252-0538-00L, Spring 2017

Shape Modeling and Geometry Processing

Surface Reconstruction

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Geometry Acquisition Pipeline

Scanning: results in range images

⇨

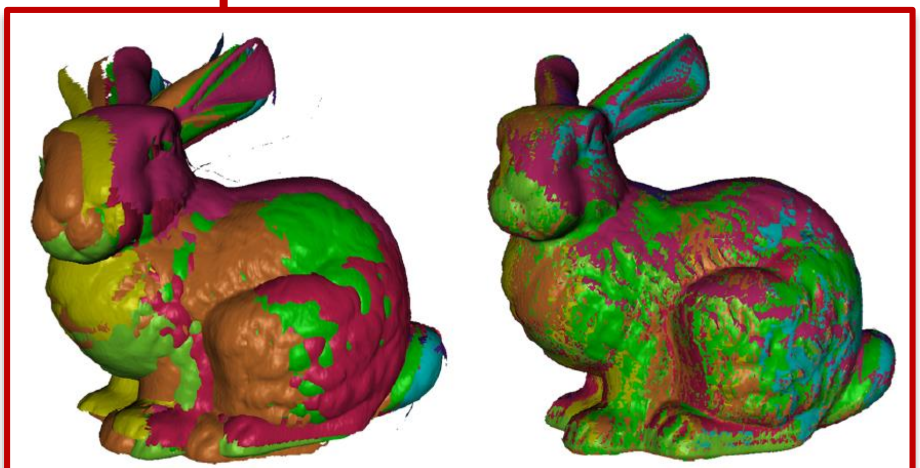
Registration: bring all range images to one coordinate system

⇨

Reconstruction: Integration of scans into a single mesh

⇨

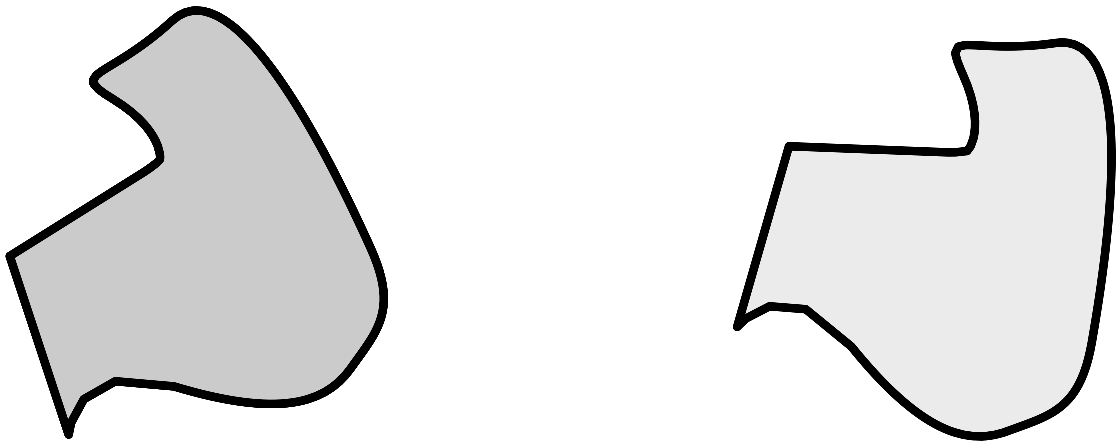
Postprocess:
- Topological and geometric filtering
- Remeshing
- Compression



