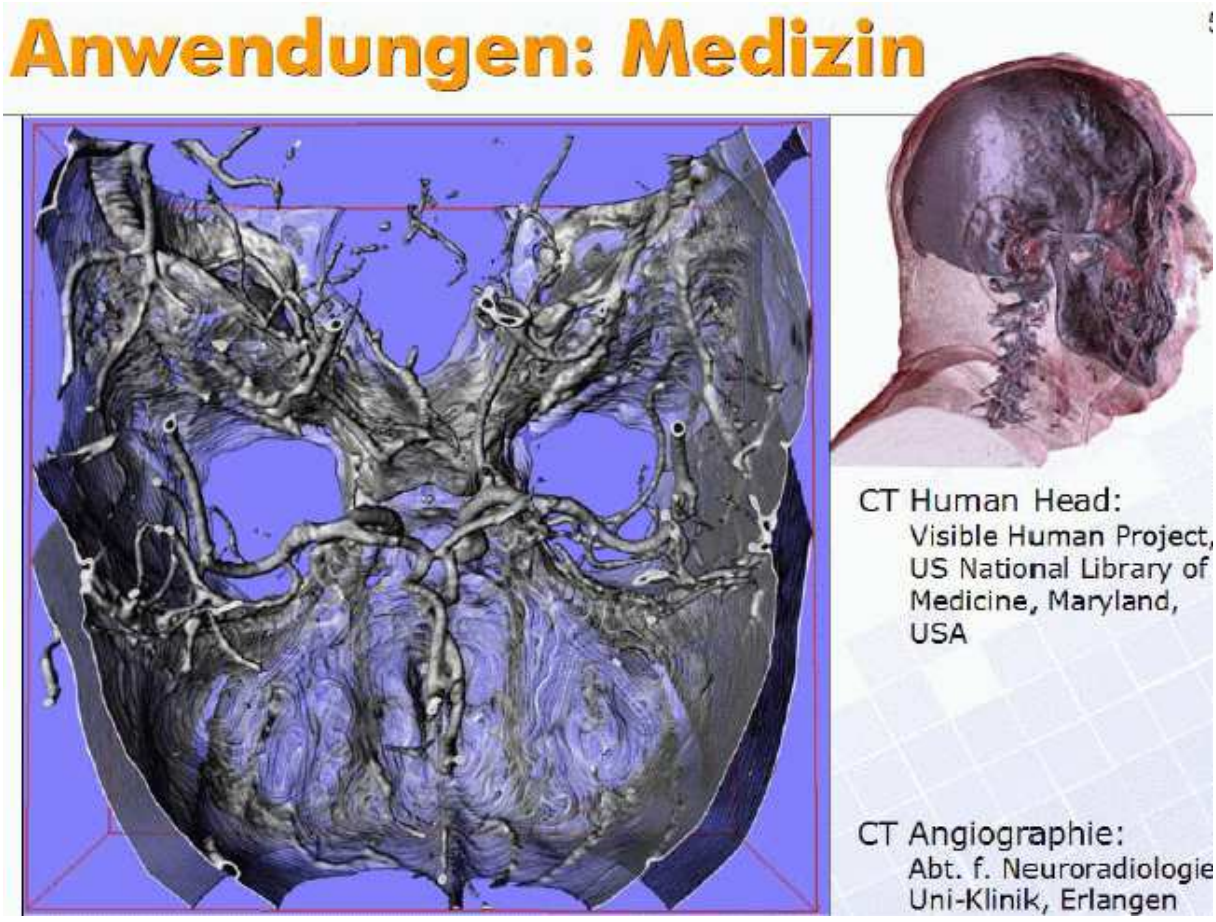
14. Volumen-Rendering

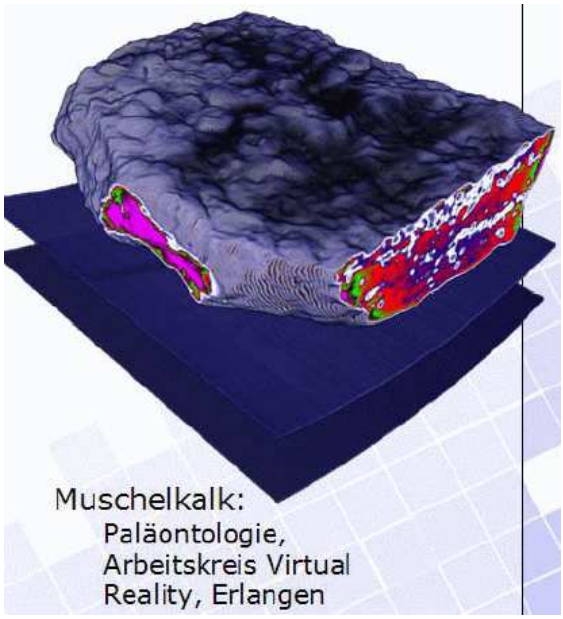
klassische fotorealistische Computergrafik:
normalerweise werden Oberflächen dargestellt
(außer bei Spezialeffekten wie Nebel, Rauch, Feuer...)

in der Visualisierung von Daten:
oft Interesse, in ein Volumen *hineinzusehen*
("Röntgenblick")



weitere Anwendungen:

Geologie



Muschelkalk:
Paläontologie,
Arbeitskreis Virtual
Reality, Erlangen

Archäologie

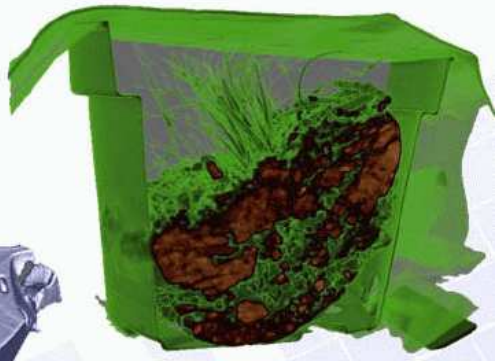
Sotades Pygmäen-Statue,
5. Jh. v. Chr.,
ARTIS, Antikensammlung
der Universität Erlangen-
Nürnberg

Werkstoffwissenschaften
Qualitätssicherung

Engine Block,
General Electric

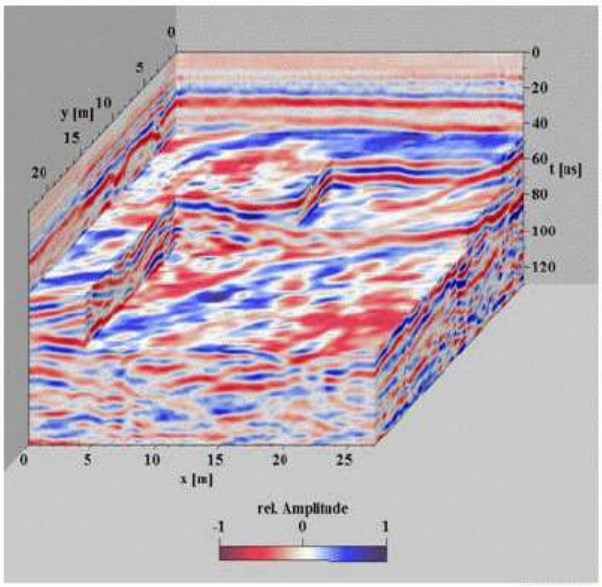
Micro CT Verbundwerkstoff,
Werkstoffwissenschaften
Universität Erlangen-Nürnberg

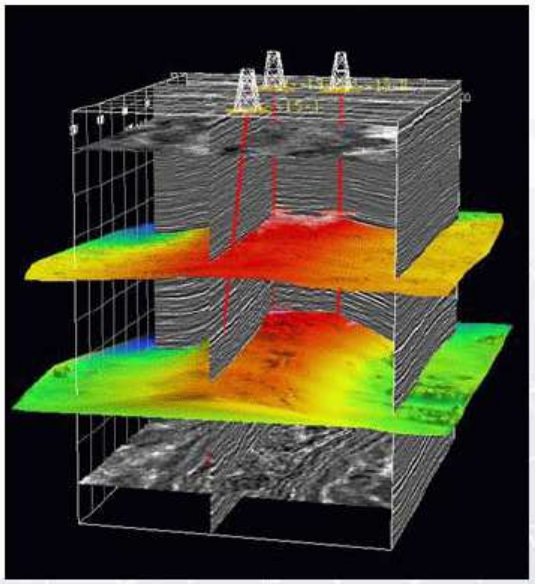
Biologie



Biologische Bodenprobe, CT,
Arbeitskreis Virtual Reality,
Universität Erlangen-Nürnberg

- Geoseismische Daten
- Erdölindustrie





(Rezk-Salama, o.J.)

Datenspezifikation:

- Beobachtungsraum: $\mathbf{R}^3$
- Datenverbund:
  Oft regelmäßig (Voxelgitter) oder strukturiert
- Merkmalsraum: [0,1]
  Es wird nur **eine** skalare Größe unterstützt

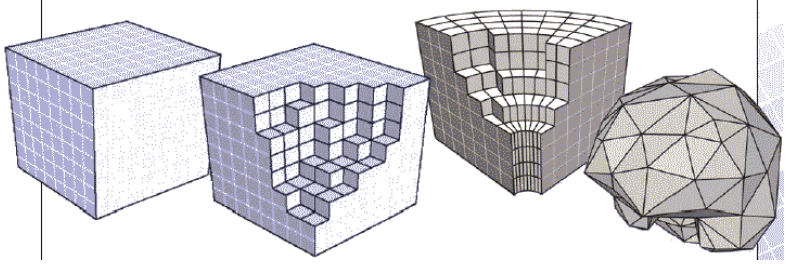
(Bartz 2005)

3D Skalarfelder

Input Daten: Kontinuierliches 3D Skalarfeld:

$$s = f(x, y, z); \quad x, y, z \in \mathbb{R}$$

abgetastet auf diskretem Gitter.



Überblick

Volumenvisualisierung

- Indirekte Verfahren

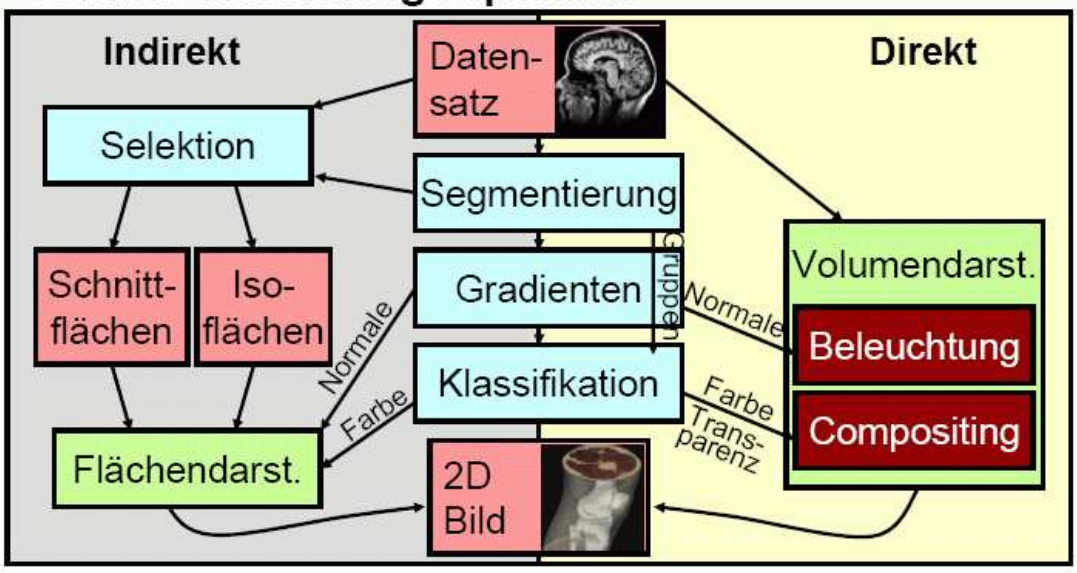
  Bestimme Oberflächen innerhalb des Skalarfeldes und stelle diese dar.

- Direkte Verfahren

  Interpretiere das Skalarfeld als transparentes Medium und verwende physikalisch basierte Strahlungs-berechnung.

(Rezk-Salama, o.J.)

Volume-Rendering Pipeline:

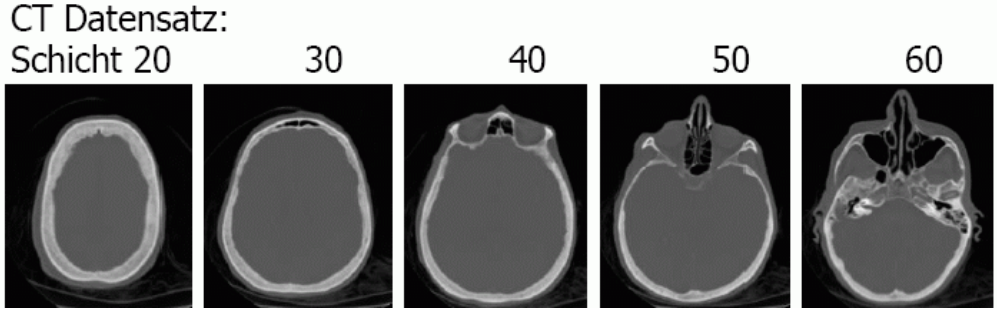


1. Ansatz: Schnittflächen

Schnittflächen (Voxelgitter):

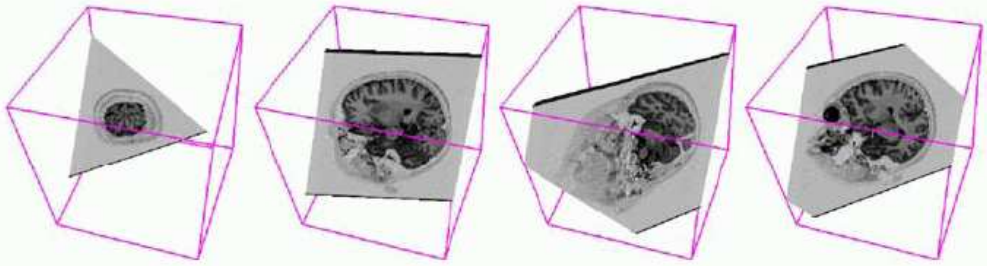
- orthogonal zu x-,y- bzw. z-Achse (re-sample)
- nutze Techniken zur 2D Darstellung (Farbe, Isolinien, Höhenfelder, ...)

