3D-Meshes aus medizinischen Volumendaten

Eine Methode zur Generierung von konformen Hexaeder-Meshes aus gelabelten Volumendaten

Sascha Zelzer, Hans-Peter Meinzer

Abteilung für Medizinische und Biologische Informatik, DKFZ Heidelberg
s.zelzer@dkfz.de

Kurzfassung. Diese Arbeit beschreibt eine template-basierte Methode zur Erzeugung von adaptiven Hexaeder-Meshes aus Volumendaten, welche komplizierte konkave Strukturen aufweisen können. Es wird ein vollständiger Satz von Templates generiert der es erlaubt, die Ränder konkaver Regionen feiner zu zerlegen als angrenzende Bereiche und somit die Gesamtzahl an Hexaeder verringert. Der Algorithmus arbeitet mit beliebigen gelabelten Volumendaten und erzeugt ein adaptives, konformes, reines Hexaeder-Mesh.

1 Einleitung

Simulationen von physikalischen Vorgängen aller Art wurden in letzter Zeit durch die rasant steigenden Rechnerkapazitäten immer realistischer bzw. erst machbar. Letzeres trifft vor allem auf die Medizin zu, die aufgrund der komplexen physikalischen Prozesse oft entweder stark vereinfachte Modelle erstellen, oder extrem lange Rechenzeiten in Kauf nehmen muss. Viele Methoden zur numerischen Behandlung dieser Modelle beruhen auf einer Diskretisierung der räumlichen Strukturen, welche in der Medizin (Gefäßbäume, Hirngewebe, ...) oft hoch komplex sind. Ein direkter Weg, den Rechenaufwand zu minimieren, liegt in der Minimierung der Anzahl von Zellen, die eine solche Diskretisierung bilden.

Die drei am häufigsten benutzen Mesh-Typen in $\mathbb{R}^3$ sind Tetraeder-, Hexaeder- und Hybrid-Meshes. Die Anforderungen an solche Gitter sind normalerweise derart, dass sie wichtige geometrische Merkmale erhalten, indem sie in diesen Bereichen fein genug sind. Gleichzeitig sollen sie aber in homogenen Bereichen nicht unnötig fein sein, um die Zellenanzahl nicht zu erhöhen. Zur Zeit bieten Tetraeder-Meshes die größte Flexibilität um komplexe Daten vollautomatisch zu modellieren [1]. Sie benötigen aber meist eine große Anzahl an Zellen.

Eine Schwierigkeit bei der Erzeugung eines adaptiven Hexaeder-Meshes aus komplexen Strukturen besteht in der Verfeinerung konkaver Regionen. In [2, 3] wurde die Directional Refinement Methode vorgestellt, die konkave Bereiche verfeinern kann, ohne eine große Zahl neuer Zellen einzuführen. Der Nachteil dieser Methode ist allerdings deren schlechte Skalierbarkeit. In unserem Ansatz verwenden wir die Methode in [2], um einen vollständigen Satz von Templates zu erstellen, welcher die Vorteile der directional refinement Methode besitzt und in template-basierten Algorithmen verwendet werden kann.

2 Methoden

2.1 Der Algorithmus

Unser Algorithmus gehört zur Klasse der Octree-basierten Template-Techniken und besteht aus den folgenden drei Schritten:

1. Erzeuge einen initialen Octree mit den gewünschten Eigenschaften
2. Identifiziere die zu verfeinernden Oktanten
3. Ersetze die Oktanten aus Schritt 2 durch passende Templates

Die Eingabe für den Algorithmus kann als Abbildung $f : D \to L$ beschrieben werden, mit D eine nicht-leere beschränkte Teilmenge von $\mathbb{R}^3$ und L eine endliche Menge, die die verschiedenen Labels (Strukturen) repräsentiert. Die Ausgabe ist ein konformes Hexaeder-Mesh.

Um den initialen Octree in Schritt 1 zu erzeugen, benutzen wir eine Kombination von vorhandenen Methoden. Die Identifizierung der Oktanten in Schritt 2 und der neue Satz von Templates bilden unseren neuen Beitrag.

Schritt 1: Erzeugung des initialen Octrees Zuerst erzeugen wir ausgehend von f einen uniformen, signierten Octree. Wenn D unendlich ist, müssen wir den tiefsten Octree-Level vorgeben. Ist D endlich (normalerweise besteht D aus den Punkten eines homogenen, rechtlinigen Gitters), dann wird der tiefste Octree-Level so gewählt, dass alle geometrischen Merkmale von D erhalten bleiben.

Wir benutzen nun den Vereinfachungsalgorithmus in [4], um den Octree so weit wie möglich zu vergröbern, ohne die Topologie der Grenzflächen zu verändern. Um das directional refinement Verfahren später anwenden zu können, muss der Octree streng 1-balanciert sein:

Definition 1. *Ein d-dimensionaler Baum heißt k-balanciert, $k \in [0, d)$, dann und nur dann wenn sich kein Blatt q mit Level $l(q)$ eine m-dimensionale Fläche, $m \in [k, d)$, mit einem anderen Blatt q' mit $l(q') > l(q) + 1$ teilt.*

Mit dieser Definition ist ein 2-balancierter Octree also ausbalanciert an den Flächen, ein 1-balancierter Octree ausbalanciert an den Flächen und Kanten und ein 0-balancierter Octree ausbalanciert an den Flächen, Kanten und Ecken.

