中国科学技术大学
University of Science and Technology of China

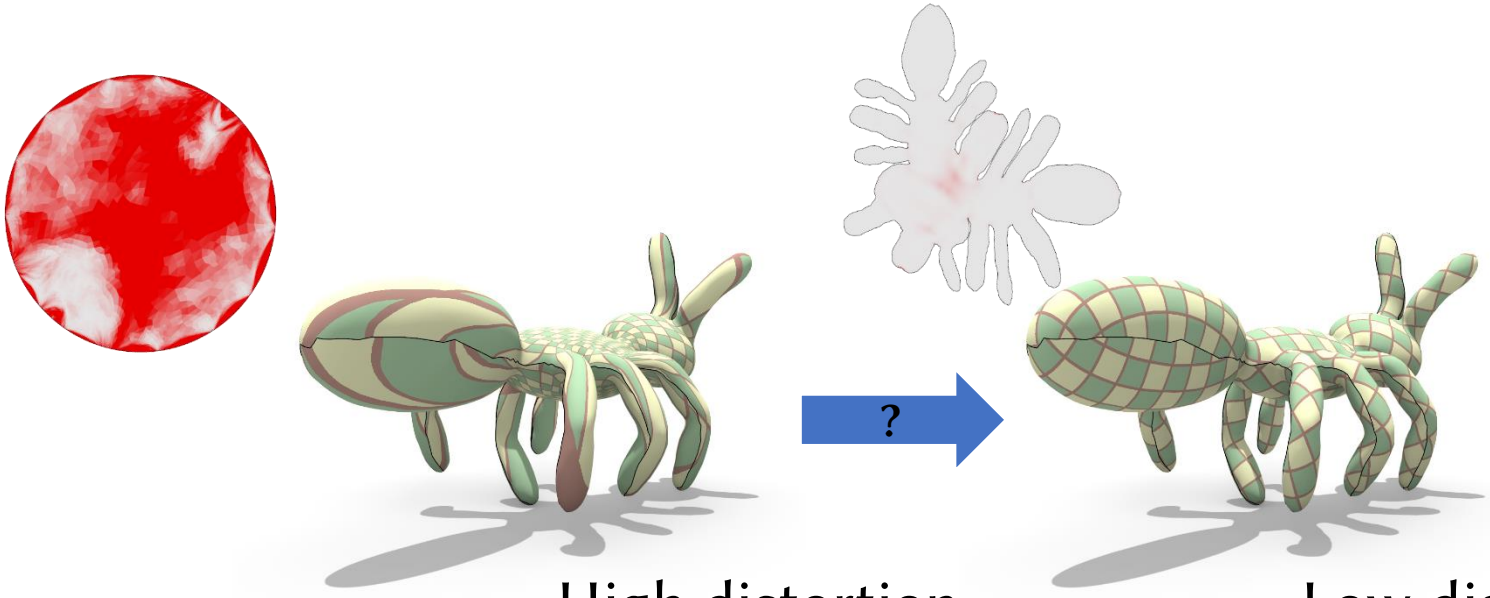
GAMES 301: 第4讲

无翻转参数化方法
初始无翻转

傅孝明
中国科学技术大学

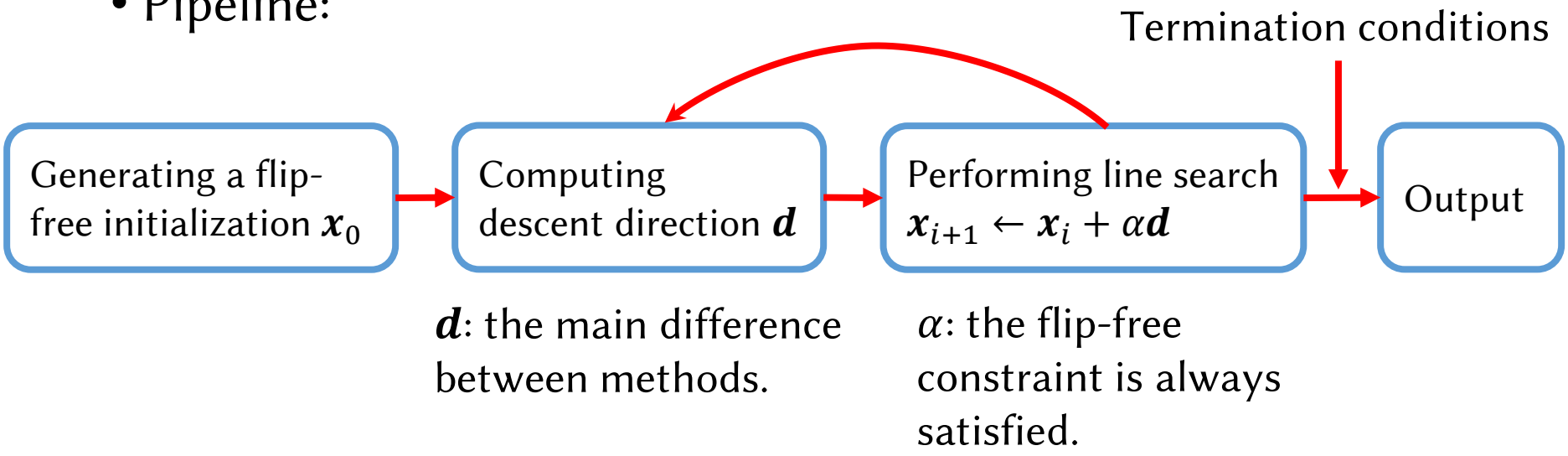
Computing flip-free mappings with flip-free initializations

Problem definition



General solution

- Key idea:
  - Starting from a flip-free initialization
  - Keeping mappings always staying in the flip-free space
- Pipeline:



Fu, Xiao-Ming, et al. "Inversion-free geometric mapping construction: A survey." *Computational Visual Media* 7.3 (2021): 289-318.

Barrier functions

- Barrier functions diverge to infinity when elements become degenerate, thus inhibiting flips.
- Auxiliary barrier
  - log of the determinant
- Distortion metrics
  - explode near degeneracies
  - MIPS: $\frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1} = \frac{\sigma_1^2+\sigma_2^2}{\sigma_1\sigma_2} = \frac{\|J_t\|_F^2}{\det J_t}$

Degenerate triangle t → $\det(J_t) \to 0$ → MIPS $\to \infty$

Descent directions

- Most approaches use a local quadratic approximation of the objective function E:

$$\tilde{E}(\boldsymbol{x}) = E(\boldsymbol{x}_i) + (\boldsymbol{x} - \boldsymbol{x}_i)^T \nabla E(\boldsymbol{x}_i) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_i)^T H_i (\boldsymbol{x} - \boldsymbol{x}_i)$$

- $\tilde{E}(\boldsymbol{x})$ is an osculating convex quadric approximation to E at $\boldsymbol{x}_i$.
- The minimization of $\tilde{E}(\boldsymbol{x})$ determines descent direction $\boldsymbol{d}$.
- The difference between the various methods lies in the choice of $\tilde{E}(\boldsymbol{x})$
  - More precisely, the choice of its proxy matrix H_i.

