Segmentierung und Approximation großer Punktwolken

Vom Fachbereich Mathematik
der Technischen Universität Darmstadt
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)
genehmigte

Dissertation

von

Dipl. Math. Wilhelm Wilke
aus Korbach

Referent :	Prof. Dr. rer. nat. Josef Hoschek
Korreferent :	Prof. Dr. rer. nat. C. Werner Dankwort

Tag der Einreichung :	19.01.2000
Tag der mündlichen Prüfung :	24.11.2000

Darmstadt 2002
D 17

Abstract

In my PhD thesis *Segmentation and Approximation of large point clouds* I created an new approach for the automation of the point cloud segmentation process. **Surface reconstruction** or **Reverse Engineering** describes the process to get a mathematical representation of a measured object (based on B-Spline surfaces) from an unorganized set of 3D points (**point cloud**). This process is nowadays interactively done in **C**omputer **A**ided **D**esign (CAD) - Systems and will take a long time of interactive work for the correct selection of a net of curves (**Segmentation**) where the surface curvature changes.

Therefore in this thesis I developed new algorithms and methods for an automatic segmentation of point clouds based on geometrically criteria. The user has to determine some intuitive thresholds and the algorithm creates automatically regions with similar curvature behaviour. The boundary lines of this regions are called **feature lines** and may be reused in CAD-systems for the creation of approximation surfaces. This methods leads to better surface approximation quality and strongly accelerates the surface reconstruction process.

The first step of the automatic surface reconstruction of an unorganized point cloud is the piecewise linear approximation of the points with a set of connected triangles. Therefore I developed two new **Triangulation** algorithm which computes automatically an triangulation of arbitrary 3D point clouds (scattered data) in a very short time. Theses triangulations may be treated as input for post-processing steps like **Rapid Prototyping** and the direct calculation of tool pathes. This triangulation may also be used as input for applications in the **Virtual Reality** area. For these three applications the surface reconstruction is finished with the complete triangulation of the point cloud. Moreover I use the neighbouring information between the points which is implicit given by the triangulation to caculate an effective and robust estimation of the surface curvature on the measured point cloud. At first there were filters applied for finding points with measurement errors and projecting these points a step in the best fit plane to the points from all neighboured triangles (**fairing** the point cloud). Afterwards we have to inspect the triangulated cloud whether it contains points which may be removed from the triangulation without loosing accuracy (**data reduction**). In flat areas of the object there are less triangles needed for an accurate surface description as in curved areas of the object or at sharp corners. These, for the surface structure unnecessary points, will be removed from the triangulation. The last step is an attempt to find automatically edges and natural segment boundaries (**feature lines**) on the triangulation. The feature lines are curves on the triangulation where the curvature behaviour of the underlying surfaces changes. For the determination of this feature line curves I expanded an **Region Growing** algorithm for 2D-Image segmentation to work on the **segmentation** of 3D triangulated point clouds. With these feature lines and some user defined continuity constrains between the neighboured regions of the feature line it is possible to make another automatic step for the surface reconstruction process of the point cloud. Examples for the surface reconstruction workflow with the developed methods are shown on measured point clouds from car models.

Vorwort

Ziel dieser Arbeit ist es, den zur Zeit noch interaktiven und sehr langwierigen Prozeß der Segmentierung von Punktwolken zu automatisieren. Der Begriff **Flächenrückführung** bezeichnet die Verfahren aus einer gegebenen Menge von Meßpunkten eines dreidimensionalen Objektes eine Flächenbeschreibung des Objektes auf der Basis von parametrisierten Flächen zu erzeugen. Dieses Flächenmodell muß in einem **C**omputer **A**ided **D**esign (CAD) - System auf Basis von B-Spline Flächen weiterverarbeitet werden können. Daher habe ich in dieser Arbeit Algorithmen und Verfahren entwickelt, die unstrukturierten Punktwolken nach Angaben von intuitiven Schwellenwerten automatisch in einzelne Regionen zerlegen. Diese Zerlegung erfolgt nach geometrischen Kriterien, das heißt, daß an allen Punkten einer Region ein ähnliches Krümmungsverhalten auftritt. Die erzeugten Regionen und deren Segmentgrenzen können in CAD - Systemen weiterverarbeitet werden, führen zu besseren Approximationsergebnissen und können den Prozeß der Flächenrückführung stark beschleunigen.
Als erster Schritt zu einer stückweisen, linearen Approximation der Punktwolke durch Dreiecke können Tringulierungen der Punktwolke angesehen werden. In dieser Arbeit habe ich zwei neue, schnelle Triangulationsverfahren für unstrukturierte Punktwolken entwickelt und implementiert. Diese **Triangulierungen von Punktwolken** können in den folgenden Anwendungen direkt weiterverarbeitet werden : Verfahren zur Erzeugung des Objektes (direktes Fräsen auf einer Triangulierung), generativen Fertigungsverfahren aus dem Bereich des **Rapid Prototyping** [Stereolithographie] und Visualisierungsmethoden aus dem Bereich der **Virtual Reality**.
Mit den durch die Triangulierung implizit gegebenen Nachbarschaftsbeziehungen zwischen einzelnen Punkten, ist eine effiziente und schnelle Art der Krümmungsabschätzung für diese triangulierten Punktwolken möglich. Diese Krümmungswerte können auch für Verfahren der **Glättung** und **Datenreduktion** der triangulierten Punktwolken verwendeten werden.
Auf diese triangulierte Punktwolke, die mit Krümmungswerten versehen wurde, werden im nachfolgenden Bearbeitungsschritt **Segmentierungsverfahren** angewendet. Diese auch als **Region Growing** Verfahren bezeichneten Segmentierungsalgorithmen stammen aus der 2D-Bildverarbeitung und wurden von mir auf dreidimensionale, triangulierte Punktwolken verallgemeinert. Die bei der Segmentierung der Punktwolke auftretenden Berandungskurven - auch als **Feature Lines** bezeichnet - werden im abschliessenden Bearbeitungsschritt zum automatischen Erzeugen von Flächenmodellen verwendet. Anhand einiger Beispiele von gemessenen Fahrzeugmodellen wird gezeigt, wie ich die mit dem Segmentierungsverfahren erzeugten Berandungskurven zur Verbesserung und Beschleunigung des **Reverse Engineering** Prozesses verwendet habe.

Für Andrea

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation 1

1.2 Stand der Technik 3

1.3 Überblick der Arbeit 4

2 Optische Messung von 3D - Objekten **9**

2.1 Einführung in die optische 3D - Meßtechnik 9

2.1.1 Das Streifenprojektionsverfahren 9

2.1.2 Fabriktauglichkeit optischer Meßsysteme 10

2.1.3 Photogrammetrie 11

2.2 Gewinnung und Analyse von Tiefenbildern 12

2.3 Funktionsprinzip optischer Sensoren 12

2.4 Beispiele für die Realisierung von Optischen 3D - Meßsystemen 18

3 Datenstruktur für große Punktwolken **23**

3.1 Entwicklung einer Datenstruktur für große Punktwolken 23

3.1.1 Möglichkeiten zur Strukturierung großer Punktwolken 23

3.1.2 Überblick von Anwendungen auf 3D - Punktwolken 24

3.1.3 Anforderungen an eine Datenstruktur für große Punktwolken 25

3.2 Vorstellung verschiedener Datenstrukturen für 3D - Punktwolken 27

3.2.1 Nichthierarchische Datenstrukturen 28

3.2.2 Hierarchische Datenstrukturen 29

3.2.3 Die Fixed - Grid Datenstruktur 31

3.2.4 Der Region - Octree 32

3.2.5 Der Matrizen - Graph 33

3.3 Die Quader - Octree Datenstruktur 35

4 Triangulierung von 3D - Punktwolken **39**

4.1 Verschiedene Verfahren zur Triangulierung von 3D - Punktwolken 39
4.1.1 Räumliche Delaunay - Triangulierung 39
4.1.2 Alpha Shapes 40
4.1.3 Räumliche Triangulierung verschiedener Ansichtsbilder 41
4.1.4 Triangulierung von Punktwolken in Scan - Linien Form 42
4.1.5 Voxelbasierte Konstruktion der räumlichen Triangulierung 43
4.2 Algorithmus zur vollständigen Triangulierung von 3D - Punktwolken 44
4.2.1 Delaunay - Triangulierung ebener Punktwolken 45
4.2.2 Verbinden der Triangulierungen der einzelnen Quader 50
4.2.3 Komplexitätsabschätzungen zum Verfahren der vollständigen Triangulierung . 52
4.2.4 Fehlerbetrachtungen zum Verfahren der vollständigen Triangulierung 54
4.3 Erzeugung einer Topologie - Triangulierung 56
4.3.1 Voraussetzungen für die Topologie - Triangulierung 56
4.3.2 Startpunktsuche für die Topologie Triangulierung 58
4.3.3 Schrittweise Erweiterung der Triangulierung 58
4.3.4 Fehlerbetrachtung für die Topologie Triangulierung 61
4.3.5 Komplexitätsabschätzungen der Topologie Triangulierung 64
4.4 Vergleich der Triangulationsverfahren für 3D - Punktwolken 65

5 Krümmungsabschätzung auf triangulierten Punktwolken 66

5.1 Möglichkeiten zur Krümmungsabschätzungen auf Punktwolken 66
5.2 Normalenvektorabschätzungen auf der Triangulierung 69
5.2.1 Direkte Normalenvektorberechnung auf Triangulierungen 69
5.2.2 Bestimmung der Normalenvektoren in den Punkten der Triangulierung 70
5.2.3 Normalenvektorberechnung mit Ausgleichsebenen 77
5.3 Krümmung als Winkeländerungen zwischen Normalenvektoren 78
5.4 Direkte Abschätzung der Gaußschen Krümmung 83
5.5 Krümmungsabschätzung über Schlitzwinkel 85
5.6 Beispiele für die verschiedenen Krümmungsabschätzungen 86

5.6.1 Vergleich der Krümmungsgabschätzungen 89
5.7 Anwendungen der Krümmungsabschätzung 90
5.7.1 Datenreduktion bei triangulierten Punktwolken 90
5.7.2 Glättung der triangulierten Punktwolke 93

6 Segmentierung von triangulierten Punktwolken 94

6.1 Möglichkeiten zur Segmentierung von räumlichen Punktwolken 94
6.2 Segmentierungsverfahren aus der digitalen Bildverarbeitung 96
6.2.1 Operatoren aus der Bildverarbeitung 97
6.3 Der Kantenverfolgungsalgorithmus 100
6.3.1 Der Kantenverfolgungsalgorithmus auf Triangulierungen 101
6.3.2 Beispiele des Kantenverfolgungsalgorithmus 102
6.3.3 Bewertung der Kantenverfolgung zur Segmentierung von 3D - Punktwolken . . 103
6.4 Der Region - Growing Algorithmus 104
6.4.1 Voraussetzungen für den Region - Growing Algorithmus 104
6.4.2 Der Region - Growing Algorithmus zur Bestimmung von Feature Lines 105
6.5 Approximation von Polygonzügen mit glatten B-Spline Kurven 109
6.6 Glättung des Feature Line Polygone mit Wavelets 113
6.6.1 Wavelet Filterung räumlicher Polygonzüge 113
6.6.2 Geometrische Bedeutung der Filtermatrizen 114
6.6.3 Fehlerabschätzung bei Waveletfilterung von Polygonen 115
6.6.4 Beispiele der Wavelet Filterung und B-Spline Approximation 116
6.6.5 Vergleich der Wavelet Filterung mit der Approximation durch B-Splines 122
6.7 Beispiele zur Feature Line Erkennung mit Region - Growing 123

7 Strategien zur Flächenerzeugung mit Feature Lines 125

7.1 Flächengenerierung mit Hilfe der Feature Lines 125
7.2 Approximationsverfahren für die segmentierten Regionen 128
7.2.1 Vergleich der Approximation mit Trimmkurven und Feature Line Kurven . . . 130
7.3 Beispiele der Flächenerzeugung 131

7.3.1 Datensatz THD3 . . . 134
7.3.2 Datensatz Mickey Maus . . . 136
7.3.3 Datensatz SLKTOPO . . . 137
7.3.4 Datensatz THD1, Firma Holometric Technologies, Aalen . . . 139
7.3.5 Datensatz Blechteil . . . 141
7.3.6 Datensatz Wave . . . 143
7.4 Bewertung der verschiedenen Approximationsstrategien . . . 144

8 Ausblick 146
8.1 Zusammenfassung der Vorgehensweise zur Approximation und Segmentierung . . . 146
8.2 Verbesserungen und Erweiterungen der vorgestellten Verfahren . . . 152
8.3 Ausblick auf zukünftige Anwendung der Ergebnisse dieser Arbeit . . . 153

9 Literatur 159

A Übersicht zum Innovationsprojekt Reverse Engineering 169

B Grundlagen aus der geometrischen Datenverarbeitung 172
B.1 Berechnung von Ausgleichsebenen einer Punktwolke . . . 172
B.2 Grundlagen aus der Theorie der B-Spline Kurven und Flächen . . . 173
B.2.1 B-Spline Basis Funktionen . . . 173
B.2.2 Darstellung von B-Spline Kurven . . . 174
B.2.3 Tensorprodukt B-Spline Flächen . . . 176
B.2.4 Approximation mit B-Spline Kurven . . . 177
B.2.5 Approximation mit Tensorprodukt B-Spline Flächen . . . 179
B.2.6 Parameterkorrektur bei der Approximation von Kurven und Flächen . . . 182
B.2.7 Approximation von Trimmkurven . . . 185

C Grundlagen aus der Wavelet Theorie 186
C.1 Einführung in die Theorie der Wavelets . . . 186
C.2 Konstruktion der Analyse und Synthesematrizen . . . 190

1 Einleitung

1.1 Motivation

Der Begriff Flächenrückführung (engl. *Surface Reconstruction*) bezeichnet die Verfahren aus einer gegebenen Menge von Meßpunkten eines dreidimensionalen Objektes eine Flächenbeschreibung des Objektes auf der Basis von parametrisierten Flächen zu erzeugen. Dieses Flächenmodell muß in einem **C**omputer **A**ided **D**esign (CAD) - System weiterverarbeitet werden können und es muß möglich sein, aus dem Flächenmodell Daten für die Fertigung des Objektes zu erzeugen (z.B. NC - Fräsbahnen) oder das Flächenmodell für Zwecke der Finite Elemente Simulation (z.B. Strömungsuntersuchungen) aufzubereiten. Der Typ der parametrisierten Flächen hängt von der internen Repräsentation der Flächenmodelle im verwendeten CAD - System ab, meistens werden aber Tensorprodukt B-Spline Flächen verwendet, die teilweise auch durch Trimmkurven begrenzt werden können.
Während der herkömmliche Konstruktionsprozeß, Erstellen eines CAD - Modells am Rechner und anschließender computerunterstützte Fertigung (**C**omputer **A**ided **M**anufacturing), als *Engineering* bezeichnet wird, beschreibt der Begriff *Reverse Engineering* den Prozeß durch die Digitalisierung eines physikalisch vorhandenen Objektes (Werkstück) eine rechnerinterne Darstellung als CAD - Modell zu erzeugen, aus dem dann die notwendigen Daten für eine Einzel- oder auch Serienfertigung abgeleitet werden können (3D - Kopie des Werkstückes). Im ersten Schritt wird eine Menge von Meßpunkten durch Digitalisieren des physikalischen Modells eines dreidimensionalen Werkstückes erzeugt. Diese oftmals unstrukturierte Menge von Meßpunkten wird als *Punktwolke* des Werkstückes bezeichnet. Bei den Werkstücken kann es sich um ein maßstabgetreues Ton -, Kunststoff - oder Holzmodell eines neuen Produktes (z.B. Karosserie eines Automobiles) handeln, das von einem Designer manuell erstellt wurde. In anderen Problemstellungen werden kleinere manuelle Änderungen (Abschleifen) an einem aus einem CAD - Datensatz gefrästen Modell vorgenommen. Das geänderte Modell wird ganz oder teilweise digitalisiert und auf Basis dieser Meßdaten sollen die mechanischen Änderungen am CAD - Datensatz nachgebildet werden (wird auch als Flächennachführung bezeichnet). Eine weitere Anwendung für die *Reverse Engineering* - Verfahren ist die Erstellung individueller Prothesen oder Spezialausrüstungen (Helme) für einzelne Menschen nach deren Digitalisierung.
Die Qualität der Meßpunktmenge (Meßwertrauschen, Meßunsicherheiten, Struktur der Meßpunkte) hängt sehr stark von dem verwendeten Meßverfahren ab und muß bei den Reverse Engineering Verfahren berücksichtigt werden.
Die heute angewendeten Methoden zur Flächenrückführung sind sehr zeitaufwendig und werden von zahlreichen Interaktionen des CAD - Konstrukteurs begleitet. Die meiste Zeit wird für das Konstruieren eines Netzes von Randkurven auf der Punktwolke benötigt. Mit diesem Netz von Randkurven wird die Punktwolke in Bereiche aufgeteilt (Segmentieren der Punktwolke), die von einem einzigen Flächenstück approximiert werden soll. Dort wo der Konstrukteur einen Krümmungswechsel vermutet (an scharfen Kanten oder beim Übergang von ebenen Bereichen in stärker gekrümmte Bereiche) werden Kurven auf der Punktwolke eingefügt. Das interaktive Anwählen der Punkte mit vermutetem Krümmungswechsel für das Erzeugen dieser Kurven ist aber mühsam, da man auf dem zweidimensionalen Bildschirm den Krümmungswechsel nur schwer erahnen kann und dazu oft ein mehrmaliges

Drehen und Verschieben der Punktwolke in dem CAD - System durchgeführt werden muß. Im linken Bild der Abbildung 1.1 ist eine Punktwolke dargestellt aus der manuell ein CAD - Modell bestehend aus 48 Flächenstücken erzeugt wurde (rechtes Bild in der Abbildung 1.1). Eine Automatisierung des Prozesses zur Segmentierung der Punktwolke würde die Flächenrückführung stark beschleunigen. Ein solcher Ansatz zur automatischen krümmungsabhängigen Segmentierung wurde in dieser Arbeit entwickelt.

Eine ausführlichen Einleitung zu den oben angedeuteten Problematiken und den verschiedenen in der Literatur vorgestellten Lösungsstrategien findet man in dem Übserichtsartikel zum Thema *Reverse Engineering* von Varády, Martin und Cox [VaMaC 97]. In dem Tagungsband *Reverse Engineering* von Dankwort und Hoschek [DanHo 96] sind die Vorgehensweisen zur manuellen Flächenrückführung mit mehreren kommerziellen CAD - Systemen für einige Datensatz Beispiele dargestellt. Auf einen Teil dieser Beispiele werden die in dieser Arbeit entwickelten Methoden zur Automatisierung der Flächenrückführung angewendet.

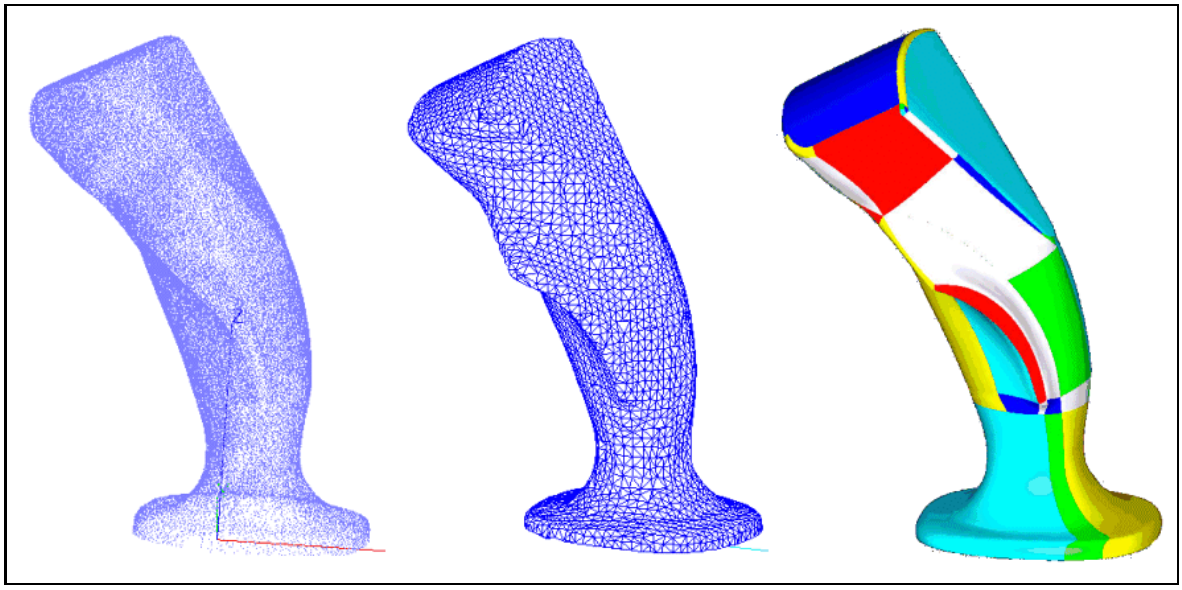


Abbildung 1.1 : Punktwolke, Triangulierung und CAD - Modell eines Sidesticks

Bild links : *Die aufbereitete Meßpunktwolke des Sidesticks zur Steuerung eines F200 Forschungsautos besteht aus 393.492 Punkten und wurde in sieben Ansichten aufgenommen.*

Bild mitte : *Das auf der Punktwolke erzeugte Triangulationsmodell des Sidesticks besteht aus 7.270 Dreiecken.*

Bild rechts : *Das auf der obigen Punktwolke rückgeführte Flächenmodell besteht aus 48 Flächen und wurde mit dem CAD - System Surfacer (Version 6.0) der Firma Imageware von Wolfgang Glocker [GLO 97] in ca. fünf Stunden konstruiert. Die bei der Flächenrückführung aufgetretenen Abweichungen zwischen Punktwolke und CAD - Modell sind in Abbildung 8.3 auf Seite 150 und der darauf folgenden Tabelle sichtbar.*

1.2 Stand der Technik

Ansätze zur automatischen Flächenrückführung finden sich in den Arbeiten von Hoppe, Eck und Bajaj. In diesen Arbeiten werden ausgehend von einer vollständigen Triangulierung der Punktwolke Verfahren beschrieben, wie man aus dieser Triangulierung komplette Flächenbeschreibungen der Punktwolke entweder mit dreieckigen Bézier - Flächen (Bajaj) [BaBeX 95a], [BaBeX 95b] oder mit kubischen Tensorprodukt Bézier - Flächen mit gleichmäßiger Struktur erhält (Eck und Hoppe) [EckHo 96]. Ein Beispiel einer Triangulierung für die Punktwolke aus dem linken Bild in Abb. 1.1 ist in der Mitte dieser Abbildung dargestellt.
Die Zerlegung der Triangulierung in gleichmäßige Teile erfolgt aber nicht nach geometrischen Kriterien, sondern nach Verfahren, die abhängig von der Datenstruktur der Triangulierung sind. Es werden ungefähr gleichgroße und gleichstrukturierte Flächenstücke erzeugt. Diese Zerlegung in Flächenstücke liefert daher eine wesentlich andere Aufteilung der Punktwolke in Flächensegmente als bei den interaktiven Verfahren der Flächenrückführung anfallen würde. Die nach diesen Methoden erzeugten Flächenverbände werden nicht von den Designern und Konstrukteuren akzeptiert, da sie sich zu sehr von den manuell erzeugten Flächenverbänden unterscheiden.
In den Arbeiten von Hugues Hoppe [Hoppe 94] und Hoppe et al. [HoDeR 94] werden ausgehend von einer vollständigen Triangulierung Subdivision Surfaces mit Multiresolution Analysis erzeugt. Diese Flächenmodelle können aber bisher nicht in geschlossene NURBS - Flächenmodelle umgewandelt werden und sind daher für die Weiterverarbeitung mit CAD - Systemen und im weiteren Fertigungsprozeß noch nicht brauchbar.
Methoden zur halbautomatischen Segmentierung von Punktwolken finden sich in den im folgenden aufgeführten Arbeiten. In der Arbeit von von Milroy, Bradley und Vickers [MiBrV 97] wird zunächst durch die Punktwolke eine Schar von drei orthogonalen Schnitten gelegt und damit ein achsparalleles Kurvennetz auf der Punktwolke aufgebaut. Auf diesem werden dann Normalenvektoren für die Punkte abgeschätzt und es wird versucht mit diesen Normalenvektoren und einem sogenannten Snake - Edge Algorithmus die Punktwolke nach einem Krümmungskriterium zu segmentieren. Für den Snake - Edge Algorithmus muß aber durch den CAD - Konstrukteur die zu findende Segmentgrenze als Polygon auf der Kurvenschar grob vorgegeben werden. Dies ist bei komplexeren 3D-Objekten mit Hinterschneidungen aber nicht immer einfach möglich und erfordert aufwendige Interaktionen des CAD - Konstrukteurs.
In der Arbeit von Isselhard, Brunett und Schreiber [IsBrS 98] wird ein ein Verfahren zur Segmentierung einer triangulierten Punktwolke mit einem Region - Growing Ansatz vorgestellt. Der verwendete Region - Growing Algorithmus benutzt die Normalenvektoränderungen von benachbarten Dreiecken der zugrundeliegenden Triangulierung als Zugehörigkeitskriterium für eine Region. Die Grenzen der erzeugten Regionen können daher bisher nur als scharfe Kanten in der triangulierten Punktwolke des Objektes erkannt werden (First Order Feature Lines). Eine Weiterverarbeitung dieser Kanten als B-Spline Kurven findet nicht statt.
In der Dissertation von Ralph Knorpp [Knor 98] wird das Finden von Kanten auf Punktwolken vorgestellt. Dazu werden in den Punkten der Punktwolke verschiedene Merkmale (z.B. Normalenvektoren, mittlere Krümmungen) über die Betrachtung von Nachbarpunkten aus kleinen Kugelumgebungen der Punkte abgeschätzt. Durch Vorgabe von Schwellenwerten werden Startpunkte automatisch erzeugt,

von denen aus mit einem Kantenverfolgungsalgorithmus versucht wird Polygonzüge entlang der Kanten zun bestimmen. Diese Kanten-Polygone sind aber nicht geschlossen, daher kann eine automatische Segmentierung der Punktwolke mit diesen Kanten nicht durchgeführt werden.
In meiner Diplomarbeit [Wil 94] habe ich ein mehrstufiges Verfahren zur Segmentierung von Punktwolken entwickelt, das nur auf Punktwolken anwendbar ist, die in Form von sich nicht schneidenden Konturlinien (Scan - Linien, parallele Abtastbahnen) vorliegen. Zunächst werden auf allen Punkten der Konturlinien Krümmungswerte abgeschätzt. Punkte, an denen Änderungen dieser Krümmungswerte auftreten (sogenannte **Feature Points**), werden als Startpunkte für einen Konturverfolgungsalgorithmus benutzt. Dieser Algorithmus liefert als Ergebnis Polygone aus Punkten der Punktwolke, an denen eine signifikante Änderung der Krümmungswerte beobachtet werden konnte. Diese Polygone aus Punkten der Punktwolke werden als **Feature Lines** bezeichnet.
Da dieses Segmentierungsverfahren nicht auf allgemeine Punktwolken angewendet werden kann, wurde in der vorliegenden Arbeit ein Verfahren entwickelt, das ausgehend von einer unstrukturierten Menge von 3D - Meßpunkten eines beliebigen dreidimensionalen Objektes fast automatisch Berandungskurven für die Flächenrückführung erzeugt. Mit "fast automatisch" ist gemeint, daß keine komplexen Interaktionen des CAD - Kontrukteures mit der Punktwolke nötig sind, sondern daß durch die Eingabe von einigen Schwellenwerten und Parametern die Segmentierungskurven erzeugt werden können. Mit diesem Netz von Segmentierungskurven können in einem nachfolgenden Schritt automatisch Flächenverbände aus Tensorprodukt B-Spline Flächen berechnet werden.

1.3 Überblick der Arbeit

Ziel dieser Arbeit ist es, den zur Zeit noch interaktiven und sehr langwierigen Prozeß der Segmentierung von Punktwolken zu automatisieren. Daher habe ich in dieser Arbeit Algorithmen und Verfahren entwickelt, die unstrukturierten Punktwolken nach Angaben von intuitiven Schwellenwerten automatisch in einzelne Regionen zerlegen. Diese Zerlegung erfolgt nach geometrischen Kriterien, das heißt, daß an allen Punkten einer Region ein ähnliches Krümmungsverhalten auftritt. Die erzeugten Regionen und deren Segmentgrenzen können in CAD - Systemen weiterverarbeitet werden und führen zu besseren Approximationsergebnissen. Mit den in dieser Arbeit entwickelten Algorithmen und Verfahren kann der Prozeß der Flächenrückführung stark beschleunigt werden.
Zu diesem Zweck habe ich zwei neue Triangulationsverfahren für unstrukturierte Punktwolken entwickelt und implementiert, die im 4. Kapitel näher vorgestellt werden. Im 5. Kapitel führe ich eine neue effiziente Art der Krümmungsabschätzung für diese triangulierten Punktwolken ein. Auf diese, mit Krümmungswerten versehene, triangulierte Punktwolke habe ich im 6. Kapitel Segmentierungsalgorithmen angewendet. Diese Segmentierungsalgorithmen stammen aus der 2D-Bildverarbeitung und wurden von mir auf dreidimensionale triangulierten Punktwolken verallgemeinert. Das 7. Kapitel zeigt anhand einiger Beispiele, wie ich die aus dem Segmentierungsverfahren erzeugten Berandungskurven zur Verbesserung und Beschleunigung des *Reverse Engineering* Prozesses, verwenden habe.

Der Prozess der Flächenrückführung lässt sich nach [VaMaC 97] in vier Phasen unterteilen :

- 1.) Datenaquisition (Aufnahme der Punktwolke, 3D - Scannen)
- 2.) Datenaufbereitung (die verschiedenen Messungen in ein gemeinsames Koordinatensystem überführen und Ausreißer entfernen)
- 3.) Segmentieren und einzelne Flächenstücke generieren
- 4.) Erzeugen des gesamten CAD - Modells

Diese Aufteilung in einzelnen Phasen wird auch in dieser Arbeit verwendet.

