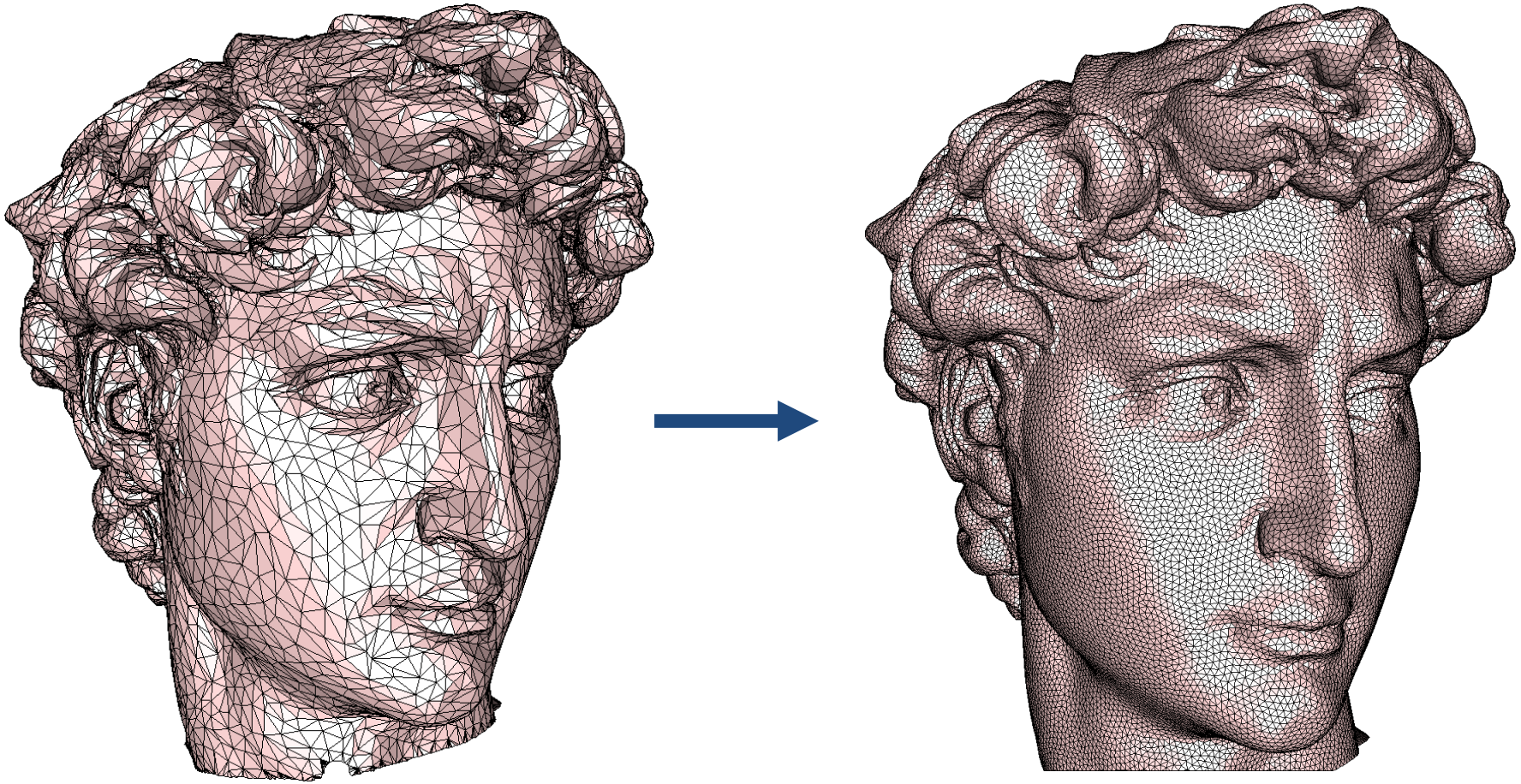
IFT 6113
REMESHING

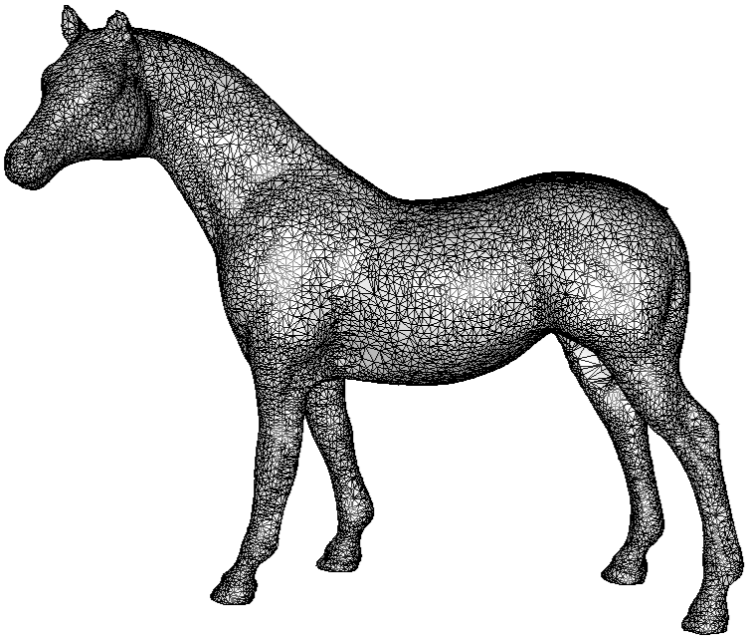
tiny.cc/ift6113

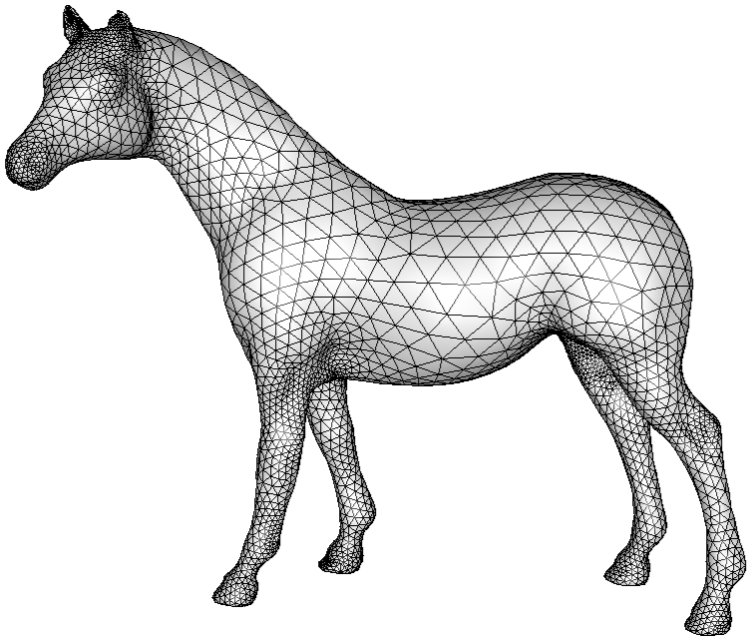


Mikhail Bessmeltsev

Re-Meshing Surfaces

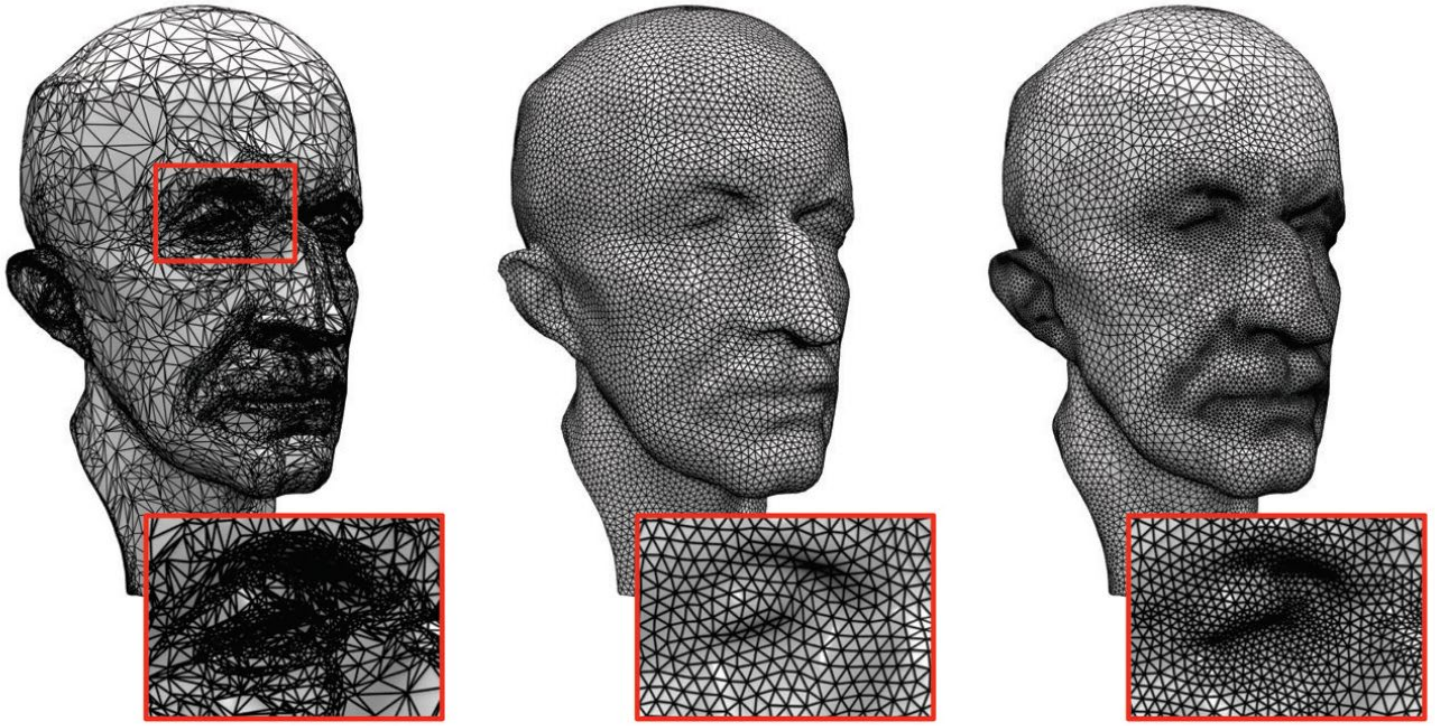
Generate a ***better*** mesh
close to the original surface



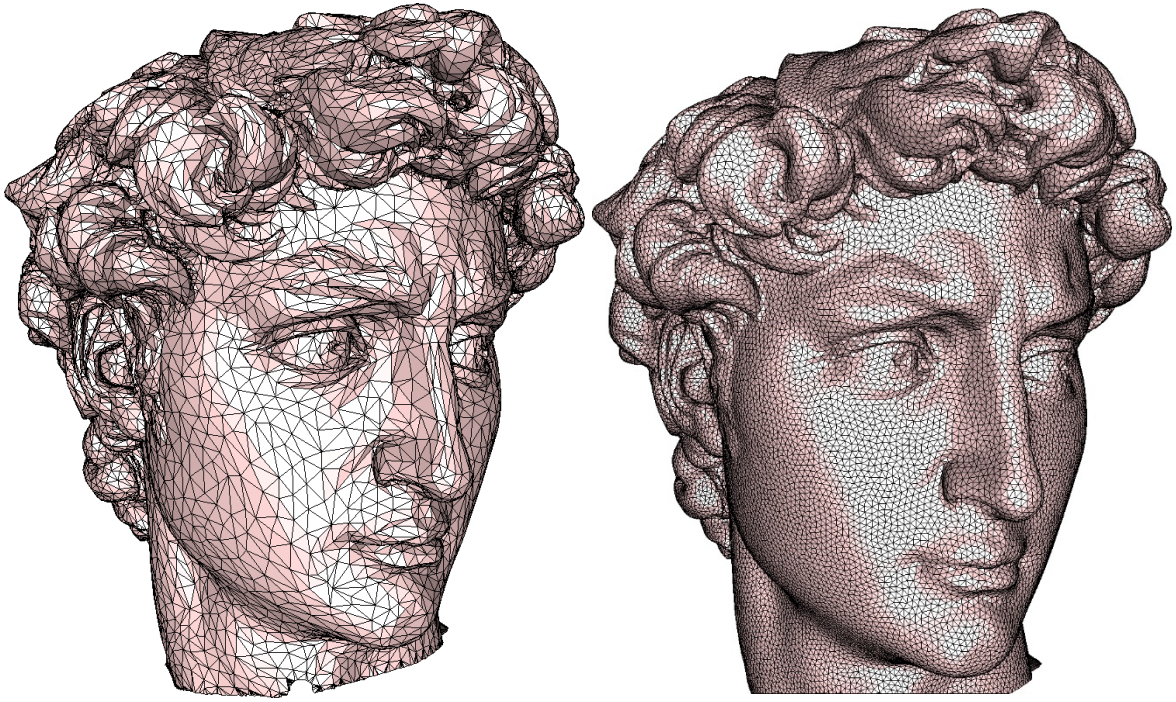


Motivation

- Numerical stability
- Easier modeling
- Quality requirements



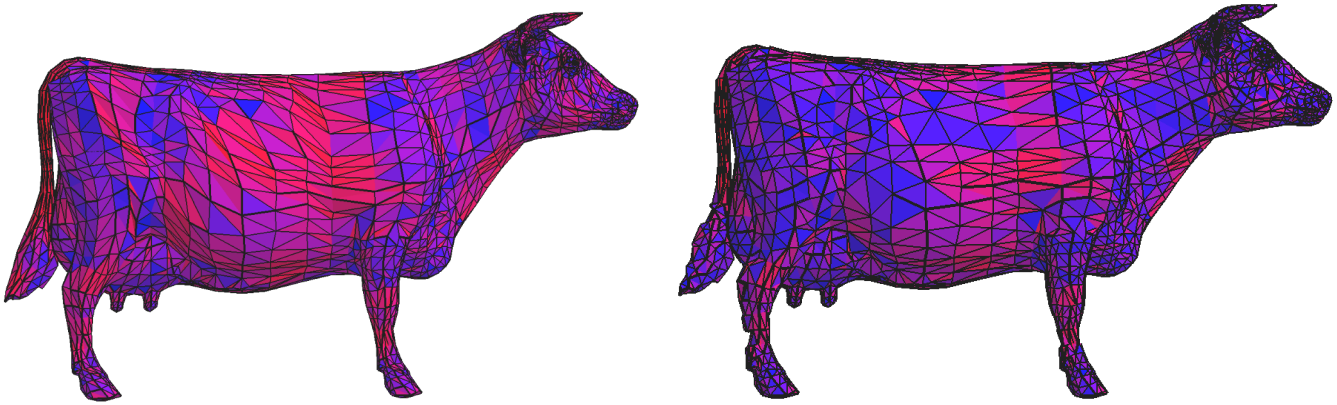
What’s a good mesh?



How to (re)mesh surfaces?

Delaunay triangulation?

- What is Delaunay criterion on surface?
  - Option 1: Use sphere instead of circle
    - Works for volumetric meshes (tets)
  - Option 2: Use pairwise test only
    - Theoretical Delaunay properties?
  - Option 3: Intrinsic Delaunay
- Boundary recovery = Approximation quality



Approaches

- Mesh adaptation/Local Remeshing
  - Locally update mesh while tracking error
- Reduction to 2D/Global Remeshing
  - Parameterize in 2D
  - Mesh in 2D
  - Project back

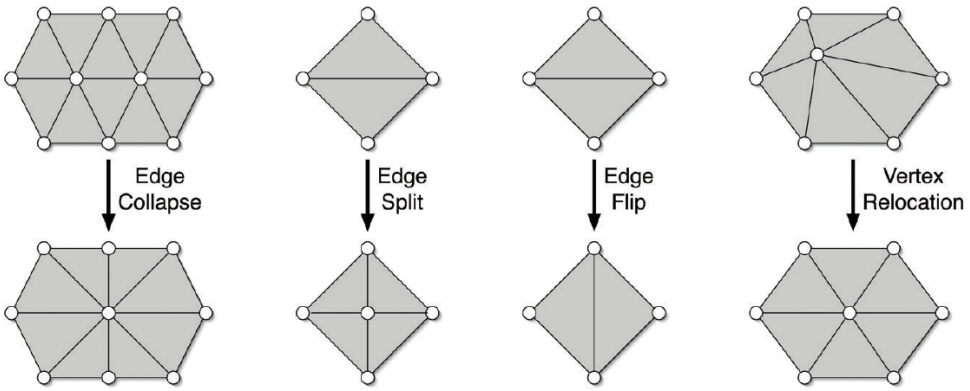
Approaches

- Mesh adaptation/Local Remeshing
  - Locally update mesh while tracking error
- Reduction to 2D/Global Remeshing
  - Parameterize in 2D
  - Mesh in 2D
  - Project back

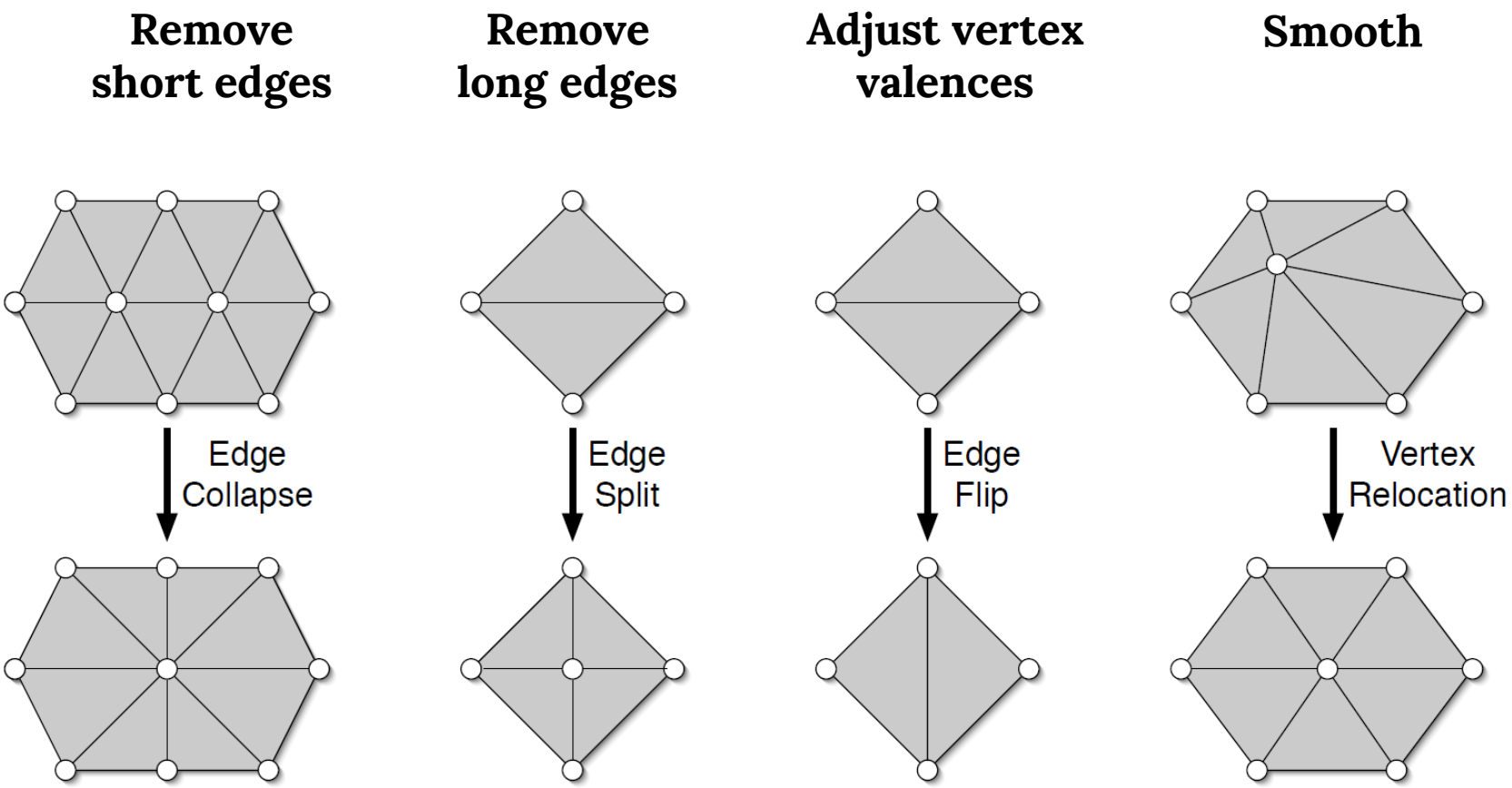
Local approach

1. Refine/Coarsen to satisfy sizing
2. Smooth mesh
3. Perform flips after every other operation
4.

Store original to compute distance/error



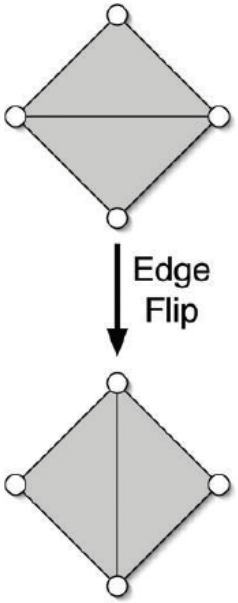
Local approach



Edge Flip

Flip one diagonal if longer than the other

3D equivalent of Delaunay test in 2D



Track approximation error (why?)

- Approximate Hausdorff metric
  - Normal error
  - Smoothness

Test self-intersection

- Complexity? *Maybe skip?*

If P, Q are sets,

$$H_P(Q) = \max_{p \in P} \min_{q \in Q} \|p - q\|$$

Hausdorff Metric:

$$H(P, Q) = \max(H_Q(P), H_P(Q))$$

On mesh approximate by

- measuring vertex to surface distance
- measuring vertex to vertex distance

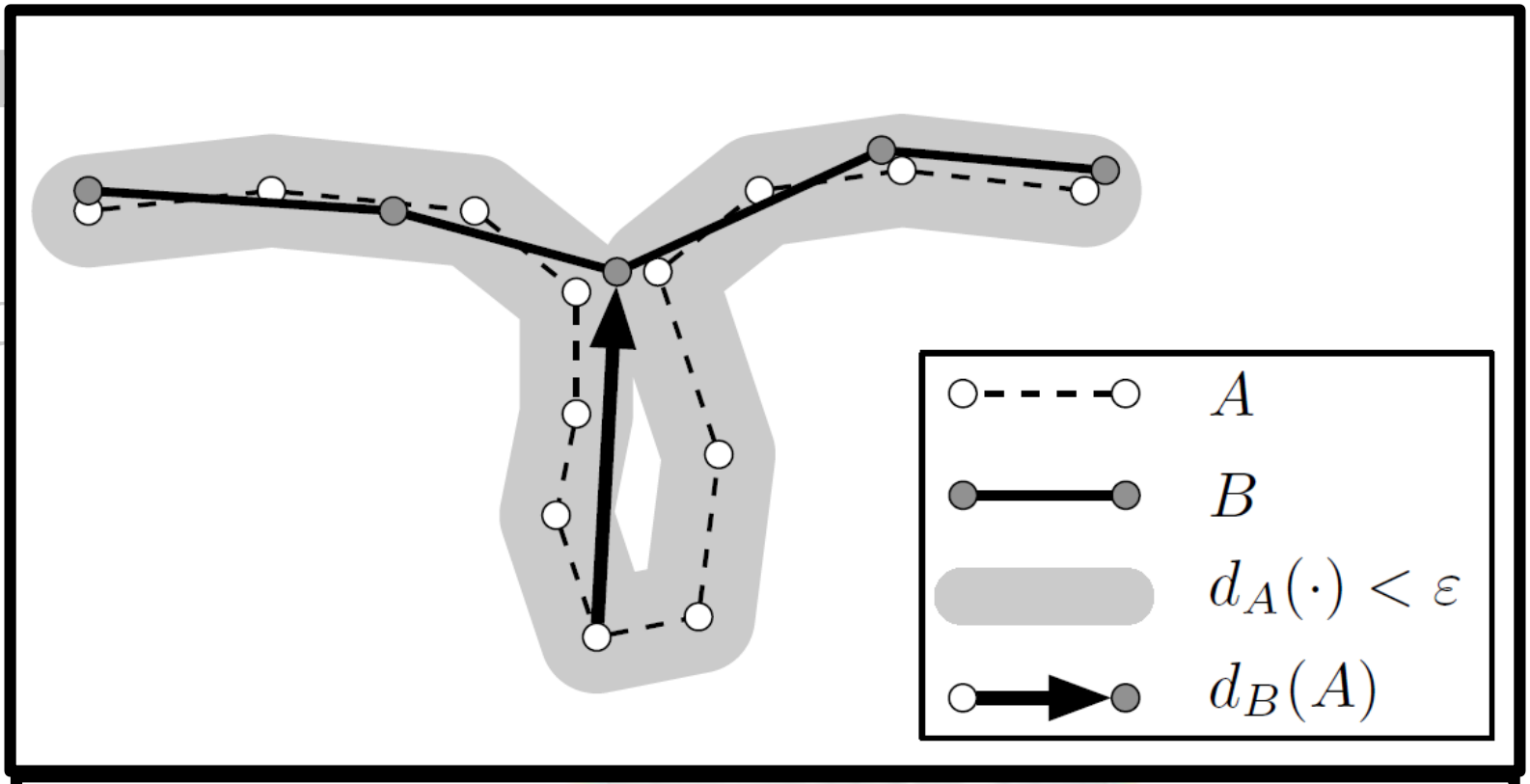
- Computation complexity?

Measuring Error

Hausdorff is expensive => cheat

Idea 1: Stay within an ε-envelope

Measuring Error



Does not limit Hausdorff distance!

Local approach: Edge split

Reach desired sizing or element count

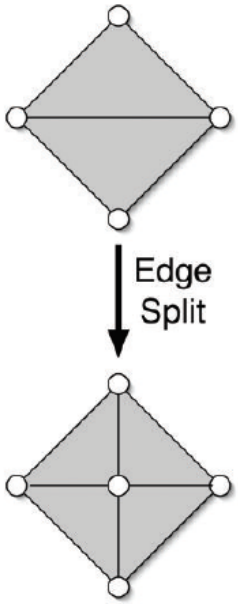
Strategy

Split long edges – insert mid-points

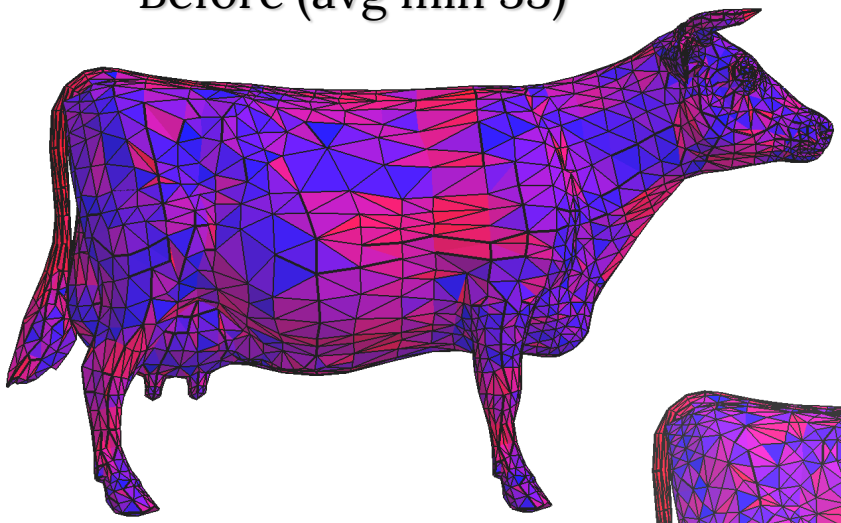
*Project to **original** mesh*

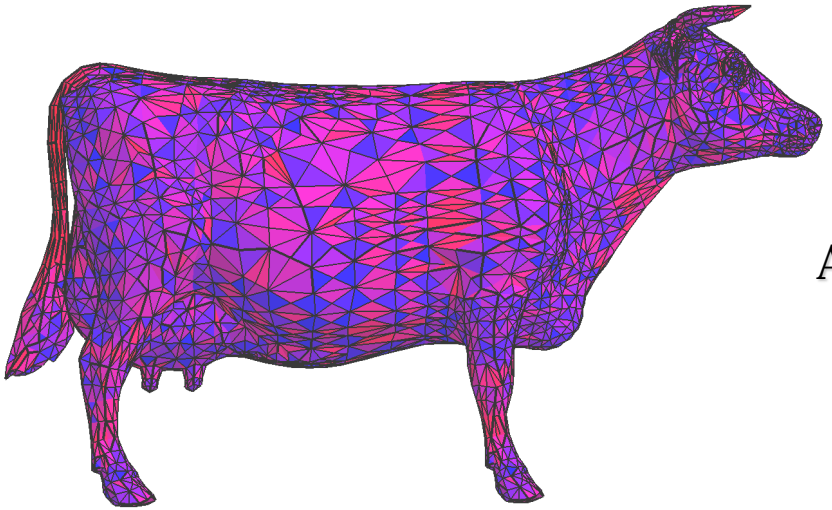
Hard to achieve good spacing

– Improve by smoothing

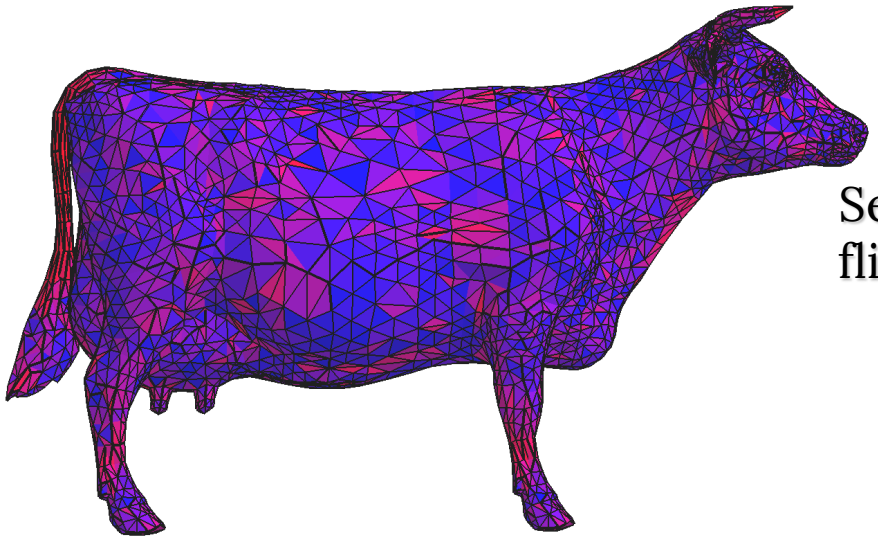


Before (avg min 33)





After (avg min 33)

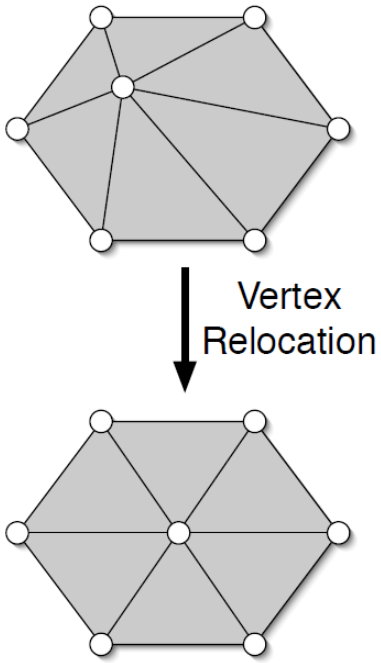


Second round of flips (avg min 37)

Mesh Adaptation: Smoothing

Local Laplacian smoothing

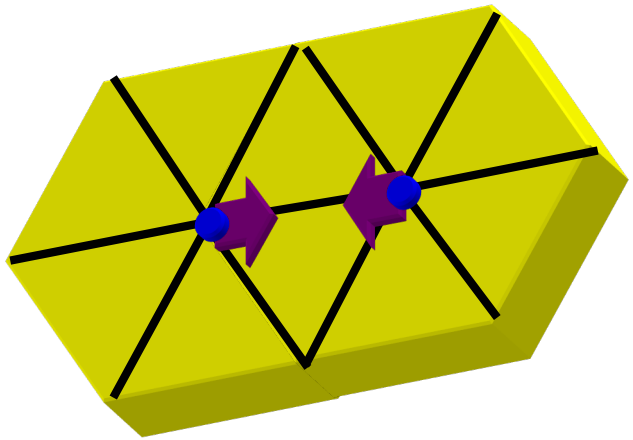
Stay on the surface!



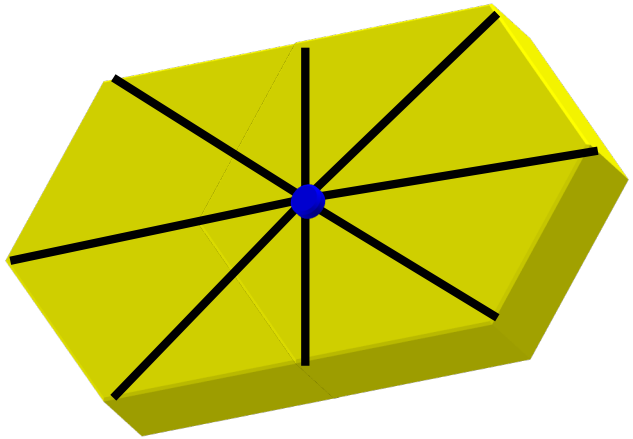
Recall:

Edge Collapse Algorithm

- Simplification operation:
  - Edge collapse (pair contraction)



- Error metric:
  distance, pseudo-global

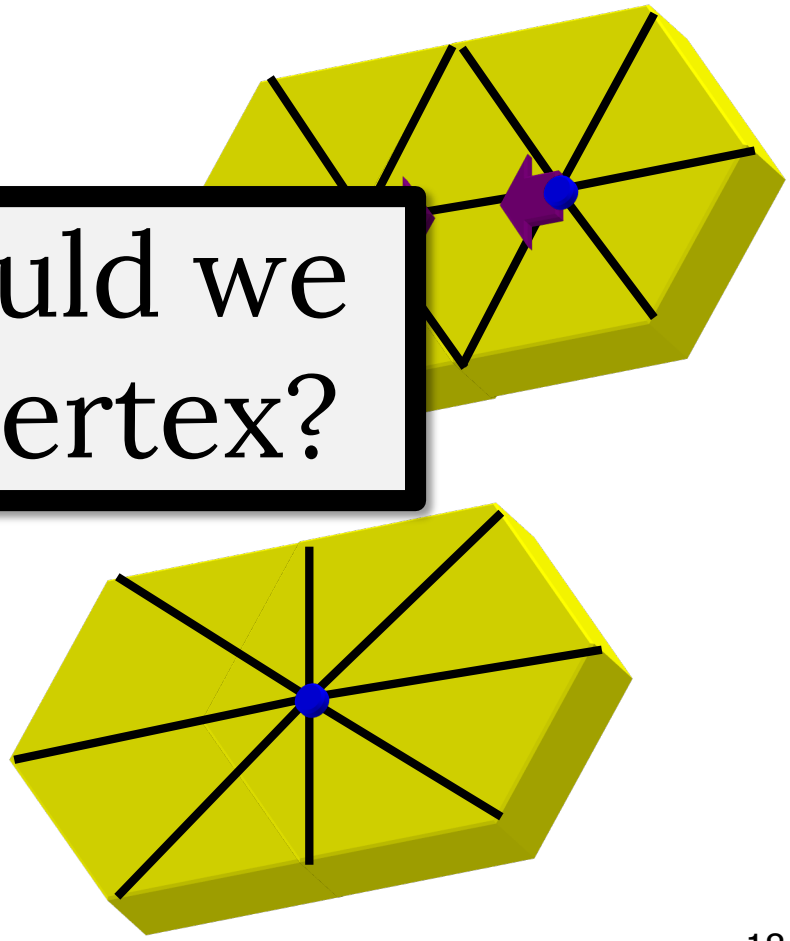


Recall:

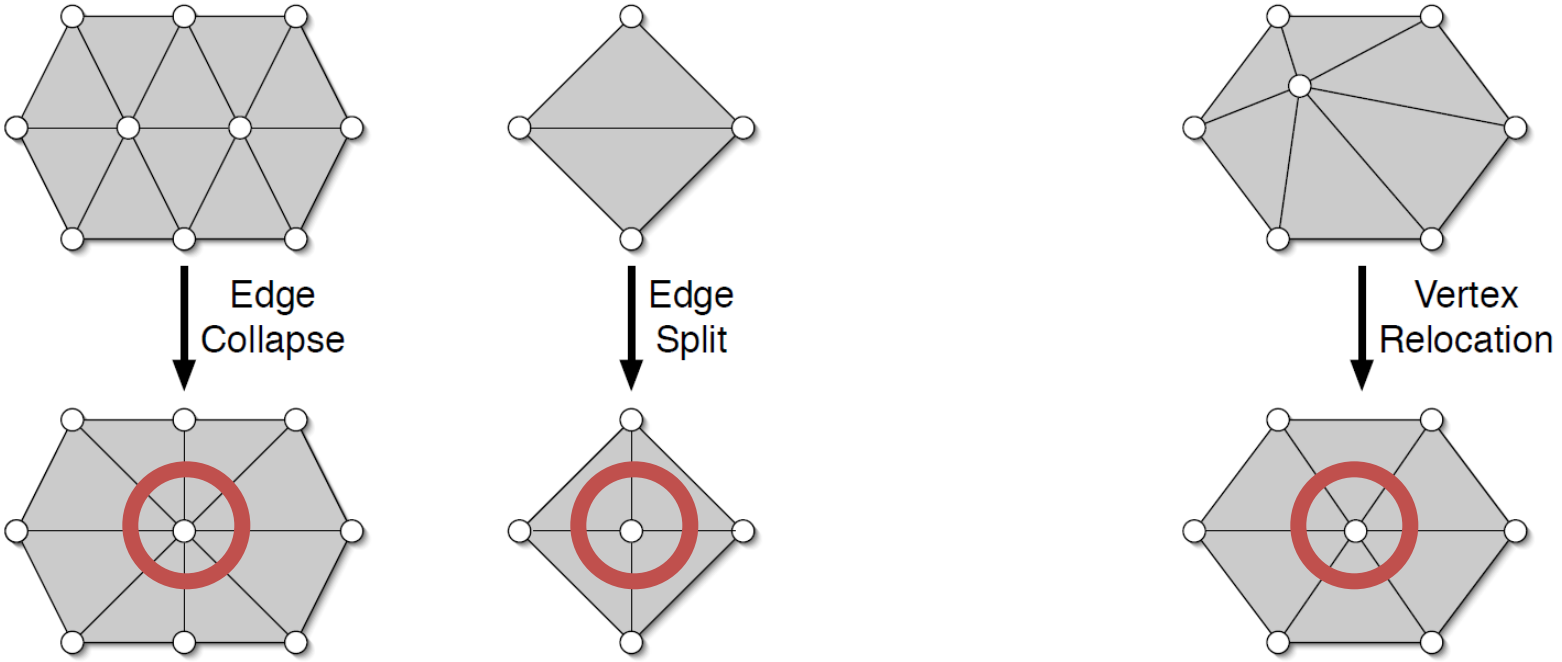
Edge Collapse Algorithm

- Simplification operation:
  - Ed
    co
- Error metric:
  distance, pseudo-global

Where should we place the vertex?



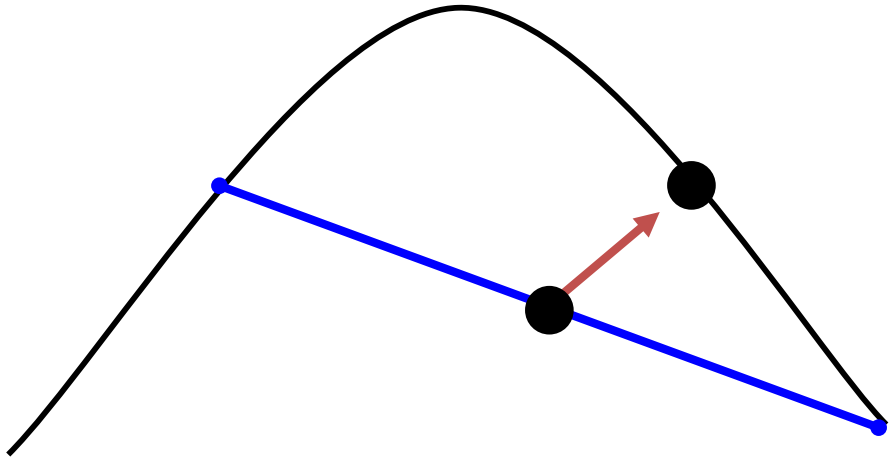
Where to place the new vertex?



Projection to Original Mesh

Nearest point

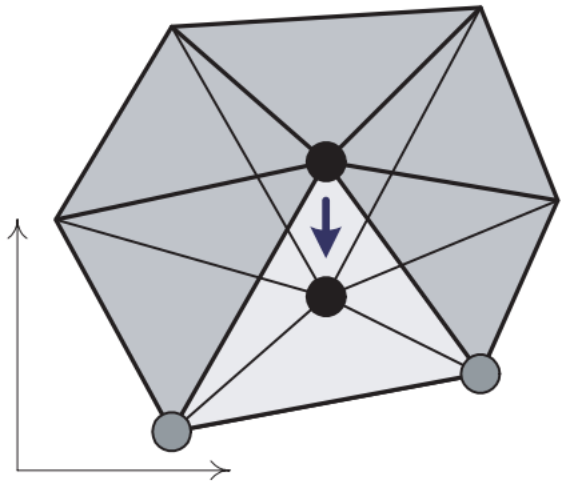
- Expensive search
  - Find original face closest to (estimated) new vertex
- Unlimited Hausdorff error



Vertex relocation

1. Project all adjacent vertices on a tangent plane
2. Find new location in the plane

   Barycentric coordinates in the **new mesh**

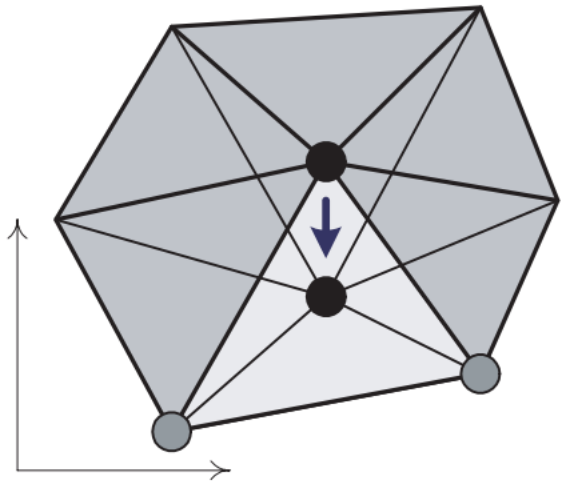


Vertex relocation

1. Project all adjacent vertices on a tangent plane
2. Find new location in the plane

   Barycentric coordinates in the **new mesh**

How to project to the original surface?



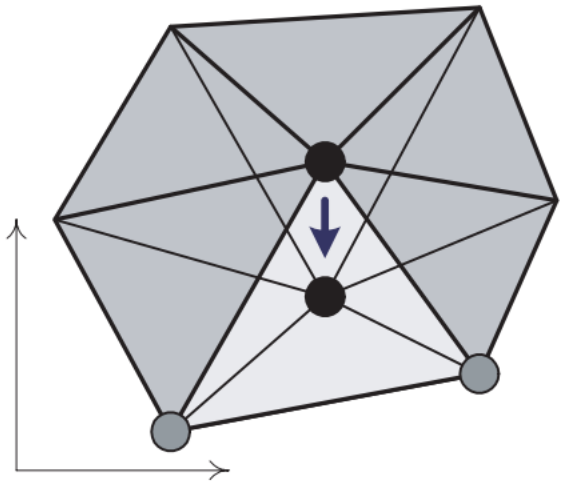
Vertex relocation

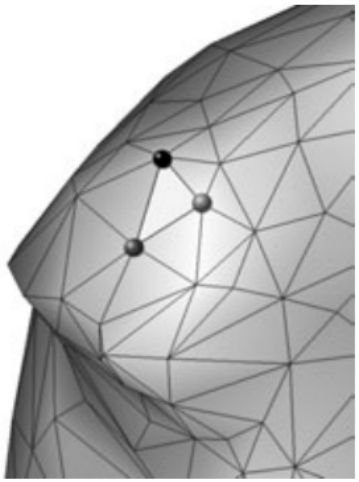
Which 2D triangle does it belong to?

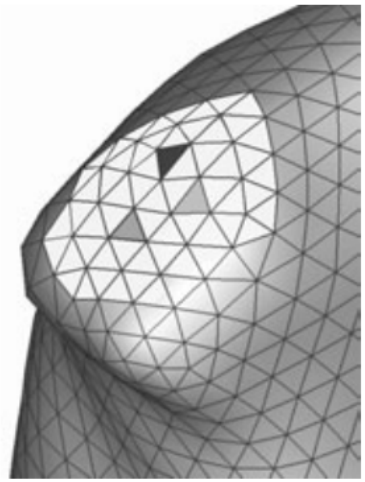
Use triangle vertices'

- triangle indices,
- barycentric coordinates

w/r to the **original mesh**





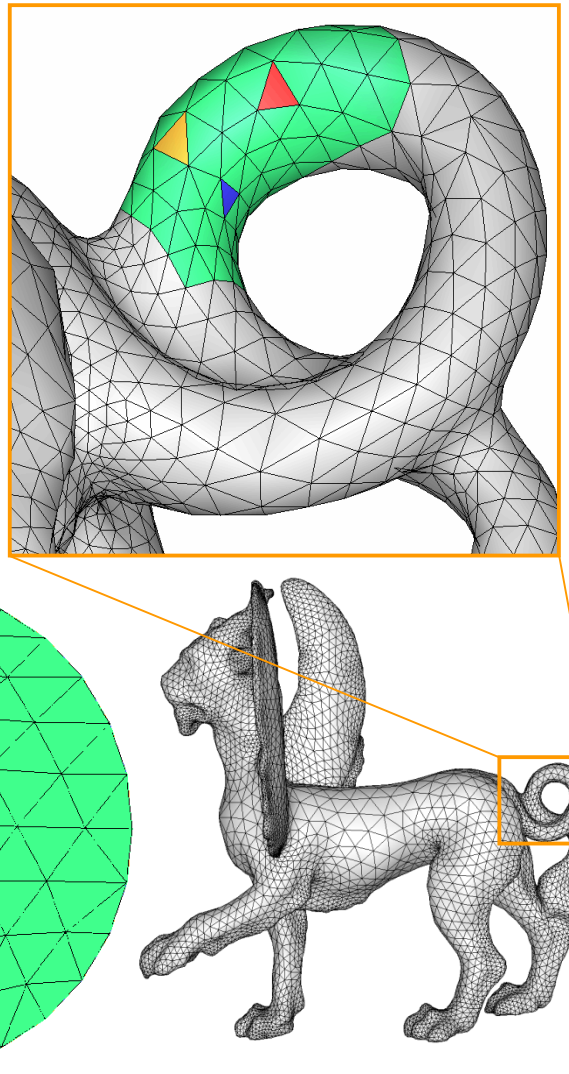


Local Parameterization

Compute a local parameterization for the original mesh

Use the barycentric coordinates to place the vertex in 2D

Lift the vertex in 3D using the parameterization



Local Parameterization

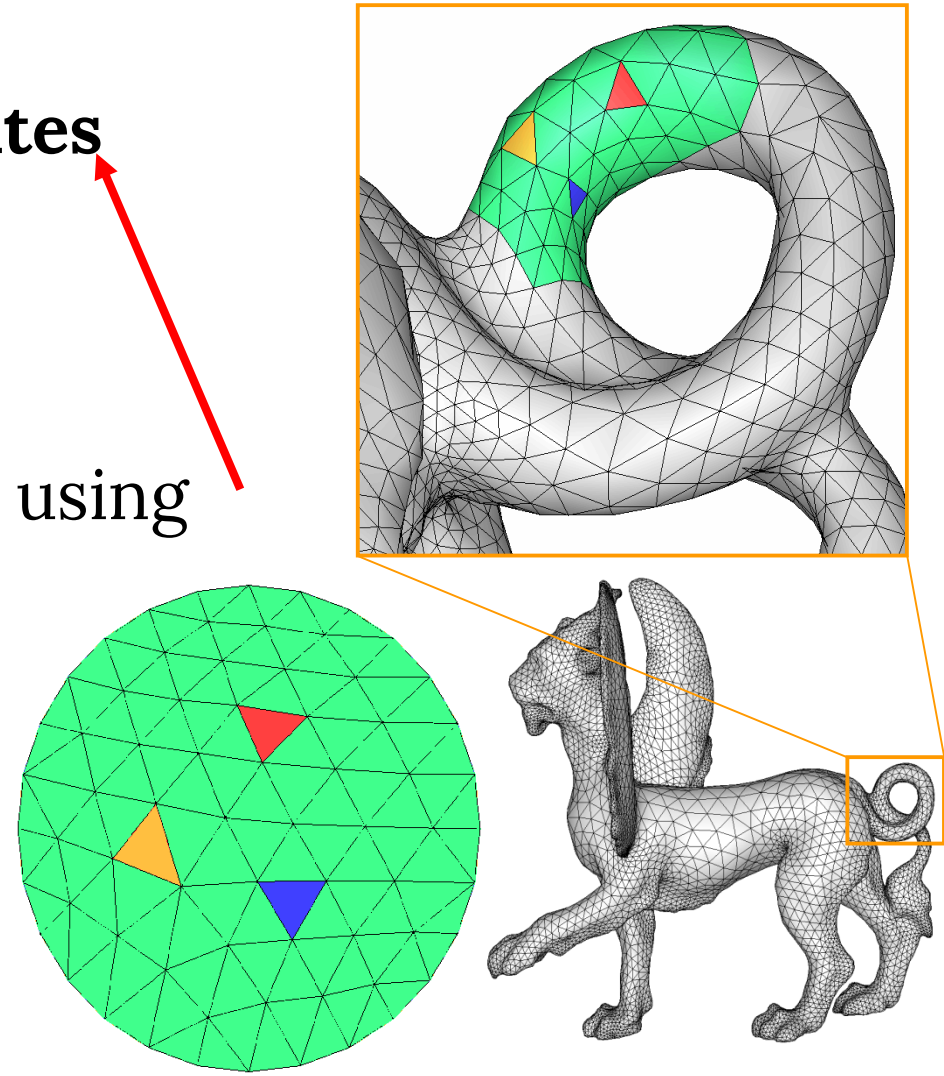
Idea:

use barycentric coordinates

Parameterize surface

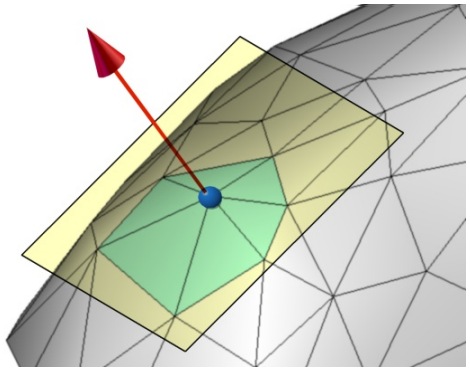
Place the new vertex in 2D using

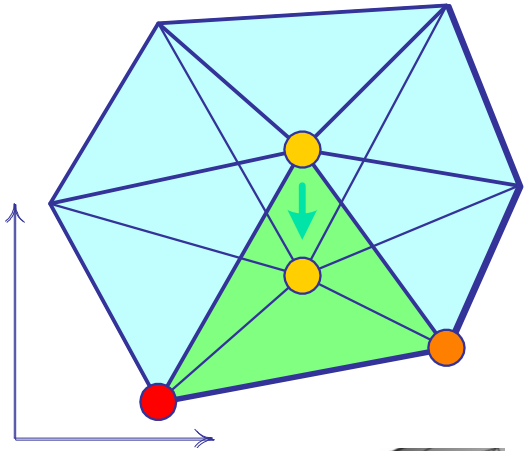
Lift to 3D

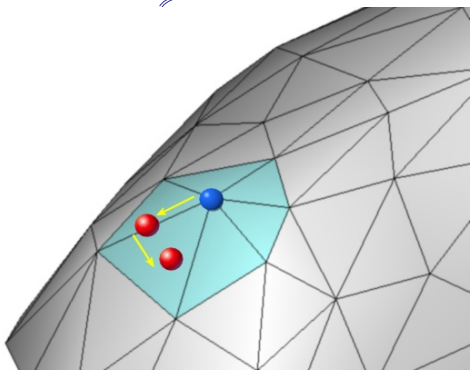


Cheap Local Parameterization

1. Project vertex v + neighbors to tangent plane
2. Move v in the plane
3. Find new triangle in which vertex is located
4. Compute barycentric coordinates in this triangle
5. Lift back to 3D



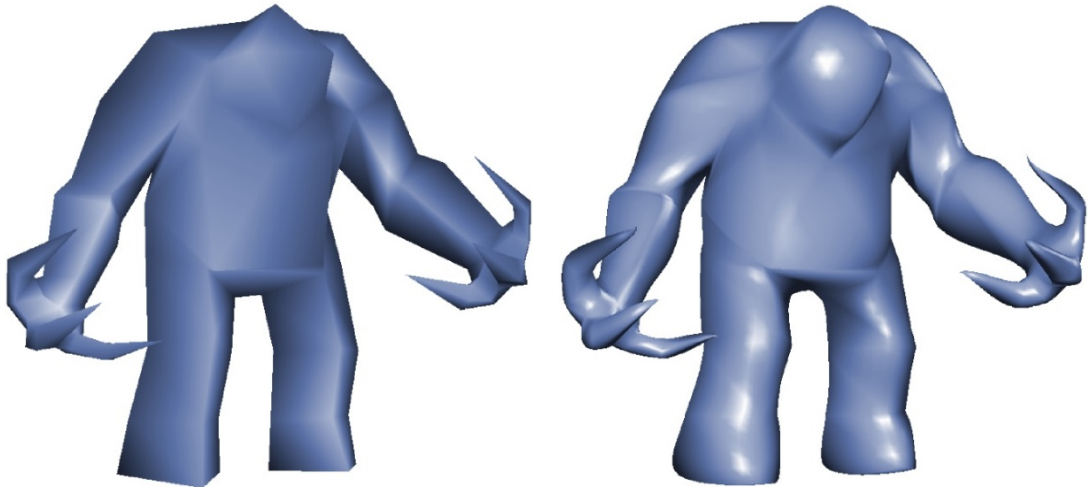




Projection to Approximate Surface

Original mesh approximates “unknown” smooth surface

- Construct local approximation (e.g. quadric)
- Or use vertices + normals of triangle to define patch
  - Hermite, Bézier,…



Local approach: Edge collapse

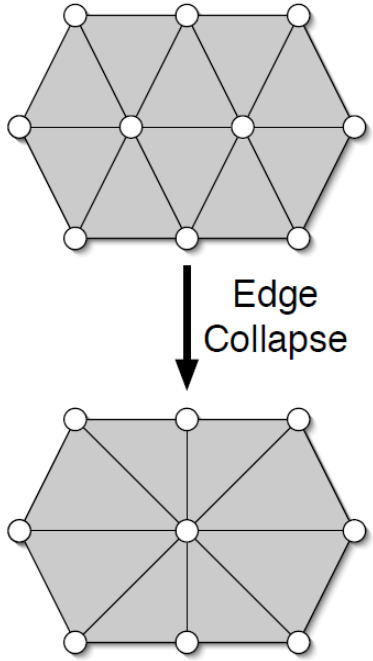
Mesh simplification!

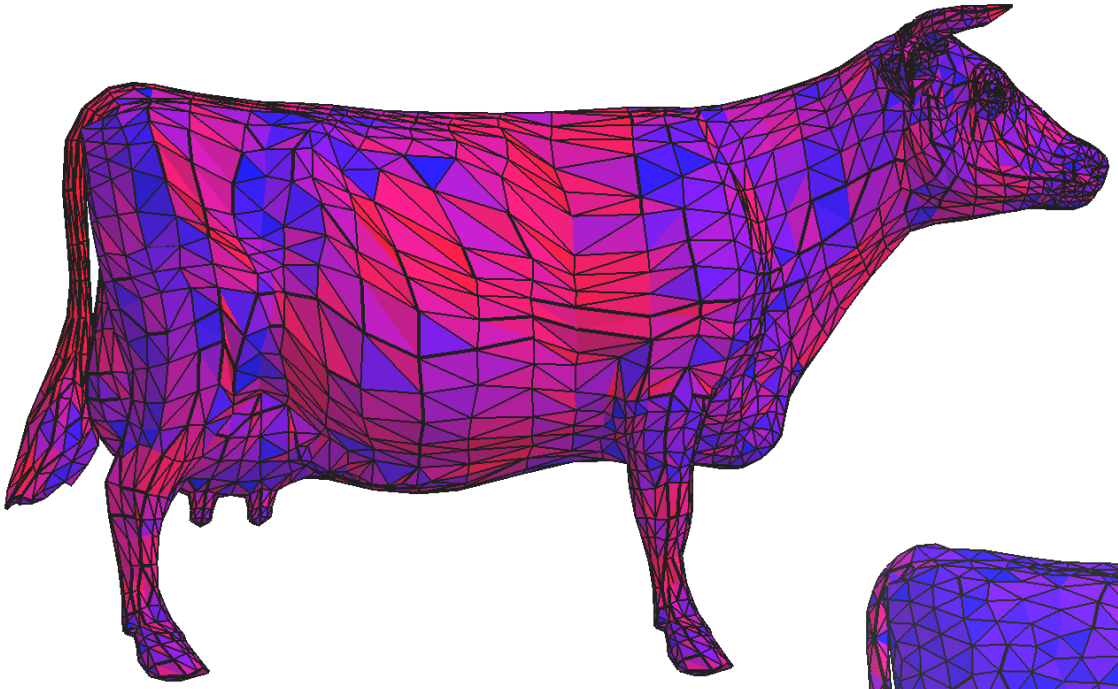
Operations:

- Vertex removal
- Edge collapse
  - Project new vertex to original surface as in refinement

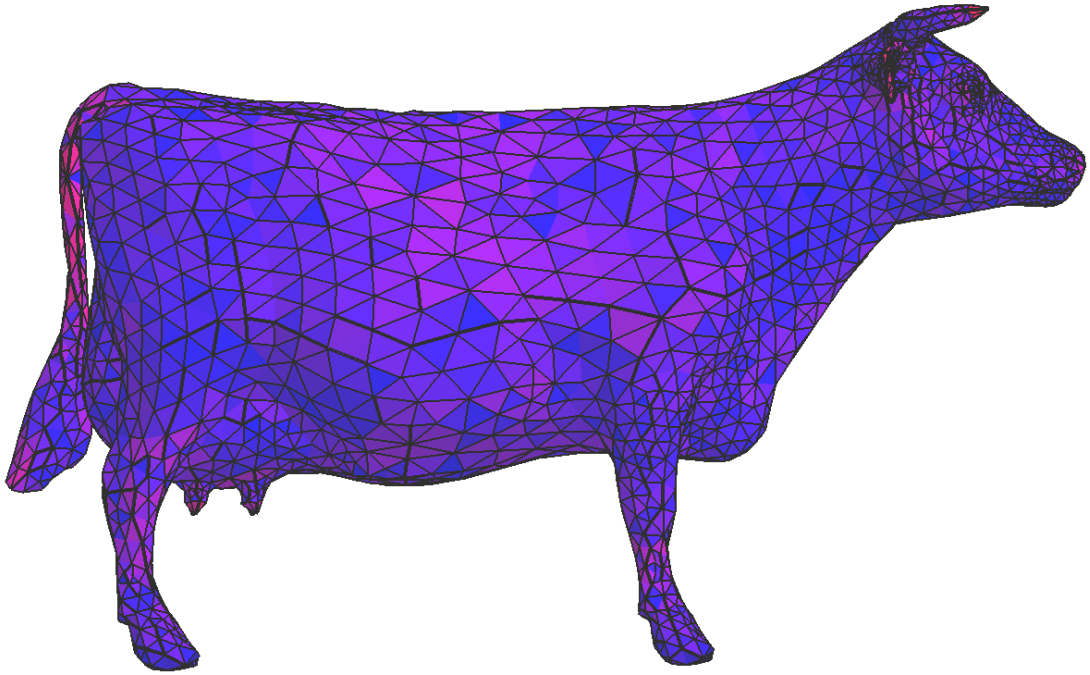
Approximation Error

- Quadrics
- Normal based





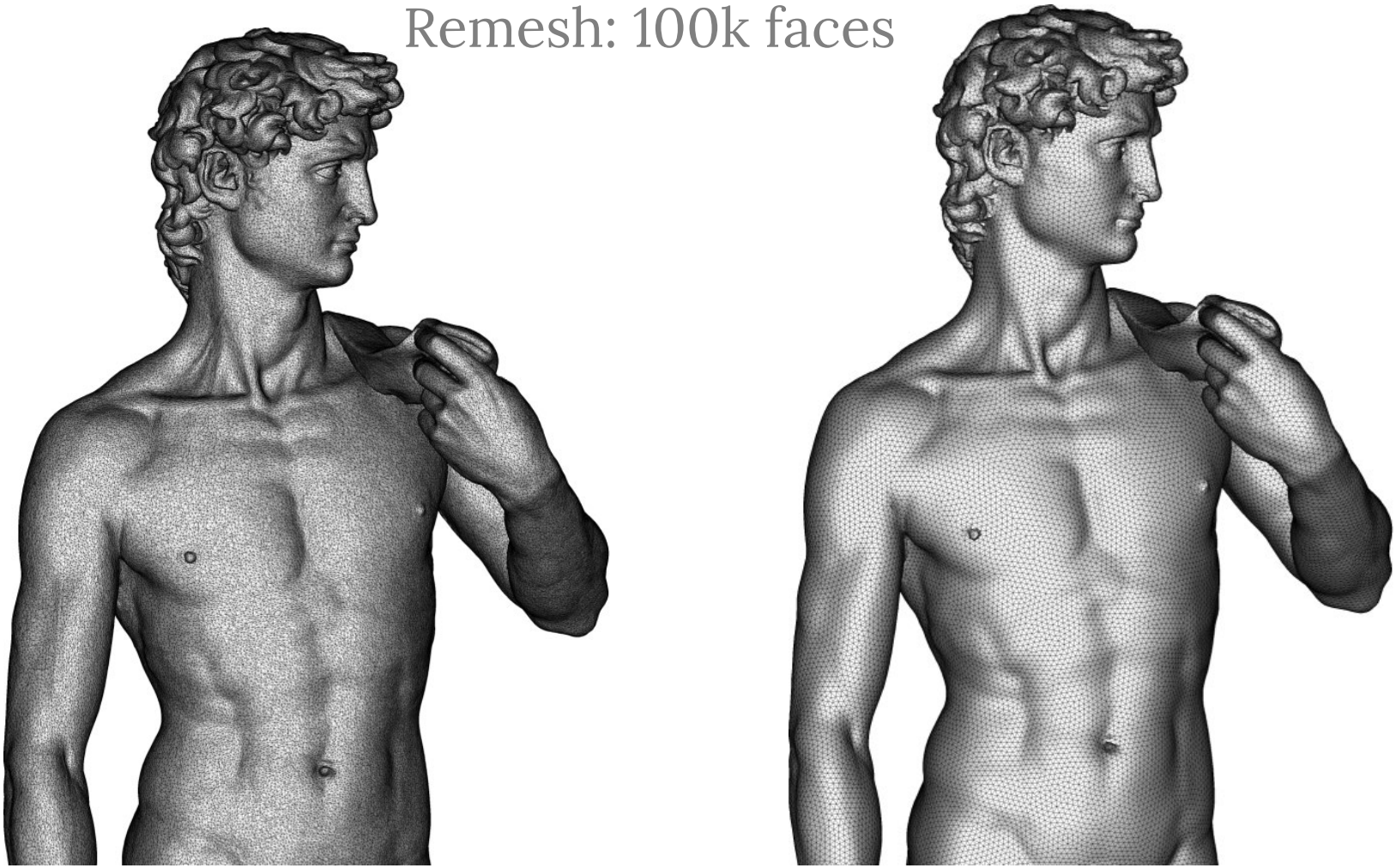
Before (avg min 30)



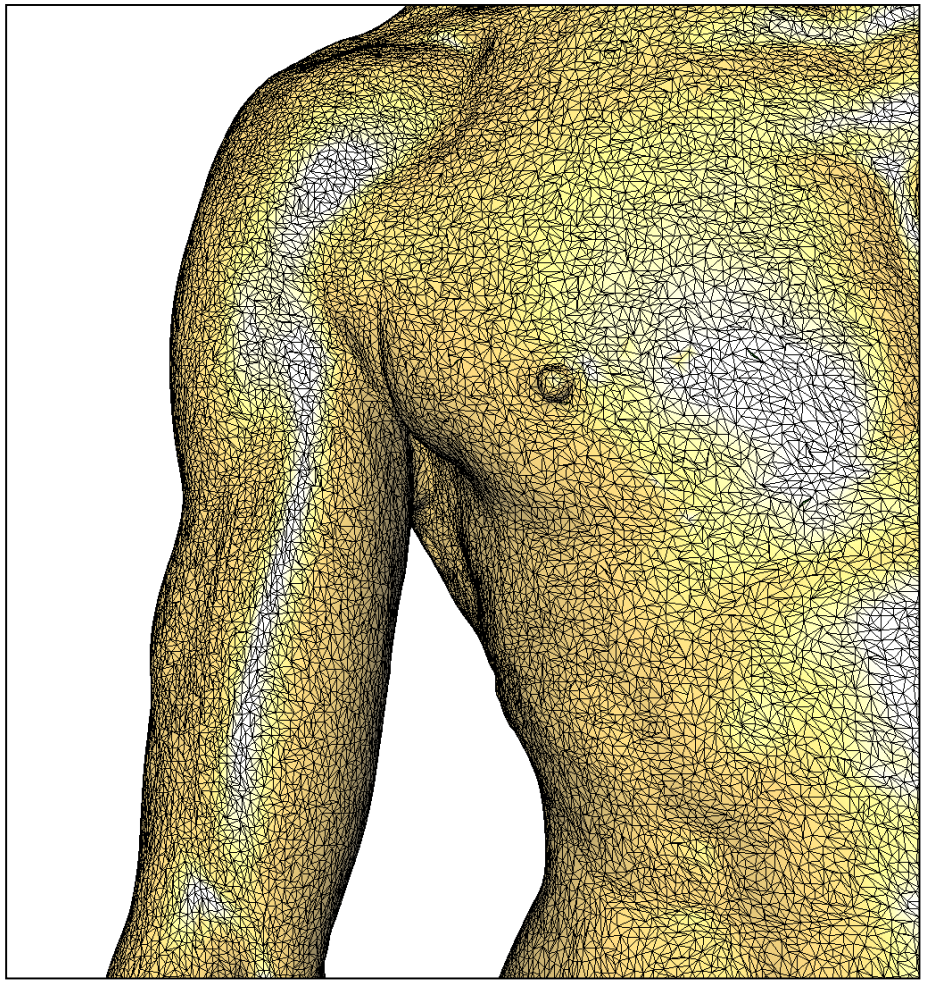
Smoothing + Flips
(avg min 45)

Michelangelo's David

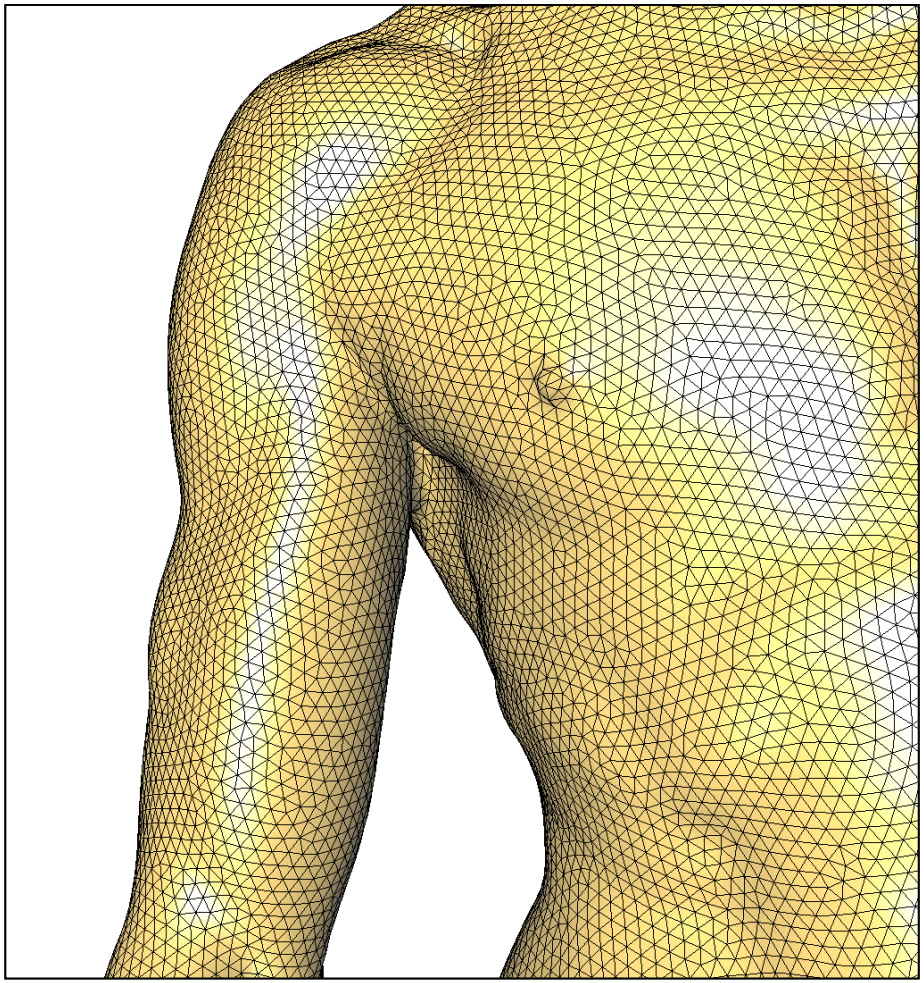
Original: 350k faces
Remesh: 100k faces



David: Zoom in



Original



Remesh

Local approach

Modify existing mesh using sequence of local operations

- Fast
- Simple to implement
- Hard to find ***good*** spacing of vertices
- Heuristic
  - How to combine local operations?

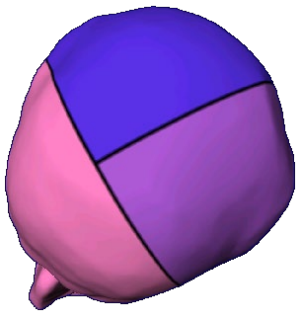
Approaches

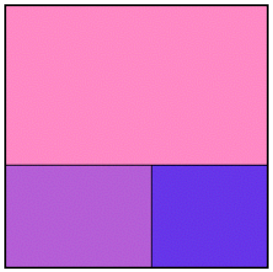
- Mesh adaptation/Local Remeshing
  - Locally update mesh while tracking error

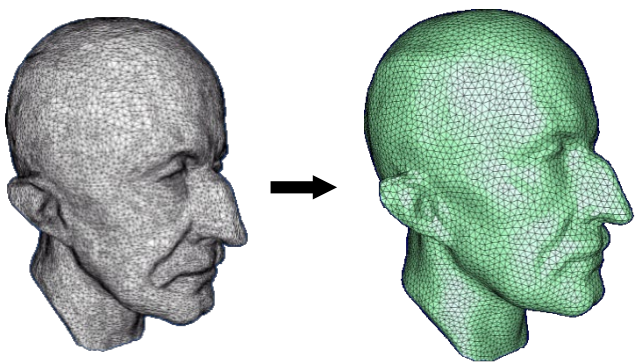
- Reduction to 2D/Global Remeshing
  - Parameterize in 2D
  - Mesh in 2D
  - Project back

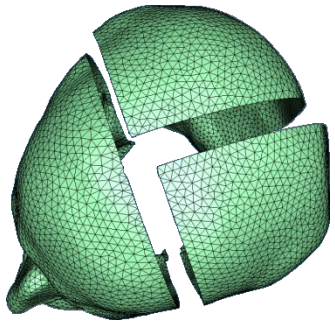
Reduction to 2D/Global Remeshing

1. Segment surface into charts
   - How? How many charts?
2. Parameterize in 2D
   - Which parameterization to choose?
3. Mesh charts in 2D (*Delaunay*)
   - Sizing ~ distortion
   - Take care of shared boundaries
4. Project back



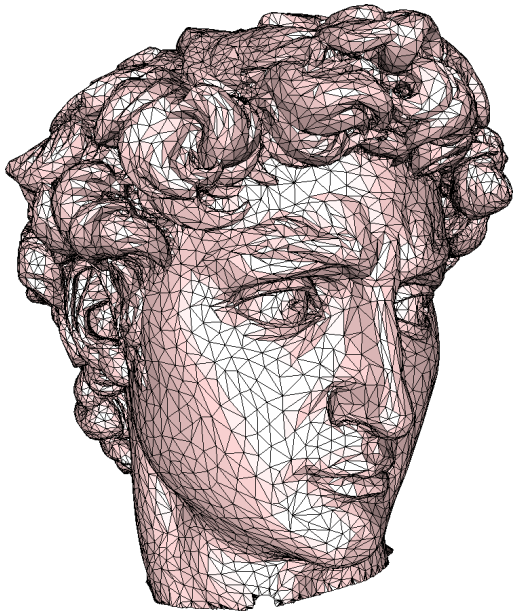






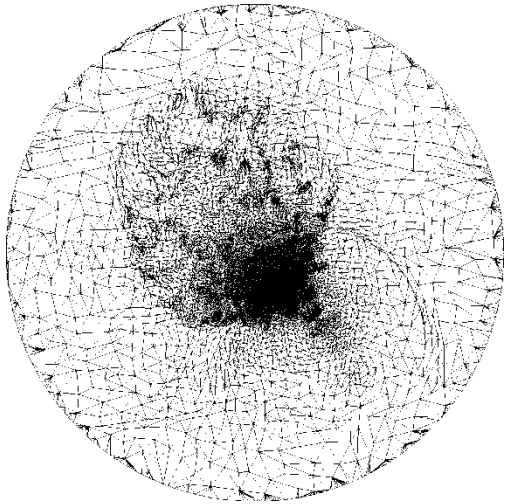
Parameterization

- Distortion is inevitable, but

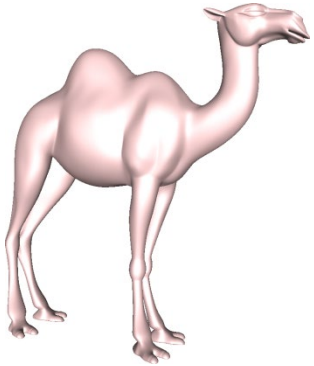


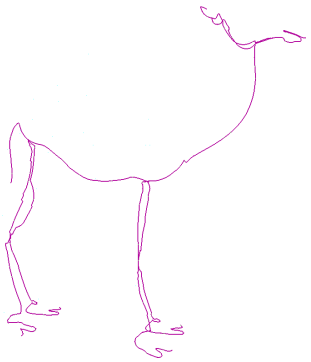
- Can handle some stretch
  - Measure & take into account during 2D meshing
    - Use as component of local sizing

→ Look for a conformal map

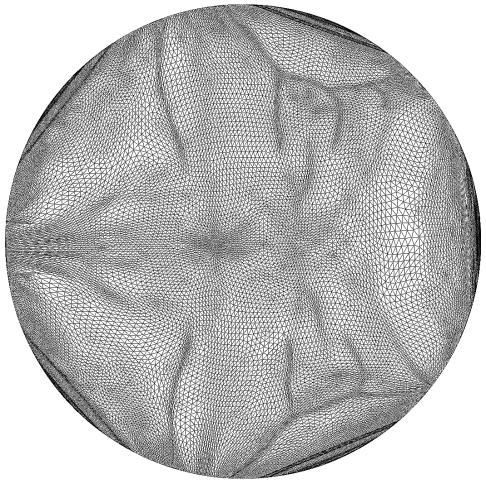


Impact of distortion

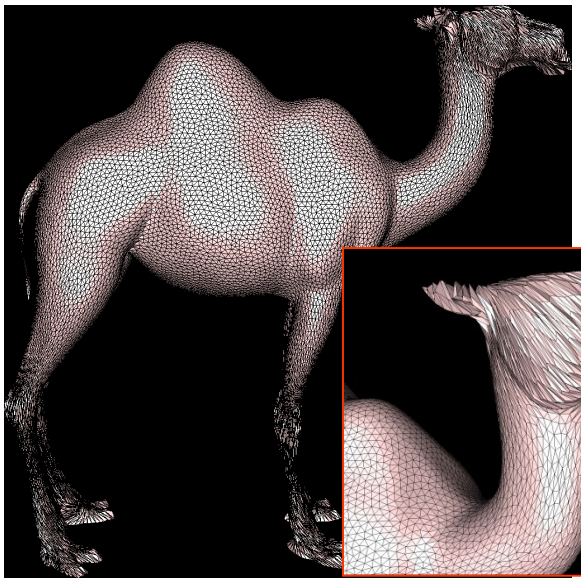


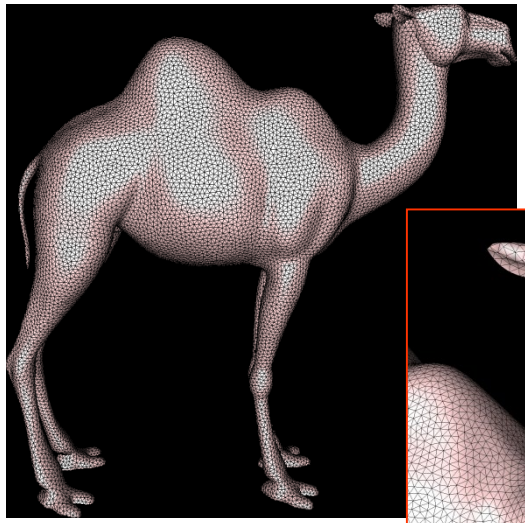


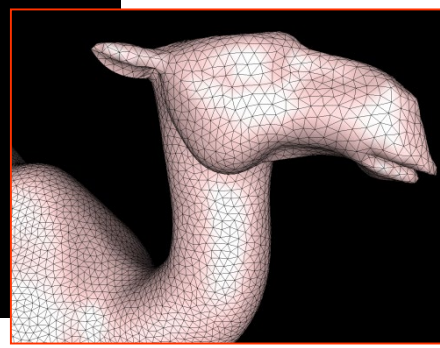
tail



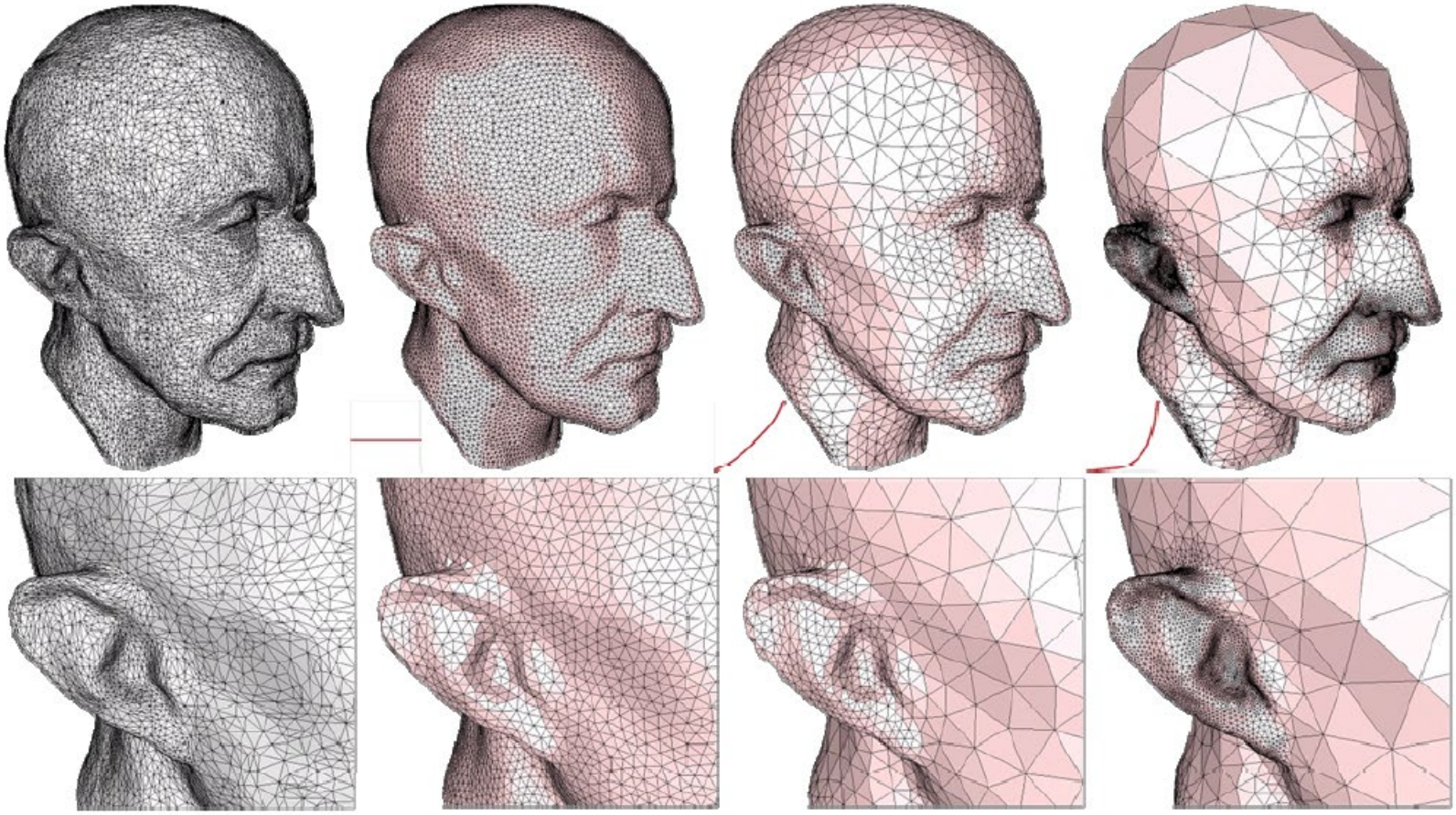
head







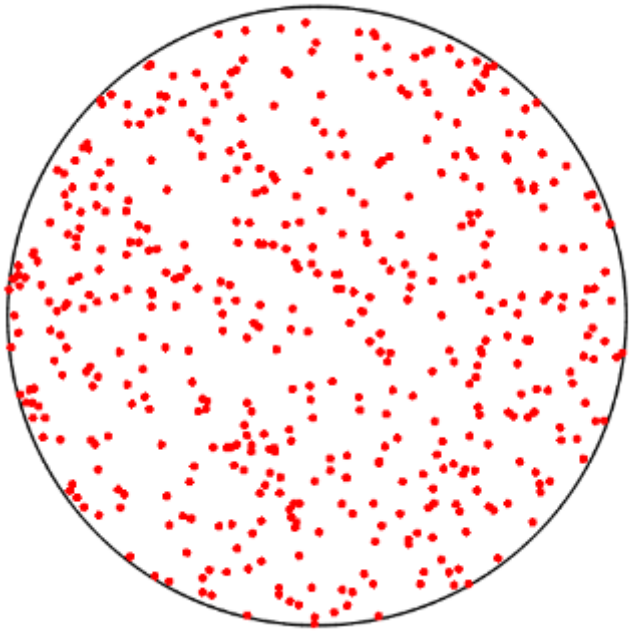
How to control sampling?



Input Uniform Non-uniform/Adaptive

How to control sampling?

- Sample random points?
  - Density ~ parameterization stretch
  - Issue?



Sampling Energy

$$E\left(\{x_i\}_{i=1,\dots,n}, \{R_i\}_{i=1,\dots,n}\right)$$
$$= \sum_{i=1,\dots,n} \int_{R_i} \|x_i - x\|^2 dx$$

Sampling Energy

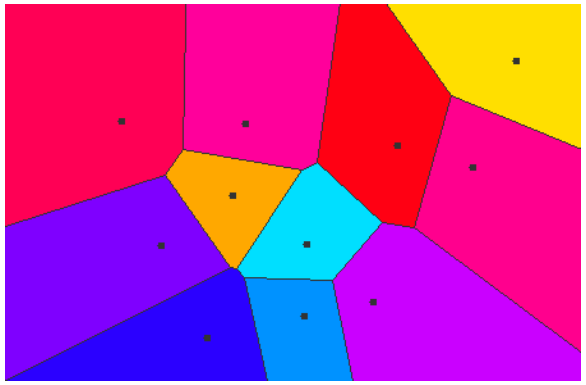
$$E\left(\{x_i\}_{i=1,\dots,n}, \{R_i\}_{i=1,\dots,n}\right)$$
$$= \sum_{i=1,\dots,n} \int_{R_i} \|x_i - x\|^2 dx$$

For fixed x_i, what are the optimal R_i?

Sampling Energy

$$E\left(\{x_i\}_{i=1,\dots,n}, \{R_i\}_{i=1,\dots,n}\right)$$
$$= \sum_{i=1,\dots,n} \int_{R_i} \|x_i - x\|^2 dx$$

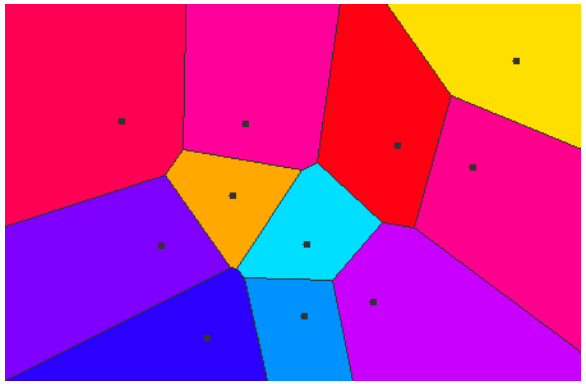
For fixed x_i, what are the optimal R_i?



Sampling Energy

$$E\left(\{x_i\}_{i=1,\dots,n}, \{R_i\}_{i=1,\dots,n}\right)$$
$$= \sum_{i=1,\dots,n} \int_{R_i} \|x_i - x\|^2 dx$$

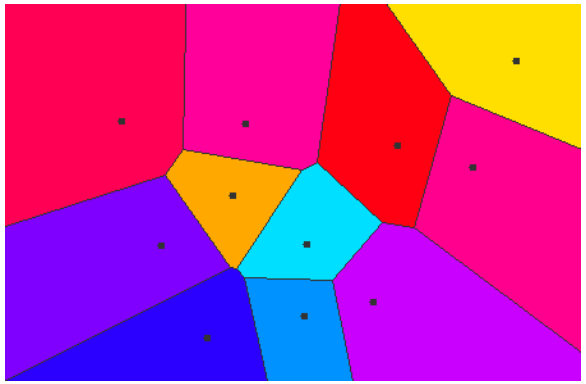
Vice-versa?



Sampling Energy

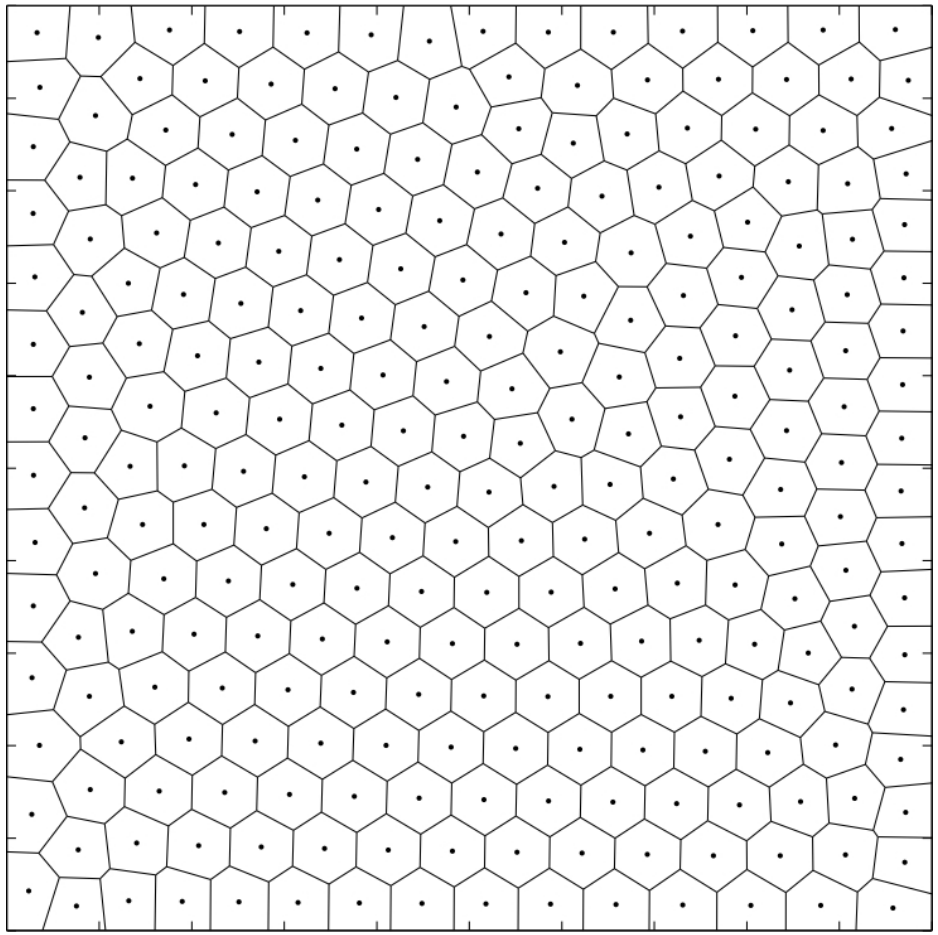
$$E\left(\{x_i\}_{i=1,\dots,n}, \{R_i\}_{i=1,\dots,n}\right)$$
$$= \sum_{i=1,\dots,n} \int_{R_i} \|x_i - x\|^2 dx$$

Global optimum: a Voronoi tessellation with sites = centroids of Voronoi cells



Centroidal Voronoi Diagram

Points spread evenly

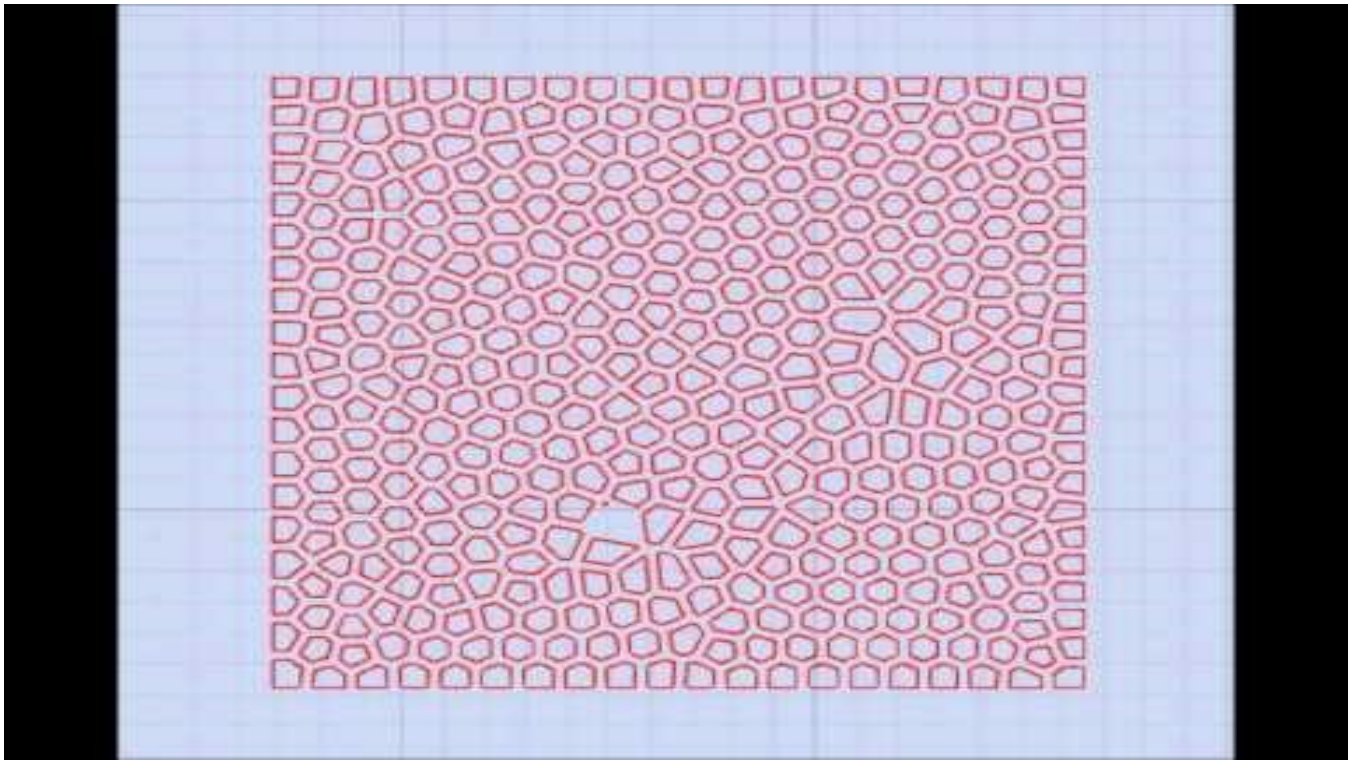


https://maxhalford.github.io/blog/halftoning-with-go---part-2/

Centroidal Voronoi Diagram

Alternate two steps:

1. Compute Voronoi cells
2. Move sites to their centroids



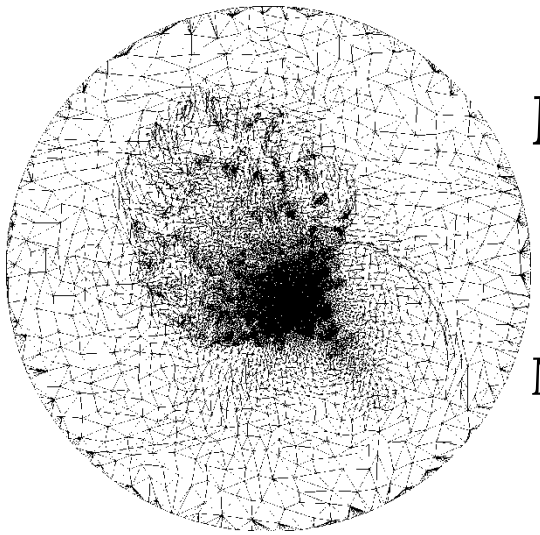
Centroidal Voronoi Diagram

Alternate two steps:

1. Compute Voronoi cells
2. Move sites to their centroids

Lloyd iterations
Same as in k-means clustering

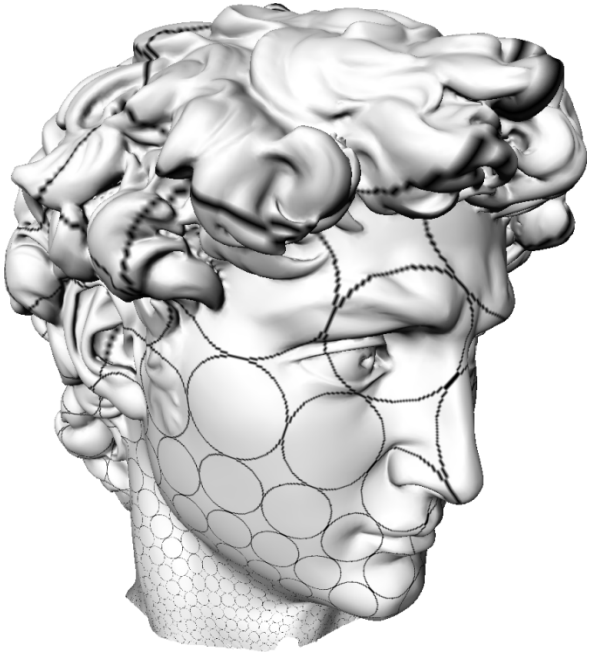
Meshing - sizing

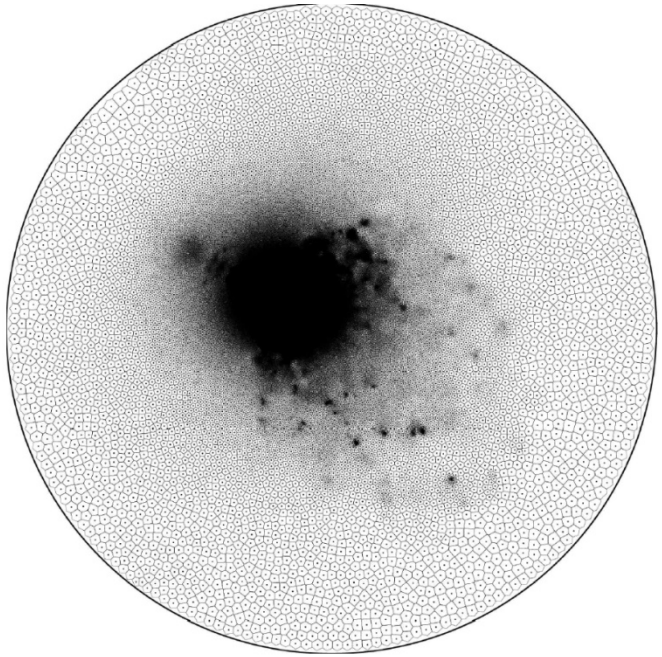


Measure parametric stretch (3D to 2D)

- Measure stretch per edge $||e_{3D}||/||e_{2D}||$
- Vertex stretch = average of edges

Multiply sizing function (at vertices) by stretch



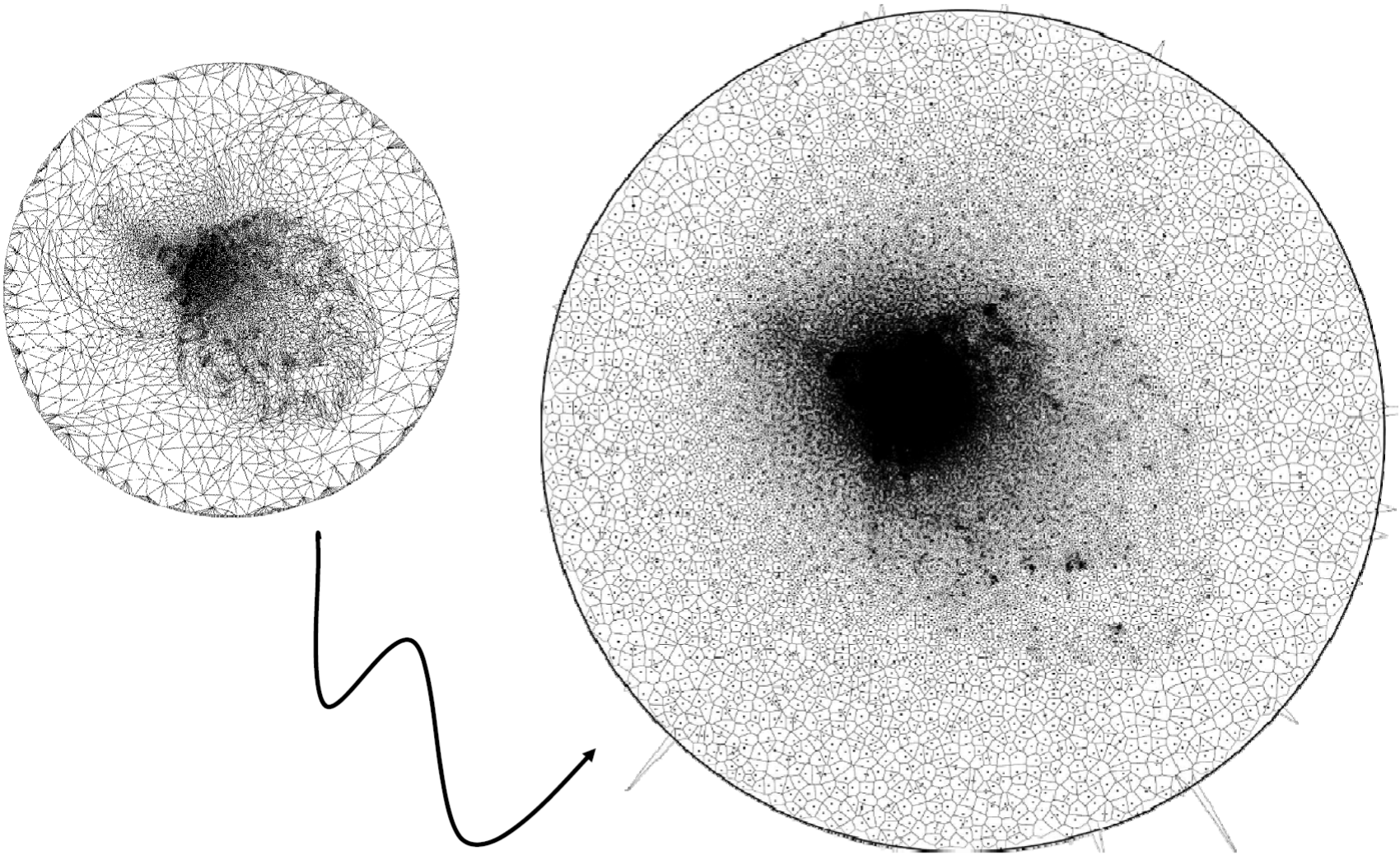


Non-uniform density

$$E\left(\{x_i\}_{i=1,\dots,n}, \{R_i\}_{i=1,\dots,n}\right)$$

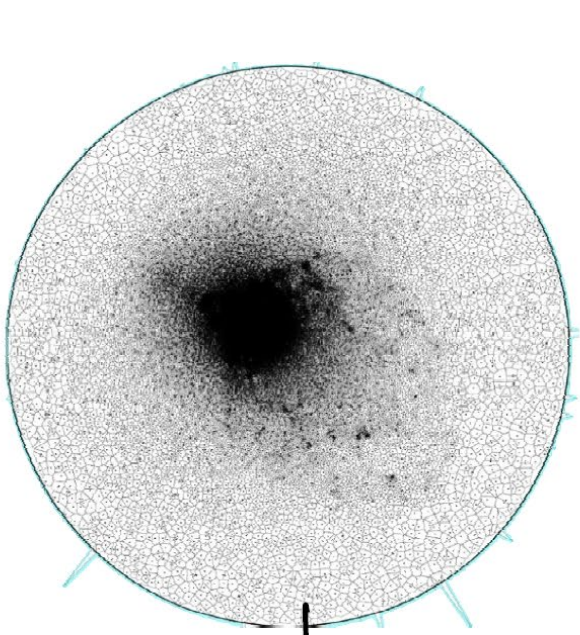
$$= \sum_{i=1,\dots,n} \int_{R_i} \rho(x) \|x_i - x\|^2 dx$$

Non-uniform density

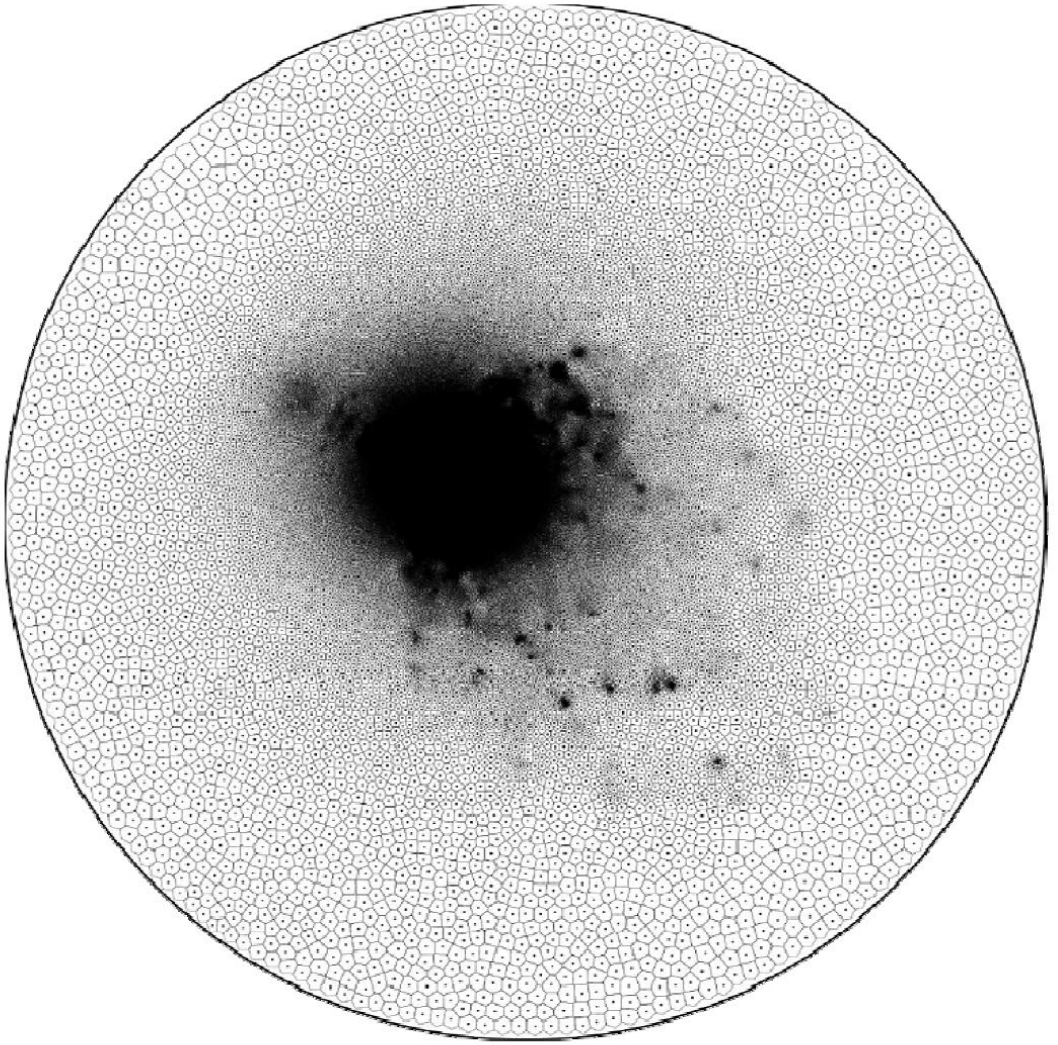


Voronoi tesselation

Non-uniform density

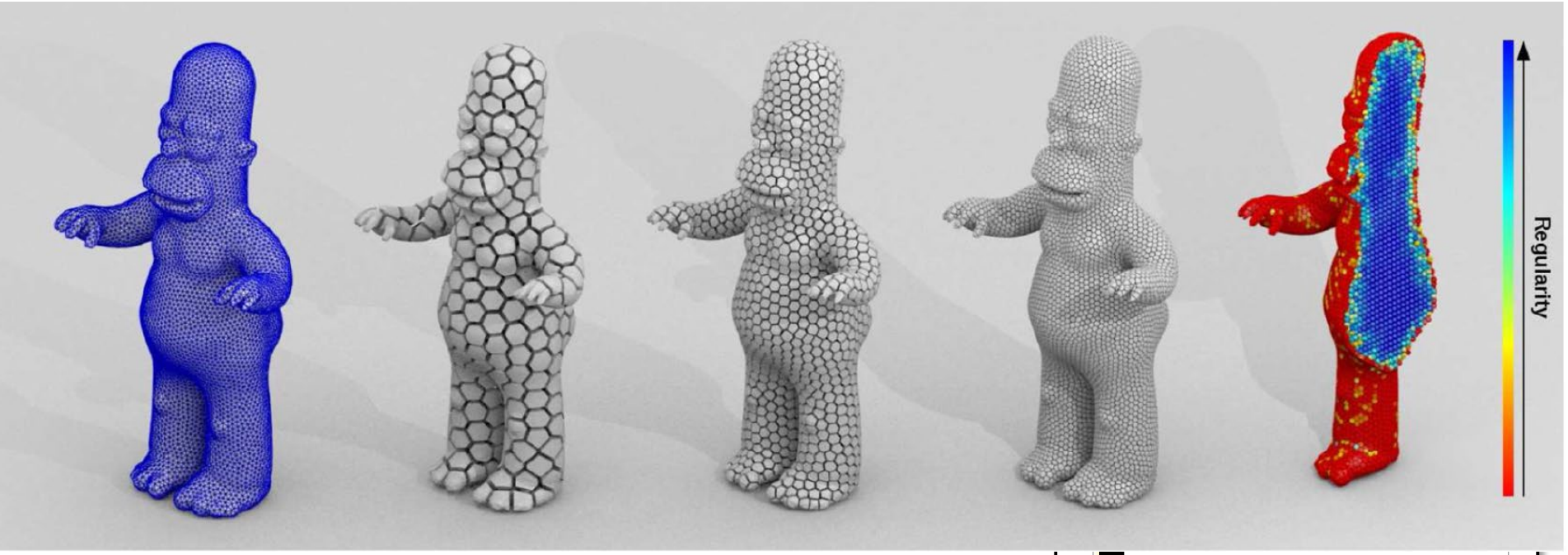


Lloyd



centroidal Voronoi tesselation

3D



DOI: 10.1111/cgf.12716

COMPUTER GRAPHICS *forum*
Volume 35 (2016), number 1 pp. 152–165

A Hierarchical Approach for Regular Centroidal Voronoi Tessellations

L. Wang, F. Hétroy-Wheeler and E. Boyer

Univ. Grenoble Alpes & Inria & CNRS, LJK, Grenoble, France
li.wang@inria.fr, Franck.Hetroy@grenoble-inp.fr, edmond.boyer@inria.fr

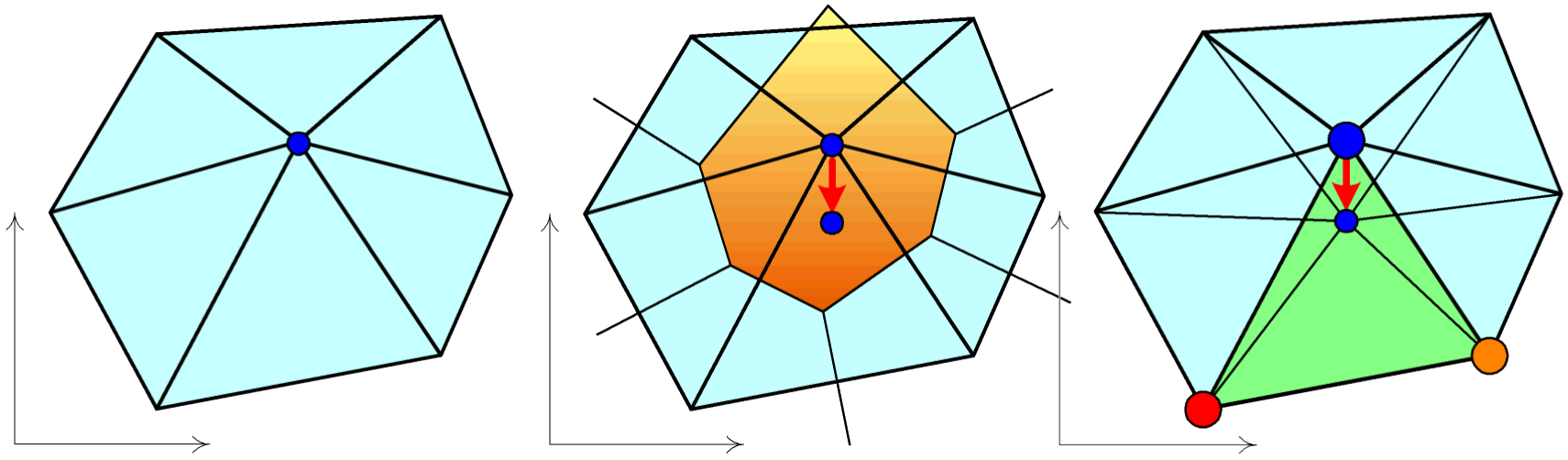
In this paper, we consider Centroidal Voronoi Tessellations (CVTs) and study their regularity. CVTs are geometric structures that enable regular tessellations of geometric objects and are widely used in shape modelling and analysis. While several efficient iterative schemes, with defined local convergence properties, have been proposed to compute CVTs, little attention has been paid to the evaluation of the resulting cell decompositions. In this paper, we propose a regularity criterion that allows us to evaluate and compare CVTs independently of their sizes and of their cell numbers. This criterion allows us to compare CVTs on a common basis. It builds on earlier theoretical work showing that second moments of cells converge to a lower bound when optimizing CVTs. In addition to proposing a regularity criterion, this paper also considers computational strategies to determine regular CVTs. We introduce a hierarchical framework that propagates regularity over decomposition levels and hence provides CVTs with provably better regularities than existing methods. We illustrate these principles with a wide range of experiments on synthetic and real models.

Keywords: categories, subject descriptors

ACM CCS: Computer Graphics [I.3.5]: Computational Geometry and Object Modelling—Curve, surface, solid and object representations; I.5.3 [Pattern Recognition]: Clustering—

Smoothing: Centroidal Voronoi Diagram

- Relocate vertices (smoothing) to control sizing (sampling)
- Lloyd algorithm on surface mesh
  - On 2D umbrella compute VD of vertex + neighbors
  - Place vertex at center of mass of it's cell
  - Repeat



Alternative: Blue noise

Blue Noise through Optimal Transport

Fernando de Goes, Caltech
Katherine Breeden, Stanford
Victor Ostromoukhov, Lyon 1 U./CNRS
Mathieu Desbrun, Caltech

Abstract

We present a fast, scalable algorithm to generate high-quality blue noise point distributions of arbitrary density functions. At its core is a novel formulation of the recently-introduced concept of capacity-constrained Voronoi tessellation as an optimal transport problem. This insight leads to a continuous formulation able to enforce the capacity constraints exactly, unlike previous work. We exploit the variational nature of this formulation to design an efficient optimization technique of point distributions via constrained minimization in the space of power diagrams. Our mathematical, algorithmic, and practical contributions lead to high-quality blue noise point sets with improved spectral and spatial properties.

CR Categories: I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—Sampling.

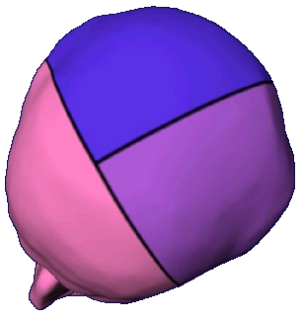
Keywords: Blue noise, power diagram, capacity constraints.

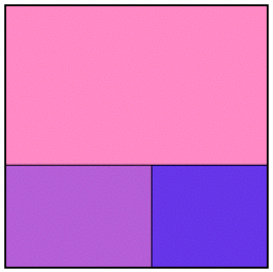
Links: DL PDF WEB CODE

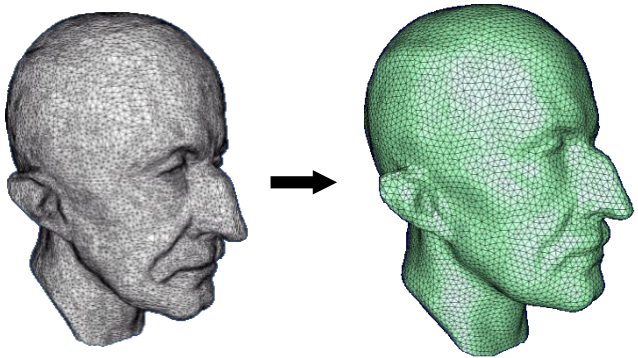
Alternative: Blue noise

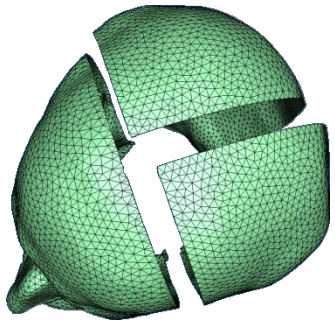
Reduction to 2D/Global Remeshing

1. **Segment surface into charts**
   - How? How many charts?
2. Parameterize in 2D
   - Which parameterization to choose?
3. Mesh charts in 2D (*Delaunay*)
   - Sizing ~ distortion
   - Take care of shared boundaries
4. Project back









Segmentation

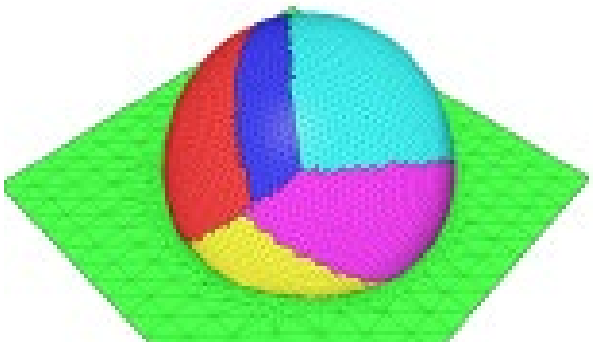
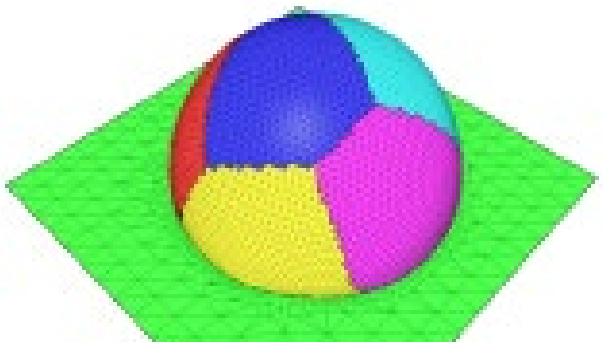
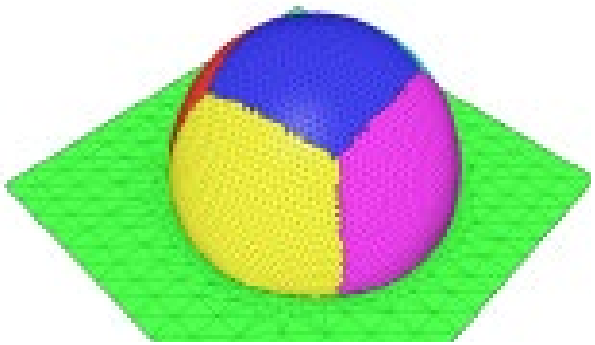
- Chart Properties
  - Disk topology
  - Low distortion
    - Ideal: Developable charts
- Approaches
  - Single chart
    - Generate (short) cuts to reduce genus
    - Cut through high curvature/distortion vertices
  - Multiple charts
    - More convex boundaries – easier to handle

Lloyd Iterations

for segmentation

Initialization: select random triangles = seeds

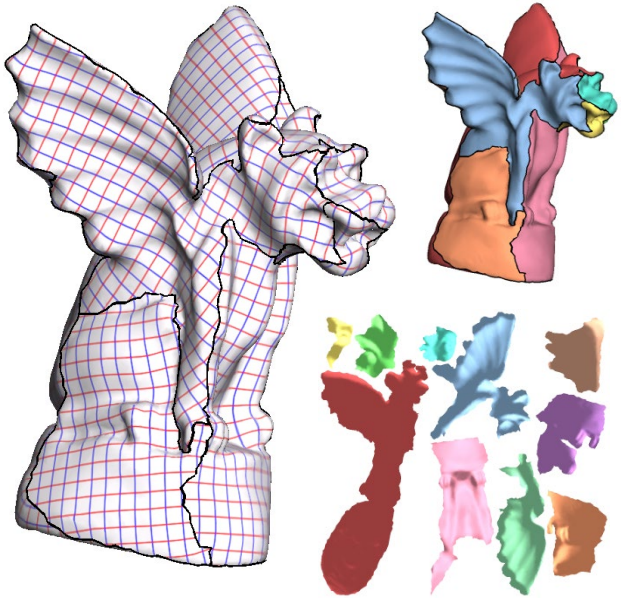
1. Grow charts around seeds greedily
2. Find new seed for each chart
   – E.g. centroid
3. Repeat

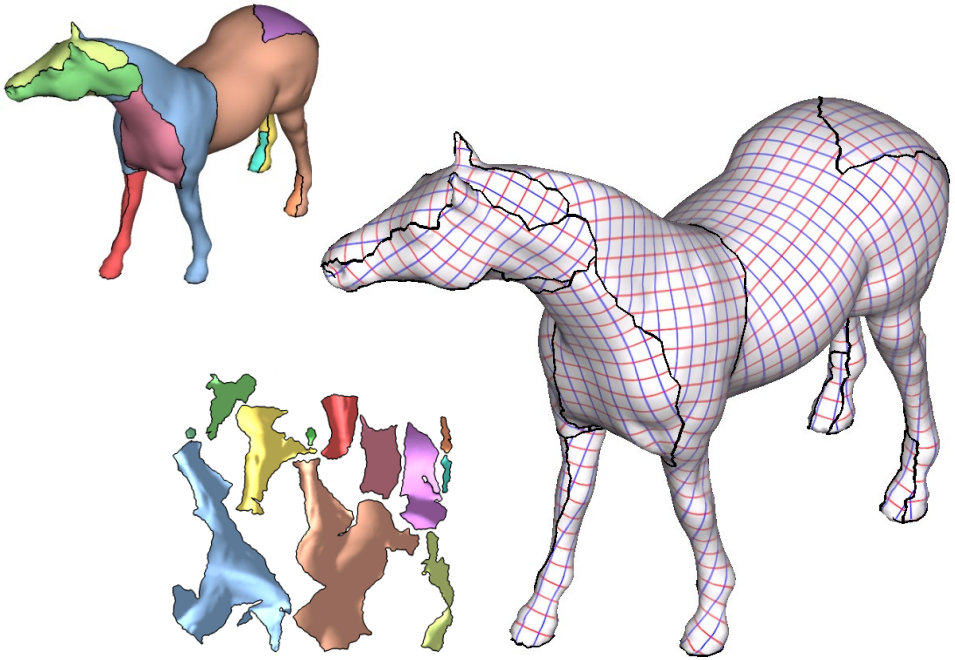




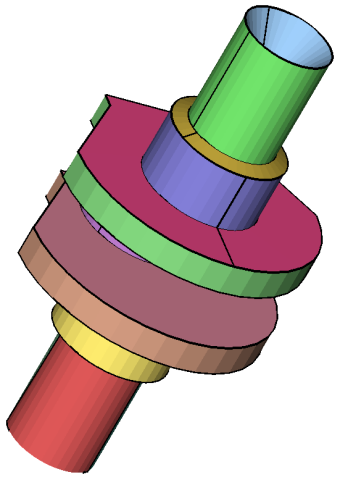
Proxies

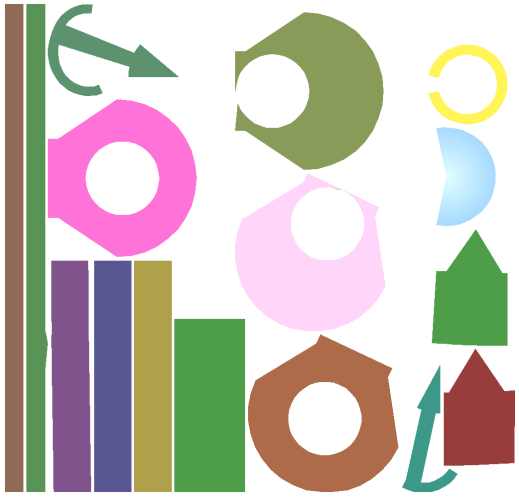
- Charts represented by proxies – used for reseeding and growth
- Example: Planar charts
  - Proxy: Normal to plane N_c
  - Compute: Average normal of chart triangles
  - Growth metric: Normal difference $N_C \cdot n_t$

Example Results









Related: zippables

Shape Representation by Zippables

CHRISTIAN SCHÜLLER, ROI PORANNE, and OLGA SORKINE-HORNUNG, ETH Zurich, Switzerland

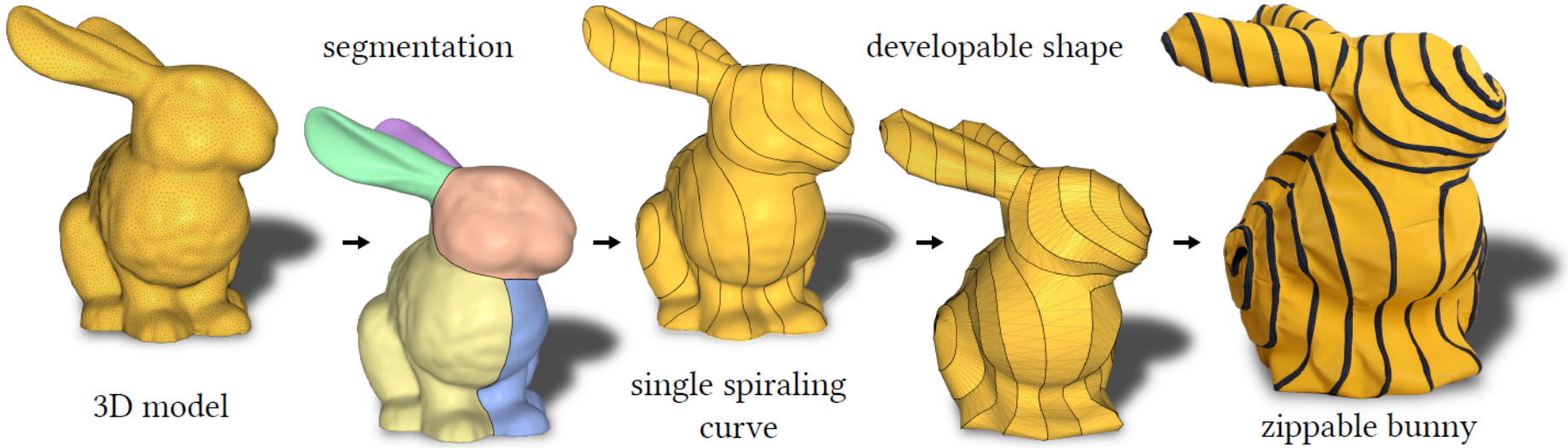
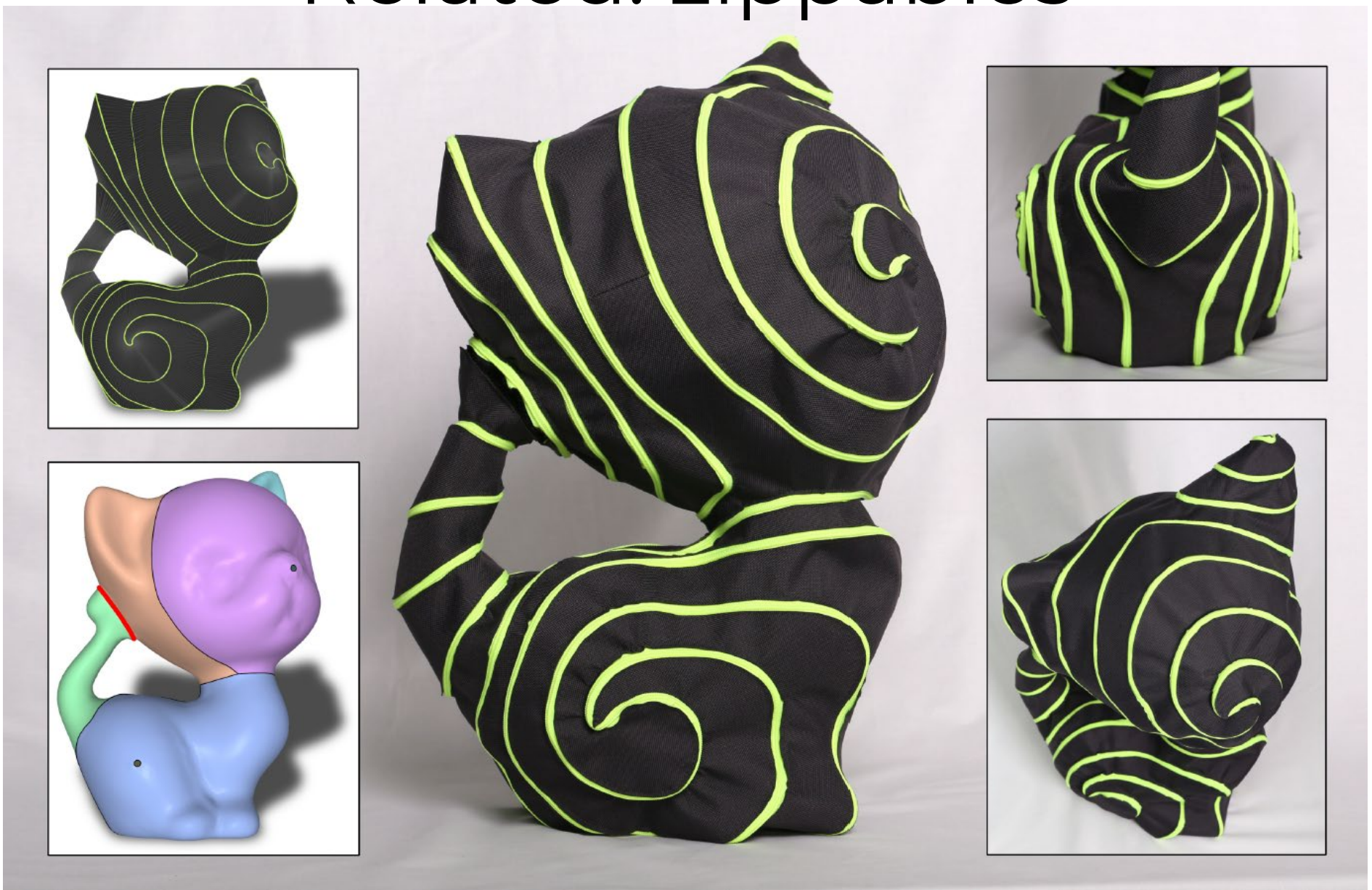


Fig. 1. The pipeline of our approach. Starting from a 3D model, the user decomposes the shape into topological cylinders. Our algorithm automatically produces a single continuous curve on the shape that spirals along the cylinders. It proceeds to cut the shape along the curve and creates a developable surface that can be trivially unfolded into a single 2D shape – the so called *zippable*. Based on the flattening, plans for laser cutting it from fabric are generated. Finally, we attach a zipper with a single slider to the boundary of the zippable. Zipping it up reproduces a faithful approximation of the input model.

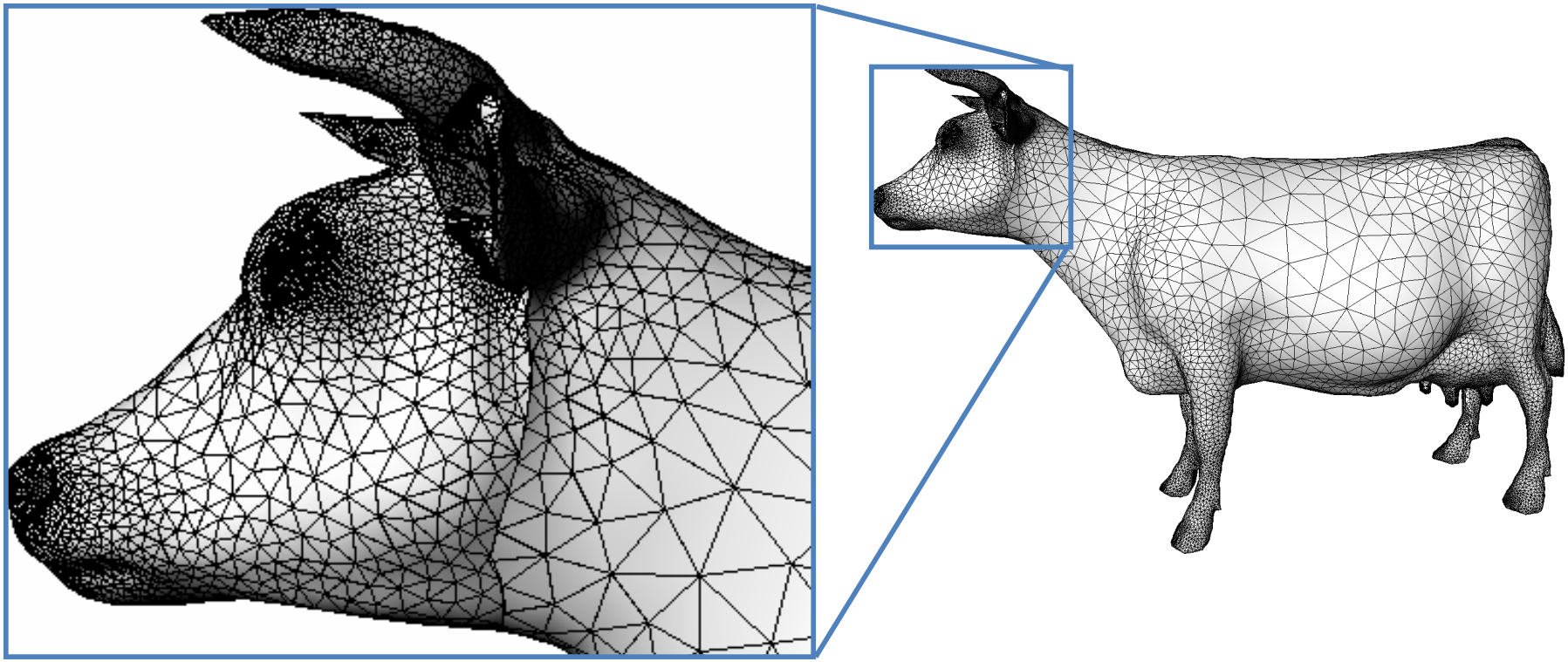
Related: zippables



Boundary

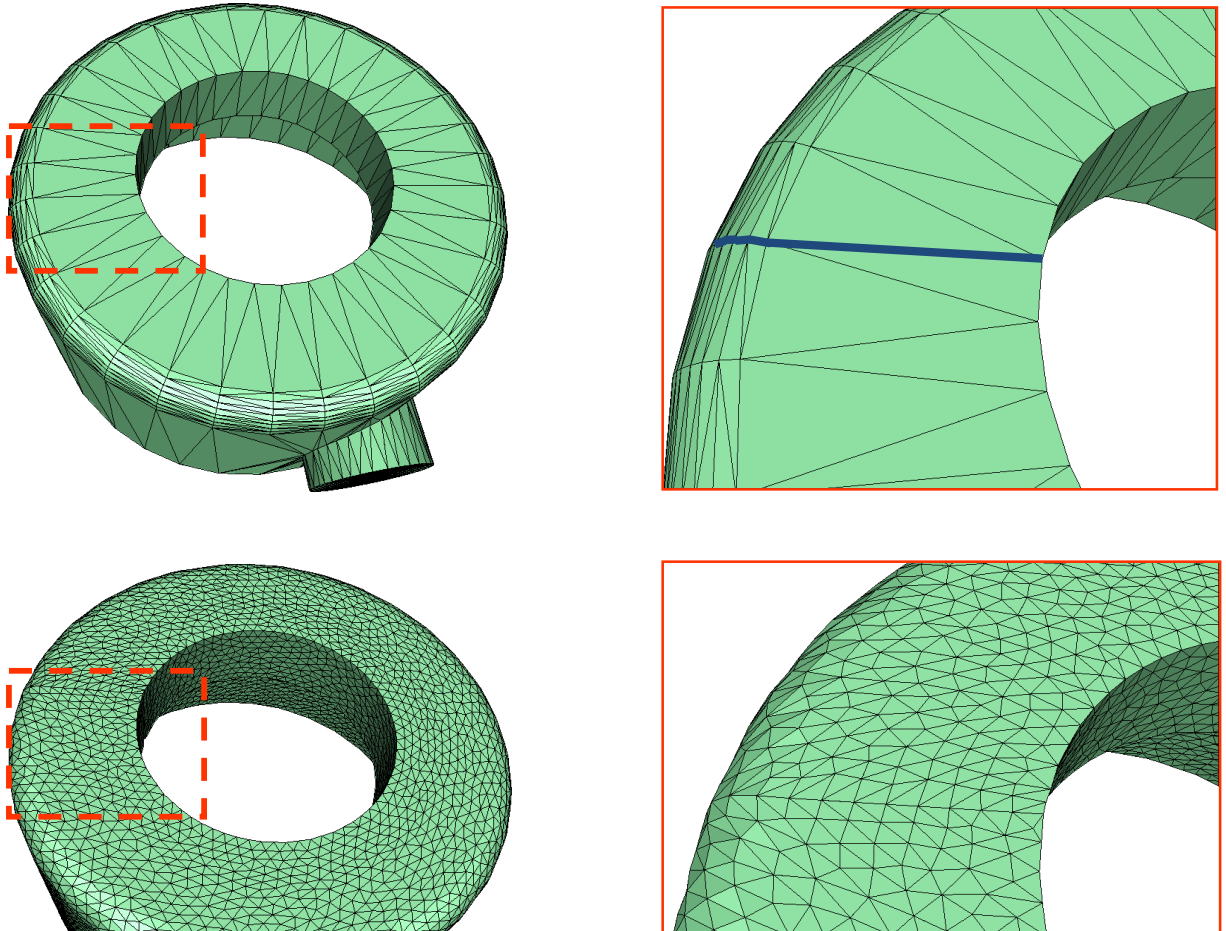
Need mesh consistency along boundaries

- Enforce shared boundary vertex positions



Boundaries

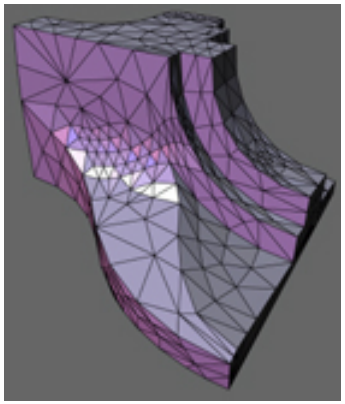
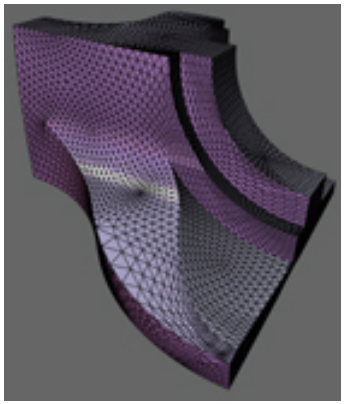
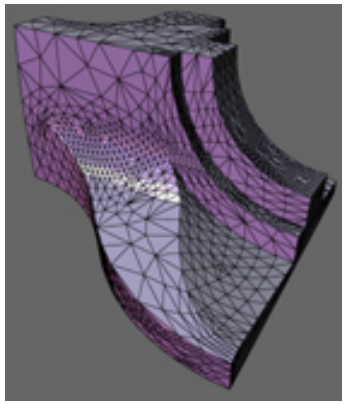
- Consistent but visible…



Features

Preserving features – locate surface creases and prevent removing them

- Special handling by segmentation and/or 2D meshing

Global Methods - Properties

- Three major components:
  - Segment
  - Parameterize
  - Mesh in 2D
- Strongly depends on parameterization quality
  - In turn depends on segmentation
- Typically more complex to implement from scratch

Tet Meshing

An active area of research!

Tetrahedral Meshing in the Wild

YIXIN HU, New York University
QINGNAN ZHOU, Adobe Research
XIFENG GAO, New York University
ALEC JACOBSON, University of Toronto
DENIS ZORIN, New York University
DANIELE PANOZZO, New York University

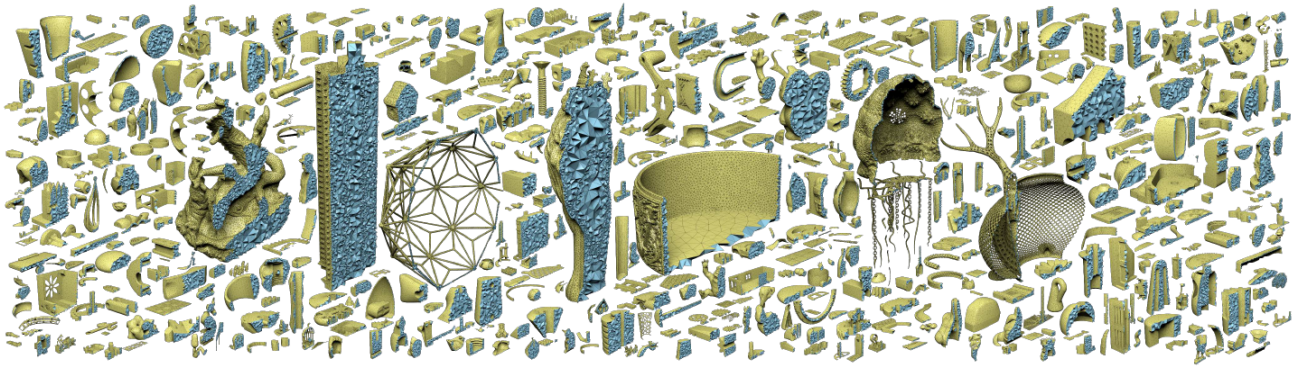


Fig. 1. A selection of the ten thousand meshes *in the wild* tetrahedralized by our novel tetrahedral meshing technique.

We propose a novel tetrahedral meshing technique that is unconditionally robust, requires no user interaction, and can directly convert a triangle soup into an analysis-ready volumetric mesh. The approach is based on several core principles: (1) initial mesh construction based on a fully robust, yet efficient, filtered exact computation (2) explicit (automatic or user-defined)

ACM Reference Format:
Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild . *ACM Trans. Graph.* 37, 4, Article 60 (August 2018), 14 pages. https://doi.org/10.1145/3197517.3201353

Hex Meshing

All-Hex Mesh Generation via Volumetric PolyCube Deformation

James Gregson[1], Alla Sheffer[1] and Eugene Zhang[2]

[1]University of British Columbia, Canada
[2]Oregon State University, United States

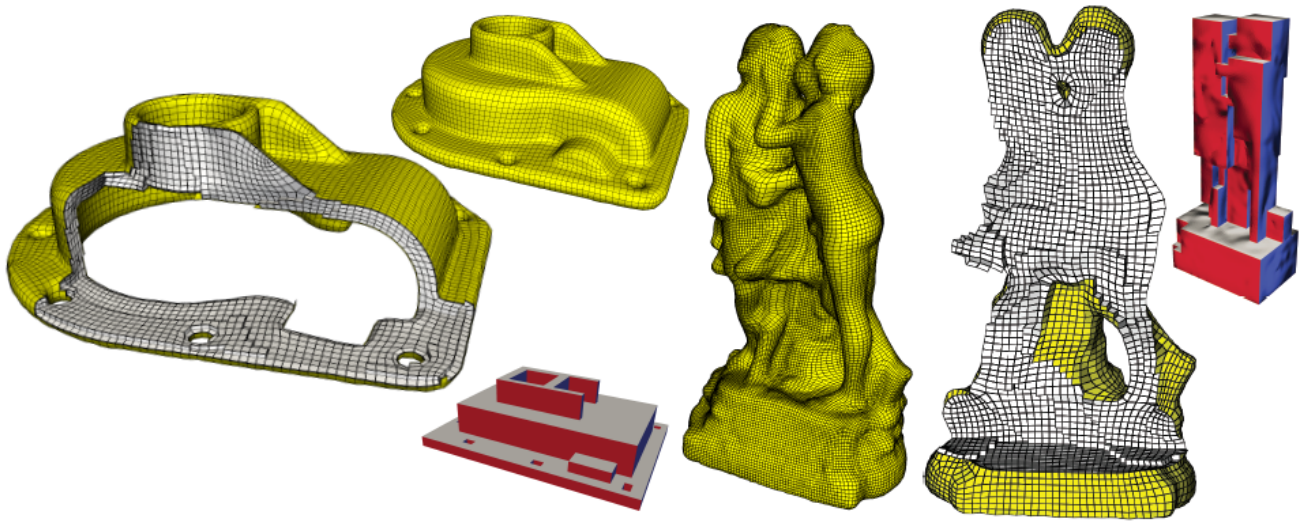
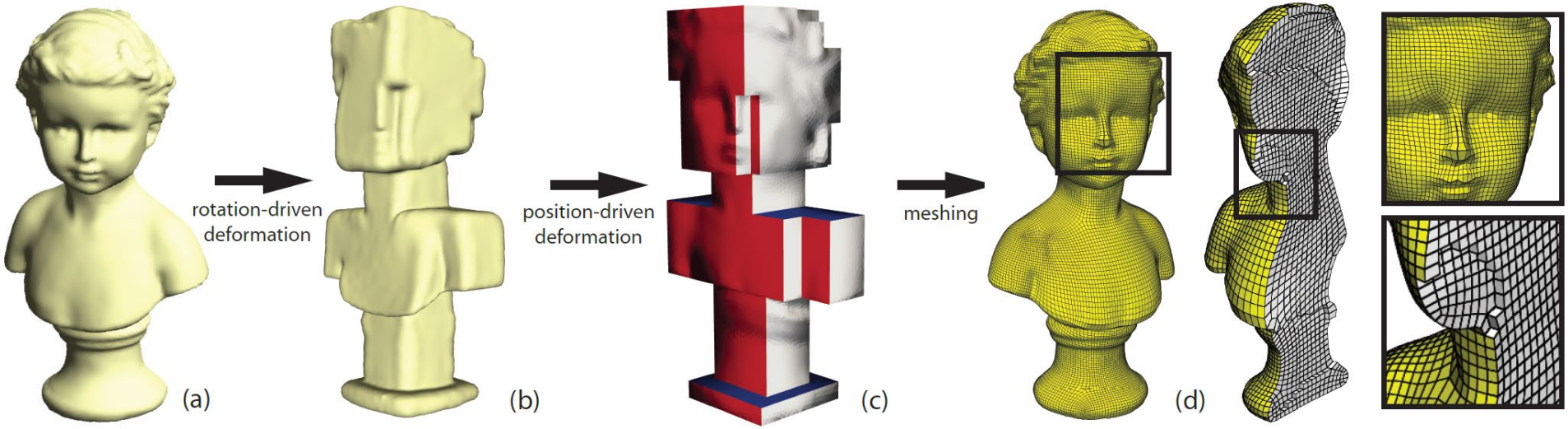


Figure 1: *High quality all-hex meshes of complex shapes automatically generated by our method and the PolyCubes we compute to create them. For the* kiss *both fine and coarse meshes are shown.*

Hex Meshing


rotation-driven
deformation
position-driven
deformation
meshing
(a)
(b)
(c)
(d)

How to (re)mesh surfaces?

Delaunay triangulation?

- What is Delaunay criterion on surface?
  - Option 1: Use sphere instead of circle
    - Works for volumetric meshes (tets)
  - Option 2: Use pairwise test only
    - Theoretical Delaunay properties?
  - Option 3: Intrinsic Delaunay
- Boundary recovery = Approximation quality

Intrinsic Delaunay

Discrete Comput Geom (2007) 38: 740–756
DOI 10.1007/s00454-007-9006-1

A Discrete Laplace–Beltrami Operator for Simplicial Surfaces

Alexander I. Bobenko · Boris A. Springbo

Received: 8 September 2005 / Revised: 24 February 200
Published online: 6 September 2007

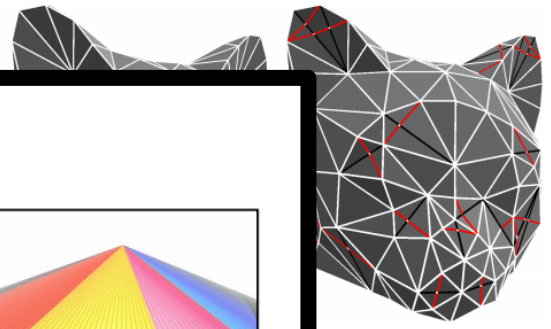
Abstract We define a
(Definition 16). It depe
edge weights are positiv
elements Laplacian (th
intrinsic Delaunay trian
tions of discrete harmo
surfaces. The definition

An Algorithm for the Construction of Intrinsic Delaunay Triangulations with Applications to Digital Geometry Processing

Matthew Fisher, Caltech
Boris Springborn, TU Berlin
Peter Schröder, Caltech
Alexander I. Bobenko, TU Berlin

Abstract

The discrete *Laplace-Beltrami* operator plays a prominent role in



d) surface as defined by the intrinsic Delaunay triangu-

Navigating Intrinsic Triangulations

NICHOLAS SHARP, Carnegie Mellon University
YOUSUF SOLIMAN, Caltech
KEENAN CRANE, Carnegie Mellon University

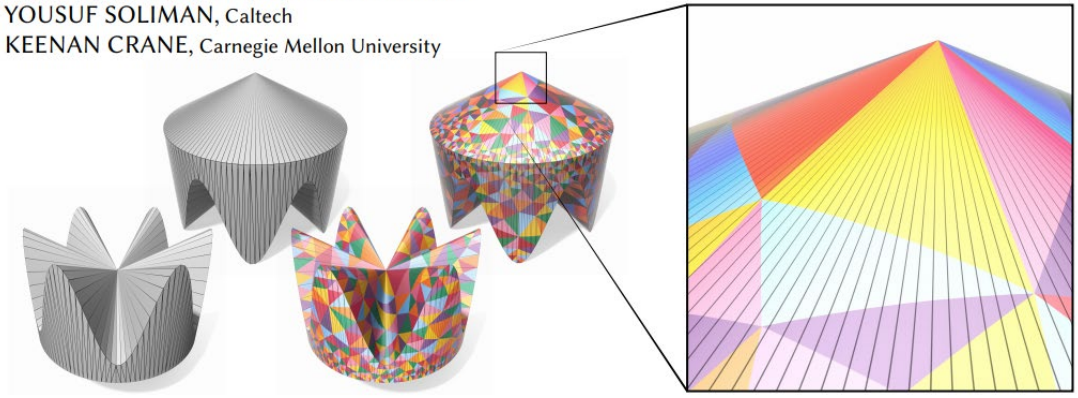
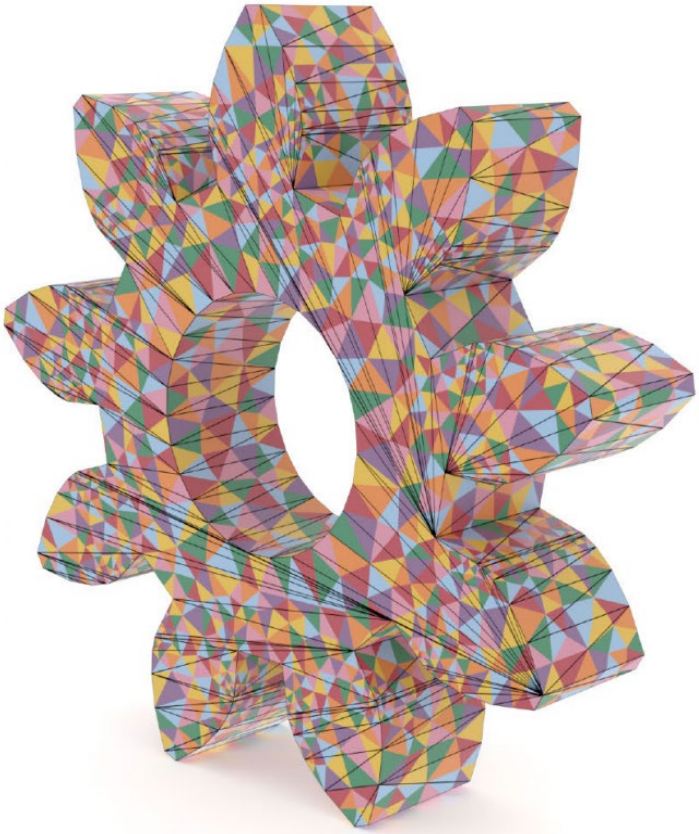


Fig. 1. Our data structure makes it possible to treat a crude input mesh *(left)* as a high-quality *intrinsic triangulation (right)* while exactly preserving the original geometry. Existing algorithms can be run directly on the new triangulation as though it is an ordinary triangle mesh. Here, a mesh with tiny input angles becomes a geometrically identical Delaunay triangulation with angles no smaller than 30°—a feat impossible for traditional, extrinsic remeshing.

Intrinsic Delaunay

- Idea: keep the geometry!
- Use Delaunay criterion for curvilinear triangles
- Edges = geodesics (locally shortest paths)



Intrinsic Delaunay

- Idea: keep the geometry!
- Use Delaunay criterion for curvilinear triangles
- Edges = geodesics (locally shortest paths)
- Generate = flips