Lehrstuhl für Mathematik mit Schwerpunkt
Digitale Bildverarbeitung

Bachelorarbeit

Gravur-Erkennung durch integrale Invarianten

Laura Anolick

Prüfer
Prof. Dr. Tomas Sauer

19. Februar 2020
(Version 1.0)

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Erkennung von Gravuren in Objekten. Um die Gravuren zu bestimmen, werden integrale Invarianten verwendet, da diese besonders robust gegenüber Störungen sind. Das Hauptobjekt dieser Arbeit ist eine aztekische Knochenflöte. Dieses Objekt wurde zunächst eingescannt und in Octave eingelesen. Dadurch sind die Punkte und die daraus erzeugten Dreiecks-Faces bekannt, welche ein geschlossenes Mesh ergeben. Das Problem beim Betrachten der Knochenflöte ist, dass diese Vertiefungen mit dem bloßen Auge sehr schlecht bis gar nicht erkennbar sind. Hebt man in einem anderen Programm, wie zum Beispiel MeshLab, die Ritzung farblich hervor, erkennt man, dass diese Gravur nicht sehr tief ist. Das führt wiederum dazu, dass die Ritzung im Plot nicht erkennbar ist. Im Zuge dieser Arbeit wird ein neuartiger Winkel-Deskriptor vorgestellt, der es nun ermöglicht, diejenigen Faces hervorzuheben, welche von der Ritzung betroffen sind. Genauer werden diejenigen Faces markiert, die am Übergang der Gravur zum relativ glatten Teil der Oberfläche liegen.
Das Resultat dieser Arbeit ist eine Software, die es ermöglicht, die Gravur in gescannten Objekten beliebiger Form zu erkennen und anschließend auszugeben.

Schlüsselwörter: integrale Invariante, Winkel-Deskriptor, Volumen-Deskriptor

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Aztekische Knochenflöte

Das Linden-Museum in Stuttgart stellt in der aktuellen Landesausstellung vom 12.10.2019 bis 03.05.2020 viele aztekische Kunstobjekte aus. Darunter befindet sich auch ein kleiner Knochen, der reich mit Einritzungen verziert ist und wohl als Flöte genutzt wurde. Die Gravuren sind mit bloßem Auge jedoch kaum zu erkennen, da sie nicht sehr tief sind und sich zudem farblich nicht von der restlichen Flötenoberfläche absetzen. Das Ziel dieser Arbeit ist die Kenntlichmachung eben dieser Ritzungen. Dazu werden zunächst die theoretischen Grundlagen zur Oberflächenerkennung von Objekten betrachtet, die ein umfangreiches Portfolio an mathematischen Werkzeugen bieten. Eine zusätzliche Beschreibungsmöglichkeit wird mit dem Winkel-Deskriptor eingeführt und in das Portfolio eingeordnet. Anschließend erfolgt die praktische Anwendung dieses Deskriptors schrittweise über einfach zu handelnde Objekte bis schließlich ein Programm zur Verfügung steht, mit dem Gravuren in beliebigen gescannten Objekten herausgearbeitet werden können. Das wird auch anhand der aztekischen Knochenflöte demonstriert.

2 Einordnung der Arbeit

2.1 Integrale Invarianten und Deskriptoren

Integrale Invarianten sind Hilfsmittel, um Krümmungen und damit den Oberflächenverlauf von Objekten zu schätzen. Ein wesentlicher Bestandteil integraler Invarianten ist die Integration, welche sich glättend[1] auswirkt und nützliche Stabilitäts- und Robustheitsaspekte von Algorithmen ohne Vorverarbeitung ermöglicht (Pottmann et al., 2006 S.1 (Abstract)). Integriert werden räumliche Funktionen über bewegliche Bereiche, die an Oberflächenpunkten, den Kernen, zentriert sind (Huang et al., 2006 2.1). So erhält man die Glättung zum Beispiel für gegebene 3D-Daten über geeignet kleine Kerndomänen, also Kreis- bzw. Kugelradien, wobei jedem Kernstandort integrale Invarianten zugeordnet sind. Diese Invarianten können aufgrund ihrer geometrischen Bedeutung als Krümmungsschätzer dienen, welche robust gegenüber Rauschen sind und ein Multiskalenverhalten[2] aufweisen.[3] Zusammenfassend weisen integrale Invarianten folgende nützliche Eigenschaften auf (Pottmann et al., 2006):

[1] Mit Hilfe der Glättung können Unebenheiten auf der Oberfläche eines Objekts retuschiert werden. Dies geschieht dadurch, dass einige Punkte, für die der Deskriptor ein schwaches Signal gegeben hat, weggelassen werden. Das Ergebnis ist ein Überblick, der das große Ganze zeigt und auf Feinheiten verzichtet. Diese weggelassenen Feinheiten nennt man Rauschen.

[2] Unter Multiskalenverhalten versteht man die Anpassungsfähigkeit des Krümmungsschätzers an die Wahl der Auflösung (Pottmann et al., 2006, S.1-2).

[3] Diese integralen Invarianten erhält man zu jedem Kernstandort, wenn man für gegebene 3D-Daten verschiedene Funktionen über geeignet kleine Kerndomänen integriert (Pottmann et al., 2006 S.1). Diese können laut Pottmann et al. (2007) aber noch leichter berechnet werden durch Faltung und anschließender Anwendung von FFT.

- Unabhängigkeit von der Oberflächen- bzw. Kurvenbeschaffenheit
- Robustheit bei der Berechnung von Formeigenschaften (aufgrund der geometrischen Bedingungen)
- Skaleninvarianz[4]
- Koordinatenunabhängigkeit in Mannigfaltigkeiten[5]

Die integralen Invarianten werden insbesondere zur Merkmalserkennung verwendet. Das gilt sowohl für Oberflächenmerkmale wie auch für Eigenschaften von räumlichen Kurven (Clarenz t al., 2004 a,b; Qi Xing et al., 2006).

In vielen Fällen besitzen integrale Invarianten, die am Grenzpunkt p einer Domäne D ausgewertet werden, die Form eines der folgenden Faltungsintegrale[6]:

$$I_r(\mathbf{p}) = \int_{\mathbf{p}+rB} g(x)w(p-x)dx, \tag{1}$$

$$I_r^{'}(\mathbf{p}) = \int_{\mathbf{p}+rS} g(x)w(p-x)dx, \tag{2}$$

wobei I für integrale Invariante steht, der Index r den Kernradius angibt, B die Einheitskugel, w die Gewichtsfunktion und g eine der Domäne zugeordnete Funktion ist. g kann zum Beispiel die Abstandsfunktion oder die Indikatorfunktion[7] sein. Gleichung (1) beschreibt die Integration über alle Punkte x innerhalb der Kugel um Mittelpunkt $\mathbf{p}$ mit Radius r. Die Variable S in Gleichung (2) steht für den Kreis. Somit wird hier über alle Punkte in der Kreisfläche um Mittelpunkt $\mathbf{p}$ um Radius r integriert.
In Abhängigkeit von $n \in \{2, 3\}$ ergeben sich für dx folgende Ausprägungen:

- *n=2 und Integration über **p**+rS:* Bogenlänge über **p**+rS, siehe Gleichung (2)
- *n=3 und Integration über **p**+rB:* Volumenintegral über **p**+rB, siehe Gleichung (1)
- *n=2 und Integration über eine Kreisscheibe:* Inhalt der Kreisfläche
- *n=3 und Integration über eine Kugel:* Oberflächeninhalt der Kugel

[4]Nach Huang et al.,ist die Berechnung der Oberflächenrauheit sehr zeitaufwendig. Gleichzeitig führt diese für kleine Radien nicht zur Glättung.
[5]Unter einer Mannigfaltigkeit versteht man in der Mathematik einen topologischen Raum, der lokal dem euklidischen Raum $\mathbb{R}^n$ gleicht. Ein Beispiel wäre eine Sphäre.
[6]"Das Faltungsintegral ist von Bedeutung, wenn bei der Fourier- oder Laplace-Transformation die Multiplikation zweier Ausdrücke in der Zeitebene (Bildebene) durch eine mathematische Vorschrift in der zugeordneten Bildebene (Zeitebene) ersetzt werden soll." (Plaßmann)
[7]Die Indikatorfunktion nimmt den Wert 1 an, wenn der Punkt x (Integrationsvariable) innerhalb der Domäne liegt, andernfalls 0.

Integrale Invarianten werden als univariate[8] Funktion des Kernradius betrachtet, also diejenige Änderung des Werts der integralen Invarianten, die auftritt wenn der Kernradius variiert wird (Pottmann et al., 2006, S.2).
Ist der Radius einer integralen Invarianten $I_r(\mathbf{p})$ oder $I_r^{'}(\mathbf{p})$ konstant, so ist die Invariante die Funktion des Punktes $\mathbf{p}$ (Pottmann et al., 2006, S.2-3). Wichtig hierbei ist, dass $\mathbf{p}$ in der Oberfläche enthalten sein muss, da ansonsten die geometrischen Eigenschaften nicht mehr greifen.
Eine weitere Möglichkeit eine Invariante zu definieren, ist über die Länge des Kreisbogens, der innerhalb der Domäne D liegt: $D \cap (\mathbf{p} + rS)$ (Pottmann et al., 2006, S.3-4):

$$CA_r(p) := \int_{\mathbf{p}+rS} 1_D \, dx = \frac{d}{dr} \int_{\mathbf{p}+rB} 1_D \, dx = \frac{d}{dr} A_r(p), \tag{3}$$

wobei $\kappa = f''(1 + (f')^2)^{-\frac{3}{2}}$ gilt.
Man erhält $CA_r(p)$, wenn man in Gleichung (2) $g(x)$ durch die Indikatorfunktion ersetzt, sowie die Gewichtsfunktion $w(p - x)$ durch die Konstante 1 (Pottmann et al., 2007). $CA_r(p)/r$, der Quotient von (3) und der Kernelgröße r heißt *Connolly-Funktion*, welche auch bei nur abschnittsweise glatten Kurven gute Ergebnisse liefert.
Mit $CA_r = \pi r - \kappa r^2 + O(r^3)$ erhält man aus Gleichung (3) durch Integration:

$$A_r = \frac{\pi}{2} r^2 - \frac{\kappa}{3} r^3 + O(r^4). \tag{4}$$

A_r stellt den Flächen-Deskriptor dar, den man aus einer integralen Invarianten sehr einfach herleiten kann. Wird eine integrale Invariante für einen festen Radius r berechnet, erhält man also einen Deskriptor, der die Berechnung stark vereinfacht. Ein Deskriptor beschreibt ein Formmerkmal. Dadurch umgeht man die aufwendige Berechnung von integralen Invarianten. Der Rechenaufwand verringert sich wesentlich bei gleichbleibender Aussagekraft. Zusätzlich erfolgt auch noch eine Glättung, falls der Radius geeignet gewählt wird (Qi Xing et al., 2006).

Da der tatsächliche Oberflächenverlauf unbekannt ist, können für die Berechnung der Deskriptoren nur die gescannten Punkte und die daraus resultierenden Faces[9] verwendet werden. Falls von einem Punkt sehr viele Kanten ausgehen, ist selbst die Berechnung des Volumen-Deskriptors äußerst aufwendig. Fällt der Kern, also der Kugelmittelpunkt, mit dem Mittelpunkt einer Kante zusammen, ist der Volumen-Deskriptor (siehe 3.4) sehr einfach mittels der beiden benachbarten Dreiecksfaces zu berechnen. In diesem Fall erhält man aus dem Winkel zwischen den beiden angrenzenden Faces jedoch das gleiche Resultat wie mit dem Volumen-Deskriptor.

[8]Eine univariate Funktion ist nur von einer Variable abhängig.
[9]Ein Face ist ein Dreieck im Oberflächengitter (Mesh). Die Eckpunkte ergeben sich hier als die gescannten Punkte eines Objects.

2.2 Grundlegende Arbeiten

Ausgangspunkt der folgenden Überlegungen sind insbesondere die beiden Arbeiten *Integral Invariants for Robust Geometry Processing* (Pottmann et al., 2006) und *Principal curvatures from the integral invariant viewpoint* (Pottmann et al., 2007), die gute Ansätze zur Extraktion von Oberflächenmerkmalen liefern:
In *Integral Invariants for Robust Geometry Processing* werden verschiedene Geometrie-Deskriptoren[10] untersucht, die für die Erkennung persistenter Merkmale nützlich sind. Es wird dort auf die numerische Robustheit der integralen Invarianten hingewiesen, die ja die Basis der verschiedenen Deskriptoren ist. Für das Problem der Gravurerkennung hat sich nach Pottmann der Volumen-Deskriptor als besonders geeignet gezeigt, da er die größte Stabilität aller Deskriptoren aufweist (Pottmann et al., 2006, S.8ff).

