King Abdullah University of Science and Technology KAUST

CS 247 – Scientific Visualization
Lecture 7: Scalar Fields, Pt. 3

Markus Hadwiger, KAUST

Reading Assignment #4 (until Feb 21)

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
  [> 17,700 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: Feb 21

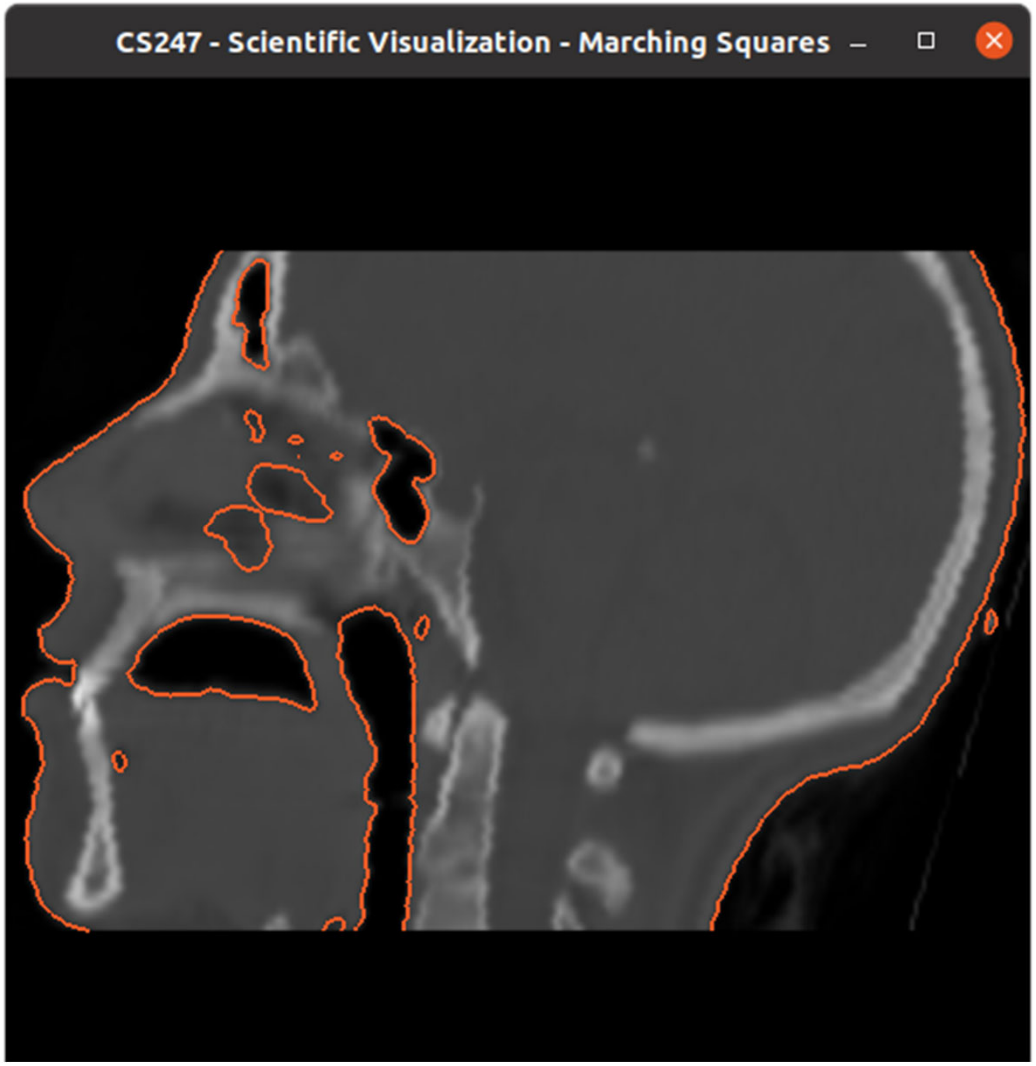
Organization

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

