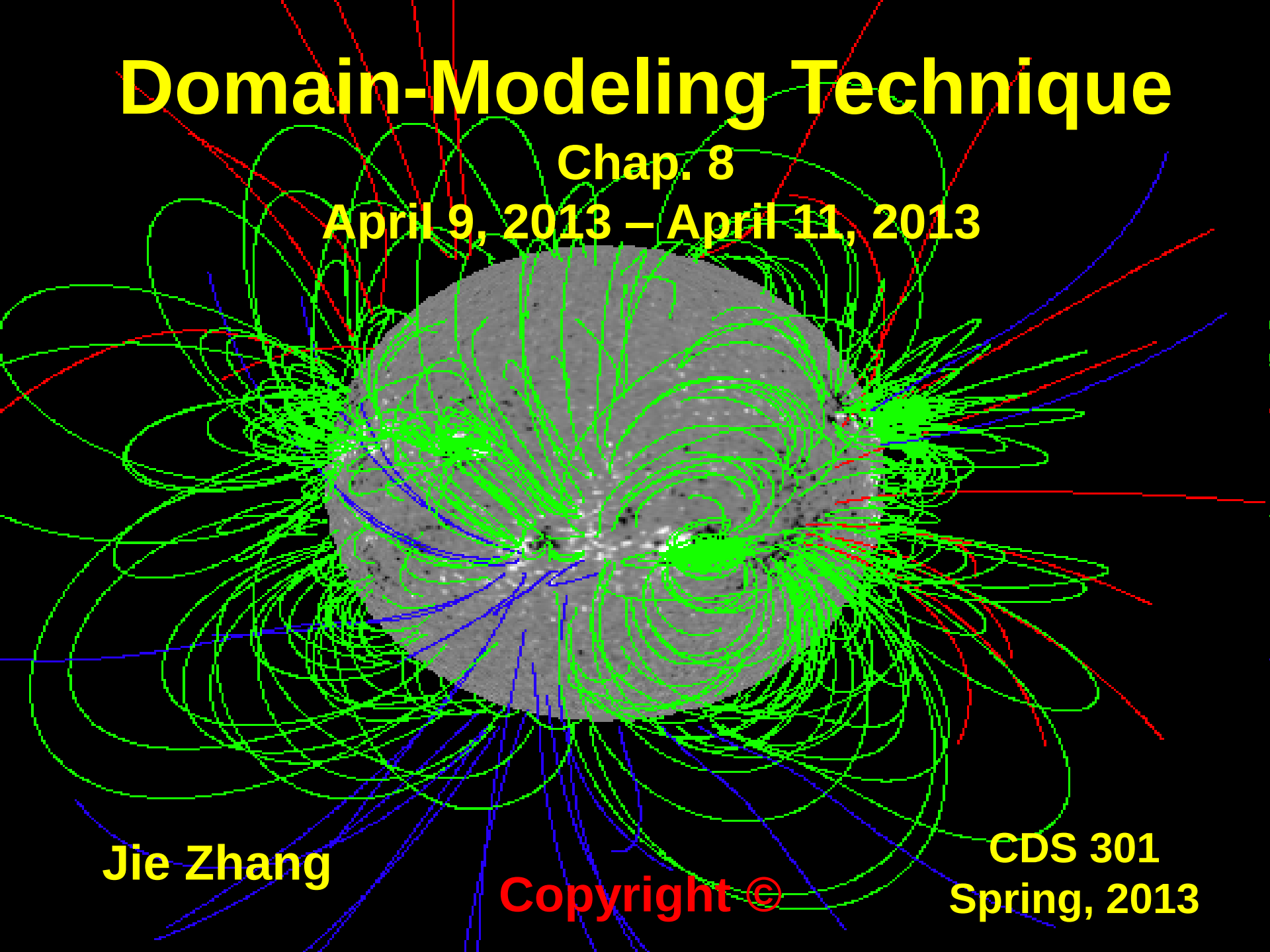
Domain-Modeling Technique

Chap. 8

April 9, 2013 – April 11, 2013



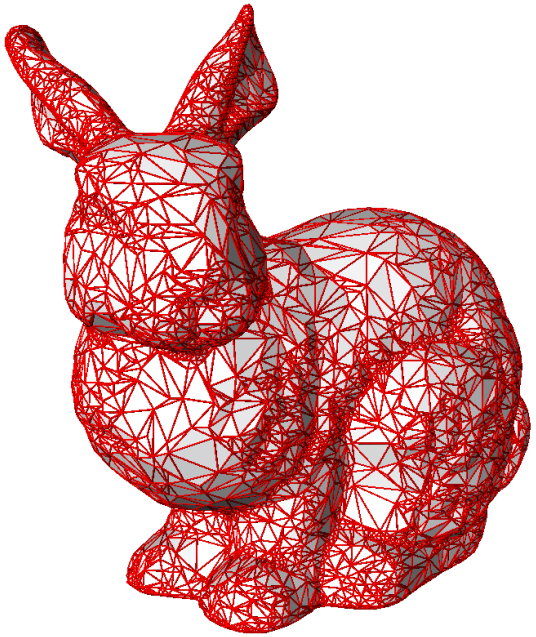
Jie Zhang

CDS 301

Spring, 2013

Outline

8.1. Cutting
8.2. Selection
8.3. Grid Construction from Scattered Points
8.4. Grid-Processing Technique



Domain-Modeling

Visualization is to operate on the domain, e.g., the sampling points and grids, but not on the sampled data.

Discrete dataset

$$\boldsymbol{D_s} = (\{p_i\}, \{C_i\}, \{f_i\}, \{\Phi_i^k\})$$

$$\boldsymbol{D} = (\mathrm{D}, \mathrm{C}, \mathrm{f})$$ **Continuous dataset**

CH8.1 Cutting

April 09, 2013

Cutting

Target domain is a subset of the source domain

Source Dataset:

$$\boldsymbol{D_s} = (\{p_i\}, \{C_i\}, \{f_i\}, \{\Phi_i\})$$

Targe Dataset:

