Georg-August-Universität Göttingen
Institut für angewandte Informatik

Seminar Computergrafik

Auffüllen von Löchern in Polygonnetzen

Nach einem Verfahren von Peter Liepa

Benjamin Willenberg

20. Februar 2012

geleitet durch Dipl.-Inf. Reinhard Hemmerling
und Prof. Dr. Winfried Kurth

Inhaltsverzeichnis

1 Einleitung

In jüngster Zeit haben Polygonnetze in der Computergrafik immer größere Popularität erlangt. Sie eignen sich hervoragend zur mathematischen Beschreibung geometrischer Daten. Ebenso vielfältig wie die Anwendungsbereiche der Polygonnetze in der modernen Computergrafik sind jedoch die Anforderungen, die an sie gestellt werden müssen, damit diverse Weiterverarbeitungen zu gültigen Ausgaben führen. Beispielsweise werden mannigfaltige und wasserdichte Polygonnetze benötigt, um 3D-Drucke (Stereo Lithography Prototyping) oder physikalische Berechnungen in Form der Finite-Elemente-Methode mit dem Modell durchführen zu können.
Bei vielen Anwendungen werden die Polygonnetzte jedoch nicht primär erzeugt, sondern aus einer Punktwolke generiert. Beispiele hierfür ist die 3D-Rekonstruktion von Gegenständen aus mehreren zweidimensionalen Bildern, die aus verschiedenen Perspektiven aufgenommen wurden, oder die Computertomographie, bei der das Bilder aus Dichteinformationen rekonstruiert wird, die nur für bestimmte Linien oder Schnitte durch das Modell in aufintegrierter Form bekannt sind. Aus diesen Punktwolken werden durch spezielle Meshingalgorithmen die Polygonnetze erzeugt, die dann nicht mehr garantiert defektfrei sind. Typische Artefakte, die in den Polygonnetzen auftreten, sind Überdeckungen, Löcher, Spalte und Selbstschnitte.
Für die Weiterverarbeitung der Polygonnetze ist es somit essentiell auf zuverlässige Algorithmen zurückgreifen zu können, die in der Lage sind, möglichst automatisch die auftretenden Artefakte zu entfernen.
Prinzipiell gibt es zwei unterschiedliche Herangehensweisen, um die Artefakte in einem Polygonnetz zu entfernen: Oberflächenbasierte und volumenbasierte Verfahren. Nach einer kurzen Gegenüberstellung der Vor- und Nachteile der jeweiligen Verfahren werde ich speziell auf ein oberflächenbasiertes Verfahren eingehen. Das von Peter Liepa in [1] vorgeschlagene Verfahren eignet sich dazu, nicht planare Löcher in dreidimensionalen Dreiecksnetzen zu füllen.

2 Oberflächen- und volumenbasierte Verfahren

Bei den Korrekturverfahren für Polygonnetze unterscheidet man zwischen volumen- und oberflächenbasierten Verfahren. Beide Verfahren haben ihre Vor- und Nachteile. Die oberflächenbasierten Verfahren erlauben eine direkte Verarbeitung des als Eingabe übergebenen Polygonnetzes. D. h. die Defekte im Polygonnetz werden gleich auf der

Oberfläche lokalisiert und auch dort beseitigt. Es sind also keine Zwischenberechnungen nötig, die temporär weitere Objekte von dem übergebenen Polygonnetz ableiten. Damit eignen sich oberflächenbasierte Methoden hervorragend dazu, Spalten durch sogenanntes snapping oder stiching, Löcher durch eine Triangulation oder kleinere Überlappungen durch das Aufspalten der Kanten und Dreiecke im kritischen Bereich zu beseitigen. Ein klarer Vorteile von oberflächenbasierten Verfahren ist der Erhalt der Struktur des Originalnetzes. Dadurch ist es möglich, auf lokal an das Polygonnetz geknüpften Eigenschaften weiterhin so zuzugreifen, wie zuvor. Außerdem bieten oberflächenbasierte Verfahren den Vorteil, dass sie nur wenige neue Dreiecke erzeugen, da sie nur in den zu überarbeitenden Bereichen angreifen und nicht das gesamte Polygonnetz neu berechnen. Hierin begründet liegen jedoch auch die Schwächen der oberflächenbasierten Verfahren. Da die Defekte im Polygonnetz erst gefunden werden müssen und dafür oft gewisse Schwellenwerte für Abstände vorgegeben werden müssen, sind die Verfahren sehr anfällig für numerische Ungenauigkeiten, was dazu führt, dass zu kleine Defekte gar nicht erkannt werden. Außerdem müssen oft spezielle Anforderungen an das Eingabenetz gestellt werden, wie zum Beispiel Mannigfaltigkeit oder Selbstschnittfreiheit, um eine gültige Ausgabe, also ein defektfreies Polygonnetz garantieren zu können.
Dem gegenüber stehen die sogenannten volumenbasierten Verfahren. Die generelle Funktionsweise geht auf die Umwandlung des übergebenen Polygonnetzes in ein Volumenmodell zurück. Das berechnete Zwischenmodell wird dabei erzeugt, indem der dreidimensionale Raum in kleine Volumenelemente unterteilt wird und für jedes dieser Volumenelemente festgelegt wird, ob es sich um das Innere, das Äußere oder den Rand des Originalmodells handelt. Aus dem Zwischenmodell wird nach entsprechenden Manipulationen dann wieder ein Polygonnetz generiert. Die Vorteile solcher Verfahren sind per Konstruktion gegeben: es können keine Selbstschnitte, Löcher, Lücken und Überlappungen entstehen. Außerdem ist ein vollautomatisches Arbeiten der Algorithmen ohne Benutzereingriffe mit einem garantiert wasserdichten Polygonnetz als Ausgabe möglich. In der Berechnung des Zwischenmodells sind jedoch auch einige Nachteile begründet. Durch das notwendige Down-Sampling bei der Erstellung des Volumenmodells können Alias-Artefakte auftreten und Detailverluste sind damit natürlich unausweichlich. Des weiteren gehen mögliche lokal an das Polygonnetz geknüpfte Eigenschaften verloren. Als besonders problematisch erweist sich außerdem gerade für größere Objekte, dass die Dreiecksanzahl der Ausgabe in der Regel um ein vielfaches größer als diejenige der Eingabe ist. Neben der enormen Speicherbelastung ist damit eine anschließende Nachbearbeitung des neu generierten Polygonnetzes unausweichlich.

