Processing 3D Surface Data

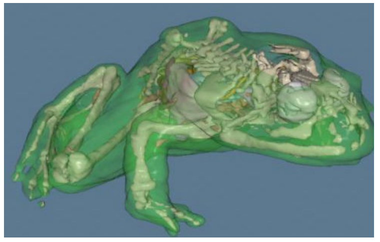
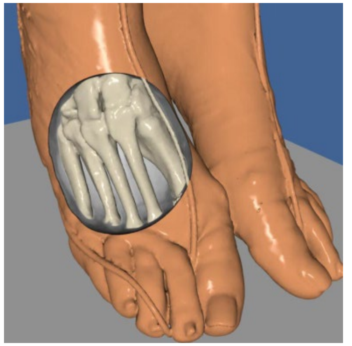
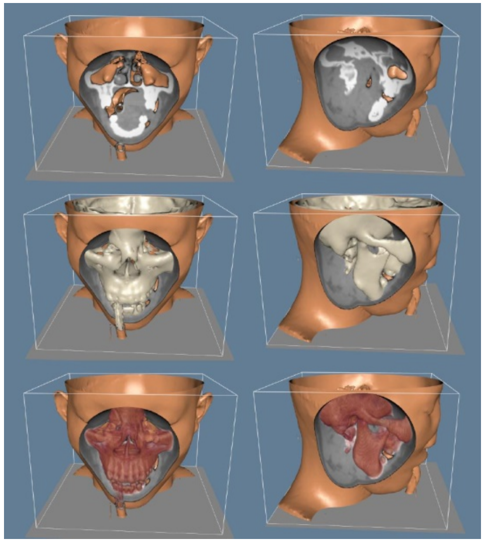
Computer Animation and Visualisation – Lecture 13

Taku Komura

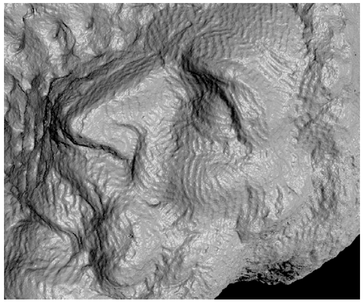
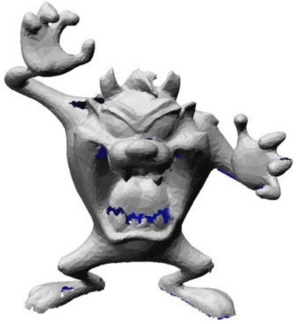
Institute for Perception, Action & Behaviour
School of Informatics

3D surface data … where from ?

- ***Iso-surfacing from scalar volumes***
  - Marching Cubes

- ***3D environment & object captures***
  - **stereo** vision & **laser range scanners**

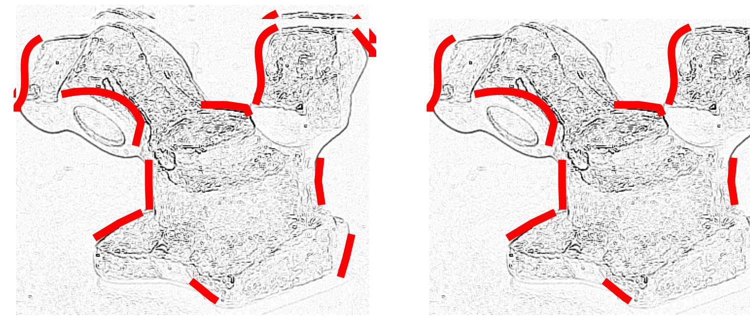
- ***Today : Algorithms for processing 3D meshes***

Processing 3D data

1) Capture the data (by stereo vision, range scanner)
2) Registration (if the data was captured by multiple attempts)
3) Adding the topology : converting to mesh data
4) Smoothing
5) Decimation
6) Remeshing

Stereo Vision : 3 key stages

1) Feature Identification

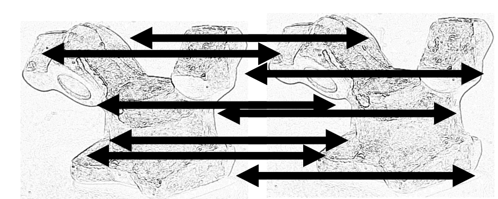


identify image features in each image

2) Correspondence

Matching: find a set of consistent feature correspondences

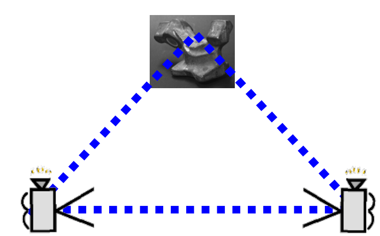
(left image ⇔ right image)



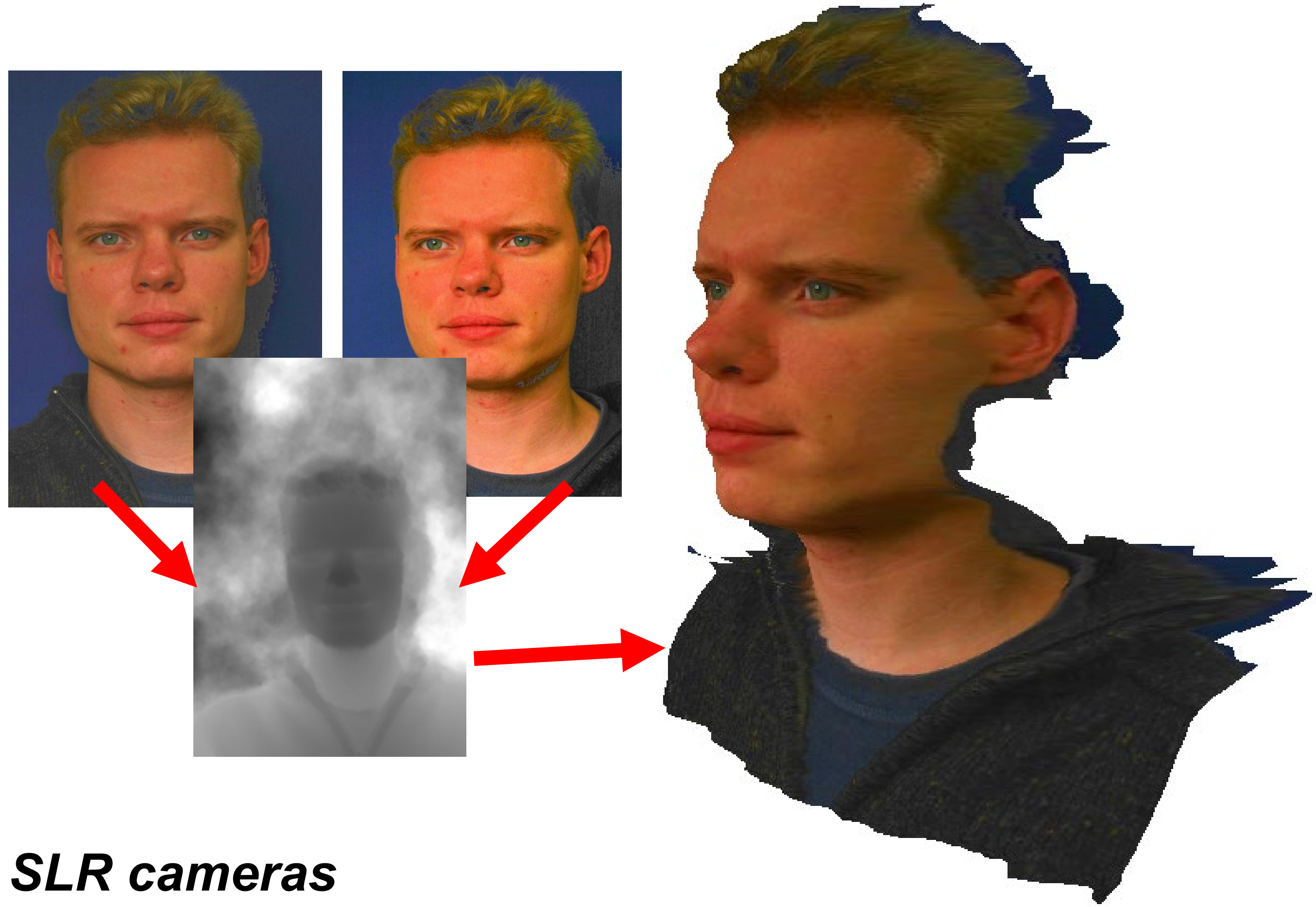
3) Triangulation

triangulate from known camera positions

recover 3D depth information



Dense Stereo Matching : Face



- ***2 x 6 mega-pixel digital SLR cameras***
- ***Commercial 3D stereo software (http://www.di3d.com/)***
  - *Results : 6 mega-pixel depth map / VRML 3D surface mesh model*

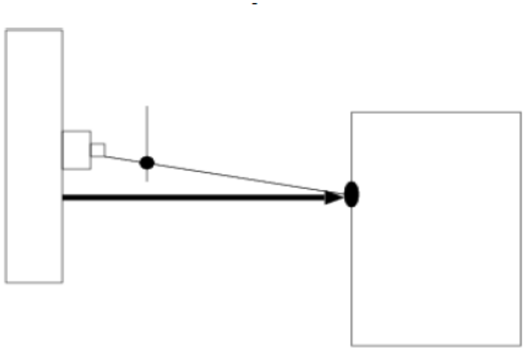
Dense Stereo Matching

- ***Advantages***
  - **passive** technique
  - uses only **image cameras** : no expensive sensors
- ***Limitations***
  - **accurate prior calibration** of cameras required
  - fails on textureless surfaces
  - **noise**: effects matching, produces errors
    - —lighting effects images (e.g. specular highlights)
  - results : often sparse incomplete surfaces

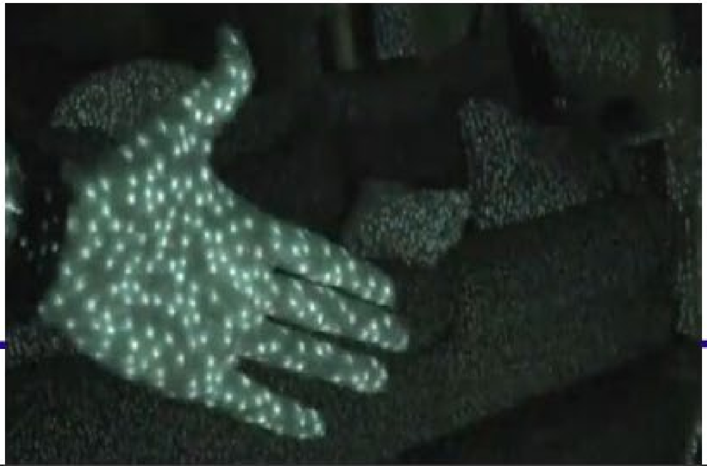
Another solution → Active techniques

Microsoft KINECT (old version)

- Estimating the depth from projected structured infra-red light
- Practical ranging limit of 1.2–3.5 m (3.9–11 ft)
- Software fits a skeleton character into the point cloud data

