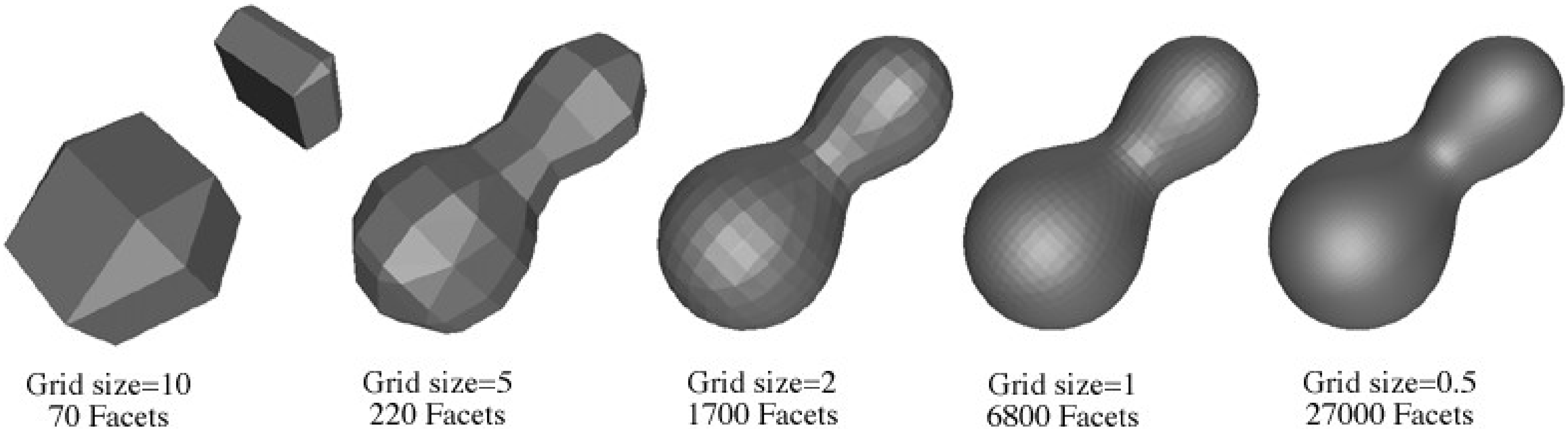
Paul Bourke

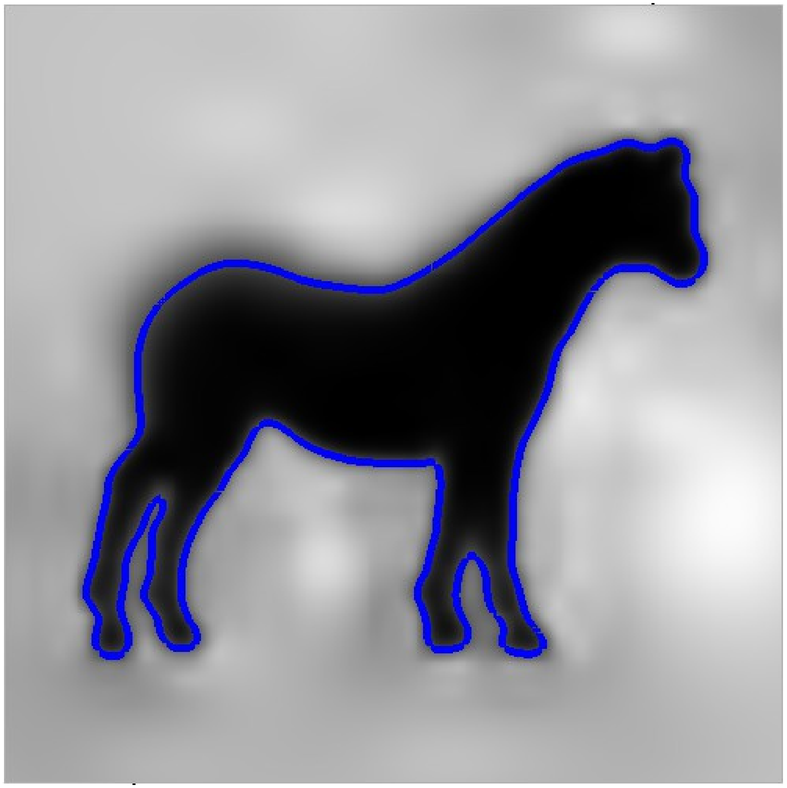


Polygonization of Implicit Surfaces

Siddhartha Chaudhuri http://www.cse.iitb.ac.in/~cs749

Recall: Final step of Poisson reconstruction

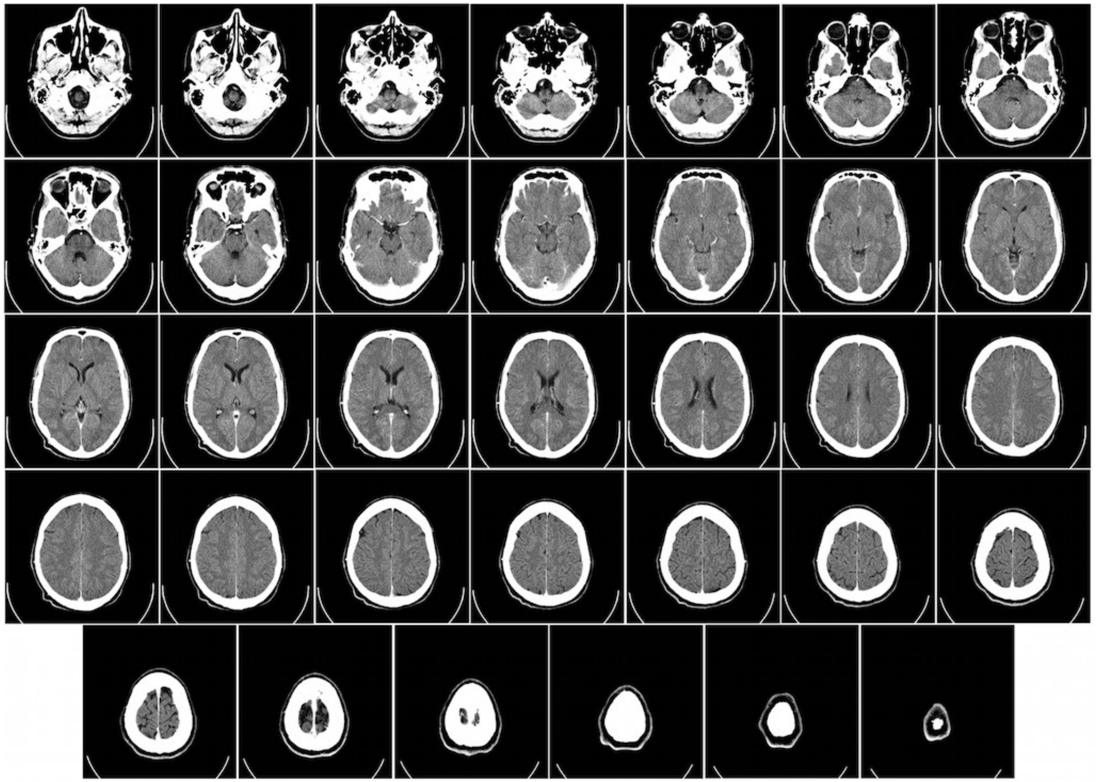
Density Function



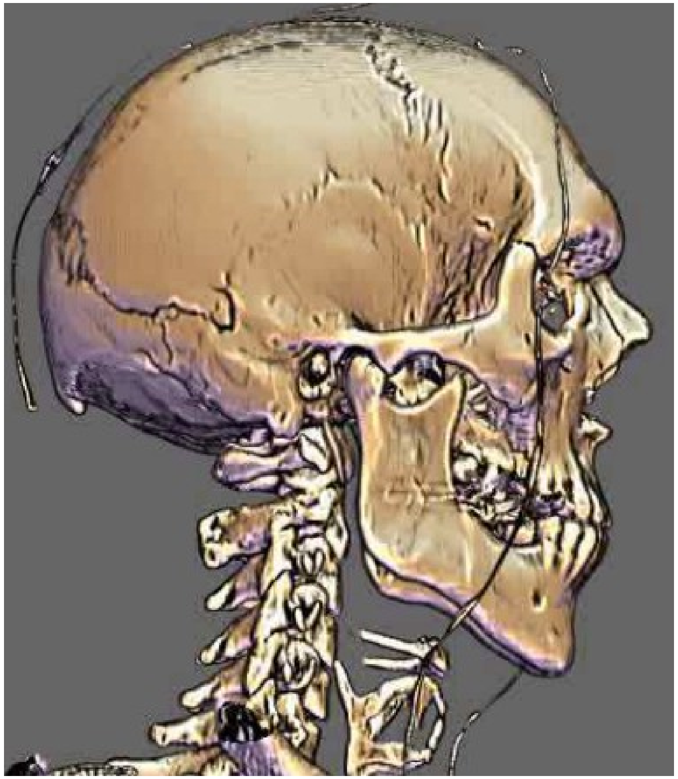
Isosurface

Kazhdan, Bolitho and Hoppe

Medical Reconstruction



Density Function from CT Scans

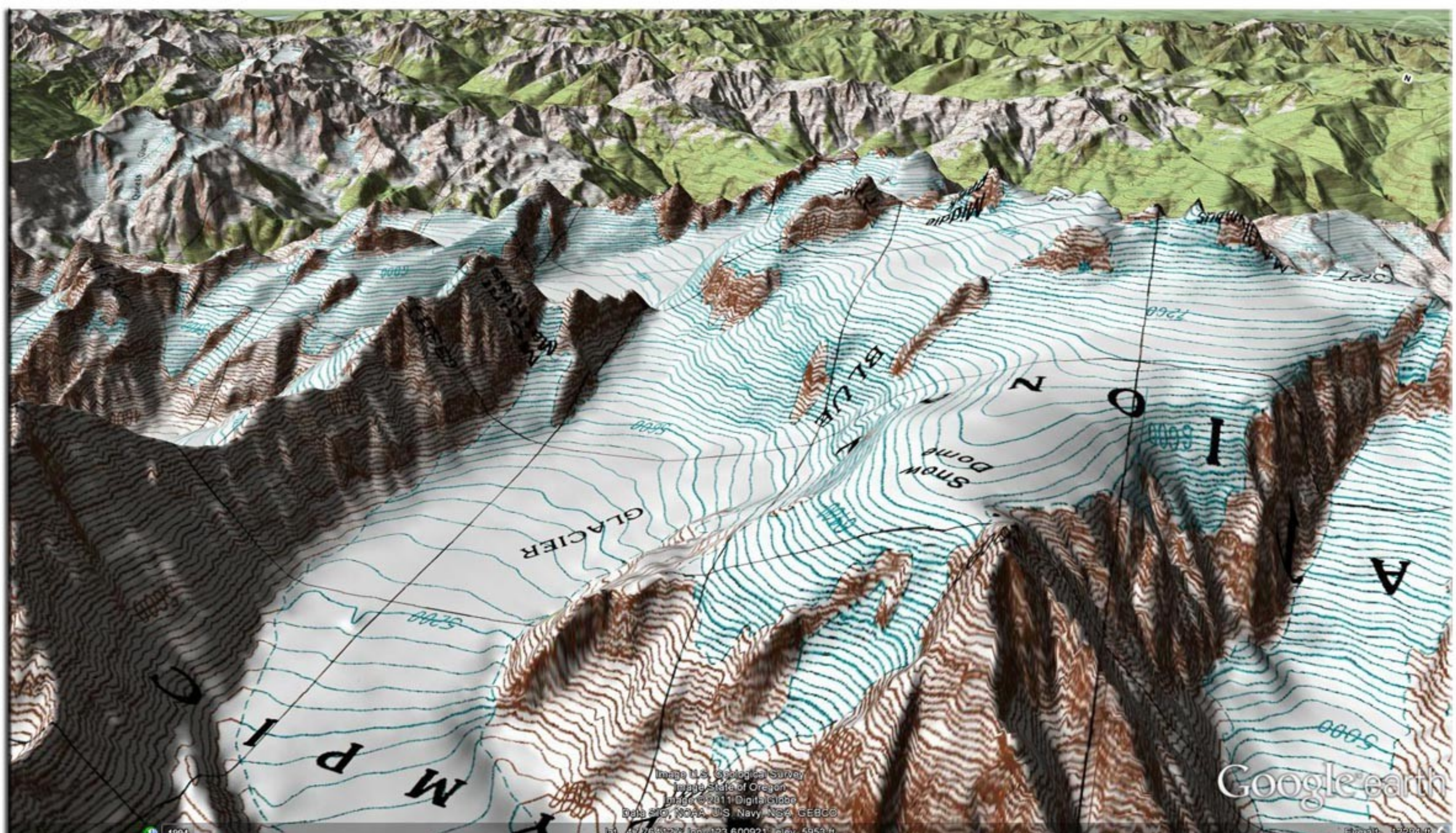


Reconstructed Skull Isosurface

Wikipedia, University of Utah

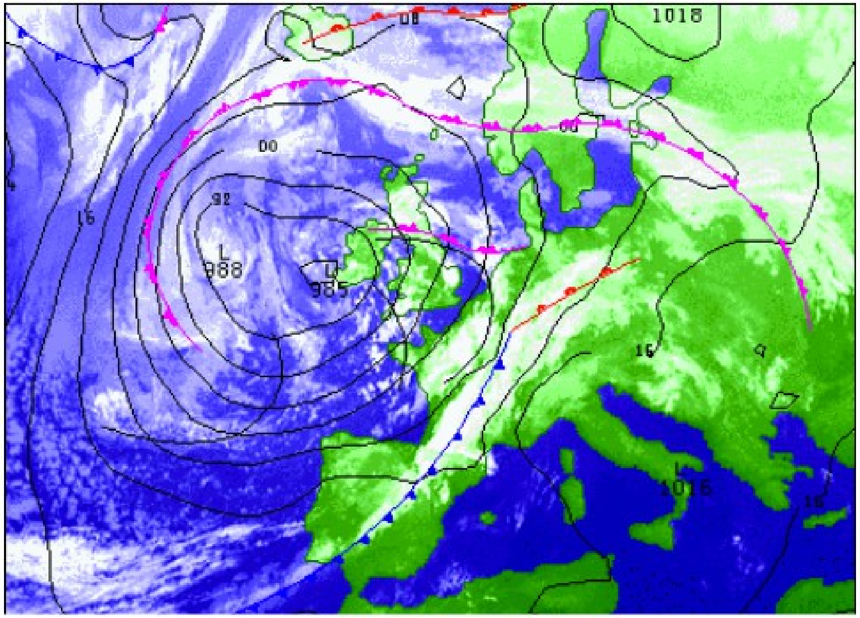
Level Set

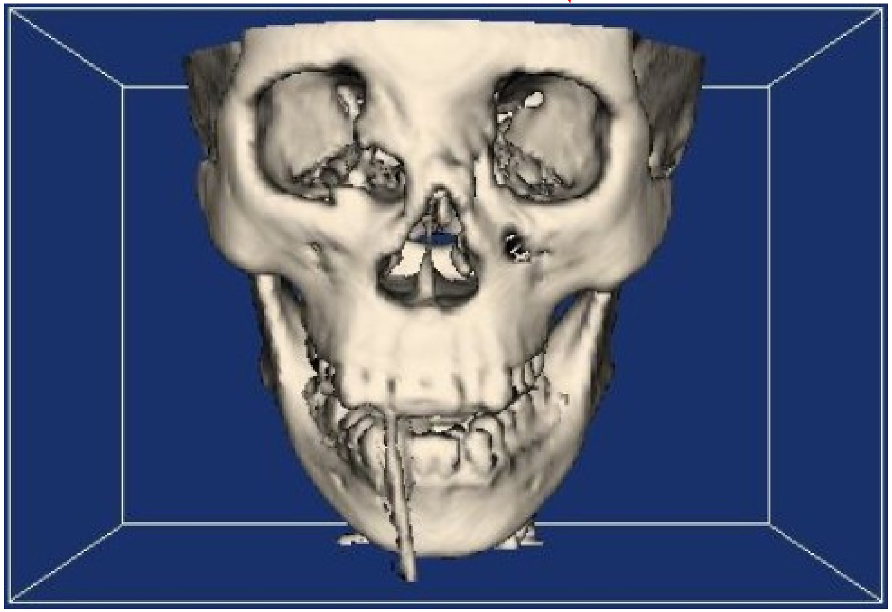
- **c-Level set**: The set of points where a function takes a constant value c



Level Set

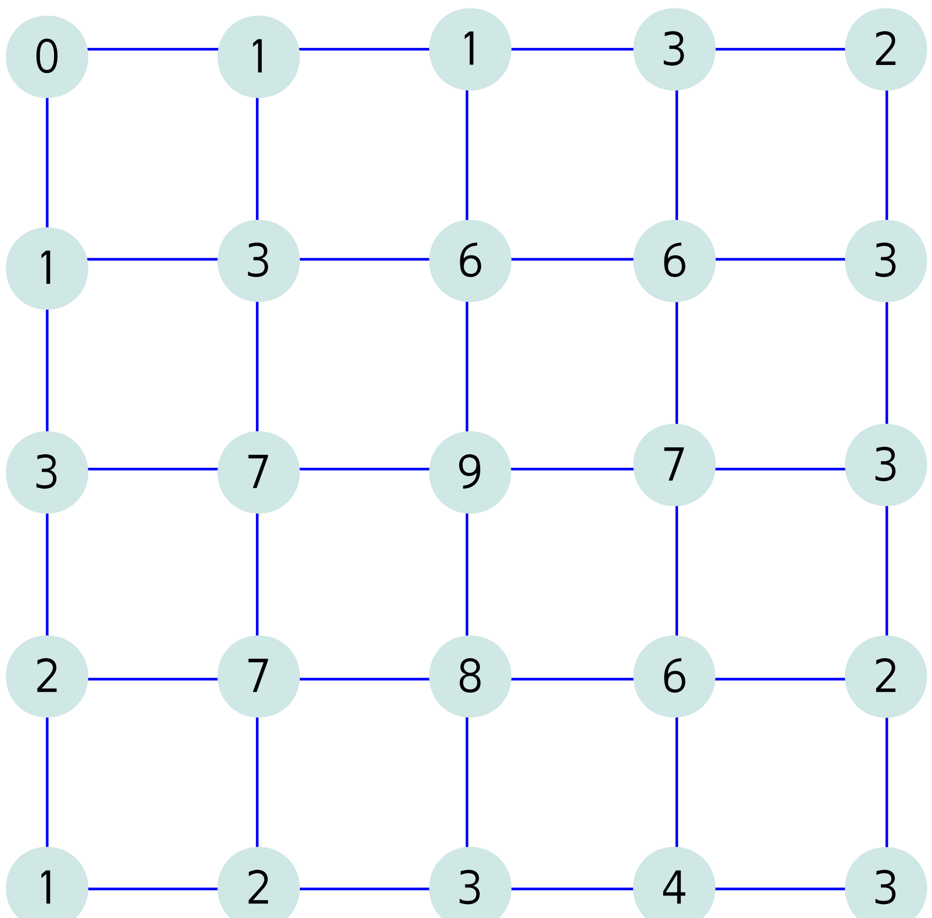
- **c-Level set**: The set of points where a function takes a constant value c
  - **Isocontour**: Level set of a 2D function
  - **Isosurface**: Level set of a 3D function





Isocontours

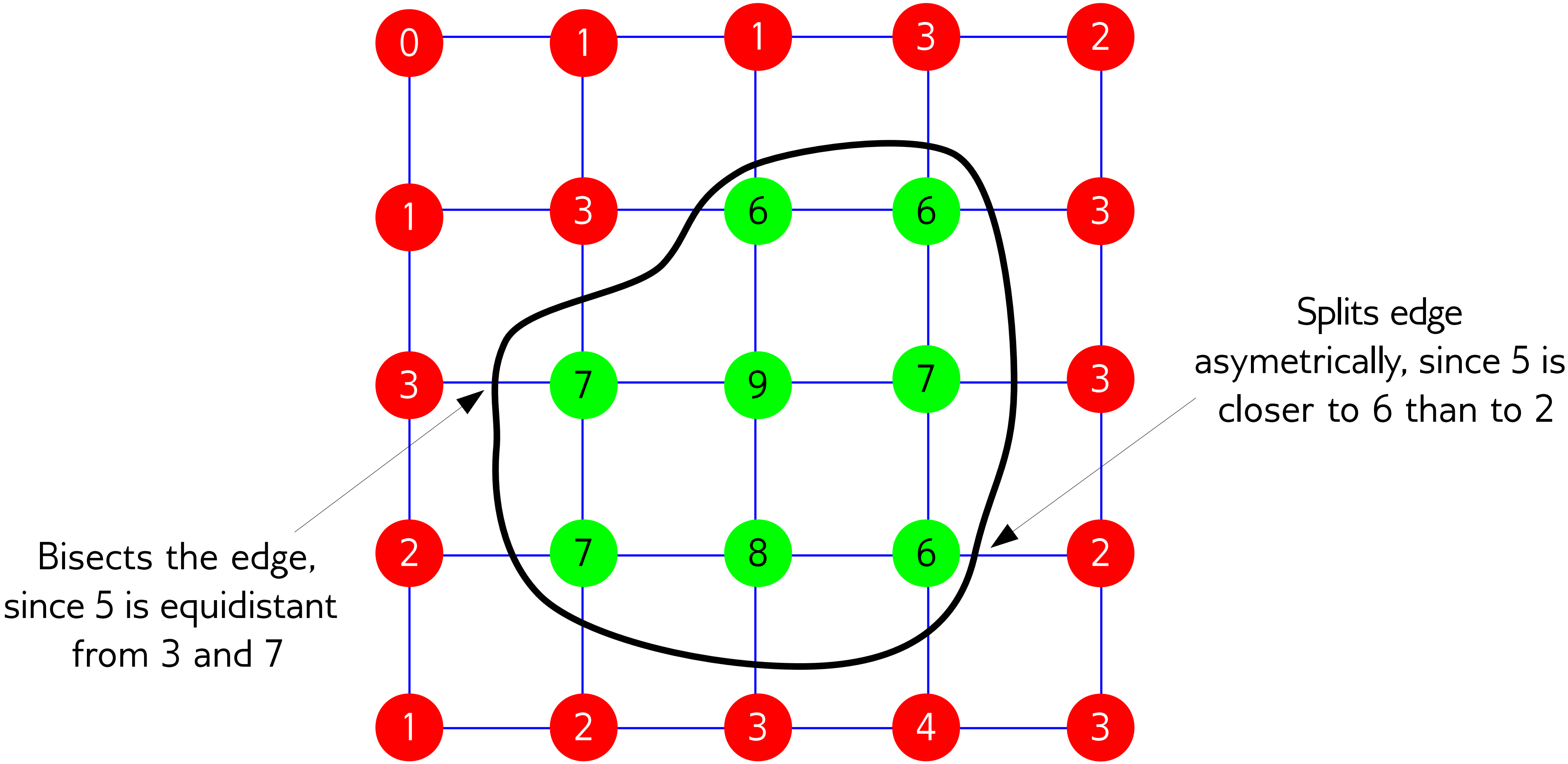
- **Data:** 2D structured grid of scalar values



Slides adapted from Toby Breckon

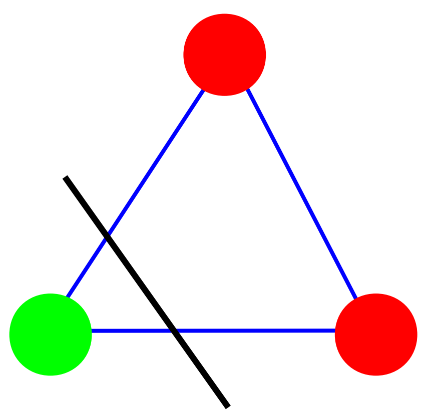
Isocontours

- The 5-level set:

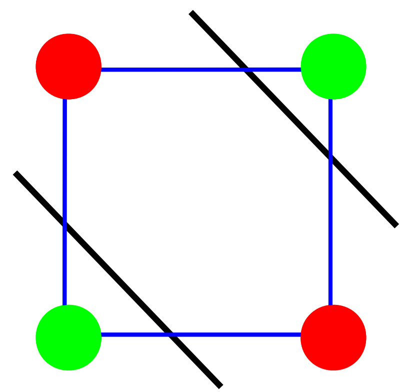


Isocontours: Ambiguity

- Where is the contour?



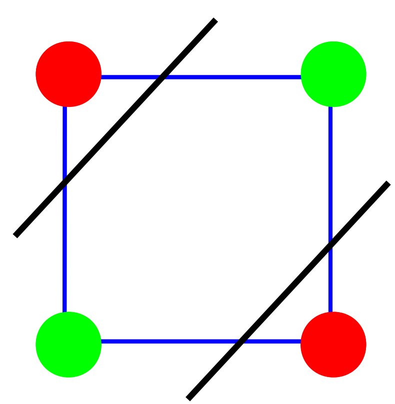
Triangular cell:
No ambiguities



“Split” green (inner) region

or

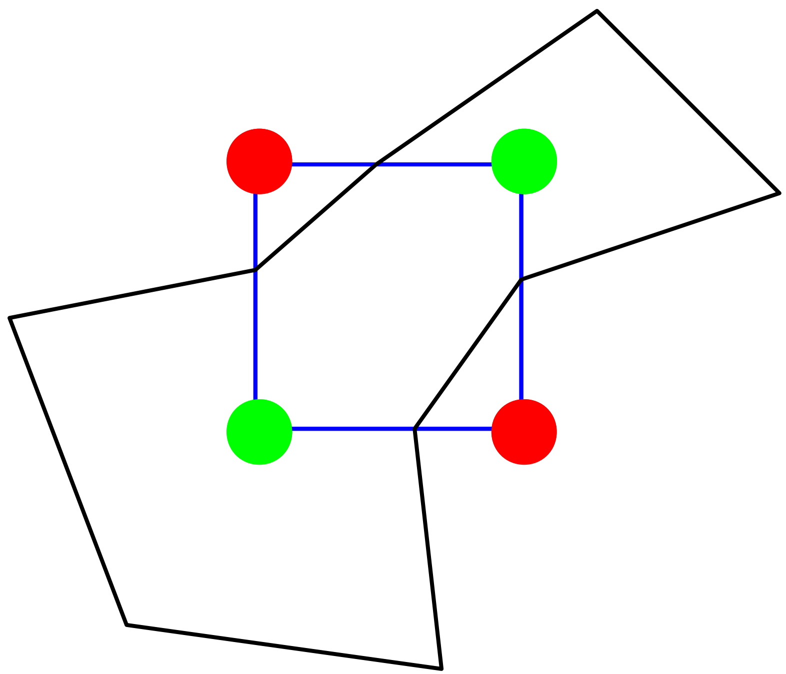
Square cell:
2 ambiguous cases



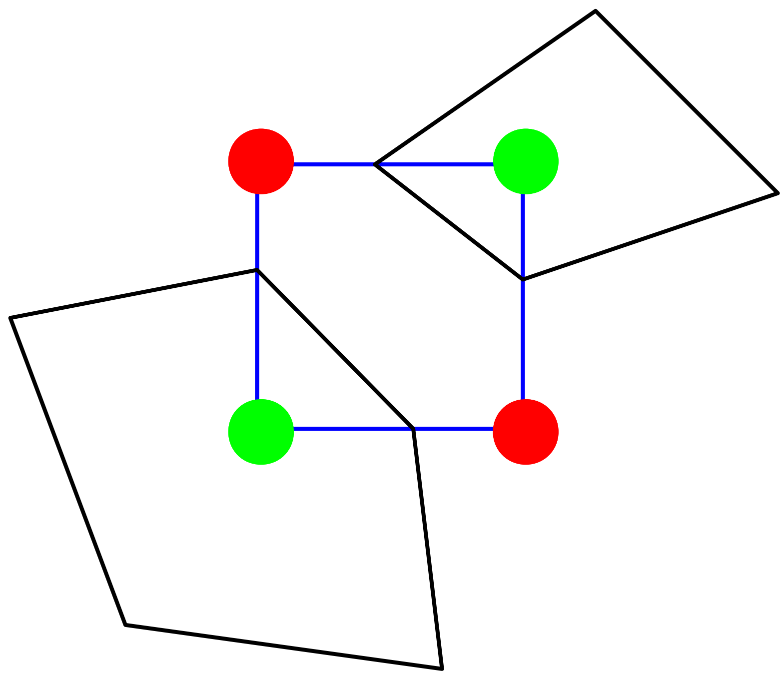
“Join” green (inner) region

Isocontours: Ambiguity

- Where is the contour?



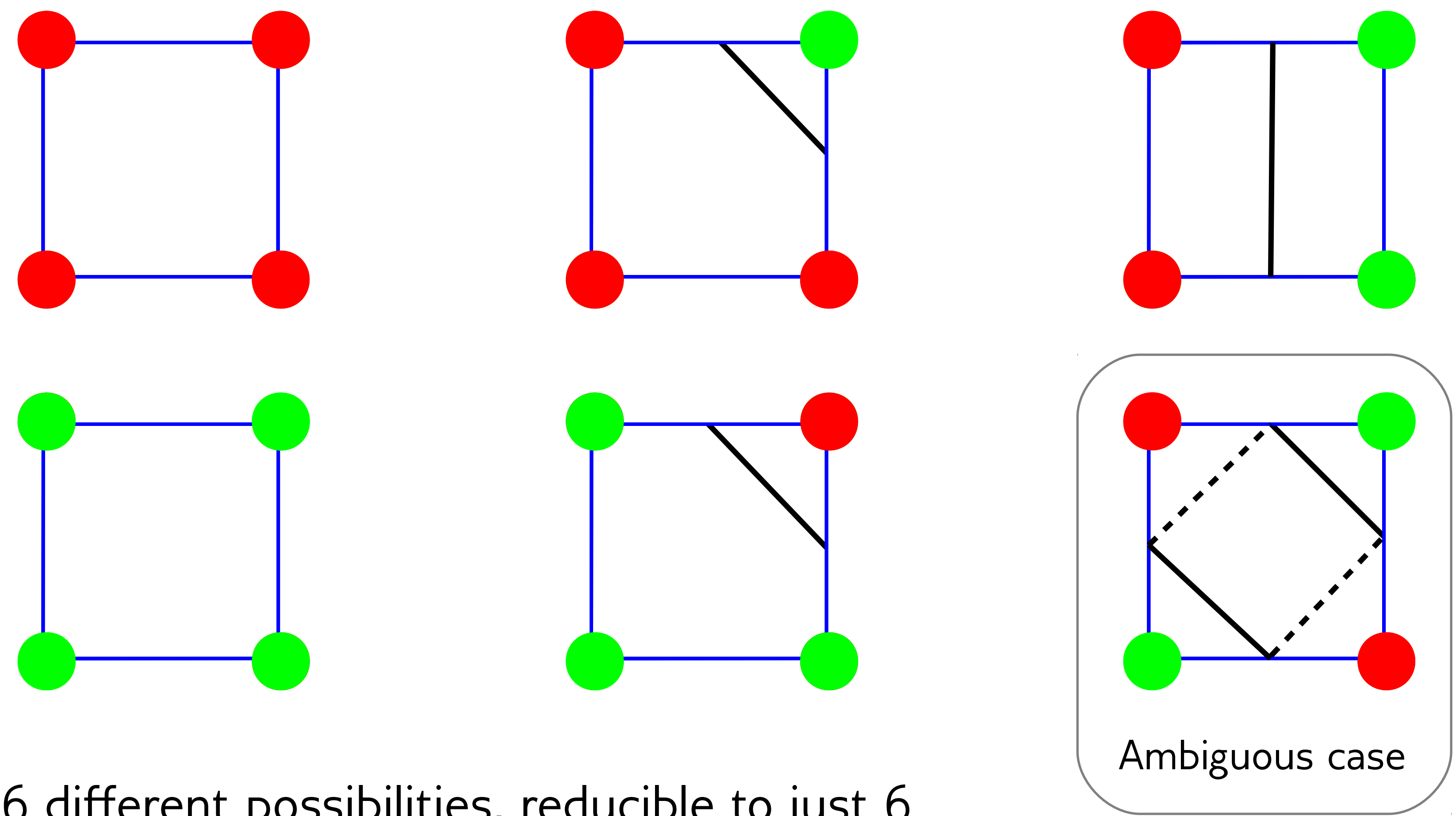
Join



Split

Isocontours: Cell Configurations

No intersections | 1 vertex different | 2 vertices different



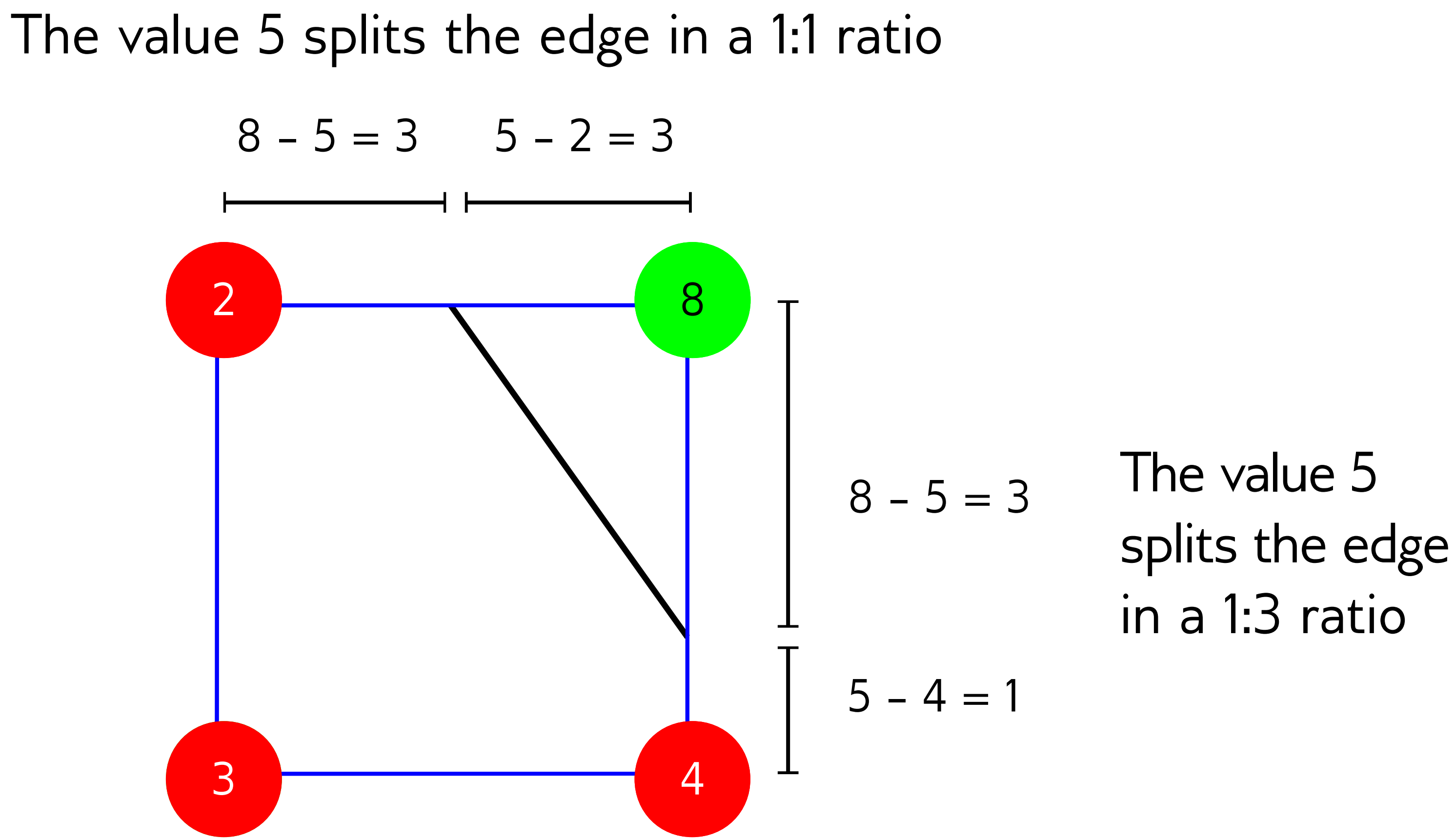
$2^4 = 16$ different possibilities, reducible to just 6 distinct cases after factoring out symmetries

Marching Squares Algorithm

- Select a starting cell
- Calculate inside/outside state for each vertex
- Classify cell configuration
  - Determine which edges are intersected
- Find exact locations of edge intersections
- Link up intersections to produce contour segment(s)
- Move (or "march") into next cell and repeat
  - … until all cells have been visited

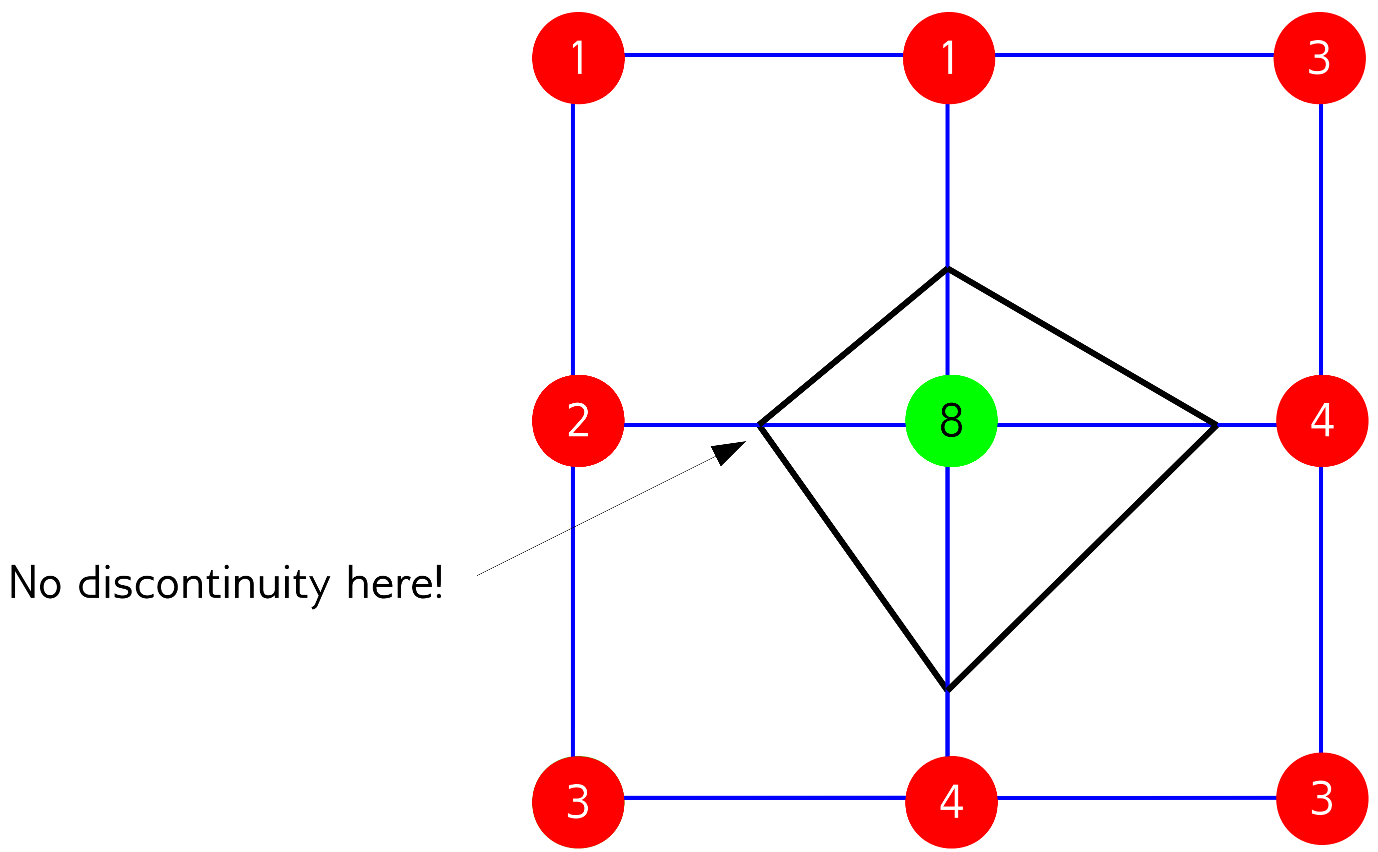
Where is the intersection?

- Find location of contour intersection with edge by interpolating vertex values



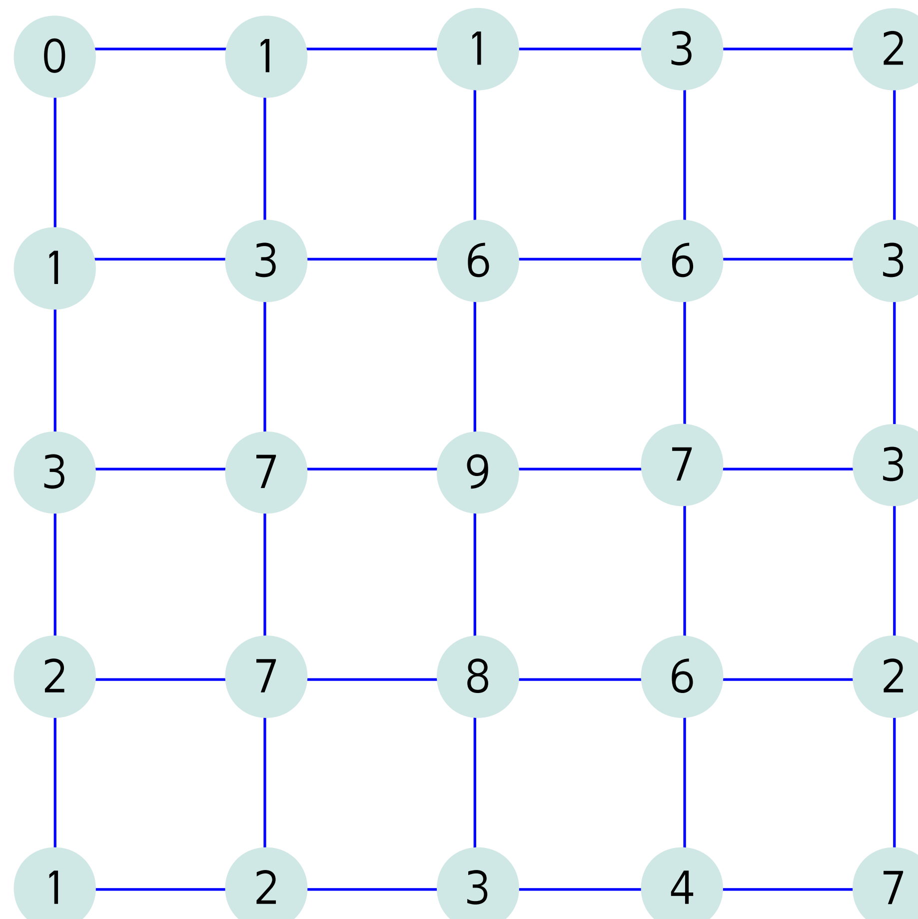
Contour continuity

- Since we only look at the endpoints of the edge, the generated contour is continuous across cells



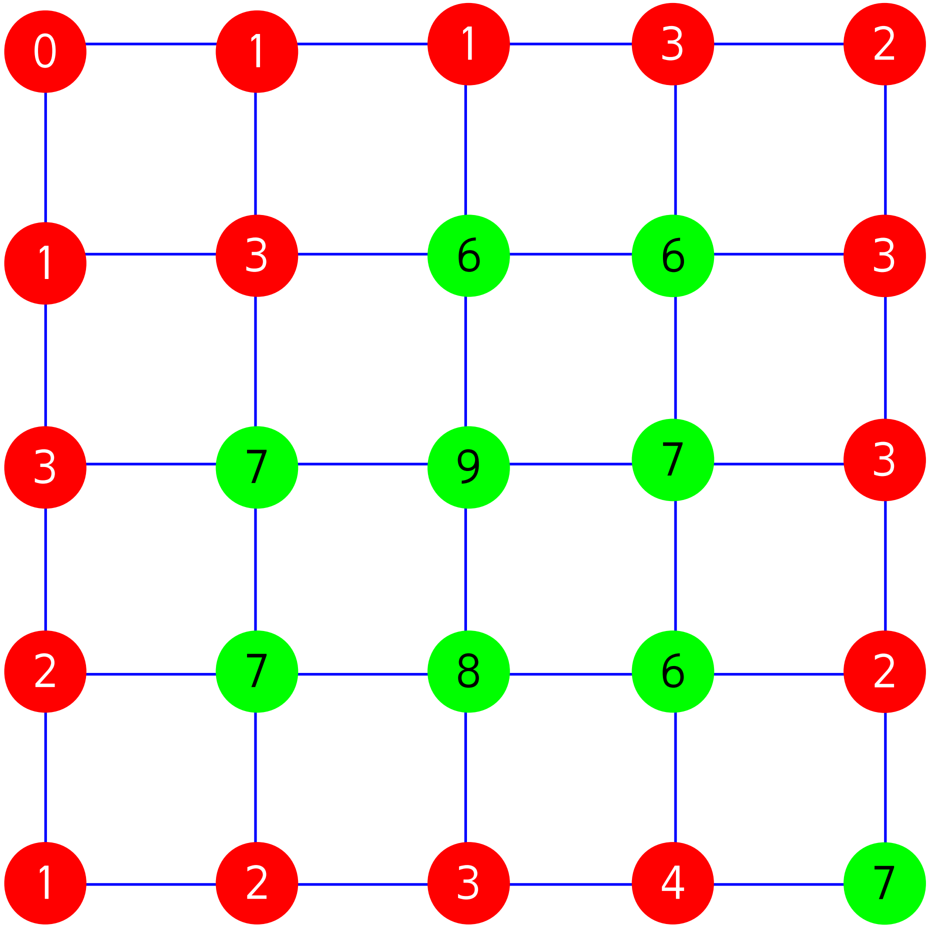
Example: Marching Squares

Find 5-contour of function represented by its values at vertices of a uniform grid



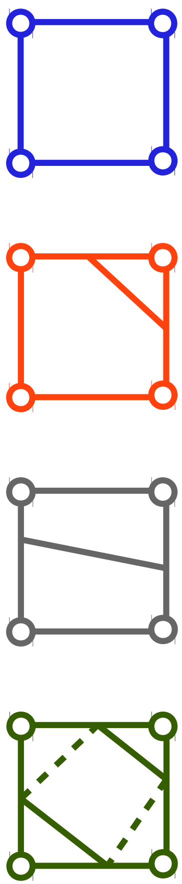
Step 1: Classify vertices

Green: inside
Red: outside



Slides adapted from Toby Breckon

Step 2: Classify cells

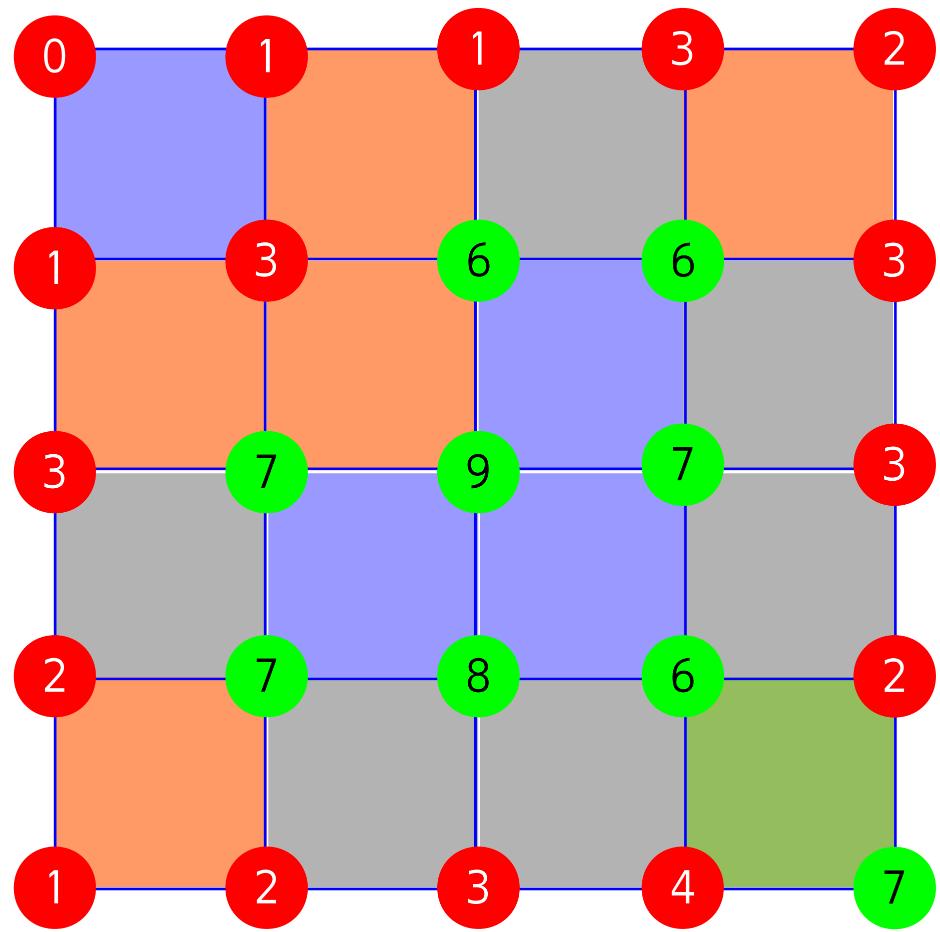


No intersections

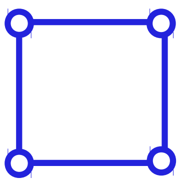
Adjacent edges

Opposite edges

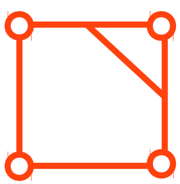
Ambiguous



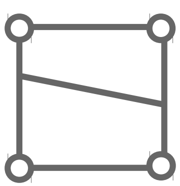
Step 3: Interpolate contour intersections



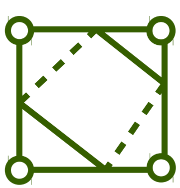
No intersections



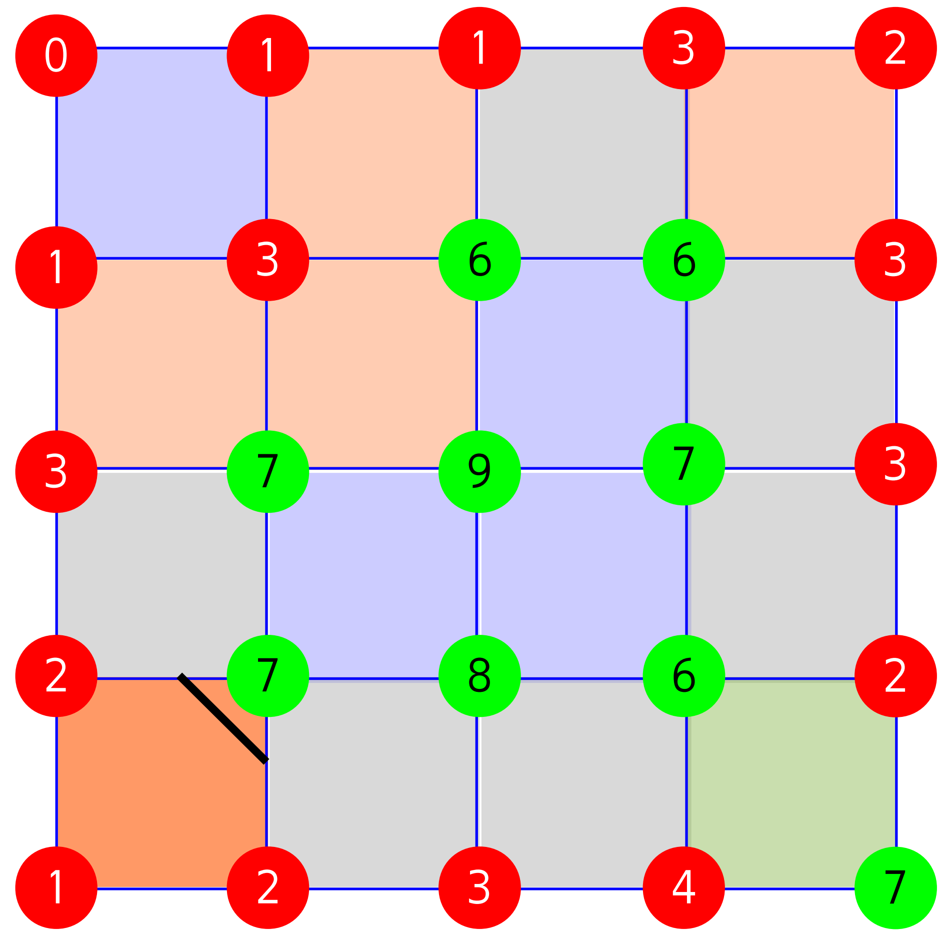
Adjacent edges



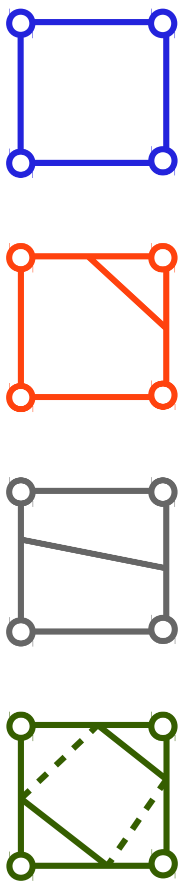
Opposite edges



Ambiguous



Step 3: Interpolate contour intersections

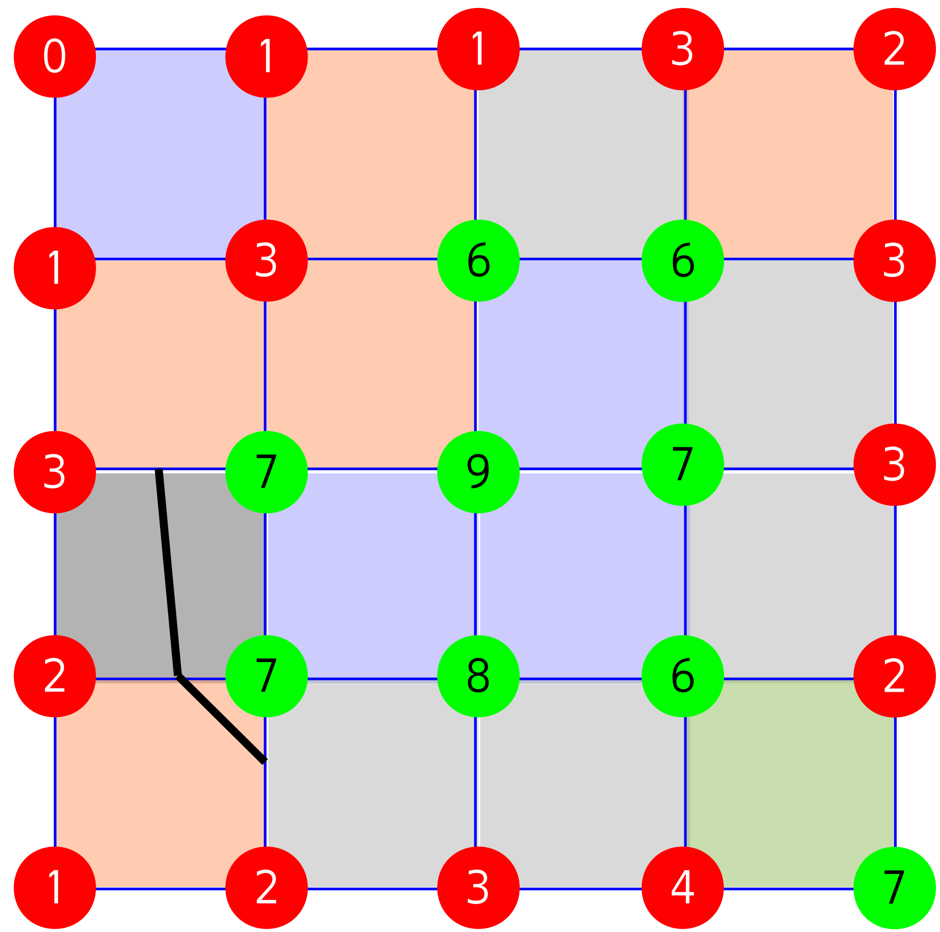


No intersections

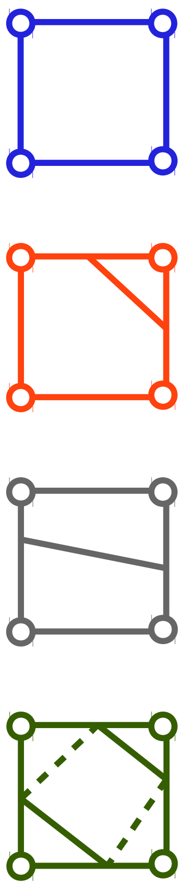
Adjacent edges

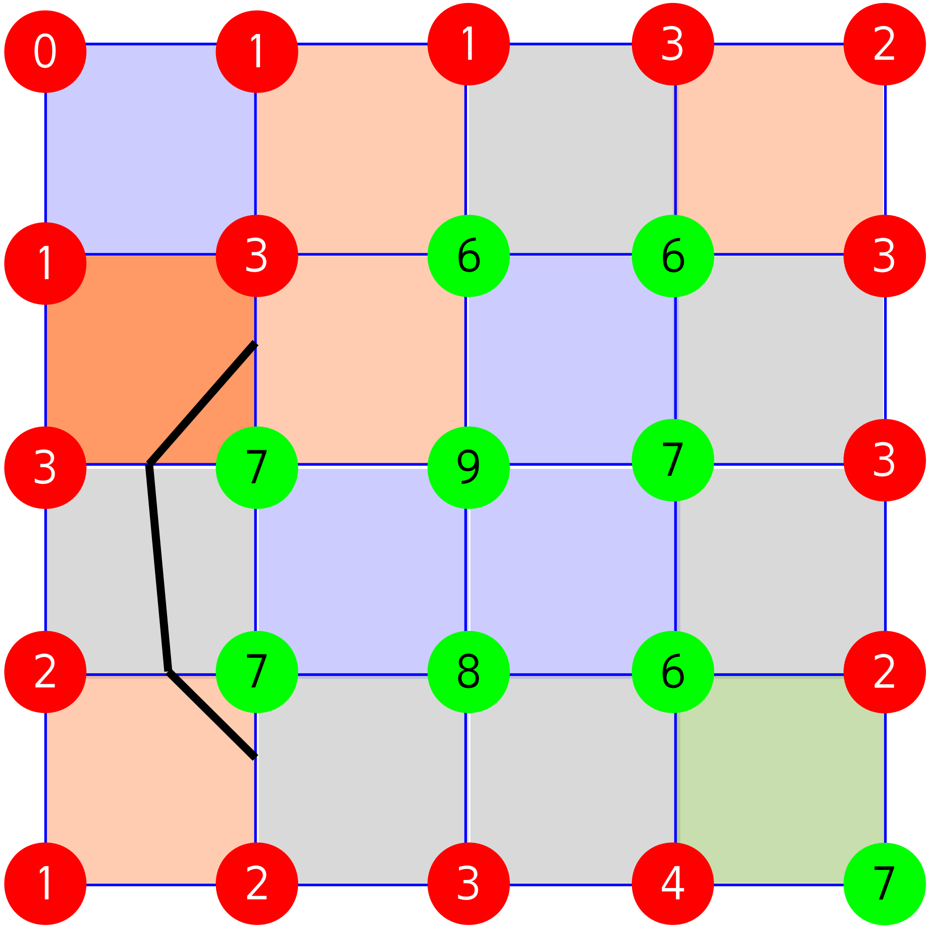
Opposite edges

Ambiguous

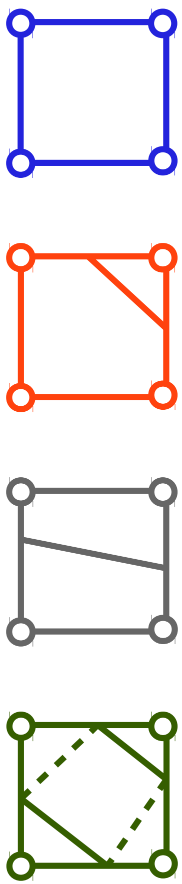


Step 3: Interpolate contour intersections





Step 3: Interpolate contour intersections

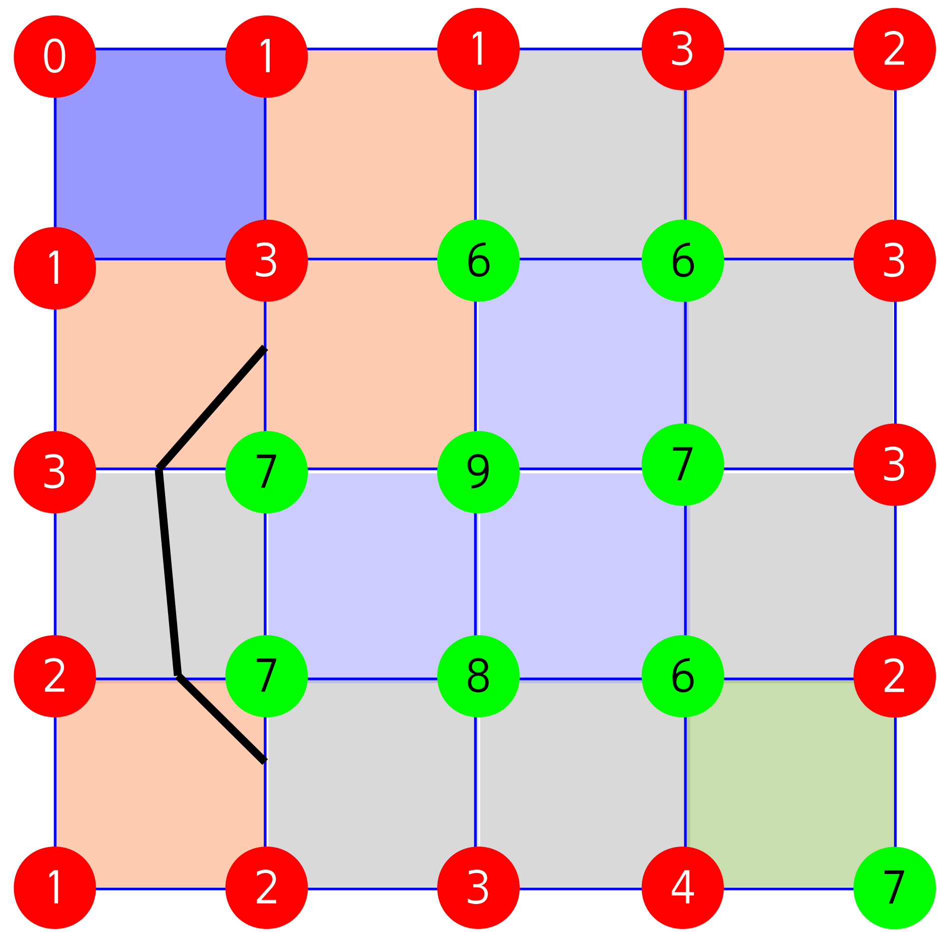


No intersections

Adjacent edges

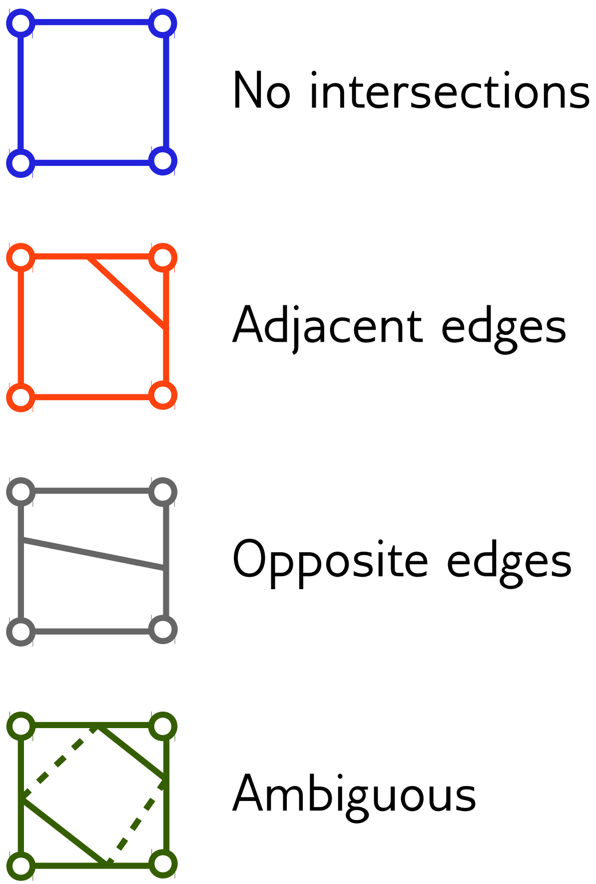
Opposite edges

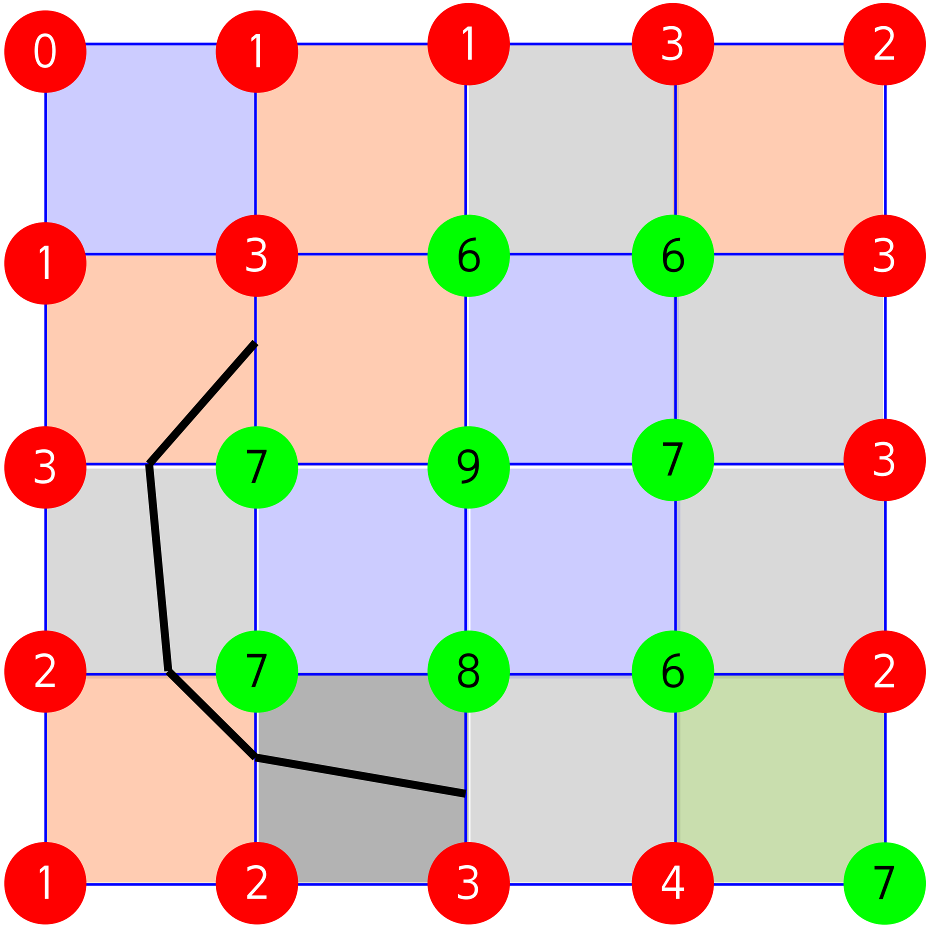
Ambiguous



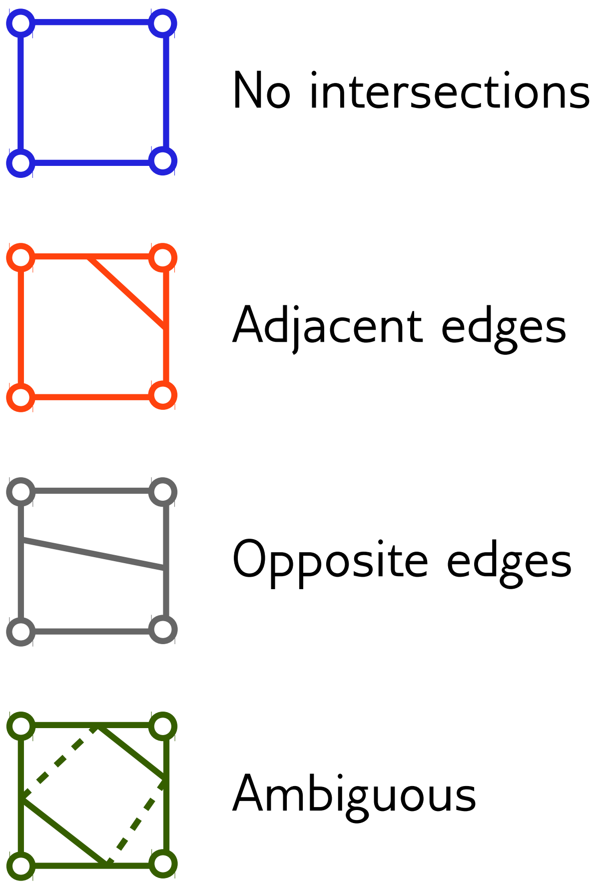
Slides adapted from Toby Breckon

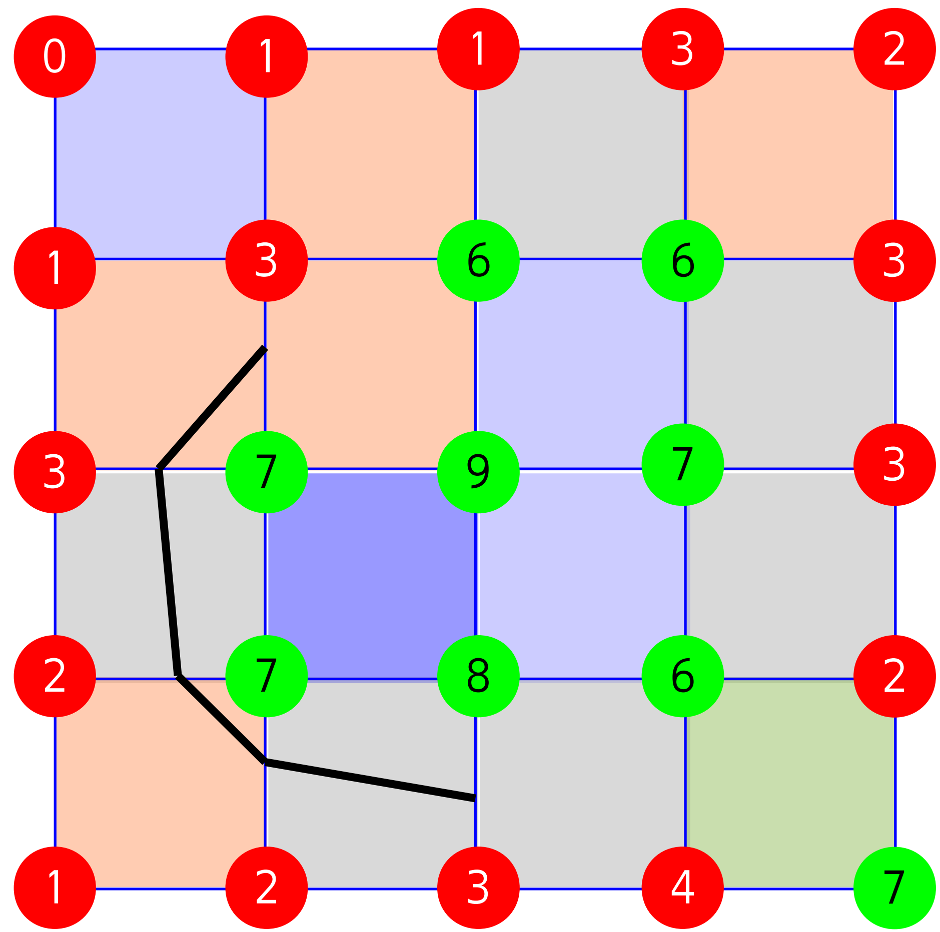
Step 3: Interpolate contour intersections





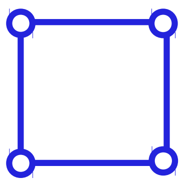
Step 3: Interpolate contour intersections

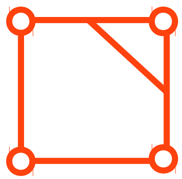


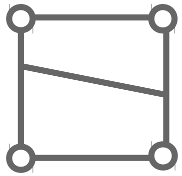


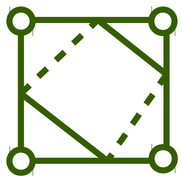
Slides adapted from Toby Breckon

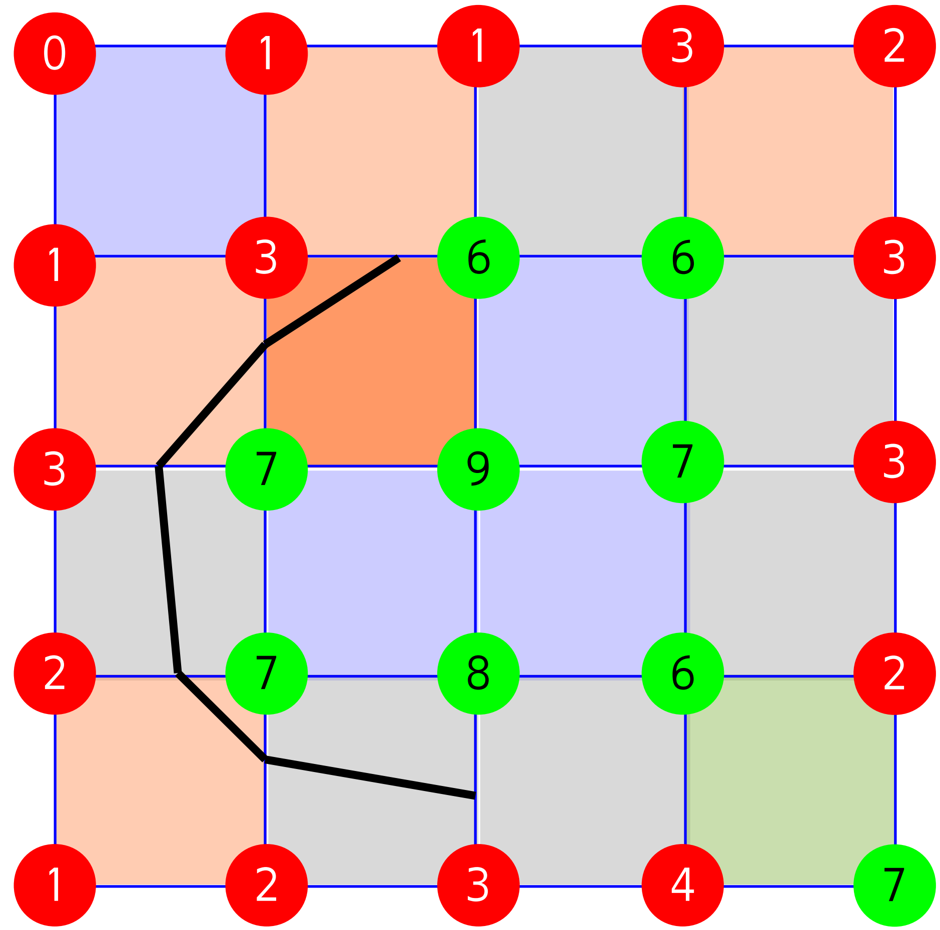
Step 3: Interpolate contour intersections


No intersections

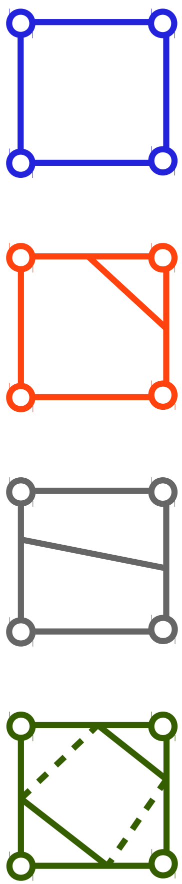

Adjacent edges


Opposite edges


Ambiguous



Step 3: Interpolate contour intersections

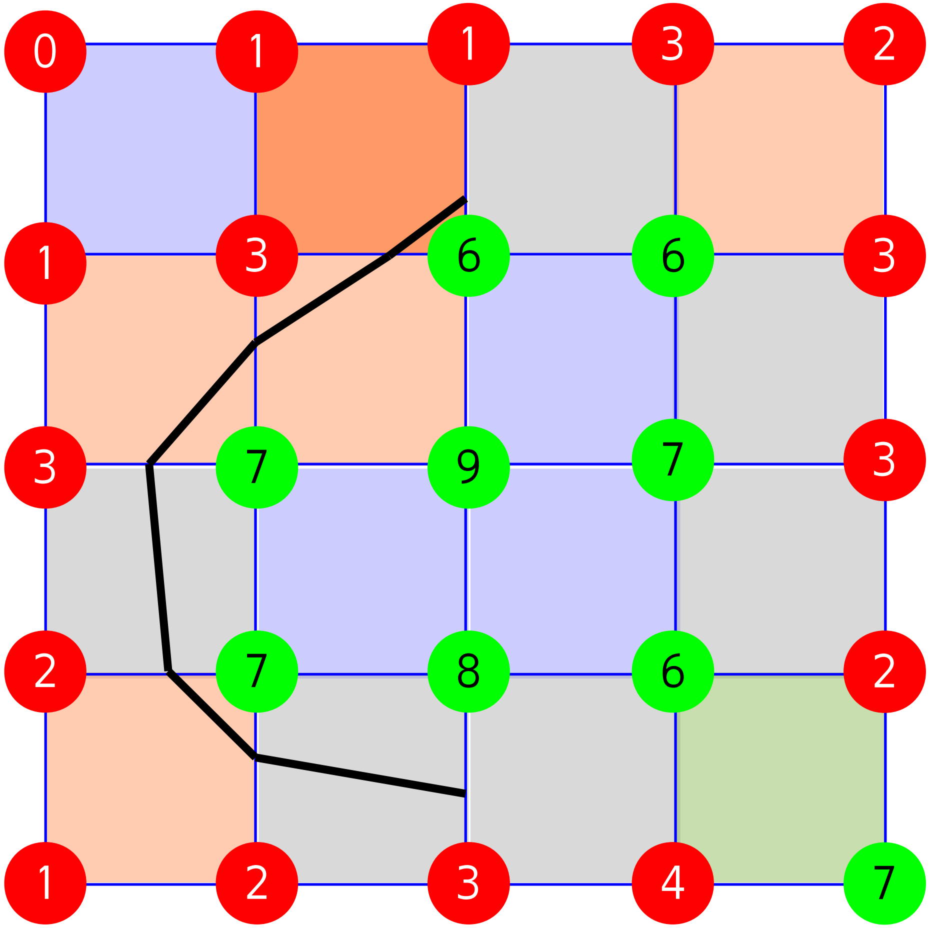


No intersections

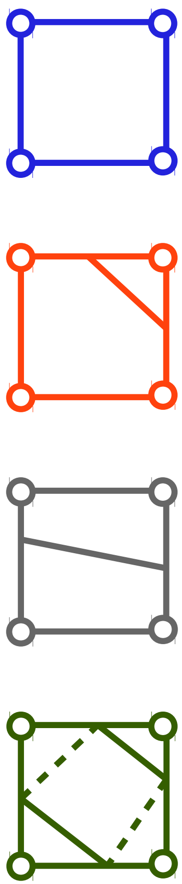
Adjacent edges

Opposite edges

Ambiguous



Step 3: Interpolate contour intersections

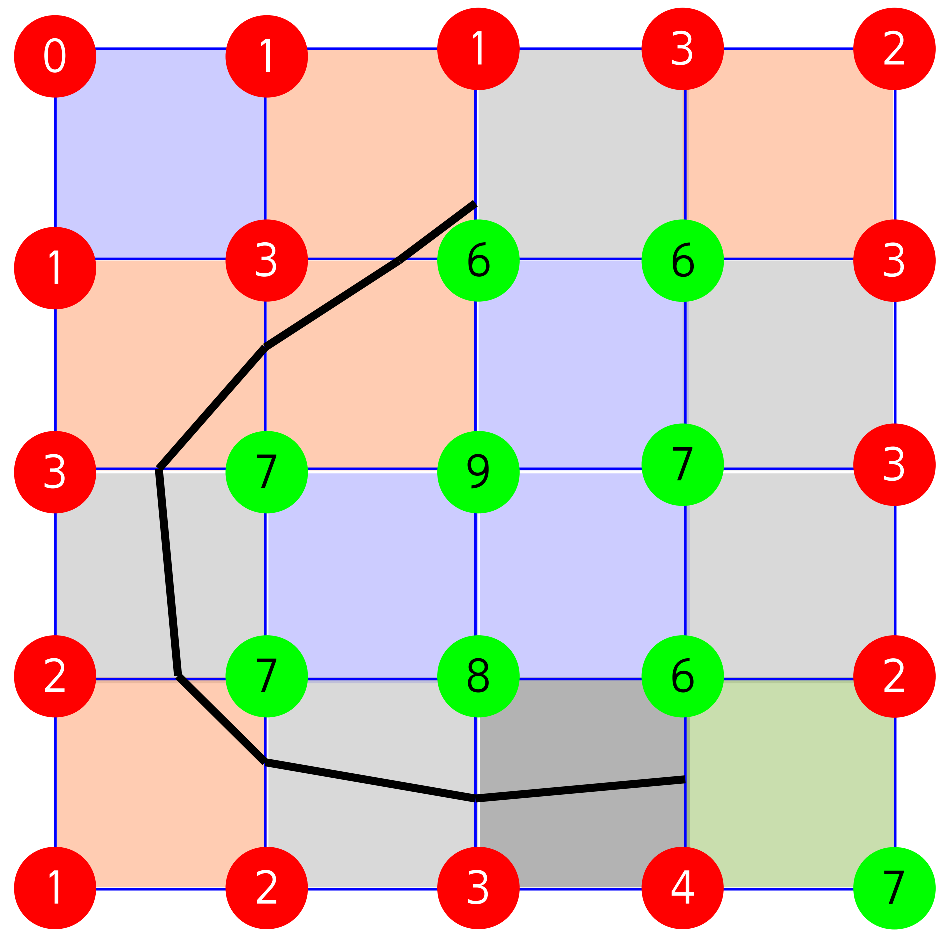


No intersections

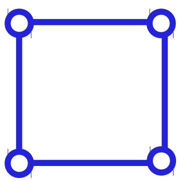
Adjacent edges

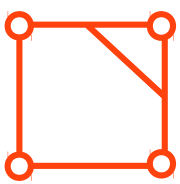
Opposite edges

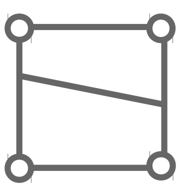
Ambiguous

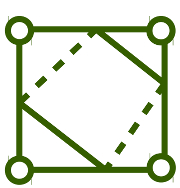


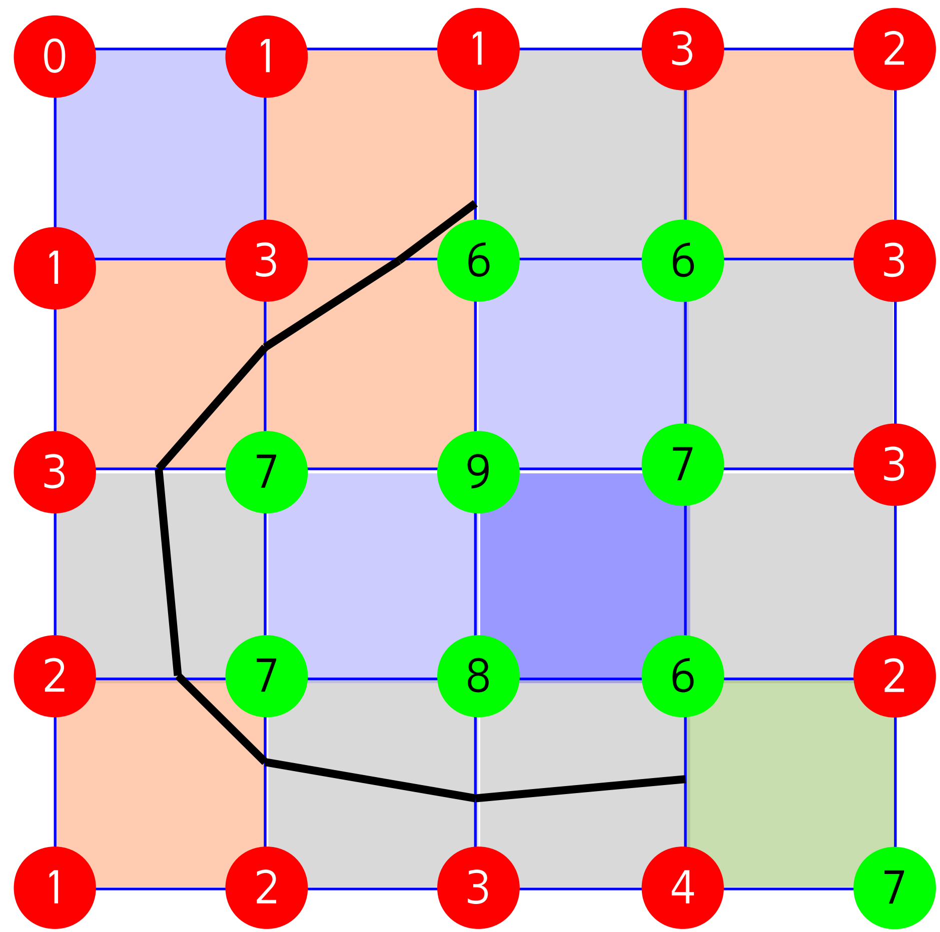
Step 3: Interpolate contour intersections


No intersections


Adjacent edges

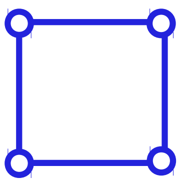

Opposite edges


Ambiguous

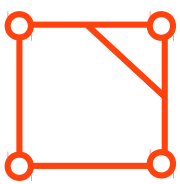


Slides adapted from Toby Breckon

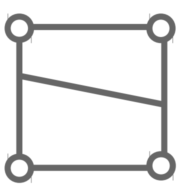
Step 3: Interpolate contour intersections



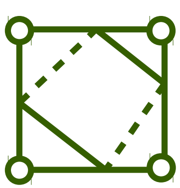
No intersections



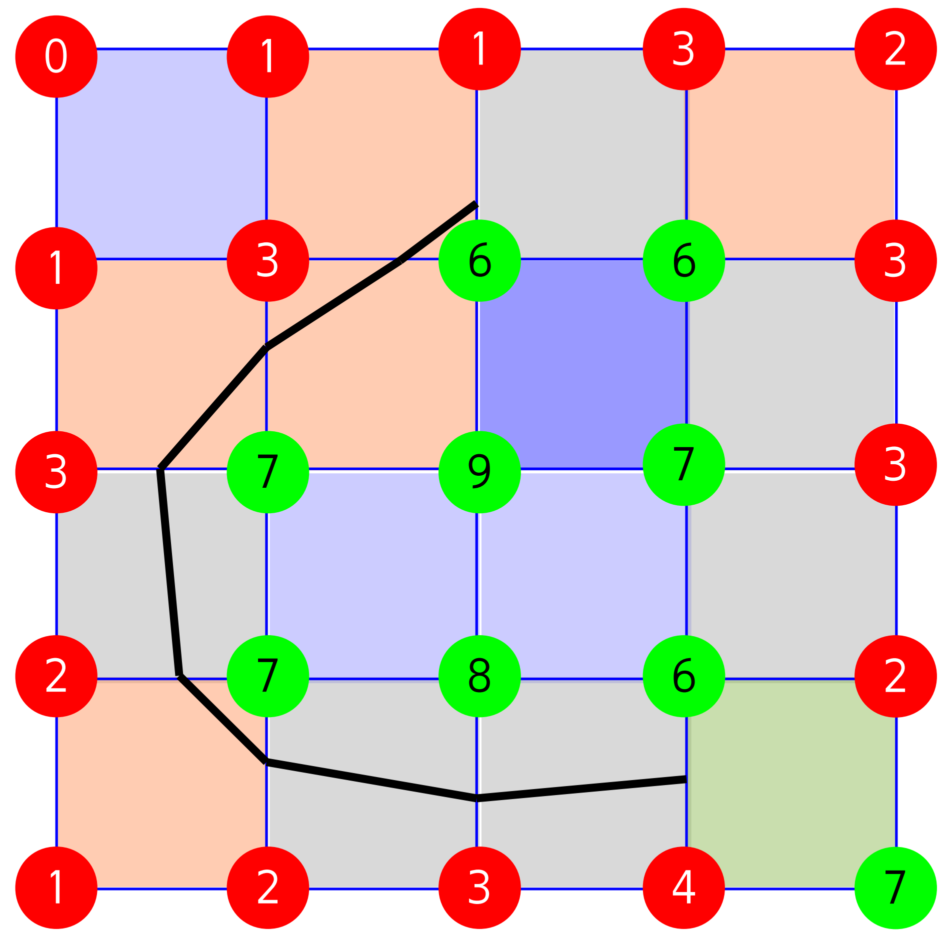
Adjacent edges



Opposite edges

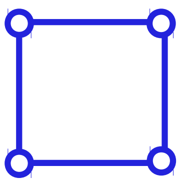


Ambiguous

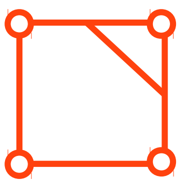


Slides adapted from Toby Breckon

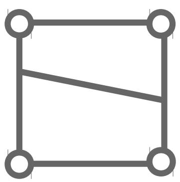
Step 3: Interpolate contour intersections



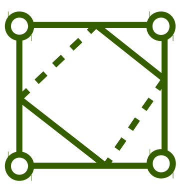
No intersections



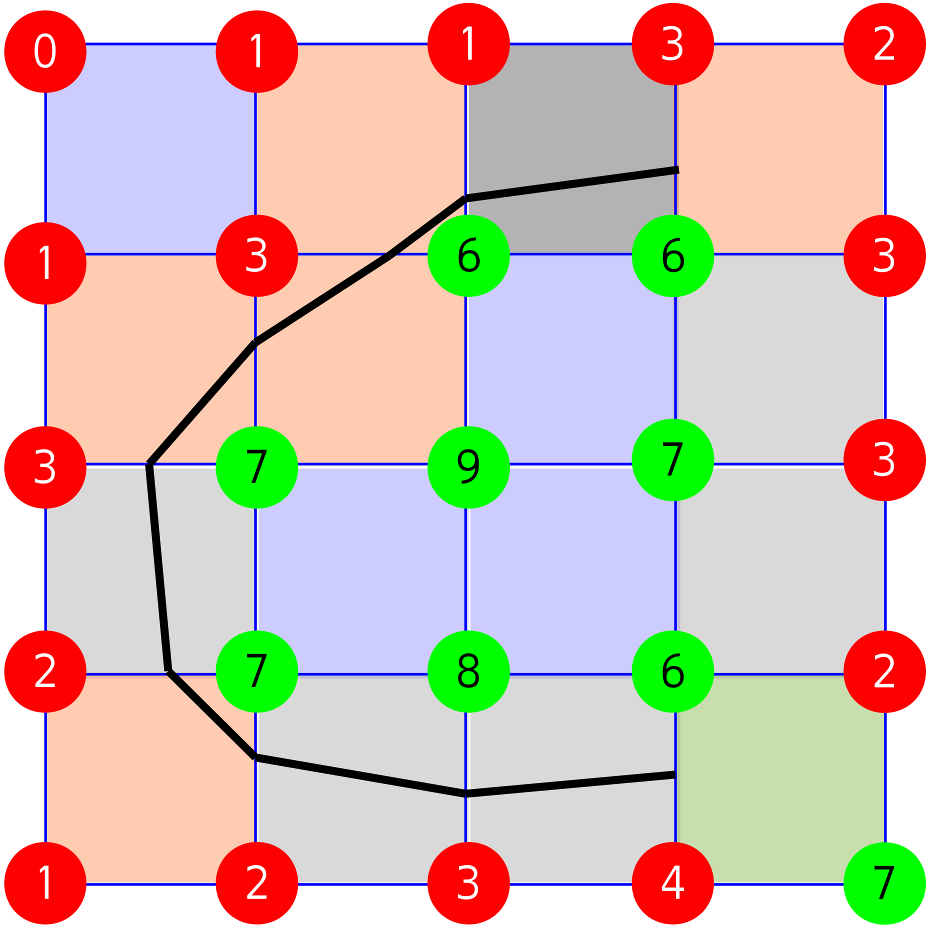
Adjacent edges



Opposite edges

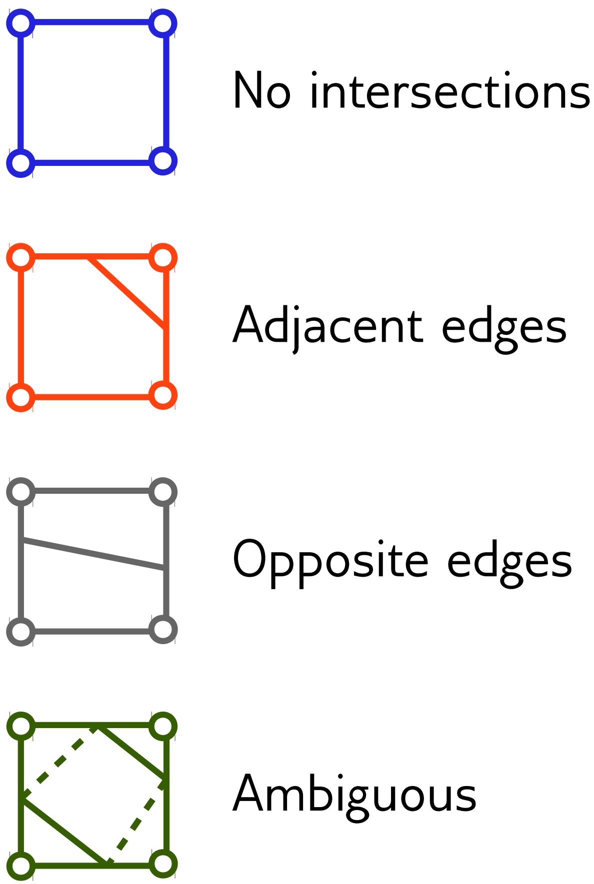


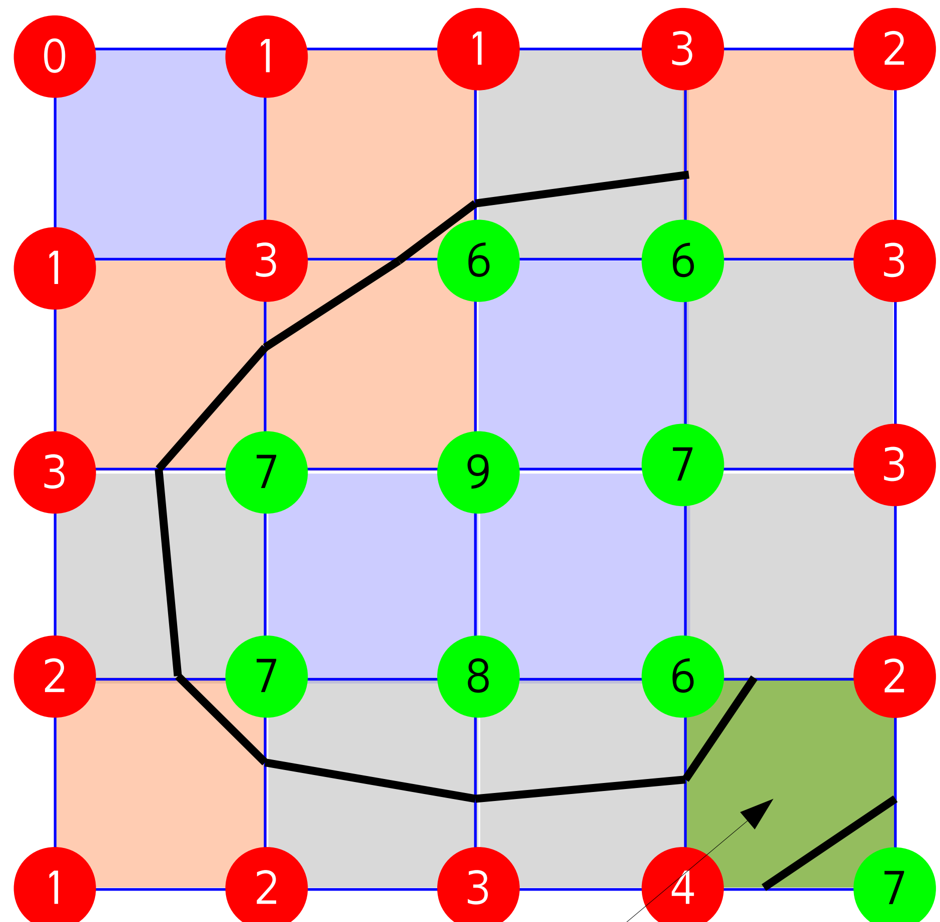
Ambiguous



Slides adapted from Toby Breckon

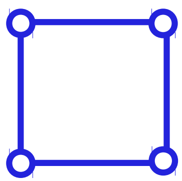
Step 3: Interpolate contour intersections



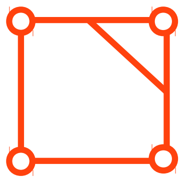


Arbitrarily choose to split here, instead of join. We could also have gone the other way.

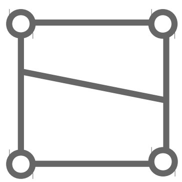
Step 3: Interpolate contour intersections



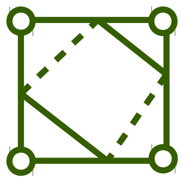
No intersections



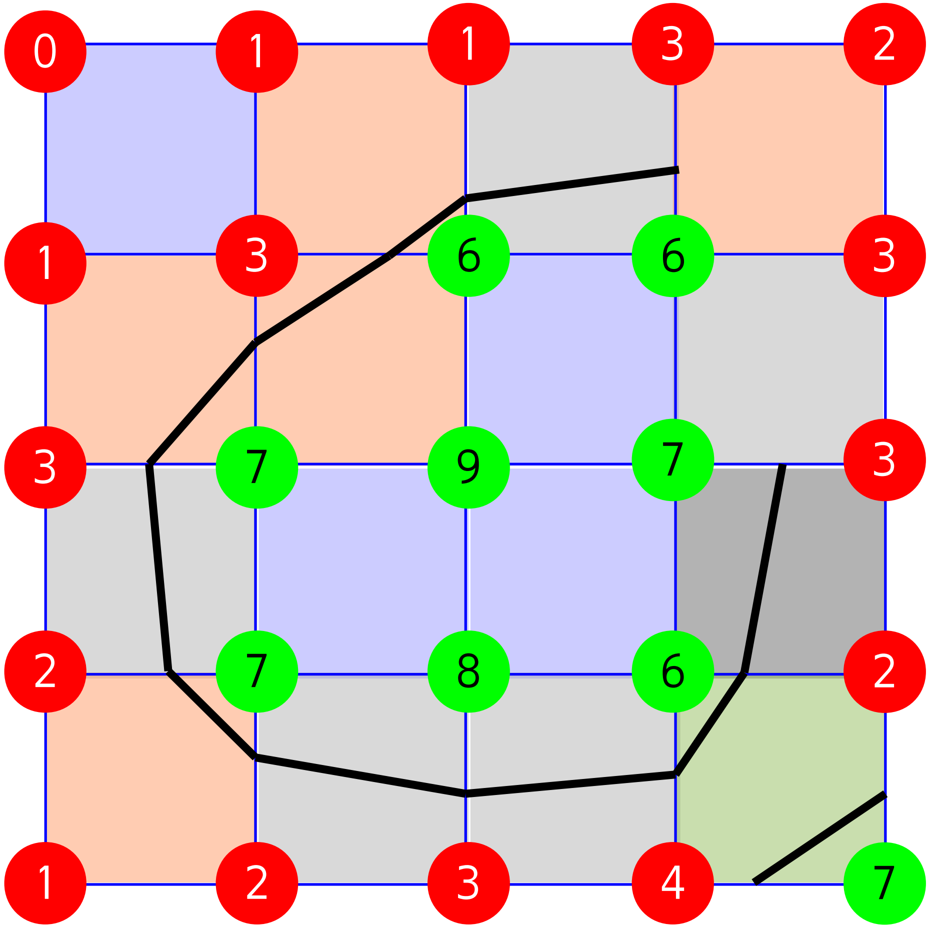
Adjacent edges



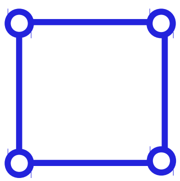
Opposite edges

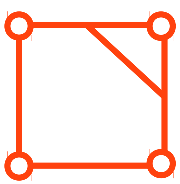


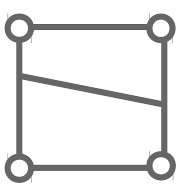
Ambiguous

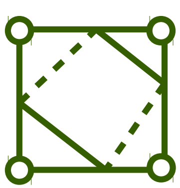


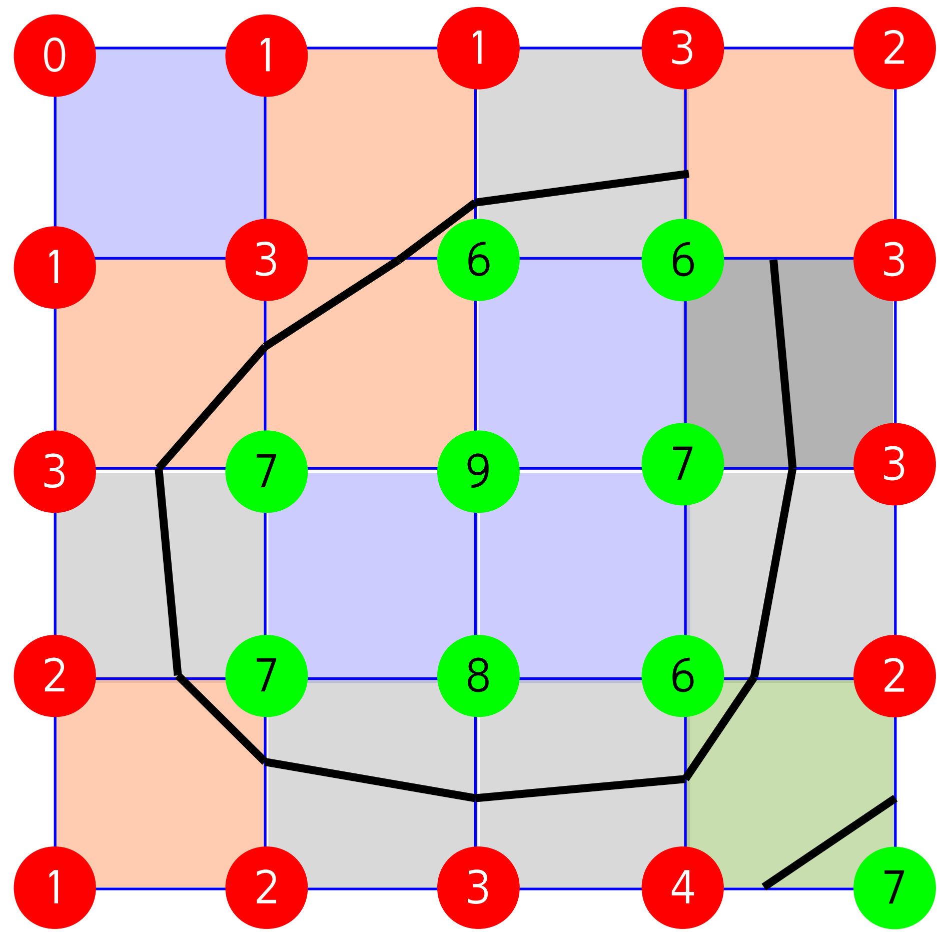
Step 3: Interpolate contour intersections


No intersections


Adjacent edges

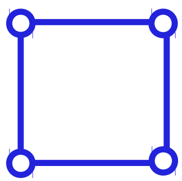

Opposite edges


Ambiguous

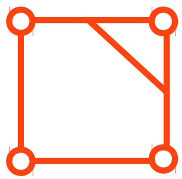


Slides adapted from Toby Breckon

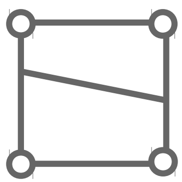
Step 3: Interpolate contour intersections



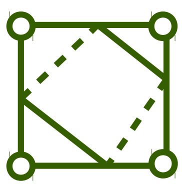
No intersections



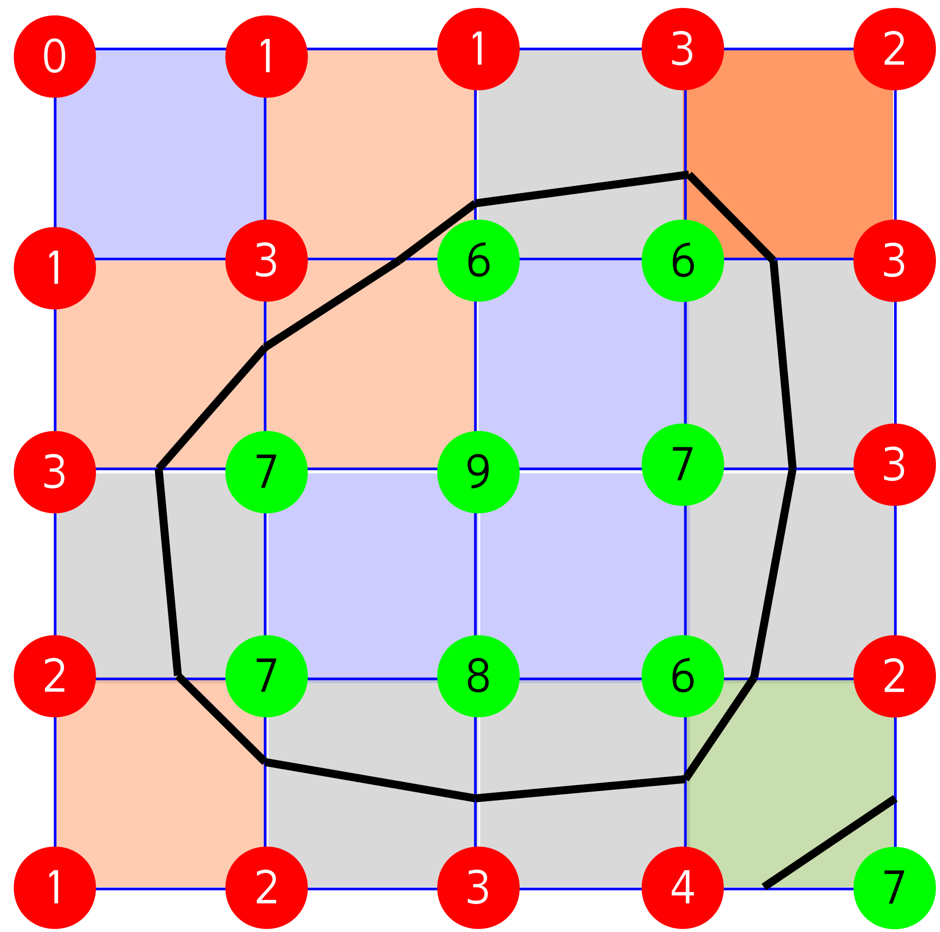
Adjacent edges



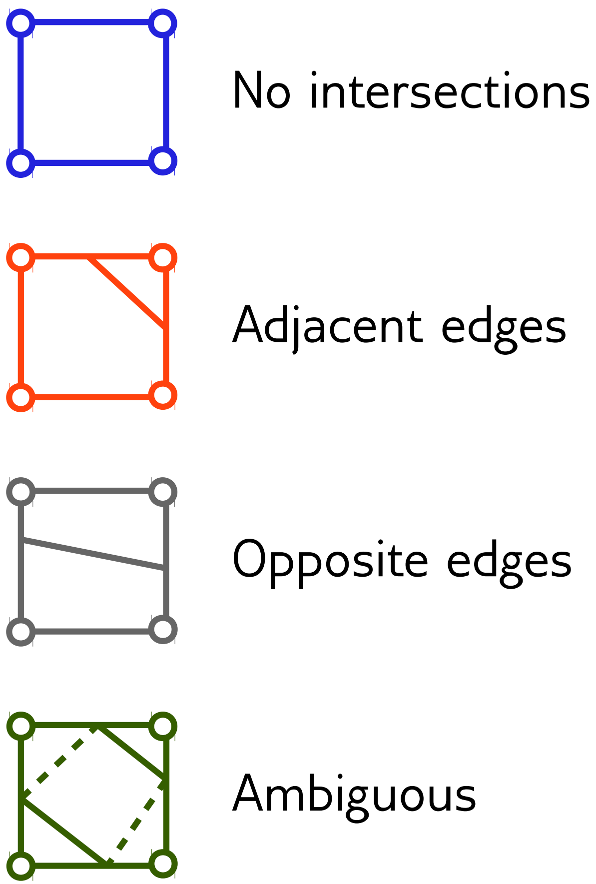
Opposite edges

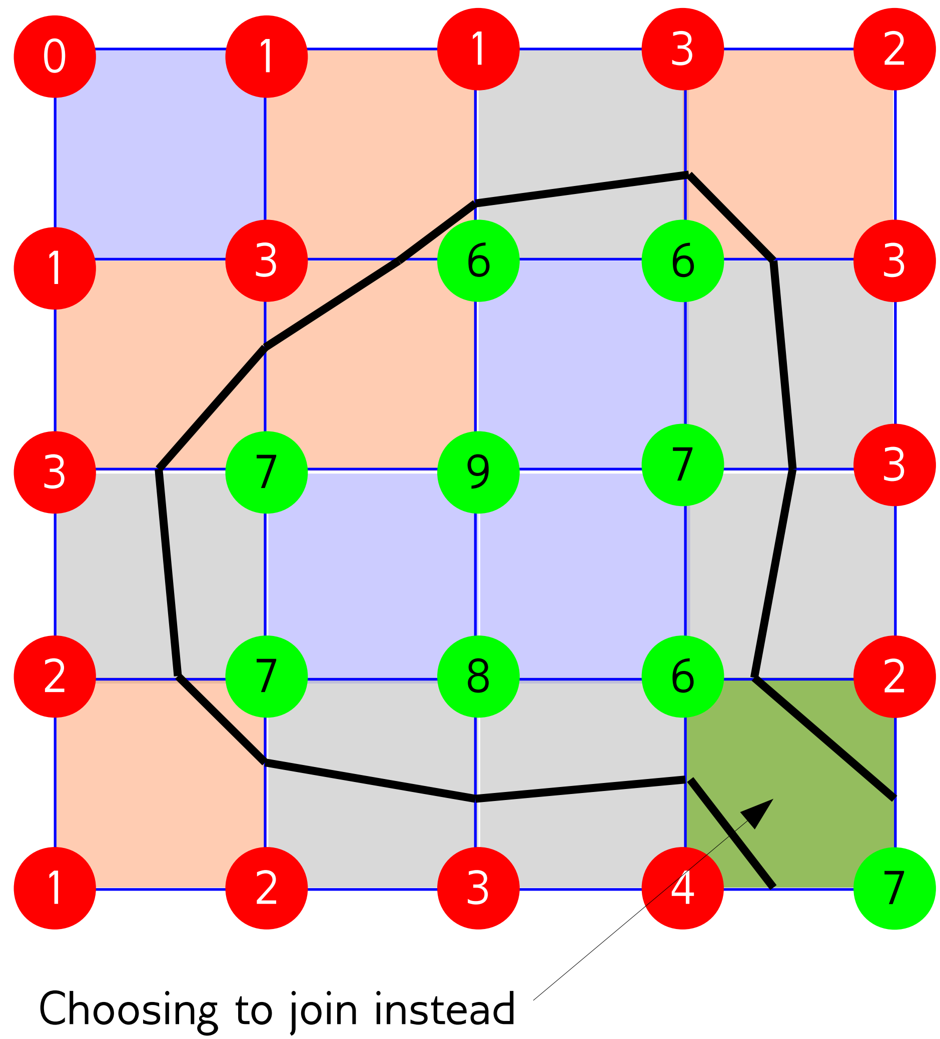


Ambiguous



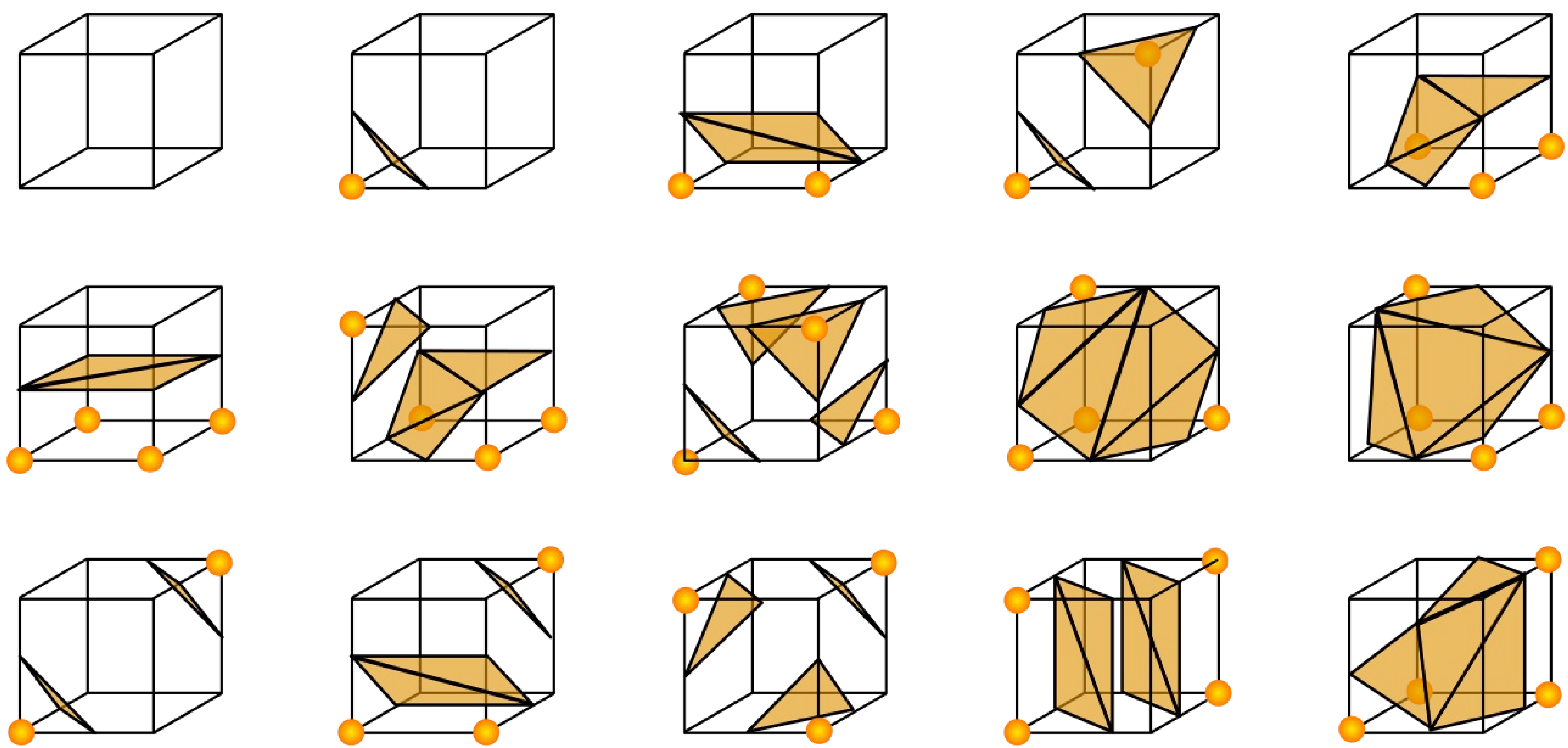
Resolving ambiguities





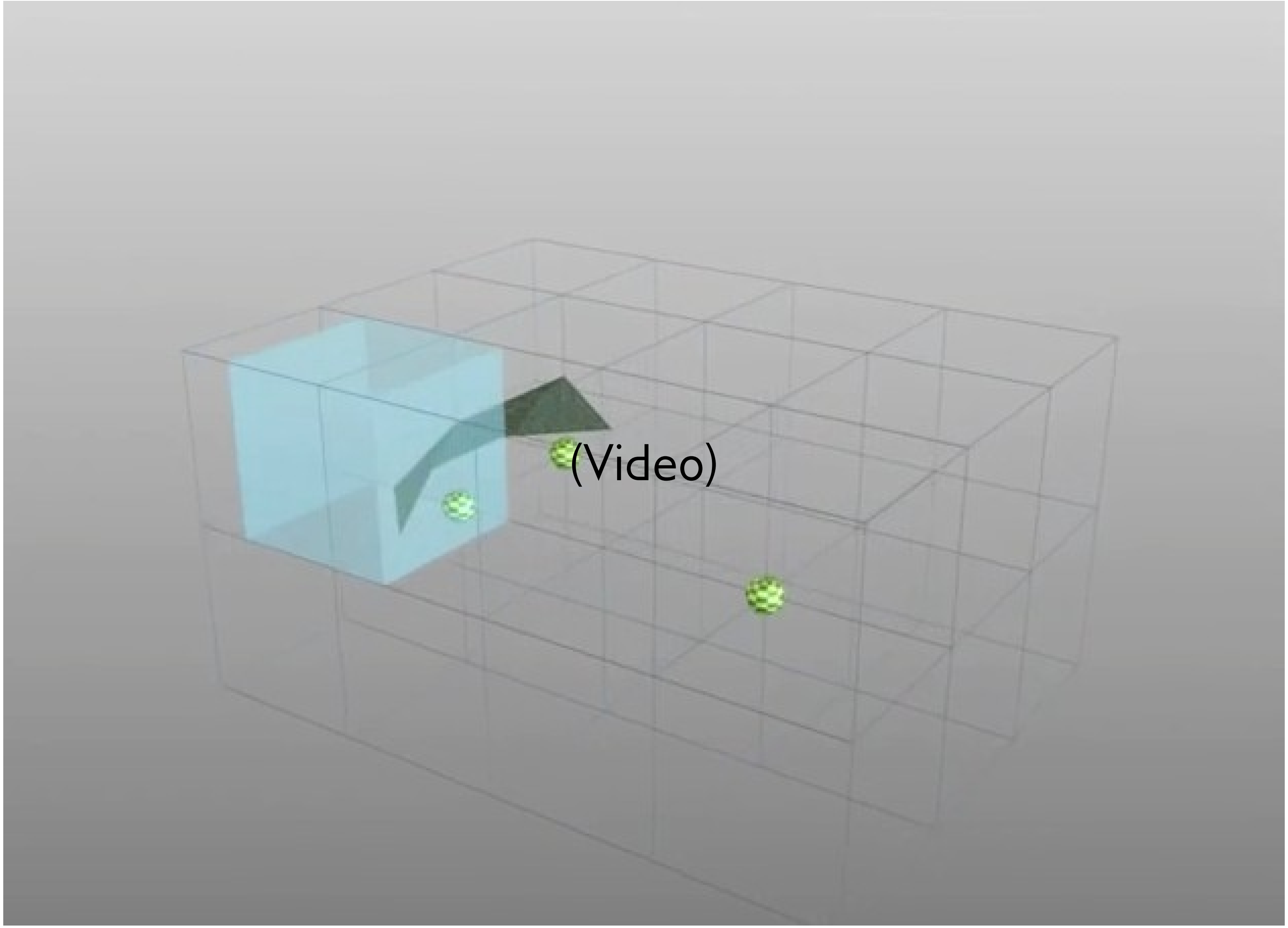
In 3D: Marching Cubes

Exactly the same algorithm, but cells are now cubes (15 distinct configurations) and output is triangles (or a polygon mix)



Wikipedia

In 3D: Marching Cubes



Koen Samyn, https://www.youtube.com/watch?v=LfttaAepYJ8

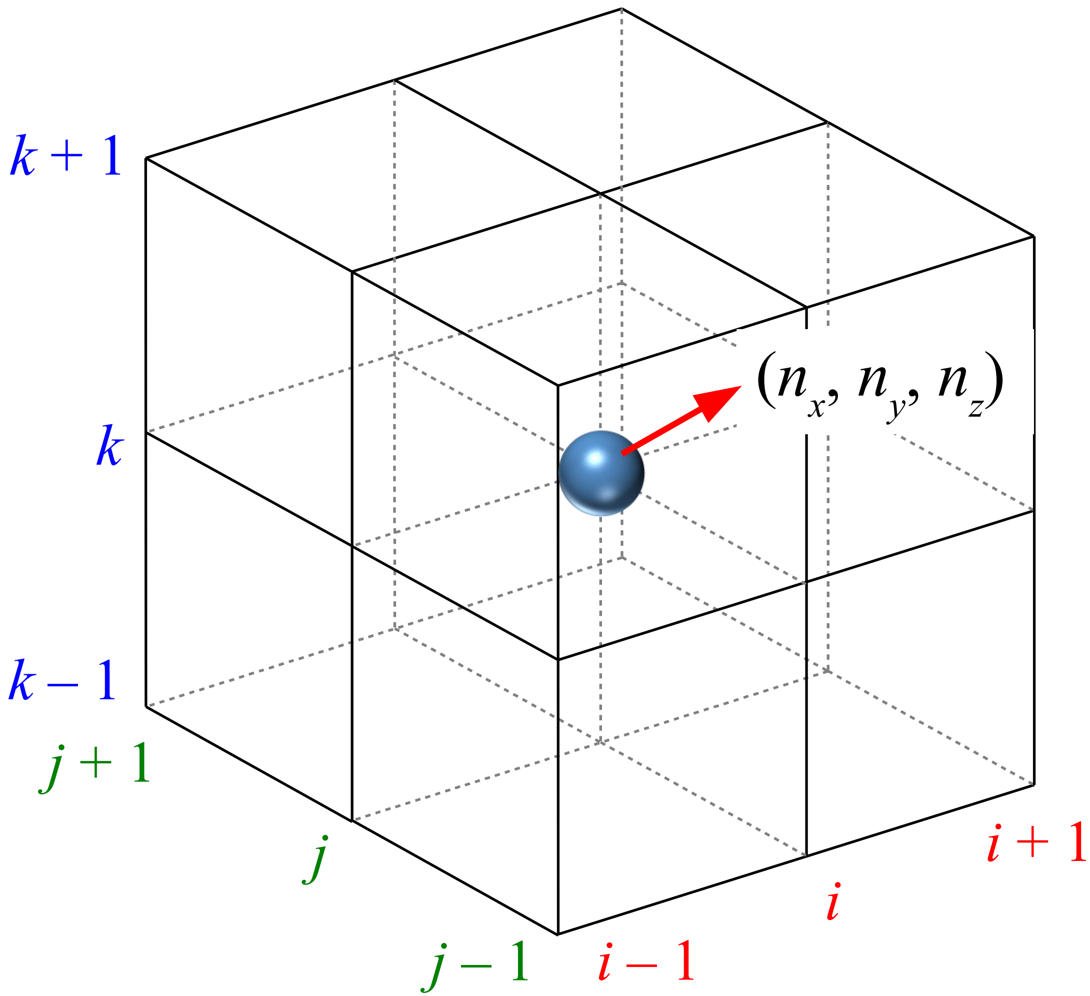
Marching Cubes: Estimating Normals

- We could estimate normals from the generated mesh, but the density function has more information
- Recall: The normal to the surface is the gradient of the density function

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$

- We will estimate the gradient from the grid of values

Normals at Cube Vertices



Discrete approximation to the gradient at the blue cube vertex

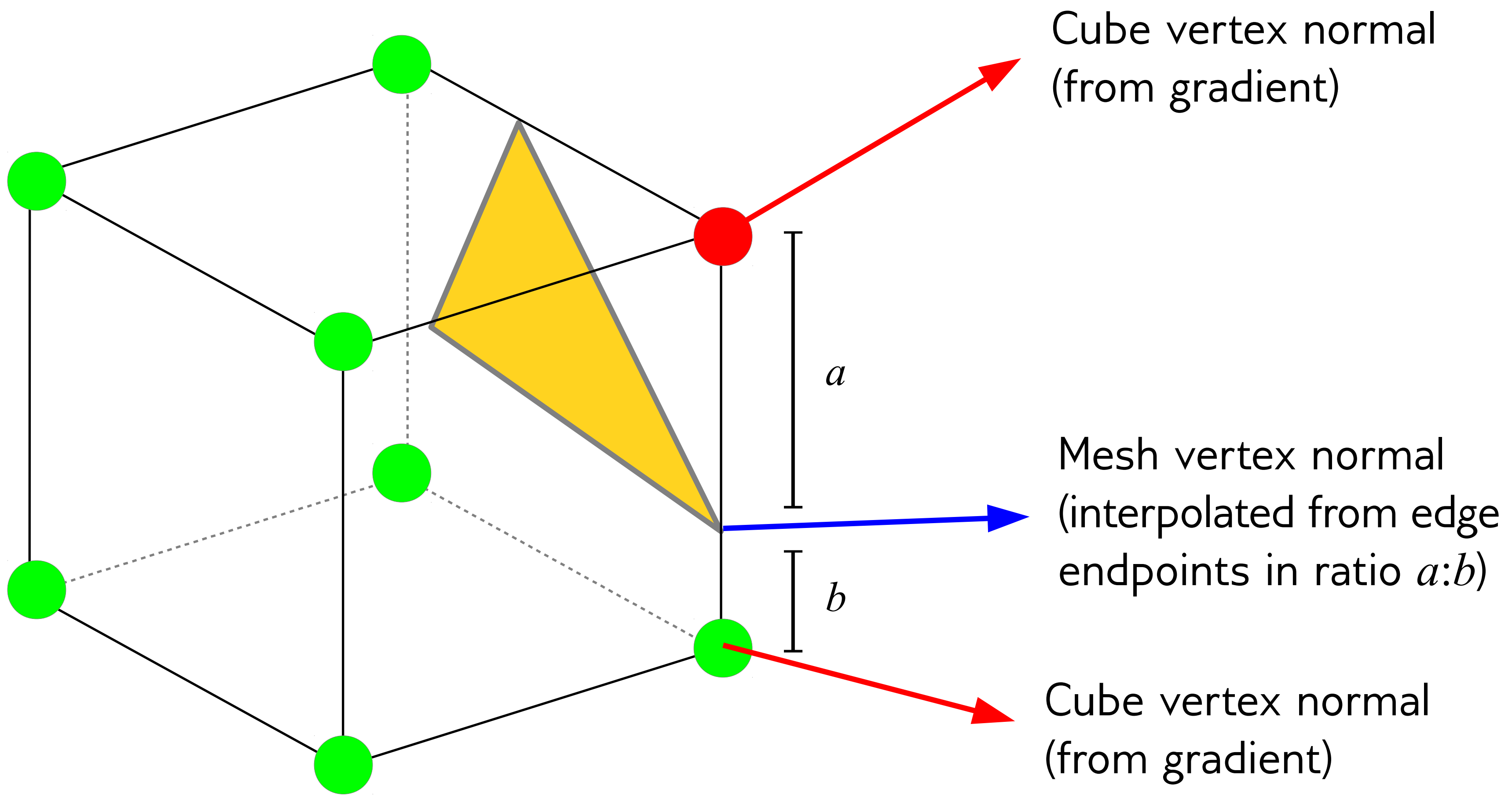
$$n_x = \frac{f(i+1, j, k) - f(i-1, j, k)}{2\Delta x}$$

$$n_x = \frac{f(i, j+1, k) - f(i, j-1, k)}{2\Delta y}$$

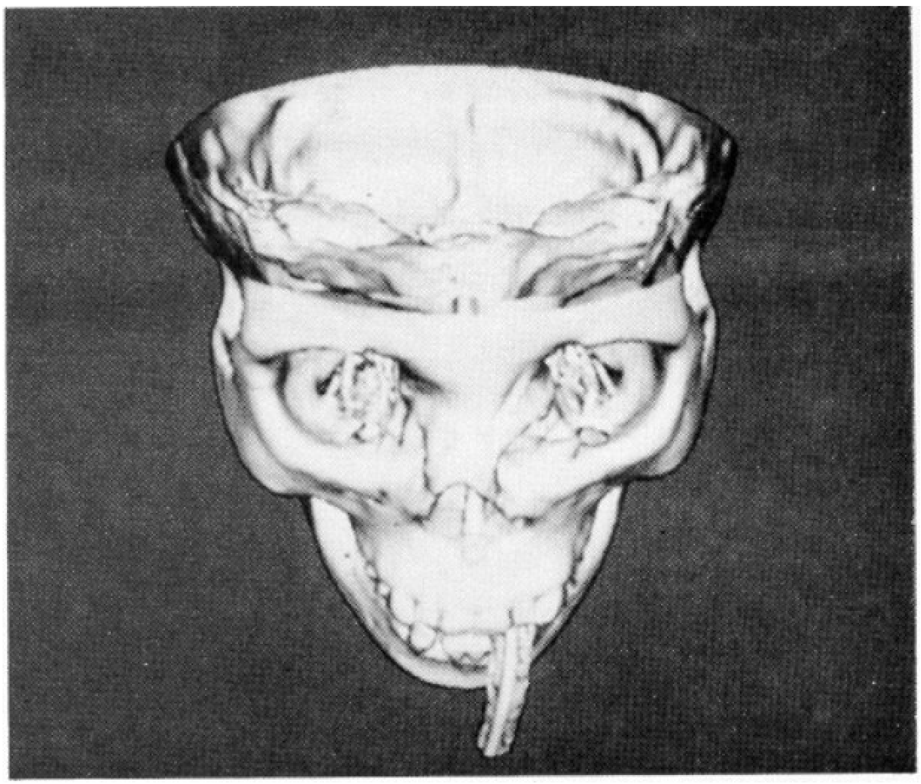
$$n_y = \frac{f(i, j, k+1) - f(i, j, k-1)}{2\Delta z}$$

(Better approximations are possible)

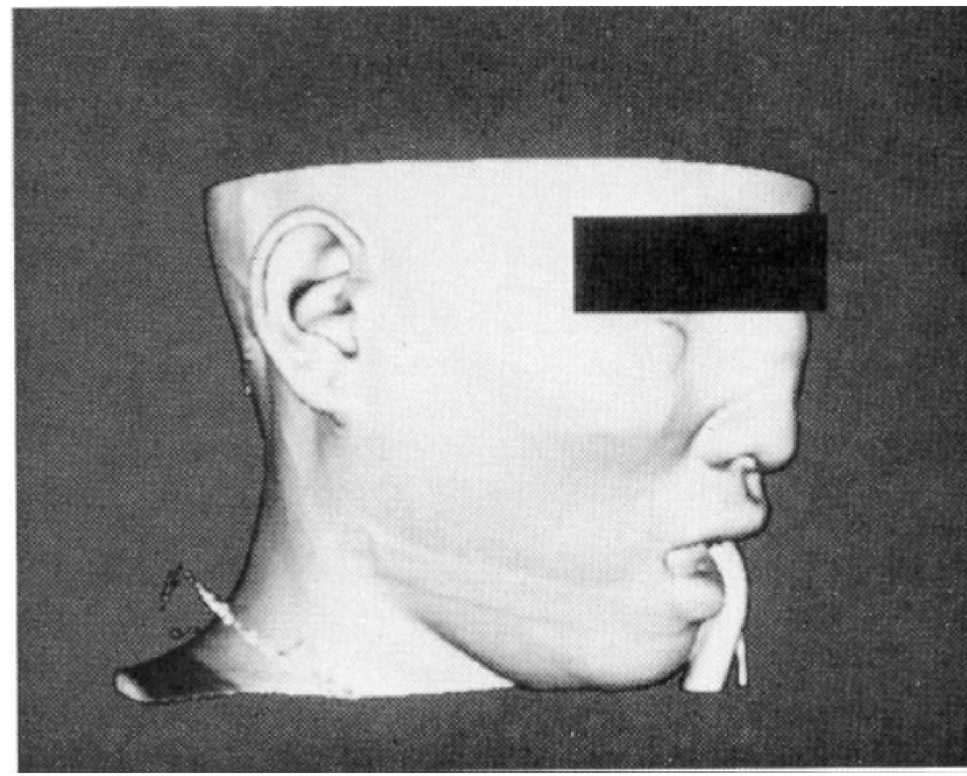
Normals at Mesh Vertices



Example: Different level sets of CT scan



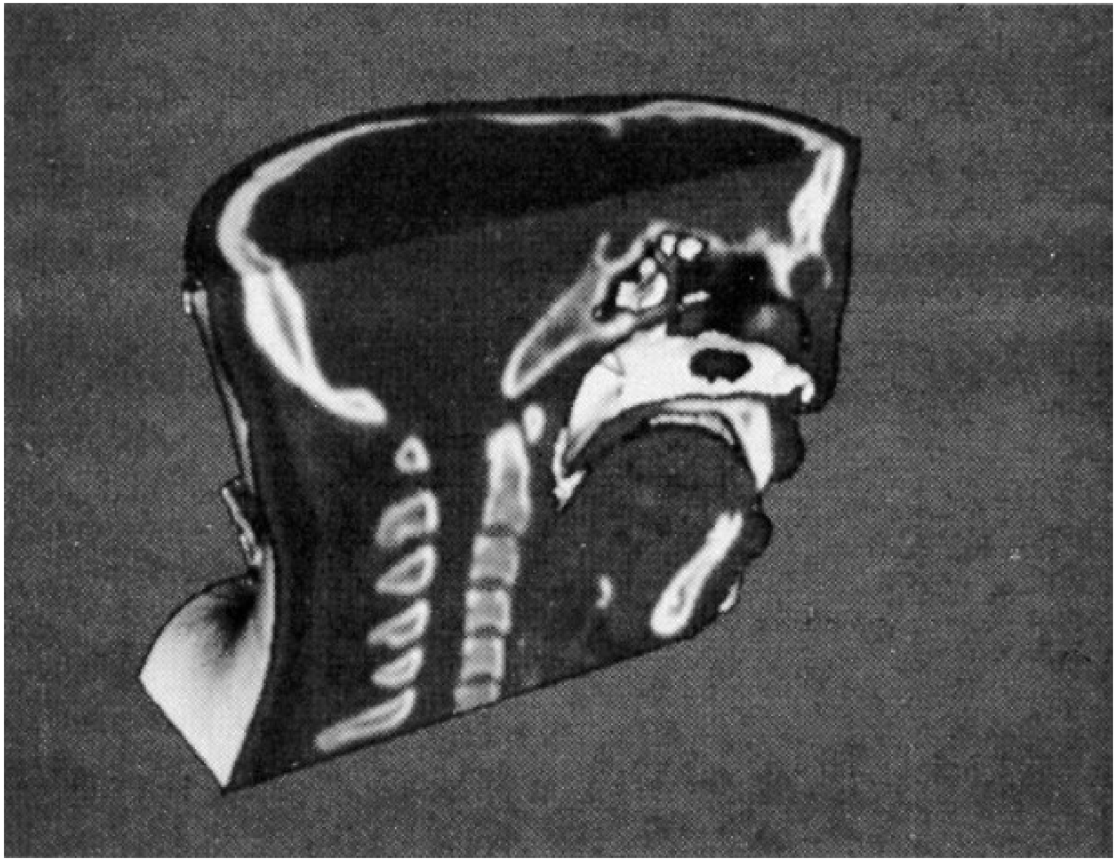
Bone surface



Soft tissue surface

Lorensen and Cline, “Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm”, SIGGRAPH '87

Example: Different level sets of CT scan

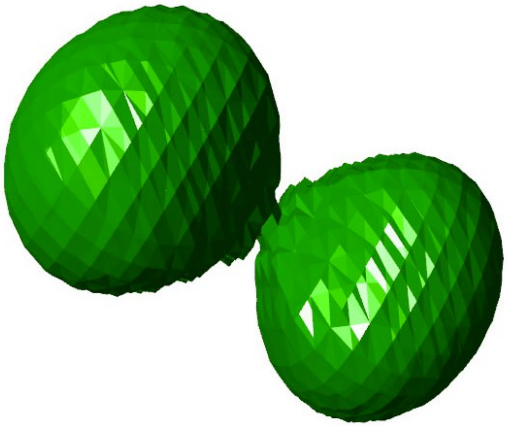


Alignment with original volumetric data

Lorensen and Cline, “Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm”, SIGGRAPH '87

Marching Cubes: Pros and Cons

- **Pros:**
  - Local computations only, so needs very little working memory and has good cache coherence
  - Works well with grid-structured input
    - E.g. medical scans
  - Simple to implement



- **Cons:**
  - No adaptive resolution, produces lots of triangles
  - Telltale patterned artifacts, since cells are cubes and output triangles are generated from a uniform grid.
  - No principled approach to resolve ambiguities