goal

- the goal will be to start with a mesh
- we will pull vertices around
- we want the mesh to naturally bend
- we want the details to rotate along with the “base”

aside: related areas

- mesh editing!!
- fair surface creation
- ffd
- physical simulation
- skinning

general energy issues

- in the real world, surfaces have thickness, so the energy comes from a “volumetric” change in the substance.
- if we just use 2D surfaces, we may have to additionally put in a “bending” term which looks at curvature.

aside: shell energy

- lets call the orig mesh $S$, and the new mesh $\hat{S}$
- we already saw greens energy of a mapping between surfaces
  \[
  \int_S dA_g |g_{ab} - \hat{g}_{ab}|^2_g
  \]
- could also pull back the second fundamental form and compare it
  \[
  \int_S dA_g |\hat{q}_{ab} - q_{ab}|^2_g
  \]
- or add the two
- these can be approximated to get a poisson or bi-poisson linear equation in the displacement.

poisson equation

- editing will give us some point constraints
- we want detail to match
  \[
  \sum_i \int_S dA_g |\nabla_a x'^i - \nabla_a x^i|^2_g
  \]
- edited mesh is defined by three new functions over $S$, $x'^i$.
- since we don’t want to freeze the $x,y,z$ components, so we will want a spatially varying rotation $R$.
  - $R$ may be variable, or somehow fixed.
- the energy is:
  \[
  \sum_i \int_S dA_g |\nabla_a x'^i - \nabla_a x^i|^2_g
  \]
for a mesh

- functions are PL, and $R$ is constant per triangle, giving us

$$\sum_t \sum_i \int_t dA_x \nabla_a x_i - \nabla_a \sum_j R_j \int_t x_j^2$$

- from PP, this is equal to

$$\sum_t \sum_{h_e \in t} \cot(\alpha_{ivw})(x_v' - x_w') - R(t)(x_v - x_w)$$

$$= \sum_{h_e \in v} \cot(\alpha_{ivw})(x_v' - x_w') - R(t_{vw})(x_v - x_w)$$

- this is very similar to what we got for ARAP parameterization

- but $R$ is 3-by-3, and $x_v$ are the 3d coordinates (instead of coordinates in a triangle’s own ortho parameterization).

**details**

- take gradient and set it to zero,

$$\forall v \sum_{w \in N(v)} \cot(\alpha_{iwv})(x_v' - x_w') - \sum_{w \in N(v)} \cot(\alpha_{wiv})(x_w' - x_v')$$

$$= \sum_{w \in N(v)} \cot(\alpha_{iwv})R(t_{vw})(x_v - x_w)$$

$$- \sum_{w \in N(v)} \cot(\alpha_{wiv})R(t_{wv})(x_v - x_w)$$

- add in the $M_{v}^{-1}$ to both sides:

$$\forall v \frac{1}{A_v} \left[ \sum_{w \in N(v)} \cot(\alpha_{iwv})(x_v' - x_w') - \sum_{w \in N(v)} \cot(\alpha_{wiv})(x_w' - x_v') \right]$$

$$= \frac{1}{A_v} \left[ \sum_{w \in N(v)} \cot(\alpha_{iwv})R(t_{vw})(x_v - x_w) - \sum_{w \in N(v)} \cot(\alpha_{wiv})R(t_{wv})(x_v - x_w) \right]$$

and we get

- the lhs two terms can be combined, but not the rhs, because of the rotations.

$$\forall v \frac{1}{A_v} \sum_{w \in N(v)} \cot(\alpha_{iwv})R(t_{vw})(x_v - x_w)$$

$$+ \cot(\alpha_{wiv})R(t_{wv})(x_v - x_w)$$

2
• SA a one-ring based poisson equation

\[ \sum_v \sum_{w \in N(v)} (\cot(\alpha_{vw}) + \cot(\alpha_{uw})) \]

\[ ||(x'_v - x'_w) - R(v_v)(x_v - x_w)||^2 \]

gives equations

\[ \frac{1}{A_v} \sum_{w \in N(v)} [\cot(\alpha_{vw}) + \cot(\alpha_{uw})](x'_v - x'_w) \]

\[ = \frac{1}{A_v} \left[ \sum_{w \in N(v)} [\cot(\alpha_{vw}) + \cot(\alpha_{uw})] \right] 
\left( \frac{1}{2} R(v_v) + \frac{1}{2} R(v_w) \right) (x_v - x_w) \]

bi-poisson

• we may instead want the laplacian of the new surface to match some rhs

\[ \sum_i \int_s dA_i |\Delta x'^i - \Delta x^i|^2 \]

• add in rotations

\[ \sum_i \int_s dA_i |\Delta x'^i - \sum_j R^i_j \Delta x^j|^2 \]

• sort of like asking for the mean curvature vectors to hit some prescription

• not exactly since the Laplace Beltrami is defined over \( S \) and not \( S' \)

mesh

• we could really compute a gradient of a PL function over a mesh, but cannot compute a real laplacian.

• lets just use our discrete laplacian.

\[ L_v(f) := \frac{1}{A_v} \sum_{w \in N(v)} (\cot(\alpha_{vw}) + \cot(\alpha_{uw}))(f_v - f_w) \]

• and we have energy

\[ \sum_i \sum_v A_v \]

\[ |\frac{1}{A_v} \sum_{w \in N(v)} (\cot(\alpha_{vw}) + \cot(\alpha_{uw}))(x'_v - x'_w) | \]

\[ - \sum_j (R_v) \frac{1}{A_v} \sum_{w \in N(v)} (\cot(\alpha_{vw}) + \cot(\alpha_{uw}))(x'_v - x'_w) \]

\[ = \sum_v \frac{1}{A_v} \]

\[ || \sum_{w \in N(v)} (\cot(\alpha_{vw}) + \cot(\alpha_{uw}))(x'_v - x'_w) || \]
\[- \sum_{w \in N(v)} (\cot(\alpha_{vw}) + \cot(\alpha_{wv})) R_v (x_v - x_w) \|^2 \]

\[= \sum_v \frac{1}{A_v} \]

\[\| \sum_{w \in N(v)} (\cot(\alpha_{vw}) + \cot(\alpha_{wv}))(x_v' - x_w') - R_v (x_v - x_w) \|^2 \]

matrix form

- in matrix form, let \( L := M^{-1} \ddot{d} \ddot{C} \ddot{d} \)
- and we get the energy

\[(Lx' - RLx)^T M_1 (Lx' - RLx)\]

- for fixed \( R_s \), one can solve for the \( x' \) as

\[L^2 x' = LRLx\]

propagate

- the hard part is figuring out the rotations.
- suppose some of the rotations are given:
- propagate that along the mesh (based on distance). use that to set the \( R \).

harmonic 1

- specify a single global axis. specify \( \theta \) at more than one place
- solve for harmonic \( \theta \) function
- this is just another poisson problem.
- use that as \( R \).

harmonic 2

- specify rotation matrices at more than one place
- solve for harmonic rotation field
- this is a non-linear problem, but can be approximated
- use that as \( R \).

estimate

- all of the above require the user to specify rotations explicitly
- cannot just pull on vertices
- look at the solution, using just positional constraints, and no rotations
- use large nbhodds to estimate regional affinities,
- set the rotations, solve again

linearize

- leave the unknown rotation in the equation:
  - (or turn it into a similarity transform)
- in 2d, similarity is an unconstrained skew symmetric
in 3d sim/rot are constrained, but can be linearized with skew symmetric matrix.

solve simultaneously

if used similarity, and want rotation, then solve a second poisson system with scales for rhs (this gives us igarashi)

**non-linear**

- the linear methods are pretty messed up.
- write it as \( \min_x \min_R E(x', R) \)
- for fixed \( R \), solve for \( m \)
- is a poisson/bipoisson problem

**local phase, laplacian**

- for fixed \( x' \), solve for best fitting \( R \)
- In surface-2d mapping (parameterization) or 2d-2d mapping this requires a SVD on the triangle’s jacobian.
- in SA, they map a ring in 3d to a ring. this also requires SVD.

**local phase, bi laplacian**

- for 3d bi-laplacian, this can be done directly.
- let \( H \) be the 3-diagonal matrix of the scales of \( Lx \).
- let \( H' \) be the 3-diagonal matrix of the scales of \( Lx' \).
- then under the best \( R \), \( RLx = HH'^{-1}Lx' \)
- so the update becomes

\[
L^2x'_{i+1} = LH'H'^{-1}Lx'_{i}
\]

**analysis**

- triangle based poisson turns out to be exactly 2D ARAP energy \( \int (\lambda_1 - 1)^2 + (\lambda_2 - 1)^2 \).
- bi-laplace approximately minimizes “integral of square of difference of mean curvature scalars”.
  - not exactly, since the mean curvature of \( S' \) would require using updated cotangents.

**nonlinear behavior**

- if you are doing 2D-2D, non linear poission is great.
- if you are deforming a triangle mesh in 3d, the mesh acts like slightly flexible triangular pieces.
  - wants to bend at edges
- SA stiffens up but i wish i understood it better
- i would suggest doing bi-laplacian.