In *Principal curvatures from the integral invariant viewpoint* wird die Extraktion von Krümmungsinformationen auf Oberflächen durch integrale Invarianten, die auf einer Hauptkomponentenanalyse der lokalen Nachbarschaften auf verschiedene Kugelgrößen basiert, hergeleitet. Diese Hauptkomponentenanalyse (PCA) ermöglicht es, die Hauptkrümmungen[11] zu einer bestimmten Skala zu bestimmen. Das Augenmerk dabei liegt auf Geometrie-Deskriptoren, da diese robust gegen Rauschen sind und Multiskalenverhalten zeigen (Pottmann et al., 2007, Schlussfolgerung).

In der folgenden Arbeit wird die Extraktion von Oberflächenmerkmalen, aufbauend auf diese beiden Arbeiten, um einen zusätzlichen Deskriptor, den Winkel-Deskriptor, erweitert.

3 Methodik

Im Folgenden wird ein Überblick über integrale Invarianten zur Gewinnung von Oberflächeninformationen gegeben, so wie auch der daraus abgeleiteten Deskriptoren zur vereinfachten Berechnung. Dazu wird in Verfahren mittels integralen Invarianten unterschieden, die eine vorherige Diskretisierung benötigen, sowie Verfahren, die ohne diese funktionieren. Daraufhin werden ausgewählte Deskriptoren besprochen. Das Ergebnis dieses Kapitels ist die Einführung eines Winkel-Deskriptors in Anlehnung an den Volumen-Deskriptor von Pottmann et al. (2006), welche in einem Sonderfall sogar übereinstimmen. Da der Winkel-Deskriptor jedoch universell einsetzbar ist (auch bei nicht-geschlossenen Meshes), wird ihm in der Implementierung der Vorzug gegeben.

[10] Deskriptoren, die geometrische Eigenschaften wie Volumen, Flächeninhalt und Abstand verwenden.
[11] Die beiden Hauptkrümmungen stehen senkrecht zueinander.

3.1 Diskretisierung von integralen Invarianten

Integrale Invarianten ermitteln für jeden möglichen Oberflächenpunkt einen zugehörigen Integralwert, der Oberflächeninformation enthält. Diese Berechnung müsste also mit unendlich vielen Oberflächenpunkten durchgeführt werden. Bei der Anwendung auf gescannte Oberflächen beschränkt man sich notwendigerweise auf einen festen Satz von Datenpunkten. Diesen Vorgang der Reduzierung der Punktemenge nennt man Diskretisierung.
Es folgen nun verschiedene Methoden zur Berechnung integraler Invarianten, die jeweils einen anderen Aspekt der Oberflächeninformation herausheben. Diese Verfahren können nur bei diskreten Oberflächen angewendet werden, da laut Pottmann et al. (2006, S.11) eine Vorverarbeitung der Datenstrukturen, sowie intelligente Methoden zur Diskretisierung von Integralen zur Berechnung notwendig sind. Schließlich wird die Hauptkomponentenanalyse betrachtet, die ohne explizite Diskretisierung auskommt.

3.1.1 Taylor-Reihenentwicklung

Die Taylorreihenentwicklung ist ein Hilfsmittel zur Glättung. Bei diesem Verfahren umgeht man gleichzeitig die Integration.

Pottmann et al. (2006, S.3) haben ein allgemeines Vorgehen zur Berechnung der ersten Glieder der Taylorentwicklung der integralen *Invariante $I_r(\boldsymbol{p})$ für planare Kurven.* entwickelt.
Die Taylorreihenentwicklung in Bezug auf Ableitung und Krümmung sieht wie folgt aus (Pottmann et al., 2006, S.3):

$$x_2 = \frac{\kappa}{2}x_1^2 + \frac{\kappa'}{6}x_1^3 + \frac{\kappa'' - 3\kappa^3}{24}x_1^4 + O(x_1^5), \tag{5}$$

mit folgender Notation:

Für jeden Punkt $\mathbf{p}$ einer Kurve f in der Ebene wählt man diesen Punkt als Ursprung des Koordinatensystems so, dass die x_1-Achse Tangente der Kurve ist. Dann gilt $x_2 = f(x_1) = \alpha x_1^2 + \beta x_1^3 + \gamma x_1^3 + O()x_1^5$. Unter der Verwendung von $\kappa = f''(1 + (f')^2)^{-3/2}$ ergibt sich dann die Abschätzung (5).

Die integrale Invariante $I_r(\boldsymbol{p})$ zur Darstellung der Oberflächenbeschaffenheit ermittelt man mit dem gleichen Verfahren. Man erhält

$$x_3 = \frac{1}{2}(\kappa_1 x_1^2 + \kappa_2 x_2^2) + R(x_1, x_2), \tag{6}$$

wobei der Restterm $R(x_1, x_2)$ beschränkt ist und κ_1 bzw. κ_2 die zueinander senkrecht stehenden Hauptkrümmungen im Punkt P sind[12].

[12]Es gilt im Übrigen für die mittlere Krümmung H: $H = \frac{\kappa_1 + \kappa_2}{2}$, sowie für die Gaußsche Krümmung K: $K = \kappa_1\kappa_2$ (Pottmann et al., 2006, S.3).

Für einen beliebigen Punkt $x \in \mathbb{R}^3$ gibt $dist(x, \Phi)$ den vorzeichenbehafteten Abstand zur Oberfläche an. Dafür gilt folgende Reihenentwicklung:

$$dist(x, \Phi) = \frac{1}{2}(\kappa_1 x_1^2 + \kappa_2 x_2^2), \tag{7}$$

mit einem Fehlerglied, das in der dritten Potenz gegen 0 geht.
Die Taylor-Entwicklung erlaubt also die Berechnung von Deskriptoren ohne eine Integration durchzuführen. Dadurch wird die Berechnung dieser Oberflächeninformationen stark vereinfacht, indem rechenintensive Schritte teilweise vorab erledigt werden können oder sogar ganz entfallen. Das Restglied gibt den Fehler gegenüber der Integration an. Es ist zwar klein, dennoch stellt der Verzicht auf den Restterm eine hilfreiche Glättung dar, die überflüssige Oberflächeninformationen herausfiltert.

3.1.2 Fast Fourier Transform

Die Fast Fourier Transform (FFT[13]) hat den großen Vorteil der Einfachheit und wird von Pottmann et al. (2007) verwendet, um die Kugeltriangulationen hierarchisch zu verkleinern. Sie ändern dabei das einfache Verfahren von Yang et al. (2006) so ab, dass man eine schnellere Annäherung an die kontinuierliche Faltung[14] erhält. Integrale Invarianten $I_r(\mathbf{p})$ werden zum Beispiel durch eine Faltungs-Notation beschrieben (Pottmann et al., 2006, 11):

$$(g \star k)(\mathbf{p}) = I_r(\mathbf{p}) = \int_{\mathbb{R}^3} g(x)k(\mathbf{p} - x), \tag{8}$$

wobei gilt $k(x) = w(x) * 1_{rB}(x)$, k also das Produkt von Gewichtsfunktion und Indiaktorfunktion der Kugel 1_{rB} ist.
In diese Kategorie fallen die Invarianten: V_r (Volumen-Deskriptor), D_r (Deskriptor für den vorzeichenbehafteten Abstand), $D_{2,r}$ (Deskriptor für den quadratischen Abstand) und diejenigen der Hauptkomponentenanalyse (siehe 3.2).
Die Berechnung der integralen Invarianten erfordert die Approximation der kontinuierlichen Faltungsintegrale durch diskrete, um dann darauf FFT anzuwenden. Im Falle eines endlichen Satzes von Gitterpunkten mit ganzzahligen Koordinaten, geht man den anderen Weg und gewinnt durch Interpolation[15] zwischen den Gitterpunkten aus der diskreten Faltung die kontinuierliche $g(x)$:

$$g(x) = \sum_j g(j)K(x - j) \tag{9}$$

[13]FFT heißt übersetzt 'Schnelle Fourier-Transformation' und ist ein Algorithmus zur Berechnung der diskreten Fourier-Transformation (DFT) [FFT1]. FFT ist zum Beispiel ein Werkzeug in der Signalverarbeitung, um zwei Signale zu einem Signal zu verknüpfen [FFTP]. Hierbei wird trotz Polynommultiplikation und der Transformation in Stützstellendarstellung und einer Rücktransformation, durch die inverse Fourier-Transformation (inverse FFT), eine schnellere Ausführungszeit erzielt, gegenüber der direkten Multiplikation in Koeffizientendarstellung [FFTF].
[14]Eine Faltung(= Produkt von Teilfunktionen) bewirkt eine Glättung der ursprünglichen Funktion.
[15]Ziel, stetige Funktion zu finden, die durch Messwerte (zu diskreten Datenpunkten) läuft z.B.: Schätzung einer Größe zwischen Messwerten, Quelle Mathematische Software Skript Uni Passau, SoSe 2019

mit $K(\mathbf{x}) = b(x_1)b(x_2)b(x_3)$, $b(x_i)$ beschreibt die Interpolation in Richtung x_i.
Damit erhält man aus (9):

$$\begin{aligned}(g \star k)(\mathbf{p}) &= \int \sum_j g(\mathbf{j})K(\mathbf{x}-\mathbf{j})k(\mathbf{p}-\mathbf{x})d\mathbf{x} \\ &= \sum_j g(\mathbf{j}) \int K(\mathbf{y})k(\mathbf{p}-\mathbf{y}-\mathbf{j})d\mathbf{y} \\ = \sum_j g(\mathbf{j})(K \star k)(\mathbf{p}-\mathbf{j}) &= g((K \star k))(\mathbf{p}).\end{aligned}$$

Die hier auftretende Fourier-Transformation $K \star k$ kann vorab berechnet werden, so dass sich die Rechenzeit verkürzt (Pottman et al., 2006, S.11).
Der große Vorteil von FFT ist also die Einfachheit, mit der Möglichkeit rechenintensive Schritte teilweise vorab berechnen zu können, der große Nachteil ist, dass die integralen Invarianten nur an festen Gitterpunkten berechnet werden können.

3.1.3 Oktreebasiertes Verfahren

Das oktreebasierte[16] Verfahren ermöglicht die tatsächliche Auswertung von Oberflächenpunkten. Ziel hierbei ist es, den Volumen-Deskriptor zu berechnen. Wie bei FFT wird die Oberfläche in Quader gegliedert, die dann so in Würfel zerlegt werden, dass die Funktion $g(x)$ durch quadratische Terme $\tilde{g}_i$ in jedem Blattwürfel approximiert werden kann. Man betrachtet hierfür zum Beispiel die Abstandsfunktion $g(x)$ und den Volumen-Deskriptor $I_r(p)$ nach (1). Die integrale Invariante kann dabei in zwei Summanden zerlegt werden:

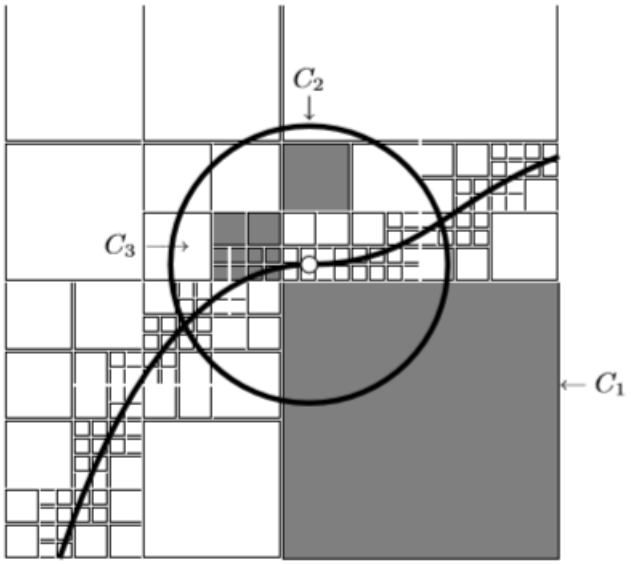
$$I_r(p) = \sum_{i:C_i \subseteq (\mathbf{p}+rB)} (\int_{C_i}) + \tag{10}$$

$$\sum_{i:M^r_{i,\mathbf{p}} \neq \emptyset \subseteq (\mathbf{p}+rB)} (\int_{i,\mathbf{p}}^{r} g) \tag{11}$$

Zur Anwendung des Oktree-Verfahrens siehe Abbildung 1.

