Processing 3D Surface Data

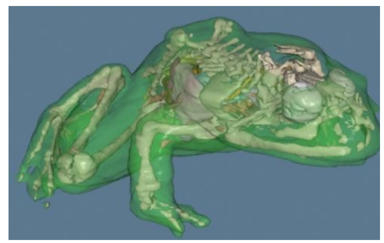
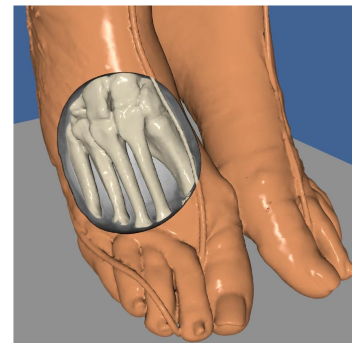
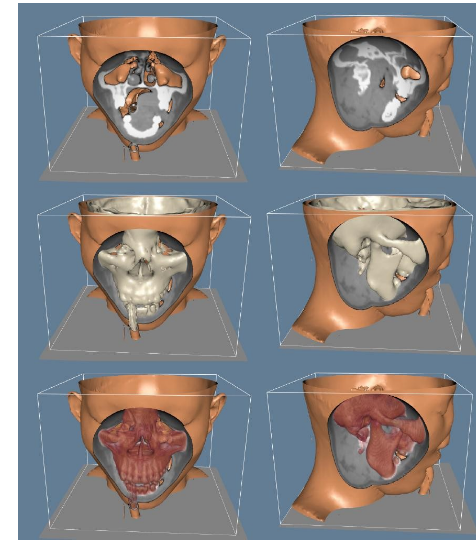
Visualisation – Lecture 16

Taku Komura

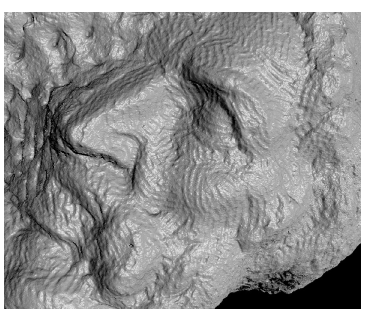
Institute for Perception, Action & Behaviour
School of Informatics

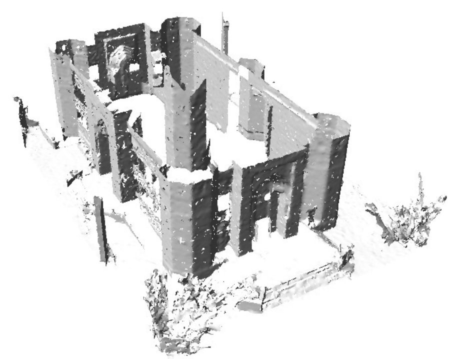
3D surface data ... where from ?

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

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






- Today : ***modelling algorithms for 3D meshes***

Processing 3D data

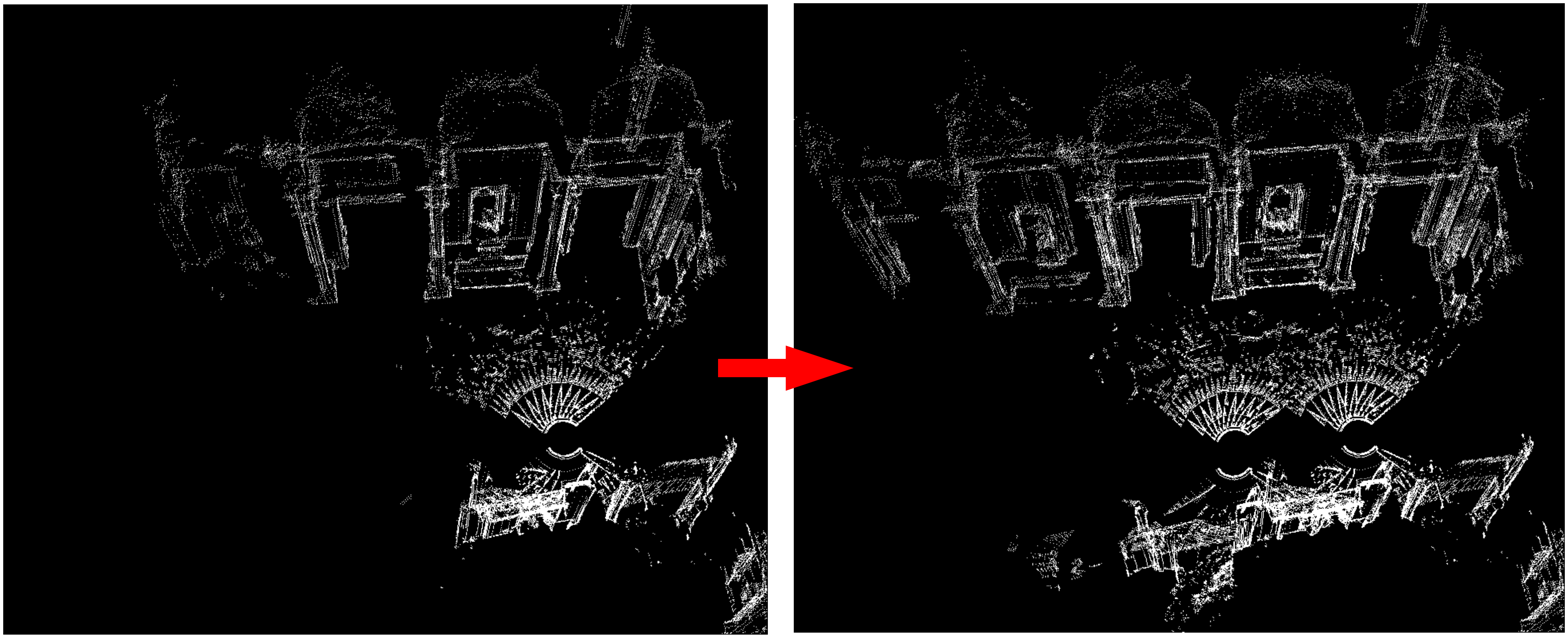
1) Capture the data (by stereo vision, range scanner)
2) Registration (if the data was captured by multiple attempts)
3) Adding the topology : converting to mesh data
4) Smoothing
5) Decimation

Great Buddha Project in Japan

- Capturing took 3 weeks x 2 trips x 10 students/staff
- http://www.cvl.iis.u-tokyo.ac.jp/movie/Nara_English_small.mpg
- 

