Geometric Optimization

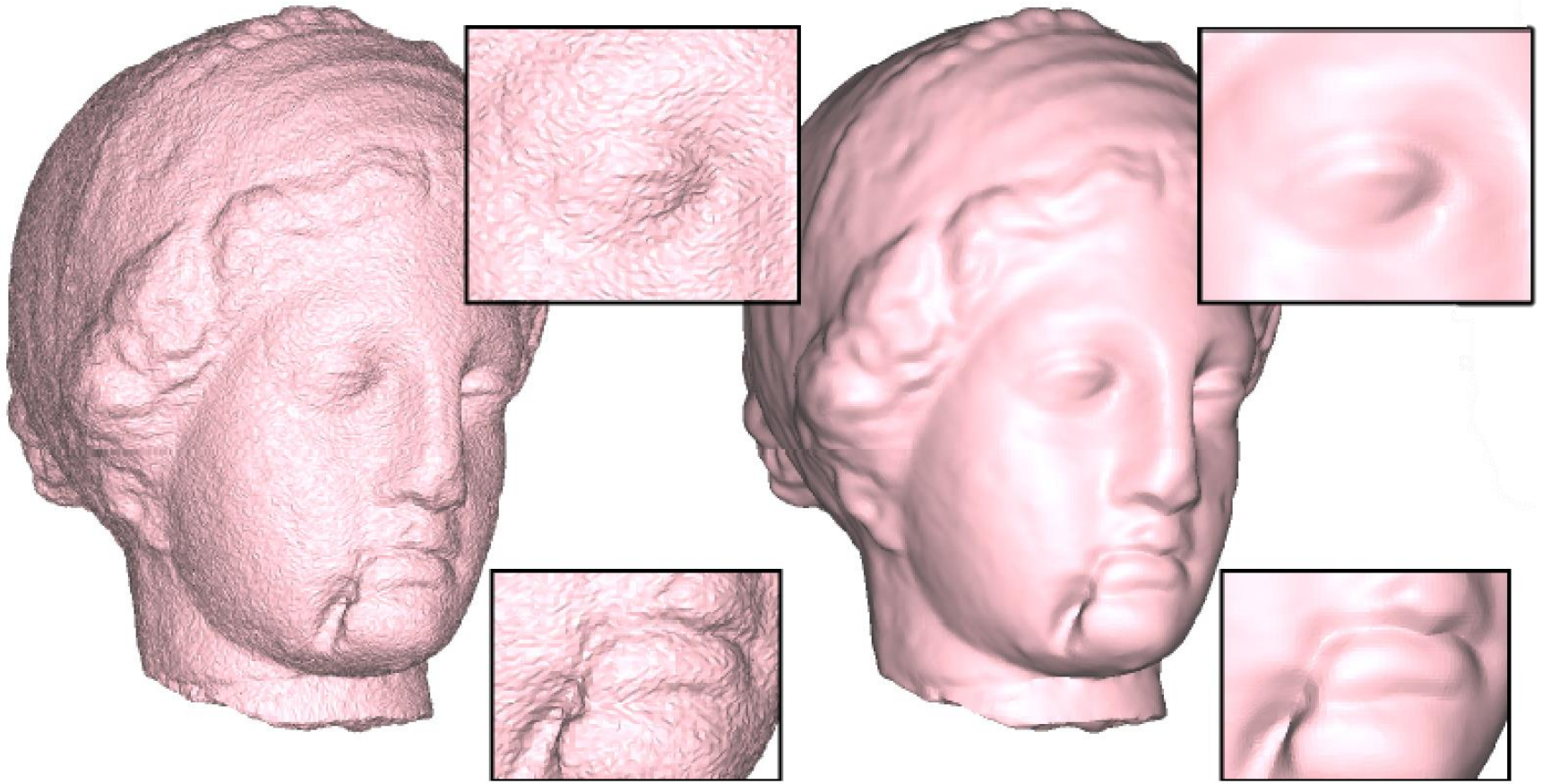
Jian-Ping Su

May 19, 2020

USTC 2020 Spring Digital Geometry Processing

Geometric Optimization

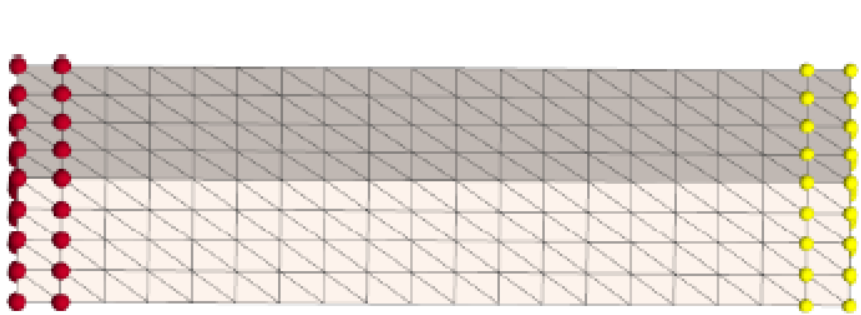
(i) Mesh denoising

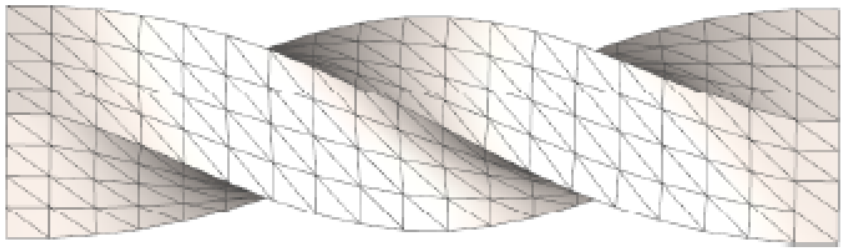


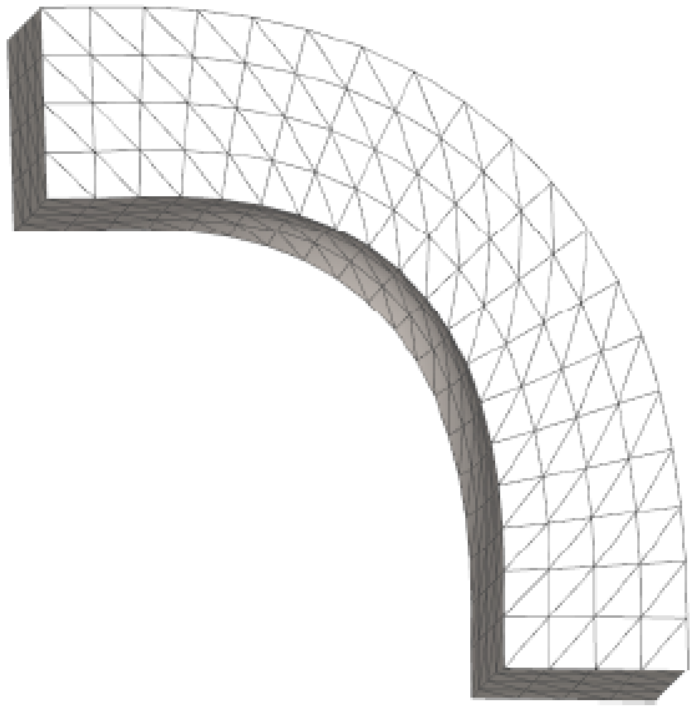
Geometric Optimization

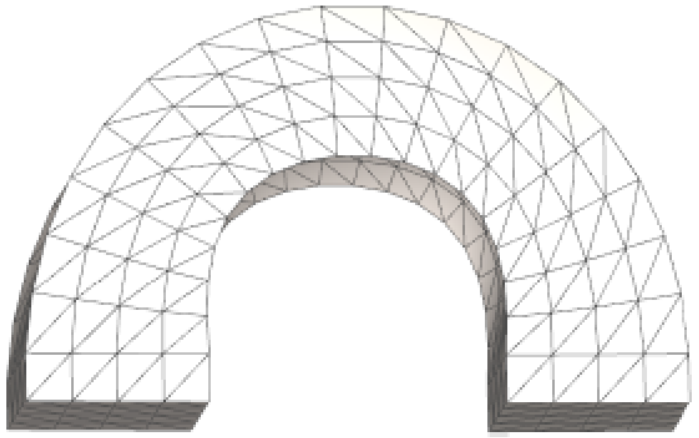
(i) Mesh denoising

(ii) Mesh Deformation







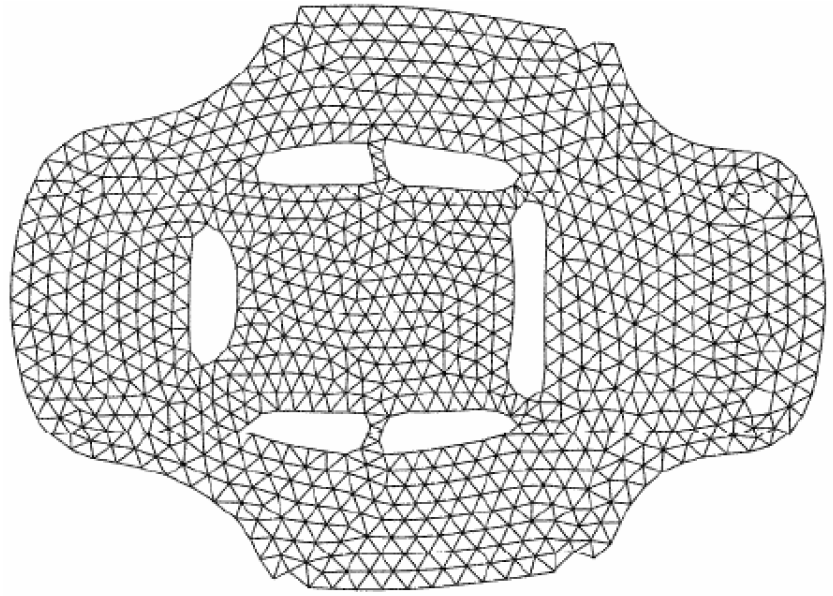


Geometric Optimization

(i) Mesh denoising

(ii) Mesh Deformation

(iii) Mesh parameterization



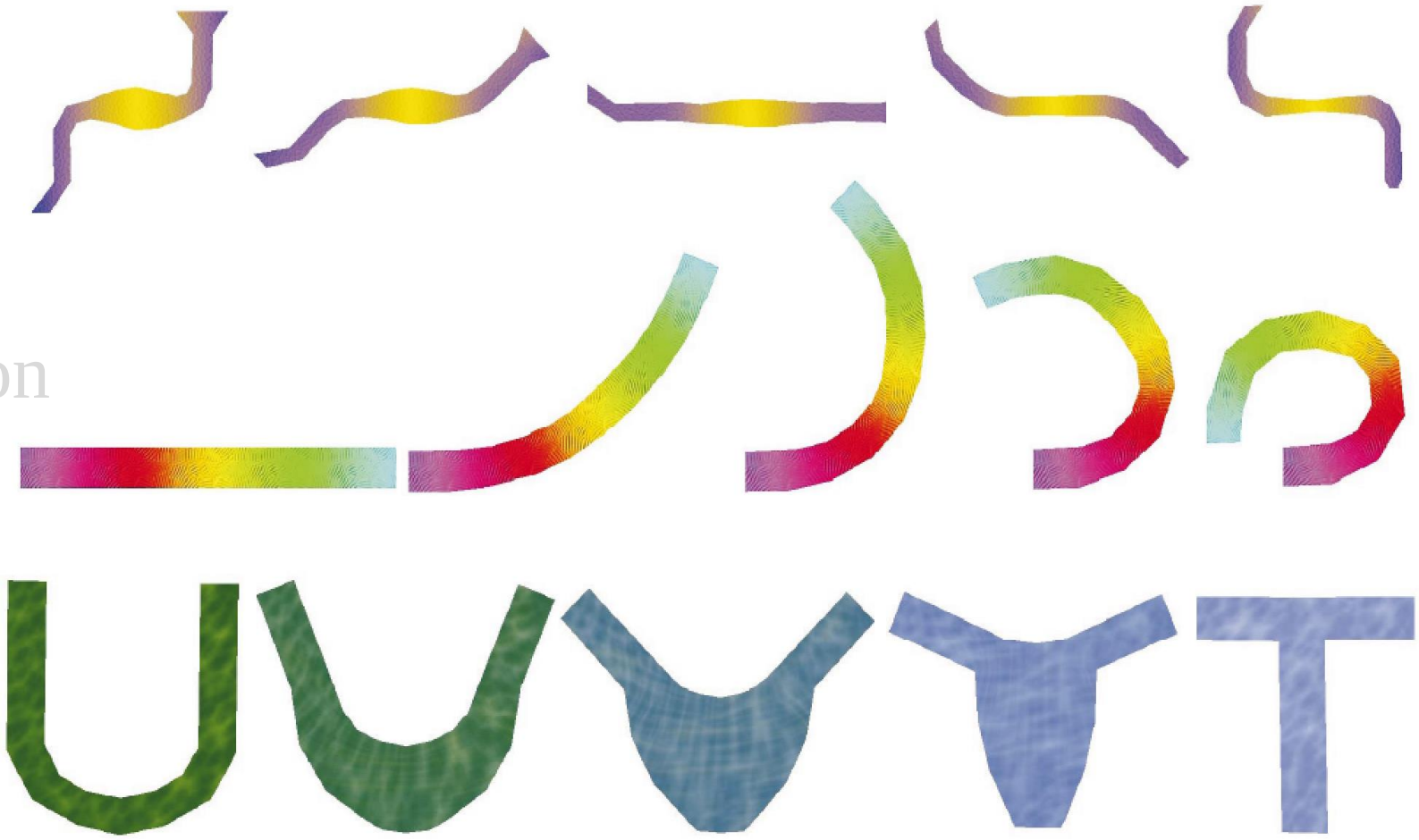
Geometric Optimization

(i) Mesh denoising

(ii) Mesh Deformation

(iii) Mesh parameterization

(iv) Mesh Interpolation



Geometric Optimization

(i) Mesh denoising

(ii) Mesh Deformation

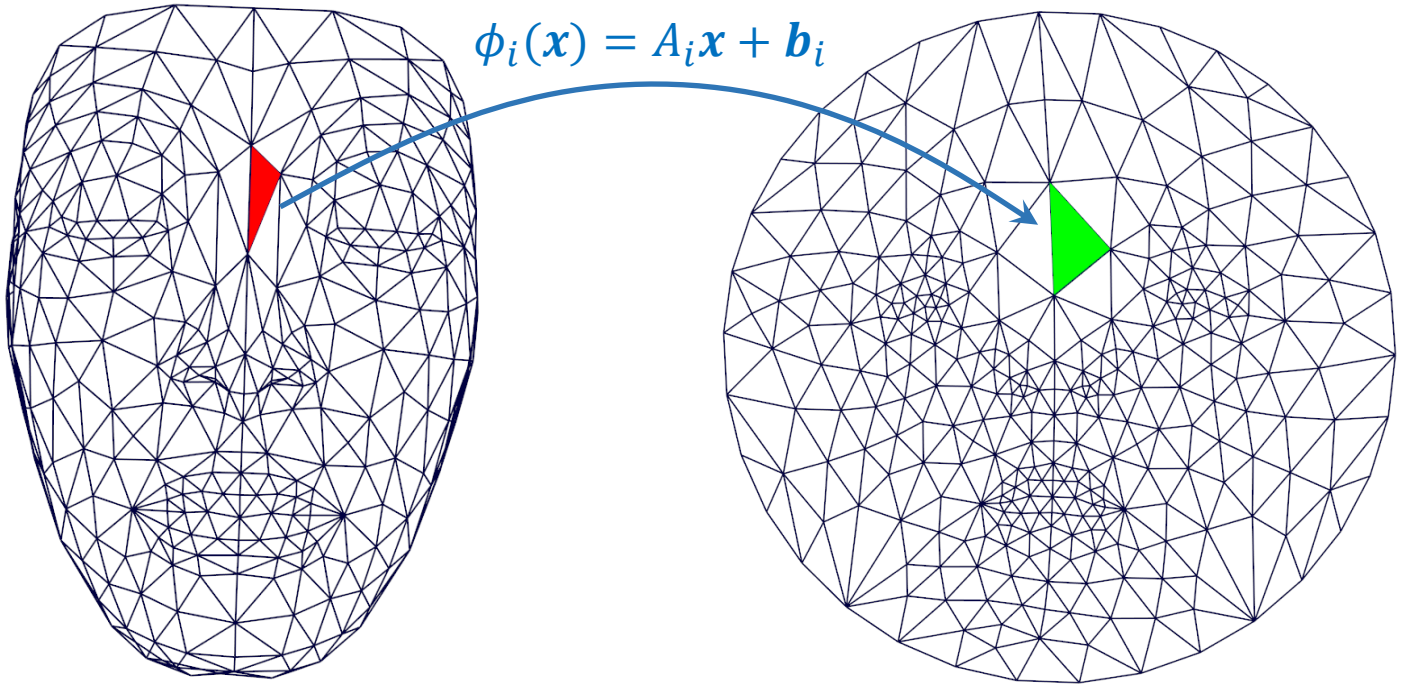
(iii) Mesh parameterization

(iv) Mesh Interpolation

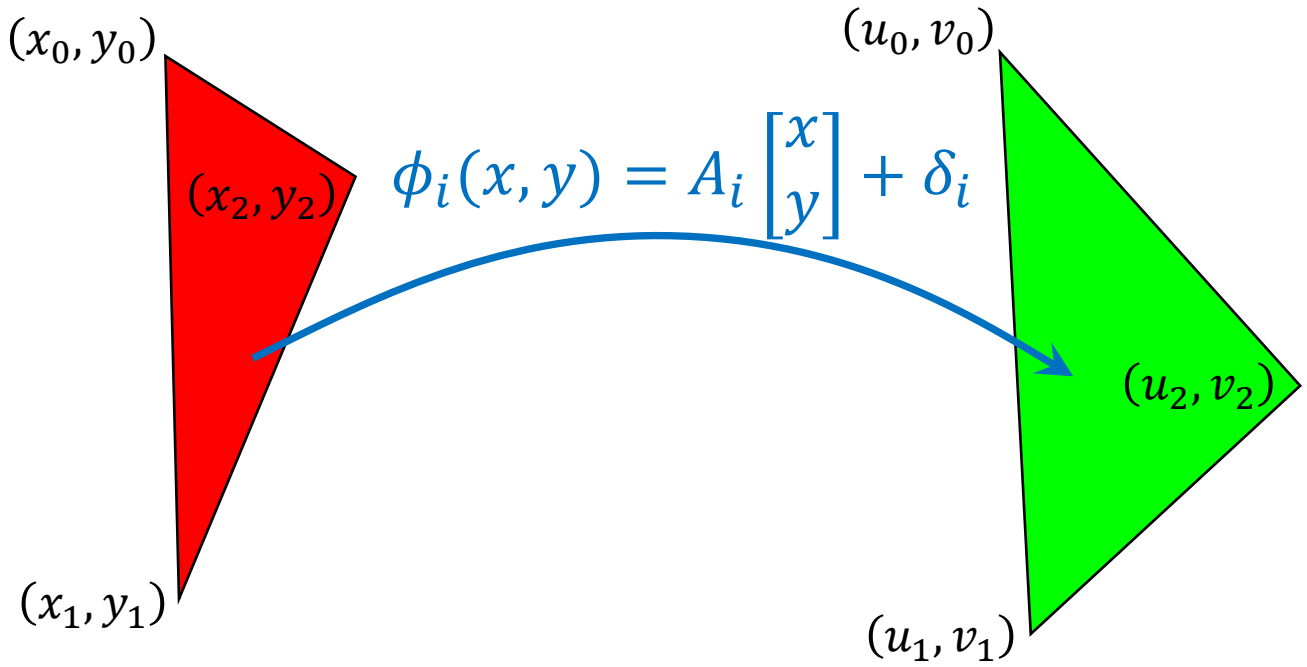
(v) Mesh Simplification

Preliminary

Jacobian Matrix



Jacobian Matrix

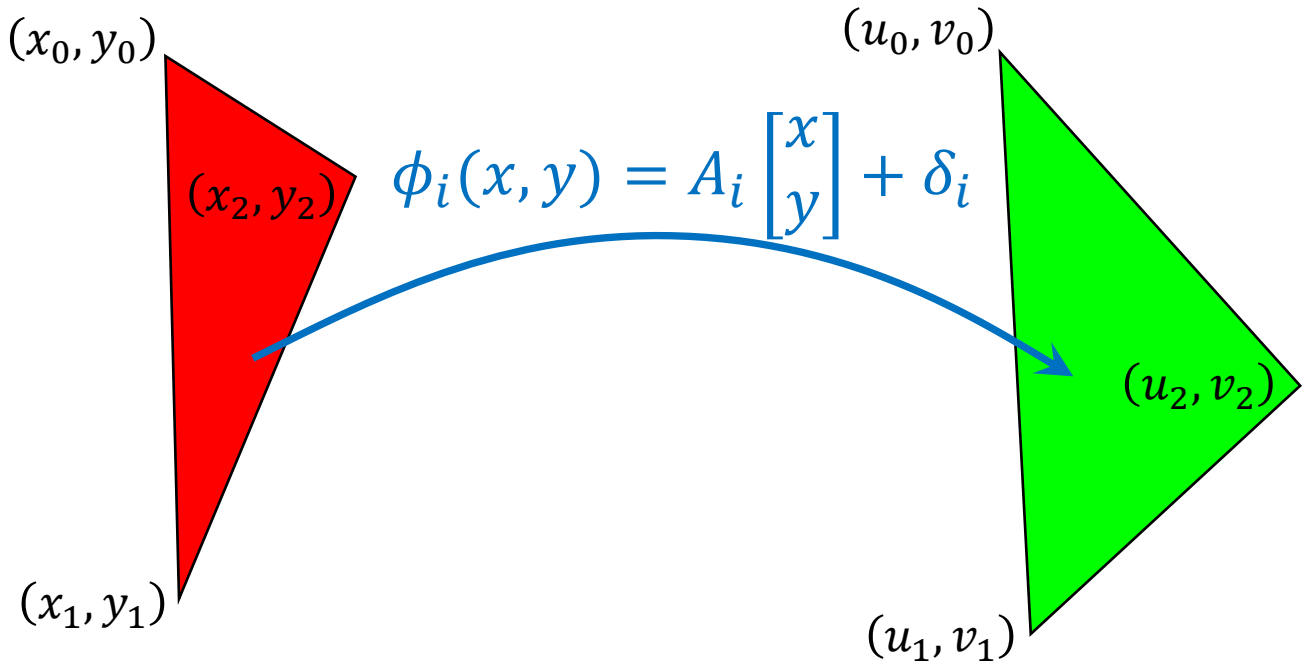


$$\begin{bmatrix} u \\ v \end{bmatrix} = A_i \begin{bmatrix} x \\ y \end{bmatrix} + \delta_i$$

$$J_i = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = A_i$$

$$A_i = \begin{bmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{bmatrix} \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}^{-1}$$

Jacobian Matrix



$$\begin{bmatrix} u \\ v \end{bmatrix} = A_i \begin{bmatrix} x \\ y \end{bmatrix} + \delta_i$$

$$J_i = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = A_i$$

$$J_i = \begin{bmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{bmatrix} \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}^{-1}$$

Jacobian Matrix

$$J_i = \begin{bmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{bmatrix} \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}^{-1}$$

$$J_i^T = \begin{bmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{bmatrix}^{-1} \begin{bmatrix} u_1 - u_0 & v_1 - v_0 \\ u_2 - u_0 & v_2 - v_0 \end{bmatrix}$$

$$J_i^T = \frac{1}{2|t_j|} \begin{bmatrix} y_2 - y_0 & y_0 - y_1 \\ x_0 - x_2 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} u_1 - u_0 & v_1 - v_0 \\ u_2 - u_0 & v_2 - v_0 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \frac{1}{2|t_i|} \begin{bmatrix} y_1 - y_2 & y_2 - y_0 & y_0 - y_1 & & & \\ x_2 - x_1 & x_0 - x_2 & x_1 - x_0 & & & \\ & & & y_1 - y_2 & y_2 - y_0 & y_0 - y_1 \\ & & & x_2 - x_1 & x_0 - x_2 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

Singular Value

The arithmetic square root of non-negative eigenvalues of J^TJ

$$J^TJ = \begin{bmatrix} a & c \\ b & d \end{bmatrix}\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a^2+c^2 & ab+cd \\ ab+cd & b^2+d^2 \end{bmatrix}$$

$$a' = \frac{a+d}{2},\ c' = \frac{a-d}{2},\ b' = \frac{c-b}{2},\ d' = \frac{c+b}{2}$$

$$\Sigma = \sqrt{a'^2+b'^2} + \sqrt{c'^2+d'^2},\quad \sigma = \left|\sqrt{a'^2+b'^2} - \sqrt{c'^2+d'^2}\right|$$

$$\|J\|_F^2 = a^2+b^2+c^2+d^2 = tr\left(J^TJ\right) = \Sigma^2+\sigma^2$$

Distortion Measure

- Dirichlet

$$f_{\mathrm{DRCH}}(\mathrm{x}) = \sum_i \|J_i\|_F^2 |t_i| = \sum_i \left(\Sigma_i^2 + \sigma_i^2\right)|t_i|$$

- ARAP

$$f_{\mathrm{ARAP}}(\mathrm{x}) = \sum_i \left\|J_i - R\left(J_i(\mathrm{x})\right)\right\|_F^2 |t_i| = \sum_i \left((\Sigma_i - 1)^2 + (\sigma_i - 1)^2\right)|t_i|$$

- Symmetric Dirichlet

$$f_{\mathrm{ISO}}(\mathrm{x}) = \sum_i \left(\|J_i\|_F^2 + \|J_i\|_F^{-2}\right)|t_i| = \sum_i \left(\Sigma_i^2 + \Sigma_i^{-2} + \sigma_i^2 + \sigma_i^{-2}\right)|t_i|$$

- Conformal

$$f_{\mathrm{CONF}}(\mathrm{x}) = \sum_i \left(\frac{\Sigma_i}{\sigma_i}\right)^2 |t_i|$$

Laplace Matrix

Laplace Matrix is Hessian of $f_{\mathrm{DRCH}}(\mathrm{x})$

$$f_{\mathrm{DRCH}}(\mathrm{x}) = \sum_i \|J_i\|_F^2 |t_i| = \sum_i \left(\Sigma_i^2 + \sigma_i^2\right)|t_i|$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \frac{1}{2|t_i|} \begin{bmatrix} y_1 - y_2 & y_2 - y_0 & y_0 - y_1 & & & \\ x_2 - x_1 & x_0 - x_2 & x_1 - x_0 & & & \\ & & & y_1 - y_2 & y_2 - y_0 & y_0 - y_1 \\ & & & x_2 - x_1 & x_0 - x_2 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

$$J = T\mathrm{x} \qquad f_{\mathrm{DRCH}}(\mathrm{x}) = \mathrm{x}^{\mathrm{T}} T^{\mathrm{T}} T\mathrm{x} \qquad L = T^{\mathrm{T}} T$$

Problem

$$\min_{\mathrm{x}} f(\mathrm{x}) = \sum_{i} |t_i| \mathcal{D}\big(J_i(\mathrm{x})\big)$$

1.Initial point: x_0， $n = 0$

2.Descent direction: $\min_{p} f(\mathrm{x}_n + p)$

3.Step size: $\min_{\alpha} f(\mathrm{x}_n + \alpha p)$

4.Update: $\mathrm{x}_{n+1} = \mathrm{x}_n + \alpha p$， $n = n + 1$

Method

$$\min_p f(\mathrm{x}_n + p)$$

$$f(\mathrm{x}_n + p) \approx f(\mathrm{x}_n) + \nabla f(\mathrm{x}_n)^T p + \frac{1}{2} p^T H p$$

$$Hp = -\nabla f(\mathrm{x}_n)$$

- First-order methods build descent steps by preconditioning the gradient with a fixed proxy matrix, which often suffer from slower convergence as lacking of higher-order information.
- Newton-type methods uses the energy Hessian, $\nabla^2 f(\mathrm{x})$, to form a proxy matrix, which can achieve the most rapid convergence but require the costly assembly, factorization and backsolve of new linear systems per iterate.

First-order method

Paper List

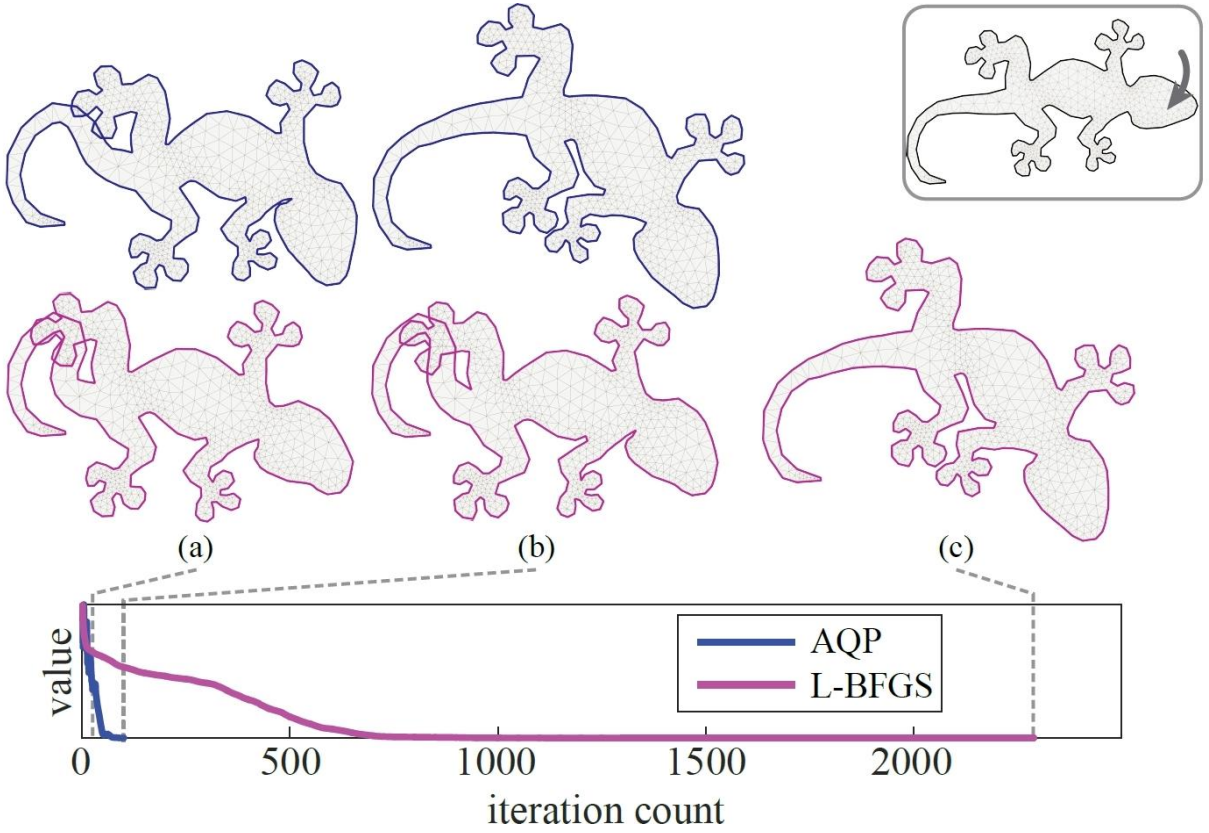
- Accelerated Quadratic Proxy for Geometric Optimization
- Scalable Locally Injective Mappings
- Blended Cured Quasi-Newton for Distortion Optimization

Accelerated Quadratic Proxy for Geometric Optimization

Shahar Kovalsky

Meirav Galun

Yaron Lipman



Motivation

ill-conditioning dominated by a Laplacian-like term in energy.

locally approximating the energy with a function whose Hessian is Laplacian.

$$f(\mathrm{x}) = h(\mathrm{x}) + g(\mathrm{x})$$

$$h(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T H\mathrm{x}$$

$$f(x_{n-1} + p) = h(x_{n-1} + p) + g(x_{n-1} + p)$$

$$f(x_{n-1} + p) \approx h(x_{n-1} + p) + g(x_{n-1}) + \nabla g(x_{n-1})^T p$$

$$H(x_{n-1} + p) + \nabla g(x_{n-1}) = 0$$

$$Hp = -Hx_{n-1} - \nabla g(x_{n-1}) = -\nabla f(x_{n-1})$$

Motivation

correctly balancing the information from its two last iterations.

$$y_n = (1+\theta)x_{n-1} - \theta x_{n-2}$$

$$f(\mathrm{x}) = h(\mathrm{x}) + g(\mathrm{x})$$

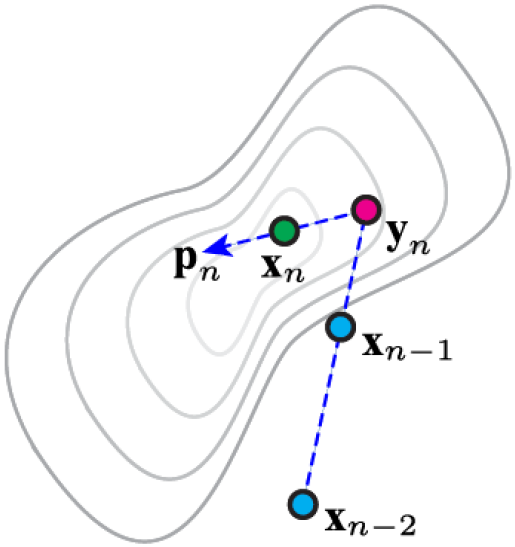
$$h(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T H \mathrm{x}$$

$$f(\mathrm{y}_n + p) = h(\mathrm{y}_n + p) + g(\mathrm{y}_n + p)$$

$$f(\mathrm{y}_n + p) \approx h(\mathrm{y}_n + p) + g(\mathrm{y}_n) + \nabla g(\mathrm{y}_n)^T p$$

$$H(\mathrm{y}_n + p) + \nabla g(\mathrm{y}_n) = 0$$

$$Hp = -H\mathrm{y}_n - \nabla g(\mathrm{y}_n) = -\nabla f(\mathrm{y}_n)$$



Method

$$f(\mathrm{x}) = h(\mathrm{x}) + g(\mathrm{x})$$

$$h(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T H\mathrm{x}$$

$$f(\mathrm{y}_n + p) = h(\mathrm{y}_n + p) + g(\mathrm{y}_n + p)$$

$$f(\mathrm{y}_n + p) \approx h(\mathrm{y}_n + p) + g(\mathrm{y}_n) + \nabla g(\mathrm{y}_n)^T p$$

$$H(\mathrm{y}_n + p) + \nabla g(\mathrm{y}_n) = 0$$

$$Hp = -H\mathrm{y}_n - \nabla g(\mathrm{y}_n) = -\nabla f(\mathrm{y}_n)$$

Algorithm 1: Accelerated Quadratic Proxy (AQP)

Data: feasible initialization $\mathbf{x}$; parameter η

$\mathbf{x}_{-1} = \mathbf{x}_0 = \mathbf{x}$;

$\theta = \frac{1-\sqrt{1/\eta}}{1+\sqrt{1/\eta}}$;

while *not converged* **do**

/* Acceleration */

$\mathbf{y}_n = (1+\theta)\mathbf{x}_{n-1} - \theta\mathbf{x}_{n-2}$;

/* Quadratic proxy minimization */

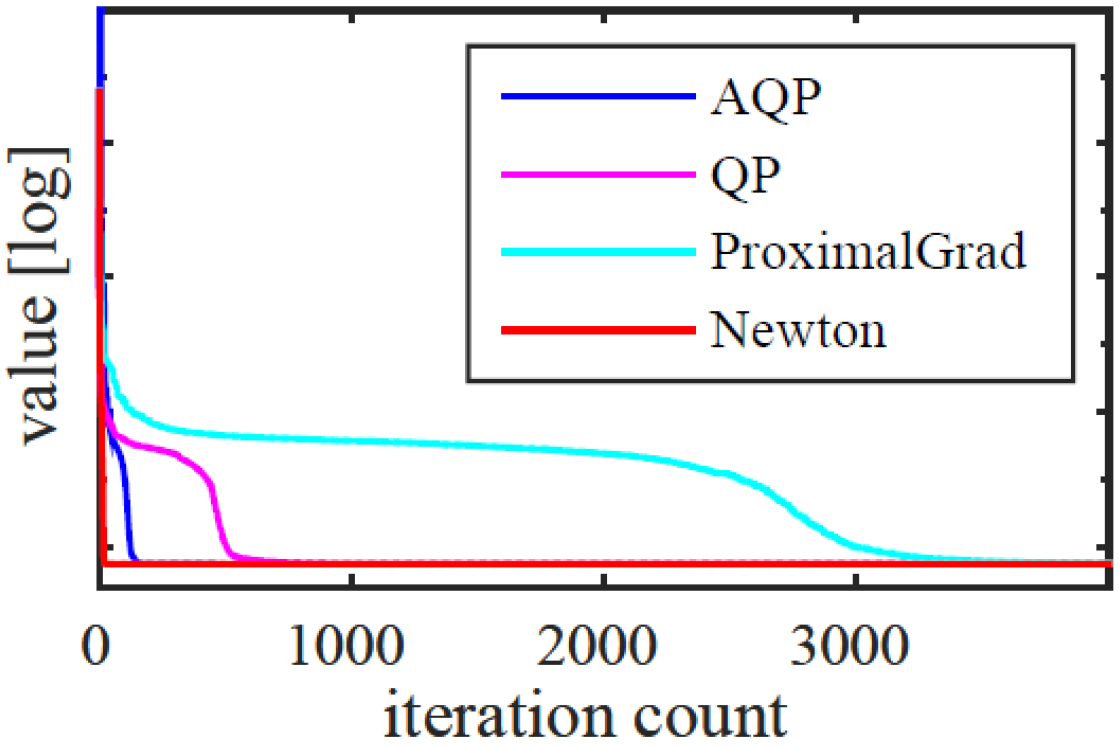
$\mathbf{p}_n = \arg\min_{\mathbf{p}} \; h(\mathbf{y}_n + \mathbf{p}) + g(\mathbf{y}_n) + \nabla g(\mathbf{y}_n)\mathbf{p}$

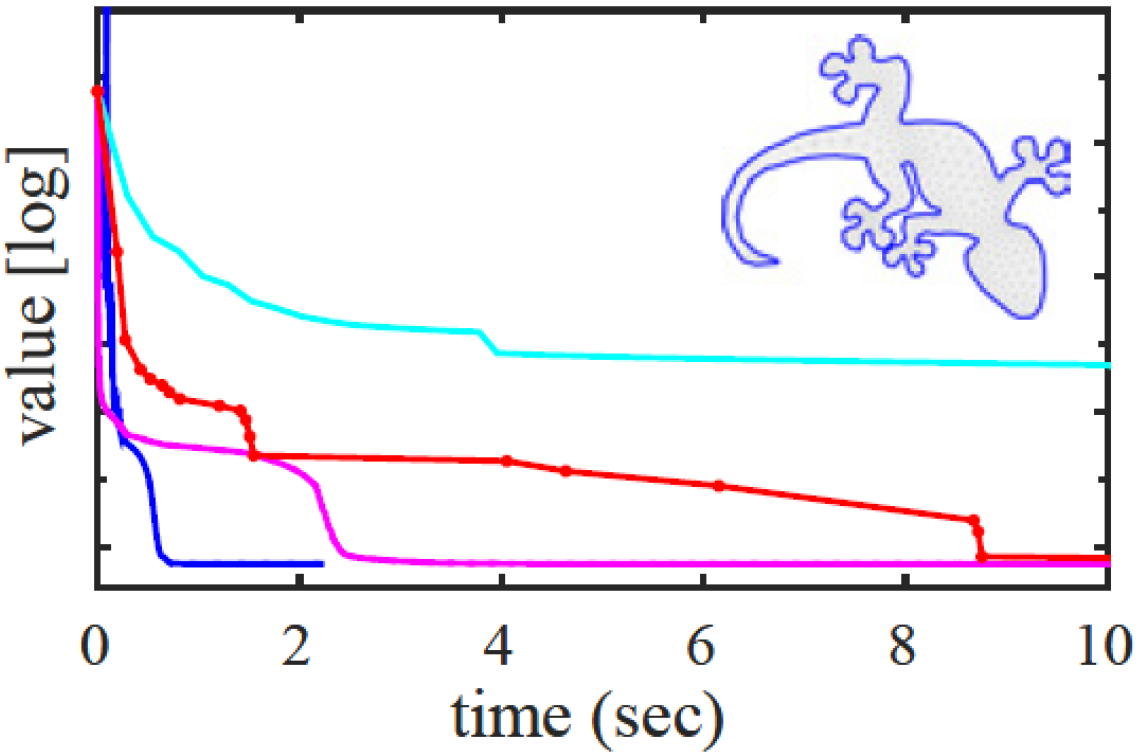
$\text{s.t.} \quad A\mathbf{p} = 0$

/* Line search */

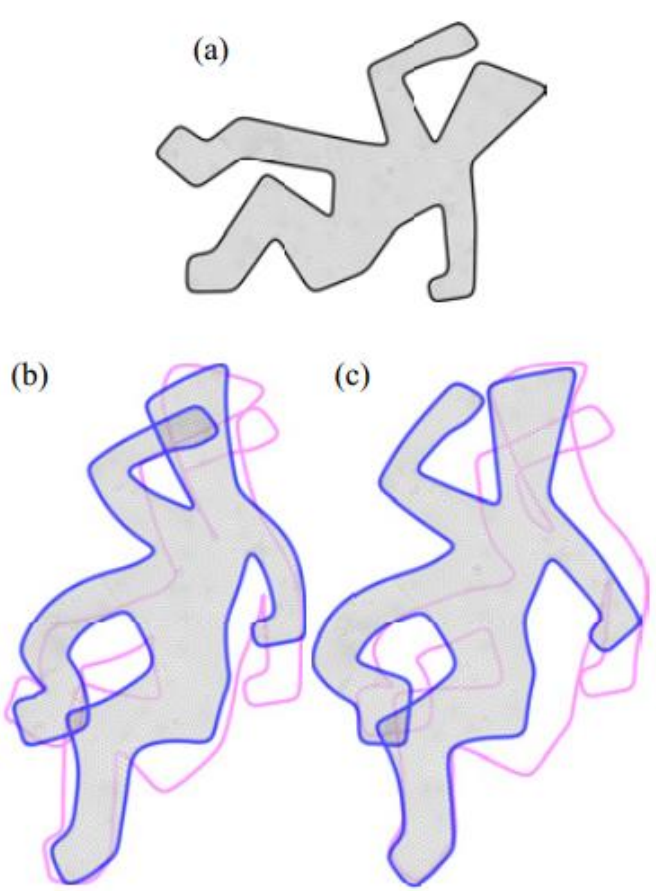
$\mathbf{x}_n = \text{linesearch}_{\,0<t\leq 1} \; f(\mathbf{y}_n + t\mathbf{p}_n)$

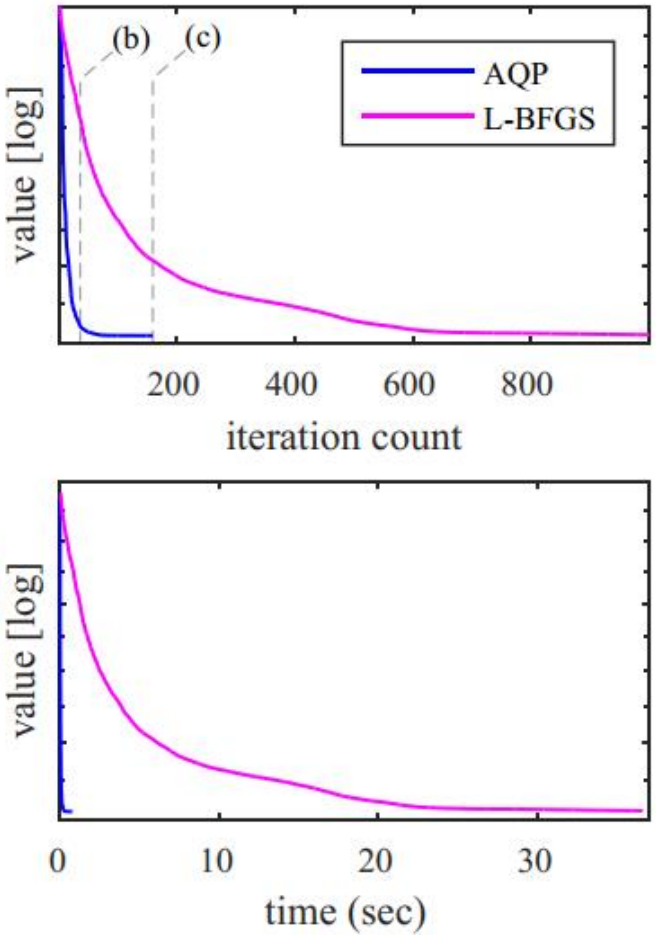
Results


AQP
QP
ProximalGrad
Newton
value [log]
0
1000
2000
3000
iteration count


value [log]
0
2
4
6
8
10
time (sec)

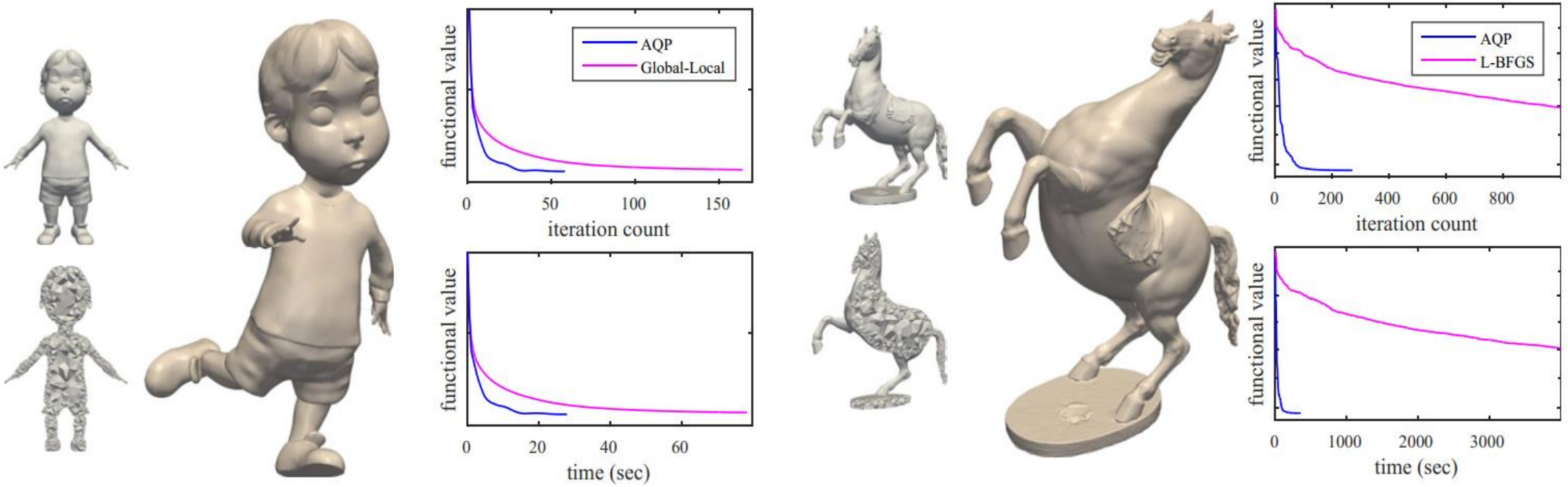
Results


(a)
(b)
(c)


(b)
(c)
AQP
L-BFGS
value [log]
200
400
600
800
iteration count
value [log]
0
10
20
30
time (sec)


(a)

Results


functional value
AQP
Global-Local
0
50
100
150
iteration count
functional value
0
20
40
60
time (sec)
functional value
AQP
L-BFGS
0
200
400
600
800
iteration count
functional value
0
1000
2000
3000
time (sec)

Limitation

$$f(\mathrm{x}) = h(\mathrm{x}) + g(\mathrm{x})$$

$$h(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T H\mathrm{x}$$

$$f_{\mathrm{ARAP}}(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T H\mathrm{x} - \sum_i \|J_i\|_* |t_i| + c_0$$

efficiency of Laplacian approximation for an arbitrary energy

$$f_{\mathrm{ISO}}(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T H\mathrm{x} + \sum_i \|J_i\|_F^{-2} |t_i|$$

$$f_{\mathrm{CONF}}(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T H\mathrm{x} + \sum_i \|J_i\|_F^2 \left(\frac{1}{{\sigma_i}^2} - 1\right) |t_i|$$

Video

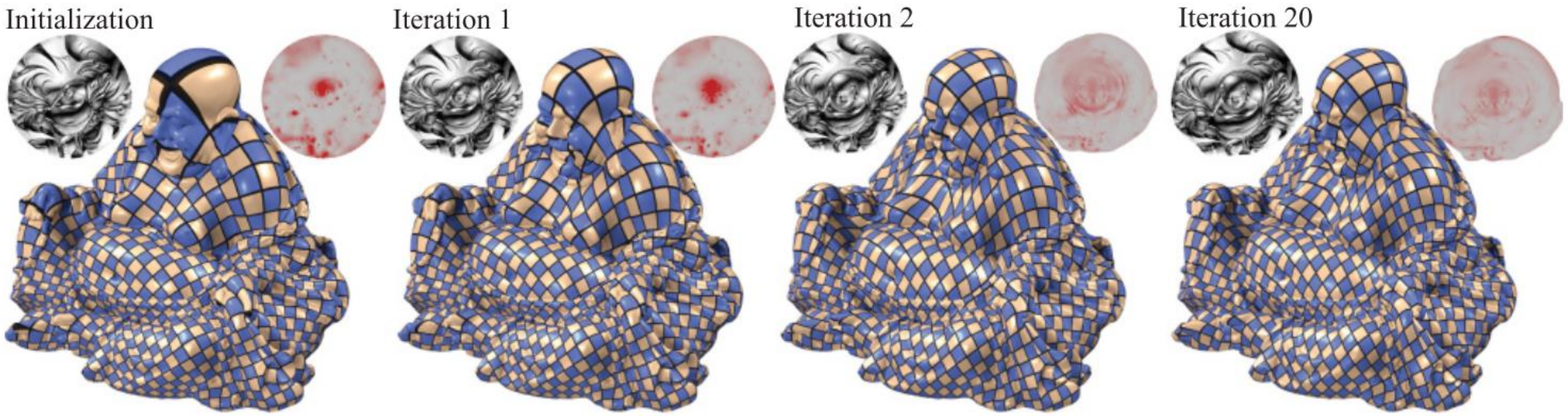
Scalable Locally Injective Mappings

Michael Rabinovich

Roi Poranne

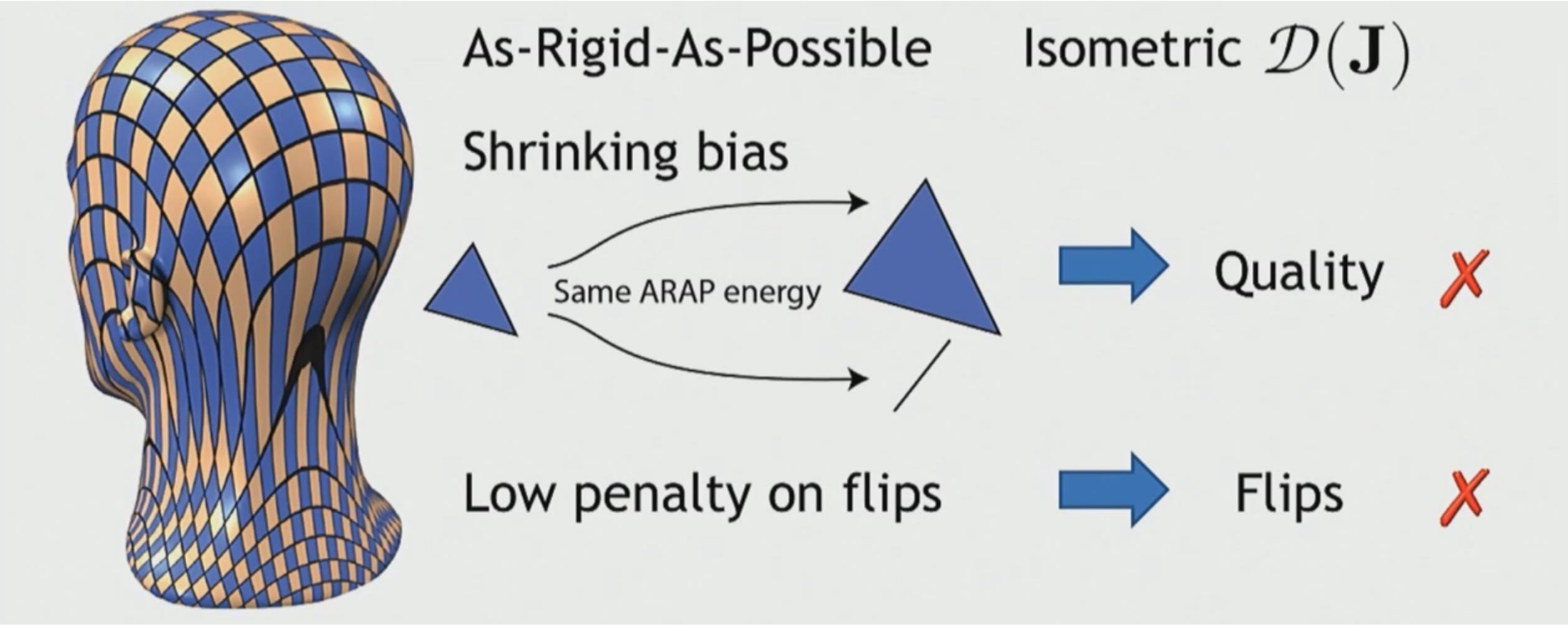
Daniele Panozzo

Olga Sorkine-Hornung



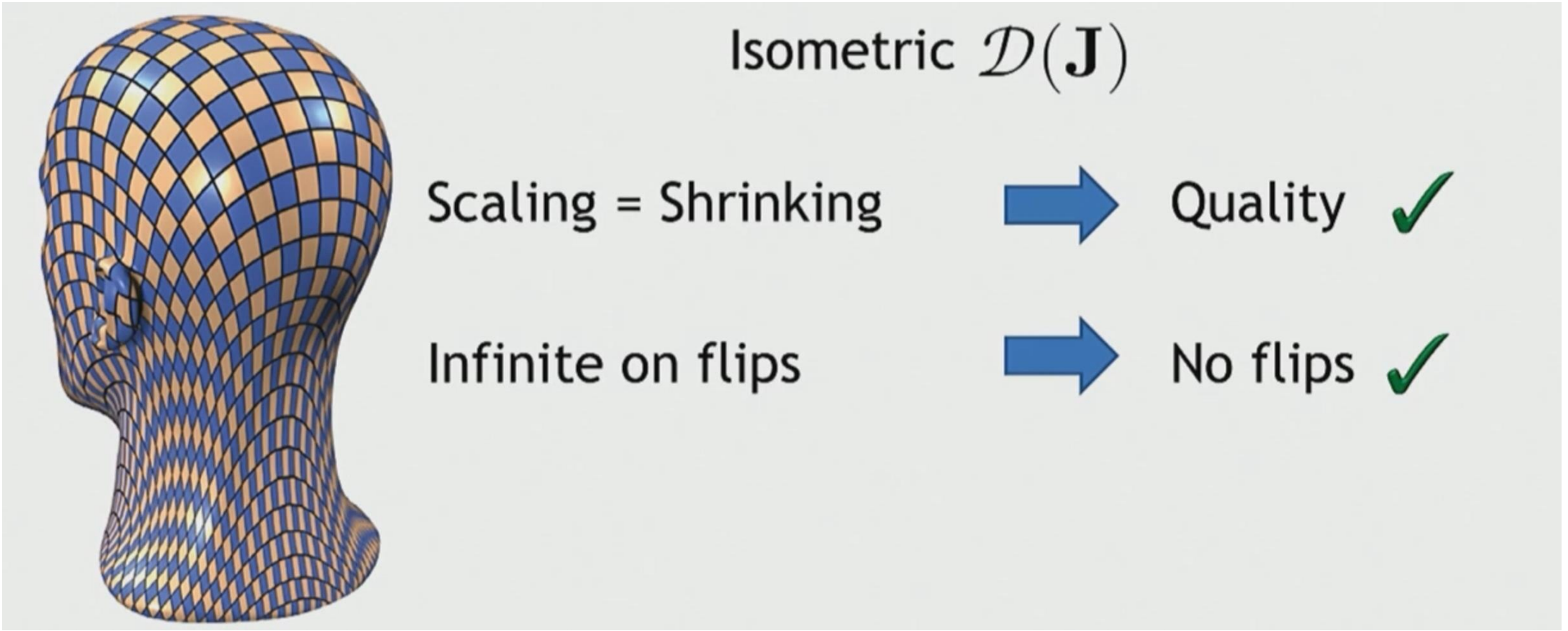
Motivation

$$f_{\mathrm{ARAP}}(\mathrm{x}) = \sum_i \left((\Sigma_i - 1)^2 + (\sigma_i - 1)^2\right)|t_i|$$



Motivation

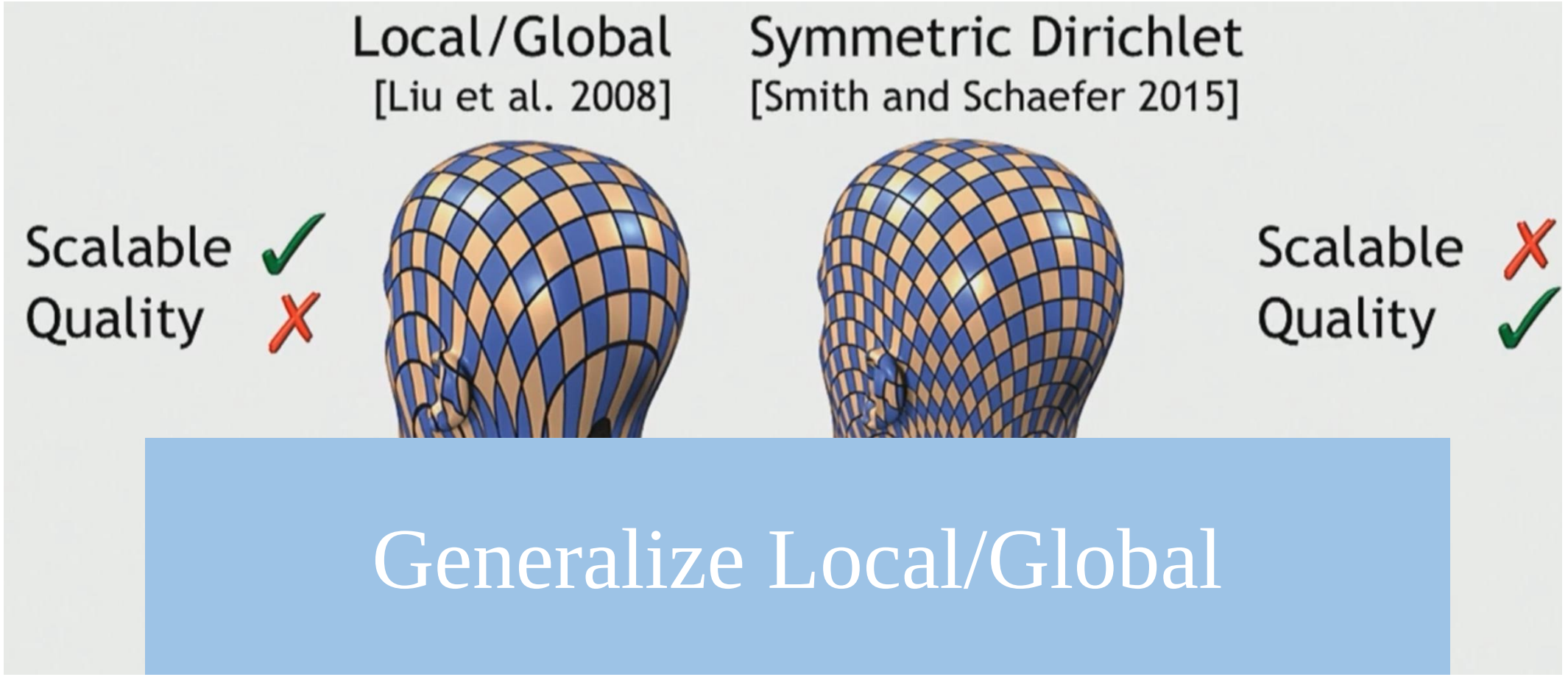
$$f_{\mathrm{ISO}}(\mathrm{x}) = \sum_i \left(\Sigma_i^2 + \Sigma_i^{-2} + \sigma_i^2 + \sigma_i^{-2}\right)|t_i|$$

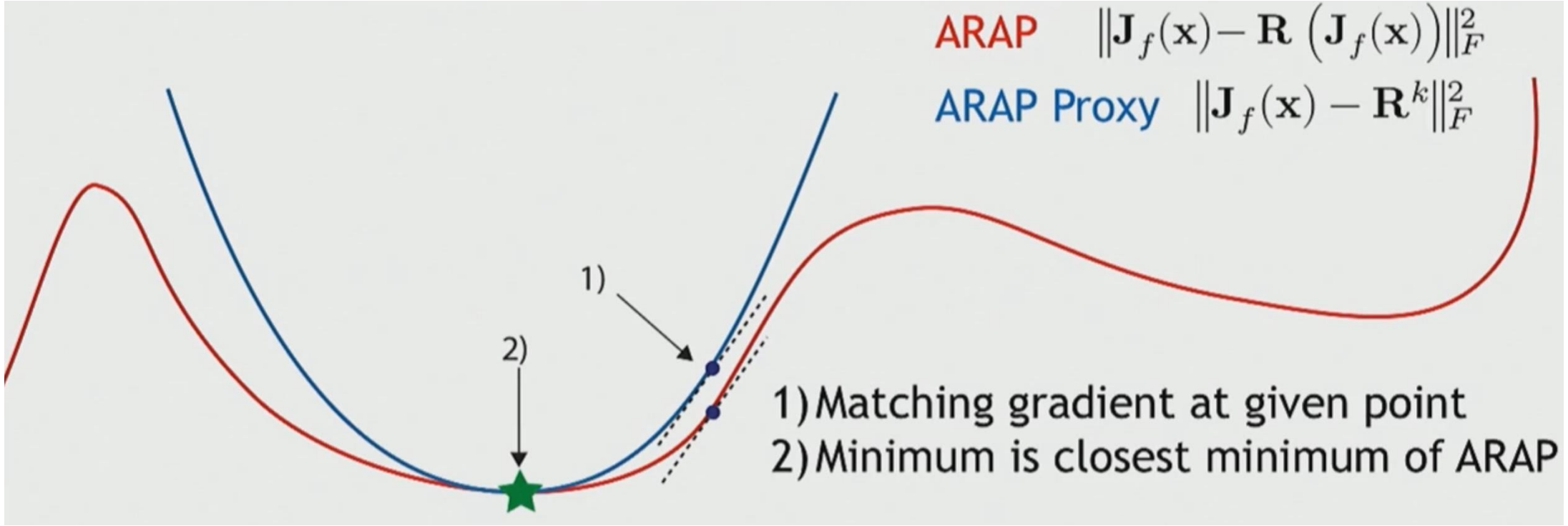


Motivation

$$f_{\mathrm{ARAP}}(\mathrm{x}) = \sum_i \left((\Sigma_i - 1)^2 + (\sigma_i - 1)^2\right)|t_i|$$

$$f_{\mathrm{ISO}}(\mathrm{x}) = \sum_i \left(\Sigma_i^2 + \Sigma_i^{-2} + \sigma_i^2 + \sigma_i^{-2}\right)|t_i|$$





ARAP Proxy $P^{R_i^n}(J) = \|J - R_i^n\|_F^2 \longrightarrow \mathcal{D}(J) = \|J - R(J)\|_F^2$ ARAP

- Majorizer: $P^{R_i^n}(J) \geq \mathcal{D}(J) \quad \forall J$
- Matching gradient: $\nabla_J P^{R_i^n}(J_i^n) = \nabla_J \mathcal{D}(J_i^n)$
- Closest minimizer: $\min_J P^{R_i^n}(J) = \text{Proj}(J_i^n)$

Generalize Local/Global

- Majorizer: $P^{R_i^n}(J) \geq \mathcal{D}(J) \quad \forall J$
- Matching gradient: $\nabla_J P^{R_i^n}(J_i^n) = \nabla_J \mathcal{D}(J_i^n)$
- Closest minimizer: $\min_J P^{R_i^n}(J) = \text{Proj}(J_i^n)$

$$J_i^n = J_i(\mathrm{x}_{n-1}) = USV^T,\ R_i^n = R(J_i^n) = UV^T$$

$$P_w^R(J) = \|W(J-R)\|_F^2 \qquad R = R_i^n$$

$$\nabla_J \|W(J-R)\|_F^2 = \nabla_J \mathcal{D}(J)$$

$$\nabla_J \text{tr}\left(W^T W(J-R)(J-R)^T\right) = \left(W^T W + WW^T\right)(J-R) = \nabla_J \mathcal{D}(J)$$

$$W = \left(\frac{1}{2}\nabla_J \mathcal{D}(J)(J-R)^{-1}\right)^{1/2}$$

$$\mathcal{D}(J) = \|J - R(J)\|_F^2$$

$$\nabla_J \mathcal{D}(J) = 2(J-R)$$

$$W = I$$

Generalize Local/Global

$$W = \left(\frac{1}{2}\nabla_J \mathcal{D}(J)(J-R)^{-1}\right)^{1/2}$$

$$J = USV^T \qquad S = \begin{bmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{bmatrix}$$

$$\mathcal{D}(J) = \mathcal{D}(S) \qquad \nabla_J \mathcal{D}(J) = U\nabla_S \mathcal{D}(S)V^T$$

$$(J-R)^{-1} = \left(USV^T - UV^T\right)^{-1} = \left(U(S-I)V^T\right)^{-1} = V(S-I)^{-1}U^T$$

$$W = \left(\frac{1}{2}U\nabla_S \mathcal{D}(S)V^T V(S-I)^{-1}U^T\right)^{1/2}$$

$$W = U\left(\frac{1}{2}\nabla_S \mathcal{D}(S)(S-I)^{-1}\right)^{1/2} U^T = US_W U^T$$

Generalize Local/Global

$$W = U\left(\frac{1}{2}\nabla_S \mathcal{D}(S)(S-I)^{-1}\right)^{1/2} U^T = US_W U^T$$

$$\mathcal{D}(J) = \|J\|_F^2 + \|J\|_F^{-2} = \left(\sigma_1^2 + \sigma_1^{-2} + \sigma_2^2 + \sigma_2^{-2}\right) \qquad S = \begin{bmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{bmatrix}$$

$$\nabla_S \mathcal{D}(S) = \begin{bmatrix}2\left(\sigma_1 + \sigma_1^{-3}\right) & 0\\ 0 & 2\left(\sigma_2 + \sigma_2^{-3}\right)\end{bmatrix}$$

$$S_W = \begin{bmatrix}\sqrt{\dfrac{\left(\sigma_1 + \sigma_1^{-3}\right)}{\sigma_1 - 1}} & 0\\ 0 & \sqrt{\dfrac{\left(\sigma_2 + \sigma_2^{-3}\right)}{\sigma_2 - 1}}\end{bmatrix}$$

Generalize Local/Global

Iterate

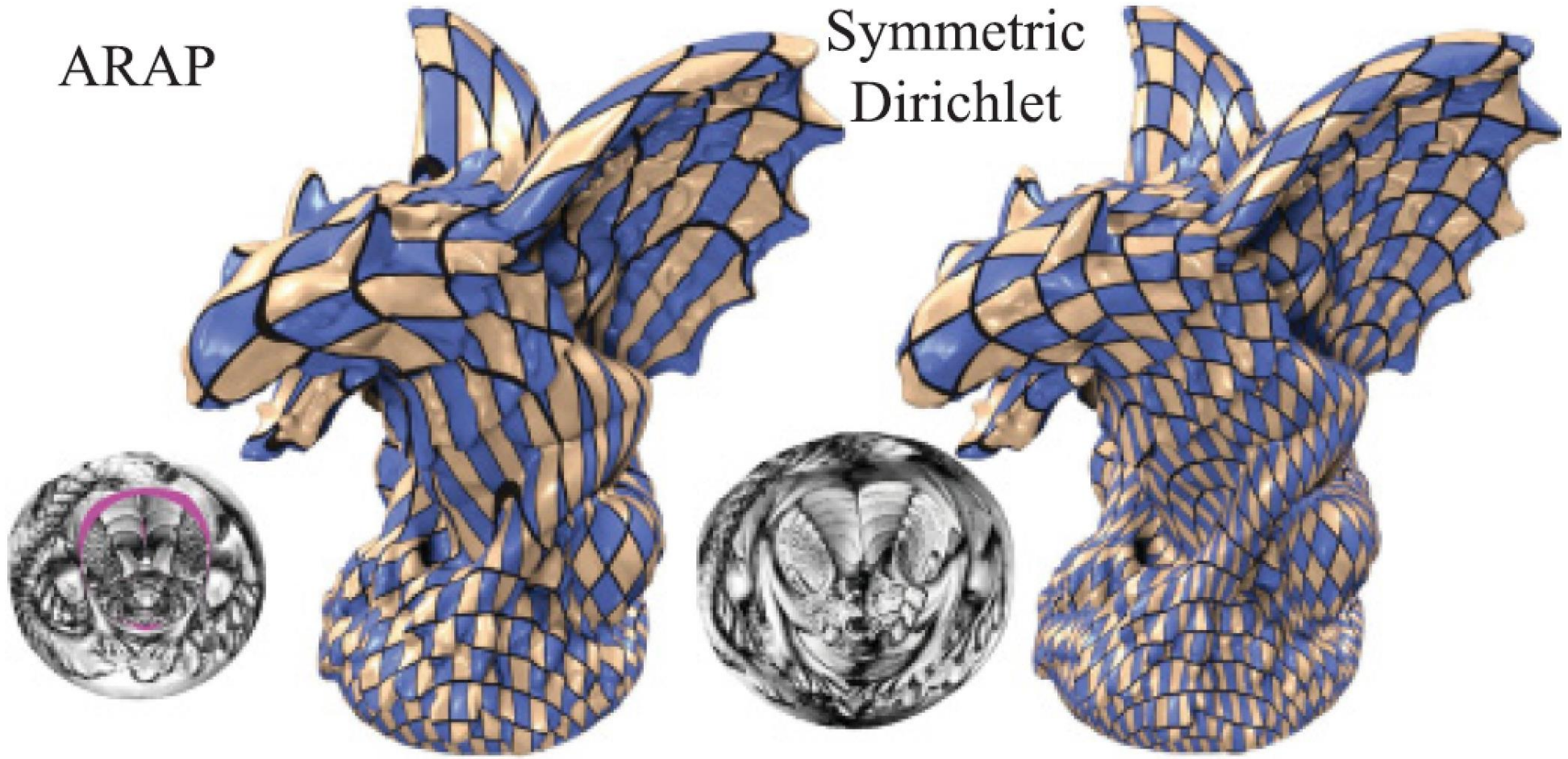
Local $\mathbf{R}^k = \mathbf{R}\left(\mathbf{J}_f(\mathbf{x}^k)\right)$

Compute weights

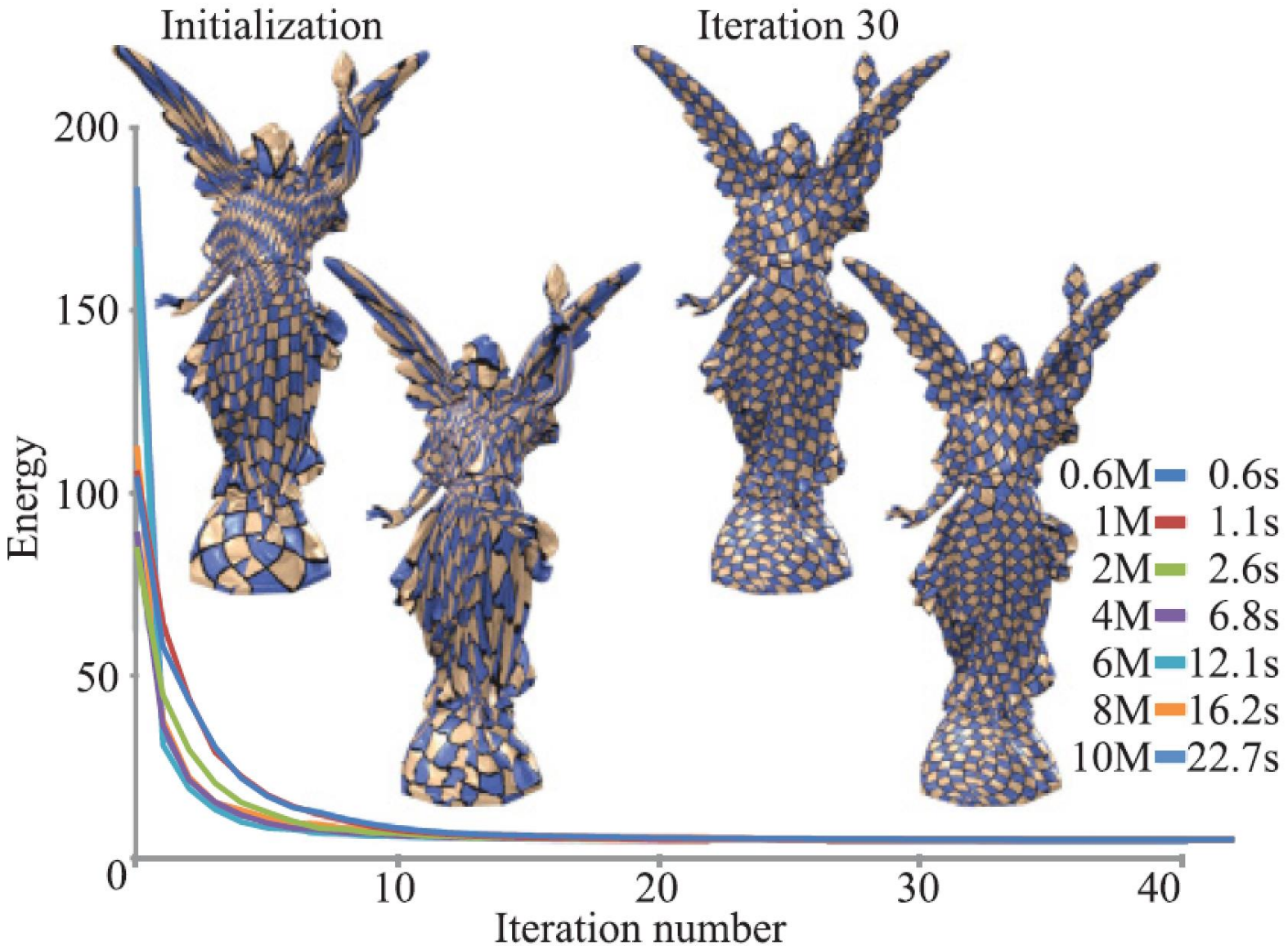
Global $\tilde{\mathbf{x}}^{k+1} = \arg\min_{\mathbf{x}} \sum_f \left\| W_f^k \left(\mathbf{J}_f(\mathbf{x}) - \mathbf{R}^k \right) \right\|_F^2$

Line search $\mathbf{d} = \tilde{\mathbf{x}}^{k+1} - \mathbf{x}^k$ [Smith and Schaefer 2015]

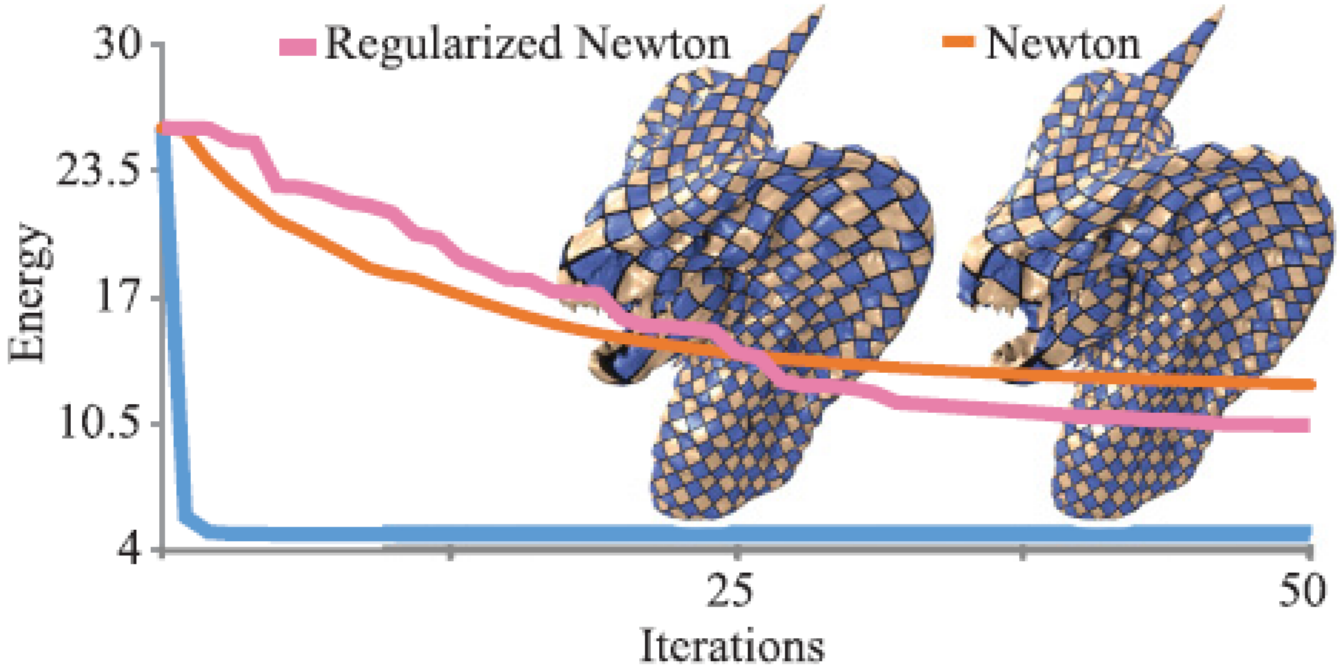
Result

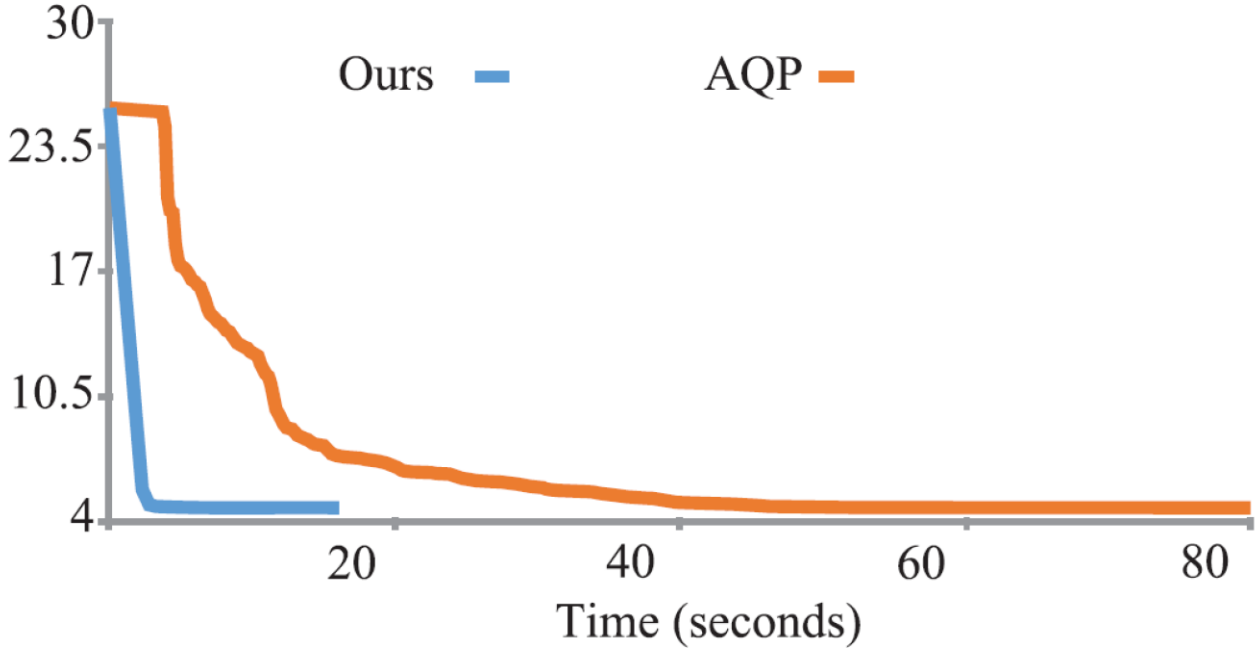


Result


Initialization
Iteration 30
200
150
100
50
0
Energy
10
20
30
40
Iteration number
0.6M 0.6s
1M 1.1s
2M 2.6s
4M 6.8s
6M 12.1s
8M 16.2s
10M 22.7s

Result


Regularized Newton
Newton
30
23.5
17
10.5
4
Energy
25
50
Iterations


Ours
AQP
30
23.5
17
10.5
4
20
40
60
80
Time (seconds)

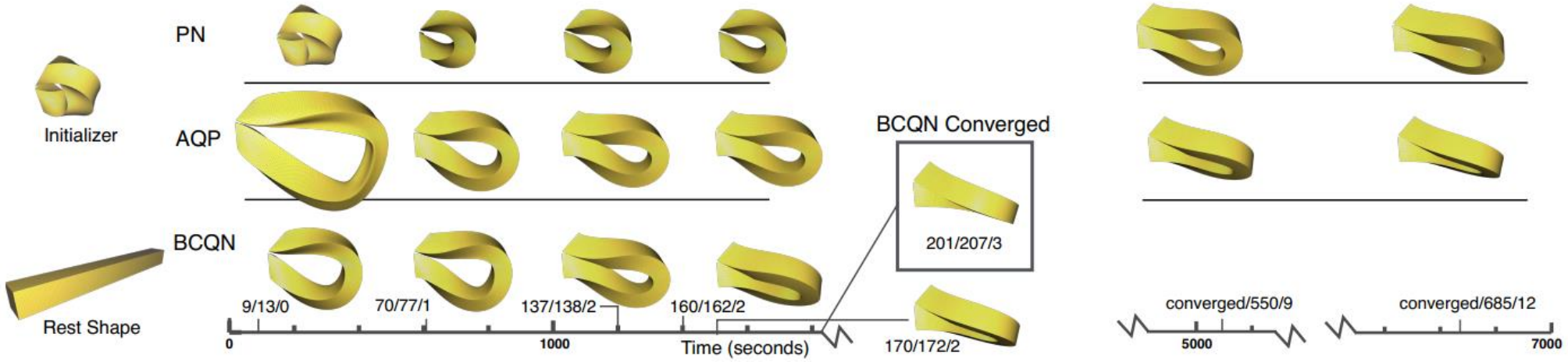
Result

Blended Cured Quasi-Newton for Distortion Optimization

YuFeng Zhu

Robert Bridson

Danny Kaufman



Motivation

$$\min_p f(\mathrm{x}_n + p)$$

$$f(\mathrm{x}_n + p) \approx f(\mathrm{x}_n) + \nabla f(\mathrm{x}_n)^T p + \frac{1}{2} p^T H p$$

$$Hp = -\nabla f(\mathrm{x}_n)$$

- First-order methods build descent steps by preconditioning the gradient with a fixed proxy matrix, which often suffer from slower convergence as lacking of higher-order information.
- Newton-type methods uses the energy Hessian, $\nabla^2 f(\mathrm{x})$, to form a proxy matrix, which can achieve the most rapid convergence but require the costly assembly, factorization and backsolve of new linear systems per iterate.

Quasi-Newton Methods

$$f(\mathrm{x}) \approx f(\mathrm{x}_{n+1}) + \nabla f(\mathrm{x}_{n+1})^T(\mathrm{x} - \mathrm{x}_{n+1}) + \frac{1}{2}(\mathrm{x} - \mathrm{x}_{n+1})^T H_{n+1}(\mathrm{x} - \mathrm{x}_{n+1})$$

$$\nabla f(\mathrm{x}_{n+1}) + H_{n+1}(\mathrm{x}_n - \mathrm{x}_{n+1}) = \nabla f(\mathrm{x}_n)$$

$$H_{n+1}(\mathrm{x}_{n+1} - \mathrm{x}_n) = \nabla f(\mathrm{x}_{n+1}) - \nabla f(\mathrm{x}_n)$$

secant equation $\boxed{H_{n+1}s_n = \mathrm{y}_n \qquad s_n = D_{n+1}\mathrm{y}_n}$

$$p_{n+1} = -D_{n+1}\nabla f(\mathrm{x}_{n+1})$$

$$D_{n+1} = QN(z, D_n) = V(z)^T D_n V(z) + \frac{s_n s_n^T}{s_n^T z}, V(z) = I - \frac{z s_n^T}{s_n^T z}$$

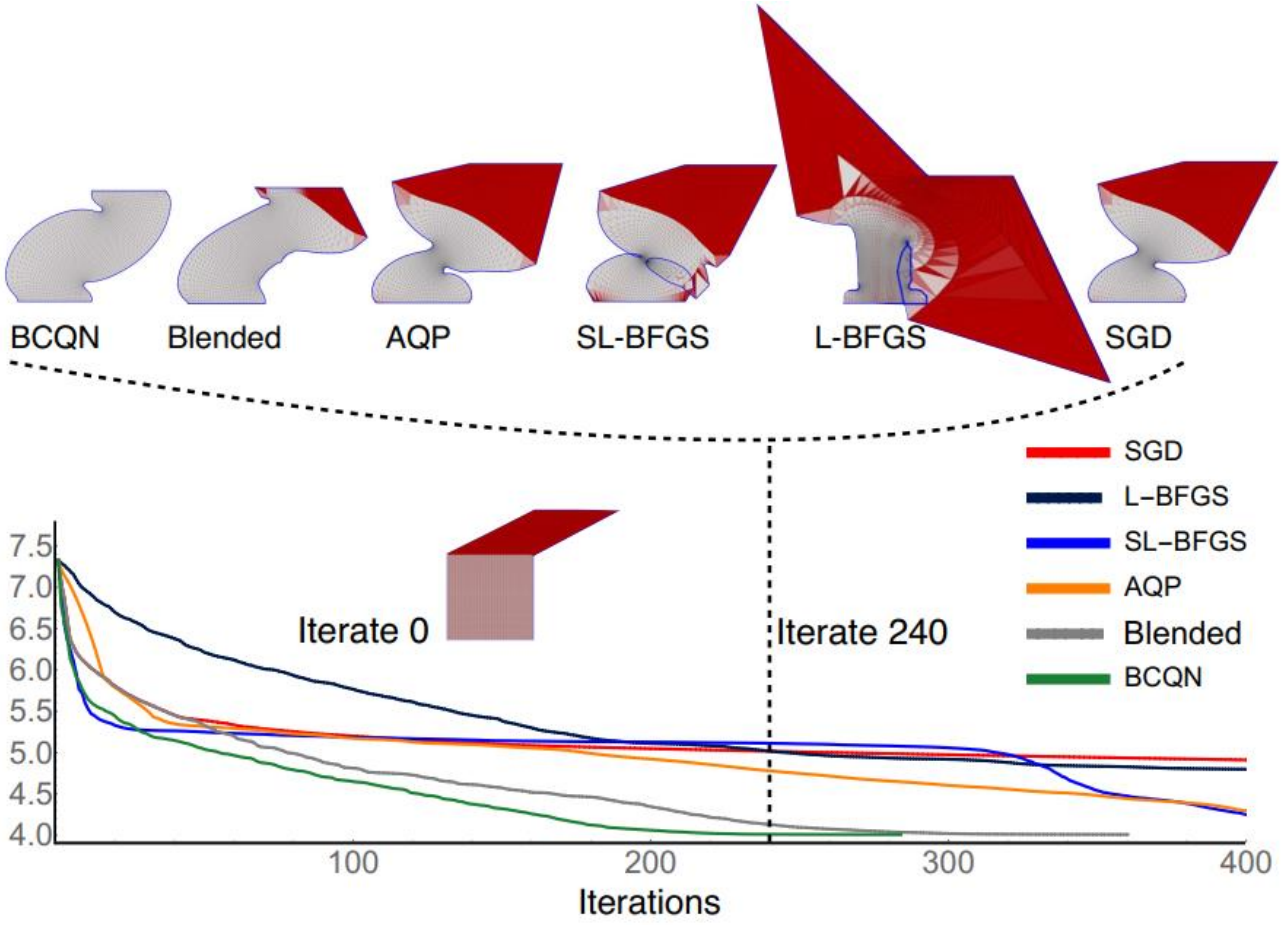
z: The difference in gradients

$$s_n = QN(\mathrm{z}, D_n)\mathrm{z}$$

$$D_{n+1} = QN(\mathrm{y}_n, D_n)$$

Quasi-Newton Methods

Quasi-Newton methods lie in between two methods. They employ sequential gradients to update approximations of the system Hessian, per descent iterate. However, the secant approximation can implicitly create a dense proxy, unlike the sparse true Hessian, direct and incorrect coupling distant vertices.



Blended Quasi-Newton

- The difference in gradients would be the better behaved Ls than y, which may introduce spurious coupling or have badly scaled entries near distorted triangles

$$D_{n+1} = QN(Ls_n, D_n)$$

- However, to achieve the superlinear convergence BFGS offers, near solutions we wish to come closer to satisfying the secant equation, and so aim to move towards using y instead.

$$z_n = (1 - \beta_n)y_n + \beta_n Ls_n$$

$$\beta_n = \min_{\beta \in [0,1]} \|y_n - \beta Ls_n\|^2, \qquad \beta_n = \text{proj}_{[0,1]}\left(\frac{y_n^T Ls_n}{\|Ls_n\|^2}\right)$$

$$D_{n+1} = QN(z_n, D_n)$$

Result

Second-order method

Paper List

- Geometric Optimization via Composite Majorization
- Piecewise Linear Mapping Optimization Based on The Complex View
- Progressive Parameterizations

Geometric Optimization via Composite Majorization

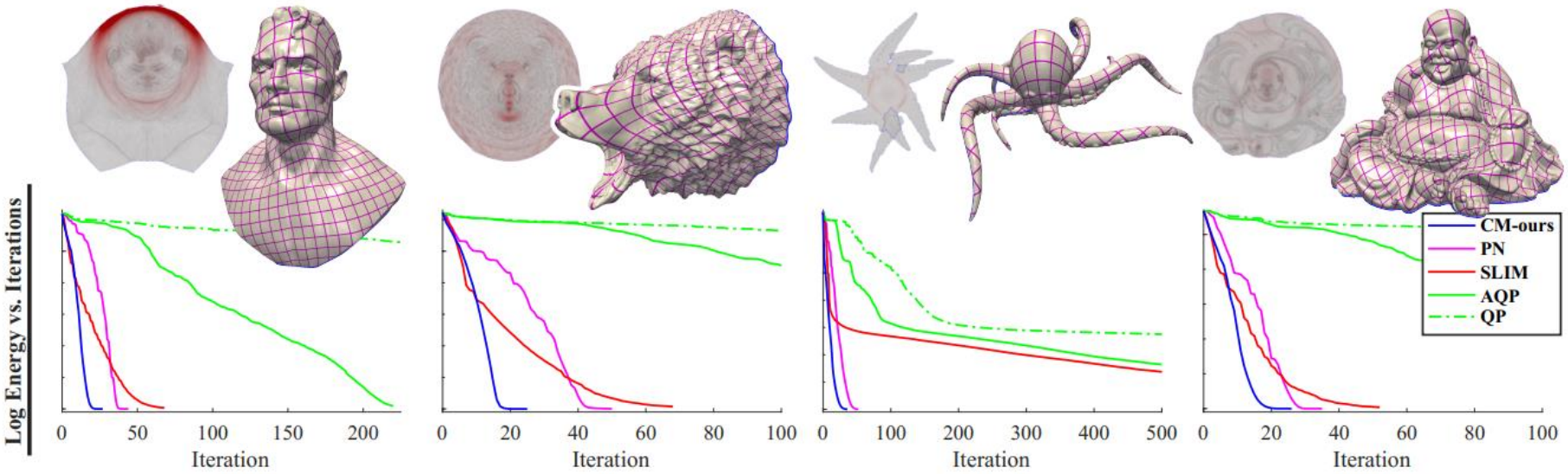
Anna Shtengel

Roi Poranne

Olga Sorkine-Hornung

Shahar Kovalsky

Yaron Lipman



Motivation

$$\min_{\mathrm{x}} f(\mathrm{x}) = \sum_{i} |t_i| \mathcal{D}\big(J_i(\mathrm{x})\big)$$

1.Initial point: x_0， $n = 0$

2.Descent direction: $Hp = -\nabla f(x_n)$

3.Step size: $\min_{\alpha} f(\mathrm{x}_n + \alpha p)$

4.Update: $\mathrm{x}_{n+1} = \mathrm{x}_n + \alpha p$， $n = n + 1$

analytic and simple to evaluate
convex approximate Hessian *H*

Method

$$f(\mathrm{x}) = \sum_i h_i\big(\boldsymbol{g}_i(\mathrm{x})\big) = \sum_i (h_i \circ \boldsymbol{g}_i)(\mathrm{x})$$

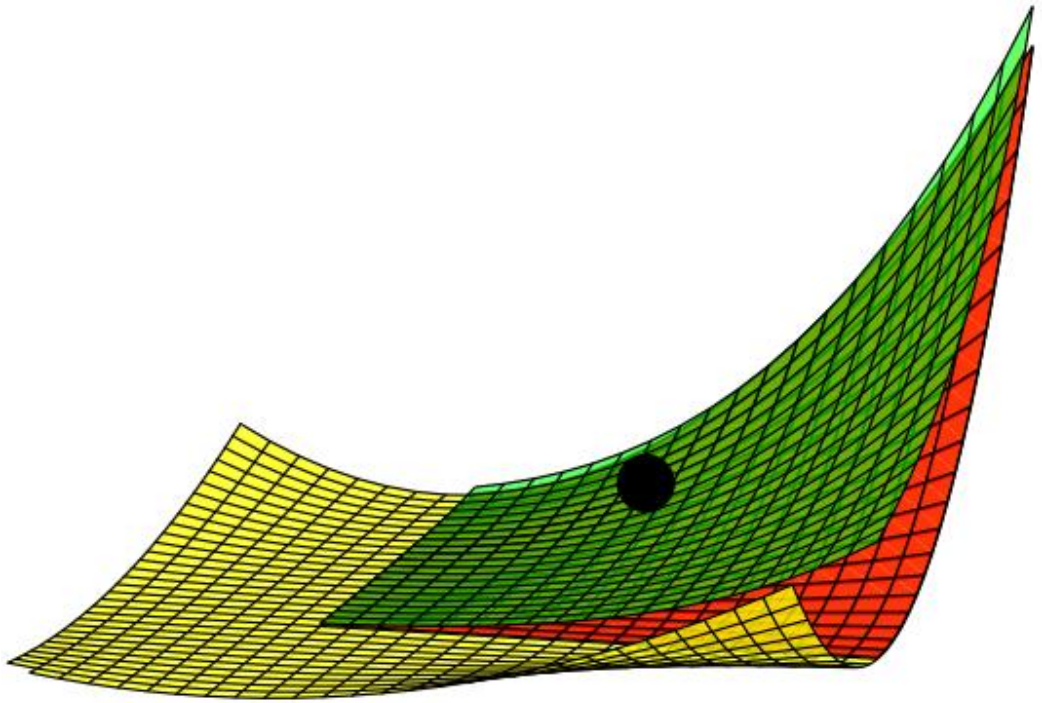
$$h_i : R^k \longrightarrow R, \qquad h_i = h_i^+ + h_{\bar{i}}$$

$$\boldsymbol{g}_i : R^n \longrightarrow R^k, \qquad \boldsymbol{g}_i = \boldsymbol{g}_i^+ + \boldsymbol{g}_{\bar{i}}$$

$$f(\mathrm{x}_n + p) \approx f(\mathrm{x}_n) + \nabla f(\mathrm{x}_n)^T p + \frac{1}{2} p^T H p$$

$\overline{f}$ is a local convex majorizer of f

$$H = \nabla^2 \overline{f}$$

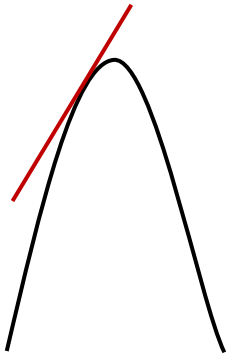


Method

$$r = r^+ + r^-$$

convex majorizer $\quad \overline{r}(\mathrm{x};\mathrm{x}_0) = r^+(\mathrm{x}) + r^-(\mathrm{x}_0) + \nabla r^-(\mathrm{x}_0)(\mathrm{x}-\mathrm{x}_0)$

concave minorizer $\quad \underline{r}(\mathrm{x};\mathrm{x}_0) = r^-(\mathrm{x}) + r^+(\mathrm{x}_0) + \nabla r^+(\mathrm{x}_0)(\mathrm{x}-\mathrm{x}_0)$



$$f(\mathrm{x}) = h\big(\boldsymbol{g}(\mathrm{x})\big) = h\big(g_1(\mathrm{x}),\cdots,g_k(\mathrm{x})\big)$$

$$\boldsymbol{u}_0 = \boldsymbol{g}(\mathrm{x}_0) \qquad s_j(\boldsymbol{u}) = \mathrm{sign}\left(\frac{\partial \overline{h}}{\partial u_j}(\boldsymbol{u};\boldsymbol{u}_0)\right)$$

$$[g_j](\mathrm{x};\mathrm{x}_0) = \begin{cases} \overline{g}_j(\mathrm{x};\mathrm{x}_0) & s_j(\boldsymbol{u}_0) > 0 \\ \underline{g}_j(\mathrm{x};\mathrm{x}_0) & s_j(\boldsymbol{u}_0) < 0 \end{cases}$$

$$\overline{f}(\mathrm{x};\mathrm{x}_0) = \overline{h}([\boldsymbol{g}](\mathrm{x};\mathrm{x}_0);\boldsymbol{u}_0)$$

Method

$$\overline{f}(\mathrm{x};\mathrm{x}_0) = \overline{h}([\boldsymbol{g}](\mathrm{x};\mathrm{x}_0);\boldsymbol{u}_0)$$

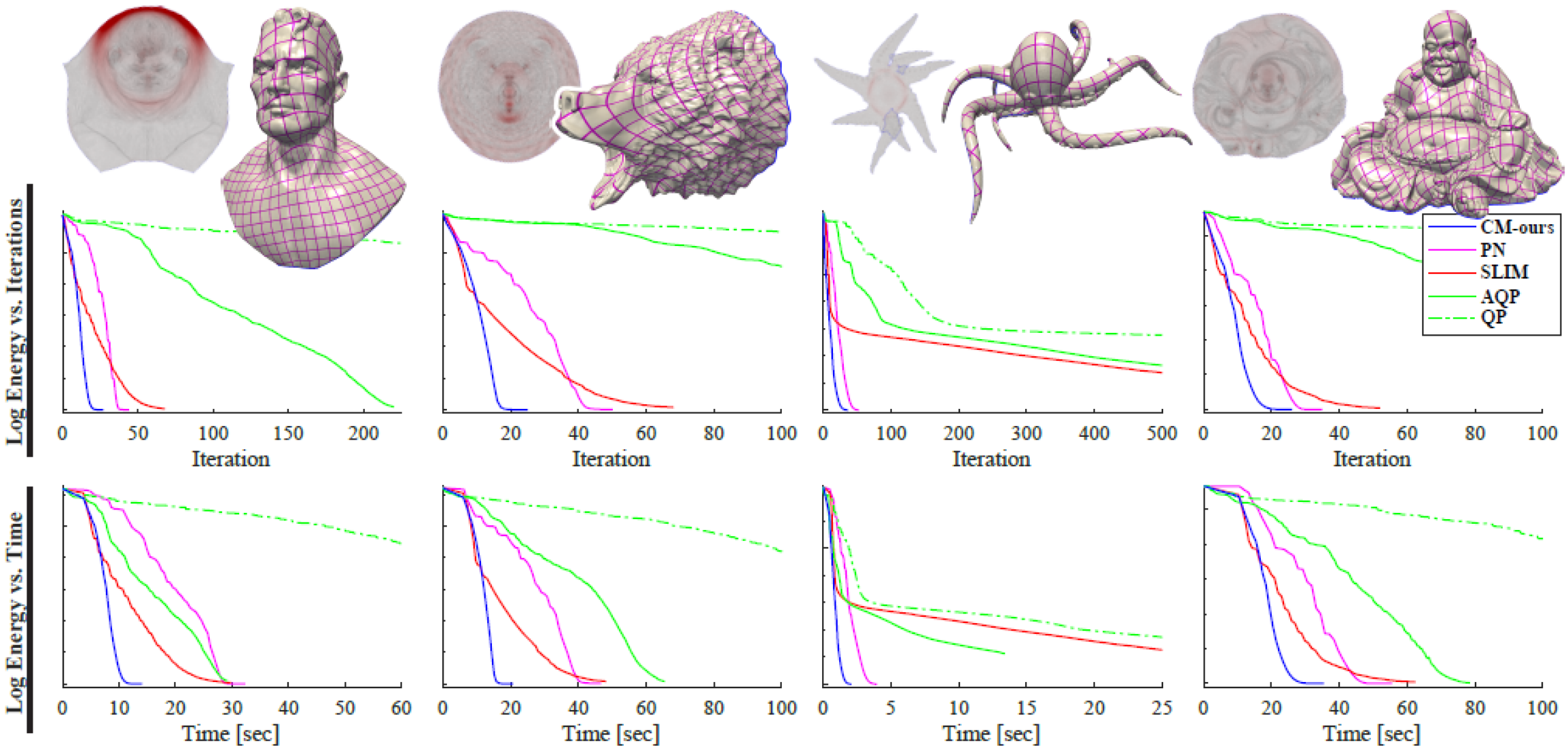
$\overline{f}$ is convex majorier

$$\nabla_{\mathrm{x}}^2\overline{f}(\mathrm{x};\mathrm{x}_0) = \frac{\partial[\boldsymbol{g}]^T}{\partial \mathrm{x}}\nabla^2 h^+\frac{\partial[\boldsymbol{g}]}{\partial \mathrm{x}} + \sum_j \frac{\partial \overline{h}}{\partial u_j}\begin{cases}\nabla^2 g_j^+ & s_j(\boldsymbol{u}_0) > 0\\ \nabla^2 g_j^- & s_j(\boldsymbol{u}_0) < 0\end{cases}$$

$$\overline{f}(\mathrm{x};\mathrm{x}_0) = \overline{h}([\boldsymbol{g}](\mathrm{x};\mathrm{x}_0);\boldsymbol{u}_0) \geq \overline{h}(\boldsymbol{g}(\mathrm{x});\boldsymbol{u}_0) \geq h\big(\boldsymbol{g}(\mathrm{x})\big)$$

$$H = \nabla^2\overline{f}(\mathrm{x};\mathrm{x}_0)\Big|_{\mathrm{x}=\mathrm{x}_0}$$

Result


Log Energy vs. Iterations
Iteration
Log Energy vs. Time
Time [sec]
CM-ours
PN
SLIM
AQP
QP

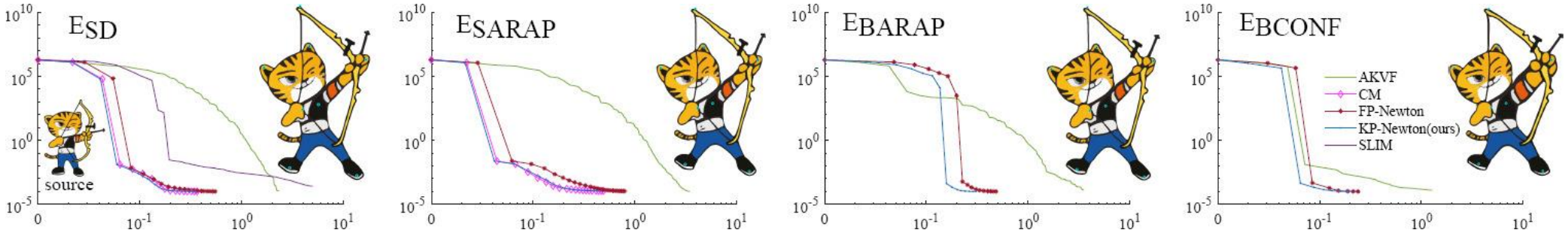
Result

Piecewise linear mapping optimization based on the complex view

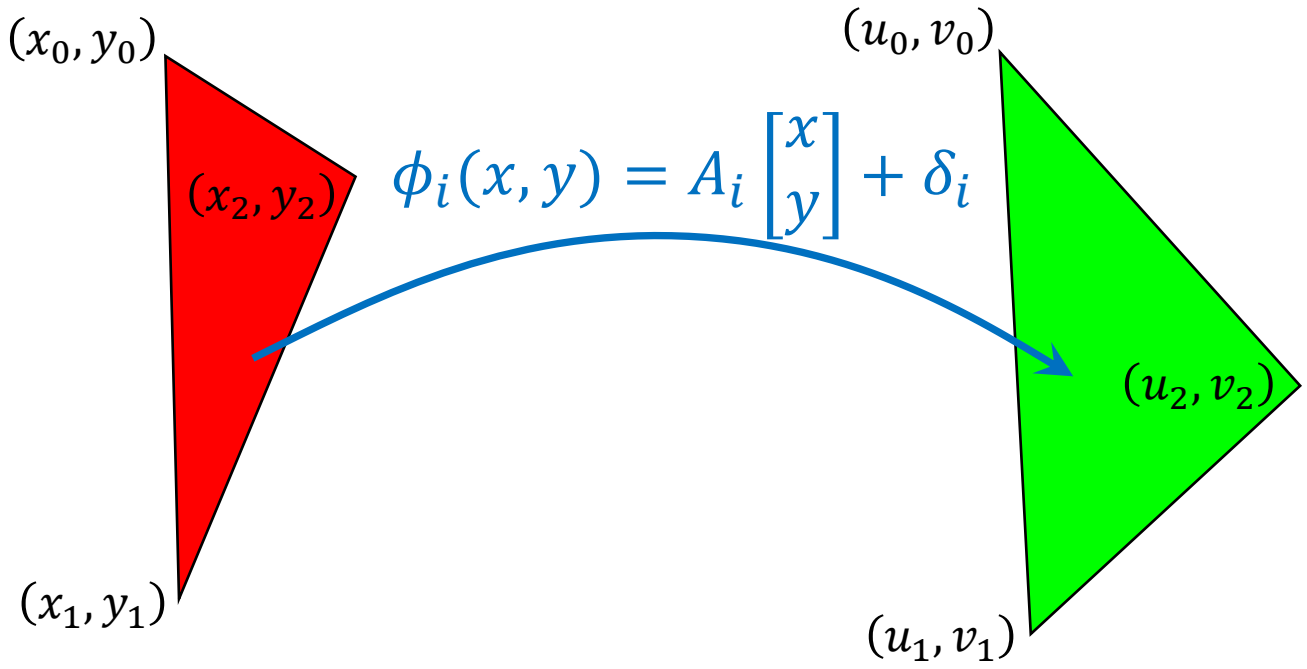
Björn Golla

Hans-Peter Seidel

Renjie Chen



Real view

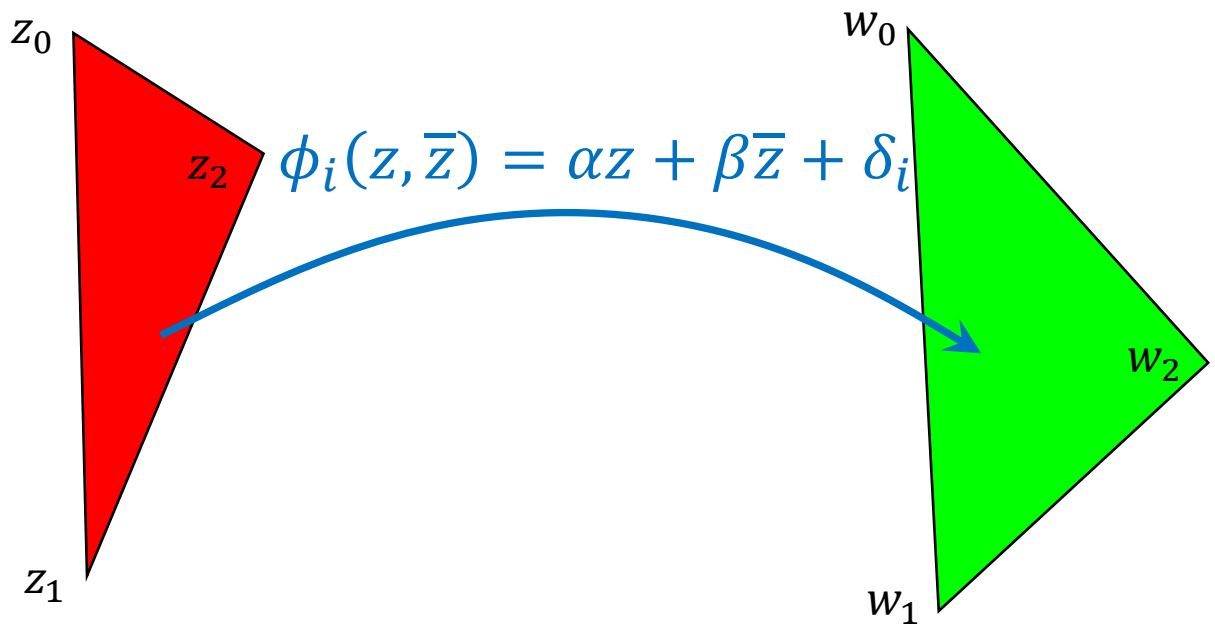


$$\begin{bmatrix} u \\ v \end{bmatrix} = A_i \begin{bmatrix} x \\ y \end{bmatrix} + c_i$$

$$J_i = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = A_i$$

$$J_i = \begin{bmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{bmatrix} \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}^{-1}$$

Complex view



$$z = x + iy,\ \ \overline{z} = x - iy$$

$$x = \frac{z+\overline{z}}{2},\ \ y = \frac{z-\overline{z}}{2i}$$

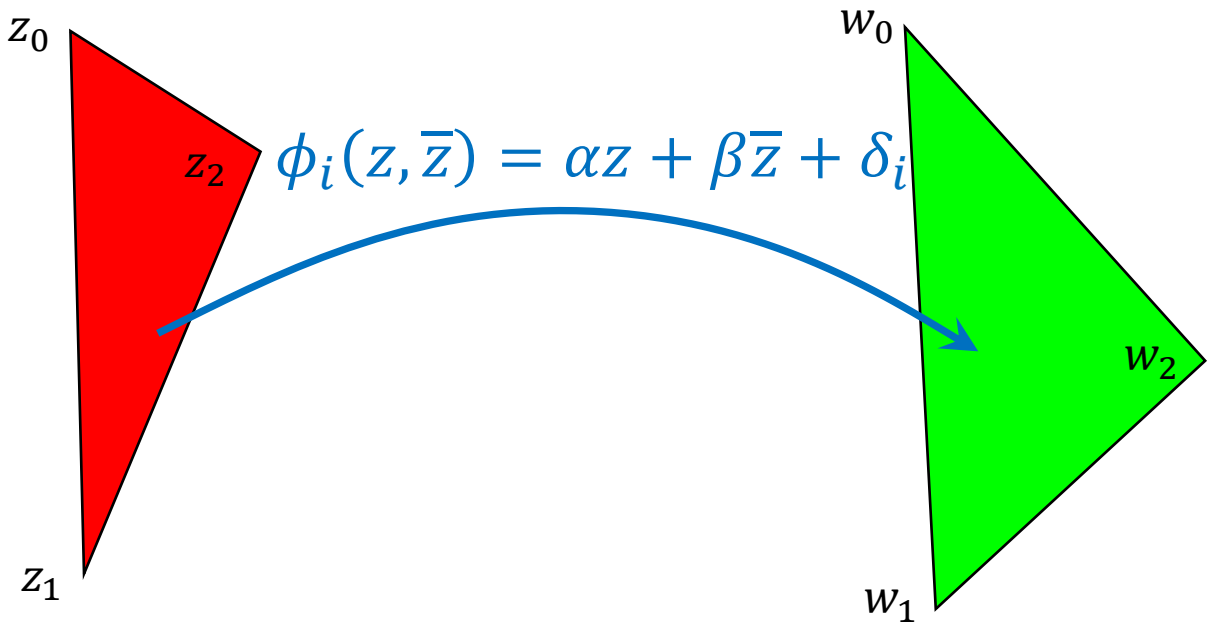
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \qquad (ax+by) + i(cx+dy)$$

$$\left(\frac{a+d}{2} + i\frac{c-b}{2}\right)z + \left(\frac{a-d}{2} + i\frac{c+b}{2}\right)\overline{z}$$

$$\alpha = \frac{a+d}{2} + i\frac{c-b}{2}, \beta = \frac{a-d}{2} + i\frac{c+b}{2}$$

$$\Sigma = |\alpha| + |\beta|,\ \ \sigma = \big||\alpha| - |\beta|\big|$$

Complex view



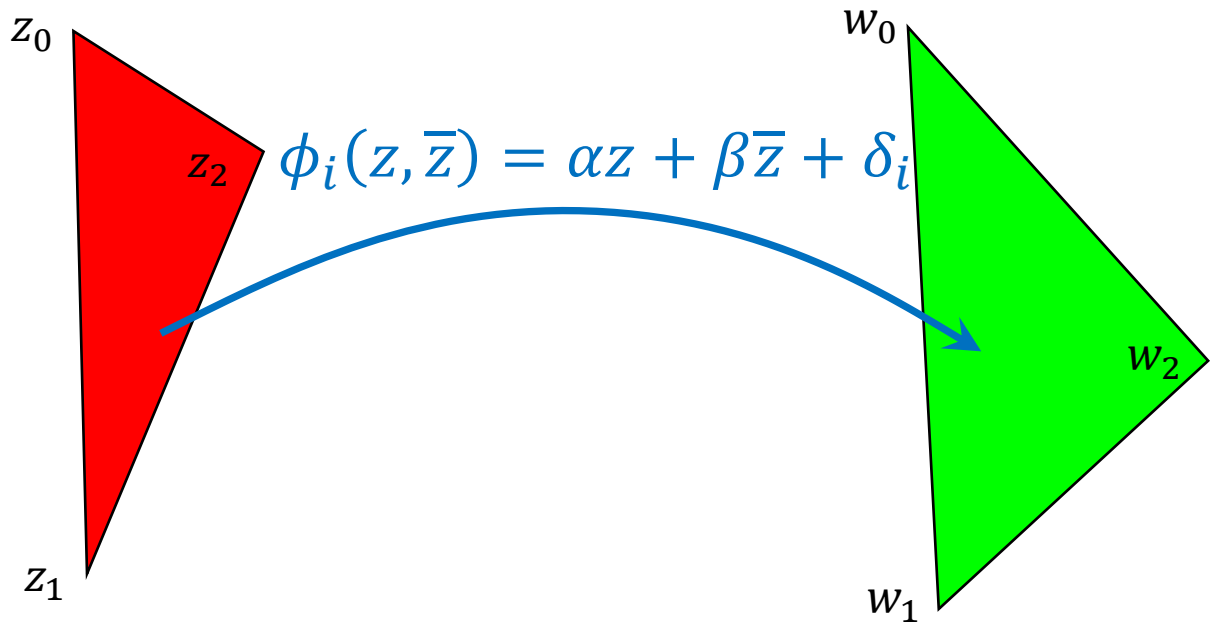
$$\begin{bmatrix} z_0 & \overline{z_0} & 1 \\ z_1 & \overline{z_1} & 1 \\ z_2 & \overline{z_2} & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \delta \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} z_1 - z_0 & \overline{z_1} - \overline{z_0} \\ z_2 - z_0 & \overline{z_2} - \overline{z_0} \end{bmatrix}^{-1} \begin{bmatrix} w_1 - w_0 \\ w_2 - w_0 \end{bmatrix}$$

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{i}{4|t_i|} \begin{bmatrix} \overline{z_2} - \overline{z_0} & \overline{z_0} - \overline{z_1} \\ z_0 - z_2 & z_1 - z_0 \end{bmatrix} \begin{bmatrix} w_1 - w_0 \\ w_2 - w_0 \end{bmatrix}$$

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{i}{4|t_i|} \begin{bmatrix} \overline{z_1} - \overline{z_2} & \overline{z_2} - \overline{z_0} & \overline{z_0} - \overline{z_1} \\ z_2 - z_1 & z_0 - z_2 & z_1 - z_0 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

Complex view



$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \frac{i}{4|t_i|}\begin{bmatrix}\overline{z_1}-\overline{z_2} & \overline{z_2}-\overline{z_0} & \overline{z_0}-\overline{z_1}\\ z_2-z_1 & z_0-z_2 & z_1-z_0\end{bmatrix}\begin{bmatrix}w_0\\ w_1\\ w_2\end{bmatrix}$$

$$e^0 = z_1 - z_2,\ \ e^1 = z_2 - z_0,\ \ e^2 = z_0 - z_1$$

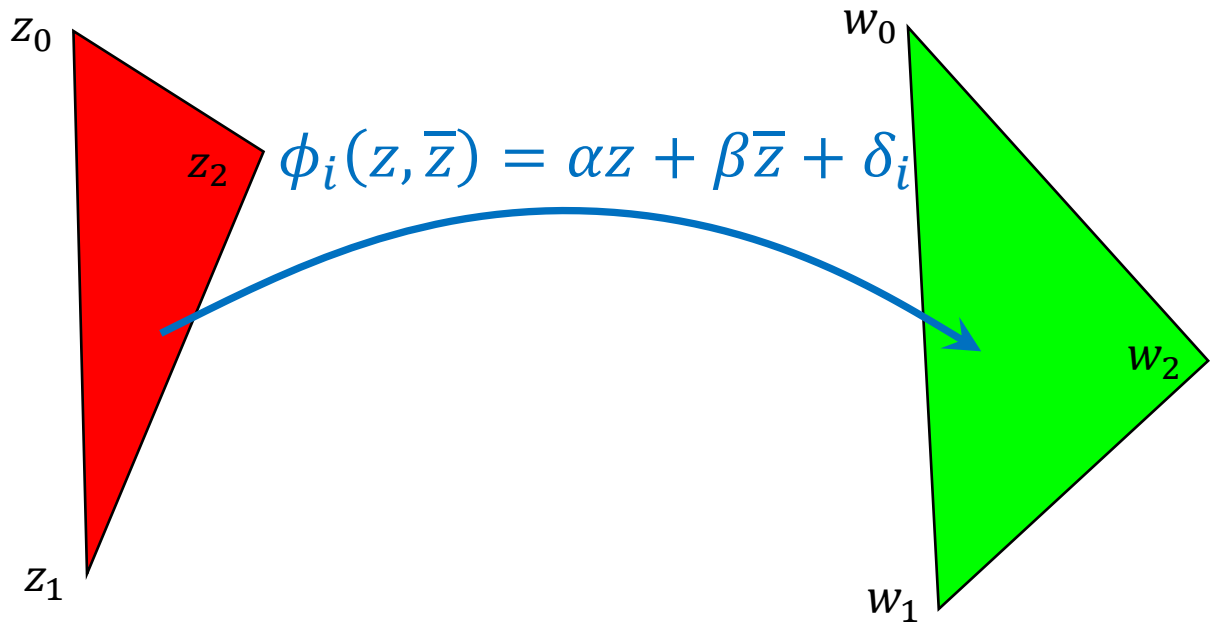
$$\alpha = \frac{i}{4|t_i|}\overline{[e^0 \quad e^1 \quad e^2]}W$$

$$\beta = -\frac{i}{4|t_i|}[e^0 \quad e^1 \quad e^2]W$$

$$D = \frac{i}{4|t_i|}\overline{[e^0 \quad e^1 \quad e^2]}$$

$$\alpha = DW,\ \ \beta = \overline{D}W$$

Complex view



$$\alpha = DW, \quad \beta = \overline{D}W$$

$$\alpha = \begin{bmatrix} \mathrm{Re}(\alpha) \\ \mathrm{Im}(\alpha) \end{bmatrix}_{2\times 1}, \quad W = \begin{bmatrix} \mathrm{Re}(W) \\ \mathrm{Im}(W) \end{bmatrix}_{6\times 1}$$

$$D = \begin{bmatrix} \mathrm{Re}(D) & -\mathrm{Im}(D) \\ \mathrm{Im}(D) & \mathrm{Re}(D) \end{bmatrix}_{2\times 6}$$

Method

$$\Sigma = |\alpha| + |\beta|, \quad \sigma = |\alpha| - |\beta|$$

$$r = |\alpha|^2, \quad s = |\beta|^2$$

$$\alpha = DW, \quad \beta = \overline{D}W$$

$$E = \left(\Sigma_i^2 + \sigma_i^2 + \Sigma_i^{-2} + \sigma_i^{-2}\right)$$

$$= \left(\Sigma_i^2 + \sigma_i^2\right)\left(1 + \frac{1}{\Sigma_i^2 \sigma_i^2}\right)$$

$$= (r + s)\left(1 + (r - s)^{-2}\right)$$

$$\xi_1 = \nabla_r E, \quad \xi_2 = \nabla_s E$$

$$\eta_1 = \nabla_r^2 E, \quad \eta_2 = \nabla_s^2 E, \quad \eta_3 = \nabla_r \nabla_s E$$

$$H_{6\times 6} = \nabla^2 E = M^T K M$$

$$M_{4\times 6} = \begin{bmatrix} D \\ \overline{D} \end{bmatrix}, \quad K = \begin{bmatrix} 2\xi_1 I + 4\eta_1 \alpha\alpha^T & 4\eta_3 \alpha\beta^T \\ 4\eta_3 \beta\alpha^T & \xi_2 I + 4\eta_2 \beta\beta^T \end{bmatrix}$$

Method

$$H_{6\times6} = \nabla^2 E = M^T K M$$

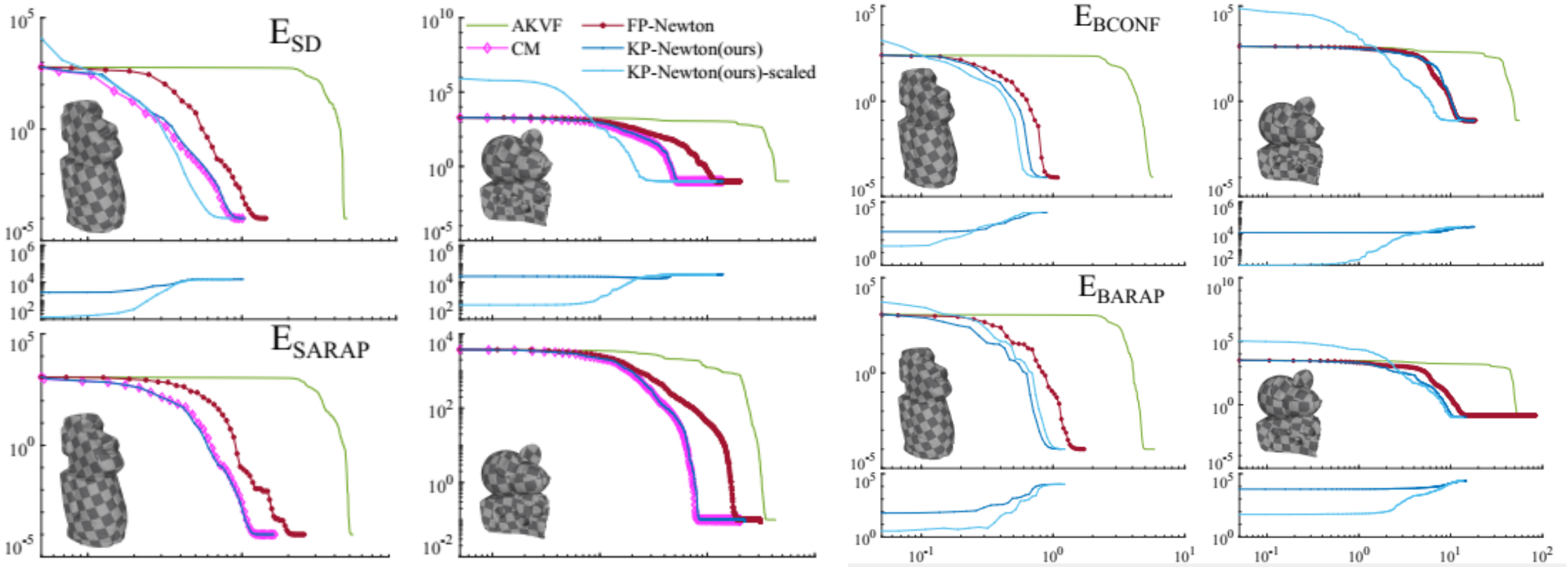
$$M_{4\times6} = R_{4\times4} Q_{\ 4\times6}$$

R is lower triangular matrix, Q is orthonormal

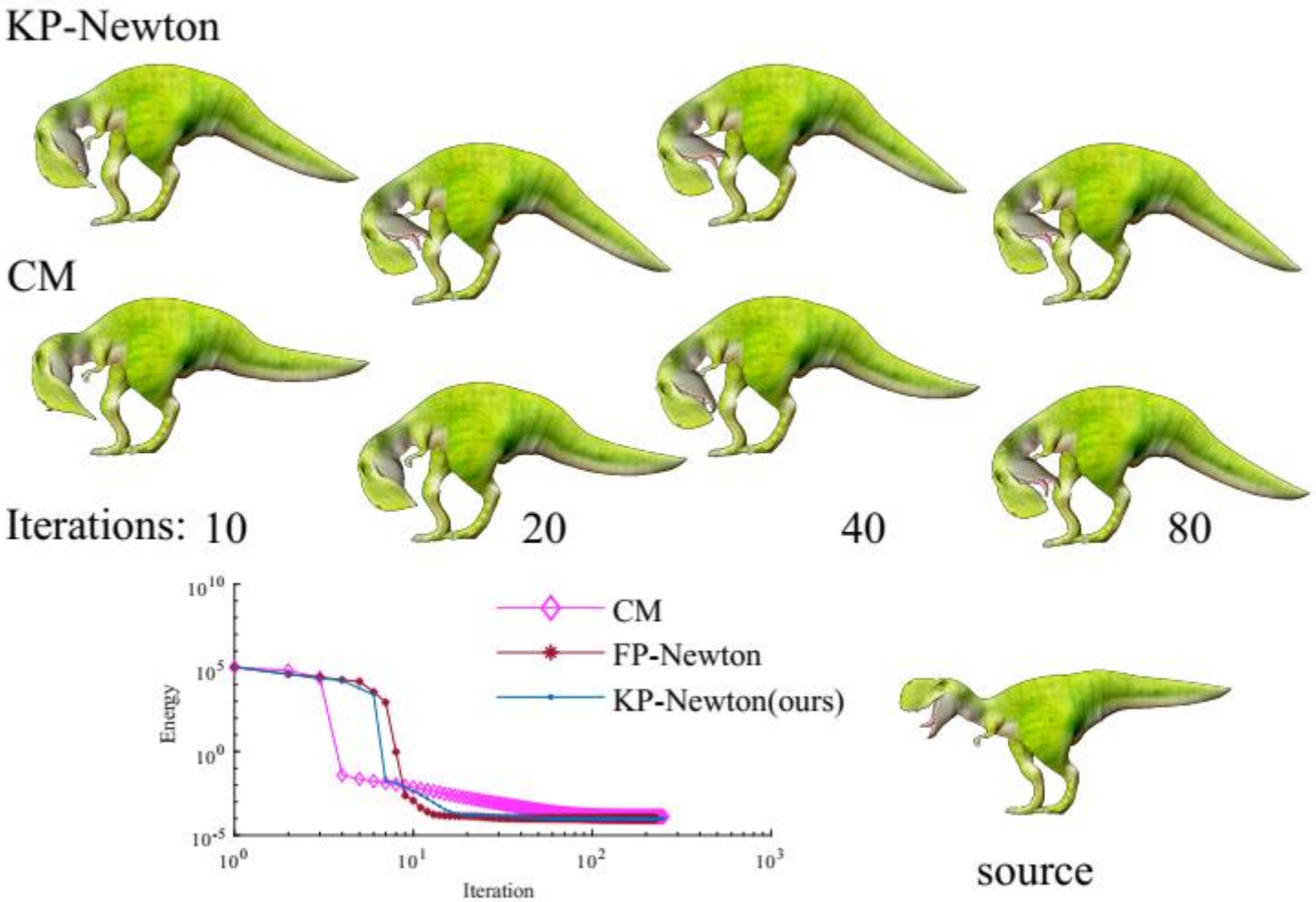
$$H = Q^T \left(R^T K R\right) Q$$

The 6×6 PSD projection of H is therefore equivalent to the 4×4 PSD projection of $R^T K R$, since Q is orthonormal.

Result


E_{SD}
E_{SARAP}
AKVF
CM
FP-Newton
KP-Newton(ours)
KP-Newton(ours)-scaled
E_{BCONF}
E_{BARAP}

Result


KP-Newton
CM
Iterations: 10
20
40
80
CM
FP-Newton
KP-Newton(ours)
Energy
Iteration
source

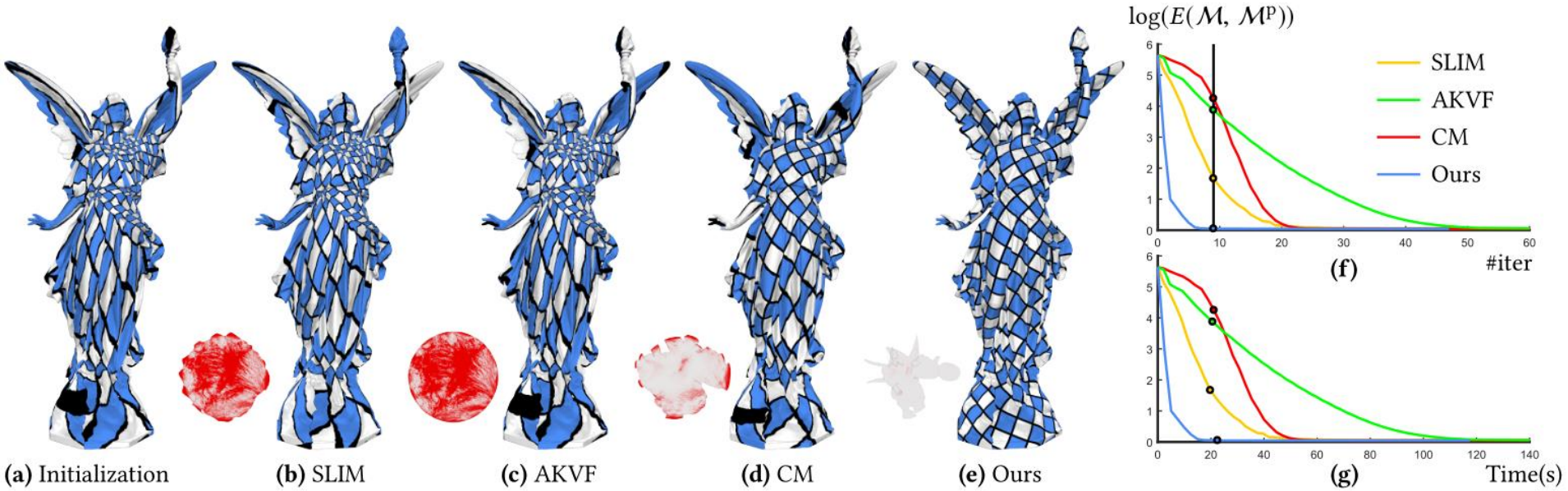
Progressive Parameterizations

Ligang Liu

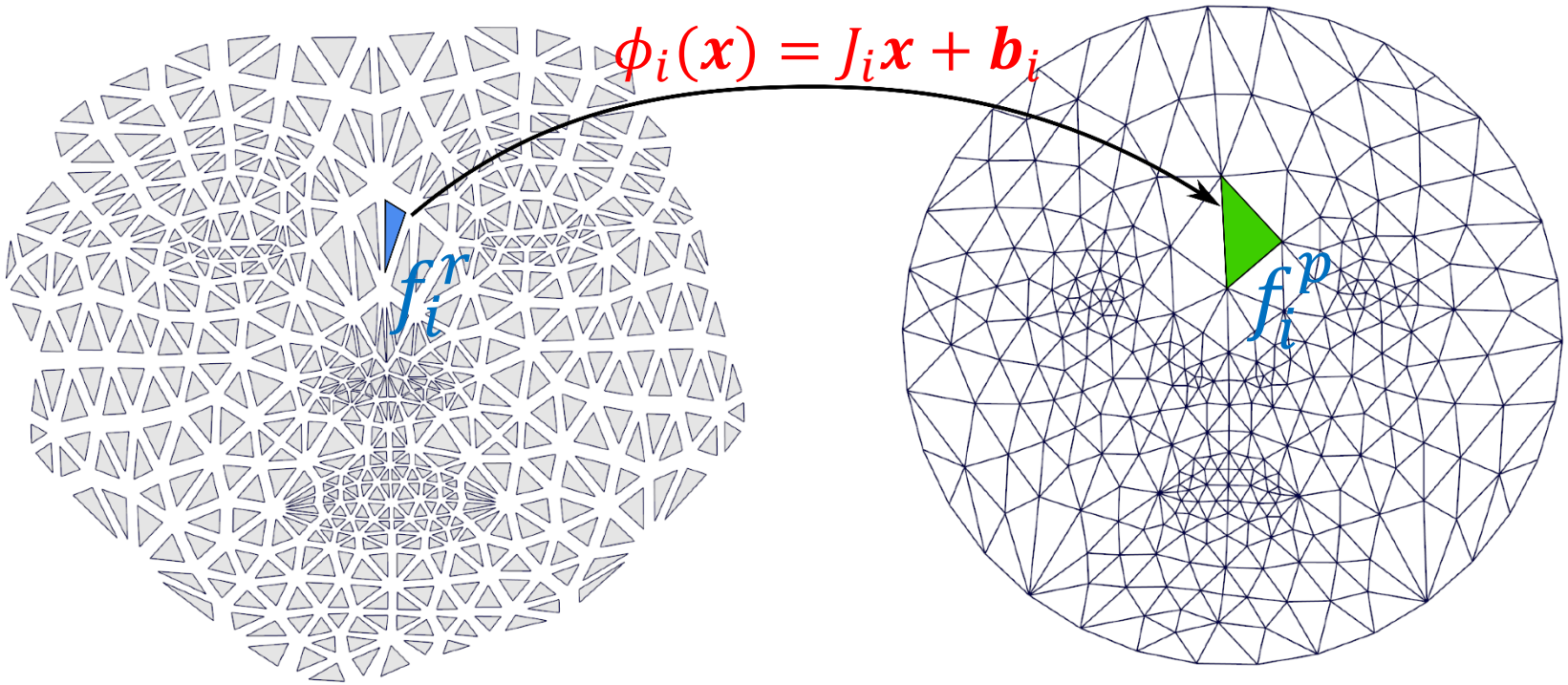
Chunyang Ye

Ruiqi Ni

Xiao-Ming Fu



Motivation

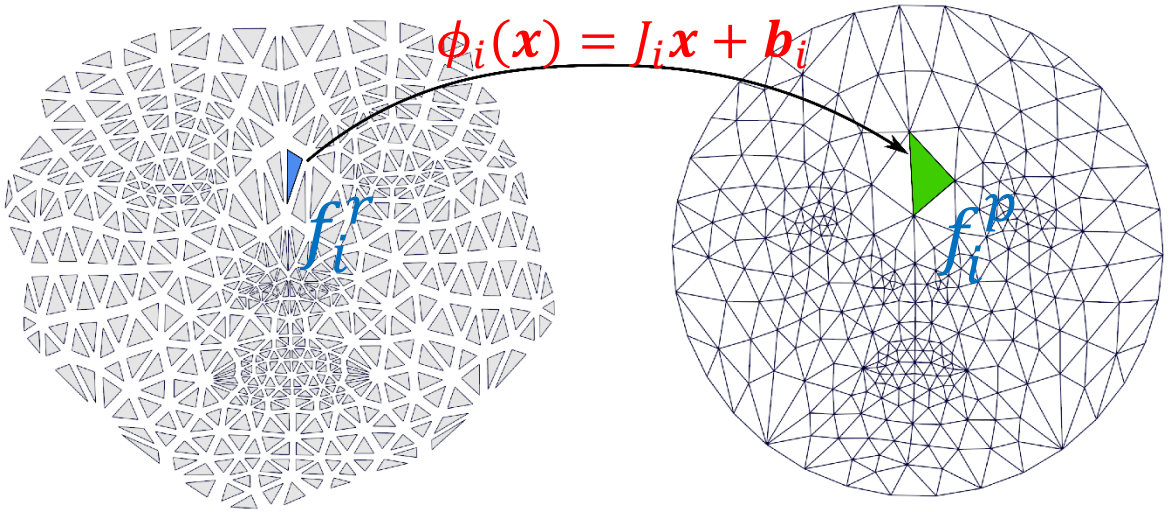


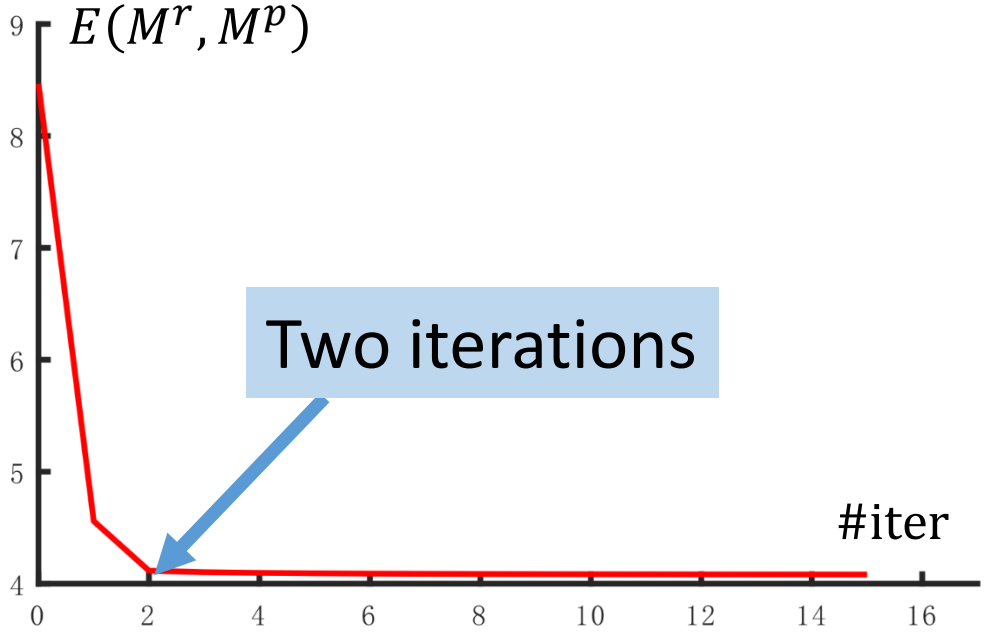
Reference M^r: A set of individual triangles

Parameterized mesh M^p

Exsiting methods choose *the triangles f_i of input mesh M* as reference triangles. The energy is *numerically difficult to optimize*, leading to numerous iterations and high computational cost.

Motivation





If $D\left(f_i^r, f_i^p\right) \leq K, \forall i$, only a few iterations in the optimization of $E(M^r, M^p)$ are necessary.

Goal: find a triangle between f_i and f_i^p as the reference f_i^r that satisfies $D\left(f_i^r, f_i^p\right) \leq K$.

New reference triangles

- Exponential function :

$$J_i(t) = U_i \mathrm{diag}(\sigma_i^t, \tau_i^t) V_i^T$$

where $J_i = U_i \mathrm{diag}(\sigma_i, \tau_i) V_i^T$

- Bounded distortion:

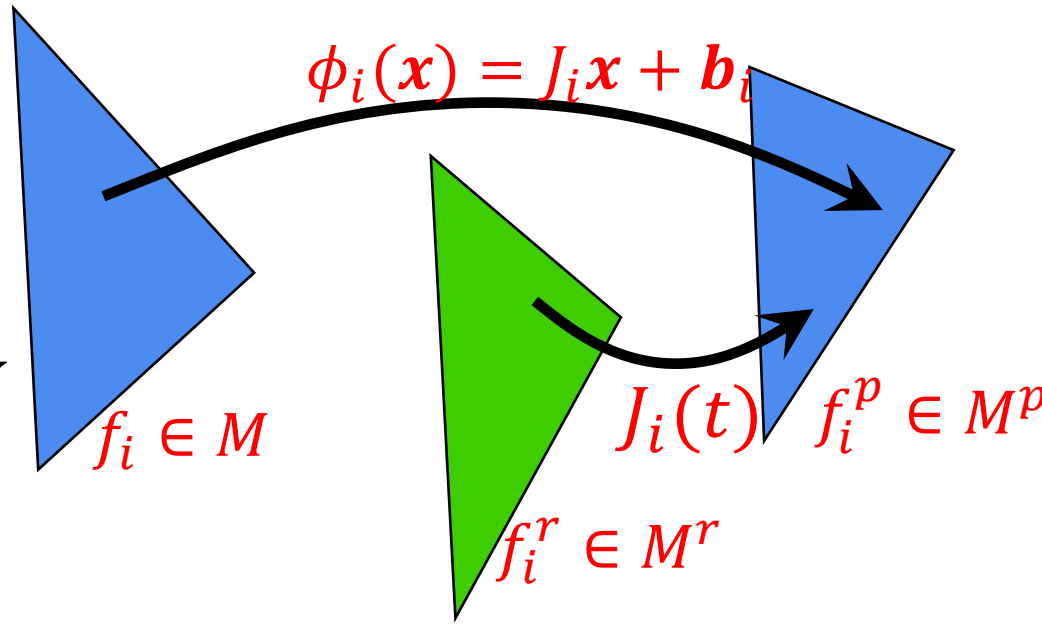
$$D(f_i^r, f_i^p) = \frac{1}{4}\left(\sigma_i^{2t} + \sigma_i^{-2t} + \tau_i^{2t} + \tau_i^{-2t}\right) \leq K$$

It is strictly increasing *w.r.t* t.

- Maximize the guidance of reference triangle:

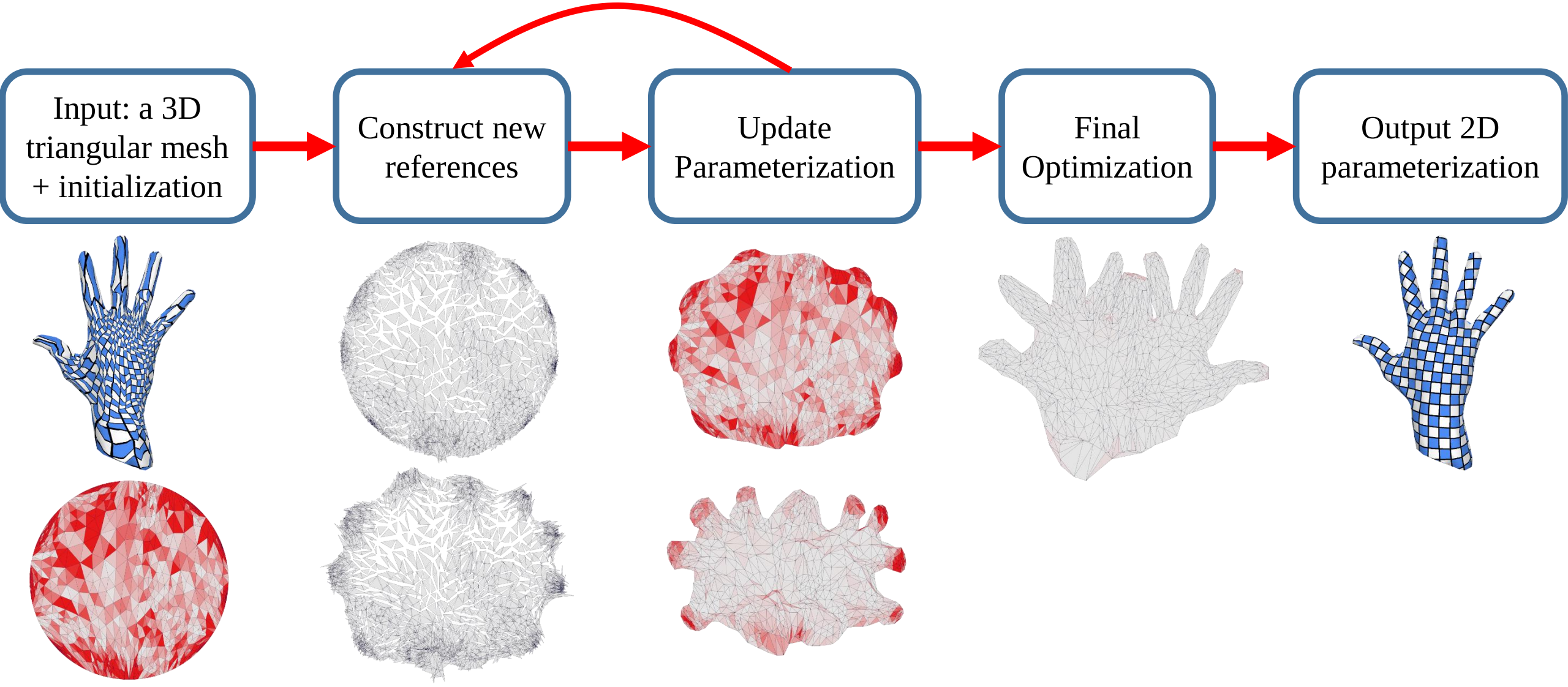
$$\frac{1}{4}\left(\sigma_i^{2t_i} + \sigma_i^{-2t_i} + \tau_i^{2t_i} + \tau_i^{-2t_i}\right) = K$$

Newton-Raphson method



Construction of new reference

Our algorithm – Progressive parameterization



Hybrid solver

- SLIM [Rabinovich et al. 2017]
  - A reweighting scheme
  - Pros: effectively penalize the maximum distortion
  - Cons: a poor convergence rate
- CM [Shtengel et al. 2017]
  - Pros: converge quickly
  - Cons: cannot reduce large distortion quickly
- Hybrid
  - First perform SLIM solver
  - Then use the CM solver

Result