08 - Single Patch Parametrization

Acknowledgements: Olga Sorkine-Hornung
Geometry Processing Pipeline

Scanning: results in range images

Registration: bring all range images to one coordinate system

Stitching/reconstruction: Integration of scans into a single mesh

Processing: - Topological and geometric filtering
             - Remeshing
             - Compression
Geometry Processing Pipeline

Scanning: results in range images

Registration: bring all range images to one coordinate system

Stitching/reconstruction: Integration of scans into a single mesh

Processing:
- Topological and geometric filtering
- Remeshing
- Compression
Geometry Processing Pipeline

Scanning: results in range images

Registration: bring all range images to one coordinate system

Stitching/reconstruction: Integration of scans into a single mesh

Processing:
- Topological and geometric filtering
- Remeshing
- Compression
Geometry Processing Pipeline

Scanning: results in range images

Registration: bring all range images to one coordinate system

Stitching/reconstruction: Integration of scans into a single mesh

Processing:
- Topological and geometric filtering
- Remeshing
- Compression

Stitching/reconstruction: Integration of scans into a single mesh
Geometry Processing Pipeline

Scanning: results in range images

Registration: bring all range images to one coordinate system

Stitching/reconstruction: Integration of scans into a single mesh

Processing:
- Topological and geometric filtering
- Remeshing
- Compression

Smoothing

Simplification/compression
Parameterization – What Is It?
Surface Parameterization

3D space \((x,y,z)\)

2D parameter domain \((u,v)\)

boundary

boundary
Parameterization – Definition

- Mapping $P$ between a 2D domain $\Omega$ and the mesh $S$ embedded in 3D (the inverse = flattening)
- Each mesh vertex has a corresponding 2D position:
  $$U(v_i) = (u_i, v_i)$$
- Inside each triangle, the mapping is affine (barycentric coordinates)
Parameterization – What Is It Good For??
Why Parameterization?

• Allows us to do many things in 2D and then map those actions onto the 3D surface
• It is often easier to operate in the 2D domain

• Mesh parameterization allows to use some notions from continuous surface theory
Main Application: Texture Mapping
Main Application: Texture Mapping
Texture Mapping

Image from Vallet and Levy, techreport INRIA
Normal/Bump Mapping

original mesh
4M triangles

simplified mesh
500 triangles

simplified mesh and normal mapping
500 triangles
Remeshing

“Interactive Geometry Remeshing”, Alliez et al., SIGGRAPH 2002
Compression

“Geometry images”, Gu et al., SIGGRAPH 2002
Geometry Images

cut

parametrize
Geometry Images

cut

sample
Geometry Images

\[ [r, g, b] = [x, y, z] \]
Parameterization Properties?

• What are “good” parameterizations?

• How do we define “good”?
Bijectivity

• Locally bijective (1-1 and onto): No triangles fold over.

• Globally bijective: locally bijective + no “distant” areas overlap
Bijectivity: Non-Disk Domains
Topological Cutting
Topological Cutting

A. Sheffer, J. Hart:
Seamster: Inconspicuous Low-Distortion Texture Seam Layout, IEEE Vis 2002
Segmentation
Segmentation

• D-Charts: Quasi-Developable Mesh Segmentation, D. Julius, V. Kraevoy, A. Sheffer, EUROGRAPHICS 2005

• Find patches that align to mesh features and are close to being developable surfaces
Good Cuts/Segmentations?

- Hide seams
- Small number/length of seams
- Or: make parameterization continuous across seams!
- Next lecture
How to Measure Distortion?
Measures of Local Distortion

\[ p(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}, \quad (u, v) \in \mathbb{R}^2 \]

What happens to tangent vectors?
Measures of Local Distortion

What happens to tangent vectors?

\[ p_u = \frac{\partial p(u, v)}{\partial u}, \quad p_v = \frac{\partial p(u, v)}{\partial v} \]
Measures of Local Distortion

• How do lengths and angles of tangents change?
  • First fundamental form!

\[
I = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} p_u^T p_u & p_u^T p_v \\ p_u^T p_v & p_v^T p_v \end{pmatrix}
\]

What happens to tangent vectors?
Measures of Local Distortion

• How do lengths and angles of tangents change?
  • First fundamental form!
    $\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} p_u^T p_u & p_u^T p_v \\ p_v^T p_u & p_v^T p_v \end{pmatrix}$

  Angle change
  Length change

• Area distortion: area element:
  $dA = \sqrt{EG - F^2} \, du dv$
Measures of Local Distortion

• How do lengths and angles of tangents change?
  • First fundamental form!

\[
\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} p_u^T p_u & p_u^T p_v \\ p_v^T p_u & p_v^T p_v \end{pmatrix}
\]

• The eigenvalues of \( \mathbf{I} \) tell us the maximal/minimal stretching of a tangent vector
Distortion on Triangle Meshes?

- Triangle in 3D is mapped to triangle in 2D
- Unique affine mapping

\[ P : \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad P(u) = Au + c \]

\[ \begin{bmatrix} P_u & P_v \end{bmatrix} = A \quad \text{Jacobian} \]
Distortion on Triangle Meshes?

- SVD of the Jacobian reveals directions of extreme stretching

\[ A = U \begin{pmatrix} \Gamma \\ \sigma \end{pmatrix} V^T \]

\[ P : \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad P(u) = A u + c \]

\[ \begin{bmatrix} P_u & P_v \end{bmatrix} = A \quad \Rightarrow \quad I = A^T A \]
Distortion on Triangle Meshes?

- Possible distortion measures:

\[ E(T) = \sqrt{\Gamma^2 + \sigma^2} \quad E(T) = \max \left\{ \Gamma, \frac{1}{\sigma} \right\} \]

\[ A = U \begin{pmatrix} \Gamma \\ \sigma \end{pmatrix} V^T \]

\[ P : \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad P(\mathbf{u}) = A\mathbf{u} + \mathbf{c} \]

\[ \begin{bmatrix} P_u \\ P_v \end{bmatrix} = A \]
How To Compute (Good) Parameterizations?
And Quickly?
Distortion Minimization

$$\arg\min_{(u_1,v_1), \ldots, (u_n,v_n)} \sum_T E(T)$$

Texture map
Distortion Minimization

Texture map

Kent et al ‘92
Floater 97
Sander et al ‘01
Literature for previous slide


  http://www.mn.uio.no/math/english/people/aca/michaelf/papers/param.pdf

  http://research.microsoft.com/en-us/um/people/hoppe/proj/tmpm/
Mesh Dependence

Alg. 1 - mesh-independent
Alg. 2 - ... less mesh-independent
Conformal Parameterization

- Angle preservation; circles are mapped to circles
Conformal Parameterization

- Angle preservation; circles are mapped to circles
Area Distortion vs. Angle Distortion

- Is it possible to preserve both angles and areas at the same time?
Harmonic Mapping – Idea

- Want to flatten the mesh $\rightarrow$ no curvature $\rightarrow$ Laplace operator gives zero.

$\Delta(\mathbf{u}) = 0$

need boundary constraints to prevent trivial solution;

which Laplacian operator?
(which weights?)

$\mathbf{u} = (u,v)$ domain
Convex Mapping (Tutte, Floater)

- Boundary vertices are fixed
- Convex weights

\[ \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \text{ - inner vertices} \]
\[ \mathbf{v}_{n+1}, \ldots, \mathbf{v}_N \text{ - boundary vertices} \]
Convex Mapping (Tutte, Floater)

- Boundary vertices are fixed
- Convex weights

\[ \Delta(u_i) = 0, \quad i = 1, \ldots, n \]

\[ L(u_i) = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (u_j - u_i) = 0, \quad i = 1, \ldots, n \]

\[ w_{ij} > 0 \]
Convex Mapping (Tutte, Floater)

• Solve the linear system

\[ Lu = 0 \quad u \in \mathbb{R}^{n \times 2} \]

• The values of the boundary vertices are known and thus substituted (transfer to right-hand side)
Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
Harmonic Mapping

• Inner mesh edges as springs
• Find minimum-energy state where all vertices lie in the 2D plane
Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
- Spring energy:

\[ \frac{1}{2} k_{i,j} \| u_i - u_j \|^2 \]

\[ u_i, u_j \in \mathbb{R}^2 \]
Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
- Total spring energy of the flattened mesh:

\[ E(u_1, \ldots, u_n) = \sum_{(i,j) \in \mathcal{E}} \frac{1}{2} k_{i,j} ||u_i - u_j||^2 \]
Demo

• Libigl Tutorial 501
Minimizing Spring Energy

\[ E(u_1, \ldots, u_n) = \sum_{(i,j) \in E} \frac{1}{2} k_{i,j} \| u_i - u_j \|^2 \]

\[ \frac{\partial E(u_1, \ldots, u_n)}{\partial u_i} = \sum_{j \in N(i)} k_{i,j} (u_i - u_j) = 0 \]

\[ \sum_{j \in N(i) \cap B} k_{i,j} u_i + \sum_{j \in N(i) \setminus B} k_{i,j} (u_i - u_j) = \sum_{j \in N(i) \cap B} k_{i,j} u_j \]

unknown flat vertex positions

known fixed boundary positions

\( v_1, v_2, \ldots, v_n \) - inner vertices
\( v_{n+1}, \ldots, v_N \) - boundary vertices
Minimizing Spring Energy

- Sparse linear system of $n$ equations to solve!

\[
\sum_{j \in \mathcal{N}(i) \cap \mathcal{B}} k_{i,j} u_i + \sum_{j \in \mathcal{N}(i) \setminus \mathcal{B}} k_{i,j} (u_i - u_j) = \sum_{j \in \mathcal{N}(i) \cap \mathcal{B}} k_{i,j} u_j
\]

\[
\begin{pmatrix}
\sum_j k_{i,j} & \ast & \cdots & -k_{i,j} \\
\ast & \sum_j k_{i,j} & \ast & \vdots \\
\vdots & \ast & \ddots & \ast \\
-k_{j,i} & \cdots & \ast & \sum_j k_{i,j}
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
\vdots \\
u_n
\end{pmatrix}
= \begin{pmatrix}\tilde{u}_1 \\
\tilde{u}_2 \\
\vdots \\
\tilde{u}_n\end{pmatrix}
\]
Choice of spring constants $k_{i,j}$

- Uniform

  $$k_{i,j} = 1$$

- Cotan

  $$k_{i,j} = \cot \phi_{i,j} + \cot \phi_{j,i}$$
Tutte’s Theorem

• If the weights are nonnegative, and the boundary is fixed to a convex polygon, the parameterization is bijective

• (Tutte’63 proved for uniform weights, Floater’97 extended to arbitrary nonnegative weights)

Comparison of Weights

**uniform weights**

Parameterization with uniform weights [Tutte 1963] on a circular domain.

**cotan weights**

Parameterization with harmonic weights [Eck et al. 1995] on a circular domain.

Discussion

- The results of cotan-weights mapping are better than those of uniform convex mapping (local area and angles preservation).
- But: the mapping is not always legal (the cotan weights can be negative for badly-shaped triangles…)

- In any case: sparse system to solve, so robust and efficient numerical solvers exist
Discussion

- Both mappings have the problem of **fixed boundary** – it constrains the minimization and causes **distortion**.
- More advanced methods do not require boundary conditions.

ABF++ method, Sheffer et al. 2005  
http://www.cs.ubc.ca/~sheffa/ABF++/abf.htm
Parametrization = 2 Scalar Functions
Designing Scalar Functions

Jacobson et al. 2012
Designing Scalar Functions

Orzan et al. 2008
Two Scalar Functions
Designing Gradients

Hatching

Texture Synthesis

Meshing
Intepolating Vectors
Poisson Problem

Condition per triangle

\[ (\nabla s)_f = u_f \]

Error per triangle

\[ P_f = \| (\nabla s)_f - u_f \|_2 \]

Error over the entire mesh

\[ P = \sum_{f \in \mathcal{F}} P_f \]
Barycentric Coordinates

\[ p = w_1 v_1 + w_2 v_2 + w_3 v_3 \]
Barycentric Coordinates

\[ p = w_1 v_1 + w_2 v_2 + w_3 v_3 \]

Partition of unity: \[ w_1 + w_2 + w_3 = 1 \]

\[ p = w_1 v_1 + w_2 v_2 + (1 - w_1 - w_2) v_3 \]
Barycentric Coordinates

\[ p = w_1 v_1 + w_2 v_2 + w_3 v_3 \]

Partition of unity: \( w_1 + w_2 + w_3 = 1 \)

\[ p = w_1 v_1 + w_2 v_2 + (1 - w_1 - w_2) v_3 \]
Barycentric Coordinates

\[ p = w_1 v_1 + w_2 v_2 + w_3 v_3 \]

Partition of unity: \( w_1 + w_2 + w_3 = 1 \)

\[ p = w_1 v_1 + w_2 v_2 + (1 - w_1 - w_2) v_3 \]
Barycentric Coordinates

\[ \mathbf{p} = w_1 \mathbf{v}_1 + w_2 \mathbf{v}_2 + w_3 \mathbf{v}_3 \]

Partition of unity: \( w_1 + w_2 + w_3 = 1 \)

\[ \mathbf{p} = w_1 \mathbf{v}_1 + w_2 \mathbf{v}_2 + (1 - w_1 - w_2) \mathbf{v}_3 \]
Piecewise linear functions on meshes

Hat functions and PL interpolation

\[ f(p) = B_i(p)f_i + B_j(p)f_j + B_k(p)f_k \]
\[ B_i(p) + B_j(p) + B_k(p) = 1 \]
Piecewise linear functions on meshes

Hat functions and PL interpolation

\[ f(p) = B_i(p)f_i + B_j(p)f_j + B_k(p)f_k \]
\[ B_i(p) + B_j(p) + B_k(p) = 1 \]

Gradients

\[ \nabla f(p) = \nabla B_i(p)f_i + \nabla B_j(p)f_j + \nabla B_k(p)f_k \]
\[ \nabla B_i(p) + \nabla B_j(p) + \nabla B_k(p) = 0 \]
Piecewise linear functions on meshes

Hat functions and PL interpolation

\[ f(p) = B_i(p)f_i + B_j(p)f_j + B_k(p)f_k \]
\[ B_i(p) + B_j(p) + B_k(p) = 1 \]

Gradients

\[ \nabla f(p) = \nabla B_i(p)f_i + \nabla B_j(p)f_j + \nabla B_k(p)f_k \]
\[ \nabla B_i(p) + \nabla B_j(p) + \nabla B_k(p) = 0 \]

\[ \nabla f(p) = (f_j - f_i)\nabla B_j(p) + (f_k - f_i)\nabla B_k(p) \]
The hat function

\[ B(p) = \frac{\text{area}}{\text{area}} \]
The hat function

\[ B(p) = \frac{\text{area} \cdot \triangle}{\text{area} \cdot \parallel x \parallel} = \frac{\parallel x \parallel}{\parallel v \parallel} \]
The hat function

\[ B(p) = \frac{\text{area}}{\|v\|} = \frac{\|x\|}{\|v\|} = \frac{(p - o) \cdot \frac{v}{\|v\|}}{\|v\|} \]
The hat function

\[ B(p) = \frac{\text{area}}{\| v \|} = \frac{\| x \|}{\| v \|} = \frac{(p - o) \cdot \frac{v}{\| v \|}}{\| v \|} = \frac{(p - o) \cdot v}{\| v \| \| v \|} \]
Gradient of the hat function

\[ B(p) = \frac{(p - o) \cdot v}{\|v\|} \]
Gradient of the hat function

\[ \nabla B(p) = \frac{v}{\|v\|\|v\|} \]

\[ B(p) = \frac{(p - o) \cdot v}{\|v\|\|v\|} \]
Gradient of the hat function

\[ \nabla B(p) = \frac{v}{\|v\|\|v\|} \]

\[ B(p) = \frac{(p - o) \cdot v}{\|v\|\|v\|} \]
Gradient of the hat function

\[
B(p) = \frac{(p - o) \cdot v}{\|v\| \|v\|}
\]

\[
\nabla B(p) = \frac{v}{\|v\| \|v\|}
\]

\[
\nabla B(p) = \frac{w^\perp}{\|w^\perp\| \|v\|}
\]

\[
v \frac{\|v\|}{\|v\|} = \frac{w^\perp}{\|w^\perp\|}
\]
Gradient of the hat function

\[ B(p) = \frac{(p - o) \cdot v}{\|v\| \|v\|} \]

\[ \nabla B(p) = \frac{v}{\|v\| \|v\|} \]

\[ \nabla B(p) = \frac{w^\perp}{\|w^\perp\| \|v\|} \]

\[ \frac{v}{\|v\|} = \frac{w^\perp}{\|w^\perp\|} \]

\[ \|w^\perp\| = \|w\| \]
Gradient of the hat function

\[ B(p) = \frac{(p - o) \cdot v}{\|v\| \|v\|} \]

\[ \nabla B(p) = \frac{v}{\|v\| \|v\|} \]

\[ \nabla B(p) = \frac{w^\perp}{\|w^\perp\| \|v\|} \]

\[ \nabla B(p) = \frac{w^\perp}{\|w\| \|v\|} \]

\[ \frac{v}{\|v\|} = \frac{w^\perp}{\|w^\perp\|} \]

\[ \|w^\perp\| = \|w\| \]
Gradient of the hat function

\[ B(p) = \frac{(p - o) \cdot v}{\|v\| \|v\|} \]

\[ \nabla B(p) = \frac{v}{\|v\| \|v\|} \]

\[ \nabla B(p) = \frac{w^\perp}{\|w^\perp\| \|v\|} \]

\[ \nabla B(p) = \frac{w^\perp}{\|w\| \|v\|} \]

\[ \frac{v}{\|v\|} = \frac{w^\perp}{\|w^\perp\|} \]

\[ \|w^\perp\| = \|w\| \]

\[ A = \frac{\|v\| \|w\|}{2} \]
Gradient of the hat function

\[ B(p) = \frac{(p - o) \cdot v}{\|v\| \|v\|} \]

\[ \nabla B(p) = \frac{v}{\|v\| \|v\|} \]

\[ \nabla B(p) = \frac{w^\perp}{\|w^\perp\| \|v\|} \]

\[ \nabla B(p) = \frac{w^\perp}{\|w\| \|v\|} \]

\[ \nabla B(p) = \frac{w^\perp}{2A} \]

\[ \frac{v}{\|v\|} = \frac{w^\perp}{\|w^\perp\|} \]

\[ \|w^\perp\| = \|w\| \]

\[ A = \frac{\|v\| \|w\|}{2} \]
Piecewise linear functions on meshes

Hat functions and PL interpolation

\[ f(p) = B_i(p)f_i + B_j(p)f_j + B_k(p)f_k \]

Gradients

\[ \nabla f(p) = (f_j - f_i)\nabla B_j(p) + (f_k - f_i)\nabla B_k(p) \]
Piecewise linear functions on meshes

Hat functions and PL interpolation

\[ f(p) = B_i(p)f_i + B_j(p)f_j + B_k(p)f_k \]

Gradients

\[ \nabla f(p) = (f_j - f_i) \nabla B_j(p) + (f_k - f_i) \nabla B_k(p) \]

\[ \nabla f(p) = (f_j - f_i) \frac{(v_i - v_k)^\perp}{2A} + (f_k - f_i) \frac{(v_j - v_i)^\perp}{2A} \]
Summary

• Parametrization have many uses in computer graphics: they allow to use 2D algorithms and data on surfaces

• Designing a parametrization is equivalent to designing two scalar functions

• Scalar functions can be designed interpolating values directly, or interpolating vectors and solving a Poisson problem
Thank you