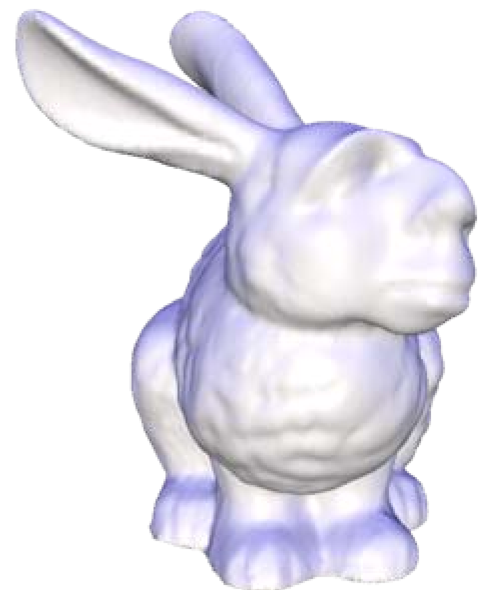
Encoding Meshes in Differential Coordinates

Daniel Cohen-Or

Tel Aviv University

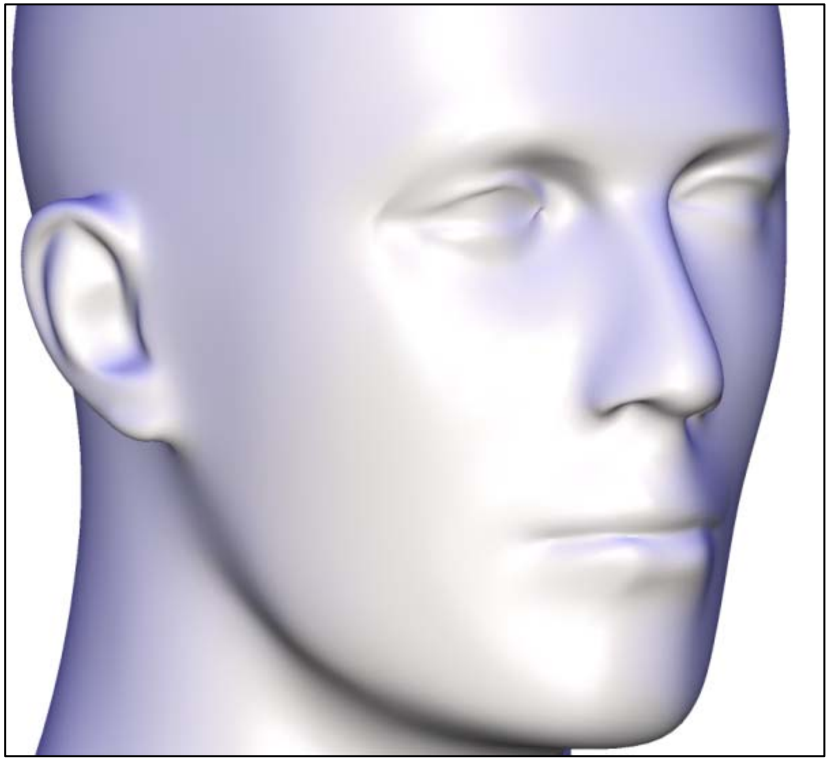
Outline

- Differential surface representation
- Compact shape representation
  - Mesh editing and manipulation
- …about surface reconstruction



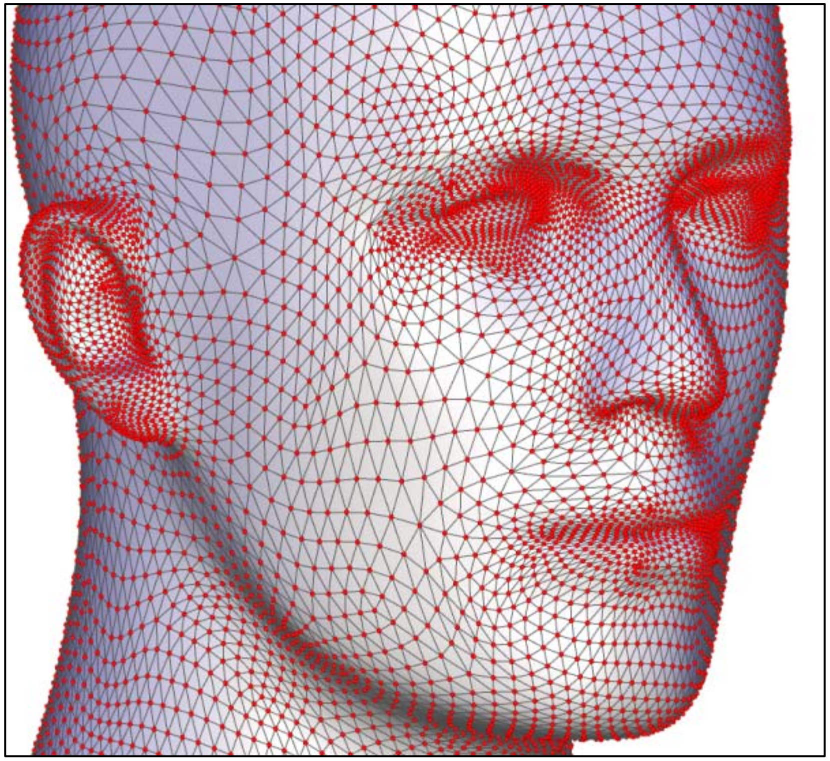
Irregular meshes

- In graphics, shapes are mostly represented by triangle meshes



Irregular meshes

- In graphics, shapes are mostly represented by triangle meshes



Irregular meshes

Geometry:

Vertex coordinates –

(x_1, y_1, z_1)

(x_2, y_2, z_2)

. . .

(x_n, y_n, z_n)

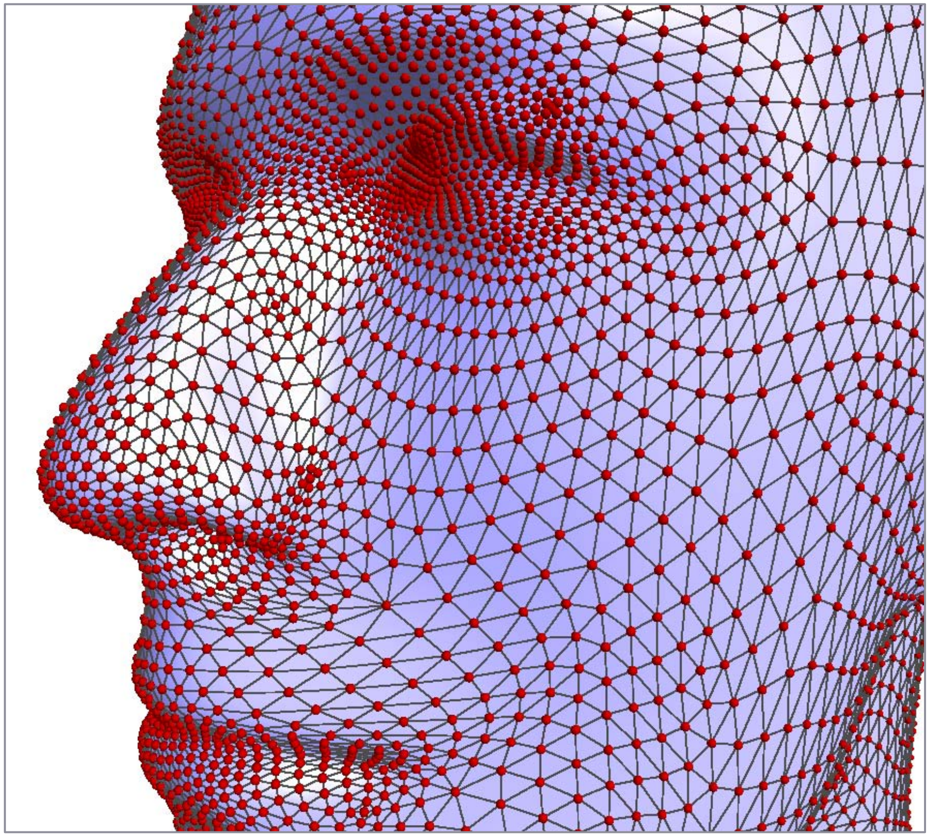
Connectivity (the graph)