[16] Ein *octree* ist eine Baumstruktur, in der jeder interne Knoten genau acht Kinder hat. [Oc]

Abbildung 1 : Berechnung des Volumen-Deskriptors
Berechnung des Volumen-Deskriptors $I_r(\mathbf{p})$ basierend auf der Octree-Zerlgegung von der Linie in einer Oberfläche.
Quelle: Pottmann et al., 2006, S.11



Diese Methode besteht aus drei Hauptschritten, die in einen Berechnungsalgorithmus (siehe Algorithm 1) münden:

1. Octree-Konstruktion:
   Die gescannte Oberfläche steht als Dreiecksmesh zur Verfügung. Verlaufen Faces durch einen Würfel, wird dieser in acht Blattwürfel zerlegt[17] (nach Nooruddin und Turk, 2003). $\tilde{g}_i$ gibt als Konstante an, wie weit C_i im Würfel liegt[18].

2. Vorabverarbeitung durch Approximation:
   Man betrachtet zunächst die quadratische Abstandsfunktion: $g(x) = dist(x, \delta D)^2$. Zur Berechnung wird unter anderem die Methode von Mitra et al. (2004) angewendet, wodurch für Blattwürfel C_i der quadratische Abstand $\tilde{g}_i$ approximiert werden kann (nach (11)).

3. Berechnung integraler Invarianten:
   Die Approximationsergebnisse aus Schritt 2 werden verwendet und mit der Berechnung der Integrale zusammengeführt, für die es keine Vorabverarbeitung durch Approximation gibt (10).

[17] Liegt eine Punktewolke vor, so wird die Methode von Frisken et al. (2000) verwendet.
[18] Diejenigen Würfel, die die Grenze enthalten, sind sehr klein. Die Berechnung mit der Hilfe der Tangente ist eine gute Näherung.

Algorithm 1 Berechnung des Volumen-Deskriptors basierend auf der Octree-Zerlegung

1. Initialisiere C_i mit dem Wurzelwürfel; Sei $I = I_r(\mathbf{p}) = 0$.
2. Berechne für den Würfel C_i:
   2a. Befindet sich C_i außerhalb von $\mathbf{p} + rB$ (außerhalb der Kugel), tue nichts.
   2b. Ist $C_i \subseteq (\mathbf{p} + rB)$, erhöhe den Wert von I um $\int_{C_i} \tilde{g}_i$ (was vorab berechnet wird).
   2c. Sonst:
      (α) Ist C_i ein Blattwürfel, dann erhöhe I um den Wert $\int_{C_i \cap (\mathbf{p}+rB)} \tilde{g}_i$.
      (β) Andernfalls beginne wieder bei 2. für alle 8 Kinderwürfel von C_i
3. I enthält das Ergebnis.

3.2 Gewinnung von Formeigenschaften unter Verwendung der Hauptkomponentenanalyse

Betrachten wir nun die PCA[19] über verschiedene Kernelgrößen[20]. Für kleine Kernelgrößen sind beliebige Invarianten eng mit der Krümmung verbunden, da diese die Hauptkrümmungen definieren. Eine wichtige Eigenschaft von Hauptkrümmungen ist, dass Oberflächenmerkmale, die kleiner als der Kernradius sind, also geglättet werden, quasi als Rauschen betrachtet werden (Pottmann et al., 2007, S.13-14).
Die Hauptkomponentenanalyse wird auf einen Punktesatz angewendet, nachdem für alle zugeordneten Punkte zuvor eine bestimmte Oberflächeninformation ausgewertet wird.

3.2.1 Merkmalsextraktion auf verschiedenen Skalen

Zur Erkennung von Merkmalen auf einer Oberfläche werden verschiedene Radien r_1, r_2,... verwendet. Bei einer Analyse dieser verschiedenen Skalen fällt auf, dass man für zu kleine Skalen jede Unebenheit erkennt. Ist der Radius zu groß gewählt, geht doch jegliche Krümmungsinformation der Oberfläche verloren. Das heißt, indem man verschiedene Radien kombiniert, kann man also relevante von unwichtigen Merkmalen trennen. Deshalb sollte in Hinsicht auf Robustheit kein zu kleiner Maßstab für Hauptkrümmungen gewählt werden.

3.2.2 Hauptkurven

Die PCA einer lokalen Nachbarschaft mit verschiedenen Kernelgrößen ist ein Schätzer für die Hauptrichtungen. Man betrachtet die auf die Oberfläche projizierten Vektoren. Durch Integration erhält man die Hauptkurven zur gewählten Skala r (Pottmann et al., 2007, S.14-15).

[19] Principal Component Analysis, deutsch: Hauptkomponentenanalyse

[20] PCA führt zu Geometrie-Deskriptoren, die von einem Kernel-Radius r abhängig sind und bei glatten Oberflächen zu den Krümmungen der betreffenden Oberfläche konvergieren (für $r \to 0$). Die Kernelgröße dient als Schwellengröße, die hilft Rauschen von Merkmalen zu unterscheiden (Pottmann et al. 2007).

3.3 Invarianten zu Abstandsfunktionen

Eine Abstandsfunktion oder auch Metrik 'ist eine Funktion, die je zwei Elementen des Raums einen nicht negativen reellen Wert zuordnet, der als Abstand der beiden Elemente voneinander aufgefasst werden kann'[21]. Eine Invariante zu (vorzeichenbehafteten) Abstandsfunktionen ist gegeben durch:

$$D_r(\mathbf{p}) = \int_{\mathbf{p}+rB} dist(x, \Phi)dx. \tag{12}$$

Der DeskriptorD_r ist dabei mit der mittleren Krümmung[22] und dem Volumen-Deskriptor verwandt (Pottmann et al., 2006, S. 6-7). Man betrachtet wegen der einfacheren Berechenbarkeit dazu die Taylor-Entwicklung:

$$dist(x, \Phi) = \frac{1}{2}(\kappa_1 x_1^2 + \kappa_2 x_2^2). \tag{13}$$

Ziel ist es nun, Invarianten aus der Abstandsfunktion zu gewinnen, die ohne Vorzeicheninformation auskommen, also nicht negativ sind. Dazu wählt man, wie üblich das Quadrat der Abstandsfunktion $dist(x, \Phi)$. Der entsprechende Deskriptor ist gegeben durch:

$$D_{2,r}(\mathbf{p}) = \int_{\mathbf{p}+rB} dist(x, \phi)^2 dx. \tag{14}$$

Die Taylorentwicklung von $D_{2,r}(\mathbf{p})$

$$D_{2,r}(\mathbf{p}) = \frac{4 * \Pi}{15} r^5 - \frac{\Pi}{105}(\kappa_1 - \kappa_2)^2 r^7 + O(r^8), \tag{15}$$

sowie die Taylorentwicklung des Oberflächenintegrals $SD_{2,r}(\mathbf{p})$:

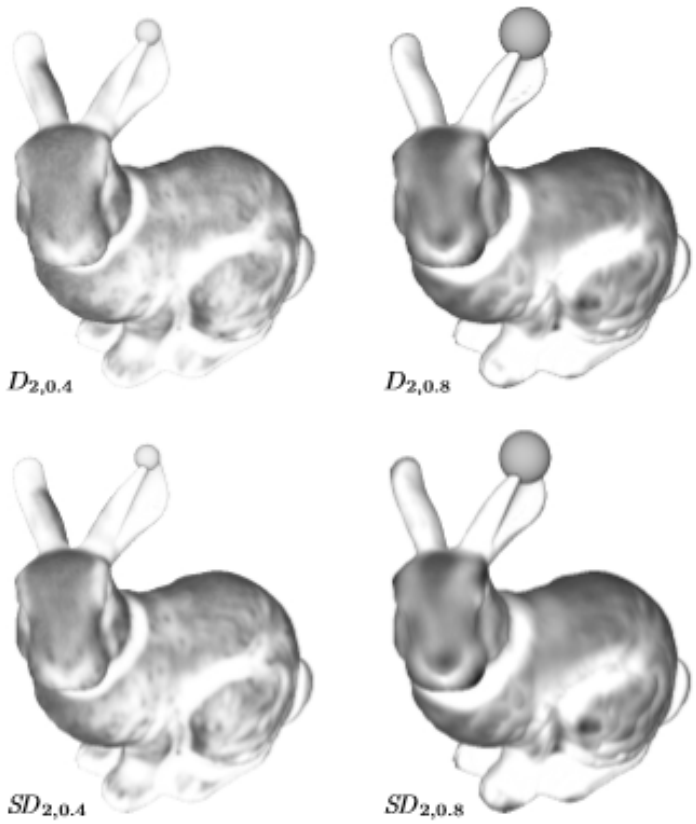
$$SD_{2,r}(\mathbf{p}) = \frac{4 * \Pi}{3} r^4 - \frac{\Pi}{15}(\kappa_1 - \kappa_2)^2 r^6 + O(r^7), \tag{16}$$

erlauben die Schätzung von $\tilde{k}^2$ und $\tilde{\tilde{k}}^2$ für die quadratische Differenz der Hauptkrümmungen. Abbildung 2 zeigt die Anwendung der Geometrie-Deskriptoren, die diesen quadratischen Abstand verwenden:

[21] [MR Forster]

[22] Im Falle einer Kugeloberfläche mit Radius r gilt für die mittlere Krümmung: $H = \frac{1}{r}$

Abbildung 2 : Geometrie-Deskriptoren, die aus dem quadratischen Abstand abgeleitet sind (Volumen und Oberflächenintegrale)
In der linken Spalte wird eine Kernelgröße von 0.4 betrachtet und rechts 0.8. Die obere Zeile wird mit Hilfe der Abstandsfunktion berechnet, während die untere Zeile das Oberflächenintegral verwendet. Man sieht hier, dass in der rechten Spalte die Kernelgröße zu groß gewählt wurde, da einige Informationen auf der Oberfläche verloren gehen und zu stark geglättet werden.
Quelle: Pottmann et al., 2006, S.7



Pottmann et al. (2006) berechnen auch das quadratische Abstandsintegral $D_{2,r}$ für eine planare (17) oder räumliche (18) Kurve c:

$$\int_{\mathbf{p}+rB} dist(x,c)^2 dx = \frac{\pi}{4} r^4 - \frac{\kappa^2 \pi}{96} r^6 + O(r^7) \tag{17}$$

$$\int_{\mathbf{p}+rB} dist(x,c)^2 dx = \frac{8 * \pi}{15} r^5 - \frac{\kappa^2 \pi}{105} r^7 + O(r^8), \tag{18}$$

wobei κ die Krümmung von c im Punkt $\mathbf{p}$ ist.

Es kann gezeigt werden, dass Invarianten, die die Abstandsfunktion verwenden, weniger

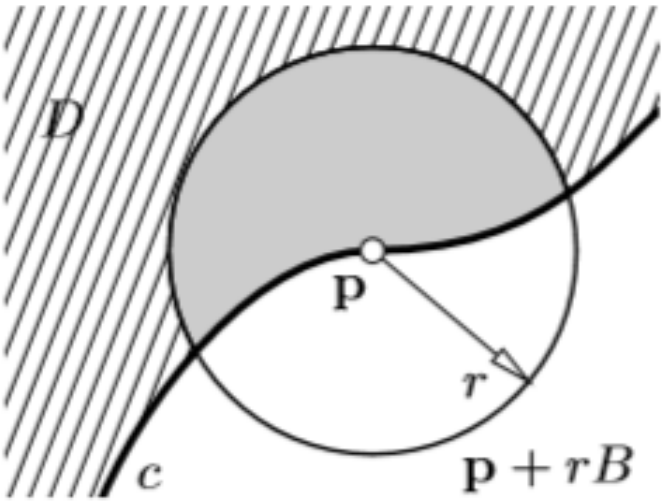
stabil sind, als diejenigen, die nur über die Indikatorfunktion integrieren (Pottmann et al., 2006, S.10).

3.4 Volumen-und Winkel-Deskriptor

3.4.1 Volumen-Deskriptor

Pottmann et al. (2007) stellen ihre grundlegende Idee zum Volumen-Deskriptor wie folgt vor: Man berechnet Größen wie Volumen oder Kovarianzmatritzen von kleinen Mengen durch die Taylor-Erweiterung, da die Punkte auf einer glatten Oberfläche verbunden sind. Diese Taylor Koeffizienten tragen die Krümmungsinformationen. Pottmann et al. (2006) stellten den Bezug auf Krümmungen durch die Analyse für Geometrie-Deskriptoren auf Basis zu Abstandsfunktionen her. Geometrie-Deskriptoren[23] von integralen Invarianten wirken wie ein Filter. So ignorieren diese zum Beispiel Details, die kleiner als eine bestimmte Schwellengröße sind (Pottmann et al., 2007, 13-16). Außerdem wird ein Geometrie-Deskriptor verwendet, da dieser Formeigenschaften erfassen kann, sowie ein Multiskalenverhalten[24] aufweist und sich daher besonders gut zur Berechnung von Hauptkurven und für die Merkmalsextraktion (Pottmann et al., 2007, S.18) eignet.