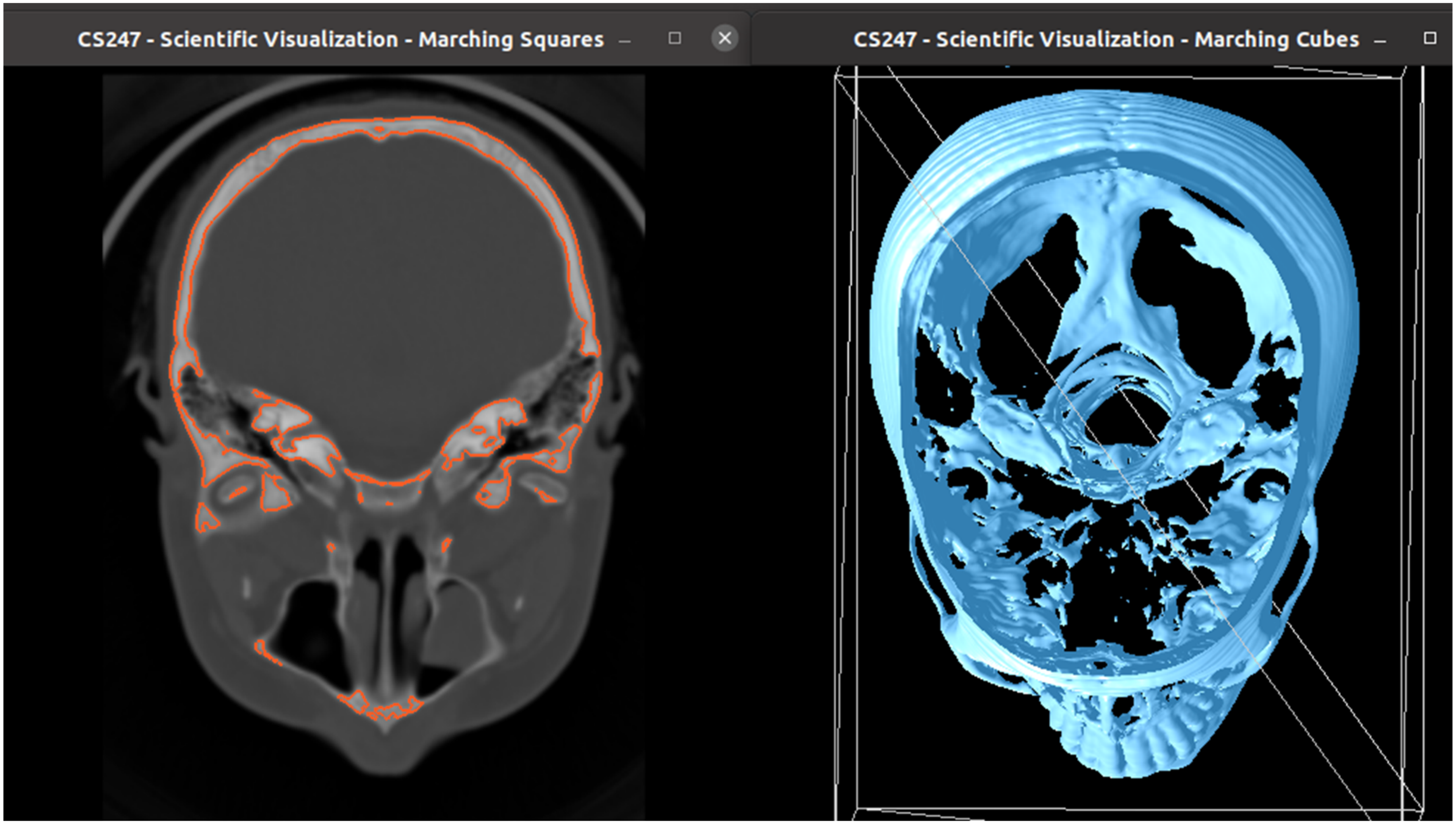
Content of questions

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

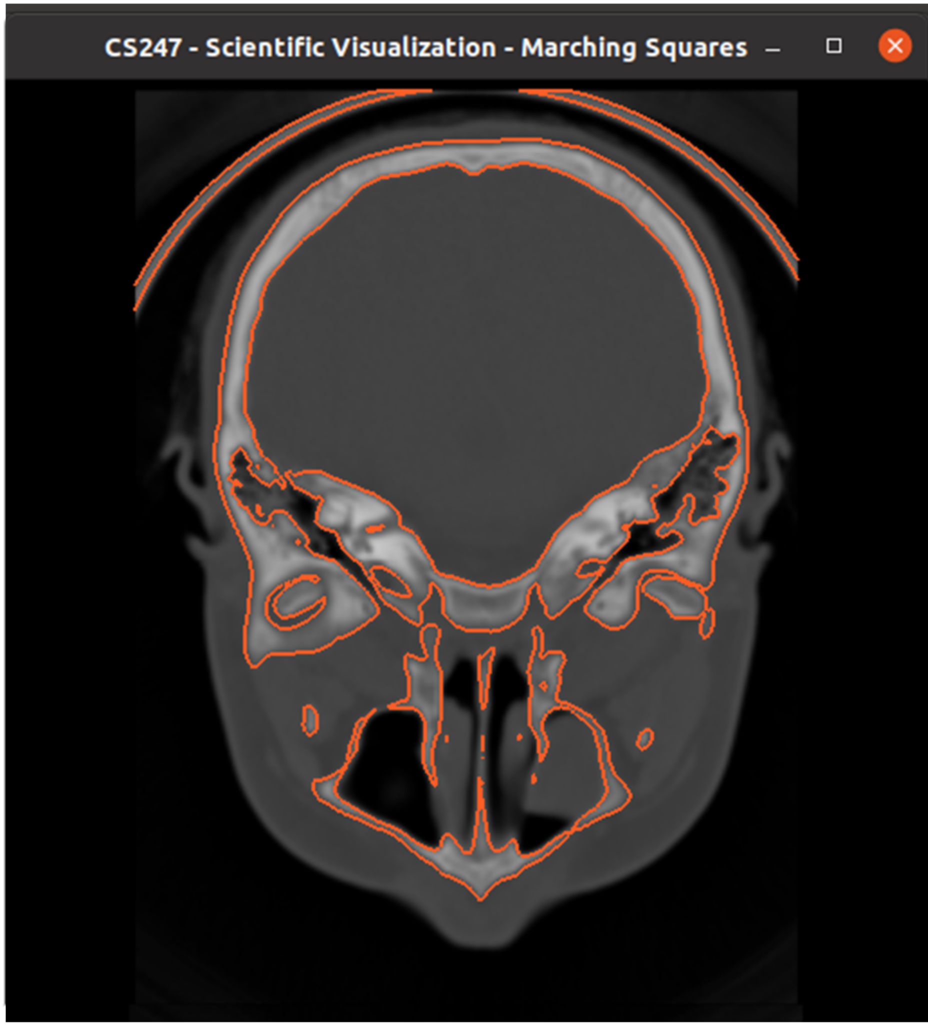
Programming Assignment 2

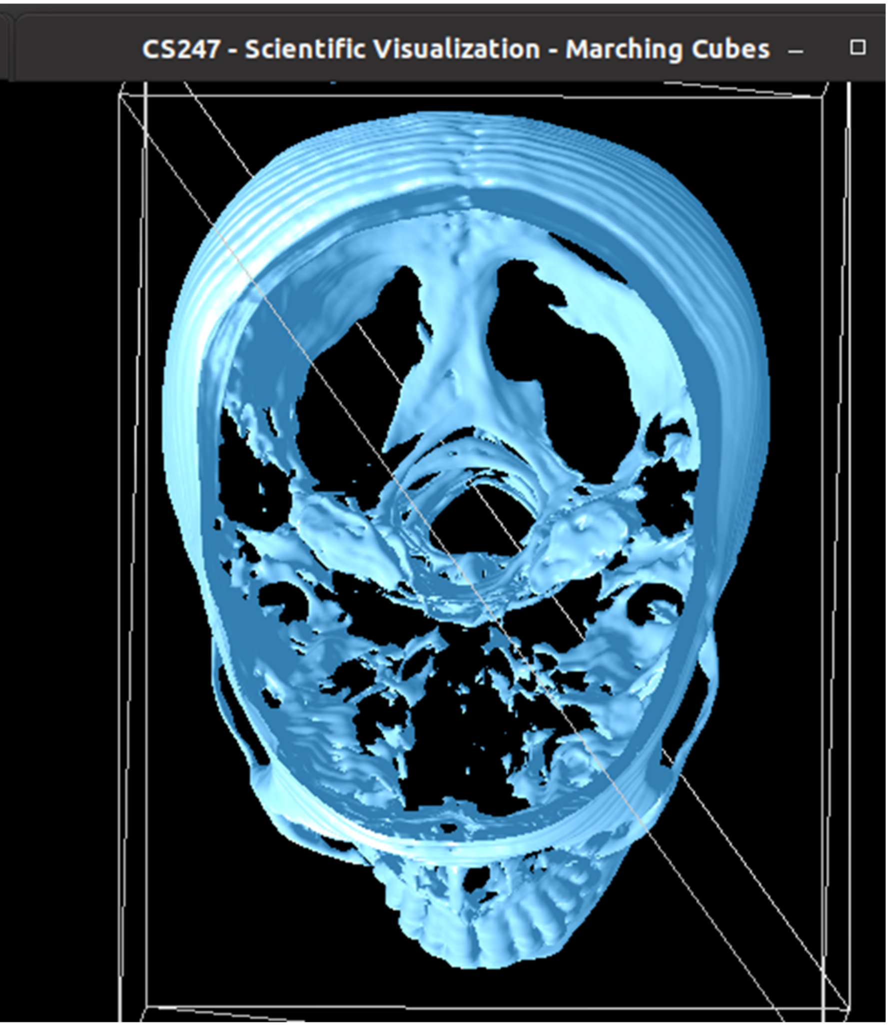


Programming Assignment 2 + 3

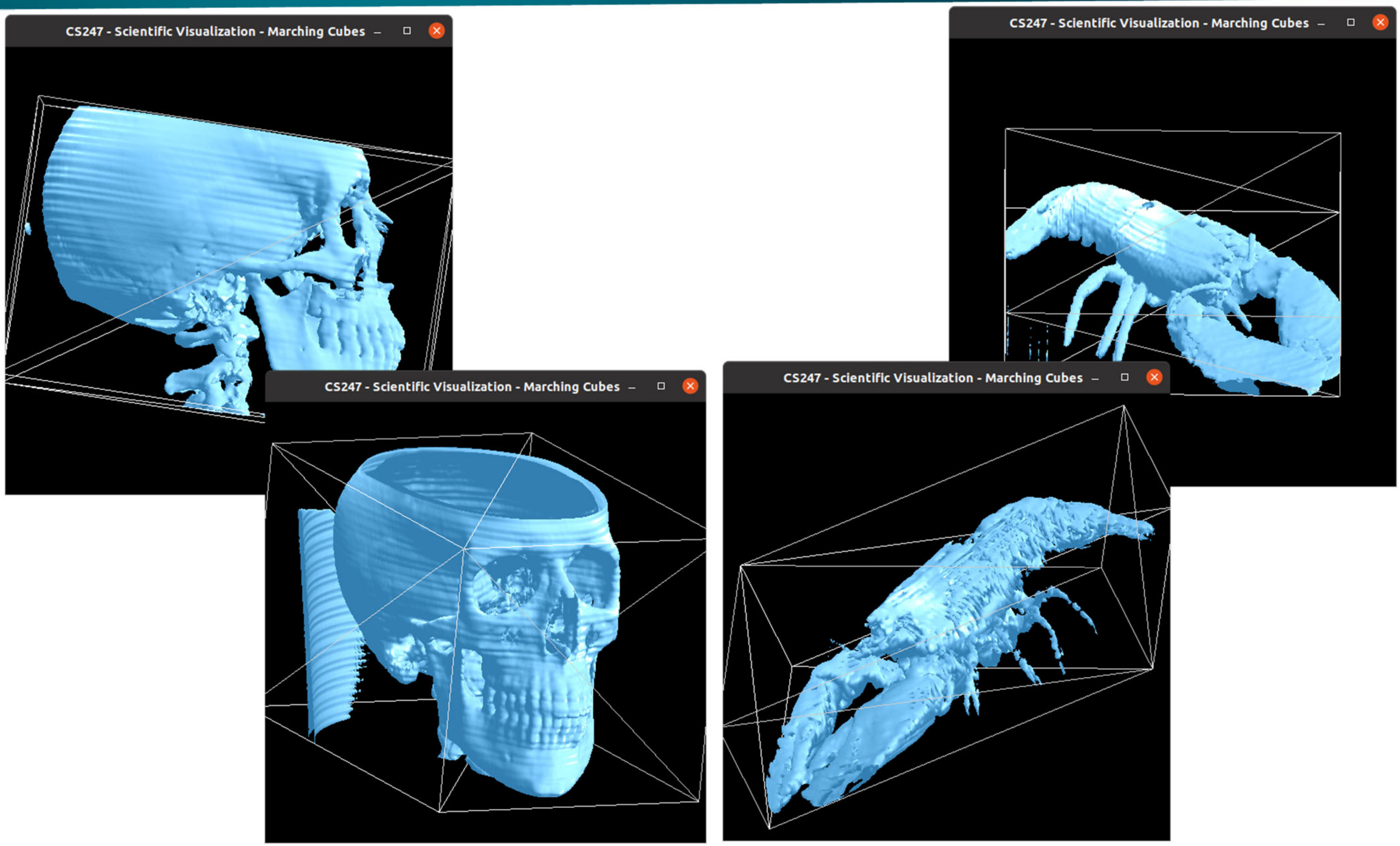


Programming Assignment 2 + 3





Programming Assignment 3



Scalar Fields

What are contours?

