

a primer on geometry optimization

CIX Summer School 2023 | U Michigan

Carnegie
Mellon
University



Prof. Alexandra Ion
 interactive structures lab



hi, I am alex.

Assistant Professor
at CMU HCII

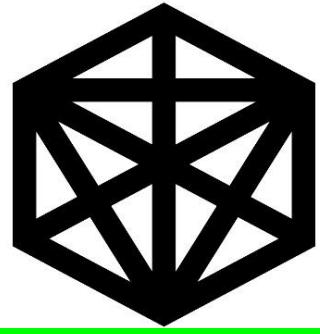


Postdoc, 2020:
ETH Zurich
Olga Sorkine-Hornung, Graphics



PhD, 2019:
Hasso Plattner Institute
Patrick Baudisch, HCI





interactive structures lab

digital fabrication



interactive structures

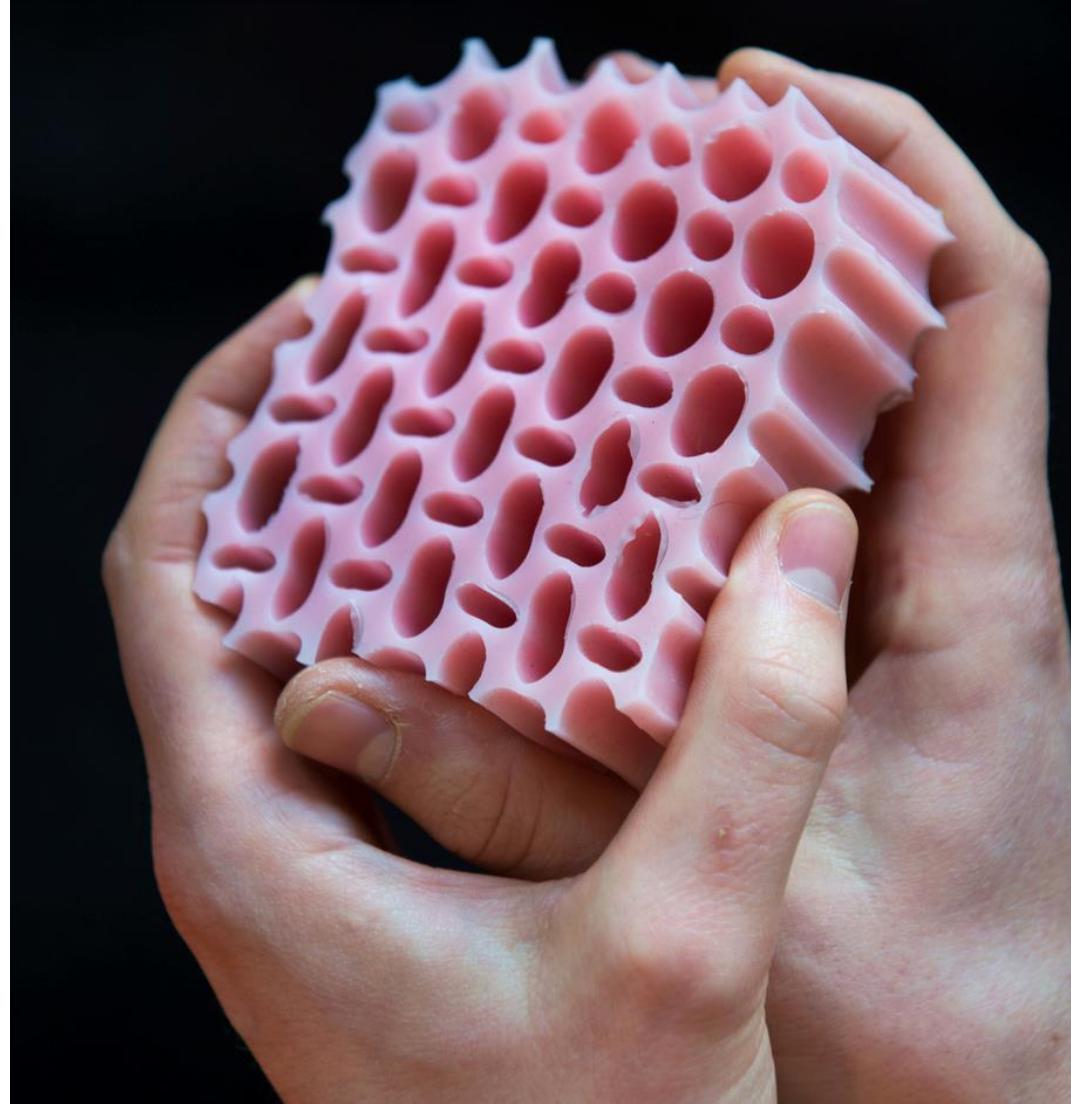


image: formlabs.com

digital fabrication



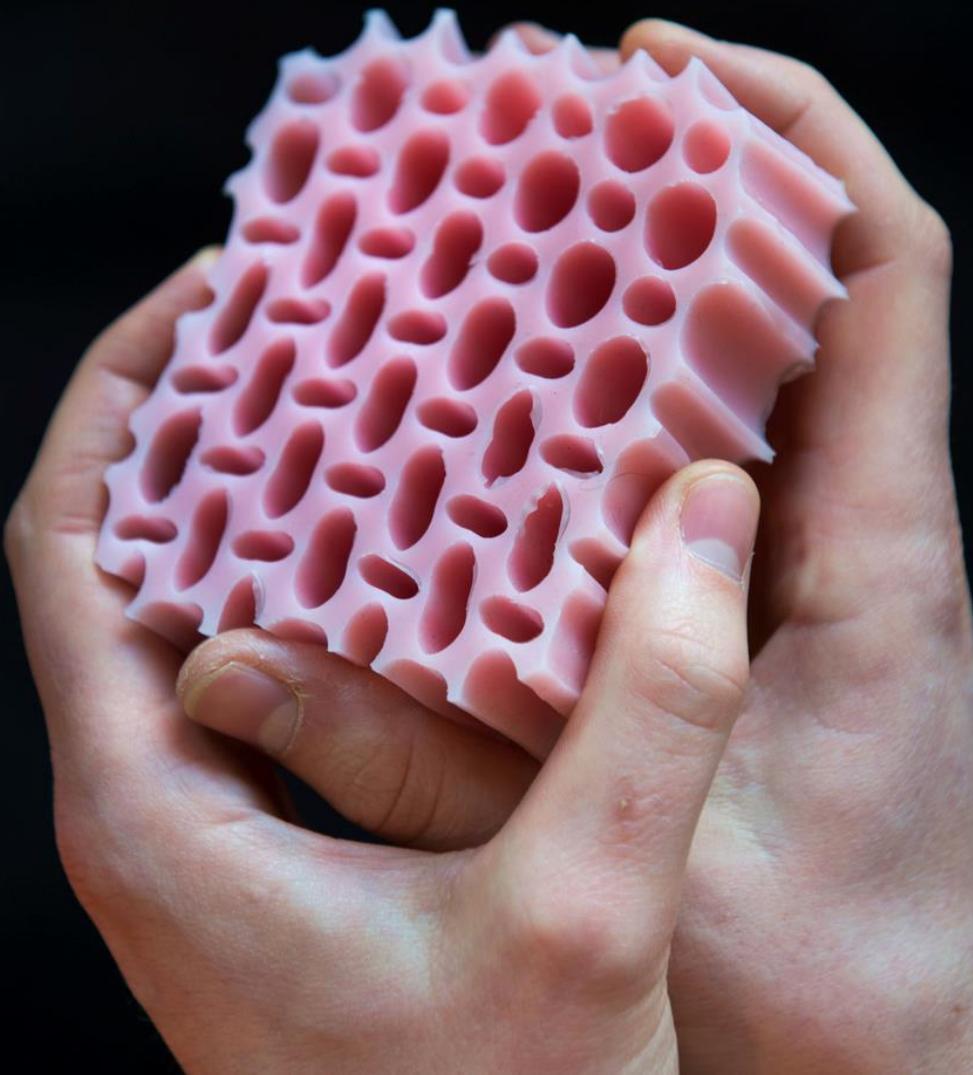
interactive structures

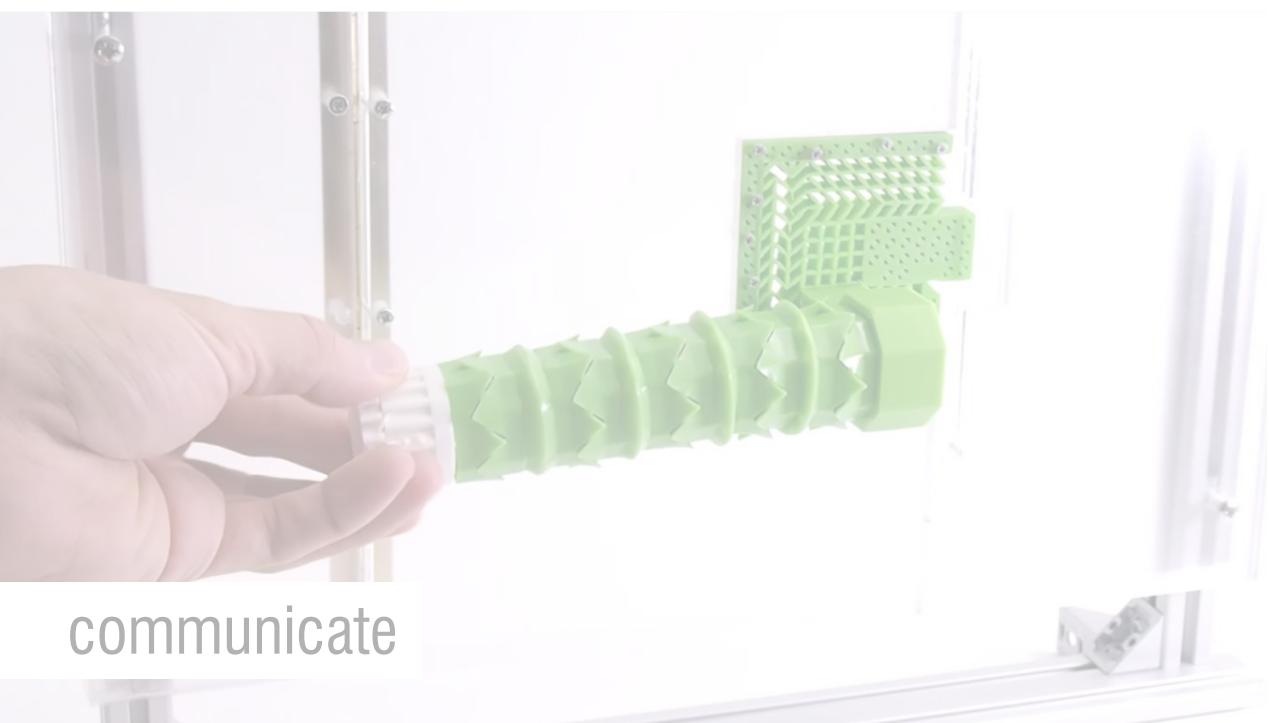
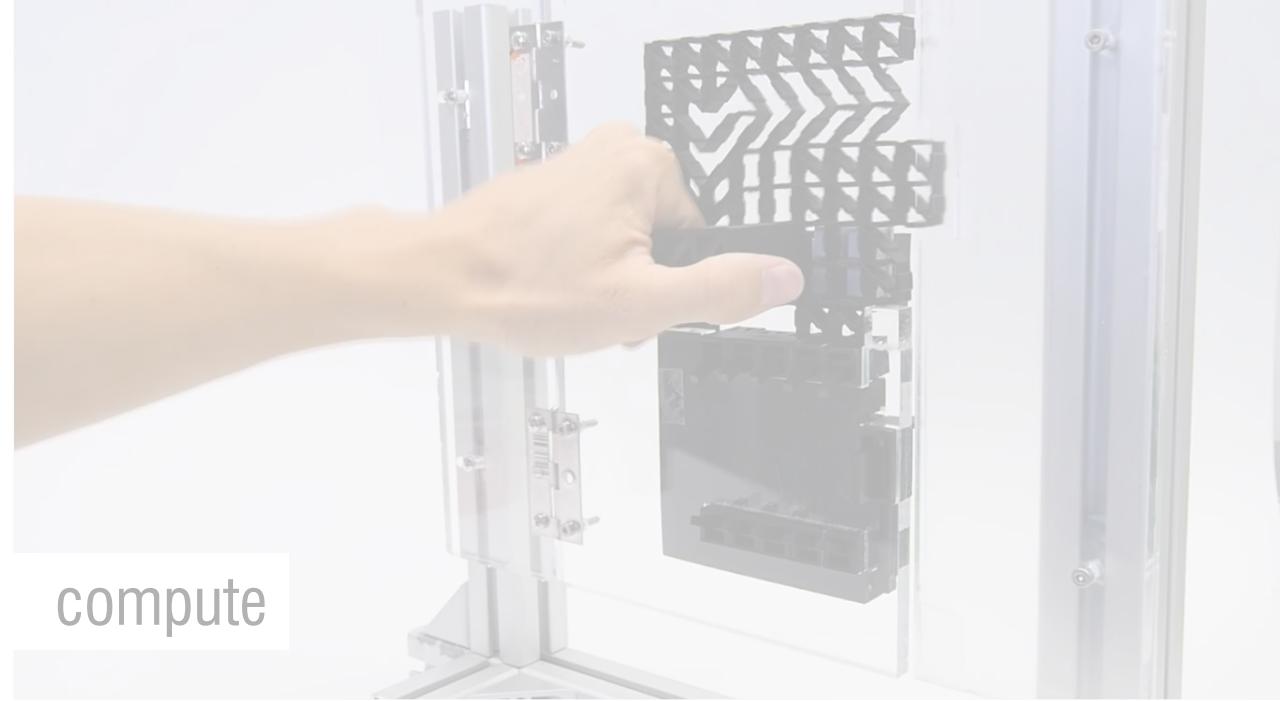
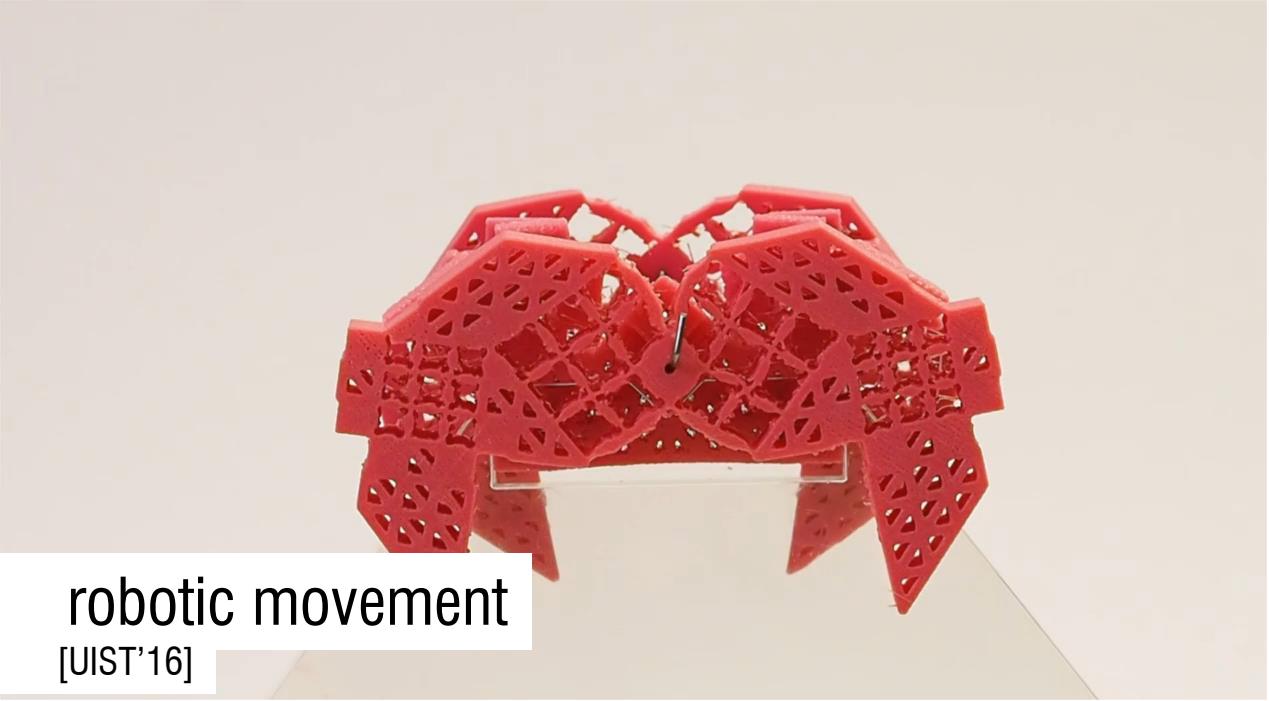
structures that **embed functionality**

such that they can react to **simple input**
with **complex behavior**.



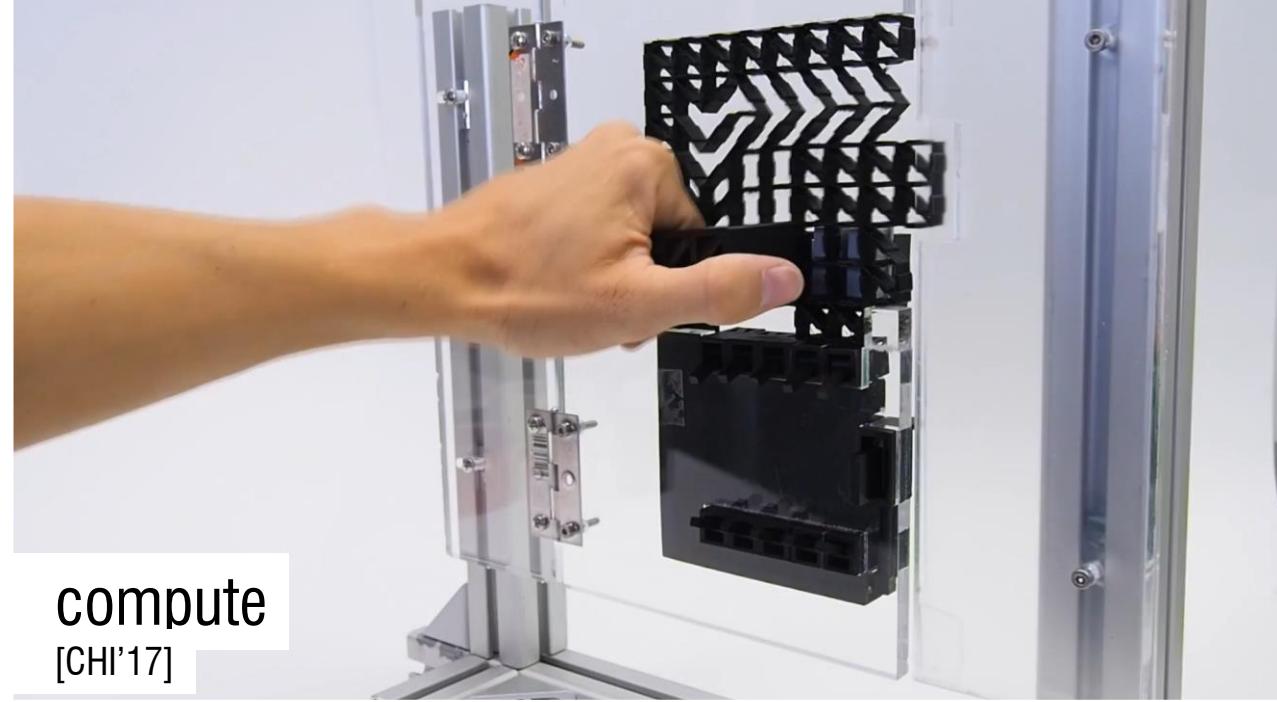
interactive structures



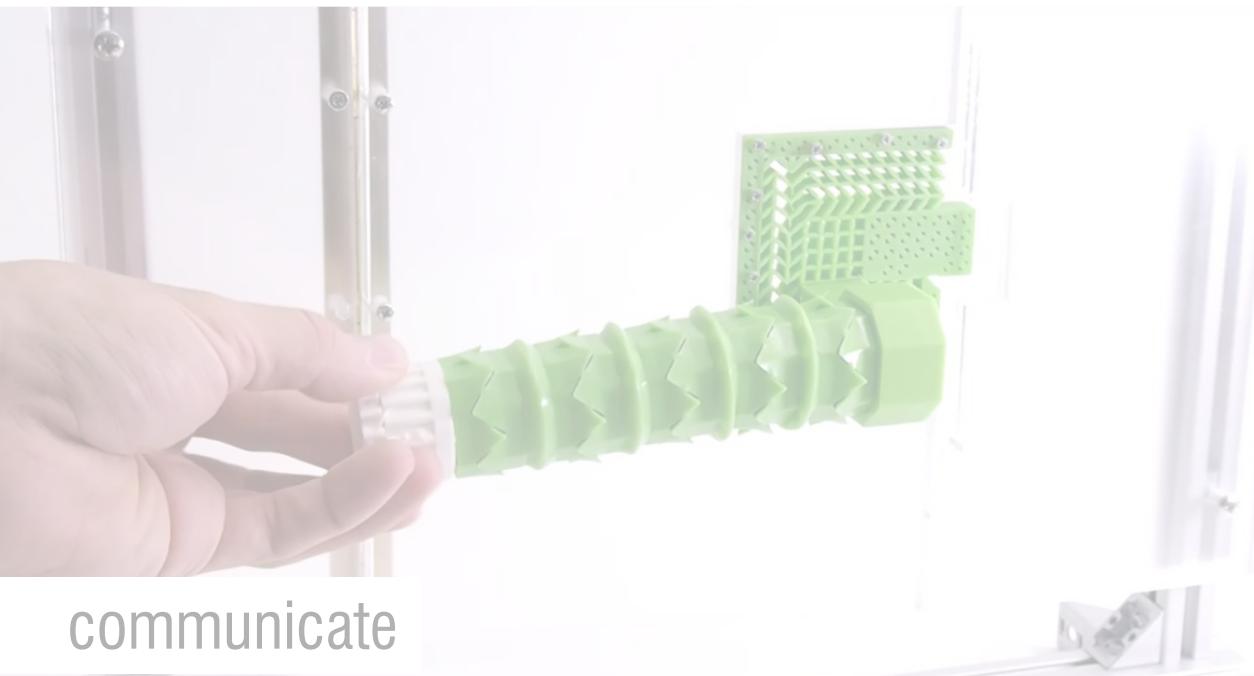




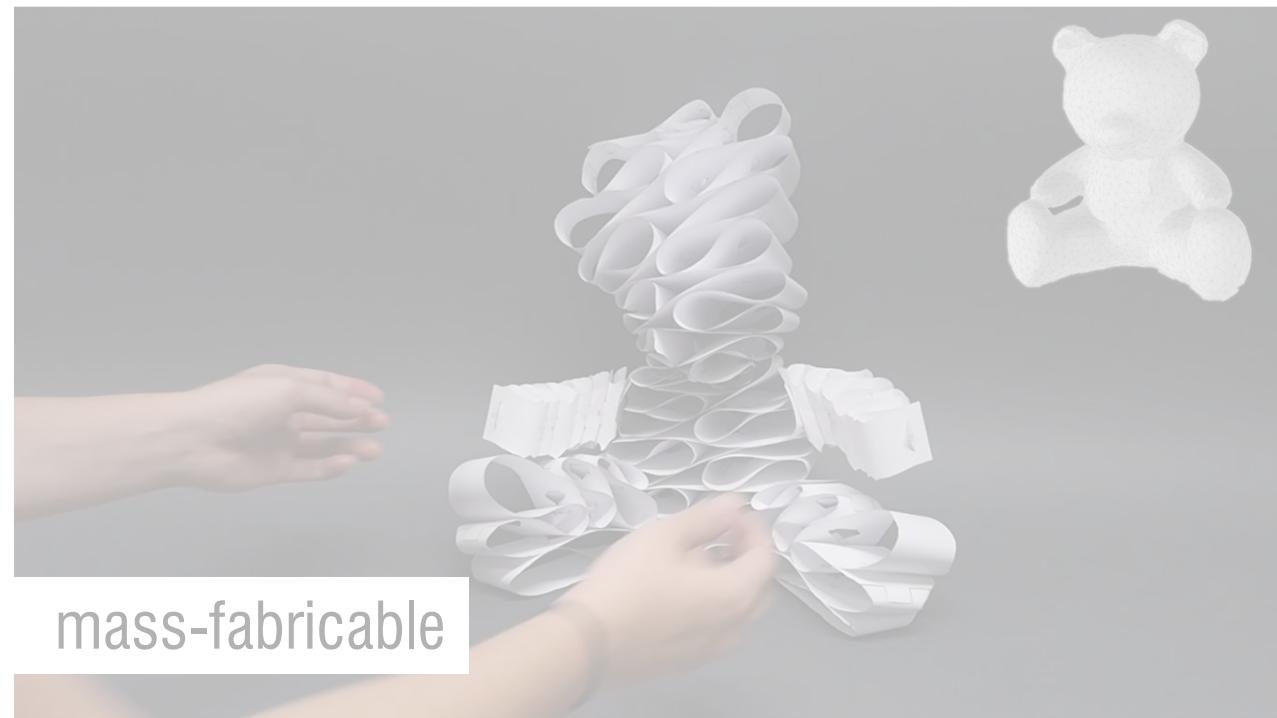
robotic movement



compute
[CHI'17]



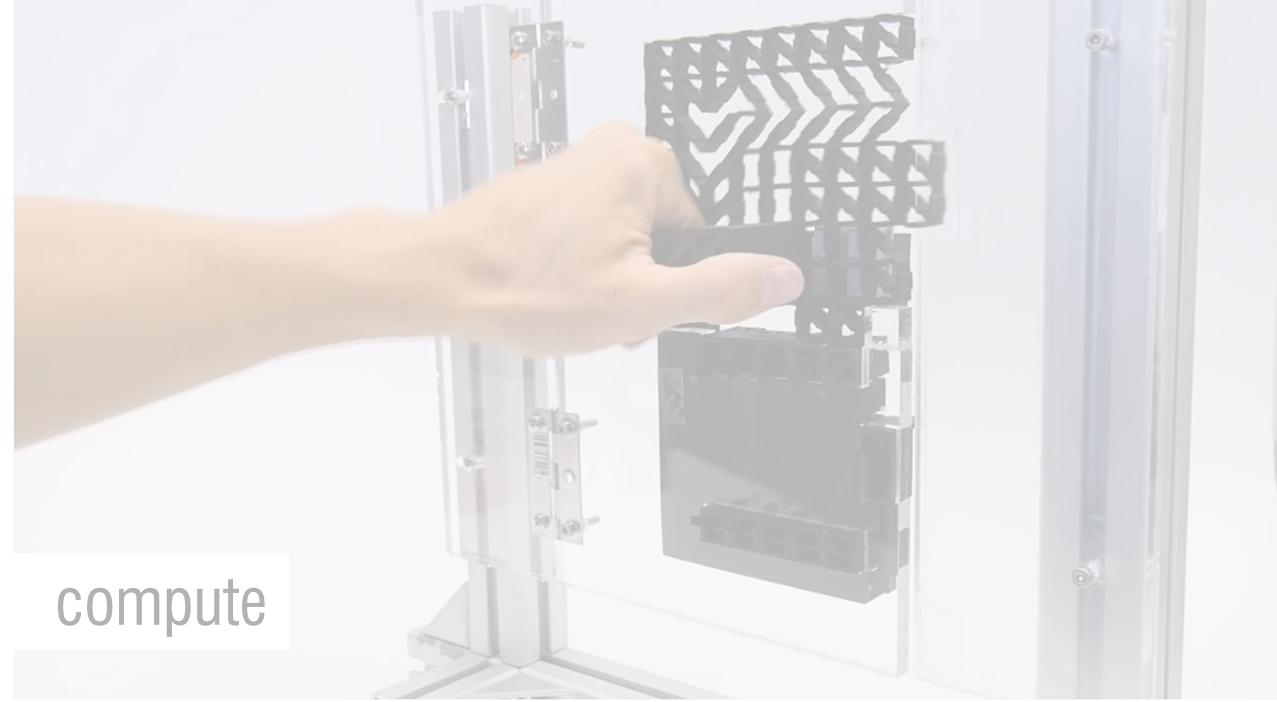
communicate



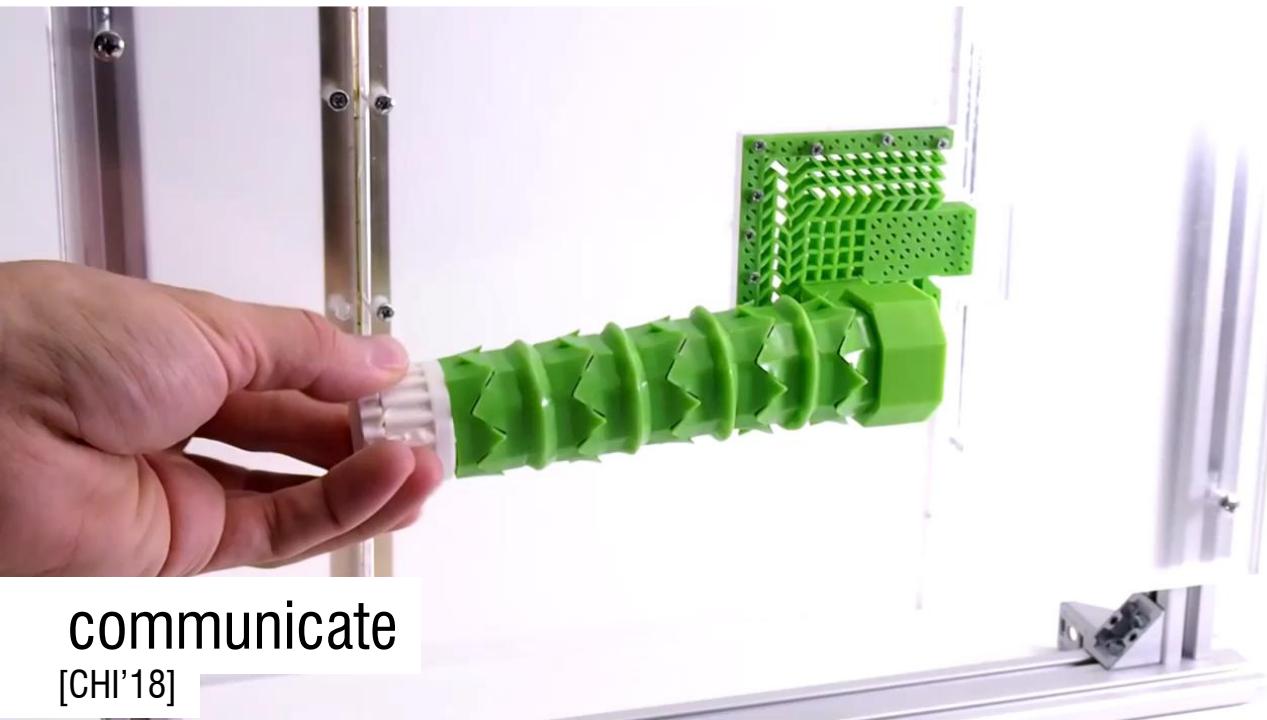
mass-fabricable



robotic movement



compute



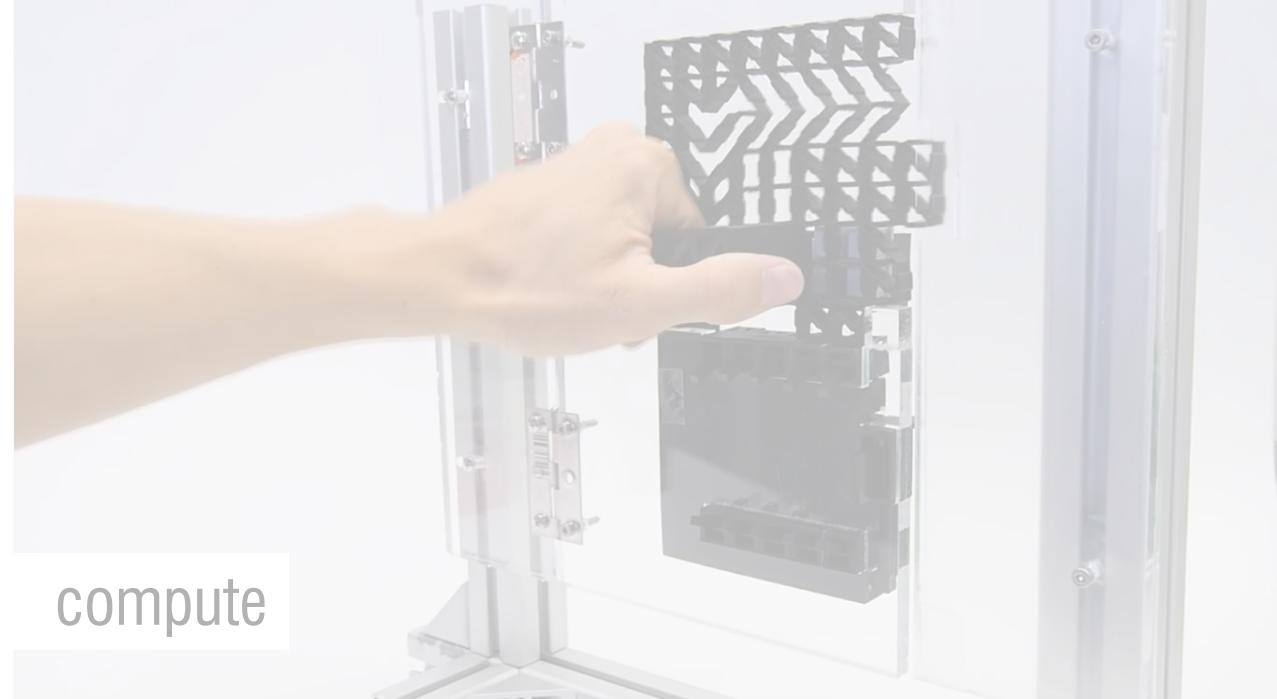
communicate
[CHI'18]



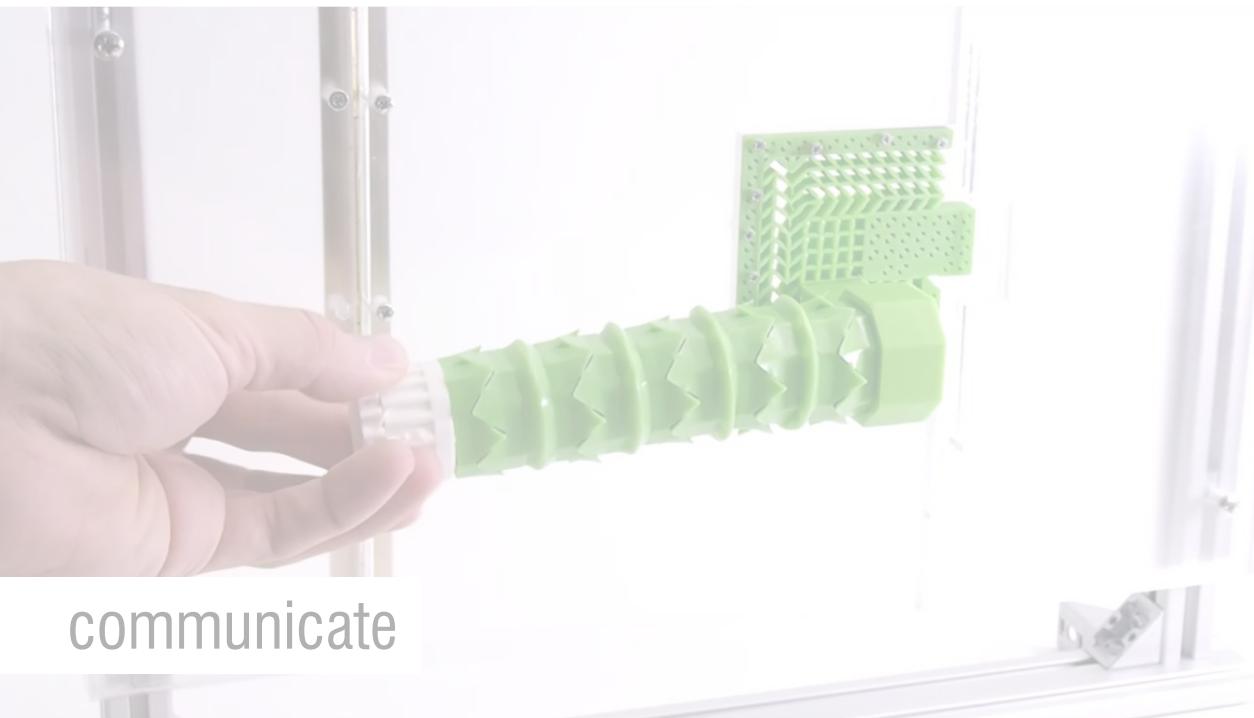
mass-fabricable



robotic movement



compute



communicate



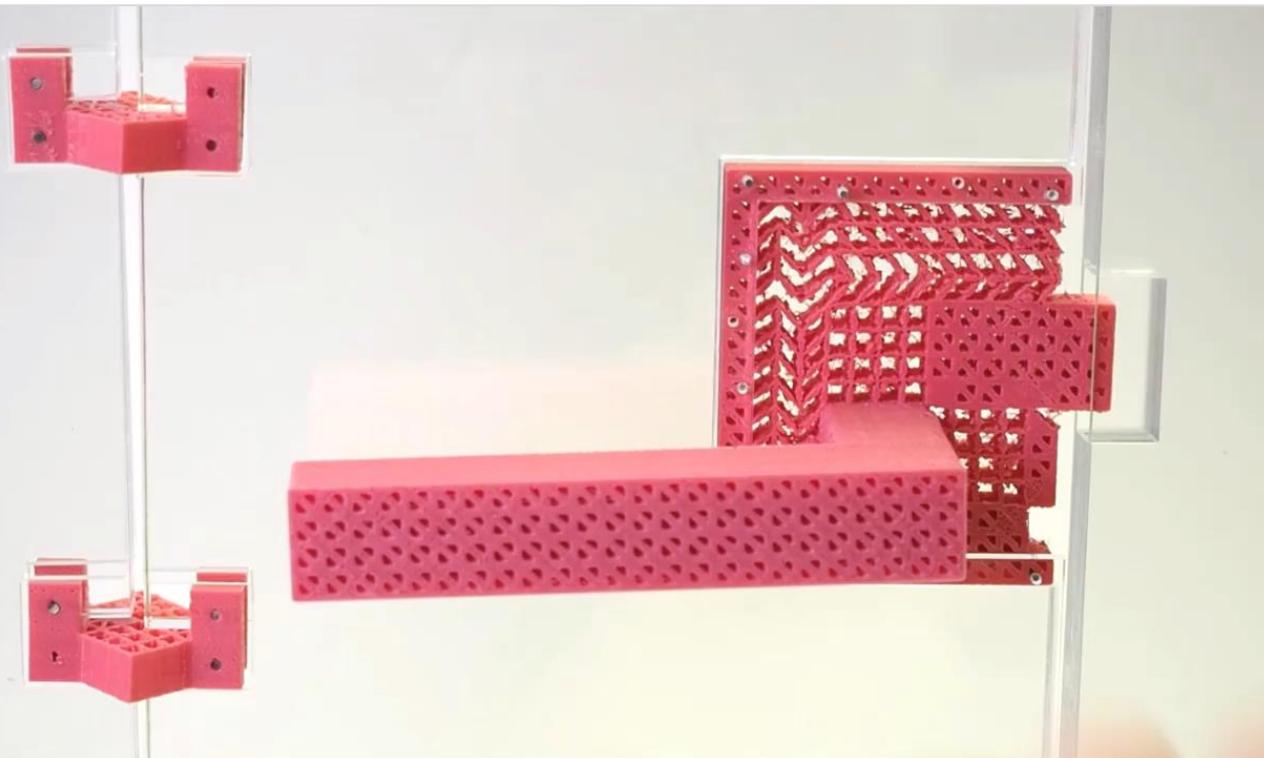
mass-fabricable
[CHI'21]