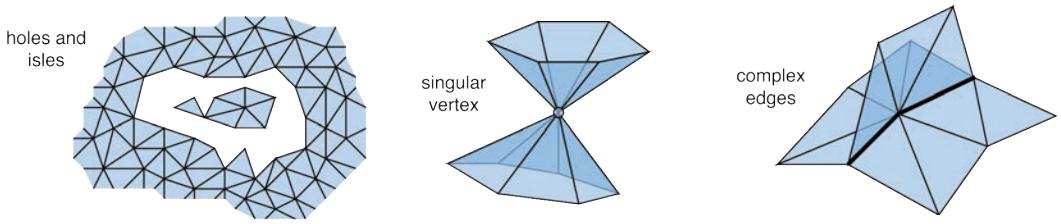

Abbildung 1: Grafische Darstellung möglicher Defekte und topologischer Besonderheiten eines Polygonnetzes: Loch mit Insel, Singulärer Eckpunkt und nicht-mannigfaltige Kante (von links nach rechts) [2].

3 Definitionen und Begriffe

Zum Verständnis des von Peter Liepa vorgeschlagenen Verfahrens sind einige Begriffe nötig, die in diesem Abschnitt eingeführt werden. Ein *Polygonnetz* besteht aus mit Kanten untereinander verbundenen Punkten einer Punktwolke. Handelt es sich bei den dabei entstehenden Polygonen ausschließlich um Dreiecke, so spricht man von einem *Dreiecksnetz*. Wählt man aus dem Dreiecksnetz zwei Dreiecke aus, so heißen diese *benachbart*, wenn sie eine gemeinsame Kante haben. Zwei benachbarte Dreiecke sind des Weiteren *konsistent orientiert*, wenn sie den gleichen Umlaufsinn besitzen. D. h. insbesondere, dass die gemeinsame Kante gegenläufig durchlaufen wird. Sind alle Dreiecke in einem Dreiecksnetz konsistent orientiert, so spricht man von einem *orientierten Dreicksnetz*.
Bei den die Punkte verbinden Kanten im Polygonnetz unterscheiden wir zwischen inneren Kanten und Randkanten. Eine *Randkante* ist eine Kante, die genau an ein einziges Dreieck angrenzt. Jede Kante, die keine Randkante ist, ist eine *innere Kante*. Eine geschlossene Kurve aus Randkanten bildet ein *Loch*. Befinden sich in einem Loch weitere Reste des Polygonnetzes, die nicht mit dem restlichen Netz verbunden sind, so spricht man von einer *Insel*.
Neben den Löchern können weitere Artefakte in einem Polygonnetz auftreten. Diese sind korreliert mit singulären Punkten oder nicht-mannigfaltigen Kanten. *Singuläre Punkte* sind jene Punkte, die mehr als zwei assoziierte Randkanten aufweisen. *Nicht mannigfaltige Kanten* entstehen, wenn mehr als zwei Dreiecke an dieselbe Kante angrenzen. Besitzt ein Polygonnetz nur mannigfaltige Kanten, so spricht man von einem mannigfaltigen Polygonnetz. In Abbildung 1 sind einige dieser möglichen Defekte eines Polygonnetzes dargestellt.

Über die Eigenschaften des Polygon- oder Dreiecksnetzes hinaus werden wir später den sogenannten Diederwinkel benötigen. Der *Diederwinkel* ist der Winkel zwischen den

beiden Normalen zweier benachbarter Dreiecke. Der Normalenvektor eines Dreiecks zeigt dabei in die Richtung des orientierten Flächenvektors. Er beschreibt also zusammen mit der Orientierung der Fläche eine Rechtsschraube.

