Introducción a la Fotografia 3D

UBA/FCEN Marzo 27 – Abril 12 2013

Clase 5 : Lunes Abril 8

Gabriel Taubin

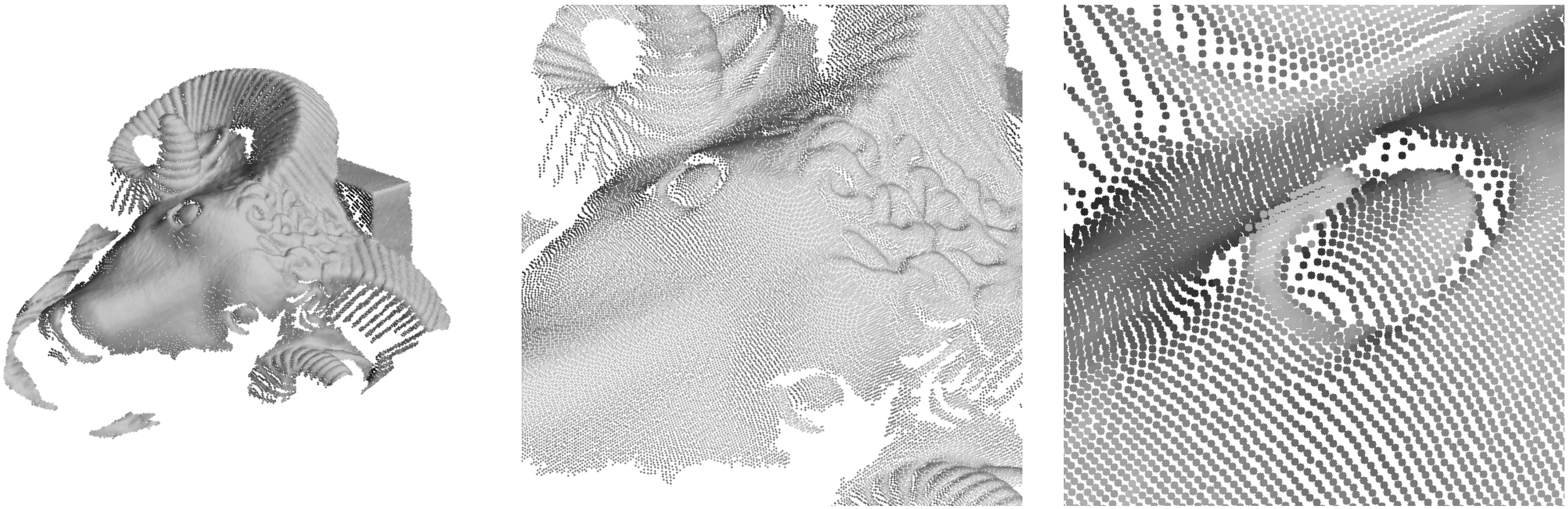
Brown University

BROWN

Course Schedule

- Introduction
- The Mathematics of 3D Triangulation
- 3D Scanning with Swept-Planes
- Camera and Swept-Plane Light Source Calibration
- ***Reconstruction and Visualization using Point Clouds***
- Combining Point Clouds Recovered from Multiple Views

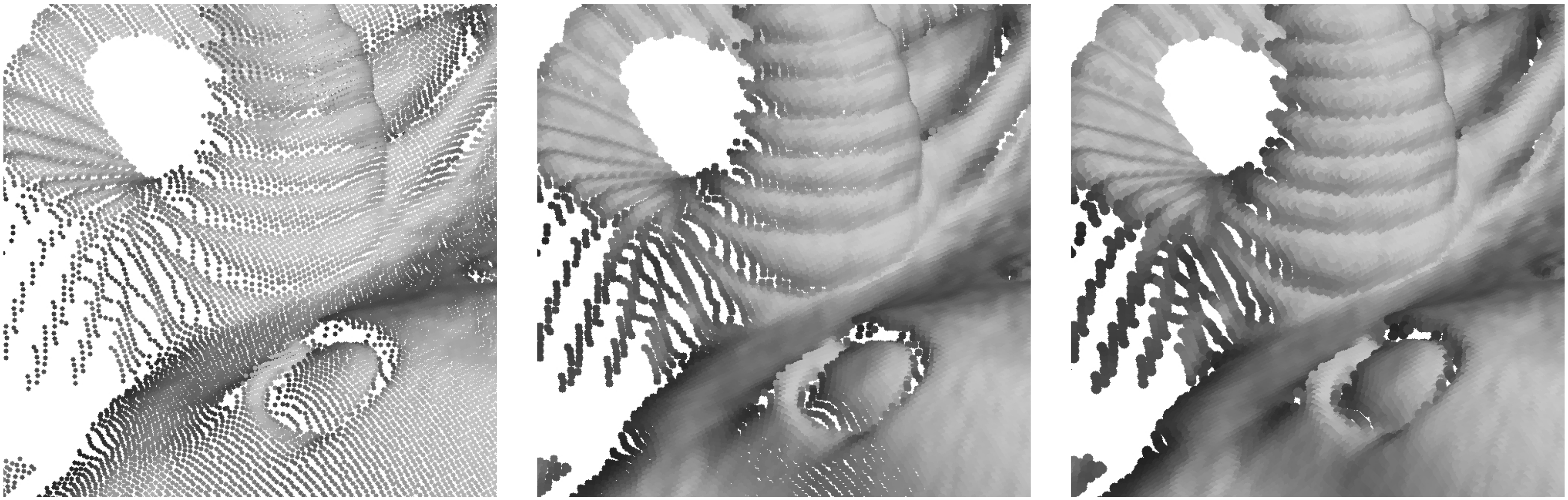
Visualizing Point Clouds:
Point-based Rendering via Splatting



- Swept-plane scanner produces a *colored point cloud*: a set of 3D points
- Problem: how to render a point cloud to make it look like as a continuous surface?
- Splatting: render points as overlapping colored disks
- If normal vectors are measured as well, render points as shaded ellipses

***See the SIGGRAPH 2009 course: Point Based Graphics – State of the Art and Recent Advances by Markus Gross.**

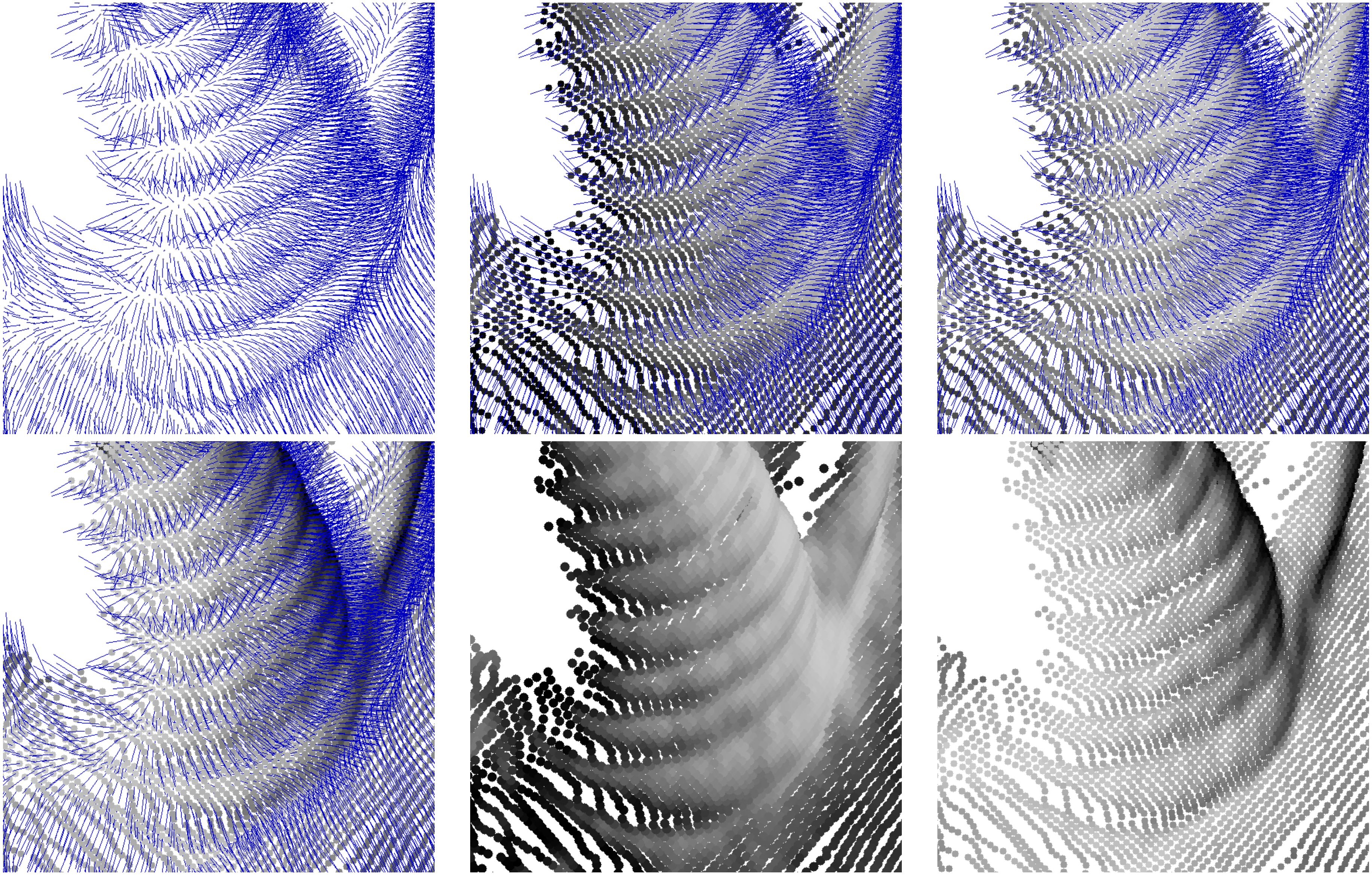
Visualizing Point Clouds:
Point-based Rendering via Splatting



- Swept-plane scanner produces a *colored point cloud*: a set of 3D points
- Problem: how to render a point cloud to make it look like as a continuous surface?
- Splatting: render points as overlapping colored disks
- If normal vectors are measured as well, render points as shaded ellipses

***See the SIGGRAPH 2009 course: Point Based Graphics – State of the Art and Recent Advances by Markus Gross.**

Visualizing Point Clouds:
Splatting with normal vectors and colors



Visualizing Point Clouds:
File Formats

- No standard file format to store point clouds
- Point = (x,y,z) plus (R,G,B) and/or (Nx,Ny,Nz)
- It is easy to create an ad-hoc file format
- Scene graph based file format: VRML
- International standard: ISO/IEC 14772-1:97 VRML’97
- PointSet node includes coordinates (x,y,z) and optional colors (R,G,B), but no normals