Im **2.** Kapitel wird ein Überblick der verschiedenen Verfahren zur optischen Messung von dreidimensionalen Objekten gegeben (*1. Phase - Datenaquisition*). Die Begriffe Meßfehler, Meßwertungenauigkeit, Punktwolke, Ansichtsbilder und Tiefenbilder werden dort erläutert. Im letzten Teil des Kapitels befinden sich einige Beispiele für die Realisierung optischer 3D - Meßsysteme. Bei den optischen Messungen entstehen sehr große Mengen von unstrukturierten 3D - Meßpunkten (Scattered Data). Bei der kompletten optischen Messung eines Mercedes SLK Automobiles mit dem Ganymed 97 Meßsystem (Abbildung 2.13) sind ca. 20 Mio. Meßpunkte angefallen. Um Algorithmen mit aktzeptablen Laufzeiten auf so großen unstrukturierten Punktmengen zu entwickeln, müssen diese Punktwolken effizient verwaltet werden.
Im **3.** Kapitel werden verschiedene Möglichkeiten zur Strukturierung großer Punktwolken vorgestellt und miteinander verglichen. Die gewählte Lösung, die sogenannte *Quader - Octree Datenstruktur*, wird anhand eines Beispieles in Abbildung 3.7 genauer dargestellt. Bei dieser Datenstruktur wird die gesamte unstrukturierte Punktwolke in kleine Quader eines gemeinsamen Koordinatensystems (*2. Phase - Datenaufbereitung*), die auch als Voxel (∼ Volumen Elemente) bezeichnet werden, zerlegt. Die einzelnen Quader sind in einer 3D - Baumstruktur (Octree) hierarchisch geordnet und man erhält somit eine Nachbarschaftsbeziehung auf Ebene der Quader.
Für die weitere Verarbeitung der Punktwolke werden im Kapitel **4** Triangulierungsverfahren vorgestellt. Am Anfang des Kapitels erfolgt die Beschreibung der verschiedenen in der einschlägigen Literatur behandelten Möglichkeiten zur Triangulierung von unstrukturierten Punktwolken. Diese Untersuchung habe ich im Rahmen einer Literaturrecherche zum Thema *Triangulierung von 3D - Punktwolken* durchgeführt. Ich habe auch deren Implementierungen, soweit sie frei zugänglich waren, an einigen Beispielen getestet. Jedes der untersuchten Verfahren hatte gewisse Einschränkungen im Hinblick auf die Strukturierung und die Größe der zu verwendenden Punktwolken. Daher stelle ich in den Abschnitten 4.2 und 4.3 zwei neue Verfahren zur Triangulierung von Punktwolken vor, die für unstrukturierte Punktwolken beliebiger Größe eingesetzt werden können. Beide Algorithmen sind parametrisierbar und können aus gegebenen Punktwolken Triangulierungen zu verschiedenen Auflösungen (Dreiecksanzahlen, Genauigkeitsanforderungen) erzeugen.
Diese, nach einem der beiden Verfahren erzeugten Triangulierungen, können als stückweise lineare Approximation einer Punktwolke, die durch Abtasten der Oberfläche eines Objektes gewonnen wurden, aufgefaßt werden. Mit "stückweise linearer Approximation" ist die Annäherung der Objektoberfläche

durch kleine ebene Flächenstücke (Dreiecke) gemeint, bei der nicht alle Meßpunkte als Eckpunkte der Triangulierung verwendet werden müssen.
Diese Triangulierungen können in Anwendungen aus verschiedenen Gebieten direkt weiterverarbeitet werden. Bei den Verfahren zur Erzeugung des Objektes (direktes Fräsen auf einer Triangulierung) oder den sogenannten generativen Fertigungsverfahren aus dem Bereich des **Rapid Prototyping** [Stereolithographie] können zur Zeit auch auf den modernsten Computern nur Triangulationsmodelle mit einer beschränkten Anzahl von ca. 100.000 Dreiecken verarbeitet werden. Bei den Visualisierungsmethoden aus dem Bereich der **Virtual Reality** liegt die Anzahl der maximal in Echtzeit darzustellenden Dreiecke in der Größenordnung von einigen 100.000 Dreiecken. Daher ist eine Verwendung aller bei der Messung des Objektes anfallenden Punkte nicht immer notwendig und sinnvoll. Einzelheiten zu den aufgeführten Methoden der Weiterverarbeitung sind anhand einiger Anwendungsbeispiele im Ausblick (Kapitel **8**) dargestellt.
Im **5.** Kapitel habe ich ein Krümmungsmaß auf triangulierten Punktwolken erarbeitet, das als Basis für eine spätere Segmentierung, Glättung und Datenreduktion der Punkwolke verwendet wird. Für eine effiziente Berechnung der Krümmungswerte bieten die Triangulierungen einen guten Ausgangspunkt, da durch die impliziten Nachbarschaftsbeziehungen auf der Triangulierung ein Finden von Nachbarpunkten und das Bestimmen von Normalenvektoren zu einem Punkt sehr schnell möglich ist. Am Ende des Kapitels werden einige Anwendungsmöglichkeiten der eingeführten Krümmungsabschätzungen aufgezeigt.
Zunächst habe ich mehrere Möglichkeiten zur Krümmungsabschätzung auf triangulierten Punktwolken mit deren differentialgeometrischen Hintergründen untersucht. Die unterschiedlichen Verfahren der Krümmungsabschätzungen wurden auf einige Beispiele angewendet und die Resultate miteinander verglichen. Das in Abschnitt 5.3 vorgestellte Krümmungsmaß der *mittleren Normalenvektoränderung* hat sich für die untersuchten Beispiele als besonders effizient und für weiterführende Analysen am geeignetsten erwiesen.
Im **6.** Kapitel werden die Grundlagen und Prinzipien der automatischen Segmentierung von Punktwolken sowie Bildern aus der digitalen Bildverarbeitung kurz erläutert (*3. Phase - Segmentierung*). Die Verfahren aus der Bildverarbeitung wurden von mir auf dreidimensionale triangulierte Punktwolken erweitert. Der in der Bildverarbeitung betrachtete Intensitäts- oder Helligkeitswert in einem Bildpunkt wird im Falle der triangulierten Punktwolken durch das im vorherigen Kapitel eingeführte Krümmungsmaß der mittleren Normalenvektoränderung ersetzt. Die aus der Bildverarbeitung bekannten Segmentierungsprinzipien der kantenorientierten bzw. regionenbasierten Verfahren habe ich auf die Krümmungswerte der triangulierten Punktwolken angewendet. Die bei der Segmentierung entstehen Polygonzüge können nach der Glättung, die auf der Approximation des sogenannten *Feature Line Polygons* mit B-Spline Kurven beruht, als **Feature Line Kurven** in dem Prozess der Flächenrückführung weiterverwendet werden. Am Ende des Kapitels befinden sich einige Beispiele für die Segmentierung von Punktwolken, die nach den vorgestellten Algorithmen automatisch bestimmt wurden.
Im **7.** Kapitel wird exemplarisch gezeigt, wie man die Feature Line Kurven sinnvoll zur Beschleunigung und Verbesserung des Prozesses der Flächenrückführung einsetzen kann. Die Feature Lines können dabei entweder als Trimmkurven verwendet werden, oder es kann mit ihnen und den ebenfalls als Ergebnis des Region - Growing Algorithmus aus dem letzten Abschnitt vorhandenen inneren Punkten

jeder Region automatisch eine Tensor-Produkt B-Spline Fläche bestimmt werden. Diese werden in einem nachfolgenden Verarbeitungsschritt mit den sogenannten Blendflächen zu einem Flächenverband, der das CAD - Modell der gesamten Punktwolke repräsentiert, zusammengefügt (*4. Phase - Erzeugen des CAD - Modells*).
Für die erzeugten Flächenmodelle wurden eine Fehlerbetrachtung und ein Vergleich mit dem Approximationsverfahren von Ulrich Dietz [Dietz 98] durchgeführt, bei dem durch die gesamte Punktwolke eine Tensor-Produkt B-Spline Fläche mit sehr vielen Segmenten erzeugt wird.
Im letzten Kapitel erfolgt eine abschließende Zusammenfassung der Vorgehensweise zur automatischen Segmentierung von unstrukturierten Punktwolken und der anschließenden Flächenerzeugung. Ferner werden noch andere Anwendungsaspekte der in dieser Arbeit vorgestellten Algorithmen gezeigt.
Die in der Arbeit angegebenen Rechenzeiten für die Algorithmen zur Triangulierung, Krümmungsabschätzung und Segmentierung der Punktwolken, wurden überwiegend auf einem handelsüblichen Personal Computer, der mit einem Intel Pentium II 266 MHz Prozessor und 128 Megabyte Arbeitsspeicher ausgestattet ist, durchgeführt. Das auf dem verwendeten PC installierte Betriebssystem ist Windows NT 4.0 der Firma Microsoft Corp. (USA). Die Rechenzeiten zu einigen Beispiele wurden auch auf Workstations der Firma Silicon Graphics Inc. (USA) mit dem IRIX (Unix Derivat) Betriebssystem Version 5.4 ermittelt (Modell O^2 mit MIPS R10000 175 MHz CPU und 128 MB RAM bzw. Modell Indigo 2 mit MIPS R4400 150 MHz CPU und 256 MB RAM). Dieses ist dann als Bemerkung zum jeweiligem Beispiel angegeben. Der Vergleich der Rechenzeiten zwischen diesen unterschiedlichen Rechner Architekturen und Betriebssystemen an einigen Beispielen zeigte, daß die Personal Computer bei Aufgaben, in denen es im wesentlichen um numerische Berechnungen ging (Floating Point Arithmethik), um ca. den Faktor zwei schneller waren als die Workstations. Bei Aufgaben, in denen zusätzlich noch ein dynamisches Speichermanagement benötigt wurde (z.B. Umbau der Quader Octree Datenstruktur), waren hingegen die Workstations etwas schneller (ca. 20 %) als die PCs.
Im Anhang befinden sich neben dem Literaturverzeichnis eine kurze Projektübersicht zum Innovationsprojekt *Reverse Engineering* am Daimler - Benz Forschungszentrum Ulm, in dessen Umfeld diese Arbeit entstanden ist. Darüber hinaus werden die in der Arbeit verwendeten mathematischen Grundlagen aus dem Bereich des **C**omptuer **A**ided **G**eometric **D**esign (CAGD) und der Wavelet Theorie aufgeführt.

Ein Teil der in dieser Arbeit vorgestellten Ergebnisse wurde zusammen mit Prof. Dr. Josef Hoschek und Dr. Ulrich Dietz in dem Artikel [HoDiW 98] veröffentlicht.

Danksagung

Der größte Teil der vorliegende Arbeit enstand während meiner Tätigkeit als Doktorand in der Abteilung Optische Meßtechnik und Modelltechnik am Daimler-Chrysler Forschungszentrum in Ulm.
Mein ganz besonderer Dank geht an dieser Stelle an Herr. Prof. Dr. J. Hoschek für die gute Betreuung und die freundliche Unterstützung meines Promotionsvorhabens sowie für die konstruktiven Diskussionen, die zum hoffentlichen Gelingen dieser Arbeit beigetragen haben. Herrn Prof. Dr. C. W. Dankwort möchte ich für die Übernahme des Koreferats danken, Herrn Prof. Dr. U. Reif für die Gehnemigung zur Veröffentlichung. Herrn Dr. U. Dietz danke ich für die Überlassung seiner Programme zur Erzeugung glatter Flächen aus unstrukturierten Punktwolken. Desweiteren möchte ich mich bei Frau S. Drexler für die freundliche Aufnahme und der organisatorische Unterstützung bei meinen Besuchen in Darmstadt bedanken. Mein weitere Dank gilt meinen Vorgesetzen Dr. G. Jünemann, Dr. F. May und Dr. K. Grebner am Daimler-Chrysler Forschungszentrum, für die Gewährung von Freiräumen, die zum Entstehen einer Promotion im Umfeld einer industriellen Forschungseinrichtung notwendig sind.
Den Mitgliedern der Arbeitsgruppe Differentialgeometrie und Kinematik des Fachbereichs Mathematik der TU Darmstadt sowie bei meinen Kollegen in den Abteilungen Optische Meßtechnik und Modelltechnik und des Virtual Reality Competence Centers am Daimler - Chrysler Forschungszentrum in Ulm danke ich für die angenehme und freundliche Zusammenarbeit.
Bei meiner Frau Andrea bedanke ich mich für das Korrekturlesen und die Unterstützungen und Ermunterungen in den letzten Jahren.

2 Optische Messung von 3D - Objekten

Überblick zur Optischen Meßtechnik

Industrielle 3D - Meßsysteme müssen hohe Anforderungen erfüllen, wenn sie erfolgreich im Fabrikumfeld eingesetzt werden sollen. Neben der Robustheit spielen vor allem auch die Meßgeschwindigkeit, die Kosten und die Benutzerfreundlichkeit eine wichtige Rolle. Streifenprojektionsverfahren bieten in dieser Hinsicht viele Vorteile.
Die optische 3D - Meßtechnik hat das Ziel, die x, y, z-Koordinaten der Oberfläche des Meßobjekts zu bestimmen. Der Vorteil gegenüber den traditionellen taktilen Verfahren mit Koordinaten Meßmaschinen liegt insbesondere darin, daß mit der optischen 3D - Meßtechnik mehrere hunderttausend Punkte gleichzeitig erfaßt werden können und die Meßzeiten dadurch extrem verkürzt werden. Weitere Vorteile folgen aus der Tatsache, daß optische Messungen berührungslos erfolgen und somit auch weiche Materialien gemessen werden können. Die optische Meßtechnik stößt an Grenzen, wenn beispielsweise hochglänzende Oberflächen oder tiefe Bohrungen am Meßobjekt vorhanden sind. Hier werden auch in Zukunft mechanische Verfahren von Vorteil sein.
Obwohl mit einem optischen 3D - Sensor sehr viele Punkte gleichzeitig gemessen werden können, kann pro Messung und Sensor immer nur eine Ansicht betrachtet werden. Um die gesamte Oberfläche eines Meßobjekts aufzunehmen, müssen die unterschiedlichen Ansichten richtig zusammengesetzt werden. Dies ist eine wichtige Anforderung an ein fabriktaugliches optisches 3D - Koordinatenmeßsystem, der mit Hilfe photogrammetrischer Verfahren zur Sensornavigation entsprochen werden kann.

2.1 Einführung in die optische 3D - Meßtechnik

2.1.1 Das Streifenprojektionsverfahren

Mit dem Streifenprojektionsverfahren können große, dichte Punktwolken erfaßt werden (vgl. Streifemmuster auf dem Kotflügel aus Abbildung 2.1). Die entsprechenden 3D - Sensoren bestehen aus einem Streifenprojektor und einer Matrixkamera, die das Meßobjekt aus unterschiedlichen Winkeln erfassen und auf Basis der Triangulation die Tiefeninformation ermitteln. Unter bestimmten Voraussetzungen ist es möglich, die vollständige 3D - Information zu einem einzigen Zeitpunkt zu erfassen. Dies ist insbesondere für zeitlich instabile oder sich bewegende Objekte wichtig.
Häufig kann aber der Gesamtaufbau von Sensorsystem und Meßobjekt über einen gewissen Zeitraum konstant gehalten werden. Dann wird man in der Regel die Messung durchführen, indem eine zeitliche Sequenz an Streifenmustern aufprojiziert und von der Matrixkamera erfaßt wird (temporale Streifencodierung). Daraus werden dann die gewünschten 3D - Informationen errechnet.
Es ist möglich unterschiedliche Streifenmuster zu verwenden. Beispielsweise können diese Streifenmuster einen Polarisations- oder Farbcode beschreiben. Für Meßobjekte mit unstetigen Oberflächen hat sich aber insbesondere der Gray-Code bewährt. Häufig findet auch das Phasenshift-Verfahren Anwendung, bei dem pro Messung mindestens drei Aufnahmen gemacht werden und das Streifengitter jeweils um den Bruchteil einer Gitterperiode verschoben wird. Durch das Phasenshift-Verfahren

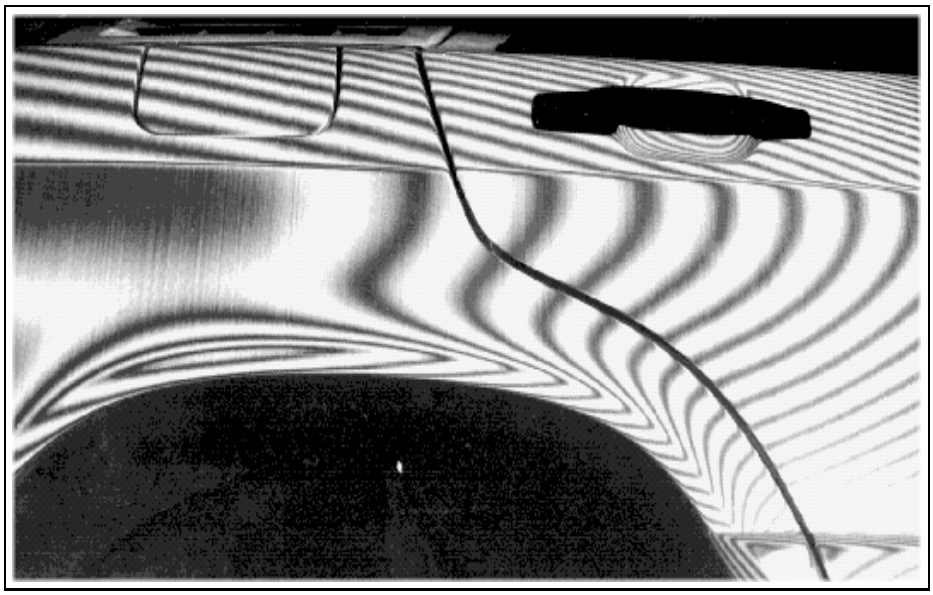


Abbildung 2.1 : Beispiel der Streifenprojektion auf einem Kotflügel

kann die Empfindlichkeit der Tiefenmessung erhöht werden. Der Gray-Code zeigt sich andererseits unempfindlich gegenüber Remissionsschwankungen. Diese Vorteile lassen sich vereinigen, indem man Messungen mit einer Kombination aus Gray-Code- und Phasenshift-Verfahren durchführt. Die nach dem optischen Triangulationsverfahren theoretisch, ermittelbaren Werte eines idealen Sensorsystems können von einem realen Streifenprojektionssystem im allgemeinen nicht geliefert werden, da hier fertigungstechnische Fehler, Fehljustagen und Verzeichnungen der Objektive mit eingehen. Aus diesem Grunde müssen die Abweichungen durch Kalibration mit bekannten Oberflächen an bekannten Positionen festgestellt und berücksichtigt werden.

2.1.2 Fabriktauglichkeit optischer Meßsysteme

In Zunkunft werden immer mehr 3D - Meßverfahren eingesetzt werden, auch wenn aus heutiger Sicht noch sehr viele technische Probleme ungelöst sind. Allerdings muß häufig die gesamte Oberfläche eines Objekts absolut vermessen werden. Dazu müssen optische 3D - Sensoren Messungen aus unterschiedlichen Ansichten durchführen. Um die unterschiedlichen Ansichten zusammensetzen zu können, müssen beispielsweise die jeweiligen Sensorpositionen genau bekannt sein. Dies kann erreicht werden, indem man den optischen 3D - Sensorkopf mit einem mechanischen Positioniersystem verbindet. Liegt die tolerierte Meßungenauigkeit bei $\leq$ 0,1 mm, muß das mechanische Positioniersystem über eine entsprechend höhere Genauigkeit verfügen. Solche Positioniersysteme sind für große Objekte aber teuer, so daß derartige Lösungen für Neuinstallationen im allgemeinen nicht in Frage kommen. Demgegenüber sind in den Fabriken aber häufig Positioniersysteme in Form von Koordinatenmeßmaschinen oder Fräsmaschinen vorhanden. Diese Maschinen können zur Positionierung optischer 3D - Sensoren verwendet werden, indem anstatt des Tastkopfs bzw. des Fräsers der optische Sensorkopf angebracht wird. Allerdings besteht auch hier noch eine Reihe offener Fragen. So sind beispielsweise

die optischen 3D - Sensorköpfe häufig zu schwer, um an Koordinatenmeßmaschinen angebracht zu werden. Das Prinzip der Steuerung einer 3D Meßung mit optischen Sensor ist in der Abbildung 2.2 dargestellt. Im Abschnitt 2.3 folgen noch einige Bilder von Realisierungen optischer Meßsysteme.

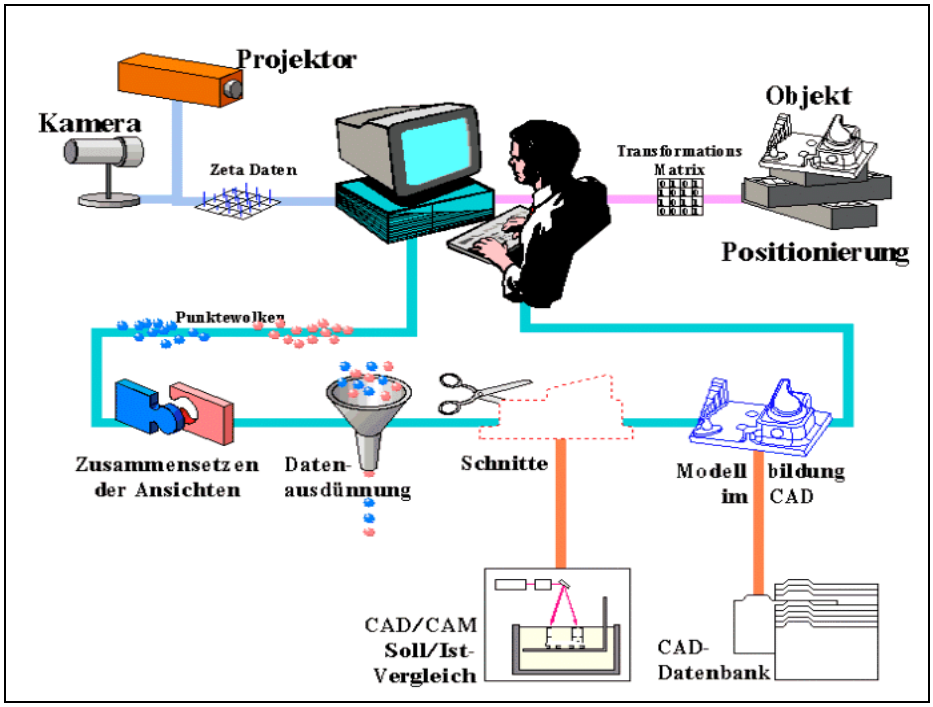


Abbildung 2.2 : Skizze zur Prozeßsteuerung optischer Messungen

2.1.3 Photogrammetrie

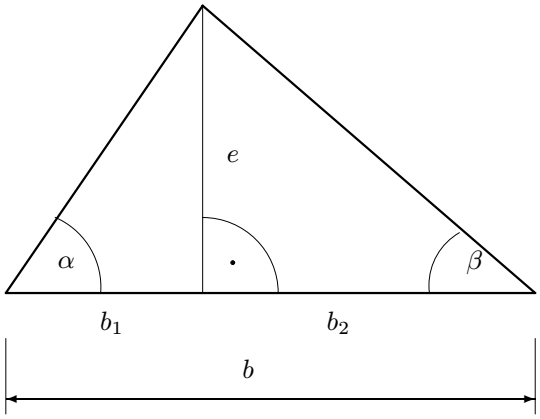
Durch die Kombination der Streifenprojektion mit der Photogrammetrie lassen sich optische 3D - Sensoren bauen, die vollkommen auf die teure mechanische Positionierung verzichten können. Die Idee dabei ist, daß mit der Matrixkamera des Sensorkopfes außer dem eigentlichen Meßobjekt auch Paßpunkte aufgenommen werden. Die Orientierung von Kameras relativ zu Paßmarken ist aber eine elementare Aufgabe der Photogrammetrie. Anhand solcher Paßpunkte läßt sich daher die Position des Sensors bestimmen; es ist eine autonome Orientierung der optischen 3D - Sensoren möglich.
Streifenprojektionssysteme, die photogrammetrisch navigiert werden können, bieten Vorteile für den Einsatz in der Fabrik. So kann das Meßobjekt vor Ort aus allen notwendigen Ansichten gemessen werden und muß nicht in einen separaten Meßraum gebracht werden; für die Positionierung des Meßsystems reicht ein einfaches Stativ. Außer zur Sensornavigation kann die Photogrammetrie auch zur Kalibrierung der Matrixkamera eingesetzt werden. Dabei werden mehrere Aufnahmen einer mit zahlreichen Signalmarken beklebten Platte gemacht. Die Kalibrierung des Projektors erfolgt analog, wobei der Projektor als inverse Kamera betrachtet wird. Innerhalb von wenigen Minuten kann so vor Ort die Kalibrierung des 3D - Sensors unmittelbar vor oder nach der Messung erfolgen.

2.2 Gewinnung und Analyse von Tiefenbildern

Im Gegensatz zu den traditionell verwendeten Grauwertbildern, bei denen ein jeder Bildpunkt die von einer Kamera aufgenommene Lichtintensität wiedergibt, repräsentieren die in einem Tiefenbild enthaltenen Daten den Abstand einzelner Punkte auf der Oberfläche der abgebildeten Objekte vom Sensor. Gegenüber Grauwertbildern liegt der größte Vorteil von Tiefenbildern in der expliziten Darstellung der Gestalt dreidimensionaler Objekte. In direktem Zusammenhang damit steht die Unempfindlichkeit von Tiefenbildern gegenüber Faktoren wie Beleuchtung, Schattenwurf und Beschädigung oder Verschmutzung von Objektoberflächen, welche die Analyse von Grauwertbildern erheblich erschweren.
Aufgrund der dreidimensionalen Natur von Tiefenbildern können viele Aufgaben der Bildanalyse wesentlich vereinfacht werden. Von noch größerer Bedeutung ist aber die Tatsache, daß die Tiefenbildanalyse Perspektiven zu neuen Anwendungen eröffnet, die bisher anhand von Grauwertbildern kaum möglich waren. Dazu zählen beispielsweise das Sortieren von unbekannten Objekten, die Navigation autonomer Fahrzeuge in einer natürlichen Umgebung und die Formprüfung zur automatischen Qualitätskontrolle. Mit neuen Entwicklungen im Gebiet der Tiefenbildanalyse wird auch das Anwendungsspektrum ständig erweitert. Heute gilt die Gewinnung, Verarbeitung und Interpretation von Tiefenbildern als eines der zentralen Forschungsthemen der Bildanalyse.
Der enorme Aufschwung des Gebietes Tiefenbildanalyse ist eng verbunden mit der rasanten Entwicklung der Sensortechnik zur Gewinnung von dreidimensionalen Daten. Für die Wiedergewinnung der bei der Projektion in die Bildebene verlorengegangenen Tiefeninformation wurde eine große Anzahl passiver und aktiver Verfahren entwickelt. Eines der passiven Verfahren ist die Verwendung von zwei (oder mehr) Stereobildern. Damit läßt sich die Tiefe nach dem optischen Triangulationsprinzip sehr einfach bestimmen, vorausgesetzt, daß korrespondierende Punkte gefunden werden können. Werden Annahmen über die Lichtquellen und die Reflexionseigenschaften der Objekte getroffen, so lassen sich auch aus der Schattierung der Objektoberflächen Rückschlüsse auf die Objektgestalt ziehen. Neben den passiven Methoden haben sogenannte aktive Verfahren großes Interesse gefunden. Bei dieser Klasse von Ansätzen wird eine der Stereokameras durch eine aktive Energiequelle ersetzt und dadurch das schwierige Korrespondenzproblem umgangen.
Obwohl aktive Tiefengewinnung keine Analogie zu biologischen visuellen Wahrnehmungssystemen besitzt, haben sich entsprechende Verfahren in der Praxis bestens bewährt. Mittlerweile sind bereits relativ günstige PC-basierte aktive Tiefensensoren auf dem Markt erhältlich [BunJi 97].

2.3 Funktionsprinzip optischer Sensoren

Im folgenden Abschnitt soll das Prinzip des optischen Sensors zur 3D Koordinatenmessung beschrieben werden. Dabei wird zuerst die prinzipielle Funktionsweise des Meßverfahrens erklärt.
Die geometrische Grundlage für dieses Meßverfahren bildet das **optische Triangulationsprinzip**, welches in der Skizze 2.3 verdeutlicht wird.
Sind die Winkel α, β und die Strecke b bekannt, so läßt sich der Abstand e nach der rechten unteren Formel in der obigen Abbildung bestimmen. Will man ein Objekt nach diesem Prinzip vermessen, so



$$b_1 = \frac{e}{\tan\alpha}$$

$$b_2 = \frac{e}{\tan\beta}$$

$$b = b_1 + b_2 = \frac{e}{\tan\alpha} + \frac{e}{\tan\beta}$$

$$\Longrightarrow$$

$$e = b \cdot \frac{\tan\alpha \cdot \tan\beta}{\tan\alpha + \tan\beta}$$

Abbildung 2.3 : Skizze und Formel des optischen Triangulationsprinzips

muß man eine Möglichkeit finden, für jeden Objektpunkt die beiden Winkel α und β zu bestimmen. Dies kann beispielsweise geschehen, indem man einen Lichtpunkt bzw. eine Lichtlinie auf das Objekt projiziert und diese dann mit einer Matrixkamera aufnimmt. Dieses Verfahren wird **Lichtschnitt** genannt.

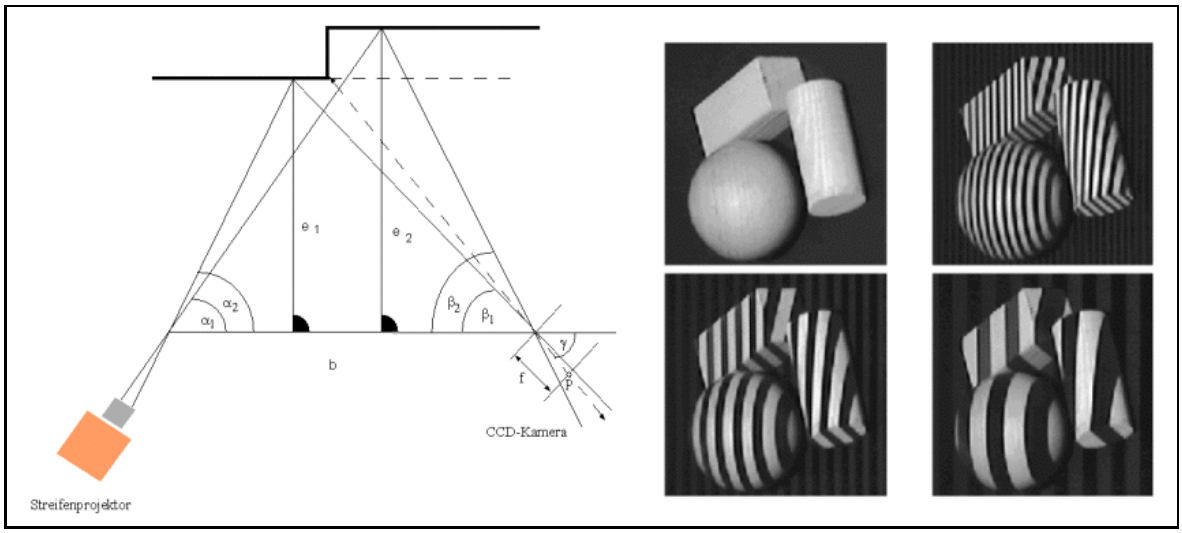


Abbildung 2.4 : Projektions Prinzip und 4 Beispiele der Streifen Projektion

Wie aus dem linken Bild der Abbildung 2.4 leicht zu sehen ist, ergibt jeder unter einem bestimmten Winkel α auf das Objekt projizierte Lichtstrahl einen Punkt an einer bestimmten Stelle der Kameramatrix. Aus der Position p dieses Punktes im Bild läßt sich der Winkel β und damit der Abstand e bestimmen. Da der Abstand f der Kameramatrix zum Scheitelpunkt von b, sowie der Winkel γ der

Kamera zur Basis b bekannt sind, kann man β auf folgende Art berechnen :

$$\beta = 180° - \gamma - \arctan(\frac{p}{f}) \tag{2.2}$$

In der Praxis kann zum Beispiel ein Laser dazu verwendet werden, eine Lichtlinie auf das Objekt zu projizieren. Dieses läßt man dann über das Objekt wandern und nimmt für jede Position ein Bild mit der Kamera auf. Ab einer gewissen geforderten Meßgenauigkeit hat dieses Verfahren jedoch einige Nachteile. Mit steigenden Genauigkeitsanforderungen an die Meßdatenauflösung und die Meßgenauigkeit steigt auch die Anzahl der aufzunehmenden Streifenposition. Will man zum Beispiel 1.000 Positionen erfassen, so müssen 1.000 Einzelbilder aufgenommen werden. Dies ist jedoch aus Zeitgründen unpraktikabel.

Ein Weg zur Lösung dieses Problems ist die gleichzeitige Projektion und Aufnahme aller Linien. Hieraus ergibt sich jedoch eine neue Schwierigkeit : Bei jeder Auswertung kann man nicht mehr ohne weiteres feststellen, welchem Streifen im Bild welche in der Lichtquelle zuzuordnen ist. Somit kann man zwar Winkel β, jedoch nicht mehr den Winkel α eindeutig bestimmen [RIC 95].

Eine Abhilfe wird durch den sogenannten **Codierten Lichtansatz** erreicht. Dabei werden nacheinander verschiedene Streifenmuster auf das Meßobjekt projiziert sowie jeweils ein Bild aufgenommen und binärisiert. Im rechten Bild aus Abb. 2.4 ist ein Beispiel von drei verschiedenen Streifenmustern, die auf die Meßobjekte aufprojiziert wurden, dargestellt. Diese Streifenmuster bilden zusammen einen Code, mit dessen Hilfe man für jeden Objektpunkt einen bestimmten Wert erhält. Dazu werden alle korrespondierenden Bildpunkte der aufgenommenen Bilder zu einem Grauwertvektor zusammengefaßt. Durch Binärisierung der Komponenten dieses Vektors und Dekodierung bekommt man den sog. Zeta-Wert (ζ), welcher die Postion im Projektor angibt und somit ein Maß für den Winkel α ist.

Mit der Projektion von Lichtebenen läßt sich die Anzahl benötigter Projektionen zwar gegenüber der punktweisen Vermessung drastisch reduzieren, aber um n Lichtebenen zu erzeugen, sind aber immer noch n Projektionen nötig. Mit der Technik der codierten Lichtprojektion kann die Anzahl Projektionen weiter reduziert werden. Hierbei reichen bereits n Projektionen zur Erzeugung von 2^n Lichtebenen aus.

Trifft eine Lichtebene auf das Meßobjekt, so resultiert daraus eine Schnittlinie. Sei P ein Punkt auf dieser Schnittlinie und $\hat{P}$ dessen Abbildung in der Bildebene der Kamera. Wird die Lichtebene projiziert, so erhält das Pixel nach der Binärisierung den Wert 1. Andernfalls nimmt $\hat{P}$ den Wert 0 an. Wird die Projektion der Lichtebene in einer zeitlichen Folge ein- oder ausgeschaltet, resultiert daraus bei $\hat{P}$ ein binäres Codewort $B = [b^1 b^2 \cdots b^n]$, dessen einzelne Bits b^i genau dem Status der jeweiligen Projektion entsprechen. Somit kann von einem Projektions- und Empfangscodewort gesprochen werden. Werden die Lichtebenen mit unterschiedlichen Codewörtern aus einem n-stelligen Code codiert, kann die einem Pixel entsprechende Lichtebene anhand des Empfangscodewortes eindeutig identifiziert werden. Auf diese Weise erlaubt ein n-stelliger Code die Projektion von bis zu 2^n Lichtebenen.