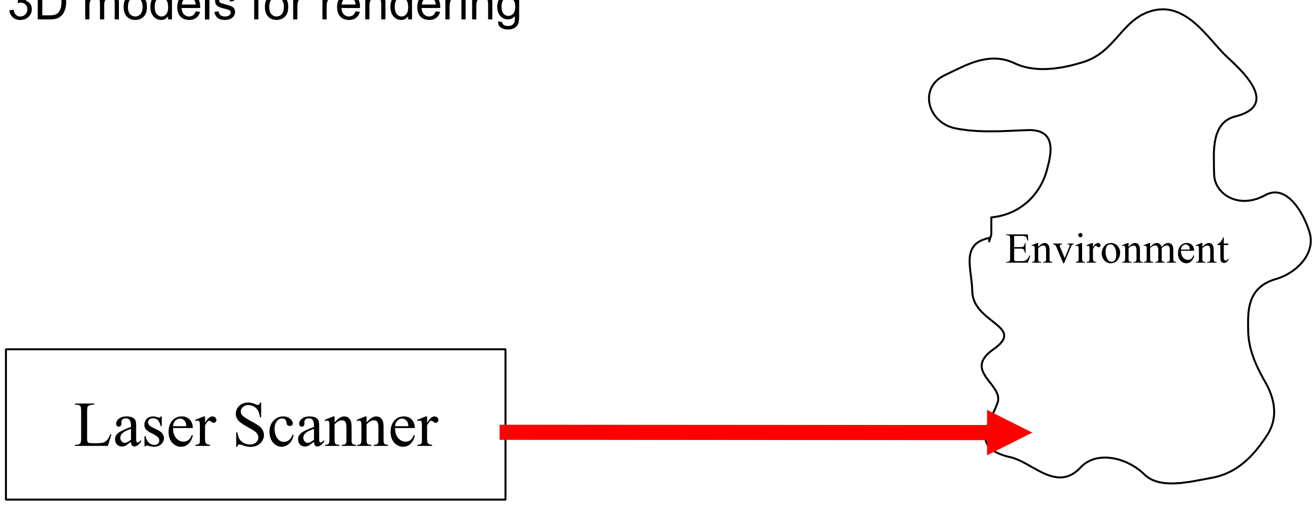


https://www.youtube.com/watch?v=dTKlNGSH9Po



3D capture : laser range scanning

- ***Active depth sensing using laser beam signal***
  - **direct, accurate 3D** scene information
  - unambiguous measurement (unlike stereo)
  - **Limitations**
    - —hidden surfaces (2½D)
    - —dark/shiny objects do not scan well
    - —dense 3D models for rendering



Great Buddha Project in Japan

- ***Capturing took 3 weeks x 2 trips x 10 students/staff***

http://www.youtube.com/watch?v=OoNr7DV0b-M&feature=channel_page




Processing 3D data

1) Capture the data (by stereo vision, range scanner)

2) **Registration (if the data was captured by multiple attempts)**

3) Adding the topology : converting to mesh data

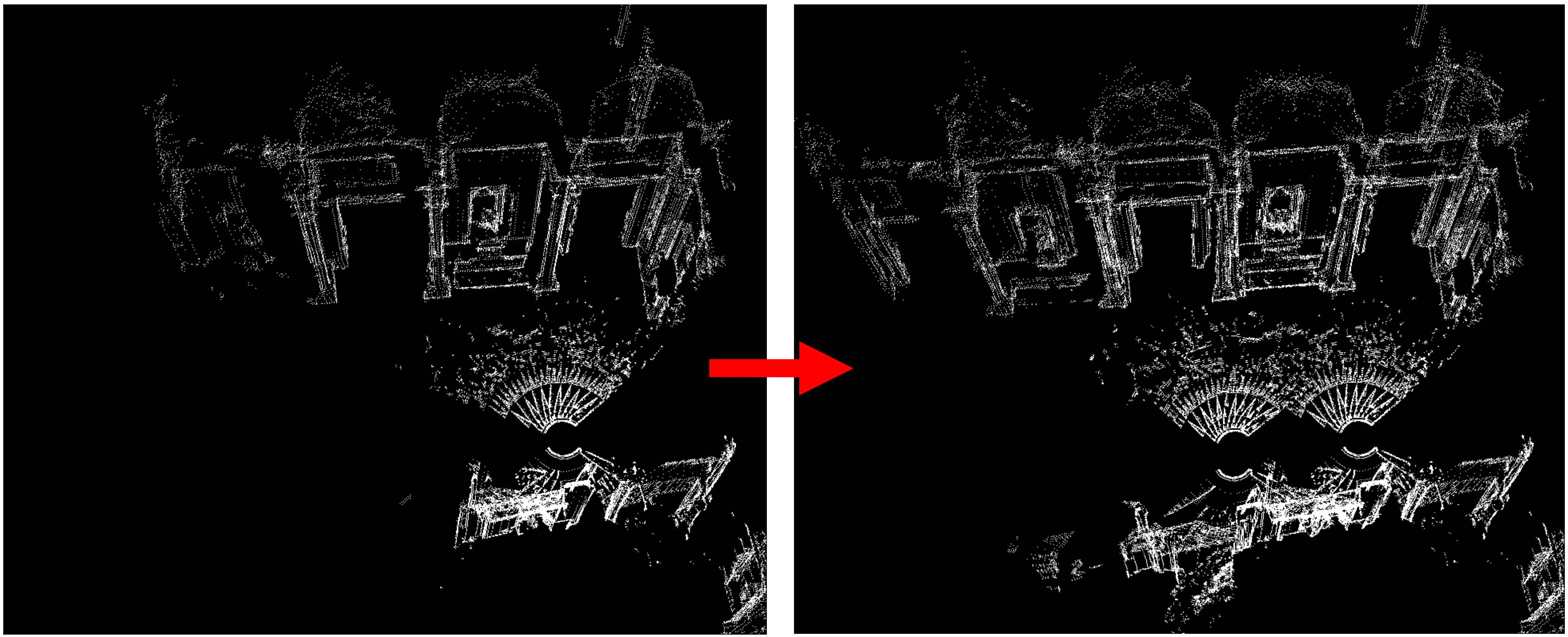
4) Smoothing

5) Decimation

6) Remeshing

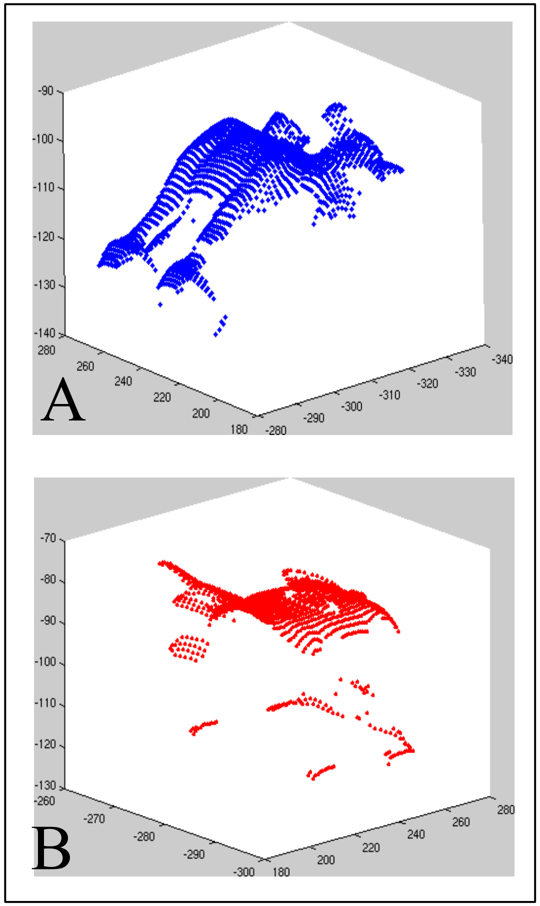
3D Registration

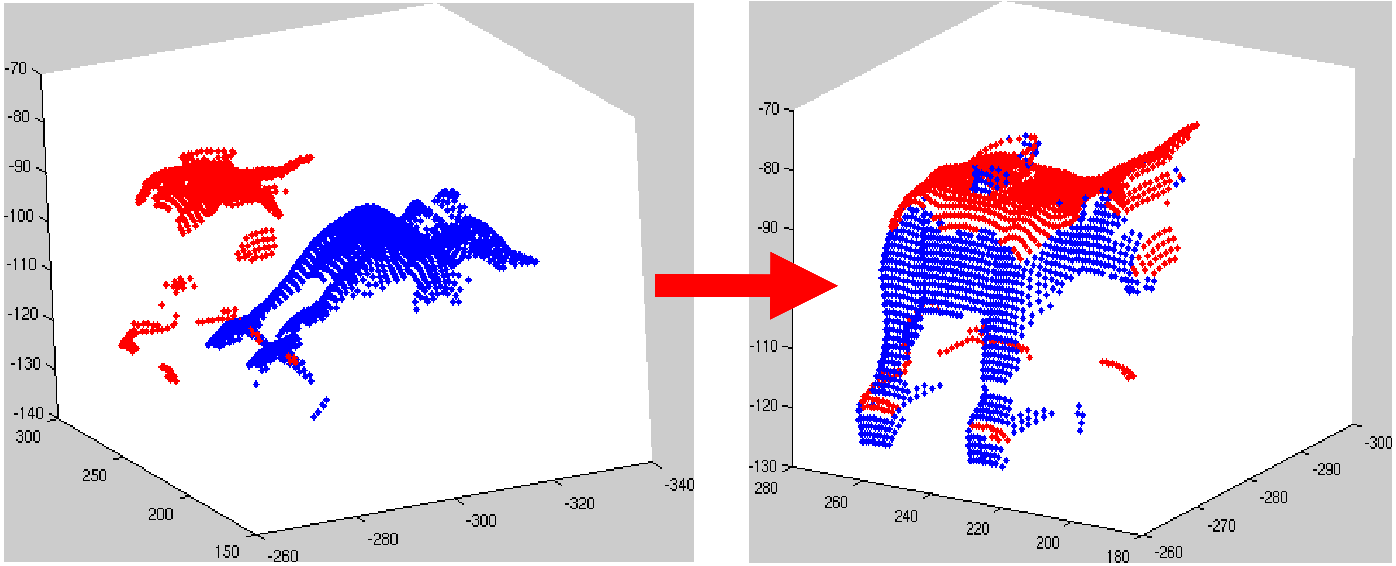
- ***Producing larger 3D models by combining multiple range scans from different viewpoints***



