Shape Modeling and Geometry Processing

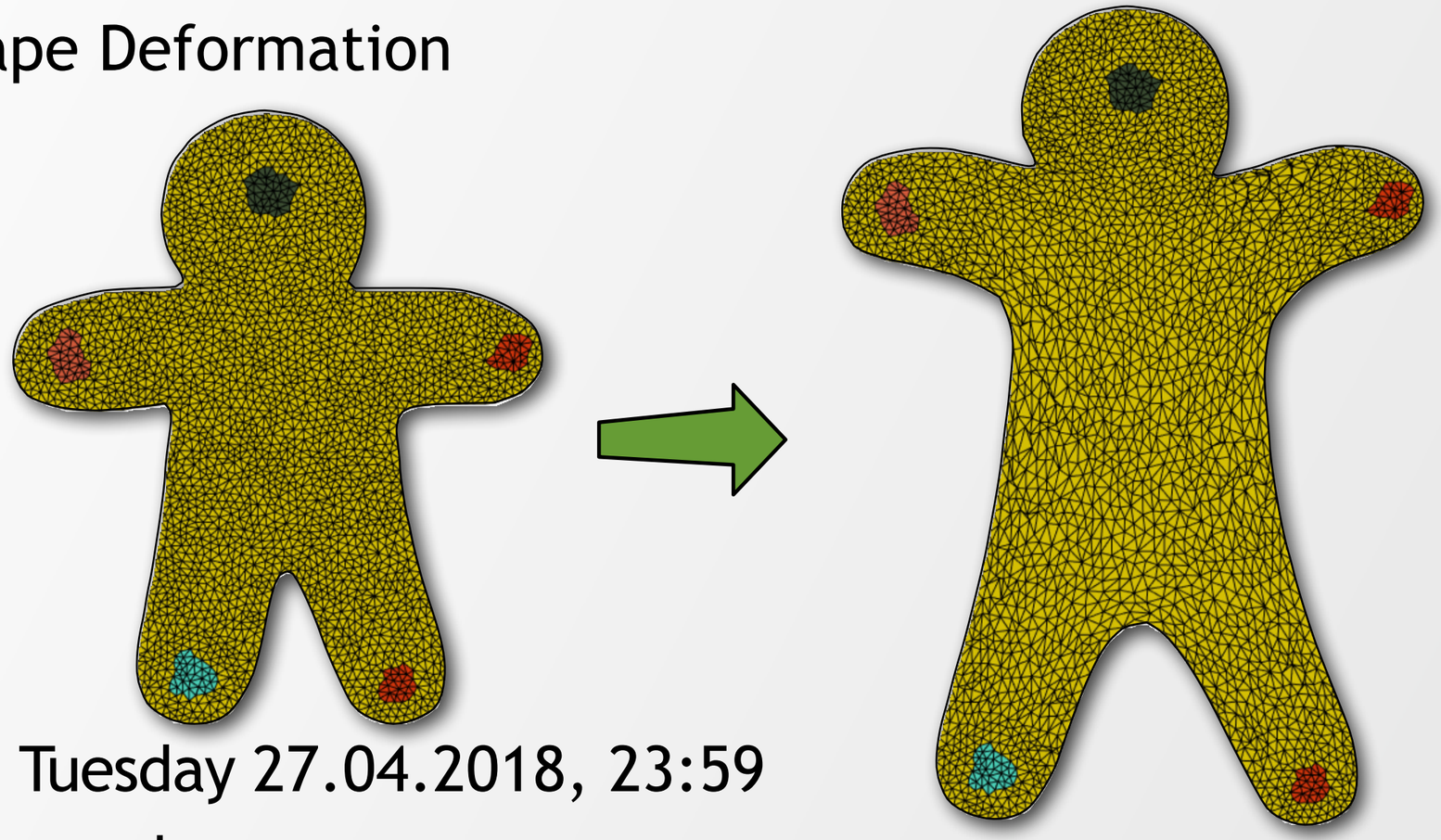
Exercise 4 - Shape Deformation

INTERACTIVE GEOMETRY LAB

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

This exercise

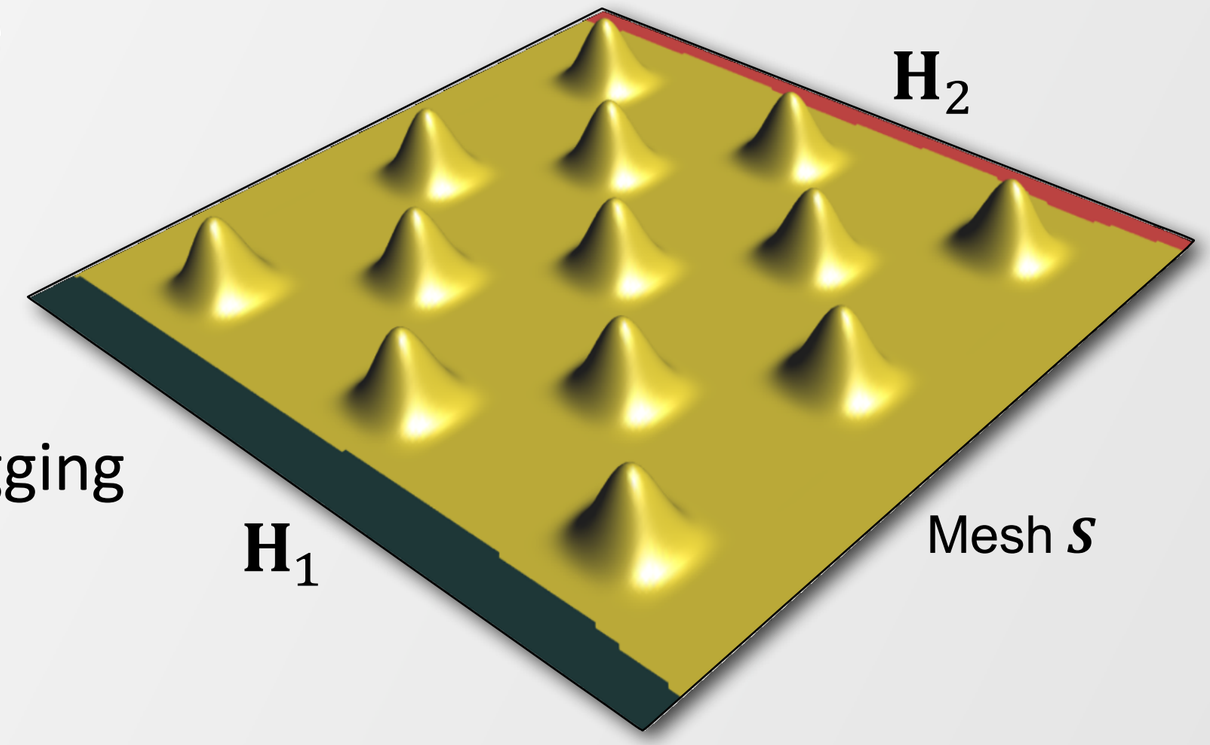
- Topic: Shape Deformation