CT Datensatz:



Schnittflächen (Voxelgitter):

- beliebige Ebenen
- nutze Hardware zum 3D-Texture-Slicing
  (berechne Schnitt zw. Ebene und Würfel)

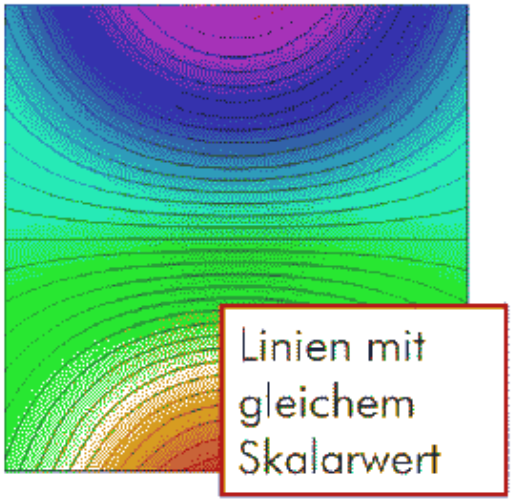


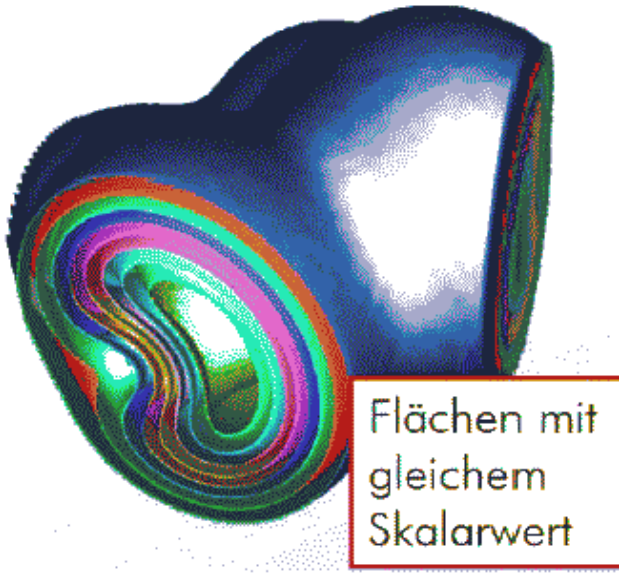
(Bartz 2005)

Isoflächen

- Analog zu Isolinien:
  Regionen mit gleichem Skalarwert

2D: Isolinien *3D: Isoflächen*

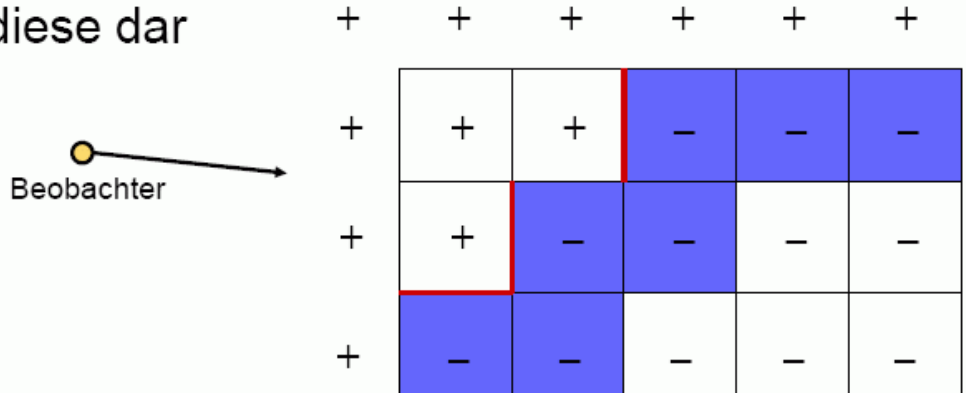




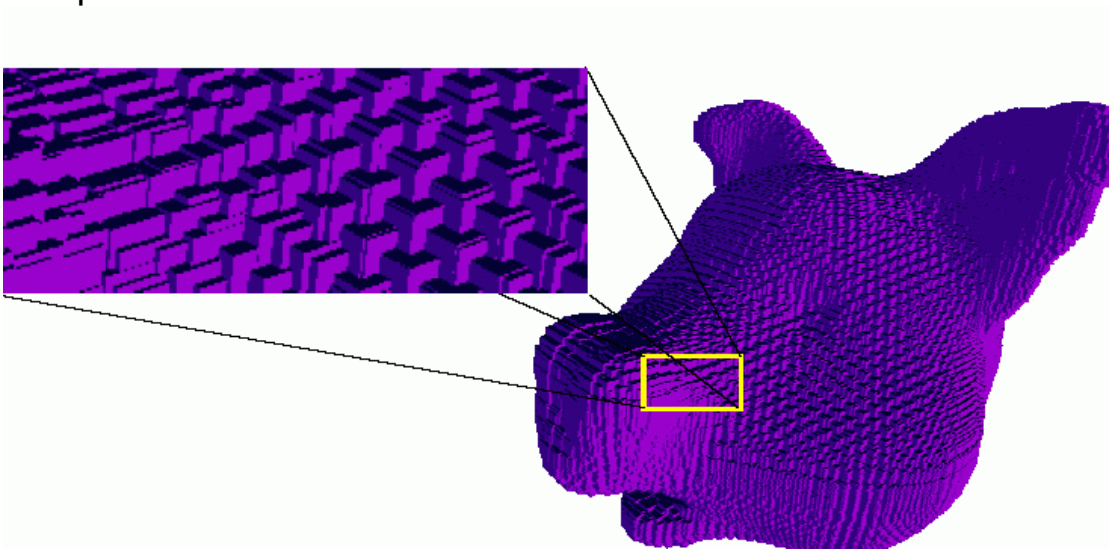
(Rezk-Salama, o.J.)

Binäre Voxelisierung (Cuberille):

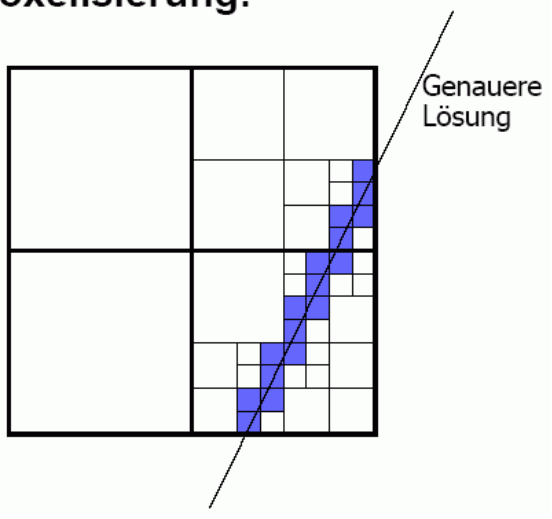
- Klassifiziere Dichte auf + oder -, je nachdem ob $\rho > \tau$ oder $\rho < \tau$
- Bestimme sichtbare Trennflächen und stelle diese dar



Beispiel:



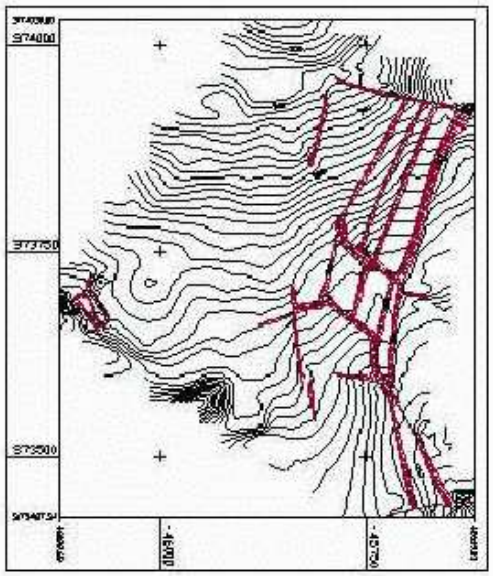
Adaptive Voxelisierung:

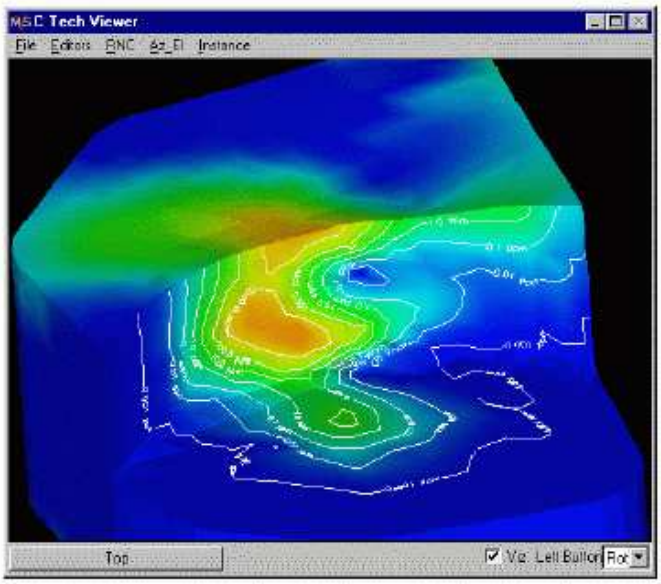


(Bartz 2005)

zur Vereinfachung zunächst 2-dimensionaler Fall:
Isolinien

- Gegeben sei ein qualitatives Merkmal über der Ebene: f(x,y)→ [a,b] (z.B. Höhenfeld) (Skalarfeld)
- **Isolinie** bezeichnet Punktmengen mit f(x,y) = const.
- **Isostreifen** bezeichnet Punktmengen mit f(x,y)∈ [v, v+Δv].
- Falls f(x,y) auf keinem ausgedehnten Bereich konstant ist, sind Isolinien Linien.





Anwendungen: Höhenlinien, Wetterkarten,...

(Bartz 2005)

wie lassen sich Isolinien erzeugen?

- für alle Pixel überprüfen, ob Isolinie hindurchgeht ("Pixel by pixel contouring") *oder*
- gröberes Gitter verwenden, interpolieren

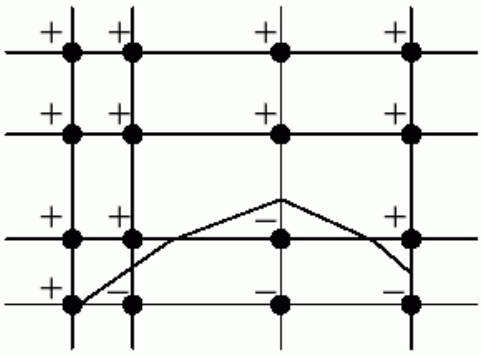
zum zweiten Ansatz:

Marching Squares:

- Die skalare Funktion wird evtl. auf einem gröberen Gitter abgetastet, als dargestellt wird.
- Für gegebenen Isowert werden Schnitte der Isolinien mit den Gitterkanten berechnet.
- In jeder Gitterzelle werden die Schnittpunkte mit Isoliniensegmenten verbunden.
- Bei mehrdeutigen Fällen werden weitere Punkte im Innern der Zellen ausgewertet.

(Bartz 2005)

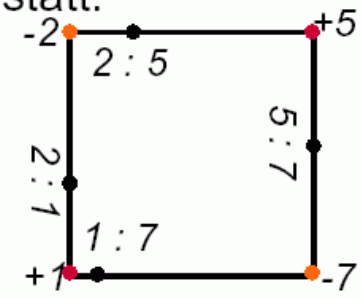
Marching Squares Beispiel:



Marching Squares Schnittberechnung:

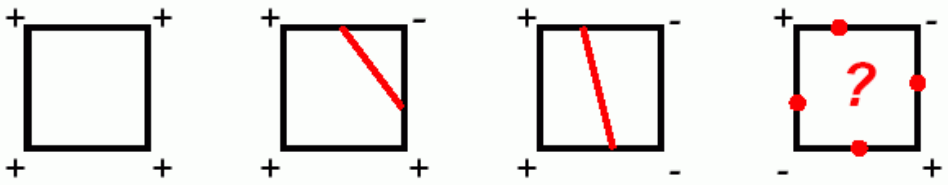
- Für gegebenen Isowert I, werden die Ecken der Zellen mit + oder – markiert, je nachdem, ob der Funktionswert größer oder kleiner I ist.
- Bei Kanten mit Ecken unterschiedlichen Vorzeichens, findet ein Schnitt statt.
- Die Schnittposition x auf der Kante [0,1] geht aus linearer Interpolation hervor:

$$x = (I - f_0) / (f_1 - f_0)$$



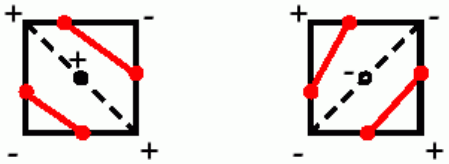
Marching Squares Fallunterscheidung:

Unter Berücksichtigung von Symmetrien (Rotationen und Komplement) bleiben noch vier verschieden Vorzeichenzuordnungnen:



Auflösung der Mehrdeutigkeit:

- Im Falle von 4 Schnittpunkten gibt es zwei Möglichkeiten, die gültige Isolinien erzeugen:



- Wenn die Isolinien keine Feinstruktur in der Zelle haben, kann über den Funktionswerte im Zellenmittelpunkt zwischen den Fällen unterschieden werden.

Marching Squares Algorithmen:

- **In Zellenreihenfolge:**

  Durchlaufe alle Zellen und generiere Segmente ➔ jeder Punkt 2 mal, höhere Interpolation schwierig

- **Linienverfolgung:**

  Verfolge Isolinien entlang von Saatzellen bis Rand erreicht wird oder die Linie geschlossen wird

➔ Hermite-Interpolation mit Hilfe von Gradienten aus zentralen Differenzen

Problem: Wie können Saatzellen schnell gefunden werden ?

