Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Studienarbeit Nr. 1991

Neuvernetzung mit Hilfe von Volumen

Alexander Kramer

Studiengang:	Informatik
Prüfer:	Prof. Dr. Thomas Ertl
Betreuerin:	Dipl. Math. Katrin Bidmon
begonnen am:	01. Dezember 2004
beendet am:	02. Juni 2005
CR-Klassifikation:	I.3.5, J.6

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Zielsetzung 2

1.2 Ansatz 2

1.3 Aufbau der Arbeit 2

2 Verwandte Arbeiten **3**

3 Verwendete Algorithmen **5**

3.1 Voxelisierung 6

3.2 Extraktion der Isofläche 11

3.3 Duales Netz 16

4 Verknüpfung der verwendeten Algorithmen **18**

4.1 Umrechnung der Koordinaten 18

4.2 Steigerung des Vierecksanteils 19

5 Erhalten der spezifischen Merkmale **23**

5.1 Anreicherung mit Merkmalpunkten 24

5.2 Merkmalbestimmung 27

6 Qualitätsverbesserung des Netzes **29**

6.1 Entfernen der Rhombuspolygone 29

6.2 Merkmalerhaltendes Glätten des Netzes 32

7 Ergebnisse **34**

7.1 Bewertung des rekonstruierten Polygonnetzes 34

7.2 Laufzeiten 35

8 Zusammenfassung und Ausblick **38**

1 Einleitung

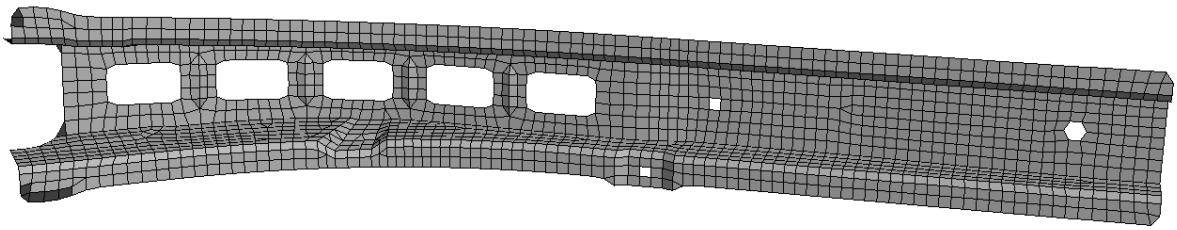


Abbildung 1.1: Ein FE-Polygonnetz eines Bauteils bestehend aus Vierecken und Dreiecken.

Durch computergestützte numerische Simulationen können Bauteile noch vor ihrer Fertigung auf physikalische Eigenschaften überprüft werden. Beispielsweise können in der Automobilindustrie die Karosserieteile durch Simulationen auf ihr Verhalten beim Aufprall untersucht werden. Die bei der Simulation eingesetzte Methode der finiten Elemente (FE) benötigt dazu eine spezielle Beschreibung der Form des Bauteils durch ein polygonales Netz, wobei die Polygone die finiten Elemente darstellen.

Bei strukturmechanischen Simulationen (z.B. Aufprallsimulationen) verhalten sich Polygonnetze mit großen Vierecksanteil stabiler als reine Dreiecksnetze. Es ist demnach für bestimmte Aufgabenbereiche erwünscht ein Vierecksnetz oder ein Polygonnetz mit großem Vierecksanteil statt eines Dreiecksnetzes zu verwenden. In Abbildung 1.1 ist ein Beispiel eines solchen FE-Polygonnetzes von einem Bauteil dargestellt, welches überwiegend aus Vierecken besteht.

Die ursprünglichen FE-Netze werden aus dem CAD-Modell[1] des jeweiligen Bauteils gewonnen. Endgültige Änderungen an den Bauteilen müssen im CAD-Modell und im FE-Netz konsistent erfolgen.

In der frühen Entwicklungsphase finden Änderungen häufiger statt. Damit die geänderten Bauteile möglichst schnell in Simulationen getestet werden können, wird die Änderung des CAD-Modells ausgelassen und das FE-Netz wird direkt geändert. Obwohl die endgültigen Änderungen wieder konsistent mit dem CAD-Modell sein müssen, können einzelne Designzyklen signifikant verkürzt werden, und somit die gesamte Entwicklungsphase.

Je nach Änderung muss für einen Teil des Netzes (lokal) oder für das ganze Netz (global) eine alternative Aufteilung in Polygone gefunden werden. Die Neuaufteilung in Polygone wird i.d.R. als Neuvernetzung bezeichnet. Man unterscheidet also zwischen dem lokalen und globalen Neuvernetzen.

[1] CAD = Computer Aided Design

1.1 Zielsetzung

Im Rahmen vorliegender Arbeit sollte ein Algorithmus für eine globale Neuvernetzung gefunden und prototypisch implementiert werden. Die Vorgaben für den Algorithmus waren:

- Als Eingabe ist ein Dreiecksnetz eines Bauteils gegeben[2].
- Als Ausgabe wird ein Vierecksnetz erzeugt.
- Als Zwischenschritt sollte eine Volumenbeschreibung des Bauteils erzeugt werden.
- Es sollte darauf geachtet werden, dass spezifische Merkmale des Netzes erhalten bleiben. Zu den wichtigsten Merkmalen gehören Randkurven (sowohl Außenrand, als auch Ränder von Löchern), scharfe Kanten und Ecken.
- Das Vierecksnetz sollte aus möglichst regelmäßigen und gleich großen Vierecken bestehen. Der Anteil an Dreiecken sollte möglichst klein sein.

1.2 Ansatz

Der entwickelte und in dieser Arbeit beschriebene Algorithmus zum Neuvernetzen erzeugt zuerst aus einem Dreiecksnetz eine Volumenbeschreibung des Bauteils. Die einzelnen Elemente einer Volumenbeschreibung, genannt Voxel, liegen in den Eckpunkten eines dreidimensionalen uniformen[3] Gitters. Pro Voxel werden Daten gespeichert, die die Form des Bauteils beschreiben. Aus der Volumenbeschreibung wird zuerst eine Isofläche extrahiert, aus der dann ein Vierecksnetz erzeugt wird.

1.3 Aufbau der Arbeit

Das zweite Kapitel gibt eine Übersicht über vorhandene Literatur aus den Bereichen Neuvernetzung, Voxelisierung und Extraktion der Isofläche. Die verwendeten Algorithmen werden im Kapitel 3 erklärt. In Kapitel 4 werden Zusammenhänge zwischen diesen Algorithmen dargestellt. Dabei werden auch die Änderungen erläutert, die notwendig waren, um ein Vierecksnetz zu erhalten. Die Erhaltung der Merkmale und ihre Bestimmung ist in Kapitel 5 dargestellt. Die Möglichkeiten der Optimierung des gewonnenen Vierecksnetzes werden in Kapitel 6 beschrieben. Die Ergebnisse werden in Kapitel 7 anhand einiger Bilder präsentiert. Außerdem wird die Eignung des Algorithmus für die FE-Methoden diskutiert. Schließlich rundet Kapitel 8 die Arbeit mit einem Fazit ab.

[2] Da man aus jedem Polygonnetz durch das Aufspalten der Polygone relativ einfach ein Dreiecksnetz erzeugen kann, ist die Beschränkung auf Dreiecksnetze bei der Eingabe keine wirkliche Einschränkung des Algorithmus

[3] uniformes Gitter = regelmäßiges Gitter mit gleichen Gitterabständen

2 Verwandte Arbeiten

Zum direkten Neuvernetzen von Dreiecksnetzen (ohne den Zwischenschritt der Volumenbeschreibung) existiert eine Reihe von Ansätzen. Die Algorithmen in [SAG03] und in [SG03] erzeugen wahlweise ein Dreiecksnetz mit gleichgroßen Dreiecken oder Dreiecken deren Größe an die Oberflächenkrümmung angepasst werden kann. Dabei werden lokale Operationen auf das gegebene Dreiecksnetz angewendet. In [BK04] wird das vorliegende Netz in eine Basisfläche und eine Detailinformation zerlegt. Nach dem Neuvernetzen der Basisfläche wird die Detailinformation auf die neue Basisfläche angewendet. In [ACSD+03] und [MK04] wird ein anisotropes Vierecksnetz aus einem Dreiecksnetz erstellt.

Da die Vorgaben für die vorliegende Arbeit eine Volumenbeschreibung als Zwischenschritt enthalten haben, sollte die Fläche aus der Volumenbeschreibung extrahiert werden. Aufgrund der gewählten Voxelisierung ist die gesuchte Fläche die Isofläche zum Isowert 0. Es existieren zahlreiche Verfahren, die Isofläche als Dreiecksnetz aus Volumendaten extrahieren. Viele sind nach dem Prinzip des *Marching Cubes*(MC) Algorithmus [LC87] aufgebaut: [MSS94], [Nie04], [KBSS01], [Che95].

In der Arbeit [MSS94] werden die Schnittpunkte entlang einer Kante nicht interpoliert, sondern immer in der Mitte angenommen. Dadurch kann die Ausrichtung der Dreiecke nicht beliebig ausfallen, sondern nur bestimmte Richtungen annehmen. Nach der Extraktion werden benachbarte Dreiecke mit der gleichen Ausrichtung zu größeren Polygonen zusammengefaßt und neu trianguliert.

In [Nie04] wird ein duales Netz dazu benutzt, die geometrische Eigenschaften der geschlossenen Marching Cubes Flächen iterativ zu verbessern.

Die Erhaltung der scharfen Merkmale ist ein bekanntes Problem des klassischen MC Algorithmus. Die Arbeit [KBSS01] beschäftigt sich ausführlich damit und stellt einen modifizierten MC-Algorithmus vor.

Ein anderes Problem des MC-Algorithmus ist die Mehrdeutigkeit einiger Fälle, die dazu führen kann, dass eine lückenhafte Isofläche ensteht. Eine der letzten Arbeiten, die sich ausführlich mit der Topologie bei Marching Cubes beschäftigt, ist [LB03].

Neben dem klassischen Marching Cubes Algorithmus und seinen Abwandlungen gibt es auch weitere Algorithmen zur Extraktion der Isofläche. Der Vorteil dieser Algorithmen gegenüber Marching Cubes liegt meistens an der geringeren Anzahl der erzeugten Dreiecke und an deren regelmäßigerer Form. Beispiele dafür sind z.B. der *Skeleton Climbing*[1] Ansatz in [PWH98] oder ein hierarchischer Ansatz beschrieben in [LHMG02]. Beim letzteren wird zuerst eine grob aufgelöste Fläche extrahiert, die dann iterativ verfeinert wird. In [WDSB00] wird beschrieben, wie ein semi-reguläres Dreiecksnetz aus Distanzfeldern gewonnen werden kann. Ein weiteres Verfahren welches ein adaptives Dreiecksnetz aus nicht-uniformen Volumendaten erzeugt ist das in [SW04] beschriebene *Primal Contouring*.

Eine Übersicht der Verfahren zur Generierung von Vierecksnetzen und Sechsecknetzen aus Volumendaten wird in [ZB05] dargestellt. Ein Verfahren, genannt *Dual Contouring*, basierend

[1] bedeutet in etwa „klettern am Gerüst“. Zuerst werden Vertices, dann Kanten und dann die Polygone erzeugt

auf dem Prinzip von Marching Cubes und *SurfaceNets*[2] zur Erzeugung von Vierecksnetzen aus speziell aufbereiteten Volumendaten, findet man in [JLSW02]. Weitere Verfahren, die aus Volumendaten Vierecksnetze erzeugen, sind z.B. *LayTracks* [QRPB00] und *20 Brick Meshing* [Dho99].

[2] SurfaceNets werden beschrieben in [Gib98]

3 Verwendete Algorithmen

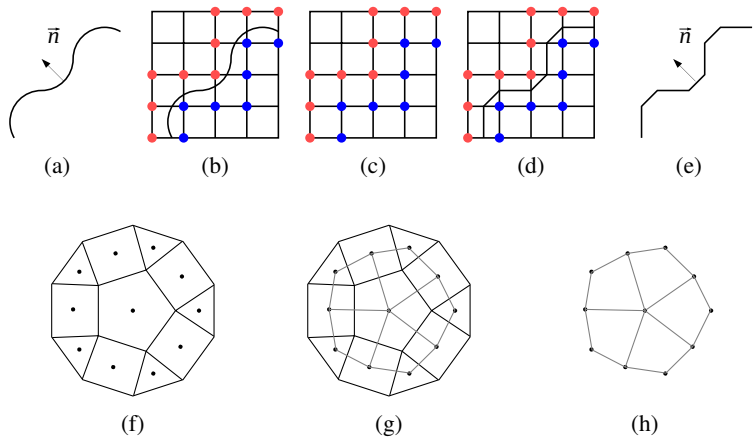


Abbildung 3.1: Der gesamte Ablauf der Rekonstruktion einer Fläche über ihre Volumendarstellung (symbolisch in 2D dargestellt).

Der Ablauf des Neuvernetzungsalgorithmus lässt sich in 4 Schritte unterteilen:

1. Erstellen einer Volumenbeschreibung aus dem gegebenem Dreiecksnetz. (Abbildung 3.1(a)-(c)) Mein Algorithmus verwendet dazu ein Distanzfeld. Bei einem Distanzfeld wird im Voxel der Abstand zur gegebenen Fläche gespeichert. Dieser Vorgang wird in Anlehnung an das Wort Voxel als Voxelisieren bezeichnet (siehe Kapitel 3.1). In Abbildung 3.1(c) ist ein Volumengitter (zur Vereinfachung auf 2D reduziert) symbolisch dargestellt. In den roten und blauen Voxeln sind Abstände zu der Fläche (Abbildung 3.1(a)) gespeichert. Dabei liegen die roten Voxel oberhalb der Fläche, die blauen Voxel unterhalb[1].

2. Extrahieren der Fläche aus der Volumenbeschreibung (Abbildung 3.1(d)). Die gesuchte Fläche verläuft zwischen den blauen und roten Voxeln (Abbildung 3.1(d)). Die Extraktion der Fläche wird in Kapitel 3.2 detailliert beschrieben.

3. Umwandeln der Fläche aus dem vorhergehenden Schritt (gespeichert als ein Polygonnetz) in ein Vierecksnetz. Dazu wird aus dem Polygonnetz ein duales Netz erzeugt (Abbildung 3.1(f)-(h)) welches fast ausschließlich aus Vierecken besteht (Kapitel 3.3). Damit das duale Netz einen hohen Anteil an Vierecken hat, musste der Marching Cubes Algorithmus entsprechend modifiziert werden (Kapitel 4).