HCI + geometry + engineering + design

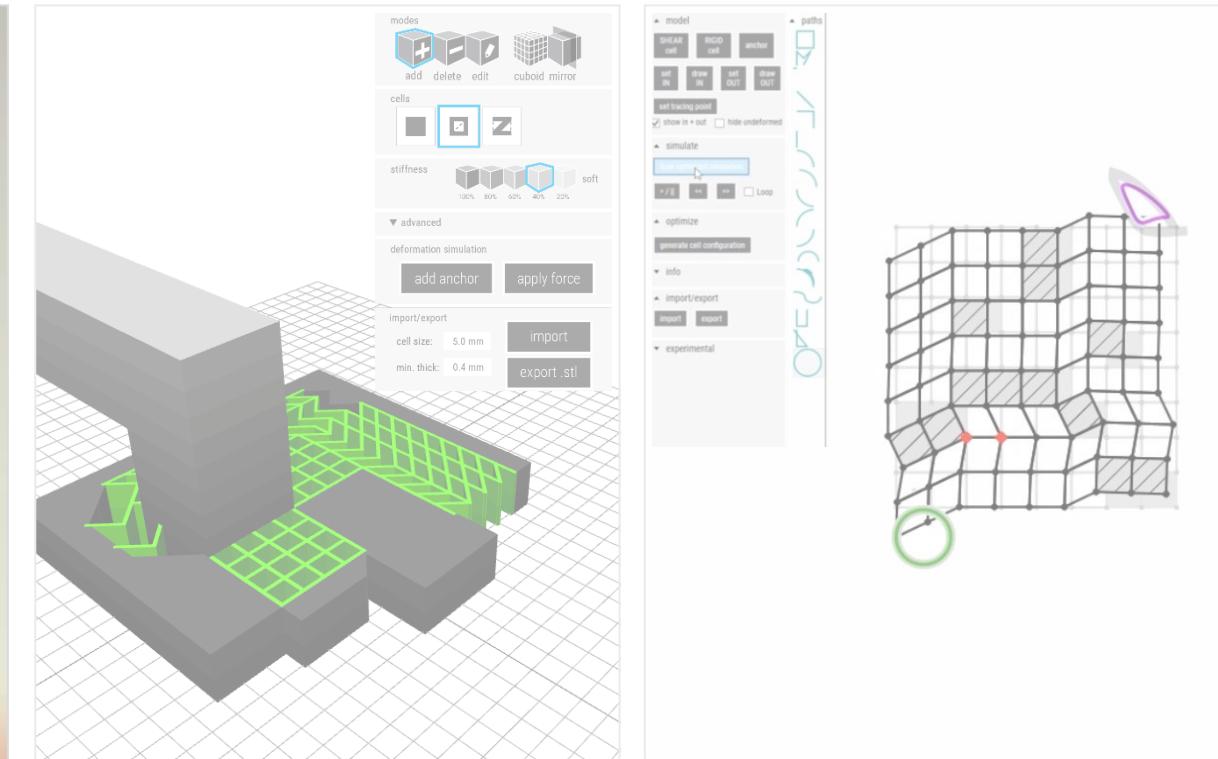
the work we do in the lab

example: Metamaterial Mechanisms

[UIST'16, CHI'19]



structures



editor

inverse design



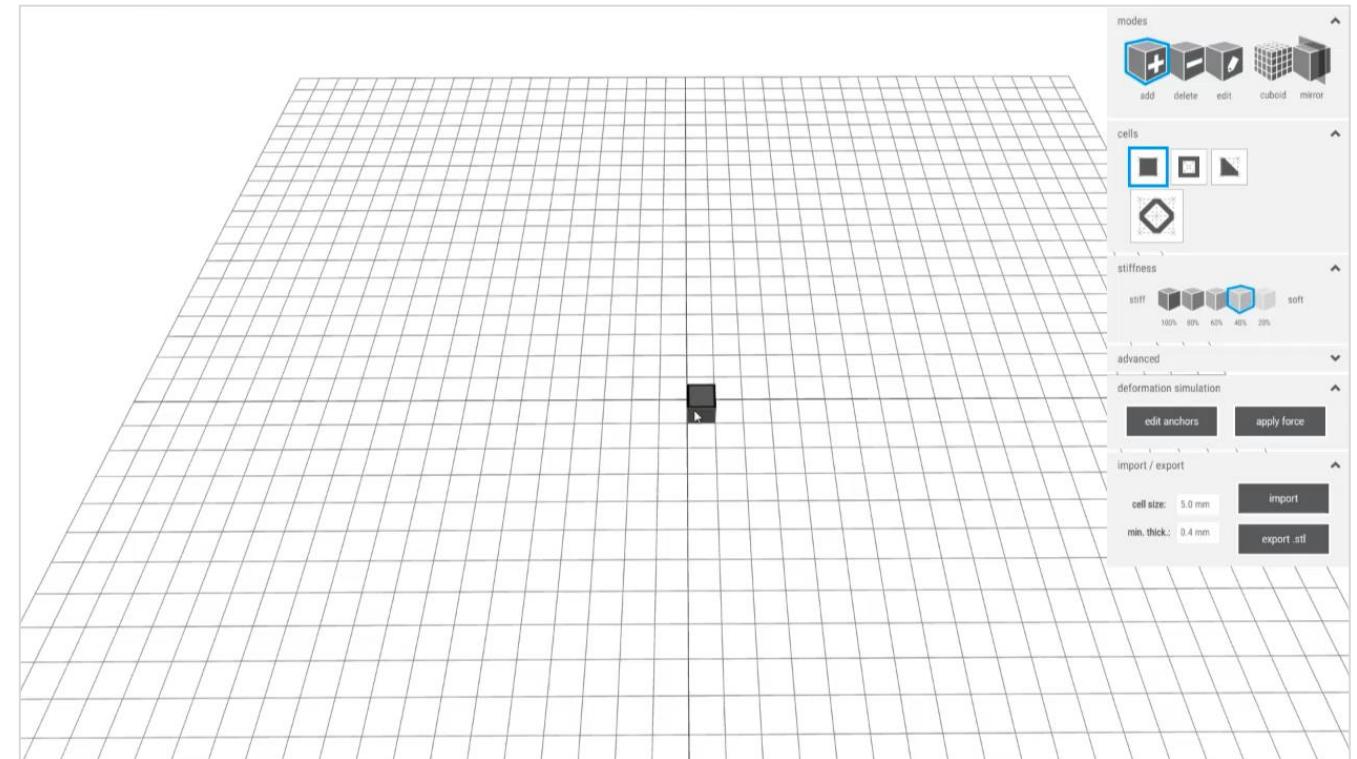
the work we do in the lab

example: Metamaterial Mechanisms

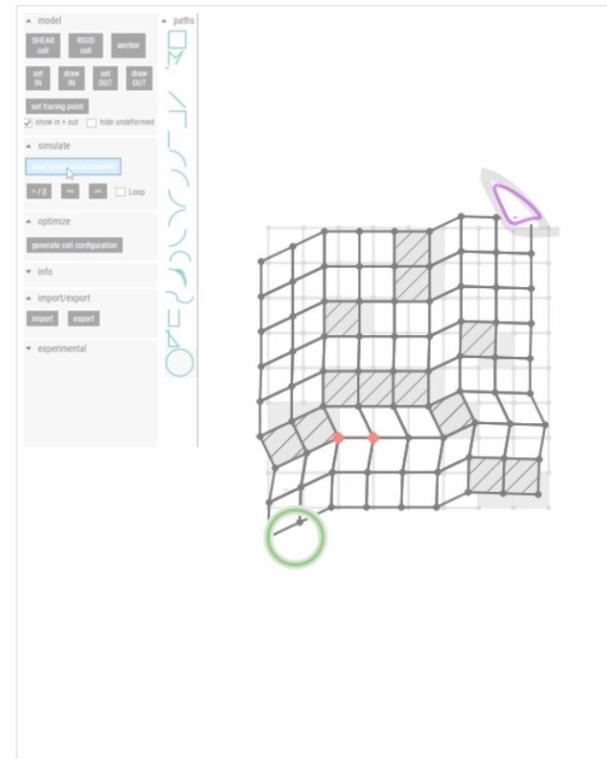
[UIST'16, CHI'19]



structures



editor



inverse design



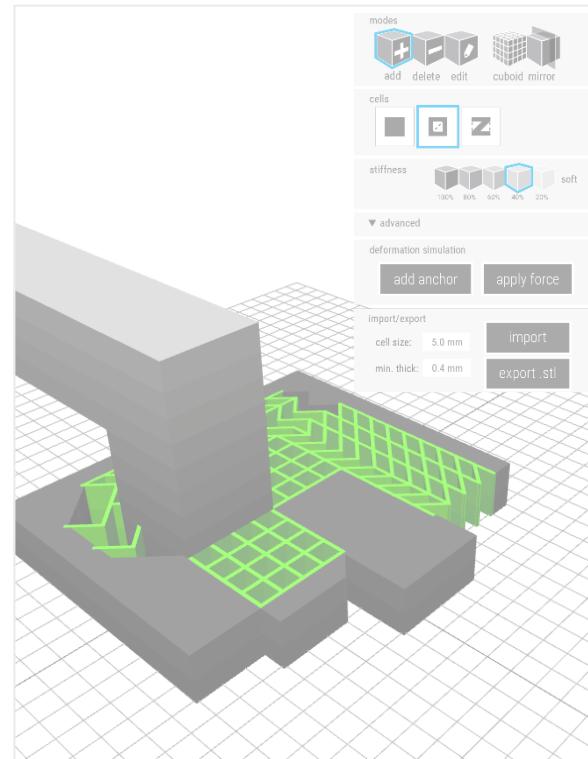
the work we do in the lab

example: Metamaterial Mechanisms

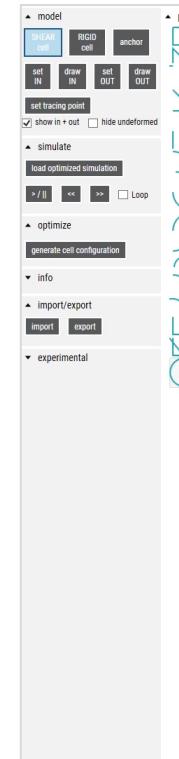
[UIST'16, CHI'19]



structures



editor

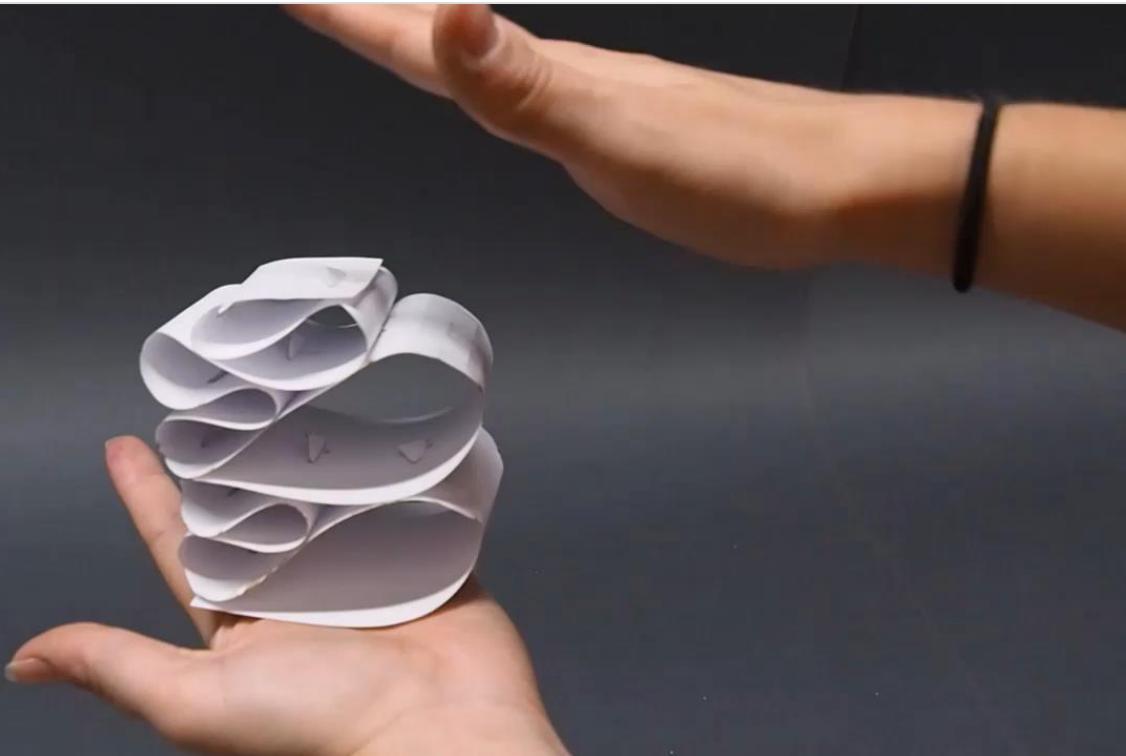


inverse design

combinatorial optimization, meta-heuristics

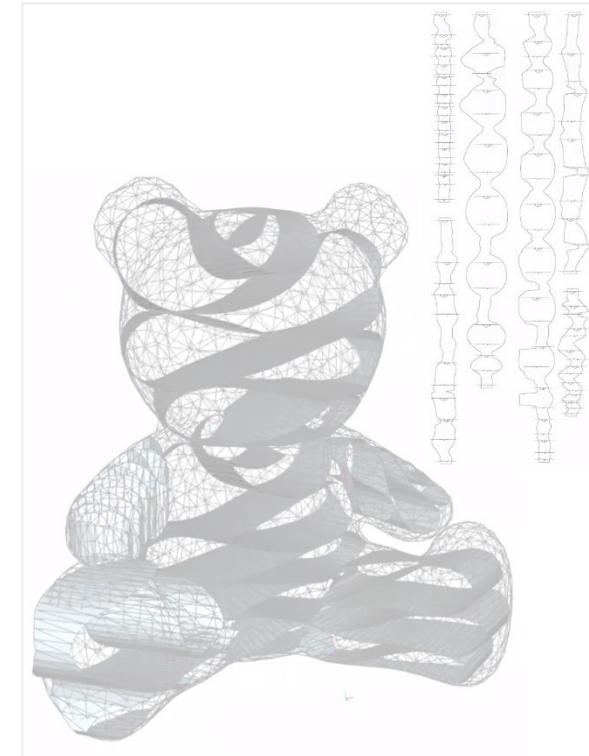
the work we do in the lab

example: Developable Metamaterials [CHI'21]



structures

developable, mass-manufacturable, easy to assemble



inverse design

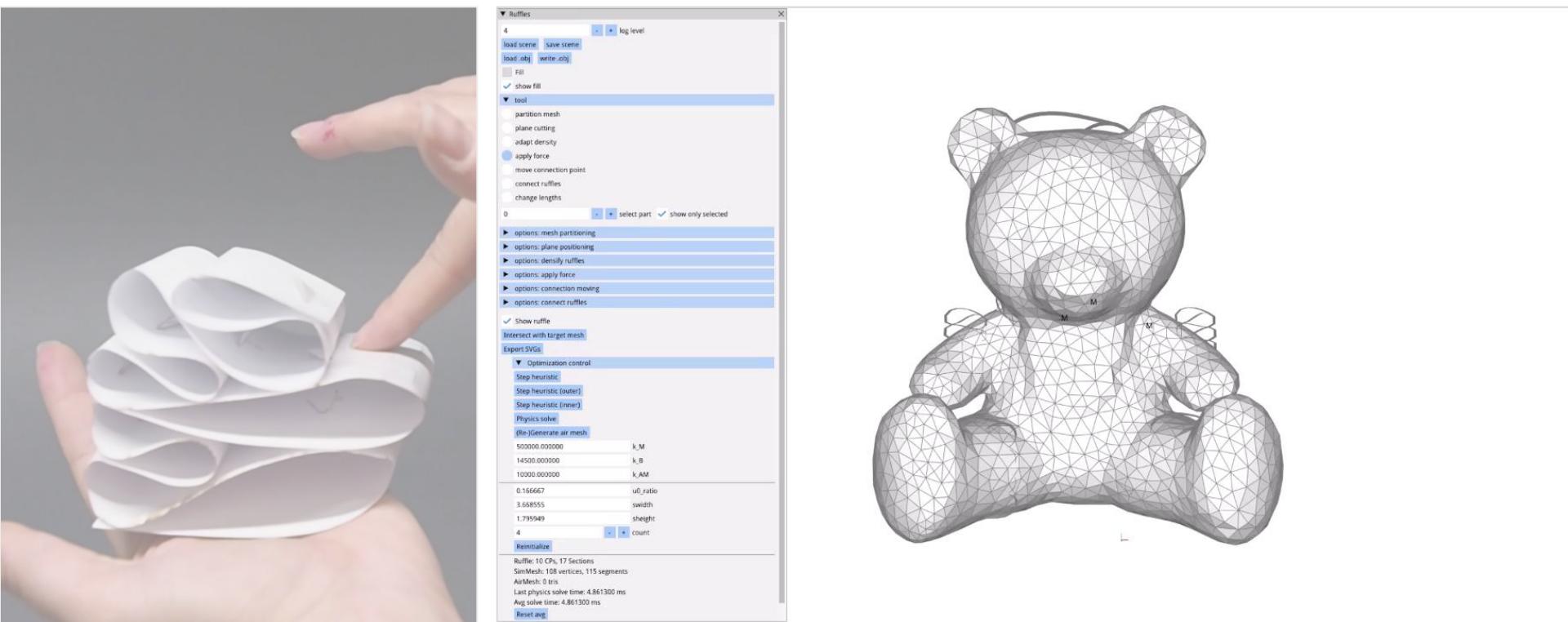


impactful applications



the work we do in the lab

example: Developable
Metamaterials [CHI'21]



structures

inverse design

thin shells, physics simulation, shape optimization



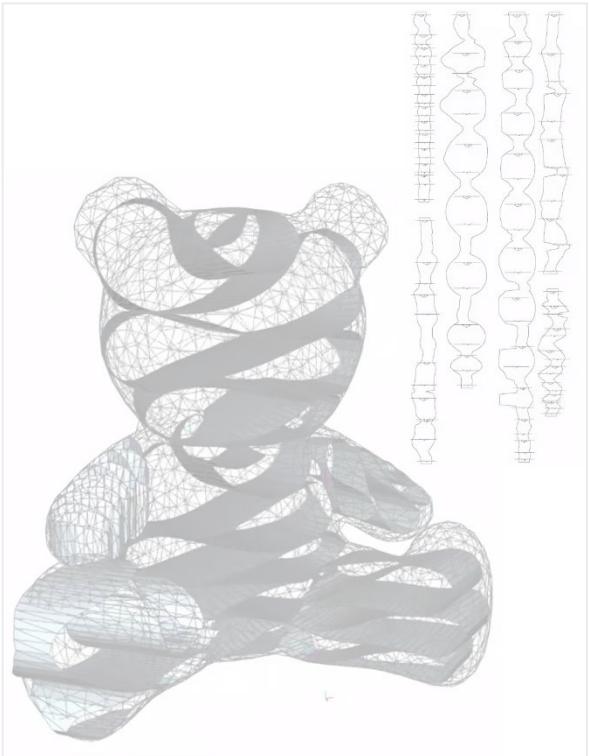
impactful applications

the work we do in the lab

example: Developable
Metamaterials [CHI'21]



structures



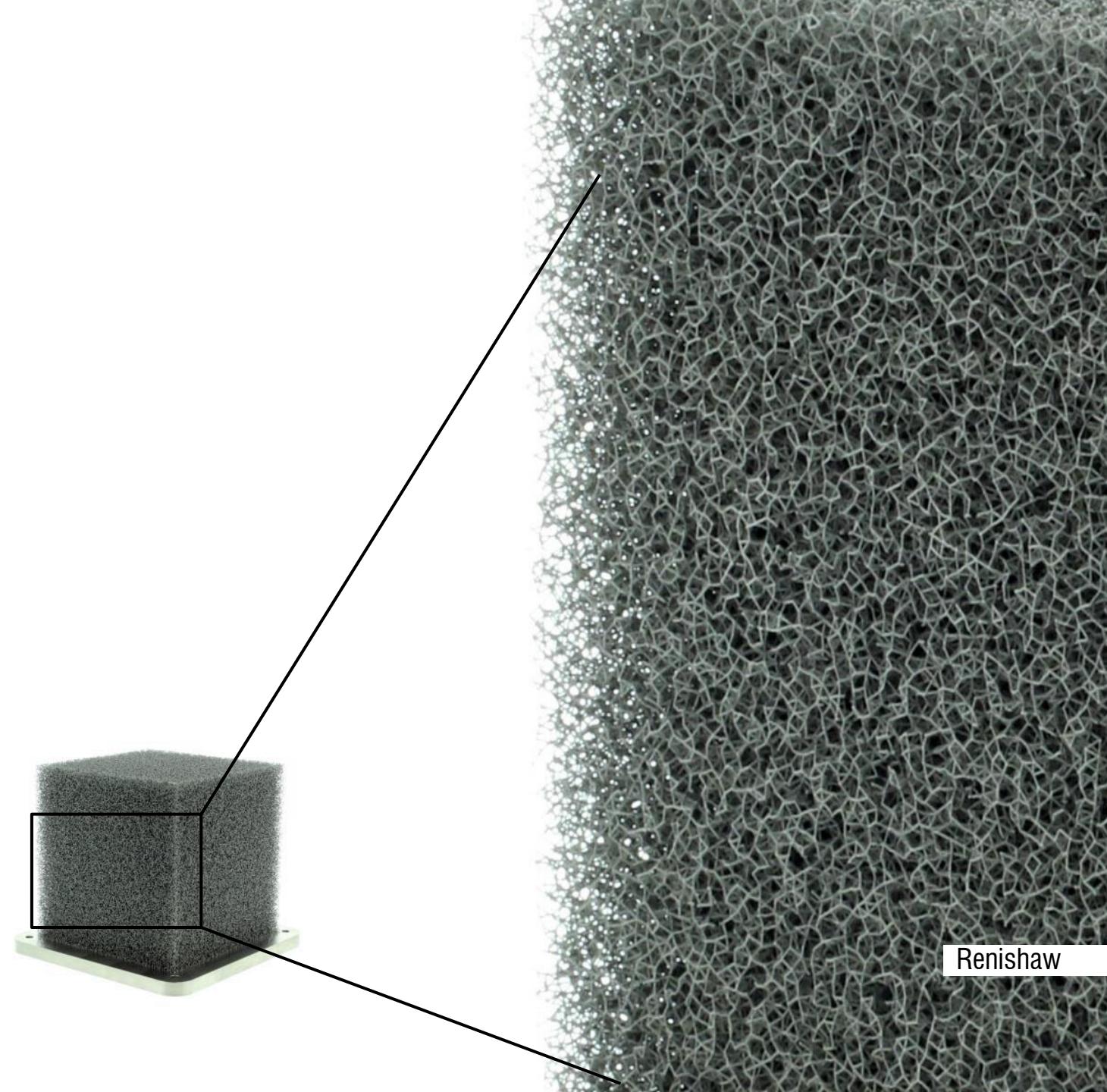
inverse design



impactful applications

customizable, affordable, sensorized and actuated prostheses

Geometric complexity

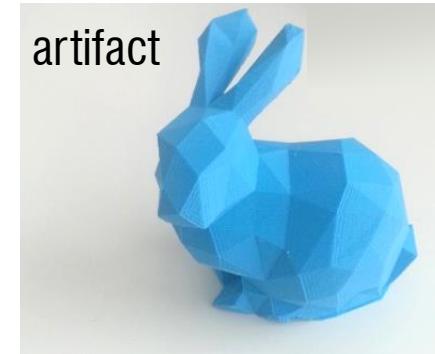




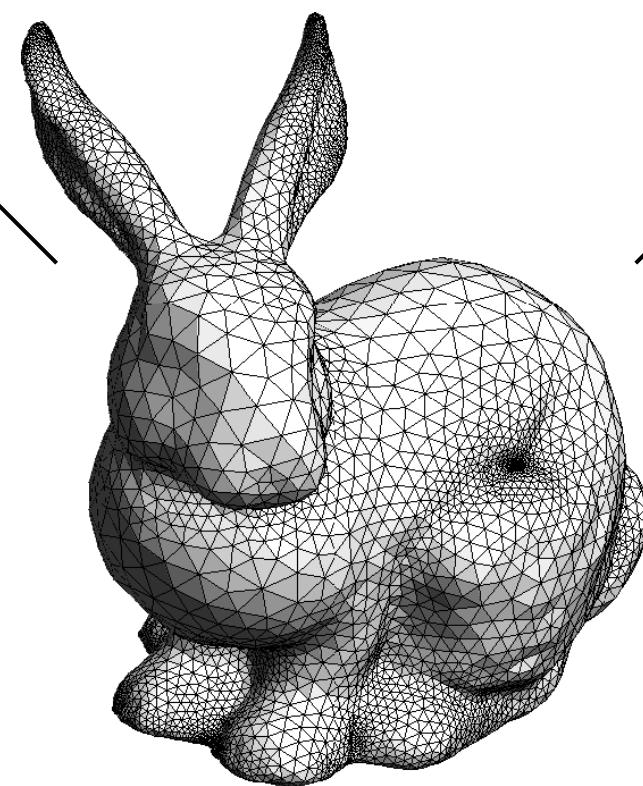
2D output



3D output

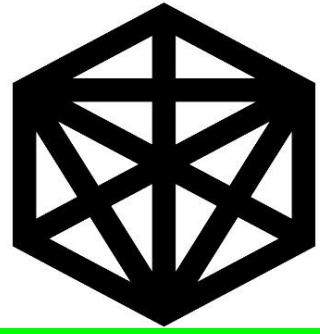


artifact



geometry

geometry



interactive structures lab

basics of geometry processing

1.1 Introduction to libigl

1.2 DDG curves

1.3 Smoothing meshes

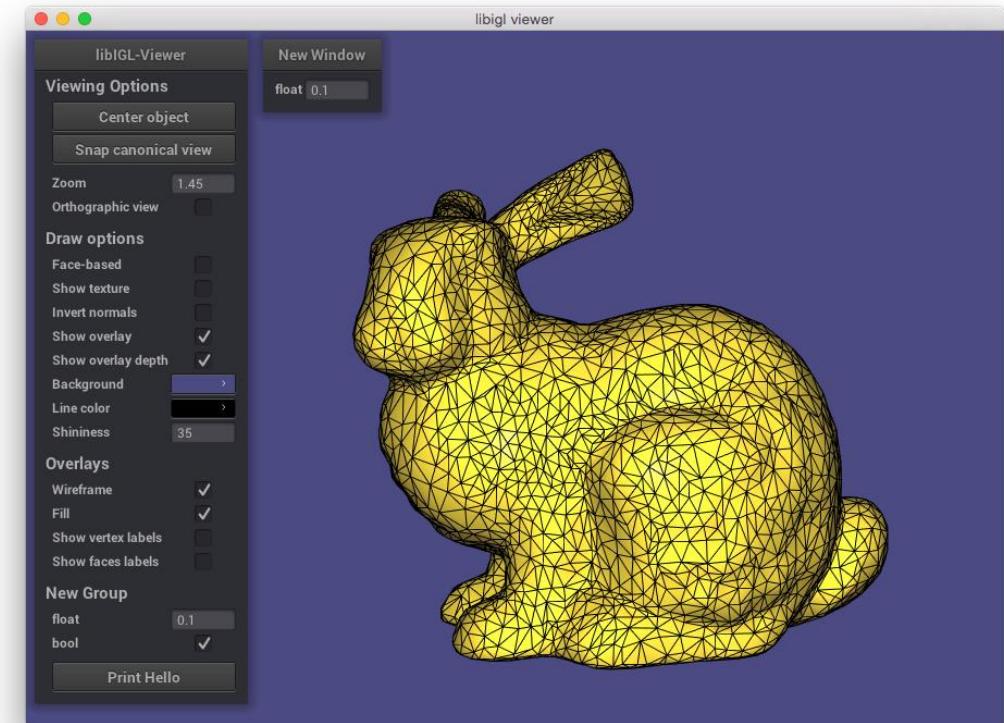
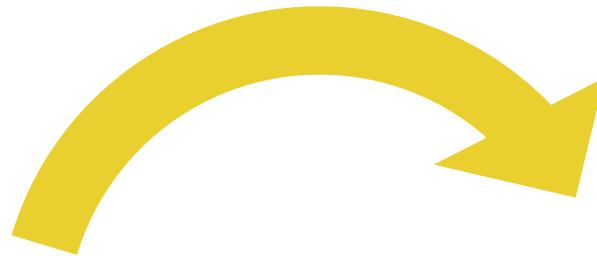
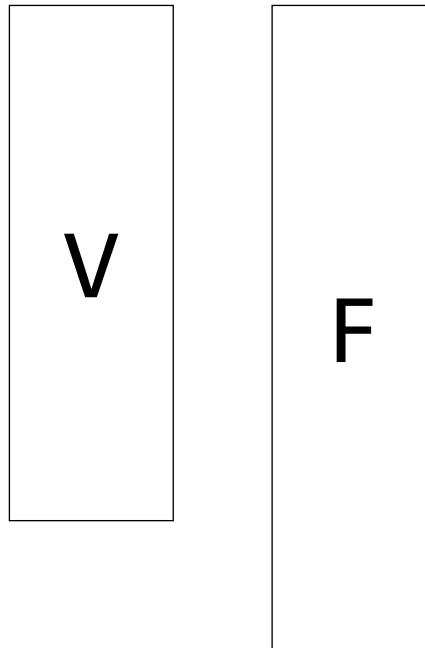
1.1 Introduction to libigl

1.2 DDG curves

1.3 Smoothing meshes

11big

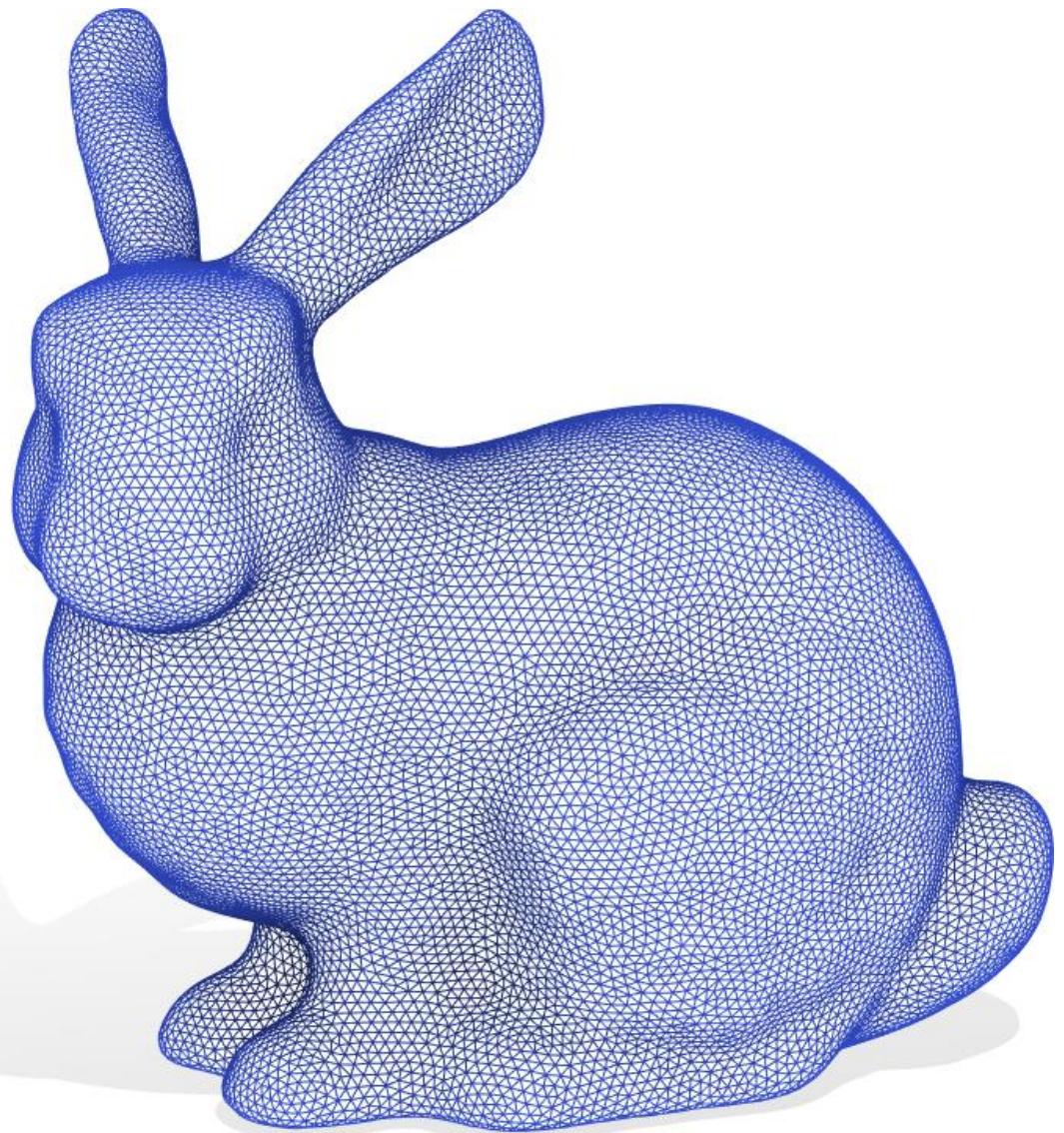
libigl data structures & style



Jacobson & Panozzo, "[libigl course](#)"

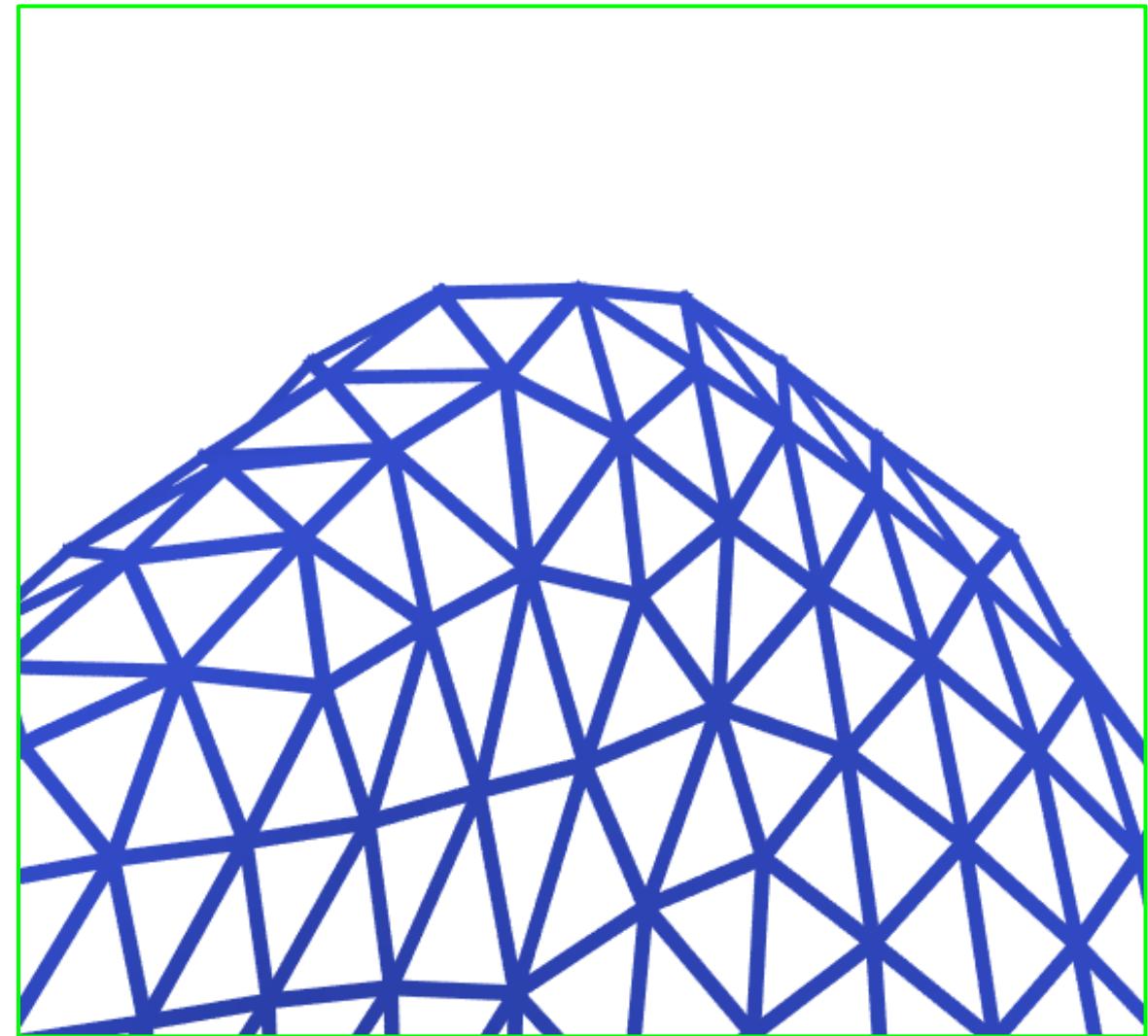
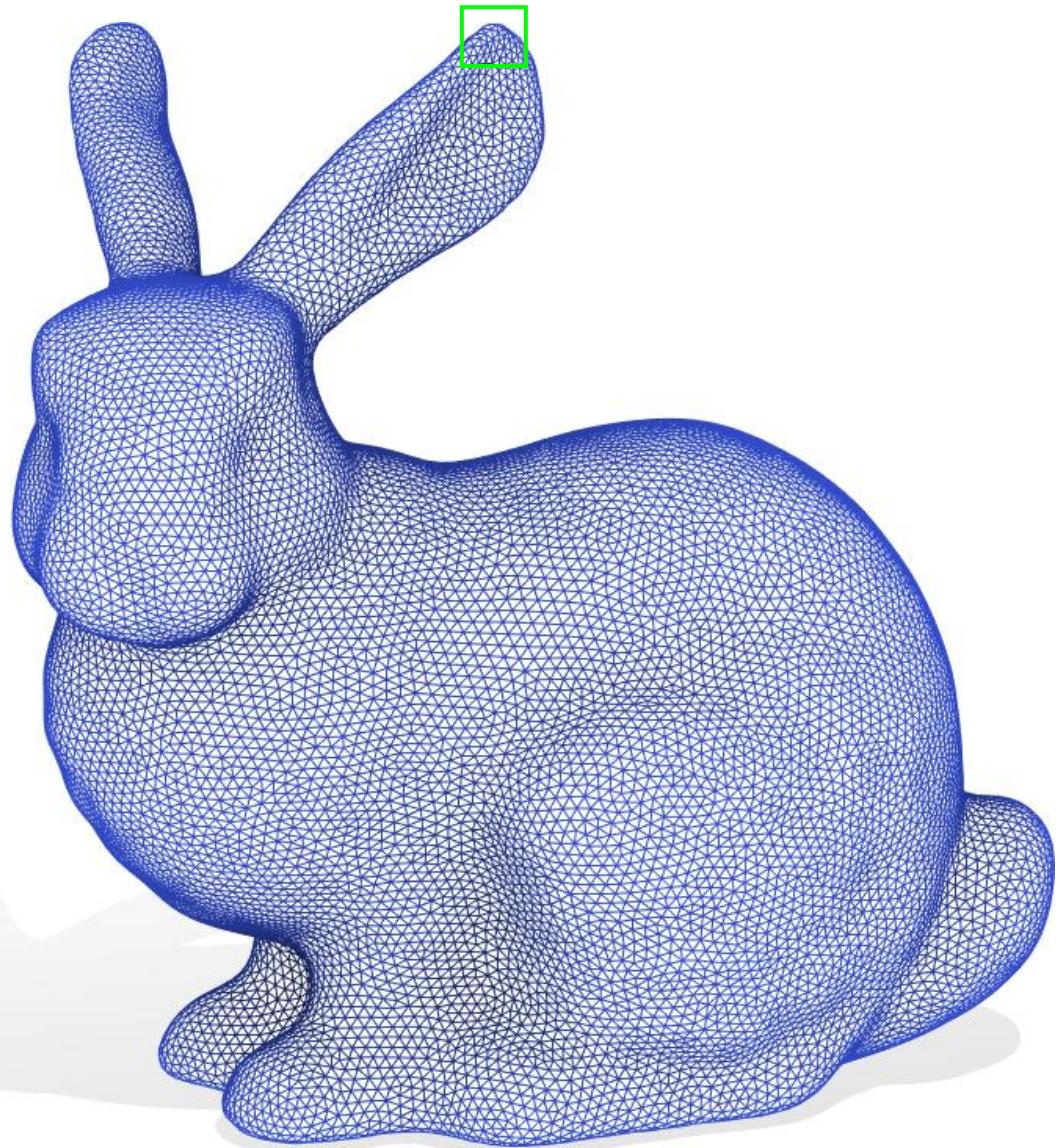


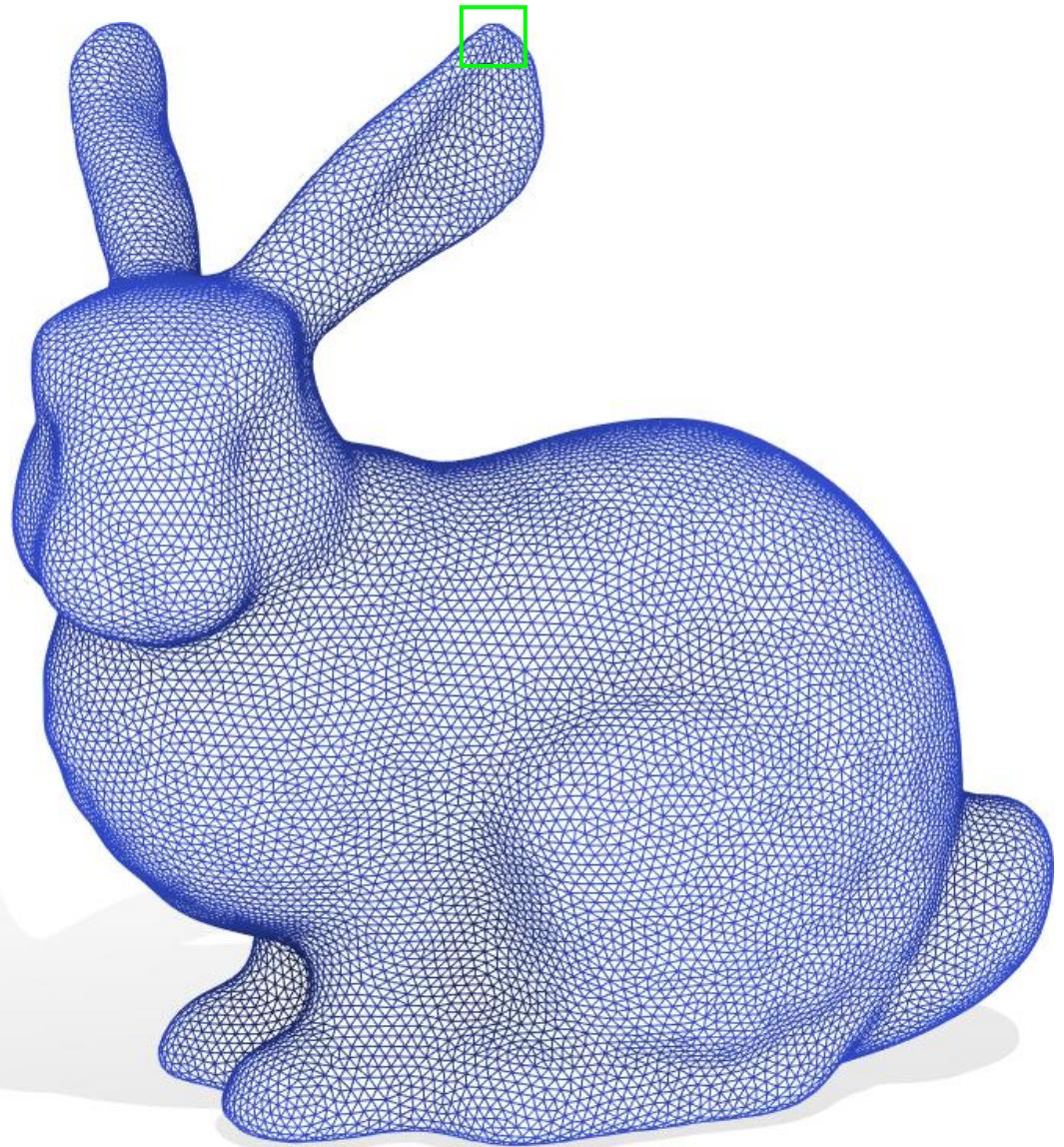
Jacobson & Panozzo, "libigl course"