ETH zürich

Problem Statement

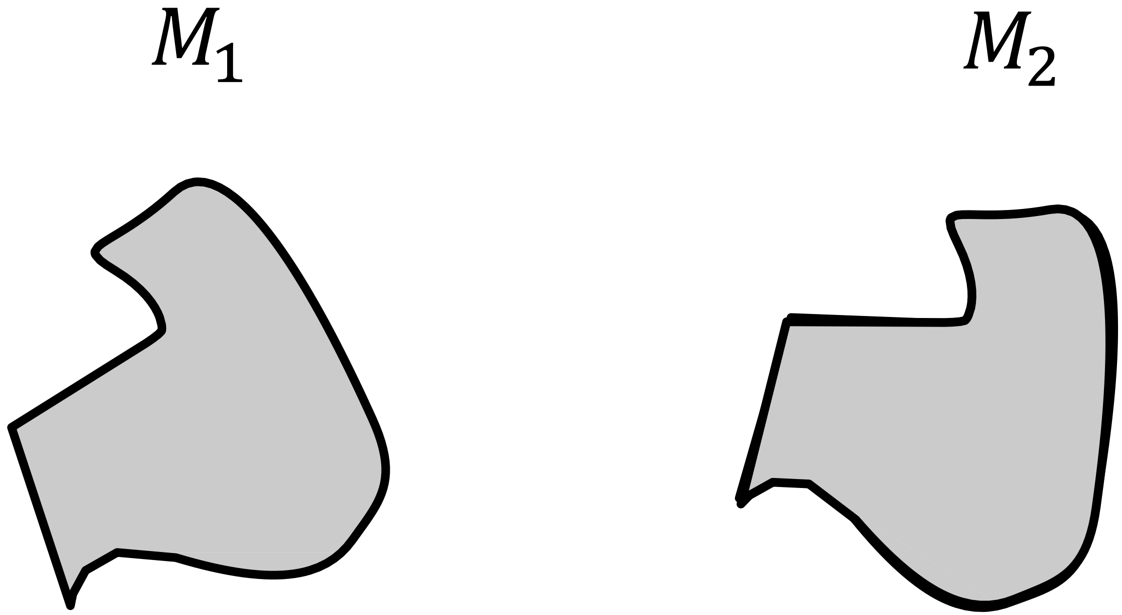
M_1 M_2



$$M_1 \approx T(M_2)$$

T: Translation + Rotation

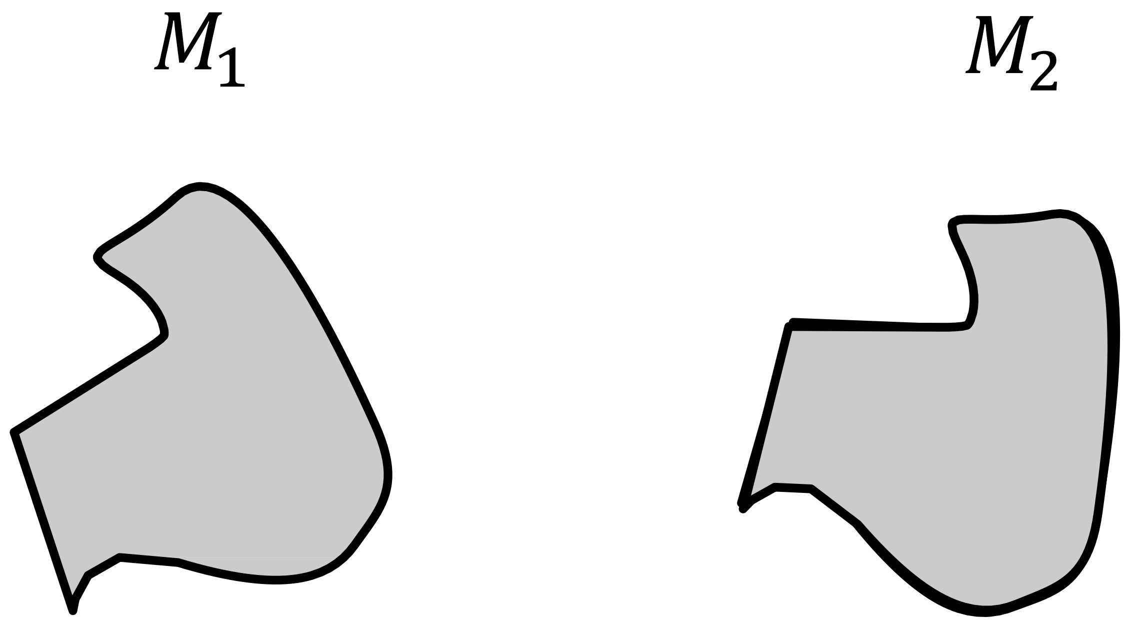
Problem Statement



$$M_1 \approx T(M_2)$$

T: Translation + Rotation

Problem Statement



Given $M_1, \ldots, M_n$ find $T_2, \ldots, T_n$ such that

$$M_1 \approx T_2(M_2) \approx \cdots \approx T_n(M_n)$$

Correspondences

How many points are needed to define a unique rigid transformation?

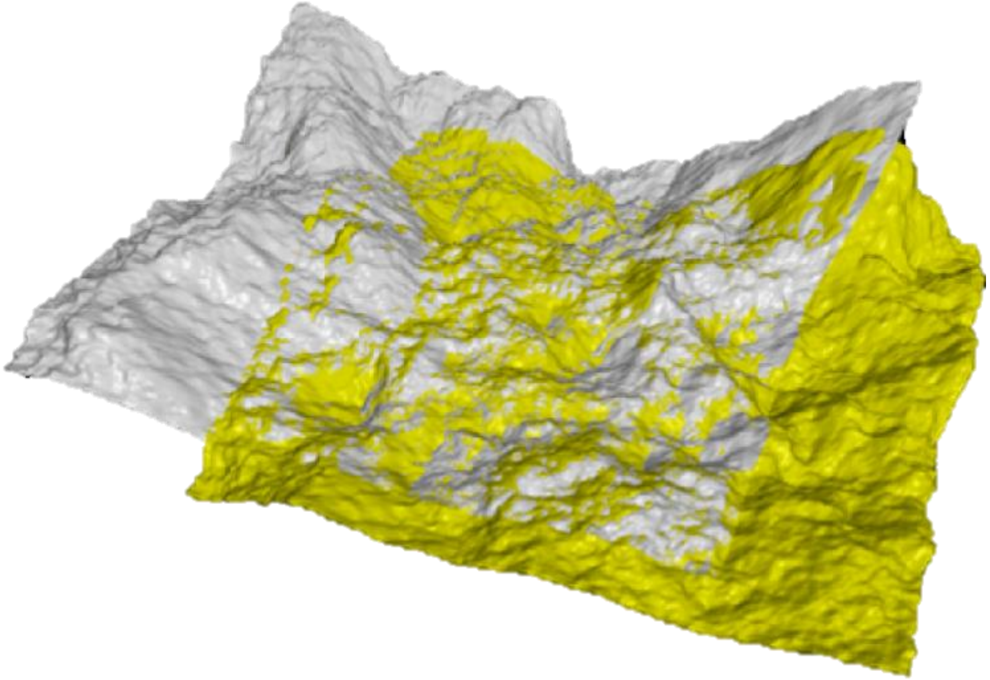
The first problem is finding pairs!

$$\mathbf{p}_1 \rightarrow \mathbf{q}_1$$

$$\mathbf{p}_2 \rightarrow \mathbf{q}_2$$

$$\mathbf{p}_3 \rightarrow \mathbf{q}_3$$

$$R\mathbf{p}_i + t \approx \mathbf{q}_i$$



ICP: Iterative Closest Point

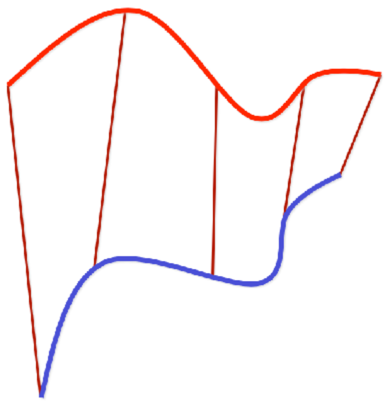
Intuition:

Correct correspondences ⇒ problem solved!

Idea:

(1) Find correspondences

(2) Use them to find a transformation



ICP: Iterative Closest Point

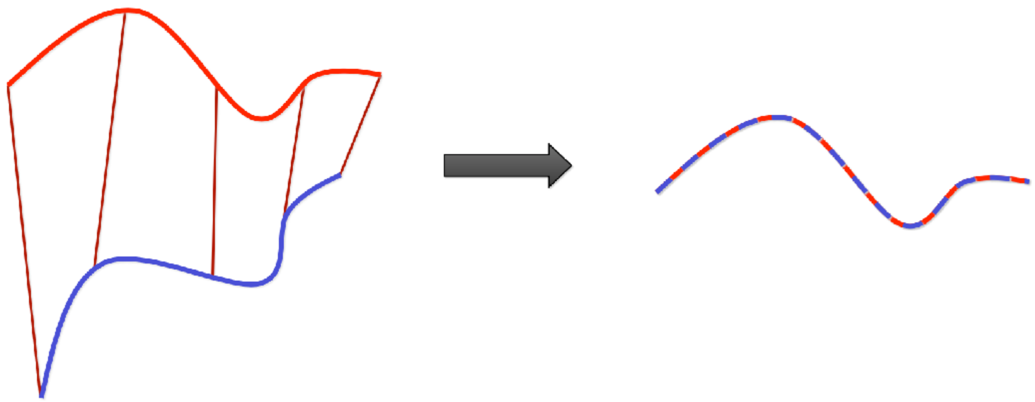
Intuition:

Correct correspondences ⇒ problem solved!

