Proseminar

Ausgewählte Themen der Computergraphik

Marching Cubes

Johannes Elsing
Institut für Informatik, Albert-Ludwigs-Universität Freiburg

10. Juni 2025

1 Einführung

Der **Marching-Cubes-Algorithmus** ist ein Verfahren aus der 3D-Computergrafik zur Erzeugung sogenannter Isoflächen innerhalb volumetrischer Datensätze. Entwickelt wurde er 1987 von *William E. Lorensen* und *Harvey E. Cline* im Rahmen ihrer Forschung bei General Electric. Der Algorithmus zeichnet sich durch seine Robustheit und Effizienz aus und gilt als einer der bedeutendsten Ansätze zur Extraktion von Isoflächen. Ziel der Entwicklung war es, medizinische Bilddaten – wie sie beispielsweise in der *Computertomografie (CT)*, der *Magnetresonanztomografie (MRT)* oder der *Single-Photon-Emissionscomputertomografie (SPECT)* entstehen – effizient und dreidimensional visualisierbar zu machen. Die erste offizielle Veröffentlichung des Algorithmus erfolgte 1987 in der renommierten Fachzeitschrift *Computer Graphics* im Rahmen der *SIGGRAPH*-Konferenz. Seitdem hat der Marching-Cubes-Algorithmus breite Anwendung in der medizinischen Bildgebung sowie in der wissenschaftlichen Visualisierung gefunden und gilt heute als Meilenstein in der computergestützten 3D-Datenverarbeitung. Als Eingabedaten erhält der Algorithmus kubische Gitterdaten und einen benutzerdefinierten Isowert und liefert als Ausgabe eine Isofläche sowie Normalen an den Eckpunkten der durch die Gitterdaten definierten Würfel.

Abbildung 1: Drache, verwendete Gitterauflösung bei Marching Cubes: 64^3

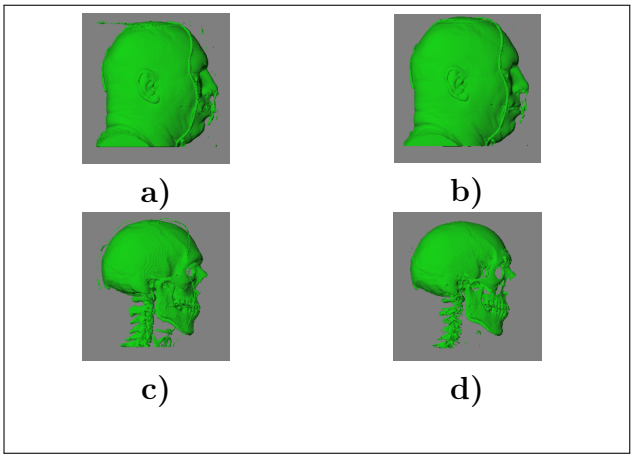


Abbildung 2: Volumenrendering eines männlichen Kopfdatensatzes mit dem Marching-Cubes-Algorithmus bei verschiedenen Isowerten: (a) 30; (b) 50; (c) 75; (d) 100.

Der Algorithmus verarbeitet den gesamten 3D-Datensatz in einem systematischen Durchlauf. Das Volumen wird in kleine Würfel unterteilt, wodurch das Problem auf kleinere Teilprobleme reduziert wird, was dem Divide-and-Conquer-Prinzip folgt und somit starke Optimierungen ermöglicht. Diese Würfel – Volumenelemente oder "Voxel" genannt – repräsentieren diskrete Werte an bestimmten XYZ-Koordinaten im Raum. Für jeden dieser Würfel berechnet der Algorithmus, ob und wie eine triangulierte Isofläche innerhalb dieses Volumenelements verläuft. Abhängig von der Verteilung der Skalardaten an den acht Ecken eines Würfels entsteht ein Polygon innerhalb des Würfels, das ein Teil der Gesamtfläche ist. Nach der Abarbeitung eines Würfels wird zum nächsten übergegangen, bis das gesamte interessierende Volumen durchlaufen wurde. Aus den einzelnen Fragmenten der Polygonflächen lässt sich in einem abschließenden Fusionsschritt eine zusammenhängende Isofläche rekonstruieren.