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

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

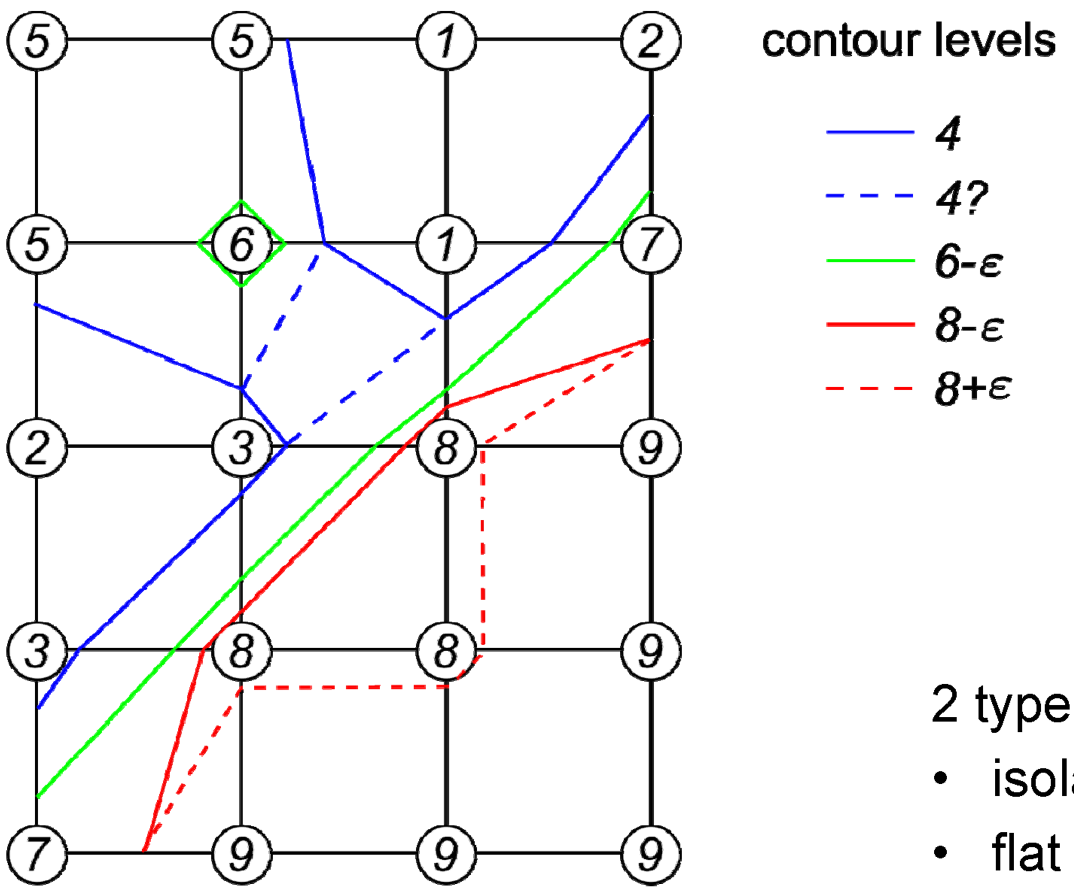
Examples in 2D:

- height contours on maps
- isobars on weather maps

Contouring algorithm:

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

Example



2 types of degeneracies:

- isolated points (c=6)
- flat regions (c=8)

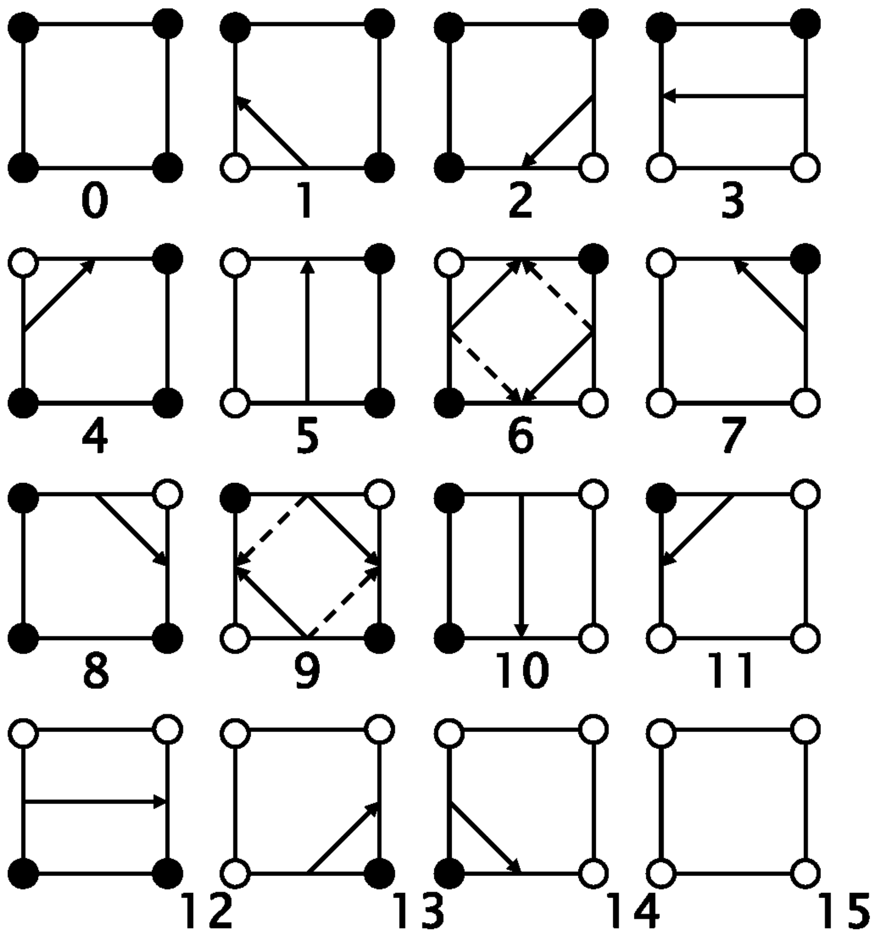
Basic contouring algorithms:

- 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



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

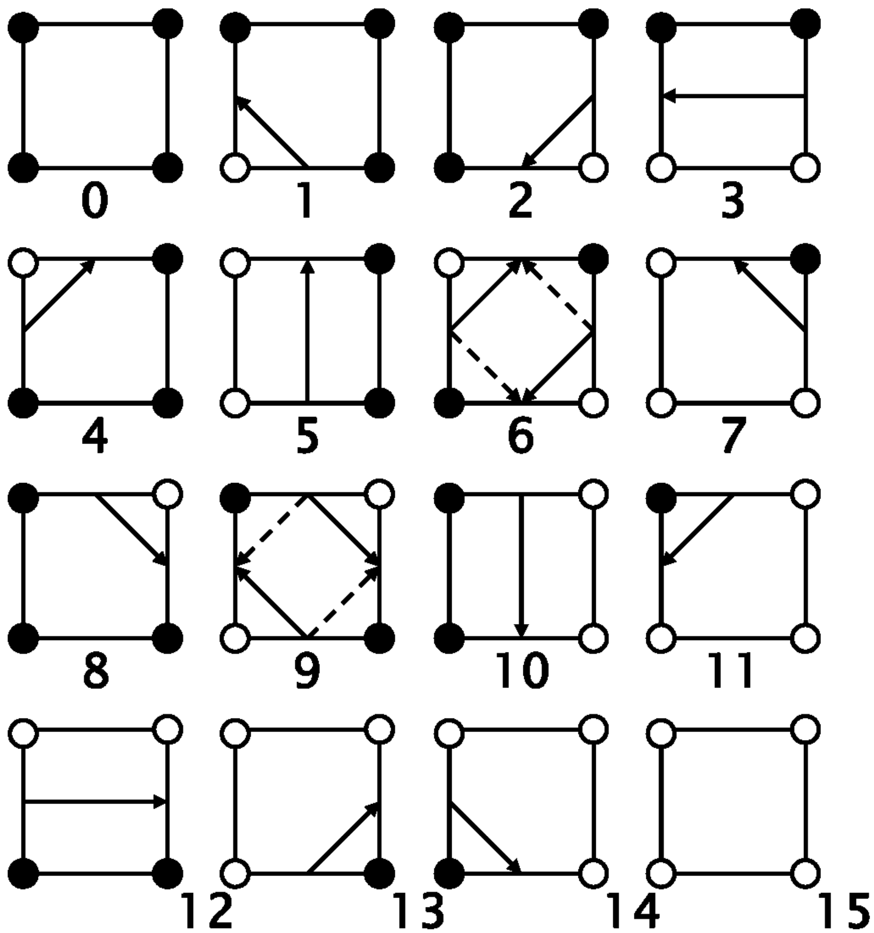
Alternating signs exist in cases 6 and 9.

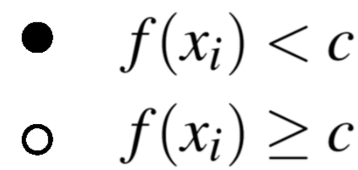
Choose the solid or dashed line?

Both are possible for topological consistency.

This allows to have a fixed table of 16 cases.

Contours in a quadrangle cell





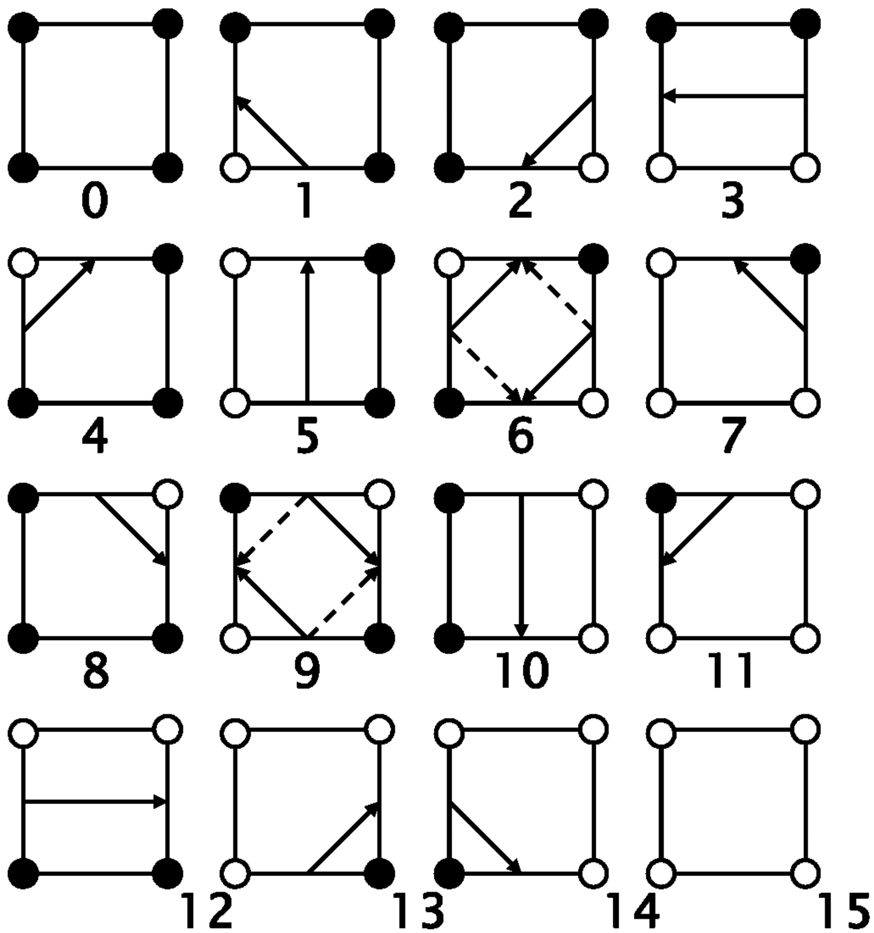
Alternating signs exist in cases 6 and 9.

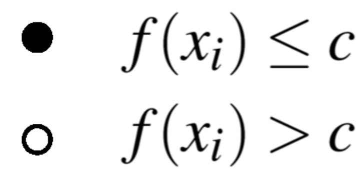
Choose the solid or dashed line?

Both are possible for topological consistency.

This allows to have a fixed table of 16 cases.

Contours in a quadrangle cell





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:

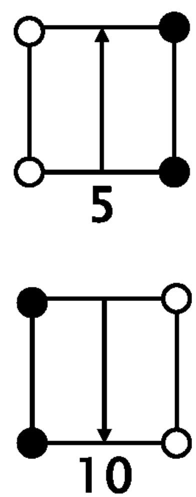
Possible to assign consistent left/right orientation

not orientable

Moebius strip
(only one side!)

