Skeleton-Based Seam Computation for Triangulated Surface Parameterization

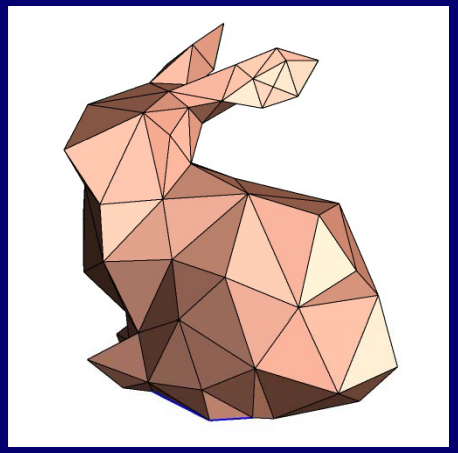
Shi-Min Hu

Tsinghua University, Beijing

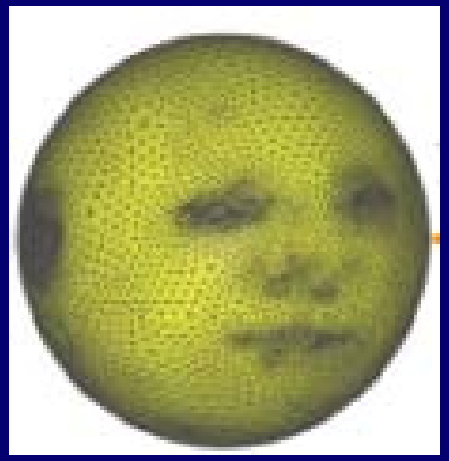
1. Parametrization and seam computing

- Triangle meshes and parameterization(参数化)
  - Due to flexibility and efficiency, triangle meshes have been widely used in the entertainment industry to represent arbitrary surfaces for 3D games and movies during the last few years.
  - Many new mesh techniques have been developed for different applications. In these techniques, parameterization is a key issue.
  - Parameterization provides mapping between meshes and a domain

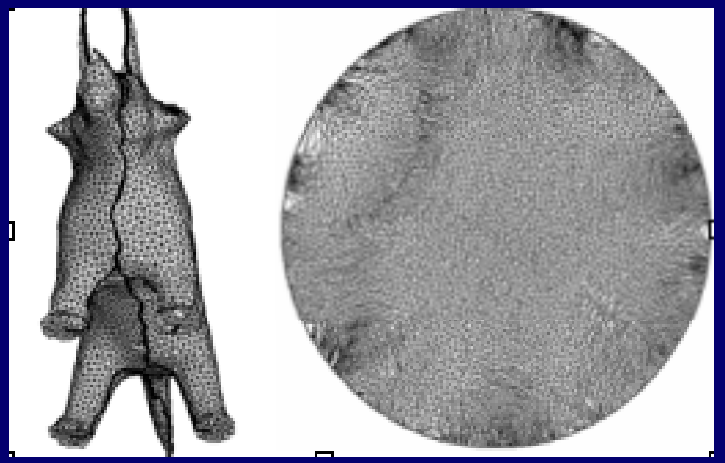
– Parameterization methods can be grouped into three categories(类别), depending on whether the domain is a polyhedron(多面体), sphere or plane.



Base-mesh



Spherical

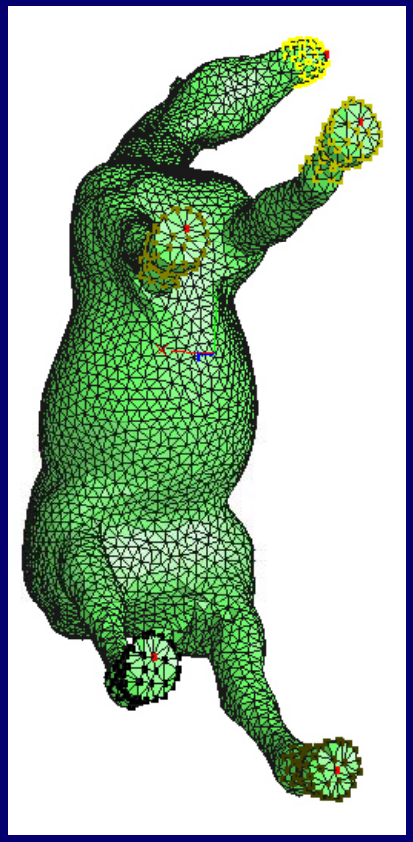


Planar

- Seam(缝) computing
  - Our goal is to map an arbitrary topology(拓扑) surface mesh to a single chart with low distortion(扭曲). Theoretically, any closed surface can be opened into a topological disc(圆盘) using a set of cut edges making up a “seam”. Cutting along the seam, we can get a disc-like patch.

– Unless the surface is developable, parameterization using a single chart inevitably(不可避免的) creates distortion(扭曲), which is especially great where protrusions(突出物) of the surface are flattened into the plane. (such as fingers of a hand and horses' legs)

- It’s found that to reduce the distortion, it is important for the seam to pass through the various so-called “*extrema*”(极值点).
- Although extrema can be found accurately, it is still difficult to guide the seam through these extrema.



The objective(目标),

is to consider seam computation with given extremal vertices (we called extrema(极值点)).

2. Related Work

- To seek a good seam, we should follow two strategies(策略):
  - The seam should pass through all extremal vertices.
  - The seam's length should be minimum(最小).

- For given extremal vertices, To find a minimum length seam connecting all extremal vertices comes down to the Steiner Tree problem in Graph Theory. For a given weighted graph, it's a NP-complete problem.
- Two main approximated algorithm for it are Minimum spanning tree (MST) method and Greedy algorithm. Sheffer and Gu use those two methods for seam computing respectively.

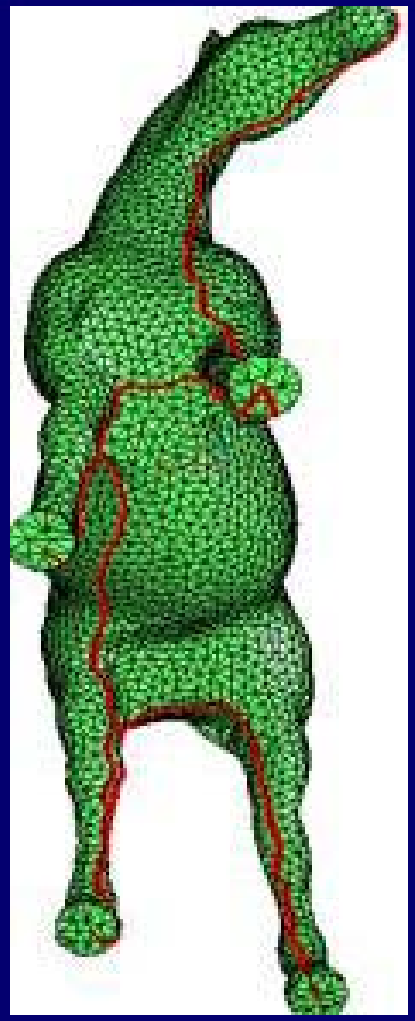
- Gu etal. “Geometry Image” (几何图像)
  - Gu et al. first find an initial cut that opens mesh *M* into a disk.
  - Extrema was found by utilizing the shape-preserving feature of Floater's parameterization.
  - when a new extremal vertex is detected, the shortest path between the current seam and the new extremal vertex is added to the seam.

