt des Laplace-Operators mit jedem der Funktionen F_o ist.

Somit ergibt sich anhand der Definitionen in [17] für jeden (o, o') - Eintrag folgende Gleichung:

$$L_{o,o'} \equiv \left\langle\frac{\partial^2 F_o}{\partial x^2}, F_{o'}\right\rangle + \left\langle\frac{\partial^2 F_o}{\partial y^2}, F_{o'}\right\rangle + \left\langle\frac{\partial^2 F_o}{\partial z^2}, F_{o'}\right\rangle \tag{6.6}$$

Um also $\tilde{\chi}$ zu lösen muss folgende Gleichung gelöst werden.

$$\min_{x\in\mathbb{R}|\theta|} \|Lx - v\|^2 \tag{6.7}$$

Die Gleichung 6.7 minimiert den Fehler zwischen Vektor v und dem Punktprodukt der Matrix L mit x. Kazhdan et al. [17] verweisen auf ein Mehrgitterverfahren, ähnlich dem in [14], um die Beschränkung von L_d von L im Funktionsraum, aufgespannt durch die Tiefe d der Funktionen, zu lösen. Diese Lösung wird zurück projiziert auf $F_{\theta,F}$.

6.1.6 Isosurface Auswertung

Um schließlich eine Iso-Oberfläche der definierten Indikatorfunktion zu entnehmen, muss ein geeigneter Iso-Wert gewählt werden. Dies wird in [17] erreicht, indem $\tilde{\chi}$ an den Messpunkten evaluiert wird und man den durchschnittlichen Wert der Eingangspunkte als Intensitätswert wählt:

$$\partial\tilde{M} \equiv \{q \in \mathbb{R}^3 | \tilde{\chi}(q) = \gamma\} \quad mit \quad \gamma = \frac{1}{|S|} \sum_{s \in S} \tilde{\chi}(s.p) \tag{6.8}$$

Wie bereits erwähnt, funktioniert die Iso-Oberflächen Auswertung ähnlich dem vorgestellten *Marching Cubes Algorithmus* [7]. Der hauptsächliche Unterschied ist, dass die Oberfläche nicht anhand fest definierter Würfelgrößen ausgewertet wird, sondern man verwendet die kleinst mögliche Auflösung des Octrees. Nachdem über den definierten Bereich iteriert wurde, wird die gewünschte Triangulierung anhand der Auswertung erhalten.

6.2 Komplexes Modell

Dem einfachen Modell war die Annahme gegeben, dass davon ausgegangen wird die Punktewolke sei konstant gleichmäßig verteilt. Besteht der Fall, dass diese Annahme nicht zutrifft, wird das Modell erweitert. Zunächst muss die lokale Punktedichte abgeschätzt werden. Anhand dieser wird versucht den jeweiligen anteiligen Einfluss eines Messwertes richtig zu skalieren. Insgesamt weisen Kazhdan et al. [17] daraufhin, dass somit eine Rekonstruktion angestrebt wird, welche in Bereichen mit einer hohen Punktedichte viele Eigenschaften rekonstruieren kann und eine glättende Form in dünn abgetasteten Bereichen verwendet.

6.2.1 Abschätzen der lokalen Punktedichte

Die lokale Punktedichte wird mit einem Kerndichteschätzer berechnet. Der Ansatz versucht herauszufinden, wie viele Nachbarn ein Messwert besitzt. Dies ist implementiert, indem alle Punkte in ein Gitter aufgespalten werden und die zugehörige „Splatting“- Funktion mit einem Glättungsfilter gefaltet wird. Die Faltung wird für jeden Messwert evaluiert. Für die Berechnung wird eine Tiefe $\widehat{D}$ eingeführt, welche kleiner oder gleich der maximalen Baumtiefe D ist.

$$W_{\widehat{D}}(q) \equiv \sum_{s \in S} \sum_{o \in Ngbr_{\widehat{D}}(s)} \alpha_{o,s} F_o(q) \tag{6.9}$$

Baumknoten mit geringerer Auflösung werden mit Gauß-Funktionen größerer Weite angenähert. Deshalb liefert der Parameter $\widehat{D}$ mit einem kleineren Wert eine Abschätzung über eine größere Region.[17]

6.2.2 Anpassung der Berechnung des Vektorfeldes

Insbesondere unter Verwendung der Tatsache, dass die Fläche umgekehrt proportional zur Abtastdichte ist, stellen Kazhdan et al. [17] folgende Gleichung auf.

$$\vec{V}(q) \equiv \sum_{s \in S} \frac{1}{W_{\widehat{D}}(s.p)} \sum_{o \in Ngbr_{\widehat{D}}(s)} \alpha_{o,s} F_o(q) \tag{6.10}$$

Somit ergibt sich jedoch noch eine schwache Rauschfilterung in dünn besetzten Regionen. [17] Die weite des Glättungsfaktors wird aus diesem Grund ebenfalls angepasst. Durch die Anpassung werden feinere Details in dichten Regionen bewahrt und Rauschen in schwach besiedelten Bereichen geglättet. Schließlich ergibt sich die Gleichung:

$$\vec{V}(q) \equiv \sum_{s \in S} \frac{1}{W_{\widehat{D}}(s.p)} \sum_{o \in Ngbr_{Depth(s.p)}(s)} \alpha_{o,s} F_o(q) \tag{6.11}$$

Depth(s.p) liefert die gewünschte Tiefe eines Messwertes $s \in S$. Berechnet wird diese mithilfe von:

$$Depth(s.p) \equiv min(D, D + log_4\Big(\frac{W_{\widehat{D}}(s.p)}{W}\Big)) \tag{6.12}$$

Somit wird gewährleistet, dass die Weite des Glättungsfilters, mit welcher s zu $\vec{V}$ beiträgt proportional zum Radius des Oberflächenbereiches ist. [17]

6.2.3 Anpassung der Wahl eines Iso-Wertes

Schließlich wird die Iso-Oberflächen Auswertung ebenfalls angepasst. Dies wird umgesetzt, indem ein gewichteter Durchschnitt der Werte als Iso-Wert berechnet wird.

$$\partial\tilde{M} \equiv \{q \in \mathbb{R}^3 | \tilde{\chi}(q) = \gamma\} \quad mit \quad \gamma = \frac{\sum \frac{1}{W_{\widehat{D}}(s.p)} \tilde{\chi}(s.p)}{\sum \frac{1}{W_{\widehat{D}}(s.p)}} \tag{6.13}$$

Hinweis

Alle theoretischen Ausformulierungen des Algorithmus beruhen auf den von Kazhdan et al. [17].

7 Ball Pivoting Algorithmus

Bernardini et al. [3] stellten erstmals 1999 ihren Algorithmus zur Oberflächenrekonstruktion vor. Der Ball Pivoting Algorithmus (BPA) erzeugt mithilfe einer Kugel ein Dreiecksnetz aus einer Punktewolke, deren Normalen bekannt ist. Insgesamt kann die Strategie als Advancing Front Verfahren eingestuft werden. Der Algorithmus beginnt mit der Suche nach potentiellen Startdreiecken und arbeitet sich dann anhand einer fortschreitenden Menge an Kanten weiter. Um jede Randkante wird eine Kugel mit festem Radius p rotiert. Streift diese Kugel einen Punkt der Punktewolke, werden anhand diesem neue Kanten und ein neues Dreieck dem Oberflächennetz hinzugefügt. In [3] wird folgendes Skelett des Algorithmus vorgestellt.

Algorithm $BPA(S, \rho)$

1. while (true)
2. while ($e_{(i,j)} = get\_active\_edge(\mathcal{F})$)
3. if ($\sigma_k = ball\_pivot(e_{(i,j)})$ && ($not\_used(\sigma_k)$ || $on\_front(\sigma_k)$))
4. $output\_triangle(\sigma_i, \sigma_k, \sigma_j)$
5. $join(e_{(i,j)}, \sigma_k, \mathcal{F})$
6. if ($e_{(k,i)} \in \mathcal{F}$) $glue(e_{(i,k)}, e_{(k,i)}, \mathcal{F})$
7. if ($e_{(j,k)} \in \mathcal{F}$) $glue(e_{(k,j)}, e_{(j,k)}, \mathcal{F})$
8. else
9. $mark\_as\_boundary(e_{(i,j)})$
10. if ($(\sigma_i, \sigma_j, \sigma_k) = find\_seed\_triangle()$)
11. $output\_triangle(\sigma_i, \sigma_j, \sigma_k)$
12. $insert\_edge(e_{(i,j)}, \mathcal{F})$
13. $insert\_edge(e_{(j,k)}, \mathcal{F})$
14. $insert\_edge(e_{(k,i)}, \mathcal{F})$
15. else
16. return

Der Code schildert den Algorithmus stark vereinfacht und lässt einige Details aus. Denn zunächst wird als Vorverarbeitung ein Voxelgitter gebildet um eine gewisse Nachbarschaft schneller erreichen zu können. Der nächste Schritt ist die Suche nach potentiellen Startdreiecken. Im Anschluss findet das Ball Pivoting statt. Schließlich wird die bestehende Triangulierung erweitert bzw. verändert durch die beiden Operationen Join und Glue. Zusätzlich zum Skelett des Algorithmus wird noch speziell eine Erweiterung vorgestellt.

7.1 Raumeinteilung

Sowohl die Funktion *ball_pivot* (Zeile 3), wie auch die Funktion *find_seed_triangle* (Zeile 10) erfordern ein effizientes Nachschlagen einer Untermenge an Punkten in einer kleinen räumlichen Nachbarschaft eines Punktes [3]. Aus diesem Grund wird für deren Abfrage der Raum in ein regelmäßiges Gitter aus kubischen Zellen eingeteilt. Diese Voxel besitzen eine Kantenlänge von $\delta = 2p$. Die genutzte Datenstruktur zum Organisieren ist eine Liste von einzelnen Unterlisten. Mithilfe von einem Bucket-Sort Algorithmus werden die Punkte den einzelnen Unterlisten repräsentativ für einen Voxel zugeteilt. Für jeden Voxel in der Liste wird gegebenenfalls ein Zeiger auf die jeweilige Unterliste gespeichert. Ist ein Voxel leer wird ein NULL Zeiger geschrieben. Die Anordnung erlaubt einen konstanten Zugriff der Punkte, kann jedoch für gewaltige Datensätze sehr groß werden. Aus diesem Grund wird im Nachhinein noch eine Erweiterung vorgestellt, welche einen External Memory Algorithmus (out-of-core) umsetzt um die Kapazitäten des Arbeitsspeichers zu schonen, da dieser im späteren Verlauf ebenfalls die Menge an aktiven Kanten (Front) beinhaltet.

Der Nutzen der Datenstruktur allgemein zeigt sich bei dem Nachschlagen einer Nachbarschaft zu einem bestimmten Punkt σ. Anhand der Koordinaten von σ kann man sofort den zugehörigen Voxel bestimmen und muss anschließend nur die 27 benachbarten Voxel durchsuchen, da alle Punkte innerhalb einer Distanz von $2p$ interessant sind [3].

7.2 Startdreieck-Auswahl

Für jede zusammenhängende Komponente, durch die kein Pivot-Ball mit Radius p geführt werden kann, ohne dass sie Punkte des Datensatzes schneidet, muss ein Startdreieck gefunden werden. Bestehen in den Messwerten größere Löcher, sodass mithilfe des Pivot-Balles die Oberfläche in diesem Bereich nicht geschlossen werden kann, entstehen Löcher in der Triangulierung. Ist ein bestimmter Bereich der Oberfläche komplett abgetrennt vom Rest, so bewirkt dies nicht eine gemeinsame Oberfläche, sondern mehrere Komponenten. Genau für diese Komponenten reicht jeweils ein Startdreieck um die Oberfläche dieser zu rekonstruieren. In [3] werden folgende Schritte vorgestellt:

1. Wählen Sie einen beliebigen Punkt σ, der noch nicht von der rekonstruierten Triangulation verwendet wird.
2. Betrachten Sie alle Punktpaare σ_a, σ_b in seiner Nachbarschaft in Bezug der Entfernung von σ.
3. Bilde potentielle Startdreiecke $\sigma, \sigma_a, \sigma_b$.
4. Überprüfe ob der Normalenvektor des Dreiecks die selbe Ausrichtung, wie die Normalen der Knotenpunkte hat.
5. Testen Sie, dass ein p-Ball mit Zentrum im äußeren Halbraum alle drei Eckpunkte berührt, jedoch keinen weiteren Messwert.
6. Beende die Suche sobald ein passendes Startdreieck gefunden wurde.

Wie bereits erwähnt, wird für jede Komponente genau ein Startdreieck benötigt. Dies kann jedoch im Falle von stark verrauschten Daten zu Komponenten nahe der Oberfläche führen, welche nicht erwünscht sind[3]. Durch Vorverarbeitung der Eingangspunkte oder durch Nachbessern der Triangulierung vermindert man diesen Effekt.

7.3 Ball Pivoting

Jede Pivoting Operation startet mit einem Dreieck $\tau = (\sigma_i, \sigma_j, \sigma_o)$ und einem Pivot-Ball mit Radius p, welcher alle drei Knotenpunkte berührt. Nehmen wir

an die zugehörige Pivot-Kante ist $e_{(i,j)}$. In der ursprünglichen Position hat der Pivot-Ball Mittelpunkt c_{ijo} und beinhaltet keine weiteren Knotenpunkte. Dies liegt entweder daran, dass Dreieck τ ein Startdreieck ist oder da es durch eine vorgehende Pivot Operation erstellt wurde. Der Pivot-Ball wird nun so um die Kante $e_{(i,j)}$ gedreht, sodass er stets Kontakt mit deren beiden Endpunkten hat. Durch die kontinuierliche Bewegung des Balles und der Berührung von σ_i und σ_j, ergibt sich für das Zentrum c_{ijo} ein Kreis γ. In Abbildung 7.1 kann dieser ebenfalls betrachtet werden. Die Pivoting Kante $e_{(i,j)}$ liegt in der Grafik auf der z-Achse, welche senkrecht zur Ebene der Grafik steht.

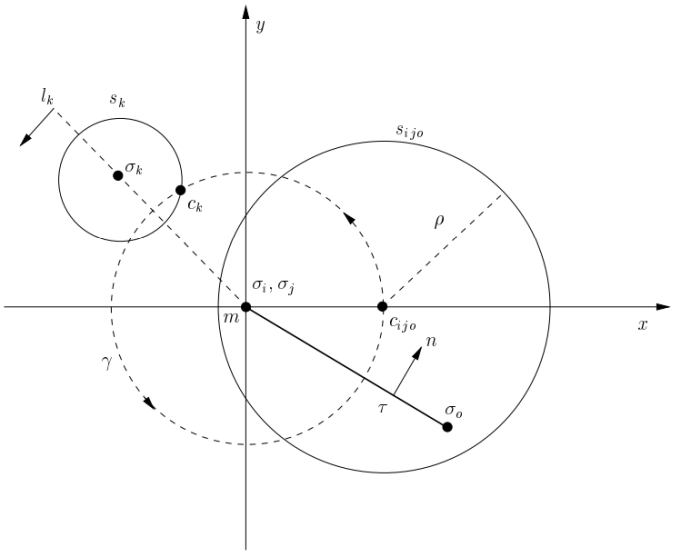

Abb. 7.1: Ball Pivoting Operation [3]

Während der Bewegung des Balles könnte er einen neuen Punkt σ_k treffen. Wird kein Punkt getroffen zählt die Kante als Grenzkante (*boundary*). Anderenfalls wird das Dreieck $(\sigma_i, \sigma_k, \sigma_j)$ als neues gültiges Dreieck gesehen.

