Georg-August-Universität
Göttingen
Zentrum für Informatik

Ausdünnung von Dreiecksnetzen

Ralf Buchbach

Matrikelnummer: 21167010

Veranstaltung: Seminar Computergrafik

Semester: Wintersemester 2011/2012

Dozenten: Prof. Dr. Winfried Kurth
Reinhard Hemmerling

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Dreiecksnetze

Bei Dreiecksnetzen handelt es sich um eine Untergruppe der allgemeinen Polygonnetze. Bei den Polygonnetzen können alle Vielecke zur Erstellung eines Netzes verwendet werden, wohingegen Dreiecksnetze, wie der Name vermuten lässt, lediglich aus dem grafischen Primitiv eines Dreiecks bestehen.

Im Bereich der Computergrafik sind Dreiecksnetze das Netz der Wahl, um Gegenstände zu modellieren. Dies ist auch gleichzeitig ein Hauptanwendungsgebiet in dem Dreiecksnetze verwendet werden. Zusätzlich ist die aktuelle Grafikhardware auf die Berechnung, bzw. Verarbeitung von Polygonen spezialisiert. Dadurch ist eine hohe Verarbeitungsgeschwindigkeit bei grafischen Operationen gegeben. Ein Beispiel der Modellierung mittels Dreiecksnetze ist in Abbildung 1 dargestellt.

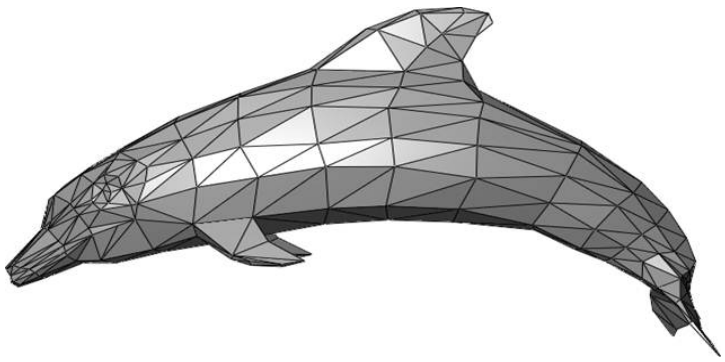


Abbildung 1: Modell eines Delphins mittels Dreiecksnetz[1]

Ein weiteres wichtiges Gebiet, in dem Dreiecksnetze verwendet werden, ist die wissenschaftliche Simulation. Zum Beispiel wird in der Strömungssimulation das zu untersuchende Objekt mit einem Dreiecksnetz umspannt (siehe Abbildung 2) und stellt damit diskrete Simulationspunkte dar. Je nach Definition der Simulation werden nun an allen

[1] `http://www.youin3d.com/images/stories/Dolphin_triangle_mesh.png`

Knoten oder in der Mitte jeder Dreiecksfläche die relevanten Parameter (z.B. Geschwindigkeit der Luftpartikel) berechnet. Über die Auflösung des Dreiecksnetz lässt sich damit ebenfalls die Simulationsgenauigkeit steuern, da ein engeres Dreiecksnetz gleichbedeutend mit mehr Simulationspunkten ist.

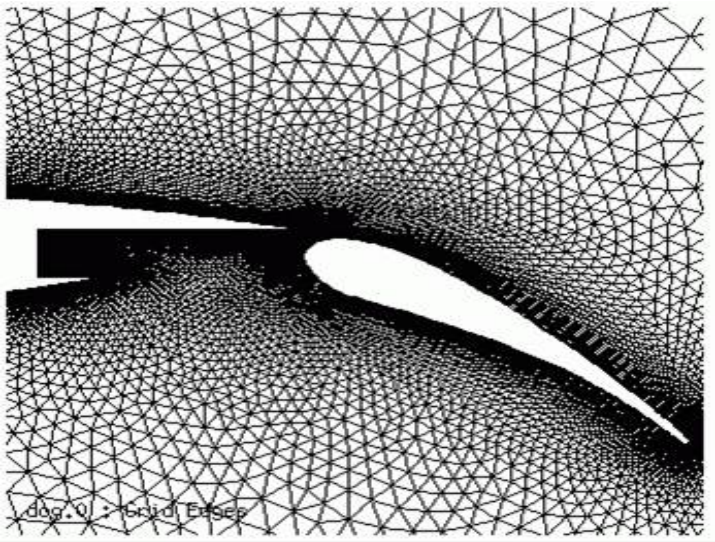


Abbildung 2: Simulation eines Flügelprofils mittels Dreiecksnetz[2]

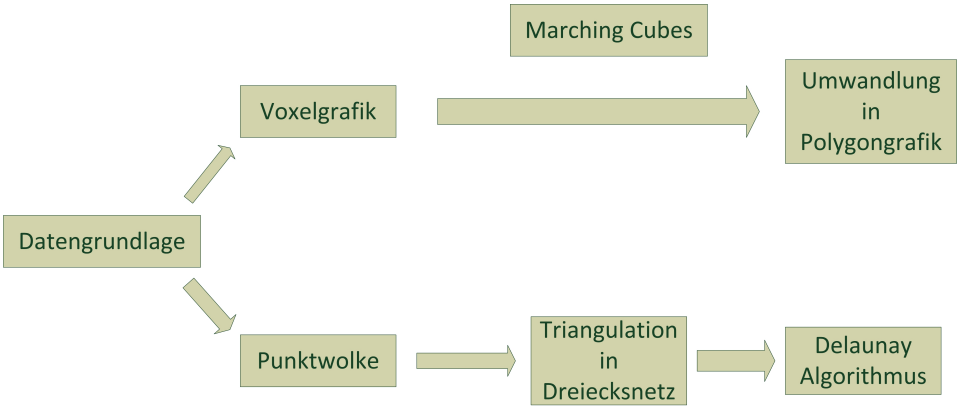


Abbildung 3: Generierung von Dreiecksnetzen

Die Generierung von Dreiecksnetzen zur Nachbildung real existierender Objekte kann auf zwei verschiedene Arten geschehen, die sich in ihrer Datengrundlage fundamental unterscheiden. Zum einen kann ein Dreiecksnetz aus Punktwolken generiert werden, die zum Beispiel bei der Abtastung eines Objektes mittels eines Lasers entstehen. Dabei

[2] `http://fun3d.larc.nasa.gov/xplt_3.png`

fährt der Laser um das zu modellierende Objekt herum und erzeugt bei jeder Abtastung eine unstrukturierte Punktwolke. Mittels einer initialen Triangulation, z.B. nach Lawsons Methode[3] wird ein erstes Dreiecksnetz erstellt. Dieses ist in der Regel noch nicht perfekt, weshalb in einem weiteren Schritt eine Verfeinerung des Netzes durch den Delaunay-Algorithmus[4], durchgeführt wird. Die einzelnen Scans des Lasers erzeugen Punktwolken, die zunächst mit einer 2D-Triangulierung bearbeitet werden. Erst durch das Zusammenführen aller einzelnen Scans entsteht das 3D-Modell [1]. Dieses Verfahren der Triangulierung wird von der Hardware unterstützt und benötigt wenig Speicher. Nachteilig ist jedoch, dass keine diffusen Objekte dargestellt werden können.