List of triangles –

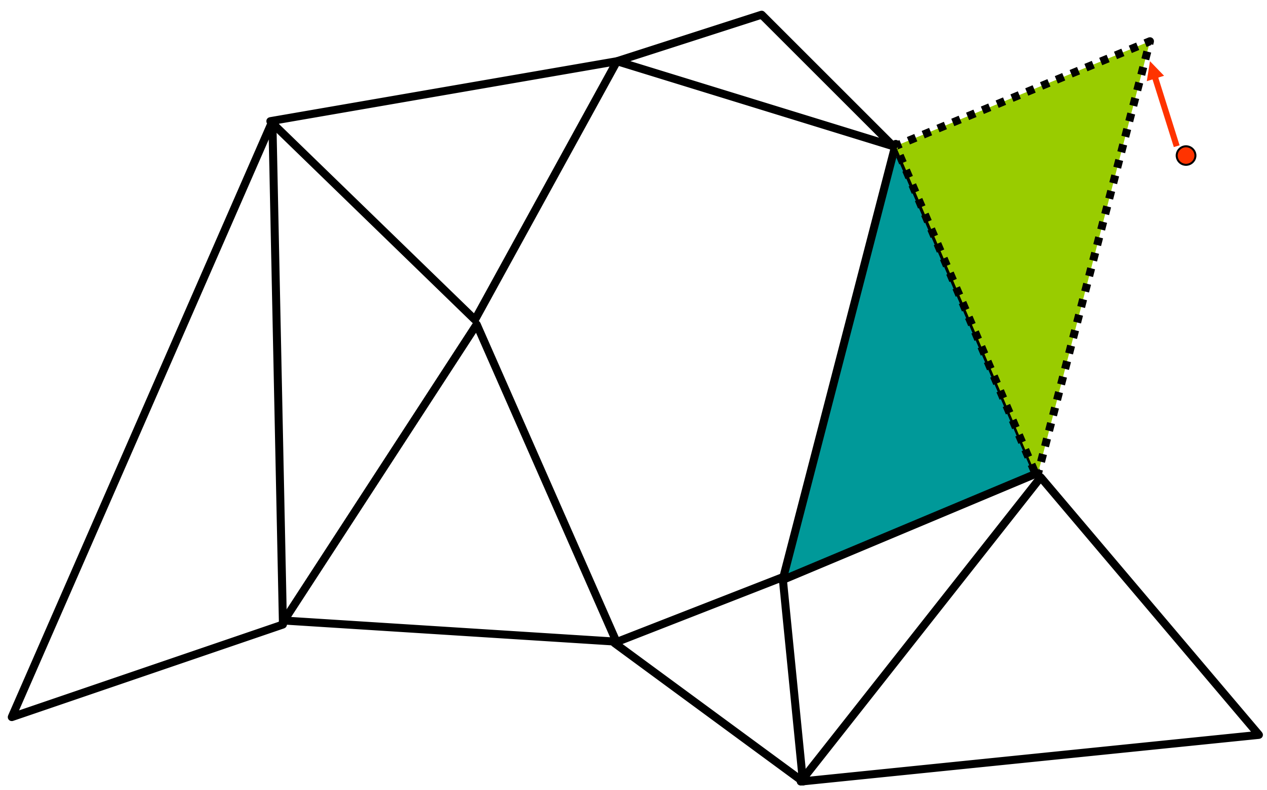
(i_1, j_1, k_1)

(i_2, j_2, k_2)

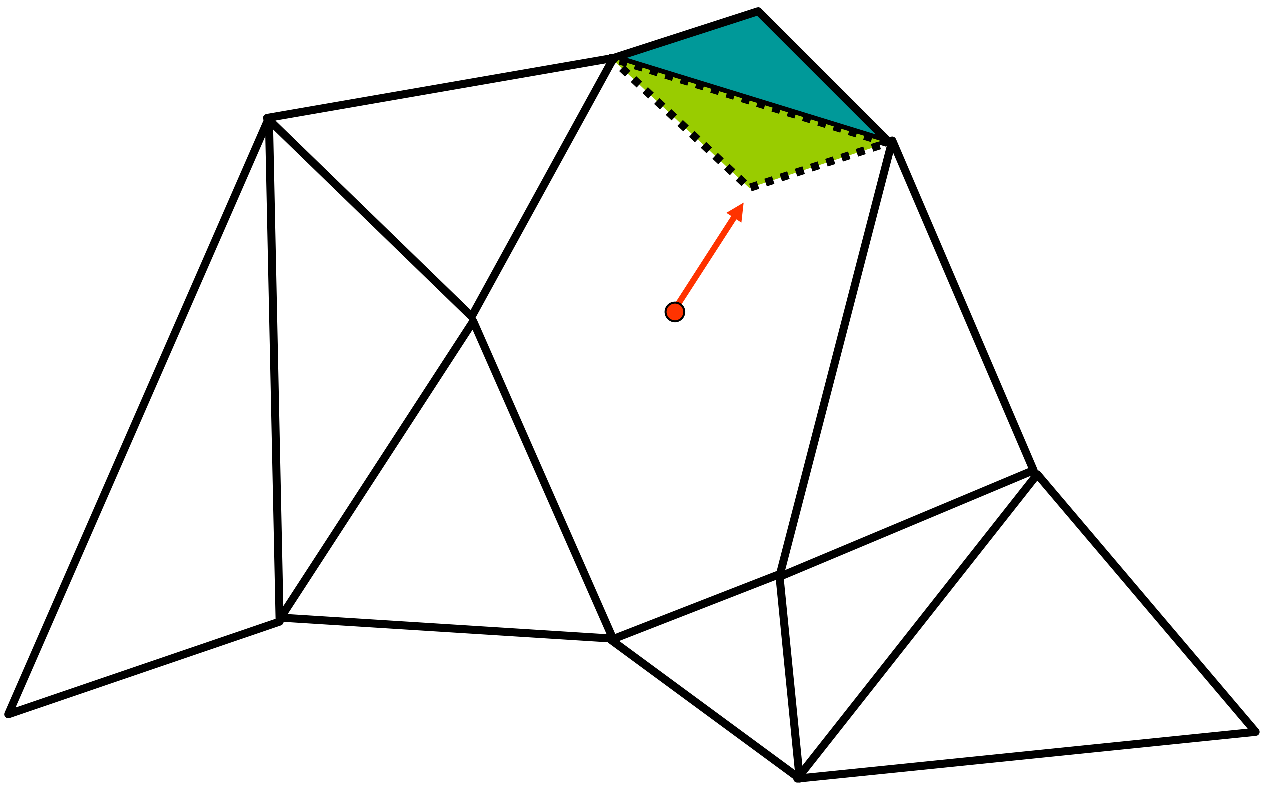
. . .



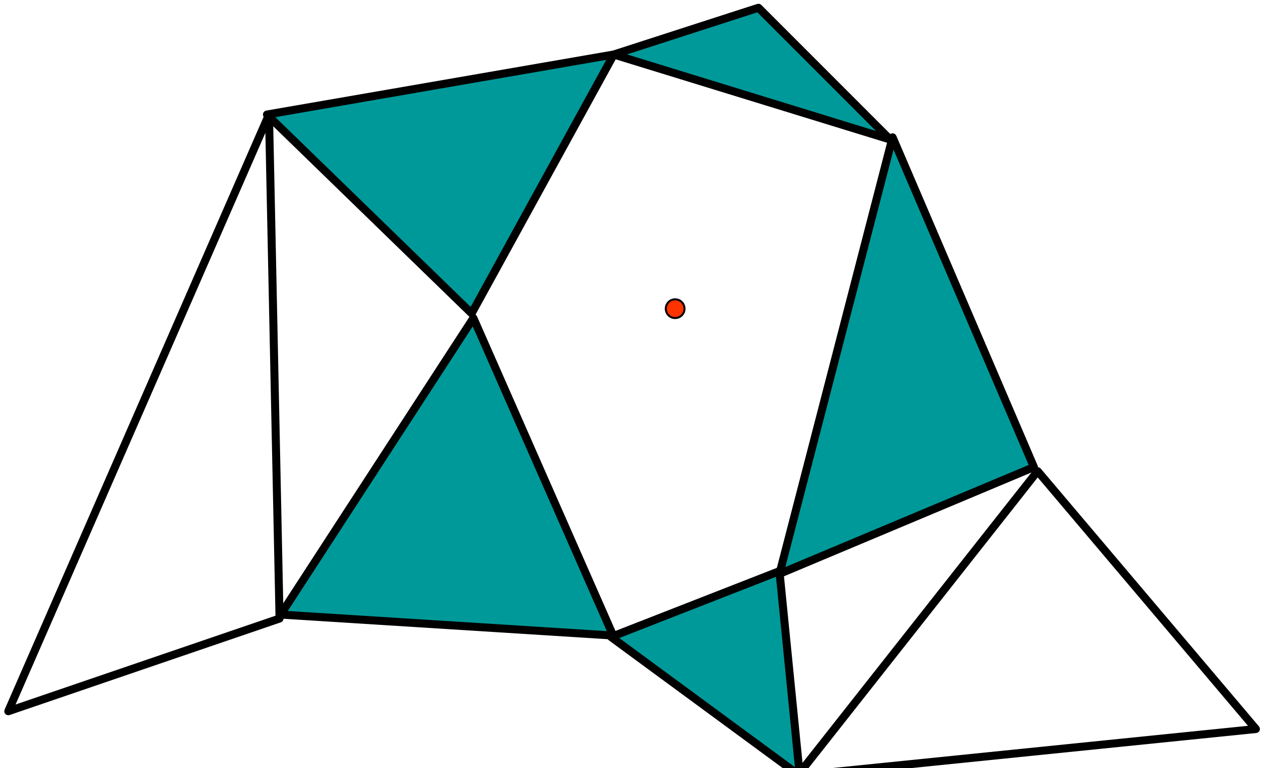
Parallelogram Prediction



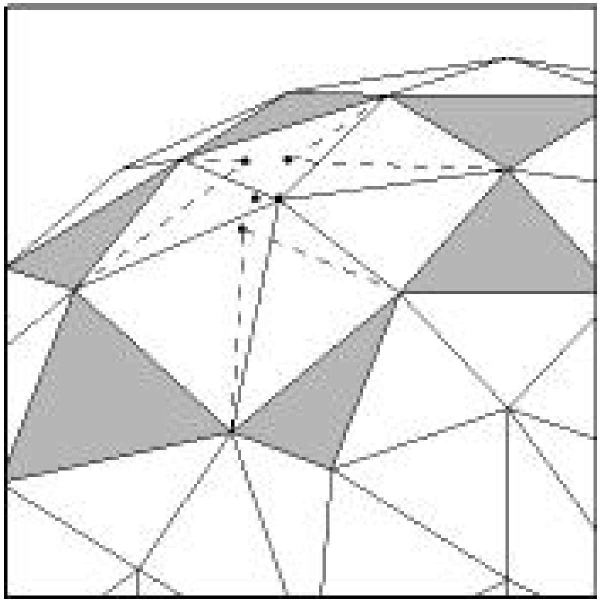
Parallelogram Prediction



K-way Prediction is better

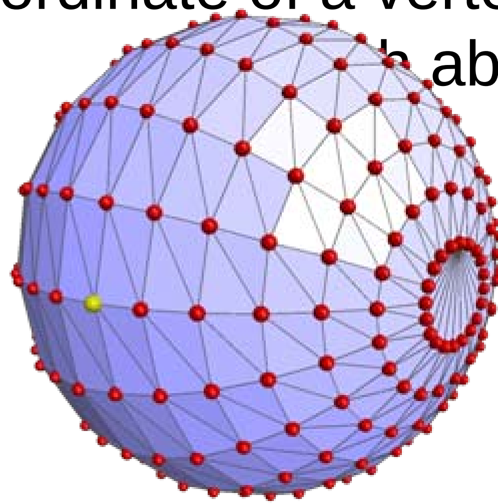


k-way prediction is like predicting that a vertex is in the average of its adjacent neighbors