- Sheffer, “Spanning Tree Seams”
  - Sheffer detects extrema by searching for vertices with high curvature(曲率).
  - Sheffer approximates Steiner tree by the minimum spanning tree (MST).

- Two points concerned in existing methods:
  - Greedy method：This incremental method depends heavily on the sequence of adding extremal vertices.
  - MST method：can’t introduce steiner points whose degree are greater than 2, then seam will pass through most extrama multiple times. This will lead to bad result。

3. Outline: The Skeleton-based method

- Idea:
  - Let’s look at the horse Model, the most obvious extrema are located at the ends of the four legs and the head, The seam computed using the MST method is poor as the MST passes through most extrema more than once.



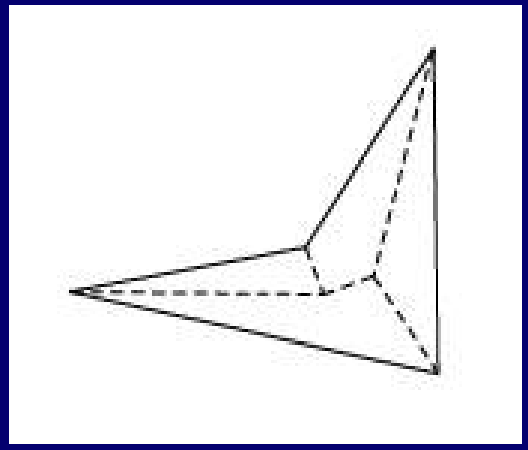
– However, extremal vertices are always at the ends of protrusions(突出), and so it is reasonable to require that the seam should pass through each extremal vertex as few times as possible. It will be best if all extremal vertices are leaves on the seam.

- For this reason, we constrain the Steiner tree to be a "full component" of the mesh, and
- Suggest a new method to compute an approximation to the minimal full component Steiner tree, deriving it from the straight skeleton.

A full component is a subtree in which each terminal is a leaf.

- Skeleton
  - The *straight skeleton* of a planar straight-edged graph G is obtained as the interference pattern of certain wavefronts propagated from the edges of G.

– If G is a simple polygon, the part of the straight skeleton interior to G is a tree whose leaves are the vertices of G. If all vertices of the polygon are terminals, the interior straight skeleton must be a full component Steiner tree.



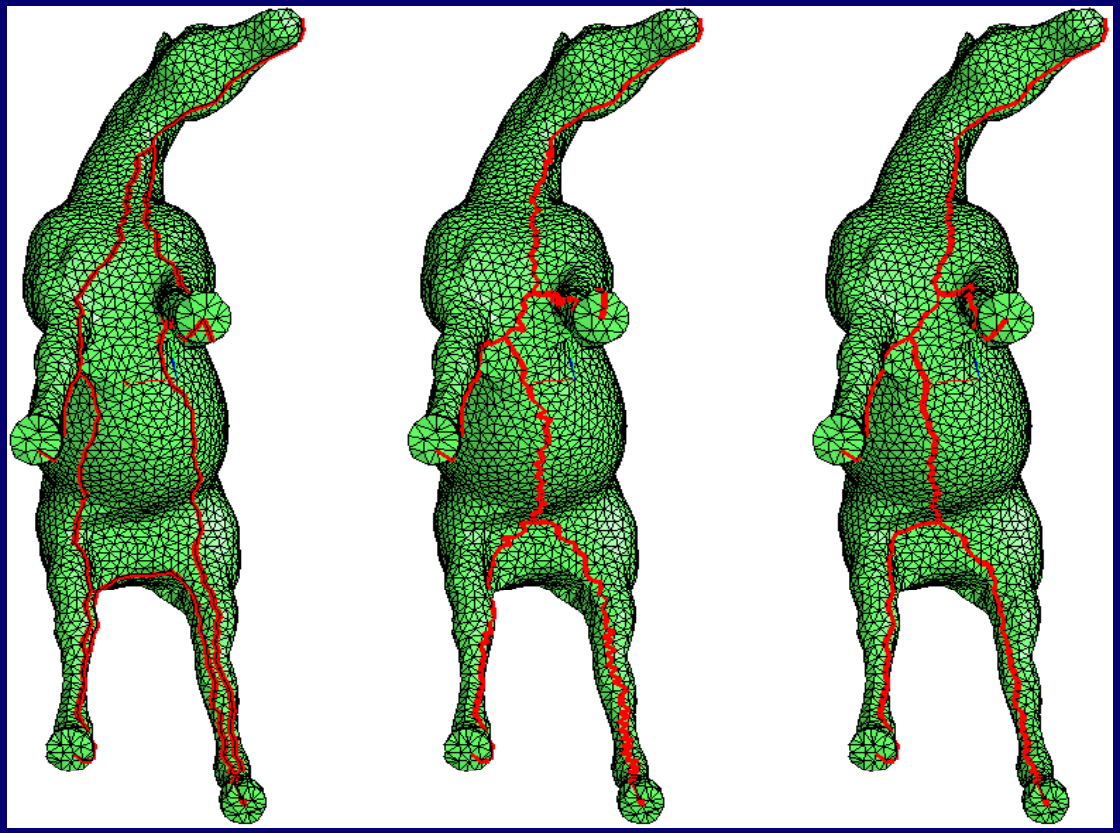
– Thus, the main idea of our method is to extend the concept of straight skeleton from the plane to a 3D triangle mesh. To do this, we must solve two problems:
  - First, a shortest *tour* that visits all terminals is found.
  - Secondly, the tour is shrunk to produce the skeleton which is a full component Steiner tree of terminals.

- Preprocessing(预处理) before computing seam:
  - For surfaces of genus(亏格) greater than zero, we use Gu's algorithm to compute an initial seam. Cutting along the initial seam, the surface is opened into a disc.
  - We find the shortest path between each pair of terminals (extremal vertices) .
  - Our algorithm assumes that there are at least two terminals. If there are only two terminals, we just use the shortest path between them as the seam.