- Deadline: Tuesday 27.04.2018, 23:59
- 14% of the grade

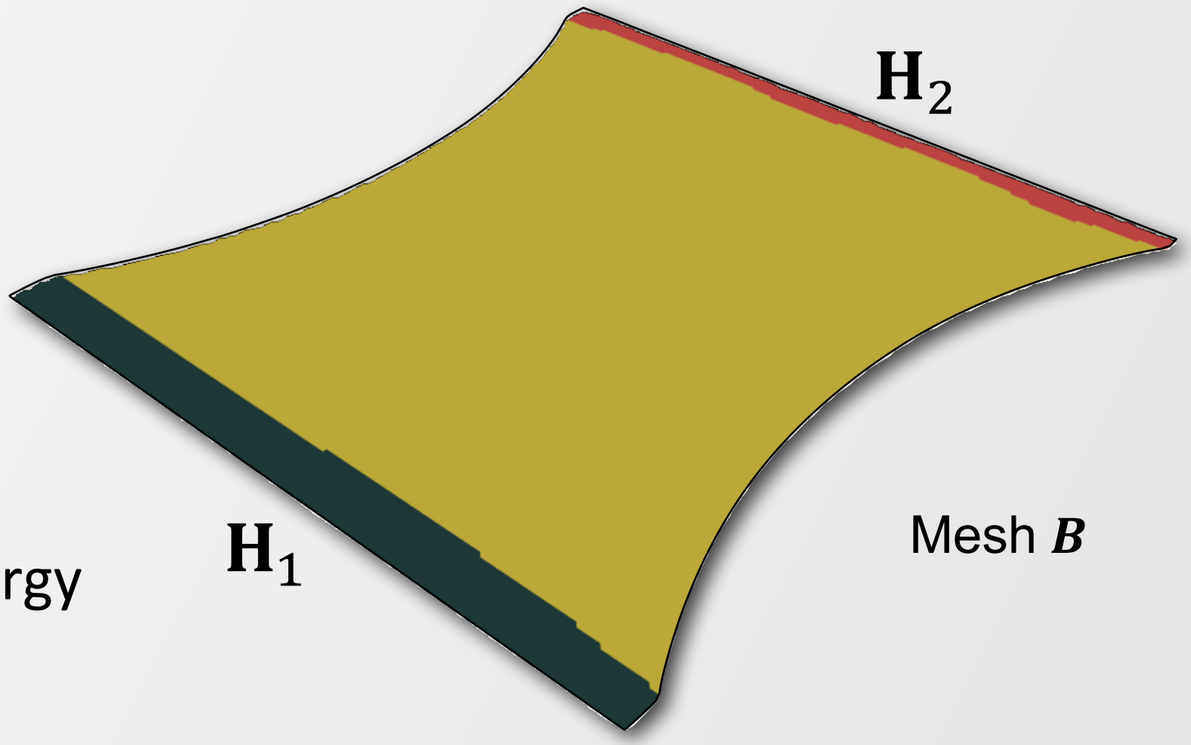
Step 1: Select Handle Regions

- Select with mouse or load from file
- Move one handle H_i at a time
- Rest of the handles stay fixed
- Code provided for the picking/dragging



