Diskrete Datenmodelle in der CAD-CAM-Kopplung

Dipl.-Ing. Martin Erler

1 Einleitung

Volumenmodelle finden seit mehreren Jahrzehnten in verschiedenen Bereichen der Produktionstechnik Anwendung. Die bekanntesten Anwendungen sind CAD, CAM und Materialabtragsimulationen. Bei den Abtragsimulationen haben sich 3-Dexelmodelle als Standardwerkstückmodell etabliert. CAD-Anwendung nutzen einen Mix aus verschiedene Beschreibungsmodellen. Neben Oberflächenmodellen kommen vor allem parametrierte Volumenprimitive zum Einsatz. CAM-Anwendungen nutzen (neben den Abtragsimulationen) vorwiegend Volumenmodelle, die auf einer Beschreibung der Oberfläche beruhen.

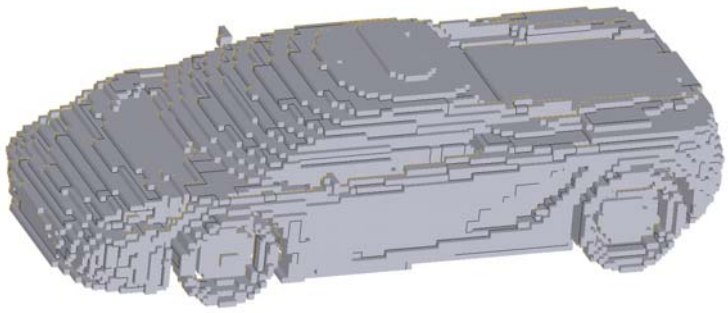

Abbildung 1: Volumenmodell eines Autos

Gegenstand der aktuellen Forschung ist der Versuch, diskrete Volumenmodelle für CAM-Anwendung nutzbar zu machen /WIR-14/. Als diskrete Volumenmodelle bezeichnet man solche Modelle, die den Raum, der beschrieben wird, in mindestens einer Dimension diskretisieren und das Volumen anstatt der Oberfläche beschreiben (Abb. 1). Die Grenzen zu den BHV (Bounding-Hierarchical-Volumes) sind dabei fließend. Die bekanntesten Vertreter sind Voxel (Akronym aus Volumen und Pixel).

Voxel unterteilen den Raum in gleichgroße Würfel, die in einem räumlichen Gitter angeordnet sind. Aufgrund ihrer primitiven Form, lassen sich sehr einfach beschreiben und auch gängige mathematische und boolesche Operationen, die in der Computergrafik nötig sind, lassen sich schnell und einfach an ihnen durchführen. Ihr großer Nachteil besteht in der kubisch steigenden Zahl der Voxel zur Auflösung des Modells und den nicht glatten Oberflächen. Um Voxelmodelle für CAM-Anwendung nutzbar zu machen, wurden verschiedene Konvertierungsmethoden und Speicherstrukturen untersucht sowie Einsatzmöglichkeiten.

2 Voxel und Speicherstrukturen

Im Folgenden werden drei verschiedene Strukturen zur Speicherung von Voxeln untersucht. Die Speicherung in

- einem Gitter,
- einem Octree und
- einem Quadtree.

2.1 Das Gitter

Die Voxel werden dabei lückenlos an einem räumlichen Gitter angeordnet. Die Gitterplätze können über einen Index angesprochen werden (Abb. 2). Dies entspricht

einer dreidimensionalen Liste. Die Speicherung von Daten in Listen ermöglicht schnelle Zugriffe auf die Daten. Hinzu kommt bei der Speicherung von Daten mit räumlichem Bezug (wie z. B. Voxeln), dass Nachbarschaft im Gitter identisch mit räumlicher Nachbarschaft ist. Das Gitter entspricht damit einer direkten Repräsentation der Daten. Nachteilig ist die mit der Auflösung kubisch wachsende Datenmenge. Um ein 100mm x 100mm x 100mm großes Bauteil mit Voxeln von 0,1mm Kantenlänge darzustellen, werden eine Milliarde Voxel benötigt. Zudem ist die Suche in Listen ungünstig, da stets die ganze Liste durchgegangen werden muss, um ein gesuchtes Element zu finden. Es gibt jedoch Möglichkeiten die Suchzeiten in Listen zu reduzieren (z. B. durch Sortierung).

