Shape Modeling and Geometry Processing

Assignment 2 - Implicit Surfaces

Michael Rabinovich -

michael.rabinovich@inf.ethz.ch

Acknowledgments: Olga Diamanti, Christian Schüller

igl

INTERACTIVE GEOMETRY LAB

March 10, 2017

ETH

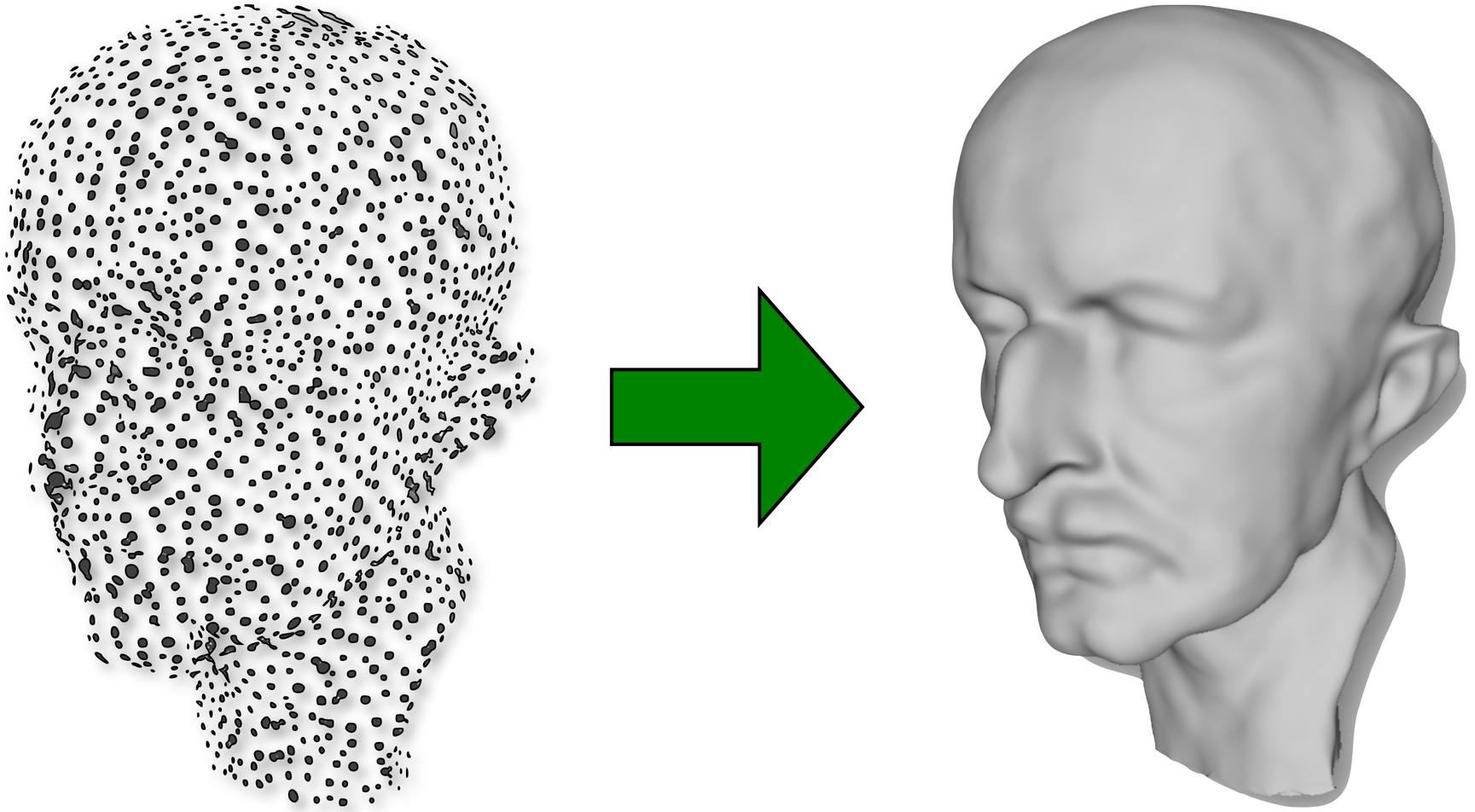
Eidgenössische Technische Hochschule Zürich

Swiss Federal Institute of Technology Zurich

Assignments

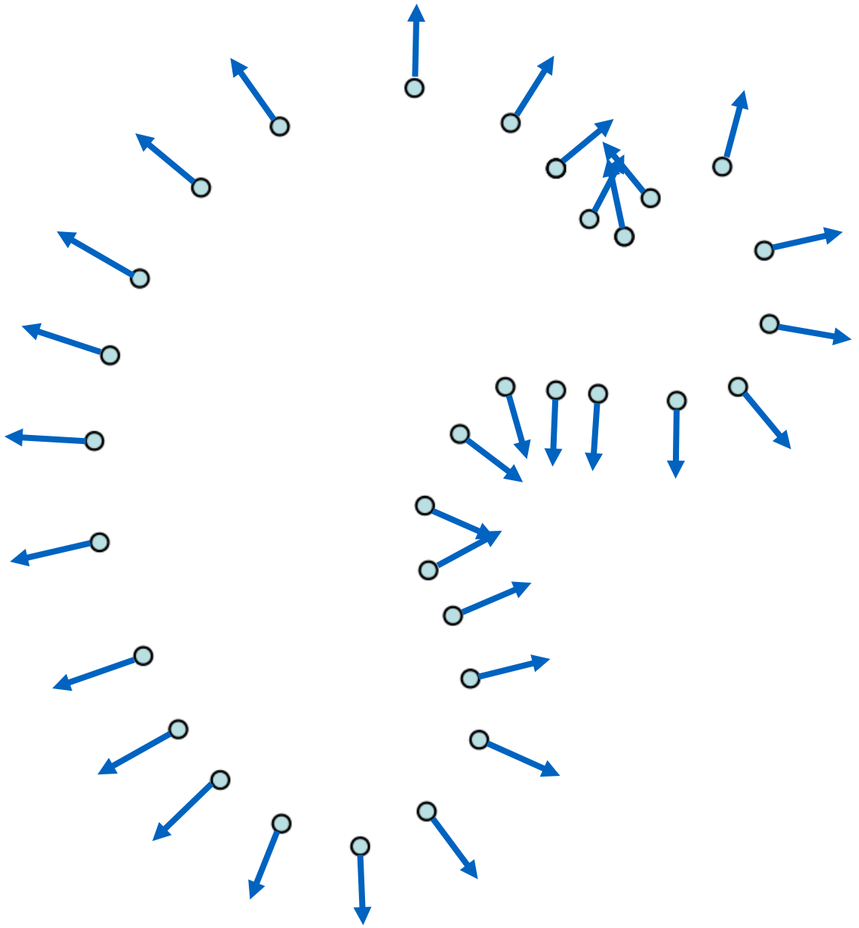
- Please regularly check the main repository for updates and new instruction:
  - https://github.com/eth-igl/GP2018-Assignments
- Any general issues, problems or questions?