2 Anwendungen

Isoflächen sind in vielen Disziplinen wichtig, da sie dreidimensionale Strukturen in kontinuierlichen Skalarfeldern durch Flächen mit einem festen Iso- oder Schwellenwert visualisieren. Ihre Anwendungen reichen von der numerischen Strömungsmechanik, wo Druck- oder Geschwindigkeitsfelder dargestellt werden, bis zur medizinischen Diagnostik, etwa zur Visualisierung von Organen oder Tumoren in CT- und MRT-Daten. Neben realen Messdaten können auch künstlich erzeugte Skalarfelder, etwa mittels Perlin oder Simplex Noise, Grundlage für die Isoflächengenerierung sein, womit sich natürliche Strukturen wie Gelände oder Wolken modellieren lassen. In all diesen Kontexten liefern Isoflächen nicht nur visuelle Einblicke, sondern auch quantitative Auswertungen wie Volumen- oder Formbestimmung.

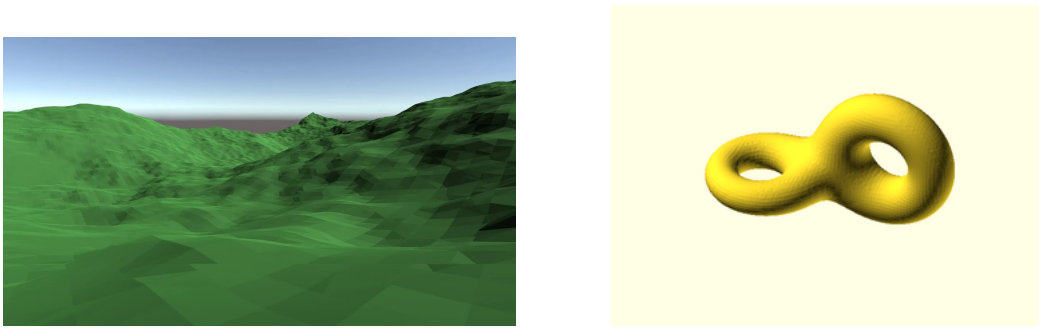


Abbildung 3: Metaballs Visualisierung

Diese künstlich erzeugten Felder können mithilfe des klassischen Marching-Cubes-Algorithmus in polygonale 3D-Oberflächen überführt werden, wodurch sie direkt in Renderings oder Simulationen einfließen können. Auch sogenannte Metaballs lassen sich auf diese Weise visualisieren. Hierbei handelt es sich um eine Technik, bei der volumetrische Daten durch die additive Überlagerung mehrerer skalierter Feldfunktionen entstehen, die jeweils um einen zentralen Punkt – meist eine Kugel – definiert sind. Durch die Summation dieser Felder entsteht ein kontinuierliches Skalarfeld, das an bestimmten Schwellenwerten organisch wirkende Oberflächen aufweist.

