1 3D-Visualisierung

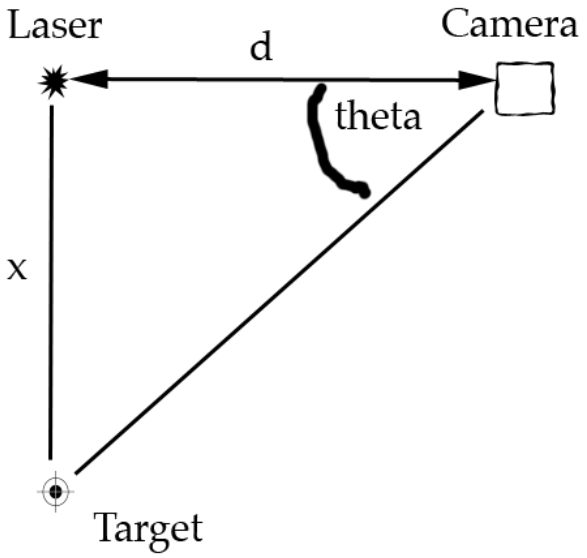
Dieses Kapitel behandelt die Grundlagen der Gewinnung, Verarbeitung und Visualisierung von 3D-Daten. Der Fokus liegt auf der Unterscheidung zwischen indirekter (Oberflächen-) und direkter (Volumen-) Visualisierung sowie den mathematischen und algorithmischen Grundlagen des Volume Renderings.

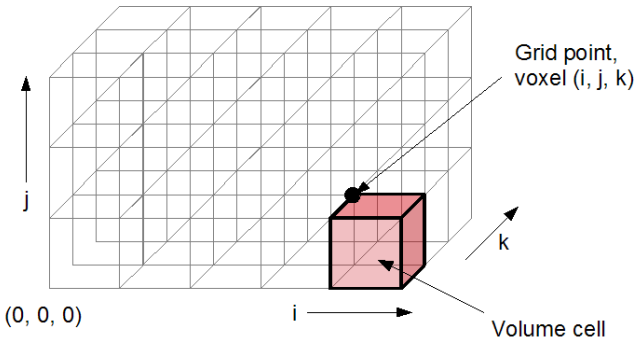
1.1 3D-Daten und Datengewinnung

3D-Daten bestehen aus Messwerten, die im dreidimensionalen Raum verteilt sind. Jeder Wert besitzt Koordinaten (x, y, z) und kann skalar (z. B. Dichte, Temperatur) oder höherdimensional (z. B. Vektorfelder wie Windrichtung) sein.

Datenquellen

- **Terrain-Daten**: Höhenmessungen an Positionen (x, y), oft ergänzt durch Satellitenbilder.
- **Laser Scanning**: Aktive Projektion eines Laserstrahls auf eine Oberfläche. Durch Triangulation (bekannter Abstand und Winkel zwischen Laser und Kamera) wird die Tiefe berechnet. Ergebnis: Unstrukturierte Punktwolke.
- **Range Images**: Bilder, bei denen jeder Pixel (u, v) eine Tiefeninformation $r(u, v)$ speichert.
- **Medizinische Bilddaten (CT/MRI)**: Messung physikalischer Eigenschaften (Protonendichte bei MRI, Gewebedichte bei CT). Die Daten liegen meist als Stapel von 2D-Schichten (*Slices*) vor, was ein reguläres 3D-Gitter bildet.
- **Simulation**: Numerische Simulationen (z. B. Wetter, Strömung), definiert auf Gittern.





1.2 Triangulation von Punktwolken

Um aus unstrukturierten Punktwolken (Menge von Punkten $s_i = (x_i, y_i, z_i)$) eine visualisierbare Oberfläche zu erzeugen, ist eine Triangulation notwendig.

1.2.1 Planare Triangulation

Bei einfachen Oberflächen (z. B. Terrain ohne Überhänge) können Punkte auf eine 2D-Ebene projiziert, dort trianguliert und das entstehende Netz anschließend anhand der z-Werte deformiert werden.

1.2.2 Voronoi-Diagramm und Delaunay-Triangulation

Diese Konzepte sind fundamental für die Vernetzung von Punkten.