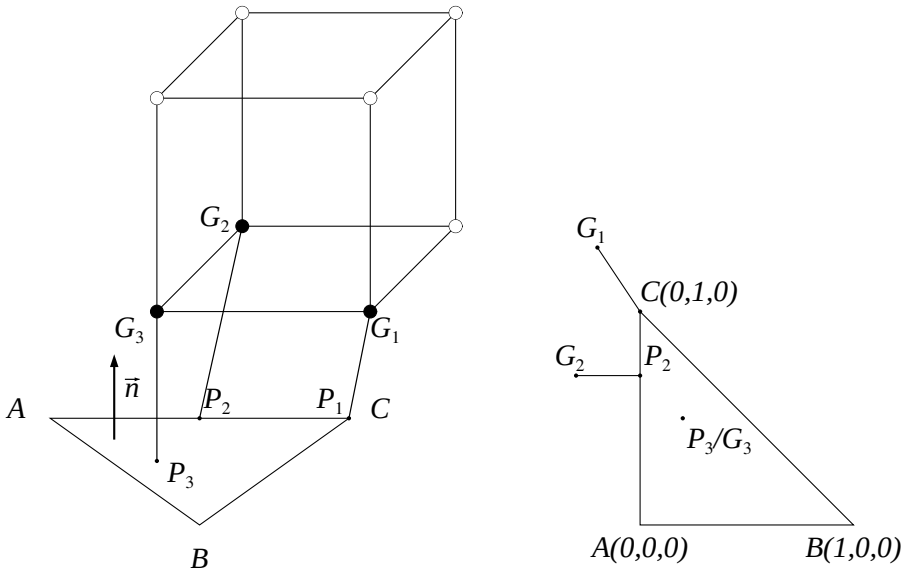
4. Verbesserung und nachträgliche Optimierung des Vierecksnetzes (siehe Kapiteln 5 und 6).

In folgenden Kapiteln 3.1, 3.2 und 3.3 werden verwendete Algorithmen unabhängig von einander detailliert erklärt.

[1] bestimmt durch den Normalenvektor $\vec{n}$

3.1 Voxelisierung

Bei der Voxelisierung wird aus der polygonalen Beschreibung einer geschlossenen oder einer berandeten Fläche eine volumenbasierte Beschreibung erzeugt. Dabei werden den Punkten eines uniformen dreidimenensionalen Gitters Werte zugeordnet, die möglichst präzise die Geometrie der gegebenen Fläche wiedergeben. Die Gitterpunkte mit den zugehörigen Daten werden im Weiteren als Voxel bezeichnet. Im vorliegenden Fall teilt das uniforme dreidimensionale Gitter den Raum in Würfeln mit Kantenlänge 1, an deren Ecken die Voxel liegen. In einem dreidimensionalen Gitter mit Abmessungen L, B, H gibt es demnach (L+1)×(B+1)×(H+1) Voxel und (L×B×H) Würfel[2]. Pro Voxel speichert man den kürzesten Abstand zur Fläche, das Vorzeichen sowie weitere Daten zur späteren Verbesserung der Ränder und scharfen Kanten. In diesem Kapitel wird zunächst der einfache Fall erläutert, bei dem außer dem Abstand mit Vorzeichen keine weiteren Daten im Voxel gespeichert werden.



(a) Allgemeiner 3D Fall

(b) 3D Fall transformiert zu einem einfachen 2D Fall. Der Normalenvektor des Dreiecks zeigt in Richtung der z-Achse

Abbildung 3.2: Kürzester Abstand von einem Punkt zu einem Dreieck

Sei P der Punkt auf der Fläche F mit minimalem Abstand d zum Gitterpunkt G. Und sei $\vec{n}$ der Normalenvektor der Fläche im Punkt P. Dann entscheidet das Vorzeichen des Skalarprodukts $\vec{n} \cdot \overrightarrow{GP}$ über das Vorzeichen des Abstandes des Voxel in G. Das Vorzeichen gibt also an, ob sich das Voxel auf der gleichen Seite der Fläche befindet, in die der Normalenvektor dieser Fläche zeigt, oder auf der gegenüberliegenden Seite. Die Bestimmung des Vorzeichens ist unabdingbar für die korrekte Extraktion der Isofläche zwischen positiven und negativen Abständen. Allerdings stellt das ein nicht triviales Problem dar. Der Grund dafür liegt in der Tatsache, dass bei einer

[2] gemeint sind jetzt und im Folgenden die Würfel im Volumen, deren Ecken in 8 benachbarten Voxel liegen

polygonalen Beschreibung der Fläche F die Normalen nicht für alle Punkte auf F eindeutig bestimmt sind.

Ist die Fläche durch ein Dreiecksnetz gegeben, dann können 3 Fälle auftreten, wie der kürzeste Abstand zwischen einem Dreieck und einem Voxel gemessen wird. Ausschlaggebend dafür ist die Lage des Punktes P (auf einem Dreieck ABC) der den kürzesten Abstand zum Voxel besitzt (siehe Abbildung 3.2(a)):

- *Fall 1*: Punkt P liegt in einer der Ecken A, B oder C
  In Abbildung 3.2(a) liegt Punkt P_1 in C
- *Fall 2*: Punkt P liegt auf einer der Kanten AB, BC, AC
  In Abbildung 3.2(a) liegt Punkt P_2 auf AC
- *Fall 3*: Punkt P liegt im Inneren des Dreiecks
  In Abbildung 3.2(a) liegt Punkt P_3 im Inneren des Dreiecks ABC

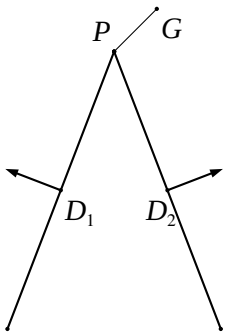


Abbildung 3.3: Dreiecke D_1 und D_2 in der Projektion, Punkt P hat den kleinsten Abstand zu G

In den Fällen 1 und 2 ist die Normale $\vec{n}$ der Fläche zunächst nicht eindeutig bestimmt, da hier mehrere Dreiecke mit i.d.R. unterschiedlichen Normalen aufeinander treffen. Bei der Voxelisierung eines einzelnen Dreiecks kann somit für Punkte wie P_1 bzw. P_2 das korrekte Vorzeichen des Abstandes nicht bestimmt werden.

Es hat sich herausgestellt, dass die Hinzunahme der Normale des Dreiecks zur Bestimmung des Vorzeichens zur Fehlentscheidungen führen kann und ist somit nicht anwendbar. Betrachtet man in der Abbildung 3.3 die Dreiecke D_1 und D_2 einzeln, so stellt man fest, dass der Punkt G einmal oberhalb und einmal unterhalb der jeweiligen Ebenen liegt. Der Abstand zu den beiden Dreiecken ist jedoch gleich. Durch das alleinige Betrachten der Normalen der Dreiecke und der Abstände kann das Vorzeichen nicht entschieden werden. Andere Ansätze zum Generieren von vorzeichenbehafteten Distanzfelder, wie z.B. Abspeichern des Winkels zwischen dem Verbindungsvektor und dem Normalenvektor des Dreiecks oder verschiedene Prioritäten der o.g. drei Fälle führten in durchgeführten Tests nicht zum gewünschten Erfolg (vgl. auch [BA02]).

Das Problem wurde mit dem Einführen der *Pseudo-Normalen* gelöst. Dabei wird für *jeden* Punkt P auf dem Dreiecksnetz eine s.g. *Pseudo-Normale* definiert, mit der dann das Vorzeichen

bestimmt wird. Laut [BA02] garantiert die Wahl der *nach Winkel gewichteten Normale* als Pseudo-Normale immer das korrekte Vorzeichen. Die Definition der *nach Winkel gewichteten Normale* $\vec{n}_P$ im Punkt P ist wie folgt:

$$\vec{n}_P = \sum_i \alpha_i \vec{n}_i \tag{3.1}$$

Dabei ist α_i der Winkel in P eines Dreiecks i und $\vec{n_i}$ der Normalenvektor des gleichen Dreiecks. Da später nur das Vorzeichen des Skalarprodukts mit $\vec{n_P}$ von Interesse ist, kann die Pseudo-Normale als gewichtete unnormalisierte Summe der Normalen berechnet werden. Das i geht über alle Dreiecke, die P enthalten. Liegt P in einer der Ecken, so ist α_i der dazugehörige Winkel im Dreieck. Liegt P auf der Kante, dann ist der Winkel gleich π. Schließlich für den Fall, dass P im Inneren des Dreiecks liegt, beträgt der Winkel 2π und die Pseudo-Normale ist gleich der Normale des Dreiecks.

Für die spätere Extraktion der Isofläche ist es nicht notwendig, für alle Voxel den minimalen Abstand zur Fläche zu berechnen. Es genügt für Voxel in einem ausreichend großen Abstand δ von der Fläche den minimalen Abstand zu bestimmen. Der Grenzwert δ sollte größer sein als $\sqrt{3}$, damit in einem Würfel alle Voxel gesetzt werden. Ist er größer als 2, dann werden u.U. unnötige Voxel gesetzt. Der optimale Wert für δ liegt also zwischen $\sqrt{3}$ und 2. Unter Berücksichtigung der Berechnungsgenauigkeit und Rundungsfehler sollte der Wert ein wenig größer sein als $\sqrt{3}$. In meiner Implementierung verwendete ich $\delta = 1.9$ um Rundungsfehler sicher auszuschließen.

Für jedes Dreieck wird zuerst eine s.g. Bounding Box[3] berechnet. In dieser Bounding Box wird für jedes Voxel der kleinste Abstand bestimmt. Für solche Dreiecke, deren Ebene exakt auf den Gitterpunkten liegt, muss die Bounding Box in der jeweiligen Dimension erweitert werden:

// Berechnen der Bounding Box des Dreiecks ABC
for i = 0..2 **do**
  $bbox_{min}[i] \leftarrow floor(min(A[i], min(B[i], C[i])))$
  $bbox_{max}[i] \leftarrow ceil(max(A[i], max(B[i], C[i])))$
  if $(bbox_{min}[i] == bbox_{max}[i])$ **then**
    // Erweitern
    $dec(bbox_{min}[i])$
    $inc(bbox_{max}[i])$
  end if
end for

Die Erweiterung ist notwendig, damit die spätere Extraktion der Isofläche mit Marching Cubes Algorithmus fehlerfrei funktioniert. Dieser Algorithmus wird im nächsten Kapitel erklärt.

Die Berechnung des minimalen Abstandes zum Dreieck erfolgt nach dem Ansatz von [Jon96]. Dabei wird für jedes Dreieck eine Matrix berechnet, die den 3D Fall zu einem einfachen 2D Fall transformiert (siehe Abbildung 3.2). Bei einem Dreieck ABC wird A nach $A' = (0, 0, 0)$, B nach

[3] Kleinster quaderförmiger Ausschnitt des Volumens, der das Dreieck komplett enthält

$B' = (1,0,0)$, C nach $C' = (1,0,0)$ und $\vec{n}_{ABC}$ nach (0,0,1) transformiert. Anders ausgedrückt bilden $\overrightarrow{AB}$, $\overrightarrow{AC}$ und $\vec{n}_{ABC}$ das Koordinatensystem des Dreiecks. Die zu testenden Voxel werden mit Hilfe der beschriebenen Matrix in das Koordinatensystem des Dreiecks überführt. Die transformierende Matrix wird wie folgt berechnet. Man nehme zuerst eine Matrix M, die als Spalten die Vektoren $\overrightarrow{AB}$, $\overrightarrow{AC}$ und $n_{\vec{ABC}}$ hat und berechne die inverse M^{-1} dazu[4]. Mit Hilfe von M, M^{-1} und einfachen Translationen bestimmt man dann den Punkt P mit dem kürzesten Abstand zum Voxel V.

// Bestimme Punkt P auf ABC mit kürzestem Abstand zu Voxel V
$M \leftarrow [\overrightarrow{AB}^T, \overrightarrow{AC}^T, \vec{n}_{ABC}^T]$
$M^{-1} \leftarrow$inverse(M)
$\overrightarrow{A'V'} \leftarrow M^{-1} * \overrightarrow{AV}$
Für V' wird Punkt P' im Koordinatensystem des Dreiecks so bestimmt,
dass P' den kleinsten Abstand zu V' hat.
$P \leftarrow (M * \overrightarrow{A'P'}) + A$

Im Initialzustand vor dem Voxelisieren haben alle Voxel als Abstand einen „∞“ Wert[5]. Würfel die nach dem Voxelisieren immer noch Voxel mit „∞“ Wert haben, werden bei der Flächenextraktion ignoriert.

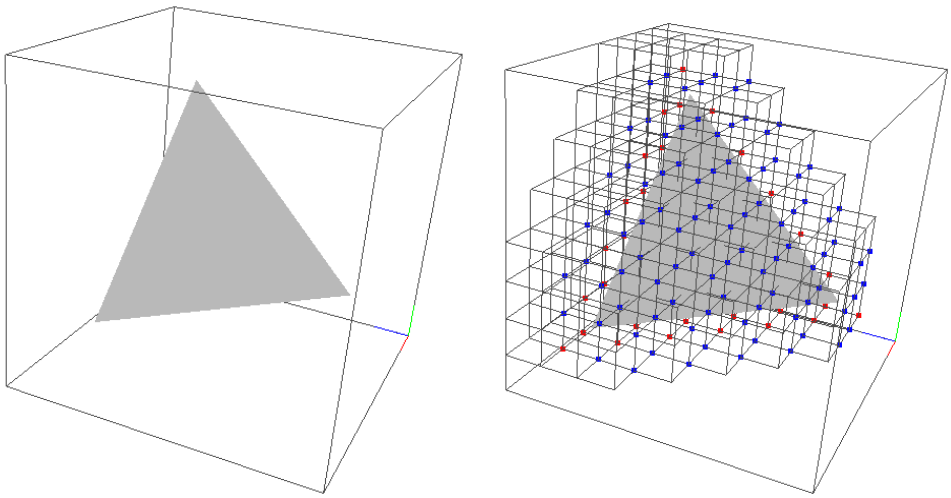
Der Voxelisierungsprozess sieht wie folgt aus:

---

for all Dreieck ABC in Mesh **do**
  berechne die Transformationsmatrix M^{-1}
  berechne Bounding Box (siehe oben)
  for all Voxel V in Bounding Box **do**
    $P \leftarrow$ Punkt auf ABC, der V am nächsten liegt (mit Hilfe M^{-1})
    $\vec{r} \leftarrow \overrightarrow{PV}$
    $d \leftarrow |\vec{r}|$
    $sign \leftarrow \text{sign}(\vec{r} \cdot \vec{n_P})$
    if $((d < \delta)$ **and** $(d < |V.d|))$ **then**
      $V.d \leftarrow sign * d$
    end if
  end for
end for

In der Abbildung 3.4 ist die Voxelisierung eines einfachen Dreiecks dargestellt. Die gesetzten Voxel sind symbolisch in Blau und Rot eingefärbt. Blaue Voxel haben einen Abstand < 0, rote Voxel ≥ 0. Das Volumengitter ist nur im Bereich der gesetzten Voxel gezeichnet.

[4] Für nicht degenerierte Dreiecke lässt sich M^{-1} immer berechnen
[5] Man nehme als „∞“ Wert eine Konstante die größer ist, als der gewählte Grenzwert δ



(a) Einfache Fläche

(b) Die Fläche und ihre Volumenbeschreibung. Blaue Voxel haben einen Abstand < 0, rote Voxel ≥ 0

Abbildung 3.4: Voxelisierung einer einfachen Fläche im Volumen $9 \times 9 \times 9$

3.2 Extraktion der Isofläche

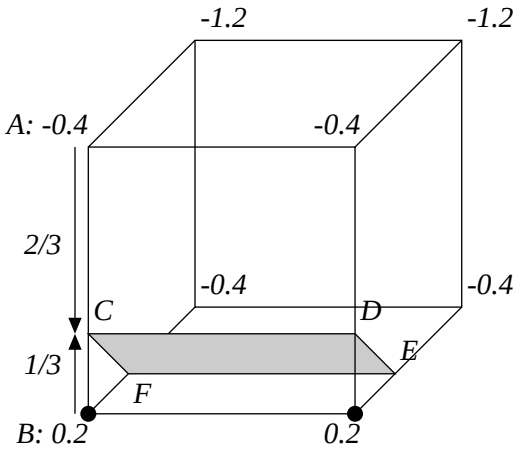


Abbildung 3.5: Fragment der Isofläche in einem Würfel. In den Ecken sind Abstände der Voxel mit Vorzeichen eingetragen.