Implicit Surface Reconstruction



Implicit Surface Reconstruction

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

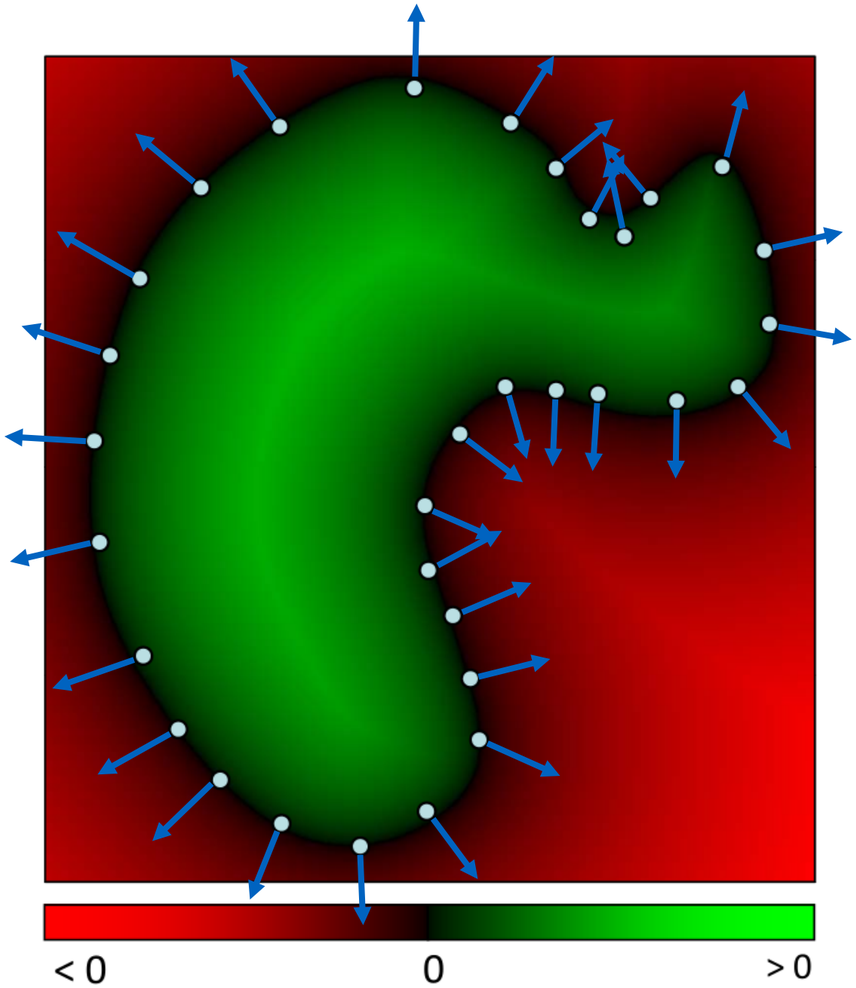


Implicit Surface Reconstruction

- Find a function (scalar-field)