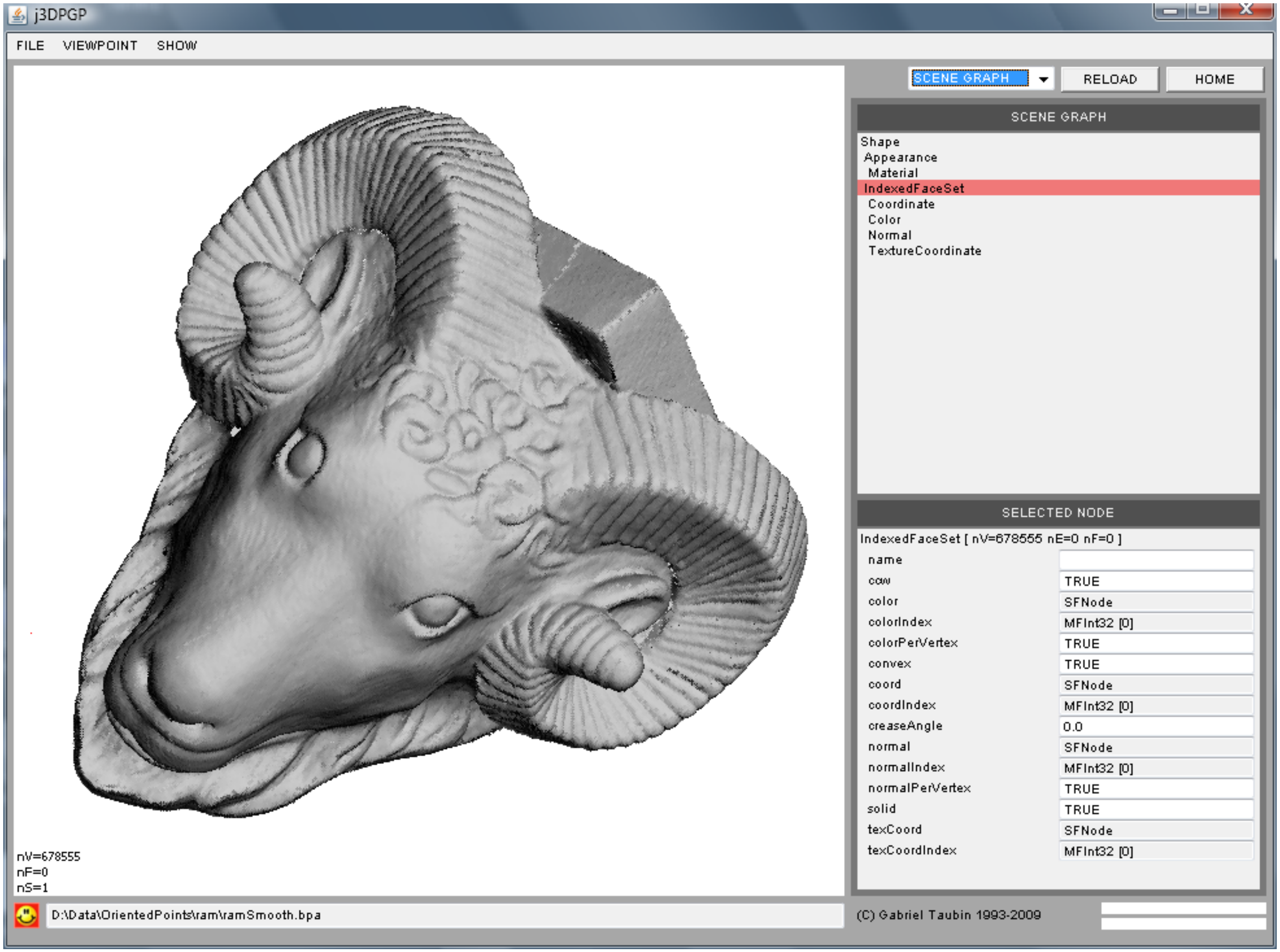
```
PointSet {
 coord Coordinate {
  point [
    0 -1  2,     1  0  0,
-2  3 -1
  ]
 }
 color Color {
  color [
   1 0 0,   0 1 0,    1 1 0
  ]
 }
}
```

Visualizing Point Clouds:
File Formats

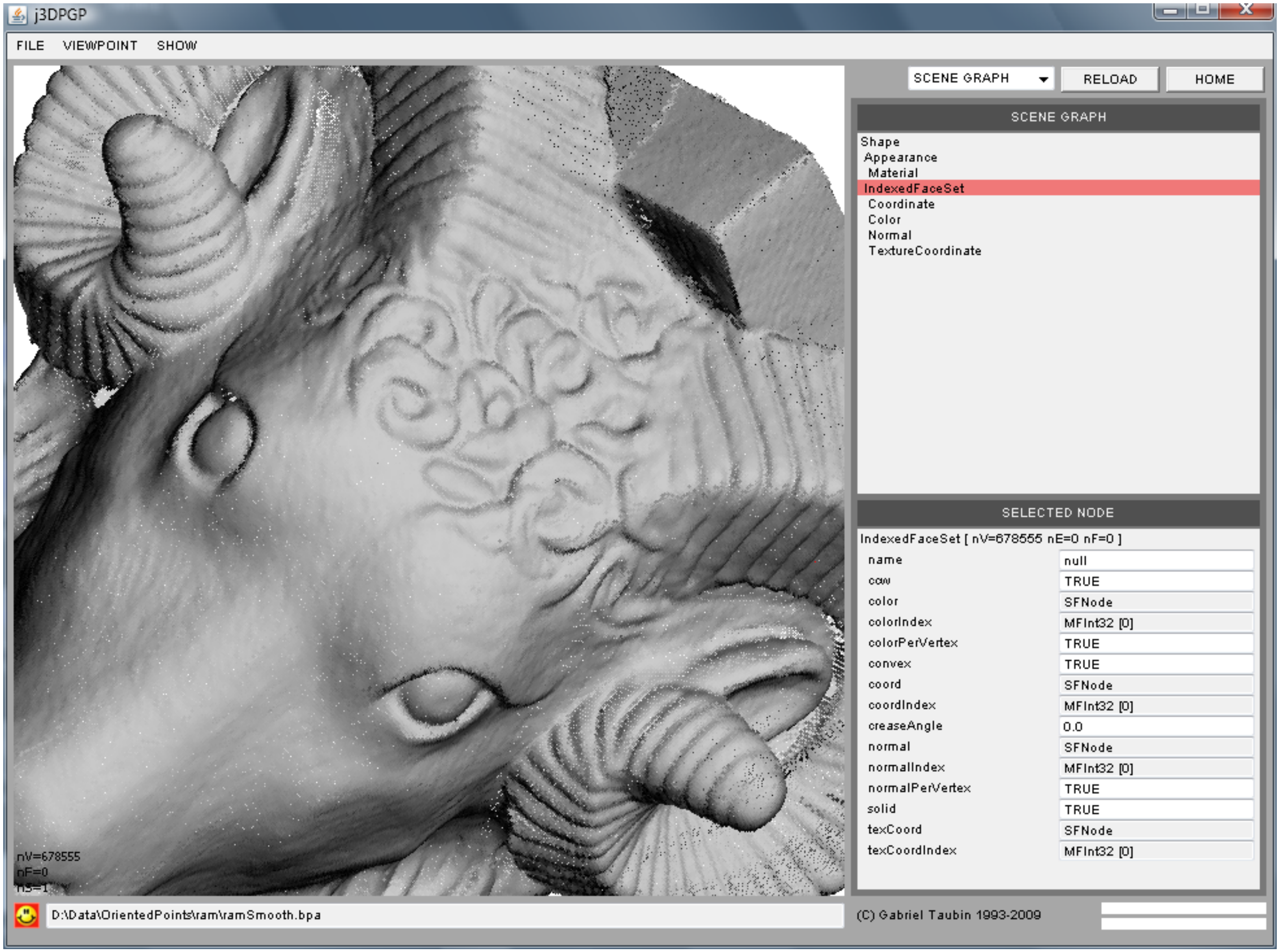
- IndexedFaceSet node designed to store a polygon mesh can be used to store point clouds with optional colors and/or normal vectors
- Store point coordinates as vertices
- Store point colors as colors per vertex
- Store point normal vectors as normals per vertex
- Degenerate polygon mesh with no faces is valid VRML syntax

```
IndexedFaceSet {
 coord Coordinate {
  point [
    0 -1  2,
    1  0  0,
   -2  3 -1
  ]
 }
 colorPerVertex TRUE
 color Color {
  color [
   1 0 0,
   0 1 0,
   1 1 0
  ]
 }
 normalPerVertex TRUE
 normal Normal {
  vector [
   1 0 0,
   0 1 0,
   0 0 1
  ]
 }
}
```

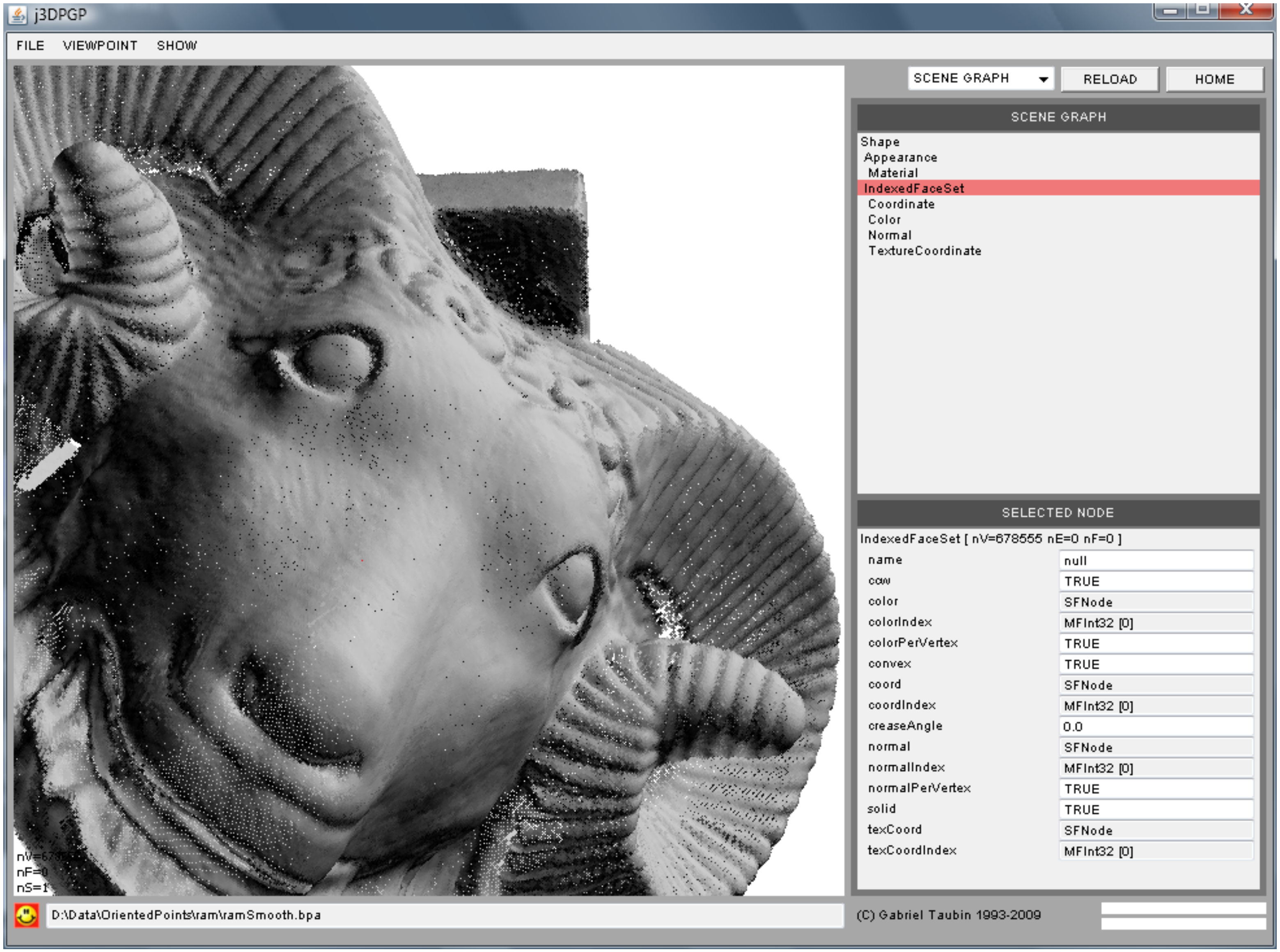
Visualizing Point Clouds: BYO3D Java Viewer