Wie bereits angesprochen, wird aus der Volumenbeschreibung des Bauteils ein Polygonnetz extrahiert, welches die Form des Bauteils beschreibt. Da die Volumenbeschreibung die Abstände zu der gesuchten Fläche enthält, wird eine Fläche gesucht, die durch die Nullpunkte im Volumen verläuft. Wenn die rekonstruierte Fläche durch die Nullpunkte verläuft, hat sie in diesen Punkten Abstand 0 zu der ursprünglichen Fläche – somit wird die ursprüngliche Fläche rekonstruiert. Eine Fläche, die einen konstanten Wert repräsentiert (wie es hier der Fall ist), nennt man Isofläche und den dazugehörigen Wert den Isowert. Im vorliegenden Fall ist die extrahierte Fläche die Isofläche zum Isowert 0.

In Abbildung 3.5 sieht man ein Fragment der Isofläche in einem Würfel des Volumens. Die Abstände der Voxel A und B haben unterschiedliche Vorzeichen. Das bedeutet, dass zwischen ihnen der Nullpunkt sein muss. Der Nullpunkt C wird dann entlang der Kante AB durch lineare Interpolation bestimmt: Das Teilungsverhältnis AC/AB erhält man mit 0.2/(0.2-(-0.4)) = 1/3. Analog dazu werden andere Nullpunkte D, E und F bestimmt. Die Isofläche verläuft also durch die Punkte C, D, E und F und kann in diesem Würfel durch das Polygon $CDEF$ repräsentiert werden. Man kann also die Fragemente der Isofläche Würfel für Würfel extrahieren und erhält somit die gesamte Isofläche. Der Algorithmus dazu wurde erstmalig in [LC87] beschrieben und heißt *Marching Cubes* Algorithmus. In der vorliegenden Arbeit wird eine Variante des Marching Cubes Algorithmus verwendet, die im Folgenden erläutert wird.

Der originale Marching Cubes Algorithmus, wie er in [LC87] beschrieben wird, extrahiert aus Volumendaten eine Isofläche, die für einen gegebenen Wert α die Voxel mit Werten kleiner α von solchen mit Werten größer gleich α trennt. In unserem Fall wird aus dem Volumen die Isofläche mit $\alpha = 0$ extrahiert, da die Voxel den Abstand zu der Fläche enthalten, die man rekonstruieren möchte.

Der Marching Cubes Algorithmus betrachtet jeweils einen Würfel des Volumens, für welchen er dann ein Fragment der Isofläche generiert (siehe Abbildung 3.5). Jeder Ecke des Würfels kann man in Abhängigkeit vom Wert des dortigen Voxel (< 0 oder ≥ 0) ein Bit eines Konfigurationswortes zuordnen. Das Bit zeigt an, auf welcher Seite der zu extrahierenden Fläche sich das Voxel befindet. Da ein Würfel 8 Ecken hat, ergeben sich für das Konfigurationswort $2^8 = 256$ Kombinationen. Nimmt man Kombinationen heraus, die unter Drehungen, Spiegelungen oder Invertierung identisch sind, bleiben 15 unterschiedliche Fälle übrig (eng.: Marching Cubes Cases), die in Abbildung 3.6 aufgeführt sind (vgl. [LC87]).

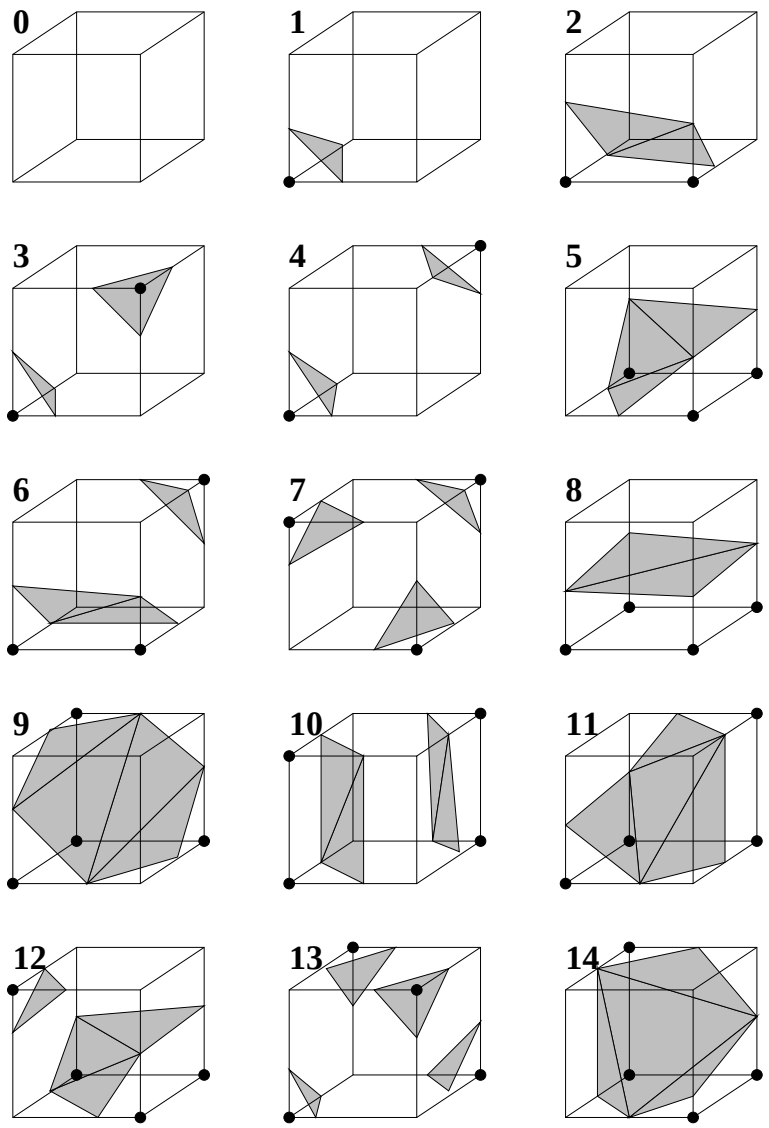


Abbildung 3.6: Ursprüngliche Marching Cubes Tabelle nach [LC87](Darstellung aus [Che95])

Allerdings sind manche Fälle der ursprünglichen Marching Cubes Tabelle nicht eindeutig[6]. Das liegt daran, dass bei bestimmten Konfigurationen die Lage der Isofläche zweideutig ist. In Abbildung 3.7 ist dazu eine Konfiguration einer Seite zu sehen und beide Möglichkeiten abgebildet, wie eine Isofläche dort verlaufen kann. Dabei scheinen in Abbildung 3.7(a) die mit Punkt markierten Voxel durch die Isofläche getrennt zu sein, die anderen zwei Voxel dagegen nicht. In Abbildung 3.7(b) verhält es sich genau anders herum. Außerdem kann die Isofläche im Inneren des Würfels so verlaufen, dass die gegenüberliegende Voxel nicht durch sie getrennt werden. Es können also zwei Arten von Mehrdeutigkeiten auftreten: Auf der Seite des Würfels (wie in Abbildung 3.7 dargestellt) und im Inneren des Würfels.

Die Abbildung 3.8 zeigt die Marching Cubes Fälle 3 und 6 die im ungünstigen Fall eine kleine Lücke auf der gemeinsamen Seite der Würfel erzeugen. Die gemeinsame Seite weist die zweideutige Konfiguration auf, wie sie in Abbildung 3.7 dargestellt wurde. Ich möchte an dieser Stelle auf [Che95] und [LB03] verweisen, wo diese Problematik ausführlich diskutiert wird.

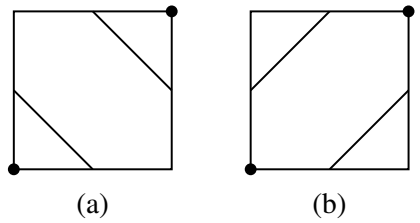


Abbildung 3.7: Zweideutigkeit an einer Seite eines Würfels bei Marching Cubes

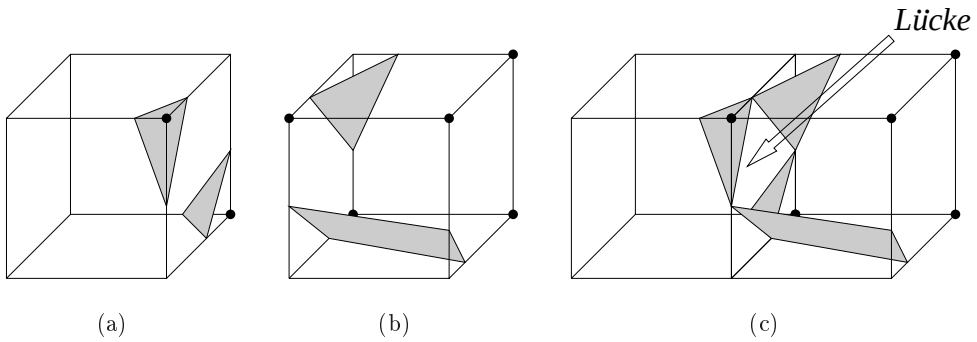


Abbildung 3.8: Beispiel von zwei benachbarten nicht eindeutigen Konfigurationen. Es entsteht u.U. eine kleine Lücke auf der gemeinsamen Seite

Eine der möglichen Lösungen für Mehrdeutigkeiten ist z.B. die Erweiterung des Marching Cubes Algorithmus wie sie in [Che95] von Chernyaev vorgeschlagen wird. Seine Tabelle für Fragmente der Isofläche ist in Abbildung 3.9 dargestellt. Beispielsweise ist der Fall 3 der ursprünglichen Tabelle nicht eindeutig und wird bei Chernyaev durch Fälle 3.1 und 3.2 ersetzt. Dabei wird die angesprochene Mehrdeutigkeit im Inneren des Würfels durch zwei Fälle abgedeckt. Die markierten Voxel sind im Fall 3.1 durch die Isofläche getrennt, im Fall 3.2 dagegen nicht. Der Fall 13 der originalen Tabelle wird sogar in 6 Fälle (13.1 bis 13.5.2) unterteilt.

[6] Das Problem wird im Englischen als „Marching Cubes Ambiguous Cases“ bezeichnet

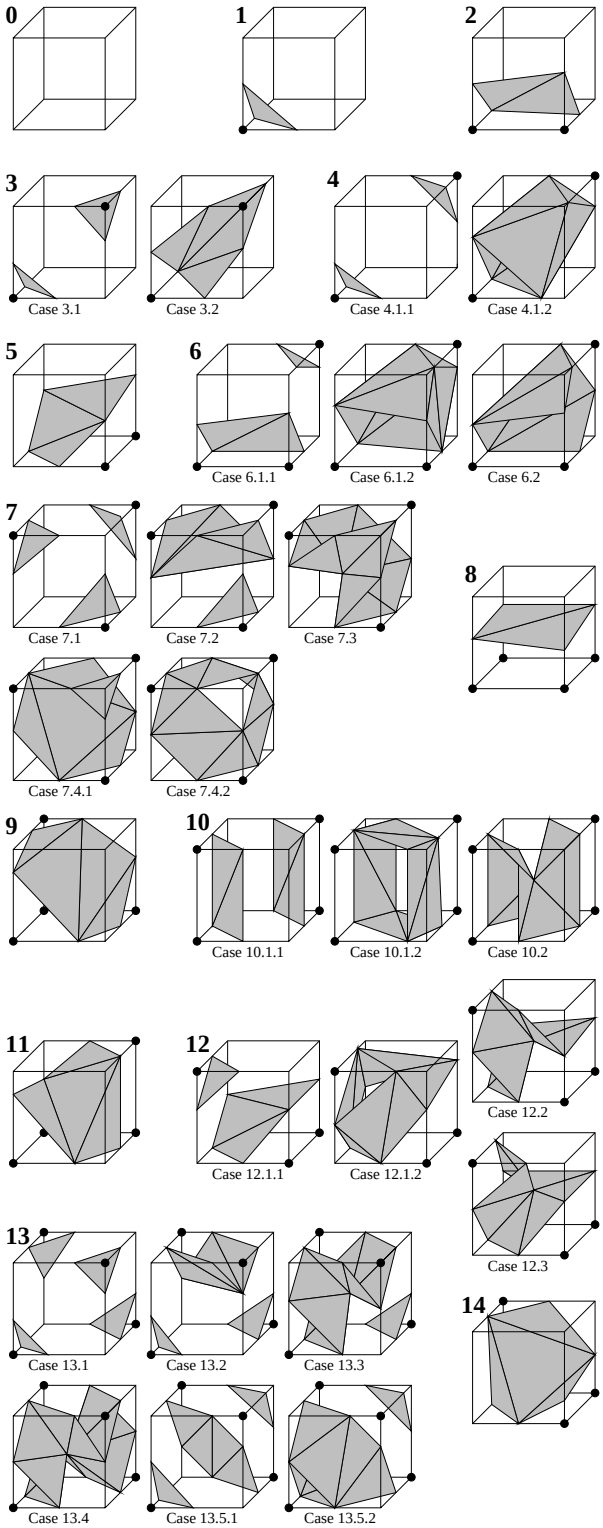


Abbildung 3.9: Marching Cubes 33 Tabelle von Chernyaev (entnommen aus [Che95])

Die Tabelle von Chernyaev (siehe Abbildung 3.9) enthält 33 mögliche Fälle. Die Fälle 12.2 und 12.3 wurden aber von ihm selbst als Spiegelfälle erkannt. Die Fälle 14 und 11 sind ebenfalls Spiegelfälle, so dass 31 topologisch unterschiedliche Fälle übrig bleiben (vgl. [LB03]).

Der in [Che95] von Chernyaev beschriebener Marching Cubes 33 Algorithmus wurde von Lewiner u.a. (vgl. [LLVT03]) untersucht, erweitert und implementiert. Ich habe diese Implementierung für meine Zwecke übernommen und erweitert.

Um geschlossene und umrandete Flächen extrahieren zu können, wurde der Algorithmus dahingehend modifiziert, dass die Würfel mit mindestens einem „∞“ Voxel ignoriert werden. Allerdings weist die Geometrie der Ränder bei dieser Methode Abweichungen in der Größenordnung des Gitterabstandes des Volumens auf. Es wird später gezeigt, wie man das mit Hilfe von erweiterten Volumendaten und dualen Techniken verbessern kann.

Eine weitere Änderung war die Möglichkeit außer Dreiecken andere Polygone für die Fragmente in der MC-Tabelle zu verwenden. In Kapitel 4 wird gezeigt, wie man das ausnutzen kann, um Vierecksnetz zu erzeugen.

3.3 Duales Netz

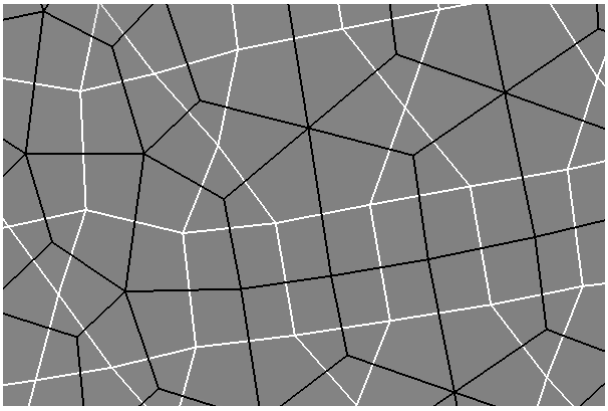


Abbildung 3.10: Duale Graphen. Polygone mit schwarzen Konturen sind dual zu den Polygonen mit weißen Konturen.