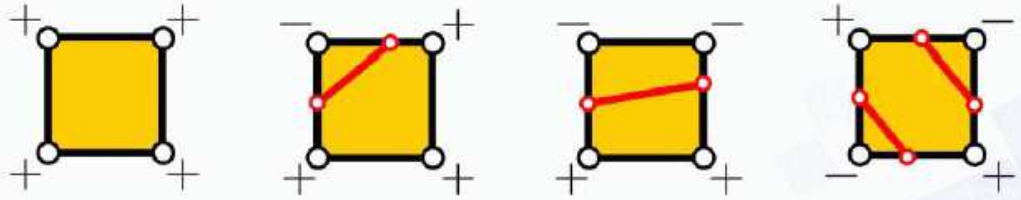
(Bartz 2005)

Zusammenfassung:

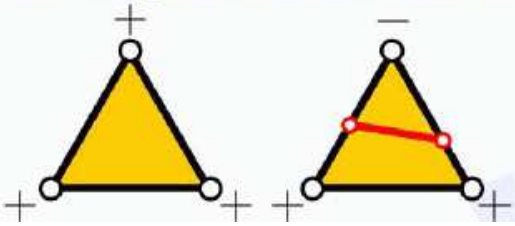
Isolinien

Sequentieller Algorithmus: Betrachte einzelne Zellen:

Rechteckige Zellen: 4 unterschiedliche Fälle:



Dreieckige Zellen: 2 unterschiedliche Fälle:



jetzt:

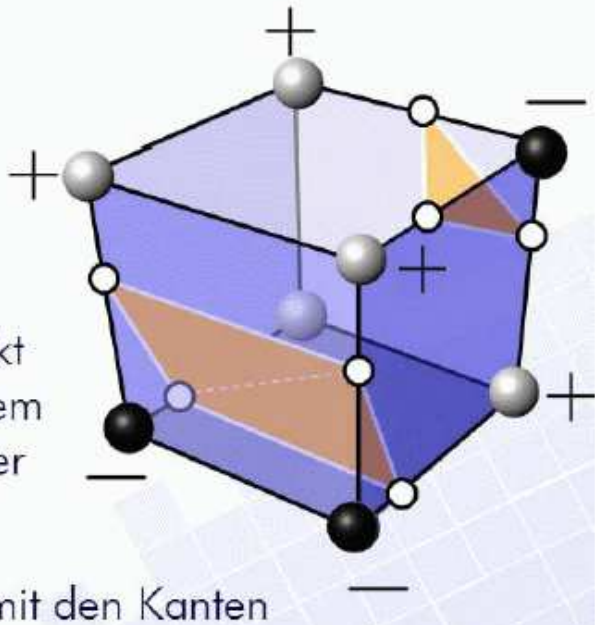
Isoflächenbestimmung

Vorgehensweise

Wähle einen Isowert

Betrachte Zellen, die aus je 8 Datenpunkten bestehen (*Cubes*)

Klassifiziere jeden Datenpunkt mit „+“ oder „−“, je nachdem ob der Datenwert größer oder kleiner als der Isowert ist.



Berechne die Schnittpunkte mit den Kanten

Bestimme eine Triangulierung

(Rezk-Salama, o.J.)

Marching Cubes Algorithmus:

- betrachte **jede Voxelzelle**
- klassifiziere **acht Voxel der Zelle** in innen/außen
- erstelle daraus 8-Bit Index
- schlage Schnittkanten nach
- interpoliere Schnittpositionen und berechne Normale
- schlage Triangulierung nach

Marching Cubes
[Lorensen, Cline, SIGGRAPH 1987]:

- arbeitet auf Originaldaten
- erzeugt Dreiecksnetz (lineare Approximation)
- Oberflächennormalen aus Gradienten
- effiziente Berechnung durch Lookup-Tabelle

(Bartz 2005)

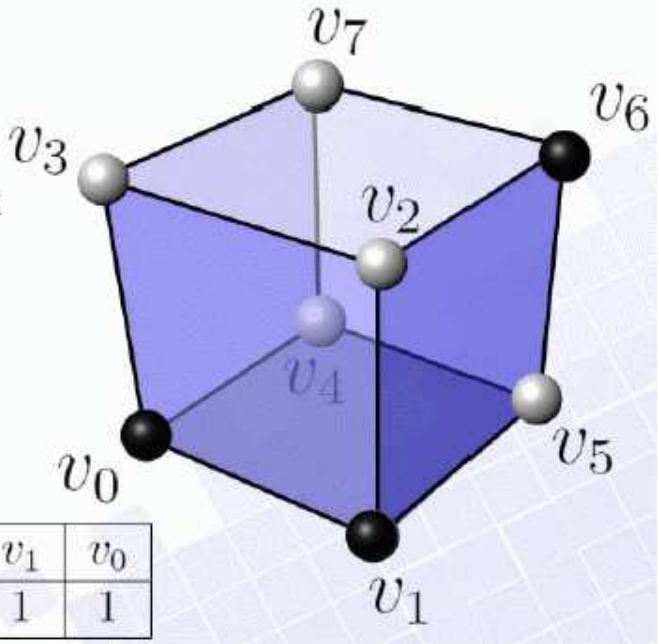
Marching Cubes -Algorithmus

nach Bill Lorensen und Harvey Cline

Jede Zellkonfiguration hat einen bestimmten Index.

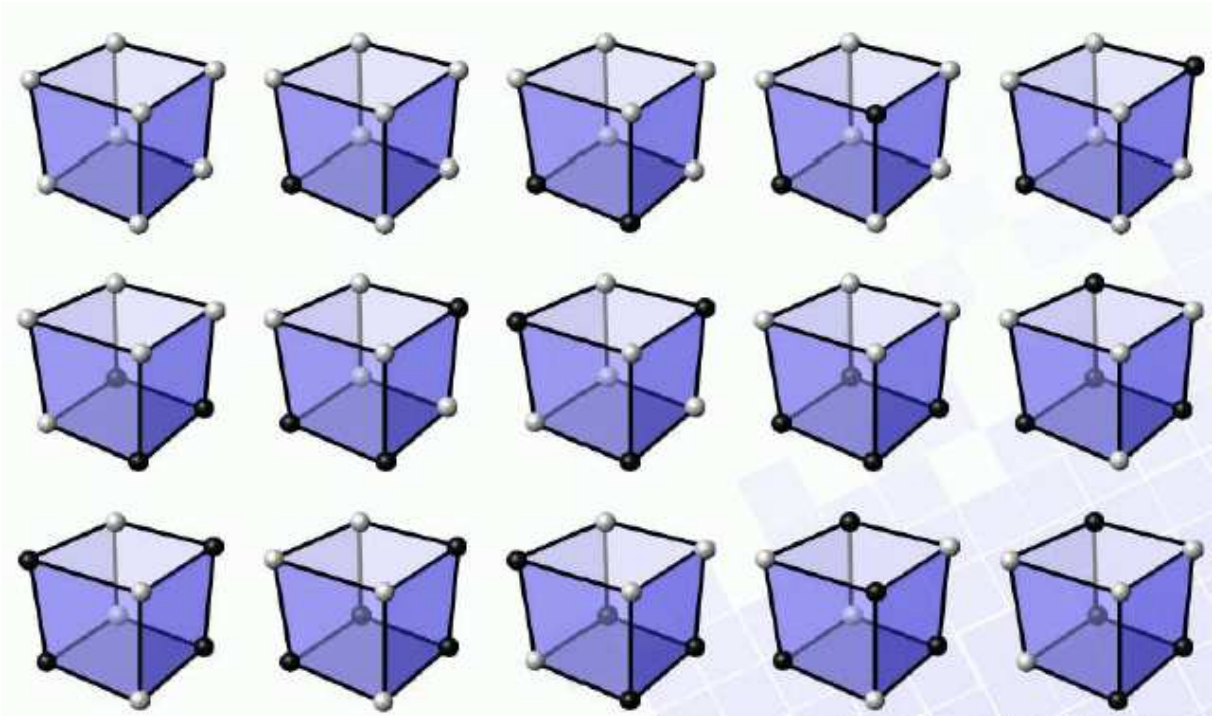
Zu einem bestimmten Index kann die *Triangulierung* in der Tabelle nachgeschaut werden.

Erstelle eine Tabelle

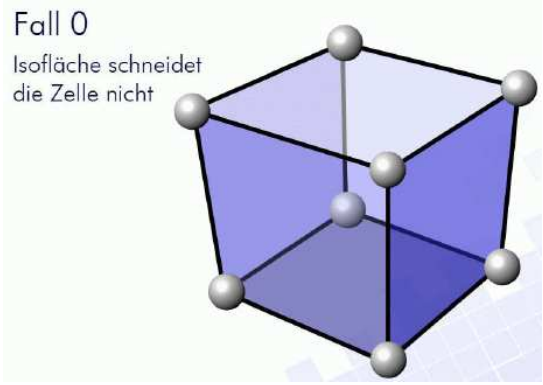


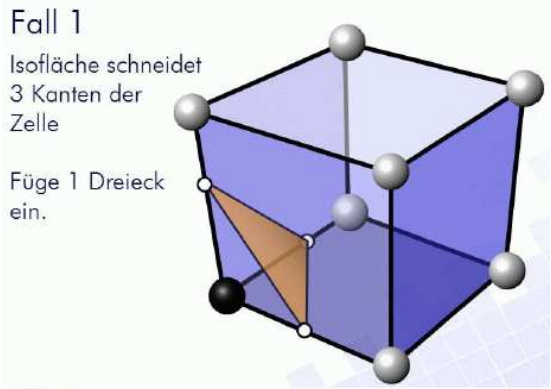
v_7	v_6	v_5	v_4	v_3	v_2	v_1	v_0
0	1	0	0	0	0	1	1

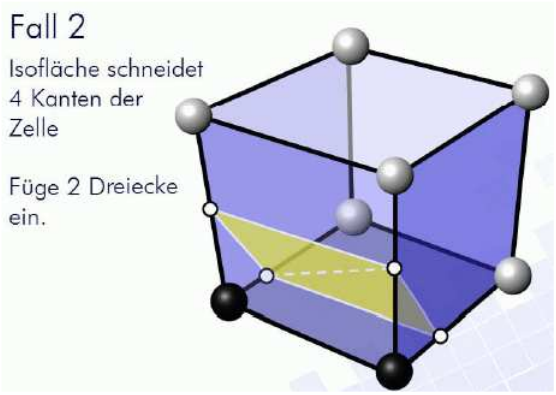
256 mögliche Fälle,
durch Ausnutzung von Symmetrien (01-Flipping, Rotationen, Spiegelungen) reduzieren sich diese auf 15 Fälle.

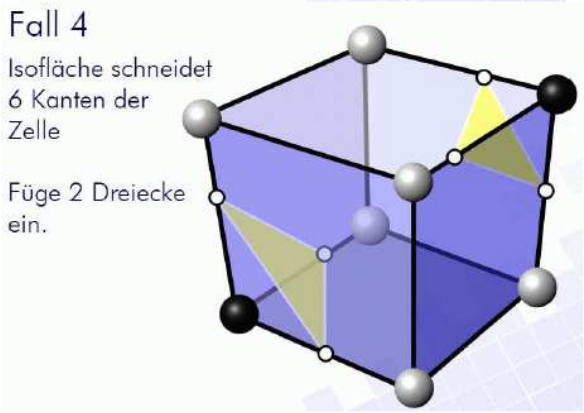


Beispiele:







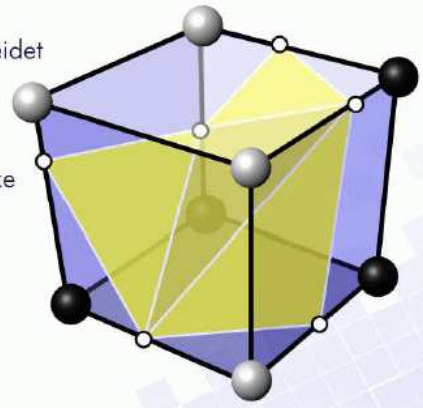


(mehrdeutig – siehe unten!)

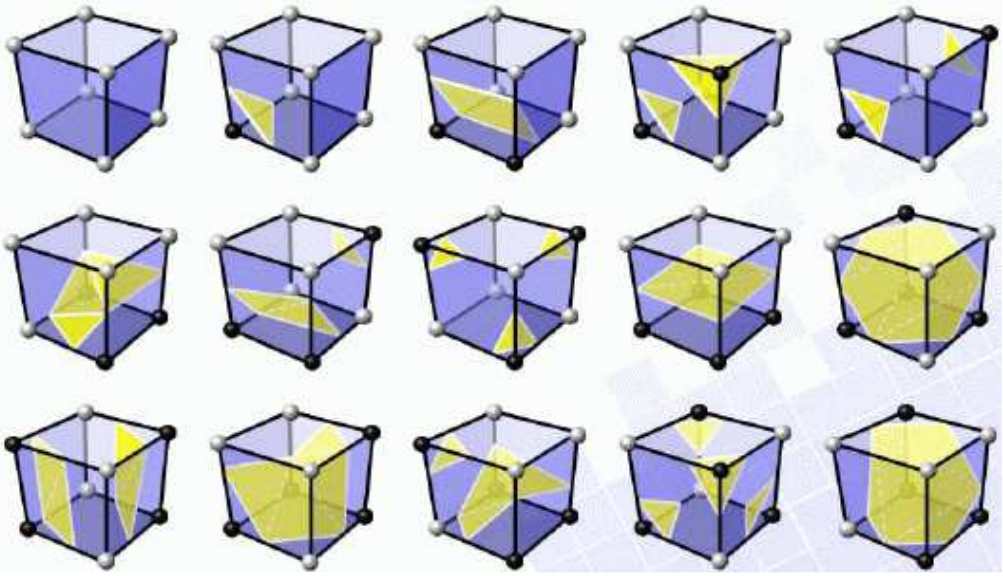
Fall 11

Isofläche schneidet 6 Kanten der Zelle

Füge 4 Dreiecke ein.



Triangulierung nach *Lorensen & Cline*

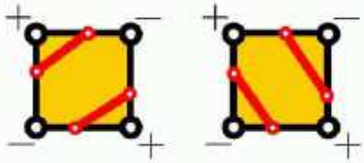


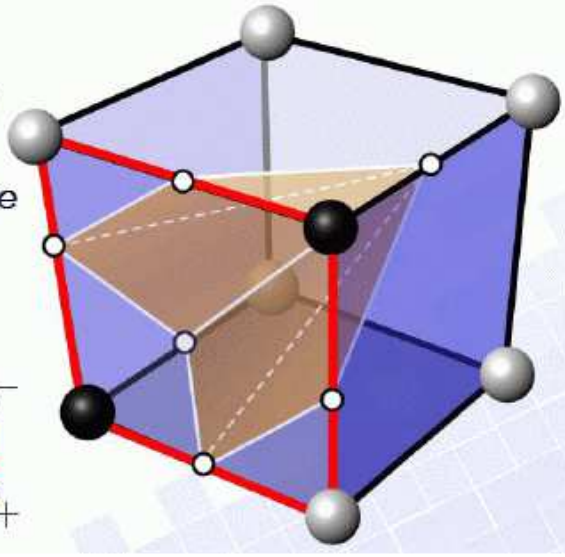
Mehrdeutigkeiten

Fall 3

Mehrdeutigkeit der Triangulierung!