3D Registration

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

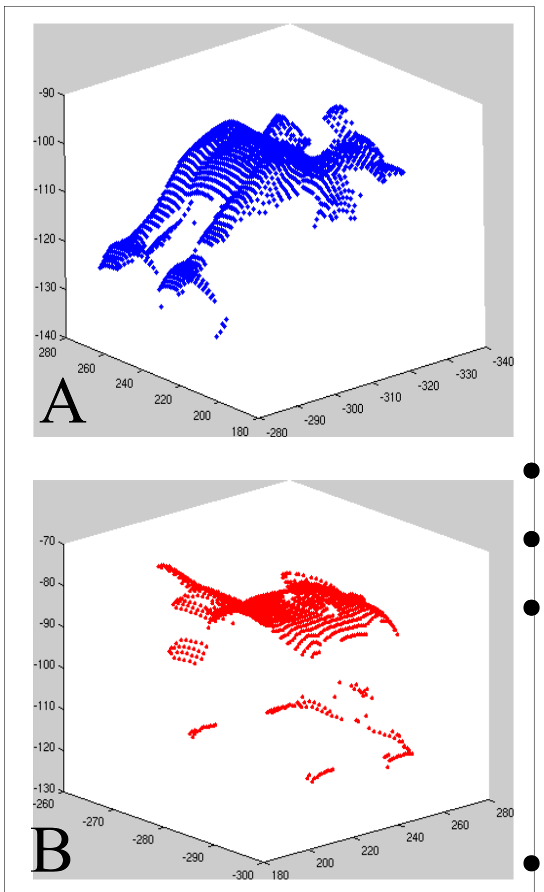


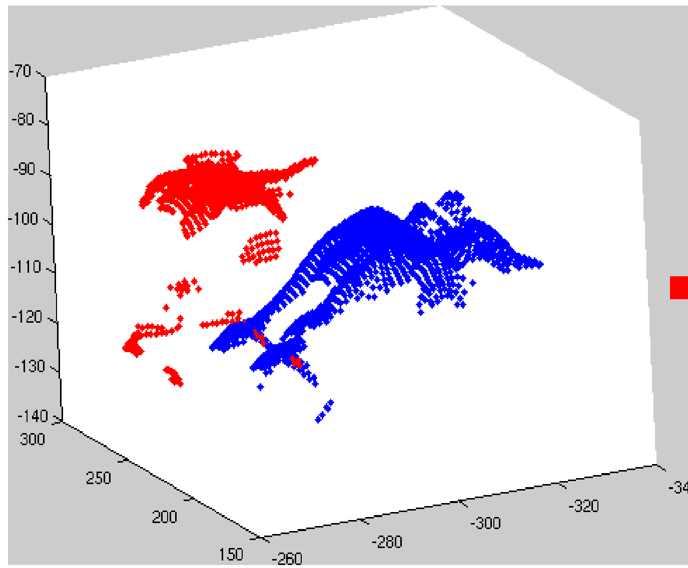
Registration – ICP algorithm

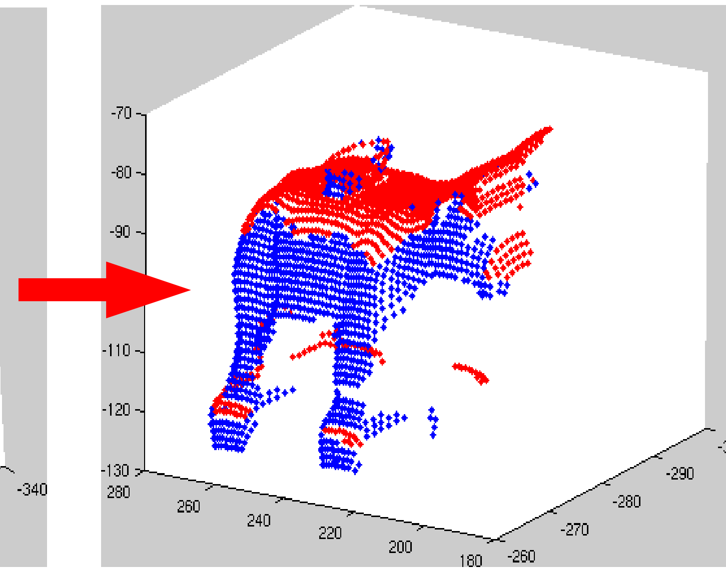
- **Problem** : merging of 3D point clouds from different, unaligned orientations
  - common in large scale range scanning *(e.g. Mosque)*
- **Solution :** iterative estimation of rigid 3D transform between point clouds *[**Iterative Closest Point (ICP)** Besl / Mckay '92]*
- ***ICP algorithm:*** *for 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*

Registration – ICP algorithm

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



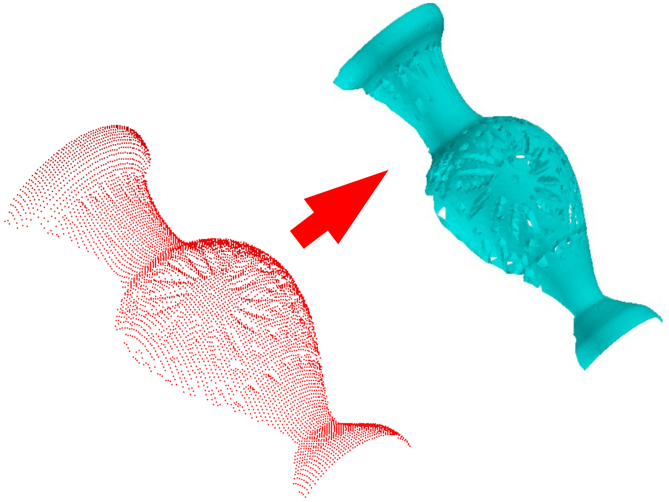




- **Input:** point clouds acquired from non-aligned viewpoints (from 3D range scanner) & initial estimation of registration
- **Output:** Transformation

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
  - **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/Triangles) to build surface

Mesh Smoothing - 1

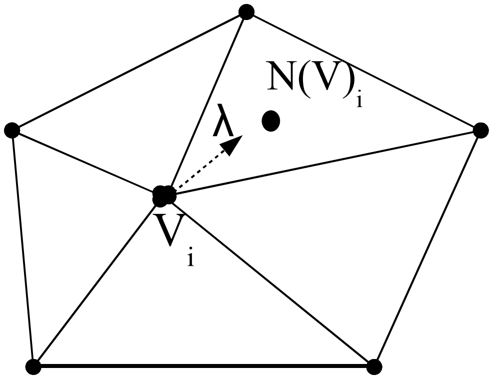
- **Surface Noise** : surfaces from stereo vision and (large scale) laser scanning often contain noise
  - removes noise, improves uniform surface curvature → helps decimation

