Summary

- Topics:

1. Bio-medical imaging
2. Image processing
3. Geometric modeling and computer graphics
4. Mesh generation
   - Marching Cubes/Dual Contouring
   - Tri/Tet Meshing
   - Quad/Hex Meshing
   - Quality Improvement
5. Computational mechanics
6. Bio-medical applications

Topic 3: Geometric Modeling and Computer Graphics – Visualization

Jessica Zhang
Department of Mechanical Engineering
Courtesy Appointment in Biomedical Engineering
Carnegie Mellon University
jessicaz@andrew.cmu.edu
http://www.andrew.cmu.edu/user/jessicaz

Visualization

- Large datasets visualization in scientific and medical applications
  - Scalar field visualization, where we have a single value at each spatial point.
  - Flow visualization, where we have a velocity at each spatial point.

Data and Geometry

- Scientific visualization is the merging of data with the display of geometric objects through computer graphics.
- Volumetric data: static or dynamic
  - Scalar visualization: where we are given a 3D volume of scalars, such as *x*-ray densities in a medical application.
  - Vector visualization: where we start with a volume of vector data, such as the velocity at each point in a fluid.
  - Tensor visualization: where we have a matrix of data at each point, such as the stresses with a mechanical part.
- Difficulties:
  - The large size of the datasets (3D, dynamic), e.g., the visible-woman data: 1734x512x512.
  - Volume data are not geometric objects, we must translate these data to create and manipulate geometric objects.
  - We may have multidimensional data points, e.g., vector/tensor attached to each grid point.

Displaying Implicit Functions

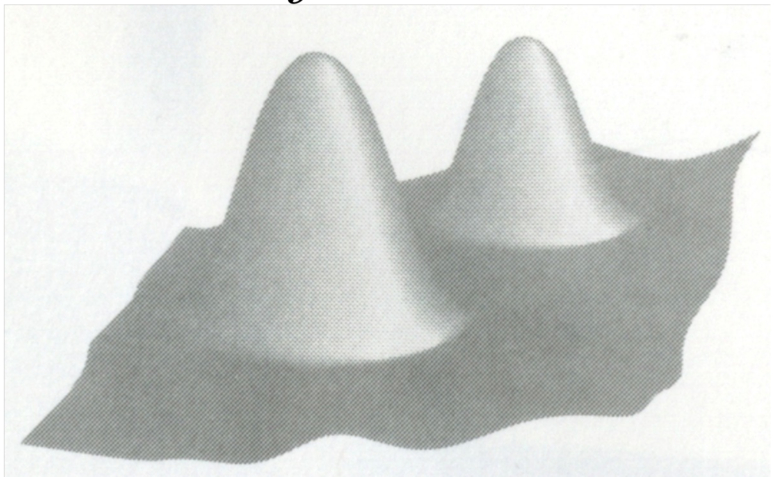
- Consider the implicit function

$$g(x, y) = 0$$

- Given an x, we cannot in general find a corresponding y;
- Given an x and a y, we can test if they are on the curve.

Height Fields and Contours

- In many applications, we have the heights given by a function of the form $z = f(x, y)$
- To find all the points that have a given height c, we have to solve the implicit equation $g(x, y) = f(x, y) - c = 0$
- Such a function determines the isosurfaces or contours of f for the isosurface value c

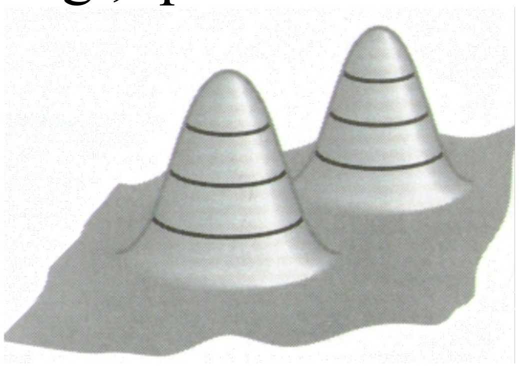


Surface described by a height field

Mechanical Engineering
Carnegie Mellon

Height Fields and Contours

- In implicit functions, although functions in this form may be known analytically, there is no general method for expressing either of the variables in terms of the other. E.g., quadric curves of the form



$$ax^2 + bxy + cy^2 + dx + ey + d = 0,$$

and the plane

$$ax + by + cz + d = 0.$$

A more complex example is the Ovals of Cassini that are the solutions of the equation

$$g(x, y) = (x^2 + y^2 + a^2)^2 - 4a^2x^2 - b^4.$$

For a given value of c, the function g describes a curve that corresponds to a constant z in the equation of f. This curve is called the contour curve, corresponding to the contour value c.

Contour Plots

If the function f is known analytically, we are looking for curves that satisfy

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

for a particular value of c. Often, we display a set of such curves corresponding to a set of values of c. This problem is an implicit function-display problem, where

$$g(x, y) = f(x, y) - c = 0.$$

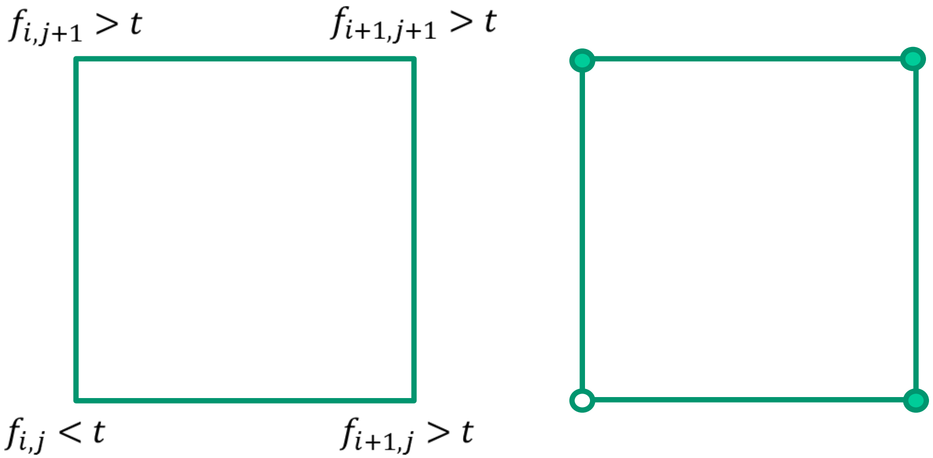
For example, if

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

If c is positive, each contour is a circle of radius $\sqrt{c}$. If we start with samples of f, then we need a technique to extract the contour from discrete sampling data, Marching Squares, which is a special case of the Marching Cubes method.

Marching Squares

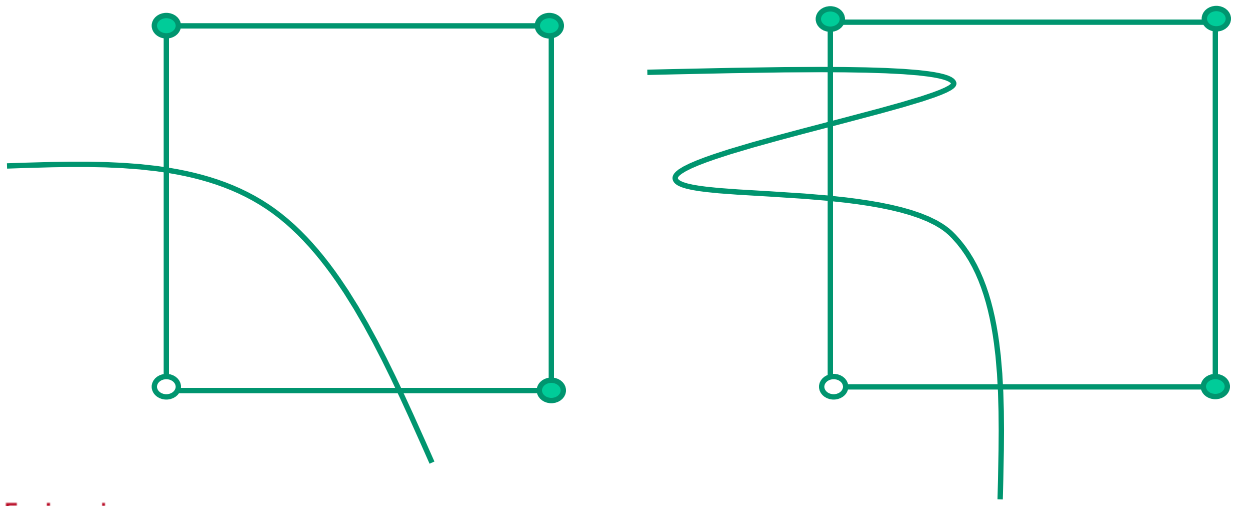
- Display isocurves or contours for functions $f(x, y) = t$
- Sample $f(x, y)$ on a regular grid yielding samples $\{f_{ij}(x, y)\}$
- These samples can be greater than, less than, or equal to t
- Consider four samples $f_{ij}(x, y), f_{i+1,j}(x, y), f_{i+1,j+1}(x, y), f_{i,j+1}(x, y)$
- These samples correspond to the corners of a cell
- Color the corners by whether they exceed or are less than the contour value t



Mechanical Engineering
Carnegie Mellon

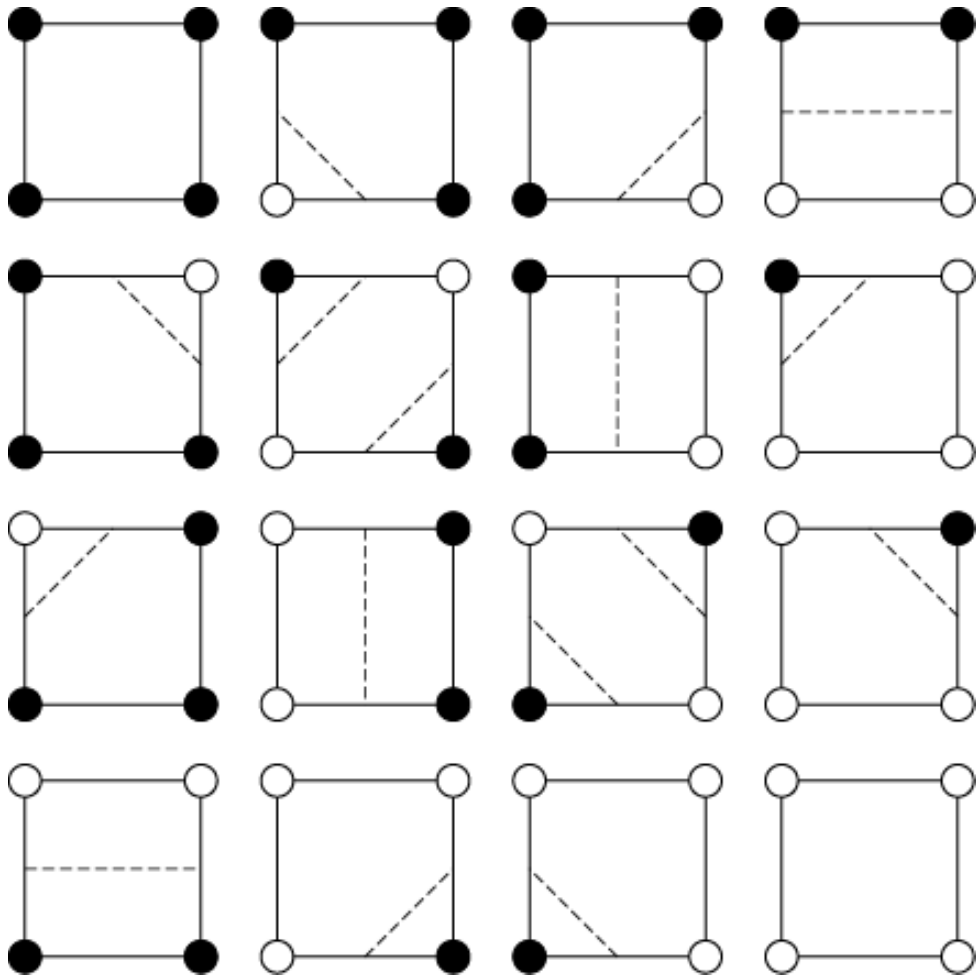
Occum's Razor

- Contour must intersect edge between a black and white vertex an odd number of times
- Pick simplest interpretation: one crossing
- Principle of Occum's Razor: if there are multiple possible explanations of a phenomenon that are consistent with the data, choose the simplest one.



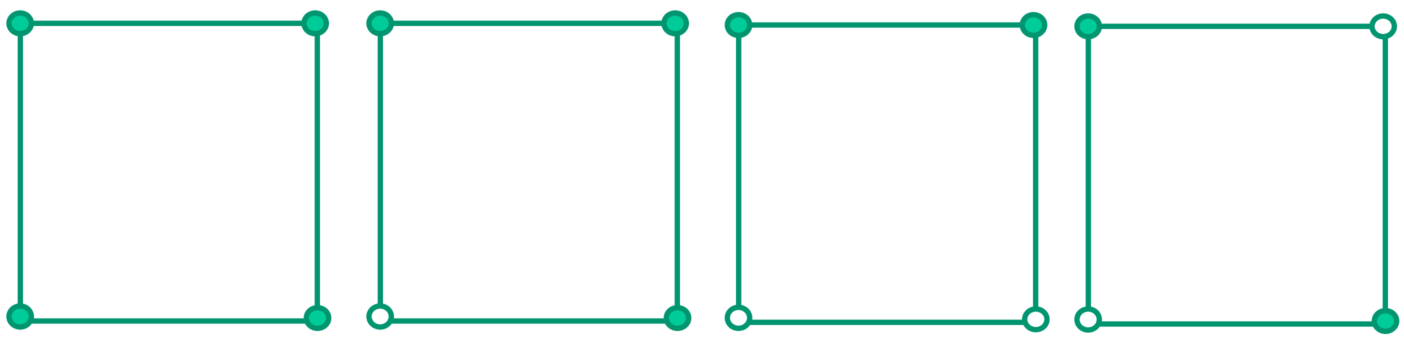
16 Cases

There are 16 (= 2^4) ways that we can color the vertices of a cell using only black and white. All could arise in our contour problem.



Unique Cases

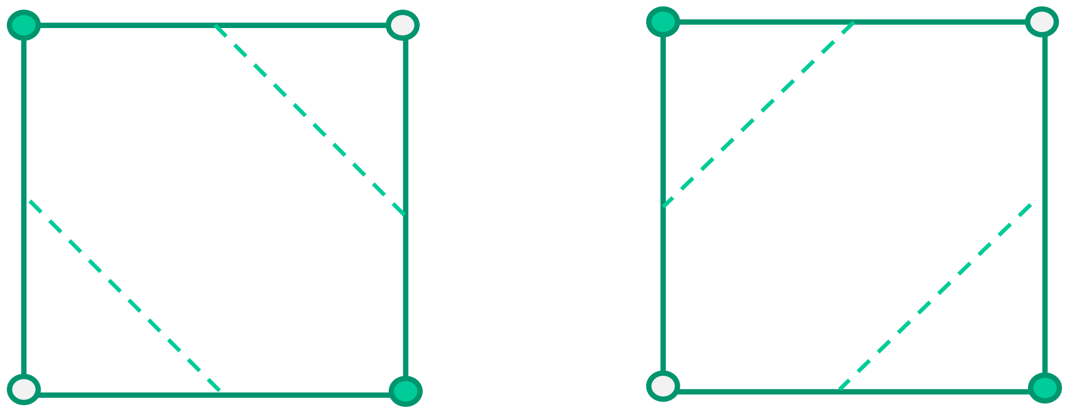
- Taking out rotational and color swapping symmetries leaves four unique cases
- All other cases can be mapped into these four unique cases
- First three have a simple interpretation



Mechanical Engineering
Carnegie Mellon

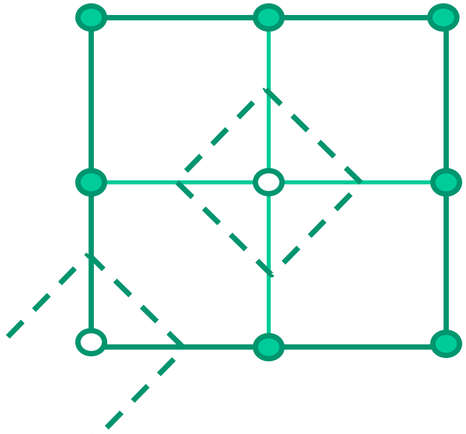
Ambiguity Problem

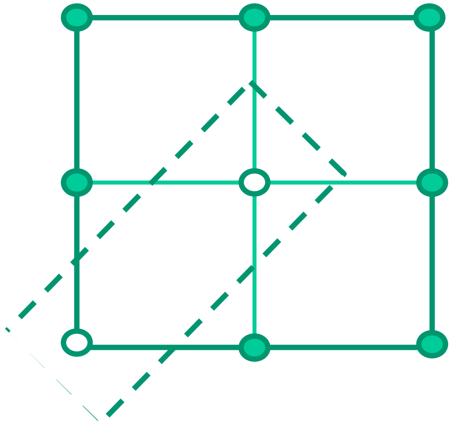
- Diagonally opposite cases have two equally simple possible interpretations



Ambiguity Example

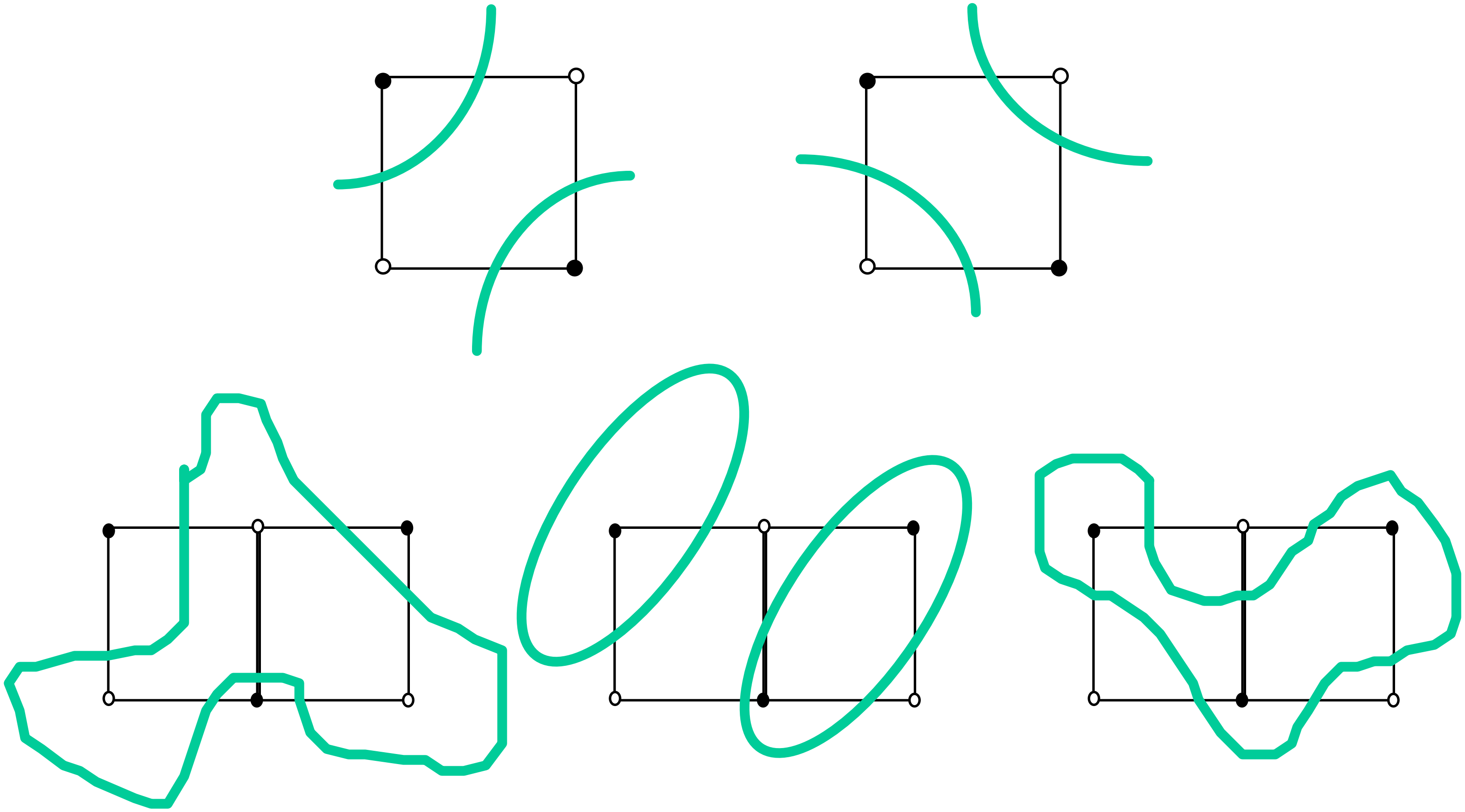
- Two different possibilities below
- More possibilities on next slide





Mechanical Engineering
Carnegie Mellon

Ambiguity Problem

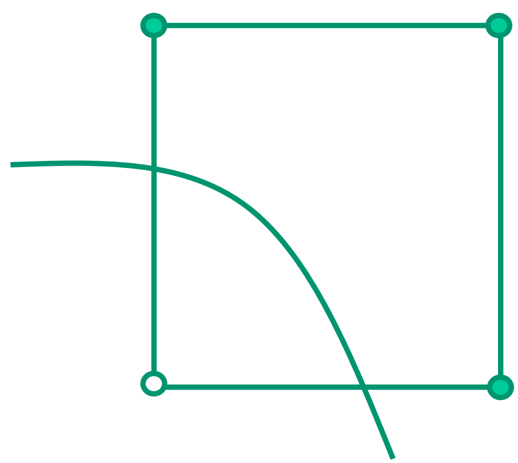


Is Problem Resolvable?

- This is a sampling problem
  - Not enough samples to know the local detail
  - No solution in a mathematical sense without extra information
- More of a problem with volume extension (marching cubes) where selecting "wrong" interpretation can leave a hole in a surface
- Multiple methods in literature to give better appearance
  - Super-sampling
  - Look at larger area before deciding

Interpolating Edges

- We can compute where contour intersects edge in multiple ways
  - Halfway between vertices
  - Interpolated based on difference between contour value and value at vertices



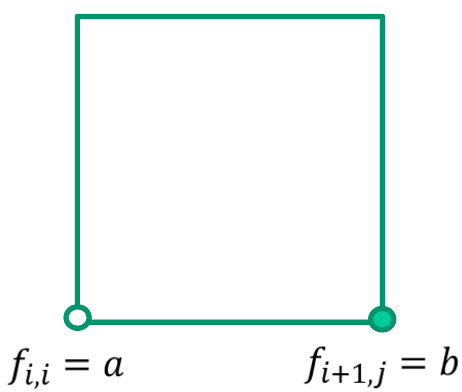
Interpolation of the Intersection of a Cell Edge

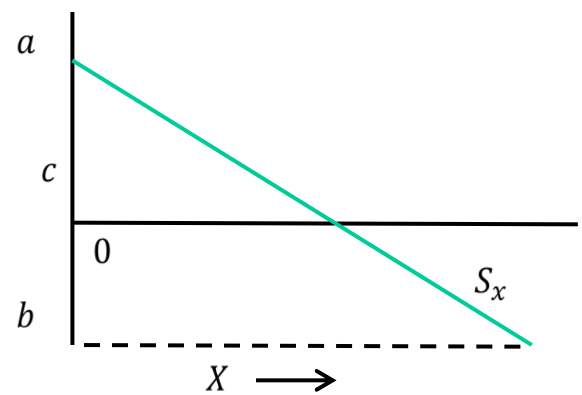
- Consider two vertices that share an edge of a cell and have values on opposite sides of c

$$f(x_i, y_i) = a, \quad a > c,$$
$$f(x_{i+1}, y_j) = b, \quad b < c.$$

- If the two vertices are separated by an x spacing Δx, then we can interpolate the point of intersection using a line segment. This line segment intersects that x-axis at

$$x = x_i + \frac{(a - c)\Delta x}{a - b}.$$





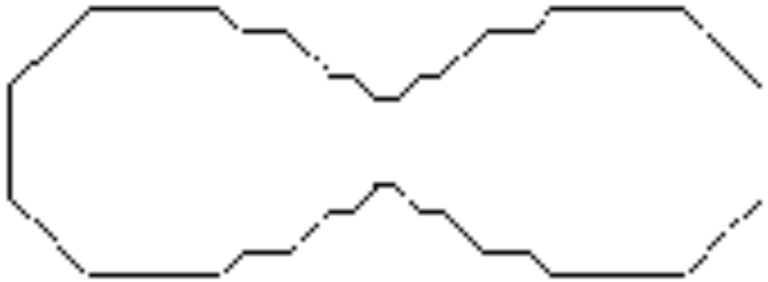
Mechanical Engineering
Carnegie Mellon

Example: Oval of Cassini

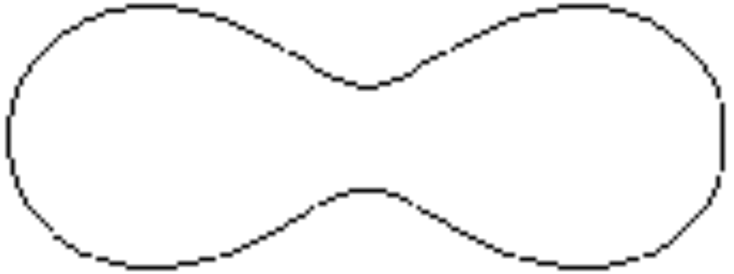
$$f(x, y) = (x^2 + y^2 + a^2)^2 - 4a^2x^2 - b^4$$

Depending on *a* and *b* we can have 0, 1, or 2 curves

midpoint intersections

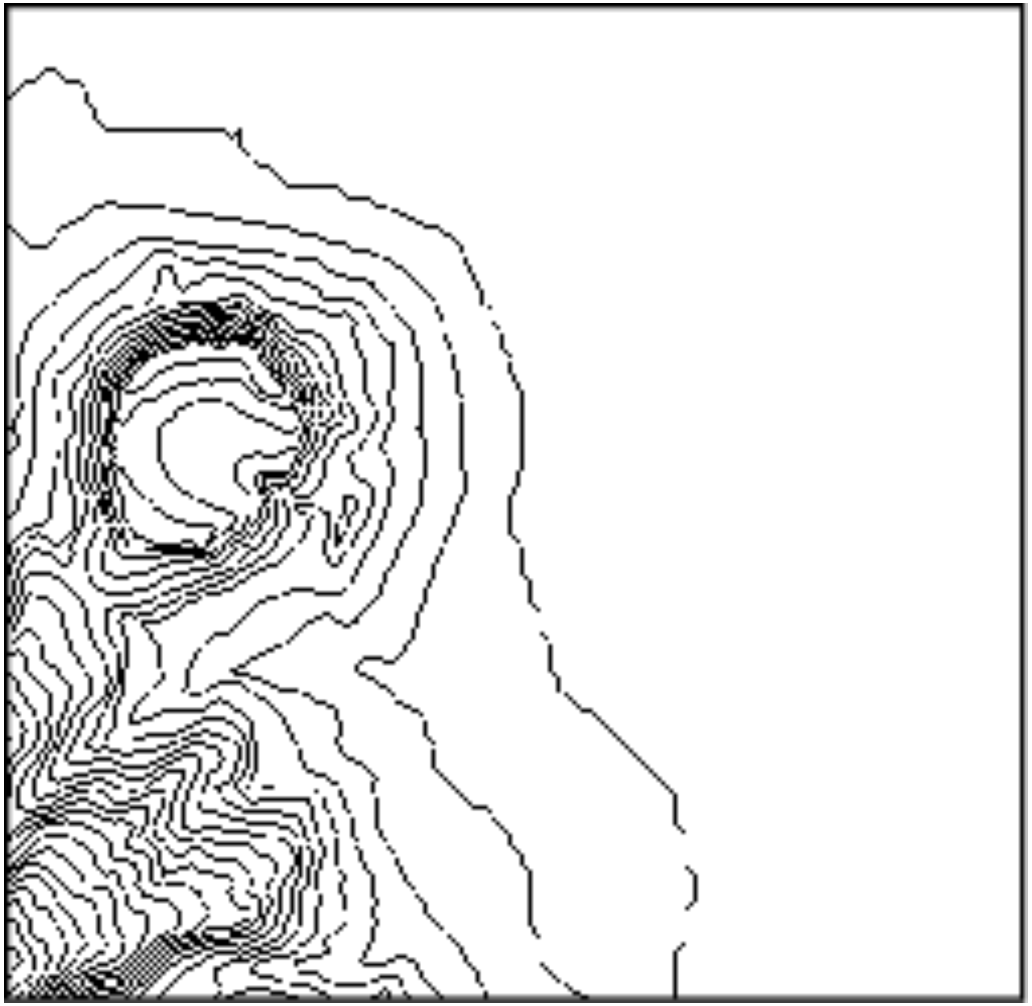


interpolating intersections



Contour Map

- Diamond Head, Oahu Hawaii
- Shows contours for many contour values



Mechanical Engineering
Carnegie Mellon

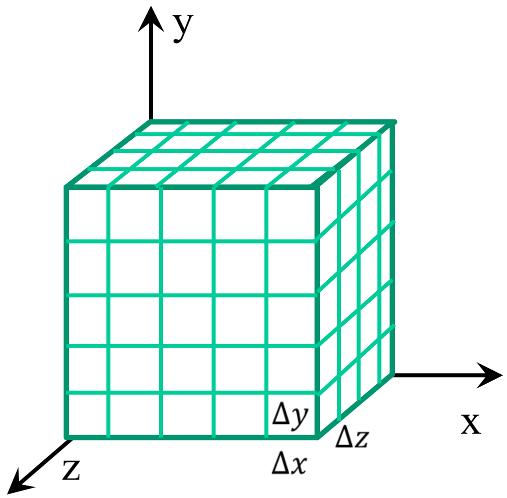
Advantage of Marching Squares

- There are other ways to construct contour curves. One is to start with a cell that is known to have a piece of the contour, and then to follow this contour to adjacent cells as necessary to complete the contour.
- The marching squares method has the advantage that all cells can be dealt with independently, and the extension to 3D for volumetric data is straightforward.

Volumetric Data Sets

- A volumetric data set is discrete data from a set of measurements, or is obtained by evaluating (or sampling) the function for a set of points. Assume that our samples are taken at equally spaced points in *x*, *y*, and *z*:

$$x_i = x_0 + i\,\Delta x$$
$$y_j = y_0 + j\Delta y$$
$$z_k = z_0 + k\,\Delta z$$
$$f_{ijk} = f\,(x_i, y_j, z_k)$$



Each f_{ijk} can be thought of as the average value of the scalar field within a right parallelepiped of sides Δx, Δy, Δz centered at (x_i, y_j, z_k). We call this parallelepiped a volume element or **voxel**.

Volumetric Data Sets

- There are two basic approaches to display these datasets
  - Direct volume rendering: make use of every voxel in producing an image.
  - Isosurfaces: use only a subset of the voxels. c is the isovalue.

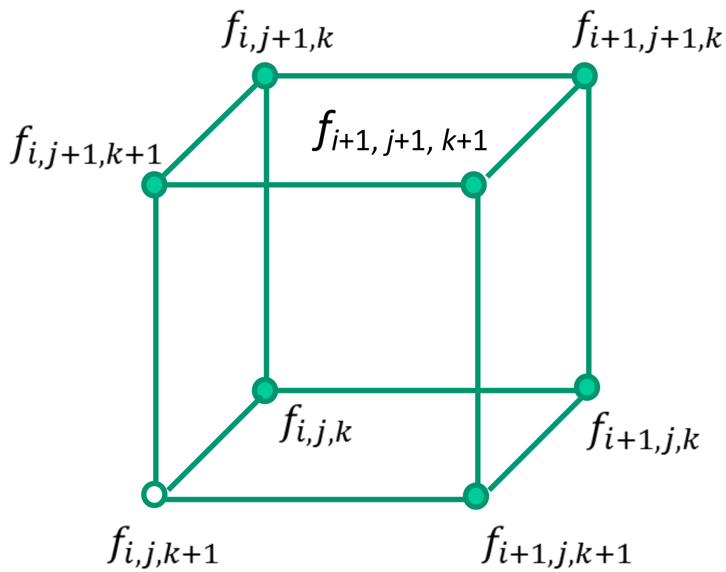
$$f(x, y, z) = c$$

Marching Cubes

- Isosurface: solution of $g(x, y, z) = c$
- Same argument as Marching Squares to derive method but use cubic cell (8 vertices, 256 colorings)
- Standard method of volume visualization
- Suggested by Lorensen and Cline before marching squares

Marching Cubes

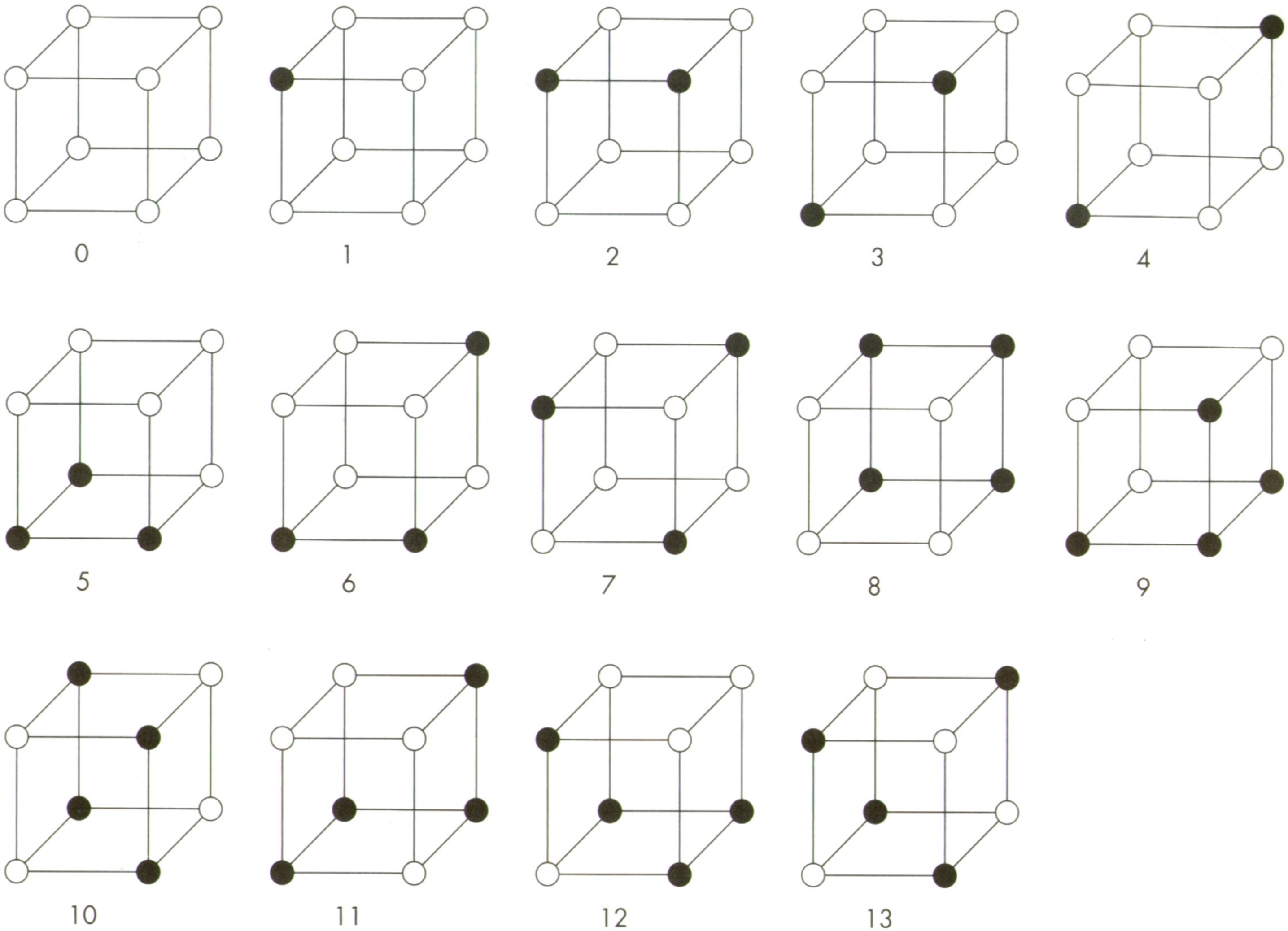
- For a given isosurface value c, we can color the vertices of each cell black or white, depending on whether the value at the vertex is greater than or less than c.



Mechanical Engineering
Carnegie Mellon

Marching Cubes

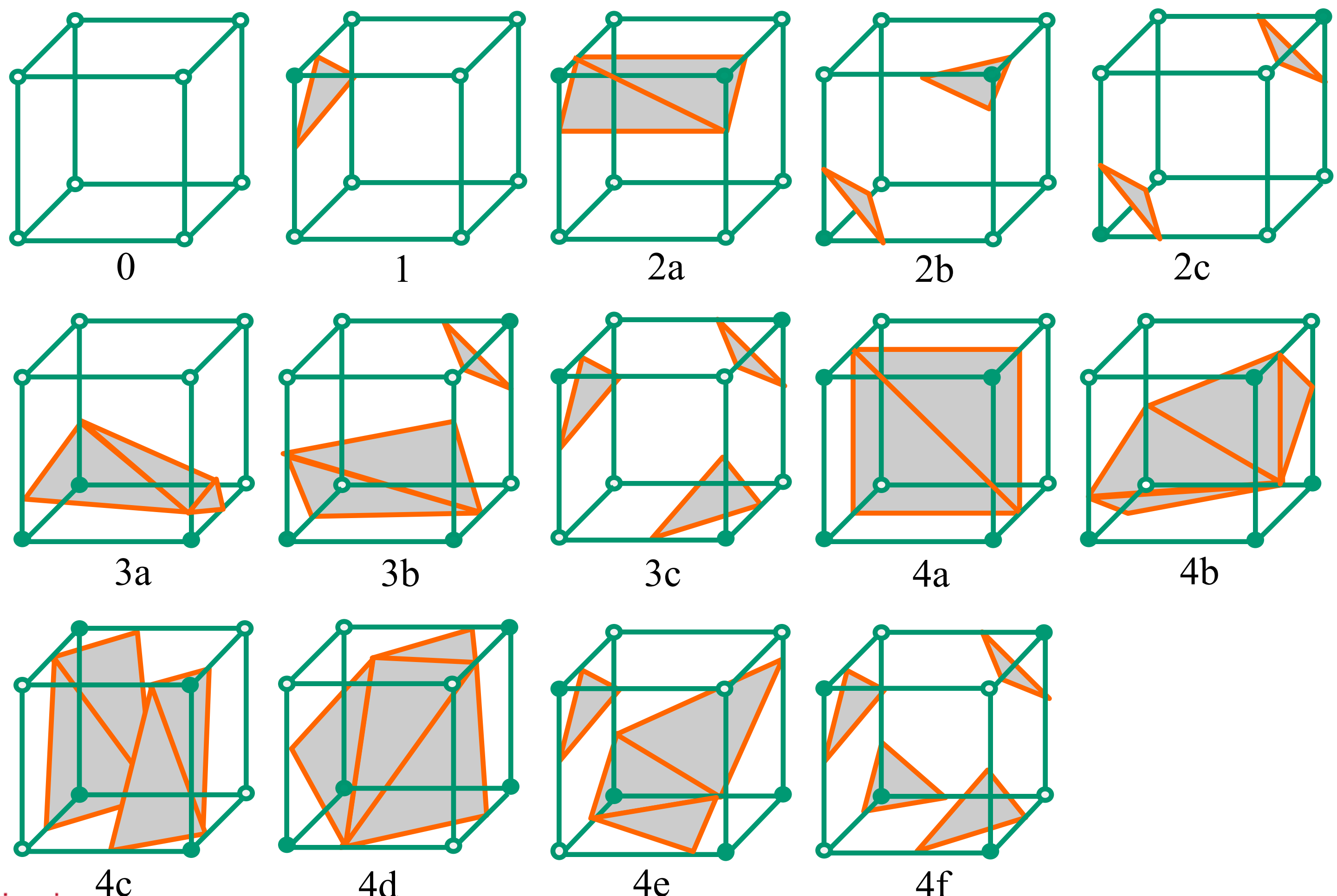
- There are $256=2^8$ possible vertex coloring, but, once we account for symmetries, there are only 14 unique cases.



Mechani

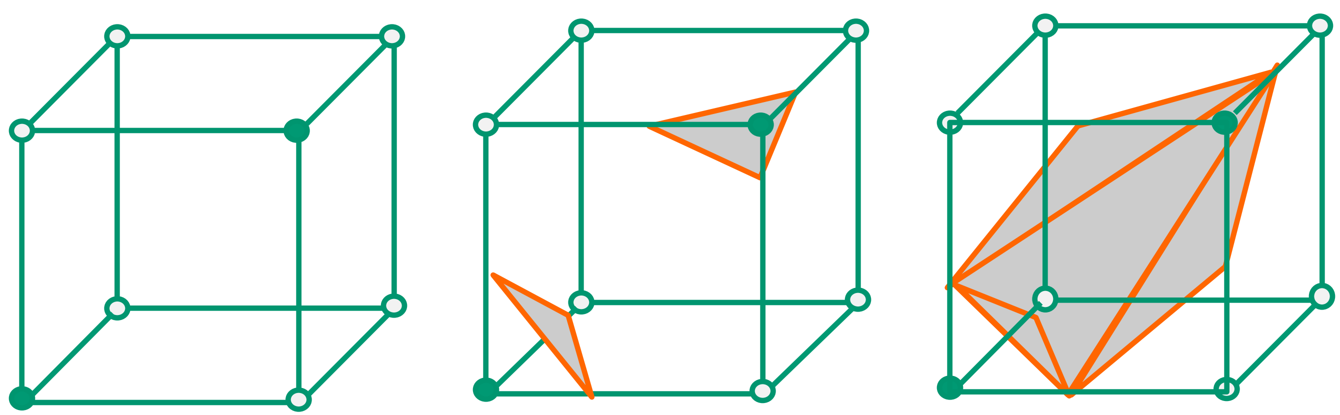
Marching Cubes

- First calculate all intersection points between the surface and the edges using linear interpolation, then use the triangular polygons to tessellate these intersections, forming pieces of a triangular mesh passing through the cell.



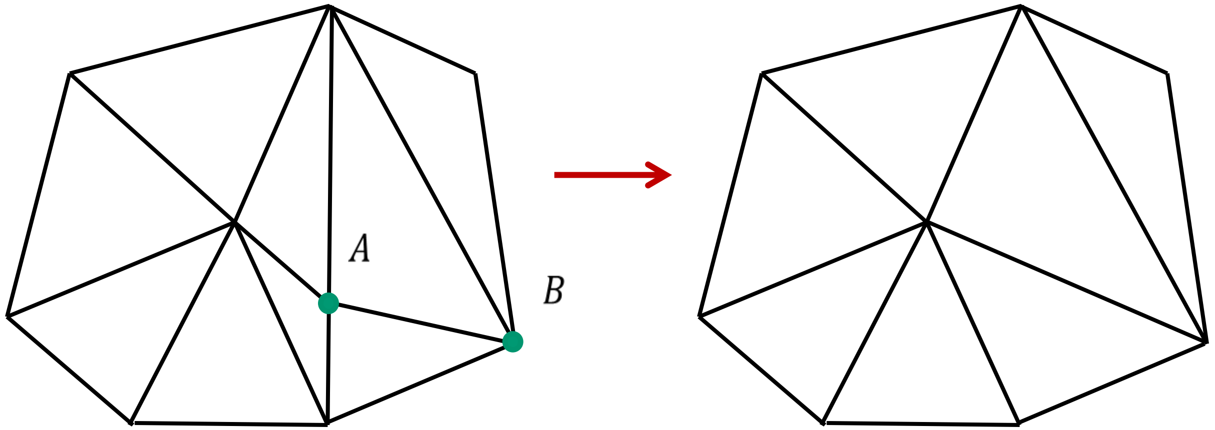
Ambiguity Problem

- If we compare two isosurfaces generated with the two different interpretations, areas where these cases arise will have completely different shapes and topologies.
- The wrong selection of an interpretation for a particular cell can leave a hole in an otherwise smooth surface.
- An always-correct solution requires more information than is present in the data.



Mesh Simplification

- Marching Cubes generates many more triangles than are really needed to display the isosurface due to the resolution of the data set. Therefore mesh simplification is necessary.
- Triangle decimation, removing some edges and vertices. Edge contraction is one method, and local smoothness or the shape of triangles can be used as the criteria.



Mechanical Engineering
Carnegie Mellon

Mesh Simplification

- Other approaches: resample the surface generated by the original mesh to create a new set of points lying on the surface, then use Delaunay triangulation to remesh them.
- Delaunay triangulation generates triangular meshes in which the circle determined by the vertices of each triangle contains no other vertices (empty circle). The complexity is $O(n\log n)$, where n is the number of points.

Direct Volume Rendering

- All voxels contribute to the final image.
- Volume rendering treats each voxel as a small cube, which is rendered by assigning color, opacity.

References

- Interactive Computer Graphics: A top-down approach using OpenGL. Edward Angel, 3rd Edition. Pearson Edition. Chapter 12.