Registration – ICP algorithm

- ***Example : registration of 3D model parts (toy cow)***





- **Input: point clouds acquired from non-aligned viewpoints**

(from 3D range scanner) & initial estimation of registration

- **Output: Transformation**

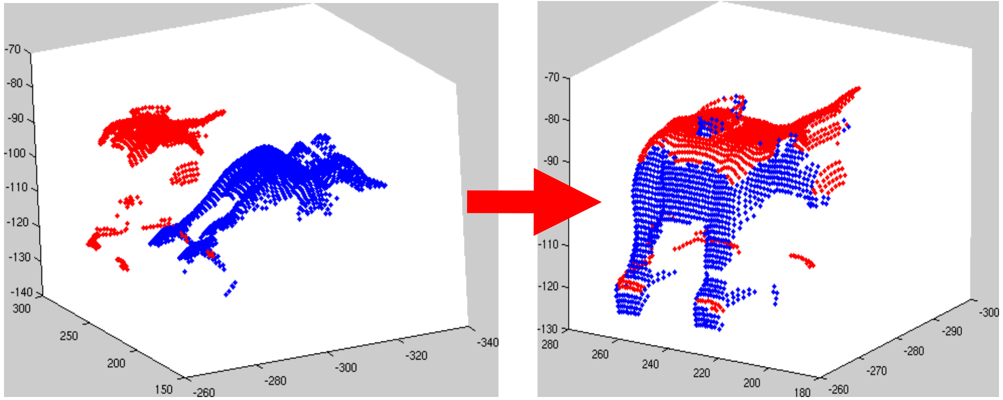
Registration – ICP algorithm

- ***Problem : merging of 3D point clouds from different, unaligned orientations***
  - common in large scale range scanning *(e.g. Mosque)*
- ***Solution : iterative estimation of rigid 3D transform between point clouds [Iterative Closest Point (ICP) Besl / Mckay '92]***
- ***ICP algorithm: for point clouds A & B***
  - ***align*** *A & B using* ***initial estimate of 3D transform***
  - ***until*** *distance between matched points < threshold*
    - *matched points =* ***N closest point pairs*** *between A & B*
    - ***estimate transformation*** *B→A between matched pairs*
    - ***apply transformation*** *to B*

ICP Criteria

Computing the transformation (rotation, translation) that minimzes the following criteria

$$E(\mathbf{R},\mathbf{t}) = \min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{q}_i - (\mathbf{R}\mathbf{p}_i + \mathbf{t})\|^2$$

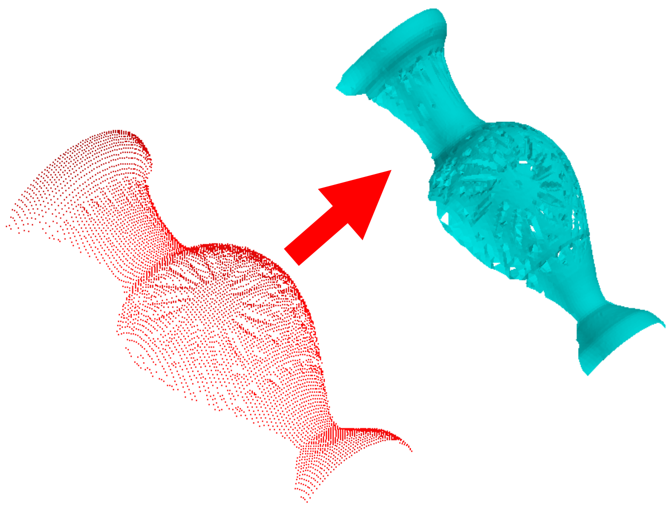


Processing 3D data

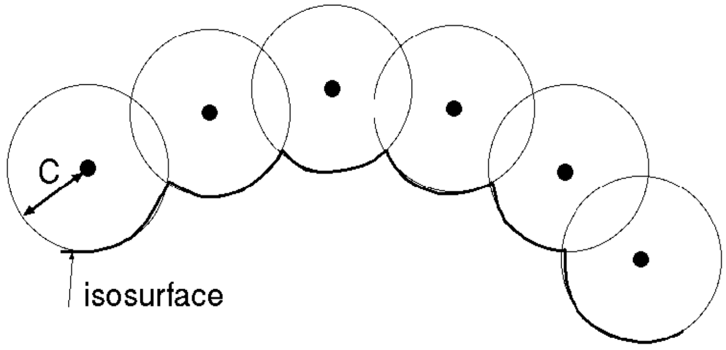
1) Capture the data (by stereo vision, range scanner)
2) Registration (if the data was captured by multiple attempts)
3) **Adding the topology : converting to mesh data**
4) Smoothing
5) Decimation
6) Remeshing

Point Clouds → Surface Meshes

- ***Input : unstructured 3D points (point cloud)***
- ***Output : polygonal data set (surface mesh)***

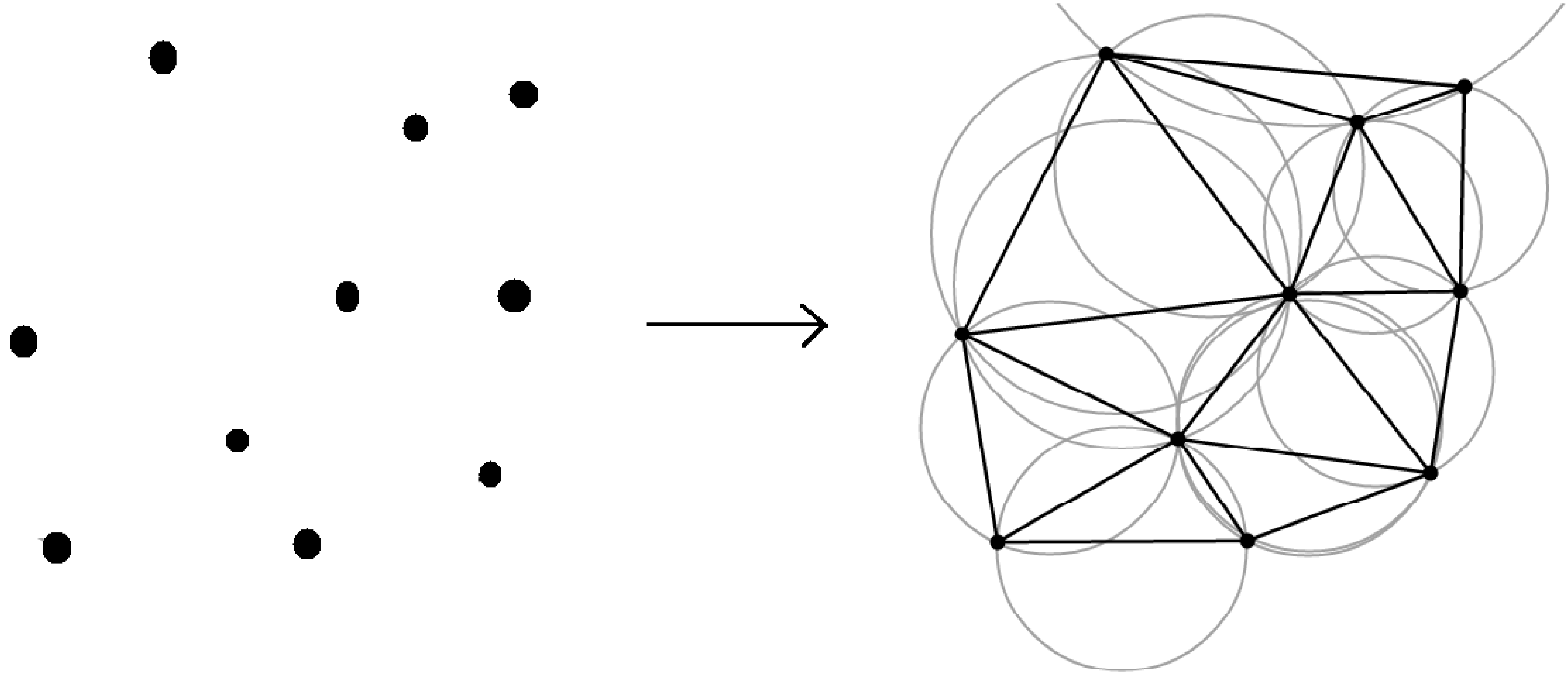


- ***Multiple techniques available***
  - **Delaunay Triangulation**
    - —For surface result: 2½D data sets only
    - —The data needs to be projected onto a plane
  - **Iso-surface** based Techniques
    - —define pseudo-implicit functional 3D space *f(x,y,z)* = *c* where *c* is distance to nearest 3D input point
    - —use iso-surface technique (Marching Cubes) to build surface

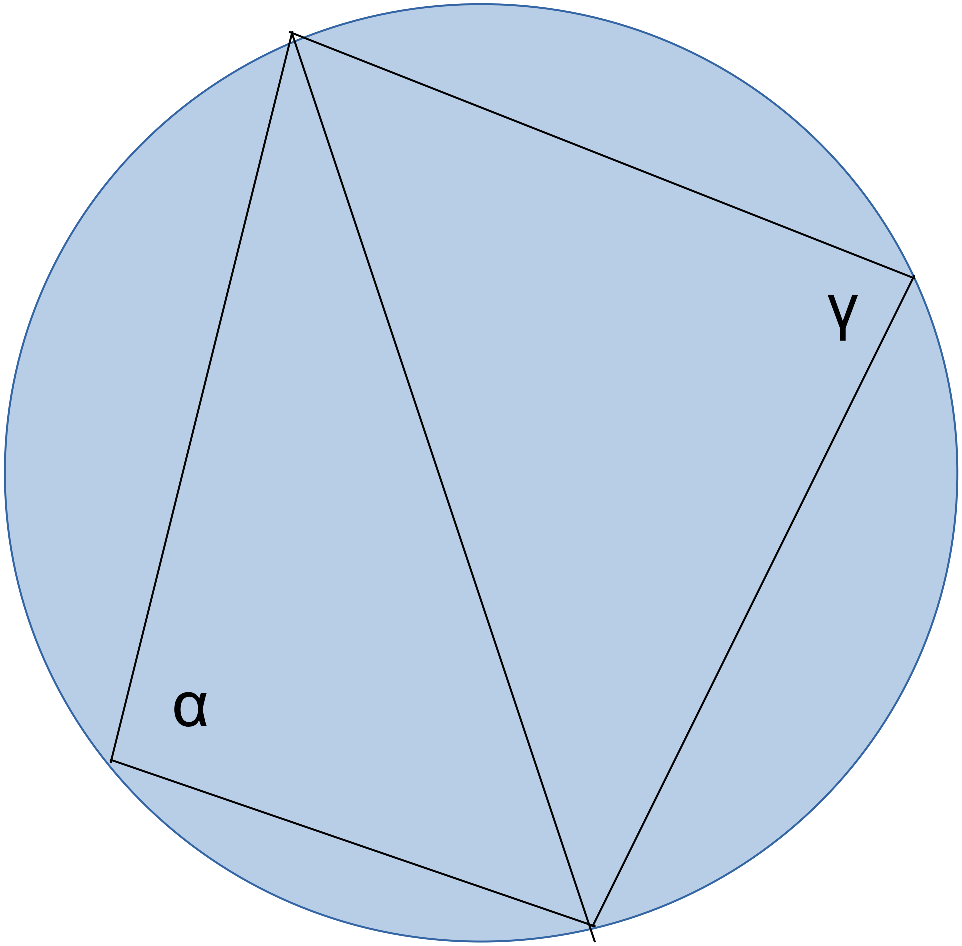


Delaunay Triangulation

Given the point cloud, compute the triangulation such that none of the other points come into the circumcircle of each triangles