$$\boldsymbol{D_s}' = (\{p_i'\}, \{C_i'\}, \{f_i'\}, \{\Phi_i'\})$$

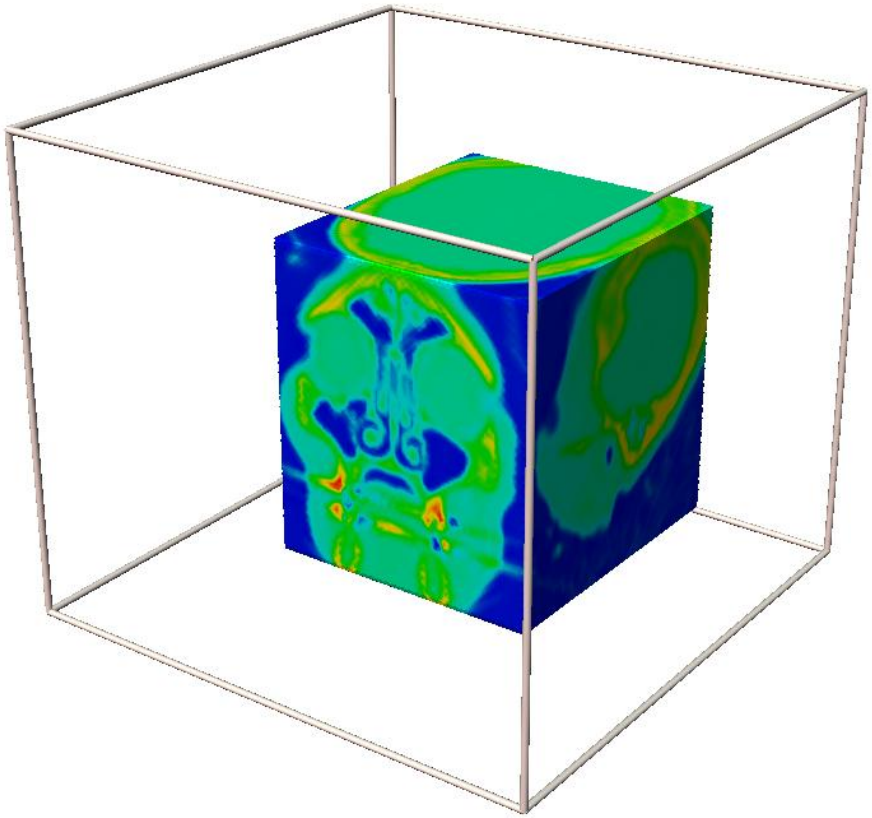
and

$$D' \in D$$

Brick Extracting

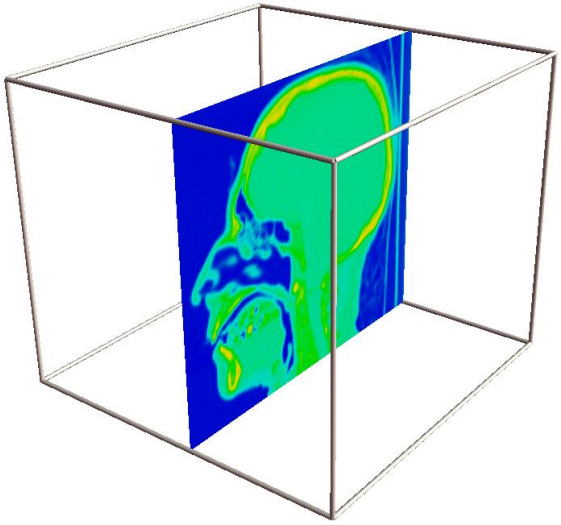
- Target domain has the same dimension of the source domain
- Extracting a volume of interest (VOI)

$$\{p_i'\} \in \{p_i\}$$

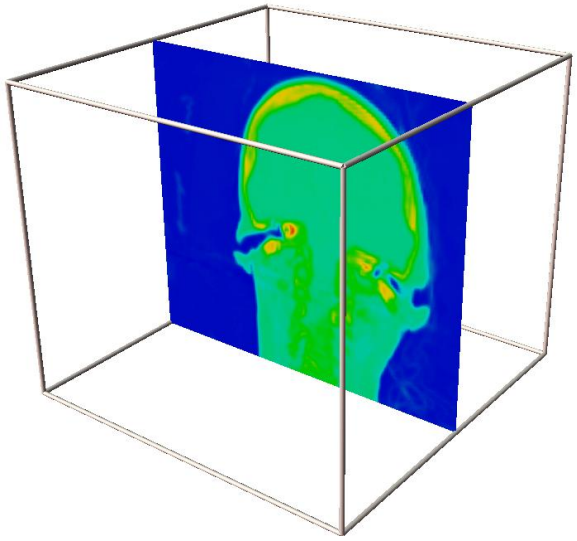


Slicing

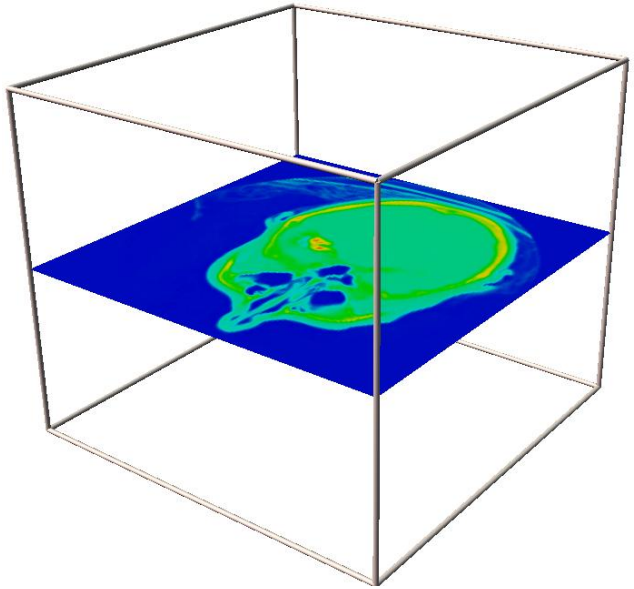
- Target domain has a smaller dimension: d-1
- Fix the coordinate in the slicing dimension



Along X-axis
Sagittal slice



Along Y-axis
axial slice



Along Z-axis
coronal slice

MATLAB: slice

```
>>load mri
>>D_tmp=squeeze(D);    %remove singleton dimension
>>Ds=smooth3(D_tmp);    %smooth 3-D data

>>h=slice(Ds,58,128,1);   %show the slice

>>set(h,'EdgeColor','none')

>>h=slice(Ds,0,0,10)

>>h=contourslice(Ds,1,1,[10,15],100);
```

Implicit Function Cutting

- Generalize the axis-aligned slicing

E.g., cutting along an arbitrary plane

$$Ax + By + Cz + D = 0$$

MATLAB: rotate, surf

```
%define a surface, rotate a surface


%define a surface
>> hsf=surf(linspace(-2,2,20),linspace(-2,2,20),zeros(20))


%rotate a surface
>>rotate(hsf,[1,0,0],30)  %rotate(surf, direction, angle)


>>xd=get(hsf,'XData')
>>yd=get(hsf,'YData')
>>zd=get(hsf,'ZData')
```

MATLAB: slice

```
%show a tilted slice
>>load mri
>>D_tmp=squeeze(D);    %remove singleton dimension
>>Ds=smooth3(D_tmp);    %smooth 3-D data


>>hsf=surf(linspace(1,128,128),linspace(1,128,128),zeros(128,128)+10) ;   %create a horizontal surface
>>rotate(hsf,[0,1,0],-45)   %rotate along Y by 45 degree
>>xd=get(hsf,'XData')
>>yd=get(hsf,'YData')
>>zd=get(hsf,'ZData')
>>h=slice(Ds,xd,yd,zd);   %show
```

CH8.2 Selection

April 09, 2013

Selection

- Cutting explicitly specifies the topology of the target domain
- **Selection explicitly specify the attribute value of the target dataset.**

$$D' = \{ p \in D \mid s(p) = true \}$$

- The output of selection is an unstructured grid
- E.g., **contouring operation, iso-surface**
- E.g., thresholding or segmentation

