Seminar
Medizinische Simulationssysteme
SS 2005

Oberflächenrekonstruktion aus Punktwolken

Wolfgang Globke
Betreuer: Dominik Fritz

1. März 2006

Zusammenfassung

In der medizinische Bildverarbeitung liegen oft Daten in Form von im Raum verteilten Punkten vor, etwa wenn sie aus CT- oder MRT-Aufnahmen gewonnen werden, oder auch durch das Abtasten von Modellen durch Laserscanner oder strukturiertes Licht. Man ist an Verfahren interessiert, aus diesen sogenannten Punktwolken eine möglichst originalgetreue Rekonstruktion der abgetasteten Fläche zu erhalten. Solche Modelle bieten eine Vielzahl von Anwendungsmöglichkeiten in der Medizin, wie etwa die Simulation von Operationen am 3D-Modell, die physikalische Simulation von Körperteilen oder die Herstellung von Knochenimplantaten.

Inhaltsverzeichnis

1 Einführung und Motivation **3**

2 Geometrische Grundbegriffe **4**

3 Der Marching Cubes-Algorithmus **6**

3.1 Schnittbilder 6

3.2 Wandernde Würfel 6

3.3 Normalen auf der Fläche 8

3.4 Resultate 8

4 Der Algorithmus von Hoppe **9**

4.1 Voraussetzungen an die Daten 9

4.2 Tangentialebenen approximieren 10

4.3 Einheitliche Orientierung 11

4.4 Die orientierte Abstandsfunktion 12

4.5 Konstruktion der Dreiecksfläche 13

4.6 Resultate 13

5 Der Power Crust-Algorithmus **14**

5.1 Mediale Achsen-Transformation und Voronoi-Diagramme 14

5.2 Approximation der MAT durch Polarkugeln 15

5.3 Innere und äußere Pole bestimmen 16

5.4 Power Crust 17

5.5 Resultate 18

6 Ausblick **20**

Literatur **21**

Index **22**

1 Einführung und Motivation

Die Daten, die aus modernen bildgebenden Verfahren wie der Computertomographie (CT) oder der Magnetresonanztomographie (MRT) oder auch durch das optische Abtasten von Modellen durch Laserscanner gewonnen werden, liefern eine Vielzahl von Anwendungsmöglichkeiten für die Medizin, die vorher nicht in dieser Weise denkbar gewesen wären. Für viele dieser Anwendungen ist es notwendig, aus den meistens in Form einer unzusammenhängenden Punktwolke vorliegenden Datenpunkten eine dreidimensionale Fläche zu rekonstruieren. In diesem Abschnitt sollen einige dieser Anwendungen kurz vorgestellt werden, bevor in den nächsten Abschnitten einige ausgewählte Algorithmen für die Flächenkonstruktion erläutert werden. Ausführlichere Beschreibungen der folgenden Anwendungsgebiete aus Sicht des Mediziners finden sich im Seminarband [10].

Dreidimensionale Modelle des Patienten ermöglichen eine **virtuelle Operationsplanung**, d.h. man kann eine bevorstehende Operation am Rechner planen und beliebig oft simulieren. Denkbar ist auch das Verschmelzen von Daten aus verschiedenen bildgebenden Verfahren zu einem dreidimensionalen Modell. Bei vorhander Technik ist es dem Chirurgen sogar möglich, sich während der Operation am dreidimensionalen Modell zu orientieren, etwa um zu wissen, wie nah er bereits einem zu entfernenden Tumor ist. Eine solche Planung ermöglicht eine minimalinvasive Operation, d.h. die Einschnitte in den Körper des Patienten sind nur so groß, wie unbedingt notwendig, um die Operation durchzuführen.

Dreidimensionale Modelle werden auch für das sogenannte **Rapid Prototyping** benötigt. Unter diesem Begriff werden Verfahren zur künstlichen Herstellung von Knochenstücken oder inneren Organen zusammengefasst. So kann es bei manchen Operationen notwendig sein, ein Stück vom Knochen des Patienten zu zerstören, um zum Operationsgebiet vorzudringen. Damit der Patient nicht an einer dauerhaften Entstellung zu leiden hat, kann vor der Operation ein genaues dreidimensionales Modell des entsprechenden Knochenstückes angefertigt werden, mit dessen Hilfe ein Knochenimplantat hergestellt wird, das nach der Operation an der Stelle des alten Knochens eingesetzt werden kann. Ein weitere Anwendung besteht in der Herstellung von Modellen innerer Organe zu Lehrzwecken. So werden für die Darstellung bestimmter Organmissbildungen keine echten Präparate mehr benötigt, die naturgemäß nur selten verfügbar sind, sondern man kann aus einem dreidimensionalen Modell eines echten Präparates beliebig viele Imitate herstellen.

Für **physikalische Simulationen** werden ebenfalls dreidimensionale Oberflächenmodelle benötigt. Beispielsweise kann die Deformation und Bewegung der Wirbelsäule durch die Finite Element-Methode berechnet werden, indem man eine Dreiecksflächendarstellung der Wirbelsäule zugrunde legt, wie sie bei einem Rekonstruktionsalgorithmus erzeugt wird. Hohe hat in seiner Dissertation [5] ein Verfahren entwickelt, um mit Hilfe einer Oberflächenrekonstruktion des Kniegelenkes die Krümmung der Knorpelplatte zu bestimmen und somit auf die Druckverteilung zwischen den Gelenkenflächen schließen zu können.

2 Geometrische Grundbegriffe

In diesem Abschnitt werden einige geometrische Begriffe und die Notation festgelegt, die im Laufe dieser Arbeit verwendet werden.

Punkte und Vektoren im Raum werden fett geschrieben, etwa $\boldsymbol{x} \in \mathbb{R}^3$. Mit $\mathcal{S} \subset \mathbb{R}^3$ wird eine Fläche im Raum bezeichnet, und es sei $\mathcal{T}(\boldsymbol{x})$ ihre Tangentialebene in $\boldsymbol{x} \in \mathcal{S}$, sofern diese existiert. Ein Einheitsnormalenvektor von $\mathcal{T}(\boldsymbol{x})$ wird mit $\boldsymbol{n}(\boldsymbol{x})$ bezeichnet.

Als **parametrische Fläche** im $\mathbb{R}^3$ wird eine Fläche $\mathcal{S}$ bezeichnet, die das Bild einer Funktion

$$\boldsymbol{x}(s,t) = \begin{pmatrix} x(s,t) \\ y(s,t) \\ z(s,t) \end{pmatrix}$$

von zwei Parametern s und t ist.

Als **algebraische** oder **implizite Fläche** wird eine Fläche bezeichnet, die die Nullstellenmenge

$$\mathcal{S} = \mathcal{Z}(f) := \{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x}) = 0\}$$

eines Polynoms $f \in \mathbb{R}[x,y,z]$ ist.

Unter einer **Dreiecksfläche** soll hier eine Menge von Dreiecken verstanden werden, von denen sich je zwei entweder in einer gemeinsamen Dreiecksseite, einem gemeinsamen Eckpunkt oder überhaupt nicht schneiden. Anschaulich sollen die Dreiecke also ordentlich aneinanderhängen und sich nicht in bizarrer Weise gegenseitig durchdringen. Entsprechend seien auch **Polygonflächen** definiert, die sich aus Polygonen mit variabler Eckenzahl zusammensetzen.

In etwas laxer Sprechweise sei mit der **Triangulierung** einer Fläche $\mathcal{S}$ eine Dreiecksfläche gemeint, die $\mathcal{S}$ approximiert und deren Knoten Punkte auf oder nahe bei $\mathcal{S}$ sind.

Von den Flächen, die im Folgenden betrachtet werden, wird verlangt, dass sie zumindest lokal durch eine stetige Funktion parametrisierbar sind.

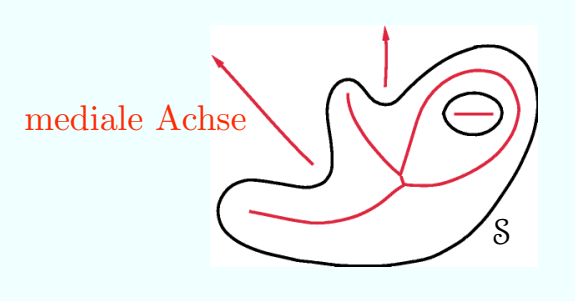
Für $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3$ bezeichne $\boldsymbol{x} \cdot \boldsymbol{y}$ das Standardskalarprodukt und $\|\boldsymbol{x}\|$ die euklidische Norm. Der Abstand von $\boldsymbol{x}$ zu einer Menge $M \subseteq \mathbb{R}^3$ ist definiert durch

$$\operatorname{dist}(\boldsymbol{x}, M) = \min\{\|\boldsymbol{x} - \boldsymbol{y}\| : \boldsymbol{y} \in M\},$$

also durch den minimalen Abstand zu einem Punkt von M.