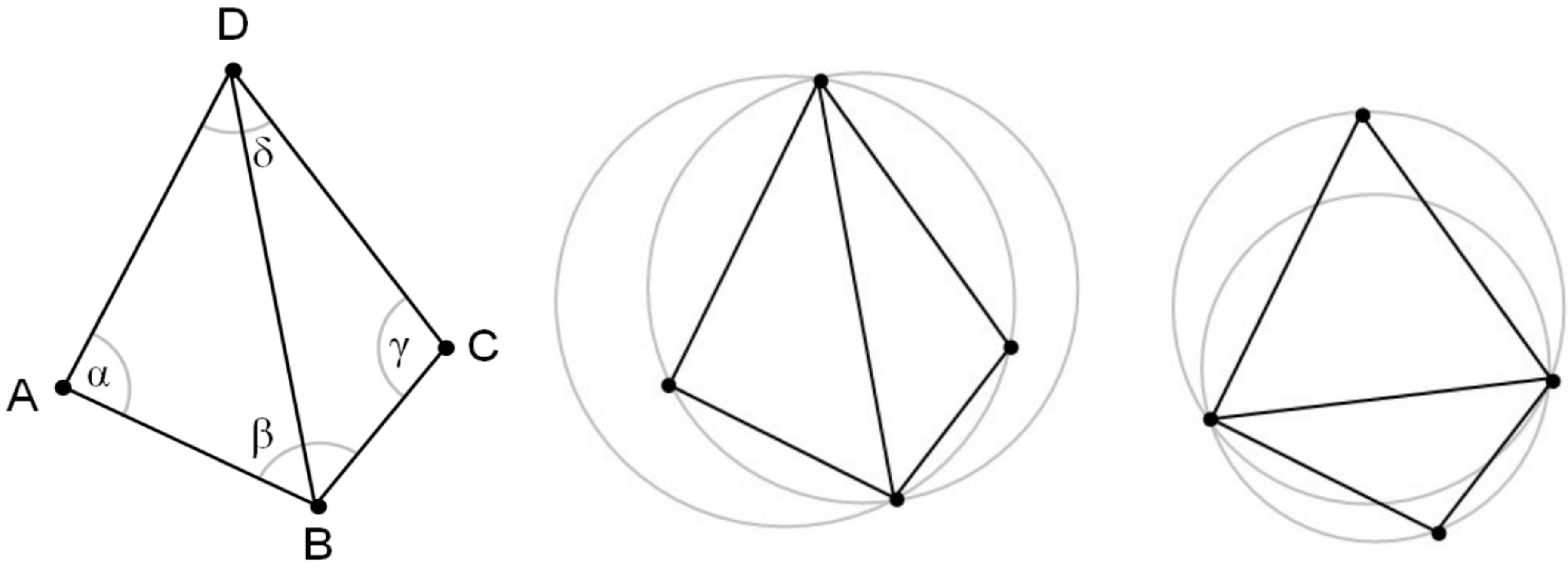
Delaunay Condition



$\alpha + \gamma <= 180$

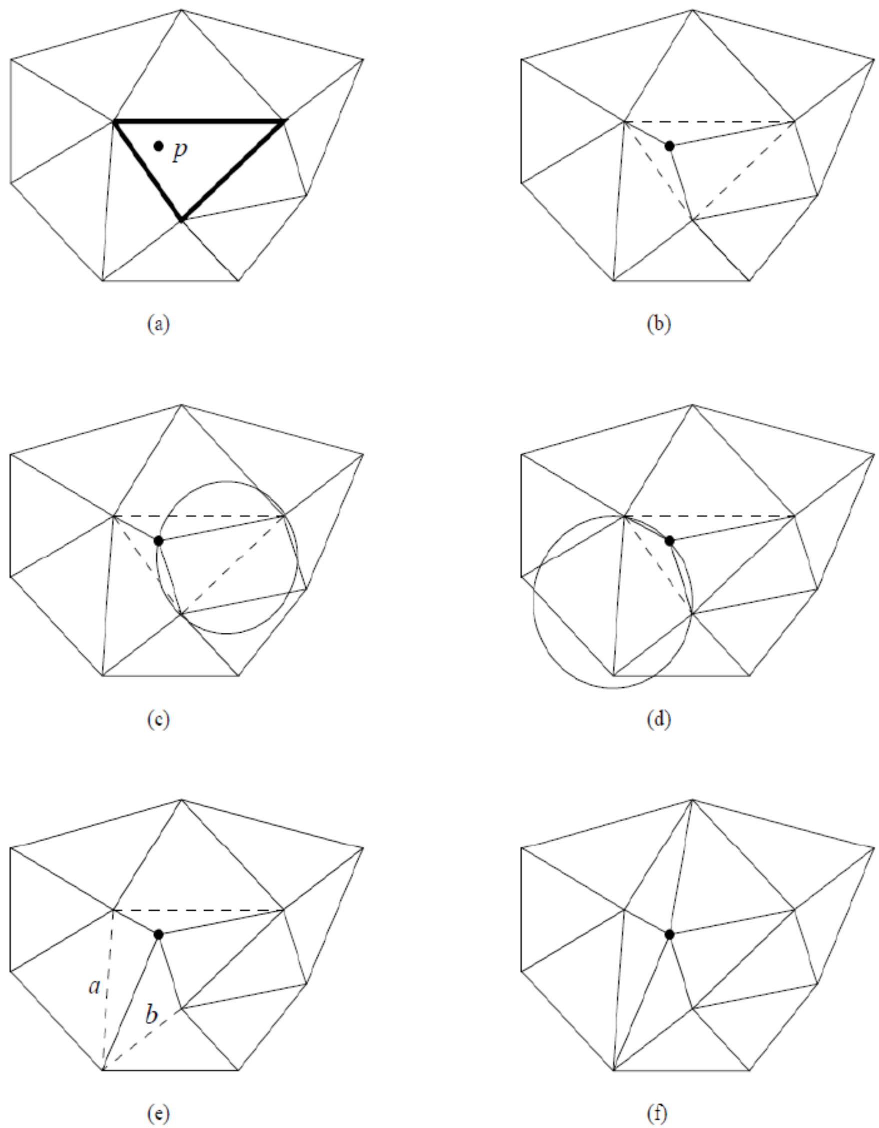
Delaunay Condition (2)

Looking at two triangles ABD and BCD with the common edge BD, if the sum of the angles α and γ is less than or equal to 180 degrees, the triangles meet the Delaunay condition.



Incremental Delaunay Triangulation

- Repeatedly add one vertex at a time (p)
- Find the triangle T that contains p
- Split T in three
- For each edge of T, apply the flip algorithm.
- Repeat



Processing 3D data

1) Capture the data (by stereo vision, range scanner)

2) Registration (if the data was captured by multiple attempts)

3) Adding the topology : converting to mesh data

4) **Smoothing**

5) Decimation

6) Remeshing

Mesh Smoothing - 1

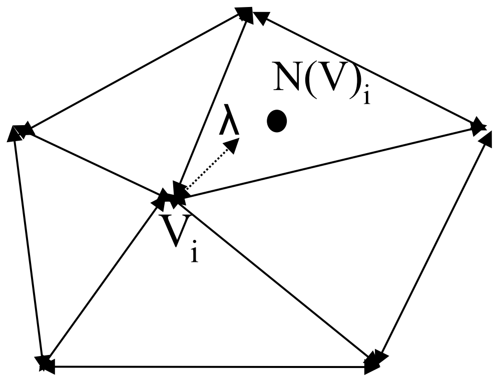
- ***Surface Noise : surfaces from stereo vision and (large scale) laser scanning often contain noise***
  - Smoothing removes noise, improves uniform surface curvature → helps c

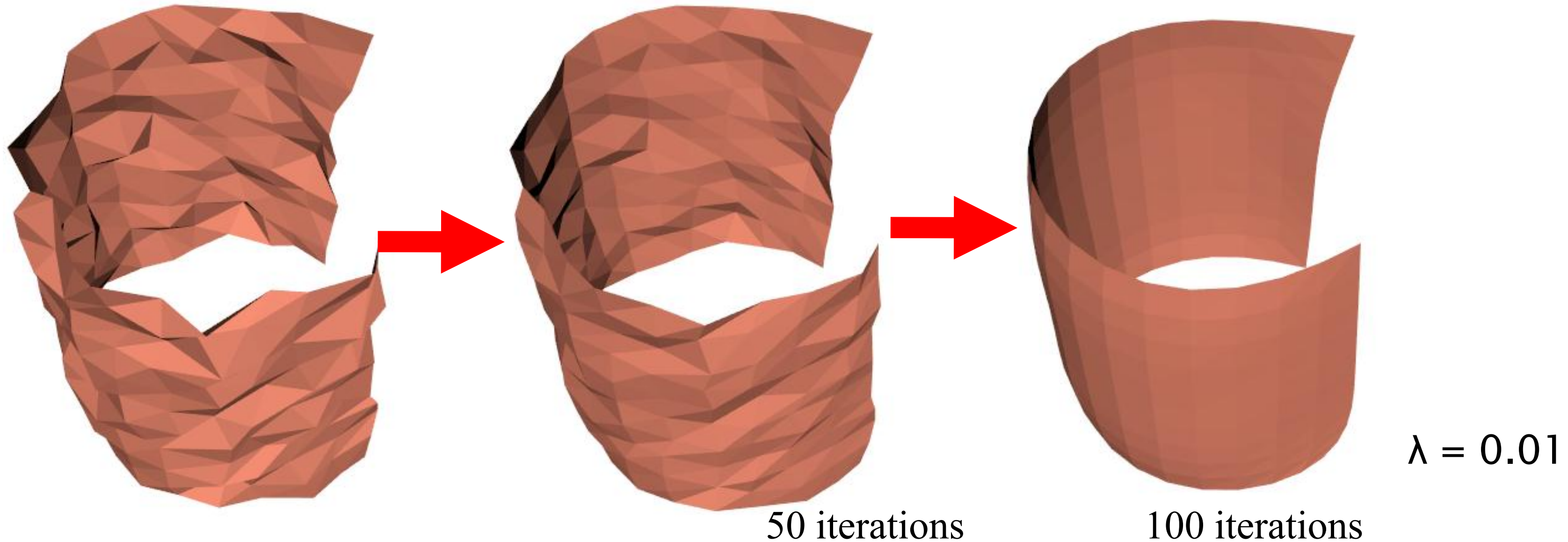
- ***Solution: Laplacian mesh smoothing***
  - **modifies geometry** of mesh
  - **topology unchanged**
  - **reduces surface curvature** and **removes noise**

