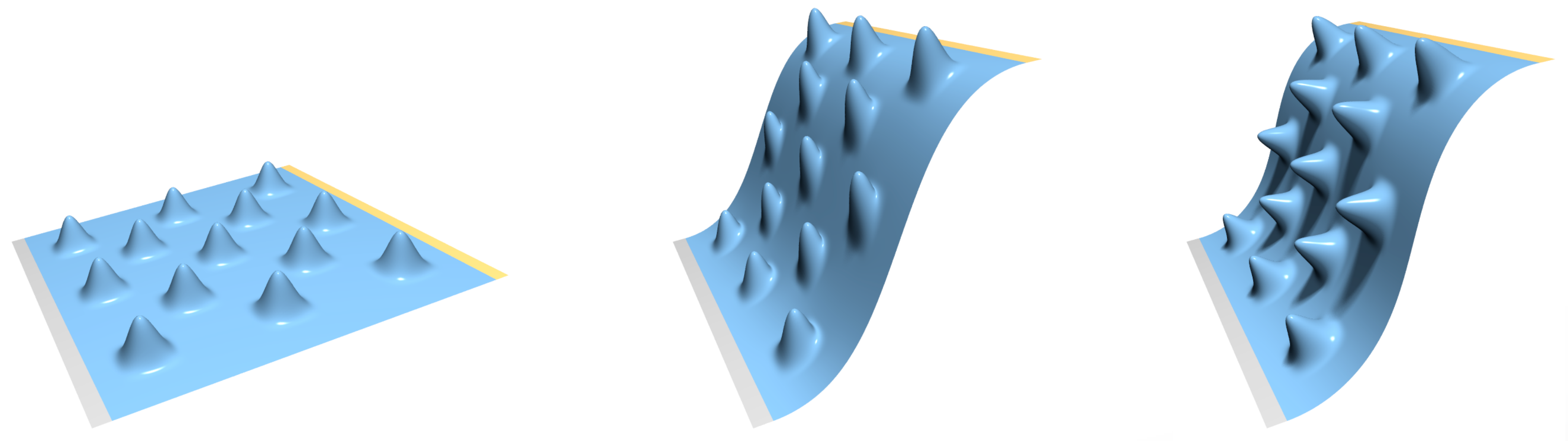
12.1 **Surface Deformation II**



Hao Li

http://cs621.hao-li.com

Last Time

Linear Surface Deformation Techniques

- Shell-Based Deformation
- Multiresolution Deformation
- Differential Coordinates

Nonlinear Surface Deformation

- **Nonlinear Optimization**
- Shell-Based Deformation
- (Differential Coordinates)

Nonlinear Optimization

- Given a nonlinear deformation energy

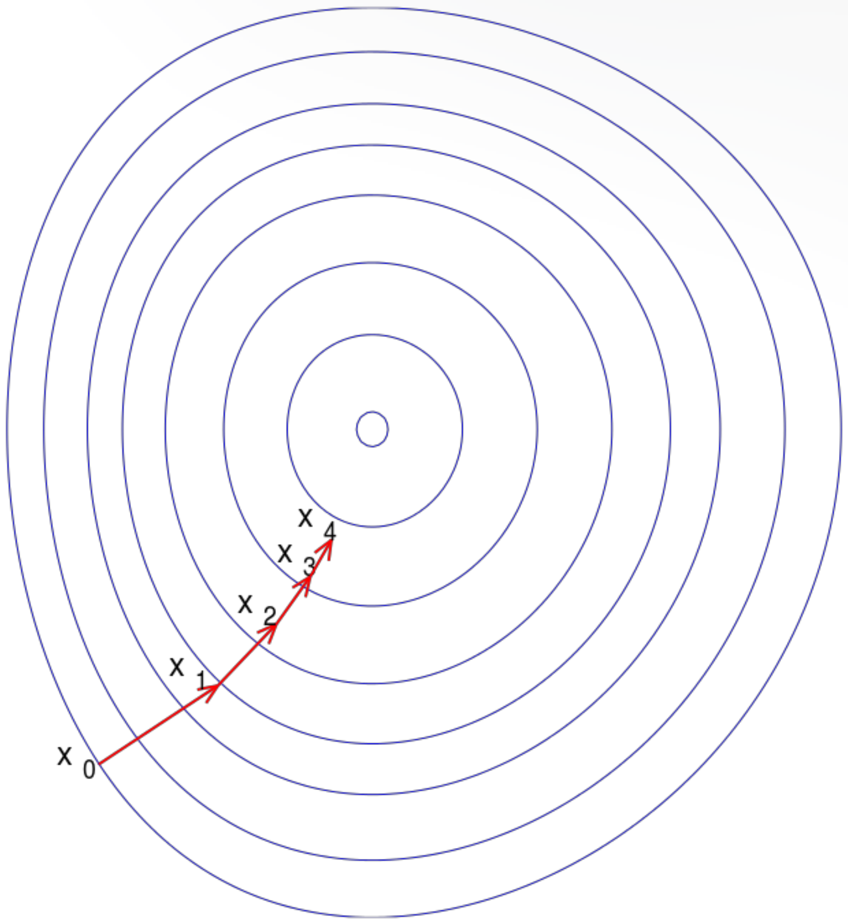
$$E(\mathbf{d}) \;=\; E(\mathbf{d}_1, \ldots, \mathbf{d}_n)$$

find the displacement $\mathbf{d}(\mathbf{x})$ that minimizes $E(\mathbf{d})$, while satisfying the modeling constraints.

- Typically $E(\mathbf{d})$ stays the same, but the modeling constraints change each frame.

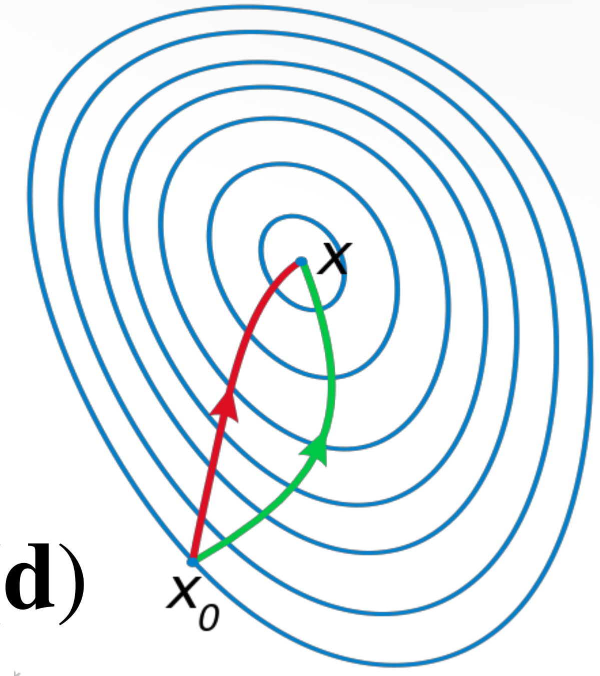
Gradient Descent

- Start with initial guess $\mathbf{d}_0$
- Iterate until convergence
  - Find descent direction $\mathbf{h} = -\nabla E(\mathbf{d})$
  - Find step size λ
  - Update $\mathbf{d} = \mathbf{d} + \lambda\mathbf{h}$



- Properties
  + Easy to implement, guaranteed convergence
  – Slow convergence

Newton's Method



- Start with initial guess $\mathbf{d}_0$
- Iterate until convergence
  – Find descent direction as $\mathbf{H}(\mathbf{d})\,\mathbf{h} = -\nabla E(\mathbf{d})$
  – Find step size λ
  – Update $\mathbf{d} = \mathbf{d} + \lambda\mathbf{h}$
- Properties
  + Fast convergence if close to minimum
  – Needs pos. def. $\mathbf{H}$, needs 2nd derivatives for $\mathbf{H}$

Nonlinear Least Squares

Given a nonlinear vector-valued error function

$$\mathbf{e}(\mathbf{d}_1, \ldots, \mathbf{d}_n) \;=\; \begin{pmatrix} e_1(\mathbf{d}_1, \ldots, \mathbf{d}_n) \\ \vdots \\ e_m(\mathbf{d}_1, \ldots, \mathbf{d}_n) \end{pmatrix}$$

find the displacement $\mathbf{d}(\mathbf{x})$ that minimizes the nonlinear least squares error

$$E(\mathbf{d}_1, \ldots, \mathbf{d}_n) \;=\; \frac{1}{2}\,\|\mathbf{e}(\mathbf{d}_1, \ldots, \mathbf{d}_n)\|^2$$

1st order Taylor Approximation

$$E(\mathbf{d}_1, \ldots, \mathbf{d}_n) \;=\; \frac{1}{2}\left\|\mathbf{e}(\mathbf{d}_1, \ldots, \mathbf{d}_n)\right\|^2$$

$$\|\mathbf{e}(\mathbf{d}^{k+1})\|^2 \approx \|\mathbf{e}(\mathbf{d}^k) + J_{\mathbf{e}}(\mathbf{d}^{k+1} - \mathbf{d}^k)\|^2$$

$$\|\mathbf{e}(\mathbf{d}^{k+1})\|^2 \approx \|\mathbf{e}(\mathbf{d}^k) + J_{\mathbf{e}}\Delta\mathbf{d}^k\|^2$$

Taylor Approx

$$\Delta\mathbf{d}^k_{\min} = \arg\min_{\Delta\mathbf{d}^k} \|\mathbf{e}\|^2$$

$$\mathbf{h} = \arg\min_{\Delta\mathbf{d}^k} \|\mathbf{e}\|^2$$

$$J_{\mathbf{e}}^\top J_{\mathbf{e}}\mathbf{h} = -J_{\mathbf{e}}^\top \mathbf{e}(\mathbf{d}^k)$$

Gauss-Newton

Gauss-Newton Method

- Start with initial guess $\mathbf{d}_0$
- Iterate until convergence
  - – Find descent direction as $(\mathbf{J}(\mathbf{d})^T \mathbf{J}(\mathbf{d}))\, \mathbf{h} = -\mathbf{J}(\mathbf{d})^T \mathbf{e}$
  - – Find step size λ
  - – Update $\mathbf{d} = \mathbf{d} + \lambda\mathbf{h}$
- Properties
  - \+ Fast convergence if close to minimum
  - – Needs full-rank $\mathbf{J}(\mathbf{d})$, needs 1st derivatives for $\mathbf{J}(\mathbf{d})$

Nonlinear Optimization

- Has to solve a linear system each frame
  - Matrix changes in each iteration!
  - Factorize matrix each time
- Numerically more complex
  - No guaranteed convergence
  - Might need several iterations
  - Converges to closest local minimum

➡ Spend more time on fancy solvers...

Nonlinear Surface Deformation

- Nonlinear Optimization
- **Shell-Based Deformation**
- (Differential Coordinates)

Shell-Based Deformation

- **Discrete Shells**
  [Grinspun et al, SCA 2003]
- Rigid Cells
  [Botsch et al, SGP 2006]
- As-Rigid-As-Possible Modeling
  [Sorkine & Alexa, SGP 2007]

Discrete Shells

- Main idea
  - Don’t discretize continuous energy
  - Define **discrete** energy instead
  - Leads to simpler (still nonlinear) formulation
- Discrete energy
  - How to measure stretching on meshes?
  - How to measure bending on meshes?

Discrete Shell Energy

- **Stretching**: Change of edge lengths

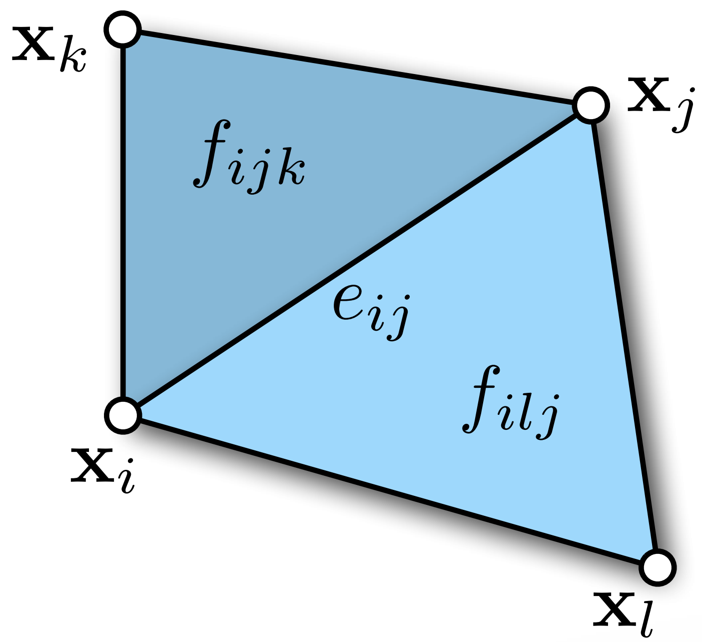
$$\sum_{e_{ij} \in E} \lambda_{ij} \left(|e_{ij}| - |\bar{e}_{ij}| \right)^2$$

- **Stretching**: Change of triangle areas

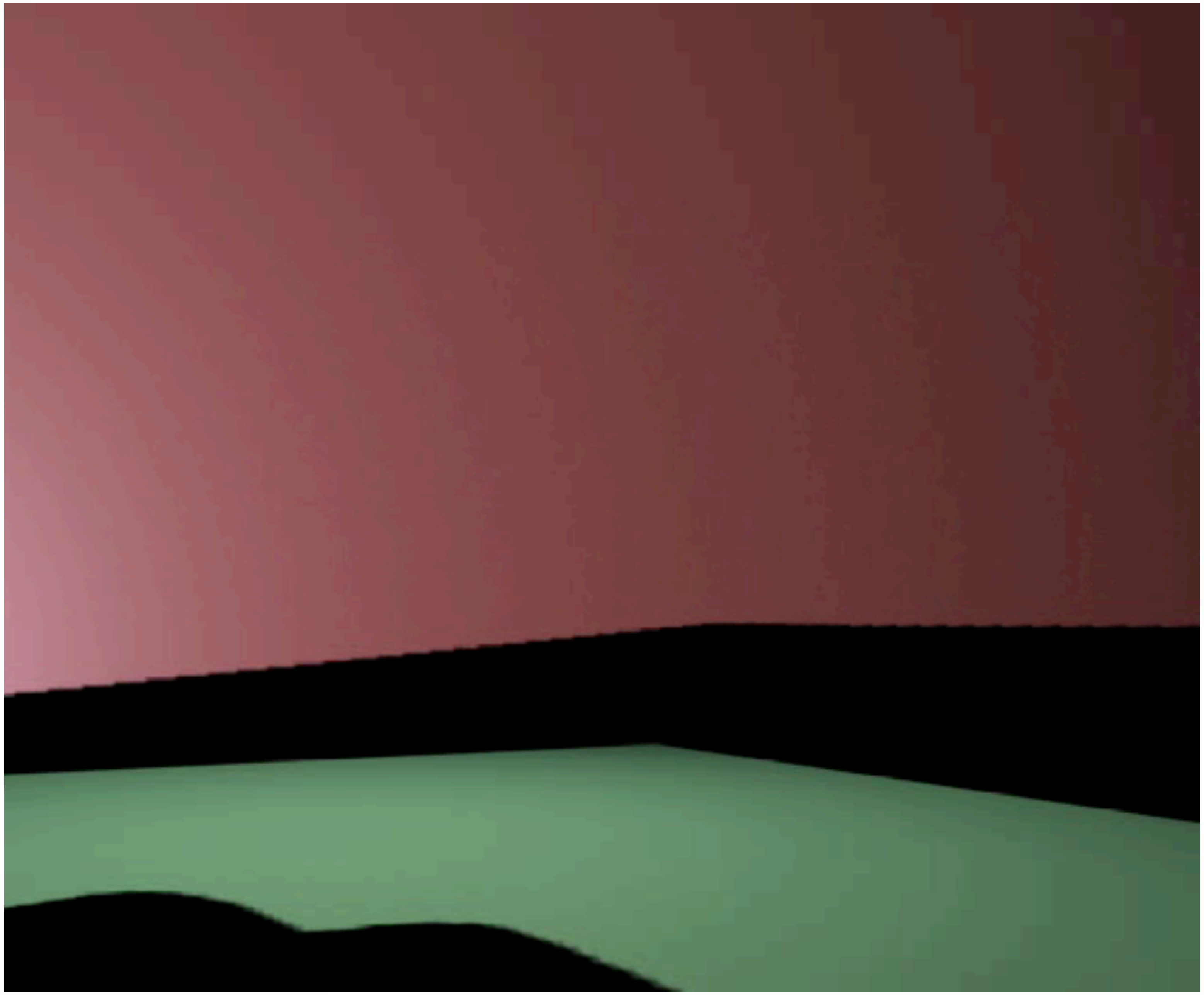
$$\sum_{f_{ijk} \in F} \lambda_{ijk} \left(|f_{ijk}| - |\bar{f}_{ijk}| \right)^2$$

- **Bending**: Change of dihedral angles

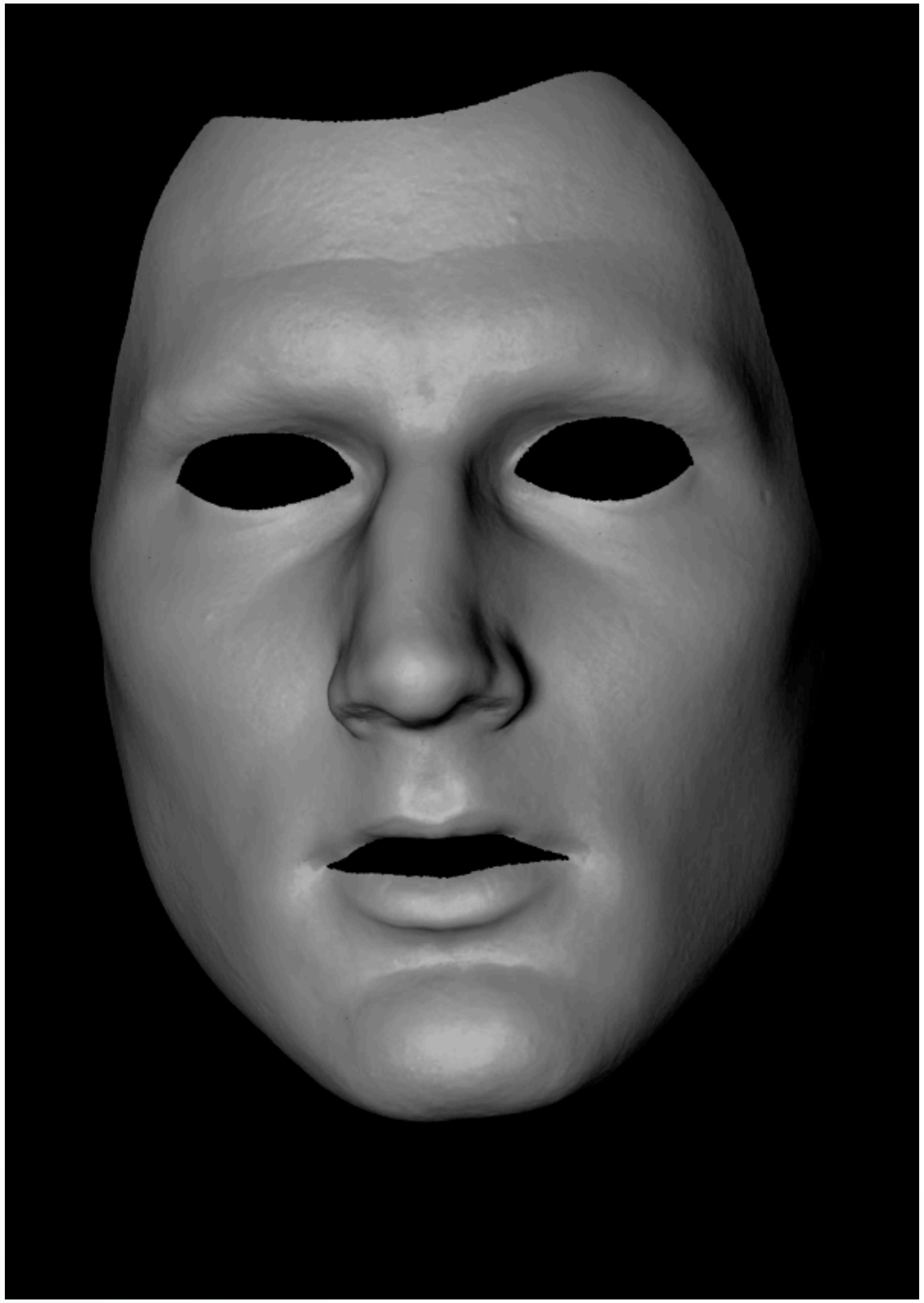
$$\sum_{e_{ij} \in E} \mu_{ij} \left(\theta_{ij} - \bar{\theta}_{ij} \right)^2$$



Discrete Shell Energy



Realistic Facial Animation



Linear model

Nonlinear model

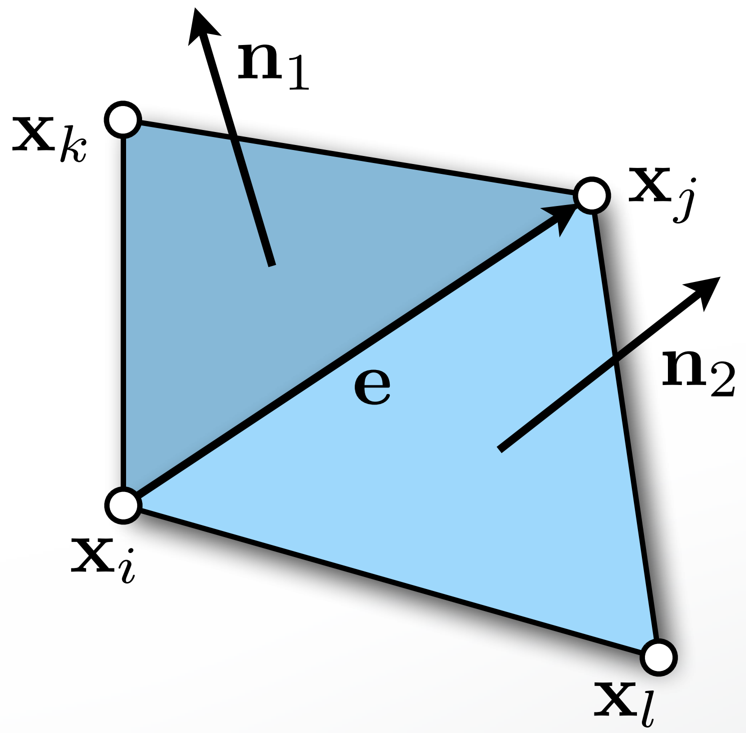
Discrete Energy Gradients

- Gradients of edge length

$$|e_{ij}| = \|\mathbf{x}_j - \mathbf{x}_i\|$$

$$\frac{\partial |e_{ij}|}{\partial \mathbf{x}_i} = \frac{-\mathbf{e}}{\|\mathbf{e}\|}$$

$$\frac{\partial |e_{ij}|}{\partial \mathbf{x}_j} = \frac{\mathbf{e}}{\|\mathbf{e}\|}$$



Discrete Energy Gradients

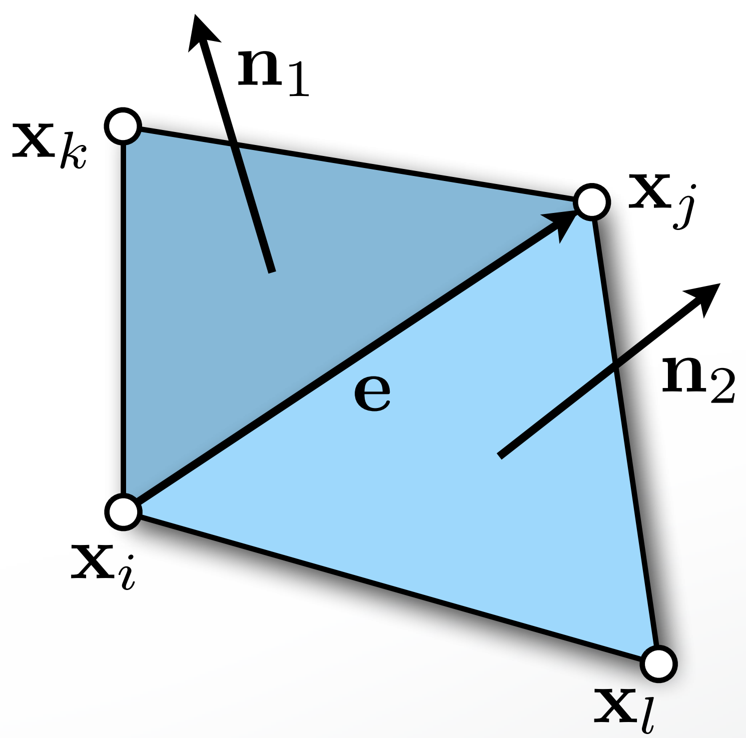
- Gradients of triangle area

$$|f_{ijk}| = \frac{1}{2}\|\mathbf{n}_1\|$$

$$\frac{\partial|f_{ijk}|}{\partial\mathbf{x}_i} = \frac{\mathbf{n}_1 \times (\mathbf{x}_k - \mathbf{x}_j)}{2\|\mathbf{n}_1\|}$$

$$\frac{\partial|f_{ijk}|}{\partial\mathbf{x}_j} = \frac{\mathbf{n}_1 \times (\mathbf{x}_i - \mathbf{x}_k)}{2\|\mathbf{n}_1\|}$$

$$\frac{\partial|f_{ijk}|}{\partial\mathbf{x}_k} = \frac{\mathbf{n}_1 \times (\mathbf{x}_j - \mathbf{x}_i)}{2\|\mathbf{n}_1\|}$$



Discrete Energy Gradients

- Gradients of dihedral angle

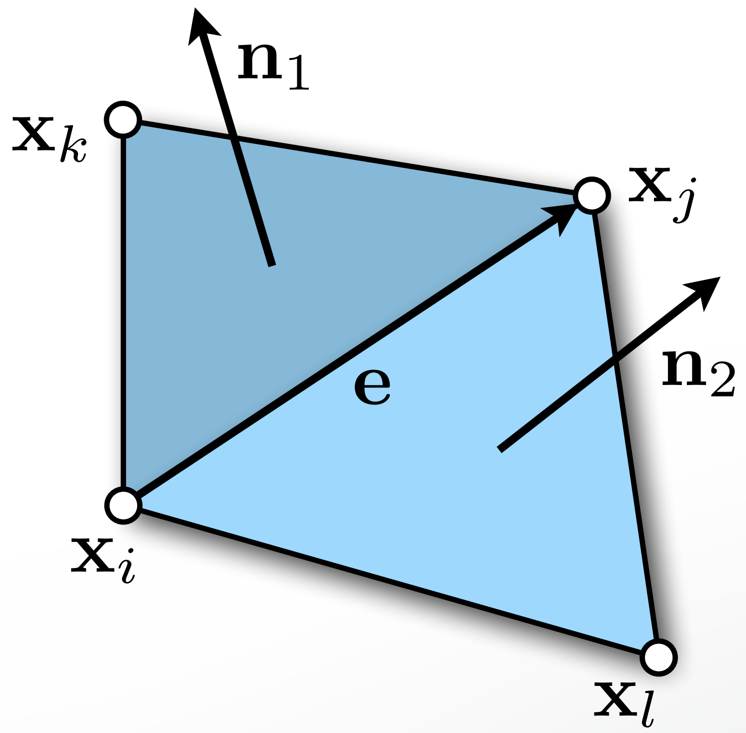
$$\theta = \operatorname{atan}\left(\frac{\sin\theta}{\cos\theta}\right) = \operatorname{atan}\left(\frac{(\mathbf{n}_1 \times \mathbf{n}_2)^T \mathbf{e}}{\mathbf{n}_1^T \mathbf{n}_2 \cdot \|\mathbf{e}\|}\right)$$

$$\frac{\partial\theta}{\partial\mathbf{x}_i} = \frac{(\mathbf{x}_k - \mathbf{x}_j)^T \mathbf{e}}{\|\mathbf{e}\|} \cdot \frac{-\mathbf{n}_1}{\|\mathbf{n}_1\|^2} + \frac{(\mathbf{x}_l - \mathbf{x}_j)^T \mathbf{e}}{\|\mathbf{e}\|} \cdot \frac{-\mathbf{n}_2}{\|\mathbf{n}_2\|^2}$$

$$\frac{\partial\theta}{\partial\mathbf{x}_j} = \frac{(\mathbf{x}_i - \mathbf{x}_k)^T \mathbf{e}}{\|\mathbf{e}\|} \cdot \frac{-\mathbf{n}_1}{\|\mathbf{n}_1\|^2} + \frac{(\mathbf{x}_i - \mathbf{x}_l)^T \mathbf{e}}{\|\mathbf{e}\|} \cdot \frac{-\mathbf{n}_2}{\|\mathbf{n}_2\|^2}$$

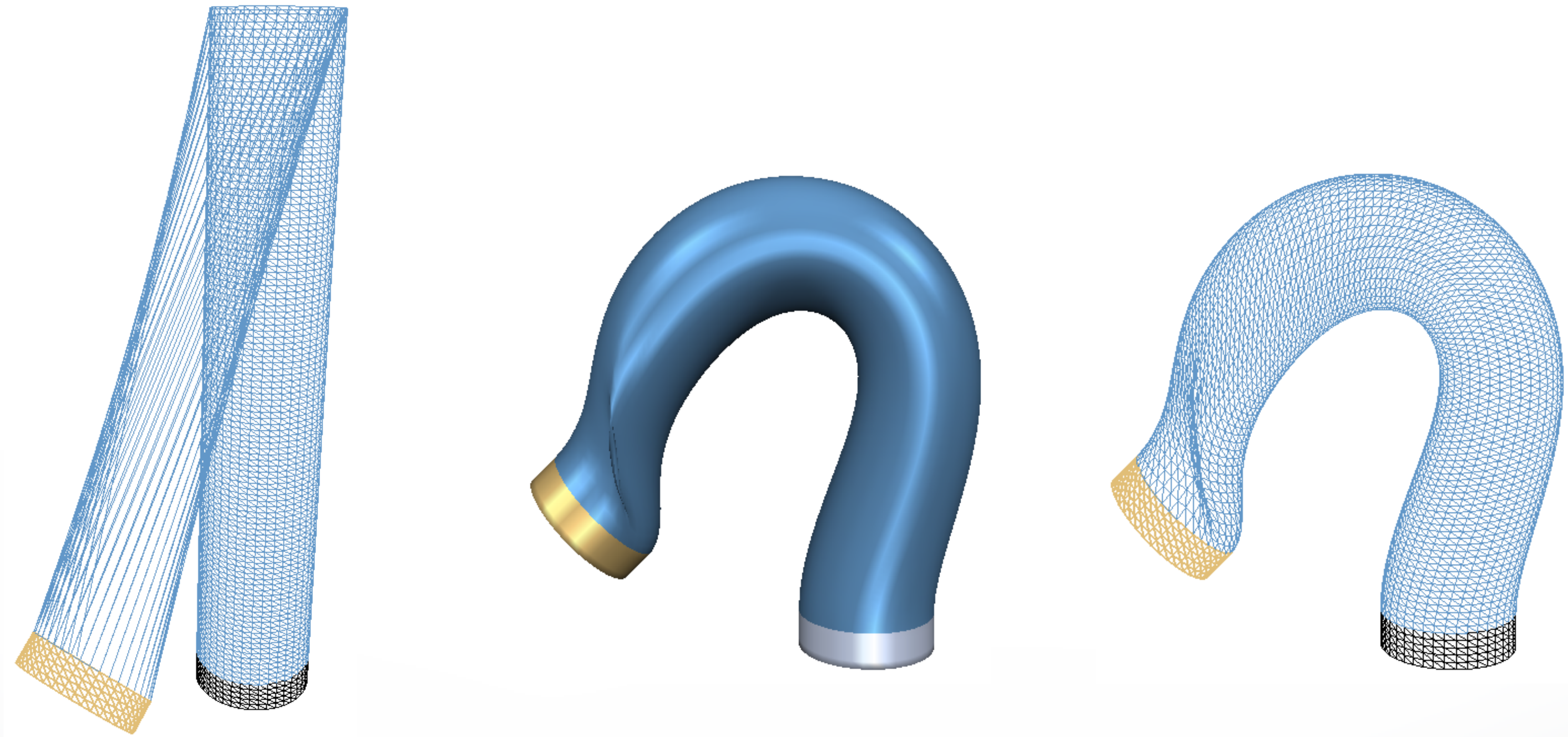
$$\frac{\partial\theta}{\partial\mathbf{x}_k} = \|\mathbf{e}\| \cdot \frac{-\mathbf{n}_1}{\|\mathbf{n}_1\|^2}$$

$$\frac{\partial\theta}{\partial\mathbf{x}_l} = \|\mathbf{e}\| \cdot \frac{-\mathbf{n}_2}{\|\mathbf{n}_2\|^2}$$



Discrete Shell Editing

- Problems with large deformation
  - Bad initial state causes numerical problems



Shell-Based Deformation

- Discrete Shells
  [Grinspun et al, SCA 2003]
- **Rigid Cells**
  [Botsch et al, SGP 2006]
- As-Rigid-As-Possible Modeling
  [Sorkine & Alexa, SGP 2007]

Nonlinear Shape Deformation

- *Nonlinear* editing too unstable?
- *<u>Physically plausible</u>* vs. physically correct

➡ Trade physical correctness for
  - Computational efficiency
  - Numerical robustness

Elastically Connected Rigid **Cells**

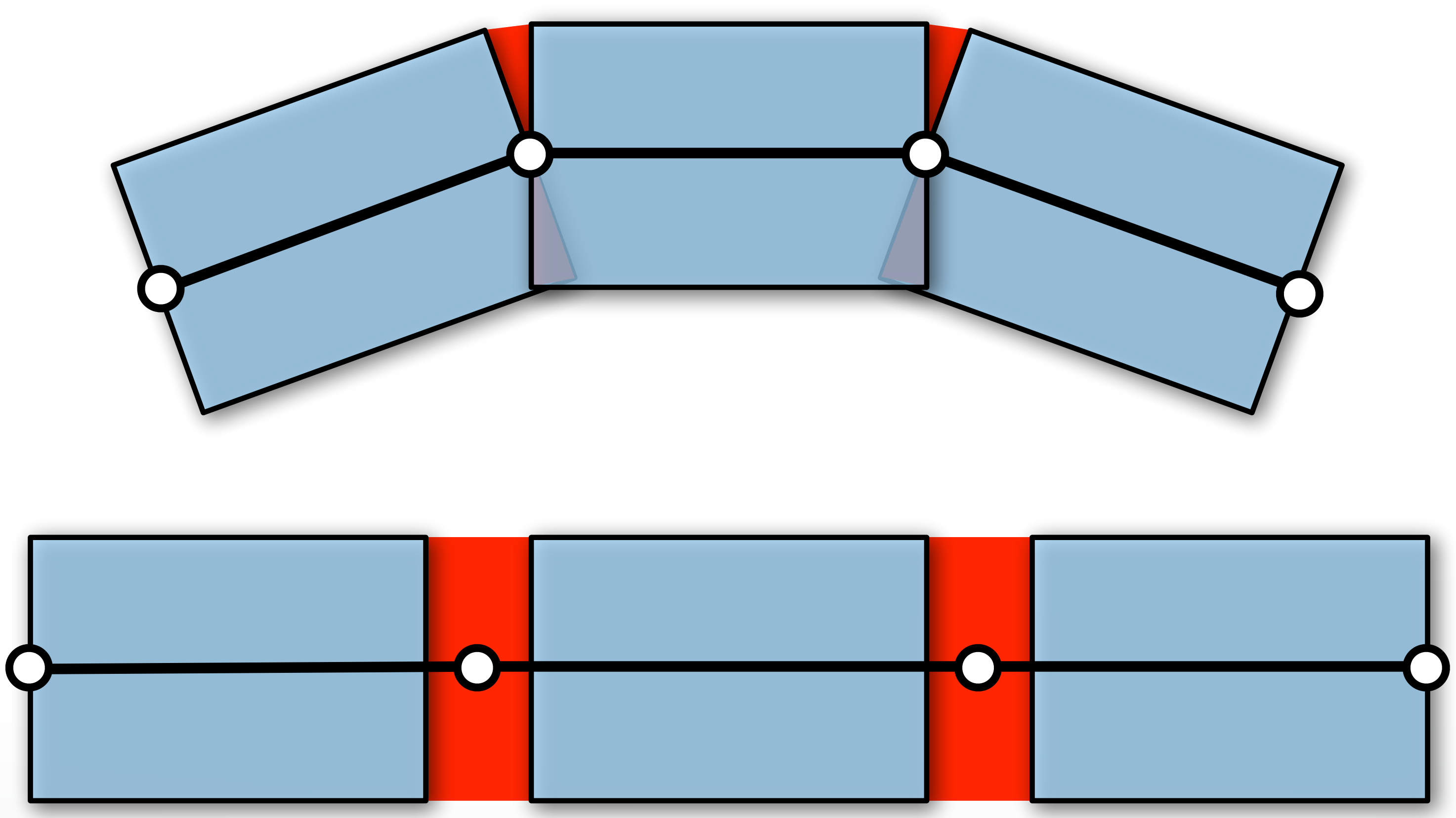
- Qualitatively emulate thin-shell behavior
- Thin volumetric layer around center surface
- Extrude polygonal cell C_i per mesh face

Elastically Connected **Rigid Cells**

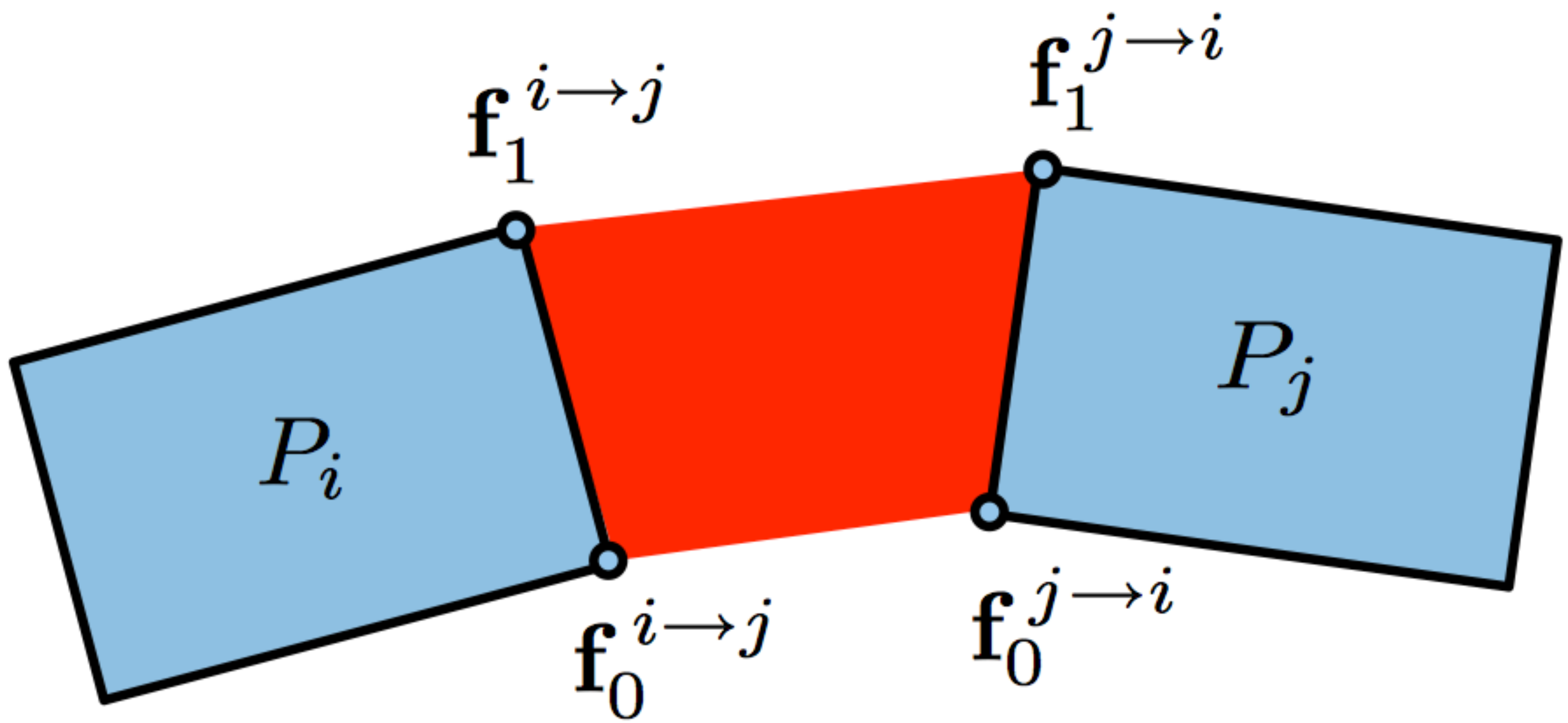
- Aim for robustness
  - Prevent cells from degenerating
  - ➡ Keep cells *rigid*

Elastically Connected Rigid Cells

- Connect cells along their faces
  - Nonlinear elastic energy
  - Measures bending, stretching, twisting, ...



Notion of Prism Elements



Nonlinear Minimization

- Find *rigid* motion $\mathbf{T}_i$ per cell C_i

$$\min_{\{\mathbf{T}_i\}} \sum_{\{i,j\}} w_{ij} \int_{[0,1]^2} \left\| \mathbf{T}_i\big(\mathbf{f}^{\,i\to j}(\mathbf{u})\big) \; - \; \mathbf{T}_j\big(\mathbf{f}^{\,j\to i}(\mathbf{u})\big) \right\|^2 \mathrm{d}\mathbf{u}$$

- Generalized global *shape matching* problem
  - Robust geometric optimization
  - Nonlinear Newton-type minimization
  - Hierarchical multi-grid solver

Newton-Type Iteration

1. Linearization of rigid motions

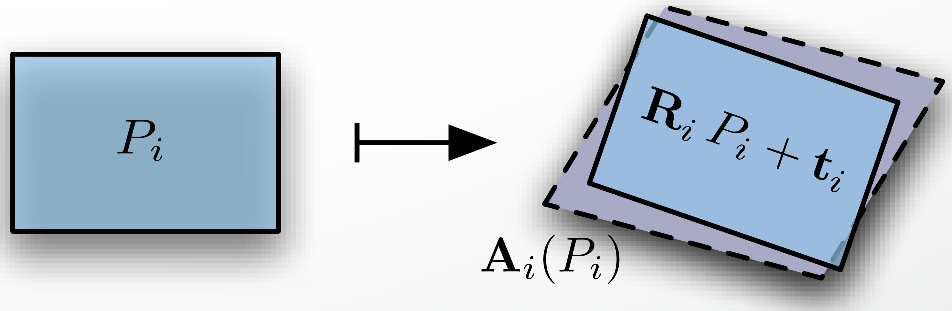
$$\mathbf{R}_i\,\mathbf{x} + \mathbf{t}_i \;\;\approx\;\; \mathbf{x} \;+\; (\omega_i \times \mathbf{x}) \;+\; \mathbf{v}_i \;\;=:\;\; \mathbf{A}_i\,\mathbf{x}$$

2. Quadratic optimization of velocities

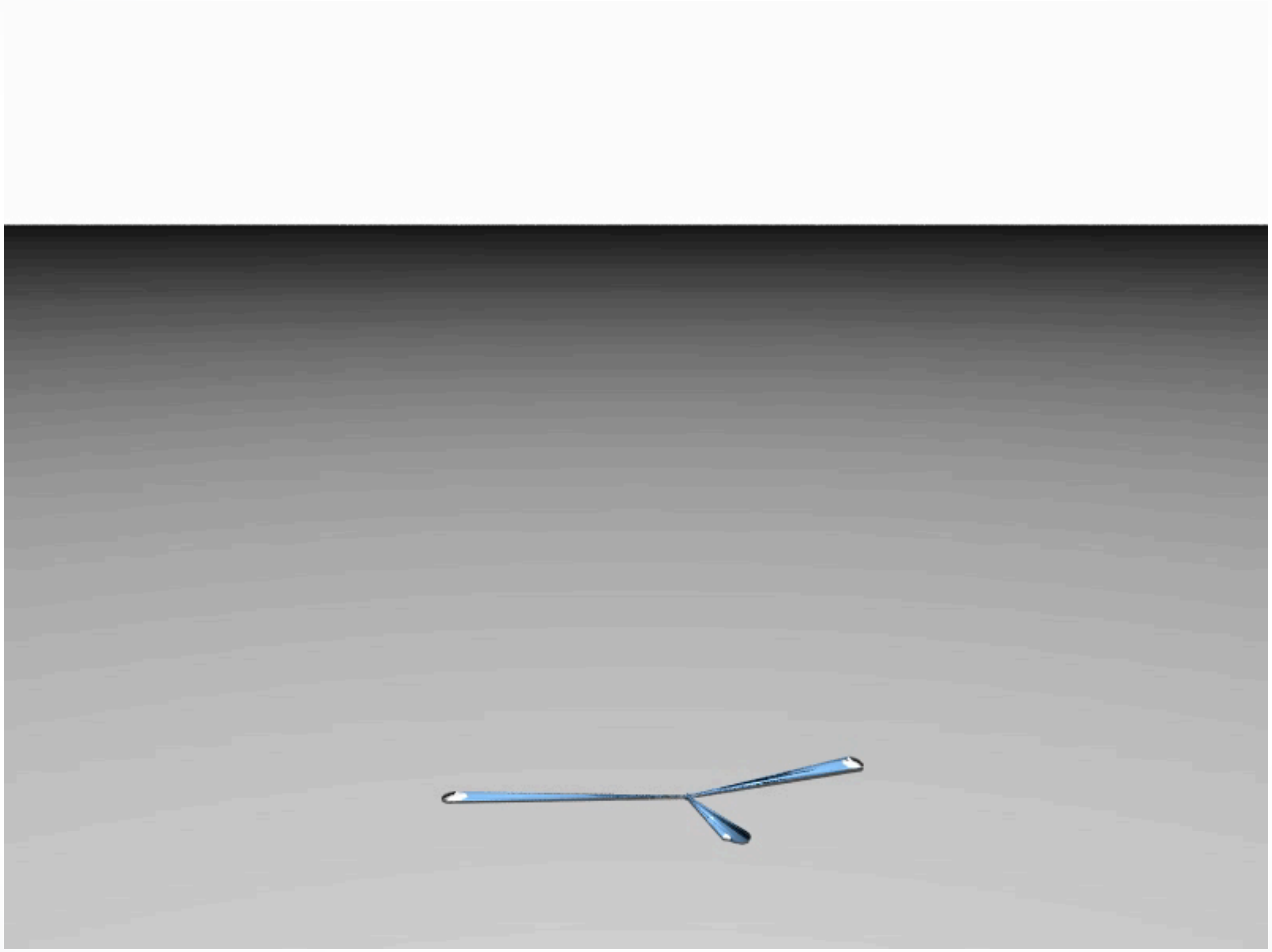
$$\min_{\{\mathbf{v}_i,\,\omega_i\}} \;\sum_{\{i,j\}} w_{ij} \int_{[0,1]^2} \left\| \mathbf{A}_i\big(\mathbf{f}^{\,i\to j}(\mathbf{u})\big) - \mathbf{A}_j\big(\mathbf{f}^{\,j\to i}(\mathbf{u})\big) \right\|^2 \mathrm{d}\mathbf{u}$$

3. Project $\mathbf{A}_i$ onto rigid motion manifold

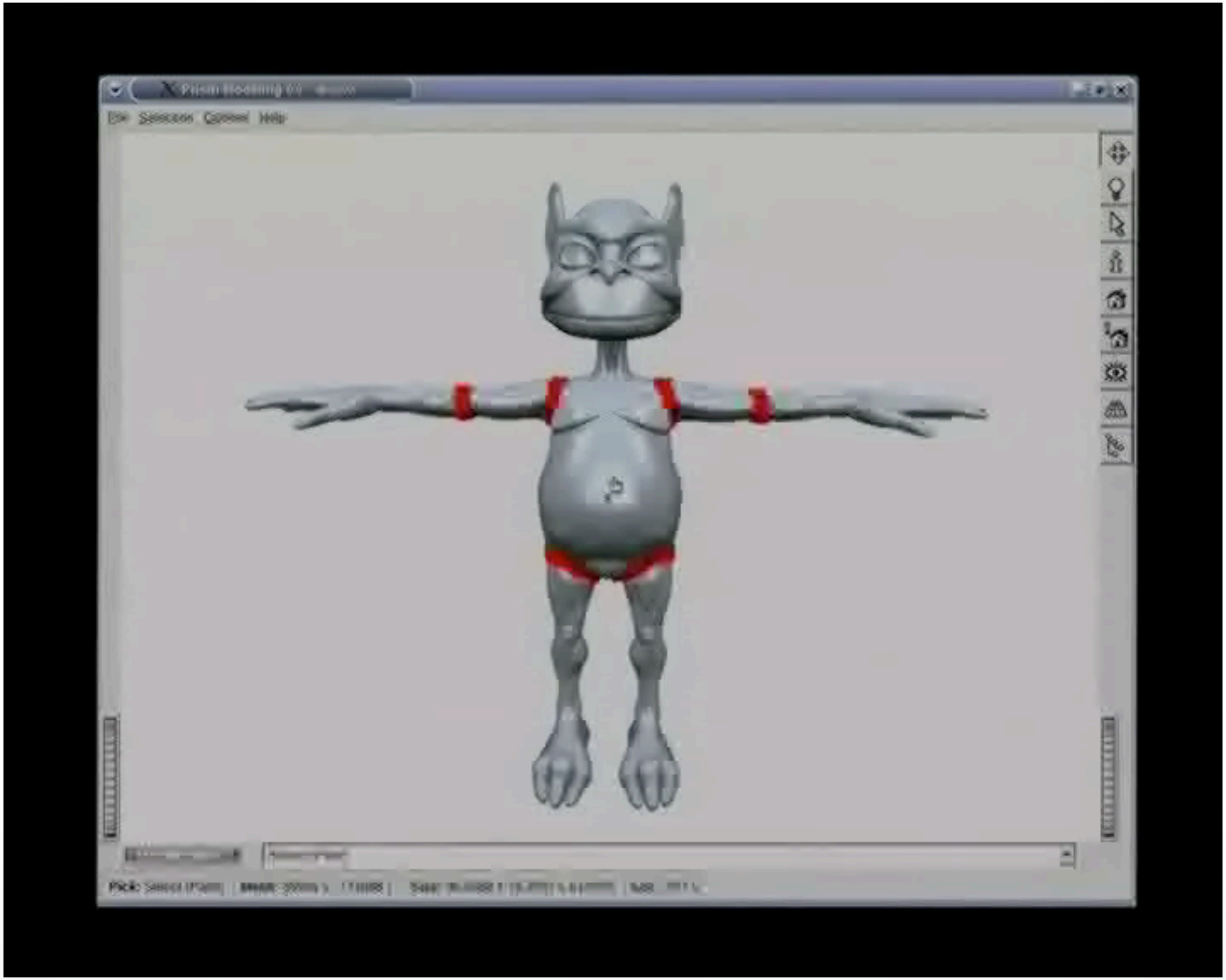
➡ Local shape matching



Robustness

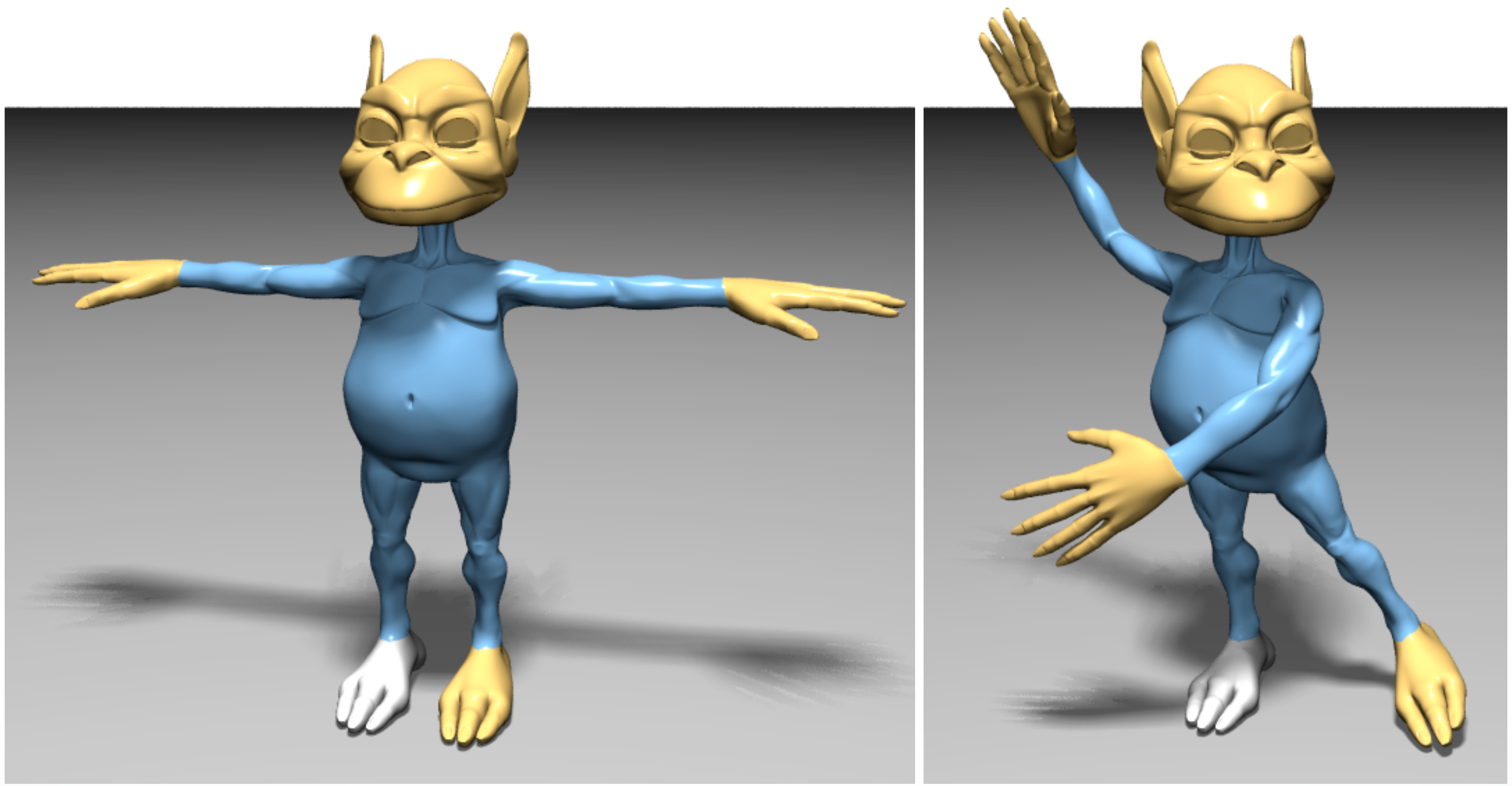


Character Posing



Goblin Posing

- Intuitive large scale deformations
- Whole session < 5 min

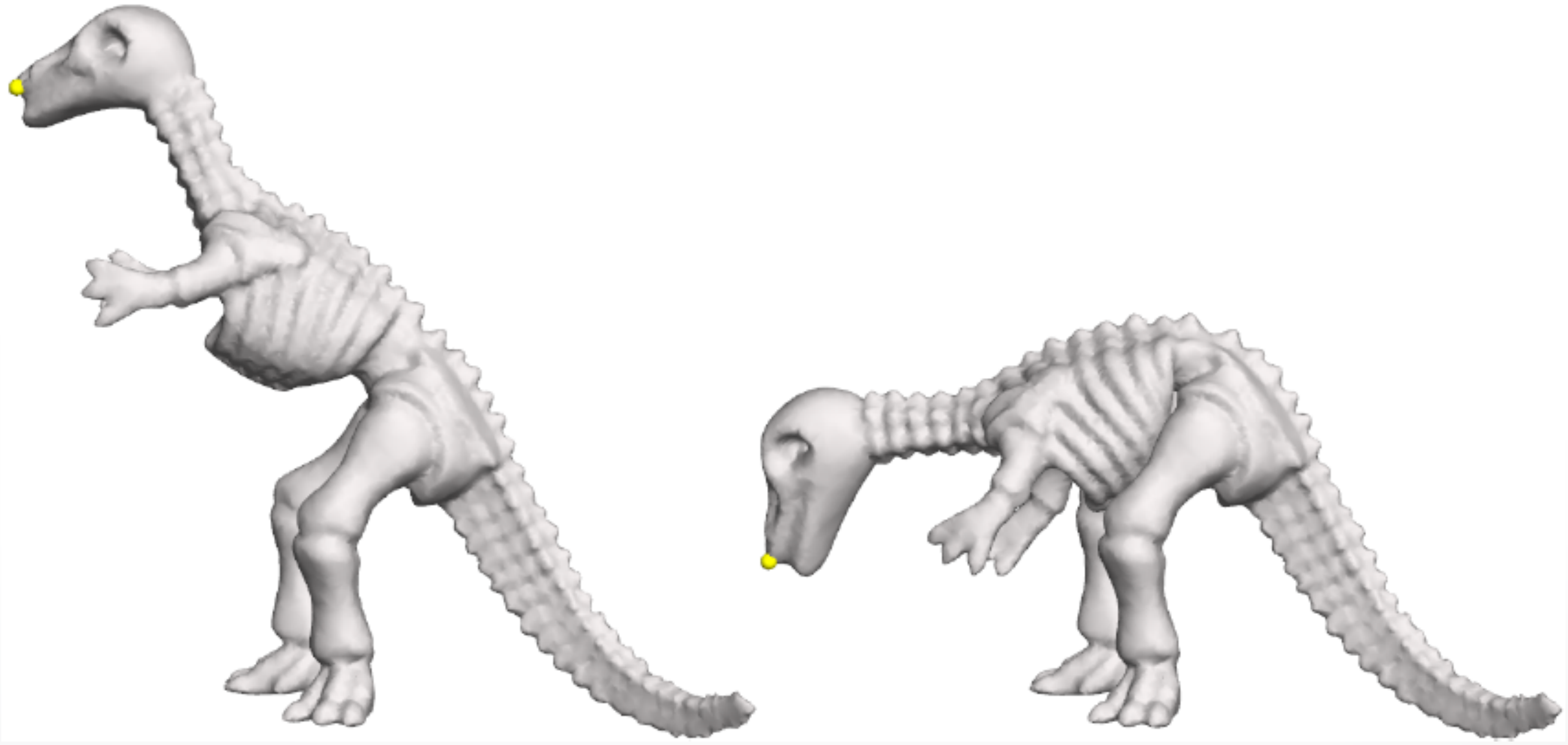


Shell-Based Deformation

- Discrete Shells
  [Grinspun et al, SCA 2003]
- Rigid Cells
  [Botsch et al, SGP 2006]
- **As-Rigid-As-Possible Modeling**
  [Sorkine & Alexa, SGP 2007]

Surface Deformation

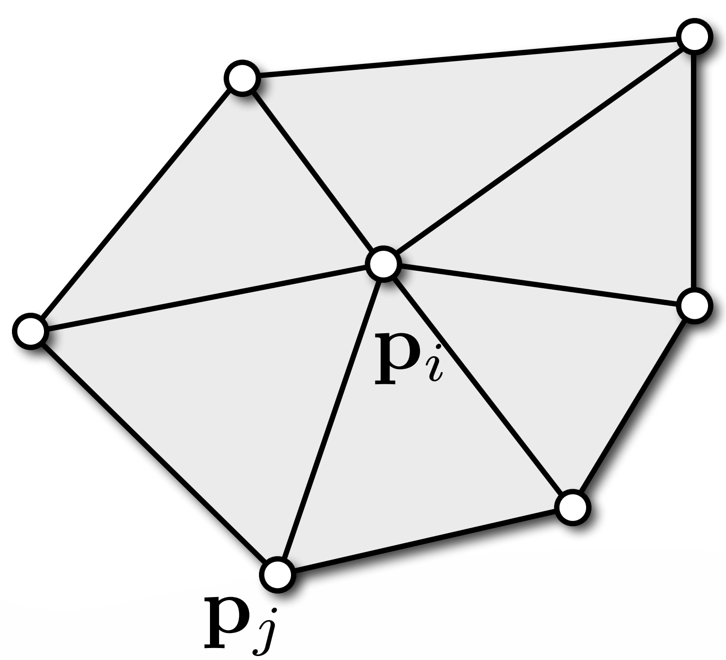
- Smooth large scale deformation
- Local as-rigid-as-possible behavior
  - Preserves small-scale details



Cell Deformation Energy

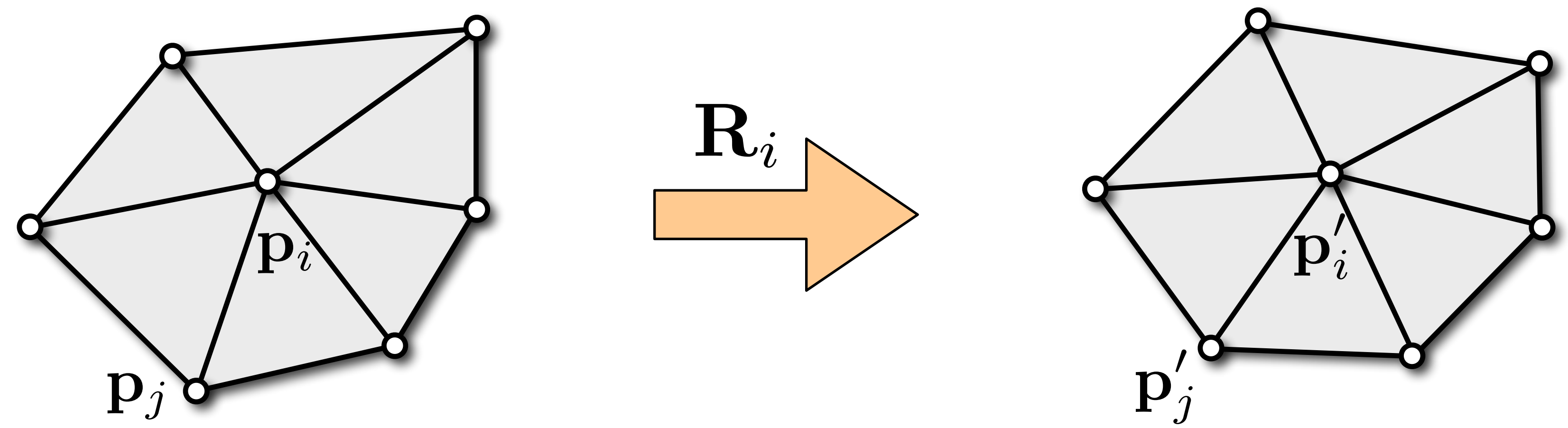
- Vertex neighborhoods should deform rigidly

$$\sum_{j \in N(i)} \left\| \left(\mathbf{p}'_j - \mathbf{p}'_i\right) - \mathbf{R}_i \left(\mathbf{p}_j - \mathbf{p}_i\right) \right\|^2 \rightarrow \min$$



Cell Deformation Energy

- If $\mathbf{p}$, $\mathbf{p}'$ are known then $\mathbf{R}_i$ is uniquely defined



- *Shape matching* problem
  - Build covariance matrix $\mathbf{S} = \mathbf{P}\mathbf{P}'^{\mathrm{T}}$
  - SVD: $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{W}^{\mathrm{T}}$
  - Extract rotation $\mathbf{R}_i = \mathbf{U}\mathbf{W}^{\mathrm{T}}$

Total Deformation Energy

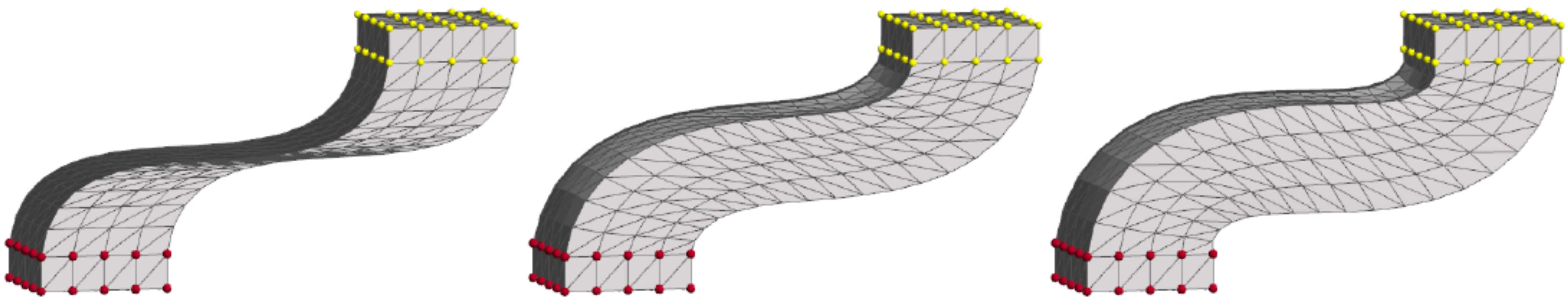
- Sum over all vertex

$$\min_{\mathbf{p}'} \sum_{i=1}^{n} \sum_{j \in N(i)} \left\| \left(\mathbf{p}'_j - \mathbf{p}'_i\right) - \mathbf{R}_i \left(\mathbf{p}_j - \mathbf{p}_i\right) \right\|^2$$

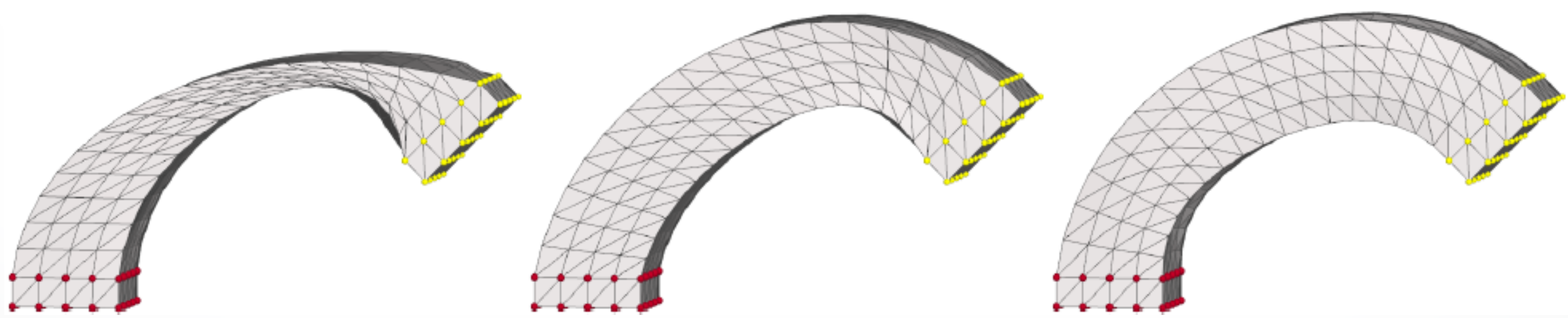
- Treat $\mathbf{p}'$ and $\mathbf{R}_i$ as separate variables
- Allows for alternating optimization
  - Fix $\mathbf{p}'$, find $\mathbf{R}_i$: Local shape matching per cell
  - Fix $\mathbf{R}_i$, find $\mathbf{p}'$: Solve Laplacian system

As-Rigid-As-Possible Modeling

- Start from naïve Laplacian editing as initial guess

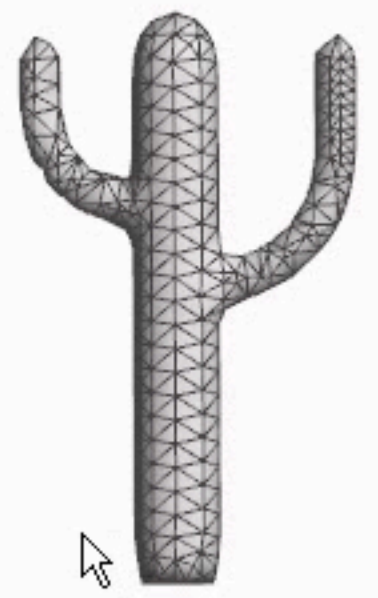


initial guess 1 iteration 2 iterations



initial guess 1 iterations 4 iterations

As-Rigid-As-Possible Modeling



Shell-Based Deformation

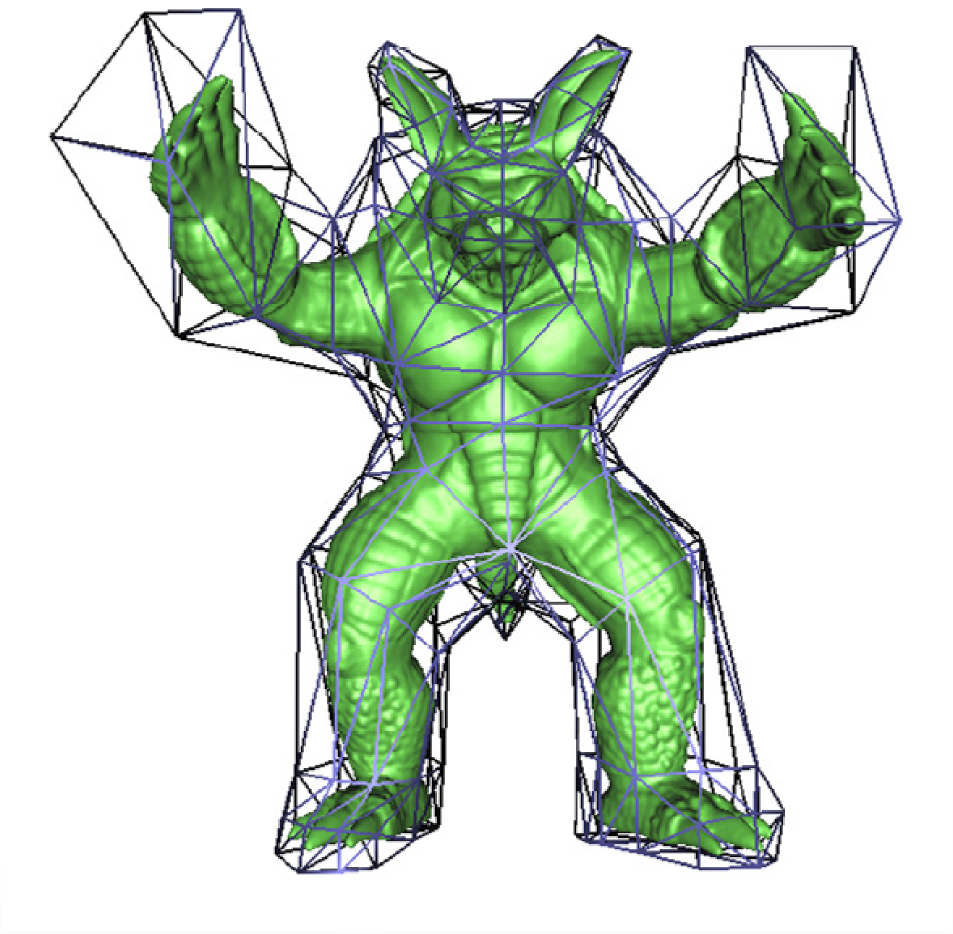
- Discrete Shells
  [Grinspun et al, SCA 2003]
- Rigid Cells
  [Botsch et al, SGP 2006]
- As-Rigid-As-Possible Modeling
  [Sorkine & Alexa, SGP 2007]

Nonlinear Surface Deformation

- Limitations of Linear Methods
- Shell-Based Deformation
- **(Differential Coordinates)**

Subspace Gradient Deformation

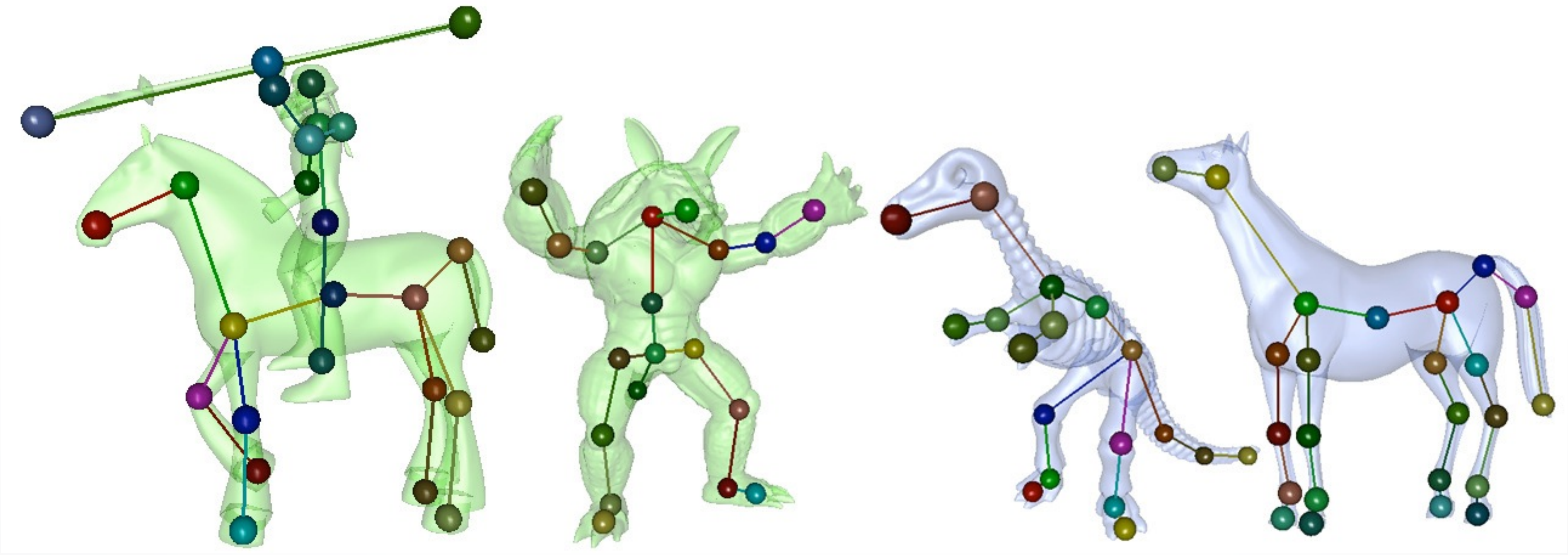
- Nonlinear Laplacian coordinates
- Least squares solution on coarse cage subspace



[Huang et al, SIGGRAPH 06]

Mesh Puppetry

- Skeletons and Laplacian coordinates
- Cascading optimization

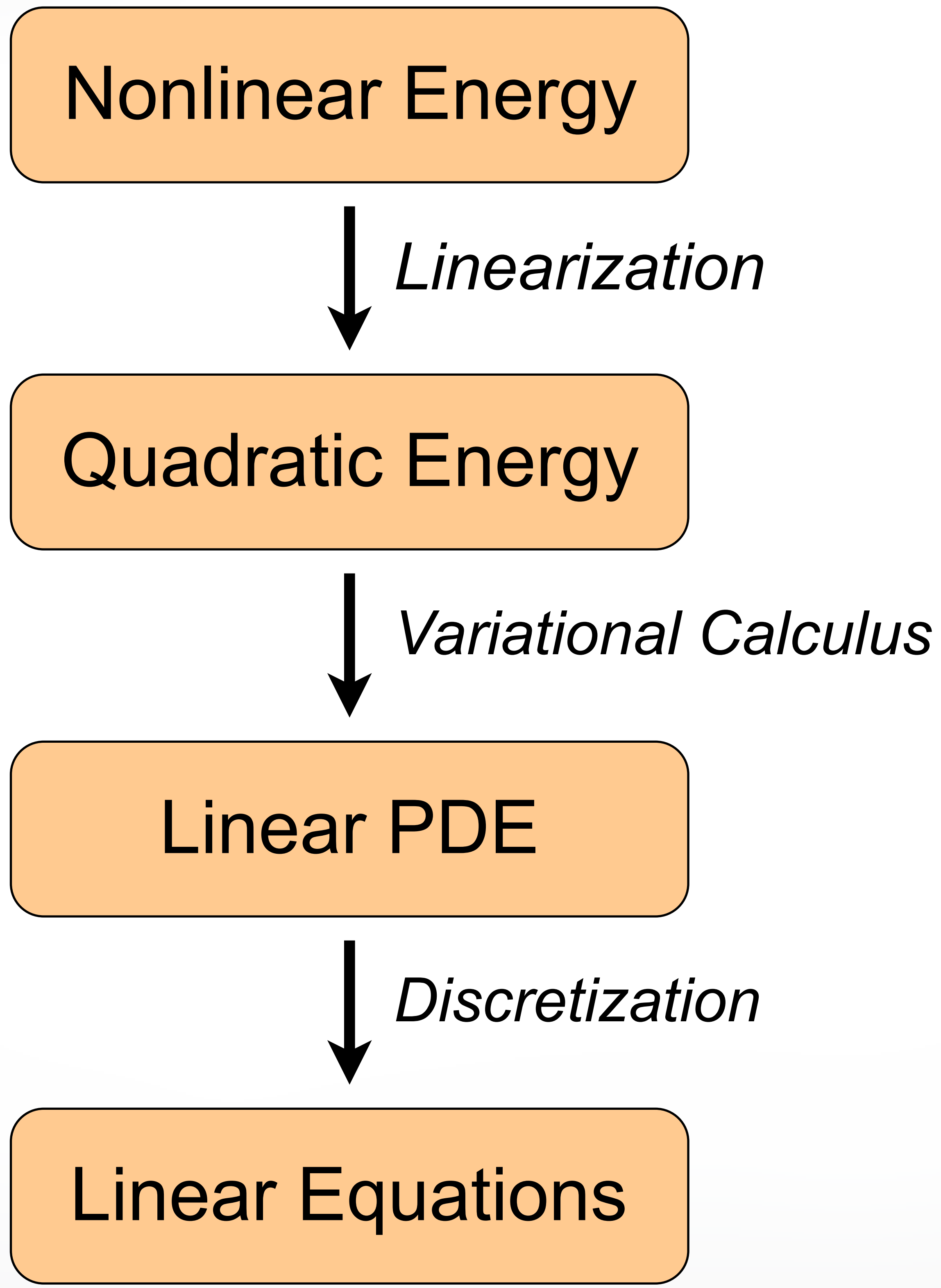


[Shi et al, SIGGRAPH 07]

Nonlinear Surface Deformation

- Limitations of Linear Methods
- Shell-Based Deformation
- (Differential Coordinates)

Linear Approaches



Linear Approaches

- Resulting linear systems
  - Shell-based $\Delta^2 \mathbf{d} = \mathbf{0}$
  - Gradient-based $\Delta \mathbf{p} = \nabla \cdot \mathbf{T}(\mathbf{g})$
  - Laplacian-based $\Delta^2 \mathbf{p} = \Delta \mathbf{T}(\mathbf{l})$
- Properties
  - Highly sparse
  - Symmetric, positive definite (*SPD*)
  - Solve for new RHS each frame!

Linear SPD Solvers

- **Dense Cholesky factorization**
  - Cubic complexity
  - High memory consumption (doesn’t exploit sparsity)
- **Iterative conjugate gradients**
  - Quadratic complexity
  - Need sophisticated preconditioning
- **Multigrid solvers**
  - Linear complexity
  - But rather complicated to develop (and to use)
- **Sparse Cholesky factorization**
  - Linear complexity
  - Easy to use

Dense Cholesky Factorization

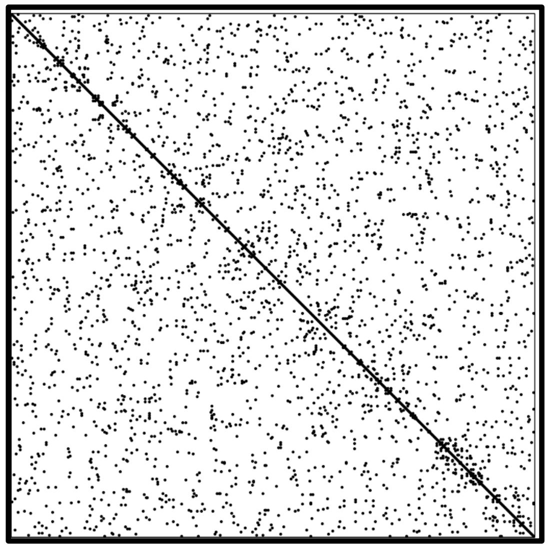
Solve $\mathbf{A}\mathbf{x} = \mathbf{b}$

1. Cholesky factorization $\mathbf{A} = \mathbf{L}\mathbf{L}^T$
2. Solve system $\mathbf{y} = \mathbf{L}^{-1}\mathbf{b}, \quad \mathbf{x} = \mathbf{L}^{-T}\mathbf{y}$

Dense Cholesky Factorization

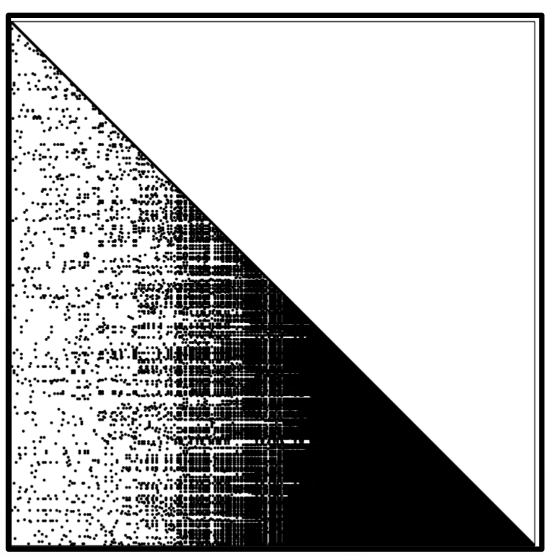
A=LL^T

500×500 matrix
3500 non-zeros



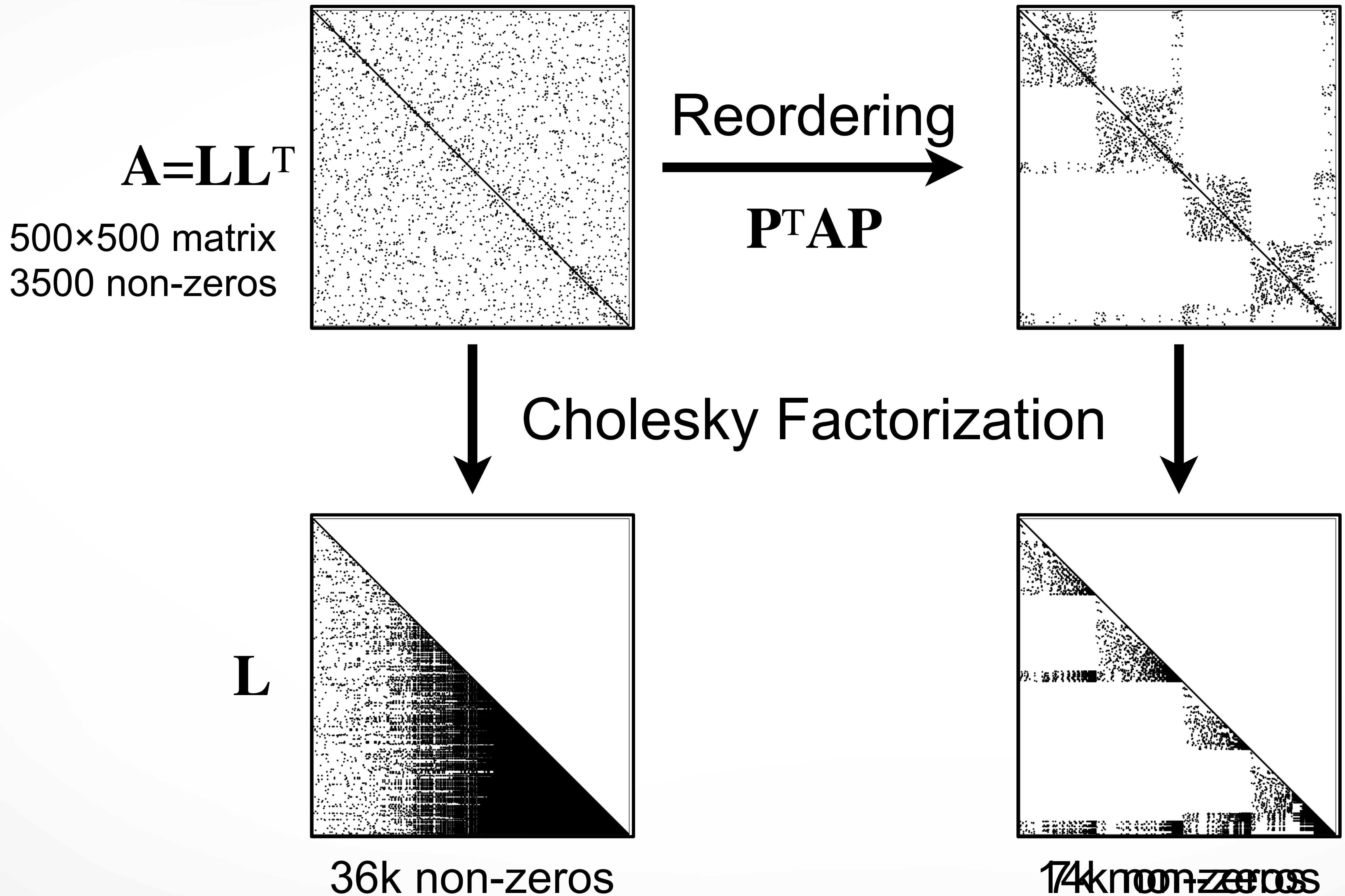
Cholesky Factorization

L



36k non-zeros

Sparse Cholesky Factorization



Sparse Cholesky Factorization

Solve $\mathbf{A}\mathbf{x} = \mathbf{b}$

Pre-computation

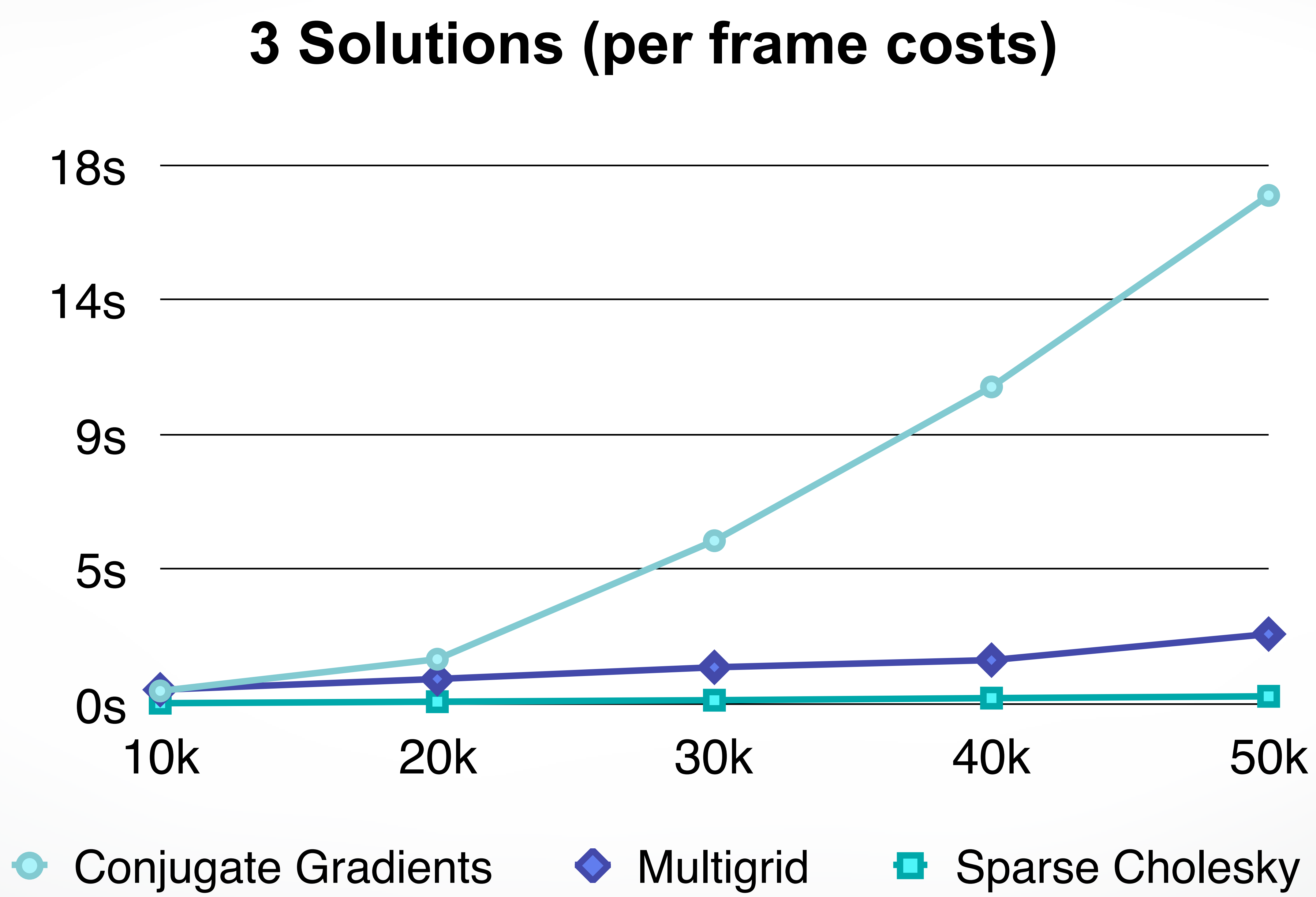
1. Matrix re-ordering $\tilde{\mathbf{A}} = \mathbf{P}^T\mathbf{A}\mathbf{P}$

2. Cholesky factorization $\tilde{\mathbf{A}} = \mathbf{L}\mathbf{L}^T$

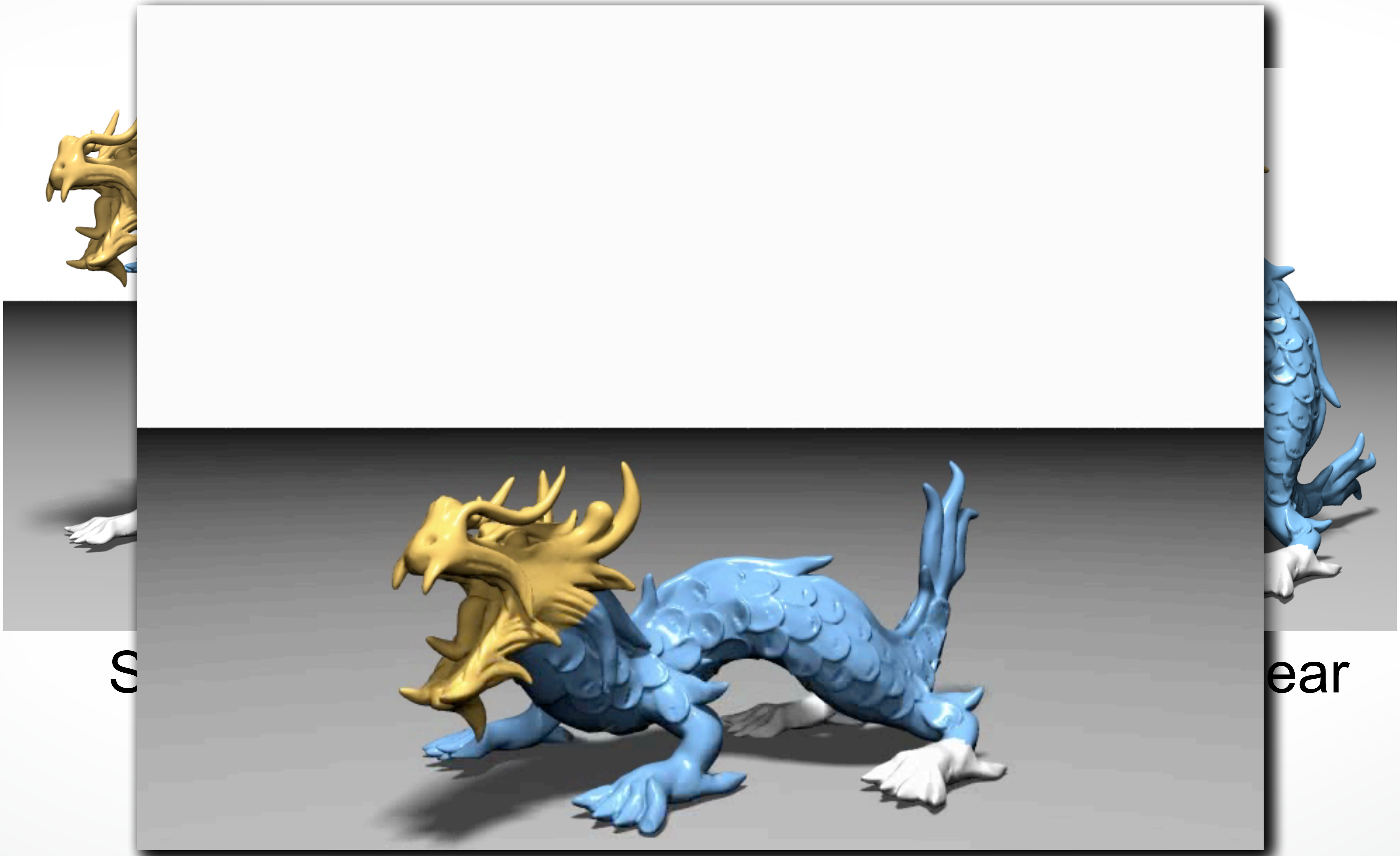
3. Solve system $\mathbf{y} = \mathbf{L}^{-1}\mathbf{P}^T\mathbf{b}, \quad \mathbf{x} = \mathbf{P}\mathbf{L}^{-T}\mathbf{y}$

Per-frame computation

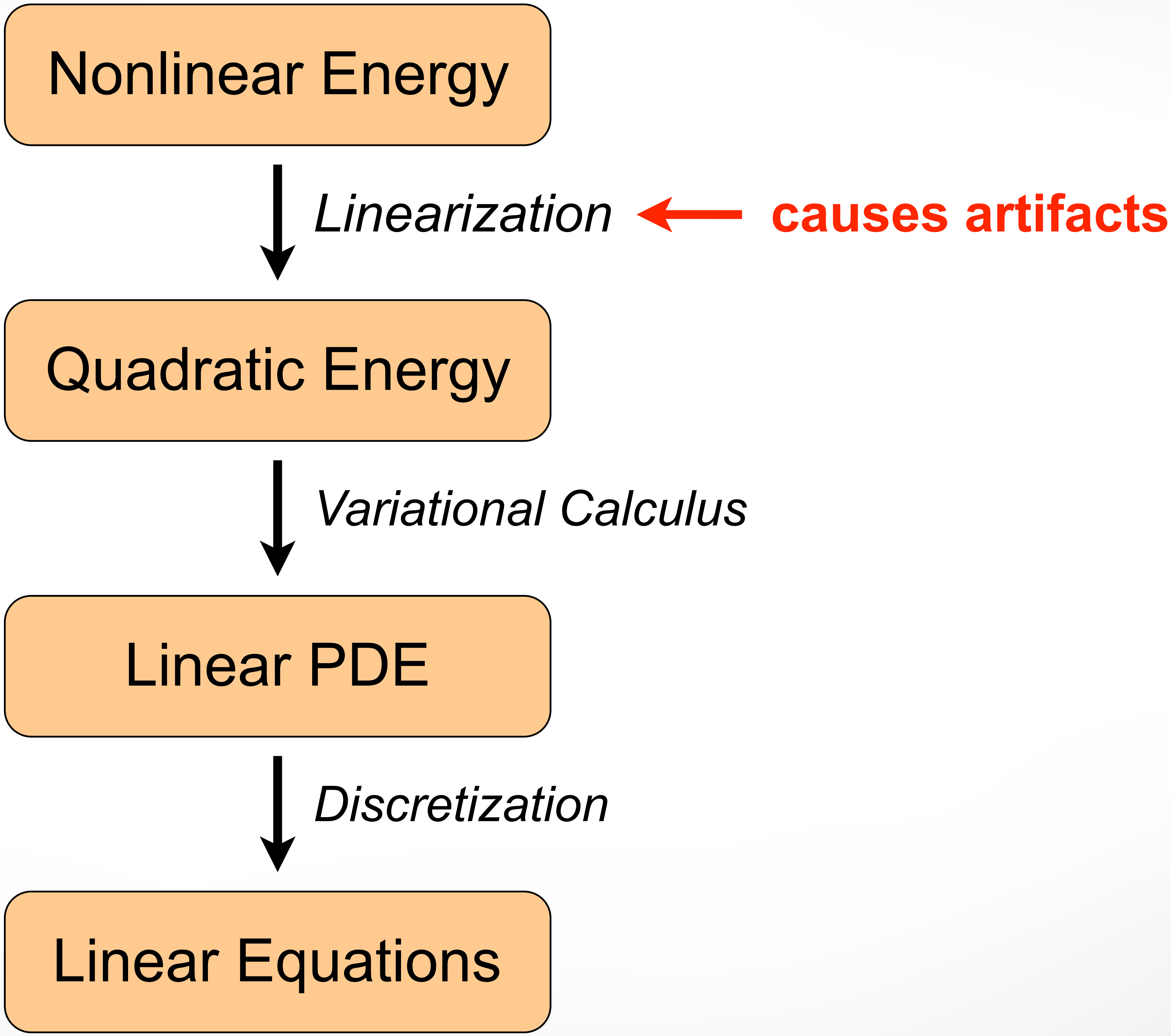
Bi-Laplace System



Linear vs. Non-Linear



Linear Approaches



Linearizations / Simplifications

- **Shell-based deformation**

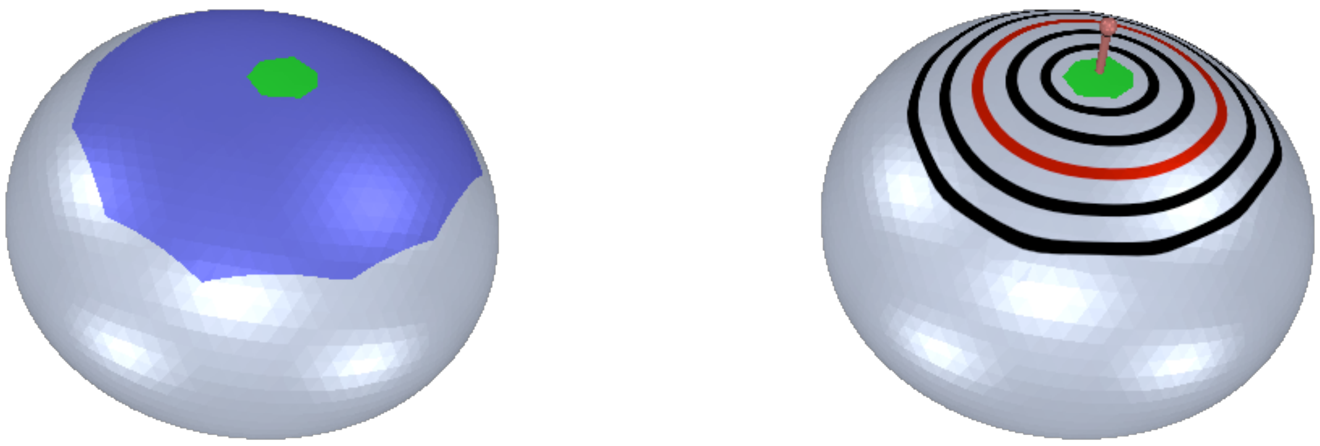
$$\int_{\Omega} k_s \left\| \mathbf{I} - \mathbf{I}' \right\|^2 + k_b \left\| \mathbf{II} - \mathbf{II}' \right\|^2 \ \mathrm{d}u\mathrm{d}v$$

$$\Downarrow$$

$$\int_{\Omega} k_s \Big(\|\mathbf{d}_u\|^2 + \|\mathbf{d}_v\|^2 \Big) \ + \ k_b \Big(\|\mathbf{d}_{uu}\|^2 + 2\,\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2 \Big) \,\mathrm{d}u\mathrm{d}v$$

- **Gradient-based editing**

$$\nabla \mathbf{T}(\mathbf{x}) = \mathbf{A}$$



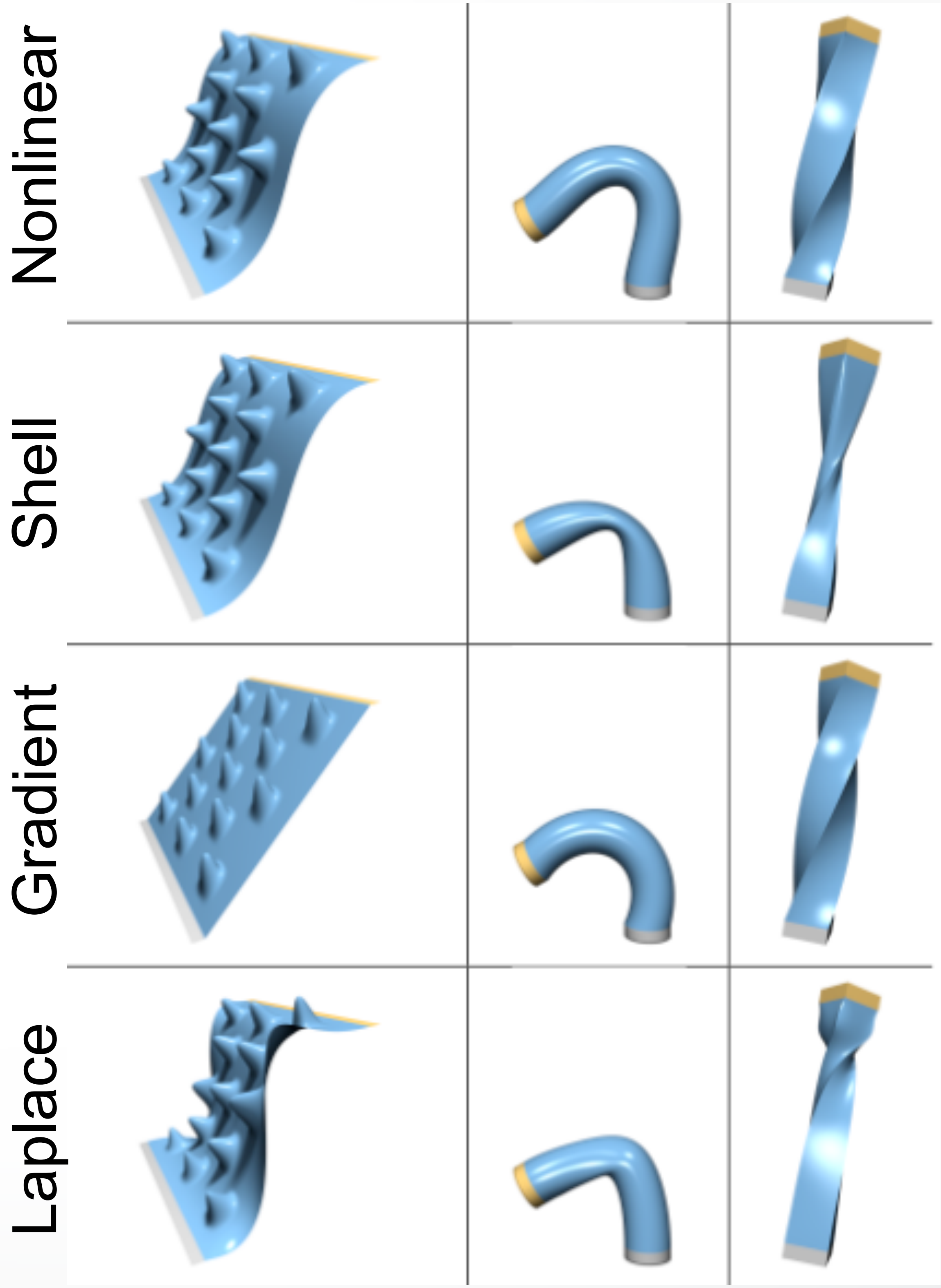
- **Laplacian surface editing**

$$\mathbf{Rx} \;\approx\; \mathbf{x} + (\mathbf{r} \times \mathbf{x}) \;=\; \begin{pmatrix} 1 & -r_3 & r_2 \\ r_3 & 1 & -r_1 \\ -r_2 & r_1 & 1 \end{pmatrix} \mathbf{x}$$

$$\mathbf{T}_i \;=\; \begin{pmatrix} s & -r_3 & r_2 \\ r_3 & s & -r_1 \\ -r_2 & r_1 & s \end{pmatrix}$$

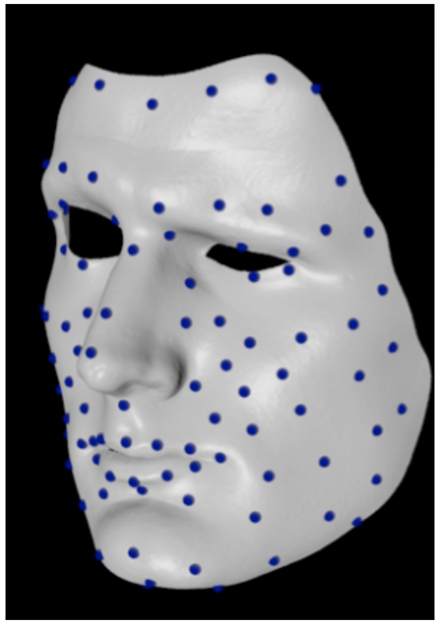
Linear vs. Non-Linear

- Analyze existing methods
  - Some work for translations
  - Some work for rotations
  - No method works for both



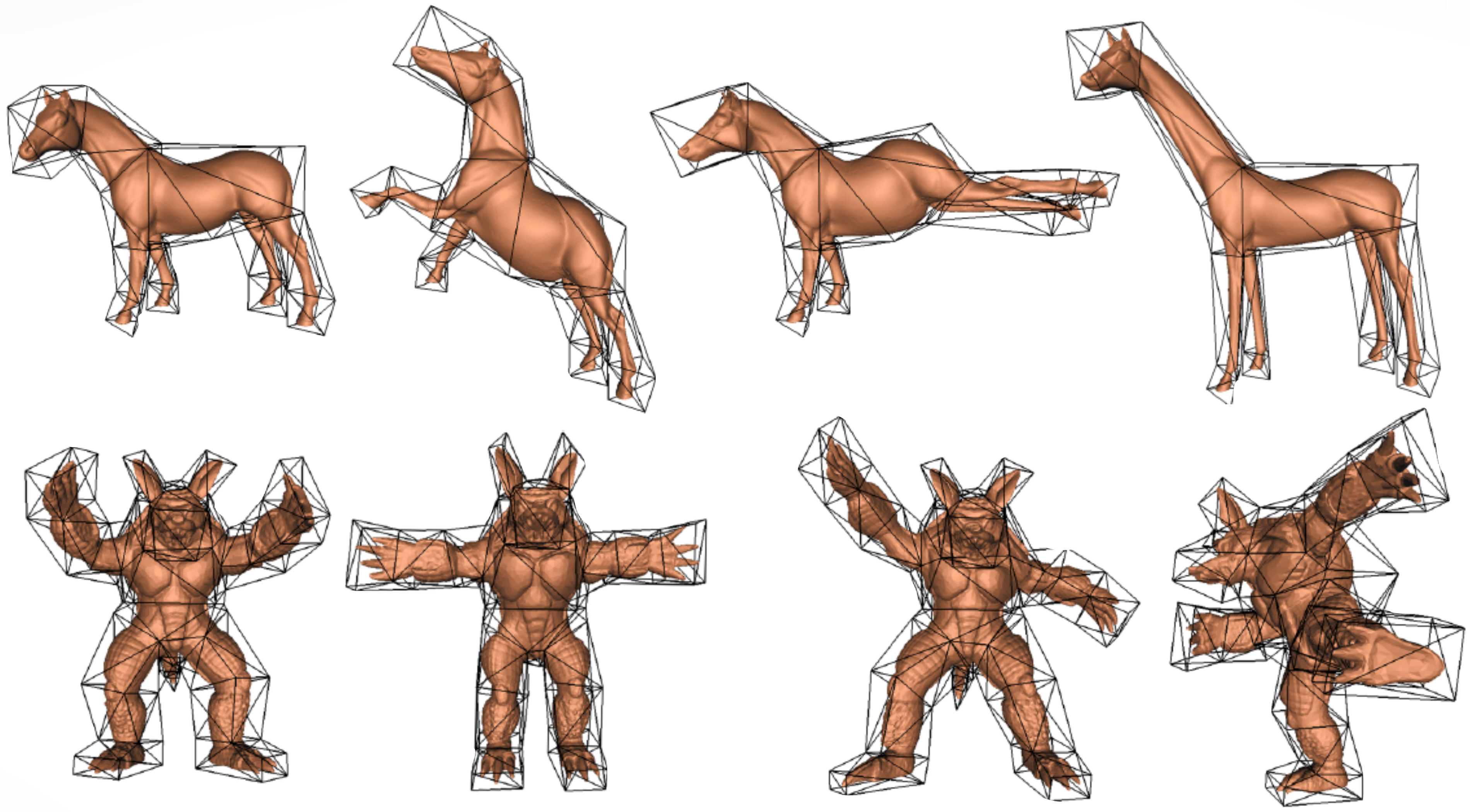
Linear vs. Non-Linear

- Linear approaches
  - Solve linear system each frame
  - Small deformations
  - Dense constraints



- Nonlinear approaches
  - Solve nonlinear problem each frame
  - Large deformations
  - Sparse constraints

Next Time



Spatial Deformation

Thanks!