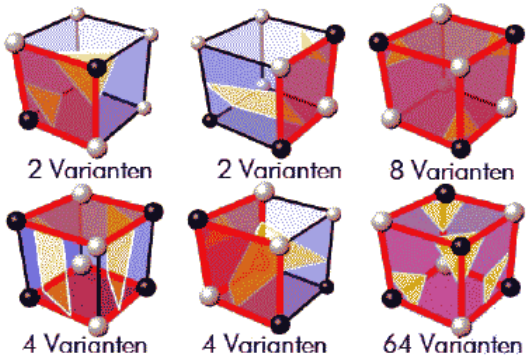
Wie kann man diese Mehrdeutigkeit auflösen?





Triangulierung nach *Lorensen & Cline*

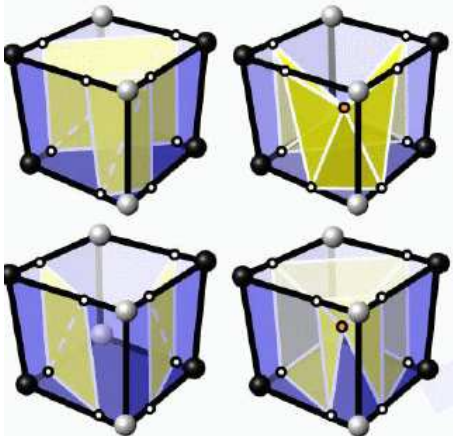
Mehrdeutige Triangulierungen



Mehrdeutige Flächen: Flächen, bei denen *jede* Kante durch die Isofläche geschnitten wird.

Die *Anzahl der mehrdeutigen Flächen* einer Zelle bestimmt die *Anzahl der möglichen* Triangulierungs-Varianten

Beispiel: Fall 10
mit Mehrdeutigkeit



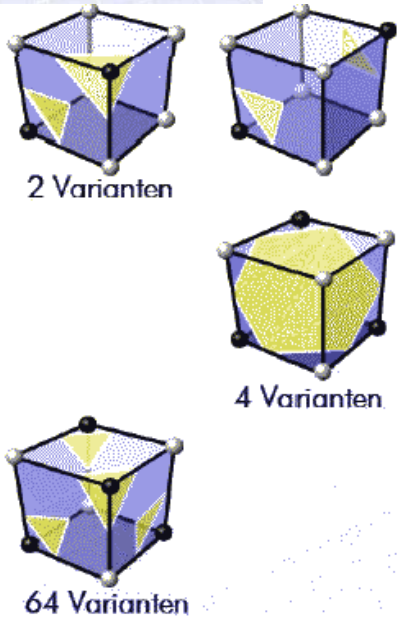
Allgemeine Vorgehensweise:

Betrachte mehrdeutige Flächen

Verwende *2D Decider* zur Entscheidung der Schnitt-Kanten

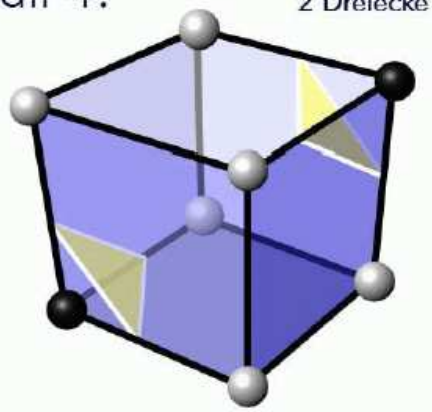
Tesselliere (Trianguliere) die entstehende(n) Randkurve(n) (Füge dazu evtl. zusätzliche Vertices ein.)

Mehrdeutige Triangulierungen

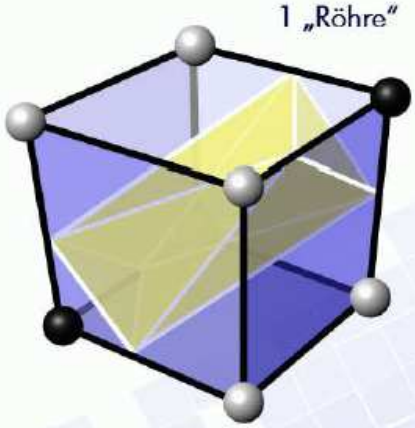


Fall 4:

2 Dreiecke



1 „Röhre"



In diesem Fall liegt eine *Mehrdeutigkeit der Zelle* vor, keine Mehrdeutigkeit der Fläche.
Es entsteht hier keine Diskontinuität der Gesamtfläche.

Für jede Zelle: Bestimme den Index der Zelle
(aus den Vorzeichen „ + " und „ - " der Vertices)

Eindeutige Zelle

Die Tabelle enthält direkt eine Liste der Kanten/Dreiecke für die Triangulierung der Zelle.

Mehrdeutige Zelle

Die Tabelle enthält eine Liste mehrdeutiger Flächen.

Für diese Flächen werden Entscheidungen für die Kanten getroffen.
(Midpoint/Asymptotic Decider)

Bestimme Sub-Index.

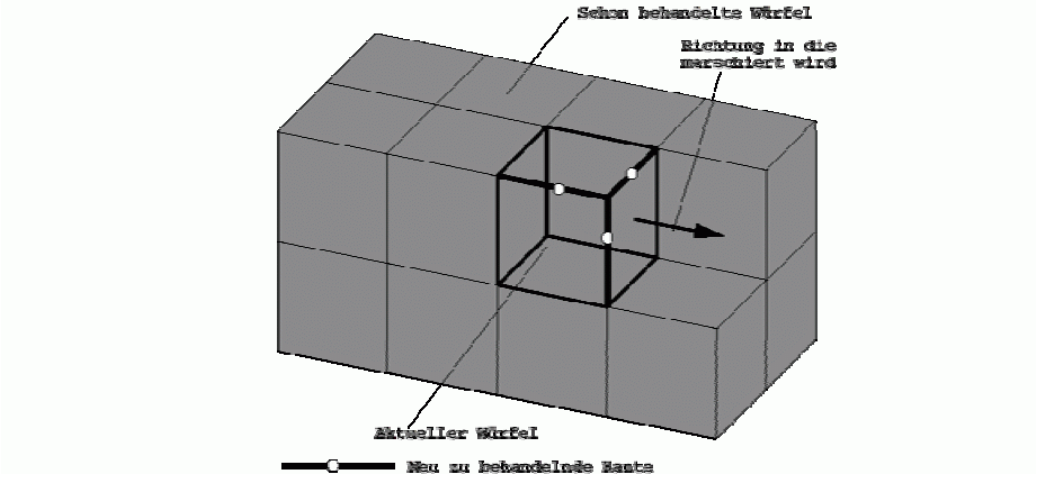
Eine Sub-Tabelle enthält die Kanten/Dreiecke für die einzelnen Varianten.

Beispiel: 2 Mehrdeutige Flächen:
00: Mitte negativ in beiden Flächen
10: Mitte positiv in erster Fläche
01: Mitte positiv in zweiter Fläche
11: Mitte positiv in beiden Flächen

(Rezk-S.)

"Marching Cubes":

Die aktuelle Zelle geht weiter (marches) zur nächsten Zelle im Gitter und fährt fort wie im Punkt 2. bzw. 1., bis das ganze Volumen abgelaufen wurde.

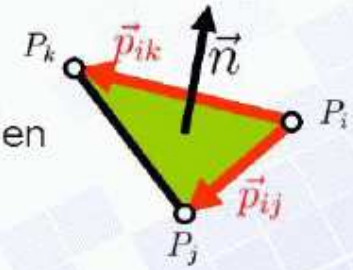


(Bartz 2005)

Ergänzungen:

Die extrahierten Oberflächen sollen später *beleuchtet* dargestellt werden. Dazu werden die *Normalenvektoren* der Oberfläche benötigt.

Methode 1: Bestimme zuerst die Isofläche und berechne anschließend die Normalen des entstandenen Dreiecksnetzes *(per Face oder per Vertex)*



Methode 2 (genauer): Berechne während der Generierung der Isofläche für jeden Vertex den **Gradientenvektor des Skalarfeldes.** *(Zentr. Differenzen + Interpolation)*

Schwachpunkte bei Marching Cubes:

- Es entstehen sehr viele kleine Dreiecke.
- Relativ lange Berechnungszeiten.

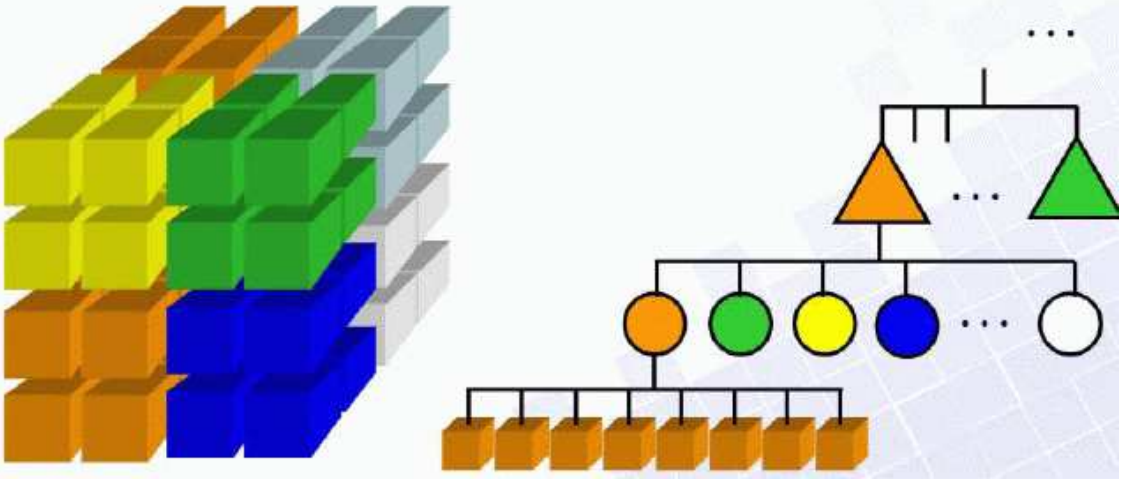
30 – 70 % der Rechenzeit wird bei der Bearbeitung von ***leeren Zellen*** benötigt.

Beschleunigungsverfahren:

Techniken, die es erlauben, große Bereiche im Skalarfeld, die von der Isofläche nicht geschnitten werden, gleich zu überspringen.

Beschleunigung

- *Idee:* Verwende eine Baum-Hierarchie (Octree),
  - Fasse 8 Zellen zu einem Block zusammen
  - Fasse 8 Blöcke zu einem neuen Block zusammen

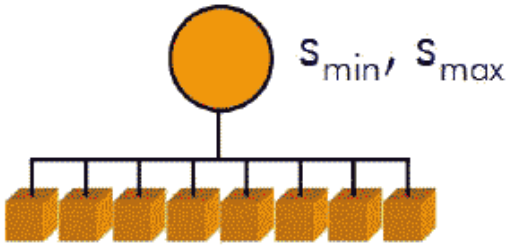


Octree-Verfahren

Beschleunigte Isoflächenberechnung:

Vorberechnung

Speichere in jedem Knoten den ***minimalen*** und ***maximalen Skalarwert*** seiner Kinder.



Isoflächenbestimmung

Traversiere den Baum (top-down) für einen bestimmten Isowert **s**.

Knoten mit

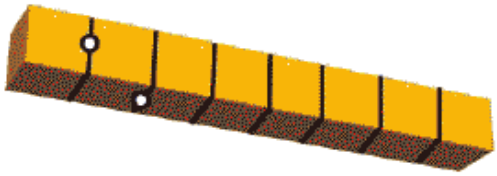
$s < s_{min}$ oder $s > s_{max}$

können sofort übersprungen werden.

Problematik: keine sequentielle Abarbeitung der Zellen mehr.

Wiederverwendbarkeit der berechneten Schnittpunkte (Position ***und*** Gradient)

Bei sequentieller Abarbeitung ist das kein Problem:



Bei sequentieller Reihenfolge, weiß ich genau wann eine bestimmte Zelle bearbeitet wird.

Bei jedem Schnittpunkt, den ich berechne, weiß ich sofort ob und wann er wieder gebraucht wird.

Bei Abarbeitung in beliebiger Reihenfolge müssen alle Schnittpunkte gespeichert werden. → ***Hashing***

(Rezk-Salama, o.J.)

Marching Tetrahedra

- Gleicher Ansatz für Tetraeder statt Hexaeder

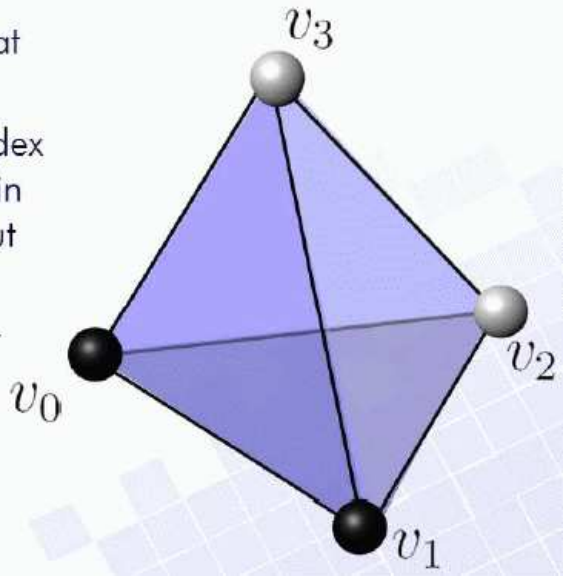
Jede Zellkonfiguration hat einen bestimmten Index.

Zu einem bestimmten Index kann die *Triangulierung* in der Tabelle nachgeschaut werden.

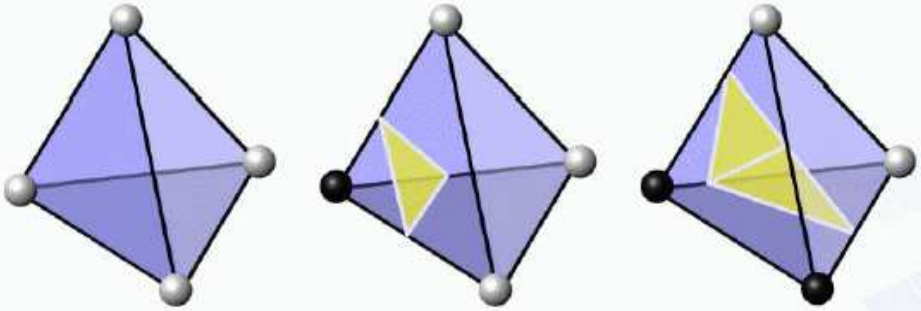
Es gibt $2^4 = 16$ verschiedene Möglichkeiten.

