Shape Modeling and Geometry Processing

Assignment 2 - Implicit Surfaces

Alexandre Binninger

alexandre.binninger@inf.ethz.ch

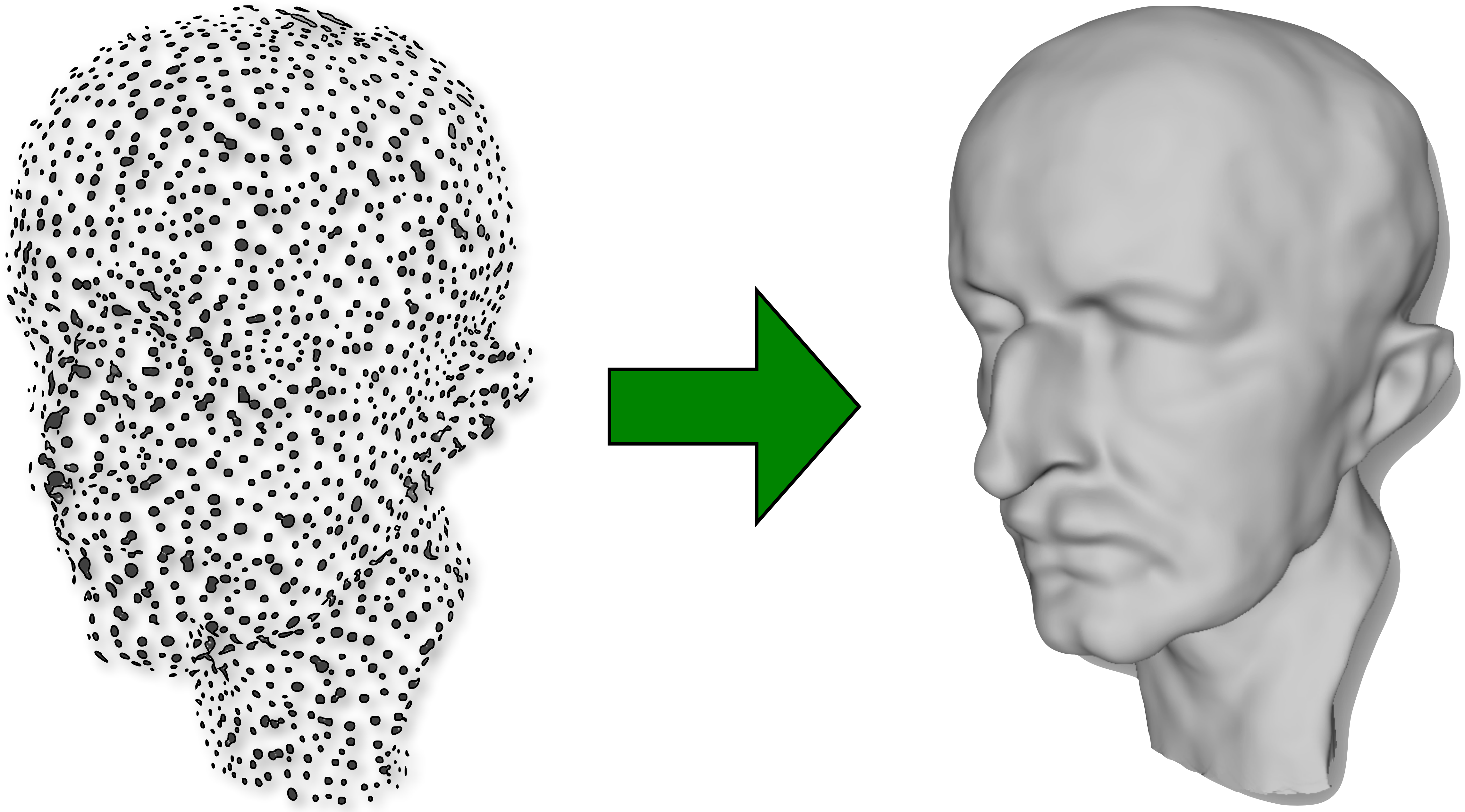
INTERACTIVE GEOMETRY LAB

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Assignments

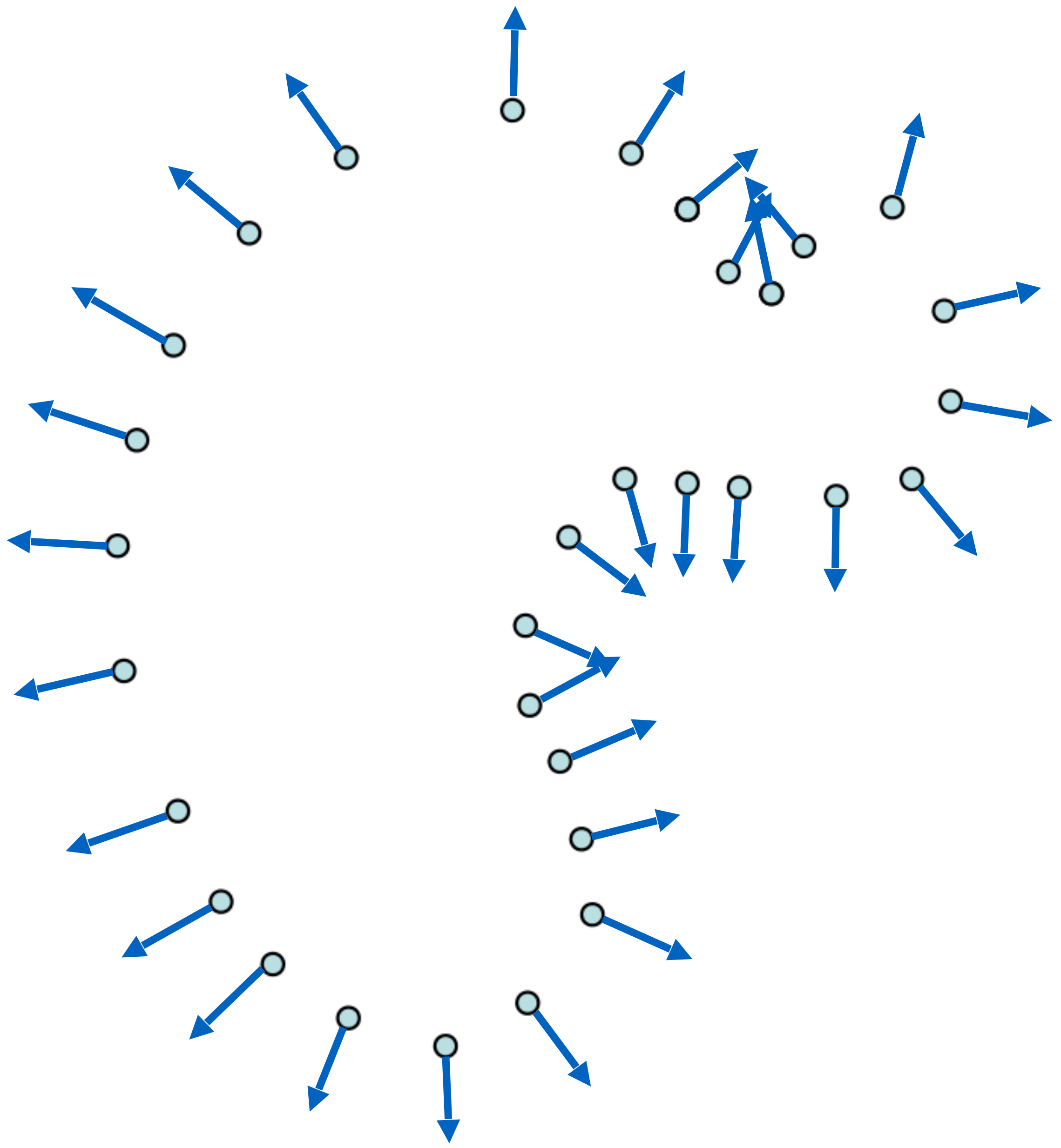
- Please regularly check the main repository for updates and new instruction:
  - https://github.com/eth-igl/GP2025-Assignments
- Questions can be asked via GitHub issues
  - Check previously asked questions before posting a new one

Implicit Surface Reconstruction



Implicit Surface Reconstruction

- A set of points given in 3D
- And normals per point

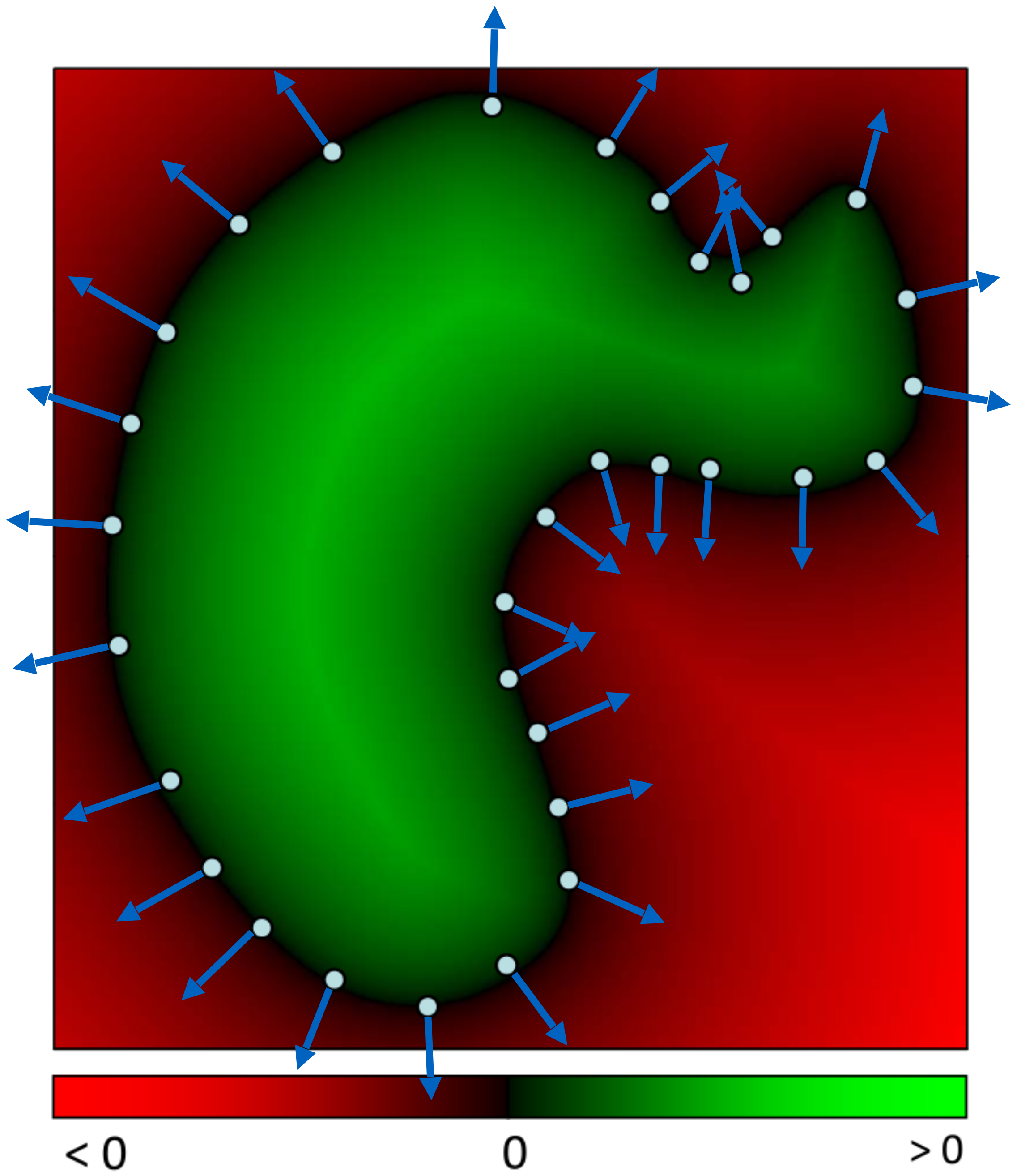


Implicit Surface Reconstruction

- Find a function (scalar-field)

$$f(x) : \mathbb{R}^3 \to \mathbb{R}$$

- Value < 0 outside
- Value > 0 inside

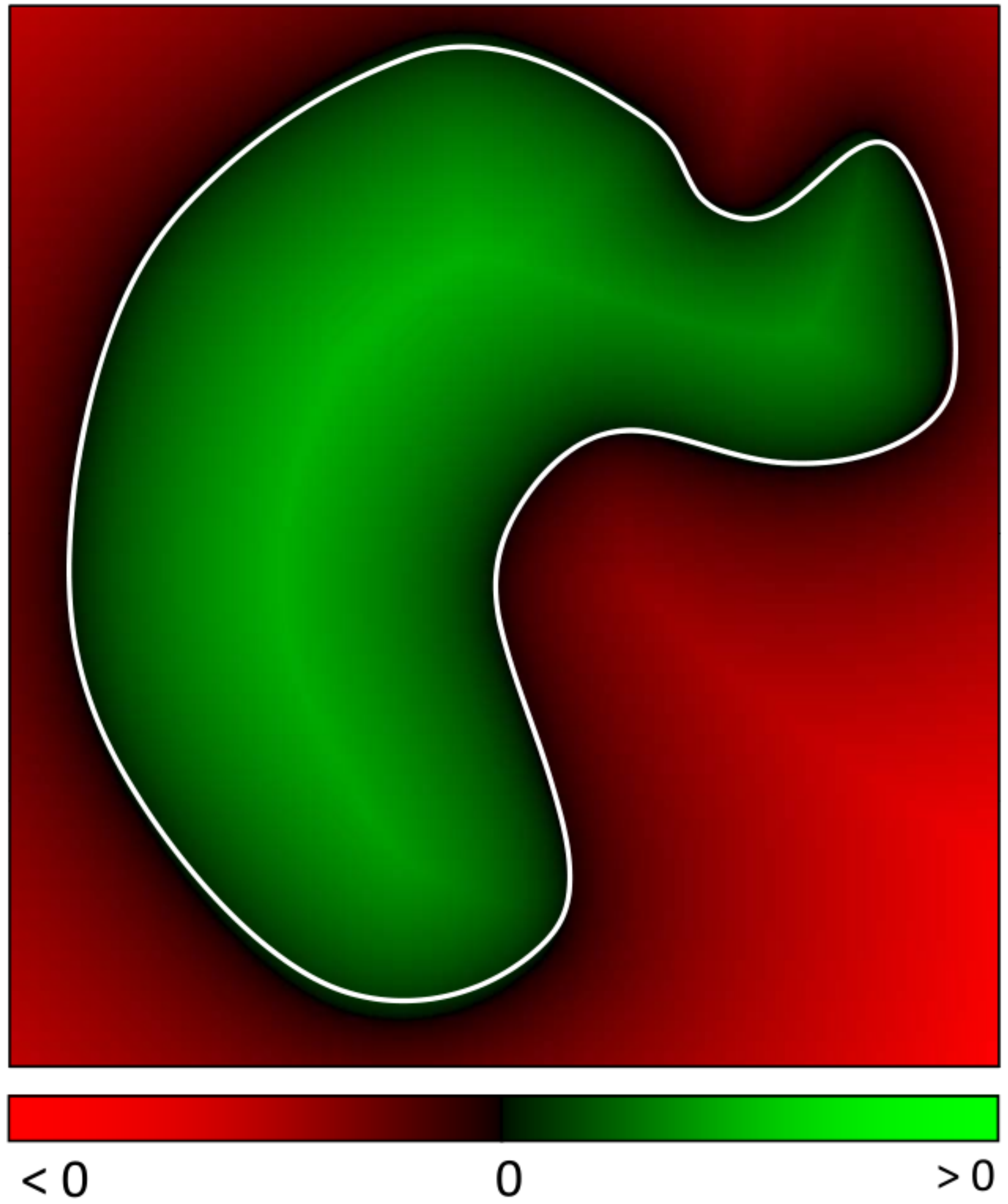


Implicit Surface Reconstruction

- Extract the zero-set

$$\{x : f(x) = 0\}$$

- Surface is guaranteed
  - 2-Manifold
  - No holes (watertight)



