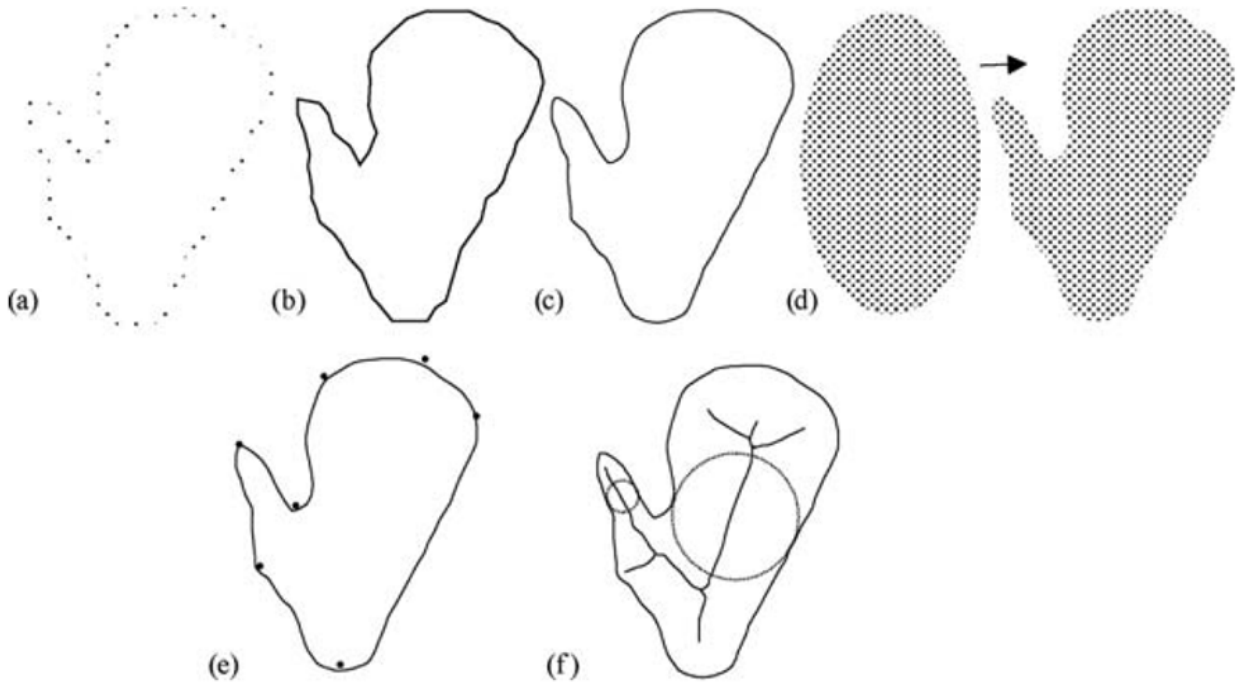
Skeleton Representation

2010

Object Representations

a) Point Clouds
b) Polygonal Meshes
c) Parametric Representations, e.g. Splines, Fourier (or spherical, in 3D) Harmonics
d) Voxel/Spatial Representations
e) Feature Representations
f) **Medial Representations**



Object Representations

c) Parametric Representations, e.g. Splines, Fourier (or spherical, in 3D) Harmonics

- Boundary representation via basis functions: splines, orthogonal function decomposition
- Represent an object of a particular topology by a smooth continuous mapping from the surface of a standard domain of the same topology
- This vector mapping function can be decomposed into orthonormal functions
- Derivatives available, normals/curvatures derived analytically

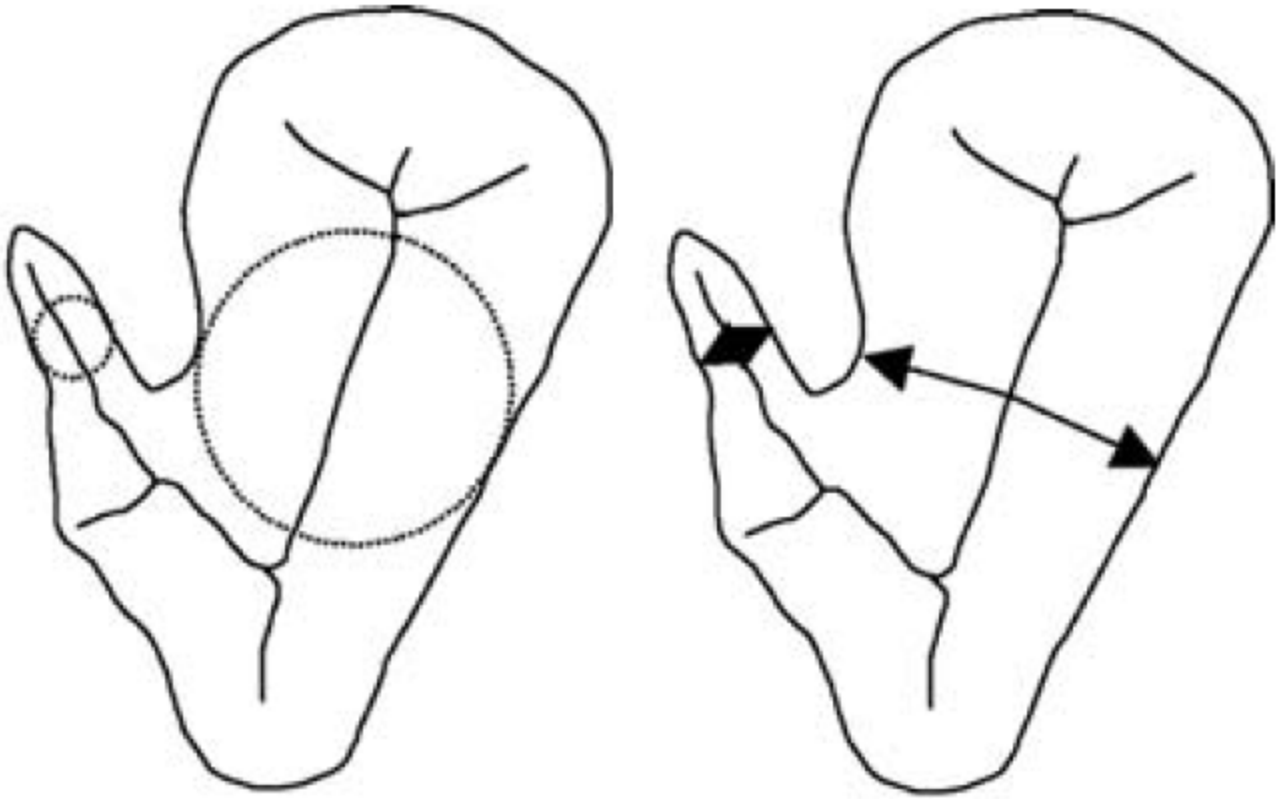
Object Representations

d) Voxel/Spatial Representations

- Simple, Hierarchical refinement, Efficiency
- Gives easy access both to boundaries and to interiors
- Large computer storage space required

Medial Representation

- Represent an object by a center locus (medial axis) and distance to the boundary (medial radius)
- Object ← union of overlapping bitangent spheres



Advantages

- an interior representation of the object and thus is subject to both geometric and mechanical operations applicable on the object's interior, such as bending, widening, elongation, and warping.
- provides rich topological and geometric information of an object.
- Effective/efficient for storage and for some applications.
  - Automatic object recognition
  - intuitive to non experts
  - …

Curve-Skeleton for 3D Objects

- Curve-Skeleton
  - Thin 1D representations of 3D Objects
- Applications
  - Visualization: virtual navigation, complex data visualization
  - Modeling/Robotics: reduced-model formulation
  - Graphics: animation (IK), morphing
  - Other Geometric Processing: shape comparison, shape decomposition
- Computation methods
  - Over image space – more hardware oriented
    - Thinning and Boundary Propagation
  - Over object space – more general and with richer math structure
    - Geometry Methods

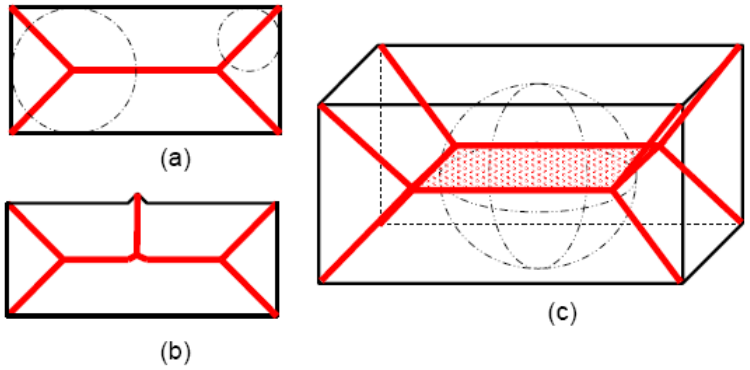
Definitions

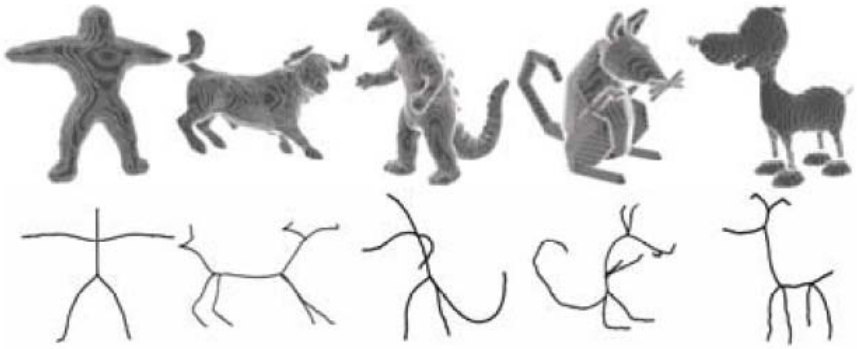
- Medial Axis:

-> A set of curves defined as the locus of points that have at least 2 closest points on the boundary

- In 3D → Medial Surface
- Grass-fire analogy:
  - Shape domain made entirely of grass
  - Set fire on all boundary points
  - Fire fronts meet and quench each other at the medial axis/surface
- Frequently interchangeably used as "Skeleton"
- The process of obtaining a skeleton

  → skeletonization





Mathematical Definition

Definition 1.2.1. Let S be a connected closed set in $\mathbb{R}^n$. A closed ball $B \subset \mathbb{R}^n$ is called a *maximal inscribed ball* in S if $B \subset S$ and there does not exist another ball $B' \neq B$ such that $B \subset B' \subset S$.

