

Mesh Fitting To Points

Michael Paone

Andrew Yuen

CMPT 469

April 18, 2005

Presentation/Demo Outline

- Introduction to Mesh Fitting to Points
 - General Approach and Implementation Details
 - Demo
 - Limitations and Possible Future Improvements
 - Unexplored Questions
 - Error Plots
 - References
-

What is Mesh Fitting to Points?

■ Input

■ Point Cloud

- An unorganized collection of points

■ Connectivity Mesh

- A connected triangle mesh
- We don't know or ignore the geometry of the vertices
- Essentially a planar graph

■ Anchor points

- A subset of the vertices in the Connectivity Mesh
- Each anchor point is “fixed” to a corresponding point in the Point Cloud
- Fixing can be a “soft” constraint

What is Mesh Fitting to Points?

■ Output

- Complete geometry for Connectivity Mesh
 - We would like the new geometry for the connectivity mesh to be defined so that the connectivity mesh fits, as best as possible, the point cloud.
 - Other Desired Attributes for new geometry
 - Fair distribution of vertices
 - Smooth
 - Can be computed efficiently
-

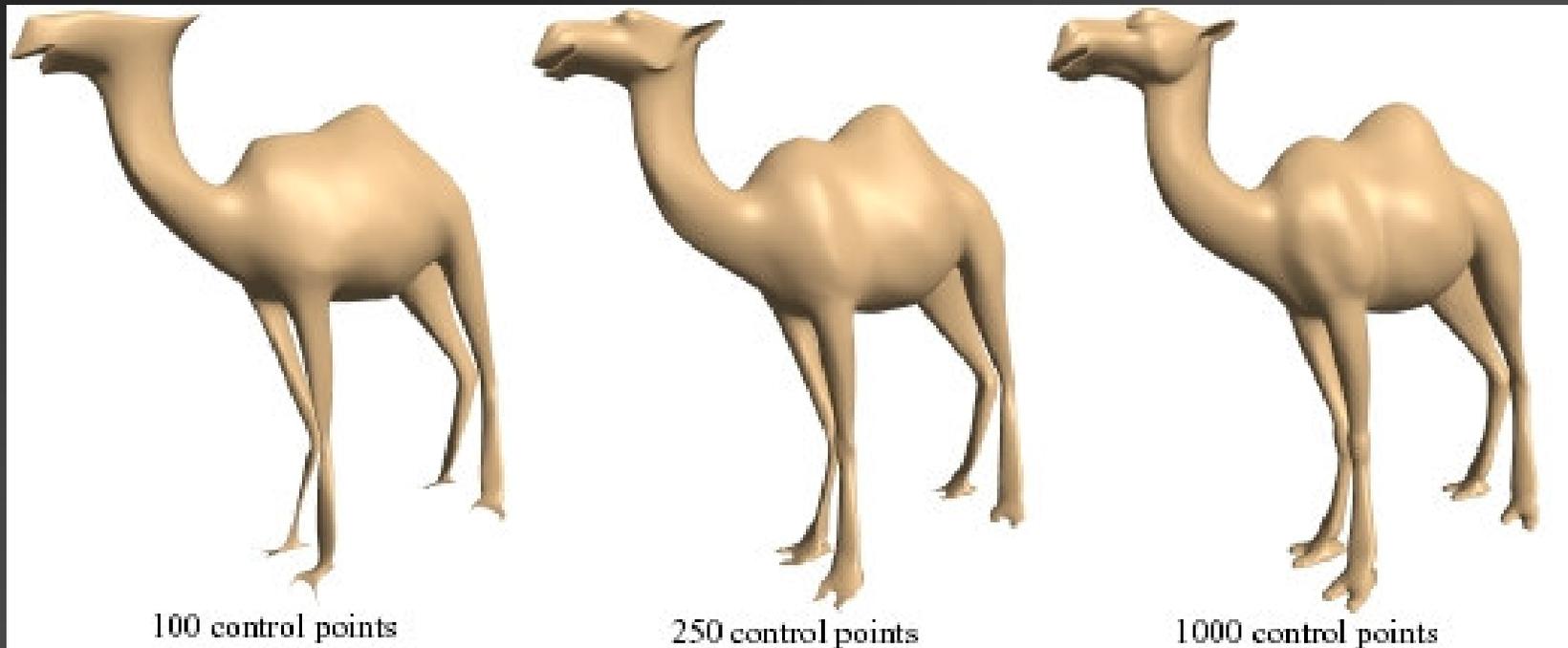
Why Mesh Fitting to Points?