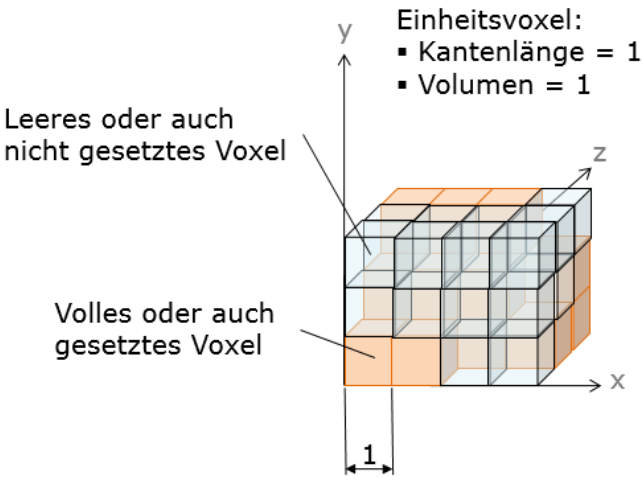

Abbildung 2: Voxel in einer Gitterstruktur

2.2 Der Octree

Es handelt sich um eine selbstähnliche Baumstruktur, die den würfelförmigen Raum in acht gleich große Würfel unterteilt (Abb. 3). Ein Würfel (auch Node oder Knoten) unterteilt sich wiederum in acht Nodes. Benutzt man diese Struktur zur Speicherung von Voxelmodellen, wird ein Node so lange unterteilt, wie seine Kinder unterschiedliche Zustände (gesetzt oder nicht gesetzt) haben.

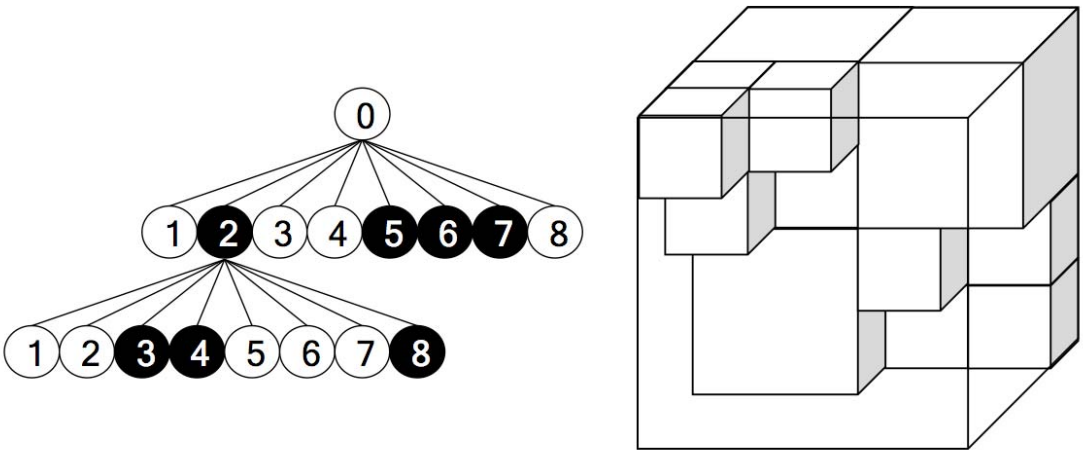

Abbildung 3: Struktur (li.) und exemplarische Nutzung für Voxel (re.) des Octreemodells

Haben alle Kinder eines Nodes den gleichen Zustand, kann man dies darstellen, indem man dem Node diesen Zustand zuweist und die Kinder löscht. Dies führt zu einer effizienteren Speichernutzung gegenüber einer Listenstruktur. Weitere Vorteile sind:

- Selbstähnlichkeit – ermöglicht standardisierte geometrische Berechnungen
- Baumstruktur – ermöglicht sehr schnelle Suchen in den Daten
- Universalität – ermöglicht vielseitige Nutzung als Datencontainer.