In [3] wird ebenfalls darauf hingewiesen, dass in der Praxis die bereits beschriebene Nachbarschaft zum Eingrenzen potentieller Knotenpunkte genutzt wird. Hierbei wird für jeden potentiellen Messpunkt der Nachbarschaft ein Kreis gebildet, welcher sich selbst und die Eckpunkte der Pivot-Kante beinhaltet. Anhand der entstehenden Zentren dieser Kreise wird der nächste Treffer ausgewählt. Neue

Kanten werden grundsätzlich als aktive Kanten angesehen. Alle Kanten können drei Zustände annehmen. Einerseits *active* oder *boundary*, wie auch andererseits den Zustand *frozen*, welcher für die Erweiterung des Algorithmus wichtig ist. In diesem Schritt des Ball Pivoting wird grundsätzlich eine aktive Kante als Pivot Kante verwendet.

7.4 Join und Glue Operation

Diese beiden Operationen überprüfen den Status der Front und bilden die Triangulierung aufgrund bestimmter Gegebenheiten. Eine Join Operation wird nach einem Ball-Pivoting verwendet. Wird bei der vorgehenden Operation ein Knotenpunkt σ_k getroffen, kann dieser Teil der aktuellen Triangulierung sein oder auch nicht.

Der einfache Fall ist, wenn dieser nicht Teil des Netzes ist. In diesem Fall wird das Dreieck $(\sigma_i, \sigma_k, \sigma_j)$ hinzugefügt. Die Pivot Kante $e_{(i,j)}$ wird von der Front entfernt und die beiden anderen Kanten werden um diese erweitert.

Im zweiten Fall ist der getroffene Messpunkt bereits ein Knotenpunkt der Triangulierung. Ob nun ein neues Dreieck hinzugefügt wird, liegt an dem Status von σ_k. Dieser kann ebenfalls zwei Zustände erreichen:

1. σ_k ist ein interner Knotenpunkt der Triangulierung
2. σ_k gehört zu einer anderen Kante der Front

Im ersten Fall kann kein neues Dreieck gebildet werden und die Kante wird als Grenzkante definiert. Gehört jedoch der Punkt zur Front, kann ein neues Dreieck dem Oberflächennetz hinzugefügt werden. Die hierbei entstehenden Kanten können gegebenenfalls mit anderen deckungsgleichen Kanten, unterschiedlicher Orientierung, zusammenfallen. Diese werden sofort mithilfe der Glue Operation entfernt.

7.5 Erweiterung

Wie bereits beschrieben wird zusätzlich zum restlichen Algorithmus ein Verfahren verwendet, welches externen Speicherplatz nutzt. Der Arbeitsspeicher kann

beispielsweise stark begrenzt sein und somit wurde in [3] präsentiert, wie bei einer großen Anzahl an aktiven Kanten der verwendete Speicherbedarf geschont werden kann und ein Teil dieser ausgelagert. Um dies zu erreichen werden zwei Ebenen als Grenzen definiert, welche den aktuell betrachteten Bereich einschränken. Wird innerhalb einer Pivoting Operation ein Punkt außerhalb dieser Grenzen getroffen, werden die neuen Kanten zunächst als *frozen* betrachtet. Sind nur noch solche Kanten vorhanden, werden die Ebenen bewegt. Somit wird ein neuer Arbeitsbereich eingerichtet und die, zu diesem Zeitpunkt, eingefrorenen Kanten werden als aktiv betrachtet.

Sobald die Ebenen über alle Messwerte im Raum bewegt wurden und die Front leer ist, endet der Algorithmus. Das Ergebnis ist eine Triangulierung mit allen getroffenen Messpunkten, allen hinzugefügten, nicht wieder entfernten Kanten und den daraus entstehenden Dreiecken.

Hinweis

Alle theoretischen Ausformulierungen des Algorithmus beruhen auf den von Bernadini et al. [3].

8 Scale Space Meshing

Das Scale Space Meshing ist eine besondere Art eines Advancing Front Algorithmus von Julie Digne et al. [11]. Das Verfahren versucht aus einer unorientierten Punktewolke eine Triangulierung zu erzeugen. Um dies zu erreichen, werden zunächst die Normalenvektoren für jeden Punkt abgeschätzt. Das Abschätzen der Normalen basiert bereits auf einem Scale Space Ansatz. Die Messpunkte werden mithilfe mehrerer iterativer Schritte angepasst. Jeder Schritt nutzt ein *mean curvature motion (MCM)* Verfahren, welches einen Messwert aufgrund der Regressionsebene seiner Nachbarschaft anpasst. In der geglätteten Punktewolke werden dann zunächst die Normalenvektoren der Punkte berechnet. Das eigentliche Rekonstruktionsverfahren wird ebenfalls auf die angepassten Daten angewandt. Im originalen Ansatz von Julie Digne et. al [11] wird für dies der bereits erläuterte *Ball Pivoting* Algorithmus verwendet. Im Anschluss werden die Knotenpunkte des Netzes auf ihre originale Position projiziert. Das Resultat besteht somit zum Größten Teil aus den ursprünglichen Messwerten. Die Scale Space Analyse liefert somit keines Falles ein geglättetes Endresultat [11].

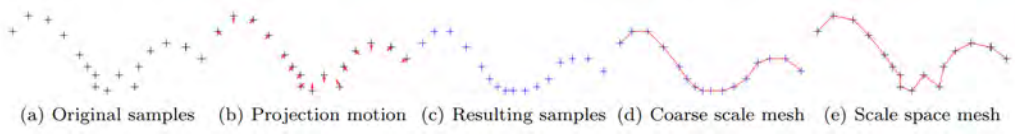

Abb. 8.1: 2D Beispiel der einzelnen Schritte des Scale Space Verfahrens [11]

In Abbildung 8.1 sind die einzelnen Schritte für ein zwei dimensionales Beispiel visualisiert. Hier kann man betrachten, wie die Messwerte der abgetasteten Kurve geglättet werden, indem sie aufgrund der mittleren Krümmung der Nachbarschaft angepasst worden sind. Im Anschluss wird die Kurve berechnet und schließlich auf die eigentlichen Messwerte zurück projiziert. Die einzelnen Abschnitte des Algorithmus werden im Folgenden genauer erläutert.

8.1 Scale Space Orientierung

Algorithm 1: OrientateFromNeighbors(p,r,t)

Data: p an unoriented point, a threshold $0 < t < 1$, a radius r, the set $\mathcal{N}_r(p)$ of p's neighbors within radius r
Result: true if the point was oriented, false otherwise

Compute the normal direction $\mathbf{n}$ of p by local PCA;
$\bar{\mathbf{n}} \leftarrow$ unit mean of neighboring oriented normals;
if $(\bar{\mathbf{n}} \cdot \mathbf{n})^2 > t$ **then**
  if $\bar{\mathbf{n}} \cdot \mathbf{n} > 0$ **then**
    $\mathbf{n}(p) = \mathbf{n}$;
  else
    $\mathbf{n}(p) = -\mathbf{n}$;
  Return true;
else
  Return false;

Algorithm 2: Scale space Orientation

Data: A point cloud $\mathcal{P}$, a radius r, an update parameter $\alpha > 1$

Iterate the projection filter T_r and keep track of each raw data point sample (mean curvature motion);
Find a point p_0 in a flat area, pick its orientation and mark it as oriented. Add its neighbors to the stack $\mathcal{S}$;
while $\mathcal{S}$ *is not empty or* $\mathcal{S}$ *does not become constant* **do**
  Take p the first point in $\mathcal{S}$;
  if *orientateFromNeighbors(p,r,t)* **then**
    Mark the point as oriented and remove p from $\mathcal{S}$;
  Add the neighbors of p to $\mathcal{S}$;
Add all remaining unoriented points to $\mathcal{S}$;
while $\mathcal{S}$ *is not empty and* $\sharp\mathcal{S}$ *does not become constant* **do**
  $r = \alpha r$;
  for p *in* $\mathcal{S}$ **do**
    Perform *orientateFromNeighbors(p,r,t)*;

Die oben aufgeführten Algorithmen 1 und 2 werden in [11] zur Verfügung gestellt um das Vorgehen der iterativen Normalen-Orientierung zu erläutern. Die Funktion *OrientateFromNeighbors(p,r,t)* wird innerhalb der Schleife der zweiten Methode aufgerufen. Mithilfe einer *Principal Component Analyse (PCA)* der lokalen Nachbarschaft eines Punktes wird die Richtung des Normalenvektors statistisch ausgewertet. Dieser Vektor kann jetzt jedoch zwei entgegengesetzte Richtungen aufweisen. Aus diesem Grund wird, falls möglich, die Orientierung der Nachbarschaft zu Hilfe gezogen und das Vorzeichen des Vektors anhand dieser lokalen Gegebenheiten angepasst. Die Parameter für das gesamte Vorgehen, gleich Algorithmus 2, sind die gegebene Punktewolke, ein Radius zur Nachbarschaftsbestimmung, ein Parameter zur Anpassung des Radius und schließlich einen Grenzwert für die Überprüfung, ob ein neuer/angepasster Normalenvektor lokal zulässig ist.

8.2 Projizieren auf Regressionsebene

Das Entrauschen der Messwerte findet in mehreren gleichen, iterativen Schritten statt. In der Praxis hat es sich bewehrt, dass die Glättung zur Rekonstruktion ausreichend genug ist, wenn man 4 Iterationsschritte durchführt [11].

Zu dem Zeitpunkt der Bewegung der einzelnen Messwerte der Punktewolke wurde der Schritt der Normalen-Abschätzung bereits durchgeführt. In [11] wird mathematisch zudem bewiesen, dass der bestehende Normalenvektor eines Punktes, bis auf einen unerheblichen Faktor, dem Vektor des Punktes zur Regressionsebene gleicht. Diese Ebene wird ebenfalls, wie zuvor anhand der lokalen Nachbarschaft aufgestellt.

Die eigentliche Anpassung pro Messwert der Punktewolke erfolgt somit aufgrund des Vektors vom Punkt selbst zur Regressionsebene. Die Bewegung pro Punkt muss für eine spätere Rückprojizierung gespeichert werden. Da jeder Punkt aufgrund mehrerer iterativer Schritte öfters angepasst wird, muss jede Translation hinterlegt werden. Die einzelnen Vektoren, um die ein Messwert pro Iteration angepasst wird, können aufsummiert werden. Somit reicht am Ende des Algorithmus eine Rücktransformation pro Punkt.

8.3 Meshing

Wie bereits erwähnt, wird die entrauschte Punktewolke mithilfe eines Advancing Front Algorithmus verarbeitet. In [11] wird für diese Rekonstruktion der bereits erläuterte *Ball Pivoting* Algorithmus verwendet. Nachdem das Oberflächennetz erzeugt wurde, wird die Rücktransformation jedes verwendeten Knotenpunktes durchgeführt. Somit ändern sich die Positionen der Eckpunkte der primitiven Flächen und die resultierende Triangulierung ist fertig.

8.4 Scale Space Problematiken

Julie Digne et al. [11] weisen in ihrer Veröffentlichung ebenfalls auf einige theoretische Problematiken hinsichtlich der einzelnen Translationen des Scale Space Verfahrens hin.

8.4.1 Problem Regressionsebene

Um eine Regressionsebene zu erstellen, benötigt man mindestens 3 Nachbarn in der Nachbarschaft des betrachteten Punktes um diese zu definieren. Besitzt diese Nachbarschaft weniger als diese, kann keine Ebene definiert werden. Somit kann auch keine Projektion des Punktes durchgeführt werden. Messwerte mit solchen

Eigenschaften sind in der Praxis laut [11] sehr wenige im Vergleich zur Gesamtheit der Anzahl der Punktewolke, sodass diese Proben vernachlässigt werden können. Zur Rekonstruktion einer Oberfläche nutzt man solche Punkte nicht mehr, sie werden in diesem Schritt eliminiert.

8.4.2 Kreuzende primitive Flächen

Nach der Rücktransformation der Knotenpunkte kann die entstehende Triangulierung kreuzende Flächen besitzen. „In der Tat, wenn zwei Punkte zu nahe beieinander liegen, dann kann [deren] Position wechseln in den Scale Space Iterationen, was zu einer komplizierten Oberflächentopologie führt“[11]. Dies wird in einem weiteren Schritt gegebenenfalls behoben, indem kreuzende Flächen identifiziert und deren Kanten geflippt werden. Für eine reine visuelle Darstellung sind ein paar überlappende Dreiecke nicht weiter schlimm bzw. auffallend [11]. Außerdem weisen Sie darauf hin, dass dieser Effekt nur sehr selten bis gar nicht in den meisten Experimenten aufgetreten ist.

Hinweis

Alle theoretischen Ausformulierungen des Algorithmus beruhen auf den von Julie Digne et al. [11].

9 Power Crust Rekonstruktion

Der Power Crust Algorithmus von Nina Amenta, Sunghee Choi und Ravi Krishna Kolluri [23] wird in diesem Kapitel genauer erläutert. Die Strategie zunächst die mediale Achsen-Transformation (MAT) anzunähern und folglich diese Verwenden um eine Oberfläche zu definieren, spiegelt sich im Grundalgorithmus vorgestellt in [23] wieder:

1. Berechne das zugehörige Voronoi Diagramm zur gegebenen Punktemenge.
2. Berechne für jeden Eingangspunkt die jeweiligen Pole.
3. Bestimme das Power Diagramm der Pole.
4. Kennzeichne jeden Pol entweder als innerhalb oder außerhalb.
5. Gib die Power Diagramm Flächen an, welche die Zellen außerhalb und innerhalb trennen. Diese Menge an Flächen ist die Power Kruste.

Die mediale Achse selbst wird angenähert durch das Verbinden der Mittelpunkte der inneren Pole [23]. Ausschließlich für eine Rekonstruktion der Oberfläche wird diese Struktur nicht zwingend erfordert.

Im Folgenden werden Konzepte der medialen Achsen-Transformation, wie auch der Definition von Polen und des Power Diagramms erläutert. Somit wird in diesem Zusammenhang der Ablauf des Algorithmus von der Punktewolke bis zur Power Kruste, also dem entstehenden Polygonnetz, geklärt.

9.1 Mediale Achsen Transformation

Die Abbildung 9.1 zeigt ein zweidimensionales Beispiel einer medialen Achsen-Transformation. In der Grafik ist stellvertretend für die ganze Schar der maximal leeren Kreise, ein Kandidat visualisiert.

