It's a small step for man, but...

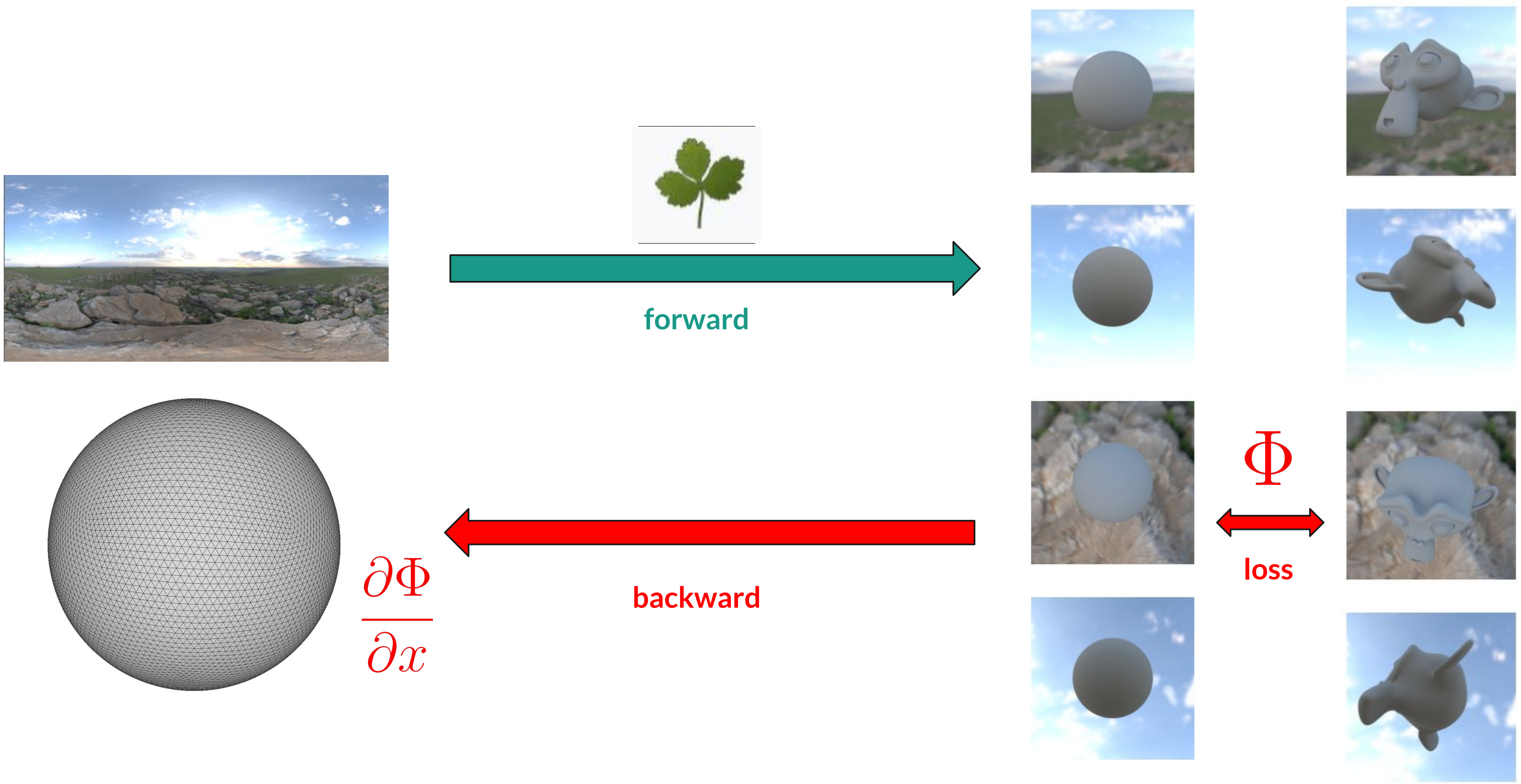
Large Steps in Inverse Rendering of Geometry

Baptiste Nicolet, Alec Jacobson, Wenzel Jakob, 2021

GDL 04 Juin, Gaspard Thévenon

Differentiable rendering, again

Differentiable rendering, again



Gradient descent

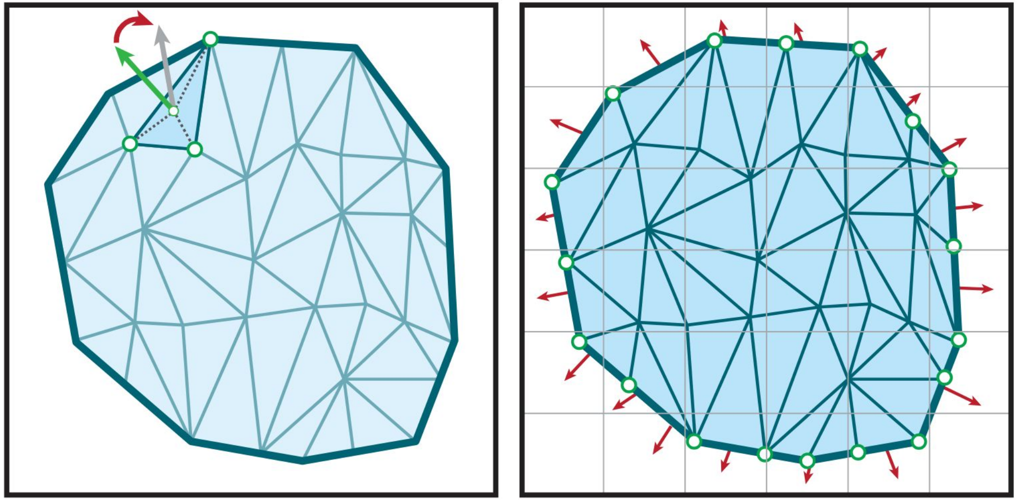
Goal: $\underset{\mathbf{x} \in \mathbb{R}^{n \times 3}}{\text{minimize}} \quad \Phi(R(\mathbf{x}))$