Motivation

- Meshes are great, but:
  - Topology is explicit, thus hard to handle
  - Geometry is represented in a *global* coordinate system
    - Single Cartesian coordinate of a vertex doesn't say
      …h about the shape

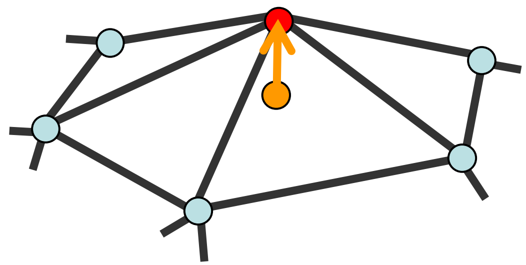


Differential coordinates

- Represent *local detail* at each surface point
  - better describe the shape
- Linear transition from global to differential
- Useful for operations on surfaces where surface details are important

Differential coordinates

- Detail = surface – *smooth*(surface)
- Smoothing = averaging



$$\boldsymbol{\delta}_i = \mathbf{v}_i - \frac{1}{d_i} \sum_{j \in N(i)} \mathbf{v}_j$$

$$\boldsymbol{\delta}_i = \sum_{j \in N(i)} \frac{1}{d_i} \left(\mathbf{v}_i - \mathbf{v}_j \right)$$

Laplacian matrix

- The transition between the δ and *xyz* is linear:

$$\begin{pmatrix} \quad \mathrm{L} \quad \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \delta_1^{(x)} \\ \delta_2^{(x)} \\ \vdots \\ \vdots \\ \delta_n^{(x)} \end{pmatrix}$$

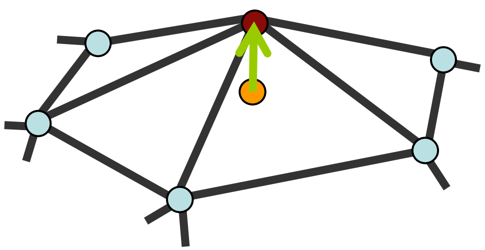
$$A_{ij} = \begin{cases} 1 & i \in N(j) \\ 0 & otherwise \end{cases}$$

$$D_{ij} = \begin{cases} d_i & i = j \\ 0 & otherwise \end{cases}$$

$$\mathrm{L} = \mathrm{I} - \mathrm{D}^{-1}\mathrm{A}$$

Laplacian matrix

- The transition between the δ and *xyz* is linear:



$$\boldsymbol{\delta}_i = \sum_{j \in N(i)} w_{ij} \left(\mathbf{v}_i - \mathbf{v}_j \right)$$

$$\mathbf{L}\,\mathbf{v_x} = \delta_\mathbf{x}$$

$$\mathbf{L}\,\mathbf{v_y} = \delta_\mathbf{y}$$

$$\mathbf{L}\,\mathbf{v_z} = \delta_\mathbf{z}$$

Basic properties

- Rank(L) = n-c (n-1 for connected meshes)
- We can reconstruct the xyz geometry from delta up to translation

$$L\mathbf{x} = \boldsymbol{\delta}$$

$$\mathbf{x} = L^{-1}\boldsymbol{\delta}$$

Quantizing differential coordinates

"High-pass Quantization for Mesh Encoding", Sorkine et al. 03

- Quantization is one of the major methods to reduce storage space of geometry data
- What happens if we quantize the δ-coordinates?
  - Can we still go back to xyz?
  - How does the reconstruction error behave?

$$L\mathbf{x} = \boldsymbol{\delta}$$

$$\boldsymbol{\delta} \;\rightarrow\; \boldsymbol{\delta}' = \boldsymbol{\delta} + \varepsilon$$

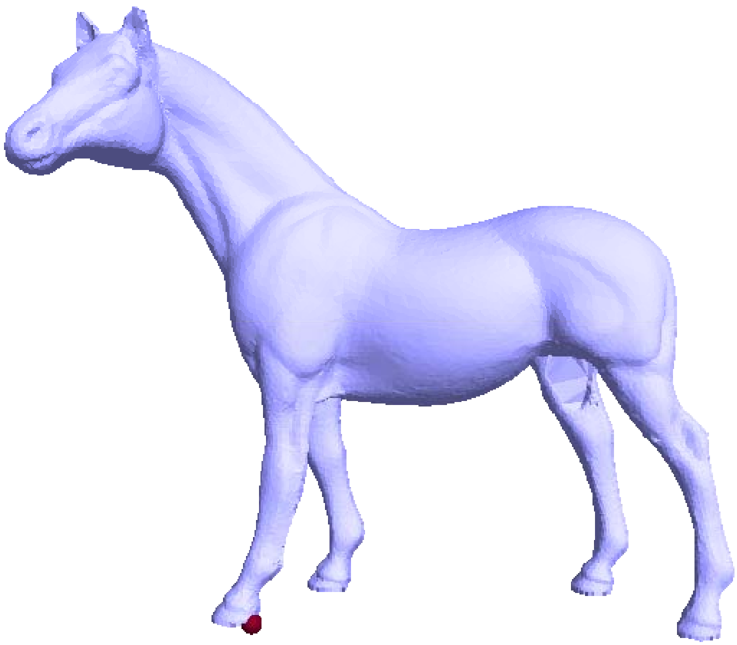
Quantizing differential coordinates

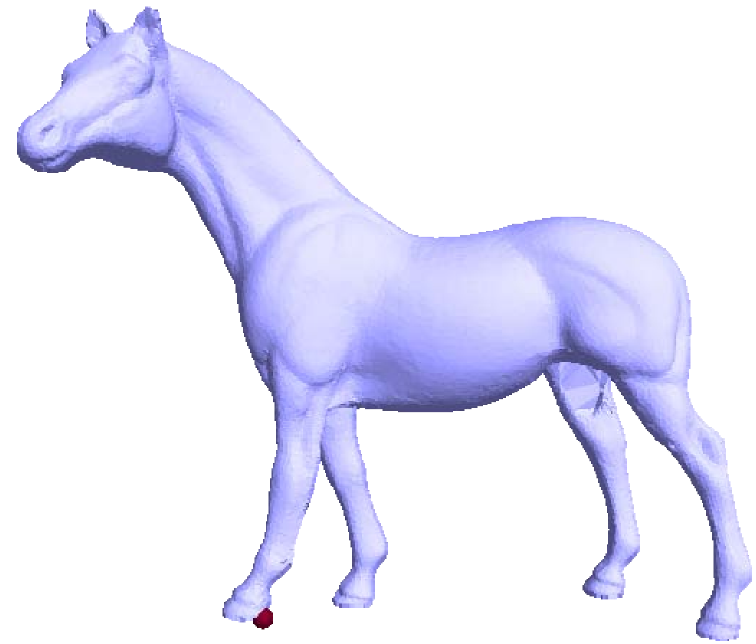
How does the reconstruction error behave?

$$\mathbf{x}' = L^{-1}\boldsymbol{\delta}' = L^{-1}(\boldsymbol{\delta} + \varepsilon)$$

Quantizing differential coordinates

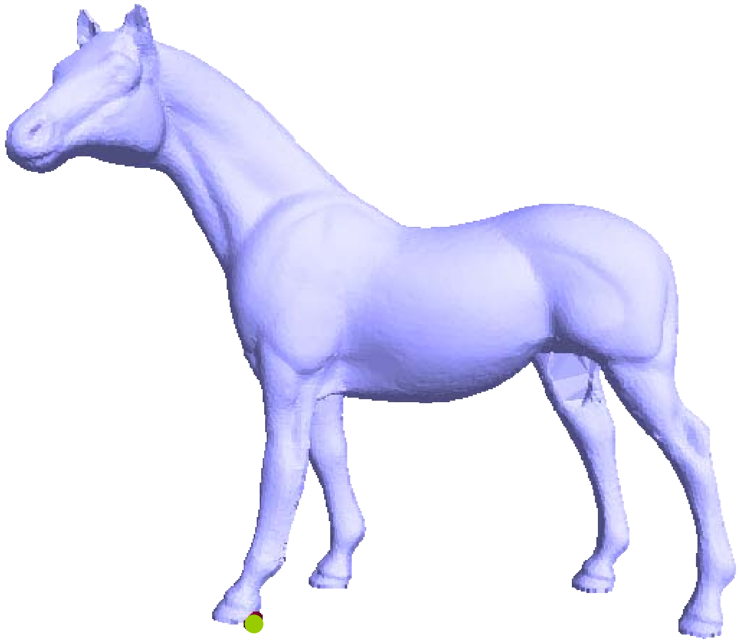
- Find the differences between the horses…