In der Praxis werden alle 2^n Lichtebenen gleichzeitig n-mal projiziert. Bei der k-ten Projektion werden die Lichtebenen gemäß dem k-ten Bit ihres jeweiligen Codewortes ein- oder ausgeschaltet. Dies läßt sich u.a. mit einem konventionellen Lichtprojektor realisieren, wobei die Projektionsmuster von einem programmierbaren LCD-Shutter erzeugt werden. Eine schematische Darstellung dieser Projektionsart befindet sich in der Abbildung 2.5 .

Beispiel : Das naheliegende Codierungsschema für die Projektion wäre der n-stellige Binärcode B_n.

Für B_3 ergeben sich folgende drei Projektionsmuster :

Lichtebene	0	1	2	3	4	5	6	7
Projektionsmuster **1**	0	0	0	0	1	1	1	1
Projektionsmuster **2**	0	0	1	1	0	0	1	1
Projektionsmuster **3**	0	1	0	1	0	1	0	1

Nach jeder Projektion wird ein Bild aufgenommen und binärisiert. Das Empfangscodewort in einem Pixel (u, v) setzt sich aus den Bits der einzelnen Projektionen zusammen. Die bekannte Lage der durch dieses Codewort identifizierten Lichtebene und die Bildposition (u, v) erlauben dann die Positionsbestimmung des Punktes auf dem Meßobjekt [BunJi 97].
Die **Binärisierung** entscheidet, ob die einem Pixel entsprechende Lichtebene bei der Projektion eingeschaltet ist oder nicht. Hier zeigt sich jedoch, daß es kaum möglich ist, einen einheitlichen Schwellwert für das gesamte Bild anzusetzen. Die Objektoberflächen weisen oft recht unterschiedliche Reflexionseigenschaften auf. Es ist auch sinnvoll, die Binärisierung möglichst robust bezüglich der Umgebungsbeleuchtung zu gestalten. Ferner muß noch das Problem des Schattenwurfs berücksichtigt werden. Teile der Oberfläche werden zwar von der Kamera gesehen, aber wegen Fremd- oder Eigenverdeckung nicht beleuchtet. Unabhängig davon, ob die Lichtebene eingeschaltet ist oder nicht, bleibt die Intensität an dieser Stelle im Bild ungefähr konstant.
Ohne großen Aufwand läßt sich eine robuste Binärisierung folgendermaßen realisieren: Vor der Projektion der codierten Lichtmuster wird zuerst ein Referenzbild erstellt, indem ein Bild bei voller Beleuchtung der Meßszene und eines bei Ausschaltung sämtlicher Lichtebenen aufgenommen werden. Das Referenzbild ergibt sich aus der Mittelwertbildung der beiden Aufnahmen. Falls sich die Grauwerte in den beiden Bildern nur unwesentlich voneinander unterscheiden, wird der Punkt im Raum als im Schatten befindlich betrachtet. Für die restlichen Pixel dient der Grauwert $s_{(u,v)}$ im Referenzbild als positionsabhängiger Schwellwert für die Binärisierung an der Bildposition (u, v). Nach der Projektion eines Lichtmusters erfolgt die Binärisierung in einem Pixel $b^k_{(u,v)}$ mit Grauwert $g_{(u,v)}$ dann durch :

$$b^k_{(u,v)} = \begin{cases} 1 & g_{(u,v)} \geq s_{(u,v)} \\ 0 & g_{(u,v)} < s_{(u,v)} \end{cases}$$

Bei einer Reihe von Tiefensensoren wurde diese einfache Technik eingesetzt. Es sind aber auch andere robuste Binärisierungsmethoden bekannt.
Statt eines Binärcodes wird in der Regel jedoch der **Gray-Code** verwendet, da dieser ein sogenannter einschrittiger Code ist. Das bedeutet, daß sich zwei aufeinanderfolgende Codeworte nur an einer Stelle und nur um eine Stufe unterscheiden. Solche einschrittigen Codes werden verwendet, um die Fehler, die zum Beispiel durch eine lokal fehlerhafte Binärisierung entstehen, möglichst gering zu halten. Gründe für derartige Fehler sind zum Beispiel elektronisches Rauschen der Kamera oder optische Störungen wie Reflexionen oder Umgebungslicht.

Nicht nur die korrekte Bestimmung des Empfangscodewortes durch die Binärisierung ist entscheidend für die Tiefenberechnung. Die Wahl des Projektionscodes spielt dabei auch eine wichtige Rolle. Im Binärcode B_3 hat die Lichtebene 3 beispielsweise das Codewort *011* . Falls sich das Projektionsmuster 1 *(00001111)* wegen einer Projektionsungenauigkeit um ein Bit nach links verschiebt, so resultiert

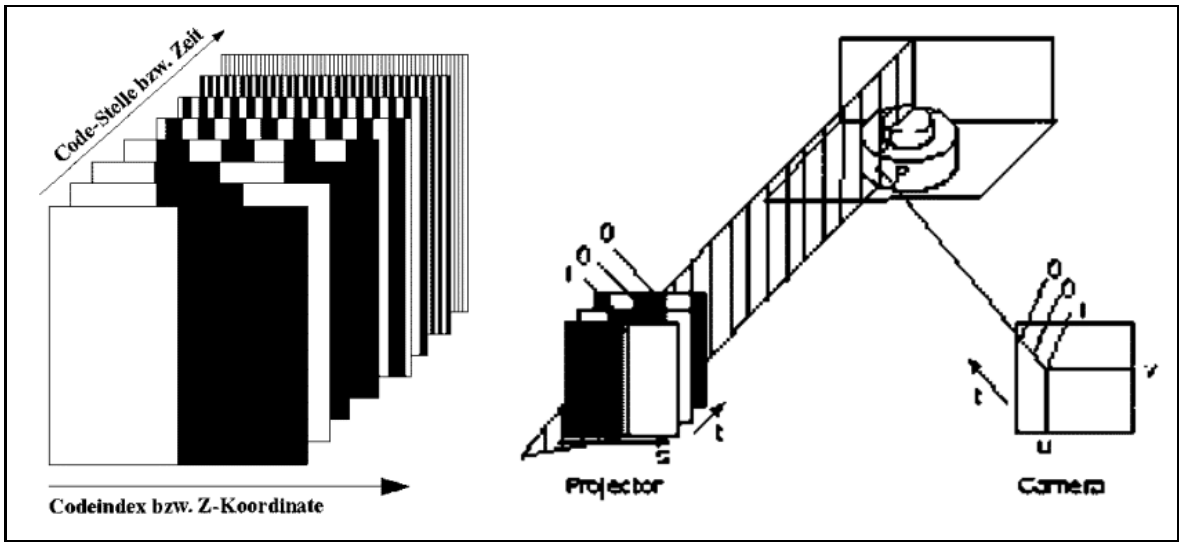


Abbildung 2.5 : Gray-Code Sequenz und Prinzip der Projektion

daraus für genau dieselbe Lichtebene das Codewort *111* , das die Lichtebene 7 impliziert. Somit ist die Lichtebenennummer um den Wert 4 nicht richtig erkannt.
Fehler dieser Art lassen sich leicht mit dem Gray-Code minimieren, der die nützliche Eigenschaft hat, daß zwei benachbarte Codewörter nur in einem Bit verschieden sind. Dadurch verursacht bei Verwendung des Gray-Codes die Verschiebung eines einzigen Projektionsmusters um ein Bit höchstens einen Fehler von l in der Lichtebenennummer. Dies erklärt die Popularität des Gray-Codes im codierten Lichtansatz. In nahezu allen derartigen Tiefensensoren wird der Gray-Code anstatt des Binärcodes verwendet.
Der n-stellige Gray-Code G_n wird rekursiv definiert. Der Code G_1 enthält die zwei Wörter *0, 1.* Der Code G_k ergibt sich aus G_{k-1} mit einer vorgestellten *0* und aus G_{k-1} in umgekehrter Reihenfolge mit einer vorgestellten *1.*

G_1 :	0
	1

$\Longrightarrow$

G_2 :	0	0
	0	1
	1	1
	1	0

$\Longrightarrow$

G_3 :	0	0	0
	0	0	1
	0	1	1
	0	1	0
	1	1	0
	1	1	1
	1	0	1
	1	0	0

Abbildung 2.6 : Beispiel der Erzeugung desr 3-stellige Gray-Codes G_3

Zwischen dem k-ten Codewort $[b^1 b^2 \cdots b^n]$ des Binärcodes B_n und dem k-ten Codewort $[g^1 g^2 \cdots g^n]$ des Gray-Codes G_n besteht die einfache Beziehung :

$$\begin{aligned} b^i &= g^1 + g^2 + \cdots + g^n \\ g^1 = b^1, \qquad g^i &= b^{i-1} + b^i, \qquad i = 2, \cdots, n \end{aligned} \tag{2.3}$$

Diese Beziehung kann dazu verwendet werden, um aus dem Empfangscodewort des Gray-Codes das entsprechende Codewort des Binärcodes zu bestimmen, das genau die Lichtebenennummer wiedergibt. Da die angestrebte Genauigkeit des Sensors im Bereich von 50 Mikrometern liegt, ist die Codierung durch den Gray-Code allein nicht ausreichend. Deshalb wird diese Codierung mit einem aus der Interferometrie bekanntem Verfahren kombiniert, der sogenannten **Phasenshift** - Codierung.

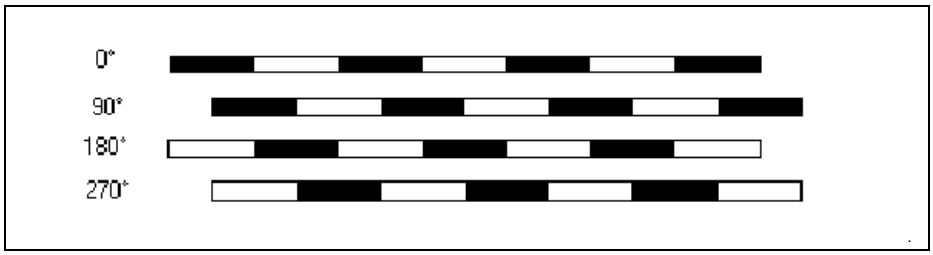


Abbildung 2.7 : Prinzip der Phasenshift Codierung hier : 4 Schritte mit jeweils 90 Grad

Dabei wird von mehreren feinen Streifenbildern ausgegangen, deren Intensitätsverlauf als cosinusförmig angenommen wird. Zueinander besitzten sie eine bestimmte Phasenverschiebung, die in Abbildung 2.7 angedeutet wird. Der Intensitätsverlauf entlang der Streifenstruktur läßt sich wie folgt beschreiben :

$$I(x) = I_0 \cdot (1 + k \cdot \cos(dq(x))) \tag{2.4}$$

I_0 bezeichnet die Hintergrundintensität, k den Kontrast und $dq(x)$ die Phasenlage innerhalb einer Periode. Geht man nun von einem sog. 4er-Phasenshift aus, bei dem die Phasenverschiebung jeweils 90° beträgt, so erhält man folgende vier Gleichungen :

$$\begin{aligned} I_1(x) = \quad I_0 \cdot (1 + k \cdot \cos(dq(x) + 0°)) \quad &= \quad I_0 \cdot (1 + k \cdot \cos(dq(x))) \\ I_2(x) = \quad I_0 \cdot (1 + k \cdot \cos(dq(x) + 90°)) \quad &= \quad I_0 \cdot (1 + k \cdot (-\sin(dq(x)))) \\ I_3(x) = \quad I_0 \cdot (1 + k \cdot \cos(dq(x) + 180°)) \quad &= \quad I_0 \cdot (1 + k \cdot (-\cos(dq(x)))) \\ I_4(x) = \quad I_0 \cdot (1 + k \cdot \cos(dq(x) + 270°)) \quad &= \quad I_0 \cdot (1 + k \cdot \sin(dq(x))) \end{aligned} \tag{2.5}$$

Subtrahiert man nun $I_2(x)$ von $I_4(x)$ und $I_1(x)$ von $I_3(x)$, so ergibt sich durch Division dann :

$$\frac{I_2(x) - I_4(x)}{I_2(x) - I_4(x)} \quad = \quad \frac{-2\ I_0\ k \cos(dq(x))}{-2\ I_0\ k \sin(dq(x))} \quad = \quad \tan(dq(x)) \tag{2.6}$$

Für die gesuchte Phasenlage erhält man schließlich :

$$dq(x) \ = \ \arctan \frac{I_2(x) \ - \ I_4(x)}{I_1(x) \ - \ I_3(x)} \tag{2.7}$$

In der Praxis steht meist kein Projektor, der einen cosinusförmigen Intensitätsverlauf aufprojizieren kann, zur Verfügung. So ist auch der verwendete Projektor nur in der Lage, ein rechteckförmiges Streifenmuster zu projizieren. Da aber abhängig von der Projektor- und Kameraoptik und vom Abstand des Objektes vom Sensor immer leichte Unschärfen der feinen Linienstruktur vorhanden sind, ist die Näherung mit dem Cosinus hinreichend genau.
Die Phasenwerte wiederholen sich jedoch periodisch, aus diesem Grunde kann man dieses Verfahren nicht alleine für die Messung verwenden. In Kombination mit dem Gray-Code kann aber die Auflösung des Sensors und damit auch die Meßgenauigkeit wesentlich erhöht werden [RIC 95].

2.4 Beispiele für die Realisierung von Optischen 3D - Meßsystemen

Geführte Sensoren

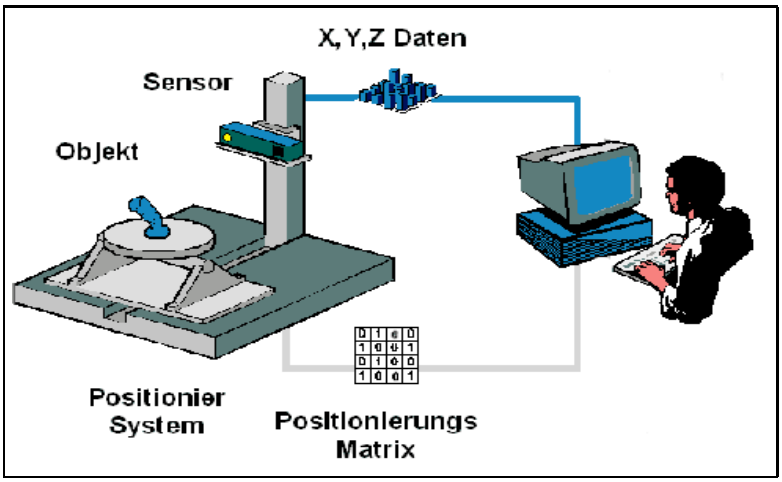


Abbildung 2.8 : Prinzipskizze eines geführten Sensors mit Positioniereinheit

Bei dem **Ganymed 6 Sensor** aus Abb. 2.9 handelt es sich um einen Streifenprojektionssensor der auf die Z-Achse eines absolut genauen Fünfachsen-Positionniersystems montiert ist. Das zu messende Objekt wird auf einen in X- und Y-Richtung verfahrbaren Dreh- und Kipptisch (links im Foto)befestigt. Das zu messende Objekt darf maximal 12 kg wiegen. Der Verfahrbereich der X und Y-Achse liegt bei ca. 100 x 100 cm. Der Sensor kann auf der Z-Achse (Bildmitte) um ca. 80 cm vertikal bewegt werden. Die Steuerung des gesamt Systems erfolgt durch die DigiMan Software auf einer Silicon Graphics Workstation auf der die aufgenommene Punktwolke visualisiert werden kann (rechter Teil des Bildes). Eine Messung mit dem auf der verfahrbaren Z-Achse montierten Streifenprojektions Sensor, wird von einer einzige Benutzeroberfläche auf diesem Rechner gesteuert. Das Steuerungsprogramm (DigiMan) synchronisiert die Messung, indem es mit den beiden Personal Computern kommuniziert, die einerseits die Positionierung des Objektes und des Sensors auf der Fünf-Achsen Positioniereinheit und andererseits die Aufnahme einer Folge von Grauwertbildern und die Berechnung des sogenannten ζ-Bild (Tiefenbild) aus dieser Folge, steuert. Mit denen vom Positionierrechner

bekannten Positionen der Achsen wird eine Transformationsmatrix aufgestellt, mit der das aktuelle vom Bildverarbeitungsrechner zur Workstation übertragene ζ-Bild in das für alle Ansichten festgelegt Objekt Koordinatensystems bestimmt wird. Danach kann das Positioniersystem in eine neue Position für die Messung der nächsten Ansicht gefahren werden oder es können Auswertungen der Meßdaten mit den in die DigiMan- Software integrierten Funktionen getätigt werden.

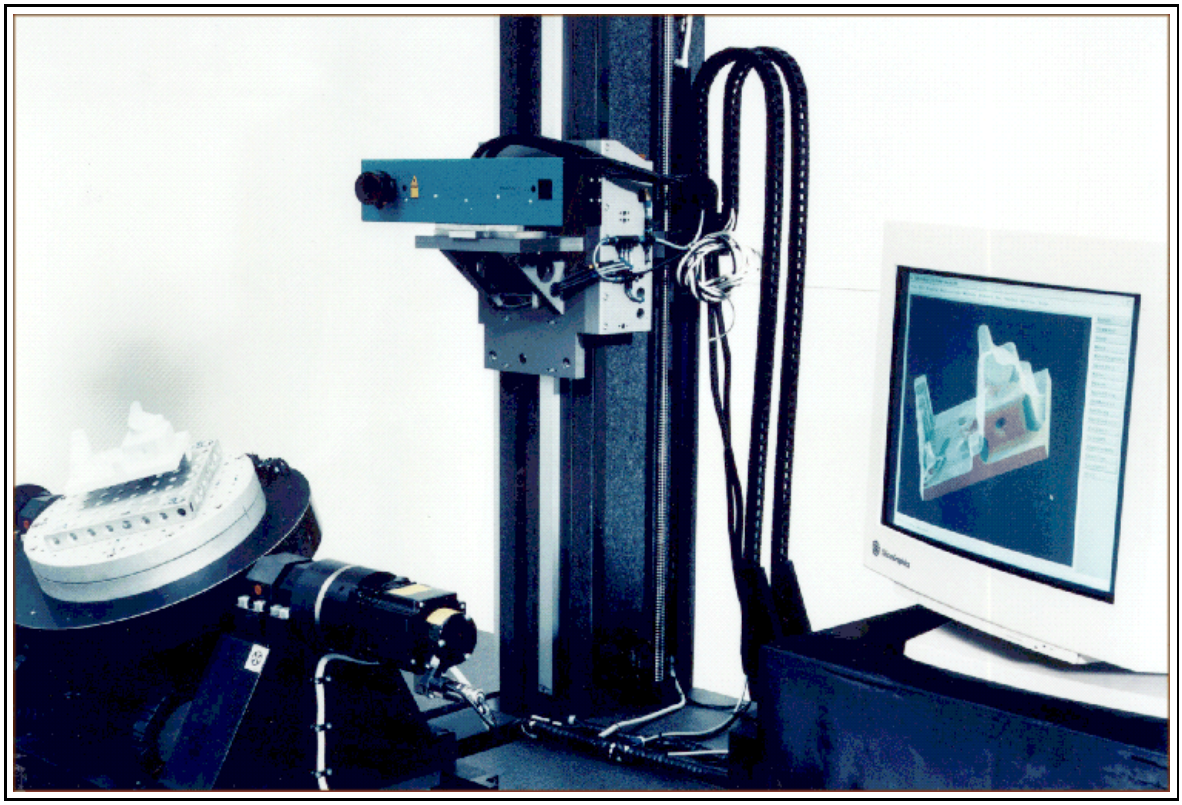


Abbildung 2.9 : Ganymed 6 Sensor, (c) Daimler-Benz AG, FT4/TM, Ulm, 1996

Fabrikrealisierungen

In Abbildung 2.10 folgen noch drei Bilder zu Realisierungen von 3D Meßsystemen als geführte Sensoren auf Basis der in den vorausgegangenen Abschnitten erläuterten Prinzipien der Streifenprojektion.

Freifliegende Sensoren

Der freifliegende Sensor Ganymed 8 plus aus Abb. 2.12 besteht aus einen Sensorkopf, der auf einem per Hand freibeweglichen Stativ montiert ist. Der Sensorkopf enthält eine hochauflösende (ca. 1.000 x 1.000 Pixel) digitale Kamera (rechts) und einen Streifenprojektor (links), mit dem ein Muster von

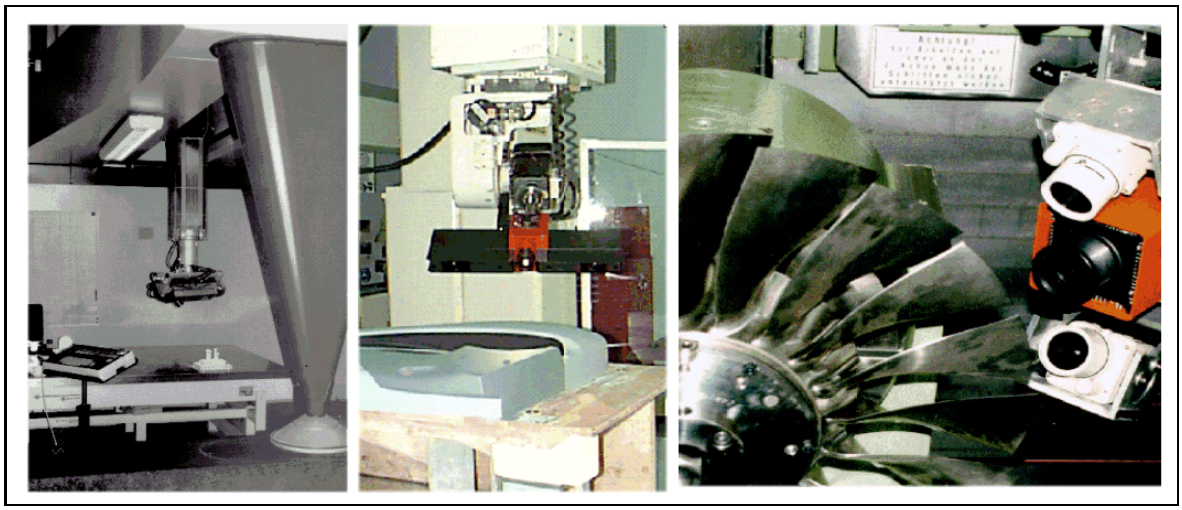


Abbildung 2.10 : 3 Beispiele geführter optischer Sensoren

Bild links : *Mercedes-Benz Werk, Bremen, 1996, Optischer Sensor an Koordinaten Meßmaschine gehängt.*

Bild mitte : *Freightliner Corp., Portland - USA, 1997, Optischer Sensor an eine Fräsmaschine gehängt.*

Bild rechts : *MTU, München, 1997, Sensorkopf für In-Prozeß Messungen.*

bis zu 2.048 Spalten auf die Oberfläche des Meßobjektes projiziert werden kann. Der Abstand des Sensors zum Meßobjekt sollte ca. 30 cm betragen (je nach Einstellung der Kameraparameter). Auf dem Meßobjekt (hier die Seitenwand einer Mercedes E-Klasse) sind reflektierende Marken (Retrotargets) aufgebracht, die zur Photogrammetrischen Navigation benutzt werden. Für ein erfolgreiches, hochgenaues Zusammensetzen von mehreren Teilansichten müssen jeweils vier dieser Marken in einer Bildsequenz des Streifenprojektionsverfahrens sichtbar sein. Die zu messende Bildgröße beträgt dabei ca. 50 x 50 cm.

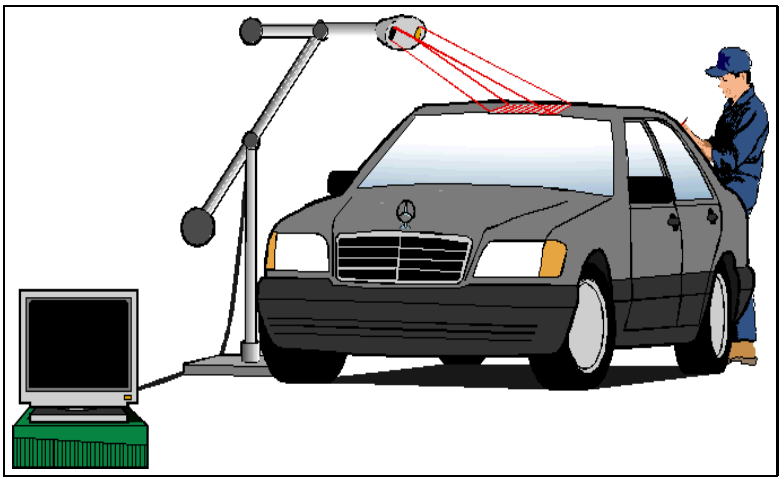


Abbildung 2.11 : Prinzipskizze eines freifliegenden Sensors ohne Positioniereinheit

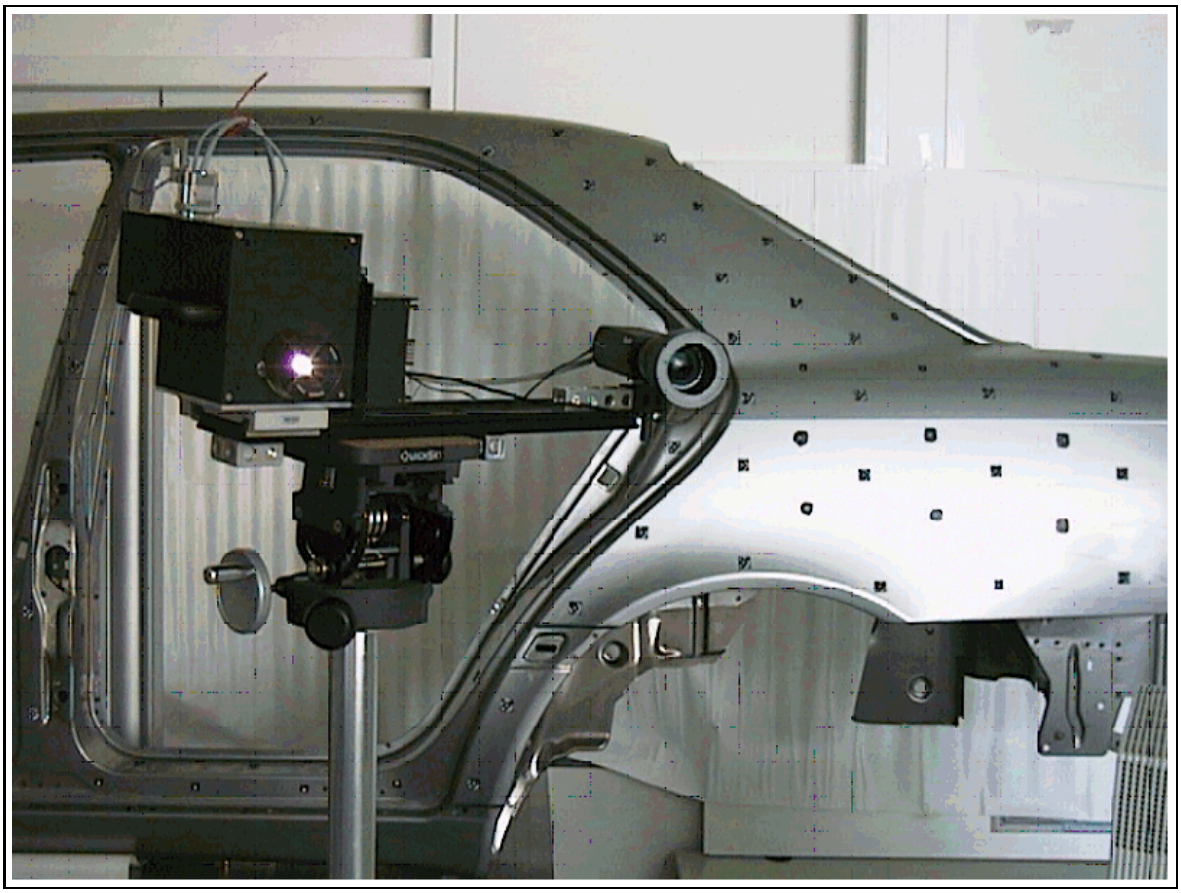


Abbildung 2.12 : Freifliegender Sensors Ganymed 8 plus, (c) Daimler-Benz AG, FT4/TM, Ulm, 1997

Abbildung 2.13 : Komplette Vermessung eines Mercedes SLK mit dem freifliegendem Sensor Ganymed 97, (c) Daimler-Benz AG, FT4/TM, Ulm, 1997

3 Datenstruktur für große Punktwolken

In diesem Kapitel werden verschiedene Möglichkeiten zur Verwaltung von großen 3D Punktwolken vorgestellt. Diese Möglichkeiten wurden in der Diplomarbeit von Werner Jungbauer [JUN 95A] untersucht und anschließend implementiert [JUN 95B]. Sie sind im folgenden in einer verkürzten Form dargestellt.
Zunächst werden die Anforderungen an eine geeignete Datenstruktur im Hinblick auf darauf aufbauende Anwendungen aufgelistet. Danach erfolgt eine Vorstellung der verschiedenen in der Literatur erwähnten Datenstrukturen. Literatur hierzu findet sich in den Gebieten der Graphischen Datenverarbeitung und der digitalen Bildverarbeitung.
Die verschiedenen Datenstrukturen werden in Bezug zu den Anforderungen gestellt. Daraus ergibt sich die gewählte Lösung, die sogenannte **Quader - Octree** Datenstruktur, die im dritten Abschnitt nochmals näher erklärt und auf einige Meßdatenbeispiele angewendet wird. Zum Abschluß des Kapitels werden einige Beispiele von Anwendungen und Algorithmen auf der entwickelten Datenstruktur aufgeführt.

3.1 Entwicklung einer Datenstruktur für große Punktwolken

Die erste Hauptaufgabe besteht in der Entwicklung der Datenbasis bzw. Datenstruktur zur Verwaltung und Archivierung von dreidimensionalen Meßdaten. Die Eingangsdaten liegen in Koordinaten-Tripeln aus einem gemeinsamen Koordinatensystem vor (Objektkoordinatensystem). Es soll eine möglichst geeignete Datenstruktur gefunden und implementiert werden, die unter Berücksichtigung der im nächsten Abschnitt aufgelisteten Anwendungen günstige Eigenschaften aufweist und eine einheitliche Schnittstelle zum Zugriff auf diese Daten ermöglicht. Eine geeignete Datenstruktur ist zudem in der Lage, bestimmte Operationen dieser Anwendungen zu vereinfachen und zu beschleunigen. Zur Entwicklung dieser Datenstruktur sollen die auch in der Literatur diskutierte Strukturen untersucht und auf ihre Einsatzfähigkeit in Bezug auf die benötigten Anwendungen untersucht werden.
Während im ersten Teilabschnitt die Herkunft und das auftretende Datenvolumen skizziert wird folgt im zweiten Teilabschnitt die Anforderungen aus möglichen Anwendungen und im dritten Abschnitt die Anforderungen aus Algorithmen an die zu entwickelnde Datenstruktur.

3.1.1 Möglichkeiten zur Strukturierung großer Punktwolken

Grundsätzlich handelt es sich bei den Eingangsdaten um Punkttripel im 3D - Raum, die durch ihre x, y und z -Koordinate beschrieben werden. Eine Menge dieser Koordinaten, die durch eine Messung gewonnen werden, wird Punktwolke genannt. Diese Punktwolke beschreibt die erhaltenen Oberflächendaten des gemessenen Objekts einer Ansicht, d.h. eine Messung. Die Summe aller Ansichten eines gemessenen Objekts ergibt die Punktwolke des gesamten Objekts, die die vollständige Oberflächenbeschreibung durch Meßpunkte darstellt. Die Form, in der diese Punktedaten vorliegen, beschränkt sich im Algemeinen auf zwei Arten die *Punkteliste* oder die *Punktematrix*.
Die Meßpunkte können linear in einer Punkteliste vorliegen. Diese Punkteliste kann unsortiert bzw.

auch nach diversen Kriterien (x, y, z-Koordinate, nächster Nachbar) sortiert vorliegen. Da die gewonnenen Datenpunkte einer optischen Messung entstammen bzw. von einer CCD-Kamera erfaßt wurden, liegen sie in ursprünglicher Form als eine Punktematrix vor. Diese Matrix verfügt in unserem Fall über eine bekannte Dimension (z.B. 640 x 512 oder 1.024 x 1.024 Pixeln). Jedes Element dieser Matrix ist ein Punkt, der durch seine 3 Koordinaten, evtl. noch durch zusätzliche Informationen (z.B. Qualitätswert) beschrieben wird.
Diese Matrix hat zusätzlich die Eigenschaft, daß nicht jedes Element einen gültigen Datenpunkt darstellt. So ist es durchaus möglich, daß z.B. nur 15% aller Punkte der Matrix gültige Datenpunkte sind. Aus der Matrix selbst ist zusätzlich noch eine Nachbarschaftsbeziehung der Datenpunkte abzuleiten. So verfügt im allgemeinen jeder Datenpunkt (außer Randpunkte) über 8 Kanten- Eckennachbarn, die allein durch Indizierung erreichbar sind.