2.3 Der Quadtree

Der Quadtree ist das zweidimensionale Pendant zum Octree. Er ist bei der Verarbeitung und Speicherung von zweidimensionalen Daten weit verbreiten (z. B. in der Bildbearbeitung). Die Beschränkung auf eine Ebene verhindert eine direkte Verwendung für dreidimensionale Daten. Um ihn dennoch nutzen zu können, wird der Raum in einer Richtung diskretisiert und einer Liste zugeordnet. An den Listenplätzen werden anschließend je ein Quadtree hinterlegt, welche identische Abmaße besitzen (Abb. 4).

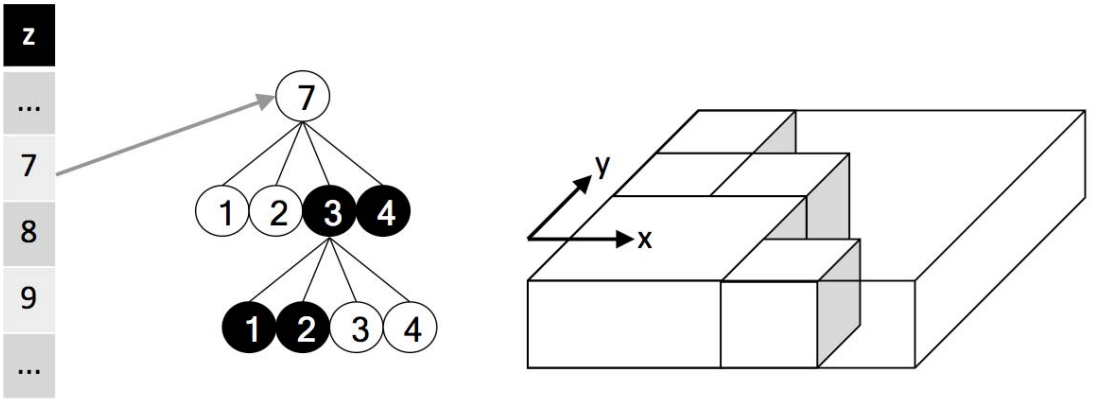

Abbildung 4: Quadtree-Struktur in Listen

Es entsteht eine Quadtreeliste. Diese besitzt eine Vorzugsebene, in der der Quadtree liegt. Im Vergleich zum Octree fällt die Speicherersparnis gegenüber dem Gitter geringer aus. Auch die Generierung dauert aufgrund der höheren Anzahl der Knoten länger. Dennoch können mit dieser Struktur flache dreidimensionale Objekte besser repräsentiert werden (Abb. 5).

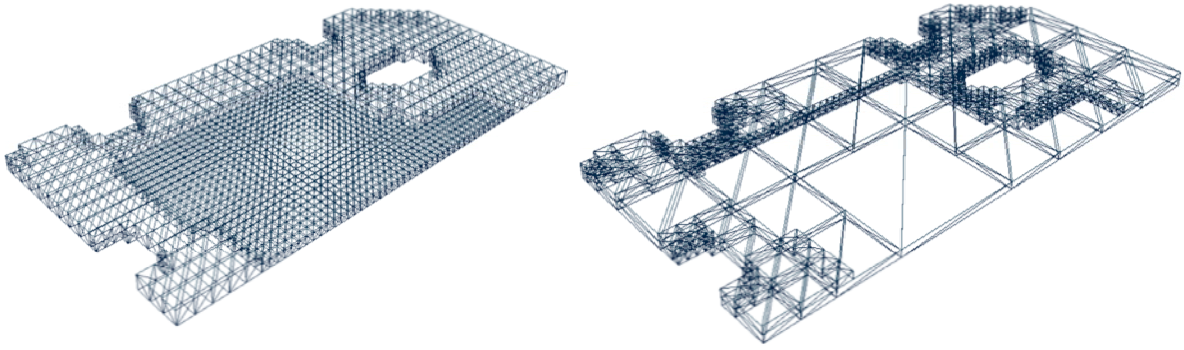

Abbildung 5: flaches Bauteil als Octreemodell (li.) und Quadtreemodell (re.)

