

CS 247 – Scientific Visualization

Lecture 7: Scalar Field Visualization, Pt. 1

Markus Hadwiger, KAUST

Reading Assignment #4 (until Feb 26)



Read (required):

- Real-Time Volume Graphics book, Chapter 5 until 5.4 inclusive (*Terminology, Types of Light Sources, Gradient-Based Illumination, Local Illumination Models*)
- Paper:
Marching Cubes: A high resolution 3D surface construction algorithm,
Bill Lorensen and Harvey Cline, ACM SIGGRAPH 1987
[> 18,600 citations and counting...]

<https://dl.acm.org/doi/10.1145/37402.37422>

Read (optional):

- Paper:
Flying Edges, William Schroeder et al., IEEE LDAV 2015

<https://ieeexplore.ieee.org/document/7348069>

Quiz #1: Mar 1



Organization

- First 30 min of lecture
- No material (book, notes, ...) allowed

Content of questions

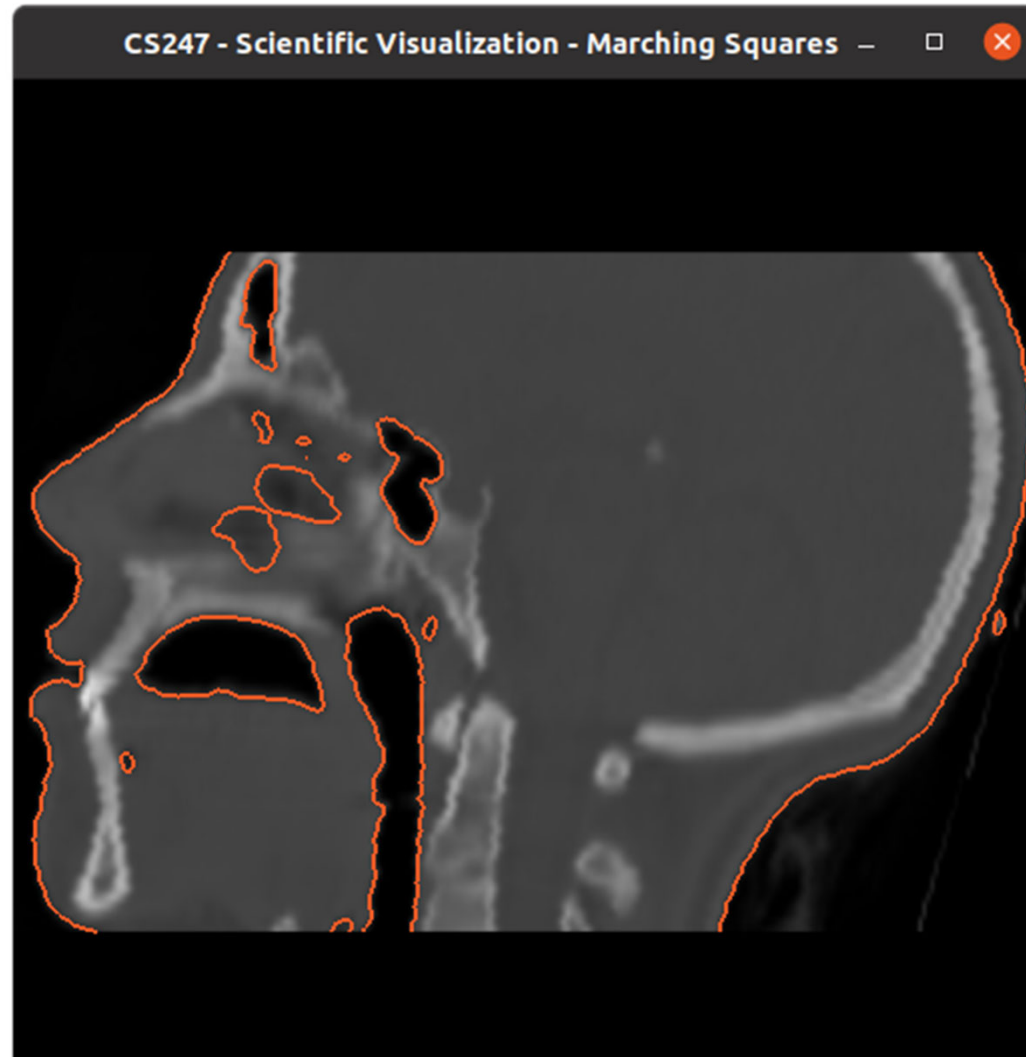
- Lectures (both actual lectures and slides)
- Reading assignments (except optional ones)
- Programming assignments (algorithms, methods)
- Solve short practical examples

Programming Assignments Schedule (tentative)