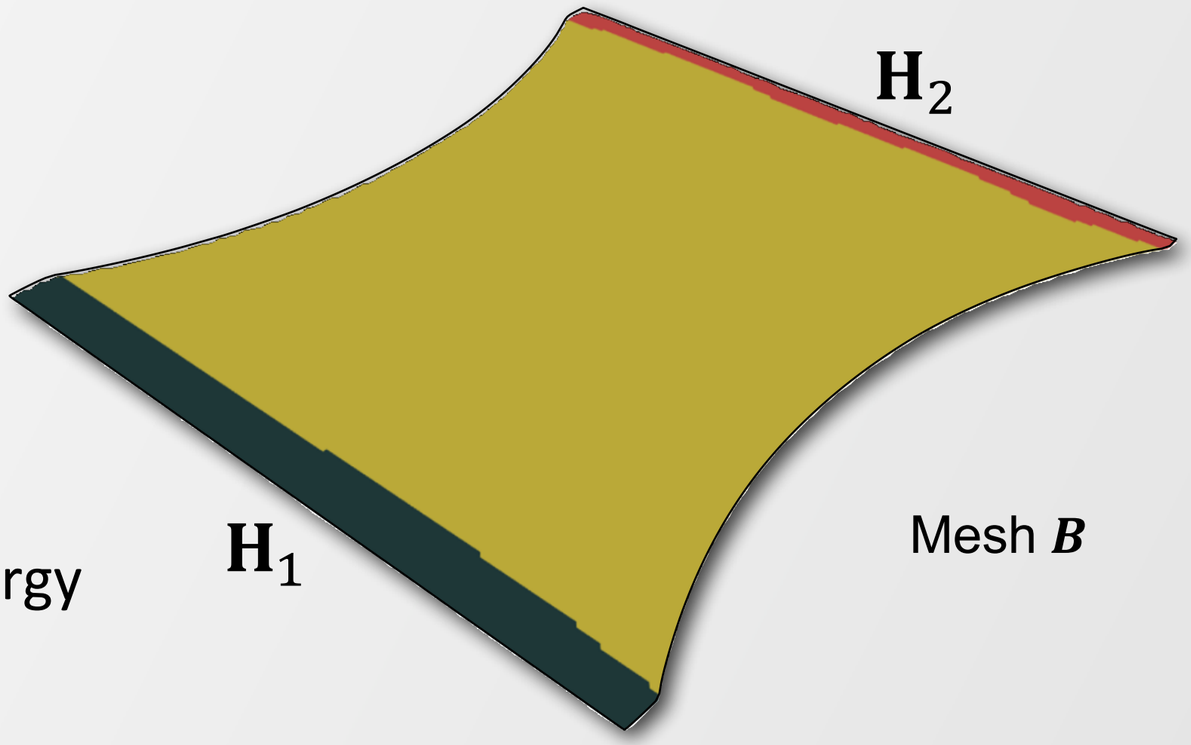
Step 2: Smoothing

- Remove high-frequency details
- Only the smoothed mesh will be deformed, and the details will be transferred later
- Solve a bi-Laplacian system
  - solution minimizes the Laplacian Energy



Step 2: Smoothing

- Remove high-frequency details
- Only the smoothed mesh will be deformed, and the details will be transferred later
- Solve a bi-Laplacian system
  - solution minimizes the Laplacian Energy



$$\min_{\mathbf{v}} \mathbf{v}^{\mathrm{T}} \mathbf{L}_{\omega} \mathbf{M}^{-1} \mathbf{L}_{\omega} \mathbf{v}$$