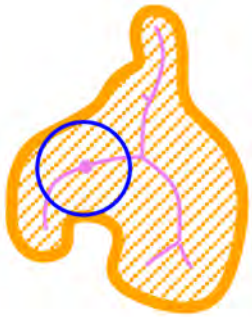

Abb. 9.1: Oberfläche und mediale Achse [23]

Die eigentliche mediale Achse wird durch die Mittelpunkte einer unendlichen Menge an Kreisen definiert. Diese exakte Skelett-Form ist laut [23] nicht besonders robust sobald die Grenze des Objektes, also die Oberfläche auch nur leicht verändert wird. Die Mithilfe der medialen Achsen-Transformation angenäherte mediale Achse wird jedoch als ausreichend und stabiler angesehen. Diese wird durch eine endliche Menge an Kreisen bzw. Kugeln im Dreidimensionalen erhalten.

Um nun die MAT genauer zu erläutern, wird in Abbildung 9.2 eine zweidimensionale Punktemenge gezeigt, welche die ursprüngliche Oberfläche S repräsentieren sollte. Das zugehörige Voronoi-Diagramm ist ebenfalls dargestellt. In der Implementierung von Amenta et al. [23] wird in der Praxis das gesamte Volumen des Raumes auf ein 5-faches der minimalen Bounding-Box des Objektes beschränkt. Somit erhält das Voronoi Diagramm Grenzen in jede Richtung.

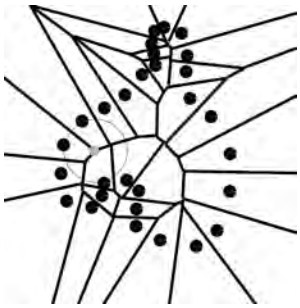

Abb. 9.2: Voronoi Diagramm und Voronoi-Ball [23]

Die MAT von S ist grundsätzlich die Menge aller maximal leeren Kugeln. Eine maximal leere Kugel ist definiert, dass sie einerseits keine Messwerte von S enthält, wie auch nicht komplett innerhalb einer Kugel mit diesen Eigenschaften enthalten ist. In Abbildung 9.2 ist ein solcher Kreis, welcher im Zweidimensionalen 3 Messwerte berührt zu erkennen. Im Dreidimensionalen schneidet eine Kugel 4 Messpunkte von S.

In der Umsetzung der Annäherung der medialen Achse werden für die Konstruktion der maximal leeren Bälle die Knotenpunkte des Voronoi-Diagramms verwendet. Somit entsteht eine endliche Schar an Voronoi-Kugeln. Diese Untermenge an Kugeln liefert im dreidimensionalen Raum noch keine brauchbare Annäherung der medialen Achse, da einige Voronoi-Knoten nicht zwingend nahe dieser sind [23]. Die Lösung hierzu ist die Definition von Polen außerhalb, wie auch innerhalb des Objektes.

9.2 Approximation der MAT mithilfe von Polarkugeln

Die Voronoi-Bälle bzw. die später, definierten Pole nähern die mediale Achse aufgrund der Form ihrer Voronoi-Flächen im Falle einer hinreichenden Dichte der Messwerte an. In Abbildung 9.3 sind die Eigenschaften bzw. Ähnlichkeiten der einzelnen Polygone zu erkennen.

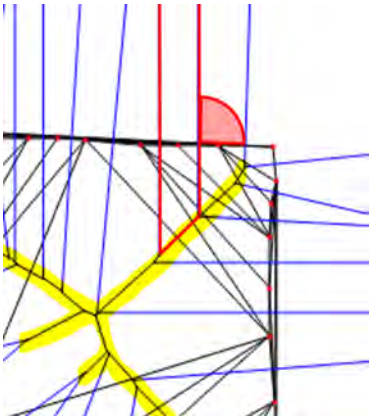

Abb. 9.3: MAT: Voronoi-Polygone und Winkel hinsichtlich Oberfläche

Alle Voronoi-Flächen in Abbildung 9.3 haben die Eigenschaft, dass sie lang und dünn sind. Die Kanten, welche zwischen den hinreichend, dichten Messwerten entstehen, stehen nahezu im Lot zur ursprünglichen Oberfläche. In Richtung der Innenseite des Objektes sind die Flächen nahe der medialen Achse beschränkt, da diese Kante oder dieser Voronoi-Knotenpunkt durch die Messwerte der gegenüberliegenden Seite definiert wird. Aus diesem Grund wird ersichtlich, dass die Voronoi-Knotenpunkte eine gute endliche Annäherung der medialen Achse sind, da sie über die Mittelsenkrechten benachbarter Messwerte definiert sind. Diese Theorie wird ebenfalls in [23] erläutert.

Jeder Voronoi-Knoten x des Voronoi-Diagramms ist somit potentiell ein Pol, welcher für die Annäherung der medialen Achse verwendet wird. Zur Bestimmung wird für jeden Messwert $s \in S$ der Voronoi-Knoten x_i außerhalb und innerhalb gewählt, welcher am weitesten entfernt ist. Jeder Punkt $s \in S$ liefert somit 2 Pole für die Annäherung der medialen Achse. Die Bestimmung, ob ein Pol als außerhalb oder innerhalb gekennzeichnet wird, findet erst in einem weiteren Schritt statt. Für die Suche nach den beiden Polen ist dies zunächst noch nicht wichtig.

Für jeden Messwert $s \in S$ wird der weitest entfernte Voronoi-Knoten x_i als p_1 gekennzeichnet. Somit wird der erste Pol pro Messpunkt einfach gefunden. Der zweite Pol p_2 ist der weitest entfernteste Voronoi-Knoten x_i, für den gilt, dass das Skalarprodukt der Vektoren $\overrightarrow{s, p_1}$ und $\overrightarrow{s, x_i}$ negativ ist. Somit wird gewährleistet, dass beide Pole auf einer anderen Seite von S liegen [23]. Welcher der beiden Pole p_1 und p_2 nun als innerhalb oder außerhalb als Kennzeichnung bekommt, wird im nächsten Schritt entschieden.

9.3 Power-Distanz

Um die Distanzen zwischen einem Messwert und einer Voronoi-Kugel aufzustellen, berechnet man das Power-Diagramm. Dieses Power-Diagramm ist anders formuliert ein gewichtetes Voronoi-Diagramm. Die Idee dahinter ist, dass für die Definition des Power-Diagramms kein euklidischer Abstand verwendet wird, sondern die einzelnen Kugeln als mit ihrem Radius gewichteten Punkte definiert werden. „Die Power-Distanz zwischen einem gewöhnlich, ungewichteten Punkt $x \in \mathbb{R}^3$ und dem Ball $B_{c,p}$ ist “ [23] wie folgt definiert:

$$d_{pow}(x, B_{c,p}) = d^2(c, x) - p^2 \tag{9.1}$$

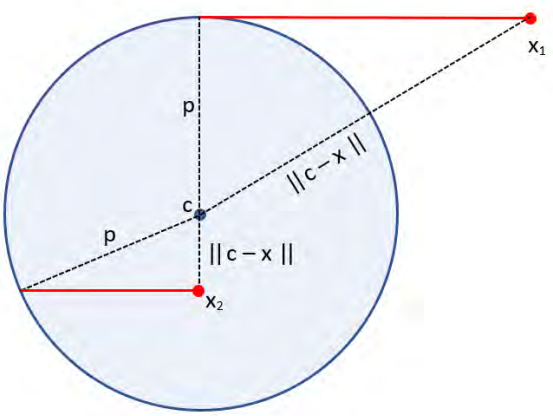

Abb. 9.4: Power-Distanz

Die Variable c bezieht sich auf das Zentrum des Balles, also den aktuell betrachteten Punkt hinsichtlich des Voronoi-Diagramms. Der gewöhnliche euklidische Abstand wird durch d angegeben. Die Gewichtung wird mithilfe des quadrierten Radius erreicht. Befindet sich nun ein Messwert x innerhalb einer Kugel, so ist die Power-Distanz negativ. Ebenso ist diese positiv, falls sich der Punkt x außerhalb befindet. In Abbildung 9.4 ist dieser Zusammenhang veranschaulicht. Die rot markierte Strecke ist jeweils die Wurzel der Power-Distanz. Schließlich besteht das Power-Diagramm aus allen gekennzeichneten Polen, die mit deren Radius, als Gewichtung, aus dem bestehenden Voronoi-Diagramm definiert werden.

9.4 Innere und äußere Pole bestimmen

Schließlich ist das Power-Diagramm aufgestellt und die beiden Pole pro Messwert wurden bestimmt. Als nächsten muss man somit feststellen, ob eine Polarkugel als außen oder innen gekennzeichnet werden kann. In [23] wird hierfür ein naiver Ansatz und ein eher praxisnäherer Ansatz vorgestellt. Wichtig für eine Definition ist die Überlegung, wie sich benachbarte Pole verhalten, wenn sie beide innerhalb, außerhalb oder jeweils eine unterschiedliche Kennzeichnung haben. Innere Polarkugeln befinden sich hauptsächlich im inneren der Oberfläche. Genauso sind

äußere Pole hauptsächlich im Raum außerhalb der Oberfläche. Aus diesem Grund wurde Beobachtet, dass sich Polarkugeln, welche eine gleiche Kennzeichnung benötigen tief schneiden und Pole unterschiedlicher Kennzeichnung nur leicht, wenn überhaupt, nahe der Oberfläche eine Interaktion aufweisen. Somit kann man behaupten, dass innere und äußere Polarkugeln einen kleinen Interaktionswinkel aufweisen. Dies wird veranschaulicht in Abbildung 9.5.

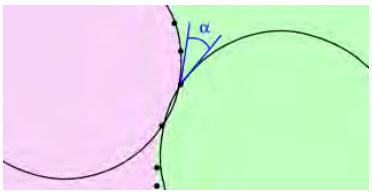

Abb. 9.5: Interaktion innere und äußere Polarkugeln [23]

Der naive Ansatz wählt zunächst die äußersten Pole im Raum und kennzeichnet diese als *außen*. Anschließend iteriert er über die Nachbarschaft der bereits gekennzeichneten Pole und bestimmt anhand der Interaktionen der Pole, ob diese als *außen* oder *innen* gekennzeichnet werden.

In der Praxis kann jedoch nicht immer davon ausgegangen werden, dass die benötigte Datendichte für die Repräsentation der Oberfläche in jedem Bereich zufrieden stellend genug gegeben ist. Deshalb werden zunächst nur Pole gekennzeichnet, bei welchen man mit einer gewissen Sicherheit diese Entscheidung rechtfertigen kann. Hierfür stellen Amenta et. al [23] einen Wahrscheinlichkeitswert vor, welcher priorisiert, wie sicher eine Entscheidung ist. Anhand dieser Wahrscheinlichkeit werden die Pole in einer Verarbeitungspipeline sortiert. Die sichersten Entscheidungen liegen oben auf diesem Stapel und werden als nächstes gekennzeichnet. Nach jedem neuen Schritt wird die Pipeline neu angeordnet und die Priorisierung der benachbarten Pole bereits gekennzeichneter Polarkugeln werden aktualisiert. Dies basiert ebenfalls auf deren gemeinsamen Winkel anhand deren Interaktion. Insgesamt wird somit gewährleistet, dass unsichere Entscheidungen möglichst spät getroffen werden. Die genaue Beschreibung dieser Heuristik ist in [23] einzusehen.

9.5 Power Kruste

Die Power Kruste ist schließlich das gewünschte Ergebnis des Algorithmus. Diese wird erhalten, indem man die Schnittflächen von äußeren und inneren Polarkugeln berechnet. Wie bereits erwähnt schneiden sich diese, wenn überhaupt, nur sehr leicht. Die entstehenden Flächen sind Polygone mit einer beliebigen Anzahl an Eckpunkten. Somit ist das Ergebnis ein Polygonnetz und keine Triangulierung, wie bei den zuvor vorgestellten Algorithmen. In [23] wird ebenfalls darauf hingewiesen, dass die Power Kruste die Oberfläche nicht nur approximiert, sondern auch die Datenpunkte interpoliert. Dies liegt daran, dass die Radien der Polarkugeln anhand der Messwerte begrenzt sind, sodass diese vollständig leer sein können. Die Messwerte liegen nahe dem Rand der einzelnen Polarkugeln, wodurch der Schnitt zweier Kugeln die Datensätze interpoliert.

Schließlich könnte ebenso die mediale Achse des Objektes berechnet werden, sobald alle inneren Polarkugeln bekannt sind. Für eine reine Rekonstruktion der Oberfläche wird dieser Schritt nicht mehr benötigt. Wie bereits erwähnt, wird die mediale Achse anhand der Mittelpunkte der inneren Polarkugeln angenähert.

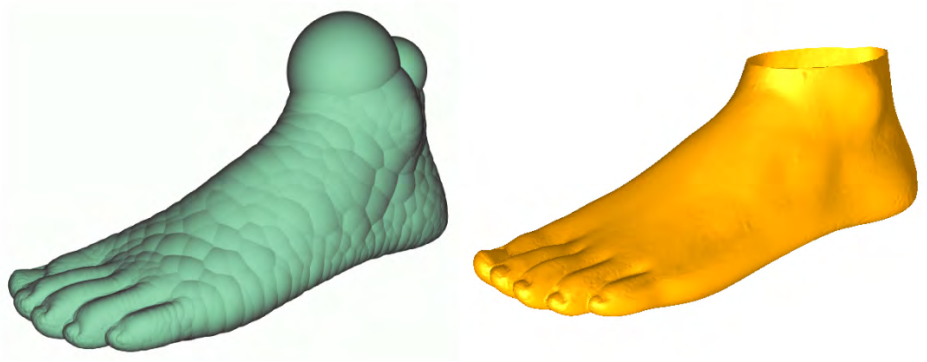

Abb. 9.6: Polarkugeln und daraus resultierendes Oberflächennetz [23]

In Abbildung 9.6 ist die Schar der Polarkugeln und das resultierende Polygonnetz veranschaulicht. Bei genauen Betrachten fällt auf, dass das Ergebnis vorherigen Aussagen widerspricht. Das Oberflächennetz ist nicht geschlossen. Dies liegt daran, dass Amenta et. al [23] ebenfalls Erweiterungen vorstellen, anhand dessen

durch die Bildung von sehr großen Polarkugeln festgestellt werden kann, ob in einer Region möglicherweise Messwerte fehlen und somit in diesen Regionen keine Flächen gebildet werden. In diesen Bereichen wird ein Loch entstehen, falls diese Erweiterung gewünscht ist.

Die Anwendung des Power Crust Algorithmus ist ebenfalls in meinen Versuchen im Kapitel Anwendungen & Ergebnisse zu betrachten.

Hinweis

Alle theoretischen Ausformulierungen des Algorithmus beruhen auf den von Nina Amenta, Sunghee Choi und Ravi Krishna Kolluri [23].

10 Anwendungen & Ergebnisse

Zunächst werden im Unterpunkt Versuchsobjekte die Herkunft der verwendeten Messwerte als Input erläutert. Anschließend werden die erzeugten Oberflächennetze hinsichtlich ihrer Eigenschaften vorgestellt.