Mesh Smoothing - 2

- ***Iterative smoothing approach***
  - At each iteration *i* move each vertex $V_{i,j}$ towards mean position of neighbouring vertices, $N(V_{i,j})$, by λ

$$V_{i+1,j} = V_{i,j} + \lambda N(V_{i,j})$$

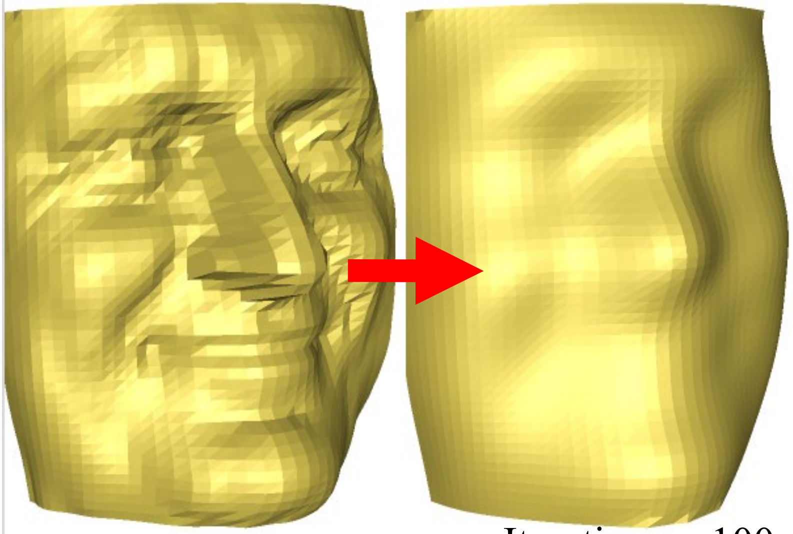




Mesh Smoothing - 3

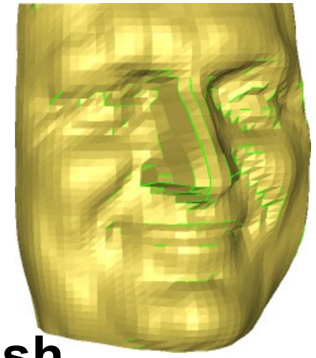
- ***Reduces high-frequency mesh information***
  - **removes noise**
- ***but also mesh detail!***

Decimate.tcl



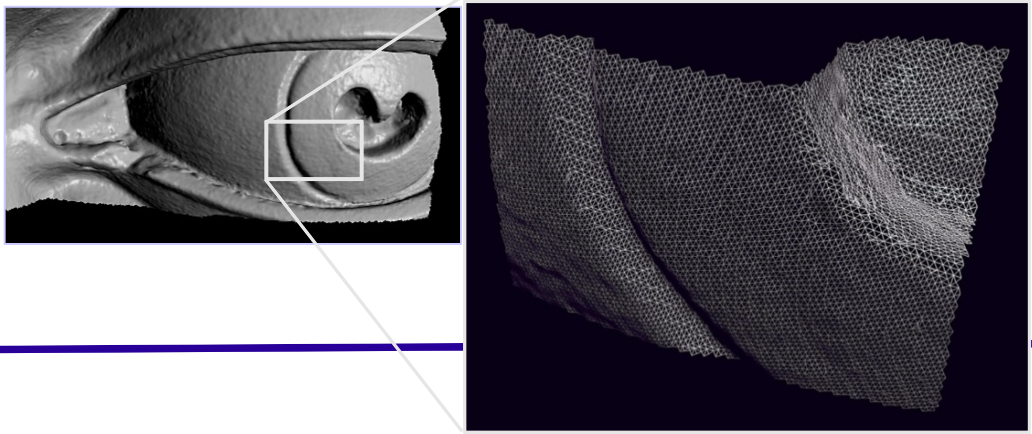
Iterations = 100; λ = 0.1

- ***Limitations : loss of detail, mesh shrinkage***
  - ***enhancements*** **: feature-preserving smoothing & non-shrinking iterative smoothing**
  - **feature points can be "anchored" to prevent movement in mesh smoothing (detail preservation)**



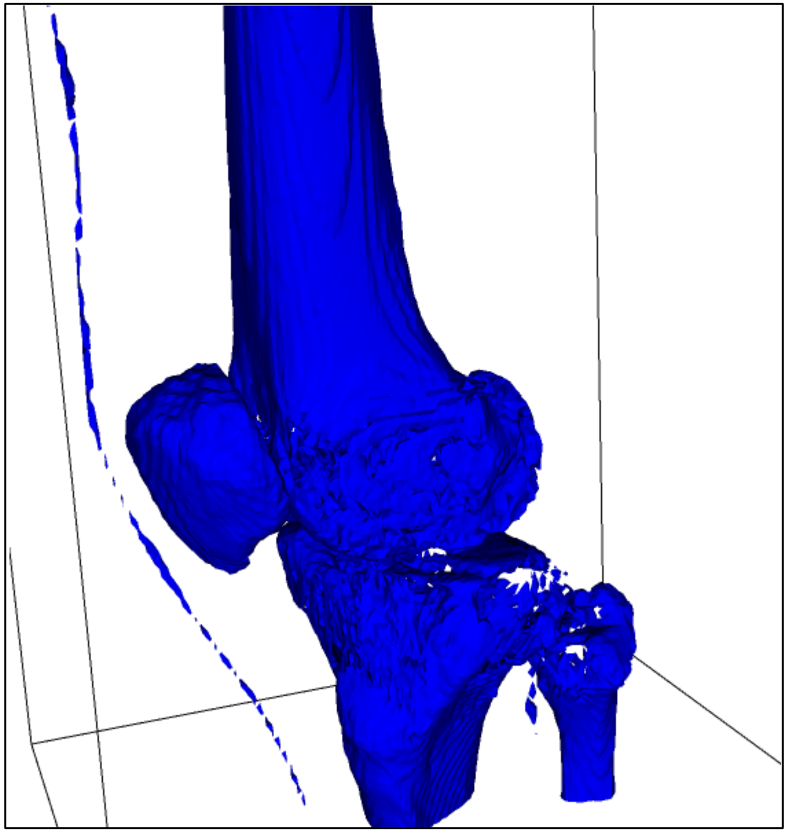
Handling Large Polygonal Datasets

- ***Large Surface Meshes***
  - **Marching Cubes** : voxel dataset of 512^3 produces a **typical iso-surface of 1-3 million triangles**
  - **Laser Scanning :** datasets in **10s millions of triangles**
- ***Problem : lots to render!***
- ***Solution : polygon reduction***
  - **sub-sampling** : *simple sub-sampling is bad!*
  - **decimation** : **intelligent sub-sampling by optimising** mesh

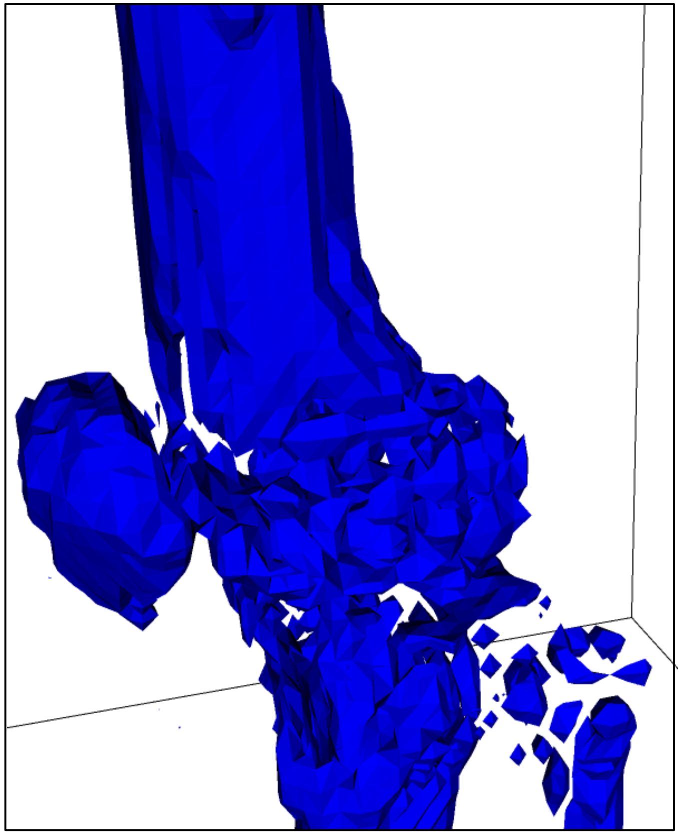


Basic Sub-sampling

- ***Basic uniform sub-sampling (take every nth sample)***
  - **loss of detail / holes / poor surface quality**
  - Marching Cubes surface example
  :