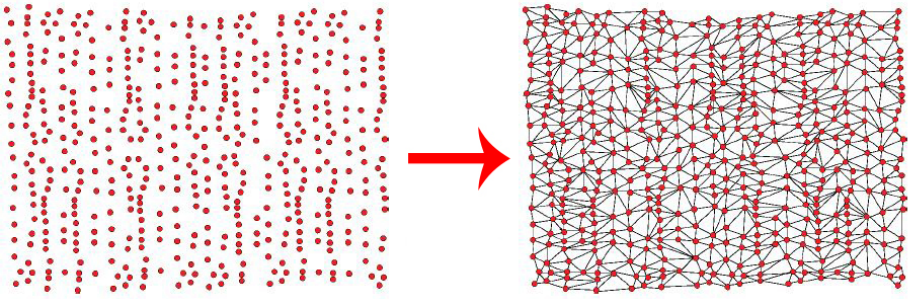


Abbildung 4: Triangulierung (nach [1])

Genau gegenteilig dazu sind die Volumengrafiken. Diese basieren nicht auf Pixeln, sondern auf den Voxeln. Dies sind Pixel, jedoch im 3D-Raum (Volumenpixel). Volumengrafiken entstehen z.B. bei der Computertomographie. Dabei wird das Objekt vermessen und für jedes Voxel die Position im Raum, sowie eine diskrete Information, wie beispielsweise die Dichte, gespeichert. Entsprechend lassen sich Volumengrafiken sehr häufig im medizinischen Bereich wiederfinden. Aber auch in der Industrie und im wissenschaftlichen Umfeld werden diese verwendet. Da die aktuelle Grafikhardware nicht auf Voxelgrafiken ausgelegt ist, müssen diese zunächst in Polygonnetze umgewandelt werden, damit eine flüssige Darstellung möglich ist. Ein Algorithmus, der diesen Schritt durchführt, ist der Marching Cubes Algorithmus[5] (vgl. [4]).

[3]Bei der Methode nach Lawson zur initialen Triangulierung eines Dreicksnetz werden die Punkte zunächst sortiert und anschließend ein Dreieck gebildet. Von diesem Dreieck ausgehend, werden Linien zu allen sichtbaren Punkten gezeichnet

[4]Beim Delaunay-Algorithmus wird ein Kreis durch die drei Punkte eines Dreiecks gezeichnet. Innerhalb dieses Kreises dürfen sich keine weiteren Punkte befinden. Ist dies dennoch der Fall, wird das Dreieck aufgeteilt, so dass das Kriterium erfüllt ist.

[5]Beim Marching Cubes Algorithmus wird das 3D-Modell in Schichten unterteilt. Diese Schichten werden nun iterativ untersucht. Bei der Untersuchung werden aus Voxeln der i-ten Schicht und (i+1)-ten Schicht ein Würfel gebildet. Die diskrete Information in den acht Ecken des Würfels wird mit ei-

Ein weiterer wichtiger Aspekt bei Dreiecksnetzen ist die Datenstruktur in der das Netz gespeichert wird. Die Wahl der Datenstruktur entscheidet bereits darüber, wie performant ein späterer Ausdünnungsalgorithmus sein kann, bzw. auch welche Operatoren zur Ausdünnung verwendet werden können. In Tabelle 1 sind einige Datenstrukturen aufgelistet.

Name	**Typ**	**Beschreibung**
Vertex / Topologie getrennt	Knotenbasiert	Positionen der Vertices und Topologie getrennt
Baumgart's Winged Edge	Kantenbasiert	Rechte und linke Fläche Anfang und Ende Vorgänger- und Nachfolgerkante (Im / Gegen Uhrzeigersinn)
Weiler's Radial Edge	Kantenbasiert	Topologie und Geometrie stark getrennt Speziell für Nicht-Mannigfaltige Netze
Mäntylä's Half Edge	Kantenbasiert	Jede Kante als 2 entgegengesetzte Halbkanten Startknoten 1 angrenzende Fläche Nachfolgerkante (Uhrzeigersinn) Gegenüberliegende Kante

Tabelle 1: Verschiedene Datenstrukturen zur Speicherung von Dreiecksnetzen

Generell kann gesagt werden, dass knotenbasierte Datenstrukturen einfach und speichereffizient sind. Da allerdings getrennte Tabellen für Knoten und Topologie genutzt werden, sind diese Strukturen schlecht für Nachbarschaftsabfragen. Für diese Abfragen eignen sich die kantenbasierten Strukturen mit dem Nachteil, dass mehr Speicher benötigt wird.

1.2 Ausdünnung von Dreiecksnetzen

Unter der Ausdünnung von Dreiecksnetzen versteht man die Reduzierung des Ausgangsnetzes um möglichst viele Dreiecke bei möglichst geringem Detailverlust. Die Motivation

nem Referenzwert verglichen und somit eine 8bit Binärzahl erzeugt. In einer LookUp-Table ist für jede mögliche Kombination der Binärzahl hinterlegt, welche Dreiecke gezeichnet werden müssen. Anschließend werden die Knotenpunkte der Dreiecke interpoliert und diese zusammen mit den Normalen der Dreiecke ausgegeben.

für Ausdünnungsalgorithmen stammt aus den Anfängen der Computergrafik. Während die Apparaturen zum Scannen, insbesondere die Erzeugung von Volumengrafiken, bereits sehr detailgetreue Netze mit vielen Dreiecken erzeugen konnte, war die Grafikhardware zu Beginn der 1990'er noch nicht in der Lage, die komplexen Modelle darzustellen. Volumengrafiken von Computertomographen in der Medizin und Industrie erzeugten bereits Modelle, die aus 256x256 bis zu 1024x1024 Pixeln und bis zu 100 Schichten bestanden. Aus solchen Datengrundlagen erzeugt der Marching Cubes Algorithmus Netze mit 500.000 bis 2 Millionen Dreiecken. Insbesondere ebene Flächen, die mit relativ wenig Dreiecken dargestellt werden können, bekommen verhältnismäßig viele Dreiecke zugewiesen.

Die Bearbeitung und Analyse dieser Modelle[6] konnte anschließend im Computer nicht mehr flüssig durchgeführt werden, da sämtliche Operationen im 3D-Raum, wie Rotation, Translation, Zoom, etc. einiges an Rechenzeit benötigten. Um einen Eindruck der Rechenleistung von frühen Grafikkarten zu bekommen, sei erwähnt, dass eine Nvidia GeForce 256 (1999) einen Dreiecksdurchsatz von 15 Millionen Dreiecken pro Sekunde besaß[7].

Ein weiterer Aspekt ist der Speicherverbrauch solch detaillierter Netze. Im damaligen Zeitraum stand Speicher nicht in dem Maß zur Verfügung, wie dies heute der Fall ist, und eine effiziente Nutzung des Speichers war sehr wichtig.