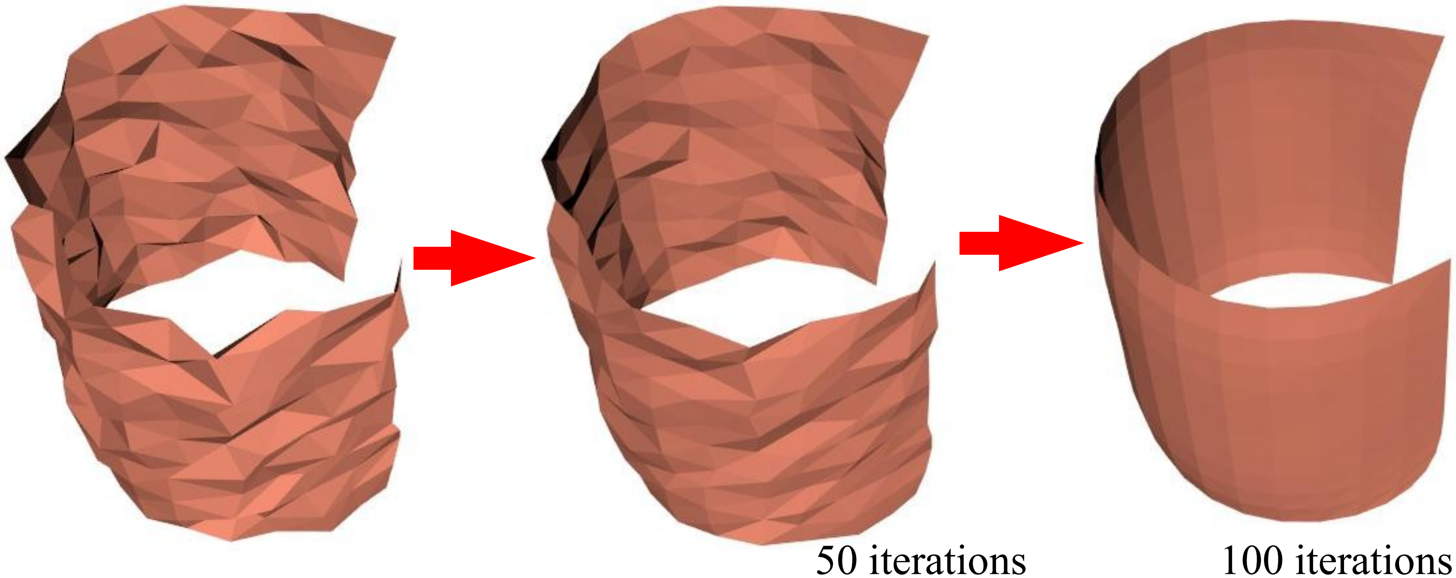
- **Solution**: Laplacian mesh smoothing
  - **modifies geometry** of mesh
  - **topology unchanged**
  - **reduces surface curvature** and **removes noise**

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

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

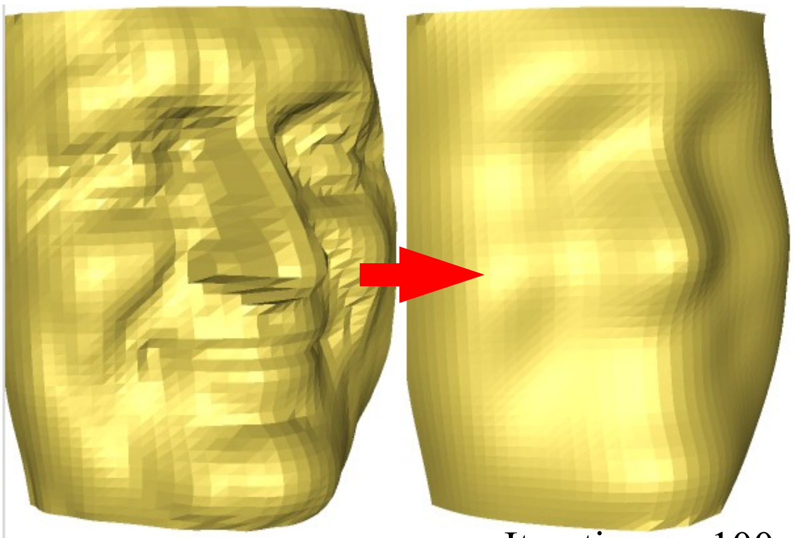




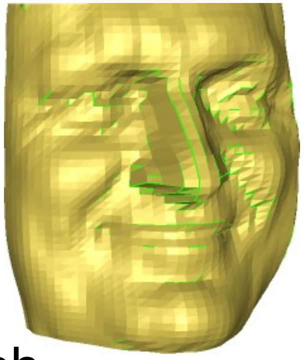
λ = 0.01

Mesh Smoothing - 3

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



Iterations = 100; λ = 0.1



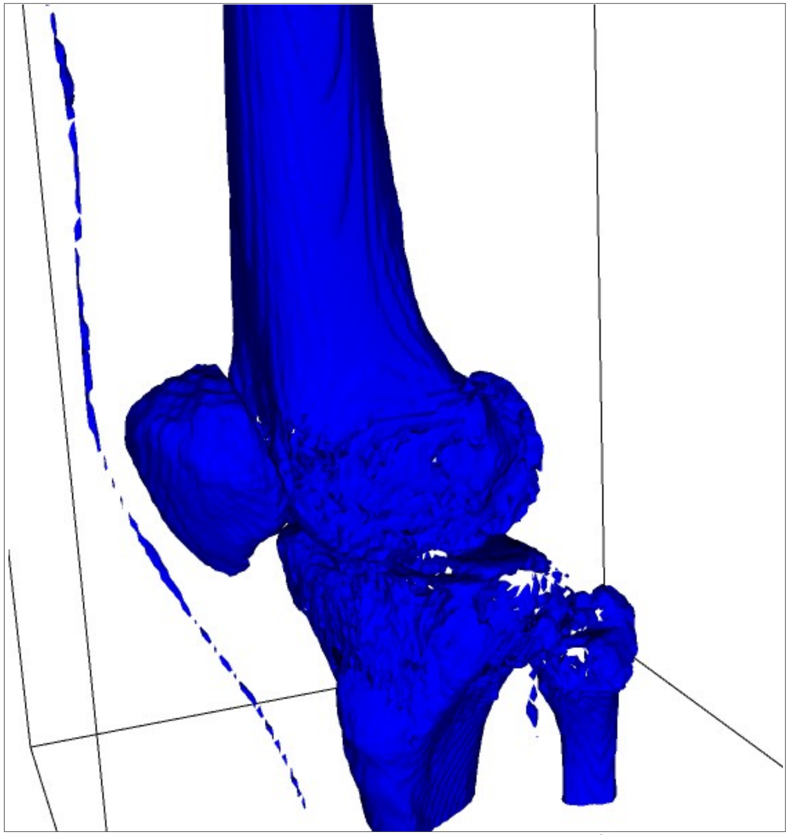
- **Limitations :** loss of detail, mesh shrinkage
  - *enhancements* : feature-preserving smoothing & non-shrinking iterative smoothing
  - feature points can be "anchored" to prevent movement in mesh smoothing (detail preservation)