3 Konvertierungsmethoden

Um die Modelle zu nutzen, müssen die Eingangsdaten in das jeweilige Format konvertiert werden. Als Ausgangsbasis wurde das STL-Format gewählt, da es

- weit verbreitetet ist,
- einfach zu implementieren ist und
- von allen großen CAD-Programmen geschrieben und gelesen werden kann.

Es wurden darauf aufbauend verschiedene Methoden zur Generierung der Modelle für die drei Strukturen untersucht.

3.1 Methoden zur Generierung einer Gitterstruktur

Scanline-Methode

Die erste Konvertierungsmethode erfolgte mittels des Scanline-Algorithmus, bei dem das Oberflächenflächenmodell (aus der STL-Datei) entsprechend der Gittergröße des Voxelgitters in Ebenen zerlegt wird, welche mit den Dreiecken des Oberflächenmodells geschnitten werden (man könnte es auch Scanplane-Algorithmus nennen). Man erhält einen oder mehrere geschlossenene Polygonzüge je Ebene. Auf jede Ebene wird nun wiederum der Scanline-Algorithmus anwendet. Als Zeilenabstand wird wieder der Gitterabstand des Voxelgitters verwendet. Die so erhaltenen Strecken kann man aufgrund des identischen Gitterabstandes wie im Voxelgitter als Anfangs- und Endpunkt einer Voxelreihe interpretieren. Diese Konvertierungsmethode hat den Vorteil, dass sie innen außen liegende Voxel mit behandelt.

Randvoxel-Füll-Methode

Der Algorithmus läuft zweistufig ab:

- Finden der Randvoxel und
- Füllen des Innenraums.

Um die Randvoxel zu finden, welche die Oberfläche des Modells repräsentieren, werden die Voxel mit den Dreiecken der STL-Datei auf Schnitt geprüft. Das Füllen des Innenraums erfolgt nach dem Prinzip des Seedings. Es wird ein Voxel gesucht, der garantiert nicht innen liegt (z.B. einer am Rand des Modells). Von diesem ausgehend wird die Außenregion wachsen gelassen. Wenn das Wachstum abgeschlossen ist, sind alle verbliebenen Voxel nur noch solche, die innen liegen. Das Problem, dass eventuell noch andere Außenregionen existieren könnten, die nicht mit der Saatregion verbunden sind, wird durch eine Maßnahme und eine Annahme behoben. Da es sich um ein Werkstückmodell aus massiven Material handeln soll, existieren keine Hohlräume im Körper. Daraus folgt, dass alle Außenregionen miteinander verbunden sein müssen. Um dies auch im Voxelmodell zu gewährleisten, wird das Voxelgitter so angelegt, dass es in jeder Richtung mindestens eine Voxelschicht größer ist, als die Ausgangs-STL-Datei.

3.2 Methoden zur Generierung einer Octree-Struktur

Voxel-zu-Octree

Bei dieser Methode wird zuerst ein gitterbasiertes Voxelmodell aufgebaut und anschließend in eine Octree-Struktur überführt. Ausgehend von einem existierenden Voxelmodell wird dazu der Octree so angelegt, dass die Elemente in der untersten Ebene genau der Größe eines Voxels entsprechen. Der Octree wird maximal unterteilt und die Voxel mit den Elementen der untersten Ebene verglichen. Über die Lage im Baum lässt sich den Elementen eine eindeutige Vergleichslage im Voxelgitter zuordnen, über die das entsprechende Schwestervoxel im Voxelgitter identifiziert werden kann. Ist dieses gesetzt, wird auch das Octree-Element gesetzt. Nach kompletten Durchlaufen des Baumes wird der Octree reduziert. Hierzu werden Nodes, die nur Kinder mit gleichen Zustand (gesetzt oder nicht gesetzt) haben, in eben diesen Zustand gesetzt und die Kinder gelöscht.