Eine Fläche $\mathcal{S}$ heißt **orientierbar**, wenn man der Fläche eine Unter- und Oberseite zuordnen kann. Formal bedeutet dies, dass es eine differenzierbare Funktion $\boldsymbol{n} : \mathcal{S} \to \mathbb{R}^3$ von Einheitsnormalenvektoren auf $\mathcal{S}$ gibt. Die Wahl einer solchen Funktion bezeichnet man als **Orientierung**. Sie entspricht der Festlegung, welche Seite von $\mathcal{S}$ oben (bzw. außen) und welche unten (bzw. innen) sein soll.

Die **mediale Achse** einer Fläche $\mathcal{S}$ ist der Abschluss der Menge aller Punkte im $\mathbb{R}^3$, die zu mindestens zwei Punkten von $\mathcal{S}$ den gleichen Abstand haben. Ist die Fläche $\mathcal{S}$ orientierbar, so spaltet sich die mediale Achse in einen inneren und äußeren Teil auf. In der Literatur wird der Begriff gelegentlich nur für den inneren Teil verwendet.



Die **Kugel** mit Radius r und Mittelpunkt $\boldsymbol{x}$ ist

$$\mathcal{B}(\boldsymbol{x}, r) = \{\boldsymbol{y} \in \mathbb{R}^3 : \|\boldsymbol{x} - \boldsymbol{y}\| \leq r\},$$

ihr Rand

$$\partial\mathcal{B}(\boldsymbol{x}, r) = \{\boldsymbol{y} \in \mathbb{R}^3 : \|\boldsymbol{x} - \boldsymbol{y}\| = r\}$$

wird als **Sphäre** mit Radius r und Mittelpunkt $\boldsymbol{x}$ bezeichnet

Unter der **Rekonstruktion** einer Fläche $\mathcal{S}$ ist hier das Berechnen eines Approximanten von $\mathcal{S}$ anhand einer gegebenen Menge $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\} \subset \mathbb{R}^3$ von Datenpunkten zu verstehen, die auf oder nahe bei $\mathcal{S}$ liegen. In laxer Sprechweise wird auch der Approximant mit $\mathcal{S}$ bezeichnen, sofern keine Verwechslungsgefahr besteht.

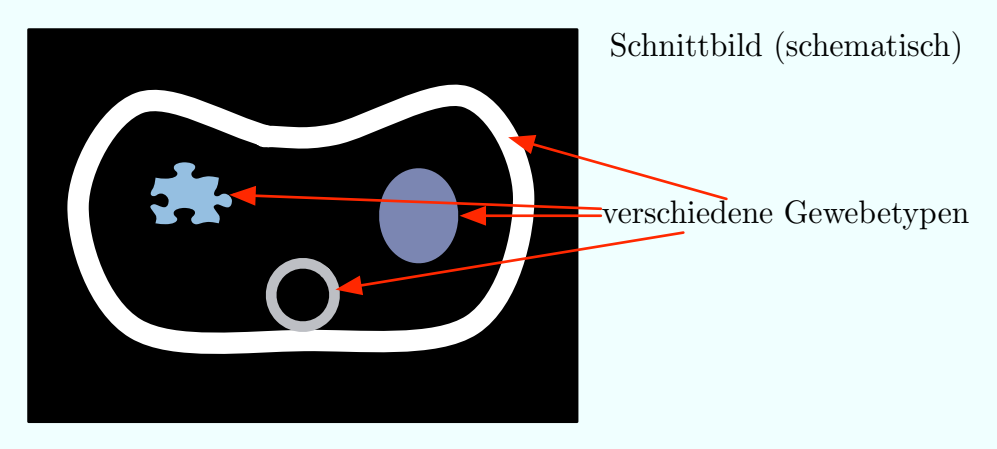
Liegt der Menge X keine (bekannte) Struktur zugrunde, so wird X auch als **Punktwolke** bezeichnet.

3 Der Marching Cubes-Algorithmus

Nimmt man es ganz genau, so ist der 1987 von Lorensen und Cline in [9] vorgestellte Marching Cubes-Algorithmus in dieser Arbeit fehl am Platz, da er dreidimensionale Oberflächen nicht aus Punktwolken, sondern aus zweidimensionalen Schnittbildern rekonstruiert. Wir besprechen ihn hier trotzdem, da er lehrreich ist und außerdem eine Variante davon im Algorithmus von Hoppe in Kapitel 4 für die Darstellung einer impliziten Fläche verwendet wird.

3.1 Schnittbilder

Als Ausgangsdaten liegen aufeinanderfolgende zweidimensionale Schnittbilder des zu rekonstruierenden Objektes vor, wie man sie etwa bei der CT oder der MRT erhalten kann. Da verschiedene Gewebetypen sich in ihrer Dichte ϱ unterscheiden, werden sie in den Schnittbildern durch unterschiedliche Grauwerte dargestellt.

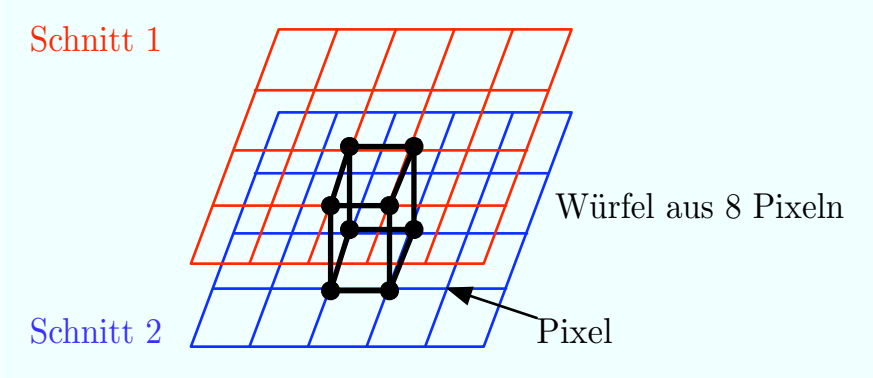


Die Idee ist nun, die Fläche $\mathcal{S}$, deren Gewebedichte einem vom Benutzer vorgegebenen Grauwert entspricht, zu rekonstruieren, indem man sie durch „Ablaufen“ der Pixel lokalisiert und dabei trianguliert. Um einen Schattierungsalgorithmus auf die rekonstruierte Fläche anwenden zu können, werden die Normalen der Fläche an jeder Ecke eines Dreiecks bestimmt.

Führt man den Algorithmus für verschiedene Grauwerte aus, so erhält man die präzise voneinander getrennten Flächen, die den Oberflächen der verschiedenen Gewebetypen entsprechen.

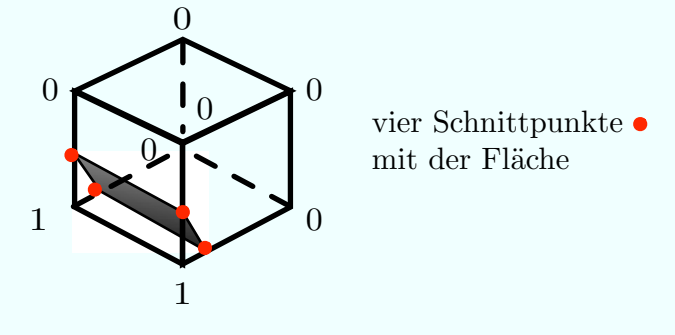
3.2 Wandernde Würfel

Der Algorithmus lokalisiert die Fläche $\mathcal{S}$ innerhalb von „logischen Würfeln“, die sich aus acht Pixeln von zwei aufeinanderfolgenden Schnittbildern zusammensetzen, wobei je vier Pixel aus dem gleichen Schnitt kommen:



Um herauszufinden, wie die Fläche diesen Würfel schneidet, wird jeder Ecke des Würfels der Wert 1 zugewiesen, wenn der Grauwert (also die Dichte) des zugehörigen Pixels größer oder gleich dem Wert der Fläche ist, ansonsten wird der Ecke der Wert 0 zugewiesen. Ein Würfel schneidet $\mathcal{S}$, wenn eine Ecke mit 0 und eine mit 1 markiert ist.