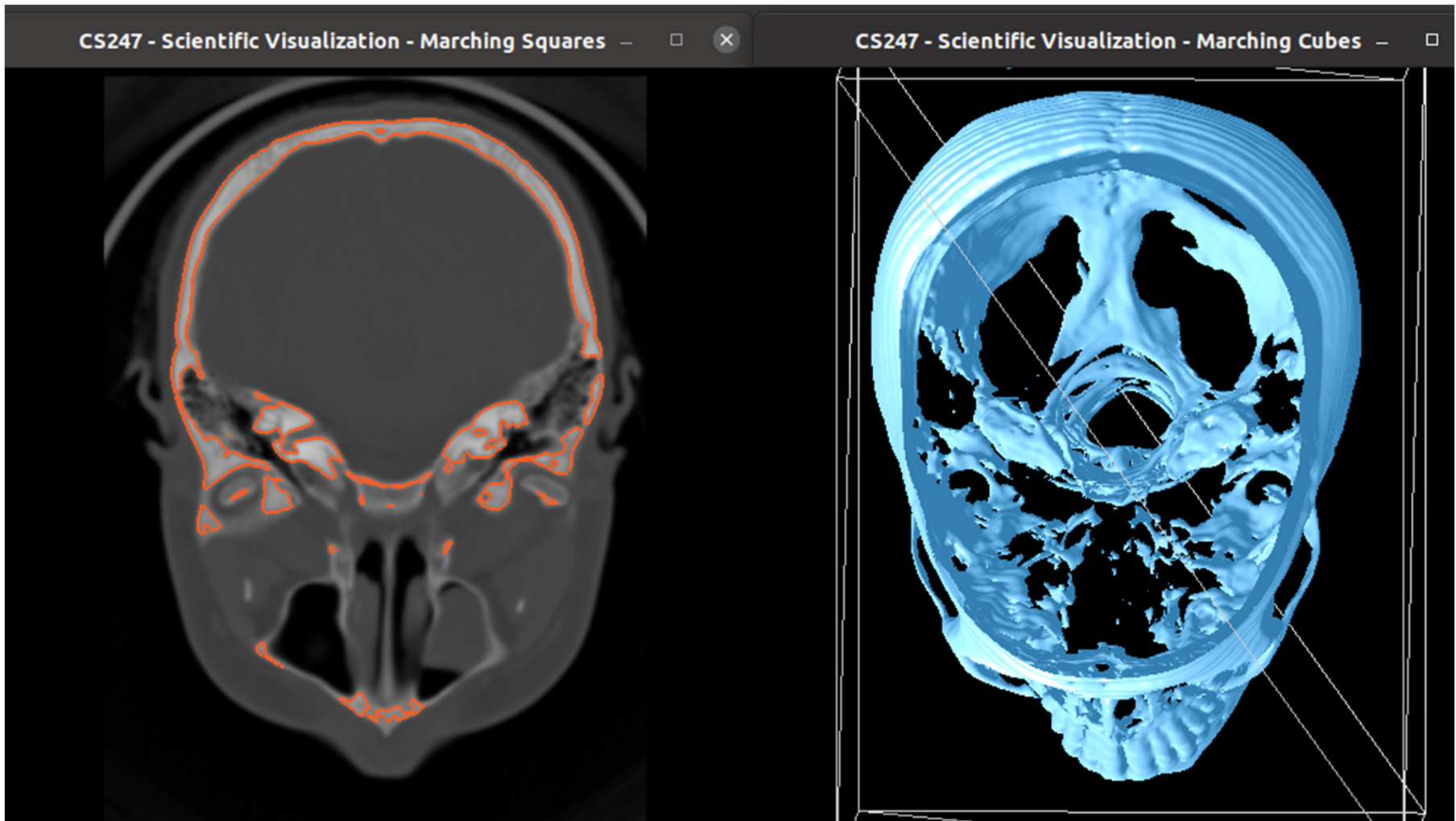


Assignment 0:	Lab sign-up: setup piazza + github account, get git repo Basic OpenGL example [we will offer a tutorial!]	until	Jan 29
Assignment 1:	Volume slice viewer	until	Feb 12
Assignment 2:	Iso-contours (marching squares)	until	Mar 1
Assignment 3:	Iso-surface rendering (marching cubes)	until	Mar 19
Assignment 4:	Volume ray-casting, part 1	until	Apr 2
	Volume ray-casting, part 2	until	Apr 9
Assignment 5:	Flow vis, part 1 (hedgehog plots, streamlines, pathlines)	until	Apr 30
Assignment 6:	Flow vis, part 2 (LIC with color coding)	until	May 10