- A typical application
 - Point Cloud obtained from laser ranges
 - Connectivity Mesh
 - Contains a stored generic model for the type of object being scanned
 - Knowledge associated with certain features of the mesh
 - Eg. Distance between two vertices of interest
 - Anchor points
 - stored with connectivity mesh
 - correspondence to point cloud established by a user/expert
 - Complete Geometry allows the knowledge to be transformed for the particular object being scanned

Approach: Overview

- Two stage approach
 1. Use the anchor points to distribute vertices in the connectivity mesh to get the geometry “close”
 - Least Squares Meshes
 2. Make adjustments to vertex geometry to better fit the point cloud locally
 - Project connectivity mesh vertices onto local neighbourhood of point cloud
-

Least Squares Meshes



[Sorkine et al. 2004]

Least Squares Meshes

*“Least Squares Meshes:
meshes with a prescribed connectivity that
approximate a set of control points in a
least-squares sense.”*

Least Squares (LS) Meshes

- Computing an LS mesh from the connectivity mesh with sparse anchor vertices involves minimizing the least squares error in a system of linear constraints:

$$A\mathbf{x} - B = \begin{pmatrix} L \\ F \end{pmatrix} \mathbf{x} - B = 0$$

$$F_{ij} = \begin{cases} 1 & \text{if } j = s_i \\ 0 & \text{otherwise} \end{cases}$$

Where s_i is the i -th anchor point

$$L_{ij} = \begin{cases} 1 & \text{if } i = j \\ \frac{-1}{d_i} & \text{if } (i, j) \in E \text{ and } \deg(i) = d_i \\ 0 & \text{otherwise} \end{cases}$$

$$B_i = \begin{cases} 0 & \text{if } 0 \leq i < n \\ x_{s_{i-n}} & \text{if } n \leq i < n + m \end{cases}$$

LS Meshes

- The first set of n constraints specifies that the component-wise distance between each vertex and the centroid of its neighbours should be zero.

$$L_{ij} = \begin{cases} 1 & \text{if } i = j \\ -1 & \text{if } (i, j) \in E \text{ and } \deg(i) = d_i \\ d_i & \\ 0 & \text{otherwise} \end{cases} \quad L\mathbf{x} = 0$$

LS Meshes

- The second set of m constraints specifies that the component-wise distance between each anchor vertex in the connectivity mesh and its associated point in the point cloud should be zero.

$$F_{ij} = \begin{cases} 1 & \text{if } j = s_i \\ 0 & \text{otherwise} \end{cases}$$

Where s_i is the i -th anchor point

$$F\mathbf{X} = \begin{bmatrix} \mathbf{x}_{s_0} \\ \vdots \\ \mathbf{x}_{s_{m-1}} \end{bmatrix}$$

LS Meshes: Solution

The least squares minimization of the following system:

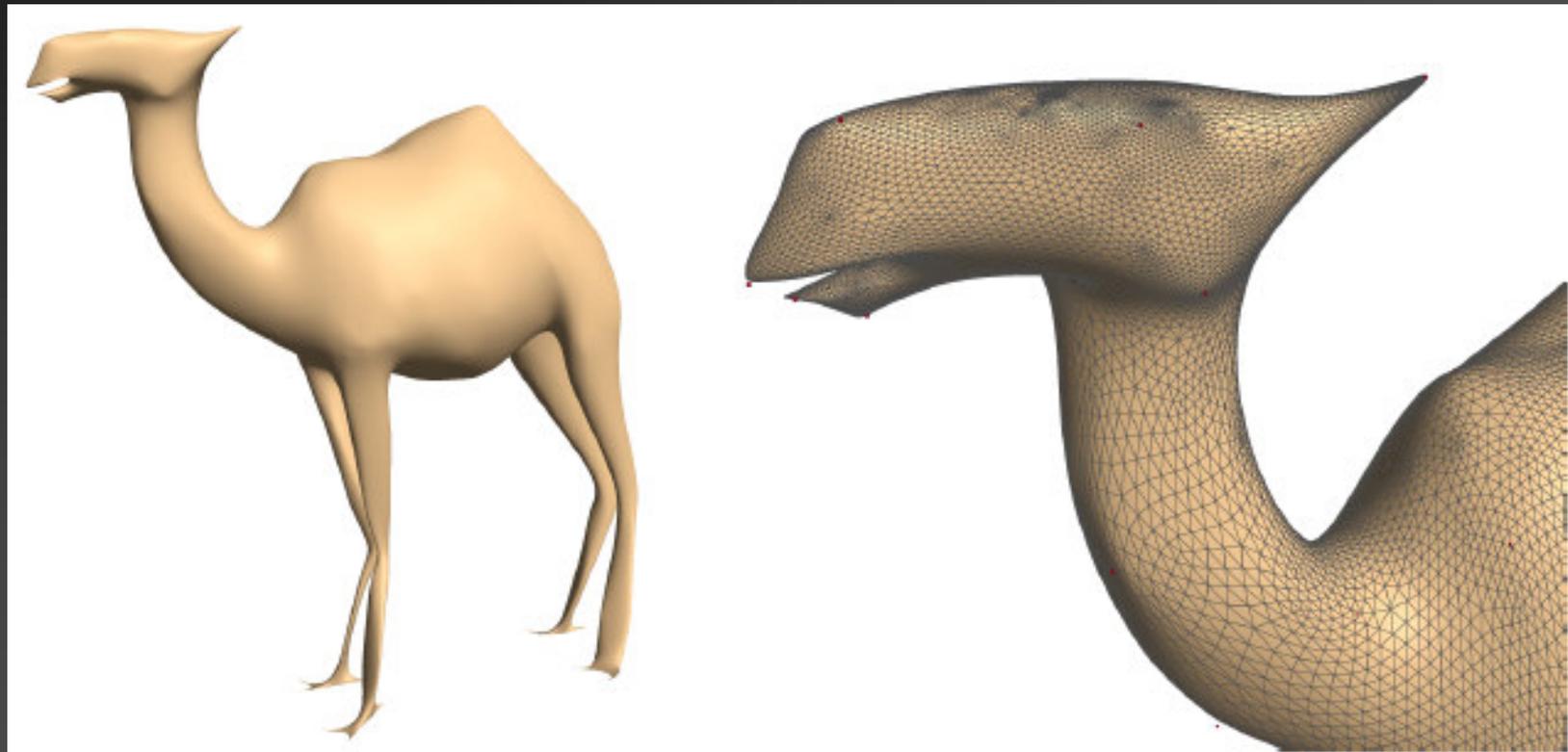
$$A\mathbf{x} - B = \begin{pmatrix} L \\ F \end{pmatrix} \mathbf{x} - B = 0$$

is equivalent to solving the following linear system:

$$A^T A\mathbf{x} = A^T B$$

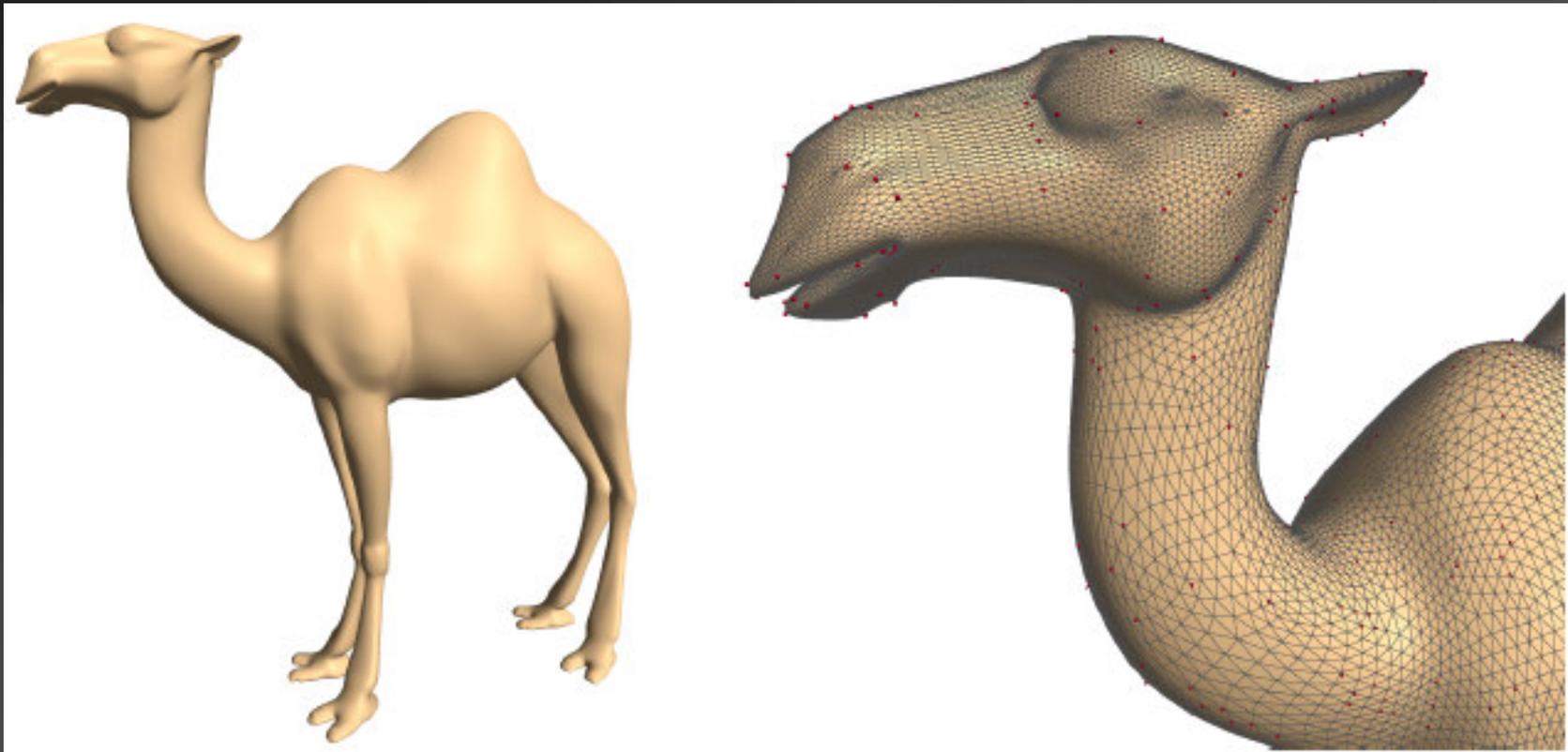
We solve that system using a direct method involving matrix multiplications, factorizations, and front and back substitutions. These are implemented using ATLAS if possible, but using “textbook” algorithms otherwise.

Example - Camel



100 Control Vertices

Example - Camel



2000 Control Vertices

Stage 2: Making Local Adjustments

- The vertices of the connectivity mesh have been distributed in such a way as to make them sufficiently close to the point cloud
- The vertices in the LS mesh must be projected onto surfaces representing nearby neighbourhoods in the point cloud.
- To find the associated neighbourhood for a vertex v in the connectivity mesh, we find:
 - A point p in the point cloud such that p is the nearest point in the point cloud to v .
 - The neighbourhood of p in the point cloud.

Associated Neighbourhood of v

- To find the associated neighbourhood for a vertex v in the connectivity mesh, we find:
 - A point p in the point cloud such that p is the nearest point in the point cloud to v .
 - Current implementation is brute-force search.
 - Performs quickly enough for moderately-sized meshes.
 - The neighbourhood of p in the point cloud.
 - Any algorithm which provides a neighbourhood of p will do.
 - Eg. K-Nearest-Neighbours

Projection Surface for v

- We have many options for finding a projection surface approximating the neighbourhood of v
 - Choice for Implementation
 - Find the best-fit plane to the unweighted neighbourhood of v
 - How?
 - PCA
-