Abbildung 3 : Integrale Invariante für planare Kurven
Die unterschiedlich gefärbten 'Kreisteile' stellen gegeneinander geneigte Flächen innerhalb einer Kugel dar. Die Erweiterung des Deskriptors für planare Kurven zum Volumen-Deskriptor ergibt sich aus der Betrachtung der Kugelvolumina oberhalb bzw. unterhalb der beiden Flächen.
Quelle: Pottmann et al., 2006, S.1



[23]Geometrie-Deskriptoren sind auch für verrauschte Oberflächen sinnvoll.
[24]Multiskalenverhalten ist gut für die Erkennung persistenter Merkmale und Anpassungsfähigkeit an die Wahl der Auflösung geeignet.

Der Zusammenhang von Volumen-Deskriptor und Krümmung zeigt:

$$V_r = \frac{2\pi}{3} r^3 + \frac{\pi H}{4} r^4 + O(r^5), \tag{19}$$

mit H als mittlere Krümmung. Pottmann et al. (2006, S.8-9) beschreiben: Wenn eine Störung $\delta(\mathrm{x})$ auf eine Oberfläche Φ angewandt wird, dann kann die Änderung des Volumen-Deskriptors $V_r(p)$ als mittlere Krümmung H bezeichnet werden.

$$\Delta V_r = r^2 \pi (\bar{\delta}_r + O(\delta_{max}^2) + O(\bar{\delta} r^2)) \tag{20}$$

Des weiteren betrachtet man das räumliche-Gegenstück der Invarianten Connolly Funktion und der Flächeninvariante. Diese sind gegeben durch den Volumen-Deskriptor (15) und den Oberflächenbezeichner (16) (Pottmann et al., 2006, S.4-5):

$$V_r(p) = \int_{\mathbf{p}+rB} 1_D(x) dx. \tag{21}$$

$$SA_r(p) = \int_{\mathbf{p}+rS} 1_D(x) dx = \frac{dV_r(p)}{dr}. \tag{22}$$

Für den Gradienten des Volumen-Deskriptors gilt:

$$\nabla V_r(\mathbf{p}) = a_r(\mathbf{p}), \tag{23}$$

mit $a_r(\mathbf{p})$ ist Vektor mit Flächeninformationen (Pottmann et al., 2006, S.5):

$$a_r(\mathbf{p}) = \frac{1}{2} \oint_{cr} \mathbf{x} x d\mathbf{x} = \frac{1}{2} \int_I \mathbf{c}_r(u) x \dot{\mathbf{c}}_r(u) du. \tag{24}$$

Dem Volumen-Deskriptor ist der Vorzug gegenüber anderen Invarianten zu geben. So weist dieser die beste Stabilität auf, da er auf Basis von Oberflächenintegralen auf einer größeren Domäne und nicht wie andere Invarianten auf Basis von Abstandsfunktionen mittelt. Ein weiterer Vorteil ist, dass dieser Deskriptor den Unterschied der Hauptkrümmungen[25] schätzt (Pottmann et al.,2006, S.7-8).

3.4.2 Winkel-Deskriptor

Die Verwendung des Volumen-Deskriptor weist einige Nachteile auf. So kann man den Volumen-Deskriptor nicht verwenden, wenn man kein geschlossenes Mesh besitzt. Das zweite Problem ist der große Rechenaufwand, falls viele Faces im Punkt P zusammentreffen.

[25] Jedem Punkt einer Fläche im dreidimensionalen euklidischen Raum $\mathbb{R}^3$ werden zwei Hauptkrümmungen zugeordnet [HK]. Es handelt sich um zwei Hauptkrümmungen, da eine Fläche zwei zueinander senkrecht stehende Tangentenvektoren in einem Punkt besitzt [HKB].

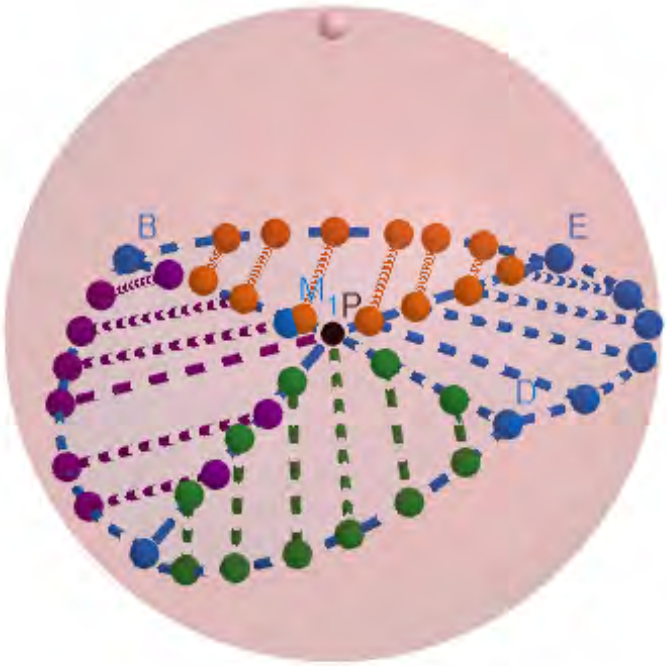
Eine mögliche Lösung entsteht durch eine Verschiebung des Punktes P auf eine Kantenmitte. Dies reduziert den Rechenaufwand deutlich und funktioniert zudem bereits, wenn an einer Kante beide angrenzenden Faces existieren. Man erhält dann:

$$V_r = \frac{V_O}{V_U} = \frac{\frac{4}{3}r^3\Pi * (\frac{\Pi+\alpha}{2\Pi})}{\frac{4}{3}r^2\Pi * (\frac{\Pi-\alpha}{2\Pi})}, \tag{25}$$

wobei V_O für das Volumen oberhalb und V_U für das Volumen unterhalb der Faces steht.

Abbildung 4 zeigt den Volumen-Deskriptor bei gescanntem Mesh:

Abbildung 4 : Volumen-Deskriptor bei gescannten Punkten
Quelle: Eigene Darstellung

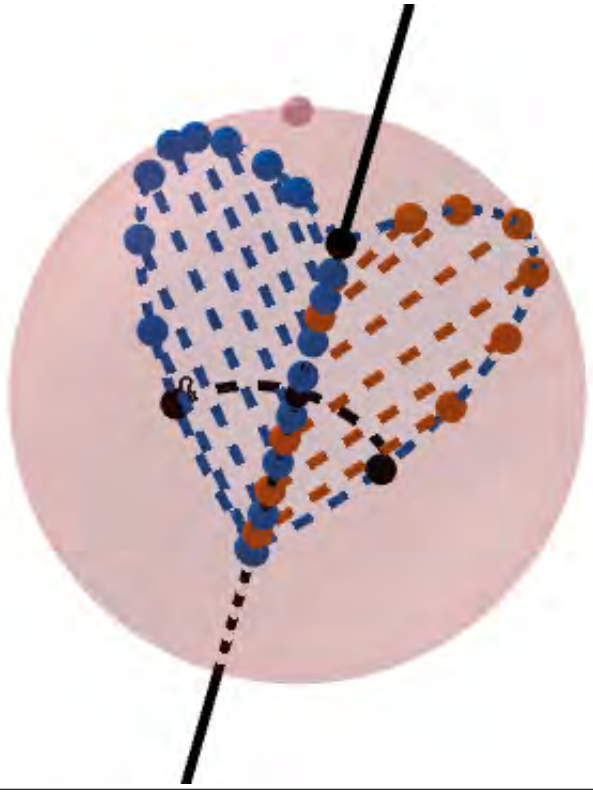


Nach Kürzen der Gleichung (25) hängt der Volumen-Deskriptor nur noch vom Winkel zwischen den Faces ab.

$$W = \frac{\Pi + \alpha}{\Pi - \alpha}. \tag{26}$$

Wählt man als Mittelpunkt der Kugel die Mitte einer gescannten Kante, ergibt sich folgende Abbildung 5:

Abbildung 5 : Volumen-Deskriptor, falls P der Mittelpunkt einer Kante ist
Volumen-Deskriptor, falls P der Mittelpunkt einer Kante ist.
Quelle: Eigene Darstellung



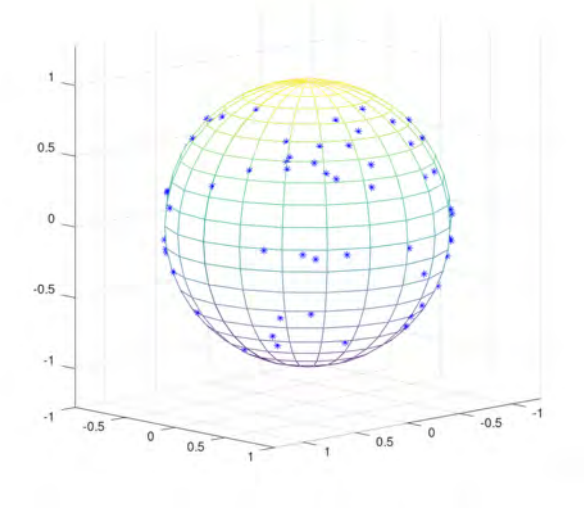
Dieser Deskriptor wird hier *Winkel-Deskriptor* genannt. Er weist einige positive Eigenschaften auf. Zum einem stimmen der Volumen-Deskriptor und der Winkel-Deskriptor für diese Art der Diskretisierung überein, da der Winkel-Deskriptor, als Verhältnis der beiden Winkel zwischen den benachbarten Faces, sämtliche Information des Volumen-Deskriptors enthält. Das heißt er bietet alle Vorteile, die der Volumen-Deskriptor gegenüber anderen Deskriptoren aufweist.
Der Winkel-Deskriptor hat aber, zusätzlich zu der noch einfacheren Berechenbarkeit, einen gravierenden Vorteil gegenüber dem Volumen-Deskriptor. Wie in (26) gezeigt, enthält der Winkel-Deskriptor keine Information über die Kernel-Größe, da der Radius gekürzt wird. Mit anderen Worten: der Winkel-Deskriptor ist also robust gegenüber der Kernel-Größe, während man den passenden Radius für den Volumen-Deskriptor situationsabhängig gewinnen muss. Dieses 'Probieren' entfällt für den Winkel-Deskriptor.

4 Implementierung

4.1 Einheitskugel

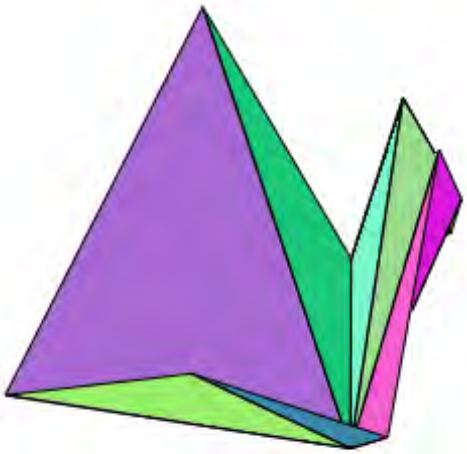
Ziel ist es, ein Programm zu entwickeln, das Gravuren in beliebigen Oberflächen erkennt. Um mögliche Schwierigkeiten entdecken zu können, wird die Komplexität der Oberfläche schrittweise erhöht. Begonnen wird mit einer Einheitskugel. Dazu wird der Mittelpunkt M in den Ursprung eines Koordinatensystems gelegt. Danach wird eine feste Anzahl von Punkten P zufällig bestimmt, indem für alle drei Koordinaten Zufallszahlen ermittelt werden. Durch Normierung des Vektors $\vec{MP}$, also $|\vec{MP}| = 1$, erreicht man, dass alle Punkte auf einer imaginären Kugeloberfläche liegen, was man in der folgenden Abbildung 6 sehen kann:

Abbildung 6 : Kugel mit 100 Punkten
Die Abbildung zeigt die selbst erzeugte Kugel mit 100 zufällig gesetzten Punkten auf der Oberfläche.
Quelle: Eigene Darstellung



Mit diesem Satz von Punkten, die man ebenso auch als Ergebnis eines Scans erhalten könnte, wird nun ein Mesh mit Dreiecks-Faces aufgebaut. Dafür bestimmt man zu jedem Punkt die (drei, vier, fünf, sechs) nächstgelegenen Punkte und ermittelt damit alle Faces um den gewählten Punkt. Ein Beispiel für zehn Faces mit drei Nachbarn, zeigt die Abbildung 7.