Idea:

(1) Find correspondences

(2) Use them to find a transformation



ICP: Iterative Closest Point

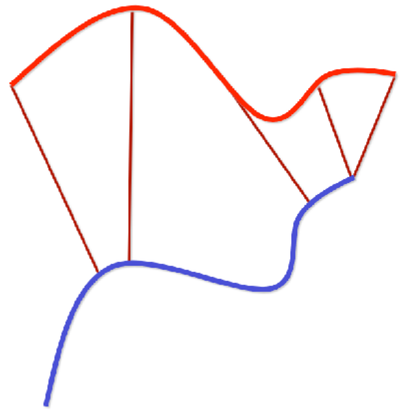
Intuition:

Correct correspondences ⇒ problem solved!

Idea:

(1) Find correspondences

(2) Use them to find a transformation



ICP: Iterative Closest Point

Intuition:

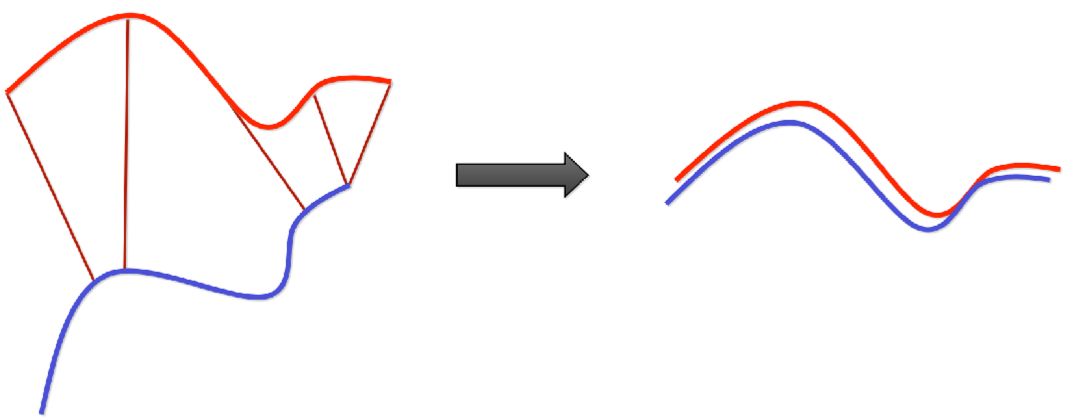
Correct correspondences ⇒ problem solved!

Idea:

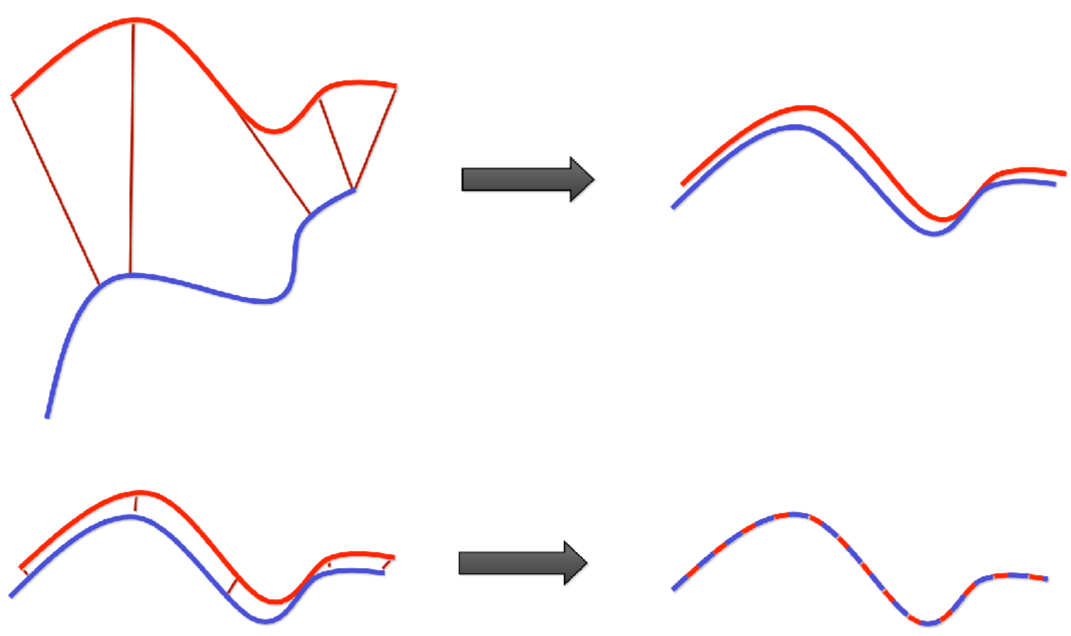
Iterate

(1) Find correspondences

(2) Use them to find a transformation



ICP: Iterative Closest Point



This algorithm converges to the correct solution **if** the starting scans are "close enough"