Allerdings liegt es auf der Hand, dass eine Reduktion von Dreiecken auch den Detailgrad des Modells verschlechtert. Da dies bei medizinischen Untersuchungen oder industrieller Qualitätssicherung fatale Auswirkungen haben kann, muss der Detailgrad nach Möglichkeit gewahrt bleiben. Hier zeigen sich die opportunen Ziele der Ausdünnungsalgorithmen. Zum einen möchte man möglichst wenig Dreiecke zur Darstellung nutzen, aber gleichzeitig möglichst viele Details des originalen Netzes behalten. Diese Balance zu erreichen ist Aufgabe der Ausdünnungsalgorithmen. Es sollen überflüssige Dreiecke, deren Beitrag zum Detailgrad eines Modells gering ist, gefunden und entfernt werden.

Im Laufe der Zeit bildeten sich verschiedene Möglichkeiten zur Ausdünnung von Dreiecksnetzen heraus. Alle Ansätze zur Ausdünnung von Dreiecken lassen sich in lokale und globale Algorithmen unterteilen. Die lokalen Algorithmen arbeiten auf einem Segment des Netzes und iterieren über das gesamte Netz. Dabei wird auf die Untermenge M des Netzes eine topologische Operation S ausgeführt und eine neue Untermenge M' erzeugt, die über weniger Dreiecke verfügt als die ursprüngliche Untermenge M. Unterschiede in

[6] z.B. Scan des menschlichen Schädels und Untersuchung auf Frakturen
[7] Siehe: http://www.nvidia.de/page/geforce256.html (Stand 17.12.2011)

den lokalen Algorithmen zeigen sich in der Wahl des topologischen Operators und einer Fehlerfunktionen, die den Fehler zum originalen Netz bestimmt. Im Gegensatz dazu betrachten die globalen Algorithmen das gesamte Netz und beziehen dieses Wissen in ihre Ausdünnungsoperationen ein. In Tabelle 2 auf Seite 7 sind verschiedene Möglichkeiten dargestellt und erklärt.

Name	Typ	Beschreibung	Vorteile	Nachteile
Vertexausdünnung	lokal	Operiert auf einem Knoten Entscheidung ob der Knoten beibehalten wird oder nicht	Sehr gut für Marching Cubes keine Topologieänderung	
Kantenkontraktion	lokal	Operiert auf einer Kante Zieht die Kante zu einem Vertex zusammen Dieses Vertex liegt irgendwo auf der ehemaligen Kante	Reduziert um genau 2 Flächen Verbreitester Operator	Kann Topologieänderung verursachen Fehlerfunktion komplex
Zusammenfallen von Halbkanten	lokal	Benötigt Half-Edge Datenstruktur Operiert auf einer Halbkante Halbkante wird immer auf Start- oder Endpunkt reduziert Einzelne Dreiecke werden entfernt	Einfachster Operator Keine neuen Knoten entstehen	
Erscheinungsbewahrung	lokal	Greift Rasterisierung des Netzes zur Darstellung auf dem Bildschirm auf Aspekte des Netzes, deren Fehlen nicht auffällt werden entfernt Aussehen des Netzes wird bewahrt	Netze sind kaum vom Original zu unterschieden	Umfassende Parametrisierung nötig, die i.d.R. nicht vorliegt
Vertex Clusterbildung	global	Jedem Vertex wird ein Wert zugewiesen Der Wert beschreibt die Wichtigkeit zur Repräsentation des Netzes 3D-Gitter wird über das Netz gelegt In jeder Zelle des Gitters werden alle Vertices auf das mit höchstem Wert reduziert	Effizient Einfach Steuerbar	Topologieänderungen möglich
Formenapproximierung	global	Ausdünnung wird als allgemeines Optimierungsproblem gesehen Netz wird in nicht überlappende Teilflächen unterteilt Jede Teilfläche wird durch ein Curve-Fit angenähert	Gute Annährungen	Effizienz

Tabelle 2: Verschiedene Möglichkeiten von Ausdünnungsoperationen [5], [3]

2 Algorithmus von Schroeder et al.

Im Jahre 1992 veröffentlichten William J. Schroeder, Jonathan A. Zarge und William E. Lorensen einen Ausdünnungsalgorithmus für Dreiecksnetze, der einen allgemeinen Ansatz verfolgte. Die Motivation der Autoren für diese Entwicklung bestand darin, dass Dreiecksnetze in einer Vielzahl von Anwendungsgebieten verwendet werden. Das allgemeine Problem der Reduktion von Netzen mit hoher Anzahl an Dreiecken wird in der Regel mit bereichsspezifischen Wissen gelöst. Daher stehen bisher entwickelte Ausdünnungsalgorithmen nur dem jeweiligen Anwendungsgebiet zur Verfügung und können nicht allgemein verwendet werden. Damit Ausdünnungsalgorithmen nicht immer von der Pike auf entwickelt werden müssen, setzten die Autoren es sich zum Ziel einen generalisierten Ausdünnungsalgorithmus zu entwicklen, der auf Netze verschiedener Anwendungsgebiete angewendet werden kann.

Als topologischen Operator wurde die Vertexausdünnung gewählt. Somit handelt es sich um einen lokalen Ausdünnungsalgorithmus, dessen einzelne Ablaufschritte

1. Lokale Geometrie und Topologie analysieren
2. Ausdünnungskriterium prüfen
3. Bei Erfolg: Vertex löschen und entstandene Lücke mit Dreiecken füllen

nachfolgend erläutert werden.

2.1 Analyse der lokalen Topologie und Geometrie

Um die lokale Geometrie und Topologie zu analysieren führen die Autoren verschiedene Vertexklassen ein. Jedes Vertex des Netzes wird auf seine Klassenzugehörigkeit untersucht. Anhand der Klasse fällt die Entscheidung, ob das Vertex ein potentieller Löschkandidat ist und welches Ausdünnungskriterium zum Einsatz kommt. Tabelle 3 zeigt die verschiedenen Klassen und ihre jeweilige Charakterisierung.

Von den dargestellten Klassen sind alle potentielle Löschkandidaten, bis auf die Klasse der komplexen Vertices.

Klasse	**Aussehen**	**Charakterisierung**
Einfache		Kompletter Kreis an Dreiecken Jede Kante wird von genau 2 Dreiecken genutzt
Komplexe		Nicht-Mannigfaltig Kante wird nicht von 2 Dreiecken genutzt Vertex wird von einem Dreiecke außerhalb der Ebene benutzt
Begrenzung		Vertex liegt am Rand des Netzes Halbkreis an Dreiecken
Innere Kante		V-Winkel zwischen zwei benachbarten Dreiecken größer als Referenzwert (Besondere Kante, fette Line) Exakt 2 Besondere Kanten
Eck		Eine oder mehr als zwei Besondere Kanten

Tabelle 3: Vertexklassen

2.2 Überprüfung des Ausdünnungskriterium

Für die Evaluierung des Ausdünnungskriteriums ist eine sortierte Schleife aller Knoten und Dreiecke, die das aktuelle Vertex benutzen, von Nöten. Diese Schleife wird im ersten Schritt des Algorithmus erzeugt. Je nachdem, welcher Klasse das Vertex angehört, werden nun zwei verschiedene Kriterien überprüft.

