

Visualisierung

Vorlesung im Wintersemester 2016/17 2) Datenquellen und -repräsentation

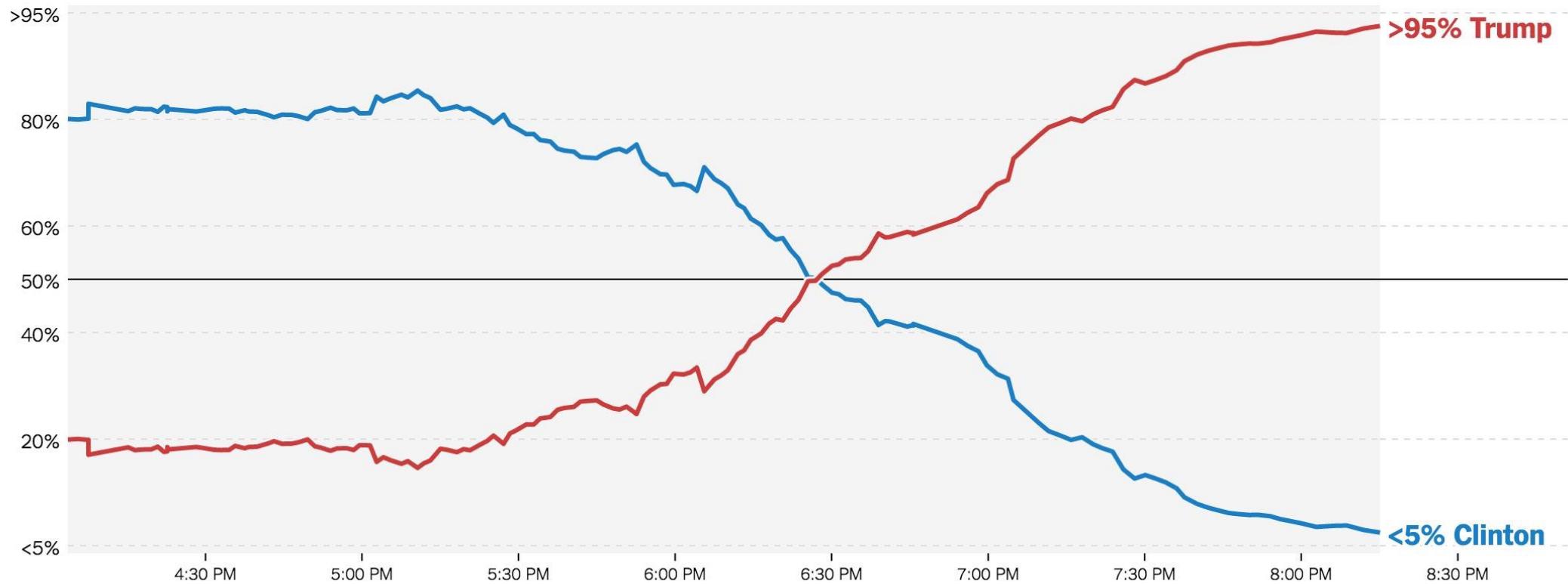
Boris Neubert und Carsten Dachsbacher
Institut für Visualisierung und Datenanalyse
Karlsruher Institut für Technologie



Visualisation of the day



Chance of Winning Presidency





Hillary Clinton

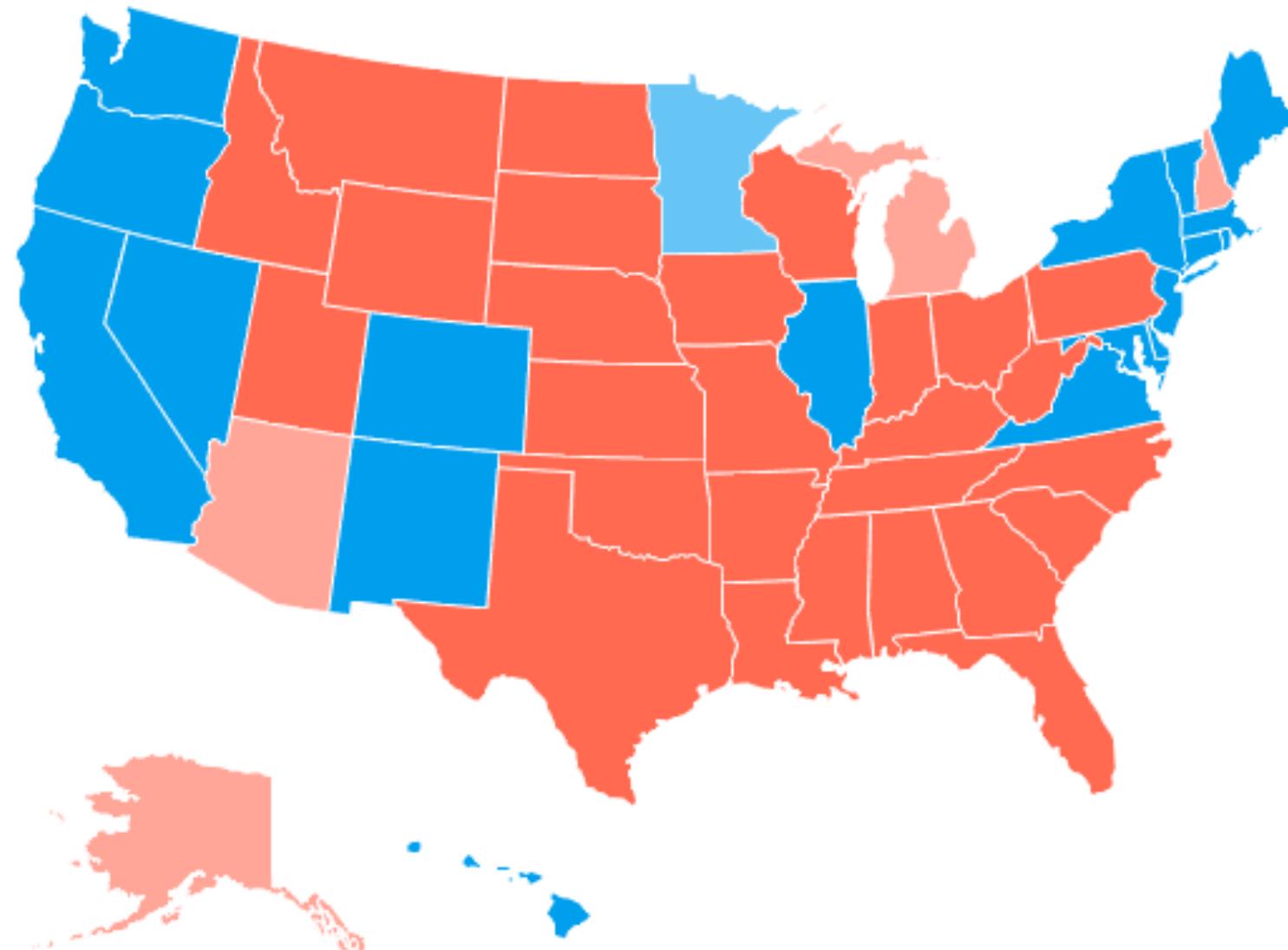
218

270 für Mehrheit



Donald Trump

276





Hillary Clinton

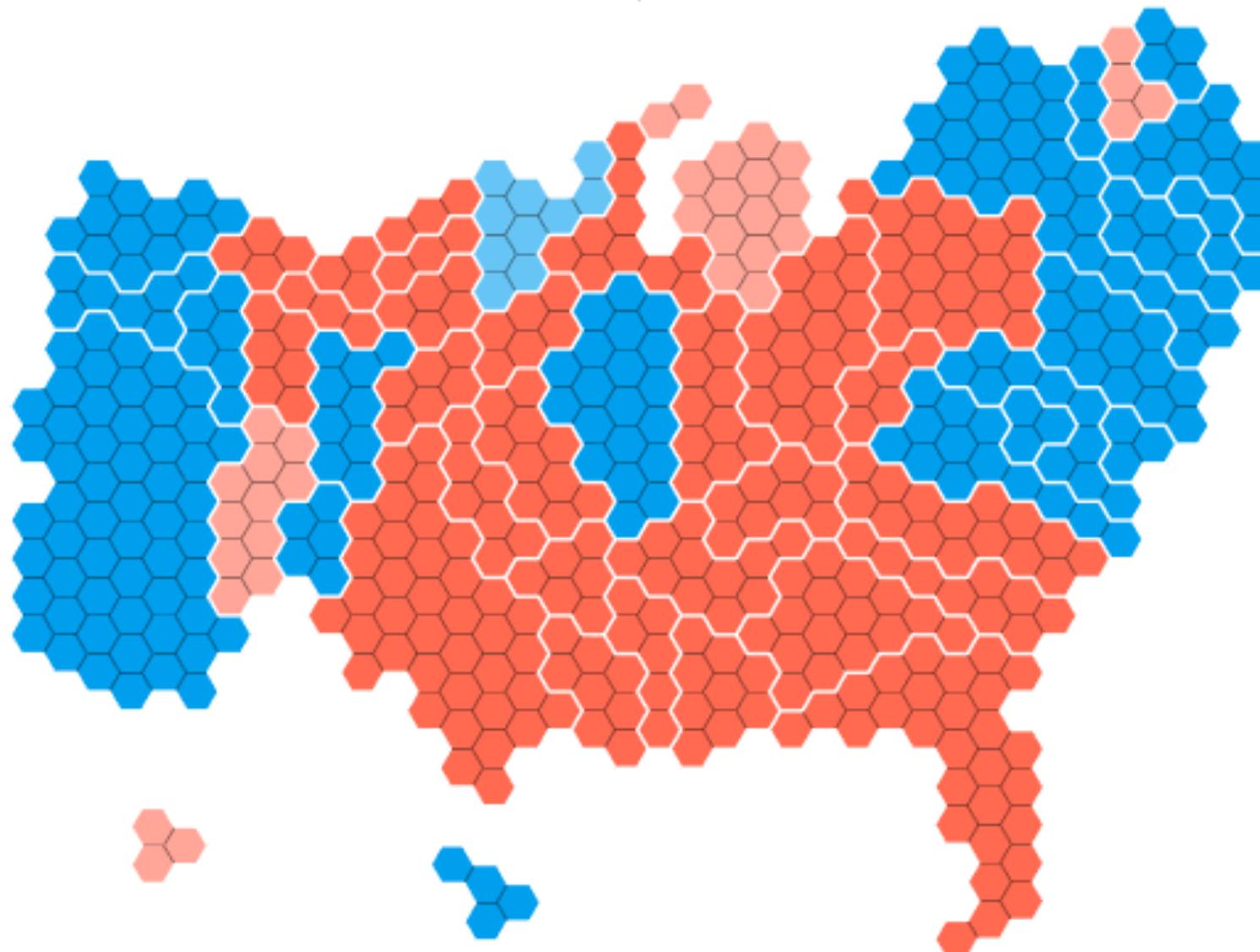
218

270 für Mehrheit



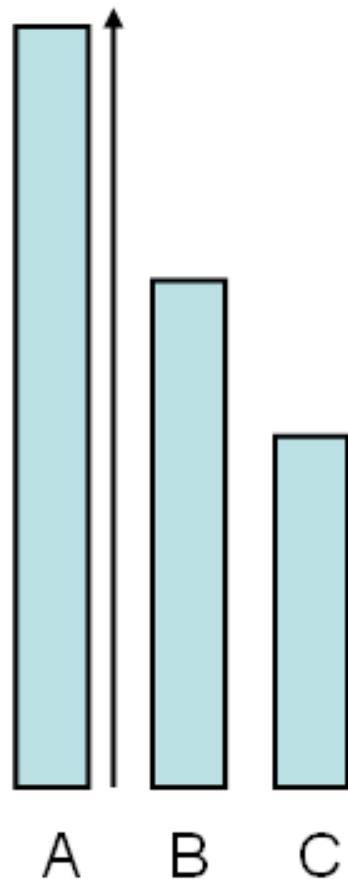
Donald Trump

276

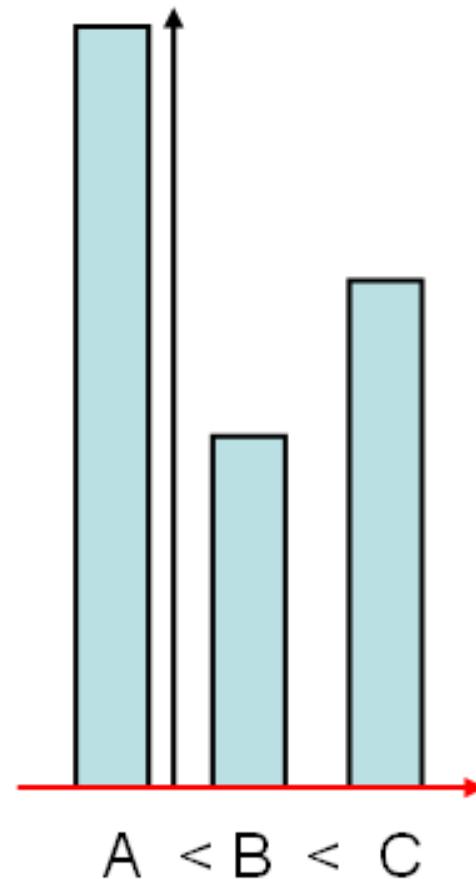


- Skalen
- Beobachtungsraum (data domain, unabhängige Variablen), Typ und Dimension der beobachteten Variablen (abh. Variablen)

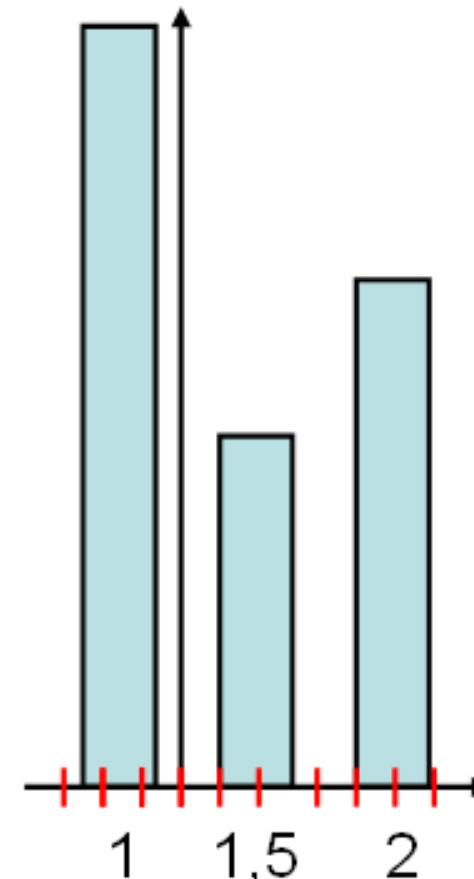
Nominalskala



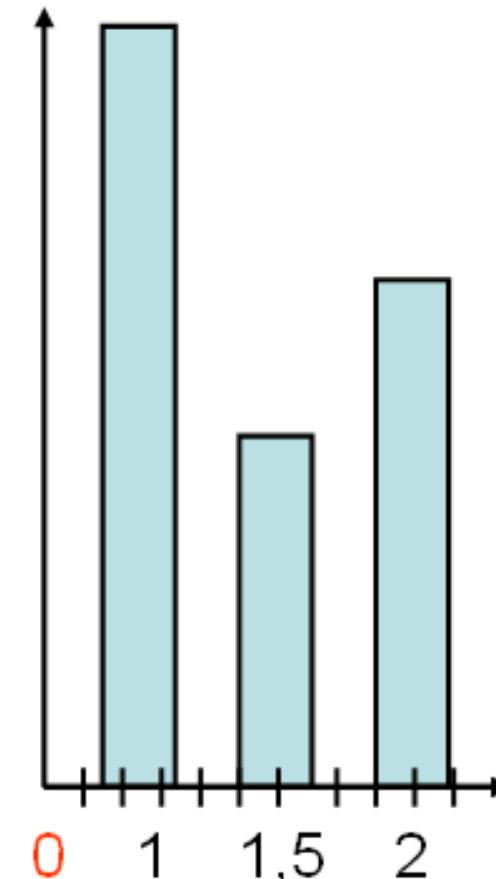
Ordinalskala



Intervallskala

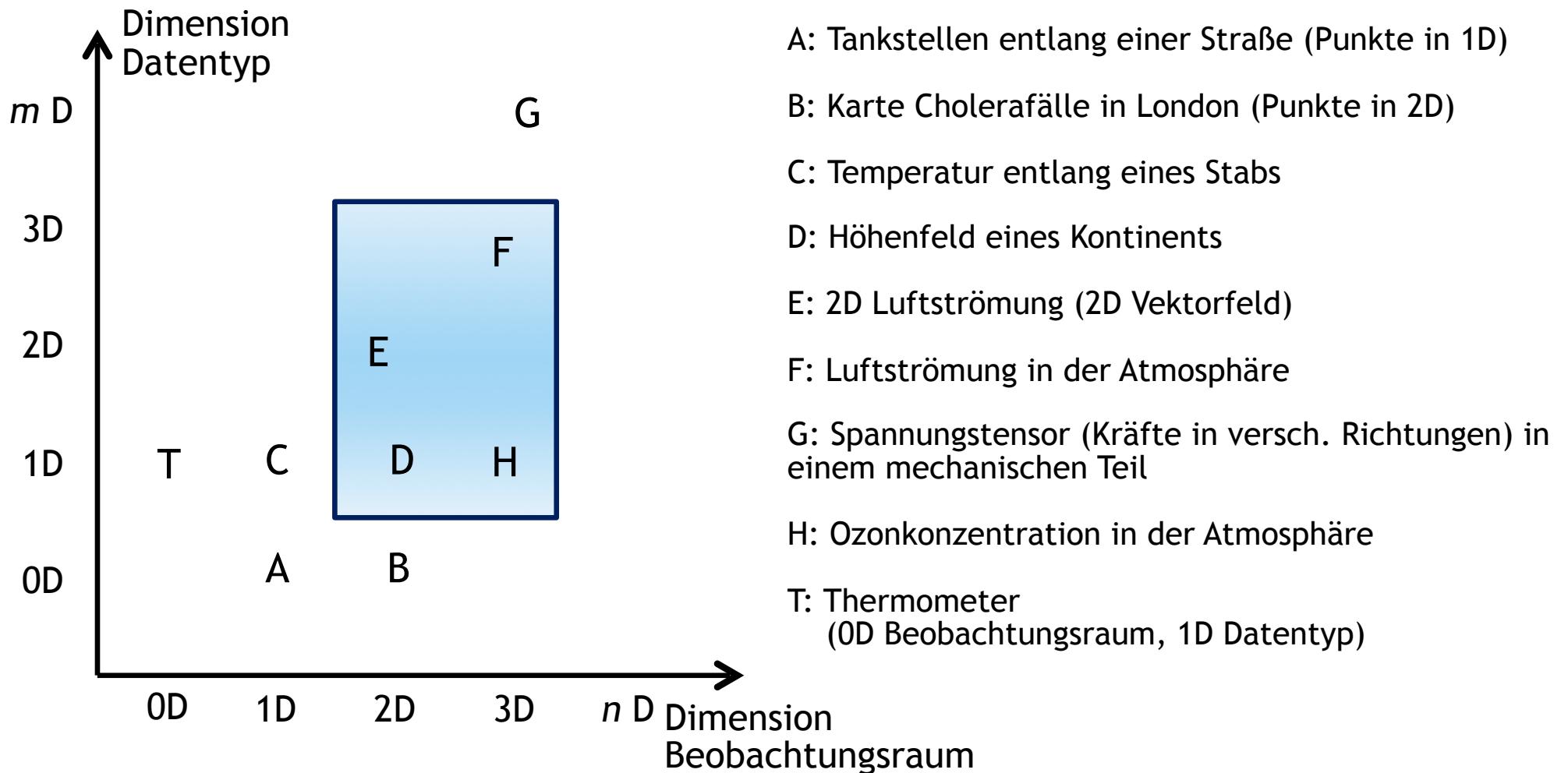


Verhältnisskala

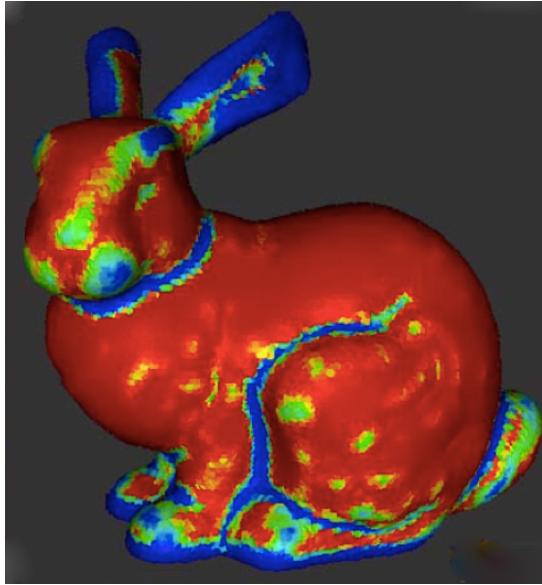
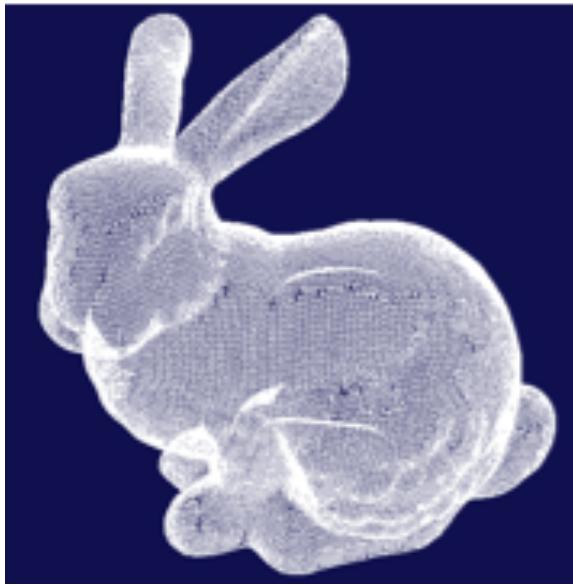


Datenrepräsentation

- wir unterscheiden Visualisierungstechniken nach
 - Dimension des Beobachtungsraums (unabh. Variablen) und
 - Typ und Dimension der zu visualisierenden Daten (abh. Variablen)



Scientific Visualization - Basic Data Characteristics



Stanford bunny 3D model
(70K triangles, 30K vertices)