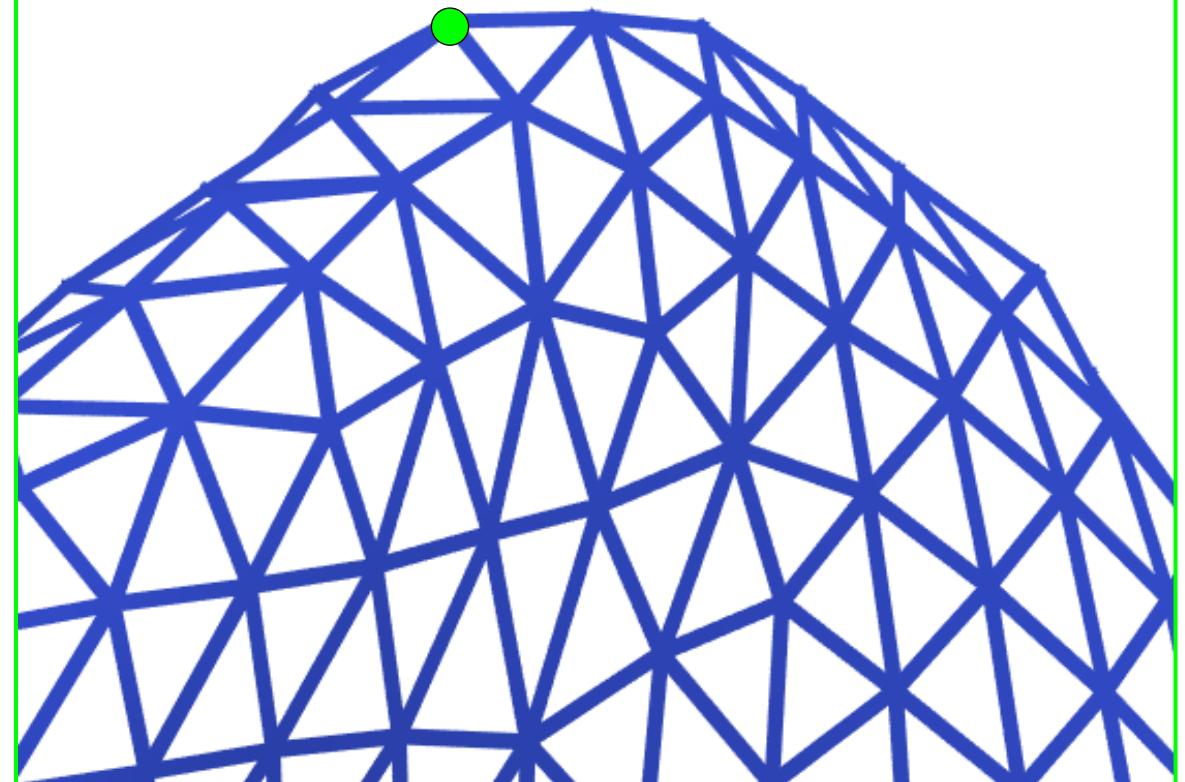
Triangle meshes discretize surfaces

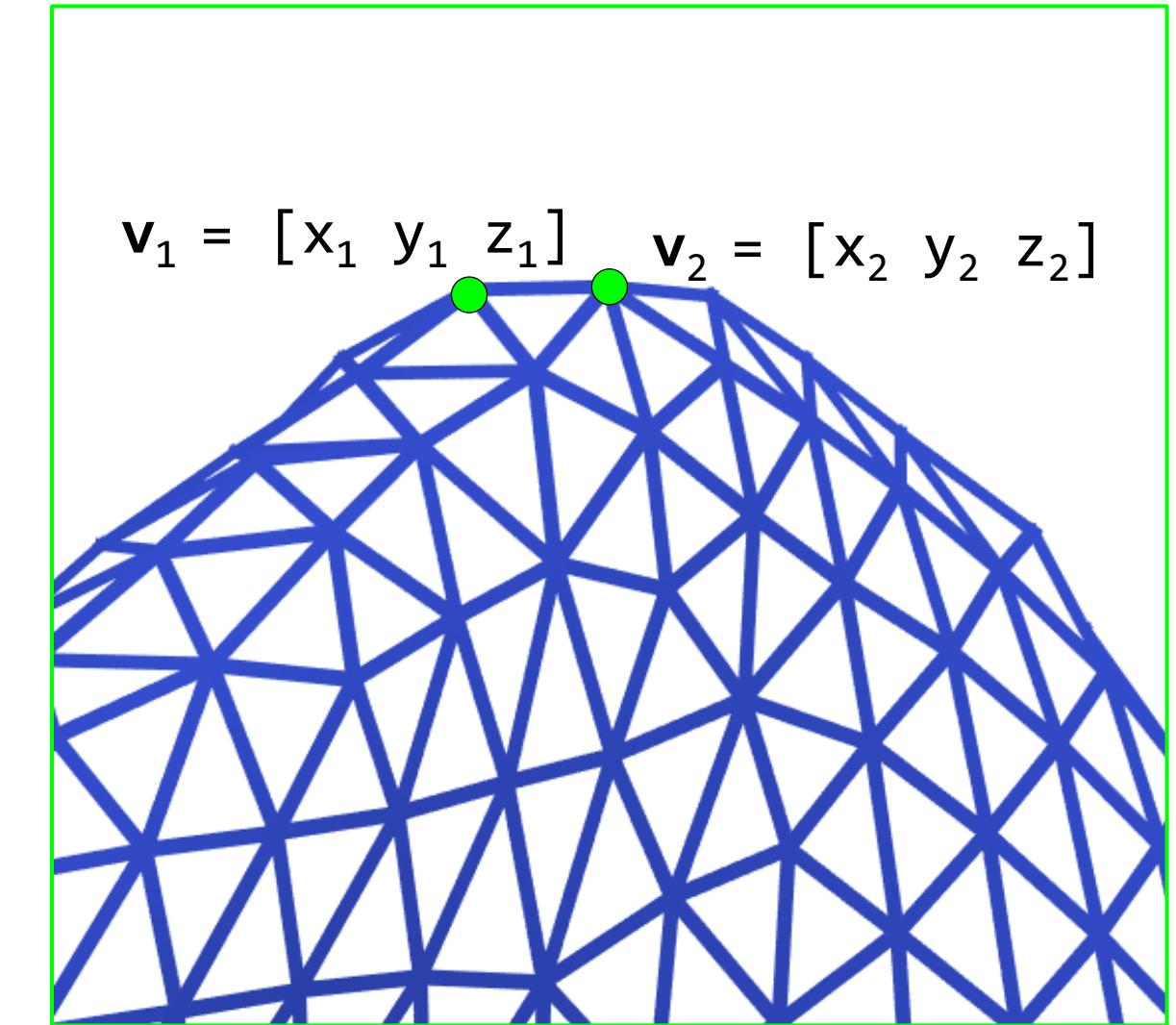
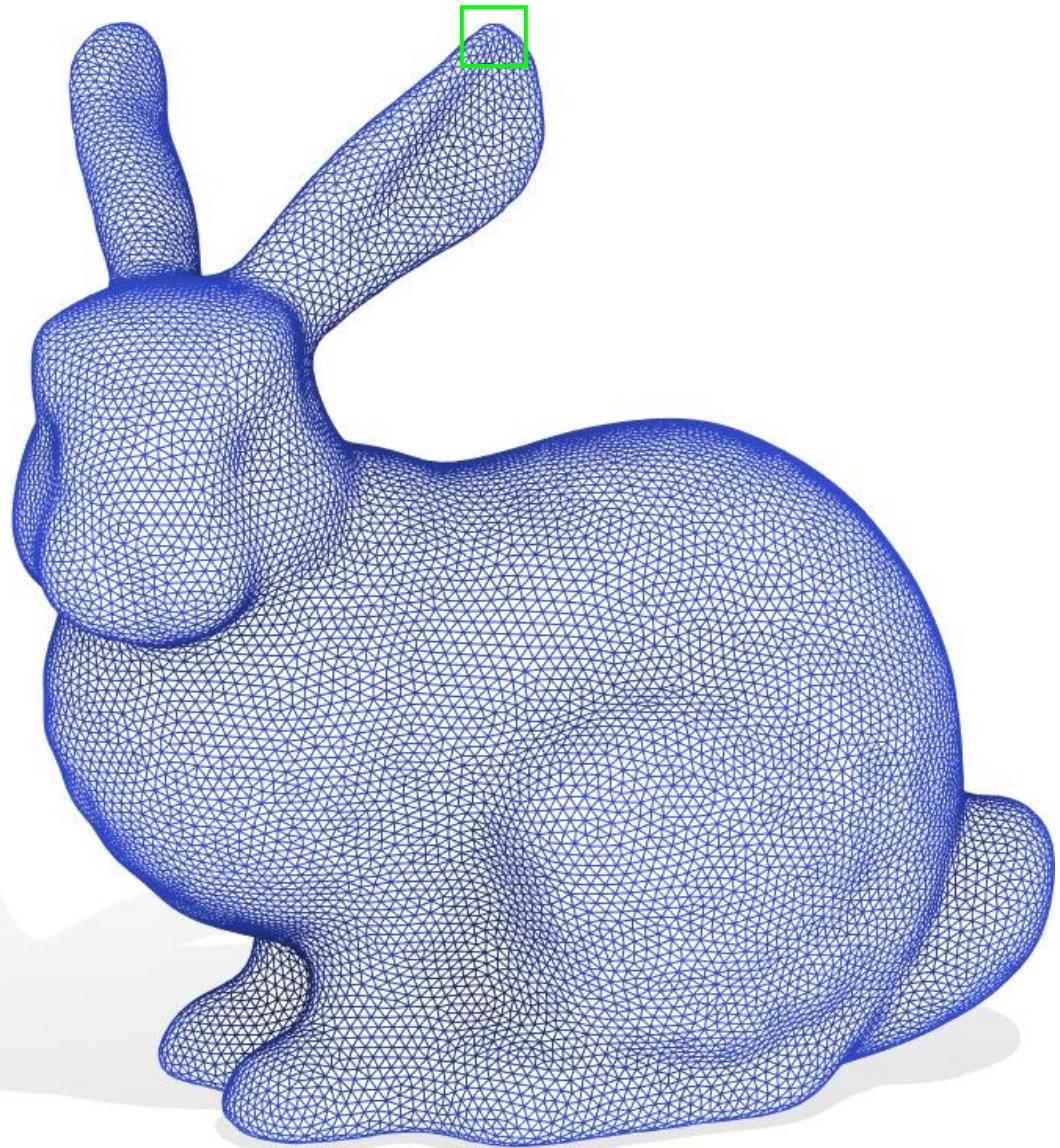
Jacobson & Panozzo, "[libigl course](#)"

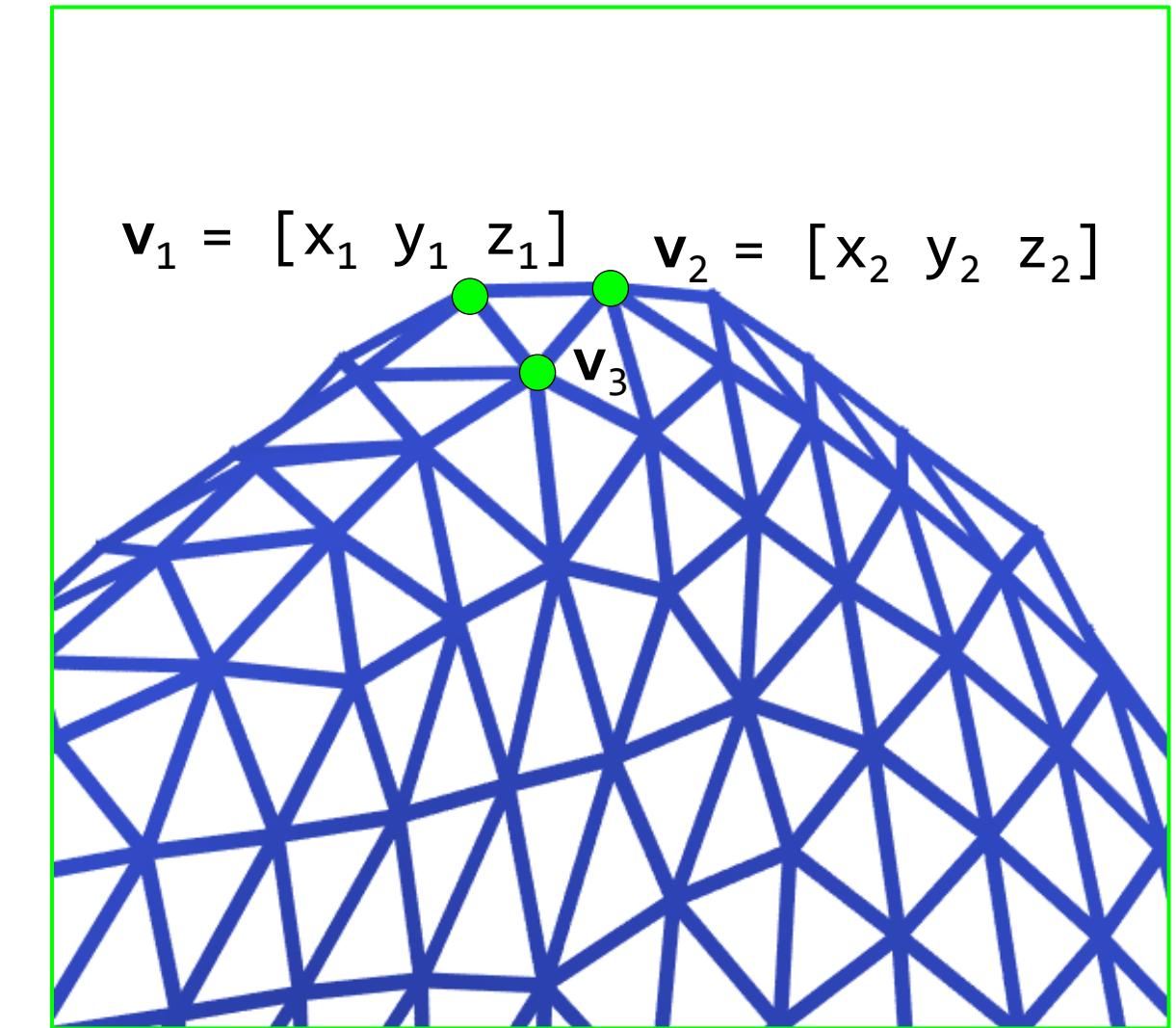
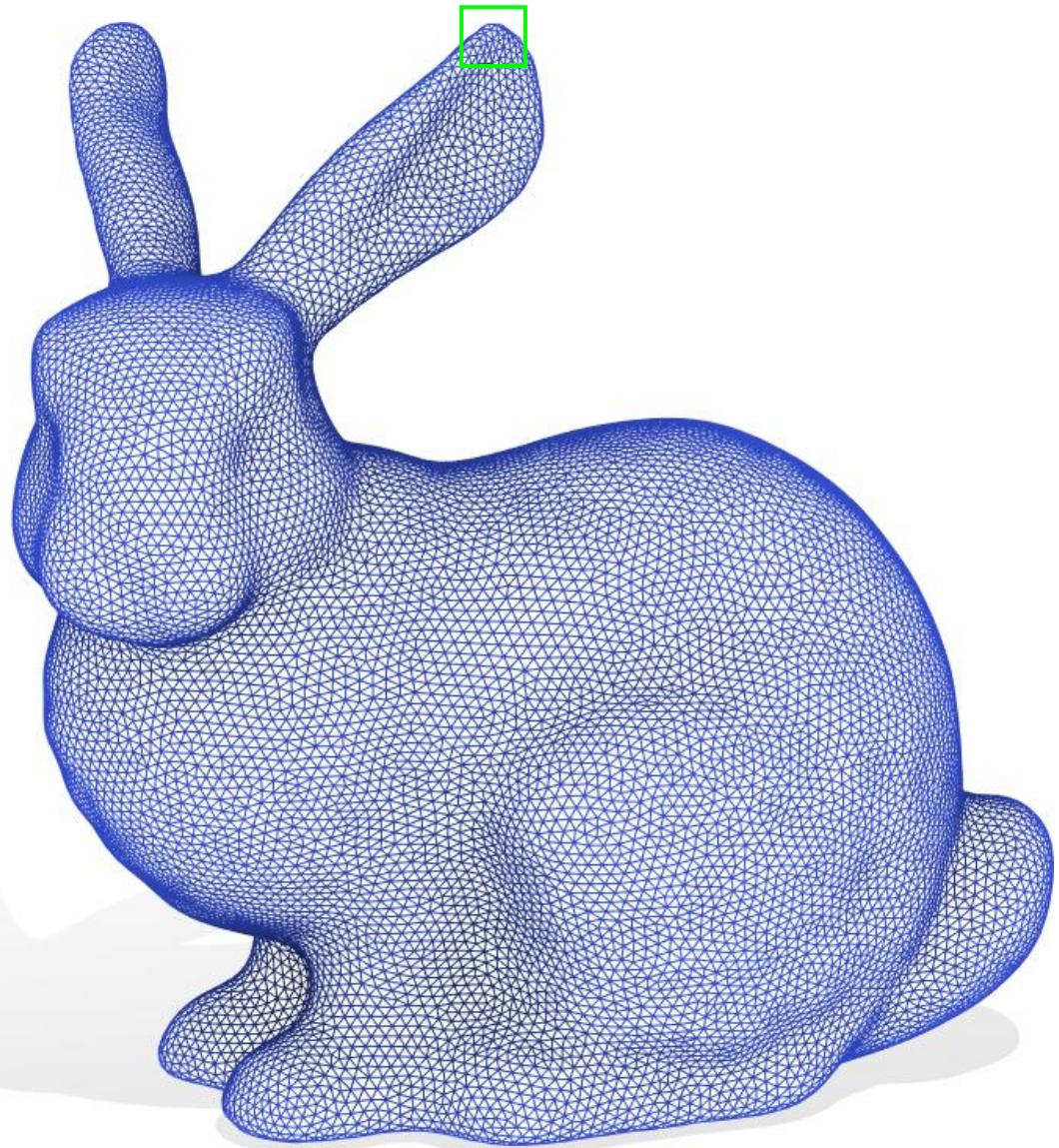


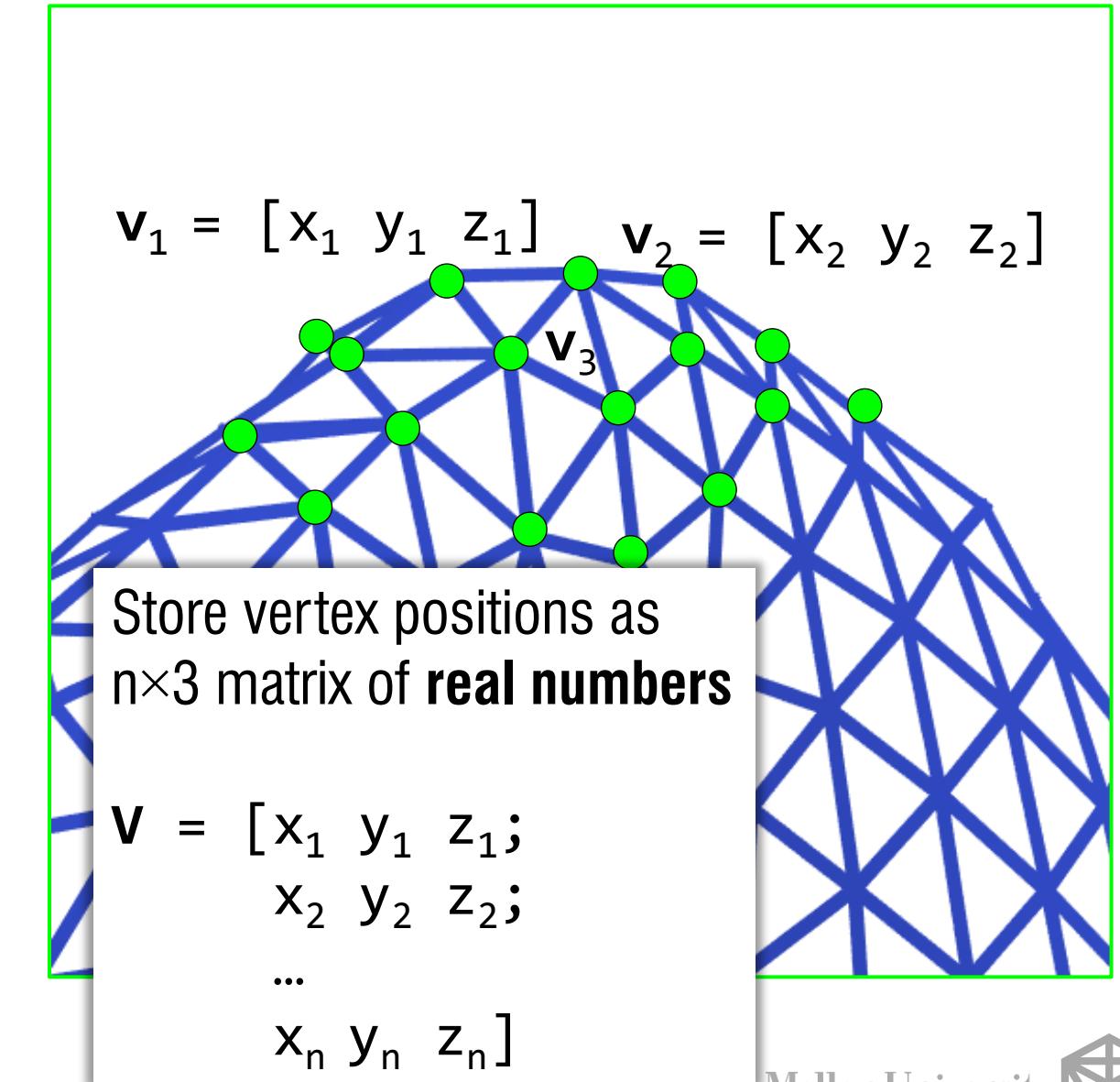
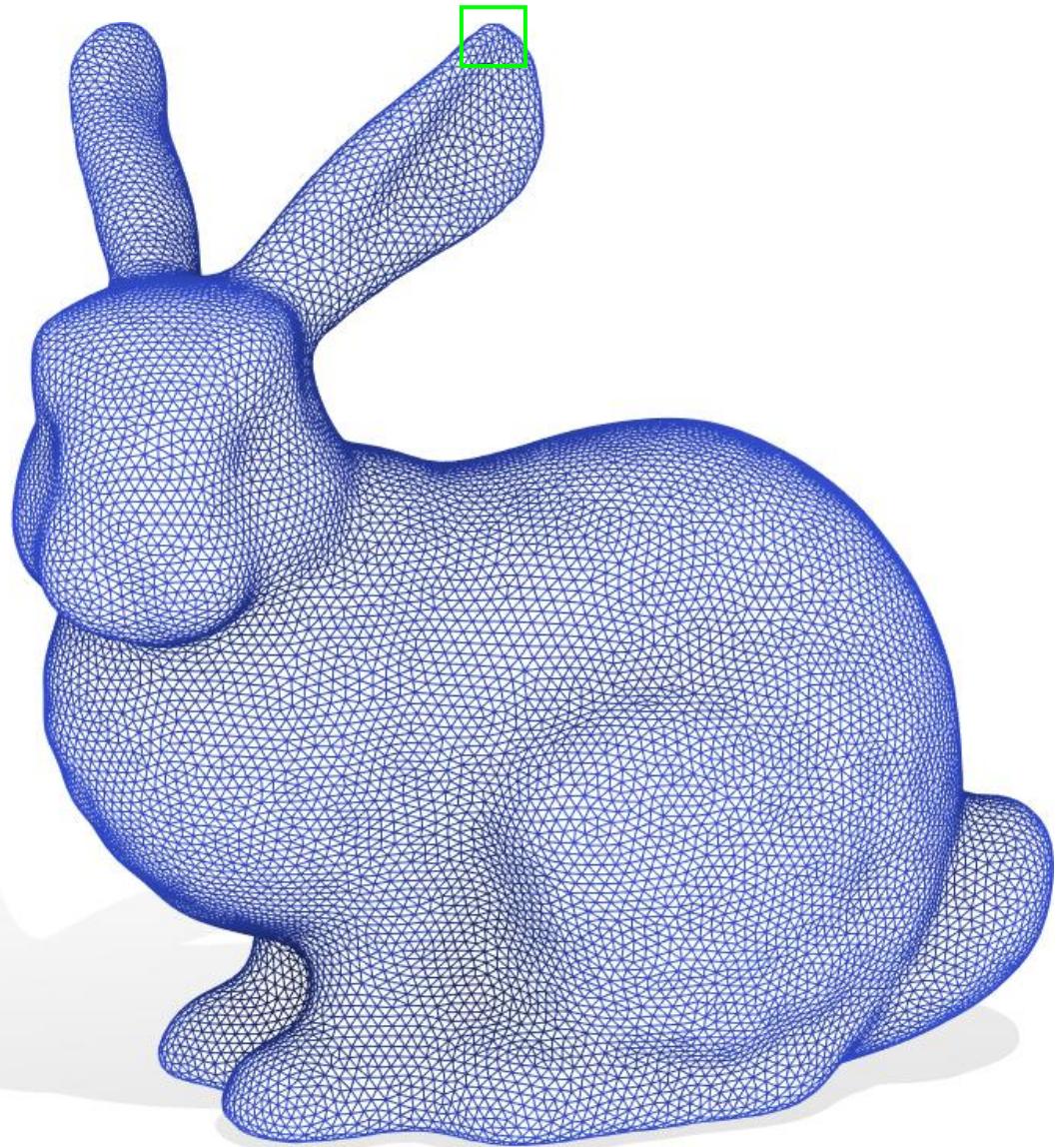


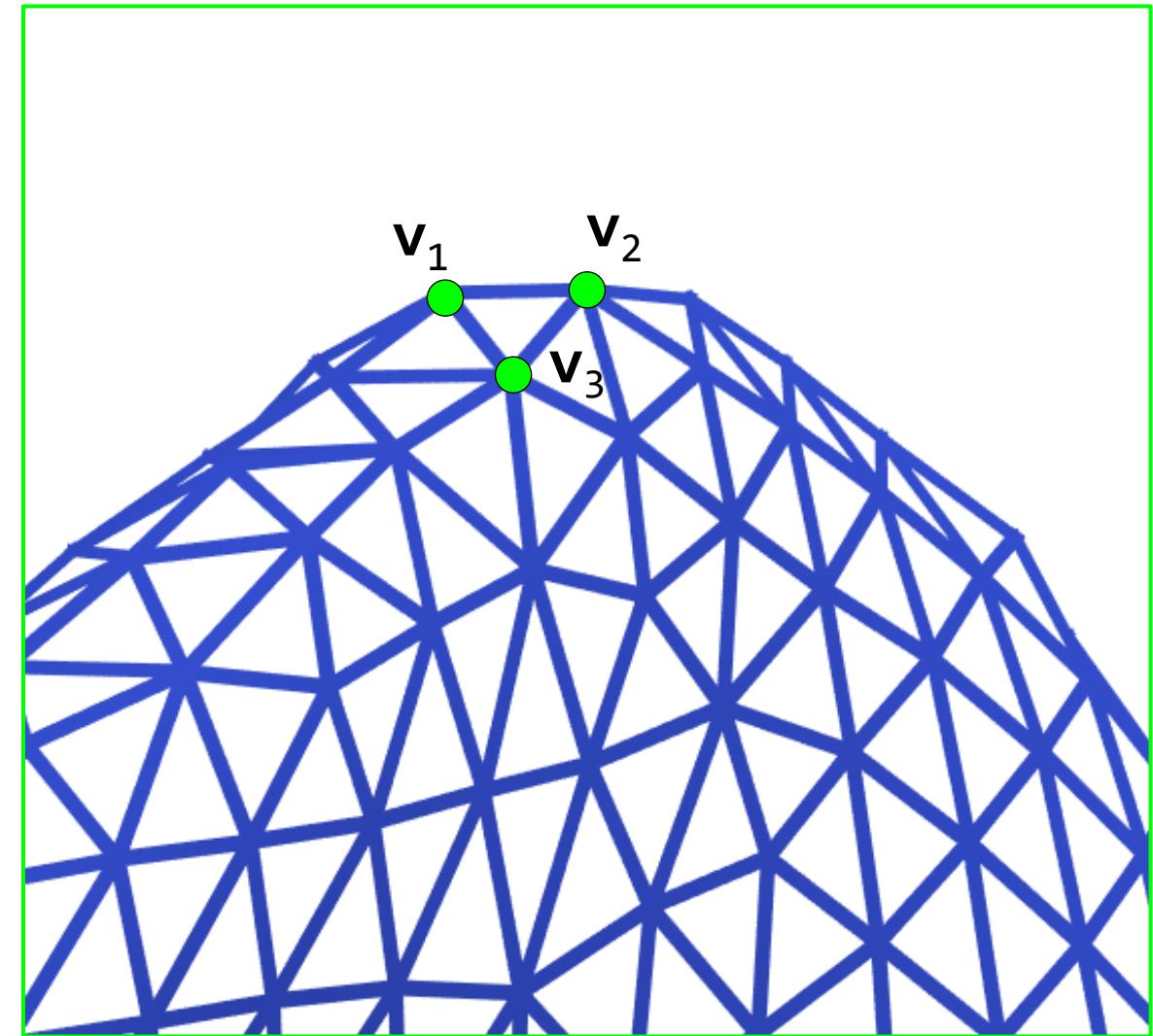
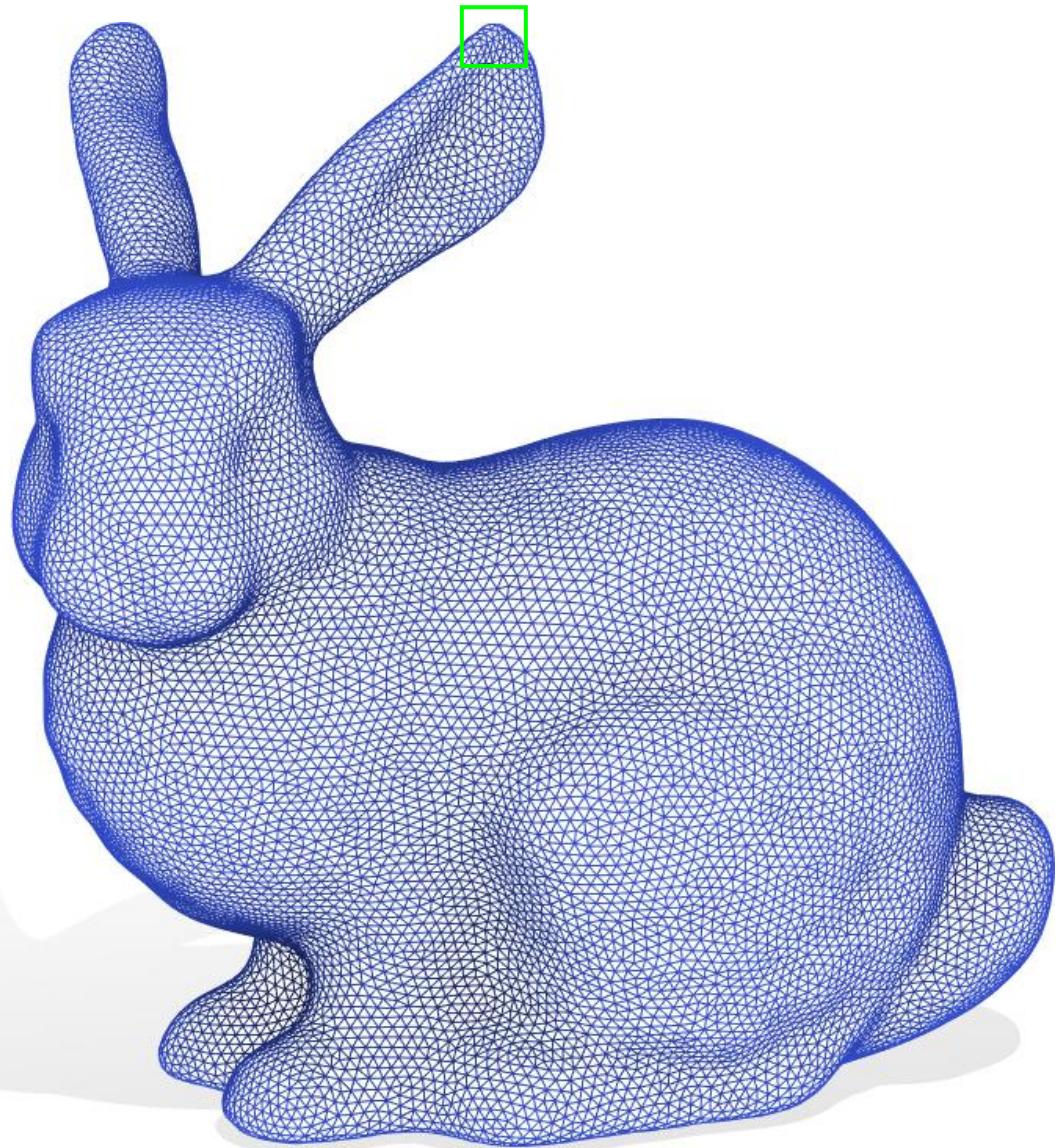
$$\mathbf{v}_1 = [x_1 \ y_1 \ z_1]$$

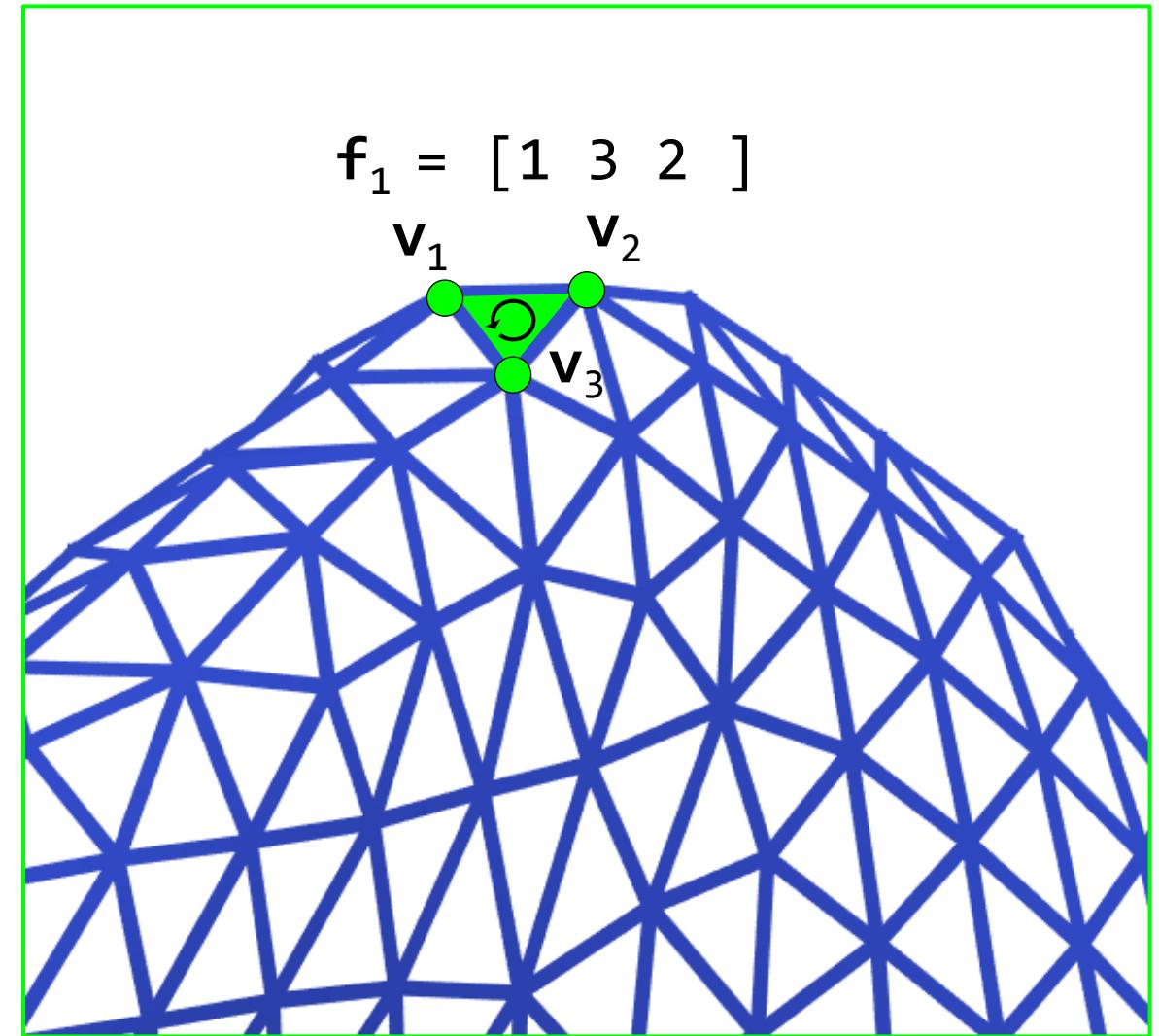
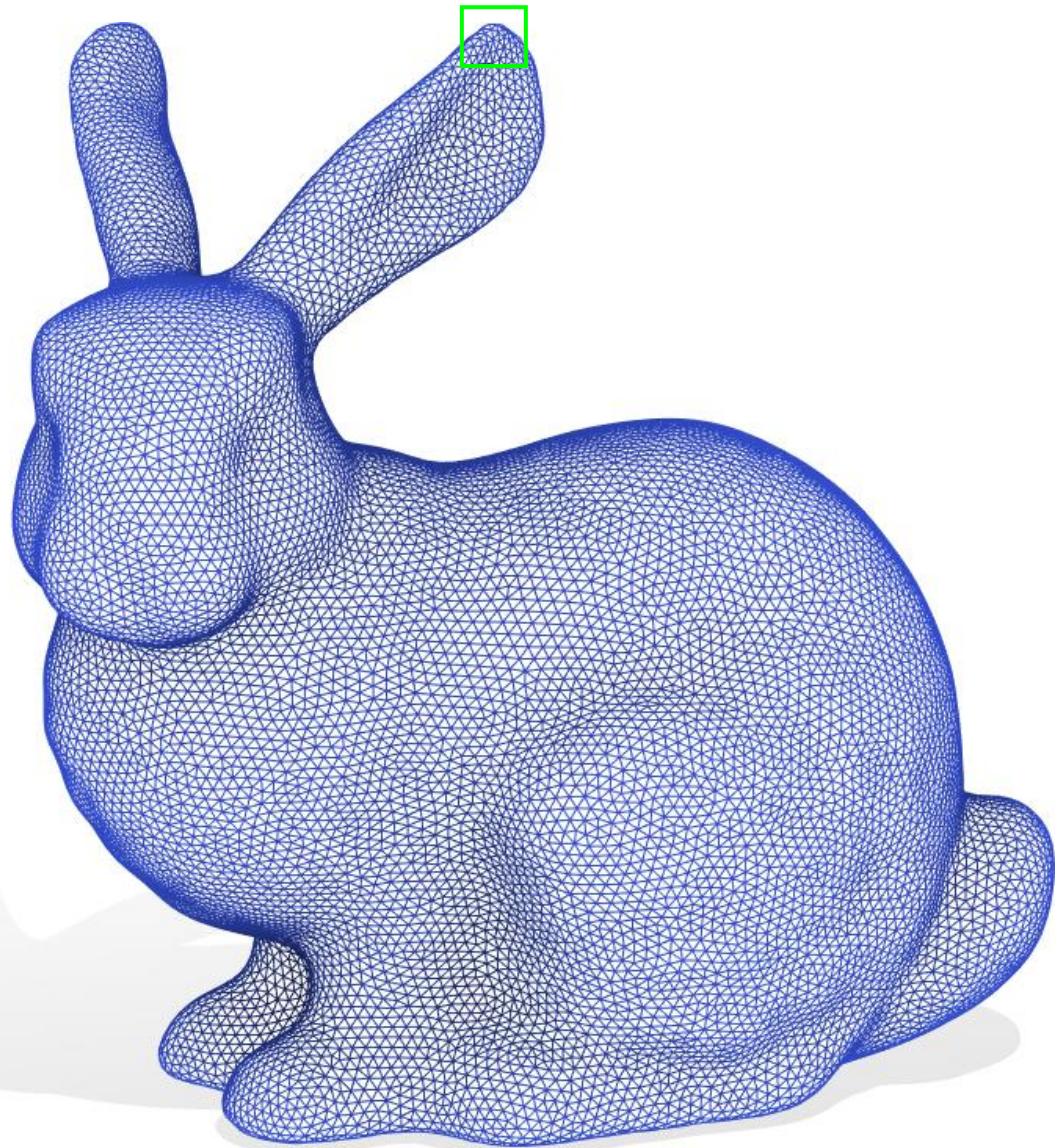