GD step: $\mathbf{x} \leftarrow \mathbf{x} - \eta \frac{\partial \Phi}{\partial \mathbf{x}}$

Φ: loss function
R: rendering function
x: vertex positions

η: learning rate

Gradients w/r to positions



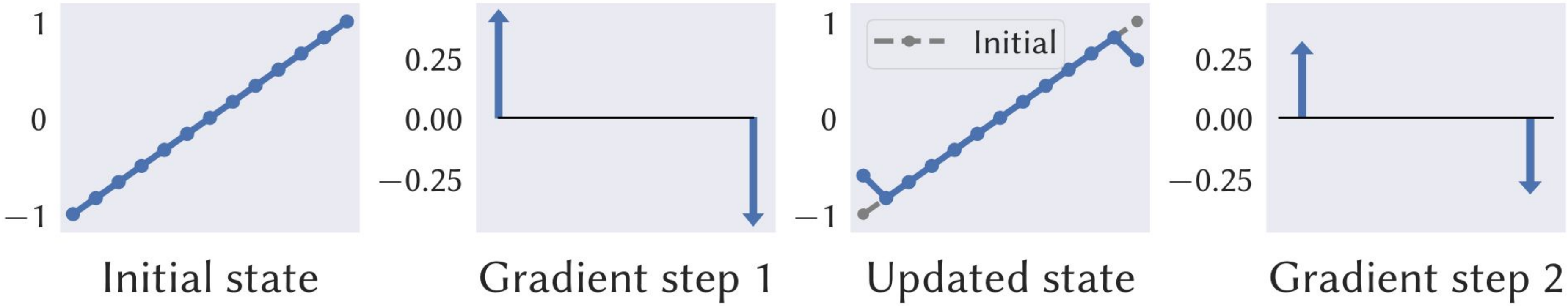
(a) Shading gradients

(b) Silhouette gradients

Smooth, small in magnitude

Sparse, very large in magnitude

In theory, why it doesn't work



Toy 1D "mesh" example:

We start from a linear progression in [-1, 1], and target a linear progression from [-0.5, 0.5].
We assume that we only have access to **sparse "silhouette" gradients**.

Result: within a few iterations, we'll have a **tangled shape, with multiple inverted elements**.

Indeed, it doesn't work

Sparse gradients

+

Connectivity ignored

Laplacian operator and Dirichlet energy

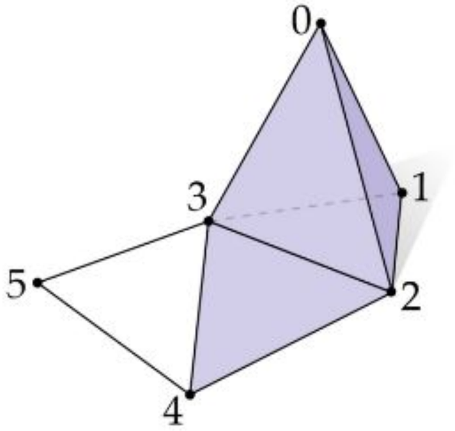
Laplacian operator on a scalar field:

$$\Delta f = \frac{\partial^2 f}{\partial x_1^2} + \cdots + \frac{\partial^2 f}{\partial x_n^2}$$

Dirichlet energy:

$$E(f) := \frac{1}{2} \int_\Omega \|\nabla f\|^2 \, \mathrm{d}\mathbf{x} = C - \frac{1}{2} \langle f, \Delta f \rangle$$

Discrete Laplacian operator

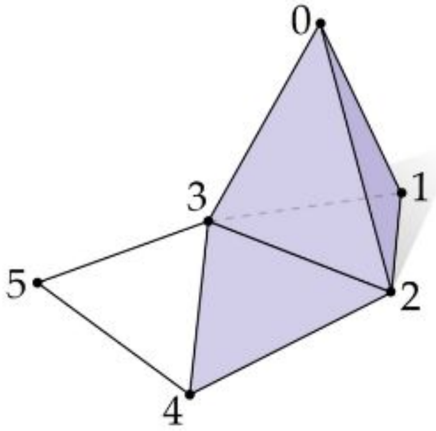


$$\mathbf{L}_{ij} = \begin{bmatrix} -\sum & w_{01} & w_{02} & w_{03} & 0 & 0 \\ w_{10} & -\sum & w_{12} & w_{13} & 0 & 0 \\ w_{20} & w_{21} & -\sum & w_{23} & w_{24} & 0 \\ w_{30} & w_{31} & w_{32} & -\sum & w_{34} & w_{35} \\ 0 & 0 & w_{42} & w_{43} & -\sum & w_{45} \\ 0 & 0 & 0 & w_{53} & w_{54} & -\sum \end{bmatrix}$$