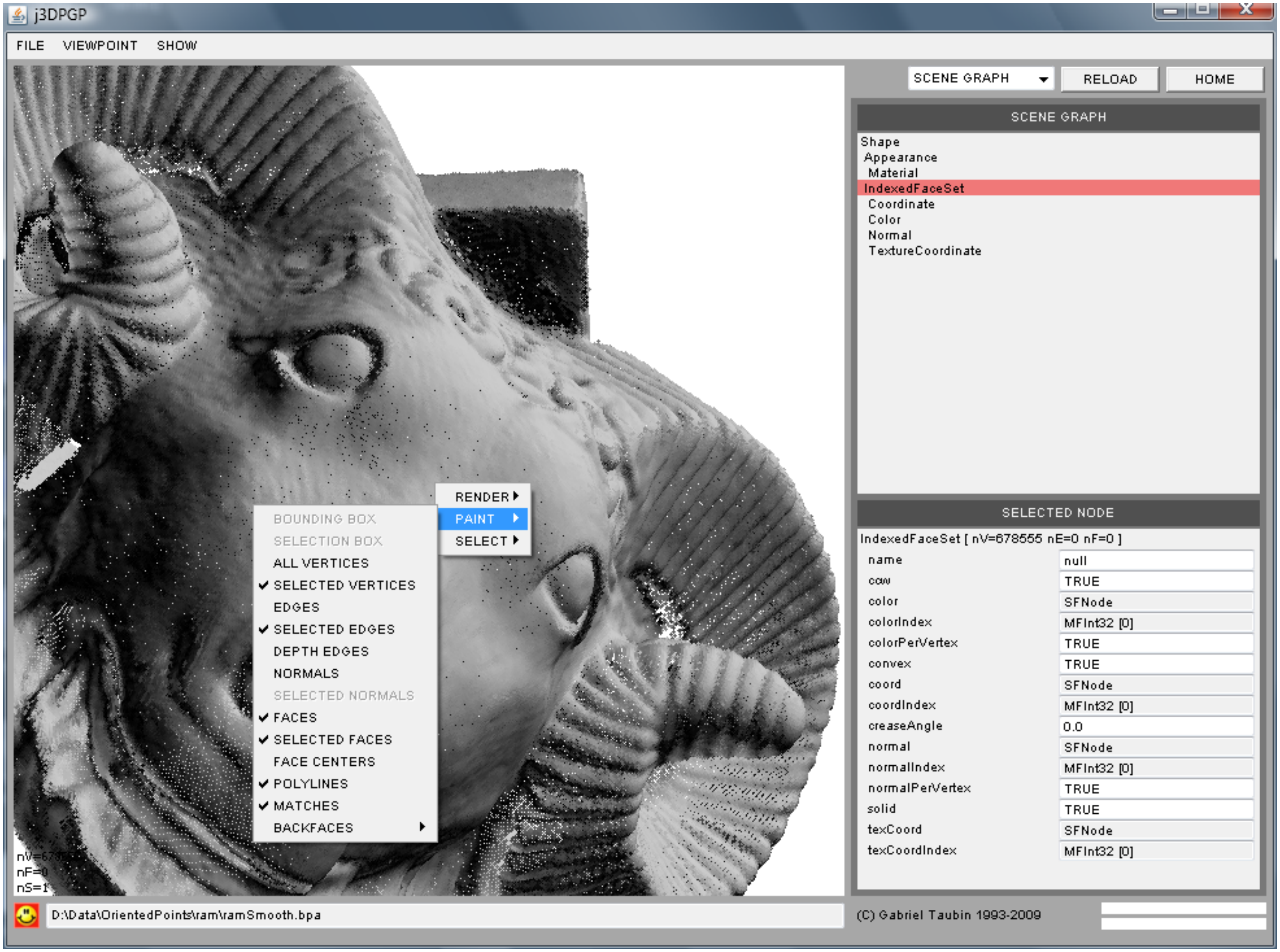
Visualizing Point Clouds:
BYO3D Java Viewer



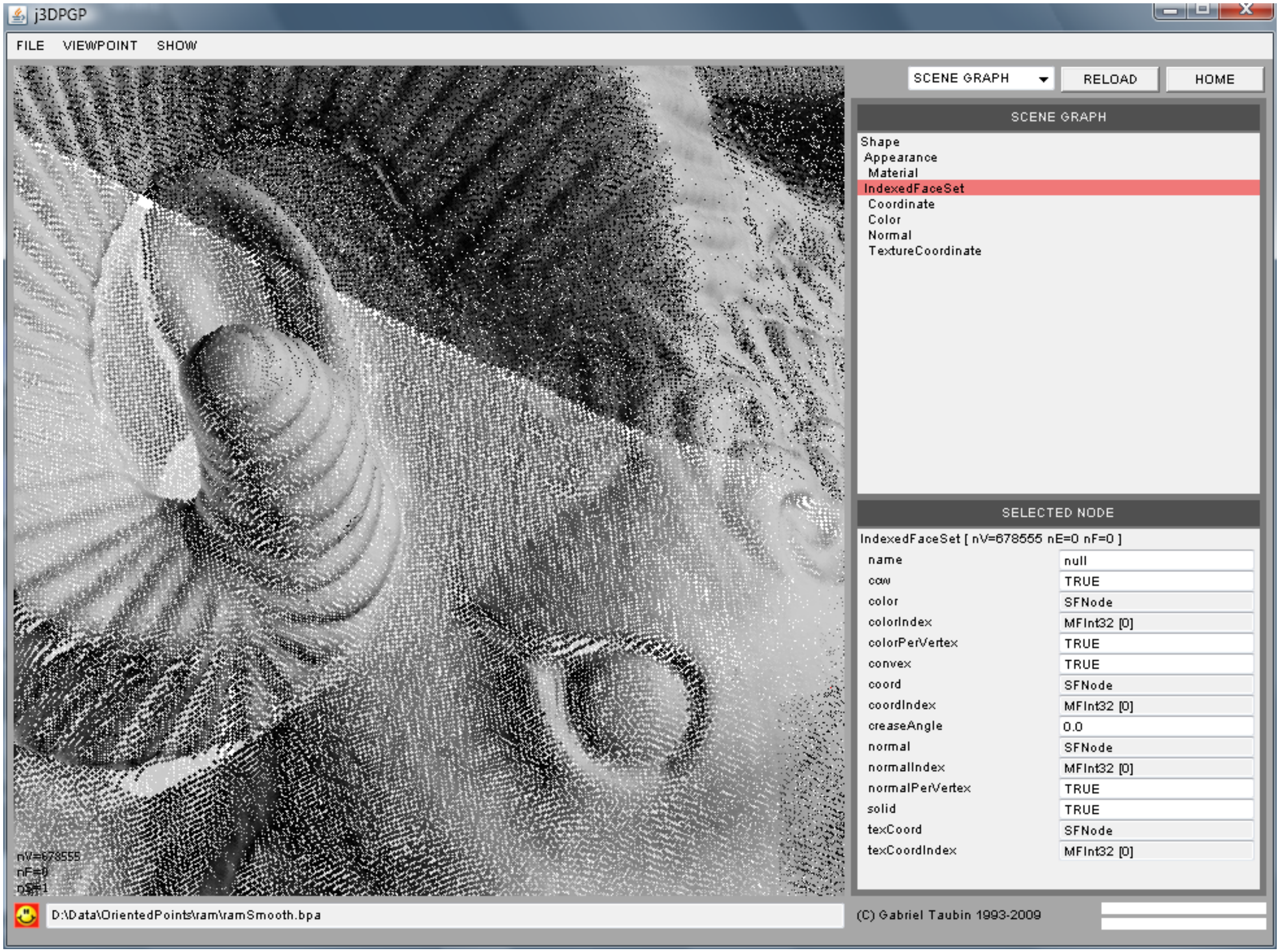
Visualizing Point Clouds:
BYO3D Java Viewer



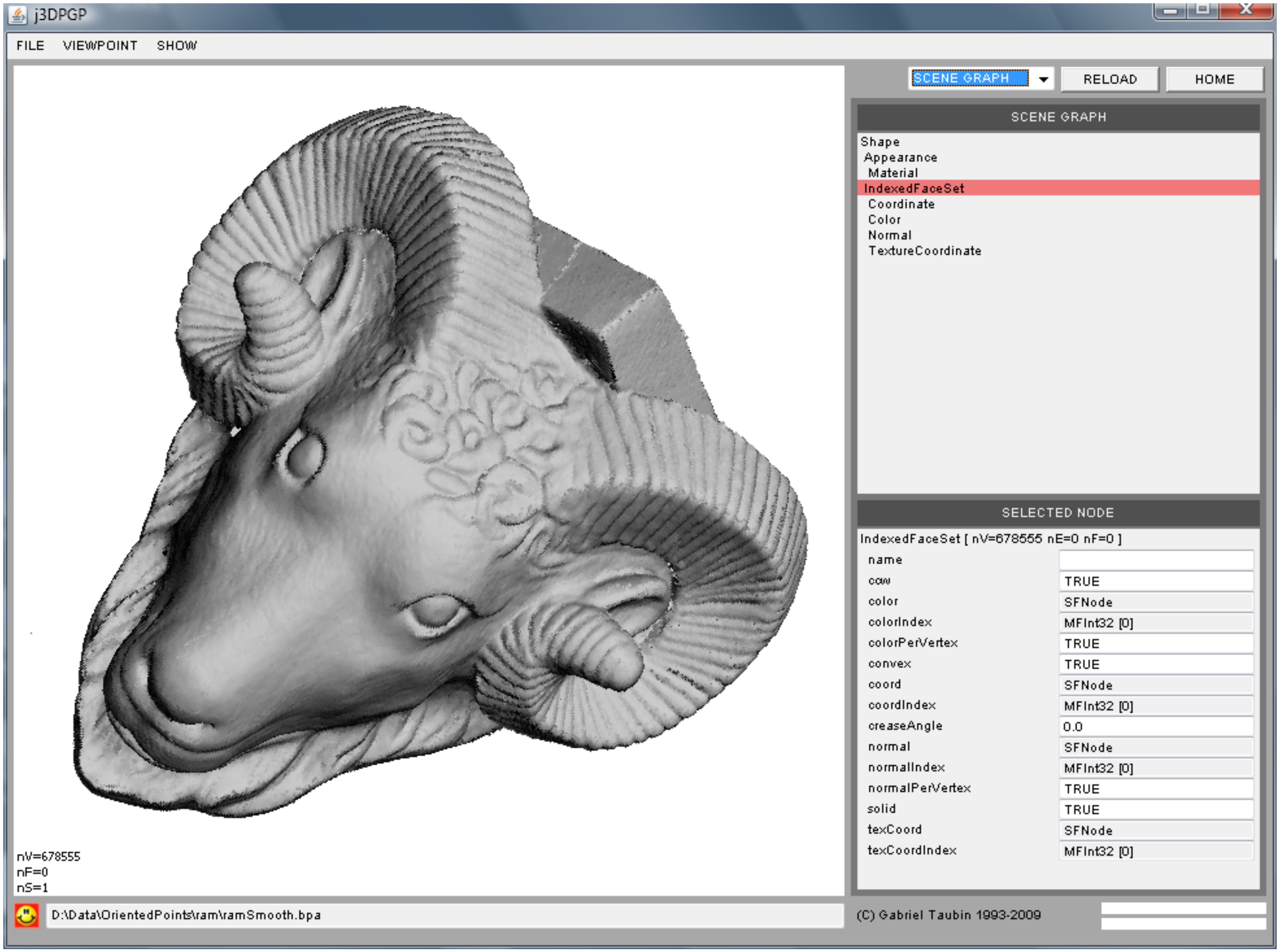
Visualizing Point Clouds:
BYO3D Java Viewer