Erstelle eine Tabelle

v_3	v_2	v_1	v_0
0	0	1	1



- 3 mögliche Fälle (Symmetrie, Rotation)



- keine mehrdeutigkeiten Flächen
- keine mehrdeutigen Zellen

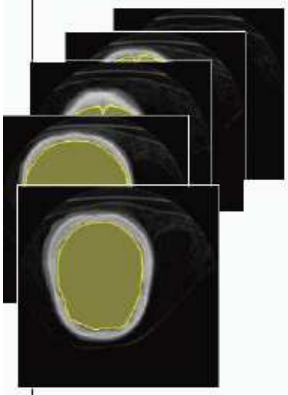
(Rezk-Salama, o.J.)

für Voxelgitter (Hexaeder) erzeugt dieses Verfahren (bei vorheriger Zerlegung der Voxel) mehr Dreiecke als der Marching-Cubes-Algorithmus (ca. doppelt so viele).

Oberflächenrekonstruktion

In vielen Anwendungsfällen soll aus dem Volumendatensatz eine Oberfläche extrahiert werden, die ***keine Isofläche*** ist.

- *Konturbasierte Segmentierung:*



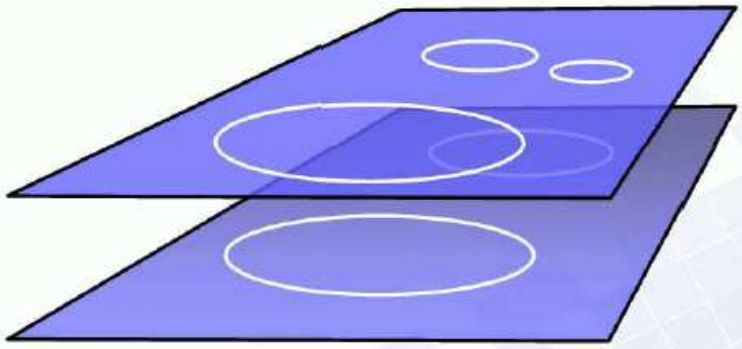
Ein bestimmtes im Datensatz enthaltenes Objekt (z.B das Gehirn) soll dargestellt werden.

Ein Segmentierungsverfahren liefert für jedes Schichtbild die Kontur des Objekts.

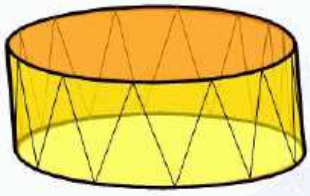
Aus diesen Konturlinien soll die Oberfläche des Objekts rekonstruiert werden.

- *Surface from Contours:*

3 verschiedene Probleme zu sind zu lösen:

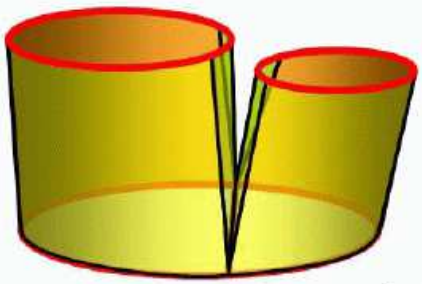


1. *Korrespondenz:* Welche Konturen in einer Schicht gehören zu welchen Konturen in der nächsten Schicht?



Wenn zwei korrespondierende Konturen gefunden wurden:

2. *Tiling:* Wie konstruiere ich die Mantelfläche zwischen den Konturen?

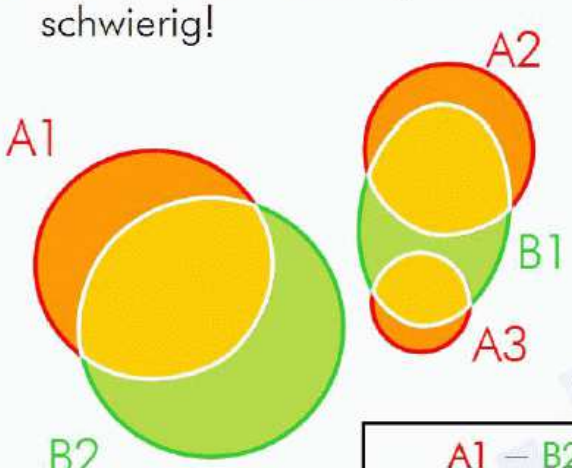


Wenn eine Kontur zwei Konturen in der nächsten Schicht entspricht:

3. *Branching:* Wie konstruiere ich die Mantelfläche im Falle einer Verzweigung?

Korrespondenz

- Automatische Lösung des Problems ist sehr schwierig!



A1 – B2
A2, A3 – B1

Einfache Methode:

- Überlagere die Konturen aus beiden Schichten und bestimme die Überlappungen.
- Überlappende Kurven werden miteinander verbunden (im Tiling-Schritt)

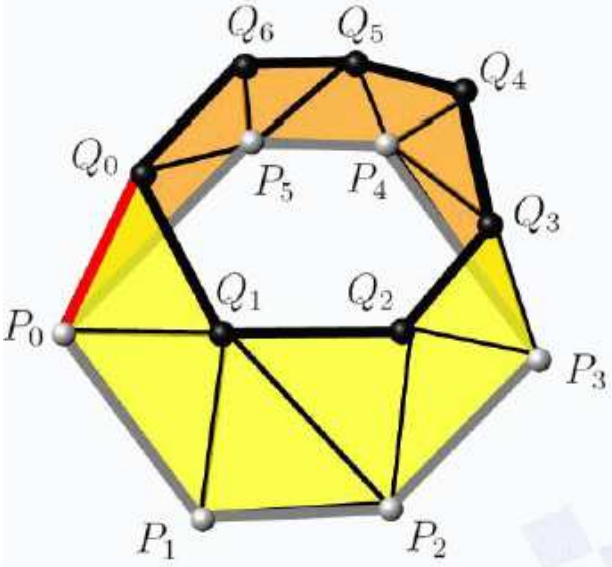
- Reale Probleme sind schwieriger und sehr stark abhängig von der Auflösung der Daten.
- *Problematik:*

keine Überlappung unerwünschte Überlappung

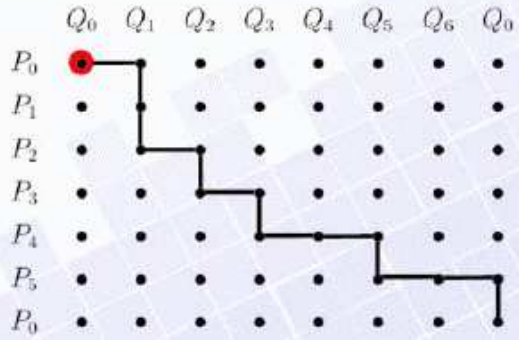
- Unterschiedliche Lösungsansätze in der Literatur
- Oft auch mit Benutzerinteraktion

Tiling

- *Finde die Triangulierung der Mantelfläche, die die Summe der Kantenlängen minimiert.*



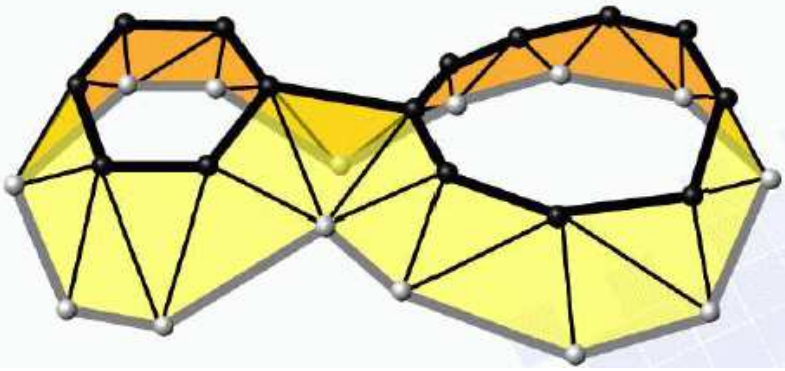
Graphensuche



Branching

- *Wie konstruiere ich die Mantelfläche bei Verzweigungen?*

Einfache Methode:
Verbinde die getrennten Konturen durch eine „virtuelle" Verbindungskante (Verdoppelung der Vertices)



Führe Tiling auf der erweiterten Kontur durch

Visualisierung von 3D Skalarfeldern

- *bisher:* Indirekte Volumenvisualisierung
  - Bestimme eine Oberfläche innerhalb des Skalarfeldes (z.B. Isofläche)
  - Stelle diese Fläche mittels traditionellen (Polygon-basierten) Rendering-Verfahren dar.
- Das Problem dabei:
  - Aufwändige Vorverarbeitung
  - Wie finde ich den richtigen Isowert?
  - Informationsverlust

Direkte Volumenvisualisierung

- Interpretiere das Skalarfeld als ein transparentes Medium.
- Betrachte das physikalische Model für die Lichtausbreitung in transparentem Medium. (Emission und Absorption von Licht)
- Verwende diese physikalischen Gleichungen um Bilder zu erzeugen.
- Vorteil: Nutze die gesamte Information
  - Transparenz und Semitransparenz
  - Darstellung von „unscharfen" Objekten

Physikalisches Modell

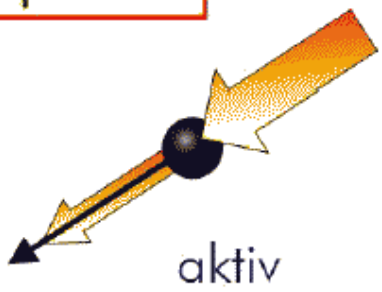
Betrachte einen Punkt im Raum und eine bestimmte Richtung:

Emission

Aktive Emission:
An dem betrachteten Punkt wird Lichtenergie erzeugt und in die betrachtete Richtung ausgesandt (Lichtquelle).

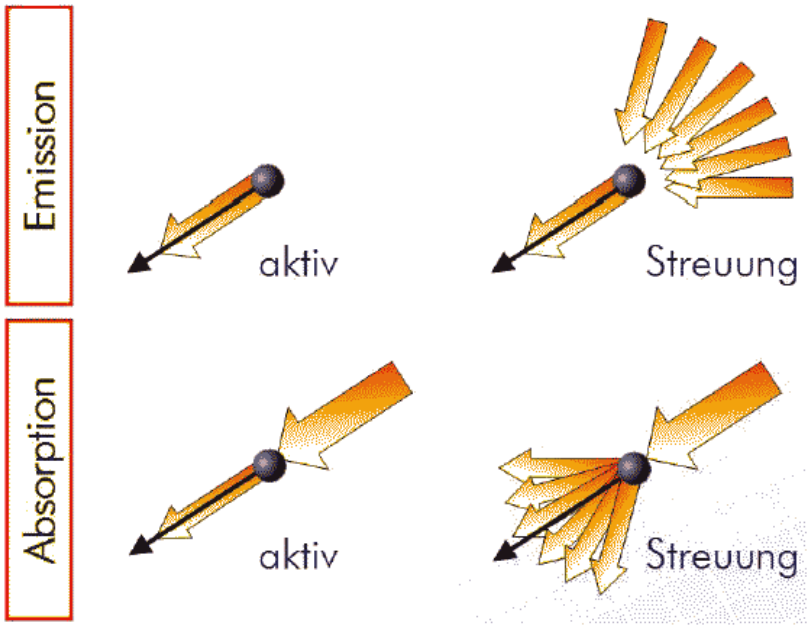
Emission durch Streuung:
Zuwachs an Lichtenergie in der betrachteten Richtung durch Brechung des Lichts, das aus anderen Richtungen einfällt.

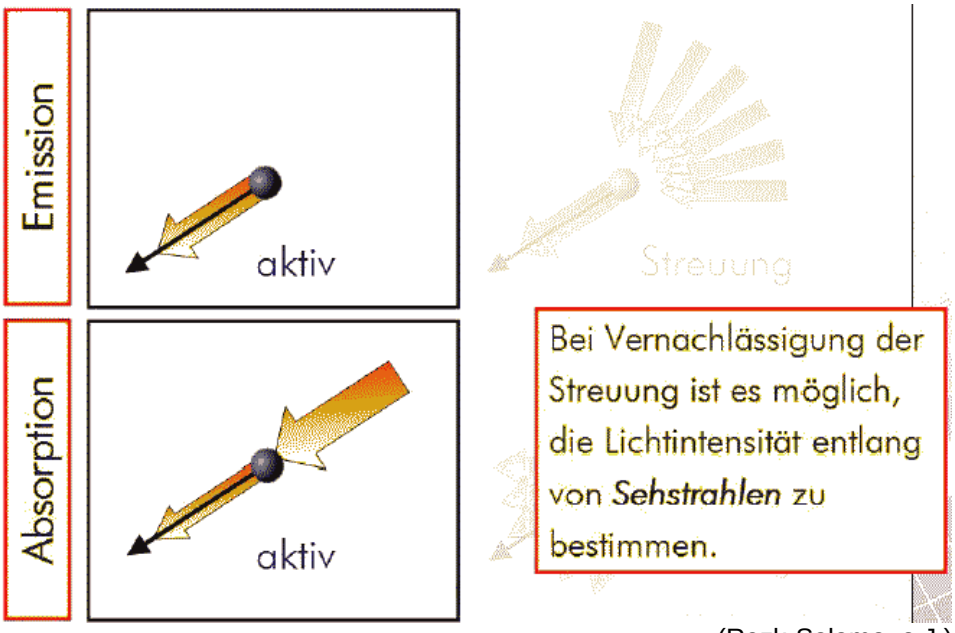
Absorption



Aktive Absorption:
An dem Punkt wird Licht absorbiert. Lichtenergie wird dem sichtbaren Spektrum entzogen und z.B. in Wärmenergie umgewandelt.

Absorption durch Streuung:
Verringerung der Lichtenergie in der betrachteten Richtung durch Brechung des Lichts in andere Richtungen.





(Rezk-Salama, o.J.)