ETH zürich

Basic Algorithm

Select (e.g., 1000) random points

Match each to closest point on other scan

Reject pairs with distance too big

Minimize

$$E := \sum_i (R\mathbf{p}_i + t - \mathbf{q}_i)^2$$

closed form solution in:
http://dl.acm.org/citation.cfm?id=250160

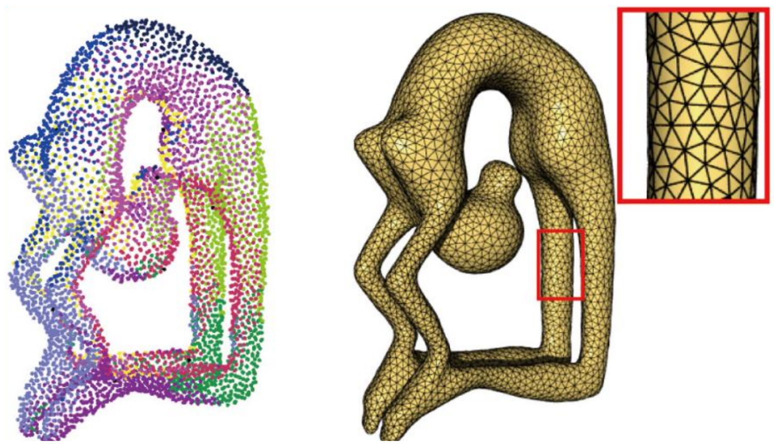
Geometry Acquisition Pipeline

Scanning: results in range images

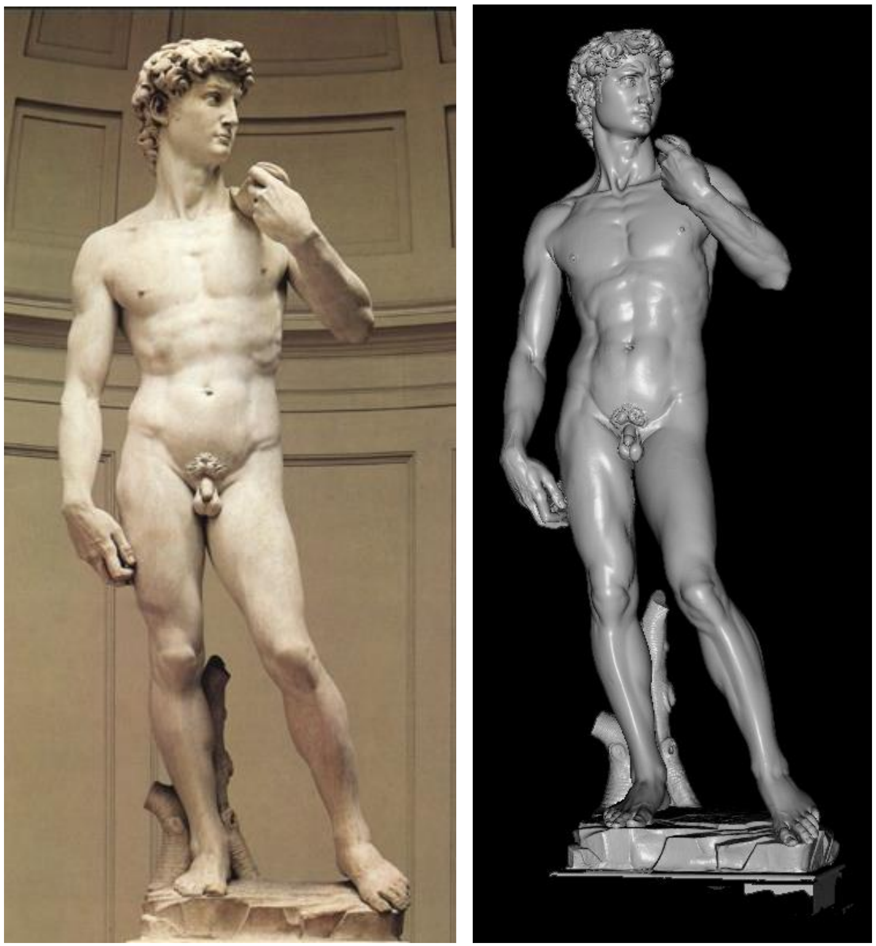
Registration: bring all range images to one coordinate system

