Institut für Rechnergestützte Ingenieursysteme

Fakultät Informatik, Elektrotechnik und Informationstechnik

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Diplomarbeit Nr. 3273

CAD Feature Recognition of Machining Parts

Dursun Kemal Erbas

Studiengang:	INFORMATIK
Prüfer:	Univ-Prof. Hon-Prof. Dr. Dieter Roller
Betreuer:	Dr. Bernadetta Kwintiana Ane,
	M.Sc. Leila Zehtaban
begonnen am:	14.11.2011
beendet am:	15.05.2012
CR-Klassifikation:	I.3.3, I.3.5, I.7.5, J.6

Abstract

The advancement of 3D scanning technology lends us the ability to capture the shape of machining parts quickly by creating a set of points from the surface of the object, so-called points cloud. The 3D scanning technology supports efficient feature recognition by the identification of geometric features on the existing points cloud automatically, as well as the reconstruction of the object for further processing with the available CAD systems. Feature recognition consists of several steps, which mostly require the extraction of geometric features like normal vector and curvature on each point of the surface and the segmentation of points cloud.

Generally, there are two fundamental approaches to calculate normal vector and curvature on the points surface: analytical and numerical. The first approach compared to the second one, is practically more complex. However, the numerical approach has some weakness particularly in detecting sharp edges and transitions (i.e., adjacent edges) between different geometric objects.

To improve the feature recognition process, a modified method is presented and implemented in this thesis that is able to perform recognition directly from unstructured points cloud. Here "unstructured" means the points cloud are arbitrarily disordered and each point has no information on its neighborhood. In the proposed method, the geometric features such as normal vector and curvature are determined at each point surface. The calculation of normal vector is performed by enhancing the existing numerical local triangulation method, which is now being called *Local Triangulation with Direct Neighboring Points*. In this method, a new process has been developed that is able to eliminate outliers in the neighborhood and thus the direct neighbors of a given point can be determined. Using this method, the local triangulations at sharp edges and transitions are optimized, so that both the normal vector and the curvature can be computed efficiently in a single step, without the analytical approach.

Using the obtained features, the normal vector based segmentation are carried out to detect the boundary edges and transitions between geometrically different objects. The detected edges and transitions are then used to identify the characteristics of surface, so that each points surface can be decomposed by the curvature-based segmentation and assigned to the related geometric primitives such as cubes, cylinders, or spheres. Finally, the recognized features of machining parts can be parameterized for further processing and conversion to the particular formats of CAD systems.

Kurzfassung

Die Oberflächen von Maschinenteilen lassen sich dank der ständigen Weiterentwicklung der 3D-Scan-Technologie immer schneller erfassen und erzeugen dabei eine Reihe von Punktdaten, die auch Punktwolke genannt werden. Die Entwicklung der Scan-Technologie erfordert effiziente Objekterkennungsprozesse, um die gescannten Maschinenteile mit ihren geometrischen Merkmalen möglichst automatisch aus einer vorliegenden Punktwolke zu erkennen und entsprechend zur Weiterverarbeitung der Objekte durch CAD-Systeme zu rekonstruieren. Objekterkennungsprozesse bestehen aus mehreren Schritten, bei denen vor der Objekterkennung die Extraktion der geometrischen Oberflächenmerkmale wie Normalenvektor und Krümmung an den einzelnen Punkten der Oberfläche sowie die Segmentierung der vorliegenden Punktwolke erforderlich ist.

Im Allgemeinen gibt es zwei grundverschiedene Ansätze, um den Normalenvektor und die Krümmung an Oberflächenpunkten zu berechnen: analytische und numerische. Analytische Ansätze sind im Vergleich zu numerischen Ansätzen operativ aufwendiger. Numerische Ansätze dagegen weisen Schwächen besonders an scharfen Kanten und Übergängen zwischen geometrisch unterschiedlichen Teilobjekten in einer Punktwolke auf.

Zur Verbesserung des Objekterkennungsprozesses wird in dieser Arbeit ein Verfahren vorgestellt und implementiert, bei dem der Objekterkennungsprozess direkt auf eine unstrukturierte Punktwolke angewendet wird. Unstrukturiert bedeutet hier, dass die Punkte aus der Punktwolke ungeordnet sind und keine Nachbarschaftsinformationen zwischen den einzelnen Punkten existieren. In dem vorgestellten Prozess werden zunächst die geometrischen Merkmale wie Normalenvektor und Krümmung aus den einzelnen Oberflächenpunkten gewonnen. Um die Normalenvektoren zu berechnen, wird ein bestehendes numerisches lokales Triangulationsverfahren zu dem neuen Verfahren *Lokale Triangulation mit direkten Nachbarpunkten* erweitert. Innerhalb dieses Verfahrens wird eine neue Methode entwickelt, mit der die störenden Punkte in der Nachbarschaft eliminiert und somit die direkten Nachbarn eines betrachteten Punktes ermittelt werden. Mit Hilfe dieser Methode werden lokale Triangulationen besonders an scharfen Kanten und Übergängen optimiert, so dass sowohl Normalenvektor als auch die Krümmung in einem Schritt effizient berechnet werden, ohne analytische Ansätze einzusetzen.

Mit Hilfe der gewonnen Merkmale wird die normalenbasierte Segmentierung durchgeführt, um die Kanten und die Übergänge zwischen geometrisch unterschiedlichen Teilobjekten in der Punktwolke zu detektieren. Die detektierten Kanten und Übergänge werden weiter verwendet, um charakteristische Teiloberflächen in einer Punktwolke einzugrenzen, so dass die Punkte einer Teiloberfläche in der krümmungsbasierten Segmentierung aufgeteilt und den zugehörigen geometrischen Grundprimitiven wie Ebene, Kugel oder Zylinder zugeordnet werden. Die Zuordnung entspricht dabei der Aufteilung des gescannten Maschinenteiles in seine Teiloberflächen, welche später zur Weiterverarbeitung parametrisiert und in die von CAD-Systemen bekannten Formate konvertiert werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

2D	Zwei Dimensional
3D	Drei Dimensional
kD-Baum	k-dimensionaler Baum
o.B.d.A.	ohne Beschränkung der Allgemeinheit
CAD	Computer Aided Design
OpenGL	Open Graphics Library
kNN	k nächsten Nachbarn

1 Einführung

1.1 Hintergrund der Forschung

In den letzten Jahrzehnten werden zur Visualisierung und Modellierung von Maschinenteilen (engl. machining parts), zur Erkennung von Gegenständen und zum Vermessen von komplexen Bauteilen, z.B. in der Photometrie [XuAl12][WLJD12] und der Medizin [MCK 11][SHG 06], Punktwolken eingesetzt. Punktwolken gewinnen damit immer mehr an Bedeutung bei der Beschreibung von dreidimensionalen Maschinenteilen in rechnergestützten Umgebungen. Die Weiterentwicklung der Scan-Technologie, sowie die Verbesserung der Objekterkennungs- und Flächenrückführungsverfahren erfordern immer effizientere und robustere Verfahren für die Beschreibung der Maschinenteile als dreidimensionale Objekten in rechnergestützten Umgebungen.

Eine Punktwolke (engl. points cloud) ist eine Menge von Punkten, die durch Abtasten [Riet05] [Boeh05][Wilk02] realer Objekte (z.B. Maschinenteile) erzeugt wird. Am häufigsten werden dazu Laser-Scan-Verfahren eingesetzt. Die generierte Punktwolke beschreibt dabei die abgetastete Oberfläche eines geometrischen Objektes oder einen Gegenstand, der aus mehreren zusammenhängenden geometrischen Teilobjekten bestehen kann. Bei dem abgetasteten Gegen-stand kann es sich auch um ein komplexes Objekt handeln, welches nicht aus gleichförmigen oder primitiven geometrischen Objekten besteht.

Die Punktwolken bieten eine Alternative für die visuelle Darstellung und Modellierung von dreidimensionalen Maschinenteilen gegenüber der herkömmlichen Beschreibung durch Polygone. Sie sind darüber hinaus als Tripel von Gleitkommazahlen einfach zu speichern und zwischen unterschiedlichen rechnergestützten Systemen leicht austauschbar.

Betrachtet man den kompletten Rekonstruktionsprozess ohne die Punktwolkenerfassung durch einen Scanner, kann der Ablauf zur Objekterkennung in drei Hauptphasen unterteilt werden:

Erste Phase: Um eine möglichst genaue Darstellung der Oberfläche eines Maschinenteiles zu erhalten, wird die Punktmenge in dieser Vorverarbeitungsphase von Abrissen und Rauschen beseitigt.

Zweite Phase: Nach der Vorverarbeitung in der ersten Phase erhält man eine Punktmenge, die als Grundlage für Algorithmen zur Gewinnung von Oberflächenmerkmalen dient, die auf geometrische 3D-Grundprimitive wie Ebenen, Zylinder, Kugeln oder Kegel in der Punktwolke hindeuten. In dieser Phase werden die Punkte der vorliegenden Punktmenge mit Hilfe der gewonnenen geometrischen Merkmale (engl. features) bezüglich der geometrischen 3D-Grundprimitive klassifiziert.

Dritte Phase: In der Computergrafik funktionieren die Prozesse zur Modellerstellung im Allgemeinen gut. Trotzdem sind nach der zweiten Phase bei rekonstruierten Oberflächen teilweise Nachkorrekturen nötig. Ursachen hierfür könnten Unzulänglichkeiten der eingesetzten Verfahren oder beispielsweise auch fehlende Punkte in der Punktwolke sein, die zu Fehlinterpretationen der Oberflächen führen.

Diese Arbeit befasst sich nach der obigen Unterteilung des Objekterkennungsprozesses mit der zweiten Phase, also der Merkmalerkennung (engl. feature recognition) an den Punkten und der Klassifizierung der Punkte anhand dieser Merkmale in CAD.

Das Abtasten mit Hilfe von 3D-Scannern ist zur Zeit sehr teuer, bietet aber den Vorteil, dass die Oberflächen von Maschinenteilen im Raum schnell abgetastet und durch Rechner zur Weiter-

verarbeitung eingelesen werden können, ohne die Maschinenteile zu berühren. Das Ergebnis eines solchen Abtastvorgangs ist eine ungeordnete Punktwolke, welche die Geometrie des abgetasteten Maschinenteiles in 3D-Punkten beschreibt. In der Praxis können Punktwolken aus mehreren Tausenden bis zu mehreren Millionen Punkten bestehen.

Unabhängig von der eingesetzten Scan-Technik, mit der die Oberfläche eines Maschinenteiles abgetastet wird, ist jedes Abtastverfahren fehlerbehaftet und führt zu Messfehlern der erzeugten Punktwolken, die sich bei einem Abtastvorgang nicht vermeiden lassen. Diese Fehler erschweren später die Rekonstruktion der Objekte in Phase 2 bei dem Objekterkennungsprozess. Nach aktuellem Stand liefert keine Scanner-Technologie exakte, rausch- und abrissfreie Daten der Objektoberfläche als Punktwolke.

Es sind zahlreiche Verfahren entstanden, um diese Unzulänglichkeiten zu lösen. Mit ihrer Hilfe werden die Eingabedaten im Vorfeld für die weitere Verarbeitung in Phase 2 vorverarbeitet (engl. preprocessing), bevor die tatsächliche Objekterkennung oder Rekonstruktion der Objektoberfläche stattfindet.

Wenn es um die Rekonstruktion der Oberfläche eines Maschinenteiles geht, stößt man in der Fachliteratur im Bereich Nachkonstruktion (engl. Reverse Engineering) auf zwei Begriffe: *Objekterkennung* und *Flächenrückführung*. Die Objekterkennung (engl. Object Recognition) unterscheidet sich wesentlich von Flächenrückführung (engl. Surface Reconstruction) in CAD. Bei der Flächenrückführung wird die Oberfläche eines Maschinenteiles durch Polygone, Freiformflächen oder durch Dreiecksnetze dargestellt. Bei der Objekterkennung hingegen werden geometrische 3D-Grundprimitive, wie z.B. Ebenen, Zylinder, Kugeln oder Kegel in einer Punktwolke durch Gewinnung von geometrischen Oberflächenmerkmalen identifiziert. Um dies zu realisieren, werden viele Informationen benötigt, wie z.B. Nachbarn von betrachteten Punkten, Krümmungen und Normalenvektoren an einzelnen Oberflächenpunkten, Kanten der gesuchten 3D-Grundprimitive oder Übergänge zwischen den in der Punktwolke vorkommenden Teilobjekten. Mit diesen Informationen oder Merkmalen kann man die Objektoberfläche näher spezifizieren. Dadurch kann klassifiziert werden, zu welcher Oberfläche von dem jeweiligen Primitiv ein Punkt gehört. Diese Merkmale sind geometrische Eigenschaften von Grundprimitiven, die unterschiedliche Flächentypen aufweisen. Um diese Merkmale zu gewinnen, werden mehrere Prozessschritte und Verfahren angewandt.

1.2 Problemstellung

Das Problem der Objekterkennung ist eine wichtige Herausforderung für die Computergrafik, bei der aus einer *unstrukturierten* Menge von 3D-Punkten geometrische Grundprimitive, wie z.B. Würfel, Kugel, Zylinder und Kegel, erkannt werden. Diese Aufgabenstellung ist nicht ohne Weiteres mit dem Verfahren der k-Nachbarn-Suche [PrSh85] und anschließender Triangulation der Punktmenge zu lösen. Um Teilobjekte in der Punktmenge identifizieren zu können, müssen geometrisch charakteristische Merkmale wie z.B. Normalenvektoren oder Krümmungen zu den einzelnen Oberflächenpunkten der 3D-Grundprimitive gesammelt werden. Die Schwierigkeit besteht dabei in der genauen Abschätzung solcher Merkmale anhand der Oberflächenstruktur der Objekte. Mit Hilfe dieser Merkmale werden Grenzen der 3D-Grundprimitive extrahiert und die Punktmenge zielführend in Teilpunktmengen aufgeteilt, um die Teilpunktmengen als geometrische 3D-Grundprimitive zu klassifizieren. Die hier verarbeitete unstrukturierte Punktmenge, welche die Oberfläche eines Maschinenteiles beschreibt, wird in dieser Arbeit als gegeben vorausgesetzt, und – wie weiter oben bereits erwähnt – z.B. durch 3D-Scanner erfasst und zur Verfügung gestellt.

1.3 Ziel und Zweck der Diplomarbeit

Ziel dieser Diplomarbeit ist es, für CAD-Anwendungen geometrische Merkmale (engl. features) aus einer *unstrukturierten* Punktmenge zu berechnen und die Punkte aus der Punktmenge mit Hilfe der geometrischen Merkmale in bekannten geometrischen 3D-Grundprimitive zu klassifizieren. Dies wird durch Gewinnung von Oberflächenmerkmalen wie Normalenvektoren und Krümmungen an den einzelnen Oberflächenpunkten erreicht. Anhand dieser Berechnungen wird die Punktmenge in Segmente aufgeteilt, mit deren Hilfe geometrischen 3D-Grundprimitive erkannt und anschließend rekonstruiert werden. Für diese Aufgabe wird ein Prototyp erstellt, der eine unstrukturiert vorliegende Punktmenge einliest und versucht, mit numerischen Verfahren statt der bisherigen analytischen Ansätze [Gomo09][Vanc03][Wilk02] den Prozess zur Objekterkennung zu beschleunigen. Der Prototyp soll dabei möglichst automatisiert, d.h. mit minimaler Benutzerinteraktion ausgeführt werden können.

Durch die Verwendung numerischer Verfahren können diese erkannten Grundprimitive später verschiedenen geometrischen Rechenoperationen unterzogen werden, um sie weiter zu verarbeiten.

1.4 Einschränkungen und Schwierigkeiten

I. Eine Punktwolke als Rohdaten besteht nur aus Punktkoordinaten (x, y, z) ohne geometrische und topologische Informationen über das gescannte Objekt. Das Problem ist die Bestimmung von Innen- und Außenseiten des durch Scanner erfassten Maschinenteiles, der als Punktwolke vorliegt oder das Zusammenfügen von mehreren kleinen Teilpunktmengen, welche durch vordefinierte Aufteilungskriterien entstehen, aber ähnliche Eigenschaften aufweisen und dadurch inhaltlich zusammenhängen.

II. Die Hauptschwierigkeit während dieser Arbeit ergab sich durch die Erweiterung bestehender Verfahren zur Triangulation, um störende Punkte zu eliminieren. Diese Erweiterungen werden in Kapitel 3.3.3.2 näher erläutert.

III. Eine Schwierigkeit der vorliegenden Arbeit besteht darin, dass der Gesamtprozess der Objekterkennung (siehe Kapitel 3.1) aus mehreren Teilschritten besteht und zu diesem Zweck zwei Prototypen entwickelt wurden, was sich als zeitaufwendig herausgestellt hat. Einer der Prototypen dient zur Ausführung aller vier Grundschritte zur Merkmalgewinnung und Klassifizierung von Punkten, wie in Kapitel 3.1 vorgestellt wird. Mit dem zweiten Prototyp können die Detail- und Zwischenergebnisse genauer analysiert werden, beispielsweise die Visualisierung von störenden Punkten in einer nicht-optimalen lokalen Triangulation.

IV. Die oben erwähnten Prototypen sind in C++ implementiert. Als Programmierungsumgebung wurde Microsoft Visual Studio 8.0 verwendet. Die Vorteile von C++ für die Implementierung im Rahmen dieser Arbeit sind die Ausführungsschnelligkeit insbesondere bei der Visualisierung, die Möglichkeit Operatoren zu überladen, sowie die weite Verbreitung auf vielen Plattformen mit vielen Codebeispielen. In der Programmierungssprache C++ stehen keine eigenen Bibliotheken zur Visualisierung von 3D-Daten zur Verfügung. Diese Lücke lässt sich aber durch die Verwendung von OpenGL schließen.

1.5 Vorteile der Forschungsarbeit

Derzeit stehen einige vollautomatische Verfahren zur Verfügung, die ohne Benutzerinteraktion geometrische 3D-Grundprimitive aus einer vorliegenden ungeordneten Punktmenge erkennen.

Die meisten von diesen Verfahren basieren funktional auf analytischen Lösungsansätzen. Diese sind robust, aber die Berechnung der Oberflächenmerkmale der gescannten Objekte sind aufwendig bzw. ineffizient.

Ein weiterer Nachteil der existierenden Verfahren ist, dass sie vor dem eigentlichen Objekterkennungsprozess einen Vorbereitungsschritt mit einer vollständigen Triangulation benötigen und das Ergebnis zwischengespeichert werden muss. Das Ergebnis der vollständigen Triangulation ist ein Dreiecksnetz, mit dessen Hilfe die Oberflächenmerkmale an den einzelnen gescannten Punkten abgeschätzt werden. Zur Triangulation der vorliegenden Punktmenge kann allerdings ein beliebiges Triangulationsverfahren eingesetzt werden.

Im Rahmen dieser Diplomarbeit werden eine neue Methode entwickelt und bestehende Verfahren erweitert, um die Objekterkennung aus unstrukturierten Punktwolken zu optimieren. Im Vergleich zu den oben genannten Nachteilen der bestehenden Verfahren, haben die Optimierungen und Erweiterungen folgende Vorteile:

a) Effizientere Lösungen zur Objekterkennung bieten numerische Verfahren. Es besteht aber der Nachteil, dass während der Informationsgewinnung über Oberflächenmerkmale Schwierigkeiten auftreten, beispielsweise bei den durch einen Scanner erzeugten Punkten an scharfen Kanten der gesuchten geometrischen 3D-Grundprimitive und bei Übergängen zwischen geometrisch unterschiedlichen Teilobjekten, welche in der vorliegenden Punktmenge enthalten sind. In dieser Arbeit werden zwei solche numerische Verfahren aus [Vanc03] implementiert und analysiert. Es handelt sich dabei um die Verfahren *Local Delauney Triangulation* und *Local Center Triangulation.* Mit der neu entwickelten Methode werden Schwächen dieser beiden Verfahren beseitigt.

b) Eines der oben genannten Verfahren wird im Objekterkennungsprozess, der im Rahmen dieser Diplomarbeit implementiert wurde, eingesetzt und so erweitert, dass daraus ein neues Verfahren mit der Bezeichnung *Lokale Triangulation mit direkten Nachbarpunkten* entstanden ist. Innerhalb dieses neuen Verfahrens wird eine neue Eliminationsmethode entwickelt, mit deren Hilfe eine lokale Triangulation optimiert wird. Die Optimierung führt dazu, dass die zur Segmentierung der Punktwolke benötigten geometrischen Oberflächenmerkmale wie Normalenvektor und die Krümmung mit dem Satz von Gauß-Bonnet ohne aufwendige analytische Verfahren in einem Schritt an dem jeweiligen Oberflächenpunkt effizient berechnet werden.

c) Das neue Verfahren erleichtert außerdem die Segmentierung der Punktwolke, in dem die zur Segmentierung nötigen Prüfungen für Abweichung der Normalenvektoren und für gleiche Krümmungswerte zwischen direkten Nachbarpunkten durchgeführt werden. Die Segmentierung ist für die Klassifizierung der Punkte in dem Objekterkennungsprozess erforderlich.

d) Der gesamte Prozess zur Objekterkennung aus unstrukturierten Punktwolken wird damit beschleunigt, so dass trotz der Nachteile von numerischen Verfahren effiziente Alternativen zur Rekonstruktion von Objekten aus unstrukturierten Punktwolken zur Verfügung stehen.

e) Mit dem entwickelten Prototyp sind zusätzliche Verbesserungs- und Erweiterungsmöglichkeiten im Rekonstruktionsprozess gegeben. Denkbar wäre beispielsweise die Konvertierung parametrisierten Daten, die von dem Prototyp gewonnenen werden, in andere von CAD-Systemen verwendete Formate wie DXF oder STEP.

g) Außerdem besteht die Möglichkeit, den Prototyp um weitere CAD Feature-Erkennungen zu erweitern.

1.6 Gliederung

Für eine übersichtliche Beschreibung und Darstellung des Stoffes, ist der Inhalt dieser Diplomarbeit in sechs Kapitel wie folgt aufgegliedert.

Kapitel 1 : Das erste Kapitel dient als Einleitung für die vorliegende Diplomarbeit und gibt einen groben Überblick über die Hintergründe der Objekterkennung und die Zielsetzung sowie die Problemstellung der Diplomarbeit.

Kapitel 2 : Nach der Einführung werden in diesem Kapitel grundlegende Definitionen, Informationen, die für Entwicklung des im Kapitel 3 vorgestellten Konzeptes benötigt sind und zur Berechnung von Normalenvektoren bisher existierende Verfahren präsentiert.

Kapitel 3 : Die einzelnen Grundschritte der Entwicklung des Konzeptes zur Objekterkennung werden in eigenen Abschnitten detailliert behandelt. Es führt außerdem die neu entwickelte Methode zur Elimination der störenden und überflüssigen Nachbarpunkte ein.

Kapitel 4 : Die Implementierung des Verfahrens und Visualisierung der Ergebnisse aus Kapitel 3 wird in diesem Kapitel beschrieben.

Kapitel 5 : In diesem Kapitel sind Analyse und Ergebnisse der Grundschritte des Objekterkennungsprozesses an einem Beispiel mit einer künstlichen Punktwolke zu finden.

Kapitel 6 : Eine Zusammenfassung und ein Überblick über zukünftige Forschungsthemen werden in diesem Kapitel gegeben.

Im Abschnitt Appendix werden die Fachbegriffe bzw. weiterführende Informationen untereinander gegliedert und erläutert.

2 Die Grundlagen

In diesem Abschnitt werden die Grundkenntnisse für den Objekterkennungsprozess, der in Rahmen dieser Diplomarbeit implementiert wird, erläutert.

2.1 Feature-Erkennung

"Der Begriff *Objekterkennung* beschreibt Verfahren zum Identifizieren eines bekannten Objektes innerhalb eines Objektraums mittels optischer, akustischer oder anderer physikalischer Erkennungsverfahren". So wird z.B. das Vorhandensein eines Objektes in einem Bild oder einem Bereich aber auch dessen Position und Lage bestimmt [Wiki12b].

Wie funktioniert die Erkennung von scharfen Bereichen, mit dem Ziel, geometrische Grundprimitive oder komplexe Objekte zu erkennen, die sich innerhalb einer Punktwolke angrenzen können, wobei eine Kante als Verbindungsstelle vorkommt. Ein einzelnes primitives Objekt zu erkennen, ist kein großes Problem. Die Hürde der Objekterkennung ist genau an scharfen Kanten, wo sich die Oberflächen von mehreren Objekten kreuzen und daher die Erkennung der Grenzen zu den Objekten stark beeinträchtigt wird.

Die Forschungsarbeit [WHH 10], die man zu einer hochrangigen Arbeit zählen kann, da die wurde basierend auf einer internationalen Konferenz erstellt, in dem Spezialisten der Merkmalerkennung von scharfen Bereichen beteiligt waren. Die Objekterkennung ist in der Arbeit ausführlich erklärt und unten wie folgt.

Objekterkennungsprozess besteht o.B.d.A. aus zwei Grundschritten:

I. **Extraktion der Merkmale:** Dieser Schritt geschieht wiederum anhand der drei genannten Teilschritte:

   1. Datenstruktur der Punkte (Raumpartitionierung durch *kD*-Baum und Speicherung der Ergebnisse (Punkte) in einem Riemmanian Graph (mehr Details über Riemmanian Graph kann man in [WHH 10] finden).
   2. Danach soll lokale Nachbarschaften zum jeweiligen Punkt P berücksichtigen (k)
   3. Anschließend wird diskretes Gauß-Mapping durchgeführt (σ) (mehr Details über Gauß-Mapping kann man in [WHH 10] finden).

II. **Erkennung von Feature:** Dieser Grundschritt läuft eventuell parallel zu dem obigen Teilschritt 3 (diskretes Gauß-Mapping). Bei der Feature Erkennung werden die von sicheren Punktkandidaten durch Flachheitstest (engl. flatness test) eliminiert, welche sich in ebenen Regionen befinden. Anschließend wird eine spezielle Behandlung durch Gauß Clustering zur Elimination der weiteren übrig gebliebenen Punkten durchgeführt.

In der Forschungsarbeit geht es um Merkmalerkennung mithilfe des Gauß-Verfahrens und einer zusätzlichen Erweiterung, die die Merkmal-Erkennung unterstützten soll. Als (scharfe) *Merkmale* werden *Kanten*, *Linien* zwischen zwei Oberflächen oder *Ecken***,** die zwischen drei oder mehr Oberflächen liegen, gemeint. Diese Angaben beantworten schon grob die Frage, wie eine Objekterkennung aussieht oder was eine Objekterkennung eigentlich ist!

Als Ergebnis wurde gezeigt, welche Art von Geometrie-Oberflächen zur Scharf-Merkmal-Erkennung in Punktwolken in Frage kommt. Primitive Körper und andere komplexe Modelle sind hier im Visier.

In der Arbeit kann man sehen, wie ein Punkt bezüglich der Eigenschaft, *dass er sich im Bereich eines scharfen Bereiches befindet*, einfach und schnell untersucht werden kann.

Die Hauptfrage ist hier, ob der für die Stichprobe ausgewählter Punkt *P* eine "*scharfe*" *Eigenschaft* besitzt bzw. ob der Punkt *P* sich im scharfen Bereich befindet. Ein Punkt, der nah zu diesem Bereich liegt, aber nicht da drauf liegt, so wird dieser Punkt entweder mit Flatness-Test oder in einem weiteren präziserem Prozess durch Gauss Clustering eliminiert. Die Zuordnung eines Punktes *P* zur "*scharfen*" *Eigenschaft* hängt von genau zwei Faktoren ab, nämlich von *k* und σ.

Um den in dieser Diplomarbeit vorgestellten Features-Erkennungsprozess zu implementieren, werden die folgenden Grundlagen benötigt.

2.2 Koordinatensystemen

Mit Hilfe eines Koordinatensystems kann man die Positionen der Punkte in einer Ebene oder in einem Raum durch Angabe der Zahlen- oder Größenwerten eindeutig bestimmen [HKKH12a] [EmOb12]. Die Dimension des Raumes drückt die Anzahl der zur eindeutigen Positionierung notwendigen Werte aus. Die Position wird dabei durch Koordinaten ausgedrückt. Den Punkt in einem Koordinatensystem, in dem alle Koordinaten den Wert Null bekommen, bezeichnet man als Koordinatenursprung, öfter auch als Nullpunkt genannt.

Sei *P* ein belibiger Punkt, dessen Position wie unten in unterschiedlichen Koordinatensystemen angegeben wird, vgl. Abbildungen 1-4.

Mit Hilfe der Kugelkoordinaten [HKKH12b] (auch sphärische Polarkoordinaten genannt) kann man die Position des Punktes *P*, der sich innerhalb der Kugel befindet, eindeutig bezeichnen. Dafür braucht man drei Größen, die als Koordinaten verwendet werden; also einen konstanten Abstand *r* zu einem Ursprung, einen Winkeln θ (auch Azimutwinkel) zwischen dem Punkt *P* und Azimut-Achse und einem Winkel φ (auch Polarwinkel), den der Punkt P mit der Polarachse (roter Vektor in Abbildung 1) einschließt. Man schreibt kurz *P(r, φ, θ)*. Die Kugelkoordinate wird als Oberbegriff für *Polarkoordinaten in der Ebene* und *Zylinderkoordinaten* verwendet.

Liegt der Punkt *P* hingegen wie in Abbildung 2 innerhalb eines Zylinder, dann sind zwei Abstände und ein Winkel erforderlich, um die Lage des Punktes *P* in Zylinderkoordinaten [HKKH12d] eindeutig anzugeben, man schreibt kurz *P(r, φ, z)*.

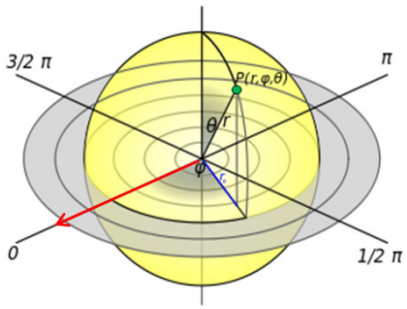


Abbildung 1: Kugelkoordinaten [Wiki12]

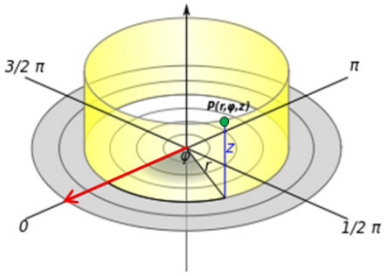


Abbildung 2: Zylinderkoordinaten [Wiki12]

Das wohl am häufigsten verwendete Koordinatensystem ist das kartesische Koordinatensystem [EmOb12], das aus drei aufeinander senkrecht stehenden Achsen besteht. Während die Positionen des Punktes *P* im Kugelkoordinatensystem mit (*r*, φ, θ)-Tupel eindeutig festgelegt werden kann, wird die Position in einem dreidimensionalen kartesischen Koordinatensystem als (*x*, *y*, *z*)-Tripel mit reellen Werten eindeutig erfasst (siehe Abbildung 3).

Zur eindeutigen Festlegung der Position des Punktes P in der Ebene der Polarkoordinaten [HKKH12c] genügen zwei Größen. Die erste Größe ist der Abstand r des Punktes von dem Ursprung, die zweite Größe ist der Polarwinkel ϕ von der Polarachse (auch als Azimut genannt) (siehe Abbildung 4). Man schreibt kurz $P(r, \phi)$, um die Position des Punktes P anzugeben.

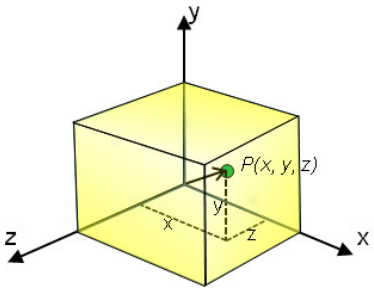


Abbildung 3: Kartesisches Koordinatensystem

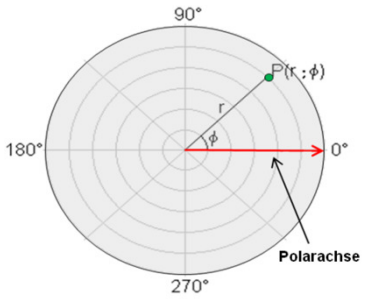


Abbildung 4: Ebene Polarkoordinaten [Wiki12]

Der innerhalb des Konzeptes eingeführte Objekterkennungsprozess wird direkt auf die Menge der Punkte angewendet. Diese Punkte sind z.B. durch einen Scanner erfasst und in dem dreidimensionalen kartesischen Raum mit x, y, und z-Koordinaten angegeben. Das in Rahmen dieses Prozesses implementierte Verfahren (siehe Kapitel 3.3.3.1) ermittelt intern eine Ebene zur Sortierung der Nachbarpunkte eines betrachteten Punktes für die lokale Triangulation. Diese Ebene heißt dort Best-Projektionsebene und entspricht der in diesem Kapitel vorgestellten Ebene der Polarkoordinaten, welche aber im 3D aufgespannt wird. D.h. zu einer lokalen Triangulation werden die Punkte aus dem dreidimensionalen kartesischen Koordinatensystem – ohne das Koordinatensystem zu verlassen – in die temporären ebenen Polarkoordinaten übertragen, um die Punkte dort untereinander nach Polarwinkeln zu sortieren. Dadurch entsteht ein Polygonzug aus Nachbarpunkten (siehe Abbildung 36), der in der Ebene liegt. Anschließend wird auf den Nachbarpunkten in derselben Ebene mit der neu entwickelten Eliminationsmethode (siehe Kapitel 3.3.3.2) operiert, um die lokale Triangulation zu optimieren, in dem man ein Dreiecknetz bei der Triangulation mit *direkten* Nachbarpunkten des jeweiligen betrachtenden Oberflächenpunktes aufbaut.

Die Ebene von Polarkoordinaten wird mit der im Kapitel 2.3.1 beschriebene Methode ermittelt und wird in dem Triangulationverfahren (siehe Abschnitt 3.3.3.1) eingesetzt, um die Punkte in dem drei-dimensionalen Raum nach Polarwinkeln zu sortieren, d.h. dabei geht es um eine Ebene im 3D-Raum. Die Ebene kommt aber in dem Triangulationsverfahren mit dem Namen Best-Projektionsebene (siehe Kapiteln 2.3.1 und 3.3.3.1) vor.

Ebene der Polarkoordinaten ↔ Best-Projektionsebene

2.3 Sortierung der Punktwolke und Identifizierung der direkten Nachbarpunkte

Für eine optimale lokale Triangulation werden die Nachbarpunkte eines betrachteten Punktes der Reihe nach auf eine Ebene projiziert, in dieser Ebene sortiert und anschließend die direkten Nachbarpunkte des betrachteten Punktes identifiziert. In diesem Kapitel werden die Bestimmung der Projektionsebene und die Grundidee zur Identifizierung der *direkten* Nachbarpunkten (zur Begriffserklärung, siehe auch Appendix D) vorgestellt.

2.3.1 Best-Projektionsebene im 3D-Raum

Die hier gesuchte Ebene kommt in verschiedenen Gebieten vor, wie z.B. in der Physik, Analysis, Geodäsie etc. und wird mit unterschiedlichen Namen wie Anpassungsebene, Ausgleichende Ebene [Drix93] usw. verwendet. Gesucht ist eine geeignete Ebene, die man später bei dem im Kapitel 3.3.3.1 eingeführten Triangulationsverfahren braucht, um die zu triangulierenden Punkte zu sortieren oder um die 3D-Punkte in 2D speichern zu können, in der die zu triangulierenden Punkte zuerst in so eine Ebene projiziert werden (siehe Kapitel 2.4). Diese Ebene wird Best-Projektionsebene genannt [Vanc03], da die triangulierenden Nachbarpunkte im Raum jedes Mal in so eine Ebene projizieren werden und die unter den in Frage kommenden Ebenen als die "bestgeeignete" Ebene ausgewählt wird, deren Ermittlung im folgenden beschrieben ist.

Sei P der betrachtete Punkt und $P_1 \ldots P_n$ die Nachbarpunkte von P. Die Best-Projektionsebene wird bezüglich dieser Punkte ermittelt. Beginnend mit einem betrachteten fix gewählten Punkt P wird ein weiterer fester Punkt P_1 (nahester Nachbar zu P) für die zu bestimmende Best-Projektionsebene gewählt. Um eine Ebene aufzuspannen braucht man einen weiteren dritten Punkt P_i, der nicht auf der Geraden liegen darf, die die Punkte P und P_1 definiert. Dieser Vorgang wird für jedes P_i wiederholt durchgeführt. Für jeden neu ausgewählte Punkt P_i wird eine neue Projektionsebene aufgespannt. Der dritte Punkt P_i, bei dem der Projektionsabstand minimal ist, spant zusammen mit den vorher bestimmten Punkten (P und P_1) die Best-Projektionsebene auf, in der auch später alle Punkte projiziert werden. Der minimale Projektionsabstand ist dabei die kleinste Summe der direkten Abstände aller Punkte zur jeweiligen Projektionsebene bezüglich eines probeweise gewählten dritten Punktes P_i. Der untere Pseudocode stellt die Bestimmung der Best-Projektionsebene dar.

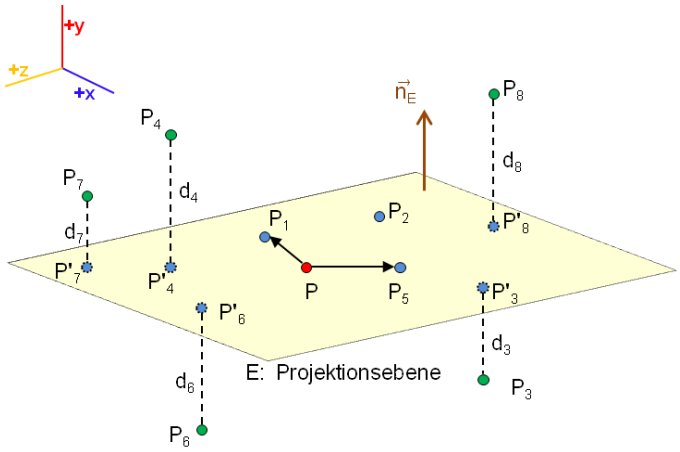


Abbildung 5: Eine durch die Punkte P, P_1 und P_5 aufgespannte Projektionsebene (in 3D)

Die blauen, gestrichelten Punkte aus der Abbildung 5 (die Punkte P'_3, P'_4 und von P'_6 bis P'_8) deuten auf die projizierten Punkte an, welche vor der Projektion nicht in der Ebene lagen. Die drei Punkte, die die Ebene aufspannen liegen immer in der Ebene (z.B. P, P_1 sowie P_5). Sicher können daneben auch Punkte sein, die bei dem Aufspannen der Ebene nicht teilnahmen, aber schon in der Ebene liegen (die blaue Punkte, wie P_2).

Bestimmung der Best-Projektionsebene bezüglich des betrachtenden Punkt P und seinen Nachbarpunkte $P_1 \ldots P_n$:

→ P ist der aktuelle Punkt und damit der erste Punkt der Projektionsebene.

→ P_1 ist der „naheste“ Nachbar von P und gleich der zweite Punkt der Projektionsebene.

→ Wähle den dritten Punkt aus restlichen Nachbarpunkten von P aus. P und P_1 spannen mit dem ausgewählten Nachbarpunkt P_i eine neue Ebene auf, in die dann jeweils alle restlichen Punkte projiziert werden.

$$\min\left(\sum_{i=0}^{n} d_i\right) \quad \Rightarrow \quad \text{Projektionsebene } E \text{ ist dann die "}\textit{Best-Projektionsebene}\text{"} \qquad (2.1)$$

2.3.2 Umlaufrichtung und Orientierung

Der Begriff *Umlaufrichtung* oder *Umlaufsinn, auch Drehrichtung oder Drehsinn* haben je nach Anwendungsgebiet unterschiedliche Bedeutungen. Die Umlaufrichtung drückt in der Geometrie, ausgehend von einem festen Punkt, die Bewegungsrichtung der Punkte auf einer Kreisfläche um ihren Mittelpunkt aus.

Gegeben seien drei Punkte A, B und C in einer Ebene. Bildet man ein Dreieck Δ(ABC) in dem man nacheinander A, B und dann C verbindet, ist die Umlaufrichtung des Dreiecks nach dieser Reihenfolge festgelegt. Verlaufen die Punkte des Dreiecks ausgehend von Punkt A, B und C *gegen den Uhrzeigersinn*, dann ist das Dreieck *rechtsdrehend* (siehe Abbildung 6 (links)), während die Punkte in Abbildung 6 (rechts) *im Uhrzeigersinn* laufen, daher das Dreieck *linksdrehend*.