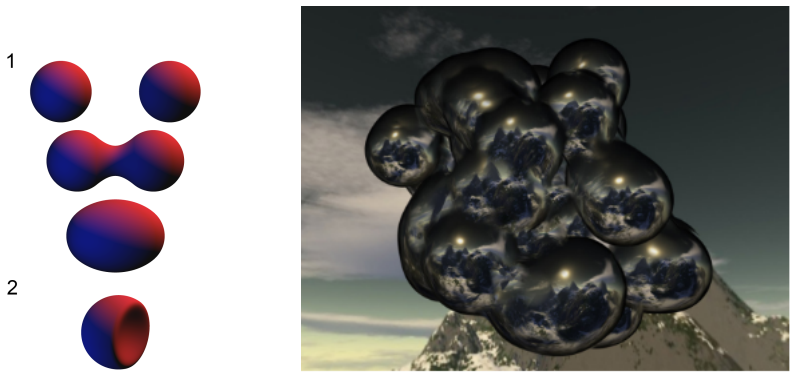


Abbildung 4: Metaballs Visualisierung

3 Funktionsweise

Um das Prinzip des Marching-Cubes-Algorithmus besser zu verstehen, bietet es sich an, zunächst das zweidimensionale Pendant, den Marching-Squares-Algorithmus, zu betrachten. Ausgehend von einem auf ein regelmäßiges 2D-Gitter projizierten Datensatz definiert ein Isowert die Isofläche. Nun wird für jede Zelle des Gitters geprüft, ob der Isowert an den vier Ecken des Quadrats überschritten wird. Jede Ecke kann entweder als "aktiv" (größer als der Isowert) oder als "inaktiv" (kleiner als der Isowert) klassifiziert werden. Daraus ergeben sich insgesamt $2^4 = 16$ mögliche Kombinationen, die sich aber aufgrund von Symmetrien auf nur vier grundlegende Konfigurationen reduzieren lassen:

- Fall 1: alle Ecken größer / kleiner
- Fall 2: genau eine Ecke größer / kleiner
- Fall 3: genau zwei Ecken mit gemeinsamer Kante größer / kleiner
- Fall 4: genau zwei gegenüberliegende Ecken größer / kleiner

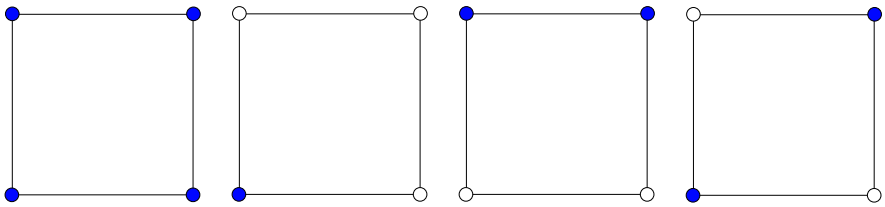


Abbildung 5: Die vier grundlegenden Fälle im Marching-Squares-Algorithmus

Diese vier grundlegenden Fälle bestimmen, wie die Isofläche innerhalb eines Quadrats verläuft. Der Marching-Squares-Algorithmus nutzt diese Reduktion, um schnell und effizient zu entscheiden, wie das Quadrat in Bezug auf den Isowert geschnitten wird. Diese Vorüberlegungen sind sinnvoll, da sie sich direkt auf den MC-Algorithmus übertragen lassen. Der Marching-Cubes-Algorithmus arbeitet auf einem dreidimensionalen Skalarfeld

$$f : D \subset \mathbb{R}^3 \to \mathbb{R}$$

und extrahiert eine Isofläche für einen gegebenen Isowert ρ_{iso}. Dabei wird das Skalarfeld f auf einem Gitter H abgebildet, wobei jeder Gitterpunkt v_i ein Skalarwert $f(v_i)$ zugewiesen bekommt. Für jeden Würfel im Gitter, der durch acht benachbarte Gitterpunkte definiert wird, wird jeder der acht Eckpunkte des Würfels mit dem Isowert ρ_{iso} verglichen. Ein Eckpunkt v_i wird als "aktiv" klassifiziert, wenn $f(v_i) \geq \rho_{iso}$ und als "nicht aktiv", wenn $f(v_i) < \rho_{iso}$. Dies bedeutet, dass für jede der acht Ecken eines Würfels $2^8 = 256$ mögliche Kombinationen von positiven und negativen Vorzeichen existieren. Die binäre Darstellung dieser Kombinationen (mit 1 für "aktiv" und 0 für "nicht aktiv") ergibt eine 8-Bit-Zahl, die als Index in eine Lookup-Tabelle dient, die für jede dieser Kombinationen eine vordefinierte Triangulierung liefert. Diese Triangulierung stellt die Isofläche dar, die den Isowert ρ_{iso} in dem jeweiligen Würfel repräsentiert.