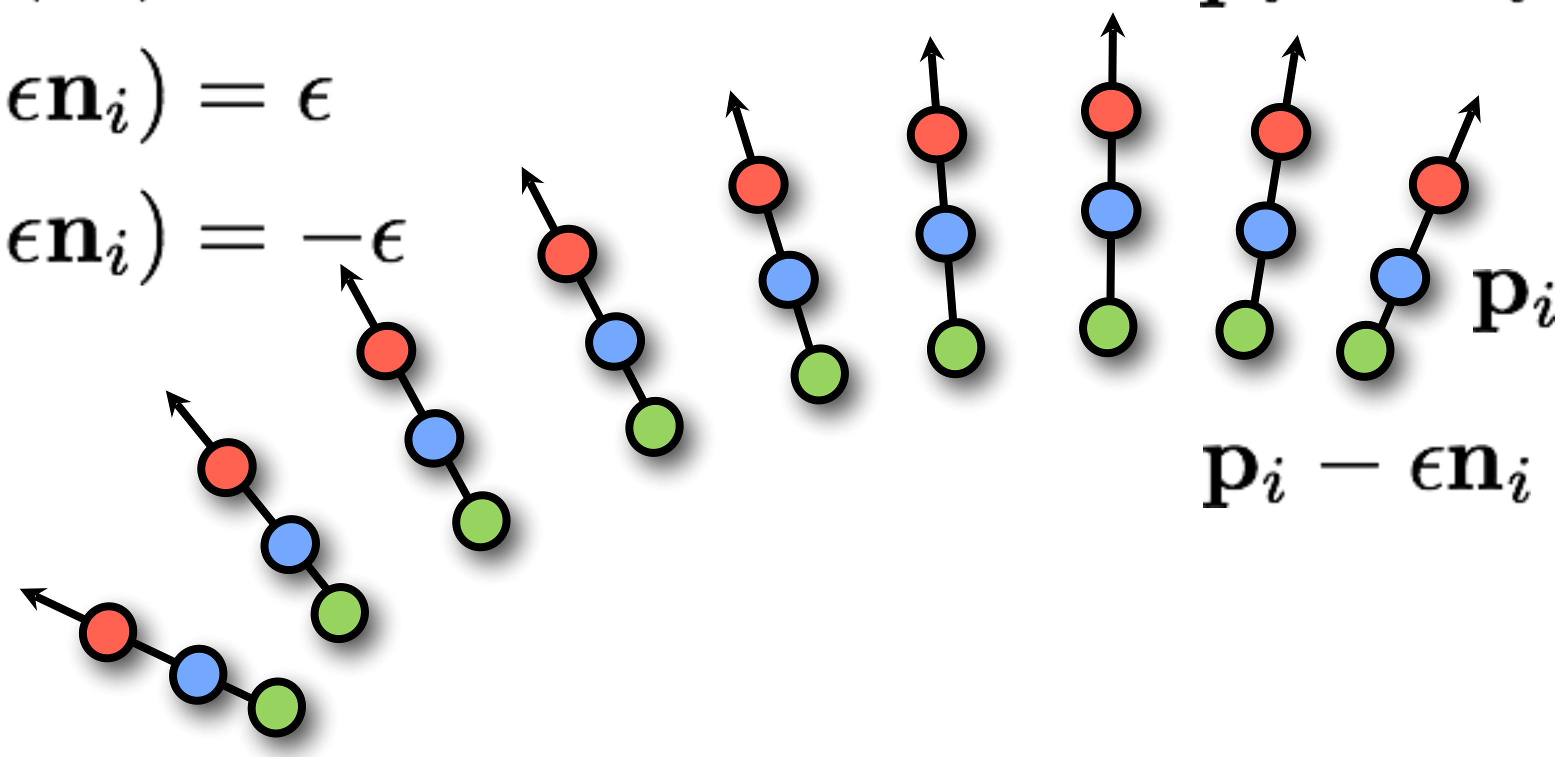
Assignment 2

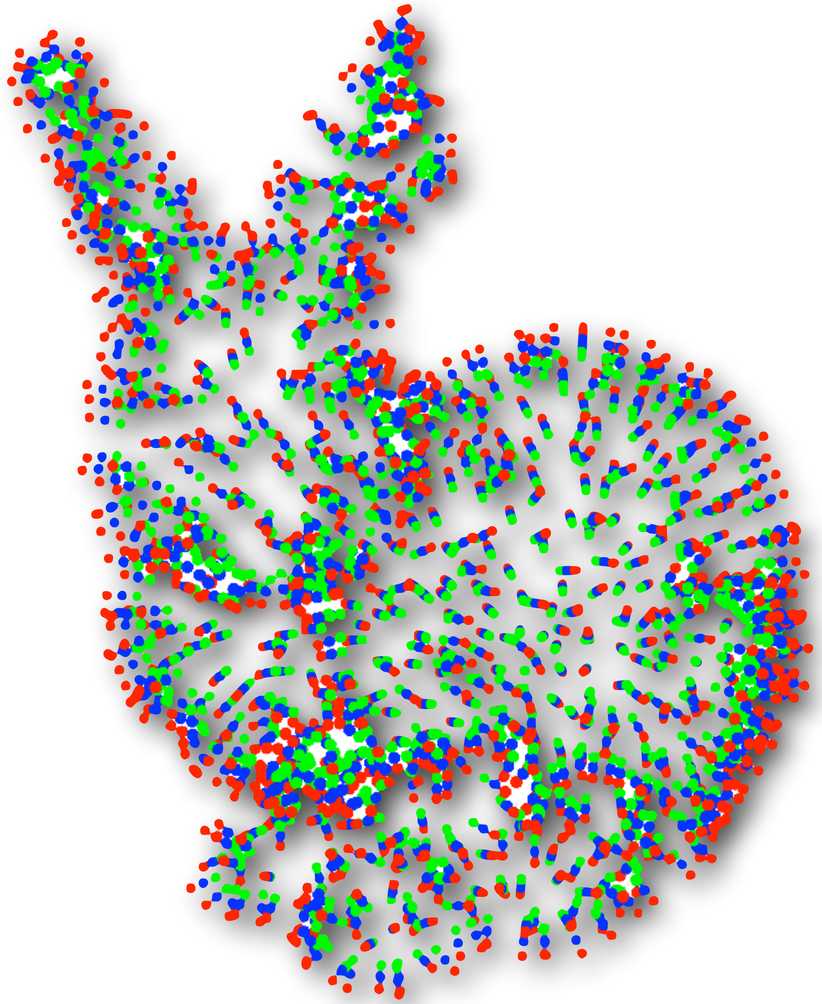
- Input:
  .off/*.obj* file
  with points and normals

Step 1: Build constraint set

Incorporate normal info width off-surface constraints:

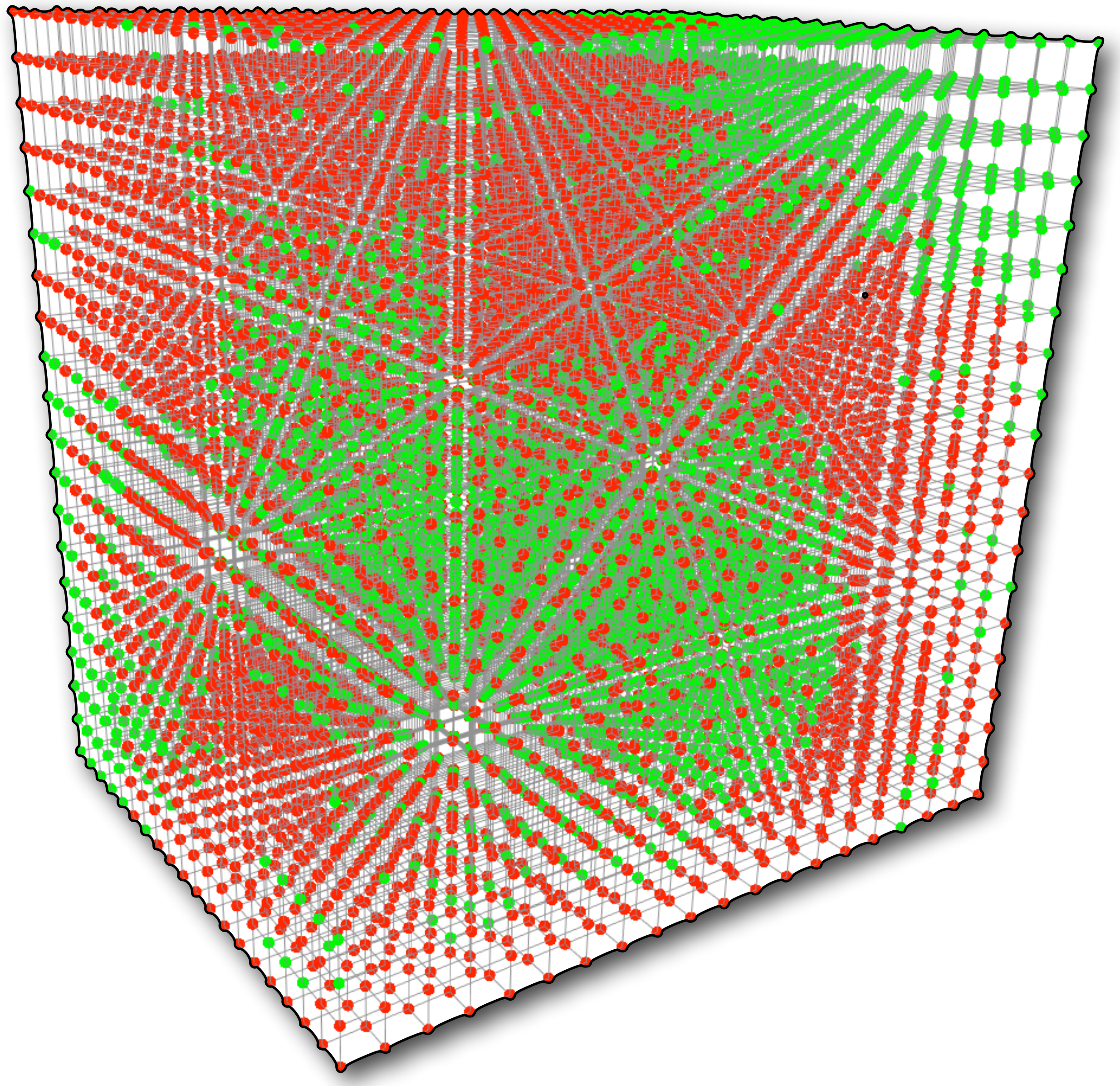
$$f(\mathbf{p}_i) = 0$$
$$f(\mathbf{p}_i + \epsilon\mathbf{n}_i) = \epsilon$$
$$f(\mathbf{p}_i - \epsilon\mathbf{n}_i) = -\epsilon$$





Step 2: Construct Interpolant

- Construct regular grid
- Compute nodal scalar field satisfying constraints (approximately)
- Method: **MLS (Moving Least Squares)**



The Least Squares Family

Please read: http://www.nealen.net/projects/mls/asapmls.pdf

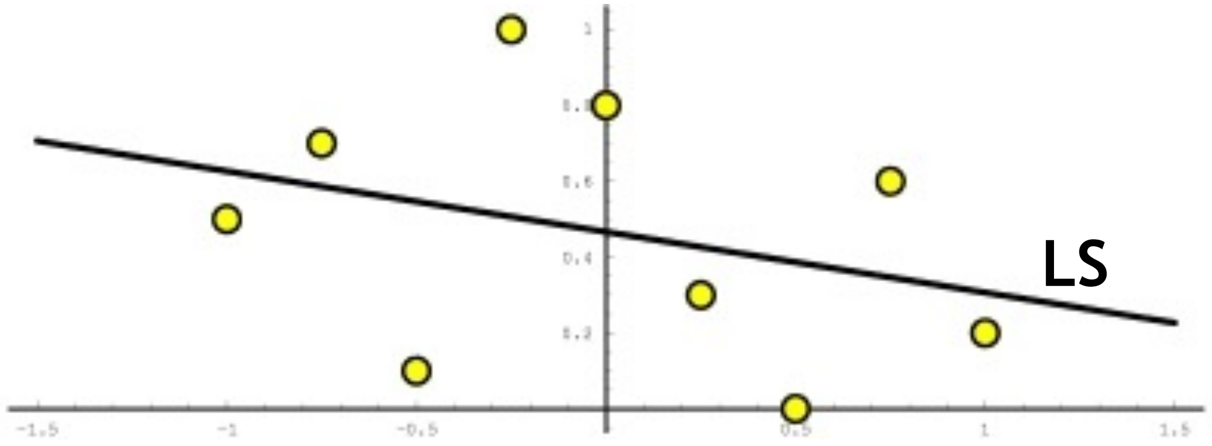
- LS $\min_{\mathbf{c}\in\mathbb{R}^k} \sum_i ||f(\mathbf{p}_i, \mathbf{c}) - f_i||^2$
- WLS $\min_{\mathbf{c}\in\mathbb{R}^k} \sum_i \theta(||\bar{\mathbf{p}} - \mathbf{p}_i||)\,||f(\mathbf{p}_i, \mathbf{c}) - f_i||^2$
- MLS $f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}_{\mathbf{x}}),$ $\min_{\mathbf{c}\in\mathbb{R}^k} \sum_i w(||\mathbf{x} - \mathbf{p}_i||)\,||f_{\mathbf{x}}(p_i, \mathbf{c}_{\mathbf{x}}) - f_i||^2$