- Overview of algorithm
  - Find the shortest tour that visit all terminals
  - Shrink(收缩) the tour to a skeleton
  - Straighten(拉直) the skeleton

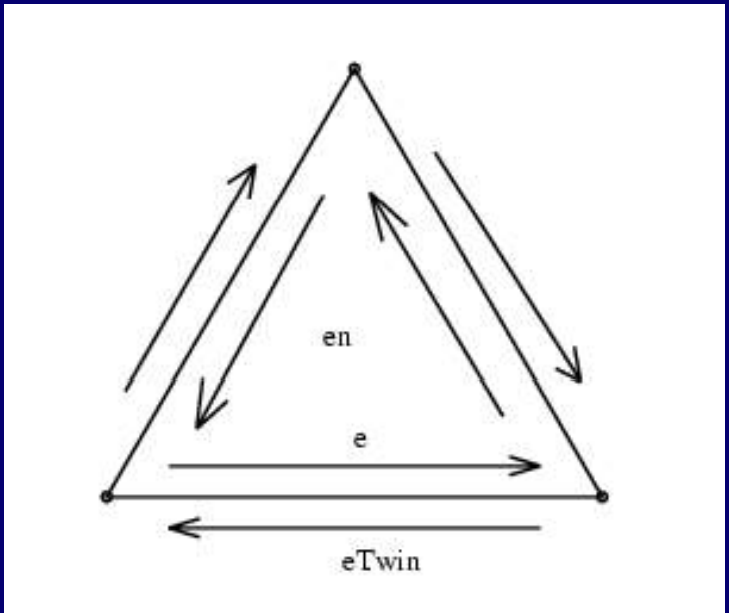
(see illustration in next page)

- Illustration of new algorithm

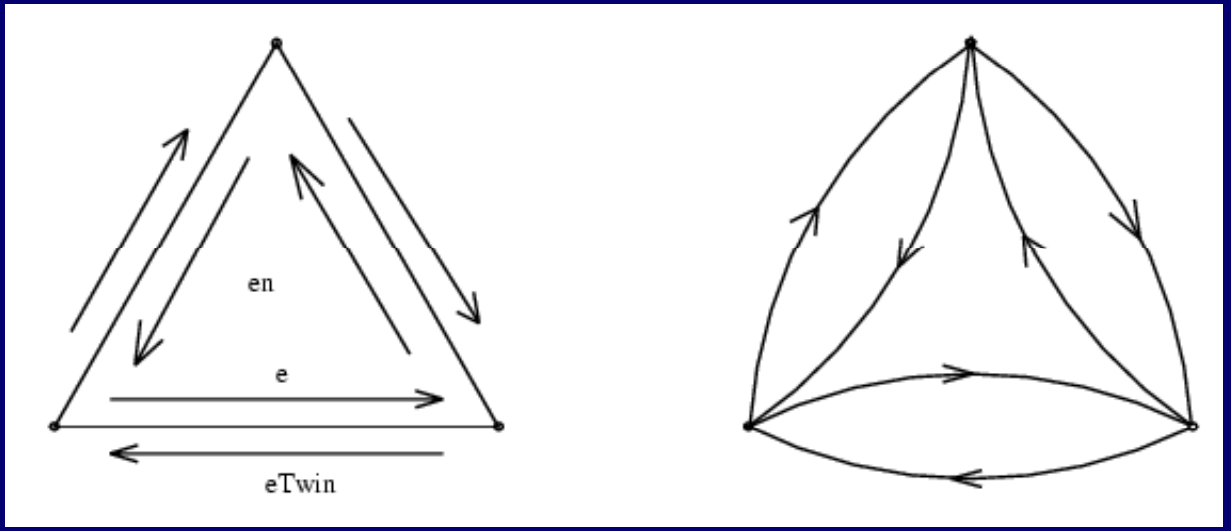


4. Constructing a tour with MST

- Half-edge data structure
  - For convenience, we use a half-edge data structure to describe the triangle mesh, i.e. we replace each edge with two opposed half-edges; the three half-edges inside a triangle are ordered anticlockwise.

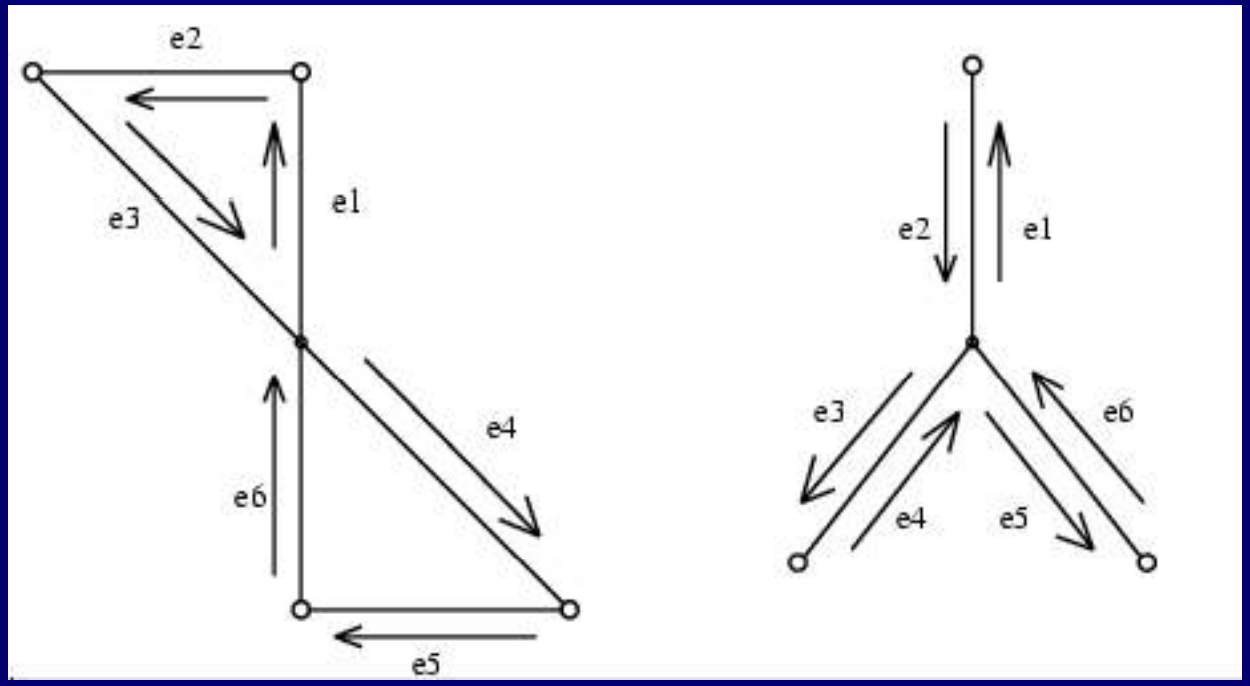


– With this data structure, the mesh can be represented by a directed weighted graph G.



- Requirements of a directed tour
  - A directed tour must be found in G which satisfies two requirements:
    - The tour must visit all terminals exactly once.
    - The tour must be not self-intersected, i.e. the tour must separate the graph into two regions.

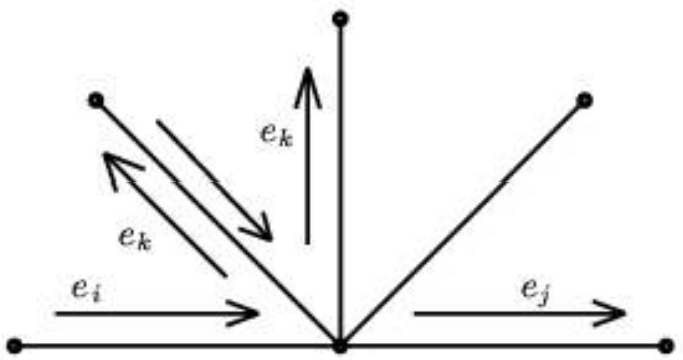
– Self-intersection of a tour



o are terminal nodes, • are non-terminals.