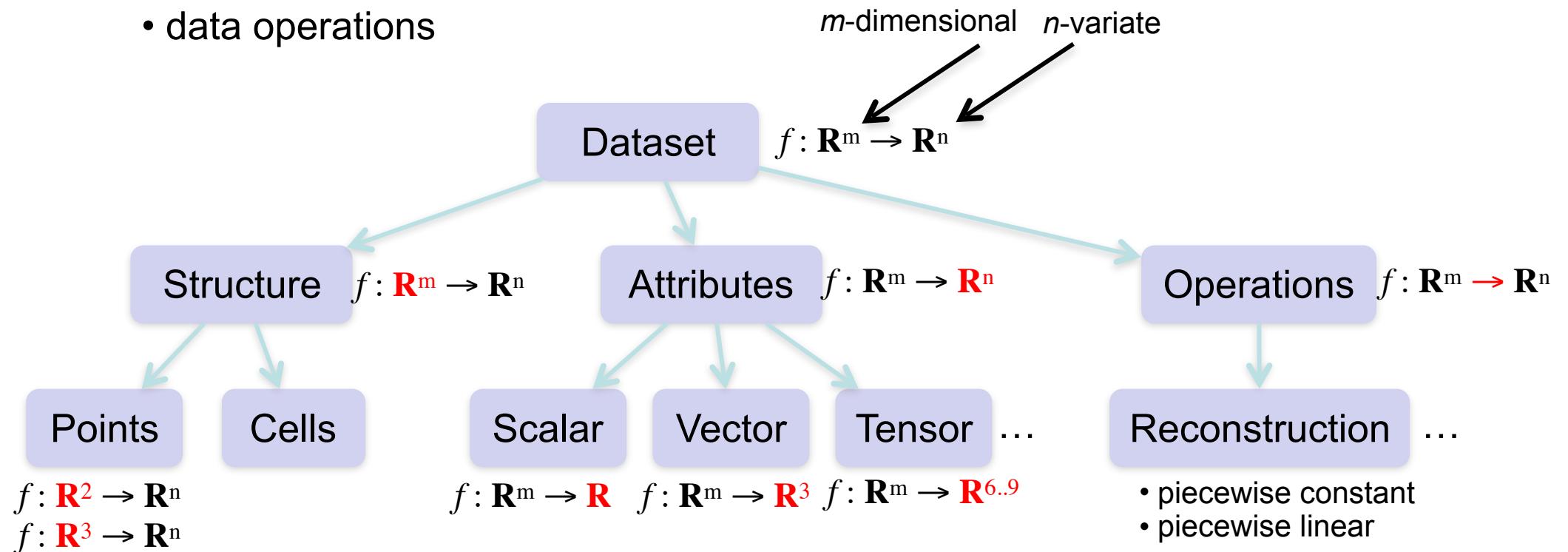
Right image shows surface curvature
(scalar dataset, red=flat, blue=curved)

- data: numerical values defined on a spatial domain, say: $f: \mathbf{R}^3 \rightarrow \mathbf{R}$
- both domain and range are continuous spaces
- hence the following are easy
 - resampling / rescaling
 - filtering
 - reconstruction (from piecewise discrete representation e.g. triangles)
 - **visual interpretation** (domain is a natural 3D shape)

Definition Datensatz

Dataset

- key notion in visualization (SciVis, InfoVis, SoftVis)
- a dataset captures all relevant characteristics of a data collection
 - structure
 - data values
 - data operations

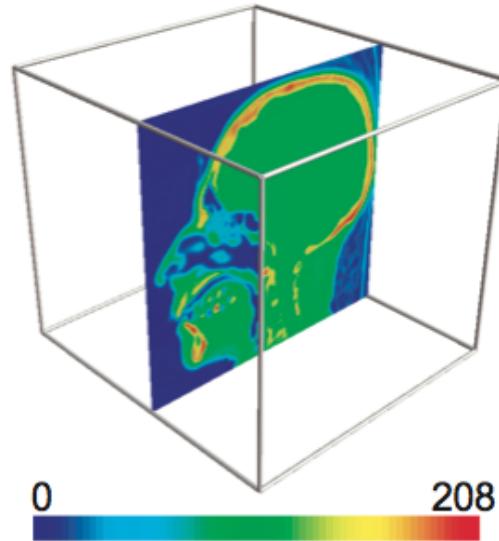


from: Data Visualization, Principles and Practice (2nd edition), CRC Press, 2014, A. C. Telea

Beispiele für Datensätze

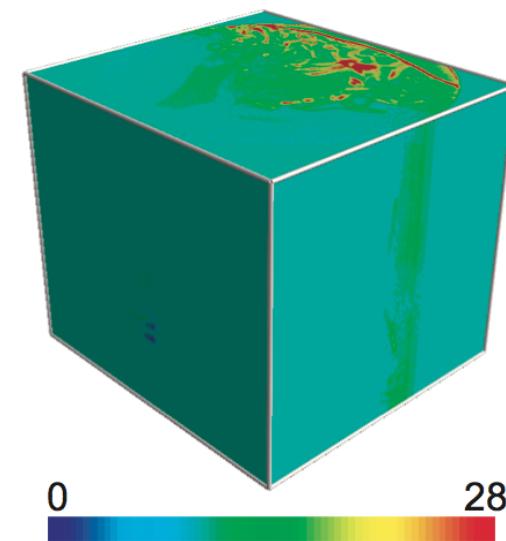
$$f: \mathbf{R}^2 \rightarrow \mathbf{R}$$

a planar slice



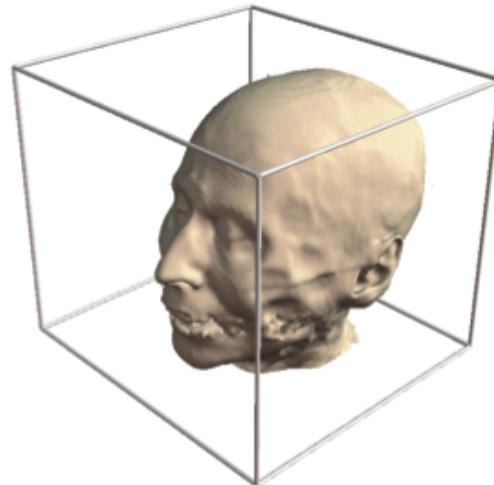
$$f: \mathbf{R}^3 \rightarrow \mathbf{R}$$

a volume



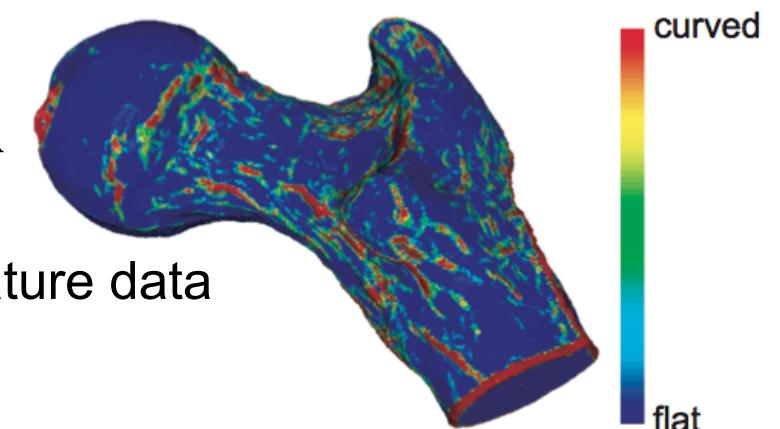
$$f: \mathbf{R}^2 \rightarrow \mathbf{R}^0$$

a surface

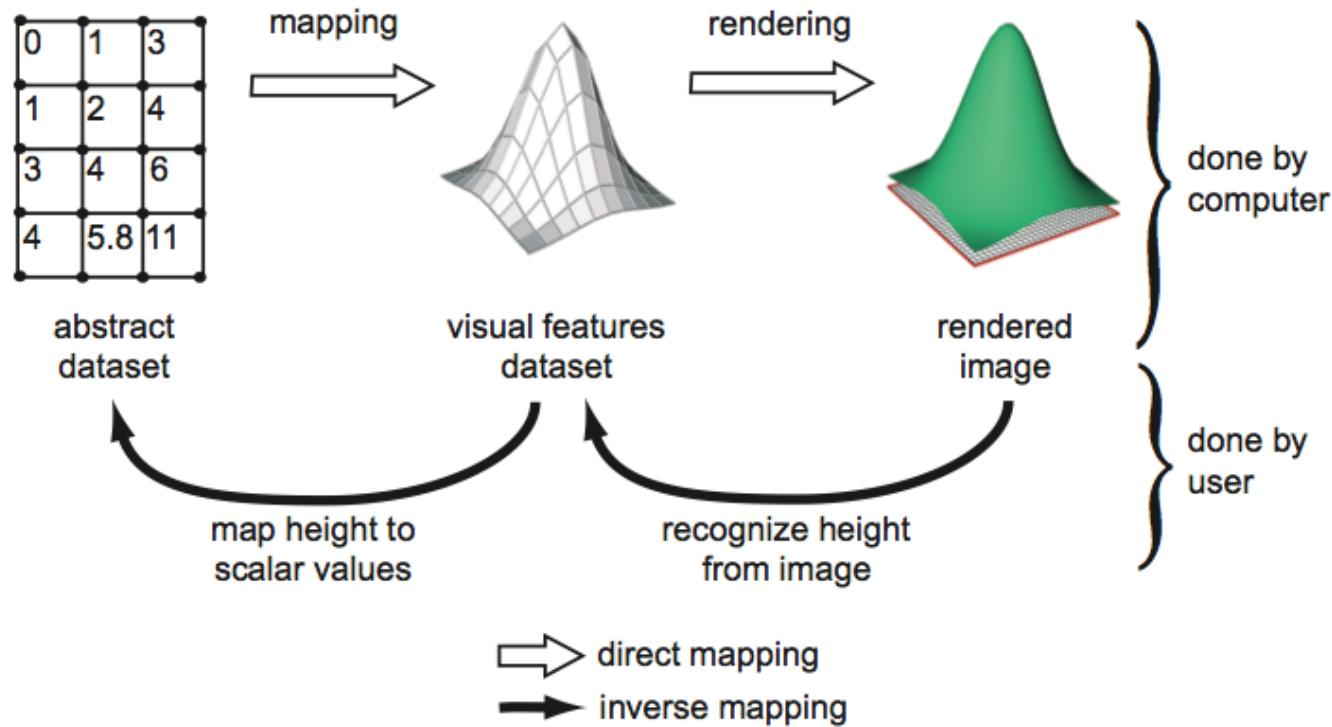


$$f: \mathbf{R}^2 \rightarrow \mathbf{R}$$

a surface
with curvature data



from: Data Visualization, Principles and Practice (2nd edition), CRC Press, 2014, A. C. Telea



- input: dataset in high-dimensional space $d \subseteq \mathbf{D}^{m \times n}$
- output: color image $i \subseteq \mathbf{R}^{2 \times 3} = \mathbf{R}^5$
- visualization: function $v : \mathbf{D}^m \rightarrow \mathbf{R}^5$ (from data to images)
- analysis: inverse function $v^{-1} : \mathbf{D}^m \rightarrow \mathbf{R}^5$ (from images to data)

from: Data Visualization, Principles and Practice (2nd edition), CRC Press, 2014, A. C. Telea

Visualization Challenges

Dimensionality

- input dataset typically of much higher dimensionality than 2D images ($m+n>>2+3$)
- where to put all those dimensions?

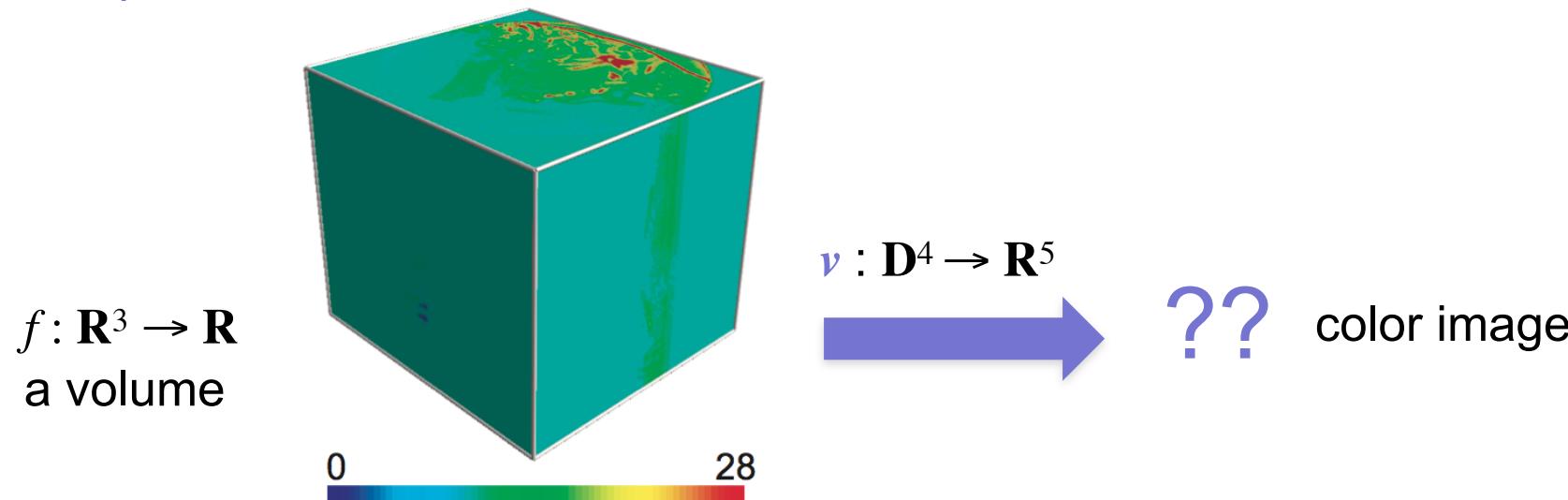
Data size

- input number of data points much higher than screen resolution
- where to draw all those data points?

Analysis

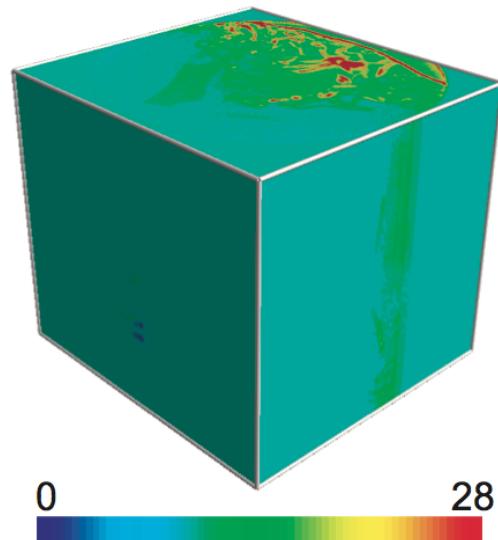
- visualization function not (fully) invertible
- how to go from shapes/colors back to data?

Example



from: Data Visualization, Principles and Practice (2nd edition), CRC Press, 2014, A. C. Telea

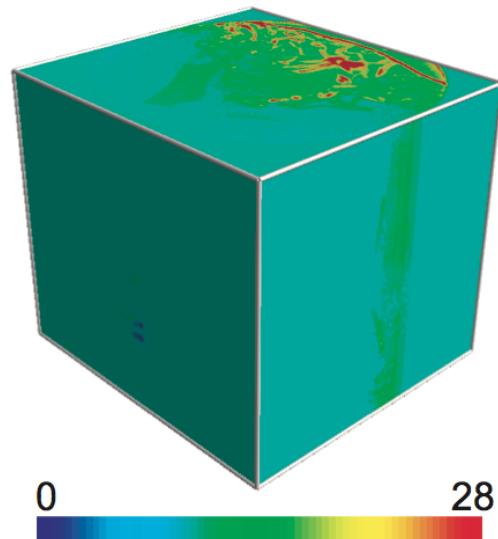
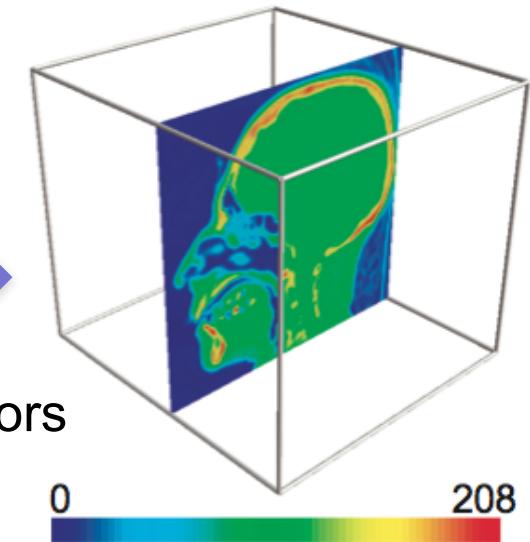
Simple Solutions



filtering
extract slice

$f: \mathbf{R}^2 \rightarrow \mathbf{R}$

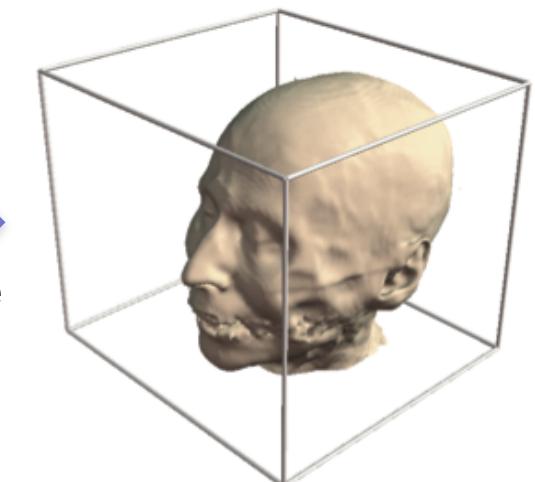
mapping
draw slice
map f to colors



filtering
extract surface

$f: \mathbf{R}^2 \rightarrow \mathbf{R}^0$

mapping
draw surface



from: Data Visualization, Principles and Practice (2nd edition), CRC Press, 2014, A. C. Telea

Diskrete Repräsentationen

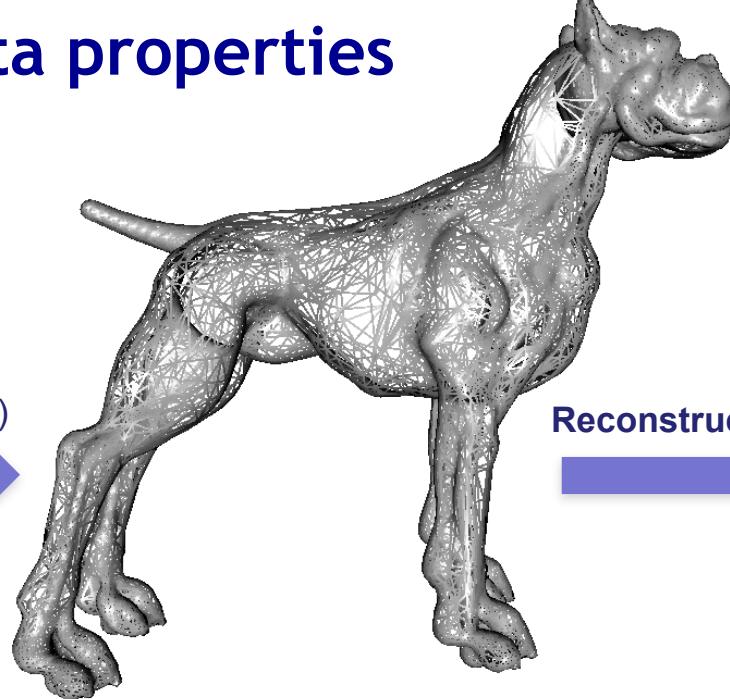
- eigentlich sind die meisten Objekte, die wir visualisieren „kontinuierlich“
- meist sind die Daten aber nur an diskreten Stellen/Zeitpunkten gegeben
- diskrete Strukturen bestehen aus Abtastwerten/Samples
 - daraus werden Gitter oder Netze, bestehend aus Zellen, generiert
- Primitive durch Verbindung einer oder mehrerer Abtastwerte

Dimension	Zelle	Netz
0D	unstrukturierte Punkte	
1D	Linien (Kanten)	Polyline(-gon)
2D	Dreiecke, Vierecke	2D Netz
3D	Tetraeder, Prismen, Hexaeder	3D Netz

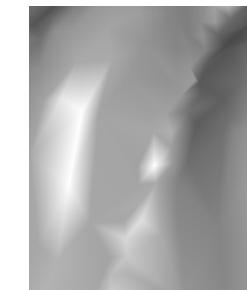
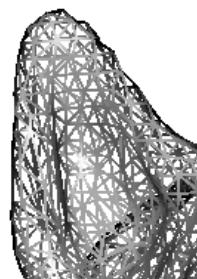
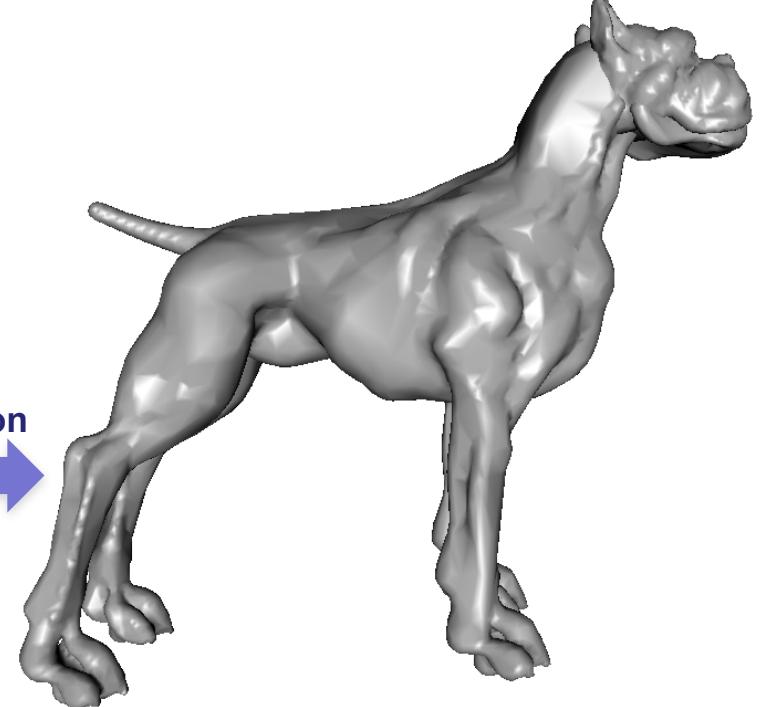
Visualization data properties



Sampling
(data importing)



Reconstruction



$f : D \rightarrow C$
Continuous data

Sampling

$\{p_i, f_i\}$ $p_i \in D$ $f_i \in C$
Measurements (samples)
at discrete set of points