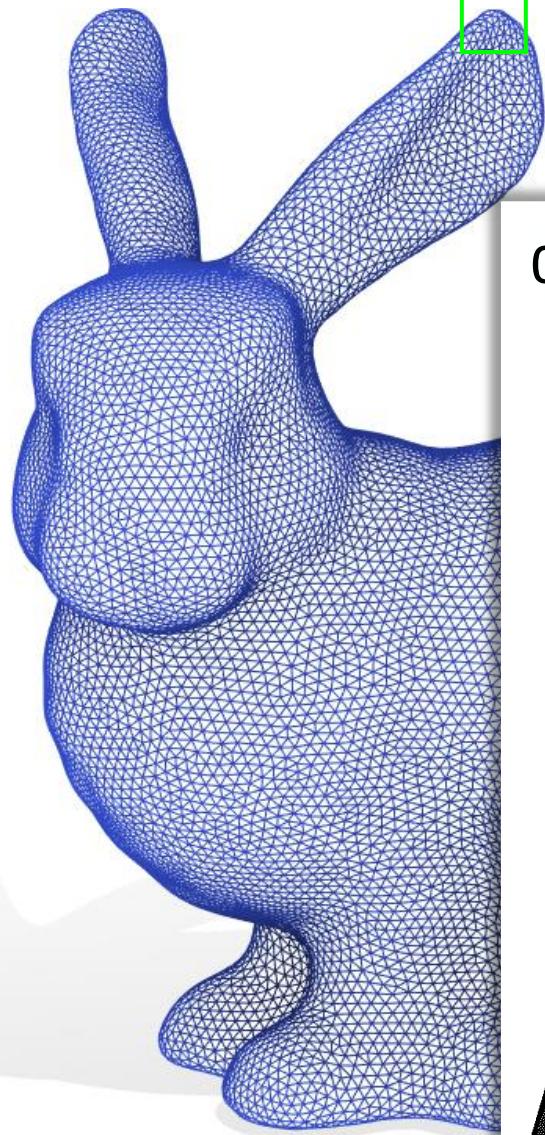




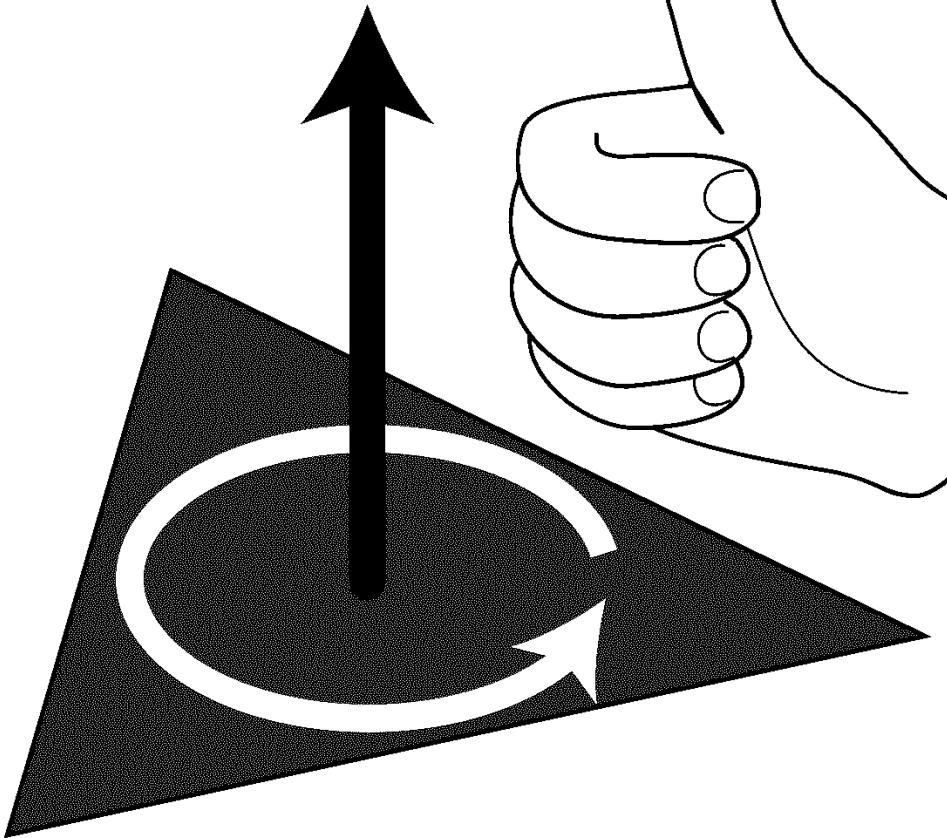




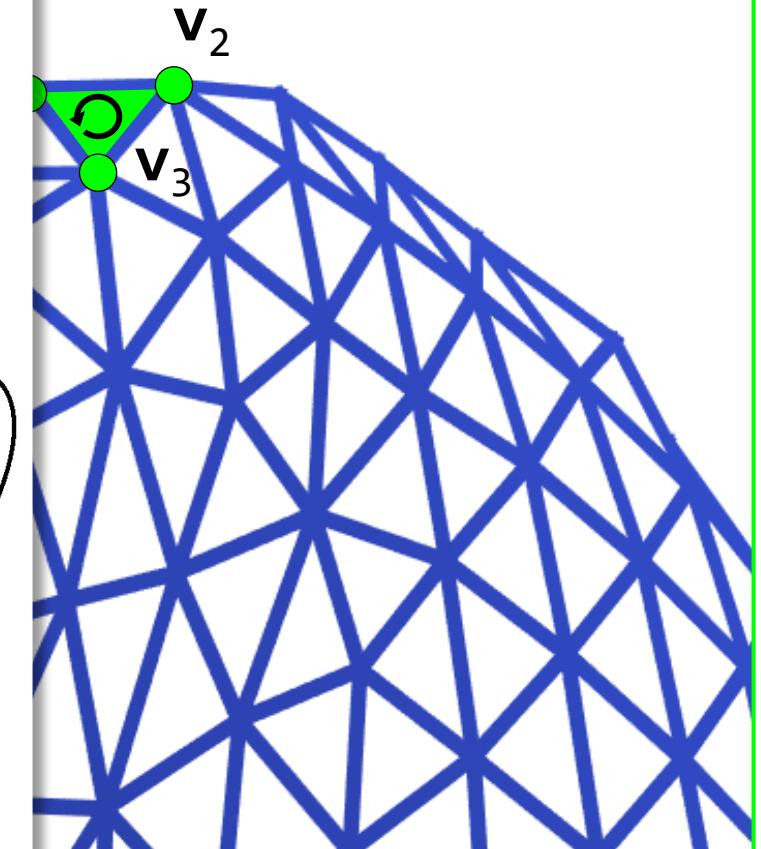


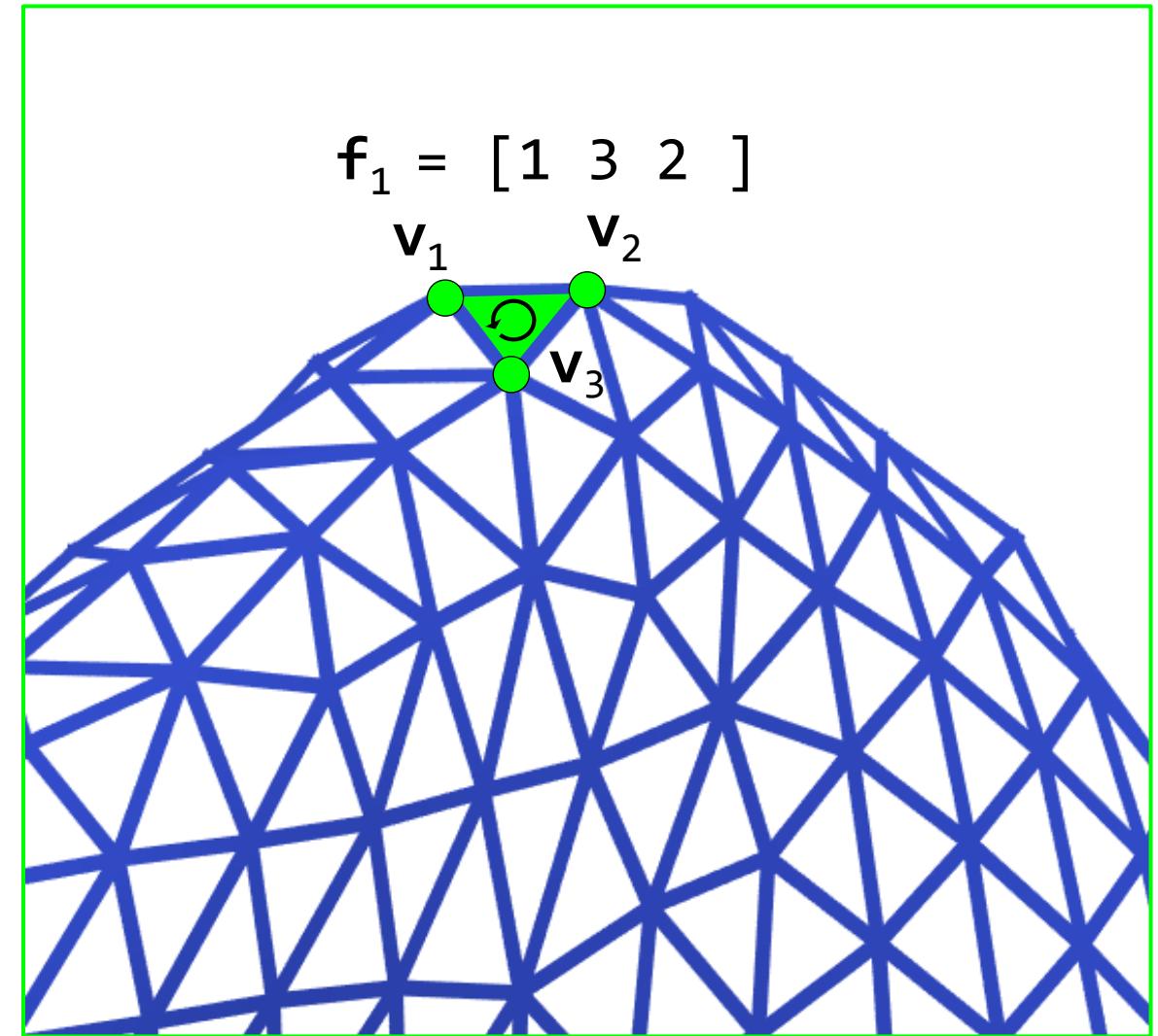
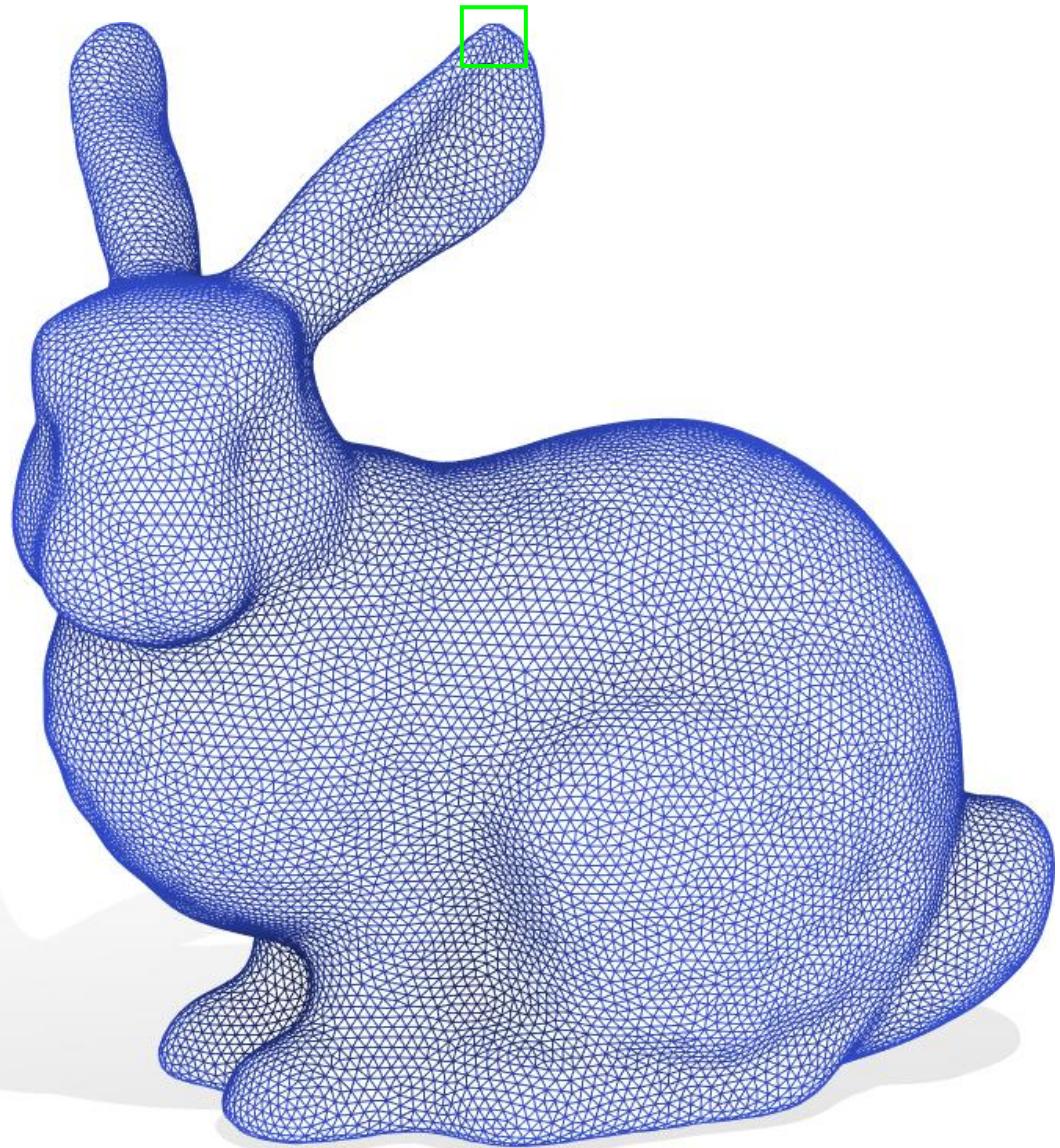


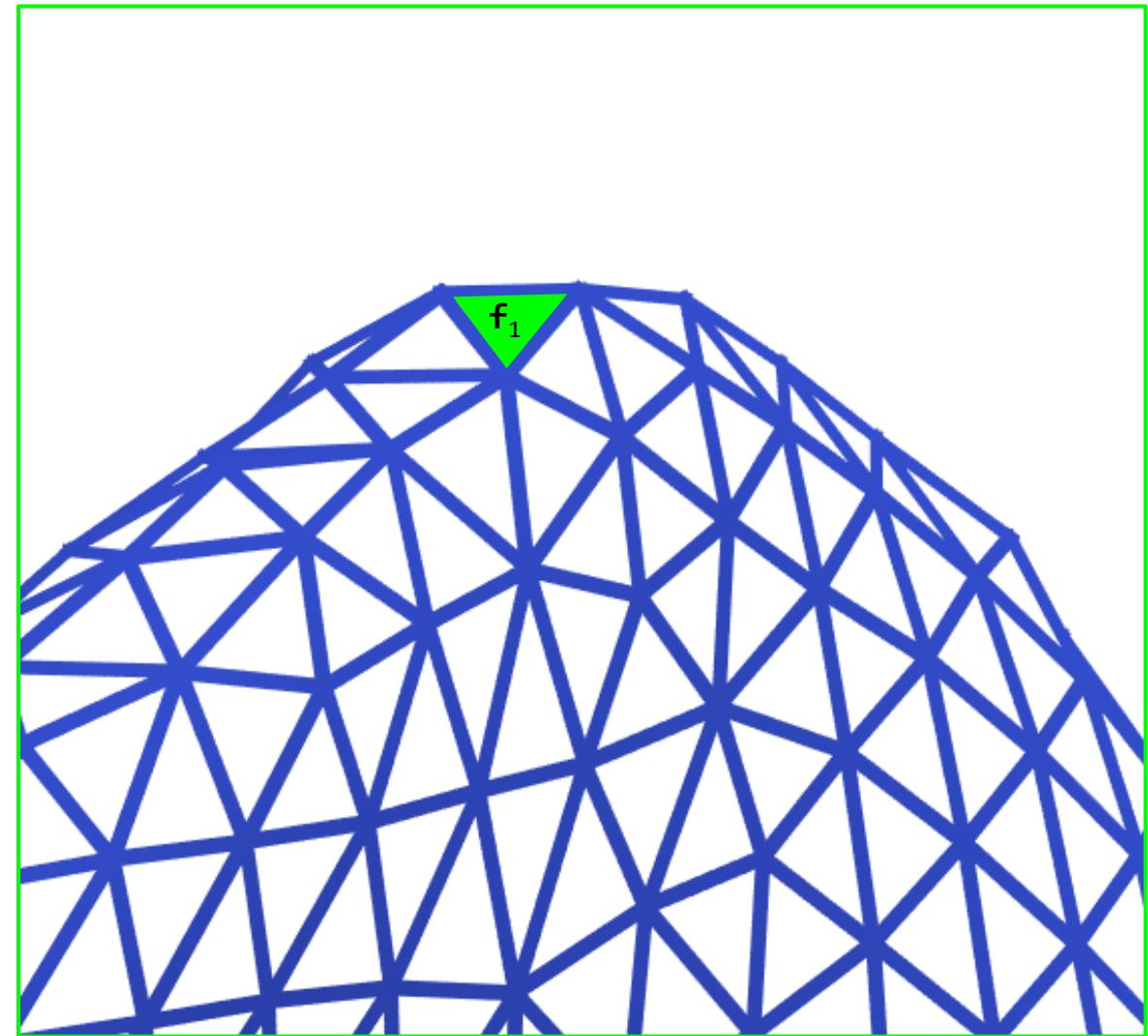
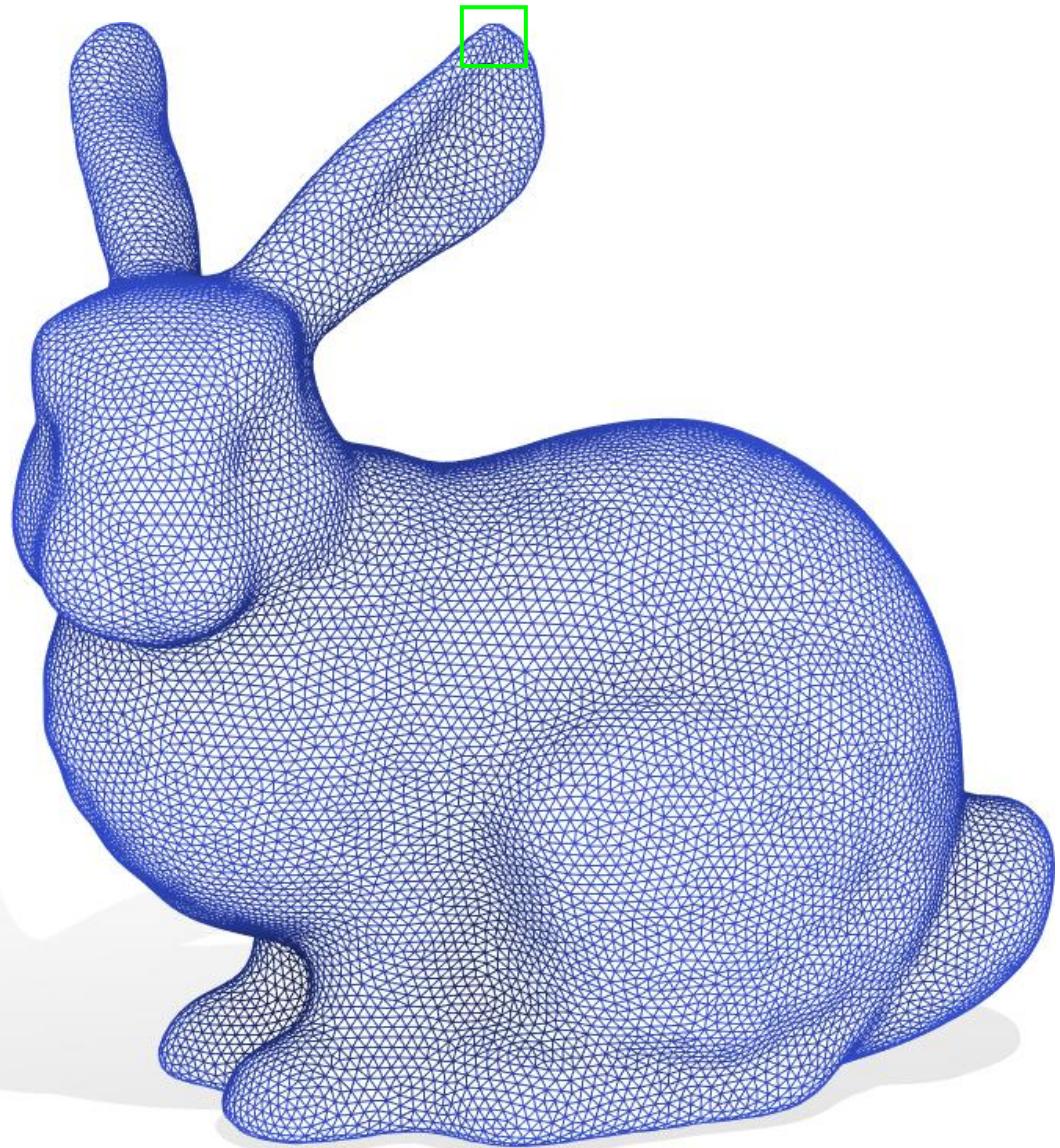
orientation matters!

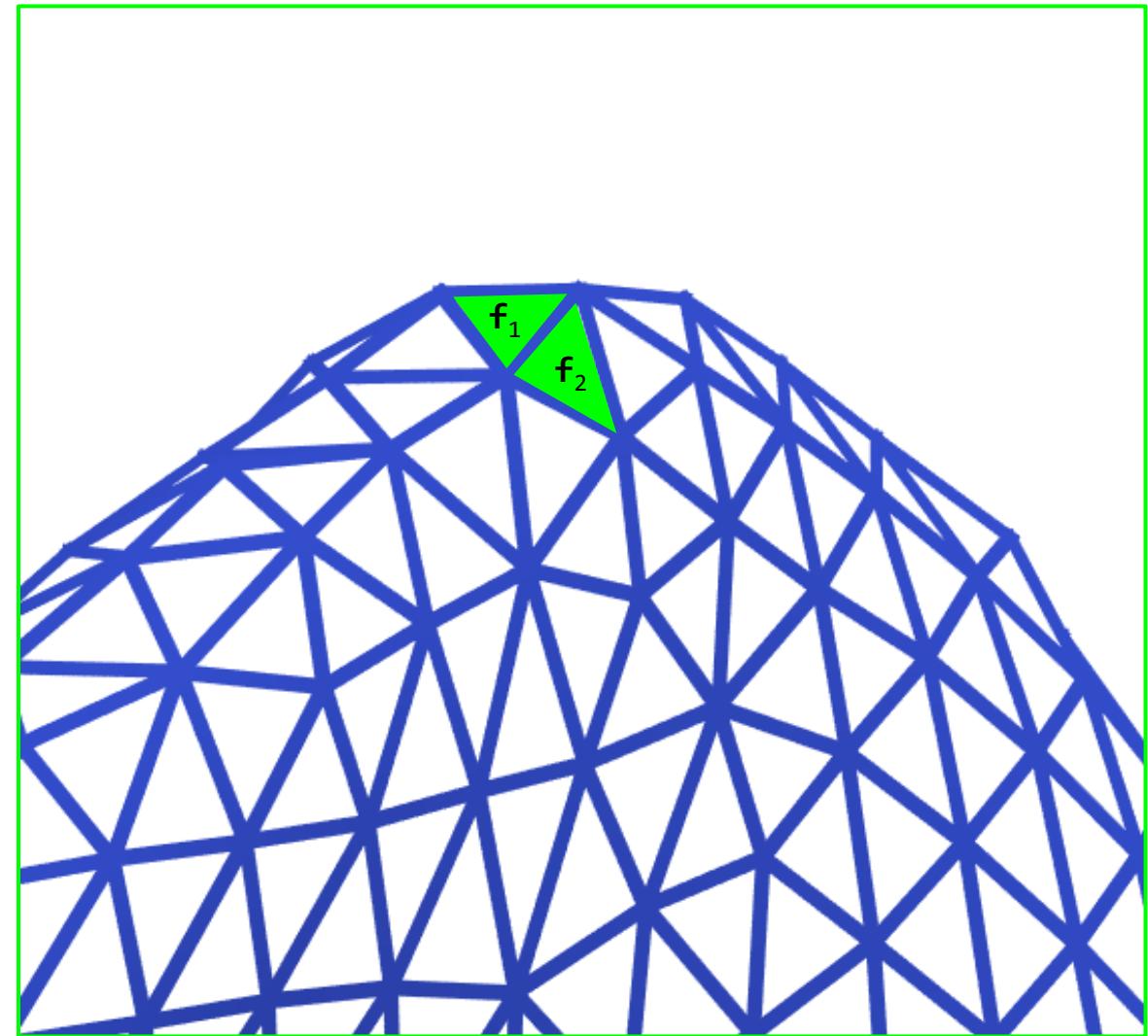
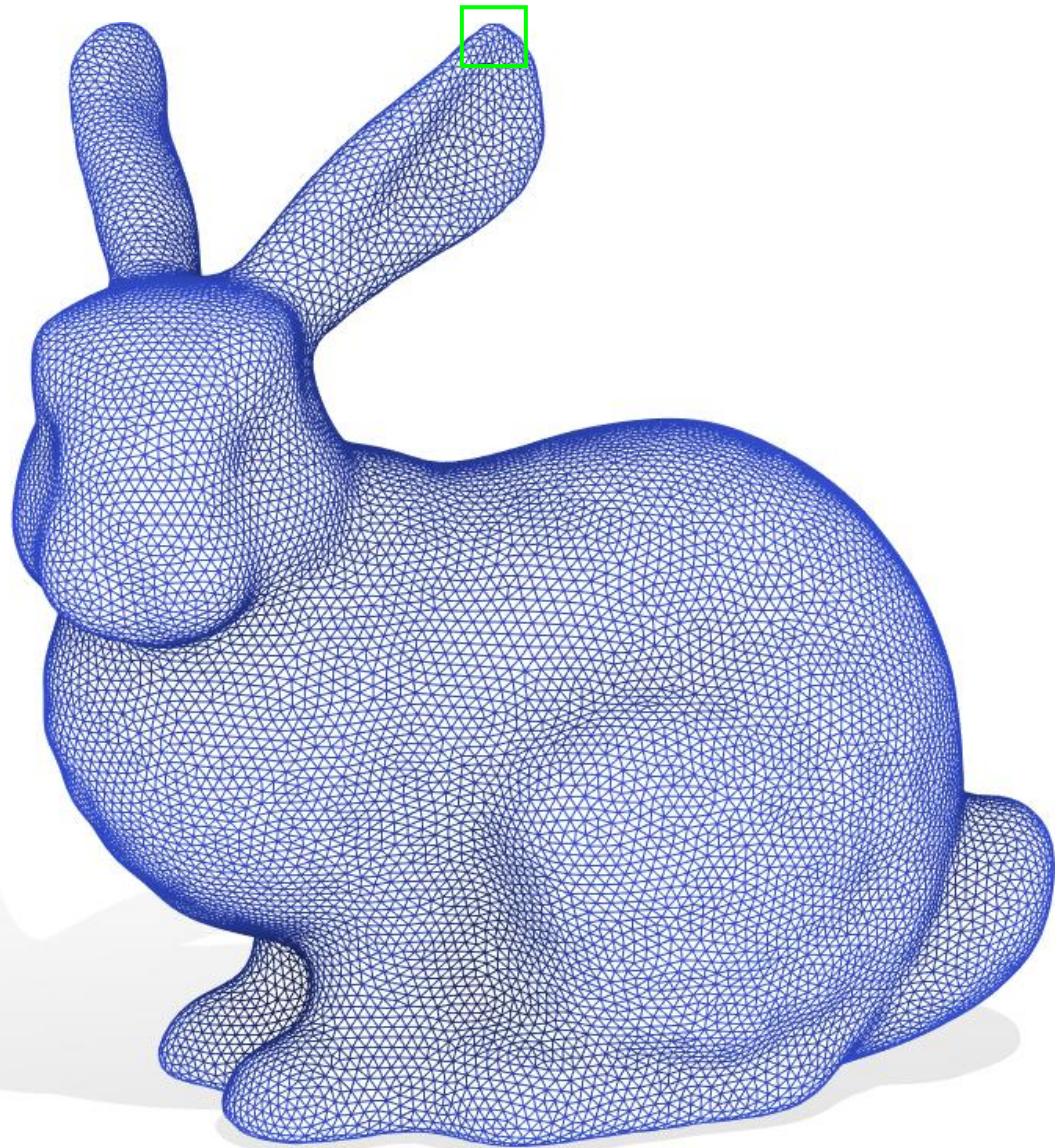


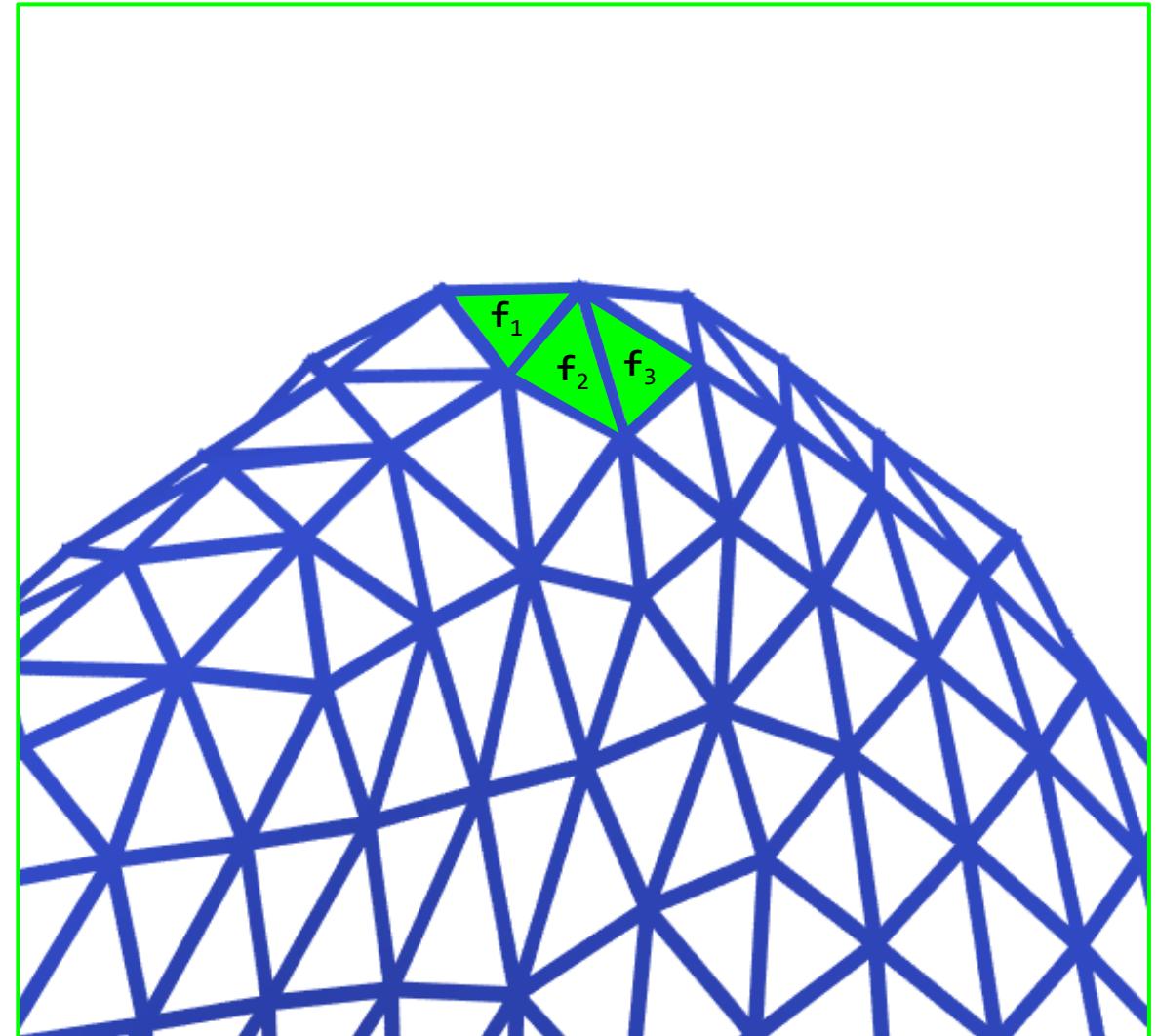
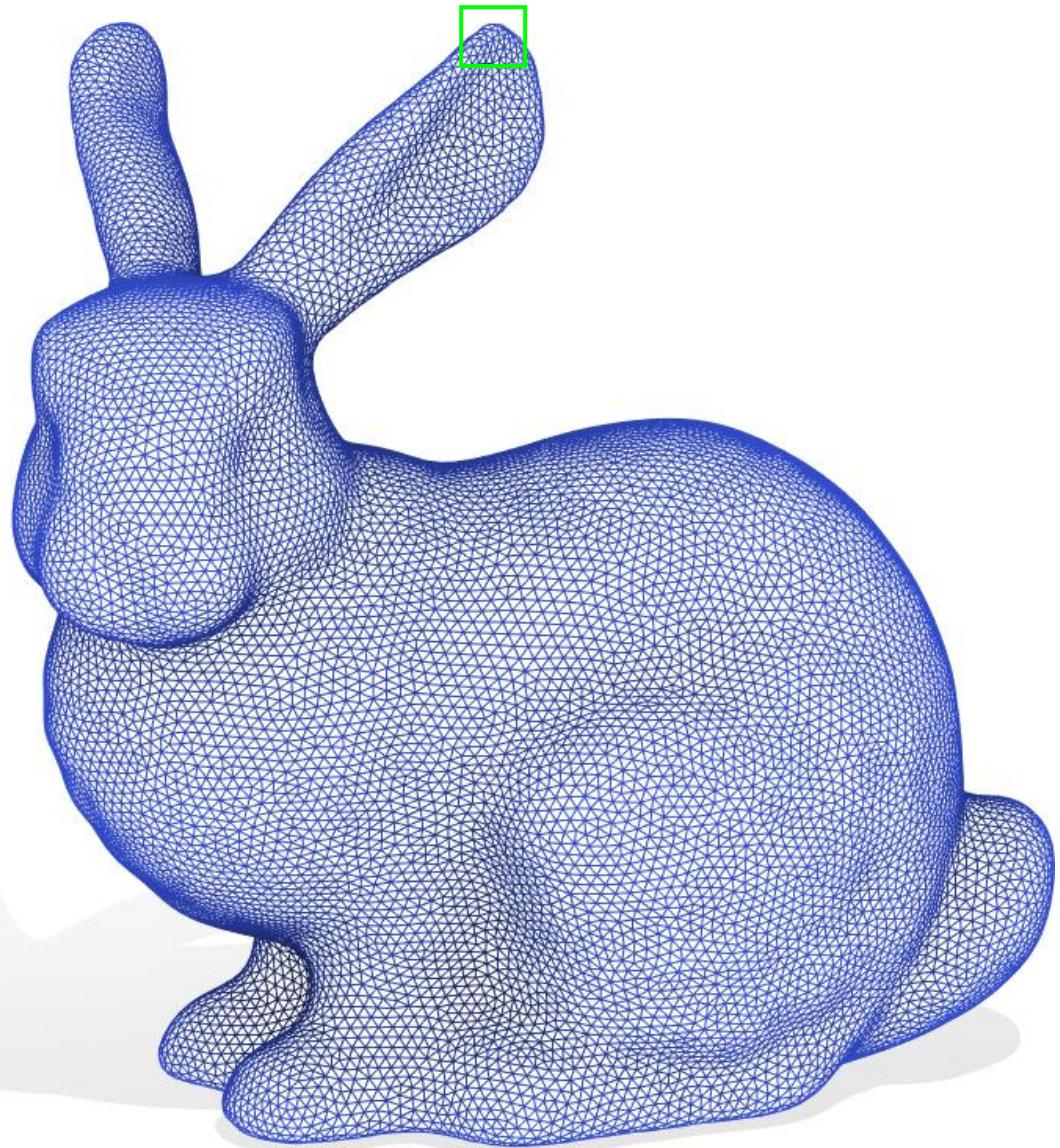
$$\mathbf{f}_1 = [1 \ 3 \ 2]$$

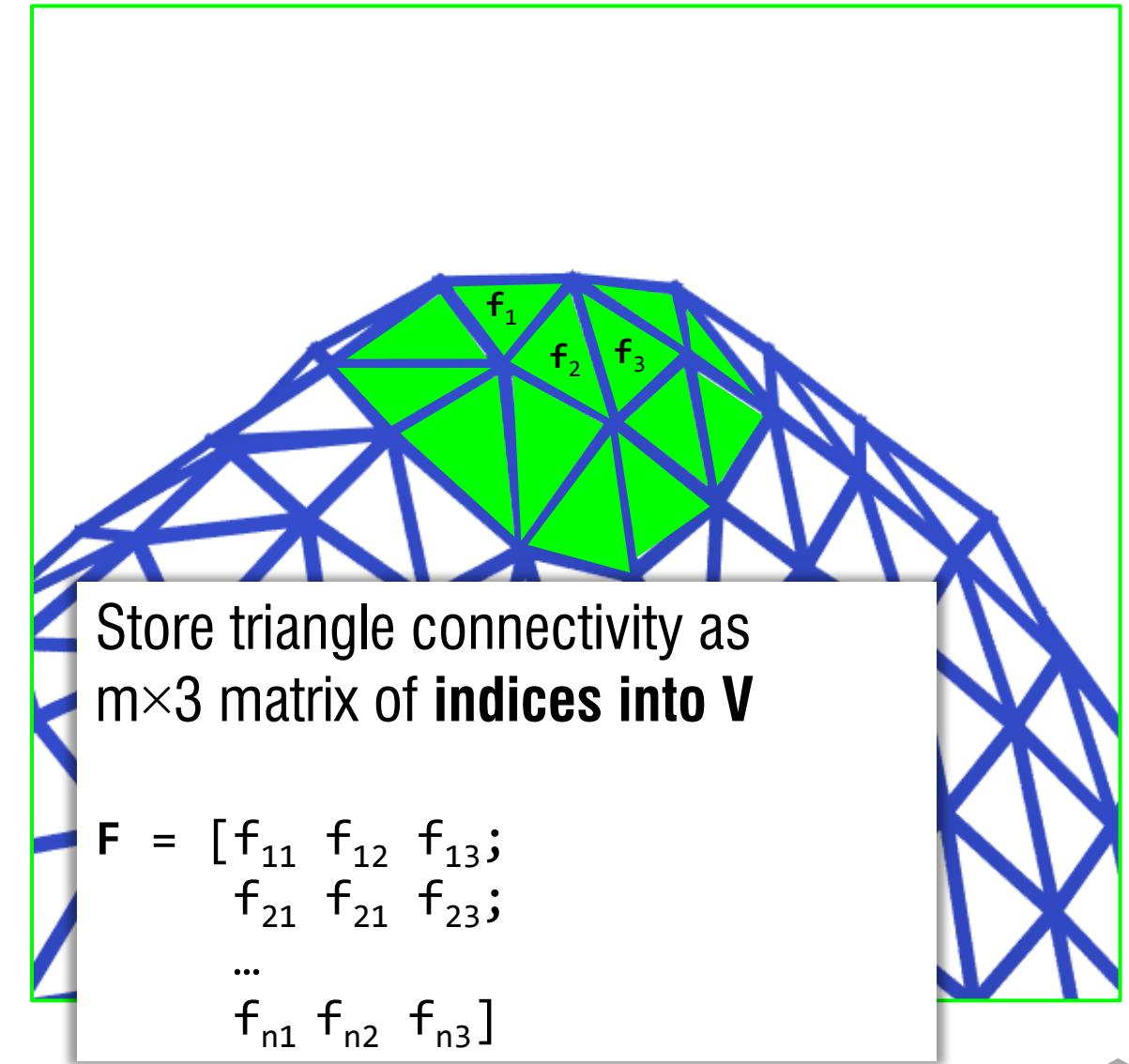
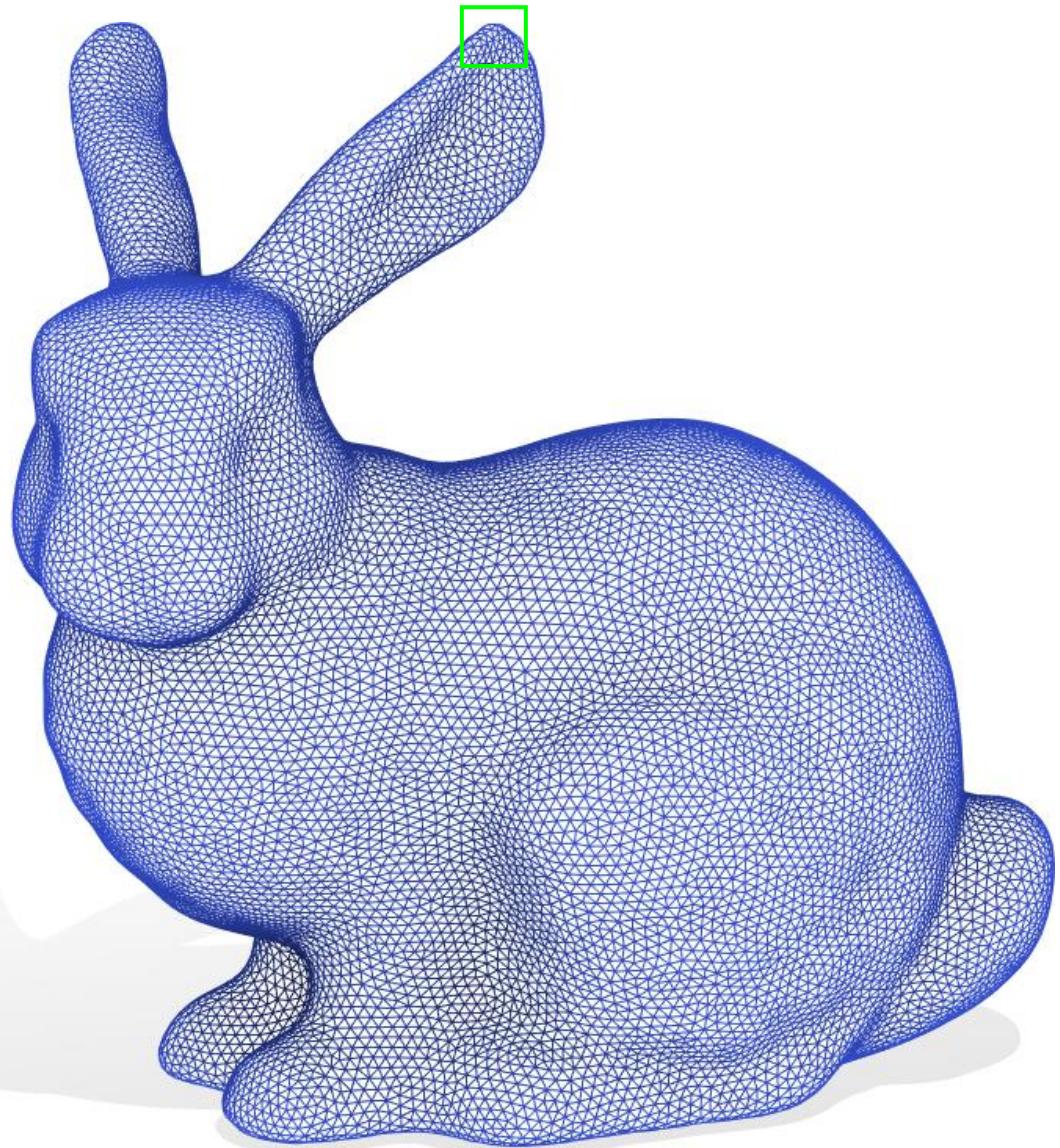