Abbildung 6: Drehsinn: (links) gegen den Uhrzeigersinn, (rechts) im Uhrzeigersinn

Die Umlaufrichtung der drei Punkte, welche eine Ebene (entspricht einer Dreiecksfläche) aufspannen, legt die Richtung des Flächennormalenvektors fest. Die Orientierung der Ebene wird durch den Normalenvektor der Ebene und einen festen Bezugspunkt, der sich gegenüber der Ebene befindet, bestimmt. Ein Bezugspunkt kann hierbei das Auge des Betrachters, eine Lichtquelle oder ein beliebiger Punkt mit einer festen Position im Raum sein. Verlaufen die aufspannenden Punkte gegen den Uhrzeigersinn, dann gibt die Richtung des Normalenvektor (Kreuzprodukt) der Punkte nach dieser Umlaufrichtung automatisch die Orientierung der Ebene an, wenn sich die Ebene im 2D-Raum befindet, d.h. die Richtung des Normalenvektors gibt die Vorderseite (Blickrichtung) der Fläche an. Somit kann man zwischen Rück- und Vorderseite der Ebene unterscheiden. Befindet sich die Ebene dagegen im 3D-Raum, dann ist zusätzlich ein fester Bezugspunkt erforderlich, um die Orientierung der Ebene festzustellen.

Bei einer positiven Orientierung zeigt die Richtung des Normalenvektors der (Dreieck-)Fläche in die Ebene hinein, bei negativer Orientierung dagegen aus der Ebene hinaus. Die Fläche ist dabei durch drei aufeinanderfolgende Punkte aufgespannt. Bei der positiv orientierten Dreiecksfläche Δ(ABC) im Beispiel von Abbildung 6 (links), ergibt das Kreuzprodukt zwischen den Vektoren $\overrightarrow{BA}$ und $\overrightarrow{BC}$ den Normalenvektor von der Dreiecksfläche, der aus der Ebene hinaus gerichtet ist.

Umlaufrichtung gegen Uhrzeigersinn	=>	positive Orientierung (in 2D)
Umlaufrichtung im Uhrzeigersinn	=>	negative Orientierung (in 2D)

Eine Ebene oder Fläche ist im Hinblick auf einen gegebenen Bezugspunkt positiv orientiert (*positive Orientierung*), wenn deren Normalenvektor in Richtung des Bezugspunktes gerichtet ist. Zeigt der Normalvektor der Fläche in die entgegengesetzte Richtung vom Bezugspunkt, dann ist die Ebene hinsichtlich des Bezugspunktes negativ orientiert (*negative Orientierung*).

Zur Feststellung der Orientierungen von Flächen in einem dreidimensionalen globalen kartesischen Koordinatensystem, gelten die folgenden drei Bezugspunkte bzgl. des Gegenuhrzeigersinns:

a. ein Punkt im Unendlichen in negativer x-Richtung,
b. ein Punkt im Unendlichen in negativer y-Richtung und
c. ein Punkt im Unendlichen in negativer z-Richtung

Zeigt ein Flächennormalenvektor in die Richtung eines jeweiligen Bezugspunktes, dann ist die entsprechende Fläche hinsichtlich dieses Bezugspunktes positiv orientiert. Solche Flächen kommen in Polygonzügen (siehe Abbildung 48) vor. Polygonzüge entstehen von einem betrachtenden Oberflächenpunkt aus bei lokaler Triangulation mit den Nachbarpunkten in einer Best-Projektionsebene (siehe Kapitel 2.3.1 und Abbildung 40). Die vorkommenden Flächen werden als Dreiecksflächen aus den aufeinanderfolgenden Polygoneckpunkten gebildet (siehe Abbildung 16 und Abbildung 50):

In Abbildung 7 sieht man die Normalenvektorrichtungen der Flächen mit positiver Orientierung zu den oben definierten Bezugspunkten.

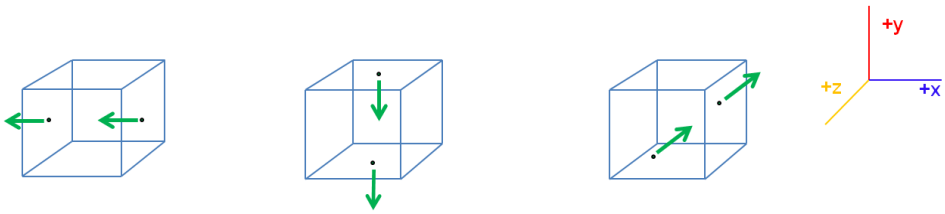


Abbildung 7: Richtung der Normalenvektoren der positiv orientierten Flächen bzgl. Bezugspunkte im Unendlichen

Da eine Fläche in einem dreidimensionalen kartesischen Raum beliebig gerichtet sein kann, zeigen die Normalenvektoren üblicherweise nicht genau in die Richtung einer der globalen kartesischen Koordinatenachse, auf der im Unendlichen der jeweilige Bezugspunkt liegt. Liegt ein Polygonzug in einer Ebene (wie in Abbildung 36 dargestellt), verlaufen alle Vektoren parallel, aber die Normalenvektoren der Dreiecksflächen können in entgegengesetzte Richtungen zeigen, je nachdem wie die entsprechende Dreiecksfläche im Polygonzug orientiert ist. Daraus folgt, dass die in eine Ebene eingebetteten Dreiecksflächen in einem Polygonzug *denselben* Bezugspunkt haben und sie weisen zu diesem Bezugspunkt eine negative oder positive Orientierung auf. Der richtige Bezugspunkt aus den drei definierten Bezugspunkten (siehe oben) wird herausgefunden, in dem entweder die Richtung der Ebene, in der diese Flächen eingebettet sind, an eine der globalen kartesischen Koordinatenachsen näherungsweise angepasst wird. Danach werden alle Punkte aus der ursprünglichen Ebene in die angepasste Ebene übertragen, indem die Richtung der Dreiecksflächen von jeweils drei Punkten eine nach der anderen (siehe Abschnitt 2.4) angepasst wird. Damit werden die Orientierungen der Flächen zum herausgefunden Bezugspunkt geprüft. Die Orientierungsprüfung ist nötig zur Identifizierung von *direkten Nachbarpunkten* bei einer lokalen Triangulation, so dass mit der neu entwickelten Eliminationsmethode die *überflüssigen* und *störenden* Punkte aus der Triangulation entfernt werden können (siehe Kapitel 3.3.3.2).

Man passt die Dreiecksflächen nacheinander einer der globalen kartesischen Koordinatenachsen näherungsweise an, indem die jeweilige Fläche in die geeignete, von zwei der globalen

x-*y*-*z*-Koordinaten aufgespannte Ebene projiziert wird, wie in Abschnitt 2.4 beschrieben ist. Dabei wird eine der *x*-*y*-*z*-Koordinaten der projizierten Punkte auf Null gesetzt. Das hat den Vorteil, dass das zur Orientierungsprüfung nötige Kreuzprodukt der zusammenhängenden Vektoren in Dreiecken in 2D (siehe Abschnitt 2.4) durchgeführt und dabei gleichzeitig der richtige Bezugspunkt für alle Flächen eindeutig festgelegt wird. Das Resultat des Produktes in 2D gibt dann zusammen mit dem festgelegten Bezugspunkt die Orientierung der Fläche in dem Polygonzug an, der wiederum in einer dreidimensionalen Best-Projektionsebene (siehe Abbildung 48) liegt. Die auf Null gesetzte und anschließend weggelassene Koordinate setzt den Bezugspunkt auf der Achse dieser Koordinate fest (siehe Abbildung 7).

Die Orientierungsprüfung der Dreiecksflächen kann mit Hilfe der Rechte-Hand-Regel (siehe Abbildung 11) veranschaulicht werden, wenn der Bezugspunkt zur Bestimmung der Orientierung festgelegt ist und alle Punkte bei einer lokalen Triangulation (Kapitel 3.3.3.1) in einer Ebene liegen. Die Richtung des Daumens bei einer positiven Flächenorientierung hinsichtlich eines Bezugspunktes nach der Rechte-Hand-Regel ist in Abbildung 12 an dem Beispiel-Würfel mit violett-farbigen Vektoren dargestellt. Die Vektoren in Abbildung 12 zeigen genau in die Gegenrichtung des Normalenvektors der jeweiligen Dreiecksfläche, wenn die Rechte-Hand-Regel angewendet wird, d.h. die Richtung des Daumens zeigt bei der Rechte-Hand-Regel immer in die Gegenrichtung eines Normalenvektor einer Fläche bei einer Orientierungsveranschaulichung (siehe Abbildung 11 und vgl. Abbildung 12 mit Abbildung 7).

Zeigt der abgespreizte Daumen bei Verwendung der Rechte-Hand-Regel für die drei in einem Polygonzug aufeinanderfolgenden Punkte in eine, der in Abbildung 11 dargestellten Richtungen, dann ist die Orientierung der von diesen Punkten aufgespannten Dreiecksfläche positiv, ansonsten negativ. Die Richtung hängt wiederum von der Ebene ab, in der die Punkte liegen. Die Ebenen (Würfeloberflächen) sind angepasste Ebenen, d.h. ihre Normalenvektoren zeigen genau in die Richtung einer der globalen kartesischen Koordinatenachsen.

2.3.3 Normalenvektorbestimmung im Allgemein

In reellen Vektorräumen (siehe Appendix A) sind die skalaren Größen reelle Zahlen. Im Gegensatz zum Skalarprodukt unterscheiden sich die Resultate vom Vektorprodukt (auch *äußeres Produkt* genannt) je nach der Dimension der Vektoren. Während das Kreuzprodukt in 3D eine vektorielle Größe ist, ist es in 2D eine skalare Größe, die negativ oder positiv sein kann. Diese Eigenschaft vom Kreuzprodukt in 2D wird ausgenutzt, um die *Orientierung* einer Dreiecksfläche mit 3D-Koordinaten (siehe Kapitel 2.4) zu bestimmen. Im Folgenden wird zunächst die Bedeutung des Kreuzproduktes in 3D und in 2D veranschaulicht.

a) Kreuzprodukt in 2D

Während das Resultat eines Kreuzproduktes zweier Vektoren $\vec{a}$ und $\vec{b}$ in einem dreidimensionalen kartesischem Vektorraum ein Vektor ist, ergibt das Kreuzprodukt von zwei Vektoren $\vec{a}$ und $\vec{b}$ in 2D einen positiven oder negativen Wert. Dieser Wert gibt an, ob die Richtung des Produktes dieser beiden Vektoren vom Betrachter aus in die Ebene hinein (symbolisch bezeichnet mit ⊗) (negativ) beziehungsweise aus der Ebene heraus (symbolisch bezeichnet mit ⊙) (positiv) zeigt (siehe Abbildung 8).

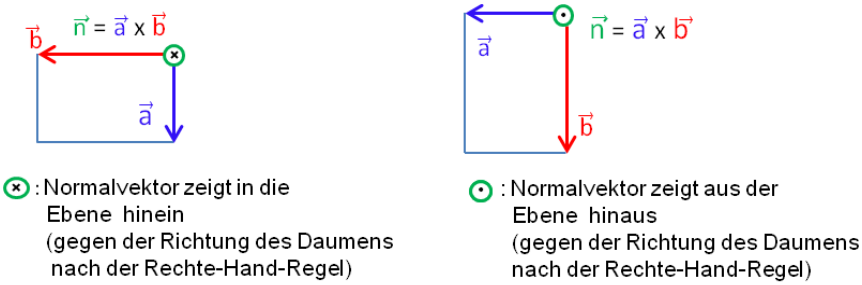


Abbildung 8: Das Kreuzprodukt in 2D

Mit Hilfe der oben genannten Eigenschaft des 2D-Kreuzprodukts wird sowohl die Richtung des Normalenvektors einer Dreiecksfläche als auch die Orientierung der Fläche bestimmt. Überträgt man die drei aufeinanderfolgenden Punkte nach der Methode in Kapitel 2.4 aus dem dreidimensionalen in den zweidimensionalen kartesischen Raum und berechnet das Kreuzprodukt in 2D, dann gibt das Vorzeichen des Kreuzproduktes an, ob die Dreiecksfläche dieser Punkte negativ oder positiv orientiert ist. Ist das Kreuzprodukt positiv, dann ist Fläche positiv orientiert (*positive Orientierung*), sonst negativ orientiert (*negative Orientierung*).

Die Orientierung einer Dreiecksfläche wird mit Hilfe der Normalenvektorrichtung und des festgelegten Bezugspunktes (siehe Kapitel 2.3.2) bestimmt. Die Orientierung einer Dreiecksfläche zu einem vordefinierten Bezugspunkt mit der Drei-Finger-Regel zu prüfen, ist nicht leicht nachvollziehbar. Aus diesem Grund wird hier eine neue Regel eingeführt, um die Orientierungen der Flächen leichter zu veranschaulichen. Diese Regel wird als *Rechte-Hand-Regel* bezeichnet und dient in der Physik zur anschaulichen Richtungsbestimmung des Magnetfeldes um einen stromdurchflossenen Leiter.

b) Kreuzprodukt in 3D

In der analytischen Geometrie [Koec97] ist das Resultat eines Kreuzproduktes von zwei Vektoren $\vec{a}$ und $\vec{b}$ im drei-dimensionalen Raum ein Vektor in *x*-, *y*-, z-Koordinaten. Diesen Vektor nennt man auch Normalenvektor $\vec{n}$. Er steht senkrecht auf der von den Vektoren $\vec{a}$ und $\vec{b}$ aufgespannten Ebene.

$$\vec{n} = \vec{a} \times \vec{b} = |\vec{a}||\vec{b}| \sin\theta \qquad (2.2)$$

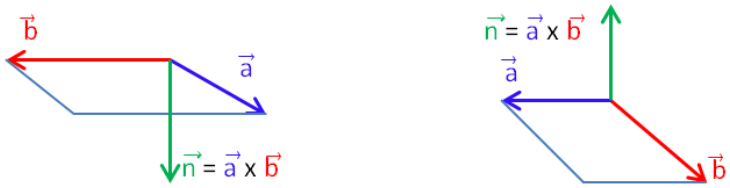


Abbildung 9: Kreuzprodukt der Vektoren $\vec{a}$ und $\vec{b}$ in 3D

Die Länge des Normalenvektors $\vec{n}$ (Betrag des Kreuzproduktes) entspricht dabei dem Inhalt der Fläche. θ ist der von den Vektoren $\vec{a}$ und $\vec{b}$ eingeschlossene Winkel. Da θ zwischen 0° und 180° liegt ($0° < \theta < 180°$) und $\sin\theta$ in diesem Bereich immer positiv ist, wird der Inhalt der Fläche immer positiv sein.

Die Richtung des Vektors $\vec{n}$ kann man mit Hilfe der bekannten Drei-Finger-Regel veranschaulichen.

c) Drei-Finger-Regel

Die Drei-Finger-Regel ist eine Merkregel zur Veranschaulichung der Richtungsbestimmung des Normalenvektors, der sich aus dem Kreuzprodukt = x ergibt. Während der Daumen der rechten Hand in Richtung des Vektor zeigt und gleichzeitig der Zeigerfinger in Richtung des Vektors , weist der senkrecht zu beiden abgespreizte Mittelfinger in Richtung des Normalenvektor (Abbildung 10).

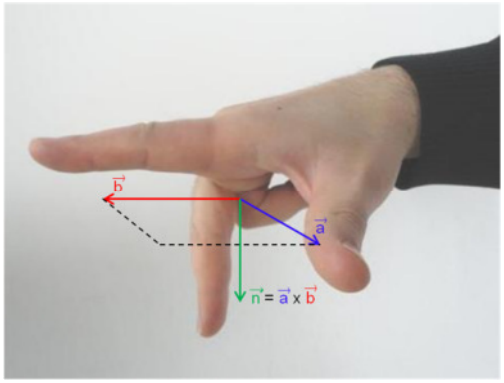


Abbildung 10: Drei-Finger-Regel zur anschaulichen Bestimmung der Richtung des Kreuzproduktes

d) Die Rechte-Hand-Regel zur anschaulichen Bestimmung der Flächenorientierung

Man fasst mit den gekrümmten Fingern die drei in einer Ebene liegenden, aufeinanderfolgenden Punkte der Reihe nach um, ohne die Faust zu ballen. Laufen die drei aufeinanderfolgenden Punkte gegen den Uhrzeigersinn (siehe Abbildung 11), dann zeigt der abgespreizte Daumen der rechten Hand nach dieser Regel nach oben (aus der Ebene heraus) sonst zeigt der Daumen nach unten (in die Ebene hinein), wenn die Punkte im Uhrzeigersinn laufen.

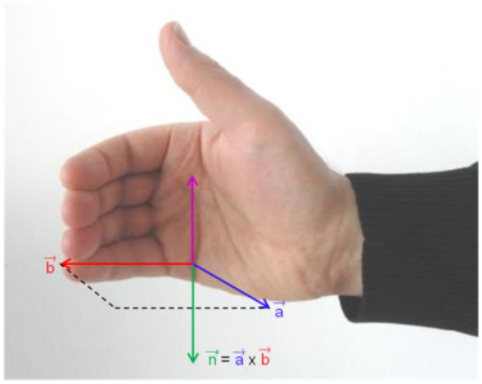


Abbildung 11: Die Rechte-Hand-Regel zur Veranschaulichung der Flächenorientierung zu einem Bezugspunk (Richtung des Daumens (violetter Vektor) zeigt genau in die Gegenrichtung des Normalenvektors)

Mit Hilfe dieser Regel wird die Richtung des jeweiligen Normalenvektors einer Dreiecksfläche festgelegt. Um die Orientierung der Dreiecksfläche zu bestimmen, wird die Richtung dieses Normalenvektors näherungsweise an die Richtung einer der geeigneten globalen kartesischen Koordinatenachse angepasst (siehe Kapitel 2.4). Das ist gleichzeitig auch die Anpassung der Dreiecksfläche oder auch Dreieckebene. Ist der Normalenvektor nach der Anpassung der Ebene in eine der in Abbildung 12 dargestellten Richtungen gerichtet, dann ist die Orientierung der entsprechenden Dreiecksfläche positiv (positive Orientierung), sonst negativ.

Zu Beachten ist dabei, dass alle Punkte in einer Ebene (Best-Projektionsebene, siehe Kapitel 2.3.1) liegen müssen, wenn man diese Regel anwendet, um die Orientierung der Dreiecksfläche zu veranschaulichen. Durch Bestimmung der Orientierungen werden die direkten Nachbarpunkte identifiziert und dabei sowohl die überflüssigen als auch die störenden Punkte aus der nicht-optimalen Triangulation eliminiert (siehe Kapitel 3.3.3.2).

Die Abbildung 12 stellt grafisch dar, wie der Daumen nach der Rechte-Hand-Regel bei positiver Orientierung für unterschiedliche Würfelflächen zum jeweiligen Bezugspunkt (siehe Abschnitt 2.3.2) gerichtet ist. Zu Beachten ist dabei, dass in diesem Beispiel die Flächen der Würfel angepasst sind, d.h. die Normalenvektoren der Würfelflächen zeigen genau in eine Richtung der globalen kartesischen Koordinatenachsen. Wenn das nicht der Fall ist, wird die Anpassung mit der in Abschnitt 2.4 vorgestellten Methode für einzelne Dreiecksflächen ausgeführt. Das in der Abbildung 12 dargestellte Beispiel kann für Dreiecksflächen, die bei der lokalen Triangulation in anderen Teilobjekten auftreten, verallgemeinert werden.

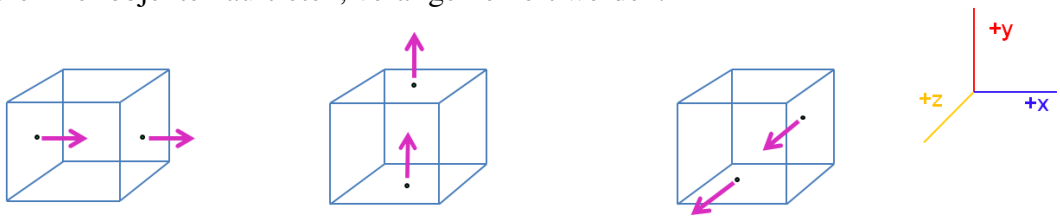


Abbildung 12: Richtung des Daumens bei der positiven Orientierung einer Dreiecksfläche zu jeweiligem Bezugspunkt in unterschiedlichen Würfelebenen nach der "Rechte-Hand-Regel"

Zu Berücksichtigen ist, dass bei einer positiven Orientierung die Richtung des Daumens entgegen der Richtung des Normalenvektors der entsprechenden Dreiecksfläche zeigt, vgl. Abbildung 12 und Abbildung 7.

2.3.4 Verwendung der Graham Scan Methode

Das Prinzip der Graham Scan Methode wird für Identifizierung der direkten Nachbarpunkte eines betrachteten Punktes bei einer lokalen Triangulation verwendet. Diese Methode wurde zu einer neuen Methode weiterentwickelt, mit der man die störenden Punkte in einer Ebene in 3D eliminiert (siehe Kapitel 3.3.3.2). Die Elimination der störenden Punkte ist wichtig für eine optimale lokale Triangulation. Dabei wird angenommen, dass eine optimale lokale Triangulation um einen betrachteten Punkt mit seinen direkten Nachbarpunkten möglich ist.

Es gibt eine Vielzahl von Algorithmen, die die konvexe Hülle (siehe Appendix B) aus einer Punktmenge berechnen. Unter diesen ist die Graham Scan Methode wegen ihrer Funktionsweise von besonderer Bedeutung für die in Kapitel 3.3.3.2 neu entwickelte Eliminationsmethode.

Der Algorithmus der Graham Scan Methode lässt sich grob folgendermaßen zusammenfassen [Lang06][PrSh85]:

Gegeben sind n Punkte $P_1 \ldots P_n$

1. Wähle den Punkt P_i mit minimalem y-Wert als Startpunkt.
2. Sortiere die restlichen Punkte mit Startpunkt entsprechend dem Polarwinkel.
3. Verbinde die Punkte nach ihrer Sortierung zu einem Polygonzug.
4. Entferne die Punkte aus dem Polygonzug, die keine Eckpunkte der konvexen Hülle sind.

Im Schritt 4 in dem obigen Algorithmus werden die Nicht-Eckpunkte der konvexen Hülle aus dem Polygonzug durch Prüfung der negativen Orientierungen der vorkommenden Dreiecksflächen bestimmt. Dies sind Dreiecke von jeweils drei im Polygonzug aufeinanderfolgenden Punkten (siehe Abbildung 16). Dabei wird untersucht, ob die Punkte um den Ursprung entgegen

dem Uhrzeigersinn angeordnet sind. Das wird mit Hilfe des Kreuzprodukts in 2D oder in 3D, wenn alle Punkte in einer Ebene (Best-Projektionsebene (siehe Abschnitt 2.3.1)) liegen, bestimmt. Die Vektoren des Kreuzproduktes werden nach der Reihenfolge unter drei aufeinanderfolgenden Punkten gebildet. Ist z.B. eine Dreiecksfläche im Polygonzug negativ orientiert, wird der Reihe nach der mittlere Punkt des Dreieckes aus dem Polygonzug entfernt. Nach der Entfernung wird auch geprüft, ob bei vorher besuchten Dreieckflächen negative Orientierungen entstanden sind. Wenn ja, werden sie gleichermaßen behandelt. Die aus dem Polygonzug entfernten Punkte sind dann innere Punkten der konvexen Hülle. Sind alle Punkte durchlaufen und der Ausgangspunkt erreicht, sind die übrig gebliebenen Punkte des Polygonzuges die Eckpunkte der gesuchten konvexen Hülle.

Die Komplexität der Methode ist $O(nlog_n)$ zur Bestimmung der konvexen Hülle von einer gegebenen Menge mit n Punkte in 2D [AuSp93].

In Abbildungen 13-14 sind die Schritte der Graham Scan Methode dargestellt. In der Abbildung 13 (links) sieht man die n Punkte in der zweidimensionalen Ebene, deren konvexe Hülle gesucht wird. Diese Ebene ist eigentlich eine Polarkoordinatenebene (sowohl in 3D als auch in 2D) (siehe Abschnitt 2.2) und diese Ebene wird andererseits im dreidimensionalen Raum innerhalb des Konzepts als *Best-Projektionsebene* bezeichnet (für die Ermittlung der Ebene siehe Abschnitt 2.3.1). Die Punkte sind in der Abbildung 13 (Mitte) nach Polarwinkel sortiert. In der Abbildung 13 (rechts) ist der Polygonzug der sortierten Punkte.

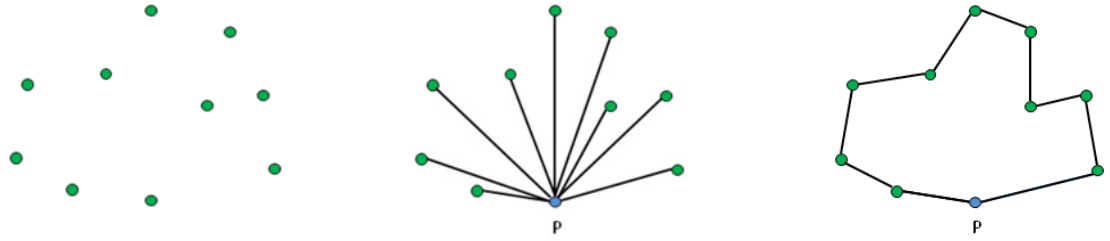


Abbildung 13: Graham Scan Methode: (links) Punkte in einer Ebene in 2D, (Mitte) die nach Polarwinkel sortierte Punkte, (rechts) Polygonzug bzgl. der Sortierung

In der Abbildung 14 ist die gesuchte konvexe Hülle der Punkte zu sehen. Die konvexe Hülle korrespondiert eigentlich dem "*Außenrand*", der alle Punkte umhüllt.

Der Algorithmus der Graham Scan Methode ist so angepasst, dass anstatt dem Außenrand (konvexe Hülle) der "*Innenrand*" (siehe Abbildung 15) der Punkte gefunden wird. Der Grund für diese Änderung ist, dass die Punkte, die zu dem Innenrand gehören, die *direkten Nachbarpunkte* (Nachbarn ersten Grades) von Punkt P ergeben. Die angepasste Version des Algorithmus nach dem Prinzip der Graham Scan Methode wird *Eliminationsmethode* (siehe Kapitel 3.3.3.2) benannt. Dadurch werden die Nachbarpunkte von einem betrachteten Punkt P eliminiert, die bei einer lokalen, nicht-optimalen Triangulation bzgl. des Punktes P als störende oder überflüssige Punkte auftreten könnten. Ziel dieser Umstellung ist es eine optimale lokale Triangulation zu erstellen, in welcher ein Dreiecksnetz nur aus direkten Nachbarn vom Punkt P aufgebaut wird. Dank der Best-Projektionsebene funktioniert die neu entwickelte Methode im 3D-Umfeld. Diese Ebene wird zur Sortierung der Nachbarpunkte in dem neuen Verfahren *Lokale Triangulation mit direkten Nachbarpunkten* verwendet.

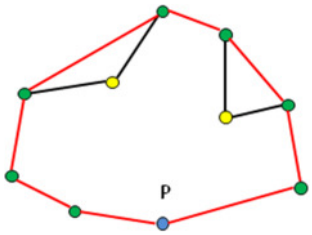


Abbildung 14: Konvexe Hülle ("Außenrand")

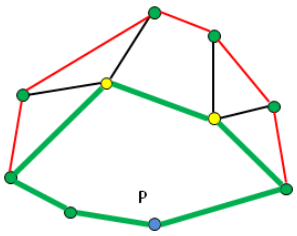


Abbildung 15: "Innenrand "der Punkte (= die Gesuchten direkten Nachbarn von Punkt *P*)

Ein Punkt ist genau dann ein direkter Nachbar eines beliebigen Punktes, wenn er in einem Dreiecksnetz, das nach einem beliebigen Triangulationsverfahren entsteht, direkt vom Punkt *P* aus erreichbar ist, ohne einen anderen Nachbarpunkt über eine Kante im Dreiecksnetz zu besuchen. Solche Punkte sind genau die gesuchten Punkte, mit denen man lokal triangulieren will, um einen Normalenvektor zu dem Oberflächenpunkt *P* des gescannten realen Objektes genauer zu berechnen.

Ein Problem bei dem Verfahren *Local Center Triangulation* (siehe Abschnitt 2.5.2) ist, dass man möglichst viele Nachbarpunkte von einem beliebigen Punkt *P* betrachtet und nicht zwischen *direkten* und *indirekten* Nachbarpunkten unterscheidet. Deswegen wurde in dem neuen, optimierten Verfahren *Lokale Triangulation mit direkten Nachbarpunkten* nach solchen *direkten Nachbarpunkten* gesucht und versucht, *indirekte* Nachbarn aus der Triangulation zu entfernen. Somit wird der Normalenvektor vom Punkt *P* besonders an den scharfen Kanten der geometrischen Grundprimitiven und bei Übergängen zwischen den Teilobjekten genauer berechnet.

In Abbildung 16 sieht man ein nicht-konvexes Polygon, welches nach der Sortierung der Punkte (P_1, …, P_{10}) entstanden ist. Die Umlaufrichtung ist dabei gegen den Uhrzeigersinn.

Während die Orientierung des Dreiecks $P_2P_3P_4$ nach der definierten Umlaufrichtung positiv ist, ist die Orientierung des Dreiecks $P_6P_7P_8$ negativ. Die Orientierung kann man leicht mit der im Abschnitt d) vom Kapitel 2.2.3 beschriebenen Rechte-Hand-Regel feststellen. Die Orientierungen der einzelnen Dreiecksflächen werden durch ein Kreuzprodukt zwischen den Vektoren und berechnet. Die Vektoren werden aus drei aufeinanderfolgenden Punkten, die durch die Sortierung nach der Graham Scan Methode entstanden sind, gebildet.

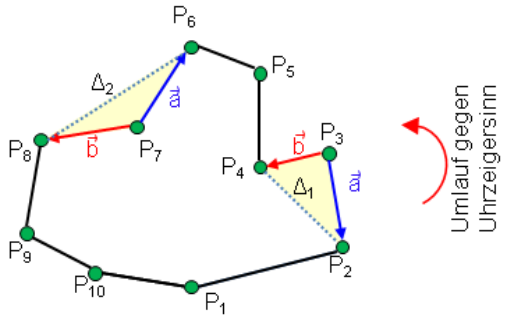


Abbildung 16: Bestimmung der Orientierung der Dreiecke mit Kreuzprodukt

Zur Festlegung der Orientierung der Dreiecksflächen werden die folgenden Regeln berücksichtigt:

1) Ist Umlaufrichtung der Dreieckpunkte gegen den Uhrzeigersinn, hat die Dreiecksfläche eine positive Orientierung.

2) Ist Umlaufrichtung der Dreieckspunkte im Uhrzeigersinn, hat entsprechende Dreiecksfläche eine negative Orientierung.

Der Nachteil von der Graham Scan Methode ist jedoch, dass die Übertragung der Methode in höhere Dimensionen nicht möglich ist, weil Graham Scan prinzipiell auf der Sortierung der *n* Punkten nach ihrem Polarwinkel in der 2D-Ebene basiert, was natürlich die Methode nicht auf höheren Dimensionen erweitern lässt. Führt man stattdessen diese Methode aber für die endliche Anzahl von Punkten in einem drei-dimensionalen Raum aus – ohne die Punkte in eine Ebene zu projizieren – ergibt es ein Volumen, das alle Punkte im drei-dimensionalen Raum umhüllt [Gems11] und welches auch Polytop (engl. polytope) [Well02][PrSh85] genannt wird.

Das ist aber während des Objekterkennungsprozesses nicht erwünscht, da somit die oberflächliche Struktur der Objekte in der Punktwolke nicht berücksichtigt würde.

Das ganze Verfahren in Rahmen dieser Diplomarbeit basiert aber auf Oberfläche der gescannten Objekte. Das oben benannte Problem wird mit Hilfe der Best-Projektionsebene gelöst, indem alle Punkte zuerst in diese Ebene projiziert werden, so dass die oben erläuterte Bestimmung der *direkten* Nachbarpunkte mit dieser Ebene durchgeführt wird.

Diese Vorgehensweise bringt aber das Problem mit sich, dass die oberflächliche Struktur der Objekte bei der Projektion nicht berücksichtigt wird, wenn es um die Betrachtung der Punkte an den scharfen Kanten und Übergänge zwischen geometrisch unterschiedlichen Objekten in einer Punktwolke geht. Dieses Problem wird mit dem im Abschnitt 3.3.3.2.2 erläutertem neuen Ansatz behoben. Dadurch kann man auf Nachbarpunkten von einem betrachteten Oberflächenpunkt in einer 3D-Ebene operieren, da die Dreiecksflächen im Polygon ihre Topologie bzgl. der Orientierung nicht verlieren.

2.4 Übertragung von 3D-Punkten in 2D-Ebene

In bestimmten Fällen ist das Arbeiten mit Punkten in 2D einfacher als mit Punkten in 3D. Deswegen ist es manchmal erforderlich einen Punkt oder mehrere Punkte von einem Raum in einen anderen "kleiner"-dimensionierten Raum zu transformieren. Wichtig sind dabei die Rückschlussinformationen, d.h. man soll bei solchen Transformationen keine Informationsverluste erhalten und ohne großen Aufwand wieder in die ursprüngliche Position zurückkehren können.

Die Idee hierbei ist, dass man die 3D-(Dreiecks-)Punkte der Reihe nach vom 3D-Raum in den 2D-Raum überträgt, ohne die Topologie der Punkte bezüglich der Orientierung zu verlieren. Die Orientierung der Dreiecksfläche wird mit Hilfe des Kreuzproduktes in 2D bestimmt. Ist das Kreuzprodukt zwischen den zwei Kanten eines Dreieckes in einer Ebene positiv, dann ist die Dreiecksfläche, die durch diese Kanten aufgespannt wird, *positiv orientiert*, ansonsten ist die Dreiecksfläche *negativ orientiert*.

In dem hier beschriebenen Verfahren werden die Punkte einer Punktwolke immer in dreier Gruppen in den 2D-Raum projiziert. Diese drei Punkte bilden die Ecken eines Dreiecks, welche in Abhängigkeit einer Sortierreihenfolge zusammenhängend sind.

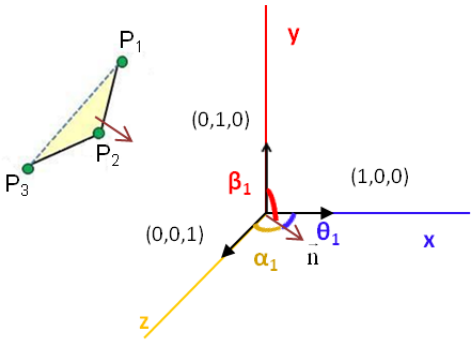
So eine Übertragung von Punkten aus dem dreidimensionalen Raum in den 2D-Raum bringt einen großen Vorteil bei der Bestimmung der Orientierung von "Dreiecksflächen" innerhalb eines Polygonzuges (siehe Graham-Scan-Methode in Kapitel 2.3.4). Dabei ist die Reihenfolge der Punkte vom Dreieck im 3D-Raum wichtig, d.h. die Reihenfolge der Punkte soll bei der Übertragung beibehalten werden, sonst könnte es zu einer falschen Orientierungsberechnung führen. Die Reihenfolge wird durch eine Sortierung von Punkten in einer Best-Projektionsebene festgelegt, bevor die 3D-Punkte in den zweidimensionalen Raum übertragen werden. Das Sortierungsverfahren wird weiter unten näher beschrieben (Kapitel 3.3.3.1).

Das Dreieck $P_1P_2P_3$ sei das in einem Polygonzug vorkommende Dreieck, deren 3D-Punkte in einem globalen kartesischen Koordinatensystem angegeben sind, wie in der Abbildung 17 dargestellt ist.

Bevor man die Punkte projiziert, wird eine Fläche zwischen diesen drei Punkten aufgespannt, was im folgenden Dreiecksfläche (oder in manchen Fällen Dreiecksebene) genannt wird. Danach wird dessen Normalenvektor (siehe Abbildung 18) durch das Kreuzprodukt zwischen den Vektoren $\overrightarrow{P_2P_1}$ und $\overrightarrow{P_3P_2}$ berechnet, welcher senkrecht auf der Dreiecksfläche steht. Die Ausrichtung der Vektoren ($\overrightarrow{P_2P_1}$ sowie $\overrightarrow{P_3P_2}$) wird ebenfalls durch die Sortierungsreihenfolge der Punkte festgelegt. Es ist wichtig diese Reihenfolge zu beachten, da ansonsten die Orientierung der Dreiecksfläche gegenüber ihrem Bezugspunkt nicht richtig bestimmt werden kann.

Abbildung 17: Eckpunkte eines Dreiecks der Reihe nach P1, P2 und P3 vor der Übertragung in eine der Ebenen, welche von den Achsen des globalen Koordinatensystems aufgespannt wird

Im nächsten Schritt werden die Winkel ($\beta_{1,2}$, $\alpha_{1,2}$, $\theta_{1,2}$) zwischen dem Normalenvektor der Dreiecksfläche und allen drei Einheitsvektoren des globalen kartesischen Koordinatensystems berechnet, siehe Abbildung18. Zu Beachten ist, dass nicht drei sondern sechs Winkel berechnet werden, weil die Winkel zwischen den Einheitsvektoren sowohl in positiver als auch negativer Richtung zu berücksichtigen sind. Unter diesen Winkeln wird der kleinste Winkel ausgewählt. Gleichzeitig wird auch die Richtung des ausgewählten Einheitsvektors bestimmt, z.B. ob der Normalenvektor der Dreiecksfläche den kleinsten Winkel mit dem Einheitsvektor (1, 0, 0) oder (-1, 0, 0) der *x*-Achse einschließt.



$\beta_{1,2} = (\vec{n}, (0, \pm1, 0))$
$\alpha_{1,2} = (\vec{n}, (0, 0, \pm1))$
$\theta_{1,2} = (\vec{n}, (\pm1, 0, 0))$

$\min(\beta_{1,2}, \alpha_{1,2}, \theta_{1,2}) = \theta_1$
$\Rightarrow$ im Beispiel werden P_1, P_2 und P_3 in die y-z-Ebene projiziert (in positive x-Richtung)

Abbildung 18: Berechnen des kleinsten Winkels, den der Normalenvektor der Dreiecksfläche mit einem der Einheitsvektoren des globalen kartesischen Koordinatensystems einschließt

Danach lässt sich die geeignete Ebene bestimmen, in die die Punkte projiziert werden sollen, ohne dass die topologische Struktur des Dreiecks verloren geht. Die geeignete Projektionsebene wird dabei durch die zwei Achsen des globalen kartesischen Koordinatensystems aufgespannt, welche den kleinsten Winkel mit dem Normalenvektor der Dreiecksfläche einschließen. Siehe Abbildung 19.

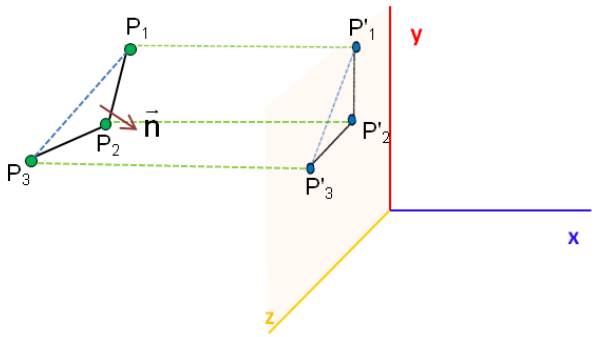
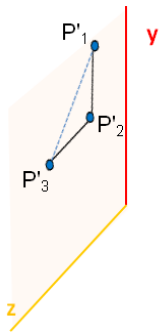


Abbildung 19: Projektion der Dreieckspunkte in die geeignete Ebene

Nachdem die geeignete Projektionsebene im globalen *x*-*y*-*z*-Koordinatensystem ermittelt ist, werden die Punkte in diese Ebene übertragen.

Wenn man nicht die geeignete Projektionsebene verwendet, sondern eine andere Ebene, so könnte unter Umständen ein Strukturverlust des Dreiecks entstehen. Das wäre dann der Fall wenn die Dreiecksfläche senkrecht auf der Projektionsebene steht. Dadurch würden die Punkte auf eine Gerade abgebildet werden.