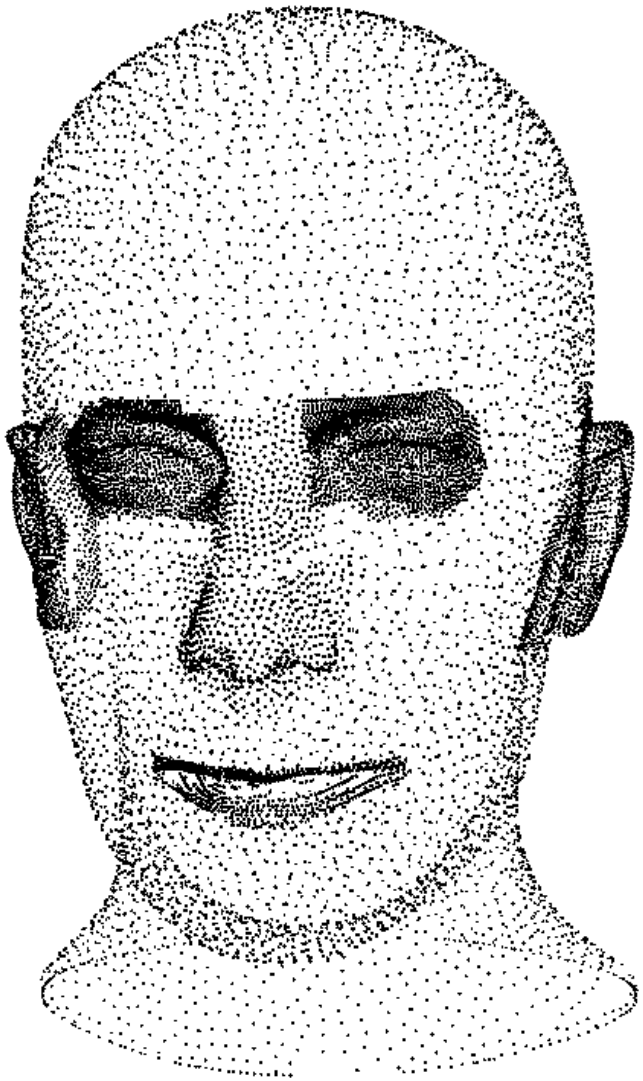
MATLAB: patch

See “MRI_isosurface.m”

CH8.3 Grid Construction from Scattered Points

April 09, 2013

Scattered Points



Gridless point cloud

12,772
3-D points
represent a
human face

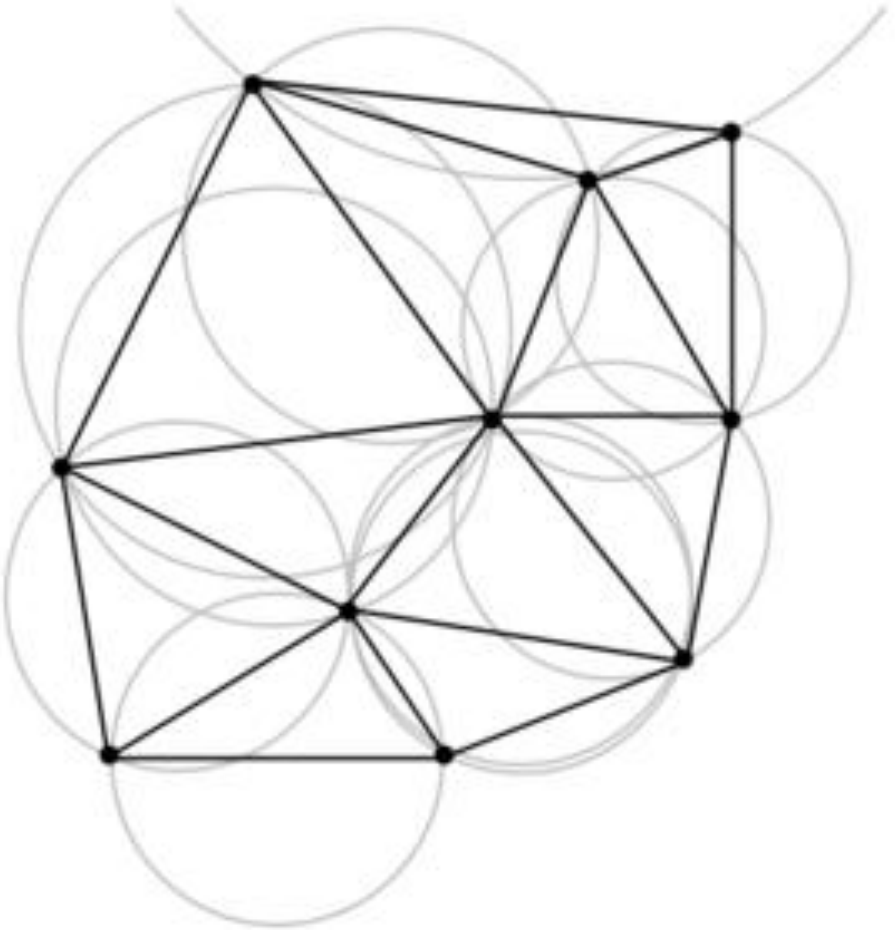
Grid Construction

- Build an unstructured grid from scattered points
- Almost all visualization software packages require data to be in a grid-based representation (structured or unstructured).
- Triangulation methods are the most-used class of methods for constructing grids from scattered points

$$\{p_i\} \text{ --> } \{p_i, C_i\}$$

Delaunay Triangulation

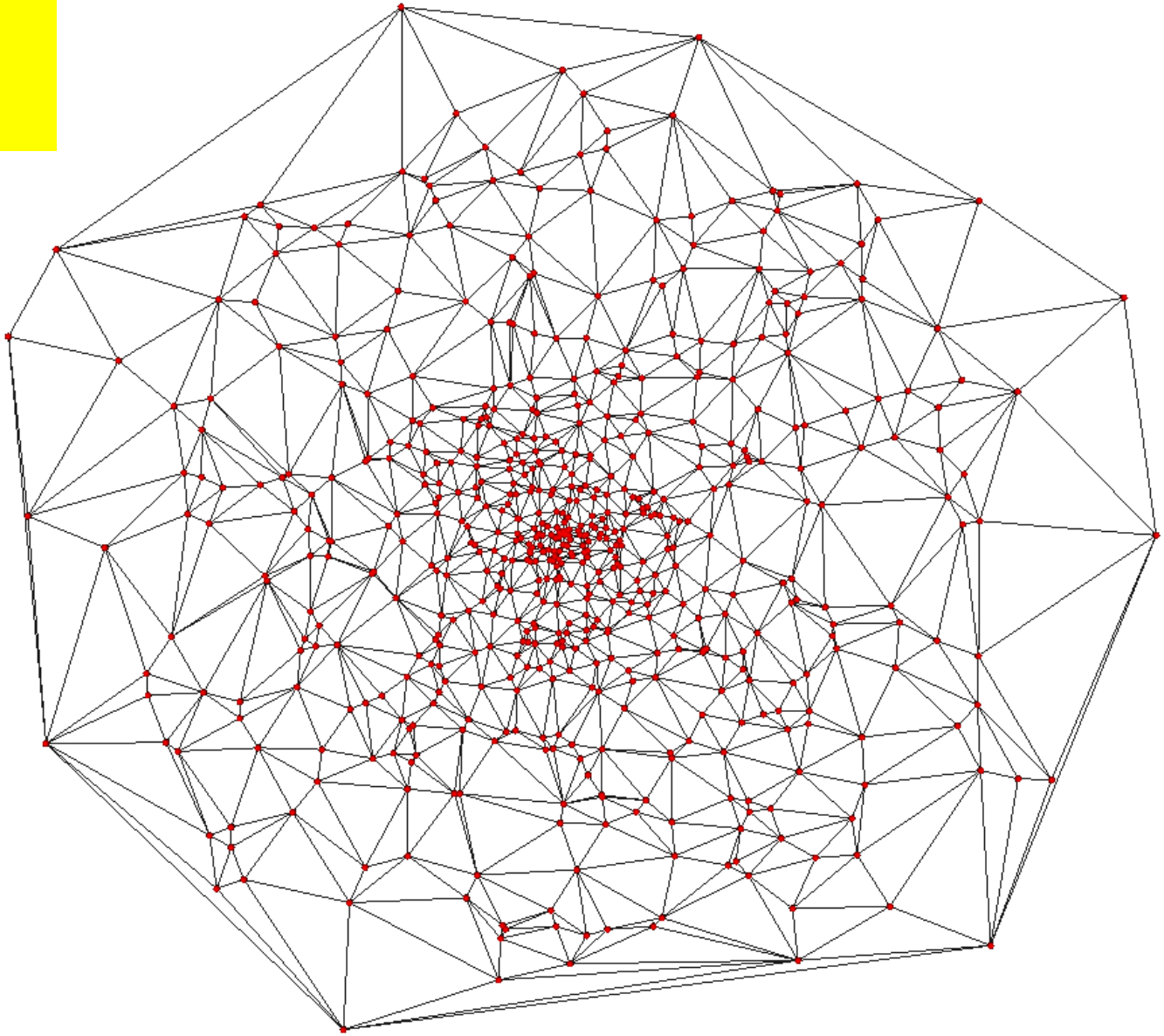
- Constructed triangles covers the convex hull of the point set
- No point lies inside the circumscribed circle of any constructed triangles