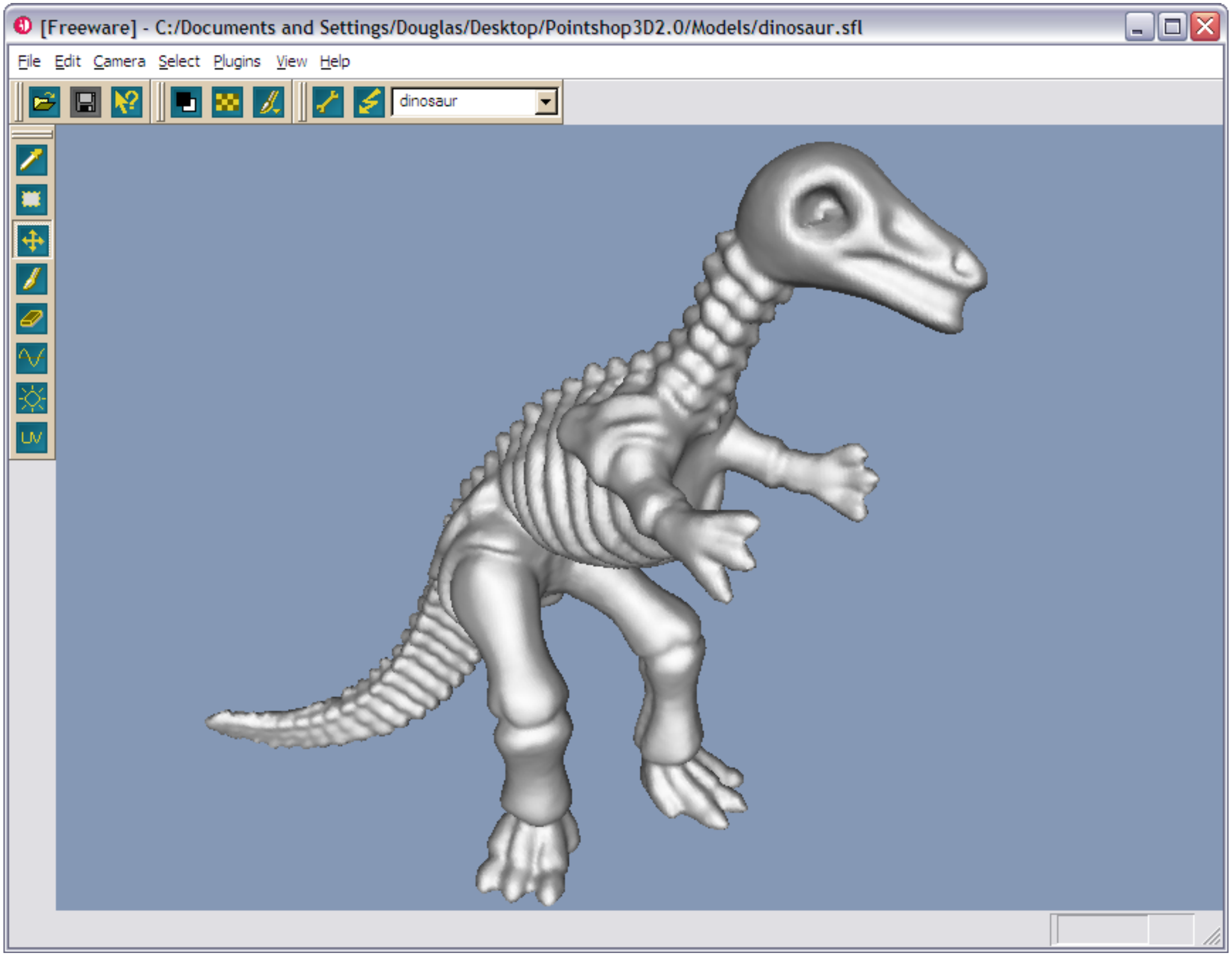
Visualizing Point Clouds: BYO3D Java Viewer



Visualizing Point Clouds:
BYO3D Java Viewer

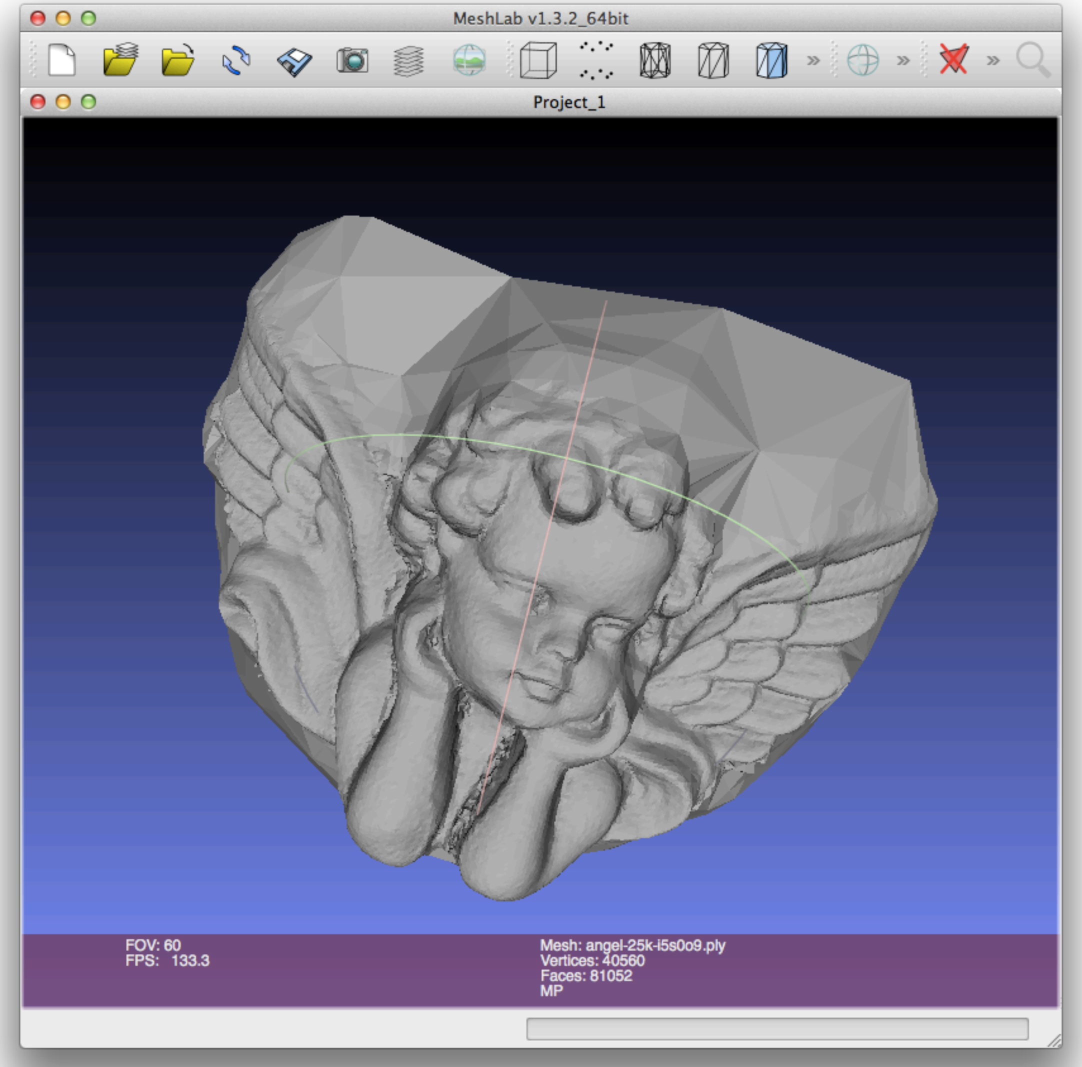


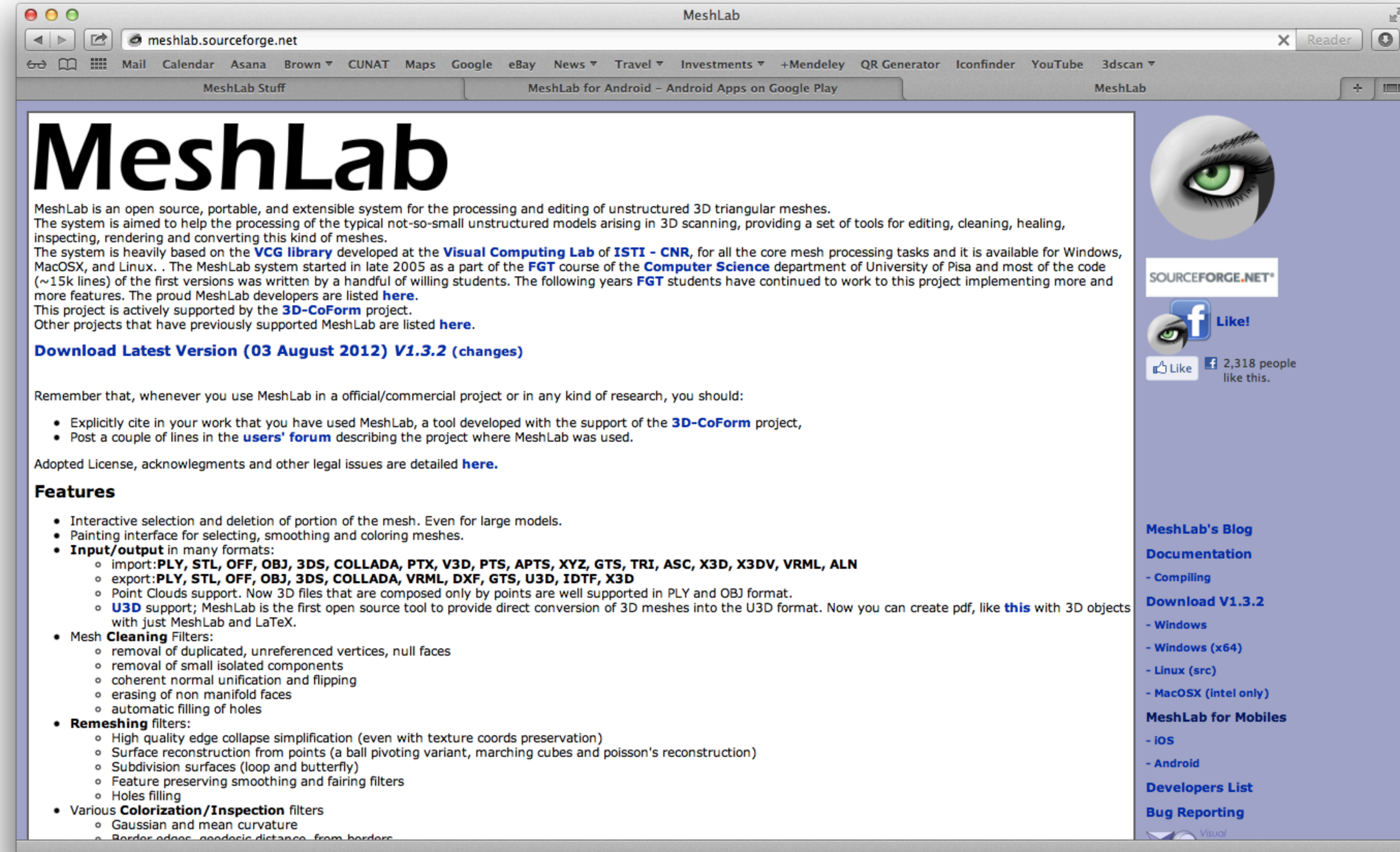
Visualizing Point Clouds:
Pointshop 3D [Zwicker et al. 2002]



M. Zwicker, M. Pauly, O. Knoll, M. Gross. Pointshop 3D: An Interactive System for Point-Based Surface Editing. ACM SIGGRAPH, 2002

Visualizing Point Clouds: MeshLab





MeshLab

MeshLab is an open source, portable, and extensible system for the processing and editing of unstructured 3D triangular meshes.
The system is aimed to help the processing of the typical not-so-small unstructured models arising in 3D scanning, providing a set of tools for editing, cleaning, healing, inspecting, rendering and converting this kind of meshes.
The system is heavily based on the **VCG library** developed at the **Visual Computing Lab** of **ISTI - CNR**, for all the core mesh processing tasks and it is available for Windows, MacOSX, and Linux. . The MeshLab system started in late 2005 as a part of the **FGT** course of the **Computer Science** department of University of Pisa and most of the code (~15k lines) of the first versions was written by a handful of willing students. The following years **FGT** students have continued to work to this project implementing more and more features. The proud MeshLab developers are listed **here**.
This project is actively supported by the **3D-CoForm** project.
Other projects that have previously supported MeshLab are listed **here**.