- Tour self-intersection testing algorithm

For each half-edge e_i on a directed tour T
{
  Let e_j be the next adjacent edge to e_i in T
  $e_k = e_i.Next$; // Next half-edge in the data structure
  While ($e_k \neq e_j$) // If e_k is not the next edge of the tour
  {
    If e_k is a half-edge in T
      return; // T is self-intersecting
    Else
      { $e_k = e_k.Twin$; $e_k = e_k.Next$; }
  }
}



- Computing the tour by MST
  - To find the shortest tour which satisfies the two requirements above, it's a problem similar to the classic traveling salesman problem (TSP)(旅行商问题) which asks for a shortest tour that visits all nodes of a graph.

    But:

    Here we only restrict the tour to visiting all terminals.

- Since the part of the shortest tour between two terminals must be the shortest path between them, we can construct a complete graph G_T based on all terminals.

  The weight of each edge in G_T is the length of the shortest path between each pair of corresponding terminals.

  Thus we can reduce the problem to a traveling salesman problem in G_T.

- Thus, we use the basic MST method together with a heuristic(启发式的) which not only approximately minimizes the tour, but also avoids self-intersections of the tour. The idea is as follows:
  - Firstly we compute the minimal spanning tree of G_T .
  - Then we simply walk around the tree in half-edge order, and obtain a directed tour which is not self-intersected in G.
  - Eliminate(去除) the repeated visits to terminals from the tour, since this tour does not satisfy the requirement of visiting each terminal only once.

- Note that:
  - We remove repeat visits to terminals one-by-one until all terminals appear in the tour only once. But this operation may produce a self-intersection in the tour. So, each time we remove a visit to a terminal, we must check whether the resulting tour is self-intersecting.
  - To decide which terminal to update, a heuristic(启发式的) rule is used. We choose the terminal t whose *Pre* and *Next* with minimal shortest path.

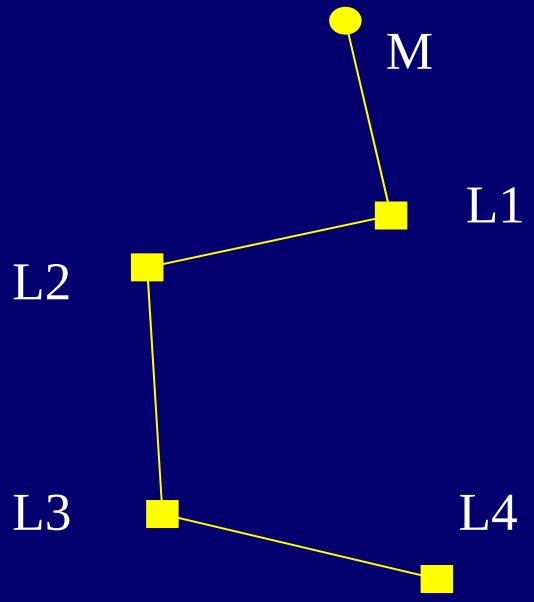
- Algorithm for computing a tour from *MST*
  - Use Kruskal's algorithm to compute the *MST* of G_T . Walk around the tree starting at any terminal. Record the sequence of terminals in a circular list *TourList*.

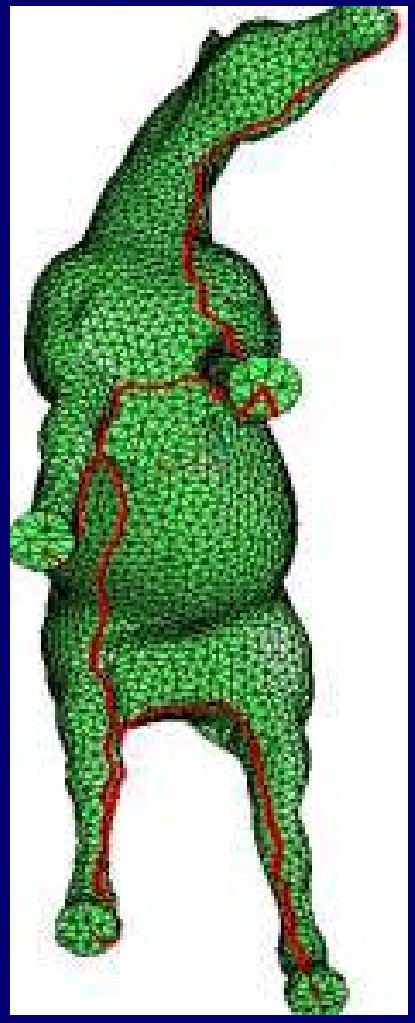
- Remove repeated terminals

```
While (size of TourList > number of terminals)
//There are repeated terminals in the tour
{
    For each repeated terminal t in TourList
    {
        If the tour is not self-intersecting when the appropriate visit
        to t is removed
            t.priority = The length of shortest path
            between t → Pre and t → Next
        Else
            t.priority = ∞
    }
    Remove the visit with minimum priority from TourList
}
```

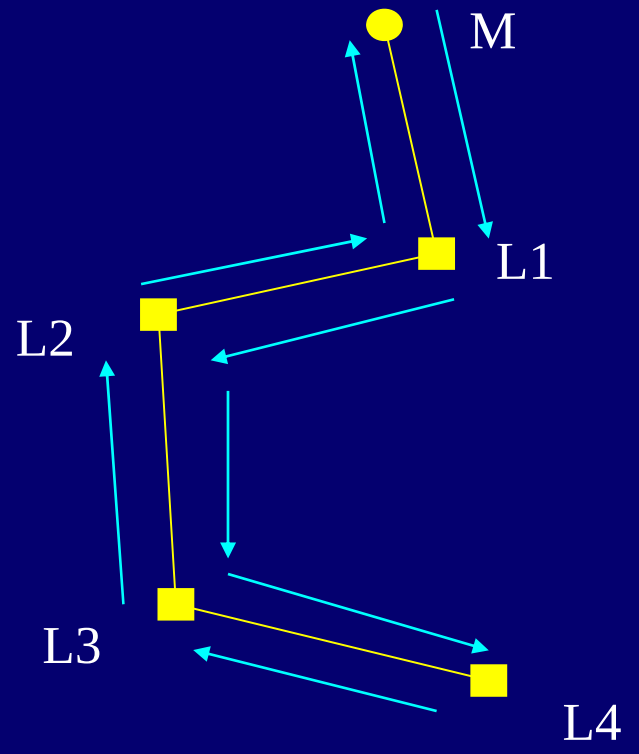
– Example of the horse

  • We obtain the MST of G_T , the pattern in 3D mesh is shown in left figure.