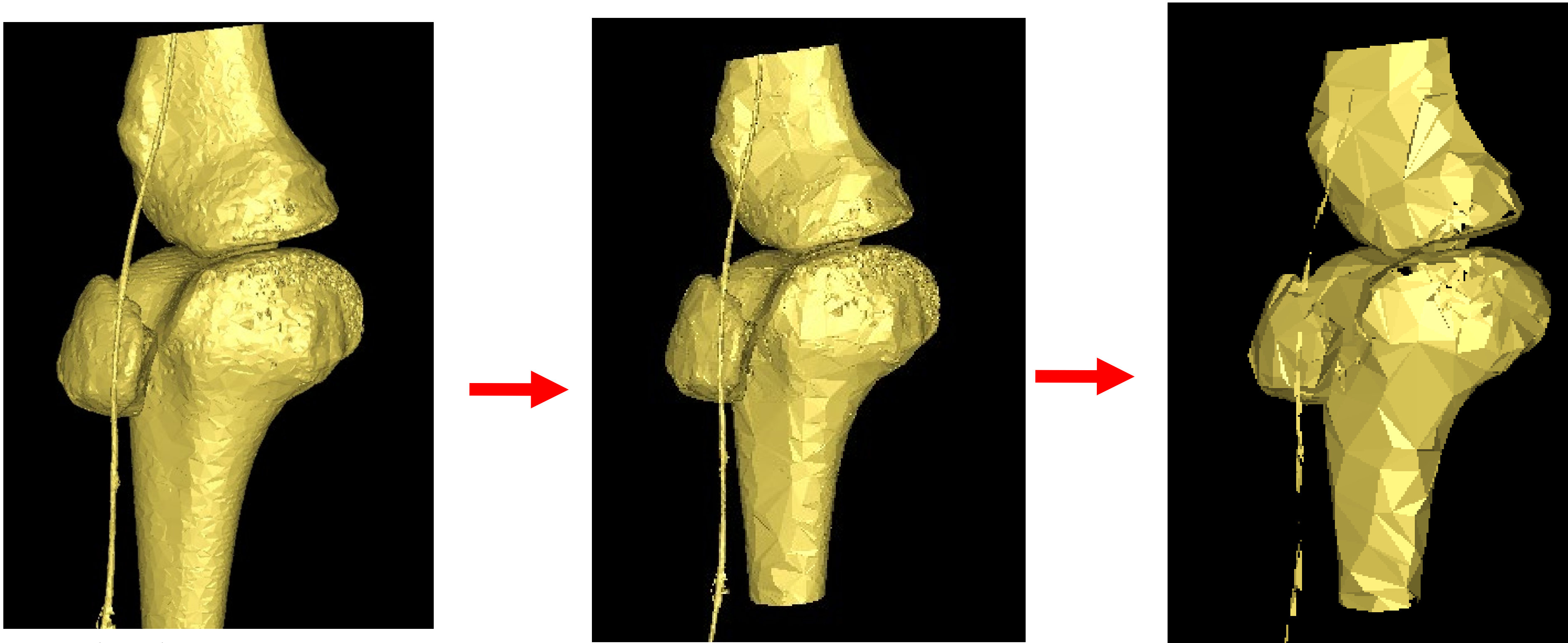


sub-sampling level = 3



sub-sampling level = 5

Mesh Decimation



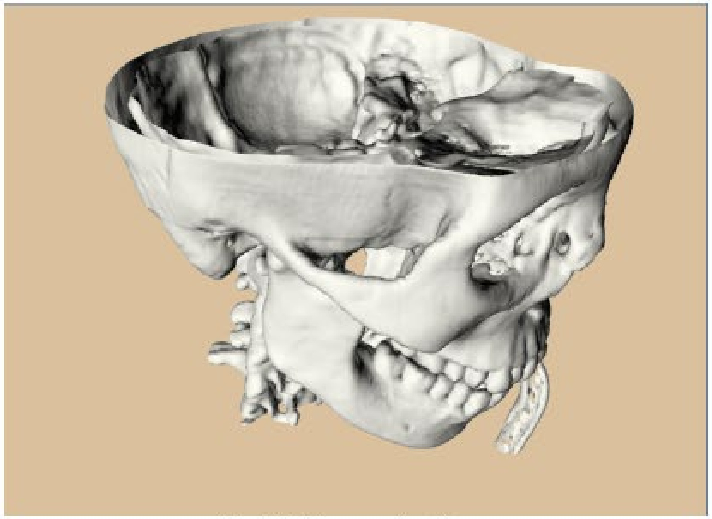
Triangles = 340997

Triangles =252717

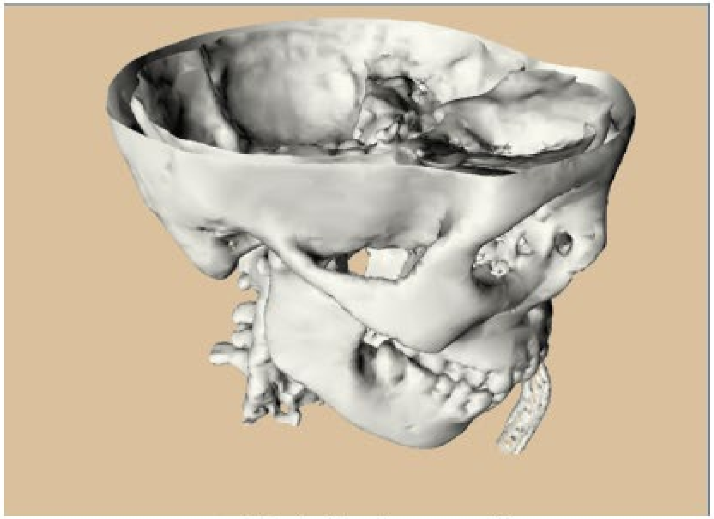
Triangles = 3303

- ***Mesh Decimation : intelligent mesh pruning***
  - **remove redundancy** in surface mesh representation

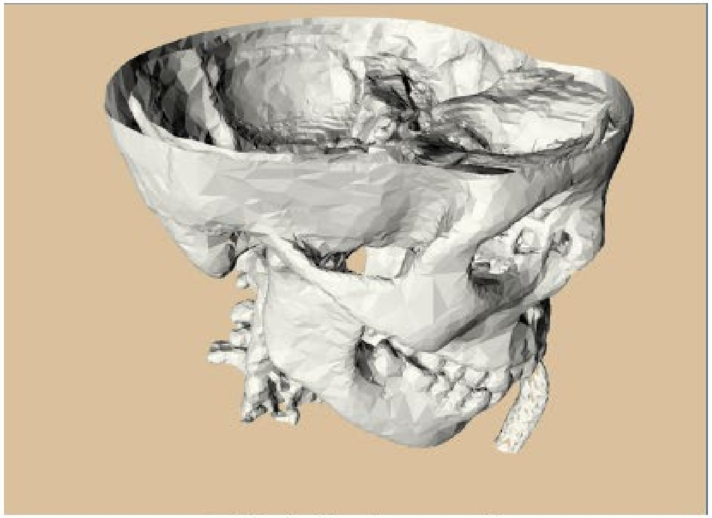
Decimation of a skeleton



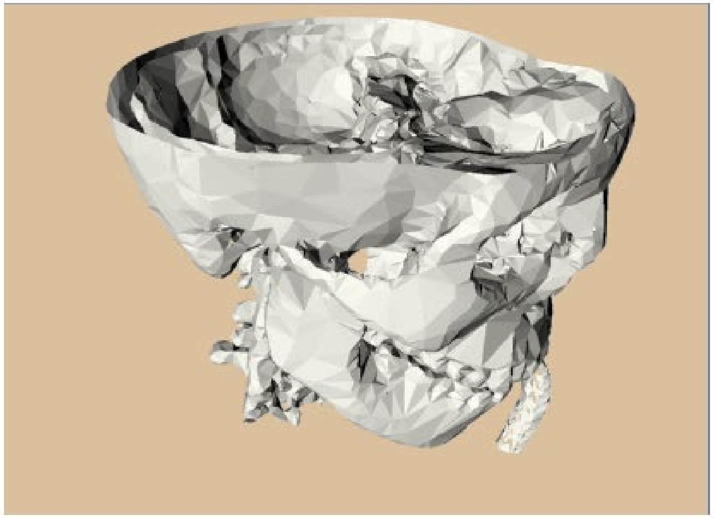
Full Resolution
(569K Gouraud shaded triangles)



75% decimated
(142K Gouraud shaded triangles)



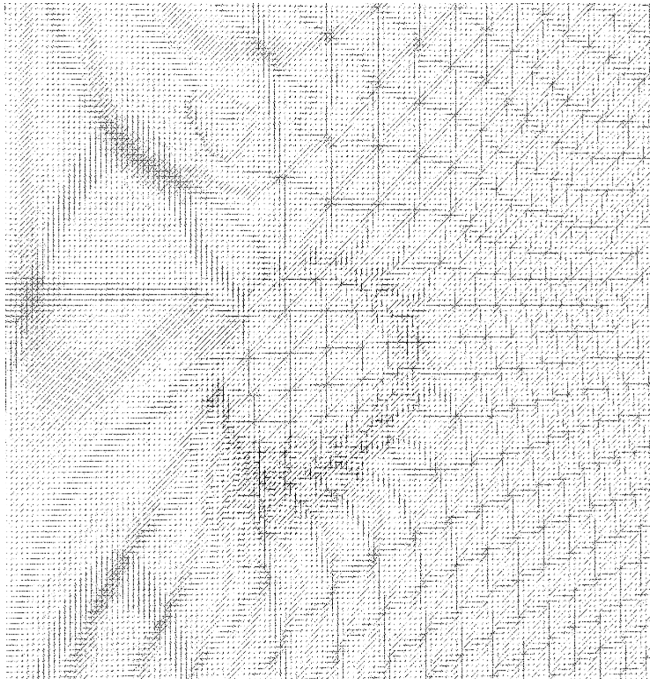
75% decimated
(142K flat shaded triangles)



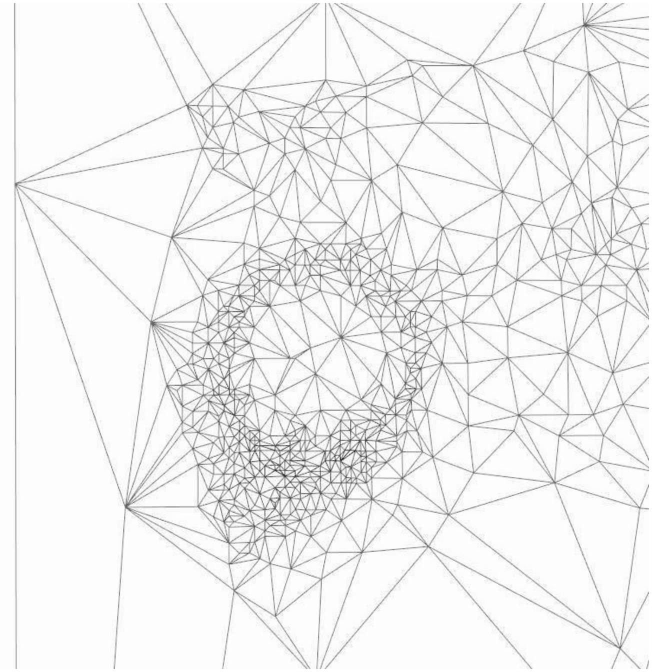
90% decimated
(57K flat shaded triangles)

Decimation : removing redundancy

Original terrain data of crater (point cloud)



90% Decimated mesh



- ***Original triangulation of point cloud – very dense***
- ***Many triangles approximate same planar area – merge co-planar triangles***
  - **Triangles in areas of high-curvature maintained**

Decimation : Schroeder's Method

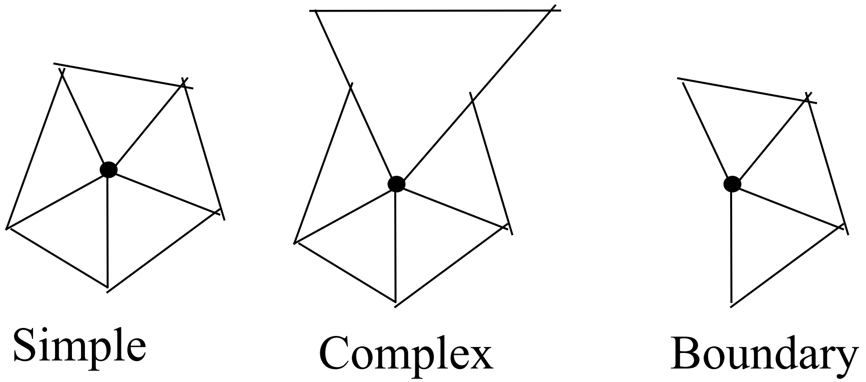
- ***Operates on triangular meshes***
  - 3D grid of regular triangular topology
  - **triangles are simplex**: reduce other topologies to triangular → simplex = triangle, tetrahedron, ...

- ***Schroeder's Decimation Algorithm:*** ***[Schroeder et al . '92]***

```
until stopping criteria reached

    for each mesh vertex

    classify vertex

    if classification suitable for decimation
       estimate error resulting from removal
       if error < threshold
            remove vertex
            triangulate over resulting mesh hole
```

Schroeder's Method : vertex classification

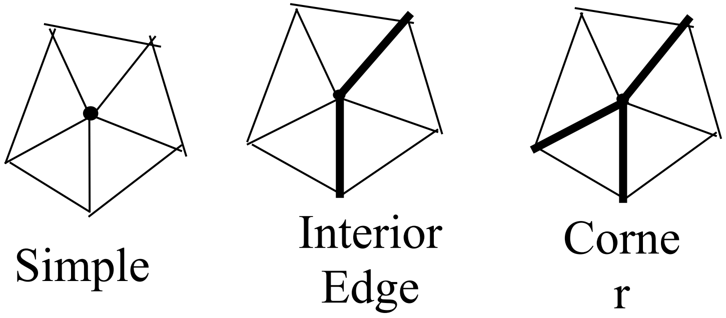
- ***Vertex classified into 5 categories:***

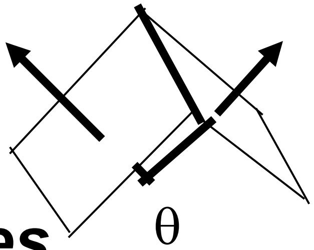


- ***Simple : surrounded by complete cycle of triangles, each edge used by exactly 2 triangles.***
- ***Complex : if edges not used by only 2 triangles in cycle***
- ***Boundary : lying on mesh boundary*** ***(i.e. external surface edge)***