Combinatorial Laplacian operator: the (non-zero) weights are equal to 1

(there are other usual definitions, but this one is sufficient here)

Discrete Laplacian operator



$$\mathbf{L}_{ij} = \begin{bmatrix} -\sum & w_{01} & w_{02} & w_{03} & 0 & 0 \\ w_{10} & -\sum & w_{12} & w_{13} & 0 & 0 \\ w_{20} & w_{21} & -\sum & w_{23} & w_{24} & 0 \\ w_{30} & w_{31} & w_{32} & -\sum & w_{34} & w_{35} \\ 0 & 0 & w_{42} & w_{43} & -\sum & w_{45} \\ 0 & 0 & 0 & w_{53} & w_{54} & -\sum \end{bmatrix}$$

L is a *sparse* operator.

We can also define the Dirichlet energy in terms of **L**:

$$E(\mathbf{f}) = \frac{1}{2}\langle \mathbf{f}, \mathbf{L}\mathbf{f} \rangle = \frac{1}{2}\mathbf{f}^T \mathbf{L}\mathbf{f}$$

Common solution - Loss regularization

New objective:

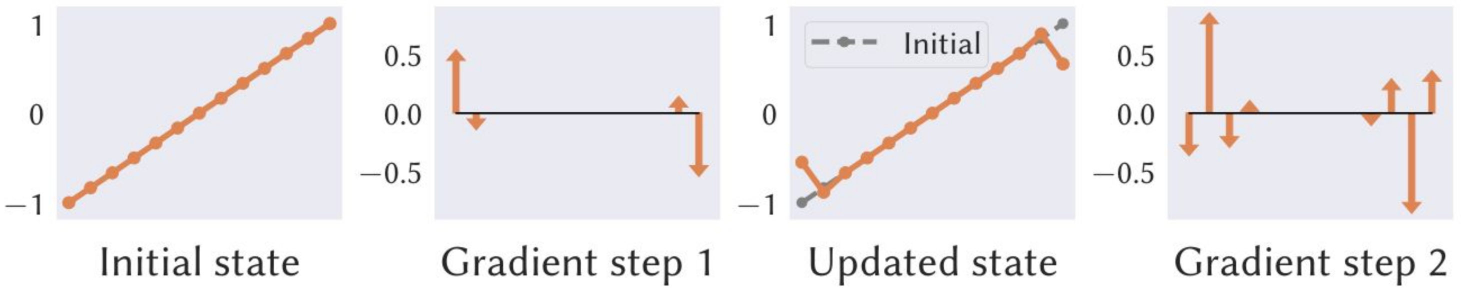
$$\min_{\mathbf{x}\in\Omega} \underbrace{\boldsymbol{\Phi}(\mathbf{x})}_{\text{"True" objective}} + \underbrace{\lambda \mathbf{x}^{\top} \mathbf{L}\, \mathbf{x}}_{\text{Regularization term}}$$

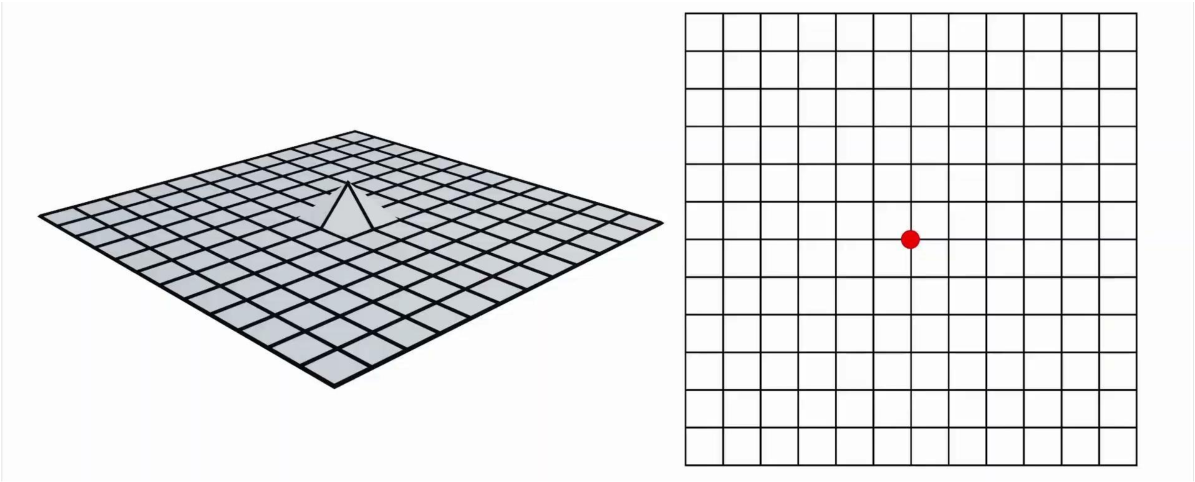
New gradient descent step:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \left(\frac{\partial \Phi}{\partial \mathbf{x}} + \lambda \mathbf{L}\mathbf{x} \right)$$