Let Ω denote an n-dimensional object and let $\partial\Omega$ denote its boundary. We are now ready to define the medial locus of Ω.

Definition 1.2.3. The *internal medial locus* of Ω is the set of centers and radii of all the maximal inscribed balls in Ω.

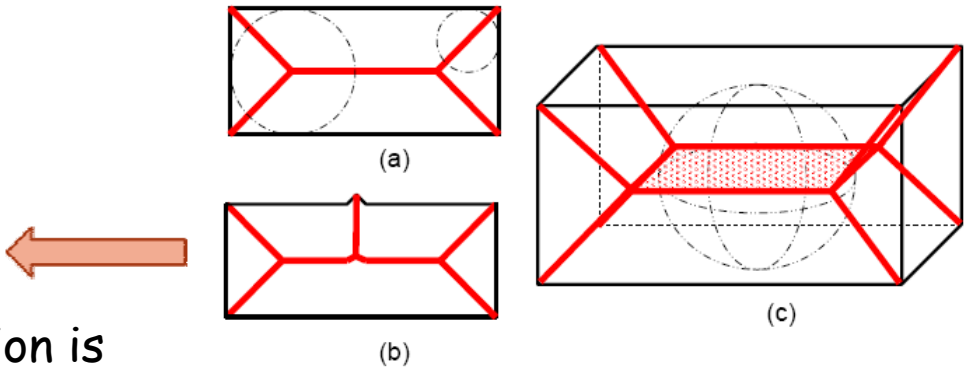
Definition 1.2.4. The *external medial locus* of Ω is the set of centers and radii of all the maximal inscribed balls in the closure of $\mathbb{R}^n / \Omega$.

Definitions (cont.)

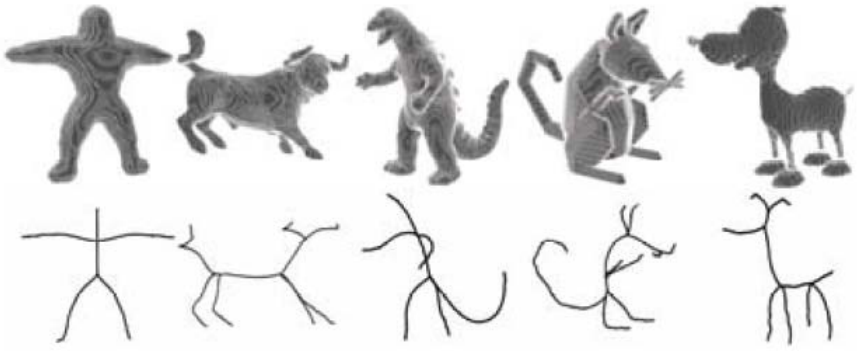
- Medial Axis is sensitive
  - see (b):

    small change on boundary can

    → large change on skeleton



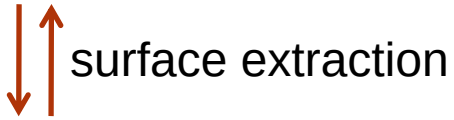
- A concise/simplified representation is desirable
  - Animation/Matching…
- More definition of curve-skeletons
  - T.K. Dey and J. Sun, "Defining and computing curve-skeletons with medial geodesic function", *Proc. SGP06*.
  - A. Lieutier, "Any open bounded subset of R^n has same homotopy type than its medial axis", *Proc. ACM SMI03*.
  - …



Smooth/Discrete Space

- Definitions formulated in continuous space
- in computer, geometric datasets are discrete
  → continuous space or discrete space

Data with geometric representation (mesh, splines)



Data with discrete representation (grids, voxel)

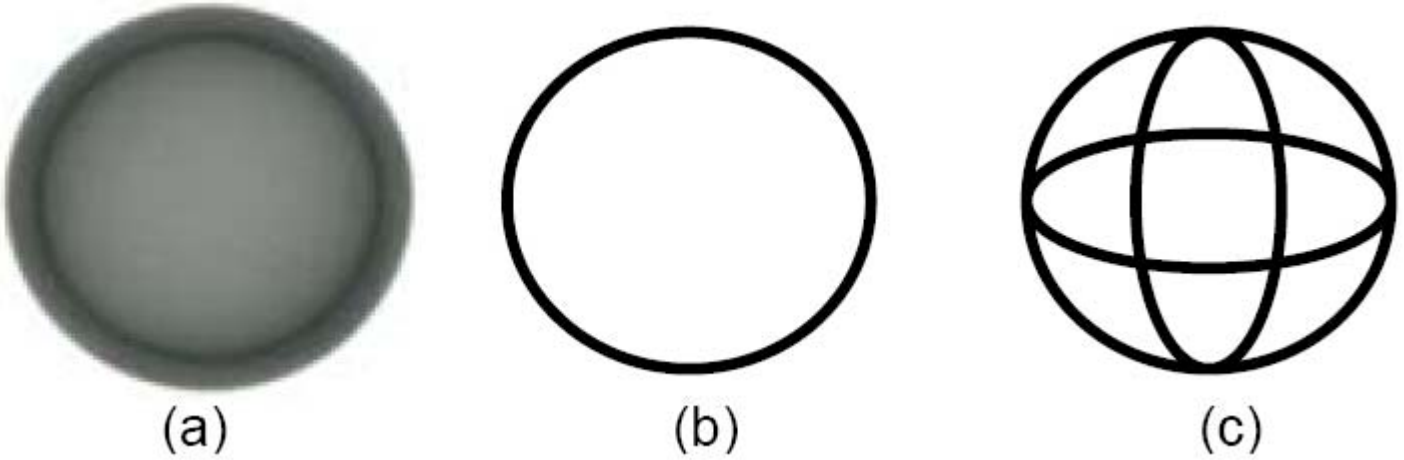
- In discrete space:
  - many efficient and intuitive algorithms
  - discretization → approximation error

Curve-Skeleton Properties

- Homotopic (topology preserving)
- Invariant under Isometry
- Reconstruction
- Thinness
- Centeredness
- Reliability
- Smoothness
- Component-wise differentiation
- Robustness
- Efficiency
- Hierachical

Curve-Skeleton Properties

1. Homotopic (topology preserving)
   - Homotopic to original object
   - Topological equivalence = same # of components, handles, (cavities)
   - one handle of the object ⟺ one loop on its skeleton
   - For cavities in 3-manifolds (relaxed definition):
   - The same number of connected components and at least one loop for each tunnel and cavity in the original object



Curve-Skeleton Properties

2. Invariant under isometric transformation

$$T(Sk(O)) = Sk(T(O))$$

- i.e. shape is invariant under rigid transformation
- important for automatic shape matching etc.

- Difficult if the discretization is not isometric invariant

Curve-Skeleton Properties

3. Reconstruction:
   - The ability to reconstruct the original object: $O - Rec(Sk(O))$
   - An intuitive indication of the quality of geometry abstraction
   - To improve the reconstruction quality:
     - Store more information in curve-skeleton nodes
     - Increase the number of branches in the curve-skeleton

4. Thin:
   - should be 1-dimentional :
     homotopic to R1 except on branch points
   - → an abstract graph with nodes and arcs

Curve-Skeleton Properties

5. Centered:
   - an important characteristic of a curve-skeleton
     → its centeredness within the object.
   - exact centeredness required/desired or not (in consideration of the sensitivity to noise) [related to robustness]

6. Reliable:
   - every boundary point is visible from at least one skeleton node
   - Ensuring all boundary points is reliably examined

7. Smooth:
   - Useful for virtual navigation (camera moving)
   - Ensuring small tangent variation across different segments

Curve-Skeleton Properties

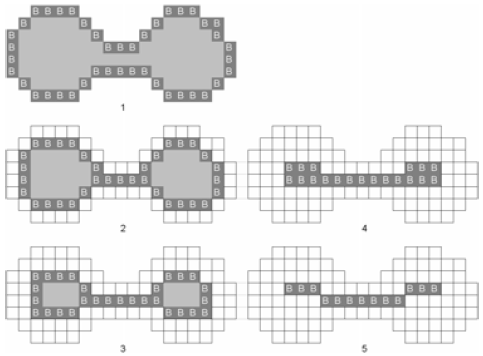
8. Hierarchy:
   - can generate a set of curve-skeletons representing hierarchical complexities
   - multi-resolution matching, partial matching

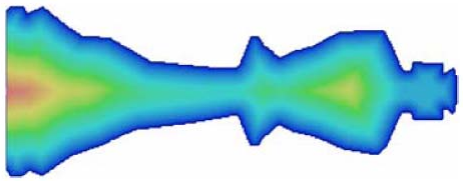
9. Efficient:
   - Real-time computation

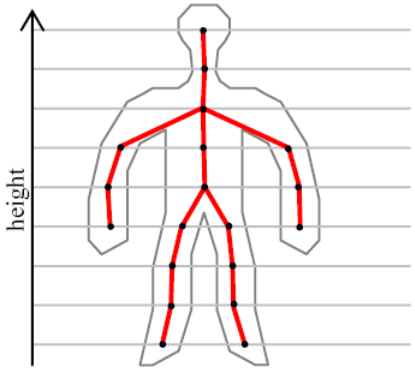
Algorithm Classes

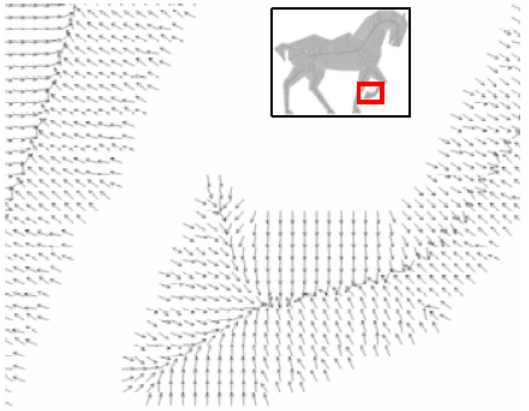
-- Following [N. Cornea, D. Silver, P. Min, “Curve-Skeleton Properties, Applications, and Algorithms”, TVCG2006]

- Thinning
- Distance field
- Geometric methods
  - Cores and M-reps
  - Reeb graph
  - …
- General-field functions
  - Potential field function
  - Electrostatic field function
  - …



