Re-Meshing Surfaces

Generate a better mesh close to the original surface
Motivation

- Numerical stability
- Easier modeling
- Quality requirements
What’s a good mesh?
On mesh approximate by

– measuring vertex to surface distance
– measuring vertex to vertex distance

• Computation complexity?
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?
  – 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
Local approach

- Remove short edges
  - Edge Collapse

- Remove long edges
  - Edge Split

- Adjust vertex valences
  - Edge Flip

- Smooth
  - Vertex Relocation
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
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?
Measuring Error

Hausdorff is expensive => cheat

Idea 1: Stay within an $\varepsilon$-envelope
Measuring Error

Hausdorff is expensive => cheat

Idea 1: Stay within an $\varepsilon$-envelope

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)

Second round of flips (avg min 37)

After (avg min 33)
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
Edge Collapse Algorithm

- Simplification operation:
  - Edge collapse (pair contraction)

- Error metric:
  - Distance, pseudo-global

Where should we place the vertex?
Where to place the new vertex?

- Edge Collapse
- Edge Split
- Vertex Relocation
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
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?
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
How to control sampling?

Input  Uniform  Non-uniform/Adaptive
How to control sampling?

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

\[ E(\{x_i\}_{i=1,\ldots,n}, \{R_i\}_{i=1,\ldots,n}) \]

\[ = \sum_{i=1,\ldots,n} \int_{R_i} \|x_i - x\|^2 dx \]
Sampling Energy

\[ E(\{x_i\}_{i=1,\ldots,n}, \{R_i\}_{i=1,\ldots,n}) = \sum_{i=1,\ldots,n} \int_{R_i} \|x_i - x\|^2 dx \]

For fixed \( x_i \), what are the optimal \( R_i \)?
Sampling Energy

\[
E(\{x_i\}_{i=1,\ldots,n}, \{R_i\}_{i=1,\ldots,n}) = \sum_{i=1,\ldots,n} \int_{R_i} \|x_i - x\|^2 \, dx
\]

For fixed \(x_i\), what are the optimal \(R_i\)?
Sampling Energy

\[ E(\{x_i\}_{i=1,...,n}, \{R_i\}_{i=1,...,n}) \]

\[ = \sum_{i=1,...,n} \int_{R_i} \|x_i - x\|^2 dx \]

Vice-versa?
Sampling Energy

$$E(\{x_i\}_{i=1,...,n}, \{R_i\}_{i=1,...,n})$$

$$= \sum_{i=1,...,n} \int_{R_i} ||x_i - x||^2 dx$$

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

Points spread evenly

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

\[ E(\{x_i\}_{i=1,...,n}, \{R_i\}_{i=1,...,n}) = \sum_{i=1,...,n} \int_{R_i} \rho(x) \|x_i - x\|^2 dx \]
Non-uniform density

Voronoi tesselation
Non-uniform density

Lloyd

centroidal Voronoi tessellation
3D
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: 1.4.1 [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 $N_c \cdot n_t$
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

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)

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

ACM Reference Format:
Hex Meshing

All-Hex Mesh Generation via Volumetric PolyCube Deformation

James Gregson\textsuperscript{1}, Alla Sheffer\textsuperscript{1} and Eugene Zhang\textsuperscript{2}

\textsuperscript{1}University of British Columbia, Canada
\textsuperscript{2}Oregon State University, United States

\textbf{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