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

- Value < 0 outside
- Value > 0 inside

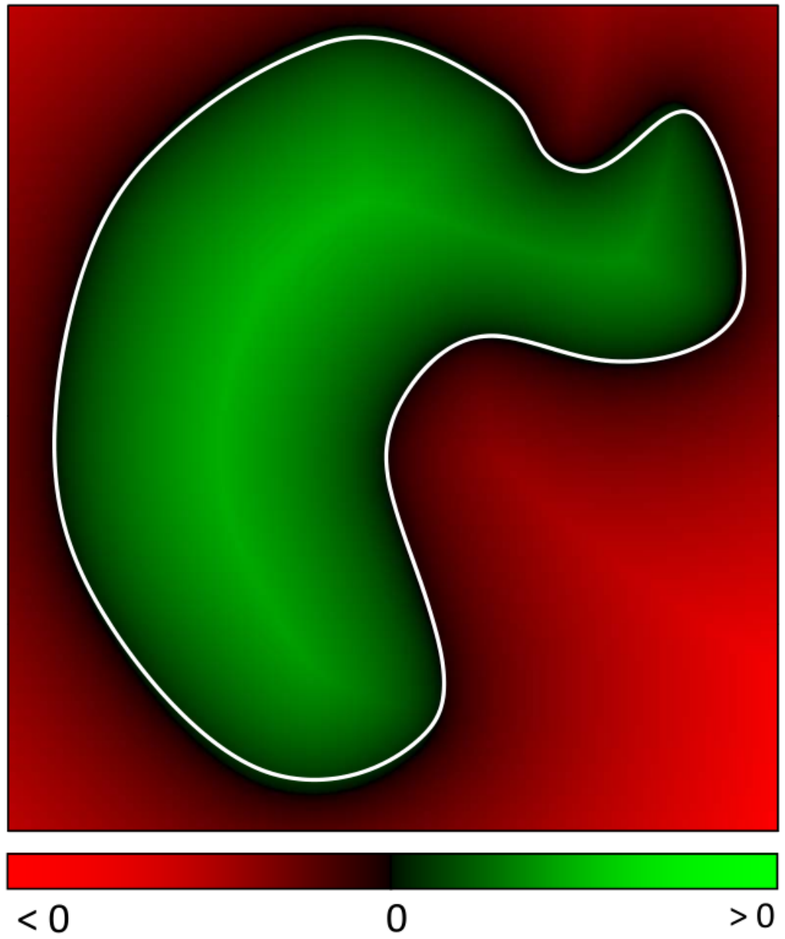


Implicit Surface Reconstruction

- Extract the zero-set

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

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



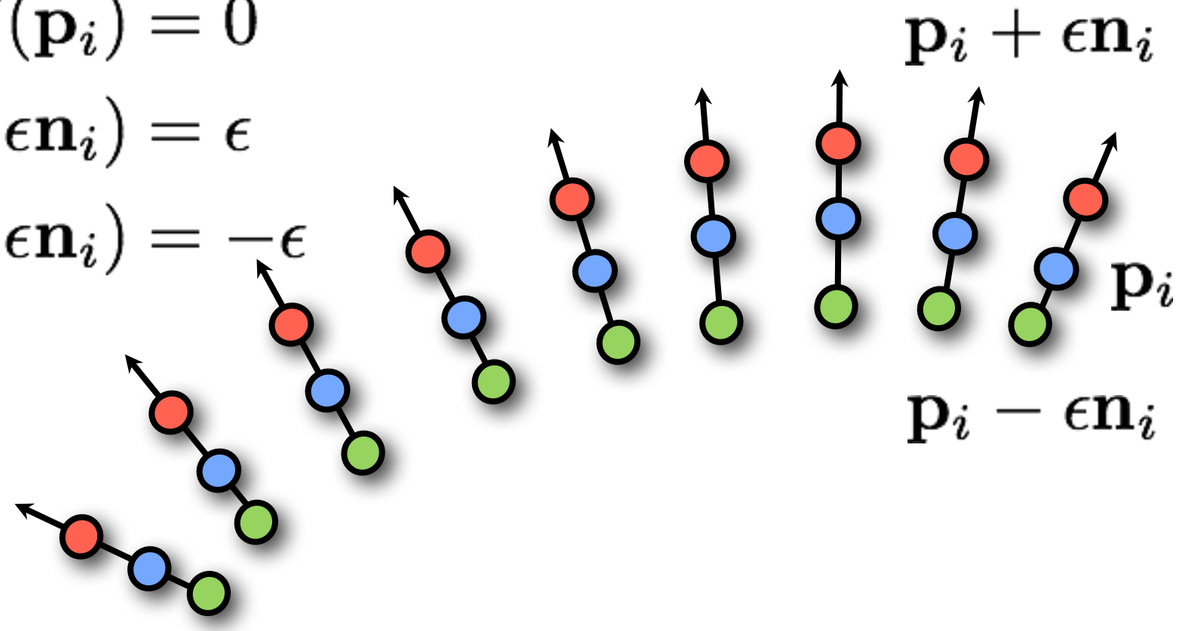
Assignment 2

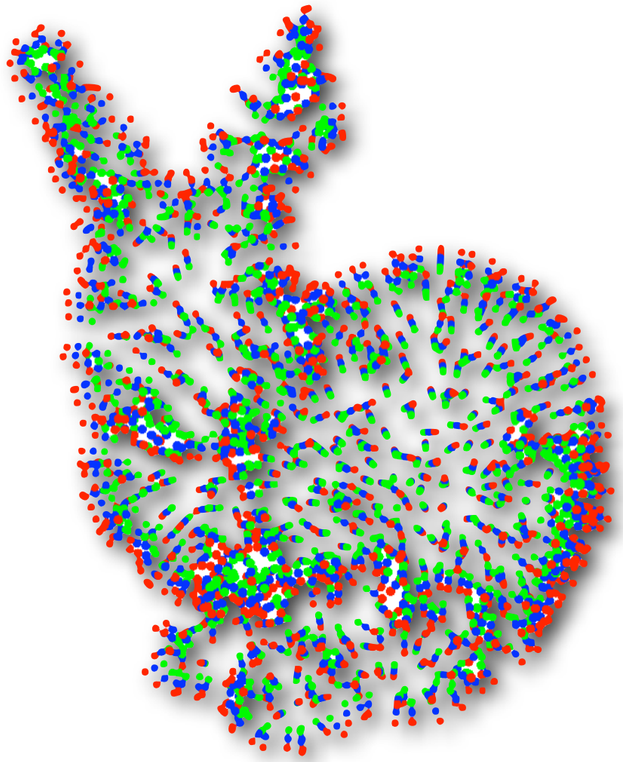
- Input:
  .off/.obj file
  with points and normals

Step 1: Build constraint set

Incorporate normal info width off-surface constraints:

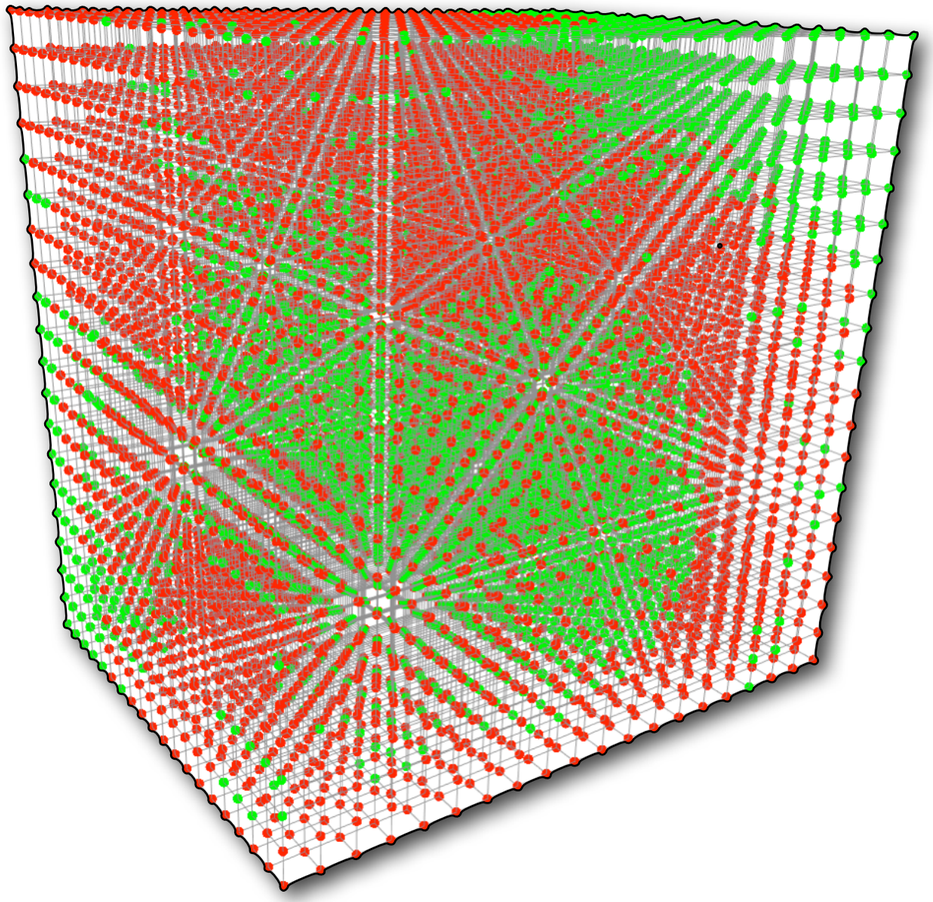
$$f(\mathbf{p}_i) = 0$$
$$f(\mathbf{p}_i + \epsilon\mathbf{n}_i) = \epsilon$$
$$f(\mathbf{p}_i - \epsilon\mathbf{n}_i) = -\epsilon$$