Für einfache Vertices wird ein Kriterium basierend auf Ebenenabstand benutzt. Dieses ist in Abbildung 5 dargestellt. Die Referenzebene wird mit Hilfe der Knoten gebildet, die das Vertex benutzen. Da eine Ebene oft nicht durch alle fünf Punkte gelegt werden kann, soll die Referenzebene einen möglichst geringen Abstand zu diesen Knoten haben. Daher kann die Ebene auch als *Durchschnittsebene* durch die beteiligten Knoten angesehen werden. Für das untersuchte Vertex wird nun der Ebenenabstand zu dieser Vergleichsebene berechnet, und ist dieser kleiner als ein Referenzwert, kann das Vertex und alle Dreiecke in der Schleife gelöscht werden.

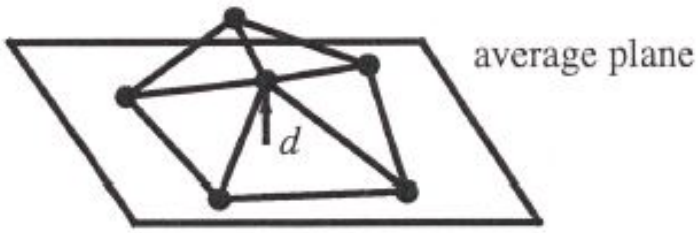


Abbildung 5: Ebenenkriterium für Vertices der Einfachen Klasse

Die entstandene Lücke wird im nächsten Schritt ohne das untersuchte Vertex erneut mit Dreiecken gefüllt.

Für Vertices der Begrenzung- und Inneren Kante-Klasse basiert das Kriterium auf der Abstandsbestimmung zu einer Kante (siehe Abbildung 6). Die beiden Knoten, die die Grenze des Netzes bilden, bzw. eine besondere Kante einschließen, definieren den Start- und Endpunkt der Referenzkante. Ist der Abstand des untersuchten Vertex geringer als der Referenzwert, wird das Vertex und die entsprechenden Dreiecke entfernt.

Die definierten besonderen Kanten bilden eine Besonderheit ab. Das Beibehalten dieser Kanten ist nicht immer gewünscht, da sehr viele besondere Kanten in einem Netz daraufhindeuten können, dass es viele kleine Dreiecke gibt, die zur Geometrie des Netzes wenig beitragen. Weiterhin können viele besondere Kanten auch das Resultat von Rauschen in den Daten sein. Daher werden Vertexe der Eck-Klasse und teilweise auch Vertexe der Inneren Kante-Klasse ebenfalls über das Ebenenabstandskriterium geprüft,

Abbildung 6: Linienkriterium für Begrenzungs- und Innere Kanten Vertices

um diese Ecken zu entfernen. Dieses Verhalten kann vom Benutzer des Algorithmus eingestellt werden.

Allgemein kann gesagt werden, dass durch diese Ausdünnungskriterien das Netz eingeebnet wird. Andere Kriterien, die ebenfalls verwendet werden können, haben andere Effekte.

2.3 Triangulation der Lücke

Für die Triangulation der entstandene Lücke muss zunächst die Schleife bearbeitet werden. War das untersuchte Vertex ein "Innere Kante"-Vertex, muss zunächst die Schleife entlang der besonderen Kante aufgeteilt werden und die entstandenen Hälften separat gefüllt werden.

Bei der Triangulation der Schleifen, bzw. der verbliebenen Knoten, können keine 2D-Algorithmen, wie z.B. Lawsons Methode, verwendet werden, da die Knoten in der Regel nicht innerhalb einer Ebene liegen. Um eine Triangulation zu erreichen, die die ursprüngliche Schleife sehr gut annährt und keine sich schneidenden Dreiecke erzeugt, erwies es sich als effektiv, die Schleife rekursiv zu teilen.

Der Teilungsalgorithmus ist in Abbildung 7 dargestellt. Die Schleife wird an einer Linie in zwei Hälften geteilt, die zwei nicht benachbarte Knoten verbindet (im Bild als *Split Line* bezeichnet). Um zu entscheiden, in welcher Hälfte der Schleife die Punkte liegen, wird auf die Bestimmung des Halbraums einer Ebene zurückgegriffen. Im Bild ist eine *Split Plane* eingezeichnet, anhand welcher die Aufteilung der Schleife vorgenommen wird. Alle Knoten im positiven Halbraum dieser Ebene bilden eine neue Schleife, sowie alle Elemente im negativen Halbraum. Die Teilungsebene ist dadurch definiert, dass sie die Teilungslinie beinhaltet und orthogonal zur Referenzebene aus Schritt eins des Algorithmus ist. Diese Aufteilung der Schleifen wird durchgeführt, bis eine Schleife lediglich

aus 3 Knoten besteht. Diese drei Knoten bilden ein Dreieck, dass dem Netz hinzugefügt wird und den rekursiven Prozess beendet.

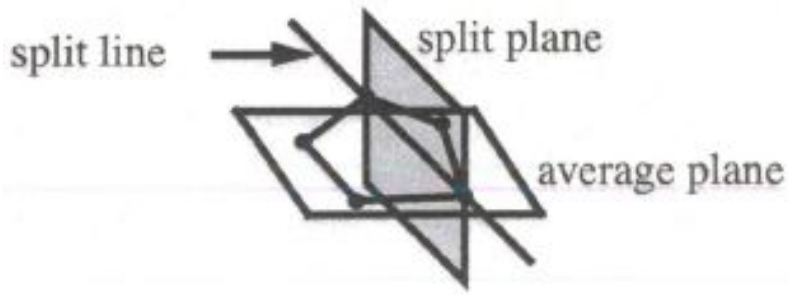


Abbildung 7: Teilungsalgorithmus

Im Bild ist bereits zu sehen, dass es mehr als ein Paar nicht benachbarter Knoten gibt, die als Ausgangspunkt für die Teilungslinie dienen können. In diesem Fall muss herausgefunden werden, welche Linie sich am besten eignet. Dafür wurde ein Kriterium entwickelt, dass sich am Verhältnis der Distanz aller Knotenpunkte zur Teilungsebene und der Länge der Teilungslinie orientiert. Dieses Verhältnis lässt sich als Formel ausdrücken:

$$ar = \frac{min(d_{v,sp})}{len(sl)} \text{ , } sp: \ split\ plane \text{ , } sl : split\ line \text{ , } v : vertex$$

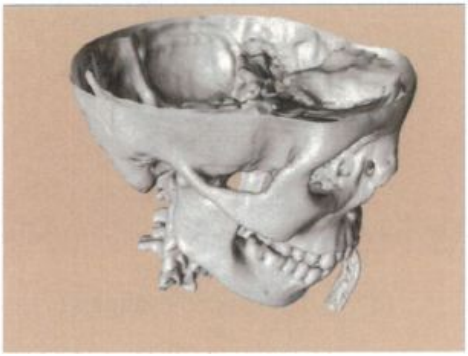
Es wird also die minimale Distanz der Knoten in der Schleife zur Teilungsebene durch die Länge der Teilungslinie dividiert. Die Teilungsebene, für die der Wert von ar ein Maximum erreicht, ist die zu wählende Teilungsebene.