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, …)

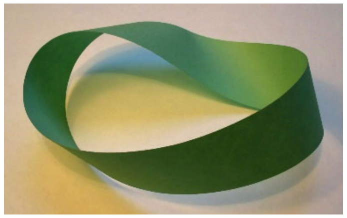


● $\tilde{f}(x_i) < 0$

○ $\tilde{f}(x_i) > 0$

Orientability (2-manifold embedded in 3D)

not orientable



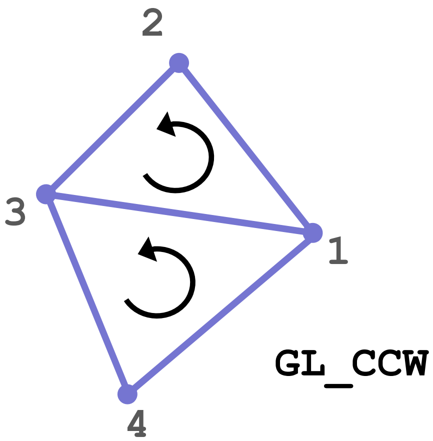
Moebius strip
(only one side!)

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”



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.

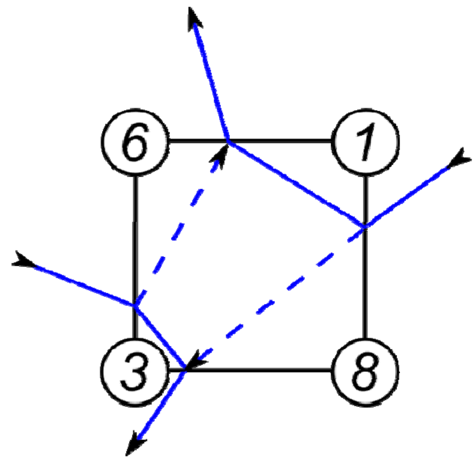
(except where the boundary is hit)

Ambiguities of contours

What is the correct contour of c=4?

Two possibilities, both are orientable:

- connect high values
- connect low values



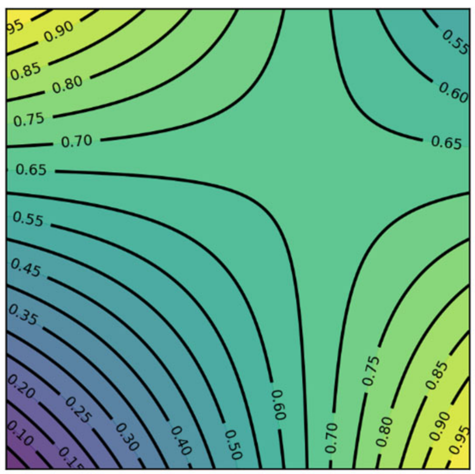
Answer: correctness depends on interior values of $f(x)$.

But: different interpolation schemes are possible.

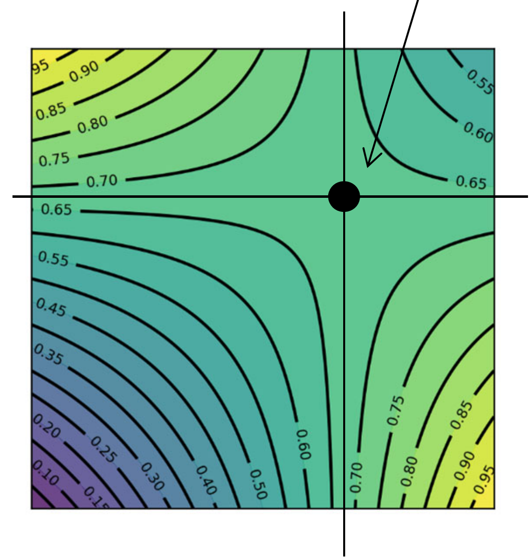
Better question: What is the correct contour with respect to bilinear interpolation?

Bi-Linear Interpolation: Critical Points

Critical points are where the gradient vanishes (i.e., is the zero vector)



critical point
(saddle point)



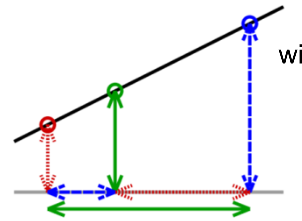
here, the critical
value is 2/3=0.666…

"Asymptotic decider": resolve ambiguous configurations (6 and 9) by
comparing specific iso-value with critical value (scalar value at critical point)

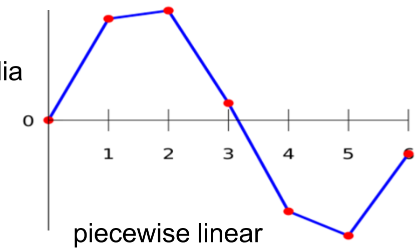
Linear Interpolation / Convex Combinations

Linear interpolation in 1D:

$$f(\alpha) = (1-\alpha)v_1 + \alpha v_2$$



wikipedia



Line embedded in 2D (linear interpolation of vertex coordinates/attributes):

$$f(\alpha_1, \alpha_2) = \alpha_1 v_1 + \alpha_2 v_2 \qquad f(\alpha) = v_1 + \alpha(v_2 - v_1)$$

$$\alpha_1 + \alpha_2 = 1 \qquad \alpha = \alpha_2$$

Line segment: $\alpha_1, \alpha_2 \geq 0$ (→ convex combination)

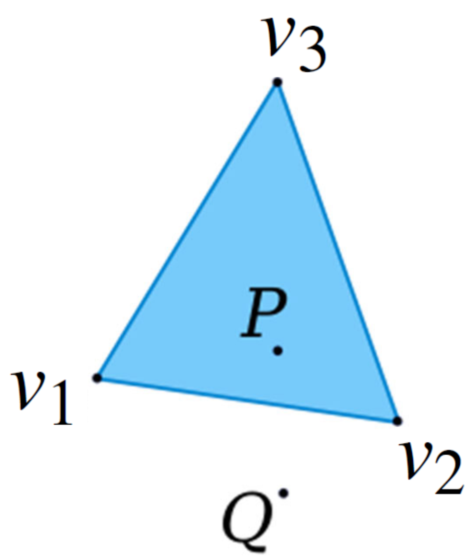
Compare to line parameterization
with parameter t:

$$v(t) = v_1 + t(v_2 - v_1)$$

Linear Interpolation / Convex Combinations

Linear combination (n-dim. space):

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i$$



Affine combination: Restrict to $(n-1)$-dim. subspace:

$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = \sum_{i=1}^{n} \alpha_i = 1$$

Convex combination: $\alpha_i \geq 0$

(restrict to simplex in subspace)

Linear Interpolation / Convex Combinations

The weights α_i are the n normalized ***barycentric*** coordinates

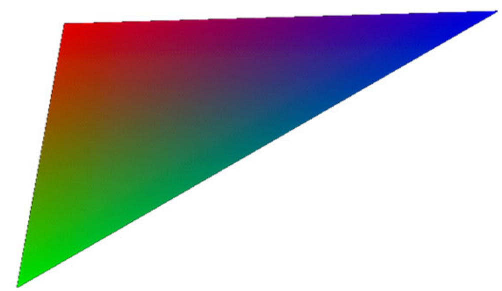
→ linear attribute interpolation in simplex

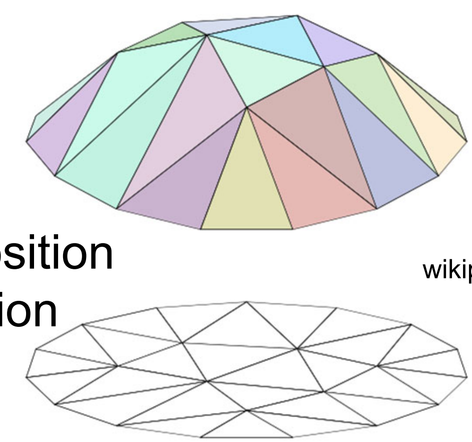
$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i$$