Reconstruction: Integration of scans into a single mesh

Postprocess:
- Topological and geometric filtering
- Remeshing
- Compression



Digital Michelangelo Project

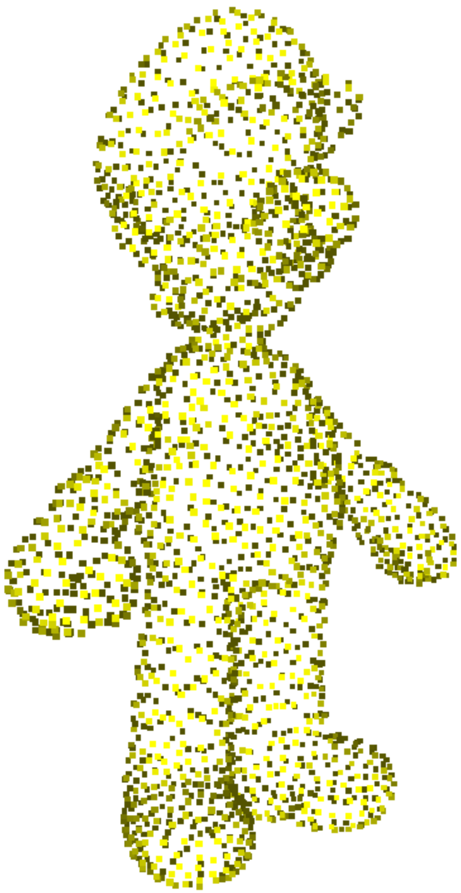


1G sample points → 8M triangles

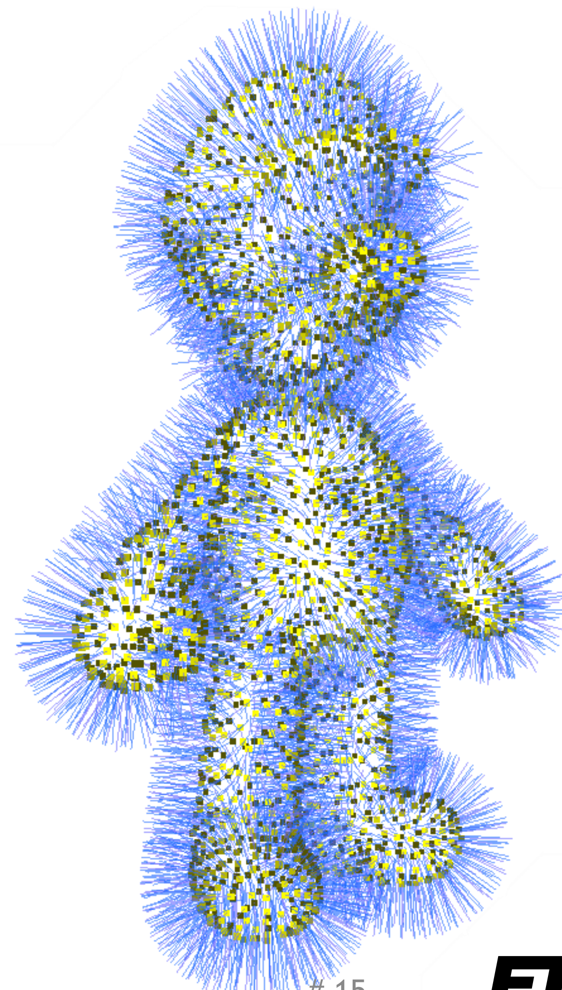
4G sample points → 8M triangles

Input to Reconstruction Process

Point cloud

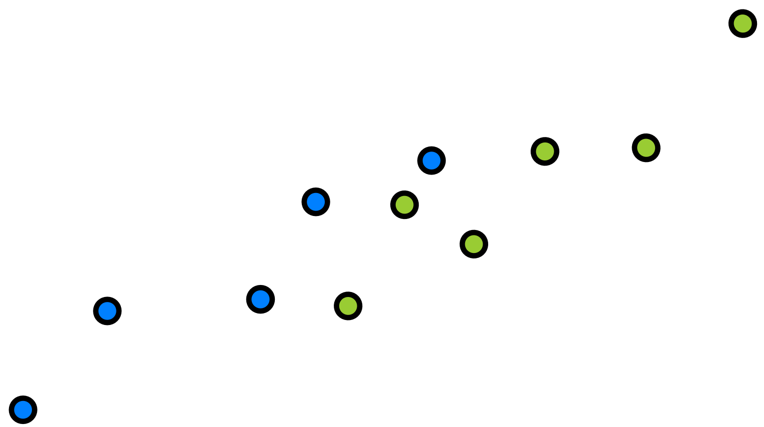


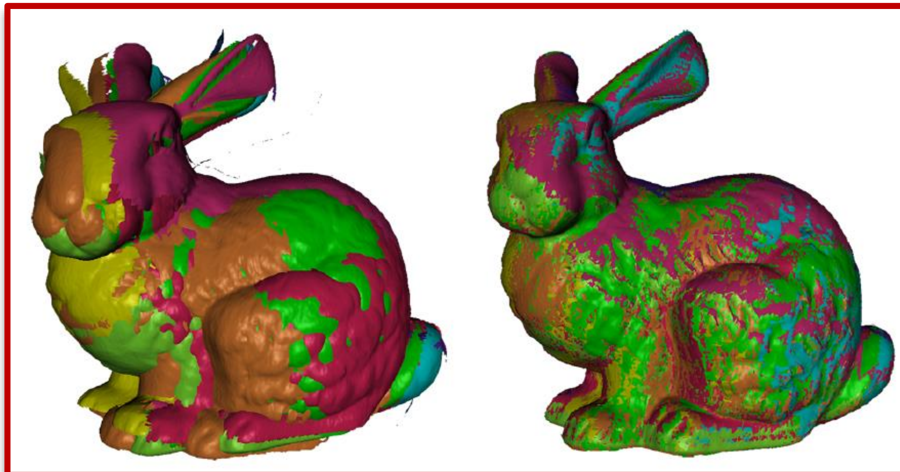
Oriented Point cloud



How to Connect the Dots?

Explicit reconstruction:
Connect sample points by triangles

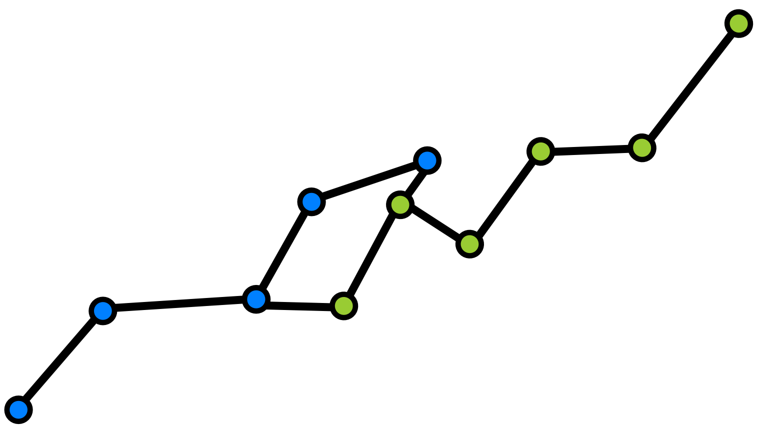


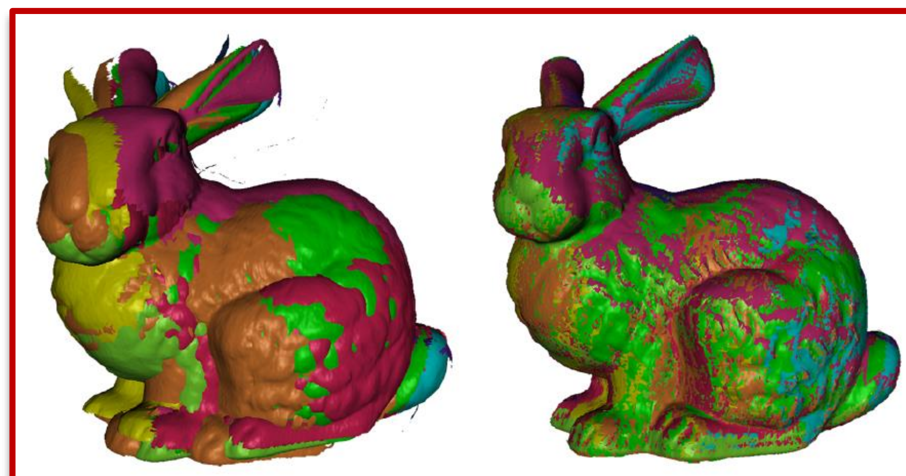


“Zippered Polygon Meshes from Range Images”, Greg Turk and Marc Levoy, ACM SIGGRAPH 1994

How to Connect the Dots?