$$\text{s.t.} \quad \mathbf{v}_{\mathbf{H_i}} = \mathbf{o}_{\mathbf{H_i}}$$

Original positions

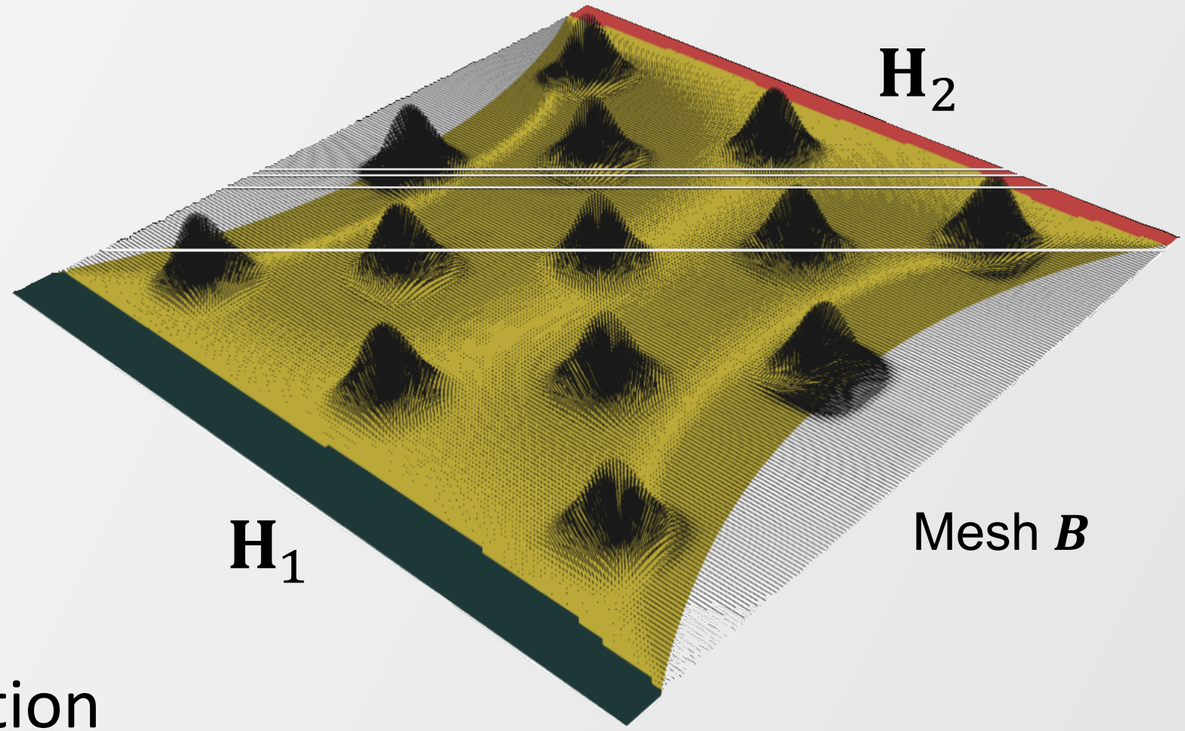
Step 3: Encode displacements

- Per-vertex displacement for $\boldsymbol{B}$ to $\boldsymbol{S}$

$$\mathbf{d}_i = \mathbf{v}_i^{\boldsymbol{S}} - \mathbf{v}_i^{\boldsymbol{B}}$$

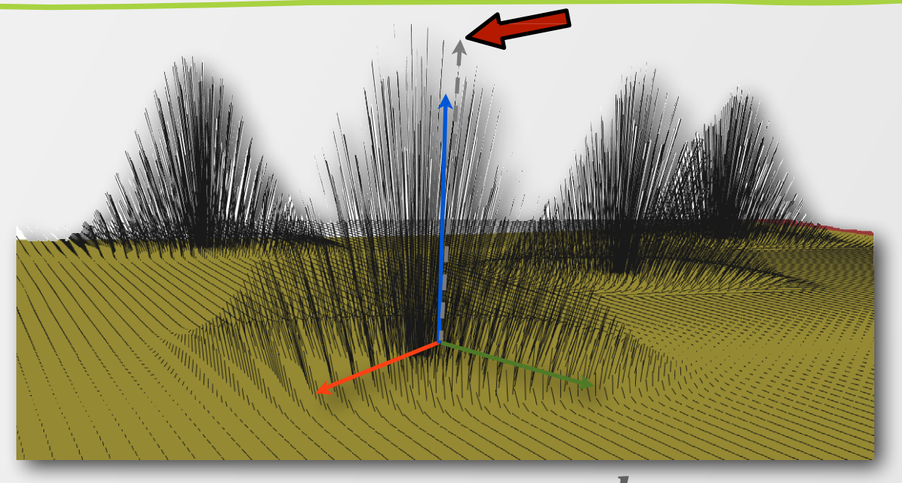
- $\mathbf{d_i}$ represent the details

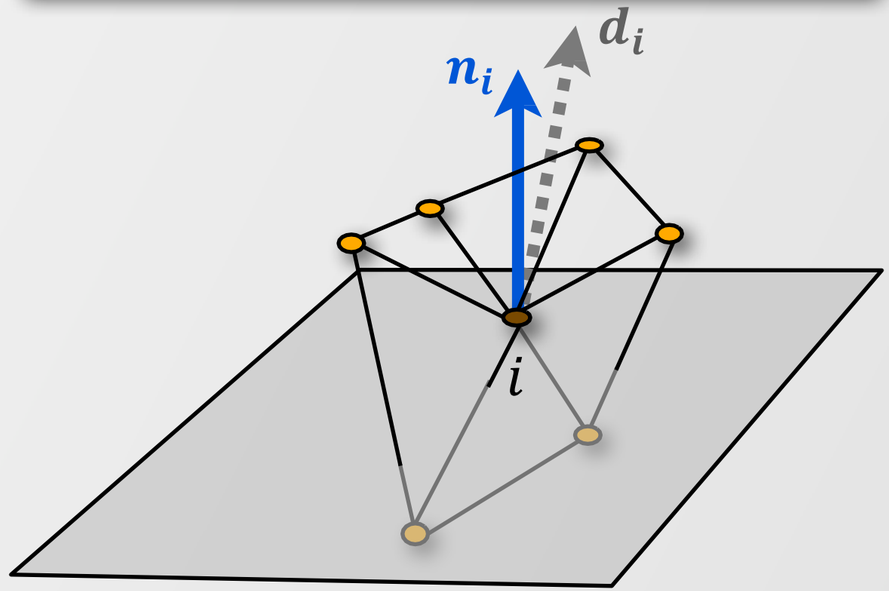
- Will be added back after deformation



Step 3: Encode displacements

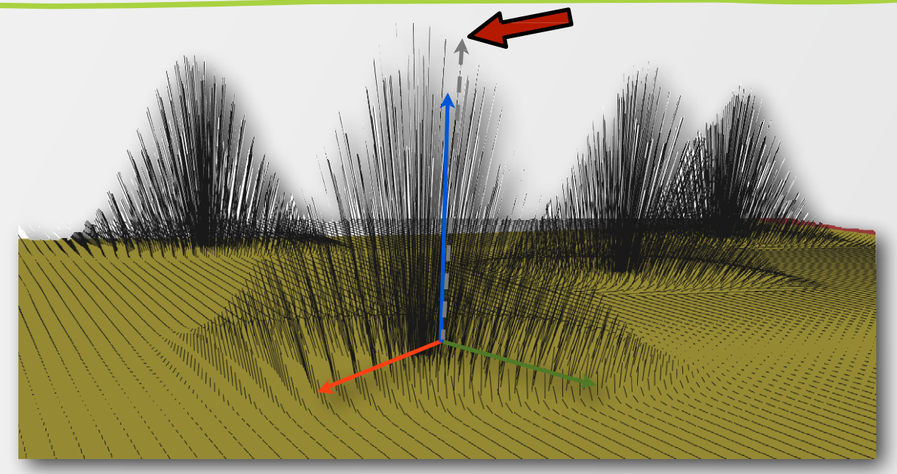
- Construct the local frame for vertex i
  - Calculate normal n_i for the smooth surface $\boldsymbol{B}$