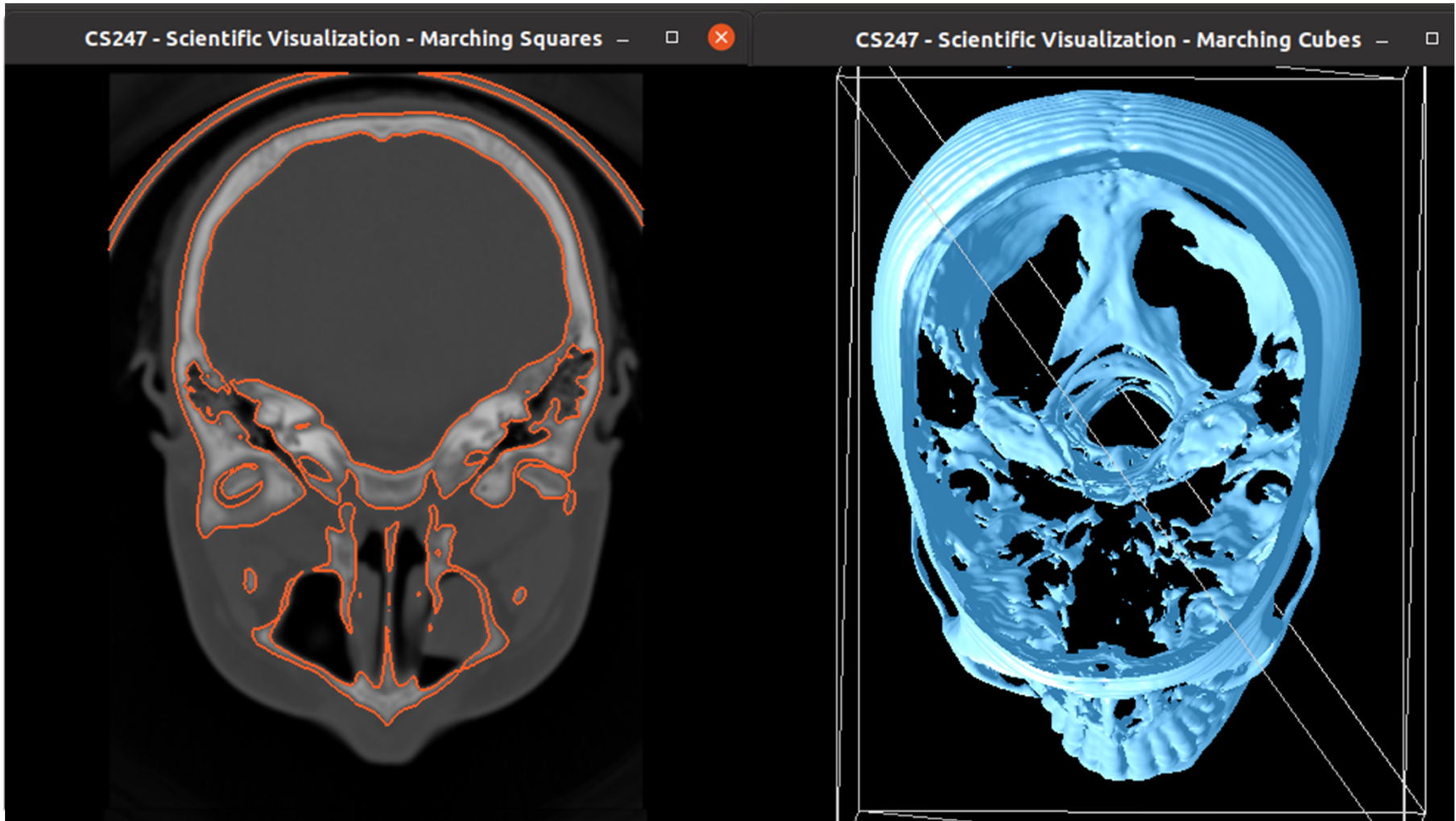
Programming Assignment 2



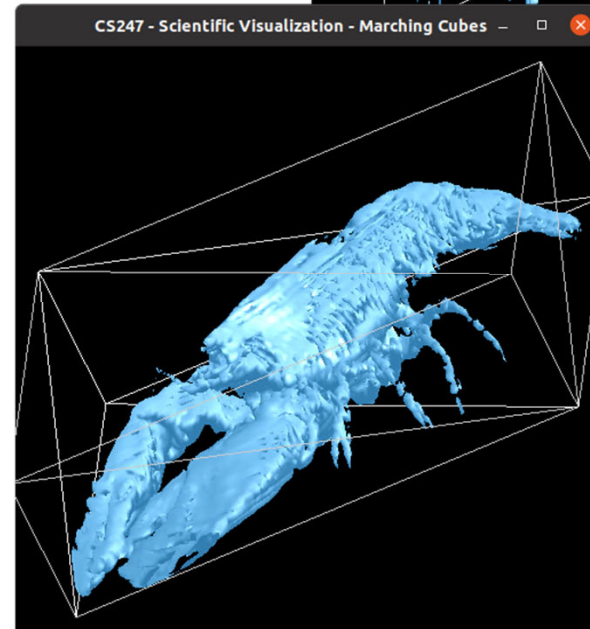
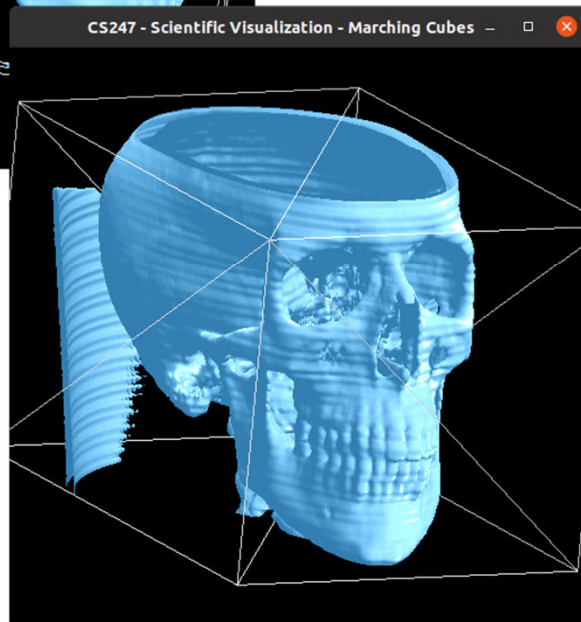
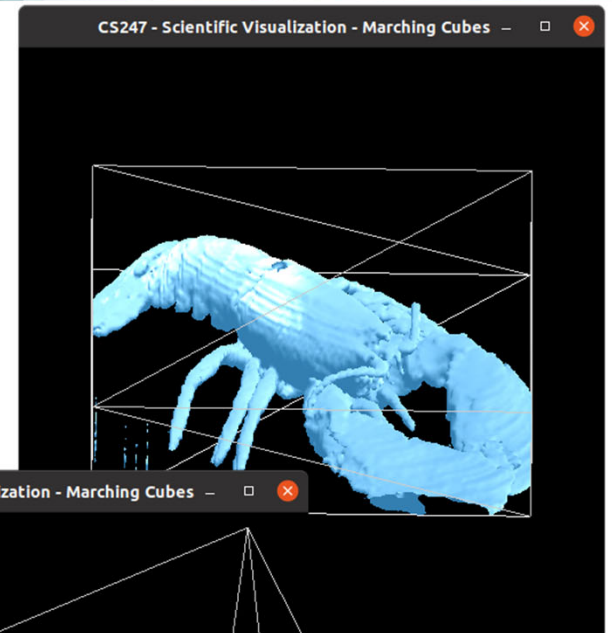
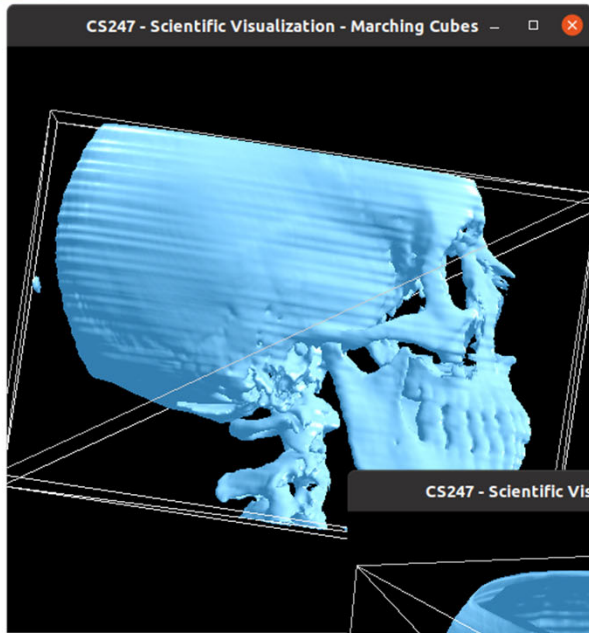
Programming Assignment 2 + 3



Programming Assignment 2 + 3



Programming Assignment 3



Scalar Fields

Scalar Fields are Functions



- 1D scalar field: $\Omega \subseteq \mathbb{R} \rightarrow \mathbb{R}$
- 2D scalar field: $\Omega \subseteq \mathbb{R}^2 \rightarrow \mathbb{R}$
- 3D scalar field: $\Omega \subseteq \mathbb{R}^3 \rightarrow \mathbb{R}$
→ **volume visualization!**

more generally: $\Omega \subseteq$ n-manifold

Basic Visualization Strategies



Mapping to geometry

- Function plots
- Height fields
- Isocontours/isolines, isosurfaces

Color mapping

Specific techniques for 3D data

- Indirect volume visualization
- Direct volume visualization
- Slicing

Visualization methods depend heavily on dimensionality of domain

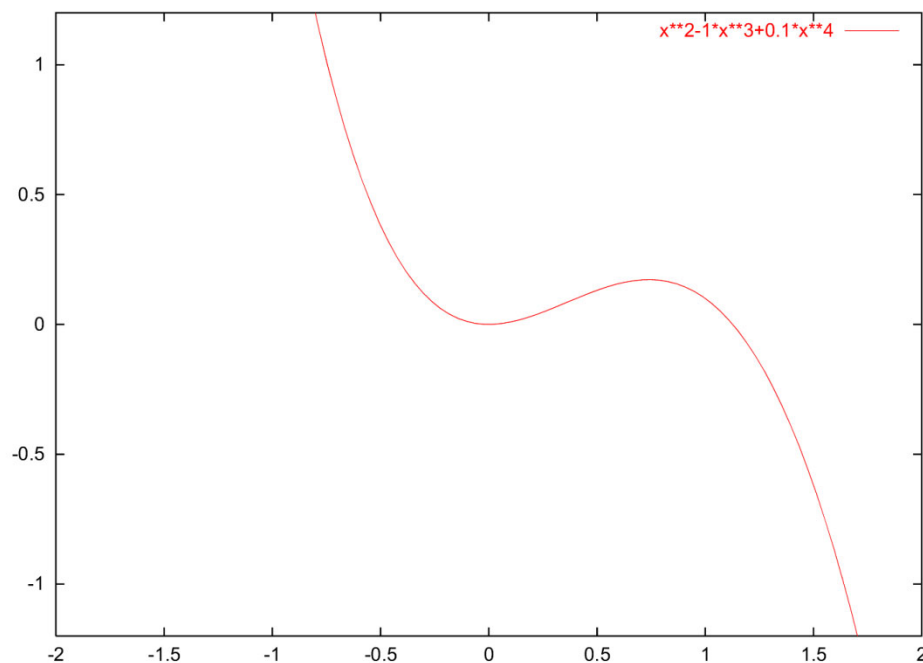
Function Plots and Height Fields (1)