Explicit reconstruction:
Connect sample points by triangles

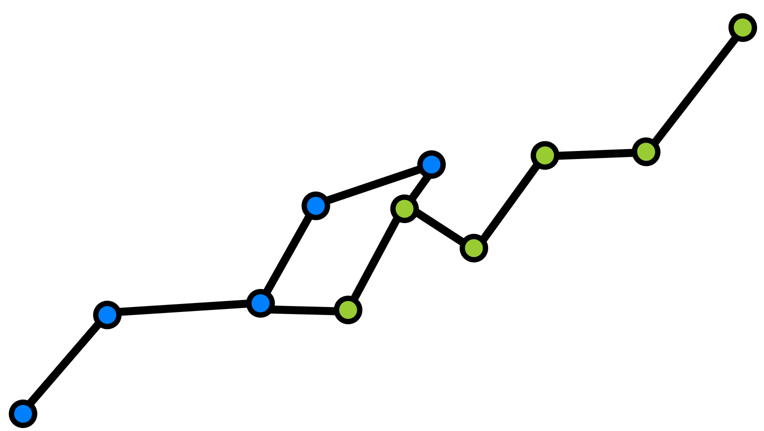




“Zippered Polygon Meshes from Range Images”, Greg Turk and Marc Levoy, ACM SIGGRAPH 1994

How to Connect the Dots?

Explicit reconstruction:
Connect sample points by triangles



Problems:

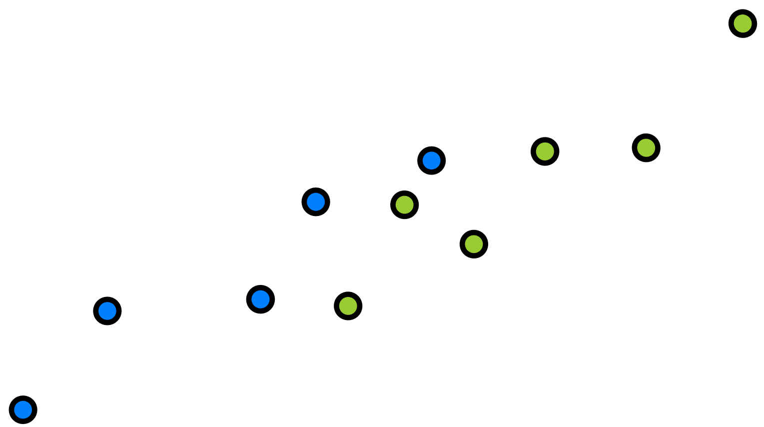
- Bad for noisy or misaligned data
- Can lead to holes or non-manifold situations

How to Connect the Dots?

Implicit reconstruction:

estimate a signed distance function (SDF)

extract zero set

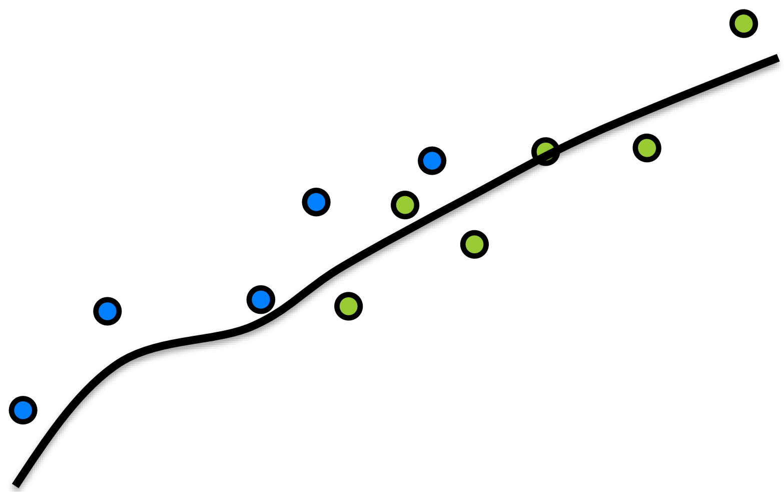


How to Connect the Dots?

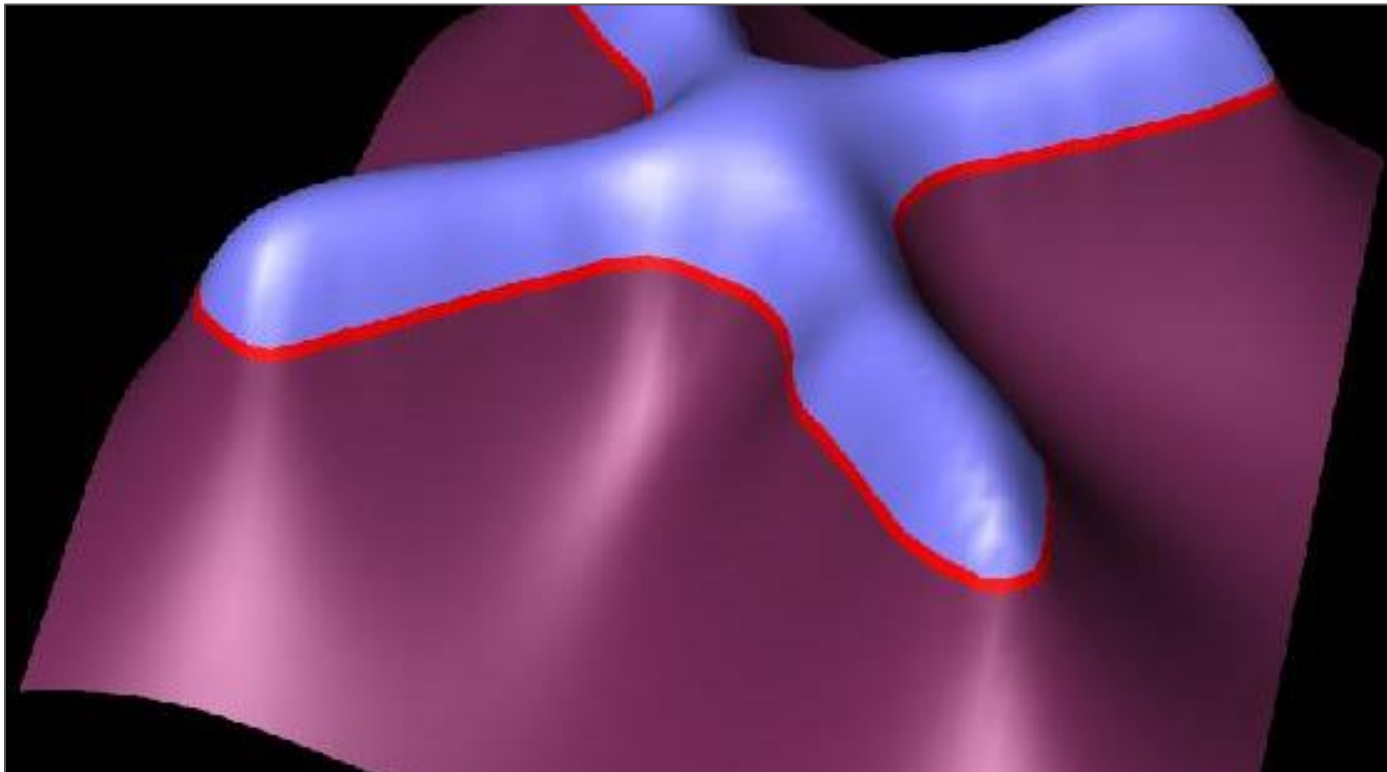
Implicit reconstruction:

estimate a signed distance function (SDF)

extract zero set



Implicit Curves and Surfaces



Implicit Curves and Surfaces

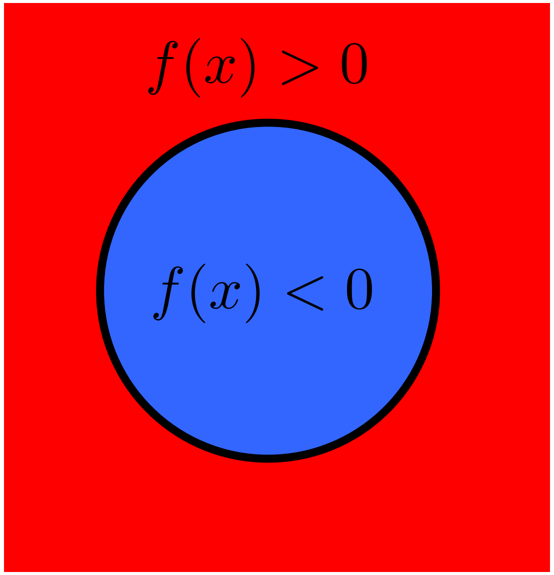
- Zero set of a scalar function $f : \mathbb{R}^m \to \mathbb{R}$
  - Curve in 2D: $S = \{x \in \mathbb{R}^2 | f(x) = 0\}$
  - Surface in 3D: $S = \{x \in \mathbb{R}^3 | f(x) = 0\}$

- Space partitioning

$\{x \in \mathbb{R}^m | f(x) > 0\}$ Outside

$\{x \in \mathbb{R}^m | f(x) = 0\}$ Curve/Surface

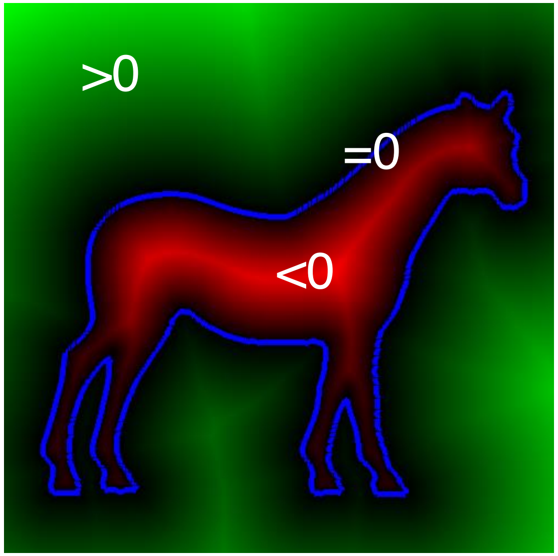
$\{x \in \mathbb{R}^m | f(x) < 0\}$ Inside



Implicit Curves and Surfaces

- Zero set of a scalar function $f : \mathbb{R}^m \to \mathbb{R}$
  - Curve in 2D: $S = \{x \in \mathbb{R}^2 | f(x) = 0\}$
  - Surface in 3D: $S = \{x \in \mathbb{R}^3 | f(x) = 0\}$

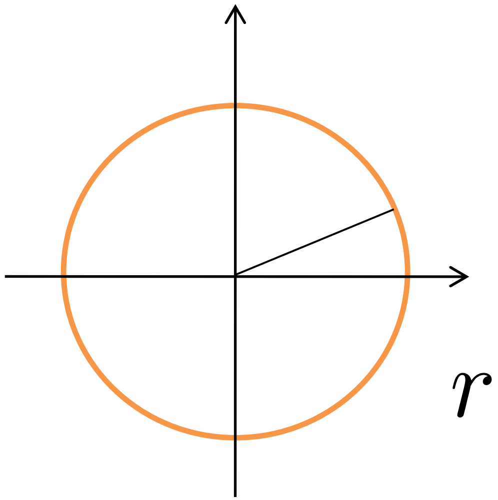
- Zero level set of signed distance function

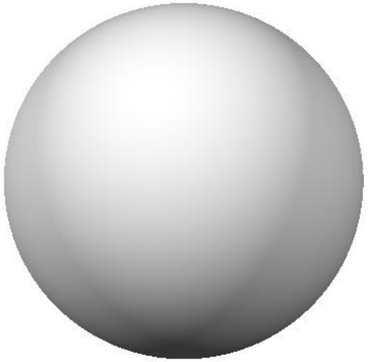


ETH zürich

Implicit Curves and Surfaces

- Implicit circle and sphere





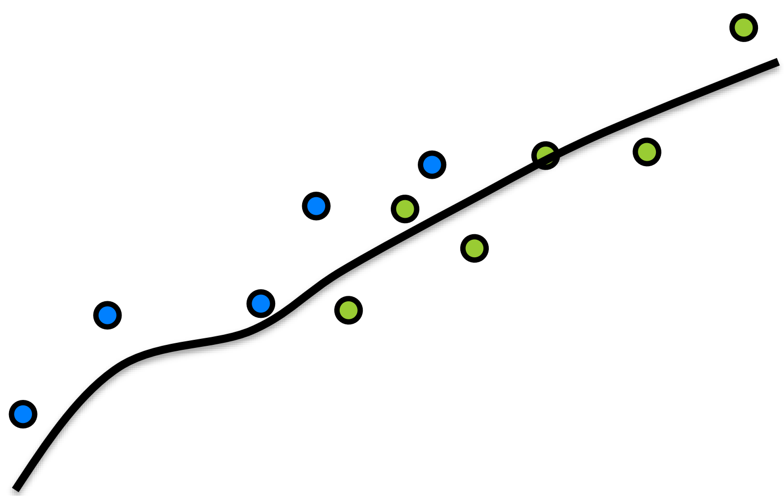
$$f(x, y) = x^2 + y^2 - r^2 \qquad f(x, y, z) = x^2 + y^2 + z^2 - r^2$$

How to Connect the Dots?