Download Latest Version (03 August 2012) *V1.3.2* (changes)

Remember that, whenever you use MeshLab in a official/commercial project or in any kind of research, you should:

- Explicitly cite in your work that you have used MeshLab, a tool developed with the support of the **3D-CoForm** project,
- Post a couple of lines in the **users' forum** describing the project where MeshLab was used.

Adopted License, acknowlegments and other legal issues are detailed **here.**

Features

- Interactive selection and deletion of portion of the mesh. Even for large models.
- Painting interface for selecting, smoothing and coloring meshes.
- **Input/output** in many formats:
  - import:**PLY, STL, OFF, OBJ, 3DS, COLLADA, PTX, V3D, PTS, APTS, XYZ, GTS, TRI, ASC, X3D, X3DV, VRML, ALN**
  - export:**PLY, STL, OFF, OBJ, 3DS, COLLADA, VRML, DXF, GTS, U3D, IDTF, X3D**
  - Point Clouds support. Now 3D files that are composed only by points are well supported in PLY and OBJ format.
  - **U3D** support; MeshLab is the first open source tool to provide direct conversion of 3D meshes into the U3D format. Now you can create pdf, like **this** with 3D objects with just MeshLab and LaTeX.
- Mesh **Cleaning** Filters:
  - removal of duplicated, unreferenced vertices, null faces
  - removal of small isolated components
  - coherent normal unification and flipping
  - erasing of non manifold faces
  - automatic filling of holes
- **Remeshing** filters:
  - High quality edge collapse simplification (even with texture coords preservation)
  - Surface reconstruction from points (a ball pivoting variant, marching cubes and poisson's reconstruction)
  - Subdivision surfaces (loop and butterfly)
  - Feature preserving smoothing and fairing filters
  - Holes filling
- Various **Colorization/Inspection** filters
  - Gaussian and mean curvature

SOURCEFORGE.NET

Like!

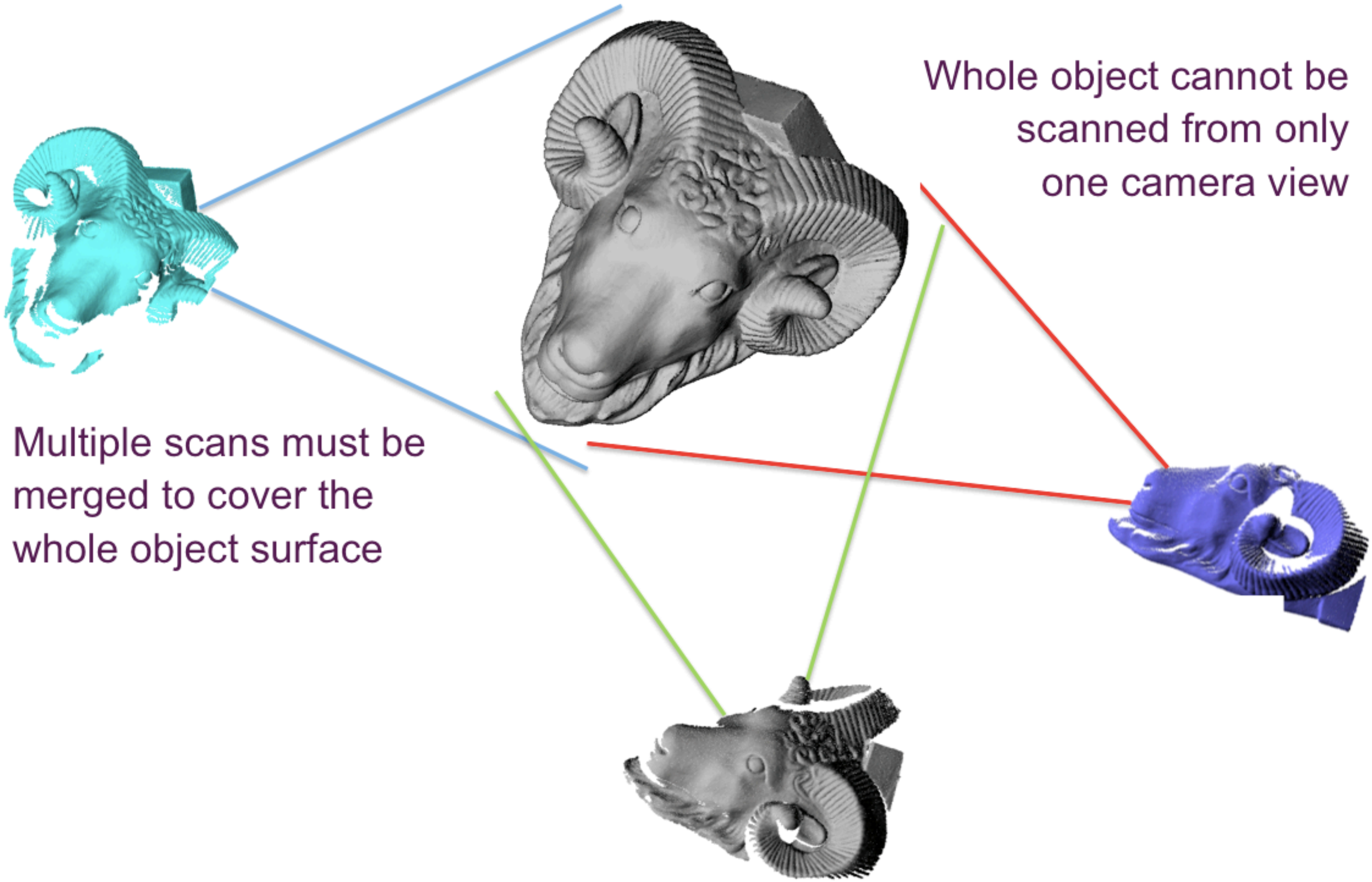
Like 2,318 people like this.

MeshLab's Blog
Documentation
- Compiling
Download V1.3.2
- Windows
- Windows (x64)
- Linux (src)
- MacOSX (intel only)
MeshLab for Mobiles
- iOS
- Android
Developers List
Bug Reporting

Course Schedule

- Introduction
- The Mathematics of 3D Triangulation
- 3D Scanning with Swept-Planes
- Camera and Swept-Plane Light Source Calibration
- Reconstruction and Visualization using Point Clouds
- ***Combining Point Clouds Recovered from Multiple Views***