$P_{0,0}$	•	$P_{0,2}$	•	$P_{0,4}$	$P_{0,5}$	$P_{0,6}$
$P_{1,0}$	$P_{1,1}$	•	$P_{1,3}$	•	$P_{1,5}$	•
•	$P_{2,1}$	$P_{2,2}$	•	$P_{2,4}$	$P_{2,5}$	$P_{2,6}$
$P_{3,0}$	$P_{3,1}$	$P_{3,2}$	$P_{3,3}$	•	$P_{3,5}$	$P_{3,6}$
$P_{4,0}$	•	$P_{4,2}$	$P_{4,3}$	•	•	$P_{4,6}$
$P_{5,0}$	$P_{5,1}$	$P_{5,2}$	$P_{5,3}$	•	$P_{5,5}$	•

Abbildung 3.1 : 6 $\times$ 7 Sensorpunktmatrix mit gültigen $P_{i,j}$ und ungültigen • Datenpunkten

Abhängig von der Größe eines zu messenden Objekts fallen dementsprechend große Datenmengen an. Wenn auch durch die beschränkte Größe des Sensorfensters eine Grenze gegeben scheint, kann diese durch Translationen des Sensors bzw. des Objekts beinahe beliebig erhöht werden. So kann z.B. die Messung eines gesamten PKW zu ca. 100 Ansichten mit je 1.024 x 1.024 = 1.048.576 Datenpunkten mit 20 Byte Speicherbedarf führen. Diese Datenmenge von ca. 2 Gigabyte Speicher gilt es effizient zu verwalten um folgende Anwendungen überhaupt bzw. vernünftig realisieren zu können. Zudem führt der Umfang der Daten auch zu gewissen Anforderungen, die das zugrundeliegende Betriebssystem vor allem im Speichermanagement zu erfüllen hat.

3.1.2 Überblick von Anwendungen auf 3D - Punktwolken

Die Anforderungen an die zu entwickelnde Datenstruktur lassen sich einerseits durch allgemeine Forderungen, andererseits durch anwendungsspezifische Eigenschaften herleiten. Anwendungen, die auf der Datenstruktur operieren sind z.B. :

- Datenanalyse und Meßqualitäts-Untersuchung
- Glättungsalgorithmen
- Ausdünnung zum Zweck der Datenreduktion
- Extraktion von Punkten

- Erkennung von Klaffungen und Lücken
- Erkennung von Regelgeometrien
- Bestimmung von Schnitten durch die Punktwolke (Cross Sections)
- Triangulierung der Punktwolken
- Segmentierung der Punktwolke
- Flächenrückführung (durch Freiformflächen)

Um die Anforderungen dieser Algorithmen an die Datenstruktur verstehen zu können, sollen diese kurz beschrieben werden.
Glättungsalgorithmen dienen der Glättung der Meßdaten. So sollen einzelne Ausreißer erkannt und eliminiert werden können, da solche Meßfehler die Funktionsweise später ablaufender Algorithmen stark beeinträchtigen können. Die Tendenz zeigt jedoch, daß Glättungsoperationen auf Seite der Bildverarbeitung günstiger zu implementieren sind als auf transformierten Koordinatendaten.
Um die enormen Datenmengen der optischen Meßtechnik besser (Speicherverbrauch und Algorithmen-Effizienz) handhaben zu können, sollten die Punktwolken intelligent ausgedünnt werden. Unter *Ausdünnung* sind hierbei Algorithmen zu verstehen, die mit dem Ziel der *Datenreduktion* Datenpunkte entfernen. Entfernungskriterien können hierbei einfache Sampling- bzw. auch krümmungsabhängige Faktoren sein. Es muß darauf geachtet werden, daß wichtige informationstragende Datenpunkte erhalten bleiben. Zu diesen zählen z.B. Punkte mit erhöhter Krümmung in Bezug auf Nachbarpunkte innerhalb des Polygonzugs, da diese Berandungspunkte für später anzugleichende Ebenen darstellen können.
Schnitte durch die Punktwolke (engl. **Cross - Sections**) sind Punkte, die innerhalb eines gewissen Toleranzbandes auf eine vorgegene Ebene parallel zu einer der drei Hauptebenen projizierbar sind. Diese Schnittoperation läßt sich zunächst auf eine Punktselektion zurückführen.
Zum Bereich der *Flächenrückführung* zählt die *Erkennung von Regelgeometrien.* So sollen die vorliegenden Punktedaten analysiert und - falls vorhanden - Regelgeometrien wie z.B. Kugelsegmente, Zylinder etc. angeglichen werden. Dieser Vorgang geschieht noch oft interaktiv, d.h. ein Benutzer erzeugt aufgrund konstruktiver Kenntnisse des Objekts auf einer selektierten Punktemenge eine Regelgeometrie.
Zu den weiteren Anwendungen einer Datenstruktur zählt die *Punktextraktion.* Hier sollen Punktedaten unter Berücksichtigung eines Zeitlimits nach vorgegebenen Kriterien aus der Datenstruktur extrahiert werden. So können z.B. in ihrer Elementanzahl eingeschränkte CAD-Systeme zur Laufzeit Punkte zur Bearbeitung anfordern.

3.1.3 Anforderungen an eine Datenstruktur für große Punktwolken

Nach der groben Beschreibung von Anwendungen, die auf Punktdaten operieren sollen, wird nun auf die Anforderungen der Datenstruktur eingegangen. Folgende Punkte werden erläutert :

- Effiziente Handhabung beliebiger 3D Punktwolken
- Dynamisches Wachstum der Datenmenge
- Selektionsmöglichkeit und Bearbeitung eines beliebigen Punktes
- Wertebereich muß alle vorkommenden Objekte verwalten können
- Datenverwaltung ohne Genauigkeitsverlust
- Kompaktheit (Geringer Verbrauch an Speicherressourcen)
- Unterscheidung mehrerer Ansichten
- Schnelle Selektion/Extraktion von Punkten
- Schnelle Bestimmung benachbarter Punkte
- Bestimmung von Punkten nach diversen Kriterien
- Einfaches und geordnetes Scanning der Objektoberfläche

In der Datenstruktur muß es möglich sein, beliebig große Punktwolken (über eine Million Punkte) zu verwalten und es soll dabei kein unnötig hoher Verwaltungsaufwand für die Handhabung von kleineren Punktmengen entstehen. Deshalb muß die Datenstruktur parametrisierbar sein, um eine *Effiziente Handhabung* von 3D - Punktwolken zu ermöglichen.
Durch das Messen neuer Ansichten muß die Datenstruktur zur Laufzeit das Einfügen neuer Datenpunkte zulassen. Dies erfordert je nach Komplexität der gewählten Struktur unter Umständen einen erhöhten Aufwand, da auch bereits bestehende Ordnungen neu aufgebaut werden müssen. Diese Operation sollte dennoch in einem erträglichen Zeitrahmen (einige Sekunden) realisierbar sein, um die praktische Verwendbarkeit der Datenstruktur zu gewährleisten. Während dies eine grundsätzliche Operation darstellt, ist z.B. das einzelne Einfügen vieler Datenpunkte in die Struktur weniger gebräuchlich, da die Meßdaten nur in einer Punktemenge anfallen. Algorithmen müssen natürlich in der Lage sein, einzelne Datenpunkte finden und diese bearbeiten zu können. Dieser Vorgang kann z.B. durch eine Selektion in einer Visualisierungsansicht ausgelöst werden.
Der *Wertebereich* der Datenstruktur muß eine Größe besitzen, die die Verwaltung aller vorkommenden Objekte erlaubt. So sollte die Datenstruktur ihren Wertebereich nicht unnötig einschränken und somit Verluste von evtl. wichtigen Datenpunkten hinnehmen. Dieses Problem kann vor allem bei festparametrisierten Datenmodellen auftauchen, deren Aufbau im voraus durch den Wertebereich der zu verwaltenden Daten bestimmt wird. Es sollte gewährleistet sein, daß das Einfügen einer neuen Ansicht, deren Wertebereich den der Datenstruktur übersteigt, nicht zu Datenverlusten führt. Aus Performancegründen kann es jedoch von Vorteil sein, den Wertebereich der Daten festzulegen. Unter Kenntnis des Wertebereichs des zu messenden Objekts bzw. des Meßsystems ist diese Vorgehensweise auch akzeptabel.
Die Datenverwaltung sollte zudem ohne *Genauigkeitsverlust* arbeiten. Dies setzt einerseits die Verwendung eines geeigneten Datentyps voraus, andererseits dürfen die verwalteten Daten nicht manipuliert oder in sonstiger Weise in ihrer Genauigkeit verändert werden.

Ein zusätzliches Kriterium an eine Datenstruktur stellt deren *Ressourcenverbrauch* dar. Da es sich bei der zu entwickelnden Datenstruktur um eine speicherresidente handelt - d.h. im flüchtigen Speicher liegend, nicht auf Permanentspeicher - sollten ihre Anforderungen momentan übliche Arbeitsspeichergrenzen nicht sprengen. Der Ressourcenverbrauch wird jedoch auch einen Kompromiß darstellen : die effiziente Speicherung und schnelle Unterstützung bestimmter Operationen ist nur unter Verwendung von zusätzlicher Information zu realisieren. Hier ist eine Lösung zu wählen, die unter möglichst großem Nutz/Verwaltungs-Verhältnis die beste Performance bietet.
Die Datenstruktur muß durch ihre Verwaltung ferner in der Lage sein, die *Punkt-Ansichtszugehörigkeit* sicherzustellen. Es muß für jeden Punkt die Möglichkeit bestehen, seine Ansichtsnummer zu erfragen, da diese für manche Algorithmen essentiell ist.
Die schnelle *Selektion und Extraktion* von Punkten zählt zu den grundsätzlichen Eigenschaften der Datenstruktur. So sollen z.B. Punkte innerhalb eines umschließenden Bereichs bzw. Punkte nach sonstigen Kriterien aus der Struktur extrahiert werden können. Eine schnelle Extraktion dieser Punkte erfordert zusätzliche Verwaltungsinformation, damit die sequentielle Prüfung aller Datenpunkte auf ein Minimum reduziert wird. Möglich sind hier Näherungsverfahren bzw. eine geordnete Strukturierung der Datenpunkte.
Ebenfalls zu den Grundanforderungen an die Datenstruktur zählt die Möglichkeit, zu einem gegebenen Punkt seine Nachbarpunkte zu finden. Diese *Nachbarschaftssuche* sollte zudem nach wählbaren Kriterien möglich sein, so sind z.B. nächste Nachbarn oder Richtungsnachbarn denkbar. Diese Anforderung wird vor allem vom Schnittalgorithmus gestellt, der in einer vorgegebenen Schnittrichtung zu einen Punkt in den nächsten Nachbarn zu finden hat.
Die Datenstruktur sollte *punkt- und punktemengenorientiertes Scanning* unterstützen. Das heißt es sollte möglich sein, mit einfacher Algorithmik die Datenstruktur und deren Inhalte durchqueren und untersuchen zu können.

3.2 Vorstellung verschiedener Datenstrukturen für 3D - Punktwolken

Nach der Vorstellung der Anwendungen und der Definition der Anforderungen an die Datenstruktur sollen nun mögliche Lösungen aufgezeigt werden. Zudem soll auch versucht werden, Datenstrukturen nach verschiedenen Kriterien zu klassifizieren. Bei den in diesem Abschnitt erwähnten Datenstrukturen handelt es sich um allgemeine, in der Literatur erwähnte Strukturen. Diese sollen als Basis für die zu findende Datenstruktur dienen. Bei den hier erwähnten Angaben handelt es sich teilweise um Datenstrukturen, die auch in [Sam 90] oder [FovDa 90] diskutiert werden.
Viele Anwendungen benötigen eine effiziente Verwaltung von räumlichen Daten. Zu diesen Anwendungen zählen z.B. Computergraphik, CAD, Solid Modeling, Robotik, Geographie, Kartographie, Bildverarbeitung, etc. . Entsprechend der Anwendung wurden bereits räumliche Datenstrukturen für Punkte, Linien, Flächen und Volumenelemente entwickelt und vorgestellt : Während z.B. in Anwendungen der Kartographie vorwiegend Datenpunkte verwaltet werden, befaßt sich der Themenbereich Solid Modeling mit der Repräsentation von Objekten durch Volumenelemente. Diese Volumendaten sind vor allem für die Kollisionserkennung bzw. allgemeine Schnitt- und Vereinigungsmethoden von

Vorteil.
Neben der Klassifikation nach der Art der zu verwaltenden Daten, kann zusätzlich der Aufbau einer Datenstruktur unterschieden werden zwischen *hierarchische* und *nicht-hierarchische* Datenstrukturen, wobei hierarchische mit dem Ziel einer effizienteren Speicherung vor allem im Bereich der Bildverarbeitung entwickelt wurden. Sie basieren auf einer rekursiven Teilung der Datenelemente, auch *divide and conquer* - Methoden genannt. Manche Operationen lassen sich mit diesen Datenstrukturen effizienter realisieren. Durch die gegebene Ordnung der Datenelemente in hierarchischen Datenstrukturen können diese sehr stark Suchoperationen unterstützen, welche auch im Umfeld dieser Diplomarbeit zu lösen sind. Um Datenstrukturen auf ihre Anwendbarkeit in Bezug auf bereits geschilderte Probleme untersuchen zu können, muß vorerst eine Eingrenzung vorgenommen werden. Dies betrifft in erster Linie die Art der zu verwaltenden Daten. Da das Grundelement einer optischen Messung ein Punkt, bestehend aus seiner $x/y/z$- Koordinate ist, sind vor allem Datenstrukturen zu untersuchen, die die Verwaltung von Punktedaten erlauben. Zusätzlich sollten jedoch auch Volumenverfahren untersucht werden, da eine Menge von Punkten auch näherungsweise ein Volumen darstellen.

3.2.1 Nichthierarchische Datenstrukturen

Bei nichthierarchischen Datenstrukturen handelt es sich um die einfachste Form der Verwaltung. Hierbei werden die Datenpunkte in einer Liste abgelegt.
Suchoperationen auf einer solchen Struktur sind von der Ordnung $O(N)$, da jedes Element von insgesamt N Listenelementen untersucht werden muß.

Name	**x**	**y**	**z**
P_1	35	40	20
P_2	50	10	30
P_3	60	75	80
P_4	5	45	55
P_5	25	35	10

x	**y**	**z**
P_4	P_2	P_5
P_5	P_5	P_1
P_1	P_1	P_2
P_2	P_4	P_4
P_3	P_3	P_3

Abbildung 3.2 : Prinzip von Punktliste (links) und Punktliste mit sortierten Verweisen (rechts)

Diese Struktur kann durch Hinzufügen von sortierten Verweislisten erweitert werden. Bei vorliegenden 3D - Datenpunkten kann z.B. nach der x, y- und z - Koordinate sortiert werden. Suchoperationen können unter Vorhandensein solcher sortierter Verweislisten sehr effizient erfolgen.
Während unter Verwendung einer ungeordneten Liste kein zusätzlicher Speicherbedarf für Verwaltungsstrukturen benötigt wird, sind für zusätzliche Ordnungslisten pro Punkt 3 Verweiseinträge nötig. Ein Verweiseintrag entspricht hierbei der Größe eines Zeigers in der jeweiligen Programmiersprache bzw. des Betriebssystems.
Eine weitere gängige Datenstruktur unter der Gruppe der nichthierarchischen stellt die Zellenunterteilung dar. Bei diesem Verfahren wird das zu verwaltende Bild bzw. der zu verwaltende 3D - Raum in Zellen bzw. Volumenelemente gleicher Größe unterteilt. Durch diese Voraussetzung handelt es sich hierbei um Fixed-Grid -Methoden. Durch die variable Gestaltung der Anzahl der Spalten und Zeilen

ist es möglich, die Auflösung dieser Datenstruktur zu verändern. Die Größe einer Zelle sollte an die gedachten Radien für Suchoperationen angepaßt werden, um das Problem der Nachbarschaftssuche effizient zu reduzieren.
Während Bild 3.3 nur eine Fixed-Grid-Struktur in 2D andeutet, kann diese natürlich ebenso in 3D implementiert werden. In diesem Fall handelt es sich um ein dreidimensionales Feld. Feldelemente bzw. Zellenelemente sind eine Liste von Punkten, die - durch ihre Koordinaten bestimmt - dieser Zelle angehören.

3.2.2 Hierarchische Datenstrukturen

Nach den nichthierarchischen Datenstrukturen sollen nun die wichtigsten hierarchischen besprochen werden, wobei unter den zahlreichen räumlichen Strukturen nur auf die eingegangen wird, die speziell zur Verwaltung von Datenpunkten geeignet sind. Die Hauptmotivation bei der Entwicklung hierarchischer Datenstrukturen stellte die Reduktion von benötigtem Speicher dar. Der Grundgedanke hierbei war die Zusammenfassung von Gebieten und Objekten, die durch identische Attribute beschrieben werden können. So werden diese Strukturen vor allem in der Digitalen Bildverarbeitung eingesetzt, um Bildsegmente mit einheitlicher Farbe als eine Verwaltungseinheit behandeln und somit reduzieren zu können. Die herkömmliche Speichermethodik sieht für jeden Bildpunkt bzw. Pixel eine separate Speicherstelle vor, was bei großen Pixelflächen zu hohem Speicherbedarf führt. Neben dem Effekt der Speicherreduktion treten unter Verwendung von hierarchischen Datenstrukturen zusätzlich Performance- Vorteile auf, d.h. sie erlauben oft die beschleunigte Operation mancher Algorithmen.
Hierarchische Datenstrukturen basieren auf Bäumen zu deren Elemente Knoten und Blätter zählen. Die einfachste Form eines Baumes ist der Quadtree, der auch in der Bildverarbeitung verwendet wird. Hierbei hält jeder Knoten Verweise auf 4 weitere Kindknoten, welche in diesem Fall Rechtecke darstellen (vgl. Abbildung 3.3 linkes Bilds).
Der Einstieg in eine hierarchische Datenstruktur erfolgt über den Wurzel- (engl. Root-) Knoten. Dieser ist selbst nur ein Knoten. Blätter stellen Elemente des Baums dar, die keine weiteren Kindknoten besitzen. Sie sind daher meist nutzdatentragende Elemente. Knoten verfügen dagegen meist über keine Nutzdaten, jedoch über Verweise auf Kindknoten, welche wiederum Blätter sein können.

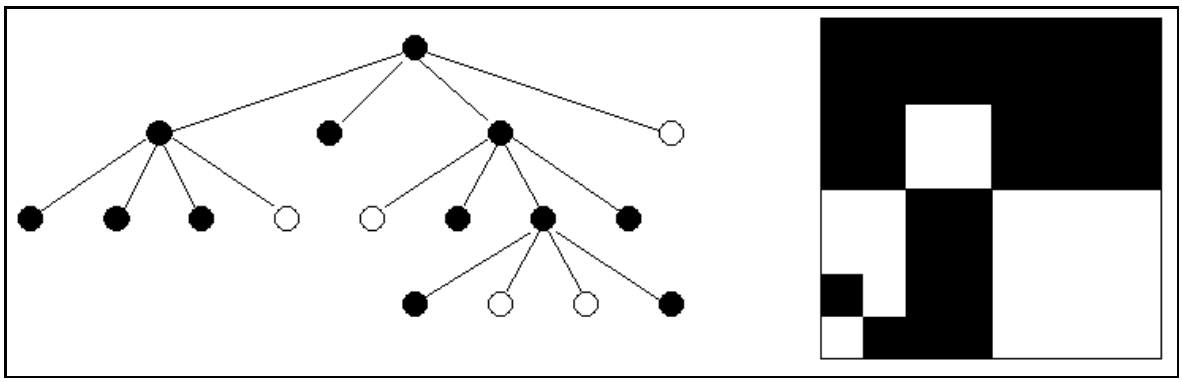


Abbildung 3.3 : Skizze zur Quadtree Datenstruktur für 2D-Punkte

Die dreidimensionale Variante des Quadtrees stellt der Octree dar. Hier bilden Knotenelemente Oktanten, da ein Knoten über maximal 8 Kindknoten verfügt. Der Aufbau hierarchischer Datenstrukturen basiert auf der rekursiven Teilung des Eingangsbildes bzw. des Eingangsraumes (rekursive Dekomposition). Sie können auch grundsätzlich folgendermaßen unterschieden werden :

- zu verwaltender Datentyp,
- Prinzip der Dekomposition,
- Auflösung.

Das Prinzip der Dekomposition definiert Bedingungen, welche die weitere Unterteilung eines Knotens in Kindknoten zuläßt. Hier werden anwendungsabhängig diverse Kriterien verwendet. Die Auflösung bestimmt die Anzahl der Elemente, die entlang einer Seitenkante gezählt werden. Durch die stetige Teilung des Raumes beträgt die Auflösung grundsätzlich 2^N, wobei N mit $N \geq 0$ die maximal zulässige Baumtiefe spezifiziert. Entscheidend für die Implementierung einer hierarchischen Datenstruktur kann auch die Anforderung sein, die Auflösung konstant bzw. zur Laufzeit oder in Abhängigkeit von einzufügenden Daten zu verändern. Änderungen der Auflösung einer bestehenden hierarchischen Struktur bedeuten nicht selten eine umfangreiche Reorganisation der bereits vorhandenen Daten. Folgende Eigenschaft liegt allen hierarchischen Datenstrukturen zugrunde :

Hierachische Datenstrukturen eignen sich zur Implementierung schneller Algorithmen

Einfügeoperationen sind jedoch meist weitaus aufwendiger als unter Verwendung konventioneller Datenstrukturen. Ihr Einsatz muß daher anwendungsabhängig geprüft und entschieden werden.
Nach der allgemeinen Einführung in hierarchische Datenstrukturen sollen nun konkrete Überlegungen folgen, die die Verwaltung von Datenpunkten zum Ziel haben.
Die erste Variante stellt der Punkt-Octree dar. Verwaltungs- bzw. Knotenelement ist ein einzelner Datenpunkt, der durch seine X, Y und Z-Koordinate beschrieben wird. In Punkt- Quadtrees und Octrees enthält jeder Knoten einen Datenpunkt, ein Knoten kann somit auch ein Blattknoten sein. Zusätzlich verfügt jeder Knoten über Verweise auf seine Kindknoten, in Octrees deren 8. Die Definition der Oktanten in Bezug auf einen einzelnen Datenknoten kann unter Verwendung der obigen Angaben erfolgen.
Eine gängige Suchanfrage an eine solche Datenstruktur könnte z.B. lauten : Finde alle Punkte innerhalb eines Suchradius ausgehend von einem bestimmten Datenpunkt. Unter Kenntnis des Suchpunktes und des Radius ist bekannt, welche Koordinatenbereiche zu untersuchen sind. Von der Wurzel ausgehend kann der Baum nun gerichtet traversiert werden. Der Vergleich der Koordinaten des aktuellen Datenknotens mit der spezifizierten Suchregion ergibt jeweils die zu untersuchenden Oktanten. Weitere Oktanten, die aufgrund ihrer Koordinaten nicht in den Suchbereich fallen können, werden nicht weiter untersucht.
Diese Darstellung zeigt bereits die Vorteile hierarchischer Datenstrukturen in Bezug auf Suchalgorithmen auf. Sie erlauben zu beurteilen, ob die weitere Bearbeitung eines Knotens Ergebnisse liefern oder ob die Suche in diesem Teilbaum bereits abgebrochen werden kann. Diese Möglichkeit führt zu

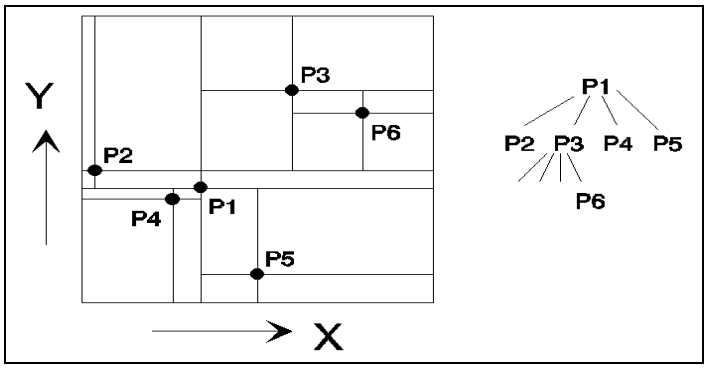


Abbildung 3.4 : Skizze zur Punkt-Quadtree Datenstruktur in 2D

wesentlich effizienteren Suchalgorithmen, als dies mit herkömmlichen Strukturen zu realisieren wären. Während bisherige Behandlungen von hierarchischen Datenstrukturen immer nur einen Datenpunkt als Element hatten, soll nun ein Bereichs-Octree (region-octree) vorgestellt werden. Dieser verwaltet Räume, denen Datenpunkte entsprechend ihrer Koordinaten zugeordnet werden können. Diese Struktur ist auch im Themengebiet des Solid Modeling oft anzutreffen, wo Objekte nicht nur durch ihre Oberfläche, sondern durch Raumelemente repräsentiert werden.
Zur Verwaltung räumlicher Daten stehen noch viele weitere Datenstrukturen zur Verfügung (z.B. B-Reps, Sweep- Representations, etc.), die jedoch nicht in den Rahmen der gedachten Anwendung fallen, da diese meist nicht in der Lage sind, einzelne Datenpunkte zu verwalten.
Nach der Vorstellung möglicher Grundstrukturen sollen im folgenden diejenigen weiterentwickelt und untersucht werden, die für die gedachten Anwendungen am besten geeignet sind. Unter Anbetracht der Randbedingungen der optischen Meßtechnik und ihrer anfallenden Datenmengen wurde von einer weiteren Untersuchung der punktorientierten Datenstrukturen wie z.B. Punkt-Octrees abgesehen und die Arbeit vor allem auf regionenbasierte Strukturen konzentriert.

3.2.3 Die Fixed - Grid Datenstruktur

Hierbei handelt es sich um eine regionenbasierte Datenstruktur, d.h. Verwaltungsgegenstand sind Raumelemente. Jedes Raumelement ist in seiner Lage und Größe genau bekannt, vorgegeben durch seine Position in der 3D - Matrix. Die Lage und Größe kann einfach durch eine Boundingbox beschrieben werden, welche z.B. aus zwei räumlich diagonal gegenüberliegenden 3D - Punkten bestehen kann. Jedes Raumelement muß in der Lage sein, die zu ihrer Region zuzuordnenden Datenpunkte verwalten zu können. Die Organisation kann beispielsweise in Form eines Feldes oder auch einer dynamischen Liste erfolgen.
Die Basis ist ein Feld mit $M \cdot N$ Verweisen auf Vektorlisten, die die Punktdaten in Form eines Feldes oder einer dynamische Liste verwalten Auch im Falle einer leeren Struktur der Kantenanzahl N wären N^3 Zeiger zu verwalten. Eine Optimierungsmöglichkeit bei nur partiell belegten Strukturen wäre der Verzicht auf die Höhen- Vektorlisten, in denen keine Daten zu verwalten sind. Im Hinblick auf die Randbedingungen, d.h. der Anzahl der zu verwaltenden Daten im Bereich der optischen Meß-

technik, ist jedoch der Verwaltungsspeicheroverhead in Bezug auf den Nutzdatenverbrauch teilweise zu vernachlässigen. Insbesondere ist dies bei Strukturen zu bedenken, die bei steigender Anzahl zu verwaltender Datenpunkte ein wesentlich günstigeres Nutz/Verwaltungs-Verhältnis bieten, wie auch in diesem Fall. Ein wesentlicher Vorteil dieser Struktur ist die schnelle Suchoperation. Während mit dieser Methode eine absolute Adressierung möglich ist, kann von diesem Punkt ausgehend durch Inkrementierung oder Dekrementierung der Indizes einfach in allen Raumrichtungen weitergesucht werden. Sind vorwiegend konstante Suchradien zu verwenden, so kann die Größe eines Raumelements diesem Suchradius angepaßt werden. Dies erfordert nur die Untersuchung derjenigen Datenpunkte, die in dem gewählten Raumelement verwaltet werden. Werden Suchradien größer gewählt, so kann durch Vergleich der Suchregion mit den benachbarten Zellen einfach festgestellt werden, welche Nachbarzellen zusätzlich zu untersuchen sind. Sprünge in Nachbarzellen sind - wie bereits erwähnt - durch relative Indizierungen einfach möglich.
Bisherige Untersuchungen an dieser Datenstruktur ergaben einen Suchaufwand der Ordnung $\mathcal{O}(F \cdot 2^k)$, wobei F die Anzahl der gefundenen Datenpunkte, 2^k die maximale Anzahl der zu untersuchenden Zellen angibt, wenn der Suchbereich den einer Zelle überschreitet.
Ein grundsätzliches Problem regionenbasierter Datenstrukturen ist folgendes : Zur Dimensionierung sind die maximale Größe der einzufügenden Daten und die Auflösung der Struktur wichtig. Diese Größen könnten einerseits vor Einfügung der Daten sinnvoll gewählt werden, was bei Vorhandensein der Daten auch kein Problem darstellt. Andererseits besteht auch die Möglichkeit, die Datenstruktur in Abhängigkeit der einzufügenden Datenpunkte anzupassen. Eine Überschreitung der einzufügenden Daten bezüglich der aktuellen Würfelgröße würde eine Reorganisation der Struktur zur Laufzeit mitsichbringen. Diese Operation kann die Zerstörung der bestehenden und den Aufbau einer neuen Struktur erfordern, was auch einen nicht unerheblichen Zeitfaktor darstellt. Insbesondere bei den großen Datenmengen der optischen Meßtechnik ist dies zu beachten. Vorteile dieser Struktur sind ihre einfachen Zugriffsmechanismen durch Inidizierung und somit auch die Möglichkeit zur einfachen Implementierung von Suchalgorithmen. Insbesondere Suchvorgänge in den $x/y/z$-Grundebenen sind einfach realisierbar.
Als Nachteil kann eine aufwendige Einfügeoperation angesehen werden, sollte eine dynamische Anpassung gewünscht werden. Ein grundsätzlicher Nachteil kann unter Umständen auch die konstante Zellengröße sein, was Anwendungen auf der Datenstruktur in ihrer Dynamik einschränken könnte.

3.2.4 Der Region - Octree

Der Region-Octree ist eine regionenbasierte hierarchische Datenstruktur, vorwiegend zur Modellierung von 3D - Objekten benutzt. Er verfügt über die bekannten Vorteile, die hierarchische Strukturen allgemein bieten. Während herkömmliche Anwendungen auf dieser Struktur ein Voxel meist nur als vorhanden bzw. nicht vorhanden interpretieren, müßte die Funktionalität eines Voxels dadurch erweitert werden, daß es Meßdatenpunkte verwalten kann. Einzelne Zellen bzw. Voxel der Struktur würden sich identisch mit denen einer Fixed- Grid-Struktur verhalten. Auch ihre Verwaltungsregion kann durch ihre Ordnung im Baum eindeutig bestimmt werden.
Bei der Implementierung muß zwischen Knoten und Blättern unterschieden werden. Knoten verfügen

über keine Datenpunkte, sie bilden nur übergeordnete Verwaltungsebenen. Kindknoten dürfen jedoch unterschiedlichen Typs sein, d.h. sowohl Knoten als auch Blatt. Jeder Knoten benötigt Verweise auf maximal 8 Kindknoten, seine Oktanten. Ein schematisches Beispiel für den hierarchischen Aufbau eines Region Octrees ist im linken Bild der Abb. 3.5 dargestellt.
Neben der Verwendung dynamischer Speicherelemente könnte eine Baumstruktur auch durch Listen realisiert werden. Dieses Verfahren erfordert jedoch die Vollallokation von Verweislisten für jede Baumebene, d.h. auch das Bekanntsein der Baumtiefe. Es stellt somit eine Einschränkung in der Baumgestaltung dar. Die Anzahl der benötigten Knoten beträgt 8^N (N = Knotentiefe pro Baumtiefe), was einen nicht unerheblichen Speicherbedarf mit sich bringt. Zudem stehen Verwaltungsinformationen für Knoten bereit, die nicht belegt sind. Allerdings spart dieses Vorgehen die benötigten 8 Oktantenverweise, die in herkömmlichen Baumstrukturen nötig sind.
Im Gegensatz zur Fixed-Grid-Struktur unterstützen Octrees weder eine direkte Indizierung in Volumenelemente, noch eine einfache Inkrementoperation oder ähnliches zur Ermittlung von Nachbarn. Unter Kenntnis eines Startpunktes muß dieser von der Wurzel aus gesucht werden. Der Suchpfad innerhalb des Baumes ist eindeutig durch Koordinatenvergleiche der Voxelregionen mit denen des Suchpunktes gegeben und erfordert keinerlei überflüssigen Vergleichsoperationen in anderen Teilbäumen. Diese Eigenschaft äußert sich in schnelleren Suchmethoden, als dies mit nichthierarchischen Datenstrukturen realisierbar wäre. Eine Bereichs- oder Nachbarschaftssuche gestaltet sich jedoch schwieriger. Diese Operation erfordert die Kenntnis über Nachbarvoxel in allen möglichen Richtungen. Als zusätzliche Erschwernis kommt hinzu, daß benachbarte Voxel nicht grundsätzlich dieselbe Größe besitzen müssen. Ein Voxel kann in 26 möglichen Richtungen Nachbarvoxel besitzen, bei Verwendung unterschiedlicher Voxelgrößen kann sich diese Zahl noch weitaus erhöhen. Zur Lösung solcher Probleme wurden bereits Algorithmen entwickelt, die auf der Findung des gemeinsamen Vaterknotens basieren und diesen Zugriffspfad zur Auffindung des Nachbarn spiegeln .
Unter Kenntnis der Regionengrenzen die jedes Voxel repräsentiert, können jedoch einfachere Verfahren Verwendung finden, wie noch gezeigt werden wird.
Zu den Vorteilen der Region-Octrees zählen schnelle und effiziente Suchalgorithmen, welche jedoch nicht durchgehend einfach implementiert werden können. Ein weiterer bedeutender Vorteil im Gegensatz zu Fixed-Grid-Methoden ist die Möglichkeit, die Voxelvolumen zumindest als Potenz von 2 variabel zu gestalten. Dies kann unter Berücksichtigung mancher Anwendungen ein bedeutender Vorteil sein.
Als Nachteil ist der höhere Verwaltungsaufwand anzusehen, welcher pro Knoten 8 Verweise für seine Oktanten erfordert. Diese Größe ist jedoch unter Relation zu den Nutzdatenmengen zu betrachten und daher unter Umständen nicht mehr ausschlaggebend.