Da der Würfel 8 Ecken hat, gibt es nur $2^8 = 256$ Möglichkeiten, wie $\mathcal{S}$ den Würfel schneiden kann. Wie die Graphik verdeutlich, kann man aus der Markierung der Ecken ablesen, wie ein Würfel die Fläche schneidet:



Diese Fälle werden durchnumeriert und man erhält so eine vollständige Tabelle für alle möglichen Schnitte der Fläche mit einem Würfel. Berücksichtigt man noch Symmetrien, so kann man diese auf 14 Möglichkeiten reduzieren, eine Auflistung findet sich in [9].

Um den genauen Schnittpunkt entlang einer Kante des Würfels zu bestimmen, wird die Dichtefunktion entlang dieser Kante linear interpoliert und daraus der Punkt auf der Kante ermittelt, der den Dichtewert der Fläche annimmt. Weiß man, wie die Fläche den Würfel schneidet, wird sie innerhalb des Würfels durch eine stückweise lineare Dreiecksfläche approximiert.

Danach „wandert“ der Würfel zum nächsten Pixel-Oktett und die Prozedur wiederholt sich. Insgesamt erhält man so eine Approximation von $\mathcal{S}$ aus Dreiecksflächenstücken.

3.3 Normalen auf der Fläche

Im letzten Schritt des Algorithmus wird für jeden Knoten der Dreiecksfläche ein Einheitsnormalenvektor auf dieser Fläche berechnet. Dies ist für die Schattierung bei der dreidimensionalen Darstellung erforderlich. Um die Normalen zu bestimmen, kann man ausnutzen, dass die Dichtefunktion ϱ entlang einer Fläche von konstanter Dichte die Ableitung 0 hat. Anders formuliert bedeutet dies, dass der Gradient $\nabla\varrho$ dort senkrecht auf der Fläche steht. Da am Rand von $\mathcal{S}$ verschiedene Gewebetypen angrenzen, ist zwangsläufig $\nabla\varrho \neq \mathbf{0}$. Somit kann man einfach den normierten Gradienten als Normalenvektor der Approximation von $\mathcal{S}$ wählen.

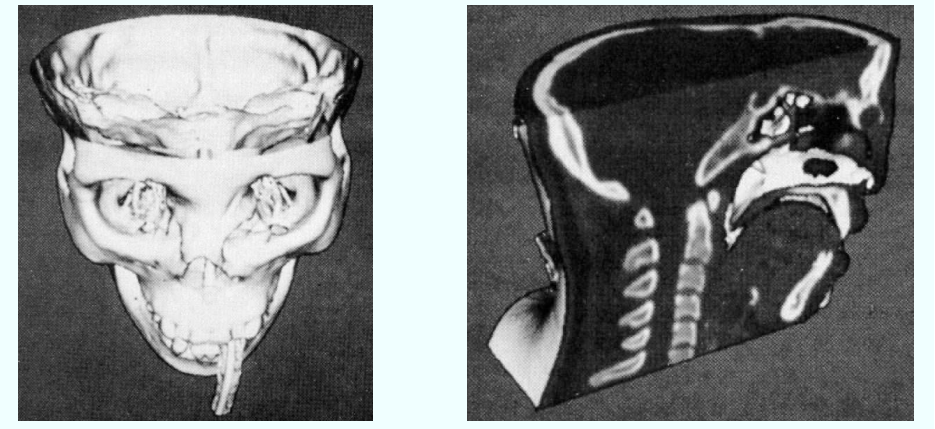
Um den Gradienten $\nabla\varrho$ in den Eckpunkten des Dreiecksnetzes abzuschätzen, schätzt man ihn zuerst in den Eckpunkten der Würfel durch

$$
\begin{aligned}
(\nabla\varrho)_x(i,j,k) &= \frac{\varrho(i+1,j,k) - \varrho(i-1,j,k)}{\Delta} \\
(\nabla\varrho)_y(i,j,k) &= \frac{\varrho(i,j+1,k) - \varrho(i,j-1,k)}{\Delta} \\
(\nabla\varrho)_z(i,j,k) &= \frac{\varrho(i,j,k+1) - \varrho(i,j,k-1)}{\Delta}
\end{aligned}
$$

ab, wobei $\varrho(i,j,k)$ die Dichte am Pixel (i,j) im k-ten Schnitt und Δ die Kantenlänge eines Würfels ist. Dann interpoliert man diese Normalen linear am Eckpunkt des Dreiecksnetzes, um die gesuchte Normale zu erhalten.

3.4 Resultate

Das linke Bild zeigt die Rekonstruktion von Knochengewebe aus einem CT-Datensatz durch den Marching Cubes-Algorithmus, das rechte Bild zeigt einen Schnitt durch die rekonstruierte Fläche, wobei für die Texturen die urspünglichen CT-Schnittbilder verwendet wurden (beide Bilder aus [9]).



Man sieht an diesen Bildern, wie sauber der Algorithmus das Knochengewebe vom Weichgewebe getrennt hat.

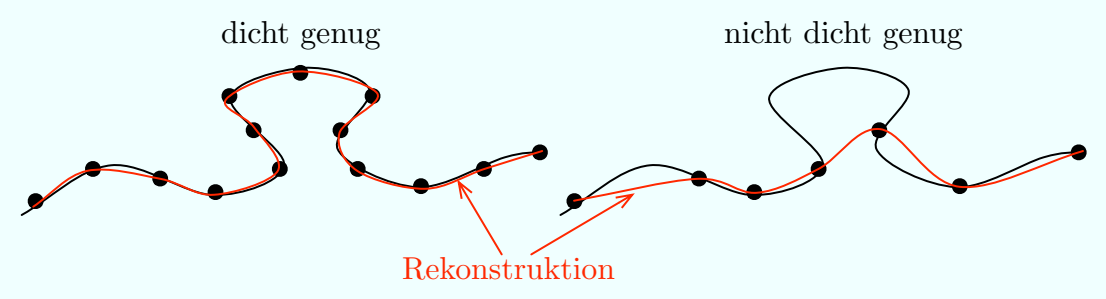
4 Der Algorithmus von Hoppe

In seiner Dissertation stellte Hoppe 1994 erstmals einen Algorithmus vor, der eine Oberfläche aus einer beliebigen Punktwolke $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ rekonstruiert, ohne, wie bis dahin üblich, Annahmen über die topologische Struktur der zu rekonstruierenden Fläche auszunutzen. Eine Beschreibung des Algorithmus findet sich in [6]. Inzwischen gibt es eine Vielzahl von Algorithmen, die nach dem gleichen Grundprinzip wie Hoppes Algorithmus verfahren, nämlich der Approximation durch eine implizite Fläche.

Der Algorithmus geht in zwei Schritten vor. Zuerst wird eine Funktion f definiert, die den „orientierten“ Abstand von $\mathcal{S}$ in einem Bereich $\mathcal{D} \subseteq \mathbb{R}^3$ abschätzen soll, der nahe bei $\mathcal{S}$ liegt. Mit der Nullstellenmenge $\mathcal{Z}(f)$ erhalten wir eine Approximation von $\mathcal{S}$ durch eine algebraische Fläche. Danach wird $\mathcal{Z}(f)$ mit einer Variante des Marching Cubes-Algorithmus durch eine Dreiecksfläche approximiert. Dass nicht die gewöhnliche Abstandsfunktion dist für den Algorithmus verwendet wird, hat den Grund, dass 0 ein singulärer Wert von dist ist und die aus der Rekonstruktion resultierende Fläche unschöne topologische Eigenschaften haben könnte. Konkret ist dies bei der Anwendung des Marching Cube-Algorithmus von Bedeutung, wenn zwischen Innen- und Außenseite der Fläche unterschieden werden muss.

4.1 Voraussetzungen an die Daten

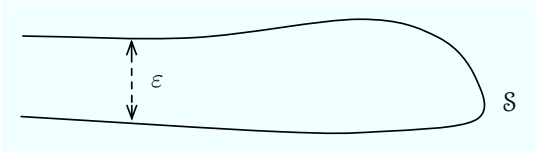
Damit eine sinnvolle Rekonstruktion der Fläche $\mathcal{S}$ aus den Datenpunkten X überhaupt möglich ist, müssen die Daten dicht genug beieinander liegen, um alle Strukturen der Fläche wiederzugeben.



X heißt **ε-dicht**, wenn für jeden Punkt $\boldsymbol{y} \in \mathcal{S}$ die Kugel $\mathcal{B}(\boldsymbol{y}, \varepsilon)$ mindestens einen Punkt aus X enthält. Anschaulich bedeutet dies, dass die Datenpunkte keine Löcher mit einem Radius größer als ε zulassen. Gilt also $\text{dist}(\boldsymbol{p}, X) > \varepsilon$ für einen Punkt $\boldsymbol{p} \in \mathbb{R}^3$, so kann $\boldsymbol{p}$ nicht auf der Fläche $\mathcal{S}$ liegen. Eine Abschätzung für ε kann man je nach Anwendung durch eine Analyse des Messvorgangs erhalten.