- Im folgenden beschränken wir uns auf die **1-dimensionale Formulierung**.
- Wir **verzichten auf Streuung**, da sie das Modell verkompliziert und rechnerisch sehr teuer ist (geht sonst in Absorption (Out-Scatter) und Emission (In-Scatter) ein)
- Wir vernachlässigen den Einfluss des Mediums (Participating Media)
- Wir betrachten **keinen Einfluss von Wellenlängen, Beugung, Interferenz, Difraktion, Polarisation, …**

- Energieerhaltung
  (Emission – Absorption = Energie in Szene)
- Randbedingungen (Hintergrundlicht, …)
- Außerdem nehmen wir sogenannte **Low-Albedo-Situationen** an:
  - Keine/kaum Reflektion
  - Keine Brechung
- Damit bleiben: **Emission** (nur Quellterm) und **Absorption** (nur „True Absorption“ - ohne Streuung)
- Wird auch **Density-Emitter-Model** genannt
  [Sabella, ACM SIGGRAPH 1988]

Damit haben wir folgende Hierarchie:

- **Transporttheorie** des Lichts
- **Rendering-Gleichung**: verzichtet auf Streuung/Participating Media; Interaktion nur an Materialgrenzen/Oberflächen (zB. Ray Tracing)
- **Volume-Rendering-Gleichung**: Low-Albedo Situationen (Ray Casting / Volume Rendering)

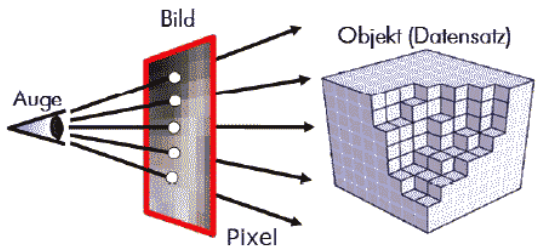
Komplexität ↓

(Bartz 2005)

2 Ansätze zur Lösung: Bildraum- und Objektraumverfahren.

Bildraumverfahren

Ansatz: Zerlege das Ergebnisbild in einzelne Pixel

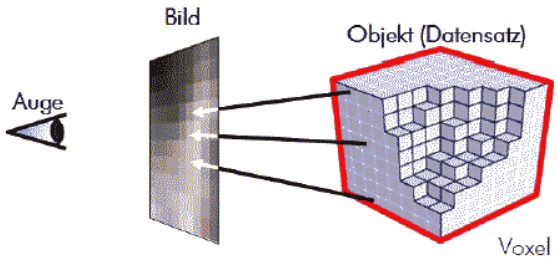


Für jeden Pixel → Bestimme den Beitrag des Volumens zur Farbe dieses Pixels

Objektraumverfahren

Alternativer Ansatz:

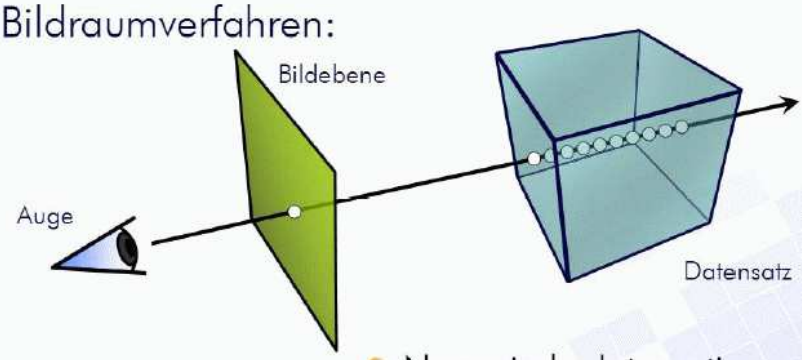
Zerlege das Objekt in einzelne Voxel



Bestimme den Beitrag des Voxels zum Ergebnisbild ⇐ Für jeden Voxel

Ray Casting

Bildraumverfahren:

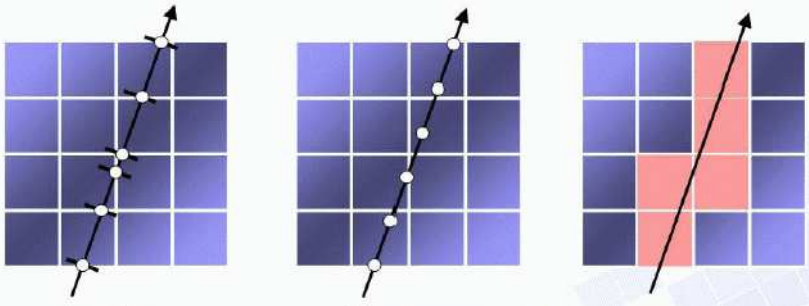


- Numerische Integration
- Neuabtastung

⇨ Hoher Rechenaufwand

Abtastung (Sampling)

Wie wird entlang des Sehstrahls diskretisiert?



Exakte Bestimmung der Schnittpunkte	Äquidistante Neuabtastung mit Interpolation	Rasterisierung z.B. Bresenham
Aufwändig	Einfach	Einfach, aber ungenau

Integration entlang des Strahls

Was wird integriert?

physikalisch korrekt: Emission und Absorption

Anfangsintensität (Emission) bei s_0

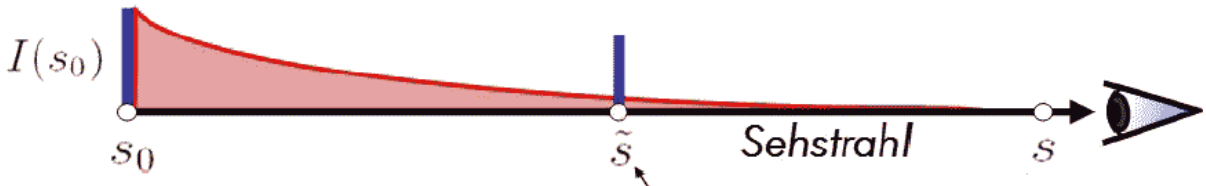
$$I(s) = I(s_0)$$

Ohne Absorption würde die Anfangsintensität ohne Verluste am Punkt **s** ankommen.

$$I(s) = I(s_0)\, e^{-\tau(s_0,s)}$$

Optische Tiefe $\boldsymbol{\tau}$
Absorbtion $\boldsymbol{\kappa}$

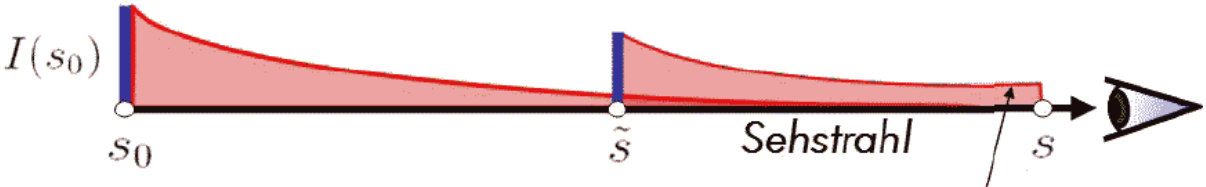
$$\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(s)\, ds.$$



An einem Punkt $\tilde{s}$ entlang des Sehstrahls wird zusätzlich Licht emittiert.

Aktive Emission bei $\tilde{s}$

$$I(s) = I(s_0)\, e^{-\tau(s_0,s)} + q(\tilde{s})$$



Absorbtion entlang der Strecke $\tilde{s}$ - s

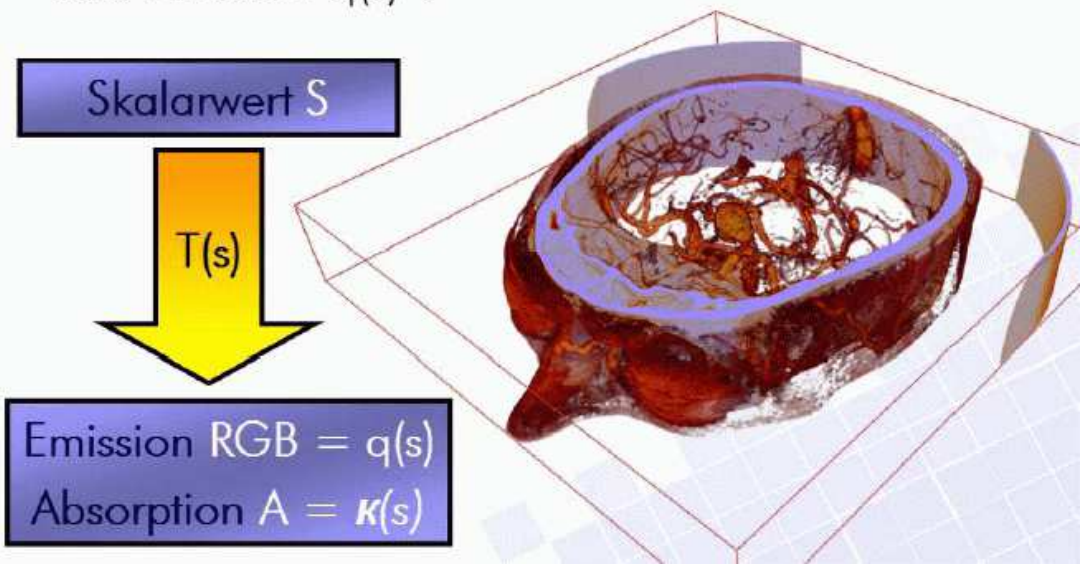
$$I(s) = I(s_0)\, e^{-\tau(s_0,s)} + q(\tilde{s})\, e^{-\tau(\tilde{s},s)}$$

An *jedem* Punkt $\tilde{s}$ entlang des Sehstrahls wird zusätzlich Licht emittiert.

$$I(s) = I(s_0)\, e^{-\tau(s_0,s)} + \int_{s_0}^{s} q(\tilde{s})\, e^{-\tau(\tilde{s},s)}\, d\tilde{s}$$

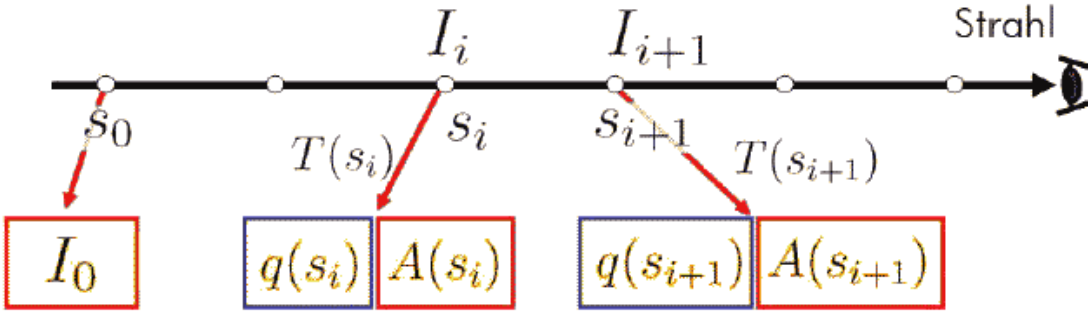
Transferfunktion

Woher bekomme ich Absorption $\kappa(s)$ und Emission q(s) ?



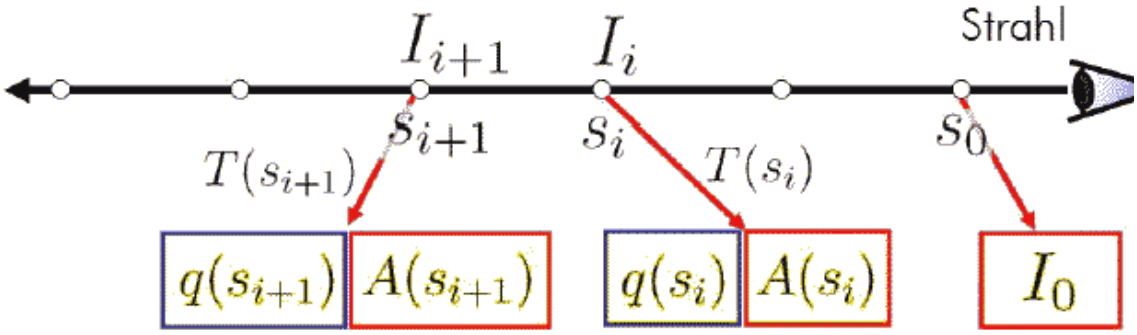
Diskrete Lösung des Integrals

Werte das Skalarfeld an diskreten Punkten entlang des Strahls aus:



Back-to-front Compositing mit $\alpha = A(s_{i+1})$

$$I_{i+1} = \alpha \cdot q(s_{i+1}) + (1 - \alpha)\, I_i$$
$$= q(s_{i+1}) \text{ OVER } I_i$$



Front-to-back Compositing.

$$I_{i+1} = I_i + T_i \cdot A(s_i) \cdot q(s_i)$$

$$T_{i+1} = T_i \cdot (1 - A(s_i))$$

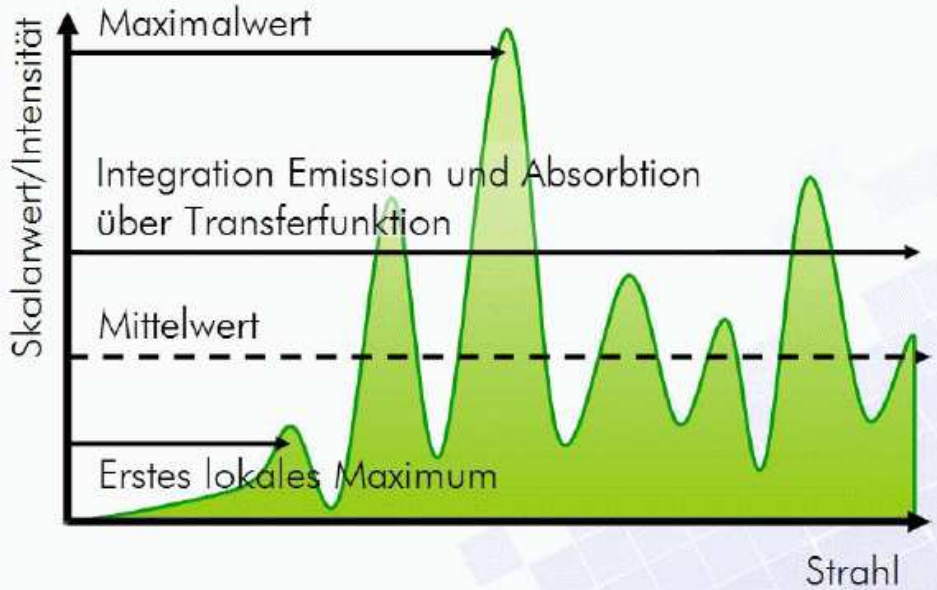
$$I_0 = 0; \quad T_0 = 1;$$

Early Ray Termination

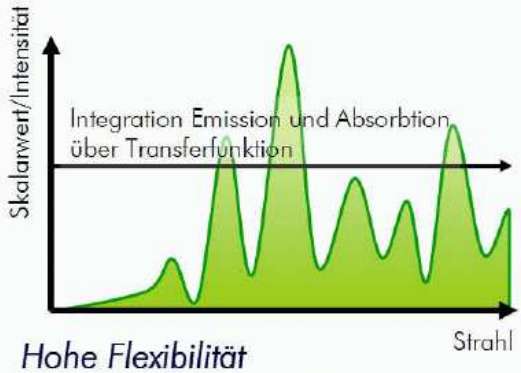
Berechnung abbrechen, wenn

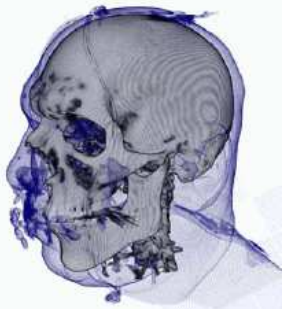
$$T_i \approx 0$$

Alternative Compositing-Verfahren



Compositing - Integration



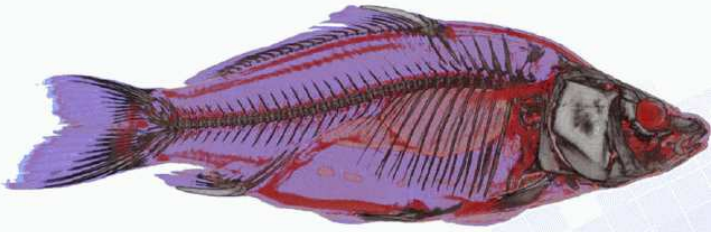


Hohe Flexibilität

- semi-transparente Darstellung, „weiche Kanten".
- Bereiche im Datensatz können unterschiedlich dargestellt werden.
- *Nachteil:* Einstellung der Transferfunktion ist aufwändig.