Absteigender Aufbau

Der Algorithmus ist zweistufig. Zuerst werden die Randnodes gefunden und gesetzt, anschließend die innenliegenden Voxel gefunden und gesetzt. Das finden der

Randnodes erfolgt ausgehend von einem Octree ohne Nodes – also nur dem Rootnode. Dieser wird mit den Dreiecken auf Schnitt geprüft, bis ein Schnitt gefunden wird. Daraufhin wird der Node unterteilt und für die Kindernodes genauso verfahren. Liegt ein Node in der untersten Ebene (Vorgabewert) und es wird ein Schnitt gefunden, so wird er nicht unterteilt, sondern gesetzt. Die so erhaltene Hüllschicht wird mittels eines Seedingalgorithmus verfüllt. Aufgrund der Baumstruktur bedeutet Nachbarschaft im Baum für zwei Nodes nicht zwangsläufig, dass sie auch räumliche Nachbarn sind. Daher müssen zuerst für alle Nodes die räumlichen Nachbarn ermittelt werden. Anschließend wird er Octree noch reduziert.

Absteigender Aufbau mit GPU-Support

Aufgrund der Vielzahl der Schnittprüfungen – gerade bei STL-Dateien mit mehreren Hunderttausend Dreiecken – wurde der erste Schritt des Algorithmus auf die Grafikkarte portiert /SEE-14/. Hierzu war es erforderlich die Threads vorher festzulegen. Die direkt absteigende Reihenfolge (auf Node folgen dessen KIndernodes und deren Kindernodes usw.) wurde in eine ebenenweise Stückelung geändert. Es werden alle Schnitttests für alle Nodes eines Levels (Ebene) auf die Grafikkarte geschoben. Die zurückerhaltenen Ergebnisse werden ausgewertet, der Octree entsprechend verändert und die nächsttiefere Ebene wieder auf der Grafikkarte gerechnet.

Absteigender Aufbau mit Dreiecksortierung

Auch dieser Algorithmus beginnt beim Rootnode. Es werden alle Dreiecke der STL-Datei dem Rootnode zugeordnet. Nun werden alle Kinder des Rootnodes auf Schnitt mit den Dreiecken geprüft. Wird ein Schnitt gefunden, wird das Dreieck diesem Node zugeordnet. Ein Dreieck kann mehreren Nodes zugeordnet sein. Ist es der erste gefundene Schnitt für diese Gruppe Nodes, wird der Elternnode unterteilt. Ist die Prüfung für alle Kinder eines Nodes abgeschlossen, werden die Dreieckzuordnungen in ihm gelöscht. So wird für alle Nodes verfahren. Durch die absteigende Reihenfolge in der Baumstruktur braucht der Baum nur einmal durchlaufen zu werden. Am Ende wurde die Dreieckliste in eine Baumstruktur einsortiert, was den positiven Nebeneffekt hat, dass die Anzahl der durchzuführenden Schnitttests mit der Ebenentiefe abnimmt.

3.3 Methoden zur Generierung einer Quadtreestruktur

Voxel-zu-Quadtree

Die Konvertierung erfolgt äquivalent zur Voxel-zu-Octree-Methode. Durch die zweidimensionale Speicherstruktur des Quadtrees, beschränkt sich diese Vorgehensweise jedoch auf die jeweilige Gitterebene. Die Konvertierung wird also für jede Gitterebene des Voxelgitters durchgeführt und für jede dieser Ebenen ein Quadtree angelegt.