Step 2: Construct Interpolant

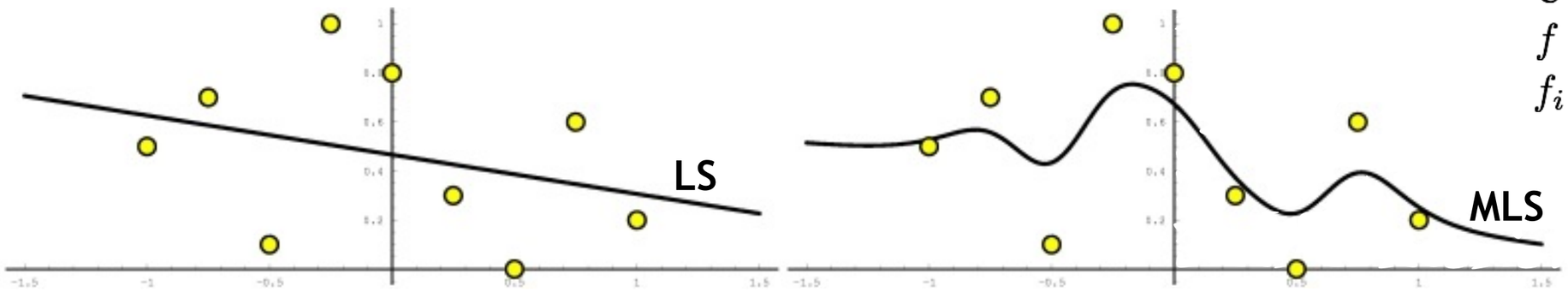
- Construct regular grid
- Compute nodal scalar field satisfying constraints (approximately)
- Method: **MLS (Moving Least Squares)**



The Least Squares Family

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



$\mathbf{p}_i$ sample points
$\mathbf{c}$ coefficients
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:

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

Standard LS reconstruction

- Standard least-squares fit

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

- Solve a overdetermined linear system (use Eigen library)

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

Standard LS reconstruction

- Standard least-squares fit

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

- 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

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

- Solve a overdetermined linear system (use Eigen library)

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

desired function values

Standard LS reconstruction

- Standard least-squares fit

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

- Solve a overdetermined linear system (use Eigen library)

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

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

coefficients of polynomial basis

MLS reconstruction

- MLS fit

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

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

MLS reconstruction

- MLS fit

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

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

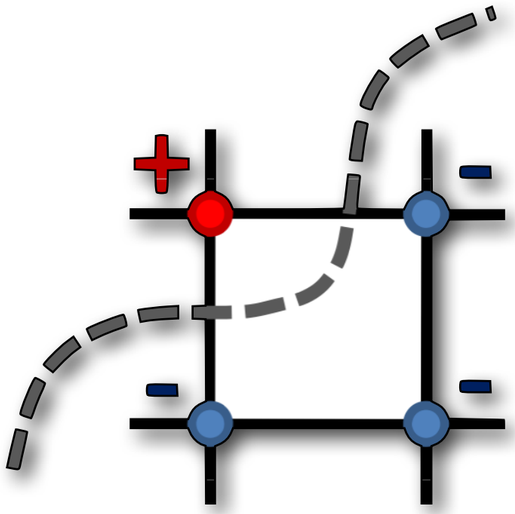
proximity weights — proximity weights

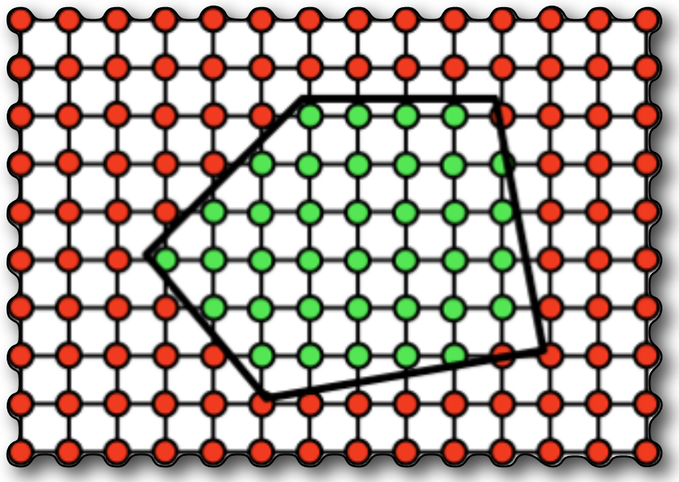
$$w(\mathbf{x}, \mathbf{p}_i) = f_w(||\mathbf{x} - \mathbf{p}_i||)$$