Strukturen von $\mathcal{S}$, die sehr klein sind verglichen mit ε lassen sich nicht auflösen. Die Punkte auf gegenüberliegenden Seiten einer „Falte“ von $\mathcal{S}$ müssten einen Abstand

größer als ε haben, damit die Falte als solche erkannt wird, denn nur dann kann der Algorithmus sichergehen, dass zwischen den beiden Seiten der Falte keine weiteren Punkte der Fläche liegen.

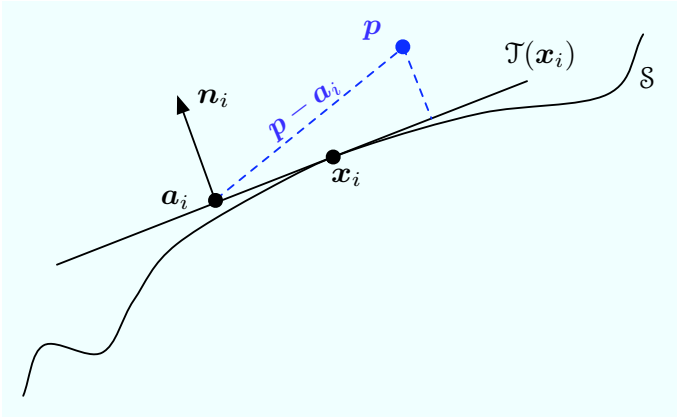


Gegebenenfalls muss man den Parameter ε noch um einen möglichen Fehler δ beim Messen der Datenpunkte X korrigieren.

4.2 Tangentialebenen approximieren

Um eine orientierte Abstandsfunktion definieren zu können, muss jedem Datenpunkt $\boldsymbol{x}_i$ eine orientierte Ebene $\mathcal{T}(\boldsymbol{x}_i)$ zugeordnet werden. Diese Ebene fasst man als Tangentialebene von $\mathcal{S}$ im Punkt $\boldsymbol{x}_i$ auf. Das bedeutet, dass $\mathcal{T}(\boldsymbol{x}_i)$ eine lineare Approximation von $\mathcal{S}$ in der Nähe von $\boldsymbol{x}_i$ ist.

Man stellt $\mathcal{T}(\boldsymbol{x}_i)$ durch einen Aufpunkt $\boldsymbol{a}_i \in \mathcal{T}(\boldsymbol{x}_i)$ und einen Einheitsnormalenvektor $\boldsymbol{n}_i$ von $\mathcal{T}(\boldsymbol{x}_i)$ dar.



Der **orientierte Abstand** von einem beliebigen Punkt $\boldsymbol{p} \in \mathbb{R}^3$ zu $\mathcal{T}(\boldsymbol{x}_i)$ wird durch

$$\mathrm{dist}_i(\boldsymbol{p}) := (\boldsymbol{p} - \boldsymbol{a}_i) \cdot \boldsymbol{n}_i$$

definiert. Für die Bestimmung von $\mathcal{T}(\boldsymbol{x}_i)$ betrachtet man die k Punkte aus X, die $\boldsymbol{x}_i$ am nächsten liegen (einschließlich $\boldsymbol{x}_i$). Diese Punkte werden als **Umgebung** $U(\boldsymbol{x}_i) = \{\boldsymbol{x}_1^{(i)}, ..., \boldsymbol{x}_k^{(i)}\}$ von $\boldsymbol{x}_i$ bezeichnet. Dann werden $\boldsymbol{a}_i$ und $\boldsymbol{n}_i$ so bestimmt, dass die Ebene

$$\mathcal{T}(\boldsymbol{x}_i) = \{\boldsymbol{p} \in \mathbb{R}^3 : (\boldsymbol{p} - \boldsymbol{a}_i) \cdot \boldsymbol{n}_i = 0\}$$

die Ausgleichsebene der Punkte aus $U(\boldsymbol{x}_i)$ ist. Dafür wählen wir

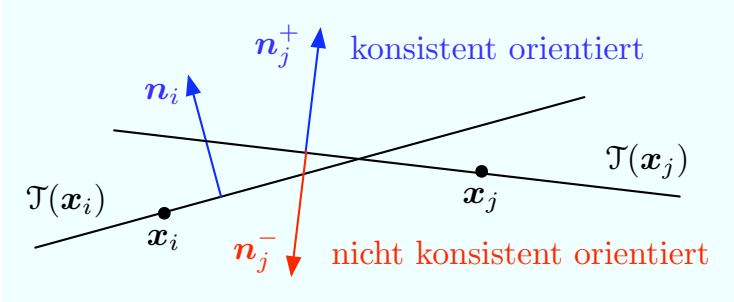
$$\boldsymbol{a}_i = \frac{1}{k}\sum_{j=1}^{k} \boldsymbol{x}_j^{(i)}$$

als Mittelpunkt der Punkte aus $U(\boldsymbol{x}_i)$ und setzen $A := (\boldsymbol{x}_1^{(i)} - \boldsymbol{a}_i, ..., \boldsymbol{x}_k^{(i)} - \boldsymbol{a}_i)$. Dann muss $\boldsymbol{n}_i$ so bestimmt werden, dass es die Fehlerquadrate minimiert, d.h. der Ausdruck $\|\boldsymbol{n}_i^\top A\|^2 = \boldsymbol{n}_i^\top A A^\top \boldsymbol{n}_i$ soll minimal sein. Da die Matrix $AA^\top$ symmetrisch und positiv definit ist, hat sie drei Eigenwerte $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ zu den normierten Eigenvektoren $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ und $\boldsymbol{v}_3$. Es zeigt sich, dass die Gleichung für $\boldsymbol{n}_i = \pm\boldsymbol{v}_3$ minimal wird. Das Vorzeichen sollte nun so gewählt werden, dass die Orientierung „benachbarter“ Tangentialflächen übereinstimmt.

4.3 Einheitliche Orientierung

Im Abschnitt 4.2 wurde bereits eine orientierte Abstandsfunktion für die approximierten Tangentialebenen definiert. Um diese Funktion auf die zu rekonstruierende Fläche $\mathcal{S}$ zu übertragen, muss die Orientierung der Tangentialebenen $\mathcal{T}(\boldsymbol{x}_1), ..., \mathcal{T}(\boldsymbol{x}_n)$ aller Datenpunkte aus X konsistent sein, d.h. es sollte keine plötzlichen Sprünge von der Oberseite zur Unterseite geben.

Liegen $\boldsymbol{x}_i$ und $\boldsymbol{x}_j$ nahe beieinander, so sollten ihre Tangentialebenen „ungefähr parallel“ sein, was $\boldsymbol{n}_i \cdot \boldsymbol{n}_j \approx \pm 1$ bedeutet. Um eine einheitliche Orientierung der Tangentialebenen zu gewährleisten, muss $\boldsymbol{n}_i \cdot \boldsymbol{n}_j \approx +1$ gelten, andernfalls sollte einer der beiden Normalenvektoren umgekehrt werden.



Dieses Problem kann man als Optimierungsproblem auf einem Graphen darstellen, wenn jeder Tangentialfläche $\mathcal{T}(\boldsymbol{x}_i)$ ein Knoten N_i im Graphen zugeordnet wird und zwei Knoten N_i und N_j genau dann eine Kante haben, wenn die Aufpunkte $\boldsymbol{a}_i$ und $\boldsymbol{a}_j$ der Ebenen „nahe genug“ beieinander liegen. Dafür bestimmt man zuerst einen Baum mit minimaler euklidischer Kantenlänge, der alle $\boldsymbol{a}_i$ verbindet. Da dieser Baum für praktische Belange oft nicht dicht genug ist, fügt man Kanten zwischen denjenigen N_i, N_j ein, für die $\boldsymbol{a}_i \in U(\boldsymbol{a}_j)$ gilt. Dieser sogenannte **Riemann-Graph** codiert nach Konstruktion die geometrische Nähe der Aufpunkte der Tangentialebenen. Wählt man als Kantengewicht den „Grad an Gleichorientiertheit“ $w_{ij} := \boldsymbol{n}_i \cdot \boldsymbol{n}_j$,

so besteht das Problem dann darin, die Orientierungen so zu wählen, dass das Gesamtgewicht des Graphen maximal wird. Da dieses Problem unglücklicherweise NP-schwer ist, drängt sich die Verwendung eines Approximationsalgorithmus auf.