Function plot for a 1D scalar field

$$\{(x, f(x)) \mid x \in \mathbb{R}\}$$

- Points
- 1D manifold: line



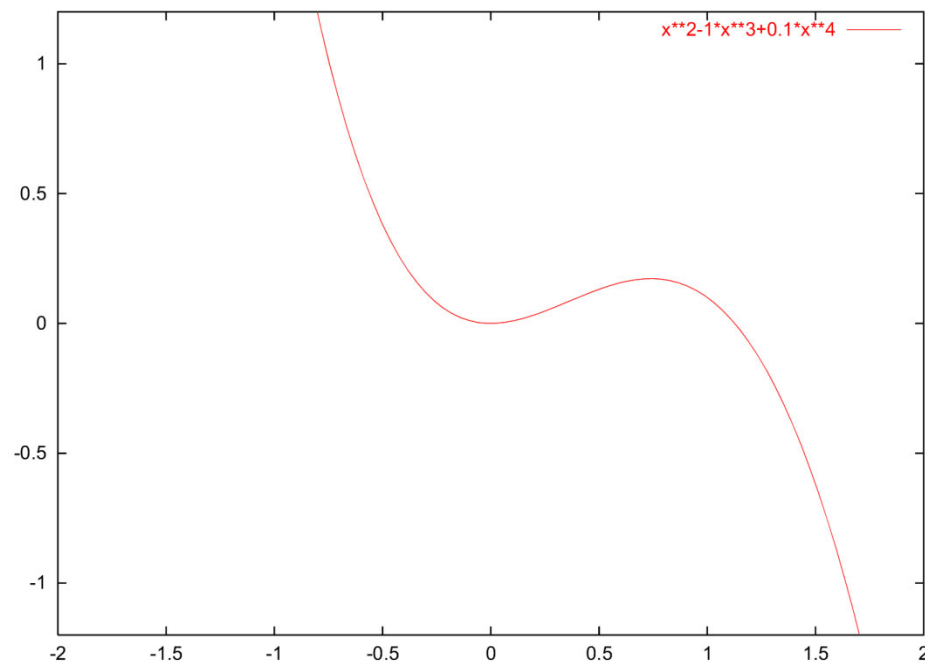
Function Plots and Height Fields (1)



Function plot for a 1D scalar field

$$\{(s, f(s)) \mid s \in \mathbb{R}\}$$

- Points
- 1D manifold: line



Function Plots and Height Fields (2)



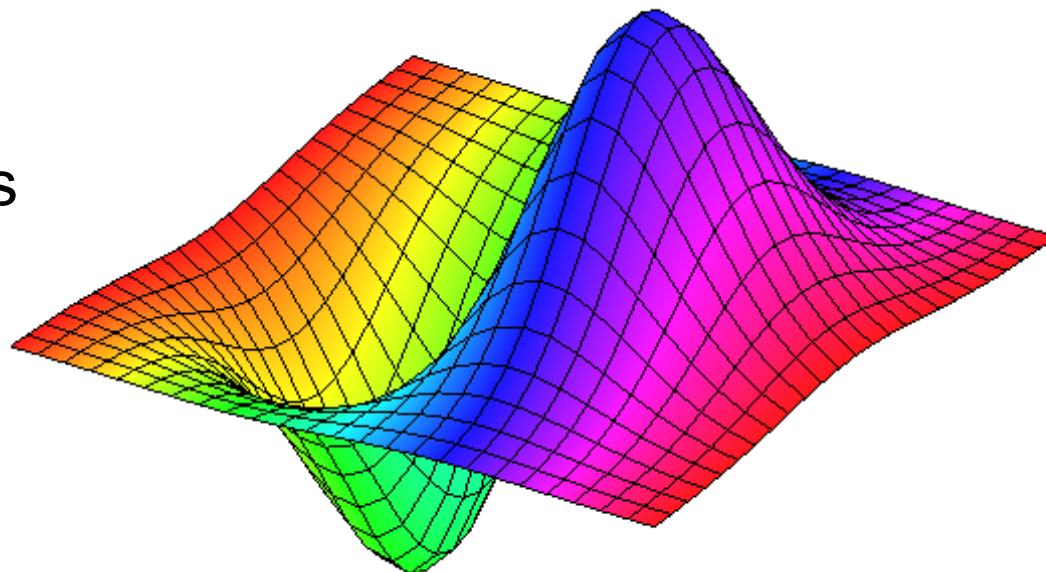
Function plot for a 2D scalar field

$$\{(x, f(x)) \mid x \in \mathbb{R}^2\}$$

- Points
- 2D manifold: surface

Surface representations

- Wireframe
- Hidden lines
- Shaded surface



Function Plots and Height Fields (2)