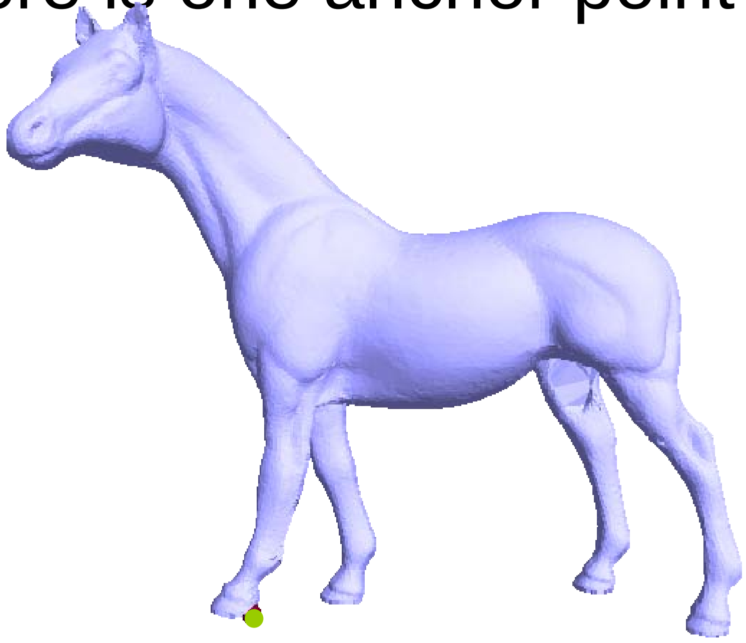
Quantizing differential coordinates

- This one is the original horse model



Quantizing differential coordinates

- This is the model after quantizing δ to 8 bits/coordinate
- There is one anchor point (front left leg)



Quantizing differential coordinates

- Original model

Quantizing differential coordinates

- This is the model after quantizing δ to 7 bits/coordinate, one anchor

Quantizing differential coordinates

Quantizing differential coordinates

Quantization error

- A coarsely-sampled sphere

Quantization error

- After quantization to 8 bits/coordinate

Quantization error

- A finely-sampled sphere:

Quantization error

- After (the same) quantization to 8 bits/coordinate…

Quantizing differential coordinates

Quantizing differential coordinates

Spectral properties of L

- Sort the eigenvalues in accending order:

"frequencies" ⟶ $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n-1} \leq \lambda_n$

eigenvectors ⟶ $\mathbf{e}_1, \quad \mathbf{e}_2, \quad \ldots \quad \mathbf{e}_{n-1}, \quad \mathbf{e}_n$

- We can represent the geometry in L's eigenbasis:

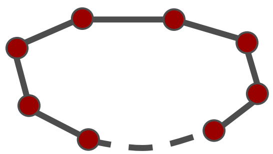
[Taubin 95]

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \underbrace{c_1\mathbf{e}_1 + c_2\mathbf{e}_2}_{\text{low frequency components}} + \ldots + \underbrace{c_{n-1}\mathbf{e}_{n-1} + c_n\mathbf{e}_n}_{\text{high frequency components}}$$

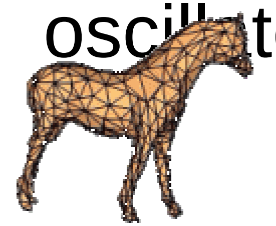
The spectral basis

"Spectral Mesh Compression", Karni and Gotsman 00

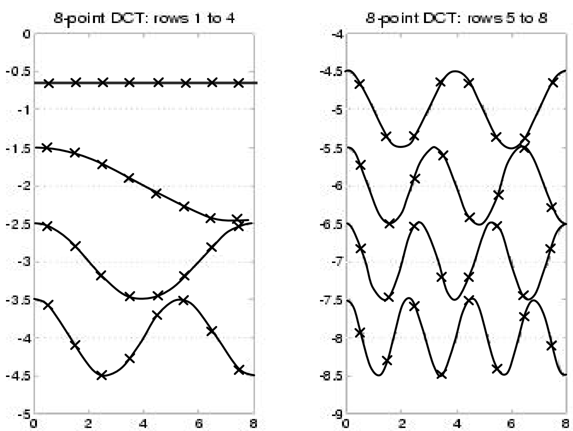
- First functions are smooth and slow, last oscillate a lot



chain topology

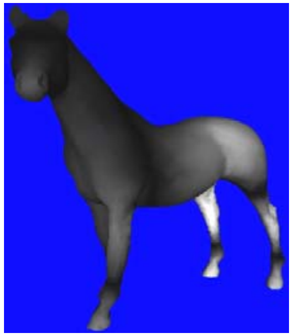


horse topology



spectral basis of L = the DCT basis

2nd basis function

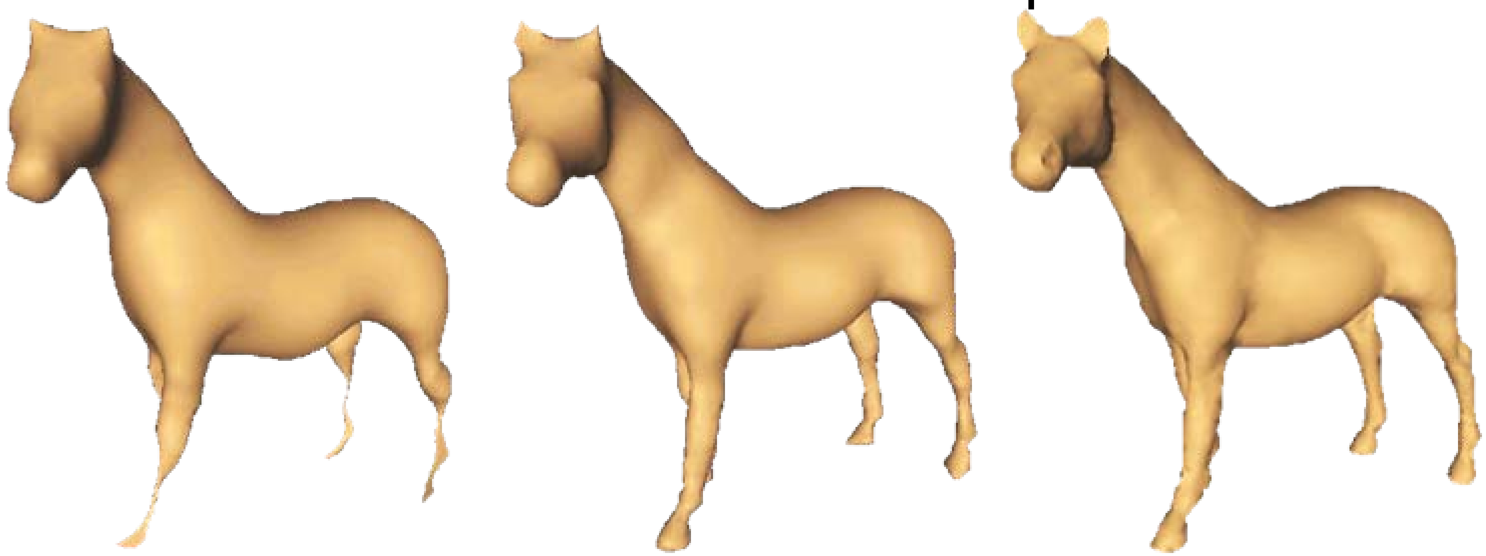


10th basis function

100th basis function

Spectral compression

- [Karni and Gotsman 2000]: progressive compression scheme (“3D JPEG”)
  - Drop the high-frequency spectral coefficients
  - Both the encoder and the decoder perform spectral decomposition of L.



Spectrum of the quantization error

“High-pass Quantization for Mesh Encoding”, Sorkine et al. 03

$$\mathbf{x}' = L^{-1}\boldsymbol{\delta}' = L^{-1}(\boldsymbol{\delta} + \varepsilon) = \mathbf{x} + L^{-1}\varepsilon$$

Spectrum of the quantization error

"High-pass Quantization for Mesh Encoding", Sorkine et al. 03

$$\mathbf{x}' = \mathbf{x} + L^{-1}\varepsilon$$

$$\varepsilon = c_1\mathbf{e}_1 + c_2\mathbf{e}_2 + \ldots + c_{n-1}\mathbf{e}_{n-1} + c_n\mathbf{e}_n$$

$$L^{-1}\varepsilon = \frac{1}{\lambda_1}c_1\mathbf{e}_1 + \frac{1}{\lambda_2}c_2\mathbf{e}_2 + \ldots + \frac{1}{\lambda_{n-1}}c_{n-1}\mathbf{e}_{n-1} + \frac{1}{\lambda_n}c_n\mathbf{e}_n$$

Large numbers = amplification

Numbers < 1 = attenuation

Low frequency error

Low frequency error

Error spectrum matters

- Quantizing Cartesian coordinates produces error with mostly high-frequency modes
- This affects the normals and thus the lighting
- Human perception is sensitive to high-frequency errors

- Quantizing delta-coordinates produces low-frequency error
- Strives to preserve local surface properties
- We are less sensitive to low-frequency errors

Low frequency error – anything we can do about it?

Low frequency error – anything we can do about it?