Ein naiver Ansatz wäre, für irgendein $\mathcal{T}(\boldsymbol{x}_i)$ die Orientierung festzulegen und ausgehend davon die Orientierung an die benachbarten Ebenen weiterzugeben. Dies kann aber zu gravierenden Fehlern in der Rekonstruktion führen.

Stattdessen soll die Orientierung zunächst an solche $\mathcal{T}(\boldsymbol{x}_j)$ weitergegeben werden, die ungefähr parallel zu $\mathcal{T}(\boldsymbol{x}_i)$ sind. Dies wird durch das Kantengewicht $w_{ij} := 1-|\boldsymbol{n}_i \cdot \boldsymbol{n}_j|$ ausgedrückt. Da das Gewicht klein ist für Ebenen, die ungefähr parallel sind, kann man die Reihenfolge, in die Knoten abgearbeitet werden, über den minimalen Spannbaum des Riemann-Graphen erhalten. Diese Reihenfolge zum Weitergeben der Orientierung ist deshalb günstiger, weil sie die Orientierung bevorzugt in eine Richtung mit geringer Krümmung weitergibt, was schwierige Situtationen an scharfen Kanten der Fläche vermeidet.

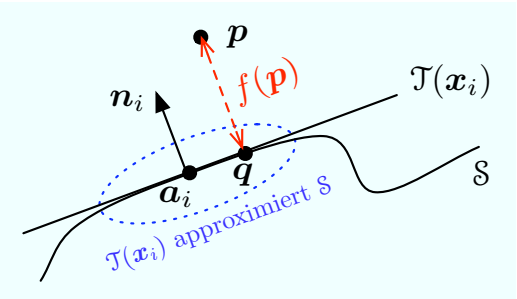
Die Anfangsorientierung spielt dabei zwar keine Rolle bei der späteren Konstruktion eines Dreiecksnetzes, ist aber wichtig für die Festlegung von Innen- und Außenseite der Fläche. Daher wählt man als Startknoten des Algorithmus das $\boldsymbol{x}_i$ mit maximaler z-Koordinate und legt $\boldsymbol{n}_i$ so fest, dass es in z-Richtung orientiert ist.

4.4 Die orientierte Abstandsfunktion

Die **orientierte Abstandsfunktion** $f(\boldsymbol{p})$ von einem beliebigen Punkt $\boldsymbol{p}$ zur Fläche $\mathcal{S}$ ist $\operatorname{dist}(\boldsymbol{p}, \mathcal{S})$ multipliziert mit einem Faktor ± 1, je nachdem, auf welcher Seite von $\mathcal{S}$ der Punkt $\boldsymbol{p}$ liegt.

Da die wirkliche Fläche $\mathcal{S}$ nicht bekannt ist, wird die orientierte Abstandsfunktion mit Hilfe der orientierten Tangentialflächen angenähert. Dafür muss eine Ebene $\mathcal{T}(\boldsymbol{x}_i)$ mit $\|\boldsymbol{a}_i - \boldsymbol{p}\| \leq \|\boldsymbol{a}_j - \boldsymbol{p}\|$ für alle j bestimmt werden. Da $\mathcal{T}(\boldsymbol{x}_i)$ eine Approximation von $\mathcal{S}$ in der Nähe von $\boldsymbol{x}_i$ darstellt, wird die orientierte Abstandsfunktion durch den Abstand von $\boldsymbol{p}$ zu seiner Projektion $\boldsymbol{q}$ auf die nächste Tangentialebene $\mathcal{T}(\boldsymbol{x}_i)$ definiert:

$$f(\boldsymbol{p}) := \operatorname{dist}_i(\boldsymbol{p}) = (\boldsymbol{p} - \boldsymbol{a}_i) \cdot \boldsymbol{n}_i.$$



Falls die Fläche offene Stellen hat, so könnte es passieren, dass diese vom Algorithmus geschlossen werden. Aus Abschnitt 4.1 wissen wir, dass die Datenpunkte keine Löcher in der Fläche mit einem Radius größer als ε zulassen. Ist also für die Projektion $\boldsymbol{q}$ von $\boldsymbol{p}$ der Abstand $\operatorname{dist}(\boldsymbol{q}, X) > \varepsilon$, so wissen wir, dass $\boldsymbol{q}$ nicht auf der Fläche liegen kann und setzen $f(\boldsymbol{p}) := \perp$.

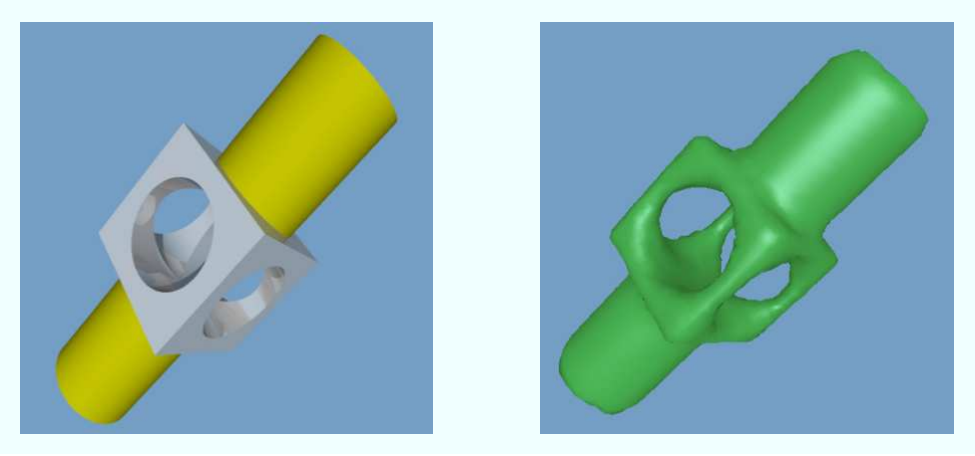
4.5 Konstruktion der Dreiecksfläche

In Kapitel 3 haben wir gesehen, wie man mit dem Marching Cubes-Algorithmus eine Fläche rekonstruieren kann, indem man eine Fläche mit festem Grauwert „entlangwandert“. Diese Verfahren soll nun zur Approximation von $\mathcal{S}$ durch eine Fläche aus Dreiecksflächenstücken angewandt werden. Anstelle eine Dichtefunktion für das Gewebe werden hier als „Grauwerte“ die Funktionswerte von f verwendet, d.h. man wandert die Punkte entlang, die den Abstand 0 zur (approximierten) Fläche $\mathcal{S}$ haben.

Da die Daten hier nicht im Pixelformat vorliegen, wird der die Datenpunkte X umgebende Raum in Würfel mit Kantenlänge $\leq \varepsilon$ unterteilt, wobei im Algorithmus nur Würfel berücksichtigt werden, die die Nullstellenmenge $\mathcal{Z}(f)$ schneiden. Diese werden daran erkannt, dass f für mindestens einen der Eckpunkte des Würfels einen Wert ungleich $\perp$ liefert. Für jeden Würfel werden dabei die Funktionswerte von f an den Eckpunkten des Würfels bestimmt und wie in Kapitel 3 die Fläche innerhalb des Würfels trianguliert.

4.6 Resultate

Das linke Bild zeigt die dreidimensionale Modellierung eines Bauteils, das rechte die Rekonstruktion davon durch den Algorithmus von Hoppe. Man sieht, dass der Algorithmus scharfe Kanten und Ecken nicht optimal rekonstruiert. Hoppe hat diverse Modifikationen vorgeschlagen, um dies zu verbessern, siehe [6].



Beide Bilder stammen aus [6], wo sich auch noch ein Vielzahl weiterer Bilder zu diesem Algorithmus finden lässt.

5 Der Power Crust-Algorithmus

Der Power Crust-Algorithmus wurde 2001 von Amenta, Choi und Kolluri in [1] vorgestellt und dient dazu, eine „wasserdichte Oberfläche“ (d.h. die Rekonstruktion hat keine offenen Stellen) aus Datenpunkten von der Oberfläche $\mathcal{S}$ eines dreidimensionalen Objektes zu berechnen. Dafür wird die Orientierbarkeit von $\mathcal{S}$ vorausgesetzt, was in der Praxis in der Regel keine Einschränkung bedeutet.