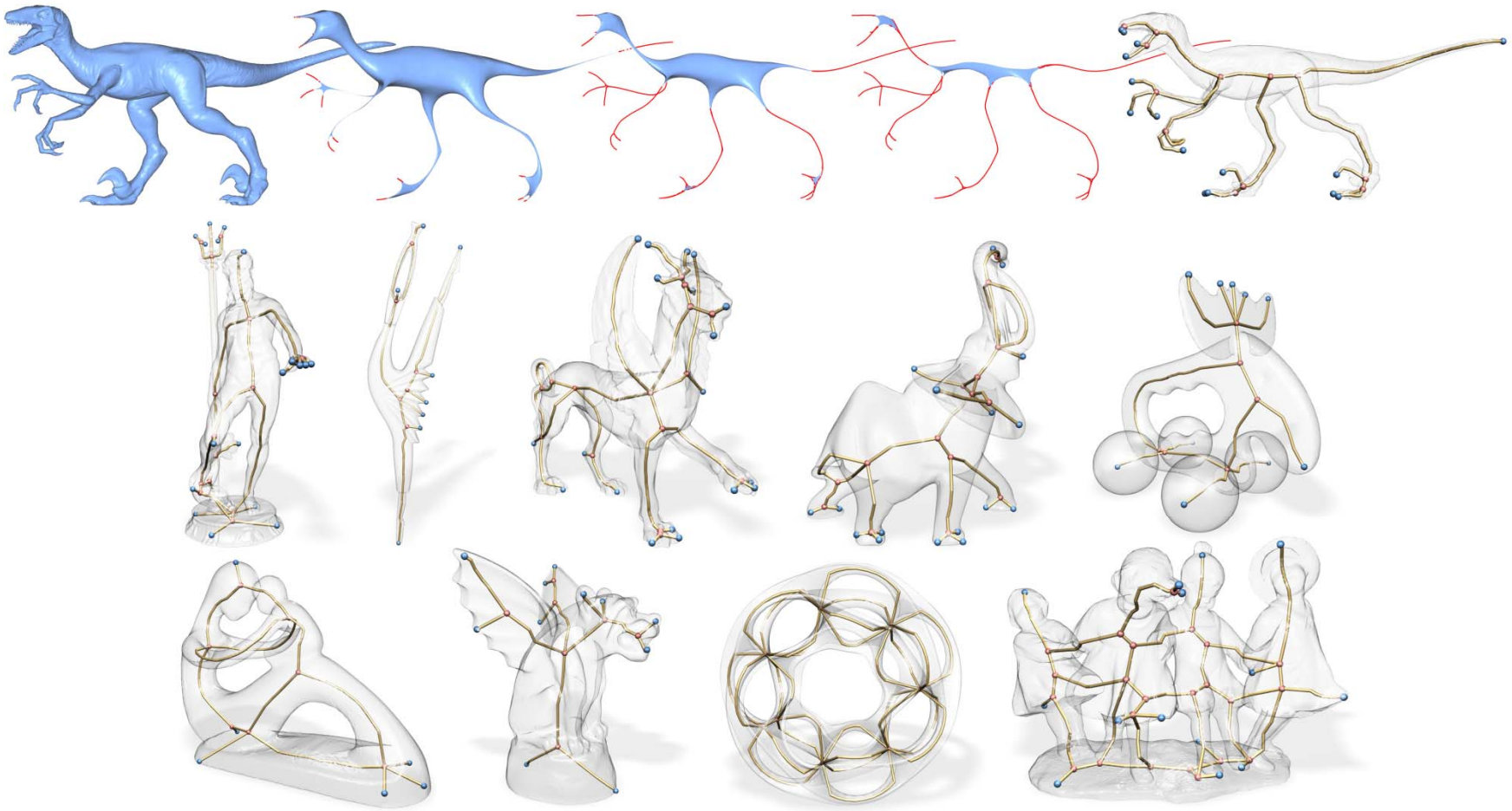






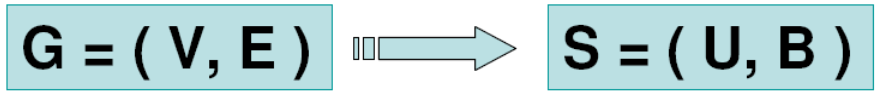
Skeleton Extraction by Mesh Contraction

— by O. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, T.-Y. Lee, in Proc. SIGGraph 2008



Algorithm Pipeline

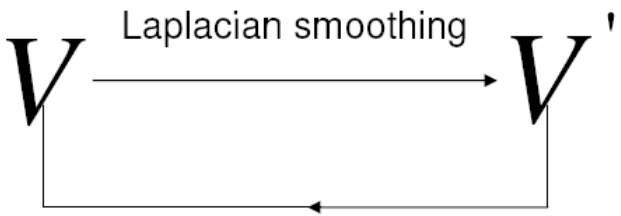
- Input: G=(V,E)
  - V → mesh vertices, and their positions
  - E → mesh edges
- Output: S=(U,B)
  - U → skeleton nodes
  - B → skeleton arcs (edges)



1. [Geometry Contraction]
   -- Contract the mesh into a 0-volume skeletal shape
2. [Connectivity Surgery]
   -- Convert the contracted mesh into a 1D curve-skeleton
3. [Embedding Refinement]

1. Geometry Contraction

- Basic Idea:
  - Contract the mesh geometry into a zero-volume skeletal shape
  - Move vertices along their approximate curvature normal directions
  - Remove surface details and noise by applying a Laplacian smoothing



1. Geometry Contraction

- Laplacian smoothing

$$\mathbf{L}_{ij} = \begin{cases} \omega_{ij} = cot\ \alpha_{ij} + cot\ \beta_{ij} & \text{if } (i,j) \in \mathbf{E} \\ \sum_{(i,k)\in\mathbf{E}}^{k} -\omega_{ik} & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases}$$

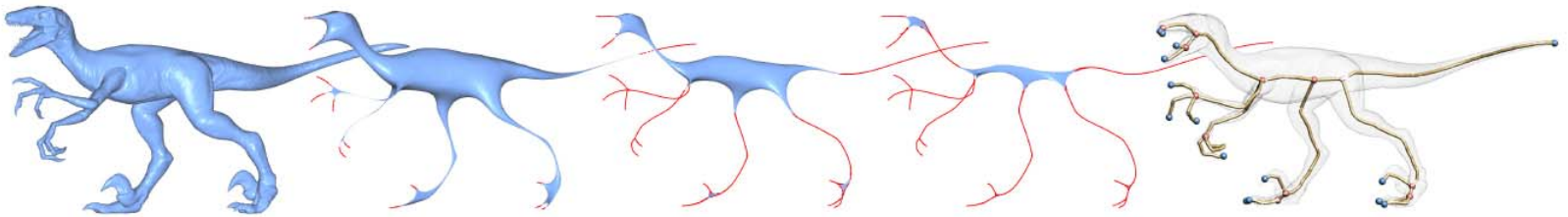
  - The Laplacian coordinate $\delta = \mathbf{LV} = [\delta_1^T, \delta_2^T, \ldots, \delta_n^T]^T$
    → approximate the (inward) curvature-flow normal $\delta_i = -4A_i\kappa_i\mathbf{n}_i$
    (local 1-ring area, local mean curvature, approximate outward normal)
  - Removing the normal component of V: $\mathbf{LV}' = 0$
    → A unique solution V'
  - To avoid degeneracy and approximate geometry:
    - constrain vertices to their current positions (soft constraints)
      a.k.a. attraction constraints
    - balance contraction constraints and attraction constraints

$$\| \mathbf{W}_L\mathbf{LV}' \|^2 + \sum_i \mathbf{W}_{H,i}^2 \|\mathbf{v}_i' - \mathbf{v}_i\|^2 \quad \Rightarrow \quad \begin{bmatrix} \mathbf{W}_L\mathbf{L} \\ \mathbf{W}_H \end{bmatrix} \mathbf{V}' = \begin{bmatrix} 0 \\ \mathbf{W}_H\mathbf{V} \end{bmatrix}$$

1. Geometry Contraction

More about contraction:

- solve once → one contraction
  - details filtered
  - same weights → may get stuck
- change weights → further contract
  - Large W_L → fast contraction
  - Intuition: vertices with smaller contracted 1-ring area to be attracted more strongly



1. Geometry Contraction

More about contraction:

- solve once → one contraction
  - details filtered
  - same weights → may get stuck
- change weights → further contract
  - Large W_L → fast contraction
  - Intuition: vertices with smaller contracted 1-ring area to be attracted more strongly

1. Solve $\begin{bmatrix} \mathbf{W}_L^t \mathbf{L}^t \\ \mathbf{W}_H^t \end{bmatrix} \mathbf{V}^{t+1} = \begin{bmatrix} 0 \\ \mathbf{W}_H^t \mathbf{V}^t \end{bmatrix}$ for $\mathbf{V}^{t+1}$,
2. Update $\mathbf{W}_L^{t+1} = s_L \mathbf{W}_L^t$ and $\mathbf{W}_{H,i}^{t+1} = \mathbf{W}_{H,i}^0 \sqrt{A_i^0 / A_i^t}$, where A_i^t and A_i^0 are the current and the original one-ring areas, respectively,
3. Compute the new Laplace operator $\mathbf{L}^{t+1}$ with the current vertex positions $\mathbf{V}^{t+1}$ using equation (1).

contraction↓

attraction↑

initial setting: $W_H^0 = 1.0 \;\; W_L^0 = 10^{-3}\sqrt{A} \;\; s_L = 2.0$

2. Connectivity Surgery

→ contracted mesh after Step-1:

- zero-volume (is visually a skeleton)
- still topologically equals to the original dense mesh

to a real 1D graph (skeleton)

Algorithm:

1) Define a cost function
2) Edge collapse based on its cost

2) Collapse

"Half edge collapse":

1. $(i \rightarrow j)$ merges vertex i to vertex j
2. remove all faces incident to the collapsed edge

(details: remove duplicated edges, but prevent "loop collapse")

2. Connectivity Surgery

1) Cost function:
   - Shape cost

     QEM (M. Garland and P. Heckbert, "Surface Simplification using Quadratic Error Metric", SIGGraph97)

     → Estimates distortion caused by an edge collapse
     by computing an error metric at each vertex

     QEM-like mechanism here → $\mathcal{F}_a(i,j) = F_i(\tilde{\mathbf{v}}_j) + F_j(\tilde{\mathbf{v}}_j)$