4 Oberflächenbasiertes Verfahren nach Peter Liepa

In [1] schlägt Peter Liepa ein oberflächenbasiertes Verfahren zum Füllen von Löchern in Dreiecksnetzen vor. Für die folgenden Überlegungen wollen wir fordern, dass das Dreiecksnetz mannigfaltig, vollständig konsistent orientiert und zusammenhängend ist. D.h. insbesondere, dass keine keine Inseln, singulären Punkte oder nicht mannigfaltigen Kanten auftreten. Das Verfahren zum Füllen der Löcher baut auf verschieden Algorithmen auf, die zusammen in vier Schritten die gewünschte Manipulation des Polygonnetzes, also die Berechnung möglichst guten Flicken für die Löcher, ermöglichen. Der erste Schritt besteht darin, die Löcher auf der Oberfläche zunächst zu lokalisieren. Im zweiten Schritt wird über eine Triangulation ein Flicken für das Loch berechnet. Dieser fügt sich jedoch zunächst nur sehr unbefriedigend dem umgebenden Polygonnetz an. Um die Qualität des Flickens zu verbessern, wird deshalb im dritten Schritt der Flicken verfeinert und im vierten Schritt geglättet und so modelliert, dass er sich der dreidimensionalen Struktur der Umgebung möglichst gut anschmiegt.

4.1 Anforderungen an das Verfahren

Bevor wir uns mit der Realisierung der einzelnen Schritte beschäftigen, wollen wir zunächst festhalten, welche Anforderungen das gesamte Verfahren erfüllen sollte.

1. Das Verfahren sollte in der Lage sein, möglichst selbstständig und automatisch die Löcher in dem Polygonnetz zu finden und adäquat zu füllen. Sollten Nutzeraktionen nötig sein, so sollten diese so gering und einfach wie möglich gehalten werden.

2. Eine geringe Laufzeit ist wünschenswert. Nur so ist im Idealfall ein interaktiver Einsatz des Verfahrens möglich und gewünschte Änderungen am Polygonnetz können in Echtzeit berechnet werden.

3. Der Flicken sollte die größtmögliche Ähnlichkeit mit dem umliegenden Polygonnetz aufweisen. Das heißt insbesondere, dass der Flicken hinsichtlich Dichte und Form bestmöglich mit dem Umgebungsnetz übereinstimmen sollte.

4. Es sollten keine Schwierigkeiten bei komplizierteren, extrem unregelmäßigen oder hochgradig nicht planaren Löchern auftreten.

4.2 Lokalisierung und Klassifizierung von Löchern

Die Lokalisierung eines Loches an sich ist relativ einfach. Sie kann entweder vollständig computergestützt stattfinden. Dann wird aus einer Liste eine Randkante als Startpunkt gewählt und entlang der Orientierung dieser Randkante wird solange ein Pfad aus Randkanten abgelaufen, bis sich eine geschlossene Kurve ergibt. Diese Kurve bildet dann den Rand des zu füllenden Loches. Alternativ kann die Auswahl der Löcher im Polygonnetz durch ein aktives Eingreifen des Benutzers bestimmt werden. Dazu markiert der Benutzer, zum Beispiel mit einem Lasso-Werkzeug, den interessanten Bereich, auf dem der Computer anschließend weitestgehend selbstständig die Löcher findet. Beide Methoden weisen jedoch ein kleines Problem auf. Da die rein topologische Beschreibung verschiedene Arten von Löchern zulässt und dadurch auch Löcher ausgewählt werden, die gar nicht gefüllt werden sollen oder dürfen, ist eine weiter Klassifizierung der Löcher notwendig. Dies gelingt mit dem zu einem Loch assoziierten Gesamtoberflächenwinkel. Dazu definieren wir den Oberflächenwinkel einer Ecke v, die zu einer Randkante gehört durch:

$$\alpha(v) = \sum_{0<i<m-1} \angle(e_i, e_{i+1}).$$

$\{e_1, ..., e_m\}$ sind hierbei die an die Ecke v angrenzenden Kanten. Sie sind in orientierter Reihenfolge um die Ecke v nummeriert. Entsprechend definieren wir den Gesamtoberflächenwinkel für ein Loch durch:

$$\alpha_{tot} = \sum_{0<j<n} \alpha(v_j).$$

$\{v_0, ..., v_n\}$ sind hierbei alle Ecken, die auf dem Rand des betrachteten Loches liegen. Mit dieser Definition können wir die in Abbildung 2 gezeigten Löcher unterscheiden. Von Interesse ist jeweils das gezackte Loch. Die planare Figur, gezeigt in Teilabbildung (a), lässt sich durch eine stetige Transformation über die in (c) gezeigte Figur in die wieder planare Figur in Teilabbildung (e) überführen. Während das gezackte Loch in (a) ein füllenswertes Loch ist, ist es eher unwahrscheinlich, dass das in (e) gezeigte gezackte Loch gefüllt werden soll. Es lässt sich leicht zeigen, dass für die Gesamtoberflächenwinkel der

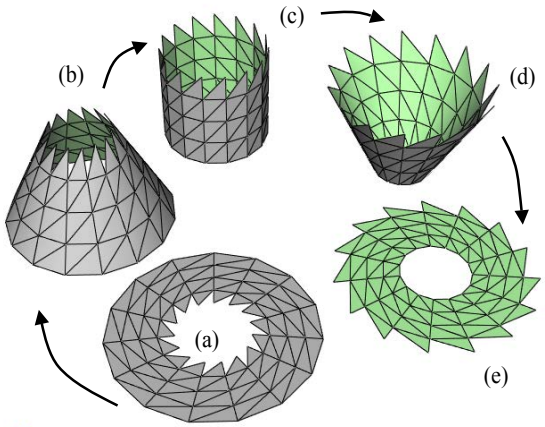

Abbildung 2: Stetige Transformation eines inneren Loches in einen äußeren Rand. Der gezackte Rand entspricht in allen Teilabbildungen ebenso wie der glatte der oben gegeben topologischen Definition eines Loches [1].