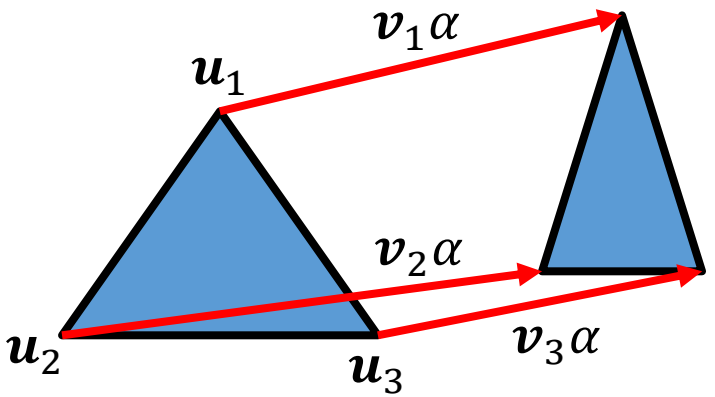
Three rough categories

- First-order methods
  - Only first derivatives
  - Do not directly use second-order derivatives
- Quasi-Newton methods
  - Iteratively update H_i to approximate second derivatives
  - Using just differences in gradients and variables
- Newton-type methods
  - Exploit expensive second-order derivative information

Line search

- Theoretical guarantee
  - Each triangle has a positive area
- The triangle $\Delta \boldsymbol{u}_1 \boldsymbol{u}_2 \boldsymbol{u}_3$ becomes degenerate when its signed area becomes zero:

$$\det \begin{bmatrix} (\boldsymbol{u}_2 + \boldsymbol{v}_2 \alpha) - (\boldsymbol{u}_1 + \boldsymbol{v}_1 \alpha) \\ (\boldsymbol{u}_3 + \boldsymbol{v}_2 \alpha) - (\boldsymbol{u}_1 + \boldsymbol{v}_1 \alpha) \end{bmatrix} = 0$$



- It is quadratic in α.
- The max step size for this triangle

= the smallest positive root.

- The max step size for all triangles

= the minimum parameter over all triangles .

Termination conditions

- The gradient is small $\|\nabla E\| \leq \varepsilon$.
- The absolute or relative error in energy and/or position are small.
- A fixed number of iterations.

First-order

Block coordinate descent

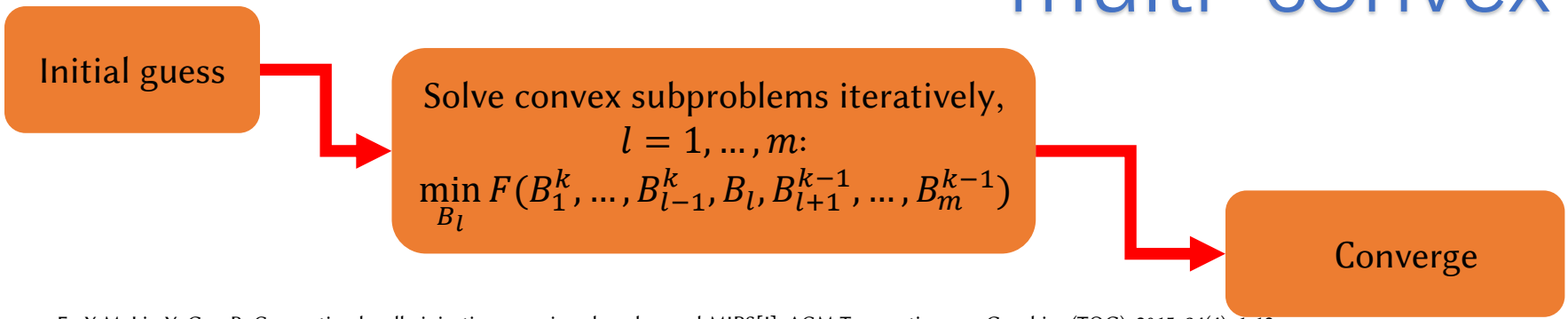
- BCD solver
- Optimization problem

$$\min_{\boldsymbol{x}} F(x_1, x_2, \dots, x_n)$$

$$\boldsymbol{x} = (x_1, x_2, \dots, x_n) \in \Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_m \subseteq \mathbb{R}^n$$

$$\boldsymbol{x} = \{B_1, B_2, \dots, B_m\}$$

multi-convex



Fu X M, Liu Y, Guo B. Computing locally injective mappings by advanced MIPS[J]. ACM Transactions on Graphics (TOG), 2015, 34(4): 1-12.

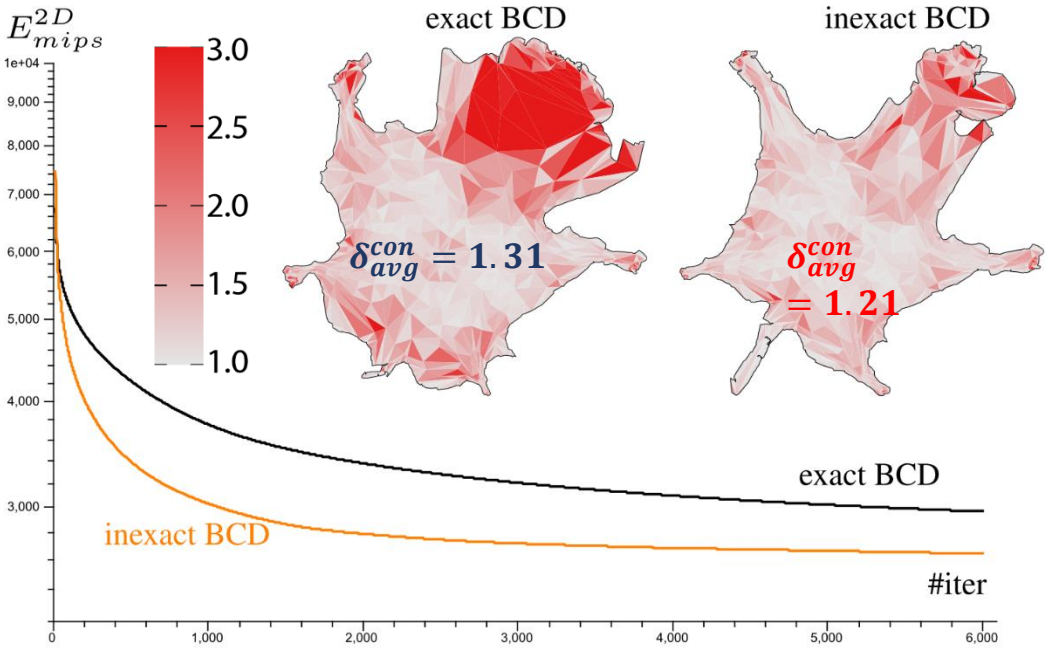
Inexact BCD vs. exact BCD

- Solver of subproblem

Exact: solve it until convergence

Inexact: one step of gradient descent [Bonettini 2011], [Tappenden et al. 2013], [Cassioli et al. 2013], [Xu and Yin 2013] ...

Efficiency: less computation
Lower objectives: not be trapped by local minimum easily



Inexact BCD for MIPS

- Initialization
  - convex combinatorial mapping for parameterization
  - Flip-free mesh for deformation, improvement
- Blocks of variables
  - parallelization (Graph coloring)
- Vertex updating
  - one step of gradient descent
- Stopping condition

Quadratic proxy

- Objective function: $f(\boldsymbol{x}) = h(\boldsymbol{x}) + g(\boldsymbol{x})$
- $h(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T H\boldsymbol{x}$
  - $h(\boldsymbol{x})$ is a quadratic function.
- $f(\boldsymbol{y}_n + \boldsymbol{p}) = h(\boldsymbol{y}_n + \boldsymbol{p}) + g(\boldsymbol{y}_n + \boldsymbol{p})$
  - $\boldsymbol{p}$ is the descent direction.
- $f(\boldsymbol{y}_n + \boldsymbol{p}) \approx h(\boldsymbol{y}_n + \boldsymbol{p}) + g(y_n) + \nabla g(y_n)^T p$
  - Tayler expansion of $g(\boldsymbol{y}_n + \boldsymbol{p})$
- Computing $\boldsymbol{p}$: $\nabla f(\boldsymbol{y}_n + \boldsymbol{p}) = H(\boldsymbol{y}_n + \boldsymbol{p}) + \nabla g(\boldsymbol{y}_n) = 0$

$$H\boldsymbol{p} = -H\boldsymbol{y}_n - \nabla g(\boldsymbol{y}_n) = -\nabla f(\boldsymbol{y}_n)$$

Kovalsky S Z, Galun M, Lipman Y. Accelerated quadratic proxy for geometric optimization[J]. ACM Transactions on Graphics (TOG), 2016, 35(4): 1-11.