3.2.5 Der Matrizen - Graph

Diese Struktur ist mit bisher erwähnten nicht zu vergleichen. Ihr liegt ein vollkommen neuer Gedanke zugrunde : anstatt alle gemessenen Punkte einer Ansicht mit den Datenpunkten der anderen Ansichten in ein gemeinsames Datenmodell zu integrieren, werden hier die Ansichten separat gehalten. Da die ursprüngliche Ansichtsstruktur eine Matrix von $x/y/z$- Koordinaten darstellt, sollte diese aus Gründen

der Beibehaltung der Nachbarschaftsbeziehungen erhalten bleiben. Jedoch kann hier die Größe einer solchen Matrix Grenzen zeigen. Bei einer gewöhnlichen Matrix von 640 x 512 Bildpunkten zu je 3 Koordinaten zu 4 Byte ergibt sich eine Matrizengröße von :

$$640 \times 512 \times 3 \times 4 \text{ Byte} = 3.932.160 \text{ Bytes.}$$

Bei gegebenen Randbedingungen die ca. 100 Ansichten für ein zu vermessendes Objekt definieren, erfordert dies einen Speicherbedarf von ca. 400 MB, welcher auch momentan übliche Speichergrenzen sprengt. Aus der Tatsache, daß längst nicht alle Bildpunkte einer solchen Matrix gültige Werte beinhalten, läßt sich eine komprimierte Darstellung ableiten : Für jede Bildzeile werden nur die gültigen Spalten verwaltet. Hierdurch bleibt die Zeilennachbarschaft zwar erhalten, die Spaltennachbarschaft kann jedoch Sprünge, die durch fehlende Meßstellen entstehen, beinhalten. Hier sind Algorithmen zu implementieren, die aufgrund von Nearest-Neighbour-Bedingungen diejenigen Punkte finden, die als Nachbarpunkte in Zeilenrichtung dienen. Während zwischen den Datenpunkten einer Ansicht die Nachbarschaftsbedingungen bekannt sind, ist die Nachbarschaft der Ansichten untereinander noch zu ermitteln. Ein Nachbarschaftstest kann z.B. durch Vergleich der Boundingboxen der Ansichten erfolgen. Ein Schnitt dieser Boundings deutet auf überlappende Bereiche und somit auf Nachbarschaft der Ansichten hin. Ein trivialer Bounding-Schnittest kann jedoch auch zu fehlerhaften Ergebnissen führen : Da Boundingboxen lediglich räumliche Näherungen der enthaltenen Daten darstellen, kann der Fall auftreten, daß sich Boundingboxen zwar schneiden, ihre enthaltenen Daten jedoch nicht. Aus diesem Grunde müßte ein Näherungsverfahren implementiert werden, das die Beschreibung der Oberflächendaten genauer untersucht. Jede neu in die Struktur einzufügende Ansicht müßte auf Nachbarschaft zu den bereits vorhandenen Ansichten getestet werden.
Die Realisierung einer solchen Datenstruktur mit Verweisen einer Ansicht auf ihre Nachbaransichten führt schließlich auf einen Graphen. Eine mögliche Implementierung könnte pro Ansicht eine dynamische Liste von Verweiszeigern auf Nachbaransichten halten.
Während diese Struktur zwar die Erhaltung der Nachbarschaftsbeziehungen einzelner Punkte einer Ansicht untereinander unterstützt, ist die gerichtete Suche von Nachbarpunkten in äußeren Regionen, d.h. in Nachbaransichten nicht mehr deutlich gegeben. So kann keine eindeutige Richtung auf eine Nachbaransicht ermittelt werden, da sich diese auch ganz oder nur teilweise in beliebigen Bereichen überlappen können. Sollte für dieses Problem keine Lösung gefunden werden, so müßten Nachbarschaftssuchen von Punkten auch auf alle Nachbaransichten ausgeweitet werden bis ein sicheres Endekriterium, z.B. außerhalb einer Region, erreicht ist. Im ungünstigsten Fall ergibt sich ein Suchraum über alle Ansichten.
Durch weit überlappende Bereiche der benachbarten Ansichten sind bei algorithmischer Betrachtung solcher Regionen sowohl die Punkte einer Ansicht 1 als auch die Punkte einer Ansicht 2 zu untersuchen. Dieses Problem könnte z.B. durch Gridding aller Punkte (Projektion auf Parameterlinien) einer wählbaren Region gelöst werden. Der Algorithmus könnte dann auf einer Schnittmatrix arbeiten, die dem Ansichts-Schnittbereich entspricht. Die vorliegende Gridmatrix könnte auch allgemein mathematische Algorithmen vereinfachen, die meist auf einer Punktematrix bzw. Parameterlinienebenen arbeiten.
Ein Vorteil dieser Datenstruktur ist die Beibehaltung der Oberflächeninformation durch direkte Wei-

terverwendung der Ansichten. Auf diese Weise können im Datenmodell keine Hinterschneidungen entstehen. Ein weiterer Vorteil liegt in der direkten Bekanntschaft der Nachbarschaftsbeziehungen von Punkten einer Ansicht, welcher allerdings unter Berücksichtigung der Nachbaransichten schnell zur Problemen führen kann.

3.3 Die Quader - Octree Datenstruktur

Der Vergleich der in der Literatur behandelten und hier kurz vorgestellten Datenstrukturen führt unter Berücksichtigung der geforderten Eigenschaften und gegebenen Randbedingungen eindeutig zu regionenbasierten Datenstrukturen. Punktbasierte Datenstrukturen sind an dieser Stelle aufgrund der anfallenden Datenmengen weniger geeignet. Aufgrund der unzureichenden Unterstützung von Suchoperationen auf der Matrizengraph-Struktur sollte auch diese nicht mehr weiter diskutiert werden. Aus dieser Einschränkung folgt nun der Vergleich der Strukturen Fixed-Grid und Region-Octree . Beides sind regionenbasierte Strukturen, wobei die Variante Region-Octree einen hierarchischen Aufbau vorweist. Die Fixed-Grid -Datenstruktur kann Operationen zur Suche, speziell der Nachbarschaftssuche, einfach realisieren. Zudem ist ihre Speicherverwaltung im Vergleich zu hierarchischen Datenstrukturen effizienter und einfacher zu implementieren. Der Name der Struktur selbst deutet jedoch auf einen Nachteil hin : Alle Raumelemente müssen dasselbe konstante Volumen besitzen. Diese Eigenschaft der Struktur führt einerseits zu vereinfachten Algorithmen, erschwert andererseits jedoch die Flexibilität von Anwendungen, welche nicht-volumenkonstante Raumelemente benötigen. Zu diesen Anwendungen zählen z.B. Schnitt und Triangulierungsverfahren, welche durch die zu realisierende Datenstruktur nach Möglichkeit berücksichtigt werden sollten. Diese Verfahren benötigen ein variables Volumenmodell eines Körpers um Raumelemente z.B. nach Gesichtspunkten der Ebenen-Fit-Faktoren zu erzeugen. Punktemengen mit erhöhten Krümmungsbereichen würden mit diesen Verfahren in kleinere Volumenelemente unterteilt werden, bis ein befriedigendes Kriterium vorliegt. Punktebereiche, die eine Ebene darstellen, sollten zusammenhängend in einem Volumenelementen gezeigt werden.
Die bestmögliche Erfüllung der genannten Probleme kann durch die Region-Octree-Datenstruktur erreicht werden. Diese erlaubt dynamische und nicht-volumenkonstante Raumelemente. Ihre hierarchische Struktur erlaubt zusätzlich die Implementierung effizienter Suchalgorithmen, welche letztendlich zu schnelleren Selektionsoperationen auch auf umfangreichen Punktmengen führen.
Mit der Entwicklung der **SPK** - Bibliothek (**S**tructured **P**oint **K**ernel) [JUN 95B] von Werner Jungbauer entstand eine Datenstruktur, die den im ersten Abschnitt gestellten Anforderungen gerecht wird. Vor allem die durch diese Datenstruktur mögliche räumliche Trennung von Datengebieten und deren erhöhte Flexibilität durch das Octree-Verfahren bieten Vorteile, die mit konventionellen Datenverwaltungsmethoden nicht realisierbar sind. Diese Vorteile sind bei der Implementierung von Anwendungen auf dieser Struktur jedoch durch eine erhöhte Komplexität zu erkaufen. Die Zerlegung der Punktwolke in Quader erfolgt aus implementierungstechnische Gründen immer parallel zu den Hauptachsen (X-, Y- und Z-Achse), die bei optischen Messungen meistens durch das interne Koordinaten System des Sensors vorgegeben sind. Für eine geschicktere Zerlegung der Punktwolke sollte vorher ein geeignetes Objektkoordinatensystem gewählt werden.

Realisierung der Quader - Octree Struktur

- Zerlegung der gesamten Punktwolke in (achsenparallele) Quader, die jeweils aus ca. 100 - 2.000 Punkten bestehen sollten.
- Unterscheidung von
  - Punkten (einzelne Meßpunkte),
  - Punktlisten (Meßpunkte einer Ansicht pro Quader),
  - Punktcluster, das heißt einer Liste von Punktlisten, die alle Meßpunkte in einem Quader beinhaltet
- Verwaltung der einzelnen Quader in einer Baumstruktur, die eine implizite Nachbarschafts Beziehung für die einzelnen Quader bereitstellt (Region Octree).
- Ausgehend von der Bounding-Box der gesamten Punktwolke (minimaler achsenparalleler Quader der betrachteten Punktwolke), wird jede Bounding Box in jeweils acht gleichgroße Quader rekursiv zerlegt bis eines der unten beschriebenen Abbruchkriterien erreicht ist.
- Knoten des Baums, denen ein Quader ohne Punkte zugeordnet ist, werden als Endknoten betrachtet und nicht weiter zerlegt.
- Abbruchkriterium für die Baumerzeugung ist eine festzulegende Baumtiefe b. Bei einer Baumtiefe b gibt es maximal 8^b Quader (für alle folgenden Beispiele gilt für b immer : $0 \leq b \leq 8$).
  Bemerkung : Im Kapitel 4.3 wird für den Topologie Triangulierungsalgorithmus gezeigt, wie man zu einer für alle Quader vorgegebenen Kantenlänge q die Bounding-Box und die Baumtiefe berechnet. Die Ergebnisse sind in Abb. 3.7 links dargestellt.
- Automatische Ansätze zur Quader - Strukturerzeugung über die maximale und minimale Anzahl von Punkten pro Quader oder der Standard Abweichung der Punkte eines Quaders von Ihrer Ausgleichsebene sind ebenfalls möglich.

Beispiele zur Wirkungsweise (unter anderem der Parametrisierbarkeit) der Quader - Octree Struktur auf verschiedenen Punktwolken folgen in den nächsten Abbildungen. Bei der Vorstellung des Algoritmus zur Topologie - Triangulierung von Punktwolken im nächsten Kapitel werden ebenfalls noch einige Beispiele zur Zerlegung von Punktwolken innerhalb der Quader - Octree Struktur aufgeführt.

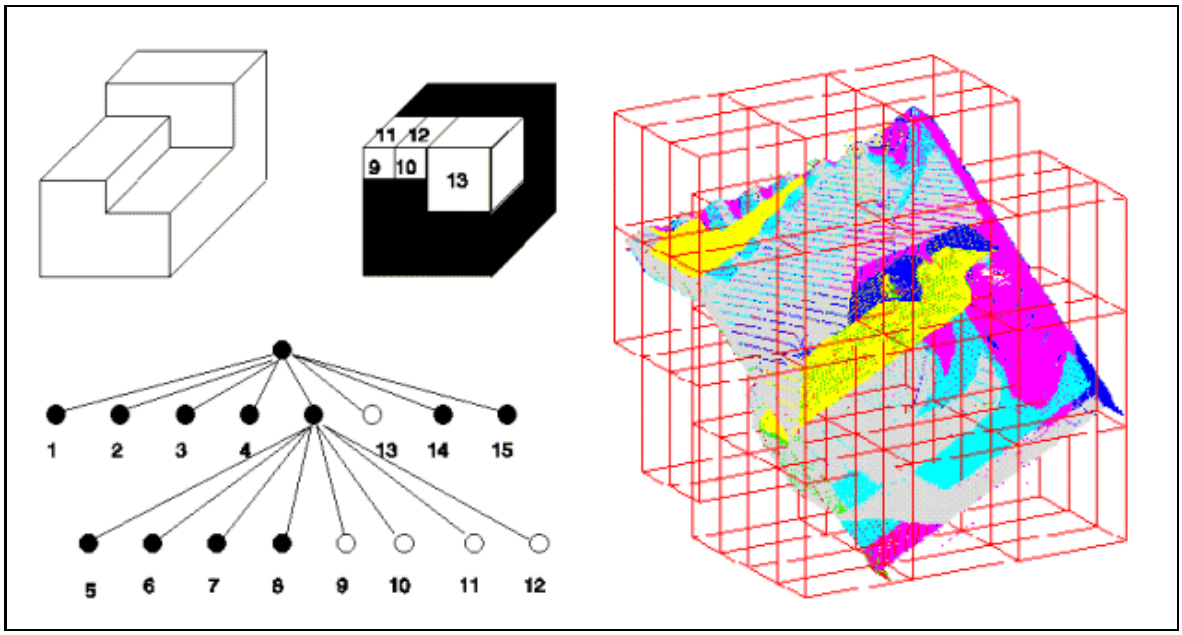


Abbildung 3.5 : Hierarchie Struktur und Punktwolken Zerlegung in Quader-Octree Struktur

Bild links : *Skizze zum Hierarchiegraph mit Knoten und Blättern eines Region - Octrees*

Bild rechts : *Zerlegung der in 7 Ansichten gemessenen Punktwolke des IMS-Teils in Quader der Baumtiefe 3. Die Quader sind rot eingezeichnet. Die Meßpunkte der unterschiedlichen Ansichten sind verschieden farbig dargestellt.*

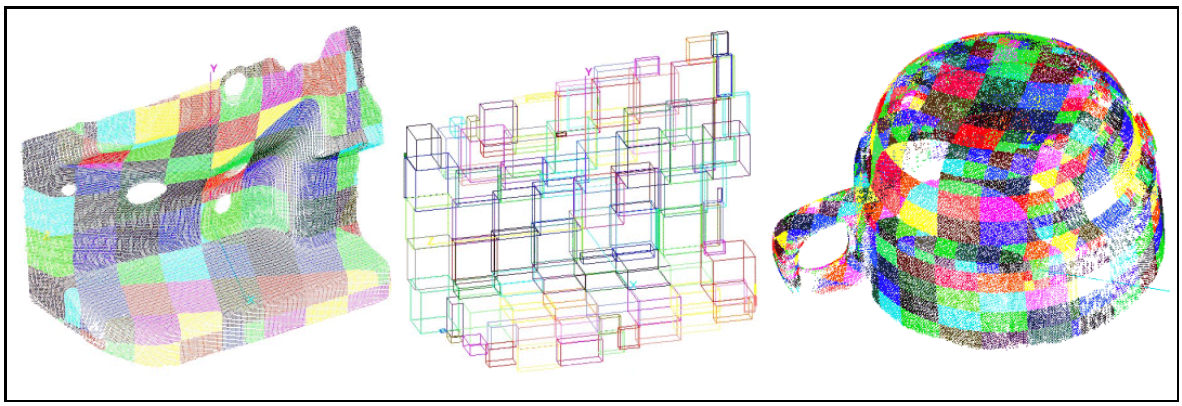


Abbildung 3.6 : Dynamische Zerlegung der Punktwolken Blechteil und Tasse

Bild links : *stellt die in Quader zerlegte Punktwolke (95.746 Punkte) des Blechteiles dar. Die Punkte wurden nach ihrer Quaderzugehörigkeit eingefärbt.*

Bild mitte : *zeigt die Bounding Boxen der insgesamt 114 Quader (Baumtiefe $b = 4$).*

Bild rechts : *In diesem Beispiel wurde die aus fünf Ansichten bestehende Meßpunktwolke einer Kaffeetasse (ca. 115.000 Punkte) dynamisch in Quader zerlegt. In den stärker gekrümmtem Bereichen (Henkel und Übergang zum Tassenboden) wurden kleinere Quader erzeugt, als in den schwach gekrümmten Bereichen.*

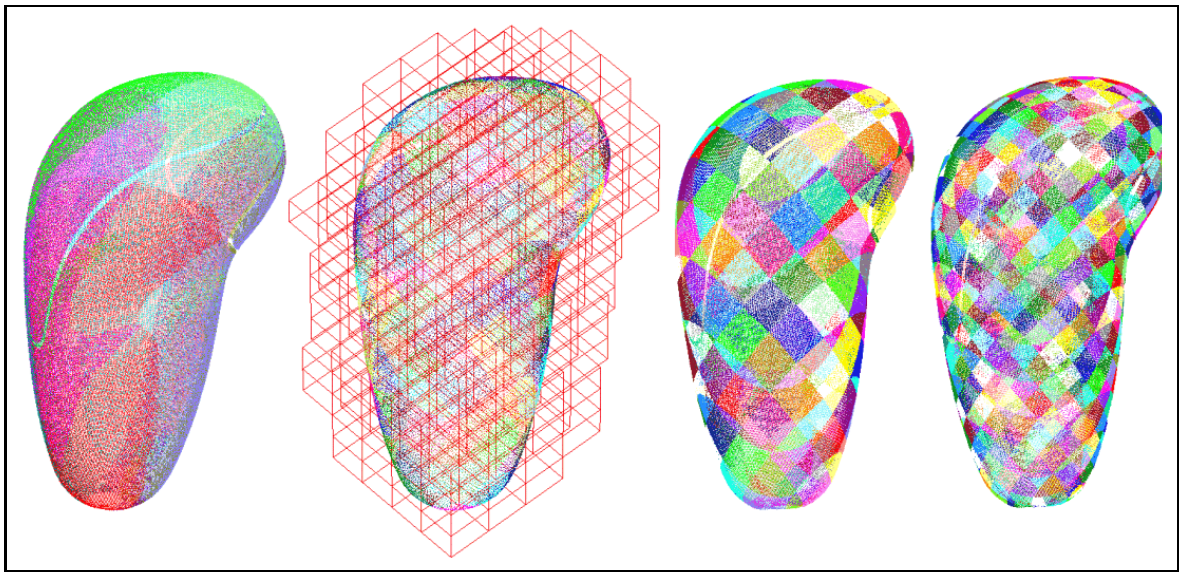


Abbildung 3.7 : Beispiel zur Strukturierung einer Punktwolke

Bild links : *Die 3D Punktwolke einer optischen Messung eines Schalthebel Knaufs (203.771 Punkte in 10 Ansichten gemessen). Die Einfärbung der Punktwolke erfolgt nach der Zugehörigkeit zu einer bestimmten Meßansicht.*

Bild mitte links : *Die Punktwolke wurde mit Ihrer ursprünglichen Bounding-Box in Quader zur Baumtiefe $b = 3$ zerlegt. Die Bounding-Boxen der einzelnen Quader sind als rote Kanten eingezeichnet.*

Bild mitte rechts : *Die gleiche Punktwolke wurde in Quader mit einer gleichmäßigen Kantenlänge von 10 mm zerlegt. Die Einfärbung der Punkte spiegelt die Quader-Octree Datenstruktur wieder (Mosaik Darstellung).*

Bild rechts : *Diesmal wurde die Punktwolke in Quader mit einer Kantenlänge von 5 mm zerlegt.*

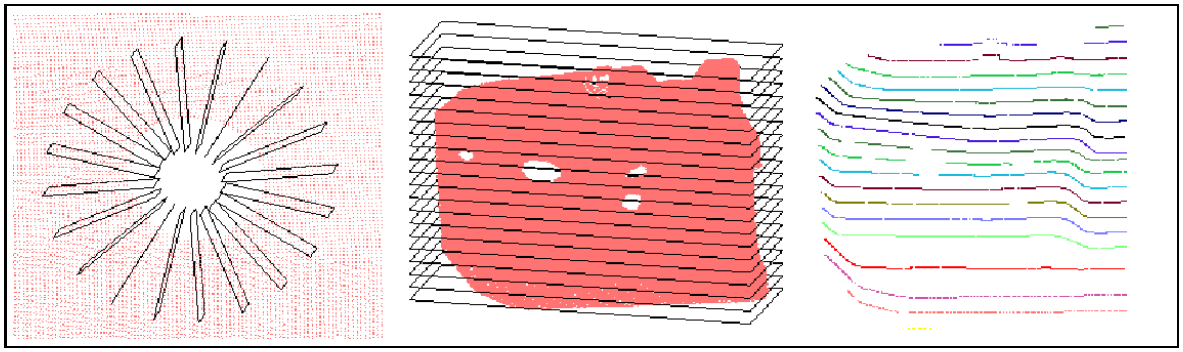


Abbildung 3.8 : Beispiele zur Berechnung von Schnitten (Cross - Sections) durch Punktwolken

Bild links : *Schnittebene für kreisförmige angeordnete Schnitte ausgehend von einem Bohrloch Mittelpunkt.*

Bild mitte : *Schar von parallelen Schnittebenen*

Bild rechts : *Sortierte Polygonzüge für diese parallele Schnitte*

4 Triangulierung von 3D - Punktwolken

Überblick

Es gibt verschiedene Möglichkeiten zur Triangulierung von 3D - Punktwolken. Diese unterscheiden sich sehr stark in der Art der algorithmischen Berechnung, den Anforderungen an die zu verarbeitenden Punktmengen (Strukturierung) und auch in der Art der Ergebnisse (Dreiecke auf der Oberfläche, Tetraeder oder Konvexe Hülle der Punktwolke) . Aus diesem Grund werden im folgenden Abschnitt die verschiedenen Verfahren zur Triangulierung von 3D - Punktwolken kurz vorgestellt und deren Einsatzmöglichkeiten zur Flächenrekonstruktion untersucht. Es wird kurz auf die Vor- und Nachteile der Methode eingegangen. In den nächsten beiden Abschnitten werden zwei neue Verfahren zur Triangulierung von Punktwolken erläutert, die für unstrukturierte Punktwolken beliebiger Größe eingesetzt werden können. Während der Algorithmus zur vollständigen Triangulierung [Kap. 4.2] sehr schnell arbeitet, aber die erzeugten Triangulierungen für einige Anwendungen manchmal interaktiv nachbearbeitet werden müssen, läuft das Verfahren der Topologie Triangulierung [Kap. 4.3] vollautomatisch ab. Bei einer hohen Auflösung hat es jedoch eine relativ lange Laufzeit. Mit Auflösung wird die mittlere Kantenlänge der entstehenden Dreiecke bezeichnet.
Am Ende dieses Kapitels befindet sich ein kurzer tabellarischer Vergleich aller aufgeführter Verfahren mit deren Stärken, Einschränkungen und Schwächen.
In der Arbeit von Robert Mencl und Heinrich Müller [MeMue 97] findet sich eine ähnliche Übersicht über die verschiedenen Strategien zur Triangulierung von unstrukturierten Punktwolken. Jedoch fehlt hier eine Bewertung der Verfahren nach Laufzeit und Stabilität.

4.1 Verschiedene Verfahren zur Triangulierung von 3D - Punktwolken

4.1.1 Räumliche Delaunay - Triangulierung

Eine Möglichkeit zur Triangulierung von 3D - Punktwolken ist die Delaunay - Triangulierung im $\mathbb{R}^3$. Hierbei werden jeweils vier Punkte der gegebenen Punktwolke zu einem Tetraeder zusammengefaßt. Die Bestimmung der Tetraeder erfolgt nach dem folgenden Kriterium :

Innerhalb der Kugel, die durch die vier Eckpunkte eines nicht entarteten Tetraeders eindeutig definiert ist, darf kein anderer Punkt der Punktwolke liegen.

Das Ergebnis dieser 3D - Delaunay - Triangulierung ist eine Menge von Tetraedern, die die konvexe Hülle der gegebenen 3D - Punktwolke definieren. Da dieses Verfahren aber nur die konvexe Hülle einer Punktwolke bestimmt, ist es auch nur für die Rekonstruktion der Oberfläche eines konvexen 3D Körpers geeignet.

Ein räumlicher Körper heißt konvex, wenn zu je zwei beliebigen Punkten, die auf der Oberfläche des Körpers liegen auch die Verbindungslinie dieser zwei Punkte vollständig innerhalb oder auf dem Rand des Körpers liegt.

Diese Bedingung der Konvexität erfüllen aber nur relativ einfache Körper wie zum Beispiel Kugeln, Zylinder, Quader oder auch Pyramiden. Die Oberflächen-Rekonstruktion einer gemessenen Punktwolke von einem konvexen Körper kann dann als Menge aller Randdreiecke der 3D - Delaunay Triangulierung dieser Punkte bestimmt werden. Bei den Randdreiecken handelt es sich um alle Dreiecke der Triangulierung, die sich an der Außenseite eines äußeren Tetraeders der Triangulierung befinden.
Für die Flächenrekonstruktion von allgemeineren 3D - Objekten ist dieses Triangulierungsverfahren nicht geeignet, da nicht die intuitiv erwartete Oberfläche erzeugt wird, sondern die konvexe Hülle der räumlichen Punktwolke des Objekts.
Es gibt viele Arbeiten in denen Verfahren und Algorithmen vorgestellt werden, welche die konvexe Hülle einer Punktmenge in beliebigen Dimensionen berechnen (vgl. [Bow 81], [PreSh 85], [Barb 95], [BaDoH 95], [Clark 95], [Fac 95]) . Die in diesen Arbeiten vorgestellten Algorithmen unterscheiden sich meistens in der Art des Hinzufügens der einzelnen Datenpunkte in die Triangulierung. Bei der Methode des iterativen Hinzufügen werden ausgehend von einem Tetraeder, der die Punktwolke umschließt, die Punkte nacheinander in die Triangulierung integriert, so daß in jedem Schritt eine 3D - Delaunay - Triangulierung, der bisher eingefügten Punkte erzeugt wird. Es sind jedoch bei jedem Einfügen eines neuen Punktes in die Triangulierung, alle Tetraeder der bisherigen Triangulierung zu bestimmen, in deren umgebender Kugel der einzufügende Punkt liegt. Dieses ist ein erheblicher Suchaufwand, der im schlimmsten Fall zu einem Algorithmus mit quadratischer Laufzeit in der Anzahl der Tetraeder führen kann. Bei Algorithmen, die nach dieser Methode die konvexen Hülle erzeugen, kann durch ein geschicktes Vorsortieren der Punktwolke (zum Beispiel um den Schwerpunkt der Punktwolke) ein besseres Laufzeitverhalten erreicht werden. Jedoch erhält man bei N gegebenen Datenpunkten ca. $4N$ Tetraeder und der Algorithmus hat immer noch eine durchschnittliche Komplexität von $O(N \log N)$, was bei einer Menge von mehr als zehntausend Punkten, eine inakzeptablen Laufzeit von mehreren Stunden zur Folge hat [Bow 81], [PreSh 85] .
Die Algorithmen von B. Joe [Joe 91] und von C.B. Barber [Barb 95], [BaDoH 95] für die dreidimensionale Triangulierung beruhen auf dem Prinzip der schrittweisen Konstruktion der konvexen Hülle der gegebenen Punktwolke. Es werden aber nur die Tetraeder berücksichtigt die Dreiecke besitzen, die zur konvexen Hülle der bisher eingefügten Punkte aus der gegebenen Punktwolke gehören. Es handelt sich also bei diesen Algorithmen nicht um eine 3D - Delaunay - Triangulierung des kompletten Datensatzes.

4.1.2 Alpha Shapes

Ein anderes Verfahren zur 3D - Flächenrekonstruktion, das auf der 3D - Delaunay Triangulierung aufbaut, ist das sogenannte **Alpha - Shape** Verfahren. Es wurde von Herbert Edelsbrunner und Ernst P. Muecke (vgl. [EdMu 94]) erstmals vorgestellt. Hierbei werden alle Dreiecke der Tetraederseitenflächen aus der 3D - Delaunay Triangulierung der Punktwolke, die mit den Verfahren von [Fac 95] erzeugt, wurde nach folgendem Prinzip untersucht :
Es werden die Seitenflächen aller Tetraeder nachträglich überprüft, ob der Radius des Kreises, auf dem die drei Eckpunkte jedes Dreiecks liegen, größer ist als ein vorzugebender Parameter α. Dieser Parameter α kann wie folgt gedeutet werden :

Man stelle sich vor, daß die gegebenen Punkte der 3D Punktwolke im Raum als sehr kleine Stahlkugeln an ihren Koordinaten positioniert werden. Der Rest des 3D - Raums ist mit einem weichem Material ausgefüllt. Wenn man nun versucht dieses weiche Material mit einem Kugelfräser mit Radius α aus allen möglichen Richtungen zu fräsen, und man weiterhin voraussetzt, daß der Fräskopf nicht durch die kleinen Stahlkugel hindurchfräsen kann, so erhält man die sogenannte Alpha Shape der 3D - Punktwolke. Das sind alle Dreiecksflächen der Tetraeder einer 3D - Delaunay Triangulierung, deren umgebender Kreis einen Radius kleiner als den Wert von α besitzen. Für einen unendlich großen Wert von $\alpha = +\infty$ erhält man so auch die Randflächen der konvexen Hülle der Punktwolke. Für $\alpha = 0$, gibt es in der Alpha Shape keine Dreiecksflächen, daß heißt die Alpha Shape einer Punktwolke ist dann gerade die Punktwolke selbst. Es gibt nur endlich viele verschiedene Alpha Shapes einer endlichen Punktwolke, da es nur endlich viele Randflächen der 3D - Delaunay Triangulierung gibt, deren verschiedenen Umkreis Radien die positive reelle Achse in endlich viele Intervalle einteilen.
Die Wahl des Parameters α beeinflußt sehr stark die Ergebnisse der Oberflächenrekonstruktion der Punktwolke. Bei zu großem Parameter α kann es in den stärker gekrümmten Bereichen der Puntkwolke und an Kanten zu einigen unerwünschten Ausrundungen kommen. Beim Absenken des Parameters α kann es in den ebenen Bereichen, in denen die Abtastdichte der Punkte relativ groß ist, zu einigen Löchern in der Triangulierung kommen.

Das Verfahren der **Alpha Shapes** Triangulierung, ist in der vorhandenen Form nicht zur Flächenrekonstruktion von Punktwolken allgemeiner 3D - Objekte anwendbar. Für einen Einsatz des Alpha - Shapes Verfahrens zur Flächenrekonstruktion ist es erforderlich, daß die zugrunde liegenden Punktwolken sehr gleichmäßige Punktabstände aufweisen. Diese Einschränkung und die ebenfalls sehr stark auftretenden Ausrundungen an Kanten oder anderen stark gekrümmten Bereichen und der sehr hohe Laufzeit und Speicherbedarf für das Erzeugen der 3D - Delaunay - Triangulierung, schränken die Möglichkeiten des Alpha - Shapes Verfahrens zur Flächenrekonstruktion stark ein. Zwar ist eine automatische Anpassung des Parameters α an das Krümmungsverhalten der Punktwolke denkbar, doch ist bisher noch kein Verfahren zur 3D - Delaunay Triangulierung bekannt, das ein akzeptables Laufzeitverhalten (für die Triangulierung von ca. 1.0 Mio Meßpunkten sollten maximal 5 Minuten benötigt werden) besitzt.