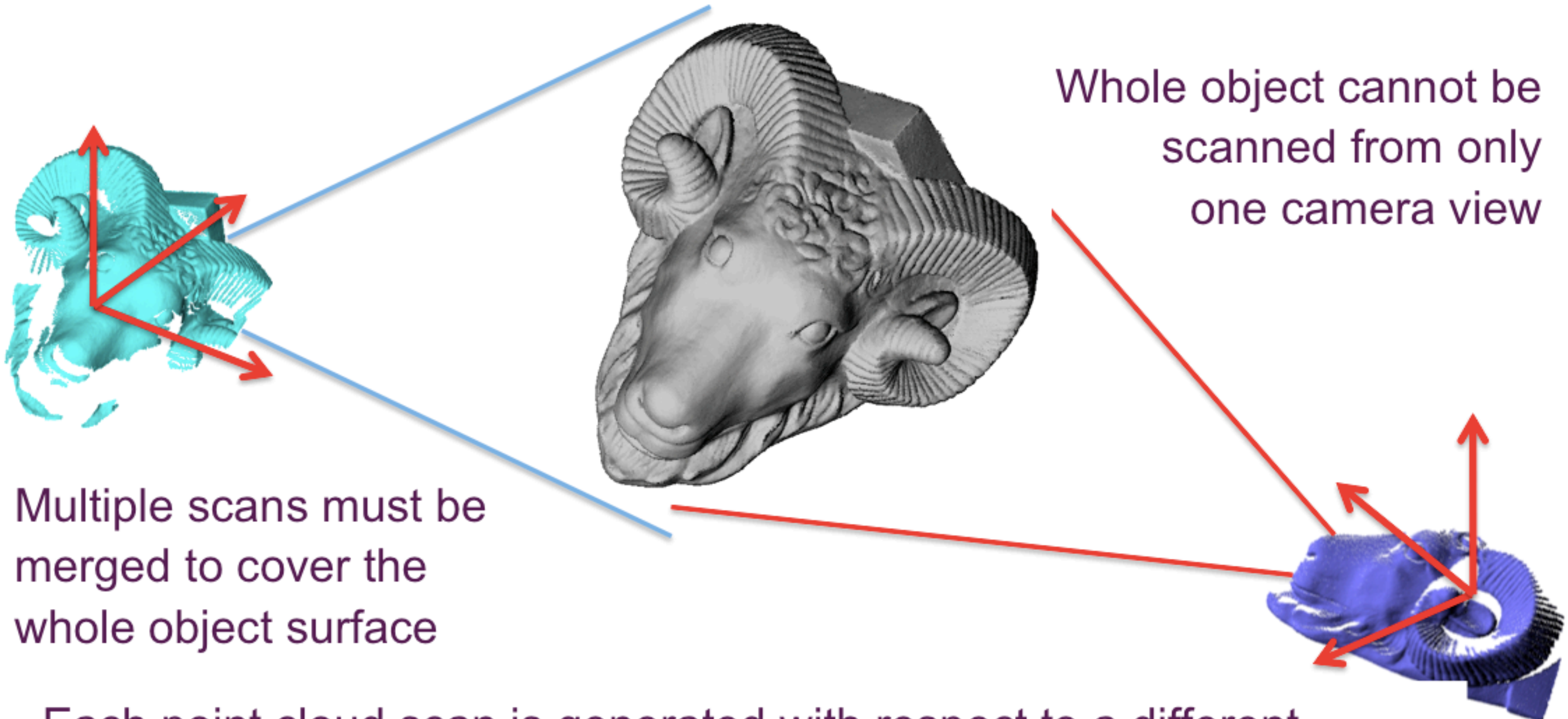
Merging Multiple Point Cloud Scans



Whole object cannot be scanned from only one camera view

Multiple scans must be merged to cover the whole object surface

Merging Multiple Point Cloud Scans

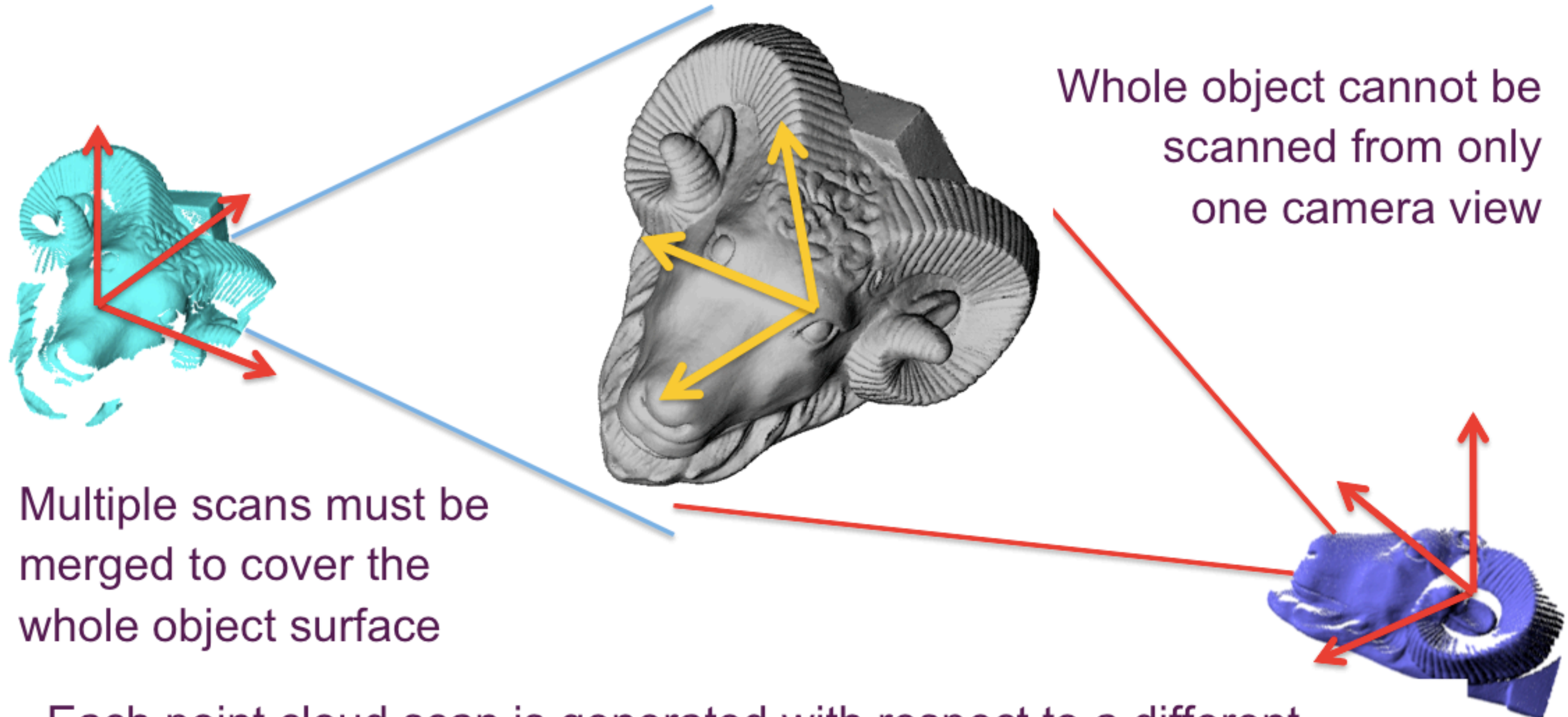


Whole object cannot be scanned from only one camera view

Multiple scans must be merged to cover the whole object surface

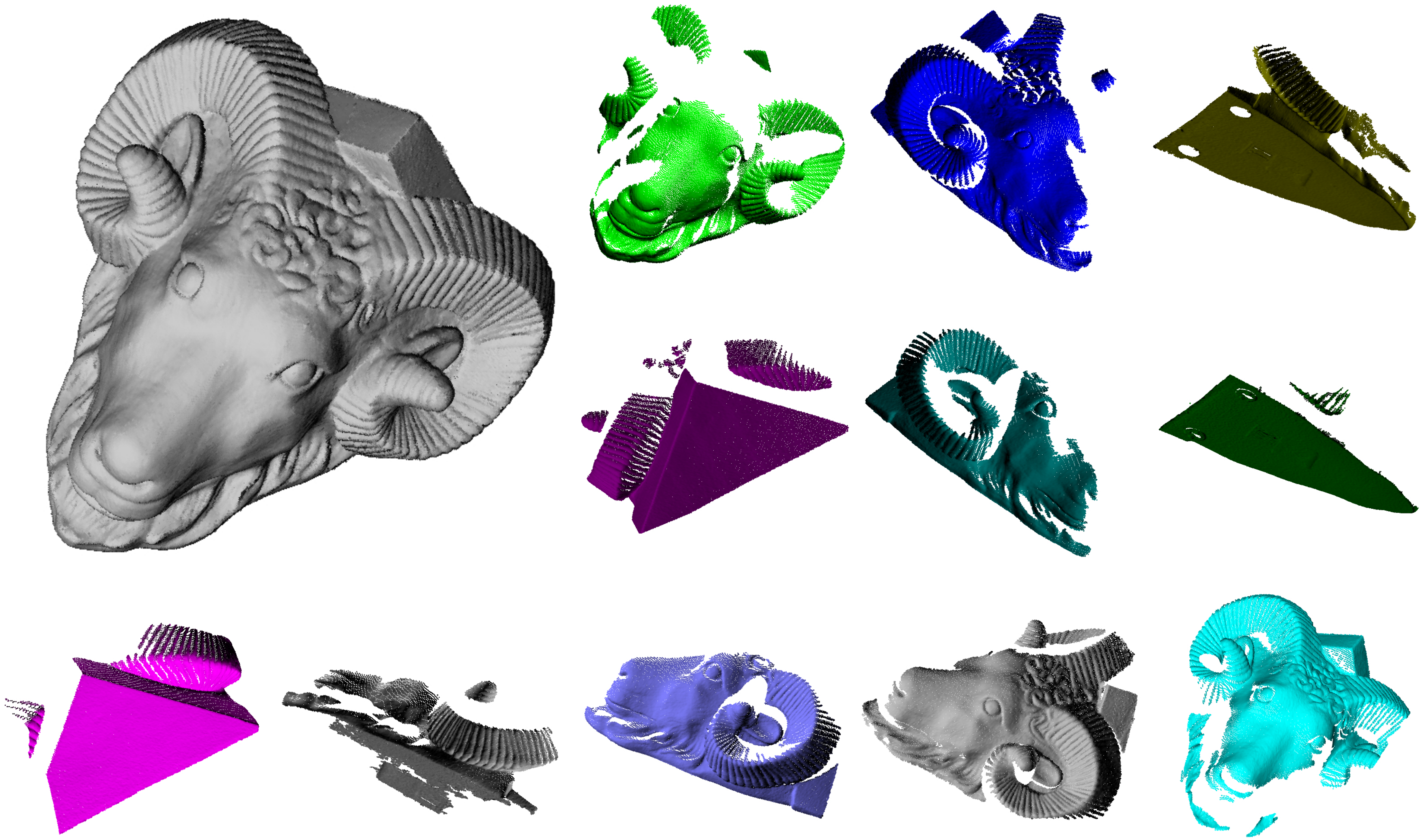
- Each point cloud scan is generated with respect to a different camera coordinate system

Merging Multiple Point Cloud Scans



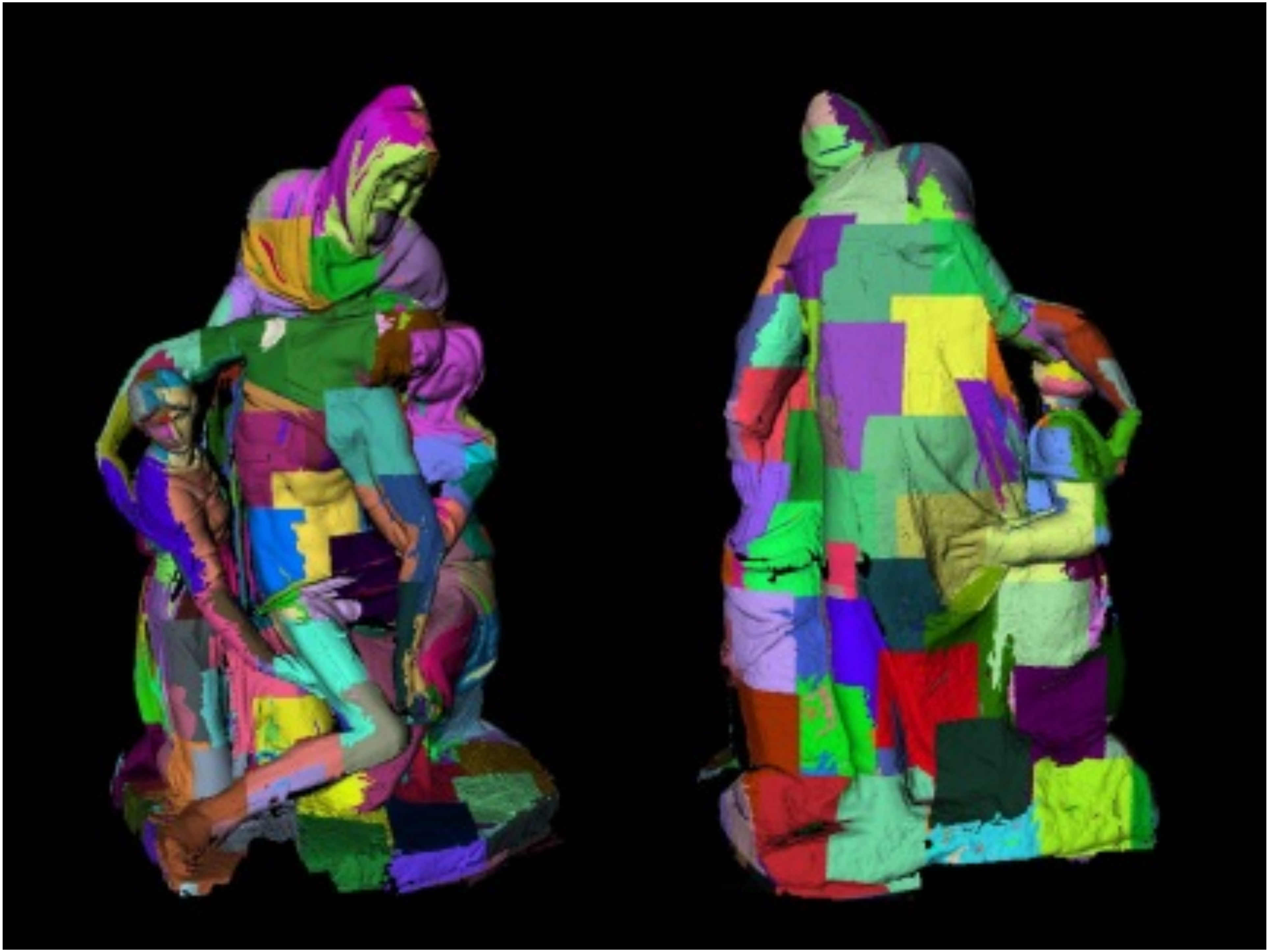
- Each point cloud scan is generated with respect to a different camera coordinate system
- Relative position and orientation of each scan with respect to a global coordinate system must be determined to produce a single merged point cloud

Merging Point Cloud Scans



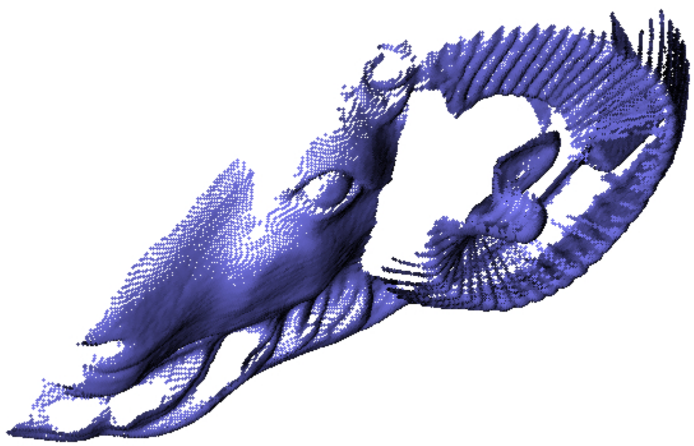
Complex Models May Require 100s of Scans