Figuren gilt: $\alpha_{tot,(a)} = (n+2)\cdot\pi$, $\alpha_{tot,(c)} = n\cdot\pi$ und $\alpha_{tot,(e)} = (n-2)\cdot\pi$. Damit sehen wir ein, dass ein im Vergleich zur Eckenanzahl n größerer Winkel eher für ein zu füllendes Loch steht als ein kleinerer.

4.3 Triangulation von Löchern in Dreiecksnetzen

Nachdem die Löcher in dem Polygonnetz gefunden wurden und entschieden wurde, ob es sich um ein zu füllendes Loch handelt oder nicht, kann mit dem Füllen begonnen werden. Dazu wird zunächst mittels einer Triangulation ein primitiver Flicken aus Dreiecken für das Loch generiert. Die von Liepa vorgeschlagene Methode geht auf die Zerlegung des Lochs in Dreiecke unter der Minimierung eines Gewichts (z.B. des Gesamtflächeninhalts), wie es Barequet *et al.* 1995 in [3] beschrieben, zurück. Bevor wir den Algorithmus zur Triangulation genauer betrachten können, benötigen wir noch drei wichtige Definitionen. Wir definieren eine *Gewichtungsmenge* L als eine geordnete Menge aus Zahlen oder Zahlentupeln, auf der wir zusätzlich eine Addition einführen. Das Nullelement der Addition bezeichnen wir mit 0_L. Sei weiterhin $V = \{v_0, ..., v_{n-1}\}$ eine Menge von Punkten $v_i \in \mathbb{R}^3$, die das Polygon $(v_0, ..., v_{n-1})$ aufspannen. Dann definieren wir als Gewichtungsfunktion die Abbildung:

$$\Omega : V^3 \to L; (v_i, v_j, v_k) \mapsto w.$$

$w \in L$ nennen wir dabei das Gewicht des Dreiecks (v_i, v_j, v_k). Das Gesamtgewicht für die Triangulation des Teilpolygons $P_{i,j} = (v_i, ..., v_j)$ mit $0 \leq i < j < n$ nennen wir $W_{i,j}$. Nun können wir das Verfahren von Barequet *et al.* benutzen, um ein Dreiecksnetz für das Loch zu berechnen. Der Algorithmus startet mit der kleinsten Polygoneinheit und erhöht schrittweise die Anzahl der Eckpunkte des Polygons.

1. Setze das Gesamtgewicht für alle Kanten zwischen den n Eckpunkten des Loches auf das Nullelement 0_L der Gewichtungsmenge, also $W_{i,i+1} = 0_L$ für $i = 0, 1, ..., (n-2)$. Für alle möglichen Dreiecke (v_i, v_{i+1}, v_{i+2}) aus den Eckpunkten des Loches setze das Gesamtgewicht auf $W_{i,i+2} = \Omega((v_i, v_{i+1}, v_{i+2}))$, $i = 0, 1, ..., (n-3)$. Setze anschließend $j = 2$, wobei $j + 1$ die Anzahl der Ecken des Teilpolygons angibt.

2. Setze $j = j + 1$ und $k = i + j$ für $i = 0, 1, ..., (n - j - 1)$. Bestimme dann die Gesamtgewichte der Teilpolygone $P_{i,k}$ durch $W_{i,k} = \min_{i<m<k}(W_{i,m} + W_{m,k} + \Omega((v_i, v_m, v_k)))$. Mit $\lambda_{i,k} = m$ merken wir uns den Index für das minimale Gesamtgewicht.

3. Für $j < (n - 1)$ wiederhole Schritt 2, ansonsten berechne das Gewicht $W_{0,n-1}$ der gewichtsminimierenden Triangulation.

4. Aus den $\lambda_{i,k}$ wird rekursiv die beste Triangulation, d.h. die Triangulation mit dem kleinsten Gewicht, bestimmt.

Als Freiheit bleibt nun noch das Gewicht und die Gewichtungsfunktion. Von diesen hängt die Qualität des Ergebnisses der Triangulation maßgeblich ab.
Zunächst wollen wir als einfachstes Gewicht die Fläche der entstehenden Dreiecke benutzen. Es gilt damit für die Gewichtungsmenge: $L_{area} = [0, \infty)$, versehen mit der üblichen Addition auf $\mathbb{R}$ und dem Nullelement $0_{L_{area}} = 0$. Die Gewichtungsfunktion ist gegeben durch: $\Omega_{area}((v_i, v_j, v_k)) = \text{area}(\triangle(v_i, v_j, v_k))$. Das Ergebnis dieser Gewichtung ist eine Triangulation mit minimaler Fläche. Gute Ergebnisse können jedoch nur für relativ planare Löcher erzielt werden. Bei Löchern mit Zinnen, wie sie in Abbildung 3 gezeigt sind, d.h. bei Triangulationsflächen, die senkrecht zu Randstücken stehen, entstehen singuläre Punkte, die nicht erwünscht sind (vgl. Abbildung 3 c). Gelöst werden kann dieses Problem, wenn anstelle der reinen Fläche das Tupel aus Diederwinkel und Fläche als Gewicht bei der Triangulation verwendet wird. Die Gewichtungsmenge ist dann gegeben durch $L_{angle} = [0, \pi] \times [9, \infty)$ mit dem neutralen Element $0_{L_{angle}} = (0, 0)$ der Addition.