   - Sample cost

     long straight edges → (1) lose edge-face correspondence
     (2) centerness violation prone

     to penalize this:
     measure total distance of adjacent verts $\mathcal{F}_b(i,j) = ||\tilde{\mathbf{v}}_i - \tilde{\mathbf{v}}_j|| \sum_{(i,k)\in\tilde{\mathbf{E}}} ||\tilde{\mathbf{v}}_i - \tilde{\mathbf{v}}_k||$

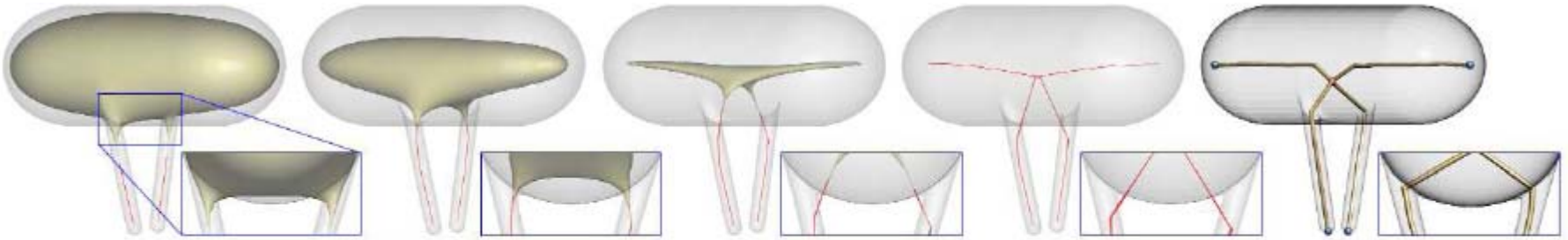
   - Total cost:

$$\mathcal{F}(i,j) = w_a\mathcal{F}_a(i,j) + w_b\mathcal{F}_b(i,j)$$

$$w_a = 1.0 \text{ and } w_b = 0.1$$

3. Embedding Refinement

Iterative contraction → skeletal shape off center or go outside the mesh (especially where adjacent object components have large differences in thickness and curvature)



The geometry contraction process does not always guarantee that the contracted shape is withinthe object. The rightmost image is the embedded result, showing that the skeleton is correctly centered.

- Idea:

1. move each skeleton node k to the approximate center of its corresponding local mesh region → shifting the skeletal nodes
2. merge junction nodes (if the merge → better centeredness)

3. Embedding Refinement

- Idea:

1. move each skeleton node k to the approximate center of its corresponding local mesh region → shifting the skeletal nodes
2. merge junction nodes (if the merge → better centeredness)

Detail:

centeredness = standard deviation of distances between the junction node and its neighboring points

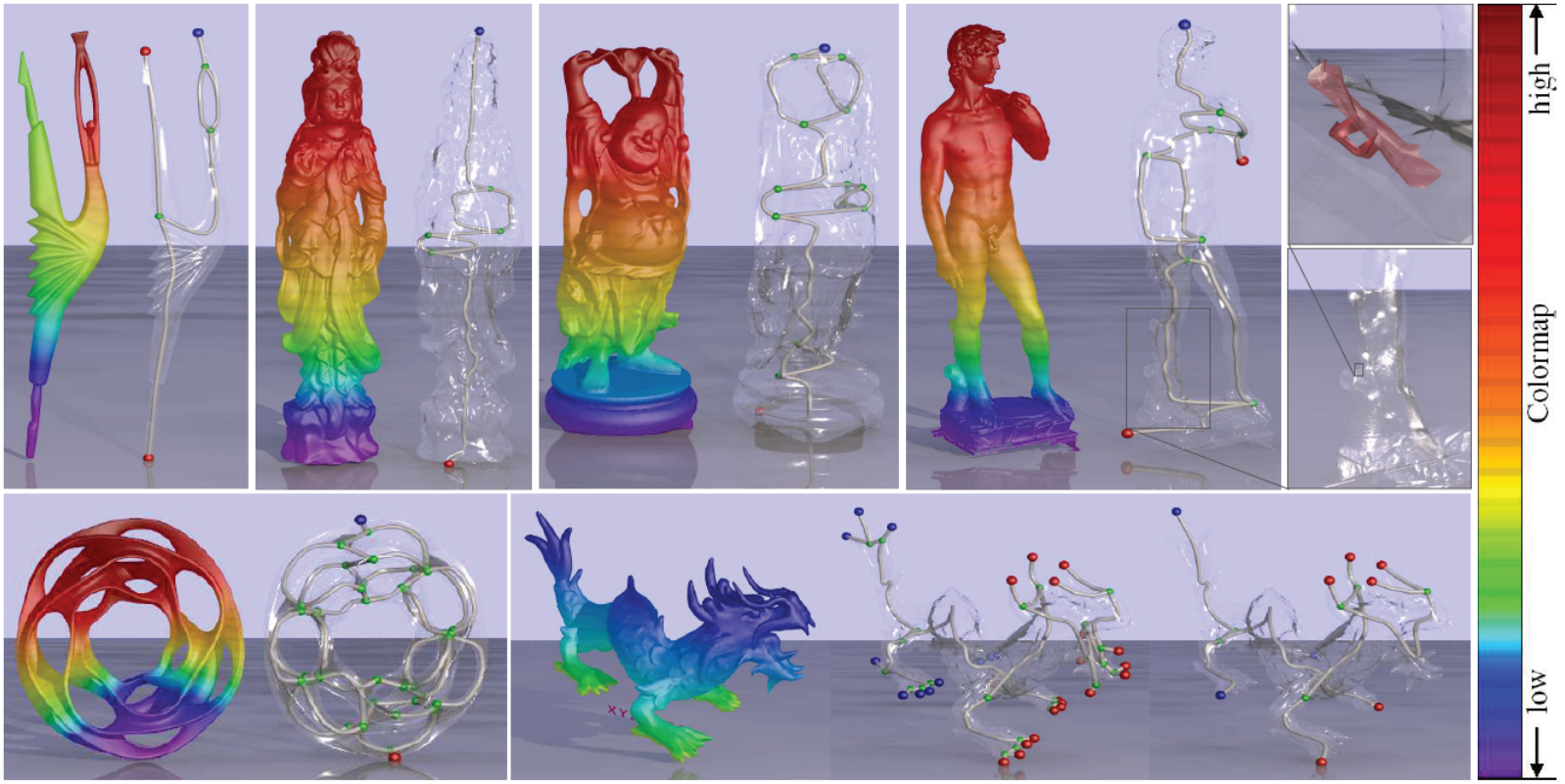
Property Discussion

- Directly apply on mesh surface data (using Laplacian operator)
- Homotopic
- Robust, insensitive to noise (due to the smoothing process)
- Invariant to rigid transformation (intrinsic, no global coordinates)
- Extra information preserved during contraction and collapse
  → reconstruction

Video

Robust On-line Computation of Reeb Graph: Simplicity and Speed

— by V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas, in Proc. SIGGraph 2007

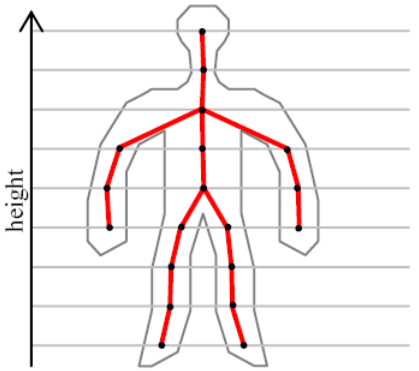


Contributions:

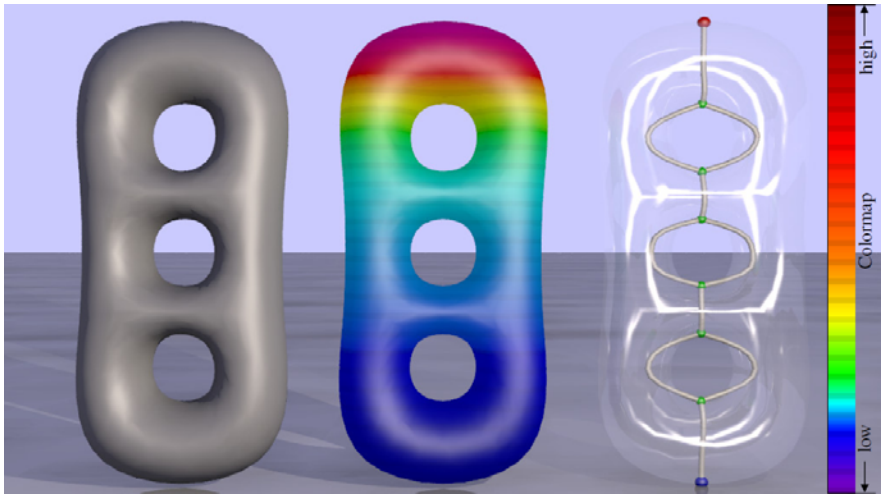
- Linear scalability to the surface vertex #
- Robustly handling noisy geometric data
  - Apply to topological denoise
- Applicable to non-manifolds, 3D, 4D and 5D simplicial meshes

Video1 Video2

Reeb Graphs (RG)



- Definition:
  - Given a simplicial mesh **M**,
  - with a piecewise linear (PL) function **F** sampled at its vertices
  - Level-set at a value **s** → set of points in **M** with **F** value equal to **s**
  - Connected components of the level-sets → contours
- RG construction:
  - by contracting contours of **F** to points
  - nodes → contours passing critical points, (associated with vertices)
  - arcs → connecting nodes
  - Path → monotonic seq. of arcs



- RG Properties

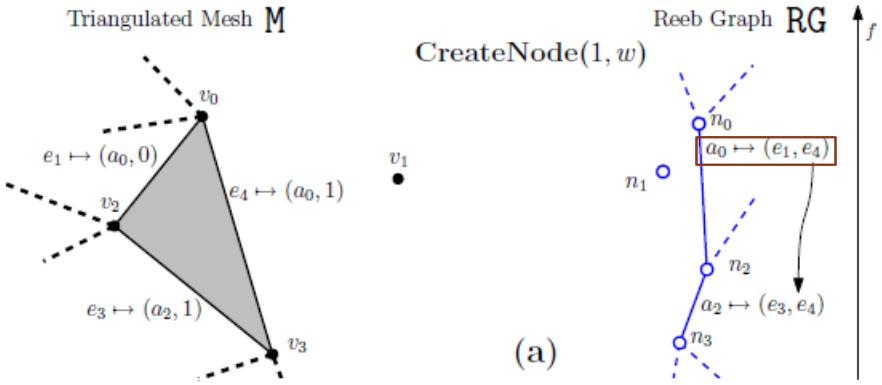
since topology changes only at critical points;

so an arc → a family of contours that do NOT change topology.