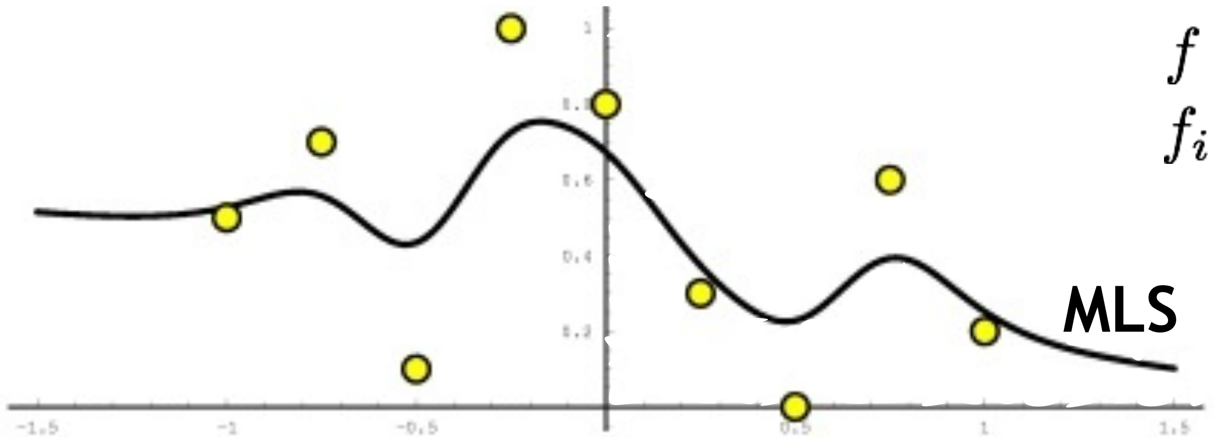
$\mathbf{p}_i$ sample points

$\mathbf{c}$ Coefficients (of polynomial)

f least-square approximation

f_i value of the desired function at $\mathbf{p}_i$





Basis function

$$\min_{\mathbf{c}\in\mathbb{R}^k} \sum_i ||f(\mathbf{p}_i, \mathbf{c}) - f_i||^2$$

- For this assignment, we'll use polynomial basis functions

$$f(\mathbf{p}_i, \mathbf{c}) = \sum_j b_j(\mathbf{p}_i) c_j = \mathbf{b}(\mathbf{p}_i)^T \mathbf{c}$$

- For polynomial degree 1 (a plane) we have:

$$\mathrm{b}(p_i)^T = [1, x, y, z]$$

$$c = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- For polynomial degree 2 we have:

$$\mathrm{b}(p_i)^T = [1, x, y, z, xy, xz, yz, x^2, y^2, z^2]$$

Standard LS reconstruction

- Standard least-squares fit

$$\min_{\mathbf{c}\in\mathbb{R}^k} \sum_i ||f(\mathbf{p}_i, \mathbf{c}) - f_i||^2$$

linear algebra reminder:
to solve $min_{x\in\mathbb{R}^n}||Ax - b||^2$

solve $A^T A x = A^T b$

- Solve a overdetermined linear system (use Eigen library)

$$\begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

$$\mathbf{b}(\mathbf{p}_i) = [1, x_i, y_i, z_i]$$

Standard LS reconstruction

- Standard least-squares fit

$$\min_{\mathbf{c}\in\mathbb{R}^k} \sum_i ||f(\mathbf{p}_i, \mathbf{c}) - f_i||^2$$

linear algebra reminder:
to solve $min_{x\in\mathbb{R}^n} ||Ax - b||^2$

solve $A^T A x = A^T b$

- Solve a overdetermined linear system (use Eigen library)

$$\begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

$$\mathbf{b}(\mathbf{p}_i) = [1, x_i, y_i, z_i]$$

polynomial basis

Standard LS reconstruction

- Standard least-squares fit

$$\min_{\mathbf{c}\in\mathbb{R}^k} \sum_i ||f(\mathbf{p}_i, \mathbf{c}) - f_i||^2$$

linear algebra reminder:
to solve $min_{x\in\mathbb{R}^n} ||Ax - b||^2$

solve $A^T A x = A^T b$

- Solve a overdetermined linear system (use Eigen library)

$$\begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} \qquad \mathbf{b}(\mathbf{p}_i) = [1, x_i, y_i, z_i]$$

desired function values

ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Standard LS reconstruction

- Standard least-squares fit

$$\min_{\mathbf{c}\in\mathbb{R}^k} \sum_i ||f(\mathbf{p}_i, \mathbf{c}) - f_i||^2$$

linear algebra reminder:
to solve $min_{x\in\mathbb{R}^n}||Ax - b||^2$

solve $A^T A x = A^T b$

- Solve a overdetermined linear system (use Eigen library)(SVD, QR, or normal equations))

$$\begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} \qquad \mathbf{b}(\mathbf{p}_i) = [1, x_i, y_i, z_i]$$

coefficients of polynomial basis

MLS reconstruction

- MLS fit

$$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}_{\mathbf{x}}), \qquad \min_{\mathbf{c}\in\mathbb{R}^k} \sum_i w(\|\mathbf{x}-\mathbf{p}_i\|)^2 \|f_{\mathbf{x}}(p_i, \mathbf{c}_{\mathbf{x}}) - f_i\|^2$$

$$\begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x},\mathbf{p}_N) \end{bmatrix} \begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c}(\mathbf{x}) = \begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x},\mathbf{p}_N) \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

MLS reconstruction

- MLS fit

$$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}_{\mathbf{x}}), \qquad \min_{\mathbf{c}\in\mathbb{R}^k} \sum_i w(\|\mathbf{x} - \mathbf{p}_i\|)^2 \|f_{\mathbf{x}}(p_i, \mathbf{c}_{\mathbf{x}}) - f_i\|^2$$

$$\begin{bmatrix} w(\mathbf{x}, \mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x}, \mathbf{p}_N) \end{bmatrix} \begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c}(\mathbf{x}) = \begin{bmatrix} w(\mathbf{x}, \mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x}, \mathbf{p}_N) \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

proximity weights   proximity weights

$$w(\mathbf{x}, \mathbf{p}_i) = f_w(\|\mathbf{x} - \mathbf{p}_i\|)$$

f_w : weight function

MLS reconstruction

- MLS fit

$$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}_{\mathbf{x}}), \qquad \min_{\mathbf{c}\in\mathbb{R}^k} \sum_i w(\|\mathbf{x} - \mathbf{p}_i\|)^2 \|f_{\mathbf{x}}(p_i, \mathbf{c}_{\mathbf{x}}) - f_i\|^2$$

$$\begin{bmatrix} w(\mathbf{x}, \mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x}, \mathbf{p}_N) \end{bmatrix} \begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \boxed{\mathbf{c}(\mathbf{x})} = \begin{bmatrix} w(\mathbf{x}, \mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x}, \mathbf{p}_N) \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

  - The coefficients $c(x)$ are local and need to be recomputed for every x (x is the coordinate of the grid node)

MLS reconstruction solution

- MLS fit $E_{MLS} = \frac{1}{2}\sum w_i^2 (b_i^T c - f_i)^2 = \frac{1}{2}\sum (w_i b_i^T c - w_i f_i)^2$

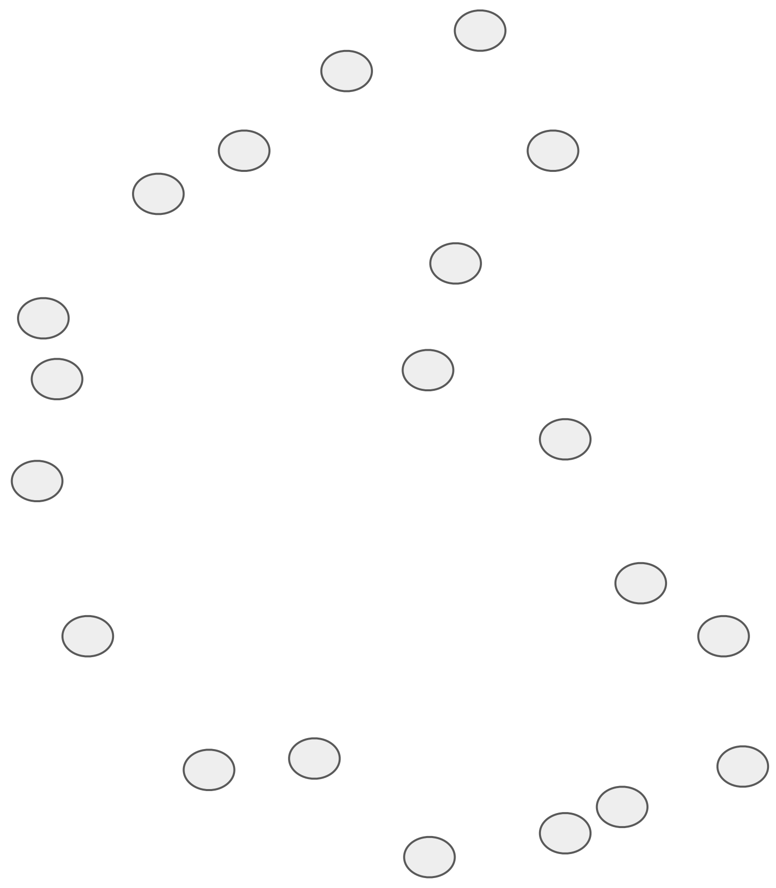
The over-constrained system: $WBc = Wf$

$$\begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x},\mathbf{p}_N) \end{bmatrix} \begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c}(\mathbf{x}) = \begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x},\mathbf{p}_N) \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

The normal equations: $B^T W^2 B c = B^T W^2 f$

MLS reconstruction

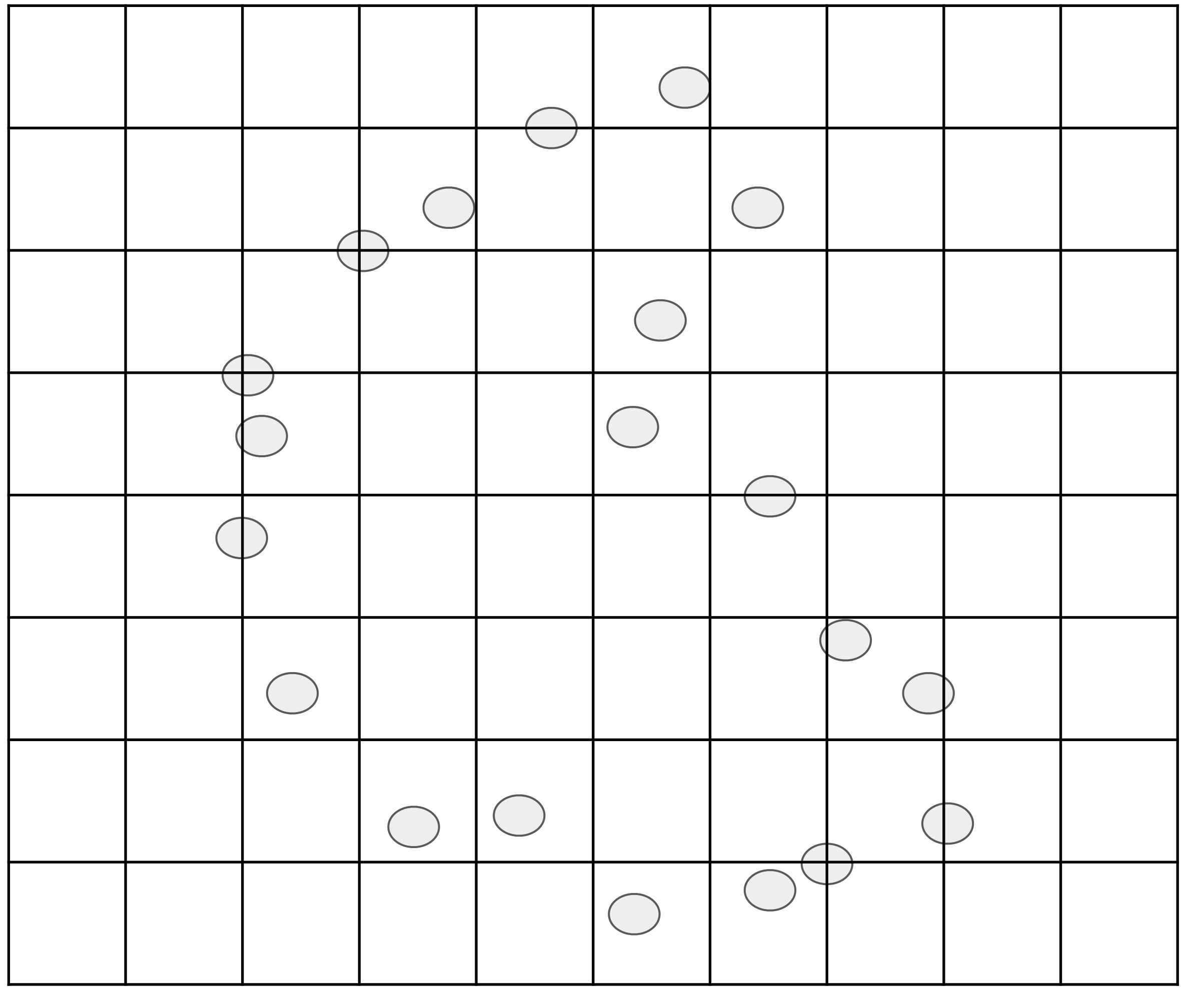
$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}_{\mathbf{x}}),$ $\quad \min_{\mathbf{c} \in \mathbb{R}^k} \sum_i w(\|\mathbf{x} - \mathbf{p}_i\|)^2 \| f_{\mathbf{x}}(p_i, \mathbf{c}_{\mathbf{x}}) - f_i \|^2$