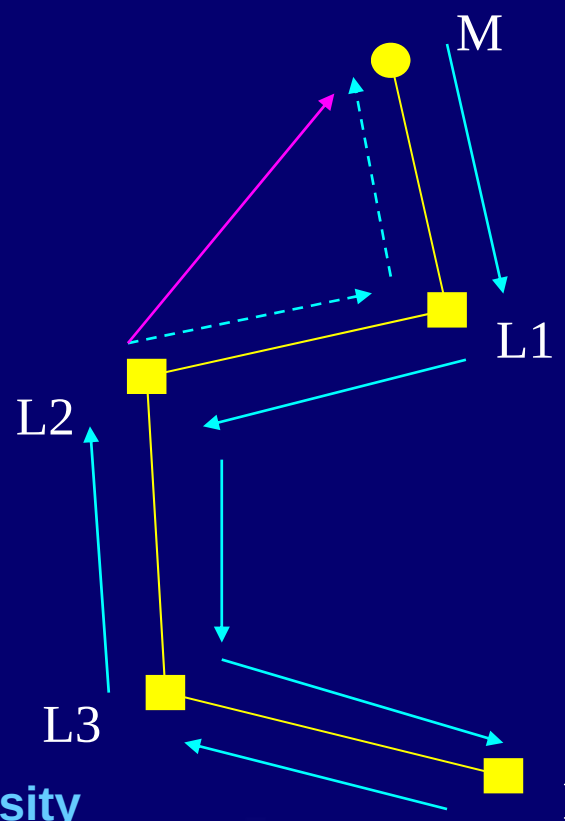




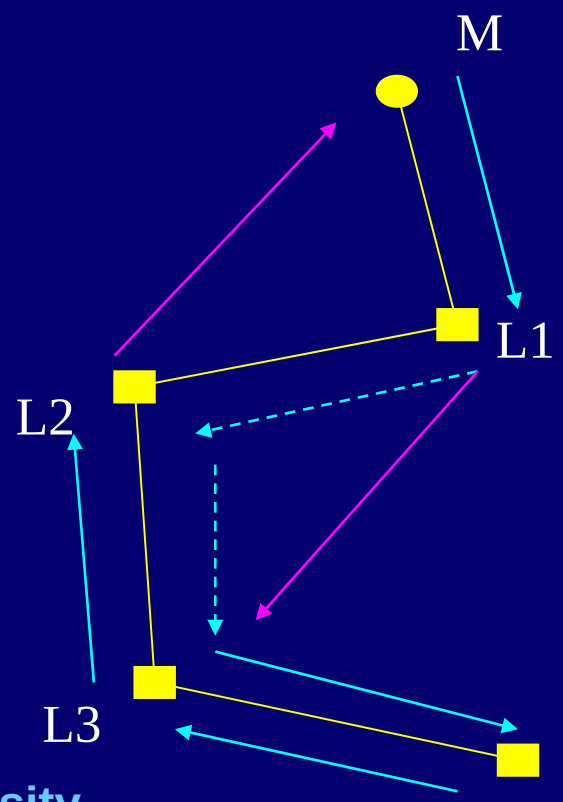
- Then we obtain an initial tour, {M, L1, L2, L3, L4, L3, L2, L1}
  Obviously, L1, L2, L3 double visited.



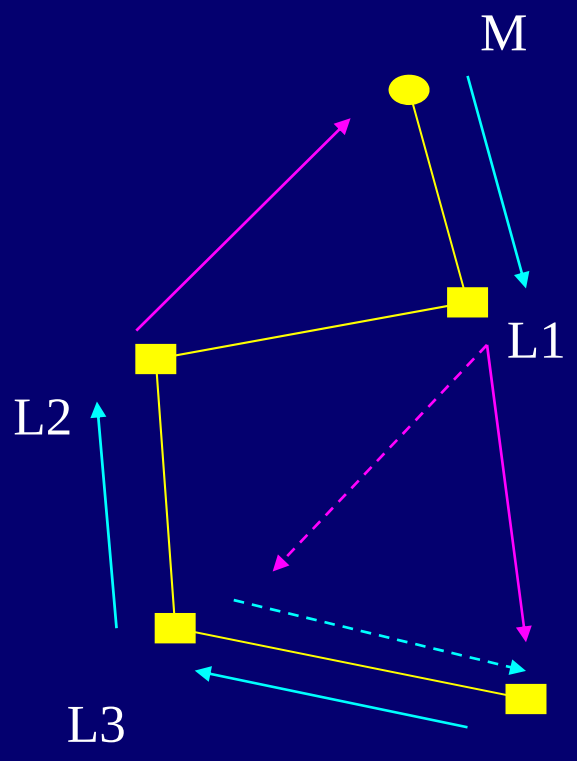
- We replace (L2, L1, M) by (L2 M): {M, L1, L2, L3, L4, L3, L2, }



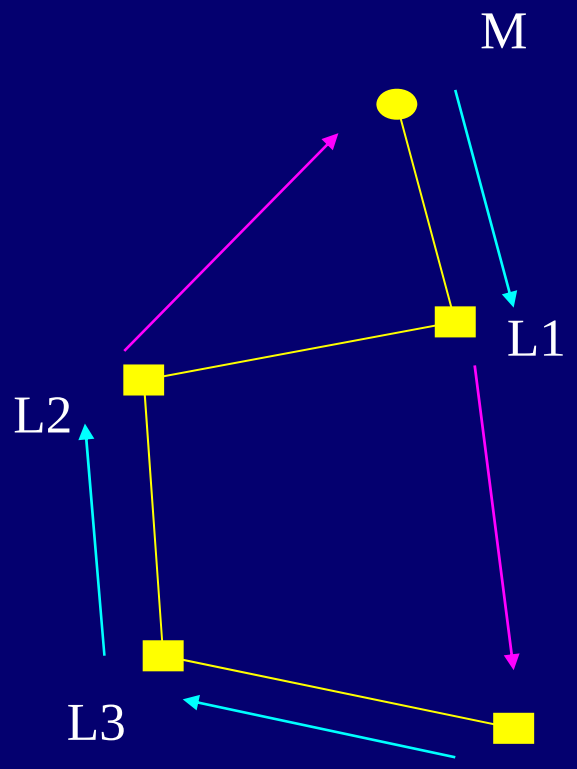
- We replace (L1, L2, L3) by (L1 L3): {M, L1, L3, L4, L3, L2,}

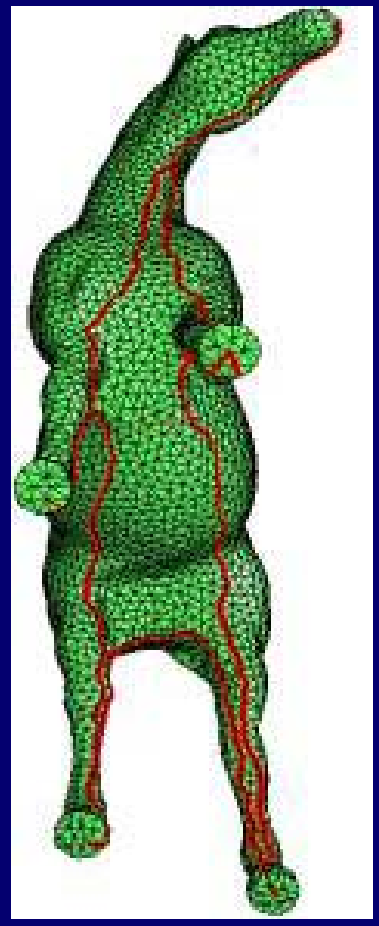


- We replace (L1, L3, L4) by (L1 L4): {M, L1, L4, L3, L2 }



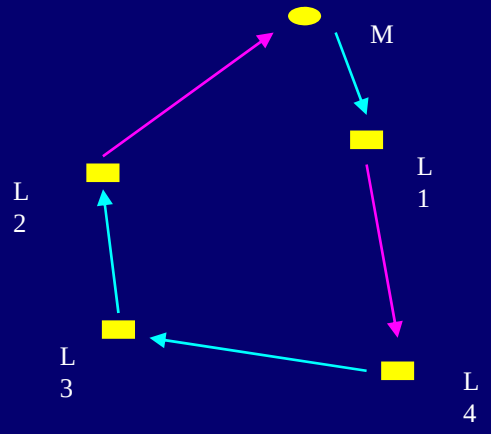
- We finally have the tour: {M, L1, L4, L3, L2 }

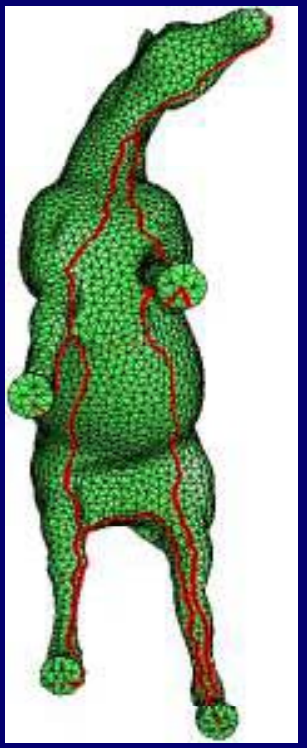




5. Shrinking the tour to a Skeleton

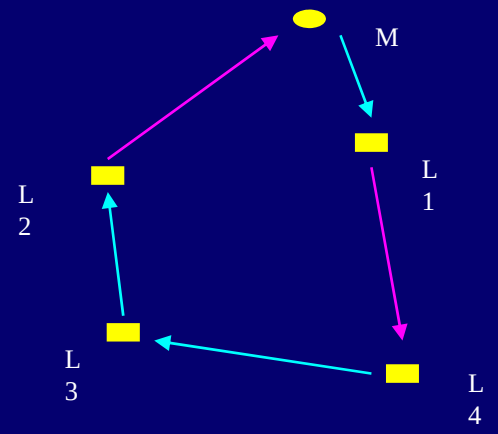
- we treat the part of the tour between two neighbouring terminals as an “edge” and shrink all “edges” inward in a uniform-speed way across the triangulation.
- The skeleton is the interference pattern of wave-fronts propagated from these “edges".

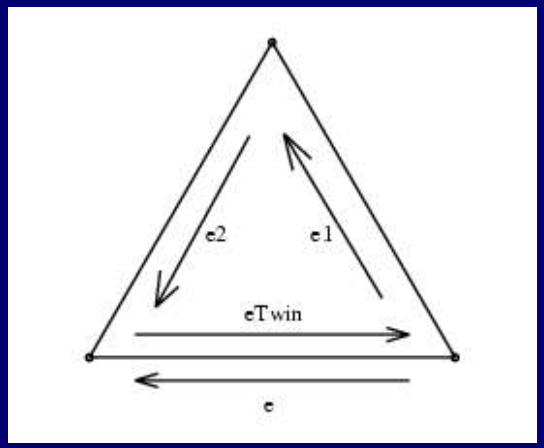


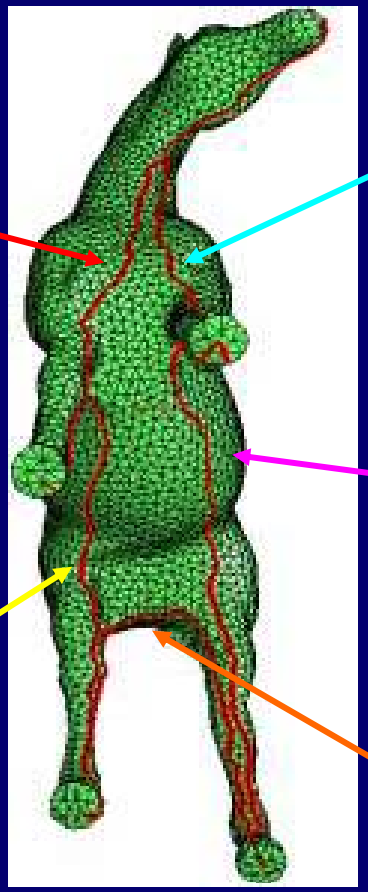


– To carry out this idea, we mark each “edge” with a different number and propagate(传播) these numbers stepwise to the triangulation of the interior(内部) of the tour.

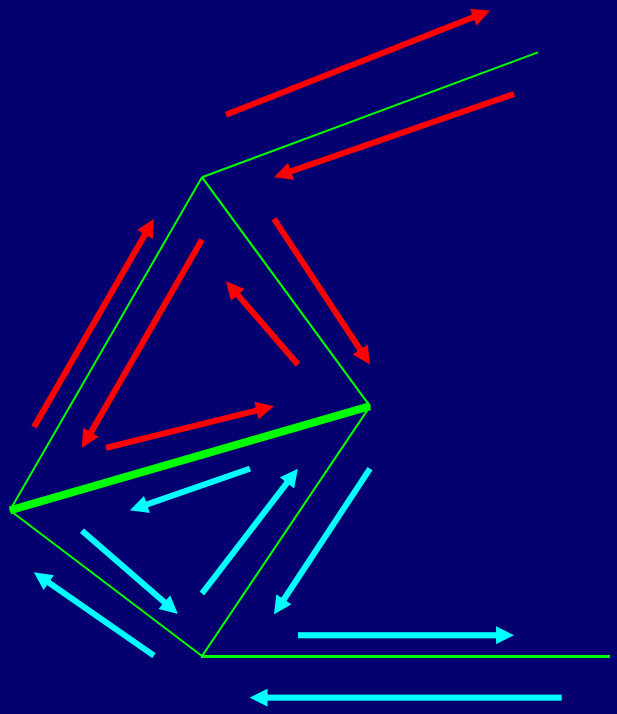
(see picture in next page)

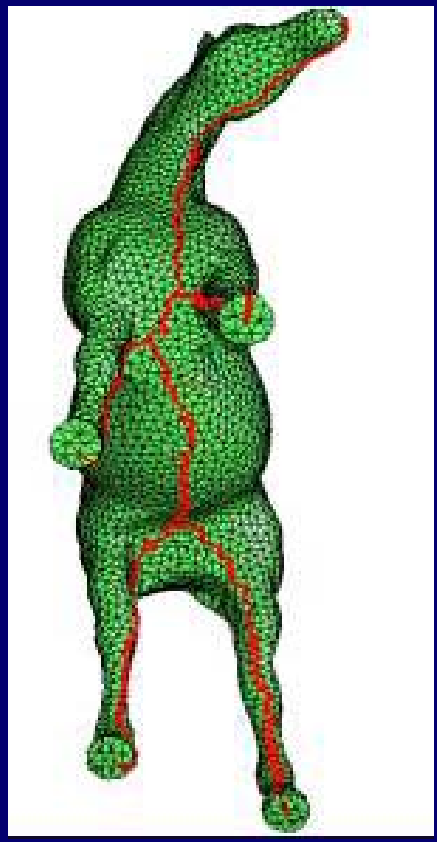

M
L 1
L 2
L 3
L 4


e2
e1
eTwin
e


1
2
3
4
5

– Places where two different “edge” numbers meet are parts of the skeleton.





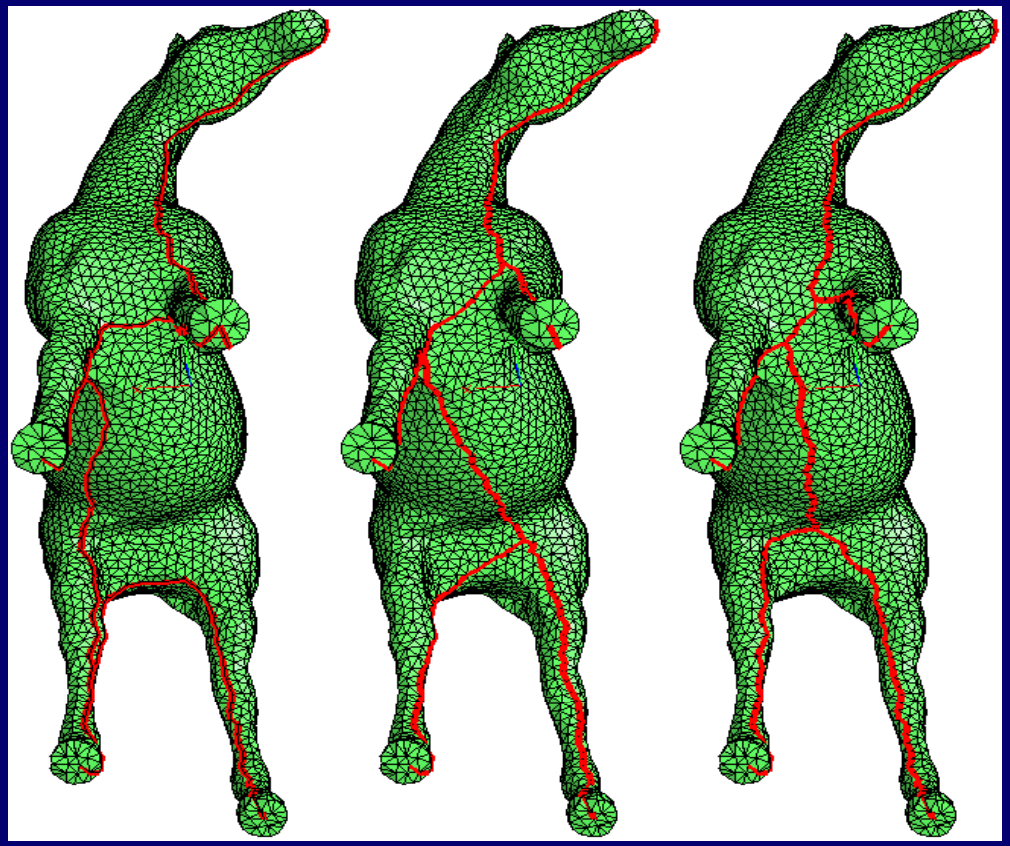
6. Straighening the Skeleton

- The skeleton as produced above is always jagged and should be smoothed to produce a better result. As intended, the skeleton is a tree spanning all terminals. While all terminals are leaves of the tree, some interior vertices have valence greater than two. Such vertices are called Steiner points.

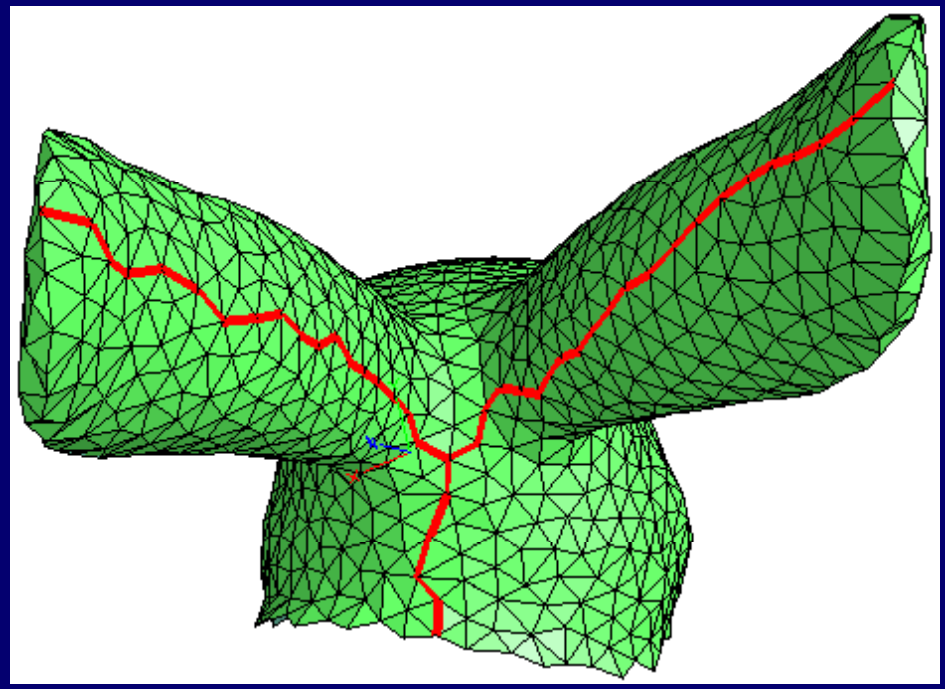
– We fix the positions of all Steiner vertices and all extremal vertices, and replace the skeleton path between each pair of such vertices by the shortest path connecting them in the triangulation

7. Results

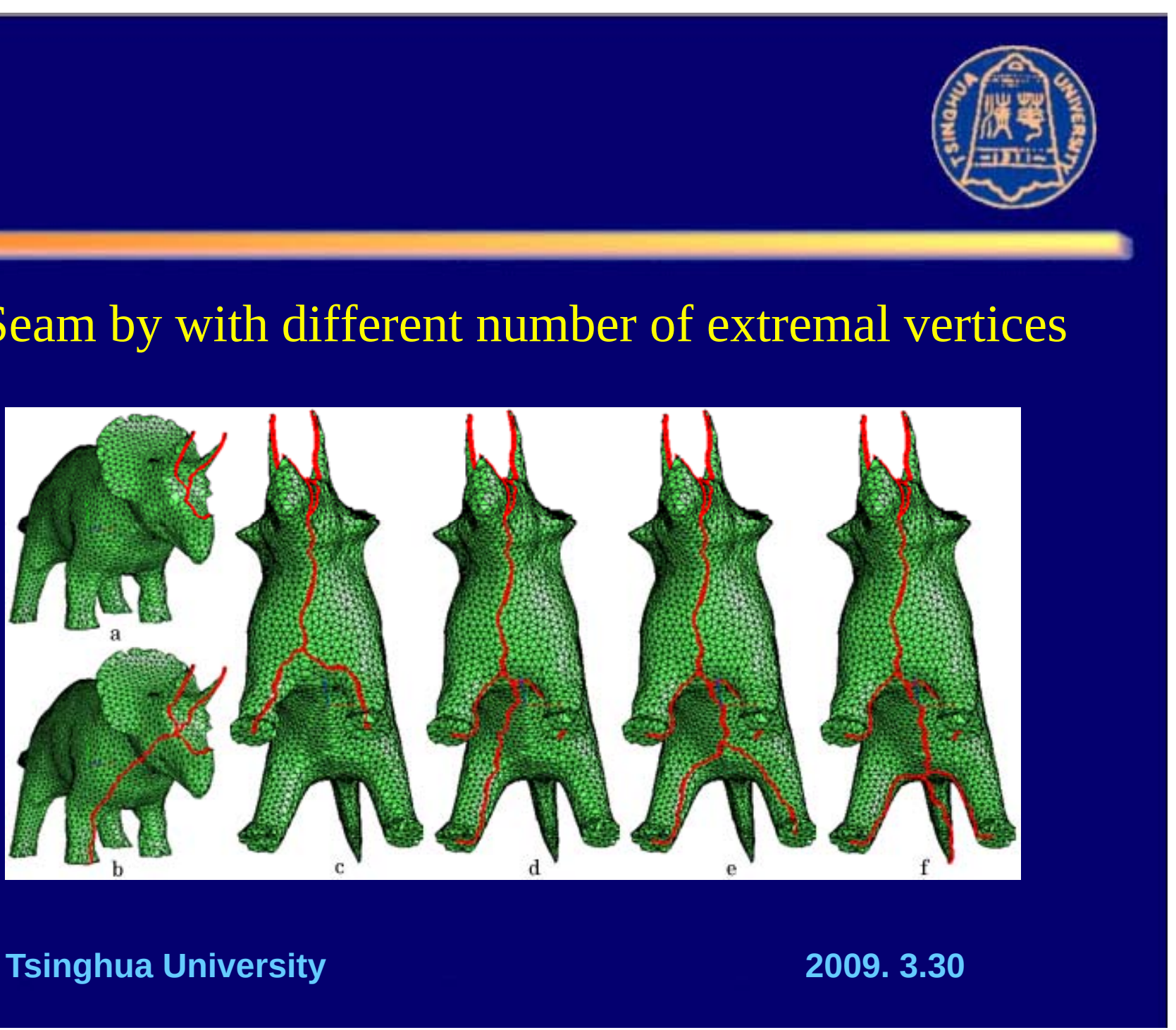
- Comparison
  - Sheffer (7.99)
  - Gu, Gortler, Hoppe (7.07)
  - Skeleton-Based (5.97)



Sheffer Gu Skeleton-based

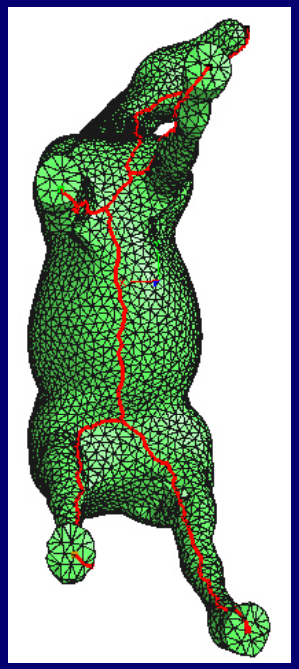


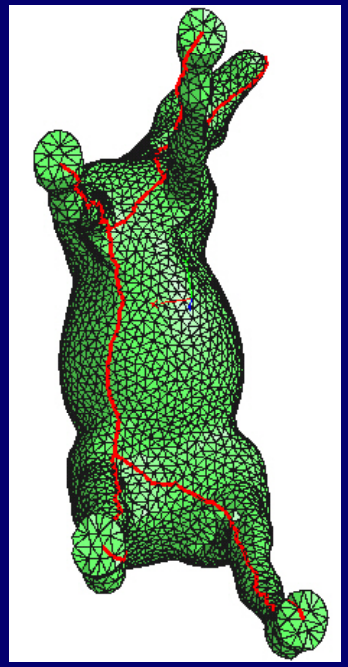
- Seam computing for a open model (bunny head)



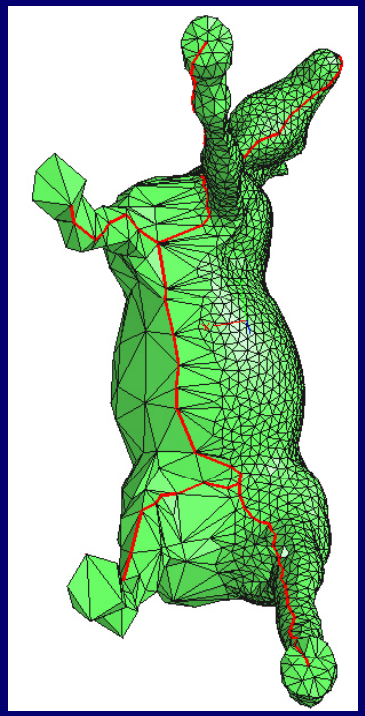
- Seam by with different number of extremal vertices

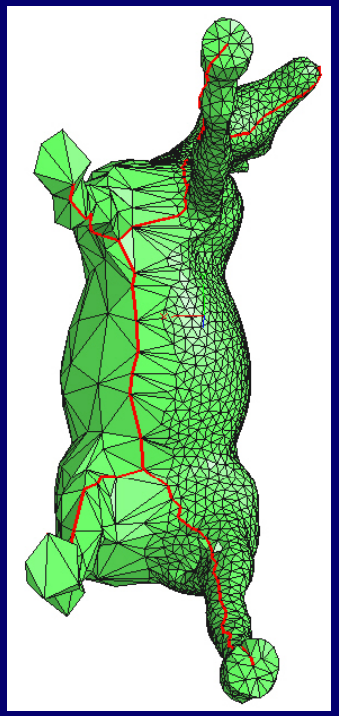
- Further improvement of skeleton-based method
  - We can modifying Steiner points in their near neighbor to further reduce length of seam.





– Adaptive(自适应) shrink(收缩) of tour: In previous shrink algorithm, we suppose the size of triangle is similar, we may use adaptive shrink algorithm which considering size of triangles.





Thank You for your attention!

Reference

- Matthias Eck, Tony DeRose, Tom Duchamp Multiresolution Analysis of Arbitrary Meshes http://research.microsoft.com/en-us/um/people/hoppe/mra.pdf
- Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe **Geometry Images** http://research.microsoft.com/en-us/um/people/hoppe/gim.pdf
- Alla Sheffer **Spanning Tree Seams for Reducing Parameterization Distortion of Triangulated Surfaces**
- Xu-Ping Zhu, Shi-Min Hu, and Ralph Martin **Skeleton-Based Seam Computation for Triangulated Surface Parameterization** http://ralph.cs.cf.ac.uk/papers/Geometry/seam.pdf