10.1 Versuchsobjekte

Abb. 10.1: Stanford Dragon

Den verschiedenen Oberflächen Rekonstruktionen liegen verschiedene Quellen von Eingangsdaten zu Grunde. Um die Eigenschaften der Algorithmen vergleichen zu können wurden der ‘Stanford Bunny’ und der ‘Stanford Dragon’ aus dem ‘Stanford 3D Scann-Verzeichnis’ [28] verwendet. Diese sind für wissenschaftliche Arbeiten frei zur Verfügung gestellt und seit vielen Jahren eine bekannte Quelle Rund um Oberflächen Rekonstruktionen. Mithilfe dieser Modelle arbeiten viele der zuvor präsentierten Algorithmen. Im Verzeichnis selbst ist die Aussage getroffen worden: “Zu den Dingen, die die Leute mit diesen Modellen gemacht

haben, gehören Simplification, Multi-Resolution-Darstellung, gekrümmte Oberflächenanpassung, Komprimierung, Textur-Mapping, Modellierung, Deformation, Animation, physikalisch-basierte Simulation, Textur-Synthese und Rendering." [28]. Somit sind diese Modelle eine gute Wahl im Bereich Visualisierung und Oberflächen Rekonstruktion.

Abb. 10.2: Stanford Bunny

Die beiden Beispiele stellen insgesamt ein in sich geschlossenes Objekt dar. Speziell die detailreiche und hochfrequente Information auf der Oberfläche der Modelle ist für spätere Vergleiche wichtig. Die Objekte wurden zur Verwendung in verschiedenen Auflösungen abgetastet. Dies führt zu einer unterschiedlichen Anzahl an Eingangspunkten der Rekonstruktionen. Beispielsweise sieht man in Abbildung 10.3 die Ergebnisse dieser Vorverarbeitung.

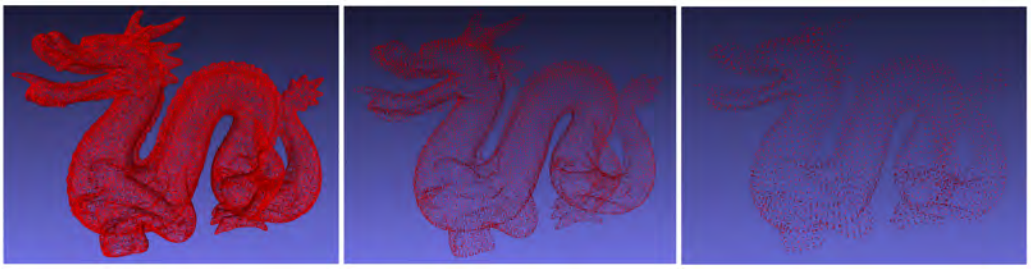

Abb. 10.3: Stanford Dragon - 100250, 22998, 5205 Messpunkte

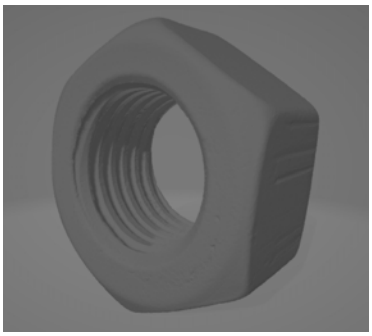

Abb. 10.4: Metallmutter

Weitere freie Modelle zum Testen der Implementierungen und theoretischen Algorithmen wurden von Artec3D [2] bezogen. Ein Beispiel ist die Metallmutter in Abbildung 10.4.

Schließlich wollte ich nicht nur Versuche mit bestehenden Messpunkten durchführen. Deswegen habe ich versucht mithilfe von Photogrammetrie eigene Punktewolken zu erstellen. Bereits zuvor war klar, dass diese Messproben eine verrauschte Punktewolke ergeben würden. Dieser Effekt ist jedoch keinesfalls negativ. In meinen Versuchen hatte ich ebenfalls das Ziel den Algorithmen fehlerhafte, unvollkommene Proben als Eingang bereit zu stellen um die Auswirkungen dieser zu bewerten.

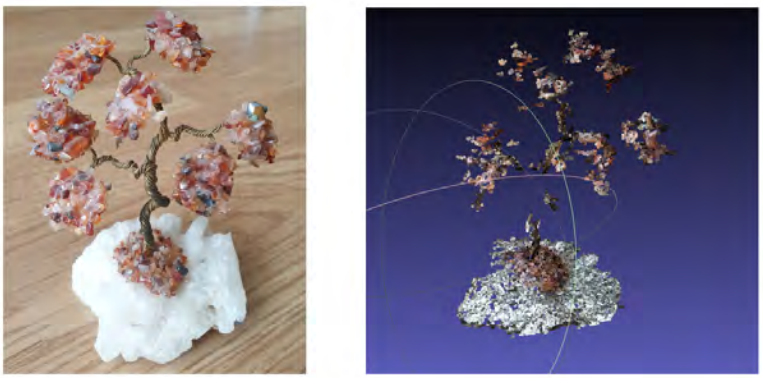

Abb. 10.5: Photogrammetrie Punktewolke

Alle Informationen der explizit verwendeten Datensätze und vorgestellten Ergebnisse werden im Kapitel Dateninformation bereitgestellt.

10.2 Implementierungen & Default Parameter

Die Parameter der einzelnen Anwendungen wurden generell den Empfehlungen deren wissenschaftlichen Arbeiten angepasst. Solange nicht extra auf eine Parameteränderung hingewiesen wird, werden die im folgenden erwähnten Parameter genutzt. Die Anwendungen wurden größtenteils von mir in C++ mithilfe der Computational Geometry Algorithms Library [30], kurz CGAL, erstellt. Ausnahme ist der Power Crust Algorithmus der nahezu der originalen Implementierung beschrieben in [23] gleicht.

Input

Um die verschiedenen Input-Daten einlesen zu können, habe ich mir zunächst zwei Wrapper-Funktionen erstellt, welche das Einlesen von Punktewolken und Oberflächennetzen erleichtert. So konnte ich über verschiedene Parameter einfach zwischen OFF-, PCD- und PLY-Format wechseln. Ebenso erstellte ich eine Unterscheidung, ob ein Input binär geschrieben wurde oder nicht. Dies nutze ich zum Einlesen von bestehenden Punktewolken, wie auch zum Lesen von eigens erstellten Punktewolken oder Oberflächennetzen. Die Eingangspunktewolken wurden auf verschiedene Weise erhalten. Wie bereits beschrieben, wurden einerseits bestehende Punktewolken verwendet oder auch die bestehenden Knotenpunkte einer Triangulierung als Input genutzt. Für eine eigene Abtastung implementierte ich eine Lasersimulation, welche um ein definiertes Zentrum Halbgeraden mit dem bestehenden Oberflächennetz schneidet. Die Schnittpunkte ergeben eine neue Punktewolke. Zur Laufzeitoptimierung wurde eine Baumstruktur von CGAL verwendet, welche eine schnellere Berechnung der Schnittpunkte verspricht. Zusätzlich wurden zwei weitere Ansätze von mir verfolgt. Einerseits war die Überlegung, dass bei einem kleinen Winkelunterschied wahrscheinlich die selbe primitive Fläche oder eine Fläche in der Nachbarschaft dieser getroffen wird. Somit wurde beim Anpassen des Lasers ein MemoryFace gespeichert, anhand welchem bzw. dessen Nachbarschaft zunächst auf einen Schnittpunkt getestet wurden, bevor im ganzen Baum erneut gesucht wurde. Andererseits waren die einzelnen Ebenen in denen der Laser hinsichtlich X und Y Koordinate verändert

wurden unabhängig von der Veränderung der folgenden Z Koordinate. Somit nutze ich als Optimierung ebenfalls mehrere Threads und führte die Punkte im Anschluss zusammen.

Preprocessing

Vorverarbeitungsschritte, wie Punktewolkenvereinfachung oder Glättung wurden nicht durchgeführt. Für die nahezu idealen Eingangspunkte von der Stanford Universität wurden diese Schritte zwar zunächst von mir getestet, jedoch wurde die Punktewolke dadurch nicht stark verändert und ich sah darin keinen Nutzen. Für schlechtere Punktewolken wollte ich diese Optimierungsschritte absichtlich nicht durchführen. Für die Punktewolke, welche durch Photogrammetrie entstanden ist wurden manuell, mithilfe von MeshLab, eine Vielzahl an falschen Messpunkten als Vorverarbeitung entfernt.

Abb. 10.6: Photogrammetrie Vorverarbeitung

Rekonstruktionen

Für die Poisson Rekonstruktion wurde zum Abschätzen der Normalen und deren Orientierung eine Nachbarschaft von 24 Punkten gewählt. Somit konnte aus einer unorientierten Punktewolke eine orientierte Punktewolke erstellt werden. Alle nicht orientier-baren Knotenpunkte wurden vor der Rekonstruktion entfernt. Für die Bestimmung eines durchschnittlichen Abstandes zwischen den Eingangspunkten wurden 6 Nachbarn verwendet. Diese Default-Parameter werden von

CGAL, in deren Dokumentation zu den Funktionen compute_average_spacing() und pca_estimate_normals() bzw. mst_orient_normals() empfohlen.

Für den Advancing Front Algorithmus wurde der Ansatz von David Cohen-Steiner und Tran Kai Frank Da [8] verwendet. Manuell sind hierbei keine Parameter zu wählen, da der Algorithmus zunächst eine Delaunay Triangulierung erstellt und sich als Startdreieck das mit dem kleinsten Radius wählt. An diesem wird die Front dann nach einer Priorisierung erweitert. Extra Grenzen für eine maximale Kantenlänge oder Ähnlichem wurden nicht definiert.

Im speziellen Fall des Scale Space Meshings wurden 4 Iterationsschritte für das Skalieren als Default-Wert festgelegt. Für die folgende Advancing Front Methode wurde die Rekonstruktion hinsichtlich einer Obergrenze der maximalen Kantenlänge von 0.5 verwendet. Auf diese beiden Parameter wird ebenfalls in einer Dokumentation von CGAL hingewiesen.

Die originale Implementierung des Power Crust Algorithmus ist in C implementiert und läuft standardmäßig unter Linux. Diese Software zugehörig zur Arbeit von Nina Amenta, Sunghee Choi and Ravi Krishna Kolluri [23] läuft unter der GNU Public License. Da es unter dieser Lizenzierung gestattet ist den bestehenden Code zu verändern, wurde für meine Zwecke eine angepasste Implementierung, welche unter Windows läuft verwendet. Viele Veränderungen am ursprünglichen Code wurden in diesem Fall nicht durchgeführt. Der Algorithmus arbeitet funktional, wie die originale Implementierung und wurde verwendet um als Resultat auch ein Polygonnetz, anstelle einer Triangulierung zu sehen.

10.3 Glättende Wirkung

Eine glättende Wirkung kann einerseits gewünscht sein, andererseits wird versucht diese zu meiden. Wie bereits beschrieben, kann eine Glättung auch durch gewisse Vorverarbeitungsschritte erhalten werden. In unserem Fall interessieren wir uns für die Rekonstruktionen, die eine solche Eigenschaft besitzen. Speziell implizite Verfahren weisen einen gewissen Grad an glättenden Eigenschaften auf. In Abbildung 10.7 kann neben dem Original und den beiden expliziten Verfahren Scale Space und Advancing Front betrachtet werden, wie sich die glättende Wirkung des impliziten Verfahrens auf die Oberfläche auswirkt. Die Struktur des Felles ist weniger stark ausgeprägt, da diese hochfrequente Information verloren ging. In den beiden expliziten Verfahren ist dies nicht der Fall. Die Falten im Fell sind noch gut erkennbar.

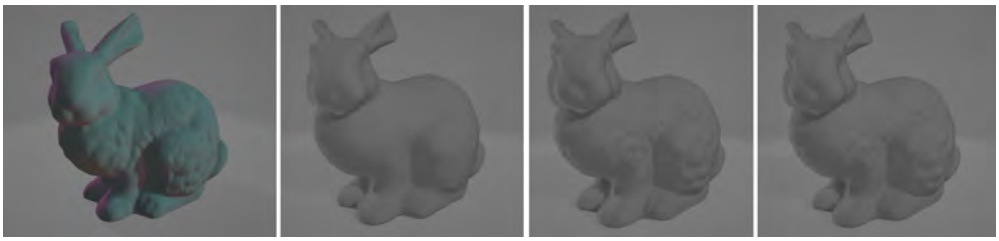

Abb. 10.7: Stanford Bunny - Original, Poisson, Scale Space, Advancing Front

Die Ergebnisse der Algorithmen in Abbildung 10.7 sind entstanden durch eine abgetastete Punktewolke aus dem Original. Für diese Versuche wurde eine Menge von 35947 Punkten verwendet. Die expliziten Verfahren verwendeten jeden Eingangspunkt als Knotenpunkt und haben somit ebenfalls 35947. Die Poisson Rekonstruktion reduzierte die Knotenpunkte auf ca. $1/14$ des Inputs und hat somit nur noch 2614 Knotenpunkte. Advancing Front produzierte 71872 Dreiecke. Scale Space ähnlich viele mit 71878. Die Poisson Rekonstruktion besitzt nur 5224 Flächen. Diese Reduktion spiegelt auch wieder, dass die Flächen der Triangulierung größer sind und somit weniger detailreich.

Die Menge an Eingangspunkten spielt bei diesen Vergleichen eine wichtige Rolle. In Abbildung 10.12 können als Beispiel Ergebnisse eines Advancing Front Algorithmus gesehen werden, die auf den ersten Blick eine glättende Wirkung zeigen. Hierbei ist zu beachten, dass eine zu geringe Punktmenge nicht ausreicht die

gewünschten Informationen zu repräsentieren. Die Abtastfrequenz ist hierfür zu niedrig und die hochfrequente Information geht verloren. Somit sieht man in diesen Versuchen nicht eine glättende Wirkung, sondern die Auswirkungen einer zu geringen Abtastdichte.