Abbildung 7 : Zehn Punkte und Dreickesflächen
Zehn Punkte, die mit den nächsten drei Nachbarn zu Dreiecksflächen verbunden wurden.
Quelle: Eigene Darstellung



Ein Plot dieser Faces offenbart bereits wesentliche Probleme, denn das Mesh ist, wie man sofort erkennen kann, nicht geschlossen. Außerdem werden einige Faces von anderen überlagert. Besonders gravierend ist die Tatsache, dass manche Faces 'mitten' durch die Kugel, also nicht an der Oberfläche verlaufen. Diese drei Feststellungen zeigt die folgende Abbildung:

Abbildung 8 : 100 Punkte mit drei Nachbarn
100 Punkte, die mit den nächsten drei Nachbarn zu Dreiecksflächen verbunden wurden.
Quelle: Eigene Darstellung

Eine deutliche Verbesserung erhält man, wenn man Faces eliminiert, deren Schwerpunkt von M weniger als 0,9 entfernt ist. Eine bessere Verteilung der Punkte ergibt sich zudem, wenn die Koordinaten der Punkte nicht völlig beliebig gebildet werden, sondern eine feste Anzahl von Punkten je Oktant auf der Kugeloberfläche platziert wird. Das heißt zum Beispiel, dass man in jedem Oktanten zehn Punkte zufällig konstruieren lässt. Das liefert ein 'geschlosseneres' Mesh als zuvor, dennoch aber immer noch mit vielen Löchern.

Abbildung 9 : Unvollständiges Mesh
Das Mesh ist weder geschlossen, noch liegen die Faces überschneidungsfrei da.
Quelle: Eigene Darstellung

Um eine Ritzung bzw. eine Gravur zu simulieren, werden nun zusätzlich einige Punkte derart entlang eines Viertelkreises der Kugeloberfläche gelegt, dass deren Abstand zu M lediglich 0,8 beträgt und die Punkte damit eine deutliche Vertiefung bilden. Der neue Plot ergibt nun ein recht geschlossenes Mesh. Allerdings ist die Ritzung nicht zu erkennen:

Abbildung 10 : Kugelmesh mit Ritzung
Kugelmesh mit künstlicher Ritzung. Die Ritzung ist in dieser Darstellung kaum erkennbar.
Quelle: Eigene Darstellung

Nun folgt also der erste Versuch, eine Gravur in einer Oberfläche kenntlich zu machen. Dazu wird zuerst der Abstand des Schwerpunktes S eines Face vom Mittelpunkt bestimmt. Faces werden aussortiert, wenn der Abstand kleiner als 0,9 ist, also gilt

$$|\vec{MS}| = |\vec{S}| < 0,9,$$

wobei $\vec{S} = \vec{MS}$, da M Ursprung des Koordinatensystems ist.
Die Kugeleigenschaften des Objektes werden derart genutzt, dass für jedes Face der Normalenvektor $\vec{n}$ bestimmt wird. Nun wird der Winkel ϕ zwischen dem Normalenvektor und dem Ortsvektor S des Schwerpunktes bestimmt:

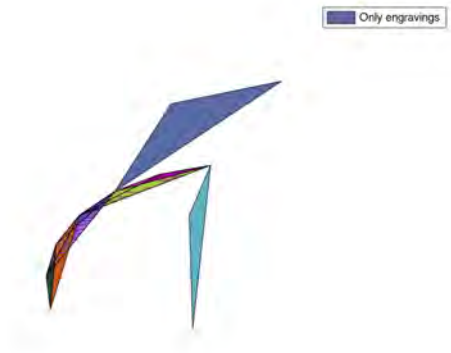
$$cos\phi = \frac{|\vec{S} \circ \vec{n}|}{|\vec{S}| * |\vec{n}|}, \tag{27}$$

Für $|cos\phi| \approx 1$ liegt das Face etwa tangential zur Kugeloberfläche. Werden nur Faces angeplottet, für die gilt,

$$|cos\phi| < 0,9 \tag{28}$$

so erhält man die folgende Darstellung (Abbildung 11), in der der vertiefte Viertelkreis deutlich zu erkennen ist.

Abbildung 11 : Kugelmesh mit Ritzung
Kugelmesh mit künstlicher Ritzung.
Quelle: Eigene Darstellung



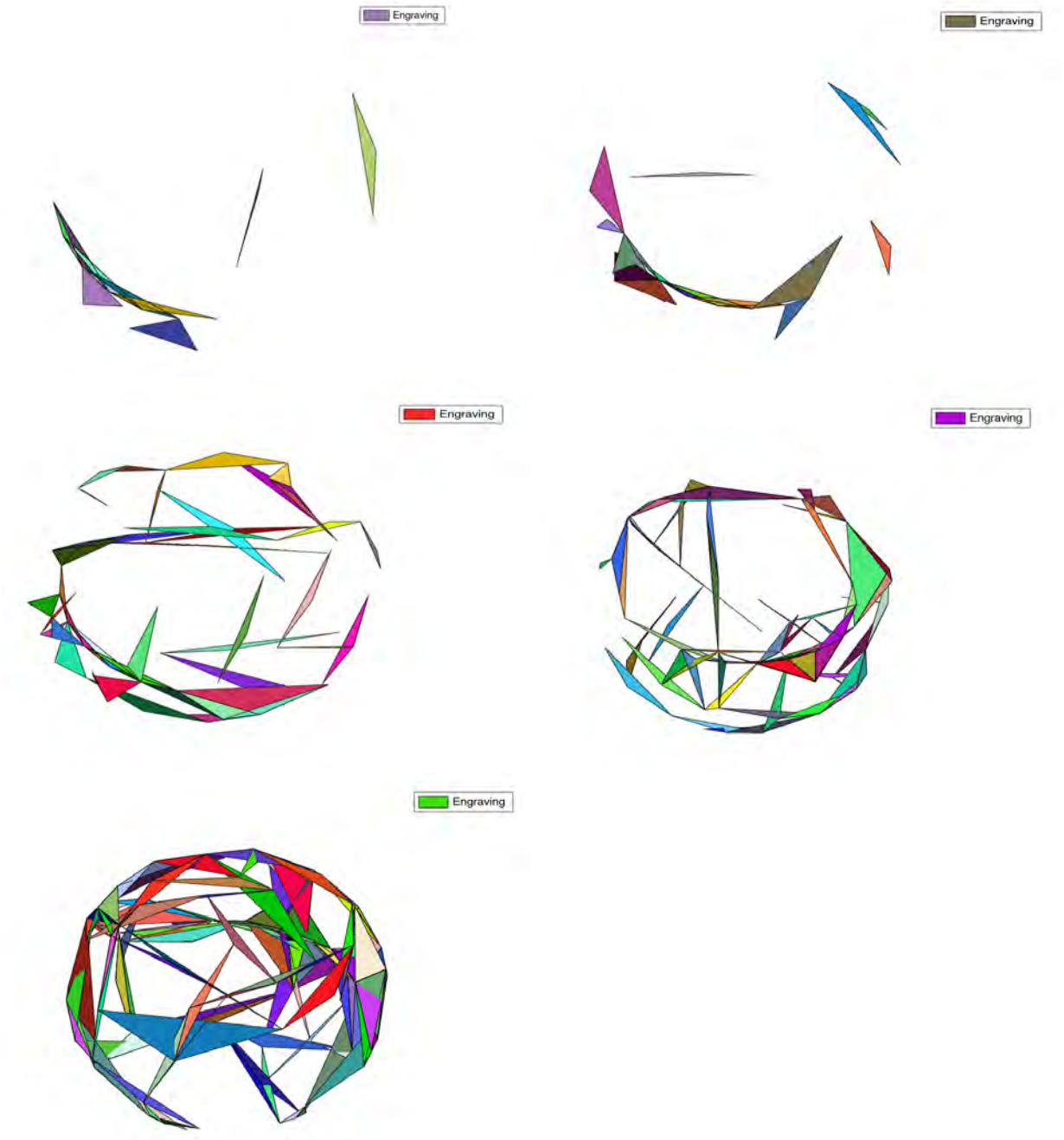
Um sich unabhängig von der Kugelform des Objektes zu machen, wird ein weiteres Verfahren entwickelt. Diesmal wird versucht, den Volumen-Deskriptor einzusetzen. Ein erster Versuch mit Wahl eines Punktes P als Zentrum der Domäne des Volumen-Deskriptors scheitert aber einmal an der komplexen Berechnung, da die Reihenfolge der Punkte bei der Erstellung völlig beliebig ist. Zum anderen schlägt der Versuch fehl, da das Mesh Löcher ausweist und somit in den meisten Kernel die Volumenberechnung nicht erfolgen kann. Abhilfe schafft die in 3.4 gezeigte Wahl des Domänenmittelpunktes als Mittelpunkt der Kante. Wenn jetzt noch Kanten ausgewählt werden, die an zwei Faces grenzen, kann man mit dem Volumen-Deskriptor, oder besser mit dem Winkel-Deskriptor, diejenigen Faces auswählen, für deren Normalenvektoren $\vec{n_1}$ und $\vec{n_2}$ gilt:

$$cos\phi = \frac{|\vec{n_1} \circ \vec{n_2}|}{|\vec{n_1}| * |\vec{n_2}|} \quad und$$

$$|cos\phi| < GRENZE \tag{29}$$

Der Nutzer wird im Programm gefragt welche GRENZE gewählt werden soll. Die Wahl der GRENZE bestimmt dann die Trennschärfe des Verfahrens.

Abbildung 12 : Darstellung mit verschiedenen Obergrenzen
Die Schwellengrößen 0,04, 0,15, 0,3, 0,5 und 0,8 ergeben diese Plots. Die Gravur ist in jeder Abbildung erkennbar, aber je kleiner die angegebene Zahl GRENZE ist, desto genauer kann auch die Gravur ausgegeben werden. Je größer die Obergrenze gewählt wird, desto mehr Störfaktoren, sprich mehr Dreiecksflächen, die nicht zur Ritzung gehören, werden mit angezeigt.
Quelle: Eigene Darstellung



Während (29) also die Abweichung von der Kugeleigenschaft untersucht, betrachtet (30), ob zwei angrenzende Faces etwa parallel zueinander liegen oder gegeneinander geneigt sind. Die Krümmungseigenschaften der Oberfläche spiegeln sich dabei nach 3.4 völlig in (30) wieder. Der Plot ist dabei nahezu identisch mit Abbildung 11.

4.2 Kugel in Grafikprogramm

In gescannten Objekten ist die Reihenfolge der Punkte nicht völlig beliebig, sondern folgt einem Muster. Auch dafür soll es eine Prüfung geben. Dazu wird ein Objekt in einem Grafikprogramm erstellt, wieder um eine Kugel. Als Ritzung wird ein eingestantzer Buchstabe gewählt. Lässt man diesen Körper nun als stl-File ausgeben, werden die Oberflächenpunkte so angeordnet, dass man ein geschlossenes Mesh erhält. Mit beiden Verfahren aus 4.1 wird nun der Buchstabe herausgefiltert. Die Abbildung 13 zeigt, eine im Grafikprogramm Blender erstellte Kugel, mit einem 'eingraviertem' Buchstaben. Die Abbildung 14 zeigt diesen Buchstaben nach Filterung der Ritzung.