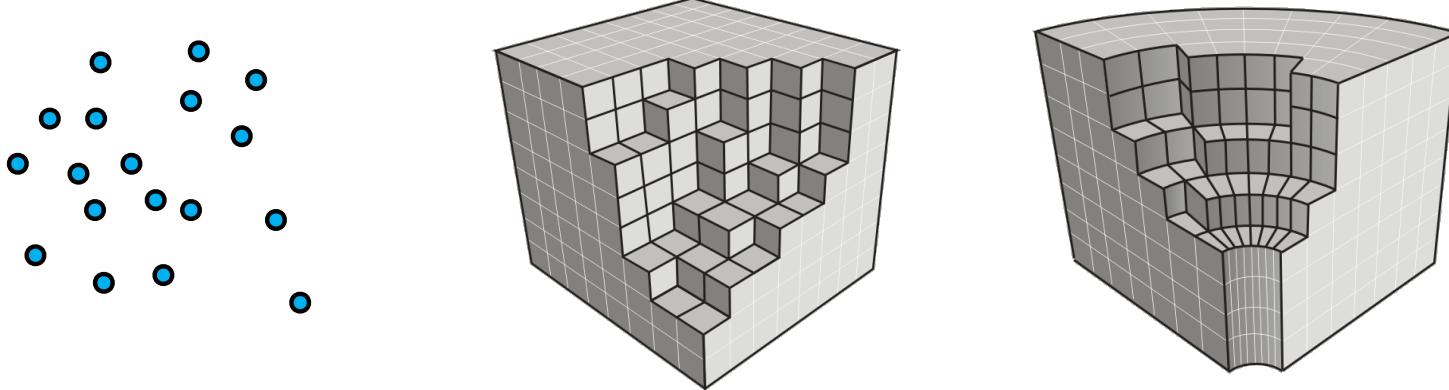
Reconstruction

$\tilde{f} : D \rightarrow C$ $\tilde{f}(p_i) = f(p_i) = f_i$
Continuous data, as close as possible to input

from: Data Visualization, Principles and Practice (2nd edition), CRC Press, 2014, A. C. Telea

Beobachtungsraum (Domain)

- betrachten wir den Beobachtungsraum genauer...
- die (geometrische) Form des Beobachtungsraums ist durch die Position der Abtast-/Messpunkte bestimmt



- der Beobachtungsraum ist außerdem charakterisiert durch
 - Dimensionalität: 0D, 1D, 2D, 3D, 4D, ...
 - Einfluss/Wirkungskreis: was sagt ein Messpunkt über seine Nachbarschaft aus?
 - **Struktur:** sind die Datenpunkte verbunden? Wie? (Topologie!)
 - im Beispiel mittig und rechts sind sie zu Hexaedern verbunden

Einfluss/Wirkungskreis

- die Werte an einem Abtastpunkt sagen *vermutlich* etwas über die Daten in seiner Umgebung aus
- um Datenwerte an beliebigen Punkten innerhalb des Beobachtungsraums zu rekonstruieren muss die Verteilung (potentiell) aller Abtastpunkte berücksichtigt werden

Einfluss/Wirkungskreis

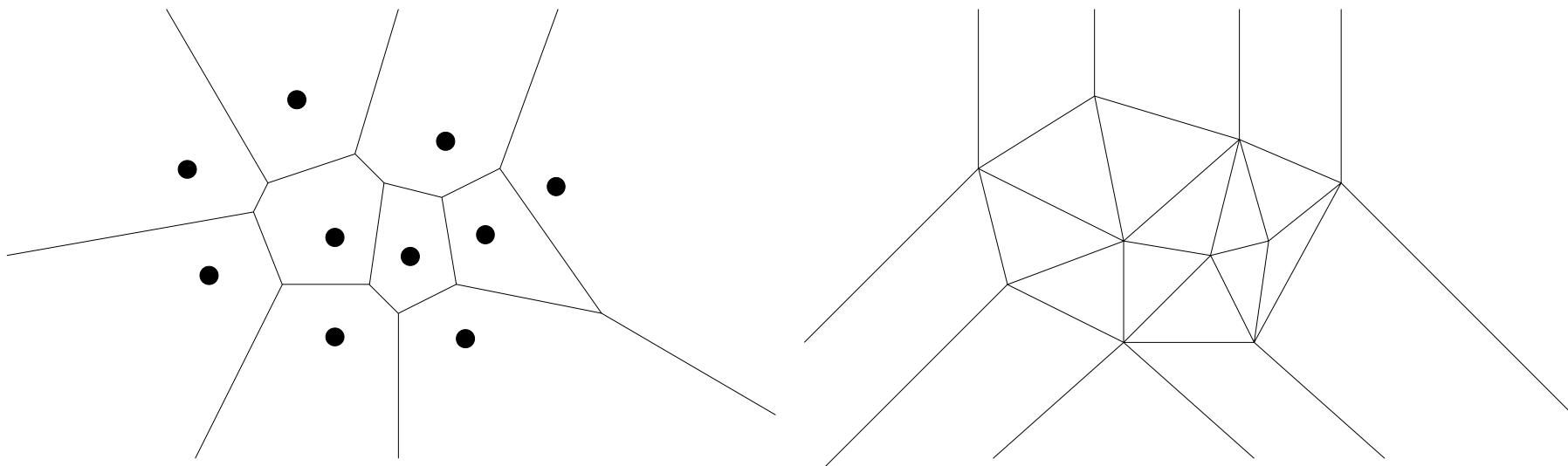
- die Werte an einem Abtastpunkt sagen *vermutlich* etwas über die Daten in seiner Umgebung aus
- um Datenwerte an beliebigen Punkten innerhalb des Beobachtungsraums zu rekonstruieren muss die Verteilung (potentiell) aller Abtastpunkte berücksichtigt werden
- man unterscheidet folgende Wirkungskreise
 - **punktuell**: die Daten gelten nur für den dazugehörigen Punkt
 - **lokal**: Daten gelten für einen gewissen Bereich um den Abtastpunkt
(aber für welchen?)
 - **global**: Abtastpunkte haben potentiell Einfluss auf gesamten Beobachtungsraum (siehe „Scattered Data“ Interpolation in Kap. 2)

Voronoi Diagramm (oder Voronoi Zerlegung, links)

- Idee: erzeuge eine Region um jeden Abtastpunkt, die alle Punkte enthält, die näher an diesem sind, als an anderen Abtastpunkten
- jeder Punkt innerhalb dieser Region erhält dann den Datenwert des Abtastpunktes

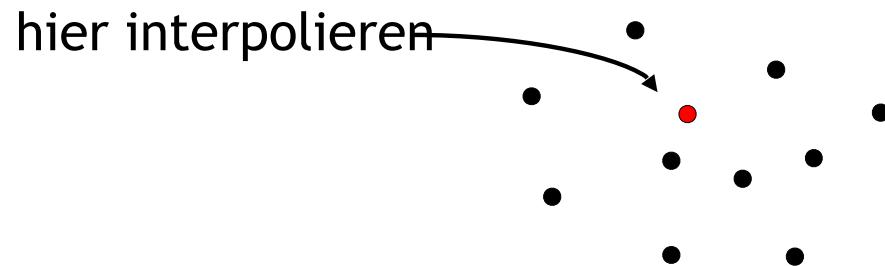
Delaunay Triangulierung (rechts)

- Triangulierung bei der Abtastpunkte zu Eckpunkten werden
- Interpolation innerhalb der Dreiecke

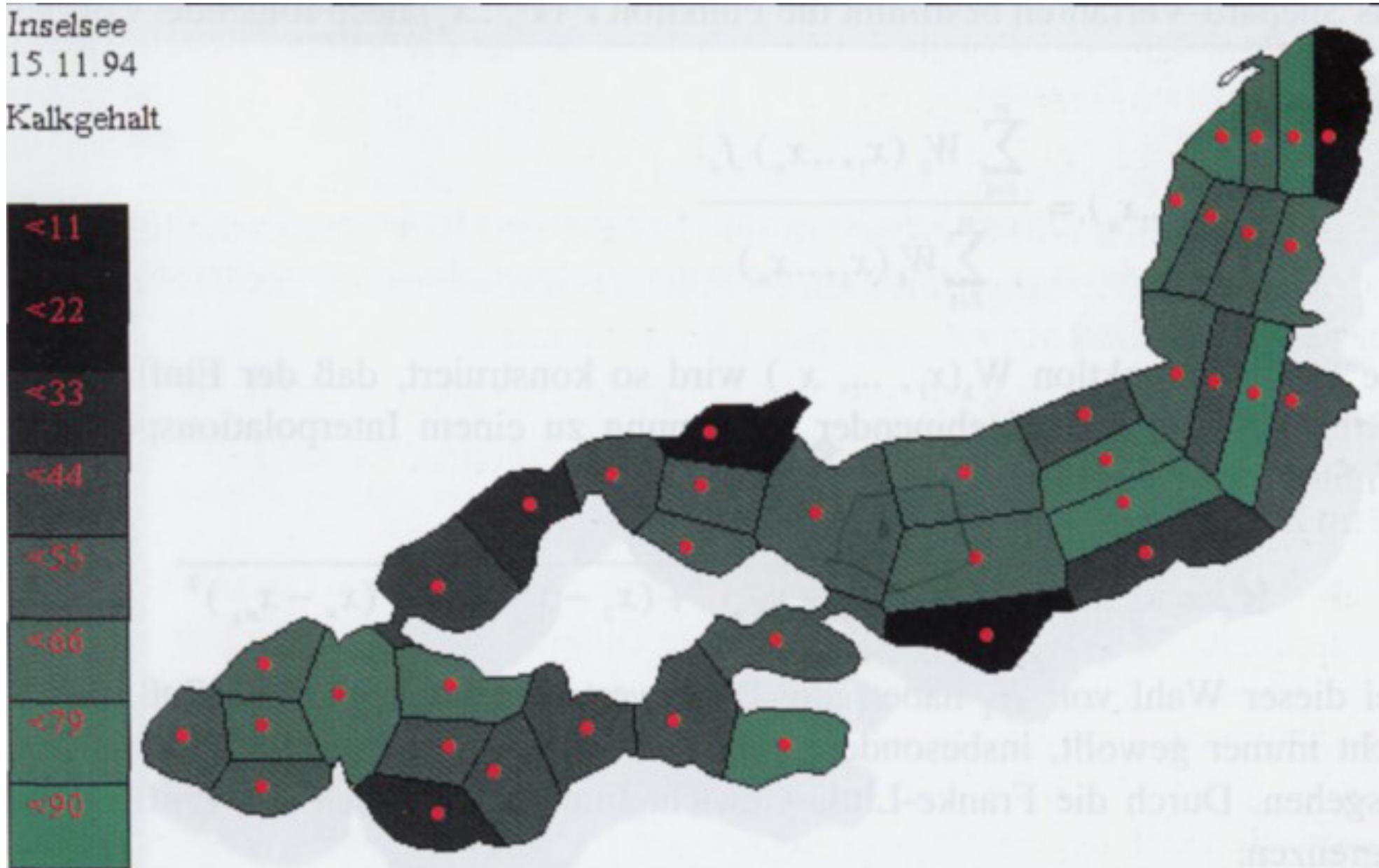


Scattered Data Interpolation

- an jedem Punkt berechnet man eine gewichtete Summe aller Werte an den Abtastpunkten
- Gewichtungsfunktionen bestimmen den Träger (Support) eines Abtastpunktes
 - radiale Basisfunktionen bilden einen Einfluss nach, der mit dem Abstand zu den Abtastpunkten abnimmt
- Achtung: die Berechnung kann aufwändig werden und ist u.U. nicht interpolierend

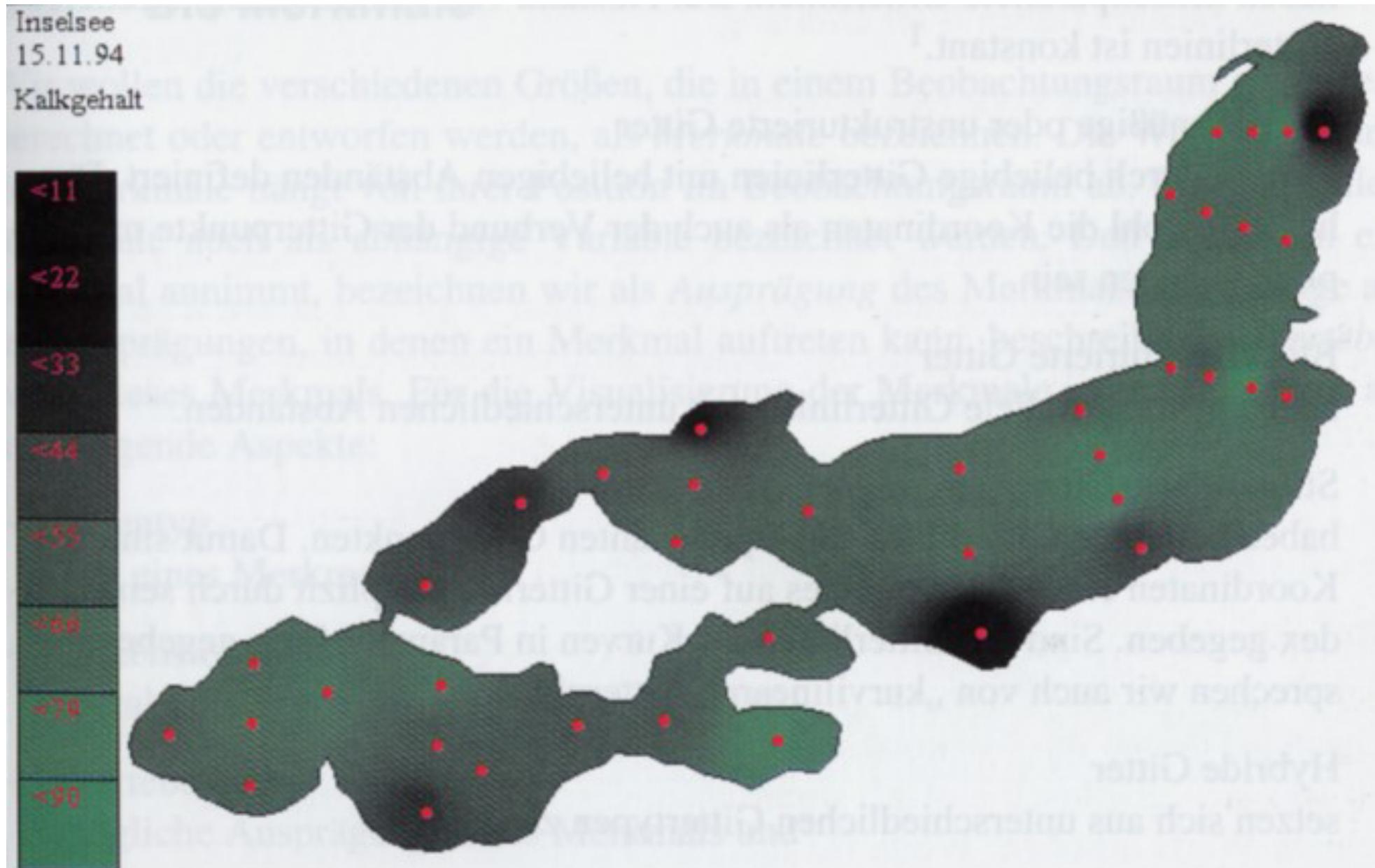


Voronoi-Zellen



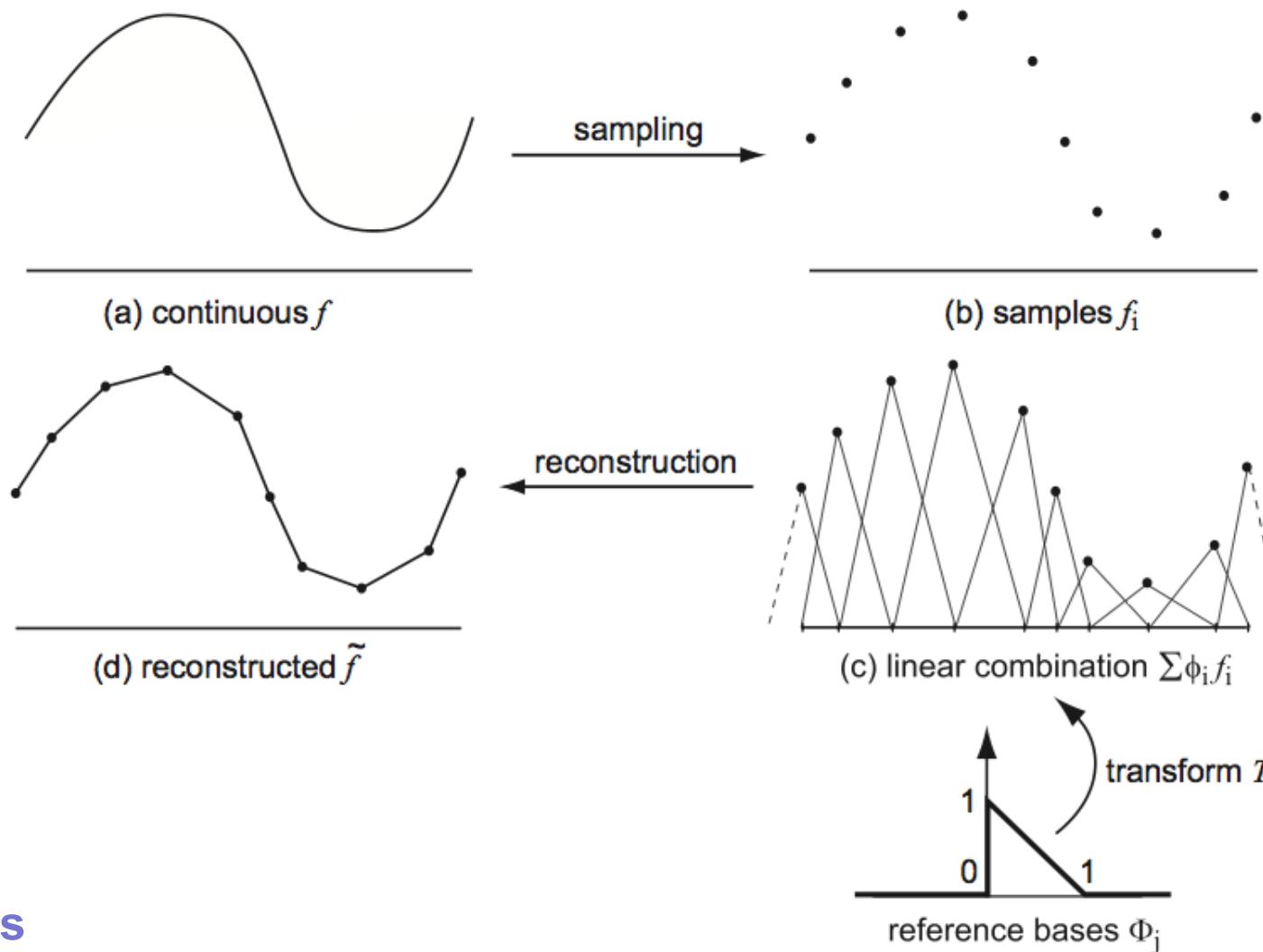
aus Schumann, Müller - Visualisierung

Shepard-Interpolation



aus Schumann, Müller - Visualisierung

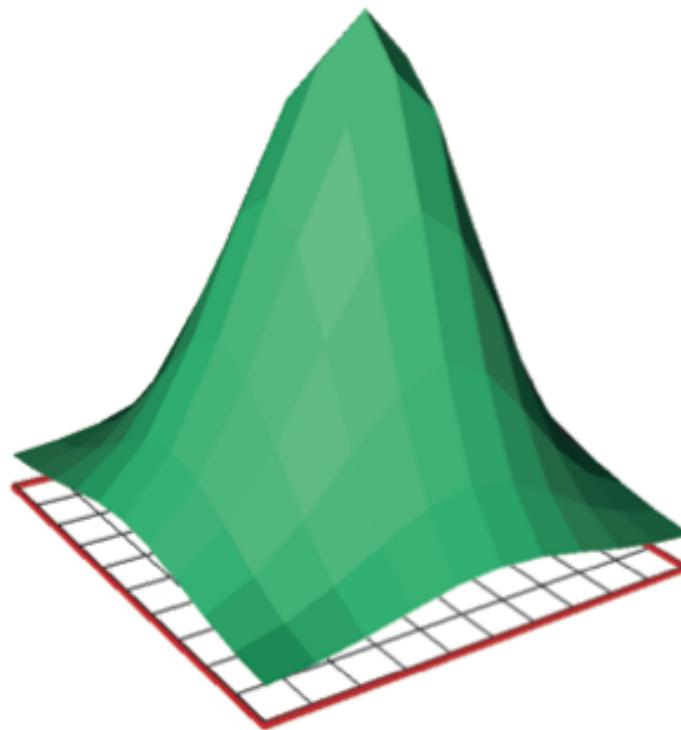
Interpolation



Remarks

- interpolation & reconstruction goes cell-by-cell
- only need sample points at a cell vertices to interpolate over that cell
- reconstruction is C^1 because ϕ_i are C^1 and interpolation formula is C^∞

Bilinear interpolation



$$\Phi_1^1(r, s) = (1 - r)(1 - s),$$

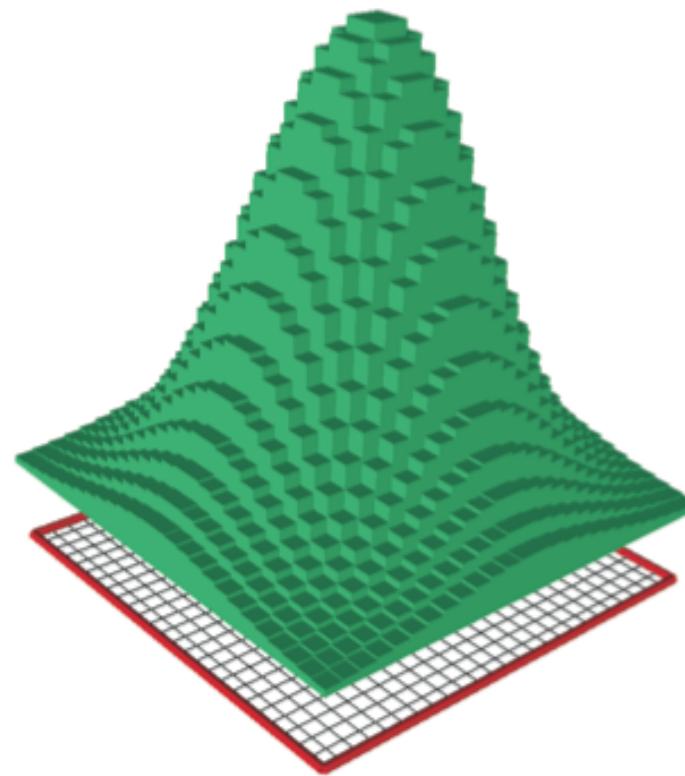
$$\Phi_2^1(r, s) = r(1 - s),$$

$$\Phi_3^1(r, s) = rs,$$

$$\Phi_4^1(r, s) = (1 - r)s;$$

- 4 functions, one per vertex
- result: C^0 but never C^1 (why?)
- good for vertex-based samples