Accelerated quadratic proxy

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 */

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

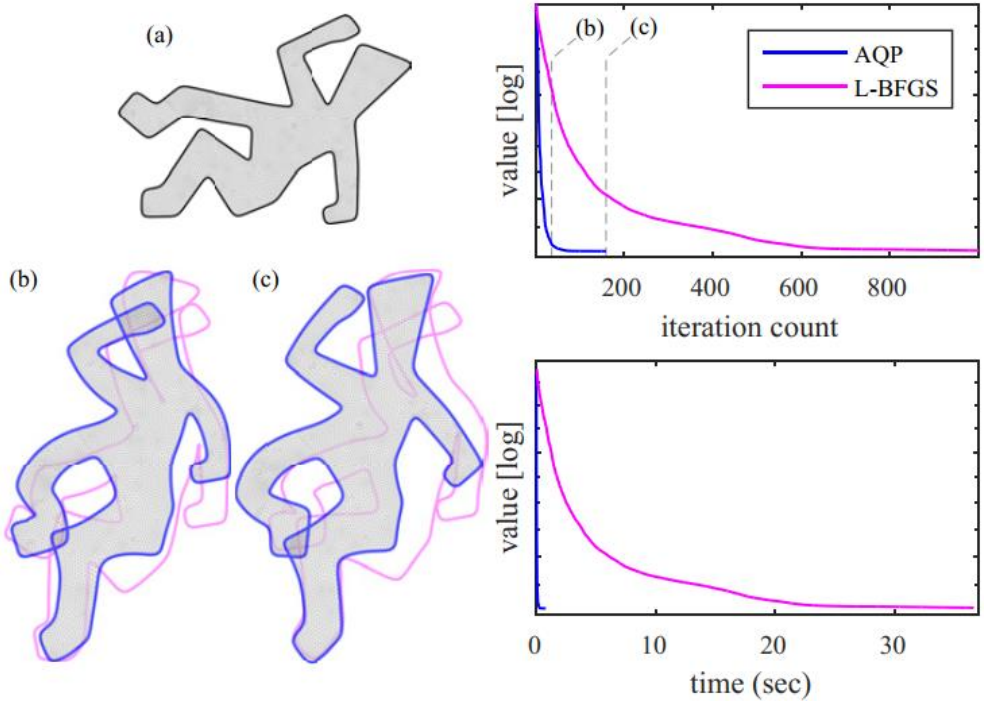
/* Quadratic proxy minimization */

$$\mathbf{p}_n = \arg\min_{\mathbf{p}} \quad h(\mathbf{y}_n + \mathbf{p}) + g(\mathbf{y}_n) + \nabla g(\mathbf{y}_n)\mathbf{p}$$
$$\text{s.t.} \quad A\mathbf{p} = 0$$

/* Line search */

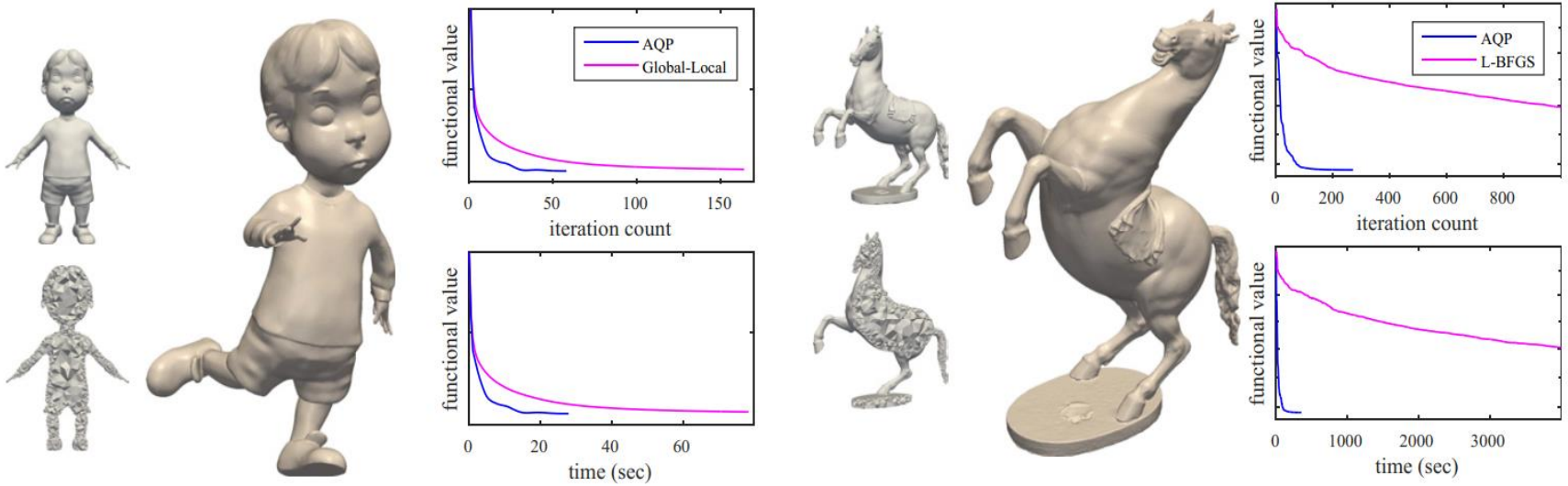
$\mathbf{x}_n = \text{linesearch}_{\,0<t\leq 1} \; f(\mathbf{y}_n + t\mathbf{p}_n)$

Results


(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


AQP
Global-Local
functional value
iteration count
time (sec)
AQP
L-BFGS
functional value
iteration count
time (sec)

Scalable Locally Injective Mappings

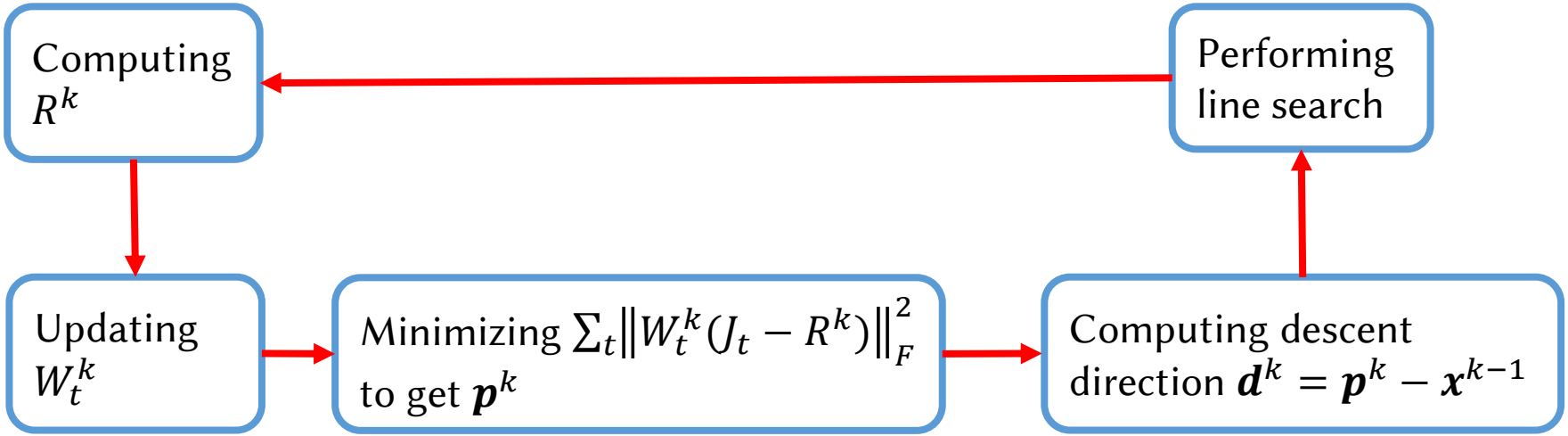
- Objective function $\sum \text{Area}(A_t)\, D(J_t)$
  - E.g., symmetric Dirichlet $D(J_t) = \|J_t\|_F^2 + \|J_t^{-1}\|_F^2$
- The proxy function: $\left\|W_t^k (J_t - R^k)\right\|_F^2$
  - W_t^k: matrix weight
  - k: iteration number
  - R^k: the projected matrix from J_t^{k-1} to the rotation matrix space
- Condition: $\nabla_{J_t} \left\|W_t^k (J_t - R^k)\right\|_F^2 = \nabla_{J_t} D(J_t)$