Schroeder's Method : vertex classification

- ***simple vertices are further classified***
  - {simple | interior edge | corner} vertex
  - based on **local mesh geometry**
  - ***feature edge*** : where surface normal between adjacent triangles is greater than specified *feature angle*



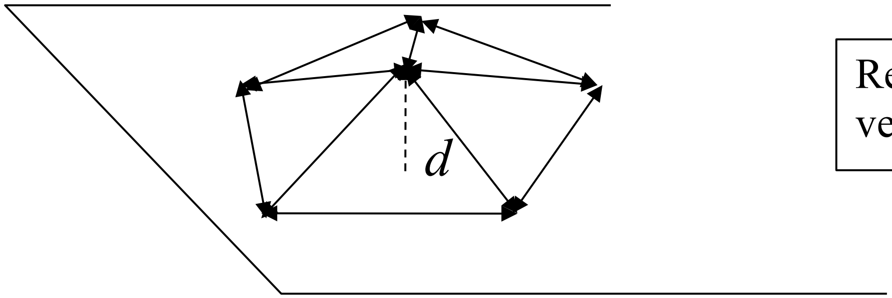


- ***interior edge : a simple vertex used by 2 feature edges***
- ***corner point : vertex used by 1, 3 or more feature edges***

Schroeder's Method : decimation criterion

- ***Remove : simple vertices that are not corner points***
  - i.e. leave important or complex mesh features

- ***Simple vertex***
  - mesh considered **locally "flat"** if **no feature edges**
  - **estimate error** from removing this vertex
    - error = distance *d* of vertex to average plane through neighbours

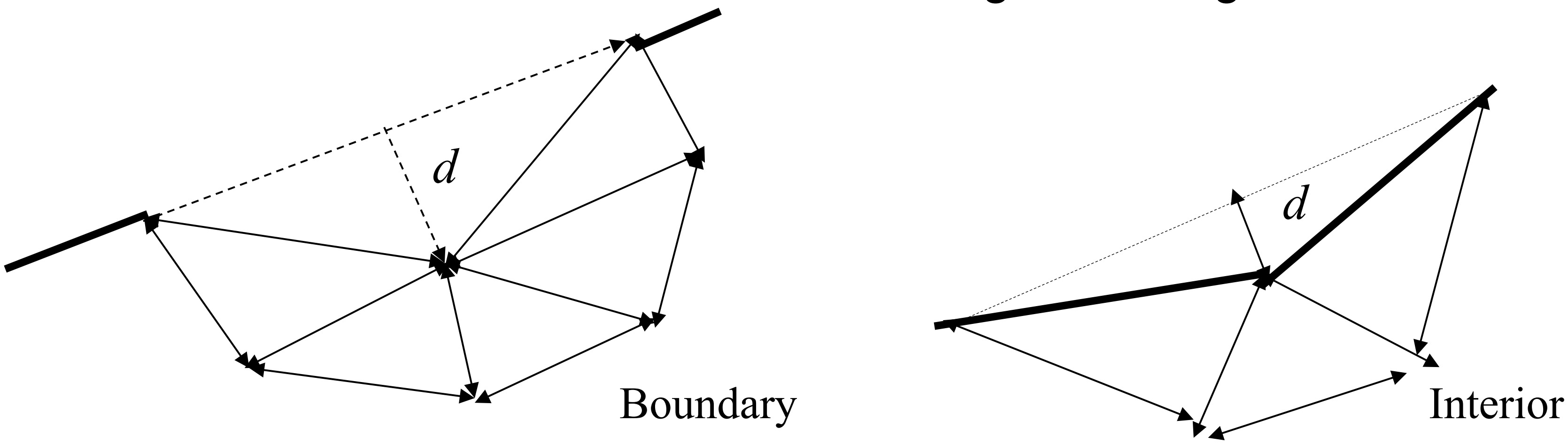


Represents the error that removal of the vertex would introduce into the mesh.

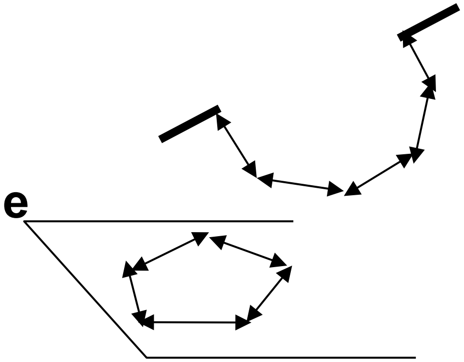
Average plane through surrounding vertices.

Schroeder's Method : decimation criterion

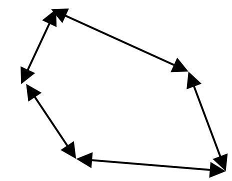
- ***When the vertex is close to the edge***
  - **vertex point considered to lie on edge**
    - error = distance *d* from vertex to edge resulting from removal



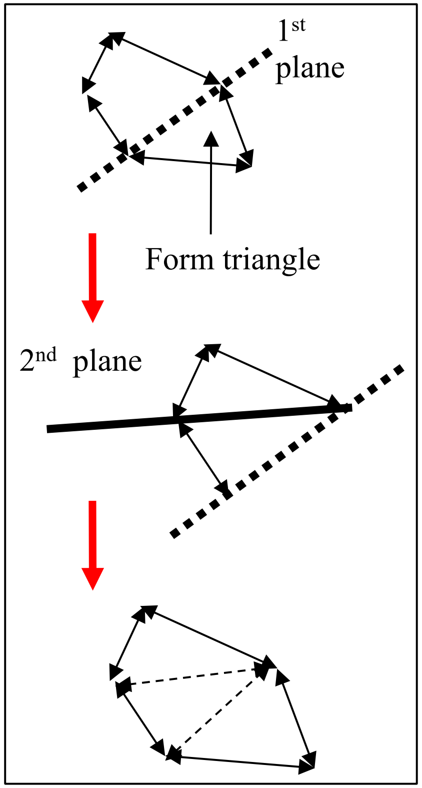
- ***Decimation Criterion : vertex removal***
  - if error **d < threshold** then **remove**
    - **vertex & all associated triangles removed ⇒ hole**



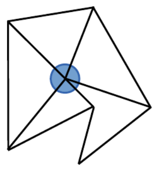
Schroeder's Method: re-triangulation



- ***Hole must be re-triangulated***
  - use less triangles
  - resulting boundary is in 3D space ⇒ **3D triangulation**



- **Recursive Division Triangulation Strategy**
  - **choose a split plane (in 3D), split into two sub-loops**
  - check it is valid – all points in each sub-loop lie on opposite sides of the plane.
    - If failure, do not remove the vertex
  - choose new split plane and recursively until all sub-loops contain 3 points
  - This is helpful for dividing concave polygons
  - form triangles

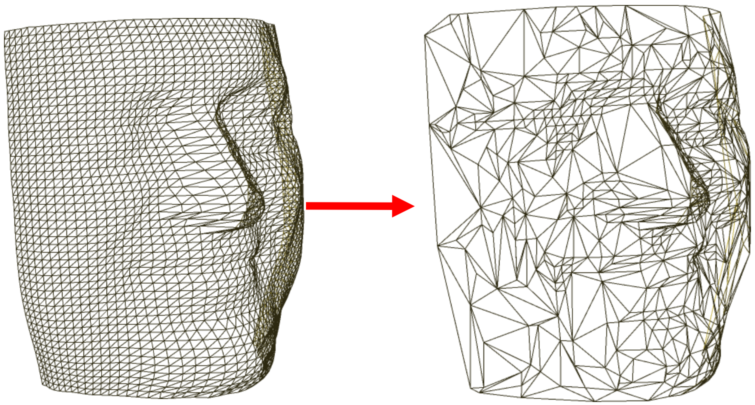


Decimation : stopping criteria

- ***Option 1 : maximum error***
  - max. error that can be suffered in mesh due to removal
    - measure of **maintaining mesh quality** (as a representation of original surface form)
- ***Option 2 : target number of triangles***
  - **number of triangles required in resulting decimation**
    - **explicitly specifying the representational complexity of the resulting mesh**
- ***Combination : combine 1 & 2***
  - **stop when either is reached**
    - **aims for target reduction in triangles but keeps a bound on loss of quality**

Decimation Discussion

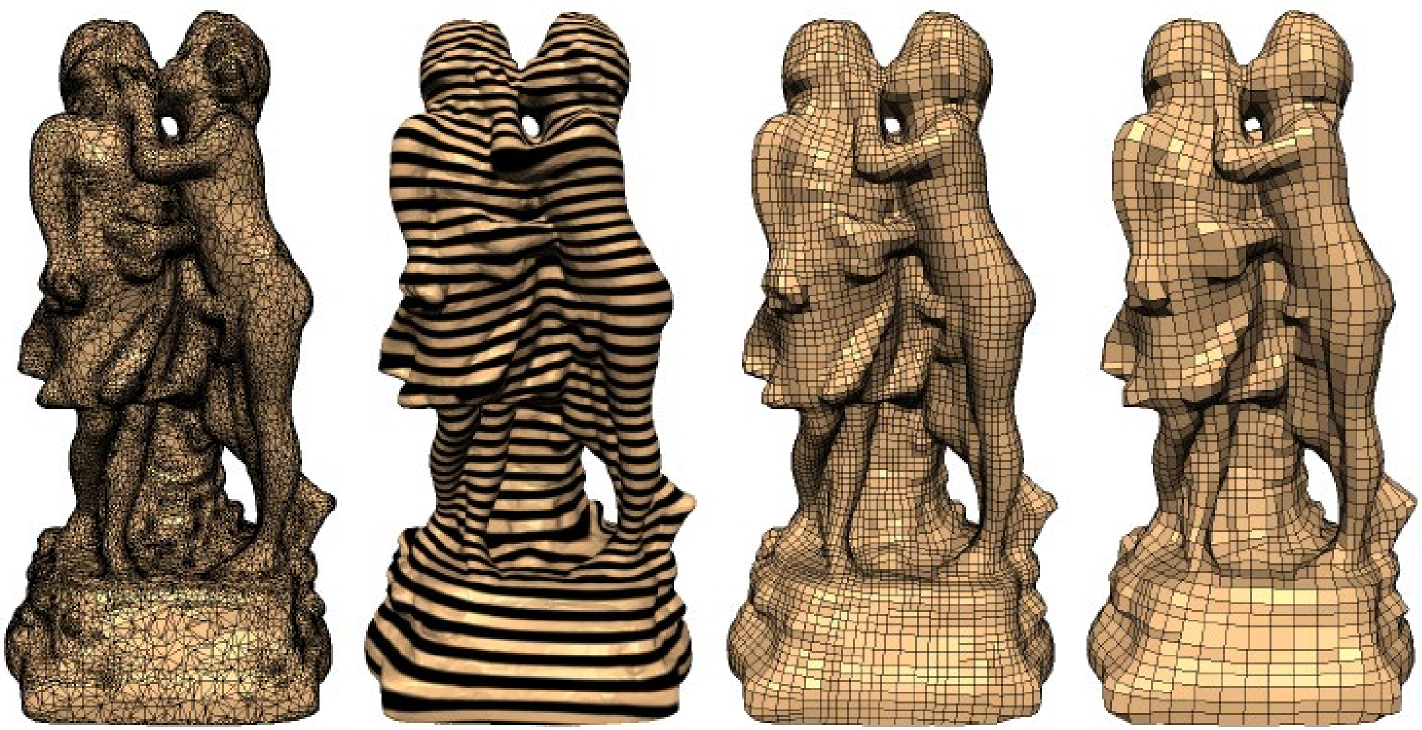
- ***Improve Efficiency of Representation***
  - **remove redundancy** within some measure of error
- ***Decimation : Mesh Compression / Polygon Reduction***
  - **generate smaller mesh representation**
  - information is permanently removed
    - **lossy compression**