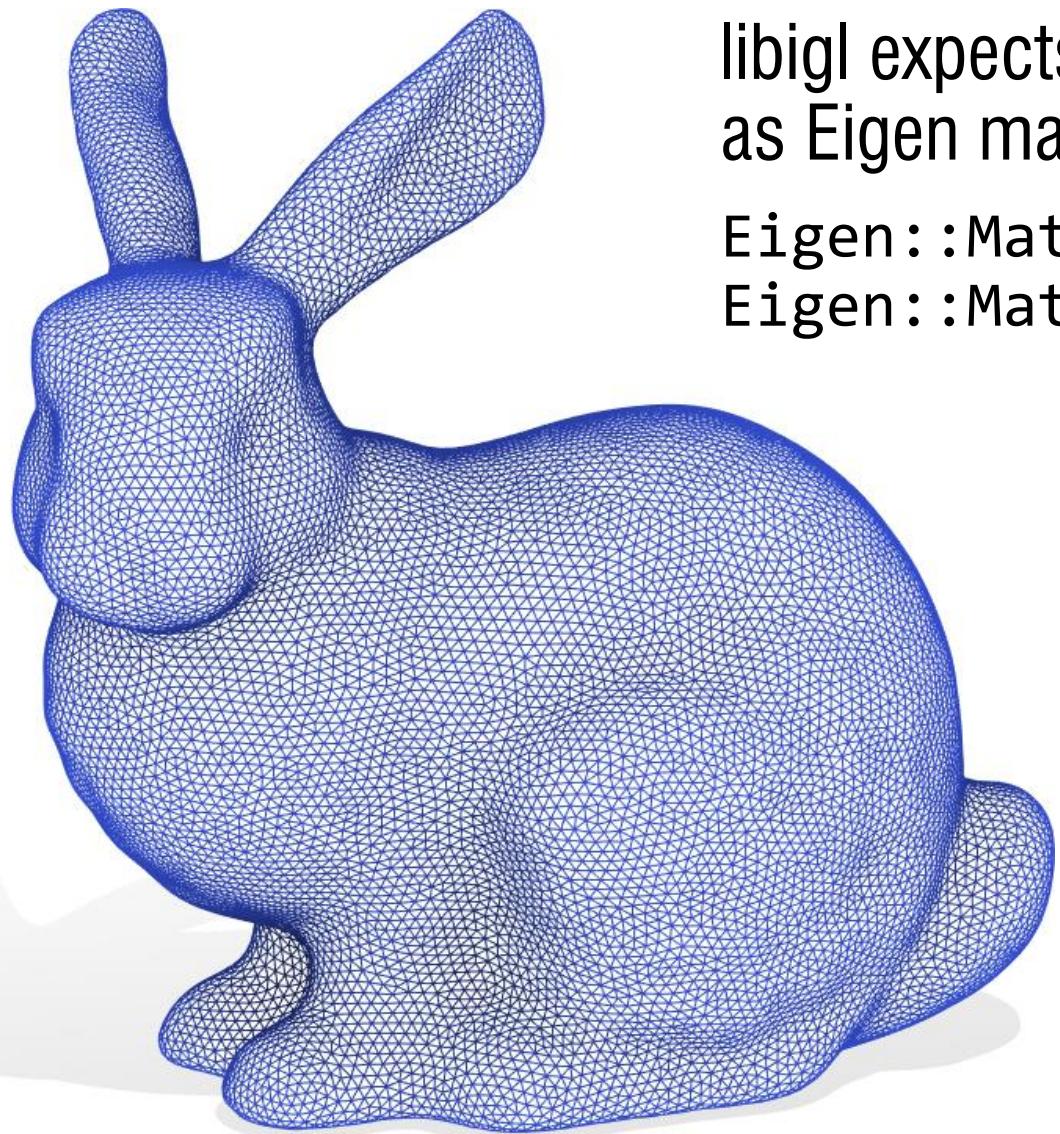








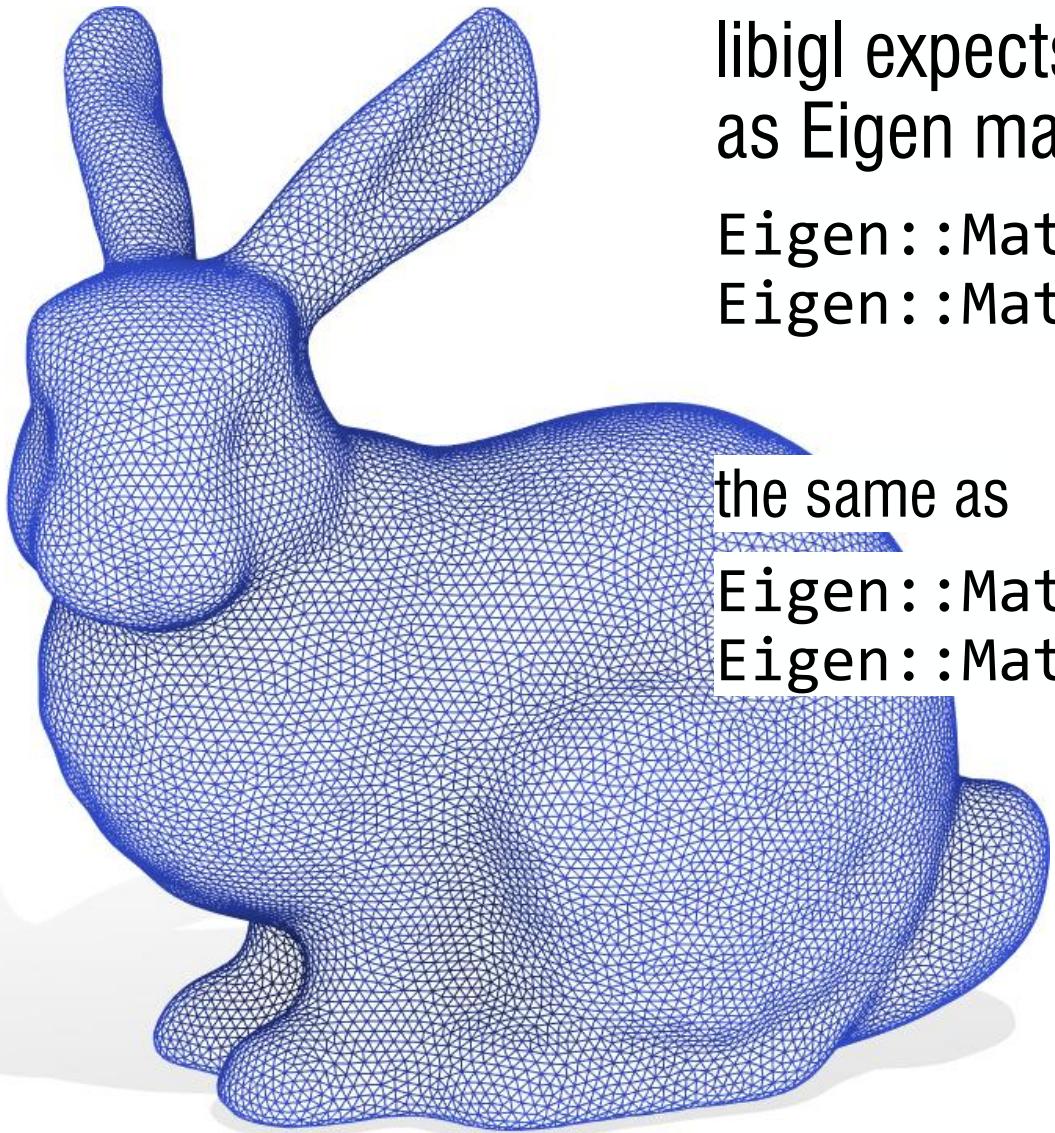




libigl expects vertex positions and triangle indices
as Eigen matrices

```
Eigen::Matrix<double, Eigen::Dynamic, 3> V;  
Eigen::Matrix<int, Eigen::Dynamic, 3> F;
```

Jacobson & Panozzo, "[libigl course](#)"



libigl expects vertex positions and triangle indices
as Eigen matrices

```
Eigen::Matrix<double, Eigen::Dynamic, 3> V;  
Eigen::Matrix<int, Eigen::Dynamic, 3> F;
```

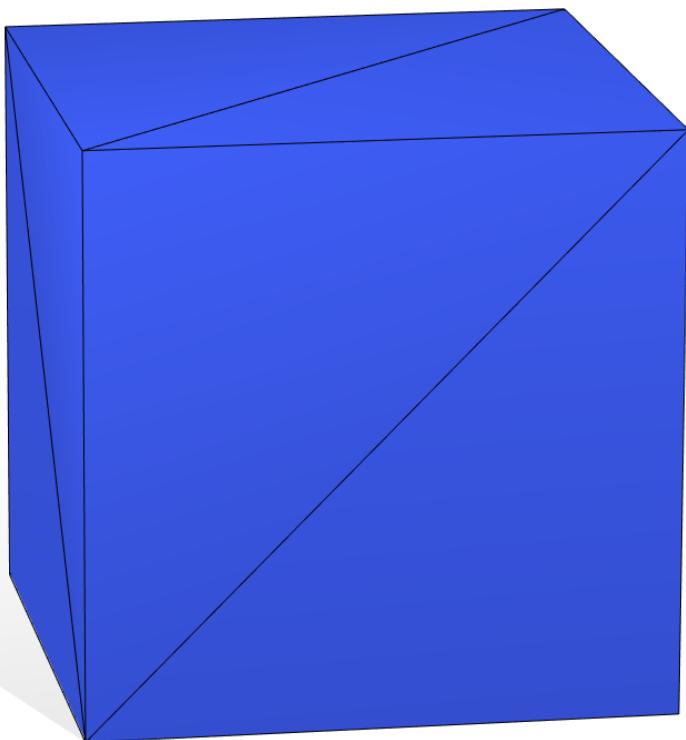
the same as

```
Eigen::MatrixXd V;  
Eigen::MatrixXi F;
```

Jacobson & Panozzo, "[libigl course](#)"



libigl expects vertex positions and triangle indices as Eigen matrices

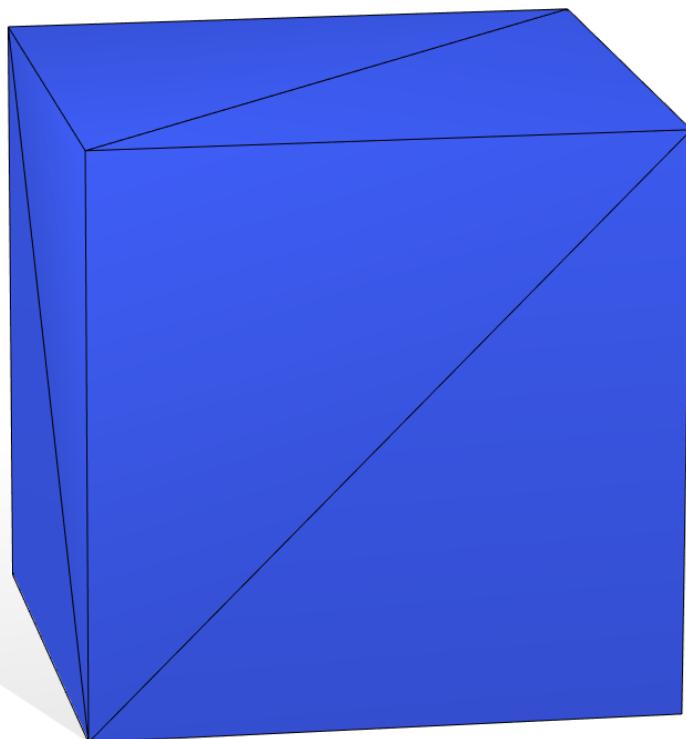


```
Eigen::MatrixXd V(8,3);  
V<<  
    0.0, 0.0, 0.0,  
    0.0, 0.0, 1.0,  
    0.0, 1.0, 0.0,  
    0.0, 1.0, 1.0,  
    1.0, 0.0, 0.0,  
    1.0, 0.0, 1.0,  
    1.0, 1.0, 0.0,  
    1.0, 1.0, 1.0;
```

```
Eigen::MatrixXi F(12,3);  
F<<  
    0, 6, 4,  
    0, 2, 6,  
    0, 3, 2,  
    0, 1, 3,  
    2, 7, 6,  
    2, 3, 7,  
    4, 6, 7,  
    4, 7, 5,  
    0, 4, 5,  
    0, 5, 1,  
    1, 5, 7,  
    1, 7, 3;
```

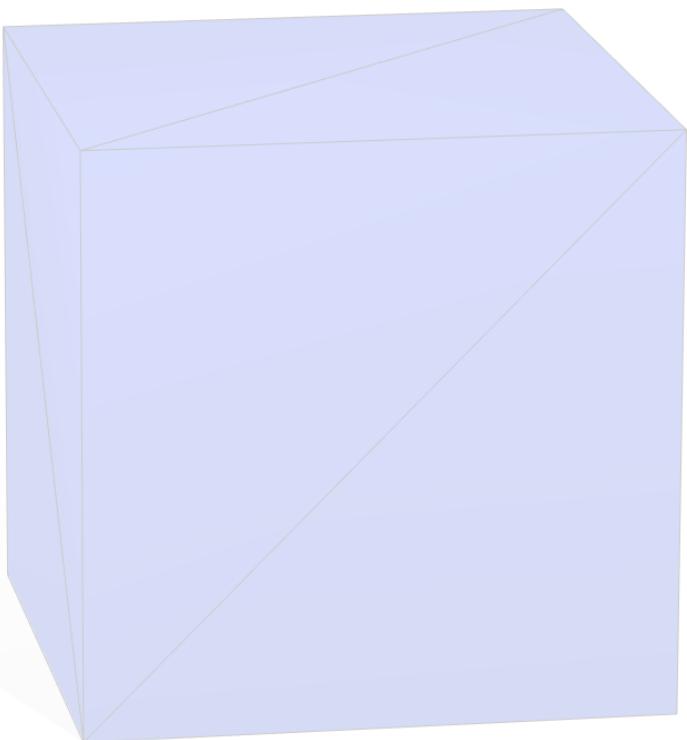
Jacobson & Panozzo, "[libigl course](#)"

libigl expects vertex positions and triangle indices
as Eigen matrices



```
Eigen::MatrixXd V;  
Eigen::MatrixXi F;  
igl::read_triangle_mesh("cube.obj",V,F);
```

Jacobson & Panozzo, "libigl course"



libigl expects vertex positions and triangle indices
as Eigen matrices

```
Eigen::Matrix<double> V;
Eigen::Matrix<int> F;
igl::read_obj("cube.obj",V,F);
```

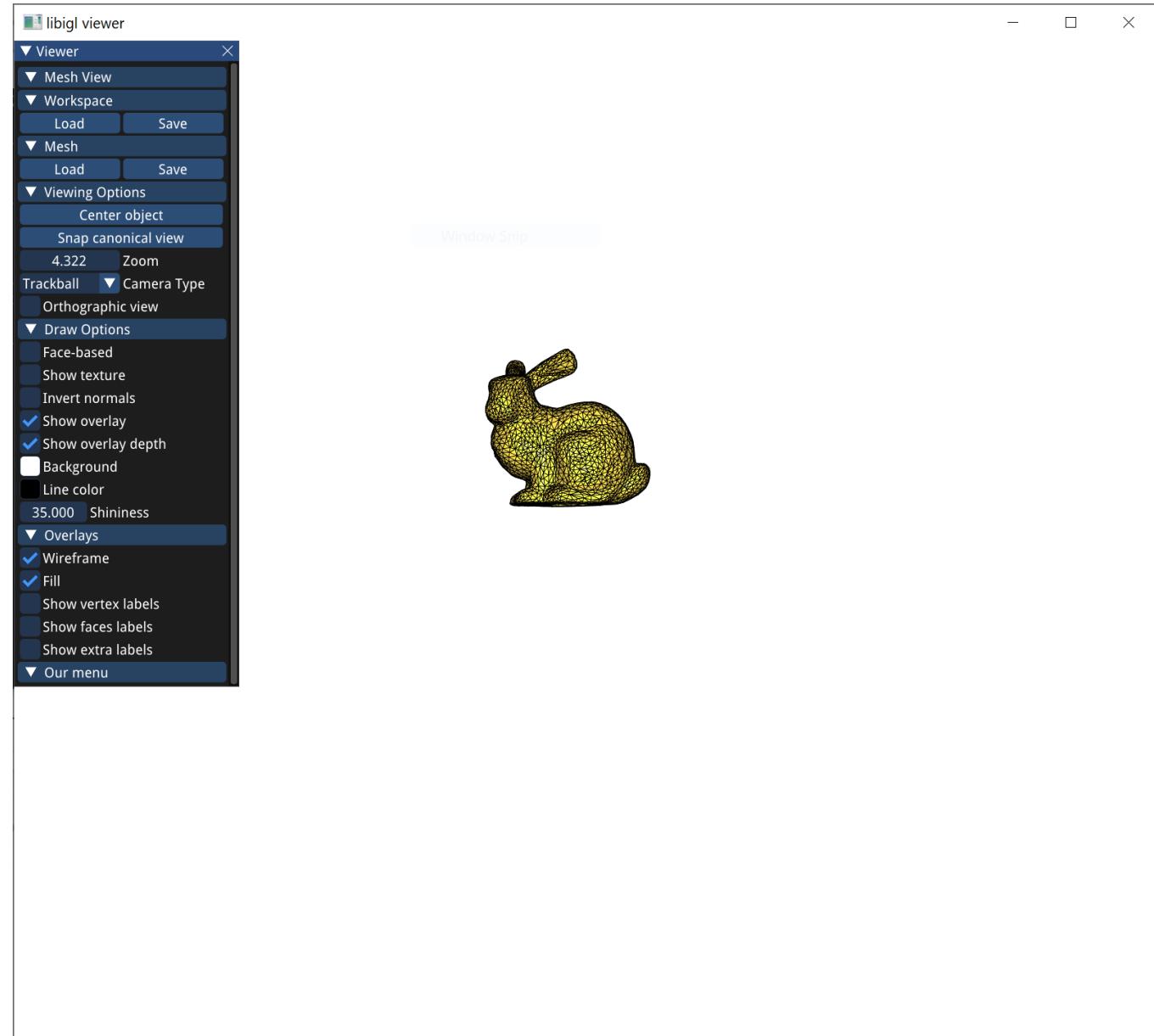
```
# cube.obj
v 0 0 0
v 0 0 1
v 0 1 0
v 0 1 1
v 1 0 0
v 1 0 1
v 1 1 0
v 1 1 1
f 1 7 5
f 1 3 7
f 1 4 3
f 1 2 4
f 3 8 7
f 3 4 8
f 5 7 8
f 5 8 6
f 1 5 6
f 1 6 2
f 2 6 8
f 2 8 4
```

Jacobson & Panozzo, "[libigl course](#)"

Let's go!

Run **0_compilation_test**

Solution is in **0_compilation_test_final**



Quick refs

libigl: great tutorials, courses, etc

<https://libigl.github.io/tutorial>

<https://libigl.github.io/tutorial/#other-matlab-style-functions>

API: none really, but search in [github](#) (one-function one-file principle)

→ IGL, main contributors Alec Jacobson & Danielle Panozzo 

Eigen: https://eigen.tuxfamily.org/dox/group__QuickRefPage.html

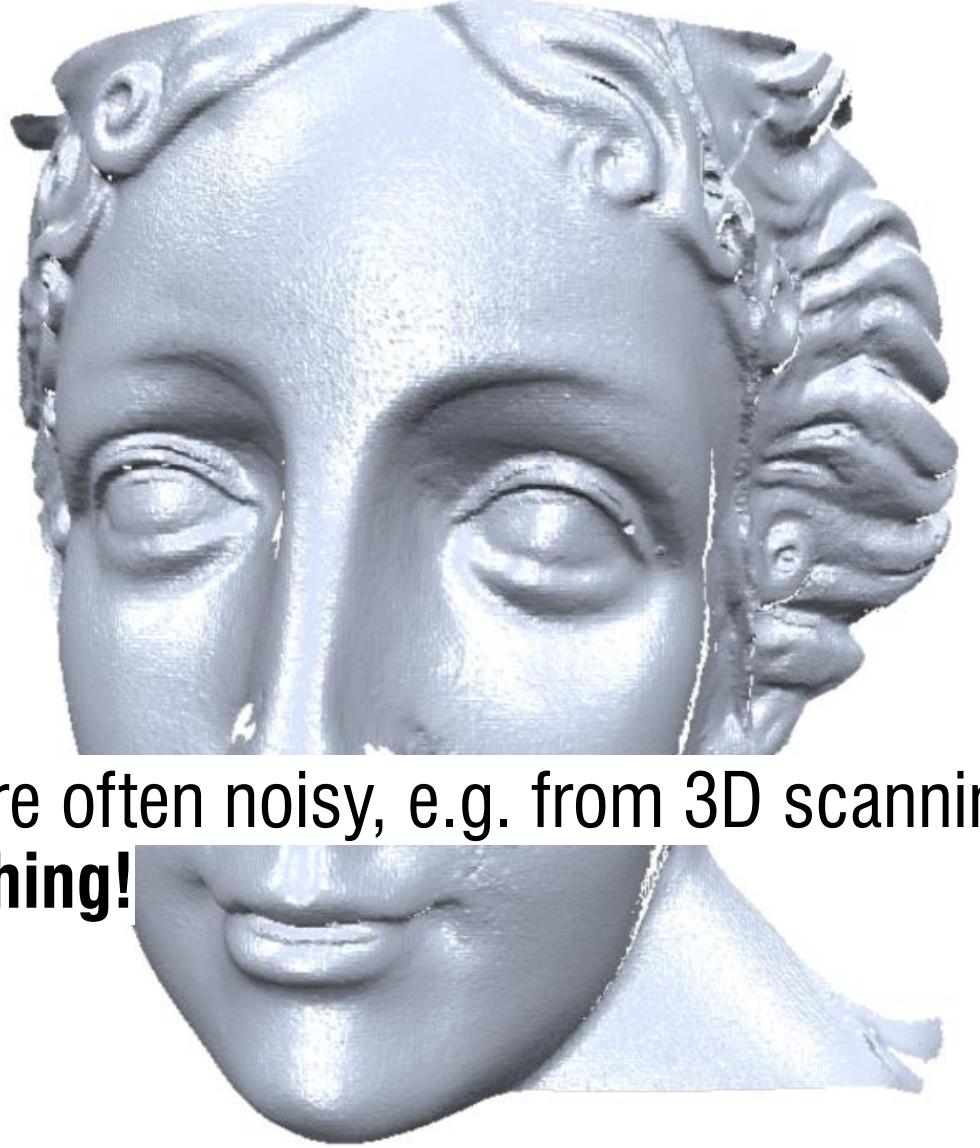
A photograph of two hands against a solid pink background. The hand on the left is gripping a yellow pencil with its fingers spread, showing the lead tip. The hand on the right is gripping a yellow pencil more tightly, with the fingers curled around it. The lighting highlights the skin texture and the yellow color of the pencils.

10min **Break**

1.1 Introduction to libigl

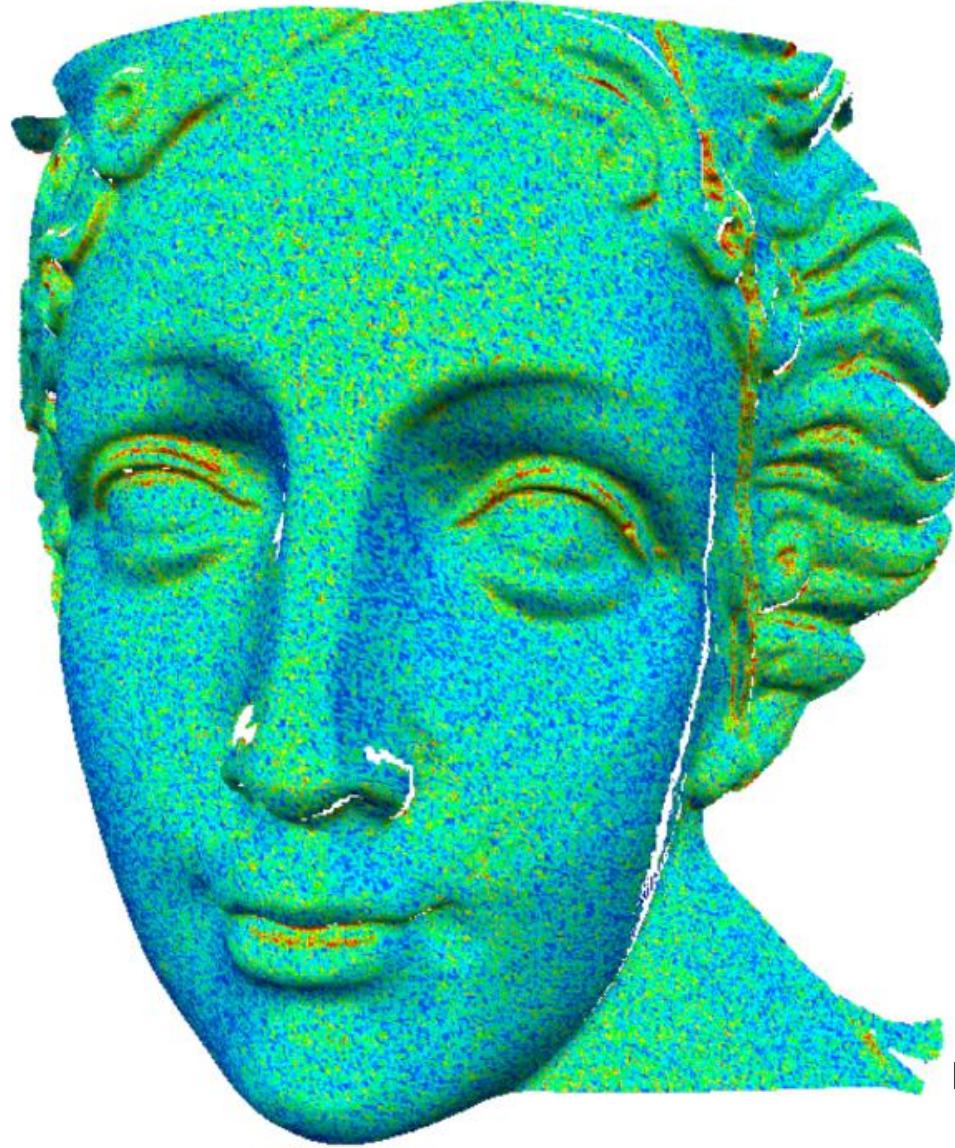
1.2 DDG curves

1.3 Smoothing meshes

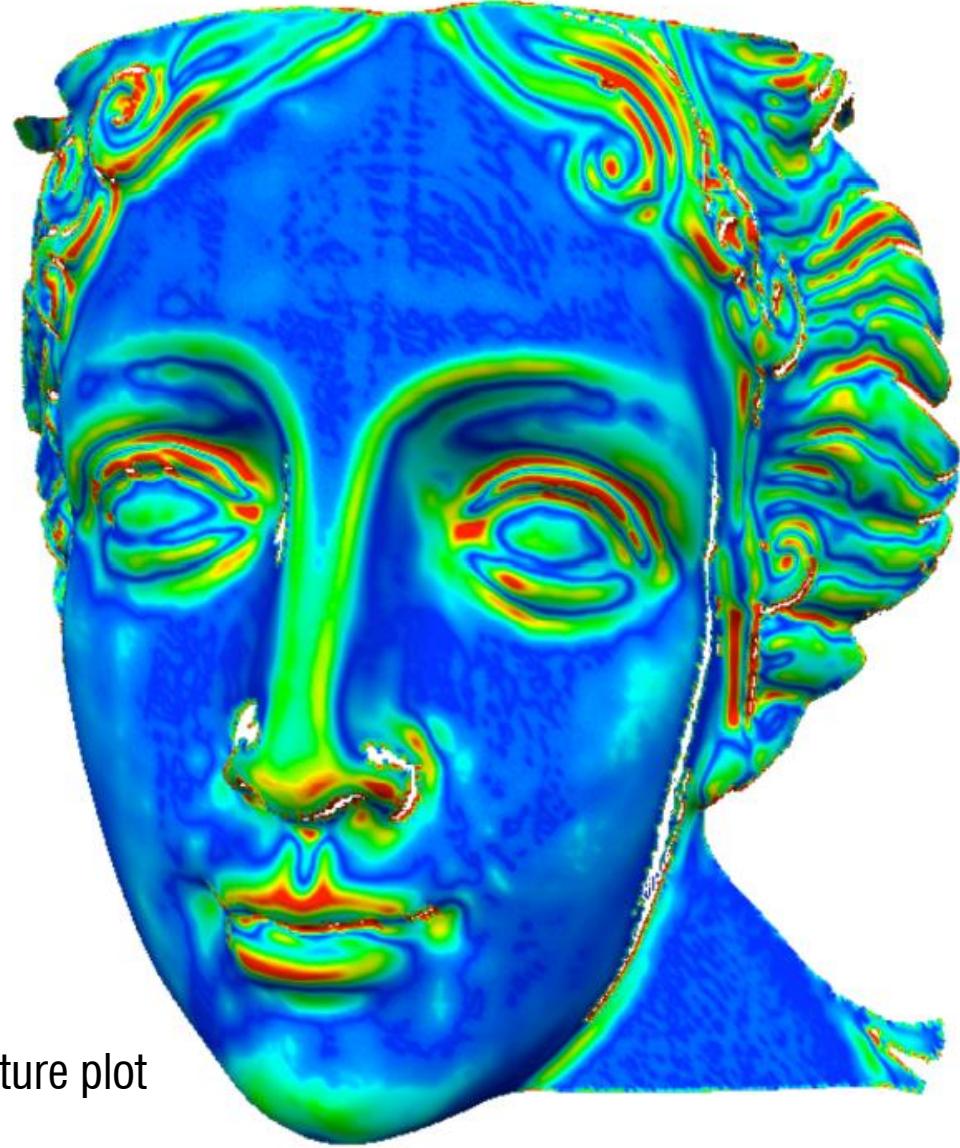


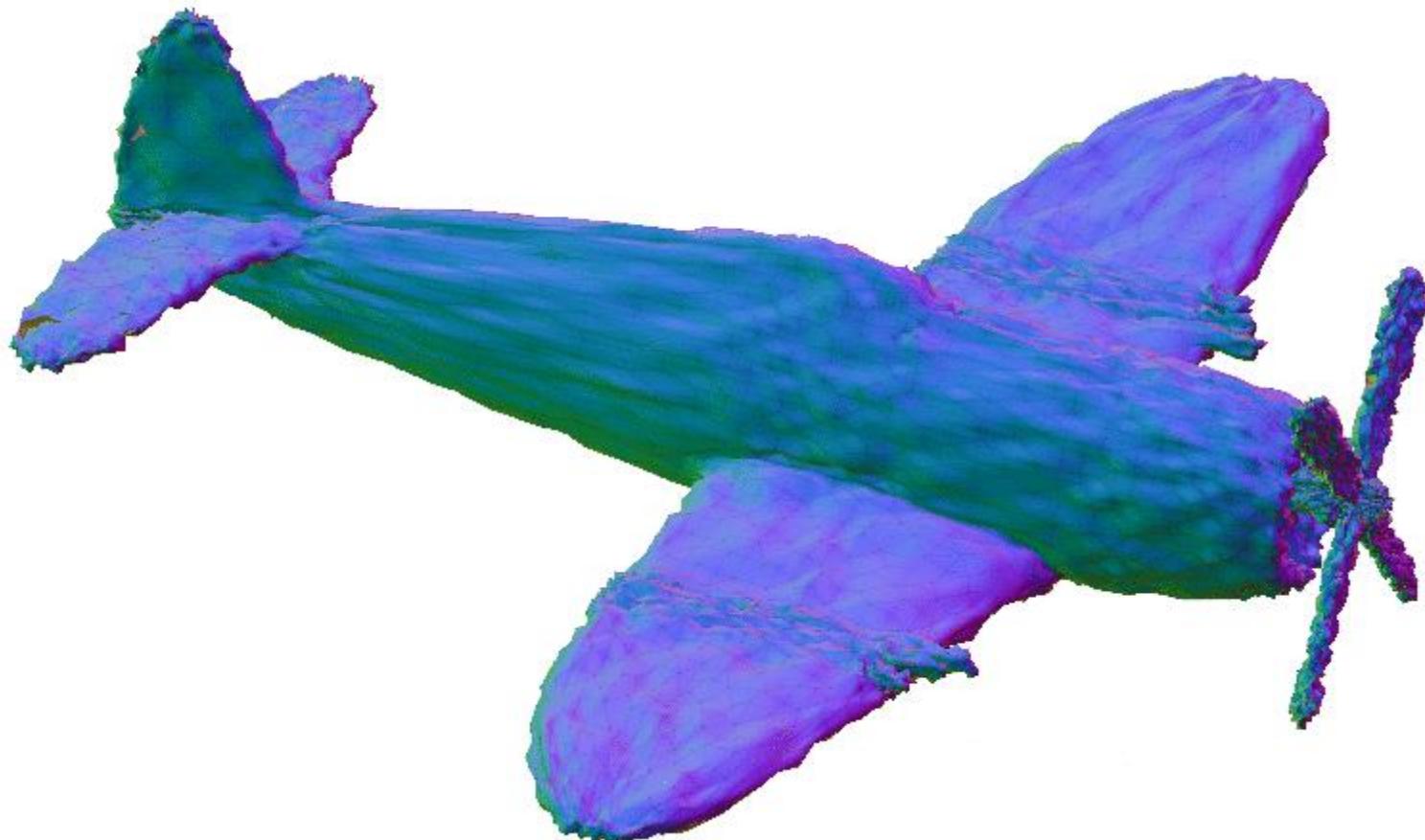
meshes are often noisy, e.g. from 3D scanning
→ **smoothing!**





mean curvature plot





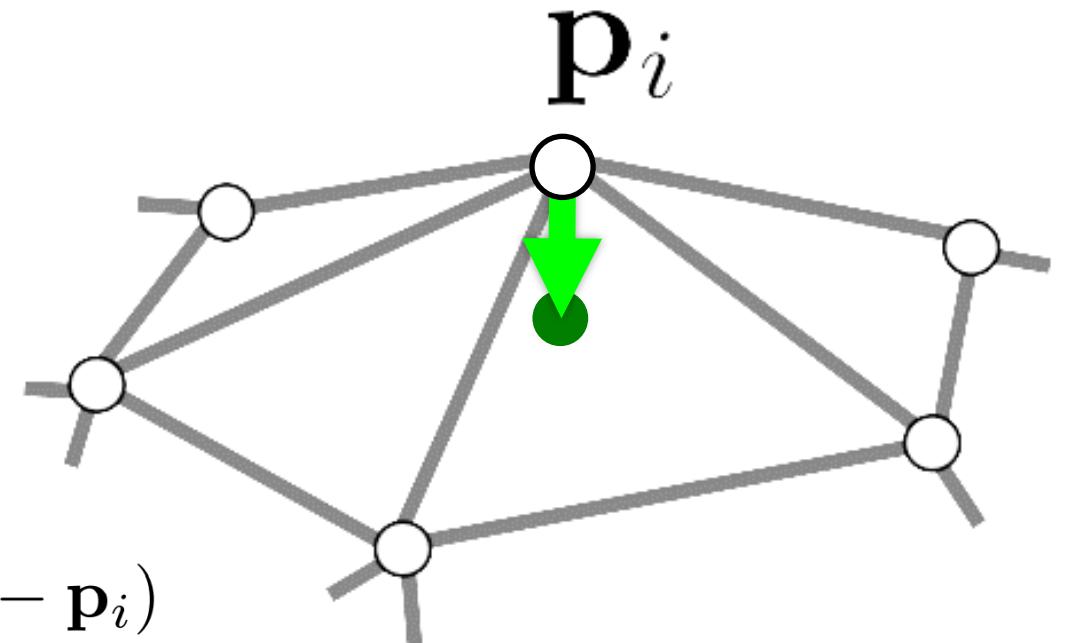
Alec Jacobson, "[Geometry Processing Smoothing](#)"

Mesh smoothing

High-pass filter: extracts local surface detail

Detail = ***smooth***(surface) – surface

Assumption: smoothing = averaging

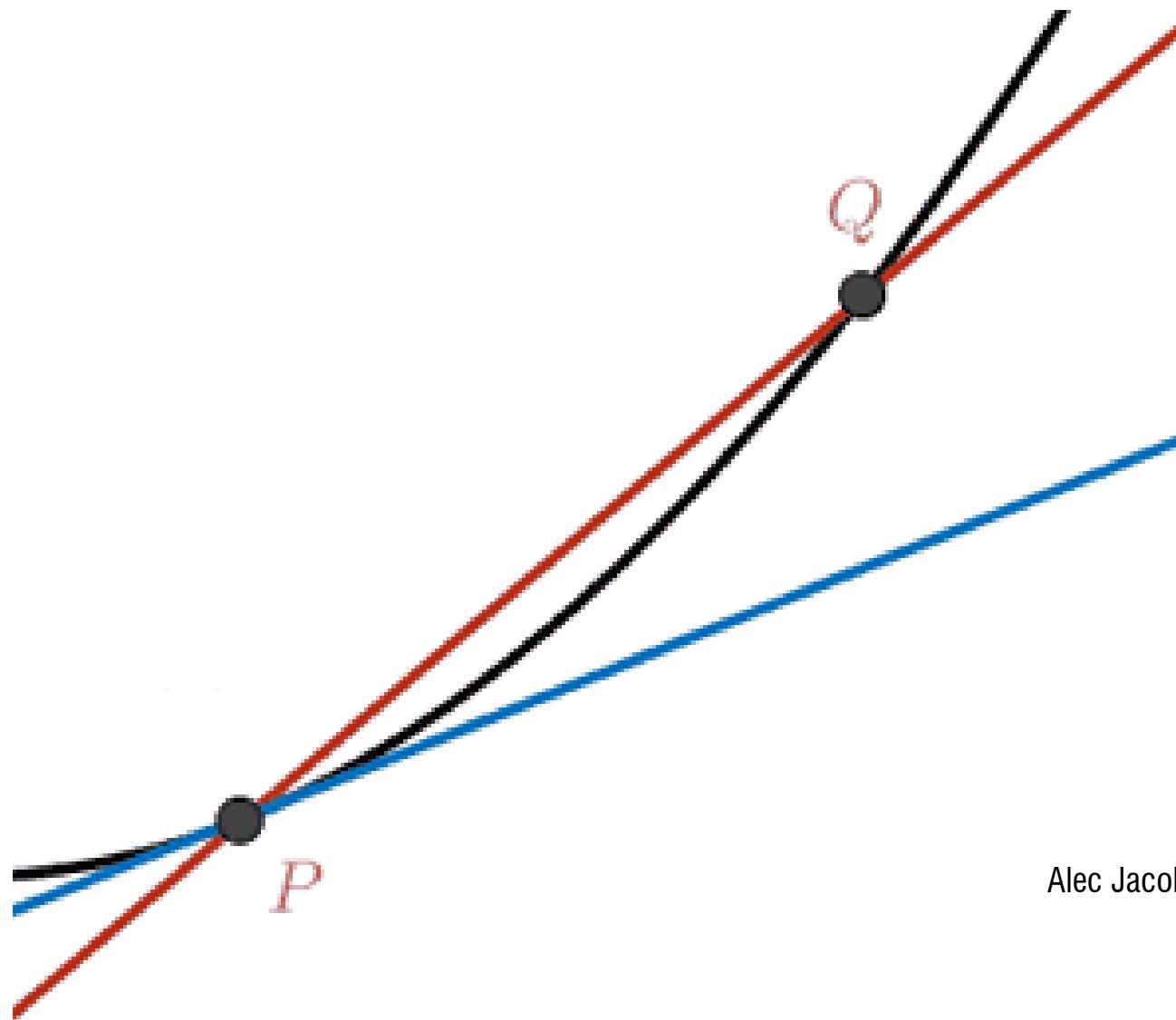


$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}_j - \mathbf{p}_i)$$

Olga Sorkine-Hornung, ETH, Shape Modeling course

Let's start with curves

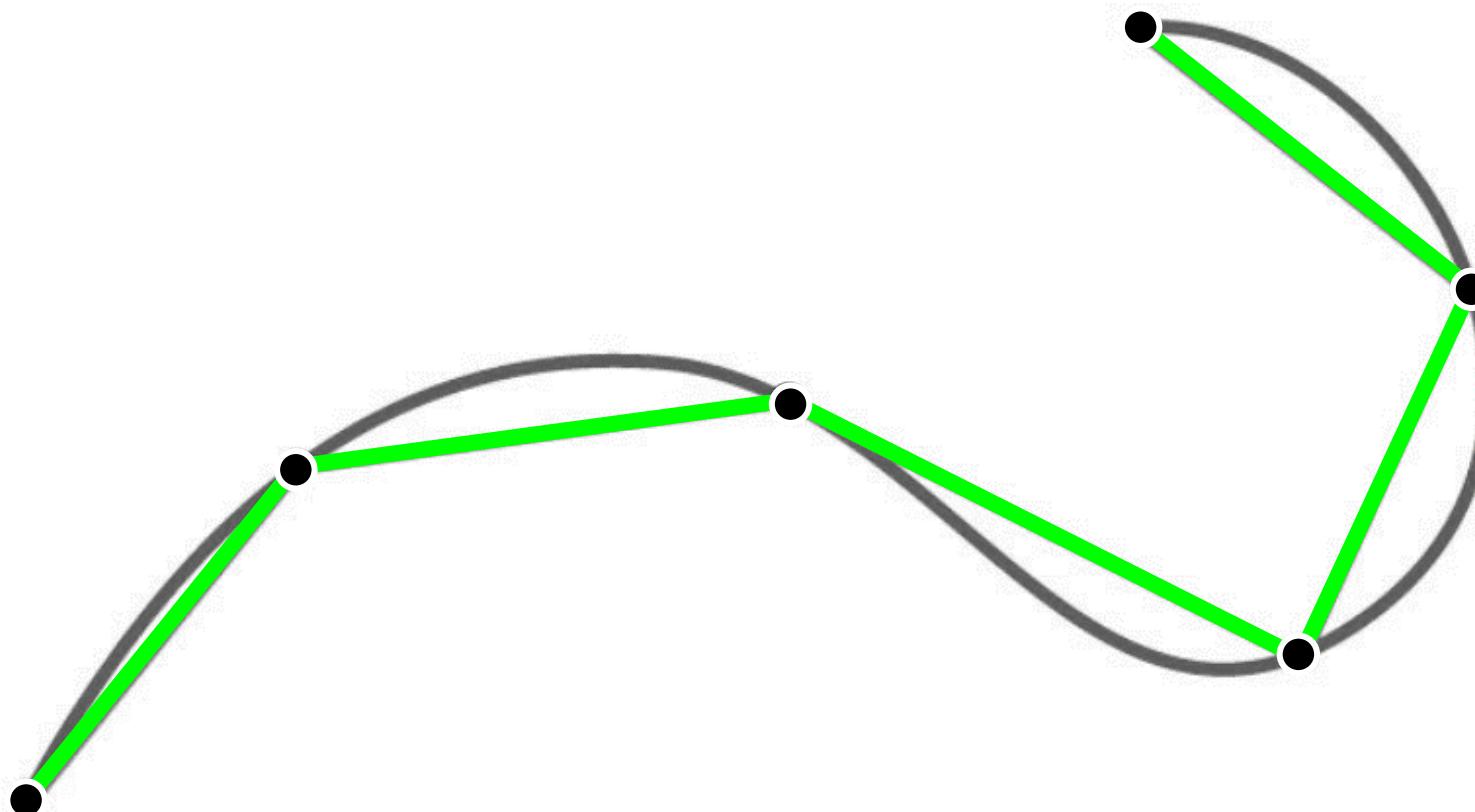
Differential Geometry



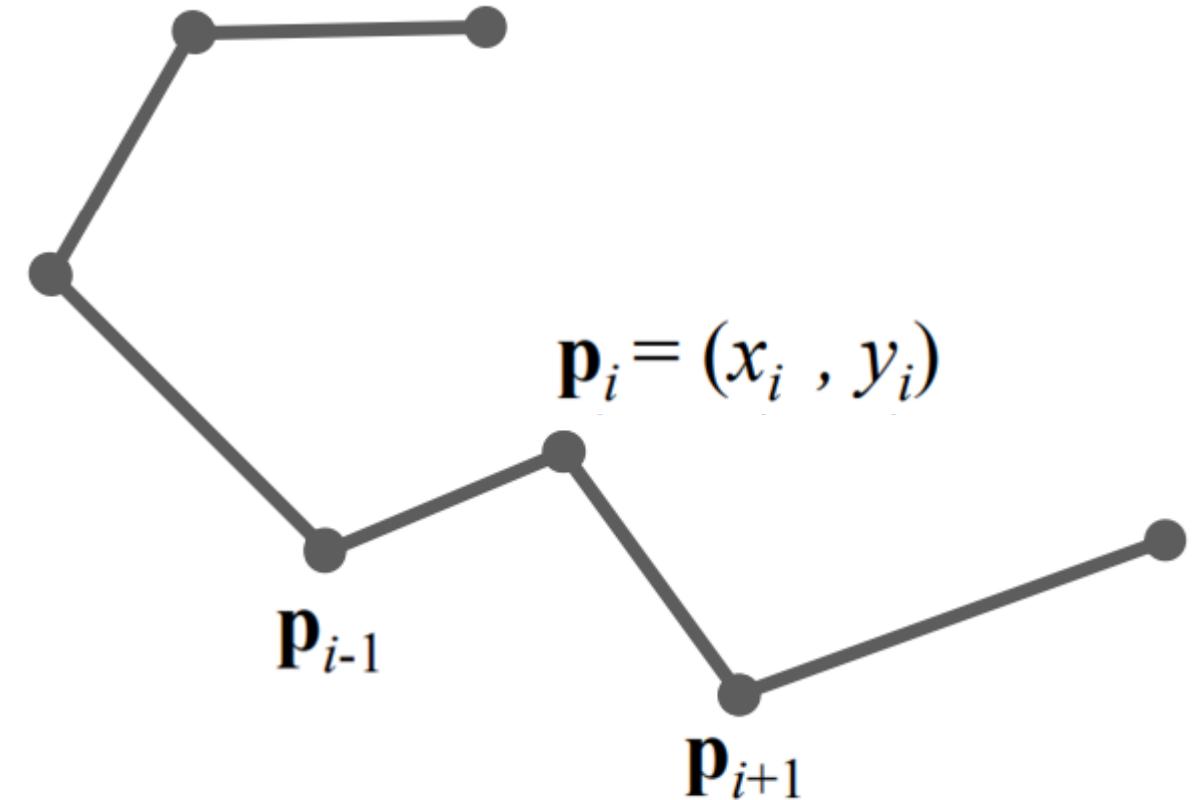
Alec Jacobson, "[Geometry Processing - Curvature](#)"