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

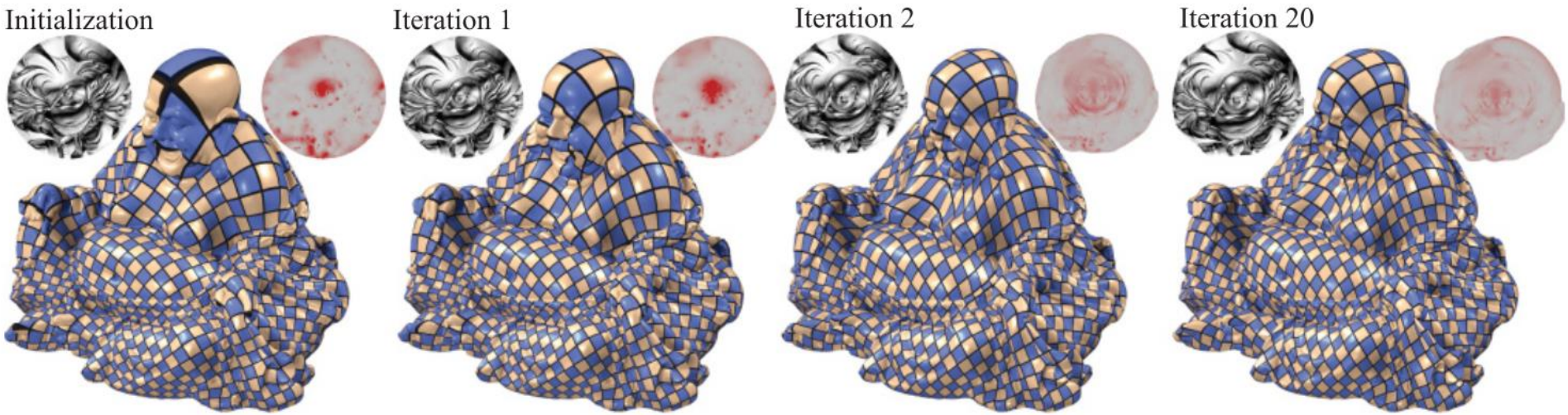
$$W = \left(\frac{1}{2} \nabla_J D(J)(J - R)^{-1}\right)^{1/2}$$

Rabinovich M, Poranne R, Panozzo D, et al. Scalable locally injective mappings[J]. ACM Transactions on Graphics (TOG), 2017, 36(4): 1.

Scalable Locally Injective Mappings



Results



Quasi-Newton

L-BFGS

- Secant equation:

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

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

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

$\boldsymbol{s}_n$: difference in positions, $\boldsymbol{y}_n$: difference in gradients

$$H_{n+1}\boldsymbol{s}_n = \boldsymbol{y}_n, \boldsymbol{s}_n = D_{n+1}\boldsymbol{y}_n, D_{n+1} = H_{n+1}^{-1}$$

- Rank-two modification

$$D_{n+1} = \left(I - \frac{\boldsymbol{y}_n \boldsymbol{s}_n^T}{\boldsymbol{s}_n^T \boldsymbol{y}_n}\right)^T D_n (I - \frac{\boldsymbol{y}_n \boldsymbol{s}_n^T}{\boldsymbol{s}_n^T \boldsymbol{y}_n}) + \frac{\boldsymbol{s}_n \boldsymbol{s}_n^T}{\boldsymbol{s}_n^T \boldsymbol{y}_n}$$

- Descent direction

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

BCQN

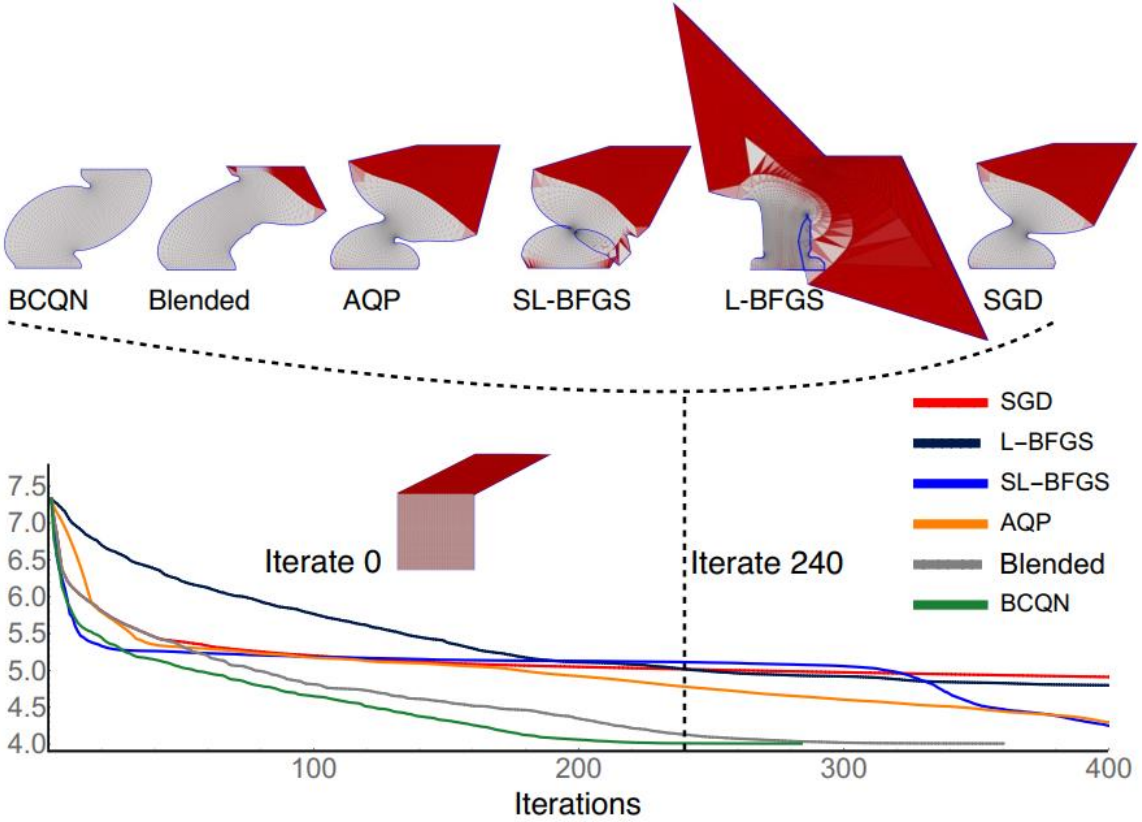
- In mesh-based mappings, the Laplacian L is often a much more effective proxy than the L-BFGS secant version.
  - Near distorted triangles, $\boldsymbol{y}_n = \nabla f(\boldsymbol{x}_{n+1}) - \nabla f(\boldsymbol{x}_n)$ may introduce spurious coupling or have badly scaled values.
  - $L\boldsymbol{s}_n = L(\boldsymbol{x}_{n+1} - \boldsymbol{x}_n)$ would be the better behaved than $\boldsymbol{y}_n$.
  - Replacing $\boldsymbol{y}_n$ with $L\boldsymbol{s}_n$.
- To achieve the super-linear convergence of L-BFGS near the solution:

$$\boldsymbol{y}_n \leftarrow (1 - \beta_n)\boldsymbol{y}_n + \beta_n L\boldsymbol{s}_n$$

Extensive testing to determine β_n.