Best-fit plane

- Fix the plane to the centroid of the neighbourhood
 - Use PCA to compute the normal vector
 - Compute 3 x 3 covariance matrix using standard “textbook” algorithm
 - Compute eigenvectors and eigenvalues of covariance matrix using VTK routine
 - The eigenvector of the covariance matrix corresponding to the smallest eigenvalue
 - Finally, we make our adjustment by projecting v onto the plane.
-

DEMO



Limitations of Current Implementation

■ Scalability

- In computing the LS Mesh, we must compute the matrix product $A^T A$ where A is an $(n + m) \times n$ matrix where n is the number of vertices in the connectivity mesh, and m is the number of anchor points
 - The optimized matrix multiplication CBLAS routine provided with ATLAS unfortunately produces inaccurate results
 - Our implementation uses a “textbook” algorithm that computes in reasonable time for meshes with at most ~ 1000 vertices
 - Solution: Use an optimized algorithm for matrix multiplication
-

Limitations of Current Implementation

■ Separation of Models

- Our software uses one mesh model to extract a connectivity mesh and to populate the point cloud.
 - This prevents the fitting of stored meshes to arbitrary point clouds.
 - Improving this first requires some method(possibly expert/user-guided) for determining the anchor points in the connectivity mesh model and the associated points in the point cloud.
 - This modification also requires that an efficient K-Nearest Neighbours algorithm be implemented for determining local neighbourhoods around arbitrary points in the point cloud.
-

Limitations of Current Implementation

- Local Adjustment Projection Surfaces
 - With PCA, our local adjustments are made by projecting onto a best-fit plane with respect to an unweighted scheme.
 - This does not take into account any surface curvature in the local neighbourhood.
 - Solution: Use MLS instead of PCA to implement the second stage of our algorithm for making local adjustments to the LS mesh geometry.
 - With MLS, we would be able to fit a polynomial to a locally-weighted height field, and then make our adjustments to the LS mesh geometry by “projecting” onto this polynomial
-

Unexplored questions

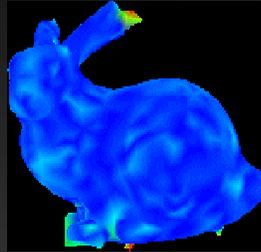
- The effect of genus > 0 should be minimal assuming appropriately chosen anchor vertices in the connectivity mesh and correspondence to the point cloud.
 - How much do results improve with anchor vertices chosen in order of descending error induced by the removal of the potential anchor vertex? What about other heuristics?
-

Error Plot – Bunny

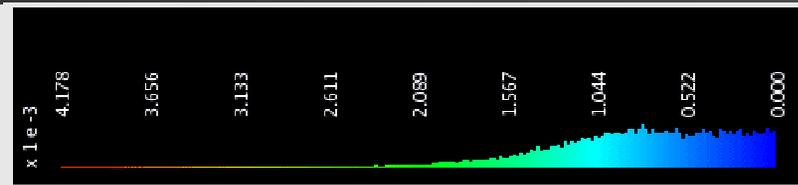
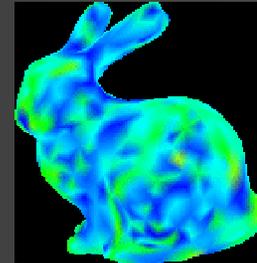


bunny mesh.
1064 vertices 2000 faces

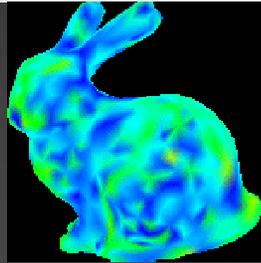
bunny:
53 anchor points



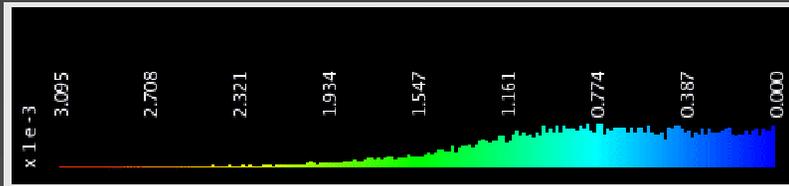
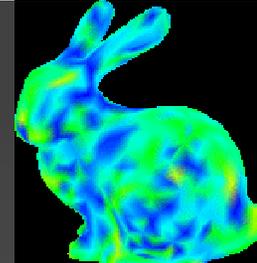
bunny:
268 anchor points



