3D Rendering

# Marching Cubes:

Der Algorithmus wurde entwickelt, um **3D-Oberflächenmodelle** aus medizinischen Daten wie CT, MRT oder SPECT zu erstellen. Diese Modelle repräsentieren Bereiche mit konstanter Dichte, wie z. B. Gewebe oder Organe.

Unterteilung in Würfel:

* Der 3D-Datensatz (z. B. CT-Daten) wird in kleine, gleichmäßige Würfel (engl. "Cubes") zerlegt, die jeweils von acht Punkten (Eckdaten) definiert werden.

Dichteprüfung an den Ecken:

* Der Algorithmus prüft an den acht Ecken jedes Würfels, ob der Dichtewert (z. B. Grauwert des Bildes) über oder unter einer bestimmten Schwelle liegt.
* Abhängig von der Kombination der Ergebnisse wird eine vordefinierte "Falltabelle" (Case Table) verwendet, um zu entscheiden, wie Dreiecke im Würfel erstellt werden.

Erzeugung der Dreiecke:

* Der Algorithmus erzeugt Dreiecke, um die Oberfläche im Inneren des Würfels darzustellen. Die Position der Dreieckspunkte wird durch **lineare Interpolation** berechnet (d. h., basierend auf den Datenpunkten zwischen den Ecken des Würfels).

Interslice Connectivity:

* Um die Konsistenz der erzeugten Oberflächen zwischen den verschiedenen Schichten der Bilddaten sicherzustellen, verwendet der Algorithmus eine *divide-and-conquer*-Methode.

Schattierung der Modelle:

* Die **Gradienten** (Richtungen der größten Intensitätsänderungen in den Daten) werden berechnet und normalisiert, um sie für Schattierungen zu nutzen. Das sorgt für eine realistischere Darstellung der Oberflächen.

### **Ergebnisse und Vorteile**

* Die erzeugten Oberflächenmodelle zeichnen sich durch ihre **Detailtreue** aus, da:
  + die Konnektivität zwischen den Schichten berücksichtigt wird,
  + die Oberflächendaten genau berechnet werden,
  + die Gradienteninformationen für Schattierungen eingebaut sind.

### Weitere Informationen:

* Der Marching Cubes-Algorithmus wandelt **3D-Daten** (z. B. CT- oder MRT-Scans) in ein Modell um, das aus Polygonen (zumeist Dreiecken) besteht. Diese Polygone bilden die Oberflächen der Bereiche mit konstanter Dichte.
* Die generierten Polygonmodelle sind für gängige **Grafik-Rendering-Techniken** geeignet. Das bedeutet, dass:
* Sie leicht in Software (wie OpenGL, Unity, etc.) angezeigt werden können.
* Da die Modelle mit Standard-Grafikalgorithmen kompatibel sind, lassen sie sich leicht in verschiedene **3D-Visualisierungsumgebungen** oder Anwendungen integrieren, ohne dass spezifische Anpassungen nötig sind.

## Vier Schritte der 3D-Verarbeitung in der Medizin

1. Datenakquisition (Data Acquisition):

* Dies ist der erste Schritt, der mit medizinischen Geräten durchgeführt wird.
* Diese Geräte (z. B. **CT, MRT oder SPECT**) erfassen Messdaten und wandeln sie in **2D-Schichtbilder** um.

1. Bildverarbeitung (Image Processing):

* Im zweiten Schritt werden Bildverarbeitungsalgorithmen angewendet.
* Diese helfen dabei, Strukturen im 3D-Datensatz zu finden oder das Originalbild zu filtern.

1. **Oberflächenkonstruktion (Surface Construction):**

* Ziel ist es, aus den 3D-Daten ein **Oberflächenmodell** zu erstellen.
* Dies wird oft mit **Voxeln** (3D-Pixeln) oder **Polygonen** (z. B. Dreiecken) dargestellt.
* Nutzer können die gewünschte Oberfläche auswählen, indem sie einen **Dichtewert** angeben.
* Optionale Bearbeitungen: Schneiden oder Deckeln der Oberfläche.

1. Anzeige (Display)

* Der letzte Schritt ist die Visualisierung der Oberfläche.
* Hier kommen Techniken wie **Ray Casting** (Strahlprojektion) oder **Schattierung** (Tiefenschattierung, Farbgebung) zum Einsatz.

## Schritte des Marching-Cubes-Algorithmus:

**1.Einlesen der Daten:**  
Vier Schichten (2D-Slices) von medizinischen Bilddaten werden in den Arbeitsspeicher geladen. Diese Schichten enthalten Informationen über die Dichte des Gewebes an verschiedenen Punkten.