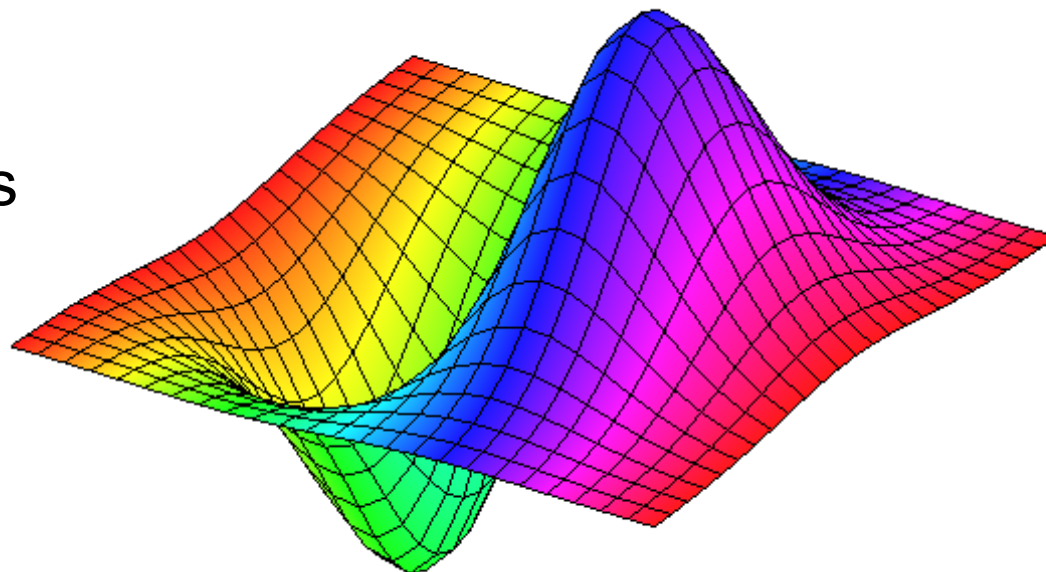
Function plot for a 2D scalar field

$$\{(s, t, f(s, t)) \mid (s, t) \in \mathbb{R}^2\}$$

- Points
- 2D manifold: surface

Surface representations

- Wireframe
- Hidden lines
- Shaded surface



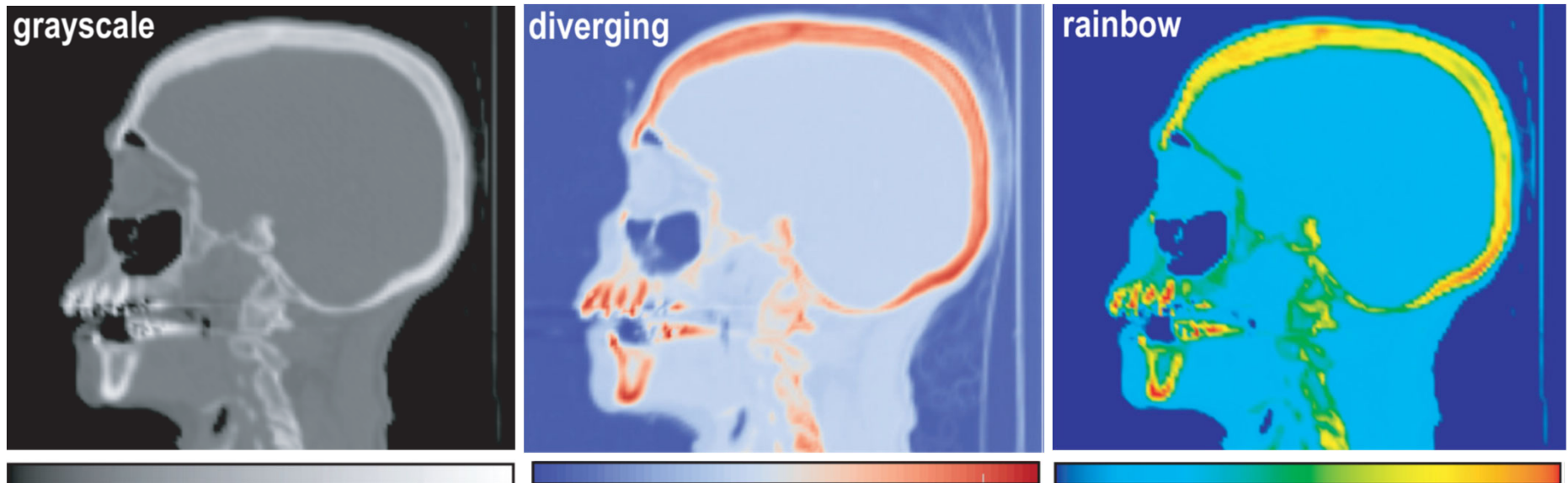
Color Mapping / Color Coding



Map scalar value to color

- Color table (e.g., array with RGB entries)
- Procedural computation; manual specification

With opacity (alpha value “A”): 1D *transfer function* (RGBA table, ...)



not recommended!

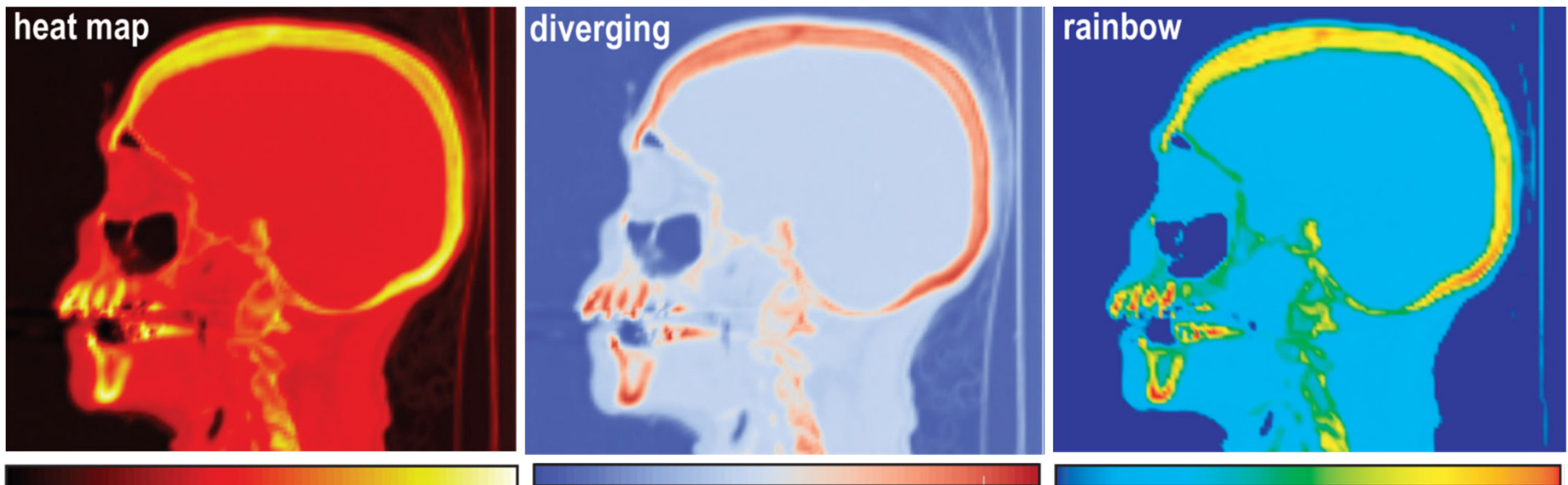
Color Mapping / Color Coding



Map scalar value to color

- Color table (e.g., array with RGB entries)
- Procedural computation; manual specification

With opacity (alpha value “A”): 1D *transfer function* (RGBA table, ...)



not recommended!