Scanline-Methode

Die Methode funktioniert identisch zu der Konvertierung in die Gitterstruktur. Es wird lediglich auf den Aufbau des gesamten Gitters verzichtet und stattdessen direkt für jede gerasterte Ebene ein Quadtree angelegt.

Absteigender Aufbau mit Dreiecksortierung

Die Methode ist identisch zum Vorgehen beim Octree, findet jedoch nur in der Ebene statt. Hierdurch ist eine Vorsortierung der Dreiecke nach ihrem Z-Wert sinnvoll. Die Ausrichtung der Hauptgitterrichtungen beeinflusst das Ergebnis.

Octree-zu-Quadtree

Basierend auf einer existierenden Octreestruktur werden die enthaltenen Octreenodes in Quadtreenodes zerlegt. Dies geschieht absteigend bei dem Rootnode beginnend. Hierzu wird dieser zuerst in *n* Quadtrees zerlegt wird, wobei $n = 2^z$ mit z als höchster Level des Octrees. Anschließend werden dessen Octreenodes zerlegt und die so entstandenen Quadtreenodes den im Elternknoten liegenden Quadtrees zugeordnet.

4 Ergebnisse

Die drei Speicherstrukturen wurden hinsichtlich ihrer Eignung zur Repräsentation von Bauteilen verglichen /VEI-14/. Die Eignung wird hier hauptsächlich durch vier Faktoren bestimmt:

- Speicherbedarf: Wieviel (Arbeits-)Speicher wird zum Vorhalten des Modells benötigt und wie entwickelt sich der Bedarf über der Auflösung?
- Aufbauzeit: Wielange dauert der einmalige Aufbau der Struktur aus einer STL-Datei und wie verhält sich diese Zeit bei verschiedenen Auflösungen?
- Zugriffszeiten: Wie schnell können Daten in der Struktur gefunden werden und wie hängen die Zugriffszeiten mit der Auflösung zusammen?
- Änderungskosten: Wie hoch sind die Aufwände zum Ändern einzelner Daten im Modell?

Tabelle 1 stellt die Ergebnisse qualitativ dar. Die vier Faktoren werden nach fixem und variablem Verhalten unterschieden. Fix bedeutet die grundsätzliche Eignung und variabel, wie sich diese Eignung mit steigender Auflösung verändert.

Tabelle 1: Übersicht Eignung der Volumenmodelle

		Speicherbedarf		Aufbauzeit		Zugriffszeit		Änderungskosten	
		fix	variabel	fix	variabel	fix	variabel	fix	variabel
Gitter	Scanline	- -	- -	+	-	- -	- -	+ +	+ +
	Randvoxel-Füll	- -	- -	o	- -				
Octree	Voxel-zu-Octree	- -	- -	o	o	+ +	+ +	-	- -
	Absteigender Aufbau ohne GPU	-	- -	-	-				
	Absteigender Aufbau mit GPU	-	-	+ +	o				
	Absteigender Aufbau mit Sortierung	+ +	+ +	o	+ +				
Quadtree	Voxel-zu-Quadtree	- -	- -	o	-	+	+	o	-
	Scanline	-	-	-	-				
	Absteigender Aufbau mit Sortierung	+	+ +	+	+				
	Octree-zu-Quadtree	o	+	o	+				

Literatur

/SEE-14/ von See, H.: Untersuchung technischer Ansätze zum GPU-basierten Multithreading von Algorithmen ausgewählter diskreter Volumenmodelle. Belegarbeit, Professur Formgebende Fertigungsverfahren, TU Dresden 2014

/WIR-14/ Wirth, M.: Entwicklung eines Ansatzes zur Segmentierung eines Octree-Volumenmodells nach Fertigungsfeatures. Diplomarbeit TU Dresden 2014

/VEI-14/ Veith, J.: Planung und Durchführung der Testphase eines Softwaremoduls für die Arbeitsvorbereitung in der spanenden Fertigung. Belegarbeit, Professur Formgebende Fertigungsverfahren, TU Dresden 2014