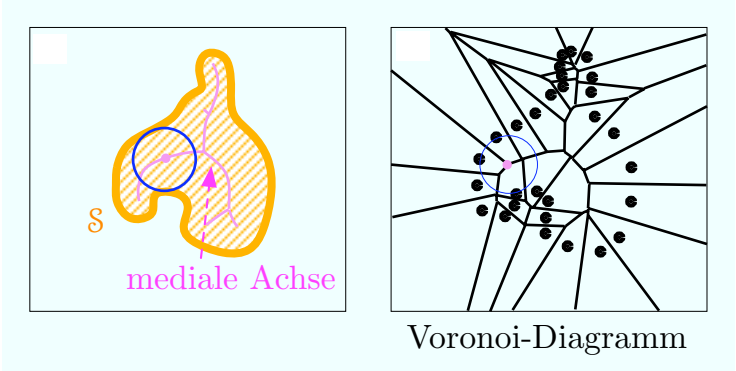
Die Idee ist dabei, die Fläche („crust“) als das Grenzgebiet zwischen einer Schar von einhüllenden Kugeln und einer Schar von aufspannenden Kugeln innerhalb der Fläche zu konstruieren. Die Kugeln sind durch die sogenannte *Mediale Achsen-Transformation* gegeben. Die Rekonstruktion der Fläche erhalten wir durch eine approximative Berechnung der inversen Transformation.

Die Mediale Achsen-Transformation kann durch die Kugeln approximiert werden, die bestimmte Voronoi-Knoten (die *Pole*) der Datenpunkte X als Kugelmittelpunkte haben und die Datenpunkte berühren, die den Polen am nächsten sind. Eine Approximation der Fläche erhalten wir, indem wir „gewichtete Voronoi-Gebiete“ der Pole berechnen und diejenigen Kanten dieser Gebiete ausgeben, die innere und äußere Pole trennen.

5.1 Mediale Achsen-Transformation und Voronoi-Diagramme

Zur Vereinfachung werden sei angenommen, dass die zu rekonstruierende Fläche $\mathcal{S}$ in einem hinreichend großen, offenen und beschränkten Volumen $\mathbb{V}^3 \subset \mathbb{R}^3$ enthalten ist. Amenta et al. schlagen in [1] vor, $\mathbb{V}^3$ als um den Faktor 5 vergrößerte „bounding box“ von X zu wählen.

Die **Mediale Achsen-Transformation** (MAT) von $\mathcal{S}$ ist die Menge aller Kugeln in $\mathbb{V}^3$, in deren Inneren keine Punkte von $\mathcal{S}$ enthalten sind und die in keiner anderen Kugel mit dieser Eigenschaft vollständig enthalten sind. Die Mittelpunkte dieser Kugeln bilden die mediale Achse, wie im linken Bild (aus [1]) angedeutet ist.



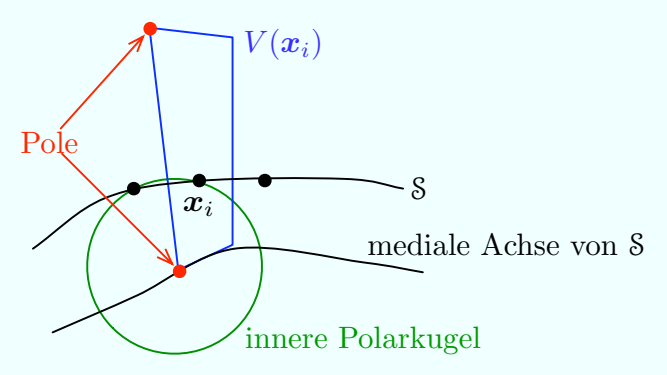
Voronoi-Diagramm

Im rechten Bild ist das **Voronoi-Diagramm** der Datenpunkte X zu sehen, also die Unterteilung $\{V(\boldsymbol{x}_1), ..., V(\boldsymbol{x}_n)\}$ von $\mathbb{V}^3$ in einzelne **Voronoi-Gebiete** $V(\boldsymbol{x}_i)$, die

diejenigen Punkte $\boldsymbol{y} \in \mathbb{V}^3$ enthalten, die dem jeweiligen $\boldsymbol{x}_i$ am nächsten liegen. Die $V(\boldsymbol{x}_i)$ sind Polyeder, deren Eckpunkte **Voronoi-Knoten** heißen. Eine **Voronoi-Kugel** ist eine Kugel, deren Mittelpunkt ein Voronoi-Knoten $\boldsymbol{v}$ ist und auf deren Rand alle Datenpunkte liegen, deren Voronoi-Gebiete den Knoten $\boldsymbol{v}$ enthalten. Dies motiviert die Idee, die Voronoi-Kugeln zur Approximation von MAT($\mathcal{S}$) zu verwenden, denn wenn die Datenpunkte X auf $\mathcal{S}$ liegen, so sind „die meisten“ Voronoi-Kugeln in MAT($\mathcal{S}$) enthalten.

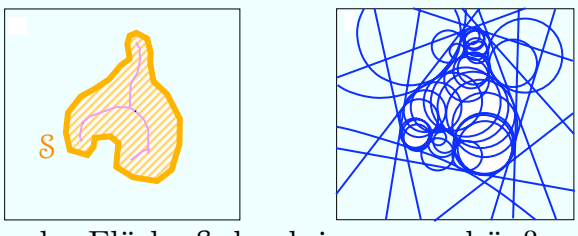
5.2 Approximation der MAT durch Polarkugeln

Liegen die Datenpunkte X sehr dicht beieinander, so sind die Voronoi-Gebiete $V(\boldsymbol{x}_i)$ sehr schlank und stehen ungefähr senkrecht auf der Fläche $\mathcal{S}$. Dies kommt daher, dass die Gebiete entlang der Oberfläche von $\mathcal{S}$ durch die Nähe der anderen Datenpunkte beschränkt sind. Senkrecht zu $\mathcal{S}$ ist das Gebiet auf der Außenseite von $\mathcal{S}$ durch den Rand von $\mathbb{V}^3$ beschränkt, und auf der Innenseite kann sich das Gebiet nicht weiter als bis zu medialen Achse ausdehnen, da $\boldsymbol{x}_i$ dort aufhört, der nächste Punkt zu sein. Als **Pole** von $\boldsymbol{x}_i$ bezeichnen wir die jeweils am weitesten von $\boldsymbol{x}_i$ entfernten Voronoi-Knoten innerhalb bzw. außerhalb von $\mathcal{S}$.



Man kann zeigen, dass die Menge P der Pole eine gute Approximation der medialen Achse bildet, wenn die Datenpunkte dicht genug liegen.

Dies führt zu der Idee, MAT($\mathcal{S}$) durch endlich viele Voronoi-Kugeln zu approximieren, deren Mittelpunkte Pole sind. Diese Kugeln nennen wir **innere** bzw. **äußere Polarkugeln**, je nach der Lage des Pols. Das Bild aus [1] zeigt rechts die Schar der $\mathcal{S}$ einhüllenden äußeren Polarkugeln und der $\mathcal{S}$ aufspannenden inneren Polarkugeln.



Approximation der Fläche $\mathcal{S}$ durch innere und äußere Polarkugeln

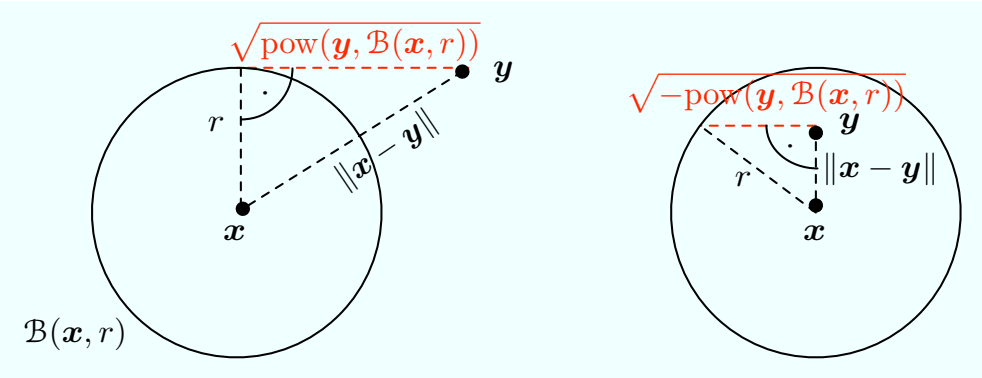
Die Berechnung der Pole ist einfach. Für einen Datenpunkte $\boldsymbol{x}_i$ wählt man den ersten Pol $\boldsymbol{p}_1$ als den Knoten von $V(\boldsymbol{x}_i)$, der am weitesten von $\boldsymbol{x}_i$ entfernt ist. Der zweite Pol $\boldsymbol{p}_2$ muss auf der gegenüberliegenden Seite von $\boldsymbol{p}_1$ liegen, so dass man $\boldsymbol{p}_2$ als denjenigen Knoten von $V(\boldsymbol{x}_i)$ wählt, der am weitesten von $\boldsymbol{x}_i$ entfernt ist *und* die Bedingung $(\boldsymbol{x}_i - \boldsymbol{p}_1) \cdot (\boldsymbol{x}_i - \boldsymbol{p}_2) < 0$ erfüllt. Mit diesem Verfahren ist zwar gewährleistet, dass die Pole $\boldsymbol{p}_1$ und $\boldsymbol{p}_2$ auf verschiedenen Seiten von $\mathcal{S}$ liegen, aber es ist noch nicht bekannt, *welcher* der beiden der innere bzw. äußere Pol ist.