Insgesamt ist dieser Triangulationsalgorithmus recht anfällig gegenüber Fehlern. So kann es passieren, dass eine Aufteilung der Schleife in zwei Teile ohne Überlappung nicht möglich ist. In solchen Fällen wird die Triangulation abgebrochen und das untersuchte Vertex nicht entfernt. Weiterhin wird bei der Triangulation überprüft, dass keine Dreiecke oder Kanten doppelt existieren. Dadurch können keine neuen Verbindungen innerhalb des Netzes entstehen und die Topologie wird gewahrt.

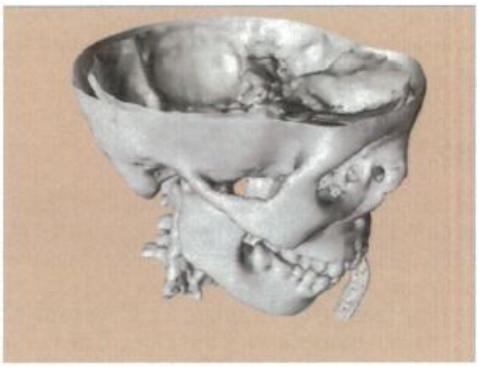
2.4 Ergebnisse

Abbildung 8(a) zeigt ein Netz, dass mit dem Marching Cubes Algorithmus erzeugt wurde. Die Datengrundlage besteht aus 256x256 Pixel und verfügt über 93 Schichten. Aus diesen Daten wurde ein Dreiecksnetz erzeugt, dass über 569.000 Dreiecke verfügte. Die Abbildungen 8(b)-(d) zeigen Ergebnisse des Ausdünnungsalgorithmus. So ist in Abbildung 8(b) zu sehen, dass unter Verwendung des gleichen Beleuchtungsmodells und eine

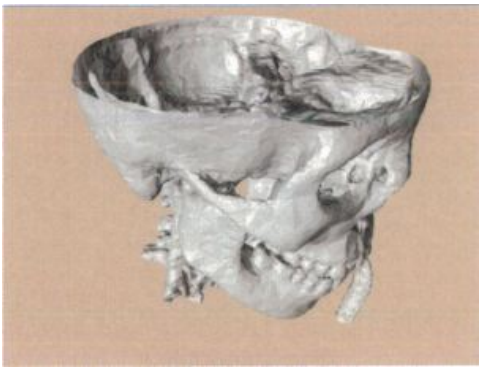
Reduktion um 75% der Dreiecke kein Unterschied zum originalen Netz zu erkennen ist. Erst durch ein Wechsel des Beleuchtungsmodell auf eine flache Schattierung werden erste Verschlechterungen der Qualität sichtbar. Die Oberfläche wirkt nicht mehr glatt, sondern rauher mit mehr Sprüngen. Alles in allem kann man aber sagen, dass dieses Modell noch sehr gut für etwaige Anwendungen benutzt werden kann. Kritischer sieht es da schon in Abbildung d aus. Bei dieser Grafik sind nur noch 10% der ursprünglichen Dreiecke vorhanden. Hier zeigen sich bereits Einbußen der Qualität, wenn es um komplexe Geometrien geht. So ist der Detailgrad des Wangenknochens, der Wirbelsäule und bei komplexeren Gebilden im Inneren des Schädels nicht mehr komplett gegeben. Unter Betrachtung der Tatsache, dass 90% der ursprünglichen Dreiecke entfernt wurden, ist dies aber immer noch eine gute Annährung an das Original.



(a) Original - 569.000 Dreiecke (Gouraud-Shading)



(b) 75% reduziert - 142.000 Dreiecke (Gouraud-Shading)



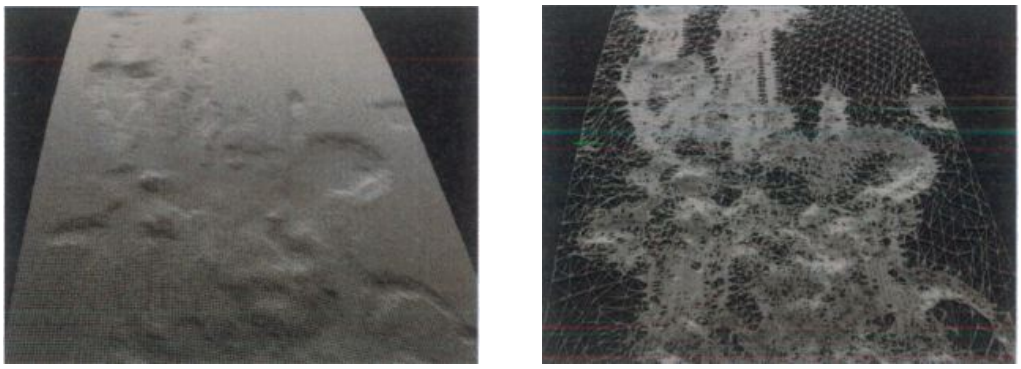
(c) 75% reduziert - 142.000 Dreiecke (Flat-Shading)

(d) 90% reduziert - 57.000 Dreiecke (Flat-Shading)

Abbildung 8: Ergebnisse von Schroeder et al. I

Interessant ist der Vergleich der Bilder in Abbildung 9. Hierbei handelt es sich um ein Bild der Marsoberfläche. Neben dem originalen Netz, stand den Autoren auch ein Netz

zur Verfügung, dass bereits in eine geringere Auflösung umgerechnet wurde. Diesen Datensatz verglichen sie mit dem Ergebnis ihres Algorithmus, der so eingestellt wurde, dass eine ähnliche Anzahl an Dreiecken übrig geblieben ist. Beide Bilder liegen als Drahtlinienmodell vor, so dass die Dreiecke gut erkennbar sind. Die umgerechnete Version des Modells verfügt über eine gleichmäßige Abdeckung an Dreiecken. Planare Flächen und Krater verfügen über gleichermaßen viele Dreiecke. Beim Ergebnis des Algorithmus hingegen kann man erkennen, dass große planare Flächen durch wesentlich weniger Dreiecke dargestellt werden. Bei Details, wie Kratern oder Schluchten, ist jedoch eine Konzentration an Dreiecken zu erkennen, so dass diese Strukturen über wesentlich mehr Details verfügen. Somit ist das Ergebnis des Algorithmus wesentlich detailgetreuer mit weniger Dreiecken, als es die umgerechnete Version der Originaldaten ist.



(a) Gerechnete niedrige Auflösung - 68.000 Dreiecke

(b) 77% reduziert - 62.000 Dreiecke

Abbildung 9: Ergebnisse von Schroeder et al. II

Hier zeigt sich auch eine Stärke dieses Algorithmus, bzw. der topologischen Operation der Vertexausdünnung. Diese Art ist sehr gut geeignet, um Netze, die mit dem Marching Cubes Algorithmus generiert wurden, zu bearbeiten.