Handling Large Polygonal Datasets

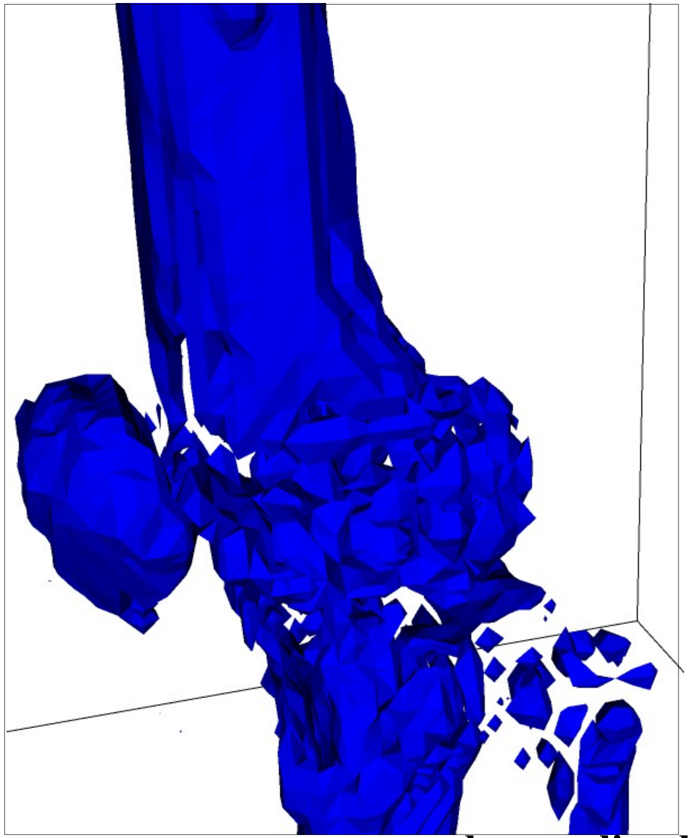
- **Large Surface Meshes**
  - **Marching Cubes** : voxel dataset of 512^3 produces a **typical iso-surface of 1-3 million triangles**
  - **Laser Scanning** : datasets in **10s millions of triangles**
- **Problem** : lots to render!
- **Solution** : polygon reduction
  - **sub-sampling** : *simple sub-sampling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