Reeb Graphs (RG) (cont.)

i. A simple robust algorithm to get RG
ii. Modifications to i. to
   1) Increase performance
   2) Compute an embedding
   3) Simplify RG

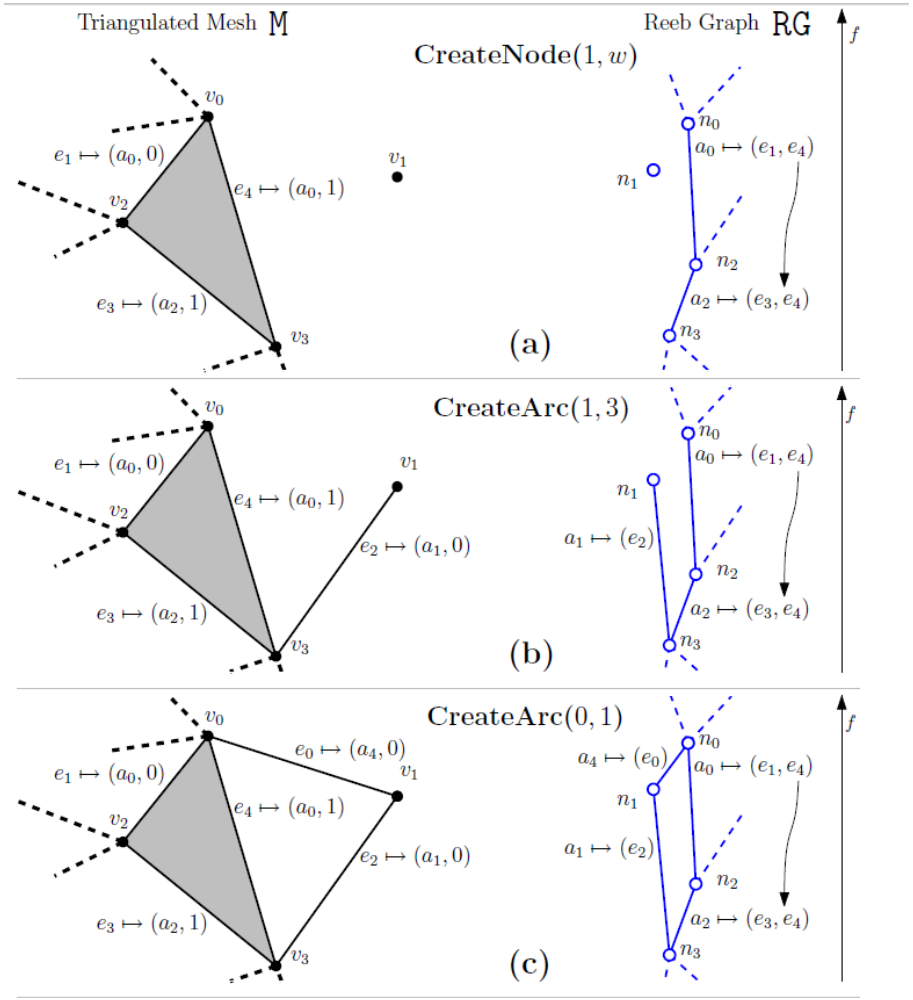
- Input Mesh: triangular mesh
- Input Function: provided/run-time computed
- Data Structure:

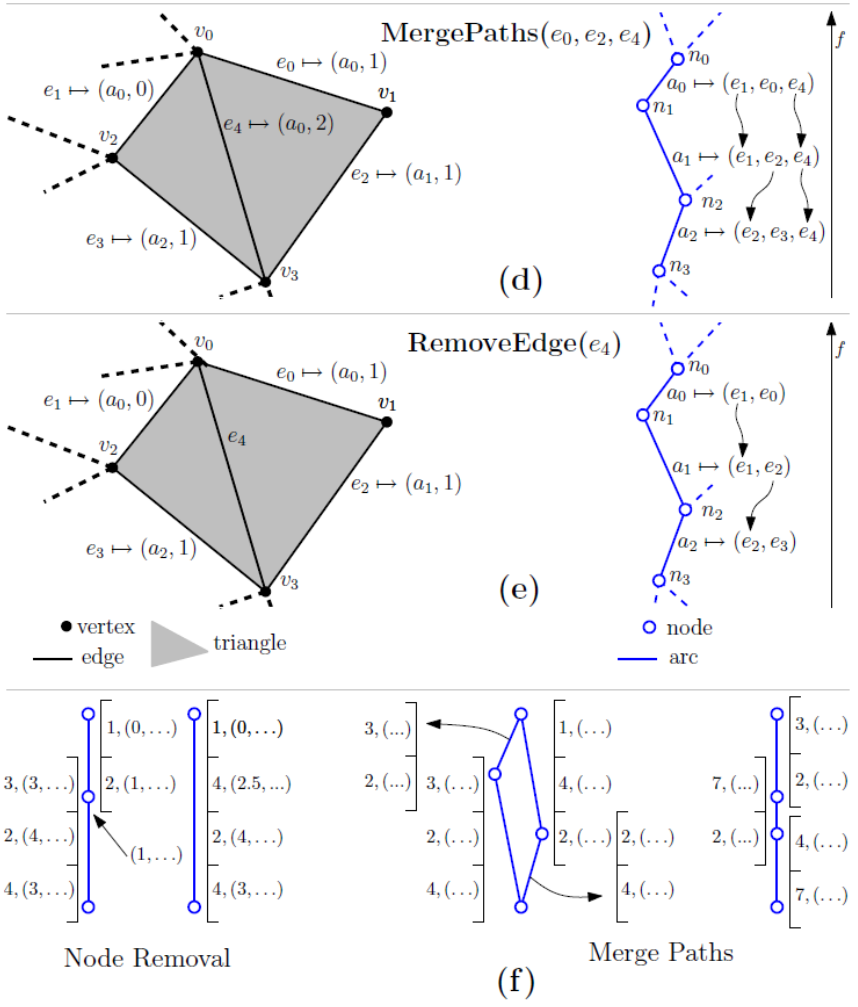


RG: { node → index to corresponding vertex
arc → a list of pointers to edges (that intersect the contours represented by this arc)

Mesh: edge e → a pointer to the highest (in F) arc in RG that has a pointer to e
intersects a family of contours that may span several arcs in RG

RG Construction Figure





RG Construction

Basic Operations (see illustration figure)

- CreateNode(i,w): add a new node with index-i, value-w to RG
- CreateArc(i,j): add a new arc connecting node i and node j to RG
- MergePaths(e_1,e_2,e_3): merge paths e_1, e_2, and e_3 in RG

General Algorithm:

- Parse the mesh, process at the same time
- Update the RG whenever a vertex or an edge is parsed
  - Read a vertex: -CreateNode
  - Read a triangle: -CreateArc , -MergePaths
- A final pass, and remove degree-two nodes

Example: in illustration figure (a)-(f)

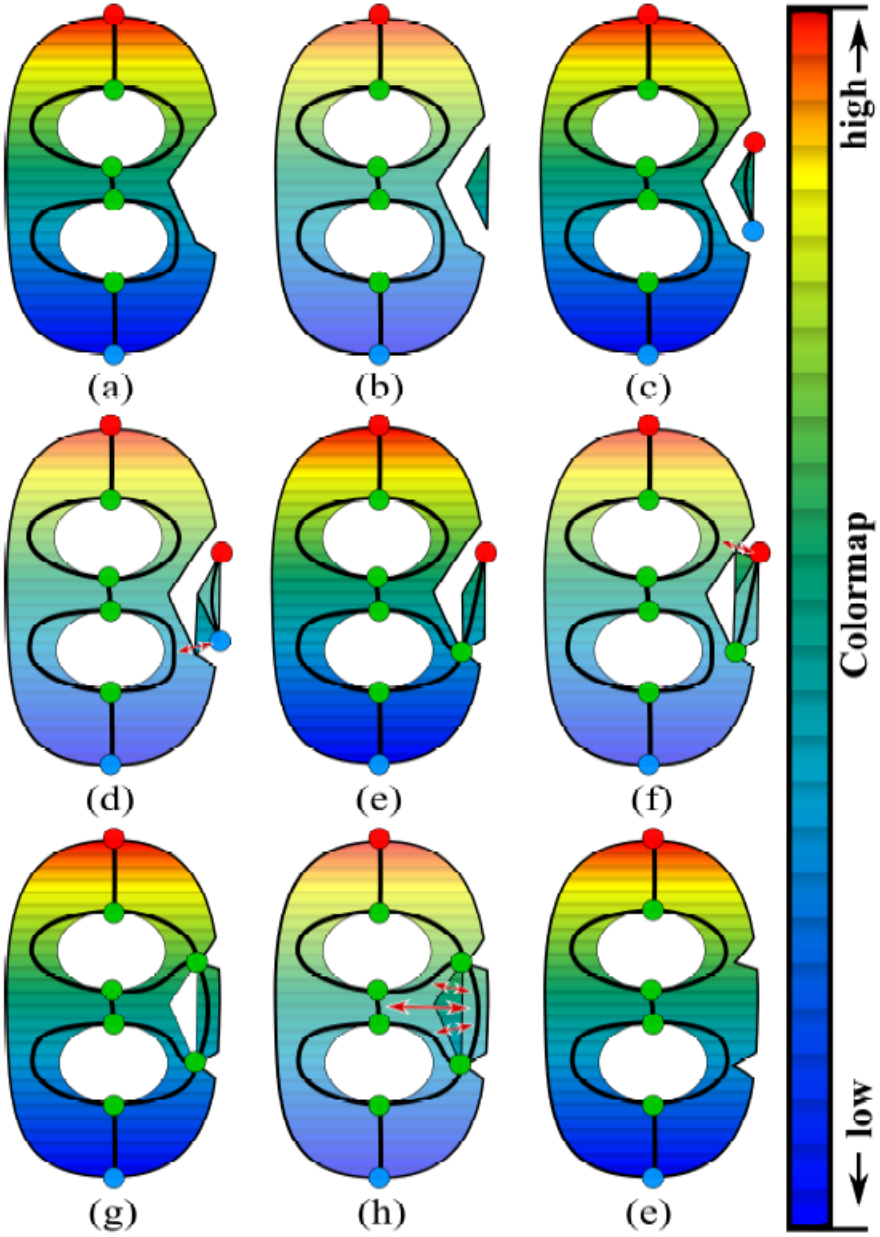
- Insert n1
- Insert arcs
- Merge
- Remove edge