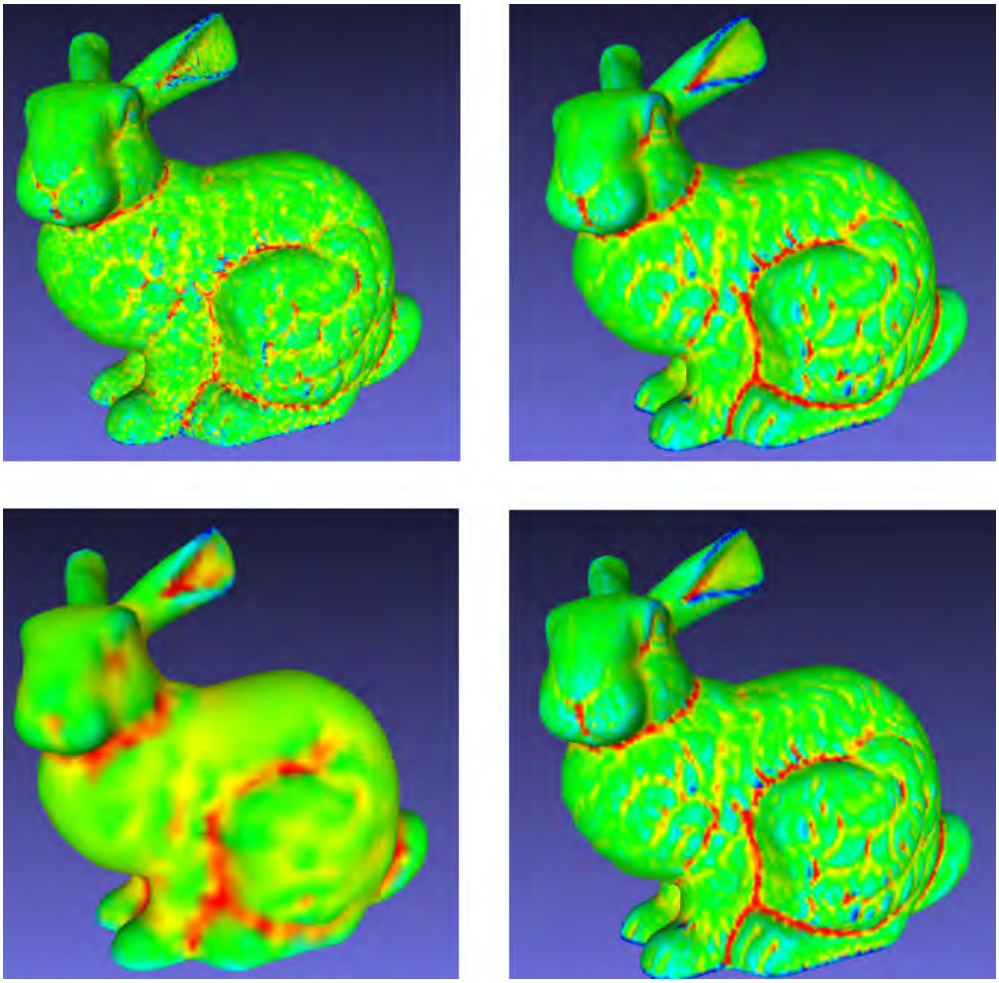

Abb. 10.8: Eingefärbte mittlere Krümmung: Power Crust, Scale Space, Poisson, Advancing Front

Eine glatte Oberfläche kann somit mehrere Gründe haben. Die glättenden Eigenschaften von impliziten Rekonstruktionen sind bei der Anwendung speziell zu beachten. Ein Vorteil der glättenden Wirkung ist, dass viele Messfehler nahe der

Oberfläche nicht zu unregelmäßigen Oberflächennetzen führen. Diese werden gegebenenfalls aufgrund der Glättung in gewisser Weise ignoriert. Wird jedoch die Anforderung gestellt, dass fast alle Eingangspunkte als Knotenpunkte genutzt werden sollten, sollte lieber ein explizites Verfahren ohne Glättung verwendet werden. Die Knotenpunkte der geglätteten Triangulierung des Poisson Verfahrens entstehen bei der Iso-Oberflächenauswertung der impliziten Funktion und haben nicht mehr direkten Bezug zu den Messwerten. Das Oberflächennetz des Scale Space Algorithmus verändert zwar zunächst die Eingangspunktewolke, projiziert diese jedoch auf die Messwerte zurück, sodass hierbei nicht von einer Glättung gesprochen werden kann.

Die Abbildung 10.8 zeigt eine Auswertung der Krümmung der Oberflächennetze der verschiedenen Verfahren. Durch die Einfärbung dieser, kann ebenfalls erkannt werden, dass bei der impliziten Rekonstruktion hochfrequente Information verloren geht. Niederfrequentere Krümmungen, wie am Hals oder dem Übergang von Körper und Bein bleibt bei allen Verfahren erhalten.

10.4 Geschlossenes Objekt & Löcher

Bestehen größere Löcher in den ursprünglichen Messwerten, kann dies zu Löchern in den erzeugten Oberflächennetzen führen. Das Stanford Bunny hat in seiner ursprünglichen Abtastung Löcher im Bereich der Unterseite. Diese kann man in der bestehenden originalen Triangulierung in Abbildung 10.9 sehen.

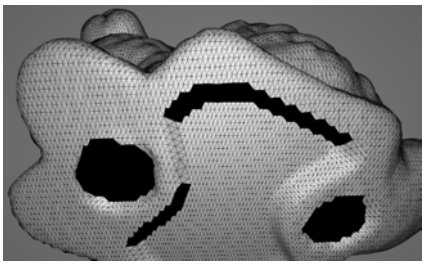

Abb. 10.9: Stanford Bunny - Originale Löcher

Implizite Verfahren, wie die Poisson Rekonstruktion, versuchen stets ein geschlossenes Objekt zu erzeugen. Die voxelbasierte Oberflächenauswertung der globalen impliziten Funktion führt zu neuen Knotenpunkten, die nicht mehr direkten Bezug zu den Messpunkten haben. Somit entstehen auch größere primitive Flächen, die im Normalfall einzelne Löcher auf der Oberfläche schließen können. Die Power Crust Methode lieferte ebenfalls ein Oberflächennetz ohne Löcher. Beide Netze sind in Abbildung 10.10 zu betrachten.

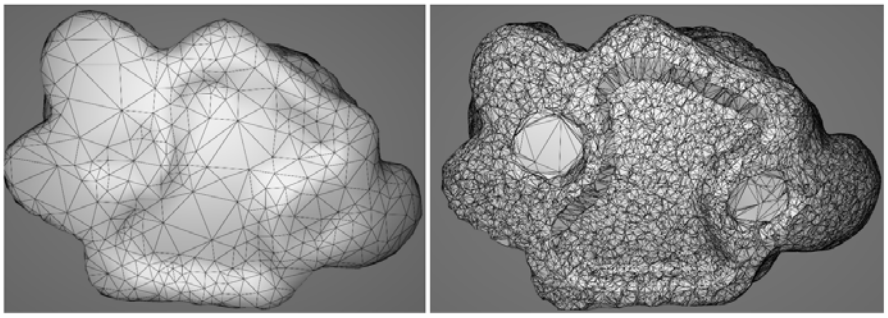

Abb. 10.10: Geschlossene Löcher: Poison, Power Crust

Bei expliziten Verfahren werden fehlende Messwerte nicht immer ersetzt. Am

einfachsten stellt man sich das anhand des Ball-Pivoting Algorithmus vor. Ein Ball mit definiertem Radius wird um die bestehende Front an Kanten gedreht. Das Loch ist gegebenenfalls so groß, dass der Pivot-Ball an der Kante keinen weiteren Messpunkt mehr trifft. Somit bleibt ein Loch im Oberflächennetz bestehen. Ist jedoch das Loch durch fehlende Eingangswerte klein, so kann es gegebenenfalls gefüllt werden, da der Ball trotzdem einen Messwert treffen kann. Zur Visualisierung hierzu in Abbildung 10.11 das Scale Space Meshing, welches einige kleinere Löcher tolerieren konnte, das größte Loch jedoch nicht.

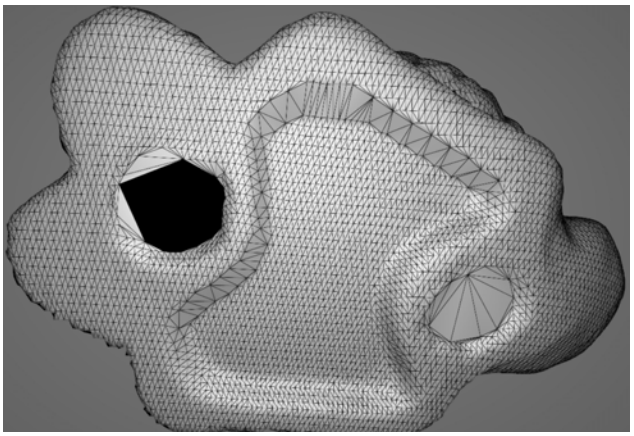

Abb. 10.11: Geschlossene und bestehende Löcher: Scale Space

Sind in den Messwerten die Löcher so angeordnet, dass sie ein Objekt in mehrere Komponenten unterteilen, werden im Falle der expliziten Verfahren mehrere nicht zusammenhängende Oberflächennetze erstellt. Sie rekonstruieren somit kein in sich geschlossenes Objekt, sondern mehrere Komponenten des Objektes im Raum. Zu Betrachten ist dieses Phänomen in Abbildung 10.14.

10.5 Abtastdichte

Die Abtastdichte spielt eine große Rolle für die Qualität des Endresultates. Für jedes Feature, das man wieder Rekonstruieren möchte, muss man sich bewusst werden, dass die Abtastdichte groß genug sein muss. Eine bekannte Regel aus der Signalverarbeitung und Informationstheorie ist in diesem Zusammenhang das Nyquist-Shannon-Abtasttheorem. Dieses besagt, dass man ein begrenztes Signal mindestens mit der doppelten Frequenz abtasten muss, sodass die Möglichkeit besteht die ursprüngliche Frequenz wieder zu rekonstruieren. Dies gilt ebenfalls für die potentiell hochfrequente Information auf der Oberfläche. Wie in Abbildung 10.12 und 10.13 zu erkennen, wird durch weniger Eingangspunkte, also einer niedrigeren Abtastdichte/-frequenz, ein weniger auflösendes Ergebnis erzeugt. Die hochfrequente Information der Schuppen auf der Haut des Drachens geht bei einer zu geringen Abtastdichte verloren.

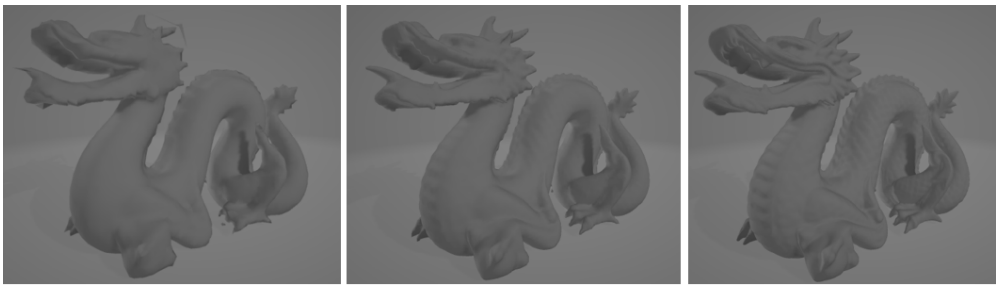

Abb. 10.12: Oberflächennetz - Input: 5205, 22998, 100250

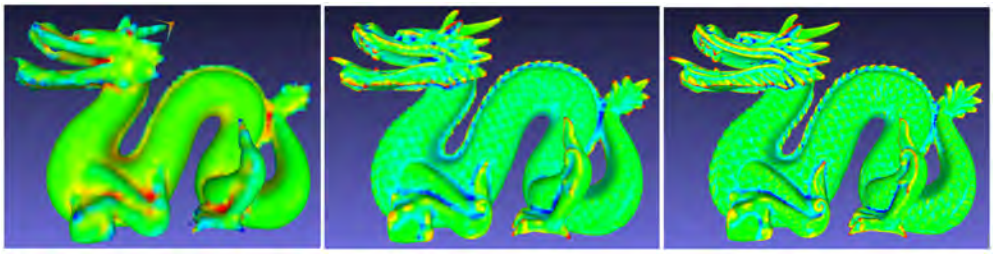

Abb. 10.13: Eingefärbte mittlere Krümmung - Input: 5205, 22998, 100250

Die Abbildungen 10.12 und 10.13 sind entstanden durch den selben Advancing Front Algorithmus mit den jeweiligen Mengen an Eingangspunkten. Wie bereits

beschrieben, kann die Nutzung einer zu niedrigen Abtastfrequenz dazu führen, dass das Endresultat geglättet wirkt, wie zum Beispiel im Falle von 5205 Messwerten. Dies ist jedoch nicht der Fall. Die glatte Oberfläche entsteht aufgrund der Missachtung des Abtasttheorems. Hat man folglich Einfluss auf die Abtastfrequenz, sollte man sich zuvor schon bewusst sein, welche Genauigkeit die Features der Oberfläche haben sollten.

Ein weiteres Problem mit der Abtastdichte kann sein, dass diese variiert. Eine ungleichmäßige Abtastdichte und vor allem dünne Bereiche mit kaum Messwerten finden wir in meinen eigens erstellten Punktewolken mithilfe von Photogrammetrie. Der Kristallbaum, zu sehen in Abbildung 10.6, weist eine hohe Abtastdichte im Bereich des Sockels auf. Eine sehr niedrige Anzahl an Messwerten befindet sich in der Komponente des Stammes und dessen angrenzenden Zweigen. Die Punkte zu den Kristallblättern sind für eine Rekonstruktion mittelstark bis gut ausgeprägt.

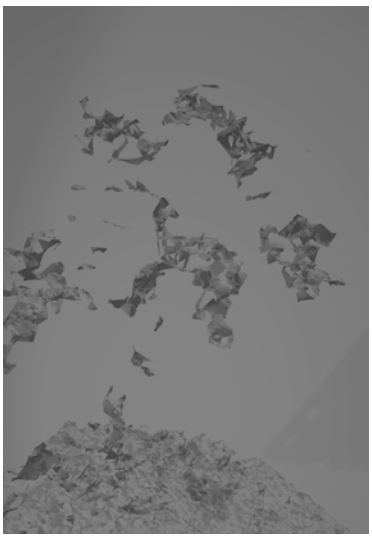

Abb. 10.14: Beispiel mangelnde Messwerte - Komponenten im Raum

Die Rekonstruktion in Abbildung 10.14 ist entstanden durch ein Scale Space Verfahren. Typisch für eine explizite Methode sieht man, dass die Oberfläche nicht eine Geschlossene ist, sondern mehrere Komponenten im Raum gebildet wurden. Der Sockel, wie auch die Kristallkronen sind als unabhängige Oberflächen von

Objekten im Raum zu erkennen. Der Stamm wurde zum Teil ebenfalls rekonstruiert, jedoch aufgrund der fehlenden Messwerte nur sehr bedingt.

Bei einem direkten Advancing Front Verfahren auf diese Punktewolke, wurde im Gegensatz zum vorherigen Algorithmus ein größerer Teil der Zweige rekonstruiert. Negativ ist beim direkten Rekonstruieren aufgefallen, dass auch viele unerwünschte, große Dreiecke zwischen den Kristallkronen eingefügt wurden, die beim Scale Space nicht vorhanden sind. Hier sieht man den Vorteil der iterativen Schritte, welche die Punktewolke anhand ihrer und der Normalen der Nachbarschaft glättet, bevor das Oberflächennetz erstellt wird. Ein Beispiel dieser unerwünschten Dreiecke sieht man in Abbildung 10.15.

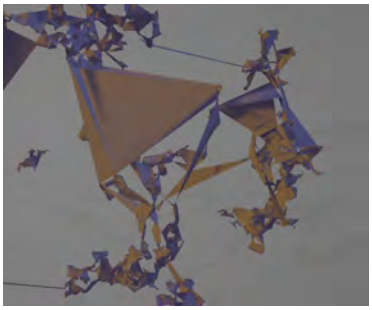

Abb. 10.15: Beispiel mangelnde Messwerte - unerwünschte Dreiecke

Die Poisson Oberflächen Rekonstruktion hat im Zusammenhang mit dieser Punktewolke nicht funktioniert. Zunächst versucht die implizite Funktion eine geschlossene Oberfläche zu erstellen. Die fehlenden Messwerte im Bereich Stamm, was ein sehr dünner Bereich des Objektes ist, würden eine höhere bzw. ausreichende Dichte an Messpunkten benötigen. Mehrere Ausreißer in den Datensätzen würden das Poisson Verfahren ebenfalls erschweren. Hingegen dessen wurde versucht diese zum groß-teil, wie bereits beschrieben, manuell zu entfernen. Die Orientierung der Eingangspunkte ist ebenfalls äußerst wichtig für die spätere Rekonstruktion. Diese Abschätzung und Orientierung der Normalen war speziell in diesem Fall äußerst fehleranfällig. Im Großen und Ganzen war die Punktewolke für diese Rekonstruktion nicht geeignet und lieferte kein erwünschtes Ergebnis.