Zhu Y, Bridson R, Kaufman D M. Blended cured quasi-newton for distortion optimization[J]. ACM Transactions on Graphics (TOG), 2018, 37(4): 1-14.

Results


BCQN
Blended
AQP
SL-BFGS
L-BFGS
SGD
Iterate 0
Iterate 240
SGD
L-BFGS
SL-BFGS
AQP
Blended
BCQN
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
100
200
300
400
Iterations

Second-order

Overview

- Second-order methods generally can achieve the most rapid convergence.
- But require the costly assembly, factorization, and backsolve of new linear systems per step.
- Second-order methods use the energy Hessian, $\nabla^2 E$ to form a proxy matrix H.
  - Work for convex energies.
  - Require modification for non-convex energies to ensure that H is at least positive semi-definite to achieve a descent direction.
- General solution: add small multiples of the identity
  - generally damp convergence too much

Overview

- The common objective form: $\sum_t D_t$
  - Sum over all locally individual elements (triangle or tetrahedra).
- The global Hessian matrix of the objective function is constructed from the element Hessian matrix.
- As long as the Hessian matrices of all elements are PSD, then the global Hessian matrix is PSD.
- Thus, most second-order methods locally modify the element Hessian matrices

$$H^+ = \sum_t H_t^+$$

the dimensions of H_t^+ are far lower than H^+.

Projected Newton

- Eigendecomposition on per-element Hessian $\nabla^2 D_t$:

$$\nabla^2 D_t = V \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} V^T$$

- Clamps all negative eigenvalues to zero to project per-element Hessian to the PSD H_t^+.
- Shortcomings:
  - eigendecomposition is time-consuming.
- Goal of the next methods:
  - analytically achieve H_t^+.

Analytic Eigensystem

- The majority of distortion energies used in geometry optimization are isotropic and can be expressed using the following invariants:
  - Polar decomposition: $J_t = R_t S_t$
  - The first invariant I_1 sums one-dimensional stretches: $I_1 = \mathrm{tr}(S_t) = \sum_i \sigma_i$
  - I_2 is the Frobenius squared norm of J_t: $I_2 = \|S_t\|^2 = \sum_i \sigma_i^2$
  - The third invariant I_3 encodes the volume change: $I_3 = \det(S_t) = \Pi_i \sigma_i$
- Example: $D_{\mathrm{ARAP}}(J_t) = \sum_i (\sigma_i - 1)^2 = I_2 - 2I_1 + d$
- Closed-form expressions for the eigensystems for invariants
  - systematically derive the eigensystems of any isotropic energy.
  - these systems can be used to project energy Hessian to PSD analytically.

Smith B, Goes F D, Kim T. Analytic eigensystems for isotropic distortion energies[J]. ACM Transactions on Graphics (TOG), 2019, 38(1): 1-15.

Problem

- Energy form:

$$f(\boldsymbol{x}) = \sum_i h_i\left(\boldsymbol{g}_i(\boldsymbol{x})\right) = \sum_i (h_i \circ \boldsymbol{g}_i)(\boldsymbol{x}),\ f: \mathbb{R}^n \longrightarrow \mathbb{R}, h_i: \mathbb{R}^k \longrightarrow \mathbb{R}, \boldsymbol{g}_i: \mathbb{R}^n \longrightarrow \mathbb{R}^k$$

- Convex-concave decompositions:

$h_i = h_i^+ + h_{\bar{i}}, \boldsymbol{g}_i = \boldsymbol{g}_i^+ + \boldsymbol{g}_{\bar{i}}$ with h_i^+ convex and the vector-valued functions $\boldsymbol{g}_i^+$ convex in each of their coordinates and, respectively, $h_{\bar{i}}$ and $\boldsymbol{g}_{\bar{i}}$ concave.

- Example:

Given $J_i = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}$, we set $\alpha_i = \frac{1}{2}\begin{bmatrix} a_i + d_i \\ c_i - b_i \end{bmatrix}$ and $\beta_i = \frac{1}{2}\begin{bmatrix} a_i - d_i \\ c_i + b_i \end{bmatrix}$.

Then, we have: $\sigma_1 = \|\alpha_i\|_2 + \|\beta_i\|_2$ and $\sigma_2 = \|\alpha_i\|_2 - \|\beta_i\|_2$.

Finally, $\boldsymbol{g}_i = (\sigma_1, \sigma_2), \boldsymbol{g}_i^+ = (\|\alpha_i\|_2 + \|\beta_i\|_2, \|\alpha_i\|_2), \boldsymbol{g}_{\bar{i}} = (0, -\|\beta_i\|_2)$

Symmetric Dirichlet energy: $h_i\left(\boldsymbol{g}_i(\boldsymbol{x})\right) = \mathrm{Area}(t_i)(\sigma_1^2 + \sigma_2^2 + \sigma_1^{-2} + \sigma_2^{-2})$ is convex with respect to $\sigma_1, \sigma_2 > 0$, thus $h_i = h_i^+$.

Shtengel A, Poranne R, Sorkine-Hornung O, et al. Geometric optimization via composite majorization[J]. ACM Trans. Graph., 2017, 36(4): 38:1-38:11.

Majorization-Minimization

- Majorization Minimization (MM)
  - at each iteration, fitting a convex surrogate upper bound, called a majorizer, which is minimized to drive the objective function downhill.
  - replacing a difficult optimization with a sequence of simpler optimizations.
- Given a scalar function r:

The majorizer at $\boldsymbol{x}_0$: $\overline{r}(\boldsymbol{x};\boldsymbol{x}_0) = r^+(\boldsymbol{x}) + r^-(\boldsymbol{x}_0) + \nabla r^-(\boldsymbol{x}_0)(\boldsymbol{x}-\boldsymbol{x}_0)$

The minorizer at $\boldsymbol{x}_0$: $\underline{r}(\boldsymbol{x};\boldsymbol{x}_0) = r^-(\boldsymbol{x}) + r^+(\boldsymbol{x}_0) + \nabla r^+(\boldsymbol{x}_0)(\boldsymbol{x}-\boldsymbol{x}_0)$

- $\overline{r}$ ($\underline{r}$) satisfies:
  - It is a convex (concave) function.
  - It coincides with r up to first order(i.e. , value and first derivative) at $\boldsymbol{x} = \boldsymbol{x}_0$.
  - It is a global majorizer (minorizer) of r: $\underline{r}(\boldsymbol{x};\boldsymbol{x}_0) \leq r(x) \leq \overline{r}(\boldsymbol{x};\boldsymbol{x}_0), \forall \boldsymbol{x}$.
- A natural idea: approximate Hessian to be the Hessian of the convex majorizer.