Nachdem die Punkte in die geeignete Ebene projiziert wurden, wird ihre Dimension um eins reduziert, in dem eine von den *x*-, *y*-, *z*-Koordinaten weggelassen wird. Dabei wird diejenige Koordinate weggelassen, deren zugehörige Achse des globalen Koordinatensystems, senkrecht auf der Projektionsebene steht (in Abbildung 19 beispielsweise ist es die *x*-Achse). Diese Koordinate lässt sich aus dem Grund weggelassen, weil nach der Projektion ihr Wert null wird.



$$P'_1 = \begin{pmatrix} 0 \\ y_1 \\ z_1 \end{pmatrix}, \quad P'_2 = \begin{pmatrix} 0 \\ y_2 \\ z_2 \end{pmatrix}, \quad P'_3 = \begin{pmatrix} 0 \\ y_3 \\ z_3 \end{pmatrix}$$

Abbildung 20: Die Dreieckseckpunkte in 2D nach der Projektion in der geeigneten Ebene

Danach befinden sich die Punkte ohne Informationsverlust im 2D-Raum, d.h. die Topologie des Dreiecks bezüglich der Orientierung bleibt beibehalten. Die neuen Punkte sind P_1', P_2' und P_3' und bilden ein neues Dreieck.

Die Vorteile für die Übertragung von 3D nach 2D zusammengefasst:

1) Mit Hilfe des Kreuzproduktes in 2D wird ein skalarer Wert berechnet, dessen Vorzeichen angibt, in welche Richtung der Normalenvektor der Dreiecksfläche zeigt, wenn man dieses Dreieck im 3D-Raum betrachtet. Die Richtung des Normalenvektors gibt an, ob der Normalenvektor in Richtung des Bezugspunktes zeigt oder entgegengesetzt ist. Somit kann man bestimmen, ob die Dreiecksfläche negativ oder positiv orientiert ist. Die weggelassene Achse (in Abbildung 19 die *x*-Achse) deutet dabei den betreffenden Bezugspunkt hinsichtlich der Dreiecksfläche an, denn es handelt sich um denjenigen festen Bezugspunkte (siehe 2.3.2), der sich auf dieser Achse befindet.

2) Diese 2D-Punkte können mit den Verfahren "*Local Delauney Triangulation*" oder "*Local Center Triangulation*" weiterverarbeitet werden (siehe Abschnitte 2.5.1 und 2.5.2).

3) Des Weiteren wird diese Methode für die Sortierung nach Polarwinkeln bei dem Verfahren "*Lokale Triangulation mit direkten Nachbarpunkten*" verwendet, um herauszufinden, ob ein Punkt links oder rechts von der Polarachse liegt (siehe Abbildung 35).

2.5 Triangulation

Um eine Oberfläche aus den unterschiedlichen geometrischen Objekten in einer Menge von Punkten aufzuspüren und anschließend klassifizieren zu können, braucht man geometrisch spezifische Oberflächenmerkmale, wie z.B. Normalenvektoren oder Krümmungen an den einzelnen Oberflächenpunkten. Die Oberflächen werden trianguliert, um diese geometrischen Informationen zu gewinnen. Diese Oberflächen werden durch die Punkte beschrieben, welche z.B. durch einen Scanner erfasst sind und als Punktwolke vorliegen.

Unter Triangulation (auch Dreiecksvermaschung) der Flächen versteht man im Allgemein das Zerlegen der Fläche in Dreiecke. Das Resultat einer Triangulation ist ein Dreieckecknetz (auch Dreieckgitter oder Triangulierung genannt), d.h. die Fläche eines Objektes wird mit Hilfe des Dreiecksnetzes näherungsweise rekonstruiert.

Bei einer Triangulation werden drei verschiedene diskrete Punkte miteinander verbunden, welche in irgendeiner Beziehung zueinander stehen, z.B. einer Nachbarschaftsbeziehung nach dem euklidischen Abstand und bilden dadurch ein Dreieck. So entsteht aus einer Menge von Punkten ein Netz von Dreiecken.

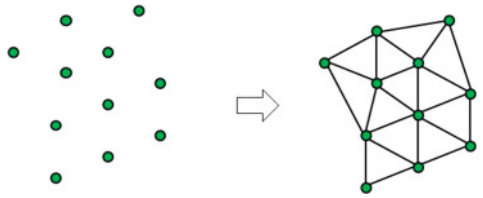


Abbildung 21: Triangulation aus einer Menge von Punkten: (links) die Punktwolke, (rechts) das entsprechende Dreiecksnetz

Man unterscheidet bei den Objekterkennungsverfahren o.B.d.A. zwischen zwei Arten von Triangulationen, um geometrische Merkmale (engl. features) der Oberfläche zu gewinnen:

- globale Triangulation (Vortriangulation)
- lokale Triangulation

Während die ganze Oberfläche eines Objekts (liegt als Punktwolke vor) bei der *globalen* Triangulation nach einem beliebigen Triangulationsverfahren dargestellt wird, wird das bei einer *lokaler* Triangulation "lokal" durchführt, d.h. durch Betrachtung der einzelnen Punkte mit der Nachbarschaft. Dabei werden möglichst viele Nachbarpunkte mit ihren Lagen in der Punktwolke um den jeweiligen betrachtenden Punkt analysiert und unter diesen Punkten ein kleiner Teil der Oberfläche temporärer dargestellt. Die lokale Triangulation wird für jeden einzelnen Oberflächenpunkt einmal durchgeführt. Die globale Triangulation kann man als eine Vortriangulation verstehen, die nur einmal vor dem Objekterkennungsprozess ausgeführt und deren Ergebnis (das Dreieckecknetz) abgespeichert wird. Die Algorithmen der Objekterkennungsverfahren, in denen eine Vortriangulation vorausgesetzt ist, basieren grundsätzlich auf Dreiecknetze. Die globale Triangulation kommt nicht, in dem im Rahmen dieser Diplomarbeit einge-

führten, Konzept vor, d.h. das innerhalb des Konzeptes dargestellte Objekterkennungsverfahren wird direkt auf die Punkte des gescannten Objektes angewendet.

Die lokalen Triangulationen werden für das Berechnen der Normalenvektoren zu den einzelnen Oberflächenpunkten des gescannten realen Objektes (siehe Kapitel 3.3.3) eingesetzt. Zu diesem Zweck wurden am Anfang die zwei in der Dissertation [Vanc03] erwähnten Algorithmen *Local Delaunay Triangulation* (LDT) und *Local Center Triangulation* (LCT) als Basis genommen, implementiert und anschließend die Resultate visualisiert. Nach der Analyse der beiden Verfahren wurde das Verfahren *Local Center Triangulation* grundsätzlich erweitert und zur Optimierung eine neue Eliminationsmethode entwickelt (siehe Kapitel 3.3.3.2). Diese Methode wird im Rahmen des Konzeptes als "*Lokale Triangulation mit direkten Nachbarpunkten* " (siehe Kapitel 3.3.3.1) benannt.

Die Pseudocodes von den Verfahren *Local Delaunay Triangulation* (LDT) und *Local Center Triangulation* sehen wie folgt aus:

2.5.1 Local Delaunay Triangulation

Da dieses Verfahren in einer zweidimensionalen Ebene funktioniert, sind die triangulierenden Punkte zuerst mit der in erläuterter Methode vom dreidimensionalen Raum in den zweidimensionalen Raum zu übertragen. Somit könnten die Punkte ohne Informationsverlust in 2D gespeichert werden [Vanc03].

Sei P der betrachtete Punkt, für den der Normalenvektor berechnet wird und P_i, $i = \{1, \ldots, n\}$, seine n Nachbarn [Vanc03]:

- Finde die Best-Projektionsebene E mit P und P_1.
- Projiziere Punkte P_i in diese Ebene und speichere ihre 2D-Koordinaten als P_i^{2D}.
- Verwende Delaunay-Triangulation für die 2D-Punkte P_i^{2D} und baue danach eine Triangulation in 3D aus den Punkten P_i, die P_i^{2D} entsprechen.

Berechne den Durchschnitt der Normalenvektoren von Dreiecksflächen, bei denen der Punkt P vorkommt.

Die in diesem Algorithmus vorkommende Ebene, in die die 3D-Punkte projiziert werden, wird wie im Abschnitt 2.3.1 beschrieben als Best-Projektionsebene bezeichnet.

Um eine lokale Triangulation nur mit direkten Nachbarn vom Punkt P auszuführen, wurden die Dreiecke, bei denen Punkt P vorkommt, aus dem Dreiecknetz gefiltert, d.h. nur die Dreiecke auswählt, in denen Punkt P als Mittelpunkt enthalten ist. Die gefilterten Punkte des Dreiecknetzes zur Erstellung der Triangulation befinden sich aber bzgl. der verwendeten Delaunay-Triangulation (in 2D) jeweils in 3D. Der geschätzte Normalenvektor wird dann als Durchschnitt der Normalenvektoren dieser gefilterten Dreiecke berechnet.

Dieses Verfahren allein ist nicht ausreichend, um die Normalenvektoren an den Oberflächenpunkten zu berechnen; das Verfahren weist Schwächen besonders bei (scharfen) Kanten und Übergänge zwischen den in der verarbeiteten Punktwolke vorkommenden Teilobjekten auf.

2.5.2 Local Center Triangulation

Die Grundidee dieser Methode ist, dass das Zentrum C_m der lokalen Nachbarschaft (bezeichnet mit $P_1, P_2, \ldots, P_k$) von P nahe am Punkt P liegt. Schätzt man den Normalenvektor in C_m ab, liegt der abgeschätzte Normalenvektor in der Nähe des Normalenvektors von P [Vanc03].

Der Algorithmus funktioniert wie folgt**:**

- Berechne das Zentrum C_m aus der Nachbarschaft von P

$$C_m = \frac{1}{k}\sum_{i=1}^{k} P_i \tag{2.3}$$

- Trianguliere die Nachbarschaft nach C_m:
  1. Finde eine geeignete Projektionsebene E wie unten beschrieben
  2. Projiziere die Punkte P_i auf diese Ebene und speichere ihre 2D-Koordinaten als P_i^{2D}
  3. Sortiere Punkte P_i^{2D} nach ihren Polarkoordinaten mit dem Vektor $\overrightarrow{C_m P_1}$ als x-Achse. Wir bezeichnen diese Punkte mit P_j^{2D}; $j = 1, 2, ..., k$.
  4. Baue eine Triangulation T_2 im 2D aus P_j^{2D}: ein Dreieck T_j^{2D} besteht aus (P_j^{2D}, C_m, $P^{2D}_{(j \bmod k)+1}$), $j = 1, \ldots, k$
  5. Baue eine Triangulation T_3 im 3D aus T_2 (jeder Punkt P_j^{2D} entspricht einem Punkt P_i im 3D)

  - Berechne den Normalenvektor $\vec{N}_i$ in jedem 3D-Dreieck
  - Setze den Normalenvektor $\vec{N}$ in C_m zu den durchschnittlichen

$$\vec{N} = \frac{1}{k}\sum_{i=1}^{k} \vec{N}_i, \tag{2.4}$$

  beachte dass k Dreiecke erstellt wurden

  - Verwende $\vec{N}$ als der geschätzte Normalenvektor in P

Die in dem Algorithmus für die Projektion der Punkte in 2D nötige Ebene E wird aus der Menge der Ebenen als diejenige Ebene ausgewählt, bei der die Summe der orthogonalen Abstände von allen P_i zu E minimal ist (siehe Abschnitt 2.3.1). Die Menge der Ebenen entsteht, in dem man die Punkte P_1, P_2 als die ersten zwei Punkte und den dritten Punkt aus P_i, $i = 3,..., k$ auswählt, so dass eine Ebene aufgespannt wird.

Dieser Algorithmus ist ein wichtiger Bestandteil des hier vorgestellten Konzepts. Zu unserem Zweck wurden einige Änderungen bei dem Algorithmus vorgenommen und mit einer neu entwickelten Methode (siehe Kapitel 3.3.3.2) optimiert, welche als *Lokale Triangulation mit direkten Nachbarpunkten* benannt wird. Der Pseudocode des angepassten Algorithmus. Die vorgenommenen Änderungen sind in Kapitel 3.3.3.1 zu finden. In dem entsprechenden Kapitel wird der Algorithmus genauer beschreiben.

Sowohl *Local Delaunay Triangulation* als auch die *Local Center Triangulation* arbeiten in einer Ebene (Best-Projektionsebene), d.h. die im 3D-Raum liegende Punkte werden zuerst in eine Ebene projiziert. Durch diese Projektion treten besonders an scharfen Kanten oder Übergängen zwischen den, in der Punktwolke enthaltenen, Teilobjekten Probleme auf. Dies ist dadurch zu erklären, dass bei der Projektion die Struktur der Oberflächen aus den gescannten realen Objekten nicht berücksichtigt wird, was eine korrekte lokale Triangulation erschwert. Ein Lösungsansatz wird in Kapitel 3.3.3.2 ausführlich erklärt.

3 Konzeptentwicklung

Für das Verständnis der innerhalb dieses Konzepts vorgestellten Prozessschritte zur Berechnung der Flächenmerkmale an Oberflächenpunkten von Objekten, wie z.B. Normalenvektoren oder Krümmungen, und zur Segmentierung von Oberflächenpunkten entsprechend berechneten Merkmalen werden die Grundlagen aus Kapitel 2 vorausgesetzt.

3.1 Grundschritte des Konzeptes

In dieser Arbeit wird ein vierstufiges Verfahren zur Erkennung von geometrischen Objekten aus einer unstrukturierten Punktwolke vorgestellt. Dies wird durch Gewinnung von verschiedenen spezifischen Merkmalen an den Oberflächenpunkten erreicht. Darüber hinaus werden zwei Segmentierungen der Punktwolke bezüglich der gewonnen Merkmale durchgeführt.

Um aus einer gegeben Punktwolke (Definition siehe Einführung) scharfe Kanten und Bereiche mit starken Krümmungen zu identifizieren, werden in der ersten Stufe Normalenvektoren berechnet und in der dritten Stufe die Punktwolke anhand dieser Normalenvektoren segmentiert. In der zweiten Stufe werden Krümmungen als zusätzliche Merkmale mit Hilfe des Gauß-Bonnet-Ansatzes gesammelt, um die Punktwolke weiter zu aufteilen und entsprechend zu klassifizieren. Die in dieser Stufe durchgeführten Berechnungen basieren auf Dreiecksnetze (Polyeder), die durch die lokalen Triangulationen in der ersten Stufe zur Berechnung der Normalenvektoren an den jeweiligen Punkten erstellt wurden. Die Kombination des Satzes von Gauß-Bonnet mit lokalen Triangulationen, verschafft zeitliche Vorteile bei dem Prozess der Objekterkennung, weil dabei auf aufwendige analytische Krümmungsberechnungen an den jeweiligen Objektoberflächenpunkten verzichtet wird. In der letzten Stufe werden dann die Oberflächenpunkte nach ihren charakteristischen Eigenschaften bezüglich der Krümmungen klassifiziert, die in der zweiten Stufe berechnet wurden. Damit werden die Punkte jeweils zu den bekannten geometrischen 3D-Grundprimitiven (siehe Abbildung 22) zugeordnet.

Abbildung 22: Beispiele für vier geometrischen 3D-Grundprimitive

In dieser Arbeit wird ein Verfahren zur Merkmalgewinnung aus einer Menge von unstrukturierten Punkten und ihre Klassifizierung mit Hilfe der berechneten Merkmale durch die Segmentierungen vorgestellt. Dieses Verfahren lässt sich in vier Grundschritte gliedern, die in [Vanc03] beschrieben sind:

a) Berechnung der Normalenvektoren

b) Abschätzung der Hauptkrümmungen

c) Normalenbasierte Segmentierung

d) Krümmungsbasierte Segmentierung

Für die in den Grundschritten verwendeten Verfahren gibt es unterschiedliche Ansätze, beispielsweise zur Abschätzung der Normalenvektoren an den jeweiligen Oberflächenpunkten. Das führt dazu, dass unterschiedliche Lösungswege existieren. So werden in jedem Grundschritt

Methoden bereitgestellt, die die Punktdaten, ihre Lage im Raum und ihre Beziehungen untereinander interpretieren. Diese Methoden nutzen Merkmale der in der Punktwolke enthaltenen Objekteoberflächen. Durch diese Merkmale lassen sich die Oberflächen dieser Objekte hinreichend beschreiben.

Durch den dritten Grundschritt werden die Grenzen der Teilpunktwolken (Segmente) in Bezug auf die Richtung und Abweichung der Normalenvektoren ermittelt (normalenbasierte Segmentierung). Dann werden die Oberflächenpunkte mit ähnlichen Eigenschaften bezüglich der Krümmungen segmentiert (krümmungsbasierte Segmentierung).

Es kann vorkommen, dass wegen errechneten oder vordefinierten Schwellwerten bei den eingeführten Segmentierungsverfahren mehrere kleine Teilpunktwolken (Segmente) entstehen, die eigentlich nach ihren charakteristischen Eigenschaften z.B. ähnliche Krümmungen, zusammengehören. In diesem Fall werden die betroffenen Teilpunktwolken zusammengefasst, um die zugehörigen Objekte als Ganzes zu erkennen.

Auf diese Konzeptionsschritte wird im folgenden Kapital detailliert eingegangen.

3.2 Programmflussdiagramm

Der Ablauf der Objekterkennung ist in Abbildung 23 skizziert. Als Eingabe dient eine Textdatei, in der Koordinaten von mehreren Punkten gespeichert sind. Die Punkte bilden zusammen die Oberfläche mehrerer zusammengesetzter geometrischen Grundkörper (Maschinenteil, engl. machining part). Die Menge aller Punkte bildet eine Punktwolke. Diese Punktwolke wird vom Programm eingelesen und als indizierter *kD*-Baum abgelegt. Der *kD*-Baum-Algorithmus und Anpassungen im Rahmen dieser Arbeit sind in Kapitel 3.3.2 erläutert.

Ausgehend von einem bestimmten Punkt, werden mit Hilfe des *kD*-Baums die Nachbarpunkte anhand der Indizes ermittelt. Diese Aktion wird für jeden Punkt aus der Punktwolke iterativ durchgeführt, da alle Punkte einmal, zur Abschätzung der oberflächlichen Merkmale zu dem jeweiligen Punkt, betrachtet werden. Die ermittelten Nachbarpunkte werden sortiert in einer Liste abgelegt. Die Sortierung erfolgt anhand der Entfernung der Nachbarpunkte zu P_i.

Als Nächstes wird für jeden Punkt P_i und seine Nachbarpunkte eine Best-Projektionsebene ermittelt. Dieser Schritt ist in Kapitel 2.3.1 näher beschrieben. Nach dem die Ebene ermittelt ist, werden alle Punkte durch Projektion in diese Ebene abgebildet. Die Projektion ermöglicht es, Punkte in einem 3D-Raum zu sortieren. Nach der Sortierung der Nachbarpunkte bezüglich P_i in der Best-Projektionsebene werden die Punkte in ihre ursprüngliche Position zurückprojiziert, um das Dreiecksnetz zu erstellen (lokale Triangulation hinsichtlich P_i).

Die Triangulation wird hier als "*lokal*" bezeichnet, weil sie temporär beim *i*-ten Vorgangsschritt nur anhand vom Punkt P_i und seiner Nachbarpunkte erfolgt. Die Erstellung dieser lokalen Triangulation als Dreiecksnetz ist in Kapitel 3.3.3.2.4 erläutert.

Zur Gewinnung von Oberflächenmerkmalen wie z.B. durch die Berechnung des Normalenvektors oder Hauptkrümmungsberechnung mit Hilfe des Satzes von Gauß-Bonnet an dem Punkt P_i, sind optimale Triangulation nötig. Die ermittelten lokalen Triangulationen sind im Allgemeinen nicht optimal, denn es entstehen *überflüssige* und *störende* Punkte. Diese können aber mit Hilfe der neu entwickelten Methode aus Kapitel 3.3.3.2 eliminiert werden. Die im Rahmen dieser Arbeit neu entwickelte Eliminationsmethode verbessert fehlerbehaftete Triangulationen und stellt damit den Hauptbeitrag dieser Arbeit für den Gesamtprozess der Objekterkennung.

In jedem iterativen Schritt erfolgt für einen betrachteten Punkt P_i mit Index i in einem Zuge sowohl die Berechnung der lokalen Triangulation und entsprechende Eliminierung der stören-

den Nachbarpunkte zur Berechnung des Normalenvektors als auch die Abschätzung der Krümmung an dem Punkt P_i. Das wesentliche dabei ist, dass beide Berechnungen auf derselben nach der Eliminationsmethode optimierten lokalen Triangulation basieren, d.h. da die optimale Triangulation auf Grund der Elimination nun vorhanden ist, wird gleich nach der Normalenvektorberechnung der Satz von Gauß-Bonnet direkt auf die gemäß P_i erstellte optimale lokale Triangulation im Laufe vom Objekterkennungsprozess angewendet. Im Vergleich zu anderen Verfahren mit "*analytischen*" Berechnungen ergibt sich daher ein Effizienzvorteil. Der bevorzugte Satz von Gauß-Bonnet (Kapitel 3.3.4.2) berechnet die maximale und minimale Hauptkrümmung für Punkt P_i der Grundkörperoberfläche.

Der für einen beliebigen Punkt P_i mit Index i dargestellte Gesamtvorgang wird für alle Punkte aus der Punktwolke wiederholt durchgeführt, um die Normalenvektoren als auch die maximalen und minimalen Hauptkrümmungen an den einzelnen Punkten aus der Punktwolke zu berechnen. Nach dem die Berechnungen für alle Punkte aus der Punktwolke durchgeführt sind, werden anschließend die Segmentierungsschritte (normalenbasierte und krümmungsbasierte Segmentierung, siehe Kapiteln 3.3.5.1 und 3.3.5.2) hintereinander ausgeführt.

3.3 Der Ablauf des Objekterkennungsprozesses

Im folgenden Kapitel werden die Schritte im Prozess zur Objekterkennung erläutert. Ausgehend von einer Eingabedatei mit einer Punktwolke, werden die Nachbarschaftsbeziehungen der Punkte bestimmt. In der anschließenden Iteration über die Punkte werden in einem Durchgang die Normalenvektoren und Krümmungen an den Punkten aus der Punktwolke abgeschätzt. Im letzten Schritt findet die Segmentierung der Punktewolke statt.

3.3.1 Schritt 1: Die Eingabedatei des angewendeten Programms

Das Verfahren zur Objekterkennung arbeitet auf einer Menge von Oberflächenpunkten. Diese wird als Eingabedatei bereitgestellt. Das Gesamtobjekt kann dabei aus einzelnen oder zusammengesetzten Teilobjekten bestehen.

Um die Funktionalität des Programmes zur Objekterkennung zu testen und die Ergebnisse zu visualisieren, wird das in dieser Arbeit vorgestellte Verfahren auf eine generierte Punktewolke angewendet. Ein im Laufe dieser Arbeit erstellter Generator dient zur Erstellung künstlicher Punktwolken, die die Oberflächen des Gesamtobjektes beschreibt. Zur Generierung der Punktwolke werden vier geometrische 3D-Grundprimitive verwendet wie z.B. Kugel, Zylinder, Kegel und Würfel.

Während der Generierung wird darauf geachtet, dass die Verteilung der Punkte in dem 3D-Raum realistisch ist, so dass keine Ausreißer vorkommen und die Objektoberflächen keine Lücken beinhalten, somit eine rauschfreie (engl. noiseless) Punktwolke zur Verfügung steht, ansonsten würde die Aufgabenstellung über das Thema dieser Diplomarbeit hinausgehen. Mit "realistisch" ist hier auch gemeint, dass die Ausgabe der ungeordneten Punkte nach der Generierung deckungsgleich gestreut ist, als wäre sie mit einem Scanner erfasst (siehe Abbildung 24).

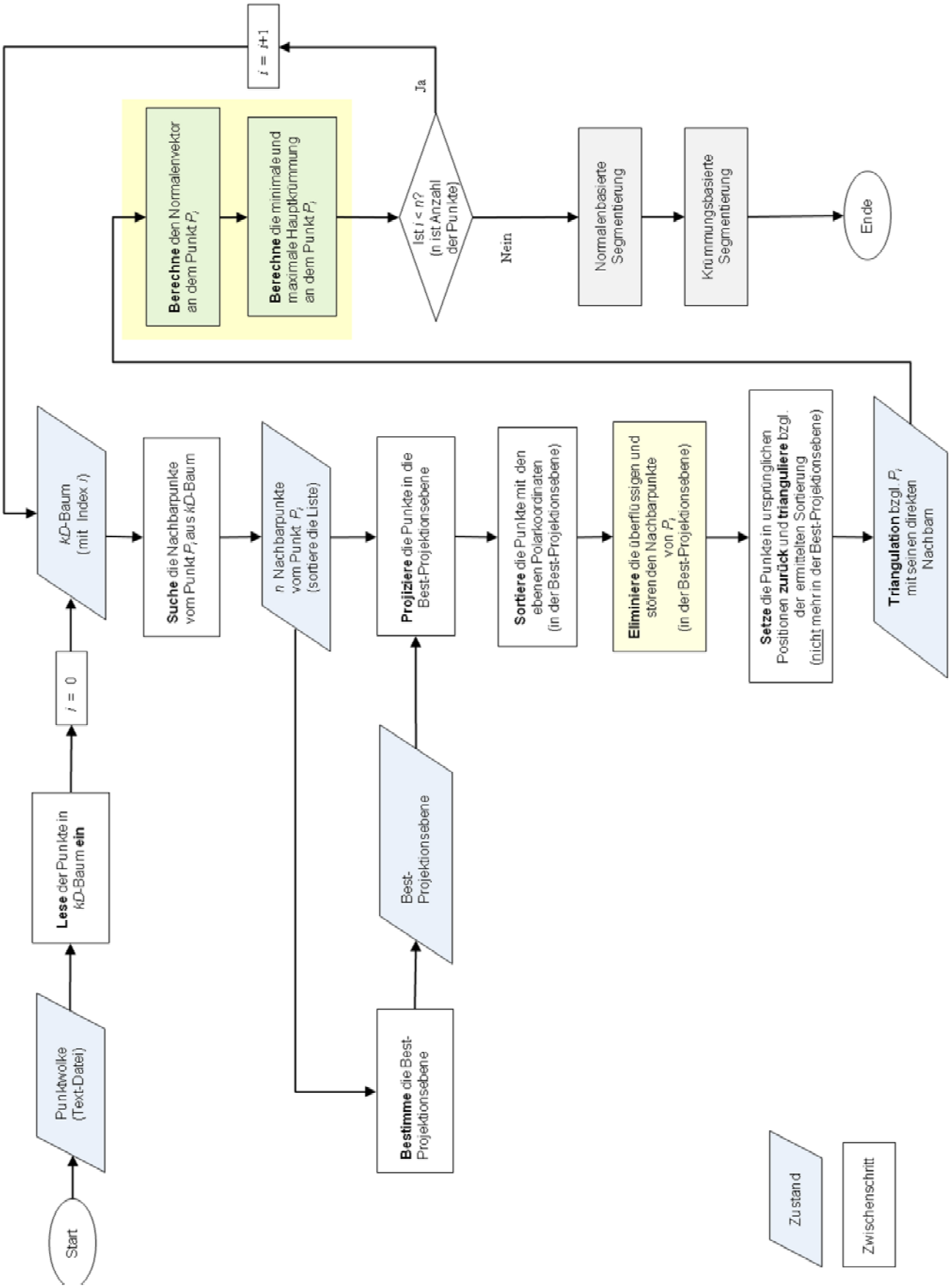


Abbildung 23: Ablaufdiagramm zur Merkmalermittelung einer Punktewolke

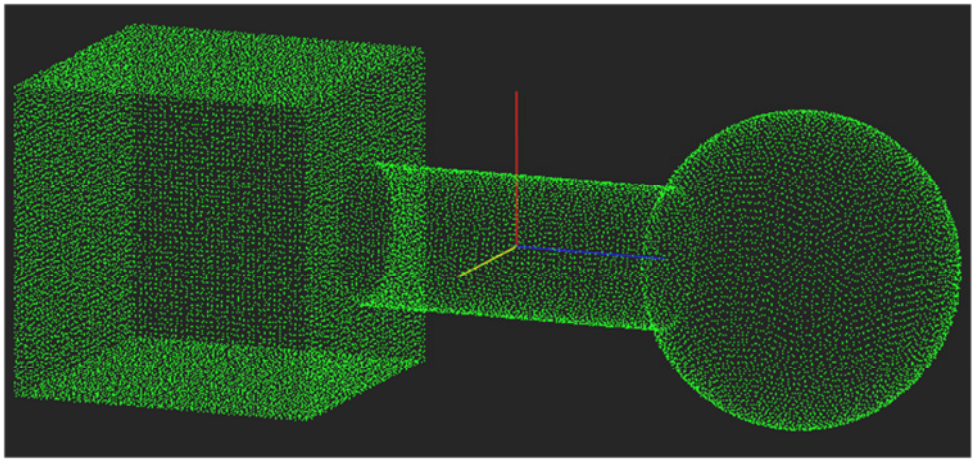


Abbildung 24: Beispiel für eine generierte Punktwolke mit geometrischen 3D-Grundprimitiven wie Würfel, Zylinder und Kugel

Zu Beachten ist dabei, dass zwischen den erfassten Punkten aus der Punktwolke keine feste Anordnung besteht, die Punkte also beliebig gestreut sind und keine weiteren Informationen zu den Punkten vorliegen, wie z.B. Nachbarschaftsbeziehungen, Kanten der Teilobjekte oder Übergänge zwischen Teilobjekten. Die Lage der Punkte ist nur durch die kartesischen *x*-, *y*- und *z*-Koordinaten angegeben. Daher wird im Rahmen dieser Diplomarbeit öfter von einer "*unstrukturierten Punktwolke*" gesprochen.

Der große Vorteil des im Rahmen dieser Diplomarbeit vorgestellten Objekterkennungsprozesses ist, dass der Prozess direkt auf eine vorliegende Punktwolke angewendet wird.

Die Eingabedatei beinhaltet in Textform zeilenweise die kartesischen Koordinaten der Punkte als Tripel (*x*, *y*, *z*). In der ersten Zeile steht die Gesamtanzahl der Punkte in der Punktwolke (siehe Abbildung 25).

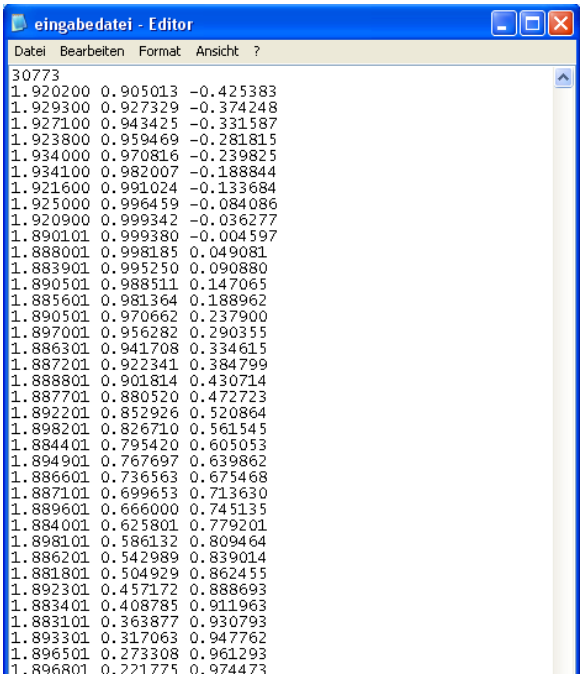


Abbildung 25: Eingabedatei mit dem Beispiel einer Punktewolke

Die Punktkoordinaten werden aus der Eingabedatei zeilenweise eingelesen und mit eindeutigen Indizes in einer Datenstruktur abgelegt. Bei der Datenstruktur handelt es sich um einen *kD*-Baum, so dass die Nachbarpunkte eines beliebigen Punktes P_i mit Index i während des Ablaufes

im Objekterkennungsprozess abgefragt werden können. Diese Abfragemöglichkeit ist notwendig, um lokale Triangulationen um einen beliebigen betrachteten Punkt mit seiner Nachbarschaft durchzuführen.

3.3.2 Schritt 2: Nachbarschaftssuche

Sowohl zur Berechnung der Normalenvektoren als auch zur Abschätzung der Krümmungen für die einzelnen Punkte der gescannten Objektoberfläche werden die Nachbarn des jeweiligen Punktes als gegeben vorausgesetzt.

Da die vorliegende Punktwolke in der Eingabedatei unstrukturiert gespeichert ist, kann aus dieser unstrukturierten Punktwolke nicht auf die Nachbarschaftsbeziehungen zwischen den Punkten geschlossen werden. Deswegen wird für einen festen Punkt P_i aus $\mathbb{R}^3$ in der Punktwolke nicht wie gewöhnlich nach einem nächstgelegenen Punkt P_j (diese Problemstellung ist bekannt als die "Nächster-Nachbar-Suche" oder das "Nächster-Nachbar-Problem", sondern nach allen Nachbarpunkten $N(P_i)=\{P_1,..,P_n\}$ aus der gleichen Punktwolke, welche in einem bestimmten Bereich um Punkt P_i herum liegen, gesucht. Diese bekannte Aufgabenstellung wird als "*k*-Nächsten-Nachbarn-Suche" oder "*k*-Nächsten-Nachbarn-Problem" (engl. *k*-nearest neighbors problem) bezeichnet [PrSh85].

Erwähnenswert ist hier außerdem, dass die Nachbarsuche nicht nur für einen Punkt, sondern für jeden Punkt aus der Punktwolke durchgeführt wird. Dies stellt eine weitere Herausforderung dar, wenn wie in unserem Beispiel ca. 30770 Punkte und in anderen Beispielen Millionen von Punkten der Reihe nach betrachtet werden sollen.

a) Aufbau des *kD*-Baumes

Um die Speicherung und Verwaltung der Punkte aus der Eingabedatei effizient zu lösen, werden die Punkte aus der unstrukturierten Punktwolke zunächst vorstrukturiert. Diese Vorstrukturierung erfordert eine Datenstruktur, mit der die Punkte effizient einsortiert und die Suche nach den Nachbarn effizient bewerkstelligt werden kann. Zum Erzeugen einer solchen Datenstruktur gibt es in der Informatik viele Beispiele und Ansätze. Eine in der Computergrafik häufig verwendete Datenstruktur ist der *kD*-Baum (engl. *k-d*-tree).

Der *kD*-Baum ist eine Erweiterung des binären Suchbaumes [BKOS00] zur Speicherung der betrachtenden Punkte aus $\mathbb{R}^3$ [Sedg91][PrSh85][Bent75]. Es gibt zwei Typen von *kD*-Bäumen, die als homogen und inhomogen bzw. heterogen bezeichnet werden. Bei homogenen *kD*-Bäumen werden innerhalb der Knoten die Punktekoordinaten und jeweils zwei Verweise auf den linken und rechten Nachfolgerknoten gespeichert. Dagegen werden bei inhomogenen *kD*-Bäumen die Punkte nur in den *Blättern* gespeichert. Die inneren Knoten enthalten bei inhomogenen *kD*-Bäumen je nach Ebene Schlüsselinformationen, die darüber entscheiden, in welchem Teilbaum der gesuchte Punkt zu finden ist und Verweise auf weitere innere Knoten. In dieser Arbeit wird ein heterogener *kD*-Baum eingesetzt.

Zum Aufbau des *kD*-Baumes wird der Suchraum jeweils in zwei möglichst gleichgroße Teilräume unterteilt. Die Unterteilung erfolgt an Hand einer Hyperebene in 3D, die zu einer von den drei globalen Koordinatenachsen aufgespannten Ebene parallel liegt. Die Hyperebene wird in einem inneren Knoten als Schlüsselinformation gespeichert. Die neu entstandenen Teilräume werden rekursiv solange aufgeteilt, bis jeder resultierende Teilraum mindestens einen und höchstens zwei Punkte enthält. Der Wert einer Hyperebene in jeder *j*-ten Ebene $j=\{1, 2, 3\}$, entscheidet, ob ein Punkt zu dem linken oder rechten Teilbaum gehört. Der Wert der Hyperebene zu einer Ebene E_j wird durch den Median der *n* Punkte ermittelt, die im nächsten Teilraum in der *j*-ten Koordinate liegen [Sedg91].

In Abbildung 26 ist die Struktur eines *kD*-Baumes in 3D zu sehen. *kD*-Bäume in 2D lassen sich unmittelbar auf mehr als zwei Dimensionen verallgemeinern [Sedg91].

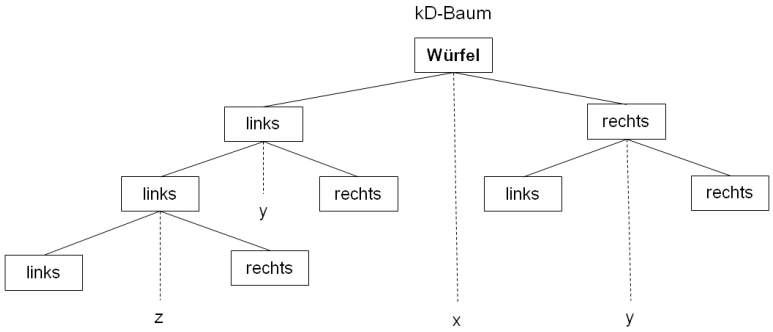


Abbildung 26: Struktur eines *kD*-Baum in 3D

Die mit diesem Algorithmus erzeugte Datenstruktur wird als 3D-Baum bezeichnet, da hier die Punkte aus dem 3-dimensionalem Raum gespeichert werden. In den Abbildung 27 werden die Aufteilung des Suchraums (links) und der Aufbau eines *kD*-Baumes (rechts) anhand eines einfachen Beispiels für den 2-dimensionalen Fall mit einer Suchebene veranschaulicht. In 2D redet man von Hypergeraden anstatt Hyperebenen, welche die Ebene in zwei ungefähr gleich große Teilebenen aufteilen. Die Hypergeraden werden durch den Median von *n* Punkten jeweils in x- und y-Koordinaten bestimmt [Sedg91][Bent75].

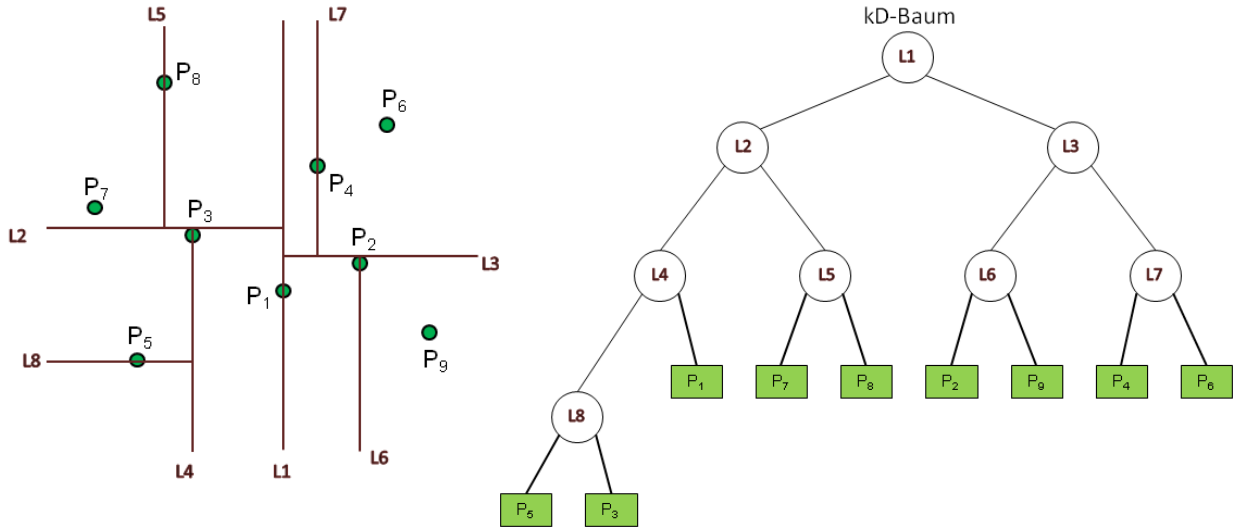


Abbildung 27: *kD*-Baum: (links) Aufteilung des Suchraumes, (rechts) Aufbau des *kD*-Baumes

b) Suche nach Nachbarn

Während in 3D bei der Suche nach allen Nachbarpunkten zu Punkt P_i der Suchbereich durch eine Kugel mit dem Radius *r* definiert wird, in dem sich die gesuchten Nachbarpunkte befinden und in dessen Mitte der betrachtende Punkt P_i liegt, wird dieser Bereich in 2D durch einen Kreis begrenzt (siehe Abbildung 28). Man sucht daher genau nach den Punkten als Nachbarn, die in dem um den Punkt P_i gelegten Kreis mit dem vorgegeben Radius *r* liegen.