Bei der Erzeugung eines Vierecksnetzes einer Fläche aus der Volumenbeschreibung ist die Extraktion der Fläche mit dem Marching Cubes (MC) Algorithmus nur ein Zwischenschritt. Im weiteren Schritt wird aus der MC-Fläche, die verschiedene Polygone enthält, ein Vierecksnetz erzeugt. Es wird gezeigt, dass ein duales Netz zum Netz der MC-Fläche fast ausschließlich aus viereckigen Polygonen besteht. Im Folgenden wird erklärt, wie ein duales Netz aufgebaut wird.

Die Umwandlung zum dualen Netz erfolgt durch die Anwendung eines Dual-Operators. Der allgemeine Dual-Operator auf Graphen (siehe auch [Wei05]) erzeugt aus einem Netz N ein duales Netz N^d indem er jedem Polygon p_i aus N ein Vertex ν_i^d in N^d zuordnet. Für jede Kante, die Polygon p_i mit einem anderen Polygon p_k gemeinsam hat, gibt es eine Kante in N^d zwischen Vertex ν_i^d und ν_k^d. Anschaulich kann man sich den dualen Vertex ν_i^d als einen Vertex innerhalb des Polygons p_i vorstellen. Die dualen Kanten schneiden dabei die Kanten des ursprünglichen Netzes. In Abbildung 3.10 sind die schwarz umrandeten Polygone dual zu den weiß umrandeten Polygonen.

Für Polygon p sei nun $grad(p)$ definiert als die Anzahl der Ecken in p. Und für einen Vertex ν soll $grad(\nu)$ die Anzahl der Polygone sein, die den Vertex ν enthalten.
Es gilt demnach für alle Polygone p_j^d in F^d:

$$grad(p_j^d) = grad(\nu_j) \tag{3.2}$$

Im Unterschied zum allgemeinen Dual-Operator für beliebige Graphen werden in unserem Fall den Vertices mit $grad(\nu_j) \leq 2$ keine dualen Polygone zugeordnet, da aufgrund der Gleichung 3.2 diese Polygone keine graphische Primitive wären. Man beachte, dass für geschlossene Flächen mit Polygonen deren Winkel kleiner π sind, die Bedingung $grad(\nu_j) > 2$ für alle Vertices ν_j erfüllt ist, so dass dort *jedem* Vertex ein duales Polygon zugeordnet werden kann. Bei nicht geschlossenen Flächen haben Vertices am Rand meistens $grad(\nu_j) \leq 2$. Diesen Vertices können direkt keine dualen Polygone zugeordnet werden, so dass das duale Vierecksnetz an äußeren Rändern „schrumpft“. Analog dazu weiten sich die Randkurven der Löcher aus. Dieser Effekt

gleicht in geringerem Maße den umgekehrten Fall bei Marching Cubes Algorithmus aus: Beim Voxelisieren werden sämtliche Voxel in der Umgebung des Rands gesetzt, was bei der Isofläche dazu führt, dass diese sich an den Rändern geringfügig ausweitet.

Diese Tatsache spielt aber im Weiteren keine große Rolle, da für die Rekonstruktion der Ränder ohnehin besondere Maßnahmen getroffen werden. Diese Maßnahmen werden in Kapitel 5 ausführlich erläutert. In Kapitel 5 wird ebenfalls erklärt, wie in meiner Implementierung die Koordinaten des dualen Vertex ν_i^d zu einem Polygon p_i berechnet werden.

4 Verknüpfung der verwendeten Algorithmen

In diesem Kapitel wird erläutert, wie durch die Verknüpfung der in Kapitel 3 beschriebenen Algorithmen ein Vierecksnetz entsteht und welche Änderungen dazu notwendig sind.

4.1 Umrechnung der Koordinaten

Wie in Kapitel 3.1 geschrieben wurde, wird beim Volumen immer ein Gitterabstand von 1.0 angenommen. Bei einer vordefinierten Anzahl der Voxel ergibt sich daraus eine bestimmte Größe der Volumens. Da die gegebene Dreiecksnetze ihre eigene Weltkoordinaten besitzen, ist es für die Berechnung notwendig ihre Koordinaten in das Koordinatensystem des Volumens umzurechnen. Damit können die Berechnungen einfacher und effizienter durchgeführt werden. Dementsprechend ist es auch notwendig die Koordinaten des dualen Netzes wieder in die Weltkoordinaten zu überführen.

Die dazu notwendige Transformation wird durch eine uniforme Skalierung[1] und einen dreidimensionalen Verschiebungsvektor beschrieben. Auf eine Rotation des Netzes wurde verzichtet.

Wegen der einheitlichen Verarbeitung wird eine Transformation bei allen Polygonnetzen und dem Volumen gespeichert. Die jeweilige Transformation ist die Abbildung der tatsächlich gespeicherten Koordinaten in die Weltkoordinaten. Bei der Umrechnung der gespeicherten Koordinaten eines Netzes wird die dazugehörige Transformation des Netzes so angepasst, dass ihre Anwendung auf die neuen Koordinaten die Weltkoordinaten ergibt. Dazu wird die Umrechnungstransformation der Koordinaten und auf die gespeicherte Transformation invers angewendet.

In der Implementierung kann der Benutzer die Anzahl der Voxel im Volumen angeben, die er für die Neuvernetzung benutzen möchte. Das Volumen wird so dimensioniert, dass es die Seitenverhältnisse des Dreiecksnetzes hat und die angegebene Anzahl der Voxel maximal annähert.

Beim Importieren aus einer Datei werden zunächst Weltkoordinaten gespeichert und die Transformationsabbildung auf Identitätsabbildung gesetzt[2]. Vor dem Voxelisieren werden Koordinaten des Eingangsnetzes so umgerechnet, dass das Netz komplett vom Volumen umschlossen wird. Die beim Dreiecksnetz gespeicherte Transformation wird wie oben beschrieben angepasst. Nach dem Voxelisieren wird diese Transformation dem Volumen zugewiesen.

Das MC-Polygonnetz und das duale Netz übernehmen vom Volumen die gespeicherte Transformation. Beim Exportieren des dualen Netzes wird schließlich die Transformation auf die Koordinaten angewendet. Damit liegt das exportierte Polygonnetz wieder in Weltkoordinaten vor.

[1] Uniforme Skalierung: Gleiche Skalierungsfaktor für alle Dimensionen. Bei der uniformen Skalierung bleibt der Normalenvektor erhalten

[2] Identitätsabbildung ist in diesem Fall Skalierung = 1 und Verschiebungsvektor = (0,0,0)

4.2 Steigerung des Vierecksanteils

Eines der verfolgten Ziele dieser Arbeit war es, im Ausgangsnetz möglichst viele Vierecke zu erhalten. Dazu wird aus dem Netz der Isofläche (die mit MC Algorithmus gewonnen wurde) ein duales Netz konstruiert. Um den Anteil an Vierecken (Polygone mit Grad 4) zu steigern und den Anteil an Polygonen mit anderen Gradzahlen zu verringern, musste der MC-Algorithmus modifiziert werden.

Laut Gleichung 3.2 sollten dazu Vertices des MC-Polygonnetzes[3] den Grad 4 haben. Wir halten also als Bedingung 4.1 fest:

> *Die Vertices des MC-Netzes sollten den Grad 4 haben, damit die dazu dualen Polygone viereckig sind.* (Bedingung 4.1)

Wir betrachten nun den Vertex ν in Abbildung 4.1. Der Vertex ν hat im MC-Netz (im Bild grau eingefärbt) den Grad 4, da er in 4 Polygonen enthalten ist. Dem Vertex ν wird im dualen Netz das Polygon $Q_1Q_2Q_4Q_3$ zugeordnet, welches ebenfalls den Grad 4 hat. Wir können außerdem festellen, dass die meisten Vertices eines MC-Netzes sich auf einer Kante befinden, die 4 Würfel des Volumens gemeinsam haben (siehe Vertex ν in Abbildung 4.1). Der Grad von einem Vertex ist also in den meisten Fällen die Summe der Gradzahlen des gleichen Vertex in 4 benachbarten Volumenwürfeln[4]. Daraus folgt für die Vertices, die sich auf den Kanten der Würfel in einer MC-Tabelle befinden:

> *Die Vertices in der MC-Tabelle, die auf den Kanten eines Würfels liegen, sollten den Grad 1 haben, damit die dazu dualen Polygone viereckig sind.* (Bedingung 4.2)

Mit anderen Worten sollen Vertices, die sich an den Rändern der Würfel befinden in nur einem Polygon innerhalb des Würfels enthalten sein.

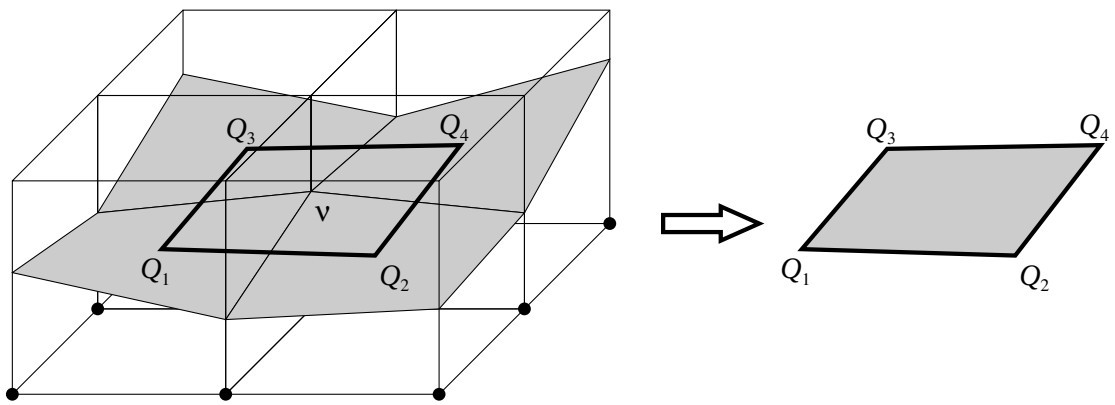


Abbildung 4.1: Duales Polygon $Q_1Q_2Q_4Q_3$ im Volumen

[3] Polygonales Netz aus der Extraktion mit Marching Cubes Algorithmus
[4] Ausnahmen entstehen z.B. durch Würfel an Volumengrenzen oder ignorierte Würfel mit „∞“ Werten

In der Arbeit von Nielson (vgl. [Nie04]) erfüllt die Tabelle für die Konstruktion eines s.g. MC-Patch-Netzes diese Bedingung für alle dort abgebildeten Fälle (siehe Abbildung 3 in [Nie04]). Sie enthält 23 Konfigurationen, die im wesentlichen von den Konfigurationen des Standard Marching Cubes Algorithmus abgeleitet wurden (siehe Abbildung 3.6).

Für die meisten Fälle der ursprünglichen Marching Cubes 33 Tabelle trifft die Bedingung 4.2 nicht zu. Die Vertices auf den Kanten haben eine höhere Gradzahl als 1. Das duale Netz würde in diesem Fall außer Vierecken und Dreiecken auch Fünfecke, Sechsecke usw. enthalten.

Im Rahmen vorliegender Arbeit habe ich versucht den in Kapitel 3.2 beschriebenen Marching Cubes 33 Algorithmus so anzupassen, dass möglichst viele Vertices die Bedingung 4.2 erfüllen.

MC-Fall	**Häufigkeit(%)**	**erfüllt Bed. 4.1 u. 4.2**
0	16,155	ja(*)
1	19,992	ja(*)
2	25,704	ja
3.1	0,086	ja(*)
3.2	0,073	ja
4.1.1	0,031	ja(*)
4.1.2	0,006	nein
5	15,189	ja
6.1.1	0,087	ja
6.1.2	0,000	nein
6.2	0,029	nein
7.1	0,000	ja(*)
7.2/7.3	0,000	nein
7.4.1/7.4.2	0,000	nein
8	20,105	ja
9	2,257	ja
10.1.1	0,031	ja
10.1.2	0,000	nein
10.2	0,001	nein
11	0,121	ja
12.1.1	0,015	ja
12.1.2	0,000	nein
12.2/12.3	0,003	nein
13.1	0,000	ja(*)
13.2-13.5.2	0,000	nein
14	0,116	ja

Tabelle 4.1: Fälle des Marching Cubes Algorithmus, die nach der Änderung die Bedingungen 4.1 und 4.2 erfüllen. Die mit (*) markierten Fälle mussten nicht geändert werden, da sie die Bedingungen schon von vornherein erfüllten. Die Häufigkeit bezieht sich auf das „Happy Buddha“ Modell voxelisiert im Volumen $74 \times 180 \times 74$.

Die Tabelle 4.1 gibt eine Übersicht der geänderten Fälle. Dabei wurde die Implementierung von Lewiner u.a. ([LLVT03]) dahin gehend erweitert, dass statt Dreiecken beliebige Polygone im

MC-Netz gespeichert werden können. Dann wurden die Triangulierungen der häufigsten MC-Fälle durch polygonale Beschreibungen ersetzt, so dass diese Fälle die Bedingungen 4.1 und 4.2 erfüllen. Die Änderungen von ausgewählten Fällen sind in der Abbildung 4.2 dargestellt. Der MC-Fall 14 ist analog zu MC-Fall 11 geändert worden und ist nicht abgebildet.

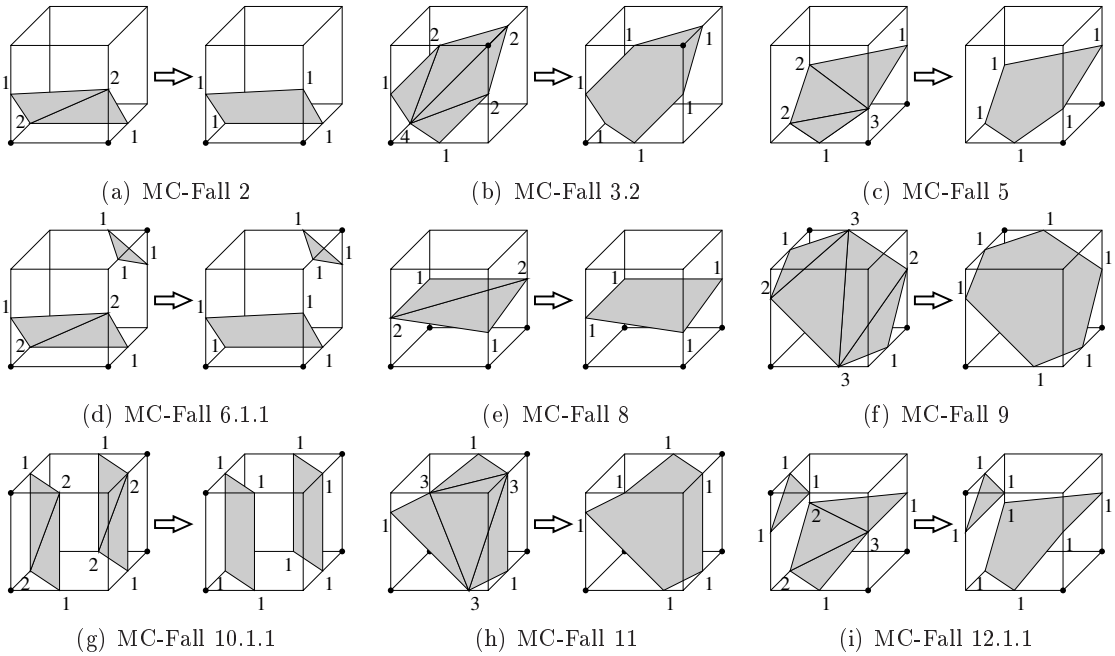

(a) MC-Fall 2
(b) MC-Fall 3.2
(c) MC-Fall 5
(d) MC-Fall 6.1.1
(e) MC-Fall 8
(f) MC-Fall 9
(g) MC-Fall 10.1.1
(h) MC-Fall 11
(i) MC-Fall 12.1.1

Abbildung 4.2: Die Gradzahl der Vertices vor und nach der Änderung der ausgewählten Fälle der Marching Cubes 33 Tabelle (vgl. Abbildung 3.9).

