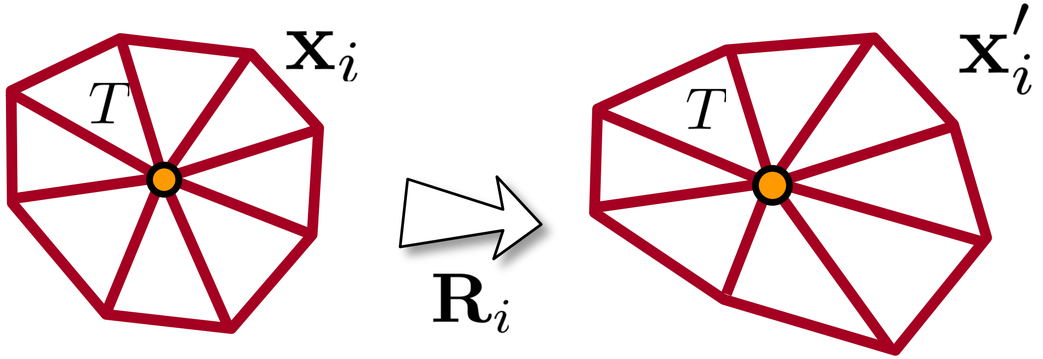
08 - As Rigid As Possible (ARAP) Deformation

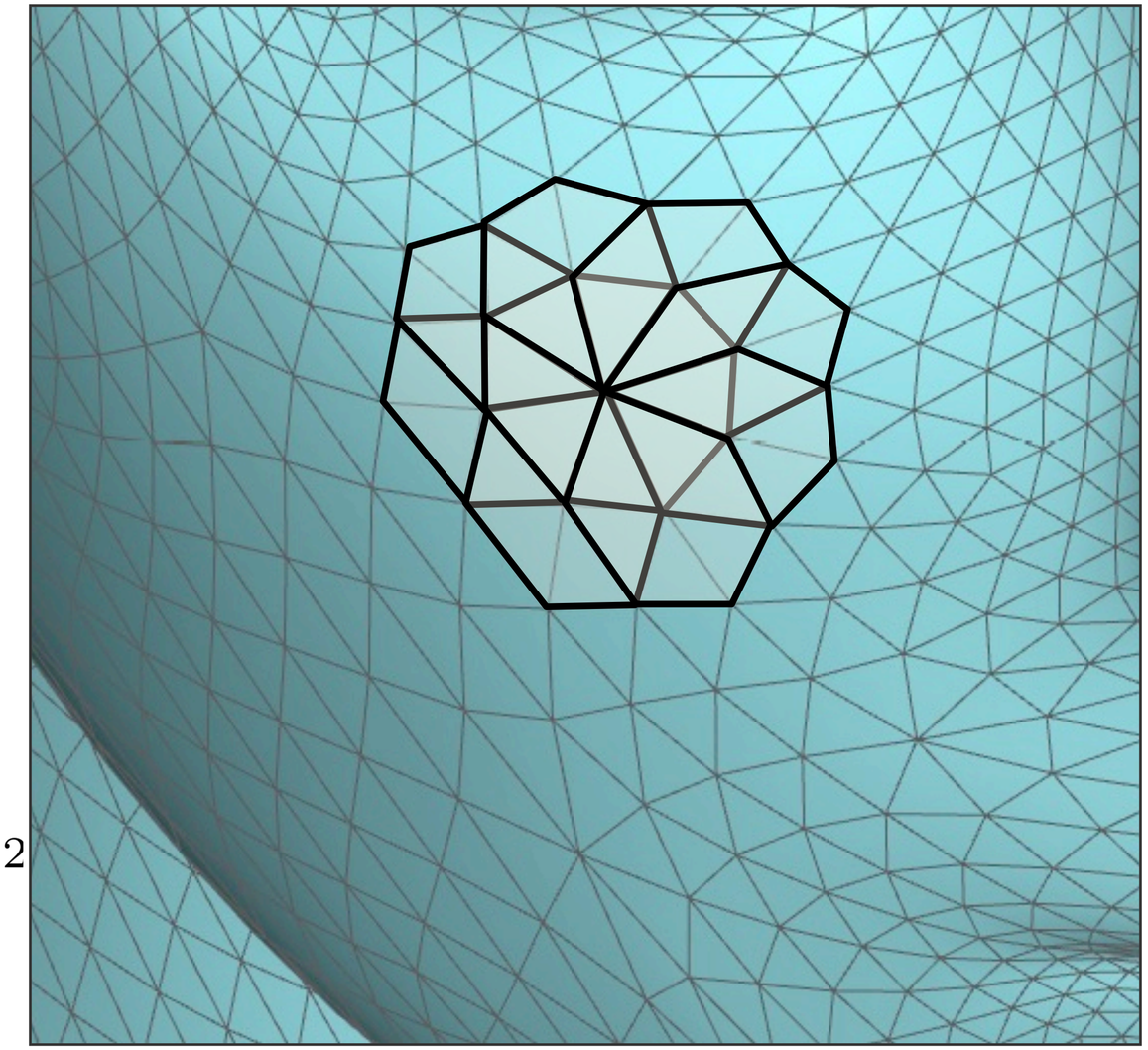
Acknowledgements: Daniele Panozzo

As-Rigid-As-Possible Deformation

- Preserve shape of cells covering the surface
- Ask each cell i to transform **rigidly** by best-fitting rotation $\mathbf{R}_i$

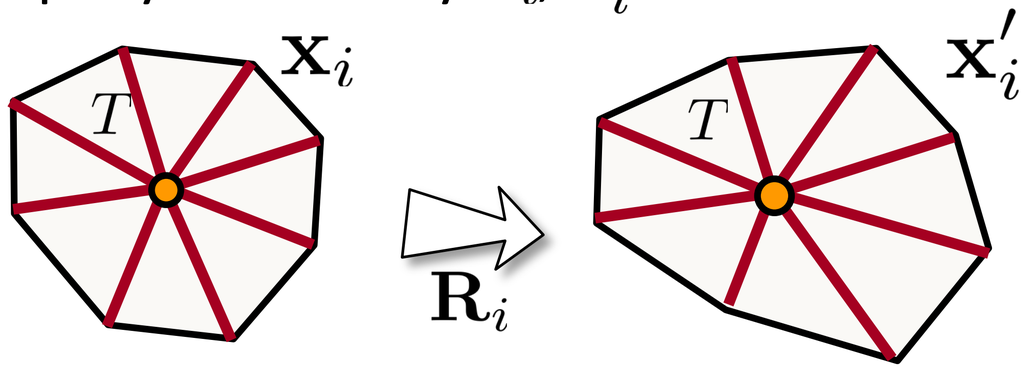


$$\min \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$



As-Rigid-As-Possible Deformation

- Optimal $\mathbf{R}_i$ is uniquely defined by $\mathbf{x}_i, \mathbf{x}'_i$

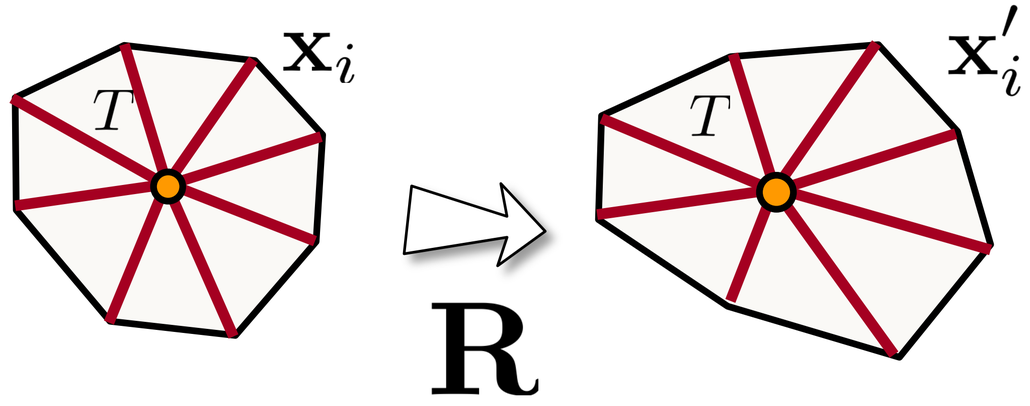


$$\min \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

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

$\mathbf{R}_i$ is a nonlinear function of $\mathbf{x}$

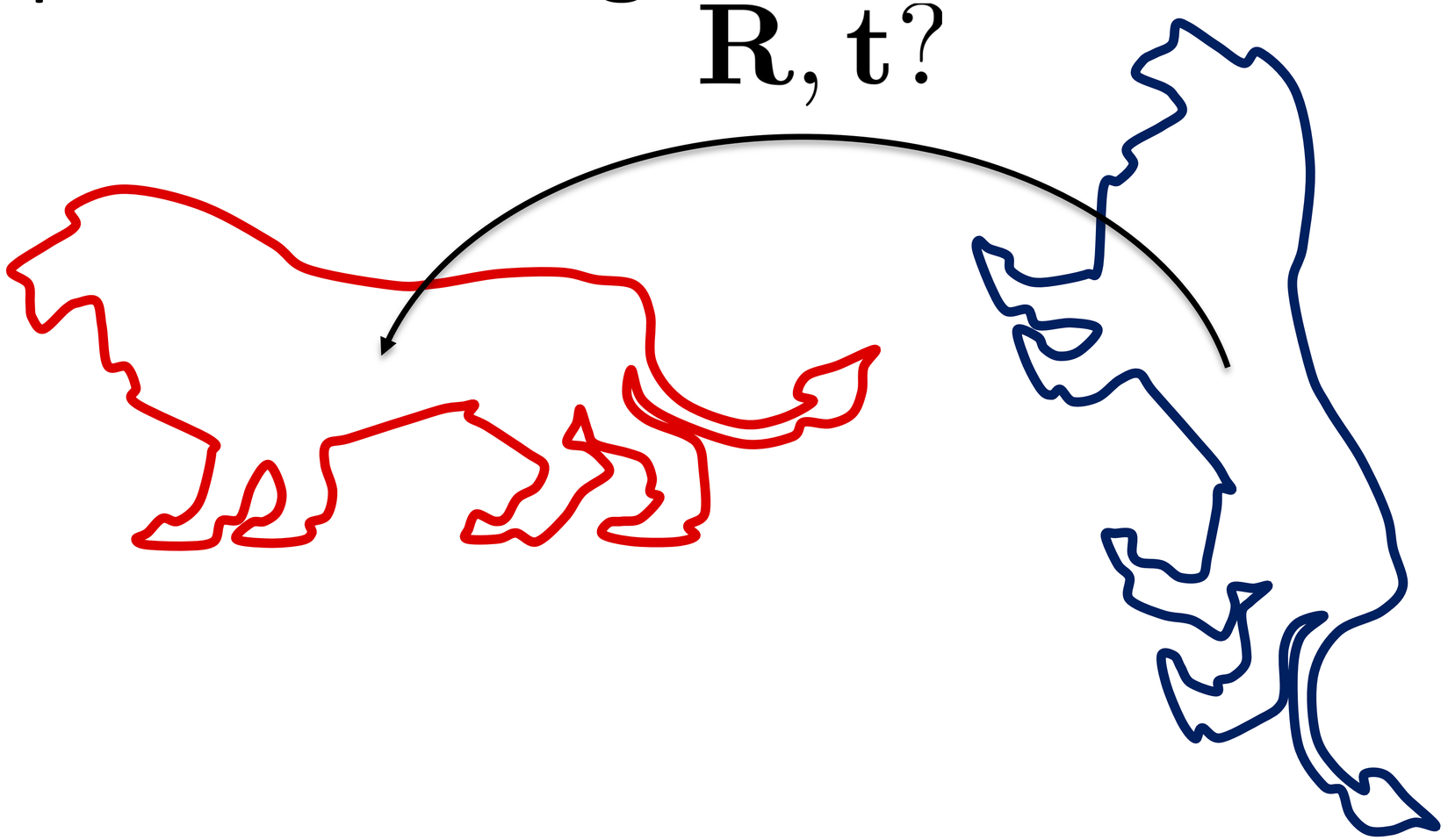
Optimal Rotation



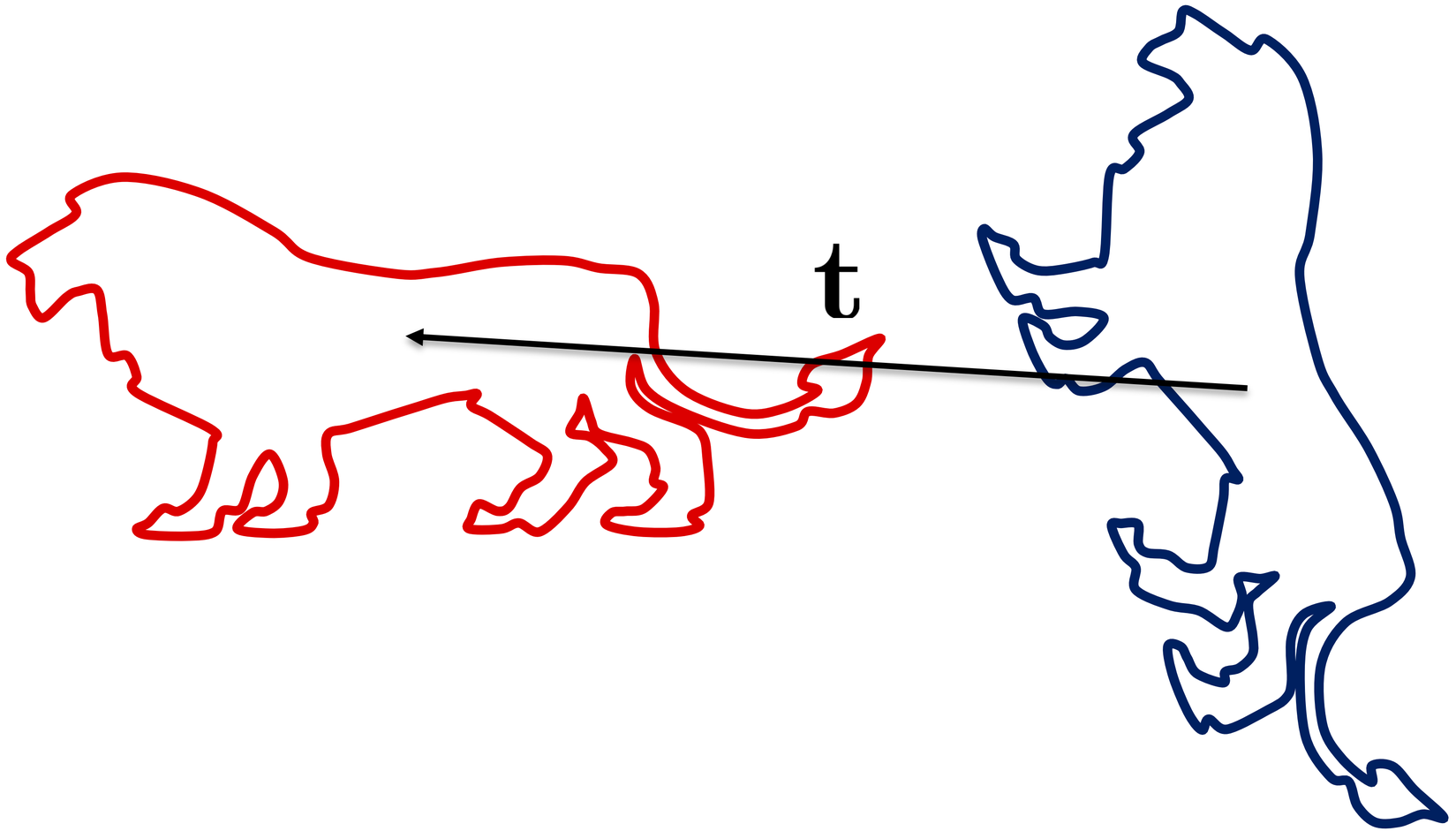
$$\min_{\mathbf{R}\in SO(3)} \sum_{T\in \mathrm{Cell}_i} \sum_{(j,k)\in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

Rotation group

Shape Matching Problem

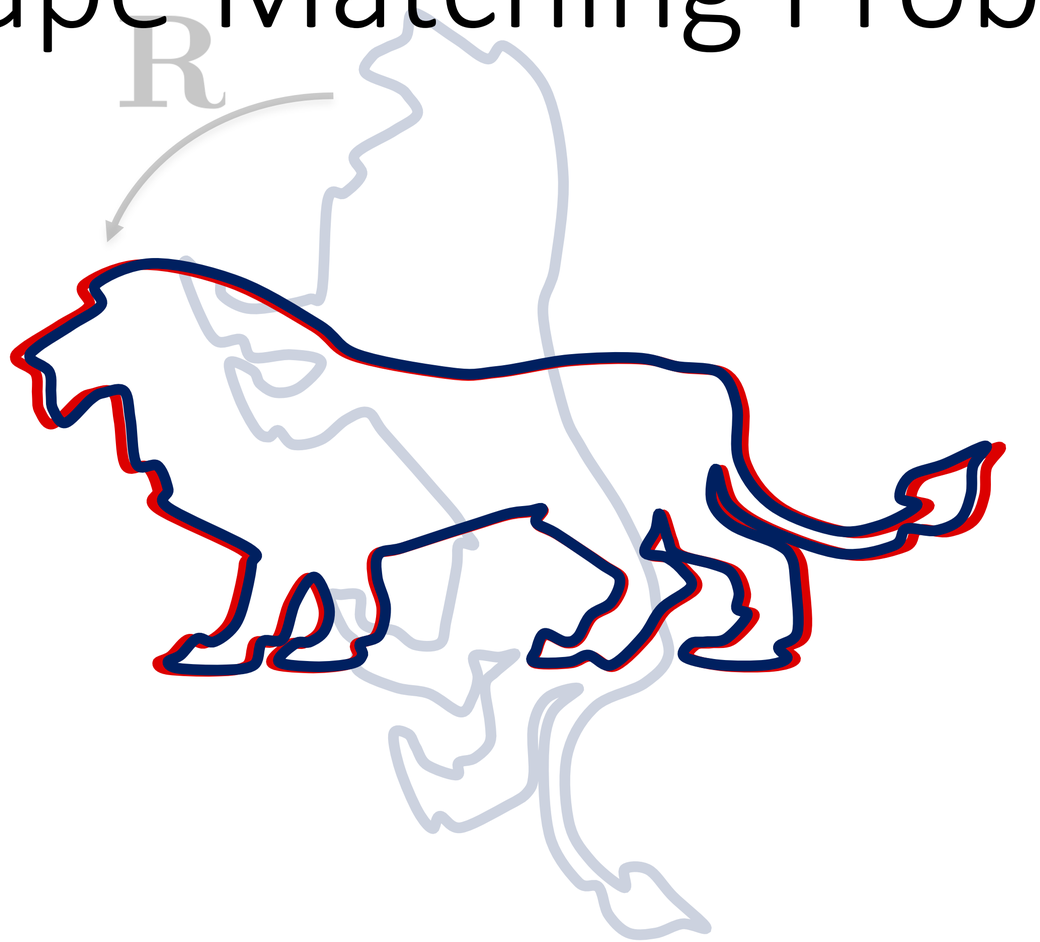


Shape Matching Problem



Shape Matching Problem

Shape Matching Problem



Shape Matching Problem

- Align two point sets

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \text{ and } \mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$$

- Find a translation vector $\mathbf{t}$ and rotation matrix $\mathbf{R}$ so that

$$\sum_{i=1}^{n} \|(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\|^2 \text{ is minimized}$$

Shape Matching – Solution

- Solve for translation first (w.r.t. **R**, **p**, and **q**)

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

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

$$\mathbf{t} = \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{q}_i\right)}_{\bar{\mathbf{q}}} - \mathbf{R}\underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i\right)}_{\bar{\mathbf{p}}}$$

Finding the Rotation $\mathbf{R}$

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

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

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

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

Finding the Rotation $\mathbf{R}$

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

$$= \sum_{i=1}^{n} \left(\mathbf{v}_i^T \underbrace{\mathbf{R}^T \mathbf{R}}_{\mathbf{I}} \mathbf{v}_i - \mathbf{v}'^T_i \mathbf{R}\mathbf{v}_i - \mathbf{v}_i^T \mathbf{R}^T \mathbf{v}'_i + \mathbf{v}'^T_i \mathbf{v}'_i \right)$$

These terms do not depend on **R**,
so we can ignore them in the minimization

Finding the Rotation $\mathbf{R}$

$$\underset{\mathbf{R}\in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} \left(-\mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i - \mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' \right) = \underset{\mathbf{R}\in SO(3)}{\operatorname{argmax}} \sum_{i=1}^{n} \left(\mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i + \mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' \right) =$$

$$= \boxed{\underset{\mathbf{R}\in SO(3)}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i}$$

$$\mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' = \left(\mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' \right)^T = \mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i$$

Finding the Rotation $\mathbf{R}$

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

$$\underbrace{\begin{bmatrix} \mathbf{v}_1'^T \\ \mathbf{v}_2'^T \\ \vdots \\ \mathbf{v}_n'^T \end{bmatrix}}_{\mathbf{V}'^T} \mathbf{R} \underbrace{\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}}_{\mathbf{V}} = \begin{bmatrix} \mathbf{v}_1'^T \\ \mathbf{v}_2'^T \\ \vdots \\ \mathbf{v}_n'^T \end{bmatrix} \begin{bmatrix} \mathbf{R}\mathbf{v}_1 & \mathbf{R}\mathbf{v}_2 & \cdots & \mathbf{R}\mathbf{v}_n \end{bmatrix}$$

Finding the Rotation $\mathbf{R}$

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

$$\begin{bmatrix} \mathbf{v}_1'^T \\ \mathbf{v}_2'^T \\ \vdots \\ \mathbf{v}_n'^T \end{bmatrix} \begin{bmatrix} \mathbf{R}\mathbf{v}_1 & \mathbf{R}\mathbf{v}_2 & \cdots & \mathbf{R}\mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1'^T \mathbf{R}\mathbf{v}_1 & & & \\ & \mathbf{v}_2'^T \mathbf{R}\mathbf{v}_2 & & \\ & & \ddots & \\ & & & \mathbf{v}_n'^T \mathbf{R}\mathbf{v}_n \end{bmatrix}$$

Finding the Rotation $\mathbf{R}$

Take a look at the Matrix Cookbook!

- Find $\mathbf{R}$ that maximizes

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

- SVD:

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

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

orthonormal matrix

Finding the Rotation $\mathbf{R}$

- We want to maximize $tr\left(\Sigma\,\mathbf{M}\right)$

$\mathbf{M}$: orthonormal matrix
all entries ≤ 1

$$\begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} \begin{bmatrix} m_{11} & \cdots & \\ \vdots & m_{22} & \vdots \\ & \cdots & m_{33} \end{bmatrix}$$

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

Finding the Rotation $\mathbf{R}$

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

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

$$\begin{aligned} \mathbf{M} = \tilde{\mathbf{U}}^T \mathbf{R} \mathbf{U} &\stackrel{!}{=} \mathbf{I} \\ \mathbf{R}\mathbf{U} &= \tilde{\mathbf{U}} \\ \boxed{\mathbf{R} &= \tilde{\mathbf{U}}\mathbf{U}^T} \end{aligned}$$

Summary of Rigid Alignment

- Translate the input points to the centroids

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

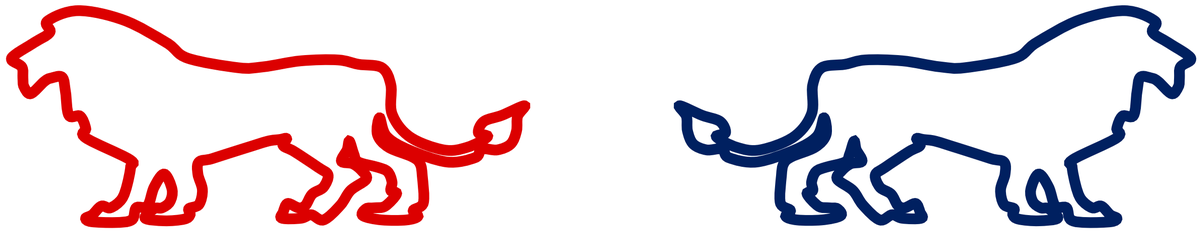
- Compute the "covariance matrix" $\mathbf{V}\mathbf{V}'^T$

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

- The optimal orthonormal **R** is $\mathbf{R} = \tilde{\mathbf{U}}\mathbf{U}^T$

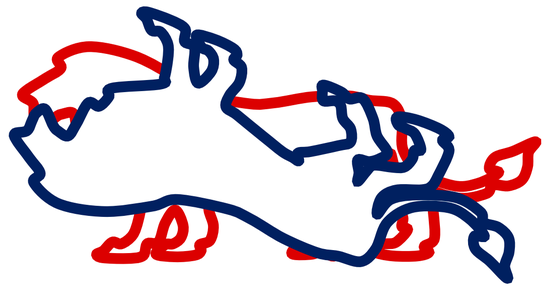
Sign Correction

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



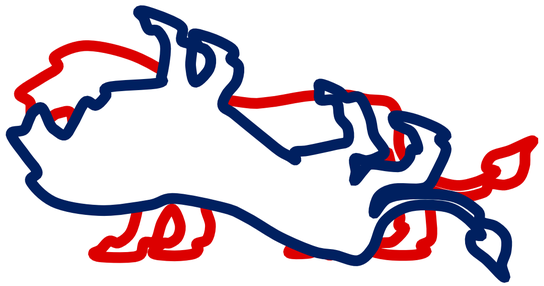
Sign Correction

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



Sign Correction

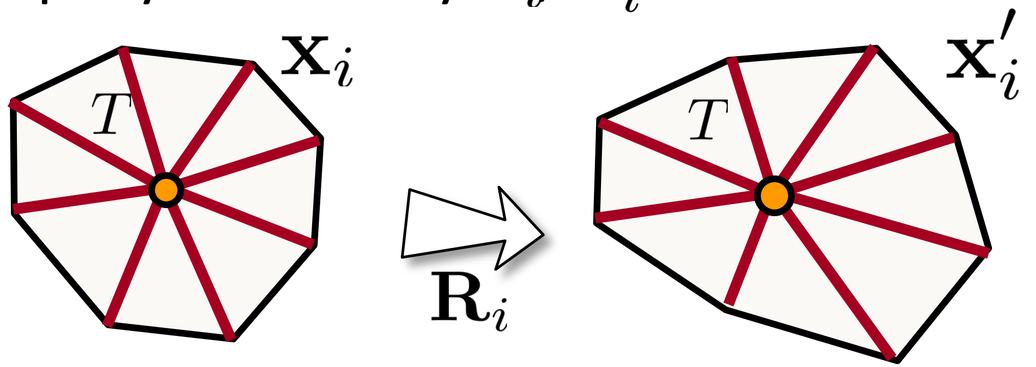
- To restrict ourselves to rotations only:
  take the last column of $\mathbf{U}$ (corresponding to the smallest singular value) and invert its sign.



- Why? See http://igl.ethz.ch/projects/ARAP/svd_rot.pdf

As-Rigid-As-Possible Deformation

- Optimal $\mathbf{R}_i$ is uniquely defined by $\mathbf{x}_i$, $\mathbf{x}'_i$



$$\min \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

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

$\mathbf{R}_i$ is a nonlinear function of $\mathbf{x}$

As-Rigid-As-Possible Deformation

- Total ARAP energy: sum up for all the cells i

$$\sum_{i} \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \left\| (\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i (\mathbf{x}_j - \mathbf{x}_k) \right\|^2$$

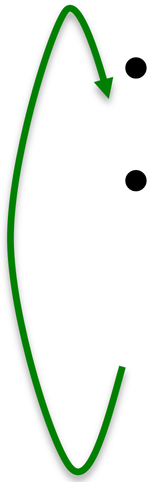
- Treat $\mathbf{x}$ and $\mathbf{R}$ as separate sets of variables
- Simple **local-global** iterative optimization process
  - Decreases the energy at each step

As-Rigid-As-Possible Deformation

- Total ARAP energy: sum up for all the cells i

$$\sum_{i} \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \left\| (\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i (\mathbf{x}_j - \mathbf{x}_k) \right\|^2$$

- Local step: keep $\mathbf{x}$' fixed, find optimal $\mathbf{R}_i$ per cell i
- Global step: keep $\mathbf{R}_i$ fixed, solve for $\mathbf{x}$' – quadratic minimization problem



$$\mathbf{L}\mathbf{x}' = \mathbf{b}$$

As-Rigid-As-Possible Deformation

- Total ARAP energy: sum up for all the cells i

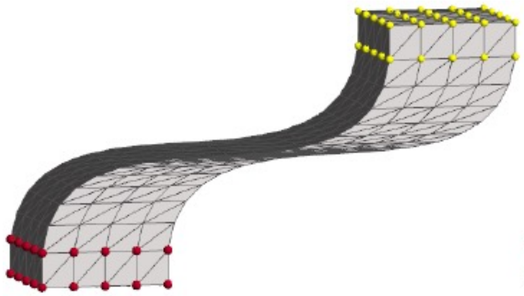
$$\sum_i \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

- Local step: keep $\mathbf{x}$' fixed, find optimal $\mathbf{R}_i$ per cell i
- Global step: keep $\mathbf{R}_i$ fixed, solve for $\mathbf{x}$' – quadratic minimization problem
  - The matrix $\mathbf{L}$ stays fixed, can pre-factorize

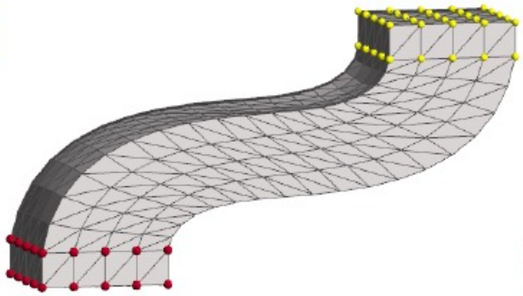
$$\Rightarrow \mathbf{L}\mathbf{x}' = \mathbf{b}$$

Initial Guess

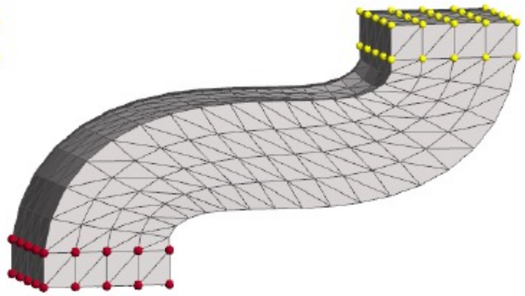
- Can use naïve Laplacian editing



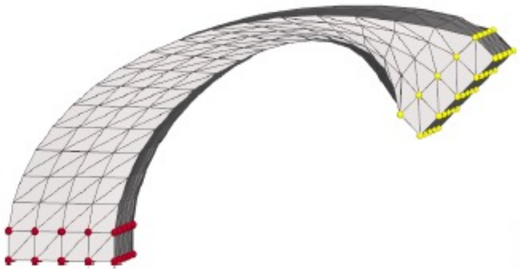
initial guess



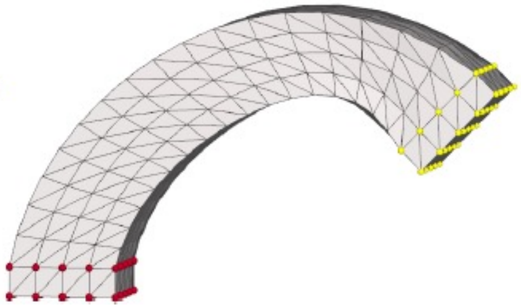
1 iteration



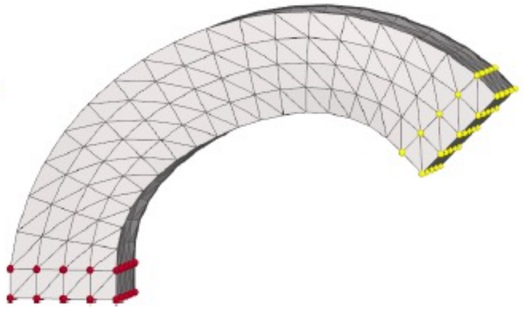
2 iterations



initial guess



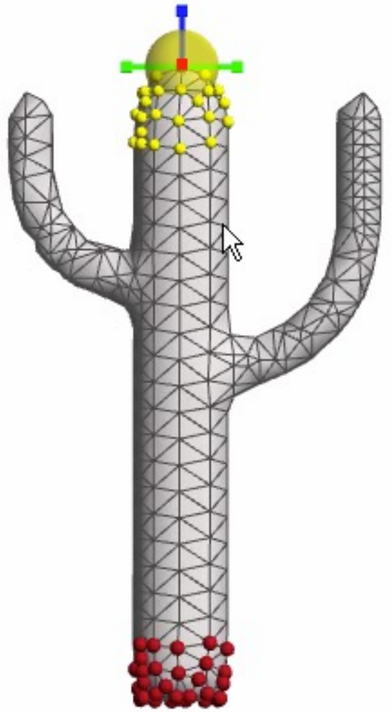
1 iterations



4 iterations

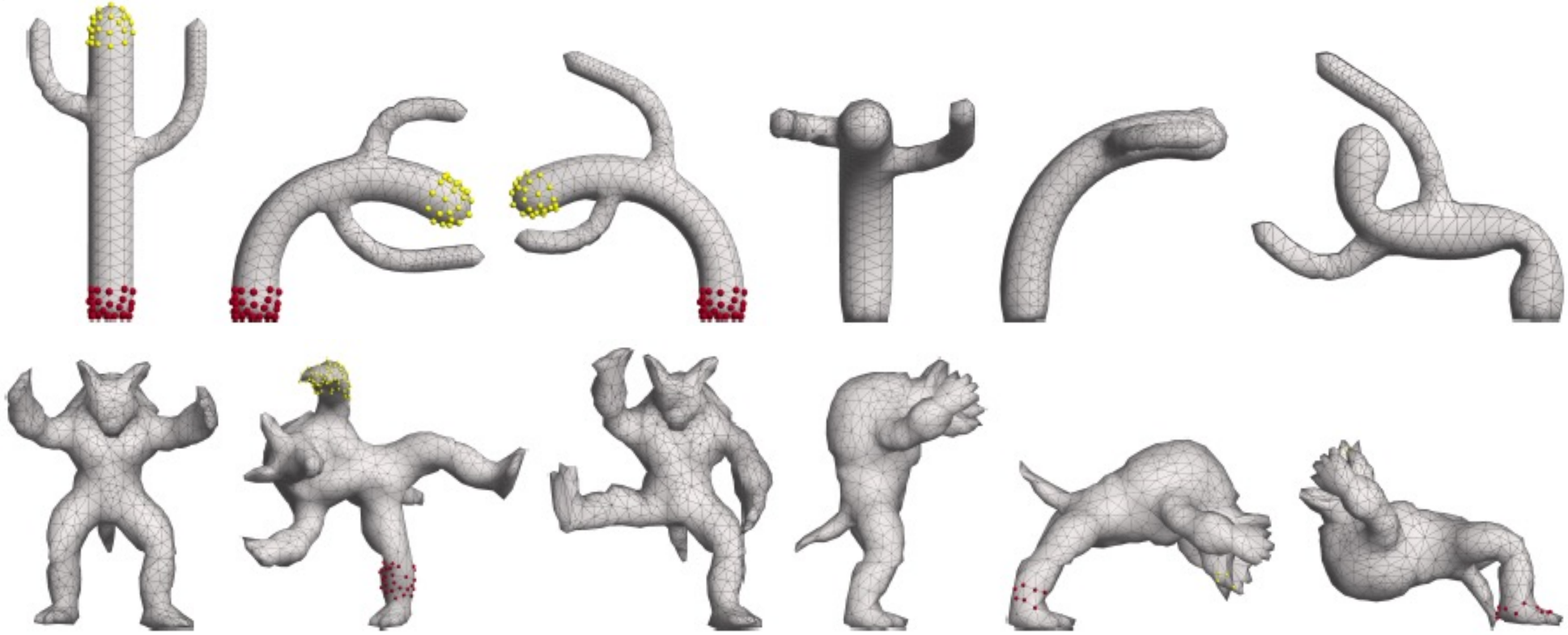
Initial Guess

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

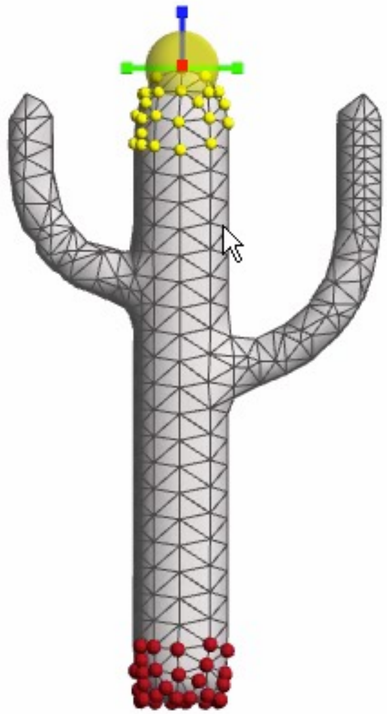


Large Rotations

- 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

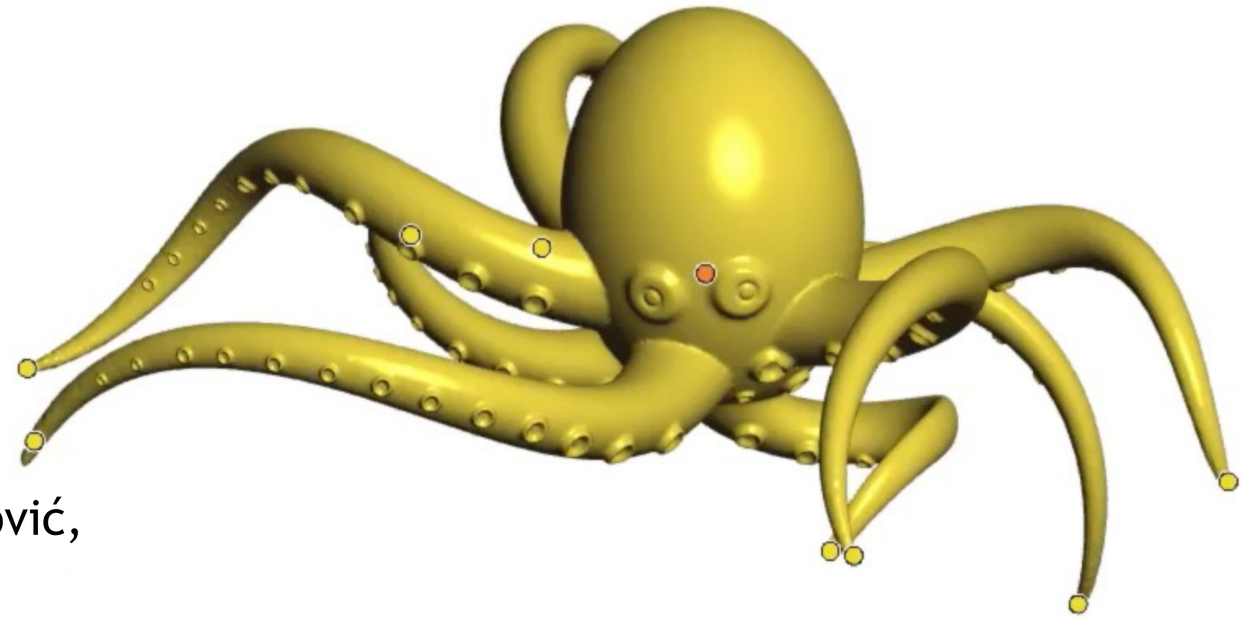
Acceleration using Subspace Techniques

- Subspace created by int
- Drastically reduces the optimization

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. “Fast Automatic Skinning Transformations,” 2012.

Acceleration using Subspace Techniques

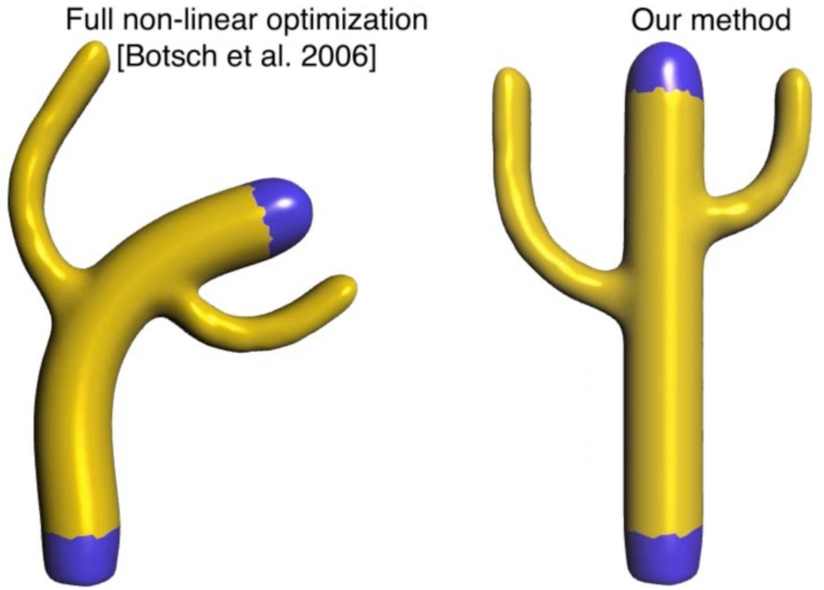
- Subspace created by influence weight functions for each handle
- Drastically reduces the number of degrees of freedom in the optimization



Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. “Fast Automatic Skinning Transformations,” 2012.

Acceleration using Subspace Techniques

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

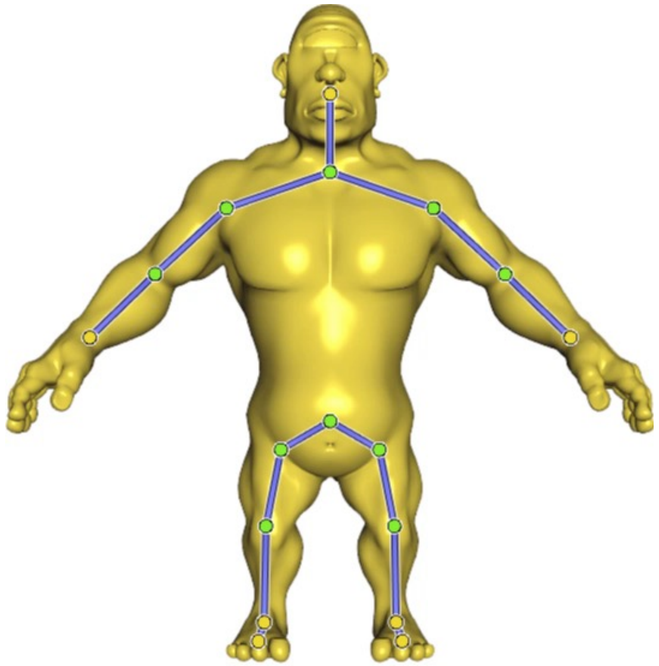


Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. “Fast Automatic Skinning Transformations,” 2012.

08 - Linear Blend Skinning

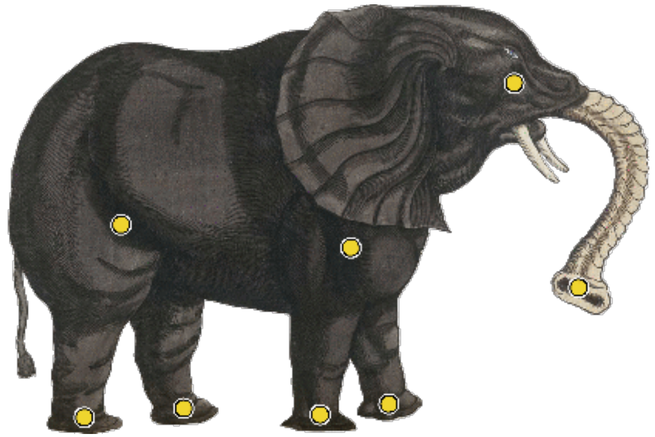
Acknowledgements: Daniele Panozzo

LBS generalizes to different handle types

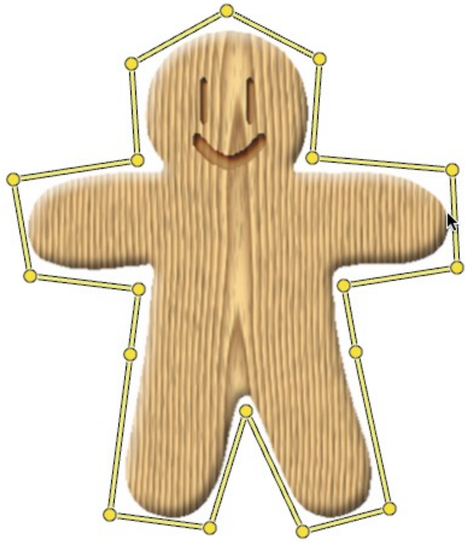


skeletons

regions

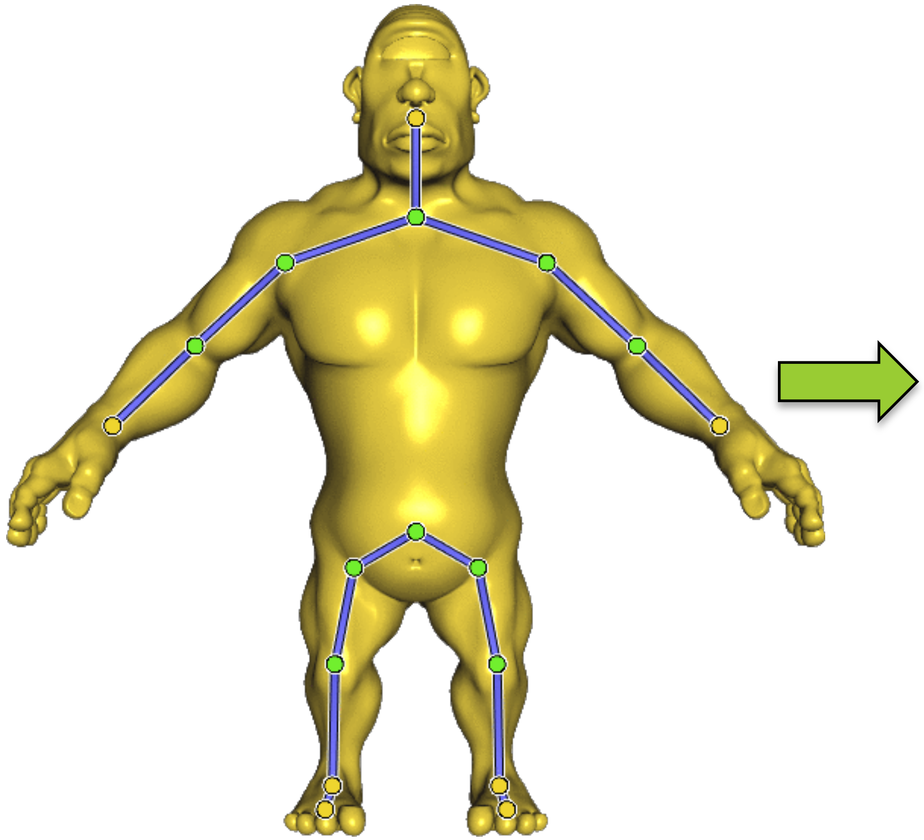


points



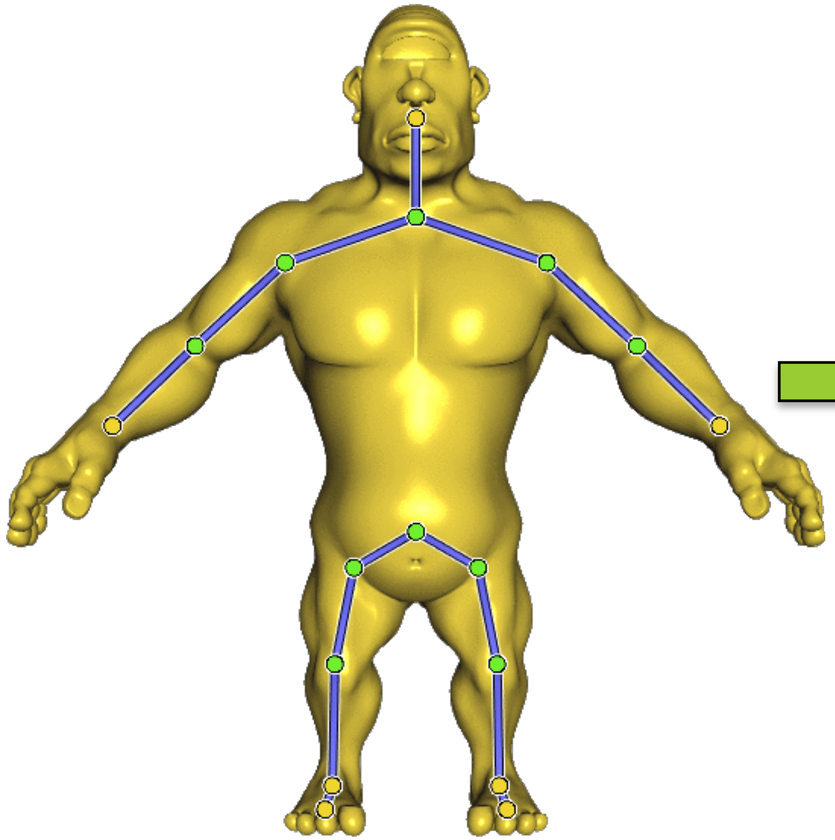
cages

Linear Blend Skinning rigging preferred for its real-time performance

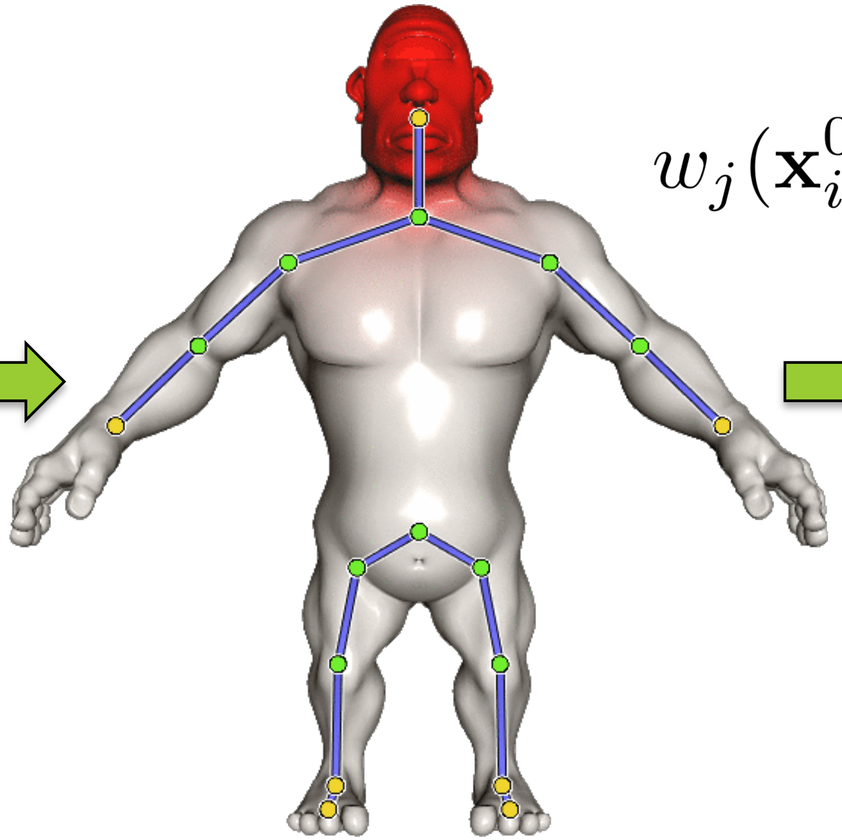


place handles in shape

Linear Blend Skinning rigging preferred for its real-time performance

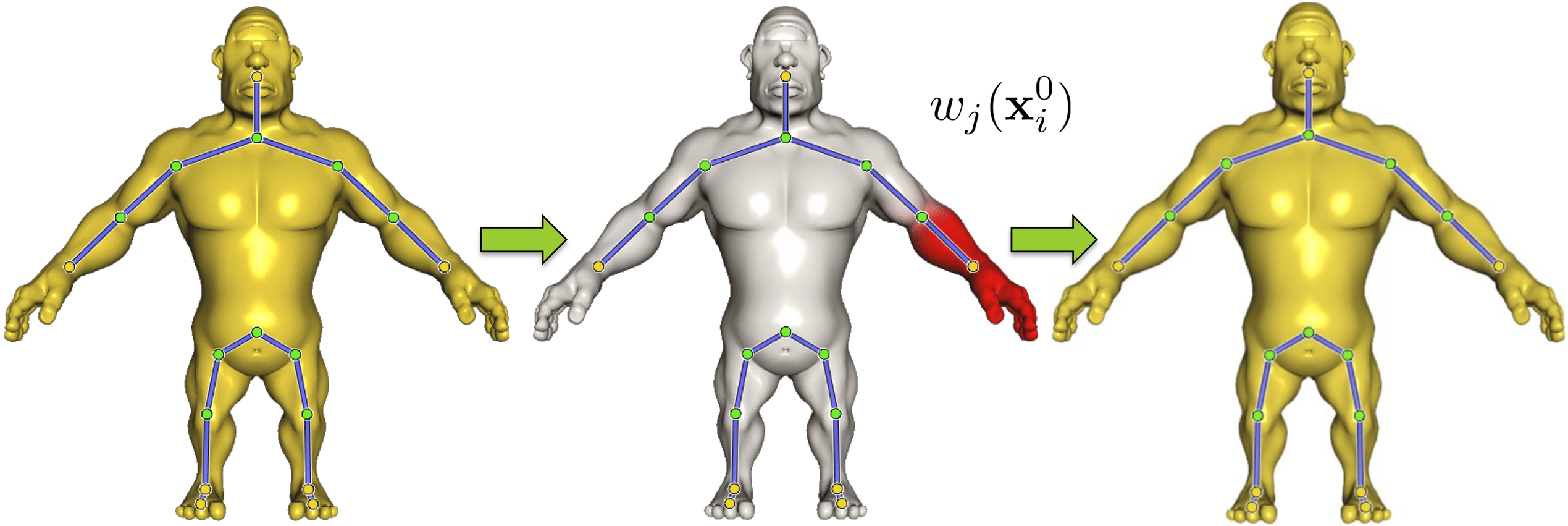


place handles in shape



paint weights

Linear Blend Skinning rigging preferred for its real-time performance

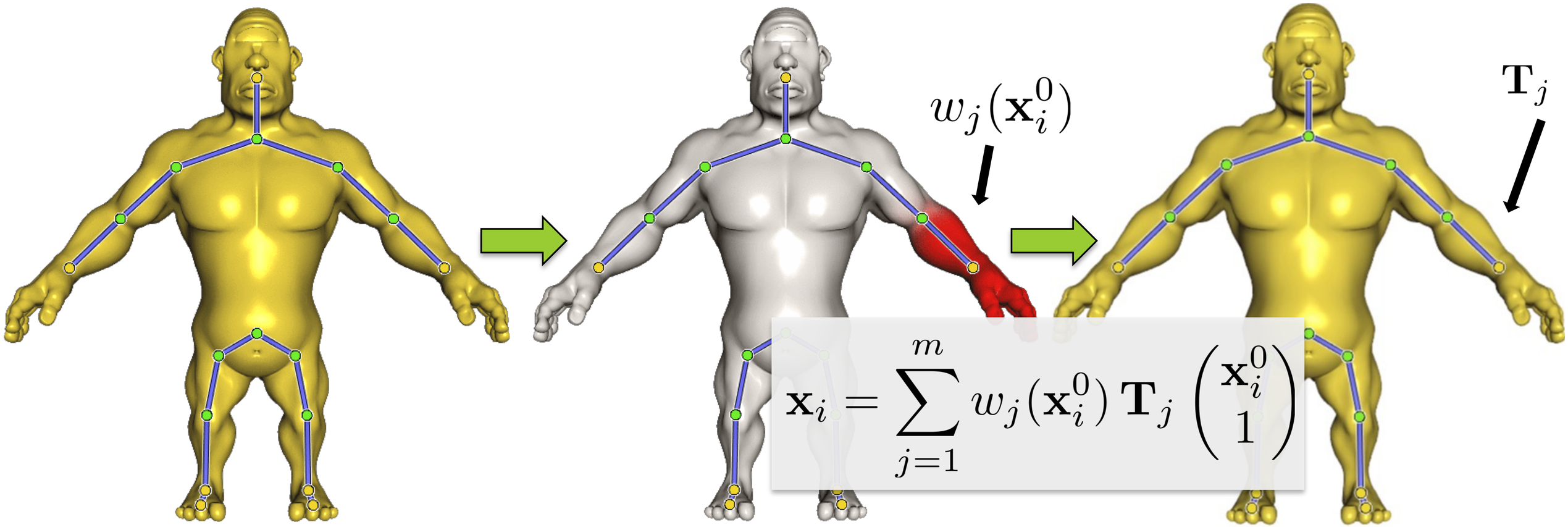


place handles in shape

paint weights

deform handles

Linear Blend Skinning rigging preferred for its real-time performance

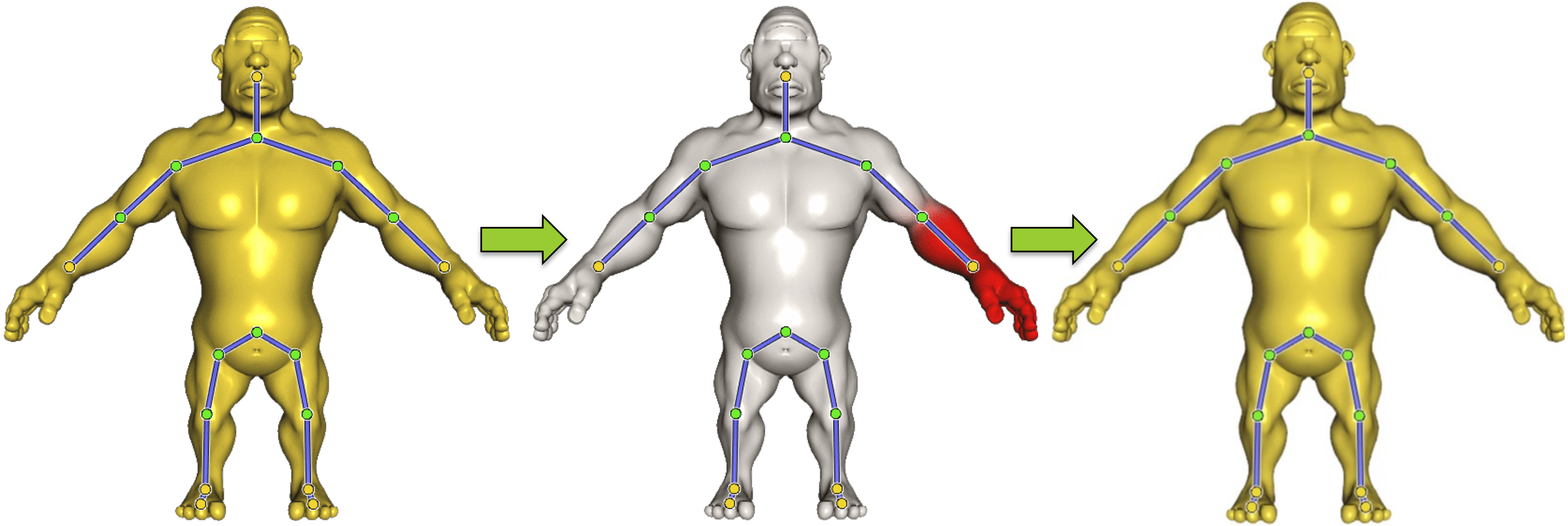


place handles in shape

paint weights

deform handles

Linear Blend Skinning rigging preferred for its real-time performance



place handles in shape

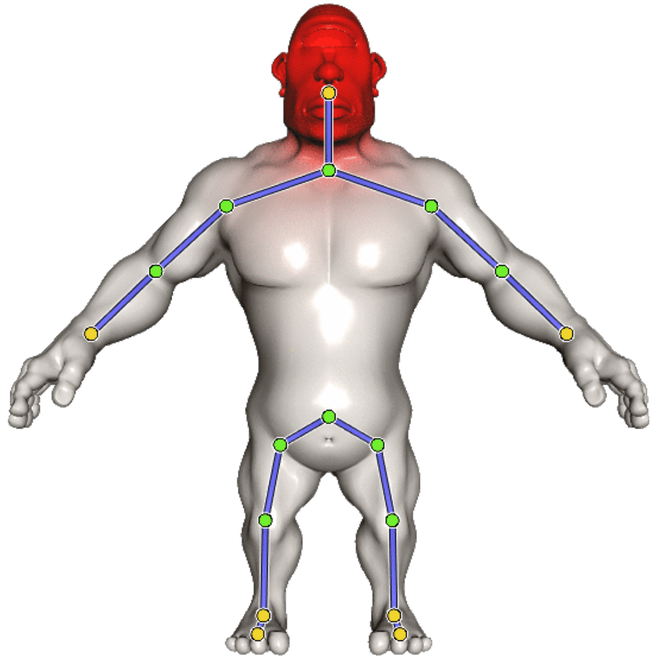
paint weights

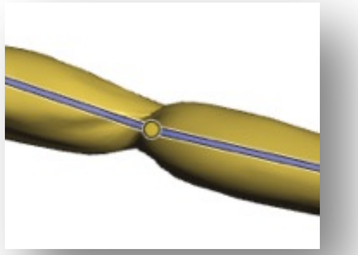
deform handles

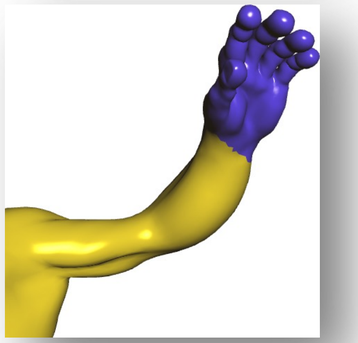
Challenges with LBS

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

$$\mathbf{x}_i = \sum_{j=1}^{m} w_j(\mathbf{x}_i^0)\, \mathbf{T}_j \begin{pmatrix} \mathbf{x}_i^0 \\ 1 \end{pmatrix}$$



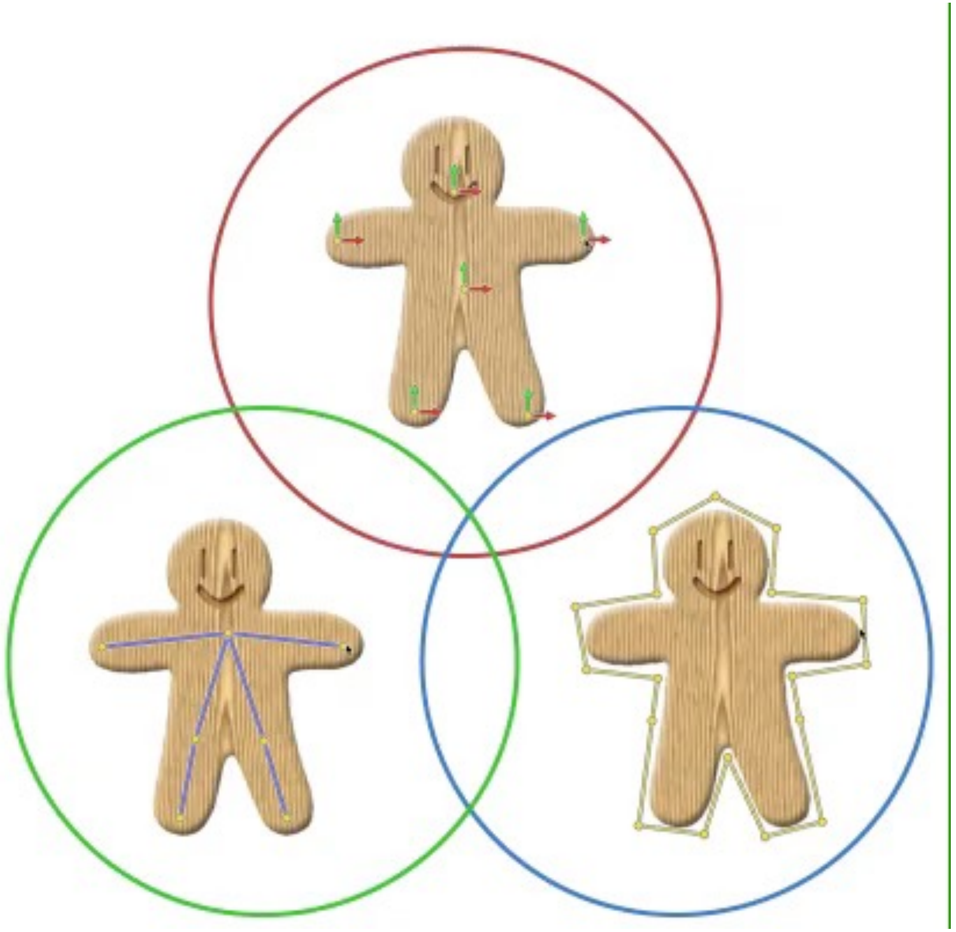




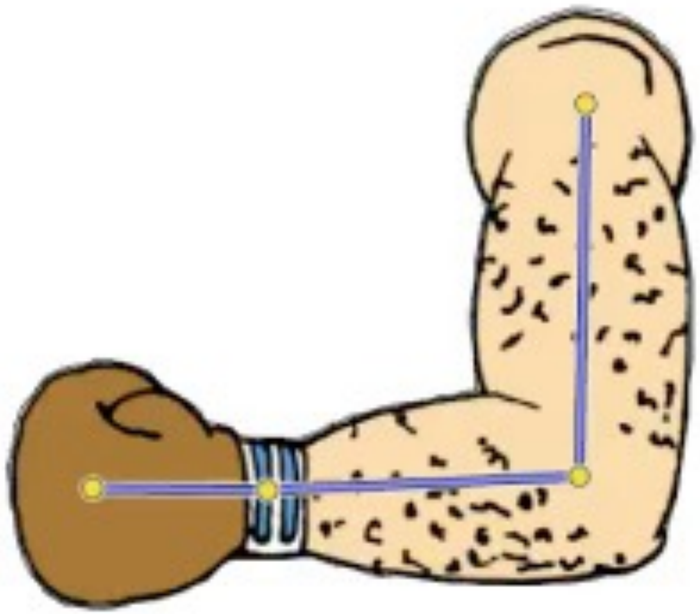
Bounded Biharmonic Weights

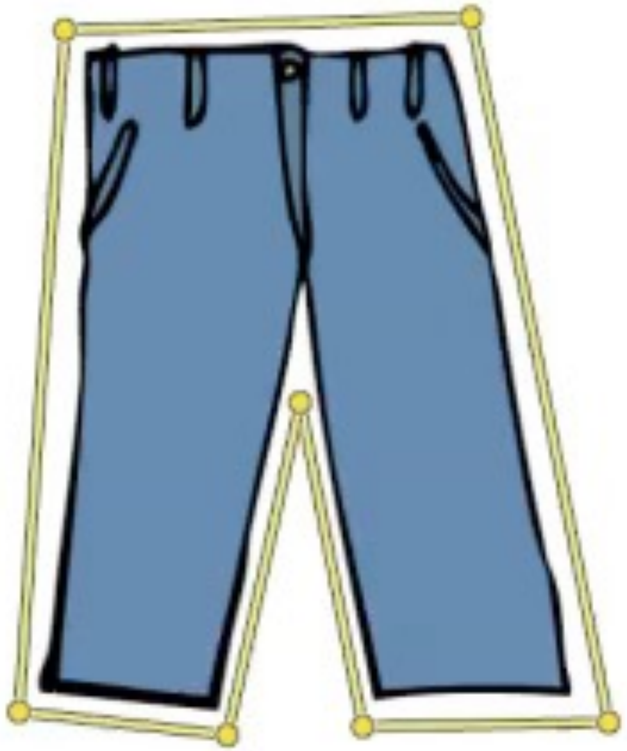
Alec Jacobson, Ilya Baran, Jovan Popović, Olga Sorkine-Hornung

Automatic weights that unify points, skeletons and cages



Weights should be smooth,
shape-aware, positive and ***intuitive***



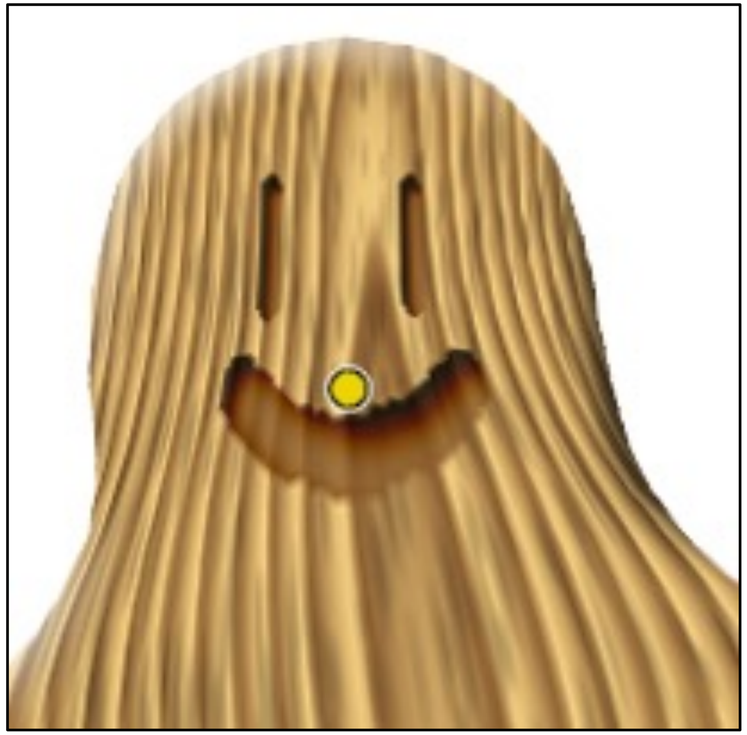


Weights must be smooth everywhere, ***especially*** at handles

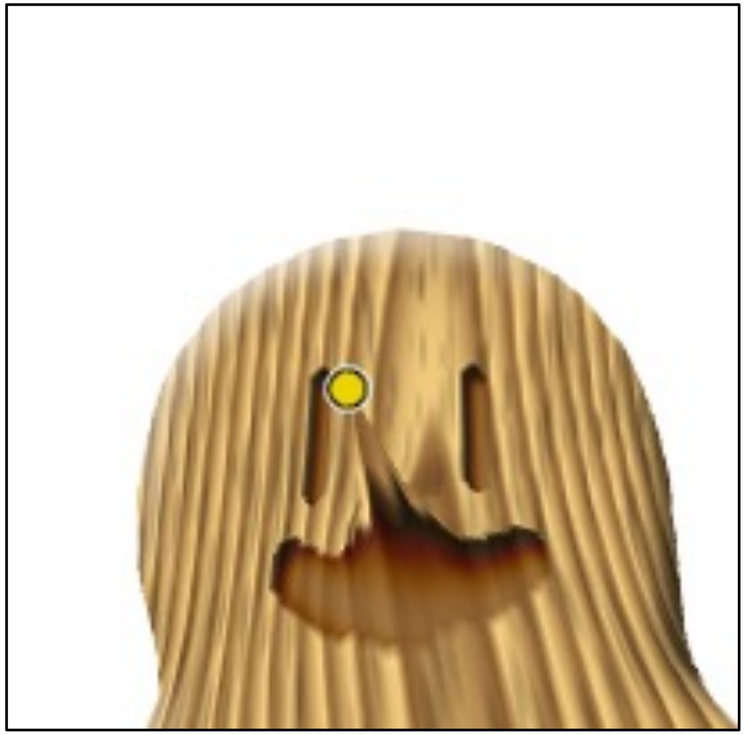
Bounded Biharmonic Weights

Extension of Harmonic Coordinates
[Joshi et al. 2005]

Weights must be smooth everywhere, ***especially*** at handles

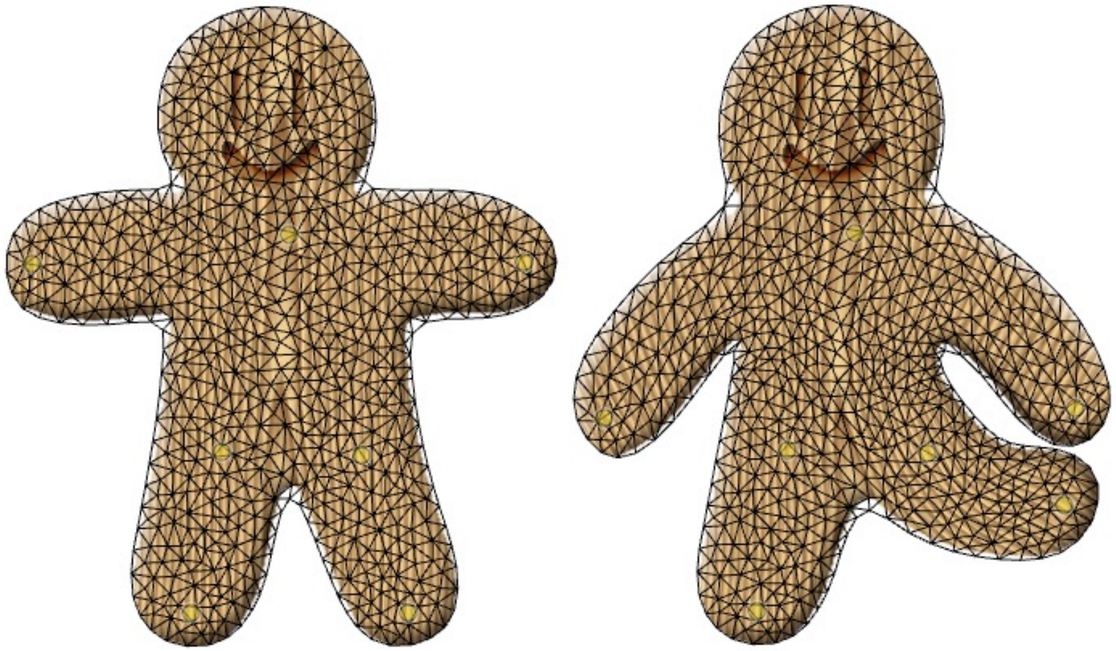


Bounded Biharmonic Weights

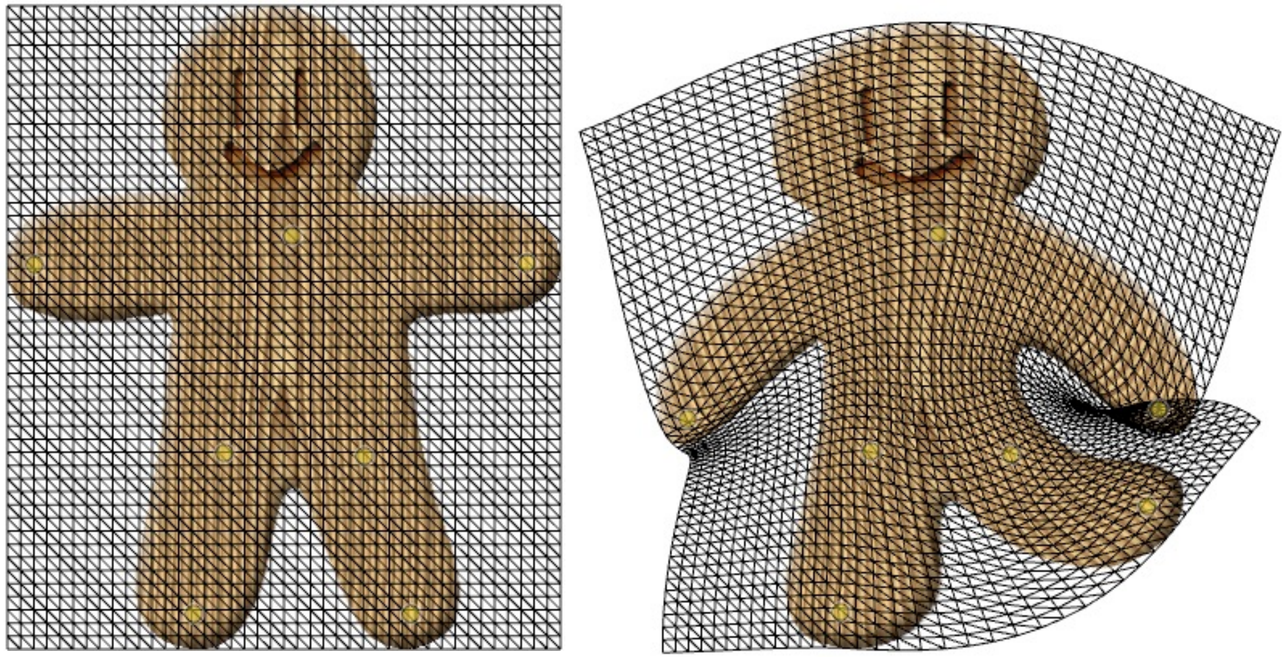


Extension of Harmonic Coordinates
[Joshi et al. 2005]

Shape-awareness ensures respect of domain's features



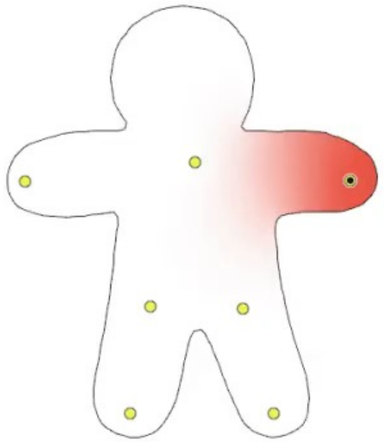
Bounded Biharmonic Weights



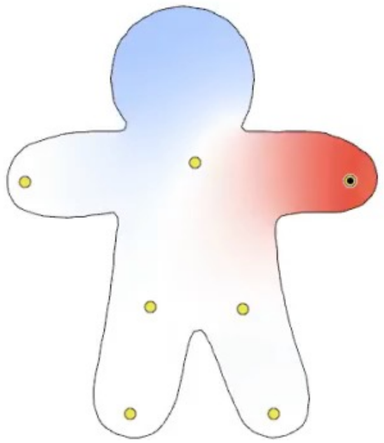
Non-shape-aware methods
e.g. [Schaefer et al. 2006]

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

$$\sum_{j \in H} w_j(\mathbf{x}^0) = 1$$

Partition of unity

Handle vertices

$$w_j \Big|_{H_k} = \delta_{jk}$$

w_j is linear along cage faces

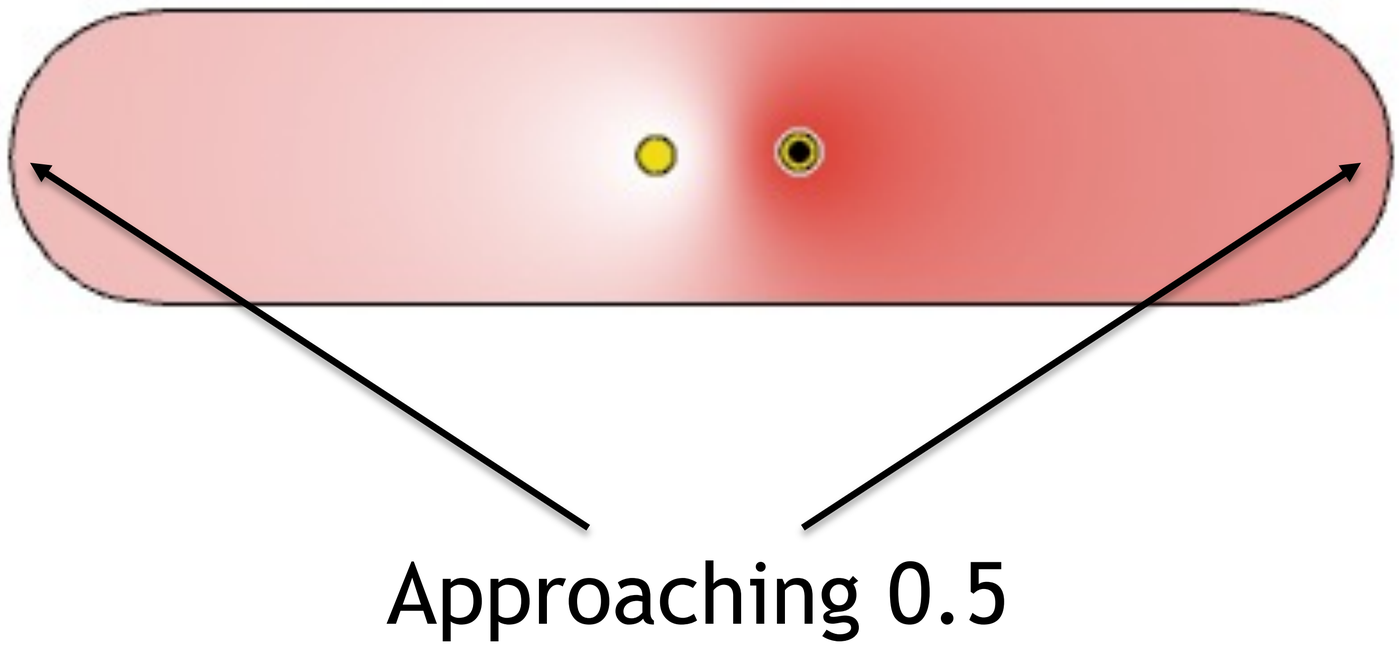
Interpolation of handles

How about $w_j(\mathbf{x}^0) = d(\mathbf{x}^0, H_j)^{-1}$?

Inverse distance methods inherently suffer from *fall-off effect*

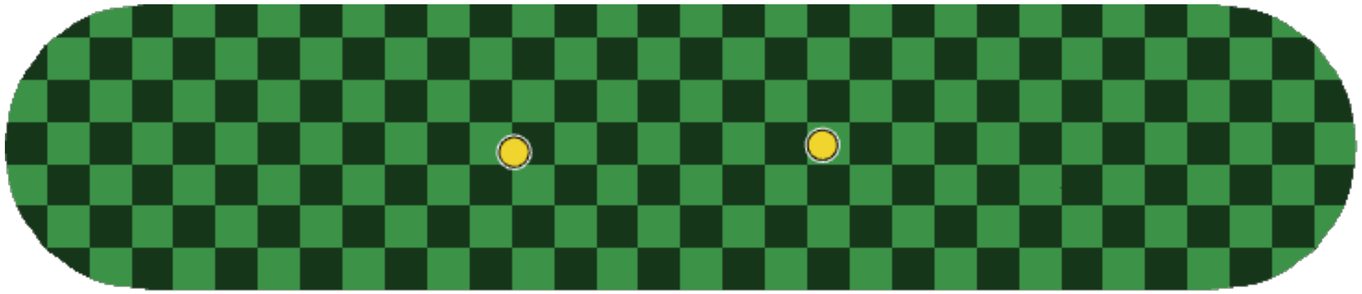
Inverse distance methods inherently suffer from ***fall-off effect***

Inverse distance methods inherently suffer from ***fall-off effect***

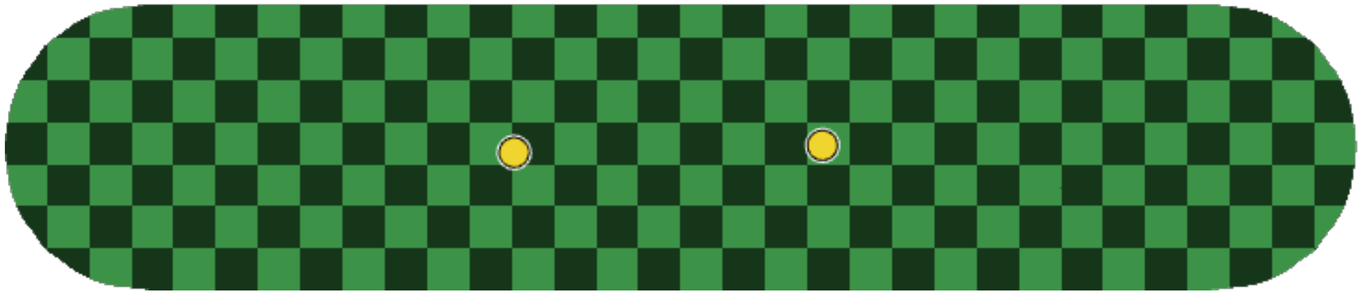


Inverse distance methods inherently suffer from *fall-off effect*

Inverse-distance weights



BBW



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

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

$$0 \leq w_j(\mathbf{x}^0) \leq 1$$

Partition of unity

$$\sum_{j \in H} w_j(\mathbf{x}^0) = 1$$

Bounded biharmonic weights enforce properties as constraints to minimization

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

$$w_j\Big|_{H_k} = \delta_{jk}$$

w_j is linear along cage faces

Constant inequality constraints

$$0 \leq w_j(\mathbf{x}^0) \leq 1$$

Solve independently and normalize

$$w_j(\mathbf{x}^0) = \frac{w_j(\mathbf{x}^0)}{\sum_{i\in H} w_i(\mathbf{x}^0)}$$

Weights optimized as precomputation at bind-time

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

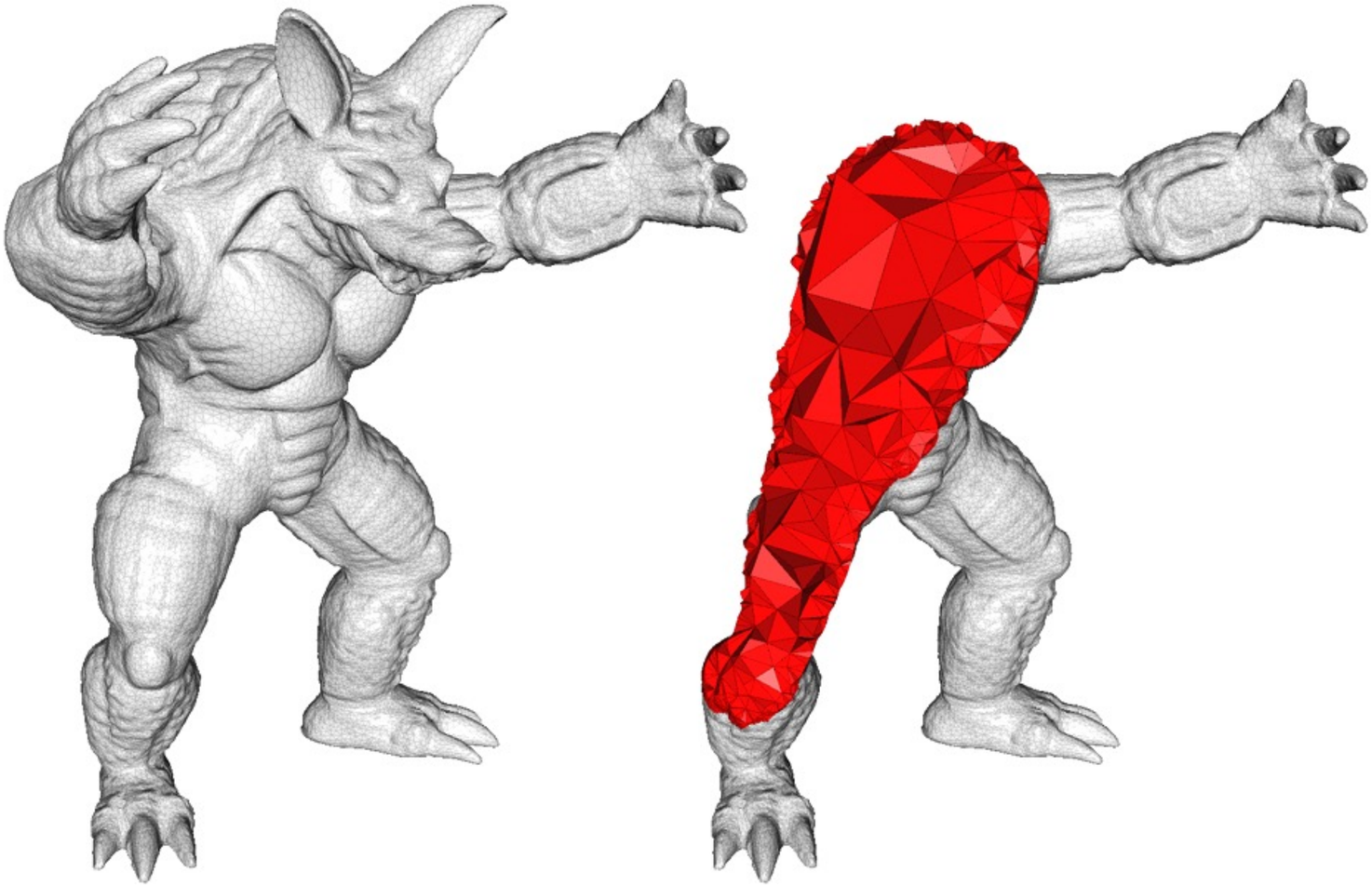
$$w_j\big|_{H_k} = \delta_{jk}$$

w_j is linear along cage faces

$$0 \le w_j(\mathbf{x}^0) \le 1$$

FEM discretization

2D → Triangle mesh
3D → Tet mesh



Weights optimized as precomputation at bind-time

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

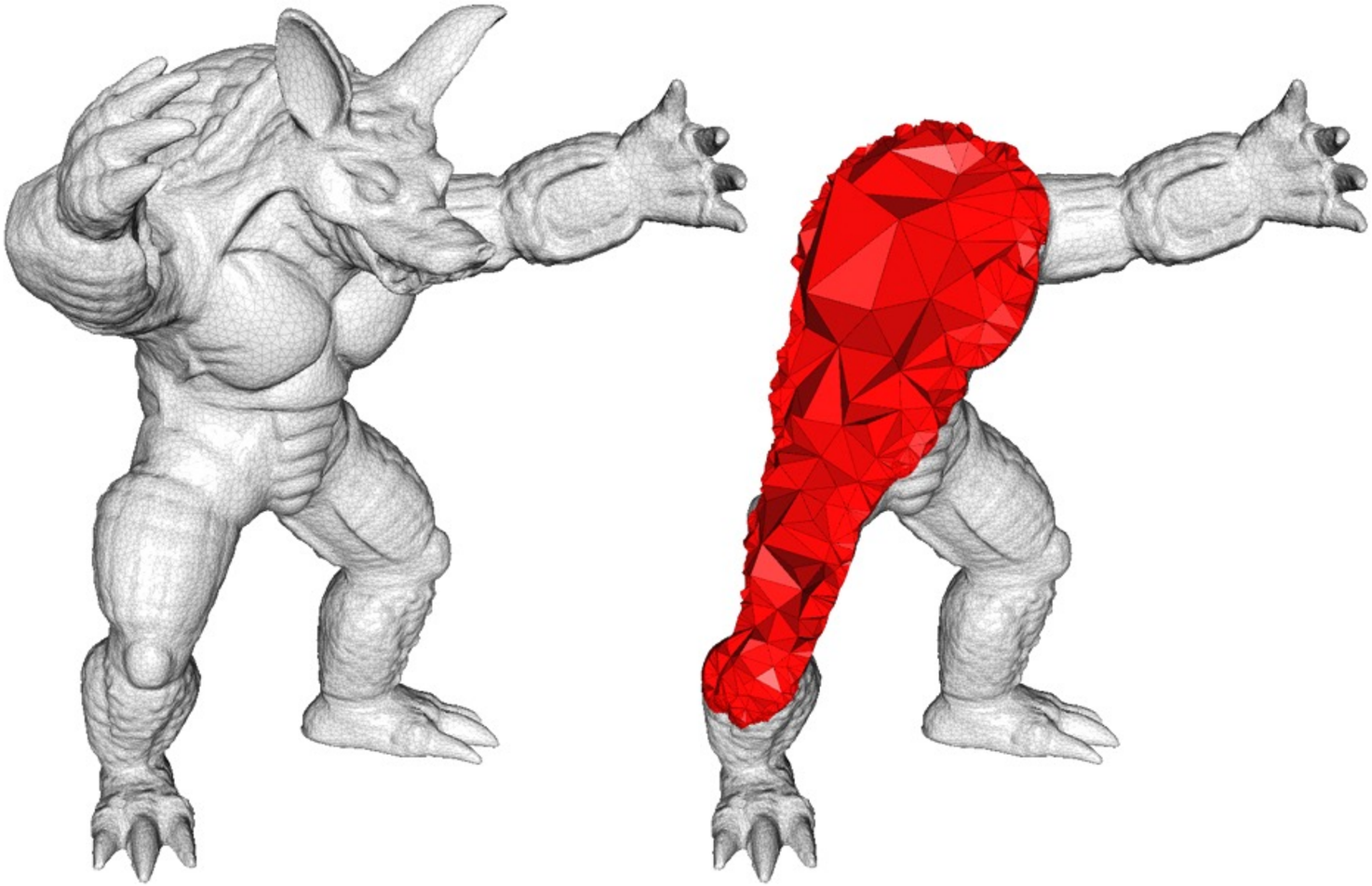
$$w_j\big|_{H_k} = \delta_{jk}$$

w_j is linear along cage faces

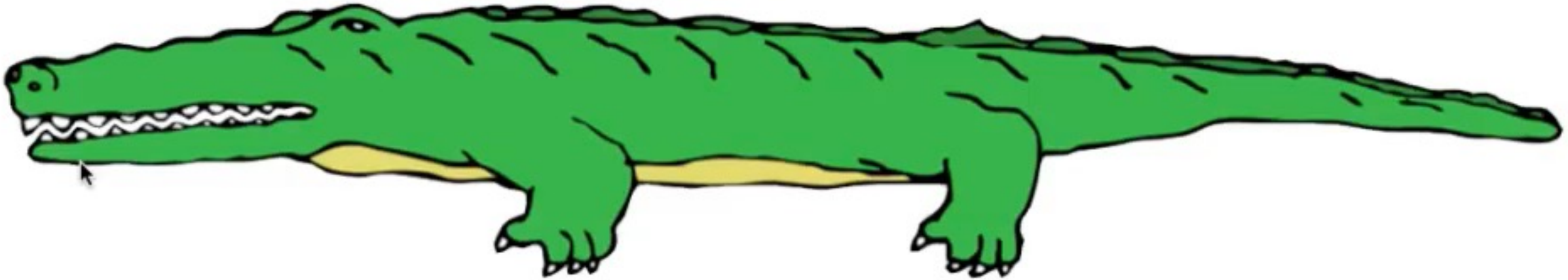
$$0 \le w_j(\mathbf{x}^0) \le 1$$

Sparse quadratic programming with constant inequality constraints

- 2D → less than second per handle
- 3D → tens of seconds per handle

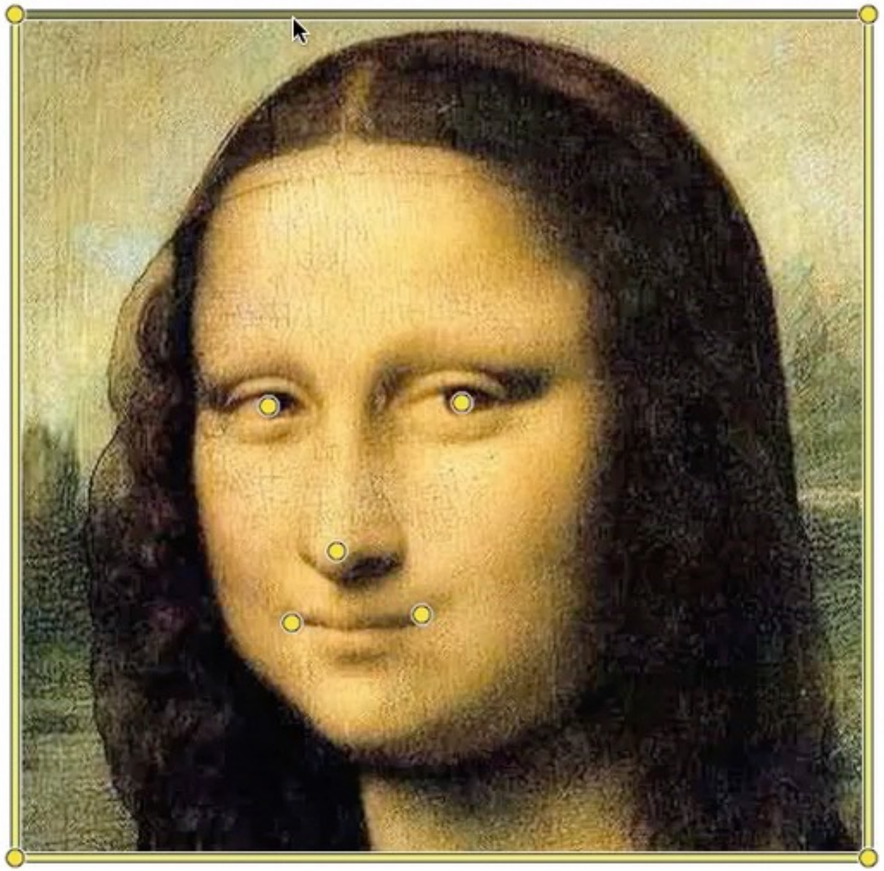


Some examples of BBW in action

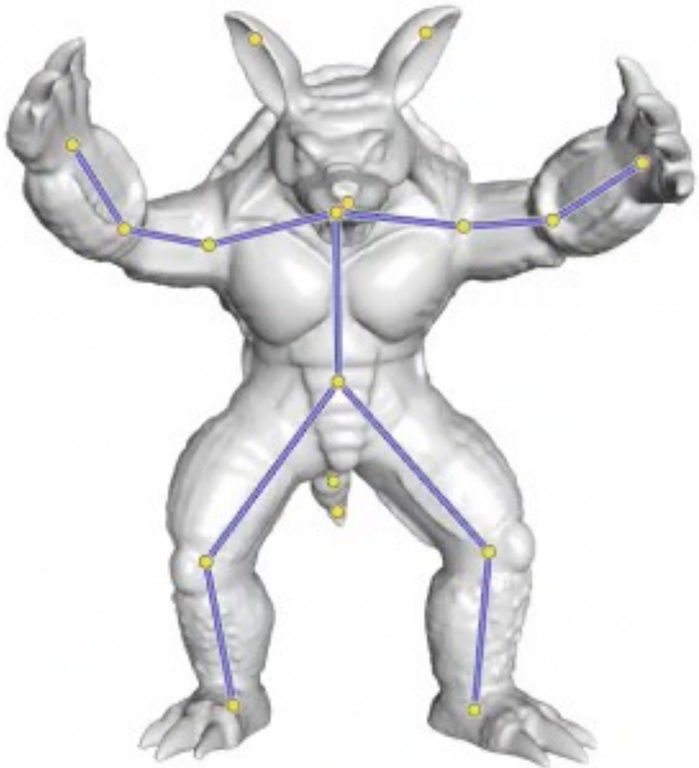


Some examples of BBW in action

Some examples of BBW in action



3D Characters



Demo

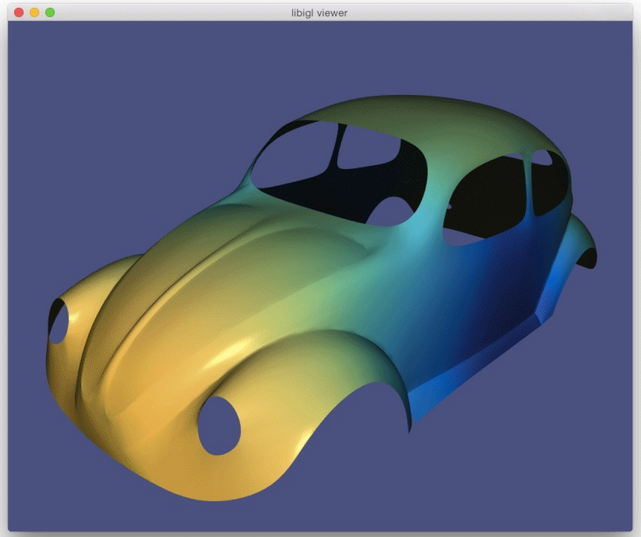
- Libigl demo 403

08 - Summary

Acknowledgements: Daniele Panozzo

Geometric Modeling and Processing

- The shape of an object is an important characteristic (not the only one…)

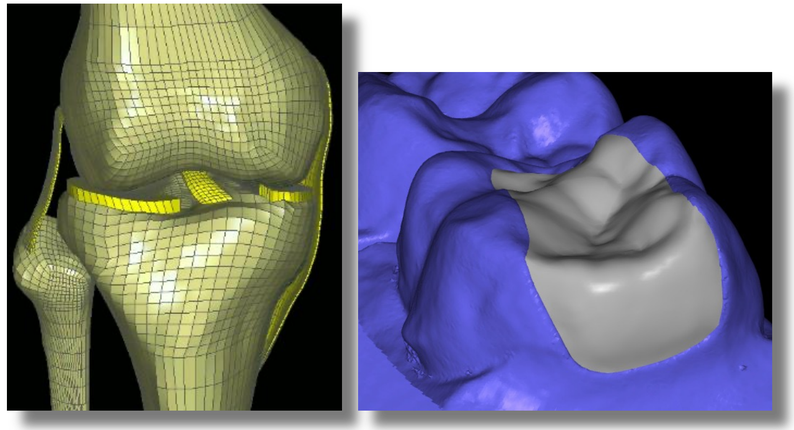


- Geometry processing: computerized modeling of 2D/3D geometry

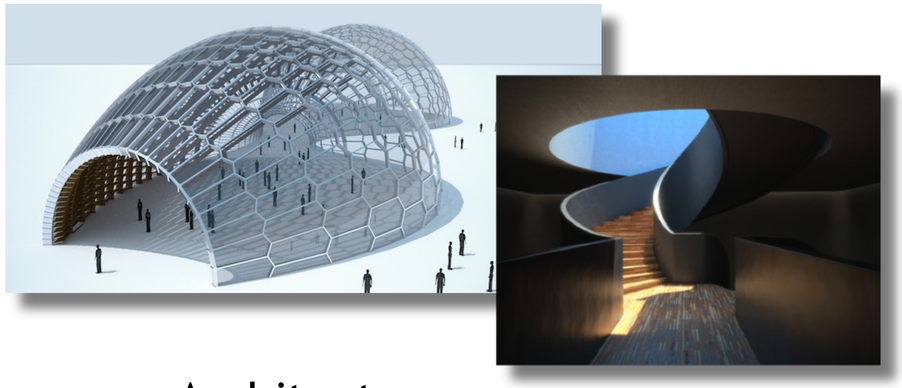
Copyright: Blender

Applications

Product design and prototyping



Medicine, prosthetics



Architecture

Cultural heritage

C++

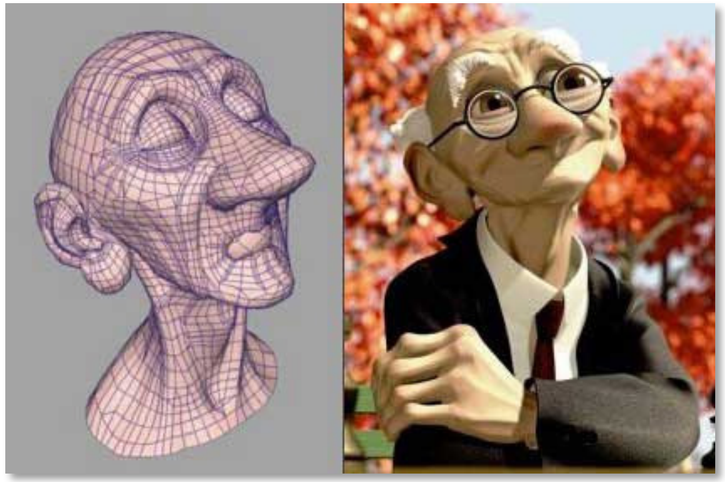
- It is flexible, and many of the features are optional:
  - it can be used as an extension of C, with no objects
  - it can be used as a fully object oriented language
  - it has many advanced features such as “templates” that are useful to write efficient code that *is also readable*

CMAKE

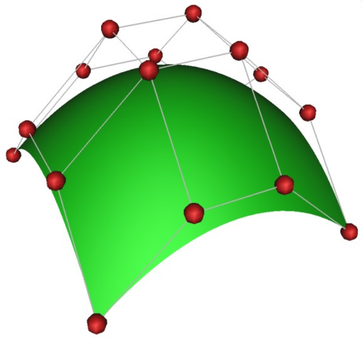
- If your project is in a folder Assignment_1, you need to:
  - mkdir build; cd build
  - cmake ../
- This will create the project. To compile it:
  - make (macosx/linux)
  - Open the project with visual studio on windows
- As an alternative, you can use Clion/Xcode/VS code that does all of this for you!

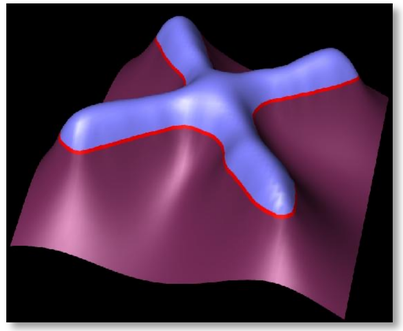
Course Topics

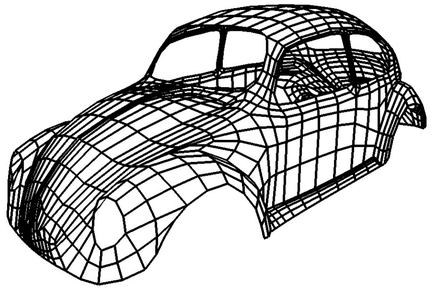
- Overview of shape representations
  - Parametric curves/surfaces
  - Implicits
  - Polygonal meshes



Pixar

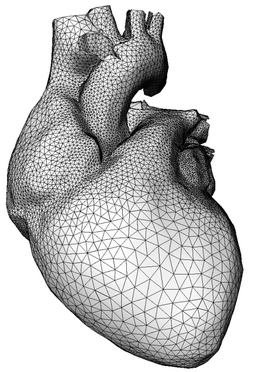


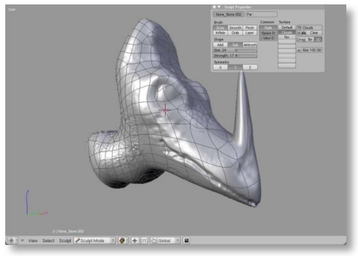


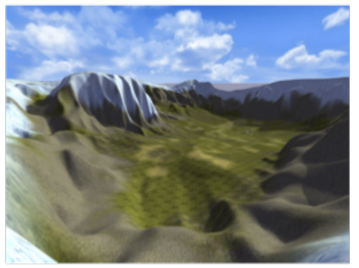


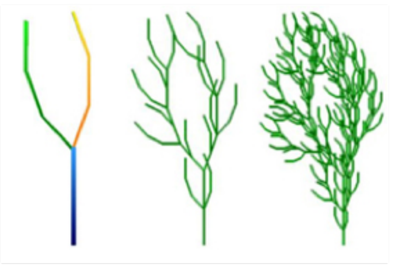
Shape Representation: Origin- and Application-Dependent

- Acquired real-world objects:
  - Discrete sampling
  - Points, meshes
- Modeling “by hand”:
  - Higher-level representations, amendable to modification, control
  - Parametric surfaces, subdivision surfaces, implicits
- Procedural modeling
  - Algorithms, grammars



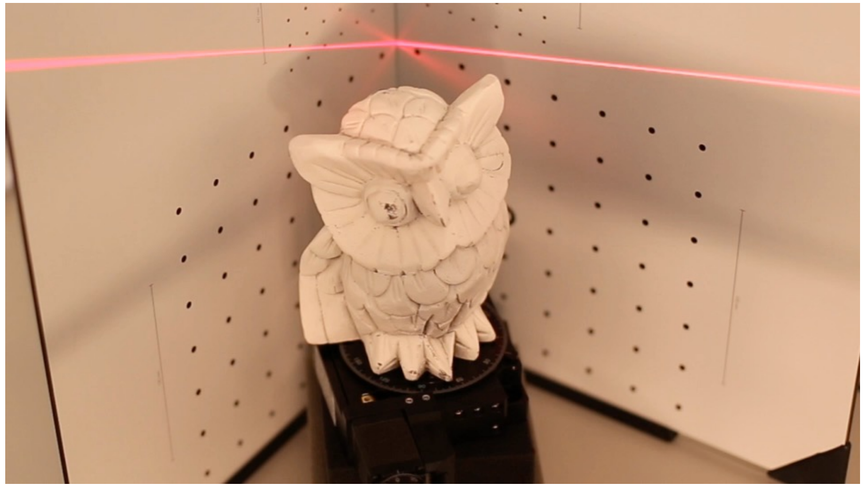


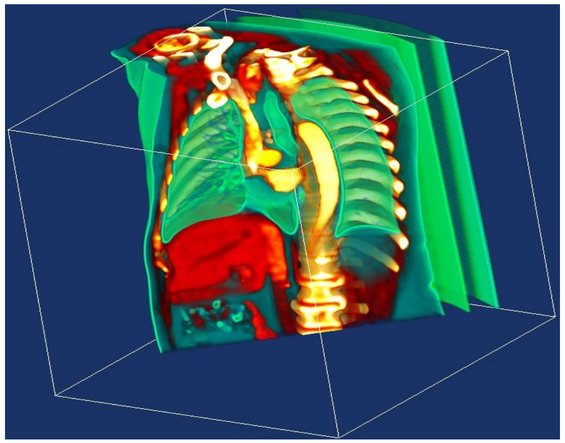


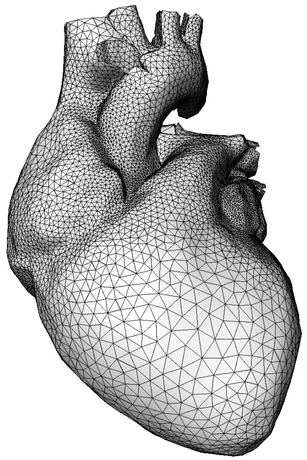


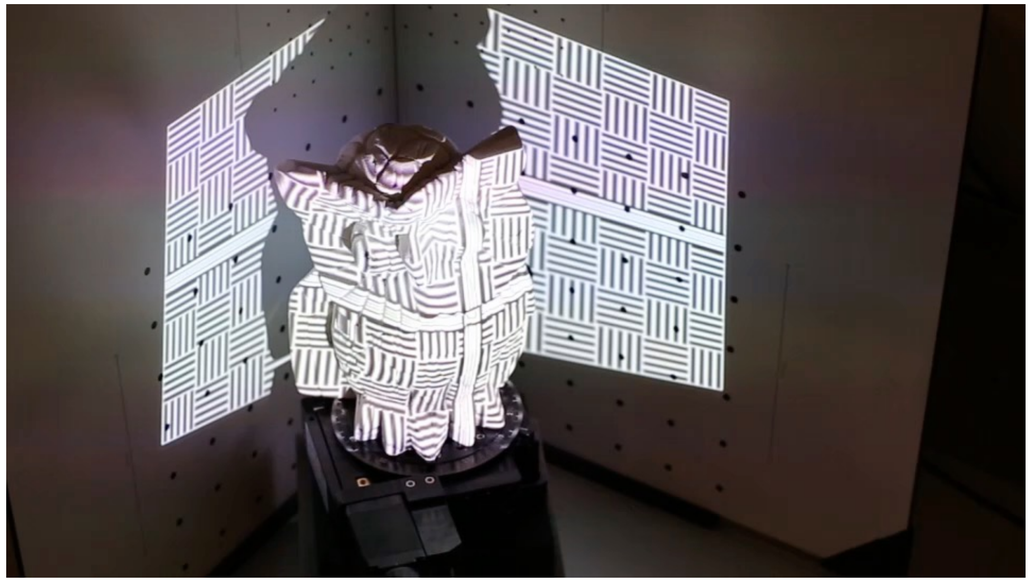
Course Topics

- Shape acquisition
  - Scanning/imaging
  - Reconstruction









Geometry Acquisition Pipeline

Scanning:
results in range images

Registration:
bring all range images to one coordinate system

Stitching/reconstruction:
Integration of scans into a single mesh

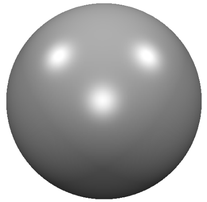
Postprocess:
- Topological and geometric filtering
- Remeshing
- Compression

Euler-Poincaré Formula

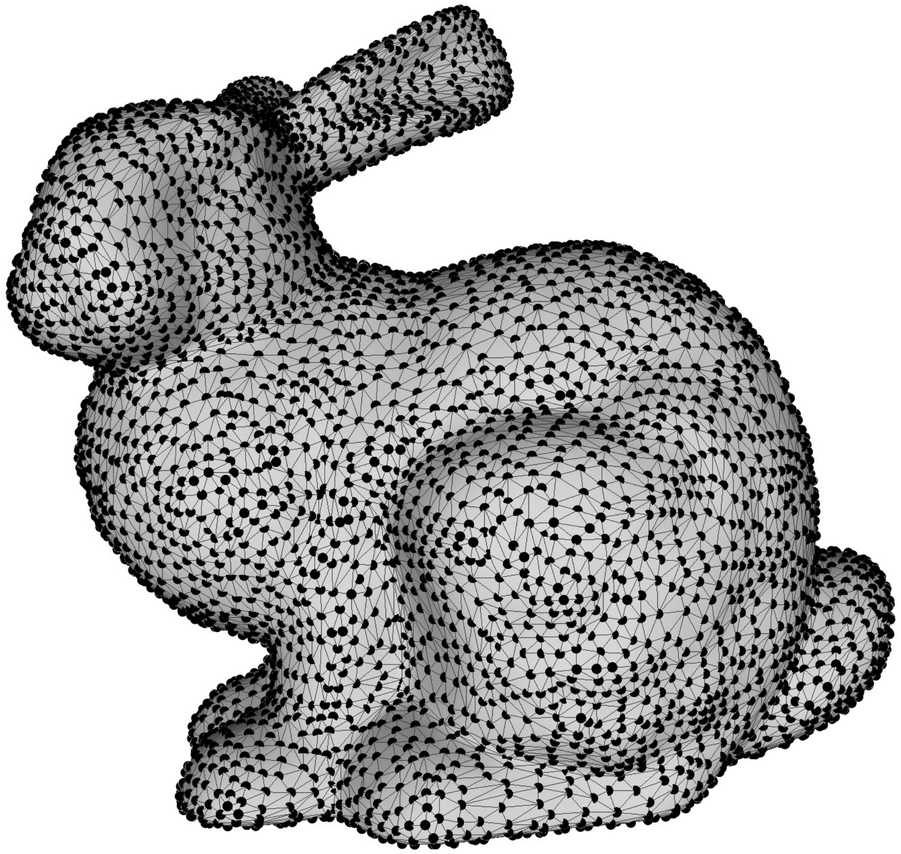
- For orientable meshes:

$$v - e + f = 2(c - g) - b = \chi(M)$$

  - c = number of connected components
  - g = genus
  - b = number of boundary loops

$\chi($  $) = 2$ $\chi($  $) = 0$

Problem Assumptions

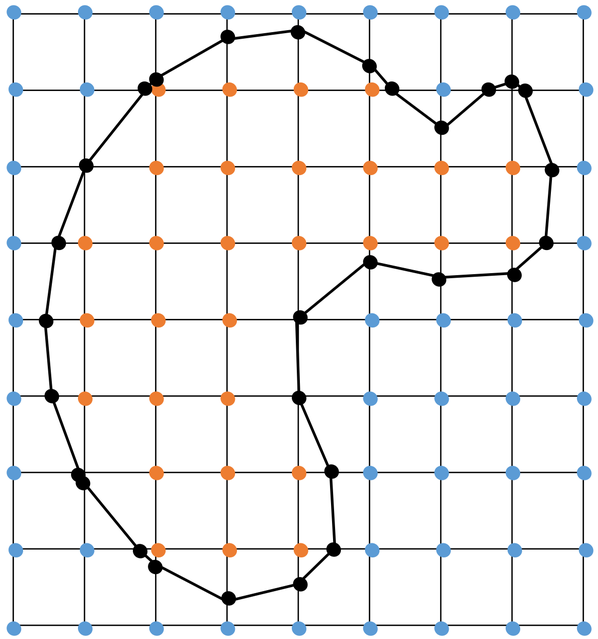


Interpolation

Approximation

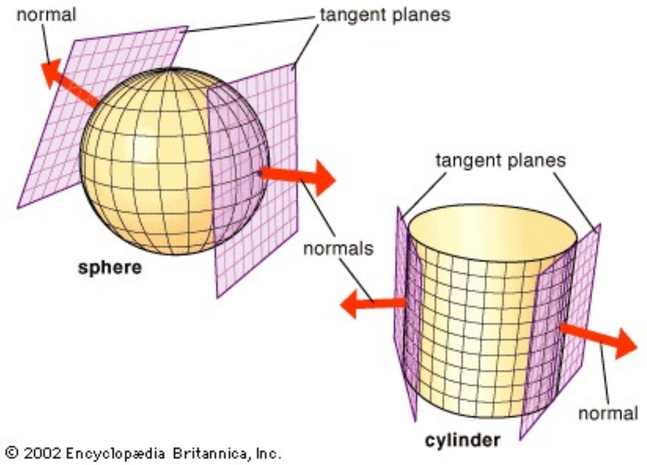
Implicit Reconstruction

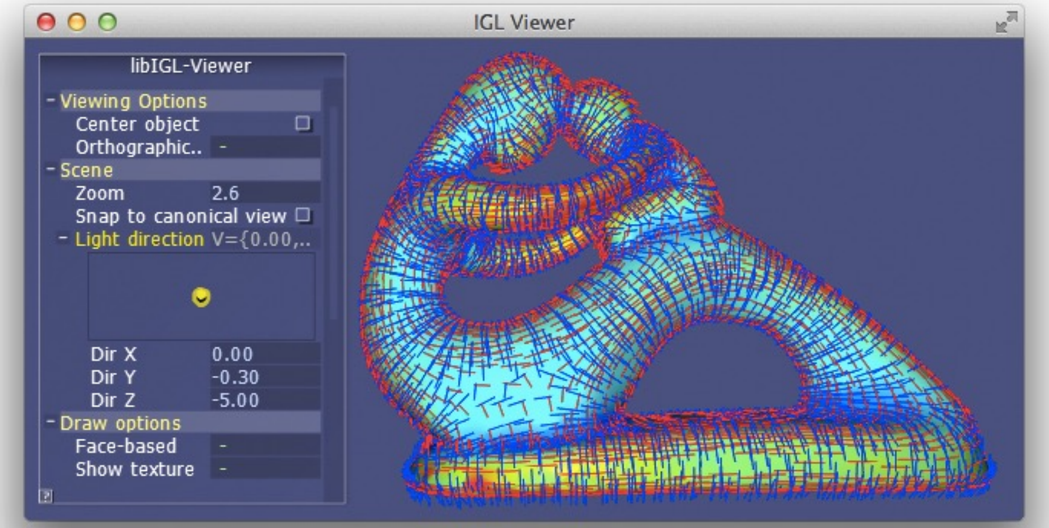
1. Estimate normal (Optional)
2. Compute function $F(x)$
3. Discretize function $F(x)$
4. Extract zero Iso-surface



Course Topics

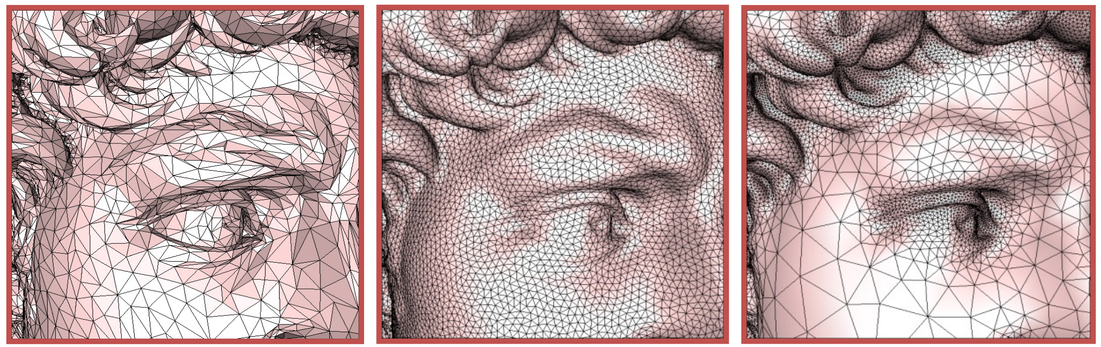
- Differential geometry
  - Continuous and (mostly) discrete
  - Powerful tool to analyze and model shapes

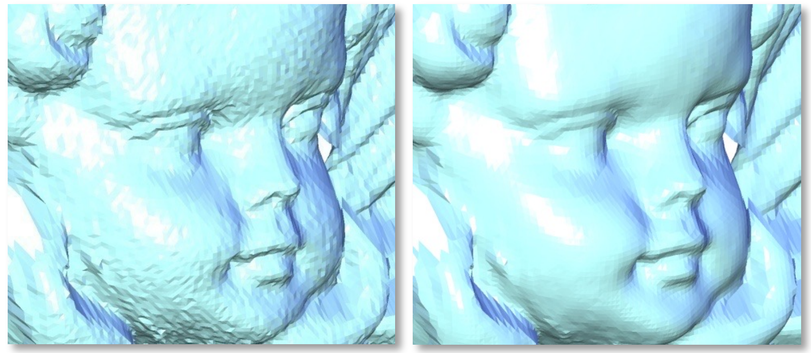




Course Topics

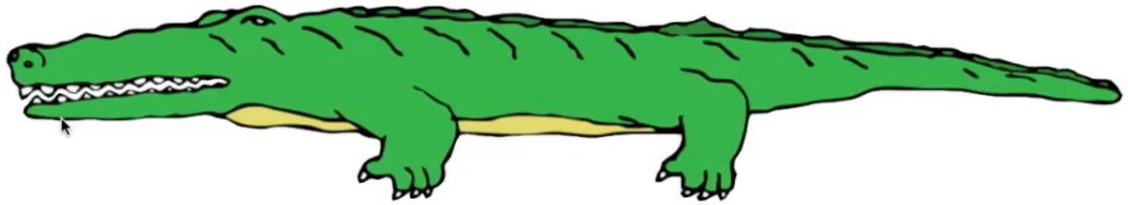
- Digital geometry processing
  - Denoising, smoothing, simplification, remeshing, parameterization, compression



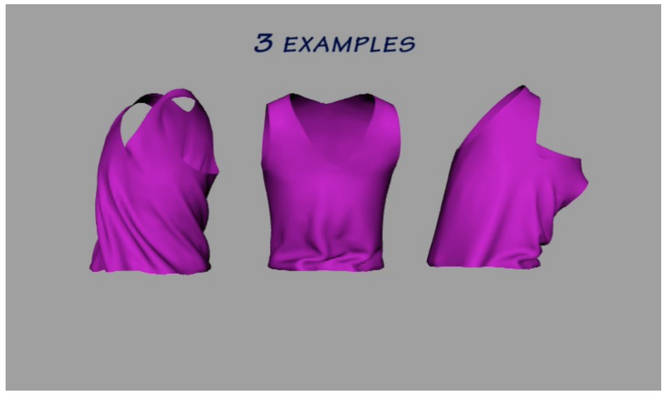


Course Topics

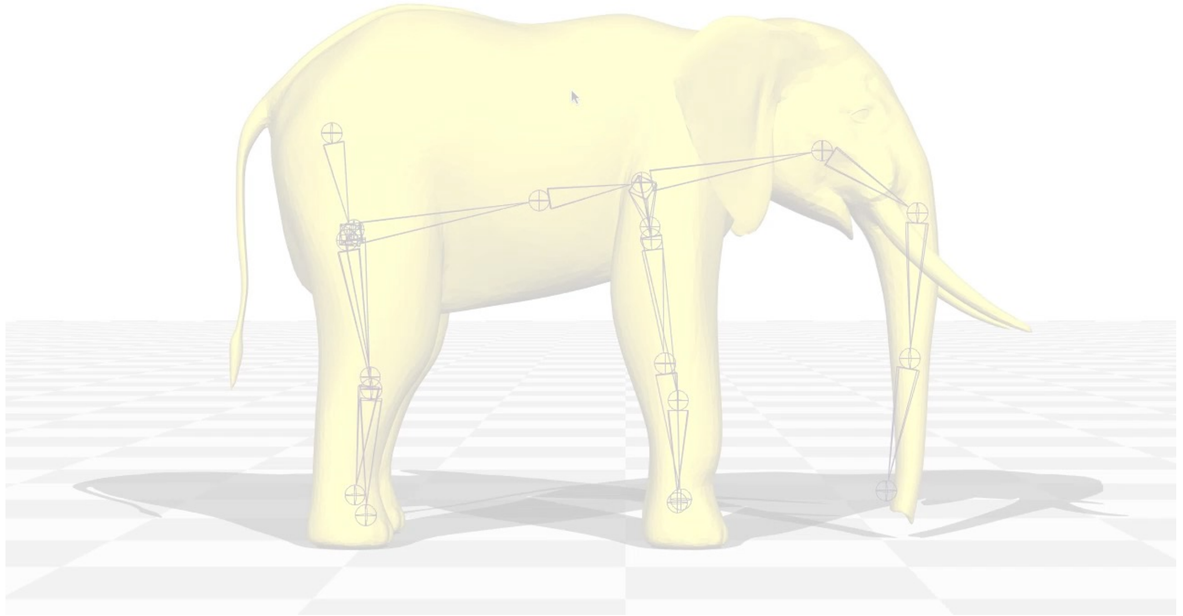
- Skinning, animation



http://youtu.be/P9fqm8vgdB8



http://youtu.be/Pjg33pH9RKo



Thank you