Bounding the low-frequency error

- "Nail" the model in place by adding more spatial constraints
- The more anchors – the higher λ_1 – the lower the error
  - Anchors cost additional storage space
  - In practice, less than 1% of the model vertices need to be anchored for visually good reconstruction

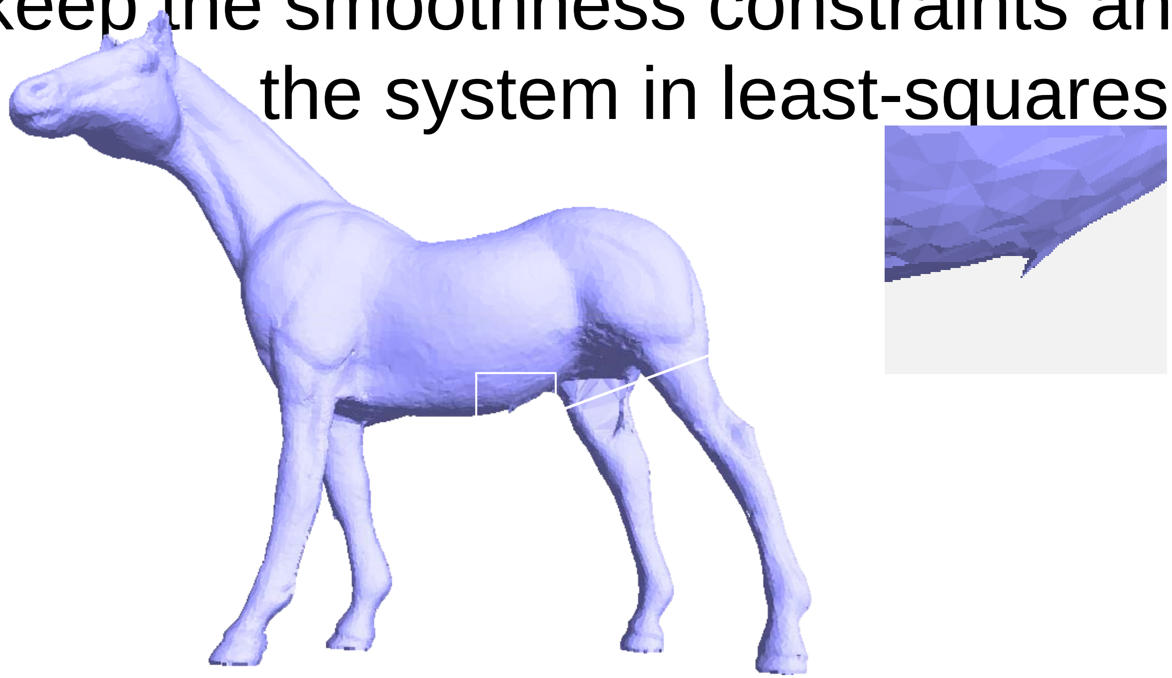
Invertible square Laplacian

- We could simply eliminate the anchors from the system, erasing the rows and the columns of the anchor vertices
- Use this “reduced” Laplacian instead of L and remember the anchors’ (x, y, z) positions separately

$$L = \begin{pmatrix} d_1 & -1 & 0 & \cdots & -1 & \cdots & \cdots & 0 \\ 0 & d_2 & & -1 & & & -1 & \\ \vdots & & d_3 & & & & & \\ \vdots & & & \ddots & & & & \\ \vdots & & & & \ddots & & & \\ \vdots & & & & & \ddots & & \\ 0 & -1 & & -1 & & -1 & d_{n-1} & \\ -1 & & -1 & & -1 & & & d_n \end{pmatrix}$$

Invertible Laplacian artifacts

- Produces bad results when we quantize δ because no smoothness constraints are posed on the anchors
- We keep the smoothness constraints and solve the system in least-squares sense!



Rectangular Laplacian

- We add equations for the anchor points
- By adding anchors the matrix becomes non-square, so we solve the system in least-squares sense:

$$\begin{pmatrix} & & L & & \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ \vdots \\ x'_n \end{pmatrix} = \begin{pmatrix} \delta'_1 \\ \delta'_2 \\ \vdots \\ \vdots \\ \delta'_n \\ x_1 \\ x_2 \end{pmatrix}$$

constrained anchor points

$$\min_{x'} \left\| A x' - \delta' \right\|$$

Low-frequency error

"High-pass Quantization for Mesh Encoding", Sorkine et al. 03

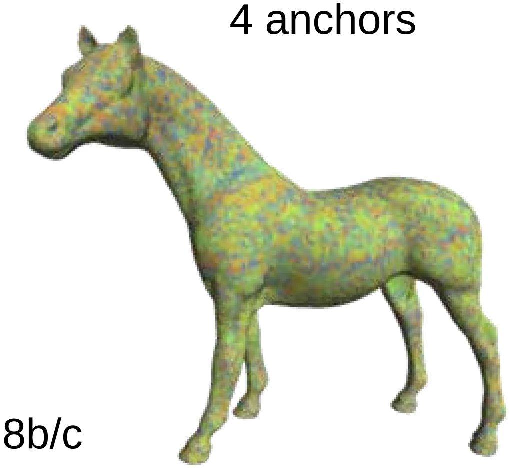
Positive error – vertex moves outside of the surface

Negative error – vertex moves inside the surface

δ-quantization 7b/c
2 anchors

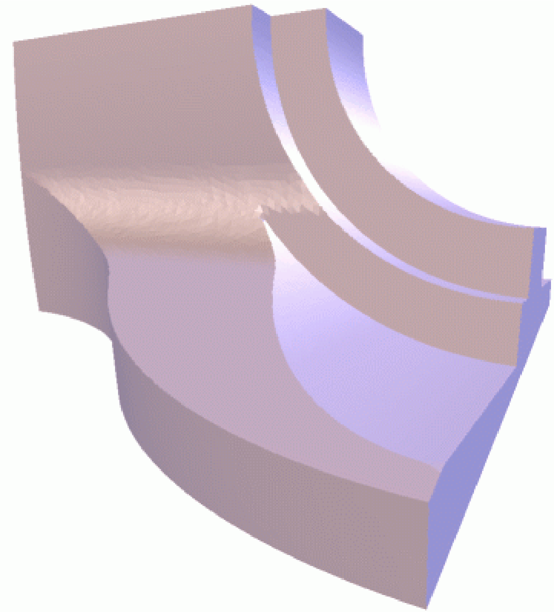
δ-quantization 7b/c
4 anchors

δ-quantization 7b/c
20 anchors

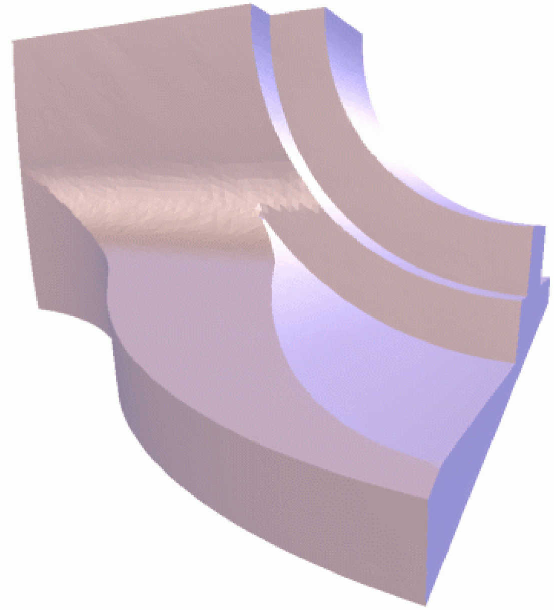


Cartesian
quantization 8b/c

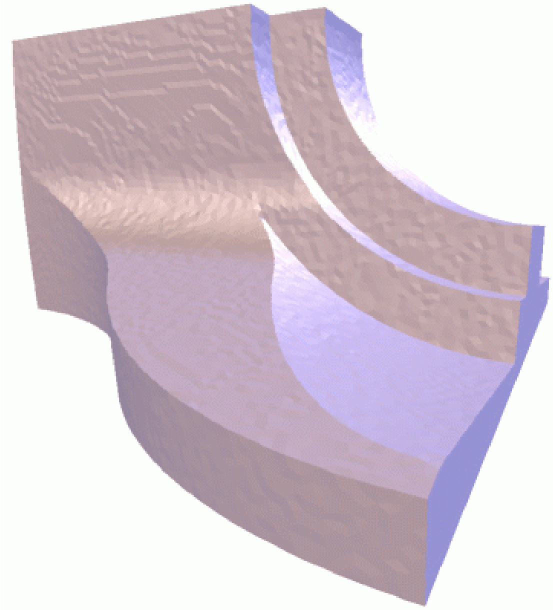
Some results



original



δ-quantization,
entropy 6.69



Cartesian quantization,
entropy 7.17

We compare to Touma-Gotsman predictive coder that uses Cartesian quantization

Some results

original

δ-quantization,
entropy 7.62

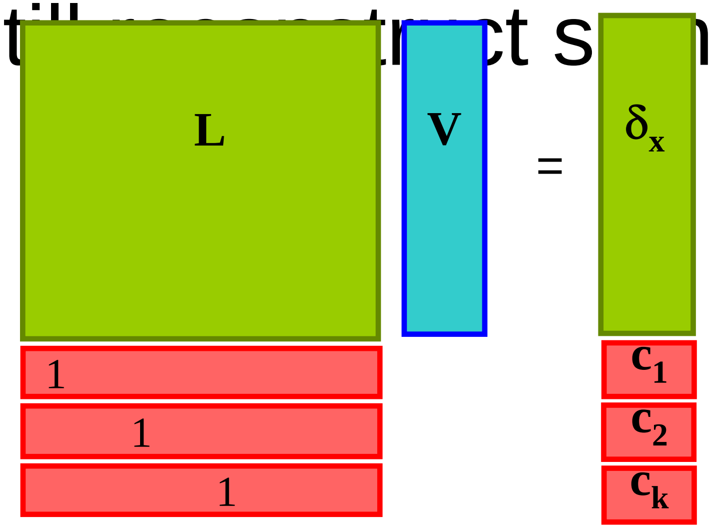
Cartesian quantization,
entropy 7.64

We compare to Touma-Gotsman predictive coder that uses Cartesian quantization

Shape from connectivity

"Least-squares Meshes", Sorkine and Cohen-Or 04

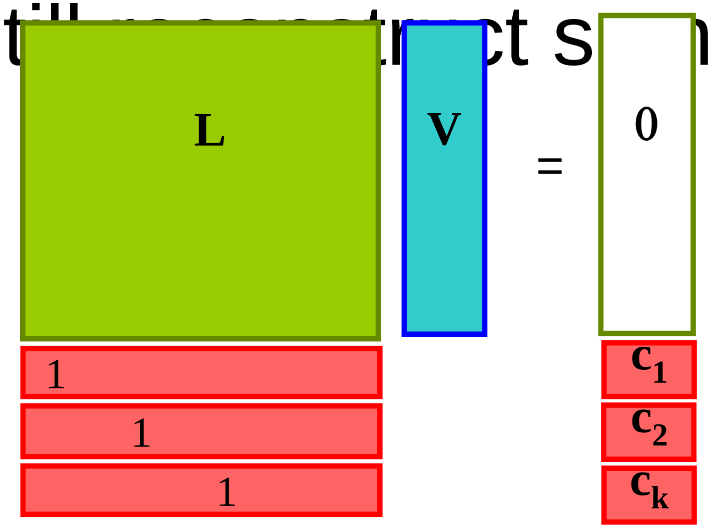
- What if we reduce delta information to zero bits??
- Can we still reconstruct same geometry?