Abbildung 13 : Kugel mit 'eingraviertem' T.
Quelle: Eigene Darstellung

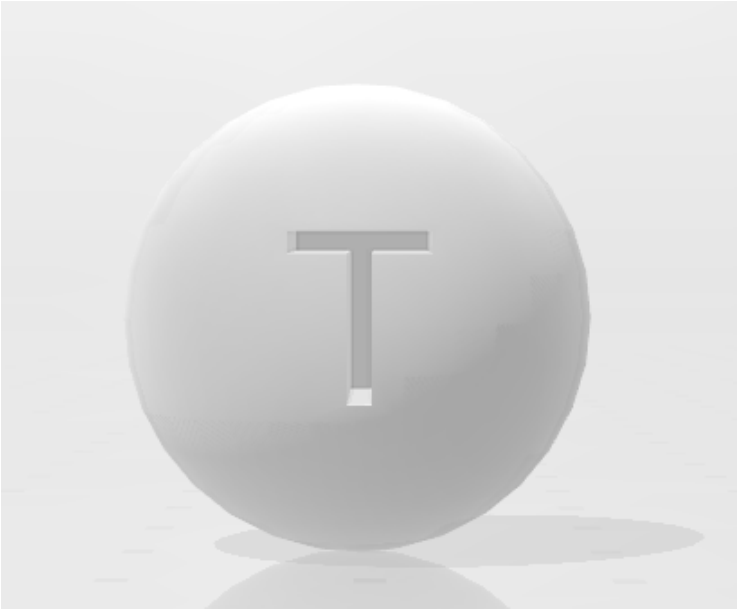
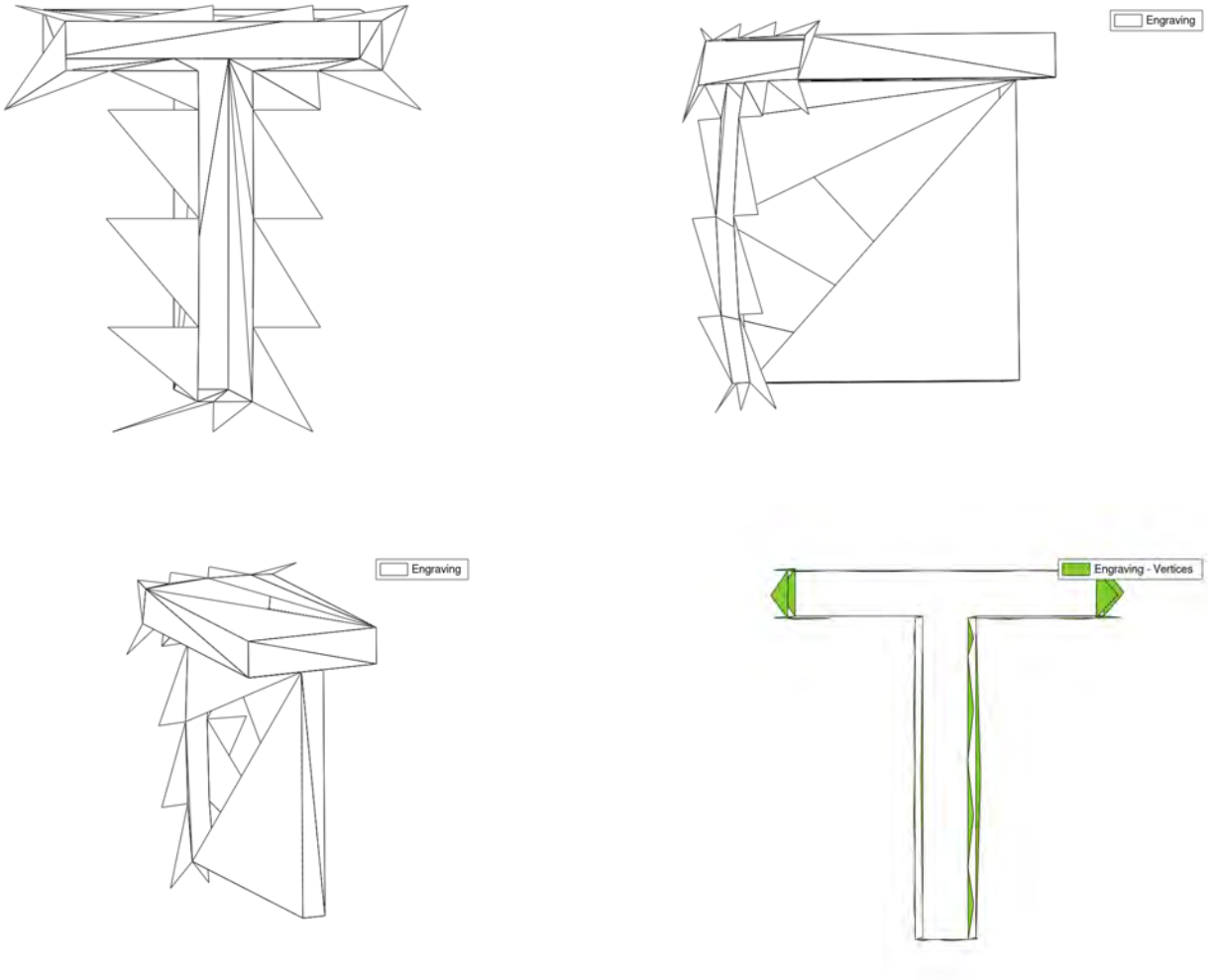


Abbildung 14 : Darstellung des gefilterten Buchstabens
Beginnend von links: $'T'$ von vorne, seitlich, von hinten oben. Die ersten drei Grafiken zeigen den gefilterten Buchstaben T, wenn man diesen durch die angrenzenden $Faces$ ausgeben lässt und die letzte Abbildung zeigt den Plot durch die gemeinsamen Kanten. Beides wird mit einer Obergrenze von 0,8 durchgeführt.
Quelle: Eigene Darstellung



Der Plot ist für beide Verfahren identisch, so dass sich der Winkel-Deskriptor auch hier als sinnvoll erweist.

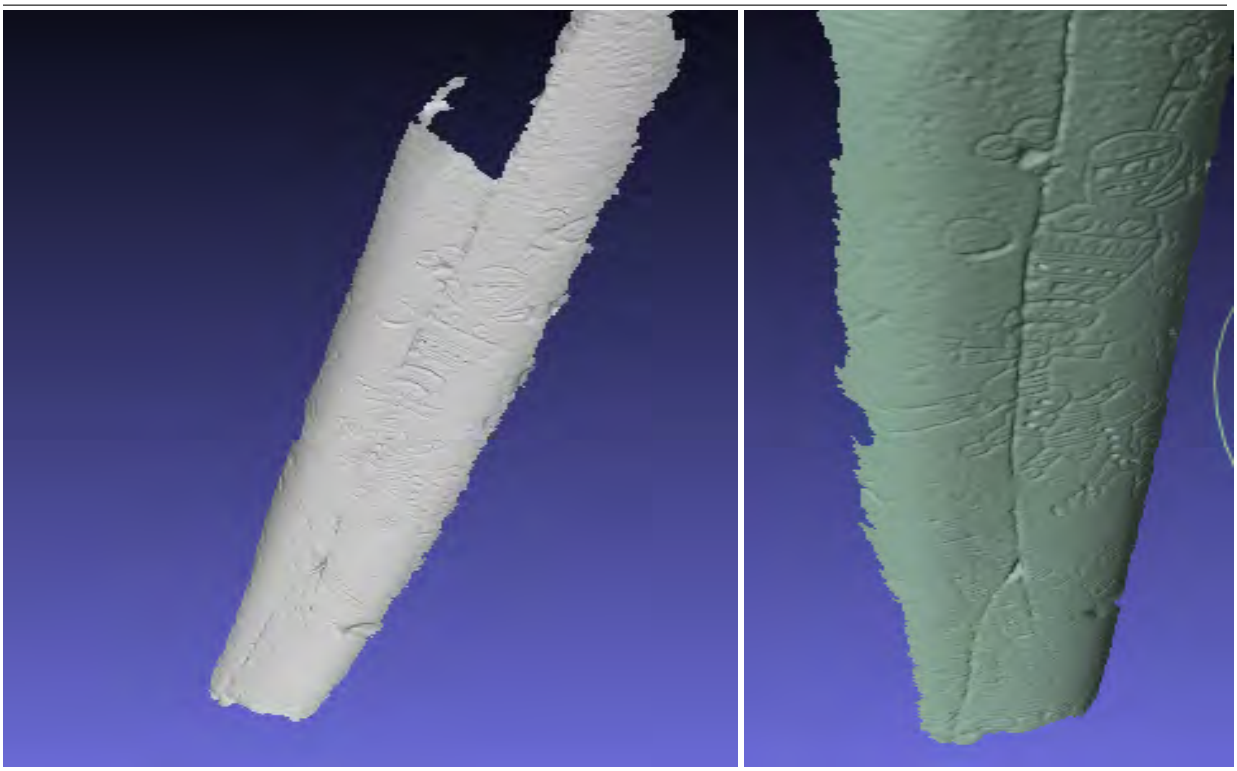
Auf dem Weg zu diesem Ergebnis hat sich allerdings ein Problem ergeben, das hier erwähnt werden soll, da es sich bei gescannten stl-Files ebenso stellt. In der Liste der Faces werden deren Eckpunkte durchnummeriert. Kommt ein Punkt P in mehreren Faces vor, so wird dem Punkt jedes mal eine neue Nummer zugeordnet. Derselbe Punkt mit den identischen

Koordinaten taucht also öfter in der Liste der Punkte auf. Um Eindeutigkeit herzustellen, werden daher alle Punkte in der Liste der Faces mit der ersten Nummer des Auftretens in der Punkteliste bezeichnet. So ist die Identifizierung beider an eine Kante grenzenden Faces möglich.

4.3 Die Knochenflöte

Das in 4.1 und 4.2 entwickelte Verfahren wird nun auf die stl-Datei der Aztekischen Knochenflöte angewandt. Abbildung 15 zeigt einen Plot der Scan-Datei.

Abbildung 15 : Die Knochenflöte
Links erkennt man die Knochenflöte, nach dem Einlesen der stl-Datei. Die Ritzungen sind hier sehr schwer zu erkennen. Rechts kann man die Gravur nach ein paar Veränderungen in MeshLab besser erkennen.
Quelle: stl-File: Modifizierung in MeshLab



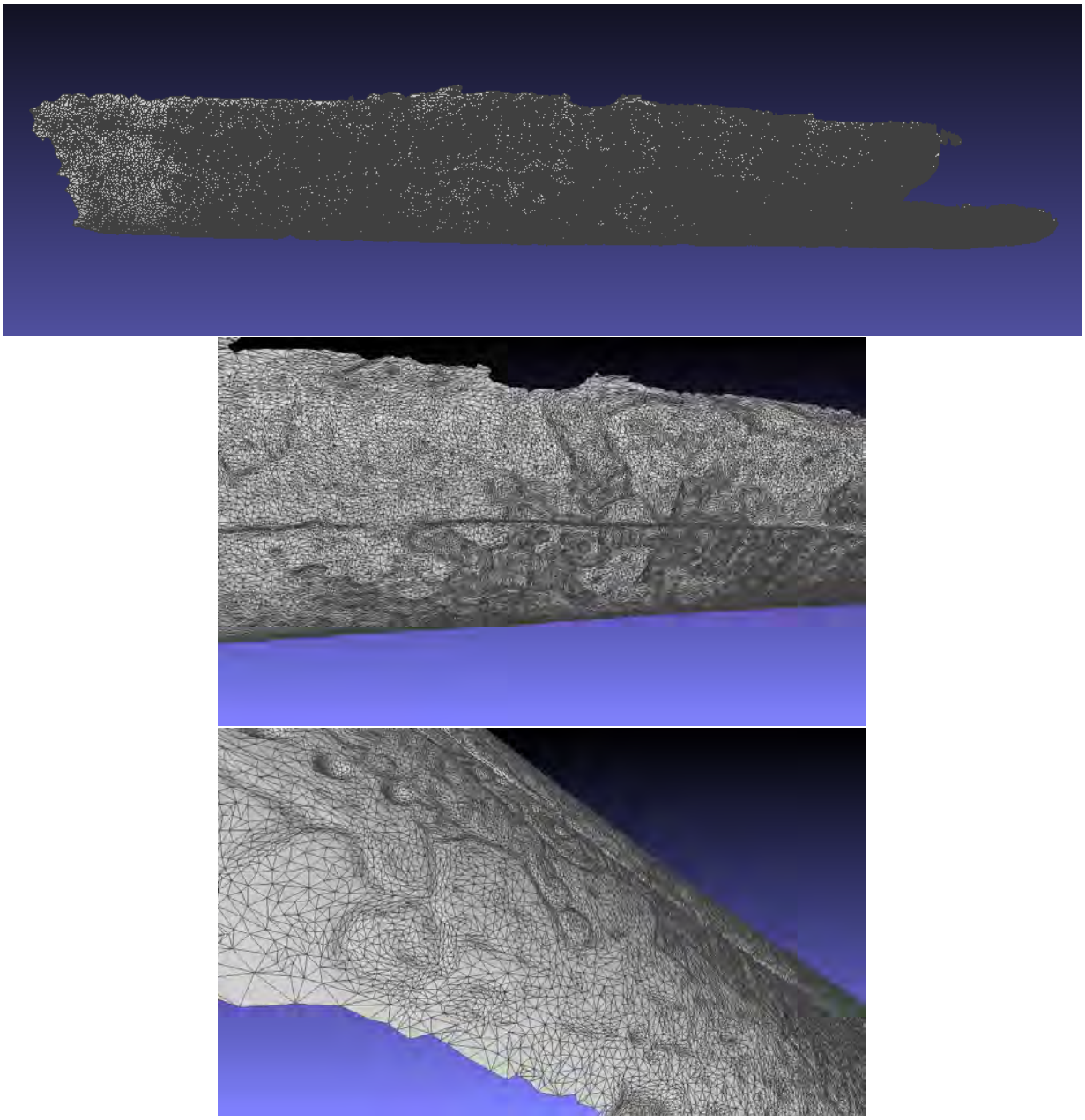
Das soll zum einen eine Vorstellung des Objektes vermitteln, zum anderen die Schwierigkeiten zeigen:

- Der Knochen ist durchgehend bleich, so dass sich die Ritzung farblich nicht abhebt (erst nach Bearbeitung im Grafikprogramm erkennbar).

- Eine deutliche natürliche Einkerbung verläuft über die gesamte Länge der Flöte, mittig durch den Knochen.

- Die Knochenflöte ist sehr rau, was zum einen die Unterscheidung zur Ritzung erschwert, zum anderen eine große Anzahl von Scan-Punkten und damit Faces liefert (siehe Abbildung 16).