Beim MC-Netz entstehen nach der Änderung nicht planare Polygone, wenn die Vertices dieses Polygons nicht in einer Ebene liegen. Da das MC-Netz ein Zwischenschritt ist, stellt diese Tatsache kein Problem dar. Wichtig ist die Information über die Zugehörigkeit der einzelnen Vertices zu einem Polygon.

Nach den Änderungen der Tabelle lag der Anteil von Vierecken im duales Netz für die bekannten Datensätze (Happy Buddha, Stanford Bunny, Fandisk u.a.) über 99% und bei dem Bauteil aus Abbildung 1.1 bei über 98%. Bei den Modellen in Abbildungen 5.1(b) und 5.2(b) sieht man bereits einen hohen Anteil an Vierecken. Falls Polygone mit mehr als vier Ecken unerwünscht sind, muss man diese in einem anschließenden Durchgang in Dreiecke und Vierecke aufteilen. Es lassen sich nach der hier beschriebener Methode leider keine reinen Vierecksnetze erzeugen, da einige seltene Marching Cubes Fälle nicht vollständig ausgeschlossen werden können. Diese Fälle sind größtenteils dafür verantwortlich, dass der Vierecksanteil keine 100% erreicht. In Tests mit verschiedenen Volumenauflösungen ließ sich eine Tendenz erkennen, dass mit steigender Auflösung der Anteil an seltenen Marching Cubes Fällen weiter abnimmt. Damit konnten für die meisten Modelle Vierecksnetze mit einem Viereckanteil von über 99,9% erzeugt werden.

Die wenigen Polygone mit einer höheren Gradzahl als 4 müssen anschließend in Vierecke und Dreiecke zerlegt werden. Eine einfache Methode dazu ist die Aufspaltung der jeweiligen Polygone an ihren Mittelpunkten oder Abspaltung von Dreiecken bzw. Vierecken. Je nach Gradzahl und

den jeweiligen Winkeln kann man eine der Methoden anwenden oder beide kombinieren (siehe Abbildung 4.3).

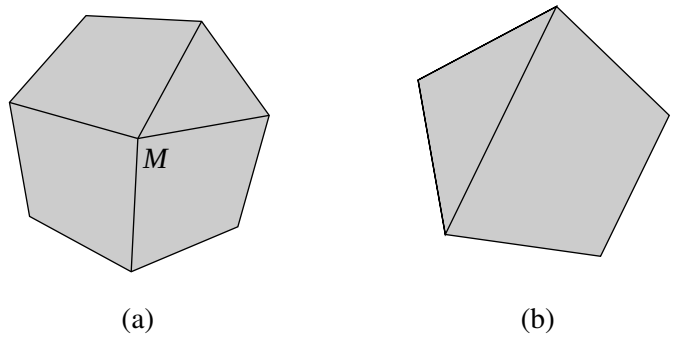
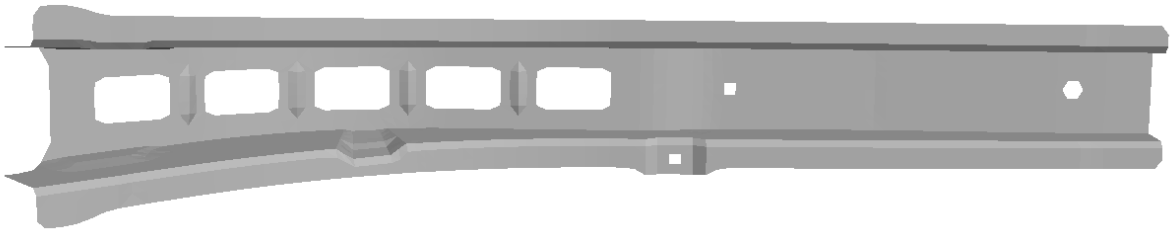


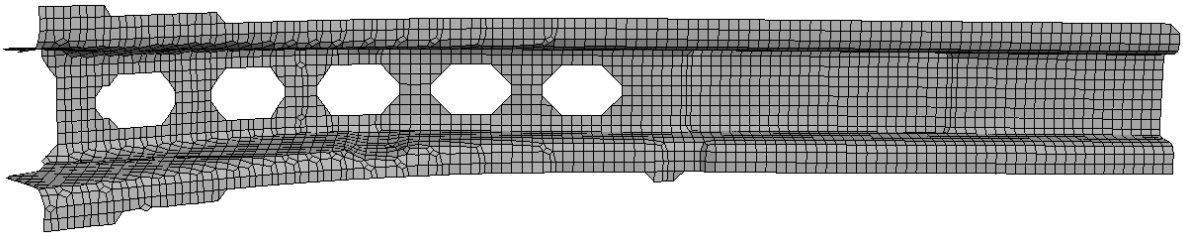
Abbildung 4.3: Aufspalten am Mittelpunkt M (a) oder Abspaltung (b)

5 Erhalten der spezifischen Merkmale

Die spezifischen Merkmale eines polygonalen Netzes sind geometrische Eigenschaften, die möglichst originalgetreu wiedergegeben werden sollen. Im vorliegenden Fall sind es beispielsweise Ränder, scharfe Kanten und Ecken. Beim Neuvernetzen der Bauteile oder Werkstücke hat das bis jetzt dargestellte Verfahren einen Nachteil. Sowohl die Kanten, als auch die Ränder werden nicht originalgetreu rekonstruiert. Die Abbildungen 5.1 und 5.2 zeigen einige Problemzonen auf. In der Abbildung 5.1 sind es die Ränder und die Kanten und in Abbildung 5.2 sind es die Kanten und die Ecken in welchen die Kanten zusammenlaufen. Man kann leicht feststellen, dass die Kanten und Ecken geglättet werden, während die Ränder einen Treppeneffekt aufweisen.



(a) Original

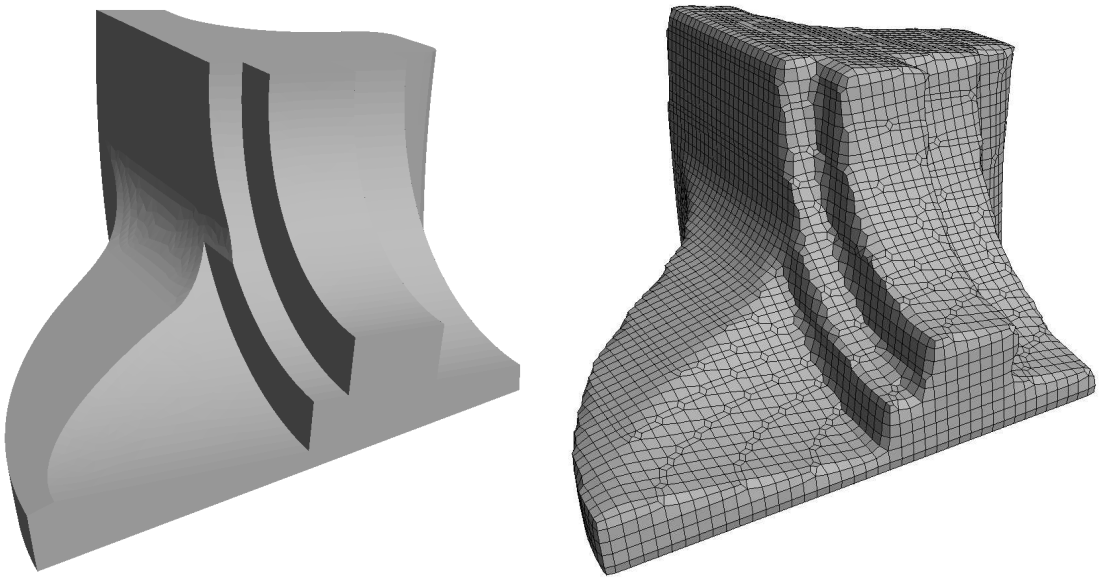


(b) Neuvernetzung ohne Merkmalverbesserung

Abbildung 5.1: Neuvernetzung eines Bauteils ohne Merkmalverbesserung. Die Ränder werden mit großen Abweichungen wiedergegeben.

Einer der Gründe dafür ist sicherlich das Auflösungsvermögen des Volumengitters. Durch die zu grobe Auflösung verschwinden die kleinen Löcher vollständig aus der Rekonstruktion. Das lässt sich in Abbildung 5.1(b) deutlich erkennen.

Es wird nun im folgenden Kapitel gezeigt, wie man durch Anreicherung der Volumenbeschreibung mit zusätzlichen Daten die Rekonstruktion der spezifischen Merkmale verbessern kann.



(a) Original (b) Neuvernetzung ohne Merkmalverbesserung

Abbildung 5.2: Neuvernetzung eines Werkstücks ohne Merkmalverbesserung. Die scharfen Kanten und Ecken werden geglättet wiedergegeben.

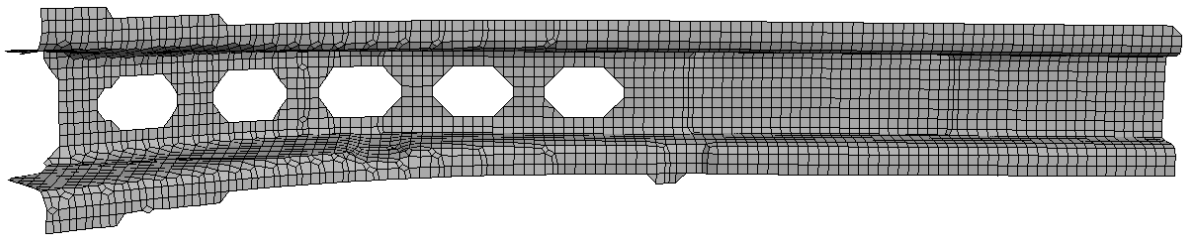
5.1 Anreicherung mit Merkmalpunkten

Wie schon in Kapitel 3.2 und 3.3 angesprochen, erzeugt der modifizierte MC Algorithmus kein striktes Dreiecksnetz, sondern ein Netz aus Polygonen mit verschiedener Anzahl der Ecken. Da beim Dual-Operator zu jedem Polygon ein dualer Vertex berechnet werden muss, kann man diese Berechnung bereits bei der Extraktion der Isofläche mit MC Algorithmus tun. Eine naheliegende Methode ist es den dualen Vertex auf den Mittelpunkt des dazugehörigen Polygons der MC-Fläche zu setzen. Dazu berechnet man schon im Vorfeld die Mittelpunkte aller Polygone der MC-Fläche als gemittelte Summe aller Vertices des jeweiligen Polygons. Beim Konstruieren der dualen Vertices setzt man dann die Koordinaten eines dualen Vertex auf den vorberechneten Mittelpunkt des jeweiligen Polygons. Die Auswahl des Mittelpunktes als dualen Vertex sorgt auch dafür, dass dabei ähnlich große duale Polygone entstehen (siehe Abbildungen 5.1 und 5.2).

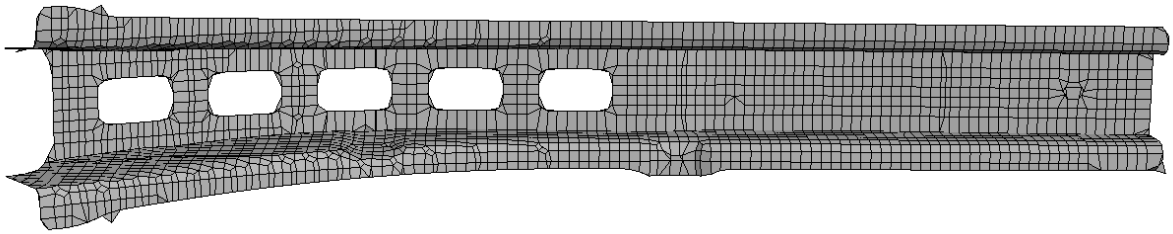
Bei der gerade beschriebenen Implementierung ist die Geometrie des dualen Netzes durch die Wahl der Mittelpunkte festgelegt. Das kann man ausnutzen, um spezifische Merkmale wiederzugeben. Zum Beispiel indem die Mittelpunkte bestimmter Polygone gezielt verschoben werden. Dazu extrahiert man zuerst aus dem Original die wichtigen Merkmale (Ränder, Kanten, Eckpunkte) und bestimmt dann die Würfel des Volumens, die die extrahierten Merkmale enthalten[1]. In diesen Würfeln wird zusätzlich ein s.g. Merkmalpunkt gespeichert, der dann dazu verwendet wird, die Mittelpunkte der Polygone gezielt zu verschieben.

Soweit die Auswahl des Merkmalpunktes Freiheitsgrade hat, soll diese hauptsächlich die Geometrie des Merkmals berücksichtigen und zusätzlich für eine optimale Verteilung unter den be-

[1] Die Koordinaten des Würfels werden effizient mittels Abschneiden der Nachkommastellen bestimmt.



(a) Neuvernetzung ohne Merkmalverbesserung

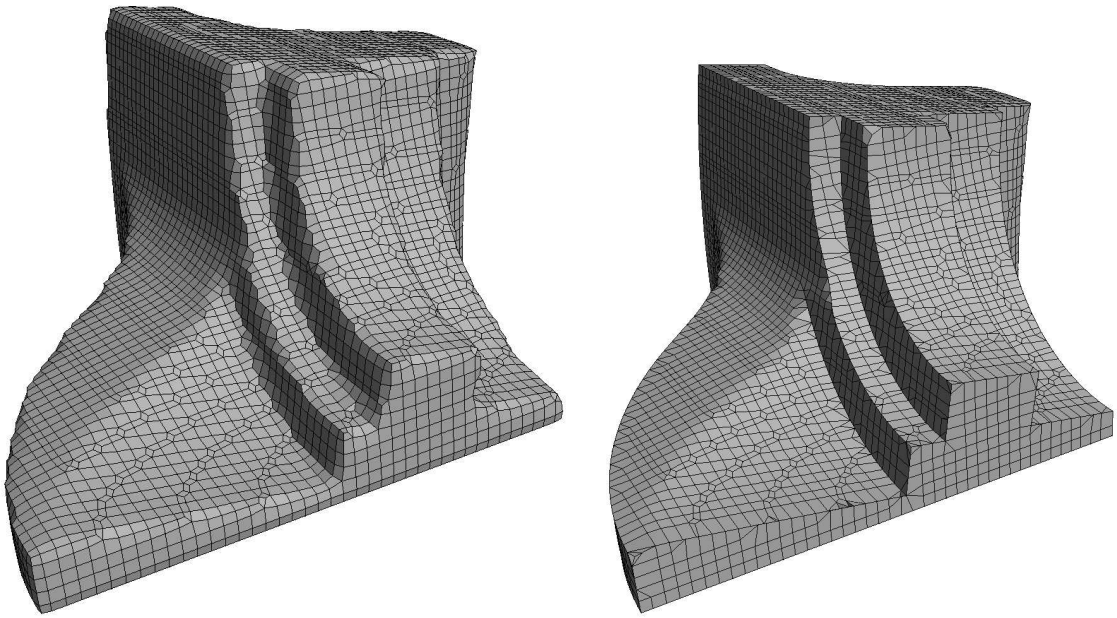


(b) Neuvernetzung mit Merkmalverbesserung

Abbildung 5.3: Neuvernetzung eines Bauteils mit Merkmalverbesserung.

nachbarten Merkmalpunkten sorgen. Bei den Kanten und Rändern nimmt man deshalb einen Merkmalpunkt, der auf der Kante bzw. auf dem Rand liegt und den geringsten Abstand zum Zentrum des Würfels besitzt. Bei den scharfen Ecken hat man keine Freiheitsgrade und der Merkmalpunkt bekommt folglich die Koordinaten der Ecke. Die Forderung, dass Merkmalpunkte möglichst nah am Zentrum des Würfels liegen sollen, dient dem Zweck eine bessere Verteilung der dualen Vertices zu erreichen.