$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = \sum_{i=1}^{n} \alpha_i = 1$$

$$\alpha_i \geq 0$$

attribute interpolation





spatial position interpolation

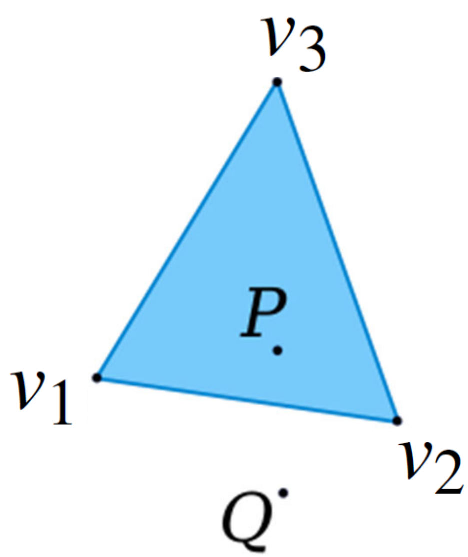
wikipedia

Linear Interpolation / Convex Combinations

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i$$

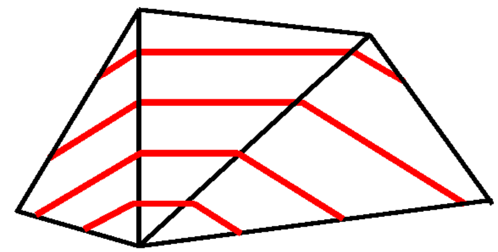
$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = \sum_{i=1}^{n} \alpha_i = 1$$

Can re-parameterize to get $(n-1)$ ***affine*** coordinates:



$$\begin{aligned} \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 &= \\ \tilde{\alpha}_1 (v_2 - v_1) + \tilde{\alpha}_2 (v_3 - v_1) + v_1 & \\ \tilde{\alpha}_1 &= \alpha_2 \\ \tilde{\alpha}_2 &= \alpha_3 \end{aligned}$$

Contours in triangle/tetrahedral cells



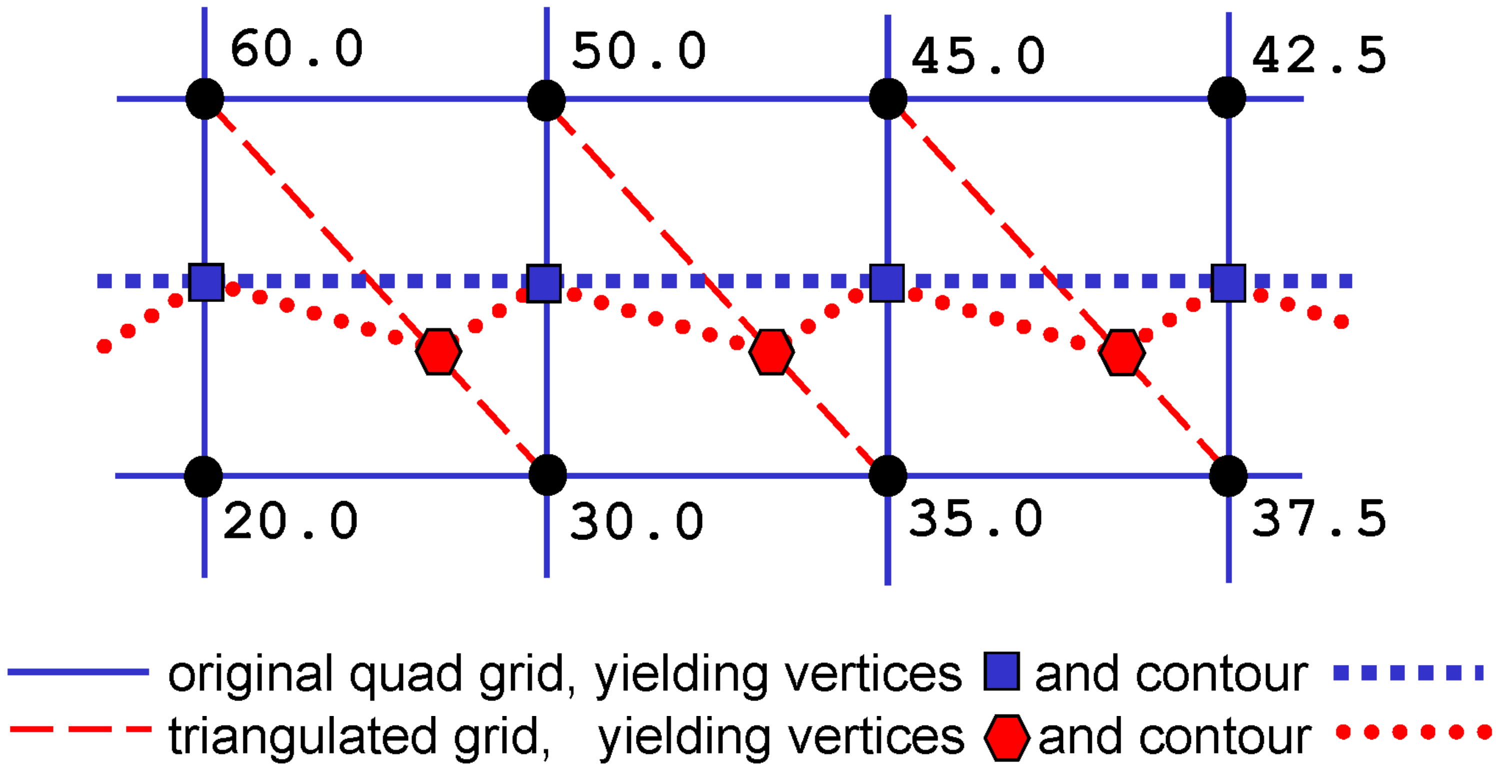
Linear interpolation of cells implies
piece-wise linear contours.

Contours are unambiguous, making "marching triangles" even simpler than "marching squares".

Question: Why not split quadrangles into two triangles (and hexahedra into five or six tetrahedra) and use marching triangles (tetrahedra)?

Answer: This can introduce periodic artifacts!

Illustrative example: Find contour at level *c*=40.0 !



From 2D to 3D (Domain)

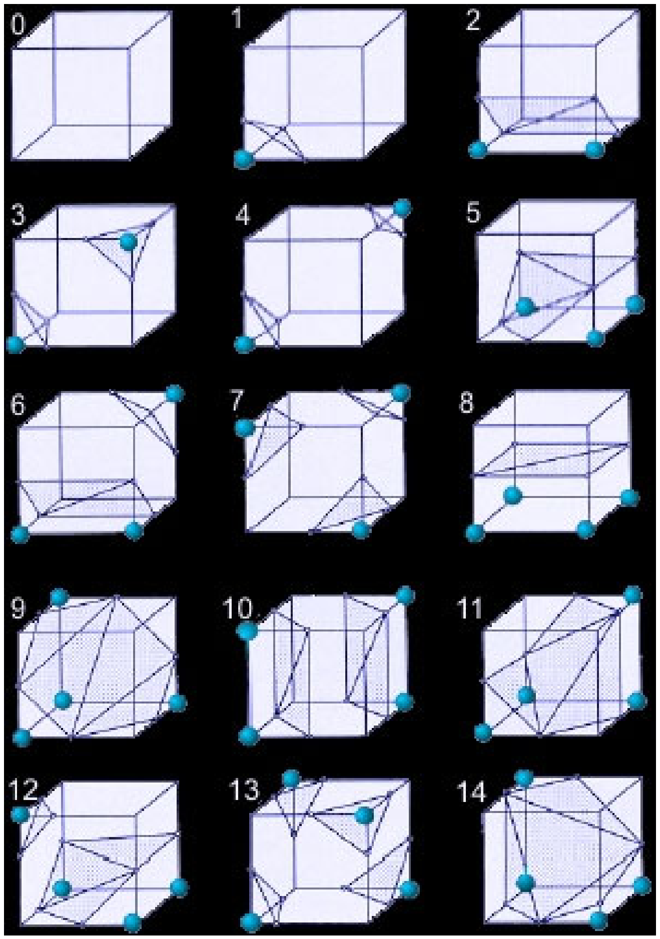
2D - Marching Squares Algorithm:

1. Locate the contour corresponding to a user-specified iso value
2. Create lines

3D - Marching Cubes Algorithm:

1. Locate the surface corresponding to a user-specified iso value
2. Create triangles
3. Calculate normals to the surface at each vertex
4. Draw shaded triangles

Marching Cubes



- For each cell, we have 8 vertices with 2 possible states each (inside or outside).
- This gives us 2^8 possible patterns = 256 cases.
- Enumerate cases to create a LUT
- Use symmetries to reduce problem from 256 to 15 cases.