Abbildung 16 : Knochenflöte mit Mesh
Oben ist die gesamte Knochenflöte mit den über 100000 Faces dargestellt. Die unteren Bilder zeigen Ausschnitte auf denen die Dreiecksfaces zu erkennen sind. Man erkennt auch, dass an Unebenheiten der Oberfläche die Scanpunkte sehr nah beieinander liegen.
Quelle: stl-File: Modifizierung in Meshlab



Da die Knochenflöte kein Kugelobjekt ist, kann nur mit dem Winkel-Deskriptor gearbeitet werden. Das Programm aus 4.2 liefert kein Ergebnis, da die Rechenzeit extrem lang ist[26]. Das Programm wurde dahingehend geändert, dass die Datei der Faces nach der Bereinigung der Punktenummern zunächst sortiert wird. Die Sortierung erfolgt im ersten Schritt innerhalb der Zeile, also innerhalb des Faces, danach über alle Zeilen. Bei der Suche nach gleichen Kanten, also zwei gleichen Punkten in einer Zeile, kann die Anzahl der zu durchlaufenden Faces dadurch sehr stark eingeschränkt werden. Die Programmlaufzeit beträgt schließlich etwa 4 Stunden.

Eine zweite Änderung gegenüber 4.2 erfolgt für die Ausgabe der Faces. Die Auswahl der anzudruckenden Faces kann genauer spezifiziert werden. Es wird zusätzlich zur Obergrenze des Kosinus nun auch eine Untergrenze abgefragt. Dies erlaubt auch besonders große Winkel herauszufiltern, was etwas bessere Plots liefert. Die Verbesserung wird beim Vergleich der Abbildung 17 mit der Neuerung in Abbildung 18 deutlich.

[26]Ein erster Versuch wurde nach 30 Stunden abgebrochen. Eine Abschätzung der Rechenzeit ergab eine Laufzeit von über 400 Stunden.

Abbildung 17 : Knochenflöte mit verschiedenen Grenzen
Die Plots zeigen die Ergebnisse für die Obergrenzen 0,99, 0,96 und 0,8. Bei der Filterung für $|cos\phi| < 0,99$ sieht man keine Trennschärfe. Für $|cos\phi| < 0,95$ sind deutlich weniger Faces zu erkennen, aber man erkennt dennoch kaum etwas. Bei $cos\phi < 0,8$ ist jedoch zu viel herausgefiltert worden.
Quelle: Eigene Darstellung

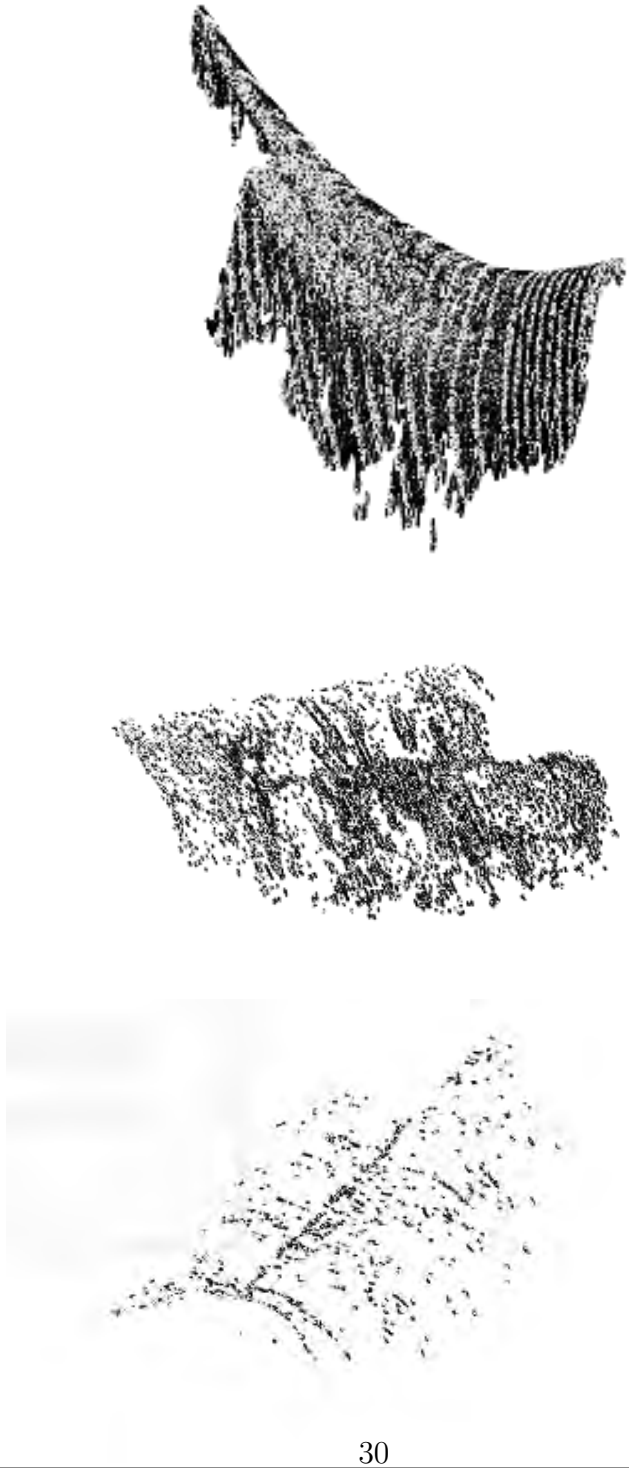
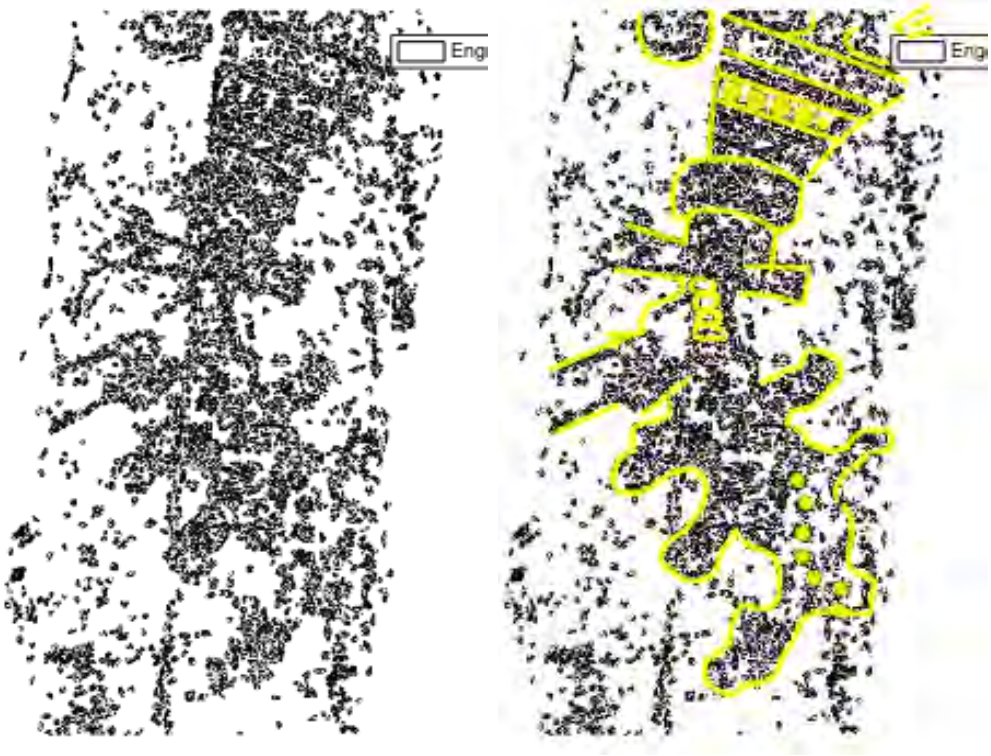


Abbildung 18 : Knochenflötenausschnitt
Im linken Bild kann man den mittleren Teil der Knochenflöte erkennen. Die Ausgabe erhält man für $0,86 < |cos\phi| < 0,96$. Rechts ist derselbe Ausschnitt mit einer farblichen Markierung abgebildet, die sich an der bearbeiteten Vergrößerung eines Teiles der Originaldatei orientiert.
Quelle: Eigene Darstellung; stl-File: Modifizierung in MeshLab



4.4 Technische Hinweise und Kurzbeschreibung des Programms

Technische Hinweise

Das Programm wurde unter Windows 10 mit GNU Octave, Version 5.1.0 erstellt. Für die Graphik-Plots wird die Octave GUI benötigt. Außerdem wurde nur das Octave-Package 'geometry' zur Implementierung verwendet. Zur Erstellung der Kugeln mit eingraviertem Buchstaben wurde Blender 2.79 verwendet. Zur besseren Darstellung und Bearbeitung der Knochenflöte wurde MeshLab 2016.12 verwendet.

Kurzbeschreibung

Mittels *createFacesAndVertices* kann das Verfahren an der selbst erstellten Kugel mit Ritzung betrachten werden[27]. Das Hauptprogramm arbeitet nur mit *Binary-STL-Files*[28]. Um das Programm zu starten, muss man in der GUI folgenden Befehl eingeben: ***getEngraving(filename)***. Dabei ist *filename* ein Platzhalter. Es sollte eine Eingabe in Hochkommata eingeschlossenen Dateinamen gefolgt von *.stl* folgen, z.B.: *'Knochenfloete.stl'*. Je nach Größe bzw. Anzahl der Faces kann es ein wenig dauern, bis die Ritzung angedruckt werden kann.
Im Vergleich braucht man mit 1010 Faces ca. 5 Sekunden gegenüber ca. 4 Stunden bei 137734 Faces.

getEngraving

Ruft nacheinander die passenden Funktionen auf, um die Gravur ausgeben zu können.

Gebrauch: getEngraving(filename)

Input: filename - der Dateinamen; Welche STL-File soll eingelesen werden

Output: Die Gravur, die in das Objekt eingeritzt/gestanzt wurde.

binarySTLread

This function reads an STL file in binary format into matrixes vertices and faces which can be used by patch to view the stl.
MATLAB code by Doron Harlev
Octave edits by John Moosemiller && @zmughal

Gebrauch: [Vertices, Faces, Color] = stlread(filename)

Input: filename - der Dateinamen; Welche STL-File soll eingelesen werden

Output: Vertices, Faces und Colour

[27] Das Vorgehen wird in 4.1 beschrieben.
[28] Sollte die gewünschte Datei nicht in einem Binary-STL-File Format vorliegen, dann kann diese zum Beispiel unter `www.meshconvert.com` zunächst konvertiert werden.

Edited by Laura Anolick, August 2019

cleanFaces

Ersetzt diejenigen Punkte, die mehr als einmal vorkommen. Somit werden alle Duplikate gelöscht.

Gebrauch: [FacesClean] = cleanFaces(Faces, Points)

Input: Faces - diejenigen Matrix, die aktualisiert wird (integer [n x 3]).

Points - Die Matrix mit den passenden Punkten der Faces. Hier werden die Duplikate gelöscht (float [n x 3]).

Output: FacesClean - Öfter auftretende Punkte, werden mit der Nummer des ersten Auftretens versehen (integer [n x 3]).

sortFaces

Da es egal ist, an welcher Stelle in der Matrix das Dreieck gespeichert wird, kann diese Matrix umsortiert werden. So wird diese aufsteigend sortiert. Danach kann diese Matrix verwendet werden, um eine schnellere Laufzeit zu erreichen.

Gebrauch: [sortFaces] = sortFaces(Faces)

Input: Faces - diejenige Matrix, die sortiert werden soll (integer [n x 3]).

Output: sortFaces - diese Matrix beinhaltet die Nummern der Faces in aufsteigender Ordnung (integer [n x 4]).

normalVector

Berechnung der Normalvektoren der Faces.

Gebrauch: [NormalVec] = normalVector(sortFaces, Points)

Input: sortFaces - Matrix mit allen aufsteigend sortierten Faces (integer [n x 3]).

Points - Die Matrix mit den passenden Punkten der Faces (float [n x 3]).

Output: NormalVec - Matrix mit allen Normalenvektoren der Faces (float [n x 3]).

calcAngle

In dieser Funktion werden diejenigen Kanten und Faces herausgesucht, die eine angegebene Schwellengröße unterschreiten.

Gebrauch: [cosVec] = calcAngle(NormalVec, sortFaces)

Input: NormalVec - Die Matrix mit allen Normalenvektoren zu jedem Face (float [n x 3]).

sortFaces - diejenigen sortierten Faces, die untersucht werden sollen (integer [n x 3]).

Output: cosVec - Vektor mit den Winkeln zwischen zwei Kanten (float [n x1]).