4 Interpolation

Beim Marching-Cubes-Algorithmus wird eine dreidimensionale Funktion $f : \mathbb{R}^3 \to \mathbb{R}$ auf einem regelmäßigen Gitter (Voxeldaten) ausgewertet. Ziel ist es, die Isofläche für einen festen Isowert ρ_{iso} zu rekonstruieren, also die Menge aller Punkte $p \in \mathbb{R}^3$, für die $f(p) = \rho_{iso}$ gilt. Eine naive Implementierung betrachtet nur, ob ein Voxelwert $s_i = f(v_i)$ kleiner oder größer als ρ_{iso} ist. Dabei werden die Dreiecke an festen Stellen der Würfelkanten gesetzt. Das führt zu blockartigen Geometrien, zu einer ungenauen Annäherung und treppenartigen Artefakten. Dies macht die lineare Interpolation zur exakten Schnittpunktbestimmung relevant. Zwei benachbarte Voxelpunkte $v_0, v_1 \in \mathbb{R}^3$ sowie ihre Funktionswerte $s_0 = f(v_0)$ und $s_1 = f(v_1)$ werden betrachtet. Angenommen, s_0 und s_1 liegen auf verschiedenen Seiten von ρ_{iso}, also $(s_0 - \rho_{iso}) \cdot (s_1 - \rho_{iso}) < 0$, dann existiert ein Schnittpunkt p auf der Kante zwischen v_0 und v_1. Gesucht ist ein Interpolationsparameter $t \in [0, 1]$, sodass

$$f((1-t) \cdot v_0 + t \cdot v_1) = \rho_{iso}$$

gilt. Unter der Annahme, dass f zwischen v_0 und v_1 linear variiert, ergibt sich die lineare Interpolation:

$$s(t) = (1-t) \cdot s_0 + t \cdot s_1$$

Setze $s(t) = \rho_{iso}$ und löse nach t auf:

$$(1-t) \cdot s_0 + t \cdot s_1 = \rho_{iso}$$

$$s_0 - t \cdot s_0 + t \cdot s_1 = \rho_{iso}$$

$$t \cdot (s_1 - s_0) = \rho_{iso} - s_0$$

$$t = \frac{\rho_{iso} - s_0}{s_1 - s_0}$$

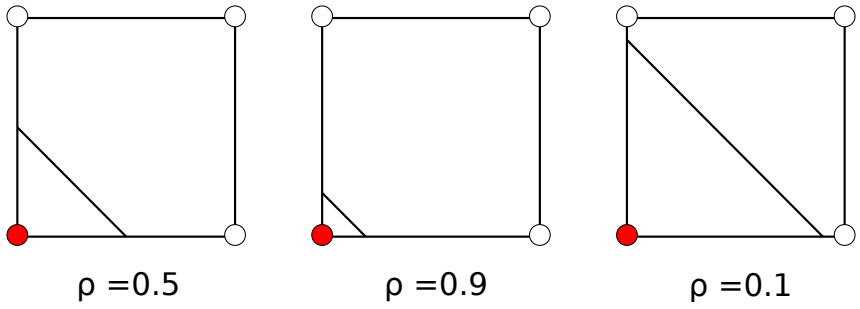
Damit lautet der Interpolationsparameter:

$$t = \frac{\rho_{iso} - s_0}{s_1 - s_0}$$

Der exakte Schnittpunkt p auf der Kante ist dann:

$$p = (1-t) \cdot v_0 + t \cdot v_1$$

Das bedeutet geometrisch für $t = 0$, dass der Punkt genau bei v_0 ist und für $t = 1$, dass der Punkt genau bei v_1 ist. Im Fall $0 < t < 1$ liegt der Punkt dazwischen.



