10 - ARAP and Linear Blend Skinning
As Rigid As Possible
Demo

• Libigl demo 405
As-Rigid-As-Possible Deformation

- Preserve shape of cells covering the surface
- Ask each cell $i$ to transform **rigidly** by best-fitting rotation $R_i$

\[
\min \sum_{T \in \text{Cell}_i} \sum_{(j,k) \in T} \| (x'_j - x'_k) - R_i(x_j - x_k) \|^2
\]
As-Rigid-As-Possible Deformation

• Optimal $R_i$ is uniquely defined by $x_i, x'_i$

$$
\min_{T \in \text{Cell}_i} \sum_{(j,k) \in T} \| (x'_j - x'_k) - R_i(x_j - x_k) \|^2
$$

• so-called shape-matching problem, solved by a 3x3 SVD

$R_i$ is a nonlinear function of $x$
Optimal Rotation

\[
\min_{R \in SO(3)} \sum_{T \in \text{Cell}_i} \sum_{(j,k) \in T} \| (x'_j - x'_k) - R(x_j - x_k) \|^2
\]

Rotation group
Shape Matching Problem

R, t?
Shape Matching Problem
Shape Matching Problem
Shape Matching Problem
Shape Matching Problem

- Align two point sets

\[ \mathcal{P} = \{p_1, \ldots, p_n\} \text{ and } \mathcal{Q} = \{q_1, \ldots, q_n\} \]

- Find a translation vector \( t \) and rotation matrix \( R \) so that

\[
\sum_{i=1}^{n} \| (Rp_i + t) - q_i \|^2 \text{ is minimized}
\]
Shape Matching – Solution

• Solve for translation first (w.r.t. $\mathbf{R}$, $\mathbf{p}$, and $\mathbf{q}$)

\[
\frac{\partial}{\partial t} \sum_{i=1}^{n} \| (\mathbf{R} \mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i \|^2 = \sum_{i=1}^{n} \frac{\partial}{\partial t} 2 \left( (\mathbf{R} \mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i \right) = 0
\]

\[
\mathbf{R} \sum_{i=1}^{n} \mathbf{p}_i + \sum_{i=1}^{n} \mathbf{t} - \sum_{i=1}^{n} \mathbf{q}_i = 0
\]

\[
\mathbf{t} = \left( \frac{1}{n} \sum_{i=1}^{n} \mathbf{q}_i \right) - \mathbf{R} \left( \frac{1}{n} \sum_{i=1}^{n} \mathbf{p}_i \right)
\]

Point sets $\{\mathbf{q}_i\}$ and $\{\mathbf{R} \mathbf{p}_i\}$ have the same center of mass.
Finding the Rotation $\mathbf{R}$

• To find the optimal $\mathbf{R}$, we bring the centroids of both point sets to the origin

$$\mathbf{v}_i = \mathbf{p}_i - \bar{\mathbf{p}}, \quad \mathbf{v}_i' = \mathbf{q}_i - \bar{\mathbf{q}}$$

• We want to find $\mathbf{R}$ that minimizes

$$\sum_{i=1}^{n} \| \mathbf{Rv}_i - \mathbf{v}_i' \|^2$$
Finding the Rotation $R$

$$\sum_{i=1}^{n} \| Rv_i - v'_i \|^2 = \sum_{i=1}^{n} (Rv_i - v'_i)^T (Rv_i - v'_i) =$$

$$= \sum_{i=1}^{n} \left( v_i^T R Rv_i - v'_i^T Rv_i - v_i^T R^T v'_i + v'_i^T v'_i \right)$$

These terms do not depend on $R$, so we can ignore them in the minimization.
Finding the Rotation $\mathbf{R}$

\[
\arg\min_{\mathbf{R} \in SO(3)} \sum_{i=1}^{n} \left( -v_i' \mathbf{R} v_i - v_i' \mathbf{R}^T v_i' \right) = \arg\max_{\mathbf{R} \in SO(3)} \sum_{i=1}^{n} \left( v_i' \mathbf{R} v_i + v_i' \mathbf{R}^T v_i' \right) =
\]

\[
= \arg\max_{\mathbf{R} \in SO(3)} \sum_{i=1}^{n} v_i' \mathbf{R} v_i
\]

\[
v_i \mathbf{R}^T v_i' = \left( v_i \mathbf{R} v_i' \right)^T = v_i' \mathbf{R} v_i
\]
Finding the Rotation $R$

$$\sum_{i=1}^{n} v'_i^T R v_i = tr \left( V'^T R V \right)$$
Finding the Rotation $\mathbf{R}$

\[
\sum_{i=1}^{n} v'_i^T \mathbf{R} v_i = tr \left( \mathbf{V}'^T \mathbf{R} \mathbf{V} \right)
\]
Finding the Rotation $\mathbf{R}$

- Find $\mathbf{R}$ that maximizes

$$
\text{tr} \left( \mathbf{V}'^T \mathbf{R} \mathbf{V} \right) \quad \text{vs} \quad \text{tr} \left( \mathbf{R} \mathbf{V} \mathbf{V}'^T \right)
$$

- SVD: $\mathbf{V} \mathbf{V}'^T = \mathbf{U} \Sigma \tilde{\mathbf{U}}^T$

$$
\text{tr} \left( \mathbf{R} \mathbf{V} \mathbf{V}'^T \right) = \text{tr} \left( \mathbf{R} \mathbf{U} \Sigma \tilde{\mathbf{U}}^T \right) = \text{tr} \left( \Sigma \tilde{\mathbf{U}}^T \mathbf{R} \mathbf{U} \right)
$$

Take a look at the Matrix Cookbook!
Finding the Rotation $\mathbf{R}$

- We want to maximize

$$tr \ (\sum \mathbf{M})$$

$\mathbf{M}$: orthonormal matrix

all entries $\leq 1$

$$tr \ (\sum \mathbf{M}) = \sum_{i=1}^{3} \sigma_i m_{ii} \leq \sum_{i=1}^{3} \sigma_i$$
Finding the Rotation $R$

$$tr \left( \Sigma M \right) = \sum_{i=1}^{3} \sigma_i m_{ii} \leq \sum_{i=1}^{3} \sigma_i$$

• Our best shot is $m_{ii} = 1$, i.e. to make $M = I$

$$M = \tilde{U}^T RU \overset{!}{=} I$$

$$RU = \tilde{U}$$

$$R = \tilde{U} U^T$$
Summary of Rigid Alignment

- Translate the input points to the centroids
  \[ \mathbf{v}_i = \mathbf{p}_i - \bar{\mathbf{p}}, \quad \mathbf{v}'_i = \mathbf{q}_i - \bar{\mathbf{q}} \]
- Compute the “covariance matrix” \( \mathbf{V V}'^T \)
- Compute its SVD: \( \mathbf{V V}'^T = \mathbf{U} \Sigma \tilde{\mathbf{U}}^T \)
- The optimal orthonormal \( \mathbf{R} \) is \( \mathbf{R} = \tilde{\mathbf{U}} \mathbf{U}^T \)
Sign Correction

- It is possible that $\det(\tilde{U}U^T) = -1$ sometimes, indicating that reflection is the best orthonormal transform.
Sign Correction

- It is possible that $\det(\tilde{U}U^T) = -1$: sometimes reflection is the best orthonormal transform.
Sign Correction

• To restrict ourselves to rotations only: take the last column of $U$ (corresponding to the smallest singular value) and invert its sign.

As-Rigid-As-Possible Deformation

- Optimal $R_i$ is uniquely defined by $x_i, x'_i$

  $\min_{T \in \text{Cell}_i} \sum_{(j,k) \in T} \| (x'_j - x'_k) - R_i(x_j - x_k) \|^2$

- so-called shape-matching problem, solved by a 3x3 SVD

$R_i$ is a nonlinear function of $x$
As-Rigid-As-Possible Deformation

• Total ARAP energy: sum up for all the cells $i$

$$\sum_{i} \sum_{T \in \text{Cell}_i} \sum_{(j,k) \in T} \| (x'_j - x'_k) - R_i(x_j - x_k) \|^2$$

• Treat $x$ and $R$ as separate sets of variables
• Simple **local-global** iterative optimization process
  • Decreases the energy at each step
As-Rigid-As-Possible Deformation

• Total ARAP energy: sum up for all the cells $i$

$$\sum_{i} \sum_{T \in \text{Cell}_i} \sum_{(j,k) \in T} \| (x'_j - x'_k) - R_i (x_j - x_k) \|^2$$

• Local step: keep $x'$ fixed, find optimal $R_i$ per cell $i$

• Global step: keep $R_i$ fixed, solve for $x'$

$$Lx' = b$$

quadratic minimization problem
As-Rigid-As-Possible Deformation

- Total ARAP energy: sum up for all the cells $i$
  \[ \sum_i \sum_{T \in \text{Cell}_i} \sum_{(j,k) \in T} \left\| (x_j' - x_k') - R_i (x_j - x_k) \right\|^2 \]

- Local step: keep $x'$ fixed, find optimal $R_i$ per cell $i$

- Global step: keep $R_i$ fixed, solve for $x'$
  \[ Lx' = b \]

- Quadratic minimization problem
  - The matrix $L$ stays fixed, can pre-factorize
Initial Guess

• Can use naïve Laplacian editing
Initial Guess

- Can also use the previous frame
- Replace all handle vertex positions by the currently prescribed ones
- Fast convergence
Large Rotations

- Use previous frame as the initial guess
Examples
Discussion

• Nonlinear deformation that models a kind of elastic behavior
• Very simple to implement, no parameters to tune except number of iterations
• Each step is guaranteed to not increase the energy
  • Compare with Gauss-Newton…
• Each iteration is relatively cheap, no matrix re-factorization necessary
Discussion

• Works fine on small meshes
• On larger meshes: slow convergence
  • Each iteration is more expensive
  • Need more iterations because the conditioning of the system becomes worse as the matrix grows
• Material stiffness depends on the cell size
  • lots of wrinkles for fine meshes when using 1-rings as cells
Acceleration using Subspace Techniques

• Subspace created by influence weight functions for each handle
• Drastically reduces the number of degrees of freedom in the optimization

Acceleration using Subspace Techniques

- Subspace created by influence weight functions for each handle
- Drastically reduces the number of degrees of freedom in the optimization

Acceleration using Subspace Techniques

- Subspace created by influence weight functions for each handle
- Drastically reduces the number of degrees of freedom in the optimization

Demo

• Libigl demo 406
Linear Blend Skinning
LBS generalizes to different handle types

- skeletons
- regions
- points
- cages
Linear Blend Skinning rigging preferred for its real-time performance

place handles in shape
Linear Blend Skinning rigging preferred for its real-time performance

place handles in shape

paint weights
Linear Blend Skinning rigging preferred for its real-time performance

place handles in shape  →  paint weights  →  deform handles

$w_j(x_i^0)$
Linear Blend Skinning rigging preferred for its real-time performance

place handles in shape  →  paint weights  →  deform handles

\[ \mathbf{x}_i = \sum_{j=1}^{m} w_j(\mathbf{x}_i^0) \mathbf{T}_j \begin{pmatrix} \mathbf{x}_i^0 \ 1 \end{pmatrix} \]
Linear Blend Skinning rigging preferred for its real-time performance

place handles in shape  →  paint weights  →  deform handles
Challenges with LBS

- Weight functions $w_j$
  - Need intuitive, general and automatic weights
- Degrees of freedom $T_j$
  - Let the energy decide!
- Richness of achievable deformations
  - Want to avoid common LBS pitfalls – candy wrapper, collapses

\[ x_i = \sum_{j=1}^{m} w_j(x_i^0) T_j \begin{pmatrix} x_i^0 \\ 1 \end{pmatrix} \]
Bounded Biharmonic Weights

Alec Jacobson, Ilya Baran, Jovan Popović, Olga Sorkine-Hornung
Automatic weights that unify points, skeletons and cages
Weights should be smooth, shape-aware, positive and *intuitive*
Weights must be smooth everywhere, *especially* at handles

Bounded Biharmonic Weights

Extension of Harmonic Coordinates

[Joshi et al. 2005]
Weights must be smooth everywhere, especially at handles.

Bounded Biharmonic Weights

Extension of Harmonic Coordinates

[Joshi et al. 2005]
Shape-awareness ensures respect of domain’s features

Bounded Biharmonic Weights

Non-shape-aware methods
e.g. [Schaefer et al. 2006]
Non-negative weights are necessary for intuitive response

Bounded Biharmonic Weights

Unconstrained biharmonic
[Botsch and Kobbelt 2004]
Weights must maintain other simple, but important properties

\[ \sum_{j \in H} w_j(x^0) = 1 \]

Partition of unity

\[ w_j \bigg|_{H_k} = \delta_{jk} \]

Interpolation of handles

\[ w_j \] is linear along cage faces
How about \( w_j(x^0) = d(x^0, H_j)^{-1} \)?
Inverse distance methods inherently suffer from *fall-off effect*
Inverse distance methods inherently suffer from *fall-off effect*
Inverse distance methods inherently suffer from fall-off effect.

Approaching 0.5
Inverse distance methods inherently suffer from *fall-off effect*
**Bounded biharmonic weights** enforce properties as constraints to minimization

\[
\arg \min_{w_j} \frac{1}{2} \int_{\Omega} |\Delta w_j|^2 dV
\]

\[w_j \bigg|_{H_k} = \delta_{jk}\]

\[w_j\] is linear along cage faces
**Bounded biharmonic weights** enforce properties as constraints to minimization.

\[
\arg \min_{w_j} \frac{1}{2} \int_{\Omega} |\Delta w_j|^2 dV
\]

\[
w_j \big|_{H_k} = \delta_{jk}
\]

\(w_j\) is linear along cage faces.

Constant inequality constraints:

\[
0 \leq w_j(x^0) \leq 1
\]

Partition of unity:

\[
\sum_{j \in H} w_j(x^0) = 1
\]
**Bounded biharmonic weights** enforce properties as constraints to minimization

\[
\arg \min_{w_j} \frac{1}{2} \int_{\Omega} |\Delta w_j|^2 dV
\]

\[
w_j \bigg|_{H_k} = \delta_{jk}
\]

\(w_j\) is linear along cage faces

**Constant inequality constraints**

\[0 \leq w_j(x^0) \leq 1\]

Solve independently and normalize

\[
w_j(x^0) = \frac{w_j(x^0)}{\sum_{i \in H} w_i(x^0)}
\]
Weights optimized as precomputation at bind-time

\[
\arg\min_{w_j} \frac{1}{2} \int_{\Omega} |\Delta w_j|^2 dV \\
 w_j \bigg|_{H_k} = \delta_{jk} \\
 w_j \text{ is linear along cage faces} \\
 0 \leq w_j(x^0) \leq 1
\]

FEM discretization
2D → Triangle mesh
3D → Tet mesh
Weights optimized as precomputation at bind-time

\[ \arg \min_{w_j} \frac{1}{2} \int_{\Omega} |\Delta w_j|^2 dV \]

\[ w_j \big|_{H_k} = \delta_{jk} \]

\( w_j \) is linear along cage faces

\[ 0 \leq w_j(x^0) \leq 1 \]

**Sparse quadratic programming** with constant inequality constraints

2D → less than second per handle

3D → tens of seconds per handle
Some examples of BBW in action
Some examples of BBW in action
Some examples of BBW in action
3D Characters
Demo

• Libigl demo 403
Thank you