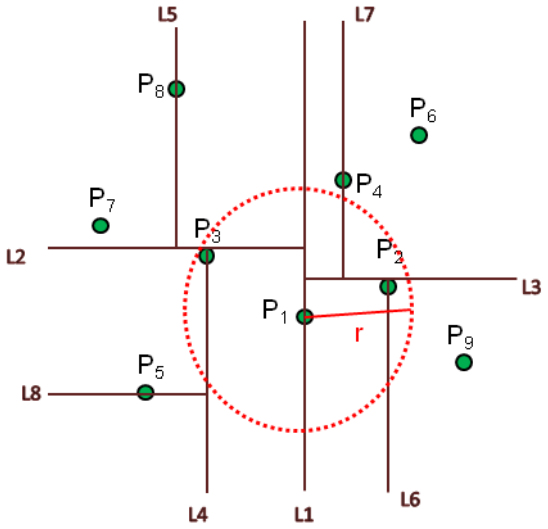


Abbildung 28: Suche nach den Nachbarn zu dem Punkt P_1

Bei der Suche nach Nachbarpunkten wird der *kD*-Baum durchlaufen und überprüft welche Teilebenen (in 3D: Teilräume) sich mit dem Kreis (in 3D: Kugel) überschneiden (siehe den rot gestrichelten Kreis in Abbildung 28) oder in diesem Kreis enthaltend sind. Es werden dabei nur die Knoten besucht, deren Teilebenen vom Kreis geschnitten werden [Sedg91][Bent75].

Im Rahmen dieser Arbeit wurden einige vorgefertigte Implementierungen aus verschiedenen Quellen zur Lösung dieses Problem herangezogen und gemäß den hier gegebenen Bedürfnissen untersucht. Die Implementierung aus [Mile11] wurde auswählt und entsprechend der Aufgabenstellung angepasst und integriert. Neben üblichen Anpassungen war das Versehen der Punkte mit Indizes die wesentliche Optimierung, während der *kD*-Baum aufgebaut wurde.

Um Missverständnisse zu vermeiden, wird das *k*, welches in der Bezeichnung "*k*-Nächsten-Nachbarn-Suche" unterscheiden von dem *k*, das in dem Begriff "*kD*-Baum" vorkommt. Das *k* in der Bezeichnung des *kD*-Baumes gibt die Anzahl der Dimensionen des Binärbaumes wieder. In unserem Fall ist $k = 3$, deswegen kann man anstatt *kD*-Baum auch von einem 3D-Baum reden. Das in der Bezeichnung "*k*-Nächsten-Nachbarn-Suche" vorkommende *k* gibt dagegen die Anzahl der Nachbarpunkte wieder, nach denen für einen festen Punkt gesucht wird. In dem hier dargestellten Prozess ist die Anzahl der Nachbarpunkte nicht festgelegt, d.h. mit Hilfe des *kD*-Baumes wird nicht nach den *k* nächsten Nachbarn sondern nach allen Nachbarn von Punkt P_i gesucht, die sich in einem Bereich befinden, in dessen Mittelpunkt P_i liegt. Diesen Bereich begrenzt in 3D eine Kugel mit einem vordefinierten Radius *r*.

3.3.3 Schritt 3: Berechnung der "Normalenvektoren der Punkte"

Da der zu verarbeitende Datensatz in dem vorgestellten Prozess aus unstrukturierten Punkten ohne Nachbarschaftsinformationen zwischen den Punkten besteht und keine geometrischen Eigenschaften der Oberfläche in den Punkten vorhanden sind, muss man aus der Punktewolke in irgendeiner Art und Weise sowohl die Nachbarschaftsbeziehungen ermitteln als auch die geometrischen Merkmale wie Normalenvektor und Krümmung in den Punkten der Objektoberfläche extrahieren.

In diesem Zusammenhang lassen sich die Verfahren zur Abschätzung der Normalenvektoren an den einzelnen Punkten aus einer Punktwolke grob in zwei Gruppen unterteilen; *analytische* und *numerische* Verfahren [Gomo09][Vanc03][Wilk02].

Diese Verfahren setzen eine hinreichend große Anzahl von Nachbarpunkte voraus. Das heißt, alle Verfahren ziehen einen Punkt, zu dem der Normalenvektor berechnet wird und seine Umgebung in Betracht. Die Ermittlung der Nachbarschaft wird in dieser Arbeit durch *kD*-Bäume (siehe Kapitel 3.3.2) erreicht.

Bei einem analytischen Verfahren wird eine Menge von Punkten, die auf der Oberfläche eines Objekts liegen, approximiert. Das Ergebnis ist eine angepasste mathematisch definierte Fläche, welche die approximierten Punkten beschreibt. Daher nennt man diese Verfahren auch Approximationsverfahren [Gomo09]. Um diese Fläche zu bekommen, muss man zuerst die *Ausgleichsebene* (für die Berechnung siehe Appendix E) bestimmen. Der Normalenvektor der Ausgleichsebene, der auch gleichzeitig der gesuchte Normalenvektor im Punkt *P* ist, ergibt sich dann als Eigenvektor zum kleinsten Eigenwert.

Bei den meisten numerischen Verfahren geht es um die Triangulation einer Oberfläche (siehe Kapitel 2.5). Dabei wird die Punktwolke entweder vollständig oder lokal trianguliert. Danach werden die Algorithmen auf die Punkte angewendet, deren Nachbarschaftsbeziehungen aus der vorhergehenden Triangulation bekannten sind. Dies ermöglicht eine näherungsweise Darstellung des Ganzen oder eines Teils der Objektoberfläche mit Dreiecken und deren Betrachtung. Daher sind solche Verfahren in der Literatur auch als Näherungsverfahren oder Lageverfahren [Gomo09] bekannt.

Um das numerische Verfahren, bei dem die gegebene Punktwolke vollständig trianguliert wird, von den numerischen Verfahren, welche die *lokalen* Triangulationen einsetzen, zu unterscheiden, wird im Rahmen dieser Arbeit eine vollständige Triangulation als *globale* Triangulation benannt.

Bei numerischen Verfahren, welche die globale Triangulation einsetzen, werden die Algorithmen auf die triangulierte Punktwolke angewendet und dabei die Vertices des Dreiecknetzes betrachtet. Im Gegensatz zu einer globalen Triangulation wird bei einer lokalen Triangulation nur die Umgebung *eines* betrachteten Punktes bezüglich seiner Nachbarschaft rekonstruiert. Dieser Rekonstruktionsvorgang ist aber für jeden Punkt aus der Punktwolke durchzuführen. Die lokalen Triangulationen sind temporär, d.h. sie sind nur zum Zweck der Abschätzung der geometrischen Merkmale an dem jeweiligen Oberflächenpunkt nötig und brauchen nicht langfristig gespeichert zu werden.

Das Ergebnis einer globalen Triangulation ist ein Dreiecksnetz, welches mit seiner speziellen Struktur abgelegt wird und während des Objekterkennungsprozesses zur Verfügung stehen soll. Neben diesem Nachteil der numerischen Verfahren mit der globalen Triangulation sind in der Regel deren Dreiecknetze hinsichtlich der Nachbarschaft der betrachteten Punkte nicht eindeutig. Dieses Problem ist mit einigen Beispielen unten näher veranschaulicht.

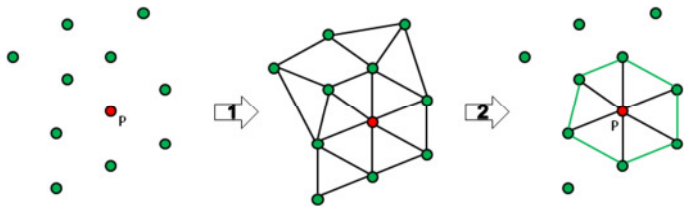


Abbildung 29: Triangulation mit Verfahren 1

Angenommen, es gäbe zwei globale Triangulationsverfahren mit unterschiedlichen Algorithmen, Verfahren 1 und Verfahren 2.

Ein mögliches Dreiecksnetz könnte nach Verfahren 1 wie in Abbildung 29 dargestellt aussehen, während nach Verfahren 2 ein Dreiecksnetz wie in Abbildung 30 entstünde, weil in dem Netz

z.B. eine Kante (blau gestrichelte Linie) fehlt. Dieser Unterschied könnte die Berechnung der Normalenvektorrichtung an dem entsprechenden Punkt *P* (rot) beeinflussen.

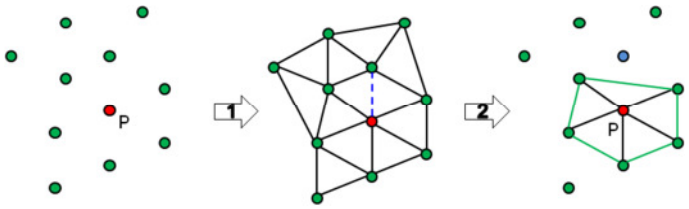


Abbildung 30: Triangulation mit Verfahren 2

Das Problem kann man mit den folgenden zwei Abbildungen leicht veranschaulichen. Seien in Abbildung 31 dieselbe, mit verschiedenen Verfahren triangulierte, Punktmenge aus vier Punkten dargestellt. Der Punkt *P* sei dabei der betrachtete Punkt. Nach gültigen Triangu-lationen entstehen bei der Verwendung des einen Verfahrens zwei *direkte* Nachbarpunkte (Abbildung 31-links), während bei der Verwendung des zweiten Verfahrens (Abbildung 31-rechts) drei *direkte* Nachbarpunkte (siehe Appendix D) zu dem betrachteten Punkt *P* entstehen.

Abbildung 31: Triangulation: (links) nach Verfahren 1, (rechts) nach Verfahren 2

Diese Unterschiede würden die Berechnung der geometrischen Eigenschaften an den jeweiligen Oberflächenpunkten verfälschen, so dass der betroffene Punkt zu einem falschen Segment zugeordnet würde.

Dieses einfache Beispiel zeigt, dass auch die mit Hilfe der Vortriangulation (globale Triangulation) gewonnenen geometrischen Merkmale zu falschen Resultaten führen und das die Klassifizierung der Punkte erschweren könnte. Eine bessere Lösung bieten numerische Verfahren mit der lokalen Triangulation, in der möglichst viele Umgebungspunkte eines betrachteten Punktes zur Erstellung der Triangulation einer Teiloberfläche mit einbezogen werden. Dies ist wiederum auch ein Nachteil von so einem Verfahren, weil dabei auch nicht-gewünschte Punkte (für "*überflüssige*" oder "*störende*" Punkte, siehe Kapitel 3.3.3.2) vorkommen, die mit zusätzlichen Methoden während des Prozesses unbedingt beseitiget werden müssten. Ansonsten würden solche Punkte die Richtung eines Normalenvektors manipulieren und auch die Abschätzung der Krümmung an dem betrachteten Punkt verfälschen. Solche störenden Nachbarpunkte treten besonders bei scharfen Kanten und bei Übergängen zwischen geometrisch unter-schiedlichen Grundprimitiven auf. Diese werden im Abschnitt 3.3.3.2 von der neu entwickelten Methode aufgespürt und in dem Vorfeld aus den nicht-optimalen lokalen Triangulationen entfernt.

Es gibt zahlreiche Methoden mit unterschiedlichen Lösungsansätzen in der Literatur, um die Normalenvektoren an den einzelnen Punkten aus der Punktwolke zu berechnen [Vanc03] [Wilk02]. Zur Abschätzung der geometrischen Merkmale wie Normalenvektor und Krümmung sind die numerischen Verfahren im Vergleich zu den analytischen Verfahren ziemlich effizient. Daher wurden in Rahmen der Diplomarbeit numerische Verfahren (siehe Kapiteln 2.5 und 3.3.3) vorgezogen.

Die Problematik bei den numerischen Lösungsansätzen im Allgemein ist aber, dass an (scharfen) Kanten und bei Übergängen zwischen geometrischen 3D-Grundprimitiven keine gute Ergebnisse erzielt werden. Daher war die hauptsächliche Bestrebung dieser Arbeit, die genann-

ten Methoden zu implementieren und die Probleme, die bei der Analyse erkannt wurden, zu beheben. Im Kapital 3.3.3.2.1 werden dazu Lösungsansätze präsentiert. Hier wird vorgestellt, wie ein Normalenvektor in einem Punkt *P*, der in einer bestimmten Oberfläche eines 3D-Grundprimitives liegt, numerisch berechnen wird.

Erwähnenswert ist es, dass im Rahmen dieser Diplomarbeit unterschiedliche Normalen-vektoren an verschiedenen Stellen des Prozessablaufes auftreten. Einerseits ein Normalenvektor der Best-Projektionsebene, in die alle Nachbarpunkte eines betrachteten Punktes *P* projiziert werden (Abschnitt 2.3.1). Andererseits kommt ein Normalenvektor in der Dreiecksfläche vor, die durch drei aufeinanderfolgende Punkte aufgespannt ist und mit deren Hilfe die Punkte aus 3D in 2D-Ebene übertragen werden, um die Orientierung der Fläche festzustellen (siehe Abschnitt 2.4). In diesem Abschnitt wird aber ein Normalenvektor *an einem Punkt* aus der Punktwolke berechnet, der für den Objekterkennungsprozess eine große Rolle spielt. Die Normalenvektoren werden eingesetzt, um die Kanten und Übergängen zwischen geometrisch unterschiedlichen Grundprimitiven im dieser Arbeit vorgestellten Objekterkennungsprozess zu extrahieren. Solche Normalenvektoren werden jeweils anhand der lokalen Triangulation durch einen betrachteten Punkt *P* und mit seinen Nachbarpunkten berechnet.

Um geometrische Oberflächenmerkmale an den einzelnen Oberflächenpunkte abzuschätzen, wurden in dieser Arbeit zwei Verfahren aus [Vanc03] zur lokalen Triangulation implementiert und anschließend analysiert. Die beiden Verfahren wiesen besonders an den scharfen Kanten und bei Übergängen zwischen unterschiedlichen geometrischen Grundprimitiven Schwächen auf. In Kapitel 3.3.3.2 wurde auf diese Schwächen eingegangen und der eingeführte Lösungsansatz detailliert beschrieben.

Nach der Implementierung der oben genannten Verfahren wurde das Verfahren *Local Center Triangulation* im Vergleich zum *Local Delaunay Triangulation* als der bessere Lösungsansatz zur lokalen Triangulation gesehen. Deshalb wurde das Verfahren *Local Center Triangulation* strukturell anpasst bzw. erweitert und das neue Verfahren *Lokale Triangulation mit direkten Nachbarpunkten* benannt. Der neue Algorithmus enthält unter anderem eine neu entwickelte Eliminationsmethode, welche die ursprünglichen Schwächen behebt. Weitere Information über *Local Center Triangulation* bzw. *Local Delaunay Triangulation* sind im Kapitel 2.5 zu finden.

Die Punktwolken werden mittels Berechnung der Normalenvektoren an einzelnen Punkten für die Verwendung bei der normalenbasierten Segmentierung aufbereitet.

Im nächstem Kapitel wird das optimale, lokale Triangulationsverfahren *Lokale Triangulation mit direkten Nachbarpunkten* ausführlich erläutert und mit einem Beispiel einer willkürlichen Punktmenge gezeigt, wie das Ergebnis einer nicht-optimalen lokalen Triangulation gegenüber dem *Lokale Triangulation mit direkten Nachbarpunkten* für einen betrachteten Punkt *P* durch Betrachtung seiner Umgebung aussieht. Anschließend wird in einem separaten Unterkapitel die neu entwickelte Eliminationsmethode beschrieben, welche beim Verfahren *Lokale Triangulation mit direkten Nachbarpunkten* eingesetzt und für eine optimale lokale Triangulation nötig ist.

3.3.3.1 Lokale Triangulation mit direkten Nachbarpunkten

Das ursprüngliche Verfahren stammt aus [Vanc03] und wurde im Rahmen dieser Diplomarbeit erweitert. Das dadurch entstandene Verfahren wurde als *Lokale Triangulation mit direkten Nachbarpunkten* bezeichnet. Die Anpassungen und Erweiterungen sind in den folgenden Abschnitten zu finden. Der Pseudocode des angepassten und optimierten Verfahrens lautet wie folgt:

Sei der Punkt P der betrachtende Punkt und die Punkte $P_1, P_2, ..., P_k$ seine Nachbarpunkte. In diesem Verfahren wird angenommen, dass der Mittelpunkt MP der Nachbarpunkte nahe an P liegt. In diesem Fall verhält sich MP wie der Punkt P daher korrespondiert der Normalenvektor von MP näherungsweise dem gesuchten Normalenvektor zu Punkt P.

- Berechne den Mittelpunkt MP aus den Nachbarn von P

$$MP = \frac{1}{k}\sum_{i=1}^{k} \mathrm{P_i} \tag{3.1}$$

- Trianguliere die Nachbarpunkte nach MP:
  1. Finde eine geeignete Projektionsebene E (siehe Kapitel 2.3.1).
  2. Projiziere die Punkte P_i in die Ebene E mit dem "*Aufklappen*"-Ansatz; die projizierten Punkte werden als P_i' bezeichnet (siehe Kapitel 3.3.3.2.2).
  3. Sortiere die "*aufgeklappten*" Punkte P_i' nach ihren Polarkoordinaten mit $\overrightarrow{MPP_1}$ als *Polarachse*. Die sortierten Punkte werden als P_j^s bezeichnet.
  4. Bilde einen Polygonzug PZ in der Best-Projektionsebene aus P_j^S (siehe Abbildung 36)
  5. Eliminiere die störende Punkte in der Best-Projektionsebene (siehe Kapitel 3.3.3.2.3)
  6. Erstelle eine Triangulation T in 3D aus dem Polygonzug PZ (jeder Punkt P_j^s entspricht einem Punkt P_i in 3D): das Resultat sind Dreiecke T_j erstellt aus (P_i, MP, $P_{(i \bmod k)+1}$), $j = 1, ..., k$
- Berechne den Normalenvektor $\vec{N}_\mathrm{i}$ für jedes 3D-Dreieck (siehe Kapitel 3.3.3)
- Den Normalenvektor $\vec{N}$ in MP erhält man aus dem Durchschnitt der k Dreiecke:

$$\vec{N} = \frac{1}{k}\sum_{i=1}^{k} \vec{N}_\mathrm{i} \tag{3.2}$$

- Verwende $\vec{N}$ als den geschätzten Normalenvektor in P

An dem Verfahren aus [Vanc03] (siehe Abschnitt 2.5.2) wurden folgende Anpassungen und Erweiterungen vorgenommen:

i. Anstatt der Bezeichnung Center C_m in Algorithmus aus [Vanc03] wird hier der *Mittelpunkt MP* benutzt, der durch die Mittelung der Nachbarpunkte errechnet wird. Dieser Mittelpunkt ist nicht wie in dem ursprünglichen Verfahren konstant, d.h. der Mittelpunkt wird jedes Mal neu berechnet, wenn ein Punkt aus der Nachbarschaft des betrachteten Punkt P mit der Eliminationsmethode entfernt wird, um die direkten Nachbarpunkte des Punktes P zu ermitteln. Daher ist der Mittelpunkt im neuen Verfahren nicht nur für die Sortierung der Nachbarpunkte wichtig, sondern auch für die Elimination der indirekten Nachbarpunkte eines betrachteten Punktes.

ii. Man braucht die Punkte nicht in 2D zu speichern, nach dem die Punkte in die Best-Projektionsebene projiziert sind. Daher sind in dem angepassten Algorithmus die Nachbarpunkte direkt in der Best-Projektionsebene sortiert. Folglich sind die im zugrunde gelegten Pseudocode als P_j^{2D} bezeichneten Punkte hier als P_i' (siehe Abbildung 32, Abbildung 33 und Abbildung 34) gekennzeichnet.

iii. Die Nachbarpunkte eines betrachteten Punktes P werden in die ermittelte Best-Projektionsebene nicht nur durch eine "einfache senkrechte" Projektion, sondern zusätzlich "*aufgeklappt*" übertragen (siehe Kapitel 3.3.3.2.2).

iv. Die erste Triangulation, die im ursprünglichen Algorithmus in 2D als T_2 bezeichnet wird, wird nach der Anpassung als *Polygonzug PZ* bezeichnet, der in der Best-Projektionsebene

in 3D erstellt wird. Das Ziel des neuen Verfahrens ist das Finden des konvexen Polygonzuges unter der Annahme, dass ein konvexer Polygonzug aus direkten Nachbarpunkten des betrachteten Punktes besteht.

v. Da im neuen Algorithmus der 3D-Raum nicht verlassen wird, um die Punkte zu sortieren, werden die sortierten Punkte als P_j^S anstatt P_j^{2D} bezeichnet (das *S* steht dabei für "sortiert"). In neuem Algorithmus werden die projizierten Punkte in der Best-Projektionsebene nach ihren Polarwinkeln in 3D sortiert.

vi. Ganz neu in diesem Algorithmus ist die Elimination der *störenden* Nachbarpunkte in einer Triangulation (siehe Kapitel 3.3.3.2), die in einer Best-Projektionsebene durchgeführt wird. Durch Elimination der störenden Nachbarpunkte wird eine lokale Triangulation optimiert, so dass sowohl der Normalenvektor als auch die Krümmung mit Hilfe des Satzes von Gauß-Bonnet an einem betrachteten Punkt *in einem Schritt* berechnet werden, ohne aufwendige analytische Lösungsansätze einzusetzen.

Die *Lokale Triangulation mit direkten Nachbarpunkten* ist ein Triangulationsverfahren, das auf Sortierung basieret. Das gesuchte Dreiecksnetz wird durch Sortierung der Nachbarn vom betrachteten Punkt *P* in einer Ebene (Best-Projektionsebene) erstellt.

Um Punkte zu sortieren, die im 3D-Raum beliebig verstreut sind, gibt es mehrere Methoden. Beispielsweise können die Punkte nach Winkel in einer Ebene sortiert werden. Dazu wird allerdings eine geeignete Ebene benötigt und die Punkte müssen auf angemessene Weise auf diese Ebene gebracht werden. Diese geeignete Ebene wird als Best-Projektionsebene bezeichnet (siehe Kapitel 2.3.1). Erst wenn die Best-Projektionsebene gefunden ist, kann die Triangulation beginnen.

Seien die Punkte $P_1,\ldots,P_8$ die Nachbarpunkte des betrachtenden Punktes *P* im dreidimensionalen kartesischem Raum wie in Abbildung 32 dargestellt.

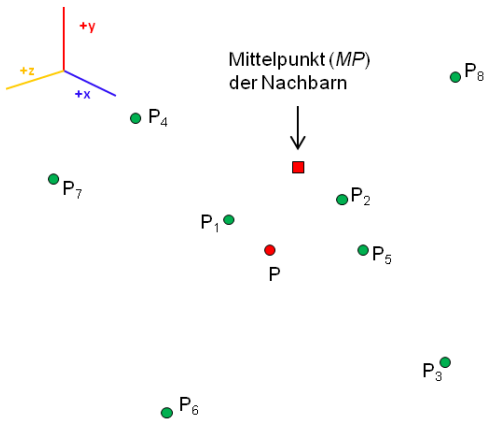


Abbildung 32: Punkte und Mittelpunkt der Nachbarn in 3D-Raum

Bei der Beschäftigung mit der Sortierung gegebener Punkte und ihrer Triangulation, hat sich erfahrungsgemäß herausgestellt, dass bei der Sortierung ohne Mittelpunkt und anschließender Triangulation, das Ergebnis ungenau wird. Eine präzisere Triangulation wird möglich, in dem man ein Zentrum von allen Nachbarpunkten verwendet. Dieses Zentrum wird in unserem Beispiel *Mittelpunkt MP* (siehe Abbildung 33) genannt. Der Mittelpunkt wird durch die arithmetische Summe der Nachbarpunkte von Punkt *P* berechnet und wie alle anderen Punkte in die Best-Projektionsebene abgebildet. Dieser Punkt liegt nach [Vanc03] in der Nähe von *P* und verhält sich daher auch wie *P*.

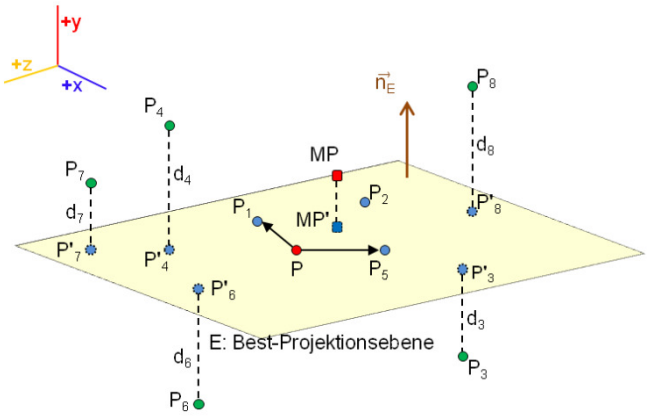


Abbildung 33: Projizierung der Punkte und deren Mittelpunkt (*MP*) in die Best-Projektionsebene

Wie aus Kapitel 2.2 hervorgeht, ist eine Achse nötig, um die Punkte mit Hilfe der ebenen Polarkoordinaten nach dem gegen den Uhrzeigersinn aufsteigenden Winkel zu sortieren. Diese Achse wird zwischen den Punkten *MP*' und P_1 (der am Nahesten zu *P* liegende Nachbarpunkt) als Vektor definiert und als Polarachse genannt (Abbildung 34).

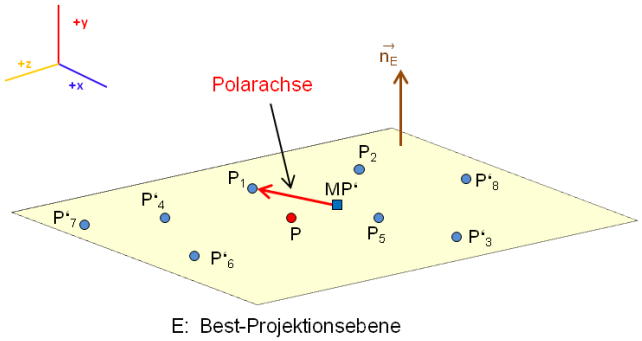


Abbildung 34: Alle Punkte in der Best-Projektionsebene mit Polarachse

Die Polarachse zwischen dem projizierten Mittelpunkt *MP*' und Punkt P_1 teilt die Ebene im Allgemeinen in zwei Bereiche (siehe Abbildung 35). Da sinθ den Wert zwischen 0 und 1 nimmt und daher der Winkel θ zwischen 0° und 180° liegt, gilt es, die Position gemäß der Polarachse festzustellen, ob ein projizierter Nachbarpunkt während der Sortierung auf der rechten oder linken Seite der Polarachse liegt, wenn die Ebene visuell durch eine Gerade in zwei Teile aufgeteilt wird.

$$|\vec{a} \times \vec{b}| = |\vec{a}|\,|\vec{b}| \sin\theta \qquad sin\theta \geq 0 \;\Rightarrow\; 0° \leq \theta \leq 180° \tag{3.3}$$

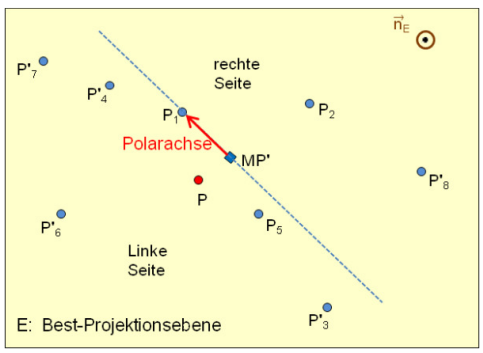


Abbildung 35: Blick von oben in die Best-Projektionsebene nach der Projektion

Die Positionen der einzelnen Nachbarpunkte wird durch die im Kapitel 2.4 vorgestellte Methode festgestellt. Beispielweise liegt ein projizierter Nachbarpunkt P_i' ($i = 2,\ldots,8$) auf der linken Seite der durch die Polarachse aufgeteilten Ebene, dann ist der Umlaufsinn der Punkte der Reihe nach P_1-MP'- P_i' linksdrehend (siehe Kapitel 2.3.2), ansonsten rechtsdrehend. Das wird mit Hilfe des Kreuzproduktes in 2D ermittelt (siehe Kapitel 2.3.3 Abschnitt a)).

In Abbildung 36 ist ein Polygonzug (siehe Appendix D) in der Best-Projektionsebene dargestellt, der nach der Sortierung der Nachbarpunkte von Punkt *P* zustande kommt. Dieser Polygonzug entspricht dem Polygonzug, welcher in der Graham-Scan-Methode (siehe Abschnitt 2.3.4) erstellt wird, um die konvexe Hülle zu berechnen. Im Unterschied zur Graham-Scan-Methode erfolgt hier die Sortierung anhand der Best-Projektionsebene im 3D-Raum und nicht in einer Ebene in dem zwei-dimensionalen Raum.

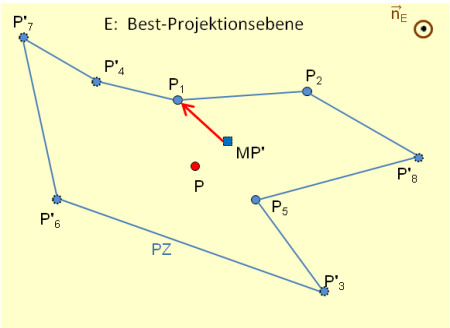


Abbildung 36: Polygonzug (*PZ*) nach der Sortierung der Nachbarpunkte in der zugehörigen Best-Projektionsebene

Der resultierende Polygonzug wird später mit der Methode aus Abschnitt 3.3.3.2 zum Auffinden von *überflüssigen* Punkten in der Best-Projektionsebene eingesetzt, mit dem Ziel die *störenden* Punkten zu eliminieren, die während der lokalen Triangulation entstehen Dabei werden die Nachbarpunkte eines betrachteten Punktes aus der Punktwolke durch "*Aufklappen*"-Ansatz (siehe Abschnitt 3.3.3.2.2) in die Best-Projektionsebene übertragen, in der die Topologie bezüglich der Orientierung der Dreiecksfläche, die bei dem zugehörigen Polygonzug vorkommen, nicht verloren geht. Diese Eliminationsmethode ist der Hauptbestandteil des neuen optimierten Verfahrens *Lokale Triangulation mit direkten Nachbarpunkten* und wird in dem folgenden Kapitel detailliert behandelt. Hier wird mit einem Beispiel, bei dem die Punkte ungeordnet im 3D-Raum verstreut sind, kurz illustriert, wie triangulierte Nachbarpunkte um einen betrachteten Punkt *P* durch das Verfahren *Local Center Triangulation* aus [Vanc03] ohne Eliminationsmethode aussehen würden.

Nach dem die Punkte in der Best-Projektionsebene an Hand der Polarwinkel aufsteigend sortiert sind, werden die Punkte wieder zurück in ihre ursprünglichen Positionen gebracht, um die tatsächliche lokale Triangulation um den betrachtenden Punkt *P* mit seinen Nachbarpunkten im 3D-Raum zu erstellen. Dabei wird das Dreiecksnetz entsprechend der Sortierreihenfolge aufgestellt, in dem die Randpunkte P_i' des Polygons den Punkten P_i ($i = 1, .., 8$) entsprechen. Das Ergebnis ist ein Dreiecksnetz, bei dem der Punkt *P* in jedem Dreieck vorkommt und in etwa in der Mitte des Netzes liegt.

Die Projektion aller Nachbarpunkte auf die Best-Projektionsebene und die anschließende Sortierung dieser Punkte führt zu einer nicht-optimalen Triangulation mit störenden Punkten.

In Abbildung 37 ist das Ergebnis einer nicht-optimalen lokalen Triangulation unter Verwendung von *Local Center Triangulation* mit allen Nachbarpunkten illustriert, bei dem auch störende Punkte auftreten. Eine Triangulation ist nicht optimal, wenn im Ergebnis solche Nachbarpunkte enthalten sind, die die Abschätzung der Merkmale an dem Oberflächenpunkt *P*, wie beispielsweise den Normalenvektor oder den Krümmungswert verfälschen. Solche Punkte werden in

Rahmen dieser Arbeit *störende* Punkte genannt und müssen für eine optimale Triangulation entfernt werden.

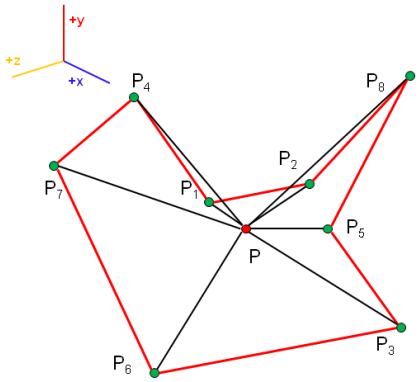


Abbildung 37: Beispiel für eine nicht-optimale lokale Triangulation in 3D mit allen Nachbarpunkten des betrachteten Punktes *P*

Die optimale lokale Triangulation des Beispiels aus Abbildung 37 ist in Abbildung 38 anhand des Dreiecknetzes mit dem grünen Polygonzug dargestellt.

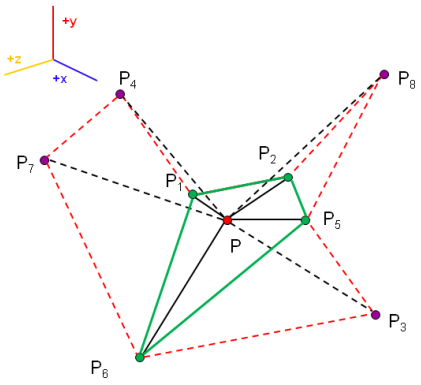


Abbildung 38: Triangulation nach dem Local Center Triangulation Verfahren aus [Vanc03] mit störenden Nachbarpunkten (rot-gestrichelter Polygonzug) vs. optimale Triangulation durch Lokale Triangulation mit direkten Nachbarpunkten um den betrachteten Punkt *P* (grün gefärbter Polygonzug) in 3D

Die Ergebnisse der bekannten lokalen Triangulationsverfahren wie z.B. *Local Center Triangulation* und *Local Delaunay Triangulation* aus [Vanc03] sind bei scharfen Kanten und Übergängen zwischen geometrisch unterschiedlichen Teilobjekten in der Punktwolke nicht zufriedenstellend. Aus diesem Grund werden häufig analytische Lösungen vorgezogen. Die analytischen Verfahren wiederum haben den Nachteil, dass sie ineffizient sind, weil sie mehrere Schritte benötigen, um die Merkmale an den einzelnen Punkten zu extrahieren.

Um die lokale Triangulation auch bei Punktwolken mit scharfen Kanten und Übergängen zwischen geometrisch unterschiedlichen Teilobjekten verwenden zu können, wird eine neu entwickelte Eliminationsmethode angewandt. Die Eliminationsmethode ist ein Teil des Verfahrens Lokale Triangulation mit direkten Nachbarpunkten und sie wird im nächsten Kapitel eingehend beschrieben.

3.3.3.2 Elimination der überflüssigen und störenden Nachbarpunkte

Den Schwerpunkt dieses Kapitels bildet eine im Rahmen dieser Diplomarbeit neu entwickelte Methode, welche auf einer Orientierungsanalyse der Dreiecksflächen in einem "*aufklappten*"

Polygonzug beruht. Ein Dreieck wird dabei durch drei aufeinanderfolgende Eckpunkte vom Polygonzug gebildet.

Diese Methode wird als *Eliminationsmethode* benannt und wird immer in einer Best-Projektionsebene durchgeführt. Hierbei geht es um Elimination der "überflüssigen" und "störenden" Punkte aus Sicht eines Bezugspunktes innerhalb der Triangulation.

Die Notwendigkeit dieser Methode ist aufgetreten nachdem die Anwendung der *Local Center Triangulation* und *Local Delaunay Triangulation* [Vanc03] unter bestimmten Punktverteilungen Schwächen aufwiesen. Das Problem lag darin, dass bei scharfen Kanten und Übergängen zwischen geometrisch verschiedenen Objekten die Oberflächenstruktur nicht richtig berücksichtigt wurde. Durch die Elimination der störenden und überflüssigen Punkte wird versucht die Oberflächenstruktur der geometrischen Objekte korrekt darzustellen.

Die zu eliminierenden Punkten lassen sich in zwei Gruppen einordnen; *störende* und *überflüssige* Nachbarpunkte. Die Punkte werden als *überflüssig* bezeichnet, wenn sie nicht die *direkten* Nachbarn vom betrachteten Punkt *P* sind. Die *störenden* Punkte sind die Punkte, bei denen der Normalenvektor der zugehörigen Dreiecksfläche die Richtung des Normalenvektors im betrachteten Punkt *P* verändern würde.

Daher wurde die Eliminationsmethode entwickelt, um eine lokale Triangulation mit den *direkten* Nachbarpunkten des betrachtenden Punktes *P* mit hoher Präzision erstellen zu können. Ansonsten würde der betrachtete Punkt durch eine falsch berechnete Richtung seines Normalenvektors zu einem falschen Segment zugeordnet werden.

3.3.3.2.1 Notwendigkeit einer Eliminationsmethode

In den Abbildungen 39 bis 43 wird anhand eines Fallbeispiels das grundsätzliche Problem der beiden lokalen Triangulationsverfahren *Local Delaunay Triangulation* und *Local Center Triangulation*, welche in Kapital 2.5 kurz vorgestellt wurden, illustriert.

In Abbildung 39 sei *P* der zu betrachtende Punkt (rot dargestellt), der in einer Würfelebene liegt und sich direkt vor einer "scharfen Kante" befindet. Seine Nachbarpunkte (grün) seien auf zwei Ebenen verstreut, welche über diese scharfe Kante miteinander verbunden sind. Die Kanten in Abbildung 39 (blaue Linien) existiert eigentlich nicht, sondern die Punkte aus der Punktwolke deuten nur drauf hin, wenn man die Punktwolke mit dem menschlichen Auge betrachtet. Das Aufspüren von solchen Kanten ist das Ziel der normalenbasierten Segmentierung (siehe Kapitel 3.3.5.1). Dies wird mit Hilfe der an den einzelnen Oberflächenpunkten berechneten Normalenvektoren erreicht.

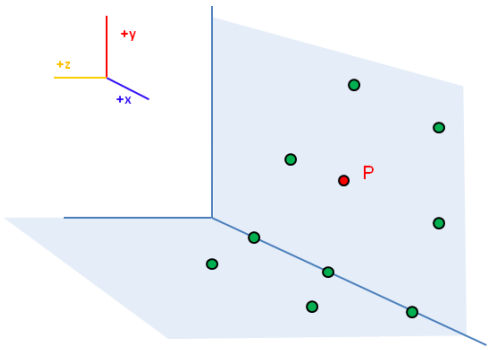


Abbildung 39: Ein betrachteter Punkt *P* direkt vor einer "scharfen Kante" mit seinen Nachbarn

Um die Nachbarpunkte hinsichtlich des Punktes *P* mit den lokalen Triangulationsverfahren verarbeiten zu können, werden alle Punkte in eine ermittelte Best-Projektionsebene (siehe Kapitel 2.3.1) projiziert. Nachdem alle Punkte in Ebene projiziert sind, werden sie beispielsweise bei

dem Verfahren *Local Center Triangulation* nach deren Polarwinkeln gemäß der Polarachse aufsteigend sortiert. In Abbildung 40 ist die Polarachse und die sortierten Punkte, welche aufsteigend durchnummeriert sind, zu sehen. Wenn man die sortierten Punkte der Reihe nach miteinander verbindet, erhält man einen Polygonzug (der blaue Zug, in Abbildung 40).

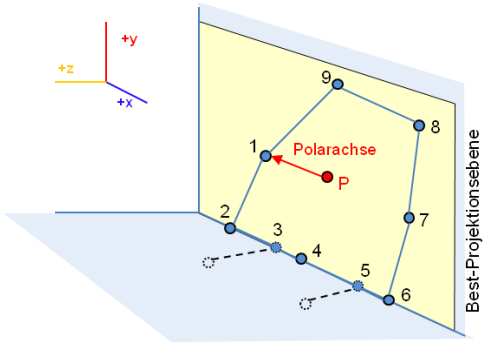


Abbildung 40: Polygonzug der sortierten Punkte in der Best-Projektionsebene

Die oben ermittelte Sortierreihenfolge wird auf die ursprünglichen Punkte übertragen. Nach dieser Reihenfolge werden die Punkte in dem dreidimensionalen kartesischen Raum trianguliert. Dabei entsteht eine nicht-optimale Triangulation, bei der die Form der Objektoberfläche nicht berücksichtigt wird (siehe Abbildung 41). Es entstehen Kanten im Dreiecksnetz (in Abbildung 41 rot gezeichnet), die nicht gemäß dem Objektäußeren verlaufen. Es werden Punkte miteinander verbunden, die auf unterschiedlichen Ebenen liegen und nicht zu einer scharfen Kante gehören. Dadurch tauchen Dreiecke im Netz auf, die die richtige Rekonstruktion der korrespondierenden Teiloberfläche verletzen.