5.3 Innere und äußere Pole bestimmen

Um ausgehend von der approximierten MAT von $\mathcal{S}$ zu einer Polyederfläche zu gelangen, die $\mathcal{S}$ approximiert, werden Voronoi-Diagramme für die Kugeln der MAT benötigt. Um Voronoi-Diagramm für Kugeln zu definieren, wird ein geeignetes Maß für den Abstand von Punkten zu Kugeln benötigt. Die Idee hinter der folgenden Definition ist es, die Kugeln als durch ihren Radius gewichtete Punkte aufzufassen. Die **Power-Distanz** eines Punkte $\boldsymbol{y} \in \mathbb{V}^3$ zu einer Kugel $\mathcal{B}(\boldsymbol{x}, r)$ ist durch

$$\mathrm{pow}(\boldsymbol{y}, \mathcal{B}(\boldsymbol{x}, r)) = \|\boldsymbol{x} - \boldsymbol{y}\|^2 - r^2$$

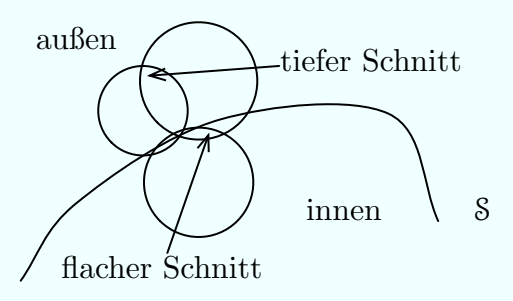
definiert. Verwendet man für die Definition des Voronoi-Diagramms anstelle des euklidischen Abstands die Power-Distanz pow, so spricht man von einem **Power-Diagramm**, entsprechend übertragen sich die anderen Begriffe. Das Bild veranschaulicht die Power-Distanz für Punkte innerhalb (rechts) und außerhalb (links) von $\mathcal{B}(\boldsymbol{x}, r)$.



Die Power-Distanz hat gegenüber dem euklidischen Abstandsmaß den Vorteil, dass eine Power-Diagramm genau wie das gewöhnliche Voronoi-Diagramm in Polyeder zerfällt. Man kann also zu seiner Berechnung und Handhabung im Wesentlichen die gleichen Algorithmen einsetzen wie für die Voronoi-Diagramme.

Es stellt somit kein großes Problem dar, die Power-Diagramme der Pole P von X zu berechnen, mit deren Hilfe man P aufteilen kann in die Pole P_i, die innerhalb von $\mathcal{S}$ liegen, und die Pole P_a, die außerhalb von $\mathcal{S}$ liegen. Dafür definiert man einen Graphen, dessen Knoten die Power-Gebiete repräsentieren und von denen je zwei

mit einer Kante verbunden werden, wenn die entsprechenden Power-Gebiete eine gemeinsame Seitenfläche haben oder zu den beiden Polen des selben Punktes $\boldsymbol{x}_i$ gehören. Als erstes werden nun diejenigen Pole, die auf dem Rand von $\mathbb{V}^3$ liegen, als „außen“ markiert. Nun wird die Orientierung an die benachbarten Knoten des Graphen weitergegeben, bis alle Pole als „innen“ oder „außen“ markiert wurden. Für eine Pol $\boldsymbol{p}$, der „außen“ ist, wird ein benachbarter Pol $\boldsymbol{q}$ ebenfalls „außen“, wenn die Polarkugeln sich tief schneiden, also wenn der Winkel an den Schnittpunkte groß ist. Entsprechend wird $\boldsymbol{q}$ als „innen“ markiert, wenn der Schnittwinkel flach ist. Dieser Regel liegt die Annahme zugrunde, dass sich bei hinreichend dichten Datenpunkten X die äußeren Polarkugeln stark überschneiden, eine innere und eine äußere Polarkugel sich gar nicht oder in einem kleinen Winkel schneiden, wie das folgende Bild illustrieren soll:

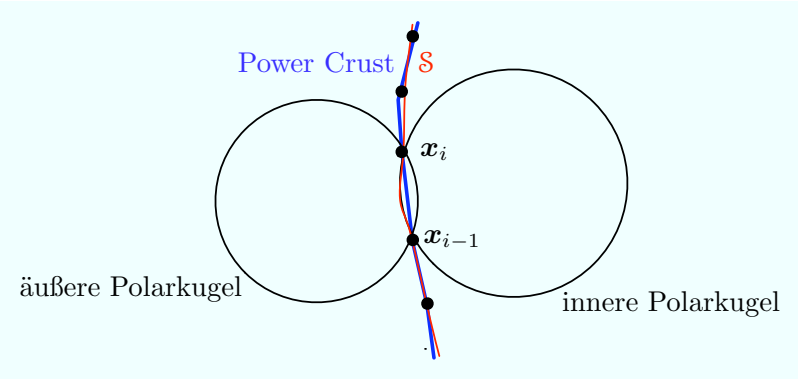


Wenn man nicht davon ausgehen kann, dass die Daten dicht genug liegen, so wird diese Entscheidungsregel abgeändert. Jedem Pol werden zwei Wahrscheinlichkeiten zugeordnet, eine dafür, dass er innen liegt, und eine dafür, dass er außen liegt. Die Pole werden dann der Reihe nach abgearbeitet, wobei zuerst die Pole markiert werden, bei denen eine der beiden Wahrscheinlichkeiten 1 ist und die andere 0. Für die übrigen Pole werden die Wahrscheinlichkeiten aus den Winkeln bestimmt, die diese mit ihren Nachbarn und deren zugehörigen Datenpunkten einschließen. Ein großer Winkel deutet darauf hin, dass der Pol auf der anderen Seite von $\mathcal{S}$ liegt wie sein Nachbar. Eine ausführliche Beschreibung dieser Heuristik findet man in [1].

5.4 Power Crust

Man stellt fest, dass die meisten Punkte im Inneren des von $\mathcal{S}$ umschlossenen Gebietes innerhalb der Vereinigung der inneren Polarkugeln liegen *und* außerhalb der Vereinigung der äußeren Polarkugeln. Im Power-Diagramm der Polarkugeln liegen diese Punkte also in Power-Gebieten, die zu inneren Polen gehören. Entsprechend liegen die meisten Punkte außerhalb von $\mathcal{S}$ in Power-Gebieten von äußeren Polen.

Eine innere Polarkugel schneidet eine äußere - wenn überhaupt - in einem sehr kleinen Winkel, da sie größtenteils im Inneren der Fläche liegt und die äußere Polarkugel größtenteils außerhalb der Fläche. Die Polyederseiten, die innere Power-Gebiete von äußeren Power-Gebieten trennen, liegen also nahe bei der Grenzfläche zwischen inneren und äußeren Polarkugeln.

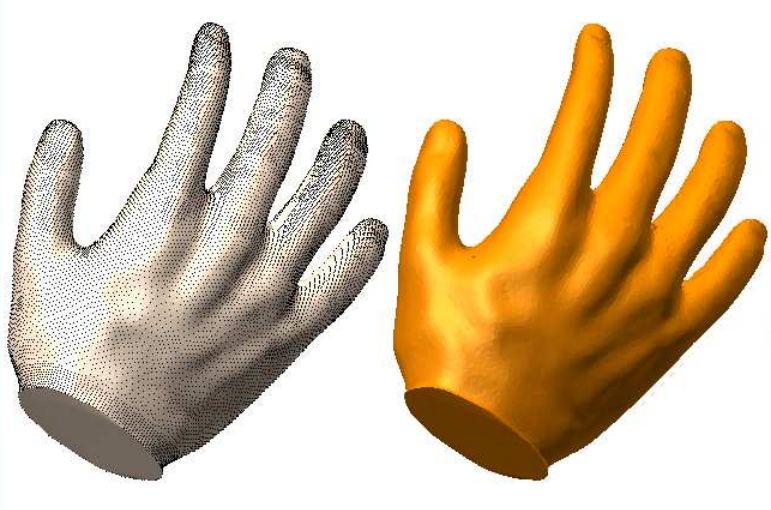


Da die Polarkugeln aber gerade so konstruiert sind, dass sie näherungsweise eine einhüllende (außen) bzw. aufspannende (innen) Kugelschar für $\mathcal{S}$ bilden, folgt, dass die Polyederseiten, die innere und äußere Power-Gebiete trennen, nahe bei der Fläche $\mathcal{S}$ liegen. Die Polygonfläche, die von diesen Polyederseiten gebildet wird, heißt **Power Crust**.