Correctness

Proof by induction:

Adding a new triangle to M
→ Changing of RG

Init: empty RG



Optimizations and Embedding

RemoveEdge(e_1)

- Follow the path corresponding to e_1, and remove all its occurrences
- Used when
  - an edge (whose two adjacent triangles have already been processed) cannot be involved in any subsequent updates of RG [see illustration figure (d)]
  - an edge when both its vertices have been finalized (all incident triangles to a vertex have been processed)
- Reduce the storage requirements, and improve performance

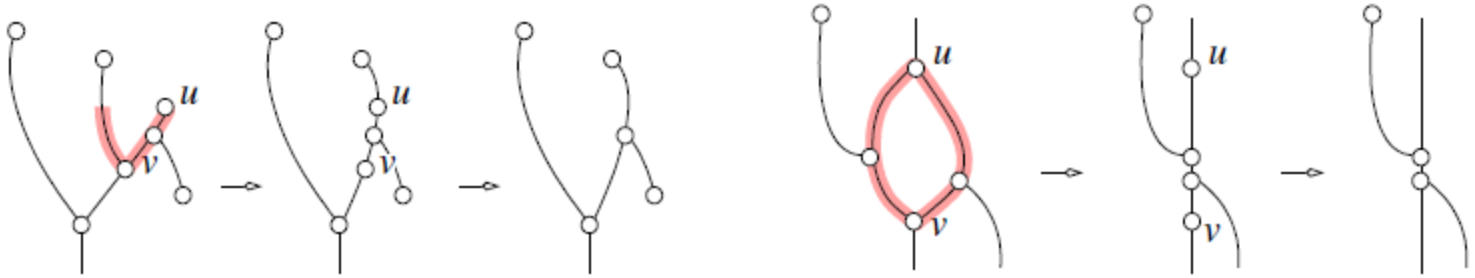
Embedding: -- based on the coordinates of the input vertices

- Store with each arc an ordered sequence of buckets
  - -- partition the range of function value spanned by the arc
- e.g. , function value in [0,100], want 200 sections:
  - if an arc spans values from 53.7 to 71.2
  - then we store buckets with separating values (53.5, 4, 54.5, …, 71.5)
  - each bucket contains NV (# of verts) and P (their average position)
- initiate buckets when CreateArc(),
- update buckets when (a) MergePaths() and (b) removing nodes of degree-2

Noise handling, Simplification and Hierarchy

On-line noise removal:

- Practical data are often noisy → RG needs noise removal
- Persistence based simplification [Edelsbrunner et al. 2000]
  - Remove extremum-saddle pairs whose persistence is less than a threshold



Simplification and Hierarchy → progressive visualization/processing

-- Loop persistence [Agarwal et al. 2004] (Pairing max-saddle, min-saddle, saddle-saddle → for measurement)

Computing:

1. Compute all possible persistence pairs
2. Cancel pairs of critical points in increasing order of persistence

   (see the above figure, (a) extremum-saddle cancellation, and (b) saddle-saddle cancellation)

Higher Dimensional Meshes

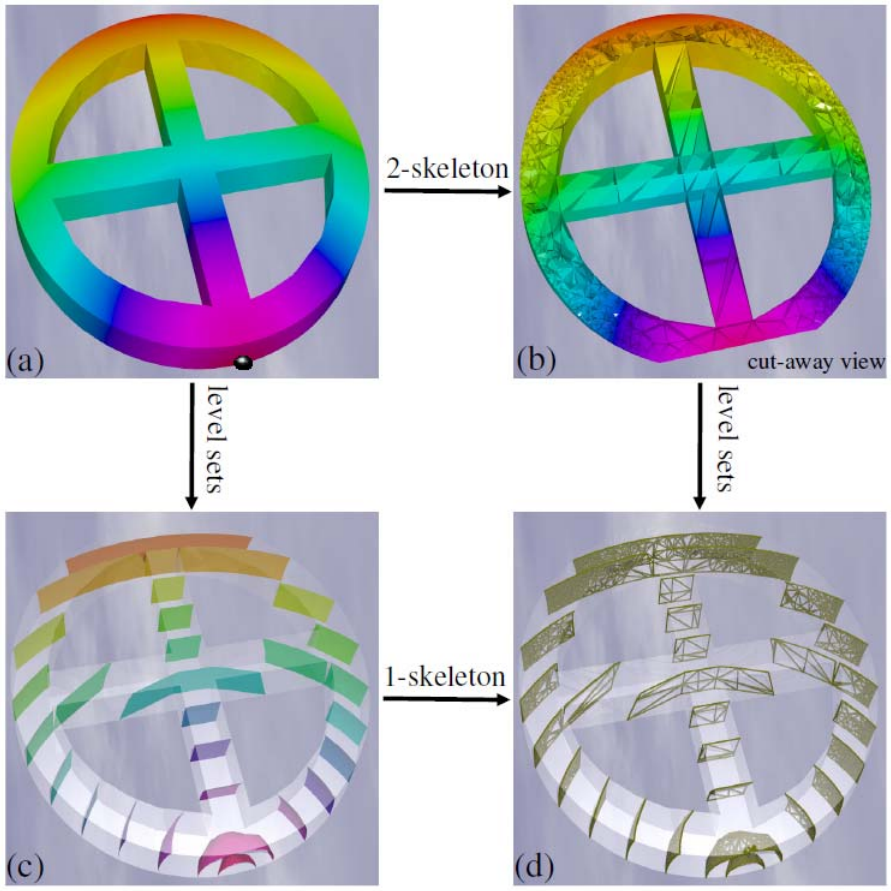
Theorem. *The Reeb graph of a function F, defined on the simplicial complex M, is identical to the Reeb graph of a function $\hat{F}$, the restriction of F to the 2-skeleton $\hat{M}$ of M.*

Proof:

- Contracting connected components of level-sets to points → RG
- A component of the level-set of F, defined on a d-manifold M:
  - This component C → (d-1)-dimensional
  - Boundary of C → (d-2)-dimensional
- The topology (connectedness) of a level-set component is determined by its 1-skeleton

---

- Extract a level-set of F defined on M and then extract its 1-skeleton
- Restrict F to M' and extract the level-set of F' defined on M'

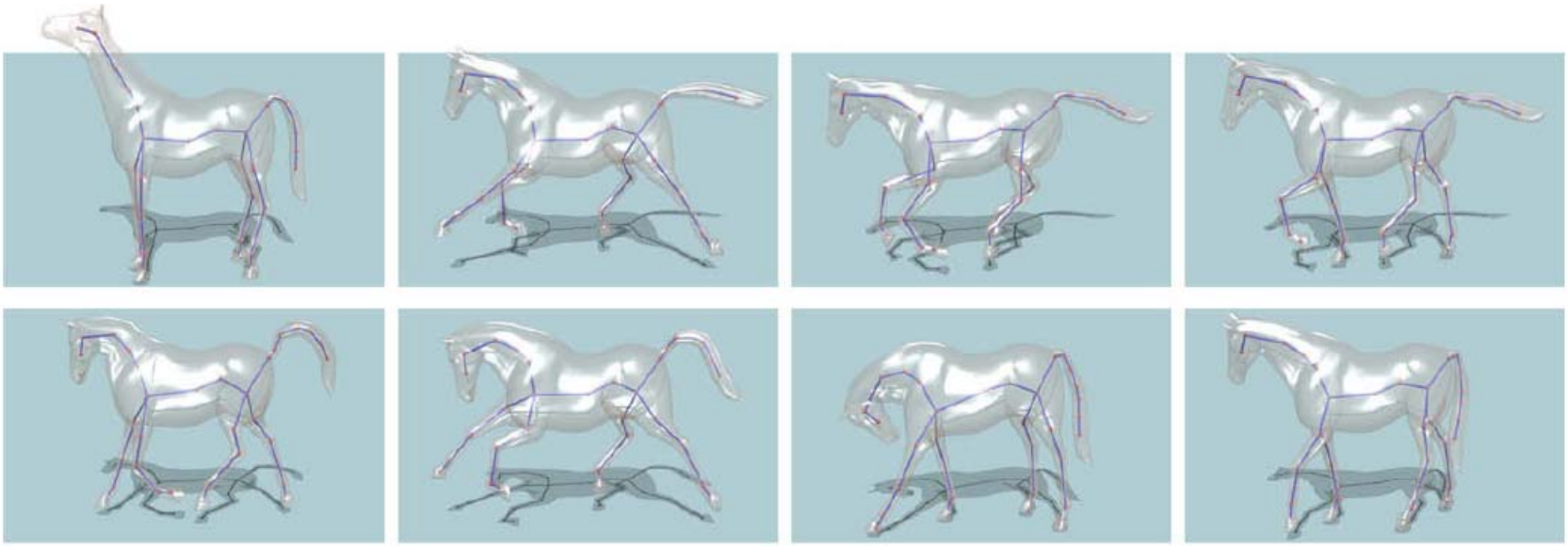


Efficiency

Model name	number of triangles	RG compute time max	RG compute time avg	Load time	Mem. usage min	Mem. usage max
cow	5.8Kt	0.01s	0.01s	0.01s	23KB	33KB
feline4k	8.3Kt	0.02s	0.01s	0.02s	66KB	94KB
indian goddess	19Kt	0.13s	0.10s	0.28s	47KB	47KB
turtle	38Kt	0.08s	0.06s	0.07s	4.1MB	4.1MB
heptoroid	40Kt	0.08s	0.06s	0.08s	1.5MB	1.5MB
dancer	50Kt	0.24s	0.11s	0.10s	1.5MB	1.8MB
elephant	50Kt	0.10s	0.05s	0.05s	0.1MB	0.1MB
ganesh	413Kt	1.1s	0.85s	0.8s	0.2MB	0.3MB
dragon	871Kt	1.8s	1.0s	0.8s	0.1MB	0.1MB
goddess2	1.0Mt	3.1s	2.3s	1.9s	0.3MB	0.3MB
happy-buddha	1.1Mt	2.8s	1.4s	5.2s	0.3MB	0.3MB
gargoyle	1.7Mt	4.2s	3.2s	3.5s	1.0MB	1.0MB
malay. goddess	3.6Mt	14s	9.5s	7.0s	0.5MB	0.7MB
asian-dragon	7.2Mt	23s	8.7s	7.4s	0.5MB	0.5MB
thai-statue	10Mt	34s	13s	23s	1.0MB	1.0MB
Lucy	28Mt	101s	48s	47s	2.1MB	2.1MB
David	56Mt	218s	108s	94s	2.1MB	2.1MB
StMatthew	371Mt	1448s	486s	311s	8.4MB	8.4MB