Most popular: 2-D → triangle
3-D → Tetrahedran

Delaunay Triangulation

600 random 2-D points

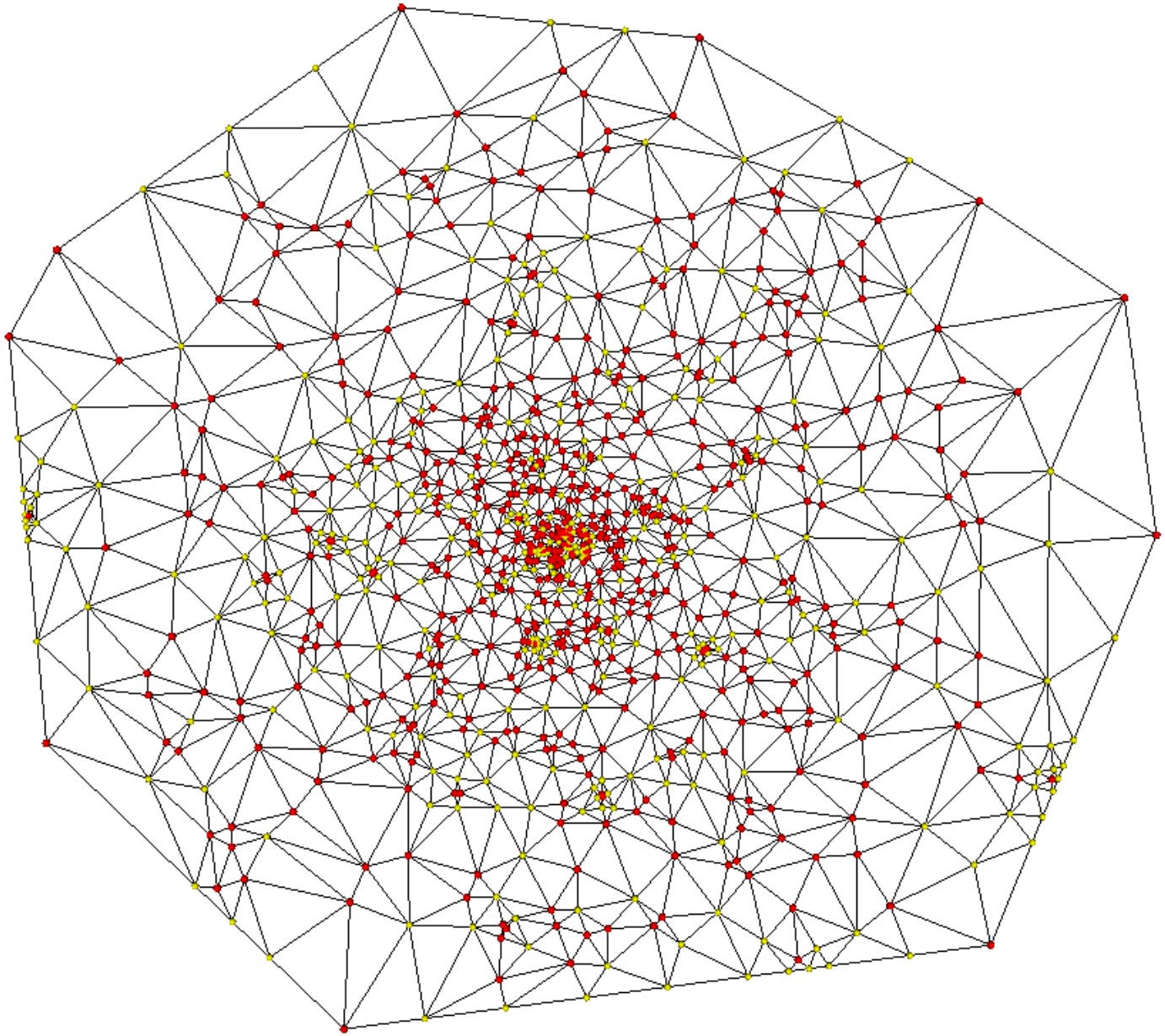


Delaunay Triangulation

Angle-constrained Delaunay Triangulation.