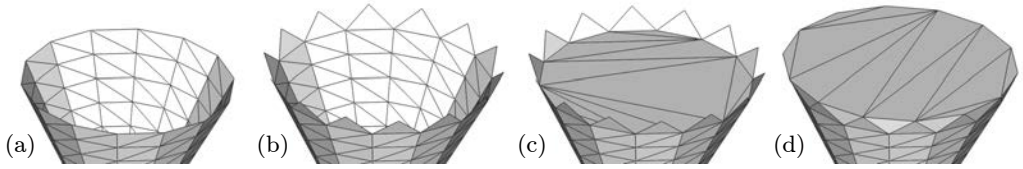

Abbildung 3: Loch ohne und mit Zinne. Bei der Triangulation mit der Fläche der Dreiecke als Gewichtung entstehen singuläre Punkte (c). Durch die Anwendung des Gewichts (Diederwinkel, Fläche) kann dies vermieden werden (d) [1].

Die Ordnung auf dieser Gewichtungsmenge wird definiert durch:

$$(a, b) < (c, d) \Leftrightarrow (a < c) \vee (a = c \wedge b < d) \text{ mit } (a, b), (c, d) \in L_{angle}.$$

Weiter definieren wir die Addition zweier Elemente $(a, b), (c, d) \in L_{angle}$ durch:

$$(a, b) + (c, d) := (\max\{a, c\}, b + d).$$

Mit der Definition von $\mu((v_i, v_j, v_k))$ als den maximalen Diederwinkel zwischen dem Dreieck $\triangle(v_i, v_j, v_k)$ und allen angrenzenden Dreiecken lässt sich die folgende Gewichtungsfunktion konstruieren:

$$\Omega_{angle}((v_i, v_j, v_k)) = (\mu((v_i, v_j, v_k)), \Omega_{area}((v_i, v_j, v_k))).$$

Benutzt man diese Gewichtungsfunktion bei der Triangulation, so werden große Diederwinkel offensichtlich bestraft, was zur Folge hat, dass zunächst eine konvexe Hülle um den Lochrand generiert wird und der dann noch übrig bleibende Rest des Loches unter der Randbedingung minimaler Fläche mit Dreiecken aufgefüllt wird. Diese deutlich besseren Ergebnisse sind in Abbildung 3 d zu sehen. Insbesondere werden singuläre Punkte bei den Zinnen vermeiden.

4.4 Verfeinerung und Entspannung des Polygonnetz-Flickens

Bei der oben beschrieben Triangulation können sehr langgezogene und spitze Dreiecke entstehen und darüber hinaus ist der erzeugte Flicken in Hinblick auf die Dichte im Allgemeinen nicht besonders ähnlich zur Umgebung. Da solche Konstellationen besonders anfällig für numerische Instabilitäten sind, kann beides bei späteren Berechnungen zu Schwierigkeiten oder falschen Ergebnissen führen. Gelöst wird dieses Problem durch die Verfeinerung und anschließende Entspannung der Dreiecke im Polygonnetz-Flicken.

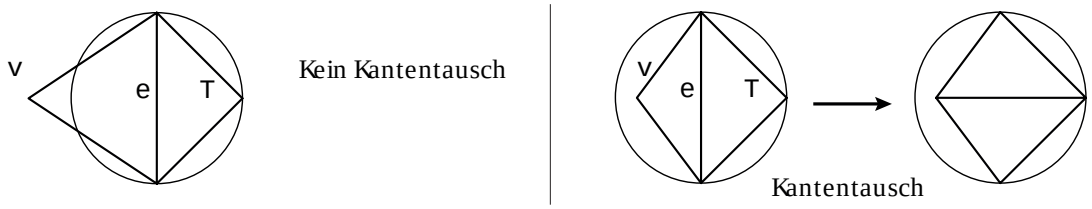

Abbildung 4: Schematische Darstellung des Flip-Algorithmus zur Entspannung innerer Kanten in einem Polygonnetz (nach [1]).

Zur Entspannung der Kanten wird der sogenannte *Flip-Algorithmus* verwendet. Er ist schematisch in Abbildung 4 dargestellt. Es werden je drei Punkte, die zu einem Dreieck T im Dreiecksnetz gehören, ausgewählt. Zu diesen drei Punkten wird der Umkreis bestimmt. Befindet sich kein weiterer Punkt v innerhalb dieses Umkreises, so findet kein Kantentausch statt. Liegt jedoch ein weiterer Punkt v innerhalb des Umkreises, so wird die innere Kante e, wie in der Abbildung gezeigt, getauscht und damit die Anzahl spitzer Dreiecke im Gesamtnetz reduziert.

Der Algorithmus zur Übertragung der Dichte des umgebenden Netzes auf den Flicken geht auf die Idee von Pfeifle und Seidel aus dem Jahr 1996 [5] zurück. Es handelt sich um eine iterative Abfolge aus Verfeinerungen und Entspannungen der inneren Kanten. Zur Regulierung der übertragen Dichte im Flicken wird der sogenannte Dichte-Kontrollparameter α definiert. Empirisch wurde gefunden, dass die Wahl $\alpha = \sqrt{2}$ ein optimales Ergebnis erzeugt, bei dem die Dichte des Flickens mit der Umgebung optisch übereinstimmt. Der Algorithmus selbst besteht aus den folgenden Teilschritten:

1. Für alle Ecken v_i auf dem Lochrand wird das Kantenlängenattribut $\sigma(v_i) = \frac{1}{N}\sum_{j=0}^{N-1} ||v_i - v_j||$ aus den benachbarten Eckpunkten v_j des umgebenden Netzes berechnet. $||\cdot||$ ist dabei die euklidische Abstandsnorm im $\mathbb{R}^3$.