4.1.3 Räumliche Triangulierung verschiedener Ansichtsbilder

Die Triangulierung kompletter räumlicher Objekte erfolgt unter zur Hilfenahme der Ansichtsinformationen, die man von den verschiedenen Ansichtsbildern (Range Images) zueinander bestimmen kann [Karb 94], [Roth 96], [SouLa 94] und [TuLe 94]. Es werden jeweils die Punkte einer Ansicht, die eineindeutig in ihre Bildebene (ein diskretes $n \times m$ Rastergitter) abbildbar sind, in der Bildebene trianguliert, wobei die Nachbarschaftsbeziehungen aus dem rechteckigen Gitter zur schnellen Triangulierung der Punkte ausgenutzt werden. Die Triangulierung wird dann mit der vom Meßystem bekannten Transformationsmatrix für die entsprechende Ansicht in den dreidimensionalen Raum transformiert. Sehr große Dreiecke, die in den Bereichen auftreten können, in denen in der Bildebene benachbarte Punkte im Raum sehr weit auseinander liegen (Überschattungen), werden aus der

Triangulierung entfernt. Danach werden die Überlappungs- und Schnittbereiche der Triangulierungen der einzelnen Ansichten bestimmt. Überlappende Dreiecke werden entfernt, Lücken zwischen den Triangulierungen geschlossen.
Mit diesem Verfahren können nur Punktwolken trianguliert werden, die aus mehreren Ansichtsbildern zusammengesetzt wurden. Für Punktwolken, für die nur die räumlichen Koordinaten der einzelnen Meßpunkte zur Verfügung stehen, und somit unabhängig vom zugrundeliegenden Meßsystem sind, kann dieses Verfahren nicht eingesetzt werden.
Das Bestimmen der Überlappungsbereiche der Triangulierungen verschiedener Ansichten und das Schließen der Lücken ist sehr kompliziert und zeitaufwendig. Die Rechenzeit für die Rekonstruktion eines in zehn Ansichten vermessenen Telefons mit ca. 160.000 Dreiecken wird in [TuLe 94] mit mehreren Stunden, bei ca. fünf Minuten interaktiven Eingaben, angegeben.

4.1.4 Triangulierung von Punktwolken in Scan - Linien Form

Punktwolken in Scan-Linien Form [Wil 94, Kap. 4] können relativ einfach trianguliert werden, da die Nachbarschaften zwischen den einzelnen Scan - Linien sowie den Punkten auf den Scan - Linien bekannt sind. Der Triangulierungsalgorithmus arbeitet sequentiell, es wird jeweils eine Triangulierung von zwei benachbarten Scan-Linien erzeugt und in eine gemeinsame Datenstruktur eingefügt. Falls eine genaue Flächenrekonstruktion an Kanten gewünscht ist, kommt es darauf an, daß die Abtastbahnen quer zu diesen Kanten verlaufen.
Dieser Triangulationsalgorithmus kann nur für Punktwolken angewendet werden, die im Scan - Linien Format vorliegen. Das sind dann Punktwolken, welche von Meßsystemen stammen, die auf einer linienförmigen Abtastung der Objekte beruhen. Dieses können zum Beispiel Laserscanner sein, bei denen ein Laser linienförmig über das Meßobjekt bewegt wird und Punkte aufnimmt oder auch im beschränktem Maße Koordinaten - Meßmaschinen, bei denen einen Taststift in parallelen Bahnen über das Objekt fährt.

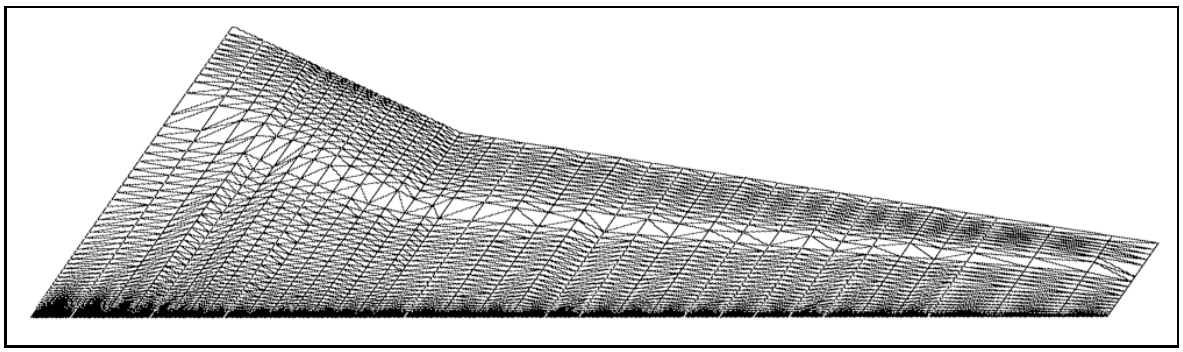


Abbildung 4.1 : Triangulierung auf Scan-Linien - Schnitte durch Flugzeug Flügel

Eine Erweiterung dieses Verfahrens auf beliebige Punktwolken ist über den folgenden mehrstufigen Prozeß möglich :

Zunächst werden durch die ungeordnete Punktwolke parallel Schnitte nach dem im Abbildung 3.8 angedeuteten Cross - Section Algorithmus gelegt. Nach dem Sortieren kann man die Schnitte in eine Scan - Linien Form überführen. Auf diesem Teil der ungeordneten Punktwolke ist dann eine Triangulierung nach obigem Prinzip möglich. Problematisch bei diesem Verfahren ist, daß sich allgemeine 3D - Objekte nicht hinreichend genau durch nur eine Schar von parallelen Schnitten annähern lassen.

4.1.5 Voxelbasierte Konstruktion der räumlichen Triangulierung

In den Arbeiten von H. Hoppe et al. [HoDeR 92], [Hoppe 94] und Roth [Roth 97] ist ein Verfahren zur Triangulierung von Punktwolken beschrieben, die auf dem sogenannten Marching Cubes Algorithmus von Lorensen und Cline [LoCli 87] sowie Wyvill et al. [Wyv 86] beruhen. Der Marching Cubes Algorithmus wurde ursprünglich zur Oberflächenrekonstruktion von medizinischen Computer - Tomographie Volumen Daten (Voxels) eingesetzt. H. Hoppe hat diesen Algorithmus zuerst auf 3D - Punktwolken erweitert, G. Roth hat diese Erweiterung beschleunigt und im Hinblick auf den Speicherbedarf optimiert.
Bei diesem Verfahren wird das Volumen, in dem die Punktwolke liegen, in gleichgroße kleine Quader zerlegt. Für jeden Punkt der Punktwolke muß ein Normalenvektor vorhanden sein, der auch die richtige Orientierung bezüglich seiner Nachbarpunkte haben muß. Diese Normalenvektoren sind bei unstrukturierten Punktwolken nicht bekannt. Bei Datensätzen die aus Messungen von einzelnen Ansichtsbildern entstanden sind, kann man die Normalen und deren konsistente Orientierung [Roth 96] sehr schnell erzeugen. Bei unstrukturierten Punkwolken müssen die Normalenvektoren und deren konsistente Orientierung in einem vorausgeschalteten Schritt abgeschätzt werden. Dies ist aber mit einem erheblichem Rechenaufwand verbunden [Roth 97], [Hoppe 94]. Danach werden die Punkte samt der Normalenvektoren in die Voxelstruktur eingefügt und es wird zu jedem der acht Eckpunkte des Voxels eine vorzeichenbehaftet Abstandsfunktion bestimmt. Diese wird als vorzeichenbehafteter Abstand des Voxel-Eckpunktes zu der Ebene, die durch den Meßpunkt und den Normalenvektor in diesem Punkt aufgespannt wird, berechnet. Sind alle Punkte eingefügt, werden die kumulierten Abstandsfunktionen der Eckpunkte ausgewertet und eine lokale Triangulierung innerhalb des Voxels bestimmt. In einem Nachbearbeitungsschritt müssen die Triangulierungen der einzelnen Voxel noch miteinander verbunden werden (**Closing Gaps**). Es werden topologisch korrekte Triangulierungen erzeugt, falls die Parameter für den Algorithmus so gewählt wurden, daß in jedem beim Marching Cubes Algorithmus verwendeten Voxel eine Mindestanzahl von Punkten liegt. Die für die enstehende Triangulierung verwendeten Eckpunkte der Dreiecke, sind aber keine Punkte der gegebenen Punktwolke, sondern werden bei der Auswertung der vorzeichenbehafteten Abstandsfunktionen der Eckpunkte auf den Kanten des Voxels synthetisch erzeugt. Es wird vorausgesetzt daß der Punktabstand zwischen dichtbeieinanderliegenden Punkte einigermaßen konstant ist, so daß nicht beliebige Punktwolken bearbeitet werden können. Die Berechnung der Triangulierungen einer großen unstrukturierten Punktwolke, mit mehr als 100.000 Punkten erfordert einen mittelgroßer Zeitaufwand.

4.2 Algorithmus zur vollständigen Triangulierung von 3D - Punktwolken

In diesem Abschnitt wird ein Triangulationsalgorithmus vorgestellt, der in kürzester Zeit eine Triangulierung von sehr großen Punktwolken berechnen kann. Ausgangspunkt für diesen Algorithmus sind die Verfahren zur Delaunay Triangulierung in der Ebene. Diese sind weitverbreitet auf dem Gebiet Netzgenerierung für Anwendungen aus der Finite Elemente Analyse. Es gibt einige Algorithmen [Clark 95] [Shew 95] , die auch sehr große Punktmengen schnell (100.000 Punkte innerhalb von fünf Sekunden) triangulieren können. Diese Algorithmen eignen sich daher sehr gut für eine schnelle Triangulierung von großen Punktwolken. Im folgenden wird gezeigt, wie man die Algorithmen zur ebenen Delaunay Triangulierung für räumliche Punktwolken anwenden kann. Das vorgestellte Triangulierungsverfahren für räumlicher Punktwolken zerlegt die gegebene Punktwolke so in Quader, daß die in jedem Quader vorhandenen Punkte eindeutig in Ihre Ausgleichsebene projizierbar sind. Im letzten Kapitel wurde eine Datenstruktur vorgestellt, mit der die Zerlegung beliebiger Punktwolken in hinterschneidungsfreie Bereiche möglich ist. Mit *eindeutig projizierbar* ist gemeint, daß es eine bijektive Abbildung zwischen den in die Ausgleichsebene projizierten Punkte und dem betrachteten Ausschnitt aus der gegebenen Punktwolke gibt, die gewährleistet, daß Punkte, die im Raum am dichtesten beieinander liegen auch in der Ausgleichsebene am dichtesten beieinander liegen (vgl. Punkte auf Rändern des Zylinders im linkes Bild der Abb.: 4.2).
Das bedeutet, daß die Punkte jedes Quaders eine sogenannte 2,5 D Darstellung besitzen müssen. Als Punktwolken in einer 2,5 D Darstellung bezeichnet man solche, deren einzelnen Punktkoordinaten sich alle in der Form $(x, y, z = f(x, y))^T$ darstellen lassen, wobei $f(x, y)$ eine in x und y stetige Funktion über einem ebenen Gebiet $\mathcal{G}$ sei.

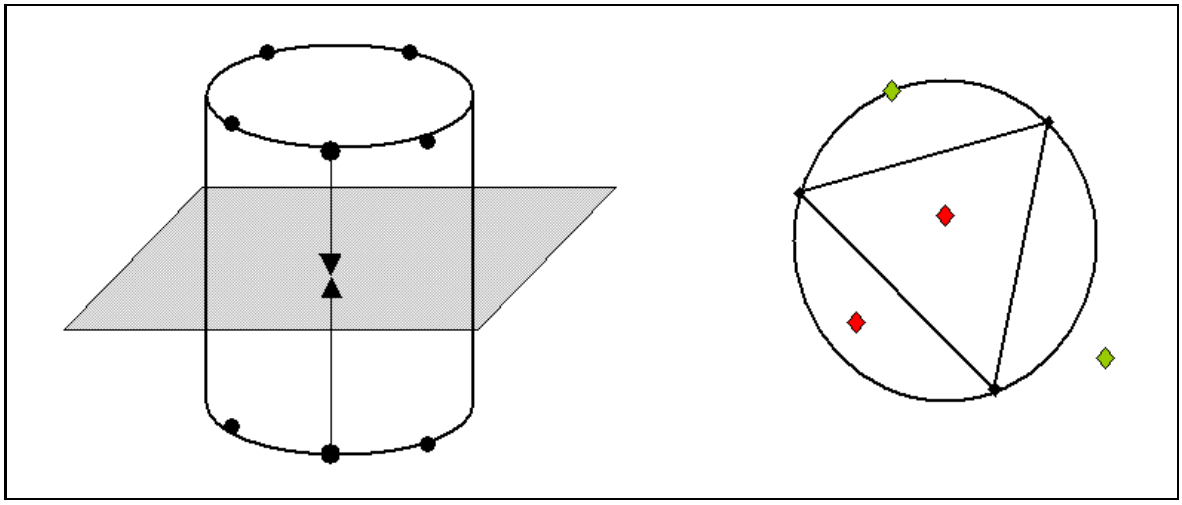


Abbildung 4.2 : Skizzen zur Erläuterung der Delaunay Triangulierung

Bild links : *Die Punkte auf dem Zylinderrand sind nicht eindeutig in ihre Ausgleichsebene projizierbar.*
Bild rechts : *Erläuterung des Delaunay Kriterium - Der durch drei schwarze Punkte verlaufende Kreis, definiert das zugehörige Delaunay Dreieck eindeutig. Die zwei roten Punkte verletzten das Delaunay Kriterium, die zwei grünen Punkte nicht.*

Überblick des hier angewendete Triangulationsverfahren

Räumliche Punktwolken, die von komplexeren 3D - Objekten stammen und Hinterschneidungen besitzen (einfache Beispiele : Punktwolken von vollständigen Kugeln, Zylinder, Würfel oder Tori) werden in kleine Quader zerlegt, in denen dann keine Hinterschneidungen mehr auftreten. Jeder Punkt der gegebenen Punktwolke gehört dann zu genau einem Quader. Für jeden Quader wird für alle Punkte, die in ihm liegen, eine Ausgleichsebene nach dem Verfahren von Bradley und Vickers [BraVi 92] berechnet. Die Punkte jedes Quaders werden dann in die zugehörige Ausgleichsebene projiziert und es wird in der lokalen Ausgleichsebene eine ebene Delaunay - Triangulierung [Bow 81], [Wil 92], [Shew 95] der projizierten Punkte berechnet, die anschließend in den kleinen zugehörigen räumlichen Quader zurückprojiziert wird. Man erhält so mehrere kleine räumliche Triangulierungen, die noch nicht miteinander verbunden sind.
Diese einzelnen Triangulierungen müssen in einem Folgeschritt miteinander verbunden werden (**Stitching)** und im letzten Arbeitsgang müssen die noch vorhandenen kleinen Löcher in der Triangulierung geschlossen werden (**Closing Gaps)** .

4.2.1 Delaunay - Triangulierung ebener Punktwolken

Überblick zu Algorithmen für die ebene Delaunay Triangulierung

Die in [Clark 95] und [Shew 95] vorgestellten schnellen Algorithmen zur ebenen Delaunay Triangulierung von C. Clarkson und J.R. Shewchuck beruhen auf dem sogenannten Prinzip des *Divide and Conquer*. Hierbei wird die gesamte ebene Punktwolke rekursiv in kleine rechteckige Bereiche zerlegt (Divide), in denen dann nur noch zwei bis vier Punkte liegen. Mit diesen wenigen Punkten wird dann eine Kante bzw. ein oder zwei Dreiecke gebildet (Conquer), die dann in einem Nachverarbeitungsschritt zur Triangulierung des gesamten Datensatzes zusammengefügt werden. Die Schwierigkeit bei diesen Verfahren besteht darin, eine geeignete Zerlegung der Punktwolke zu finden, aus der sich dann die gesamte Triangulierung des Datensatzes sehr einfach bestimmen läßt. Da aber immer nur lokal ein nach dem Umkreis Kriterium für Delaunay - Triangulierungen die gesamte Triangulierung aufgebaut wird, ist es nicht immer gewährleistet, daß die gesamte Triangulierung diesem Kriterium genügt. Das ist aber für Verfahren zur Flächenrekonstruktion nicht so wichtig.
Andere Algorithmen zur Delaunay Triangulierung von ebenen Punktmengen, [Bow 81] [PreSh 85] [Dwy 87] die auf dem Prinzip der schrittweisen iterativen Konstruktion der Triangulierung beruhen, haben eine etwas längere Laufzeit als die rekursiven Algorithmen. Bei dem Prinzip der iterativen Konstruktion der Delaunay Triangulierung werden die nach aufsteigendem Punktabstand zum Schwerpunkt aller zu triangulierenden Punkte sortierten Punkte schrittweise in die Dreiecksstruktur eingefügt. Es ist somit gewährleistet, daß nach jedem Einfügen eines Punktes eine korrekte Delaunay Triangulierung für alle bisher eingefügten Punkte erzeugt wurde. Durch das Sortieren aller Punkte nach steigendem Abstand zu ihrem Schwerpunkt, wird erreicht, daß die Anzahl, der in jedem Schritt zu durchsuchenden Dreiecke, relativ klein gehalten werden kann.

Grundlagen bei der Triangulierung von ebenen Punktwolken

Bei der ebenen Triangulierung müssen je drei Punkte der zu triangulierenden Punktmenge zu einem Dreieck zusammengefaßt werden. Die Zuordnung der Datenpunkte zu den Dreiecken sollte so erfolgen, daß die am nächsten zusammenliegenden Punkte jeweils ein Dreieck bilden und innerhalb des Umkreises jedes Dreiecks kein anderer Datenpunkt liegt (vgl. rechtes Bils aus Abb. 4.2) Eine nach diesem Kriterium erzeugte ebene Triangulierung heißt Delaunay - Triangulierung. Die Menge aller so erzeugten Dreiecke ist dann die konvexe Hülle der zu triangulierenden ebenen Punkte.
Es gilt für jedes Delaunay - Dreieck :

Die drei Eckpunkte jedes Delaunay Dreiecks definieren einen Kreis in dem kein anderer Datenpunkt der Punktwolke liegt. Die Delaunay - Triangulierung einer Menge von ebenen Punkten ist bis auf lokale Symmetrien eindeutig [HoLa 92][Kap. 9.3].

Es besteht folgender Zusammenhang zwischen der Delaunay - Triangulierung und der sogenannten Dirichlet - Parkettierung (engl.: tessellation), die auch manchmal als Voronoi - Diagramm bezeichnet wird. Bei der Dirichlet Parkettierung handelt es sich um die Aufteilung der Punktwolke in polygonal begrenzte Gebiete. Es wird jedem Punkt der Punktwolke ein konvexes, durch Polygone begrenztes Gebiet zugeordnet, das alle Punkte der Ebene enthält, die näher an diesem Meßpunkt als an eindem anderen Punkt der Punktwolke liegen. Die Eckpunkte der Gebiete der Dirichlet Parkettierung sind jeweils die Mittelpunkte der Kanten der Dreiecke aus der Delaunay - Triangulierung(vgl Abb. 4.3).

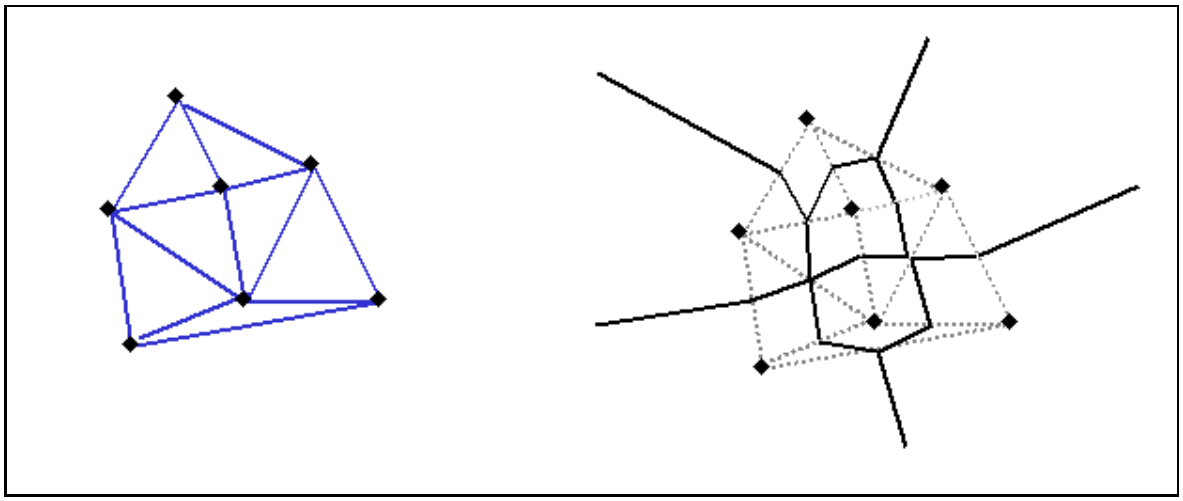


Abbildung 4.3 : Zusammenhang zwischen Delaunay Triangulierung und Dirichlet Parkettierung

Bild links : *zeigt eine Delaunay Triangulierung von sieben Punkten.*

Bild rechts : *stellt die dazugehörige Dirichlet Parkettierung dar.*

Implementiertes ebenes Triangulierungsverfahren

Es seien N Punkte der Ebene als Menge der zu triangulierenden Punkte $\mathcal{Q} := \{Q_0, Q_1, \cdots, Q_{N-1}\}$ gegeben. Zunächst werden der Schwerpunkt $S := \frac{1}{N} \cdot \sum_{i=0}^{N-1} Q_i$ und die minimalen und maximalen x- und y- Werte der Punktmenge $\mathcal{Q}$ bestimmt. Der Datensatz wird dann in seinen Schwerpunkt verschoben. Die minimalen und maximalen x und y Koordinaten der verschobenen Punktwolke werden um 10 Prozent verkleinert bzw. vergrößert. Zusätzlich werden mit diesen skalierten extremalen Koordinaten vier Hilfspunkte $Q_N, \cdots, Q_{N+3}$, bestimmt, die ein die Punktwolke umgebendes Rechteck definieren, daß alle Punkte der Punktwolke umschließt. Dieses erste Rechteck wird in zwei Dreiecke aufgeteilt, die als initiale Triangulierung der Punktwolke betrachtet werden können [Bild (a) aus Abb. 4.4].

Bemerkung : Die Aufteilung des Rechteckes ist beliebig, es können sowohl die zwei Dreiecke $\{Q_N, Q_{N+1}, Q_{N+3}\}$ *und* $\{Q_{N+1}, Q_{N+2}, Q_{N+3}\}$ *als auch die Dreiecke* $\{Q_N, Q_{N+1}, Q_{N+2}\}$ *und* $\{Q_{N+2}, Q_{N+3}, Q_N\}$ *erzeugt werden. Die Wahl des Skalierungsfaktors von 10 Prozent des Rechteckdurchmessers erscheint zunächst etwas willkürlich, wichtig ist die Vergrösserung nur dafür, daß für Punkte, die auf der Kante des nicht vergrösserten Rechtecks liegen, keine besondere Fallunterscheidung gemacht werden muß. Der Wert 10 Prozent hat sich für den nachfolgenden Schritt der Vernetzung als brauchbar erwiesen. Weitere Einflüsse auf den Algorithmus haben diese Festlegungen nicht.*

Die Konstruktion dieser beiden Hilfsdreiecke hat folgenden Zweck :
Bei dem schrittweisen Hinzufügen der Punkte zur Delaunay Struktur liegen nun alle neu eingefügten Dreiecke innerhalb dieses Hilfsrechteckes. Man kann also auf eine Fallunterscheidung für Dreiecke, die außerhalb der aktuell berechneten Delaunay - Struktur liegen, verzichten, was der Übersichtlichkeit des Algorithmus zu Gute kommt.
Wenn alle Punkte in die Delaunay - Triangulierung eingefügt sind, sollen alle Dreiecke, die einen der oben konstruierten Eckpunkte ($Q_N, \cdots, Q_{N+3}$) des Hilfsrechtecks enthalten, aus der Delaunay - Dreiecks - Struktur entfernt werden. (Abb. 4.4 Bild (i)). Die restlichen Dreiecke bilden dann die ebene Delaunay - Triangulierung der gegebenen Punktmenge $\mathcal{Q}$. Die Menge aller so erzeugten Dreiecke ist nicht in jedem Fall die konvexe Hülle der zugrunde liegenden Punktmenge. Die wird jedoch für das Vernetzungsverfahren im folgenden Abschnitt auch nicht benötigt. Wenn man die konvexe Hülle der Punktwolke $\mathcal{Q}$ berechnen will, muß man nur die Punkte des äußeren Hilfsrechteckes mit einem Faktor von mindestens fünfzig Prozent anstelle der verwendeten zehn Prozent skalieren. Die nach dem Entfernen der Dreiecke, die einen der Hilfspunkte besitzen, übriggebleiben bilden dann die konvexe Hülle der gegebenen Punktmenge. Dies gilt, weil jeder beliebigen Punkt der Punktwolke von einem Hilfspunkt weiter entfernt ist, als jeder andere Datenpunkt und damit zu jedem äußeren Punkt der Punktwolke mindestens ein Dreieck existiert, daß diesen Punkt enthält, aber keinen der vier Hilfspunkte. Das bedeutet, daß die maximale Entfernung von zwei Datenpunkten (Durchmesser des die Punktwolke umschließenden Rechtecks) kleiner sein muß, als der Abstand von einem Hilfspunkt zu jedem Randpunkt der Punktwolke.
Der Nachteil dieser Überlegung ist, daß sich die Rechenzeit geringfügig erhöht, da einige Dreiecke mehr erzeugt werden, als für die konvexen Hülle des Datensatzes benötigt werden.

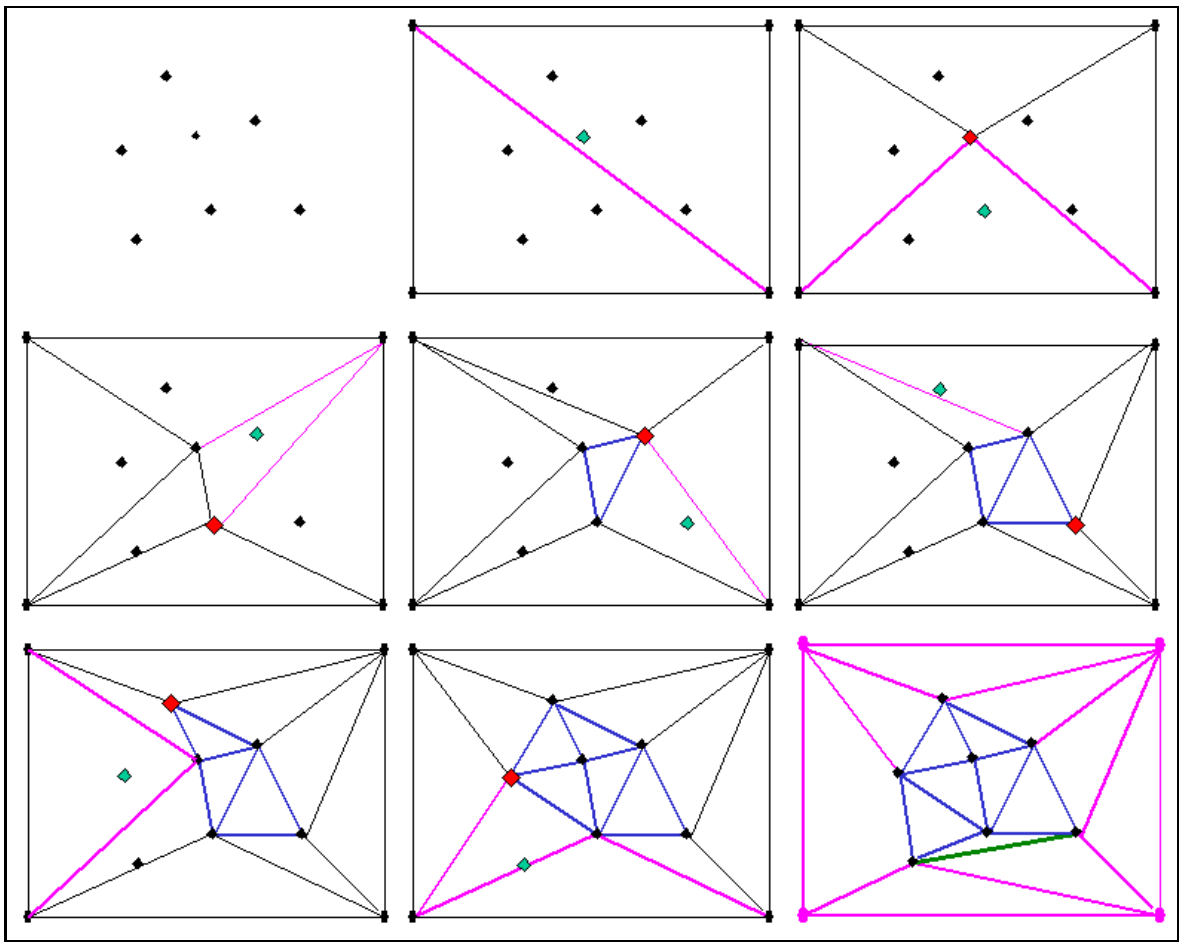


Abbildung 4.4 : Beispiel zum Algorithmus für ebene Delaunay Triangulierung

Bild (a,d,g) : *links* Bild (b,e,h) : *mitte* Bild (c,f,i) : *rechts*

In der Abbildung 4.4 werden die einzelnen Schritte zur Konstruktion einer ebenen Delaunay Triangulierung anhand eines Beispiel von $N = 7$ Punkten erläutert. Bild (a) : *zeigt die gegebenen sieben Punkte (markiert mit einer schwarzen Raute). Die Punkte sind nach aufsteigendem Abstand zu Ihren Schwerpunkt ($S \sim$ Mittelpunkt des Rechtecks) sortiert. Der Punkt, der im nächsten Schritt in die Dreiecksstruktur eingefügt werden soll ist türkis gefärbt.* Bild (b) : *stellt das Rechteck bestehend aus vier Hilfspunkten und zwei Dreiecken dar.* Bild (c) - (i) : *Bei diesen Bildern sind jeweils die rot markierten Punkte in die Dreiecksstruktur eingefügt worden. Durch das Einfügen entstehen in jedem Schritt zwei neue Dreiecke. Bei den violett markierten Kanten handelt es sich um die Kanten, die aus der Dreiecksstruktur entfernt werden müssen, wenn der türkise Punkt eingefügt werden soll.* Bild (i) : *zeigt die Situation nach dem Einfügen aller Punkte in die Dreiecksstruktur. Insgesamt sind $2N+2 = 16$ Dreiecke erzeugt worden. Nun müssen noch die Dreiecke, die einen der Hilfspunkte des Rechtecks enthalten, aus der Struktur entfernt werden. Diese Dreiecke liegen an den violett markierten Kanten. Die grüne Kante gehört zu einem Randdreieck, das einen sehr großen Winkel besitzt ($\geq$ ca. 160 Grad). Dieses Randdreieck wird ebenfalls aus der Dreiecksstruktur gelöscht.*

Der Kern des Algorithmus zur ebenen Delaunay Triangulierung besteht nun aus dem schrittweisen Einfügen der nach aufsteigendem Abstand zum Schwerpunkt S der Punktmenge $\mathcal{Q}$ vorsortierten Datenpunkte Q_i in die Delaunay - Struktur. Beim schrittweisen Einfügen der N vorsortierten Punkte in diese Dreiecksstruktur werden in jedem Schritt zwei neue Dreiecke erzeugt. Ausgehend von den zwei Dreiecken der rechteckigen Hilfsstruktur erhält man somit insgesamt genau $2N + 2$ Dreiecke. Zur Vereinfachung der Schreibweise seien die Punkte der Punktmenge $\mathcal{P}$ nun so bezeichnet, daß ein Punkt mit Index i nicht weiter vom Schwerpunkt S der Punktmenge $\mathcal{P}$ entfernt liegt als ein Punkt mit größerem Index.

$$\forall\, i,\ j\ \in\ \{\, 0, 1, \cdots, N-1 \,\}\ \ : i < j \quad \Longleftrightarrow \quad \|Q_i - S\|\ \leq\ \|Q_j - S\|$$

Diese Bedingung ergibt sich nach dem Vorsortieren des Datensatzes nach aufsteigendem Punktabstand zum Schwerpunkt S automatisch, so daß die Punkte nur noch umbenannt werden müssen.