Shape from connectivity

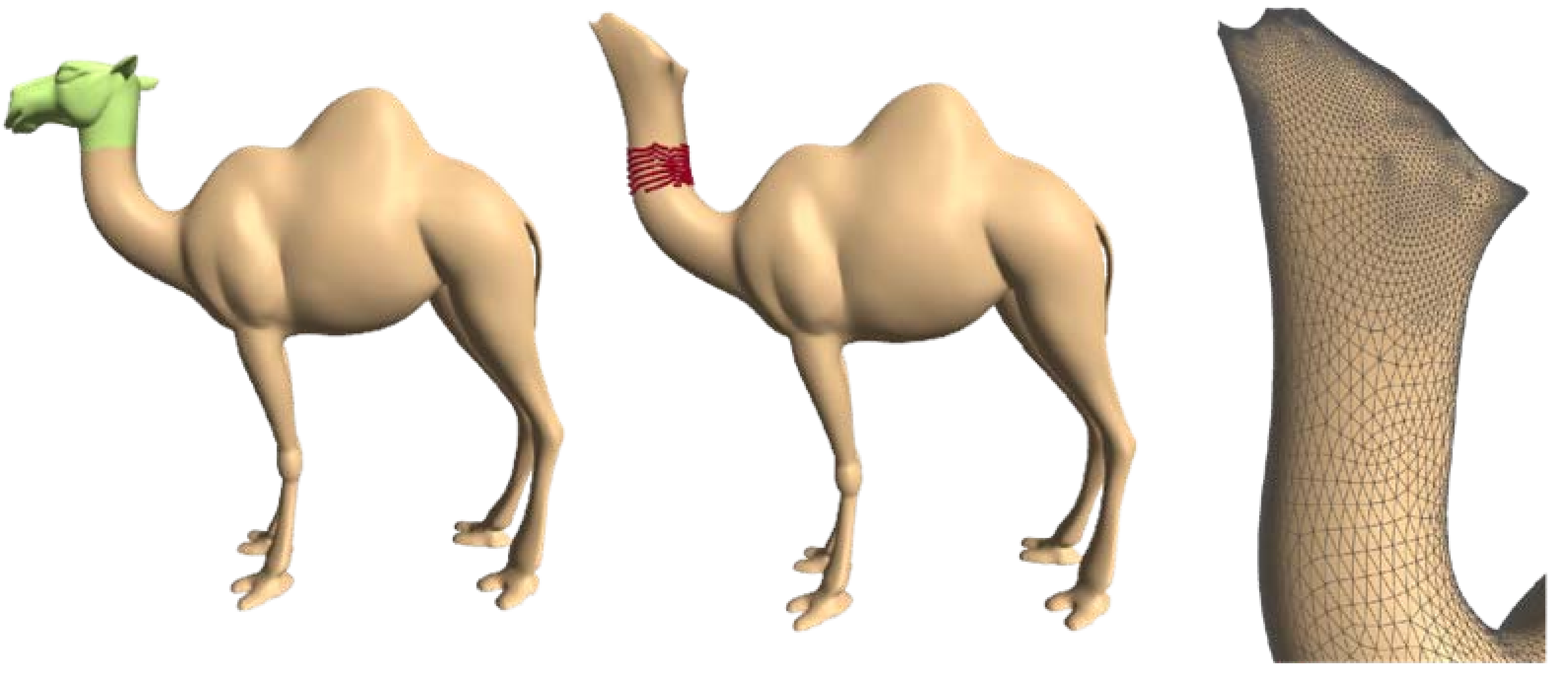
"Least-squares Meshes", Sorkine and Cohen-Or 04

- What if we reduce delta information to zero bits??
- Can we still reconstruct some geometry?



Geometry hidden in connectivity

"Least-squares Meshes", Sorkine and Cohen-Or 04

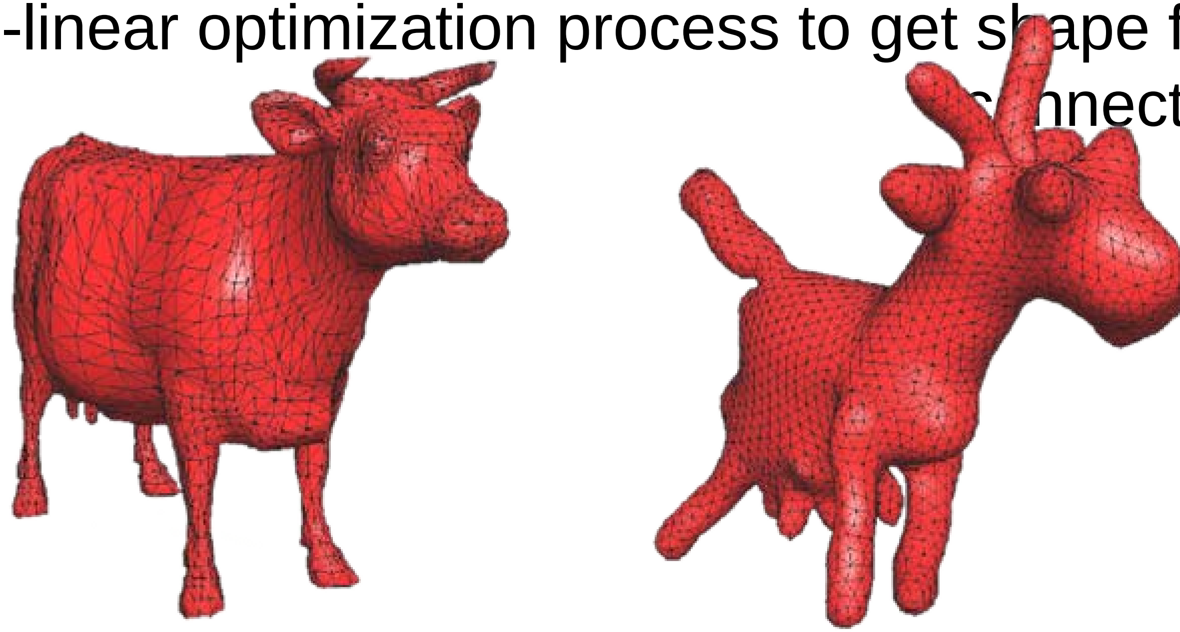


There is geometry in connectivity

Connectivity Shapes

“Connectivity Shapes”, Isenburg and et. 01

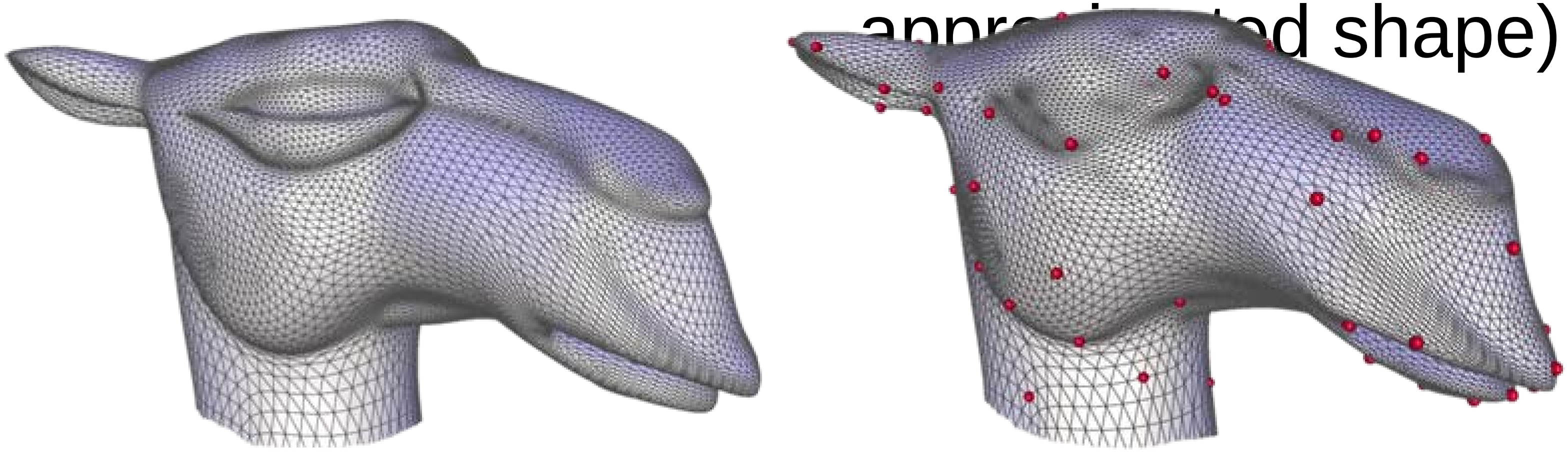
- Connectivity has geometric information in it
- Isenburg et al. showed how to get a shape from connectivity by assuming uniform edge length and smoothness
  - Non-linear optimization process to get shape from connectivity



Least-squares Meshes

"Least-squares Meshes", Sorkine and Cohen-Or 04

- Enrich the connectivity by sparse set of control points with geometry
- Solve a linear least-squares problem to reconstruct the geometry of all vertices (the approximated shape)