5 Berechnung von Oberflächennormalen

Für realistische Beleuchtung, wie sie beispielsweise im Phong-Beleuchtungsmodell eingesetzt wird, sind korrekte Oberflächennormalen von zentraler Bedeutung. Sie ermöglichen es, Lichtreflexionen und Schattierungen physikalisch plausibel darzustellen und tragen entscheidend zur visuellen Qualität des generierten 3D-Modells bei. Die Normalen dienen dabei als Orientierung der Fläche im Raum und beeinflussen maßgeblich, wie Licht auf der Oberfläche wirkt. Im Marching-Cubes-Algorithmus werden diese Normalen aus dem Skalarfeld, also der zugrunde liegenden Dichtefunktion $D(x, y, z)$, abgeleitet. Da die Isoflächen entlang von Flächen konstanter Dichte verlaufen, sind sie per Definition orthogonal zum Gradienten ∇D dieser Funktion. Die Oberflächennormale $\mathbf{n}$ an einem Punkt auf der Isofläche ergibt sich somit als der *negative* Gradient der Dichtefunktion:

$$\mathbf{n} = -\nabla D = -\begin{pmatrix} \frac{\partial D}{\partial x} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial z} \end{pmatrix}$$

Da die Dichtefunktion nur diskret auf einem regelmäßigen Gitter gegeben ist, wird der Gradient numerisch approximiert, typischerweise mit der zentralen Differenzenmethode:

$$\begin{aligned} G_x(i,j,k) &= \frac{D(i+1,j,k) - D(i-1,j,k)}{2\Delta x} \\ G_y(i,j,k) &= \frac{D(i,j+1,k) - D(i,j-1,k)}{2\Delta y} \\ G_z(i,j,k) &= \frac{D(i,j,k+1) - D(i,j,k-1)}{2\Delta z} \\ \mathbf{n} &= -\begin{pmatrix} G_x \\ G_y \\ G_z \end{pmatrix}, \quad \hat{\mathbf{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|} \end{aligned}$$

Die Normale wird anschließend auf Einheitslänge normiert, um sie direkt für Beleuchtungsberechnungen nutzen zu können. Da der Gradient in Richtung steigender Dichtewerte zeigt, wird für die Oberflächennormale der negative Gradient verwendet, da dieser nach außen, also vom Objekt weg zeigt. Die so berechneten, normalisierten Gradientenvektoren bilden die Grundlage für akkurate Beleuchtungsmodelle und verbessern maßgeblich die visuelle Qualität der Isoflächen.

6 Lookup-Tabelle

Ein zentrales Element des Marching-Cubes-Algorithmus ist die Lookup-Tabelle, die für jede mögliche Konfiguration der Eckwerte eines Voxels angibt, wie die lokale Triangulierung der Isofläche erfolgt. Dabei verzeichnet die Tabelle für jede Kombination die Kanten, an denen die Isofläche interpoliert wird, und definiert die daraus entstehenden Dreiecke. Durch diese Vorgehensweise entfällt die manuelle Topologiebestimmung für jede Konfiguration, was eine konstante Laufzeit pro Voxel ermöglicht und den Algorithmus sowohl effizient als auch einfach implementierbar macht.

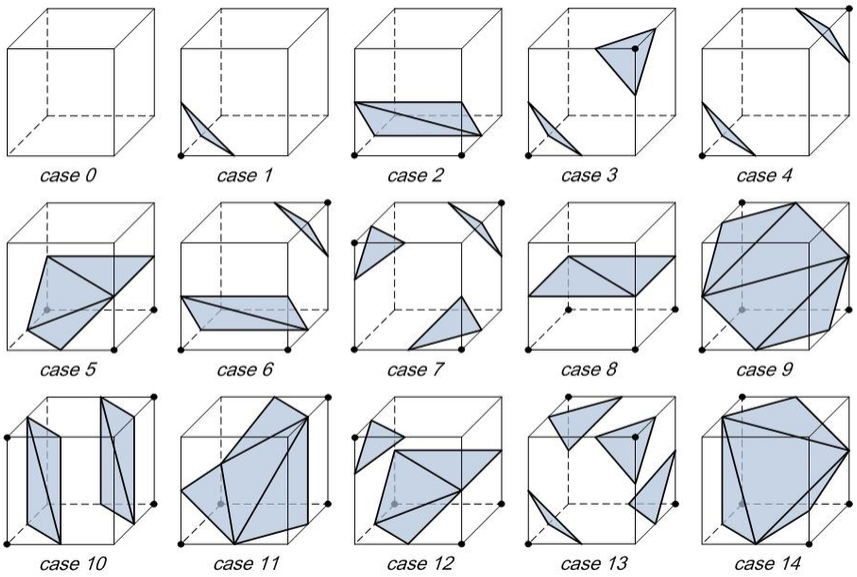


Abbildung 6: Flächenkonfigurationen im Marching-Cubes-Algorithmus.