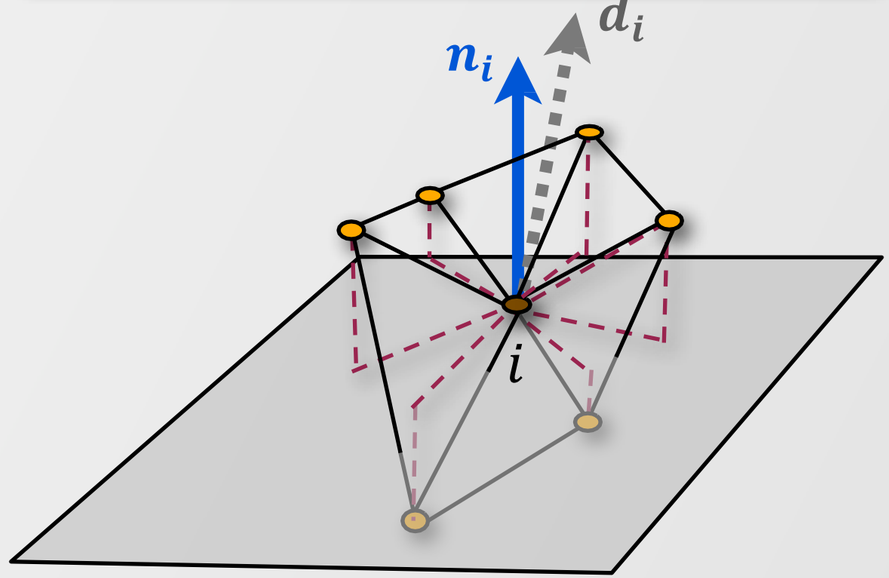




Step 3: Encode displacements

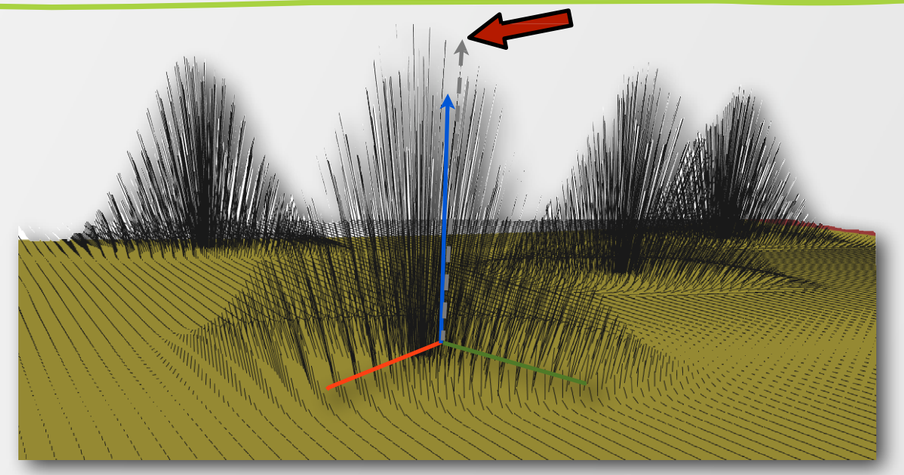
- Construct the local frame for vertex i
  - Calculate normal n_i for the smooth surface $\boldsymbol{B}$
  - Project all neighboring vertices on the tangent plane, perpendicular to n_i

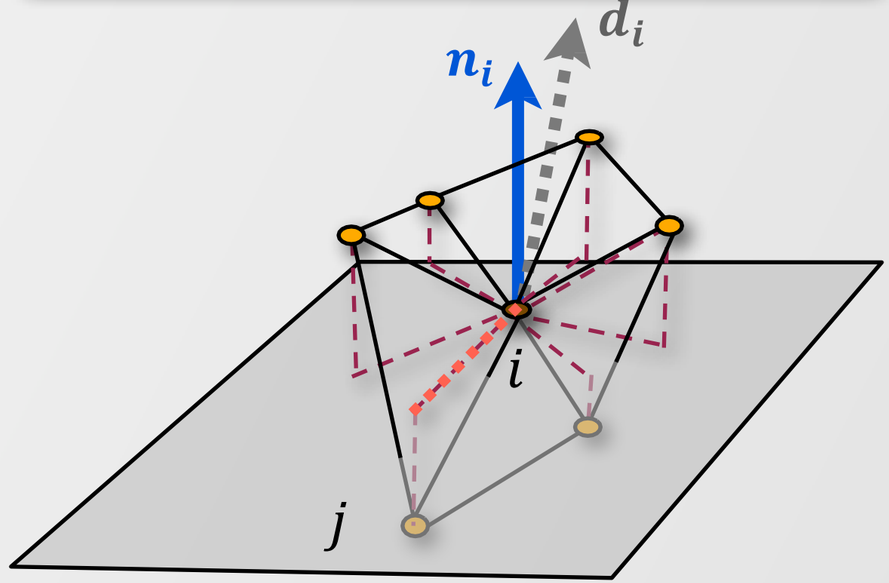




Step 3: Encode displacements

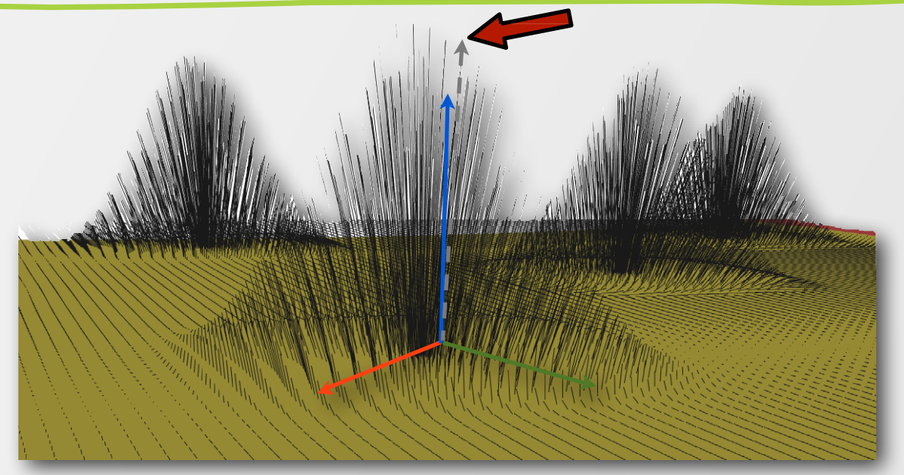
- Construct the local frame for vertex i
  - Calculate normal n_i for the smooth surface $\boldsymbol{B}$
  - Project all neighboring vertices on the tangent plane, perpendicular to n_i
  - Find the longest projected edge (the segment ij in the figure). Normalize it and call it x_i

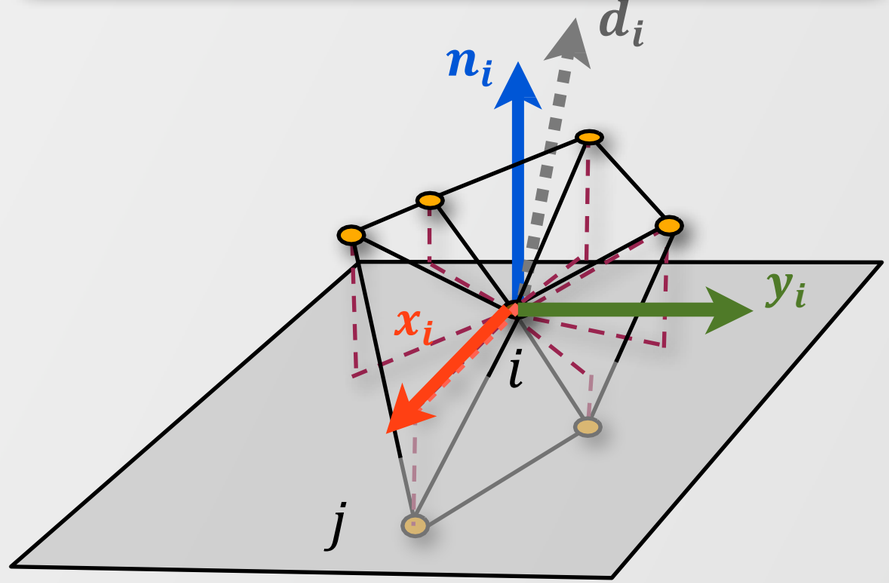




Step 3: Encode displacements

- Construct the local frame for vertex i
  - Calculate normal n_i for the smooth surface $\boldsymbol{B}$
  - Project all neighboring vertices on the tangent plane, perpendicular to n_i
  - Find the longest projected edge (the segment ij in the figure). Normalize it and call it x_i
  - Construct y_i using the cross product



