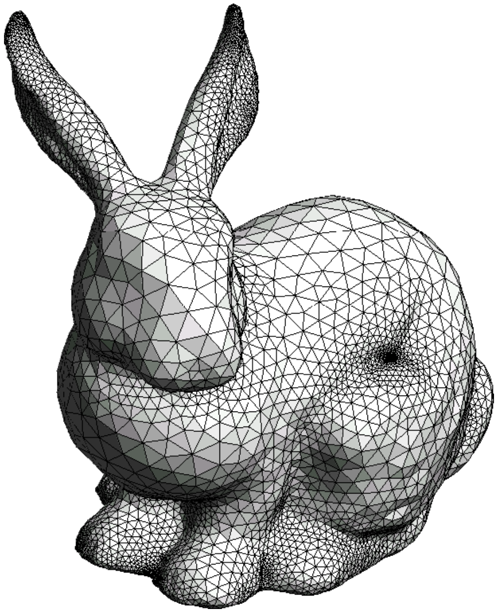


Geometric Modeling in Graphics

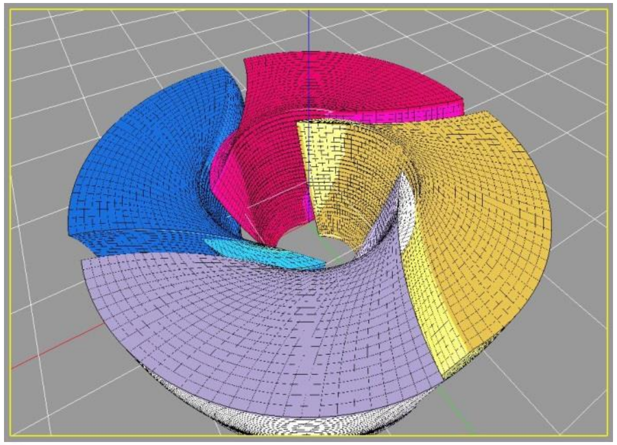
Part 8: Volumes

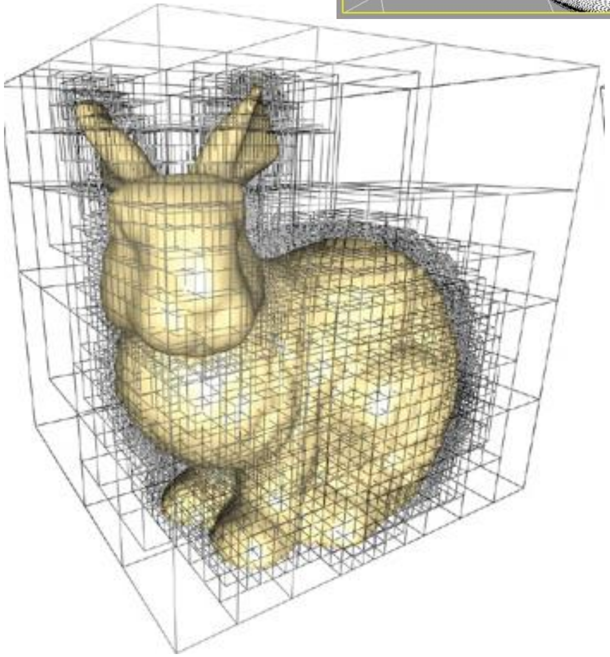
Martin Samuelčík
www.sccg.sk/~samuelcik
samuelcik@sccg.sk

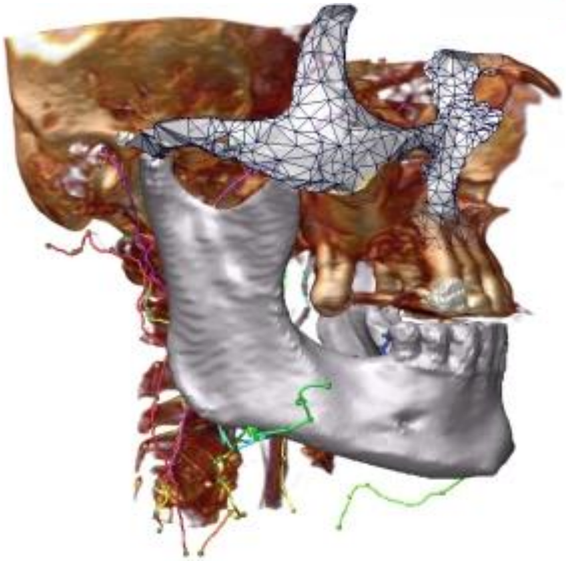
Volumes

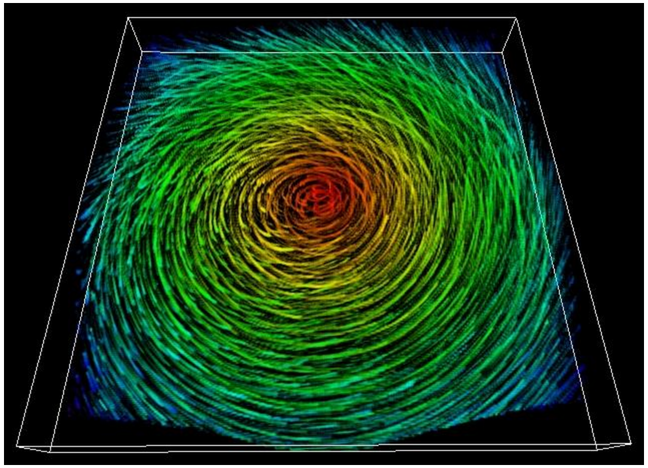
- 3D set of points, embedded in space E^3
- Representing also interior of object
- Discrete grid
  - Sampling function values in grid points
  - Function – binary, distance, intensities, function values, distance vectors, axial distance vectors, physical properties, …
  - Grid – uniform, octree, tetrahedral, …
- Parametric volumes
  - Set of all points $X \in E^3$ such that $X = f(u,v,w)$,
    $u \in \langle u_0,u_1 \rangle$, $v \in \langle v_0,v_1 \rangle$, $w \in \langle w_0,w_1 \rangle$
- FREP
  - Set of all points $X \in \mathbf{E}^3$ such that $f(X) \leq 0$

Volumes









Distance function

- Function defined as distance of point to object
  - d: $\boldsymbol{R}^3 \rightarrow \boldsymbol{R}^+$
- For set Σ, distance function without sign is

$$\operatorname{dist}_\Sigma(\mathbf{p}) = \inf_{\mathbf{x}\in\Sigma} \|\mathbf{x} - \mathbf{p}\|.$$

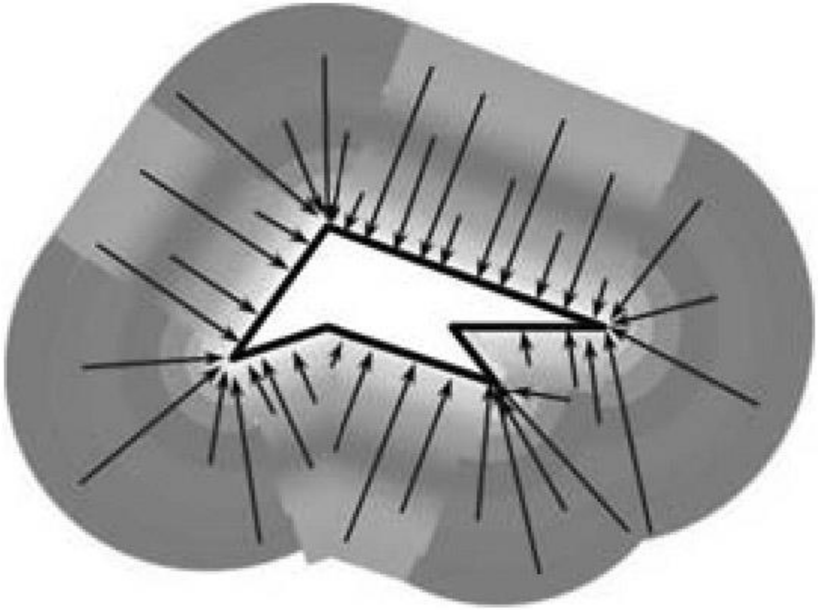
- Extension - distance vectors
- Distance to object with sign

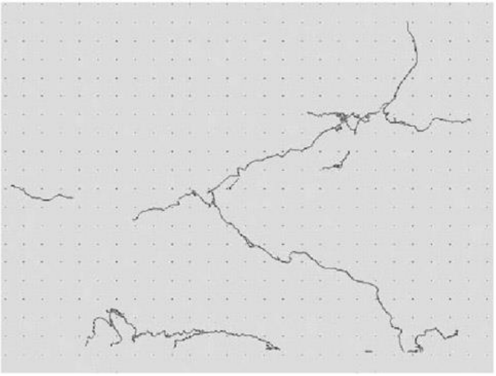
$$\mathrm{d}_S(\mathbf{p}) = \operatorname{sgn}(\mathbf{p}) \inf_{\mathbf{x}\in\partial S} \|\mathbf{x} - \mathbf{p}\|,$$

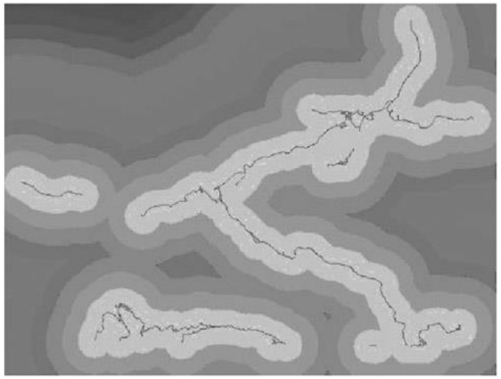
where

$$\operatorname{sgn}(\mathbf{p}) = \begin{cases} -1 & \text{if } \mathbf{p} \in S \\ 1 & \text{otherwise.} \end{cases}$$

Distance function







Distance function

- Isosurface for isovalue τ: $\{\mathbf{p} | \mathrm{d}(\mathbf{p}) = \tau\}$
- Surface, boundary of object is isosurface for isovalue 0
- Vector of first order derivative, gradient $\|\nabla \mathrm{d}\| = 1$
  - Perpendicular to isosurface at point – <u>normal</u> approximation
- Hessian

$$H = \begin{pmatrix} \mathrm{d}_{xx} & \mathrm{d}_{xy} & \mathrm{d}_{xz} \\ \mathrm{d}_{yx} & \mathrm{d}_{yy} & \mathrm{d}_{yz} \\ \mathrm{d}_{zx} & \mathrm{d}_{zy} & \mathrm{d}_{zz} \end{pmatrix}$$

- Mean curvature

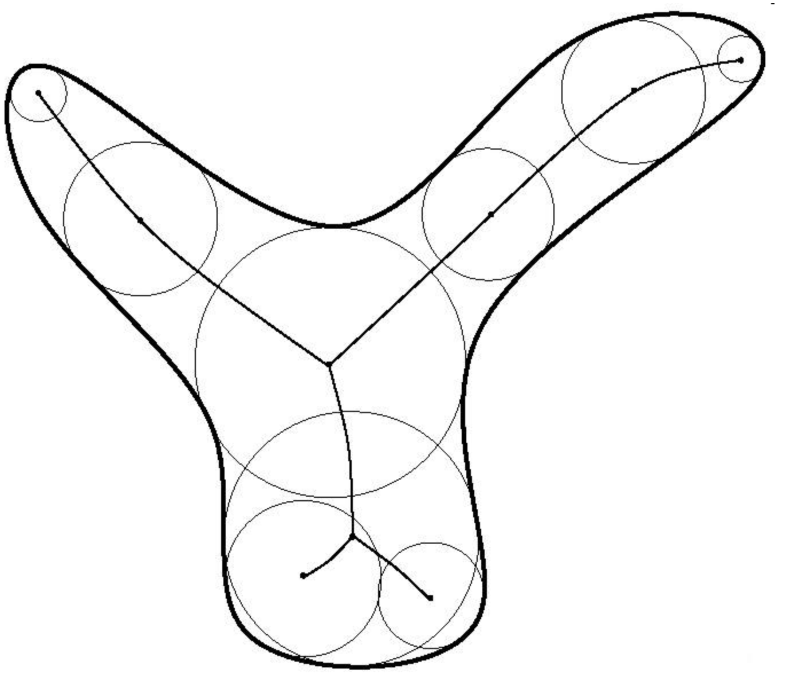
$$\kappa_M = \frac{1}{2}\left(\mathrm{d}_{xx} + \mathrm{d}_{yy} + \mathrm{d}_{zz}\right)$$

- Gauss curvature

$$\kappa_G = \begin{vmatrix} \mathrm{d}_{xx} & \mathrm{d}_{xy} \\ \mathrm{d}_{yx} & \mathrm{d}_{yy} \end{vmatrix} + \begin{vmatrix} \mathrm{d}_{xx} & \mathrm{d}_{xz} \\ \mathrm{d}_{zx} & \mathrm{d}_{zz} \end{vmatrix} + \begin{vmatrix} \mathrm{d}_{yy} & \mathrm{d}_{yz} \\ \mathrm{d}_{zy} & \mathrm{d}_{zz} \end{vmatrix}$$

Distance function

- Distance function is continuous – C^0
- Problem points – points that have same distance from at least two different points on object's surface – cut locus
- Function is C^1 except points from cut locus
- For C^k surface, distance function is C^k in some neighborhood of point on surface



Discretization

- Distance function sampling – distance field
- Topology of sample points
  - Uniform grid
  - Octree
  - Tetrahedral grid
  - Octahedral grid
- Voxel, Cell – volume element of sampling grid
- Voxelization
- Criterion of representation – distance of all cut locus points is larger than sampling resolution
- Approximating gradient

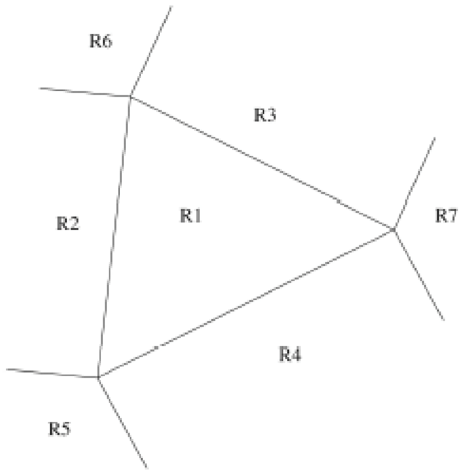
$$\begin{aligned} g^x_{i,j,k} &= d_{i+1,j,k} - d_{i-1,j,k} \\ g^y_{i,j,k} &= d_{i,j+1,k} - d_{i,j-1,k} \\ g^z_{i,j,k} &= d_{i,j,k+1} - d_{i,j,k-1} \\ n_{i,j,k} &= \frac{g_{i,j,k}}{\|g_{i,j,k}\|}\,. \end{aligned}$$

Voxelization

- Given surface of object S
- Definition and placement of grid points
- Topology of grid points – shape of grid
- For each grid point G, computation of distance G from S
- Based on representation of S
  - Polyhedral mesh
    - Point-polygon distance computation
  - Implicit surface
    - Direct approximation, numerical solution
  - Parametric surface
    - Numerical methods for distance computation
- Computation only near surface of S, then fast propagation of distance values to remaining grid points

Mesh voxelization

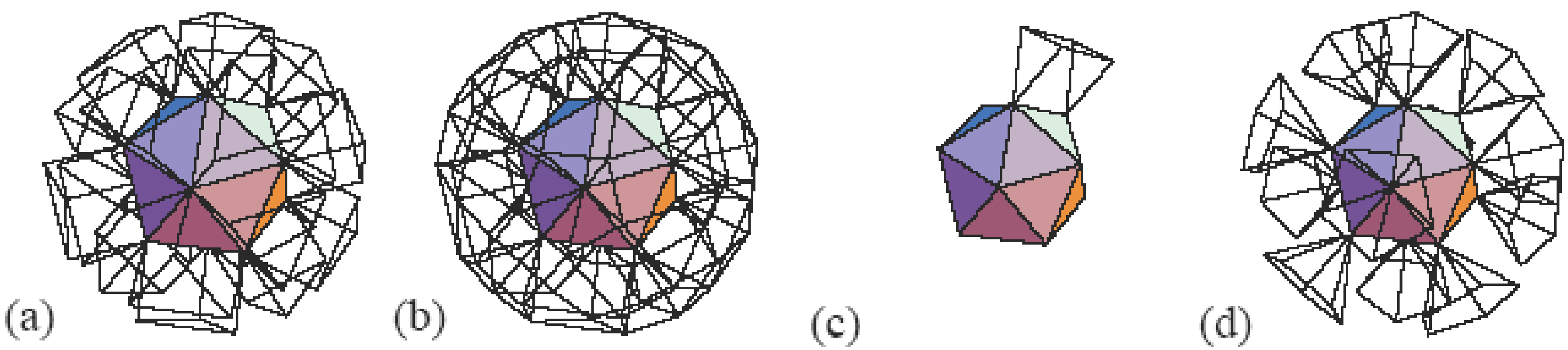
- Given closed triangular 2-manifold mesh
- Choosing triangle that is closest to given grid point
  - Optimization using bounding volumes, grid, octrees
- Computation of grid point – triangle distance
- 7 cases of grid point projection to triangle plane



- given triangle ABC, given grid point S, looking for distance DST
- using barycentric cooridnates of point X in triangle plane, X = uA+vB+wC, u+v+w=1
- let a=(A-C,A-C), b=(A-C,B-C), c = (B-C,B-C), d = (A-C,C-S), e=(B-C,C-S), f=(C-S,C-S)
- let SX is projection of S to plane of triangle ABC, SX=sA+tB+(1-s-t)C
- $s=(be-cd)/(ac-b^2)$, $t=(bd-ae)/(ac-b^2)$
- if $0 \le s \le 1$, $0 \le t \le 1$, $0 \le 1-s-t \le 1$, then SX is inside triangle ABC, and VZD=|S,SX|
- else if $s<0$ or $s>1$, then find point SXS as projection of point SX to line BC
  - SXS=pB+(1-p)C, p=(SX-C,B-C)/(B-C,B-C)
  - if $p<0$, then VZD=|S,C|
  - if $0 \le p \le 1$, then VZD=|S, SXS|
  - if $p>1$, then VZD=|S,B|
- similarly for $t<0$, $t>1$, $(1-s-t)<0$, $(1-s-t)>1$

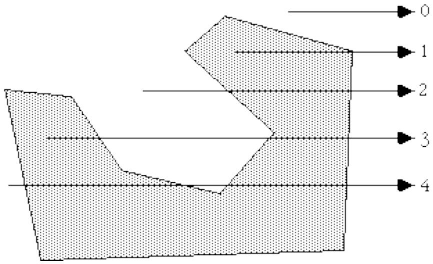
Mesh voxelization

- Local methods
- Computing exact distance only for grid points in close vicinity of mesh surface
- Extruded objects for vertices, edges and triangles of mesh
- Identifying grid points lying inside extruded objects
- Simple computation of distance for points inside extruded objects

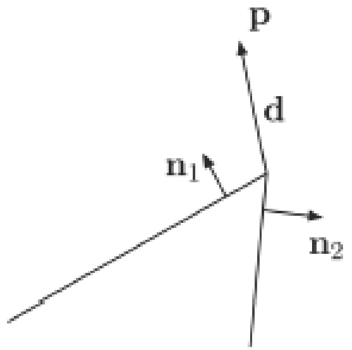


Mesh voxelization

- Sign computation
- Determining if grid point is inside or outside of object
- Number of intersections between arbitrary ray from grid point and mesh boundary
  - Odd number of intersections – inside
  - Even number of intersection – outside



- For C^1 surfaces, dot product of normal and distance vector
  - If dot product is positive, grid point is inside
  - Mesh – not C^1 – using angle-weighted pseudo-normals for edges and vertices of mesh



Distance transforms

- Propagation of distance values in computed grid points to remaining unprocessed grid points
- Grid propagation:
  - Sweeping – uniform slices propagation
  - Wavefront – from surface to higher distances
- Computation for voxel:
  - Chamfer:
    - New distance in grid point is computed from already known distances in neighboring grid points
  - Vector:
    - New distance vector in grid point is computed from already known distance vectors in neighboring grid points
  - Eikonal:
    - Distance in grid points is filled using iterative solution of differential equation

Distance transforms

- Initialization – computation of distance for grid points near surface of object

$$F(\mathbf{p}) = \begin{cases} 0 & \mathbf{p} \text{ is exterior} \\ \infty & \mathbf{p} \text{ is interior,} \end{cases} \qquad F(\mathbf{p}) = \begin{cases} d_S(\mathbf{p}) & \text{in the shell} \\ \infty & \text{elsewhere.} \end{cases}$$

- Chamfer methods
  - Sweeping

```
/* Forward Pass */
FOR(z = 0; z < f_z; z++)
  FOR(y = 0; y < f_y; y++)
    FOR(x = 0; x < f_x; x++)
      F[x,y,z] =
  inf_{∀i,j,k∈fp}(F[x+i,y+j,z+k]+m[i,j,k])

/* Backward Pass */
FOR(z = f_z-1; z ≥ 0; z--)
  FOR(y = f_y-1; y ≥ 0; y--)
    FOR(x = f_x-1; x ≥ 0; x--)
      F[x,y,z] =
  inf_{∀i,j,k∈bp}(F[x+i,y+j,z+k]+m[i,j,k])
```

Forward pass

	f		f	
f	e	d	e	f
	d		d	
f	e	d	e	f
	f		f	

z=-2

f	e	d	e	f
e	c	b	c	e
d	b	a	b	d
e	c	b	c	e
f	e	d	e	f

z=-1

	d		d	
d	b	a	b	d
	a	0		

z=0

Backward pass

		0	a	
d	b	a	b	d
	d		d	

z=0

f	e	d	e	f
e	c	b	c	e
d	b	a	b	d
e	c	b	c	e
f	e	d	e	f

z=1

	f		f	
f	e	d	e	f
	d		d	
f	e	d	e	f
	f		f	

z=2

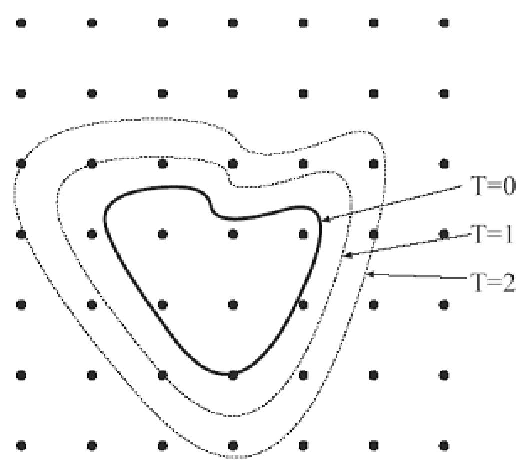
Transform	a	b	c	d	e	f
City Block (Manhattan)	1					
Chessboard	1	1				
Quasi-Euclidean $3 \times 3 \times 3$	1	$\sqrt{2}$				
Complete Euclidean $3 \times 3 \times 3$	1	$\sqrt{2}$	$\sqrt{3}$			
$< a, b, c >_{opt}$ $3 \times 3 \times 3$[102]	0.92644	1.34065	1.65849			
Quasi-Euclidean $5 \times 5 \times 5$	1	$\sqrt{2}$	$\sqrt{3}$	$\sqrt{5}$	$\sqrt{6}$	3

  - Wavefront – priority queue for grid points with minimal distance

Fast marching method

- Eikonal distance transform
- Simulating expanding surface with constant speed – inflating balloon
- Time of surface (balloon) arrival to grid point – distance
- T – time of arrival to grid point x
- F – speed of surface expansion in x
  - F is constant

$$||\nabla T(\mathbf{x})|| F(\mathbf{x}) = 1$$



Fast marching method

- „frozen" point – final distance was computed for point
- „narrow band" point – there is some distance computed, but is not final
- H – set of „narrow band" points, priority queue

```
Initialization()
{
    for each voxel v in I
        {
            Freeze v;
            for each neighbour vn of v
                {
                    compute distance d at vn;
                    if vn is not in narrow band
                        {
                            tag vn as narrow band;
                            insert (d,vn) in H;
                        }
                    else
                        decrease key of vn in H to d;
                }
        }
}
```

```
Loop()
{
    while H is not empty
        {
            Extract v from top of H;
            Freeze v;
            for each neighbour vn of v
                if vn is not frozen
                    {
                        compute distance d at vn;
                        if vn is not in narrow band
                            {
                                tag vn as narrow band;
                                insert (d,vn) in H;
                            }
                        else
                            decrease key of vn in H to d;
                    }
        }
}
```

Fast marching method

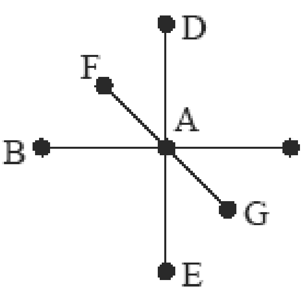
Computation of distance for grid point from neighboring point distances using constant gradient

$$1/F^2 = \begin{cases} \max(D_2^{-x}G, -D_2^{+x}G, 0)^2 + \\ \max(D_2^{-y}G, -D_2^{+y}G, 0)^2 + \\ \max(D_2^{-z}G, -D_2^{+z}G, 0)^2 \end{cases}$$

$$||\nabla T||^2 = \begin{cases} \max(V_A - V_B, V_A - V_C, 0)^2 & + \\ \max(V_A - V_D, V_A - V_E, 0)^2 & + \\ \max(V_A - V_F, V_A - V_G, 0)^2 \end{cases}$$

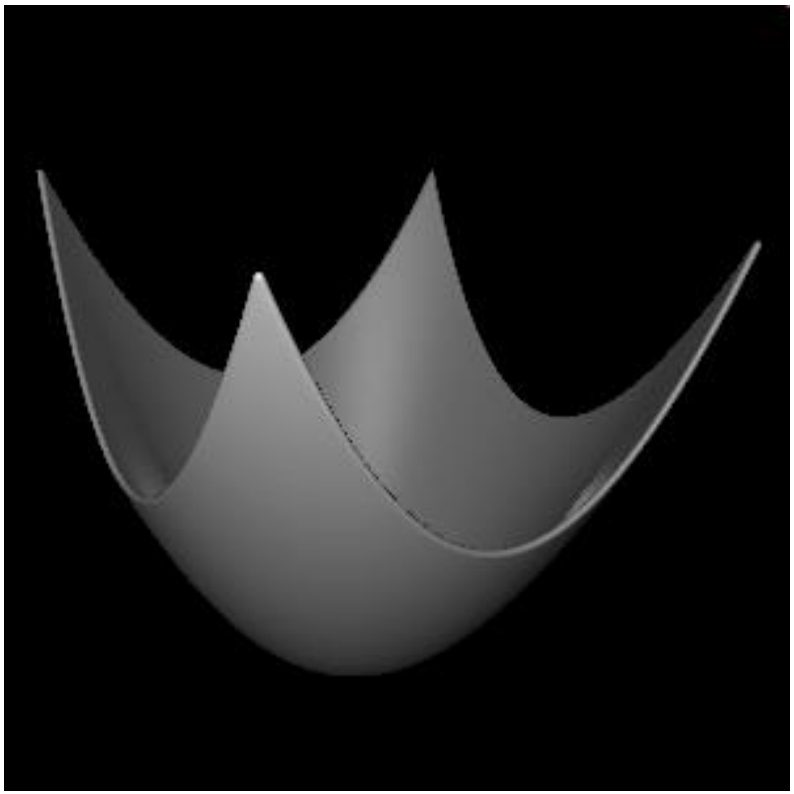
$$D_2^{-x}G = \frac{3G[x,y,z] - 4G[x-1,y,z] + G[x-2,y,z]}{2}$$

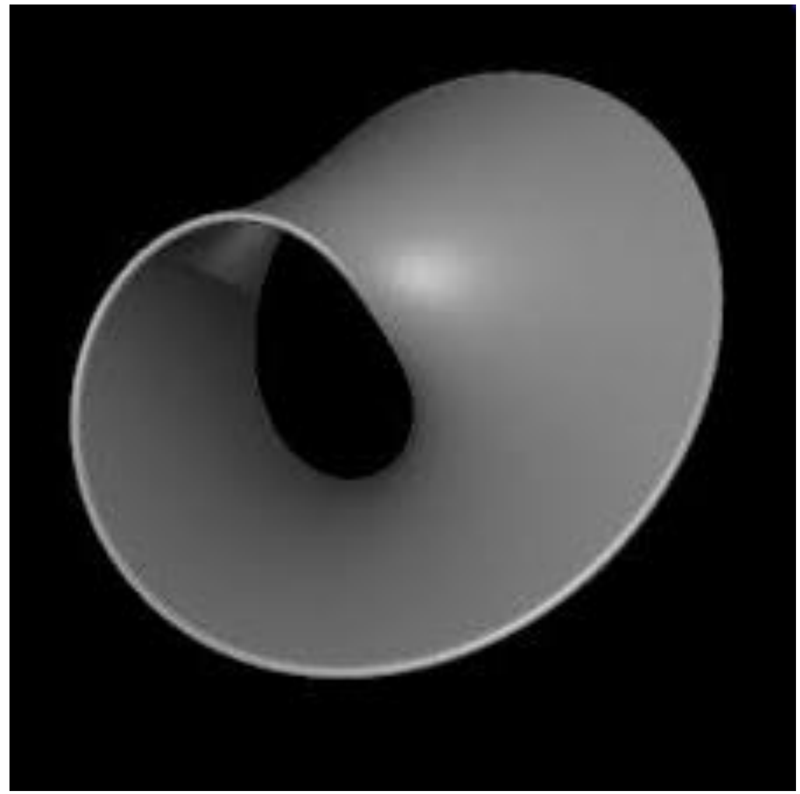
$$D_2^{+x}G = -\frac{3G[x,y,z] - 4G[x+1,y,z] + G[x+2,y,z]}{2}$$



Parametric surface voxelization

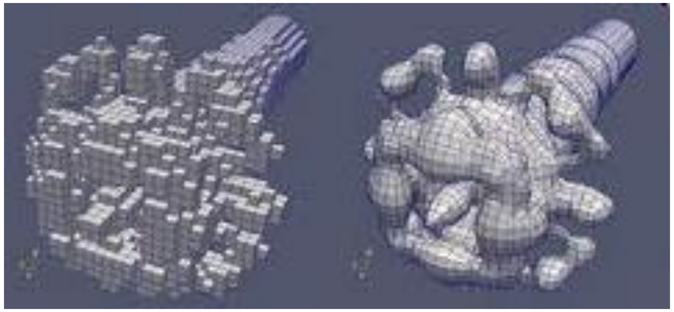
- Conversion to polyhedral or implicit representation
- Minimization of $\mathrm{d}(u,v) = ||\mathbf{S}(u,v) - \mathbf{p}||$ using numerical iterative solutions





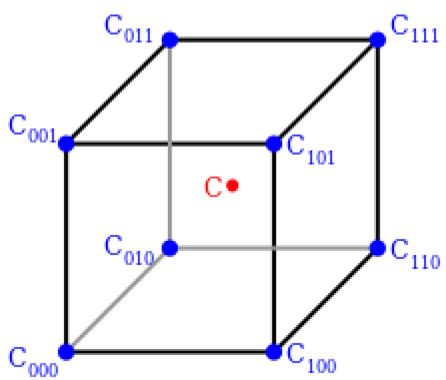
Implicit surface voxelization

- Isosurface of function $\{X \in E^3; f(X) = 0\}$
- For some surface, it is sufficient to sample just f
- Sampling function $\frac{f}{\|\nabla f\|}$
- Iteratively find closest point to grid point on implicit surface in the gradient direction
  - Let (x_0, y_0, z_0) is given grid point
  - $(x_{i+1}, y_{i+1}, z_{i+1}) = (x_i, y_i, z_i) - \frac{f(x_i,y_i,z_i)}{f_x(x_i,y_i,z_i)^2+f_y(x_i,y_i,z_i)^2+f_z(x_i,y_i,z_i)^2}(f_x(x_i, y_i, z_i), f_y(x_i, y_i, z_i), f_z(x_i, y_i, z_i))$
  - Finish when one iteration does not change position of approximation so much



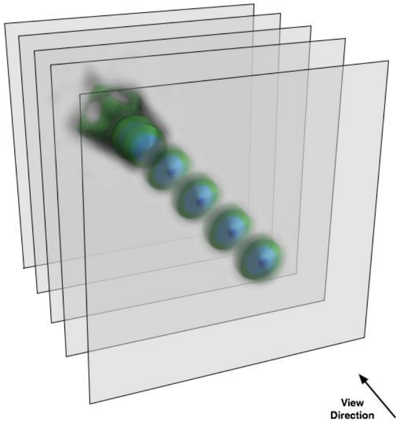
Interpolation

- Approximation of distance function for arbitrary space point from grid values - interpolating grid values
- Nearest neighbor interpolation
  - Given space point C, find grid point G that is closest to C
  - $d(C) = d(G)$
- Trilinear interpolation
  - Given space point C, find voxel **V** where it is located
  - Compute C as linear combination **V**'s corner points
  - $\lambda_1 = (C - C_{000}, C_{100} - C_{000})$
  - $\lambda_2 = (C - C_{000}, C_{010} - C_{000})$
  - $\lambda_3 = (C - C_{000}, C_{001} - C_{000})$
  - $C = w_{000}C_{000} + w_{100}C_{100} + \ \dots + w_{111}C_{111}$
  - $w_{ijk} = (1-\lambda_1)^{1-i}{\lambda_1}^i(1-\lambda_2)^{1-j}{\lambda_2}^j(1-\lambda_3)^{1-k}{\lambda_3}^k$
  - $w_{000} + w_{100} + \ \dots + w_{111} = 1$
  - $d(C) = w_{000}d(C_{000}) + w_{100}d(C_{100}) + \ \dots + w_{111}d(C_{111})$
- Tricubic interpolation



Visualization

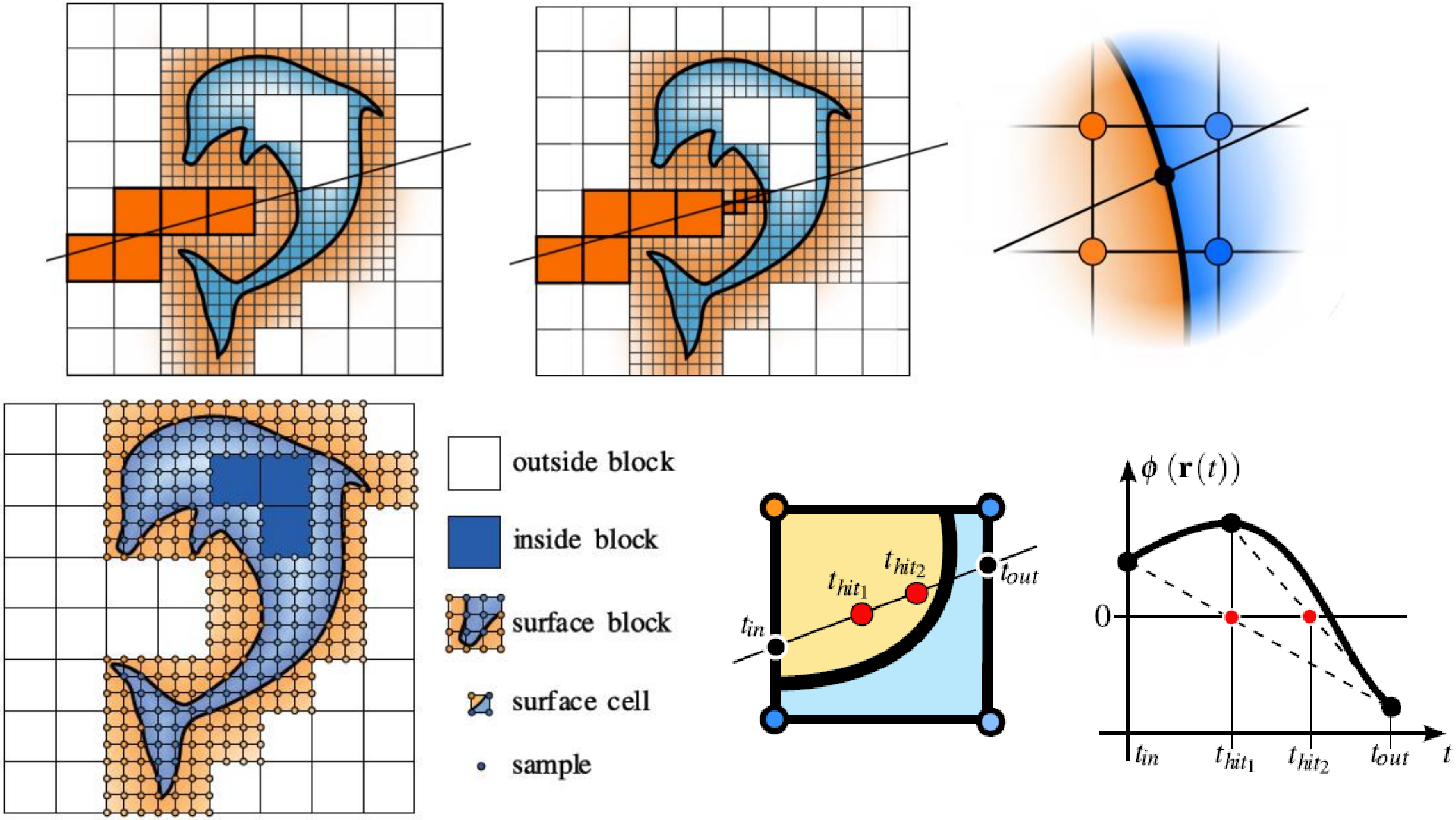
- Conversion to other representations
  - Polyhedral representation – marching cubes
  - Point clouds – projections of grid point onto surface in the opposite direction of gradient
- Direct visualization
  - Slicing



- Raytracing
  - Traversal of grid along ray
  - Finding first voxel containing isosurface
  - Using distance function interpolation to find more accurate intersection
  - Using subsampled values for finer approximation
- Points sampling
  - Approximation of closest point on surface $\mathbf{p}_f = \mathbf{p} - \nabla d_S(\mathbf{p}) d_S(\mathbf{p})$

Raytracing

http://dcgi.felk.cvut.cz/_media/en/events/praguecvut-jamriska-ondrej.pdf

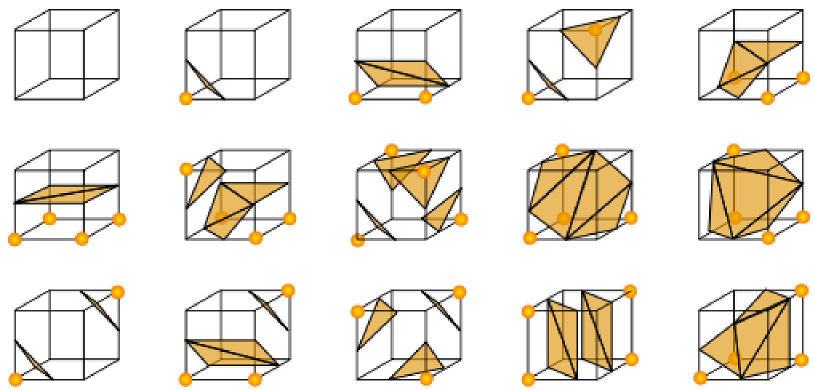


Marching cubes

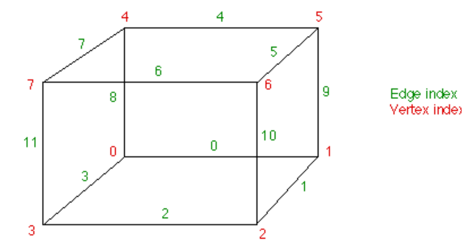
- Generating set of triangles that approximate isosurface of distance function for given isovalue τ
- Generating triangles for each voxel separately
  - Get 8 grid values in corners of voxel
  - Mark each corner C as inside or outside by comparing distance value in corner and isovalue τ
    - Outside - $d(C) \geq \tau$
    - Inside - $d(C) < \tau$
  - For each edge AB of voxel, if it connect inside and outside corner, construct edge vertex using linear interpolation
    - $V_{AB} = \frac{d(B)-\tau}{d(B)-d(A)} A + \frac{\tau-d(A)}{d(B)-d(A)} B$
    - Interpolating gradients in A, B to get normal in V_{AB}
  - Connect all edge vertices in voxel forming several triangles based on configuration of inside and outside corners

Marching cubes

- Basic configurations of inside, outside corners



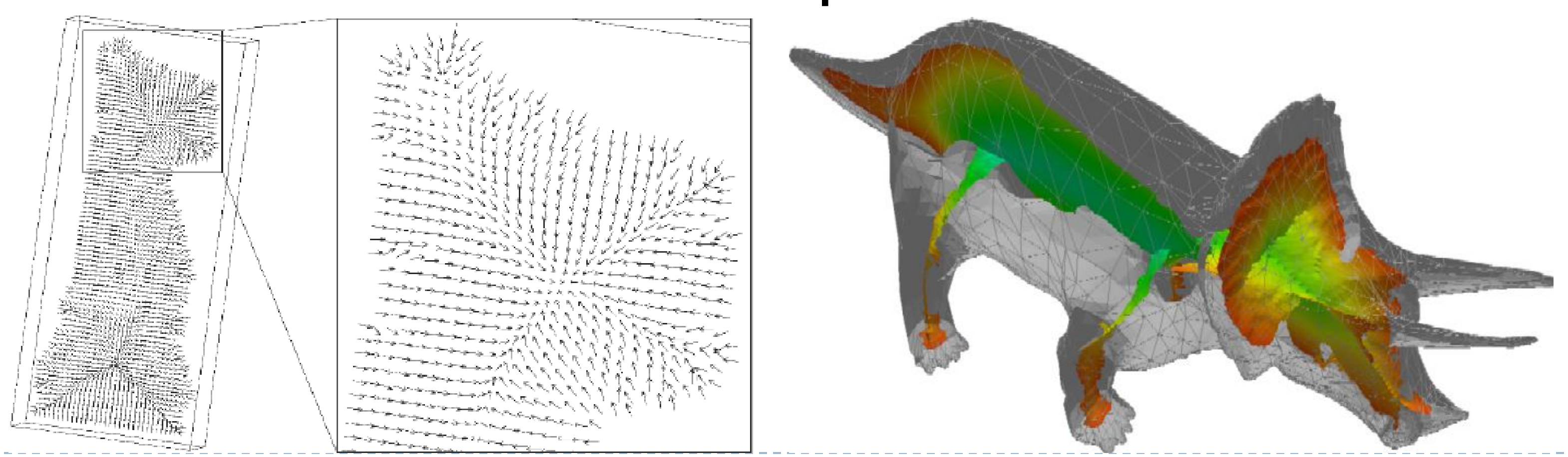
- 256 total configurations (rotation, mirroring)
- Implementation



  - http://paulbourke.net/geometry/polygonise/
  - Preparing vertex code – marking each corner as inside or outside, 8bit
  - Computing edge vertex for each of 12 edges, if necessary
  - Connecting edge vertices into triangles based on vertex code, using lookup table with 256 records, each record has list of edge indices pointing to edge vertices

Skeleton, medial axis

- Simple primitives (line segments) representing shape of whole object with same topological properties
- Detecting cut locus points – discontinuities in derivation of distance function
- Finding extremal values inside object
- Comparison of distance vectors
- Used for skeleton animation, parametrization, …

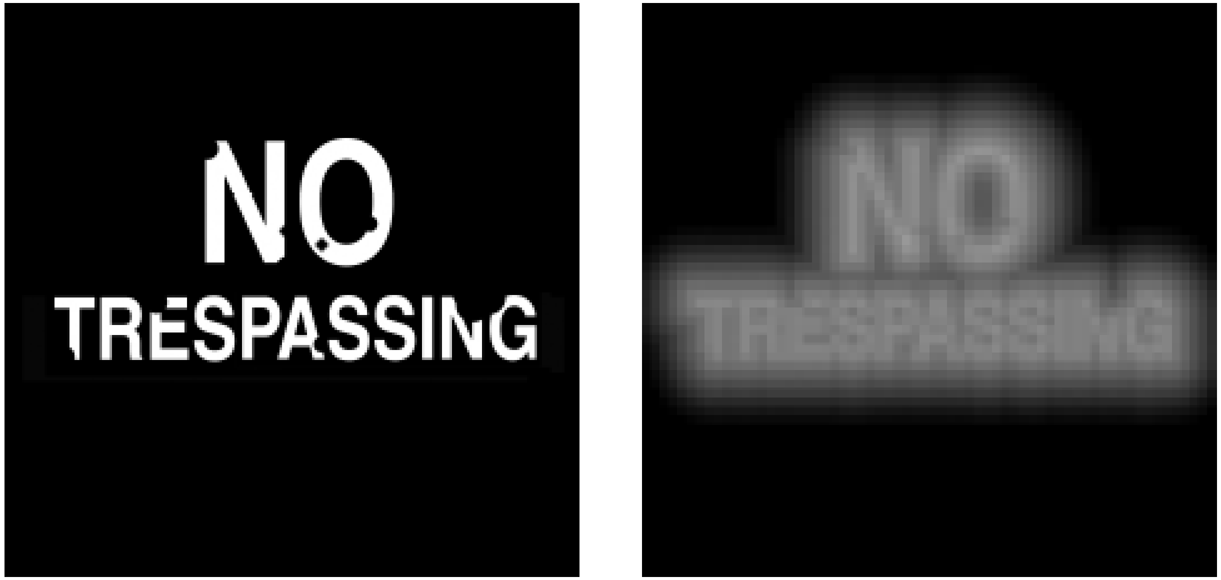


Fonts representation

- http://www.valvesoftware.com/publications/2007/SIGGRAPH2007_AlphaTestedMagnification.pdf
- Using 2D distance field for representation of each glyph
- More precise representation of border
- Easier rendering of border effects

(a) 64x64 texture, alpha-blended (b) 64x64 texture, alpha tested (c) 64x64 texture using our technique

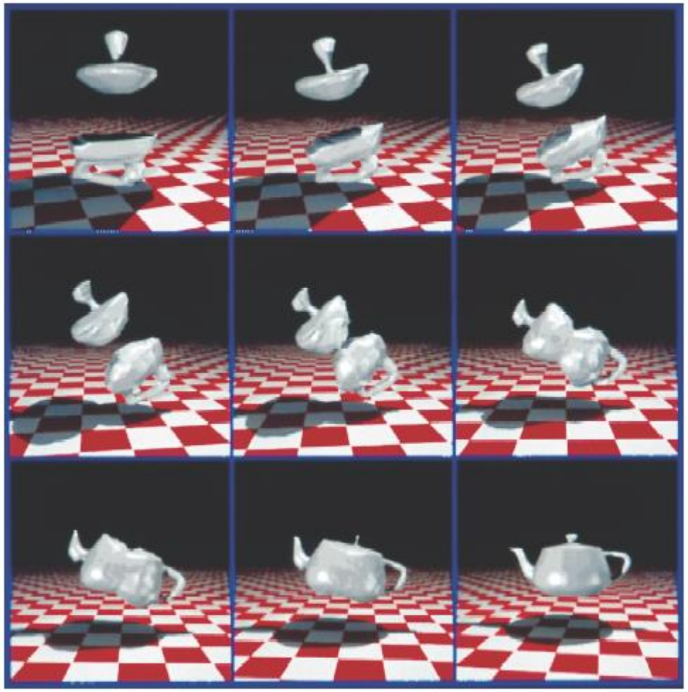
Fonts representation



(a) High resolution input (b) 64x64 Distance field

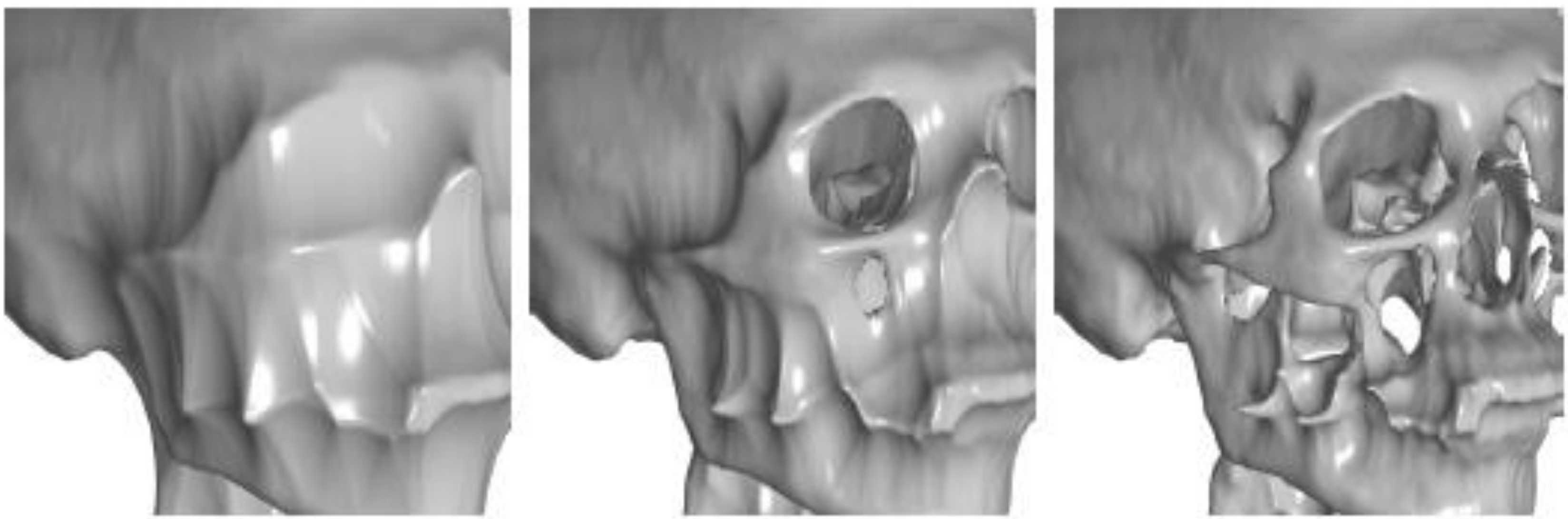
Morphing

- Interpolation between two objects in time
- Compacting representation of given objects – same sampling grid points for both representations
- Linear interpolation of two values in each grid points



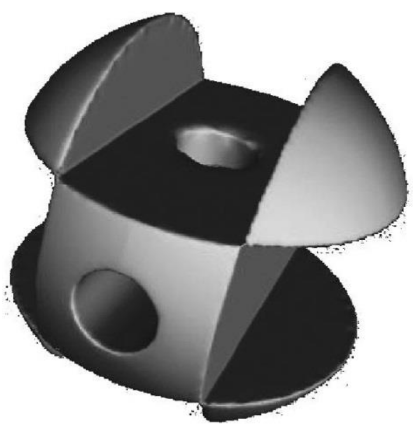
Morphology

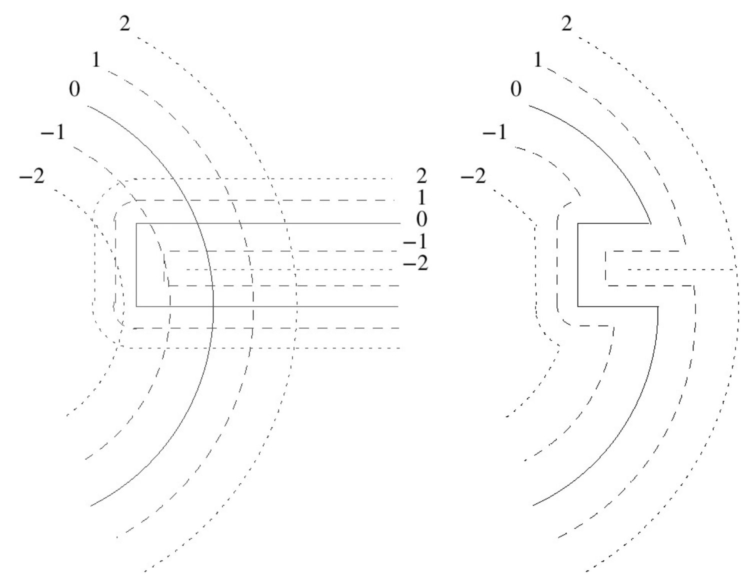
- Operations for discrete signal processing
- Erosion $X \ominus B = \{\mathbf{p} | B_{\mathbf{p}} \subset X\}$
- Dilatation $X \oplus B = \{\mathbf{p} | B_{\mathbf{p}} \cap X \neq \emptyset\}$
- Opening $X \bullet B = (X \oplus B) \ominus B$
- Closing $X \circ B = (X \ominus B) \oplus B.$



CSG operations

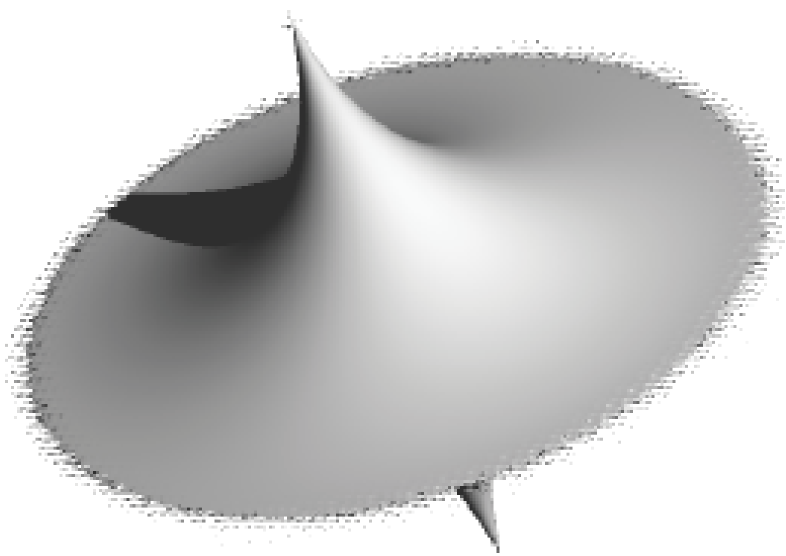
- Simple and fast Boolean operations on two objects
- Distance fields of objects must be compacted – must have same sampling grid points
- Approximation and alias near sharp features
- <u>Union</u> - D=min(D1,D2)
- <u>Intersection</u> - D=max(D1,D2)
- <u>Difference</u> - D=max(D1,-D2)

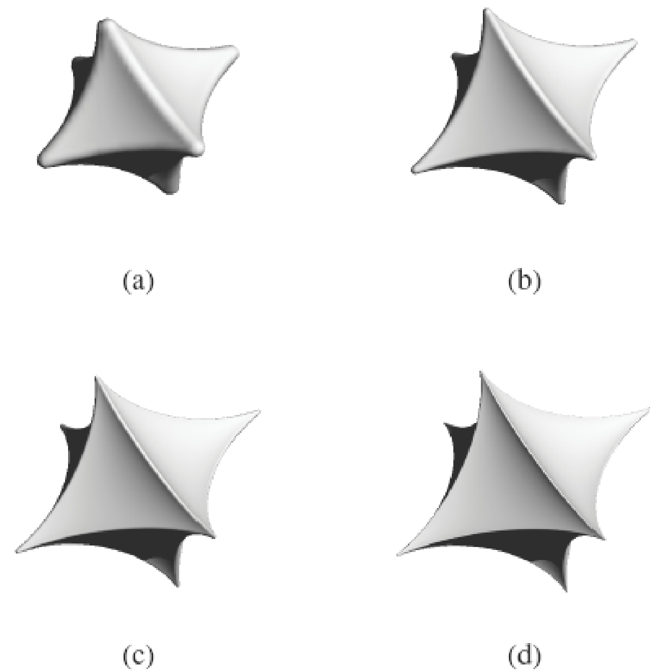




CSG operations repair

- http://www.sccg.sk/~novotny/doc/vg05.pdf
- Improvement of representation after Boolean operation
- Detecting and repairing distance function near sharp features with insufficient sampling density



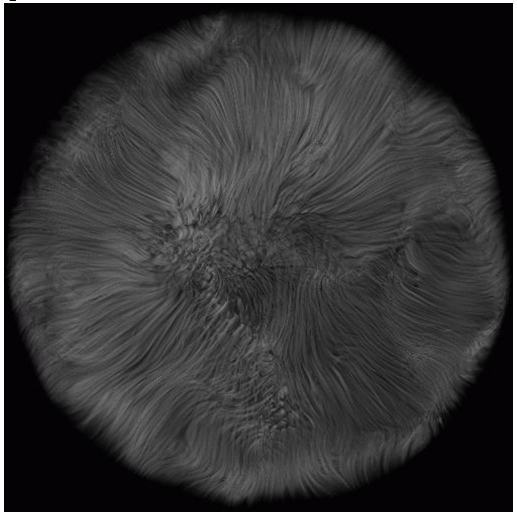


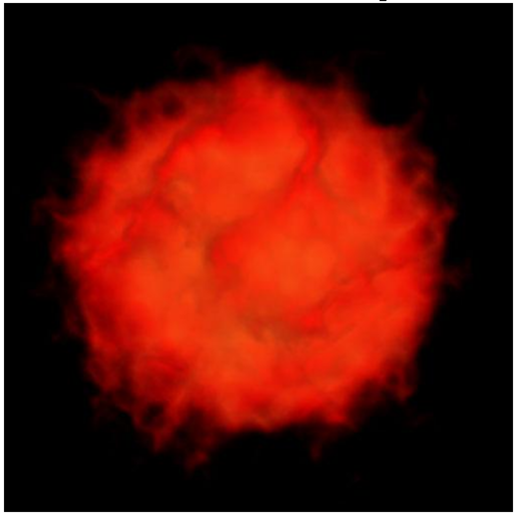
Hypertextures

- Adding rendering details over object surface
- Defining region over surface for texture mapping

$$D(p) = \begin{cases} 1 & \text{if } \mathrm{d}(p)^2 \le r_i^2 \\ 0 & \text{if } \mathrm{d}(p)^2 \ge r_o^2 \\ \frac{r_o^2 - \mathrm{d}(p)^2}{r_o^2 - r_i^2} & \text{otherwise,} \end{cases}$$

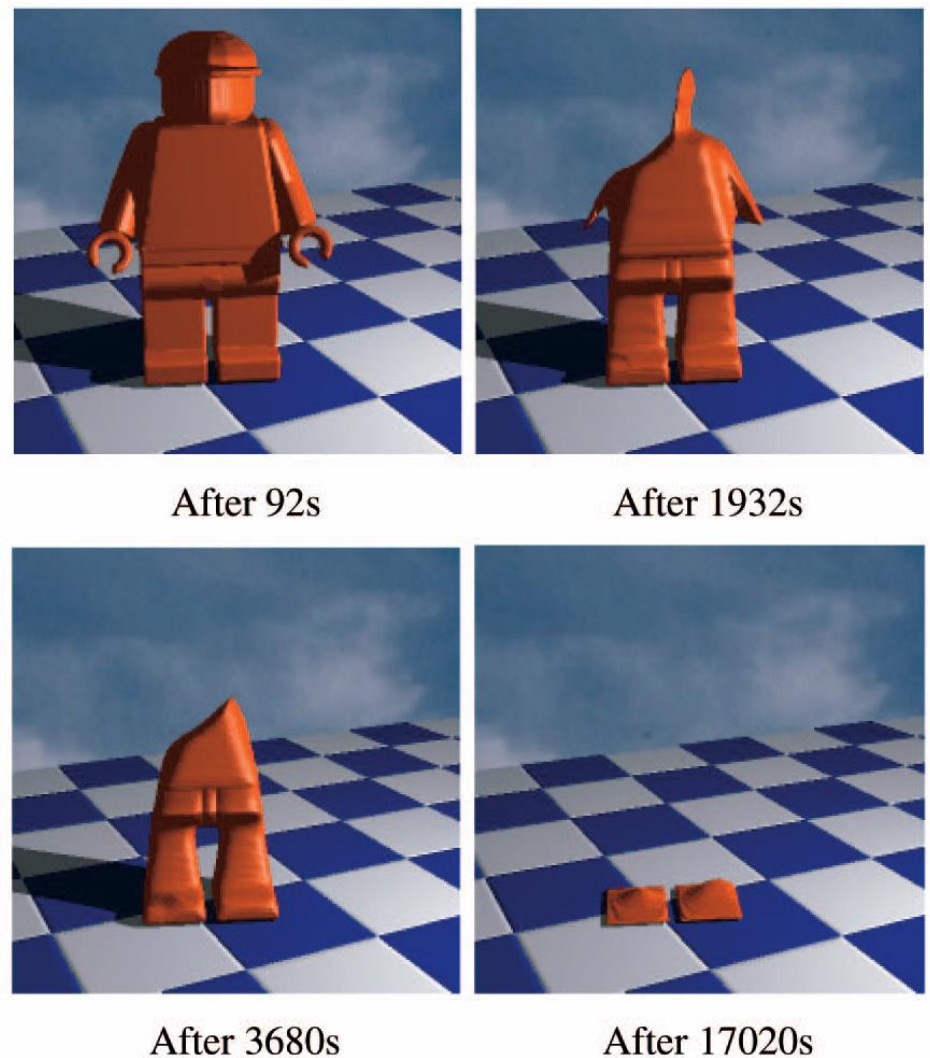
- Using D(p) to obtain data from 3D texture or to generate other properties such like direction, density, tangent plane





Object modeling

- Surface smoothing
- Subdivision
- Parametrization
- Error computation
- Objects comparison
- Collision detection
- Simulations, animation
- Reconstruction



The End
for today