Basis functions

"Geometry-aware Bases for Shape Approximation", Sorkine et al. 05

- The geometry reconstructed by

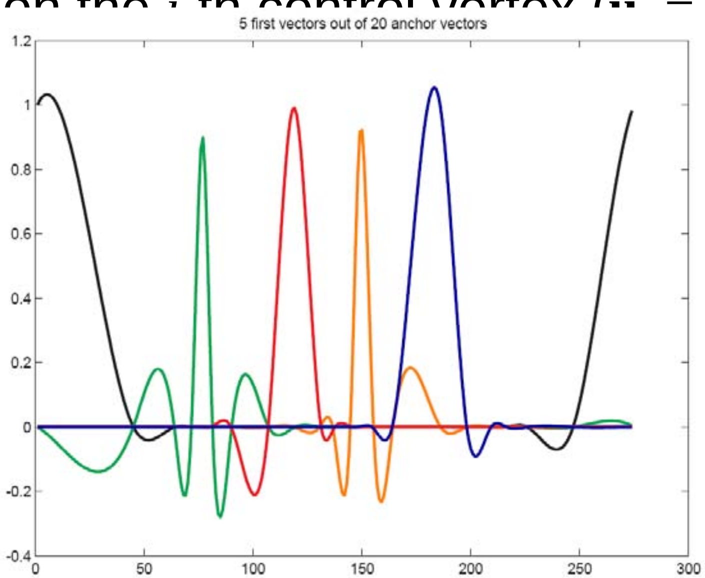
$$\mathbf{x} = A^*\mathbf{b} \qquad \left(A^* = (A^T A)^{-1} A^T \right)$$

is in fact a combination of k basis functions:

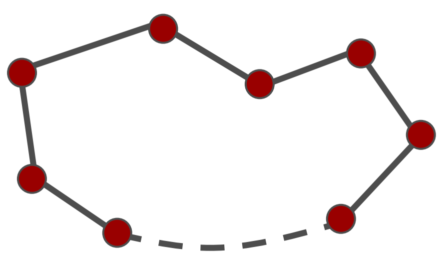
$$\mathbf{x} = A^* \begin{pmatrix} 0 \\ \vdots \\ 0 \\ c_1 \\ c_2 \\ \vdots \\ c_k \end{pmatrix} = c_1 A^* \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + c_2 A^* \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \ldots + c_k A^* \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

Basis functions

- The basis functions are defined on the entire mesh
  - Connectivity data
  - Tagging of the control vertices
- The bases satisfy (in LS sense):
  - Smooth everywhere : $L\mathbf{u_i} = 0$
  - Large on the i th control vertex ($\mathbf{u_i} = 1$) and vanish on all others



5 basis functions on a 2D mesh (simple chain)



Spectral basis vs. LS basis

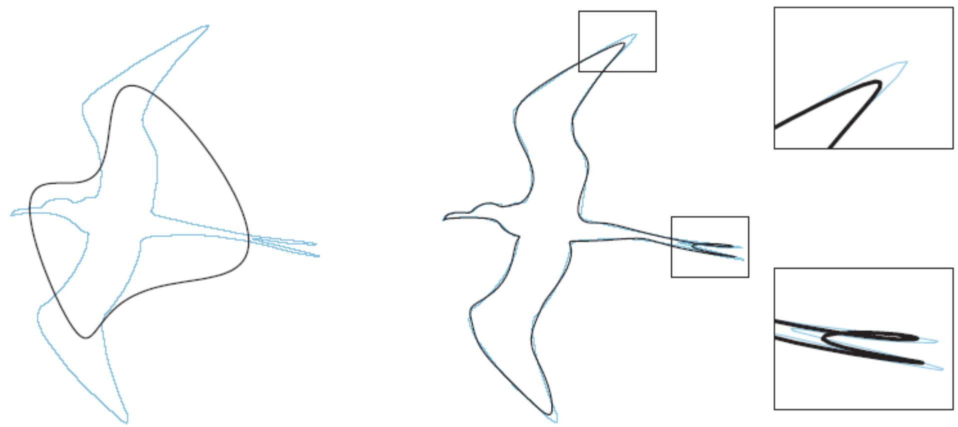
Spectral Basis

- The spectral basis does not take any geometric information into account
- Requires eigendecomposition – impractical for today’s meshes

LS basis

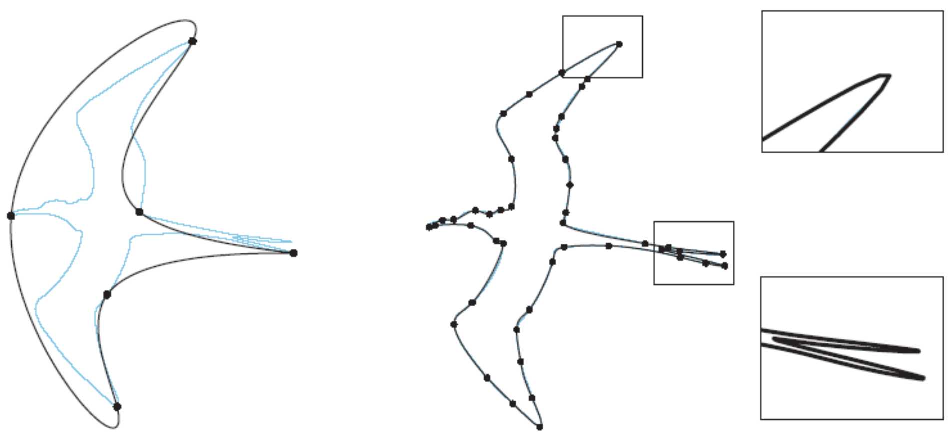
- The LS basis tags specific vertices, which makes it “geometry-aware”
- Requires solving sparse linear least-squares problem – can be done efficiently

Spectral basis vs. LS basis



6 spectral basis vectors

45 spectral basis vectors

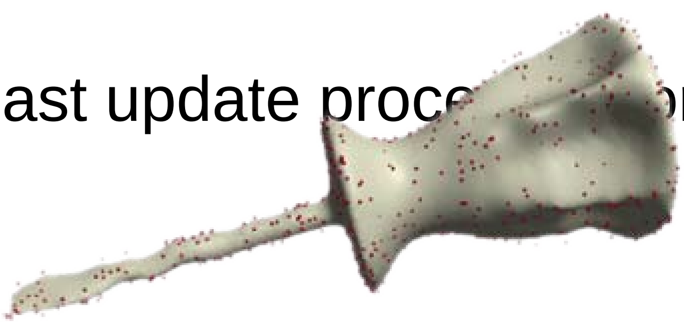


6 geometry-aware basis vectors

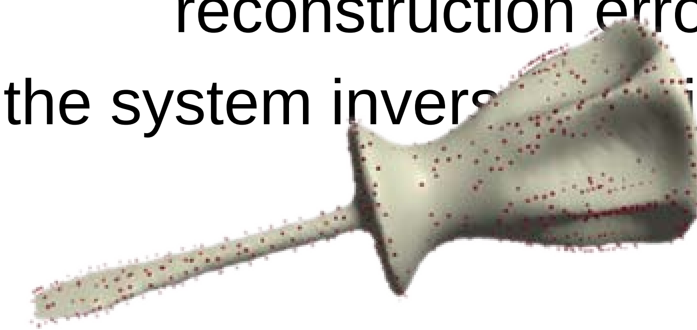
45 geometry-aware basis vectors

Selecting the control points

- Random selection
  - Faster, but less effective approximation
- Greedy approach
  - Place one-by-one at vertices with highest reconstruction error
  - Fast update proce[illegible]r the system invers[illegible]ix