2. Für jedes Dreieck $\triangle(v_i, v_j, v_k)$ wird der Schwerpunkt $v_c = \frac{1}{3}(v_i + v_j + v_k)$ berechnet. Das Kantenlängenattribut für den Schwerpunkt wird auf $\sigma(v_c) = \frac{1}{3}(\sigma(v_i) + \sigma(v_j) + \sigma(v_k))$ gesetzt. Gelten für einen der Eckpunkte v_m, $m = i, j, k$ eines Dreiecks die beiden Ungleichungen $\alpha \cdot ||v_c - v_m|| > \sigma(v_m)$ und $\alpha \cdot ||v_c - v_m|| > \sigma(v_c)$, so wird das Dreieck in die drei kleineren Dreiecke $\triangle(v_i, v_j, v_c)$, $\triangle(v_i, v_c, v_k)$ und $\triangle(v_c, v_j, v_k)$ unterteilt. Anschließend werden die Kanten des ursprünglichen Dreiecks (v_i, v_j), (v_j, v_k) und (v_k, v_i) mit dem Flip-Algorithmus entspannt.

3. Wenn in Schritt 2 keine neuen Dreiecke erzeugt wurden, ist die Verfeinerung vollständig.

4. Es werden alle inneren Kanten des Flickens entspannt.

5. Wenn in Schritt 4 kein Kantenaustausch stattgefunden hat, dann wird bei Schritt 2 fortgefahren, ansonsten wird Schritt 4 wiederholt.

Mit diesem Verfahren gewinnen wir einen Flicken, der in seiner Dichte dem umgebenden Polygonnetz entspricht. Jedoch ist der Flicken im Allgemeinen relativ planar und führt nur schlecht bis garnicht die Form des umgebenden Netzes fort. Abhilfe schafft hier das nun folgende Glätten und Modellieren (engl. fairing) des Flickens.

4.5 Glätten und Modellieren des Polygonnetz-Flickens

Ziel dieses Arbeitsschritts ist es zum einen die Oberfläche des Polygonflickens zu glätten und gleichzeitig die Wölbung des Flickens an den Rand des Loches, bzw. das umgebende Netz anzupassen. Dazu schlägt Liepa ein Verfahren vor, das auf dem von Kobbelt et al. aus dem Jahr 1998 aufbaut [6]. Wieder wird eine Gewichtungsfunktion definiert, dieses Mal jedoch nicht für die Flächen im Polygonnetz, sondern für die Kanten. Sei $\omega : V^2 \to \mathbb{R}$ eine solche Gewichtungsfunktion, die wir nachher genauer spezifizieren. Dann lässt sich der *gewichtete Umbrella-Operator* definieren durch:

$$U_\omega(v) = -v + \frac{1}{\omega(v)} \sum_i \omega(v, v_i) \cdot v_i \quad \text{mit} \quad \omega(v) = \sum_i \omega(v, v_i).$$

Die Summen laufen hierbei gerade über die benachbarten Punkte v_i des betrachteten Punktes v. Eine Glättung des Flickens kann mit dem Umbrella-Operator erzielt werden, wenn alle Punkte v, die zum Inneren des Flickens gehören, durch die Summe $v+U_\omega(v)$ aus ursprünglichem Ortsvektor und Umbrella-Operator ersetzt werden. Setzt man hierbei die Gewichtungsfunktion durch $\omega(v_i, v_j) = 1 \, \forall v_i, v_j$ fest, so erhält man den *gleichmäßigen Umbrella-Operator*, der eine globale Glättung durchführt. Möchte man auch die lokale Krümmung des umgebenden Polygonnetzes auf den Flicken übertragen, so bietet sich der *abstandsabhängige Umbrella-Operator* an, den man durch die nicht-konstante Gewichtungsfunktion $\omega(v_i, v_j) = ||v_i - v_j||^{-1}$ erhält. Für mannigfaltige Flächen ist über die Anpassung der ersten Ableitung, was über die einmalige Anwendung der obigen Umbrella-Operatoren geschieht, eine Anpassung der zweiten Ableitung möglich. Dazu wird der Umbrella-Operator iterativ angewandt, was speziell für die quadratische An-

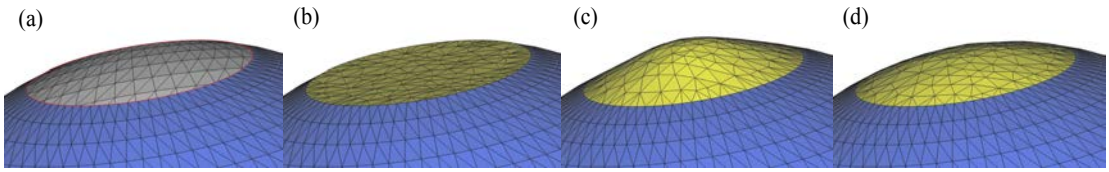

Abbildung 5: Gegenüberstellung unterschiedlicher Glättungen: Kugel mit Loch (Kugelinneres grau) (a), Loch nach reiner Triangulation (b) sowie Glättung mit gleichmäßigem (c) und abstandsabhängigem (d) Umbrella-Operator [1].