20° < α < 140°

Added 361 extra points

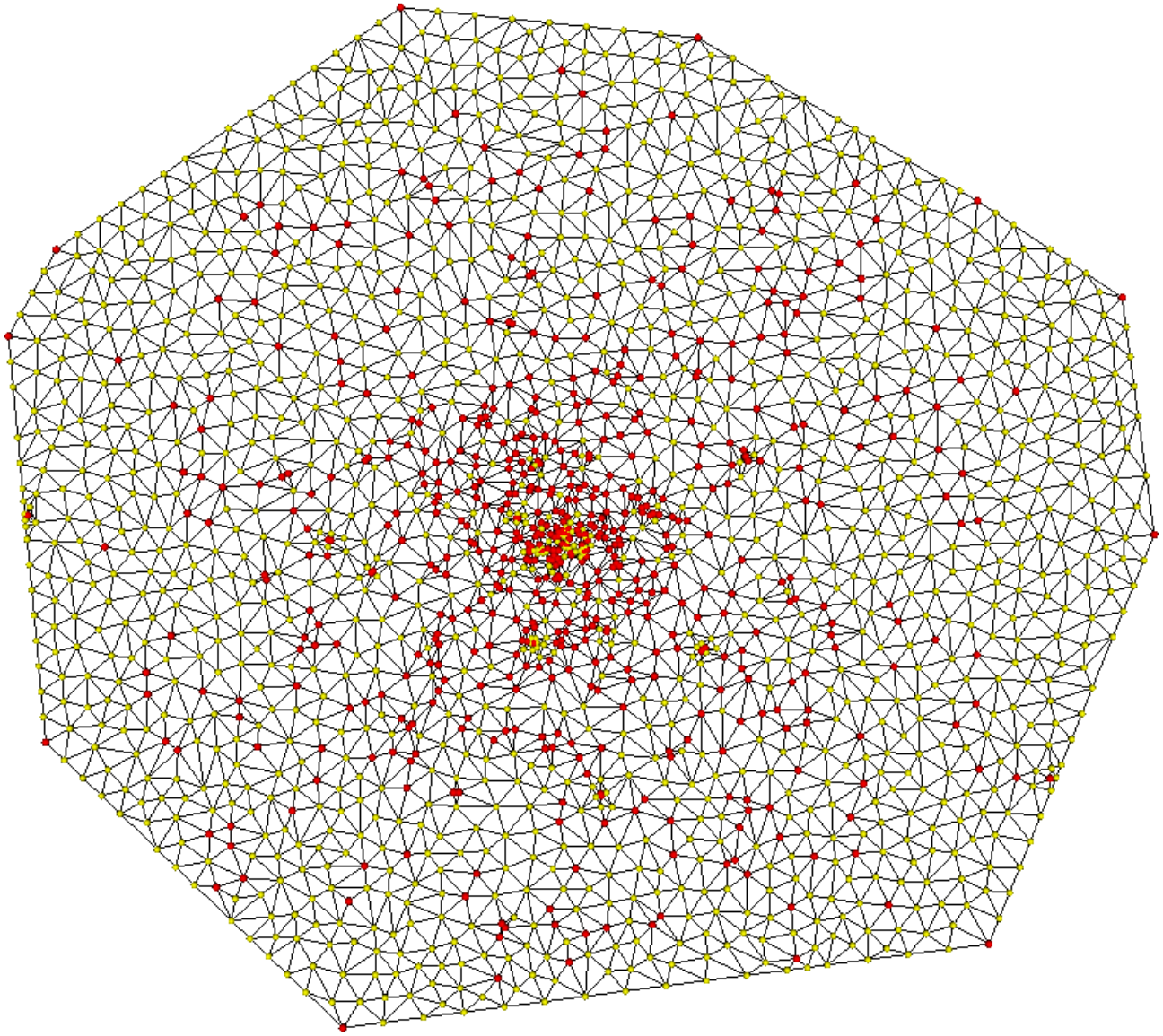


Delaunay Triangulation

Area-constrained Delaunay Triangulation.

$a < a_{max}$

Added 1272 extra points

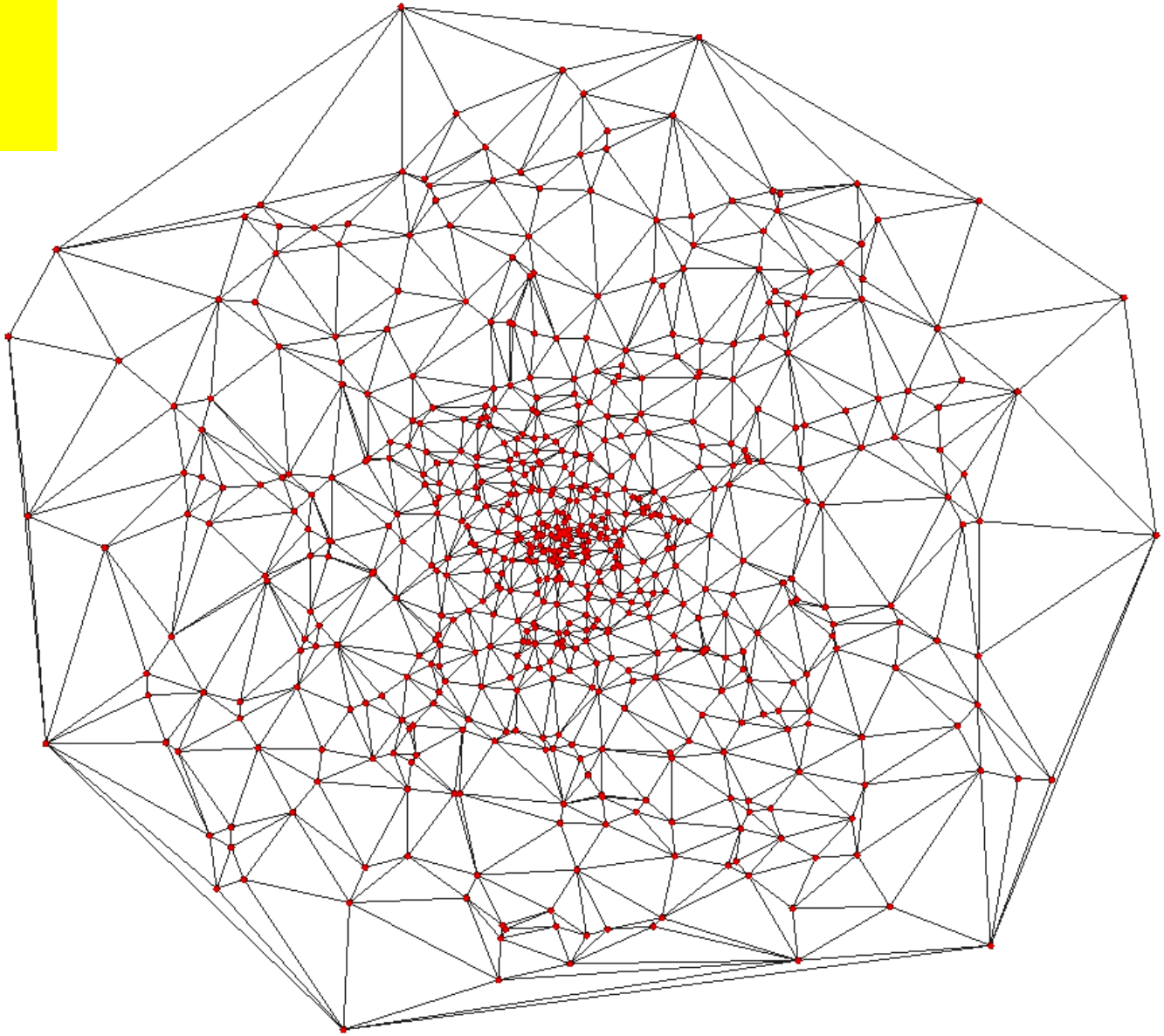


CH8.3 Grid Construction from Scattered Points (Continued)