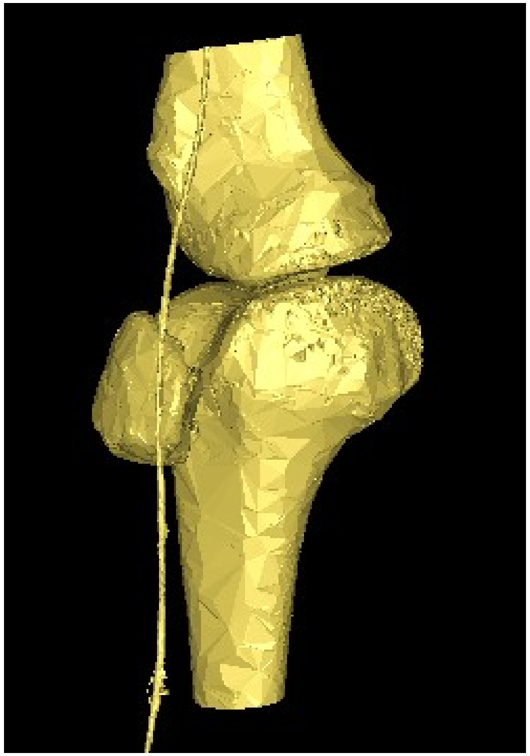
sub-sampling level = 3



sub-sampling level = 5

Mesh Decimation

Triangles
= 340997

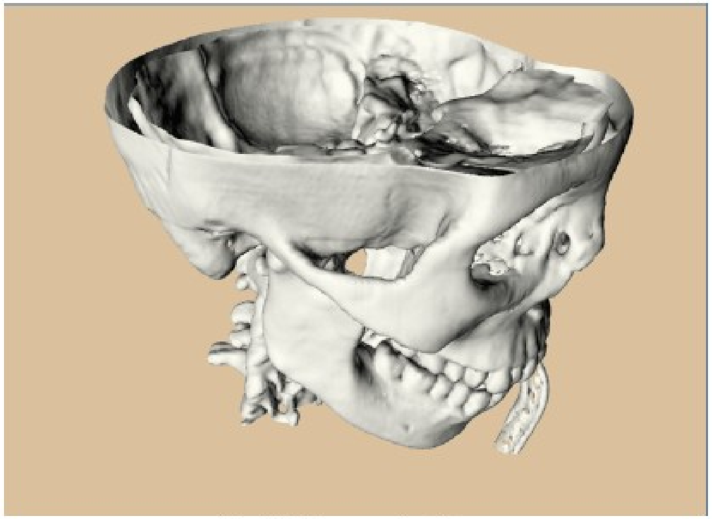


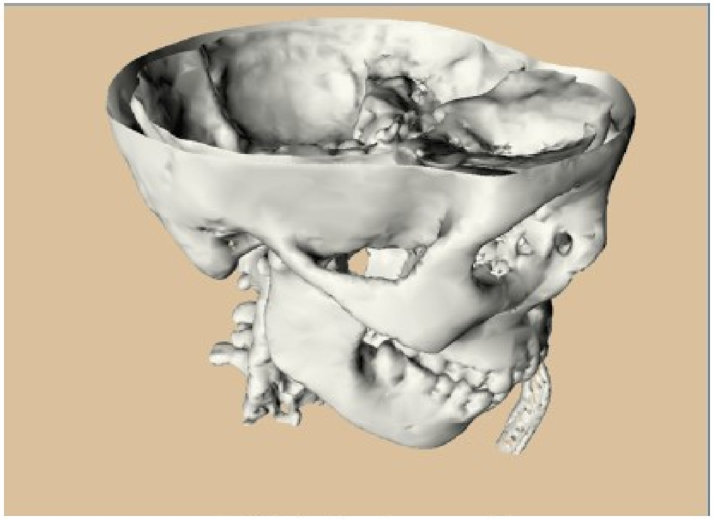
Triangles
=252717

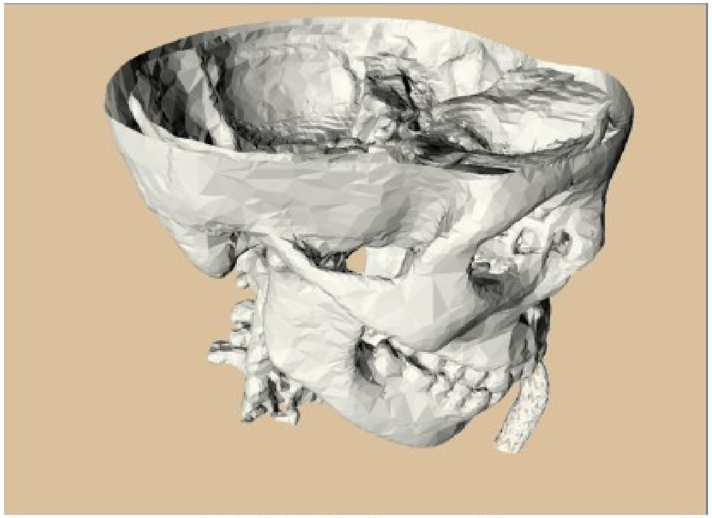
Triangles
= 3303

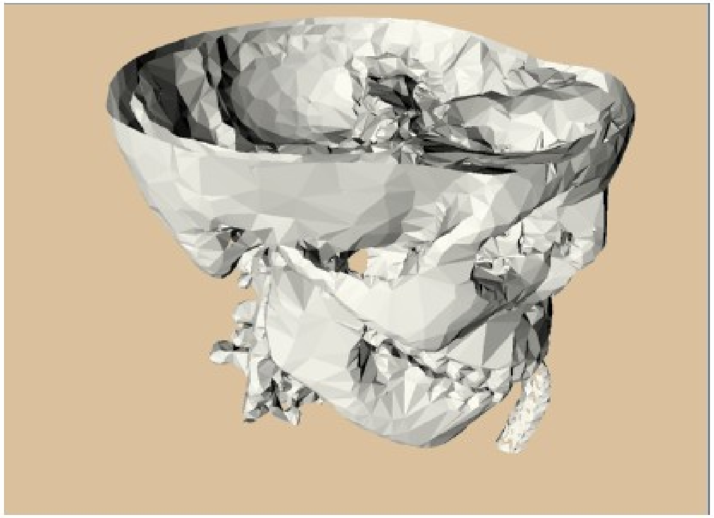
- Mesh Decimation : **intelligent mesh pruning**
  - **remove redundancy** in surface mesh representation

Decimation of a skeleton


Full Resolution
(569K Gouraud shaded triangles)

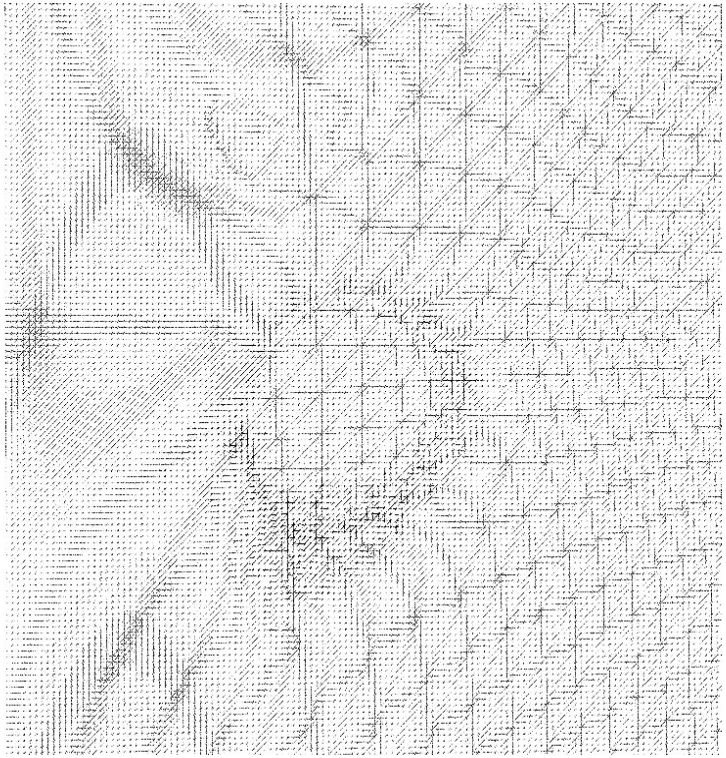

75% decimated
(142K Gouraud shaded triangles)


75% decimated
(142K flat shaded triangles)

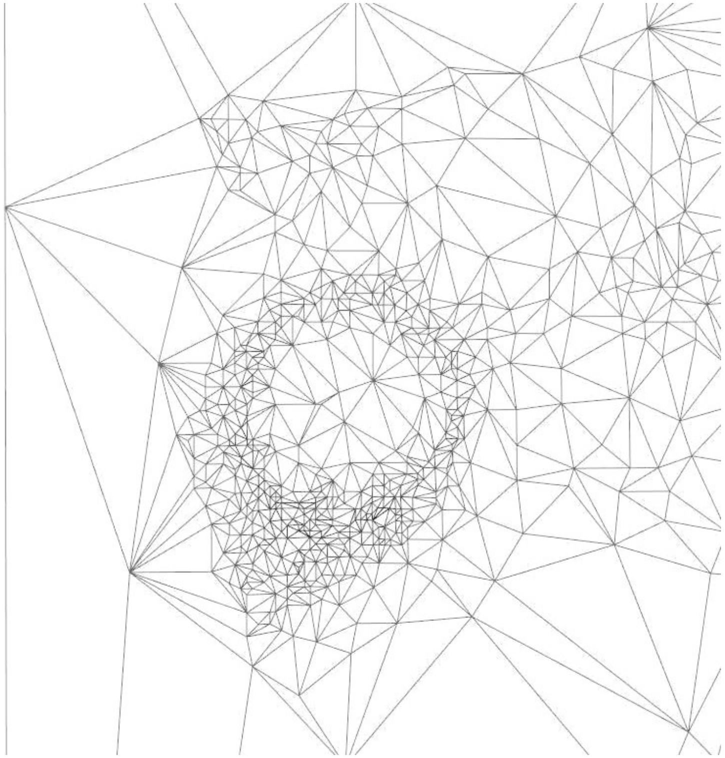

90% decimated
(57K flat shaded triangles)

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

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

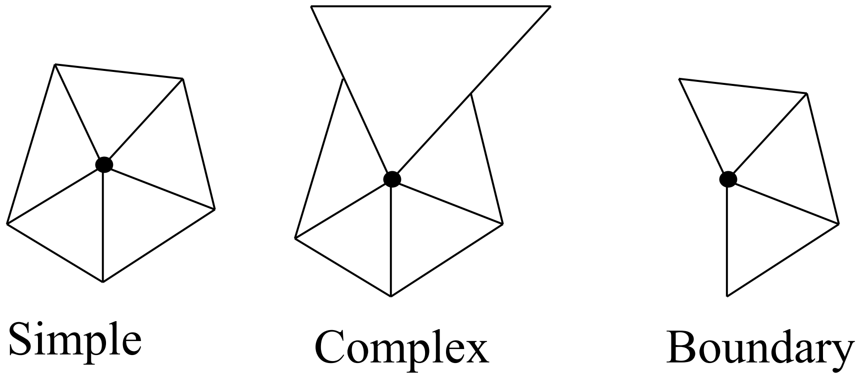
```
until stopping criteria reached
   for each mesh vertex
      classify vertex
         if classification suitable for decimation
            estimate error resulting from removal
            if error < threshold
               remove vertex
               triangulate over resulting mesh hole
```