wendung durch

$$U_\omega^2(v) = -U_\omega(v) + \frac{1}{\omega(v)} \sum_i \omega(v, v_i) U_\omega(v_i)$$

gegeben ist. Fordert man nun für alle Randpunkte v des Flickens, dass $U_\omega^2(v) = 0$, und erlaubt nur eine Variation der v_i im Inneren des Flickens und nicht im umgebenden Polygonnetz, so erhält man ein Gleichungssystem, das zum Beispiel mit Hilfe der konjugierten Gradienten Methode in den meisten Fällen zuverlässig approximativ gelöst werden kann. Abbildung 5 zeigt deutlich den Unterschied zwischen der Modellierung des Flickens mit dem gleichmäßigen und dem abstandsabhängigen Umbrella-Operator. Bei globaler Glättung mit dem gleichmäßigen Umbrella-Operator wird die Krümmung des Randes mit konstanter Skalierung bis zur Mitte des Flickens getragen. Bei der abstandsabhängigen Modellierung kann das gewünschte Ergebnis eines kontinuierlichen Krümmungsübergangs geschaffen werden.

4.6 Anwendungsbeispiele des Verfahrens

Abbildung 6 und 7 zeigen zwei Anwendungsbeispiele des vierstufigen Verfahrens von Peter Liepa zum Füllen von Löchern in Dreiecksnetzen. Von der ursprünglichen Kugel wurden nahezu 40% des gesamten Polygonnetzes entfernt. Dennoch gelingt eine nahezu perfekte und von der Umgebung ununterscheidbare Berechnung der fehlenden Netzstücke. Deutlich zu erkennen ist, dass nach der reinen Triangulation (Teilabbildung b) zunächst sehr große Dreiecke erzeugt werden, die zum einen sehr spitz und langgestreckt sein können und zum anderen nicht die Dichte der Umgebung aufweisen. Nach der Verfeinerung ist dieses jedoch weitestgehend der Fall. Einzig im Polbereich gibt es kleinere Abweichungen. Beim zweiten Beispiel handelt es sich im Original um den Ha-

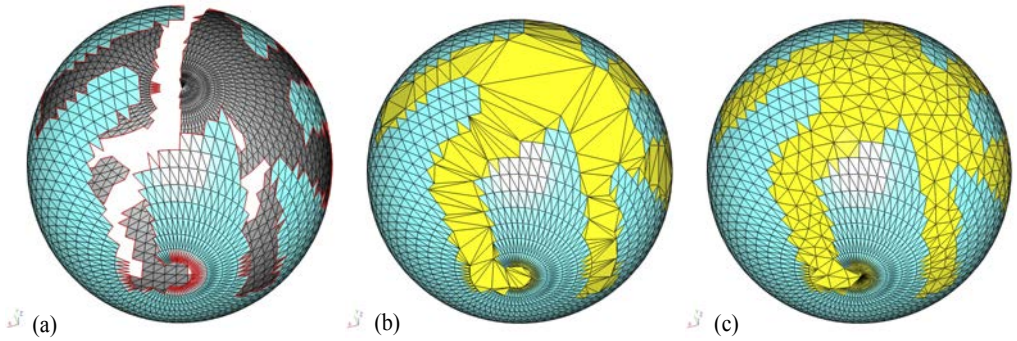

Abbildung 6: Rekonstruktion von Löchern auf einer Kugel: Füllung des Lochs nach reiner Triangulation (b) und nach Verfeinerung, Entspannung der inneren Kanten und Glättung des Flickens (c) [1].

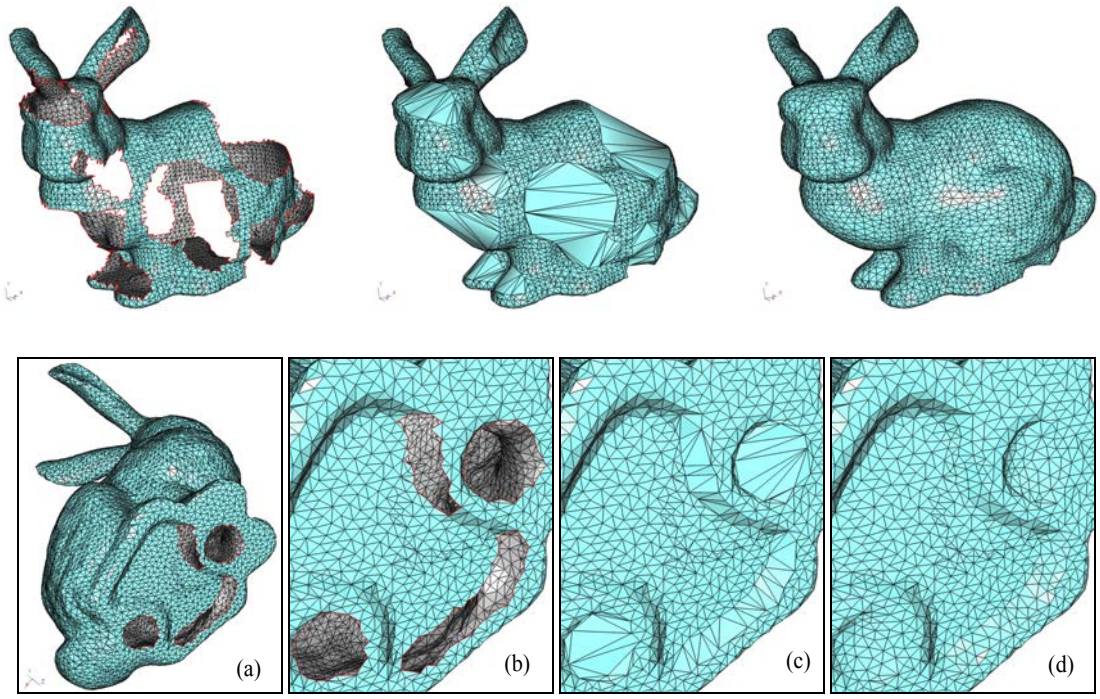

Abbildung 7: Füllen von Löchern in Dreiecksnetzen am Beispiel des Hasens der Stanford Universität. Oben: der komplette Hase nach der reinen Triangulation und der Anwendung des vollständigen Verfahrens. Unten: Vergrößerung der Hasenunterseite mit eingefügten Löchern (b), nach der Triangulation mit dem Tupel aus Diederwinkel und Fläche als Gewicht (c) und nach dem durchlaufen des vollständigen Verfahrens (d) [1].

sen der Stanford Universität. Auch hier wurden große Teile des Polygonnetzes gelöscht und die Rekonstruktion liefert ein erstaunlich gutes Ergebnis. Es ist ebenfalls der Zustand des Modells nach der reinen Triangulation und nach der Verfeinerung des Flickens mit anschließender Modellierung gezeigt. In der vergrößerten Bildreihe zur Unterseite des Hasens ist sehr schön zu erkennen, dass der Algorithmus mit diversen Lochformen umgehen kann und durchweg brauchbare Ergebnisse erzielt.

5 Diskussion und Potential der Methode nach Liepa

Im vorangegangen Abschnitt haben wir gesehen, dass das Verfahren von Liepa sehr gut in der Lage ist auch anspruchsvolle Löcher zufriedenstellend zu füllen - natürlich um so präziser, je kleiner und strukturell einfacher die Löcher sind. Problematisch bei dem ganzen Verfahren gestaltet sich jedoch die Laufzeit. Allein durch die Triangulation bedingt, liegt die Laufzeit für das gesamte Verfahren bei $\mathcal{O}(n^3)$. Damit ist die Methode zumindest für sehr große Modelle nur schwer bis garnicht bei Echtzeitberechnungen einsetzbar. Es gibt jedoch neue Ansätze von Wei Zhao, Shumin Gao und Hongwei Lin aus dem Jahr 2007 [7], die ohne die Neuberechnung der Triangulation in jedem Schritt auskommen und somit deutlich bessere Laufzeiten erzielen können. Auf der anderen Seite ist die Methode von Liepa leicht auf beliebige Polygonnetze erweiterbar. Auch die Behandlung von Polygonnetzen mit Inseln stellt keine größeren Probleme da, da vor der Anwendung des Verfahrens rücken zwischen den Inseln und dem übrigen Polygonnetz erstellt werden können. Abgesehen von der Aufbereitung von Polygonnetzen für die Weiterverarbeitung zum Beispiel in physikalischen Berechnungen eignet sich das Verfahren von Liepa auch dazu Rauschen auf der Oberfläche eines Modells zu beseitigen oder sogar ganze Bereiche durch eine einfachere Formgebung zu ersetzen. Dazu wird einfach der unerwünschte Bereich des Polygonnetzes verworfen und die entstandenen Löcher mit dem Verfahren von Liepa wieder aufgefüllt.
Das Verfahren von Liepa hat insgesamt ein hohes Potential: Sowohl in der Anwendung als Methode zum Entfernen von Artefakten in Polygonnetzen, als auch zur Vereinfachung derselben. Die einzige Voraussetzung ist, dass bei großen Modellen die Berechnungen aufgrund der nicht optimalen Laufzeit des Verfahrens nicht zeitkritisch seien dürfen.

Literatur

[1] Peter Liepa. *Filling holes in meshes.* In Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP '03). Eurographics Association, Aire-la-Ville, Switzerland, p. 200-205, Switzerland, 2003.

[2] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing.* AK Peters, 2010.

[3] Gill Barequet und Micha Sharir. *Filling Gaps in the Boundary of a Polyhedron.* Computer-Aided Geometric Design, 12(2): p. 207-229, März 1995.

[4] G. Barequet, M. Dickerson, and D. Eppstein. *On triangulating three-dimensional polygons.* Computational Geometry: Theory and Applications, 10(3): p. 155-170, Juni 1998.

[5] R.Pfeifle und H.-P. Seidel. *Triangular B-Splines for Blending and Filling of Polygonal Holes.* Proc. Graphics Interface '96. p. 186-193. Morgan Kaufmann Publishers, 1996.

[6] L. Kobbelt, S. Campagna, J. Vorsatz, H.-P. Seidel. *Interactive Multi-Resolution Modeling on Arbitray Meshes.* SIGGRAPH 98 Conference Proceedings, 1998.

[7] W. Zhao, S. Gao, H. Lin. *A robust hole-filling algorithm for triangular mesh.* The Visual Computer: International Journal of Computer Graphics 23, 12 (November 2007), p. 987-997, 2007.