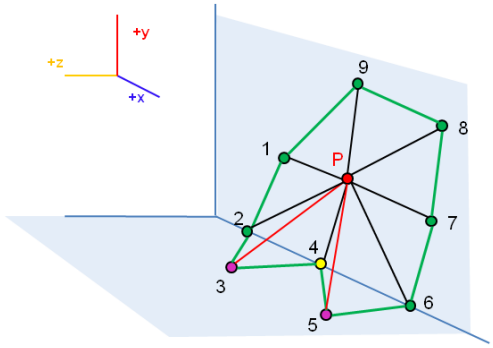


Abbildung 41: Die nicht-optimale lokale Triangulation

Eine richtige Rekonstruktion wäre z.B. ein wie in Abbildung 42 aufgestelltes Dreiecksnetz, in dem die triangulierte Teiloberfläche die Form des betrachteten Objektes um den Punkt P einhält.

Bei beiden oben erwähnten lokalen Triangulationsverfahren treten solche Probleme hauptsächlich bei Punkten auf, welche in der Nähe von scharfen Kanten liegen. Das gleiche gilt für Punkte, die sich im Bereich von Übergängen zwischen verschiedenen geometrischen 3D-Grundprimitiven befinden, wie z.B. Zylinder zu Kugel.

Die Punkte, die die Teiloberflächenrekonstruktion bei einer lokalen Triangulation verletzen (in Abbildung 41 Punkt 3 und 5), sind die bereits oben erwähnten *störenden* Punkte, die unbedingt aus der Triangulation entfernt werden müssen. Anderenfalls würden die Normalenvektoren der Dreiecksfläche, welche durch die Beteiligung der störenden Punkte gebildet werden, die Richtung des gesuchten Normalenvektors an dem betrachteten Punkt *P* manipulieren. Eine nicht-

optimale Triangulation führt dazu, dass der betrachtete Punkt bei der normalenbasierten Segmentierung (siehe Kapitel 3.3.5.1) zu einem falschen Segment zugeordnet wird.

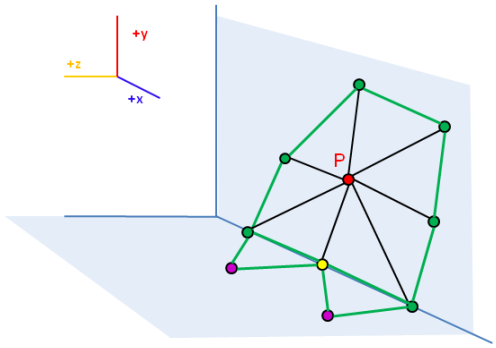


Abbildung 42: Ein richtig erstelltes Dreiecknetz, in dem die äußere Gestalt der Teiloberfläche berücksichtigt ist

Außerdem verwendet das im Rahmen dieser Arbeit vorgestellte Konzept die lokal erstellte Triangulation, um die Hauptkrümmungen (siehe Kapitel 3.3.4) an den betrachteten Punkten abzuschätzen. Durch die nicht-optimale Triangulation wäre auch die Krümmungsberechnung betroffen, was das dazu führen würde, dass die betrachteten Punkte bei der krümmungsbasierten Segmentierung (siehe Kapitel 3.3.5.2) falsch klassifiziert werden.

Daher ergibt sich eine Notwendigkeit der Eliminationsmethode, um die Nachteile der nicht-optimalen Triangulationen zu beseitigen. Deshalb wurde in dieser Arbeit das *Local Center Triangulation*-Verfahren weiterentwickelt, was im folgenden als *Lokale Triangulation mit direkten Nachbarpunkten* genannt wird. Dieses Verfahren erstellt ein Dreiecknetz um den betrachteten Punktes *P* mit seinen *direkten* Nachbarpunkten (Nachbarn *ersten Grades*), in dem es zuerst die störenden Punkte aufspürt und eliminiert (siehe Abbildung 43). Die Funktionsweise der Methode wird im Kapitel 3.3.3.2.3 näher beschrieben.

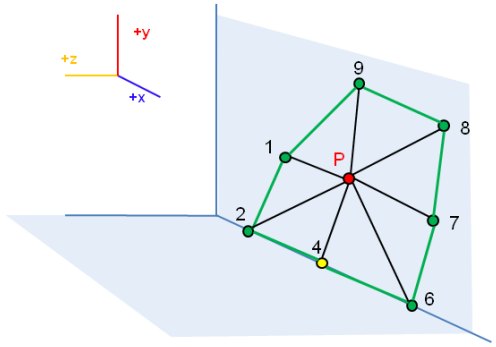


Abbildung 43: Triangulation der Punkte mit der Elimination von störenden Punkten

Die vorgestellte Methode bildet eine wertvolle Ergänzung zu heutigen CAD-Systemen für die Feature-Erkennung. Somit lassen sich neben der Berechnung eines Normalenvektors, gleichzeitig die Hauptkrümmungen an den betrachteten Punkten aus einer Punktmenge mit hoher Genauigkeit abschätzen. Die Segmentierung der Punktwolke und damit die Rekonstruktion der Oberfläche eines Modells werden dadurch effizient und ohne aufwendige analytische Berechnungen unterstützt.

3.3.3.2.2 Aufklappen der Nachbarpunkte für die Sortierung

Um die störenden Nachbarpunkte zu eliminieren, die bei einer nicht-optimalen Triangulation auftreten könnten, müssen alle Nachbarpunkte in die Best-Projektionsebene übertragen werden.

Die Übertragung wird dabei nicht nur durch eine "einfache senkrechte" Projektion der Punkte in die Ebene realisiert, sondern es wird in diesem Ansatz ein zusätzlicher Schritt ausgeführt. Dieses Verfahren der Übertragung wird als Aufklappen der Nachbarpunkte bezeichnet.

Der Zweck dieses Ansatzes ist das Auffinden der *störenden* Nachbarpunkte (siehe Appendix D) in der jeweiligen Best-Projektionsebene als *überflüssige* Nachbarpunkte (auch *indirekte* Nachbarpunkte genannt, siehe Appendix D). Anders formuliert; die störenden Punkte werden nach der Projektion als überflüssige Nachbarpunkte behandelt. Damit können alle störenden Punkte aus dem Polygonzug in der Best-Projektionsebene auf eine einfache Art und Weise entfernt werden. Die übrig bleibenden Punkte entsprechen dann den gesuchten direkten Nachbarpunkten um den betrachteten Punkt *P*.

In der folgenden Abbildung 44 ist das Problem grafisch dargestellt, wieso man das Verfahren des Aufklappens benötigt. Es sei der betrachte Punkt P und seine senkrecht projizierten und sortierten Nachbarpunkte in einer Best-Projektionsebene im dreidimensionalen Raum zu sehen, ohne dass man das Verfahren des Aufklappens angewendet hat. Der Punkt *P* befindet sich direkt vor einer denkbaren (möglichen) scharfen Kante. Ein derartiger Fall tritt auf, wenn sich vor der Projektion die Punkte z.B. auf einer Würfeloberfläche befinden und die Nachbarpunkte ungeordnet im dreidimensionalen Raum verstreut sind. Man kann hier an den Punkten 6 und 8 erkennen, dass nach der senkrechten Projektion die Information über die Struktur der Punkte (eine mögliche Kante oder Orientierung) verloren geht. Der Grund dafür ist, dass diese beiden Punkte sich auf einer anderen Seitenfläche eines Objektes (z.B. Würfel) befinden, als der betrachtete Punkt *P*. Aus diesem Grund benötigt man bei der Projektion das Verfahren des Aufklappens, um die Strukturinformationen nicht zu verlieren und die überflüssigen Punkte schneller zu erkennen.

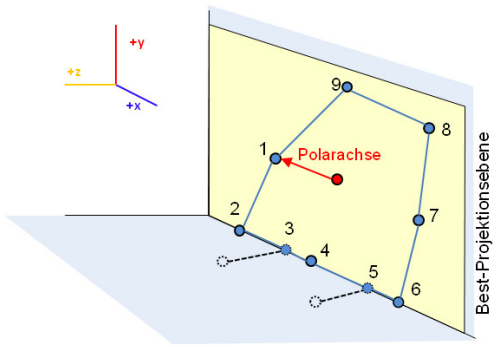


Abbildung 44: Ein betrachtete Punkt direkt vor einer scharfen Kante mit Nachbarpunkten

Was bedeutet das Aufklappen? Das Aufklappen eines Punktes im geometrischen Sinne bedeutet, eine Drehung eines Punktes um eine feste Achse mit einem bestimmten Winkel. Für das obige Beispiel würde es bedeuten, dass man einen Punkt aus der nicht-triangulierten Punktwolke um eine Achse drehen müsste, bis sich der Punkt in der Best-Projektionsebene befindet. Als Achse müsste man hier eine "mögliche scharfe Kante" verwenden (in Abbildung 45 die braune Linie), während man die Punkte in die Best-Projektionsebene projiziert. Allerdings steht die zur Drehung nötige Kante nicht zur Verfügung, da sie anfangs nicht existiert und die Aufgabe darin besteht diese erst herauszufinden. Nur die Punkte in der Punktwolke deuten auf eine solche Kante hin und man kann diese nur mit dem menschlichen Auge als solche wahrnehmen.

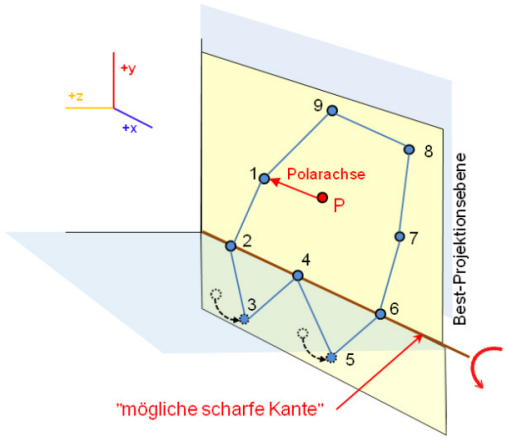


Abbildung 45: Gewöhnliches Aufklappen der Punkte um eine Achse, d.h. "mögliche scharfe Kante"

Die Identifizierung der scharfen Kanten in der Punktwolke ist die Aufgabe der im Rahmen dieser Arbeit vorgestellten normalenbasierten Segmentierung (siehe Kapitel 3.3.5.1) innerhalb des Objekterkennungsprozesses. Um die Segmentierung durchführen zu können, werden zuerst die Normalenvektoren zu allen Punkten aus der Punktwolke berechnet. Um die Normalenvektoren korrekt zu berechnen, ist die präzise lokale Triangulation um die einzeln betrachteten Oberflächenpunkte erforderlich. Das bringt allerdings einige Herausforderungen mit sich, um diese Aufgabe präzise und effizient zu lösen. Dies wird weiter unten in Kapitel 3.3.3.2.4 beschrieben.

Da man anfangs noch keine Information über eine "mögliche scharfe Kante" hat, wurde hier ein neuer Näherungsansatz eingeführt, um einen Punkt quasi aufzuklappen, der vor der Projektion noch nicht in der Best-Projektionsebene liegt.

In diesem Ansatz wird jeder Nachbarpunkt P_i zuerst in die Best-Projektionsebene projiziert und gleichzeitig sein kürzester Abstand n_i zu der Best-Projektionsebene berechnet. Durch den projizierten Punkt P_i' und dem betrachteten Punkt P wird eine Gerade gezogen (in der Abbildung 46 die Geraden PP_i' bzw. PP_j'). Danach wird die Position des projizierten Punktes P_i' auf dieser Geraden in entgegengesetzter Richtung von dem Punkt P um die berechnete Länge n_i verschoben (siehe Abbildung 46). Dieser Vorgang wird für alle Nachbarpunkte angewendet und erst danach gilt die Übertragung in die Best-Projektionsebene als abgeschlossen.

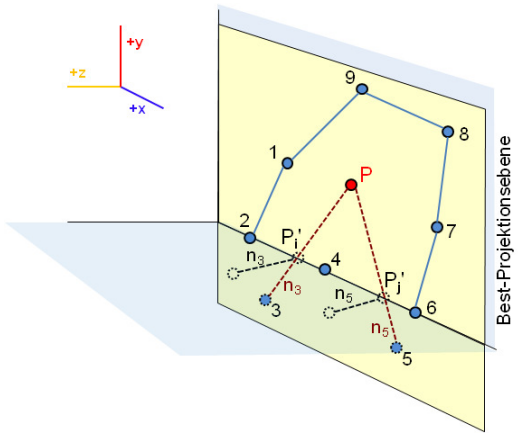


Abbildung 46: Aufklappen nach Projektionspunkt

Im Vergleich zum Aufklappen um eine vordefinierte Achse, wird natürlich bei diesem Näherungsansatz die Position der projizierten Punkte leicht verschoben. Dennoch erhält man einen aufgeklappten Polygonzug in der Best-Projektionsebene, in dem die Orientierungen der danach

entstehenden Dreiecksflächen unverändert bleiben. Die Dreiecksflächen im Polygonzug entstehen dadurch, dass man mit drei aufeinanderfolgenden Punkten dieses Polygonzugs ein Dreieck aufgespannt. Anders formuliert bleiben die Strukturinformationen erhalten und die nach dieser Art des Aufklappens entstehenden Polygonzüge bleiben orientierungstreu; vgl. den Polygonzug in Abbildung45 mit dem Polygonzug in Abbildung 47.

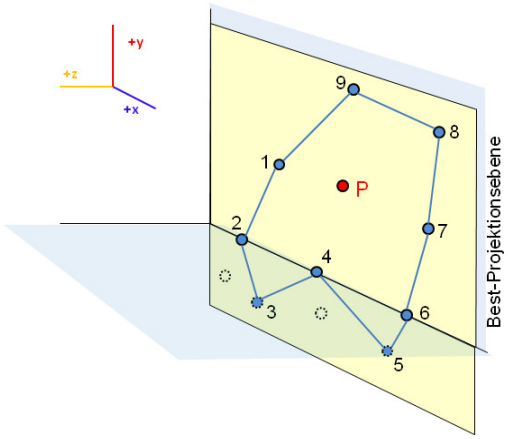


Abbildung 47: Nach dem Aufklappen ist Topologie bzgl. der Orientierungen der im zugehörigen Polygonzug vorkommenden Dreiecksfläche beibehalten

Dieser Näherungsansatz zum Aufklappen ist nötig, um auf den Nachbarpunkten eines betrachteten Punktes *P* in der Best-Projektionsebene die neu entwickelte Eliminationsmethode anzuwenden (siehe Kapitel 3.3.3.2.3). Somit werden die *störenden* Nachbarpunkte vor einer Triangulation als *überflüssige* Nachbarpunkte (beide auch *indirekte* Nachbarpunkte genannt) in der Best-Projektionsebene erkannt und deren Eliminationen ausgeführt. Wie das Auffinden der indirekten (überflüssigen) Nachbarpunkte anhand einer Orientierungsanalyse von Dreiecksflächen verwirklicht ist, wird im nächsten Kapitel ausführlich beschrieben.

3.3.3.2.3 Funktionsweise der Eliminationsmethode

Das Ergebnis der Eliminationsmethode ist ein näherungsweise konvexer Polygonzug der Nachbarpunkte eines betrachteten Punktes *P* in der Best-Projektionsebene. Eine optimale lokale Triangulation der Punkte wird von diesem Polygonzug abgeleitet. Diese optimale Triangulation (Dreiecknetz) wird sowohl zur Berechnung des Normalenvektors (siehe Kapitel 3.3.3.2.4) als auch zur Abschätzung der Krümmung mit Hilfe des Satzes von Gauß-Bonnet (siehe Kapitel 3.3.4.2) an dem betrachteten Punkt *P* eingesetzt.

Der blau gezeichnete Zug in Abbildung 48 veranschaulicht den Polygonzug *PZ* der "aufgeklappten" Nachbarpunkte von dem betrachteten Punkt *P*, welche bezüglich der Sortierung aufsteigend von 1 bis 9 durchnummeriert sind. In Abbildung 48 sind einige Dreiecksflächen hellblau hervorgehoben. Die grün-gefärbten Normalenvektoren dieser Dreiecksflächen zeigen entweder in die Best-Projektionsebene hinein oder aus der Best-Projektionsebene hinaus. $\vec{n}_{891}$ und $\vec{n}_{789}$ bezeichnen die Normalenvektoren der positiv orientierten Flächen der Dreiecke *891* und *789*. Der Normalenvektor $\vec{n}_{345}$ gehört zu der einzigen Fläche mit negativer Orientierung in diesem Polygonzug. Die Dreiecke werden nach der Sortierreihenfolge aus jeweils drei aufeinanderfolgenden Polygoneckpunkten gebildet und den entsprechenden Dreiecksflächen aufgespannt.

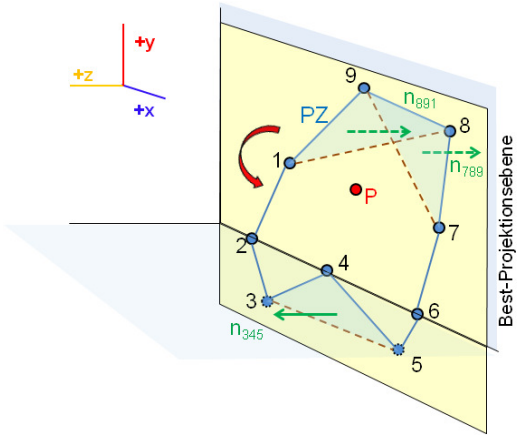


Abbildung 48: Normalenvektoren von einigen Dreiecksflächen im Polygonzug in der Best Projektionsebene

Die *störenden* Nachbarpunkte werden als *indirekte* Nachbarpunkte auf Basis der Orientierungsprüfung von Dreiecksflächen in der Best-Projektionsebene identifiziert. Dabei wird angenommen, dass ein *konvexer* Polygonzug in der Best-Projektionsebene die *direkten* Nachbarpunkte des betrachteten Punktes *P* beinhaltet. Das Ziel der Eliminationsmethode ist die Gewährleistung der Konvexität des Polygonzuges durch Entfernung der *indirekten* Nachbarn aus dem Polygonzug.

Ist eine Dreiecksfläche in einem Polygonzug *negativ orientiert* (z.B. das Dreieck *345* in Abbildung 48 nach der Sortierung), dann zeigt deren Normalenvektor aus der Ebene hinaus. Der Orientierungswechsel ändert die Richtung des Normalenvektors. Die Dreiecksflächen mit negativer Orientierung verletzen die Konvexität des Polygonzuges.

Der Polygonzug wird startend mit den ersten drei Eckpunkten auf negativ orientierte Dreiecksflächen hin untersucht. In jedem Schritt wird eine neue Dreiecksfläche mit den zwei zuletzt besuchten und in Folge dem nächsten Eckpunkt aufgespannt und ihre Orientierung geprüft. Dies wird solange wiederholt, bis das Dreieck mit dem Ausgangspunkt und dem Nachfolger des Ausgangspunktes erreicht ist (Anschlussstelle des Polygonzuges).

Bei einem nicht-konvexen Polygonzug werden diejenigen Punkte in dem Dreieck untersucht, die entsprechend ihrer Reihenfolge eine negativ orientierte Fläche aufspannen und damit die Konvexität des Polygonzuges verletzen. Ist die Fläche eines Dreieckes negativ orientiert, dann ist der mittlere Punkt (beispielsweise Punkt 4 in Abbildung 48) des betroffenen Dreieckes mit großer Wahrscheinlichkeit der gesuchte *direkte* Nachbar von Punkt *P* und die anderen zwei Punkte (Punkt 3 und 5 in Abbildung 48) Kandidaten für mögliche *indirekte* Nachbarpunkte. Einer der Kandidaten wird als *indirekter* Nachbar von Punkt *P* aus dem Polygonzug entfernt. Vor der Entfernung der möglichen indirekten Nachbarpunkte wird geprüft, ob durch die Entfernung des betreffenden Punktes die Konvexität des Polygonzuges in der Umgebung dieses indirekten Nachbarpunktes wiederhergestellt wird. Es entstehen dann folgende Fälle:

1) Wird durch die jeweilige Entfernung der beiden Kandidaten die Konvexität des Polygonzuges in der Umgebung der entfernten Kandidaten nicht erfüllt – d.h. es entsteht keine neue positiv orientierte Dreiecksfläche – dann wird der Kandidat mit dem größeren Abstand zu Punkt *P* als indirekter Nachbar aus dem Polygonzug eliminiert.

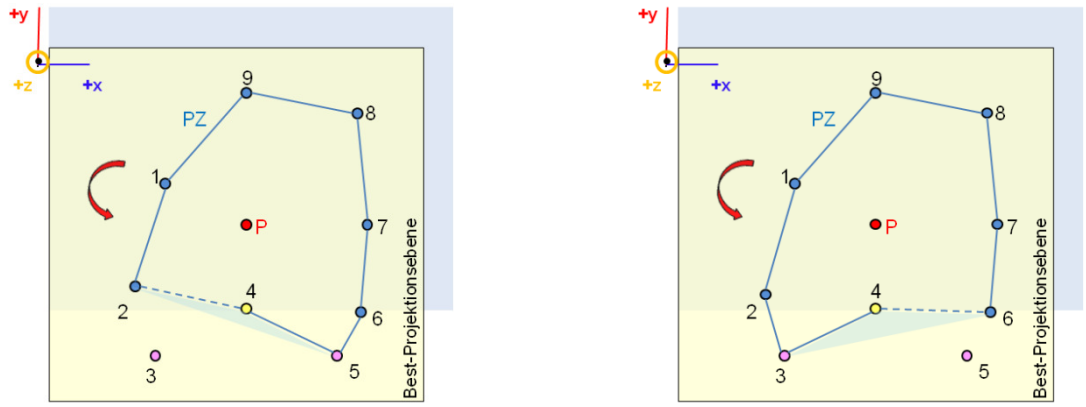


Abbildung 49: Überprüfen, ob durch Entfernung eines von beiden Kandidaten (die mit 3 und 5 kennzeichnete, rosa gefärbte Punkte) eine negativ oder positiv orientierte Dreiecksfläche entsteht

2) Wird durch Entfernen eines von beiden Kandidaten die Konvexität des Polygonzuges in der Umgebung des entfernten Kandidaten erfüllt – d.h. es entsteht eine neue positiv orientierte Dreiecksfläche – dann wird der entsprechende Kandidat als indirekter Nachbar aus dem Polygonzug eliminiert.

3) Entstehen durch Entfernung von beiden Kandidaten jeweils positiv orientierte Dreiecksflächen, dann wird der Kandidat mit dem größeren Abstand zu Punkt *P* als *indirekter* Nachbarn aus dem Polygonzug eliminiert.

Die Punkte des Polygonzuges werden solange durchlaufen, bis die Konvexität des Polygonzuges in der Best-Projektionsebene näherungsweise erfüllt ist.

Die Abbildung 50 zeigt mit dem in diesem Kapitel durchgängig verwendeten Beispiel alle möglichen Dreiecksflächen dar, die entsprechend ihrer Orientierung geprüft werden, Beispielsweise ist die Fläche des Dreieckes *345* negativ orientiert, während die ersten beiden Flächen der Dreiecke *123* und *234* bzgl. der Drehrichtung gegen den Uhrzeigersinn (roter Pfeil) in der Best-Projektionsebene positiv orientiert sind. Das kann man mit Hilfe der Rechte-Hand-Regel oder Drei-Finger-Regel (siehe Kapitel 2 im Abschnitte c und d) veranschaulichen. Damit ist der nach der Sortierung als Punkt 4 gekennzeichnete Punkt mit großer Wahrscheinlichkeit ein direkter Nachbarpunkt. In diesem Fall sind die Punkte 3 und 5 Kandidaten als indirekte Nachbarpunkte bezüglich Punkt *P* und werden daher der Eliminationsprüfung herangezogen. Der Kandidat, der eliminiert wird, ist dann der jeweilige indirekte Nachbar.

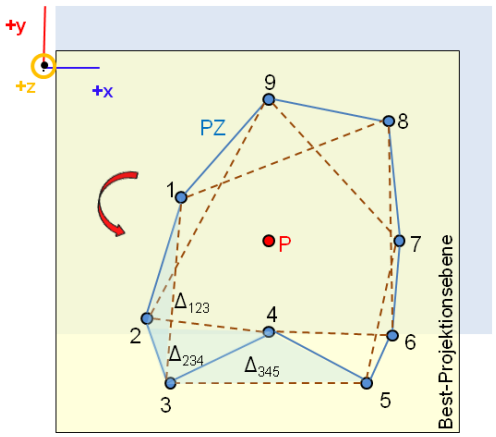


Abbildung 50: Polygonzug in der Best-Projektionsebene mit allen Dreiecksflächen zwischen den aufeinanderfolgenden Eckpunkten

Ein Sonderfall tritt auf, wenn neben Nachbarn ersten Grades (direkte Nachbarn) und zweiten Grades auch Nachbarn dritten oder höheren Grades in der Nachbarschaft von Punkt *P* auftreten. Der Grund dafür ist, dass keine feste mittlere Entfernung zwischen den Punkten in der Punktwolke existiert und daher der *kD*-Baum, der intern eine Suchkugel mit vordefiniertem Radius (siehe Kapitel 3.3.2 Abschnitt b)) verwendet, auch Nachbarn dritten und höheren Grades liefern kann. Deshalb werden die Punkte des Polygonzuges zuerst mit der neu entwickelten Elimina-tionsmethode in der Best-Projektionsebene durchlaufen und dabei nach zwei hinter-einander liegenden negativ orientierten Dreiecksflächen gesucht (z.B. in Abbildung 51 die zwei Dreiecke in der Reihenfolge rot-orange-gelb-orange), um die Nachbarn dritten oder höheren Grades zu eliminieren. Als Entfernungskriterium wird der Abstand zum betrachtenden Punkt *P* genommen, d.h. in dem Fall, in dem zwei hintereinander liegende negativ-orientierte Dreiecks-flächen vorkommen, wird der Nachbarpunkt mit dem größten Abstand aus dem Polygonzug entfernt (roter Punkt im Polygonzug), weil dieser Punkt mit großer Wahrscheinlichkeit ein Nachbar dritten oder höheren Grades ist.

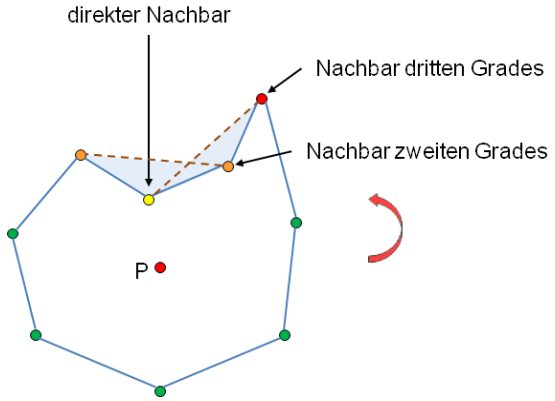


Abbildung 51: Sonderfall, bei dem in der Nachbarschaft des betrachtenden Punkt P Punkte dritten oder höheren Grades auftreten

Die Punkte im Polygonzug werden bei diesem Schritt solange durchlaufen, bis keine negativ orientierten Dreiecksflächen im Polygonzug hintereinander in Folge auftreten.

Bisher wurde die optimierte lokale Triangulation, welche auf Sortierung der Nachbarpunkte eines betrachteten Punktes *P* in einer Ebene basiert, durch das Verfahren *Lokale Triangulation mit direkten Nachbarpunkten* mit Hilfe der oben vorgestellten Eliminationsmethode erstellt. Das Ergebnis der optimalen Triangulation ist ein Dreiecknetz in dem betrachteten Punkt *P* mit seinen *direkten* Nachbarpunkten. Der große Vorteil ist somit, dass durch Anwendung dieses Netzes die Normalenvektorrichtung an dem betrachteten Punkt *P* genauer abgeschätzt (siehe Kapitel 3.3.3) und dasselbe Dreiecknetz zur effizienten Abschätzung der Krümmung an dem betrachteten Punkt *P* eingesetzt wird, ohne dabei aufwendige analytische Berechnungen (siehe [CFGG07]) durchzuführen. Die Abschätzung der Krümmungen mit Hilfe des Satzes von Gauß-Bonnet wird in nächstem Kapitel behandelt.

Die Richtung eines gesuchten Normalenvektors in einem Oberflächenpunkt, welcher nach seiner Lage in der Punktwolke betrachtet wird, hängt wesentlich von der Nachbarschaft des betrach-teten Punktes ab. Normalenvektoren werden im Bereich der Computergrafik unter anderen zur Beleuchtungsberechnungen an der Oberfläche der Modelle genutzt. In unserem Fall werden die Richtungen der Normalenvektoren an den einzelnen betrachteten Punkten verwendet, um die einzelnen Punkte zu segmentieren und anschließend zu klassifizieren. Bei der normalenbasierten Segmentierung spielen nicht die Beträge der Normalenvektoren an den Punkten aus der Punkt-wolke, sondern deren Richtungen die Hauptrolle (siehe Kapitel 3.3.5.1). Die Richtung eines Normalenvektors hängt von der lokalen Triangulation aus den Nachbarpunkten von Punkt *P* ab.

Eine lokale Triangulation um einen betrachteten Punkt *P* mit seiner Nachbarschaft beschreibt die Teiloberfläche um den Punkt *P*.

3.3.3.2.4 Berechnung des Normalenvektors an einem betrachteten Punkt

Die Berechnung der Normalenvektoren an einzelnen Punkten ist ein Teilprozess der Objekterkennung und gehört zum ersten Schritt des Konzepts.

Wie schon erwähnt ist, da keine triangulierte Punktwolke zur Verfügung steht, wird die Oberfläche *teilweise* trianguliert, um die geometrischen Informationen an den Oberflächenpunkten zu gewinnen. Mit "teilweise" wird die *lokale* Triangulation gemeint, d.h. die Rekonstruktion der Teiloberfläche um einen betrachteten Punkt mit seiner Nachbarschaft, welche in dieser Fläche des entsprechenden Teilobjekts liegt.

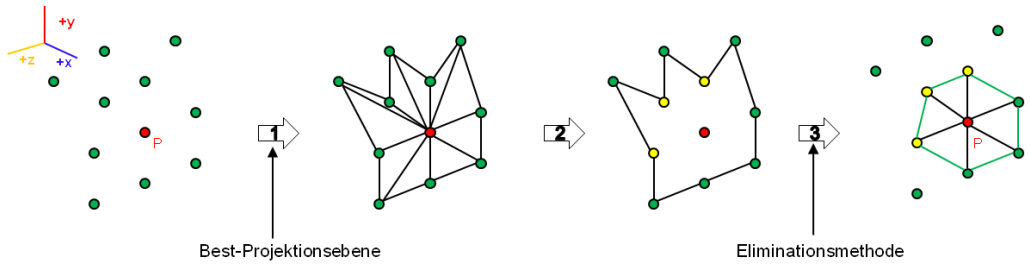


Abbildung 52: Ablauf bis zu optimaler Triangulation mit direkten Nachbarpunkten von Punkt *P*

Um mit der Berechnung eines Normalenvektors zu einem betrachteten Punkt beginnen zu können, müssen die in der Abbildung 52 gezeigten Schritte abgeschlossen sein, d.h. das Verfahren *Lokale Triangulation mit direkten Nachbarpunkten* (siehe Kapitel 3.3.3.1) mit der internen Eliminationsmethode (siehe Kapitel 3.3.3.2) vollendet sein. Eine präzise lokale Triangulation leitet man aus dem konvexen Polygonzug in der Best-Projektionsebene ab.

Das Ergebnis einer optimierten lokalen Triangulation ist ein Dreiecknetz um den betrachteten Punkt *P* mit seiner ermittelten direkten Nachbarpunkte. Der gesuchte Normalenvektor an dem betrachteten Punkt *P* wird durch die Mittelung aller Normalenvektoren der in dem Dreiecknetz vorkommenden Dreiecksflächen berechnet. Alle Dreiecke beinhalten den betrachteten Punkt *P*. Der Punkt *P* liegt ungefähr in der Mitte des Dreiecksnetzes (siehe Abbildung 53).

Ein Dreieck im Netz wird aus zwei aufeinanderfolgenden Punkten und dem betrachteten Punkt *P* gebildet. Die Reihenfolge der Nachbarpunkte ergibt sich nach dem im Kapitel 3.3.3.1 vorgestellten Sortierungsansatz.

In Abbildung 53 ist ein Ausschnitt aus einer lokal triangulierten Oberfläche dargestellt. Die Eckpunkte des Dreiecknetzes werden mit $P_i \in \mathbb{R}^3$ (i = 1,…,6) und der betrachtete Punkt mit *P* bezeichnet, wobei die P_i's Nachbarpunkte vom Punkt *P* sind. Eine Verbindungsstrecke entspricht der Kante e_i zwischen dem Punkt *P* und einem Nachbarpunkt P_i im Netz [MSR 07] [Werl11].

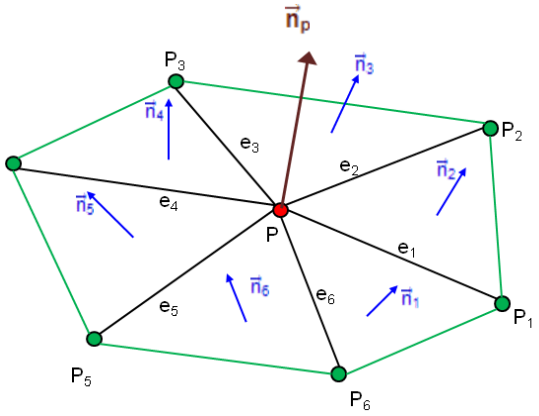


Abbildung 53: Berechnung des Normalenvektors an dem betrachteten Punkt *P*

Der Normalenvektor einer umliegenden Fläche des entsprechenden Dreiecks $\Delta_i(P_i, P, P_{(i+1) \bmod d})$ wird durch das Kreuzprodukt der Dreieckkanten $e_i = \overline{PP_i}$ und $e_{i+1} = \overline{PP}_{(i+1)\bmod d}$ bestimmt.

$$\vec{n}_i = \frac{(P_i - P) \times (P_{((i+1) \bmod d)} - P)}{\|(P_i - P) \times (P_{((i+1) \bmod d)} - P)\|} \tag{3.4}$$

Der Normalenvektor $\vec{n}_p$ an dem betrachtenden Punkt *P* kann damit als Mittelwert der Normalenvektoren der umliegenden Dreiecksfläche berechnet werden:

$$\vec{n}_p = \frac{\sum_{i=1}^{N_p} \vec{n}_i}{N_p} \tag{3.5}$$

N_p : Anzahl der Nachbarpunkte des betrachteten Punkt *P*.

$\vec{n}_i$: Normalenvektor der *i*-ten Dreiecksfläche.

3.3.4 Schritt 4: Abschätzung der Krümmungen

In der normalenbasierten Segmentierung (siehe Kapitel 3.3.5.1) werden anhand der Normalenvektoren zu den Punkten (siehe Kapitel 3.3.3) die scharfen Kanten und die Übergänge zwischen geometrisch unterschiedlichen Teilobjekten detektiert und damit die relevanten Teilobjekte dementsprechend eingrenzt. Um die eingegrenzten Punkten zu einzelnen geometrischen 3D-Grundprimitiven anzupassen, sind weitere Charakteristika nötig. Zu diesen Charakteristika zählen die Abschätzungswerte der Krümmungen an den einzelnen Punkten in der Punktwolke.

Die *Krümmung* (siehe Appendix C) [Baer10][Kühn03] ist der zentrale Begriff der Flächentheorie und der Differentialgeometrie. Die charakteristische Flächeneigenschaft *Krümmung* wird als geometrisches Merkmal (Feature) verwendet, um die gescannten Objekte hinsichtlich der Oberflächentypen zu klassifizieren. Die Punktdaten, welche z.B. mit Hilfe eines Scanners oder eines (Stereo-)Kameras erfasst sind und als Punktwolke vorliegen, enthalten i. Allg. keine Informationen über die Oberflächeneigenschaften. Die charakteristische Eigenschaft Krümmung gibt es eigentlich nur für die mathematischen Kurven und Flächen, daher müssen die zur Klassifizierung benötigten Informationen mit Hilfe der Objektoberfläche, welche lokal durch die erfassten Punkten näherungsweise dargestellt ist, abgeschätzt werden. Lokal heißt in diesem Zusammenhang, dass die entsprechende Fläche um jeweils einen betrachteten Punkt mit seiner Umgebung abgebildet wird.

Um die Krümmung an einem betrachteten Punkt *P* abzuschätzen, der sich auf einer Fläche befindet, gibt es unterschiedlichen Methoden. Diese Methoden kann man generell in zwei Gruppen einteilen; analytische bzw. numerische [Boeh05][Boeh09][Vanc03]. In beiden Gruppen

werden die Umgebung vom Punkt *P* berücksichtigt, d.h. die Krümmung an dem betrachteten Punkt *P* wird hinsichtlich seiner Nachbarpunkte abgeschätzt. Das Ziel ist dabei die näherungsweise Rekonstruktion der Fläche in der Umgebung von dem betrachteten Punkt *P*.

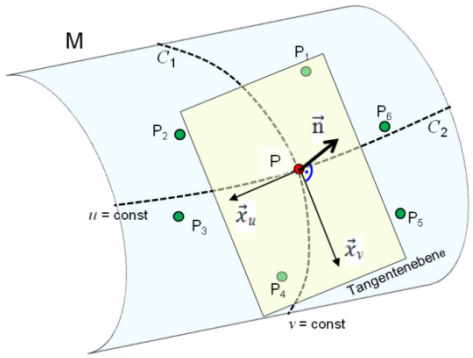


Abbildung 54: Fläche mit eingebetteten Flächenkurven

Um eine Fläche im dreidimensionalen Raum darzustellen, verwendet man oft Kurven (die gestrichelten Kurven in Abbildung 54), die in dieser Fläche eingebettet sind. Diese Kurven nennt man Flächenkurven (siehe Appendix C). Durch einen Punkt, der auf der Fläche liegt (z.B. in Abbildung 54 der Punkt *P* in der Fläche *M*), verlaufen beliebig viele Flächenkurven. Die Krümmung einer eingebetteten Kurve (z.B. c_1 bzw. c_2) an dem betrachteten Punkt *P* der Fläche *M* ist der Radius des Kreises, der den Punkt *P* berührt und sich der Form anpasst, welche die entsprechende Kurve in der Umgebung des Punktes *P* annimmt. Beschreibt man die Fläche *M* an dem Punkt *P* durch zwei parametrische Kurven $C_1(u)$ und $C_2(v)$ (siehe Appendix C), dann gibt es zwei Radien r_1 und r_2 der beiden Kreise, die den Punkt *P* berühren. Da es viele solche eingebetteten Kurven gibt, ergeben sich auch viele Krümmungen in dem Punkt *P*. Es gibt aber eine Krümmung mit dem kleinsten Wert (Kreis mit dem minimalen Radius) und eine mit dem größten Wert (Kreis mit dem maximalen Radius), warum man auch deswegen von einer minimalen und einer maximalen Krümmung sprechen kann. Solche Krümmungen werden als Hauptkrümmungen (engl. principal curvature) genannt und mit κ_1 bzw. κ_2 [Kühn03] oder in manchen Literaturen voneinander leicht zu unterscheiden auch κ_{min} bzw. κ_{max} bezeichnet.

Neben Hauptkrümmungen gibt es auch Gauß'sche Krümmung und mittlere Krümmung in dem Teilgebiet der analytischen Mathematik, welche formal miteinander in einer Beziehung stehen. Das Produkt der Hauptkrümmungen nennt man Gauß'sche Krümmung K_G und die mittlere Krümmung K_M ergibt sich aus dem arithmetischen Mittel der Hauptkrümmungen [AuSp93], also

$$\text{Gauß'sche Krümmung:} \quad K_G = \kappa_{max} \cdot \kappa_{min} \tag{3.6}$$

$$\text{Mittlere Krümmung:} \quad K_M = \frac{1}{2}(\kappa_{max} + \kappa_{min}) \tag{3.7}$$

3.3.4.1 Analytische Krümmungsabschätzung

Ziel der analytischen Krümmungsabschätzung ist es, für eine gegebene Menge von Punkten durch lineare Approximation der Nachbarschaft die lokale parametrische Flächenfunktion $x(u,v)$ (siehe Appendix C) zu ermitteln. Die Berechnung der partiellen Ableitungen aus den Funktionenparametern der zweifach differenzierbaren Flächenfunktion gibt dann die Krümmung für den Punkt *P* an.