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**

Schroeder's Method : vertex classification

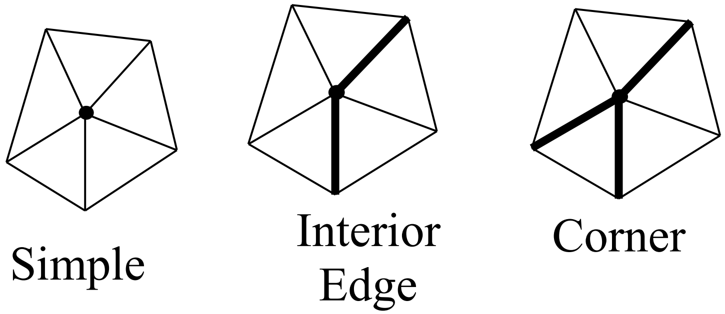
- Vertex classified into **5 categories**:

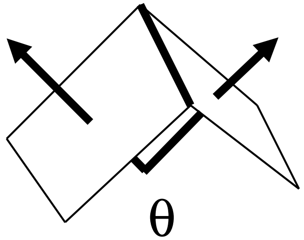


- **Simple** : surrounded by complete cycle of triangles, each edge used by exactly 2 triangles.
- **Complex** : if edges not used by only 2 triangles in cycle
- **Boundary :** lying on mesh boundary *(i.e. external surface edge)*

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*



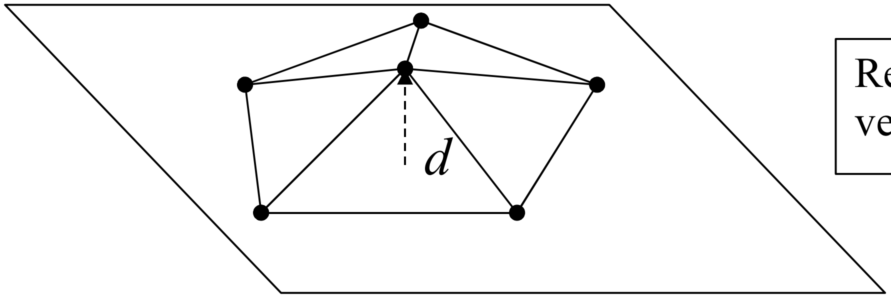


- **interior edge** : a simple vertex used by 2 feature edges
- **corner point** : vertex used by 1, 3 or more feature edges

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"** : as **no feature edges**
  - **estimate error** from removing this vertex
    - error = distance *d* of vertex to average plane through neighbours

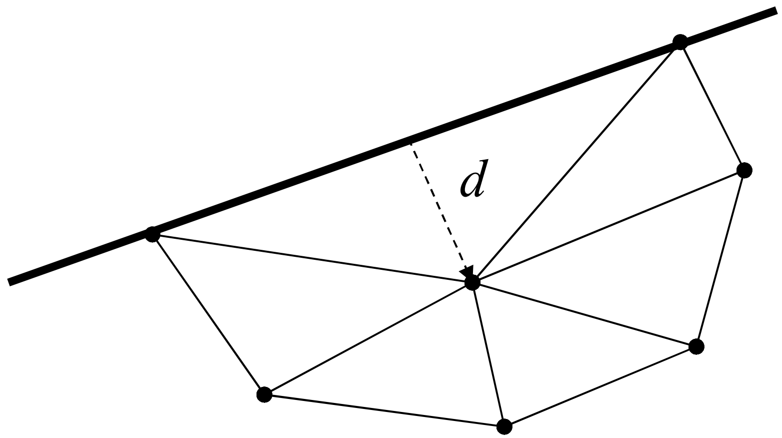


Represents the error that removal of the vertex would introduce into the mesh.

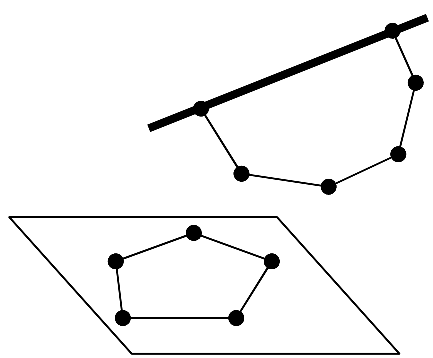
Average plane through surrounding vertices.

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

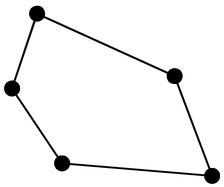


- **Decimation Criterion :** vertex removal
  - if error **d < threshold** then **remove**
    - vertex & all associated **triangles removed** ⇒ **hole**

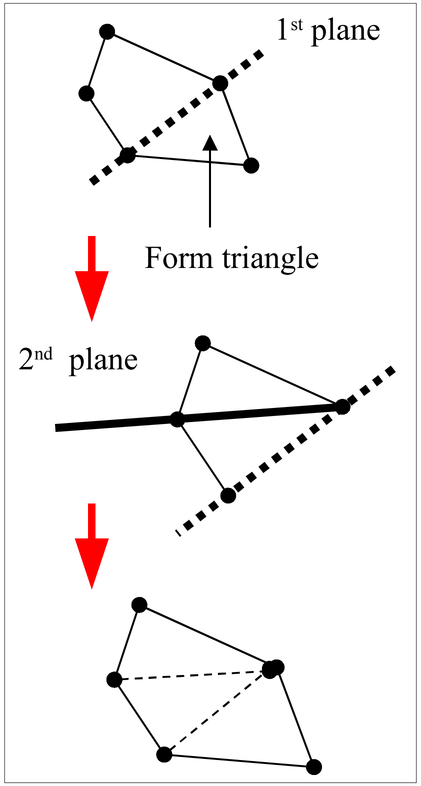


Schroeder's Method: re-triangulation

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

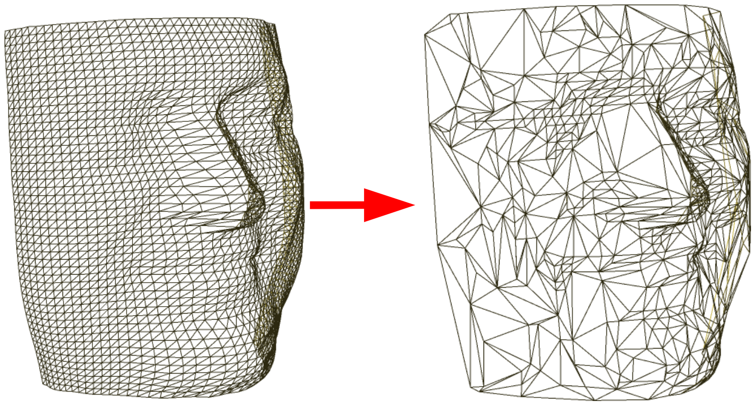


  - **Recursive Division Triangulation Strategy**
    - choose a split plane (in 3D), split into two sub-loops
    - check it is valid – all points in each sub-loop lie on opposite sides of the plane.
    - choose new split plane and recurse until all sub-loops contain 3 points
    - form triangles