April 11, 2013

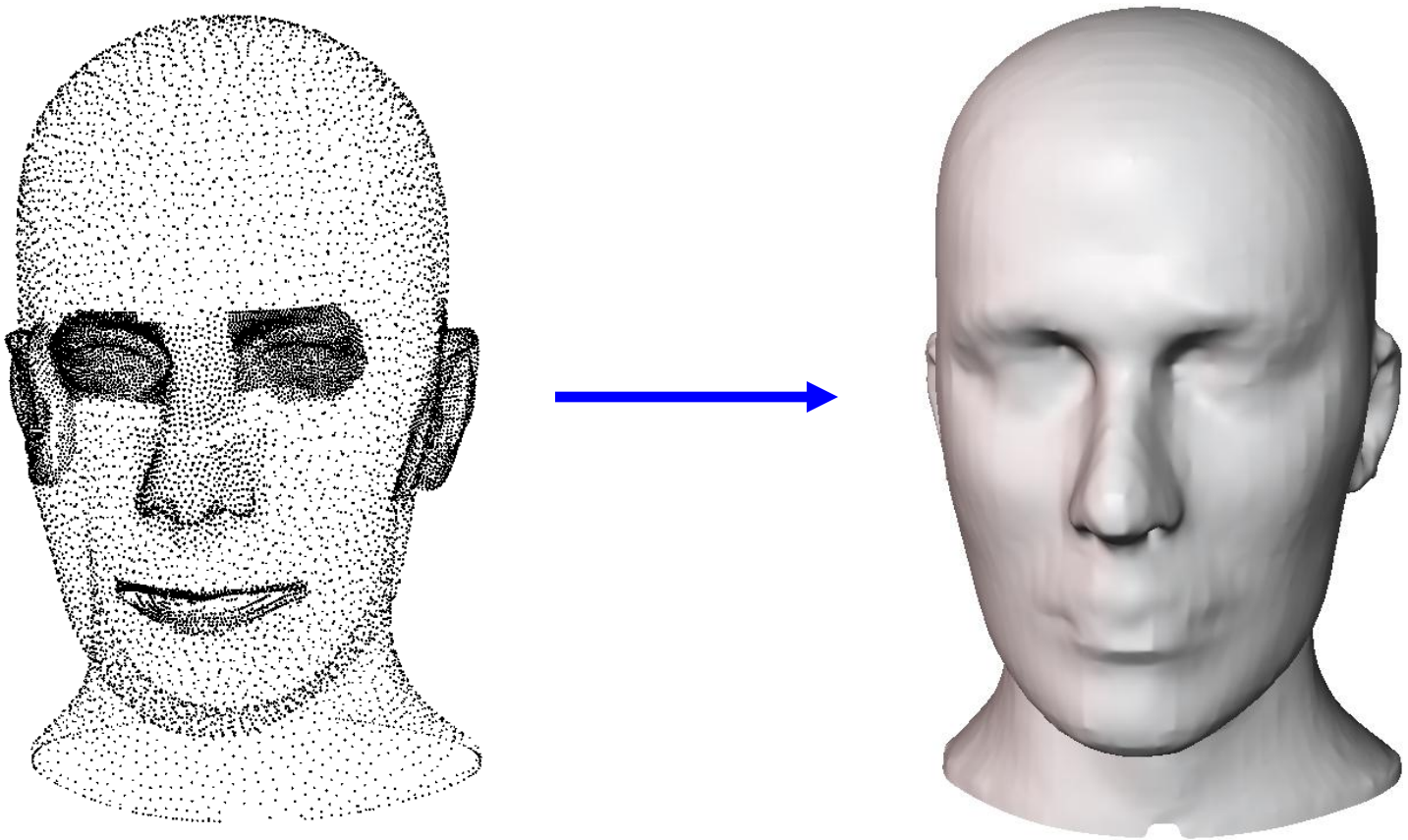
Review: Delaunay Triangulation

600 random 2-D points



Surface Reconstruction

- Render a 3-D surface from a point cloud



Radial Basis Function method

1. Computing a 3D volumetric dataset, using the 3D RBF (radial basic function) for construction.
2. **Find the isosurface (f=1)** using marching cube algorithm

$$\tilde{f}(x) = \sum \Phi(T_i^{-1}(x)), \forall x \in \mathbf{R}^3$$

$$\Phi = \begin{cases} e^{-kr^2}, & r < R \\ 0, & r \geq R \end{cases}$$

Φ: reference RBF
R: support radius
K: decay speed coefficient
T^{-1}: word-to-reference coordinate transform

Signed Distance method

1. Computing a tangent plane T_i that approximates the local surface in the neighborhood of p_i
2. Calculate the distance function f between a given point (x) and the tangent plane at the sample point closest to (x)
3. **The surface is simply the isosurface as f=0**

Find Local Tangent Plane

Tangent plane T_i is defined by its (geometric) center c_i and normal $\vec{n}_i$

$$c_i = \frac{\sum_{p \in N_i} \vec{p}}{|N_i|}$$

N_i the set of neibouring points within support radius R_i

Find Local Tangent Plane

To find the normal : Covariane Matrix

$$A = (a_{jk}) = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$= \sum_{p \in N_i} (p^j - c_i{}^j)(p^k - c_i{}^k)$$

The normal n_i is the eigenvector corresponding to the smallest eigenvalue of the matrix

The Distance function

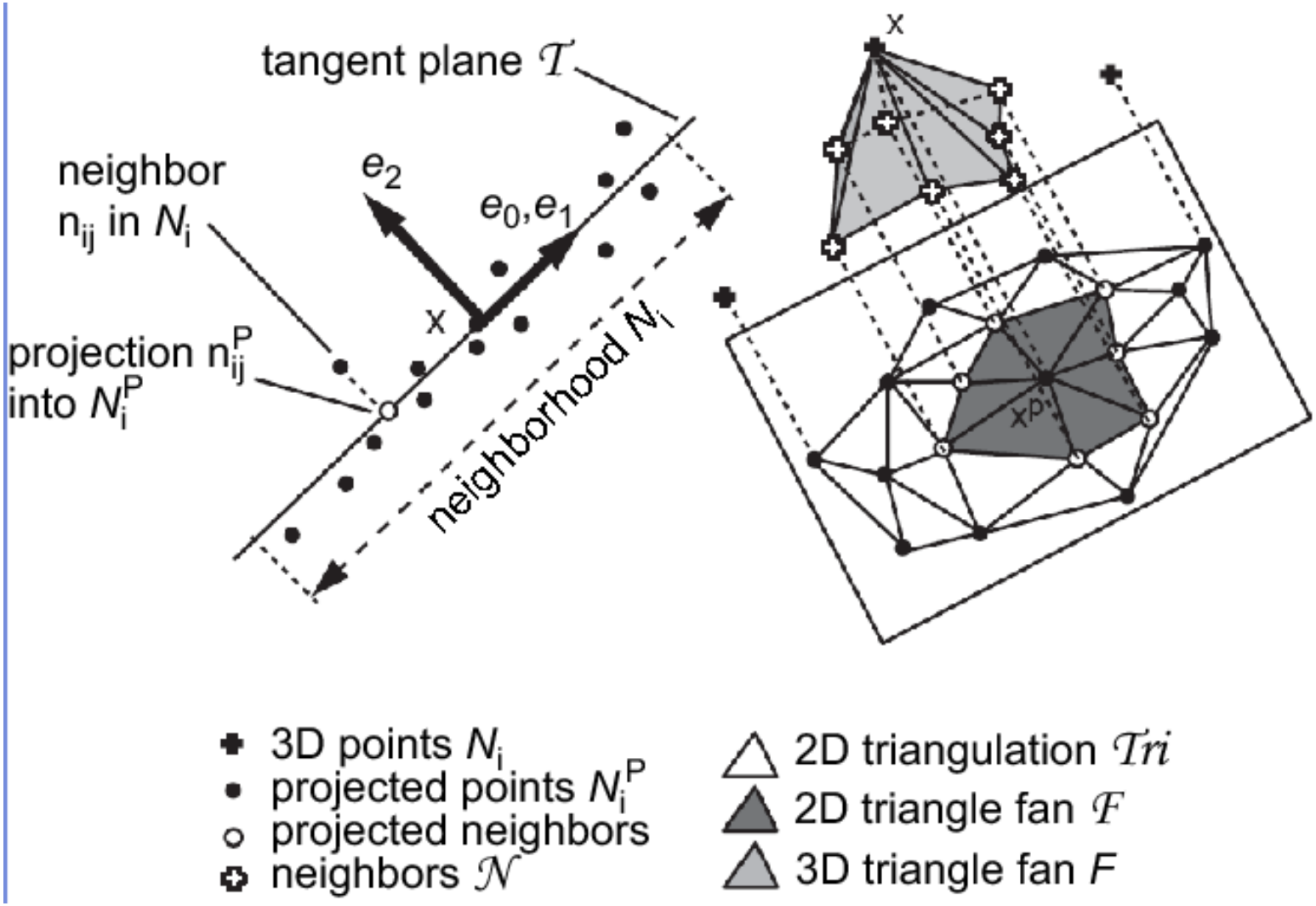
$$f(\vec{x}) = (\vec{x} - \vec{c}_i).\vec{n}_i$$