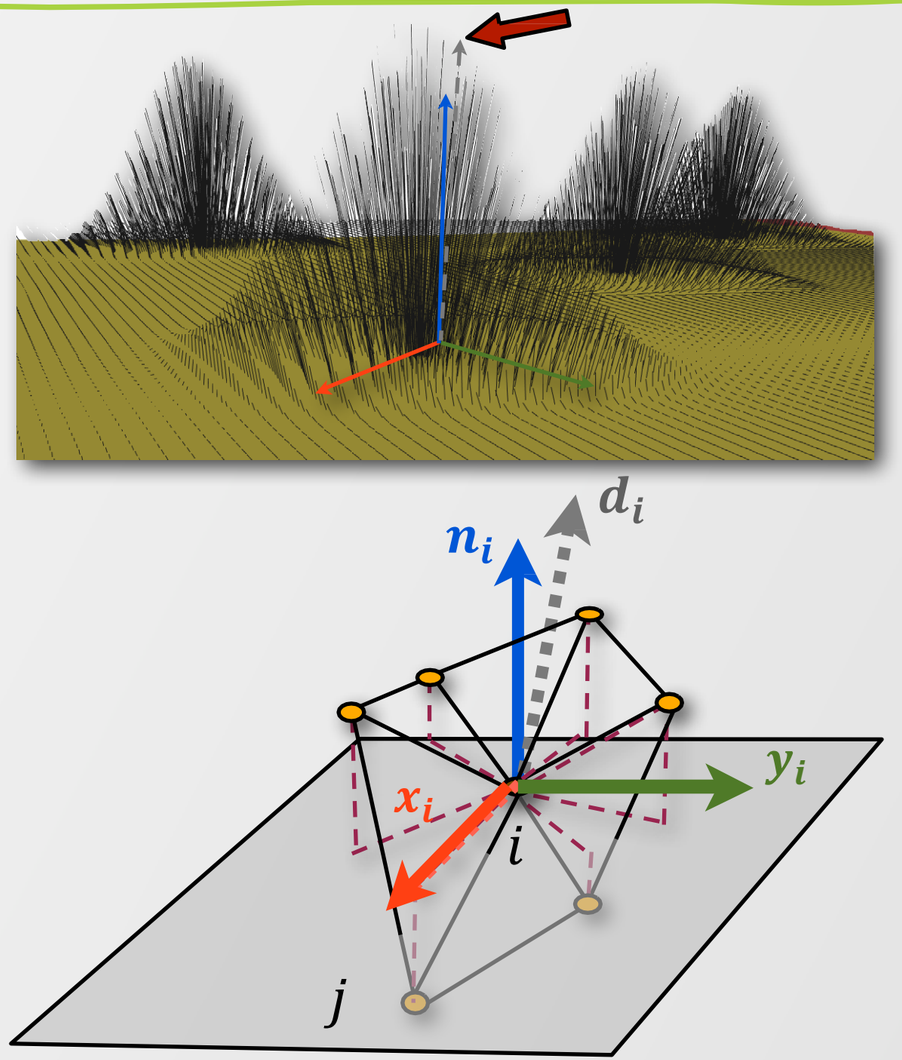


Step 3: Encode displacements

- Write d_i in terms of (x_i, y_i, n_i)