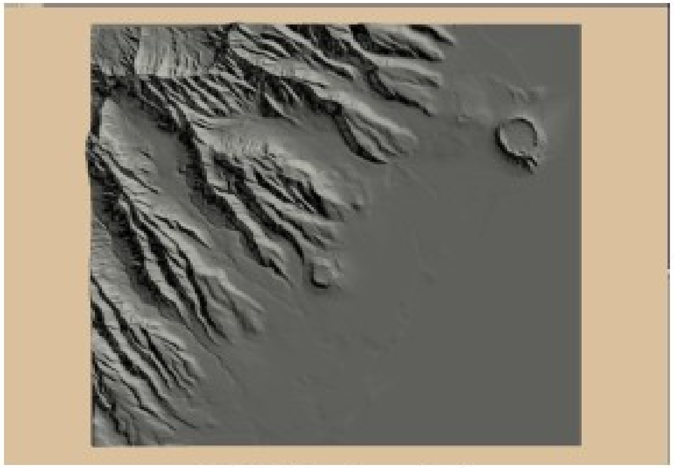
Decimation Discussion

- **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**

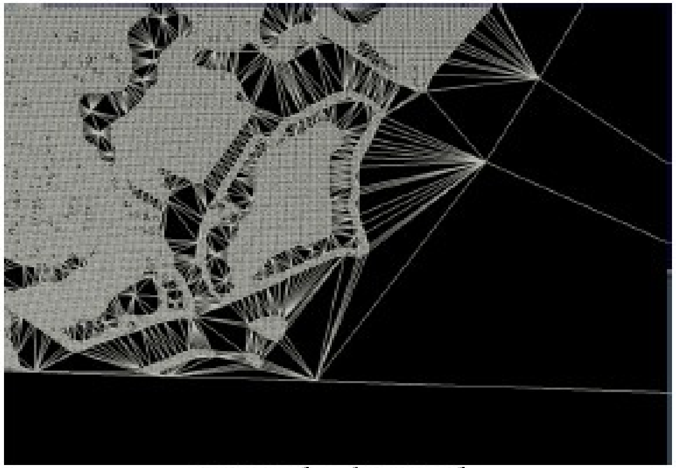


  - same concept as lossy image compression *(e.g. JPEG)*
- **Advantages** : less to store & less to render
- **Dis-advantages** : less detail

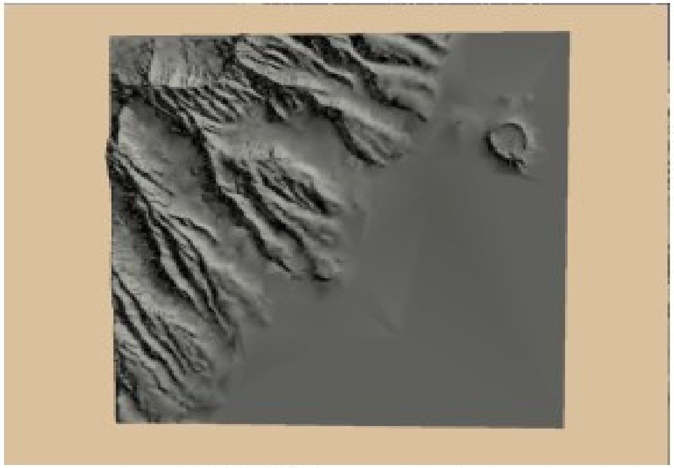
Some more decimation results



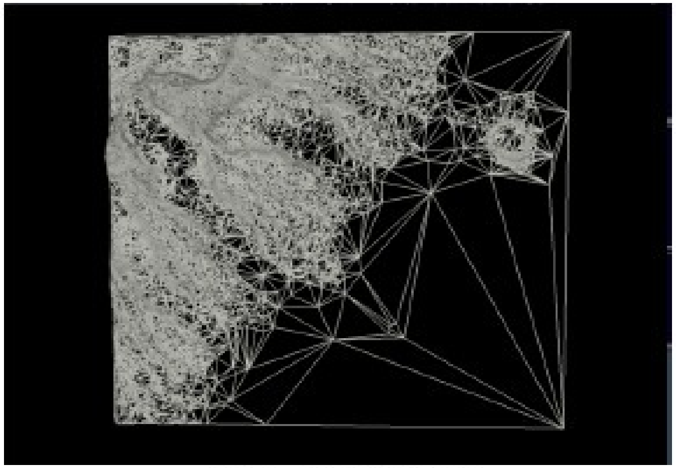
32% decimated
(276K flat shaded triangles)



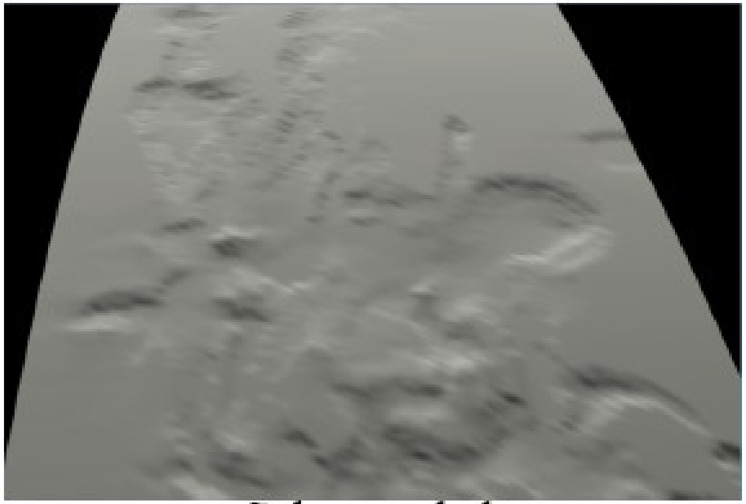
32% decimated
(shore line detail, wireframe)