Local Triangulation Method

- Constructing the unstructured triangle mesh from local 2D Delaunay triangulation

1. Computing a tangent plane T_i that approximates the local surface in the neighborhood of p_i
2. Project its neighbor set N_i on T_i, and compute 2D Delaunay triangulation
3. Add to the mesh those triangles that have p_i as a vertex

Local Triangulation Method



CH8.4 Grid-Processing Techniques

April 11, 2013

Grid-Processing Techniques

- Grid-processing techniques change
  1. The grid geometry: location of grid sample points
  2. The grid topology: the grid cells

Geometric Transformation (in General)

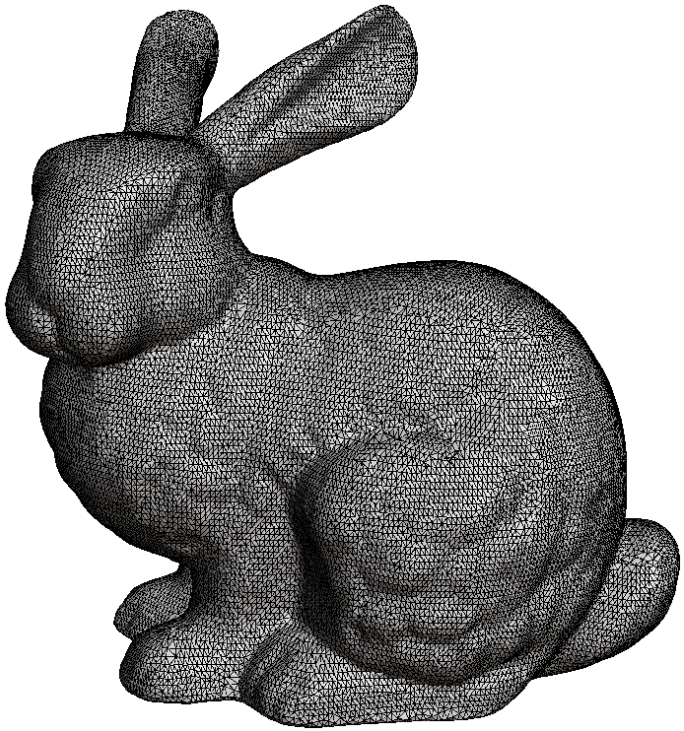
- change the position of the sample points; not modify the cells, attributes, and basis functions
- Affine operation: carry straight lines into straight lines and parallel lines into parallel lines.
  - Translation; rotation; scaling
- Nonaffine operation:
  - Bending, twisting, and tapering
- Based on attributes
  - Warping, height plot, and displacement plot
- Based on grid
  - Grid-smoothing technique

Grid Simplification

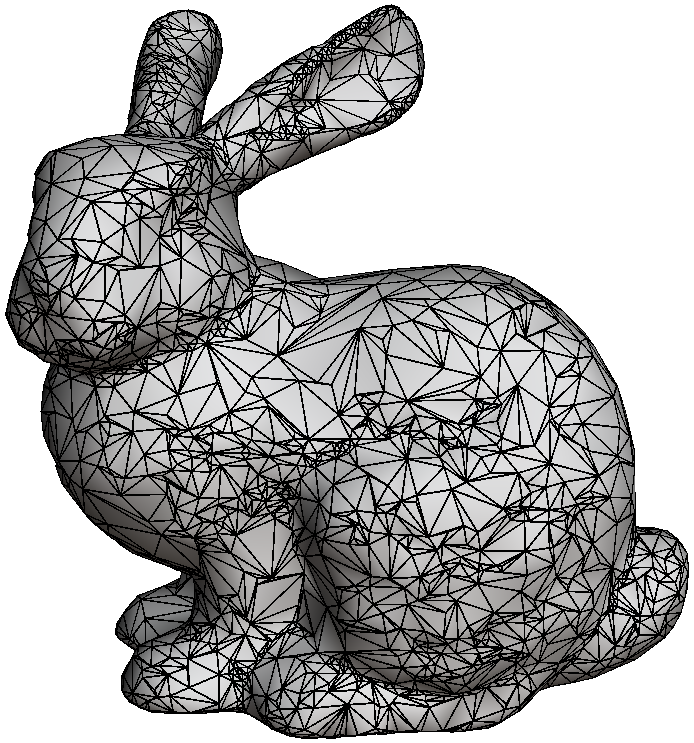
- Reducing the number of sampling points (but need to maintain the reconstruction quality)
- Uniform sampling yields too many grid points
- Adaptive sampling
  - fewer points in area of low surface curvature
  - More points in area of high surface curvature

Triangle Mesh Decimation

- Recursively reduce the vertex and its triangle fan if the resulting surface lies within a user-specified distance from the un-simplified surface



36000 grid points



3600 grid points

Vertex Clustering

- By clustering or collapsing vertices
- Assign each vertex an important value, based on the curvature (high curvature → more important)
- Overlaid a grid onto the mesh
- All vertices within the grid cell are collapsed to the most important vertex within the cell

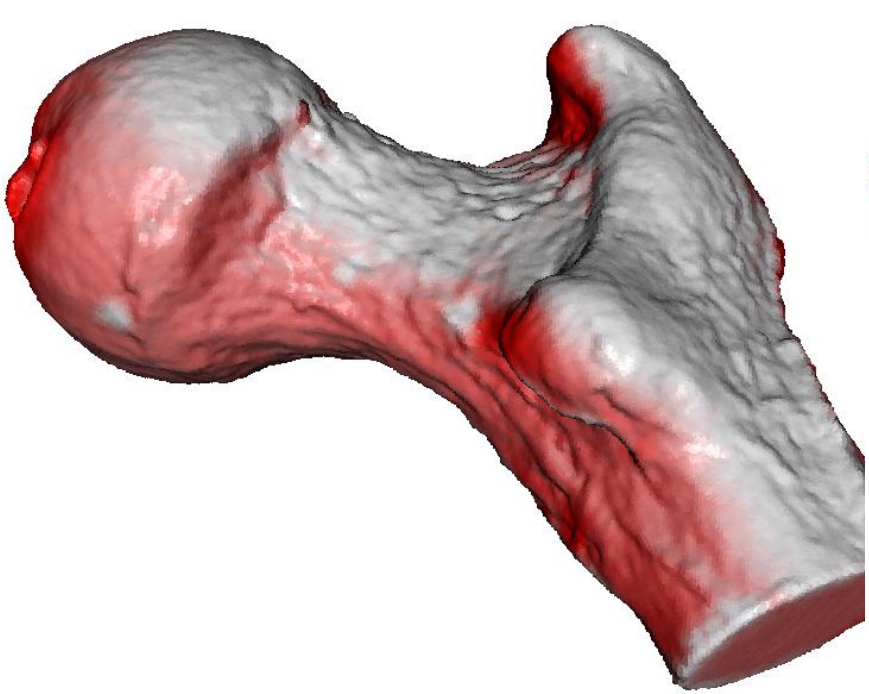
Grid Refinement

- An opposite operation of grid simplification
- Generate more sample points
- A refined grid gives better results when applying grid manipulation operation
- Inserting more points when the surface varies more rapidly