10.6 Parameter Anpassung Nachbarschaft

Die Parameter für die einzelnen Algorithmen spielen ebenfalls eine wichtige Rolle. Dies kann aus Performanz-, wie auch Qualitätsgründen behauptet werden. Speziell die Parameter zur Definition einer Nachbarschaft können ein Ergebnis stark verändern. In vielen der aufgeführten Algorithmen wird eine Nachbarschaft aufgrund eines Radius bzw. einer Anzahl an Nachbarn definiert. Ist dieser Parameter zu klein gehen wichtige lokale Informationen verloren. Eine Regressionsebene zum Beispiel kann sich somit einerseits stark verändern im Gegensatz zur größeren Nachbarschaft oder andererseits aufgrund mangelnder Nachbarn garnicht erst gebildet werden. Das Resultat verliert somit an Qualität.

Wird die Nachbarschaft entgegen der Empfehlungen zu groß gewählt, kann dies zu enormen Performanzproblemen führen. Jeder Schritt in der Verarbeitung braucht potentiell mehr Zeit. Zudem muss man sich auch bewusst sein, dass eine Abschätzung einer Regressionsebene oder eines Normalenvektors aufgrund einer Nachbarschaft lokale Eigenschaften besitzt. Werden hierfür zu viele Nachbarn in Betracht gezogen, ist das Resultat eher nicht mehr so lokal und könnte somit verändert sein. Insgesamt würde ich dies gegebenenfalls als Qualitäts-, wie auch Performanzproblem einstufen.

Eine richtige Wahl gibt es allgemein in diesem Zusammenhang nicht. Die Empfehlungen der einzelnen Veröffentlichungen oder auch der Bibliotheken, wie zum Beispiel CGAL, liefern einen Anhaltspunkt, an dem man sich orientieren sollte. Die einzelnen bekannten Definitionen der Parameter haben aufgrund mehrerer Tests in verschiedenen Arbeiten ihre Berechtigung.

10.7 Primitive Flächen

Die Größe der einzelnen Polygone des Oberflächennetzes bzw. die Größe der Dreiecke einer Triangulierung sind natürlich an die Messdichte der Eingangspunkte gebunden. Die Abtastdichte des Inputs bestimmt hierbei in welcher Skalierung die Eigenschaften der Oberfläche dargestellt werden können. Bei jeder Rekonstruktion muss man sich bewusst sein, dass man nicht die spezielle Oberfläche rekonstruiert, sondern eine Oberfläche, die anhand gegebener Messwerte repräsentiert wird. Nutzt man die gleiche Menge an Messpunkten für die verschiedenen Rekonstruktionen, erhält man verschiedene Oberflächennetze als Resultat. Die einzelnen Algorithmen weisen hier aufgrund ihrer unterschiedlichen Strategie andersartige Knotenpunkte, wie auch Flächen auf. Die Anzahl und Größe dieser unterscheidet sich stark zwischen expliziten und impliziten Rekonstruktionen.

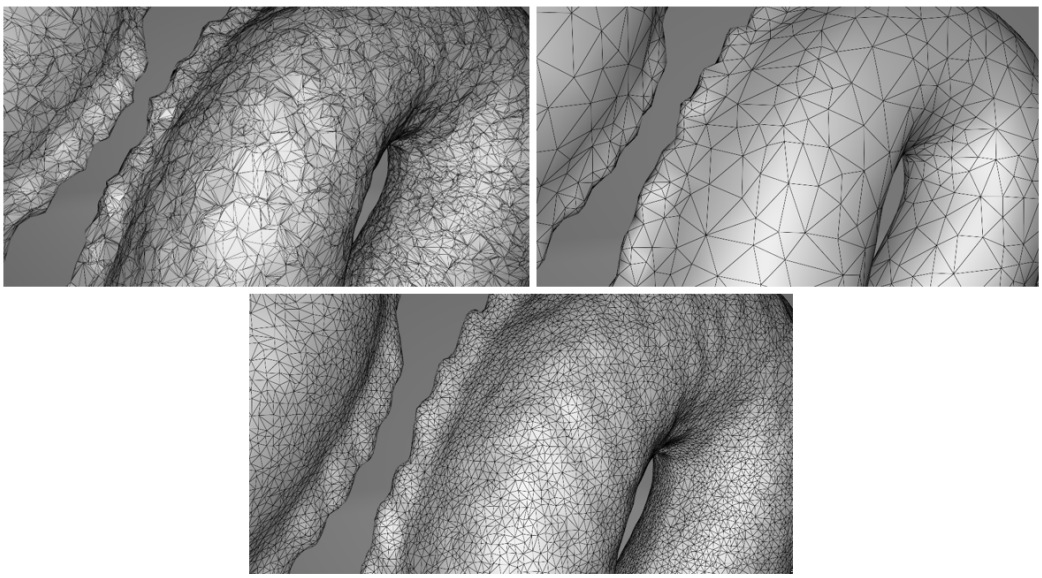

Abb. 10.16: Primitive Flächen: Power Crust, Poisson, Advancing Front

Explizite Verfahren, wie ein Advancing Front Algorithmus oder ein Scale Space Meshing, nutzen nahezu alle Eingangspunkte als Knotenpunkte. Aus diesem Grund weisen sie auch eine hohe Anzahl an Knotenpunkten auf, was ebenfalls zu einer großen Menge an Flächen führt. Wie in Abbildung 10.16 zu erkennen, weist das

Oberflächennetz des Advancing Front Verfahrens die größte Anzahl an Dreiecken auf. Somit sind die einzelnen primitiven Flächen auch relativ klein.

Im Gegensatz dazu wird bei einer impliziten Rekonstruktion die Iso-Oberfläche ausgewertet. Wird dies beispielsweise mit einem Octree realisiert, kann der Abstand der Knotenpunkte schwanken und somit auch die Größe der einzelnen Flächen. Die Knotenpunkte dieser Netze haben nicht mehr direkt Bezug zu den Eingangswerten. Allgemein weisen sie eine niedrige Anzahl an Flächen, wie auch Punkten auf. Das geglättete Objekt kann somit auch mit weniger Speicherbedarf geschrieben werden. Für die primitiven Flächen bedeutet dies, dass sie im Vergleich zu den anderen Varianten relativ groß sein können.

Als dritte Methode wird in Abbildung 10.16 die Power Crust Rekonstruktion gezeigt. Die primitiven Flächen wirken hier zunächst etwas unsortiert. Im Kontrast zur Delaunay-Eigenschaft, dass schmale Dreiecke vermieden werden sollten, sind hier eine Vielzahl von ihnen vorhanden. Die Umkreisbedingung wurde in diesem Fall natürlich nicht angewandt, doch der Gegensatz ist klar erkennbar. Dieses Phänomen ist der Nachbearbeitung des Oberflächennetzes geschuldet. In Abbildung 10.16 sind nur Triangulierungen zum Vergleich der einzelnen primitiven Dreiecken aufgeführt. Das eigentliche Resultat der Power Crust Variante ist ein Polygonnetz, welches in Abbildung 10.17 zu sehen ist. Für einen direkten Vergleich zu den anderen Rekonstruktionen, wurden die einzelnen Polygone durch Hinzufügen von weiteren Kanten zu einer Triangulierung umgewandelt.

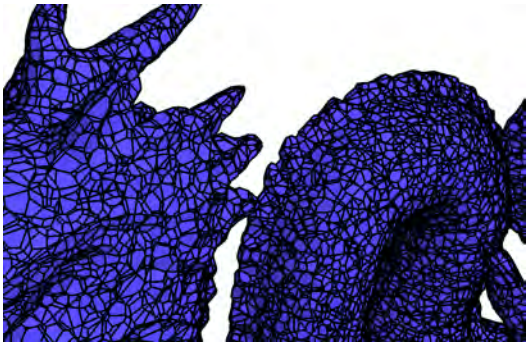

Abb. 10.17: Primitive Flächen: Polygonnetz

Betrachtet man die Anzahl an Polygonen der Power Kruste ist bereits diese Zahl

sehr hoch. Dies ist ebenfalls im Kapitel Dateninformation zu sehen.

10.8 Verrauschte Eingangsdaten

Um die Toleranz der Algorithmen hinsichtlich Rauschen zu überprüfen, habe ich die nahe zu idealen Testdaten der Stanford Universität verwendet. Zur Visualisierung werden im Folgenden die Ergebnisse des Stanford Bunny vorgestellt. Zunächst habe ich die ideale Punktewolke eingelesen. Im Anschluss habe ich mithilfe eines Pseudo-Random-Number-Generators ein zufälliges Rauschen den Daten hinzugefügt.

10.8.1 Zufälliges Verrauschen im Raum

Dies wurde zunächst umgesetzt, indem für jeden Messpunkt der Wert hinsichtlich einer Koordinatenachse um einen bestimmten Prozentsatz erhöht bzw. erniedrigt wurde. In Abbildung 10.18 sind die unterschiedlich stark verrauschten Punktewolken zu begutachten. Die erste Wolke wurde erhalten durch einen zufälligen Faktor zwischen $0,975$ - $1,025$ pro Koordinatenabschnitt der Messpunkte. Die beiden anderen haben einen Faktor zwischen $0,95$ - $1,05$ und $0,9$ - $1,1$.
Die Ergebnisse des ersten Rauschens sind gut geeignet um die Toleranz der einzelnen Algorithmen vorzustellen. Die Poisson Oberflächen Rekonstruktion präsentiert in diesen Versuchen am Besten die ursprüngliche Form des Hasen. Dies liegt daran, da versucht wird eine globale Lösung der Indikatorfunktion zu finden. Diese wird im Anschluss mithilfe einer Iso-Oberflächen-Auswertung zu einer Triangulierung. Diese Repräsentation ist eine geglättete Version der Oberfläche. Die geglättete Oberfläche kann somit mit einem gewissen Grad an Rauschen umgehen. Alle anderen Algorithmen hatten hier ihre Schwierigkeiten. Das Scale Space und das Power Crust Ergebnis zeigen eine nahezu geschlossene Oberfläche. Die hohe Frequenz des Rauschens ist jedoch in beiden sehr gut erkennbar. Im Power Crust Versuch ist ebenfalls eine Oberfläche zwischen den Ohren und dem Körper entstanden. Diese Oberflächenverbindung ist unerwünscht. Das Advancing Front Resultat spiegelt bereits bei diesem geringen Rauschen wieder, dass Löcher in der Oberfläche entstehen, da durch das zufällige Verschieben der Messpunkte im Raum Löcher in den Messwerten entstehen konnten.

Im Falle eines zufälligen Rauschfaktors zwischen $0,95$ - $1,05$ sind die Ergeb-

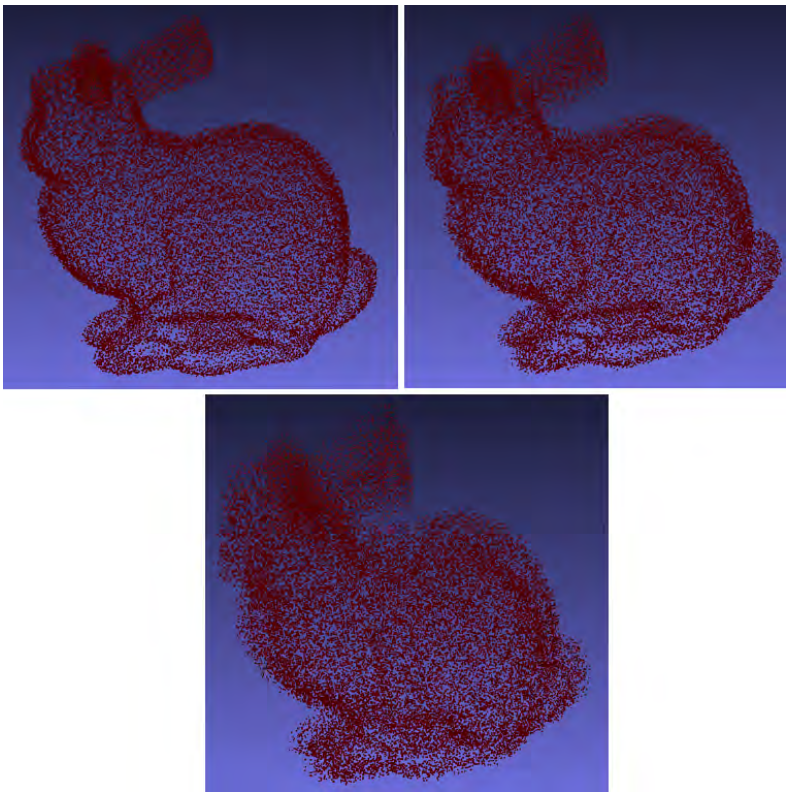

Abb. 10.18: Verrauschte Punktewolke des Stanford Bunny

nisse in Abbildung 10.20 entstanden. Diese weisen die selben Resultate wie beim vorherigen Versuch auf. Die implizite Rekonstruktion ist noch einigermaßen gut erhalten und beinhaltet relativ wenig hochfrequente Information des Rauschens. Die expliziten Verfahren repräsentieren einen hohen Anteil des Rauschens und beinhalten eine Vielzahl an Löchern. Der Power Crust Algorithmus weist vermehrt Oberflächenverbindungen auf, die ursprünglich nicht vorhanden waren. Der dritte Versuch mit verrauschtem Input weist ebenfalls die selben Eigenschaften auf. Die Auswirkungen sind im Vergleich zu den Vorgängern nur extremer.