Rechenergebnisse zeigen, dass numerische Krümmungsabschätzungen im Vergleich zu den analytischen Krümmungsabschätzungen effizient arbeiten und schnellere Ergebnisse auf großen Punktdaten liefern [Gomo09][Boeh05]. Die operativ aufwändigen Schritte zur Krümmungs-

berechnung, z.B. durch Anlegen von Quadriken (Flächen zweiter Ordnung), sind in [CFGG07] beschrieben. Man kann auch in [Boeh05] die Herleitung der Krümmungen mit analytischen Methoden und die geometrische Bedeutung der zweiten Abteilung der parametrischen Flächenfunktion für einen Punkt finden.

3.3.4.2 Numerische Krümmungsabschätzung

In [MSR 07] wurden fünf Methoden experimentell miteinander verglichen. Das Ergebnis zeigt, dass der Satz von Gauß-Bonnet eine effiziente und robuste Methode für die Abschätzung der Gauß'schen Krümmung ist. Der Satz von Gauß-Bonnet ist eine numerische Methode zur Krümmungsabschätzung.

Der Satz von Gauß-Bonnet drückt das Zusammenspiel zwischen der topologischen Eigenschaft einer geschlossen Fläche und der Krümmung dieser Fläche aus. Der Satz von Gauß-Bonnet hat verschiedene Fassungen (siehe Appendix F). Die spezielle Form des Satzes von Gauß-Bonnet basiert auf einem Polyeder (siehe Appendix D); diese spezielle Form wird in den folgenden Abschnitten vorgeführt. Dieser Polyeder entspricht genau der lokalen Triangulation um einen betrachteten Punkt P mit seinen direkten Nachbarn. Die vom Gauß-Bonnet-Satz geforderte Triangulation wird im Rahmen dieser Diplomarbeit mit dem Verfahren *Lokale Triangulation mit direkten Nachbarpunkten* (siehe Kapitel 3.3.3) aufgestellt.

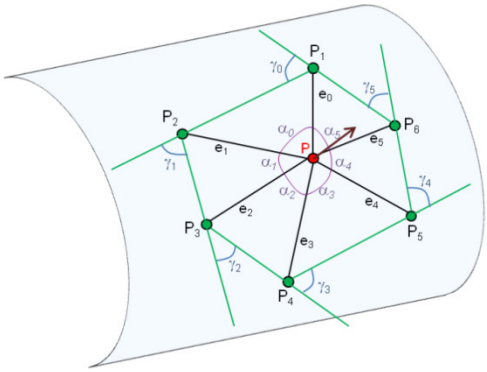


Abbildung 55: Polyeder: optimierte lokale Triangulation mit direkten Nachbarn vom Punkt P

An einem Beispiel in Abbildung 55 ist die optimale lokale Triangulation auf einer Fläche um einen betrachteten Punkt P mit seinen direkten Nachbarpunkten dargestellt. Der Punkt P_i (i = 1, …, n) ist dabei ein direkter Nachbar von dem Punkt P. Dabei gibt n die Anzahl der Nachbarpunkte an und ist in diesem Bespiel n = 6. Die inneren Kanten des Dreiecknetzes wurden mit $e_i = \overline{PP_i}$ bezeichnet. Sei γ_i dabei der Außenwinkel zwischen zwei aufeinanderfolgenden äußeren Kanten vom Polyeder, also der Winkel zwischen der ab dem Punkt P_i verlängerten Kante $\overline{P_iP_{(i+1) \bmod n}}$ und der Kante $\overline{P_{(i+1)}P_{(i+2) \bmod n}}$, wobei sich die Kanten an dem Punkt P_{i+1} schneiden (siehe Abbildung 55), dann ist α_i der Winkel zwischen zwei aufeinanderfolgenden inneren Kanten e_i [MSR 07]. Dann gilt;

$$\iint_A K_G dA = 2\pi - \sum_{i=0}^{n-1} \gamma_i \Rightarrow \iint_A K_G dA = 2\pi - \sum_{i=0}^{n-1} \alpha_i \tag{3.8}$$

Das heißt, die Summe der Außenwinkel γ_i ist gleich der Summe der Innenwinkel α_i in dem Polyeder.

Mit der Annahme, dass Gauß'sche Krümmung in der direkten Nachbarschaft des betrachteten Punktes P konstant ist, lautet die obige Formel folgendermaßen [Werl11]:

$$K_G = \frac{2\pi - \sum_{i=0}^{n-1} \alpha_i}{\frac{1}{3}A_{ges}} \tag{3.9}$$

A_{ges} in der Gleichung gibt die Gesamtsumme der Inhalte von allen Dreiecksflächen in dem Polyeder [MeWa00][KKK 02].

Dann gilt die folgende Gleichung für die mittlere Krümmung [MSR 07]:

$$K_M = \frac{\frac{1}{4}\sum_{i=0}^{n-1} \|e_i\| \beta_i}{\frac{1}{3}A_{ges}} \tag{3.10}$$

β_i bezeichnet dabei die Normalenvektorenabweichung und ist gleich dem Winkel zwischen den Normalenvektoren von zwei aufeinanderfolgenden Dreiecksflächen im Polyeder, also

$$\beta_i = \measuredangle(\vec{n}_i, \vec{n}_{(i+1) \bmod n}) \tag{3.11}$$

$\|e_i\|$ ist der Betrag der inneren Kante e_i vom Polyeder.

Somit kann man die minimale Hauptkrümmung κ_{min} und maximale Hauptkrümmung κ_{max} aus der Gauß'schen und der mittleren Krümmung wie folgt ableiten [Agos05][Boeh05]:

$$\kappa_{min} = K_M - \sqrt{K_M^2 - K_G} \tag{3.12}$$

$$\kappa_{max} = K_M + \sqrt{K_M^2 - K_G} \tag{3.13}$$

3.3.5 Schritt 5: Segmentierung

Für viele CAD-Anwendungen ist es notwendig, durch Scanner abgetastete reale Objekte, in mehrere sinnvolle Bereiche zu unterteilen, so dass eine leichtere Weiterverarbeitung ermöglicht wird. Diese Bereiche beschreiben dabei geometrischen 3D-Grundprimitive (Teilobjekte) wie z.B. Ebenen, Kugel oder Zylinder. Um nach der Abtastung einer Objektoberfläche, die als Punktwolke vorliegt, die darin enthaltenen geometrischen 3D-Grundprimitive identifizieren zu können, werden die Punkte mit nahezu ähnlichen Eigenschaften zu Segmenten zusammengefasst. Um die Punkte segmentieren zu können, müssen zunächst Informationen (wie z. B. Normalenvektoren, Krümmungen, etc.) hinsichtlich der Oberfläche gesammelt werden (siehe Kapitel 3.3.3 und Kapitel 3.3.4). Die Oberflächeninformationen deuten auf unterschiedliche Charakteristiken von Teilobjekten und lassen damit Teilobjekte identifizieren. Beispielsweise sind Normalenvektoren von Punkten einer Ebene näherungsweise gleich orientiert oder Krümmungen an Punkten in einer Ebene sind gleich 0. Unter Segmentierung versteht man die Aufteilung einer Punktwolke in mehrere relevanten Teilpunktwolken, die geometrischen 3D-Grundprimitiven entsprechen und nach einem bestimmten *Homogenitätskriterium* oder *Diskontinuitätskriterium* benachbarte Punkte zusammenfassen.

Diskontinuitätskriterium bedeutet dabei, dass man überprüft, ob zwei benachbarte Punkte eine große Differenz bei den geometrischen Merkmalswerten (wie z.B. Winkeldifferenz zwischen Normalenvektoren) aufweisen, wie es z.B. bei einer Kante [Uvir09]. Verfahren, die Homogenitätskriterien verwenden, untersuchen benachbarte Punkte in der Punktwolke auf Ähnlichkeit bei den geometrischen Merkmalswerten (z.B. Krümmungen). Falls sie in einem angegebenen Schwellenwert sind, werden die Punkte dem gleichen Segment zugeordnet [Uvir09].

Die Segmentierung einer Punktwolke ist im Allgemein ein Vorbereitungsschritt zur Extraktion von geometrischen 3D-Grundprimitiven aus der Punktwolke und der anschließenden mathematischen Beschreibung. In der Regel sind Segmentierungsverfahren nicht ganz zuverlässig, funk-

tionieren nicht automatisch und eine saubere Segmentierung ist aufwendig. Meistens sind Korrekturen durch Konstrukteure erforderlich. Mit den zwei in diesem Kapitel vorgestellten Segmentierungsverfahren werden (siehe Kapitel 3.3.5.1 und 3.3.5.2) hauptsächlich folgende Aufgaben gelöst:

- Ermitteln von Objektgrenzen (scharfe Kanten und Bereiche, in denen starke Krümmungsänderungen vorkommen, die Teilobjekte abgrenzen) und
- Auffinden von geometrischen 3D-Grundprimitiven (Teilobjekte mit nahezu ähnlichen Eigenschaften)

Herauszufinden an welchen Stellen die Objekte angehängt sind oder wo die Teilobjekte ihre Grenze haben sollen Segmentierungen durchgeführt werden. Diese Art der Segmentierung wird hier mit der *normalenbasierten Segmentierung* bewerkstelligt.

Um Teilobjekte aus der unstrukturiert vorliegenden Punktwolke aufzufinden, die den bekannten geometrischen 3D-Grundprimitiven entsprechen (Klassifikation), wird eine *krümmungsbasierte Segmentierung* durchgeführt, welche die Punkte unter Nachbarschaftsbeziehungen nach nahezu ähnlichen Eigenschaften überprüft und anschließend dem Segment (Cluster) zuordnet, wenn die vordefinierten Homogenitätskriterien erfüllt sind.

Das Ergebnis der Segmentierung sind mehrere Segmente. *Segmente* (auch Cluster oder Teilpunktwolke genannt) sind endliche Teilpunktmengen, deren Begrenzungen durch Überprüfen großer Normalenvektorabweichungen zwischen angrenzten Punktmengen definiert werden. Die hier behandelten Segmentierungen sind ein wichtiger Bestanteil der Objekterkennung aus der ungeordneten Punktwolke.

In der Bildverarbeitung gibt es zahlreiche automatische und teilautomatische Segmentierungsverfahren, welche meistens auf Basis von Pixeln in 2D arbeiten. Dem gegenüber gibt es wenige Verfahren zur Segmentierung, die mit Punkten in 3D arbeiten. Sie werden im Allgemein in folgende zwei Gruppen unterteilt [Vanc03][Wilk02]:

- kantenorientierte Verfahren
- flächenorientierte Verfahren

Die kantenorientierten Verfahren zur Segmentierung von Punktwolken verwenden Diskontinuitätskriterien. Bei diesen Verfahren werden die Kanten der Teilobjekte extrahiert. Dabei werden zwei benachbarte Punkte auf große Differenzen bezüglich festgelegter Kriterien (z.B. große Richtungsänderung zwischen Normalenvektoren) überprüft.

Die Hintergrundidee ist dabei, dass die in der Punktwolke erhaltenden Teilobjekte durch Kanten abgegrenzt sind. Die benachbarten Punkte, die die Diskontinuitätskriterien erfüllen, bilden zusammen eine Kante, daher treten zwischen solchen Punkten starke Änderungen auf. Die extrahierten Kanten als Segmente sind Umrandungen der in der Punktwolke erhaltenen Teilobjekte und trennen diese von ihren unmittelbaren Umgebungen.

Die implementierte normalenbasierte Segmentierung geht von einem geeigneten Startpunkt aus und versucht daraus Objektgrenzen mittels starken Abweichungen der Normalenvektoren zwischen benachbarten Punkten zu erzeugen. Das Zugehörigkeitskriterium eines Punktes zu einer Kante oder einem Übergang ist dabei eine starke Richtungsänderung der Normalenvektoren der benachbarten Punkte entlang der Oberfläche.

Flächenorientierte Verfahren setzen bei der Segmentierung Homogenitätskriterien ein und beginnend mit Startpunkten (auch Staatpunkte genannt) untersuchen dabei benachbarte Punkte

auf Ähnlichkeit. Erfüllt ein untersuchter benachbarter Punkt mit seiner Eigenschaft die vorgeschriebenen Homogenitätskriterien, d.h. der jeweilig Punkt verhält sich gemäß einem angegebenen Schwellenwert gleich den anderen Punkten im aktuellen Segment, dann wird der Punkt in das aktuelle Segment aufgenommen. Mit der Zuordnung der Nachbarpunkte, welche die Homogenitätskriterien erfüllen, werden die Segmente ausgedehnt und dies wird solange fortgesetzt, bis sämtliche Punkte der Punktmenge einem Segment zugeordnet sind.

3.3.5.1 Schritt 5.1: Normalenbasierte Segmentierung

Ausgangspunkt für die normalenbasierten Segmentierung ist das oben genannte kantenorientierte Verfahren. Die Vorgehensweise und die Randbedingungen der Segmentierung wird im folgenden kurz erläutert.

Die Grundidee der normalenbasierten Segmentierung ist das Orten von Diskontinuitäten (= Unstetigkeiten) in der vorliegenden Punktwolke, welche sich durch starke Winkelabweichungen der Normalenvektoren zu den benachbarten Punkten offenbaren. In der normalenbasierten Segmentierung spielen nicht die Beträge der Vektoren an den Oberflächenpunkten eine Rolle, sondern die Richtungen, mit den man die Abweichungen zwischen den Normalenvektoren untersucht.

Die Grundvoraussetzung für diese Segmentierung ist allerdings eine genauere Abschätzung der Normalenvektoren an jedem Punkt in der Punktwolke. Dabei wird angenommen, dass die Normalenvektoren an den einzelnen Punkten richtig abgeschätzt sind (siehe Kapitel 3.3.3).

Ziel dieser Segmentierung ist die Trennung der Bereiche von ihren unmittelbaren Umgebungen in der Punktwolke. Die normalenbasierte Segmentierung ermöglicht die Detektion der scharfen Kanten von Teilobjekten (geometrische 3D-Grundprimitive) und der Bereiche (Übergänge zwischen den Teilobjekten), in denen starke Krümmungen vorkommen.

Die Arbeitsweise der normalenbasierten Segmentierung beruht darauf, dass ein benachbarter Punkt aufgrund seiner Lage und der Abweichung seiner Normalenvektorrichtung einem Segment zugeordnet wird. Die Entscheidung als Diskontinuitätskriterium wird auf Basis eines vordefinierten Schwellenwertes getroffen; die Winkelabweichung zwischen zwei Normalenvektoren der benachbarten Punkte muss unter diesem Schwellenwert liegen.

Das Resultat der Segmentierung ist eine Menge von Punkten, welche die Teilobjekte eingrenzen. Die von der normalenbasierten Segmentierung aufgespürten Kanten und Übergänge bilden die Grenzen der geometrischen 3D-Grundprimitiven (Teilobjekte). Im nächsten Kapitel wird mit der krümmungsbasierten Segmentierung versucht, die durch diese Grenzen eingeschlossenen Punkte weiter zu differenzieren und entsprechend zu klassifizieren.

3.3.5.2 Schritt 5.2: Krümmungsbasierte Segmentierung

Auf die normalenbasierte Segmentierung folgt die flächenbasierte Segmentierung innerhalb des Objekterkennungsprozesses, wobei auf die Homogenität zwischen den benachbarten Punkten geachtet wird.

Ausgangspunkt für die krümmungsbasierte Segmentierung ist das flächenorientierte Verfahren. Nachdem die lokale Krümmungsberechnung (siehe Kapitel 3.3.4) an jedem Punkt der abgetasteten Objektoberfläche durchgeführt ist, wird die Punktwolke in Teilsegmente ähnlicher Krümmung aufgeteilt. Die *Krümmung* (siehe Appendix C) einer Objektoberfläche in einem Punkt ist ein Maß zur Entscheidung der Zugehörigkeit des Punktes zu einem Typ eines geometrischen 3D-Grundprimitives. Die krümmungsbasierte Segmentierung nutzt diesen Maß, um Teilobjekte in der Punktwolke zu separieren.

Bei der normalenbasierten Segmentierung kann es vorkommen, dass manche Bereichen in der Punktwolke nicht als Übergang zwischen unterschiedlichen Grundprimitiven wahrgenommen werden. Dieses Problem wird in [Vanc03] an einem Beispiel zwischen einem Zylinder und einer Kugel gut veranschaulicht. Die krümmungsbasierte Segmentierung ergänzt somit die normalenbasierte Segmentierung und hilft dabei, die aufgeteilten Teilpunktwolken zu klassifizieren.

Die Segmentierung der Punktwolke hängt stark von der Untersuchung der Nachbarschaft innerhalb der Punktdaten ab. Durch die Ermittlung der Nachbarschaft eines untersuchten Punktes aus dem aktuellen Segment und die Überprüfung von Oberflächenmerkmalen, wie z.B. die maximale und minimale Hauptkrümmung der benachbarten Punkte, können einzelne Segmente mit speziellen Verfahren zu größeren Segmenten ausgedehnt werden.

Während bei der normalenbasierten Segmentierung die Umrandungen der Teilobjekte detektiert werden, werden durch die Verwendung von *Bereichswachstumsverfahren* (engl. region growing) [Vanc03][Wilk02][Riet05] in der flächenbasierten Segmentierung die einzelnen Teilpunktmengen als spezifische Objektoberflächen extrahiert. Es gibt viele Varianten von Bereichswachstumsverfahren, wobei viele mit digitalen Bildern [Kurt12] funktionieren. Bei dem, in dieser Arbeit verwendeten, Bereichswachstumsverfahren geht man folgendermaßen vor:

Zuerst werden die Punkte aus der vorliegenden Punktmenge anhand deren minimalen und maximalen Hauptkrümmungswerte und lexikographisch aufsteigend sortiert [Vanc03].

Seien P_i und P_j $(i \neq j)$ zwei Punkte aus der Punktmenge und die Funktion Index() gäbe die Stelle eines Punktes in der sortierten Liste, dann herrscht die folgende Ordnung innerhalb der Liste l:

$$\text{Index}(P_i) < \text{Index}(P_j) \text{ wenn } |\kappa_{min}(P_i)| < |\kappa_{min}(P_j)|$$

oder

$$|\kappa_{min}(P_i)| = |\kappa_{min}(P_j)| \text{ und } |\kappa_{max}(P_i)| < |\kappa_{max}(P_j)|$$

Die Sortierung der Punkte und die Auswahl eines Startpunktes für ein neues Segment aus dieser Liste stellt es sicher, dass die Segmentierung mit dem geometrisch einfacheren Teilobjekt anfängt und zu komplexeren Teilobjekte hin fortgesetzt wird [Vanc03].

Begonnen wird mit einem Startpunkt als neues Segment. Der Startpunkt wird aus einer sortierten Liste l ausgewählt und als "zugeordnet" markiert.

1. Jeder *direkte* Nachbar eines beliebigen Punktes aus dem Segment wird auf gleiche Krümmungseigenschaft (Homogenitätskriterium) hin untersucht. Die *direkten* Nachbarn für einen beliebigen Punkt in einem Segment sollen durch die in Kapitel 3.3.3.2 beschriebenen Eliminationsmethode ermittelt werden.

2. Das aktuelle Segment wird erweitert, indem ein *direkt-benachbarte* Punkt, der das Homogenitätskriterium erfüllt, d.h. er gemäß dem vordefinierten Schwellenwert eine nahezu ähnliche Krümmungseigenschaft aufweist, in das aktuelle Segment hinzugefügt wird. Der hinzugefügte Punkt wird in der sortierten Liste l als "zugeordnet" markiert.

3. Wird kein neuer Punkt mit nahezu ähnlicher Krümmungseigenschaft in der Nachbarschaft der Punkte vom aktuellen Segment gefunden, muss ein neues Segment mit einem neuen Startpunkt erzeugt werden, dessen Startpunkt aus der obigen sortierten Liste ausgewählt wird und bisher zu keinem anderen Segment gehört (nicht markiert). Das neue Segment wird als aktuelles Segment wie oben erweitert.

4. Die Schritte werden solange durchgeführt bis alle Punkte in der sortierten Liste l markiert sind.

Mit der krümmungsbasierten Segmentierung wird die gesamte Punktwolke, die man sich als ein Gesamtsegment vorstellen kann, am Ende in mehrere einheitliche Teilsegmente zerlegt. Die mathematische Definition (in [Kurt12] für digitale Bilder) der krümmungsbasierten Segmentierung sieht wie folgt aus:

Aufteilung der vorliegenden Punktwolke S als Gesamtsegment aller Punkte in Teilsegmente (Cluster) S_i (i = 1, ..., n), so dass gilt:

1) $\bigcup_{i=1}^{n} S_i = S$
2) S_i (i = 1, ..., n) ist zusammenhängend
3) für $i \neq j$ gilt $S_i \cap S_j = \emptyset$
4) $H(S_i)$ = true für alle i = 1, …, n
5) $H(S_i \cup S_j)$ = false für $i \neq j$, wenn S_i und S_j Nachbarsegmente sind.

H ist dabei das Homogenitätskriterium und bezieht sich auf die Krümmungen mit unten vorgestellten Bedingungen.

Aufteilung der Punktwolke geschieht nach dem Charakteristikum der maximalen und minimalen Hauptkrümmungen [Vanc03]. Die minimalen und maximalen Hauptkrümmungen (κ_{min} und κ_{max}) auf einer Oberfläche (engl. *surface*) Ω definieren folgende Oberflächentypen:

Wenn $\kappa_{min} = \kappa_{max} = 0$ dann Ω ist ein Teil von Fläche.
Wenn $\kappa_{min} = \kappa_{max} \neq 0$ dann Ω ist ein Teil von Kugel.
Wenn $\kappa_{min} = 0 \wedge \kappa_{max}$ = konstant $\neq 0$ dann Ω ist ein Teil von Zylinder.
Wenn $\kappa_{min} = 0 \wedge \kappa_{max}$ erhöht/verringert sich linear entlang einer Achse, dann ist Ω ein Teil eines Kegels.

Bei der krümmungsabhängigen Segmentierung geht es um die Gruppierung der benachbarten Punkte mit gleichen Krümmungseigenschaften zu inhaltlich zusammenhängenden Bereichen, deren Punkte nach ihren spezifischen Merkmalen die Oberflächentypen eines bekannten geometrischen 3D-Grundprimitives repräsentieren.

Die hier vorgestellten und implementierten Segmentierungsverfahren basieren direkt auf Punkten mit deren Nachbarschaftsbeziehungen in der Punktwolke und auf den extrahierten geometrischen Oberflächeninformationen, wie z.B. Normalenvektoren und Krümmungen an den einzelnen Punkten. Die Nachbarschaftsbeziehungen werden mit Hilfe vom *kD*-Baum (siehe Kapitel 3.3.2) ermittelt. Die ermittelte Nachbarschaft für einen beliebigen Punkt wird durch die in Kapitel 3.3.3.2 neu entwickelte Eliminationsmethode so verfeinert, dass die Überprüfung der vordefinierten Kriterien nur zwischen direkten Nachbarpunkten durchgeführt wird. Durch die Erfüllung des Kriteriums wird der jeweilige direkte Nachbarpunkt dem aktuellen Segment hinzufügt.

4 Implementierung und Visualisierung

In diesem Kapitel wird die Implementierung der Programmlogik für den Prototyp ausführlich dargestellt. Dabei werden die im Verlauf des Programms erwähnungswerten Klassen, sowie ihre Methoden, Attribute und Operationen beschrieben.

In dieser Diplomarbeit wurden als Nachschlagewerke die Quellen [Wolf09] und [Apet10] herangezogen, um die Grundschritte des Objekterkennungsprozesses und die Visualisierung der Ergebnisse zu implementieren.

Aufgrund der großen Datenmengen und der daraus resultierenden Performanceanforderung wurde die Programmiersprache C++ verwendet. Microsoft Visual Studio wurde als Entwicklungsumgebung für die Programmierung benutzt.

4.1 Implementierung

Der Prototyp besteht aus einer ausführbaren Anwendungsdatei (exe-Datei) ***OBJECT_RECOGNITION*** und mehreren implementierten statischen Bibliotheken, die mit der ausführbaren Datei verlinkt werden. Die Anwendungsdatei ist als Startprojekt festgelegt, in dem die Main-Funktion (WinMain) definiert ist. Die Bibliothek-Dateien führen die Grundschritte (siehe Kapitel 3.1) aus und beinhalten somit die für den Objekterkennungsprozess nötigen Funktionen.

In diesem Abschnitt werden auf einzelnen wichtigen Bibliotheken näher eingegangen und am Ende dieses Abschnittes die Struktur der Implementierung dargestellt.

4.1.1 POINTS_OF_CLUSTER

Die Bibliothek ***POINTS_OF_CLUSTER*** dient zur Generierung einer künstlichen Punktwolke, die zum Testen des im Rahmen dieser Diplomarbeit implementierten Objekterkennungsprozesses geeignet ist.

4.1.2 ALL_GLOBAL_TYPES

In der Bibliothek ***ALL_GLOBAL_TYPES*** wurden die Datentypen bzw. die Datencontainer definiert, welche die nötigen Informationen für die Punkte aus der Punktwolke speichern. Diese sind als Strukturen oder Klassen realisiert worden und enthalten Attribute und Methoden, um die Punktdaten weiter zu verarbeiten. Zur Funktionalität gehören unter anderem alle grundlegenden Operationen auf Vektoren (wie z.B. Addition, Subtraktion, Kreuzprodukt im 3D und in 2D, Skalarprodukt) usw.

Folgende Header-Dateien sind darin implementiert worden:

simple_types.h: Diese Header-Datei enthält folgende definierte Klassen:

a) vertex2D:
Ähnlich wie vertex3D, aber nur im zweidimensionalen Raum.

b) vector2D:
Ähnlich wie vertex3D, aber nur im zweidimensionalen Raum.

c) vertex3D:
Diese Klasse dient dazu einzeln eingelesene Punkte aus einer Eingabedatei zu speichern und enthält Attribute und Methoden, mit denen man die Eigenschaften eines Punktes belegen und ausgeben kann.
Die Attribute dieser Klasse sind:

`wx, wy, wz`	:	Diese Attribute sind die Punktkoordinaten eines Punktes *P*, der in einem dreidimensionalen kartesischen Koordinatensystem angegeben ist.
`index`	:	Die Position des Punktes *P* in einer Array-Liste, in der alle Punkte aus der Punktwolke abgelegt sind.
`wnormal`	:	In diesem Attribut wird das geometrische Merkmal Normalenvektor zu einem betrachteten Punkt *P* abgelegt. Der Punkt *P* ist vom Typ vertex3D.
`kmin`	:	In diesem Attribut wird die minimale Hauptkrümmung gespeichert, die an einem betrachteten Punkt *P* vom Typ vertex3D berechnet wird.
`kmax`	:	In diesem Attribut wird die maximale Hauptkrümmung gespeichert, die an einem betrachteten Punkt *P* vom Typ vertex3D berechnet wird.
`midpoint_wx, midpoint_wy, midpoint_wz`	:	Diese Attribute sind die *x*-, *y*- und *z*-Koordinaten eines Mittelpunktes, der aus den Nachbarpunkten eines betrachteten Punktes *P* ermittelt wird (siehe Abbildung 33).
`projectplane_p1_wx, projectplane_p1_wy, projectplane_p1_wz`	:	Die *x*-, *y*- und *z*-Koordinaten des nächst nahe liegenden Nachbarpunktes P_1 vom betrachteten Punkt *P* aus.
`projectplane_p2_wx, projectplane_p2_wy, projectplane_p2_wz`	:	Die *x*-, *y*- und *z*-Koordinaten eines weiteren Punktes P_2, der mit dem nächst nahe liegenden Nachbarpunkt P_1 und dem betrachteten Punkt *P* gemeinsam die jeweilige Best-Projektionsebene aufspannt (siehe Abbildung 33).

Die Methoden der Klasse vertex3D sind:

i. `get_wx(),get_wy(),get_wz()`: Diese Methoden geben die dreidimensionalen kartesischen Koordinaten des jeweiligen Punktes zurück.

ii. `get_midpoint_wx(),get_midpoint_wy(), get_midpoint_wz()`: Diese Funktionen liefen die dreidimensionalen kartesischen Koordinaten des Mittelpunktes zurück, welcher aus den Nachbarpunkten von dem jeweiligen betrachteten Punkt *P* ermittelt wurde. Der Mittelpunkt ist nötig, um die Nachbarpunkte in einer Best-Projektionsebene nach ihren Polarwinkeln zu sortieren (siehe Kapitel 3.3.3.1).

iii. `get_projectplane_p1_wx(), get_projectplane_p1_wy(), get_projectplane_p1_wz()`: Mit diesen Methoden wird der nächst nahe liegende Nachbarpunkt P_1 des jeweiligen betrachteten Punktes ausgegeben (siehe Abbildung 33).

iv. `get_projectplane_p2_wx(), get_projectplane_p2_wy(), get_projectplane_p2_wz()`: Mit diesen Methoden wird der Nachbarpunkt P_2 des

betrachteten Punktes *P* ausgegeben. Der Punkt P_2 ist nötig, um die entsprechende Best-Projektionsebene mit dem Punkt *P* und dem nächst nahe liegenden Nachbarpunkt des betrachteten Punktes *P* aufzuspannen (siehe Abbildung 5, in dem Beispiel entspricht P_5 dem Punkt P_2 hier).

v. `get_index()`: Diese Funktion liefert den Index von dem Punkt *P*.

vi. `get_kmin()`: Diese Methode gibt den Wert der minimalen Hauptkrümmung zurück, die als geometrisches Merkmal von dem Punkt *P* berechnet wurde.

vii. `get_kmax()`: Diese Methode gibt den Wert der maximalen Hauptkrümmung zurück, die als geometrisches Merkmal von dem Punkt *P* berechnet wurde.

viii. `get_wnormal()`: Diese Methode gibt den Normalenvektor zurück, der an einem betrachteten Punkt *P* als geometrisches Merkmal berechnet wurde (siehe Kapitel 3.3.3.2.4).

ix. `set_wnormal()`: Diese Methode weist dem Attribut `wnormal` einen Normalenvektor zu. Dieser Vektor wird nach einer optimierten lokalen Triangulation an dem betrachteten Punkt *P* berechnet (siehe Kapitel 3.3.3.2.4).

x. `set_midpoint()`: Diese Methode setzt die Koordinaten des Mittelpunktes, der aus den Nachbarpunkten zu einem betrachteten Punktes *P* ermittelt wurde.

xi. `set_projectplane_p1()`: Die Koordinaten des nächst nahe liegenden Nachbarpunktes P_1 von dem betrachteten Punkt *P* werden mit dieser Methode gesetzt.

xii. `set_projectplane_p2()`: Diese Methode setz die Koordinaten von Punkt P_2, der zusammen mit dem nächst nahe liegenden Punkt P_1 und dem betrachteten Punkt *P* eine Best-Projektionsebene aufspannt. (siehe Kapitel 2.3.1).

xiii. `set_index()`: Diese Methode versieht den Punkt mit einem eindeutigen Index, während er eingelesen wird.

xiv. `set_kmin()`: Mit dieser Methode wird das Attribut `kmin` belegt.

xv. `set_kmax()`: Mit dieser Methode wird das Attribut `kmax` belegt.

d) vector3D:
Diese Klasse stellt einen benutzerdefinierten Datentyp für Vektoren dar, mit deren Hilfe z.B. die Vektoren für die Best-Projektionsebene (siehe Kapitel 2.3.1) definiert werden oder die Polarachse festgelegt wird (siehe Abbildung 34), usw.

e) OrgNeighbors:
Diese Klasse ist ein selbst definierter Datentyp, in dem der Vergleichsoperator "<" überladen wurde, um mit Hilfe eines "*multiset*" (STL) die Nachbarpunkte aufsteigend nach ihrem Abstand zu dem betrachteten Punkt *P* zu sortieren. Der erste Punkt im *multiset* entspricht dabei dem betrachteten Punkt *P* und der zweite Punkt dem nächst nahe liegenden Punkt P_1, welche zur Bestimmung der Best-Projektionsebene (siehe Kapitel 2.3.1) nötig ist.

f) ProjNeighbors:
ProjNeighbors ist ähnlich wie OrgNeighbors ein Datentyp für die Sortierung der Nachbarpunkte mit einem *mulitset*. Der Unterschied ist, dass hier die Nachbarpunkte nach der Projektion in die Best-Projektionsebene nach Polarwinkeln sortiert werden.

simple_types.cpp: In dieser Quelldatei sind folgende Hilfsfunktionen definiert:

i. `IsTriangleOrientedCounterclockwise()`: Diese Hilfsfunktion bestimmt zuerst die Projektionsebene, die durch zwei der drei kartesischen globalen Koordinatenachsen aufgespannt wird. Nachdem die Projektionsebene ermittelt wurde, werden die Eckpunkte einer Dreiecksfläche in diese Ebene projiziert, so dass man mit dem Kreuzprodukt der zwei Kanten des Dreiecks die Orientierung der Dreiecksfläche in 2D bestimmt (siehe Kapitel 2.4).

ii. `projected_point()`: Diese Hilfsfunktion projiziert einen Punkt in eine vorher bestimmte Best-Projektionsebene (siehe Kapitel 2.3.1).

iii. `d_of_plane()`: Diese Hilfsfunktion berechnet die Konstante d nach der Ebenengleichung E : a.x + b.y + c.z + d = 0.

iv. `distan3d()`: Diese Hilfsfunktion berechnet den Abstand zwischen zwei Punkten, deren Koordinaten im dreidimensionalen kartesischen globalen Koordinatensystem angeben sind.

v. `calcangle_radian_3D()`: Diese Hilfsfunktion berechnet den Winkel zwischen zwei Vektoren in 3D. Der Ausgabewert ist Radiant.

vi. `calcangle180_grad_3D()`: Diese Hilfsfunktion berechnet den Winkel zwischen zwei Vektoren in 3D. Der Ausgabewert ist Grad.

vii. `crossproduct3D()`: Diese Hilfsfunktion berechnet das Kreuzprodukt zwischen zwei Vektoren in 3D. Das Kreuzprodukt ist ein Vektor vom Datentyp vertex3D.

viii. `crossproduct2D()`: Diese Hilfsfunktion berechnet das Kreuzprodukt zwischen zwei Vektoren in 2D. Das Kreuzprodukt ist ein Skalar.

4.1.3 KDTREE

Die Bibliothek ***KDTREE*** ist zum Aufbau des *kD*-Baumes zuständig. Mit Hilfe des *kD*-Baumes werden die Nachbarpunkte eines betrachteten Punktes *P* ermittelt. Die Nachbarpunkte braucht man, um die lokale Triangulation in dem Punkt *P* zu erstellen, womit man dann die geometrischen Oberflächenmerkmale wie Normalenvektor und Krümmung an dem Punkt *P* abzuschätzen kann.

4.1.4 LOCAL_TRIANGULATION_WITH_DIRECT_NEIGHBOR_POINTS

In dieser der Bibliothek sind folgende Funktionen implementiert:

i. `EliminationOfHigherGrades()`: Bevor die Elimination der störenden Punkte durchgeführt wird, entfernt diese Funktion die Nachbarpunkte zweiten oder höheren Grades in der zugehörigen Best-Projektionsebene (siehe Abbildung 51).

ii. `EliminateIndirectNeighbors()`: Diese Funktion eliminiert die *störenden* Nachbarpunkte eines betrachteten Punktes *P*, welche entstehen können, wenn man eine nichtoptimale lokale Triangulation durchführen würde. Die Elimination findet ebenfalls in einer zugehörigen Best-Projektionsebene statt (siehe Kapitel 3.3.3.2.3).

iii. `getDirectNeighborsForTriangulation()`: Die Funktion übernimmt alle Nachbarpunkte eines betrachteten Punktes *P,* welche vorher nach ihren Polarwinkeln in der Best-Projektionsebene sortiert wurden. Der Rückgabewert der Funktion ist eine Liste der *direkten* Nachbarpunkte des Punktes *P*. Für die Erzeugung der Liste werden die beiden oben

genannten Methoden `EliminationOfHigherGrades()` und `EliminateIndirectNeighbors()` aufgerufen. Diese Liste wird in der folgenden Bibliothek weiter verwendet.

4.1.5 CALCULATION_OF_GEOMETRIC_FEATURES

Die Methode `calculateNormalvectorAndCurvature()` der in dieser Bibliothek vorhandenen Klasse führt die Berechnungen der geometrischen Oberflächenmerkmale wie Normalenvektor und Hauptkrümmungen an einem betrachten Punkt *P* durch. Der Eingabewert der Methode sind die direkten Nachbarpunkte eines betrachten Punktes *P*, d.h. die benötigten Nachbarpunkte, mit denen man die optimale lokale Triangulation um den Punkt *P* (siehe Kapitel 3.3.3.2.4) durchführen kann. Die Methode übernimmt die Punkte und berechnet daraus sowohl den Normalenvektor als auch die minimale und maximale Hauptkrümmung gemäß dem Satz von Gauß-Bonnet an dem Punkt *P*.

4.1.6 SEGMENTATION_OF_POINTS

Mit Hilfe der implementierten Methoden innerhalb dieser Bibliothek werden die Punkte aus der Punktwolke nach ihren zuvor berechneten Normalenvektoren segmentiert. Diese Art der Segmentierung wird als normalenbasierte Segmentierung bezeichnet und wurde in Kapitel 3.3.5.1 näher beschrieben. Durch die normalenbasierte Segmentierung werden die Kanten und die Übergänge zwischen geometrisch unterschiedlichen Teilobjekten in einer vorliegenden Punktwolke detektiert. Gleich nach der normalenbasierten Segmentierung wird die krümmungsbasierte Segmentierung durchgeführt, um die Oberfläche des gescannten Objektes aufzuteilen. Damit werden die Oberflächenpunkte den zugehörigen geometrischen 3D-Grundprimitiven zugeordnet.

4.1.7 OBJECT_RECOGNITION (Hauptprogramm)

Das Startprojekt ***OBJECT_RECOGNITION*** beinhaltet die main-Funktion. Es besteht aus der folgenden Header- und Quelldatei:

pointscluster.h: Diese Header-Datei enthält sowohl die Deklaration als auch die Definition der Klasse *pointscluster*, die die wichtigen Methoden und Attribute für die Verwaltung von einzelnen Punkten aus einer Punktwolke kapselt.

Die Attribute dieser Klasse sind wie folgt definiert:

`pnts`	:	Das Attribut `pnts` ist ein Array vom Datentyp `vertex3D`, in dem eingelesene Punkte aus einer Punktwolke gespeichert werden.
`vertex_count`	:	Setzt die Anzahl der eingelesenen Punkte aus einer Punktwolke.
`neighbor_count`	:	Setzt die Anzahl der Nachbarpunkte für einen betrachteten Punkt *P*, die vom *kD*-Baum geliefert werden.
`SegmentpointsWithNormalvectors`	:	Dieses Attribut ist ein Array vom Datentyp *long*, in dem die Indizes der segmentierten Punkte abgelegt werden. Die Punkte ergeben sich aus dem Resultat der normalenbasierten Segmentierung. Alle gefundenen Punkte zusammen bilden die scharfen Kanten und Übergänge zwischen geometrisch unterschied-

lichen Teilobjekten in einer Punktwolke.

Die Methoden dieser Klasse sind wie folgt definiert:

i. `kDTree()`: Durch den Aufruf dieser Funktion wird ein *kD*-Baum aufgebaut, in dem die Punkte aus einer Eingabedatei zeilenweise eingelesen werden. Dabei werden die Punkte mit eindeutigen Indizes versehen und gespeichert.

ii. `ConsiderPoint()`: In dieser Methode wird ein Objekt erzeugt, über das man auf den *kD*-Baum zugreift, um für einen betrachteten Punkt *P* mit Index *i* seine Nachbarpunkte abzufragen.

iii. `ProjectPlane()`: Diese Methode bestimmt die Best-Projektionsebene aus den Nachbarpunkten eines betrachteten Punktes *P*. Die Nachbarpunkte müssen zuvor mit *kD*-Baum ermittelt werden.

iv. `TriangulationWithPolarcoordiante()`: Diese Methode ist die Kernfunktion des im Rahmen dieser Diplomarbeit vorgestellten Objekterkennungsprozesses. Durch diese Methode wird eine lokale Triangulation für einen betrachteten Punkt *P* ausgeführt und dabei die Elimination der störenden Nachbarpunkte in der lokalen Triangulation intern realisiert. Somit entsteht eine optimierte lokale Triangulation.

v. `CalculateGeometricFeatures()`: Nachdem die direkten Nachbarpunkte zu einem betrachteten Punkt *P* für eine optimale lokale Triangulation ermittelt worden sind, werden in einem Schritt die geometrischen Oberflächenmerkmale wie Normalenvektor und Hauptkrümmungen an dem Punkt *P* berechnet.

vi. `NormalvectorBasedSegmentation()`: In dieser Methode wird ein Objekt vom Typ *SegmentationOfPoints* instanziiert. Über dieses Objekt wird die Methode für die Ausführung der normalenbasierten Segmentierung aufgerufen.

vii. `display()`: Diese Methode ist zur graphischen Darstellung der Punkte und deren Eigenschaften wie Normalenvektor, Krümmung, etc. zuständig.

viii. `update_pos()`: Diese Methode aktualisiert die Position der einzelnen Punkte aus der Punktwolke und deren berechnete Eigenschaften bei der Bewegung (wie Drehung, Verschiebung etc.) eines dargestellten Objektes in dem OpenGL-Fenster.

application.cpp: Innerhalb dieser Datei werden alle Header-Dateien der statischen Bibliotheken zur Verbindung mit der Anwendungsdatei inkludiert, von denen die Anwendungsdatei abhängig ist. Außerdem enthält sie weitere globale Funktionen und Variablen zur besseren Benutzbarkeit. Diese Quelldatei enthält die Hauptfunktion WinMain(), welche die Funktionalität des Gesamtprogramms ausführt. Die Hauptfunktion wird von Betriebssystem als Einstiegspunkt erkannt und wird aufgerufen, um das implementierte Programm zu starten und anschließend die Objekte im OpenGL-Fenster darzustellen. Die definierte Funktion `handle_input()` in der Quelldatei ermöglicht dem Benutzer das im OpenGL-Fenster dargestellte Objekt über die Tastatur zu bewegen.