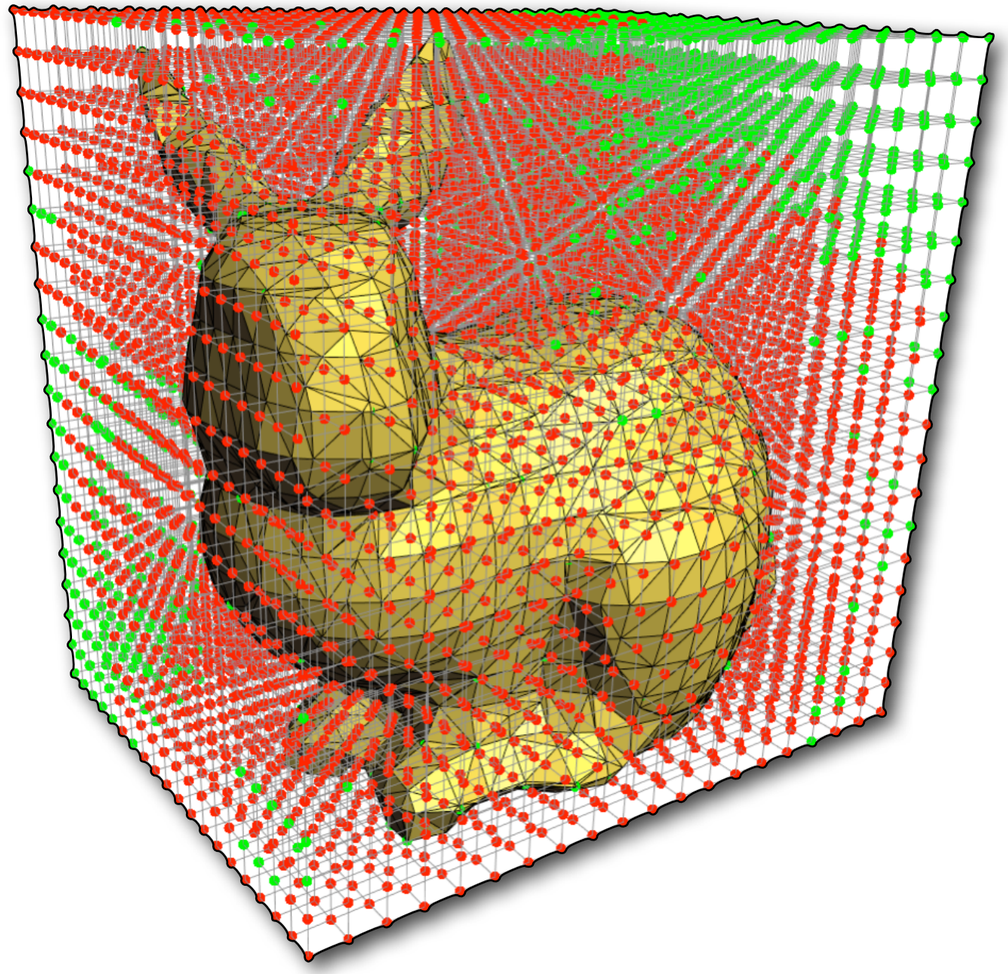
f_w : weight function

Step 3: Marching Cubes

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

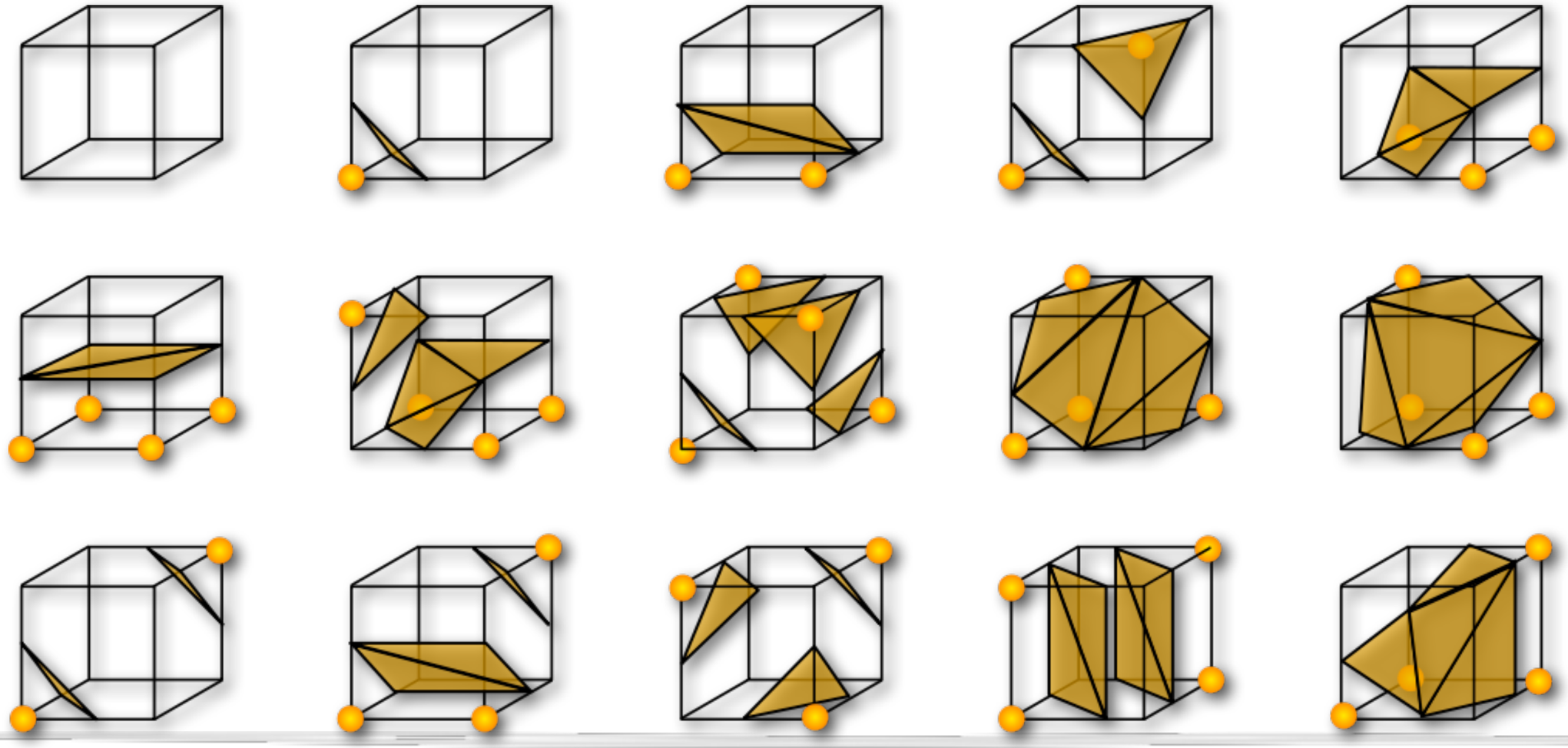




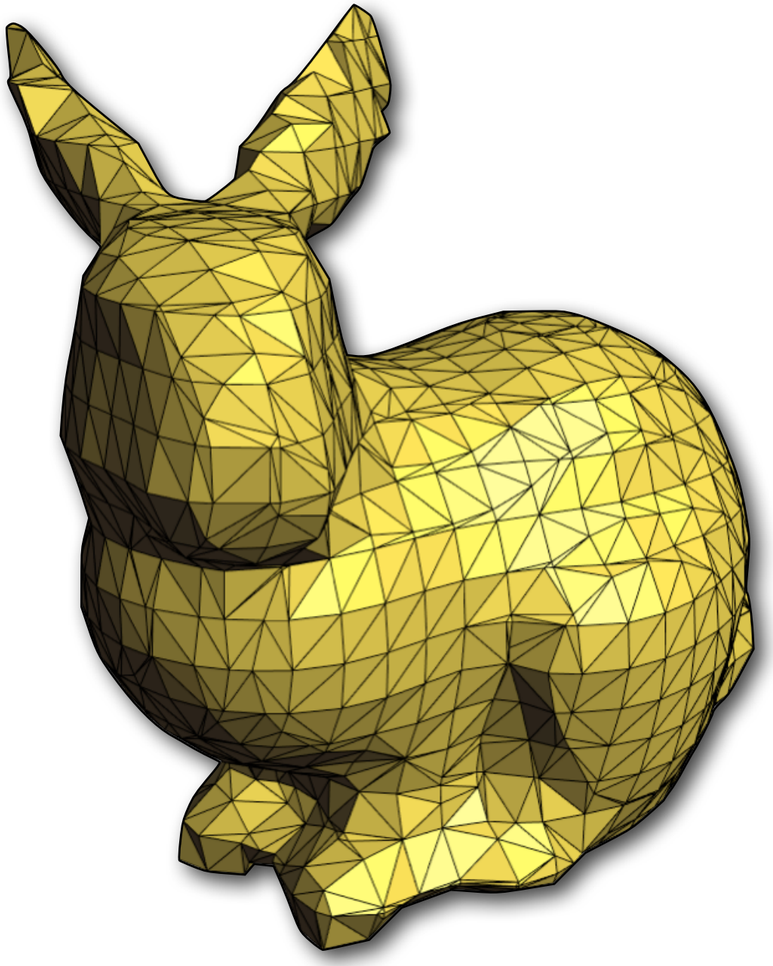


Step 3: Marching Cubes

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