**2.Würfelbildung:**  
Der Algorithmus bildet kleine Würfel aus je vier Punkten in zwei benachbarten Schichten. Jeder Würfel hat 8 Ecken.

**3.Indexberechnung:**  
Jede Ecke des Würfels erhält eine **0** oder **1**, abhängig davon, ob die Dichte an diesem Punkt über oder unter einem definierten Wert liegt (z. B. dem Wert, der eine Knochenoberfläche beschreibt). So entsteht ein **Index**, der den Zustand des Würfels beschreibt.

**4.Oberflächenfindung:**  
Mit dem berechneten Index wird in einer Tabelle nachgeschlagen, wie die Oberfläche den Würfel schneidet. Es gibt 256 mögliche Würfelsituationen, aber durch Symmetrien reduziert sich die Anzahl der Muster auf nur 14.

**5.Oberflächenschnittpunkte berechnen:**  
Für die Schnittkanten im Würfel wird die genaue Position berechnet, indem zwischen den Dichten an den Ecken interpoliert wird (lineare Annäherung).

**6.Dreiecke erstellen:**  
Die Schnittpunkte werden verbunden, um Dreiecke zu erzeugen, die die Oberfläche darstellen.

**7.Berechnung der Normalen:**  
Um die Dreiecke schön darzustellen (z. B. mit Schatteneffekten), werden Normalenvektoren berechnet, die senkrecht zur Oberfläche stehen.

**8.Ausgabe der Oberflächendaten:**  
Die erzeugten Dreiecke mit ihren Positionen und Normalenvektoren werden für die Darstellung der Oberfläche ausgegeben.

### **Einfach gesagt:**

Der Algorithmus untersucht, wo ein bestimmter Schwellenwert (z. B. für Knochen oder Gewebe) in einem 3D-Bild überschritten wird, und baut daraus kleine Dreiecke zusammen, um die Oberfläche dieser Struktur zu zeigen.

## Funktionale Erweiterungen:

**Boolean-Operationen**: Der Algorithmus unterstützt das Schneiden, Verschließen (Capping) und Extrahieren von mehreren Oberflächen, ähnlich wie bei chirurgischen Operationen.

**Index-basierte Modellierung**:

index = 0: außerhalb der Oberfläche.

index = 255: innerhalb der Oberfläche.

0 < index < 255: auf der Oberfläche.

**Clipping von Dreiecken**: Bei sich überschneidenden Oberflächen wird der Sutherland-Hodgman-Algorithmus verwendet.

**Texturierung**: Original-Slice-Daten werden verwendet, um Oberflächen mit Texturen zu versehen.

### Implementierung:

* Der Algorithmus wurde in C programmiert und läuft auf verschiedenen Plattformen (Unix, VMS, IBM, etc.).
* Modelle mit hoher Dreiecksanzahl (bis zu 500.000 Dreiecken) können bei Bedarf durch Schneiden und Reduzieren der Auflösung effizienter gemacht werden.
* Beispielzeit: Erstellung eines Modells aus 93 CT-Schnitten dauert auf einem VAX 11/780 etwa 30 Minuten.

### Ergebnisse:

* **CT-Scans**: Detaillierte Rekonstruktion von Knochen- und Weichgewebeoberflächen (bis zu 550.000 Dreiecke) mit Details wie einem gepiercten Ohr und Klebebandabdruck.
* **MR-Scans**: Darstellung komplexer Anatomie mit Textur-Interpunktionen, z. B. an sagittalen Schnitten (max. 330.000 Dreiecke).
* **SPECT-Scans**: Performanz auf niedrig aufgelösten Daten getestet (z. B. 5.000 Dreiecke für die Oberfläche des Herz-Blutpools).