Schrittweises Einfügen der Punkte in die Delaunay Struktur

Das Einfügen des i-ten Datenpunktes Q_i $(i = 0, 1, ..., n-1)$ erfolgt nach den in Abb. 4.4 (b) - (h) dargestellten Prinzip :

1.) Es wird für jedes Dreieck der Suchliste (im Beispiel : Dreiecke an den schwarz und gelb markierten Kanten) untersucht, ob der neueinzufügende Punkt Q_i innerhalb des Kreises um das Dreieck liegt. Dabei wird der euklidische Abstand des Punktes zum Mittelpunkt mit dem Radius des Kreises verglichen. Falls der Abstand kleiner ist, als der Radius des Kreises, wird das Dreieck in die Liste der zu ändernden Dreiecke aufgenommen. Andernfalls wird das Dreieck als nicht zu ändern markiert. (Aus Gründen der Rechengeschwindigkeit wird jeweils der quadratische Punktabstand bzw. Radius betrachtet.) Wenn ein zu änderndes Dreieck gefunden wurde, wird kontrolliert ob der einzufügende Punkt im Rahmen der Rechengenauigkeit oder eines vorzugebenden minimalen Punktabstandes (**Auflösung**) für zwei Punkte die in einem Dreieck liegen dürfen, zu dicht an einem der Eckpunkte des Dreiecks liegt. Falls dieses der Fall ist, wird der Punkt Q_i nicht in die Delaunay Struktur eingefügt, die Liste der zu durchsuchenden Dreiecke wird nicht geändert und die Funktion zum Einfügen der Punkte in die Dreiecksstruktur wird verlassen. In dem Beispiel der Abbildung 4.4 (b) - (h) liegen die zu löschenden Dreiecke an den gelb markierten Kanten beim Einfügen des türkis markierten Punktes.

2.) Die Liste der zu ändernden Dreiecke wird nach Kanten durchsucht, die mit dem neueinzufügenden Punkt ein neues Dreieck bilden. Um eine solche Kante handelt es sich, wenn sich das zu dieser Kante benachbarte Dreieck nicht in der Liste der zu ändernden Dreiecke befindet. Diese Kante wird zusammen mit dem Index ihres benachbarten Dreiecks in die Liste der Kanten geschrieben, die mit dem neueinzufügenden Punkt ein neues Delaunay - Dreieck bilden. In dem Beispiel der Abbildung 4.4 (c) - (h) sind dies die gelb markierten Kanten.

3.) Berechnung der neuen Dreiecke und der neuen Nachbarschaften

4.) Aktualisierung der Delaunay Struktur und der Suchliste

Kontrolle der Delaunay - Dreiecks - Struktur

Nach dem Einfügen aller Punkte in die Delaunay - Struktur, müssen noch alle Dreiecke, die einen der vier Eckpunkte des Hilfsrechtecks enthalten, aus der Dreiecksstruktur entfernt werden [Abb. 4.4 (h) und (i)]. Danach werden zusätzlich noch Randdreiecke entfernt, bei denen der maximale eingeschlossene Winkel des Dreiecks größer als ein vorgegebener Schwellenwert [Abb. 4.4 (i)] ist. Für diesen Schwellenwert werden ungfähr 160 Grad gewählt. Die Randdreiecke mit großen eingeschlossenen Winkeln können den im folgenden Teilabschnitt beschriebenen Vernetzungsalgorithmus stören.

4.2.2 Verbinden der Triangulierungen der einzelnen Quader

Das Verbinden der einzelnen Triangulierungen, die sich nicht überlappen und bei denen keine Selbstdurchdringungen zwischen verschiedenen Triangulierungen auftreten, zu einer geschlossenen räumlichen Triangulierung, wird als **Vernetzen** (engl. **stitching**) bezeichnet. Im ersten Schritt werden die Randkanten der Triangulierung eines Voxels zusammen mit den Randkanten der Triangulierung der Nachbarvoxel bestimmt [Bild links in Abb. 4.5]. Danach wird mit dem ebenen Triangulationsalgorithmus von J.R Shewchuck [Shew 95], der eine Triangulierung mit Nebenbedingungen erzeugen kann, die Triangulierung zwischen den Voxeln bestimmt [Bild rechts in Abb. 4.5]

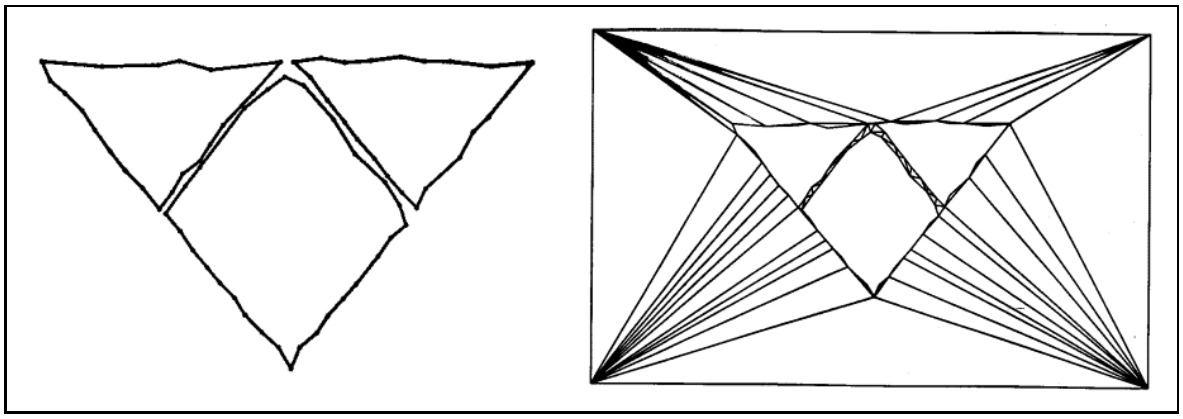


Abbildung 4.5 : Vorgehensweise bei der Vernetzung (Stitching) einzelner Triangulierungen

Bild links : *zeigt die Randkanten der ebenen Delaunay Triangulierung von drei Voxeln.*

Bild rechts : *stellt die im Schritt des Vernetzen zusätzlich erzeugten Dreiecke zwischen den Randkanten dar. Die Dreiecke mit den äußeren vier Hilfspunkten werden nicht in die Gesamttriangulierung übernommen.*

Beispiel für den Algorithmus zur vollständige Triangulierung von 3D - Punktwolken

In den folgenden Bildern wird die Vorgehensweise des Algorithmus zur vollständigen Triangulierung von 3D - Punktwolken anhand eines Beispieles erläutert.

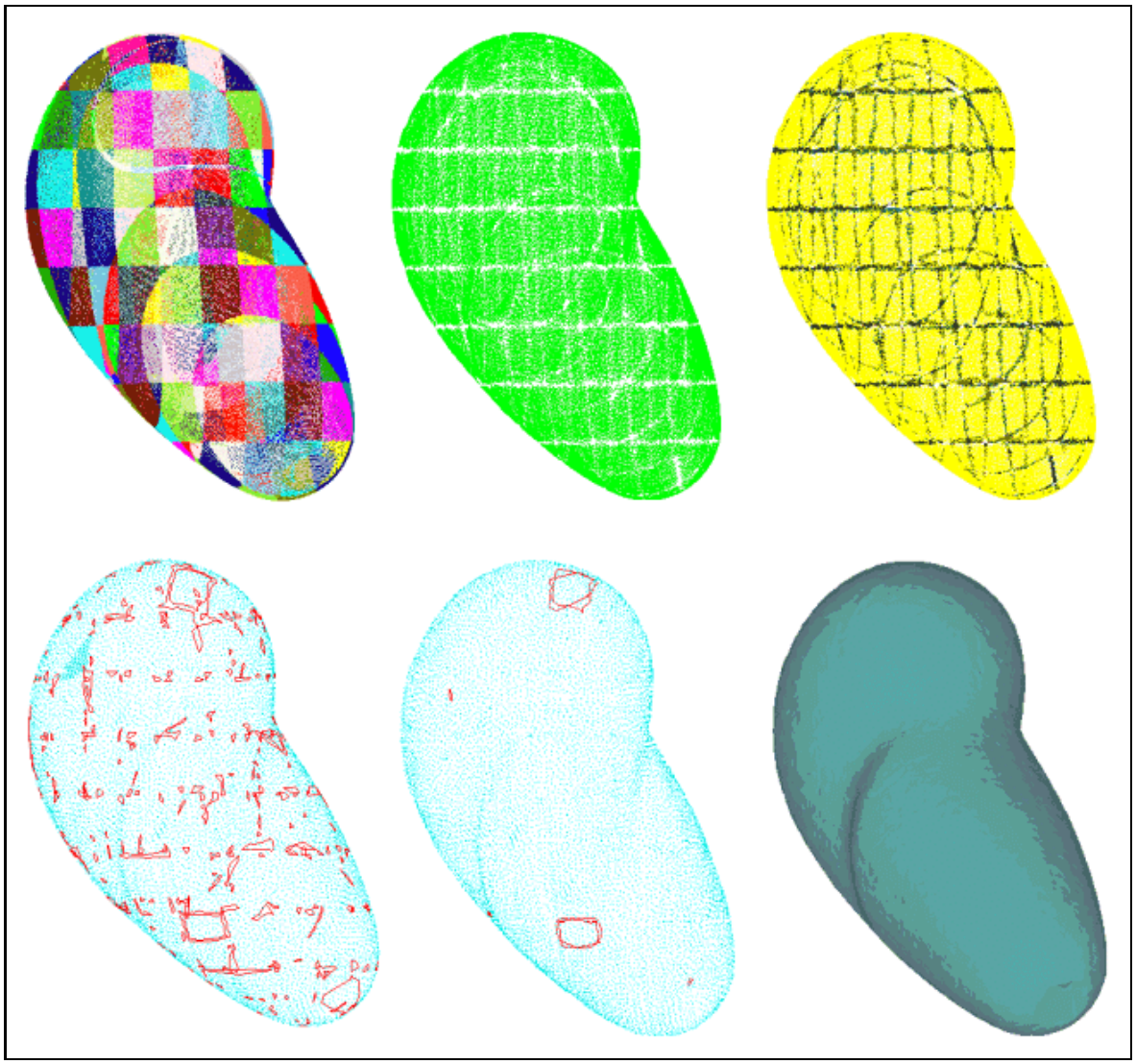


Abbildung 4.6 : Vorgehensweise zur vollständigen Triangulierung von Punktwolken

Bild (a) : *oben links* Bild (b) : *oben mitte* Bild (c) : *oben rechts*
Bild (d) : *unten links* Bild (e) : *unten mitte* Bild (f) : *unten rechts*

Details zur Abbildung 4.6 Beispiel der vollständige Triangulierung

Bild (a) : Der Datensatz **sh-clean.pgk**, bestehend aus 203.771 Meßpunkten eines Schalthebelknaufs, wurde in 228 gleichgroße Quader zerlegt der Kantenlänge 9,3 mm zerlegt. Die Einfärbung der Punkte erfolgte nach deren Quader - Zugehörigkeit. Alle Punkte eines Quaders erhalten die gleiche Farbe aus einer beschränkten Anzahl von Farben, so daß auch Farben mehrfach für nicht benachbarte Quader verwendet wurden.
Bild (b) : Es wurde eine Triangulierung für die einzelnen Punkte der Quader jeweils mit der ebenen Delaunay Triangulierung erzeugt. Dabei wird die Ausgleichsebene durch alle Punkte des Quaders berechnet, die Punkte werden in diese Ausgleichsebene projiziert und es wird jeweils eine ebene Delaunay-Triangulierung berechnet. Als Ergebnis der Triangulierung erhält man 21.668 Dreiecke mit 4.950 offenen Randkanten, die bei dem Parameter 1,0 mm für den minimalen Punktabstand der bei Triangulierung zu verwendeten Punkte erzeugt wurden. Die Dreiecke sind als grünes Netz dargestellt.
Bild (c) : Erzeugung von Verbindungen (Vernetzen, Stitching) zwischen den Triangulierungen der einzelnen Quader. Die inneren Triangulierungen der Voxel sind als gelbes Netz gezeichnet, die bei der automatischen Vernetzung erzeugten Dreiecke sind dunkel schattiert gezeichnet. Es sind nun 26.192 Dreiecke mit 1.351 offenen Kanten vorhanden. Die maximale Kantenlänge der Verbindungsdreiecke wurde auf 3,0 mm festgelegt.
Bild (d) : Die 1.351 offenen Kanten nach dem automatischen Vernetzungs - Schritt sind rot gezeichnet, die bei der Triangulierung verwendeten Punkte sind cyanfarbend. Danach werden die Randkanten auf geschlossene Lochumrandungen untersucht, die im ersten Schritt automatisch geschlossen werden (Close Gaps). Die nun noch vorhandenen 97 offenen Kanten und die bei der Triangulierung verwendeten Punkte sind in Bild (e) dargestellt.
Bild (f) : zeigt das Ergebnis der vollständigen Triangulierung : 26.941 Dreiecke in schattierter Darstellung. Damit die Triangulierung ein Volumen umschließt, müssen die acht noch vorhandenen Löcher interaktiv geschlossen werden. Dazu wurde in [AMS 97] ein Modul entwickelt, das nacheinander die einzelnen Löcher in der Triangulierung anzeigt und anschließend das Loch, durch interaktives Löschen von eventuell auftretenden Überlappungen zwischen Dreiecken oder durch das Hinzufügen von einigen wenigen neuen Verbindungsdreicken, halbautomatisch schließt. Nach dem interaktiven Schließen der Löcher gibt es 27.036 Dreiecke.

4.2.3 Komplexitätsabschätzungen zum Verfahren der vollständigen Triangulierung

Durch die Zerlegung der gesamten Punktwolke in kleine Quader, in denen gewährleistet ist, daß alle Punkte innerhalb des Quaders von einer hinterschneidungsfreien Punktwolke stammen, und die ebene Triangulierung der Punkte eines Quaders für sich, kann eine deutliche Steigerung der Laufzeit des gesamten Triangulierung einer beliebigen 3D - Punktwolke erreicht werden. Es ist bei der Triangulierung einer beliebigen räumlichen Punktwolke auf diese Art möglich, eine Million Punkte in zwei Minuten zu triangulieren (auf einer SGI Indigo 2 R4400 Workstation mit genügend Hauptspeicher ca. 256 MB). Dieses kann dadurch erreicht werden, daß durch die Zerlegung der gesamten Punktwolke in kleinere Quader auch die Anzahl der Punkte pro Quader nur ein Bruchteil der Gesamtpunktanzahl in der Punktwolke beträgt und so der Berechnungsaufwand für die ebene Delaunay Triangulierung von

$O(N \log N)$ nur noch für sehr kleine N in die Gesamtrechenzeit einfließt.
Der Vernetzungsalgorithmus nutzt dabei die aus der Zerlegung der gesamten Punktwolke in kleine Quader bekannten Nachbarschaftsbeziehungen zwischen den Quadern aus (Baumstruktur), so daß im Vernetzungsalgorithmus ebenfalls nur lokal nach möglichen Verbindungsdreiecken zwischen den Triangulierungen der einzelnen Quader gesucht werden muß.

Triangulierungs - Statistik Datensatz `sh-clean.pgk` 203.771 Punkte 4.818 KByte					
Parameter Min. Punktabstand in mm	8,0	4,0	1,0	0,5	0,25
Parameter Max. Kantenlänge in mm	12,0	6,0	3,0	1,2	0,6
Randkanten vor Stitching	749	1.972	5.258	9.418	15.688
Dreiecke vor Stitching	919	2.350	21.940	68.602	214.366
Randkanten nach Stitching	538	1.318	1.407	2.466	8.184
Dreiecke nach Stitching	1.473	3.908	26.801	76.974	225.338
Randkanten nach autom. Close Gaps	141	244	256	560	1.565
Dreiecke nach autom. Close Gaps	1.729	4.578	27.458	78.054	229.209
Zeitbedarf autom. Triangulierung (sec.)	8	10	16	32	62
mittlere Kantenlänge der Dreiecke	4,576	2,823	1,175	0,698	0,411
Anzahl der restl. Löcher	7	9	8	7	14
Gesamtanzahl der Dreiecke	1.850	4.798	27.686	78.514	230.508
Gesamtanzahl der benutzten Punkte	927	2.401	13.845	39.259	115.256
Oberfläche der Triangulierung (mm^2)	14.960	14.989	15.053	15.095	15.211
Volumen der Triangulierung (mm^3)	139.616	140.824	141.494	141.572	141.584
Zusatzspeicherbedarf Triangulier. (KB)	1.772	2.054	4.244	9.109	23.655
Zusatzspeicherbedarf / Dreieck (Byte)	980,86	438,41	158,30	118,80	105,68
Gesamtspeicherbedarf / Dreieck (Byte)	3634,13	1466,72	333,01	207,89	127,21

Der Rechenzeitbedarf für das der vollständige Triangulierung von beliebigen 3D - Punktwolken mit der Zerlegung in hinterschneidungsfreie Bereiche und deren ebener Delaunay - Triangulierung ist, wie aus den beiden Diagrammen aus Abb. 4.7 ersichtlich, linear abhängig von der Anzahl der erzeugenten Dreiecke. In dieser Abbildung werden im linken Diagramm der Speicherbedarf, die Anzahl der erzeugten Dreiecke und die Laufzeiten des Algorithmus dargestellt. Für das Diagramm im rechten Bild wurde die ursprünglich gegebene Meßpunktwolke des Schalthebelknaufs (203.771 Meßpunkte) durch einfaches Subsampling in Punktwolken verschiedener Größen ausgedünnt (bei gleicher Qauderzerlegung). Die ausgedünnten Punktwolken (Auflösung mit einem Datenvolumen von ca. 2,5 %, 6,25 %, 12,5 %, 25,0 %, 50,0 %, 75,0 % der gesamten Punktwolke) wurden jeweils mit dem oben vorgestellten Algorithmus trianguliert. Es wurde die benötigte Zeit und die Anzahl der erzeugten Dreiecke in das Diagramm eingetragen.
Die Steigerung der Rechenzeit bei der Erhöhung der Anzahl der insgesamt zu bearbeitenden Punkte ist, wie im rechten Diagramm zu sehen, ebenfalls nur linear. Dieses wird durch die Zerlegung der Punktwolke in Quader erreicht.

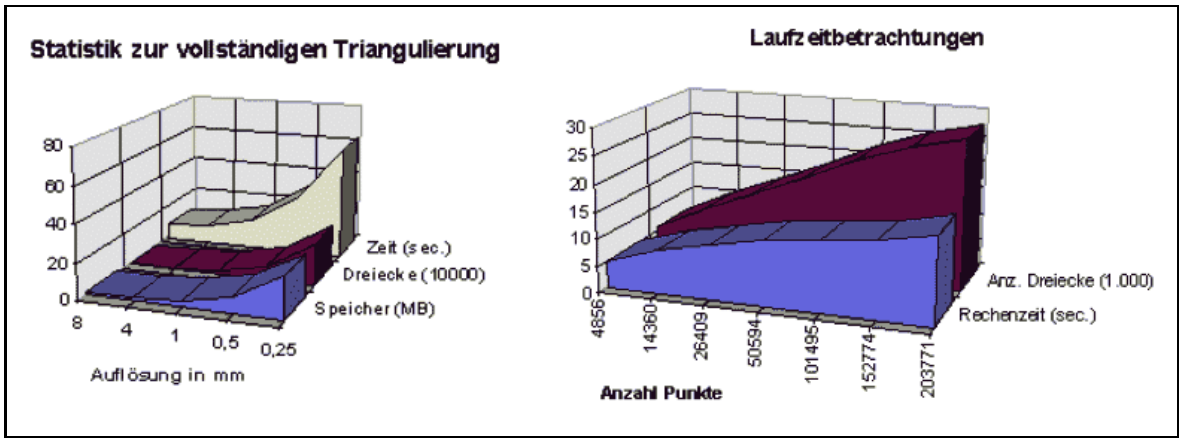


Abbildung 4.7 : Diagramme zu den Triangulierungstatistiken

4.2.4 Fehlerbetrachtungen zum Verfahren der vollständigen Triangulierung

Die Genauigkeit mit der das soeben vorgestellte Verfahren der vollständigen Triangulierung, die gegebene Punktwolke annähert, kann nicht durch eine allgemeine Fehlerabschätzungsformel angegeben werden.
Die experiementelle Berechnung der Fehler erfolgt nach folgendem Prinzip : Zu jedem Dreieck T_i der Triangulierung wird ein umgebender Quader (Bounding - Box) bestimmt. Der Quader enthält die drei Eckpunkte des Dreiecks $(V_{i,0}, V_{i,1}j, V_{i,2})$ sowie die beiden Punkte $R_i^+ := S_i + d \cdot N_i$ mit dem Schwerpunkt $S_i = \frac{1}{3}(V_{i,0} + V_{i,1} + V_{i,2})$ und dem Normalenvektor N_i des Dreiecks (vgl. Skizze in Abb. 4.8) sowie einem vorzugebenden maximalen Suchabstand d.

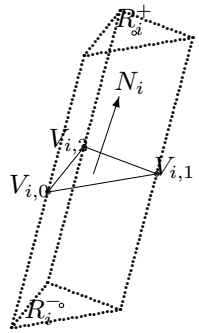


Abbildung 4.8 : Skizze zur Bestimmung einer Suchregion zur Fehlerabschätzung

Es werden alle Punkte, die innerhalb des Quaders liegen, in die durch S_i und N_i aufgespannte Ebene projiziert und der orthogonale Abstand für alle Punkte deren Projektion innerhalb oder auf dem Rand des Dreiecks T_i liegt als Abstand des Punktes zur Triangulierung definiert. Diese Suchregion ist durch die gepunkteten Linien in Abb. 4.8 angedeutet. Da ein Punkt aber in den Suchregionen mehrere Dreiecken liegen kann, wird der Punkt dem Dreieck zugeordnet, zu dem er den kleinsten orthogornalen Abstand besitzt. In der auf Seite 55 folgenden Tabelle der Fehlerberechnungen für drei Triangulierung unterschiedlicher Auflösungen zur Punktwolke des Schalthebelknaufs wurde experimentell gezeigt, daß der vorgegebene Parameter der Auflösung der Triangulierung als eine obere

Schranke für den maximal auftretenden Fehler angesehen werden kann. Da in der Punktwolke auch einige Meßwertausreißer vorhanden sind, überschreitet der maximal auftretende Fehler in einigen wenigen Punkten diese Schranke. Man sieht aus der Tabelle der Fehlerverteilungen, daß mindestens 99 Prozent aller Punkte der Punktwolke in einem Intervall von $[-\frac{r}{5}, +\frac{r}{5}]$ liegen, wobei r den Parameter der Auflösung (*engl.* **r***esolution*) beschreibt. Als Fehlerschranke d wurde jeweils $d = 2r$ verwendet.

Fehlerabschätzungen für Datensatz `sh-clean.pgk` mit 203.771 Punkten

Parameter / Auflösung in mm	8,0	~ 7,7%	4,0	~ 3,7%	1,0	~ 0,9%
mit. Kantenlänge der Dreiecke	4,4805		2,8191		1,1791	
Anzahl der Dreiecke	1.850		4.798		27,686	
max. / mit. pos. Abweichung	2,2433	0,1145	0,4995	0,0610	3,9650	0,0367
max. / mit. neg. Abweichung	-3,3647	-0,0341	-3,3343	-0,0246	-3,3276	-0,0235
arithm. Mittel / Standardabw.	0,0009	0,0143	0,0010	0,0098	0,0016	0,0164
Fehlerverteilung	# Pkt.	Proz.	# Pkt.	Proz.	# Pkt.	Proz.
-4,0 - -1,0	35	0,017	11	0,006	5	0,002
-1,0 - -0,4	618	0,305	4	0,002	6	0,003
-0,4 - -0,2	1.556	0,766	180	0,088	61	0,030
-0,2 - 0,0	25.248	12,435	41.686	20,472	88.618	43,498
0,0 - 0,2	126.891	62,496	154.194	75,726	114.599	56,251
0,2 - 0,4	38.587	19,005	7.450	3,659	330	0,162
0,4 - 1,0	9.492	4,676	97	0,048	30	0,014
1,0 - 4,0	611	0,3	0	0,000	79	0,039

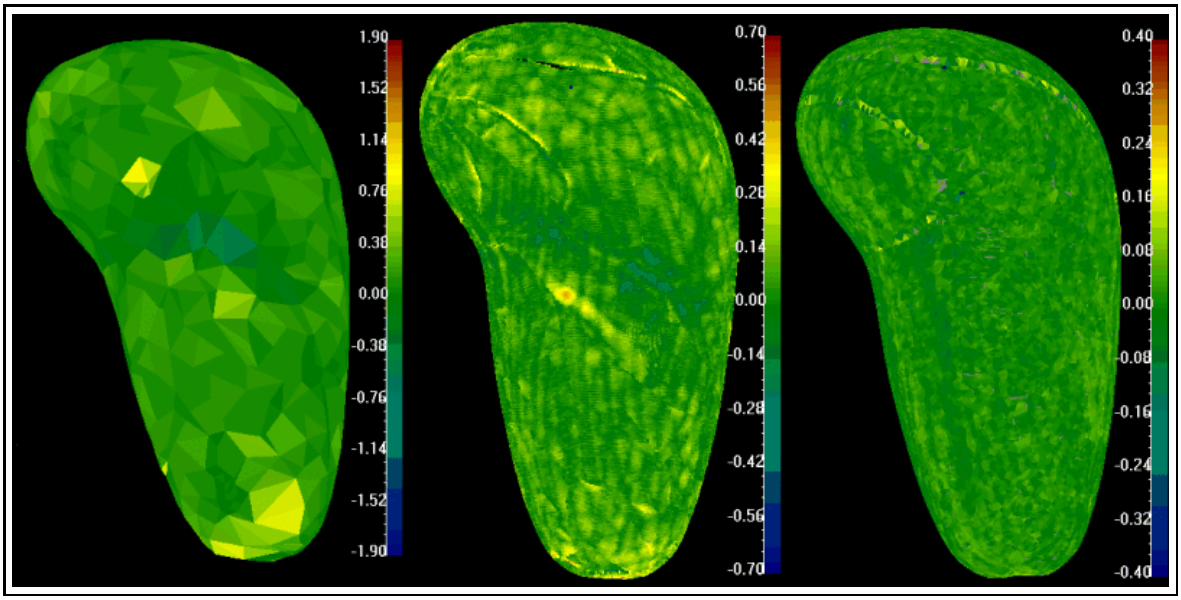


Abbildung 4.9 : Fehlerdarstellung für vollständige Triangulierung in verschiedenen Auflösungen

Bild links : *Fehlerdarstellung bei Auflösung 8 mm - Dreiecke farblich skaliert 1.9mm*

Bild mitte : *Fehlerdarstellung bei Auflösung 4 mm - Punkte farblich skaliert 0.7mm*

Bild rechts : *Fehlerdarstellung bei Auflösung 1 mm - Dreiecke farblich skaliert 0.4mm*

4.3 Erzeugung einer Topologie - Triangulierung

In diesem Abschnitt wird ein Verfahren zur schnellen Erzeugung einer räumlichen Triangulierung von beliebig strukturierten Punktwolken erläutert. Grundlagen für den Algorithmus ist die Darstellung der Punktwolke in der Quader - Octree Datenstruktur aus dem letzten Kapitel. Die erzeugte Triangulierung wird Topologie Triangulierung genannt, da als Ergebnis eine topologisch korrekte Rekonstruktion der der Punktwolke zugrundeliegende Fläche erzeugt wird, welche die Topologie der Fläche nachbildet.

4.3.1 Voraussetzungen für die Topologie - Triangulierung

Die Punktwolke $\mathcal{P}$ mit N Punkten werde in gleichgroße Quader zur vorzugebenden Baumtiefe $b \mid b \in \{1, \cdots, 8\}$ zerlegt. Der Maximalwert 8 ergibt sich aus den Speicherplatzbeschränkungen heutiger Computerssysteme, ein Wert von $b = 8$ hätte zur Folge, daß die maximale Anzahl möglicher Quader von $8^8 = 1.677.216$ in einem Baum verwaltet werden müsste. Dieses würde die Speichergrenzen der heutzutage verwendeteten Computersysteme überschreiten. Es bezeichnen $X_{\min}, Y_{\min}$ und $Z_{\min}$ bzw. $X_{\max}, Y_{\max}$ und $Z_{\max}$ die minimalen bzw. maximalen $x-/y-$ und z-Koordinaten der Punktwolke. Ferner bezeichnen $\Delta X := X_{\max} - X_{\min}, \Delta Y := Y_{\max} - Y_{\min}$ und $\Delta Z := Z_{\max} - Z_{\min}$ die Kantenlängen der Bounding Box der Punktwolke $\mathcal{P}$.
Wenn man die Quader - Octree Datenstruktur zur Punktwolke $\mathcal{P}$ zu einer vorgegebenen Baumtiefe b aufbaut, teilt man die Bounding Box der gegebenen Punktwolke in 8^b gleichgroße Quader auf. Diese Quader haben die Kantenlängen von $\frac{\Delta X}{2^b}, \frac{\Delta Y}{2^b}, \frac{\Delta Z}{2^b}$.
Es ist auch möglich die Kantenlängen der Quader q vorzugeben, und die Punktwolke in Würfel mit der für alle Raumrichtungen gleichen Kantenlänge q zu zerlegen. Dies geschieht dadurch, daß man aus der Boundingbox, des die Punktwolke umgebenden Quaders $\mathcal{B} = (X_{\min}, X_{\max}, Y_{\min}, Y_{\max}, Z_{\min}, Z_{\max})$ eine neue Boundingbox zu der Kantenlänge $q \cdot 2^b$ (siehe unten) bestimmt. Die Baumtiefe b berechnet man aus der Boundingbox $\mathcal{B}$ der gegebenen Punkte und der vorgegebenen Kantenlänge q wie folgt :

- 1 .) Bestimmung der maximale Kantenlänge $q_{\max}$ der Boundingbox $\mathcal{B}$.

$$q_{\max} := \max \{ X_{\max} - X_{\min},\ Y_{\max} - Y_{\min},\ Z_{\max} - Z_{\min} \} \tag{4.2}$$

- 2.) Bestimmung der Baumtiefe b für Zerlegung der Punktwolke in gleichgroße Würfel zur vorgegebenen Kantenlänge q :

$$b := \left\lceil \frac{\log\ q_{\max}}{\log\ 2} \right\rceil \tag{4.3}$$

- 3.) Bestimmung der Bounding Box $\tilde{\mathcal{B}}$, die bei der Zerlegung zur Baumtiefe b in 8^b Würfel der Kantenlänge q zerfällt.

$$\begin{aligned}\tilde{\mathcal{B}} := (&\frac{X_{\max} + X_{\min}}{2} - q \cdot 2^{b-1}, \frac{X_{\max} + X_{\min}}{2} + q \cdot 2^{b-1}, \frac{Y_{\max} + Y_{\min}}{2} - q \cdot 2^{b-1}, \\ &\frac{Y_{\max} + Y_{\min}}{2} + q \cdot 2^{b-1}, \frac{Z_{\max} + Z_{\min}}{2} + q \cdot 2^{b-1}, \frac{Z_{\max} + Z_{\min}}{2} + q \cdot 2^{b-1}\ )\end{aligned} \tag{4.4}$$

Die Bounding Box $\tilde{\mathcal{B}}$ hat dann in allen Koordinatenrichtungen die Seitenlänge $q \cdot 2^b$. Bei einer Zerlegung zur Baumtiefe b erhält man also für jeden der 8^b Würfel, die bei der Zerlegung entstehen, die vorgegebene Kantenlänge von $q = \frac{q \cdot 2^b}{2^b}$.
In den Abbildungen 4.10 und 4.11 ist die Vorgehensweise während des Algorithmus zur Topologie Triangulierung am Beispiel einer optisch gemessenen Punktwolke eines Schalthebelsknaufs (linkes Bild aus Abb. 4.10) skizziert.
Für den Algorithmus zur Erzeugung der Topologie Triangulierung werden nur die Quader berücksichtigt, die auch Punkte der Punktwolke $\mathcal{P}$ enthalten. Diesen Sachverhalt muß man bei der Wahl der Baumtiefe b oder bei der Vorgabe der Kantenlänge q der Quader zur Bestimmung der Anzahl von Quadern berücksichtigen. Wenn man die Baumtiefe b zu groß wählt bzw. die Quader Kantenlänge q zu klein, kann es vorkommen, daß in einigen Quadern keine Punkte liegen und somit bei der Triangulierung Lücken entstehen, die man nachher manuell schließen muß. Wird die Baumtiefe b zu klein bzw. die Kantenlänge q zu groß gewählt, kann es vorkommen, daß einige feine Details der Punktwolke (Details und Strukturen, die komplett innerhalb eines Quader liegen) bei der Triangulierung nicht berücksichtigt werden. Beispielsweise können die Topologie - Triangulierungen zu den Punktwolken einer kompletten Kugel und eines Zylinder ohne Deckel ab einer Baumtiefe von $b = 2$ erzeugt werden. Für die Punktwolke eines Torus wird eine Baumtiefe ab $b \geq 3$ benötigt.
Für die Wahl der Quader - Kantenlänge q und die damit zusammenhängende Bestimmung der Baumtiefe b kann man die Anzahl der gewünschten Dreiecke (und damit die gewünschten Genauigkeit vgl. Abschnitt 4.3.4) heranziehen. Für die in dieser Arbeit verwendeten Beispiele wurden für die Baumtiefe b Werte zwischen drei und sechs verwendet. Die maximalen Anzahlen der Quader lagen dann zwischen $8^3 = 512$ und $2^6 = 262.144$, was für die meisten Anwendungen auch ausreichen dürfte.
Es wird für jeden mit Punkten der gegebenen Punktwolke belegten Quader genau ein Punkt der Punktwolke ausgewählt und zwar genau der Punkt der gegebenen Punktwolke, der am nächsten am Mittelpunkt des zugehörigen Quaders liegt. Dieser Punkt wird als Quadermittelpunkt Q_i des Quaders mit der Nummer i bezeichnet (vgl. Abbildung 4.10 Bild b). Der Algorithmus zur Erzeugung der Topologie Triangulierung verwendet nur die Mittelpunkte der Quader für die Erzeugung der Dreiecke, da durch diese Auswahl erreicht werden kann, daß die erzeugten Dreiecke möglichst regulär (möglichst kleine Unterschiede in den Flächeninhalten der einzelnen Dreiecke, inneren Winkel der Dreiecke möglichst nahe bei 60 Grad und gleichmäßige Kantenlänge aller Dreiecke) werden.
Aus der Quader Octree - Struktur kennt man zu jedem Quader (Voxel) die Menge der benachbarten Quader. Das sind all diejenigen Quader mit Punkten, die mit dem betrachteten Quader eine gemeinsame Randfläche besitzen. Die Menge dieser Nachbarquader wird als Nachbarn erster Ordnung bezeichnet (Schreibweise : $\mathcal{N}_i^1$ für die Nachbarindizes der Nachbar Quader des Quaders mit Nummer i). Da diese Quader alle die gleiche Größe besitzen sind dies maximal sechs Nachbarquader erster Ordnung. Die Menge der Nachbarquader zweiter Ordnung $\mathcal{N}_i^2$ zu dem Quader Nr. i erhält man als Vereinigung der Menge aller Nachbarquader erster Ordnung zu allen zum Quader Nr. i benachbarten Quadern erster Ordnung ohne den Quader Nr. i selbst. Die Menge der Nachbarn zweiter Ordnung kann maximal 22 Elemente besitzen, da alle Quader gleichgroß sind.

$$\mathcal{N}_i^2 \; := \; \Big(\bigcup_{k \, \in \, \mathcal{N}_i^1} \mathcal{N}_k^1 \Big) \setminus \{i\} \tag{4.5}$$

4.3.2 Startpunktsuche für die Topologie Triangulierung

Die Topologie - Triangulierung wird nun schrittweise beginnend mit einem Quader, der die meisten Nachbarn zweiter Ordnung besitzt, aufgebaut (Quader-Index : m). Zunächst werden durch Q_m und die Mittelpunkte Q_l aller Quader der Nachbarn erster Ordnung zu Quader Nr. m (Index $l \in \mathcal{N}_m^2$) eine ebene Delaunay Triangulierung, der in ihre Ausgleichsebene projizierten Mittelpunkte, berechnet (vgl. Abschnitt 4.2.1). Für das angeführte Beispiel kann man den markierten Startpunkt und die im ersten Schritt erzeugten Dreiecke im Bild (c) der Abb. 4.10 betrachten.