4.2 Visualisierung der Ergebnisse mit OpenGL

Der in dieser Arbeit vorgestellte Objekterkennungsprozess ist in der Sprache C++ programmiert. Allerdings biete die Programmiersprache C++ keine eigenen Bibliotheken, um die Ergebnisse eines Programms zu visualisieren.

Die Bibliothek OpenGL wurde eingesetzt, um die Gesamtergebnisse der Grundschritte des Objekterkennungsprozesses zu visualisieren und auch die Zwischenergebnisse, wie z.B. eine lokale Triangulation um einen betrachteten Punkt *P*, visuell zu analysieren. Mehr über OpenGL kann man in [Apet10] finden.

Im Folgenden werden die wesentlichen Header-Dateien für die Visualisierung und einige zugehörige Klassen kurz beschrieben. Diese befinden sich in der Bibliothek **ALL_GLOBAL_TYPES**:

user_interface.h: Diese Header-Datei enthält die Klasse `user_interface`, welche unabhängig vom Betriebssystem die Benutzerinteraktion über eine Tastatur und Maus steuert, indem der Zustand der Eingabegeräte abgefragt wird.

matrix.h: In dieser Header-Datei ist die Klasse *matrix* deklariert, in der aufwendige Matrizenoperationen enthalten sind. Diese Operationen werden dazu verwendet, um die Bewegungen (wie z.B. Rotation, Verschiebung usw.) eines im OpenGL-Fenster dargestellten Objektes zu realisieren. Die Objekte werden dabei in einem dreidimensionalen kartesischen Koordinatensystem dargestellt.

screen_interface.h: Diese Header-Datei enthält die Klasse `hardware_interface`, die die Schnittstelle mit der Grafikkarte herstellt. Die Klasse kapselt die Methoden und Attribute, die für Erzeugung eines OpenGL-Fensters und dessen Inhalt erforderlich sind.

In Abbildung 56 wurde das Klassendiagramm mit Hauptklassen und deren Beziehungen dargestellt, welche oben vorgestellten wurden.

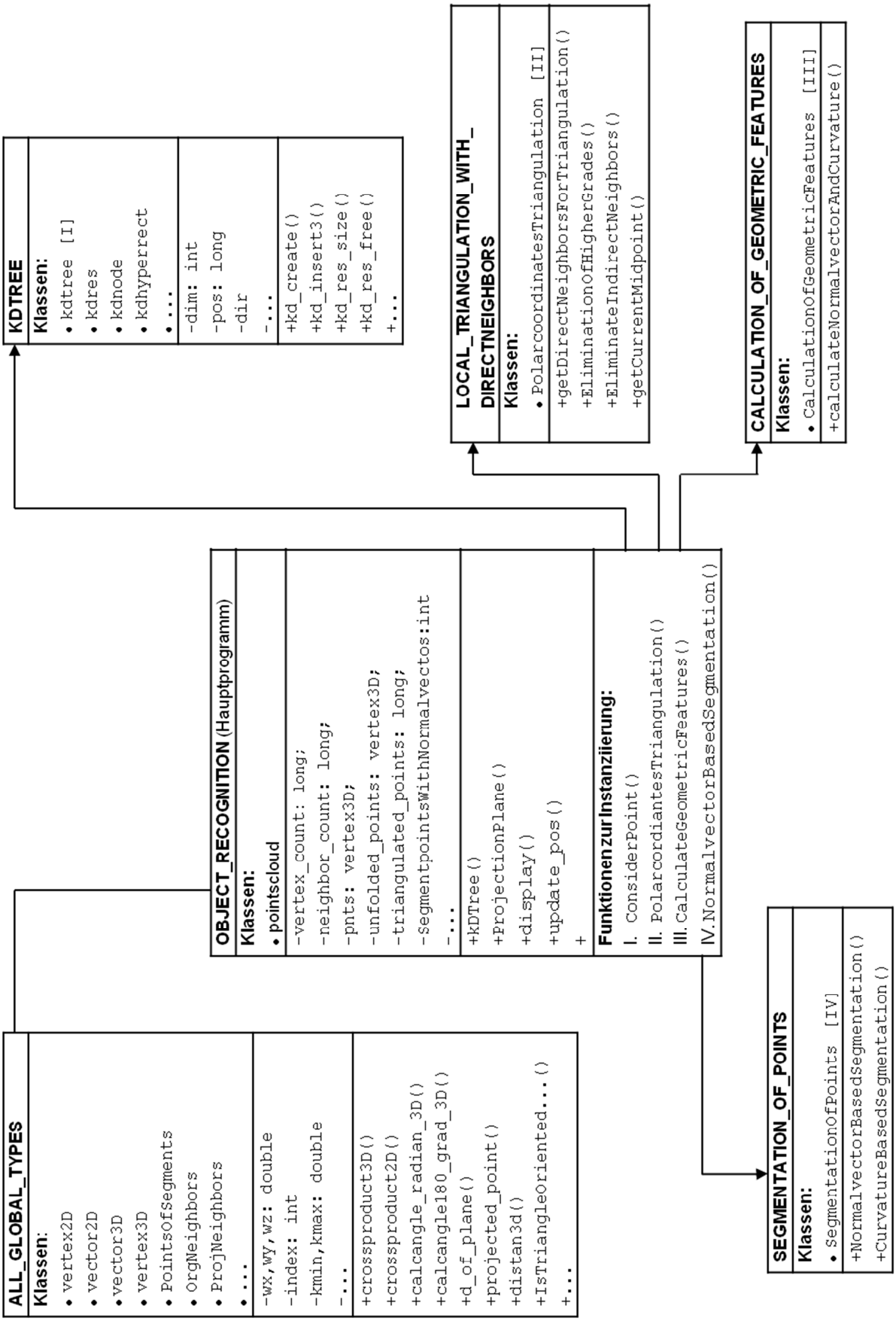


Abbildung 56: Klassendiagramm mit Hauptklassen und ihren Beziehungen

5 Ergebnisse und Analyse

Zur Veranschaulichung und Prüfung der eingesetzten und weiterentwickelten Verfahren und Algorithmen wurden im Rahmen dieser Arbeit künstliche Punktwolken (siehe Beispiel in Abbildung 57) aus den Punktdaten von zusammengesetzten geometrischen 3D-Grundprimitiven wie Würfel, Zylinder und Kugeln generiert.

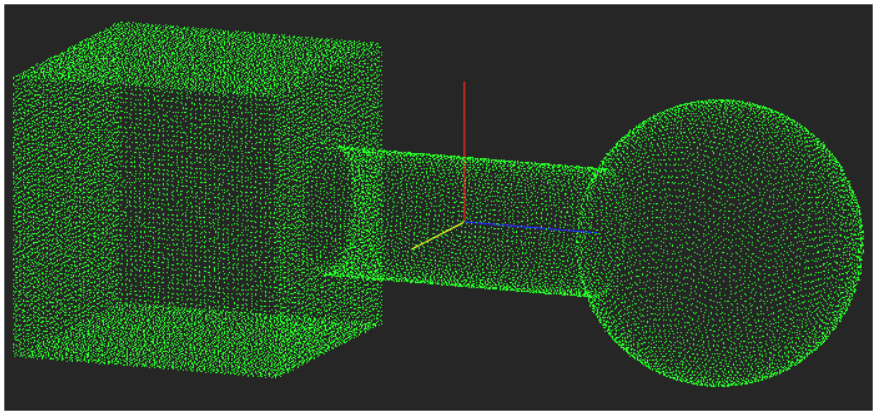


Abbildung 57: Beispiel für eine künstliche Punktwolke

In diesem Kapitel werden verschiedene Beispiele lokaler Triangulation und die Gesamtergebnisse der Grundschritte beim Objekterkennungsprozess präsentiert.

5.1 Analyse von Beispielen mit lokalen Triangulationen

Während der Implementierung des Objekterkennungsprozesses waren die Analyse und Visualisierung der Zwischenergebnisse an Hand von praktischen Beispielen erforderlich. Aus diesem Grund wurde ein Prototyp zur Analyse einzelner Punkte entwickelt, mit dessen Hilfe beispielsweise eine optimale im Vergleich zu einer nicht-optimalen lokalen Triangulation für einen beliebigen betrachteten Oberflächenpunkt aus der Punktwolke visualisiert wird.

In den folgenden Abbildungen sind einige Ausschnitte aus den lokalen Triangulationen um einzelne Oberflächenpunkte aus den verschiedenen Bereichen der Punktwolke dargestellt. Die Bereiche, in dem sich der betrachtete Punkt *P* mit seinen Nachbarpunkten befinden, sind in den Miniaturansichten der Punktwolke mit einem gelben Kreis markiert.

Das Dreieck in den Abbildungen 58-62 weiter unten entspricht der jeweiligen Best-Projektionsebene (siehe Kapitel 2.3.1), in die die ermittelten Nachbarpunkte eines betrachteten Punktes *P* (Eckpunkt des Dreieckes, mit der Farbe cyan gekennzeichnet) projiziert und anschließend nach ihren Polarwinkeln sortiert werden. Die rot-gefärbte Kante des Dreieckes ist die *Polarachse*, anhand der die Nachbarpunkte sortiert werden.

In den einzelnen Abbildungen 58-62 sind drei unterschiedliche Polygonzüge abgebildet. Jeder Polygonzug wird gebildet, indem die aufeinanderfolgenden Nachbarpunkte hinsichtlich ihrer Lage und Position miteinander verbunden werden. Die Reihenfolge der Punkte wird dabei durch die Sortierung der Nachbarpunkte in der zugehörigen Best-Projektionsebene nach ihren Polarwinkeln bestimmt.

Die gelb gefärbten Züge sind die Polygonzüge von allen Nachbarpunkten eines betrachteten Oberflächenpunktes *P*. Diese Polygonzüge bilden den Rand des Dreiecksnetzes einer nicht-optimalen lokalen Triangulation. Zur besseren Veranschaulichung wurden in den Abbildungen 58-62 die inneren Kanten dieser Dreiecksnetze nicht gezeichnet.

Eine lokale Triangulation von allen Nachbarpunkten eines betrachteten Punktes *P* kann überflüssige und störende Punkte enthalten, die auch als *indirekte Nachbarn* bezeichnet werden. Diese Punkte werden mit Hilfe der neu entwickelten Eliminationsmethode (siehe Kapitel 3.3.3.2.3) aus der *nicht-optimalen* lokalen Triangulation entfernt. Das Ergebnis ist ein Polygonzug mit den *direkten Nachbarn* (die cyan gefärbten Punkte), mit denen die *optimale* lokale Triangulation um den Punkt *P* erstellt wird (in den Abbildungen 58-62, die cyan gefärbten Polygonzüge).

Der dunkelblau gefärbte Polygonzug markiert alle Nachbarpunkte, die "aufgeklappt" (siehe Kapitel 3.3.3.2.2) in die Best-Projektionsebene projiziert und anschließend sortiert wurden. Auf diesen Polygonzug, wird die neu entwickelte Eliminationsmethode angewendet, um die *direkten Nachbarpunkte* des betrachteten Punktes *P* zu ermitteln.

In der Abbildung 58 ist der "aufgeklappte" Polygonzug (in den anderen Abbildungen dunkelblau) nicht abgebildet, da die Punkte dieses Polygonzuges mit den Punkten der nicht-optimalen Triangulation übereinstimmen. Der Grund dafür ist, dass alle Nachbarpunkte in einer Ebene liegen und daher kein "Aufklappen" dieser Punkte stattfindet. Um die Konvexität des Polygonzuges zu gewährleisten, d.h. die lokale Triangulation zu optimieren, wurden die überflüssigen Punkte (in Abbildung 58 in gelber Farbe) mit Hilfe der Eliminationsmethode aus dem Polygonzug entfernt. Der entstandene Polygonzug (cyan) ist konvex und enthält die *direkten Nachbarpunkte*, mit denen die optimierte lokale Triangulation an dem betrachteten Punkt *P* erstellt wird.

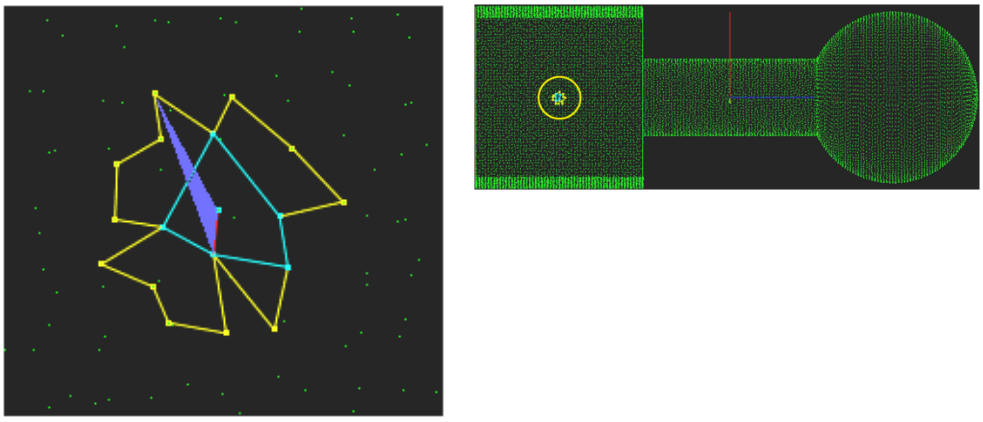


Abbildung 58: Beispiel für eine lokale Triangulation auf einer Ebene

Der Unterschied zwischen Abbildung 59 und Abbildung 58 besteht darin, dass in Abbildung 59 der betrachtete Punkt *P* auf der Oberfläche eines Zylinders liegt, so dass die in die Best-Projektionsebene "aufgeklappt" projizierten Nachbarpunkte und deren Polygonzug deutlich zu sehen sind. Die aufgeklappten Nachbarpunkte liegen in der zugehörigen Best-Projektionsebene.

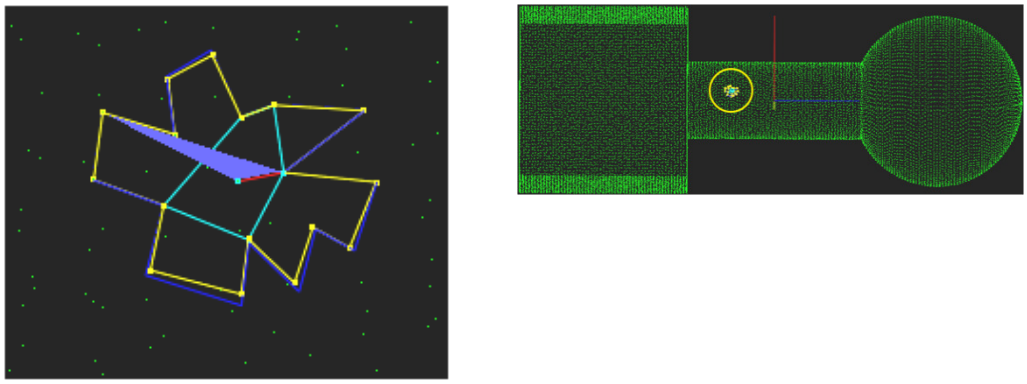


Abbildung 59: Beispiel für eine lokale Triangulation auf einem Zylinder

In Abbildung 60 sieht man den Rand einer nicht-optimalen Triangulation (gelb) und einer optimalen lokalen Triangulation (cyan) mit den "aufgeklappten" Nachbarpunkten (Eckpunkte des dunkelblauen Polygonzuges). Der Punkt *P* befindet sich in diesem Beispiel auf einer Kugel.

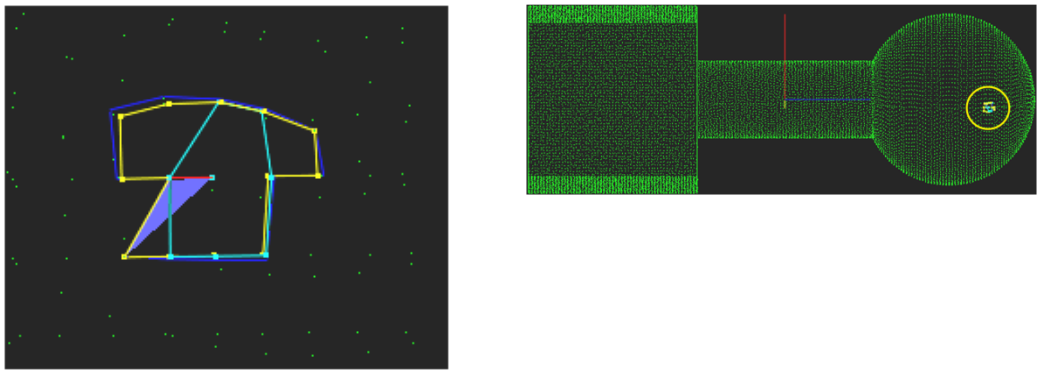


Abbildung 60: Beispiel für eine lokale Triangulation auf einem Kugel

In Abbildung 61 wird ein Punkt *P*, der an einer möglichen scharfen Kante (gestrichelte weiße Linie) auf einem Würfel, betrachtet. Nach der Sortierung der Nachbarpunkte entstehen neben den *überflüssigen Punkten* (gelb gefärbt), auch die *störenden Punkte* (markiert mit gestrichelten weißen Kreisen). Während die überflüssigen Punkte nur die Konvexität des entstandenen Polygonzuges stören, verfälschen die störenden Punkte die Richtung des Normalenvektors und den Wert der Krümmung, die an dem Punkt *P* berechnet werden. Alle diese Punkte wurden mit Hilfe der Eliminationsmethode (siehe Kapitel 3.3.3.2.3) basierend auf den "aufgeklappten" Polygonzug (dunkelblau) eliminiert. Das Ergebnis der Elimination ist ein Polygonzug (cyan) mit den *direkten Nachbarpunkten* (cyan), mit denen eine optimale lokale Triangulation erstellt und damit die Richtung des Normalenvektors als auch die Krümmung mit Hilfe des Satzes von Gauß-Bonnet (siehe Kapitel 3.3.4.2) präzise abgeschätzt werden.

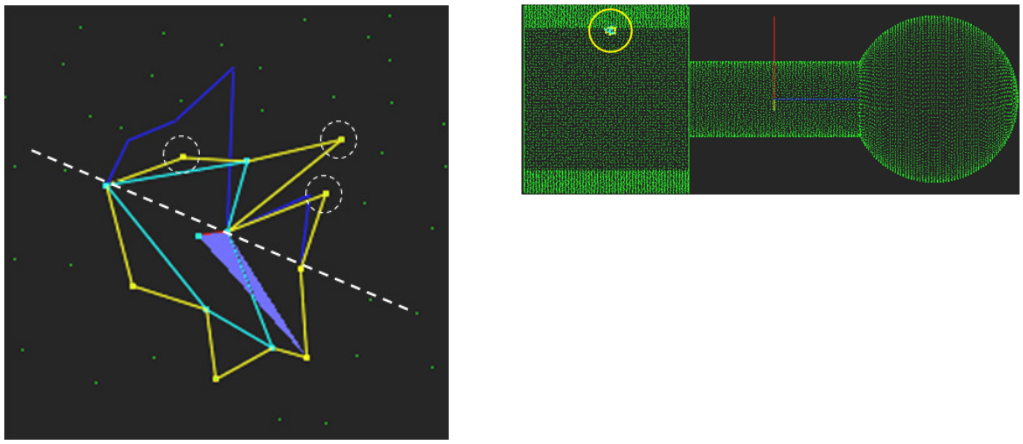


Abbildung 61: Beispiel für eine lokale Triangulation an einer scharfen Kante

Während sich der betrachtete Punkt *P* in Abbildung 61 an der Würfelkante befindet, liegt Punkt *P* in Abbildung 62 an dem Übergang (gestrichelter weiße Bogen) zwischen Zylinder und Kugel.

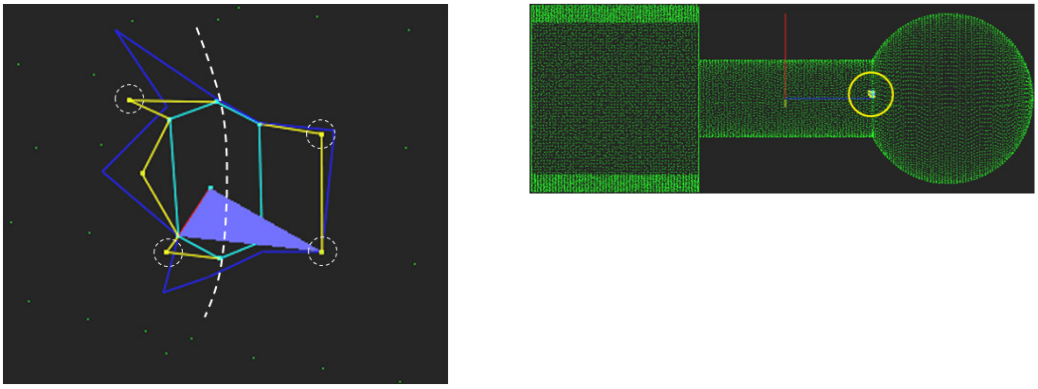


Abbildung 62: Beispiel für eine lokale Triangulation an einem Übergang zwischen Zylinder und Kugel

5.2 Ergebnisse der Grundschritte der Objekterkennung

Die visualisierten Ergebnisse der gewonnen geometrischen Oberflächenmerkmale wie Normalenvektor und Krümmung an den einzelnen Punkten, sowie Ergebnisse der normalenbasierten und krümmungsbasierten Segmentierungen, sind in den folgenden Abbildungen 63-65 veranschaulicht.

In Abbildung 63 sind die berechneten Normalenvektoren an den Oberflächenpunkten mit einem Farbverlauf dargestellt, wobei Anfang und Ende der Normalenvektoren jeweils von rot nach weiß verlaufen. Die berechneten Normalenvektoren sind zur Durchführung der normalenbasierten Segmentierung nötig, mit der die Kanten und Übergänge zwischen geometrisch unterschiedlichen Teilobjekten aufgespürt werden.

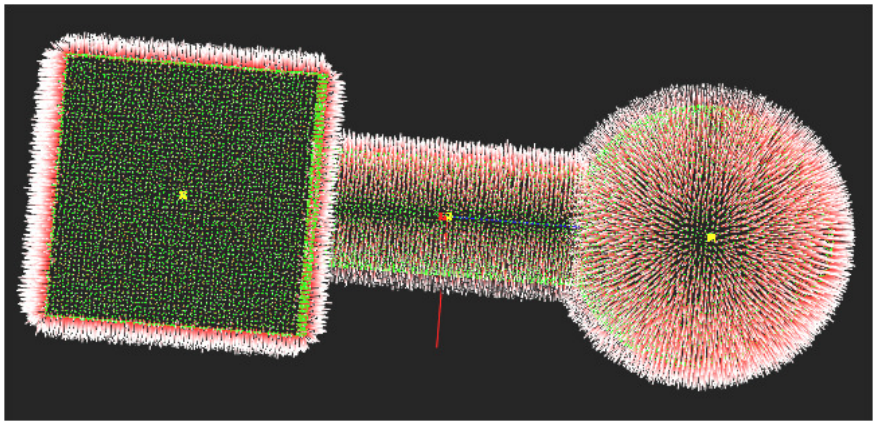


Abbildung 63: Normalenvektoren an den einzelnen Punkten aus der Punktwolke

In Abbildung 64 sind die Krümmungswerte an den jeweiligen Oberflächenpunkten wiedergegeben. Die weißen Punkte deuten darauf hin, dass auf dieser Oberfläche die minimale und maximale Hauptkrümmung Null sind. Dadurch lässt sich der Flächentyp Ebene erkennen. Mit gelb sind diejenigen Punkte markiert, bei denen die minimale Hauptkrümmung Null, die maximale Hauptkrümmung aber ungleich Null ist. Durch die dunkelblauen Punkte sind diejenigen Oberflächenpunkte gekennzeichnet, deren minimale und maximale Hauptkrümmungen ungleich Null sind. Dieser Fall tritt an scharfen Kanten und Übergängen (wie z.B. zwischen Würfel und Zylinder) als auch auf der Kugel ein.

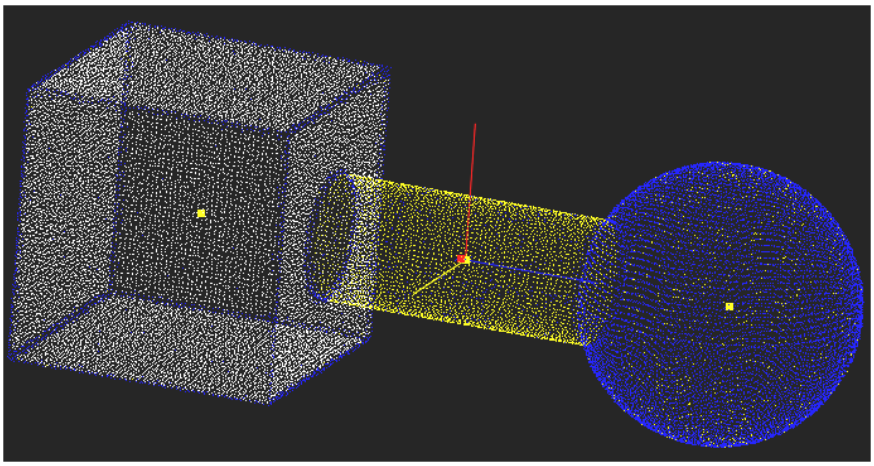


Abbildung 64: Krümmungen an den einzelnen Punkten aus der Punktwolke

Nach der Berechnung der geometrischen Oberflächenmerkmale wie Normalenvektor und Krümmung an den einzelnen Punkten aus der Punktwolke, wird die normalenbasierte Segmentierung durchgeführt.

Das Ergebnis der normalenbasierten Segmentierung ist eine Teilpunktmenge (Segment), welche die zusammenhängenden Punkte (die roten Punkte in Abbildung 65) enthält, die Teilobjekte in der vorliegenden Punktwolke eingrenzen. Solche Teilobjekte können Ebenen, Zylinder und Kugeln sein. Die rot gefärbten Punkte heben die scharfen Kanten des Würfels hervor und die Übergänge, an denen der Zylinder mit dem Würfel und der Kugel in der Punktwolke angrenzen.

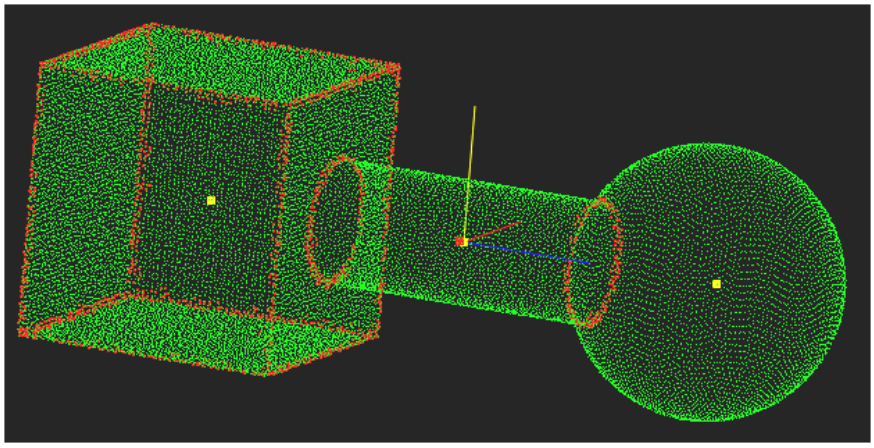


Abbildung 65: Beispiel für eine normalenbasierte Segmentierung. Mit Hilfe dieser Segmentierung werden die scharfen Kanten und Übergängen (rote Punkte) zwischen den geometrisch unterschiedlichen Teilobjekten in der Punktwolke detektiert.

In dem vorgestellten Objekterkennungsprozess wird gleich nach der normalbasierten Segmentierung die krümmungsbasierte Segmentierung durchgeführt, um die Oberfläche des gescannten Objektes in Teilflächen aufzuteilen. Dies dient zur Klassifizierung der Punkte in einer Punktwolke nach ihrer Krümmung als geometrisches Merkmal.

Das Ergebnis einer krümmungsbasierten Segmentierung sind mehrere Teilpunktmengen (auch Segmente genannt), welche die zusammenhängenden Punkte enthalten, die gleiche Krümmungswerte aufweisen und durch Kanten oder Übergänge eingegrenzt sind. In Abbildung 66 sieht man die Segmente, deren Punkte nach ihren einheitlichen Krümmungswerten mit gleicher Farbe dargestellt sind.

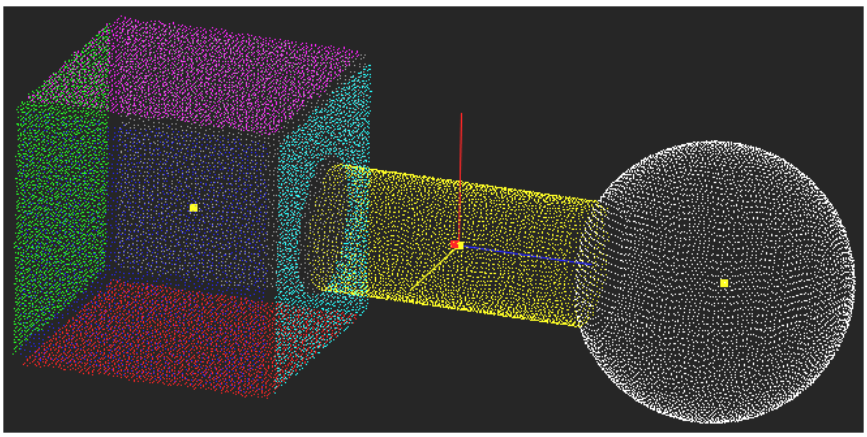


Abbildung 66: Beispiel für eine krümmungsbasierte Segmentierung

In diesem Fall besteht ein Würfel aus mehreren Teilflächen oder Segmenten (siehe Abbildung 66), die jeweils eine Ebene (in 3D) bilden, d.h. ein Würfel wird in einer vorliegenden Punktwolke nicht als Ganzes, sondern die Würfelseiten als ebene Flächen erkannt. Das Charakteristikum jeder solchen Ebene ist, dass sie durch Kanten eingegrenzt ist und jeder Punkt in dieser Ebene den Krümmungswert Null hat. Der Zylinder und die Kugel in diesem Beispiel haben eine eigene Oberfläche, d.h. es handelt sich um zwei getrennte Segmente, deren Punkten jeweils mit gelb und weiß visualisiert sind. Während die minimale Hauptkrümmung an allen Oberflächenpunkten bei einem Zylinder Null ist, besitzen alle Punkte eine maximale Hauptkrümmung, die ungleich Null ist. Bei einer Kugel sind die minimale als auch die maximale Hauptkrümmung für alle Punkte ungleich Null.

6 Zusammenfassung und zukünftige Arbeit

In diesem Kapitel werden die Grundzüge vom in dieser Diplomarbeit präsentierten Objekterkennungsprozess zusammengefasst und die relevanten Verbesserungen für zukünftige Arbeiten geschrieben.

6.1 Zusammenfassung

Diese Arbeit hat gezeigt, dass die Erkennung von geometrischen 3D-Grundprimitiven wie Ebene, Zylinder, Kugel etc. aus einer Punktmenge (Objekterkennungsprozess), in sich ein komplexer Prozess ist. Die Punkte werden durch Erfassung eines Maschinenteiles mit Hilfe eines Scanners generiert und liegen als unstrukturierte Punktwolke vor. Der Objekterkennungsprozess besteht aus mehreren Schritten und braucht einige optimale Methoden, um geometrische Merkmale an den gescannten Oberflächenpunkten zu gewinnen und effiziente Segmentierungsverfahren, um diese Punkte nach ihren Merkmalen in einheitlichen Teilpunktemengen aufzuteilen, so dass die aufgeteilten Punkte den bekannten geometrischen 3D-Grundprimitiven zugeordnet werden. Somit können die aus der Punktwolke erkannten geometrischen 3D-Grundprimitive parametrisiert und zur Weiterverarbeitung in die von CAD-Systemen bekannten Dateiformate konvertiert werden.

Zu diesem Zweck wurde im Rahmen dieser Diplomarbeit ein Prototyp implementiert, welcher eine unstrukturierte Punktwolke als Eingabedatei einliest und daraus die geometrischen Merkmale an den Oberflächenpunkten abschätzt. Anschließend werden die Punkte nach den abgeschätzten Oberflächenmerkmalen segmentiert und schließlich visualisiert.

Der wesentliche Vorteil von diesem Prototyp ist, dass das Gesamtverfahren direkt auf der unstrukturierten Punktwolke angewendet wird.

Bei der Entwicklung dieses Prototyps wurden analytische und numerische Lösungsansätze verglichen und schließlich zwei numerische Verfahren aus [Vanc03] implementiert und analysiert. Nach der Analyse wurde eines dieser Verfahren implementiert und so erweitert, dass ein neues Verfahren entstanden ist, das im Rahmen dieser Diplomarbeit als *Lokale Triangulation mit direkten Nachbarpunkten* bezeichnet wird. Innerhalb dieses Verfahrens wurde eine neue Eliminationsmethode entwickelt, mit deren Hilfe die lokalen Triangulationen um die betrachteten Oberflächenpunkten optimiert werden. Die Optimierung ermöglicht eine lokale Triangulation mit direkten Nachbarn eines betrachteten Punktes auf einer Teiloberfläche des gescannten Maschinenteiles. Mit dem Ergebnis einer optimierten lokalen Triangulation werden sowohl der Normalenvektor als auch die Krümmung mit Hilfe des Satzes von Gauß-Bonnet für einen betrachteten Punkt in einem einzigen Schritt effizient berechnet, ohne aufwendige analytische Lösungsansätze einzusetzen. Die Normalenvektoren und Krümmungen sind in den folgenden Grundschritten des Objekterkennungsprozess zur Segmentierung der Punktwolke nötig. Das neu entstandene Verfahren mit der internen Eliminationsmethode wird im Kapitel 3.3.3.1 näher beschrieben.

Die Ermittlung der direkten Nachbarpunkte für jeden Punkt in der Punktwolke erleichtert auch die Durchführung von zwei im Objekterkennungsprozess vorkommenden Segmentierungen. Beispielweise wird in der normalenbasierten Segmentierung die Prüfung der Normalenvektorabweichungen zwischen dem aktuellen Punkt und seinen *direkten* Nachbarpunkte durchgeführt, so dass bei der Detektion der scharfen Kanten und Übergängen zwischen geometrisch unterschiedlichen Teilobjekte keine Sprünge von einem Bereich in einen anderen stattfinden, die

durch diese Kanten oder Übergänge getrennt sind. Bei der krümmungsbasierten Segmentierung können dagegen die erzeugenden Segmente von einem Startpunkt aus mit Hilfe der *direkten* Nachbarn der Punkte, die sich schon in dem aktuellen Segment befinden, ohne Sortierung der Nachbarpunkte an Hand ihrer Abständen zu dem ursprünglichen Punkt *schrittweise* erweitert werden.

Zusammengenommen beschleunigen die oben genannten Vorgehen den Objekterkennungsprozess deutlich, so dass man bei einer generierten Punktwolke mit ca. 30.770 Punkten unter dem Betriebssystem Microsoft Windows XP, mit einem Intel-Prozessor Core 2 Quad CPU 2.84 GHz, 3,00 GB RAM innerhalb ca. 90 Sekunden gute Ergebnisse erzielt werden.

6.2 Zukünftige Arbeit

Die folgenden Arbeiten können den vorgestellten Prozess weiter verbessern und ergänzen:

- Der im Rahmen dieser Diplomarbeit implementierte Prototyp ist nicht durch die Erkennung von geometrischen 3D-Grundprimitiven, wie Würfel, Zylinder und Kugel, eingeschränkt. Durch Erweiterung des Prototyps wäre die Erkennung von weiteren geometrischen 3D-Grundprimitiven wie Kegel, Torus, Prisma etc. und deren 3D-Merkmale wie z.B. Löcher in einem Teilobjekt möglich.
- Eines von den Problemen der Objekterkennung und auch der Flächenrückführung aus einer Punktmenge ist das Nachbarschaftsproblem, da die Punktmenge als Eingabedatei unstrukturiert ist und keine Nachbarschaftsinformationen zwischen den einzelnen Punkten beinhaltet. Die Nachbarpunkte eines betrachteten Punktes werden im innerhalb dieser Diplomarbeit behandelten Objekterkennungsprozess durch einen *kD*-Baum ermittelt, mit dem man auch gute Ergebnisse erzielt. Aber durch Optimierung des *kD*-Baumes oder durch Verwendung eines anderen Algorithmus, könnte man den Objekterkennungsprozess beschleunigen. Eine Möglichkeit wäre beispielsweise. die Ermittlung der Nachbarpunkte mit einer Hash-Tabelle, welche in [Vanc03] näher erläutert wird.
- Eine größere Schwierigkeit bei einem Objekterkennungsprozess und einer Flächenrückführung ist die Bestimmung der Innen- und Außenseite der rekonstruierten Punktwolke. Die Lösung dieses Problem muss noch in den Prototyp, der im Rahmen dieser Diplomarbeit implementiert wurde, integriert werden, so dass der Prozess ohne Benutzerinteraktion funktioniert. Eine mögliche Lösung für dieses Problem ist in [Vanc03] zu finden.
- Da die meisten Maschinenteile (engl. machining parts), deren Oberfläche z.B. durch einen Scanner erfasst werden, nicht vollständig aus den bekannten geometrischen 3D-Grundprimitiven bestehen, ist oft eine Erweiterung vom Objekterkennungsprozess mit einem Flächenrückführungsalgorithmus nötig, so dass die vom Objekterkennungsprozess nicht als geometrische 3D-Grundprimitive erkannten Teile beispielsweise mit Dreieckstreifen rekonstruiert und als solche weiter bearbeitet werden.
- Nachdem die erfassten Oberflächenpunkte der Maschinenteile mit den bekannten geometrischen 3D-Grundprimitiven am Ende des vorgestellten Prozesses segmentiert werden, sollen die segmentierten Teilpunktmengen anschließend den jeweiligen geometrischen Objekten angepasst und parametrisiert werden, so dass die erkannten Objekte in den CAD-Systemen nicht mit zugehörigen Teilpunktmengen, sondern mit deren Parametern übertragen werden.
- Eine weitere Herausforderung für die Entwickler ist die Konvertierung der parametrisierten Ausgabedaten vom Objekterkennungsprozess in die bekannten Dateiformaten, wie z.B. STEP [Ande00], DXF [RSW 98] etc., so dass die von CAD-Systemen wie Inventor oder

Solidworks gelesen werden können. Dadurch werden die durch einen Scanner erfassten und anschließend rekonstruierten Maschinenteile am Rechner weiterverarbeitet.