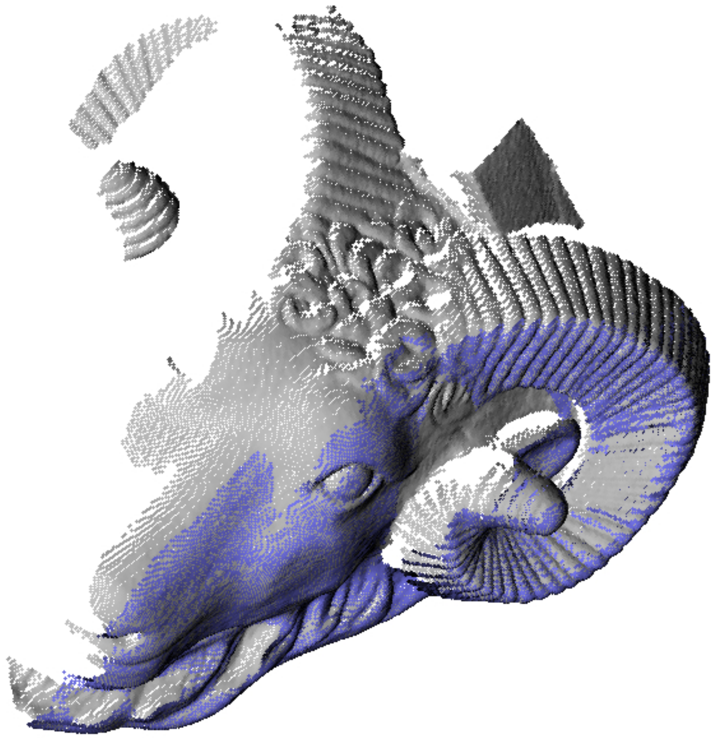
http://www.research.ibm.com/pieta



Merging Point Cloud Scans

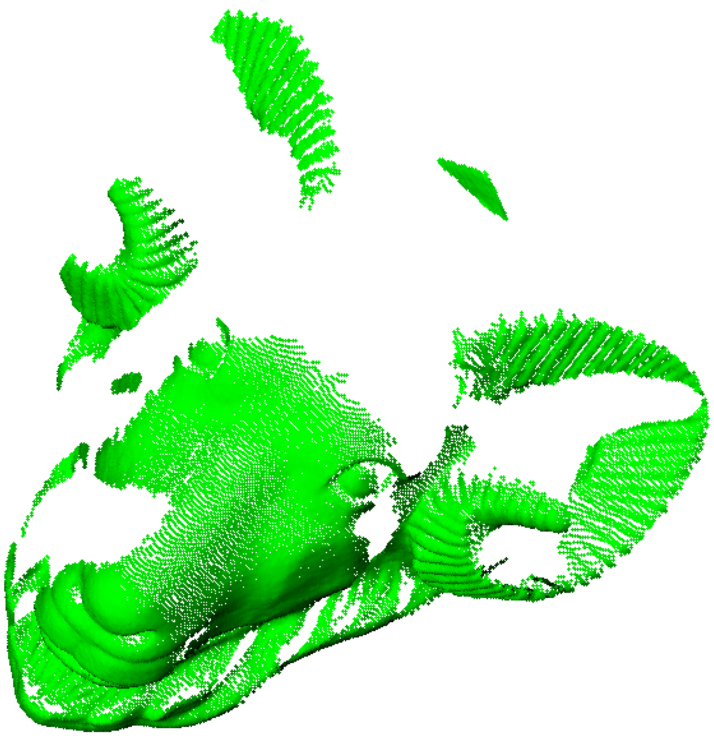
- Incremental registration and merging
- Followed by global relaxation to remove accumulated errors

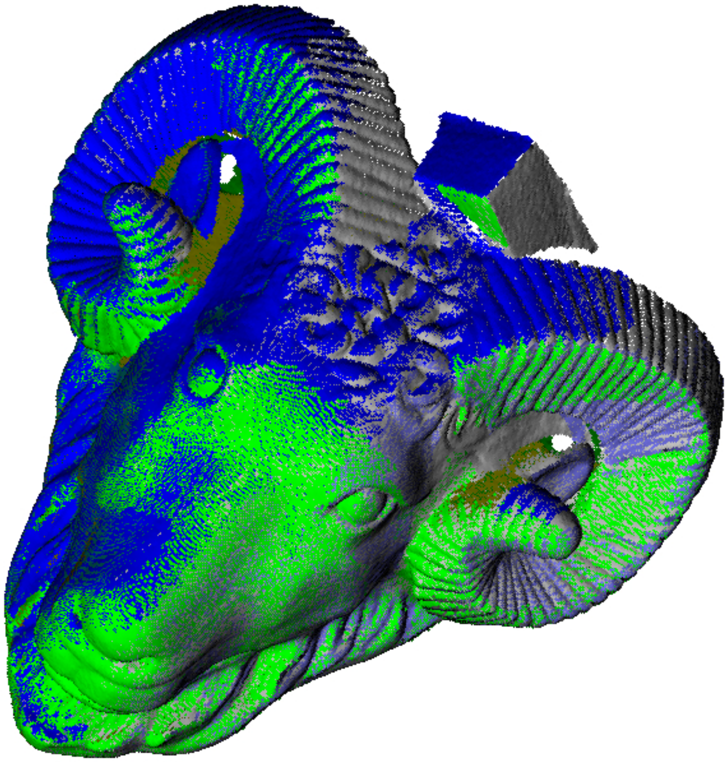




Merging Point Cloud Scans

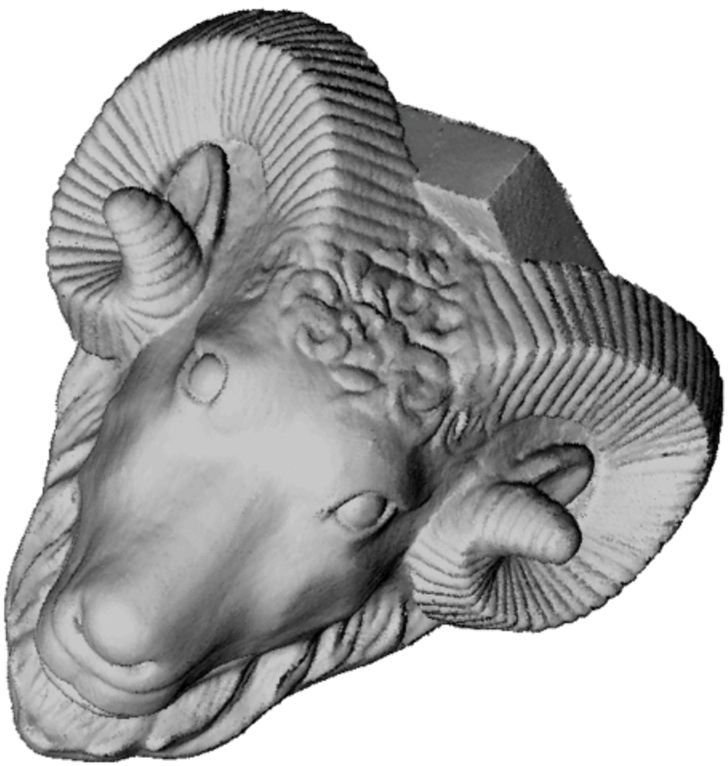
- Incremental registration and merging
- Followed by global relaxation to remove accumulated errors

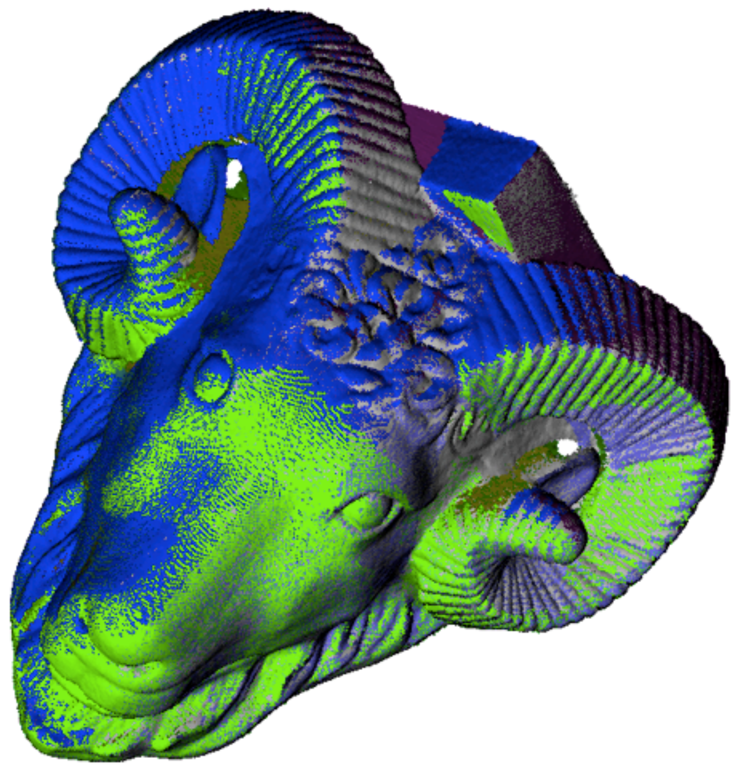




Merging Point Cloud Scans

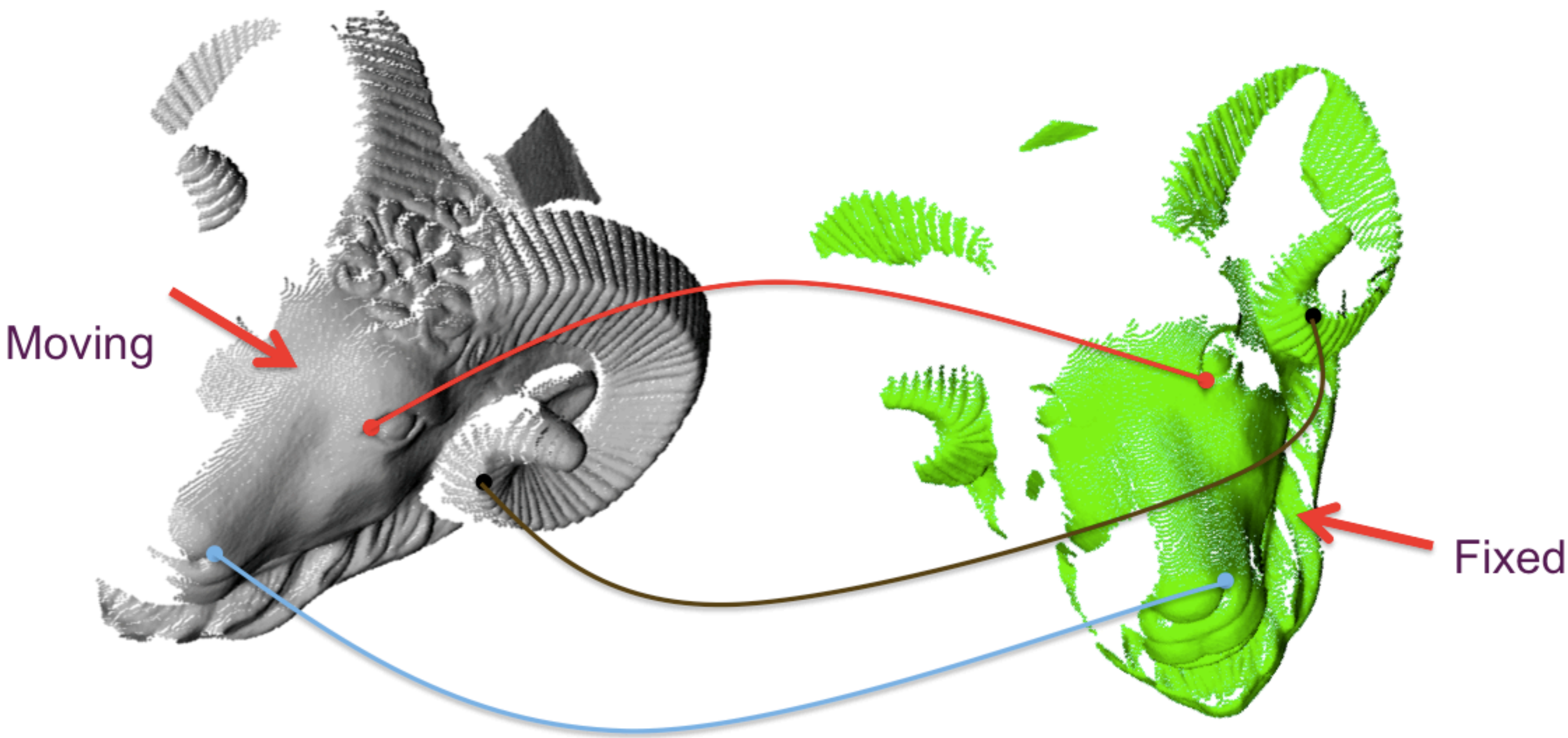
- Incremental registration and merging
- Followed by global relaxation to remove accumulated errors