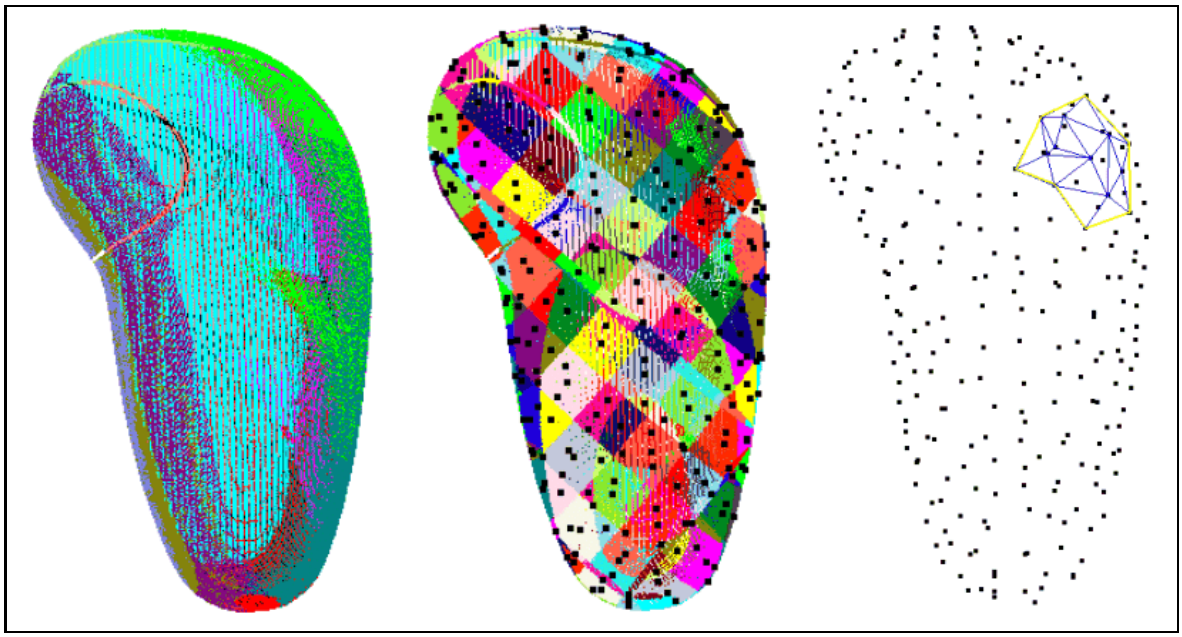


Abbildung 4.10 : Beginn des Algorithmus zur Topologie Triangulierung

Bild links : *Beispiel Datensatz sh-clean.pgk 203.771 Meßpunkte eines Schalthebelknaufs (in 10 Ansichten aufgenommen, Länge der maximalen Bounding Box Kante : 93 mm)*

Bild mitte : *Die Meßpunkte des obigen Schalthebels wurden in 263 gleichgroße Quader (Kantenlänge 9,3 mm $\sim$ 10 % der längsten Kante der Punktwolke) zerlegt, Einfärbung der Punkte nach Quadern. Der fett markierte Punkt beschreibt den Meßpunkt innerhalb des Quaders mit der Nr. i, der am nächsten an dem Mittelpunkt seiner Bounding-Box liegt (Bezeichnung : Quadermittelpunkt Q_i).*

Bild rechts : *Suchen eines Startpunktes. Es wird der Punkt aus den Quader-Mittelpunkten bestimmt, der die meisten Nachbarn zweiter Ordnung besitzt. Berechnung einer Ausgleichsebene durch den Punkt und all dessen Nachbarpunkte und anschließender ebener Delaunay-Triangulierung.*

4.3.3 Schrittweise Erweiterung der Triangulierung

An die Randkanten dieser Triangulierung wird versucht schrittweise neue Dreiecke anzufügen, bis alle Quader-Mittelpunkte bei der Triangulierung verwendet wurden und keine Dreiecke mehr an die Rand-

kanten der Triangulierung angehängt werden können.
Zu einer Randkante der Triangulierung kennt man deren zwei Eckpunkte (Mittelpunkte der Quader mit Nr. i und $j, i \neq j$ Bezeichnung : Q_i bzw. Q_j), zusätzlich ist noch das Dreieck zu dem die betrachtete Kante gehört und somit auch der dritte Eckpunkt des Dreiecks bekannt der mit Q_k ($k \neq i,\ k \neq j$) bezeichnet wird. Zum Finden eines neuen Dreiecks wird die Schnittmenge aus den Mengen der Nachbarn zweiter Ordnung der Quader i und j bestimmt. Aus diesem Durchschnitt aller Quaderindizes l mit $l \in \mathcal{N}_i^2 \bigcap \mathcal{N}_j^2$ werden alle zu den Quaderindizes gehörenden Mittelpunkte Q_l als potentielle Kandidaten für ein neues Dreieck mit der Kante $(Q_i\ ,\ Q_j)$ betrachtet, die den folgenden vier Bedingungen genügen :
1.) Der Punkt Q_l wurde bei der Triangulierung noch nicht verwendet oder der Punkt gehört zu mindestens einer Kante, die sich am Rand der bisher erzeugte Triangulierung befindet.
2.) Der Punkt Q_l liegt auf der anderen Seite der Kante (Q_i, Q_j) als der Punkt Q_k aus vorhandenen Dreieck (Q_i, Q_j, Q_k). Dabei darf der Winkel zwischen den Normalenvektoren der Dreiecke (Q_i, Q_j, Q_k) und (Q_j, Q_i, Q_l) maximal den vorzugebenden Schwellenwinkel (ca. 105 Grad)betragen.
3.) Der Punkt Q_l liegt nicht weiter entfernt von der betrachteten Randkante als der Schwellenwert von $\sqrt{2} \cdot q$.
4.) Das Dreieck bestehend aus dem Punkt Q_l und den Punkten Q_i und Q_j erzeugt keine Durchdringungen mit schon bestehenden Dreiecken der Triangulierung.
Falls mehrere Punkte diesen Anforderungen genügen, wird derjenige Punkt für ein neues Dreieck ausgewählt, bei dem der Winkel zwischen den hinzugefügten Dreieckskanten am wenigsten von 60 Grad abweicht (Regularitätsbedingung). Der neue Punkt sei mit Q_l bezeichnet.
Das Dreieck mit den Punkten Q_l, Q_j, Q_i wird in die Liste der erzeugten Dreiecke eingefügt und die Liste der Randkanten wird aktualisiert (Die Kante (Q_i, Q_j) wird entfernt und die Kanten (Q_l, Q_j) und (Q_i, Q_l) dieser Liste hinzugefügt bzw. falls sie schon vorhanden sind werden sie ebenfalls gelöscht).
Falls kein Punkt für die Erzeugung eines Dreiecks gefunden werden kann, wird die Kante (Q_i, Q_j) aus der Liste der Randkanten gelöscht. Die Kante (Q_i, Q_j) ist dann eine, die zum Rand der Topologie Triangulierung gehört.

Details zur Abbildung 4.11 auf Seite 60 - Beispiel einer Topologie Triangulierung

Bild (a) (b) (c) und (d) : Einfügen eines neuen Dreiecks an eine der gelb markierten Randkanten des voherigen Bildes (insgesamt wurden 15, 16, 17, bzw. 19 Dreiecke erzeugt). Die gelben Kanten beschreiben die Randkanten der Triangulierung an den die neue Dreiecke angehängt werden. Die blauen Kanten gehören zu den vollständig bearbeiteten Punkten.
Bild (e), (f), (g) und (h) : Einfügen eines neuen Dreiecks an eine der gelb markierten Randkanten des vorherigen Bildes (insgesamt wurden 32, 64, 128 bzw. 256 Dreiecke erzeugt).
Bild (i) und (j) : Einfügen eines neuen Dreiecks an eine der gelb markierten Randkanten des vorherigen Bildes (insgesamt wurden 512 bzw. 522 Dreiecke erzeugt).
Bild (k) und (l) : Ergebnis der Topologie Triangulierung : 522 Dreiecke mit einfacher Schattierung Bild (k) (flat shaded) und mit Gouraud Schattierung Bild (l) (smooth shaded).

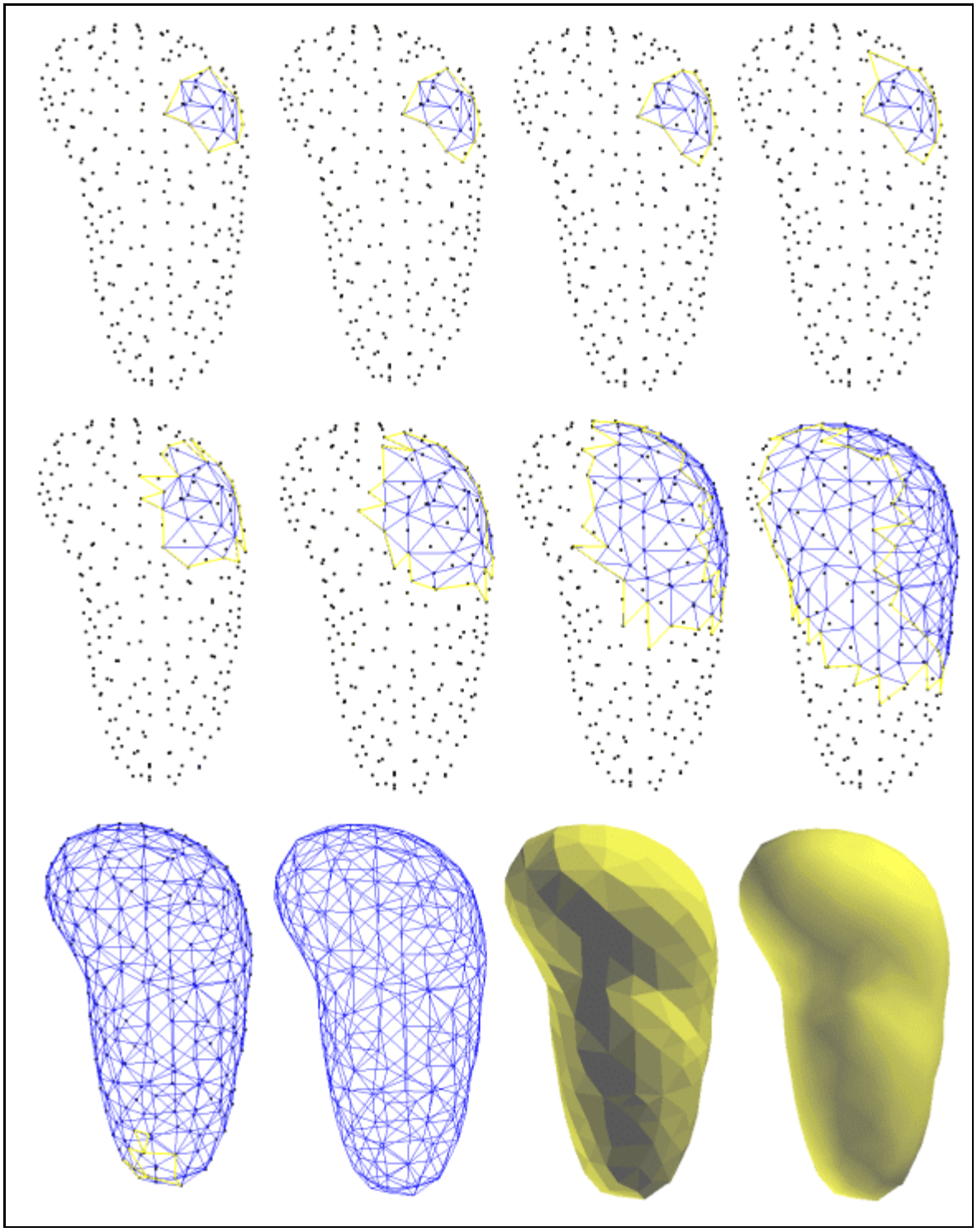


Abbildung 4.11 : Schrittweise Ausführung des Topologie Triangulierungsalgorithmus
Bild (a,e,i) : Bild (b,f,j) : Bild (c,g,k) : Bild (d,h,l) :

Triangulierungs - Statistik für Datensatz `sh-clean.pgk` mit 203771 Punkten					
Parameter Voxel Kantenlänge (mm)	9,3	4,0	2,0	1,0	0,5
entspricht Auflösung von %	10,00	3,72	1,86	0,93	0,46
Anzahl der Quader	263	1.378	5.369	20.374	49.762
Anzahl der autom. erzeugten Dreiecke	522	2.736	10.697	39.582	90.467
Dauer autom. Triangulierung (sec.)	9	15	41	109	418
mittlere Kantenlänge der Dreiecke	8,3890	3,6902	1,8737	0,9718	0,6291
Anzahl der restl. Löcher	0	1	2	6	48
Gesamtanzahl der Dreiecke	522	2.738	10.710	40.582	98.266
Anzahl bei Triangul. benutzter Punkte	263	1.371	5.357	20.293	49.160
Oberfläche der Triangulierung (mm^2)	14.789	14.981	14.981	15.022	15.184
Volumen Triangulationsmodells (mm^3)	136.349	140.604	141.398	141.600	141.657
Speicherbedarf Quader Octree (KB)	4.818	4.978	5.541	7.630	11.676
Zusatzspeicher Triangulierung (KB)	1.645	1.857	2.620	5.478	10.933
Gesamtspeicherbedarf / Dreieck (Byte)	12678,7	2556,1	780,3	330,8	237,9

4.3.4 Fehlerbetrachtung für die Topologie Triangulierung

Die Genauigkeit, mit der die Topologie - Triangulierung die gegebene Punktwolke annähert, kann durch die vorgegebene Quaderkantenlänge q abgeschätzt werden. Es sei d der Durchmesser eines Quaders zur Baumtiefe b mit :

$$d := \frac{\sqrt{(\Delta X)^2 + (\Delta Y)^2 + (\Delta Z)^2)}}{\sqrt{2^b}} \tag{4.6}$$

wobei $\Delta X, \Delta Y$ und ΔZ die Kantenlängen der Bounding Box der Punktwolke bezeichnen. Mit der vorgegebenen Qauderkantenlänge q gilt für d :

$$d \;\leq\; \sqrt{3}\, q \quad \text{da} \;\; d = \frac{\sqrt{(\Delta X)^2+(\Delta Y)^2+(\Delta Z)^2)}}{\sqrt{2^b}} \leq \frac{\sqrt{3\, q_{max}^2}}{\sqrt{2^b}} \leq \sqrt{3}\, q \; .$$

Die Punkte innerhalb des Quaders Nr. i können im ungünstigsten Fall (Mittelpunkt Q_i liegt fast an einem Eckpunkt des Quaders Nr. i) um maximal d vom Mittelpunkt Q_i abweichen. Da jeder Quader - Mittelpunkt bei der Triangulierung verwendet wird, ist auch der maximale Abstand zwischen Punktwolke und Triangulierung durch en Wert d nach oben beschränkt. Bei Punktwolken mit gleichmäßigem Punktabstand und passender Wahl der Baumtiefe b (nicht zu klein) sollte die Abweichung der Punkte im Quader vom Mittelpunkt des Quaders durch $\frac{d}{2}$ nach oben beschränkt sein, was auch in der auf Seite 62 folgenden Tabelle der Fehlerabschätzungen für das hier vorgestellte Beispiel erreicht wird. Die Fehlerberechnungen erfolgten nach dem im Abschnitt 4.2.4 vorgestellten Prinzip.
Die Kantenlänge der erzeugten Dreiecke liegt im Mittel ebenfalls bei dem Durchmesser der einzelnen Quader d (vgl. Tabelle mit Triangulierungs - Statistik).
Der Algorithmus arbeitetet sehr gut bei Punktwolken von Objekten ohne scharfe Kanten (Torus, Kugel, Beispiel des Schalthebels, ...). Bei Punktwolken von scharfkantigen Objekten (z.B. Pyramide) liegen die Mittelpunkte der einzelnen Quader nicht immer genau auf der Kante. Das hat zur

Folge, daß auch die Abweichungen zwischen Punktwolke und Triangulierung in diesen Bereichen am größten werden und daß die Kanten in diesen Bereichen nicht genau rekonstruiert werden können. Eine Abhilfe wäre hier eine adaptive Anpassung der Triangulierung an die Punktwolken, bei denen zunächst Dreiecke mit relativ großer Abweichung zur Punktwolke gesucht werden. Diese Dreiecke werden dann, durch Hinzufügen eines neuen Punktes in der Dreiecksmitte, in drei neue Dreiecke zerlegt. Als neuer Punkt des Dreiecks (Q_i, Q, j, Q_l) wird derjenigen Punkt der Punktwolke aus den Quader mit den Nummern i, j, k ausgewählt, dessen Projektion in die durch das Dreieck aufgespannte Ebene am nächsten zu dem Schwerpunkt $S = \frac{1}{3}(Q_i + Q_j + Q_k)$ des Dreiecks liegt.
Da die Zerlegung der Punktwolke in Quader aus implementierungstechnischen Gründen immer in achs-parallele Quader erfolgt, ist die Triangulierung einer beliebig im Raum liegenden Punktwolke nachdem in diesem Abschnitt beschriebenen Verfahren, abhängig von der Wahl dieses Koordinatensystems. Für einige beliebig im Raum liegenden Datensätze kann es daher nötig sein, die Punktwolke in ein geeignetes Koordinatensystems (z.B. Hauptachsentransformation) zu transformieren. Dieser Transformationsprozeß wird meistens interaktiv ablaufen. Bei einem Zylinder sollten die Achsen beispielsweise wie folgt gewählt sein : Der dem Zylinder zugrundeliegende Kreis sollte als X/Y Ebene und die Rotationsachse des Zylinders als Z - Achse definiert sein.

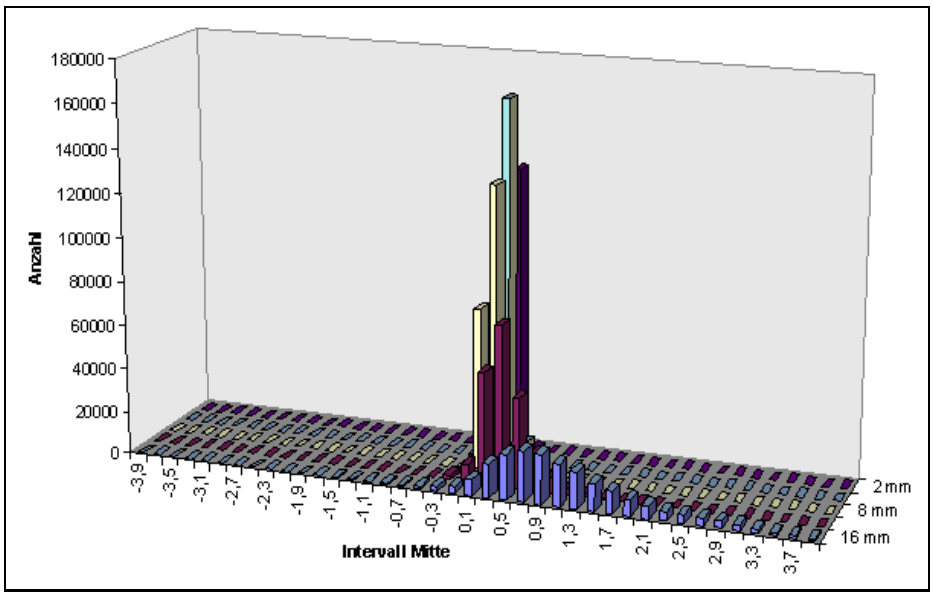


Abbildung 4.12 : Fehlerdiagramm bei der Topologie Triangulierung

Fehlerabschätzungen für Datensatz `sh-clean.pgk` mit 203.771 Punkten						
Parameter (mm)/# Dreiecke	9,3	522	4,0	2.738	2,0	10.710
max./mit. Abst. Punkte - Q_i	8,5502	5,6591	3,6292	2,3812	2,1588	1,1389
max./mit. Abw. Ausgl.ebene	3,3604	2,9238	0,4912	0,2476	0,5723	0,0870
max./mit. pos. Abweichung	1,8963	0,3386	0,6889	0,0774	0,4081	0,0393
max./mit. neg. Abweichung	-2,8736	-0,0595	-3,1495	-0,2522	-2,8017	-0,0255
arithm. Mittel/Standardabw.	0,0009	0,0196	0,0009	0,0094	0,0009	0,0073
Fehlerverteilung	# Pkt.	Proz.	# Pkt.	Proz.	# Pkt.	Proz.
-4,0 - -1,0	15	0,007	15	0,007	14	0,006
-1,0 - -0,4	277	0,138	8	0,004	0	0,000
-0,4 - -0,2	1681	0,834	478	0,235	8	0,004
-0,2 - 0,0	7.688	3,814	30.182	14,822	73.794	36,217
0,0 - 0,2	50.147	24,875	164.265	80,669	129.494	63,554
0,2 - 0,4	72.197	35,813	8.532	4,190	443	0,217
0,4 - 1,0	64.181	31,837	147	0,072	1	0,000
1,0 - 4,0	5.407	2,685	1	0,000	0	0,000

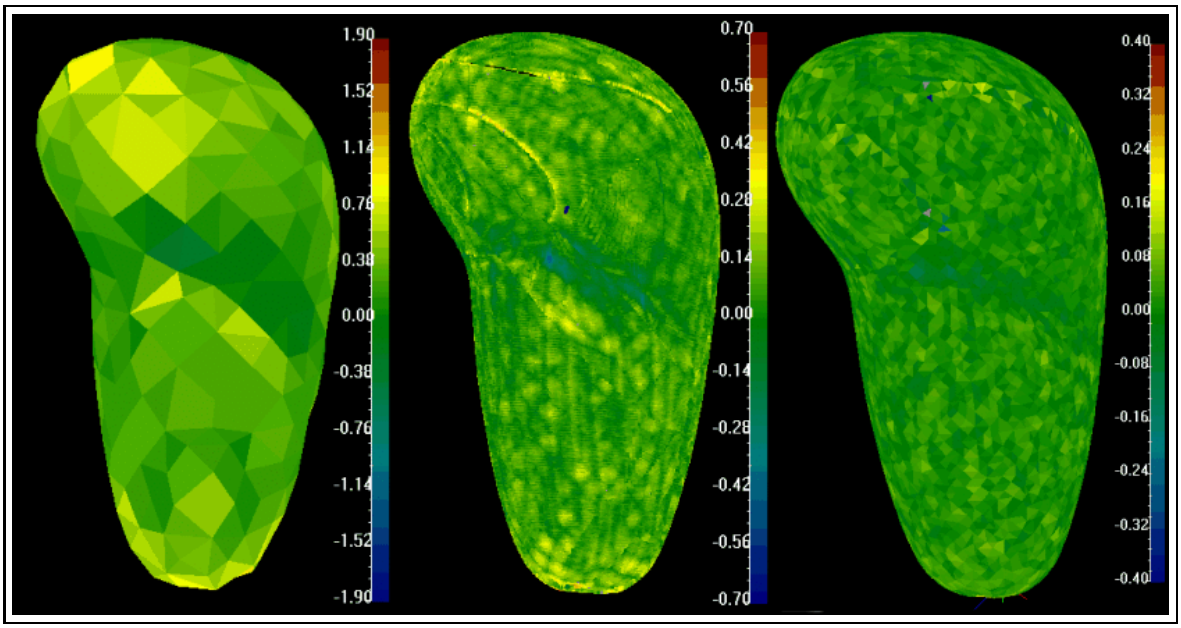


Abbildung 4.13 : Fehlerdarstellung für Topologie Triangulierungs Methode

Bild links : *Fehlerdarstellung bei Auflösung 9,3 mm - Dreiecke farblich skaliert 1,9mm*

Bild mitte : *Fehlerdarstellung bei Auflösung 4,0 mm - Punkte farblich skaliert 0,7mm*

Bild rechts : *Fehlerdarstellung bei Auflösung 2,0 mm - Dreiecke farblich skaliert 0,4mm*

4.3.5 Komplexitätsabschätzungen der Topologie Triangulierung

Der Algorithmus zur Erzeugung der Topologie Triangulierung ist linear in der Anzahl der Quader, die mit Punkten aus der Punktwolke belegt sind. Beim Suchen der neuen Punkte zu einer bestehenden Randkante (Q_i, Q_j), die zu dem vorhandenen Dreieck (Q_i, Q_j, Q_k) gehört, wird zunächst der Durchschnitt der aus jeweils maximal 22 Elemente bestehenden Mengen der Nachbarn zweiter Ordnung der Quader i und j bestimmt. Danach wird für jeden Index $l \neq k$ aus diesem Durchschnitt der Punkt Q_l als ein potentieller Eckpunkt für ein neues Dreieck betrachtet. Dabei wird das potentielle neue Dreieck (Q_j, Q_i, Q_l) auf Durchdringung mit bereits bestehenden Dreiecken überprüft. Hierzu werden nur die aus der Datenstruktur für die Triangulierung bekannten Nachbardreiecke der Punkte Q_i, Q_j, Q_l und Q_k durchsucht. Diese Suche ist ebenfalls lokal möglich und hängt nicht von der Gesamtanzahl der Quader oder der Dreiecke ab.

Rechenzeitbedarf für Datensatz `sh-clean.pgk` mit 203.771 Punkten							
Parameter in mm	16,0	9,3	8,0	4,0	2,0	1,0	0,5
entspricht Auflösung von %	14,83	10,0	7,41	3,70	1,85	0,93	0,46
Anzahl der Quader	86	263	346	1.378	5.369	20.374	49.762
Anzahl der autom. erzeugten Dreiecke	160	522	686	2.736	10.697	39.582	90.467
Rechenzeitbedarf Gesamt (sec.)	8	9	11	15	41	109	350
davon Anteil Umstrukturierung Punkte	4	5	6	7	9	21	32
davon Anteil Aufbau Nachbarschaft	1	1	2	3	10	29	155

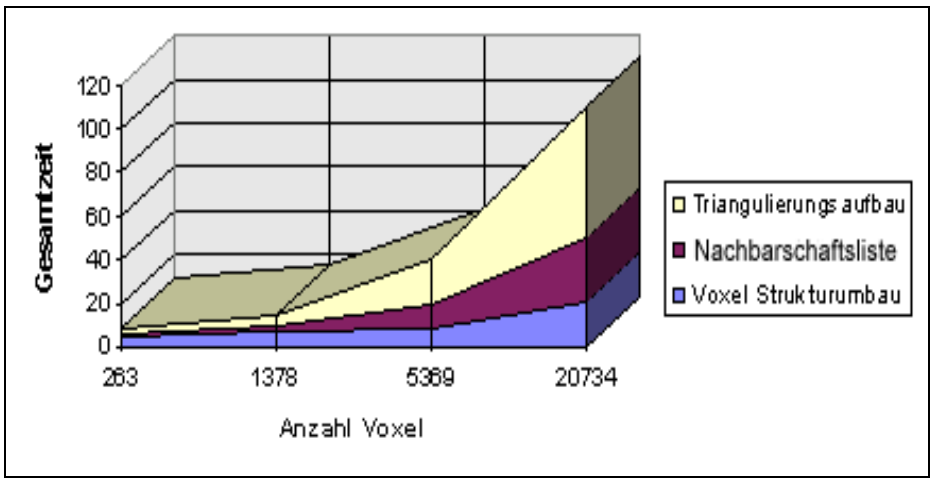


Abbildung 4.14 : Diagramm der Laufzeiten für die Topologie Triangulierung

4.4 Vergleich der Triangulationsverfahren für 3D - Punktwolken

Verfahren / Literatur	Vorteile	Nachteile
3D - Delaunay Triangulierung [Barb 95], [Fac 95], [PreSh 85], [Joe 91]	- unstrukturierte Punkte - automatisch bei konvexen Objekten - verschiedene Auflösungen - für alle Anwendungen geeignet	- sehr hohe Laufzeit - nur kleine Punktanzahlen - Tetraeder - konvexe Hülle - Nachbearbeitung notwendig
Alpha Shapes [BaBeX 95a], [EdMu 94], [Clark 95]	- verschiedene Auflösungen - unstrukturierte Punkte	- baut auf Tetraedern auf - sehr hohe Laufzeit - nur kleine Punktanzahl - Probleme an Kanten und bei unterschiedlichen Punktdichten
Triangulierung von Ansichtsbildern [Karb 94], [Roth 96], [SouLa 94], [TuLe 94]	- sehr schnelle Laufzeit - Triangulierung einzelner Bilder. - verschiedene Auflösungen - sequentielles Verfahren - hohe Punktanzahl - gute Rekonstruktion von Kanten	- nur auf Punktwolken im Range Image Format anwendbar - aufwendige Integration der einzelnen Triangulierungen - nur für Visualisierungs-Anwendungen geeignet
Triangulierung von Scan-Linien Kapitel 4.1.4 [BIE 96B] [Wil 94]	- vollständige Triangulierung - alle Anwendungen - schnelle Laufzeit - hohe Punktanzahl möglich	- nur für Punktwolken im Scan-Linien Format anwendbar. - Probleme an Kanten - Auflösung ist von Scan-Linien Dichte abhängig
Voxelbasierte Triangulierung [LoCli 87], [HoDeR 92], [Hoppe 94], [Roth 97], [Wyv 86]	- hohe Punktanzahl - iteratives Verfahren - mittelmäßige Laufzeit - verschiedene Auflösungen	- synthetische Dreieckspunkte - nur für Visualisierungs-Anwendungen geeignet - Nachbearbeitung notwendig - Probleme an Kanten - nur auf Range Images schnell
Vollständige 3D - Triangulierung Kapitel 4.2 [Bow 81], [JUN 95A] [Shew 95], [Wil 92]	- sehr schneller Algorithmus - unstrukturierte, große Punktwolken - beliebige Auflösungen - alle Anwendungen - Kanten Rekonstruktion möglich	- Bei starkem Meßwertrauschen enstehen Löcher, die interaktiv nachbearbeitet werden können - Zerlegung der Punktwolke in, Hinterschneidungsfreie Bereiche
Topologie Triangulierung Kapitel 4.3 [Sam 90], [JUN 95B]	- automatische Triangulierung - unstrukturierte, große Punktwolken - vollständige Methode, auch für Röntgendaten - alle Anwendungen - verschiedene Auflösungen	- lange Laufzeit bei hoher Auflösung - Probleme an Kanten