10.8.2 Verrauschen in Richtung der Normalenvektoren

Für einen weiteren Versuch hinsichtlich Rauschen, habe ich zunächst anhand einer Regressionsebene einer Nachbarschaft von 24 Punkten die Normalenvektoren

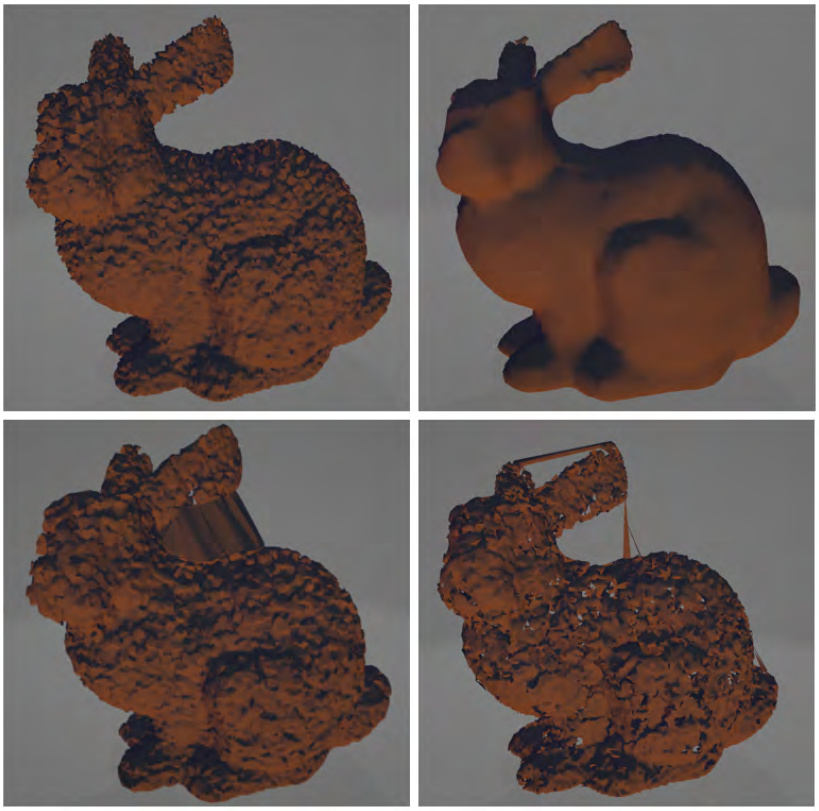

Abb. 10.19: Rauschen 1: Scale Space, Poisson, Power Crust, Advancing Front

zu den jeweiligen Eingangspunkten abgeschätzt. Das Rauschen wurde somit in eine definierte Richtung durchgeführt. In meinem Versuch generierte ich einen zufälligen Faktor innerhalb eines bestimmten Intervalls, mithilfe dessen jeder Messwert anhand seines Richtungsvektors im Raum verschoben wurde. Dies schien mir etwas realitätsnaher als das zufällige Verrauschen der Messwerte im Raum. Meine Überlegung hierbei war, dass bei einem realen Scann mit einem Laser der Abstand vom Ursprung bis zur auftreffenden Oberfläche gemessen wird. Somit ist in der Realität anhand eines definierten Winkels des Lasers im Raum eher die Abstandsmessung vom Rauschen betroffen. Die hochfrequente Information aufgrund von einem Rauschfaktor würde sich, anhand dieser Überlegung, hauptsächlich in die Richtung des Laserstrahles auswirken. Da dieser zum Zeitpunkt einer bestehenden Punktewolke nicht mehr nachvollziehbar ist, schien es mir

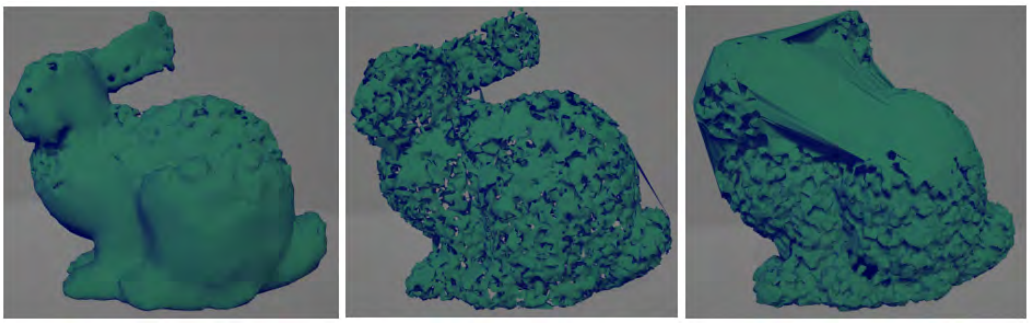

Abb. 10.20: Rauschen 2: Poisson, Advancing Front, Power Crust

plausibel einen Rauschfaktor anhand der Normalenvektoren der Eingangspunkte zu generieren. Dies führte zu folgenden verrauschten Eingangsdaten.

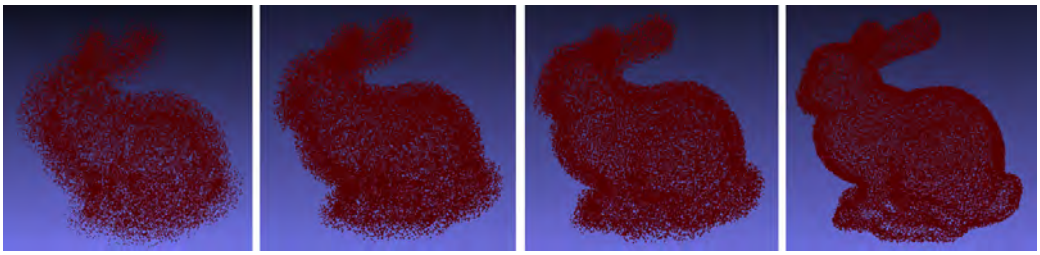

Abb. 10.21: Verrauschte Eingangsdaten mithilfe der Normalenvektoren

Wie in Abbildung 10.21 zu erkennen, sind die einzelnen Punktewolken unterschiedlich stark verrauscht, jedoch sind im Vergleich zu den Punktewolken in Abbildung 10.18, die einzelnen Komponenten noch besser zu erkennen und es sind keine größeren Löcher in der Oberfläche entstanden. Anhand dieser Punktewolken erwartete ich mir, dass ein glättender Algorithmus sehr gut mit dieser Art an Rauschen umgehen kann und die zusätzlich eingefügte Information weitestgehend toleriert. Von den expliziten Methoden erwartete ich, dass die neue hochfrequente Information des Rauschens zu begutachten ist, jedoch nicht allzu viele Löcher, also einzelne Komponenten, entstehen. Die Ergebnisse des Advancing Front, des Scale Space und des Poisson-Oberflächen Algorithmus sind in Abbildung 10.22, 10.23 und 10.24 zu sehen.

Abb. 10.22: Verrauschte Eingangsdaten: Triangulierung - explizit vs. implizit

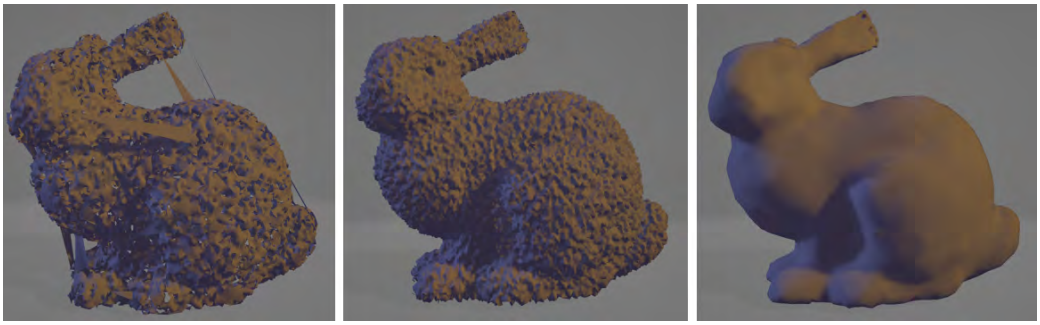

Abb. 10.23: Verrauschte Eingangsdaten: Oberfläche ohne Kanten - explizit vs. implizit

Die Abbildung 10.22 zeigt die Triangulierung der Ergebnisse. In dieser sind die Kanten der primitiven Flächen zu erkennen. Die Abbildung 10.23 zeigt die selben Ergebnisse, jedoch ohne das Hervorheben der Kanten der Triangulierung. In den rekonstruierten Oberflächen ist gut zu erkennen, dass sich meine Vermutungen bestätigt haben. Die Poisson Oberflächen Rekonstruktion liefert ein ähnliches Ergebnis, als wenn der nahezu ideale Input verarbeitet worden wäre. Dies spiegelt wieder, dass dieses Verfahren einen gewissen Anteil an Rauschen in den Eingangsdaten sehr gut tolerieren kann. Den Vorteil der Scale Space Verfahrens im Vergleich zur Advancing Front Methode ist ebenfalls zu erkennen. Die iterative Vereinfachung der Punktewolke, bevor die eigentliche explizite Rekonstruktion stattfindet, bewirkt ein Entgegenwirken zu meinem Hinzufügen eines Rauschens. Die Anpassung der Punkte anhand einer Regressionsebene ist in gewisser Weise eine Art Umkehrung meines generierten Rauschens. Da dieser Schritt 4 Iterationen

durchlaufen hat, weist das Oberflächennetz des Scale Space Algorithmus keine primitiven Flächen auf, welche eine Verbindung zwischen Körper und Ohr, wie beispielsweise auch dem Beinbereich generiert. Mein normaler Advancing Front Algorithmus weist solche Dreiecksverbindungen auf. Dies ist ebenfalls den Abbildungen 10.22 und 10.23 zu entnehmen. Insgesamt repräsentieren jedoch beide explizite Verfahren das eingefügte Rauschen. Die Scale Space Methode projiziert schließlich die angepasste Punktewolke zurück auf die ursprünglichen Messwerte, welche das Rauschen repräsentieren.

Ist der Anteil an Rauschen irgendwann zu hoch, generiert auch ein implizites Verfahren, wie die Poisson Oberflächen Rekonstruktion, ein verrauschtes Oberflächennetz. In Abbildung 10.24 ist dies zu erkennen. Überraschenderweise sind jedoch die expliziten Verfahren nicht deutlich schlechter, wie bei dem vorherigen Rauschen. Es sind noch relativ wenig Löcher entstanden und der Hase ist noch gut erkennbar.

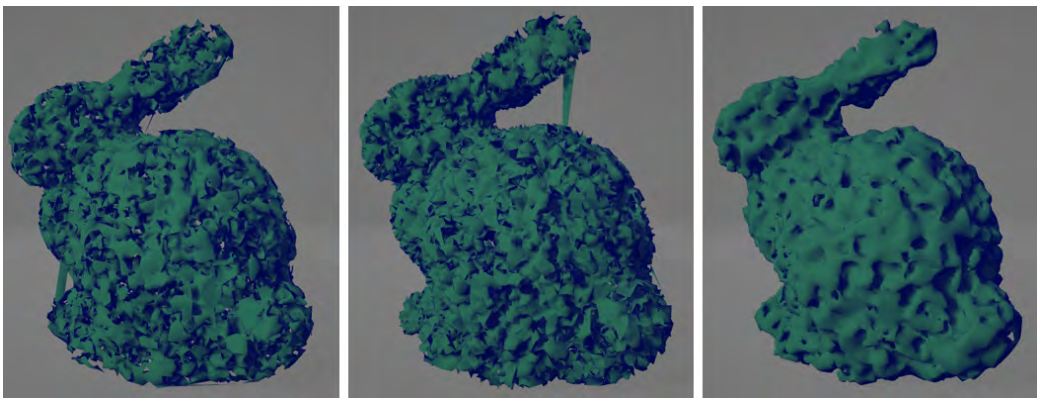

Abb. 10.24: Starkes Rauschen: Oberfläche ohne Kanten - explizit vs. implizit

11 Zusammenfassung Eigenschaften

11.1 Implizite Rekonstruktion

Der vorgestellte Poisson Oberflächen Rekonstruktionsalgorithmus ist ein implizites Verfahren und liefert immer ein geglättetes Ergebnis. Grundsätzlich wird dir Indikatorfunktion so ausgewertet, dass stets eine geschlossene Oberfläche erzeugt wird. In den Versuchen zeigte sich somit, dass der Algorithmus gut mit Rauschen umgehen kann, da er mithilfe der Glättung dem Rauschen entgegenwirkt. Ebenfalls ist ein Vorteil der Methode, dass einige fehlende Messwerte toleriert werden können. Zusätzlich beeinflussen auch einzelne Ausreißer das Ergebnis nicht stark.

Probleme hat der Algorithmus, falls die Punktewolke an einigen Stellen nicht dicht genug ist. Speziell in dünnen Bereichen des Objektes sollte die Abtastdichte hoch sein, sodass diese Methode dünne Bereiche rekonstruieren kann. Echte Kanten in einem Objekt werden durch den Algorithmus ebenfalls geglättet. Wird somit eine Oberfläche eines mechanischen Bauteils mit echten Kanten rekonstruiert, liefert das Resultat meist kein gewünschtes Ergebnis, da die echten Kanten verloren gehen. Wie sich im Versuch mit der Punktewolke, erstellt durch Photogrammetrie, herausstellte, hat der Algorithmus starke Probleme falls eine Vielzahl an falsch orientierten Normalenvektoren besteht.

Abgesehen vom qualitativen Ergebnis liefert der Poisson Oberflächen Rekonstruktionsalgorithmus ein Resultat, welches weniger Speicherbedarf benötigt. Die Knotenpunkte haben keinen direkten Bezug mehr zu den Eingangsdaten und sind quantitativ einige weniger. Dies führt zu größeren primitiven Flächen, welche so ebenfalls durch eine kleinere Anzahl an Flächen, die rekonstruierte Oberfläche darstellt. Das Ergebnis beinhaltet nur Dreiecke als Flächen und ist somit insgesamt eine Triangulierung.

11.2 Explizite Rekonstruktion

Die beiden vorgestellten expliziten Verfahren waren eine Variante eines Advancing Front Algorithmus, wie auch die spezielle Methode des Scale Space Meshings. Das Scale Space Meshing lieferte insgesamt einige Vorteile gegenüber einem normalen Advancing Front Algorithmus, dies jedoch auch nur zu einem gewissen Grad, mit dem die iterative Vereinfachung den fehlerhaften Daten entgegenwirken konnte.

Insgesamt verwendeten beide Algorithmen fast alle Eingangspunkte als Knotenpunkte, solang dies durch größeren Löcher in den Daten möglich war. Diese Verwendung spiegelt auch den höheren Speicherbedarf der Resultate wieder. Die Triangulierungen liefern eine ähnliche Anzahl an Knotenpunkten, wie Eingangspunkten, und somit auch eine sehr hohe Anzahl an primitiven Flächen. Diese Eigenschaft kann positiv, wie auch negativ sein.

Positiv betrachtet geht somit fast keine Information im Bezug auf den Scann verloren. Wenn die Eingangsdaten die Oberfläche gut repräsentieren liefern die Algorithmen somit ein sehr gutes Ergebnis. Löcher in den Datensätzen könnten zudem absichtlich bestehen und wollen nicht geschlossen werden. Speziell Oberflächen mit hohen Frequenzen auf der Oberfläche werden mit diesen Verfahren unter Berücksichtigung des Abtasttheorems gut rekonstruiert. Das Ergebnis wird grundsätzlich nicht geglättet. Ausreißer mit größerer Entfernung zur Oberfläche werden ignoriert. Dünne Bereiche müssen nicht zwingend eine enorme Menge an Eingangsdaten aufweisen, sondern können auch rekonstruiert werden, solange der Abstand zwischen den Messwerten nicht größer ist als die Grenze der Erweiterung der wachsenden Front.

Negativ gesehen werden hohe Frequenzen, wie auch Rauschen, mithilfe der Algorithmen rekonstruiert. Ausreißer nahe der Oberfläche beeinflussen das Ergebnis. Löcher im Resultat könnten unerwünscht sein. Der hohe Speicherbedarf könnte unnötig sein, da viele kleine Flächen des Ergebnisses auf der selben Ebene im Raum liegen könnten und diese somit durch größere primitive Flächen ähnlich gut repräsentiert werden könnten.

