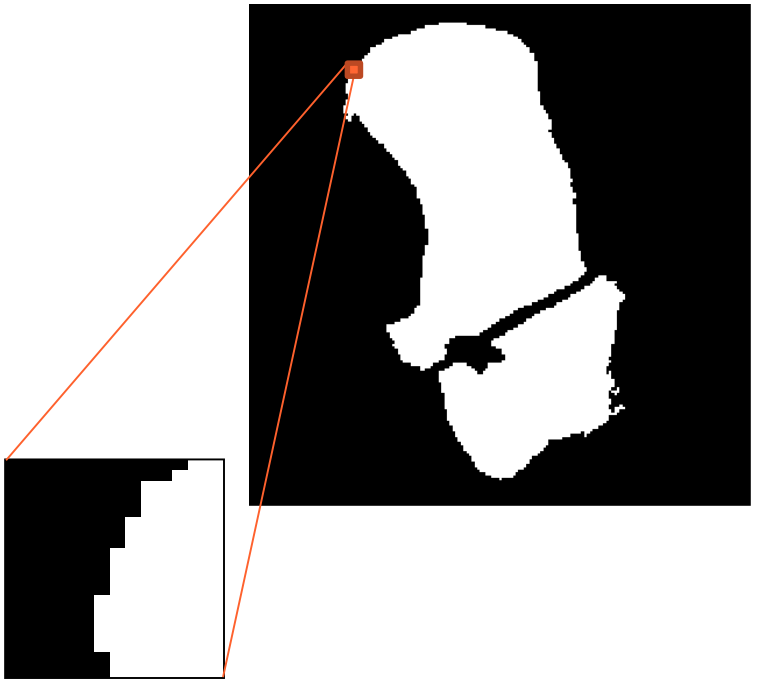
CSE 554

Lecture 4: Contouring

Fall 2016

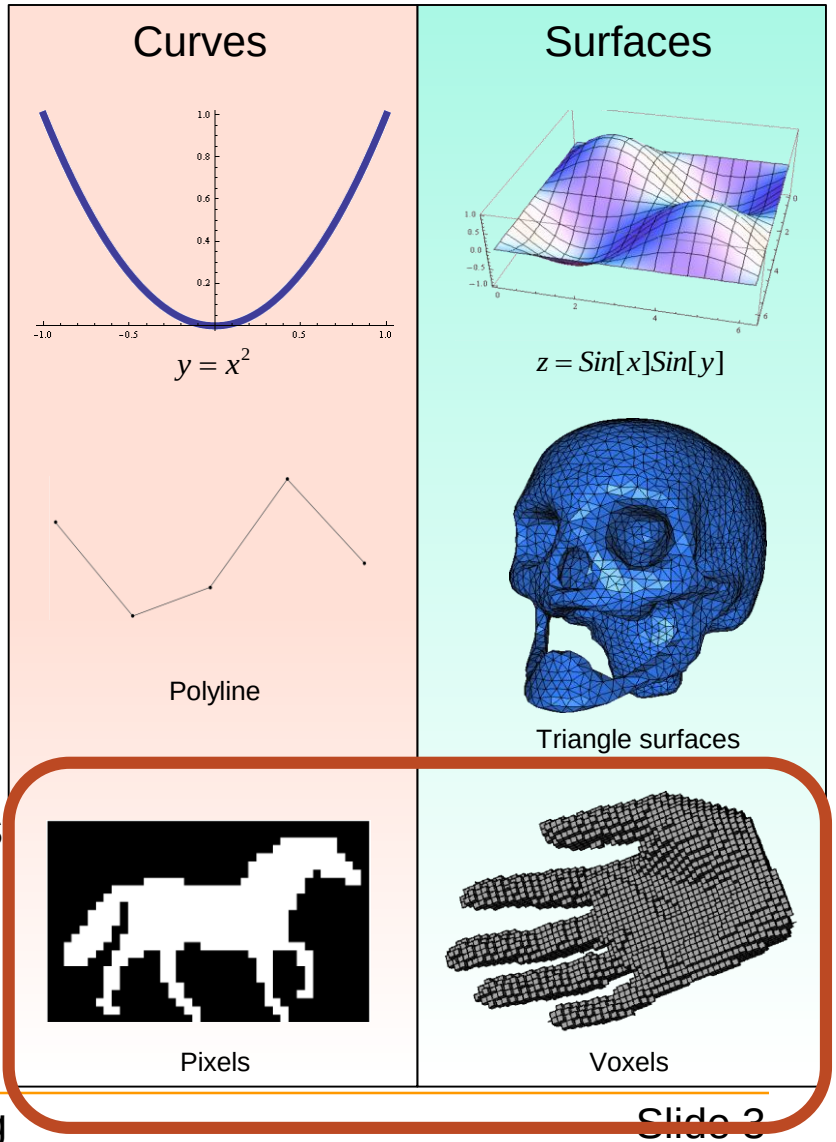
Review

- Binary pictures
  - Pros: natural geometric form for bio-medical data; easy to operate on
  - Cons:
    - Blocky boundary
    - Large memory footprint



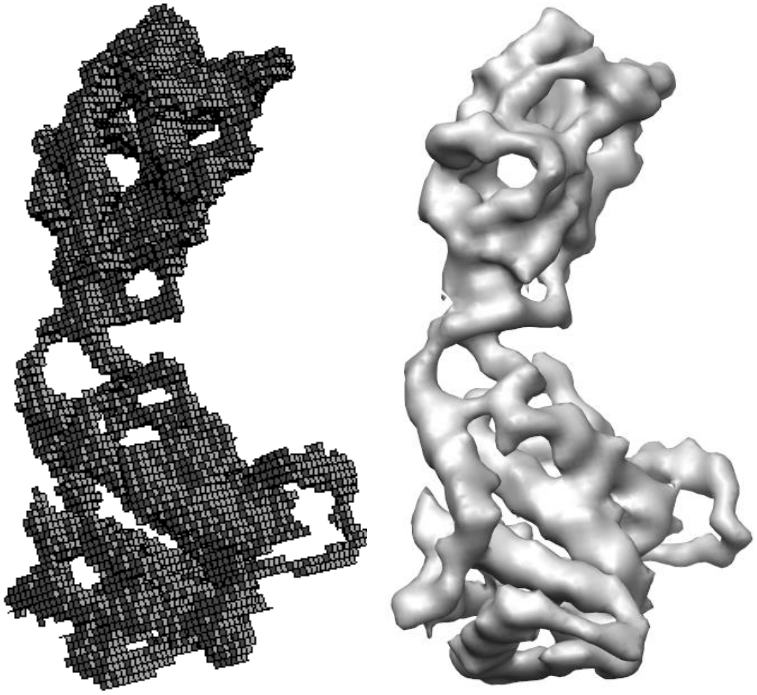
Geometric Forms

- Continuous forms
  - Defined by mathematical functions
  - E.g.: parabolas, splines, subdivision surfaces
- Discrete forms
  - Disjoint elements with connectivity relations
  - E.g.: polylines, triangle surfaces, pixels and voxels



Boundary Representations

- Polylines (2D) or meshes (3D) that tile the object boundary
  - Smoother appearance
  - Less storage (no interior elements)



Binary picture Boundary mesh

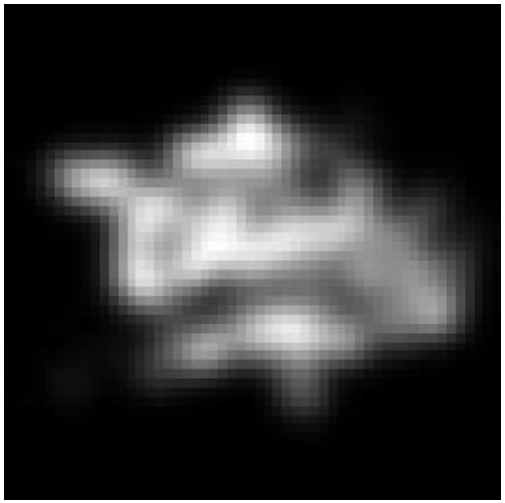
Boundary Representations

Washington University in St. Louis
School of Engineering & Applied Science

- We will cover (in a sequence of lectures):
  - Extracting a boundary from a grayscale image (volume)
  - Denoising and simplification
  - Alignment and deformation

Thresholding - Revisited

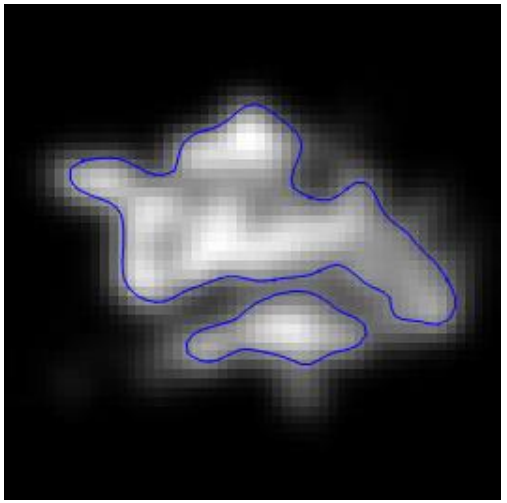
- Creates a binary picture from a grayscale image
- How to define a smooth boundary at the threshold?
  - Such boundary is known as a *contour* (or iso-curve, iso-surface, etc.)



Grayscale image



Thresholded binary picture



Boundary curve

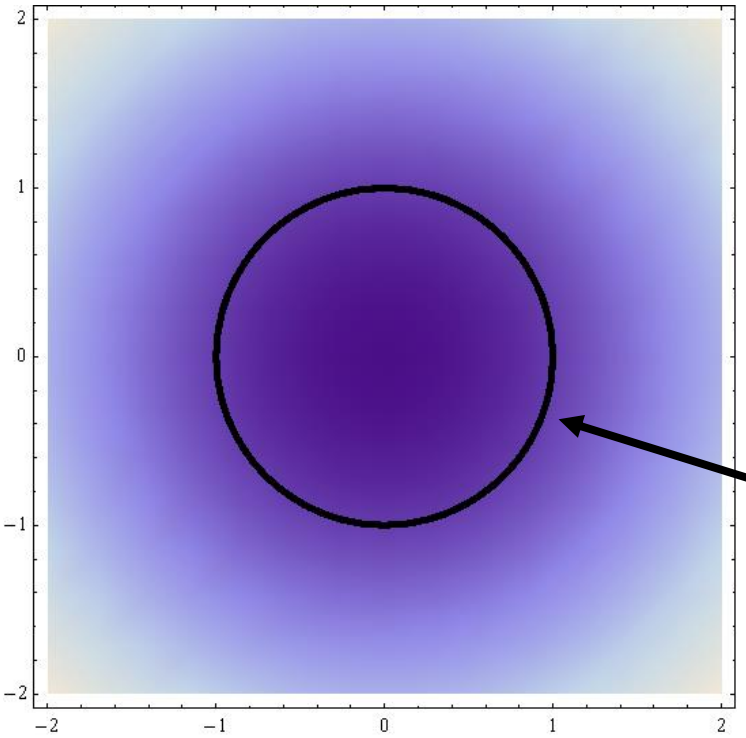
Definition

- Given a continuous function f defined over the space,
- A contour at iso-value c is the set of all points where f evaluates to be c

Examples

- Contours of 2D functions (iso-curves)

$$\{\{x, y\} \mid f(x, y) = c\}$$



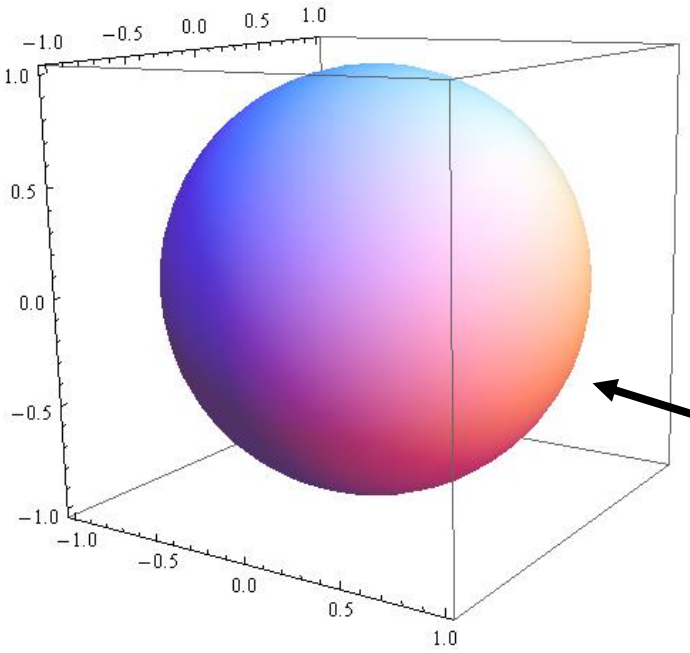
$$f(x, y) = x^2 + y^2$$

$$f(x, y) = 1$$

Examples

- Contours of 3D functions (iso-surfaces)

$$\{\{x, y, z\} \mid f(x, y, z) = c\}$$

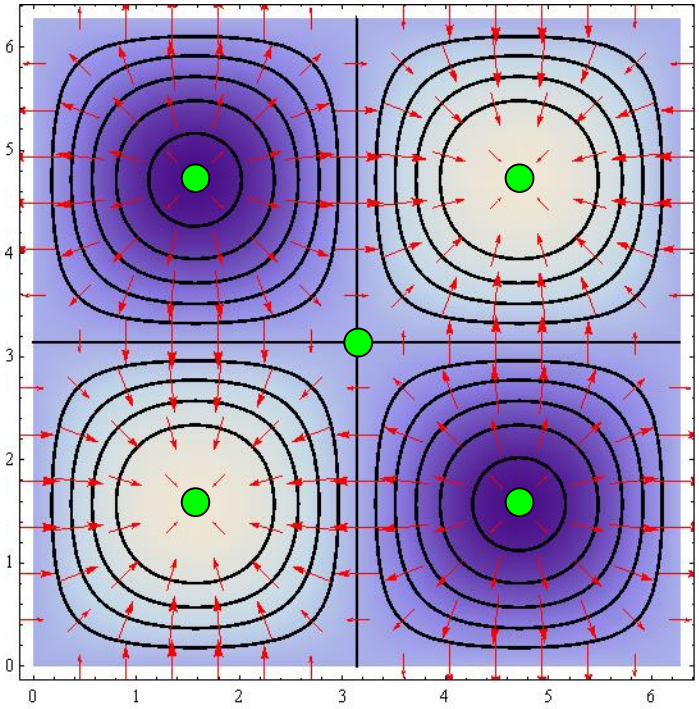


$$f(x, y, z) = x^2 + y^2 + z^2$$

$$f(x, y, z) = 1$$

Contour Properties

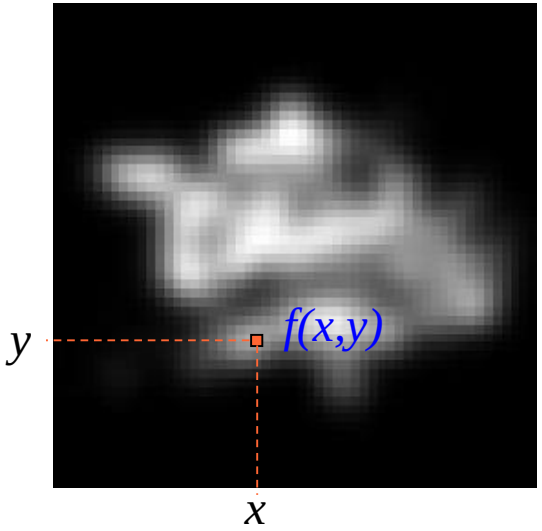
- Closed
  - With a well-defined inside and outside
    - Inside: points above the iso-value
    - Outside: points below the iso-value
- In general, a manifold
  - A non-degenerate curve (surface) without branching or boundaries
    - Except at critical points (l*ocal maxima, minima, saddle)*
- Orthogonal to gradient directions
  - Critical points: where gradient is zero

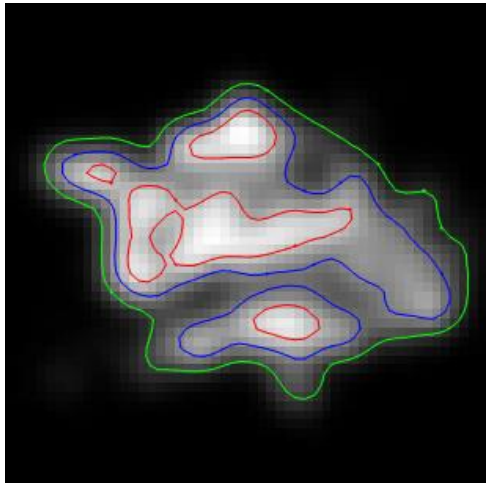


Black curves: contours at multiple iso-values
Green dots: critical points
Red arrows: gradient directions

Discrete Iso-contours

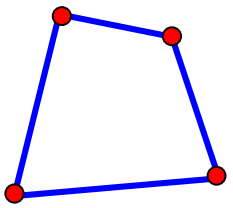
- Consider an image as a sampling of some continuous function f
  - We seek discrete approximations of the continuous contours of f



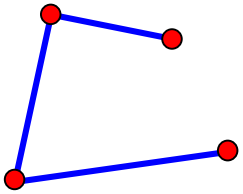


"Good" Approximations

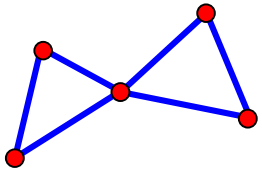
- Closed (with inside and outside)
  - Polyline: a vertex is shared by even # of edges
- Manifold
  - Polyline: a vertex is shared by 2 edges
- Non-intersecting



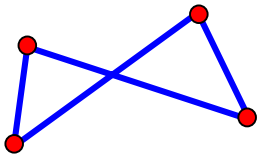
Closed
Manifold
Non-intersecting



Open
Non-manifold
Non-intersecting



Closed
Non-manifold
Non-intersecting

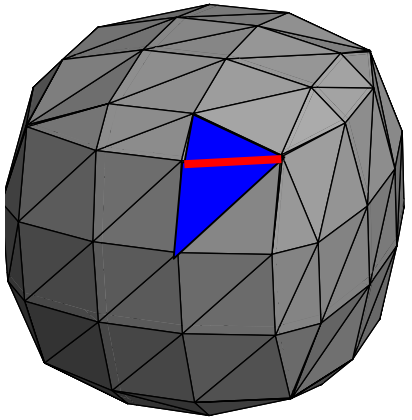


Closed
Manifold
Intersecting

"Good" Approximations

Washington University in St. Louis
School of Engineering & Applied Science

- Closed (with inside and outside)
  - Polyline: a vertex is shared by even # of edges
  - Mesh: an edge is shared by even # of polygons
- Manifold
  - Polyline: a vertex is shared by 2 edges
  - Mesh: an edge is shared by 2 polygons, and a vertex is contained in a ring of polygons
- Non-intersecting

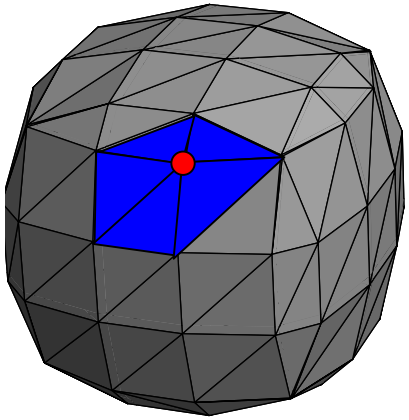


A closed, manifold, non-intersecting triangular mesh

"Good" Approximations

Washington University in St. Louis
School of Engineering & Applied Science

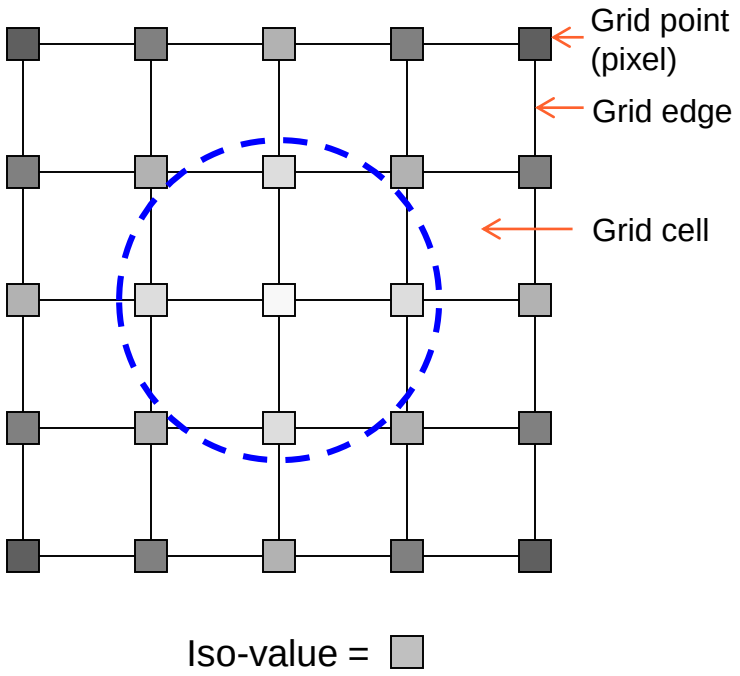
- Closed (with inside and outside)
  - Polyline: a vertex is shared by even # of edges
  - Mesh: an edge is shared by even # of polygons
- Manifold
  - Polyline: a vertex is shared by 2 edges
  - Mesh: an edge is shared by 2 polygons, and a vertex is contained in a ring of polygons
- Non-intersecting



A closed, manifold, non-intersecting triangular mesh

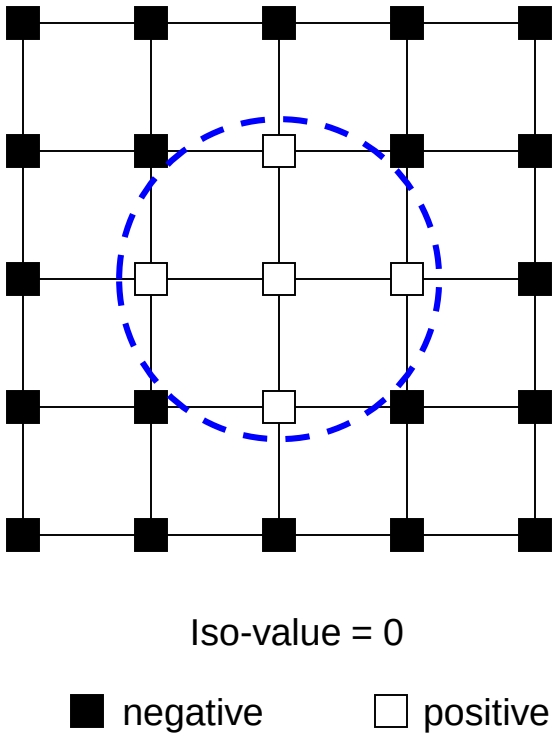
Contouring (On A Grid)

- Input
  - A grid where each grid point (pixel or voxel) has a value (color)
  - An iso-value (threshold)
- Output
  - A closed, manifold, non-intersecting polyline (2D) or mesh (3D) that separates grid points above the iso-value from those that are below the iso-value.



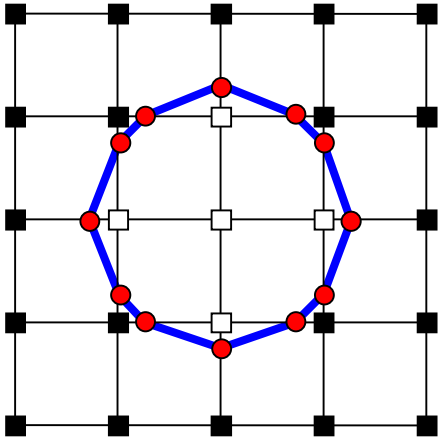
Contouring (On A Grid)

- Input
  - A grid where each grid point (pixel or voxel) has a value (color)
  - An iso-value (threshold)
- Output
  - Equivalently, we extract the zero-contour (separating negative from positive) after subtracting the iso-value from the grid points

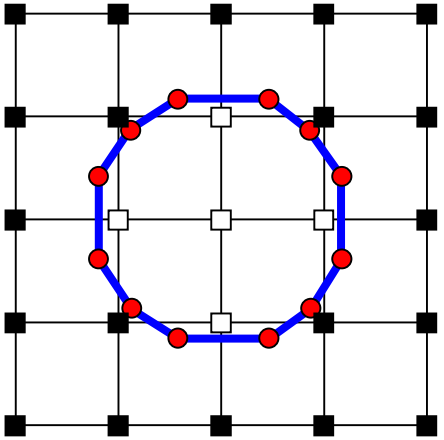


Algorithms

- Primal methods
  - Marching Squares (2D), Marching Cubes (3D)
  - Placing vertices on grid edges

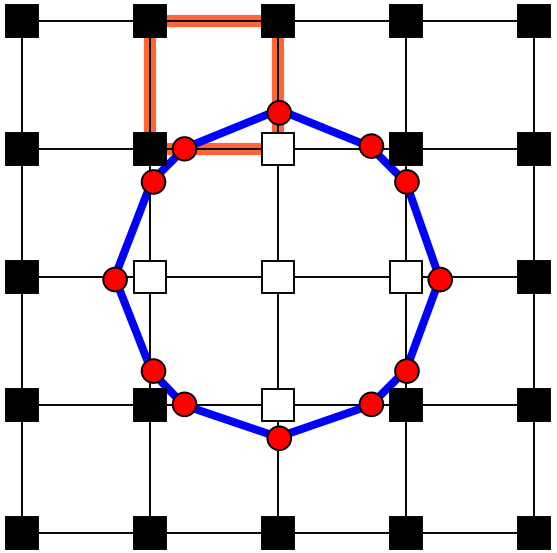


- Dual methods
  - Dual Contouring (2D,3D)
  - Placing vertices in grid cells



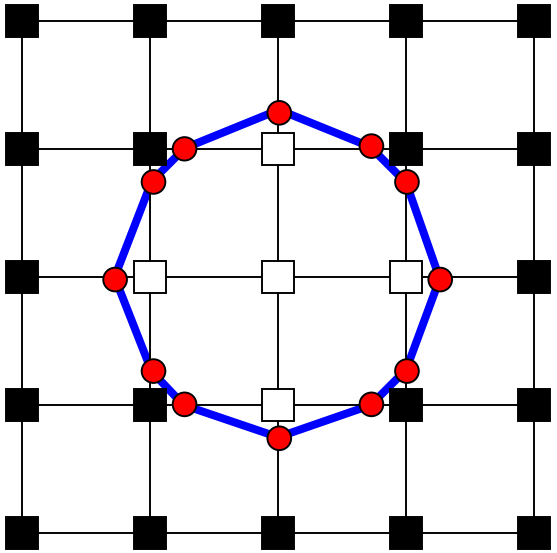
Marching Squares (2D)

- For each grid cell with a sign change
  - Create one vertex on each grid edge with a sign change
  - Connect vertices by lines



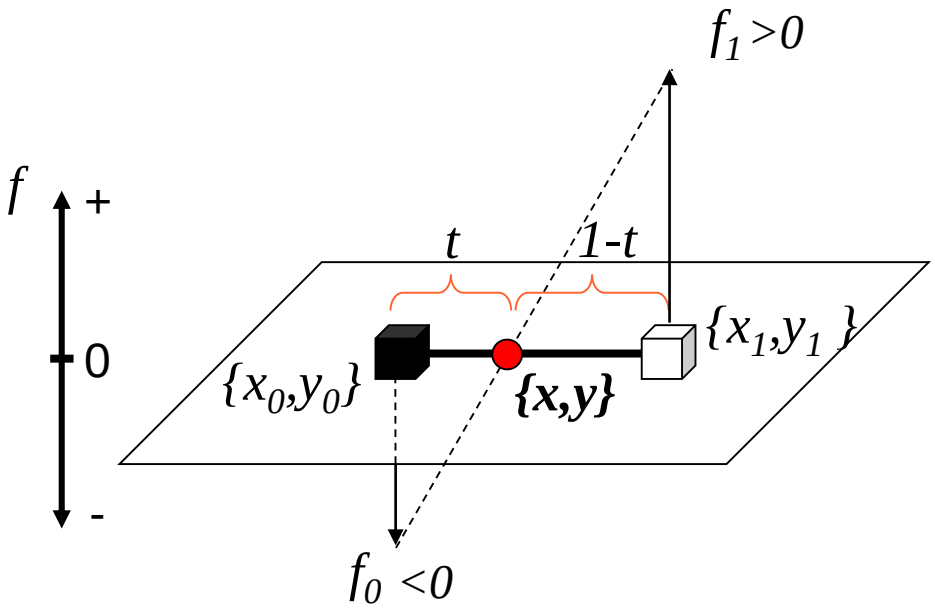
Marching Squares (2D)

- For each grid cell with a sign change
  - Create one vertex on each grid edge with a sign change
  - Connect vertices by lines



Marching Squares (2D)

- Creating vertices: linear interpolation
  - Assuming the underlying, continuous function is linear on the grid edge
  - Linearly interpolate the positions of the two grid points



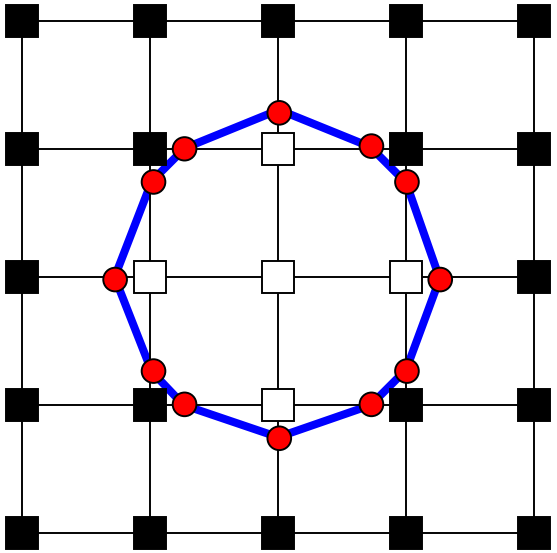
$$t = \frac{f_0}{f_0 - f_1}$$

$$x = x_0 + t(x_1 - x_0)$$

$$y = y_0 + t(y_1 - y_0)$$

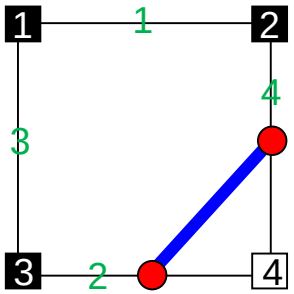
Marching Squares (2D)

- For each grid cell with a sign change
  - Create one vertex on each grid edge with a sign change
  - Connect vertices by lines



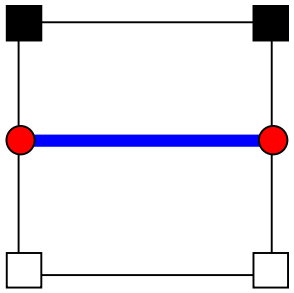
Marching Squares (2D)

- Connecting vertices by lines
  - Lines shouldn't intersect
  - Each vertex is used once
    - So that it will be used exactly twice by the two cells incident on the edge
- Two approaches
  - Do a walk around the grid cell
    - Connect consecutive pair of vertices
  - Or, using a pre-computed look-up table
    - 2^4=16 sign configurations
    - For each sign configuration, it stores the indices of the grid edges whose vertices make up the lines.



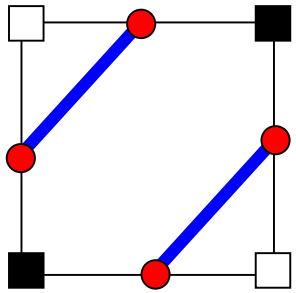
Key: 0 0 0 1

Data: {{2,4}}



Key: 0 0 1 1

Data: {{3,4}}

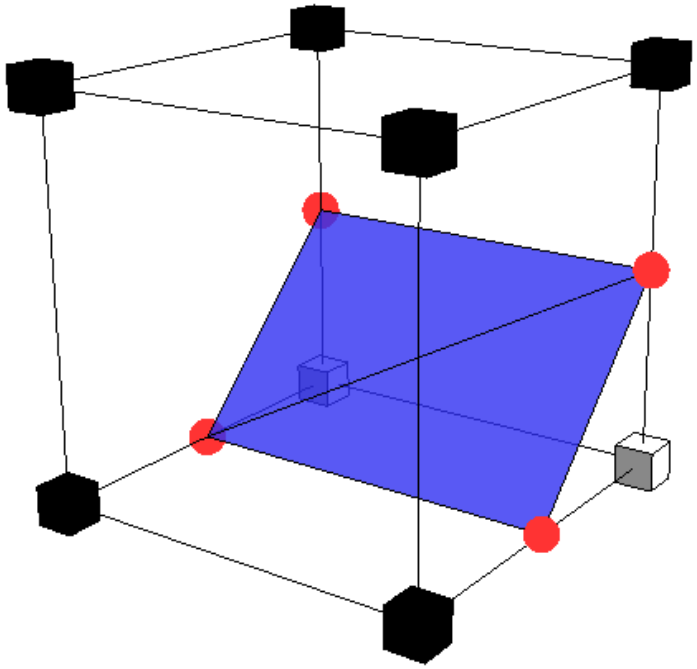


Key: 1 0 0 1

Data: {{1,3}, {2,4}}

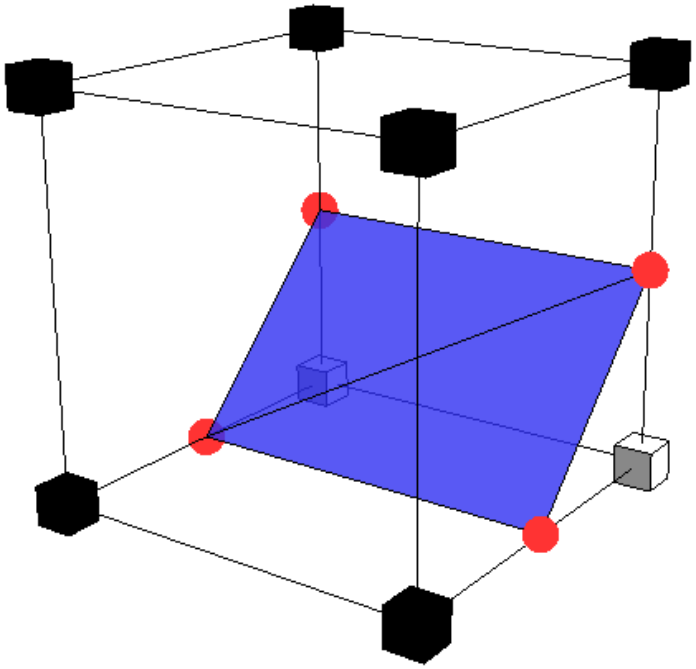
Marching Cubes (3D)

- For each grid cell with a sign change
  - Create one vertex on each grid edge with a sign change (using linear interpolation)
  - Connect vertices into triangles



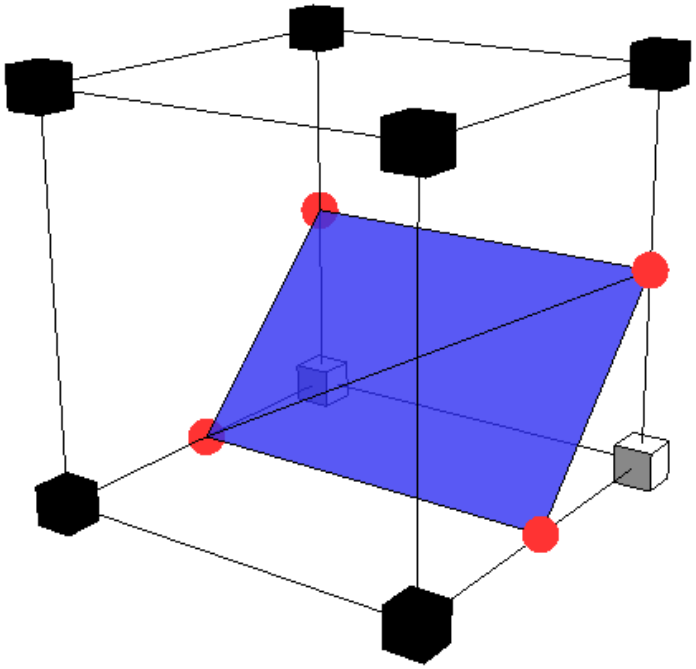
Marching Cubes (3D)

- For each grid cell with a sign change
  - Create one vertex on each grid edge with a sign change (using linear interpolation)
  - Connect vertices into triangles



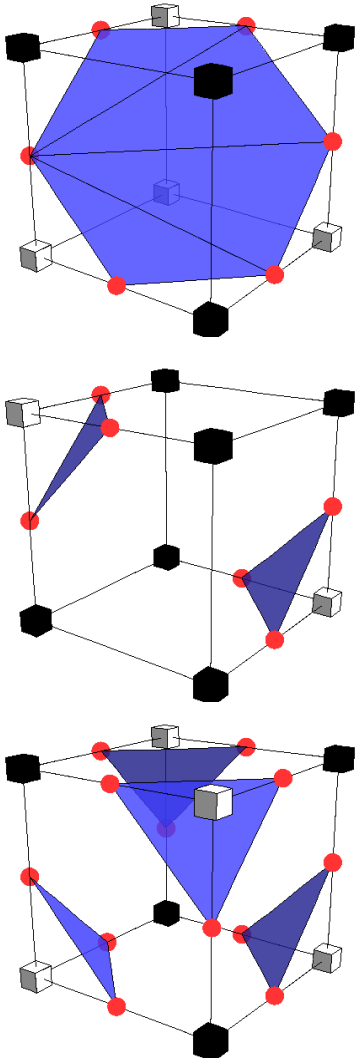
Marching Cubes (3D)

- Connecting vertices by triangles
  - Triangles shouldn't intersect
  - To be a closed manifold:
    - Each vertex used by a triangle "fan"
    - Each mesh edge used by 2 triangles (if inside grid cell) or 1 triangle (if on a grid face)



Marching Cubes (3D)

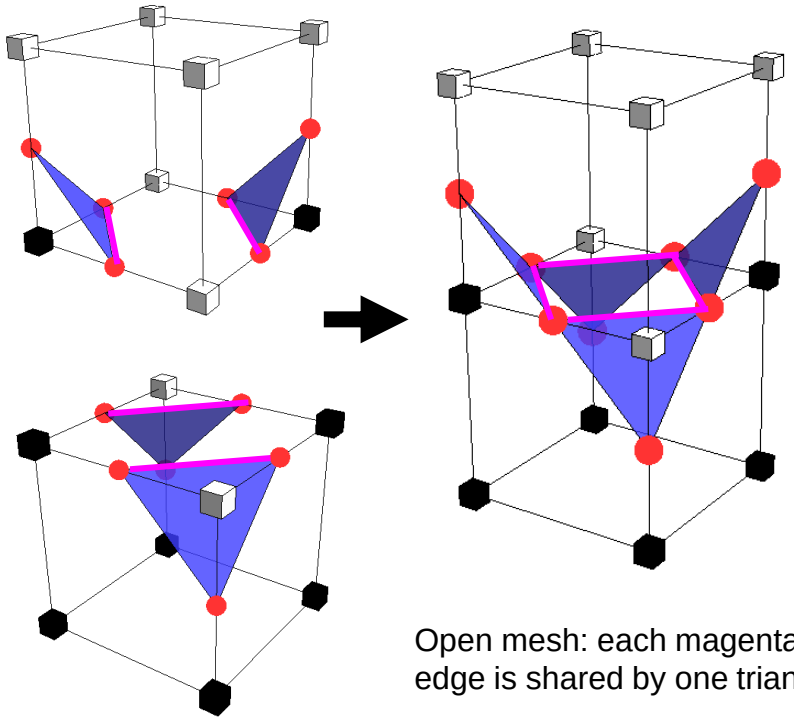
- Connecting vertices by triangles
  - Triangles shouldn't intersect
  - To be a closed manifold:
    - Each vertex used by a triangle "fan"
    - Each mesh edge used by 2 triangles (if inside grid cell) or 1 triangle (if on a grid face)



Marching Cubes (3D)

Washington University in St. Louis
School of Engineering & Applied Science

- Connecting vertices by triangles
  - Triangles shouldn't intersect
  - To be a closed manifold:
    - Each vertex used by a triangle "fan"
    - Each mesh edge used by 2 triangles (if inside grid cell) or 1 triangle (if on a grid face)

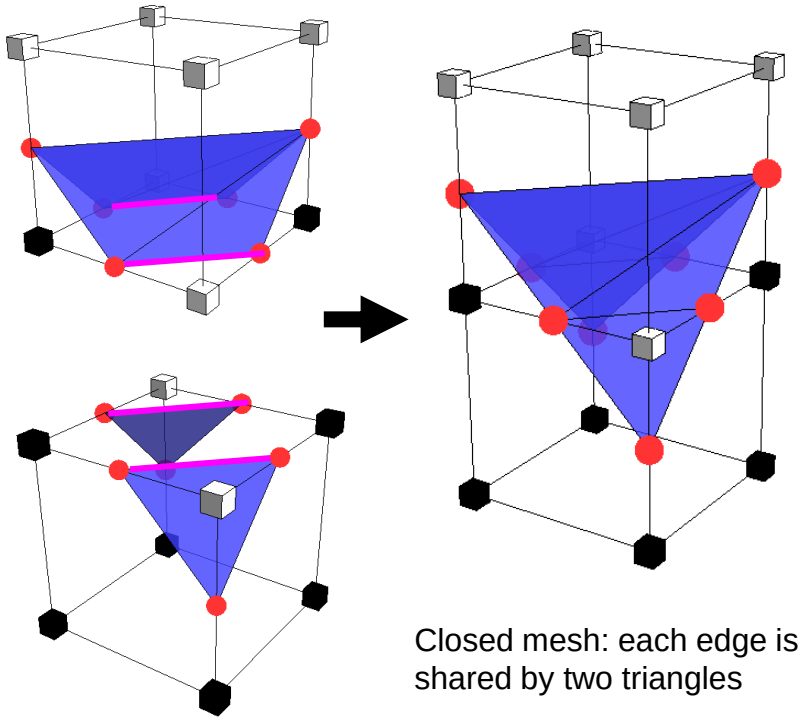


Open mesh: each magenta edge is shared by one triangle

Marching Cubes (3D)

Washington University in St. Louis
School of Engineering & Applied Science

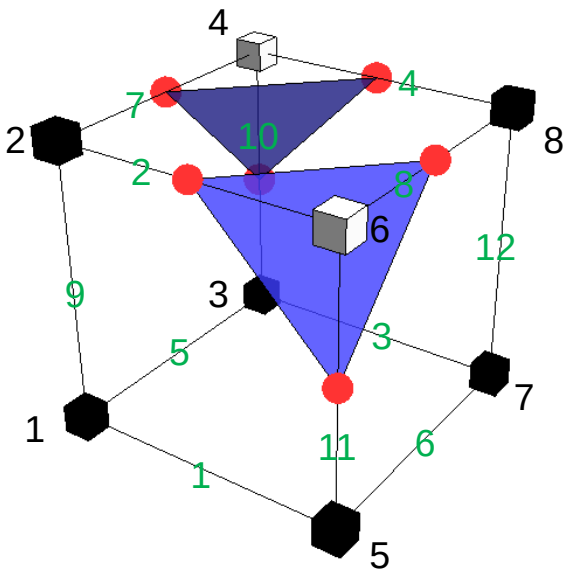
- Connecting vertices by triangles
  - Triangles shouldn't intersect
  - To be a closed manifold:
    - Each vertex used by a triangle "fan"
    - Each mesh edge used by 2 triangles (if inside grid cell) or 1 triangle (if on a grid face)
    - Each mesh edge on the grid face is shared between adjacent cells



Closed mesh: each edge is shared by two triangles

Marching Cubes (3D)

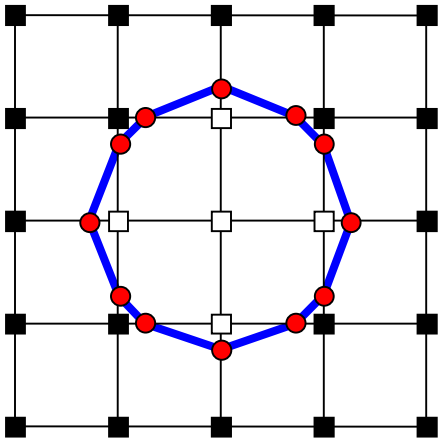
- Connecting vertices by triangles
  - Triangles shouldn't intersect
  - To be a closed manifold:
    - Each vertex used by a triangle "fan"
    - Each mesh edge used by 2 triangles (if inside grid cell) or 1 triangle (if on a grid face)
    - Each mesh edge on the grid face is shared between adjacent cells
- Look-up table
  - 2^8=256 sign configurations
  - For each sign configuration, it stores indices of the grid edges whose vertices make up the triangles



Sign: "0 0 0 1 0 1 0 0"
Triangles: {{2,8,11},{4,7,10}}

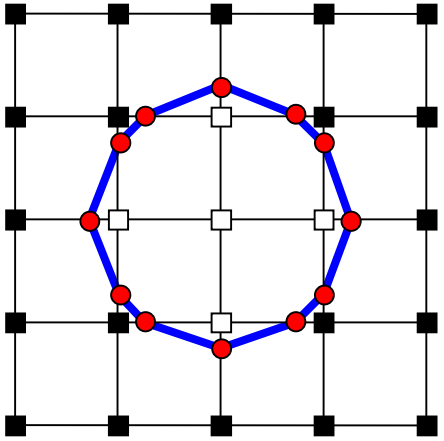
Implementation Notes

- Avoid computing one vertex multiple times
  - Compute the vertex location once, and store it in a hash table
- When the grid point's value is same as the iso-value
  - Treat it either as "above" or "below", but be consistent.

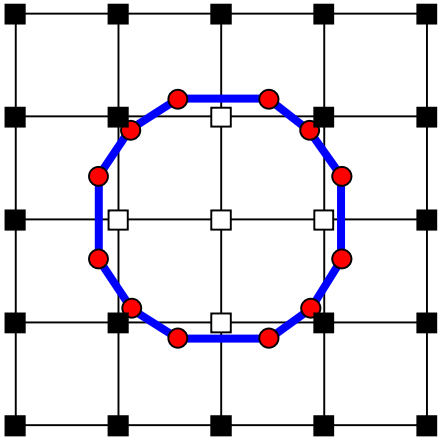


Algorithms

- Primal methods
  - Marching Squares (2D), Marching Cubes (3D)
  - Placing vertices on grid edges

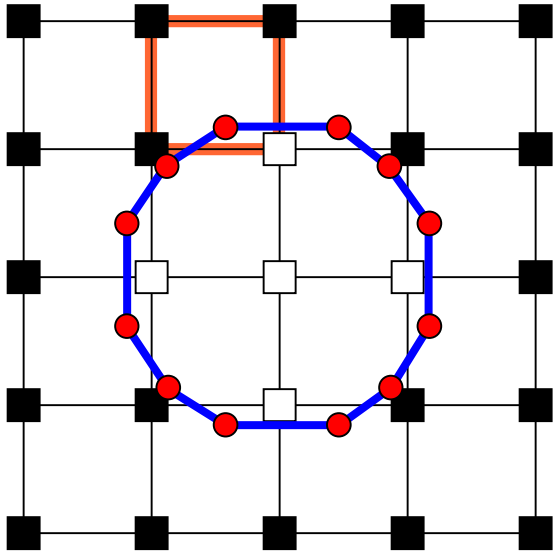


- Dual methods
  - Dual Contouring (2D,3D)
  - Placing vertices in grid cells



Dual Contouring (2D)

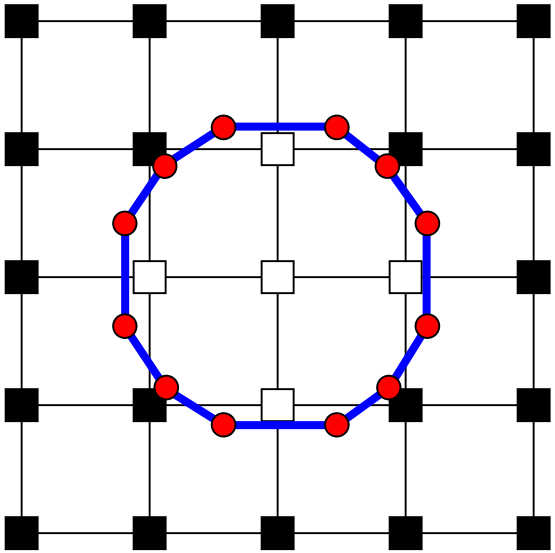
- For each grid cell with a sign change
  - Create one vertex
- For each grid edge with a sign change
  - Connect the two vertices in the adjacent cells with a line segment



Dual Contouring (2D)

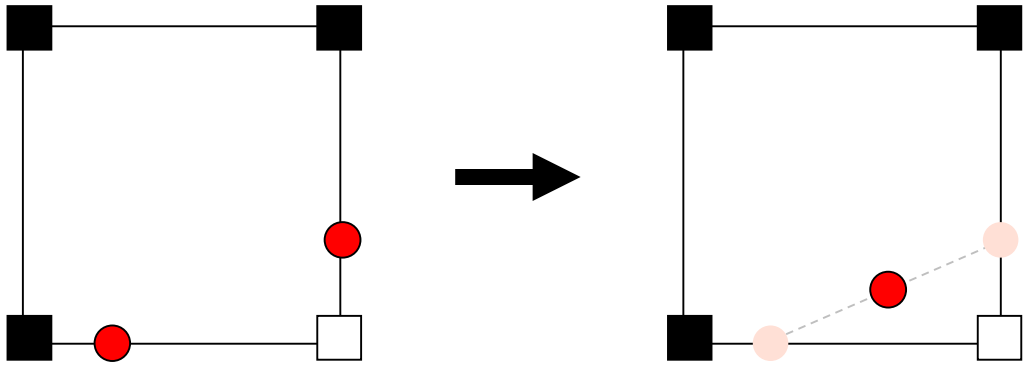
Washington University in St. Louis
School of Engineering & Applied Science

- For each grid cell with a sign change
  - Create one vertex
- For each grid edge with a sign change
  - Connect the two vertices in the adjacent cells with a line segment



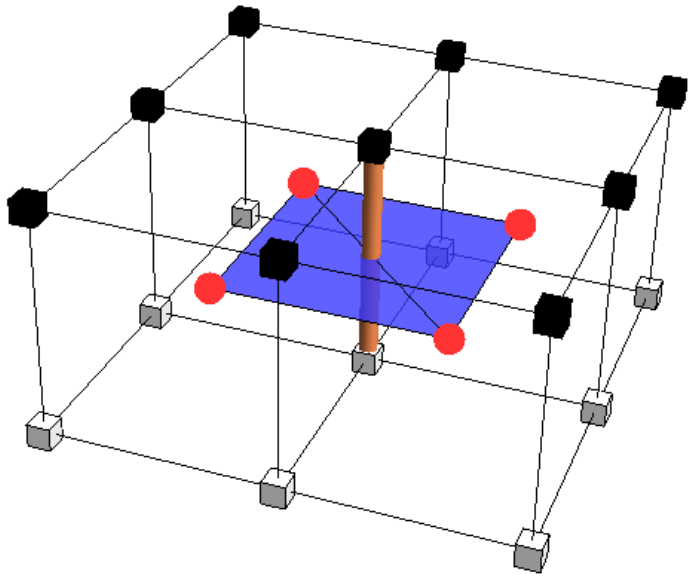
Dual Contouring (2D)

- Creating the vertex within a cell
  - Compute one point on each grid edge with a sign change (by linear interpolation)
    - There could be more than two sign-changing edges, so >2 points possible
  - Take the centroid of these points



Dual Contouring (3D)

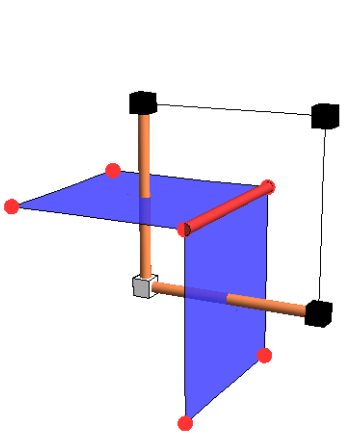
- For each grid cell with a sign change
  - Create one vertex (same way as 2D)
- For each grid edge with a sign change
  - Create a quad (or two triangles) connecting the four vertices in the adjacent grid cubes
  - No look-up table is needed!



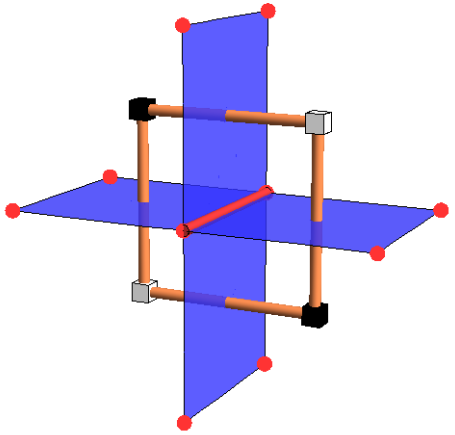
Dual Contouring: Discussion

Washington University in St. Louis
School of Engineering & Applied Science

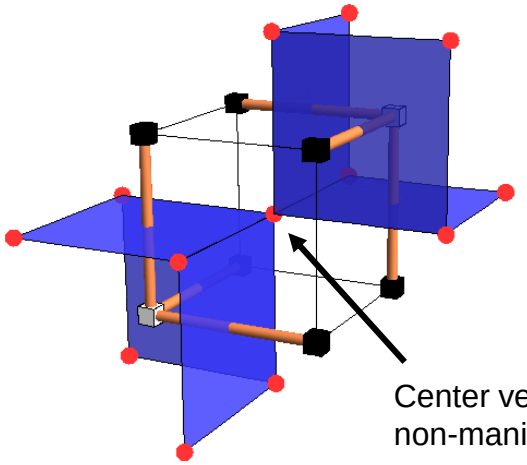
- Closed
  - Each mesh edge is shared by even number of quads
- Possibly non-manifold
  - An edge may be shared by 4 quads
  - A vertex may be shared by 2 rings of quads
- Can be fixed
  - But with more effort (e.g., multiple vertices per cell)



Red edge is shared by 2 quads

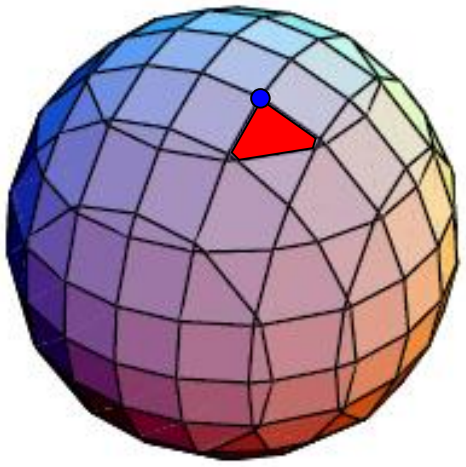


Red edge is shared by 4 quads (non-manifold)

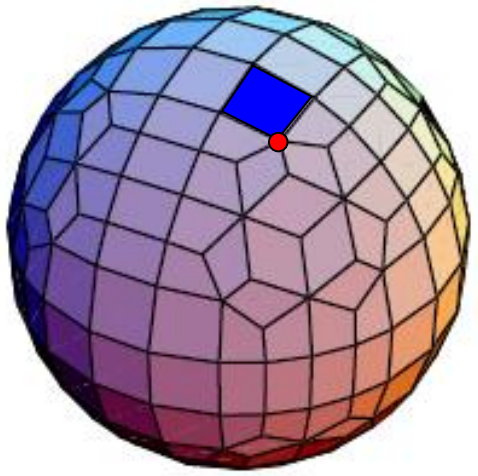


Duality

- The two outputs have a dual structure
  - Vertices and quads of Dual Contouring correspond (roughly) to un-triangulated polygons and vertices produced by Marching Cubes



Marching Cubes



Dual Contouring

Primal vs. Dual

- Marching Cubes
  - ✓ Always manifold
  - × Requires look-up table in 3D
  - × Often generates thin and tiny polygons



Marching Cubes

- Dual Contouring
  - × Can be non-manifold
  - ✓ No look-up table needed
  - ✓ Generates better-shaped polygons

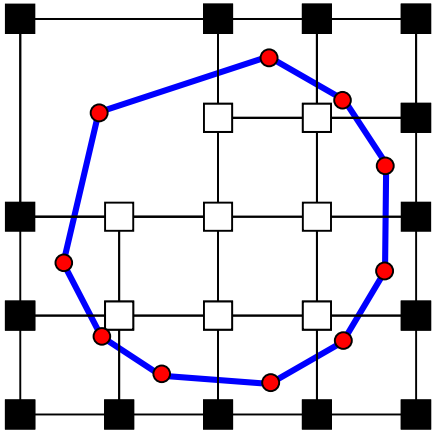


Dual Contouring

Primal vs. Dual

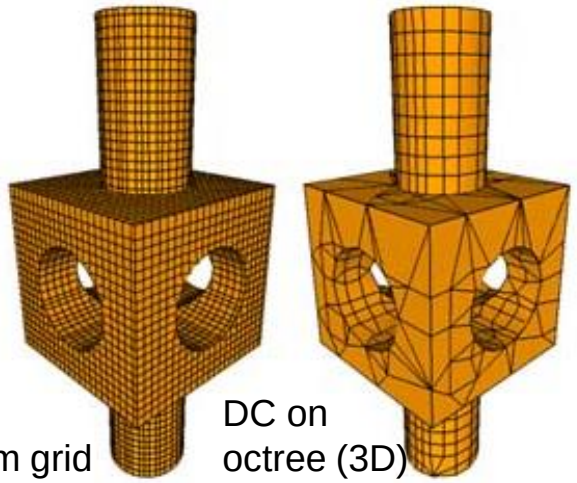
Washington University in St. Louis
School of Engineering & Applied Science

- Marching Cubes
  - ✓ Always manifold
  - × Requires look-up table in 3D
  - × Often generates thin and tiny polygons
  - × Restricted to uniform grids



DC on a Quadtree (2D)

- Dual Contouring
  - × Can be non-manifold
  - ✓ No look-up table needed
  - ✓ Generates better-shaped polygons
  - ✓ Can be applied to any type of grid



DC on uniform grid

DC on octree (3D)

Further Readings

- Marching Cubes:
  - "*Marching cubes: A high resolution 3D surface construction algorithm*", by Lorensen and Cline (1987)
    - >11000 citations on Google Scholar
  - "*A survey of the marching cubes algorithm*", by Newman and Yi (2006)

- Dual Contouring:
  - "*Dual contouring of hermite data*", by Ju et al. (2002)
    - >500 citations on Google Scholar
  - "*Manifold dual contouring*", by Schaefer et al. (2007)