Constant interpolation

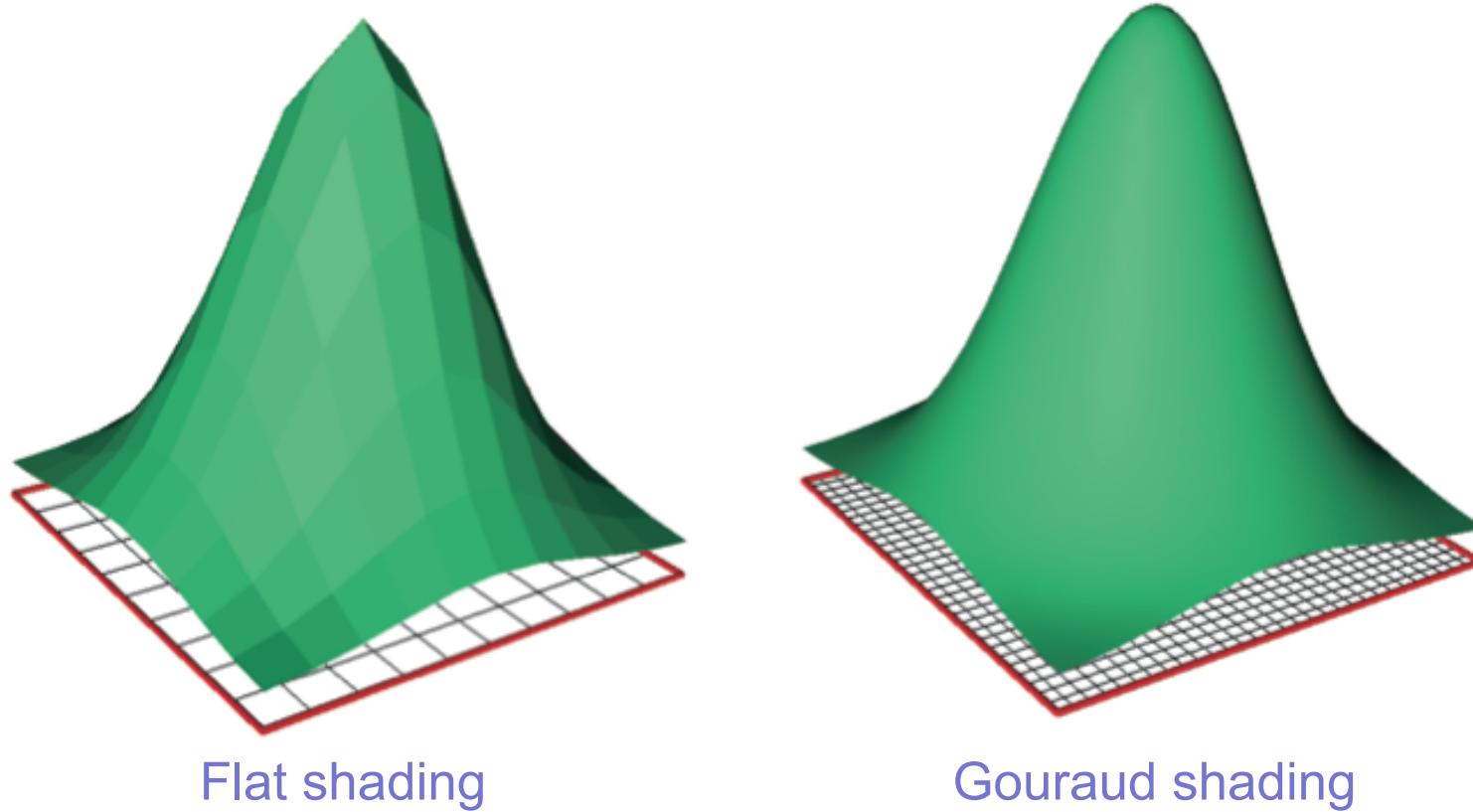


$$\phi_i^0(x) = \begin{cases} 1, & x \in c_i, \\ 0, & x \notin c_i. \end{cases}$$

- 1 functions per whole cell
- result: not even C^0
- good for cell-based samples

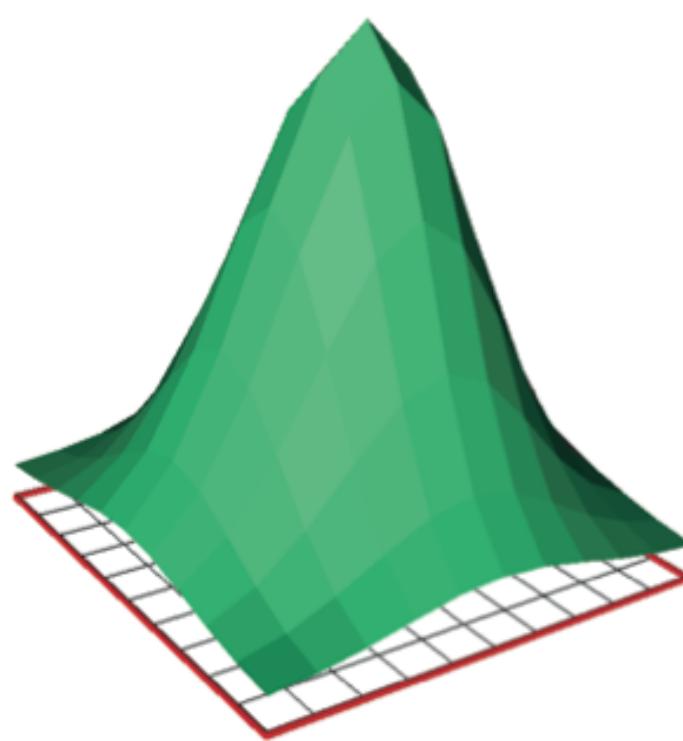
Computer Graphics Intermezzo

What is the difference between flat and Gouraud (smooth) shading?

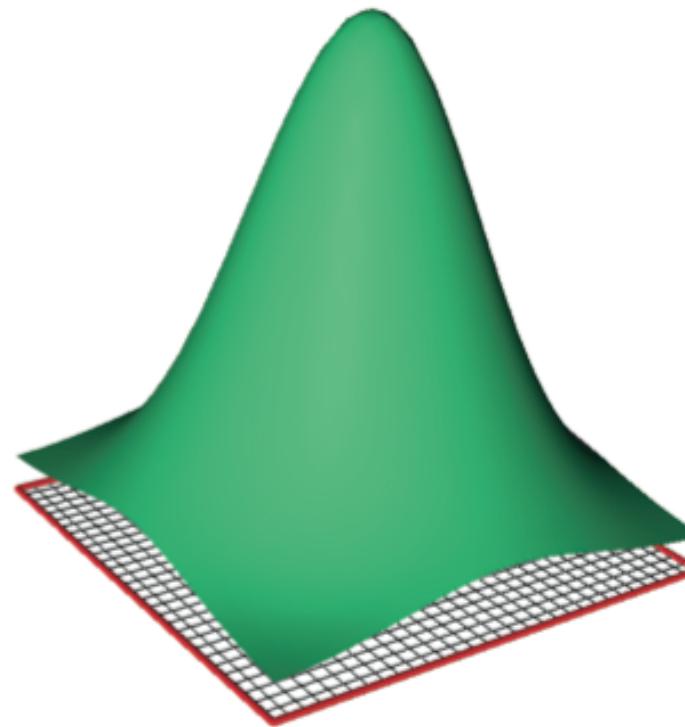


Computer Graphics Intermezzo

What is the difference between flat and Gouraud (smooth) shading?



Flat shading

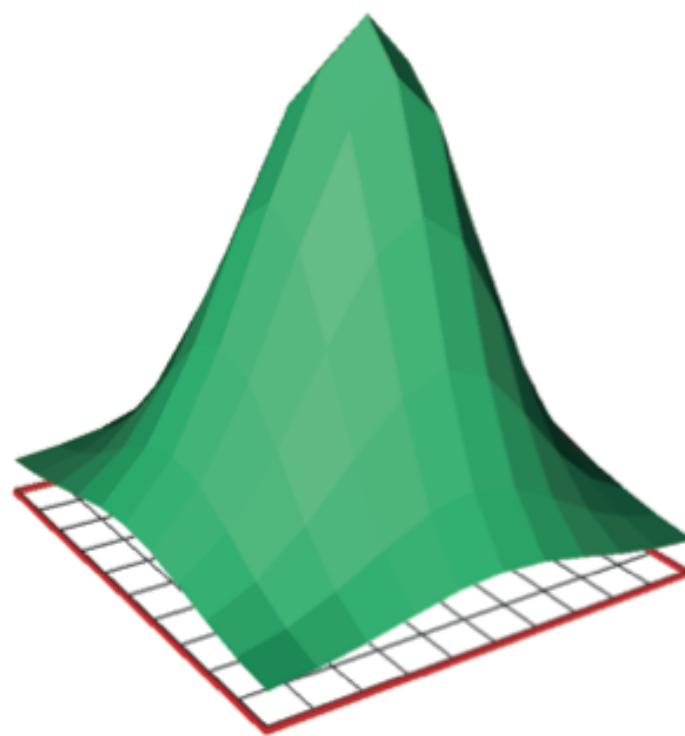


Gouraud shading

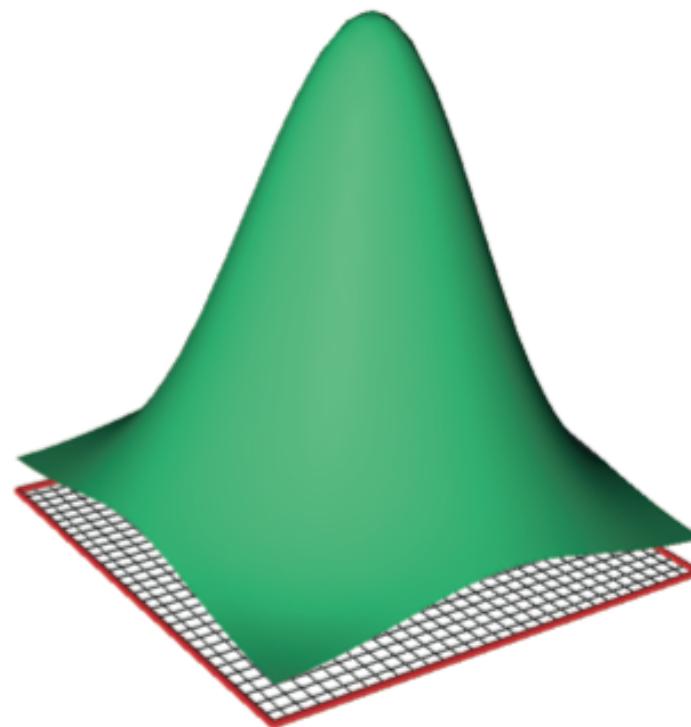
- surface: bilinear interpolation
- colors: constant interpolation

Computer Graphics Intermezzo

What is the difference between flat and Gouraud (smooth) shading?



Flat shading



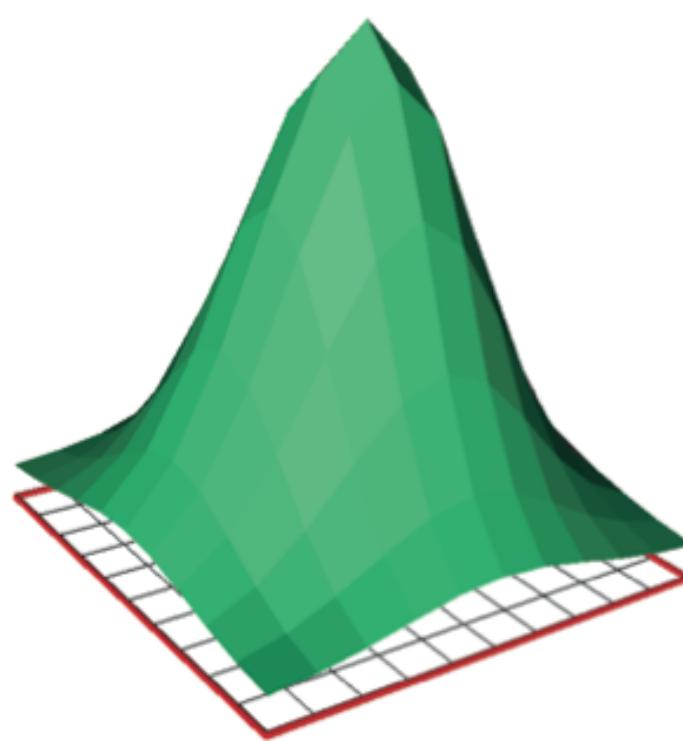
Gouraud shading

- surface: bilinear interpolation
- colors: constant interpolation

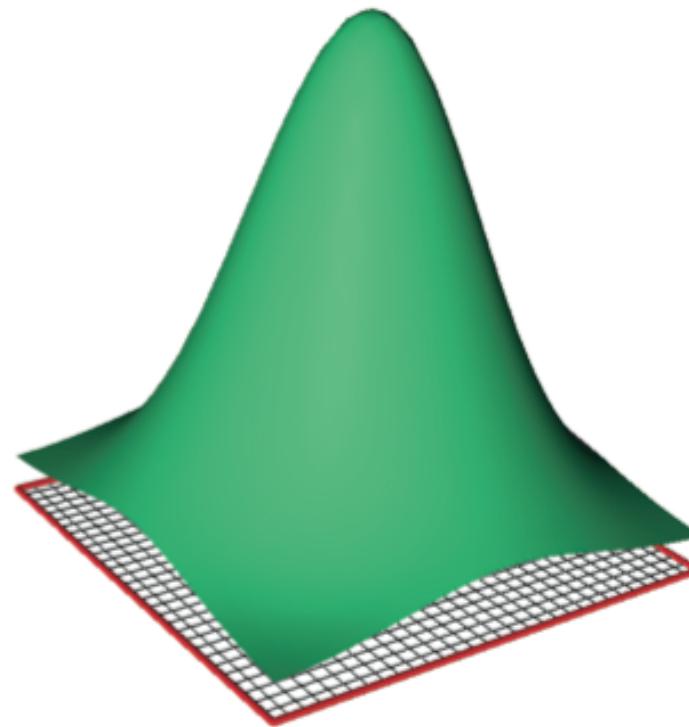
- surface: bilinear interpolation
- colors: bilinear interpolation

Computer Graphics Intermezzo

What is the difference between flat and Gouraud (smooth) shading?



Flat shading



Gouraud shading

- surface: bilinear interpolation
 - colors: constant interpolation
- surface: bilinear interpolation
 - colors: bilinear interpolation

Note: do not confuse *Gouraud shading* (color interpolation) with the *Phong lighting model* (color computation from normals)

- wie bei anderen Anwendungen (z.B. Simulationen) unterscheiden sich Datenstrukturen hinsichtlich
 - Effizienz beim Zugriff auf die Daten (z.B. welcher Datenpunkt ist am nahsten bei einer Position?)
 - Speichereffizienz (z.B. Dreiecksnetze mit/ohne Shared Vertex Struktur)
 - verlustfreie oder verlustbehaftete Speicherung
 - ...
- Definition
 - wenn Daten beliebig verteilt und ohne Konnektivität vorliegen, dann spricht man von „gitterfreien Daten“ oder „scattered data“
 - ansonsten ist der Beobachtungsraum in Zellen unterteilt
 - dabei unterscheiden wir (wie bei Dreiecksnetzen) zwischen
 - **Topologie** beschreibt die **Struktur** (Konnektivität) der Daten
 - **Geometrie** beschreibt die **Position** der Daten(werte)

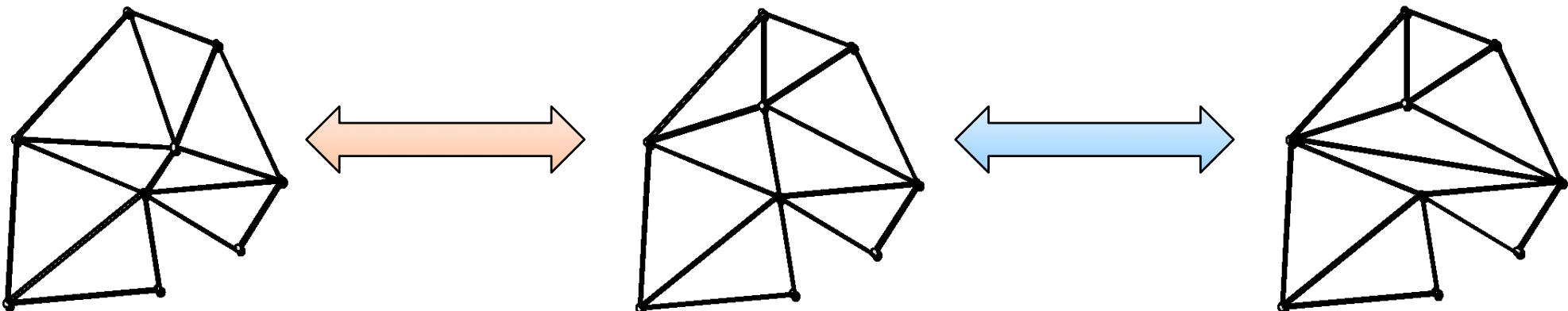
Topologie

- Eigenschaften, die sich **nicht** bei (leichten) Deformationen ändern
- topologisch äquivalente Strukturen lassen sich durch Streckung bzw. Stauchung (und ähnlichen Deformationen) ineinander transformieren, ohne die Konnektivität zu ändern

unterschiedliche Geometrie

(Vertexpositionen)

äquivalente Topologie (Konnektivität)



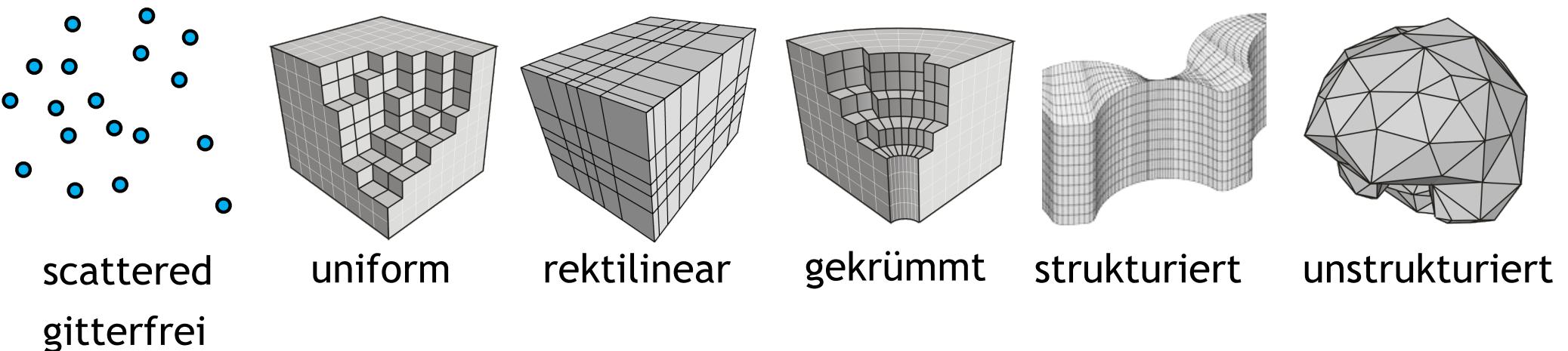
gleiche Geometrie (Vertexpositionen)

unterschiedliche Topologie

(Konnektivität)

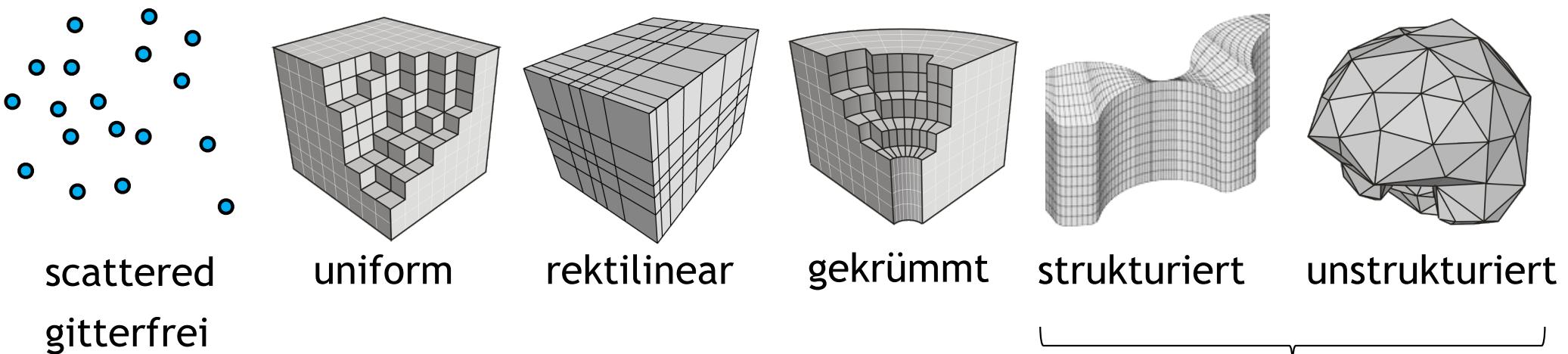
Gittertypen

- Gitter unterscheiden sich in
 - Art und Form der Zellen aus denen sie aufgebaut sind
 - in der Art und Weise, wie topologische Information gegeben ist
 - z.B. ist Topologie implizit, oder muss sie explizit gespeichert werden?