Common solution - Loss regularization -> not magic

Common solution - Loss regularization -> not magic

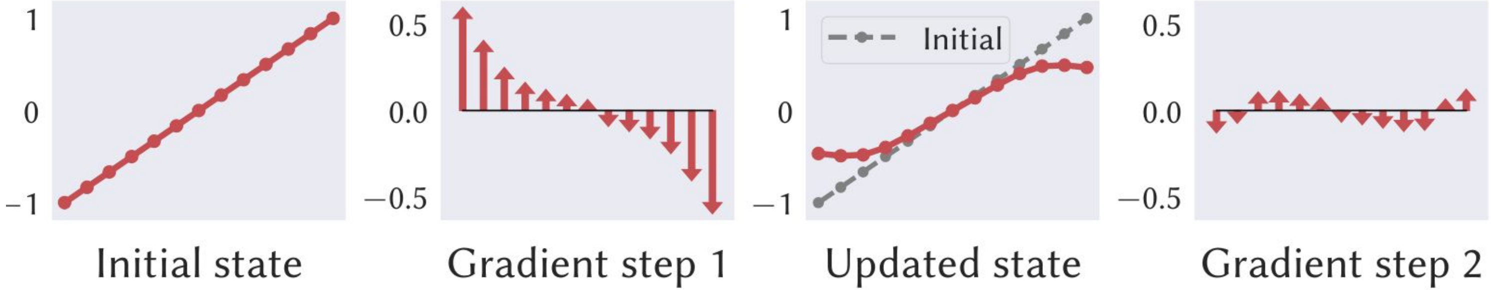




Second-order - Newton's method

Equivalent optimisation step:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \left(\frac{\partial^2 \Phi}{\partial \mathbf{x}^2} + \lambda \mathbf{L} \right)^{-1} \left(\frac{\partial \Phi}{\partial \mathbf{x}} + \lambda \mathbf{L} \mathbf{x} \right)$$

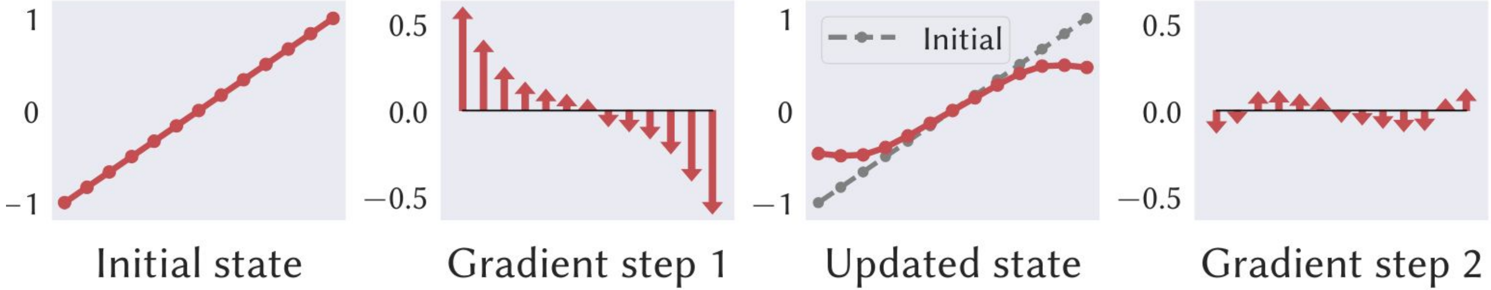


Second-order - Newton's method

Equivalent optimisation step: $$\mathbf{x} \leftarrow \mathbf{x} - \eta \boxed{\left(\frac{\partial^2 \Phi}{\partial \mathbf{x}^2} + \lambda \mathbf{L}\right)^{-1}} \left(\frac{\partial \Phi}{\partial \mathbf{x}} + \lambda \mathbf{L}\mathbf{x}\right)$$

but no hope of having access to that

propagates gradient updates to the whole mesh



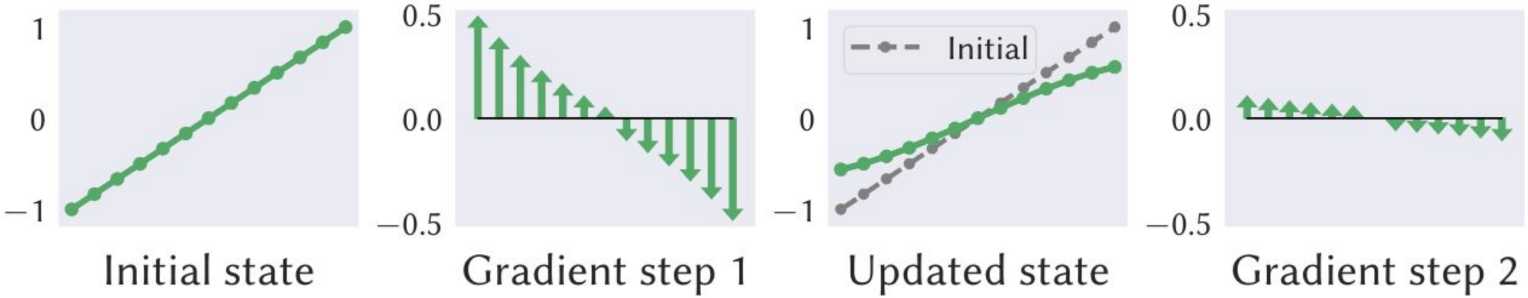
Their method

p=1,2,...