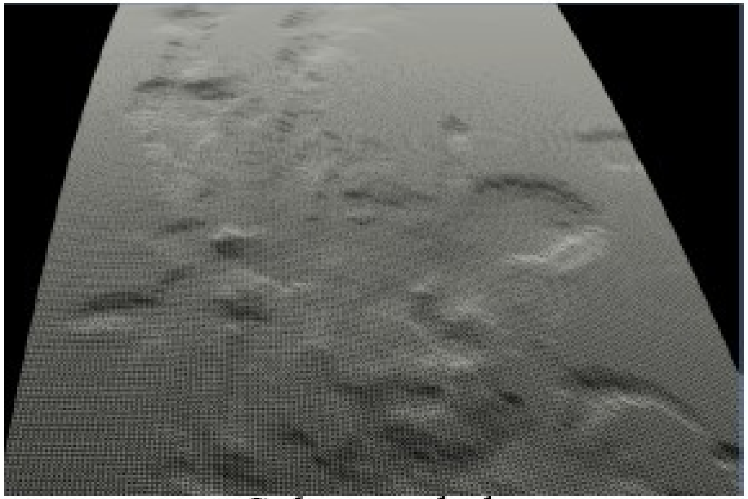
90% decimated
(40K Gouraud shaded triangles)



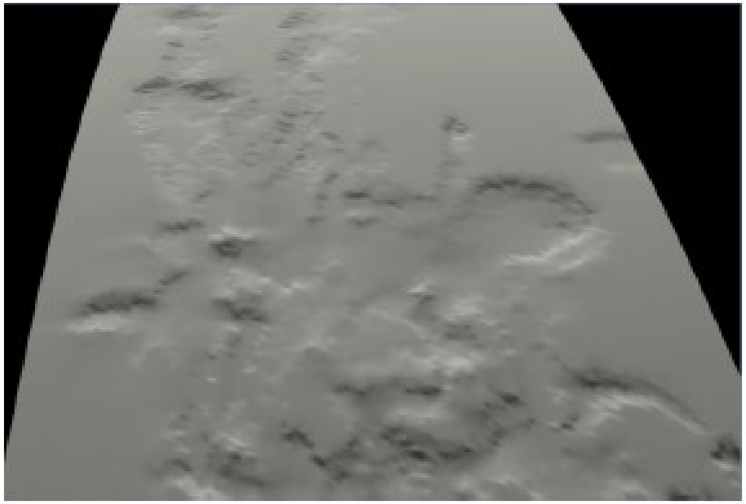
90% decimated
(40K wireframe)



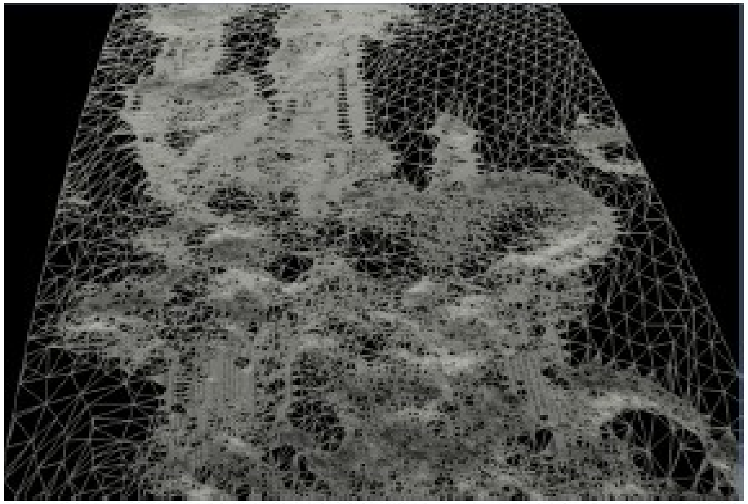
Sub-sampled
(68K Gouraud shaded triangles)



Sub-sampled
(68K wireframe)



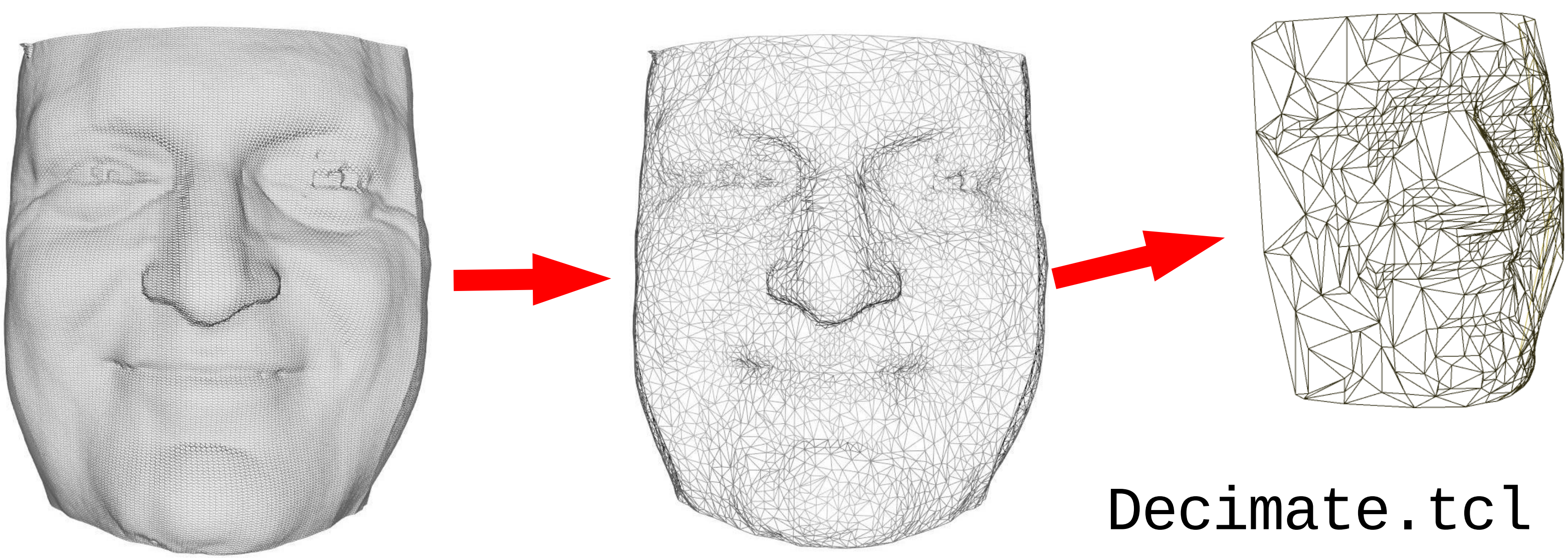
77% decimated
(62K Gouraud shaded triangles)



77% decimated
(62K wireframe)

Decimation in VTK

- `vtkDecimate` *(Schroeder's Method)*
  - PolyData input, *decimated* PolyData output
  - position between Marching Cubes filter (or alt. triangle source) and output Mapper object
  - **stopping criteria :** specify fraction of polygons to remove



`Decimate.tcl`

Summary

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