Harmonic 1-form based skeleton extraction from examples

— by Y. He, X. Xiao, H.-S. Seah, in Graphical Models 2009



Static Models vs Deforming Models

- Static skeletonization:
  - Single model, not for skinning (dynamic parameters such as transformations, joint locations, vertex weights are hard to obtain from one static shape)

- Example based skeletonization:
  - Possible to determine the joints
  - Intuition:
    - Many real-world deformations are isometric or near-isometric
    - Near-joint skins → similar Gaussian curvature, different mean curvature
    - Far-from-joint skins → similar Gaussian and mean curvature
    - So curvatures → indentify and locate joints

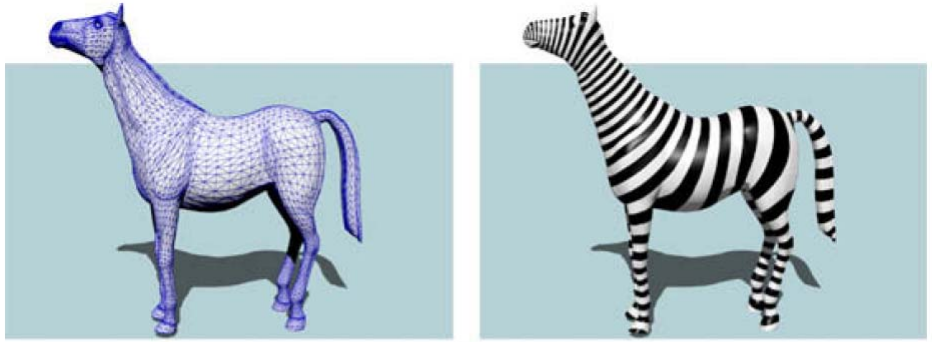
Deformations are caused by bending instead of stretching

- Surfaces have similar metric but different embedding (appearance in $\mathbb{R}^3$)

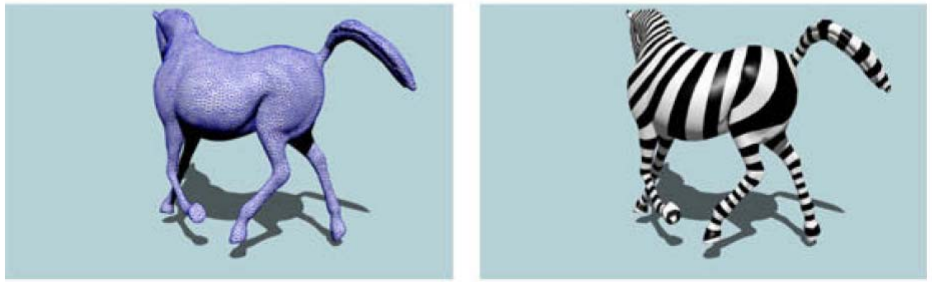
- Their Gaussian curvatures are similar but mean curvatures are not

Harmonic 1-form

- Challenges:
  - need mapping
  - need robust curvature estimation (discrete operators sensitive to noise and meshing)



(a) Reference pose (# of triangles = 16K)



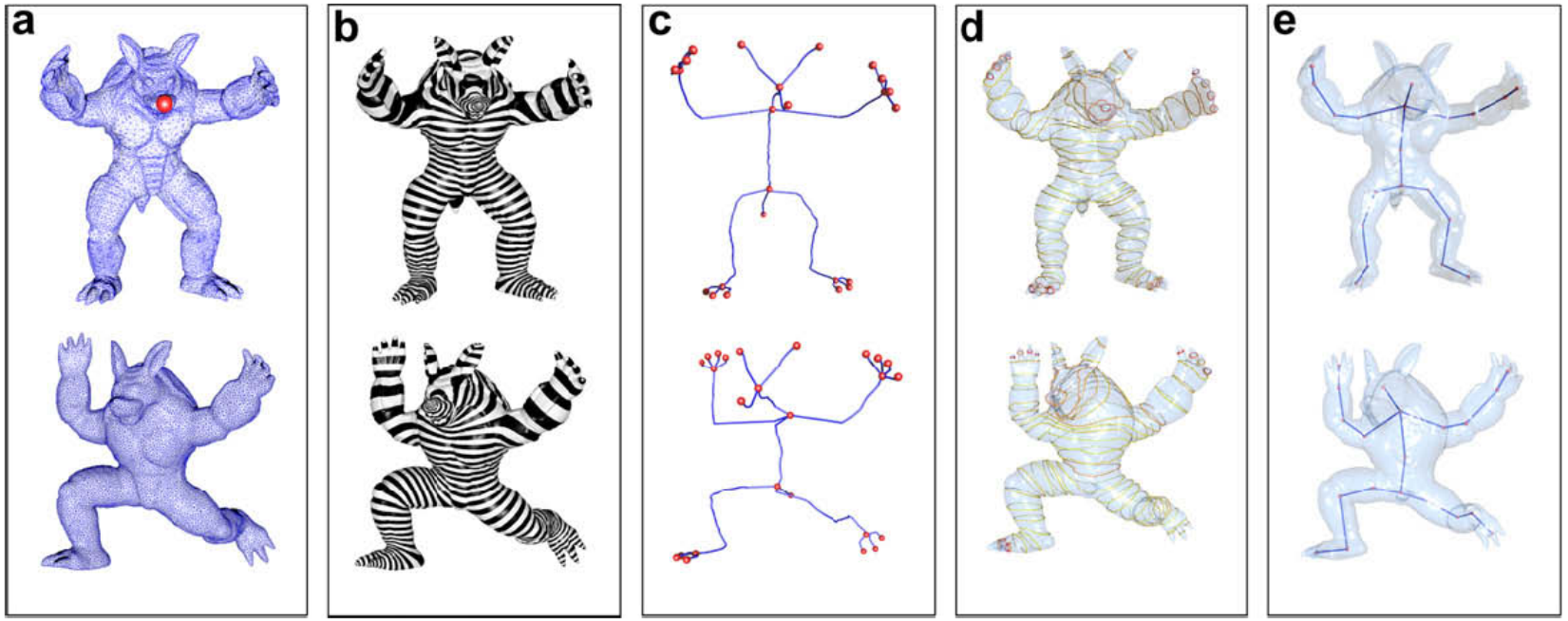
(b) Running pose (# of triangles = 30K)

- Harmonic 1-form

1. Determined by metric, invariant under isometry
2. Independent of meshing
3. Computational efficient
4. Symmetric on symmetric shapes

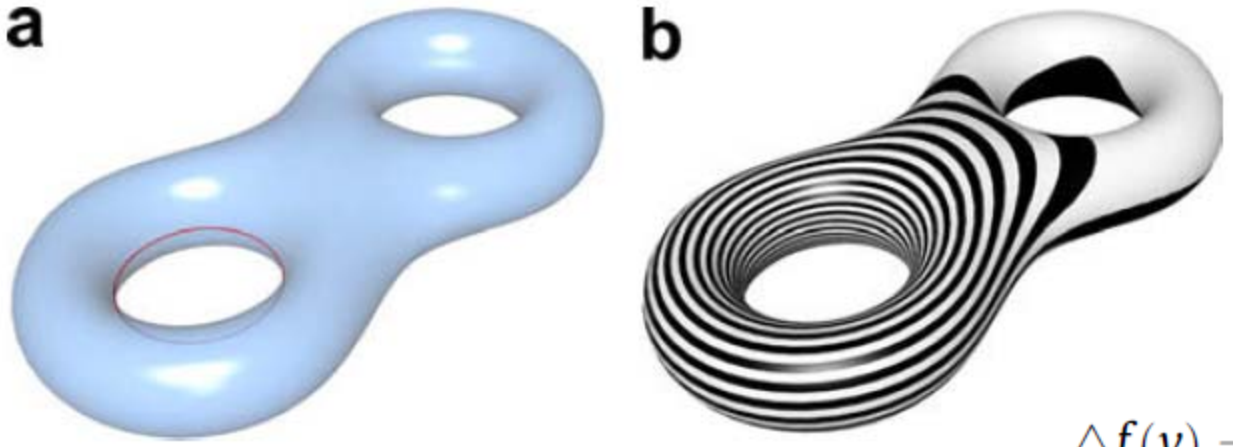
- Algorithm pipeline:
  1) Compute harmonic 1-forms
  2) Isocurve → skeleton-like Reeb graph of harmonic functions
  3) Identify initial locations of joins (by mean curvature changes)
  4) Refine joint locations by solving a constrained optimization problem

Algorithm



- Algorithm pipeline:
  1) Compute harmonic 1-forms
  2) Isocurve → skeleton-like Reeb graph of harmonic functions
  3) Identify initial locations of joints (by mean curvature changes)
  4) Refine joint locations by solving a constrained optimization problem