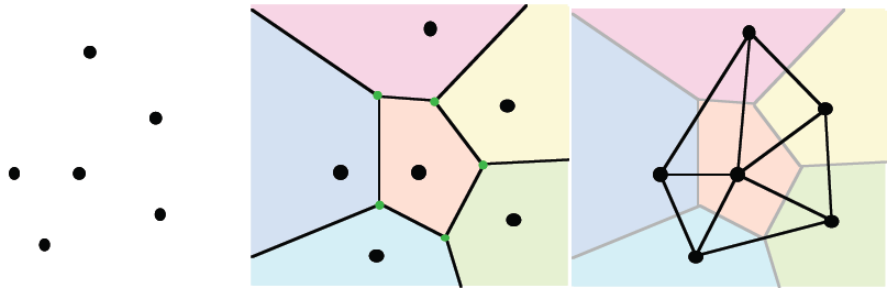
Voronoi-Diagramm

Für eine Menge von Punkten definiert die **Voronoi-Zelle** eines Punktes S_i den Bereich, der näher an S_i liegt als an jedem anderen Punkt der Menge. Die Kanten sind die Orte, die zu zwei Punkten den gleichen Abstand haben; Knoten haben zu mindestens drei Punkten den gleichen Abstand.

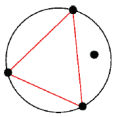
Delaunay-Triangulation

Der duale Graph des Voronoi-Diagramms. Verbindet die Zentren benachbarter Voronoi-Zellen.

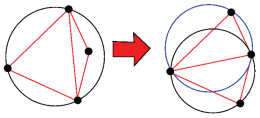
- **Leer-Kreis-Eigenschaft**: Der Umkreis eines jeden Dreiecks in einer Delaunay-Triangulation enthält keine anderen Punkte der Punktmenge.
- **Maximierung der kleinen Winkel**: Vermeidet schmale, langgestreckte Dreiecke ("Sliver Polygons"), was für die numerische Stabilität und das Rendering vorteilhaft ist.



Voronoi-Parkettierung (Tesselation) → Voronoi-Diagramm → Delaunay-Triangulation



nicht erlaubt



Korrektur der Delaunay-Triangulation durch Umdrehen der Kanten („Edge Flipping"):

1.3 Indirekte Volumenvisualisierung (Oberflächenrendering)

Bei der indirekten Visualisierung wird aus den Volumendaten eine explizite geometrische Zwischenrepräsentation (meist Polygone) extrahiert.

1.3.1 Isoflächen (Isosurfaces)

Analog zu Konturlinien in 2D (Linien gleicher Höhe) sind Isoflächen in 3D Trennflächen zwischen Strukturen unterschiedlicher Dichte. Eine Isofläche ist definiert durch die implizite Gleichung:

$$i(x) = V(x) - \tau = 0$$

wobei $V(x)$ der Voxelwert und τ der Isowert (Schwellenwert) ist.

1.3.2 Marching Cubes Algorithmus

Der Standardalgorithmus zur Extraktion von Isoflächen aus Gitterdaten.

1. Betrachte eine logische Volumenzelle, gebildet durch 8 Voxel-Nachbarn.
2. Klassifiziere jeden Eckpunkt als "innen" ($V \geq \tau$) oder "außen" ($V < \tau$).
3. Dies ergibt $2^8 = 256$ mögliche Konfigurationen.
4. Unter Ausnutzung von Symmetrien (Rotation, Invertierung) Reduktion auf 15 Basisfälle.
5. Generiere Dreiecke basierend auf der Konfiguration (Interpolation der Schnittpunkte auf den Kanten).

Das 2D-Äquivalent hierzu ist der **Marching Squares** Algorithmus (4 Pixel, 16 Fälle).

1.3.3 Optimierung der Rendering-Pipeline

Da Marching Cubes extrem viele Dreiecke erzeugen kann, sind Optimierungen notwendig:

- **Culling**: Entfernen nicht sichtbarer Geometrie.
  - *Backface-Culling*: Rückseiten ignorieren.
  - *View-Frustum-Culling*: Außerhalb des Sichtbereichs ignorieren.
  - *Occlusion-Culling*: Verdeckte Objekte ignorieren.
- **Meshreduktion**: Vereinfachung des Netzes durch Zusammenfassen von Polygonen in flachen Bereichen, um die Anzahl der Primitive zu senken.
- **Mesh-Glättung**: Entfernung von Artefakten und Rauschen (z. B. Laplacesche Glättung), wobei darauf geachtet werden muss, das Volumen nicht zu stark zu verfälschen ("Shrinking").