bunny:
532 anchor points



bunny:
798 anchor points

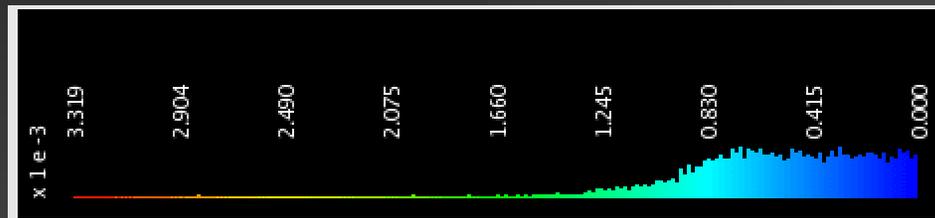
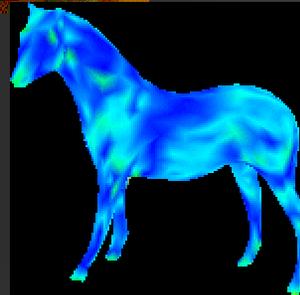


Error Plot = H

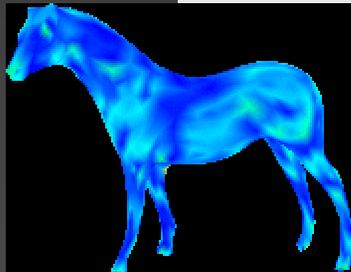
horse mesh:
1002 vertices 2000 faces



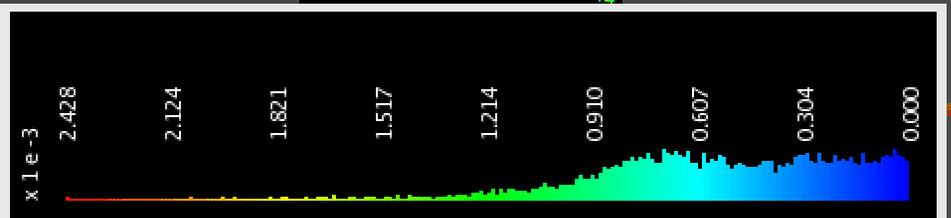
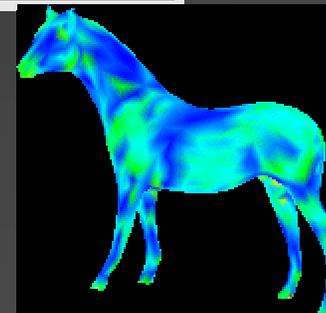
horse:
501 anchor points



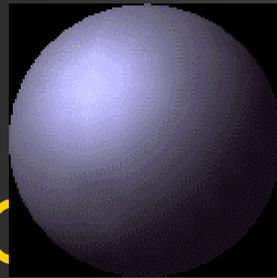
horse:
251 anchor points



horse:
751 anchor points

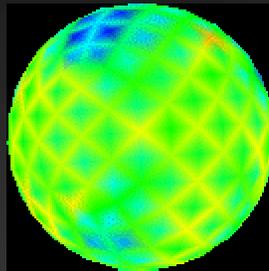


Error Plot – Sp

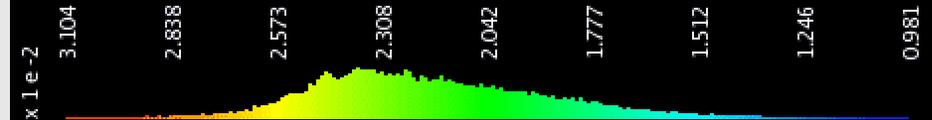
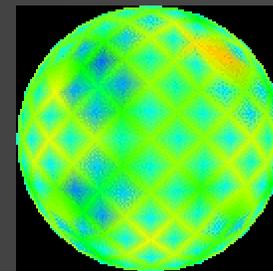