Computing Harmonic 1-form



$$E(f) = \int_M |\nabla f|^2$$

$$\triangle f(v) = \sum_{[v,w] \in M} k_{v,w} \omega([v,w]) = 0$$

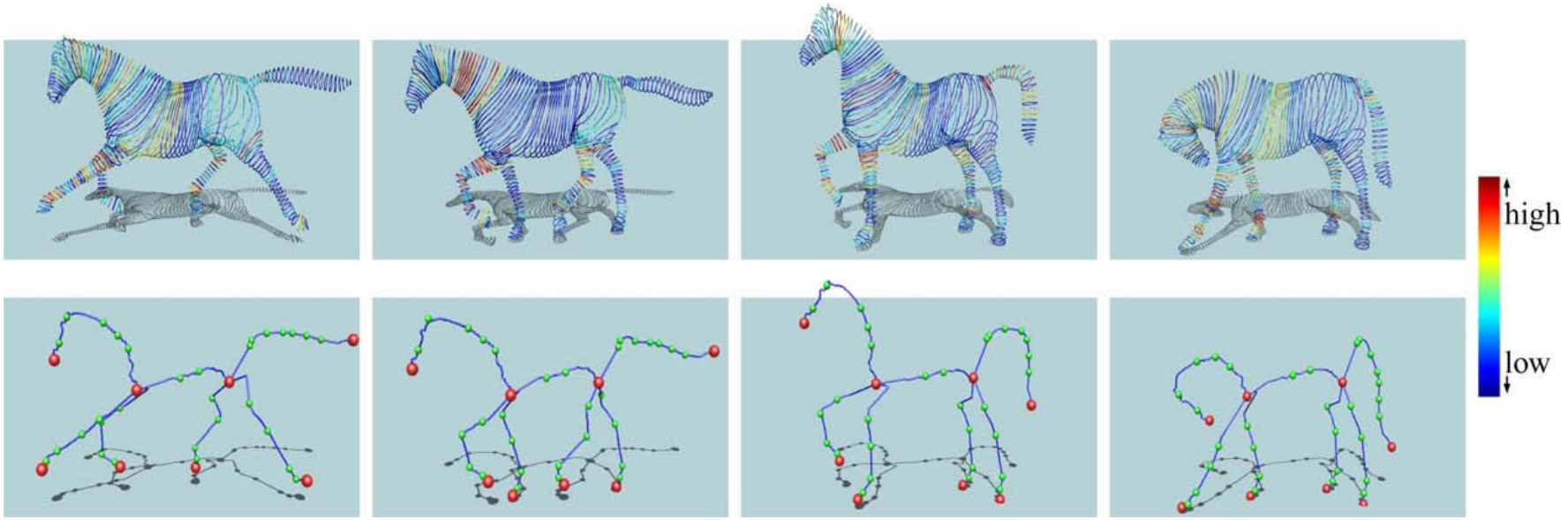
- Computing harmonic 1-form:
  1) Boundary condition: homology basis

     Or [Dong et. al. Harmonic functions for quadrilateral remeshing of arbitrary manifolds, CAGD2005]: pick a minimal vertex, solve a Poisson equation (mimics the curvature of the input mesh), then find the local extrema as the boundary vertices

  2) Solving harmonic functions using cotangent weights

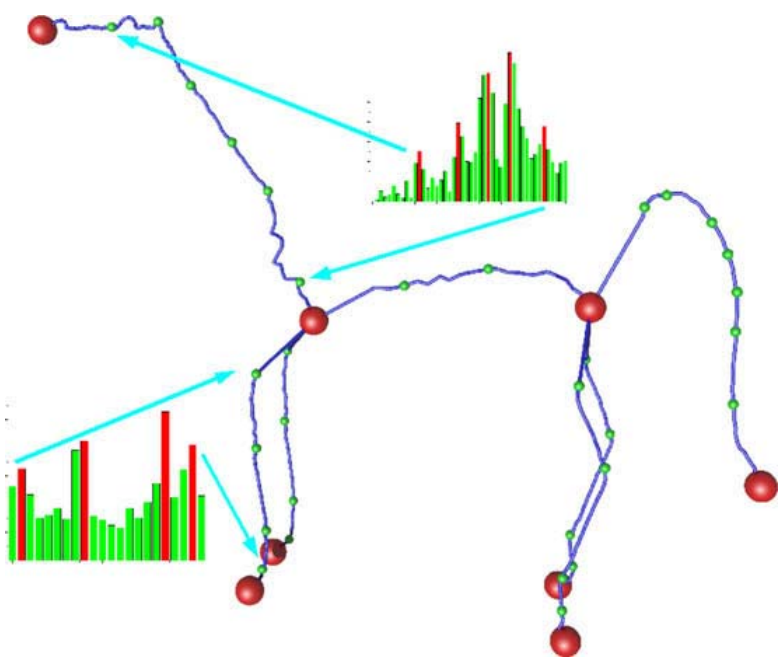
Computing Skeleton-like Reeb Graph

- Contracting the connected components of the isocurves:
  - Normalize the function value to [0,1]
  - Uniformly sample isovalues
  - Add in critical points
  - Sweep algorithm to connect them → RG



Identify Joints, Optimize their Locations

- Joints indentification ← mean curvature
- Optimizing the joint locations:
  - RG nodes → not anatomical skeletons
  - By examples: length-preserving:



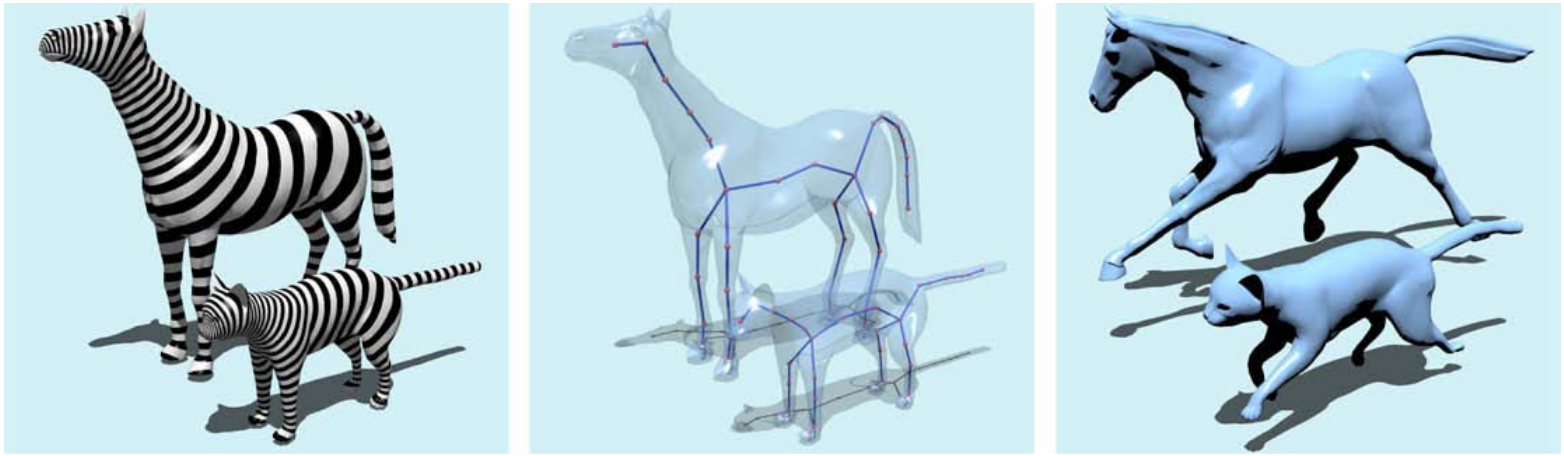
$$\arg\min_{J_k^i} \sum_{k\geqslant 1} \sum_i \left| d(c_k^i, J_k^1, \ldots, J_k^m) - d(c_0^i, J_0^1, \ldots, J_0^m) \right|^2$$

subject to the constraints

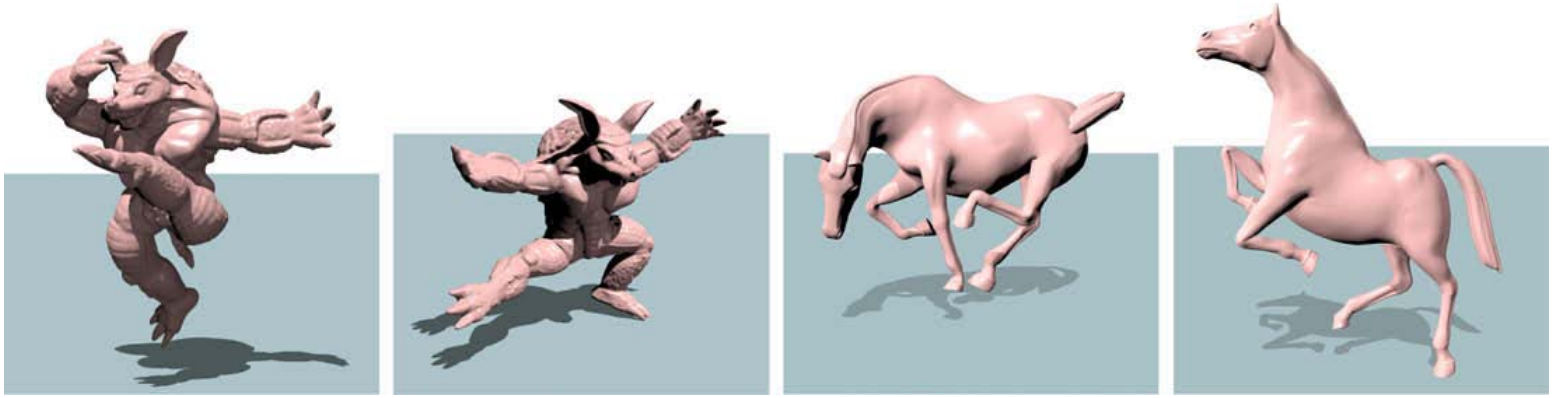
$$B_0^l = B_k^l \quad \forall l, k$$

Applications

- Skeleton Transfer

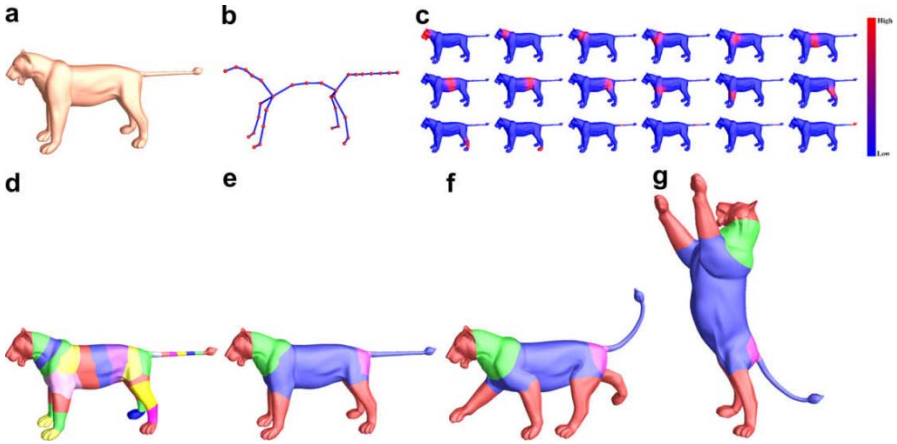


- Pose space deformation



Applications (cont.)

- Skeleton-driven Shape Segmentation



- Pose-invariant Shape Signature
  - Signature invariant of "pose"
  - Tolerate rotation/scaling/translation/deformation
  - Defined over iso-contour
  - Normalized length (NL) + Integration of Gaussian Curvature (IGC)