Optimisation step:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \boxed{(\mathbf{I} + \lambda \mathbf{L})^{-p}} \frac{\partial \Phi}{\partial \mathbf{x}}$$

L is a sparse operator, so can be solved efficiently



Their method - interpretation

Related to the heat diffusion equation.
Given an initial heat distribution **u**, the diffused heat distribution **x** after a time interval λ is the solution of:

$$\underset{\mathbf{x}}{\operatorname{argmin}}\ \frac{1}{2}\|\mathbf{x}-\mathbf{u}\|^2+\lambda\frac{1}{2}\operatorname{tr}\left(\mathbf{x}^\top\mathbf{L}\mathbf{x}\right)$$

It is given by (Euler-Lagrange equation):

$$\mathbf{x}=\underbrace{(\mathbf{I}+\lambda\mathbf{L})^{-1}}_{\text{diffusion operator}}\mathbf{u}$$

diffusion operator

Related to the general formula: $(\mathbf{I}-\mathbf{A})^{-1}=\sum_{k\geq 0}\mathbf{A}^k$

Their method - interpretation

Back to our minimization problem. If we decide to express **x** as the diffusion of some latent variable **u**, i.e. we want to minimize:

$$\Phi(\mathbf{x}(\mathbf{u}))$$

So the gradient descent step becomes:

$$\mathbf{u} \leftarrow \mathbf{u} - \eta \frac{\partial \mathbf{x}}{\partial \mathbf{u}} \frac{\partial \Phi}{\partial \mathbf{x}}$$

Using $\mathbf{x} = (\mathbf{I} + \lambda \mathbf{L})^{-1} \mathbf{u}$

the gradient descent step becomes:

$$\mathbf{x} \leftarrow (\mathbf{I} + \lambda \mathbf{L})^{-1} \left(\mathbf{u} - \eta \frac{\partial \mathbf{x}}{\partial \mathbf{u}} \frac{\partial \Phi}{\partial \mathbf{x}}\right) = \mathbf{x} - \eta \, (\mathbf{I} + \lambda \mathbf{L})^{-2} \frac{\partial \Phi}{\partial \mathbf{x}}$$

which is the optimization step of their method (with p=2).

Optimization scheme

Gradient descent with (first-order) momentum

$$\mathbf{g} \leftarrow (\mathbf{I} + \lambda \mathbf{L})^{-p} \frac{\partial \Phi}{\partial \mathbf{x}},$$
$$\mathbf{m}_1 \leftarrow \beta_1 \mathbf{m}_1 + (1 - \beta_1)\mathbf{g}$$
$$\mathbf{u} \leftarrow \mathbf{u} - \eta \frac{\mathbf{m}_1}{1 - \beta_1^k}$$

Adam optimizer

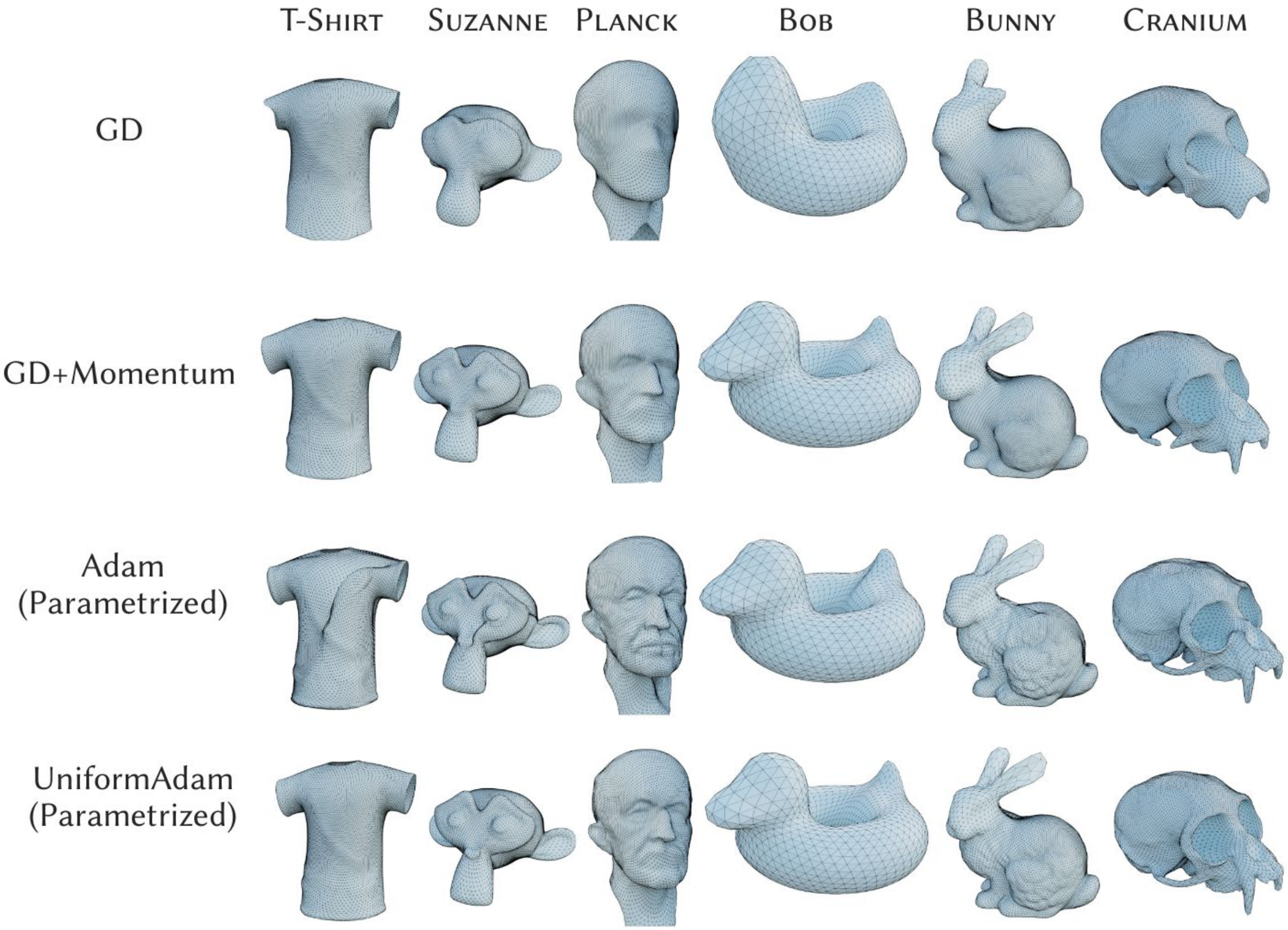
$$\mathbf{g} \leftarrow (\mathbf{I} + \lambda \mathbf{L})^{-p} \frac{\partial \Phi}{\partial \mathbf{x}},$$
$$\mathbf{m}_1 \leftarrow \beta_1 \mathbf{m}_1 + (1 - \beta_1)\mathbf{g}$$
$$\mathbf{m}_2 \leftarrow \beta_2 \mathbf{m}_2 + (1 - \beta_2)\mathbf{g}^2$$
$$\mathbf{u} \leftarrow \mathbf{u} - \eta \left(\frac{\mathbf{m}_1}{1 - \beta_1^k} \right) \Big/ \left(\sqrt{\frac{\mathbf{m}_2}{1 - \beta_2^k}} + \varepsilon \right)$$

UniformAdam optimizer (their)

$$\mathbf{g} \leftarrow (\mathbf{I} + \lambda \mathbf{L})^{-p} \frac{\partial \Phi}{\partial \mathbf{x}},$$
$$\mathbf{m}_1 \leftarrow \beta_1 \mathbf{m}_1 + (1 - \beta_1)\mathbf{g}$$
$$\mathbf{m}_2 \leftarrow \beta_2 \mathbf{m}_2 + (1 - \beta_2)\mathbf{g}^2$$
$$\mathbf{u} \leftarrow \mathbf{u} - \frac{\eta}{(1 - \beta_1^k)\sqrt{\frac{\|\mathbf{m}_2\|_\infty}{1-\beta_2^k}}} \mathbf{m}_1$$

default values for Adam's betas in PyTorch/dr.JIT: (0.9, 0.999)

Optimization scheme



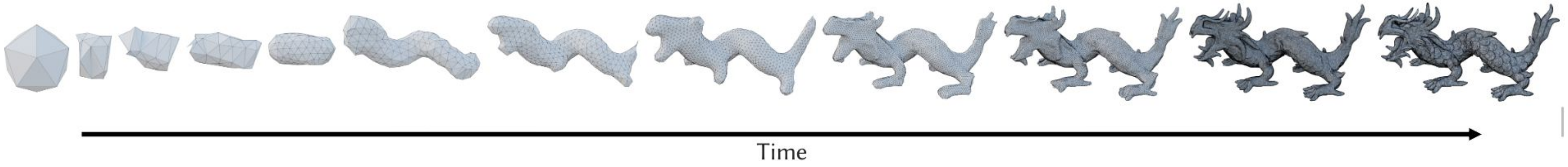
Optimization scheme

$$
\begin{aligned}
\mathbf{g} &\leftarrow (\mathbf{I} + \lambda \mathbf{L})^{-p} \frac{\partial \Phi}{\partial \mathbf{x}}, \\
\mathbf{m}_1 &\leftarrow \beta_1 \mathbf{m}_1 + (1 - \beta_1)\mathbf{g} \\
\mathbf{m}_2 &\leftarrow \beta_2\, \mathbf{m}_2 + (1 - \beta_2)\, \mathbf{g}^2 \\
\mathbf{u} &\leftarrow \mathbf{u} - \frac{\eta}{(1 - \beta_1^k)\sqrt{\frac{\|\mathbf{m}_2\|_\infty}{1-\beta_2^k}}}\, \mathbf{m}_1
\end{aligned}
$$

Computed with a **Cholesky factorization**

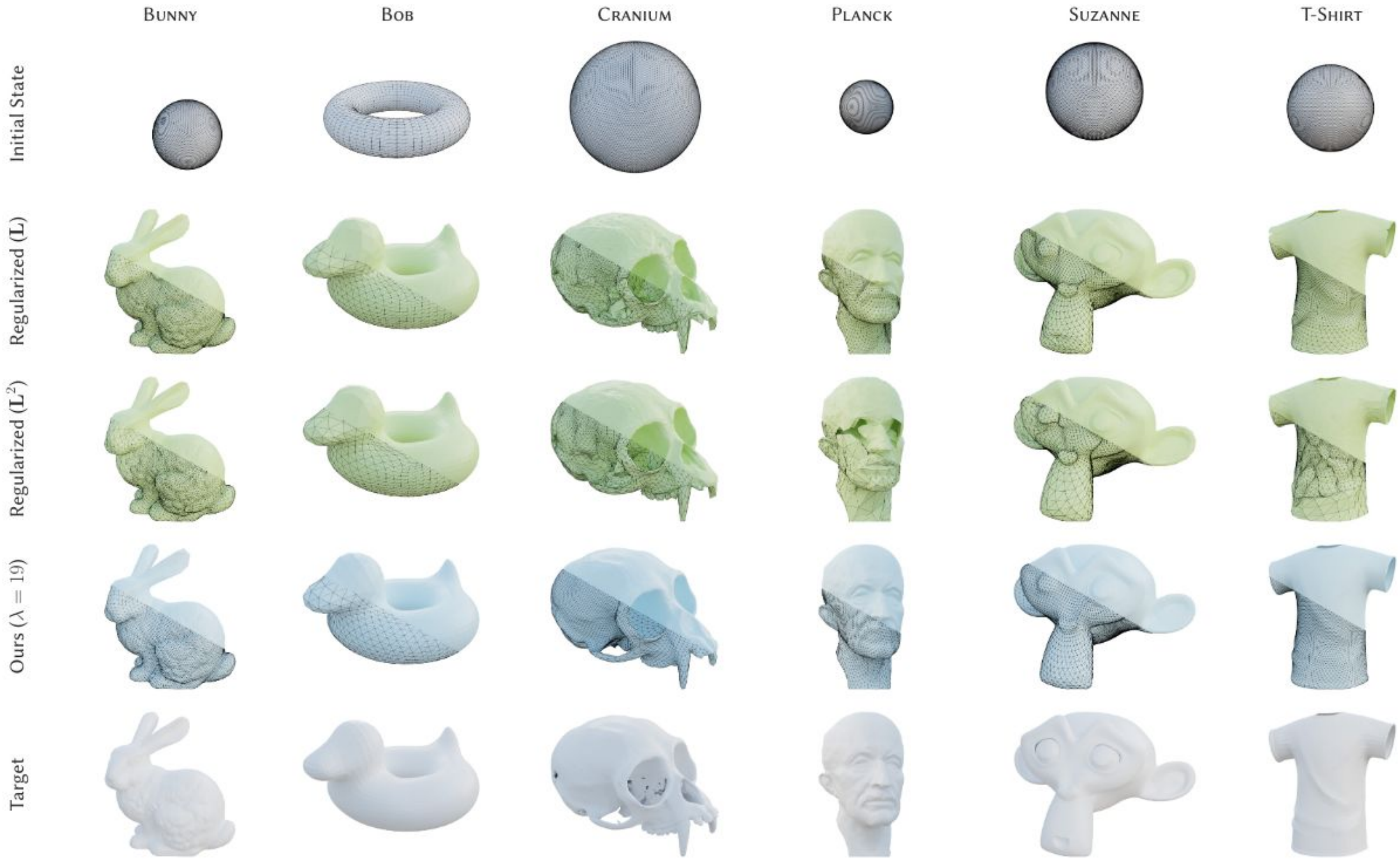
The combinatorial Laplacian is used: it **only depends on the connectivity**, not on the vertex positions. So the factorization can be **re-used across iterations**.

They also implement a remeshing strategy: at (manually-specified) iterations, they refine their mesh, using isotropic remeshing.

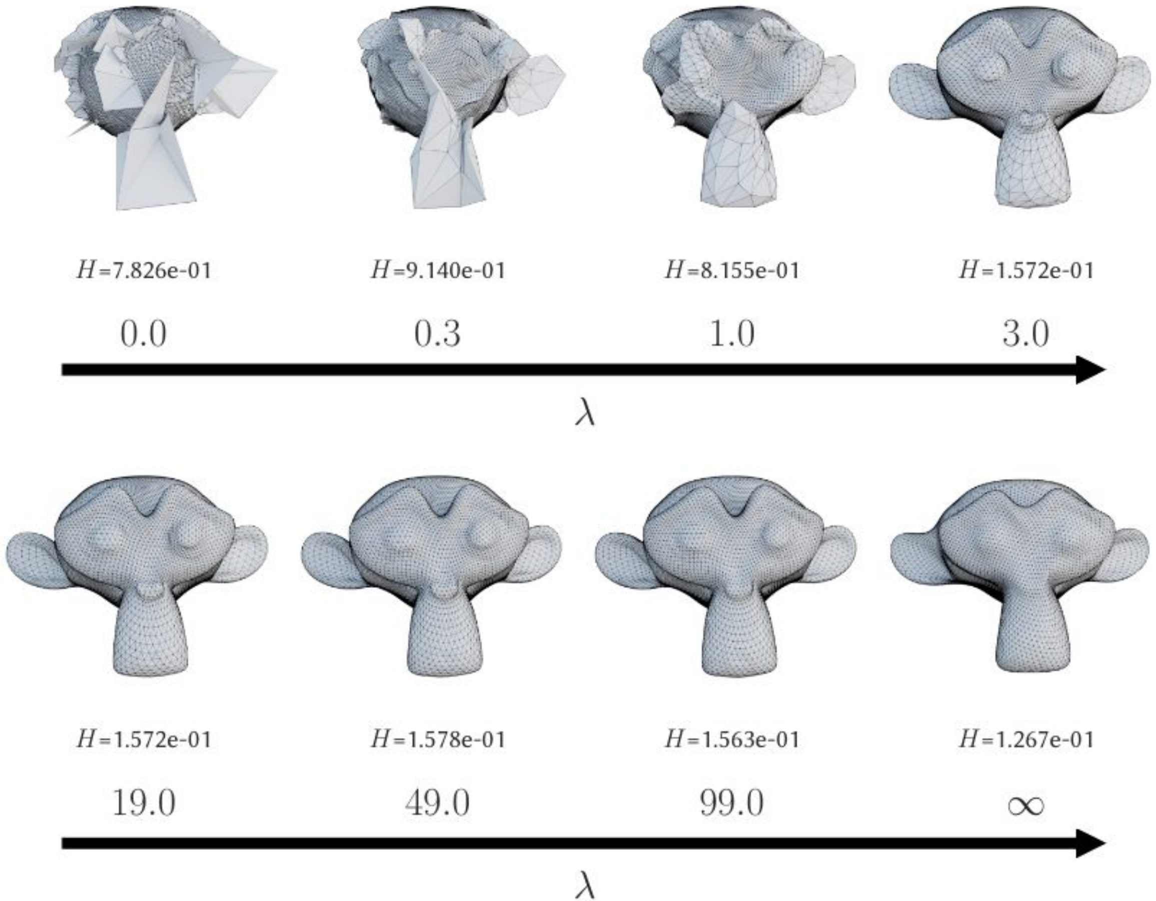


Video

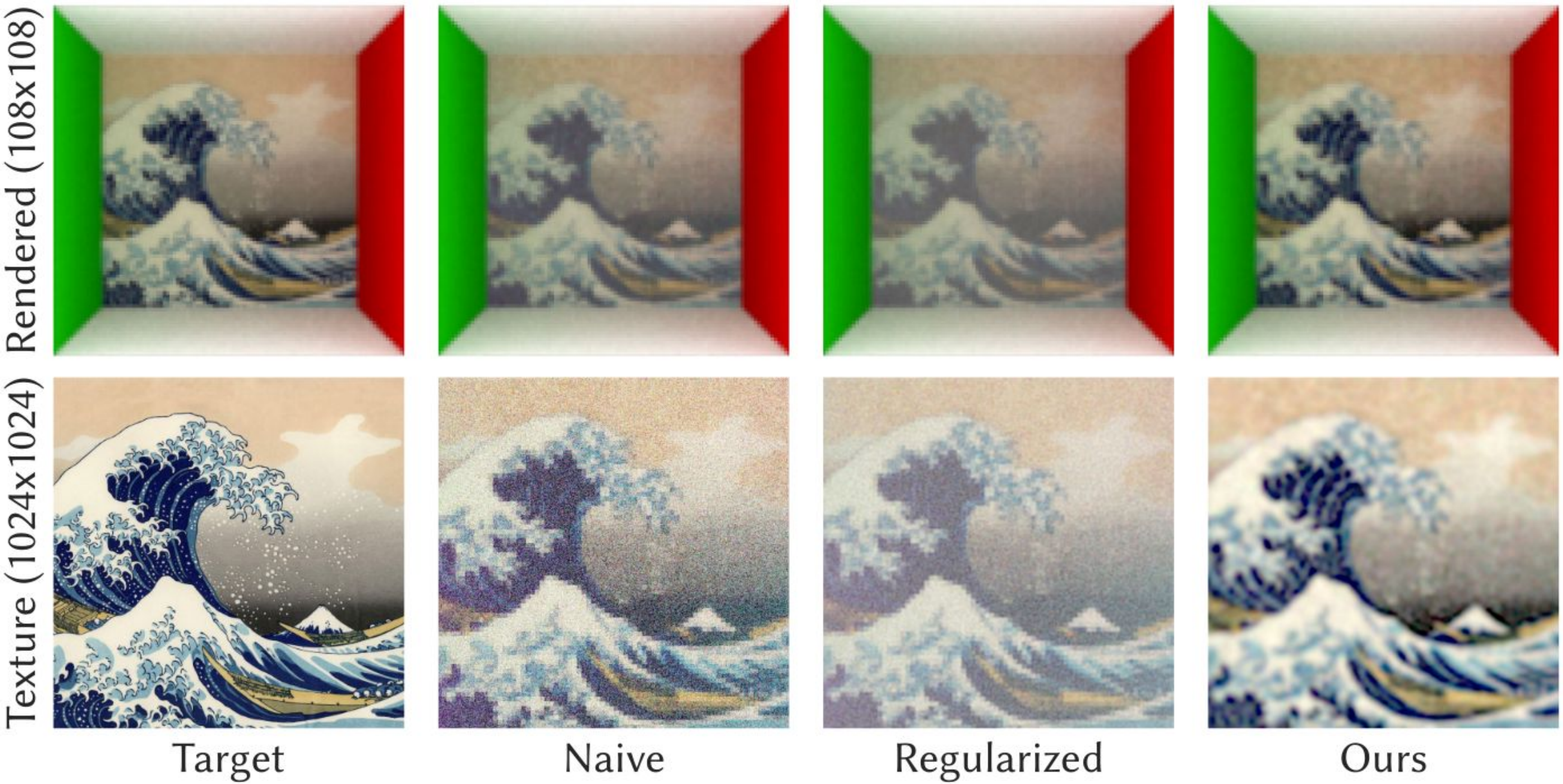
Some pictures



Some pictures



Some pictures



Merci !