Gittertypen

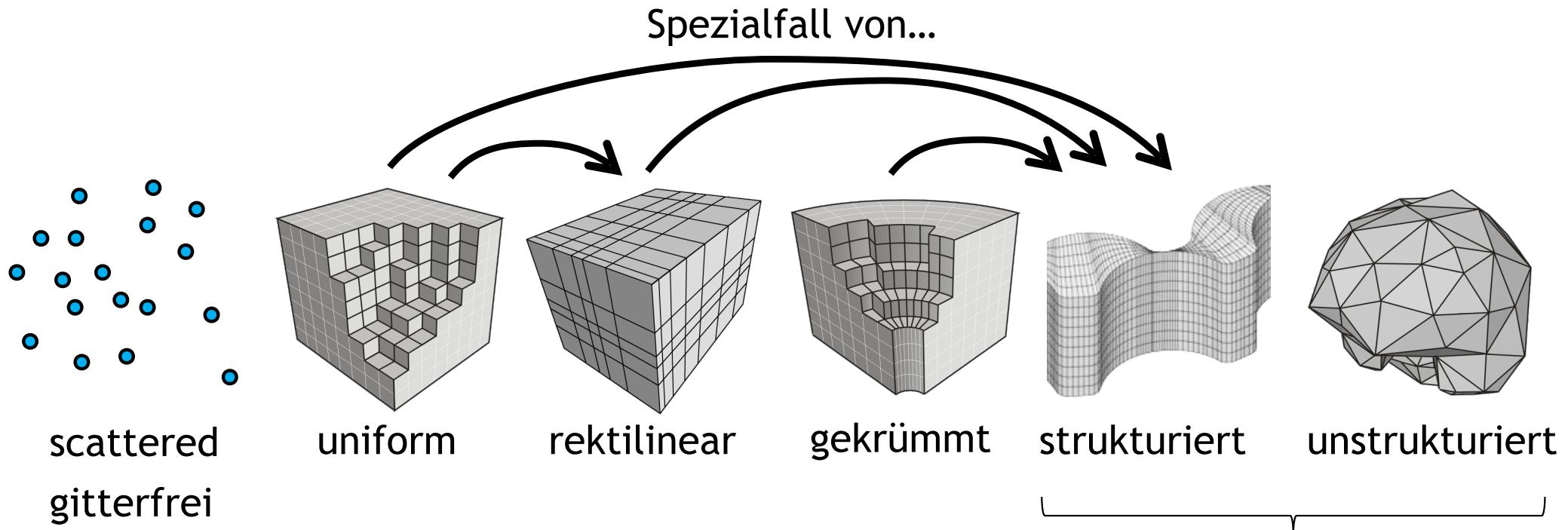
- Gitter unterscheiden sich in
 - Art und Form der Zellen aus denen sie aufgebaut sind
 - in der Art und Weise, wie topologische Information gegeben ist
 - z.B. ist Topologie implizit, oder muss sie explizit gespeichert werden?



Anm. manchmal als irreguläre Gitter bezeichnet (aber gewöhnen Sie sich das nicht an)

Gittertypen

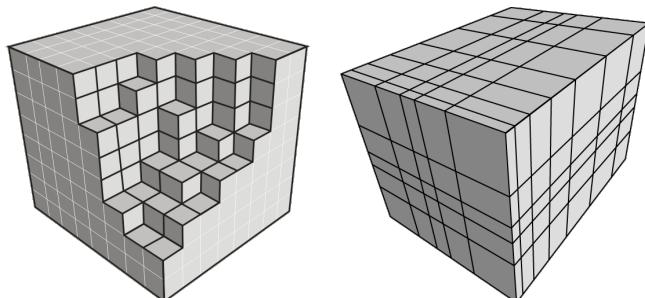
- Gitter unterscheiden sich in
 - Art und Form der Zellen aus denen sie aufgebaut sind
 - in der Art und Weise, wie topologische Information gegeben ist
 - z.B. ist Topologie implizit, oder muss sie explizit gespeichert werden?



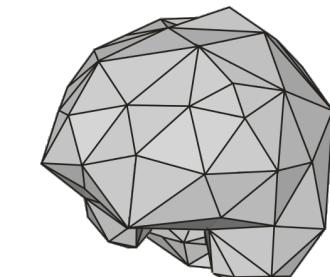
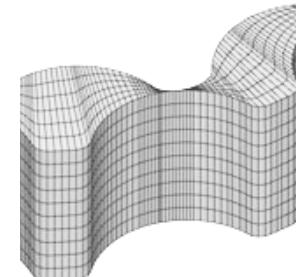
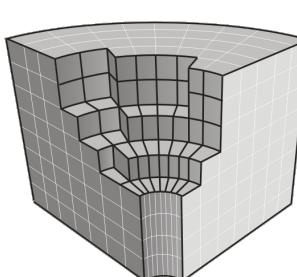
Anm. manchmal als irreguläre Gitter bezeichnet (aber gewöhnen Sie sich das nicht an)

Strukturierte und unstrukturierte Gitter

- ... unterscheiden sich in der Weise, wie Zellen aneinander liegen
- strukturierte Gitter
 - besitzen eine reguläre Topologie und reguläre/irreguläre Geometrie
 - wenn, dann muss nur Geometrieeinformation (Vertizes, Spacing) gespeichert werden
- unstrukturierte Gitter (= „keine feste Nachbarschaft“)
 - besitzen irreguläre Topologie und Geometrie
 - Geometrie (Vertizes) und Topologie muss gespeichert werden



strukturiert



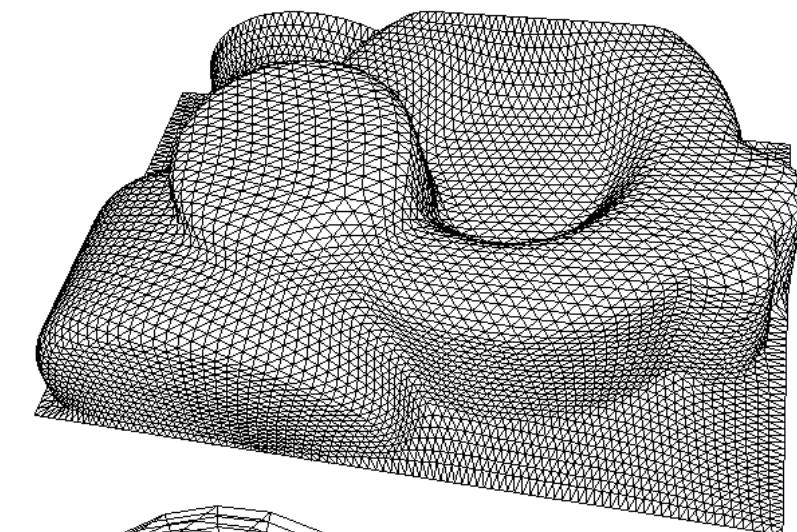
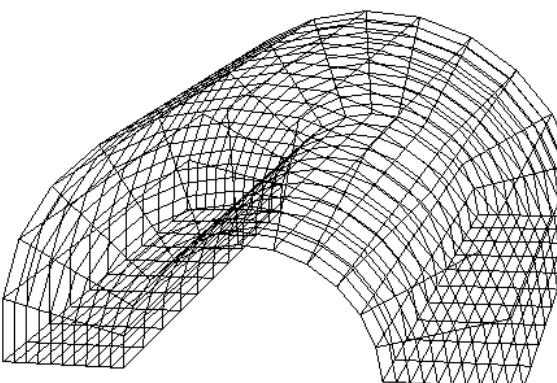
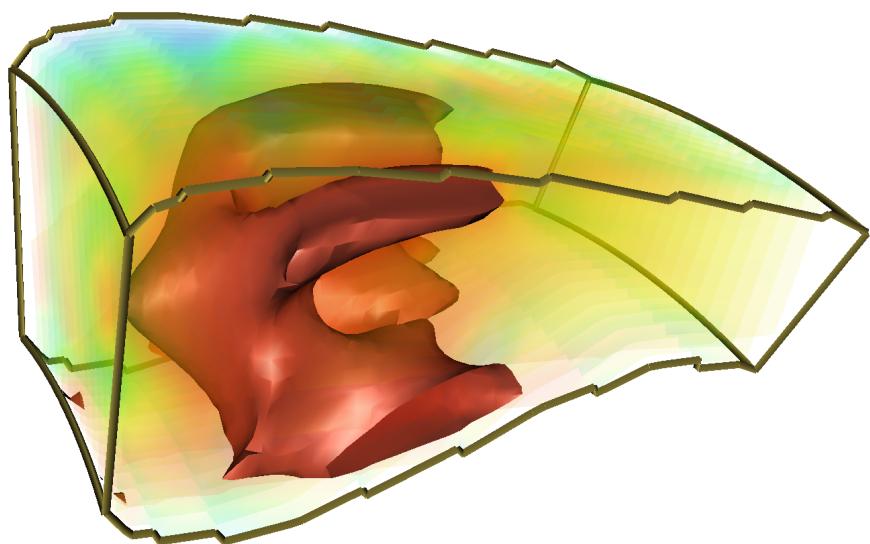
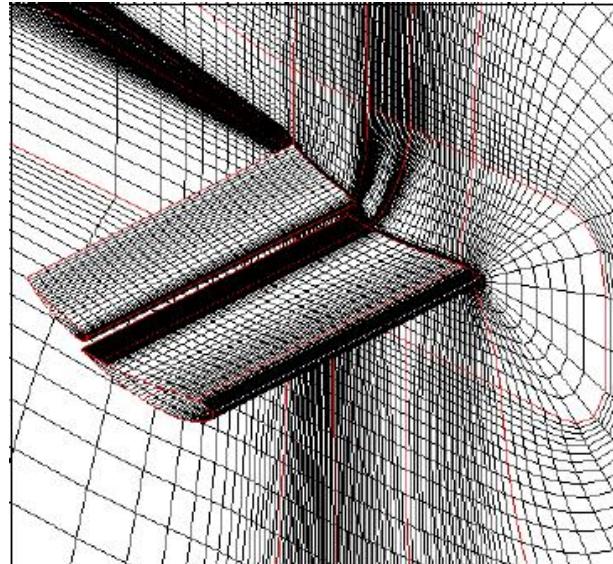
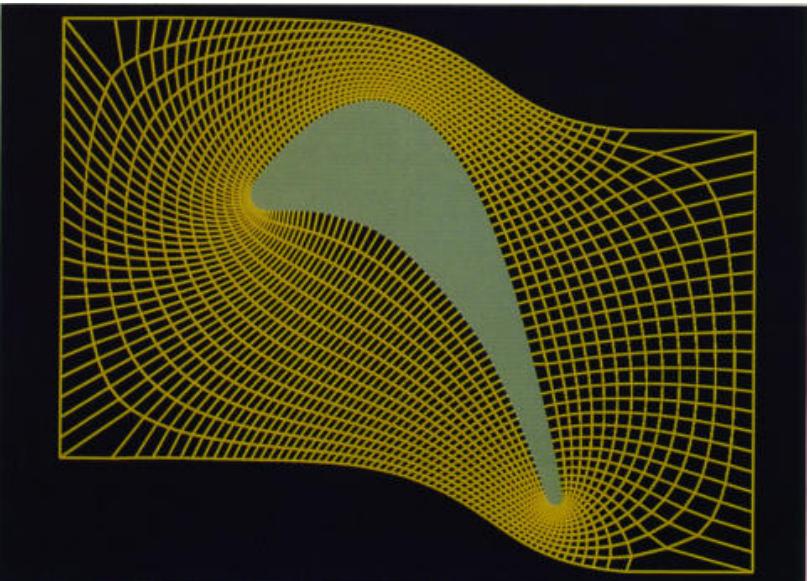
unstrukturiert

Strukturierte Gitter

- i.d.R. einfacher in der Handhabung und bei Berechnungen
- oft zusammengesetzt aus verbundenen Parallelogrammen (Hexaeder)
 - mit gleichförmigen Zellen oder
 - Zellen deformiert durch (nicht-lineare) Transformationen
- **keine lokale Adaptivität:** u.U. sind mehr oder stark deformierte Zellen notwendig, um den Raum präzise abzutasten
- Topologie ist implizit gegen durch
 - Dimension und Zellentyp, z.B. 3D-Hexaeder-Gitter oder 2D-kurvilineares-Viereckgitter
 - jeder innere Punkt hat dieselbe Anzahl Nachbarn
- Geometrie wird explizit gespeichert, z.B. durch ein Array von Punkten

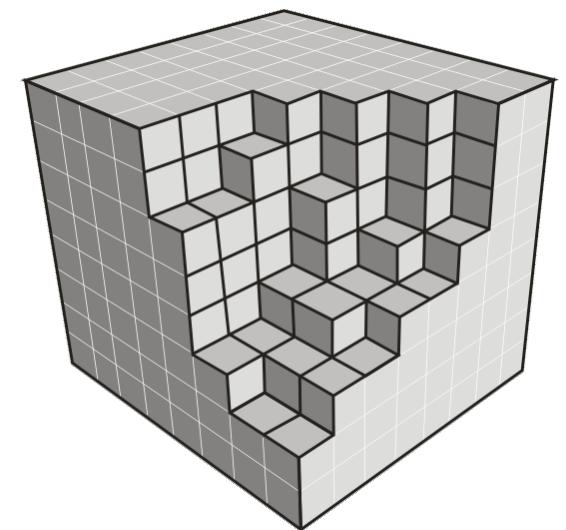
Datenstrukturen

Beispiele: Strukturierte Gitter



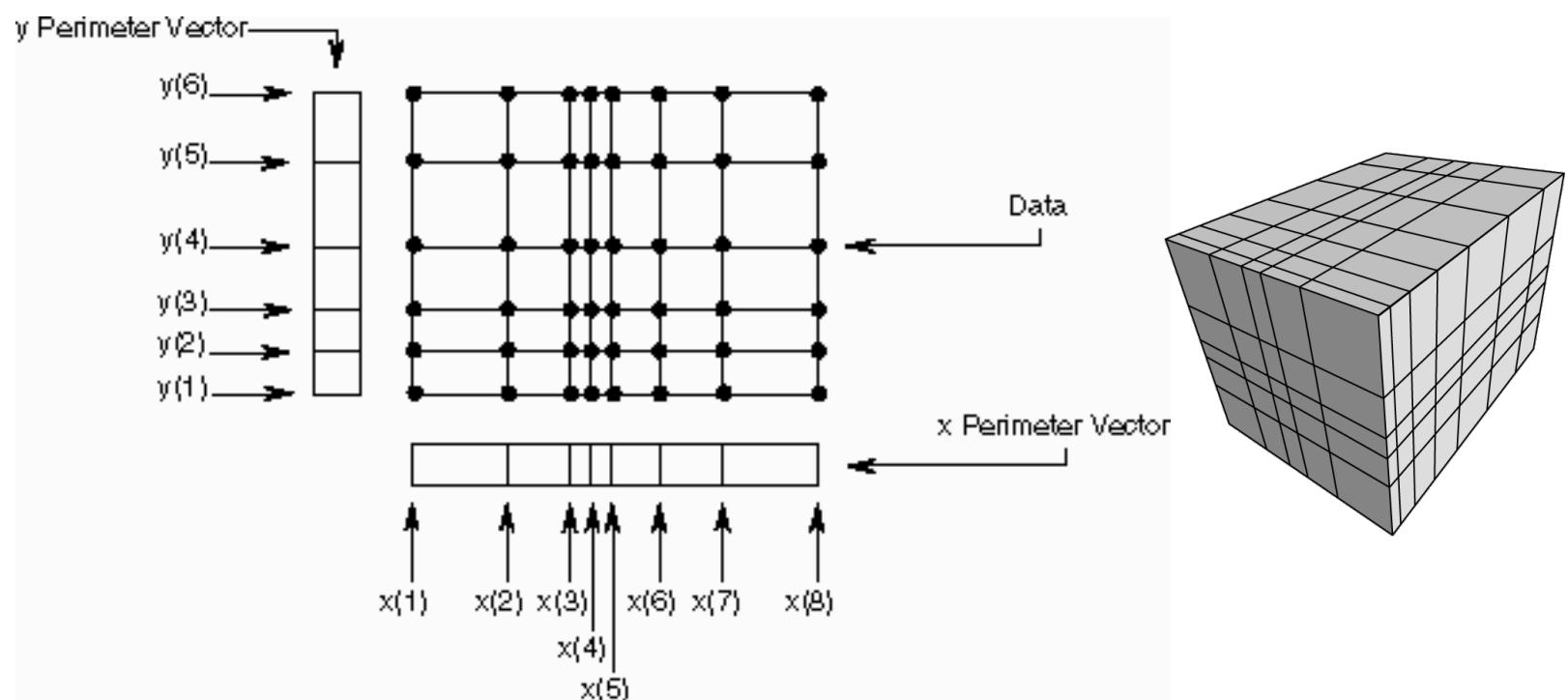
- Uniforme Gitter

- ▶ ...bestehen aus **gleichförmigen Zellen**
- ▶ unterschiedliche Ausdehnung, aber konstant, entlang der Achsen
- ▶ **einfaches Indizierungsschema** bzw. Abbildung von Position auf Zelle und umgekehrt
- ▶ meist bei Anwendungen bei denen die Daten aus einem 3D Scanner mit unterschiedlichen Auflösungen in jeder Dimension arbeiten
 - ▶ z.B. medizinische Volumendaten aus Schichtbildern
 - ▶ Schichtbild mit quadratischen Pixeln ($dx = dy$)
 - ▶ größerer Schichtabstand ($dz > dx = dy$)
- ▶ Spezialfall: kartesische oder äquidistante Gitter
 - ▶ Zellen mit gleicher Ausdehnung in allen Achsenrichtungen



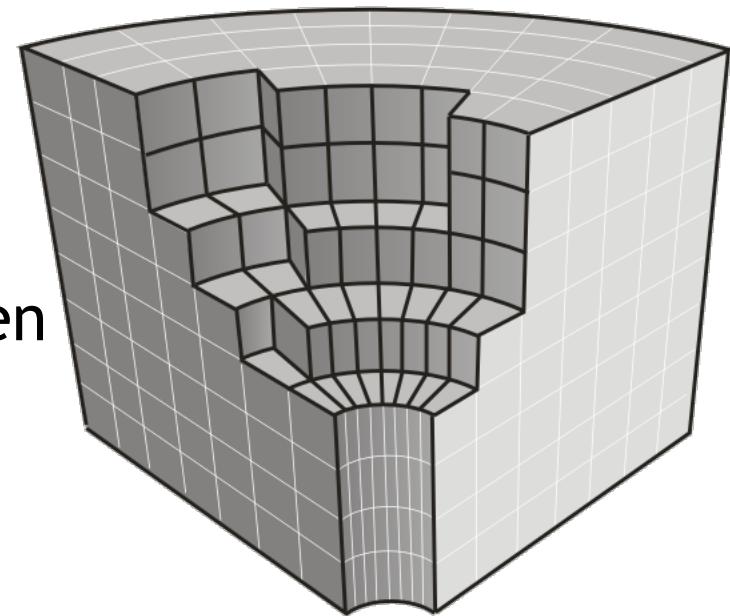
- Rektilineare Gitter

- Topologie ist regulär und implizit
- aber die Abstände zwischen den Gitterpunkten sind nicht konstant
 - nicht-lineare Skalierung der Positionen entlang der Achse(n)
 - oder: Abstände müssen explizit gespeichert werden
 - „orthogonales Produkt“ zweier Koordinatenfunktionen
- Beispiel: 2D rektilineares Gitter mit Spacing $x(\cdot)$ und $y(\cdot)$

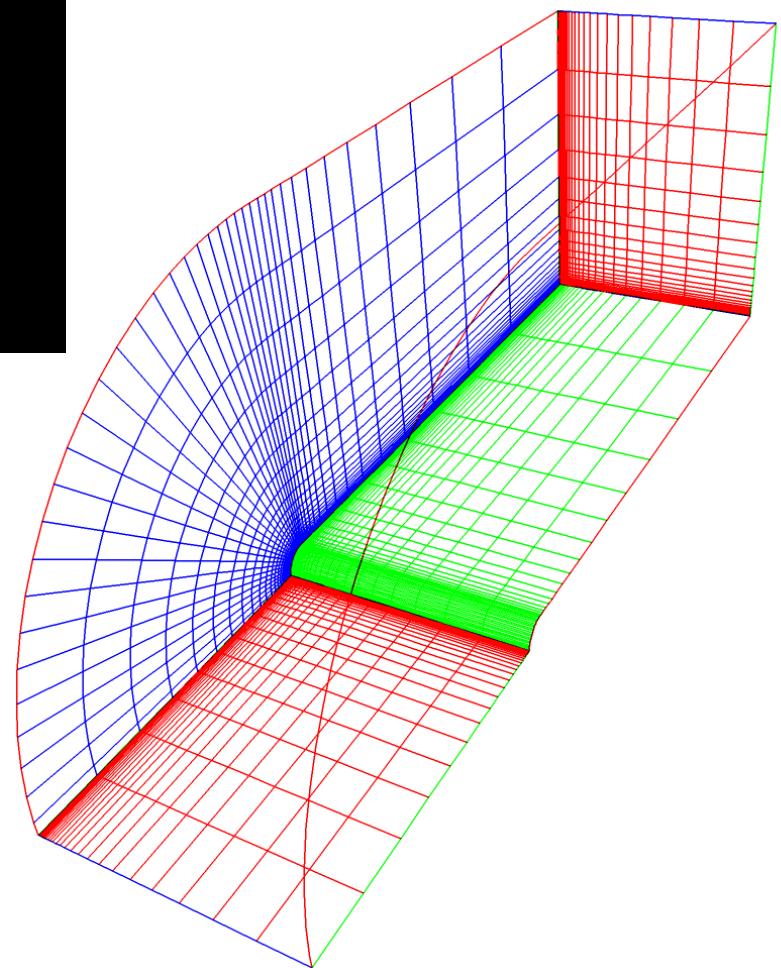
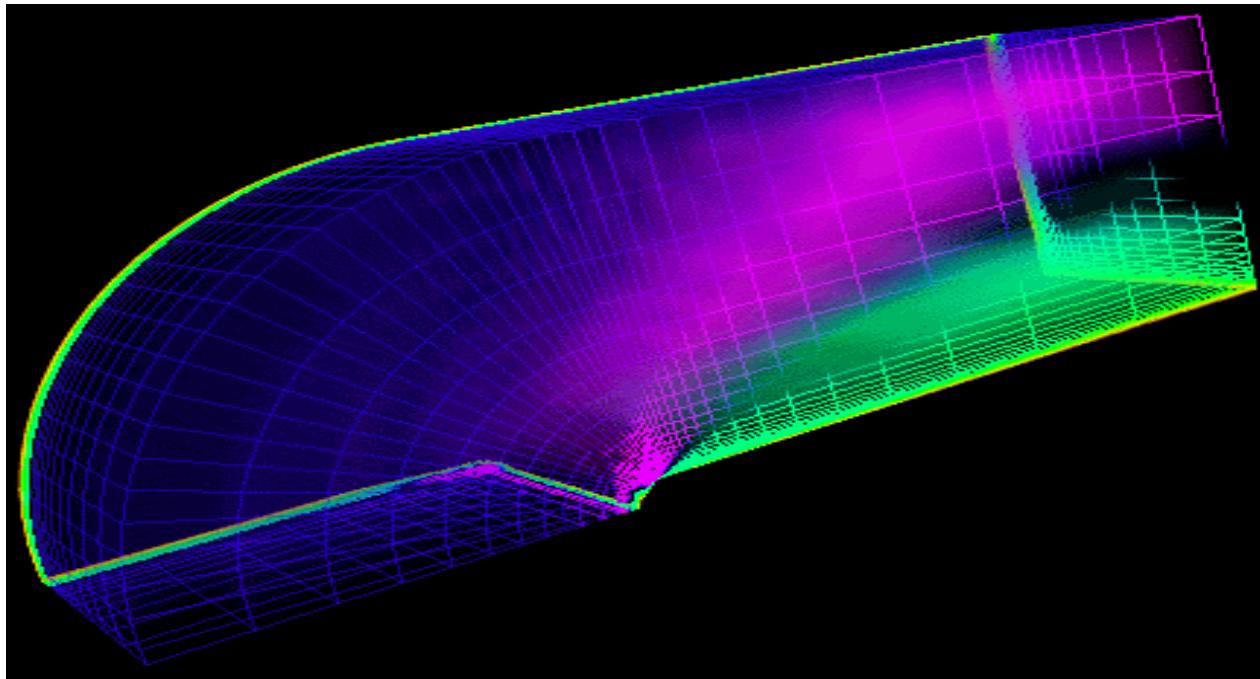