7 Optimierung

Der Marching-Cubes-Algorithmus lässt sich in vielerlei Hinsicht optimieren. Eine zentrale Eigenschaft, die dies ermöglicht, ist seine hohe Parallelisierbarkeit: Jede Zelle eines 3D-Gitters kann unabhängig von den anderen verarbeitet werden. Diese Eigenschaft macht den Algorithmus *embarrassingly parallel* und somit hervorragend für die Nutzung auf modernen Mehrkernprozessoren und GPUs geeignet. Ein wesentlicher Nachteil des naiven Algorithmus besteht darin, dass er alle Zellen des Gitters sequenziell durchläuft, unabhängig davon, ob in der jeweiligen Zelle überhaupt eine Isofläche verläuft. Dies führt dazu, dass auch leere Zellen verarbeitet werden, was unnötigen Rechenaufwand verursacht und die Laufzeit des Algorithmus verlängert. Um diese ineffiziente Vorgehensweise zu verbessern, können räumliche Datenstrukturen wie etwa Oktalbäume oder k-d-Bäume verwendet werden. Diese Strukturen erlauben es, nur jene Zellen zu identifizieren und zu prozessieren, in denen tatsächlich eine Isofläche vorhanden ist. Dadurch kann die Anzahl der zu betrachtenden Zellen erheblich reduziert und die Gesamtlaufzeit optimiert werden.

8 Ausblick

Der Marching-Cubes-Algorithmus wurde über Jahrzehnte hinweg weiterentwickelt und ist nach wie vor ein aktives und relevantes Forschungsgebiet. Ein bekanntes Problem des ursprünglichen Verfahrens sind topologische Mehrdeutigkeiten, die bei bestimmten Konfigurationen der Voxeldaten auftreten können – insbesondere dann, wenn die Isofläche mehrere mögliche Verläufe innerhalb eines Würfels zulässt. Solche Mehrdeutigkeiten können zu Löchern, fehlerhaften Verbindungen oder inkorrekter Topologie in der rekonstruierten Oberfläche führen. Bereits 1991 wurden mit dem Marching-Tetrahedra-Algorithmus und dem Asymptotic Decider zwei bedeutende Erweiterungen vorgestellt, die die im ursprünglichen Verfahren auftretenden Ambiguitäten adressieren. Im Jahr 2004 wurde der Dual Marching Cubes-Algorithmus von Scott Schaefer und Joe Warren vorgestellt. Der Marching-Tetrahedra-Algorithmus vermeidet Mehrdeutigkeiten, indem er den Würfel in Tetraeder unterteilt und so eindeutigere Entscheidungen bei der Flächenbildung ermöglicht. Im Jahr 2021 wurde ein wissenschaftlicher Beitrag zu **Neural Marching Cubes** veröffentlicht, der auf neuronalen Netzen basiert. Zahlreiche weitere Erweiterungen wurden vorgeschlagen, um die Topologieerhaltung zu verbessern, scharfe Kanten zu rekonstruieren oder die Netzqualität zu erhöhen. Im Gegensatz zu klassischen Ansätzen berücksichtigt dieser Algorithmus Kohärenz und Abhängigkeiten zwischen benachbarten Würfeln. Insbesondere zeigt das zugrunde liegende Netzwerk die Fähigkeit, scharfe Merkmale wie Kanten und Ecken zu rekonstruieren – ein langjähriges Problem klassischer MC-Varianten.