The input points p_i

MLS reconstruction

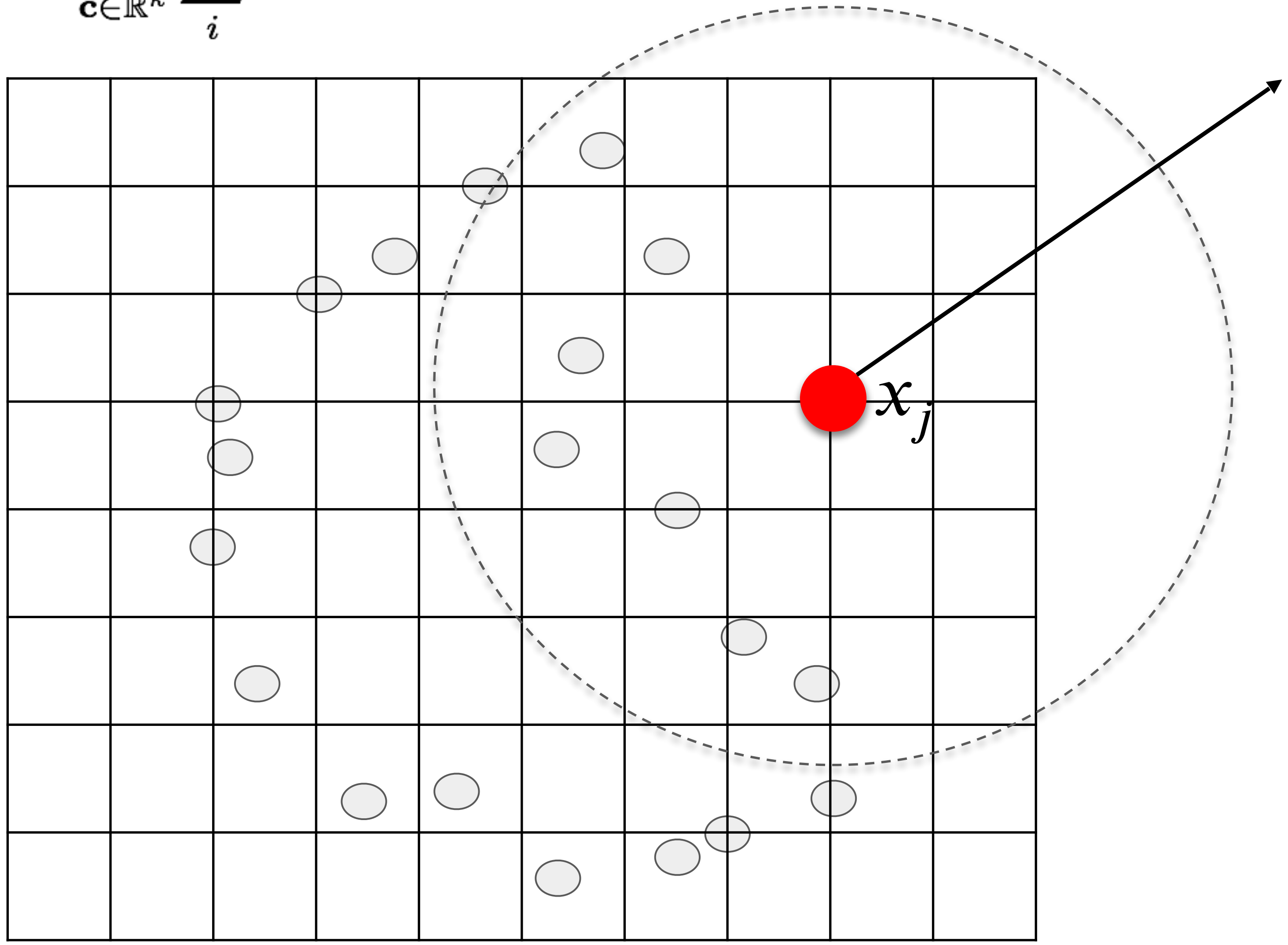
$$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}_{\mathbf{x}}), \qquad \min_{\mathbf{c} \in \mathbb{R}^k} \sum_i w(\|\mathbf{x} - \mathbf{p}_i\|)^2 \| f_{\mathbf{x}}(p_i, \mathbf{c}_{\mathbf{x}}) - f_i \|^2$$



We construct regular grid points x_j over input points p_i

MLS reconstruction

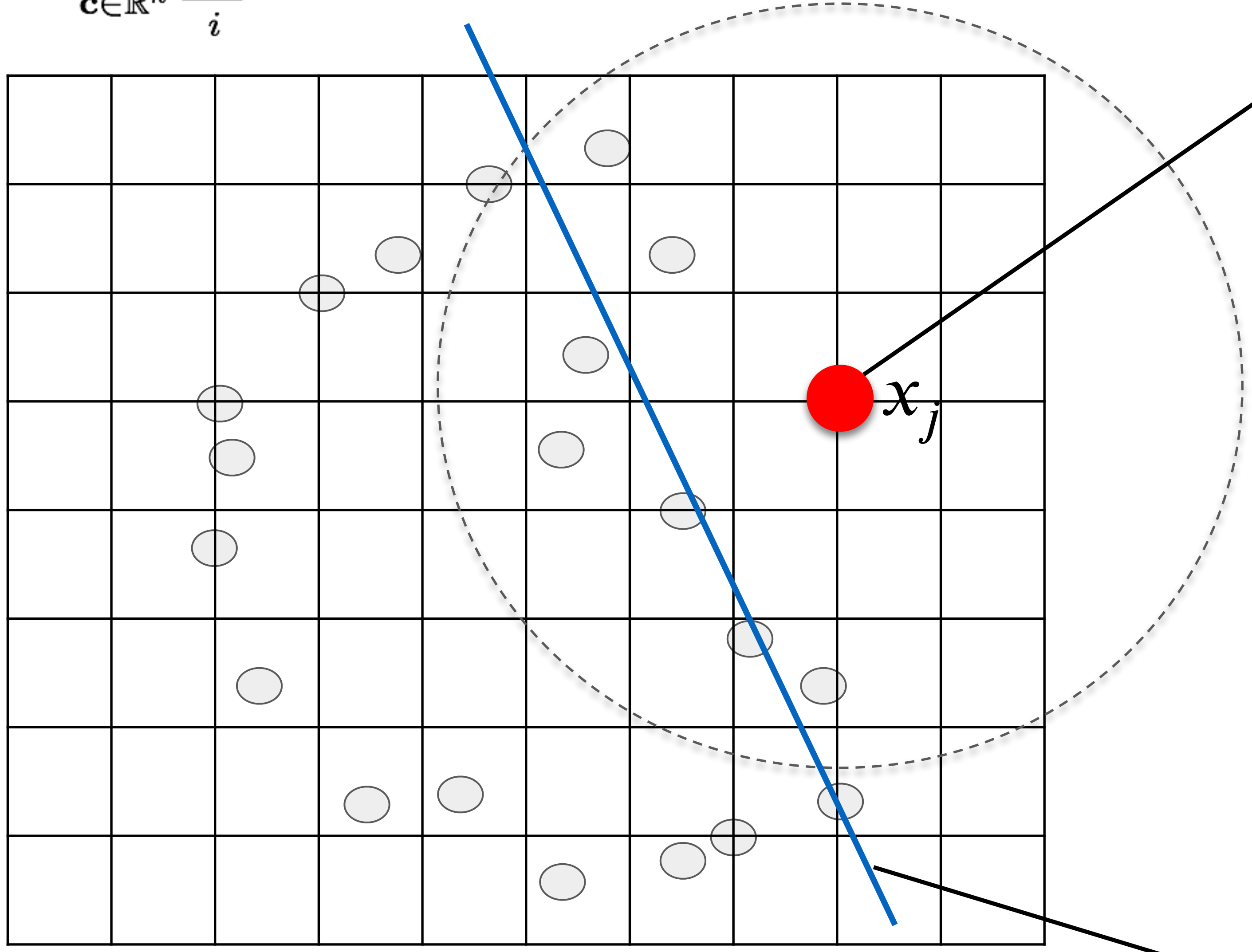
$$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}_{\mathbf{x}}), \qquad \min_{\mathbf{c} \in \mathbb{R}^k} \sum_i w(\|\mathbf{x} - \mathbf{p}_i\|)^2 \|f_{\mathbf{x}}(p_i, \mathbf{c}_{\mathbf{x}}) - f_i\|^2$$



- The current grid point x_j we are considering
- We need to optimize the coefficients $c(x_j)$
- The weighting function $w(x_j, p_i)$ defines the local neighborhood

MLS reconstruction

$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}_{\mathbf{x}}),\qquad \min_{\mathbf{c}\in\mathbb{R}^k} \sum_i w(\|\mathbf{x} - \mathbf{p}_i\|)^2 \|f_{\mathbf{x}}(p_i, \mathbf{c}_{\mathbf{x}}) - f_i\|^2$

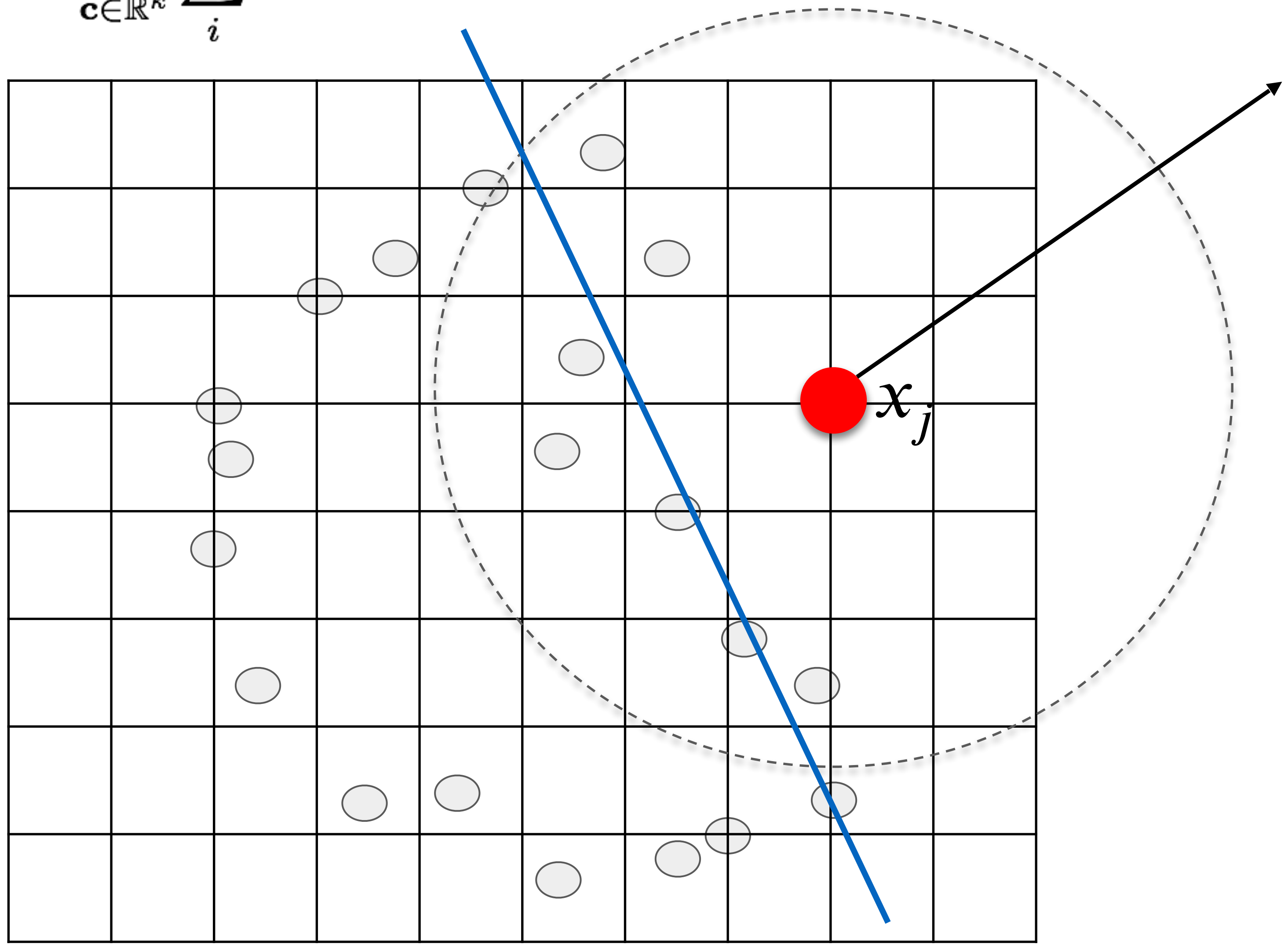


The current grid point x_j we are considering
We need to optimize the coefficients $c(x_j)$

The polynomial (degree 1) defined by the optimized coefficients $c(x_j)$

MLS reconstruction

$$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}_{\mathbf{x}}), \qquad \min_{\mathbf{c}\in\mathbb{R}^k} \sum_i w(\|\mathbf{x} - \mathbf{p}_i\|)^2 \| f_{\mathbf{x}}(p_i, \mathbf{c}_{\mathbf{x}}) - f_i \|^2$$