$$d_i = d_i^x x_i + d_i^y y_i + d_i^n n_i$$

- The basis is orthonormal so you can find the coefficients using dot products



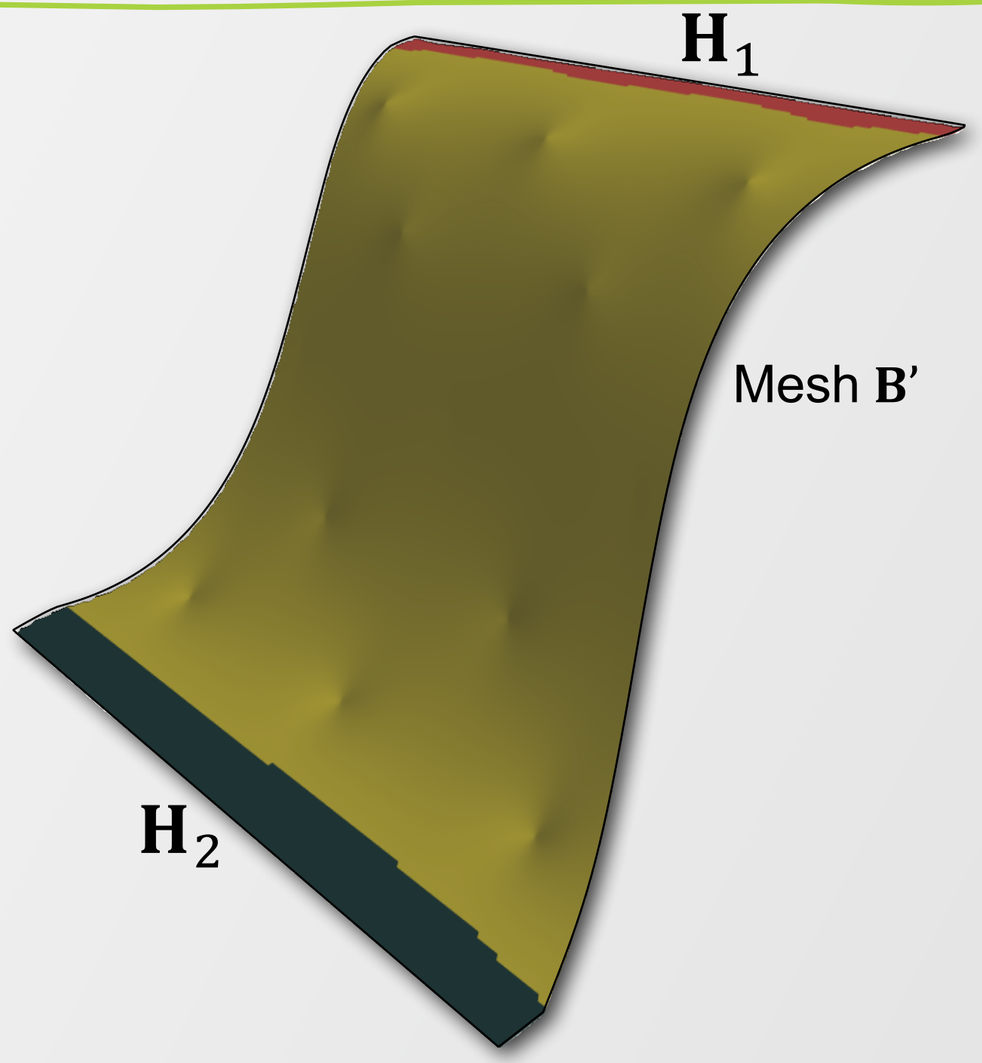
Step 4: Deform

- Manipulate the handles
- Solve for the deformed smooth mesh $\mathbf{B}'$
- Solve bi-Laplacian system with new positions

$$\min_{\mathbf{v}} \mathbf{v}^T \mathbf{L}_\omega \mathbf{M}^{-1} \mathbf{L}_\omega \mathbf{v}$$

$$\text{s.t.} \quad \mathbf{v}_{\mathbf{H_i}} = \mathbf{o}_{\mathbf{H_i}}$$

New positions

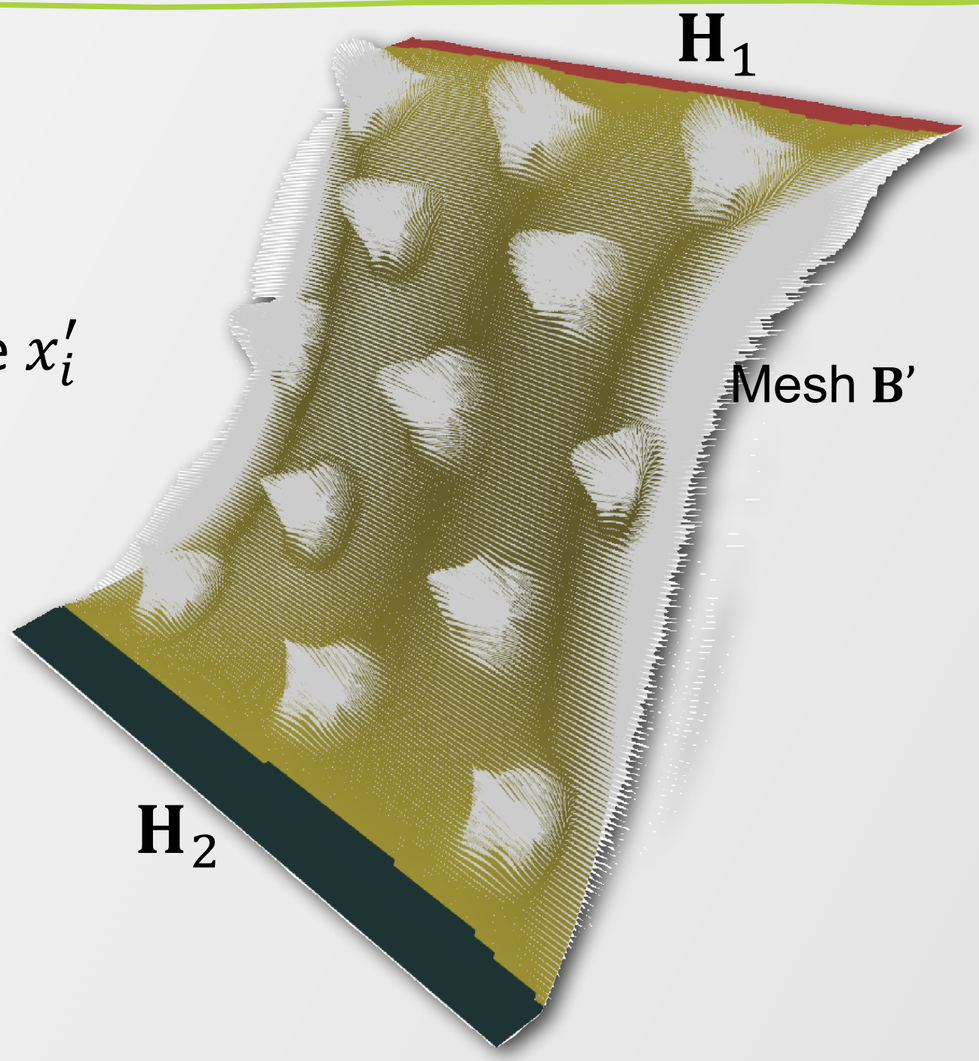


Step 5: Add local detail

- Compute for each v_i a local frame on $\mathbf{B}'$
  - Calculate normal n_i'
  - Use the same longest edge ij but on $\mathbf{B}'$ to define x_i'
  - Use cross product to compute y_i'
- Construct the transformed displacement d_i

$$d_i' = d_i^x x_i' + d_i^y y_i' + d_i^n n_i'$$

Same d_i's as before!

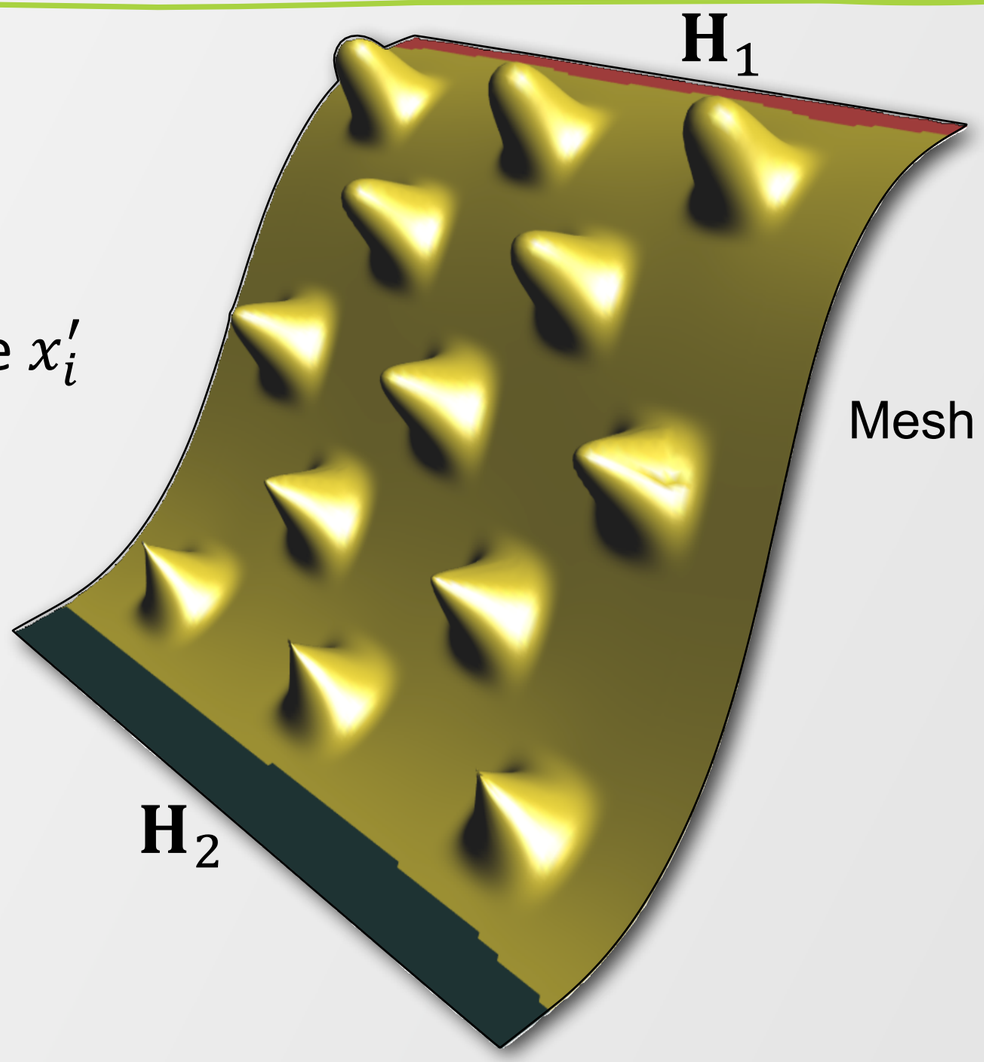


Step 5: Add local detail

- Compute for each v_i a local frame on $\mathbf{B}'$
  - Calculate normal n_i'
  - Use the same longest edge ij but on $\mathbf{B}'$ to define x_i'
  - Use cross product to compute y_i'
- Construct the transformed displacement d_i

$$d_i' = d_i^x x_i' + d_i^y y_i' + d_i^n n_i'$$

- Add them to $\mathbf{B}'$ to form $\mathbf{S}'$



How to solve a bi-laplacian system

$$\min_{\mathbf{v}} \mathbf{v}^{\mathrm{T}} \mathbf{L}_{\omega} \mathbf{M}^{-1} \mathbf{L}_{\omega} \mathbf{v}$$

$$\text{s.t.} \quad \mathbf{v}_{\mathbf{H_i}} = \mathbf{o}_{\mathbf{H_i}}$$

- Position are imposed as hard constraints
- This can be done using Lagrange multipliers, but in this assignment we will use substitution

$$A = \mathbf{L}_{\omega} \mathbf{M}^{-1} \mathbf{L}_{\omega} = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \qquad \begin{bmatrix} A_{ff} & A_{fc} \end{bmatrix} \begin{bmatrix} v_f \\ v_c \end{bmatrix} = 0 \Rightarrow A_{ff} v_f = -A_{fc} v_c$$

Pre-factoring the bi-laplacian

```
//PickingPlugin.h
Eigen::SimplicialCholesky<SparseMatrixType, Eigen::RowMajor > solver;
solver.compute (BiLaplacian_ii); // the interior part of the (almost) bi-laplacian
```

- It is the bottleneck of the solve
- Prefactorization is crucial to achieve real- time performance
  - Should only be performed when a different handle is selected for dragging

Provided Code

- Enables basic picking and dragging of handles
- You will fill it in with your deformation code
- Shortcuts:
  - ’S’: select
  - ’A’: accept selection
  - ALT+’T’: translation, ALT+‘R’: rotation

Provided Code

- Picking infrastructure

```
//for saving constrained vertices
//vertex-to-handle index, #V x1 (-1 if vertex is free)
Eigen::VectorXi handle_id(0,1);
//list of all vertices belonging to handles, #HV x1
Eigen::VectorXi handle_vertices(0,1);
//centroids of handle regions, #H x1
Eigen::MatrixXd handle_centroids(0,3);
//updated positions of handle vertices, #HV x3
Eigen::MatrixXd handle_vertex_positions(0,3);

//index of handle being moved
int moving_handle = -1;

//rotation and translation for the handle being moved
Eigen::Vector3f translation(0,0,0);
Eigen::Vector4f rotation(0,0,0,1.);
```

Provided Code

- While handle is being dragged

```
void get_new_handle_locations()
```

updates all handle vertex positions based on rotation and translation

stores them in handle_vertex_positions

- replace solve() with your code

```
bool solve(igl::Viewer& viewer, bool update_constraints)
{
  igl::slice_into(handle_vertex_positions, handle_vertices, 1, V);

  /* etc. update variables*/
 return true;
};
```

Questions?

Thank you!