Gekrümmte Gitter (curvilinear grids)

- Topologie ist regulär und implizit
- Abstände zwischen den Gitterpunkten sind irregulär: die Positionen sind nicht-linear transformiert
- Vertexpositionen werden explizit gespeichert
 - z.B. in 3D-Arrays `x_coord[L,M,N]`, `y_coord[L,M,N]`, `z_coord...`
- durch die Krümmung kann die geometrische Struktur auch konkave Formen annehmen
 - das klingt an dieser Stelle trivial, hat aber Auswirkungen auf einige Visualisierungs-/Rendering-Techniken

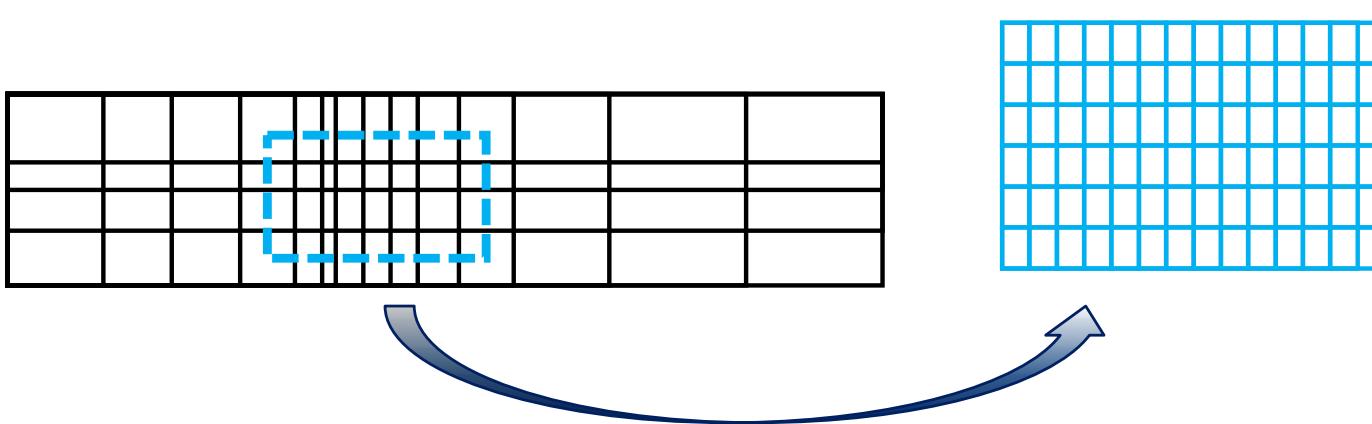
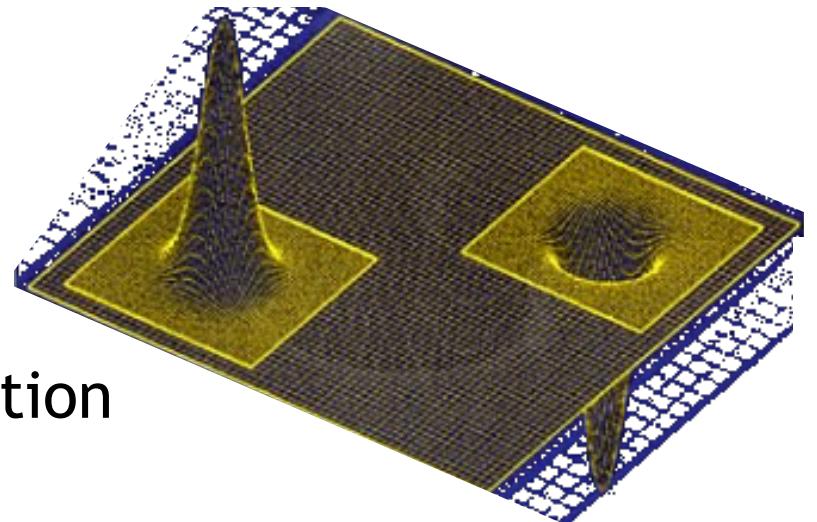


Gekrümmte Gitter (curvilinear grids)



Multigrids / Nested Grids

- verschachtelte oder überlagerte Gitter
- hohe Auflösung in wichtigen Bereichen, kein unnötiges Detail in unwichtigen
- eventuell Schwierigkeiten im Übergangsbereich, z.B. bei der Interpolation

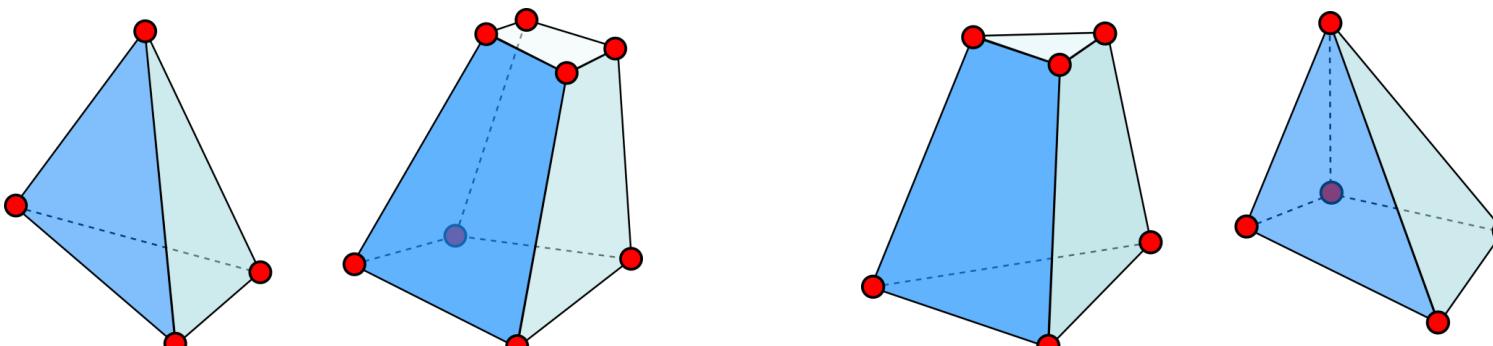


Fazit/Eigenschaften von strukturierten Gittern

- können in einem 2D bzw. 3D Array gespeichert werden
- Zugriff auf Zellen direkt durch Indizierung eines Eintrags
- topologische Information ist implizit vorhanden
 - daher: direkter Zugriff auf benachbarte Elemente/Zellen
- kartesische, uniforme und rektilineare Gitter sind konvex
 - daher: Sortierung der Zellen, z.B. vorne-hinten, ist für eine beliebige Blickrichtung implizit
 - aber: das rigide Layout erlaubt keine lokale Adaptivität
- gekrümmte Gitter erlauben eine flexiblere Modellierung von Beobachtungsräumen für beliebig geformte Objekte
 - aber: bei der Visualisierung benötigt man oft eine Sortierung, die durch die geometrische Form aufwändiger wird

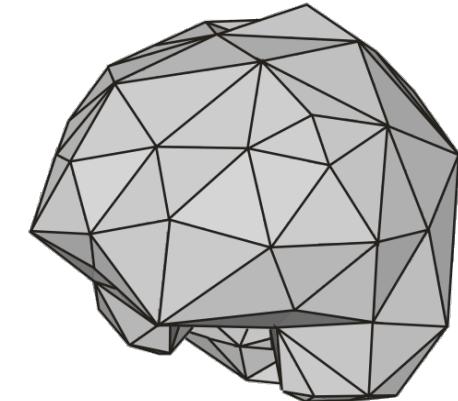
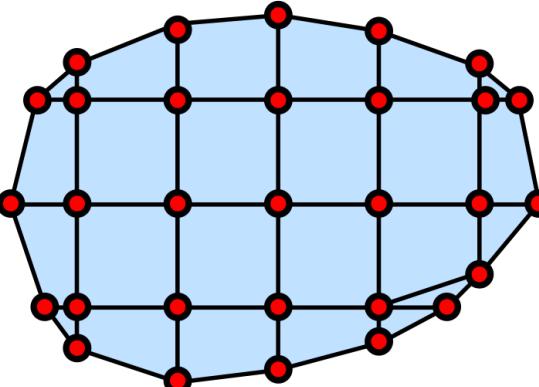
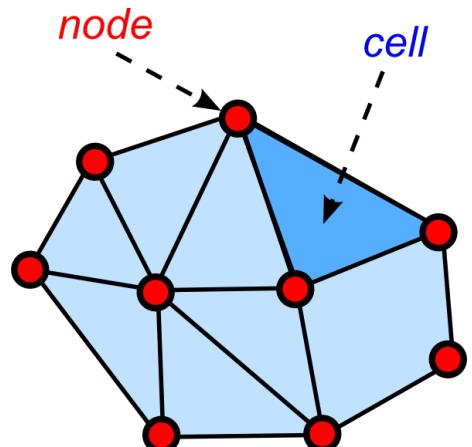
- **Unstrukturierte Gitter**

- ▶ unstrukturiert \leftrightarrow keine implizite Topologie (Konnektivität) gegeben
 - ▶ oft berechnet durch Triangulation von Punktmenzen also aus gitterfreien Daten (scattered data/points)
- ▶ Eigenschaften
 - ▶ Geometrie und Topologie müssen gespeichert werden
 - ▶ **spezielle Datenstrukturen für effiziente Traversierung und Suche**
 - ▶ oft bestehend aus Dreiecken/Vierecken (2D), Tetraeder/Prismen/Hexaeder/Pyramiden (3D)



Unstrukturierte Gitter

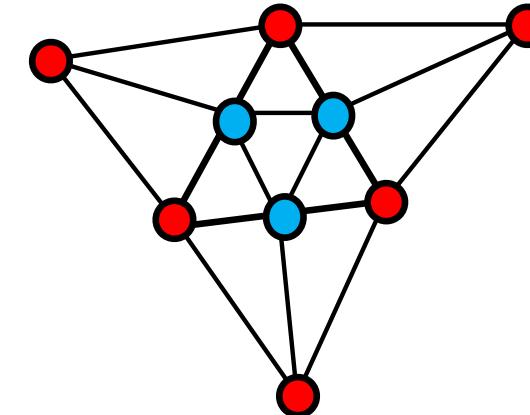
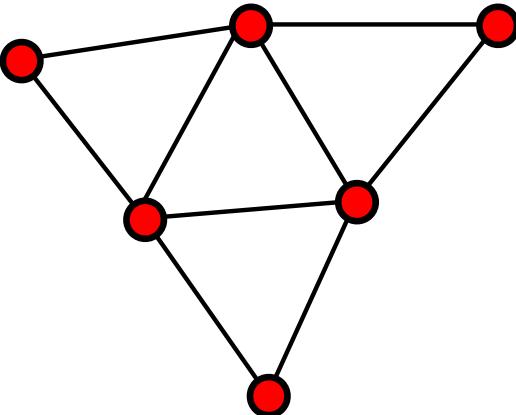
- oft bestehen die Gitter aus einem Zellentyp, aber es sind Hybrid- und Mischformen möglich
- 2D-Zellen sind oft im 3D-Raum definiert, z.B. um Randbedingungen in Simulationen zu definieren
- am häufigsten: Dreiecksnetze in 2D, Tetraedergitter in 3D



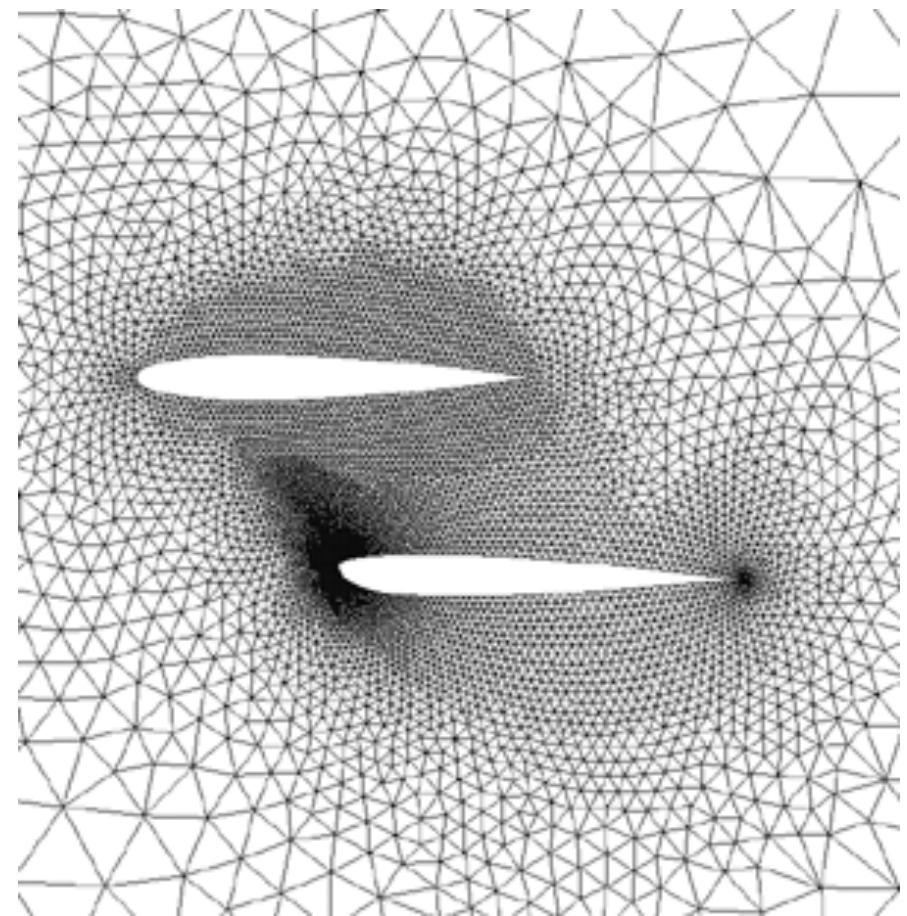
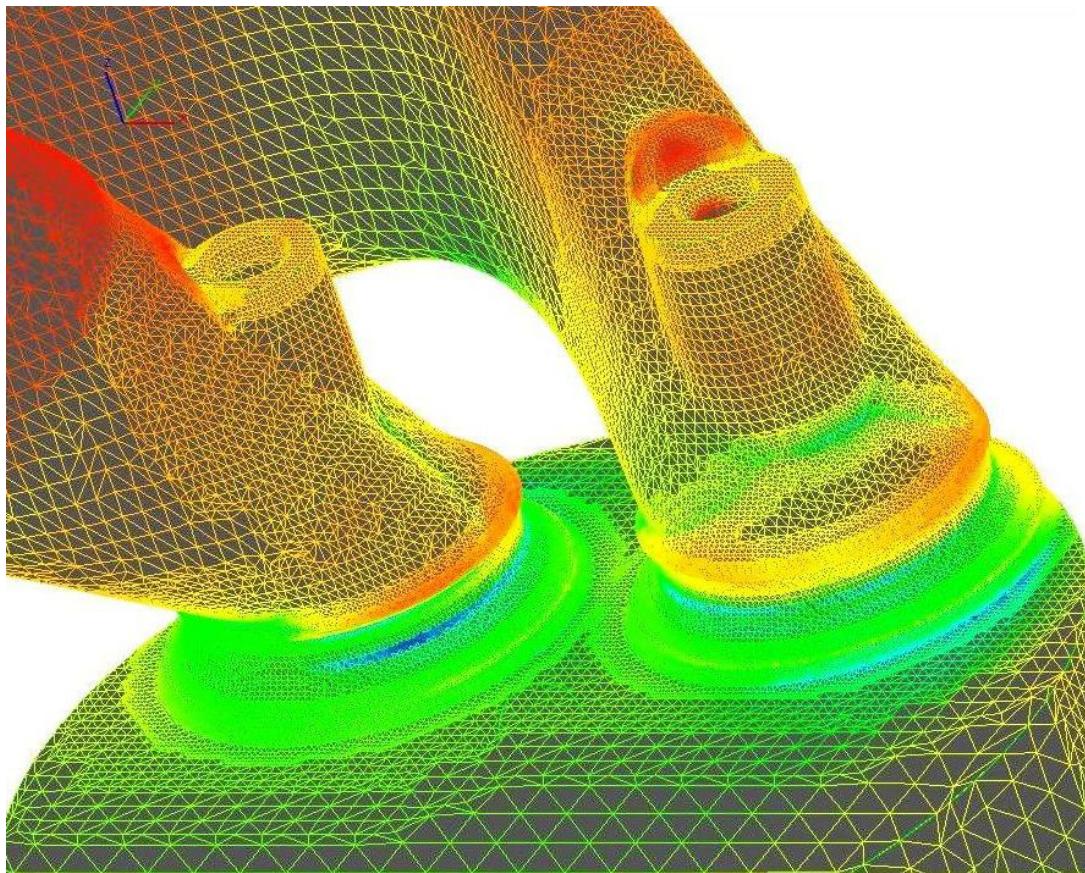
Unstrukturierte Gitter

- Vorteile:

- Anpassung an lokale Features (kleine und große Zellen) durch „Adaptive Refinement“
- einfache lineare Interpolation in den Simplizes (aber nicht immer einfach in anderen Zelltypen)