2.5 Weiterentwicklung

Als eine Weiterentwicklung des vorgestellten Algorithmus kann man das allgemeine Framework zur Netzausdünnung bezeichnen, dass 1998 von der Universität Erlangen vorgestellt wurde [3]. In ihrer Arbeit untersuchten die Autoren eine Vielzahl von Ausdünnungsalgorithmen, um die wesentlichen Elemente eines solchen Algorithmus herauszustellen und die Implementierung eines Ausdünnungsalgorithmuses als Baukasten zu abstrahieren. Weiterhin implementierten sie anhand dieser Kriterien einen Beispielalgorithmus,

deren Ergebnisse hier kurz gezeigt werden.

Die Autoren identifizierten drei wesentliche Aspekte, die einen lokalen Ausdünnungsalgorithmus ausmachen. Zum einen der topologische Operator, dann eine Distanzmessung, um den maximal zulässigen Fehler nicht zu überschreiten, und als letzten Aspekt ein Fairnesskriterium, dass die Qualität des aktuellen Netzes misst (vgl. [3]).

Die Wahl des topologischen Operators stellte sich nicht als der entscheidende Faktor heraus. Die Ergebnisse bei unterschiedlichen Netzen waren sich immer ähnlich. Viel entscheidender für das Ergebnis, ist die Entscheidung, an welcher Stelle des Netzes die Ausdünnung als nächstes vorgenommen wird. Aufgrund dieser Tatsache sollte der topologische Operator möglichst einfach sein, weshalb in der Beispielimplementierung der Autoren das Zusammenfallen von Halbkanten gewählt wurde.

Die Distanzmessung dient als Fehlerfunktion und misst die Toleranz zwischen originalem und aktuellen Netz. Dabei darf ein bestimmter maximaler Wert nicht überschritten werden, da sonst das Netz für nachfolgende Anwendungen nicht mehr brauchbar wäre. Eine einfache Realisierung dieser Distanzmessung erfolgt durch die lokale Aufsummierung der Abweichung in den einzelnen Reduktionsschritten. Dabei handelt es sich um eine sehr konservative Berechnung, die den Fehler überschätzt.

Eine genauere Berechnung des wirklichen Fehlers kann über die Berechnung der Hausdorff-Distanz[8] erfolgen. Diese Methode berechnet den Fehler wesentlich genauer, jedoch ist dies eine mathematisch sehr aufwendige Berechnung und das Eingabenetz ist nicht immer sehr gut geeignet. Daher wurde dieser Schritt in der Beispielimplementierung der Autoren vereinfacht. Zunächst wird der Suchraum zur Berechnung des Fehlers lokal auf das von der Ausdünnung betroffene Subnetz beschränkt. Anschließend wird das Wissen ausgenutzt, dass man die Distanz einer Punktwolke zu einem Fächer an Dreiecken berechnen möchte (vgl. [3]). So kann man einen Halbraumvergleich mit den Punkten der Punktwolke durchführen und dadurch den Slot mit minimaler Distanz zu einem Dreieck finden (siehe Abbildung 10).

Ein entscheidender Punkt in der Arbeit von Kobbelt et al. ist die Einführung des Fairnesskriteriums. Dieses Kriterium führt eine Unterscheidung zwischen dem Fehler, der durch die Reduktionsschritte eingeführt wird und dem Einfluss auf die Fairness des resultierenden Netzes. Bisher wurde in der Regel beides zusammengelegt für die Entscheidung, wo der nächste Reduktionsschritt durchgeführt wird. Die abstrakte Sichtweise, dass ein Netz R gesucht wird, dass dem Netz O maximal um den Fehler ϵ abweicht,

[8] Die Hausdorff-Distanz berechnet die Entfernung einer Punktwolke zu einer anderen. Dafür werden die Abstände aller Kombinationen der Punkte berechnet und aufsummiert.

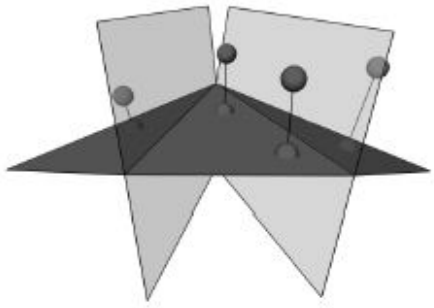


Abbildung 10: Minimale Distanz mit Halbraumvergleichen [3]

lässt lokale Ausdünnungsalgorithmen als *Greedy*[9]-Algorithmen erscheinen. Jedoch ist der Zusammenhang wesentlich komplexer. Reduktionsschritte, die in der Vergangenheit durchgeführt wurden, bestimmen die möglichen Reduktionsschritte zum Wahlzeitpunkt (vgl. [3]). Die Einführung des Fairnesskriteriums ermöglicht nun eine Sortierung aller durchzuführenden Reduktionsschritte im vornherein. Dadurch werden mehr Informationen in die Entscheidung miteinbezogen und die Entscheidungen sind durch Fakten abgesichert.

Somit werden im Wesentlichen drei Funktionen herausgestellt:

- Kostenfunktion (Aussage über die übrig gebliebene Komplexität)
- Kapazitätsfunktion (Setzt vorgeschriebene Toleranz durch)
- Entscheidungskriterium (Kante soll zusammengeführt werden oder nicht)

Für das Fairnesskriterium spielen geometrische Aspekte eine wichtige Rolle. Beispiele für Kriterien, nach denen gewichtet werden kann, sind der lokale Fehler, der eingeführt wird, oder die Verzerrung und die Krümmung der Dreiecke. Somit können z.B. Dreiecke mit einer starken Verzerrung "bestraft" werden, indem sie erst sehr spät in der Reduktion berücksichtigt werden. Eine Fairnessfunktion nach der eine Gewichtung durchgeführt werden kann und die den lokalen Fehler, die Verzerrung und die Krümmung miteinbezieht, sieht wie folgt aus:

$$F(R) := \sum_{p \in R} \alpha E(p) + \beta R(p) + \gamma S(p)$$

Über die Faktoren α, β, γ kann die Gewichtung der einzelnen Faktoren (Fehler, Verzerrung, Krümmung) geändert werden. Die Wahl des Fairnesskriteriums ist entscheidend

[9]Greedy-Algorithmen sind Algorithmen, die zum Wahlzeitpunkt den Folgezustand auswählen, der das beste Ergebnis verspricht