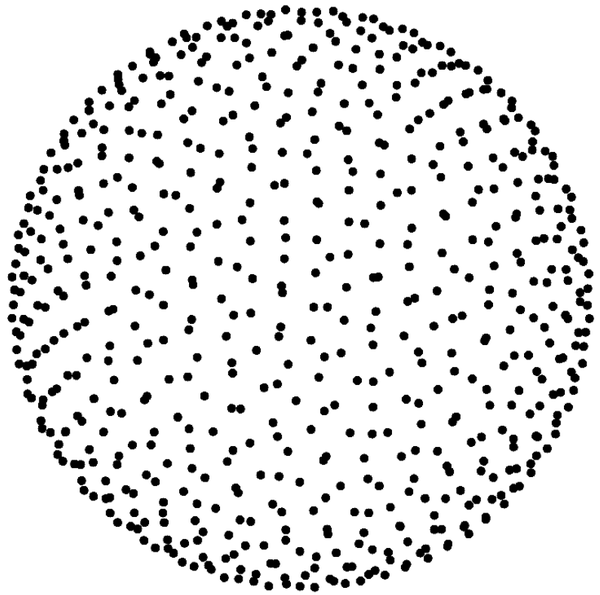


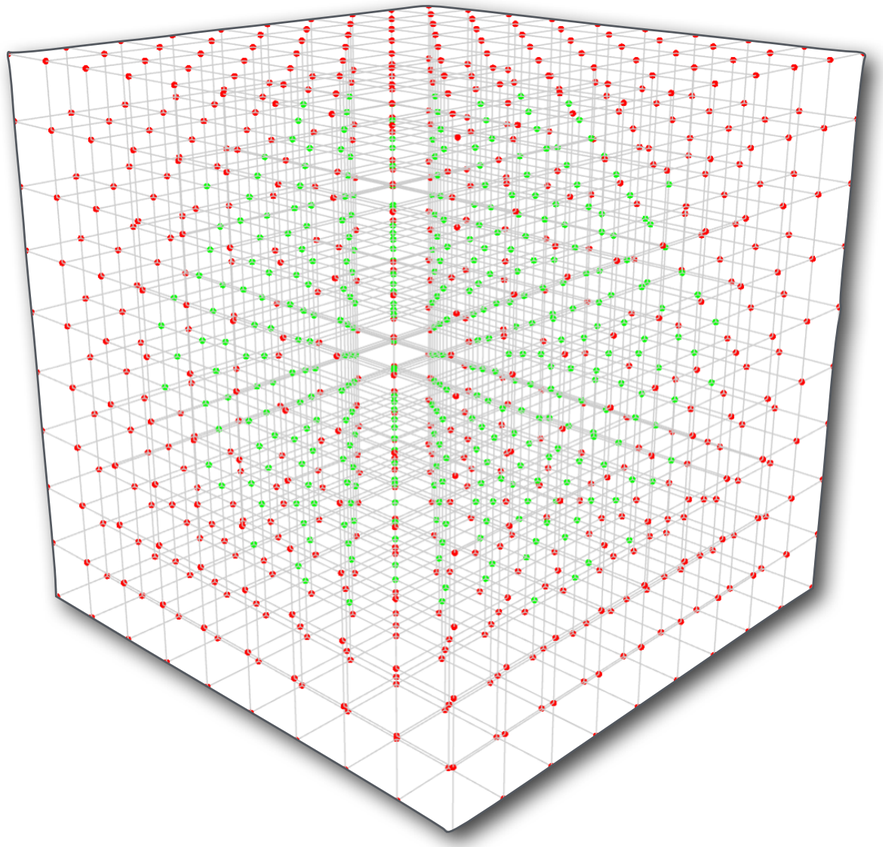
Final Mesh

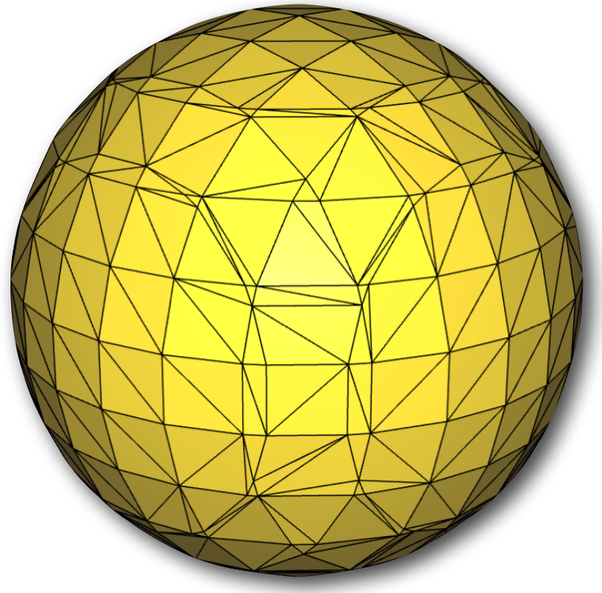


Provided Example

- 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

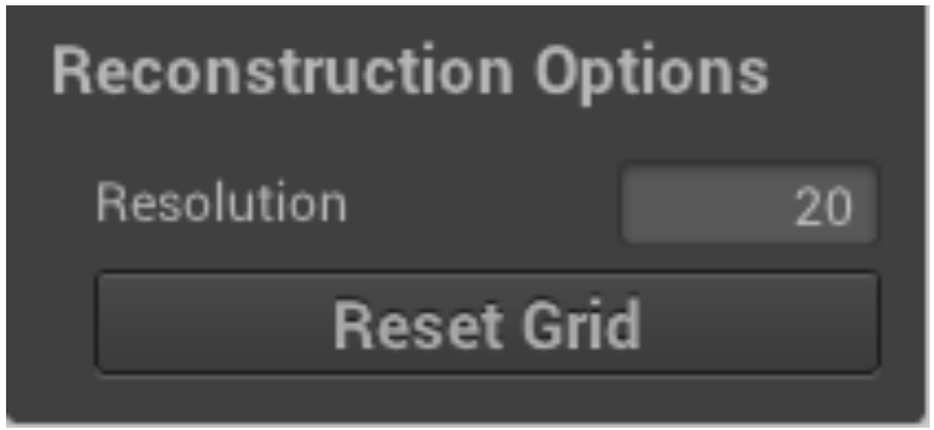
- Step 4: run marching cubes

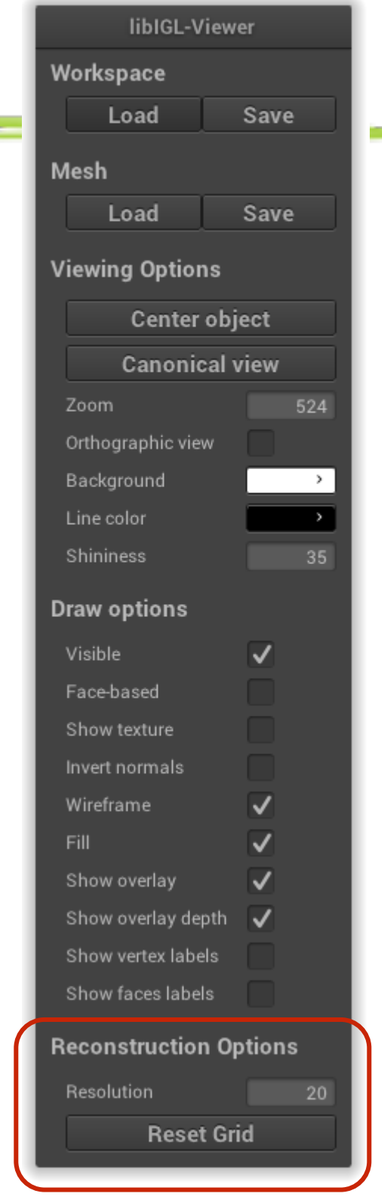
```