1.4 Direkte Volumenvisualisierung (Volume Rendering)

Im Gegensatz zur indirekten Methode wird hier keine Geometrie erzeugt. Das Bild wird direkt aus den Voxeldaten berechnet, indem physikalische Lichtinteraktion simuliert wird.

1.4.1 Physikalisches Modell: Density Emitter Model

Das Volumen wird als halbtransparente Wolke betrachtet, die Licht sowohl emittiert (leuchtet) als auch absorbiert (abschwächt).

1.4.2 Volumen-Rendering-Gleichung

Die Intensität $I(s)$ entlang eines Sichtstrahls berechnet sich aus dem Hintergrundlicht I_{s_0}, das durch das Volumen abgeschwächt wird, und dem kumulierten Licht, das von den Voxeln entlang des Strahls emittiert und wiederum abgeschwächt wird.

Integralform

$$I(s) = I_{s_0} \cdot e^{-\int_{s_0}^{S} \tau(t)dt} + \int_{s_0}^{S} Q(\tilde{s}) \cdot e^{-\int_{\tilde{s}}^{S} \tau(t)dt} ds$$

- $\tau(t)$: Optische Dichte (Absorptionskoeffizient).
- $Q(\tilde{s})$: Emission (Leuchtkraft) an der Stelle $\tilde{s}$.
- Der Exponentialterm beschreibt die Transparenz (Transmission) über eine Strecke.

1.4.3 Diskrete Näherung und Ray Casting Pipeline

Für die computergrafische Umsetzung wird das Integral durch eine Summe approximiert (Ray Casting). Der Strahl wird in Segmente Δs unterteilt.

$$I(S) \approx I_0 \prod_{k=0}^{n-1} t_k + \sum_{k=0}^{n-1} \left(Q_k \cdot \Delta s \cdot \prod_{j=k+1}^{n-1} t_j \right)$$

Hierbei ist $t_k = e^{-\tau_k \Delta s}$ die Transparenz des k-ten Segments. Die Pipeline besteht aus vier Schritten:

1. Abtastung (Sampling): Entlang des Sichtstrahls werden Werte in Abständen Δs genommen. Da diese Positionen meist nicht exakt auf Voxelzentren fallen, ist **Interpolation** nötig:

- *Nearest Neighbor*: Blockartige Artefakte, schnell.
- *Trilinear*: Glatter, Standardverfahren (interpoliert in x, y, z Richtung).
- *Shannon-Theorem*: Die Abtastrate muss hoch genug sein (Nyquist-Frequenz), um Aliasing-Artefakte zu vermeiden ($\Delta s < 0.5\times$ Voxelgröße).

2. Klassifikation und Beleuchtung (Shading): Den interpolierten Skalarwerten müssen optische Eigenschaften zugeordnet werden.

- **Transferfunktionen**: Bilden Dichtewerte auf Farbe (Q) und Opazität/Transparenz (t) ab.
  - *Farbtransferfunktion*: Ordnet Farbe zu (z. B. Knochen = weiß, Muskel = rot).
  - *Opazitätstransferfunktion*: Bestimmt Sichtbarkeit (z. B. Luft = transparent).
- **Shading**: Um 3D-Strukturen besser erkennbar zu machen, wird eine Beleuchtung simuliert (z. B. Phong-Shading). Der Normalenvektor für das Shading wird meist aus dem Gradienten der Voxeldaten berechnet.

3. Komposition (Compositing): Das Aufsummieren der Farb- und Opazitätswerte entlang des Strahls.

Back-to-Front vs. Front-to-Back

- **Back-to-Front** (vom Hintergrund zum Auge): Ähnlich dem Painter's Algorithm.

$$I_k = I_{k-1} \cdot t_k + C_k$$

  wobei C_k der emittierte Farbanteil ist.
- **Front-to-Back** (vom Auge zum Hintergrund): Ermöglicht Optimierung. Man muss Intensität I und akkumulierte Transparenz τ tracken.

$$I_{k+1} = I_k + C_{k+1} \cdot \tau_k$$

$$\tau_{k+1} = \tau_k \cdot t_{k+1}$$

- **Early Ray Termination**: Wichtige Optimierung bei Front-to-Back. Wenn die akkumulierte Opazität fast 100% erreicht (Transparenz ≈ 0), kann der Strahl abgebrochen werden, da dahinterliegende Voxel nicht mehr sichtbar sind.