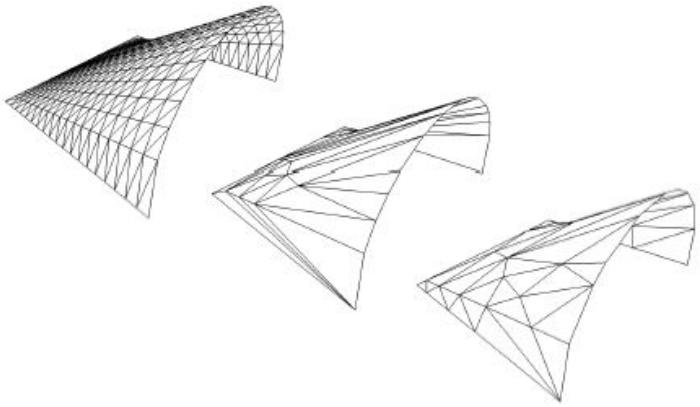
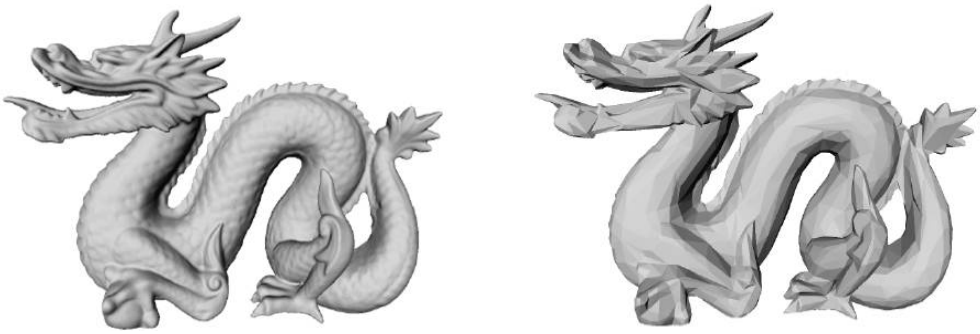


Abbildung 11: Auswirkung verschiedener Fairnesskriterien [3]

für die Auswirkung auf das Ergebnis und ermöglicht so dem Benutzer eine Steuerung des Algorithmus. Abbildung 11 zeigt das Ergebnis einer Reduktion mit unterschiedlichen Kriterien. Links ist das originale Netz gezeigt, in der Mitte das Ergebnis unter Bestrafung großer V-Winkel und rechts unter Bestrafung verzerrter Dreiecke.



(a) Original - 871.414 Dreiecke (b) 99% reduziert - 7402 Dreiecke

Abbildung 12: Ergebnisse von Kobbelt et al. [3]

Abbildung 12 zeigt das Ergebnis der Beispielimplementierung der Autoren. Wie man sieht, ist das reduzierte Netz mit nur noch 0.85% der originalen Dreiecksanzahl eine sehr gute Approximation. Lediglich genaue Oberflächendetails, wie die Schuppenstruktur des Drachenkörpers und kleinere Verzierungen sind nicht mehr zu erkennen. Insgesamt ist der Drache unter Berücksichtigung der massiven Reduktion aber noch sehr gut erkennbar. Die Durchführung dieser Reduktion auf einer SGI R10000 mit 195 Mhz benötigte ca. 585 s und ist damit auch performant.

3 Geometry Images

Ein gänzlich anderer Ansatz zur Darstellung komplizierter Netze wurde von Gu et al. in 2002 vorgestellt [2]. Die Netze werden in quadratische 2D-Bilder, sogenannte *Geometry Images*, umgewandelt. In den Farbkanälen Rot, Grün und Blau werden die X, Y und Z-Koordinaten der vorherigen Pixel kodiert. Für weitere Informationen, wie z.B. die Normalen der einzelnen Oberflächen, werden ebenfalls weitere quadratische 2D-Bilder verwendet.

Um diese Konvertierung zu ermöglichen, ist eine umfassende Untersuchung des Netzes nötig, damit eine Parametrisierung erstellt werden kann. Mit Hilfe dieser Parametrisierung ist es anschließend möglich, geeignete Schnittkanten zu finden. Da es sich bei den Geometry Images um quadratische Bilder handelt, muss auch das Originalnetz zunächst derart aufgeschnitten werden, dass es einer Kreisscheibe entspricht, die anschließend zu einem Quadrat gestreckt wird (vgl. Abbildung 13). Das Finden dieser Schnittkanten ist der problematischste Prozess bei der Konvertierung von Dreiecksnetzen, da die Netze zum einen in die passende Form gebracht werden müssen und zum anderen an den Schnittkanten keine doppelten Elemente entstehen dürfen.

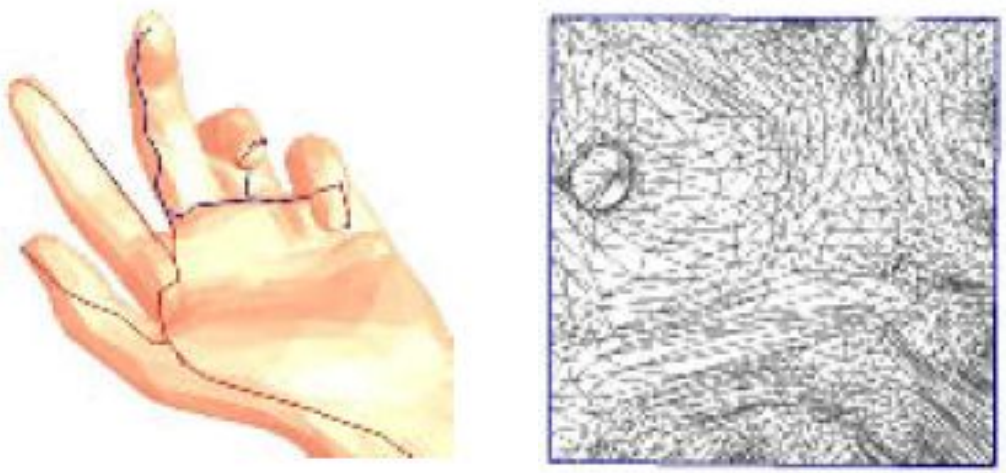
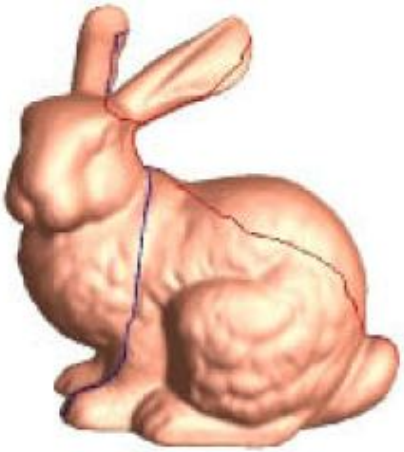


Abbildung 13: Aufschneiden eines Netzes [2]

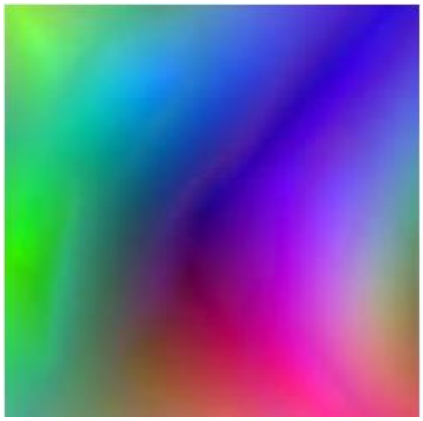
Die Verwendung von Geometry Images bietet einige Vorteile. So handelt es sich bei diesen Bildern um gewöhnliche 2D-Bilder, wie sie in der Computergrafik häufig vorkommen. Dies bedeutet, dass die Grafikhardware eine Bearbeitung dieser Bilder unterstützt. Die Bilder können in die Grafikpipeline eingefügt werden, und eine Rekonstruktion des Originalnetzes kann hardwareunterstützt durchgeführt werden. Weiterhin wird hier eine Brücke zur digitalen Bildverarbeitung geschlagen. Anstatt komplizierte Ausdünnungsalgorithmen auf ein Netz anzuwenden, können im Fall der Geometry Images bereits existierende Kompressionsalgorithmen für Bilder verwendet werden. Werden die ursprünglichen Geometry Images komprimiert, ist auch das Netz vereinfacht. Ändert man am Geometry Image nichts, kann auch das Originalnetz genau nachgebildet werden.