Now $c(x_j)$ is obtained,

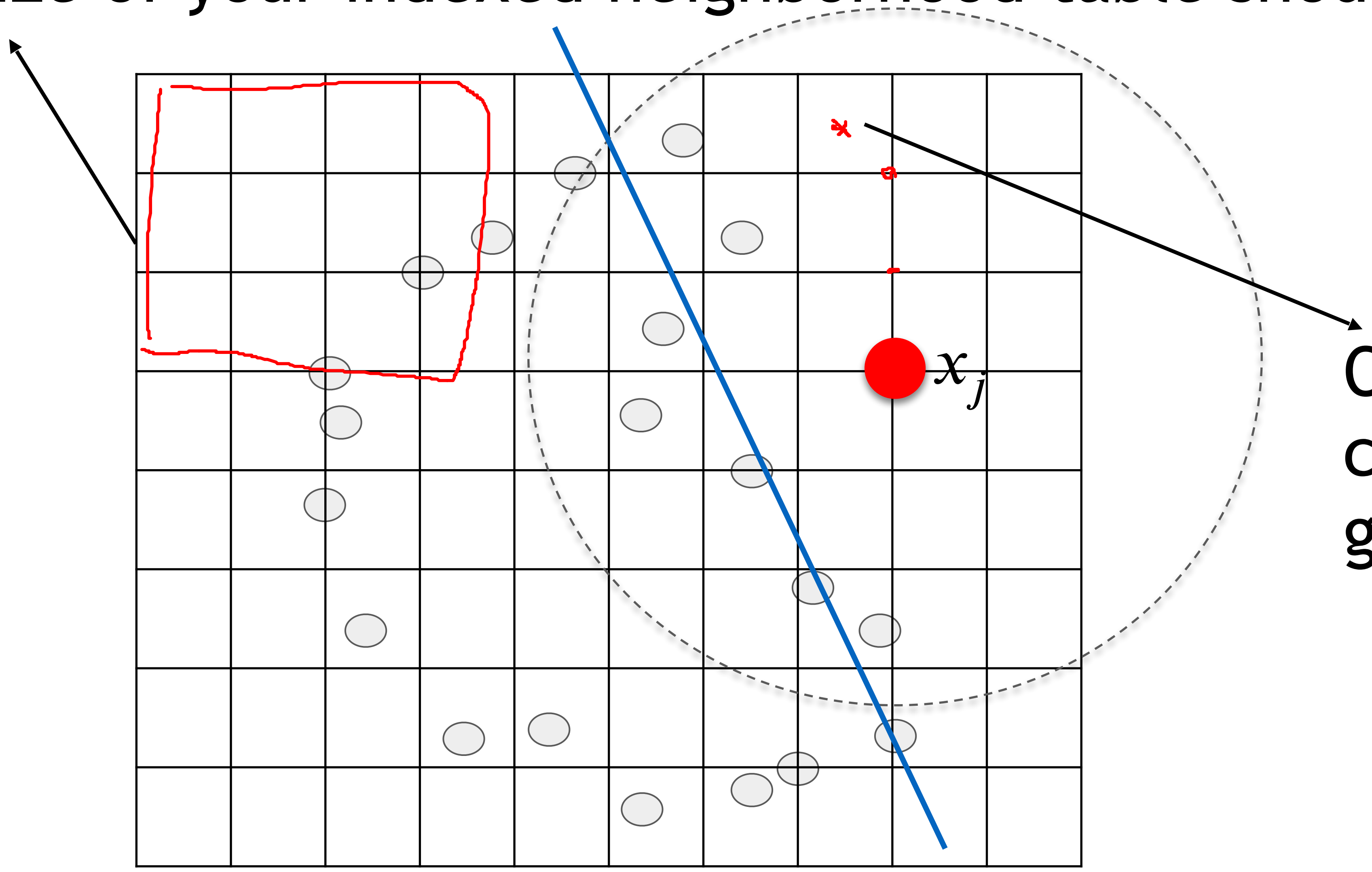
the SDF value of grid point x_j is

$$f\left(x_j\right) = b\left(x_j\right)^T c(x_j)$$

Repeat and compute the SDF values for all grid points

MLS reconstruction

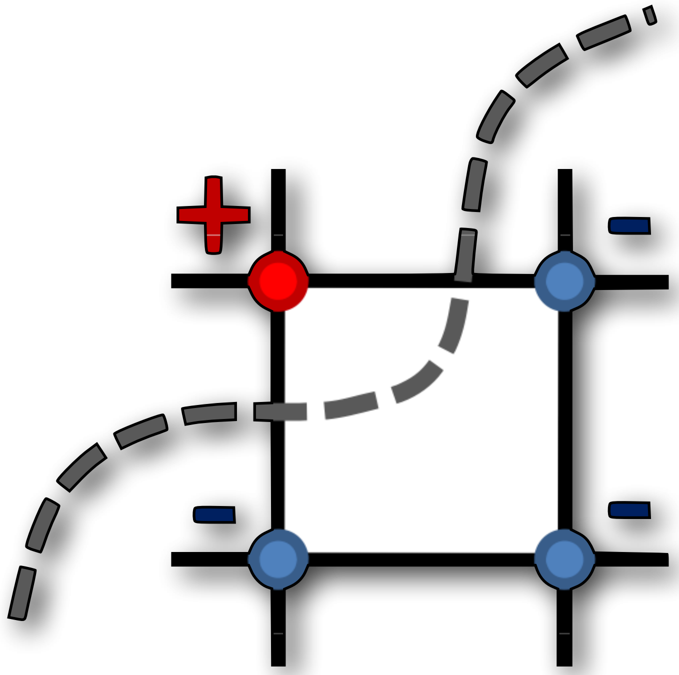
The size of your indexed neighborhood table should be larger than the grid size

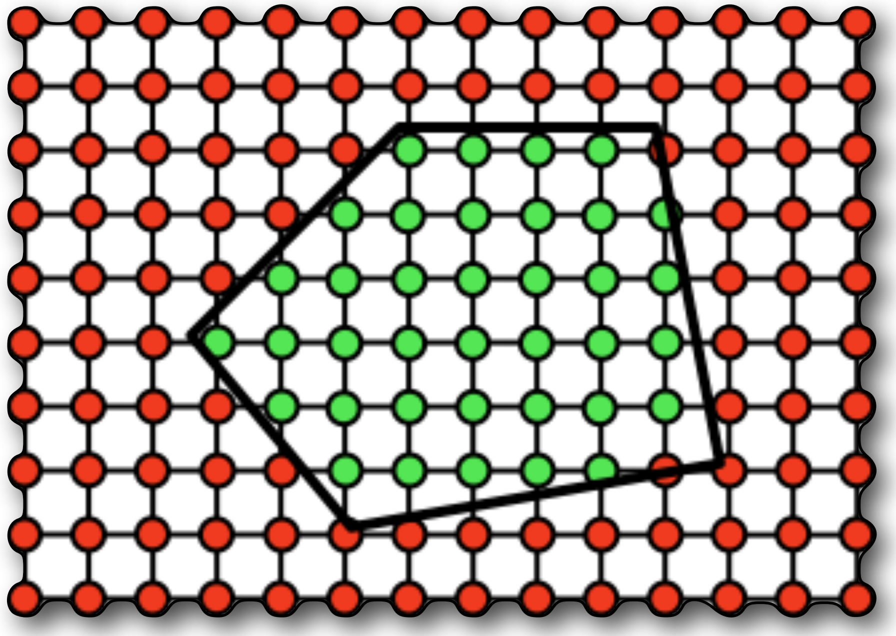


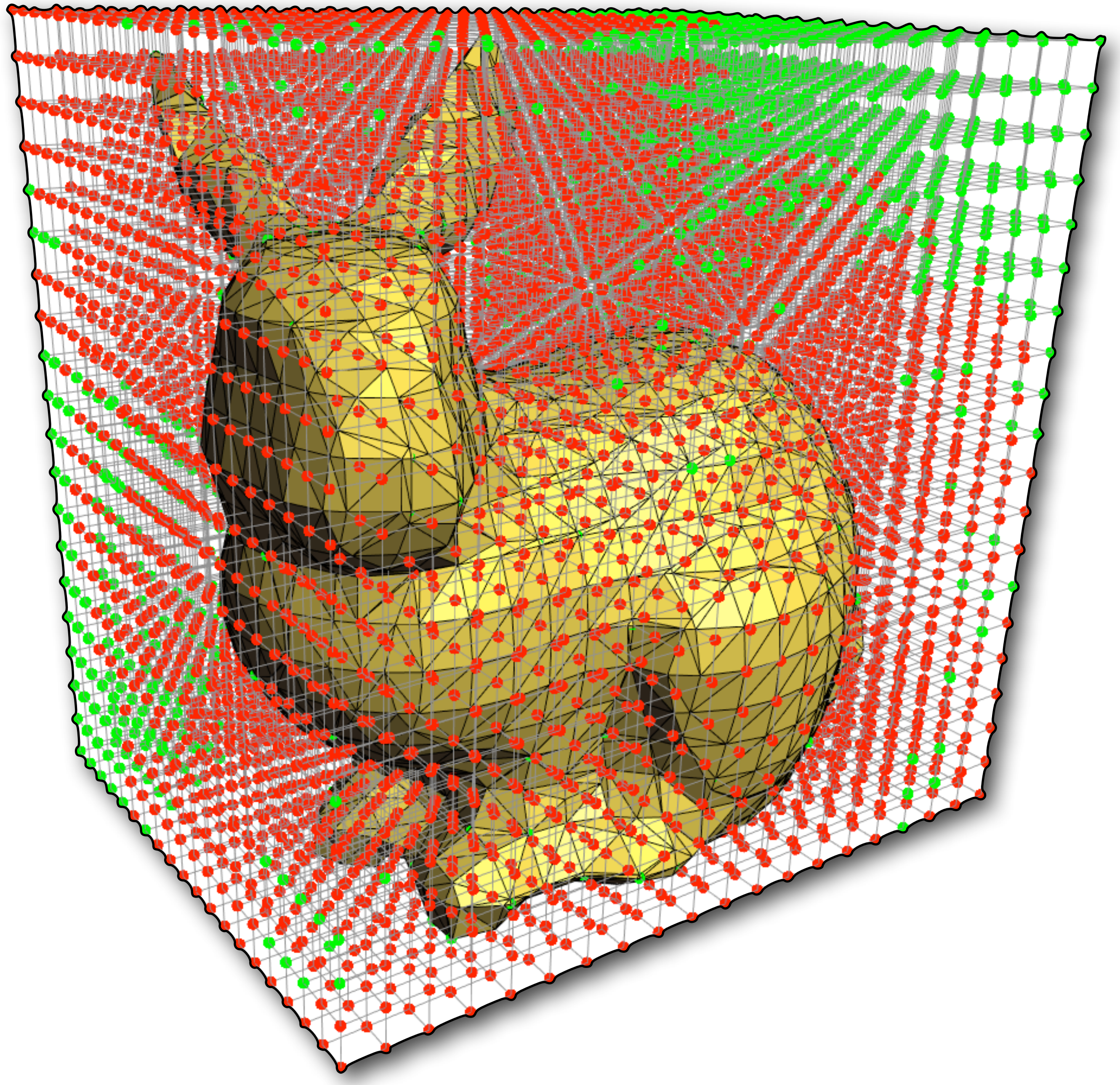
One should not work on the centers, but the corners of each grid

Step 3: Marching Cubes

- Use the **marching cubes** algorithm to extract the grid function's zero isosurface
- Use igl::copyleft::marching_cubes