Nach der Extraktion der Isofläche mit MC Algorithmus setzt man bei denjenigen Polygonen, die sich in Würfeln mit Merkmalpunkten befinden, die Mittelpunkte auf die Merkmalpunkte. Bei den restlichen Polygonen berechnet man die Mittelpunkte als gemittelte Summe aller Vertices des jeweiligen Polygons. Die Merkmalpunkte werden in Mittelpunkten der Polygone gespeichert und die Mittelpunkte werden ihrerseits zu dualen Vertices. Dadurch werden bestimmte duale Vertices auf den Merkmalen platziert. Die Abbildungen 5.3 und 5.4 zeigen die dadurch erreichten Verbesserungen bei scharfen Kanten und Rändern. Im folgenden Kapitel wird nun erläutert, wie man spezifische Merkmale bestimmen kann.



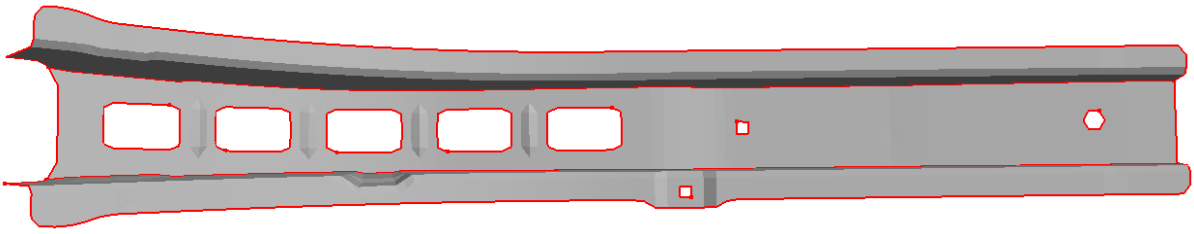
(a) Neuvernetzung ohne Merkmalverbesserung (b) Neuvernetzung mit Merkmalverbesserung

Abbildung 5.4: Neuvernetzung eines Werkstücks mit Merkmalverbesserung.

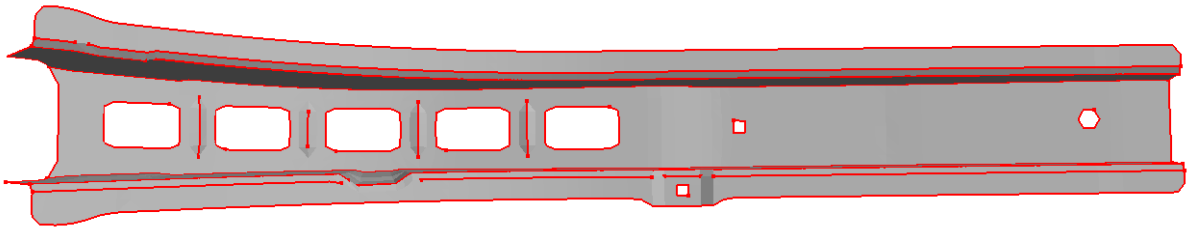
5.2 Merkmalbestimmung

Bei der Merkmalbestimmung geht es darum, Ränder, Kanten und Ecken aus einem Polygonnetz zu extrahieren. Im Folgenden werden einige Ansätze dazu präsentiert.

Um die Merkmalbestimmung effizient durchführen zu können, braucht man eine geeignete Datenstruktur, mit der man zu jeder Kante die dazugehörigen Polygone bestimmen kann. Bei einer kantenorientierten Speicherung ist diese Information relativ leicht zu bekommen. Bei einer polygonorientierten Speicherung wird ein zusätzlicher Kantenindex benötigt, der einen effizienten Zugriff auf adjazente Polygone über eine Kante erlaubt. Für meine Implementierung habe ich eine polygonorientierte Speicherung des Netzes als primäre Datenstruktur gewählt und habe daraus den Kantenindex aufgebaut.



(a) Kantendetektion mit $\psi = 0.3$



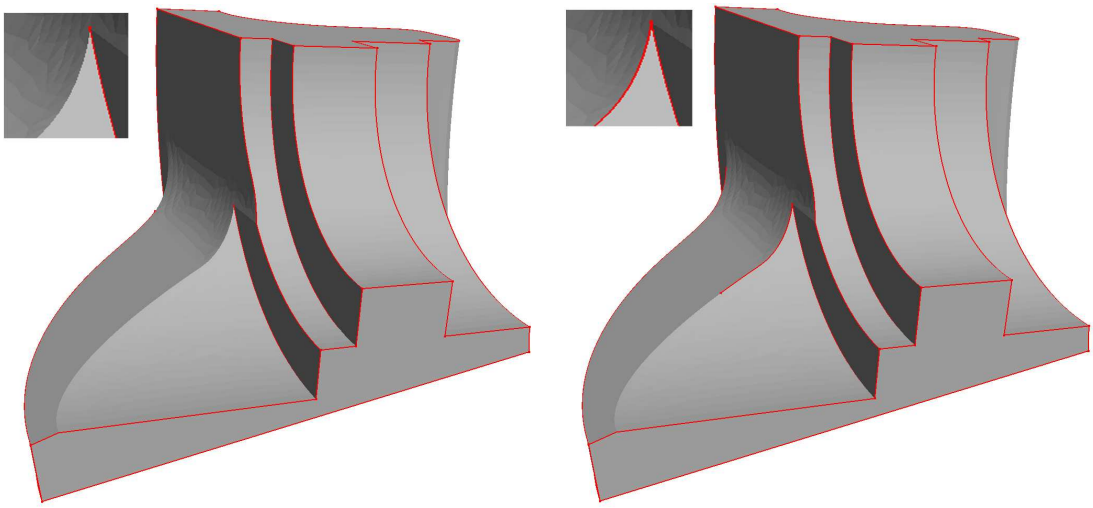
(b) Kantendetektion mit $\psi = 0.8$

Abbildung 5.5: Merkmalbestimmung bei einem Bauteil

Es wird im Weiteren davon ausgegangen, dass eine Kante in maximal zwei Polygonen enthalten sein kann (anderenfalls muss man das Polygonnetz entsprechend aufteilen). Ist die Kante in nur einem Polygon enthalten, dann gehört sie zum Rand. Damit können Ränder auf eine einfache Weise bestimmt werden.

Um scharfe Kanten zu finden, berechnet man den Winkel zwischen zwei adjazenten Polygonen über das Skalarprodukt ihrer Normalenvektoren. Ist er größer als vorgegebene Grenzwert ψ, handelt es sich um eine scharfe Kante. Einzelne scharfen Kanten werden zu längeren Kantenzügen zusammengefaßt. Das Zusammenfassen erfolgt dann, wenn die Richtungsvektoren der beiden Kanten einen hinreichend kleinen Winkel einschließen. Die Merkmalpunkte können dann entlang dieses Kantenzuges wie in vorherigen Kapitel beschrieben platziert werden. Die Abbildungen 5.5 und 5.6 zeigen die Kantenerkennung bei verschiedenen Werten für ψ.

Die scharfen Ecken befinden sich dort, wo sich zwei oder mehrere scharfe Kantenzüge zusammentreffen. Man kann außerdem den Winkel zwischen den Kantenzügen als Kriterium für eine Ecke



(a) Kantendetektion mit $\psi = 0.3$ (b) Kantendetektion mit $\psi = 0.8$

Abbildung 5.6: Merkmalbestimmung bei Fandisk

heranziehen. Dieser Grenzwert sollte dann größer sein, als der Grenzwert ψ für die Bestimmung der scharfen Kanten[2].

In [KBSS01] wird Merkmalbestimmung auf eine ähnliche Weise zusammen mit einer speziellen Volumenbeschreibung und einem erweiterten Marching Cubes Algorithmus verwendet, um bessere Approximation der scharfen Merkmale in Dreiecksnetzen zu erhalten.

[2]Leider konnte die Bestimmung der scharfen Ecken nicht in dem Umfang implementiert werden, wie es hier beschrieben ist. Es ist zu untersuchen, wie genau diese Methode die scharfen Ecken bestimmen kann.

6 Qualitätsverbesserung des Netzes

In diesem Kapitel werden einige Aspekte der Qualitätsverbesserung besprochen, die für den entwickelten Algorithmus spezifisch sind.

Eines der wichtigsten Kriterien ist außerdem der Anteil an Vierecken. Wie bereits in Kapitel 4 besprochen, war es im Rahmen dieser Arbeit nicht erforderlich reine Vierecksnetze zu erzeugen. Da ein sehr hoher Anteil an Vierecken erreicht werden konnte, wird auf dieses Kriterium nicht näher eingegangen.

Ein weiteres Qualitätskriterium ist das s.g. „Warping“. Ein Polygon besitzt Warping, wenn seine Ecken nicht in einer Ebene liegen. Es ist offensichtlich, dass dieses Problem nur bei Polygonen mit einem Grad größer 3 auftritt, da in einem Dreieck die Eckpunkte immer in einer Ebene liegen. Ein zu starkes Warping kann eine FE-Simulation negativ beeinflussen und ist somit nach Möglichkeit zu vermeiden bzw. zu korrigieren. Mehr über Warping findet man in [Cap90].

Eine der erwünschten Eigenschaften ist es auch, möglichst regelmäßige und gleichgroße Vierecke zu erhalten, deren Winkel nicht zu stumpf und nicht zu spitz sind. Dazu können die s.g. Rhombuspolygone entfernt werden.

6.1 Entfernen der Rhombuspolygone

Die Vierecksnetze, die mit beschriebenen Algorithmen erzeugt werden, weisen fast immer rhombusartige Polygone auf. In Abbildung 6.1 wurden diese Polygone rot eingefärbt.

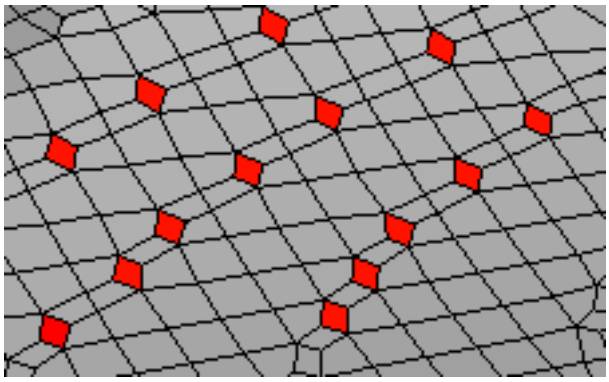


Abbildung 6.1: Rhombus Polygone in einem Netz. Zur Veranschaulichung rot eingefärbt.

Die Rhombuspolygone besitzen Vertices mit ungleichen Gradzahlen, was zu einer unregelmäßigen Netztopologie führt und fast immer zu spitze oder zu stumpfe Winkel mit sich bringt. Man kann Rhombuspolygone nicht im Vorfeld ausschließen, weil der Marching Cubes Algorithmus Polygone mit verschiedenen Gradzahlen erzeugt und die Gradzahlen der Polygone den Gradzahlen der Vertices des dualen Netzes entsprechen (vgl. Kapitel 3.3).

Die Rhombuspolygone können im dualen Netz an den Gradzahlen ihrer Vertices erkannt werden. In Abbildung 6.2 ist ein isoliertes Rhombuspolygon $ABCD$ dargestellt. Seine Ecken B und

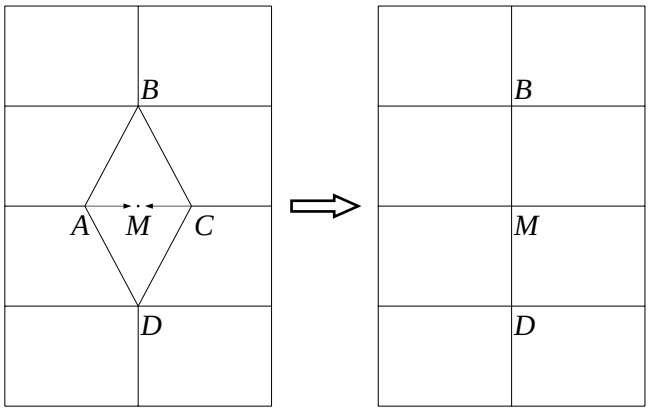
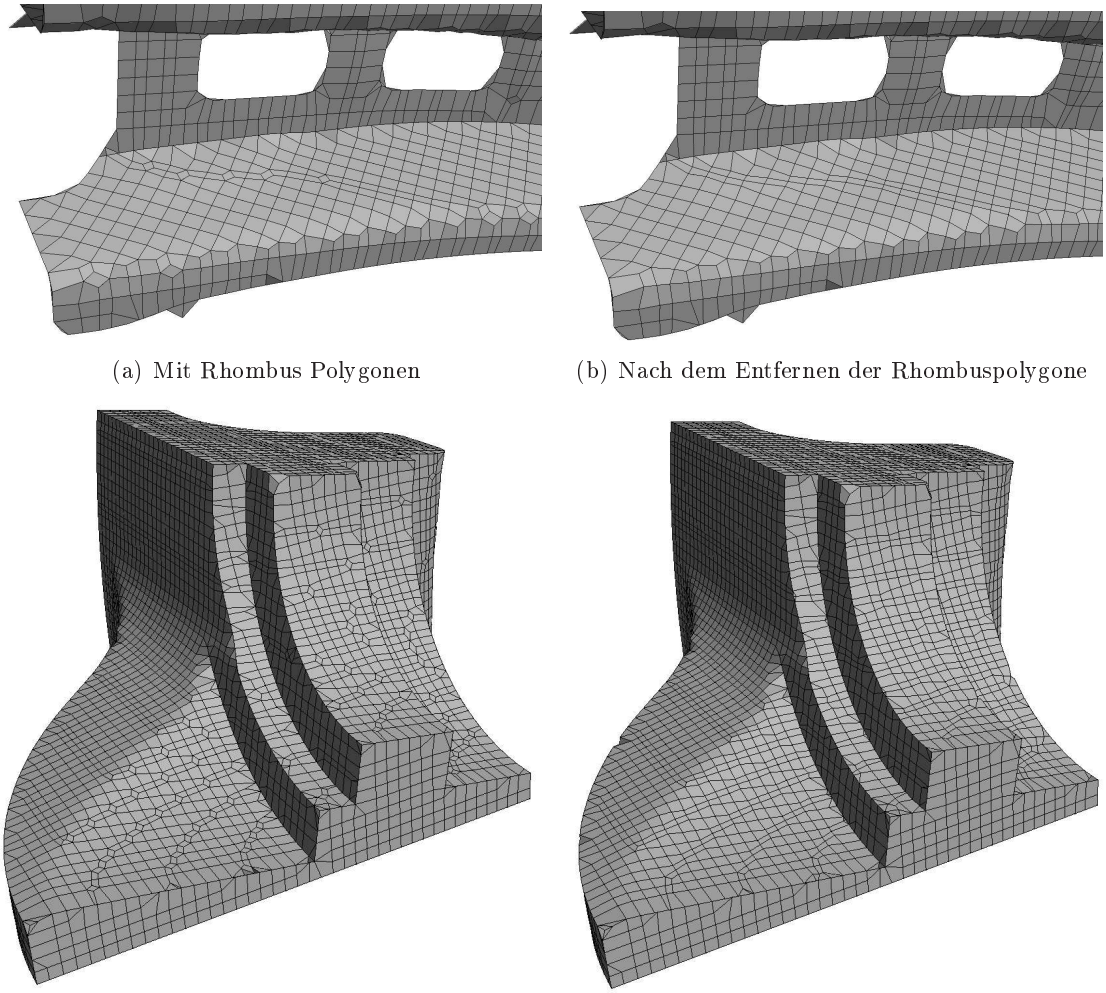