Contours



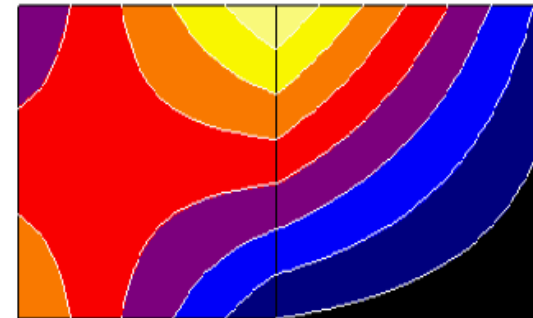
Set of points where the scalar field s has a given value c :

$$S(c) := f^{-1}(c) \quad S(c) := \{x \in \mathbb{R}^n : f(x) = c\}$$

Common contouring algorithms

- 2D: marching squares, marching triangles
- 3D: marching cubes, marching tetrahedra

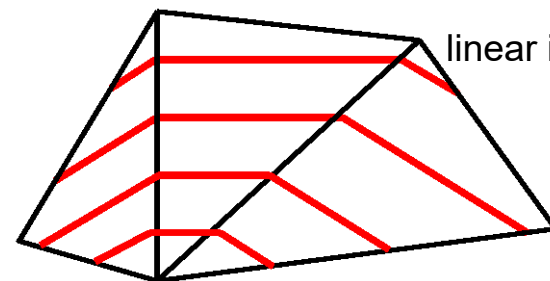
bilinear interpolation



Implicit methods

- Point-on-contour test
- Isosurface ray-casting

linear interpolation



Contours



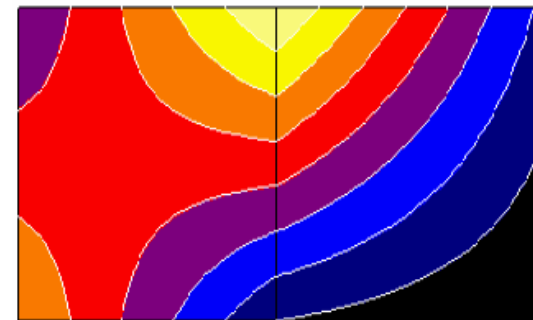
Set of points where the scalar field s has a given value c :

$$S(c) := f^{-1}(c) \quad S(c) := \{x \in \mathbb{R}^2 : f(x) = c\}$$

Common contouring algorithms

- 2D: marching squares, marching triangles
- 3D: marching cubes, marching tetrahedra

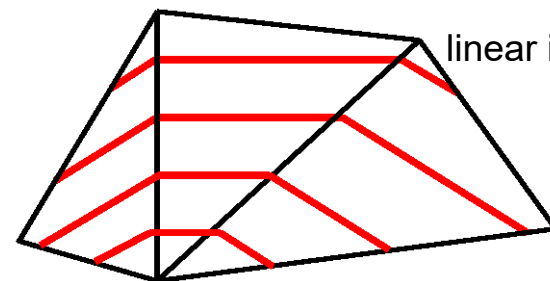
bilinear interpolation



Implicit methods

- Point-on-contour test
- Isosurface ray-casting

linear interpolation



Contours



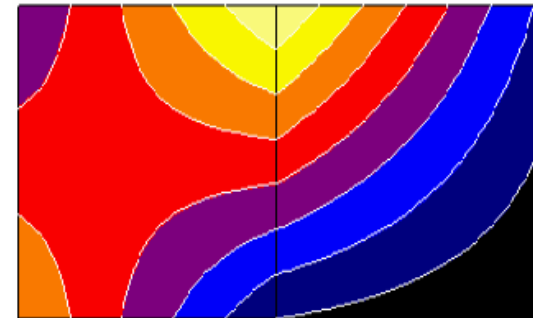
Set of points where the scalar field s has a given value c :

$$S(c) := f^{-1}(c) \quad S(c) := \{x \in \mathbb{R}^3 : f(x) = c\}$$

Common contouring algorithms

- 2D: marching squares, marching triangles
- 3D: marching cubes, marching tetrahedra

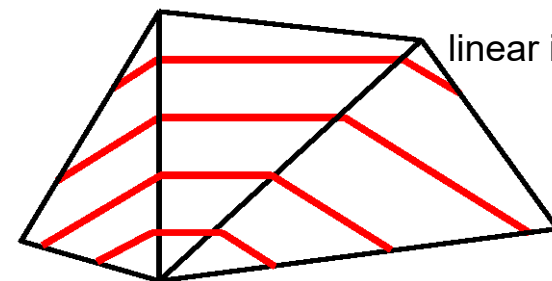
bilinear interpolation



Implicit methods

- Point-on-contour test
- Isosurface ray-casting

linear interpolation



What are contours?

Set of points where the scalar field s has a given value c :

$$S(c) := \{x \in \mathbb{R}^n : f(x) = c\}$$

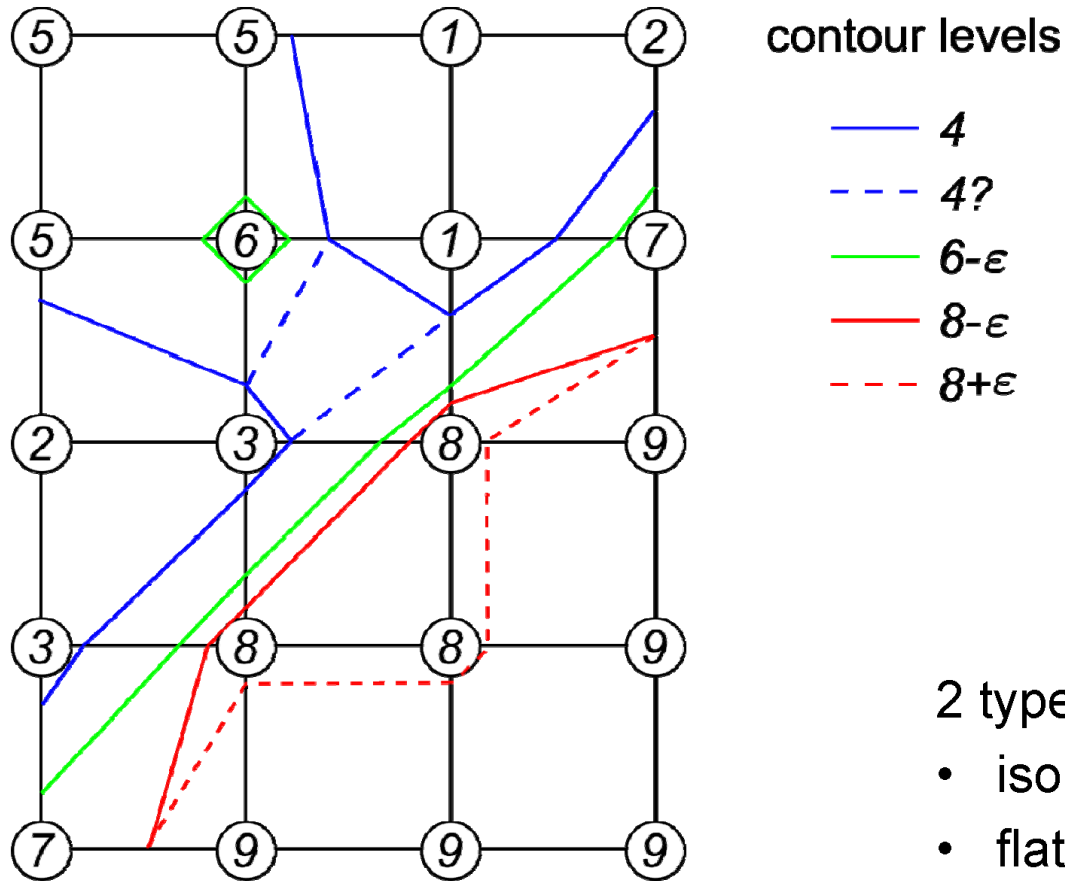
Examples in 2D:

- height contours on maps
- isobars on weather maps

Contouring algorithm:

- find intersection with grid edges
- connect points in each cell

Example



Contours in a quadrangle cell

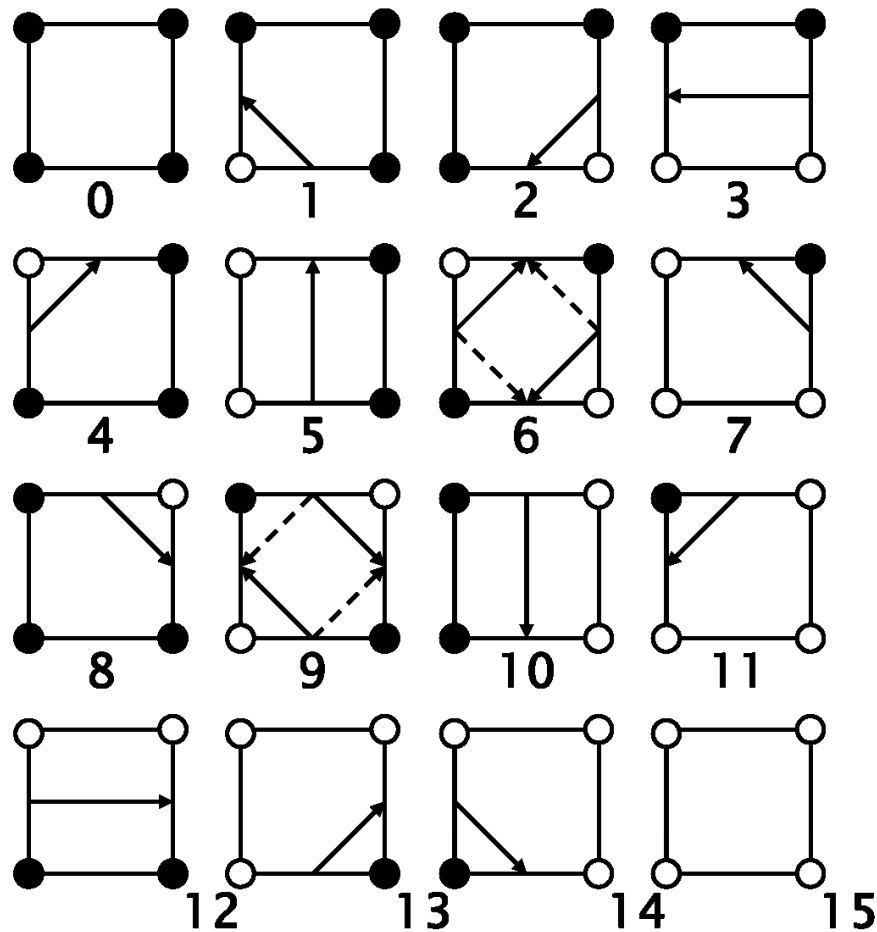
Basic contouring algorithms:

- **cell-by-cell** algorithms: simple structure, but generate disconnected segments, require post-processing
- **contour propagation** methods: more complicated, but generate connected contours

"Marching squares" algorithm (systematic cell-by-cell):

- process nodes in ccw order, denoted here as x_0, x_1, x_2, x_3
- compute at each node \mathbf{x}_i the reduced field $\tilde{f}(x_i) = f(x_i) - (c - \varepsilon)$ (which is forced to be nonzero)
- take its sign as the i^{th} bit of a 4-bit integer
- use this as an index for lookup table containing the connectivity information:

Contours in a quadrangle cell



- $\tilde{f}(x_i) < 0$
- $\tilde{f}(x_i) > 0$

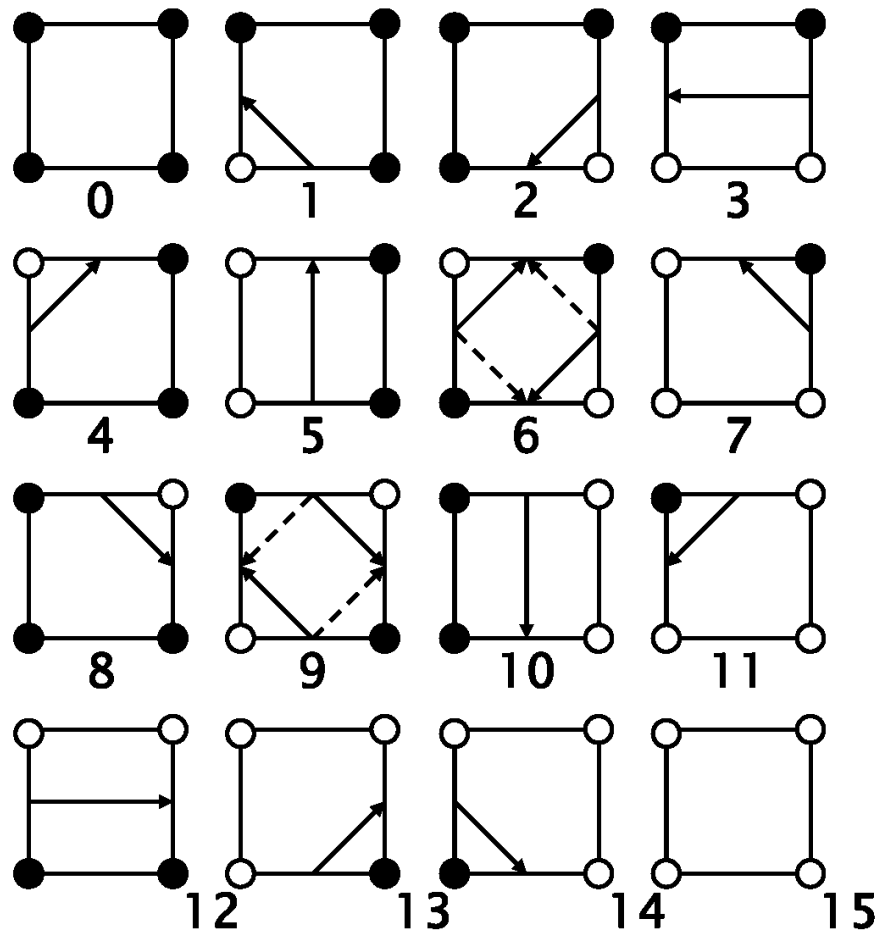
Alternating signs exist in cases 6 and 9.

Choose the solid or dashed line?

Both are possible for topological consistency.