Discrete Differential Geometry

Piecewise-linear curve: finite number of vertices on the curve connected by lines

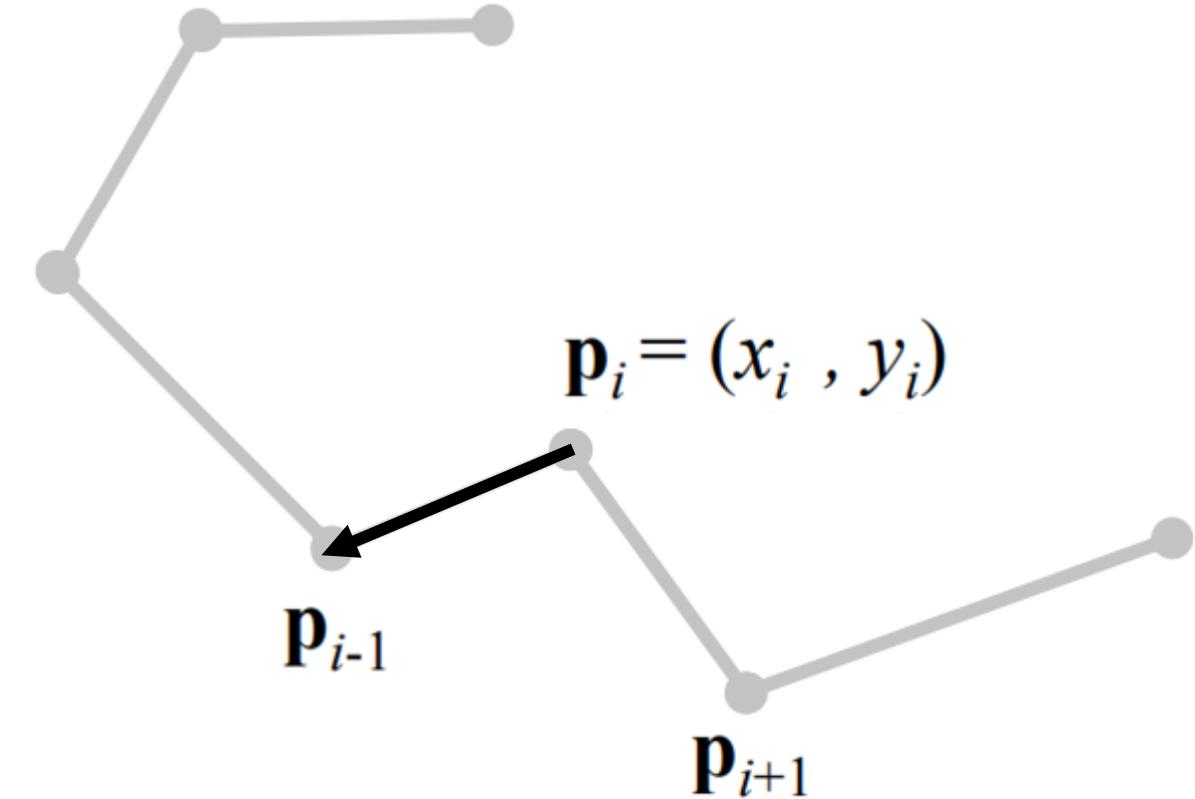


Laplacian smoothing
on a planar discrete curve



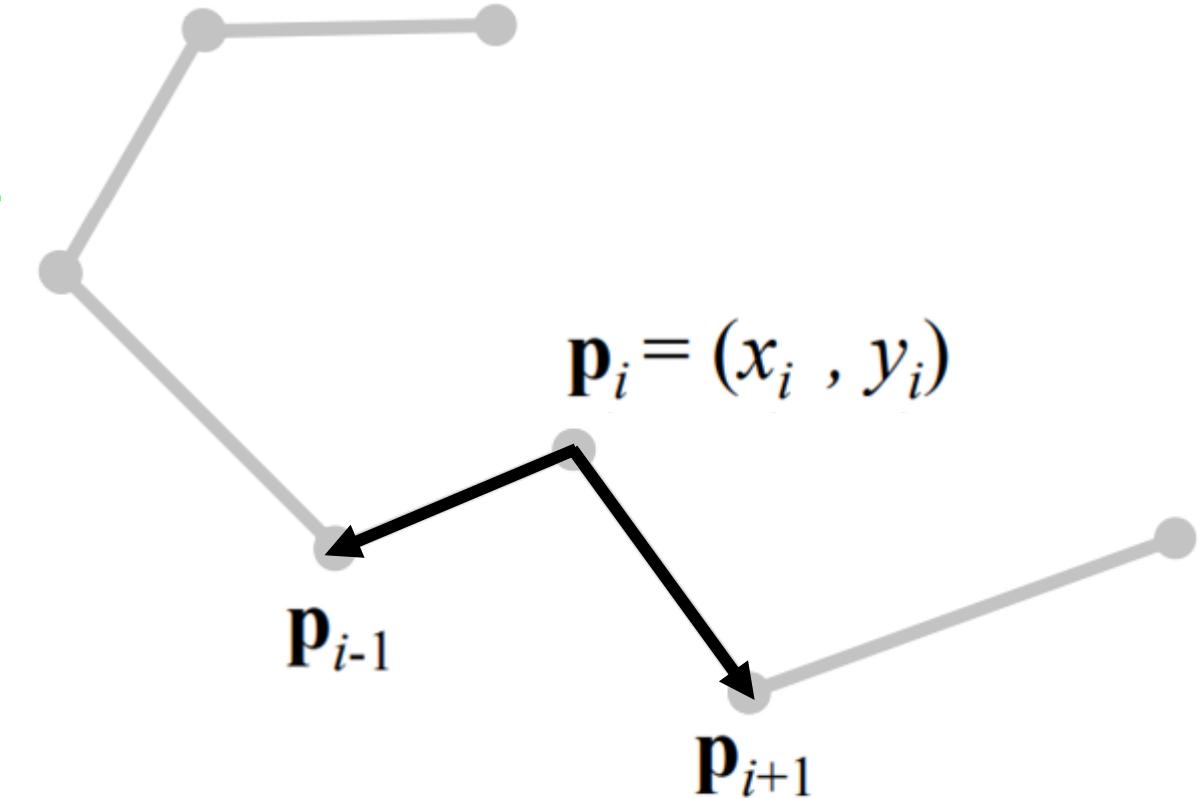
Laplacian smoothing

$$L(\mathbf{p}_i) = \frac{1}{2} (\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2} (\mathbf{p}_{i+1} - \mathbf{p}_i)$$



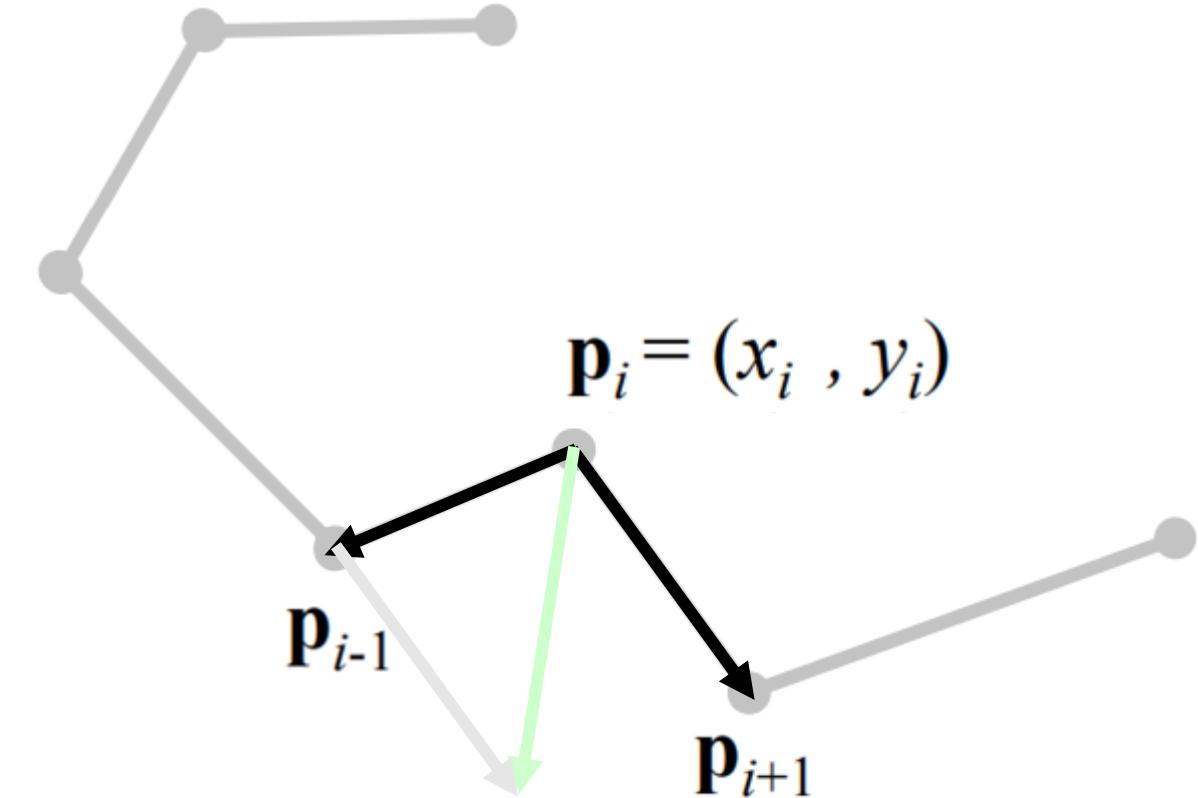
Laplacian smoothing

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$



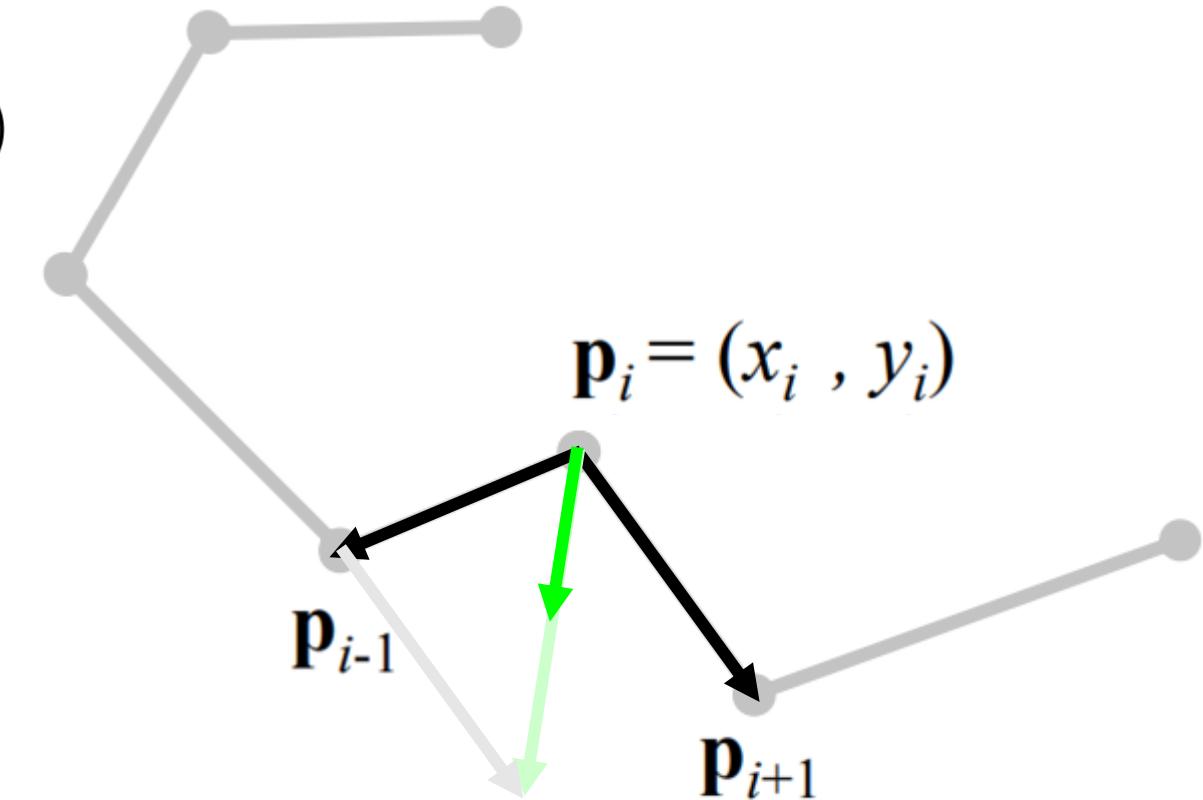
Laplacian smoothing

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$



Laplacian smoothing

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$

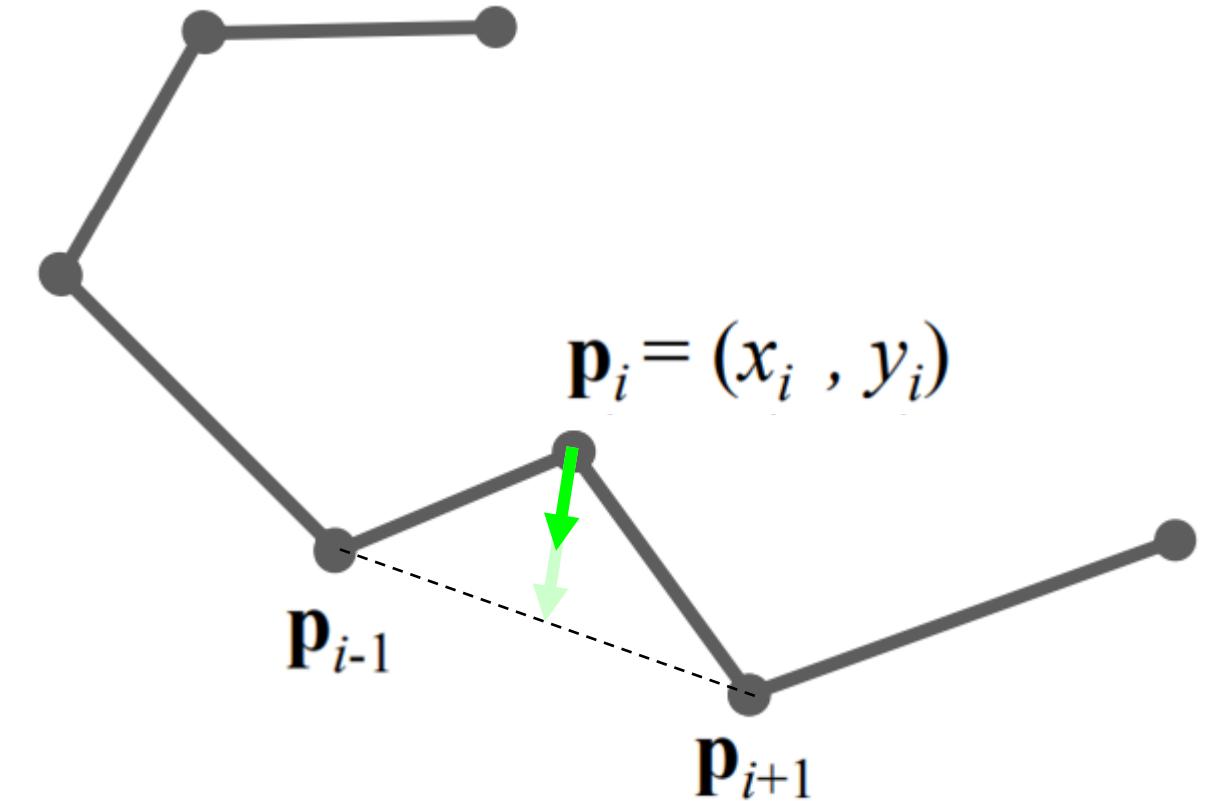


Laplacian smoothing

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$

$$\tilde{\mathbf{p}} = \mathbf{p} + \underline{\lambda L \mathbf{p}}$$

move the vertex **stepwise** towards their smooth position

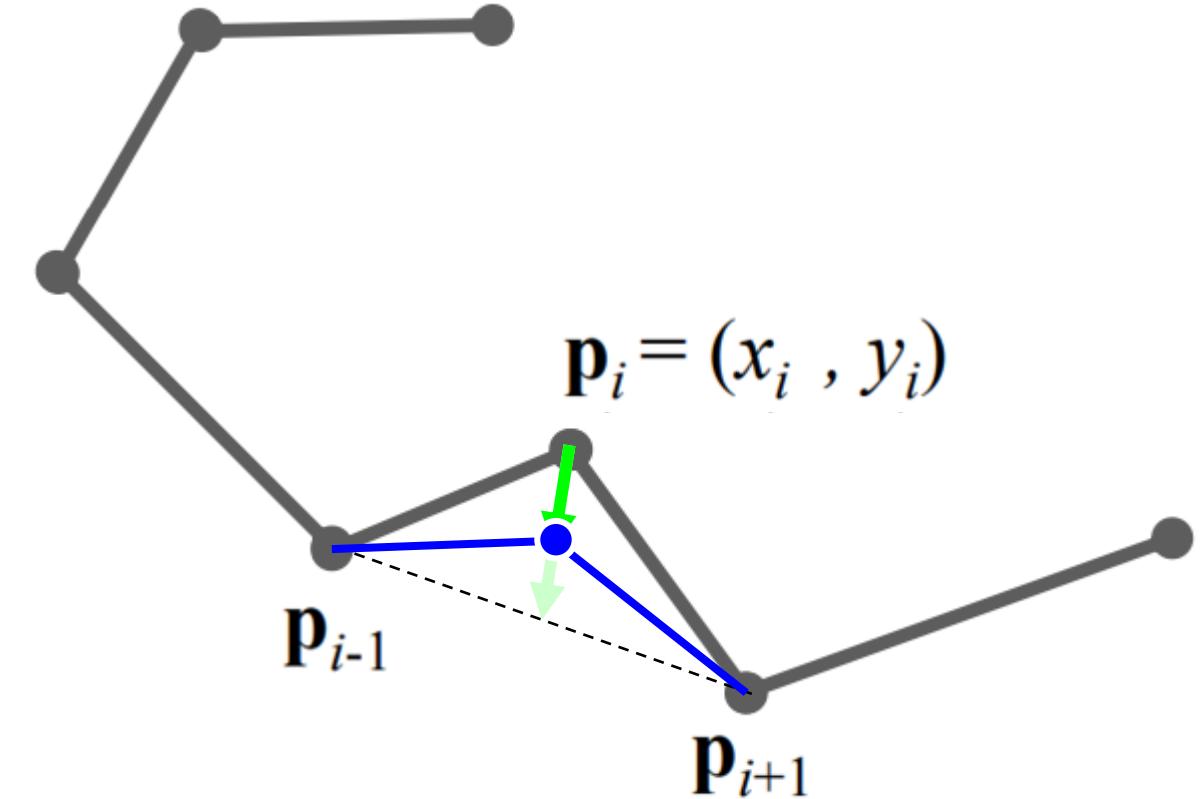


Laplacian smoothing

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$

$$\tilde{\mathbf{p}} = \mathbf{p} + \underline{\lambda L \mathbf{p}}$$

move the vertex **stepwise** towards their smooth position



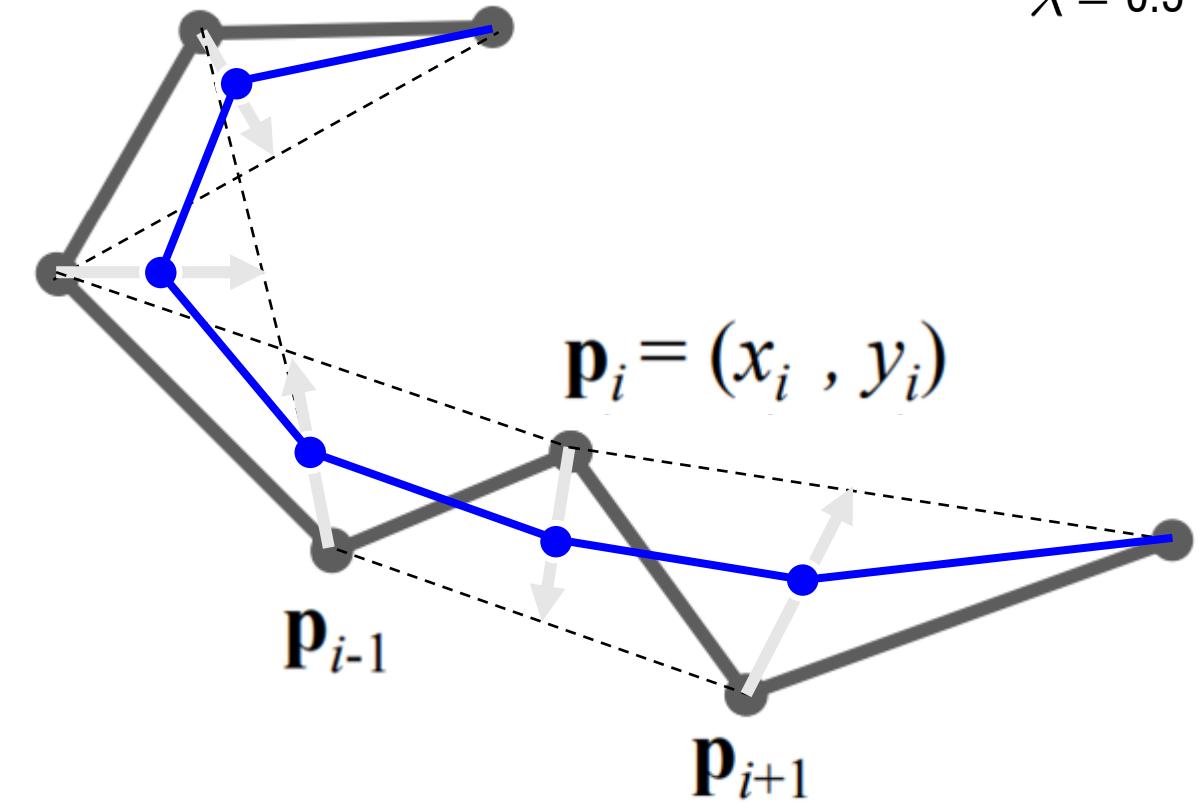
Laplacian smoothing

$\lambda = 0.5$

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$

$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L\mathbf{p}$$

move the vertex **stepwise** towards their smooth position



Any drawbacks?

<30sec brainstorming>

Let's see...

run 1_mesh_smoothing
[key c]

Taubin smoothing

$\lambda = 0.5$

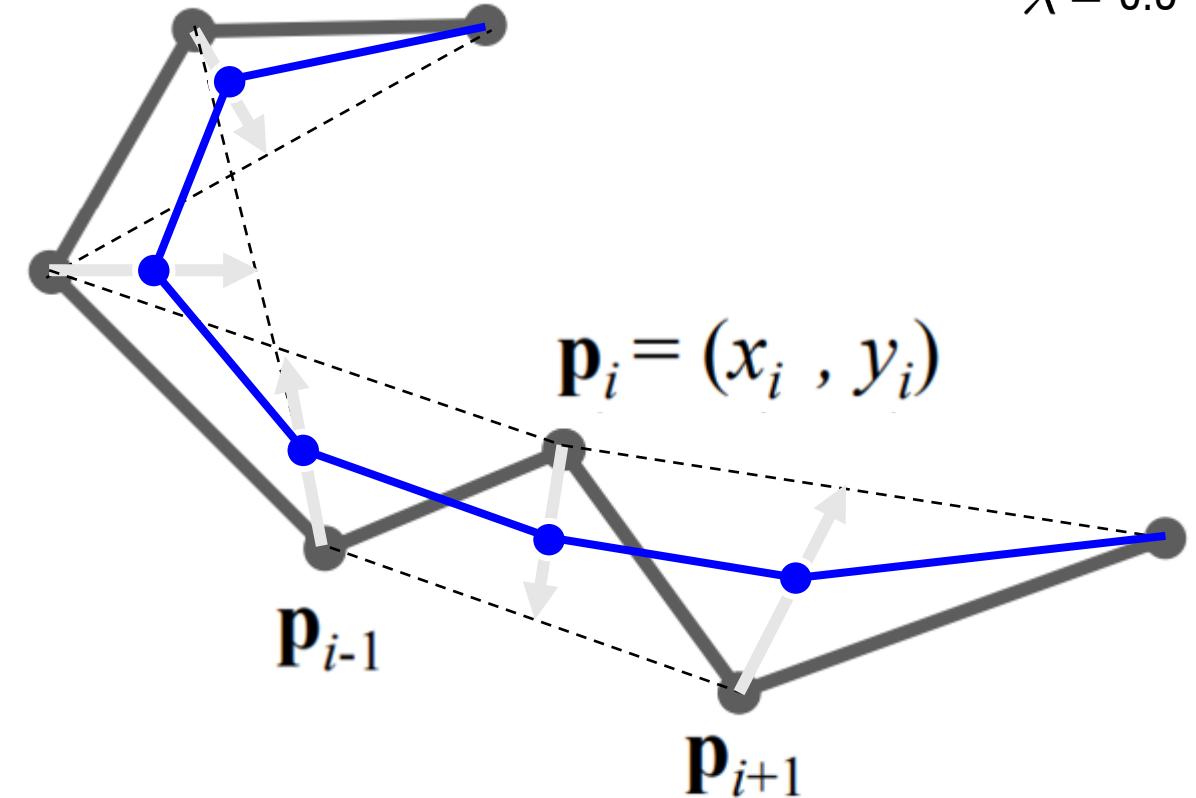
$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$

$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L\mathbf{p}$$

$$\tilde{\mathbf{p}} = \mathbf{p} + \mu L\mathbf{p}$$

$\lambda > 0$ to smooth

$\mu < 0$ to **inflate**



Taubin smoothing

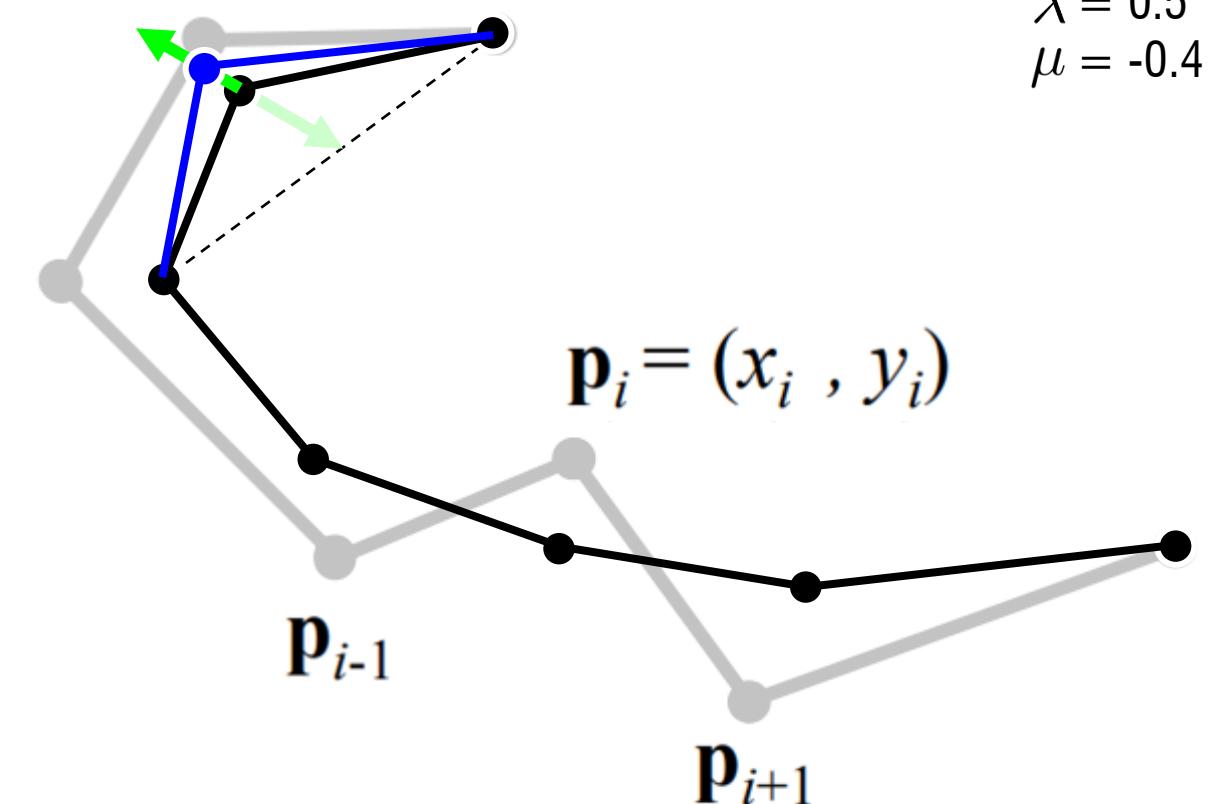
$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$

$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L\mathbf{p}$$

$$\tilde{\mathbf{p}} = \mathbf{p} + \mu L\mathbf{p}$$

$\lambda > 0$ to smooth

$\mu < 0$ to **inflate**



Taubin smoothing

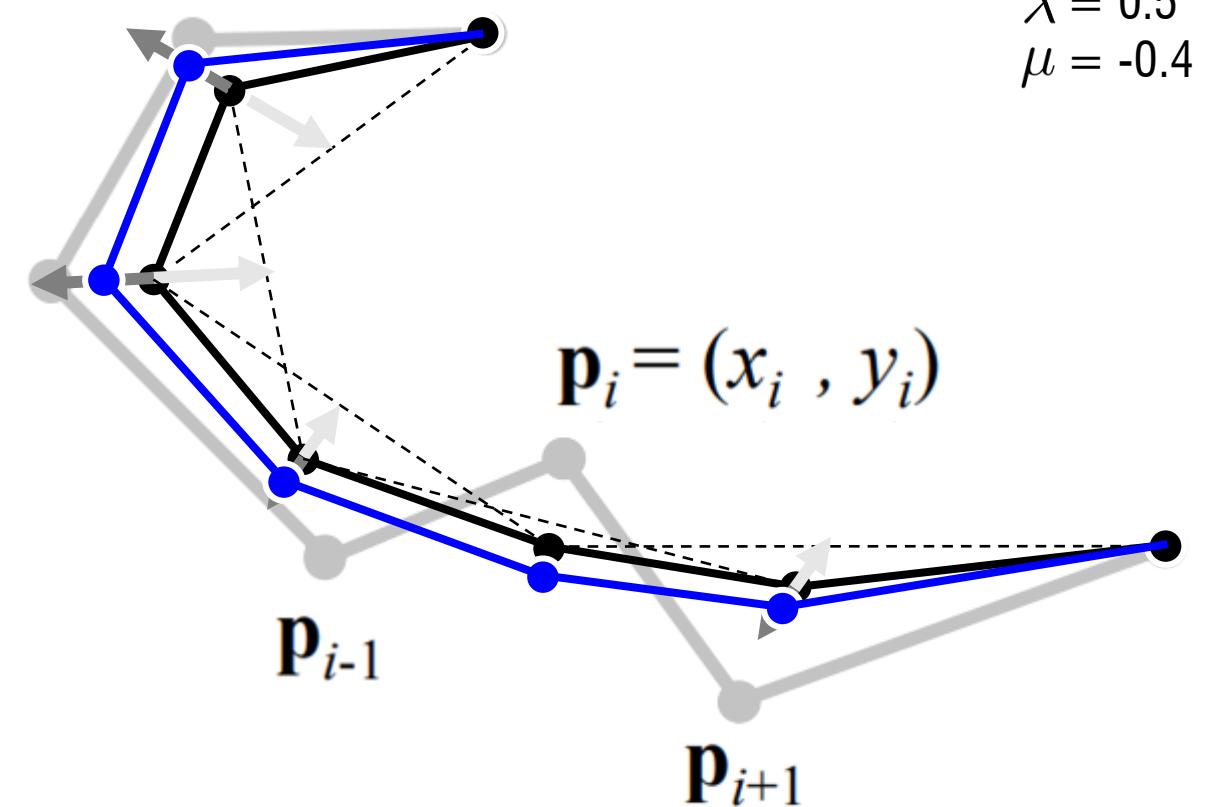
$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$

$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L\mathbf{p}$$

$$\tilde{\mathbf{p}} = \mathbf{p} + \mu L\mathbf{p}$$

$\lambda > 0$ to smooth

$\mu < 0$ to **inflate**



1.1 Introduction to libigl

1.2 DDG curves

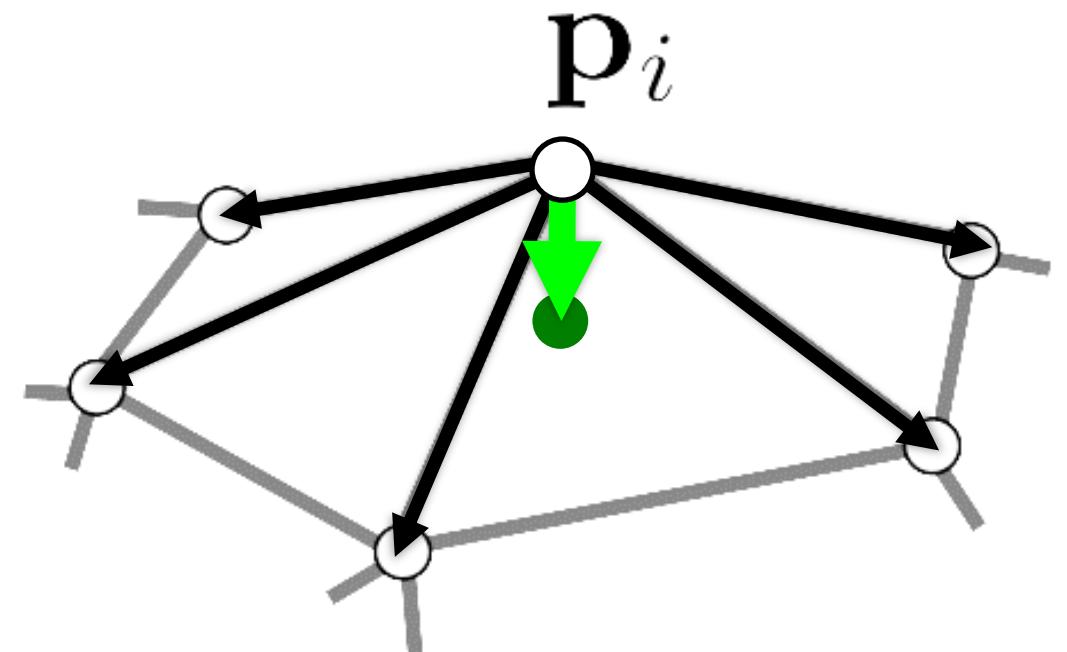
1.3 Smoothing meshes

Same idea as for curve smoothing

Assumption: smoothing = averaging

Uniform Laplacian or **Umbrella** operator

$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}_j - \mathbf{p}_i)$$



Weighing schemes

Ignoring the geometry:

δ_{uniform} : $W_i = 1$, $w_{ij} = 1/|N(i)|$

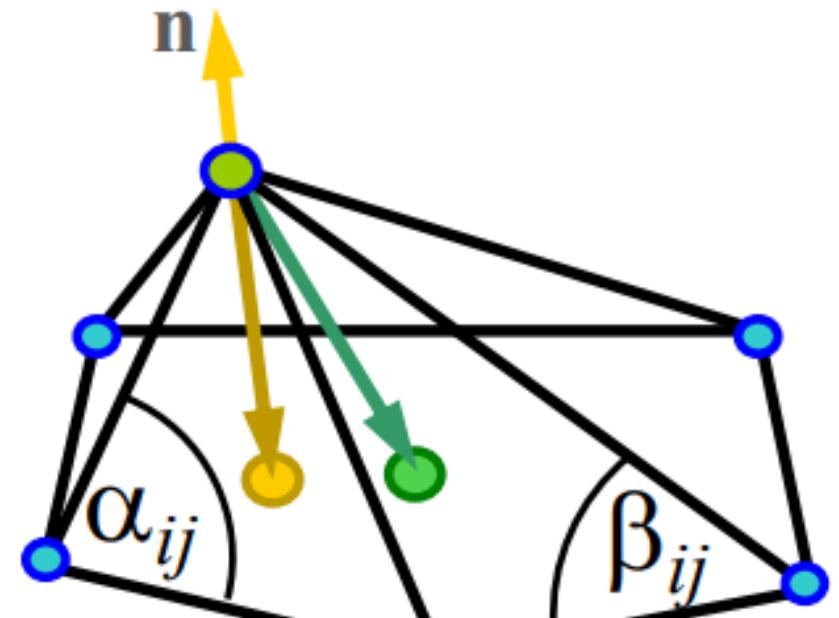
Integrate over Voronoi region of the vertex:

δ_{cotan} : $w_{ij} = 0.5(\cot \alpha_{ij} + \cot \beta_{ij})$

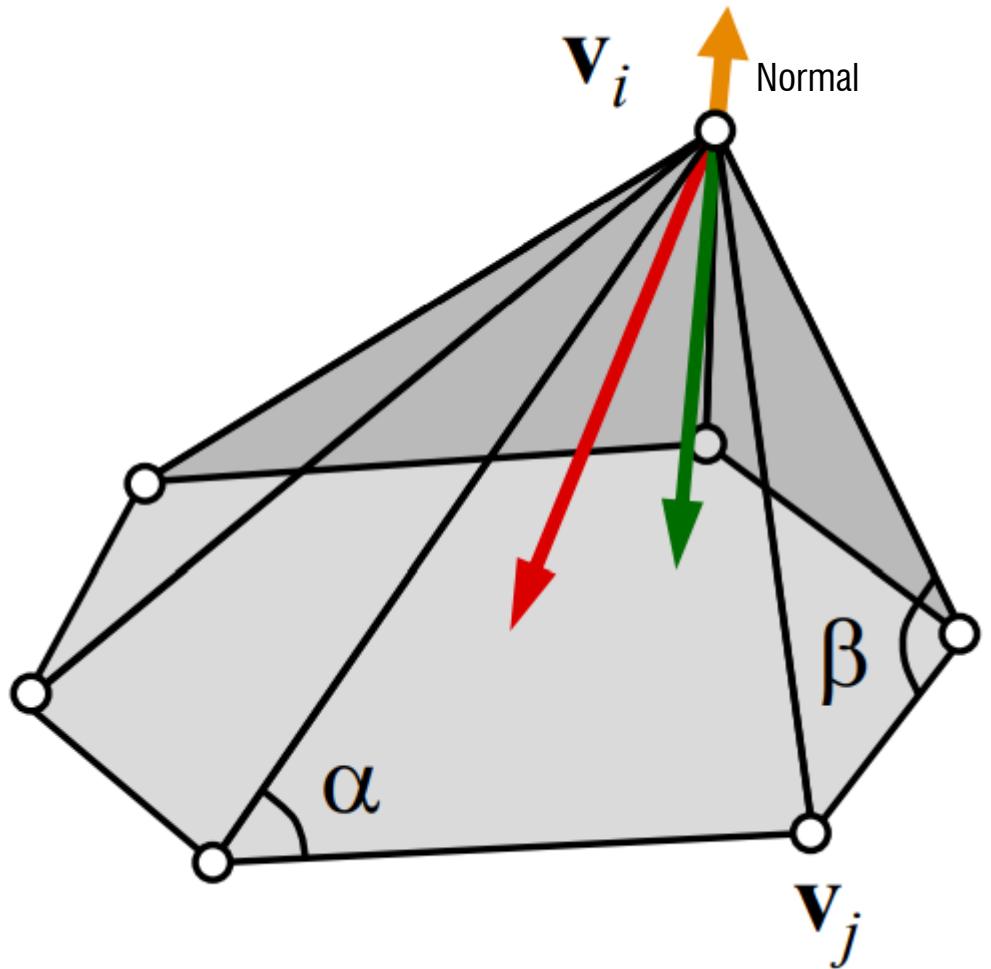


$$W_i = A_i$$

$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}_j - \mathbf{p}_i)$$



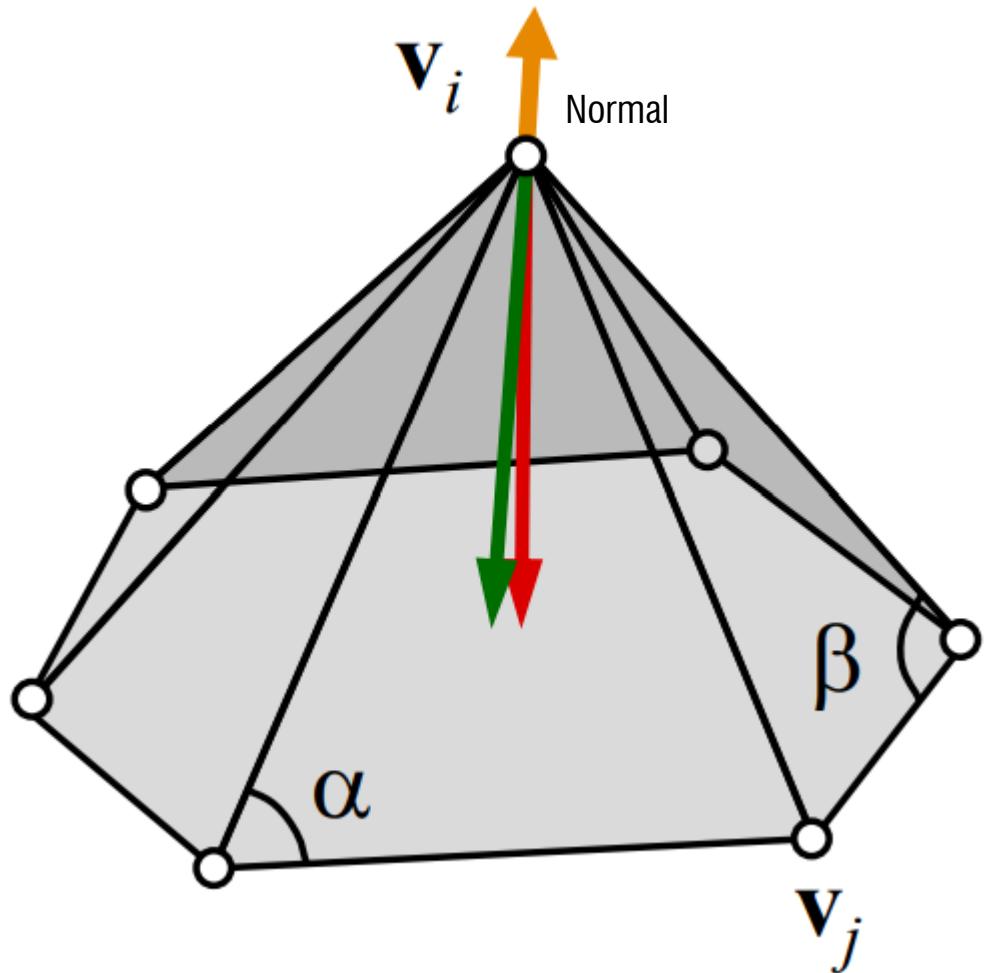
Olga Sorkine-Hornung, ETH, Shape Modeling course



Uniform Laplacian (Umbrella operator)
Cotangent Laplacian (Laplace-Beltrami)

For nearly equal edge lengths
Uniform \approx Cotangent

Olga Sorkine-Hornung, ETH, Shape Modeling course

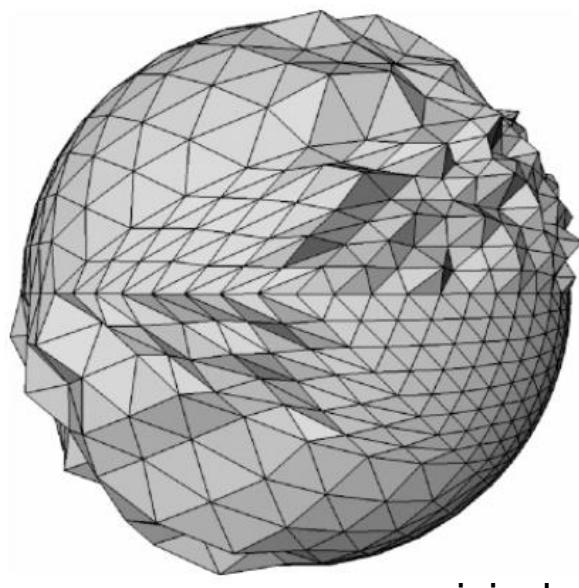
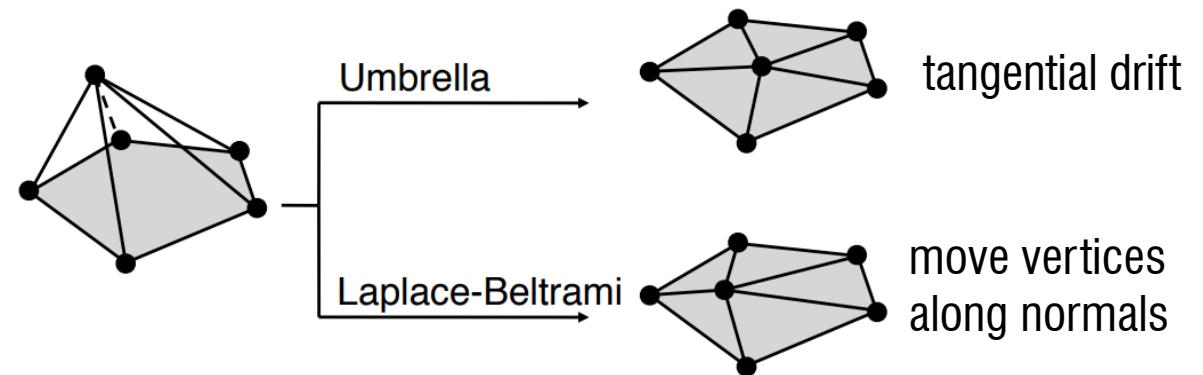


Uniform Laplacian (Umbrella operator)
Cotangent Laplacian (Laplace-Beltrami)

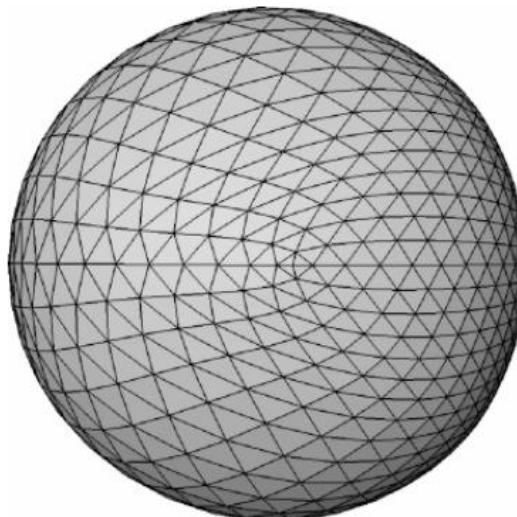
For nearly equal edge lengths
Uniform \approx Cotangent

Olga Sorkine-Hornung, ETH, Shape Modeling course

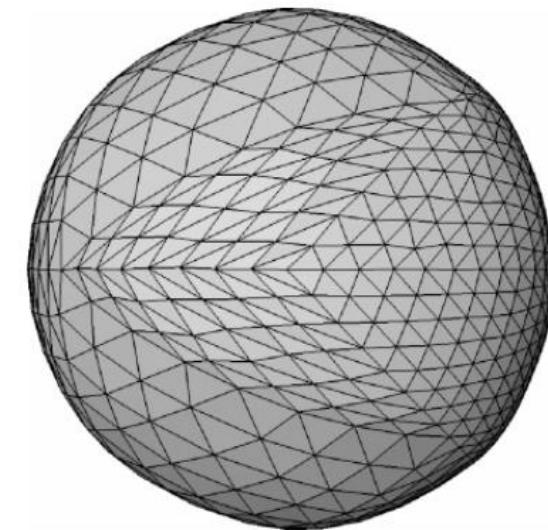
The Uniform Laplacian moves the geometry tangentially as well



original



uniform

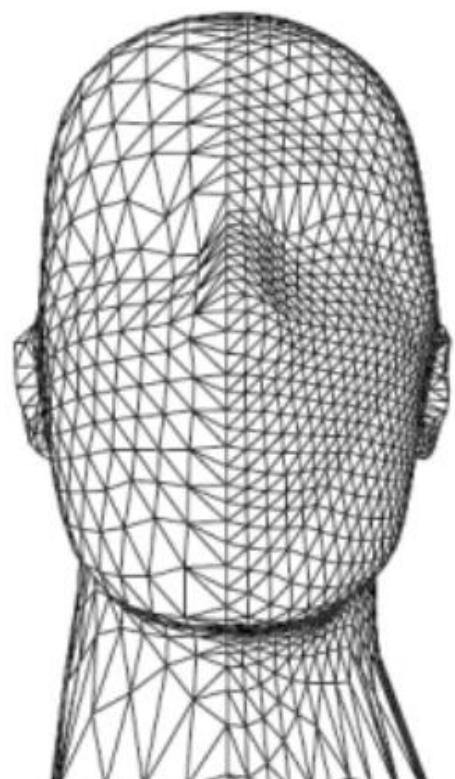


cotan

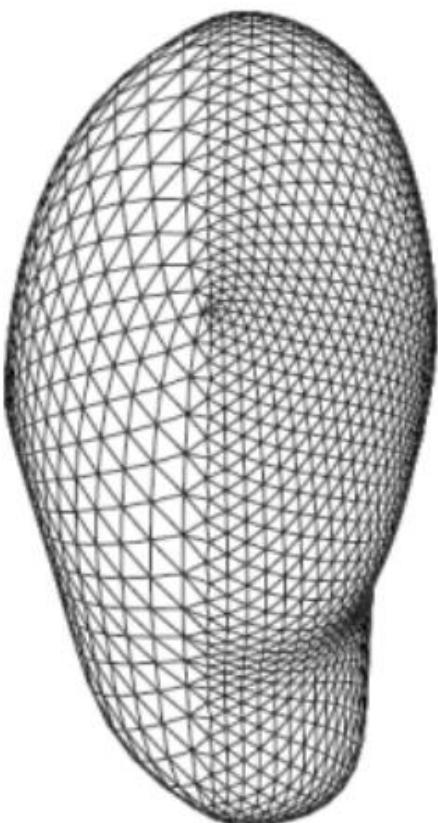
Carnegie Mellon University



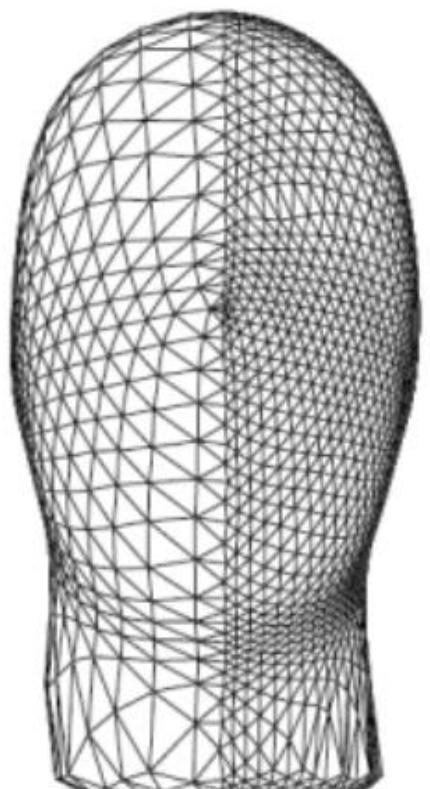
The Uniform Laplacian moves the geometry tangentially as well



original



uniform



cotan



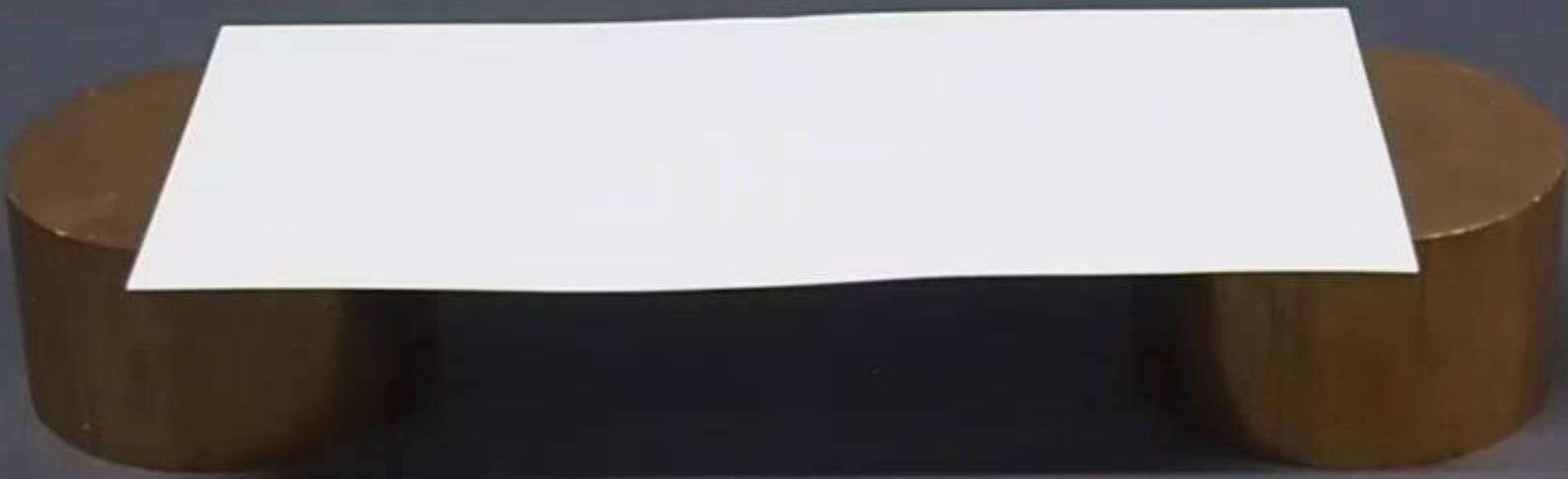
See an example in **1_mesh_smoothing** [key m]

10min **Break**

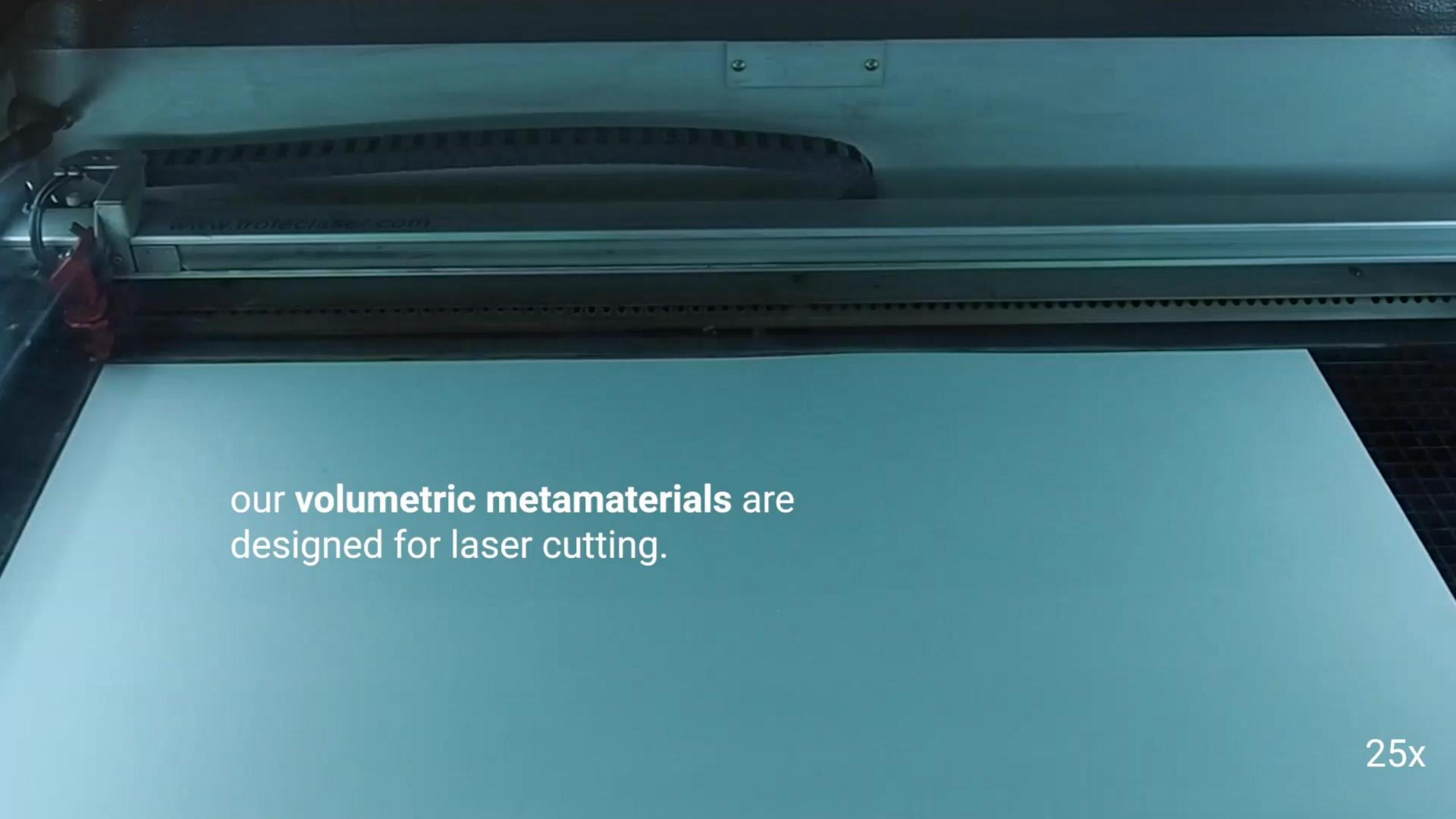


Ruffles

thin sheets, such as paper,
have a **very low bending stiffness**



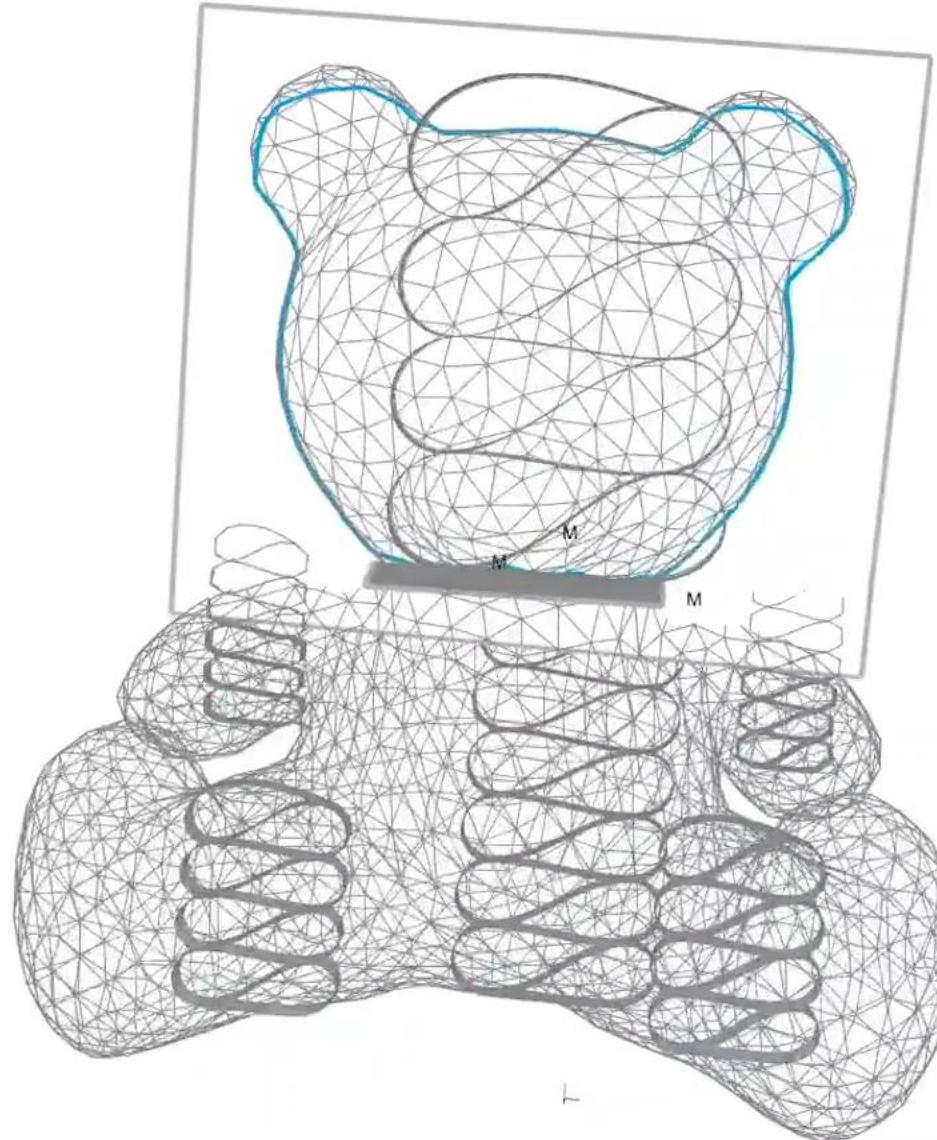


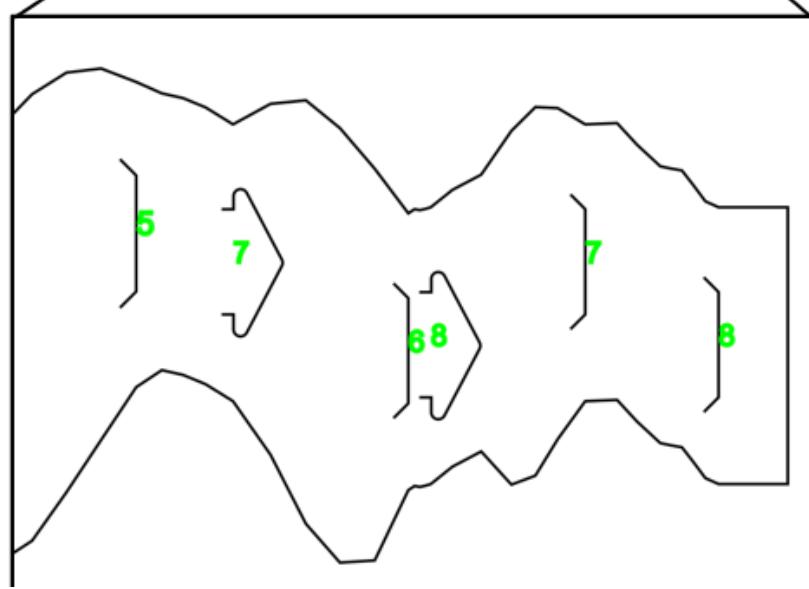
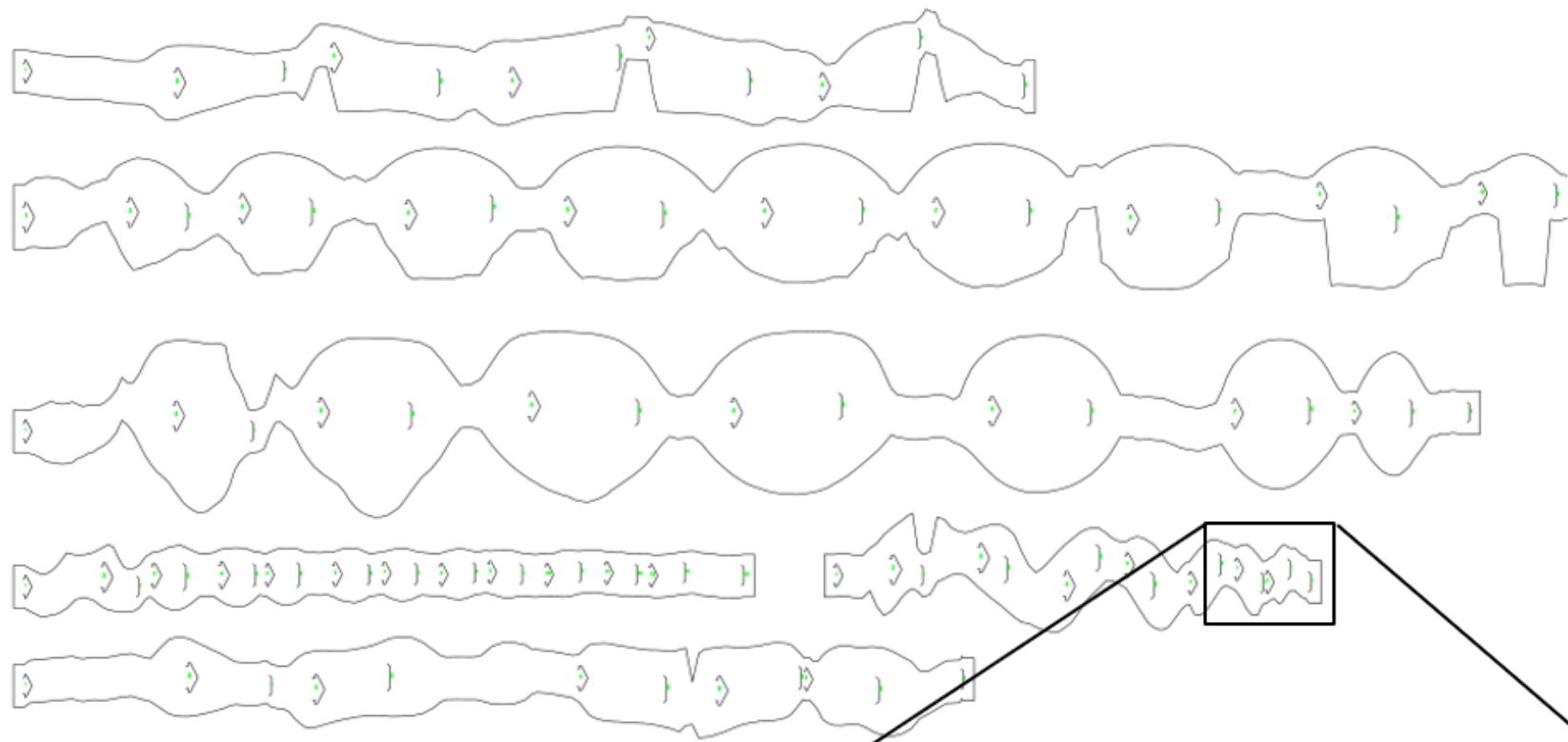
A large industrial laser cutting machine is shown from a low angle, focusing on its internal components. A red safety light is visible on the left side. The machine has a dark, metallic frame with various mechanical parts and cables. A small label on the side of the machine reads "www.holecutter.com".

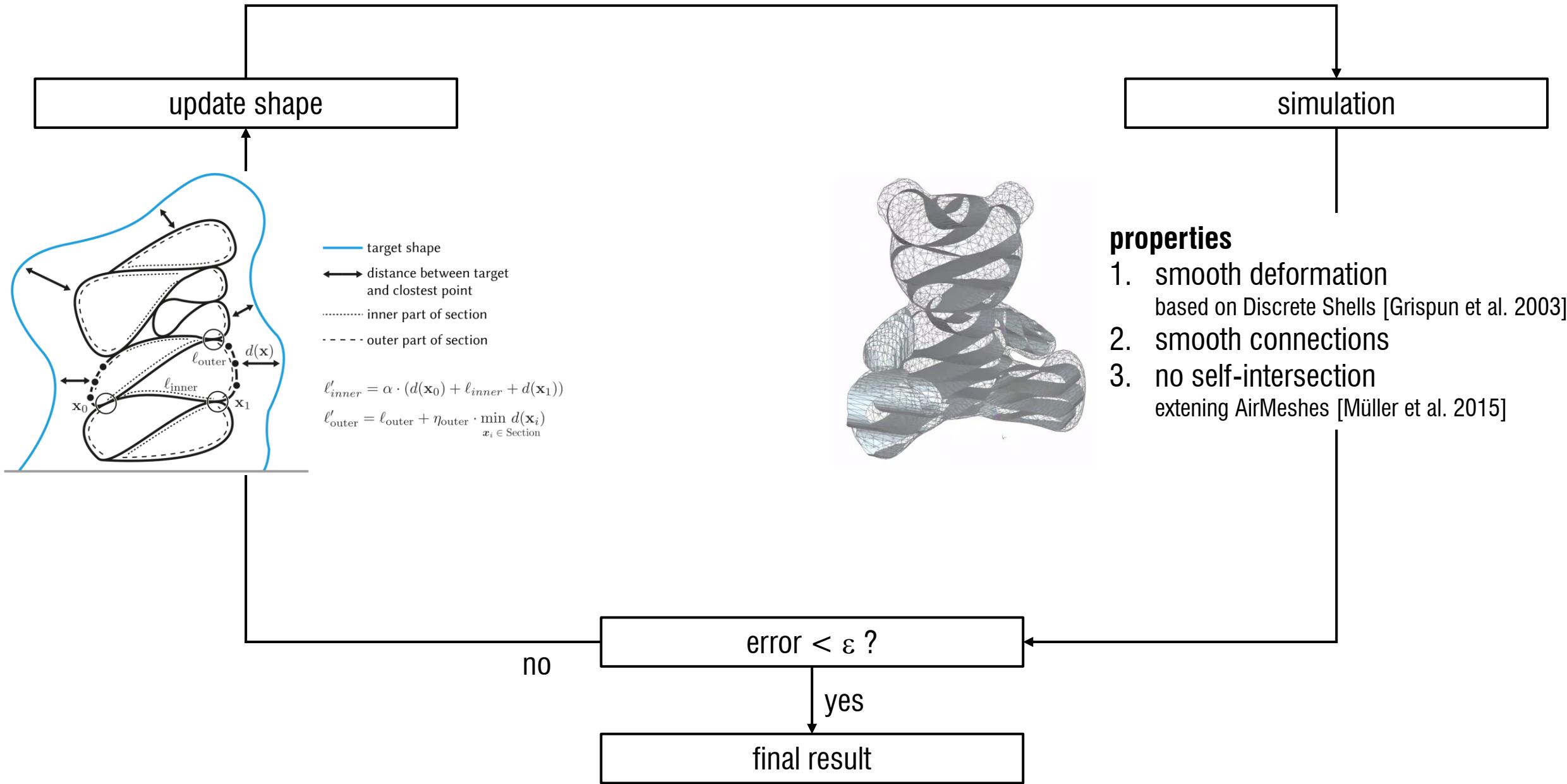
our **volumetric metamaterials** are
designed for laser cutting.

25x

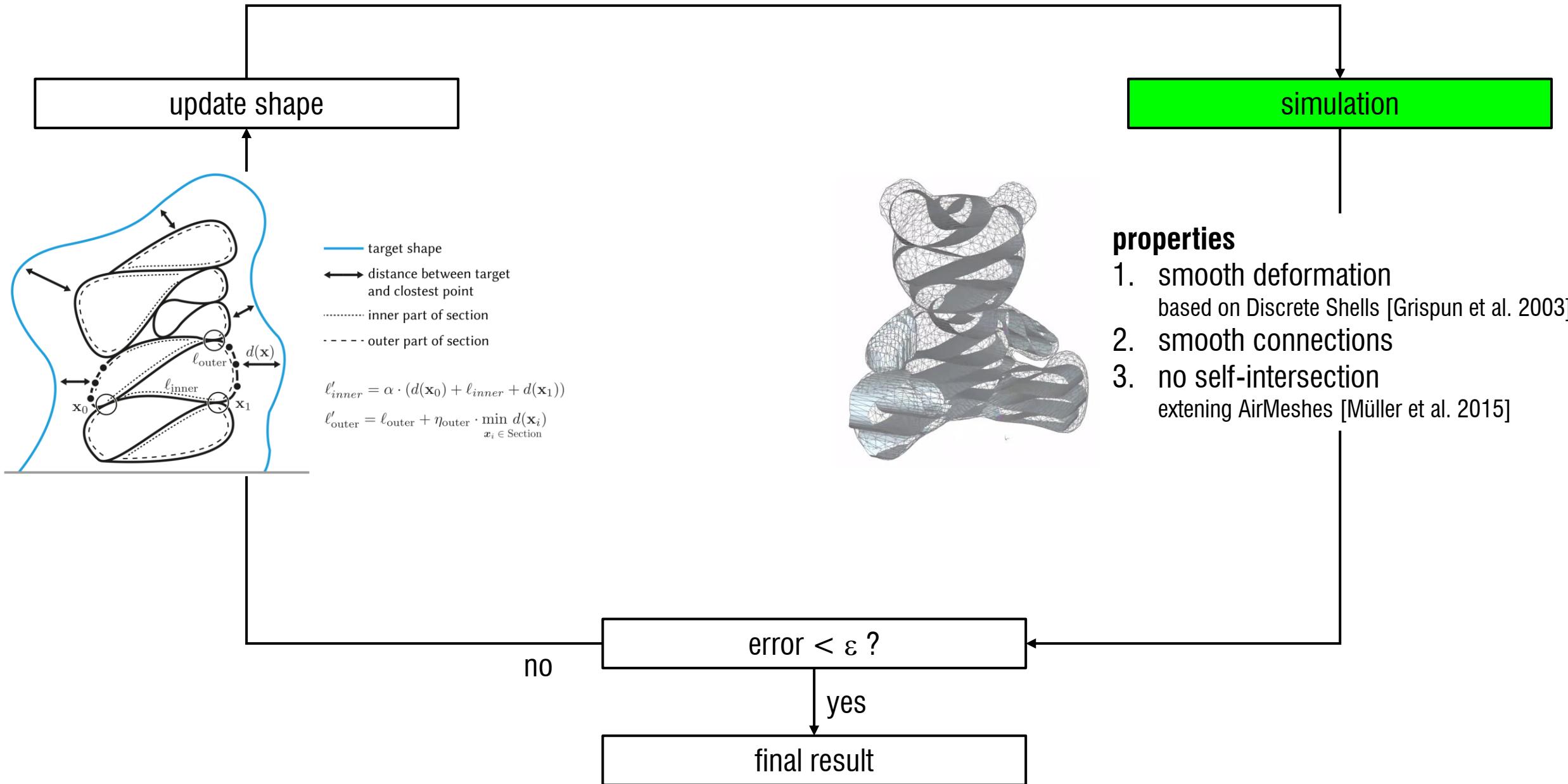






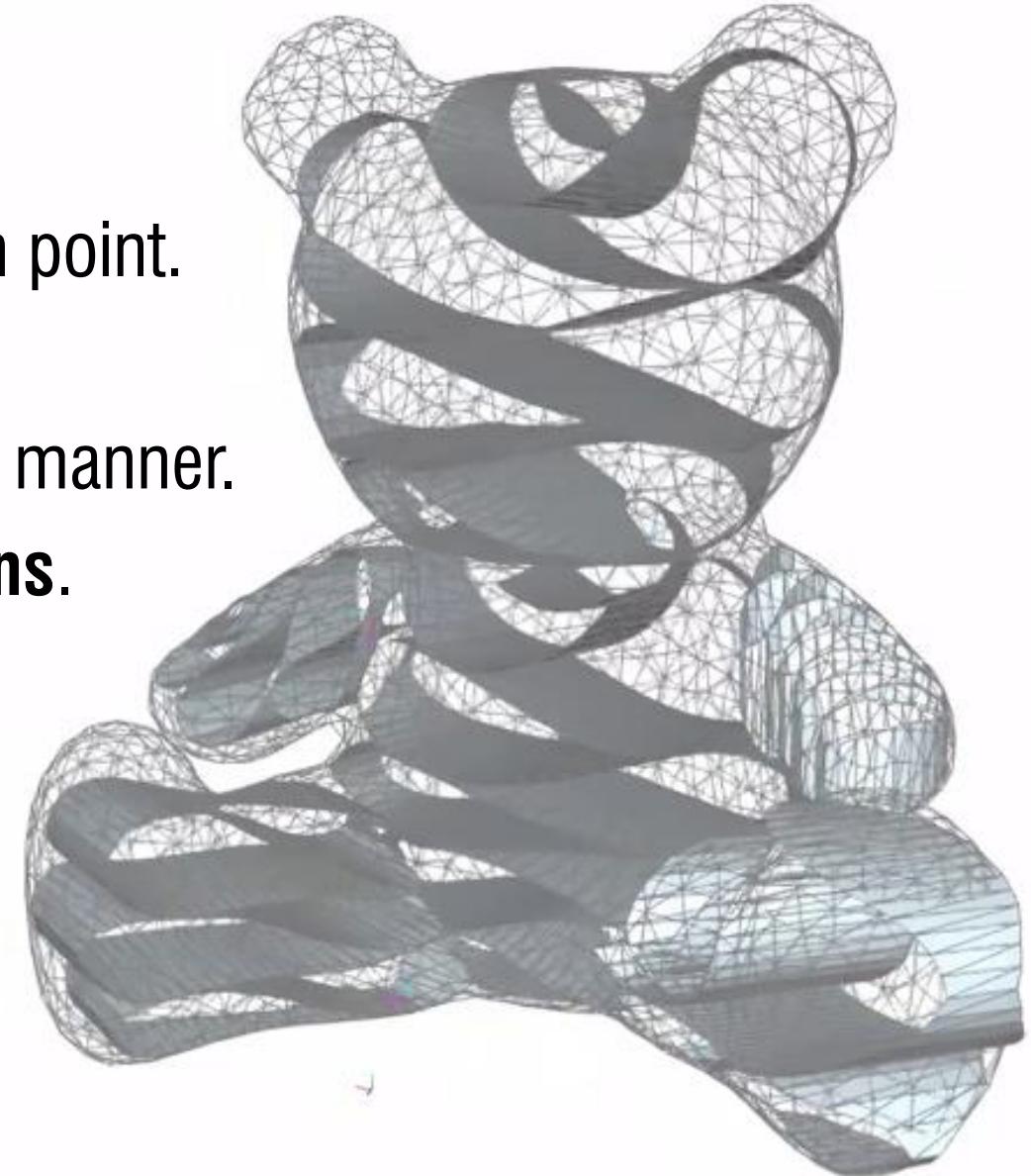


inner: simulation



properties

1. The ruffle deforms **smoothly** at any given point.
2. The ruffle does **not stretch**, like paper.
3. The ruffle **connections** meet in a smooth manner.
4. The ruffle does not have **self-intersections**.



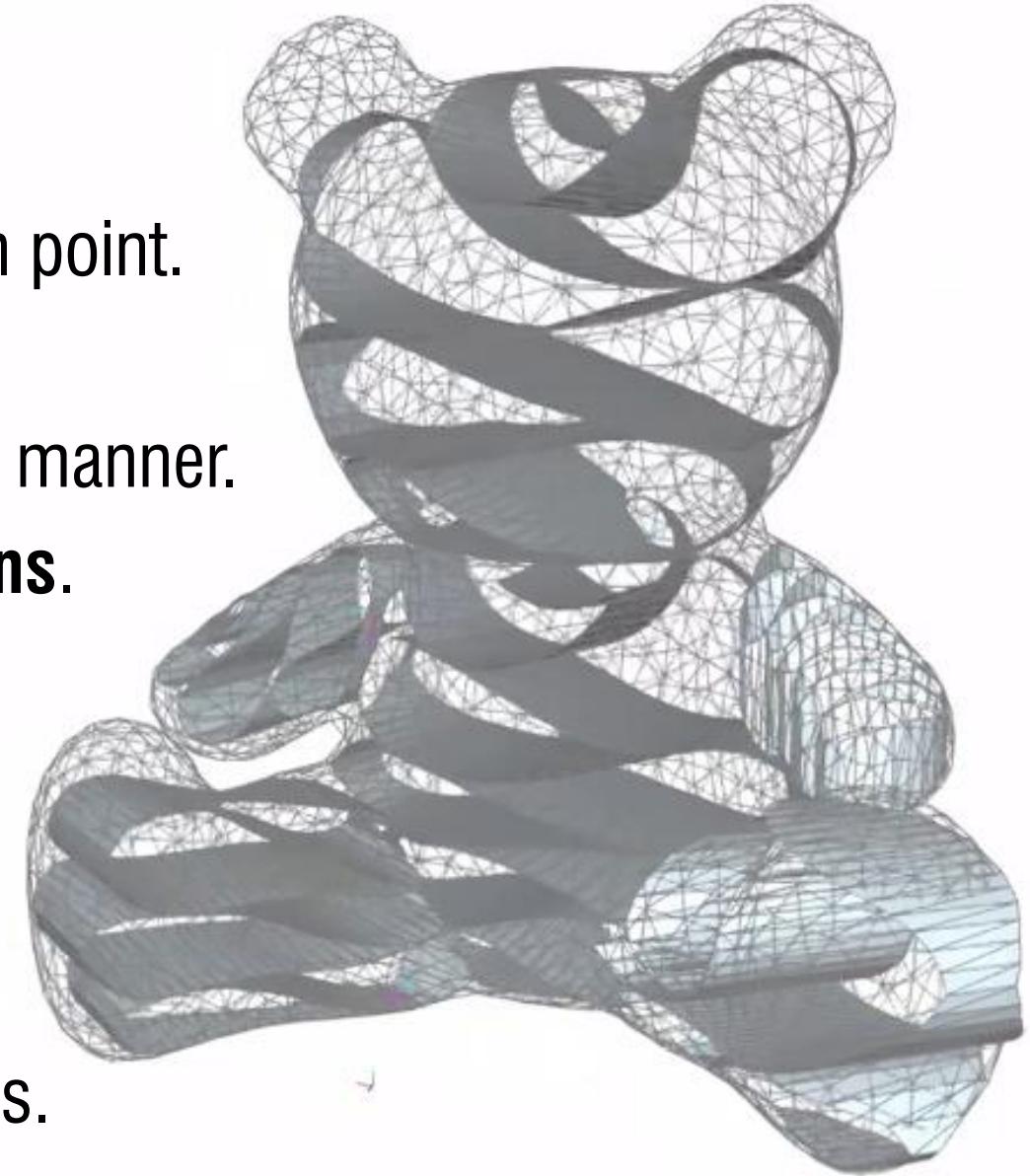
properties

1. The ruffle deforms **smoothly** at any given point.
2. The ruffle does **not stretch**, like paper.
3. The ruffle **connections** meet in a smooth manner.
4. The ruffle does not have **self-intersections**.

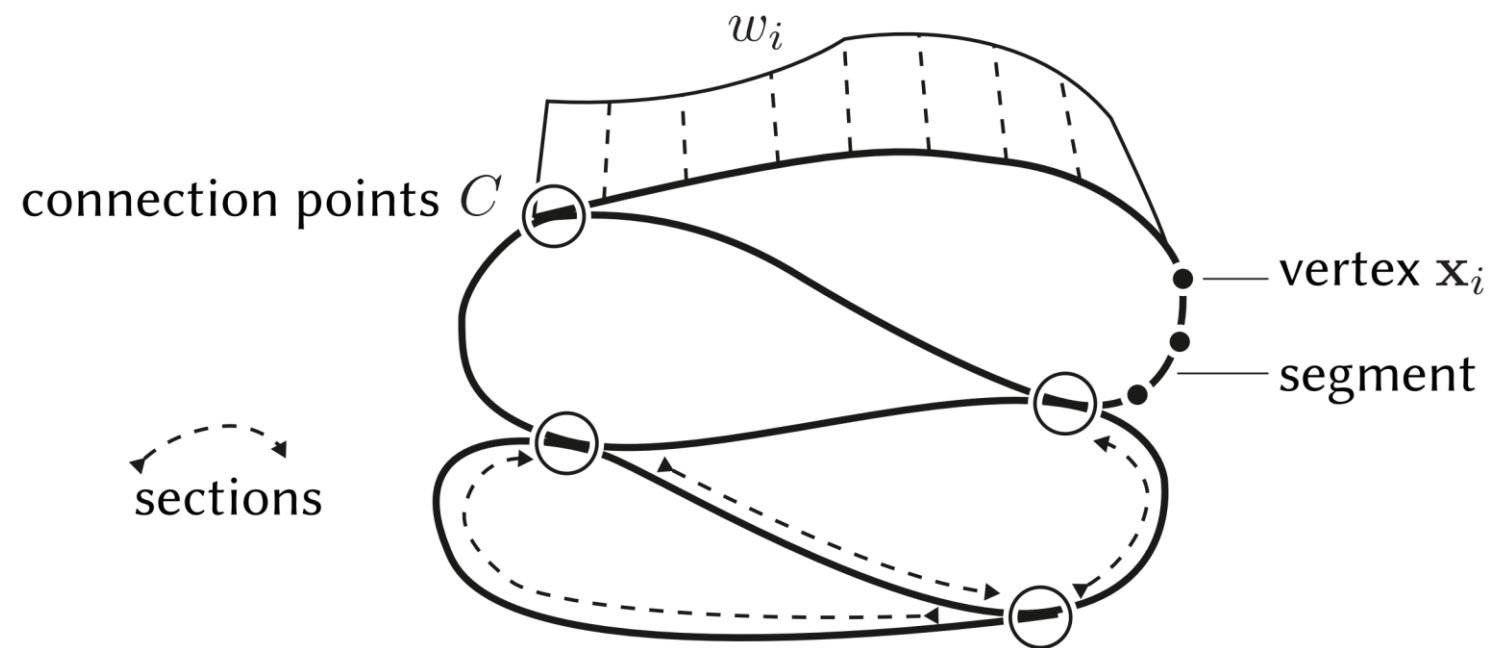
The simulation will be a minimizer

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} E(\mathbf{x})$$

where the energy **E(x)** models the properties.

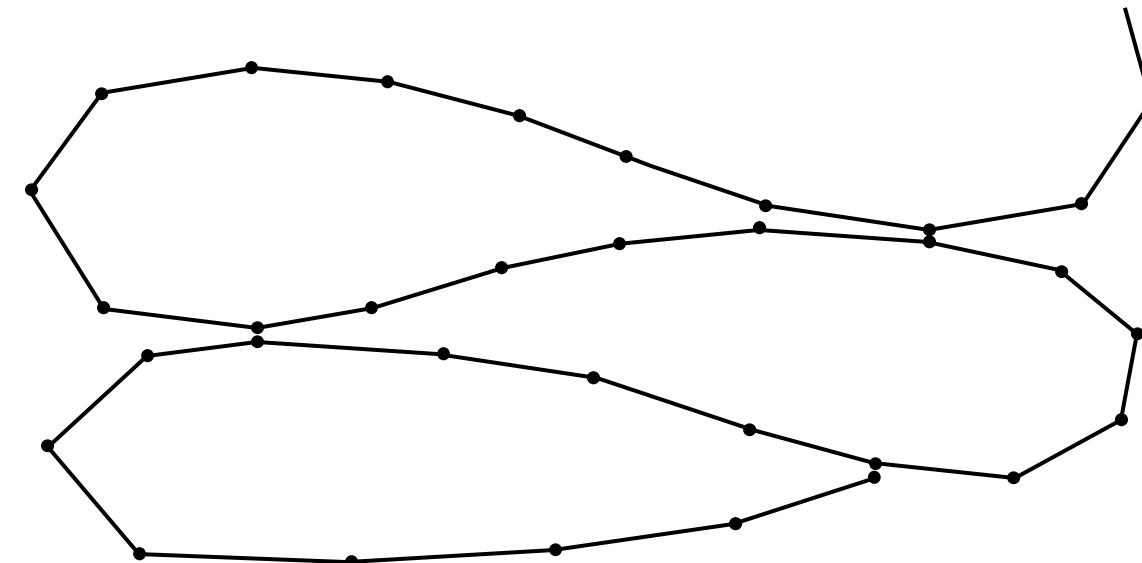


notation



properties → energies

1. The ruffle deforms **smoothly** at any given point.

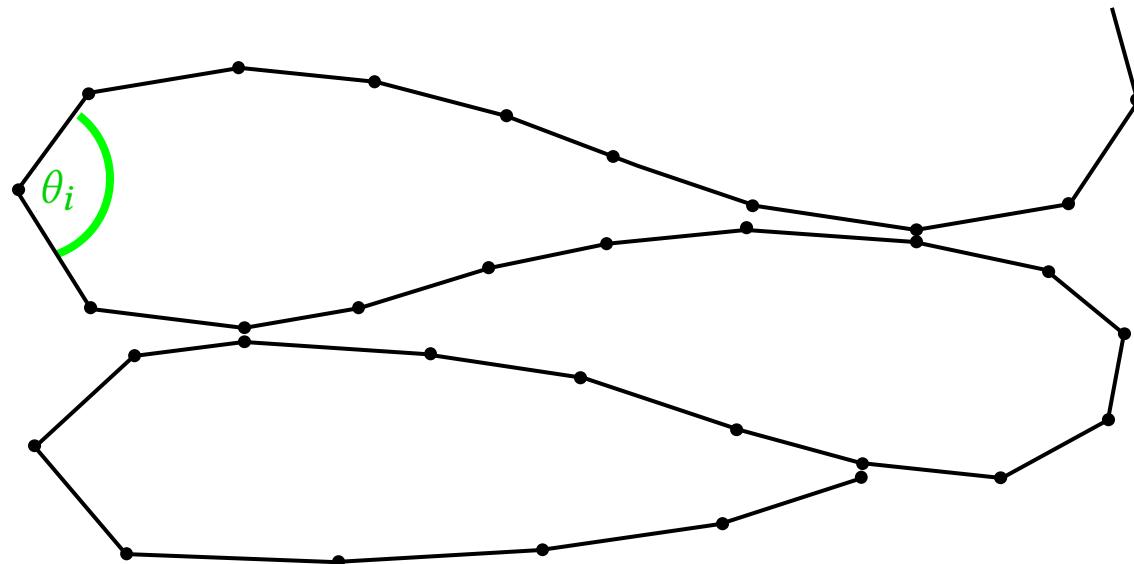


We want to get a small energy when the paper is at rest (*flat*), and a higher number the more we deform it. How would you model this smoothness energy?

<30sec brainstorming>

properties → energies

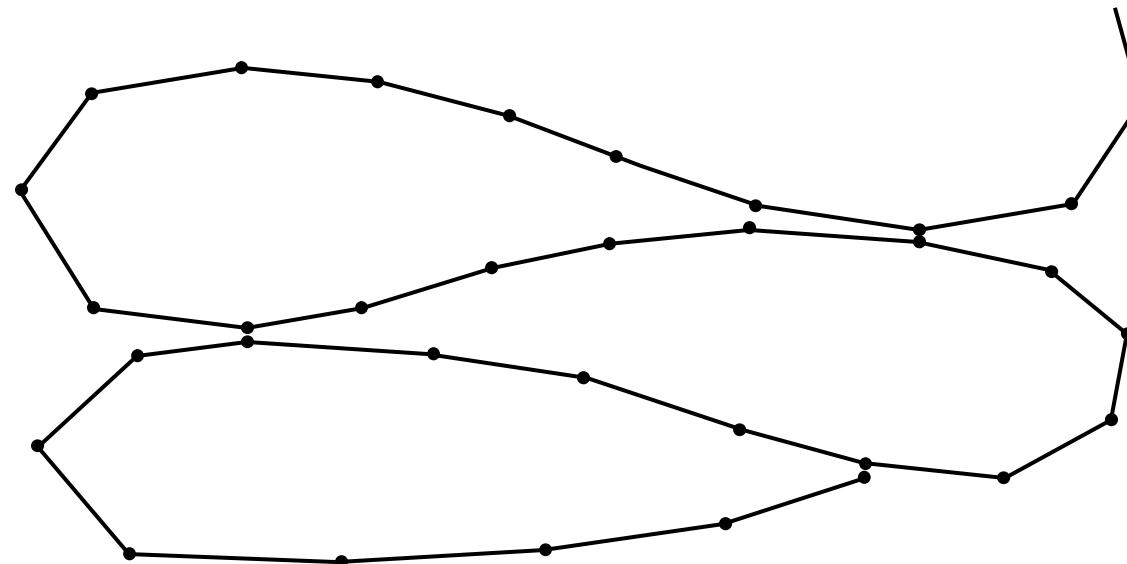
1. The ruffle deforms **smoothly** at any given point.



$$E_B(\mathbf{x}) = \sum_{i=2}^{N-1} (\theta_i - \pi)^2 w_i / \bar{\ell}_i$$

properties → energies

2. The ruffle does **not stretch**, like paper.

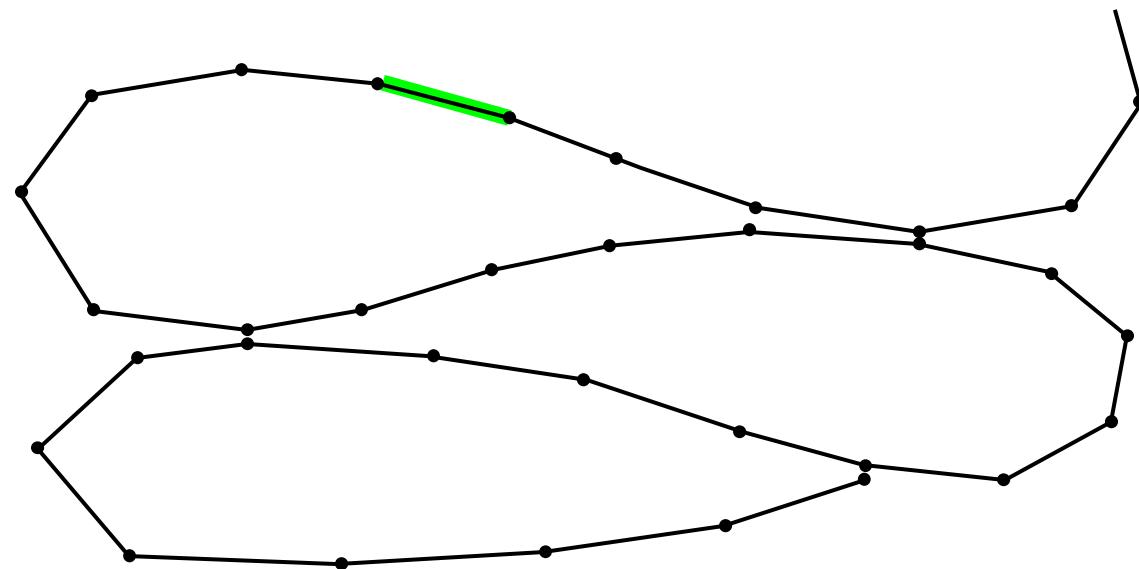


We want: small energy when the *lengths* stay the same, higher energy when deformed. How would you model this stretch energy?

<30sec brainstorming>

properties → energies

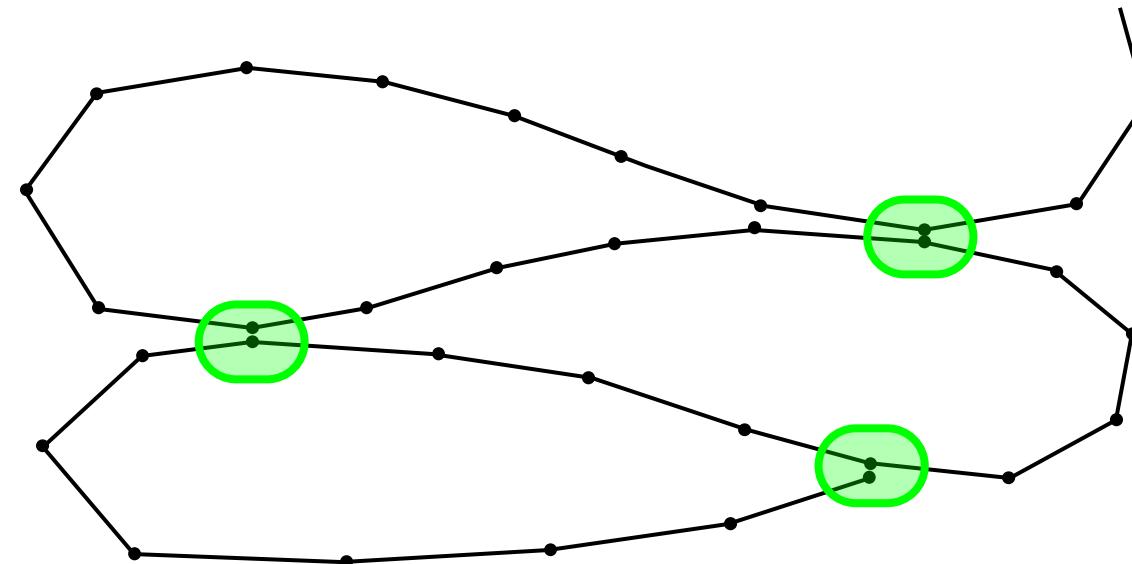
2. The ruffle does **not stretch**, like paper.



$$E_M(\mathbf{x}) = \sum_{i=1}^{N-1} (\|\mathbf{x}_{i+1} - \mathbf{x}_i\| - \ell_i)^2$$

properties → energies

3. The ruffle **connections** meet in a smooth manner.

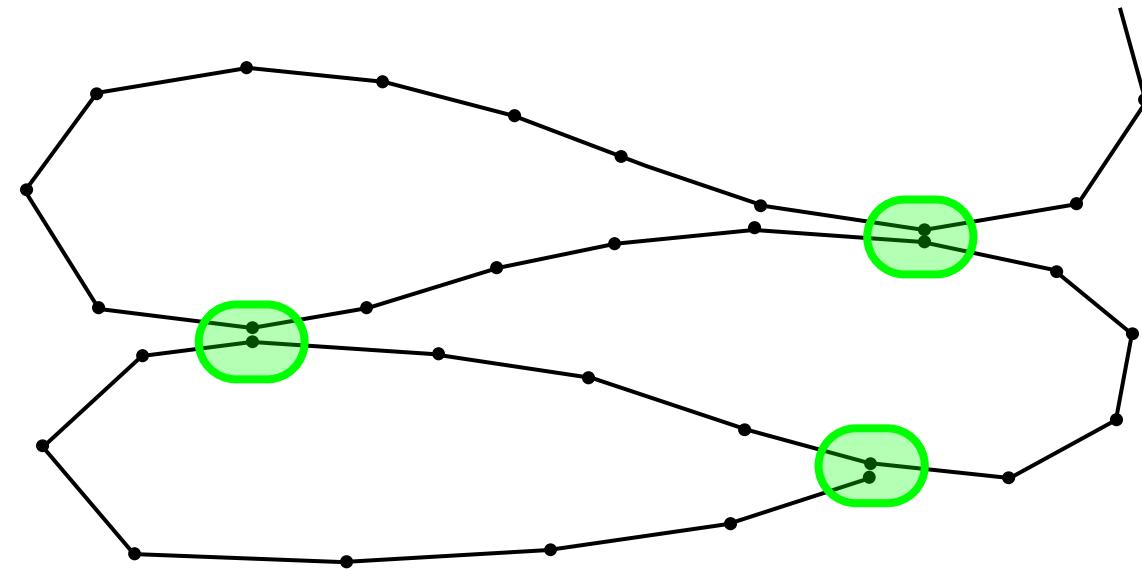


We want: small energy when the *tangents* are the same, higher energy when different. How would you model this energy?

<30sec brainstorming>

properties → energies

3. The ruffle **connections** meet in a smooth manner.

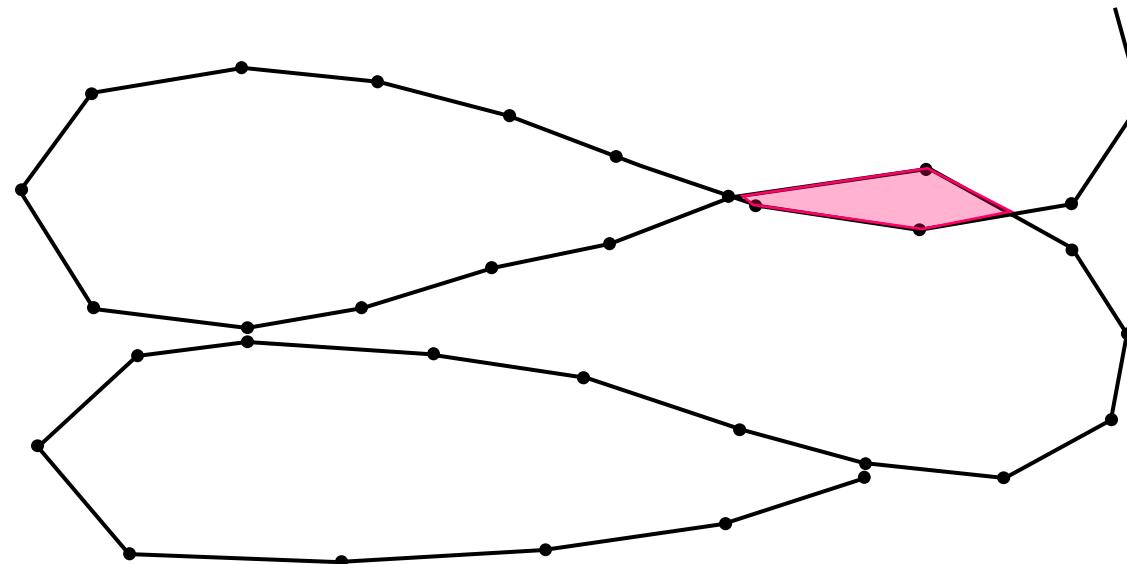


$$E_C(\mathbf{x}) = \sum_{(i,j) \in C} (\theta_{ij}^+ - \pi)^2 w_i / \bar{\ell}_{ij}^+ + (\theta_{ij}^- - \pi)^2 w_i / \bar{\ell}_{ij}^-$$

$$\theta_{ij}^\pm = \angle(\mathbf{x}_{i\pm 1}, \mathbf{x}_i, \mathbf{x}_{j\pm 1})$$

properties → energies

4. The ruffle does not have **self-intersections**.

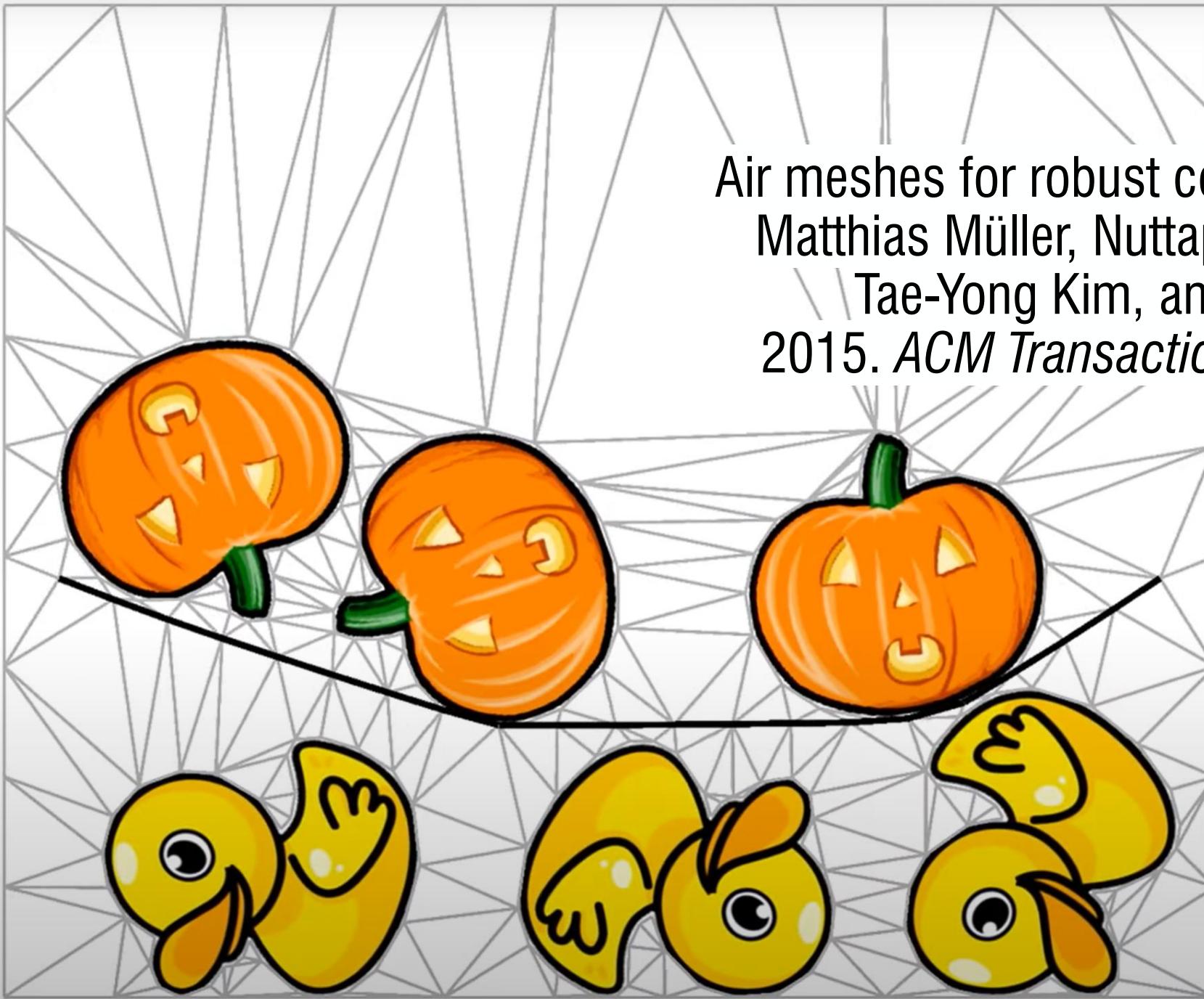


We want: small energy at no intersection, high energy otherwise. This is a long-standing challenge in engineering.

We use...

[Video](#)

Air meshes for robust collision handling.
Matthias Müller, Nuttapong Chentanez,
Tae-Yong Kim, and Miles Macklin.
2015. *ACM Transactions on Graphics*.



Simulation

Putting it all together

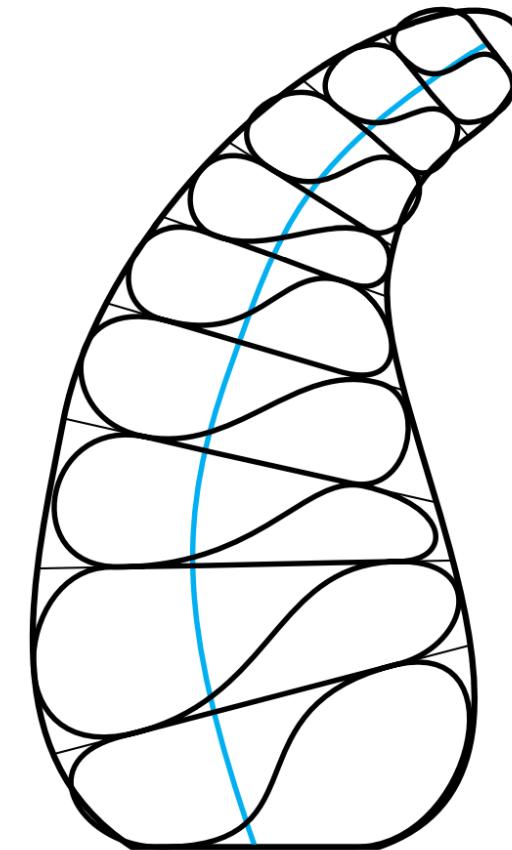
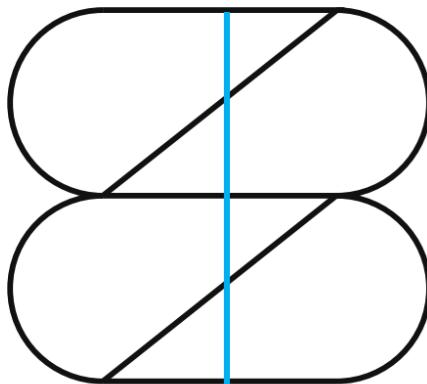
$$E(\mathbf{x}) = k_M E_M(\mathbf{x}) + k_B(E_B(\mathbf{x}) + E_C(\mathbf{x})) + k_{AM} E_{AM}(\mathbf{x})$$

and solve for

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} E(\mathbf{x})$$

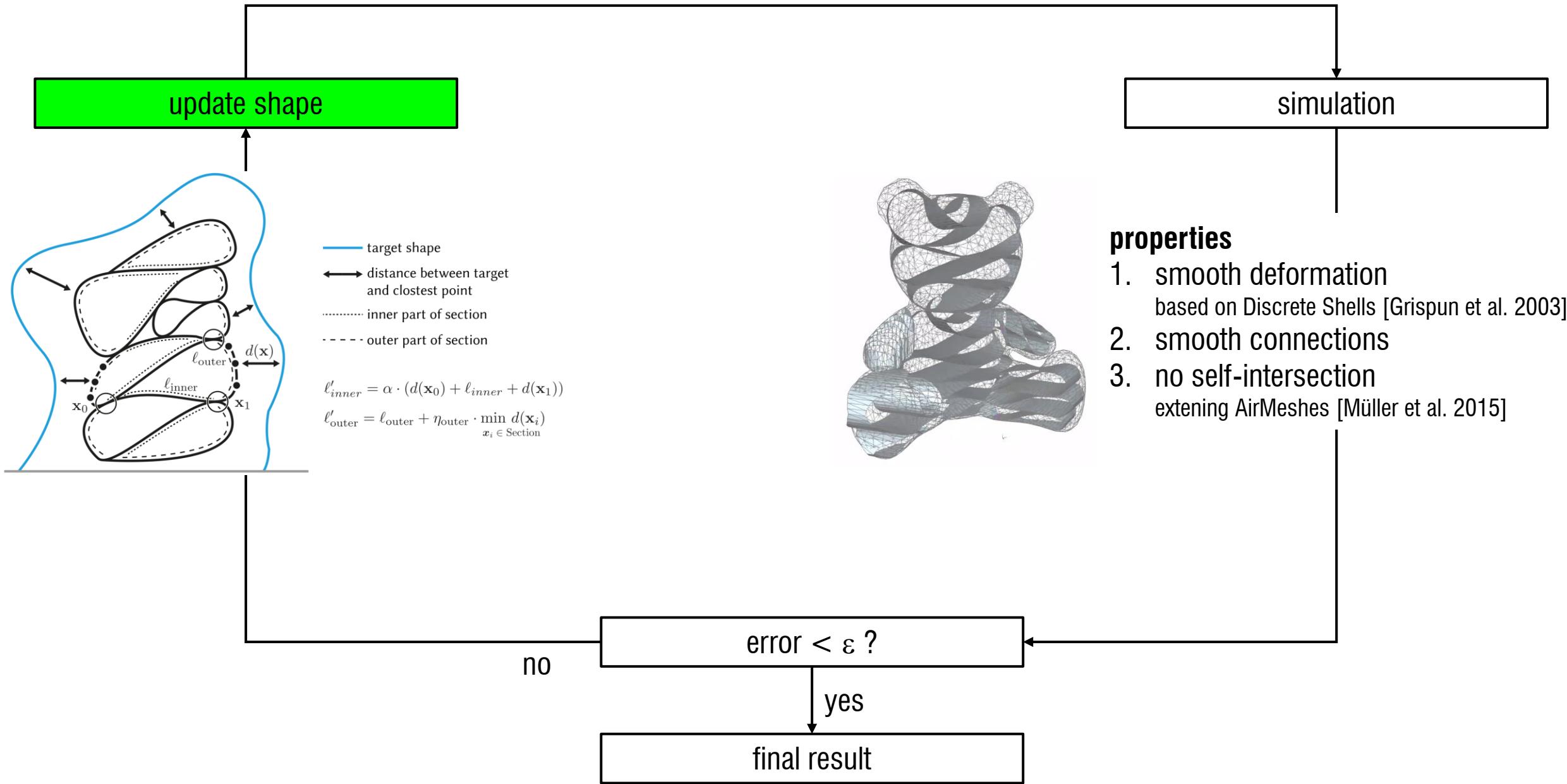
Initial guess

Ruffle construction by *generator curve*

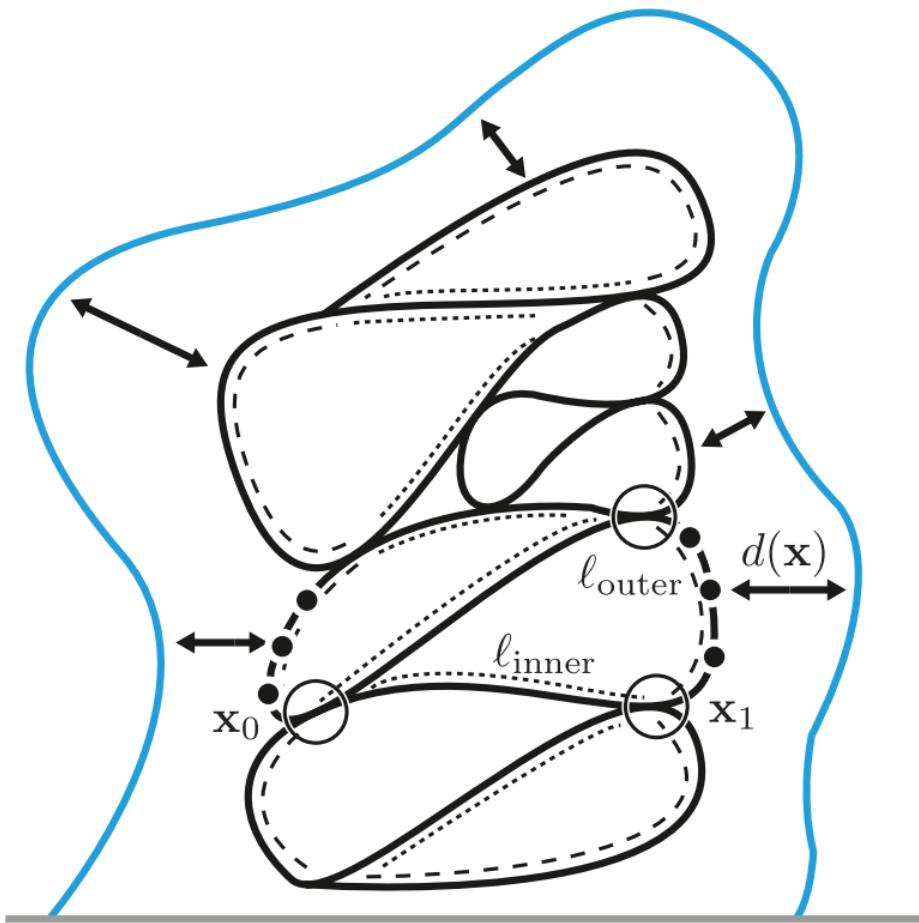


Let's see this in **3_ruffle_simulation**

outer: optimization



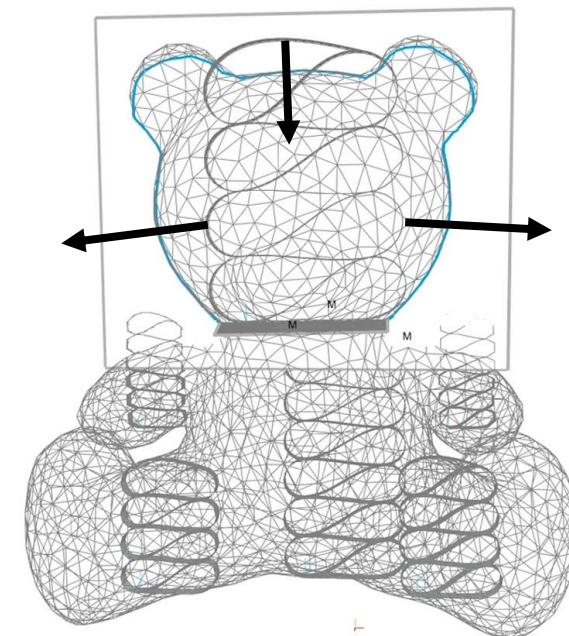
shape approximation

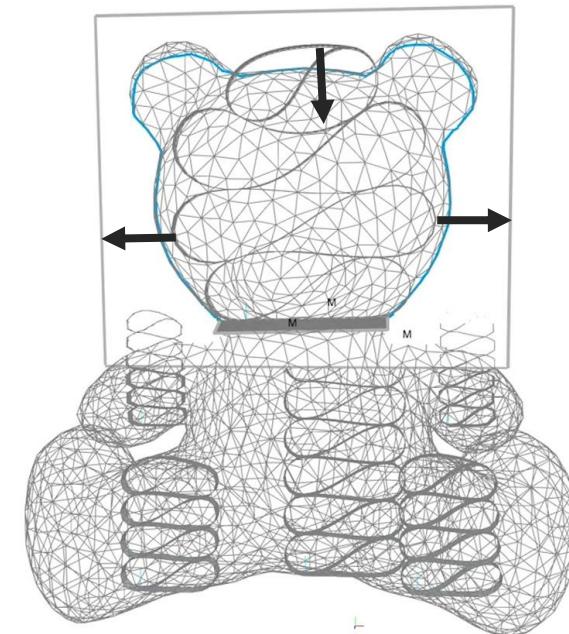


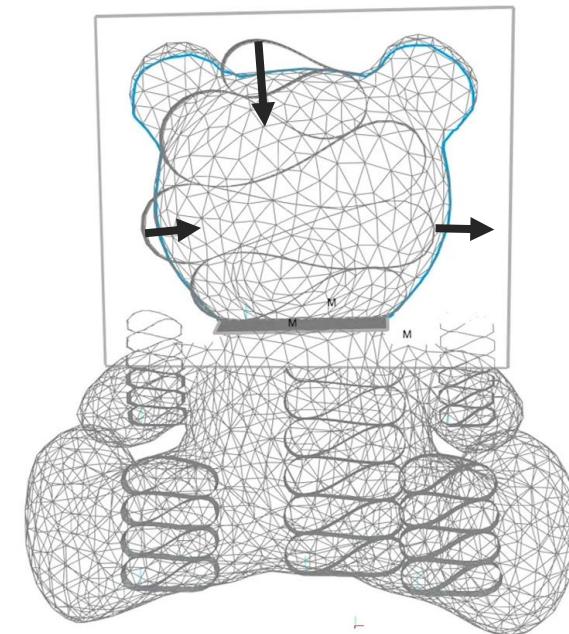
- target shape
- ↔ distance between target and closest point
- inner part of section
- - - outer part of section

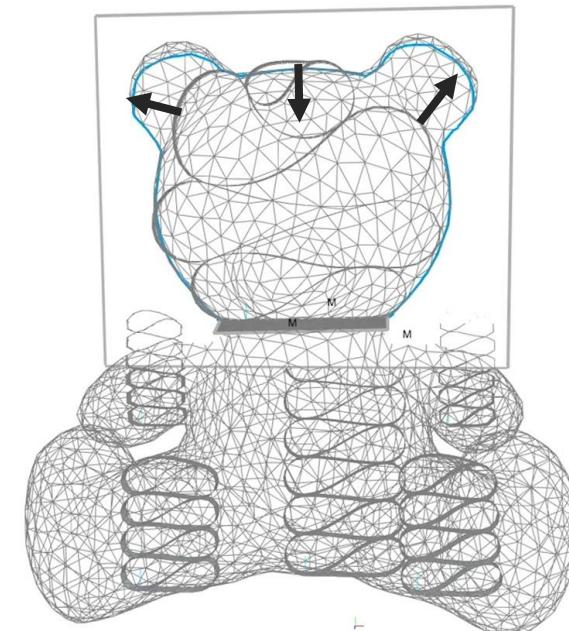
$$\ell'_{inner} = \alpha \cdot (d(\mathbf{x}_0) + \ell_{inner} + d(\mathbf{x}_1))$$

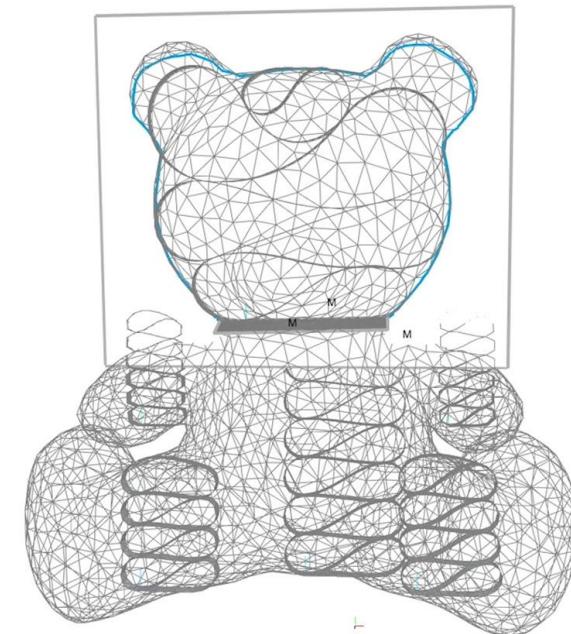
$$\ell'_{outer} = \ell_{outer} + \eta_{outer} \cdot \min_{\mathbf{x}_i \in \text{Section}} d(\mathbf{x}_i)$$











scope



current
optimize shape

next
→ optimize for **forces**

summary

Summary

Geometry is awesome

Generate & edit meshes, send to 3D
printer → physical objects!

Generate complex materials



Useful tools & data

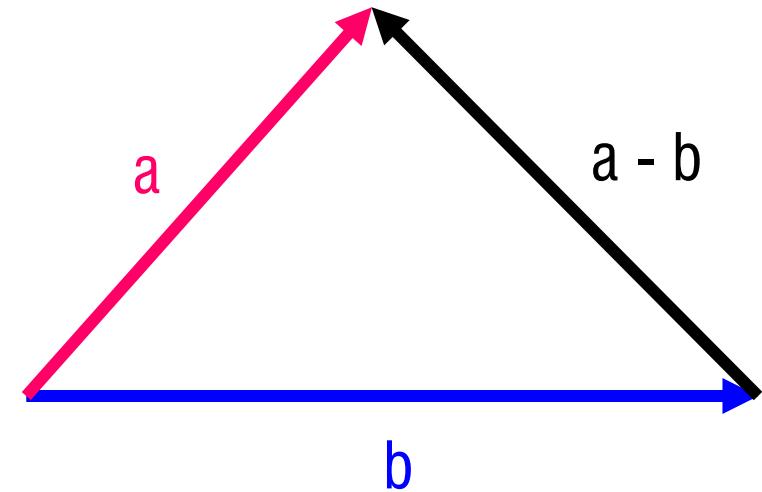
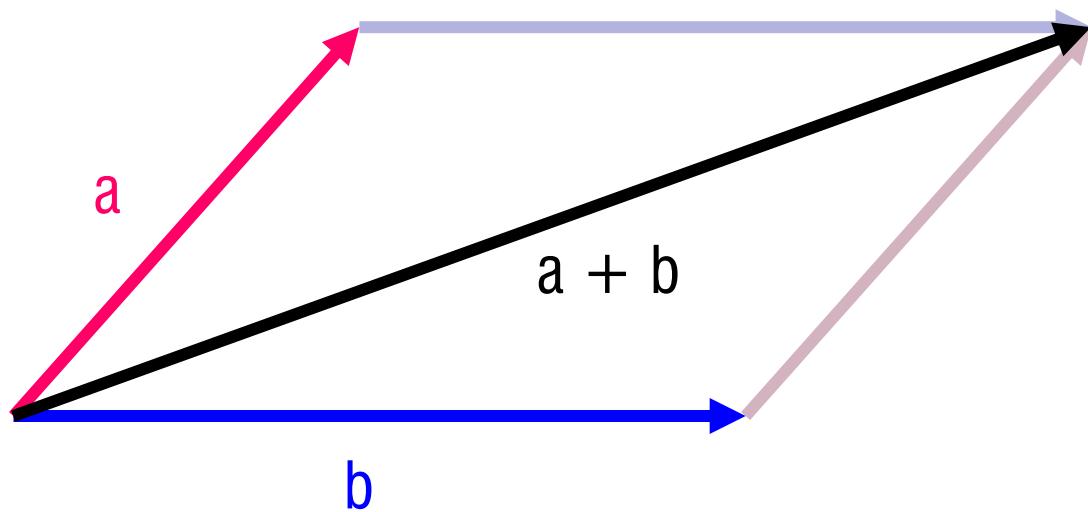
MeshLab

Open Flipper

Meshes: <https://github.com/libigl/libigl-tutorial-data>

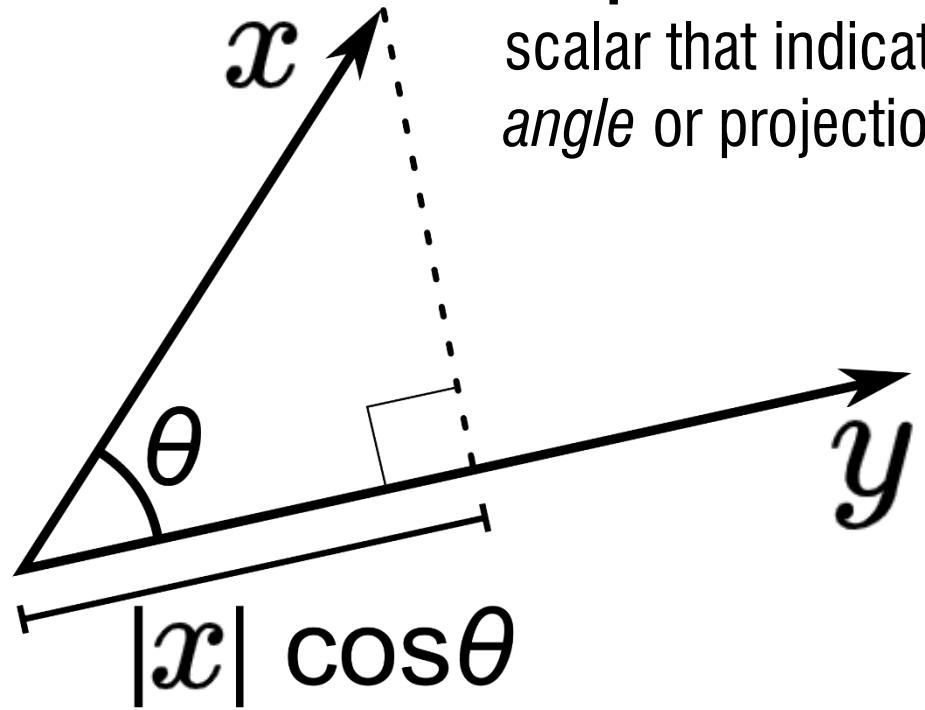
end

Vector cheat sheet



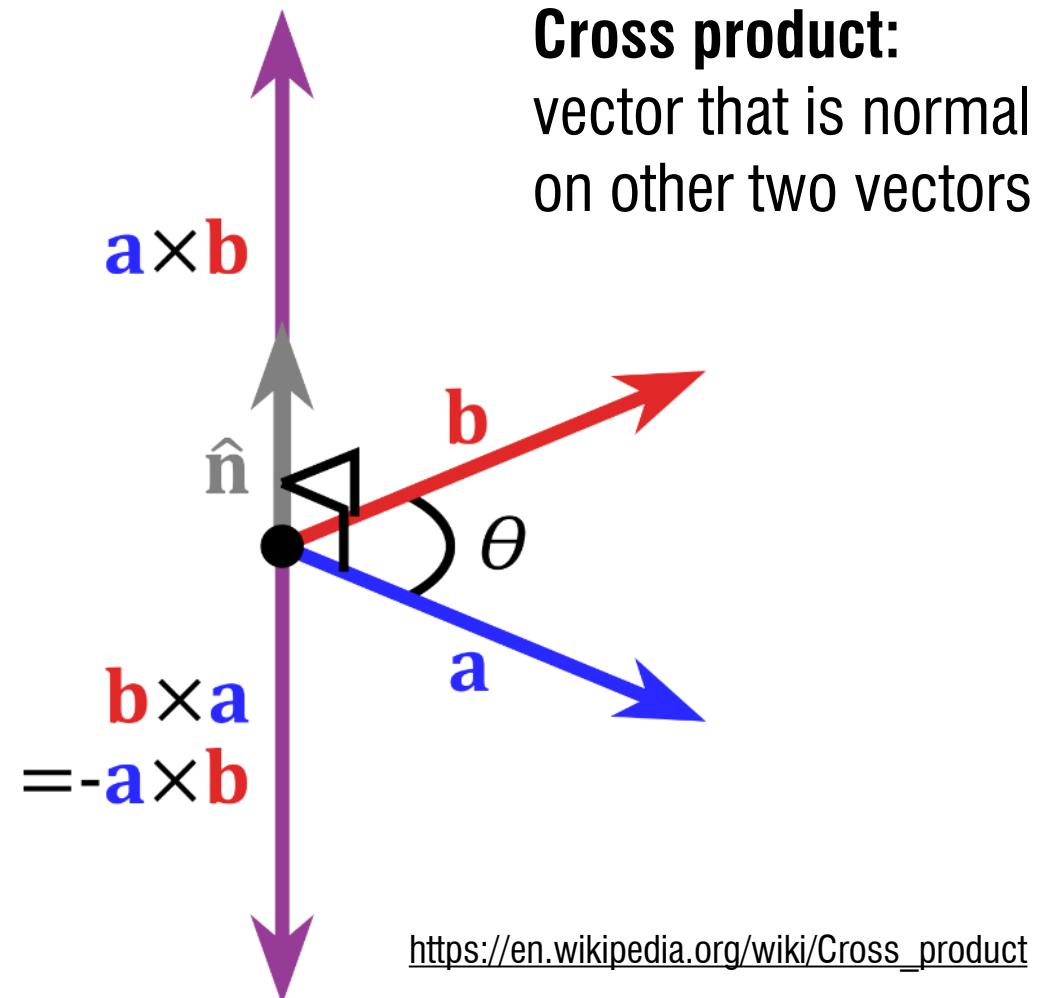
https://en.wikipedia.org/wiki/Euclidean_vector

Vector cheat sheet



Dot product:
scalar that indicates
angle or projection

$$\angle(x, y) = \arccos \frac{\langle x, y \rangle}{\|x\| \|y\|}$$



https://en.wikipedia.org/wiki/Cross_product