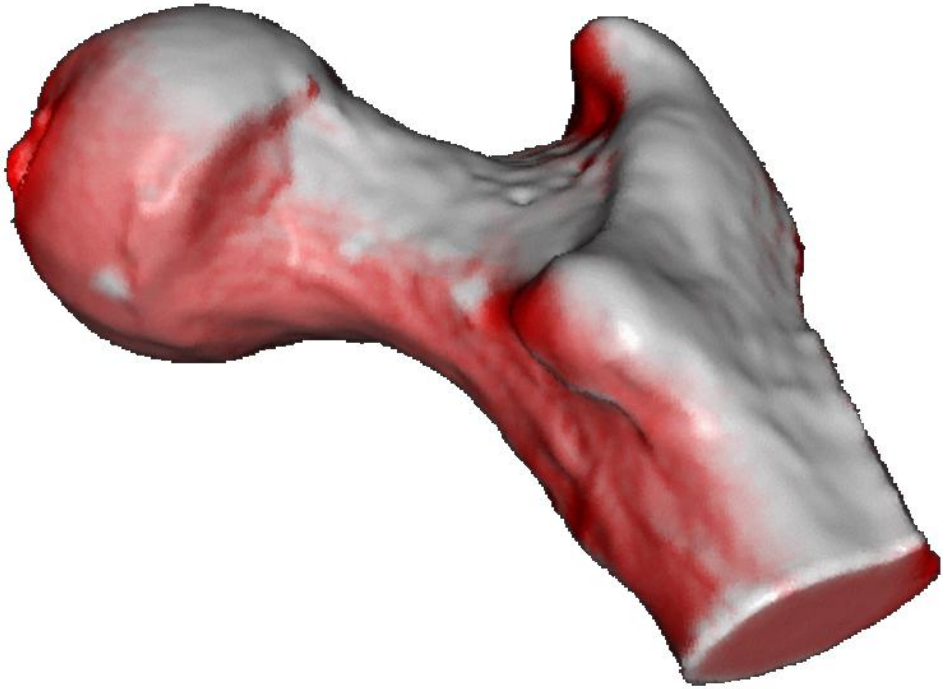
Grid Smoothing

- Question: how to reduce geometric noise?
- Modify the positions of the grid points such that the reconstructed surface becomes smoother
- Smoothing removes the high frequency, small scale variation, e.g, small spikes

Laplacian Smoothing



Before smoothing

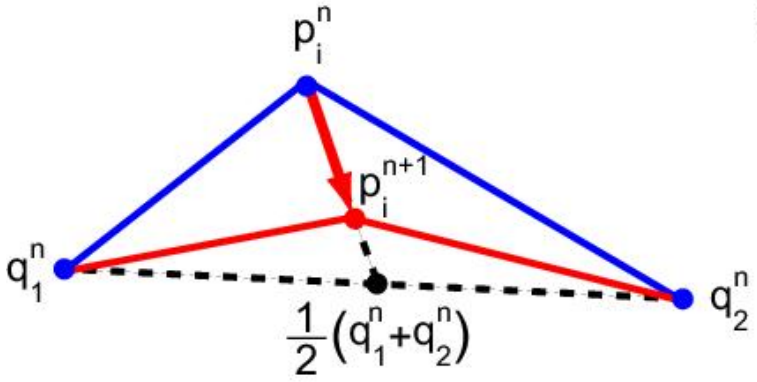


After smoothing

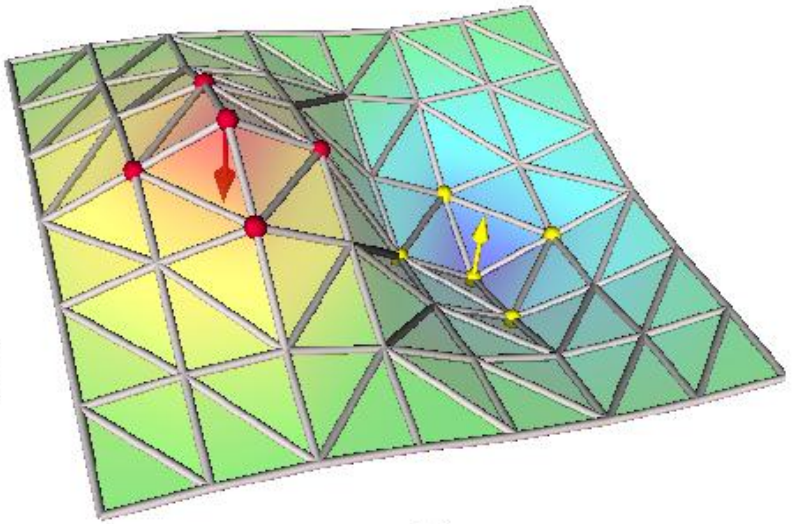
Laplacian Smoothing

Laplacian process shifts grid points toward the barycenter of its neighboring point set

$$\mathrm{b} = \frac{1}{N}\sum_{j=1}^{N}(q_j{}^n)$$



a)



b)

Laplacian Smoothing

Diffusion Equation :

$$\frac{\partial u}{\partial t} = k\Delta u$$

Δ : Laplacian operator

$$\Delta u = \nabla \cdot (\nabla u) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

For discrete grid, solving diffusion iteratively

$$p_i^{n+1} = p_i^n + k\sum_{j=1}^{N}(q_j^n - p_i^n)$$

MATLAB: Triangulation

```
%the raw data
clear;clf
load seamount  %Matlab dataset of a seamount


figure(1)
plot3(x,y,z,'.','markersize',12)
xlabel('Longitude'), ylabel('Latitude'),zlabel('Depth')


grid on
```

MATLAB: Triangulation (cont.)

```
%2-D Triangulation
figure(2)
plot(x,y,'.','markersize',12)
xlabel('Longitude'), ylabel('Latitude')
tri=delaunay(x,y);  %create a 2-D triangulation
hold on
triplot(tri,x,y);     %plot the triangulation
```

MATLAB: Triangulation (cont.)

```
%3-D triangulation
figure(3)


trimesh(tri,x,y,z)
%3-D triangular surface in wireframe mode
```

MATLAB: Triangulation (cont.)

```
%3-D triangulation in shaded mode
figure(4)

hsf=trisurf(tri,x,y,z)
%3-D triangular surface in shaded mode
set(hsf,'EdgeColor','none')
```

MATLAB: Triangulation (cont.)

```
%add visual effects
light
lighting gouraud
%lighting flat
%shading flat
shading interp
axis off


hold on
plot(x,y,'.','markersize',12)
```

End
of Chap. 8