Shape deformation (III)

Surface-based deformation

Laplacian deformation
ARAP - As Rigid As Possible

Spatial / Surface based deformation

Consider a shape defined as $(p_i)_{i \in [0, N-1]}$ positions

Spatial/volume deformation

Define spatial deformation $\forall p \in \mathbb{R}^3$, $f : p \to f(p)$

Apply f to every positions

$\forall i \in [0, N-1]$, $q_i = f(p_i)$

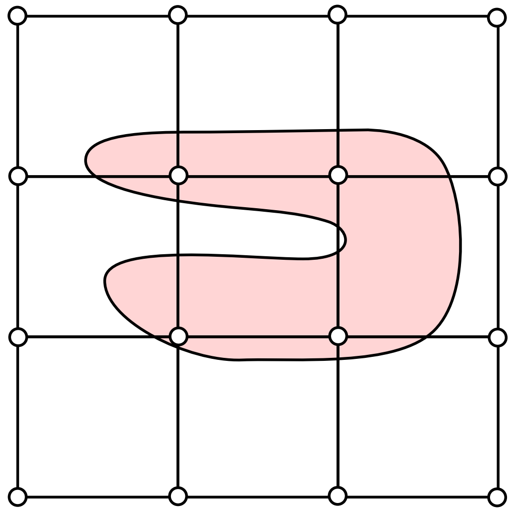
(+) Independant from shape representation

Point sets, Surface, Volume.

(-) Indirect control

Spatial deformation $\rightarrow$ shape deformation

(-) Deformation fully defined by spatial position



Surface-based

Compute deformation on surface only

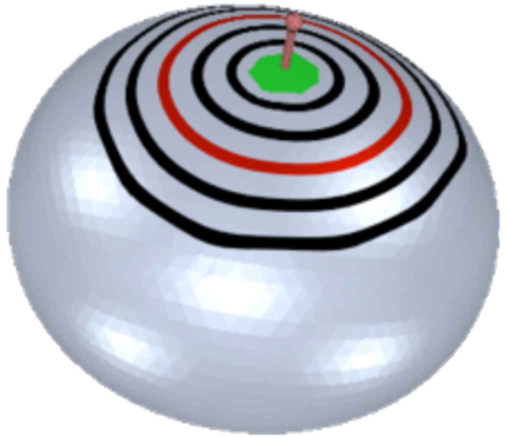
Deformation is defined at p_i

But not at arbitrary position in space

ex. $q_i = p_i + n_i$, n_i:surface normal at position p_i

(+) Can integrate/preserve surface properties (neighborood, curvature, etc.)

(-) Sensible to surface representation & connectivity.



Laplacian/Differential Coordinates Editing

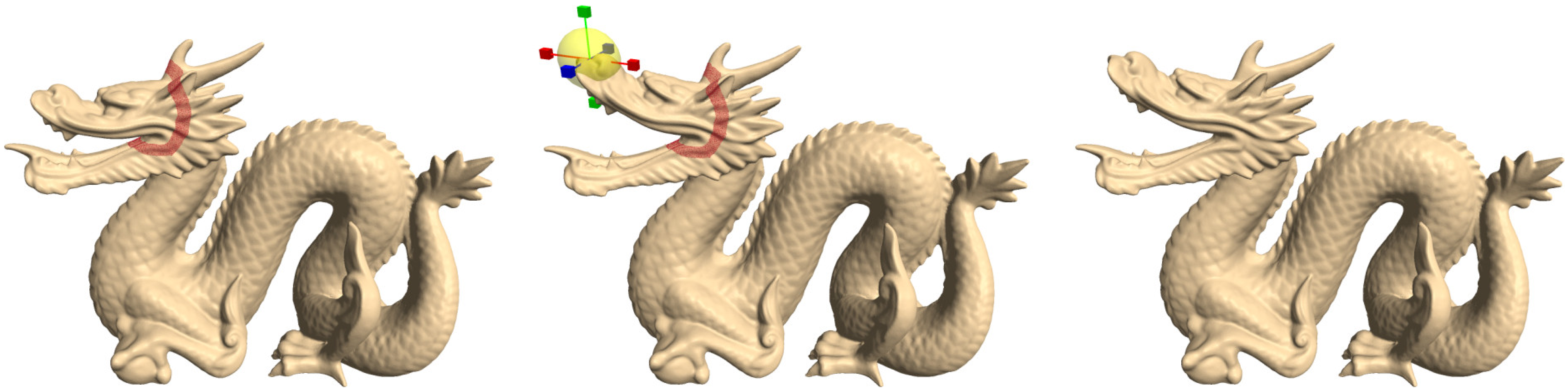
Initialization: Encode vertex position as *differential coordinates* w/r neighbors

Ex. Laplacian coordinate value δp

During deformation

- Try to match constrained position for subset of vertices
- While preserving the differential coordinates

  Local neighborhoods appearance



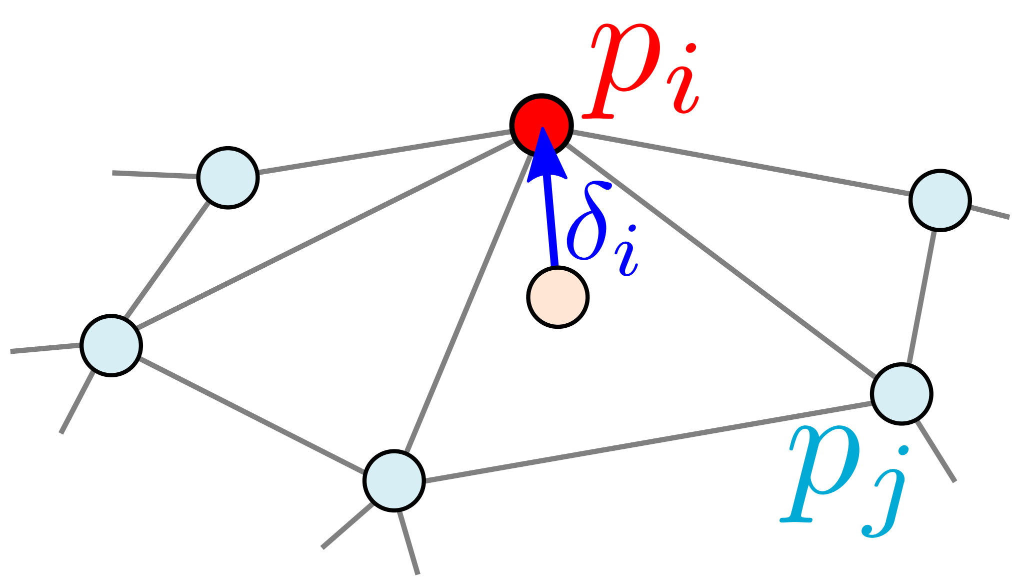
Laplacian of the coordinates

- For a scalar/vector function f defined in $\mathbb{R}^2$ with parameter (u, v): $\triangle f = \frac{\partial^2 f}{\partial u^2} + \frac{\partial^2 f}{\partial v^2} = div(grad(f))$
- Laplacian on a manifold: Called Laplace-Beltrami operator.
- Differential coordinates δ: Laplace-Beltrami operator applied on coordinates functions $f = (x, y, z)$ itself.

Common approximation in the discrete case

$$\delta_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} (p_i - p_j) = p_i - \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} p_j$$

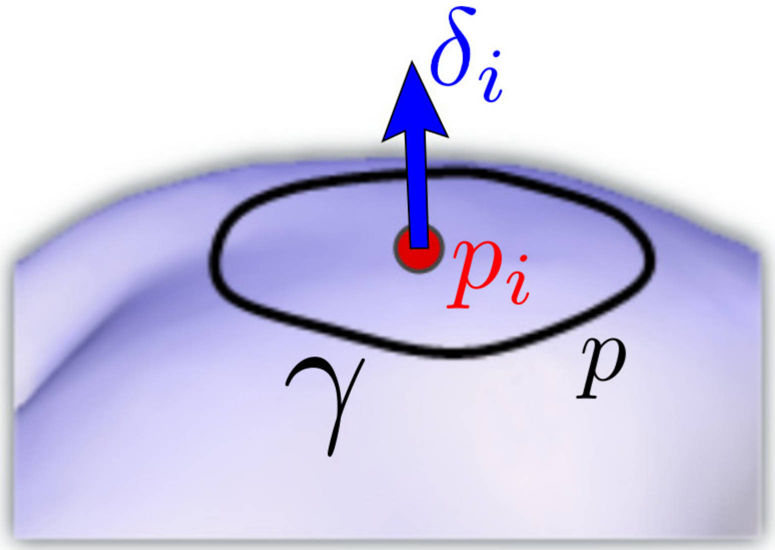
$\mathcal{N}_i$: one ring neighborhood



Note: $\delta_i \simeq \lim_{|\gamma| \to 0} \frac{1}{|\gamma|} \int_{p \in \gamma} (p_i - p) \, \mathrm{d}l(p) = \mathrm{H}(p_i) \, n_i$

γ: small contour around p_i

$\mathrm{H}(p_i)$: Mean curvature at p_i, n_i normal at p_i



$\Rightarrow$ Encode an approximation of the mean curvature

Laplacian of the coordinates: Approximation

Simplest approximation of the laplacian (Graph-Laplacian): $\delta_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} (p_i - p_j)$

(-) *Depends on the connectivity, assume homogeneous and regular geometrical sampling*

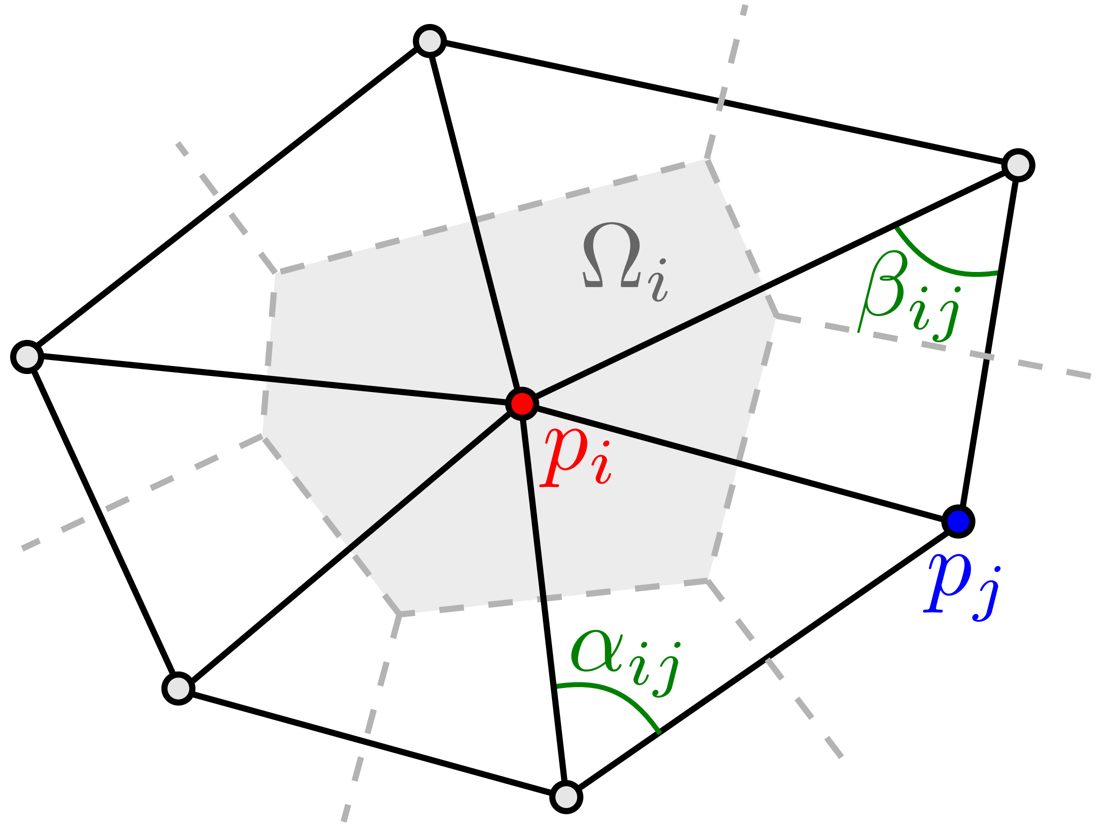
Other possible approximations: $\delta_i = \sum_{j \in \mathcal{N}_i} \omega_{ij} \, (p_i - p_j)$

Ex. Cotangent weights ("Mesh Laplacian")

$$\delta_i = \frac{1}{|\Omega_i|} \sum_{j \in \mathcal{N}_i} \frac{1}{2} (\cot\alpha_{ij} + \cot\beta_{ij}) \, (p_i - p_j)$$

Better geometrical properties: independant of the connectivity

(-) Can become negative



Or using mean-value coordinates

No perfect approximation (satisfying all properties of the continuous Laplacian operator)

Discrete Laplace operators: No free lunch. M. Wardetzky et al. SGP 2007

Laplacian deformation formulation

Given initial position $(p_i)_{i\in[0,N-1]}$ + set of N_c positional constraints c_i, $i \in C$.

Find q_i such that

$\delta(q_i) = \delta(p_i)$: Same differential coordinates

$\forall i \in C, q_i = c_i$: Positional constraints

Formulate using quadratic energy w/r to unknown q

$$E = \sum_{i=0}^{N-1} \|\delta(q_i) - \delta(p_i)\|^2 + \omega \sum_{i\in C} \|q_i - c_i\|^2$$, w: weight associated to positional constraints.

$$E = \sum_{i=0}^{N-1} \left\| q_i - \frac{1}{|\mathcal{N}_i|} \sum_{j\in\mathcal{N}_i} q_j - \delta(p_i) \right\|^2 + \omega \sum_{i\in C} \|q_i - c_i\|^2$$

Laplacian deformation least square formulation

$$E = \sum_{i=0}^{N-1} \left\| q_i - \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} q_j - \delta(p_i) \right\|^2 + \omega \sum_{i \in C} \|q_i - c_i\|^2$$

Finding q minimizing E is similar to solve the least square problem

$$q^\star = arg\,\mathrm{min}_q \ \|\mathrm{M}\, q - b\|^2$$

$q = (q_0, \cdots, q_{N-1})^T$: N unknown

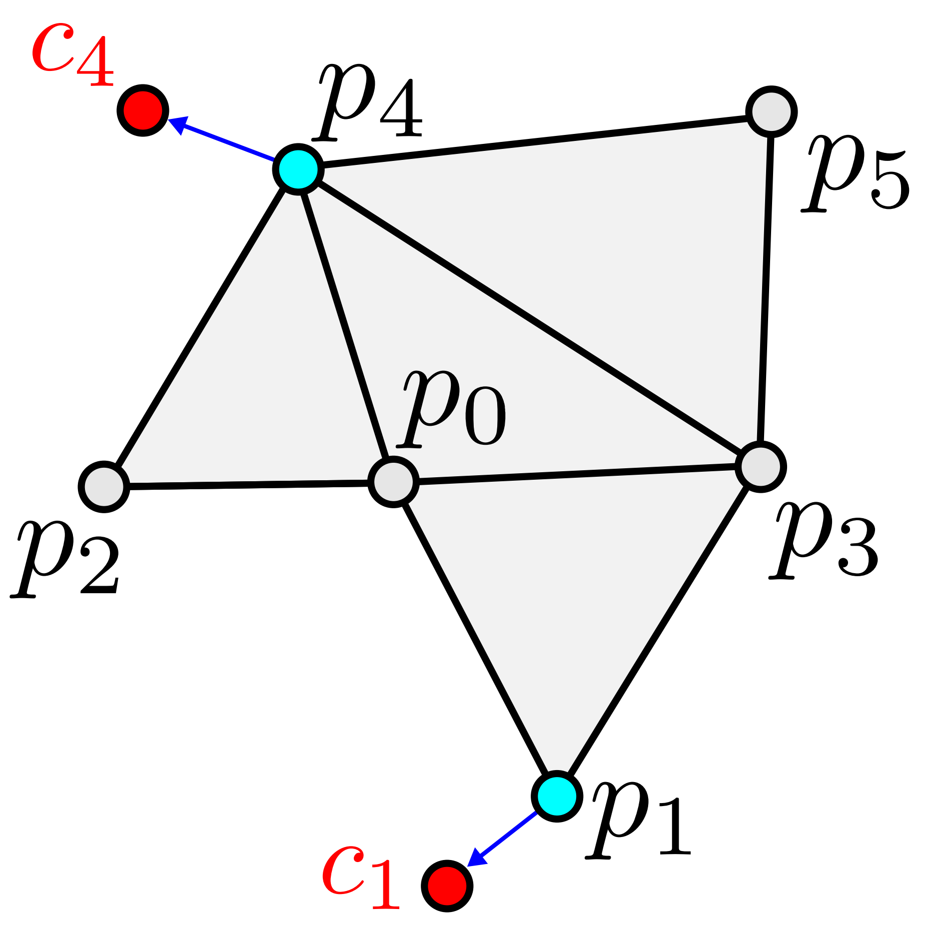
$\mathrm{M} = \left(\frac{\mathrm{D}}{\mathrm{C}}\right)$ with D ($N \times N$) ref. Laplacian weights; C ($N_c \times N$) ref. constraints.

- $\forall i \in [0, N-1]$, $\mathrm{D}_{ii} = 1$, $\mathrm{D}_{ij} = -1/|\mathcal{N}_i|$ if $j \in \mathcal{N}_i$
- $\forall k \in [0, N_c - 1]$, $\mathrm{C}_{ki} = \omega$, with i refering to the vertex index of the kth constraint.

$b = (\delta(p_0), \cdots, \delta(p_{N-1}),\ \omega\, c_0, \cdots, \omega\, c_{N_c-1})^T$

Example of matrix

$$\underbrace{\left(\begin{array}{cccccc} X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \\ \hline X & X & X & X & X & X \\ X & X & X & X & X & X \end{array}\right)}_{\mathrm{M}} \underbrace{\begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{pmatrix}}_{q} = \underbrace{\left(\begin{array}{c} \delta_0 \\ \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \delta_5 \\ \hline \omega\, c_1 \\ \omega\, c_2 \end{array}\right)}_{b}$$



Q. Fill the values of the matrix for this case

Note: Only 2 soft positional constraints on q_1 and q_4

Laplacian deformation least square solution

Need to minimize $\|\mathrm{M}\,q - b\|^2$

Similar to solve the linear system $(\mathrm{M}^T\,\mathrm{M})\,q = \mathrm{M}^T\,b$ (*Normal equations*)

$\mathrm{M}^T\mathrm{M}$ is sparse (diagonally dominant)

Can use standard sparse solver (CG, Jacobi, Gauss Seidel, etc.)

Reminder: Never try to compute the inverse matrix ! Only solve the linear system.

Special solvers (QR factorization) doesn't even need to compute the normal equations

System can be solved independently on x, y, z coordinates. Same matrix M

$$\mathrm{M}\,q_x = b_x,\quad \mathrm{M}\,q_y = b_y,\quad \mathrm{M}\,q_z = b_z$$

Changing constraints position c_i only impact rhs b

$\rightarrow$ M can be factorized once, and reused during interactive deformation

Changing constrained vertex impact M

$\rightarrow$ need to be re-factorized

Solving a dense least square problem

```cpp
#include <iostream>
#include <Eigen/Dense>

int main()
{
    // Build matrix
    Eigen::MatrixXf M;
    // M.resize(N,N)
    // Fill the matrix ... M(i,j) = value
    M = Eigen::MatrixXf::Random(3, 3); // example

    //QR Decomposition (different types are proposed in Eigen)
    auto solver = M.colPivHouseholderQr();

    // Build RHS
    Eigen::VectorXf b;
    // b.resize(N); b(i) = value ...
    b = Eigen::VectorXf::Random(3);

    // Solve linear system
    Eigen::VectorXf x = solver.solve(b);

    std::cout << x << std::endl;
    std::cout <<"Check solution : "<< (M*x-b).norm() << std::endl;

    return 0;
}
```

Solving a sparse least square problem

```
#include <iostream>
#include <Eigen/Sparse>

int main()
{
    // Build matrix
    Eigen::SparseMatrix<float> M;

    // Resize and reserve space to store coefficients
    //   ex. N rows, 10 coeffs per rows
    int N = 50;
    M.resize(N,N);
    M.reserve(Eigen::VectorXi::Constant(N,10));
    // Fill the matrix ... M.coeffRef(i,j) = value
    for(int k=0; k<N; ++k) {
        M.coeffRef(k,k)=1.0f; M.coeffRef(k,(k+1)%N)=-0.3f;
    }

    // Compress the matrix representation
    M.makeCompressed();

    // Factorization for a solver (here Conj. Gradient)
    Eigen::LeastSquaresConjugateGradient< Eigen::SparseMatrix<float> > solver;
    solver.compute(M);
    solver.setTolerance(1e-6f);

    // Build RHS
    Eigen::VectorXf b;
    b = Eigen::VectorXf::Random(N);

    // Solve linear system
    Eigen::VectorXf x = solver.solve(b);
    // or solver.solveWithGuess(b, x0)
    //   faster with good guess
    std::cout <<"Check solution : "
       << (M*x-b).norm() << std::endl;
    std::cout <<"N iterations : "
       << solver.iterations() << std::endl;
    return 0;
}
```

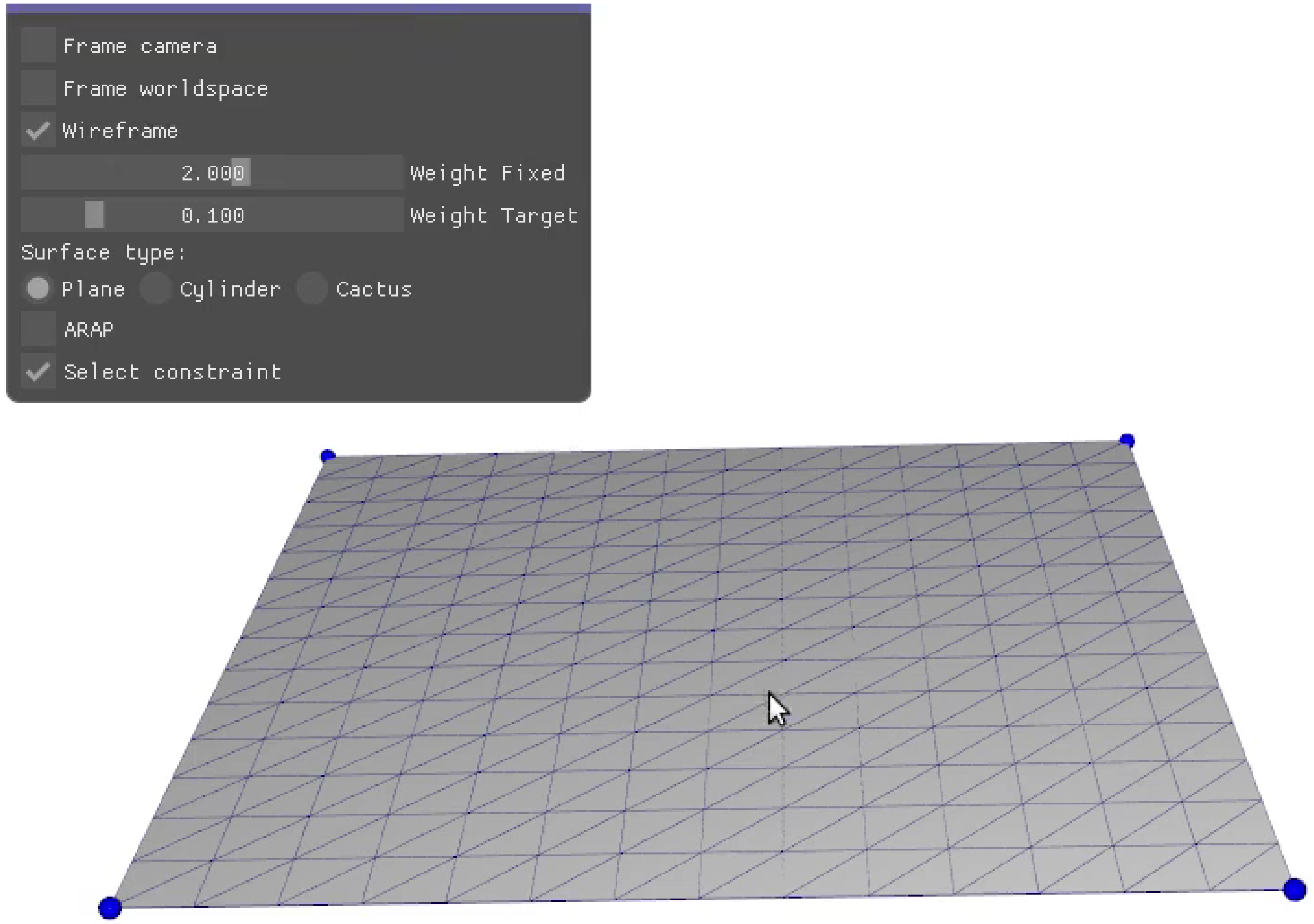
Laplacian deformation: Results

Rem. Discrete solution of the Poisson equation with Dirichlet boundaries

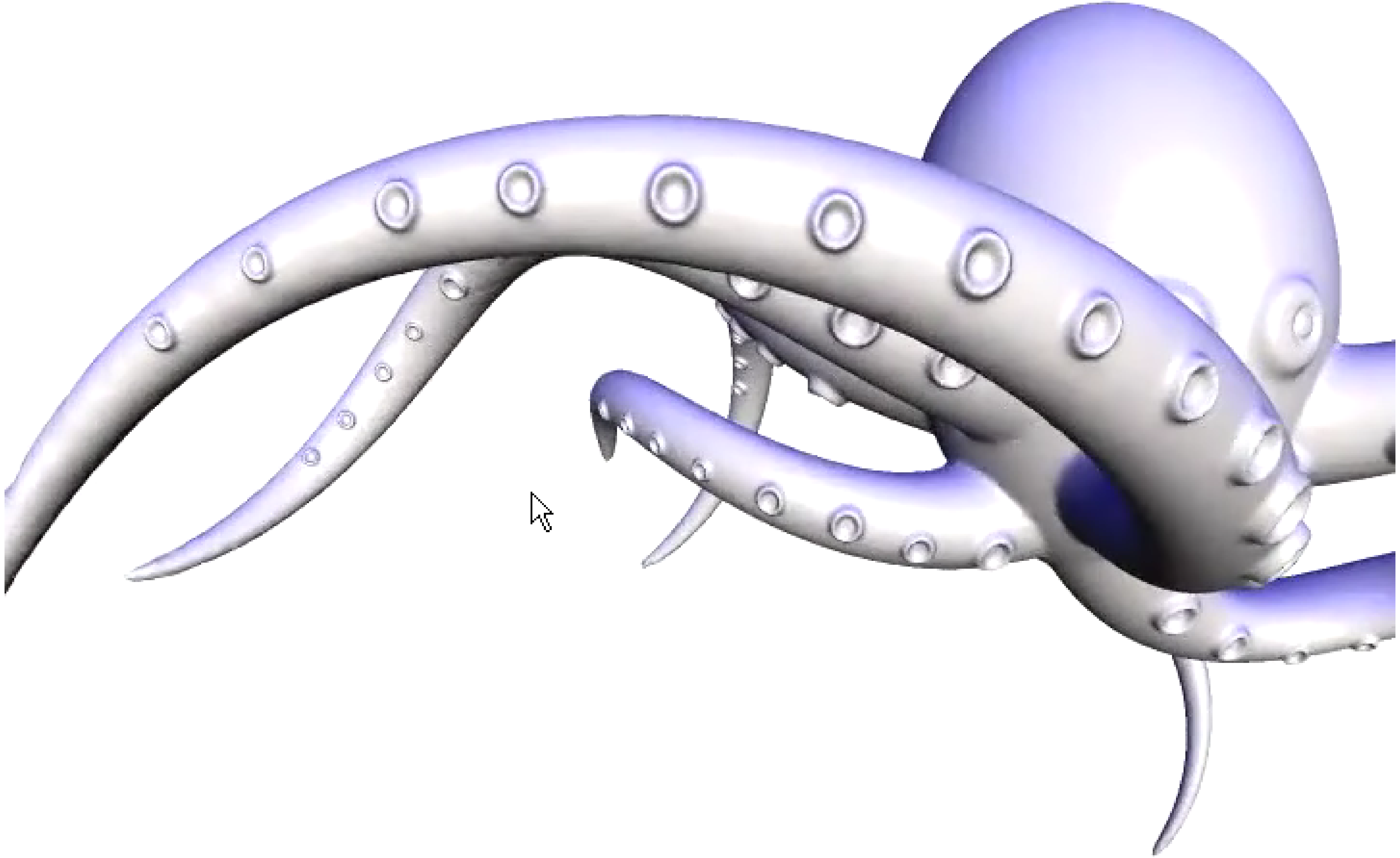
$$\triangle f = \varphi$$
$$f = f^0 \text{ on } \partial\Omega$$

In the case of a plane: Laplacian equation $\triangle f = 0$

- Solution are membranes
  - *Surface of minimal mean curvature*



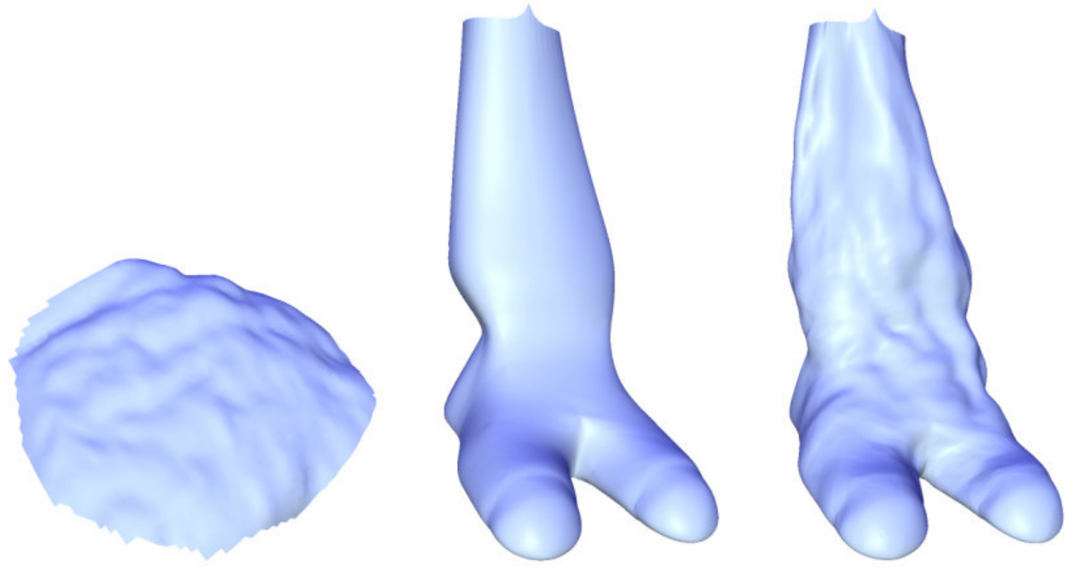
Laplacian deformation: Results

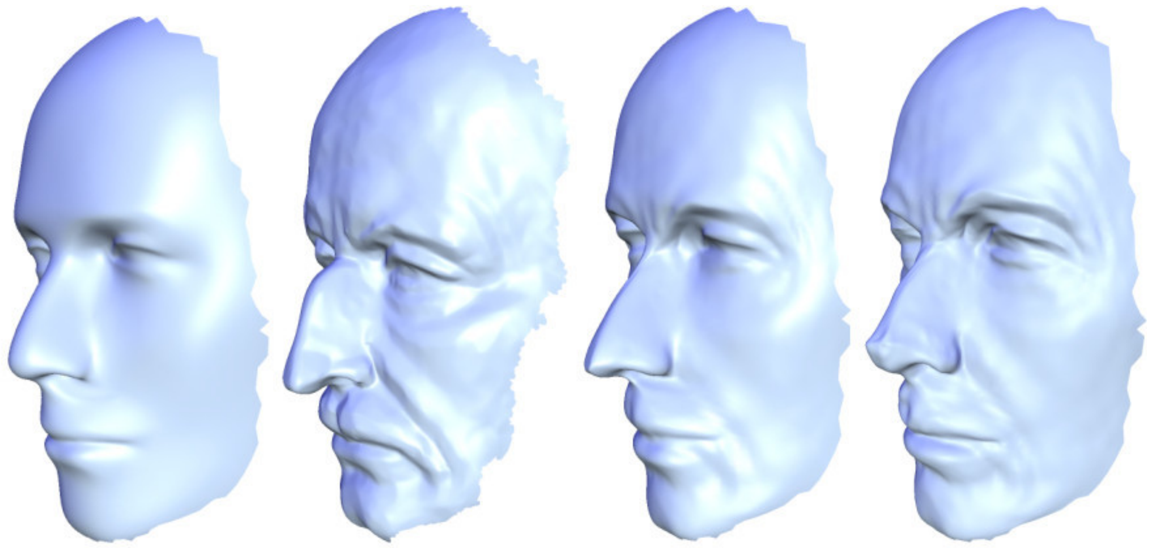


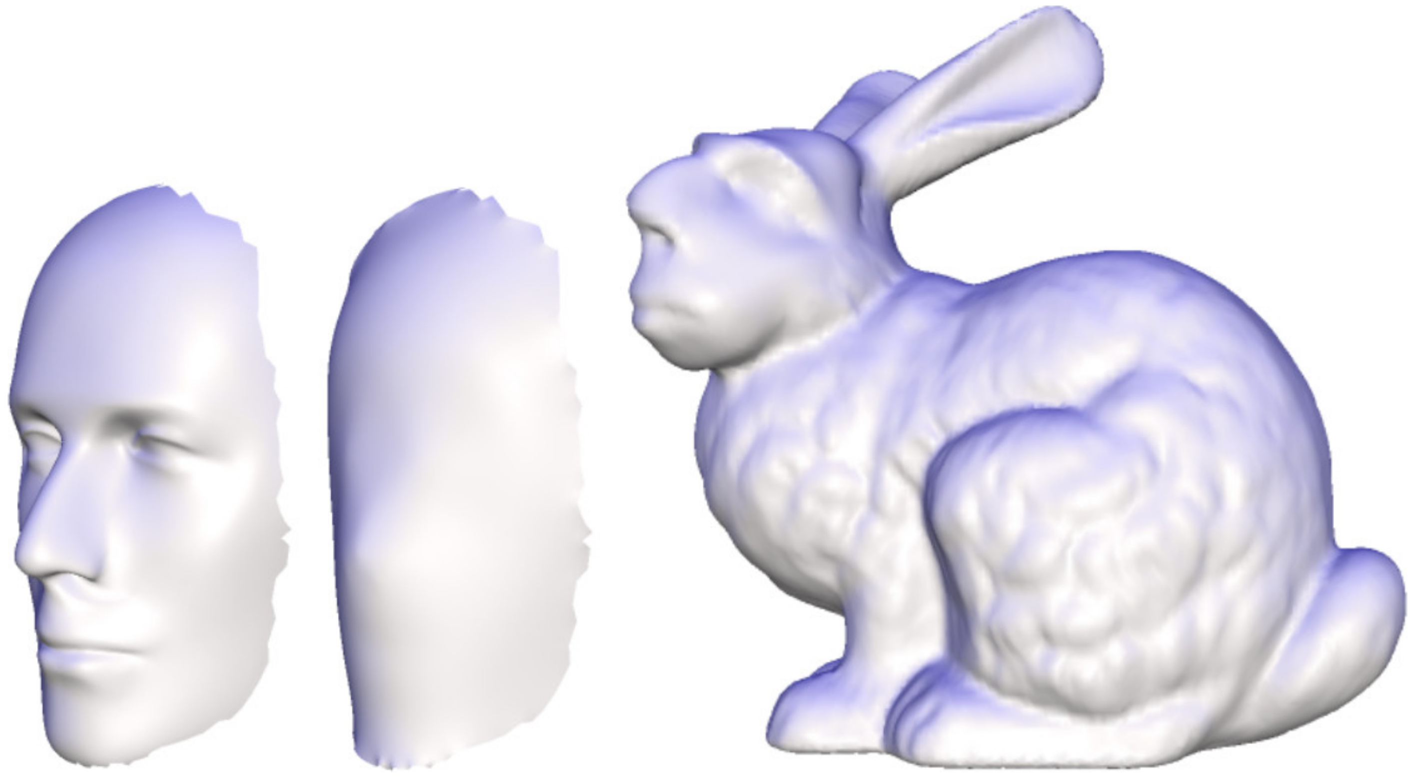
Laplacian Surface Editing. O. Sorkine et al. SGP 2004

Extension: Detail transfert

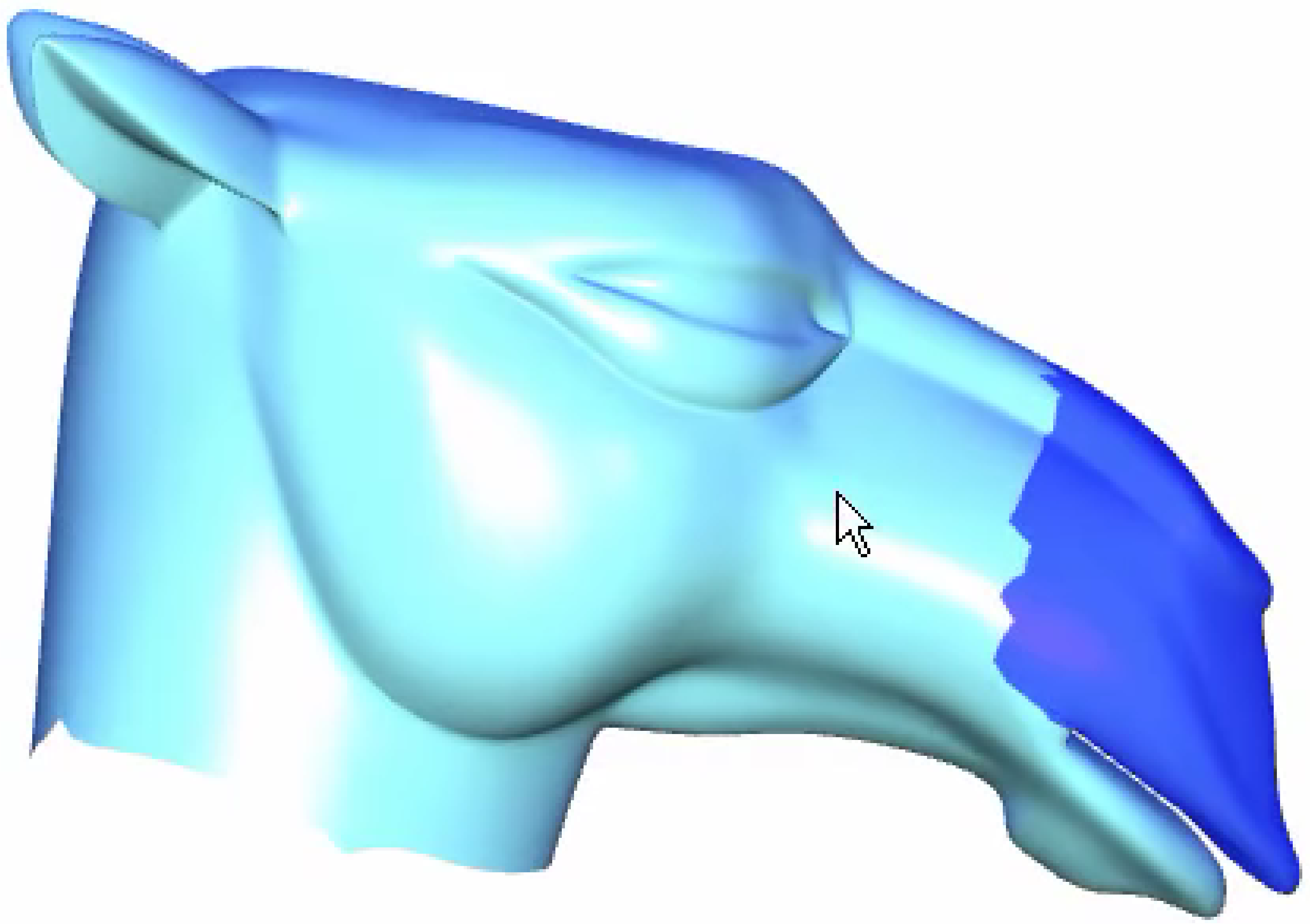
- Use laplacian from another surface to transfert details
  - Requires a per-vertex correspondance
    - *User defined anchors*







Laplacian editing: Sketching silhouette deformation

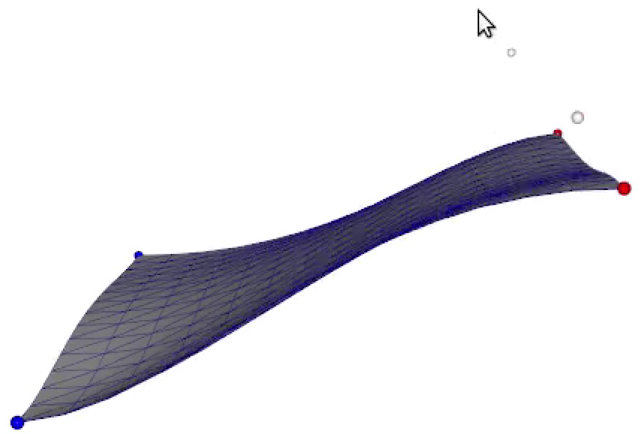


A Sketch-Based Interface for Detail-Preserving Mesh Editing. A. Nealen et al. SIGGRAPH 2005

As Rigid As Possible (ARAP)

The Laplacian formulation tries to preserve the surface orientation.

While we could expect surface rotation



No ARAP

Idea: Include rotation transformation R in the formulation

$$E = \sum_{i=0}^{N-1} \|\delta(q_i) - R(q_i)\,\delta(p_i)\|^2 + \omega \sum_{i \in C} \|q_i - c_i\|^2$$

$R(q_i)$: best rotation at point q_i

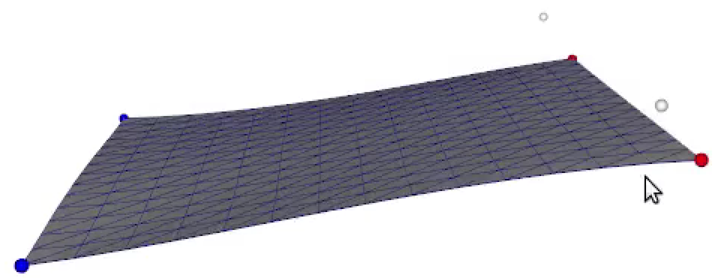
Finding the best rotation at a given point

1- Look at surrounding edges at the point q_i/p_i

$$e_j = q_j - q_i, e_j^0 = p_j - p_i$$

2- Compute covariance matrix $\sigma = \sum_j e_j\,(e_j^0)^T$

3- Compute rotation using polar decomposition: $R = polar(\sigma)$



ARAP

As Rigid As Possible (ARAP) Solution

Minimize the following term

$$E = \sum_{i=0}^{N-1} \|\delta(q_i) - R(q_i)\,\delta(p_i)\|^2 + \omega \sum_{i \in C} \|q_i - c_i\|^2$$

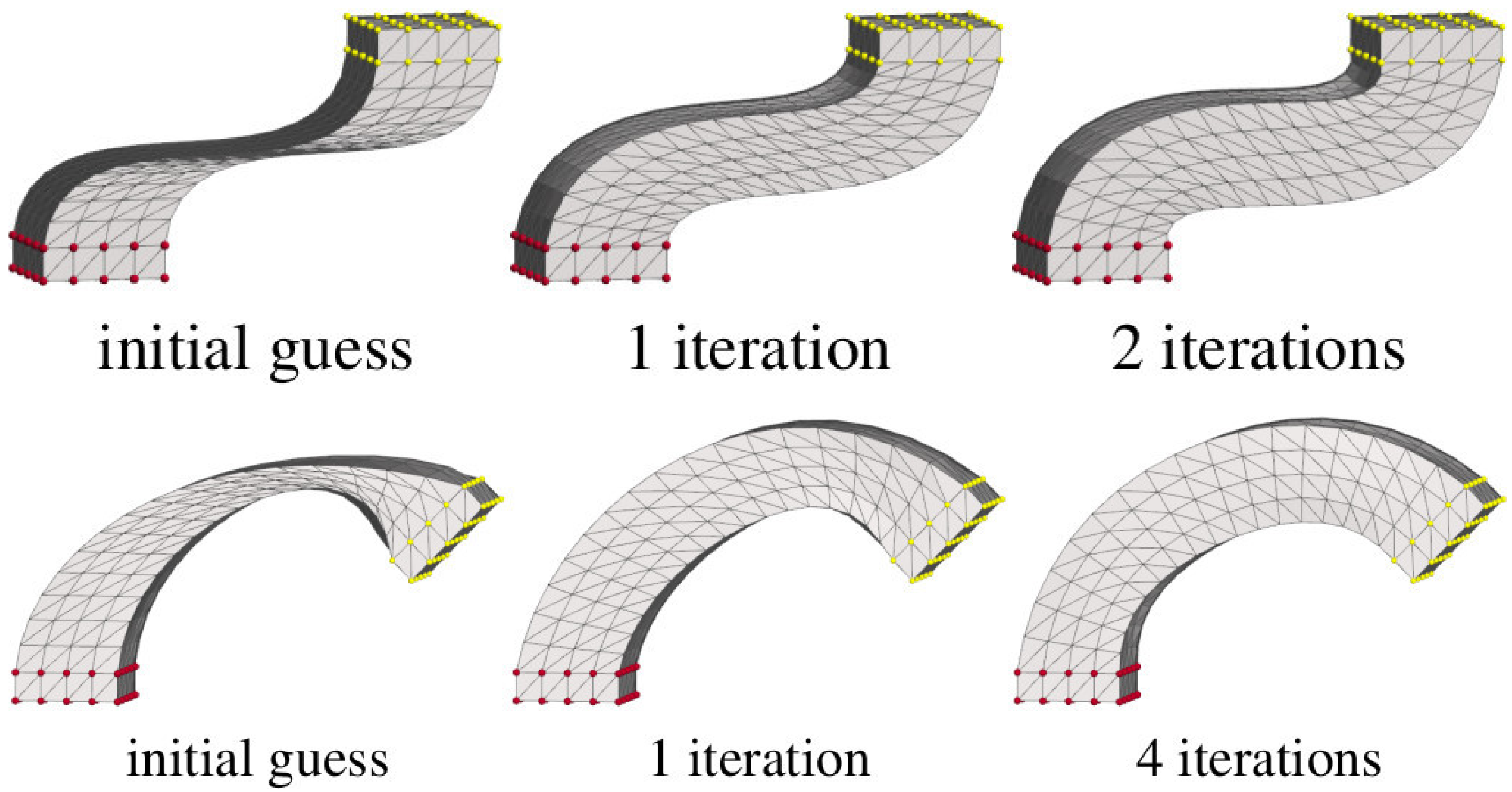
Problem: $R(q_i)$ is a non linear function of q_i

E is non quadratic, cannot use least square.

Idea: Interleave and iterate on these two steps

Find optimal rotation $R(q_i)$ with fixed q_i

Find q_i optimizing E with fixed $R(q_i)$

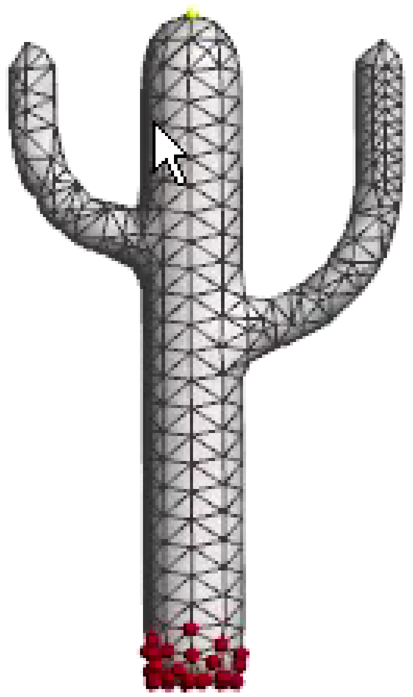


Gain: Surface rotates during deformation

Limit: Slightly more costly

Iterations + polar decomposition

As Rigid As Possible: Results



link:assets/arap.pdf[As-Rigid-As-Possible Surface Modeling. O. Sorkine, M. Alexa. SGP 2007]

Other approaches

Gradient representation

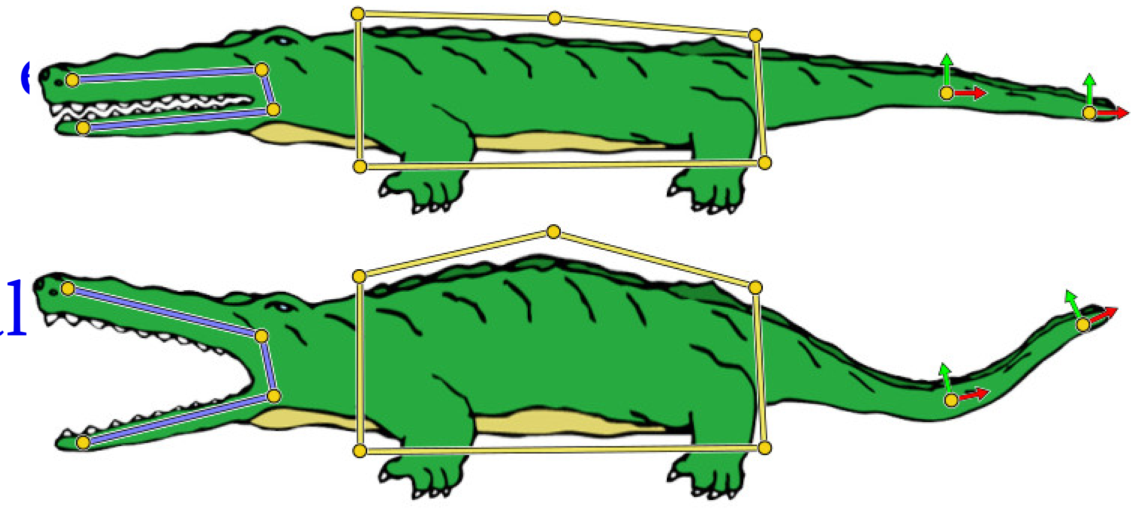
Mesh editing with Poisson-based gradient field manipulation. Y.Yu et al. SIGGRAPH 2004

Bi-Laplacien weights

Bounded Biharmonic Weights for Real-Time Deform. A. Jacobson

Local frame representation

Linear Rotation-invariant Coordinates for Meshes. Y. Lipman et al



Deformation propagation

Harmonic Guidance for Surface Deformation. R. Zayer et al. EG 2005

State of the art

On Linear Variational Surface Deformation Methods. M. Botsch and O. Sorkine. TVCG 2008