sphere mesh: 379 vertices 754 faces

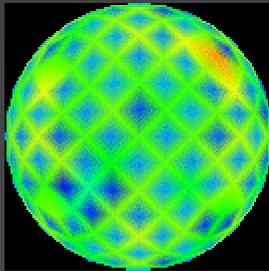
sphere:
18 anchor points



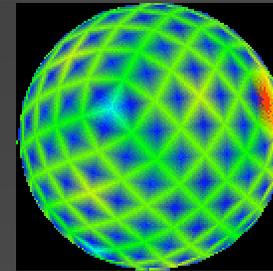
sphere:
94 anchor points



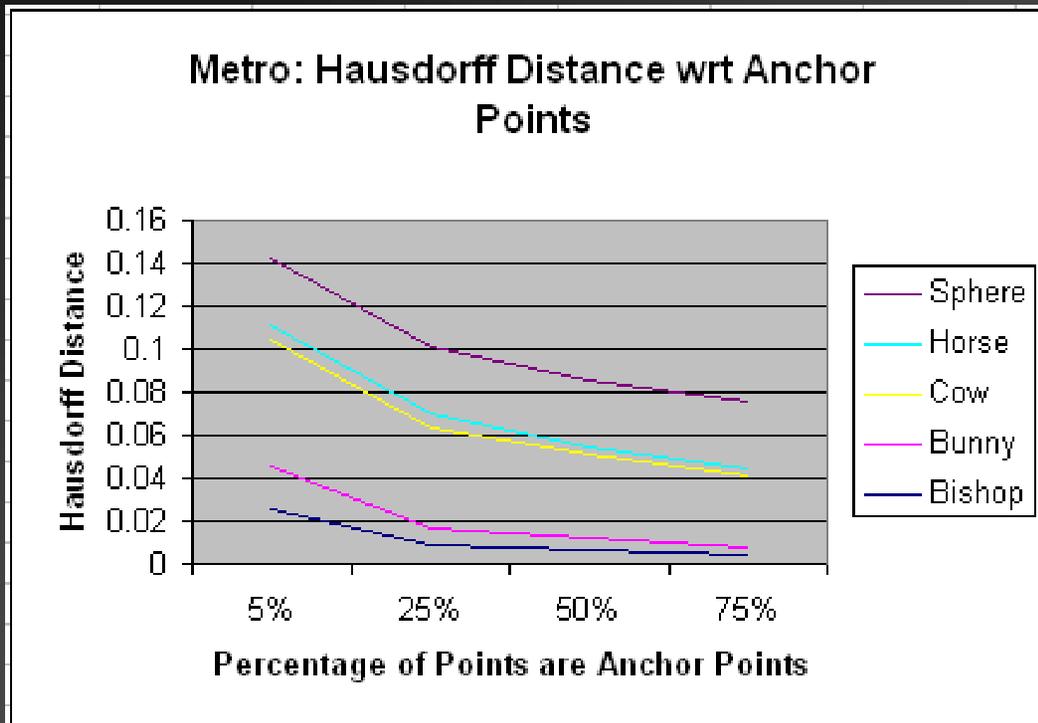
sphere:
189 anchor points



sphere:
284 anchor points



Error Plot – Metro



Data Tab Hausdorff Distance				
	5%	25%	50%	75%
Bishop	0.025938	0.009151	0.006293	0.004
Bunny	0.019911	0.007852	0.00563	0.003713
Cow	0.058247	0.046626	0.039156	0.033382
Horse	0.006654	0.005833	0.003618	0.003559
Sphere	0.031936	0.031532	0.030732	0.030791

References & Acknowledgements

- Alexa, M. et al., "Point Set Surfaces," *IEEE Visualization 2001*
- Burden, R.L., and Faires, J.D., "Numerical Analysis," 7th ed., 2001
- Hoppe, H., et al., "Surface Reconstruction from unorganized points," *ACM SIGGRAPH 1992*, 71-78
- Levin, D., "Mesh-Independent Surface Interpolation," *Advances in Computational Mathematics*, 2001
- Martin, K., et al., "VTK Home Page," <http://www.vtk.org/>
- Sorkine, O. and Cohen-Or, D., "Least-Squares Meshes," *SMI 2004*
- Whaley, R.C., and Dongara, J.J., "ATLAS Homepage," <http://math-atlas.sourceforge.net/>

Thank you for your attention! ☺