A convex majorizer

- A convex majorizer to f centered at a general point $\boldsymbol{x}_0$:

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

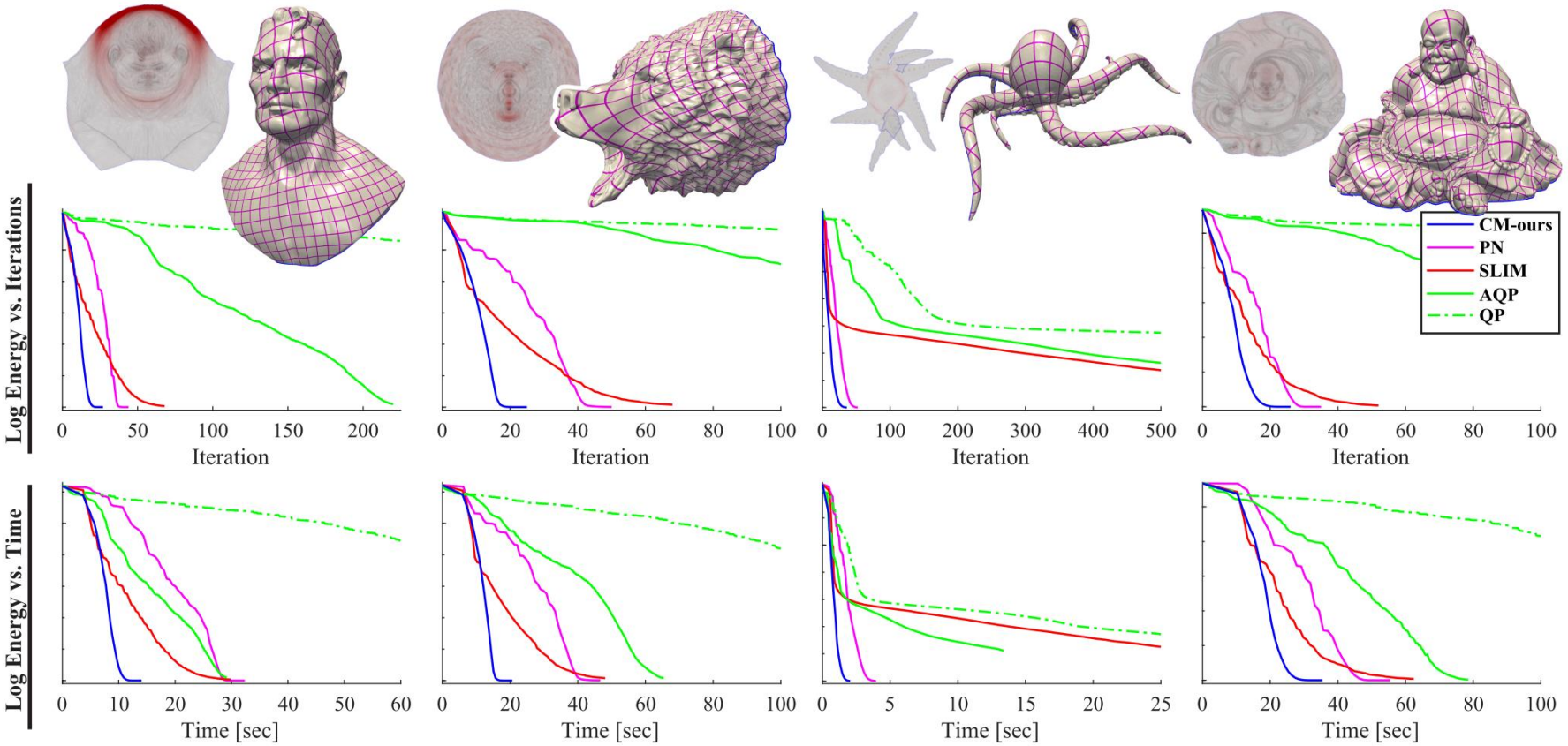
$\overline{h}_i$ is a majorizer of h_i centered at $\boldsymbol{g}_i(\boldsymbol{x}_0)$ and $[\boldsymbol{g}_i](\boldsymbol{x};\boldsymbol{x}_0)$ is a vector function whose entries are either majorizers or minorizers of $\boldsymbol{g}_i$ centered at $\boldsymbol{x}_0$.

For one entry $g_{i,j}$ of $\boldsymbol{g}_i$, we define:

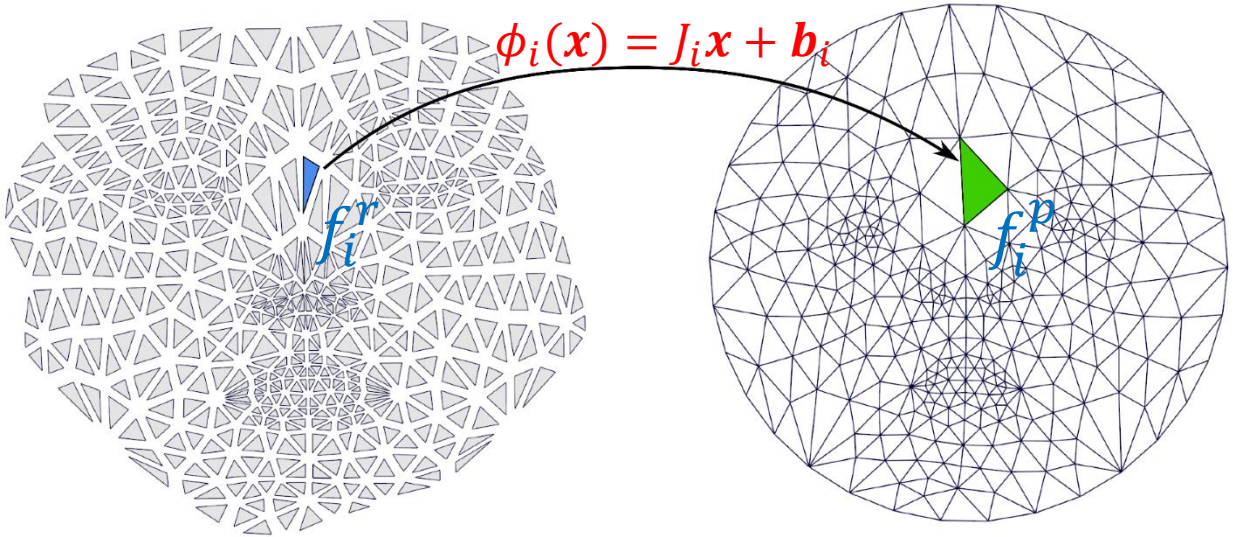
$$[g_{i,j}](\boldsymbol{x};\boldsymbol{x}_0)=\begin{cases}\overline{g}_{i,j}(\boldsymbol{x};\boldsymbol{x}_0) & \frac{\partial\overline{h}_i}{\partial g_{i,j}}(\boldsymbol{g}_i(\boldsymbol{x});\boldsymbol{g}_i(\boldsymbol{x}_0))\geq 0\\ \underline{g}_{i,j}(\boldsymbol{x};\boldsymbol{x}_0) & \frac{\partial\overline{h}_i}{\partial g_{i,j}}(\boldsymbol{g}_i(\boldsymbol{x});\boldsymbol{g}_i(\boldsymbol{x}_0))<0\end{cases},$$

Proof: $\nabla_x^2\overline{f}(\boldsymbol{x};\boldsymbol{x}_0)=\frac{\partial[\boldsymbol{g}]^T}{\partial\boldsymbol{x}}\nabla^2h^+\frac{\partial[\boldsymbol{g}]}{\partial\boldsymbol{x}}+\sum_j\frac{\partial\overline{h}}{\partial g_{i,j}}\begin{cases}\nabla^2g_j^+ & \frac{\partial\overline{h}_i}{\partial g_{i,j}}(\boldsymbol{g}_i(\boldsymbol{x});\boldsymbol{g}_i(\boldsymbol{x}_0))\geq 0\\ \nabla^2g_j^- & \frac{\partial\overline{h}_i}{\partial g_{i,j}}(\boldsymbol{g}_i(\boldsymbol{x});\boldsymbol{g}_i(\boldsymbol{x}_0))<0\end{cases}$

Results


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

Progressive Parameterizations



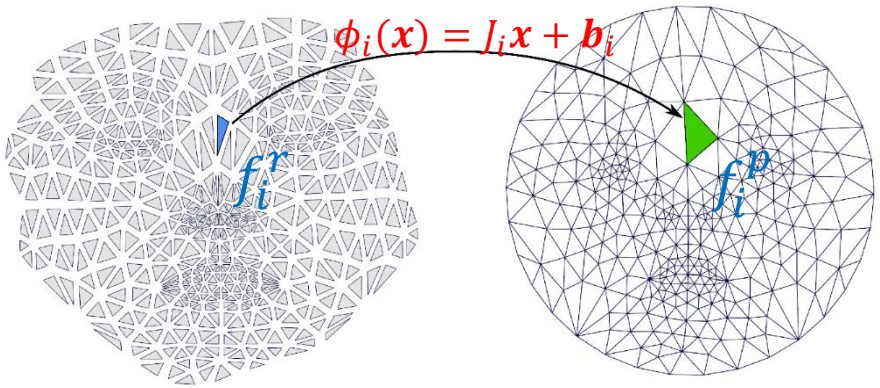
Reference M^r: A set of individual triangles

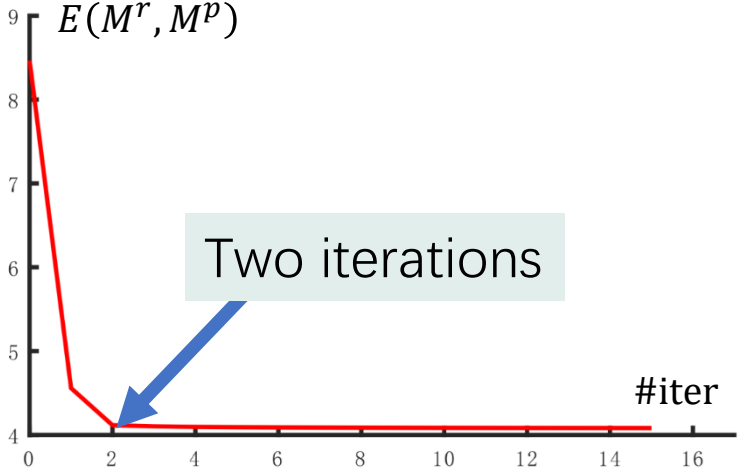
Parameterized mesh M^p

Existing 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 costs.

Liu L, Ye C, Ni R, et al. Progressive parameterizations[J]. ACM Trans. Graph., 2018, 37(4): 41:1-41:12.

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:

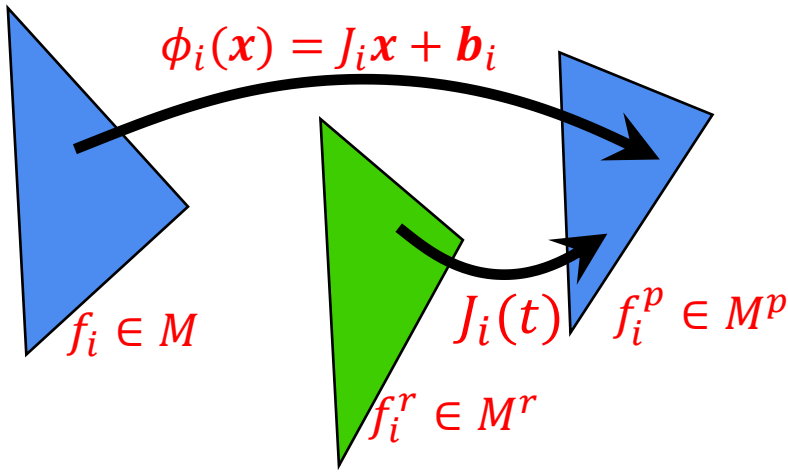
$$D(f_i^r, f_i^p) = \left(\sigma_i^{2t} + \sigma_i^{-2t} + \tau_i^{2t} + \tau_i^{-2t}\right)/4 \leq K$$

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

- Maximize the guidance of reference triangle:

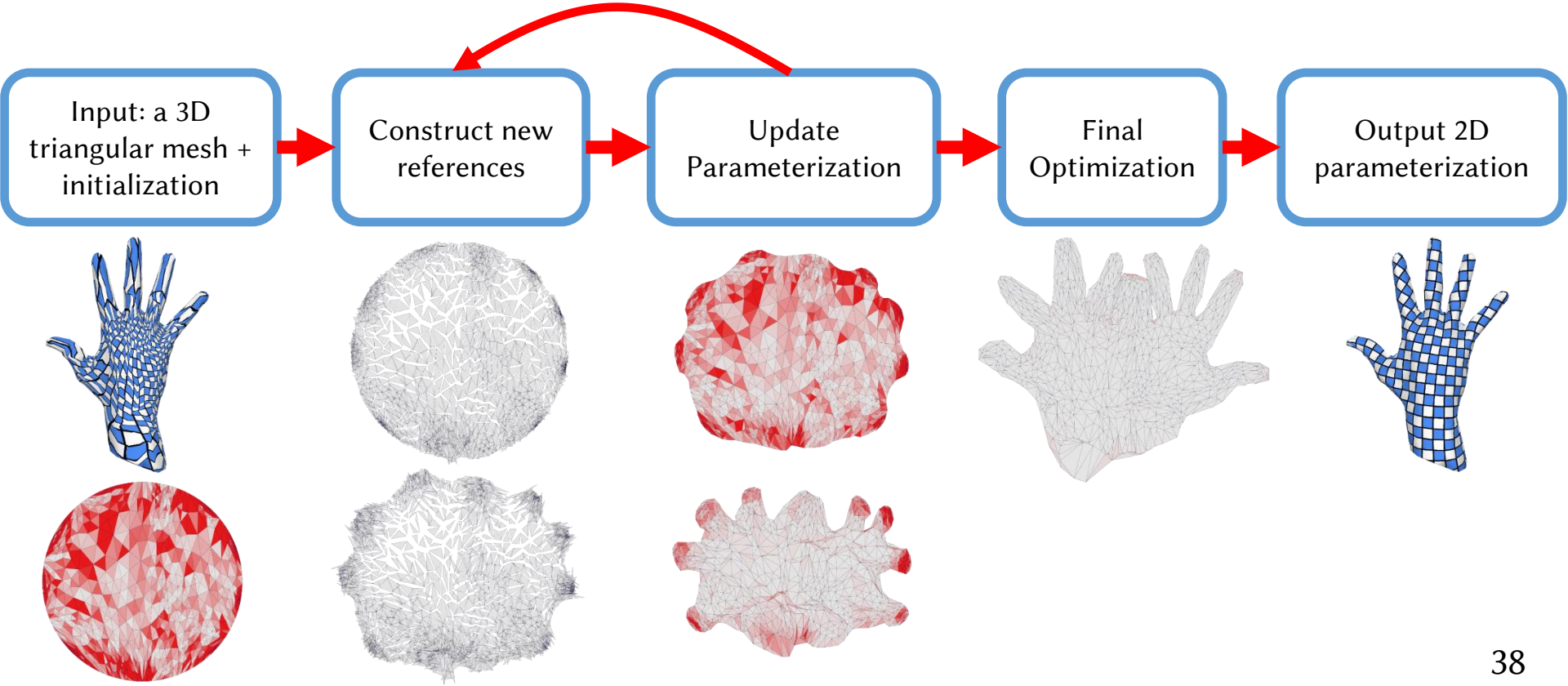
$$\left(\sigma_i^{2t} + \sigma_i^{-2t} + \tau_i^{2t} + \tau_i^{-2t}\right)/4 = K$$

Newton-Raphson method



Construction of new reference

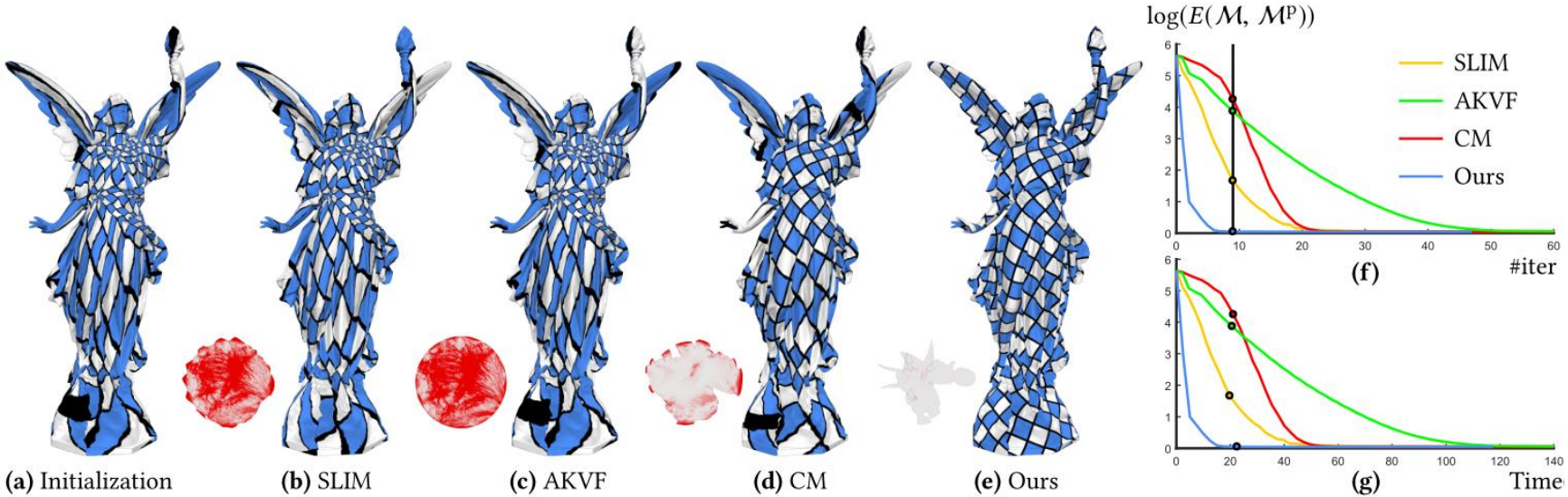
Pipeline


Input: a 3D triangular mesh + initialization
Construct new references
Update Parameterization
Final Optimization
Output 2D 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

Results



(a) Initialization (b) SLIM (c) AKVF (d) CM (e) Ours

Connectivity-updated methods

Formulation

$$E(u,\mathcal{T}) = E_d(u,\mathcal{T}) + E_c(u) + E_b(u,\mathcal{T})$$

Distortion energy:

$$E_d(u,\mathcal{T}) = \sum_t (\sigma_1 - 1)^2 + (\sigma_2 - 1)^2$$

Position constraints:

$$E_c(u) = \sum_i \|u_i - p_i\|^2$$

Barrier functions:

$$E\_b\ (u,\mathcal{T}) = \sum_t -\log(A_t)$$

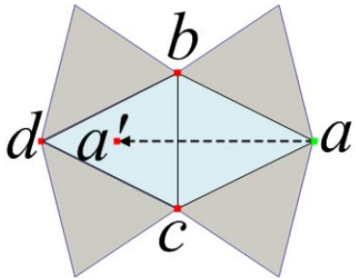
Jin Y, Huang J, Tong R. Remeshing-assisted optimization for locally injective mappings[C]//Computer Graphics Forum. 2014, 33(5): 269-279.

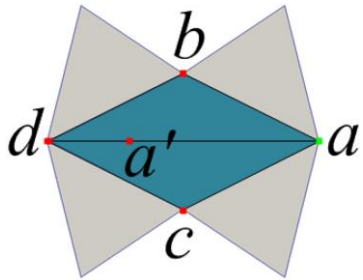
Remeshing

$$E(u,\mathcal{T}) = E_d(u,\mathcal{T}) + E_c(u) + E_b(u,\mathcal{T})$$

$\Delta_{bc} = E_{bc} - \bar{E}_{bc}$

$\Delta_{bc} > \eta E_{bc}$





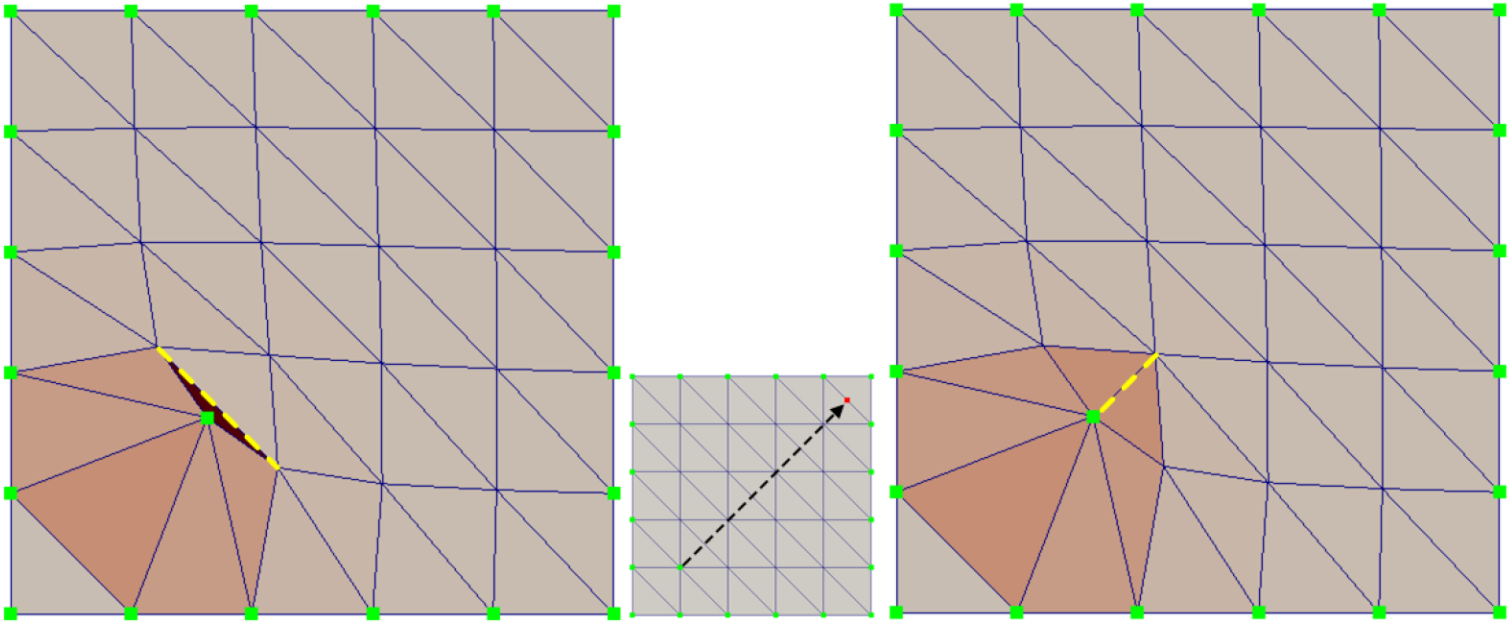
$$\text{cond}(E) = \frac{\|\nabla E\|_\infty \|u\|_\infty}{E}$$

$\text{cond}(E_{bc}) > \text{cond}(\bar{E}_{bc})$

$\mathcal{T}_{bc} = \{\Delta_{cbd}, \Delta_{cab}\}$ $\quad \bar{\mathcal{T}}_{bc} = \{\Delta_{dab}, \Delta_{dca}\}$

$A_t > 0, t \in \{\Delta_{dab}, \Delta_{dca}\}$

Remeshing



ARAP Energy Deformation for the Woody Shape (Without Flip v.s. With Flip)

中国科学技术大学

University of Science and Technology of China

谢 谢 ！