11.3 Rekonstruktion im Bezug auf die MAT

Schließlich wurde noch der Power Crust Algorithmus vorgestellt. Dieser rekonstruiert die Oberfläche mithilfe einer medialen Achsentransformation und beschreibt die Oberfläche als Schnittmenge von äußeren mit inneren Polarkreisen. Im Vergleich zu den anderen Verfahren ist die Power Kruste ein Polygonnetz mit beliebiger Größe und Anzahl der Eckpunkte der einzelnen Polygone. Der Algorithmus wird mit Erweiterungen vorgestellt, welche Löcher im Resultat möglich machen. Grundsätzlich liefert der Standardprozess immer eine geschlossene Oberfläche.

In den Versuchen zeigte sich, dass das Verfahren eine eher hohe Anzahl an Knotenpunkten, wie auch primitiven Flächen erzeugt. Dies ist im Kapitel Dateninformation ebenfalls zu begutachten. Ähnlich den Ergebnissen der expliziten Verfahren, könnten die Oberflächennetze nachbearbeitet werden und eine gewisse Menge an Flächen vereinfacht werden, ohne eine sichtliche Verschlechterung des Ergebnisses zu erhalten.

Der Algorithmus liefert die Grenzschicht eines soliden Objektes, somit sind die Knotenpunkte teilweise Eingangsdaten, manchmal jedoch auch nicht. Insgesamt sind Knotenpunkte nahe den Eingangsdaten. Dies ist positiv, da diese ja die ursprüngliche Oberfläche repräsentieren sollten. Die mediale Achse wird aufgrund der Pole relativ gut abgeschätzt, lieferte jedoch bei starken Rauschen und Ausreißern Flächen, welche die Oberfläche nicht mehr gut repräsentieren. In den Versuchen mit den annähernd idealen Datensätzen konnte man ein Ergebnis begutachten, welches hohe Information auf der Oberfläche gut darstellt. Eine Glättung wurde somit nicht festgestellt.

12 Abbildungshinweise

Die einzelnen Abbildungen in dieser Arbeit sind entweder von eigen erstellten Objekten oder Bilder aus angegebenen Quellen. Nicht eigens erstellte Grafiken sind mit dem jeweiligen Index versehen. Die selbst erstellten Abbildungen sind mithilfe verschiedener Software visualisiert worden. Genutzte Software ist hier MeshLab, 3D-Viewer und der Basic-Viewer(CGAL).

MeshLab [20] ist eine bekannte, freie Software zum Erstellen und Filtern von Punktewolken. Zusätzlich sind verschiedene Funktionen zur Oberflächen-Generierung gegeben. Die grafische Oberfläche kann hierbei Punktewolken, wie auch Oberflächennetze visualisieren. Die Rekonstruktionsfilter sind im Zusammenhang dieser Arbeit nicht verwendet worden. Alle Rekonstruktionen, bis auf den Power Crust Algorithmus, wurden mithilfe von CGAL erstellt. Meshlab wurde speziell zum manuellen Entfernen einiger starken Ausreißern der Photogrammetrie Versuche verwendet. Somit wurden die fehlerhaften Eingangsdaten zu einem gewissen Grad aufgewertet. Ebenso wurde mithilfe von MeshLab die Krümmung der Oberflächen ausgewertet und diese entsprechend eingefärbt.

Der 3D-Viewer [21] wurde speziell zum Veranschaulichen der Ergebnisse verwendet. Die Software bietet verschiedene Möglichkeiten zum visualisieren. Einerseits kann die Oberfläche ohne Kanten und Knotenpunkte dargestellt werden. In Verbindung mit der Möglichkeit das Objekt aus verschiedenen Winkeln mit farbigem Umgebungslicht anstrahlen zu lassen, bietet er eine gute Grundlage um verschiedene Eigenschaften der Triangulierungen genauer zu sehen. Speziell hochfrequente Information auf der Oberfläche ist so besser erkennbar. Ein weiterer Vorteil ist das Umschalten auf zwei weitere Ansichten. Einerseits kann man sich die primitiven Flächen (Dreiecke) der Triangulierung anzeigen lassen oder andererseits die Scheitelpunkte des Oberflächennetzes fokussieren.

Die hauptsächlich für die Rekonstruktionen verwendete Bibliothek CGAL [30] bietet ebenfalls eine Möglichkeit Punktewolken, wie auch Oberflächennetze zu

visualisieren. Um den Basic-Viewer verwenden zu können, muss das Makro CGAL_USE_BASIC_VIEWER definiert sein und die QT5 Komponente von CGAL muss dem Projekt verlinkt sein. Anschließend können Punktewolken, Triangulierungen und Delaunay-Hüllen visualisiert werden. Die Punktewolken werden standardmäßig mithilfe von roten Quadraten dargestellt. Von den verschiedensten Oberflächennetzen werden standardmäßig Kanten und Knotenpunkte in schwarz, so wie die Flächen in blau eingefärbt dargestellt. Mit Hilfe einer Tastenkombination können die Farben der primitiven Flächen geändert werden, sodass benachbarte Polygone unterschiedliche Farbeigenschaften bekommen, um so das Objekt besser wahrnehmen zu können.

Die Oberfläche wird beim Basic-Viewer unabhängig von der Richtung des Normalenvektors der einzelnen Flächen visualisiert. Das heißt, dass man bei einer Perspektive innerhalb des Objektes die Oberfläche eingefärbt von innen betrachten kann. Diese Eigenschaft ist beispielsweise beim zuvor vorgestellten 3D-Viewer nicht möglich. Hierbei kann es auch passieren, dass die Funktion des 3D-Viewers das Objekt aus verschiedenen Winkeln zu beleuchten, nicht funktioniert, da die Vektoren in die entgegengesetzte Richtung zeigen. Die von innen betrachtete Oberfläche ist im Normalfall schwarz und ohne Kanten, wie auch Knotenpunkten. Im Falle der entgegengesetzten Normalenvektoren wird die Oberfläche von außen schwarz dargestellt und von innen beleuchtet. Dieser Fall war nicht brauchbar und unerwünscht. In Verbindung mit meinen Versuchen ist mir diese Eigenschaft jedoch aufgefallen. Um somit diese Visualisierung nutzen zu können, müssen die Normalenvektoren des Ergebnisses in einem folgenden Schritt neu orientiert werden. Dies wird einfach durch das Ändern des Vorzeichens der Vektoren erreicht.

13 Dateninformation

Folgend wird für alle Eingangsdaten und erzeugten Oberflächen, die in dieser Arbeit erwähnt werden, die Anzahl an Knotenpunkten und Flächen angegeben.

Original Stanford Bunny - 35947 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	35947	71872 Dreiecke
Scale Space	35947	71878 Dreiecke
Poisson	2614	5224 Dreiecke
Power Crust	116200	232380 Polygone
Original	35947	69451 Dreiecke

Bunny - zufälliges Rauschen stark - 35947 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	35947	64779 Dreiecke
Scale Space	35947	67320 Dreiecke
Poisson	18040	37174 Dreiecke
Power Crust	130099	85762 Polygone

Bunny - zufälliges Rauschen mittel - 35947 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	35947	64880 Dreiecke
Scale Space	35947	68952 Dreiecke
Poisson	10354	20910 Dreiecke
Power Crust	117601	77630 Polygone

Bunny - zufälliges Rauschen leicht - 35947 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	35947	66768 Dreiecke
Scale Space	35947	71374 Dreiecke
Poisson	3048	6094 Dreiecke
Power Crust	101764	66831 Polygone

Bunny - orientiertes Rauschen stark - 35947 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	35947	64558 Dreiecke
Scale Space	35947	64986 Dreiecke
Poisson	28173	58764 Dreiecke

Bunny - orientiertes Rauschen mittel - 35947 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	35947	61389 Dreiecke
Scale Space	35947	66039 Dreiecke
Poisson	26018	54288 Dreiecke

Bunny - orientiertes Rauschen mittel2 - 35947 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	35947	62424 Dreiecke
Scale Space	35947	65865 Dreiecke
Poisson	18989	38862 Dreiecke

Bunny - orientiertes Rauschen leicht - 35947 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	35947	63811 Dreiecke
Scale Space	35947	71661 Dreiecke
Poisson	2544	5108 Dreiecke

Metallmutter - 446350 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	446350	892108 Dreiecke

Stanford Dragon - 437645 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Original	437645	871414 Dreiecke

Stanford Dragon - 100250 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	100250	200381 Dreiecke
Poisson	9310	18636 Dreiecke
Power Crust	206328	412644 Polygone

Stanford Dragon - 22998 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	22998	45940 Dreiecke
Poisson	3466	6946 Dreiecke
Power Crust	98708	67641 Polygone

Stanford Dragon - 5205 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	5205	10349 Dreiecke

Kristallbaum - 73442 Eingangspunkte		
Algorithmus	Knotenpunkte	Flächen
Advancing Front	73442	140491 Dreiecke
Scale Space	73442	142387 Dreiecke

Literaturverzeichnis

[1] Alliez, P. ; Giraudot, S. ; Jamin, C. ; Lafarge, F. ; Mérigot, Q. ; Meyron, J. ; Saboret, L. ; Salman, N. ; Wu, S. ; Yildiran, N. F.: Point Set Processing. Version: 5.3, 2021. https://doc.cgal.org/5.3/Manual/packages.html#PkgPointSetProcessing3. In: *CGAL User and Reference Manual*. 5.3. CGAL Editorial Board

[2] *Artec3D - Professionelle 3D-Scanner*. https://www.artec3d.com

[3] Bernardini, F. ; Mittleman, J. ; Rushmeir, H. ; Silva, C. ; Taubin, G. : The ball-pivoting algorithm for surface reconstruction. In: *IEEE Transactions on Visualization and Computer Graphics* (1999)

[4] Boissonnat, J.-D. ; Cazals, F. : Smooth Surface Reconstruction via Natural Neighbour Interpolation of Distance Functions. In: *Proceedings of the sixteenth annual symposium on Computational geometry* (2000), S. 223–232

[5] Bourke, P. : *Polygonising a Scalar Field Using Tetrahedrons*. http://paulbourke.net/geometry/polygonise/. Version: 1997

[6] CARR, J. ; BEATSON, R. ; CHERRIE, H. ; MITCHEL, T. ; FRIGHT, W. ; MCCALLUM, B. ; EVANS, T. : Reconstruction and representation of 3D objects with radial basis functions. In: *SIGGRAPH* (2001), S. 67–76

[7] Cline, W. E. L. E.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In: *ACM SIGGRAPH Computer Graphics 21* (1987), S. 163–167

[8] Cohen-Steiner, D. ; Da, T. K. F.: A greedy delaunay-based surface reconstruction algorithm. In: *The Visual Computer* (2004), S. 4–16

[9] *Marching Tetrahedra*. https://daac.hpc.mil/gettingStarted/Marching_Tetrahedra.html. – Latest Accessed: 01.08.2021

[10] *Delaunay Triangulation.* https://de-academic.com/dic.nsf/dewiki/312396. – Academic dictionaries and encyclopedias

[11] Digne, J. ; Morel, J.-M. ; Souzani, C.-M. ; Lartigue, C. : Scale space meshing of raw data point sets. In: *Computer Graphics Forum* (2011), S. 1630–1642

[12] Dwyer, R. A.: A faster divide-and-conquer algorithm for constructing delaunay triangulations. In: *Algorithmica* (1987), S. 137–151

[13] Edelsbrunner, H. : Triangulations and meshes in computational geometry. In: *Acta Nummerica* (2000), S. 1–81

[14] Grinspun, E. ; Krysl, P. ; Schröder, P. : CHARMS: a simple framework for adaptive simulation. In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002)

[15] Hoppe, H. ; DeRose, T. ; Duchamp, T. ; McDonald, J. ; Stuetzle, W. : Surface Reconstruction from Unorganized Points. In: *SIGGRAPH Comput. Graph.* 26 (1992), Jul., Nr. 2, 71–78. http://dx.doi.org/10.1145/142920.134011. – DOI 10.1145/142920.134011. – ISSN 0097–8930

[16] Huang, H. ; Li, D. ; Zhang, H. ; Ascher, U. ; Cohen-Or, D. : Consolidation of unorganized point clouds for surface reconstruction. In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2009)* 28 (2009), S. 176:1–176:7

[17] Kazhdan, M. ; Bolitho, M. ; Hoppe, H. : Poisson Surface Reconstruction. In: Sheffer, A. (Hrsg.) ; Polthier, K. (Hrsg.): *Symposium on Geometry Processing*, The Eurographics Association, 2006. – ISBN 3–905673–24–X

[18] Kindermann, P. : *Delaunay Triangulation.* https://syncandshare.lrz.de/getlink/fiEfQfpZPG6vZPtvqSntFAd6/ag-ss20-vl08-delaunay-triangulation.pdf. – Computational Geometry - Lecture 08 (Universität Passau)

[19] Kindermann, P. : *Voronoi Diagram.* https://syncandshare.lrz.de/getlink/fiQDuyfL3SFQ8k15J3B3DeHD/ag-ss20-vl07-voronoi-diagram.pdf. – Computational Geometry - Lecture 07 (Universität Passau)

[20] *MeshLab.* https://www.meshlab.net/. – CNR

[21] *3D - Viewer.* https://www.microsoft.com/de-de/p/3d-viewer/9nblggh42ths?activetab=pivot:overviewtab. – Microsoft Corporation

[22] MURAKI, S. : Volumetric shape description of range data using "blobby model". In: *Computer Graphics 25* (1991), S. 227–235

[23] Nina Amenta, K. K. Sunghee ChoiRavi C. Sunghee ChoiRavi: The Power Crust. In: *CiteSeer* (2004)

[24] *PLY - Polygon File Format.* http://paulbourke.net/dataformats/ply/

[25] Pauly, M. ; Gross, M. ; Kobbelt, L. : Efficient simplification of point-sampled surfaces. In: *IEEE Visualization, 2002. VIS 2002.* (2002), S. 163–170

[26] *Object File Format (.off).* https://shape.cs.princeton.edu/benchmark/documentation/off_format.html

[27] Rusu, R. B. ; Cousins, S. : 3D is here: Point Cloud Library (PCL). In: *IEEE International Conference on Robotics and Automation (ICRA).* Shanghai, China, May 9-13 2011

[28] *The Stanford 3D Scanning Repository.* http://graphics.stanford.edu/data/3Dscanrep/. – Latest Accessed: 11.08.2021

[29] Su, P. ; Drysdale, R. L. S.: A Comparison of Sequential Delaunay Triangulation Algorithms. In: *Computational Geometry: Theory and Applications* (1996), S. 361–385

[30] The CGAL Project: *CGAL User and Reference Manual.* 5.3. CGAL Editorial Board https://doc.cgal.org/5.3/Manual/packages.html

[31] TURK, G. ; O'BRIEN, J. : Modelling with implicit surfaces that interpolate. In: *TOG* (2002), S. 855–873

[32] *LAS Point Cloud Format.* https://www.usna.edu/Users/oceano/pguth/md_help/html/las_format.htm

Abbildungsverzeichnis

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit ohne fremde Hilfe und nur unter Verwendung der angegebenen Hilfsmittel selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich kenntlich gemacht.

Straßkirchen, den 11. Januar 2022

_____
Florian Himmelsbach