Random selection



Greedy approach

1000 control points

Some results – varying number of control points

Original camel
39074 vertices

100 control points

600 control points

1200 control points

3600 control points

Some results – varying number of control points

Original feline
49864 vertices

100 control points

500 control points

4000 control points

9000 control points

Applications

- Progressive geometry compression and streaming

100
control points

1000
control points

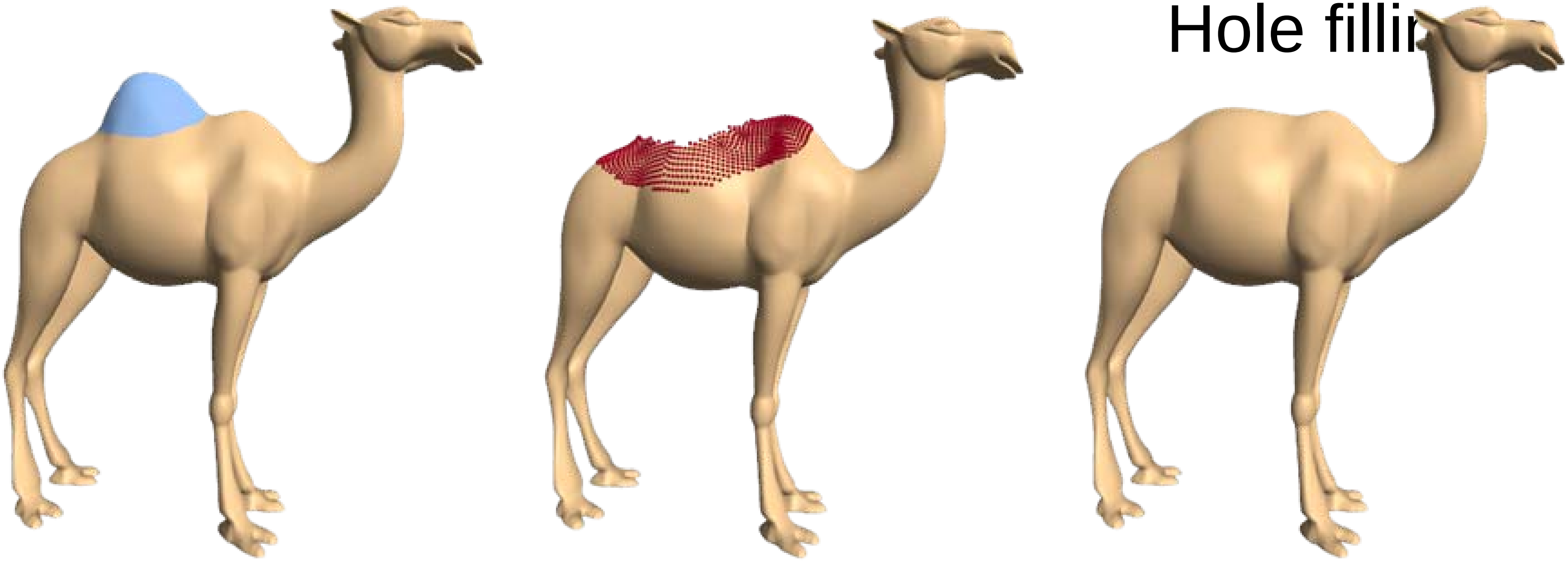
3000
control points

10000
control points

Applications

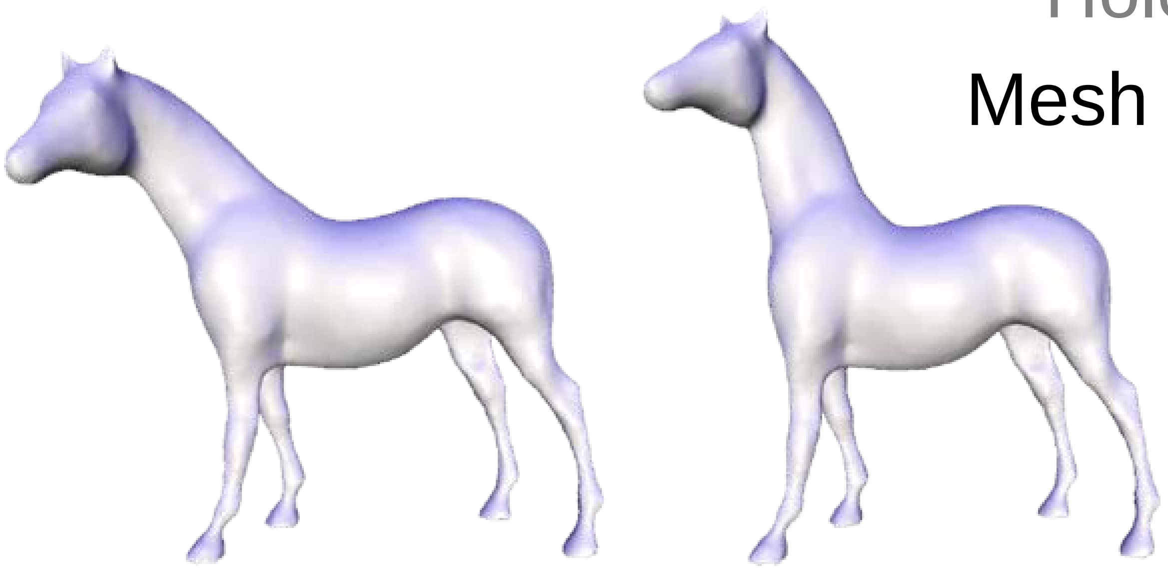
- Progressive geometry compression and streaming

Hole filli



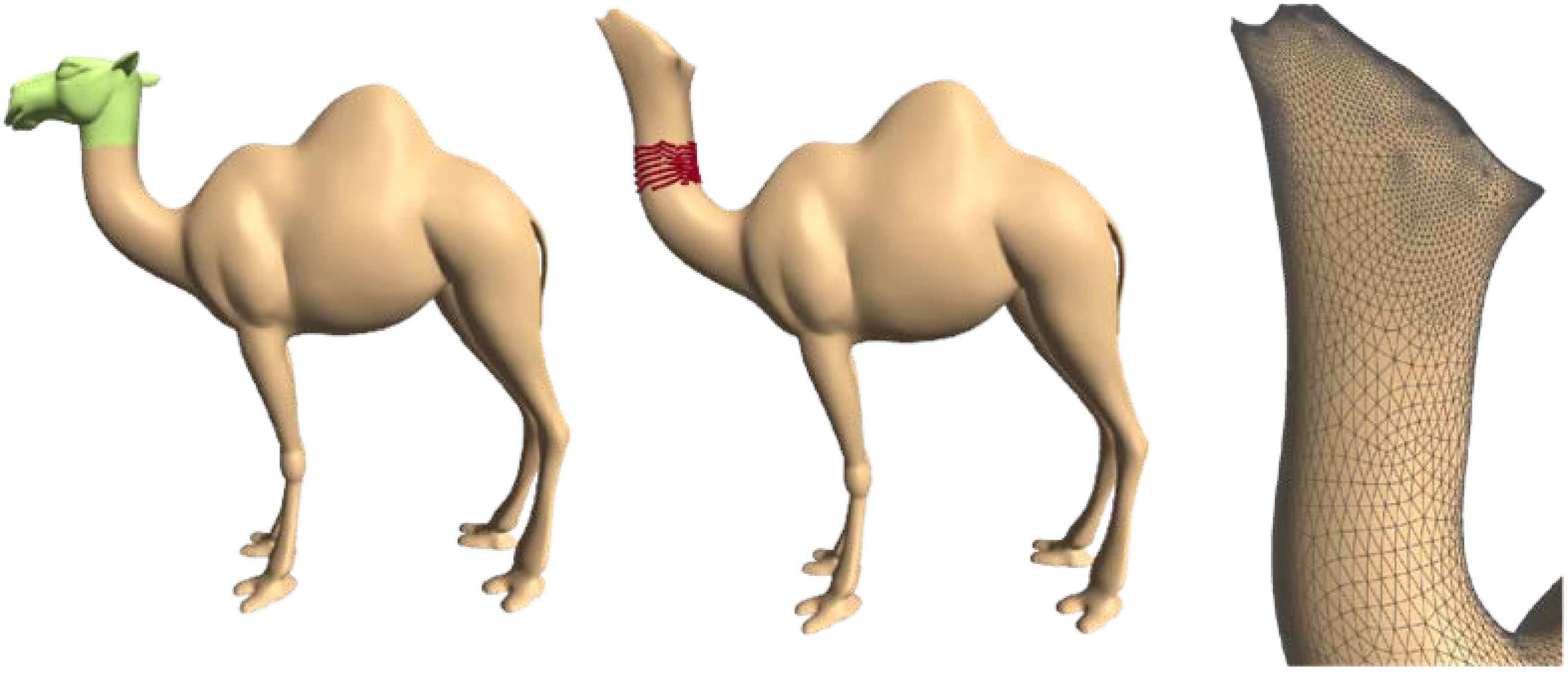
Applications

- Progressive geometry compression and streaming
- Hole filling
- Mesh editing



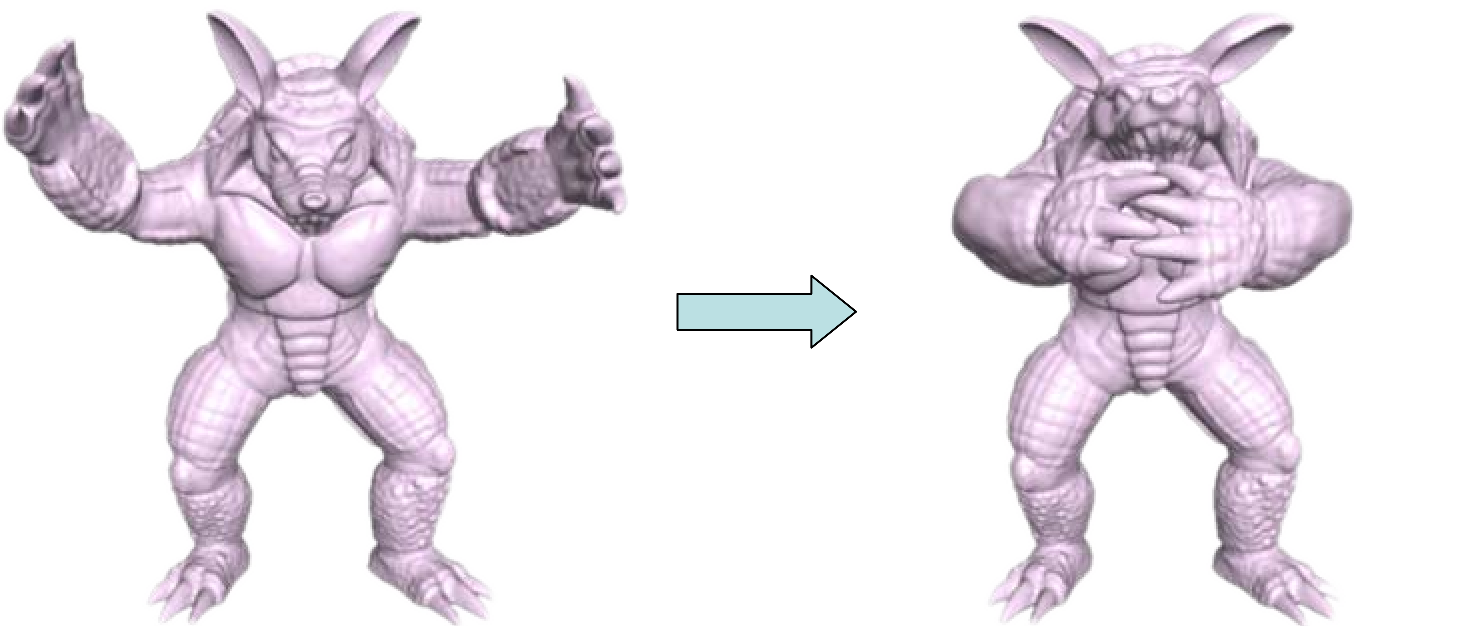
Geometry hidden in connectivity

“Least-squares Meshes”, Sorkine and Cohen-Or 04



Differential coordinates for editing

- Intrinsic surface representation
- Allows various surface editing operations that preserve local surface details



Why differential coordinates?

- Local detail representation – enables detail preservation through various modeling tasks
- Representation with sparse matrices
- Efficient linear surface reconstruction