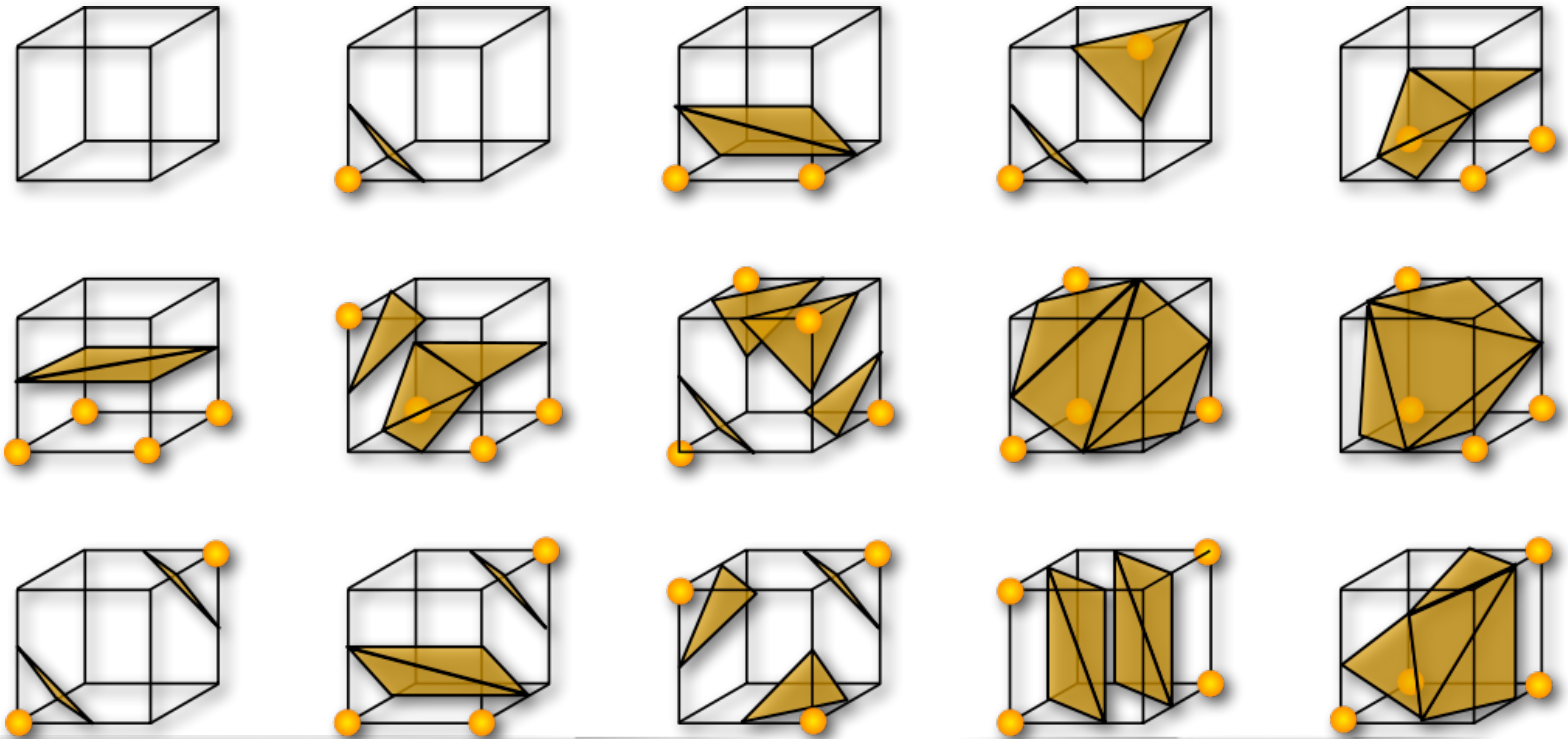




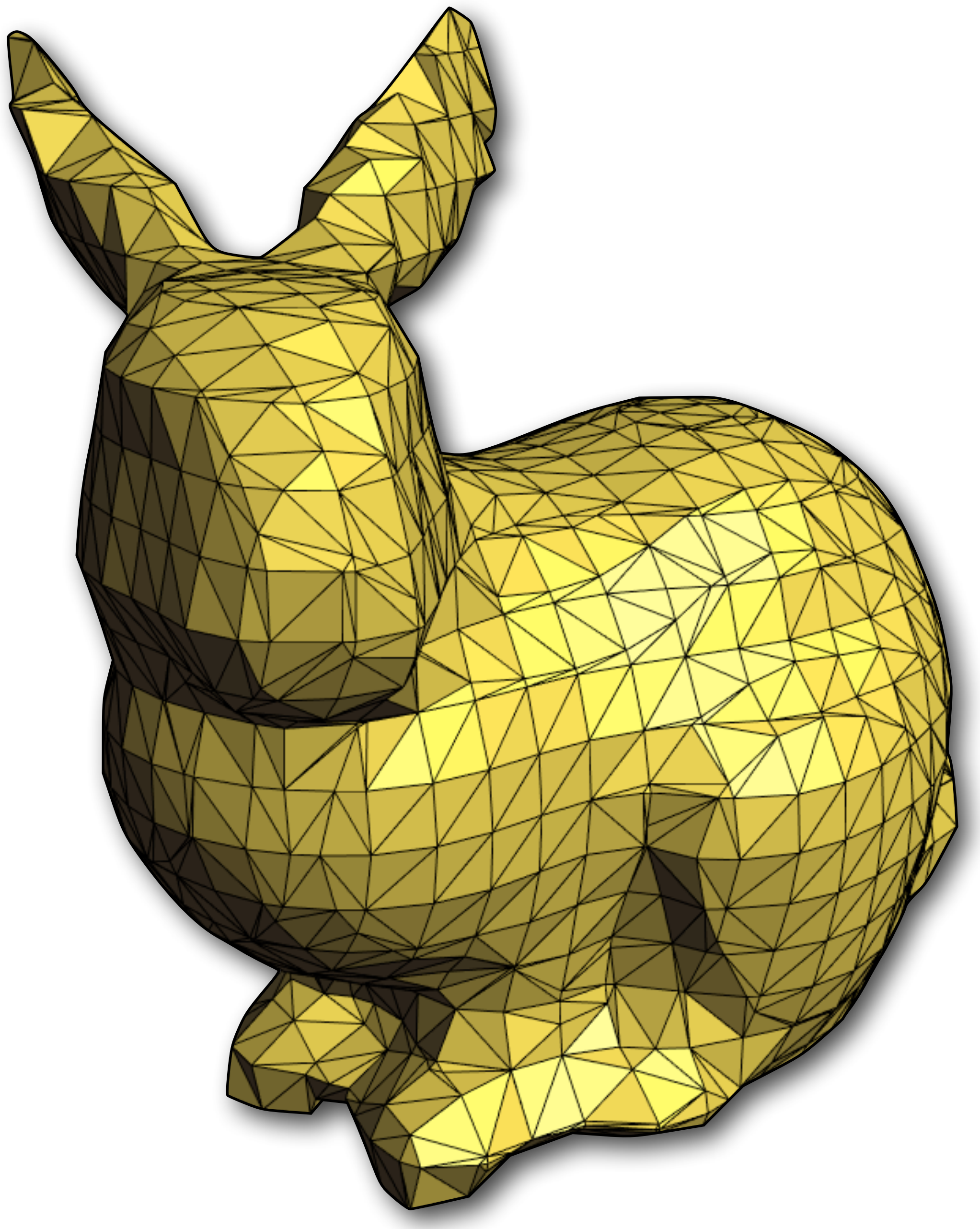
ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Step 3: Marching Cubes

- Look up triangles to be created in each grid cell, based on corner values:

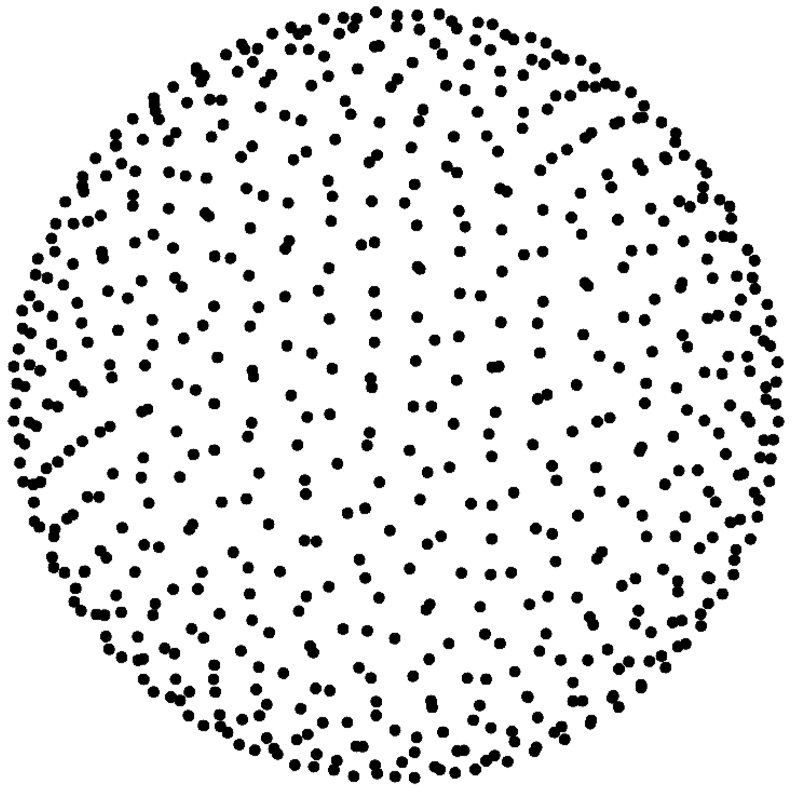


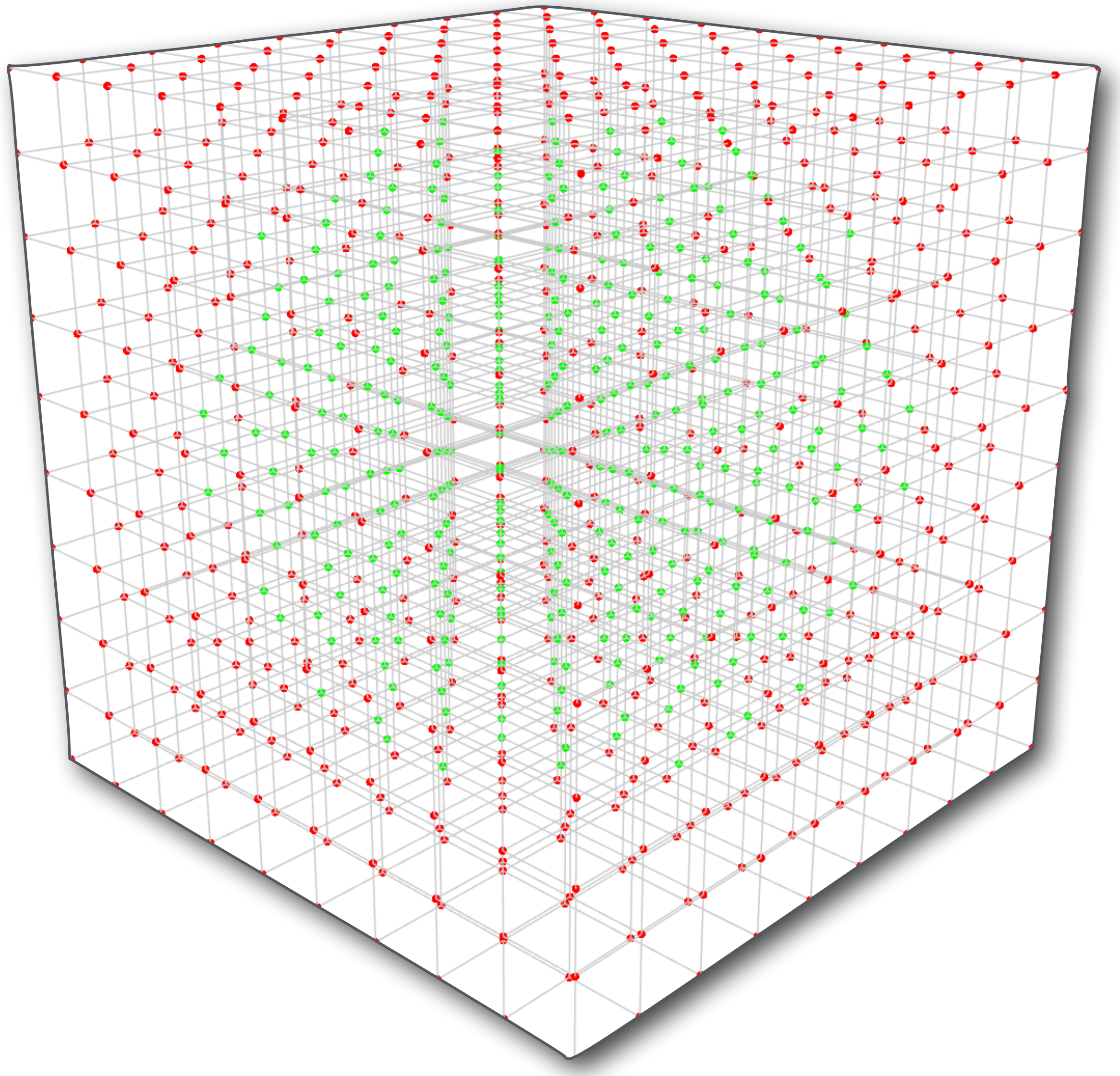
Final Mesh

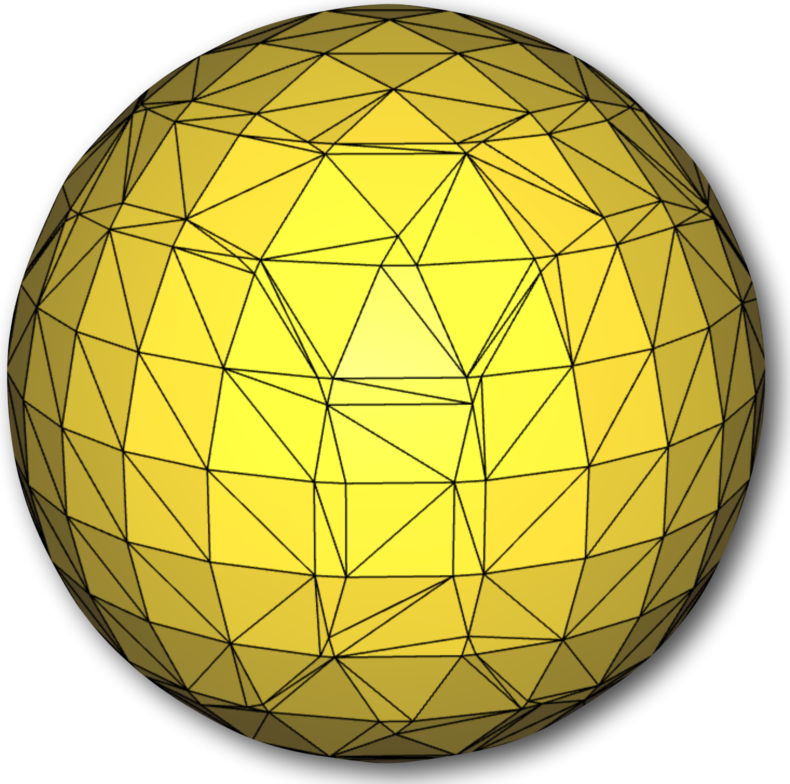


Provided Example

- Implements pipeline but uses analytic signed distance function for sphere in place of MLS







Provided Example: Implicit Sphere

- Step 1: Compute an axis-aligned bounding box

```
// Grid bounds: axis-aligned bounding box
Eigen::RowVector3d bb_min, bb_max;
bb_min = P.colwise().minCoeff();
bb_max = P.colwise().maxCoeff();

// Bounding box dimensions
Eigen::RowVector3d dim = bb_max - bb_min;
```

Provided Example: Implicit Sphere

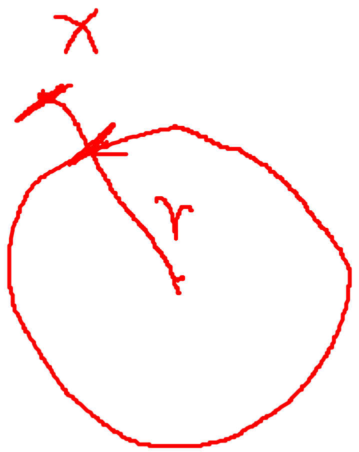
- Step 2: construct a grid over the bounding box

```
// Grid spacing
const double dx = dim[0] / (double)(resolution - 1);
const double dy = dim[1] / (double)(resolution - 1);
const double dz = dim[2] / (double)(resolution - 1);
// 3D positions of the grid points -- see slides or marching_cubes.h for ordering
grid_points.resize(resolution * resolution * resolution, 3);
// Create each gridpoint
for (unsigned int x = 0; x < resolution; ++x) {
    for (unsigned int y = 0; y < resolution; ++y) {
        for (unsigned int z = 0; z < resolution; ++z) {
            // Linear index of the point at (x,y,z)
            int index = x + resolution * (y + resolution * z);
            // 3D point at (x,y,z)
            grid_points.row(index) = bb_min + Eigen::RowVector3d(x * dx, y * dy, z * dz);
        }
    }
}
```