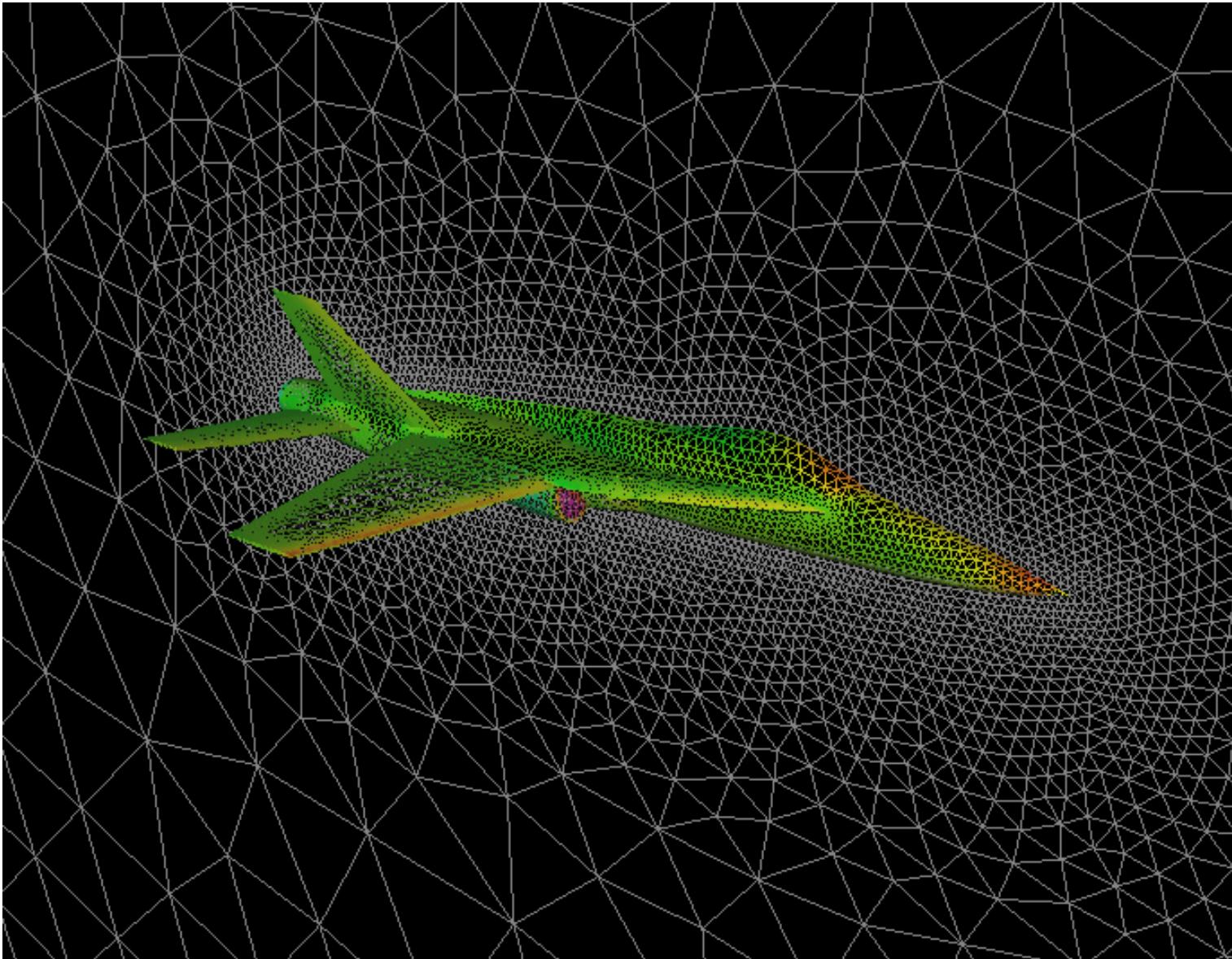


Beispiele: Unstrukturierte Gitter



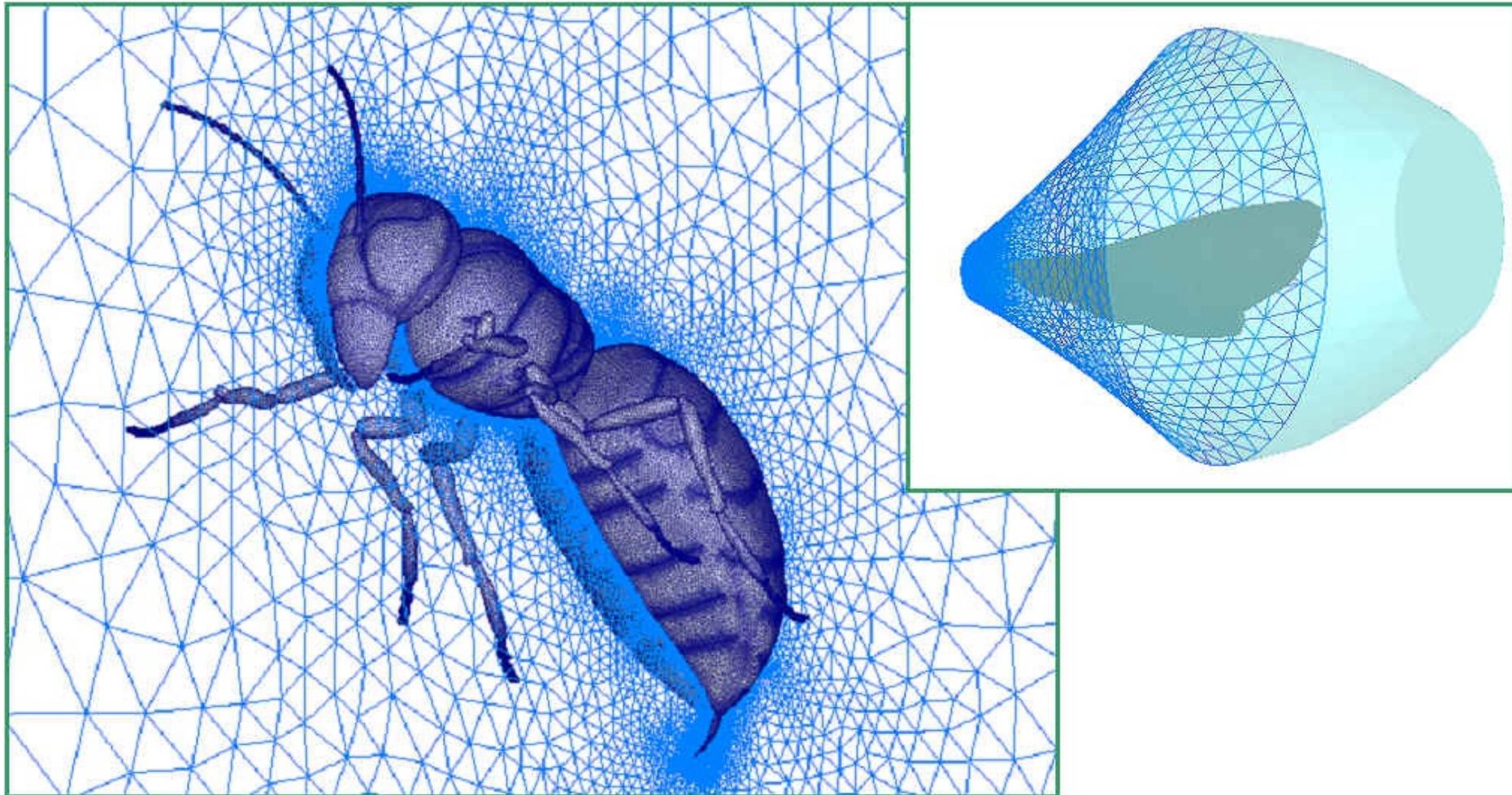
Unstrukturierte Gitter

- Beispiel: Anpassung an lokale Details



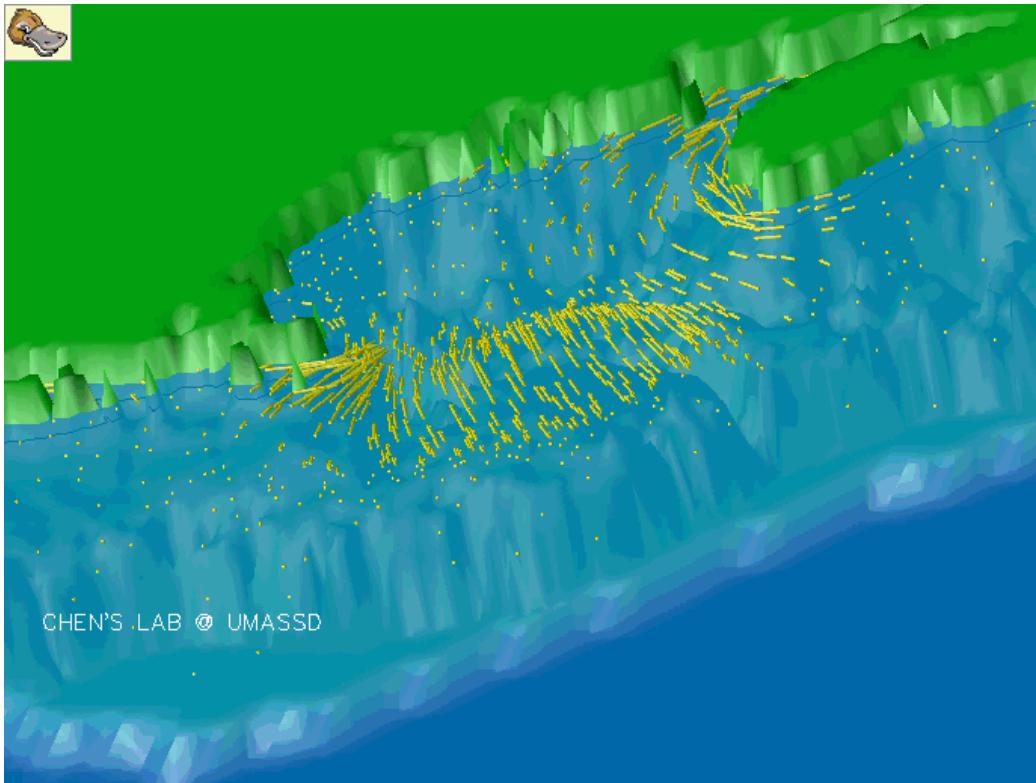
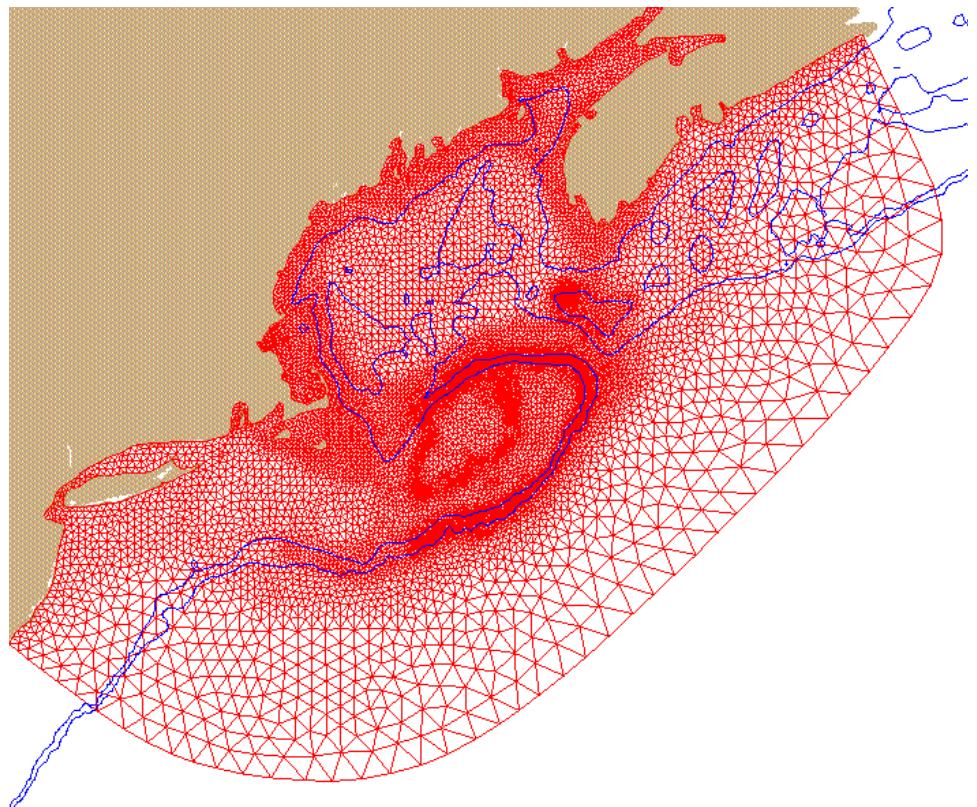
Unstrukturierte Gitter

- Beispiel: Anpassung an lokale Details



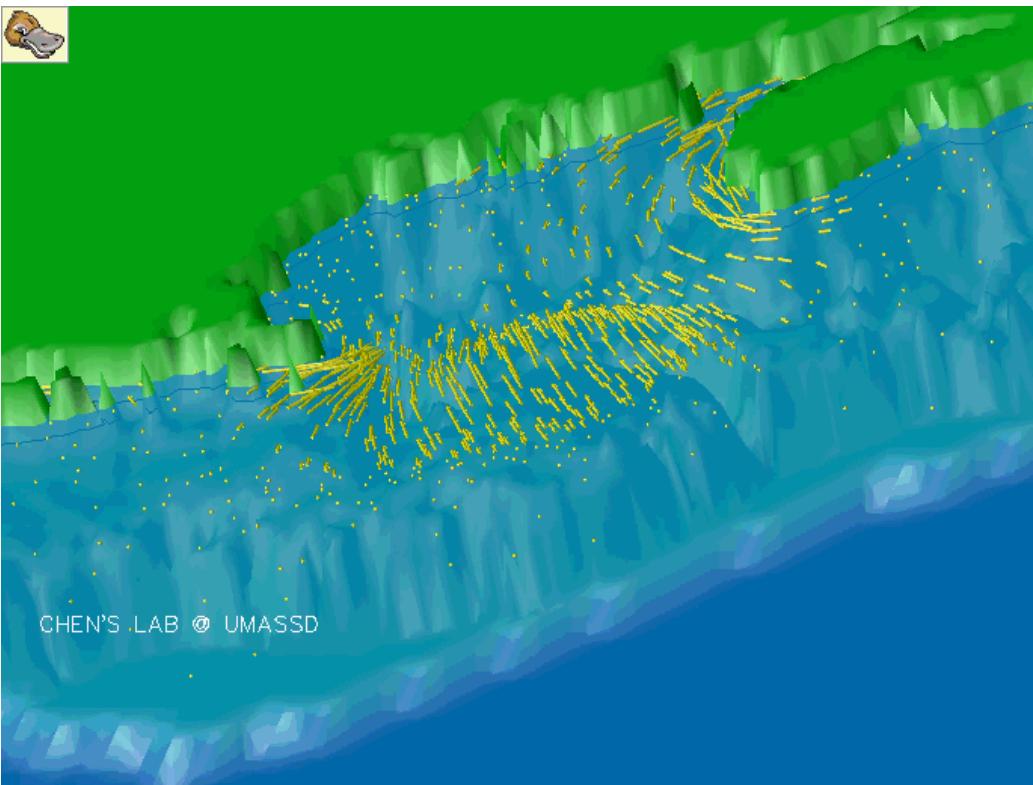
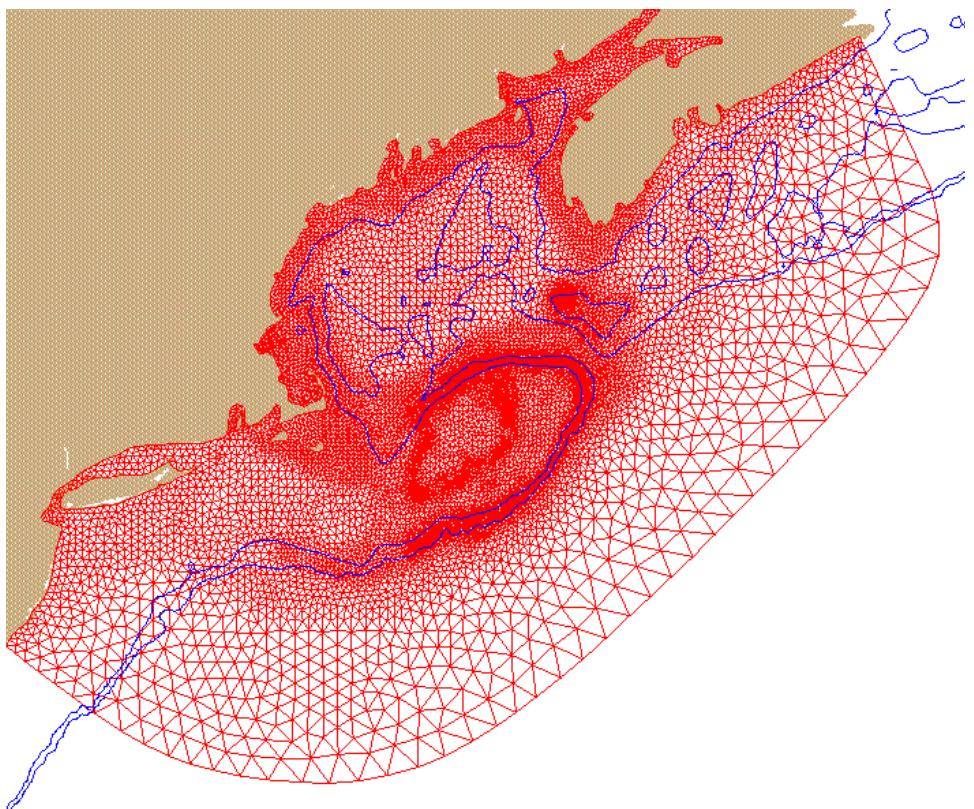
Unstrukturierte Gitter

- Beispiel: Anpassung an lokale Details



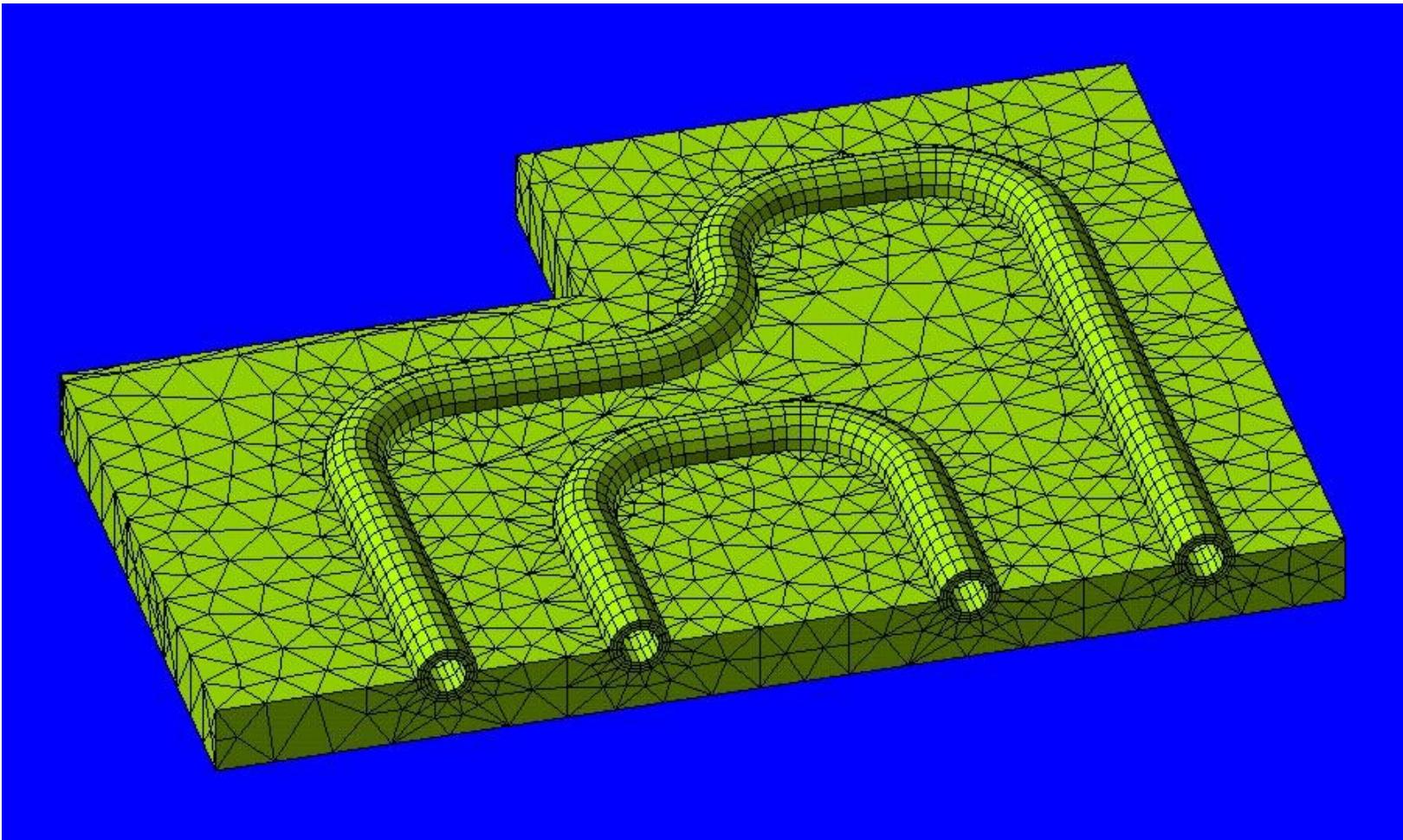
Unstrukturierte Gitter

- Beispiel: Anpassung an lokale Details

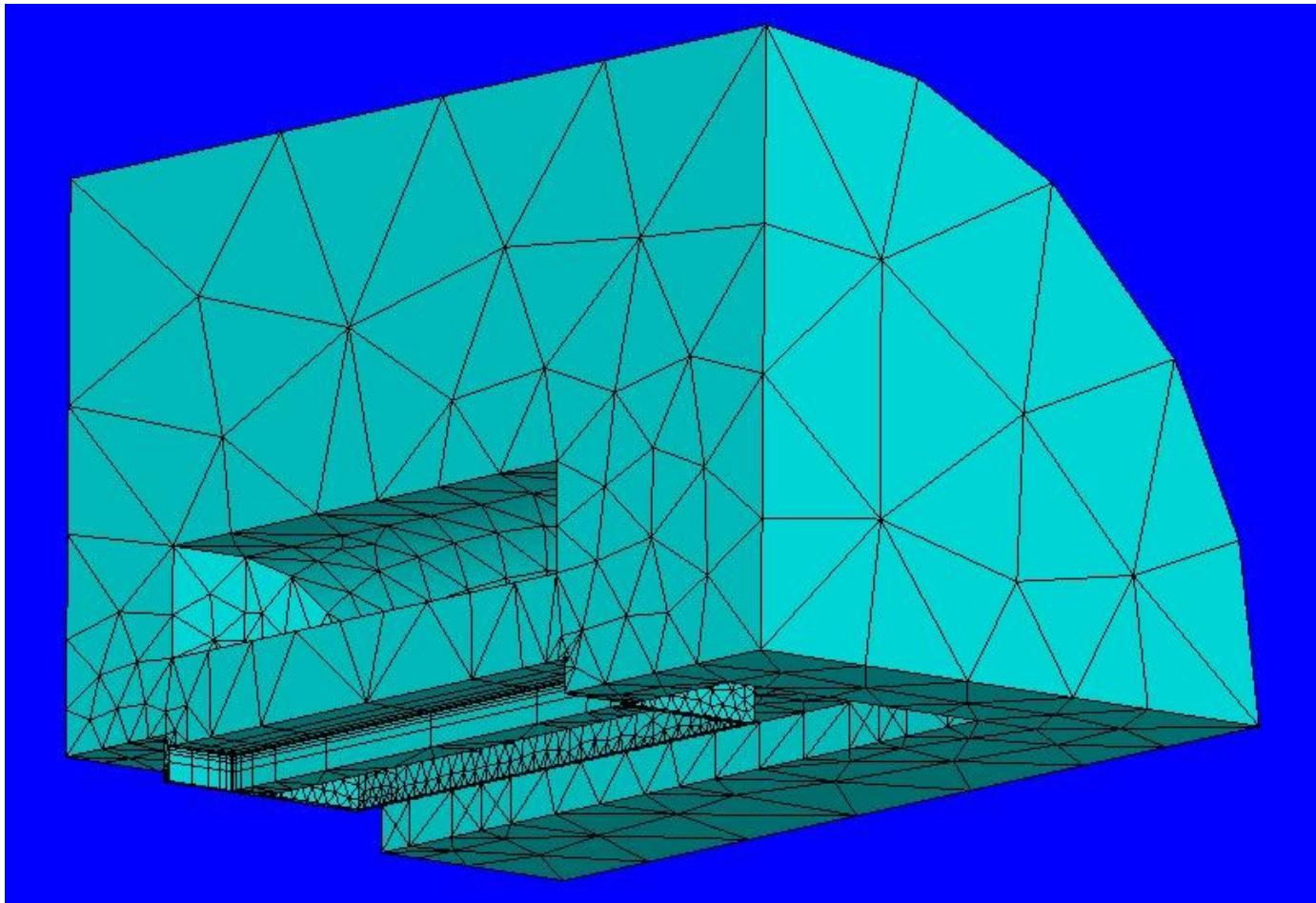


Hybride Gitter

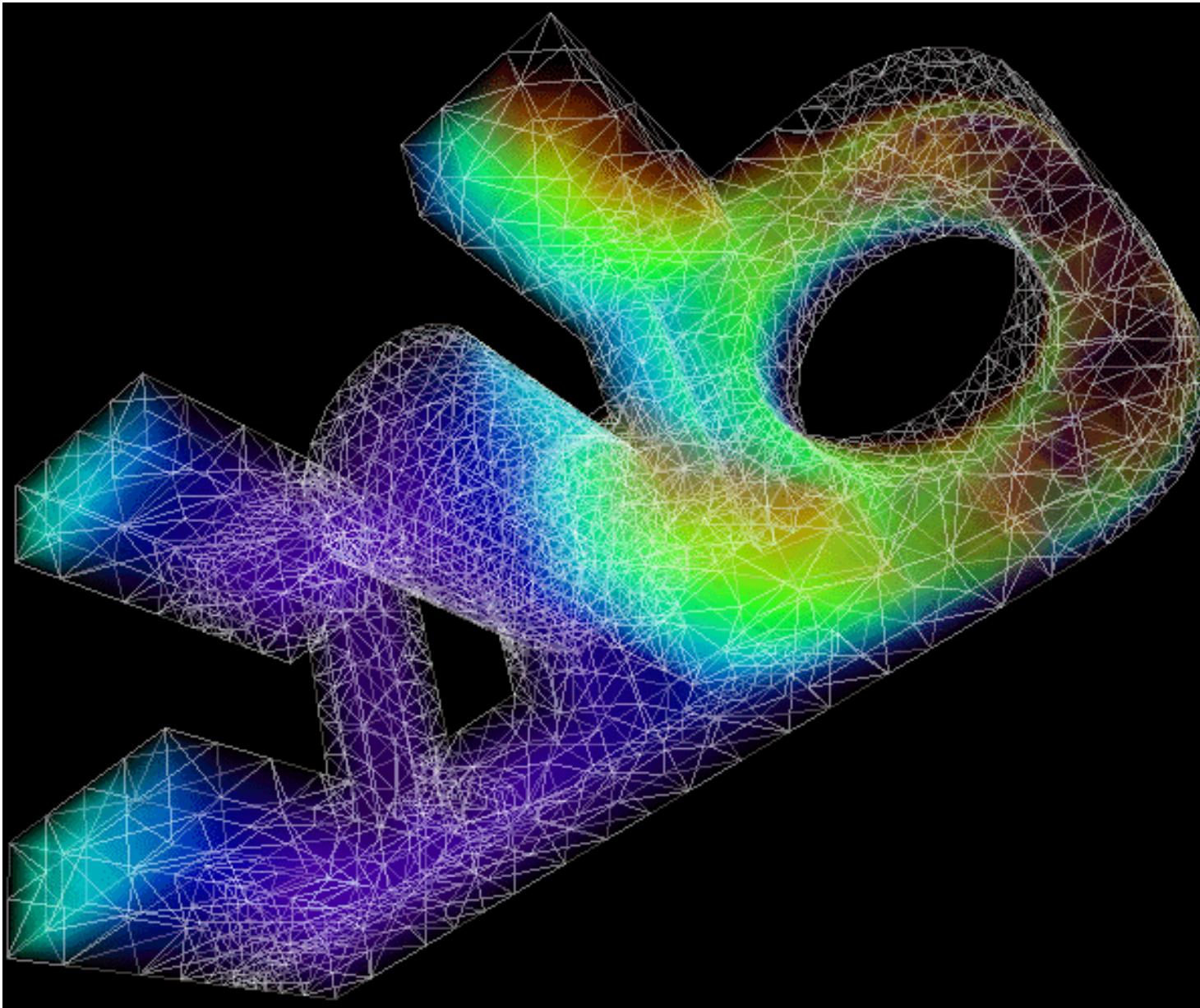
- hier: eine Kombination aus teils strukturiertem und teils unstrukturiertem Gitter