Explanations

- Data Visualization book, 5.3.2
- Marching Cubes: A high resolution 3D surface construction algorithm, Lorensen & Cline, ACM SIGGRAPH 1987

The marching cubes algorithm

Contours of 3D scalar fields are known as isosurfaces.

Before 1987, isosurfaces were computed as

- contours on planar slices, followed by
- "contour stitching".

The marching cubes algorithm computes contours directly in 3D.

- Pieces of the isosurfaces are generated on a cell-by-cell basis.
- Similar to marching squares, a 8-bit number is computed from the 8 signs of $\tilde{f}(x_i)$ on the corners of a hexahedral cell.
- The isosurface piece is looked up in a table with 256 entries.

How to build up the table of 256 cases?

Lorensen and Cline (1987) exploited 3 types of symmetries:

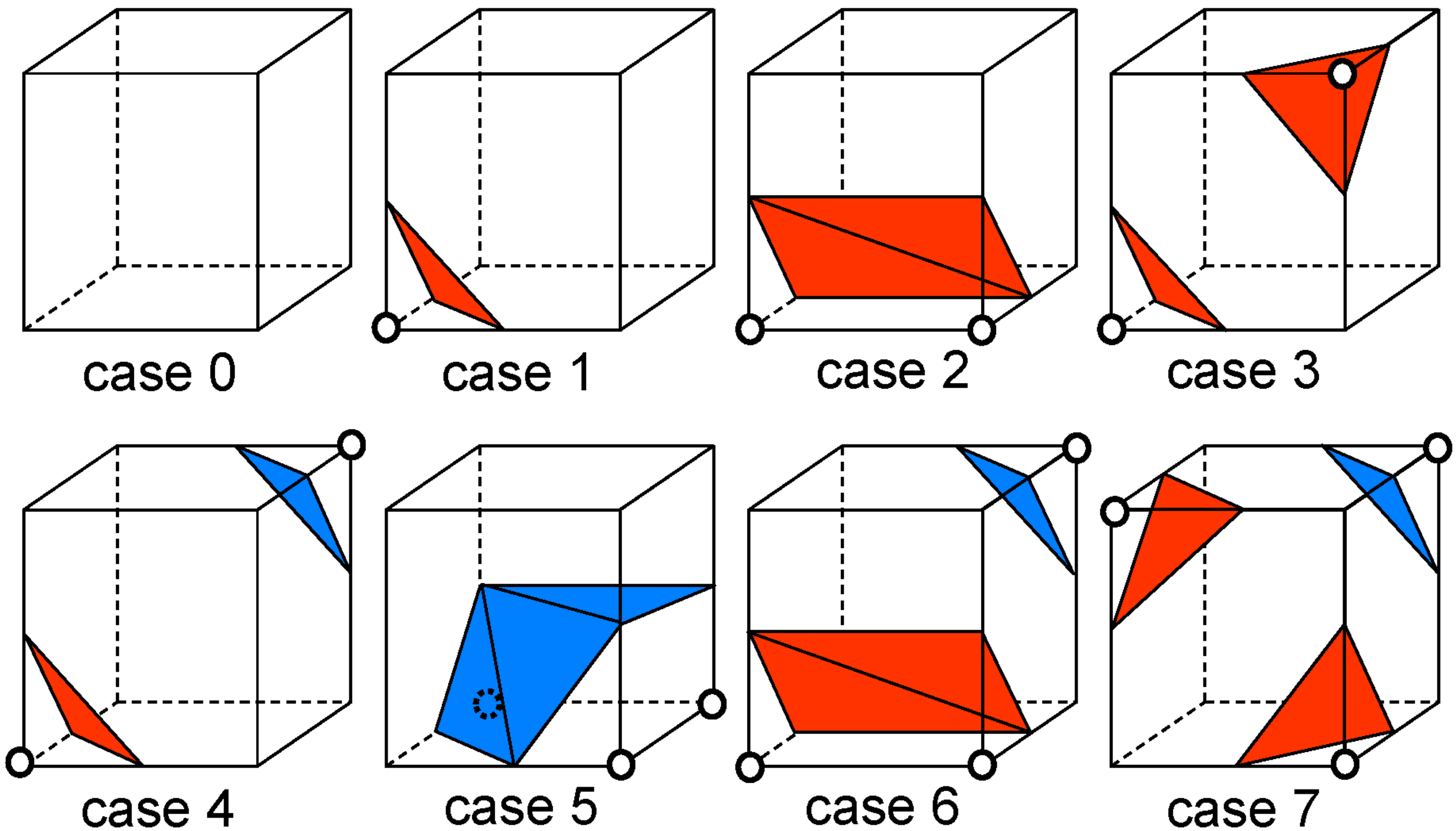
- rotational symmetries of the cube
- reflective symmetries of the cube
- sign changes of $\tilde{f}(x_i)$