Vertices - diejenigen Kanten, die die Ritzung darstellen (integer [n x 2]).

cosComparison

Es wird getestet, ob der Winkel zwischen den Normalenvektoren angrenzender Kanten eine Schwellengröße unterschreiten.

Gebrauch: angle = cosComparison(NormalVec, i, k)

Input: NormalVec - Matrix mit allen Normalenvektoren der Faces (float [n x 3]).

i - die i-te Zeile; hier wurde die gleiche Kante gefunden wie in der k-ten Zeile ((integer [1 x 1])).

k - die k-te Zeile; hier wurde die gleiche Kante gefunden wie in der i-ten Zeile (integer [1 x 1]).

Output: angle - Der Cosinus zwischen den beiden betrachteten Kanten (float [1x1]).

cosFilter

Der Nutzer kann entscheiden, welche Grenze er betrachten möchte.

Gebrauch: [FacesRes] = cosFilter(cosVec, sortFaces)

Input: cosVec - der Winkel zwischen zwei Kanten (float [n x 1]).

sortFaces - die sortierten Faces (integer [n x 3]).

Output: FacesRes - diese Matrix beinhaltet alle Faces, die angedruckt werden sollen (integer [n x 3]).

printRes

Gibt die eingeritzen Objekte aus.

Gebrauch: printRes(FacesRes, Points)

Input: FacesRes - diejenigen Faces, die die Ritzung darstellen (float [n x 3]).

Points - Die Matrix mit den passenden Punkten der Faces (float [n x 3]).

Output: plot der gesuchten Ritzung.

plot_mesh

plot a 3D mesh
Copyright (c) 2009, Gabriel Peyre
Edited by Laura Anolick, Aug 2019

Gebrauch: h = $plot_mesh(F, V, face\_vertex\_color)$

Input: F - Faces - diejenigen Faces die untersucht wurden (integer [n x 3]).

V - in meinem Fall, die Punkte Martix, die zu den Faces gehört (float [n x 3]).

$face\_vertex\_color$ - Matrix mit den Farben der Kanten bzw. Faces (integer [n x 3]).

Output: plot des meshes.

Selbsterstellte Kugel und Gravur:

createFacesAndVertices

Ruft nacheinander die passenden Funktionen auf, um eine Kugel und eine Gravur selbst zu erstellen und anschließend die Ritzung herauszufiltern.

Gebrauch: createFacesAndVertices

Input: -

Output: 4 Plots: 1) Kugel mit zufälligen Punkten in den 8 Oktanten 2) Erzeugtes Mesh nur zu den Punkten 3) Erzeugtes Mesh mit der Ritzung 4) Nur die Ritzung.

kugel

Erstellt eine Kugel mit jeweils zehn zufällig verteilten Punkten in den 8 Oktanten[29].

Gebrauch: [Points] = kugel

Input: -

Output: Points - an welche Stellen wurden die Punkte gelegt (float [n x 3]).

withoutEngraving

Erstellen der Faces und des Meshes ohne eine Gravur.

Gebrauch: [Faces, CenterOfMassMat] = withoutEngraving(Points)

Input: Points - wo befinden sich die zufällig gewählten Punkte auf der Kugel (float [n x 3]).

Output: Faces - die das Mesh darstellen sollen (integer [n x 3]).

CenterOfMassMat - Matrix mit den Schwerpunkten zu jedem Face (float [n x 3]).

distPoints

Berechnung der 6 nächstgelegenen Punkte und Erzeugung dieser Faces.

Gebrauch: [Faces] = distPoints(Points)

Input: Points - alle Punkte, die sich auf der Kugel befinden (float [n x 3]).

Output: Faces - alle neu erzeugten Faces (integer [n x 3]).

removeDuplicate

Entfernt diejenigen Faces, die mehr als einmal vorkommen, so dass nur noch eines dieser Faces übrig bleibt.

Gebrauch: [FacesNew] = removeDuplicate(Faces)

Input: Faces - Matrix mit allen Faces (integer [n x 3]).

Output: FacesNew - die duplikatfreie Matrix der Faces (integer [n x 3]).

[29]Die Anzahl der verteilten Punkte kann jederzeit durch Veränderung der Variable 'number' im Code geändert werden

centerOfMass

Berechnung der Schwerpunkte der Faces.

Gebrauch: [CenterOfMassMat, deleteVector] = centerOfMass(Faces, Points)

Input: Faces - Matrix mit allen duplikatfreien Faces (integer [n x 3]).

Points - Die Matrix mit den passenden Punkten der Faces (float [n x 3]).

Output: CenterOfMassMat - Matrix mit allen Schwerpunkten der Faces (float [n x 3]).

deleteVector - in der i-ten Zeile wird der Schwellenwert von 0.75. unterschritten, deshalb wurde hier eine '1' hineingesetzt (integer [n x 1]).

randomColour

Erstellt für jedes Face, zur besseren Darstellung eine andere Farbe.

Gebrauch: [C] = randomColour(Faces)

Input: F - Faces - diejenigen Faces, die geplottet werden sollen (integer [n x 3]).

Output: C - colour-Matrix, die für jede Zeile eine andere Farbe speichert (float [n x 3]).

withEngraving

Erstellt künstlich eine Viertelgravur.

Gebrauch: [FacesWith, Points, CenterOfMassMat] = withEngraving(Points)

Input: Points - aus den Punkten neue Faces konstruiert werden sollen (float [n x 3]).

Output: FacesWith - die Faces-Matrix, die auch die Ritzungen beeinhaltet (integer [n x 3]).

Points - die neuen Punkte wurden zur alten Punktematrix hinzugefügt (float [n x 3]).

CenterOfMassMat - die Schwerpunkte aller Faces (float [n x 3]).

engraving

Es wird nur noch die Gravur ausgegeben, wenn eine gewisse Schwellengröße von 0.04 unterschritten wird.

Gebrauch: [Faces] = engraving(Faces, Points, CenterOfMassMat)

Input: Faces - Matrix mit allen Faces (integer [n x 3]).

Points - Die Matrix mit den passenden Punkten der Faces (float [n x 3]).

CenterOfMassMat - Matrix mit allen Schwerpunkten der Faces (float [n x 3]).

Output: Faces - Faces, die an der Gravur angrenzen (integer [n x 3]).

normalVector

Berechnung der Normalenvektoren der Faces.

Gebrauch: [NormalVec] = normalVector(Faces, Points)

Input: Faces - Matrix mit allen Faces (integer [n x 3]).

Points - Die Matrix mit den passenden Punkten der Faces (float [n x 3]).

Output: NormalVec - Matrix mit allen Normalenvektoren der Faces (float [n x 3]).

plot_mesh

plot a 3D mesh
Copyright (c) 2009, Gabriel Peyre
Edited by Laura Anolick, Aug 2019.

Gebrauch: h = $plot_mesh(F, V, face\_vertex\_color)$

Input: F - Faces - diejenigen Faces, die untersucht wurden (integer [n x 3]).

V - in meinem Fall, die Punkte Martix, die zu den Faces gehört (float [n x 3]).

$face\_vertex\_color$ - Matrix mit den Farben der Kanten bzw. Faces (integer [n x 3]).

Output: plot des meshes.

5 Ausblick

Das vorliegende Programm ermöglicht es für beliebig geformte Objekte, Krümmungen der Oberfläche herauszufiltern. Dies gilt insbesondere für Ritzungen in Körperoberflächen, deren Grenze kenntlich gemacht werden kann, da angrenzende Faces gegeneinander geneigt sind. Eine Umformung des Volumen-Deskriptors ergibt bei geeigneter Wahl des Kernelzentrums den hier vorgestellten Winkel-Deskriptor. Bei ursprünglich glatten Oberflächen liefert der Winkel-Deskriptor sehr gute Ergebnisse. Bei der untersuchten Knochenflöte ist genau dieses Kriterium nicht erfüllt. Dadurch ist die Ausgabe verrauscht. Die Angabe eines Intervalles für die auszuwählenden Winkel liefert zwar befriedigende Ergebnisse, kann das Rauschen aber nicht völlig unterdrücken.

Künftige Arbeiten könnten auf Volumen- oder Winkel-Deskriptor aufbauen. Bei Ritzungen kommt es aber nur auf Vertiefungen in der Objektoberfläche an. Erhöhungen, die im vorliegenden Fall für das Rauschen besonders verantwortlich sind, sollten also herausgefiltert werden.
Eine weitere Verbesserung könnte sich ergeben, wenn die Vertiefungsflächen, die zwischen den bereits hier abgebildeten Rändern der Vertiefung liegen, auch mit angeplottet würden. Dies würde bei großflächigen Vertiefung die Unterscheidung verbessern, zum Beispiel zwischen Kreis und Ring.

6 Literatur

Paper und Abschlussarbeiten

BBSK Bachelorarbeit- Markus Bruhn- Saklierungsinvariante Konturidentifikation

CE03 Cazals, F., and Pouget, M. 2003. Estimating differential quantities using polynomial fitting of osculting jets. In Symp. Geometry Processing, Eurographics Assoc., L. Kobbelt, P. Schröder, and H. Hoppe, Eds., 177-187.

CM03 CAZALS, F., CHAZAL, F., AND LEWINER, T. 2003. Molecular shape analysis based upon the Morse-Smale complex and the Connolly function. In SCG '03: Proc. 19th Symposium on Computational Geometry, ACM Press, 351–360.

FA00 Frisken, S. F., Perry, R. N., Rockwood, A. P., and Jones T. R. 2000. Adaptively sampled distance fields: a gerneal representation of shape for computer graphics. In Proc. Siggraph, 249-254. Gelfand, N., Mitra.

LG Liu - Geometric Modeling with Conical Meshes and Developable Surfaces description

LS Levallois - Scale-space Feature Extraction on Digital Surfaces description

MG04 MITRA, N., GELFAND, N., POTTMANN, H., AND GUIBAS, L. 2004. Registration of point cloud data from a geometric optimization perspective. In Symp. Geometry Processing, Eurographics Assoc., R. Scopigno and D. Zorin, Eds., 23–32.

MI06 Manay - Integral Invariants for Shape Matching description -2006

PI06 Pottman et al. Integral Invariants for Robust Geometry Processing - 2006 - Hauptpaper

PP07 Pottman et al. Principal curvatures from the integral invariant viewpoint - 2007

QR06 Huang et al. - Reassembling Fractured Objects by Geometric Matching - Wiederzusammenfügen zerbrochener Objekte durch 'Geometrische Übereinstimmung'-2006

WU55 Wilkens - Über die Integral-Invarinaten der Störungtehorie- 1955

YR06 YANG, Y.-L., LAI, Y.-K., HU, S.-M., AND POTTMANN, H. 2006. Robust principal curvatures on multiple scales. In Symp. Geometry processing, Eurographics Assoc., K. Polthier and A. Sheffer, Eds., 223–226.

Internetquellen

Deskriptor - Merriam Webster `https://www.merriam-webster.com/dictionary/descriptor`

HKB Hauptkrümmung - Anne Bläsius http://www.anne-blaesius.de/arbeit_flaechen/flaechenCa.html

Hauptkrümmungen https://deacademic.com/dic.nsf/dewiki/966263 - Hauptkrümmungen

Faltungsintegral - Plaßmann Wilfried https://ui.adsabs.harvard.edu/abs/2016hael.book.1363P/abstract

FFT1 FFT - Heideman, Michael T.; Johnson, Don H.; Burrus, Charles Sidney (1984). "Gauss and the history of the fast Fourier transform"(PDF). IEEE ASSP Magazine. 1 (4): 14–21. doi:10.1109/MASSP.1984.1162257

FFTF FFT - FH Flensburg http://www.iti.fh-flensburg.de/lang/algorithmen/fft/fft.htm

FFTP FFT - Uni Paderborn - Seite 14 http://www3.math.uni-paderborn.de/~walter/teachingSS04/VortragThema10.pdf

Mannigfaltigkeiten John M. Lee: Introduction to Smooth Manifolds. 2. Auflage. Springer, New York 2003, ISBN 0-387-95448-1 (englisch).

mittlere Krümmung https://deacademic.com/dic.nsf/dewiki/966263

octree; Oc Meagher, Donald (October 1980). Öctree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer". Rensselaer Polytechnic Institute (Technical Report IPL-TR-80-111).

MR Forster Otto Forster: Analysis. Band 2: Differentialrechnung im Rn. Gewöhnliche Differentialgleichungen. 7. verbesserte Auflage. Vieweg, Wiesbaden 2006, ISBN 3-8348-0250-6 (Vieweg-Studium. Grundkurs Mathematik) - metrischer Raum

univariate Grünwald, Robert. Ünivariate Statistik in SPSS". novustat.com (in German). Retrieved 29 October 2019.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie, dass ich die Bachelorarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Passau, 19. Februar 2020

Laura Anolick