Abbildung 6.2: Entfernen der Rhombuspolygone

D haben den Grad 5, die Ecken A und C den Grad 3. Wenn man dieses Rhombuspolygon enfernt, indem man die Punkte A und C zum Punkt M zusammenlegt, dann erhält man ein regelmäßiges Vierecksnetz, bei dem jeder Vertex den Grad 4 hat. Allgemein ausgedrückt ändern sich die Gradzahlen der Vertices wie folgt:

$$grad(M) = grad(A) + grad(C) - 2$$
$$grad(B_{neu}) = grad(B_{alt}) - 1$$
$$grad(D_{neu}) = grad(D_{alt}) - 1$$

Wir können dies auch bei solchen Rhombuspolygonen anwenden, deren gegenüberliegende Ecken B und D einen höheren Grad als 5 haben. Das ist auch aus dem Grund zu empfehlen, weil eine der Ecken B oder D gleichzeitig in einem anderen Rhombuspolygon enthalten sein kann. Die Entfernung der beiden Rhombuspolygone würde dann den Grad des gemeinsamen Vertex von 6 auf den gewünschten Grad 4 reduzieren. Meine Implementierung überprüft die Vertexgrade jedes Polygons und legt ggf. die Punkte A und C zu einem neuen Punkt M zusammen (siehe Abbildung 6.2 rechts), welcher die Koordinaten des vorberechneten Polygonmittelpunktes (siehe Kapitel 5) erhält. In Abbildung 6.3 sieht man die Ergebnisse dieses Algorithmus.



(a) Mit Rhombus Polygonen

(b) Nach dem Entfernen der Rhombuspolygone

(c) Mit Rhombuspolygonen

(d) Nach dem Entfernen der Rhombuspolygone

Abbildung 6.3: Entfernen der Rhombuspolygone

6.2 Merkmalerhaltendes Glätten des Netzes

Beim Glätten des Netzes werden einzelne Vertices des Netzes verschoben, um gleichmäßigere Polygone zu erhalten.

Dabei werden zuerst für jeden Vertex seine neuen Koordinaten berechnet und in einer separaten Datenstruktur abgelegt. Anschließend weist man die neuen Koordinaten den Vertices zu. In meiner Implementierung werden die neuen Koordinaten als gemittelte Summe der Koordinaten der Vertices in direkter Nachbarschaft berechnet.

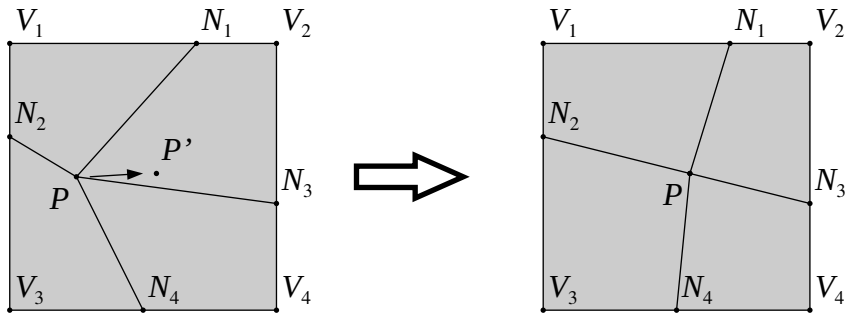
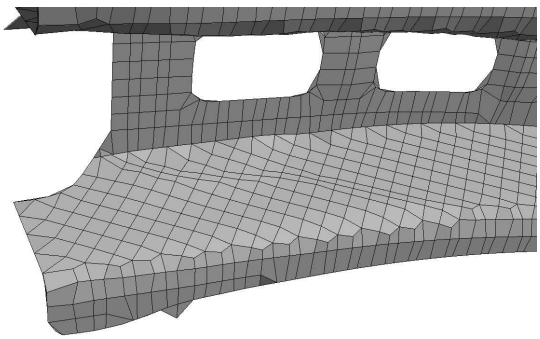


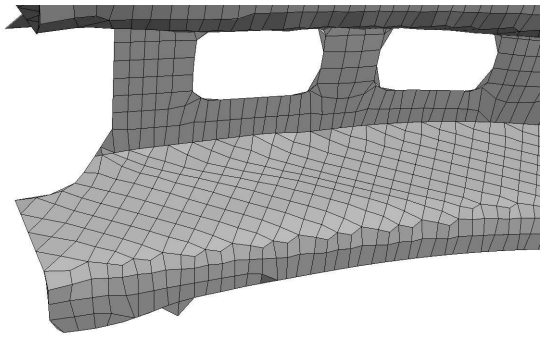
Abbildung 6.4: Glättung des Netzes

In Abbildung 6.4 werden für Vertex P neue Koordinaten berechnet (dargestellt mit P'), indem die Koordinaten seiner Nachbarvertices N_1,N_2,N_3 und N_4 gemittelt werden. Der Vertex P wird später nach P' verschoben. Das Zuweisen der neuen Koordinaten erfolgt nachdem für alle Vertices die Berechnungen abgeschlossen sind. Sonst können die bereits aktualisierten Koordinaten die laufenden Berechnungen verfälschen.

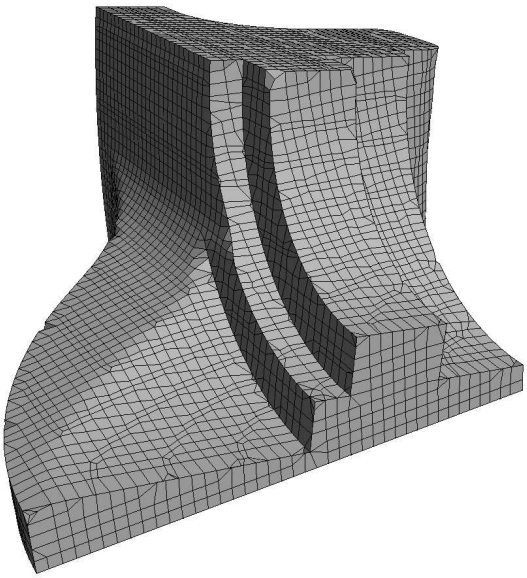
Bei Rändern, scharfen Kanten und Ecken wirkt sich die Glättung negativ aus. Um diesem vorzubeugen, werden in meiner Implementierung den Vertices unterschiedliche Merkmalattribute zugewiesen. Bei der Glättung werden die Vertices mit Merkmalattributen nicht verschoben. Die Attribute werden bei der Merkmalextraktion (siehe Kapitel 5) im Volumen mitgespeichert und dann auf die Polygonmittelpunkte des MC-Netzes übertragen. Durch den Dual-Operator werden diese Attribute zu den Vertex-Attributen. Abbildung 6.5 zeigt die Auswirkung des Glättens. Man kann erkennen, dass die Polygone nach dem Glätten besser ausgerichtet sind.



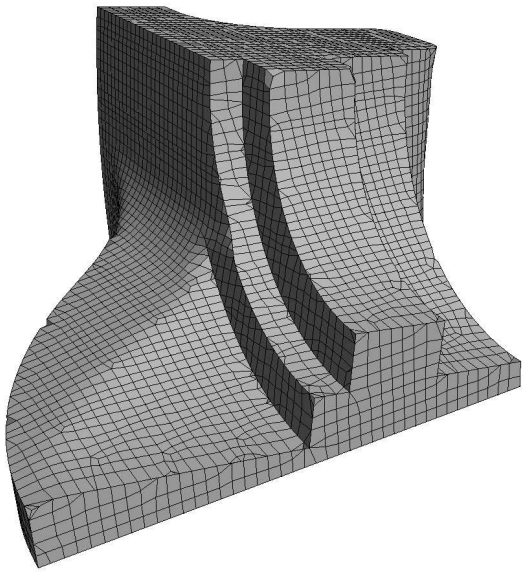
(a) Vor dem Glätten



(b) Nach dem Glätten



(c) Vor dem Glätten



(d) Nach dem Glätten

Abbildung 6.5: Merkmalerhaltendes Glätten eines Netzes

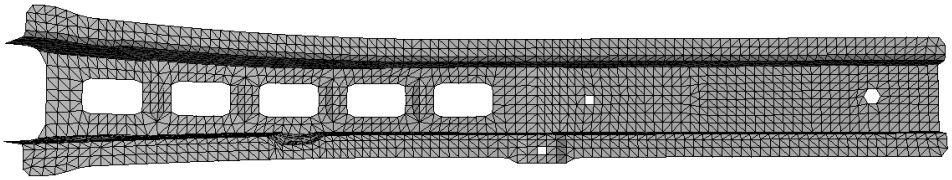
7 Ergebnisse

In diesem Kapitel werden die Ergebnisse bewertet und die Laufzeiten des Algorithmus für verschiedene Modelle präsentiert.

7.1 Bewertung des rekonstruierten Polygonnetzes

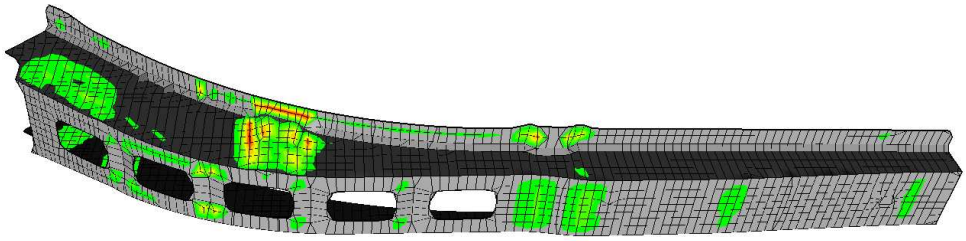
Abbildung 7.1 zeigt die Unterschiede zwischen dem rekonstruierten Vierecksnetz (Abbildung 7.1(b),(c)) und dem Originalnetz (Abbildung 7.1(a)). Die größten Abweichungen treten in Bereichen der scharfen Kanten auf, die bei der Merkmalbestimmung nicht erkannt und deswegen geglättet rekonstruiert wurden (in Abbildung 7.1 rot eingefärbt). Das lässt sich in einigen Fällen mit einem höheren Wert ψ verbessern (siehe Kapitel 5.2), wobei der optimale Wert für ψ für jedes Netz individuell sein kann. Für die Rekonstruktion, die in Abbildung 7.1 dargestellt ist, wurde $\psi = 0.8$ gewählt.

Die flachen und schwach gekrümmten Bereiche weisen dagegen kaum Unterschiede auf. Sie sind in der gleichen Abbildung grau bzw. grün eingefärbt.



(a) Original

(b) Rekonstruktion



(c) Rekonstruktion (gedreht)

Abbildung 7.1: Unterschiede zum Original. Grüne Bereiche haben eine Abweichung von 0.01 mm, rote Bereiche über 1 mm. Abmessungen des gesamten Bauteils $778 \times 135 \times 184$ mm^3. Das Bild wurde mit *scFEMod* erstellt (siehe [fem]).

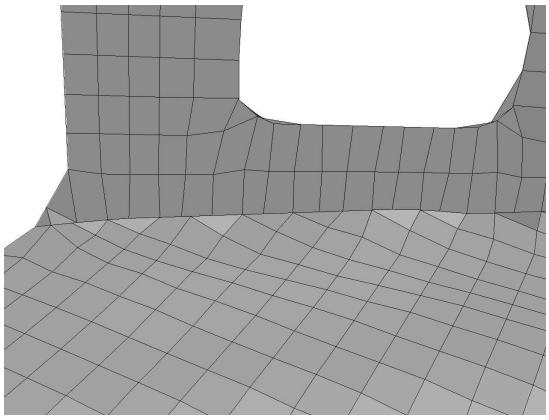


Abbildung 7.2: Degenerierte Vierecke im Bereich der scharfen Kanten.

Im Bereich der scharfen Kanten werden Vertices auf die Kantenlinie verschoben. Es können dabei degenerierte Polygone entstehen (siehe Abbildung 7.2). Wenn 3 Vertices auf die Kantenlinie verschoben werden, dann sehen die Vierecke wie Dreiecke mit einem T-Vertex[1] aus. Solche Vierecke können in zwei Dreiecke aufgeteilt werden, indem eine Kante zwischen dem vermeintlichen T-Vertex und dem gegenüberliegenden Vertex hinzugefügt wird. Dieses Problem könnte eventuell besser gelöst werden, wenn sich die Ausrichtung der Polygone an den scharfen Kanten bzw. Rändern orientiert. Leider ist die Ausrichtung bei dem beschriebenen Algorithmus durch das Volumengitter bestimmt.

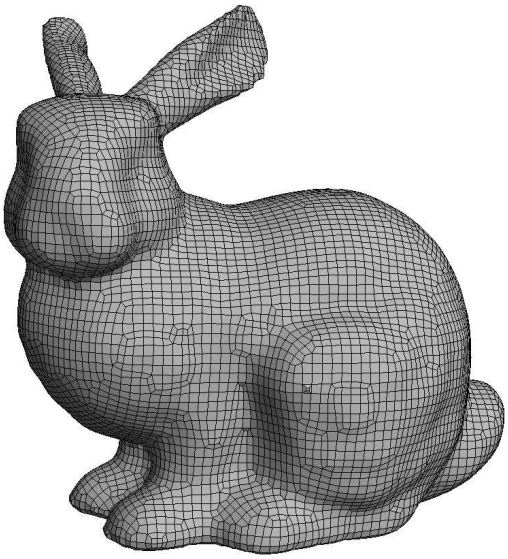
Kleinere Löcher können nur dann rekonstruiert werden, wenn ihre Größe die Auflösung des Volumens deutlich übertrifft[2]. Es ist offensichtlich, dass alle Details der ursprünglichen Fläche, die kleiner sind als die Volumenauflösung gar nicht oder ungenau rekonstruiert werden können.

Der Algorithmus lässt sich auch für Modelle anwenden, bei welchen die Erhaltung der scharfen Merkmale nicht im Vordergrund steht. In Abbildung 7.3 sind dazu einige Beispiele zusammengestellt worden.

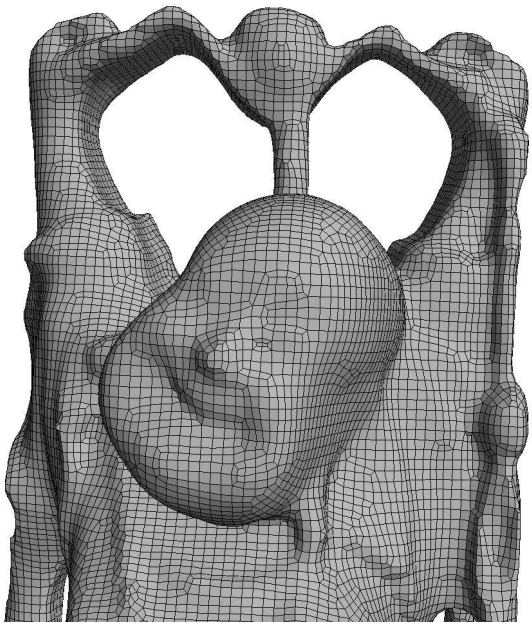
7.2 Laufzeiten

In der Tabelle 7.1 sind die Laufzeiten eingetragen, die der gesamte Algorithmus für verschiedene Modelle braucht. Neben den Modellen „Träger“ und dem „Fandisk“ wurden bekannte Datensätze wie „Stanford Bunny“ und „Happy Buddha“ zum Vergleich herangezogen. Die Tests wurden auf einem Athlon XP 1800+ mit 1GB RAM durchgeführt. Für die Voxelisierung wurde jeweils ein Volumen genommen, welches die Seitenverhältnisse des Modells besitzt und dabei die angegebene maximale Anzahl der Voxel nicht überschreitet. In den letzten beiden Spalten steht die Anzahl der erzeugten Polygone und die jeweilige Laufzeit. Die Laufzeit ist maßgeblich von der Anzahl der Dreiecke im Modell und der gewählten Volumengröße abhängig.

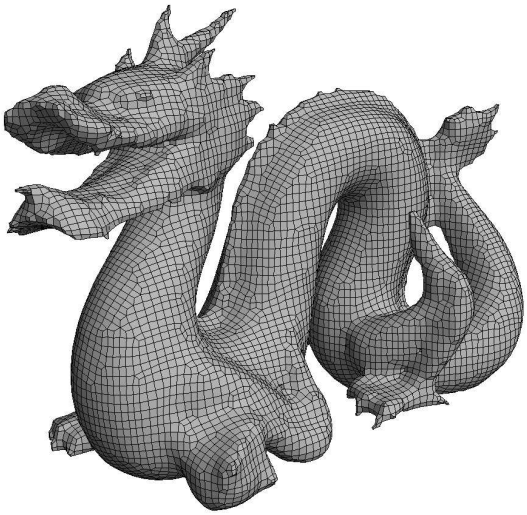
[1] Ein Vertex auf der Kante. Die Kante selbst ist durch zwei andere Vertices gegeben
[2] Volumenauflösung ist gleich der Kantenlänge der Würfel im Volumen



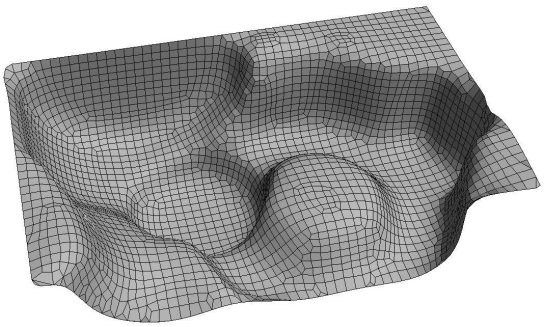
(a) Stanford Bunny



(b) Happy Buddha



(c) Dragon



(d) Aschenbecher

Abbildung 7.3: Beispiele der Neuvernetzung

Modell	# Dreiecke	Max. # Voxel	# Polygone	Dauer
Träger	4276	10^5	3117	0,9 s
Träger	4276	10^6	18154	2,2 s
Fandisk	12946	10^5	6630	1,2 s
Fandisk	12946	10^6	36496	3,4 s
Bunny	69451	10^5	6376	2,4 s
Bunny	69451	10^6	32460	4,8 s
Buddha	1087716	10^6	53114	21,9 s

Tabelle 7.1: Laufzeiten des Algorithmus

8 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Algorithmus präsentiert, welcher ein Dreiecksnetz in ein Polygonnetz mit hohem Vierecksanteil umwandelt. Dabei wird als Zwischenschritt ein vorzeichenbehaftetes Distanzfeld aufgebaut. Die Merkmale des Dreiecksnetzes werden bestimmt und es wird pro Würfel ein repräsentativer Merkmalpunkt gespeichert, der auf dem Merkmal liegt und zum Mittelpunkt des Würfels den kleinsten Abstand hat. Aus dem Distanzfeld wird eine Isofläche mit modifiziertem Marching Cubes Algorithmus extrahiert. Die Modifikation sorgt dafür, dass möglichst viele Vertices den Grad 4 haben und Ränder rekonstruiert werden können. Die Mittelpunkte der Polygone der Isofläche werden auf die Merkmalpunkte im Distanzfeld gesetzt. Beim Konstruieren des dualen Netzes werden Mittelpunkte der Polygone als duale Vertices übernommen. Schließlich wird das duale Netz, das überwiegend aus Vierecken besteht, optimiert. Dabei werden zuerst die Vertexgrade ausgeglichen, indem die Rhombuspolygone entfernt werden. Anschließend unterzieht man das Netz einem merkmalerhaltenden Glätten, um eine bessere Ausrichtung der Polygone zu erreichen.

Die Genauigkeit des beschriebenen Verfahrens hängt hauptsächlich von der Auflösung des Volumens ab. Allerdings steigt mit der Auflösung des Volumens nicht nur die Laufzeit des Algorithmus, sondern auch die Anzahl der erzeugten Polygone.

Das Neuvernetzen ohne Berücksichtigung der scharfen Kanten erzeugt eine sehr regelmäßige Vernetzung. Die scharfen Kanten können zwar reproduziert werden, weisen aber teilweise zu stark degenerierte Polygone auf. Dieses Problem muss weiter untersucht werden, da es für FE-Simulationen nicht akzeptabel ist. Der Grund für eine starke Degeneration liegt zum Teil daran, dass die Ausrichtung der Vierecke nicht an den Merkmalen orientiert, sondern durch das Volumengitter bestimmt ist. Ein möglicher Ansatz hier wäre die Aufspaltung der Vierecke im Bereich der Merkmale in zwei Dreiecke, um eine bessere Ausrichtung zu erhalten.

Es ist ebenfalls zu untersuchen, ob die Zwischenschritte der Isoflächenextraktion und Bilden des dualen Netzes zusammengefasst werden können. Die Extraktion der Fläche aus dem Volumen könnte dann ähnlich wie bei [JLSW02] oder [SW04] ablaufen. Dabei sollte man alle topologischen Fälle im Würfel, die im Zusammenhang mit dem Marching Cubes Algorithmus untersucht wurden, genau berücksichtigen.

Eine Volumenbeschreibung hat in bestimmten Fällen einige Vorteile gegenüber der polygonalen Beschreibung. Die meisten Nachteile resultieren aus der uniformen Verteilung der Gitterpunkte im Volumen und dem daraus folgenden Auflösungsvermögen. Es ist an dieser Stelle zu untersuchen, in wie weit ein adaptives, nicht uniformes Volumengitter dieses verbessert. Die scharfen Ecken und Kanten mit ihrer sprunghaften Änderung des Normalenvektors stellen aber unabhängig von der Auflösung ein Problem dar.

Es konnte bereits im Vorfeld festgestellt werden, dass eine optimale Neuvernetzung in Vierecksnetze und insbesondere eine für FE-Simulation geeignete Neuvernetzung von Flächenbauteilen ein nicht triviales Problem darstellt. Die vorliegende Arbeit präsentiert dazu einen neuen Ansatz, auch wenn zugegeben werden muss, dass weitere Maßnahmen notwendig sind, um das erzeugte Polygonnetz für FE-Simulationen verwenden zu können.

Literaturverzeichnis

[ACSD+03] Alliez, Pierre ; Cohen-Steiner, David ; Devillers, Olivier ; Levy, Bruno ; Desbrun, Mathieu: *Anisotropic Polygonal Remeshing.* 2003

[BA02] Bærentzen, J. A. ; Aanæs, Henrik: *Generating Signed Distance Fields From Triangle Meshes.* 2002

[BK04] Botsch, Mario ; Kobbelt, Leif P.: *A Remeshing Approach to Multiresolution Modeling.* 2004

[Cap90] Capra, Renato: Preliminary mesh checking for structural analysis. In: *APL '90: Conference proceedings on APL 90: for the future.* New York, NY, USA : ACM Press, 1990. – ISBN 0–89791–371–X, S. 61–70

[Che95] Chernyaev, Evgeni V.: *Marching Cubes 33: Construction of Topologically Correct Isosurfaces.* 1995

[Dho99] Dhondt, Guido D.: *Unstructured 20-Node Brick Element Meshing.* 1999

[fem] *scFEMod.* `http://www.science-computing.de/software/scfemod.html`

[Gib98] Gibson, Sarah F. F.: *Using Distance Maps for Accurate Surface Representation in Sampled Volumes.* 1998

[JLSW02] Ju, Tao ; Losasso, Frank ; Schaefer, Scott ; Warren, Joe: Dual Contouring of Hermite Data. In: *Siggraph 2002, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 339–346

[Jon96] Jones, M.W.: *The production of volume data from triangular meshes using voxelisation.* 1996

[KBSS01] Kobbelt, Leif P. ; Botsch, Mario ; Schwanecke, Ulrich ; Seidel, Hans-Peter: Feature-Sensitive Surface Extraction From Volume Data. In: Fiume, Eugene (Hrsg.): *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, 57–66

[LB03] Lopes, Adriano ; Brodlie, Ken: Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing. In: *IEEE Transactions on Visualization and Computer Graphics* (2003)

[LC87] Lorensen, William E. ; Cline, Harvey E.: *Marching Cubes: A High Resolution 3D Surface Construction Algorithm.* 1987

[LHMG02] Labsik, Ulf ; Hormann, Kai ; Meister, Martin ; Greiner, Günther: *Hierarchical Iso-Surface Extraction.* 2002

[LLVT03] Lewiner, Thomas ; Lopes, Hélio ; Vieira, Antônio W. ; Tavares, Geovan: *Efficient implementation of Marching Cubes cases with topological guarantees.* 2003

[MK04] Marinov, M. ; Kobbelt, L.: *Direct anisotropic quad-dominant remeshing.* 2004

[MSS94] Montani, C. ; Scateni, R. ; Scopigno, R.: *Discretized Marching Cubes.* 1994

[Nie04] Nielson, Gregory M.: *Dual Marching Cubes.* 2004

[PWH98] Poston, Tim ; Wong, Tien-Tsin ; Heng, Pheng-Ann: *Multiresolution Isosurface Extraction with Adaptive Skeleton Climbing.* 1998

[QRPB00] Quadros, W. R. ; Ramaswami, K. ; Prinz, F. B. ; B.Gurumoorthy: *LayTracks: A New Approach To Automated Quadrilateral Mesh Generation using MAT.* 2000

[SAG03] Surazhsky, Vitaly ; Alliez, Pierre ; Gotsman, Craig: Isotropic Remeshing of Surfaces: a Local Parameterization Approach. In: *Proceedings of 12th International Meshing Roundtable.* Santa Fe, New Mexico, USA, September 2003, S. 215–224

[SG03] Surazhsky, Vitaly ; Gotsman, Craig: Explicit Surface Remeshing. In: *Proceedings of Eurographics Symposium on Geometry Processing.* Aachen, Germany, Juni 2003, S. 17–28

[SW04] Schaefer, Scott ; Warren, Joe: *Dual Marching Cubes: Primal Contouring of Dual Grids.* 2004

[WDSB00] Wood, Zoe J. ; Desbrun, Mathieu ; Schroeder, Peter ; Breen, David: *Semi-Regular Mesh Extraction from Volumes.* 2000

[Wei05] Weisstein, Eric W.: *Dual Graph. From MathWorld – A Wolfram Web Resource.* `http://mathworld.wolfram.com/DualGraph.html`. Version: 2005

[ZB05] Zhang, Yongjie ; Bajaj, Chandrajit: *Adaptive and Quality Quadrilateral/Hexahedral Meshing from Volumetric Data.* 2005

Abbildungsverzeichnis

1.1 Ein FE-Polygonnetz eines Bauteils bestehend aus Vierecken und Dreiecken. . . . 1

3.1 Der gesamte Ablauf der Rekonstruktion einer Fläche über ihre Volumendarstellung (symbolisch in 2D dargestellt). 5

3.2 Kürzester Abstand von einem Punkt zu einem Dreieck 6

3.3 Dreiecke D_1 und D_2 in der Projektion, Punkt P hat den kleinsten Abstand zu G 7

3.4 Voxelisierung einer einfachen Fläche im Volumen 9×9×9 10

3.5 Fragment der Isofläche in einem Würfel. In den Ecken sind Abstände der Voxel mit Vorzeichen eingetragen. 11

3.6 Ursprüngliche Marching Cubes Tabelle nach [LC87](Darstellung aus [Che95]) . . 12

3.7 Zweideutigkeit an einer Seite eines Würfels bei Marching Cubes 13

3.8 Beispiel von zwei benachbarten nicht eindeutigen Konfigurationen. Es entsteht u.U. eine kleine Lücke auf der gemeinsamen Seite 13

3.9 Marching Cubes 33 Tabelle von Chernyaev (entnommen aus [Che95]) 14

3.10 Duale Graphen. Polygone mit schwarzen Konturen sind dual zu den Polygonen mit weißen Konturen. 16

4.1 Duales Polygon $Q_1Q_2Q_4Q_3$ im Volumen 19

4.2 Die Gradzahl der Vertices vor und nach der Änderung der ausgewählten Fälle der Marching Cubes 33 Tabelle (vgl. Abbildung 3.9). 21

4.3 Aufspalten am Mittelpunkt M (a) oder Abspaltung (b) 22

5.1 Neuvernetzung eines Bauteils ohne Merkmalverbesserung. Die Ränder werden mit großen Abweichungen wiedergegeben. 23

5.2 Neuvernetzung eines Werkstücks ohne Merkmalverbesserung. Die scharfen Kanten und Ecken werden geglättet wiedergegeben. 24

5.3 Neuvernetzung eines Bauteils mit Merkmalverbesserung. 25

5.4 Neuvernetzung eines Werkstücks mit Merkmalverbesserung. 26

5.5 Merkmalbestimmung bei einem Bauteil 27

5.6 Merkmalbestimmung bei Fandisk 28

6.1 Rhombus Polygone in einem Netz. Zur Veranschaulichung rot eingefärbt. 29

6.2 Entfernen der Rhombuspolygone 30

6.3 Entfernen der Rhombuspolygone 31

6.4 Glättung des Netzes 32

6.5 Merkmalerhaltendes Glätten eines Netzes 33

7.1 Unterschiede zum Original. Grüne Bereiche haben eine Abweichung von 0.01 mm, rote Bereiche über 1 mm. Abmessungen des gesamten Bauteils $778 \times 135 \times 184$ mm^3. Das Bild wurde mit *scFEMod* erstellt (siehe [fem]). 34

7.2 Degenerierte Vierecke im Bereich der scharfen Kanten. 35

7.3 Beispiele der Neuvernetzung 36