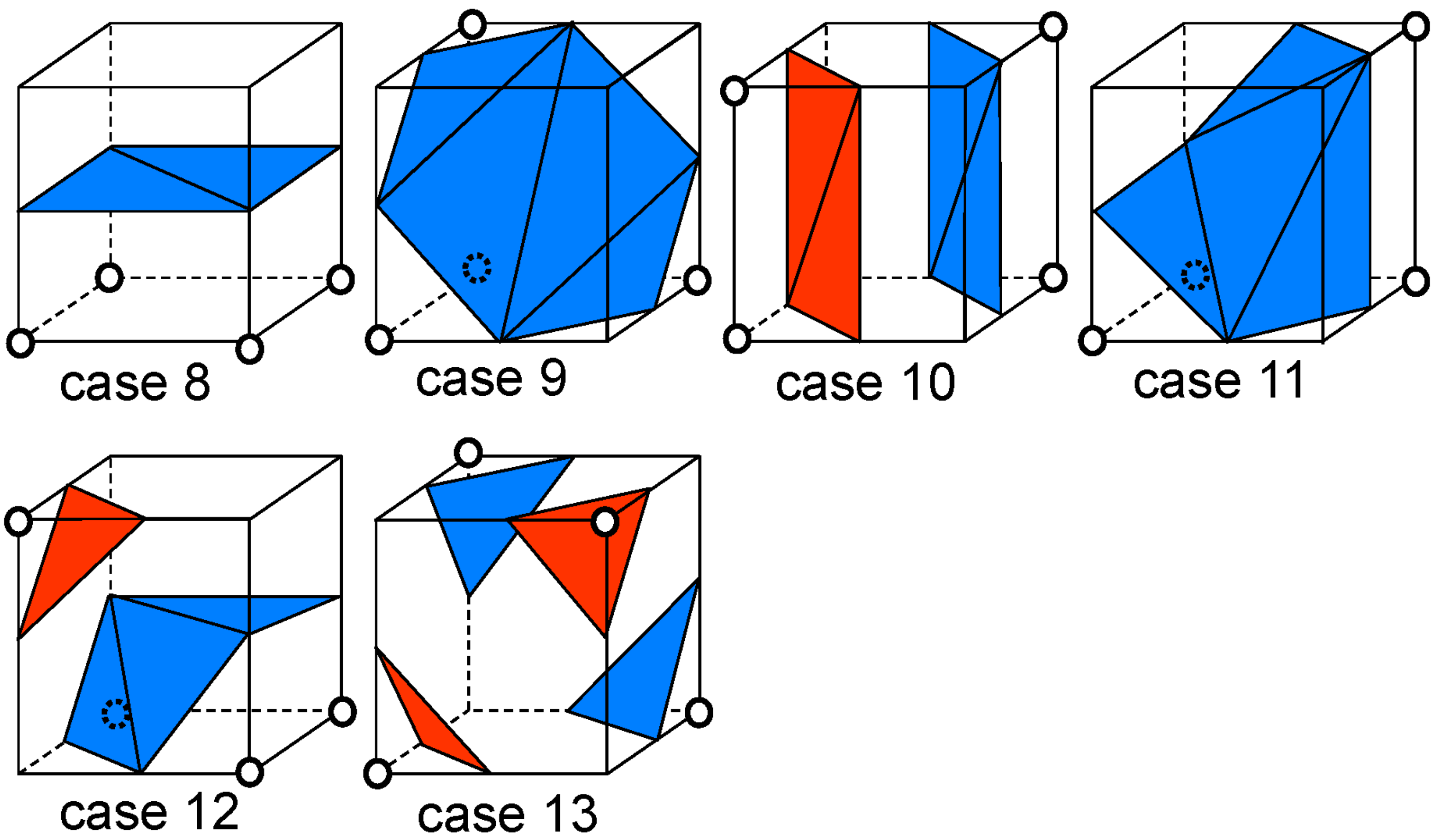
They published a reduced set of 14*) cases shown on the next slides where

- white circles indicate positive signs of $\tilde{f}(x_i)$
- the positive side of the isosurface is drawn in red, the negative side in blue.

*) plus an unnecessary "case 14" which is a symmetric image of case 11.

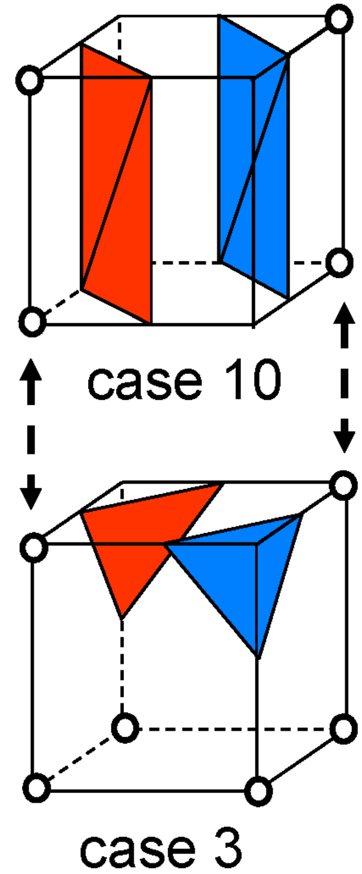
The marching cubes algorithm




case 8
case 9
case 10
case 11
case 12
case 13

Do the pieces fit together?

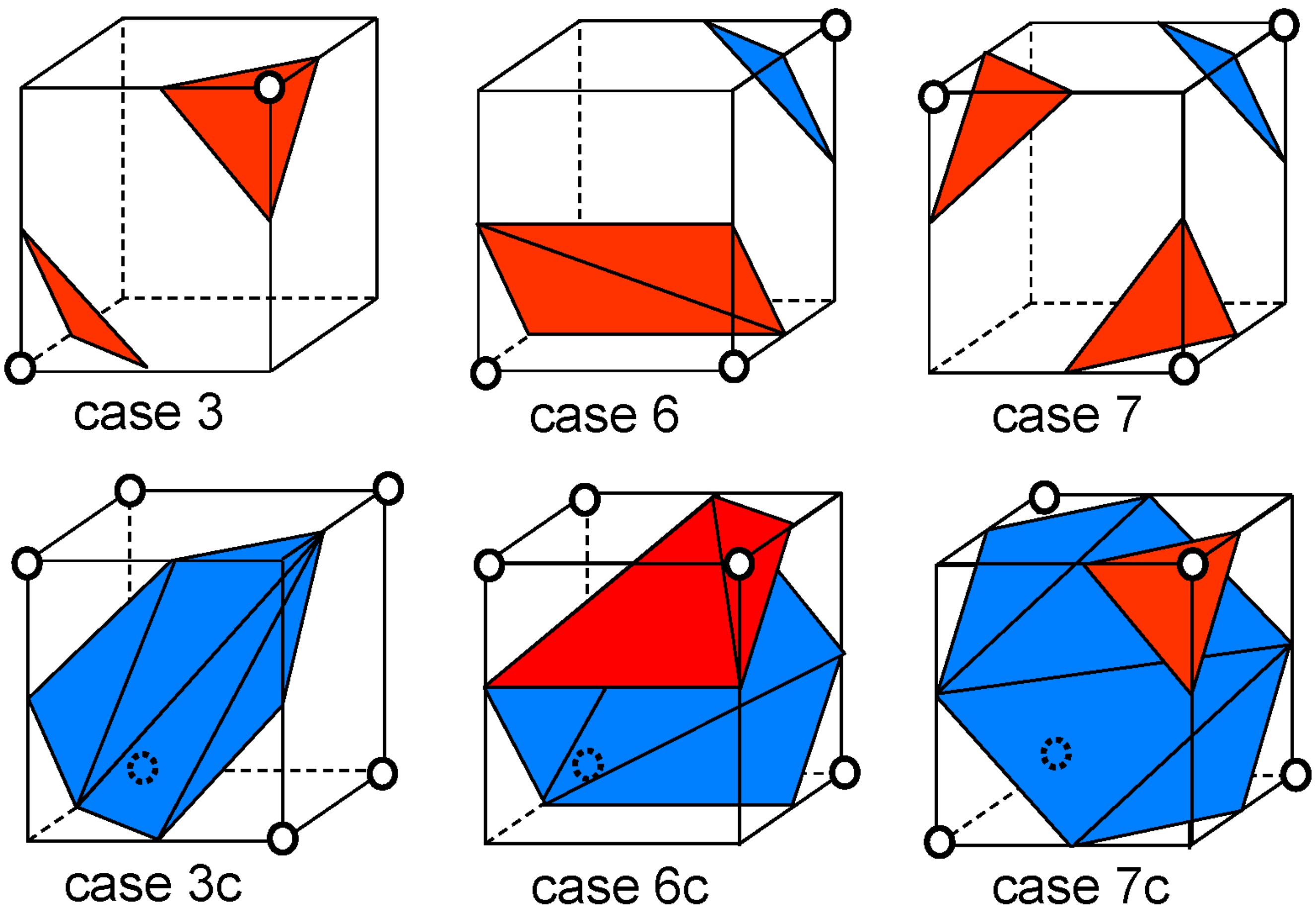
- The correct isosurfaces of the trilinear interpolant would fit (trilinear reduces to bilinear on the cell interfaces)
- but the marching cubes polygons don't necessarily fit.



Example

- case 10, on top of
- case 3 (rotated, signs changed)

have matching signs at nodes but polygons don't fit.


case 3
case 6
case 7
case 3c
case 6c
case 7c

Thank you.

Thanks for material

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