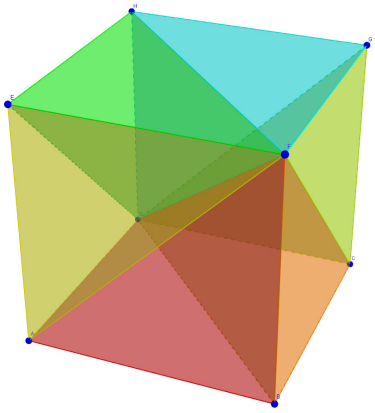


Abbildung 7: Marching-Tetrahedra Würfel

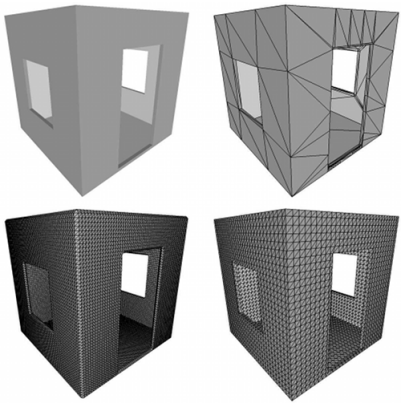


Abbildung 8: Dual-Marching-Cubes Beispiel

9 Zusammenfassung

Der Marching-Cubes-Algorithmus, entwickelt 1985 von Lorensen und Cline, extrahiert Isoflächen aus 3D-Skalardaten und ist ein Meilenstein der Computergrafik, insbesondere in der medizinischen Bildverarbeitung. Das Volumen wird in Würfel unterteilt; abhängig von den Skalaren an den Ecken wird durch Lookup-Tabellen eine triangulierte Oberfläche erzeugt. Durch lineare Interpolation werden glattere Oberflächen erzielt. Der Algorithmus ist hochgradig parallelisierbar und kann mit räumlichen Datenstrukturen optimiert werden. Erweiterungen wie Marching Tetrahedra, Asymptotic Decider und moderne Ansätze wie Neural Marching Cubes verbessern Topologieerhaltung und Oberflächenqualität. Ein weiterer Vorteil liegt in seiner Einfachheit und Robustheit, die ihn bis heute in zahlreichen Anwendungen relevant macht. Trotz seines Alters bleibt er ein grundlegender Baustein in der Flächenrekonstruktion aus Volumendaten.

Quellenverzeichnis

[1] Scott Schaefer und Joe Warren.
Dual Marching Cubes: Primal Contouring of Dual Grids.
In: *Computer Graphics Forum*, Vol. 24, Nr. 2, 2005, S. 311–318.
The Eurographics Association and Blackwell Publishing Ltd.
DOI: 10.1111/j.1467-8659.2005.00843.x

[2] Zhiqin Chen und Hao Zhang.
Neural Marching Cubes.
In: *ACM Transactions on Graphics (TOG)*, Vol. 40, Nr. 6, 2021, Artikel Nr. 251, S. 1–15.
DOI: 10.1145/3478513.3480518

Weitere Quellen

- https://en.wikipedia.org/wiki/Marching_cubes
- https://en.wikipedia.org/wiki/Marching_squares
- https://journal-bcs.springeropen.com/articles/10.1186/s13173-019-0086-6
- https://www.researchgate.net/publication/257797896_Marching_cubes_technique_for_volumetric_visualization_accelerated_with_graphics_processing_units
- https://research.nvidia.com/labs/toronto-ai/flexicubes/

Bildquellen

- https://www.researchgate.net/figure/llustration-of-the-15-basic-cases-of-the-marching fig1_257797896
- https://www.researchgate.net/figure/Type-of-surface-combinations-for-the-marching-cub fig2_282209849
- https://www.researchgate.net/figure/Rendering-results-of-each-dataset-generated-by-th fig6_257797896
- https://commons.wikimedia.org/wiki/File:Marching_tetrahedrons.png
- https://research.nvidia.com/labs/toronto-ai/flexicubes/assets/wireframe/gargoyle100K_mc.png
- https://users.polytech.unice.fr/~lingrand/MarchingCubes/resources/marchingS.gif
- https://webspace.science.uu.nl/~telea001/uploads/PAPERS/GEMS/paper_original.pdf
- https://en.wikipedia.org/wiki/Metaballs#/media/File:Metaballs.png