Der **Marching Cubes-Algorithmus** ist ideal für Anwendungsfälle, bei denen ein 3D-Oberflächenmodell aus diskreten Volumendaten (wie Voxel) generiert werden muss. Insbesondere eignet sich dieser Algorithmus für medizinische Bildverarbeitung, bei der komplexe anatomische Strukturen aus volumetrischen Daten (z.B. CT, MRI, SPECT) extrahiert werden. Anwendungsbereiche umfassen:

1. **Medizinische Bildgebung**: Um detaillierte 3D-Oberflächen von Organen, Knochen oder Tumoren zu rekonstruieren.
   1. Hierbei kann er auf Daten aus **DICOM**-Dateien angewendet werden, wie es im Zusammenhang mit deinem Projekt zur Anzeige von **Herzmodellen** und **DICOM-Objekten** verwendet wird.
2. **Wissenschaftliche Visualisierung**: Bei der Modellierung und Visualisierung komplexer Objekte, wie sie in Biologie, Geologie oder Meteorologie vorkommen, zum Beispiel die Visualisierung von **Geländestrukturen** oder **Fluidströmen**.
3. **Computergrafik und Simulationen**: Besonders bei der **Erstellung von 3D-Modellen** aus wissenschaftlichen Daten oder beim Rendern komplexer 3D-Strukturen (wie z.B. **Skelette oder Anatomiemodelle**).

Video zu Marching Cube Algorythmus

<https://youtu.be/voqnQ2v9Oxw?si=ltNpfgiqxoe83285>

[](https://youtu.be/voqnQ2v9Oxw?si=ltNpfgiqxoe83285)

<https://youtu.be/M3iI2l0ltbE?si=sKvop_ohrI3efi2h>

[](https://youtu.be/M3iI2l0ltbE?si=sKvop_ohrI3efi2h)

[UdomkarnBoonyaprasert/dicom2mesh](https://github.com/UdomkarnBoonyaprasert/dicom2mesh) ---> GitHub Projekt das 3D mesh von Dcom Image Stack lädt

## Nützliche Python Pakete:

**scikit-image - Marching Cubes**

scikit-image enthält eine sehr praktische Funktion namens **marching\_cubes** für 3D-Volumen-Daten.

marching\_cubes verwenden, um aus einem Dichtefeld (zum Beispiel aus DICOM-Daten oder numpy-Arrays) eine 3D-Oberfläche zu erzeugen.

pip install scikit-image

**VTK (Visualization Toolkit)**

**VTK** ist ein umfangreiches Toolkit für 3D-Visualisierung, das auch den Marching Cubes Algorithmus implementiert und oft in medizinischer Bildverarbeitung verwendet wird.

DICOM-Dichtefeld in ein 3D-Modell zu konvertieren

pip install vtk

import vtk

# VTK-Daten für dein Volumen erzeugen (z.B. DICOM- oder .nii-Datei einlesen)

reader = vtk.vtkNIFTIImageReader()

reader.SetFileName('path/to/your/dicom\_or\_nifti\_file')

reader.Update()

# Marching Cubes anwenden

mc = vtk.vtkMarchingCubes()

mc.SetInputData(reader.GetOutput())

mc.ComputeNormalsOn()

mc.SetValue(0, 100) # Isosurface-Wert, um das Dichtefeld zu durchsuchen

# Visualisierung

mapper = vtk.vtkPolyDataMapper()

mapper.SetInputConnection(mc.GetOutputPort())

actor = vtk.vtkActor()

actor.SetMapper(mapper)

renderer = vtk.vtkRenderer()

render\_window = vtk.vtkRenderWindow()

render\_window.AddRenderer(renderer)

render\_window\_interactor = vtk.vtkRenderWindowInteractor()

render\_window\_interactor.SetRenderWindow(render\_window)

renderer.AddActor(actor)

renderer.SetBackground(1.0, 1.0, 1.0)

render\_window.Render()

render\_window\_interactor.Start()

## MATLAB: Isosurface-Funktion

In **MATLAB** ist die einfachste Methode zur Erzeugung von 3D-Modellen aus Volumendaten die Verwendung der eingebauten **isosurface** Funktion:

Surf

Sclice

vol3d

## DICOM-Daten:

Die Konvertierung von DICOM-Daten in 3D-Modelle erfolgt normalerweise in zwei Schritten:

**Daten einlesen**:

**Triangulieren/Visualisieren**:

Tools wie pydicom (Python) oder das dicomread in MATLAB können verwendet werden, um DICOM-Daten zu laden.

# /Marching Triangles:

**Marching Triangles (MT)** ein neuer Algorithmus zur Triangulation von **impliziten Oberflächen** (Flächen, die durch mathematische Gleichungen definiert sind, anstatt explizit mit Punkten und Kanten dargestellt zu werden). Im Vergleich zu herkömmlichen Methoden wie **Marching Cubes (MC)**, die den Raum in Volumen unterteilen, um Oberflächen darzustellen, bietet MT eine **effizientere** und genauere Lösung.