Weiterhin können die Netze in Geometry Images mit relativ wenig Speicheraufwand aufbewahrt werden. Im Paper von Gu et al. wurden Netze auf Geometry Images mit einer Auflösung von 257x257 Pixel abgebildet. Hinzu kommt noch ein Bild für die Speicherung der Normalen. Hier wurde mit einer Auflösung von 512x512 Pixeln gearbeitet. In Abbildung 14 ist ein Beispiel der Anwendung von Geometry Images gezeigt. Es ist zu erkennen, dass sowohl eine komplette Rekonstruktion des Netzes anhand des originalen Bildes möglich ist, wie auch eine komprimierte Darstellung, die ebenfalls noch gut aussieht und lange kleinere Oberflächendetails behält.

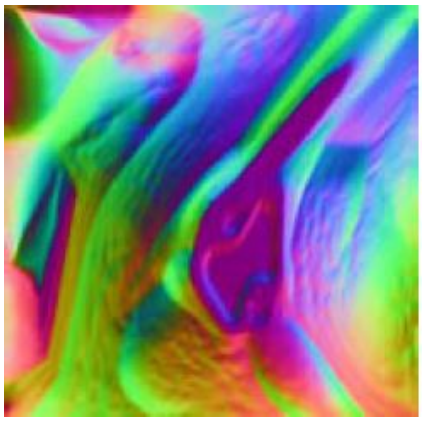
Nachteilig an den Geometry Images kann gesehen werden, dass sie nicht auf Nicht-Mannigfaltige Netze angewandt werden können und die Erstellung eines solchen Bildes im ersten Schritt wesentlich zeitaufwändiger ist, als die Anwendung eines Ausdünnungsalgorithmus. So dauert die Umrechnung eines Netzes mit ca. 500.000 Dreiecken ungefähr eine Stunde.



(a) Original - 70.000 Dreiecke

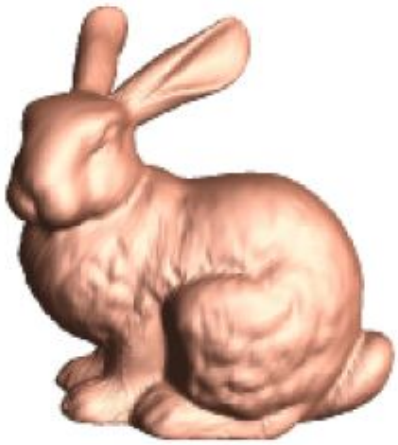


(b) Geometry Image (257x257) (49KB)



(c) Geometry Image der Normalen (512x512)

(d) Rekonstruktion



(e) Rekonstruktion von 12KB Geometry Image

(f) Rekonstruktion von 1.5KB Geometry Image

Abbildung 14: Ergebnisse von Gu et.al. [2]

4 Zusammenfassung

Zusammenfassend kann man sagen, dass effiziente Ausdünnungsalgorithmen nötig sind. Im Bereich der Anwendungen im medizinischen, industriellen und wissenschaftlichen Bereich ist eine flüssige Interaktion mit Modellen notwendig. Zwar hat sich die Rechenleistung im Grafikbereich erheblich verbessert, so dass heutige Grafikkarten aktueller Generationen auf Spitzenwerte von über 3 Milliarden Dreiecke pro Sekunde kommen, im Durchschnitt aber auch einige hundert Millionen bis 1 Milliarde Dreiecke[10]. Im gleichen Maß hat sich allerdings auch die Scantechnik verbessert, so dass heutige Computertomographen bis zu 640[11] Schichten erstellen können. Auch im Bereich der Spieleengines werden Polygon-, bzw. Dreiecksnetze zur Modellierung von Objekten verwendet und dort können effiziente Algorithmen ebenfalls dabei helfen, den Rechenaufwand zu verringern.

Im Bereich der Ausdünnungsalgorithmen zeigt sich auch die zeitliche Weiterentwicklung der Verfahren. So wurde die Auswahl der topologischen Operatoren von der Vertexausdünnung, die zum ersten Mal von Schroeder et al. vorgestellt wurde, über die Kantenkontraktion hin zum Zusammenfallen von Halbkanten kontinuierlich verbessert. Weiterhin wurde der Aufbau dieser Algorithmen immer weiter abstrahiert, so dass mittlerweile eine Art Baukasten für Ausdünnungsalgorithmen besteht, der die verschiedenen Kategorien der Implementierung beschreibt. Dadurch können auch verschiedene Teile der Algorithmen, die unterschiedliche Aspekte implementieren, kombiniert werden.

Mit der Einführung der Geometry Images wurde noch ein weiteres Feld zur effizienten Darstellung komplizierter Netze auf dem Bildschirm eingeführt. Hierbei wurde eine Verknüpfung zur digitalen Bildverarbeitung geschaffen, die es ermöglichte, bereits vorhandene, optimierte und erprobte Kompressionsalgorithmen zu benutzen, um Netze zu reduzieren.

[10] http://www.pcgameshardware.de/aid,817166/Geforce-GTX-590-im-Test-Triumphiert-der-doppelte-Fermi-ueber-die-Radeon-HD-6990/Grafikkarte/Test/ - 18.12.2011

[11] http://www.toshiba-medical.de/computertomographie/aquilion-one - 18.12.2011

Literaturverzeichnis

[1] D. Fellner and T. Kalbe. Graphische Datenverarbeitung II. Vorlesung TU Darmstadt, 2009. `http://www.gris.tu-darmstadt.de/teaching/courses/ss09/gdv2/slides/`, Stand 27.03.2012.

[2] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 355–361, New York, NY, USA, 2002. ACM.

[3] Leif Kobbelt, Swen Campagna, and Hans-Peter Seidel. A General Framework for Mesh Decimation. In *Proceedings of the Graphics Interface Conference '98*, pages 43–50, 1998.

[4] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of Triangle Meshes. *SIGGRAPH*, pages 65–70, 1992.

[5] Jerry O. Talton. A Short Survey of Mesh Simplification Algorithms. Course notes, University of Illinois at Urbana-Champaign, 14.10.2004. `https://truesculpt.googlecode.com/hg/Doc/mesh_simplification.pdf`, Stand: 27.03.2012.