Digital Geometry
-- Surface Deformations

Junjie Cao @ DLUT

Spring 2018

http://jjcao.github.io/DigitalGeometry/

Pleasure may come from illusion, but happiness can come only of reality.

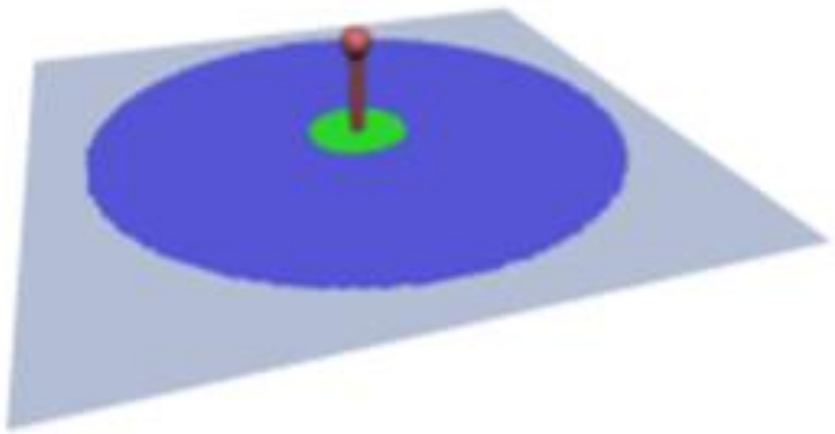
Overview

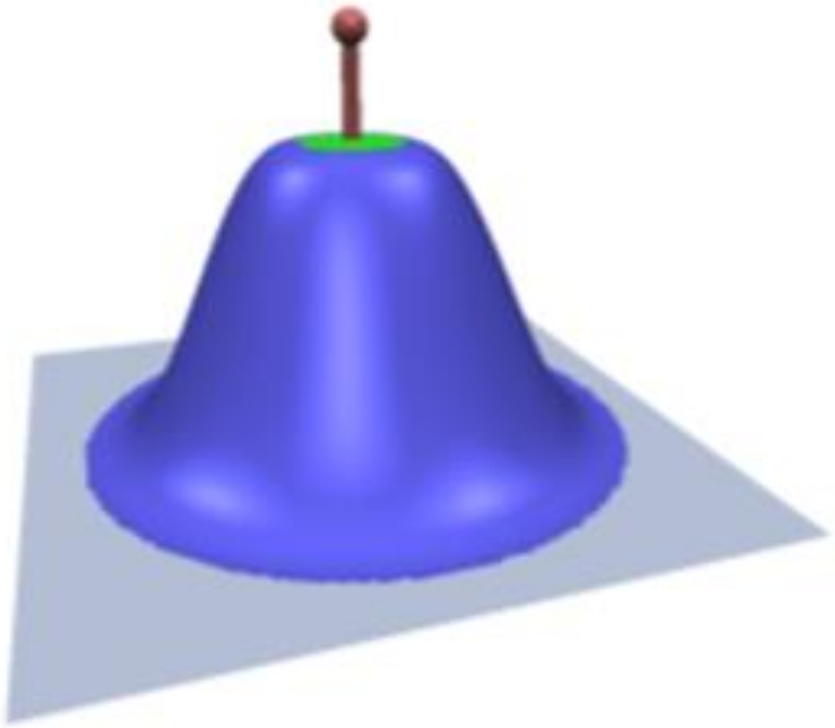
- **Surface-based deformation**
  - **Energy minimization**
  - Multiresolution editing
  - Differential coordinates

Physically-Based Deformation

Modeling Metaphor

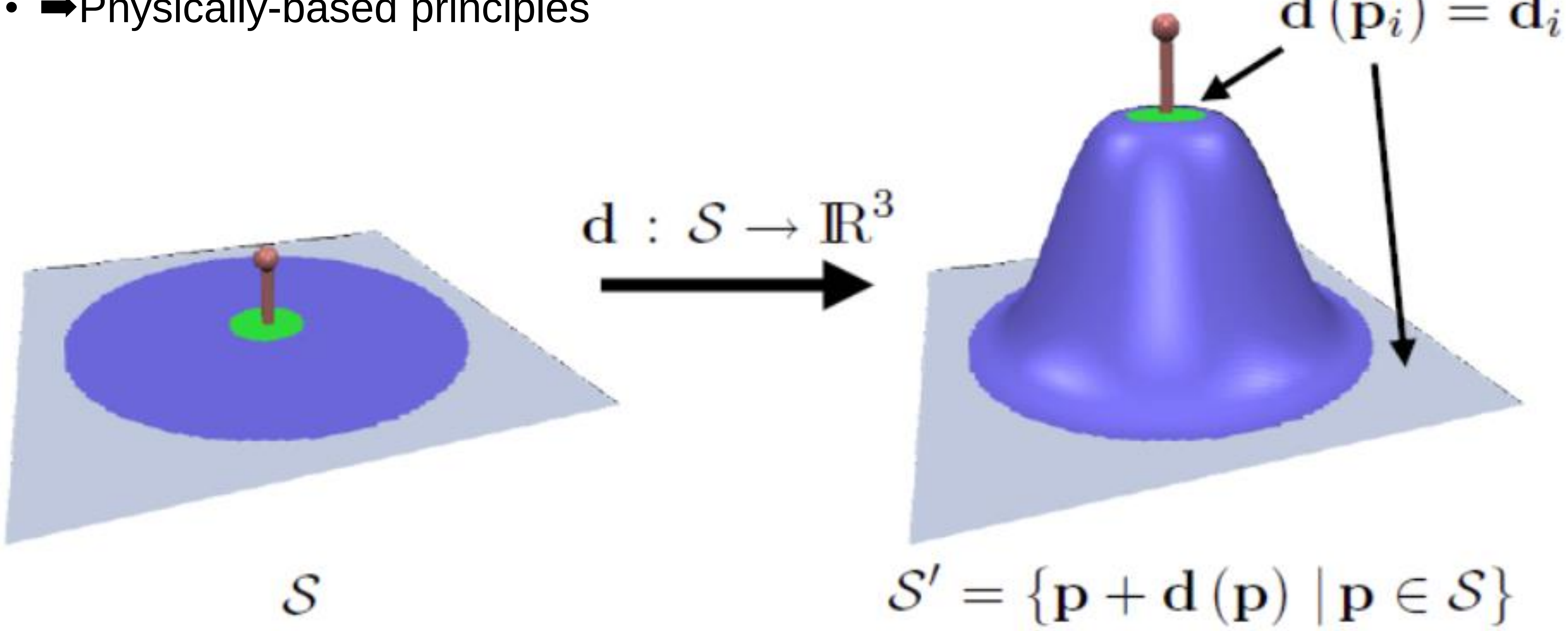
- Paint three surface regions
  - Support region (blue)
  - Fixed vertices (gray)
  - Handle regions (green)





Modeling Notation

- Mesh deformation by displacement function **d**
  - Interpolate prescribed constraints
  - Smooth, intuitive deformation
  - ➡Physically-based principles



Shell Deformation Energy

- **Stretching**
  - Change of local distances
  - Captured by 1st fundamental form
- **Bending**
  - Change of local curvature
  - Captured by 2nd fundamental form
- **Stretching & bending is sufficient**
  - Differential geometry: "1st and 2nd fundamental forms determine a surface up to rigid motion."

$$\boxed{\int_\Omega k_s \left\| \mathbf{I} - \bar{\mathbf{I}} \right\|^2}$$

$$\mathbf{I} = \begin{bmatrix} \mathbf{x}_u^T \mathbf{x}_u & \mathbf{x}_u^T \mathbf{x}_v \\ \mathbf{x}_v^T \mathbf{x}_u & \mathbf{x}_v^T \mathbf{x}_v \end{bmatrix}$$

$$\boxed{\int_\Omega k_b \left\| \mathbf{II} - \bar{\mathbf{II}} \right\|^2}$$

$$\mathbf{II} = \begin{bmatrix} \mathbf{x}_{uu}^T \mathbf{n} & \mathbf{x}_{uv}^T \mathbf{n} \\ \mathbf{x}_{vu}^T \mathbf{n} & \mathbf{x}_{vv}^T \mathbf{n} \end{bmatrix}$$

Energy Models [Botsch & Kobbelt, SIGGRAPH 04]

- Nonlinear stretching & bending energies

$$\int_\Omega k_s \underbrace{\boxed{\left\| \mathbf{I} - \mathbf{I}' \right\|^2}}_{\text{stretching}} + k_b \underbrace{\boxed{\left\| \mathbf{II} - \mathbf{II}' \right\|^2}}_{\text{bending}} \, \mathrm{d}u\mathrm{d}v$$

- Linearize terms → Quadratic energy

$$\int_\Omega k_s \underbrace{\boxed{\left(\|\mathbf{d}_u\|^2 + \|\mathbf{d}_v\|^2 \right)}}_{\text{stretching}} + k_b \underbrace{\boxed{\left(\|\mathbf{d}_{uu}\|^2 + 2\,\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2 \right)}}_{\text{bending}} \mathrm{d}u\mathrm{d}v$$

Energy Models [Botsch & Kobbelt, SIGGRAPH 04]

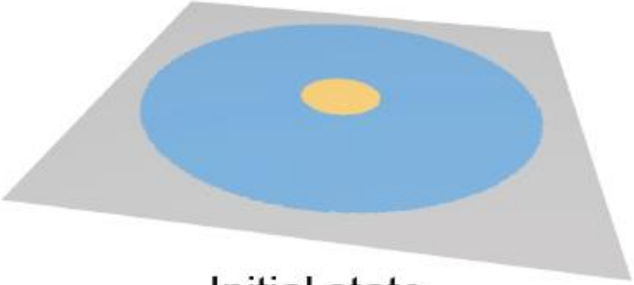
- Minimize linearized bending energy:

$$E(\mathbf{d}) \;=\; \int_{\mathcal{S}} \|\mathbf{d}_{uu}\|^2 + 2\,\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2 \,\mathrm{d}\mathcal{S} \qquad \boxed{f(x) \to \min}$$

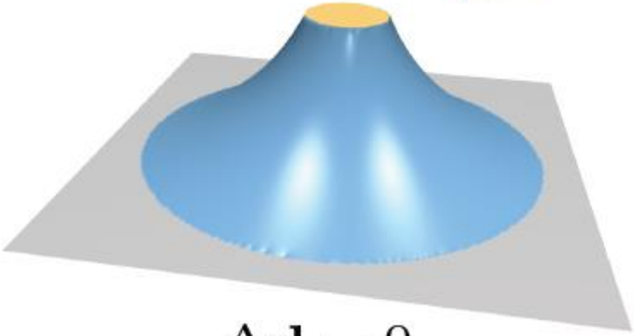
- Variational calculus → Euler-Lagrange PDE

$$\Delta^2 \mathbf{d} \;:=\; \mathbf{d}_{uuuu} + 2\mathbf{d}_{uuvv} + \mathbf{d}_{vvvv} \;=\; 0 \qquad \boxed{f'(x) = 0}$$

- "Best" deformation that satisfies constraints

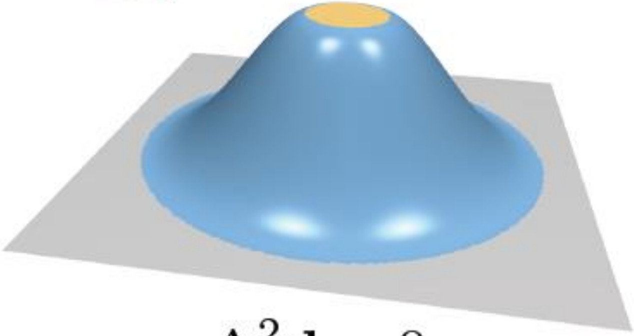


Initial state



$\Delta \mathbf{d} = 0$

(Membrane)

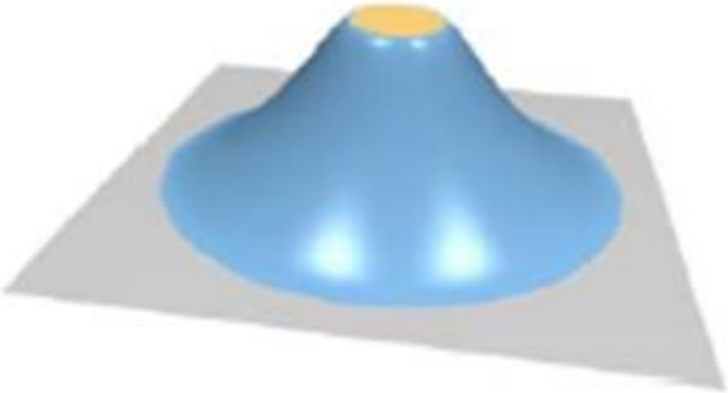


$\Delta^2 \mathbf{d} = 0$

(Thin plate)

$\Delta \mathbf{p} = 0$

$\Delta^2 \mathbf{p} = 0$



pure stretching with ks = 1, kb = 0
pure bending with ks = 0, kb = 1
a weighted combination with ks = 1, kb = 10

K = 1 & 2 is not enough

Minimizing E_k => C_{k-1} blending

membrane surface (k = 1), thin-plate surface (k = 2), minimal curvature variation (k = 3).

[Botsch & Kobbelt, SIGGRAPH 04]

$$E_k(d) = \int_{\Omega} \left\|\nabla^k d\right\|^2 du dv$$

$$\int_{\mathbb{R}^3} \|\mathbf{d}_{xxx}\|^2 + \|\mathbf{d}_{xyy}\|^2 + \ldots + \|\mathbf{d}_{zzz}\|^2 \; dx\,dy\,dz$$

$$\min E_k(d(x)) \Rightarrow \Delta^k d(x) = 0, x \in \Omega \backslash \delta\Omega$$

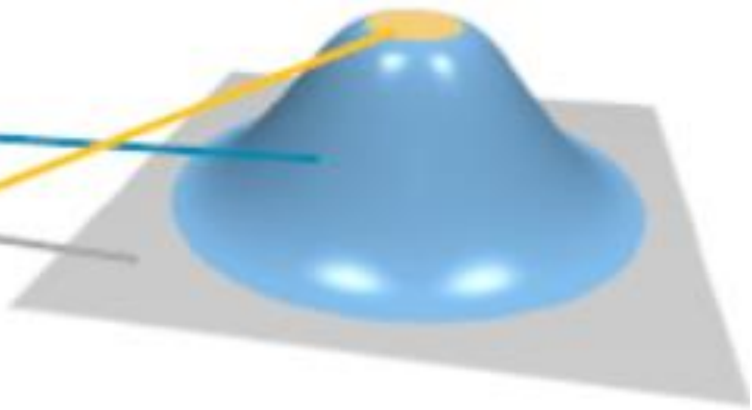
s.t. $\Delta^j d(x) = b_j(x), x \in \delta\Omega, j < k$

$$\Rightarrow \left(\begin{array}{c|c} \multicolumn{2}{c}{\bar{\Delta}^k} \\ \hline 0 & I_{F+H} \end{array} \right) \begin{pmatrix} \mathbf{p} \\ \mathbf{f} \\ \mathbf{h} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{f} \\ \mathbf{h} \end{pmatrix}$$

PDE Discretization

- Euler-Lagrange PDE

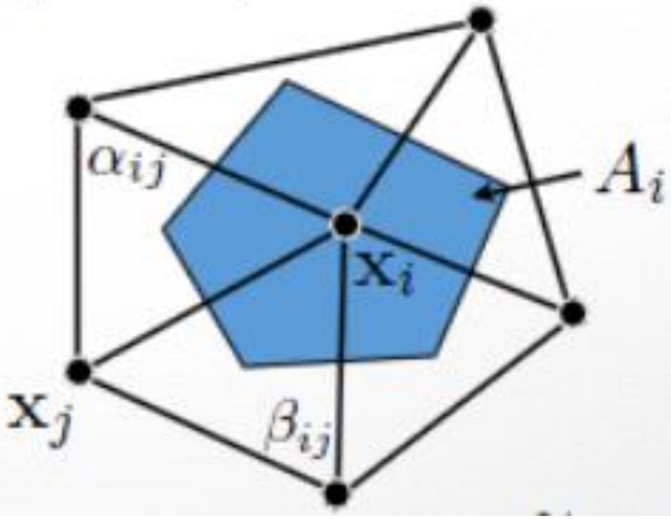
$$\Delta^2 \mathbf{d} = 0$$
$$\mathbf{d} = 0$$
$$\mathbf{d} = \delta\mathbf{h}$$



- Laplace discretization

$$\Delta \mathbf{d}_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}_i} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{d}_j - \mathbf{d}_i)$$

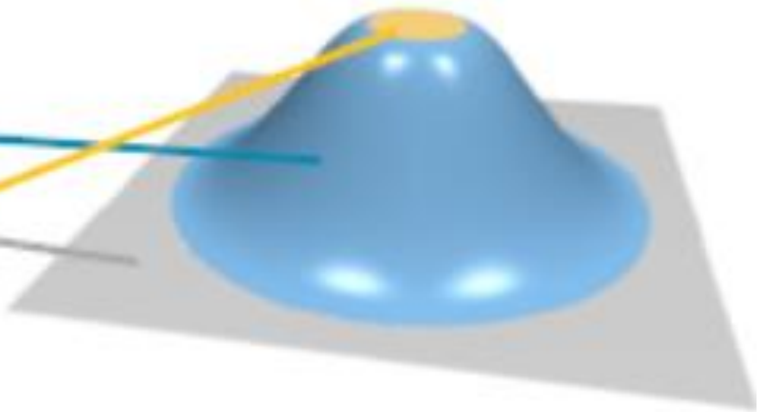
$$\Delta^2 \mathbf{d}_i = \Delta(\Delta \mathbf{d}_i)$$



Linear System

- Sparse linear system
  - Turn into symmetric positive definite system

$$\begin{pmatrix} & \Delta^2 & \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \vdots \\ \mathbf{d}_i \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \delta \mathbf{h}_i \end{pmatrix}$$



- Solve this system *each frame*
  - Use efficient linear solvers !!!
  - Sparse Cholesky factorization

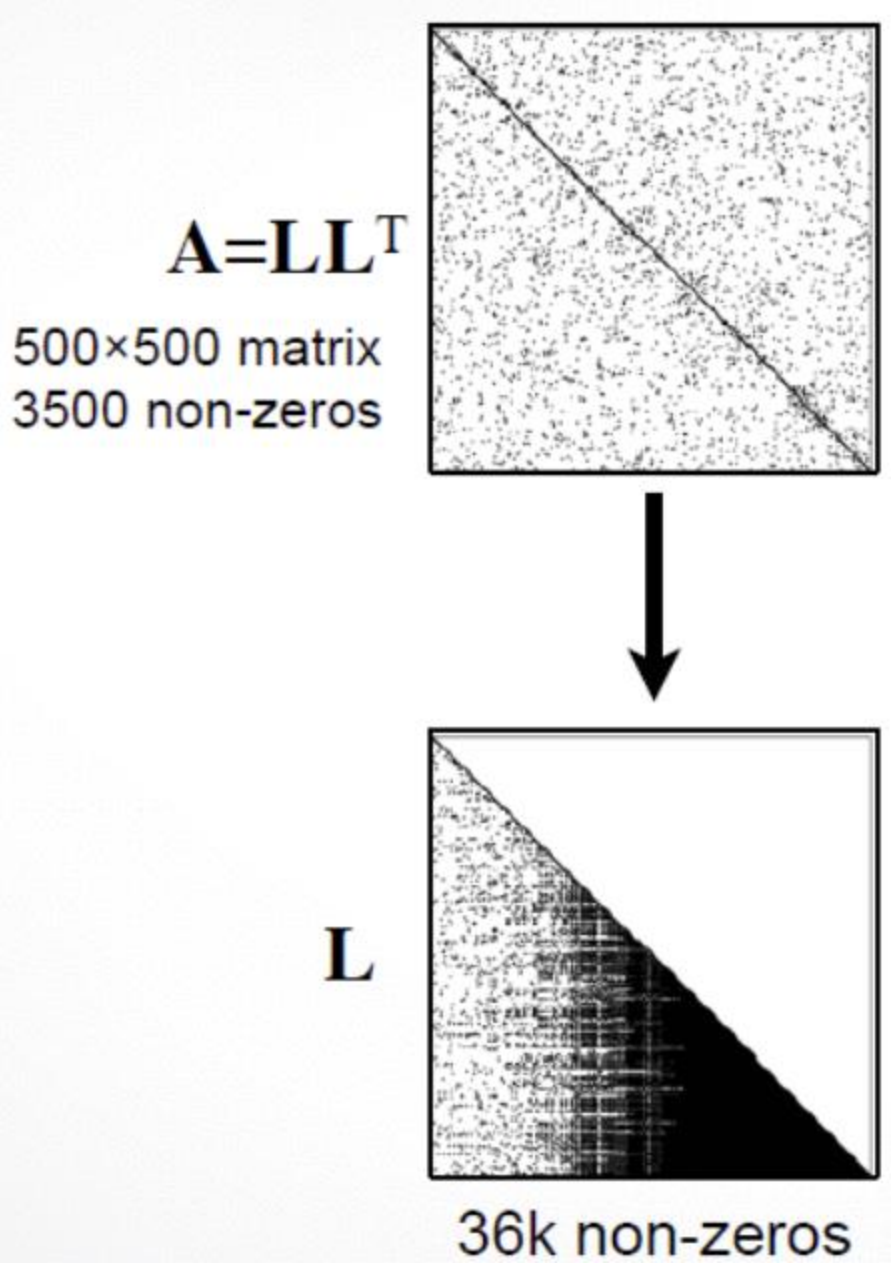
Sparse SPD Solvers

- 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?

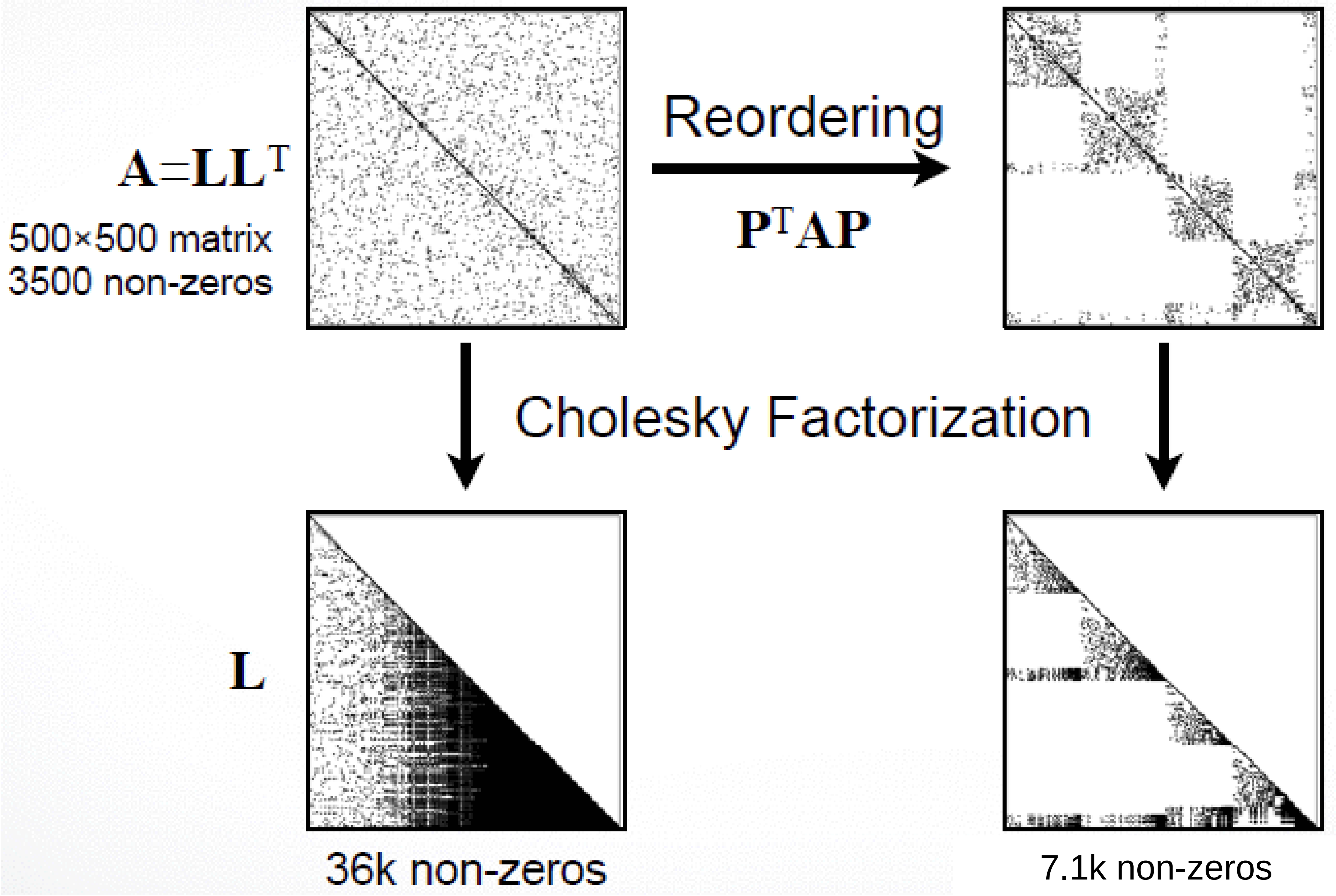
Dense Cholesky Solver

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

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



Sparse Cholesky Factorization



Sparse Cholesky Solver

Solve $\mathbf{Ax} = \mathbf{b}$

Pre-computation

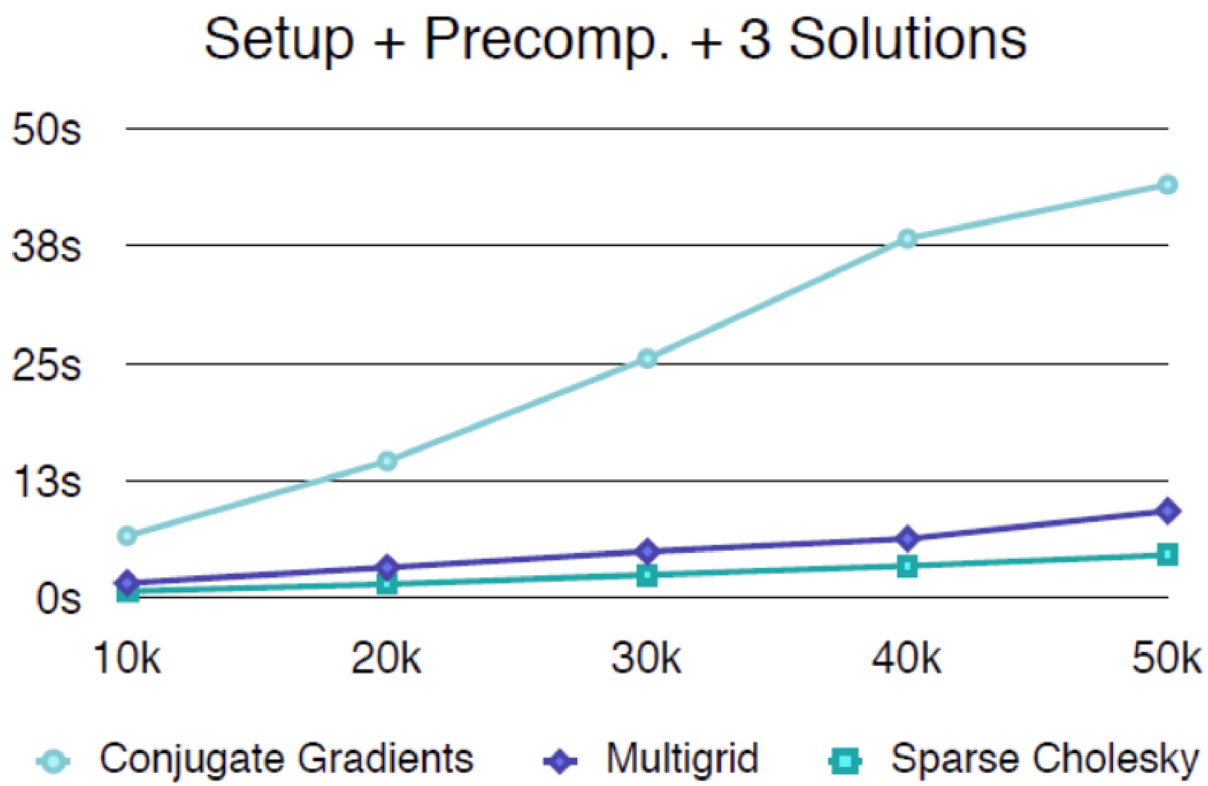
1. Matrix re-ordering $\tilde{\mathbf{A}} = \mathbf{P}^T\mathbf{AP}$

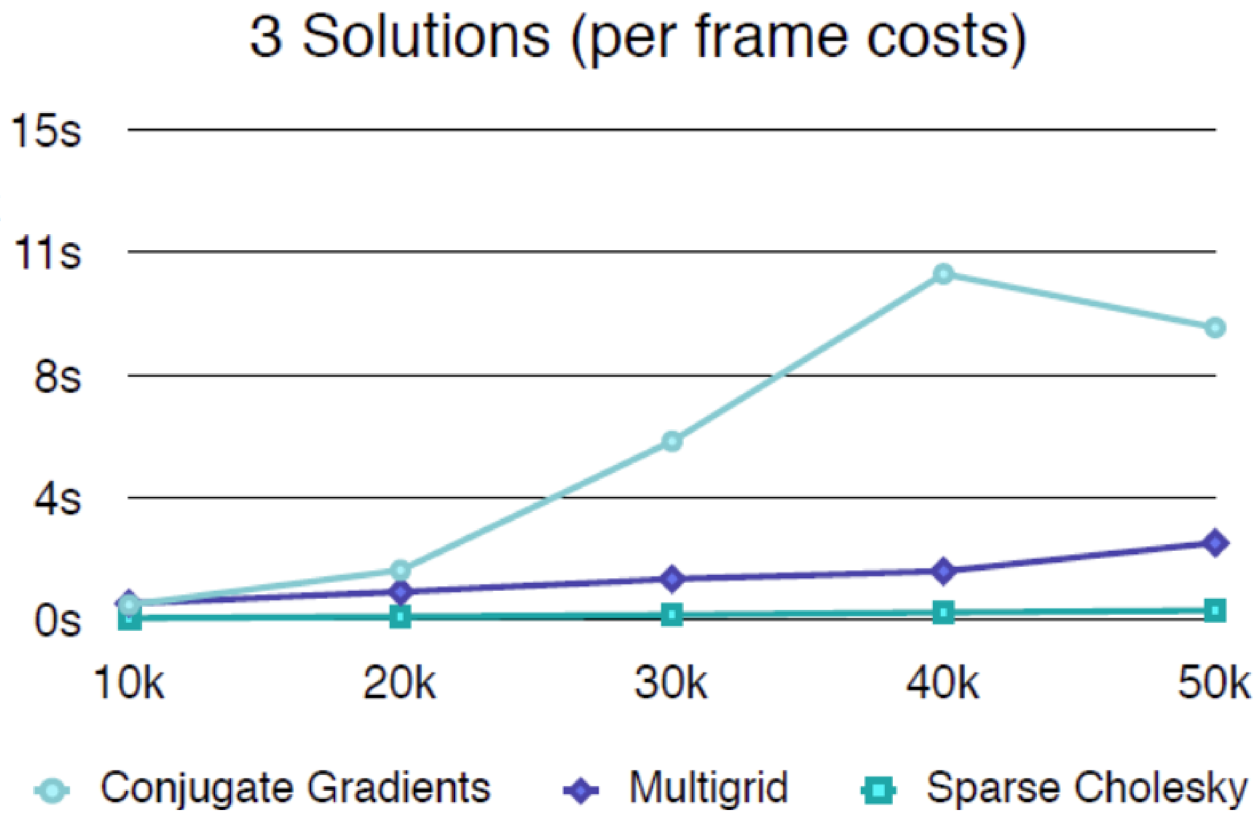
2. Cholesky factorization $\tilde{\mathbf{A}} = \mathbf{LL}^T$

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

Per-frame computation

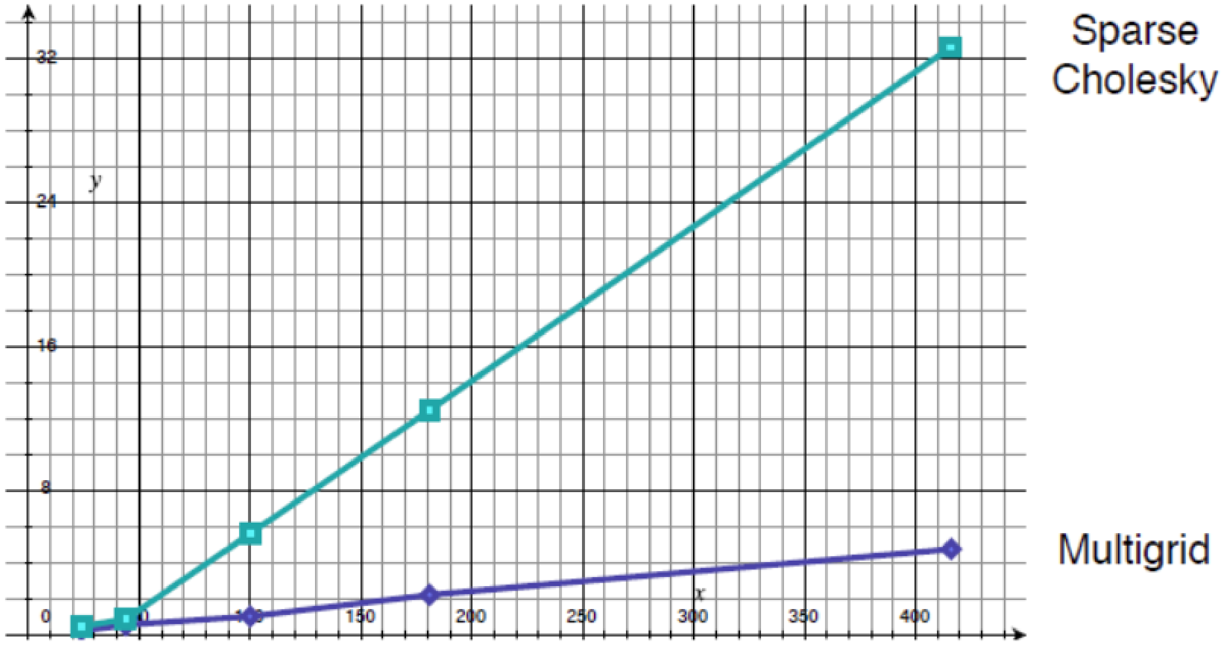
Bi-Laplace Systems



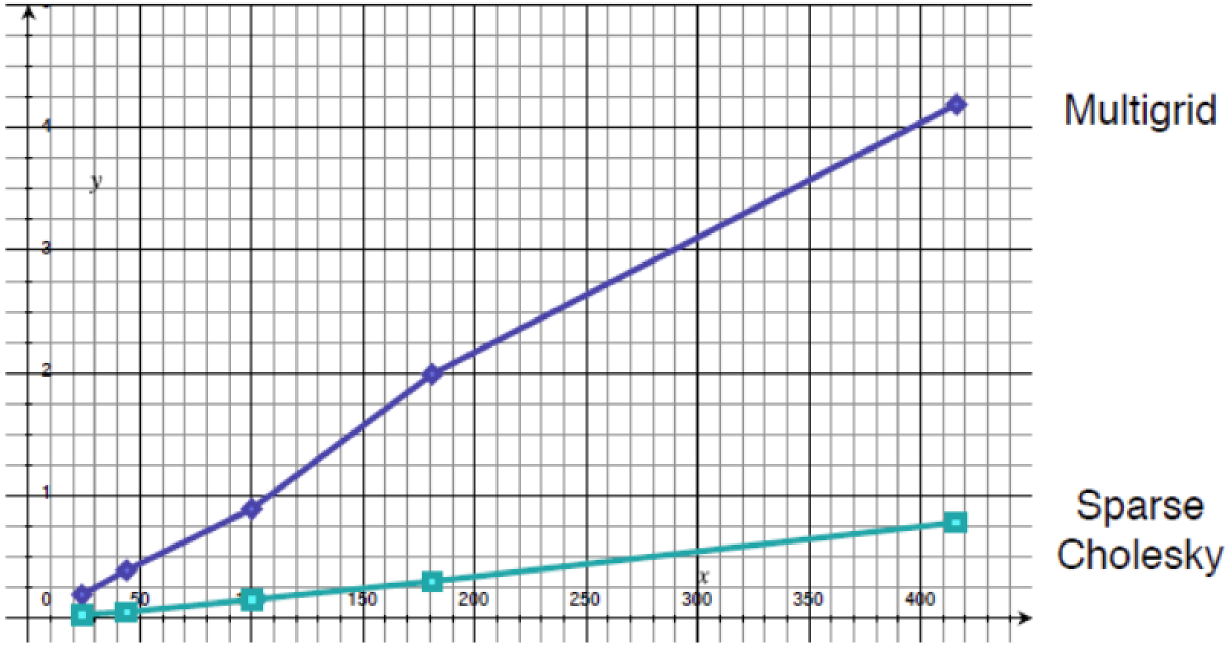


Laplace Systems

Setup + Precomp. + 3 Solutions



3 Solutions (per frame costs)

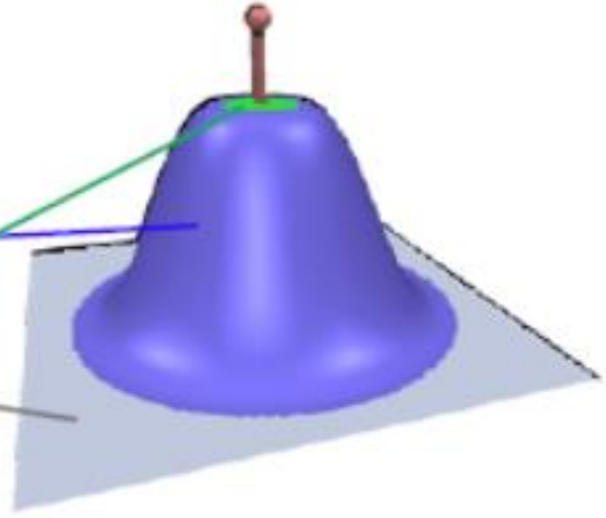


[Shi et al, SIGGRAPH 06]

Linear System

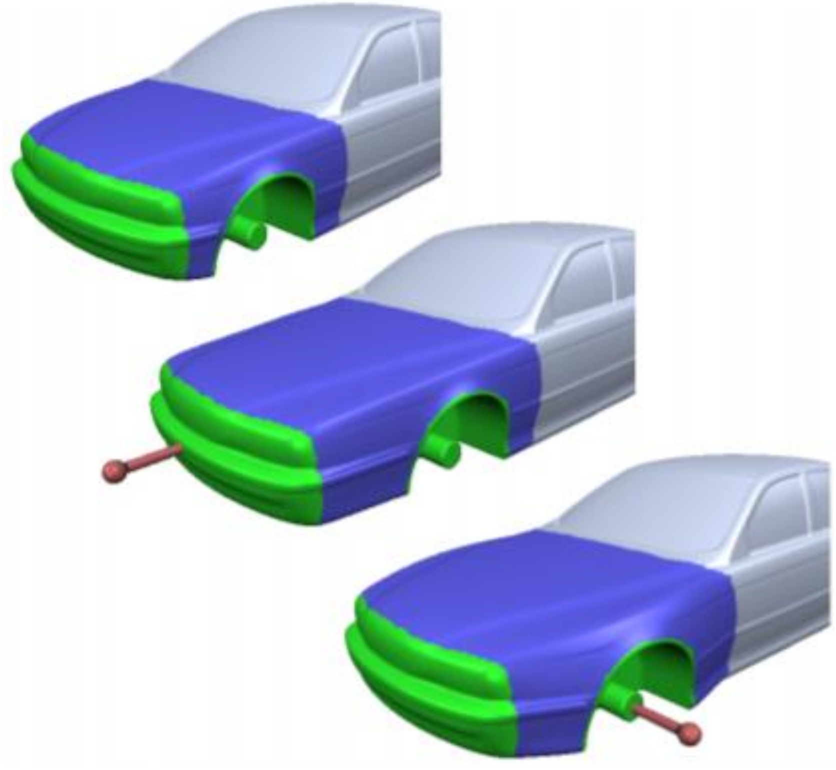
- Sparse linear system
  - Turn into symmetric positive definite system

$$\underbrace{\begin{pmatrix} & \Delta^2 & \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix}}_{=:\mathbf{M}} \begin{pmatrix} \vdots \\ \mathbf{d}_i \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \delta \mathbf{h}_i \end{pmatrix}$$



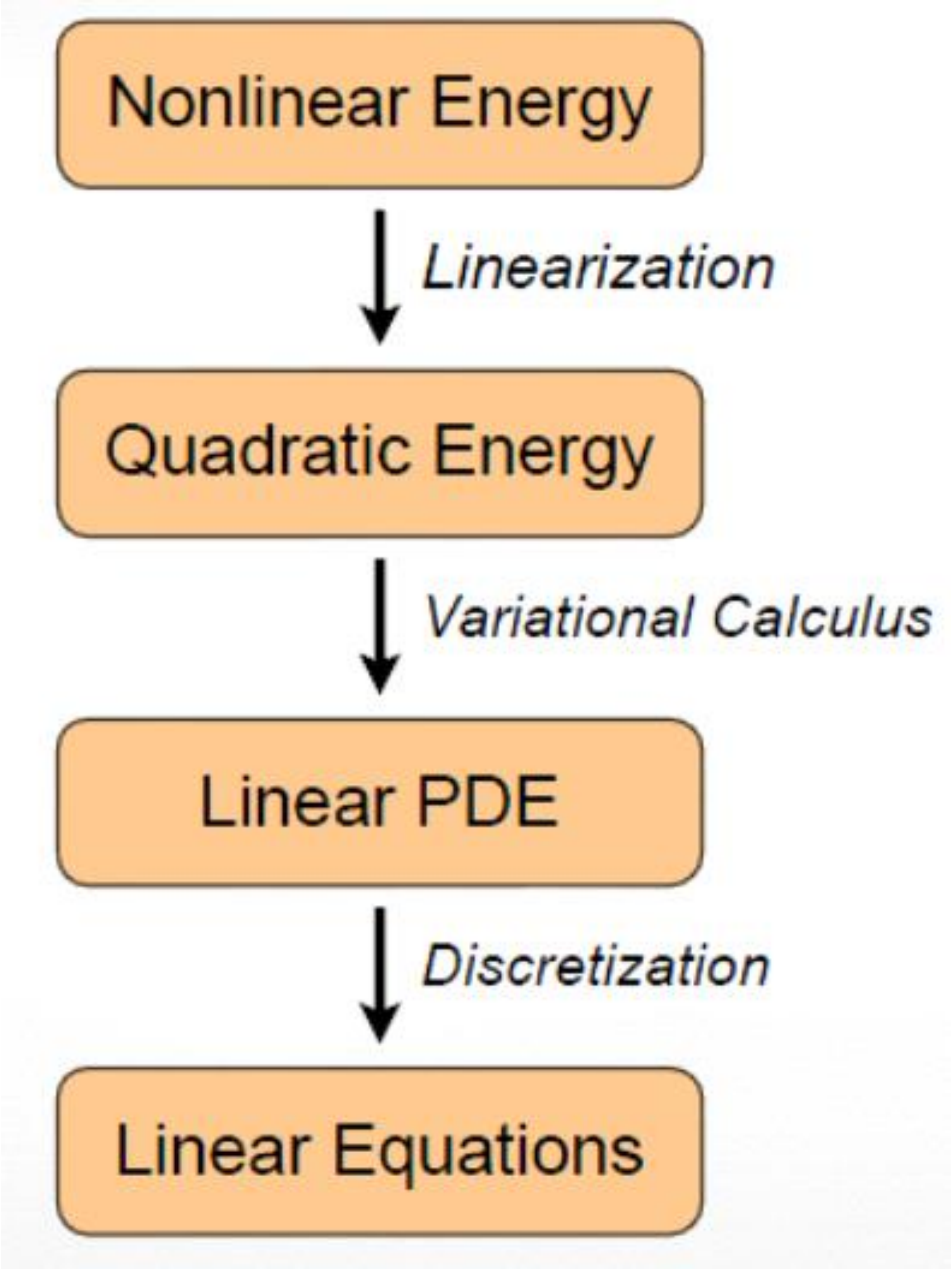
- Can be turned into symm. pos. def. system
  - Right hand sides changes each frame
  - Sparse Cholesky factorization
  - Very efficient implementations publicly available

CAD-Like Deformation & Facial Animation



[Botsch & Kobbelt, SIGGRAPH 04]

Derivation Steps

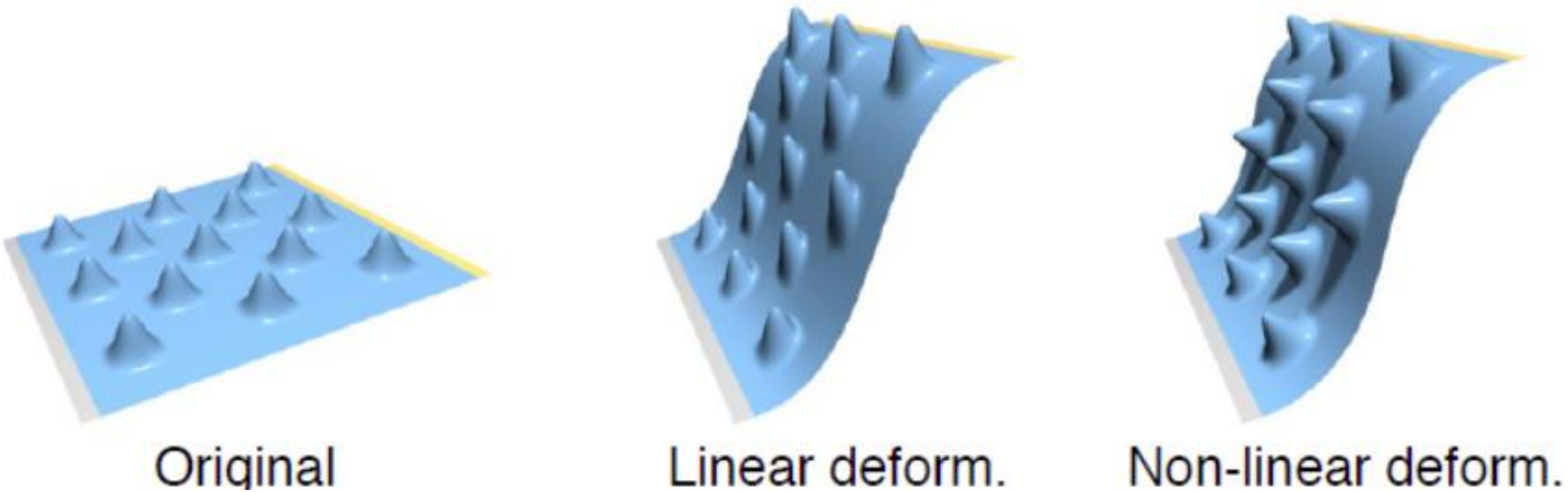


Overview

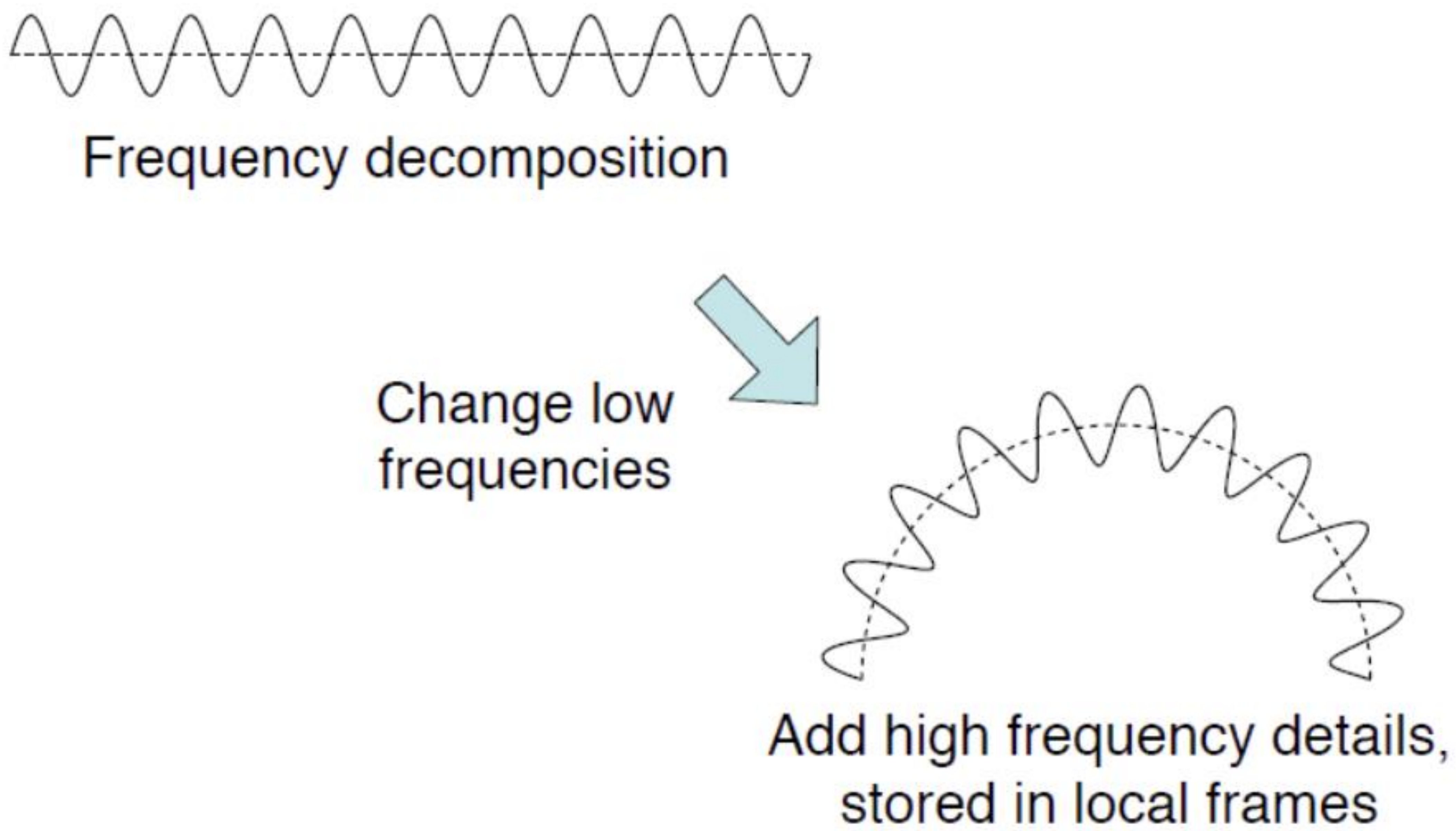
- **Surface-based deformation**
  - **Energy minimization**
  - Multiresolution editing
  - Differential coordinates

Multiresolution Modeling

- Even pure translations induce local rotations!
  - ➡ Inherently non-linear coupling
- Alternative approach
  - Linear deformation + multi-scale decomposition...

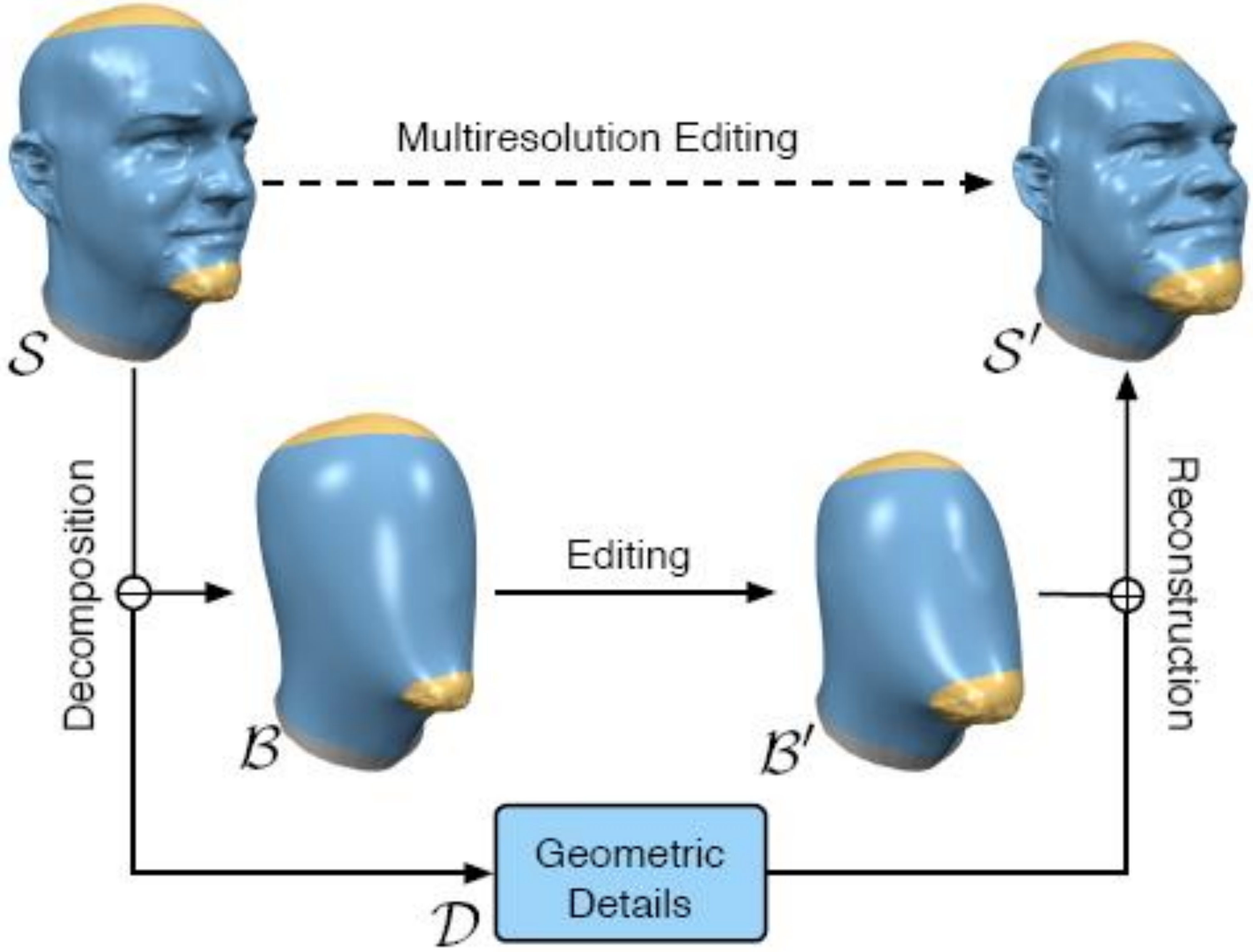


Multiresolution Editing



Multiresolution Editing [Michael Garland el at 99]

- Geometric signal – Low (global shape) /High (geometric details)



3 operators:
- **Decomposition**
- **Deformation**
- **reconstruction**

Displacement Vectors

- Decomposition: $$\mathbf{p}_i \;=\; \mathbf{b}_i + \mathbf{h}_i, \quad \mathbf{h}_i \in \mathbb{R}^3,$$
  - Represent hi via global vs local frame

$$\mathbf{h}_i \;=\; \alpha_i\,\mathbf{n}_i + \beta_i\,\mathbf{t}_{i,1} + \gamma_i\,\mathbf{t}_{i,2}.$$

- Reconstruction:

$$\mathbf{p}_i' \;=\; \mathbf{b}_i' + \alpha_i\,\mathbf{n}_i' + \beta_i\,\mathbf{t}_{i,1}' + \gamma_i\,\mathbf{t}_{i,2}'.$$

Choose t_i heuristicaly

Store hi via global vs local frame

Normal Displacements

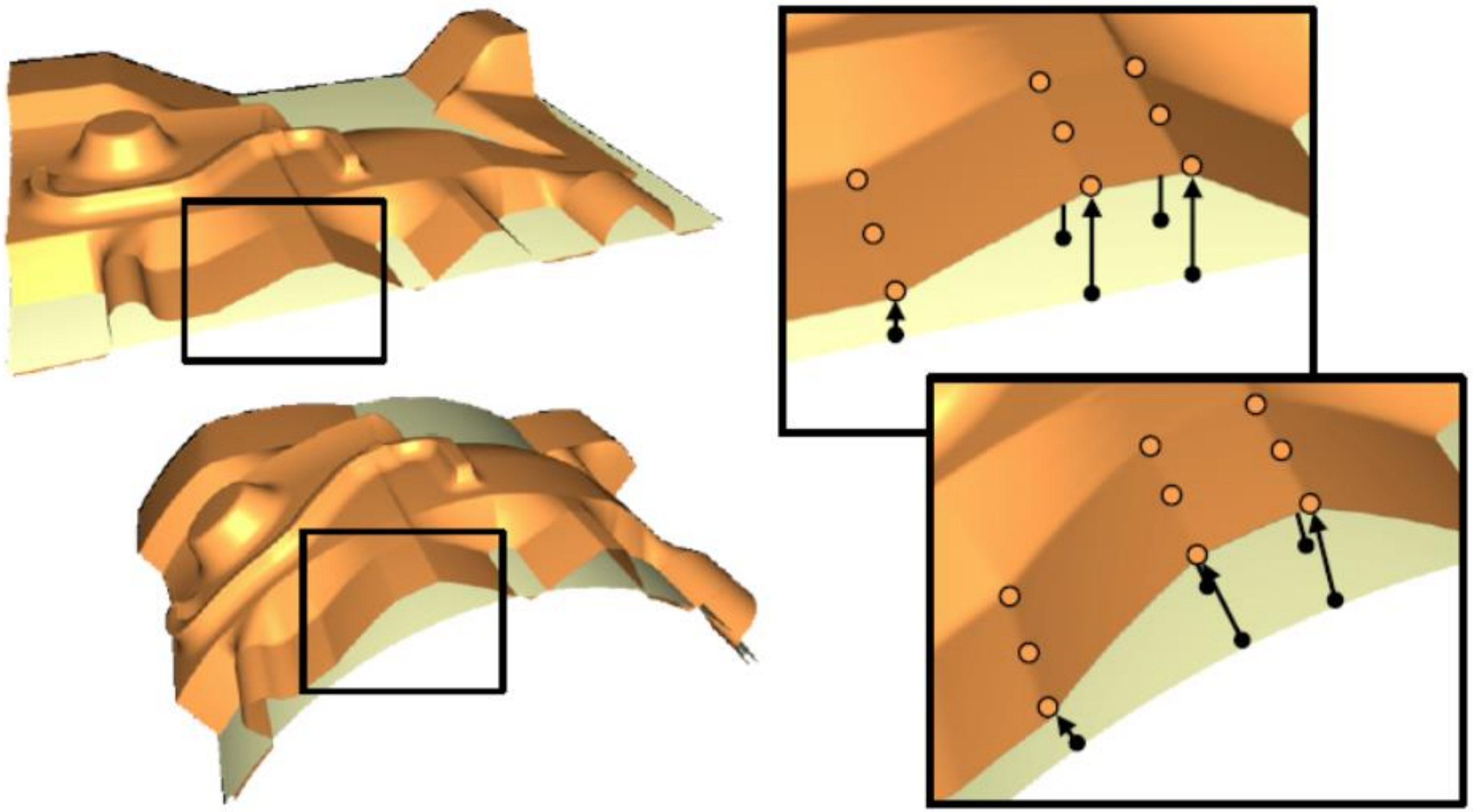
- **long** displacement vectors might lead to **instabilities**, in particular for bending deformations
- Displacement vectors should connect vertices pi of S to their closest surface points on B instead of their corresponding vertices bi of B.

$$\mathbf{p}_i = \mathbf{b}_i + h_i \cdot \mathbf{n}_i, \quad h_i \in \mathbb{R}.$$

- How to compute bi
- Resampling may introduce

alias artifacts

- Local Newton iteration



Normal Displacements – compute bi via Newton iteration

- find the barycentric coordinates of the base point bi as the root of the function

$$f(\alpha,\beta,\gamma) = (\mathbf{p}_i - \mathbf{b}_i) \times \mathbf{n}_i$$

- where

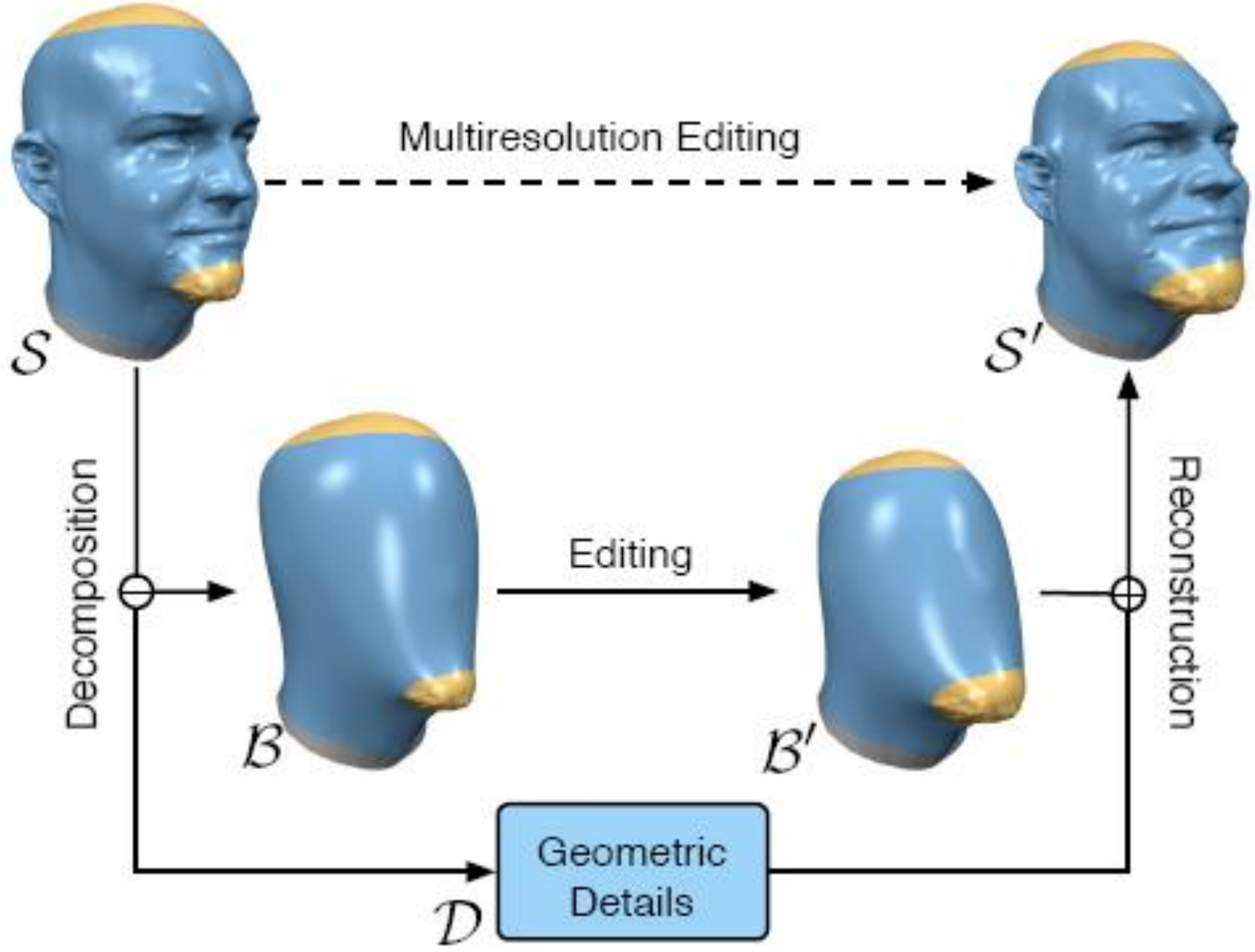
$$\mathbf{b}_i = \alpha\,\mathbf{a} + \beta\,\mathbf{b} + \gamma\,\mathbf{c}.$$

$$\mathbf{n}_i = \frac{\alpha\,\mathbf{n_a} + \beta\,\mathbf{n_b} + \gamma\,\mathbf{n_c}}{\|\alpha\,\mathbf{n_a} + \beta\,\mathbf{n_b} + \gamma\,\mathbf{n_c}\|}.$$

- The process is **initialized** with the triangle closest to pi. If a barycentric coordinate becomes negative during the Newton iteration, one proceeds to the respective neighboring triangle.
- Then graph of S and B is no longer restricted to be identical, which can be exploited to **remesh B** for the sake of higher **numerical robustness**.

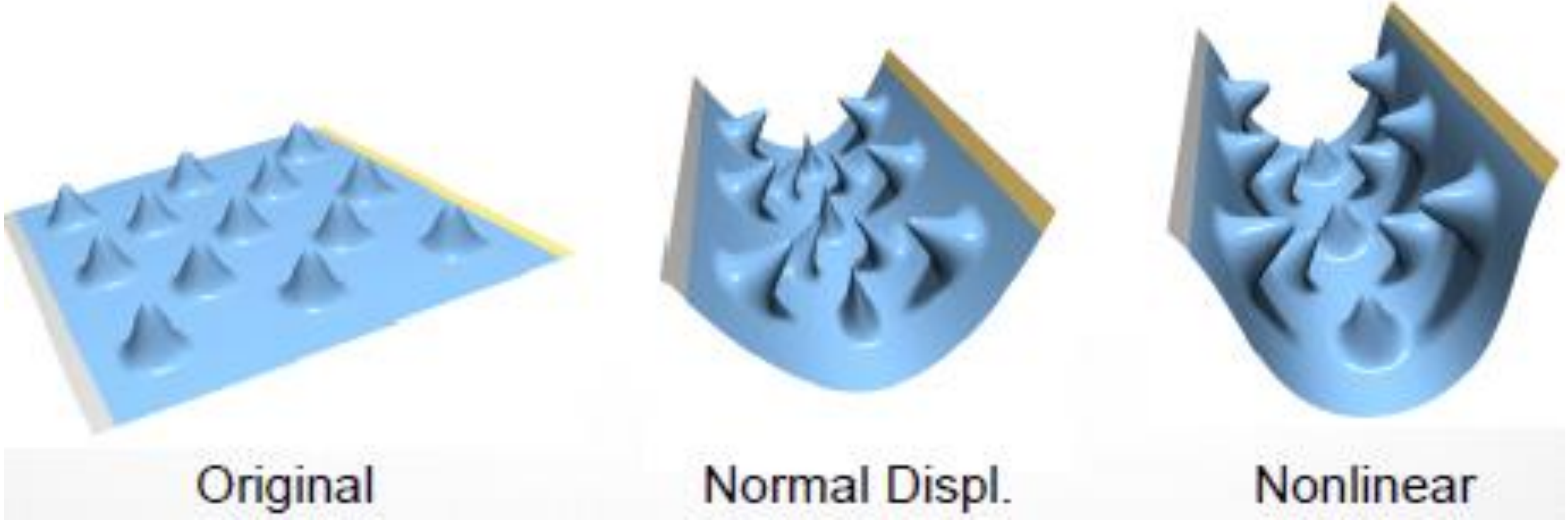
Multiresolution Editing [Michael Garland el at 99]

- the general displacements are in average about 9 times longer than normal displacements



Limitations

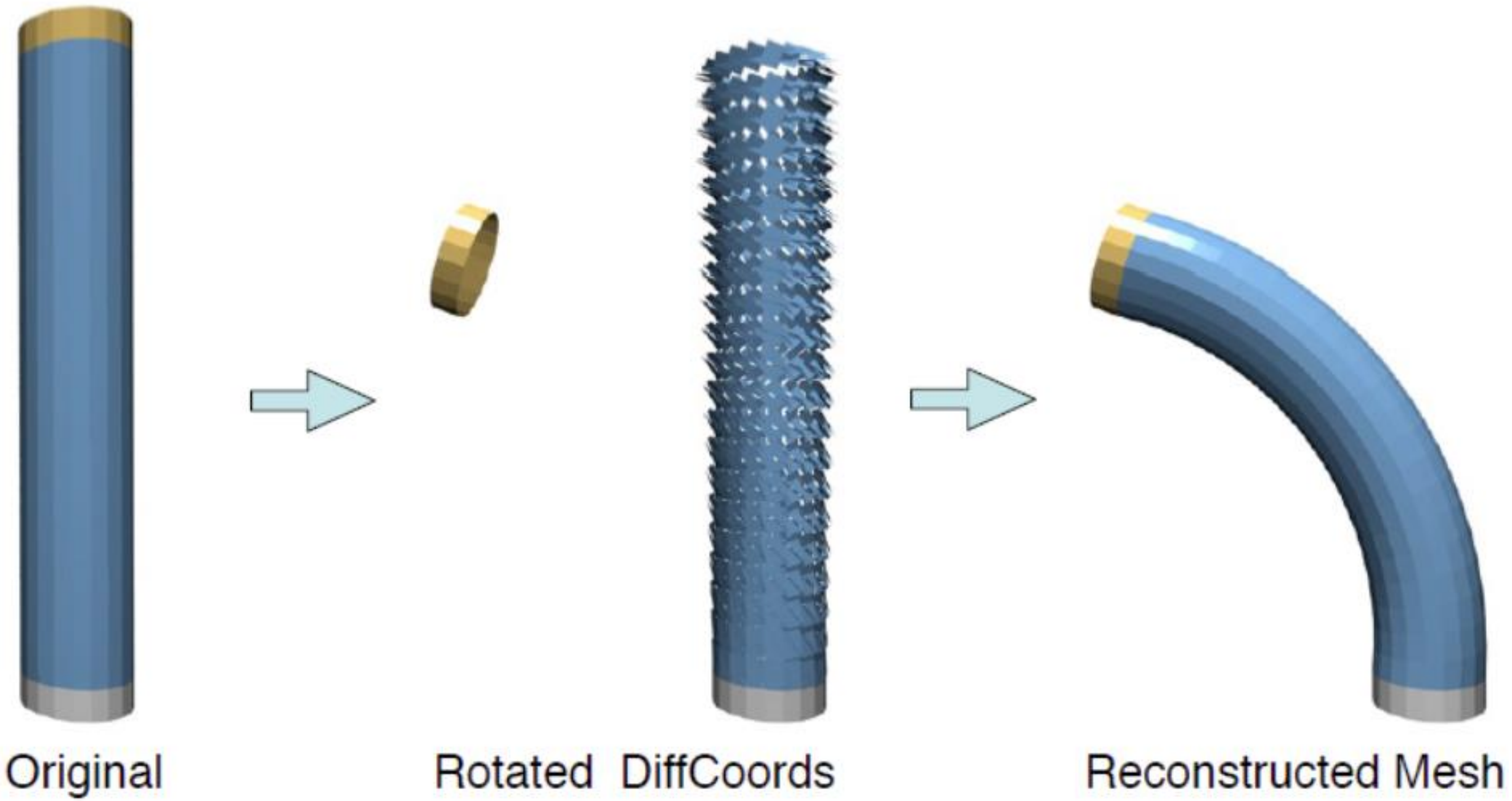
- Neighboring displacements are not coupled
  - Surface bending changes the angle between neighboring displacement vectors
  - Leads to volume changes or self-intersections
- Multiresolution hierarchy difficult to compute
  - Complex topology or Complex geometry might require more hierarchy levels



Differential Coordinates

- Manipulate *differential coordinates* instead of *spatial* coordinates
  - Gradients, Laplacians, local frames
  - Intuition: Close connection to surface normal
- Find mesh with desired differential coords
  - Formulate as energy minimization

Using gradient-based editing to bend the cylinder by 90



Rotating the handle and propagating its **damped local rotation** to the individual triangles (resp. their gradients JT) breaks up the mesh (center),
but solving the **Poisson system** reconnects it and yields the desired result (right).

Gradient-Based Editing

- Manipulate gradient field of a function (surface)

$$\mathbf{g} = \nabla \mathbf{p} \qquad \mathbf{g} \mapsto \mathbf{g}'$$

- Find function **f**' whose gradient is (close to) **g**'

$$\int_{\mathcal{S}} \left\| \nabla \mathbf{p}' - \mathbf{g}' \right\|^2 \mathrm{d}\mathcal{S} \;\rightarrow\; \min$$

- Variational calculus yields Euler-Lagrange PDE

$$\Delta \mathbf{p}' = \operatorname{div} \mathbf{g}'$$

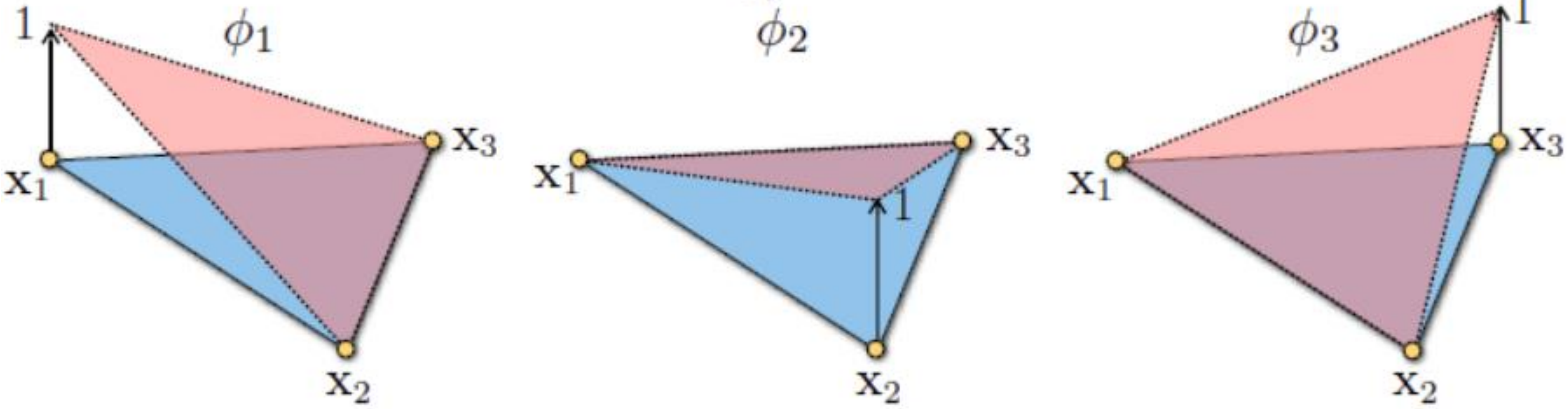
Gradient-Based Editing

- Use piecewise linear coordinate function

$$\mathbf{p}(u,v) = \sum_{v_i} \mathbf{p}_i \cdot \phi_i(u,v)$$

- Its gradient is

$$\nabla \mathbf{p}(u,v) = \sum_{v_i} \mathbf{p}_i \cdot \nabla \phi_i(u,v)$$



Gradient-Based Editing

$$\nabla \mathbf{p}|_T = \begin{bmatrix} \nabla \mathbf{p}_x|_T \\ \nabla \mathbf{p}_y|_T \\ \nabla \mathbf{p}_z|_T \end{bmatrix} =: \mathbf{J}_T \in \mathbb{R}^{3\times 3}$$

- Constant per triangle $\nabla \mathbf{p}|_{f_j} =: \mathbf{G}_j \in \mathbb{R}^{3\times 3}$

Gj =: J_T

$$\begin{pmatrix} \mathbf{G}_1 \\ \vdots \\ \mathbf{G}_F \end{pmatrix} = \underbrace{\mathbf{G}}_{\in \mathbb{R}^{3F\times V}} \cdot \begin{pmatrix} \mathbf{p}_1^T \\ \vdots \\ \mathbf{p}_V^T \end{pmatrix}$$

- Manipulate per-face gradients

$$\mathbf{G}_j \mapsto \mathbf{G}'_j$$

$$\mathbf{J}'_T = \mathbf{M}_T \mathbf{J}_T$$

Gradient-Based Editing

- Reconstruct mesh from changed gradients
  - Overdetermined problem $\mathbf{G} \in \mathbb{R}^{3F \times V}$
  - Weighted least squares system
  - Linear Poisson (Laplace) system

$$\underbrace{\boxed{\mathbf{G}^T\mathbf{D}\mathbf{G}}}_{\mathrm{div}\nabla = \Delta} \cdot \begin{pmatrix} \mathbf{p}_1'^T \\ \vdots \\ \mathbf{p}_V'^T \end{pmatrix} = \underbrace{\boxed{\mathbf{G}^T\mathbf{D}}}_{\mathrm{div}} \cdot \begin{pmatrix} \mathbf{G}_1' \\ \vdots \\ \mathbf{G}_F' \end{pmatrix}$$

Laplacian-Based Editing

- Manipulate Laplacians of a surface

$$\boldsymbol{\delta}_i = \Delta(\mathbf{p}_i) \ , \quad \boldsymbol{\delta}_i \mapsto \boldsymbol{\delta}'_i$$

- Find surface whose Laplacian is (close to) **δ**'

$$\int_{\mathcal{S}} \left\| \Delta \mathbf{p}' - \boldsymbol{\delta}' \right\|^2 \mathrm{d}\mathcal{S} \ \rightarrow \ \min$$

- Variational calculus yields Euler-Lagrange PDE

$$\Delta^2 \mathbf{p}' = \Delta \boldsymbol{\delta}'$$

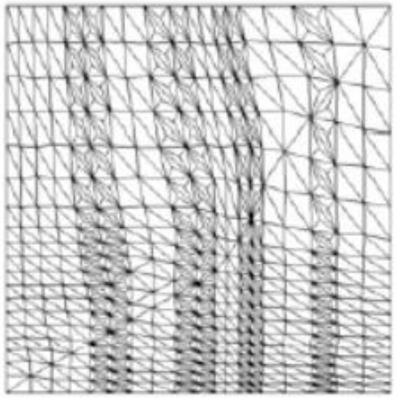
Discretization

- Discretize Euler-Lagrange PDE

$$\Delta^2 \mathbf{p}' = \Delta \delta' \longrightarrow \mathbf{L}^2 \mathbf{p}' = \mathbf{L}\delta'$$

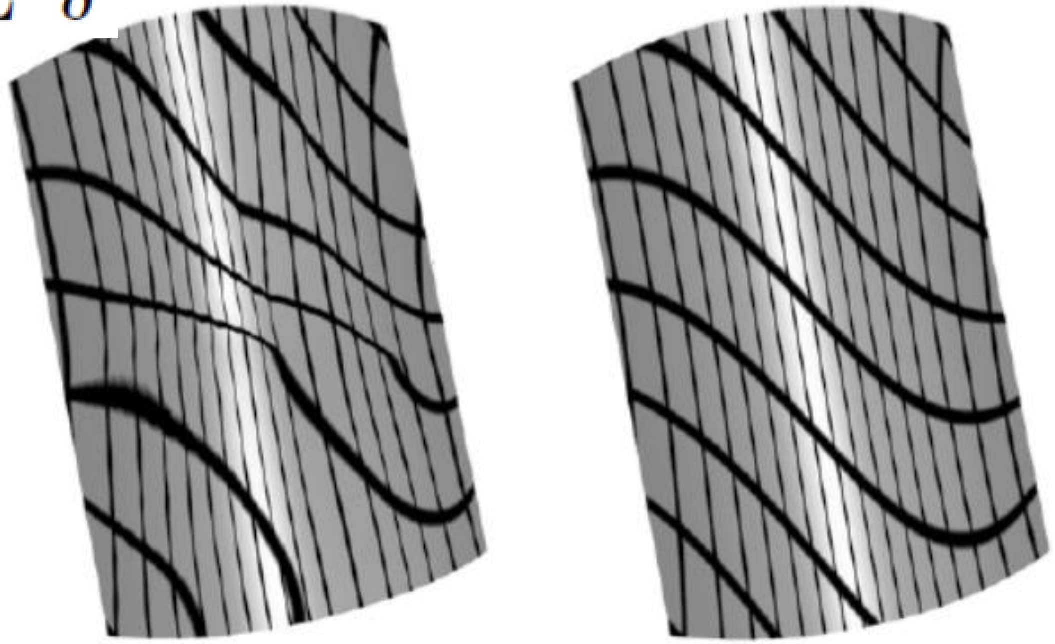
- Frequently used (wrong) version

$$\delta = \mathbf{L}\mathbf{p} \longrightarrow \delta \mapsto \delta' \longrightarrow \mathbf{L}^T\mathbf{L}\mathbf{p}' = \mathbf{L}^T\delta'$$



Irregular mesh

Wrong



$\mathbf{L}^T\mathbf{L}\mathbf{p}' = \mathbf{L}^T\delta'$ $\qquad$ $\mathbf{L}^2\mathbf{p}' = \mathbf{L}\delta'$

Connection to Plate Energy?

- Neglect change of Laplacians for a moment...

$$\int \|\Delta \mathbf{p}' - \delta\|^2 \rightarrow \min \quad \longrightarrow \quad \Delta^2 \mathbf{p}' = \Delta\delta$$

$$\Big\downarrow \; \begin{matrix} \mathbf{p}' = \mathbf{p} + \mathbf{d} \\ \delta = \Delta \mathbf{p} \end{matrix}$$

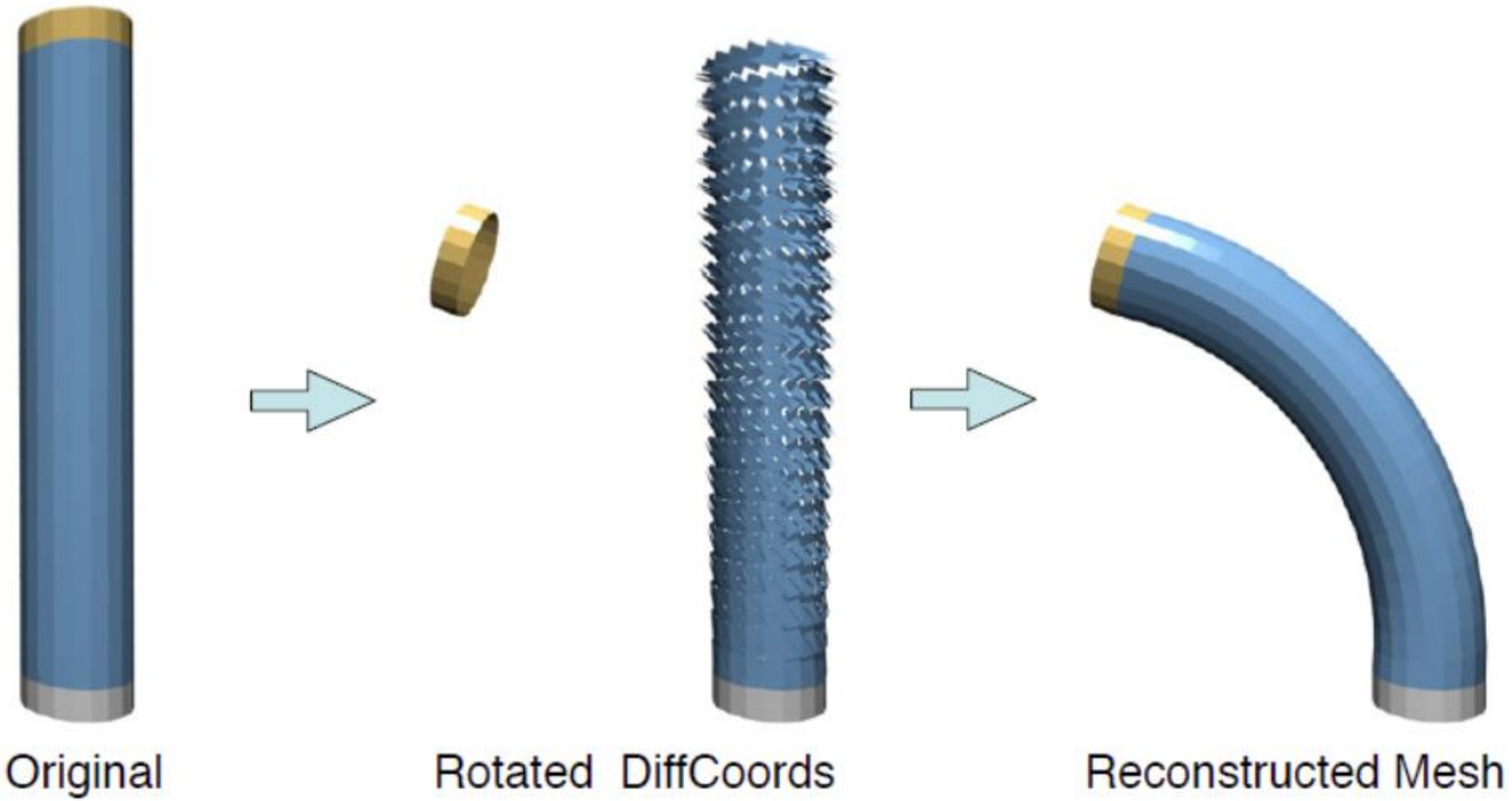
$$\Delta^2(\mathbf{p} + \mathbf{d}) = \Delta^2 \mathbf{p}$$

$$\Big\downarrow$$

$$\int \|\mathbf{d}_{uu}\|^2 + 2\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2 \rightarrow \min \quad \longleftarrow \quad \Delta^2 \mathbf{d} = 0$$

- Basic formulations equivalent!
- Differ in detail preservation
  - Rotation of Laplacians
  - Multi-scale decomposition

Using gradient-based editing to bend the cylinder by 90



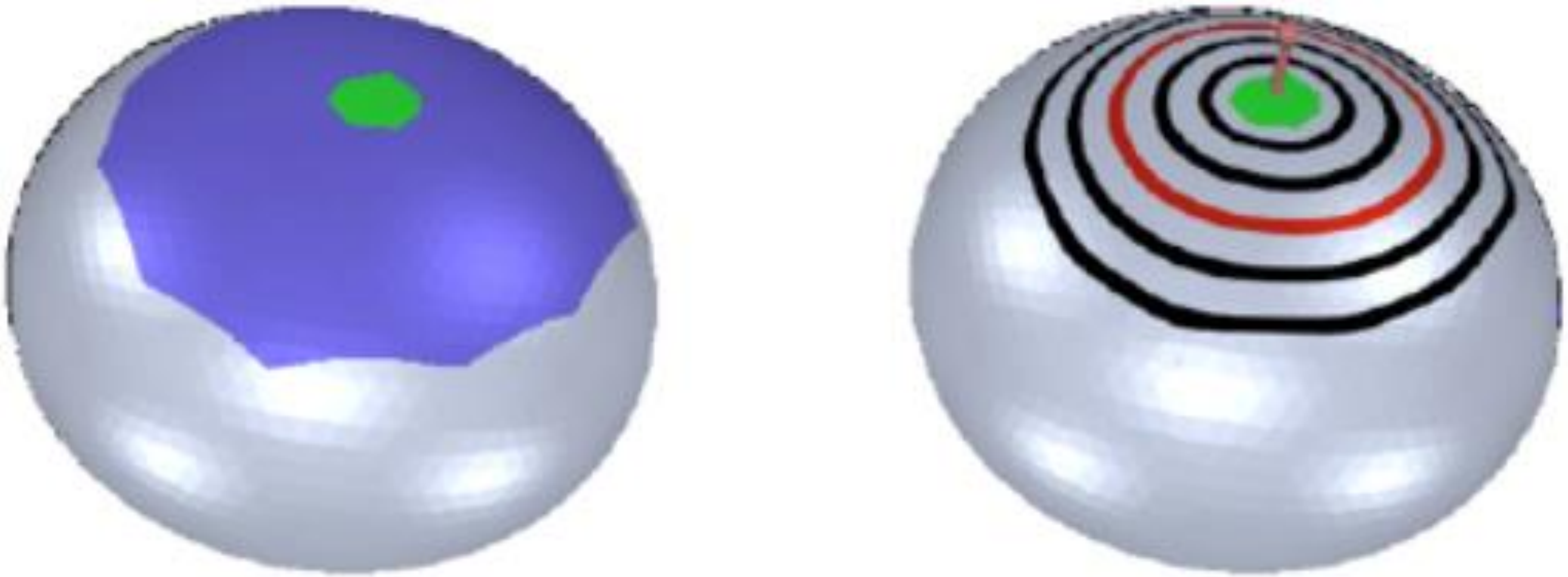
Rotating the handle and propagating its **damped local rotation** to the individual triangles (resp. their gradients JT) breaks up the mesh (center),
but solving the **Poisson system** reconnects it and yields the desired result (right).

Differential Coordinates

- Which differential coordinate $\boldsymbol{\delta}_i$?
  - Gradients
  - Laplacians
  - ...
- **How to get local transformations $T_i(\boldsymbol{\delta}_i)$?**
  - Smooth propagation
  - Implicit optimization
  - ...

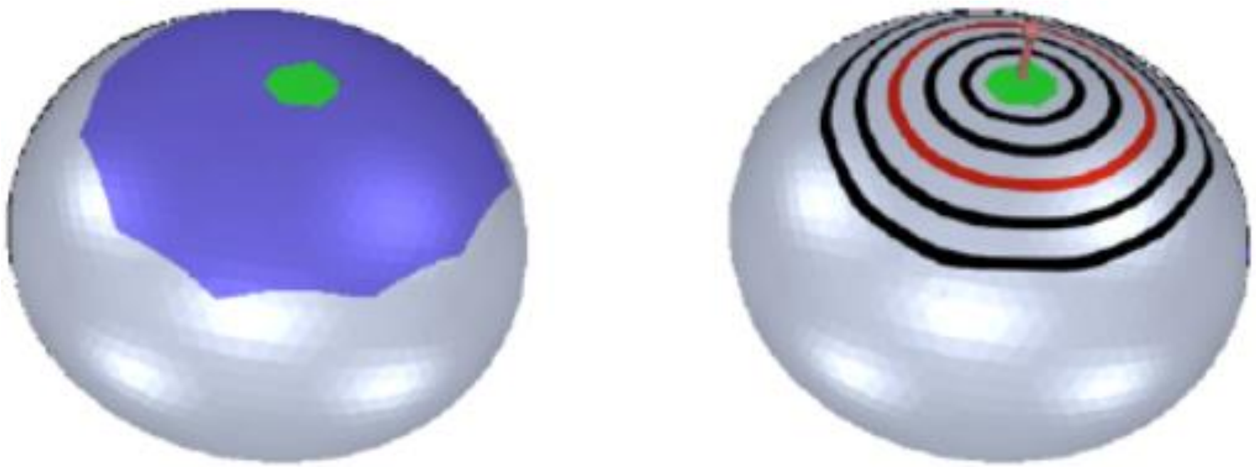
Construct Scalar Field

- Construct smooth scalar field [0,1]
  - $s(x)=1$: Full deformation (handle)
  - $s(x)=0$: No deformation (fixed part)
  - $s(x)\in(0,1)$: Damp handle transformation (in between)



Construct Scalar Field

- Construct a smooth harmonic field
  - Solve $\Delta(s) = 0$
  - with $s(\mathbf{p}) = \begin{cases} 1 & \mathbf{p} \in \text{handle} \\ 0 & \mathbf{p} \in \text{fixed} \end{cases}$



Damp Handle Transformation

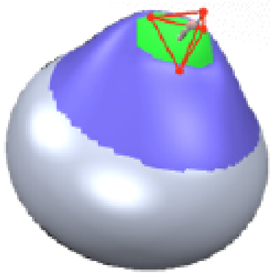
- Full handle transformation
  - Rotation: $R(\mathbf{c}, \mathbf{a}, \alpha)$
  - Scaling: $S(s)$

- Damped by scalar λ
  - Rotation: $R(\mathbf{c}, \mathbf{a}, \lambda \cdot \alpha)$
  - Scaling: $S(\lambda \cdot s + (1-\lambda) \cdot 1)$

Damp Handle Transformation

- Handle has been transformed *affinely*

$$\mathbf{T}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{t}$$



- Deformation gradient is

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

- Extract rotation $\mathbf{R}$ and scale/shear $\mathbf{S}$

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \quad \Rightarrow \quad \mathbf{R} = \mathbf{U}\mathbf{V}^T,\ \mathbf{S} = \mathbf{V}\Sigma\mathbf{V}^T$$

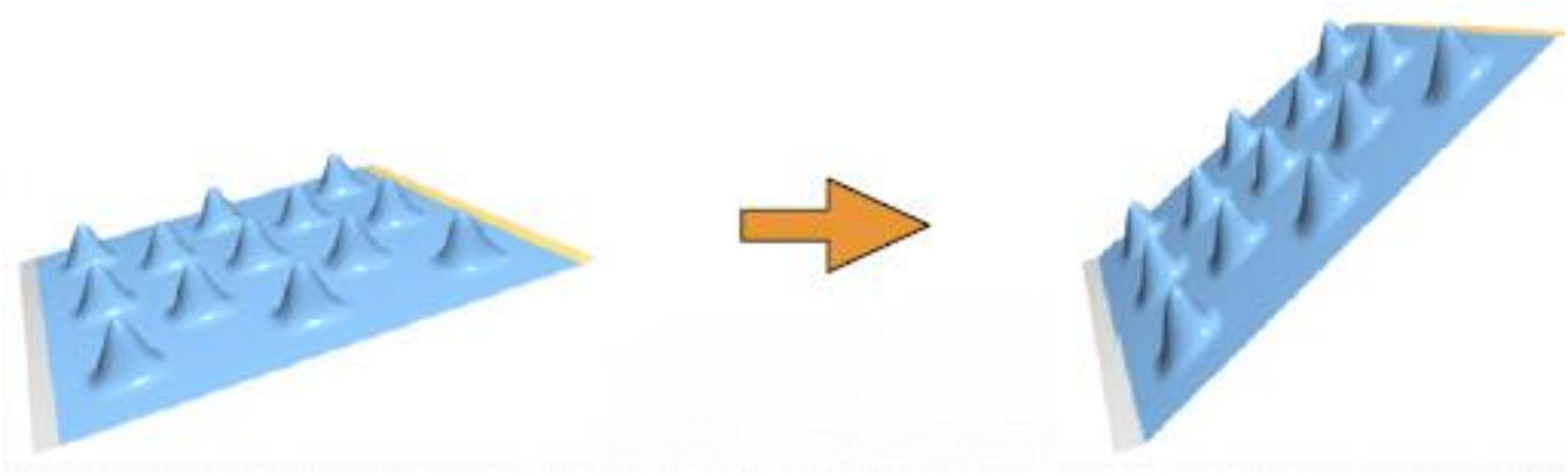
polar decomposition

$$\mathbf{RS} = \mathbf{U}\mathbf{V}^T\mathbf{V}\Sigma\mathbf{V}^T = \mathbf{U}\Sigma\mathbf{V}^T = \mathbf{M}.$$

$$\mathbf{M}_i = \mathrm{slerp}(\mathbf{R}, \mathbf{Id}, s_i) \cdot ((1 - s_i)\,\mathbf{S} + s_i\,\mathbf{Id})\,,$$

Limitations

- Differential coordinates work well for **rotations**
  - Represented by deformation gradient
- **Translations** don't change deformation gradient
  - Translations don't change differential coordinates
  - *"Translation insensitivity"*



Implicit Optimization

- **Simultaneously** optimize: new vertex positions p’ & local rotations Mi

$$E(\mathbf{p}') = \sum_{i=1}^{n} A_i \left\| \mathbf{M}_i \, \delta_i - \Delta \mathbf{p}'_i \right\|^2$$

- Local transformations are restricted to linearized **similarity transformations**

$$\mathbf{M}_i = \begin{bmatrix} s_i & -h_{i,z} & h_{i,y} \\ h_{i,z} & s_i & -h_{i,x} \\ -h_{i,y} & h_{i,x} & s_i \end{bmatrix}$$

- Extracting (si; hi) as linear combinations of p’i.

$$\mathbf{M}_i(\mathbf{p}_i - \mathbf{p}_j) = \mathbf{p}'_i - \mathbf{p}'_j, \quad \forall \mathbf{p}_j \in \mathcal{N}_1(\mathbf{p}_i)$$

Thanks