Beispiel Transferfunktion

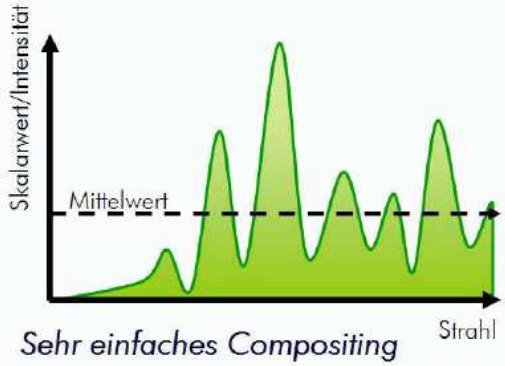
Karpfen (Computertomographie)

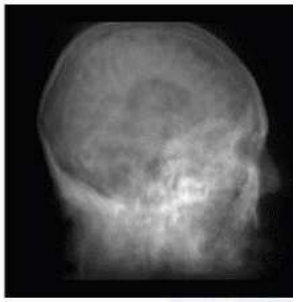


Die Transferfunktion erlaubt die *Klassifikation* der Daten.

- Welche Bereiche sollen welche Farbe haben?
- Welche Bereiche sollen transparent/semi-transparent oder opak dargestellt werden?

Compositing - Mittelwert

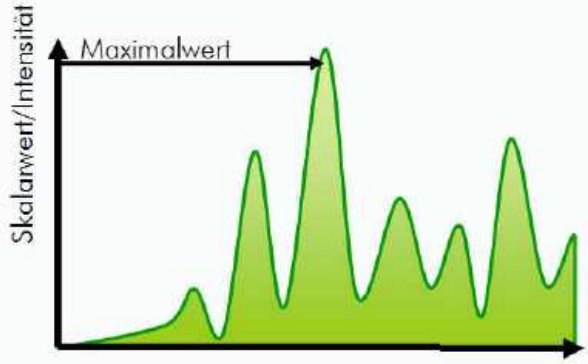




Sehr einfaches Compositing

- Ergebnisbilder sind *Röntgenbildern* sehr ähnlich.
- *Nachteil:* Geringe Flexibilität, keine Klassifikation
- *Vorteil:* Einstellung einer Transferfunktion ist nicht notwendig.

Compositing - MIP



Maximum Intensity Projection (MIP)

- gut geeignet für Blutgefäße mit Kontrastmittel.
- *Vorteil:* Einstellung einer Transferfunktion nicht nötig
- *Nachteil:* Irreführende Tiefenrelationen!
  (Tiefeninformation geht verloren!)

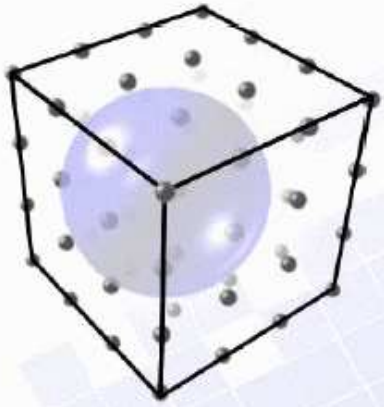
Zusammenfassung Ray-Casting

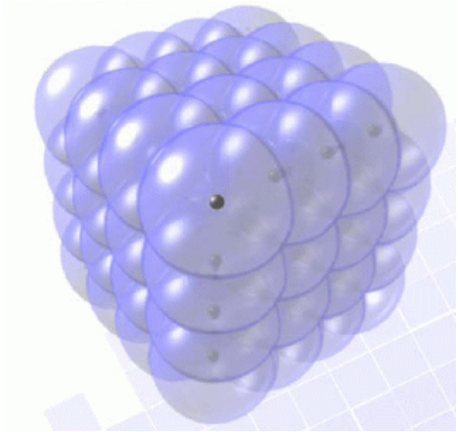
- Bildraumverfahren: Bestimme einen Strahl für jeden Pixel des Ergebnisbildes
  - Berechne die Farbe des Pixels mittels
    - Integration von Emission und Absorption
    - Maximum Intensity Projektion ODER
    - Summation der Skalarwerte
  - Diskretisiere entlang des Strahls
    - Exakte Bestimmung der Schnittpunkte
    - Resampling äquidistant ODER
    - Rasterisierung (z.B. 3D Bresenham)

Objektraumverfahren

Jeder diskrete Datenpunkt im Volumen beeinflußt die optischen Eigenschaften des Raumes in einem bestimmten Bereich.

Dieser Bereich wird durch seine *charakteristische Funktion* $h(x, y, z)$ beschrieben.
(z.B. durch eine Kugel)



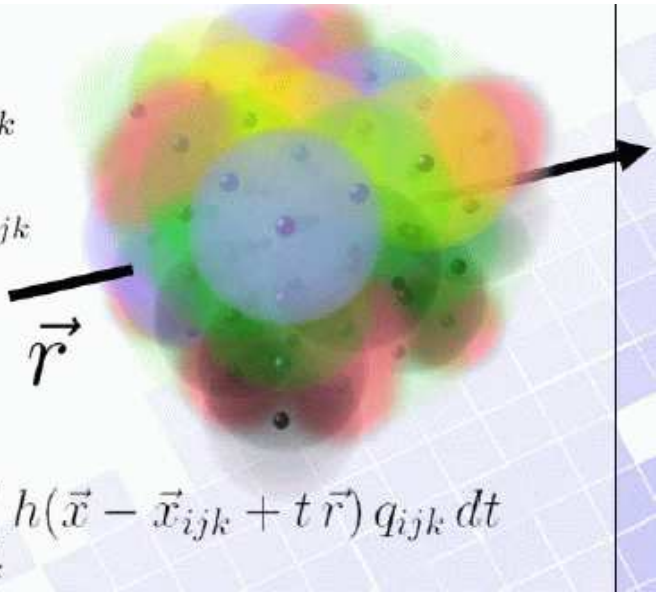


Das komplette Skalarfeld lässt sich beschreiben als:

$$S(\vec{x}) = \sum_{i,j,k} h(\vec{x} - \vec{x}_{ijk})\, I_{ijk}$$

Die Transferfunktion wird nun angewendet: $I_{ijk} \rightarrow q_{ijk}$

$$S_q(\vec{x}) = \sum_{i,j,k} h(\vec{x} - \vec{x}_{ijk})\, q_{ijk}$$



Das Integral der Emissionen entlang eines Sehstrahles:

$$\int S_q(\vec{x} + t\,\vec{r})\, dt = \int \sum_{i,j,k} h(\vec{x} - \vec{x}_{ijk} + t\,\vec{r})\, q_{ijk}\, dt$$

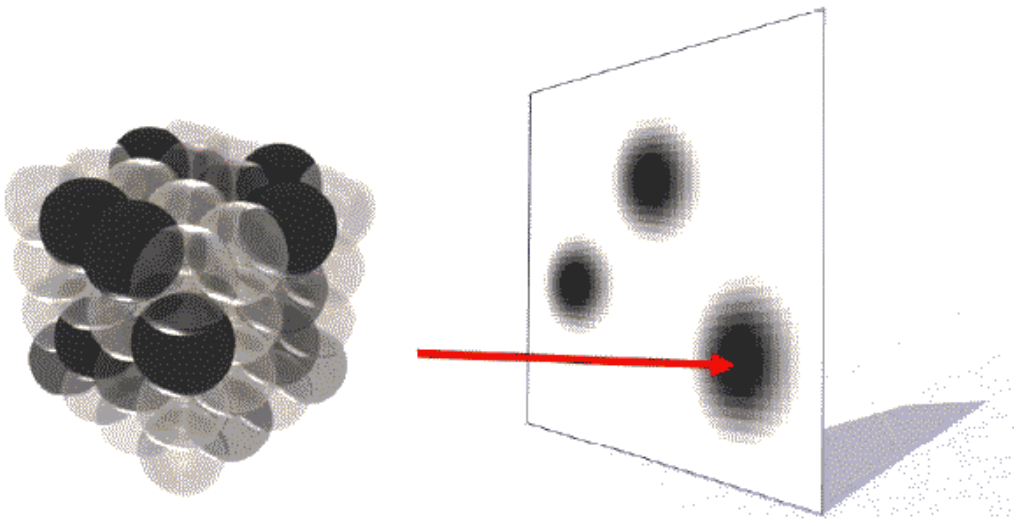
$\Rightarrow$

$$\int S_q(\vec{x} + t\,\vec{r})\, dt = \sum_{i,j,k} \left(\int h(\vec{x} - \vec{x}_{ijk} + t\,\vec{r})\, dt \cdot q_{ijk} \right)$$

Splatting

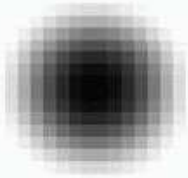
$$\int S_q(\vec{x} + t\,\vec{r})\, dt = \sum_{i,j,k} \left(\int h(\vec{x} - \vec{x}_{ijk} + t\,\vec{r})\, dt \cdot q_{ijk} \right)$$

Der Beitrag eines Abtastpunkts zum Ergebnisbild *(footprint)*

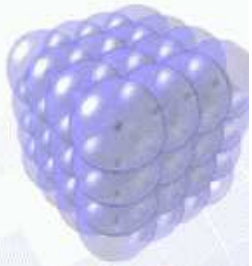


Splatting-Algorithmus

Das Abbild („footprint") eines einzelnen Voxels kann vorberechnet, und als eine Maske gespeichert werden.

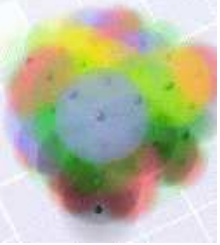


Alle Voxel werden in der Reihenfolge von hinten nach vorne auf die Bildebene projiziert.



Für jeden Voxel wird nun

- der Mittelpunkt auf die Bildebene projiziert,
- die Maske mit dem Emissionswert gewichtet
- mit dem **over**-Operator verknüpft.



Der Shear-Warp Algorithmus

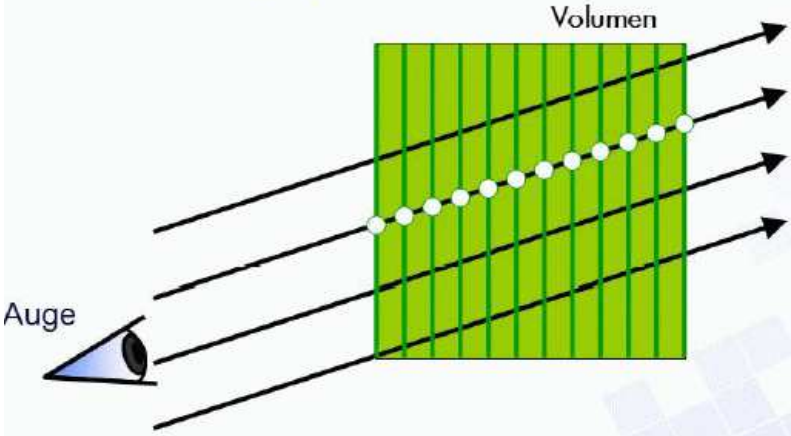
Hybridverfahren: Mischung zwischen Bildraum- und Objektraumverfahren

❑ Das schnellste der reinen Software-Verfahren.

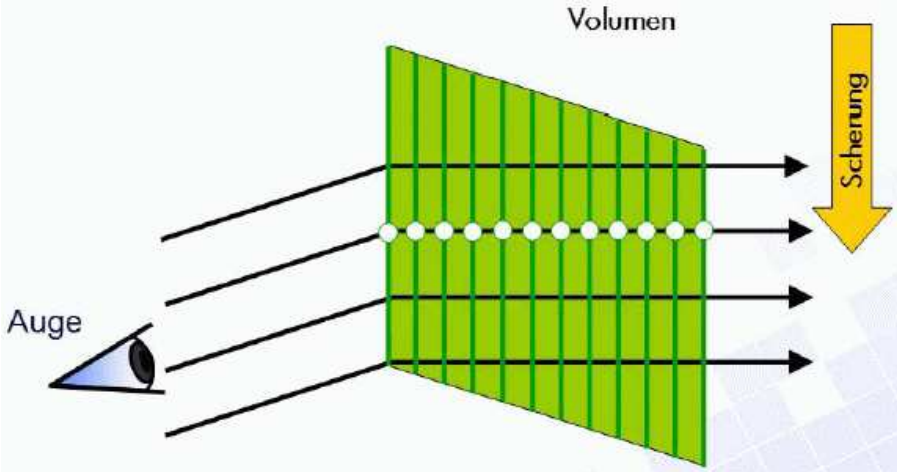
Berechnung des Ergebnisbildes in zwei Schritten:

- **Objektraumverfahren:** Erzeuge zunächst ein verzerrtes Zwischenbild
- **Bildraumverfahren:** Entzerre das Zwischenbild in einem 2D Nachbearbeitungsschritt