This allows to have a fixed table of 16 cases.

Contours in a quadrangle cell



- $f(x_i) < c$
- $f(x_i) \geq c$

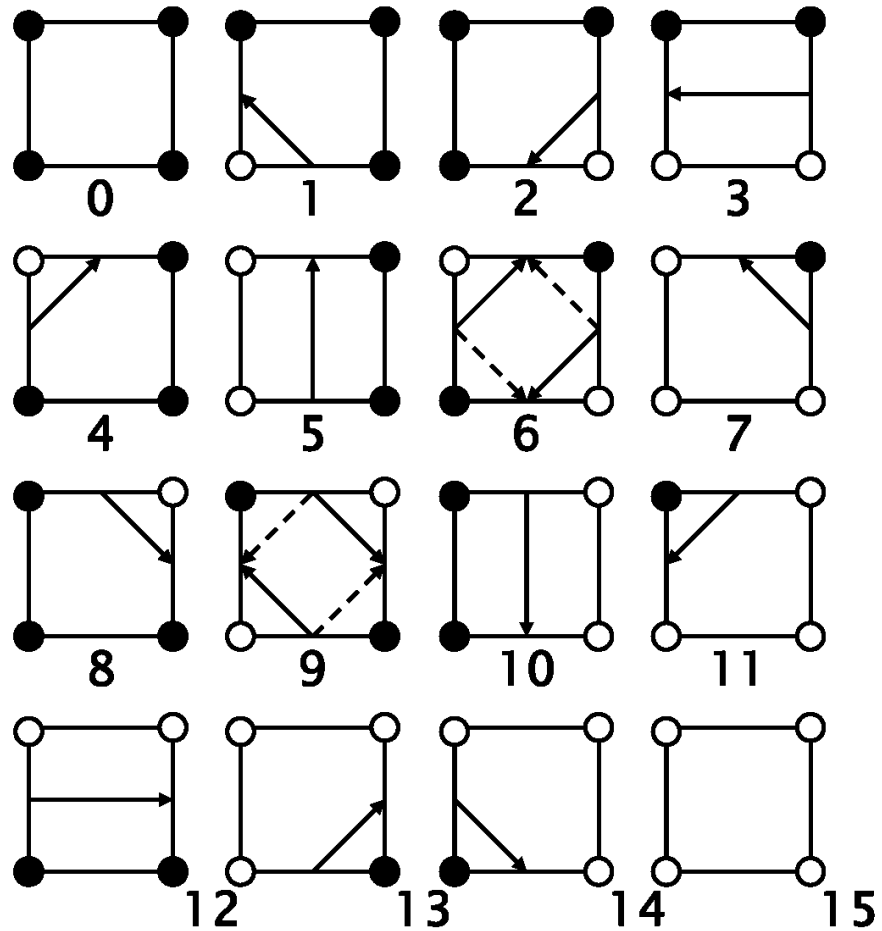
Alternating signs exist in cases 6 and 9.

Choose the solid or dashed line?

Both are possible for topological consistency.

This allows to have a fixed table of 16 cases.

Contours in a quadrangle cell



- $f(x_i) \leq c$
- $f(x_i) > c$

Alternating signs exist in cases 6 and 9.

Choose the solid or dashed line?

Both are possible for topological consistency.

This allows to have a fixed table of 16 cases.

Orientability (1-manifold embedded in 2D)

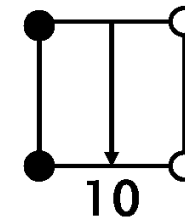
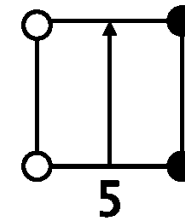


Orientability of 1-manifold:

Possible to assign consistent left/right orientation

Iso-contours

- Consistent side for scalar values...
 - greater than iso-value (e.g., *left* side)
 - less than iso-value (e.g., *right* side)
- Use consistent ordering of vertices (e.g., larger vertex index is “tip” of arrow; if (0,1) points “up”, “left” is left, ...)



not orientable



Möbius strip
(only one side!)

● $\tilde{f}(x_i) < 0$

○ $\tilde{f}(x_i) > 0$

Orientability (2-manifold embedded in 3D)



Orientability of 2-manifold:

Possible to assign consistent normal vector orientation

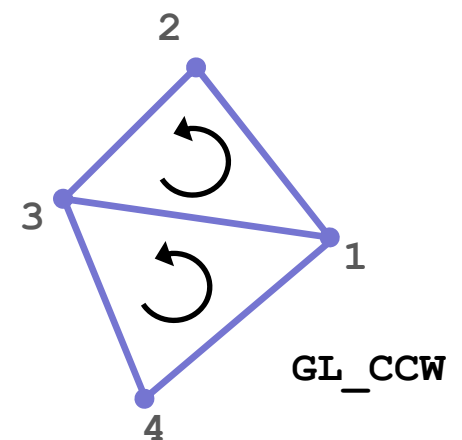
Triangle meshes

- Edges
 - Consistent ordering of vertices: CCW (counter-clockwise) or CW (clockwise) (e.g., (3,1,2) on one side of edge, (1,3,4) on the other side)
- Triangles
 - Consistent front side vs. back side
 - Normal vector; or ordering of vertices (CCW/CW)
 - See also: “right-hand rule”

not orientable



Möbius strip
(only one side!)



Topological consistency

To avoid degeneracies, use **symbolic perturbations**:

If level c is found as a node value, set the level to $c - \varepsilon$ where ε is a symbolic infinitesimal.

Then:

- contours intersect edges at some (possibly infinitesimal) distance from end points
- flat regions can be visualized by pair of contours at $c - \varepsilon$ and $c + \varepsilon$
- contours are **topologically consistent**, meaning:



Contours are **closed, orientable, nonintersecting lines**.

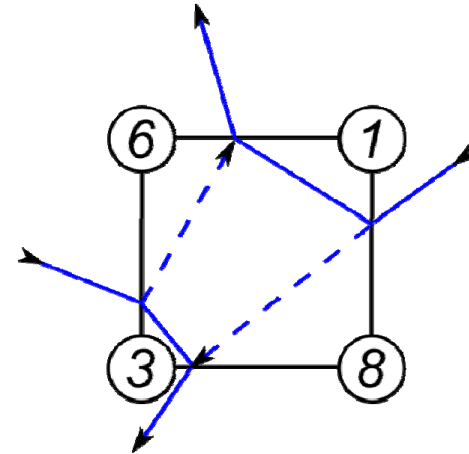
(except where the
boundary is hit)

Ambiguities of contours

What is the **correct** contour of $c=4$?

Two possibilities, both are orientable:

- connect high values 
- connect low values 



Answer: correctness depends on interior values of $f(x)$.

But: different interpolation schemes are possible.

Better question: What is the correct contour with respect to bilinear interpolation?

Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama