CSC 433/533 Computer Graphics

Alon Efrat
Credit Joshua Levine

Lecture 28 Implicit Modeling

Dec. 9, 2019

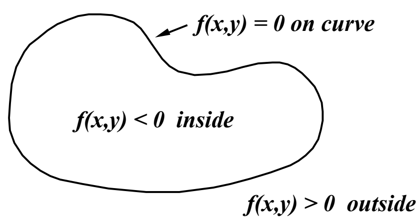
Implicit Modeling of Shapes

Recall: Shape Models That We Have So Far

- Implicit Shapes ($f(\mathbf{p}) = 0$ for all $\mathbf{p}$ on shape):
  - Sphere: $f(\mathbf{p}) = (\mathbf{p} - \mathbf{c}) \cdot (\mathbf{p} - \mathbf{c}) - R^2 = 0$
  - Plane: $f(\mathbf{p}) = (\mathbf{p} - \mathbf{a}) \cdot \mathbf{n} = 0$
- Parametric Shapes ($\mathbf{p}(t)$ is a point on shape for all t):
  - Rays: $\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$
  - Triangles: $\mathbf{p}(\alpha,\beta,\gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$, and triangle meshes
  - Splines: Hermite, Bézier, B-Splines, Catmull-Rom
  - Subdivision Surfaces: Loop, Catmull-Clark

Implicit Surfaces

- Surface defined implicitly by function:
  - f (x, y, z) = 0 (on surface)
  - f (x, y, z) < 0 (inside)
  - f (x, y, z) > 0 (outside)



Turk

Benefits of Implicit Shapes

- Many operations that can be simplified:
  - Rendering: Can check intersections efficiently
  - Collisions: Can check whether a point is inside/outside
  - Design: Blending/Solid modeling
  - Simulation: Can naturally represent certain simulation techniques used in animation

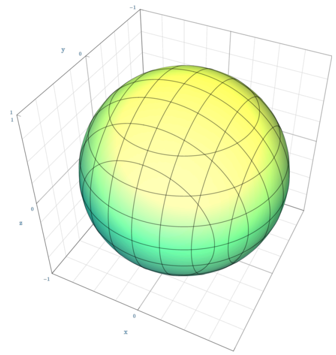
Implicit Functions

Defining $f(x, y, z) = 0$

- Methods to define implicit functions:
  - Algebraic Equations
  - Distance Functions
  - Voxels

Algebraic Functions

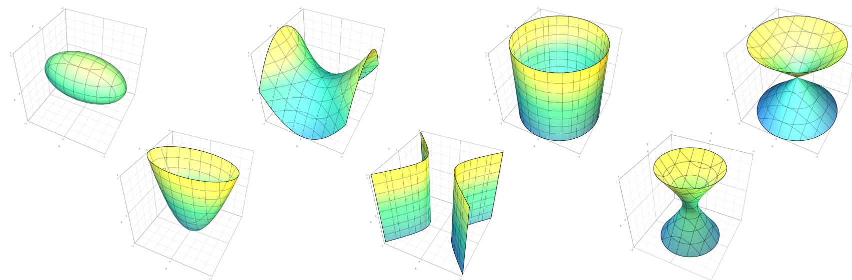
- Can use any polynomial form
    - e.g. sphere,
      $f(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{x}^2 + \mathbf{y}^2 + \mathbf{z}^2 - r^2$



https://en.wikipedia.org/wiki/Quadric

Quadric Surfaces

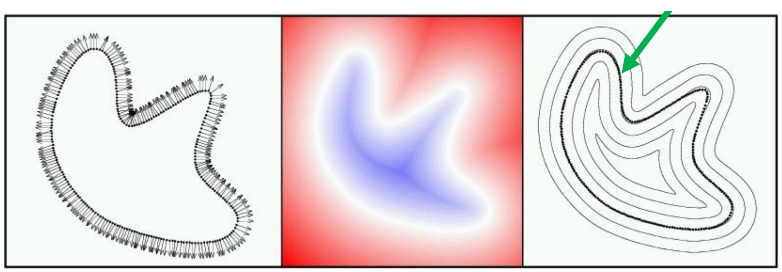
- More generally, can define f as any order two polynomial
  $f(\mathbf{x}, \mathbf{y}, \mathbf{z}) = A\mathbf{x}^2+B\mathbf{y}^2+C\mathbf{z}^2 + D\mathbf{xy}+E\mathbf{xz}+F\mathbf{yz} + G\mathbf{x}+H\mathbf{y}+I\mathbf{z} + J$
- Provides a family of shapes: ellipses, paraboloids, hyperboloids, cylinders, cones



https://en.wikipedia.org/wiki/Quadric

Signed Distance Fields

- Idea: design a function that computes distance from some shape.
- Distance is signed: positive is outside, negative inside



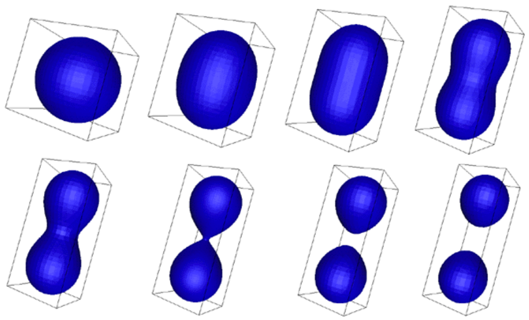
Building Implicit Functions w/ Signed Distance Fields

- Simple example: working with point primitive
- Distance function:
    - Let $\mathbf{x} = (x,y,z)$ and define $d(x,y,z)$ be the distance from $\mathbf{x}$ to a point $\mathbf{p}$, e.g. $d(\mathbf{x}) = ||\mathbf{x}\text{-}\mathbf{p}||$
- Consider the function $f(\mathbf{x}) = d(\mathbf{x}) - r$.
    - This forms a collection of spherical shells of radius r

Composition

- Can combine multiple constraints additively.

$$f(x,y,z) = \sum_{i=1}^{i=n} f_i(x,y,z)$$



http://paulbourke.net/geometry/implicitsurf/index.html

Fall-off Filters

- We compose the distance function with a fall-off filter, or basis function to help limit its local effect

$$f_i(x,y,z) = g_i \circ d_i(x,y,z)$$

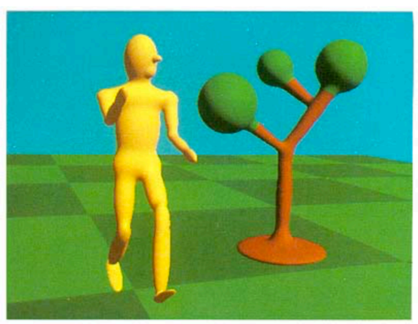
- Many potential choices for g_i

Blinn's Blobs

- Modeled after electron density field using Gaussians:

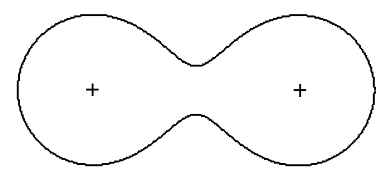
$$g(d) = e^{-rd^2}$$

- r based on target radius
- Can also multiply by a constant to affect "blobbiness"



Blobby Models

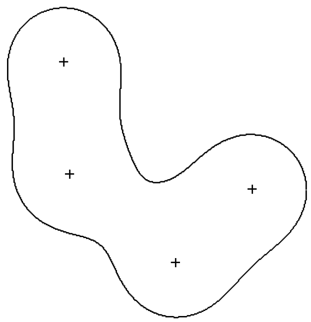
- Sum of two blobs



Turk

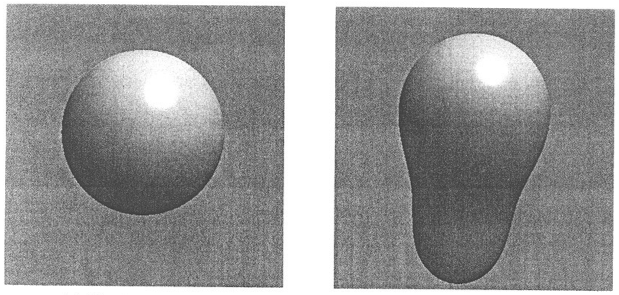
Blobby Models

- Sum of four blobs



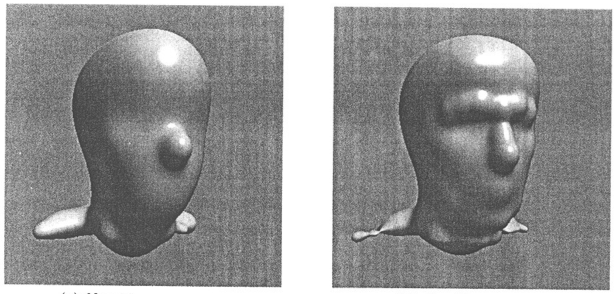
Turk

Blobby Model of Head



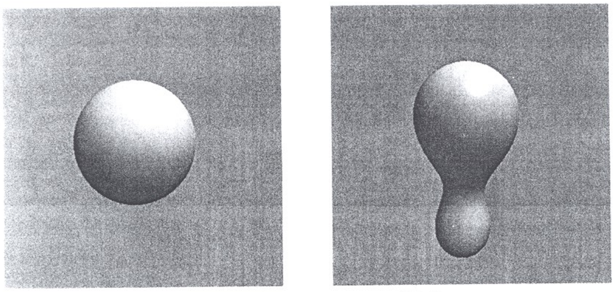
(a) $N = 1$ (b) $N = 2$

Blobby Model of Head



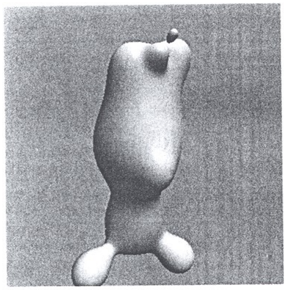
(c) $N = 20$ (d) $N = 60$

Blobby Model of Face

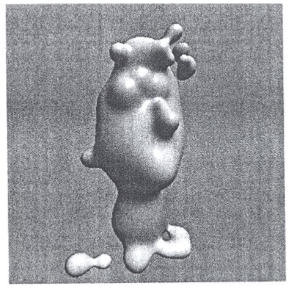


(a) $N = 1$ (b) $N = 2$

Blobby Model of Face

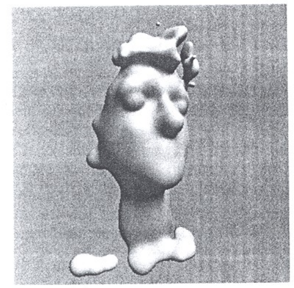


(c) $N = 10$

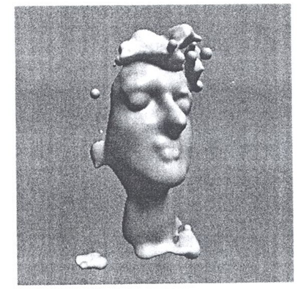


(d) $N = 35$

Blobby Model of Face



(e) $N = 70$

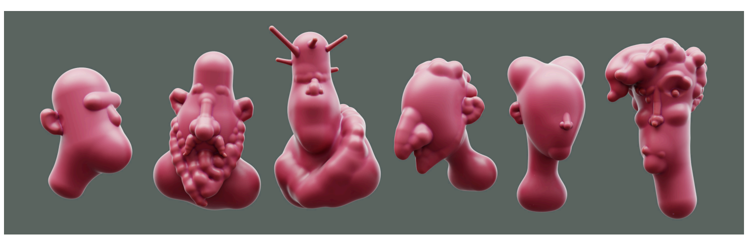


(f) $N = 243$

Metaballs: Nishimura et al.

- Offers finite support
- Commonly used in tools like Blender

$$g(d) = \begin{cases} 1 - 3(\frac{d}{r})^2 & 0 \le d \le \frac{r}{3} \\ \frac{3}{2}(1 - \frac{d}{r})^2 & \frac{r}{3} \le d \le r \\ 0 & d > r \end{cases}$$



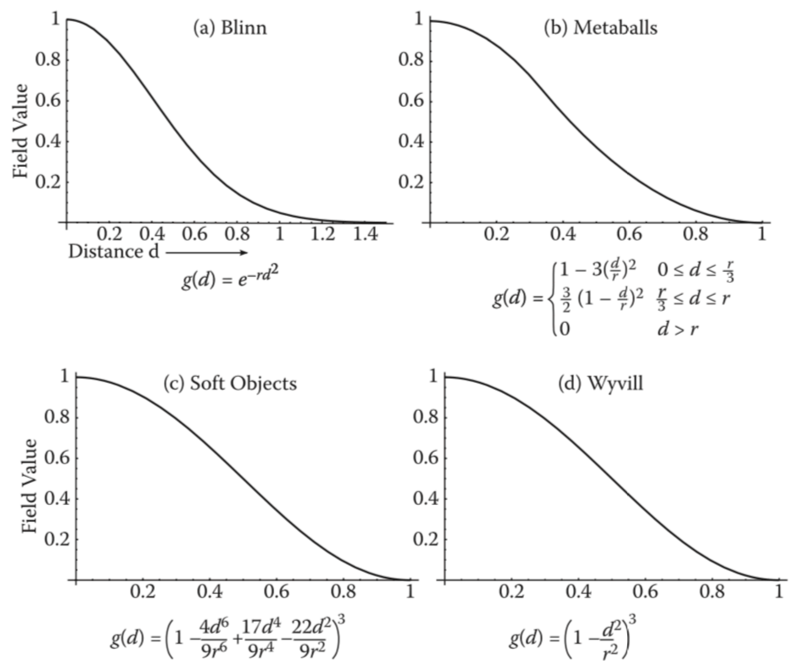
https://www.blender.org/conference/2017/presentations/359

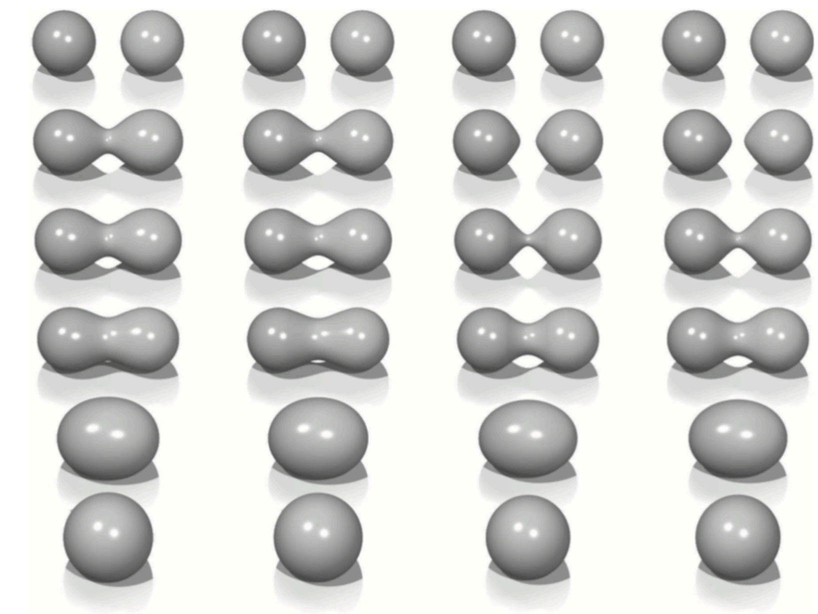
Soft Objects: Wyvill and Wyvill

- Truncated expansion of the exponential, defined for $d < r$.

$$g(d) = \left(1 - \frac{4d^6}{9r^6} + \frac{17d^4}{9r^4} - \frac{22d^2}{9r^2}\right)^3$$

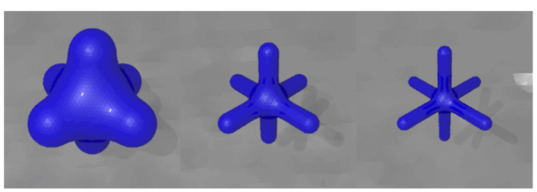
$$g(d) = \left(1 - \frac{d^2}{r^2}\right)^3$$

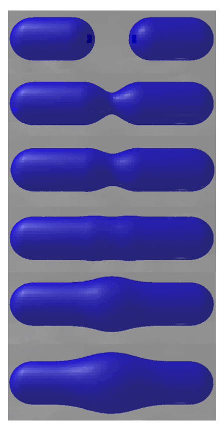




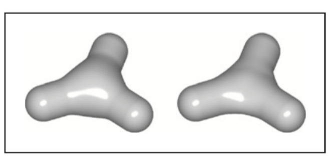
Other Primitives

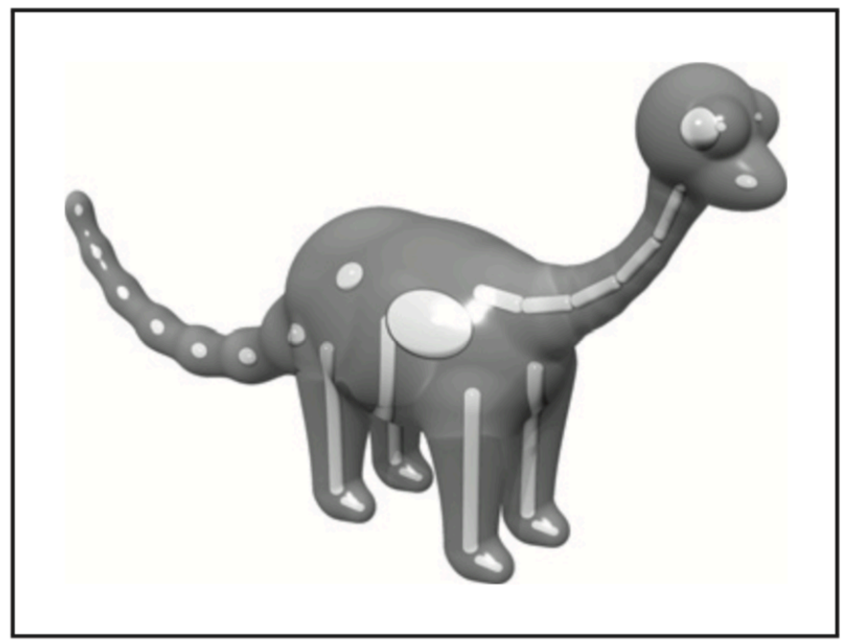
- Commonly used: distances to line segments





- Can be tricky to handle bulging

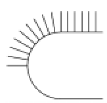




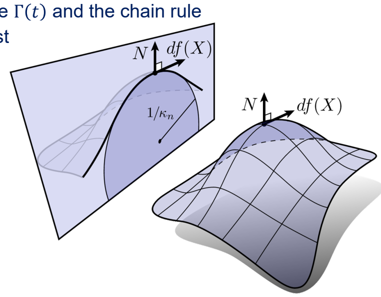
Defining Surface Normals

- Convention: use the gradient of the implicit function:

- Normals defined by partial derivatives
  - Normal - $N(x,y,z) = normalize\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right) = normalize(\vec{\nabla} f)$
    - Example: circle $x^2 + y^2 - 3^2 = 0$
    - Proof: straight forward with an arbitrary curve $\Gamma(t)$ and the chain rule
    - Intuition: Max change rate direction of f must perpendicular to isosurface direction



Normals



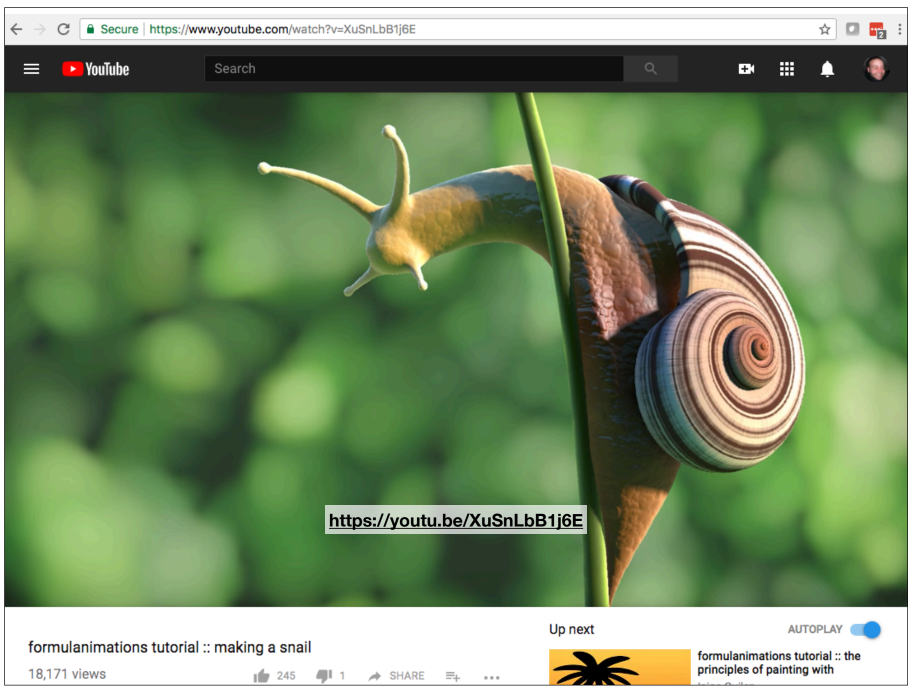
Bloomenthal

Approximating the Gradient

- Might be able to get this from a closed form.

- Alternatively, use numerical differentiation to compute:

$$
\begin{aligned}
\nabla h(X) &= \left(\frac{\partial h(X)}{\partial x}, \frac{\partial h(X)}{\partial y}, \frac{\partial h(X)}{\partial z}\right) \\
&\approx \frac{1}{2\epsilon}\left(h(X+\hat{x}\epsilon) - h(X-\hat{x}\epsilon), h(X+\hat{y}\epsilon) - h(X-\hat{y}\epsilon), h(X+\hat{z}\epsilon) - h(X-\hat{z}\epsilon)\right) \\
&\approx \frac{1}{\epsilon}\left[h(X) - (h(X+\hat{x}\epsilon), h(X+\hat{y}\epsilon), h(X+\hat{z}\epsilon))\right] \qquad (12)
\end{aligned}
$$



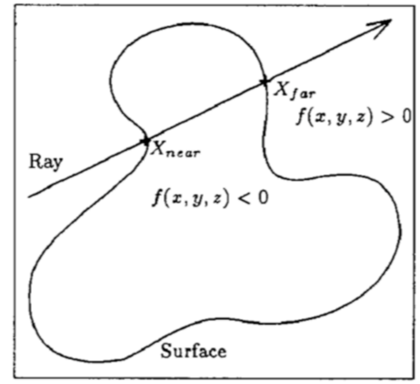
Rendering Implicit Functions

Rendering Methods

- Raytracing:
  - Given a ray, we can test whether or not the ray intersects the surface by plugging into the implicit equation and finding roots of $f(\mathbf{p}(t)) = 0$
- Rasterization:
  - Implicit functions do not come with geometry (i.e. triangles) to draw. To render we first construct geometry and then apply standard rasterization primitives.

Finding Intersections Between Rays and Implicit Functions

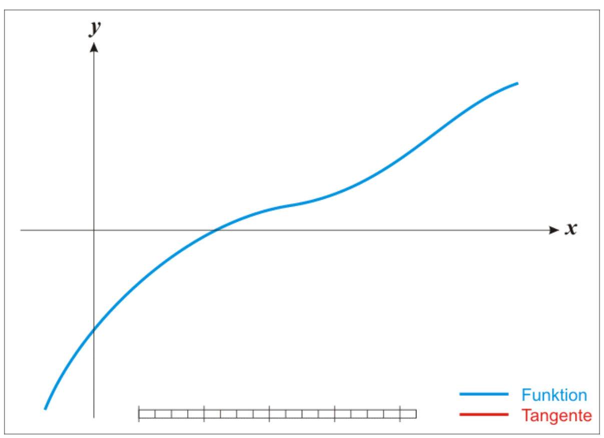
- Intermediate Value Theorem:
  - If a continuous function f defined on an interval [a,b] takes on values f(a) and f(b), it also takes on any value between f(a) and f(b)
- Can use root-find techniques like Newton's method



Kalra and Barr, Guaranteed Ray Intersections with Implicit Surfaces

Newton's Method

- Use f' to take a step towards the root.
- $t_1 = t_0 - f(t_0)/f'(t_0)$
- Numerous variations, regula falsi, secant method, etc.



https://en.wikipedia.org/wiki/Newton%27s_method

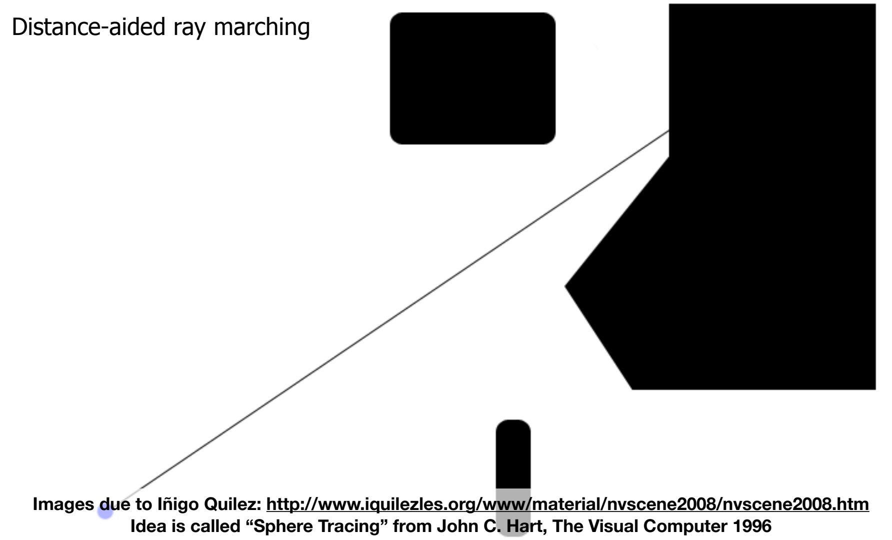
Marching along Rays

- Naive approach:

```
findRoot(t_min, t_max) {
  dt = 0.01
  t = t_min
  while(t < t_max and f(t) > 0) {
    t += dt
  }
  return t
}
```

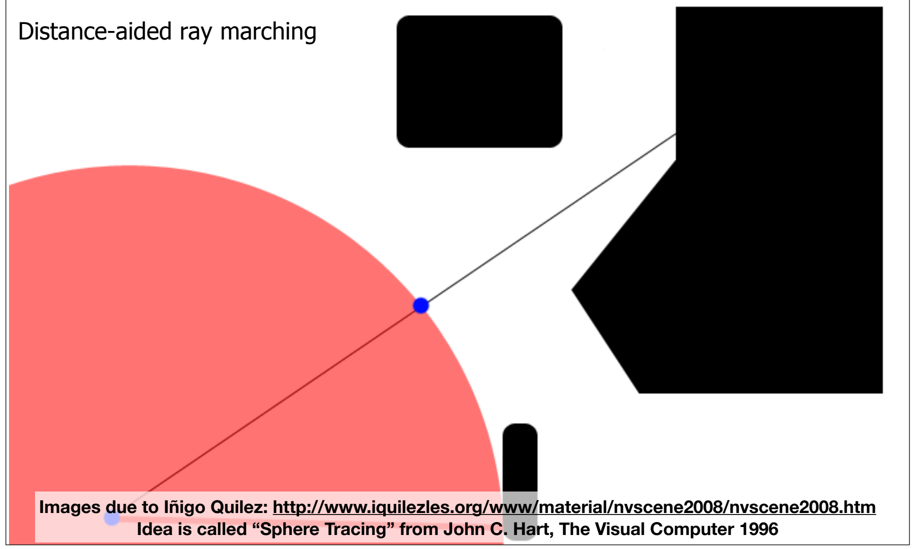
- Many alternatives:
  - Binary search for where the sign changes, varying the dt
  - If f is a distance function, can take a step equal to the value of $f(t)$

If f is a distance function, can take a step equal to the value of $f(t)$



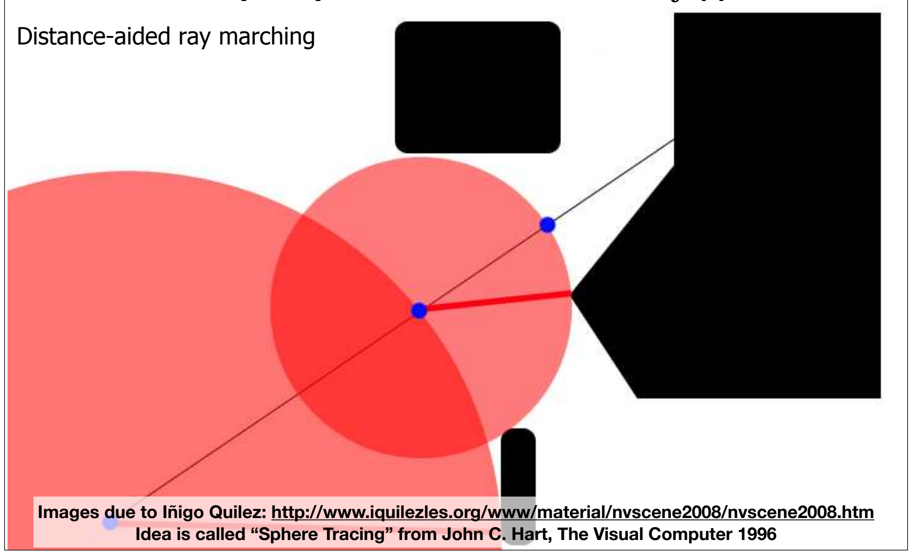
Images due to Iñigo Quilez: http://www.iquilezles.org/www/material/nvscene2008/nvscene2008.htm
Idea is called "Sphere Tracing" from John C. Hart, The Visual Computer 1996

If f is a distance function, can take a step equal to the value of $f(t)$



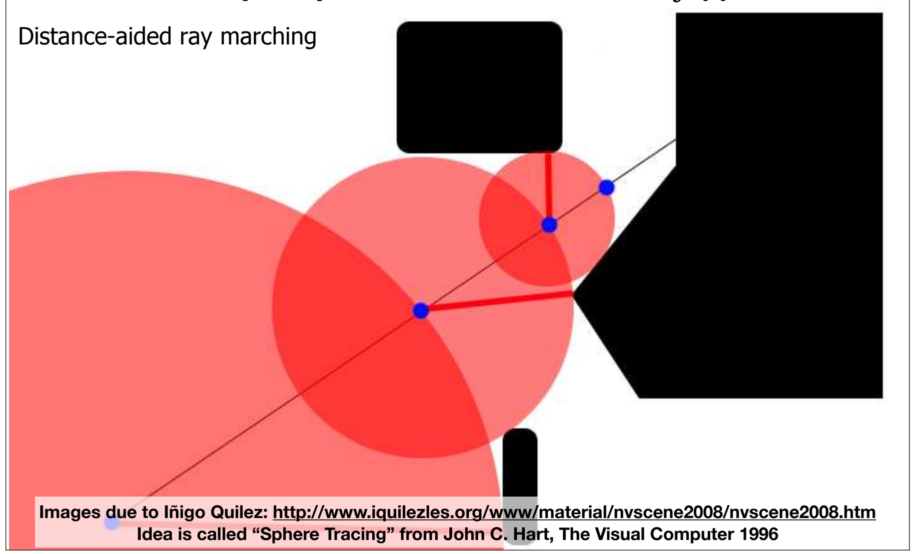
Images due to Iñigo Quilez: http://www.iquilezles.org/www/material/nvscene2008/nvscene2008.htm
Idea is called "Sphere Tracing" from John C. Hart, The Visual Computer 1996

If f is a distance function, can take a step equal to the value of $f(t)$



Images due to Iñigo Quilez: http://www.iquilezles.org/www/material/nvscene2008/nvscene2008.htm
Idea is called "Sphere Tracing" from John C. Hart, The Visual Computer 1996

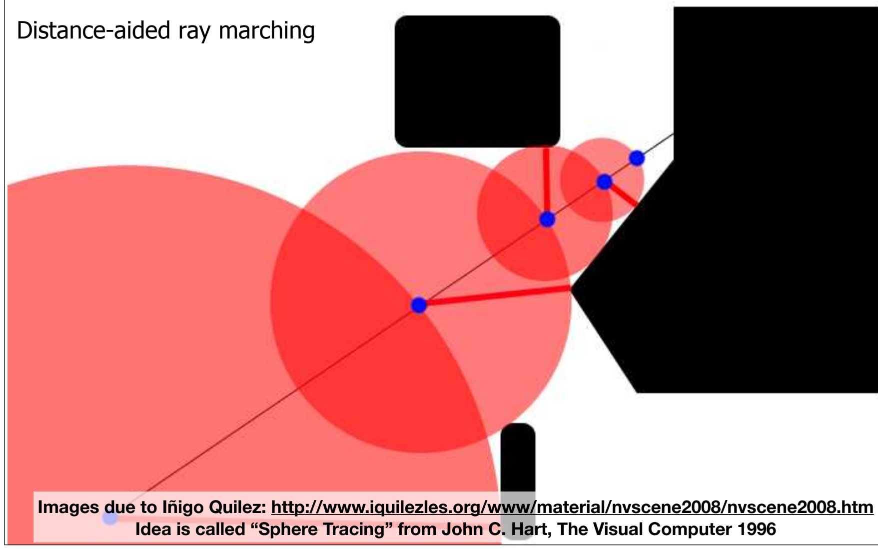
If f is a distance function, can take a step equal to the value of $f(t)$



Images due to Iñigo Quilez: http://www.iquilezles.org/www/material/nvscene2008/nvscene2008.htm
Idea is called "Sphere Tracing" from John C. Hart, The Visual Computer 1996

If f is a distance function, can take a step equal to the value of $f(t)$

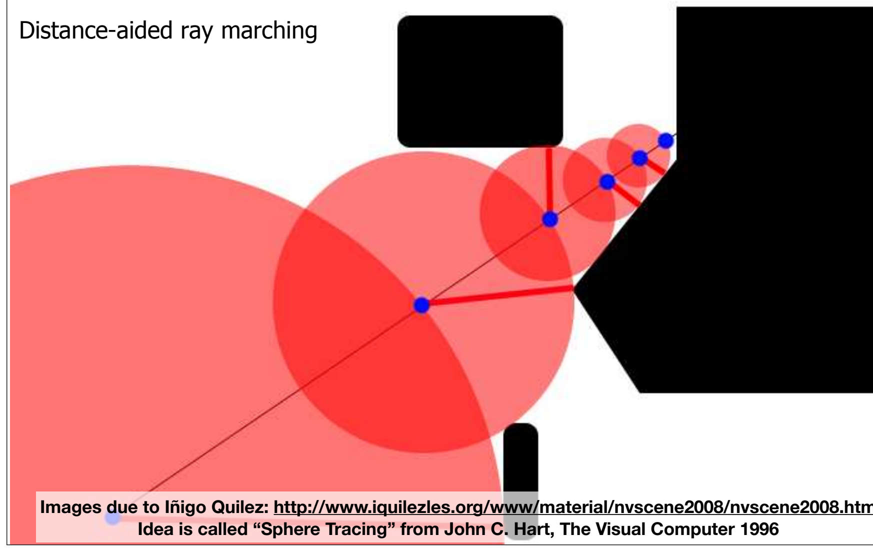
Distance-aided ray marching



Images due to Iñigo Quilez: http://www.iquilezles.org/www/material/nvscene2008/nvscene2008.htm
Idea is called "Sphere Tracing" from John C. Hart, The Visual Computer 1996

If f is a distance function, can take a step equal to the value of $f(t)$

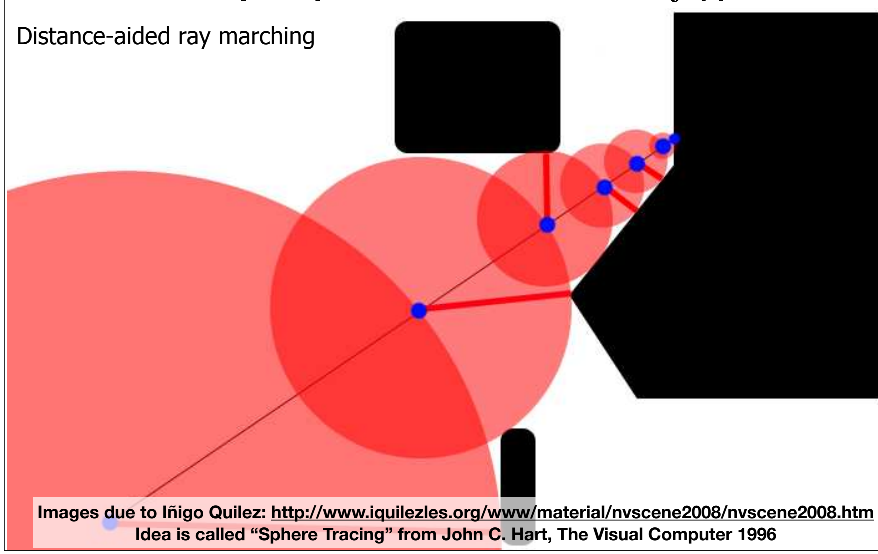
Distance-aided ray marching



Images due to Iñigo Quilez: http://www.iquilezles.org/www/material/nvscene2008/nvscene2008.htm
Idea is called "Sphere Tracing" from John C. Hart, The Visual Computer 1996

If f is a distance function, can take a step equal to the value of $f(t)$

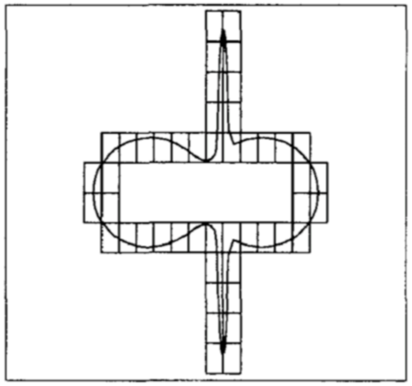
Distance-aided ray marching



Images due to Iñigo Quilez: http://www.iquilezles.org/www/material/nvscene2008/nvscene2008.htm
Idea is called "Sphere Tracing" from John C. Hart, The Visual Computer 1996

Skipping Space

- Precompute a set of bounding boxes that pinpoint areas where the sign of f changes
  - First intersect ray with box, and then search for the root within the box
- Can also build a bounding hierarchy for this, e.g. an octree



Kalra and Barr, Guaranteed Ray Intersections with Implicit Surfaces

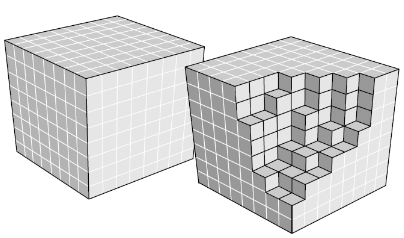
Using Voxels for Representing Implicit Functions

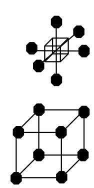
3 Methods for Defining and Implicit Function $f(x, y, z) = 0$

- Methods to define implicit functions:
  - Algebraic Equations
  - Distance Functions
  - **Voxels**

Voxels

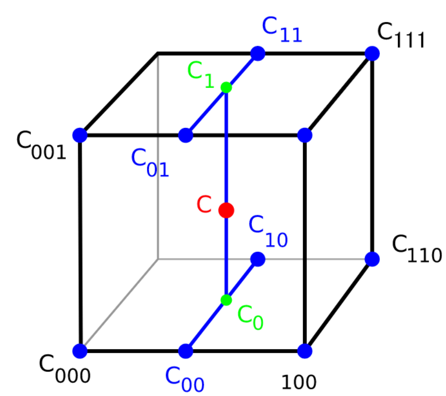
- Can represent any volumetric function by sampling f at a finite set of positions
  - Each sample is called a **voxel** (volume element)
- Similar to images: analogy is pixels are picture elements
- Replace the functional representation with grid-based sample





Interpolating Voxel Data

- Linear interpolation (lerp) extended into three dimensions:
- Find the nearest 8 values to C:
  - First do four linear interpolations in one direction, e.g. C_{00} = lerp(C_{000},C_{100})
  - Next do two linear interpolations in the next direction, e.g. C_0 = lerp(C_{00},C_{10})
  - Finally, do one more linear interpolation in third direction, e.g. C = lerp(C_0,C_1)



https://en.wikipedia.org/wiki/Trilinear_interpolation

Raytracing Voxel Grids

- Similar to our method for raytracing implicit functions, but why not just replace the root finding for hit points with trilinear interpolation to find the root?
  - See more on this in CSC 444/544, e.g. volume rendering
- Potential problems? Trilinear function only approximates the original implicit function, and they might disagree

Constructing Geometry for Rasterization

Using Voxels to Construct Geometry

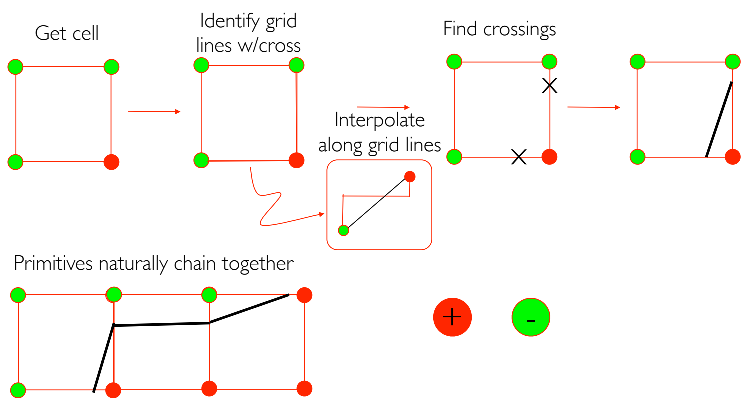
- Instead of raytracing voxel-based data, can we using the voxels to build a geometric approximation of the implicit surface?
- Idea: for each cell in the voxel grid, define a piece of the surface.
- Advantages: potentially much faster to render

Contouring in 2D

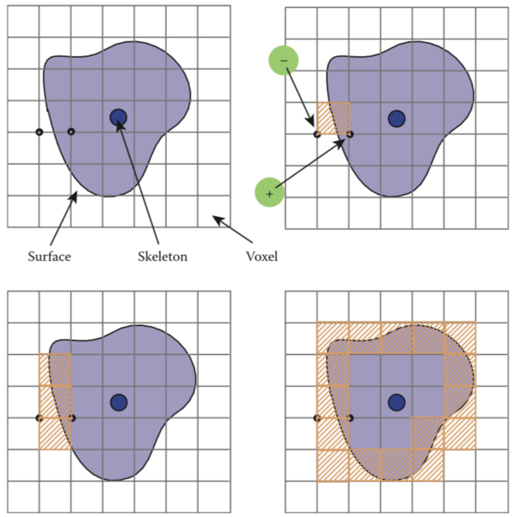
- Idea: Assign geometric primitives to individual cells
  - In 2D, we will use line segments
- Method: Consider the sign of the values at vertices relative to if they are above or below 0
  - Intersections MUST occur on edges with sign change by Intermediate Value Theorem
- Determine exact position of intersection by interpolation along grid edges

Approach to Contouring in 2D

- Contour must cross every grid line connecting two grid points of opposite sign



Only Need to Track Cells that Have a Sign Change



Ambiguities Occur when the Bilinear Interpolant Has Both Hyperbolic Arcs in the Grid Cell

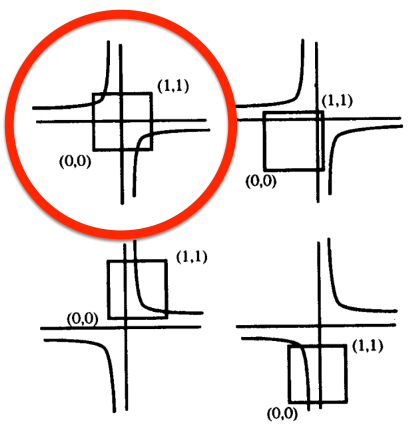
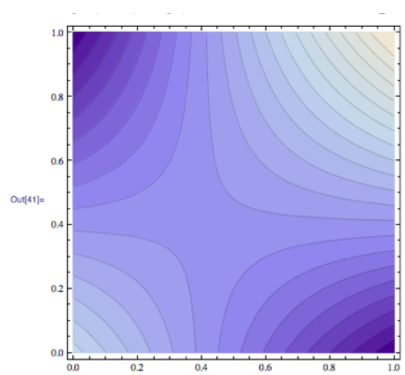


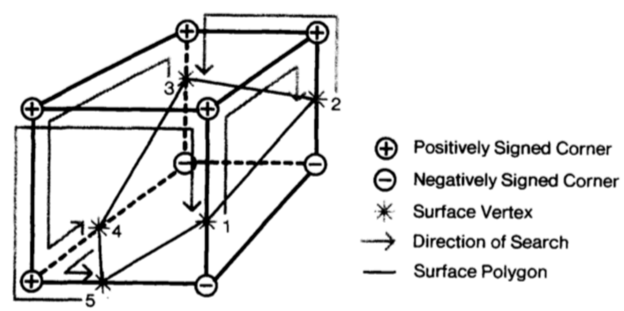
Figure 6. Contours of bilinear interpolant



Nielson, Hamann, The Asymptotic Decider, 1991

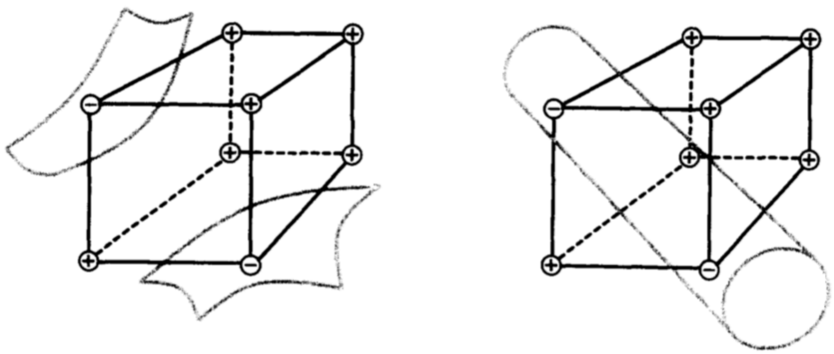
Contouring in 3D

- Uses a similar approach to 2D, but considers cells of 8 nearby voxels



Bloomenthal, Polygonization of implicit surfaces, 1988

Ambiguities in 3D can be more complex



Bloomenthal, Polygonization of implicit surfaces, 1988

MARCHING CUBES: A HIGH RESOLUTION
3D SURFACE CONSTRUCTION ALGORITHM

William E. Lorensen
Harvey E. Cline

General Electric Company
Corporate Research and Development
Schenectady, New York 12301

15,576 Citations on Google Scholar

Related (earlier) paper: Wyvill et al. "Soft objects." Advanced Computer Graphics. Springer Japan, 1986. 113-128.

using linear interpolation. We find the gradient of the original data, normalize it, and use it as a basis for shading the models. The detail in images produced from the generated surface models is the result of maintaining the inter-slice connectivity, surface data, and gradient information present in the original 3D data. Results from computed tomography (CT), magnetic resonance (MR), and single-photon emission computed tomography (SPECT) illustrate the quality and functionality of *marching cubes*. We also discuss improvements that decrease processing time and add solid modeling capabilities.

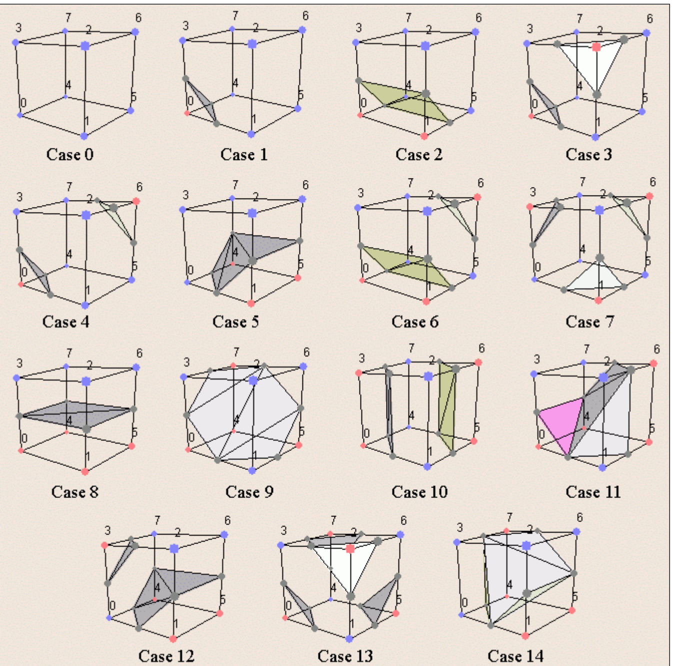
CR Categories: 3.3, 3.5

Additional Keywords: computer graphics, medical imaging, surface reconstruction

Existing 3D algorithms lack detail and sometimes introduce artifacts. We present a new, high-resolution 3D surface construction algorithm that produces models with unprecedented detail. This new algorithm, called *marching cubes*, creates a polygonal representation of constant density surfaces from a 3D array of data. The resulting model can be displayed with conventional graphics-rendering algorithms implemented in software or hardware.

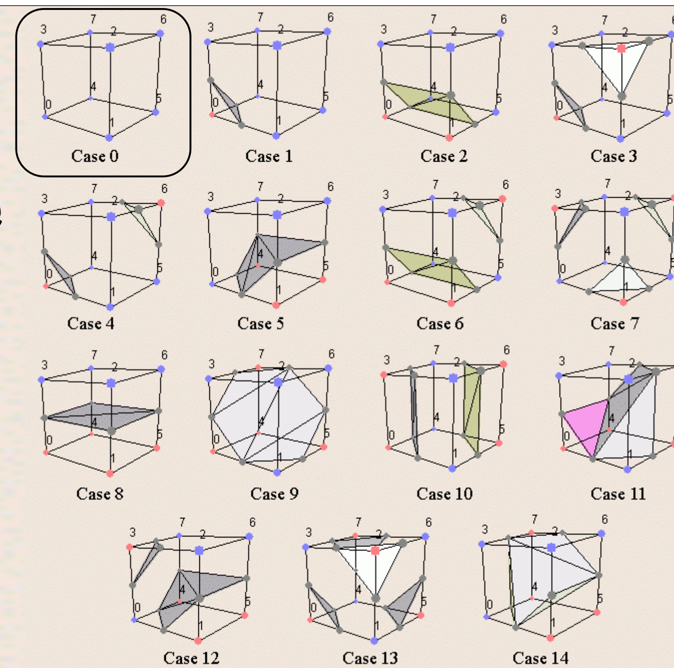
After describing the information flow for 3D medical applications, we describe related work and discuss the drawbacks of that work. Then we describe the algorithm as well as efficiency and functional enhancements, followed by case studies using three different medical imaging techniques to illustrate the new algorithm's capabilities.

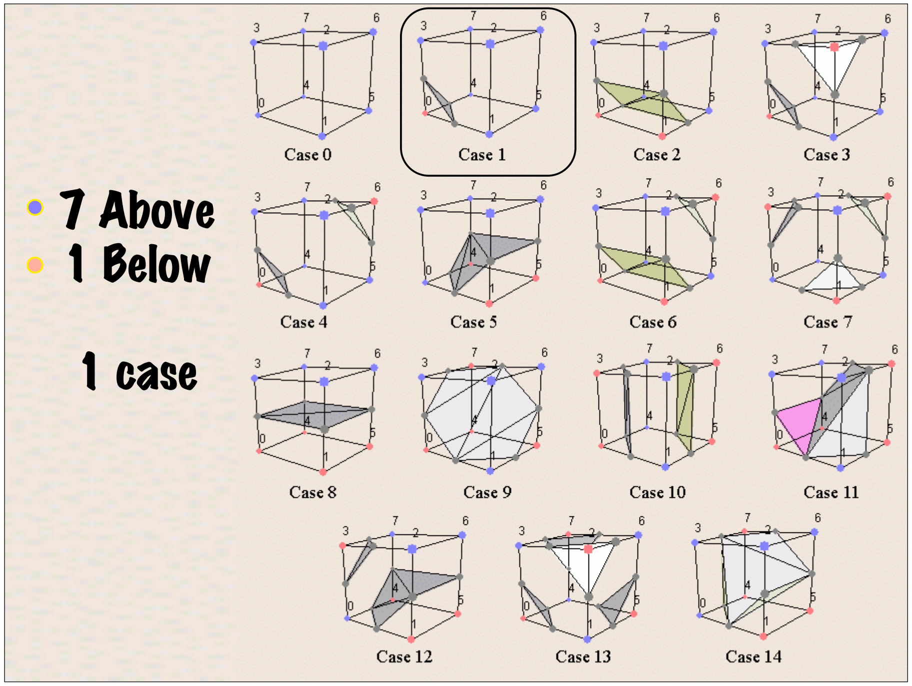
Case Table

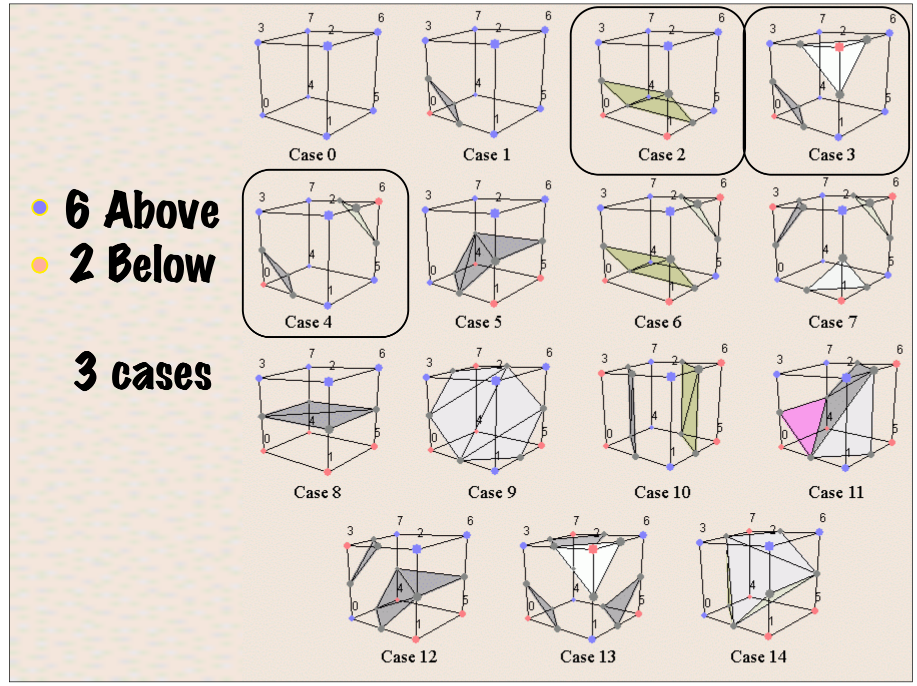


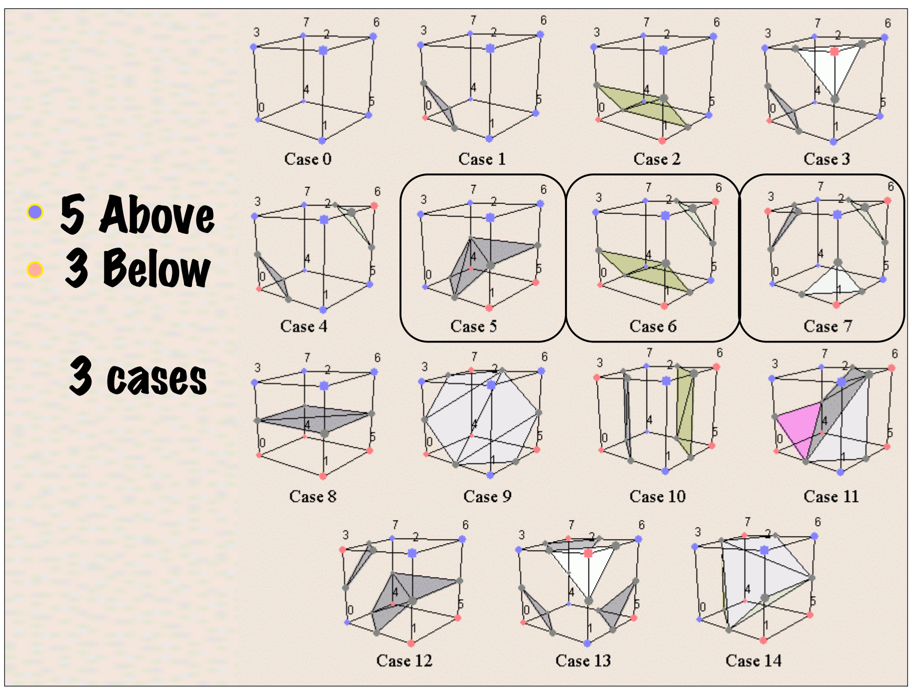
- 8 Above
- 0 Below

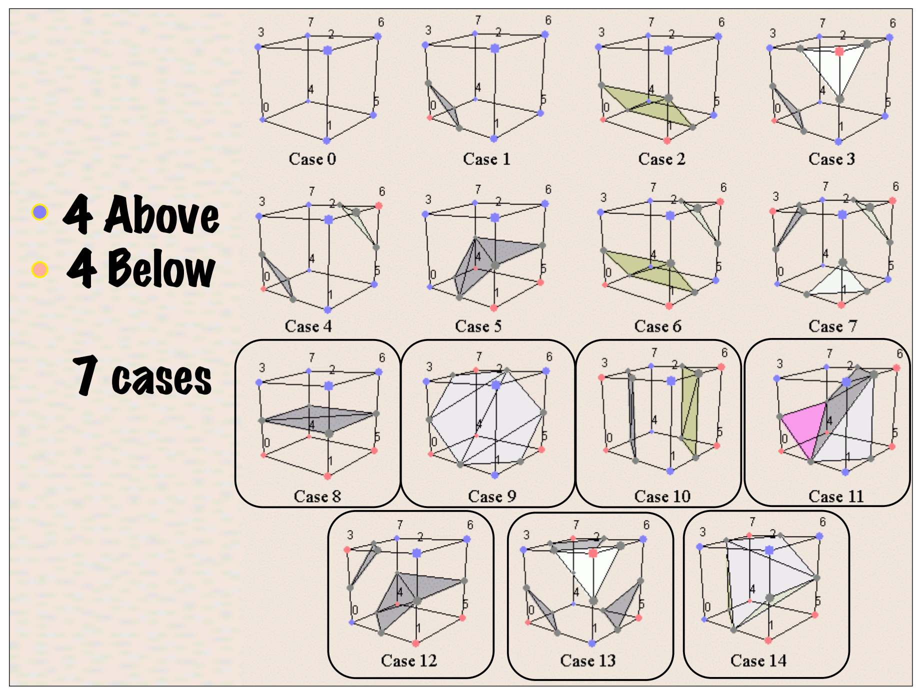
1 case




7 Above
1 Below
1 case
Case 0
Case 1
Case 2
Case 3
Case 4
Case 5
Case 6
Case 7
Case 8
Case 9
Case 10
Case 11
Case 12
Case 13
Case 14

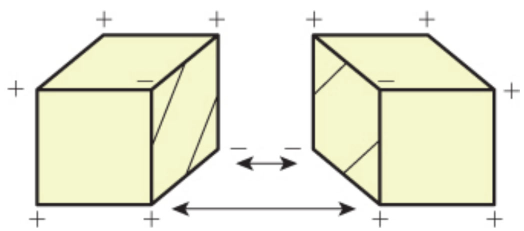

6 Above
2 Below
3 cases
Case 0
Case 1
Case 2
Case 3
Case 4
Case 5
Case 6
Case 7
Case 8
Case 9
Case 10
Case 11
Case 12
Case 13
Case 14


5 Above
3 Below
3 cases
Case 0
Case 1
Case 2
Case 3
Case 4
Case 5
Case 6
Case 7
Case 8
Case 9
Case 10
Case 11
Case 12
Case 13
Case 14

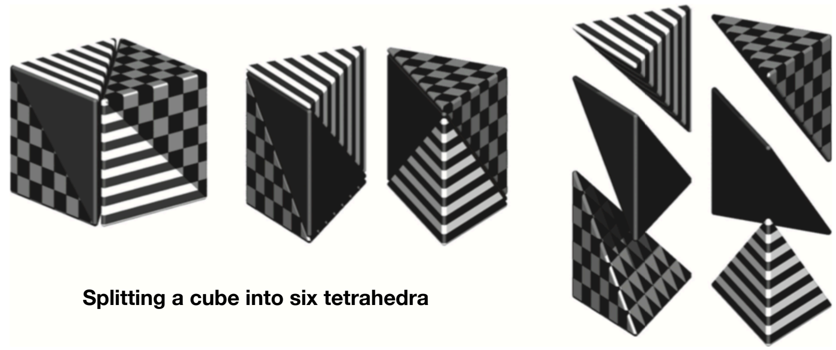

4 Above
4 Below
7 cases
Case 0
Case 1
Case 2
Case 3
Case 4
Case 5
Case 6
Case 7
Case 8
Case 9
Case 10
Case 11
Case 12
Case 13
Case 14

Building a Mesh from Cell Data

- After computing the geometric piece for each cell, you can render this geometric as is.
- Adjacent cells agree on the placement of vertices because they interpolate the same values
- But, can also group vertices and build a manifold mesh by tracking shared edges

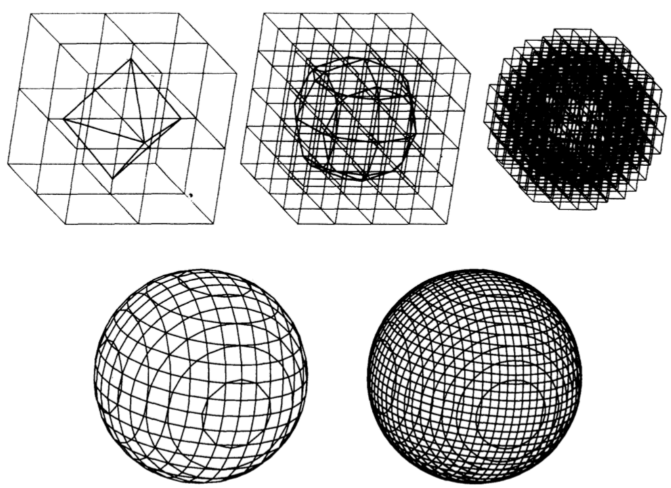


One Can Also Resolve Ambiguities by Tetrahedralization



Splitting a cube into six tetrahedra

Control the resolution of surface based on the resolution of the grid



Bloomenthal, Polygonization of implicit surfaces, 1988

Some Final Thoughts on Implicit Shapes

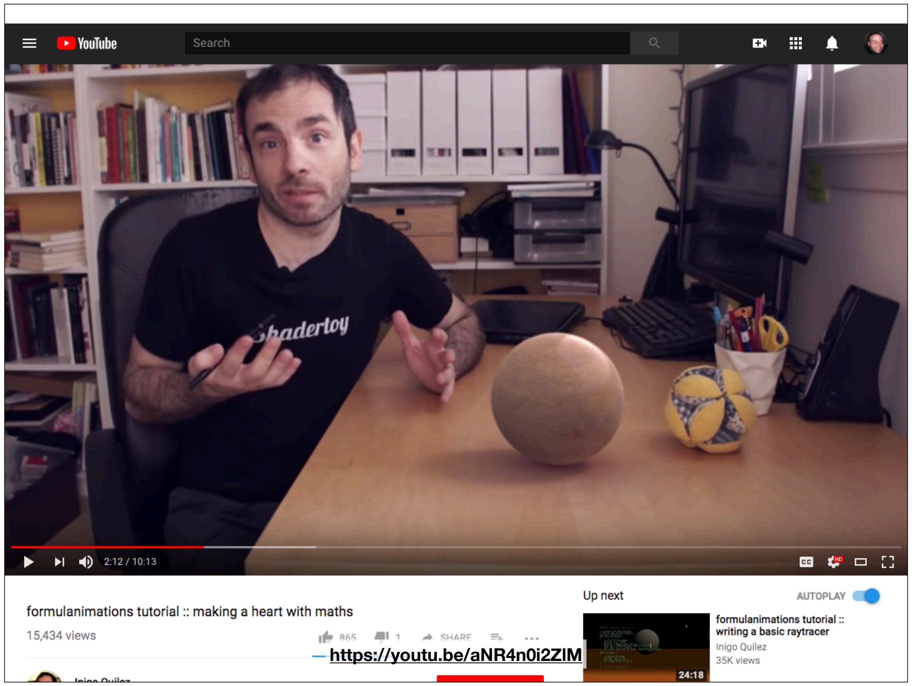
Signed Distance Fields from Triangle Meshes

- Given a triangle mesh, could also construct a distance field stored as a voxel grid
- Interpolating on this would approximate the surface
- Distance to a triangle? Project to the plane of the triangle and then check barycentric coordinates (this can be made faster as well)
- Implicit function is minimum distance over all triangles

Implicits in Animation

- Numerous methods use level set techniques to perform animation
- Basic idea: each animation step updates the implicit function (and other information too).
- Rendering by techniques described today

Enright, Marschner, Fedkiw. Animation and Rendering of Complex Water Surfaces, SIGGRAPH 2002
http://physbam.stanford.edu/~fedkiw/



https://youtu.be/aNR4n0i2ZIM