Wichtigste Merkmale von **Marching Triangles (MT)**:

* **Effizientere Mesh-Darstellung**: MT wächst ein **trianguliertes Netz** auf der Oberfläche, basierend auf der lokalen Geometrie und Topologie der Fläche, was zu gleichmäßigeren Dreiecksformen führt.
* **Lokal Delaunay**: Der Algorithmus gewährleistet, dass jedes Dreieck lokal **Delaunay** ist – das bedeutet, die Dreiecke haben eine gute geometrische Form, was für die Genauigkeit der Darstellung wichtig ist.
* **Globale Delaunay-Eigenschaft**: MT sorgt dafür, dass das gesamte Netz die Delaunay-Eigenschaft beibehält, was zu einer besseren Oberflächenapproximation führt.
* **Offene Mannigfaltigkeiten**: MT kann auch „offene“ (unendliche) Oberflächen, die oft in medizinischen Abbildungen vorkommen, effizient darstellen.

### **Vorteile im Vergleich zu Volumen-basierten Verfahren (wie Marching Cubes):**

* **Kostengünstiger**: Die Repräsentations- und Rechenkosten sind **deutlich niedriger** als bei traditionellen volumenbasierten Techniken wie MC.
* **Bessere Geometrie**: MT führt zu weniger ineffizienten und besser geformten Dreiecken, was eine genauere und effizientere Darstellung ermöglicht.
* **Flexibilität**: MT ermöglicht die **dynamische Integration neuer Daten** und ist für **offene** und komplexe Oberflächen geeignet, was zu einer besseren Skalierbarkeit führt.

**Delaunay-Bedingung**: Eine **Delaunay-Bedingung** wird eingeführt, um sicherzustellen, dass die Verbindungen zwischen den Mesh-Punkten lokal Delaunay sind. Dies führt zu einer besseren geometrischen Struktur der Dreiecke.

**Eignung für Anwendungen**: Das resultierende Mesh ist besonders geeignet für **Visualisierung** und **finite Elemente Analyse**, da es gleichmäßig geformte Dreiecke enthält.

**Rekonstruktion offener Mannigfaltigkeiten**: MT überwindet die Limitationen traditioneller Methoden und eignet sich auch für die Darstellung von **offenen Mannigfaltigkeiten** (z.B. Oberflächen mit Löchern).

**Anwendungen auf echte Objekte**: MT wird verwendet, um Modelle von realen Objekten zu rekonstruieren, basierend auf **mehrdimensionalen Rang-Abbildungen** (Range-Image), was eine präzisere und zuverlässigere Darstellung von komplexer Geometrie ermöglicht.

**Dynamische Datensfusion**: MT unterstützt die **dynamische Fusion** von Oberflächeninformationen und ist daher flexibler als volumetrische Verfahren.

**Delaunay-Triangulation auf einer Mannigfaltigkeitsoberfläche:** Die Delaunay-Triangulation eines Punktesets auf einer Mannigfaltigkeitsoberfläche wird erstellt, um die Oberfläche zu approximieren, indem tetraedrische Volumina gebildet werden, bei denen jede Umkugel keine inneren Punkte enthält. Damit die Triangulation eine korrekte Annäherung an die Oberfläche darstellt, muss sie die relativen Positionen der Punkte auf der Oberfläche respektieren und sicherstellen, dass jedes Dreieck lokal „gekrümmt“ ist, um zur Mannigfaltigkeit zu passen.

**Delaunay-Oberflächenbedingung:** Ein Dreieck kann nur zur triangulierten Fläche hinzugefügt werden, wenn es der Delaunay-Oberflächenbedingung entspricht. Diese Bedingung stellt sicher, dass das Dreieck keine Teile des Modells schneidet, wodurch die Geometrie der Oberfläche erhalten bleibt. Obwohl es sich nicht um eine exakte Delaunay-Bedingung handelt, gewährleistet diese entspannte Version, dass die Triangulation lokal Delaunay-ähnlich bleibt und die Oberfläche ohne Überlappung oder Selbstüberschneidung annähert wird, insbesondere bei komplexen Geometrien. Das resultierende Modell wird die Eigenschaften von Nachbarschaftssymmetrie und Isotropie haben und der optimalen Delaunay-Triangulation nahestehen.

## Zusammengefasst:

Das "Marching Triangles"-Algorithmus (MT) ist ein Verfahren zur Triangulierung einer Mannigfaltigkeit, die durch eine implizite Oberfläche definiert ist. Die Idee des Algorithmus ist, ein Netz (Mesh) von Dreiecken zu erstellen, das die Oberfläche möglichst genau annähert, und dabei lokale Oberflächenstrukturen wie Krümmungen zu berücksichtigen.