Definition 2. *Ein d-dimensionaler Baum heißt streng k-balanciert, $k \in [0, d)$, dann und nur dann wenn (i) für alle Knoten gilt, dass entweder alle oder keines der Kinder wiederum Kinder hat und (ii) Definition 1 gilt.*

Schritt 2: Verfeinerungskonfiguration Wir müssen nun diejenigen Oktanten des streng 1-balancierten Octrees finden, die in Schritt 3 durch passende Templates ersetzt werden. Dazu betrachten wir die Oktanten mit Blättern als Kinder und nennen sie Pseudoblätter. Diese Pseudoblätter werden in Schritt 3 ersetzt, daher bestimmen wir für jedes Pseudoblatt q eine eindeutige Verfeinerungskonfiguration wie folgt:

Betrachte alle Flächen-, Kanten- und Ecken-Nachbarn von q im selben Octree-Level. Ein solcher Nachbar q' ist entweder ein Blatt, ein Pseudoblatt oder ein Oktant, dessen Kinder alle wieder Kinder besitzen (siehe Definition 2). Nur im letzten Fall markieren wir die Ecken von q, die auch gleichzeitig Ecken von q' sind. Sei nun M_q die Menge der markierten Ecken von q. Die Verfeinerungskonfiguration von q ist dann durch

$$T_q = \{m \in M_q \,|\, \exists m' \in M_q, m' \neq m \text{ , sodass } \overline{m\,m'} \text{ eine Kante von } q \text{ ist } \}$$

gegeben. Wir streichen also alle markierten Ecken, die nicht mit einer anderen markierten Ecke durch eine Kante von q verbunden werden können (der Grund liegt in der directional refinement Methode, welche nur Flächen und Kanten verfeinert). Die Menge T_q identifiziert dann in eindeutiger Weise das passende Template, siehe Tabelle 1.

Schritt 3: Templates Das Problem mit existierenden Templates besteht in deren Inflexibilität bezüglich der Verfeinerung von konkaven Bereichen (siehe Abb. 1 (a) und (b)). In medizinischen Datensätzen mit komplexen Strukturen resultiert dies in eine enorme Steigerung der Zellenanzahl.

Wir konstruieren nun unseren neuen Satz von Templates mit Hilfe des directional refinement Verfahrens. Für jede mögliche Verfeinerungskonfiguration eines Pseudoblattes (siehe Tabelle 1, Spalte 1) verfeinern wir die Kinder wie in [2] beschrieben. Die Resultate sind unsere neuen Templates (Tabelle 1, Spalte 2), welche den Vorteil haben keine hängenden Flächen zu besitzen. Diese neuen Templates können nun verwendet werden um die Pseudoblätter aus Schritt 2 zu ersetzen und ein konformes Gitter zu erzeugen.

2.2 Evaluierung

Um unsere Templates zu evaluieren, haben wir sie mit zwei anderen Sätzen von gängigen Templates aus [5] und [6] verglichen. Als Datensätze verwendeten

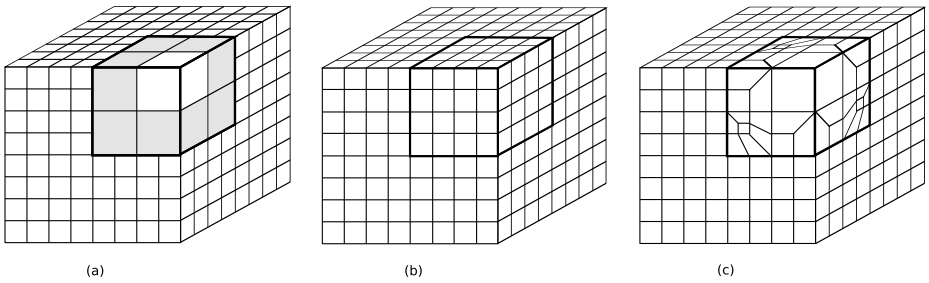
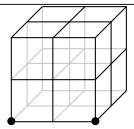
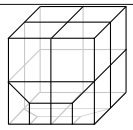
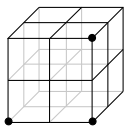
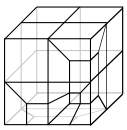
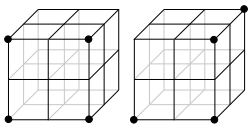
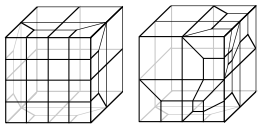
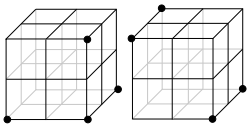
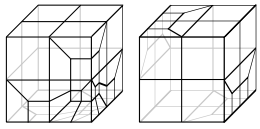
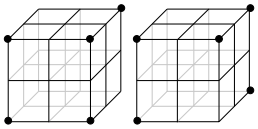
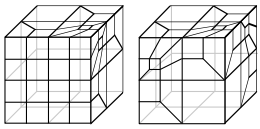
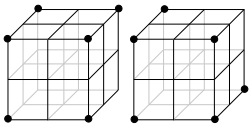
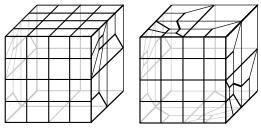
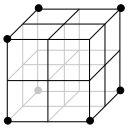
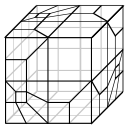