  - same concept as lossy image compression *(e.g. JPEG)*
- ***Advantages : less to store & less to render***
- ***Dis-advantages : less detail***

Remeshing & Simplification

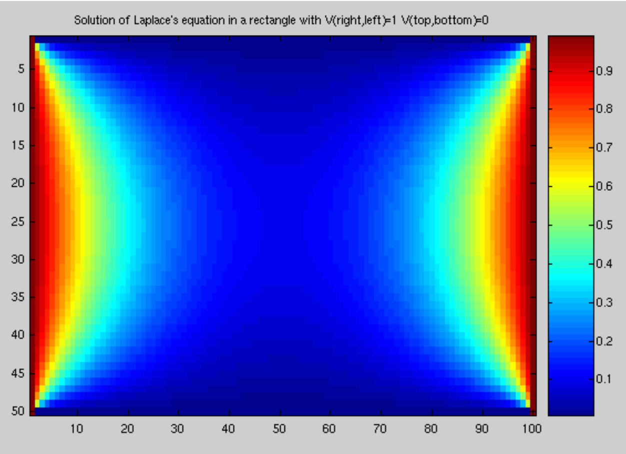
- After the decimation, the triangle shapes can be very skewed.
- We may prefer a more uniform point distribution
- Also sometimes we prefer quadmeshes → simulation



Harmonic Scalar Fields

$$\Delta u = 0 \ \text{s.t.} \ u_i = c_i$$

- Laplace equation (↑ $\Delta u = 0$) boundary conditions (↑ $u_i = c_i$)
- A u that satisfies this equation is called harmonic function
- u produces a smooth transition between u_i s.
- When heat diffuses over some material, it follows the Laplace equation



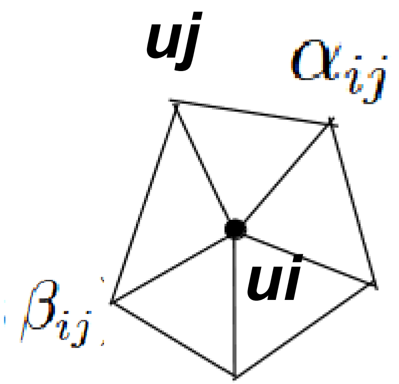
Using Harmonic fields for computer graphics

- ***By defining a harmonic function over the surface of a 3D object, we can produce a scalar field***
- ***This can be used for various purposes, such as***
  - Re-meshing
  - Simplification
  - Matching different shapes

Laplace Equation on a Triangle Mesh

At each vertex, the Laplacian of the parameter u can be written as

$$\Delta u_i = \sum_{j \in N_i} w_{ij}(u_j - u_i)$$



where the weights are computed by

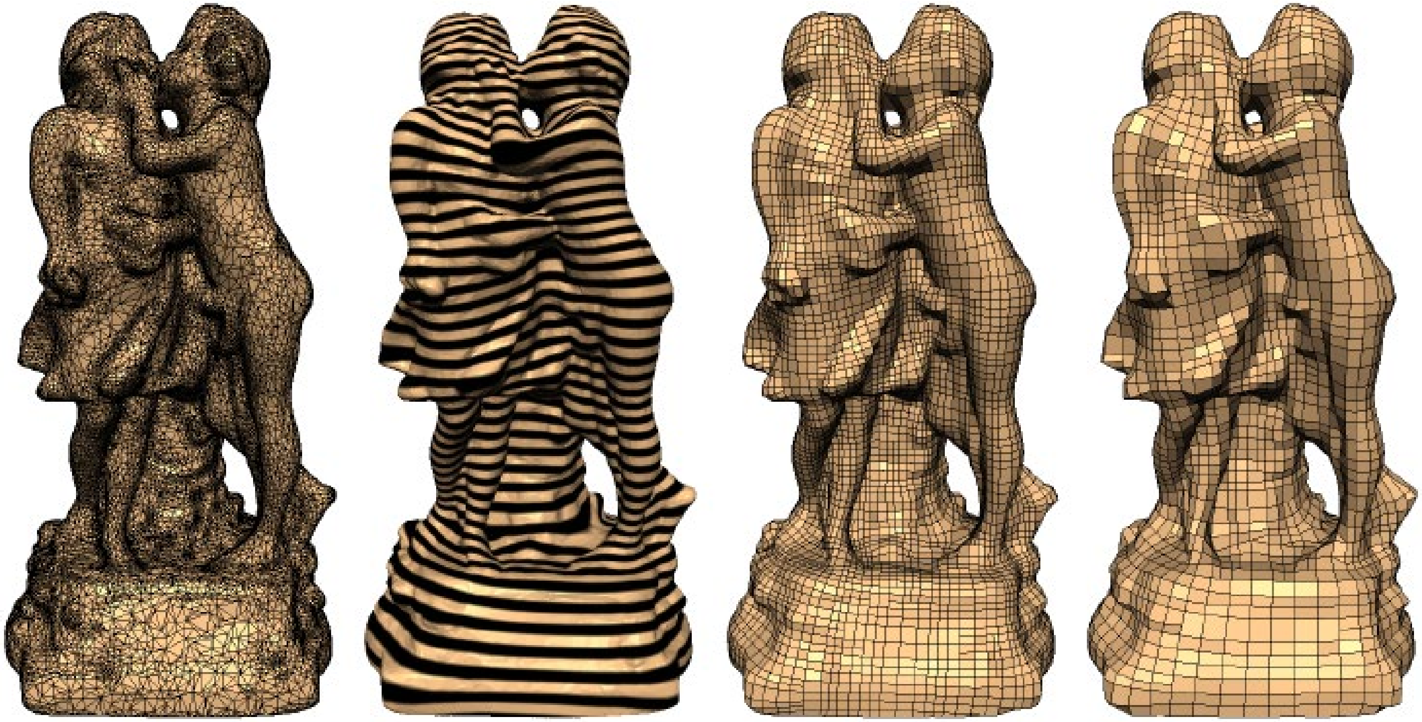
$$w_{ij} = -\frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij})$$

Solve for $\Delta \mathbf{u} = -\mathbf{L}\mathbf{u} = 0$

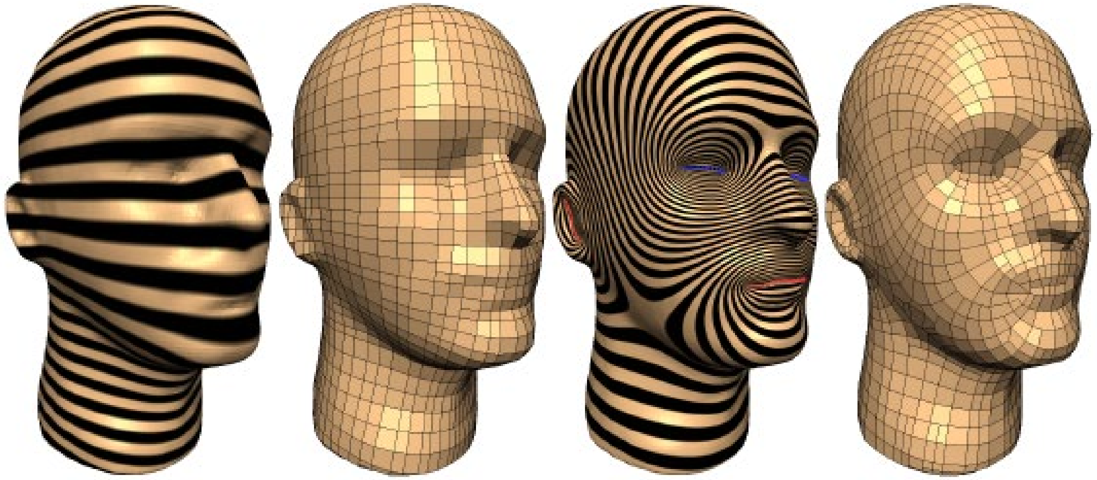
subject to Dirichlet boundary constraints

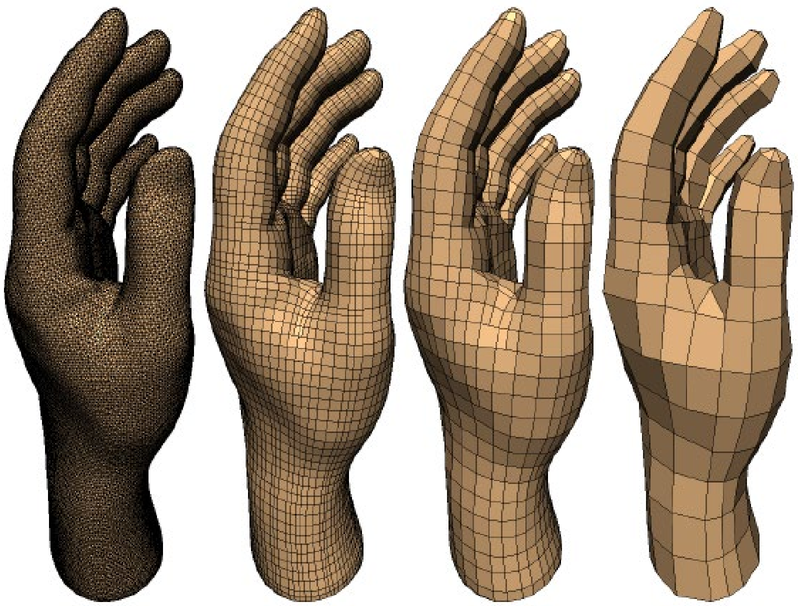
Remeshing & Simplification by Harmonic Scalar Field

- Using the harmonic scalar field and the isoparametric lines, we can re-mesh the object
- Sampling on the surface uniformly with the u.
- By reducing the sampling rate, we can conduct simplification



More results





(a) Input (b) Fine (c) Medium (d) Coarse

(a) Input (b) Fine remesh (c) Coarse remesh

Summary

- ***Modelling Algorithms for 3D surface data***
  - ***Triangulation : Delaunay & iso-surfacing***
  - ***Decimation : Schroeder's Method***
  - ***Registration : Iterative Closest Point (ICP)***
  - ***Smoothing : Laplacian mesh smoothing***
  - ***Remeshing : Using the Laplace equation***

Readings

- Hoppe et al. "Progressive Mesh", SIGGRAPH '96
- Shroeder et al. "Decimation of Triangle Meshes", SIGGRAPH '92
- Rusinkiewicz, Levoy, "Efficient variants of the ICP algorithm"
- Miller, "Efficient algorithms for local and global accessibility shading", SIGGRAPH '94
- Harmonic functions for quadrilateral remeshing of arbitrary manifolds Dong et al. '2004