Provided Example: Implicit Sphere

- Step 3: Fill grid with the values of the implicit function

```
// Scalar values of the grid points (the implicit function values)
grid_values.resize(resolution * resolution * resolution);

// Evaluate sphere's signed distance function at each gridpoint.
for (unsigned int x = 0; x < resolution; ++x) {
    for (unsigned int y = 0; y < resolution; ++y) {
        for (unsigned int z = 0; z < resolution; ++z) {
            // Linear index of the point at (x,y,z)
            int index = x + resolution * (y + resolution * z);

            // Value at (x,y,z) = implicit function for the sphere
            grid_values[index] = (grid_points.row(index) - center).norm() - radius;
        }
    }
}
```



Provided Example: Implicit Sphere

- Step 4: run marching cubes

```
// Run marching cubes
igl::copyleft::marching_cubes(grid_values, grid_points, resolution, resolution, resolution, V, F);
```

input: implicit function values at grid points

Provided Example: Implicit Sphere

- Step 4: run marching cubes

```
// Run marching cubes
igl::copyleft::marching_cubes(grid_values, grid_points, resolution, resolution, resolution, V, F);
```

input: grid point positions

Provided Example: Implicit Sphere

- Step 4: run marching cubes

```
// Run marching cubes
igl::copyleft::marching_cubes(grid_values, grid_points, resolution, resolution, resolution, V, F);
```

input: grid size (x, y, z)

Provided Example: Implicit Sphere

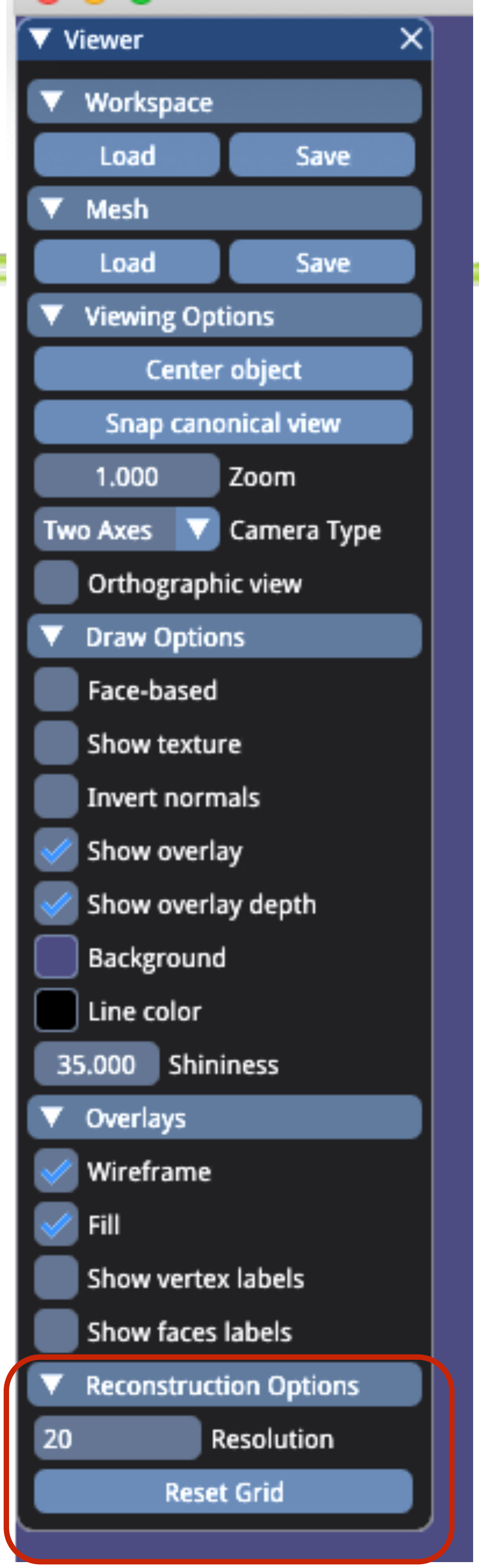
- Step 4: run marching cubes

```
// Run marching cubes
igl::copyleft::marching_cubes(grid_values, grid_points, resolution, resolution, resolution, V, F);
```

output: vertices and faces

ImGui

- IGL Viewer uses ImGui: https://github.com/ocornut/imgui
- You'll need to add widgets to configure additional variables.

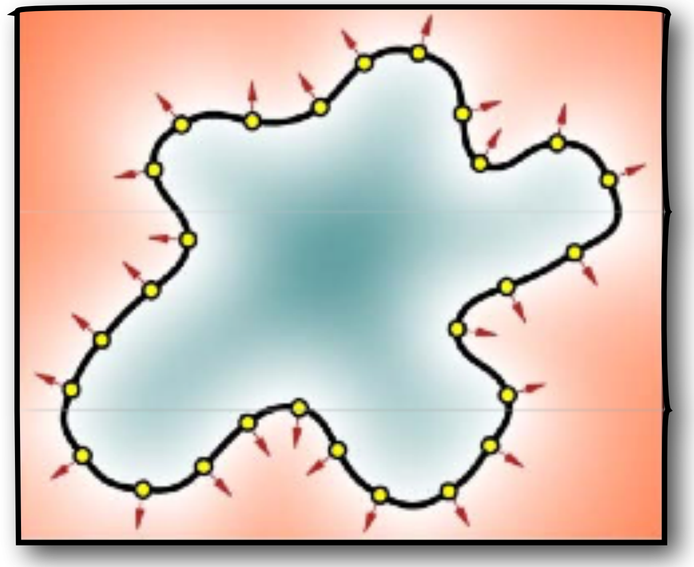


ImGui: Adding Settings

```
igl::opengl::glfw::imgui::ImGuiMenu menu;
viewer.plugins.push_back(&menu);

menu.callback_draw_viewer_menu = [&]()
{
  // Draw parent menu content
  menu.draw_viewer_menu();

  // Add new group
  if (ImGui::CollapsingHeader("Reconstruction Options", ImGuiTreeNodeFlags_DefaultOpen))
  {
    // Expose variable directly ...
    ImGui::InputInt("Resolution", &resolution, 0, 0);
    if (ImGui::Button("Reset Grid", ImVec2(-1,0)))
    {
      std::cout << "ResetGrid\n";
      // Recreate the grid
      createGrid();
      // Switch view to show the grid
      callback_key_down(viewer,'3',0);
    }

    // TODO: Add more parameters to tweak here...
  }

};
```

Better Normal Constraints

- In the previous, we require the implicit function to approximate some desired ***values*** at points
- The normals are simulated in the constraints by using inward and outward value constraints
  - Leads to undesirable surface oscillation



- Solution: use the normal to define a **linear function** at each sample point; interpolate these functions with MLS.
  - Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. "Interpolating and Approximating Implicit Surfaces from Polygon Soup". In *Proceedings of ACM SIGGRAPH 2004*, pages 896–904. ACM Press, August 2004. (Section 3.3)

MLS reconstruction with normal constraints

$$\begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x},\mathbf{p}_N) \end{bmatrix} \begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c}(\mathbf{x}) = \begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x},\mathbf{p}_N) \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

MLS reconstruction with normal constraints

$$\begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x},\mathbf{p}_N) \end{bmatrix} \begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c}(\mathbf{x}) = \begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x},\mathbf{p}_N) \end{bmatrix} \begin{bmatrix} S_1(\mathbf{x}) \\ \vdots \\ S_N(\mathbf{x}) \end{bmatrix}$$

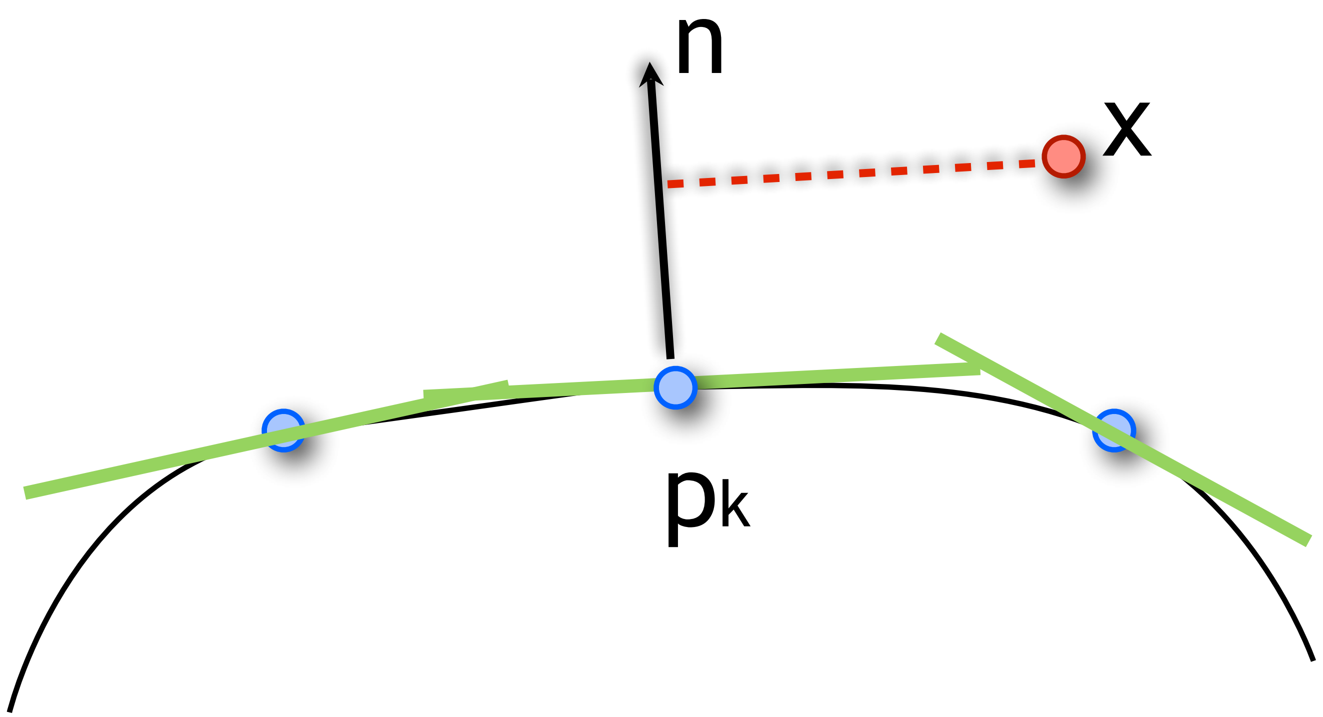
function values

$$\begin{aligned} S_k(\boldsymbol{x}) &= \phi_k + (\boldsymbol{x} - \boldsymbol{p}_k)^{\mathsf{T}} \, \hat{\boldsymbol{n}}_k \\ &= \psi_{0k} + \psi_{xk}\, x + \psi_{yk}\, y + \psi_{zk}\, z \end{aligned}$$

Instead of a blend between constant values associated with each (grid) point, we blend between functions associated with them

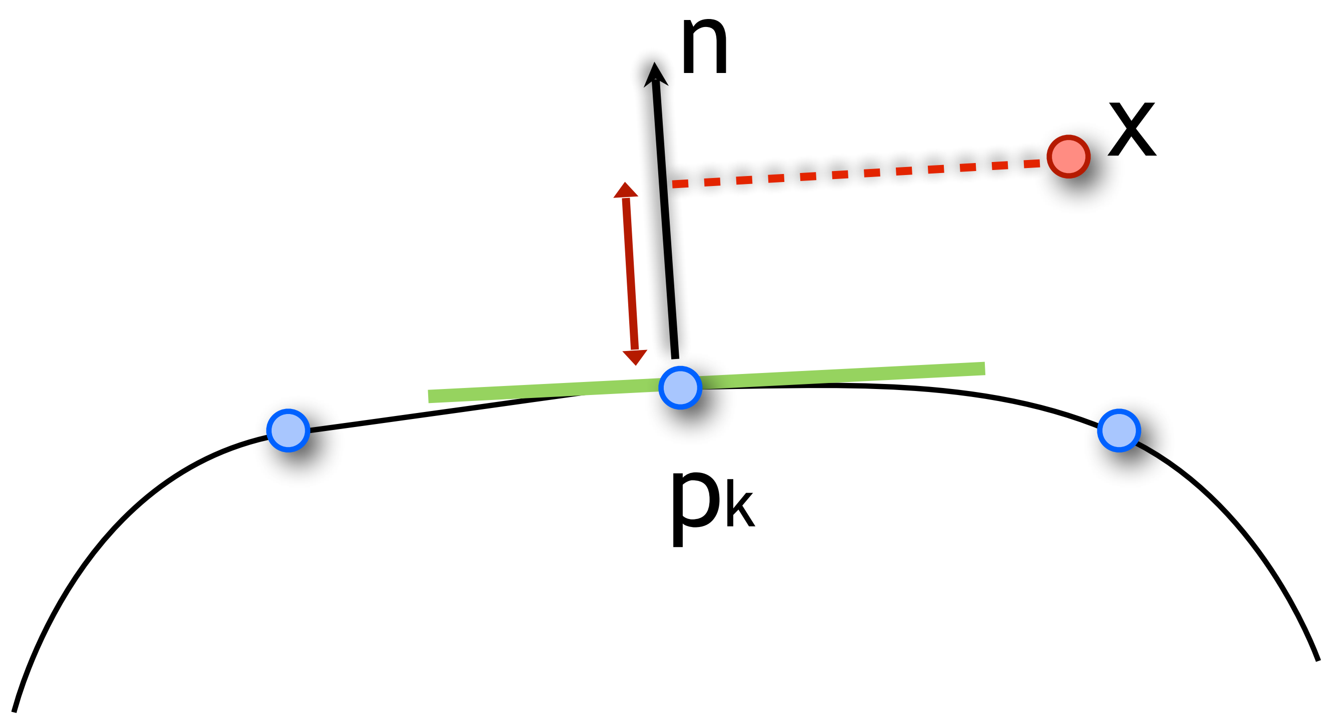
Normal Constraints

$$\begin{aligned} S_k(\boldsymbol{x}) &= \phi_k + (\boldsymbol{x} - \boldsymbol{p}_k)^\mathsf{T}\, \hat{\boldsymbol{n}}_k \\ &= \psi_{0k} + \psi_{xk}\, x + \psi_{yk}\, y + \psi_{zk}\, z \end{aligned}$$