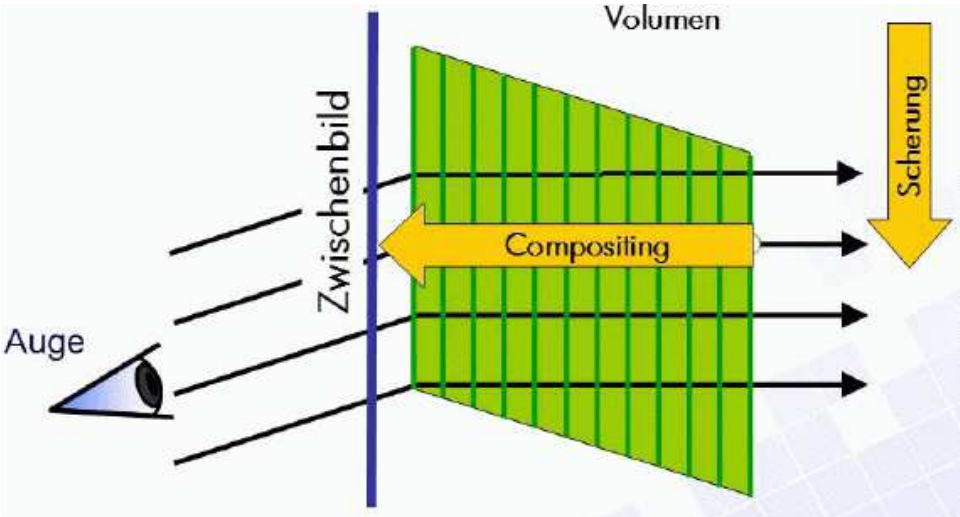
1. Fall: Parallelprojektion



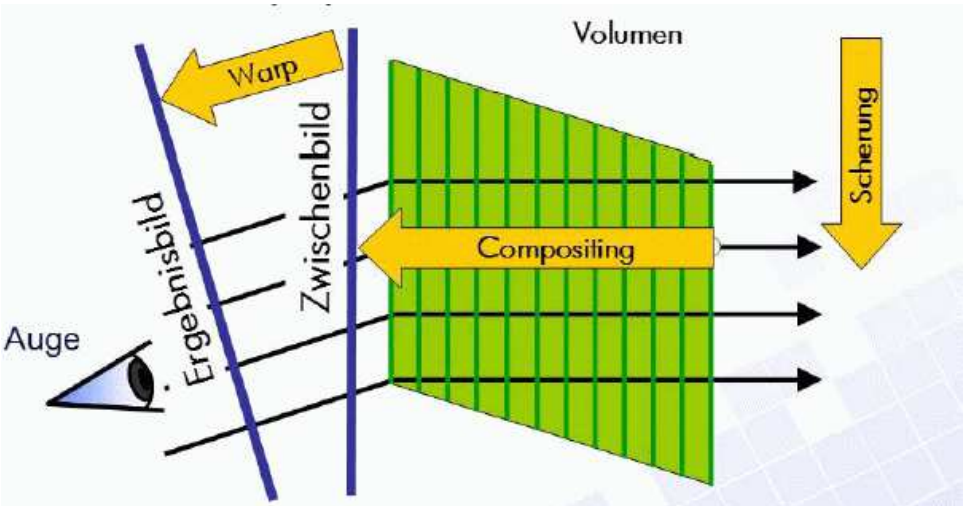
Schritt 1: Zerlege das Volumen in achsenparallele Schichten



Schritt 2: Bestimme eine Scherung des Raumes so daß die Sehstrahlen senkrecht auf den Schichten stehen.

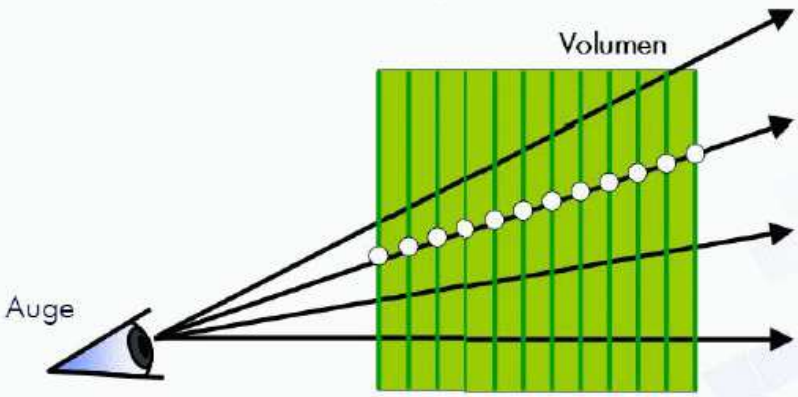


Schritt 3: Erzeugung eines (verzerrten) Bildes als Zwischenergebnis

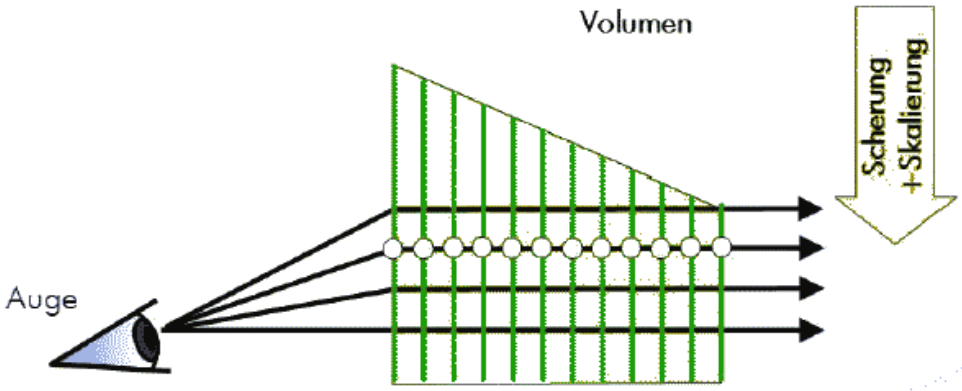


Schritt 4: Entzerrung (Warp) des Zwischenbildes liefert das Ergebnisbild.

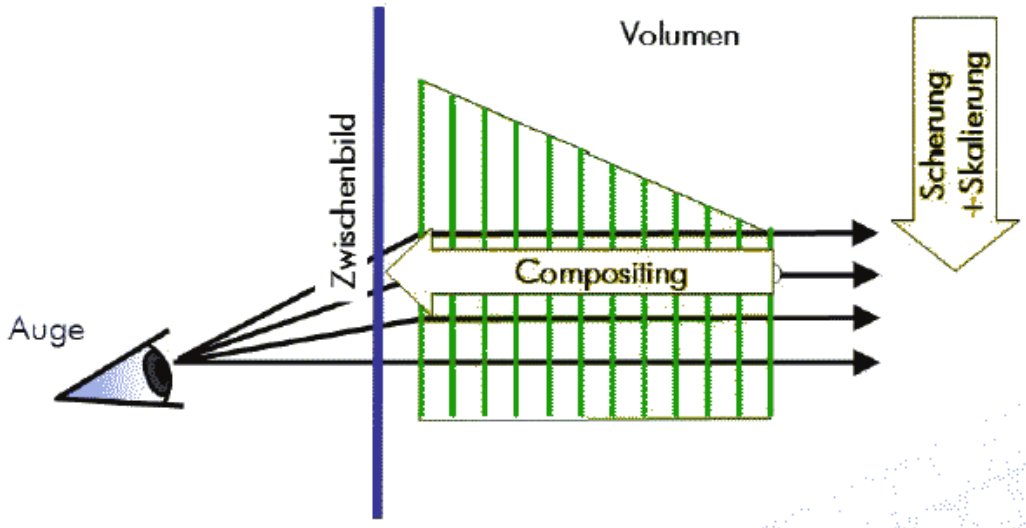
2. Fall: Perspektivische Projektion



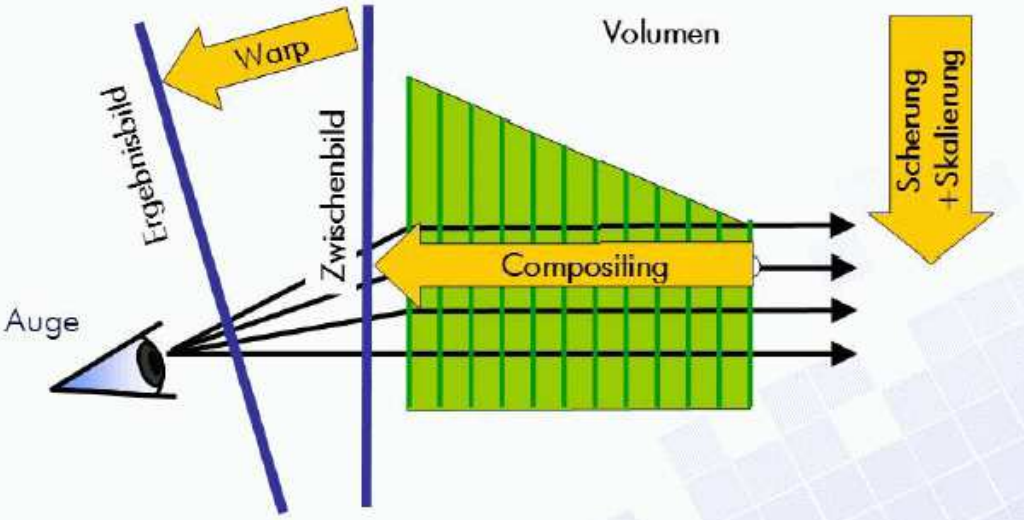
Schritt 1: Zerlege das Volumen in achsenparallele Schichten



Schritt 2: Scherung und Skalierung, so daß die Sehstrahlen senkrecht auf den Schichten stehen.



Schritt 3: Erzeugung eines (verzerrten) Bildes als Zwischenergebnis



Schritt 4: Entzerrung (Warp) des Zwischenbildes liefert das Ergebnisbild.

Mathematisch gesehen:

- Faktorisierung der Viewing Matrix in eine 3D-Scherung und eine 2D-Verzerrung

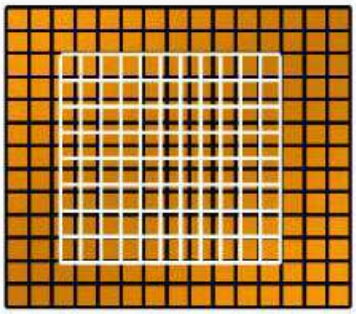
$$\mathbf{M}_{\text{view}} = \mathbf{P} \cdot \mathbf{S} \cdot \mathbf{M}_{\text{warp}}$$

$\mathbf{M}_{\text{view}}$ = Viewing Matrix
$\mathbf{S}$ = 3D Scherung
$\mathbf{M}_{\text{warp}}$ = 2D Verzerrung
$\mathbf{P}$ = Permutationsmatrix

Was bringt das ganze?

Nützliche Eigenschaften:

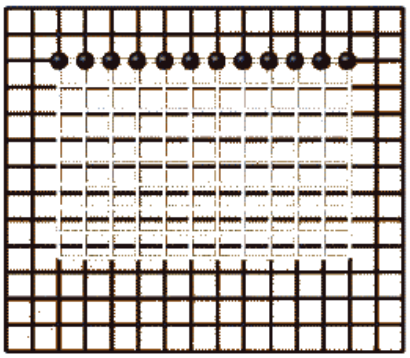
1. Scanlines der Pixel im Zwischenbild verlaufen parallel zu den Scanlines der Voxel im Datensatz.
2. Alle Voxel in einer Schicht werden mit dem gleichen Faktor skaliert.



Vorteil: Interpolationsgewichte müssen nicht für jeden Punkt einzeln berechnet werden.

Sie können Zeilen- und Spaltenweise berechnet werden.

3. *Nur bei Parallelprojektion:* Alle Voxelschichten haben den gleichen Skalierungsfaktor.



Dieser Faktor kann frei gewählt werden.

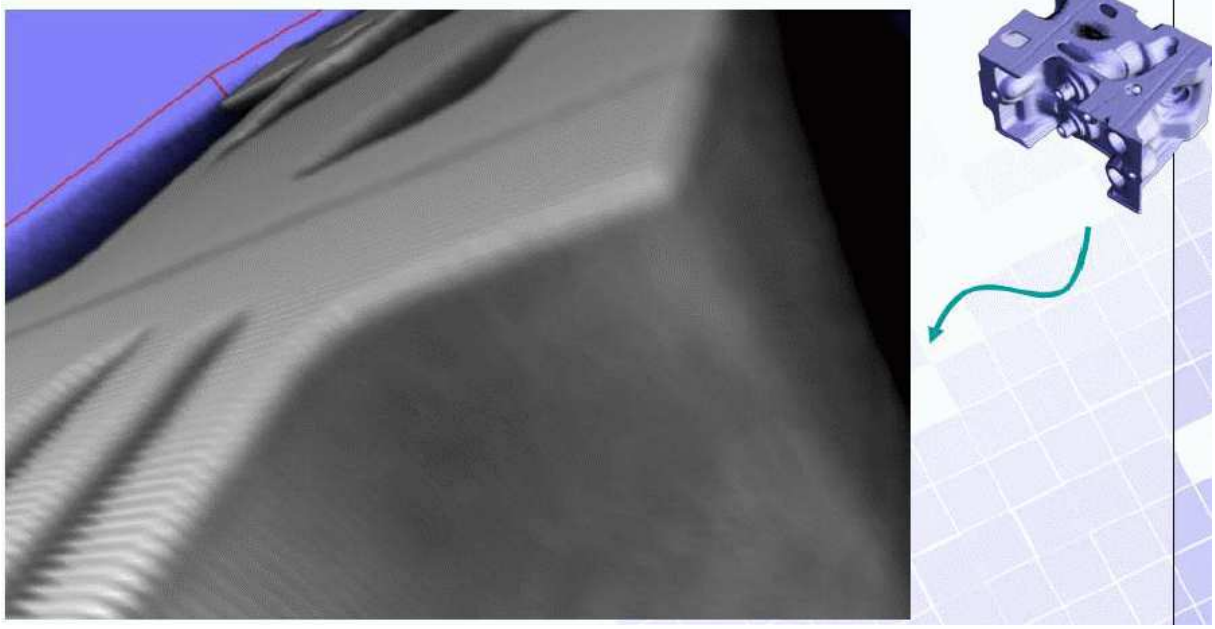
Wähle den Faktor so, dass für eine bestimmte Voxel-Scanline eine 1-zu-1 Abbildung zwischen Voxeln und Pixeln im Zwischenbild gibt.

Vergleich

Shear-Warp

Ray-Casting

- Nachteil – Visuelle Bildartefakte (bei starkem Zoom)



Zusammenfassung Shear Warp

Mischung aus Bildraum- und Objektraumverfahren:

1. Erzeuge ein verzerrtes Zwischenbild,
2. Entzerre dieses anschließend in einem 2D Schritt.

- Zerlege die Viewing-Matrix in
  - Scherung (+ Skalierung)
  - Permutationsmatrix
  - 2D Warp-Matrix
- Vorteile:
  - Effizientere Interpolationsberechnung
  - Schnellstes Software-Verfahren
- Nachteile:
  - 3-facher Speicherbedarf für 3 Schichtstapel.
  - visuelle Artefakte

(Rezk-Salama, o.J.)