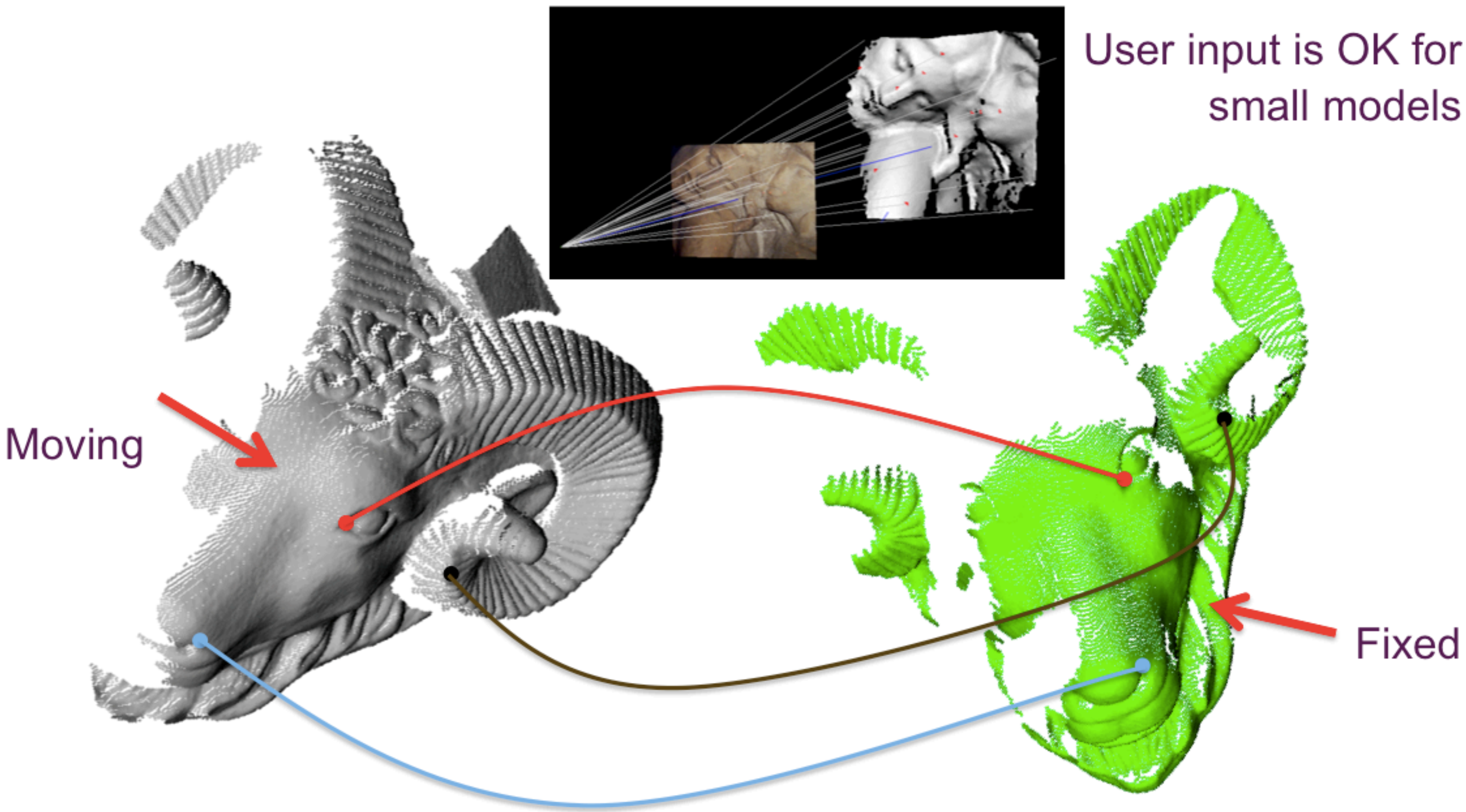
Merging Two Point Cloud Scans

- Select at least 3 pairs of corresponding points, but ideally N>>3



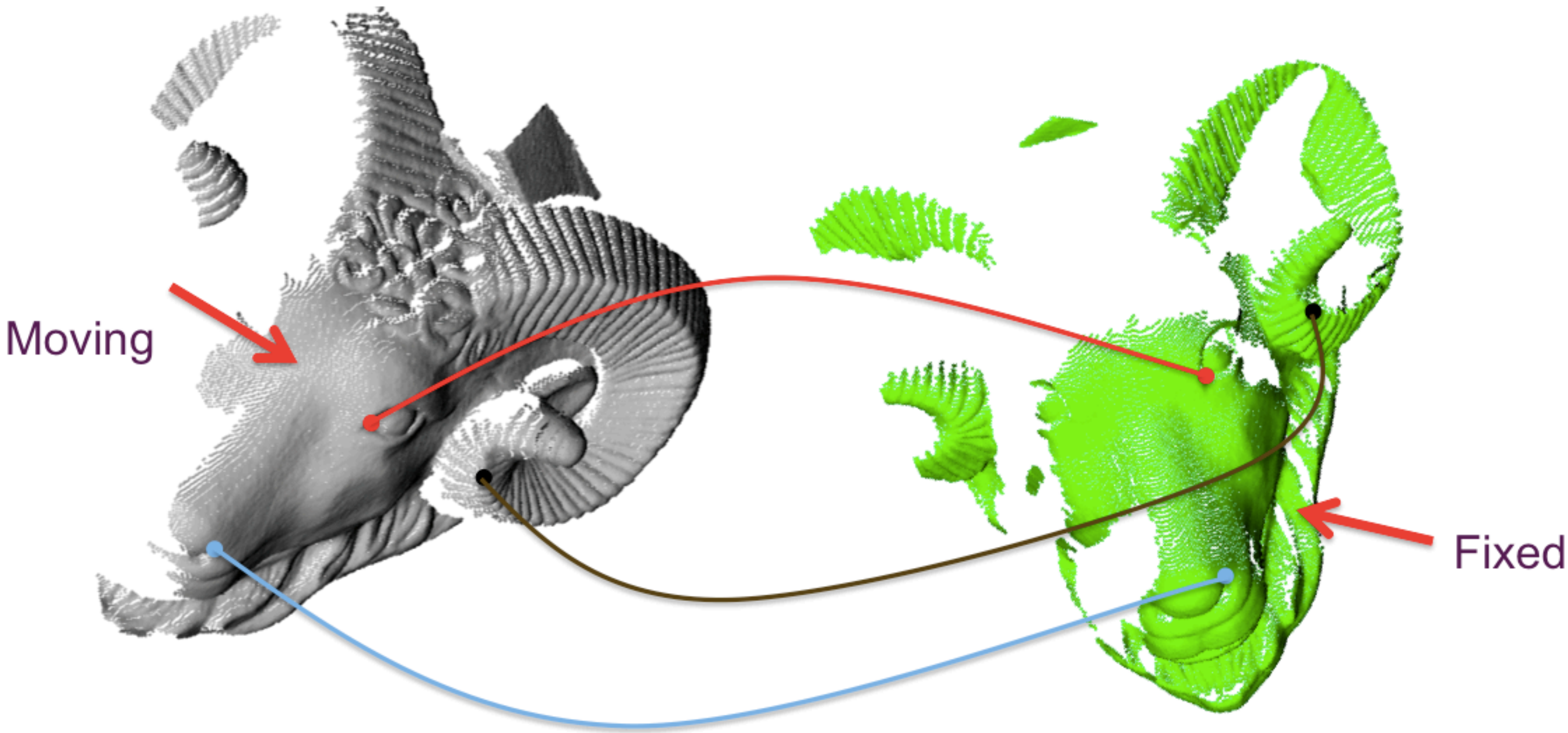
Merging Two Point Cloud Scans

- Select at least 3 pairs of corresponding points, but ideally N>>3



Merging Two Point Cloud Scans

- Select at least 3 pairs of corresponding points, but ideally N>>3
- Solve in close form for the matching rigid body transformation
- Refine solution using the Iterative Closest Point Algorithm (ICP)



P. Besl, N.D. McKey, A method for Registration of 3D Shapes.
IEEE Transactions on PAMI, 1992

Computing the Matching Transformation

- Given N pairs of corresponding 3D points $(p_1,q_1),\ldots,(p_n,q_n)$ we are looking for a rotation matrix R and a translation vector T so that

$$Rp_j + T = q_j \quad j = 1,\ldots,n$$

- In general, solution does not exists: solve in the Least-Squares sense
- Now we are looking for the minimizer of the quadratic energy function

$$E(R,T) = \frac{1}{n}\sum_{j=1}^{n}\left\| Rp_j + T - q_j \right\|^2$$

- This problem has a closed form solution

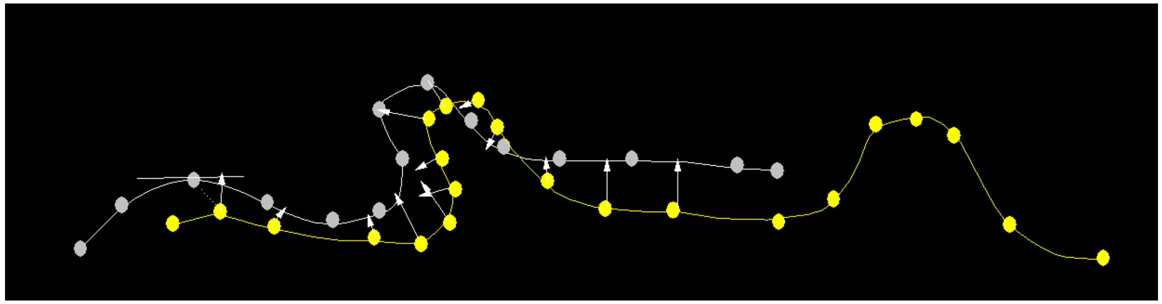
$$R = V^t U, \; T = \overline{q} - R\,\overline{p}$$

- Where

$$\overline{p} = \frac{1}{n}\sum_{j=1}^{n} p_j \quad \overline{q} = \frac{1}{n}\sum_{j=1}^{n} q_j \quad M = \frac{1}{n}\sum_{j=1}^{n}(p_j - \overline{p})(q_j - \overline{q})^t$$

- And $M = U\Delta V^t$ is the Singular Value Decomposition (SVD) of M

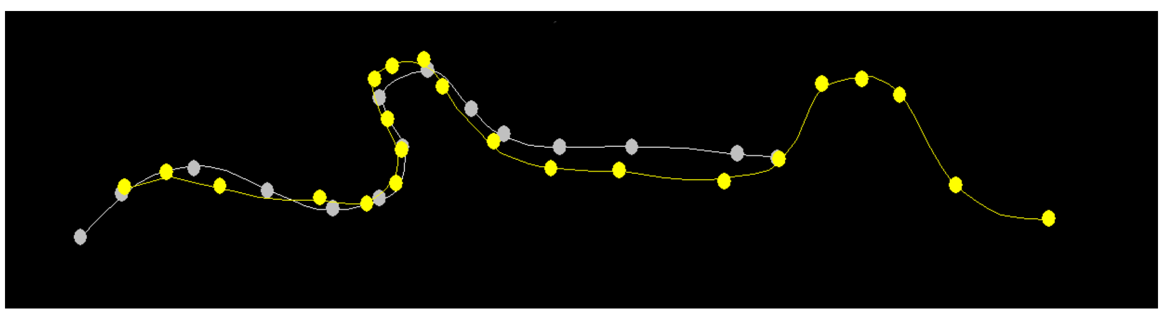
Iterative Closest Point Algorithm (ICP)



1. Automatically select N points $p_1, \ldots, p_n$
2. Find closest corresponding points $q_1, \ldots, q_n$
3. Solve in close form for the matching rigid body transformation which minimizes the energy function

$$E(R,T) = \frac{1}{n}\sum_{j=1}^{n}\left\| Rp_j + T - q_j \right\|^2$$

4. Repeat 1-3 while until convergence



Finding Closest Points

- Problem: find the point of the set $D = \{p_1, \ldots, p_n\}$ closest to the point q
- Naïve algorithm: sequential search O(N)
- Too expensive if the same computation must be performed for many points $q_1, \ldots, q_n$
- Efficient algorithm requires space partition data structure
  Quadtree/Octree, BSP tree