// Run marching cubes
igl::copyleft::marching_cubes(grid_values, grid_points, resolution, resolution, resolution, V, F);
```

output: vertices and faces

NanoGUI

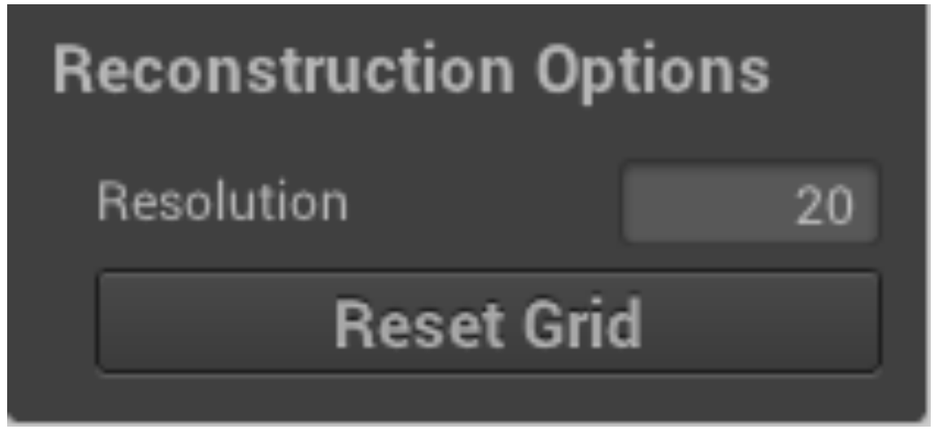
- IGL Viewer uses NanoGUI: http://nanogui.readthedocs.io/
- You'll need to add widgets to configure additional variables.