Normal Constraints

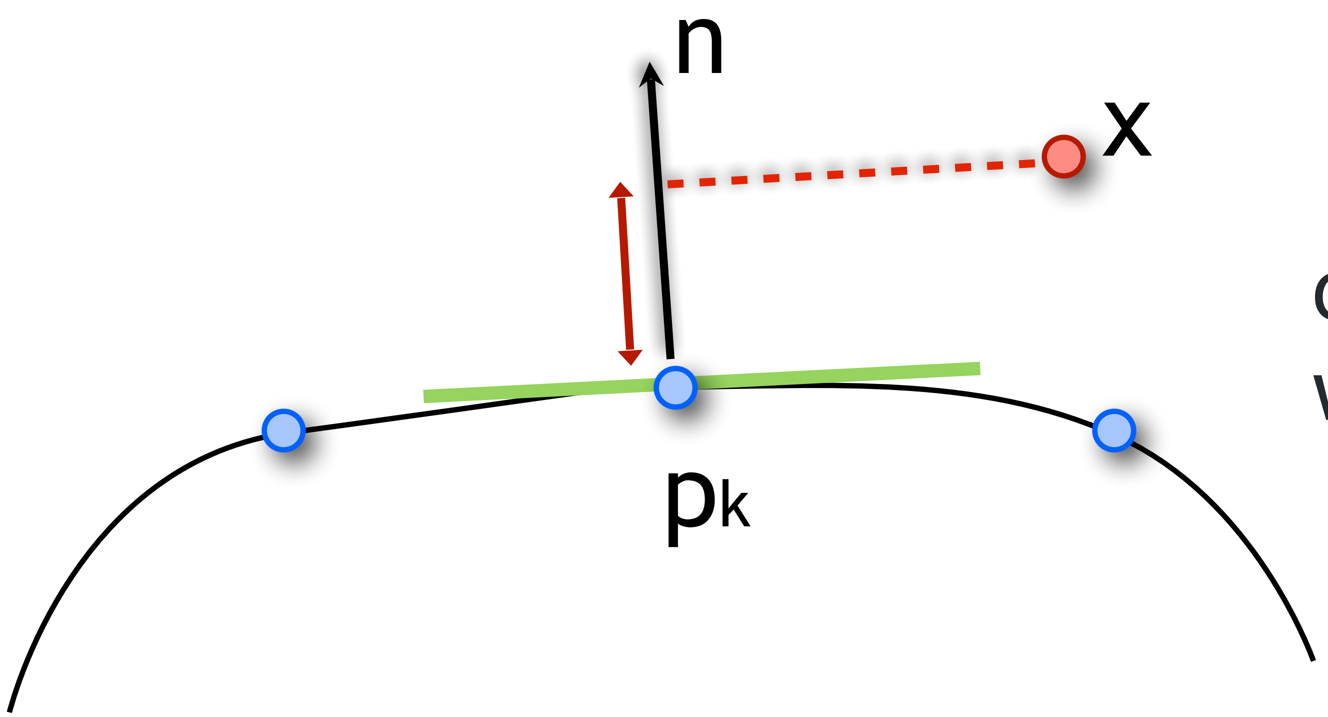
$$\begin{aligned} S_k(\boldsymbol{x}) &= \phi_k + (\boldsymbol{x} - \boldsymbol{p}_k)^{\mathsf{T}}\, \hat{\boldsymbol{n}}_k \\ &= \psi_{0k} + \psi_{xk}\, x + \psi_{yk}\, y + \psi_{zk}\, z \end{aligned}$$



$$\nabla_{\mathbf{x}} S_k(\mathbf{x}) = \hat{\mathbf{n}}_{\mathbf{k}}$$

Normal Constraints

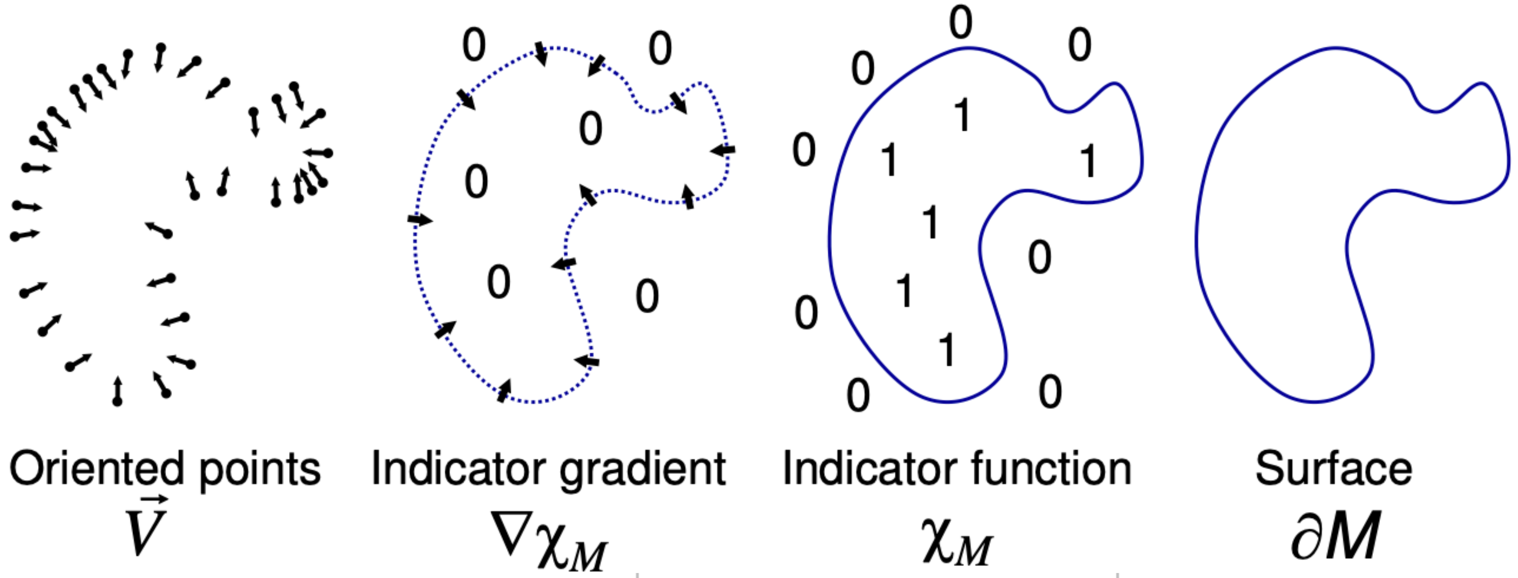
$$S_k(\boldsymbol{x}) = \phi_k + (\boldsymbol{x} - \boldsymbol{p}_k)^T \hat{\boldsymbol{n}}_k$$
$$= \psi_{0k} + \psi_{xk}\, x + \psi_{yk}\, y + \psi_{zk}\, z$$



only uses the original (N) input points
without the generated (2N) constrained points

Poisson Reconstruction

- Explicitly fit a scalar function's gradient to the normal
  - Smooth out sampled normals to create a global vector field $\vec{V}$
  - Find scalar function whose gradient best approximates this vector field $min_{\chi}\|\nabla\chi - \vec{V}\|$
- Advantages
  - No spurious sheets far from surface
  - Robust to noise
- Michael Kazhdan, Matthew Bolitho, Hugues Hoppe
  "Poisson Surface Reconstruction"
  In *Eurographics Symposium on Geometry Processing, 2006*



- No implementation required: use [MeshLab]

Robust Implicit MLS

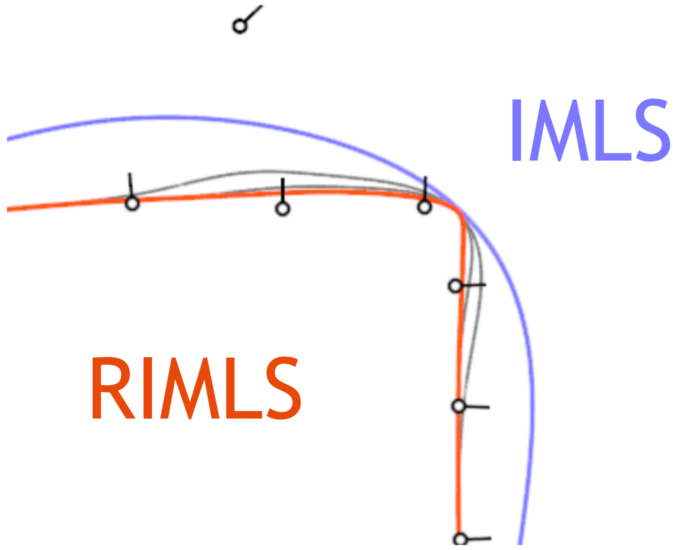
- Improved optimization problem to solve
  - Use an estimator ρ giving less weight to outliers (R - robust)

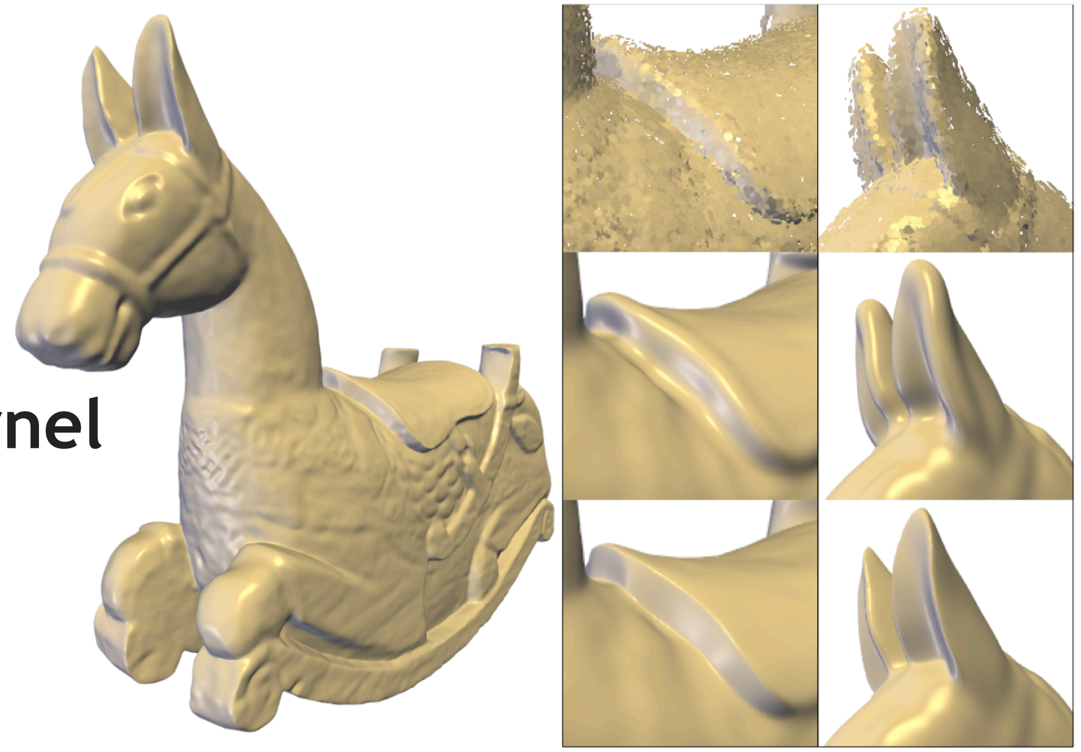
$$\arg\min_{\mathbf{s}} \sum \rho(y_i - g_{\mathbf{s}}(\mathbf{x}_i))\,\phi_i(\mathbf{x})$$

  - Use an iterative method

$$\mathbf{s}^k = \arg\min_{\mathbf{s}} \sum \phi_i(\mathbf{x}) w(r_i^{k-1})(y_i - g_{\mathbf{s}}^k(\mathbf{x}_i))^2$$

- Advantages
  - Better reconstruction of sharp features
  - Robust to noise
- Cengiz Öztireli, Gael Guennebaud, and Markus Gross.
  "Feature preserving point set surfaces based on non-linear kernel regression"
  In Computer Graphics Forum, 2009
- No implementation required: use MeshLab



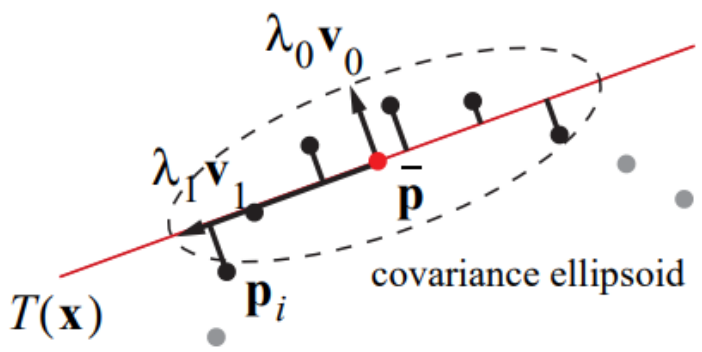


ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

PCA Points Normal Estimation

- Lecture of this week
  - Step 1: construct 3x3 covariance matrix C

$$\mathbf{C} = \begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \cdots \\ \mathbf{p}_{i_k} - \bar{\mathbf{p}} \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \cdots \\ \mathbf{p}_{i_k} - \bar{\mathbf{p}} \end{bmatrix}, i_j \in N_p ,$$



By Pauly et al.

  - Step 2: compute eigenvalues and eigenvectors of C

$$\mathbf{C} \cdot \mathbf{v}_l = \lambda_l \cdot \mathbf{v}_l, l \in \{0, 1, 2\}$$

  - Step 3: take the eigenvector corresponding to the smallest eigenvalue as normal.

Assignment 2

- Due date: Friday 28.03.2025 at 10:00 am (in 2 weeks)
  - push on your repo with the commit message "Solution Assignment 2"
  - the README.md is self-explanatory for the report
  - add results (files/notes) to the folder assignment2/res/ and refer to them from the README.md

Assignment 2

- Any questions?
- Next Friday is Q&A of assignment 2

Thank you!

Acknowledgments:
Oliver Glauser, Michael Rabinovich, Olga Diamanti, Christian Schüller, Shihao Wu