## Ablauf des Algorythmus:

**1.Initialisierung:** Der Algorithmus beginnt mit einem kleinen Ausgangsmodell, das entweder ein einfaches Dreieck oder bereits ein bestehendes Modell sein kann. Das Modell wird durch eine Liste von Kanten und Ecken beschrieben.

**2.Berechnung neuer Ecken:** Bei jedem Schritt wird eine neue Ecke auf der Oberfläche der impliziten Funktion geschätzt. Die Schätzung erfolgt, indem der Mittelpunkt einer Kante auf der Modellgrenze genommen und eine neue Ecke projiziert wird, die einen konstanten Abstand von der Kante in der Fläche der Kante hat.

**3.Positionierung der neuen Ecke:** Der neue Punkt wird dann auf der impliziten Oberfläche positioniert. Das bedeutet, dass der Punkt die nächste Position auf der Fläche der Oberfläche finden muss. Der Algorithmus prüft, ob die Projektionsmethode für diesen Punkt sinnvoll ist oder ob er an einer falschen Stelle landet (z.B. auf einer dünnen Stelle der Oberfläche).

**4.Delaunay-Bedingung anwenden:** Der neue Punkt bildet ein Dreieck mit bestehenden Punkten auf dem Modell. Dieses Dreieck muss die Delaunay-Bedingung erfüllen, die besagt, dass der Umkreis des Dreiecks keine anderen Punkte enthalten darf. Dadurch wird sichergestellt, dass die Dreiecke gut verteilt und stabil sind.

**5.Fehlerbehandlung:** Wenn das Dreieck die Delaunay-Bedingung nicht erfüllt, prüft der Algorithmus benachbarte Dreiecke und passt die Grenzen des Modells so an, dass diese Bedingung eingehalten wird.

**6.Mesh-Wachstumsprozess:** Sobald ein Dreieck zur Oberfläche hinzugefügt wird und die Bedingung erfüllt, wird das neue Dreieck dem Netz hinzugefügt und die Kantenliste wird um neue Kanten ergänzt, damit das Modell weiterwachsen kann.

**7.Terminierung:** Der Algorithmus hört auf, wenn keine weiteren Kanten mehr getestet werden können oder wenn keine neuen Dreiecke zum Modell hinzugefügt werden können. Wenn die Kantenliste leer wird, bedeutet dies, dass das Modell vollständig ist.

# Blender:

Ein Tutorial zur Verwendung von OrtogOnBlender

<https://youtu.be/HrgijOMqK0s>

[](https://youtu.be/HrgijOMqK0s)

MedBlender wäre auch eine Option

**Erweiterbare Plattform**: Dank der offenen Architektur von Blender können individuelle Workflows erstellt und Anpassungen durch Python-Scripting vorgenommen werden.

Add-ons ermöglichen die direkte Unterstützung von medizinischen Bildformaten wie DICOM, was die Nutzung erleichtert.

**Exportmöglichkeiten**: Blender unterstützt viele 3D-Formate (z. B. OBJ, STL, FBX), die für 3D-Druck, Simulationen oder Visualisierungen in anderen Softwarelösungen verwendet werden können.

in Blender ist es theoretisch und praktisch möglich, gezielt Teile von CT-Bildern wie den Brustkorb auszublenden oder nicht auszugeben. Dies wird häufig durch **Segmentierung** und **Maskierung** der Daten erreicht. Hier ist, wie das funktioniert:

### **Segmentierung im Vorfeld (z. B. in 3D Slicer)**

* **Warum?**: CT-Bilder enthalten oft ein komplettes Dichtefeld (Hounsfield-Werte), das verschiedene Gewebe (Knochen, Organe, Weichteile) umfasst.
* **Wie?**:
  + Verwenden Sie eine Segmentierungssoftware wie **3D Slicer**, um spezifische Gewebe oder Strukturen (z. B. Brustkorb, Organe) auszuwählen oder auszuschließen.
  + Exportieren Sie die segmentierten Daten als **STL-, OBJ- oder NRRD-Datei**, die Blender importieren kann.

**Literaturverzeichnis:**

Lorensen, W. E., & Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. ACM SIGGRAPH Computer Graphics, 21(4), 163–169.

Hilton, A., & Illingworth, J. (1997). Marching triangles: Delaunay implicit surface triangulation. In Proceedings of the IEEE International Conference on Image Processing (pp. 903–906). IEEE.