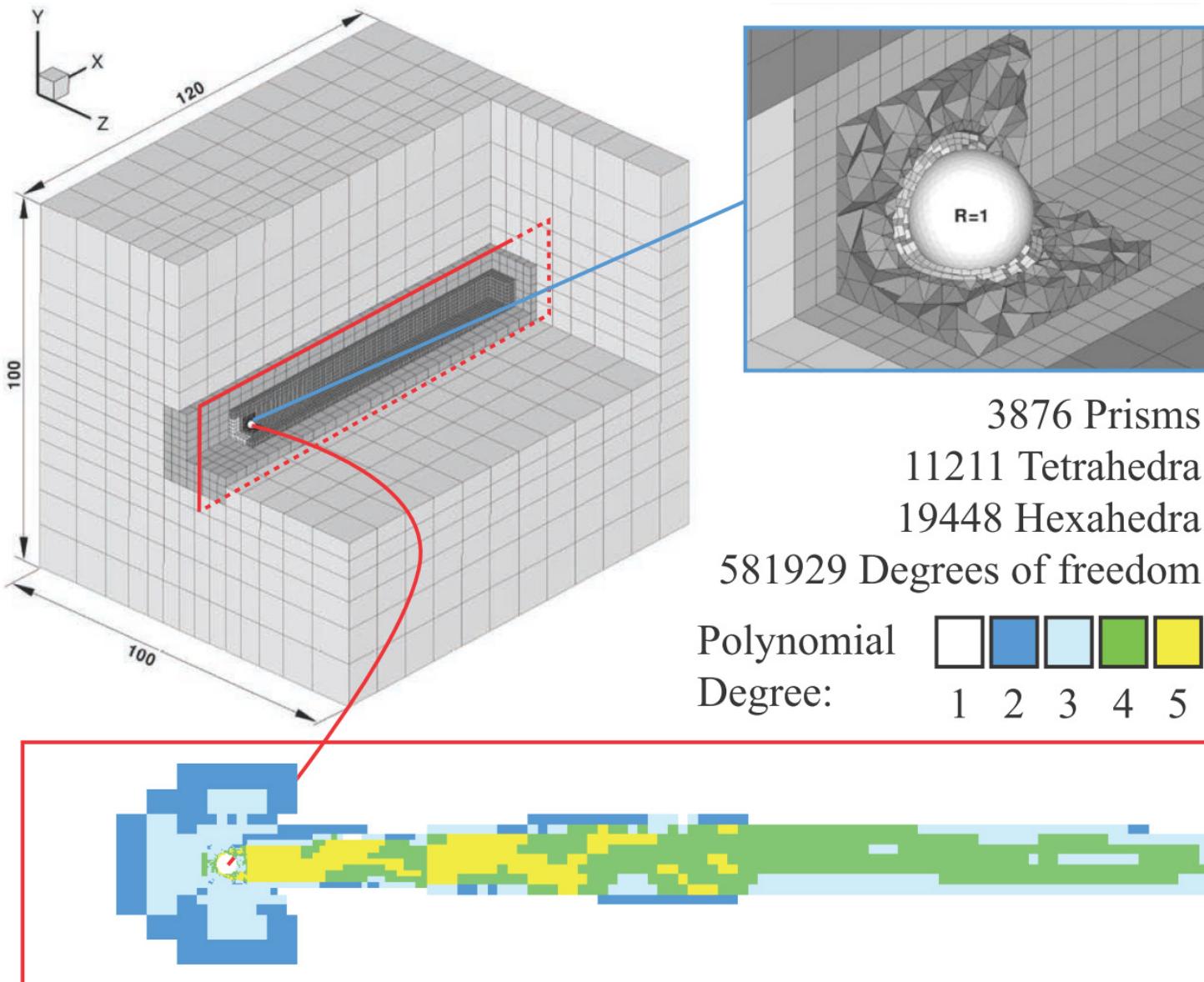
- Beispiel: hybrides unstrukturiertes und rektilineares Gitter



- Beispiel: unstrukturiertes Gitter



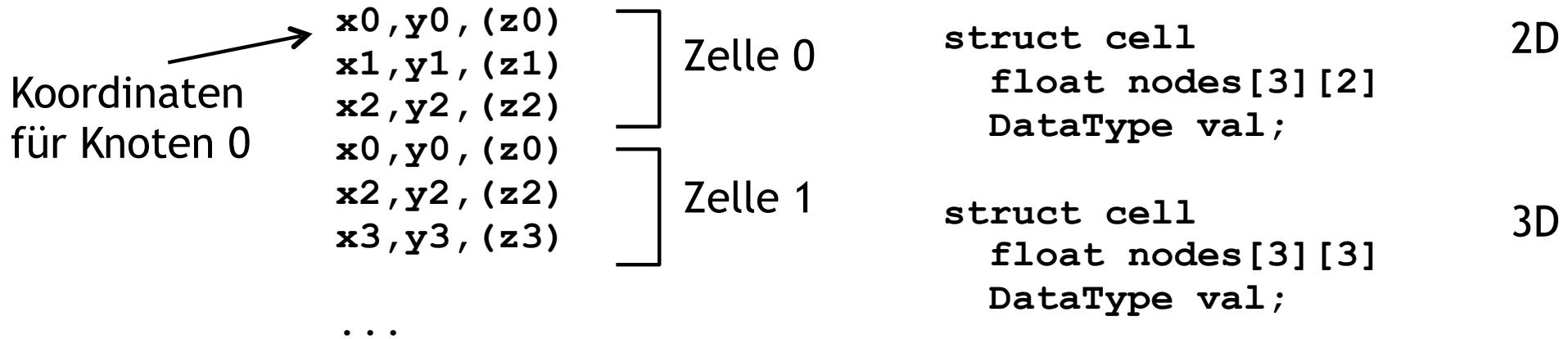
Kombination unterschiedlicher Gittertypen (Beispiel: Simulation Flüssigkeitsströmung um eine Kugel)



- Datenstrukturen für unstrukturierte Gitter

- ▶ die einfachste Repräsentation ist die **direkte Form**

- ▶ Beispiel: ein Dreiecksnetz in \mathbb{R}^2 bzw. \mathbb{R}^3 mit Daten pro Zelle (= Dreieck)



- ▶ oft kommen noch Datenwerte an den Knoten hinzu (hier weggelassen)
- ▶ Probleme dieser Datenstruktur
 - ▶ Redundanz und Speicherverbrauch
 - ▶ keine Suche nach Nachbarzellen o.ä.

- Datenstrukturen für unstrukturierte Gitter

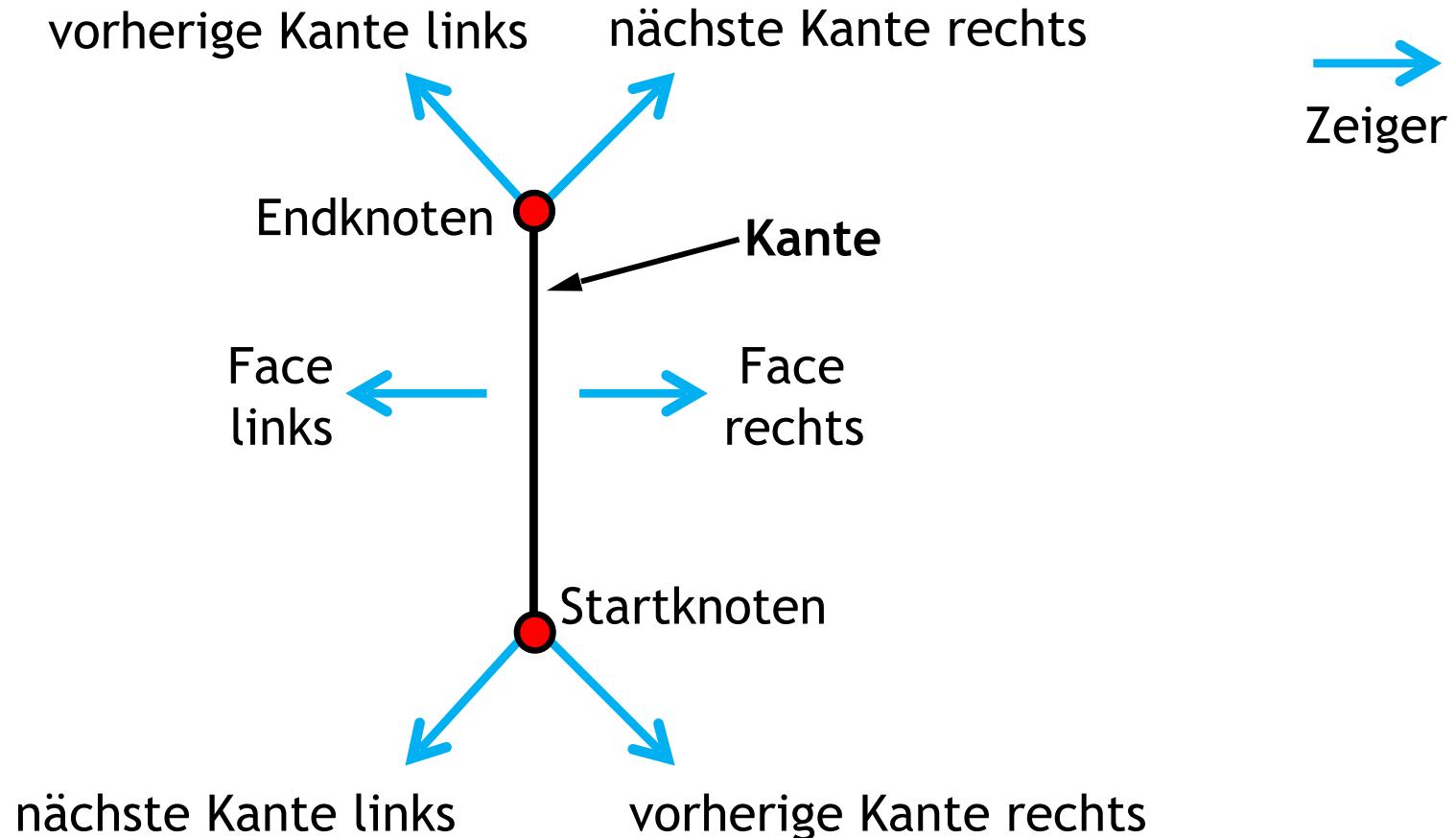
- ▶ die **indirekte Form** kennen Sie schon...
- ▶ Beispiel: 2D-Netz in \mathbb{R}^2 (\mathbb{R}^3), Daten pro Knoten und Zelle

	Knotenliste	Knotendaten	Zelltypliste	Zellenliste	Zellendaten
Koordinaten für Knoten 0	x0,y0,(z0) x1,y1,(z1) x2,y2,(z2) x3,y3,(z3)	4.32 6.72 5.78 -3.54	Dreieck Dreieck Viereck	0,1,2 0,2,3 2,3,0,1	2.43 -8.68 6.32

- ▶ wird auch Indexed Cell Set genannt (den Spezialfall Indexed Face Set bzw. Shared Vertex kennen Sie schon aus der CG Vorlesung)
- ▶ benötigt weniger Speicher als die direkte Form
- ▶ üblicherweise gibt es Daten pro Knoten oder Zelle, nicht beides
 - ▶ und meistens dann auch mehr als nur einen Skalarwert
- ▶ aber: immer noch keine Möglichkeit zur Suche von Nachbarn etc. (abgesehen von einer globale Suche über alle Zellen)

Beispiel: Datenstrukturen für unstrukturierte Gitter

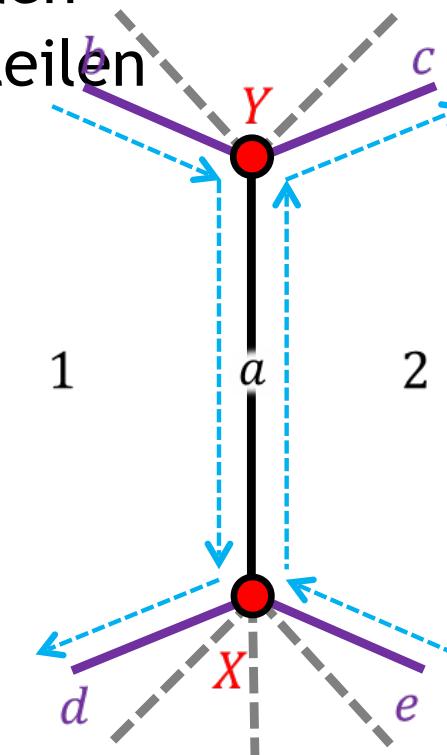
- die kantenbasierte Winged-Edge Datenstruktur [Baumgart 1975]



- weitere Datenstrukturen (für Tetraedernetze): <http://www.cc.gatech.edu/~topraj/SPM09files/sot.pdf>

Winged-Edge Datenstruktur

- eine kantenbasierte Datenstruktur, die ermöglicht...
 - Suche nach Facetten, die sich eine Kante teilen
 - Suche nach Facetten, die sich einen Knoten teilen
 - laufen entlang der Kanten um ein Face
- speichert für jeden Knoten einen Zeiger auf eine anliegende Kante
- speichert für jedes Face einen Zeiger auf eine seiner Kanten
- implizite Annahme in dieser Datenstruktur: jede Kante hat max. 2 anliegende Flächen (2-Mannigfaltigkeit, 2-manifold topology)

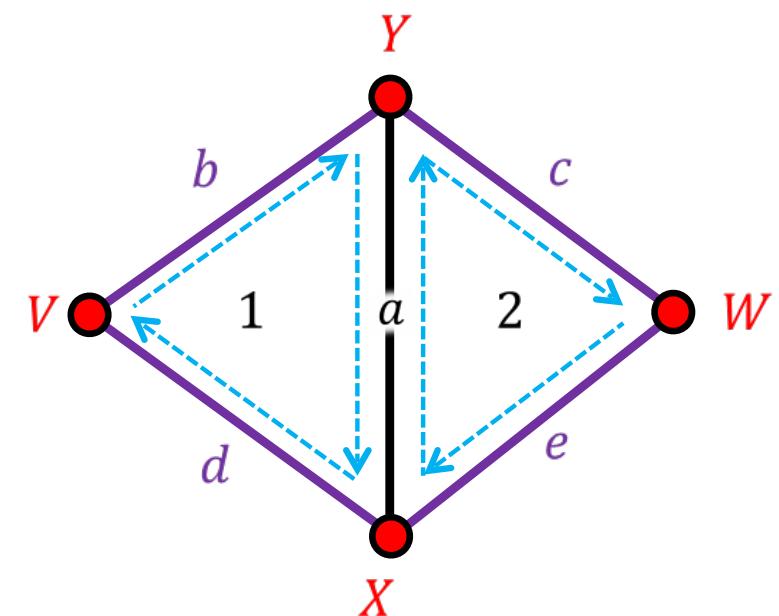


Kante	Knoten		Faces		Traversierung links		Traversierung rechts	
Name/Idx	Start	Ende	links	rechts	Vorgänger	Nachfolger	Vorgänger	Nachfolger
a	X	Y	1	2	b	d	e	c

Beispiel: Winged-Edge Datenstruktur

Dreieck	Kante
1	a
2	c

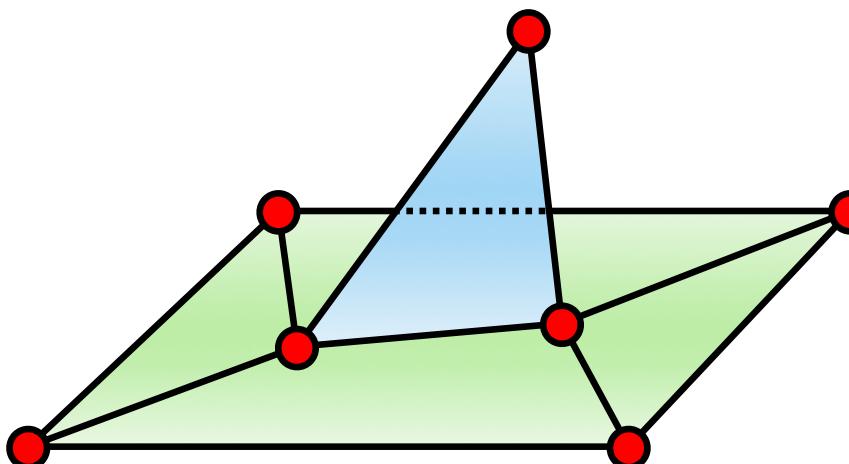
Knoten	Position	Kante
V	...	b
W	...	c
X	...	a
Y	...	c



Kante	Knoten		Faces		Traversierung links		Traversierung rechts	
	Name/Idx	Start	Ende	links	rechts	Vorgänger	Nachfolger	Vorgänger
a	X	Y	1	2	b	d	e	c
b	V	Y	--	1	--	--	d	a
c	Y	W	--	2	--	--	a	e
d	X	V	--	1	--	--	a	b
e	W	X	--	2	--	--	c	a

- **Netze mit 2-Mannigfaltigkeit**

- ▶ salopp: das Netz repräsentiert **eine** geschlossene Fläche ohne zusätzliche Polygone die daran angebracht sind
- ▶ Eigenschaften
 - ▶ local disc property: für jeden Punkt auf der Fläche lässt sich eine ϵ -Umgebung mit Kreistopologie (Halbkreis an den Kanten) finden
 - ▶ Edge Ordering Property: Nachbarvertices jedes Vertex lassen sich eindeutig im/gegen den Uhrzeigersinn aufzählen
 - ▶ scharfe Kanten sind natürlich möglich (es geht hier um Topologie!)
- ▶ Beispiel für **keine** 2-Mannigfaltigkeit:

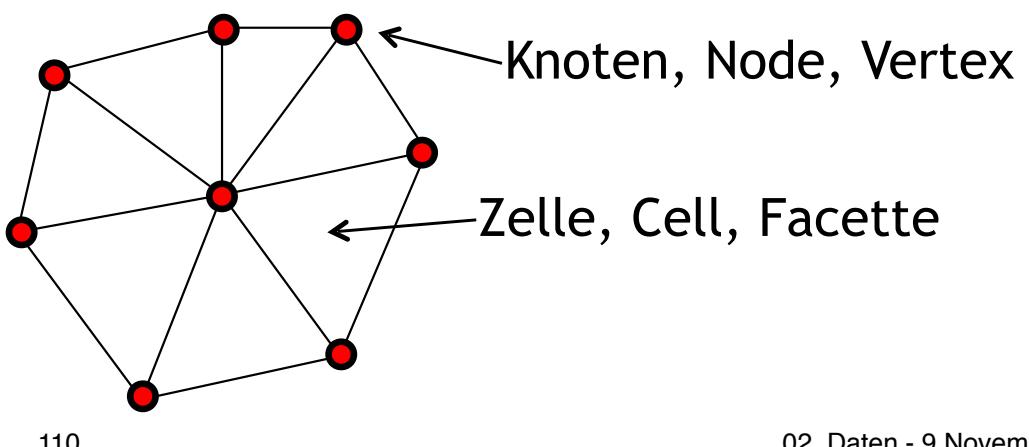
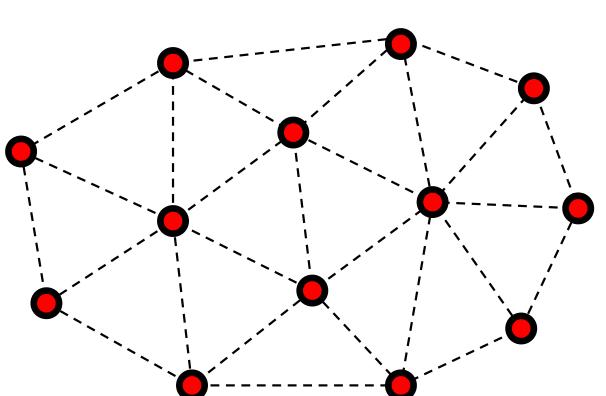


Gitterfreie Daten

- irregulär verteilt Positionen ohne Konnektivitätsinformation
- um Konnektivität zu erhalten (z.B. wenn man nicht Scattered Data Interpolation verwenden möchte) sucht man eine „gute“ Triangulierung
die Punkte werden zu Knoten des Dreiecks- oder Tetraedernetzes
es gibt viele mögliche Triangulierungen - welche ist gut?
ein Qualitätsmerkmal ist das Aspect Ratio von Dreiecken
vermeide lange, dünne Dreiecke

betrachte Verhältnis Inkreis zu Umkreis, oder minimalen und maximalen Winkel im Dreieck

Delaunay Triangulation



- unstrukturierte Gitter für gitterfreien Daten (Triangulierung)
- Interpolation mit/ ohne Gitter (Radial Basis Functions, bilinear, barycentric)
- Filtern