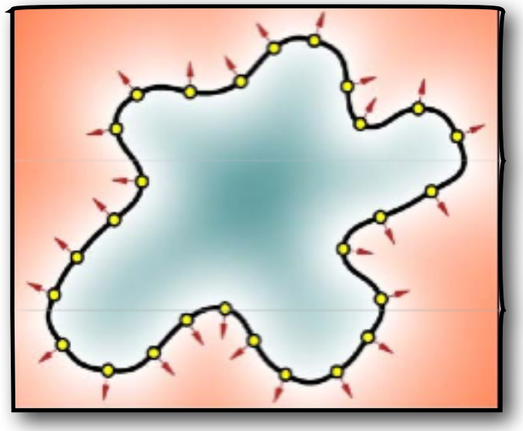
NanoGUI: Adding Settings

```
viewer.callback_init = [&](Viewer &v) {
    // Add widgets to the sidebar.
    v.ngui->addGroup("Reconstruction Options");
    v.ngui->addVariable("Resolution", resolution);
    v.ngui->addButton("Reset Grid", [&](){
        // Recreate the grid
        createGrid();
        // Switch view to show the grid
        callback_key_down(v, '3', 0);
    });

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

    v.screen->performLayout();
    return false;
};
```



Bonus: 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

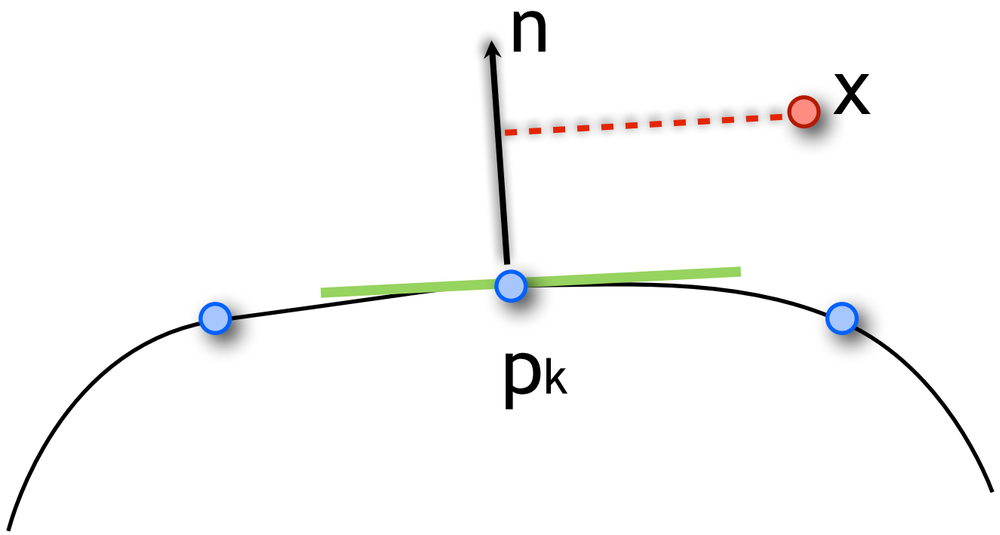
$$\begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x},\mathbf{p}_N) \end{bmatrix} \begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \mathbf{c}(\mathbf{x}) = \begin{bmatrix} w(\mathbf{x},\mathbf{p}_1) & & \\ & \ddots & \\ & &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

function values

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

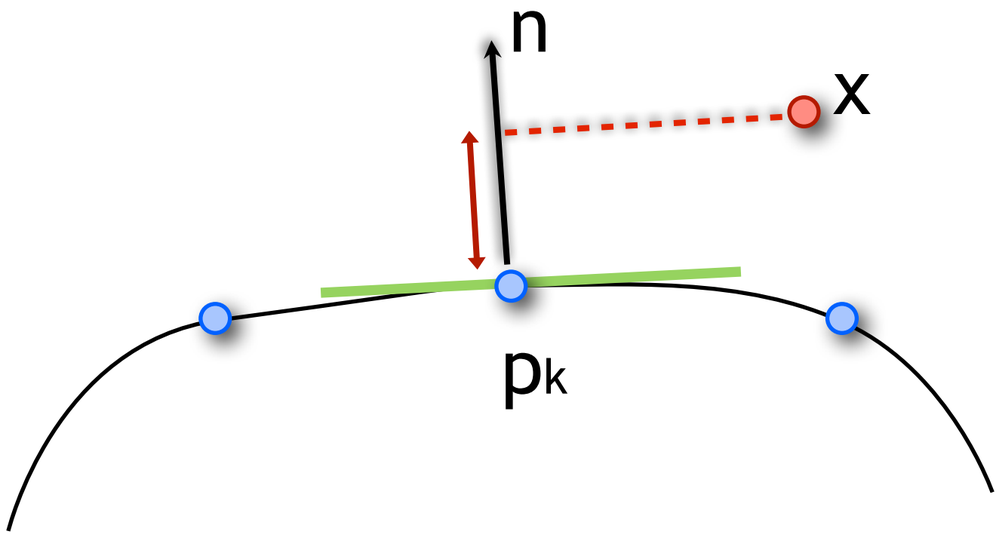
Normal Constraints

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



Normal Constraints

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



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

Bonus: 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*

Assignment 2

- Due-date: Wednesday 23.03.2018
- Add results (files/notes) to your private repository, and refer to them from the ‘README.md’:
  - directory: assignment2/results/
- 16% of the final grade
- Bonus tasks: directly added to your exercise grades
  - Eg. 67/80 points from mandatory exercises + 4 bonus points = 71/80
  - total cannot exceed 80

Questions?

- Exercise sessions
- Office hours: ***Wednesday, 13:00 - 14:00, CNB 108***
- Email: michael.rabinovich@inf.ethz.ch
- Appointment
- Bug reports/suggestions also welcome!