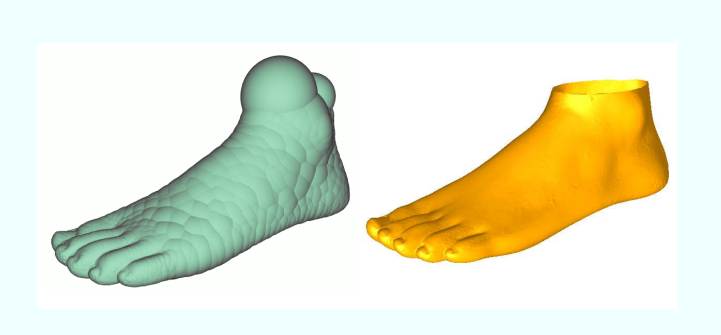
Die Power Crust-Fläche approximiert nicht nur $\mathcal{S}$, sondern interpoliert auch die Datenpunkte X, da alle $\boldsymbol{x}_i$ auf dem Rand ihrer inneren bzw. äußeren Polarkugel liegen und somit im Schnitt der beiden. Der Schnitt zweier Kugeln wiederum liegt immer auf der Randfläche, die ihre jeweiligen Power-Gebiete trennt. Die $\boldsymbol{x}_i$ sind im Allgemeinen jedoch keine Knoten der Polyeder, aus denen sich die Power Crust-Fläche zusammensetzt.

5.5 Resultate

Das Bild zeigt eine Punktwolke (links) und die vom Power Crust-Algorithmus rekonstruierte Fläche (rechts).



Der Power Crust-Algorithmus erzeugt immer geschlossene Flächen, auch wenn die ursprüngliche Fläche offene Stellen hatte. Im linken Bild sieht man die Schar der inneren Polarkugeln, die die Fläche approximieren. Das rechte Bild zeigt die Power Crust-Fläche, nachdem mit einer Heuristik an den Stellen Löcher in die Fläche eingefügt wurden, an denen die Polarkugeln „zu groß“ sind.



Sämtliche Bilder in diesem Abschnitt stammen aus [1].

6 Ausblick

Die hier vorgestellten Algorithmen (und viele andere) berechnen aus einer gegebenen Punktwolke eine Darstellung aus polygonalen Flächen. In dem Foliensatz [11] finden sich viele Bilder mit Resultaten der hier behandelten und ähnlicher Algorithmen. Darüberhinaus findet man dort auch eine Vielzahl von Literaturverweisen zum Thema. Eines der wenigen Bücher zu dem Thema ist das Buch [2] von Brunnet, Hamann, Müller und Linsen, das auch viele weiterführende Aspekte des Themas behandelt.

Neben der Darstellung als Polygonfläche gibt es noch andere Ansätz zur Darstellung von Punktwolken. In seiner Dissertation [8] schlägt Linsen vor, die Punktwolke durch „Dreiecksfächer“ im jeden Punkt darzustellen. Diese Fächer bilden nicht unbedingt eine zusammenhängende Fläche, sehen aber in der Praxis in der Regel so aus. Dieser Ansatz bietet insofern eine Vereinfachung, als dass man keine Vorberechnung zur Erzeugung der Dreiecksfläche mehr benötigt. Nachteilhaft ist, dass häufige Neuberechnungen der Fläche stattfinden müssen und die Fläche „strukturärmer“ ist als eine Polygonfläche.

Eine weitere Alternative zu Polygonflächen ist die Darstellung der rekonstruierten Fläche als B-Spline-Fläche. Dafür muss aus den Messdaten ein Spline-Kontrollnetz generiert werden, das eine gewisse Regelmäßgkeit besitzt. Eck und Hoppe geben in [3] ein Verfahren zur Rekonstruktion mit Splines über Viereckснetzen an, und in [4] stellen He und Qin ein Verfahren mit Splines über Dreiecksnetzen vor. Letzteres hat den Vorteil, dass es Flächen beliebigen topologischen Typs darstellen kann und die resultierende Spline-Fläche eine beliebig hohe vorgegebene Differenzierbarkeitsordnung haben kann. Eine solche Darstellung ist besser für eine mathematische Analyse geeignet als eine Darstellung durch Dreiecksflächen, allerdings ist die Darstellung durch Splines in der Regel mit einen sehr viel höheren Rechenaufwand verbunden.

Li beschreibt in [7] ein Verfahren, das die Rekonstruktion der Oberfläche eines Objektes nur mit Hilfe eines Projektors und einer Digitalkamera erlaubt. Dabei werden mit dem Projektor strukturierte Muster auf das Objekt projiziert, die es dem Algorithmus ermöglichen, topologische Information über das Objekt zu gewinnen. Mit nur wenigen Aufnahmen aus verschiedenen Perspektiven kann dann ein dreidimensionales Modell der Oberfläche errechnet werden. Das Verfahren ist hochautomatisiert und ist somit auch ohne Kenntnis des zugrundeliegenden Algorithmus und dessen Eingabeparametern anwendbar.

Literatur

[1] Nina Amenta, Sunghee Choi, Ravi Krishna Kolluri: *The Power Crust*, Solid Modeling Applications, 2001.

[2] Guido Brunett, Bernd Hamann, Heinrich Müller, Lars Linsen (Eds.): *Geometric Modeling for Scientific Visualization* Springer, 2004.

[3] Matthias Eck, Hugues Hoppe: *Automatic reconstruction of B-spline surfaces of arbitrary topological type*, Computer graphics and interactive techniques, 1996.

[4] Ying He, Hong Qin: *Surface Reconstruction with Triangular B-splines*, Geometric Modeling and Processing, 2004.

[5] Jan Hohe: *Entwicklung von Bildverarbeitungsmethoden zur Analyse der Gelenkflächengröße und -krümmung sowie der chondralen Signalintensität aus magnetresonanztomographischen Bildern*, Dissertation, Medizinische Fakultät, LMU München, 2001.

[6] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle: *Surface reconstruction from unorganized points*, Computer Graphics, 26, Nr. 2, 1992.

[7] Hao Li: *Rekonstruktion farbiger Objekte aus strukturiert beleuchteten Ansichten*, Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe, 2005.

[8] Lars Linsen: *Point Cloud Representations*, Technical Report 2001-3, Fakultät für Informatik, Universität Karlsruhe, 2001.

[9] William E. Lorensen, Harvey E. Cline: *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*, Computer Graphics, 21, Nr. 4, 1987.

[10] Christoph Schommer (Ed.): *Der Patient im Spiegel der heutigen Zeit*, Seminarband Universität Potsdam, 2003.

[11] Rafael Straub: *Netze und Punktwolken*, Folien zur Vorlesung im WS 04/05, http://i33www.ira.uka.de/vorlesungen/nup.html

Index

Abstand, 4
    orientierter, 10, 12
algebraische Fläche, 4
Algorithmus
    Hoppe, 9
    Marching Cubes, 6, 13
    Power Crust, 14
Ausgleichsebene, 11

CT, 3, 6

dicht
    ε-, 9
Dreiecksfläche, 4

Fehlerquadrate, 11
Fläche
    algebraische, 4
    implizite, 4
    parametrische, 4

Hoppes Algorithmus, 9

implizite Fläche, 4

Knochenimplantat, 3
Kugel, 5

Laserscanner, 3

Marching Cubes-Algorithmus, 6, 13
mediale Achse, 5
mediale Achsen-Transformation (MAT), 14
minimalinvasive Operation, 3
MRT, 3, 6

Normalenvektor, 4

orientierbar, 4
orientierter Abstand, 10, 12
Orientierung, 4

parametrische Fläche, 4
physikalische Simulationen, 3
Pol, 15
Polarkugel, 15
Polygonfläche, 4
Power Crust, 18
Power Crust-Algorithmus, 14
Power-Diagramm, 16
Power-Distanz, 16
Punktwolke, 5

Rapid Prototyping, 3
Rekonstruktion, 5
Riemann-Graph, 11

Schnittbild, 6
Simulationen, 3
Sphäre, 5
Splines, 20

Tangentialebene, 4
Triangulierung, 4

Umgebung, 10

virtuelle Operationsplanung, 3
Voronoi
    -Diagramm, 14
    -Gebiet, 14
    -Knoten, 15
    -Kugel, 15

wasserdicht, 14