Appendix A : Vektorräume und Vektoren

A.1 Körper

Eine Menge K mit zwei inneren Verknüpfungen + und · heißt Körper, wenn die folgenden zehn Bedingungen erfüllt sind [AHKK08]:

$\forall$ a, b, c $\in K$:

Gesetze der Addition:

(i) a + (b + c) = (a + b) + c	(Assoziativität)
(ii) a + b = b + a	(Kommutativität)
(iii) a + 0 = a	(neutrales Element oder Nullelement)
(iv) $\exists$a′, so dass a + a′ = 0	(inverses Element)

Gesetze der Multiplikation:

(v) a · (b · c) = (a · b) · c	(Assoziativität)
(vi) a · b = b · a	(Kommutativität)
(vii) 1 · a = a	(neutrales Element oder Einselement)
(viii) $\exists$a′, a $\neq$ 0, so dass a · a′ = 1	(inverses Element)

Distributivitätsgesetze:

(ix) a · (b + c) = (a · b) + (a · c)	(Linksdistributivität)
(x) (a + b) · c = (a · c) + (b · c)	(Rechtsdistributivität)

Die wichtigsten Körper sind die Mengen der reellen Zahlen $\mathbb{R}$, der rationalen Zahlen $\mathbb{Q}$ und der komplexen Zahlen $\mathbb{Z}$.

Vektoren sind aus der modernen Physik, Mathematik und natürlich auch der Informatik kaum noch wegzudenken. Um Punkte im Raum mit reellen Zahlen mit Hilfe von Vektoren zu beschreiben und auf ihnen Rechenoperationen wie Addition oder Multiplikation durchführen zu können, werden Vektorräume verwendet

A.2 Vektorraum

Im dreidimensionalen euklidischen Vektorraum mit dem Ursprung eines dreidimensionalen kartesischen Koordinatensystems als Anfangspunkt, werden Punkte als Vektoren beschrieben, welche miteinander addiert, subtrahiert oder mit einer reellen Zahl multipliziert werden, so dass z.B. die Länge eines Vektors oder der Abstand zwischen zwei Punkten berechnet werden kann. Für die formale Beschreibung des Vektorraumes ist zunächst die mathematische Definition des Körpers benötigt (siehe Körper).

Ein Tripel (V, +, ·) mit einer nichtleeren Menge V mit zwei Verknüpfungen „+“ (Vektor-Addition von Elementen aus V) und „·“ (skalare Multiplikation von Elementen aus V mit (skalaren) Elementen aus K) heißt ein Vektorraum über K (**Körper**) oder kurz K-Vektorraum, wenn die folgenden Axiome gelten [AHKK08]:
$\forall$ u, v, w $\in V$ und $\forall$ λ, $\mu \in K$:

(V1) v + w $\in V$ und λv $\in V$	(Abgeschlossenheit)
(V2) (u + v) + w = u + (v + w)	(Assoziativität)

(V3) $\exists 0 \in V$ mit v+0 = v (neutrales Element nach Addition)
(V4) $\exists v' \in V$ mit v + v′ = 0 (inverses Element nach Addition)
(V5) v + w = w + v (Kommutativität)
(V6) 1.v = v (neutrales Element nach Multiplikation)
(V7) λ (v + w) = λ v + λw
(V8) (λ + μ) v = λ v + μw
(V9) (λ μ) v = λ (μ v)

A.3 Euklidischen Vektorraum

Im Allgemein versteht man unter einem n-dimensionalen euklidischen Vektorraum einen reellen n-dimensionalen Vektorraum, der mit einem Skalarprodukt versehen ist.

Sei *V* ein reeller Vektorraum, dann ist ein *Skalarprodukt* auf den reellen Vektorraum *V* eine bilineare Abbildung [Ruzi12][Haro12a][Welk12]:

$$\langle , \rangle : V \times V \rightarrow \mathbb{R}$$

$$(u, v) \mapsto \langle u , v \rangle,$$

d.h. jeder Vektorenpaar (*u*, *v*) wird einer reellen Zahl $\langle u , v \rangle$ abgebildet, wobei für alle *u*, *v*, *w* ∈ *V* und für alle $\lambda, \beta \in$ IR die folgenden Eigenschaften erfüllt sind $\mathbb{R}$

(S1) $\langle u, v \rangle = \langle v, u \rangle$ (Symmetrie)

(S2) $\langle \lambda u + \beta v, w \rangle = \lambda \langle u, w \rangle + \beta \langle v, w \rangle$ (Bilinearität)

(S3) $u \neq 0 \Rightarrow \langle u, u \rangle > 0$ (positive Definite)

Seien a und b zwei Vektoren in dreidimensionalem euklidischen Vektorraum (a, b ∈ $\mathbb{R}^3$)

$$a = (a_x, a_y, a_z)^T \text{ und } b = (b_x, b_y, b_z)^T,$$

dann ist *Länge* (auch als *Betrag* oder *Norm* bezeichnet) eines Vektor a ∈ $\mathbb{R}^3$ in euklidischem Vektorraum definiert durch:

$$\|a\| = \sqrt{\langle a, a \rangle} = \sqrt{(a_x)^2 + (a_y)^2 + (a_z)^2}$$

Das *Skalarprodukt* (auch genannt inneres Produkt) zweier Vektoren a und b ergibt sich mit:

$$\langle a, b \rangle = a_x b_x + a_y b_y + a_z b_z$$

Der *Winkel* zwischen zwei von Nullvektor verschiedenen Vektoren a und b:

$$\cos \measuredangle(a, b) = \frac{\langle a, b \rangle}{\|a\| \|b\|}$$

$$\measuredangle(a, b) = \arccos \frac{\langle a, b \rangle}{\|a\| \|b\|}$$

Der (euklidische) Abstand von zwei Vektoren a und b:

$$d(a, b) = \langle a - b \rangle$$

Stehen zwei Vektoren *a* und *b* senkrecht zueinander, dann ist deren Skalarprodukt 0:

$$a \perp b \iff \langle a, b \rangle = 0$$

Die Addition zweier Vektoren ergibt wiederum ein Vektor:

$$a + b = (a_x, a_y, a_z)^T + (b_x, b_y, b_z)^T = (a_x + b_x, a_y + b_y, a_z + b_z)^T \in \mathbb{R}^3$$

Die Formel zur Multiplikation eines Vektors *a* mit einer reellen Zahl:

$$\lambda . a = \lambda . (a_x, a_y, a_z)^T = (\lambda a_x, \lambda a_y, \lambda a_z)^T$$

A.4 Normalenvektor

Der Normalenvektor kommt in dieser Arbeit in zwei unterschiedlichen Bedeutungen vor:

i. Der *Normalenvektor einer Ebene* ist ein Vektor, der senkrecht auf dieser Ebene steht und sich aus dem Kreuzprodukt von zwei Vektoren ergibt, die diese Ebene aufspannen. So eine Ebene kann beispielsweise eine Best-Projektionsebene (siehe Kapitel 2.3.1) oder eine Dreiecksfläche (siehe Kapitel 2.4) sein.

ii. Der *Normalenvektor an einem Punkt* ist der Durchschnitt der Normalenvektoren von Dreiecksflächen, die in einem Dreiecknetz einer lokalen Triangulation um einen betrachteten Punkt P vorkommen (siehe Kapitel 3.3.3.2.4).

Appendix B : Konvexität

B.1 Konvexe Menge

Eine Menge $M \subseteq \mathbb{R}^d$ ist konvex, falls je zwei beliebige Punkte der Menge *M*, also p ∈ *M* und q ∈ M als auch die Strecke S, die p und q verbindet,

$$\overline{pq} = S = \{\lambda p + (1 - \lambda)q : 0 \leq \lambda \leq 1\ \}$$

vollständig in der Menge M liegt (siehe Abbildung 67), d.h. S ⊆ M. [Hinr01][AuSp93].

Abbildung 67: Konvexe Menge: (links) konvexe Menge, (rechts) Menge, die nicht konvex ist

B.2 Konvexe Hülle (engl. convex hulls)

Sei M eine Menge von endlichen Punkten in der Ebene. Geometrisch ist die konvexe Hülle von *M* die kleinste konvexe Menge, die *M* umfasst. [Lang06].

B.3 Konvexer Polygon

Die konvexe Hülle von *M* ist ein *konvexes Polygon*, wenn *M* eine endliche Menge ist. Ein konvexes Polygon ist ein Spezialfall einer allgemeinen konvexen Menge von Punkten [Lang06]. Aus einer konvexen Hülle aus einer Menge M von Punkten im Koordinatensystem lässt sich ein Polygon erzeugen.

Ist die Verbindungsstrecke zwischen zwei beliebigen Punkten P_i und P_j der Menge von endlichen Punkten (Abbildung 68 (links)) nicht ganz im Polygon dieser Punktmenge enthalten (siehe Abbildung 68 (Mitte))), d.h. die Verbindungsstrecke schneidet den Rand des Polygons, dann ist das Polygon nicht konvex. Nichtkonvexe Polygone werden als *sternförmige* bezeichnet. Die Abbildung 68 (rechts) zeigt ein Beispiel für ein konvexes Polygon einer gegebenen Menge von endlichen Punkten.

Um Normalenvektoren an den Oberflächenpunkten von gescannten Objekte zu berechnen, werden mit Hilfe der *Lokalen Triangulation mit direkten Nachbarpunkten* (siehe Abschnitt 3.3.3.1) lokale Triangulationen, um diesen Punkten erstellt. So eine Triangulation entspricht in der Regel einem geschlossenen Polygon, innerhalb dessen der jeweilige betrachtende Punkt *P* liegt.

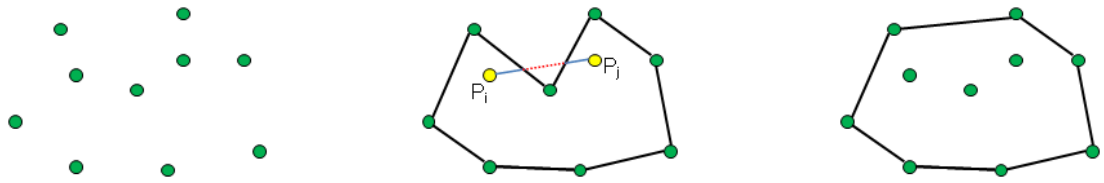


Abbildung 68: Konvexer Polygon: (links) Punktmenge, (Mitte) nicht konvexes Polygon, (rechts) konvexes Polygon der Punktmenge

Appendix C : Kurve und Fläche

C.1 Dreiecksfläche

Eine Dreiecksfläche ist eine Fläche, deren Umfang durch die Kanten eines Dreieckes abgegrenzt ist und die sowohl in 2D als auch in 3D die Eigenschaften einer Ebene aufweist (siehe Kapitel 2.4).

C.2 Kurve und Krümmung [MeWi95]

Die Kurve kann man in der Ebene mit folgen Formen darstellen:

1)	explizite (kartesische) Darstellung	$y = f(x)$,	$a \leq x \leq b$
2)	implizite (kartesische) Darstellung	$F(x, y) = 0$	
3)	parametrische Darstellung	$\vec{x} = \vec{x}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$,	$t_0 \leq t \leq t_1$

Ist $\vec{x} = \vec{x}(s)$ eine parametrische Darstellung einer ebenen Kurve bzgl. Bogenlänge s als Parameter und bezeichnet " ' " die Ableitung nach s, dann ist

i. $\vec{t}(s) := \vec{x}'(s)$ der *Tangenteneinheitsvektor* an $\vec{x}(s)$
ii. $\vec{x}''(s)$ ein Normalenvektor in $\vec{x}(s)$, so dass $\vec{t}(s) \perp \vec{x}''(s)$ gilt und $\vec{x}''(s)$ zeigt vom Kurvenpunkt in Richtung des *Krümmungskreismittelpunktes* M und
iii. Sei $\delta = \delta(s)$ der Winkel des Tangentenvektors zur positiven x-Richtung, dann ist die *Krümmung* der ebenen Kurve in $\vec{x}(s)$

$$\kappa(s) = \delta'(s)$$

und es gilt

$$|\kappa(s)| = |\vec{x}''(s)|$$

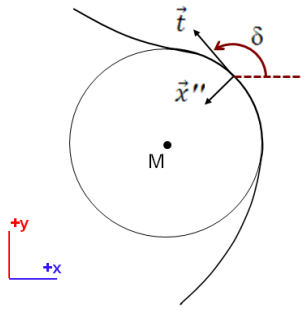


Abbildung 69: Die Krümmung δ der Ebene, Kurve in $\vec{x}(s)$

I. Die Kurve ist mit der parametrischen Darstellung in Raum:

$$\vec{x} = \vec{x}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}, t_0 \leq t \leq t_1$$

Ist $\vec{x} = \vec{x}(s)$ eine parametrische Darstellung einer Raumkurve bzgl. Bogenlänge s als Parameter und bezeichnet " ' " die Ableitung nach s, dann ist:

i. $\vec{t}(s) := \vec{x}'(s)$ der *Tangenteneinheitsvektor* an $\vec{x}(s)$
ii. $\vec{x}''(s)$ zum Krümmungsmittelpunkt weisender *Normalenvektor* in $\vec{x}(s)$
iii. $\kappa(s) = |\vec{x}''(s)|$ *Krümmung* der Raumkurve

C.3 Fläche [MeWi95][AuSp93]

Eine Fläche im anschaulichen Sinn ist eine zweidimensionale Teilmenge des dreidimensionalen Raumes, wie z.B. eine Ebene, eine zweidimensionale geometrische Figur oder die Begrenzungsfläche eines dreidimensionalen Körpers. Eine Fläche kann somit sowohl flach als auch gekrümmt sein.

Die Fläche im Raum kann mit folgenden Formen angeben:

1) explizite (kartesische) Darstellung $z = f(x, y)$, $(x, y) \in \mathbb{R}^2$
2) implizite (kartesische) Darstellung $F(x, y, z) = 0$
3) parametrische Darstellung $\vec{x}(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}$, wobei

   $(u, v) \in B \subseteq \mathbb{R}^2$, B der Parameterbeich, u und v die Parameter sind.

Definition: Seien $B \subseteq \mathbb{R}^2$ das Parametergebiet und

$$\varphi = \begin{cases} B \rightarrow \mathrm{E}^3 \\ (u, v) \mapsto X(u, v) = X\,(\,x(u, v) \mid y(u, v) \mid z(u, v)) \end{cases}$$

eine C^r-Abbildung ($r \geq 1$). Dann heißt die Punktmenge $M := \{ X(u, v) \mid (u, v) \in B \}$ eine C^r-Fläche mit der parametrischen Darstellung φ.

C.4 Flächenkurve [MeWi95][AuSp93]

Satz: Ist $M : \vec{x}(u, v)$, $(u, v) \in B$ eine reguläre C^r-Fläche ($r \geq 1$) und

$$\varphi = \begin{cases} I \subset IR \rightarrow B \\ t \mapsto (u(t), v(t)) \end{cases}$$

eine C^r-Abbildung mit $\left(\frac{du}{dt}(t), \frac{dv}{dt}(t)\right) \neq (0, 0) \quad \forall t \in I$, so ist die Flächenkurve

$$c : y(t) := \vec{x}(u(t), v(t)), \quad t \in I$$

von M eine reguläre C^r-*Kurve*.

Die Ableitungsvektoren $\vec{x}_u = \begin{pmatrix} x_u(u(t), v(t)) \\ y_u(u(t), v(t)) \\ z_u(u(t), v(t)) \end{pmatrix}$ und $\vec{x}_v = \begin{pmatrix} x_v(u(t), v(t)) \\ y_v(u(t), v(t)) \\ z_v(u(t), v(t)) \end{pmatrix}$ von $\vec{x}(u(t), v(t))$ nach den Parametern u und v sind *Tangentialvektoren* an die Parameterlinien. Man redet von den *Parameterlinien* oder *Koordinatenlinien*, wenn die Parameter u *oder* v bei der Fläche $\vec{x}(u(t), v(t))$ auf konstant gesetzt werden, also u = const und $v = t$ *oder* v = const und $u = t$. Die Tangentialvektoren spannen die *Tangentialebene* auf, auf der der Normalenvektor $\vec{n} := \vec{x}_u \times \vec{x}_v$ senkrecht steht.

Appendix D : Geometrische Grundbegriffe

D.1 Objekt

Im Rahmen dieser Diplomarbeit werden Gegenstände als Objekt bezeichnet, die z.B. durch Scanner erfasst werden. Bei Objekten in diesem Sinne kann es sich um Einzelobjekte handeln oder sie können aus mehreren Teilobjekten bestehen. Beispiele für solche Objekte sind Maschinenteile (engl. Machinig Parts), die meistens aus mehreren Teilobjekten bestehen.

Mit einem **Teilobjekt** bezüglich einer Punktwolke ist eine Teilmenge gemeint, die die Punkte enthält, an denen das jeweilige Teilobjekt ähnliche Oberflächeneigenschaften besitzt wie Normalenvektor oder Krümmung. Manche dieser Teilpunktmengen beschreiben die geometrischen 3D-Grundprimitive, z.B. Würfel, Zylinder, Kugel etc., die von dem Objekterkennungsprozesses in einer Punktwolke erkannt werden.

Ein **komplexes Objekt** (oder Teilobjekt) ist ein Objekt, das aus seiner Struktur her zu keinem geometrischen 3D-Grundprimitive zugeordnet werden kann. Daher werden solche Objekte speziell behandelt und falls nötig, mit einem bekannten Flächenrückführungsverfahren rekonstruiert.

D.2 Grundprimitiv

Ein geometrisches 3D-Grundprimitiv ist ein dreidimensionales Objekt, das geometrisch gleiche Eigenschaften an seinen Oberflächenpunkten aufweist, wie gleiche Krümmungen an jeder Stelle der Objektoberfläche. Beispiele für diese Objekte sind Ebenen, Zylinder, Kugel, Kegel etc.

D.3 Punkt

Um in der Computergrafik Positionen festzulegen werden Punkte eingesetzt. Ein Punkt ist ein geordnetes Paar reeller Zahlen; die Anzahl der Elemente des geordneten Paares ist gleich der Dimension des Raumes, der den Punkt enthält [Apet10].

Im Rahmen dieser Diplomarbeit kommen die folgenden unterschiedlichen Bezeichnungen für Punkte vor:

a. Ein ***störender* Punkt** ist ein Nachbarpunkt eines betrachtenden Punkt *P*, dessen Vorkommnis in einer lokalen Triangulation die Abschätzung des Normalenvektors und der Krümmung an dem Punkt *P* verfälschen kann und daher zur optimalen Triangulation entfernt wird. Zu diesem Zweck ist eine neue Methode entwickelt und in Kapitel 3.3.3.2 beschrieben.

b. Ein ***überflüssiger* Punkt** ist ein Begriff für einen Nachbarpunkt eines betrachtenden Punktes, dessen Elimination oder Beibehaltung die Richtung des gesuchten Normalenvektors zu dem betrachtenden Punkt nicht beeinflusst. Bei diesen Punkten handelt es sich eigentlich um *indirekte* Nachbarn. Sie werden bei lokalen Triangulationen mit Hilfe der neu entwickelten Eliminationsmethode zusammen mit den *störenden* Punkten entfernt, um die Konvexität des Polygonzuges zu gewährleisten. Der Polygonzug entsteht dabei durch Sortierung der Nachbarpunkte in der Best-Projektionsebene.

c. Ein ***Nachbarpunkt*** hinsichtlich eines betrachteten Punktes *P* ist ein Punkt, der in der Umgebung vom Punkt P innerhalb eines bestimmten Bereiches liegt. Die Nachbarpunkte für einen Punkt im 3D-Raum werden mit Hilfe des *kD*-Baumes (siehe Kapitel 3.3.2) ermittelt.

d. Es gibt keine feste Definition für die ***direkten Nachbarpunkte*** eines beliebigen Punktes in einer Punktmenge. Eine mögliche Beschreibung hängt von einer Triangulation ab, welche ein Dreiecksnetz ergibt. Ein *direkter* Nachbarpunkt (auch Nachbarpunkt *ersten Grades* genannt) ist der Punkt, der von dem betrachteten Punkt *P* aus in einem vorhandenen Dreiecksnetz nur über eine Kante (*direkt*) erreichbar ist. Die Nachbarpunkte, die von dem betrachteten Punkt *P* aus über *direkte* Nachbarpunkte erreichbar sind, heißen *indirekte* Nachbarpunkte oder Nachbarpunkte *zweiten* oder *höheren Grades*. Einen direkten Nachbarpunkt kann man sich auch als einen Punkt vorstellen, der nach der Sortierung der Punktmenge zu den "*inneren"* Punkten in der Graham-Scan-Methode bzw. zu dem Innenrand gehören, wobei der Innenrand auch Punkte aus der konvexen Hülle (Außenrand) (siehe Abbildung 14 und Abbildung 15) enthalten kann.

D.4 Scharfe Kante

Eine scharfe Kante (engl. sharp edge) ist ein Übergang zwischen zwei geometrisch gleichen Objekten (3D-Grundprimitiven), die an dieser Stelle angrenzen, wie z.B. eine Kante zwischen zwei Ebenen eines Würfels. An dieser Stelle weisen die jeweiligen geometrischen 3D-Grundprimitive Unregelmäßigkeiten in ihren Oberflächeneigenschaften auf wie z.B. starke Abweichungen zwischen den Normalenvektoren der einzelnen Punkte.

D.5 Merkmal

Als *geometrisches Merkmal* (engl. feature) an den einzelnen Oberflächenpunkten oder kurz *geometrische Oberflächenmerkmale* eines Objektes werden Werte oder Informationen bezeichnet, wie Normalenvektor oder Krümmung, mit deren Hilfe die untersuchten Punkte aus einer Punktwolke zu den zugehörigen Teilobjekten oder geometrischen Grundprimitiven zugeordnet werden.

D.6 Polyeder [Baer10]

Ein Polyeder X ist eine endliche Menge von Dreiecken $\Delta_j \subset \mathbb{R}^n$,

$$X = \{ \Delta_1, \ldots, \Delta_j \},$$

mit folgender Eigenschaft: Je zwei dieser Dreiecksfläche schneiden sich entweder

i. gar nicht oder
ii. in genau einer Ecke oder
iii. in genau einer Kante

Die Vereinigung dieser Dreiecke; $|X| := \bigcup_{j=1}^{k} \Delta_j$ heißt *geometrische Realisierung* von *X*.

D.7 Polygon [Sedg91]

Ein Polygon ist eine Liste von Punkten und dabei ist angenommen, dass aufeinanderfolgende Punkte durch Linien miteinander verbunden sind. Ein Polygon ist geschlossen, wenn der erste mit dem letzten Punkt verbunden ist.

D.8 Polygonzug [AuSp93]

Seien P_1, P_2,…, P_N, P_{N+1} die Punkte aus der Menge *M* und *M* ist Teilmenge der euklidischen Ebene E^2. Die Vereinigungsmenge $P_1P_2 \ldots P_NP_{N+1}$ von N Strecken $\overline{P_1P_2}$, …, $\overline{P_NP_{N+1}} \subset E^2$, bei denen je zwei aufeinanderfolgende Strecken genau einen Endpunkt gemeinsam haben, heißt *Streckenzug* (*von P_1 nach P_{N+1}*) *der Länge N*. Er heißt *einfach*, wenn je zwei nicht aufeinanderfolgende Strecken keinen Punkt gemeinsam haben. Er ist *geschlossen*, falls $P_1 = P_{N+1}$ gilt. Ein einfacher, geschlossener Streckenzug heißt Polygonzug.

Appendix E : Ausgleichsebene

Sei M der Schwerpunkt einer Menge $(P_1, \dots, P_n)$ von Punkte, dann ist M der arithmetische Mittel der Punkte und wird berechnet wie folgt [Wilk02]:

$$M = \frac{1}{n}\sum_{j=0}^{n} P_j$$

Mit Hilfe des Schwerpunktes berechnet man die Kovarianzmatrix *C:*

$$C = \sum_{j=0}^{n}(P_j - M).(P_j - M)^T \quad \text{mit } P_j = \begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix},\ M = \begin{pmatrix} x_M \\ y_M \\ z_M \end{pmatrix}$$

$$C = \begin{pmatrix} c_{1\_1} & c_{1\_2} & c_{1\_3} \\ c_{2\_1} & c_{2\_2} & c_{2\_3} \\ c_{3\_1} & c_{3\_2} & c_{3\_3} \end{pmatrix}$$

$$c_{1\_1} = \sum_{j=0}^{n}(x_j - x_M).(x_j - x_M)\ ;$$

$$c_{1\_2} = \sum_{j=0}^{n}(x_j - x_M).(y_j - y_M);$$

$$c_{1\_3} = \sum_{j=0}^{n}(x_j - x_M).(z_j + z_M);$$

$$c_{2\_2} = \sum_{j=0}^{n}(y_j - y_M).(x_j - x_M);$$

$$c_{2\_2} = \sum_{j=0}^{n}(y_j - y_M).(x_j - y_M);$$

$$c_{2\_3} = \sum_{j=0}^{n}(y_j - y_M).(z_j + z_M);$$

$$c_{3\_1} = \sum_{j=0}^{n}(z_j - y_M).(x_j - x_M)\ ;$$

$$c_{3\_2} = \sum_{j=0}^{n}(z_j - y_M).(y_j - y_M);$$

$$c_{3\_3} = \sum_{j=0}^{n}(z_j - y_M).(z_j + z_M);$$

Seien λ_1 , λ_2 , λ_3 die reellen Eigenwerte der symmetrischen Kovarianzmatrix *C*, für die O.b.d.A. die Bedingung gilt: $\lambda_3 \leq \lambda_2 \leq \lambda_1$. Ist das nicht der Fall, dann werden die Eigenwerte der Matrix so vertauscht, dass die Bedingung erfüllt wird.

Ist bei der erfüllten Bedingung $E_i = \begin{pmatrix} e_x^i \\ e_y^i \\ e_z^i \end{pmatrix}$ der normierte Eigenvektor des jeweiligen Eigenwertes λ_i der Matrix *C*, dann wird die Ausgleichsebene durch zwei Eigenvektoren E_1 und E_2 aufgespannt, zu der E_3 senkrecht steht.

Die Ausgleichsebene in Hessescher Normalform lautet somit folgendermaßen:

$$e_x^3 x + e_y^3 y + e_z^3 z = d = E_3.M$$

Der Normalenvektor N der Ausgleichsebene ist dann gleich dem Eigenvektor E_1 zum kleinsten Eigenwert λ_3.

Appendix F : Satz von Gauß-Bonnet

In diesem Teil des Appendixes werden die verschiedenen Formulierungen des Satzes von Gauß-Bonnet vorgesellt.

Sei $M \subset \mathbb{R}^3$eine Fläche, die die Oberfläche eines Objektes mit gescannten Punkten beschreibt und $\Omega \subset \mathbb{R}^3$eine geschlossene (orientierte) Fläche mit glattem Rand (ohne Randkurven mit Ecken) und einfach zusammenhängend in M, dann ist die Summe der Gaußschen Krümmung über diese Fläche plus die Summer der geodätischen Krümmung ist gleich 2π mal die Euler-Charakteristik der Fläche [Kühn03][Carm76]:

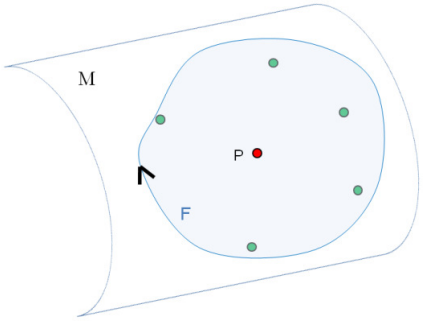


Abbildung 70: Fläche Ω mit Rand ohne Randkurven mit Ecken in einer orientierter Fläche M

$$\iint_{\Omega} K_G dM + \int_{\partial\Omega} k_g ds = 2\pi$$

Falls $M \subset \mathbb{R}^3$eine Fläche und Ω ein Polygon in M und Γ eine glatte und nach Bogenlänge parametrisierter geschlossener Randkurve Γ: [a,b] → M mit Außenwinkel $\gamma_1 \ldots \gamma_n$ des Polygons, die die Fläche berandet. Die Formel lautet dann für Randkurven mit Ecken [Roos06][Lang10] [Kühn03] (lokale Fassung):

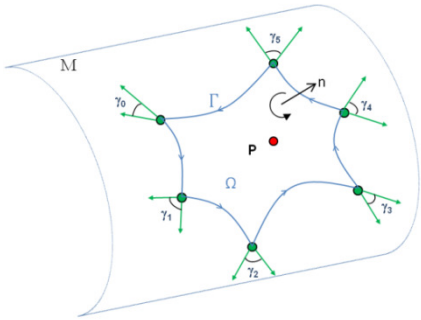


Abbildung 71: Polygon Ω mit Randkurve Γ in einer orientierten Fläche M

$$\iint_{\Omega} K_G dM + \int_{a}^{b} k_g ds + \sum_{i=0}^{n-1} \gamma_i = 2\pi,$$

wobei k_g die geodätische Krümmung von Γ bezüglich der in M festgelegten Orientierung ist und dM beschreibt das gewöhnliche Flächenelement von M.

Globale Fassung vom Ansatz erhält man, in dem man den auf kompakten randlosen Flächen verwendet. In diesem Fall ist der Teil

$$\int_{a}^{b} k_g ds$$

in der Gleichung gleich Null, Beweis siehe [Kühn03]. Dann sieht die Gleichung so aus:

$$\iint_{\Omega} K_G dM + \sum_{i=0}^{n-1} \gamma_i = 2\pi$$

Die Vereinfachung der Gleichung zur Abschätzung der minimalen und maximalen Hauptkrümmung bzgl. eines Polyeders (Dreiecksnetz) wird in dem Abschnitt 3.3.4 besprochen.

Literaturverzeichnis

[Agos05] Max K. Agoston: "*Computer Graphics and Geometric Modeling: Mathematics. Springer*", London, 2005

[AHKK08] Tilo Arens, Frank Hettlich, Christian Karpfinger, Ulrich Kockelkorn: "*Mathematik*", Spektrum Akademischer Verlag, ISBN: 978-3-8274-1758-9, Heidelberg 2008

[Ande00] Reiner, Anderl**:** "*STEP: standard for the exchange of product model data ; eine Einführung in die Entwicklung, Implementierung und industrielle Nutzung der Normenreihe ISO 10303*", Stuttgart - Leipzig, Teubner, 2000

[Apet10] Marius Apetri: "*3D-Grafik mit OpenGL: Das umfassende Praxis-Handbuch*", 1. Auflage, mitp Verlag, 2010

[AuSp93] Günter Aumann, Klaus Spitzmüller: "*Computerorientierte Geometrie*", Wissenschaftsverlag, Mannheim , 1993

[Baer10] Christian Bär: "*Elementare Differentialgeometrie*", 2. Auflage, De Gruyter Studium, 2010

[Bent75] Jon Luis Bentley: "*Multidimensional binary search trees used for associative searching*", In: Communications of the ACM , Vol. 18 , Nr. 9 , September (1975), S. 509-517

[BKOS00] Mark de Berg, Marc van Kreveld, Mark Overmars, Otfried Schwarzkopf: "*Computational Geometry – Algorithms and Applications*", Springer Verlag, 2000

[Boeh05] Jahn Böhm: "*Modellbasierte Segmentierung und Objekterkennung aus Distanzbildern*", Dissertationen, Fakultät für Luft- und Raumfahrttechnik und Geodäsie der Universität Stuttgart, 2005

[Boeh09] Jan Böhm: "*Übersicht über Ansätze zur automatischen Objektrekonstruktion aus Punktwolken*", Folien, Institut für Photogrammetrie Universität Stuttgart, http://www.geomatik-hamburg.de/tls/tls2009/images/03_tls2009_boehm.pdf, 2009

[Carm76] Manfredo Perdigao do Carmo: "*Differential Geometry of Curves and Surfaces*", Prentice-Hall, Inc., New Jersey, ISBN 0-13-212589-7, 1976,

[CFGG07] Adam Chachaj, David Frenzel, Jens Garstka, Lars Griehsel: "*Endbericht der Projektgruppe 489 Graphaeology.NET*", Universität Dortmund, 2007

[Drix93] Erwin Drixler: "*Analyse der Form und Lage von Objekten im Raum*". Deutsche Geodätische Kommission bei der Bayerischen Akademie der Wissenschaften, Reihe C: Dissertationen, Heft Nr. 409, 1993

[EmOb12] Franz Embacher und Petra Oberhuemer: "*Zeichenebene und Koordinatensystem*", Eine Galerie multimedialer Lernhilfen für Schule, Fachhochschule, Universität und Selbststudium", http://www.mathe-online.at/mathint/zeich/i.html, 2012

[Erne12] M. Erne: "*Skript zur Vorlesung Mathematik I für Bauingenieure, Kapi-tel 2.3 Vektorprodukt und Spatprodukt*", Universität Hannover, http://www. iazd.uni-hannover.de/~erne/Mathematik1/dateien/maple/MB_2_3.pdf, 2012

[Gems11] Andreas Gemsa: "*Folien für Vorlesung Algorithmische Geometrie, Konvexe Hülle im* $\mathbb{R}^3$", Universität Karlsruhe, 2011

[Gomo09] Thomas Gomoll: "*Numerische Bestimmung von Randkurven auf triangulierten Oberflächen zur Erstellung von NURBS Flächen*", Masterarbeit, Beuth Hochschule für Technik Berlin, 2009

[Harn12] Hergen Harnisch: "*Vortrags-Folien zur Vorlesungen Lineare Algebra und analytische Geometrie I-II*", Institut für Mathematik Humboldt-Universität zu Berlin, http://www.mathematik.hu-berlin.de/~harnisch/lehre/krzprod.pdf, 2012

[Haro12a] Dorothee D. Haroske: "*Skript zur Vorlesung Mathematik I für Ingenieure*", Kapital 1 Zahlen und Vektoren, Leibniz Universität Hannover, http://users.minet.uni-jena.de/~haroske/math_ing/math_ing-2.pdf, 2012

[Haro12b] Dorothee D. Haroske: "*Skript zur Vorlesung Mathematik I für Ingenieure, Kapital 1 Zahlen und Vektoren*", Leibniz Universität Hannover, http://users.minet.uni-jena.de/~haroske/math_ing/math_ing-3.pdf, 2012

[Hinr01] Klaus Hinrichs: "*Vorlesung Algorithmische Geometrie, Kapitel: Konvexe Hülle*“, Universität Münster, 2001

[HKKH12a] Klaus Höllig, Wolfgang Kimmerle, Werner Kratz, Jörg Hörner: "*Koordinatensystem*", Mathematik-Online-Lexikon, http://mo.mathematik.uni-stuttgart.de/lexikon/K/koordinatensystem.html, 2012

[HKKH12b] Klaus Höllig, Wolfgang Kimmerle, Werner Kratz, Jörg Hörner: "*Kugelkoordinaten*",Mathematik-Online-Lexikon, http://mo.mathematik.uni-stuttgart.de/ lexikon/K/koordinatensystem_kugelkoordinaten.html, 2012

[HKKH12c] Klaus Höllig, Wolfgang Kimmerle, Werner Kratz, Jörg Hörner: "*Polarkoordinaten*", Mathematik-Online-Lexikon, http://mo.mathematik.uni-stuttgart.de/ lexikon/K/koordinatensystem_polarkoordinaten.html, 2012

[HKKH12d] Klaus Höllig, Wolfgang Kimmerle, Werner Kratz, Jörg Hörner: "*Zylinderkoordinaten*", Mathematik-Online-Lexikon, http://mo.mathematik.uni-stuttgart.de/ lexikon/K/koordinatensystem_zylinderkoordinaten.html, 2012

[KKK 02] Sun-Jeong Kim, Soo-Kyun Kim, Chang-Hun Kim: "*Discrete Differential Error Metric for Surface Simplification*", IEEE Computer Society, Washington, 2002

[Koec97] Max Koecher: "*Lineare Algebra und analytische Geometrie*", 4. Auflage, Springer-Lehrbuch, 1997

[Kurt12] Winfried Kurth: "*Skript für Vorlesung Bildanalyse und Bildverstehen, Teil 5*", Institut für Informatik BTU Cottbus, http://www.uni-forst.gwdg.de/~wkurth/cb/html/bia3_v05.pdf, 2012

[Kühn03] Wolfgang Kühnel: "*Differentialgeometrie, Kurven – Flächen – Mannigfaltigkeitent*", Vieweg Verlag, 2003

[Lang06] Hans Werner Lang: "*Algorithmen in Java*", 2.Auflage, Oldenbourg Wissenschaftsverlag, 2006

[Lang10] Urs Lang: "*Differentialgeometrie I/II, Partielles Skript zu den Vorlesungen*", ETH Zürich, http://www.math.ethz.ch/~lang/dg.pdf, 2010

[MeWa00] Dereck S. Meek, Desmond J. Walton: "*On surface normal and Gaussian curvature approximations given data sampled from a smooth surface*", Computer Aided Geometric Design, 2000

[MeWi95] Gerhard Merziger, Thomas Wirth: "*Repetitorium der Höheren Mathematik*", 3. Auflage, Binomi-Verlag, 1995

[Mile11] Anton Milev: "*Kd Tree –Searching in N-dimensions, Part I, Optimized kd-tree and multidimensional nearest neighbours search*", http://www.codeproject.com/Articles/18113/KD-Tree-Searching-in-N-dimensions-Part-I, 2011

[MSR 07] Evgeni Magid, Octavian Soldea, Ehud Rivlin: "*A comparison of Gaussian an mean curvature estimation methods on triangular meshes of range image data*", Computer Vision and Image Understanding, 2007

[PrSh85] Franco P. Preparata, Michael lan Shamos: "*Computational Geometry An Introduction*", Springer Verlag, 1985

[Riet05] Andreas Rietdorf: "*Automatisierte Auswertung und Kalibrierung von scannenden Messsystemen mit tachymetrischem Messprinzip*", Dissertation, Technischen Universität Berlin, 2005

[Roos06] Beatrice Roost: "*Solitonlösungen des mittleren Krümmungsflusses*", Inaugural Dissertation, Universität Freiburg (Schweiz), http://ethesis.unifr.ch/theses/downloads.php?file=RoostB.pdf, 2006

[RSW 98] Dietmar Rudolph, Thomas Stürznickel, Leo Weissenberger: "*DXF intern*", CR/LF GmbH, 1998

[Ruzi12] Michael Ruzicka: "*Skript 3 - Kapital 2: Der euklidische Raum*", Skript zur Vorlesung Analysis II, Universität Freiburg, http://aam.mathematik.uni-freiburg.de/IAM/Teaching/scripts/ana2_SS05/skript-3.pdf, 2012

[Sedg91] Robert Sedgewick: "*Algorithmen*", 1. Auflage, Addison Wesley, Bonn, 1991

[Uvir09] Daniela Uvira: "*Methoden der digitalen Bildverarbeitung zum Auffinden von fluoreszierenden Punkten auf Messproben*", Seminararbeit, RWTH Aachen University, 2009

[Vanc03] Marek Vanco: "*A Direct Approach for the Segmentation of Unorganized Points and Recognition of Simple Algebraic Surfaces*". PhD thesis, University of Technology Chemnitz, 2003.

[Welk12] Dr. Martin Welk: "*Skript zur Vorlesung Mathematik für Informatiker II, § 40. Euklidische Vektorräume, Skalarprodukt*", Universität Saarland http://www.mia.uni-saarland.de/Teaching/MFI07/kap40.pdf, 2012

[Well02] Thomas Wellen: "*Algorithmen zur Suche in geometrischen Daten, Konvexe Hüllen*", Universität Bonn, 2002

[Werl11] Stefan Bruno Werling: "*Deflektometrie zur automatischen Sichtprüfung und Rekonstruktion spiegelnder Oberflächen*", Dissertation, Fakultät für Informatik des Karlsruher Instituts für Technologie (KIT), 2011

[WHH 10] Christopher Weber, Stefanie Hahmann, Hans Hagen: "*Sharp Feature Detection in Point Clouds*", IEEE International Conference On Shape Modeling And Application (SMI), 2010

[Wiki12a] Wikipedia, die freie Enzyklopädie, "*Polarkoordinaten*", http://de.wikipedia.org/wiki/Polarkoordinaten, 2012

[Wiki12b] Wikipedia, die freie Enzyklopädie, "*Objekterkennung*", http://de.wikipedia.org/wiki/Objekterkennung, 2012

[Wilk02] Wilhelm Wilke: "*Segmentierung und Approximation großer Punktwolken*", Dissertation, Technische Universität Darmstadt, 2002

[WLJD12] Chenglei Wu, Yebin Liu, Xiangyang Ji, Qionghai Dai: "*Multi-View Reconstruction Under Varying Illumination Conditions*", Department of Automation, Tsinghua University, Beijing China, 2012

[Wolf09] Jürgen Wolf: "*C++ von A bis Z: Das umfassende Handbuch*", Galileo Computing, 2. Auflage, 2009

[XuAl12] Yi Xu Daniel G. Aliaga: "*High-Resolution Modeling of Moving and Deforming Objects Using Sparse Geometric and Dense Photometric Measurements*", Department of Computer Science Purdue University, 2012

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift:

Stuttgart, den 14.05.2012