Abb. 1. Konkave Verfeinerung. (a) Die grauen Oktanten müssen verfeinert werden, um ein konformes Gitter zu erzeugen (b) Die Verwendung bestehender Templates führt zu übermäßiger Verfeinerung (c) Verfeinerung der Kinder des Pseudoblatts mit passenden Templates.

Tabelle 1. Links, mögliche Verfeinerungskonfigurationen für Pseudoblätter. Mitte, die dazugehörigen Templates, konstruiert mit dem directional refinement Verfahren (die inneren Ecken und Kanten wurden nicht visualisiert).

Verfeinerungskonfiguration (T_q)	Template	# Vertices / # Hexaeder
		38 / 14
		54 / 24
		67 / 32 70 / 34
		77 / 40 49 / 20
		90 / 48 86 / 44
		110 / 62 113 / 64
		129 / 74

wir eine segmentierte Leber mit einem Gefäß und Tumor (65x41x27 Voxel), ein segmentiertes Herz (Perikard und die vier Kammern, 286x261x158 Voxel) und ein segmentiertes Gehirn (15 Strukturen, 181x217x181 Voxel).

3 Resultate

Tabelle 2 gibt eine Übersicht über die Anzahl der Punkte und Hexaeder in den erzeugten Meshes für die drei Datensätze. Die Verfahren in [5] und [6] erzeugen für die ersten beiden Datensätze (Leber und Herz) neunmal bzw. ca. fünfmal so

Tabelle 2. Die Anzahl der Vertices und Zellen eines Hexaeder-Meshes erzeugt aus gelabelten Datensätzen mit drei verschiedenen Methoden.

# Vert/# Zellen	Unsere Methode	Methode von [5]	Methode von [6]
Leber	14000 / 12000	112000 / 112000	49000 / 49000
Herz	498000 / 494000	4504000 / 4503000	2882000 / 2880000
Gehirn	1532000 / 1526000	4384000 / 4383000	3798000 / 3798000

viele Zellen im Vergleich zu unserem Verfahren. Im letzten Datensatz sind es 2.9 bzw. 2.5 mal so viele.

4 Diskussion

Mit der Einführung unserer neuen Templates sind wir in der Lage, beliebige konkave Strukturen am Rand feiner aufzulösen und gleichzeitig die Anzahl der neuen Zellen gering zu halten. Vergleicht man die Zahlen in Tabelle 2 wird klar, dass die zuvor vorhandenen Templates nicht ausreichten, um medizinische Strukturen mit einer vertretbaren Anzahl von Zellen zu diskretisieren. Insbesondere ist das Verfahren in [5] überhaupt nicht - und das Verfahren in [6] nur eingeschränkt - in der Lage, konkave Regionen lokal zu verfeinern. Mit zunehmender Komplexität der geometrischen Strukturen erzeugen die bisherigen Ansätze fast homogene Gitter, die bis auf Voxelgröße verfeinert wurden. Unser Ansatz hingegen reduziert in solchen Fällen die Anzahl der Zellen und Vertices auf weniger als die Hälfte.

Zukünftige Arbeiten umfassen eine systematische Untersuchung der Mesh-Qualität, eine Anpassung der Knoten des Meshes an die internen Grenzflächen und die Anwendung eines Glättungsverfahrens. Eine solche Glättung würde die Qualität der Diskretisierung weiter erhöhen und somit zu einer erhöhten numerischen Genauigkeit bei Simulation führen.

Literaturverzeichnis

1. Pons JP, Ségonne E, et al JDB. High-quality consistent meshing of multi-label datasets. Inf Process Med Imaging. 2007;20:198–210.
2. Schneiders R. Octree-based hexahedral mesh generation. Int J Comput Geom Appl. 2000;10(4):383–398.
3. Tchon KF, Dompierre J, Camarero R. Automated refinement of conformal quadrilateral and hexahedral meshes. Int J Numer Methods Eng. 2004;59:1539–1562.
4. Ju T, Losasso F, Schaefer S, et al. Dual contouring of hermite data. Procs SIGGRAPH. 2002; p. 339–346.
5. Schneiders R. Refining quadrilateral and hexahedral element meshes. Procs Int Conf Numerical Grid Generation in Computational Field Simulations. 1996; p. 699–708. Available from: citeseer.ist.psu.edu/schneiders96refining.html.
6. Zhang H, Zhao G. Adaptive hexahedral mesh generation based on local domain curvature and thickness using a modified grid-based method. Finite Elem Anal Des. 2007;43(9):691–704.