Implicit reconstruction:

estimate a signed distance function (SDF)

extract zero set

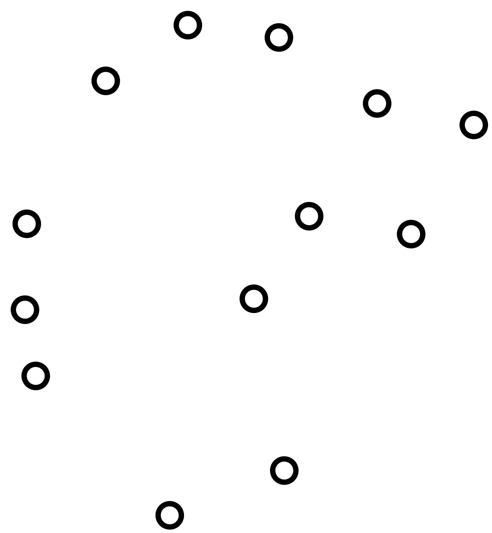


How to Connect the Dots?

Implicit reconstruction:

estimate a signed distance function (SDF)

extract zero set



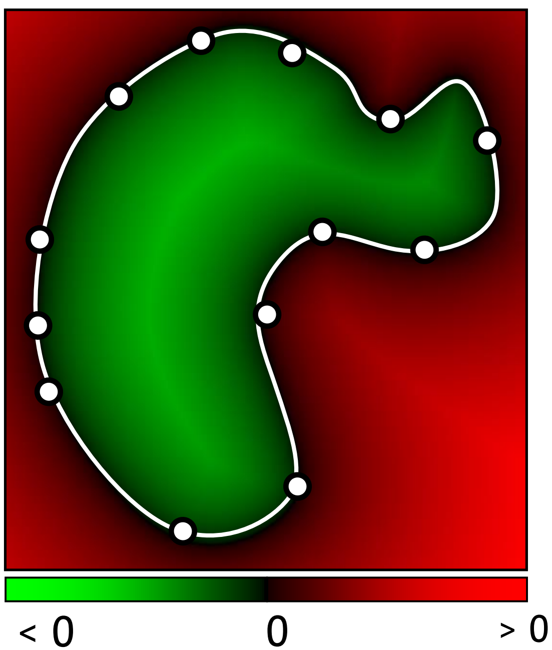
ETH zürich

How to Connect the Dots?

Implicit reconstruction:

estimate a signed distance function (SDF)

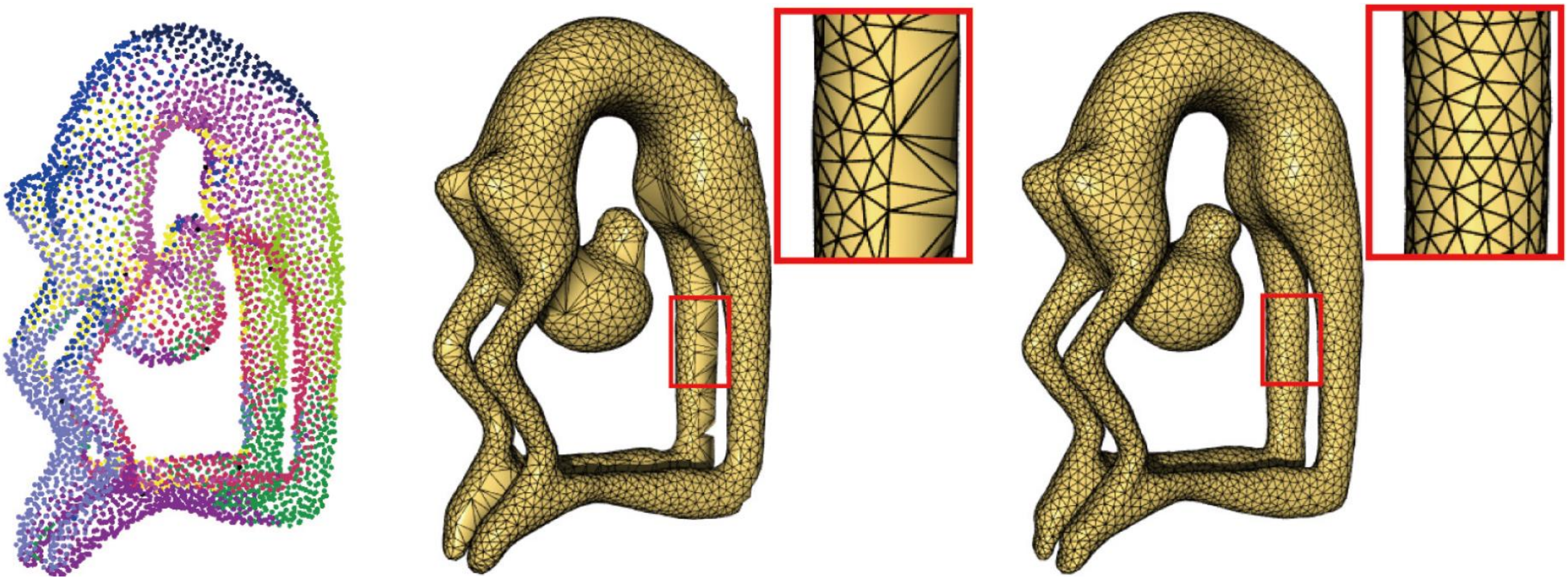
extract zero set



Advantages:

- Approximation of input points
- Watertight manifold results by construction

Implicit vs. Explicit

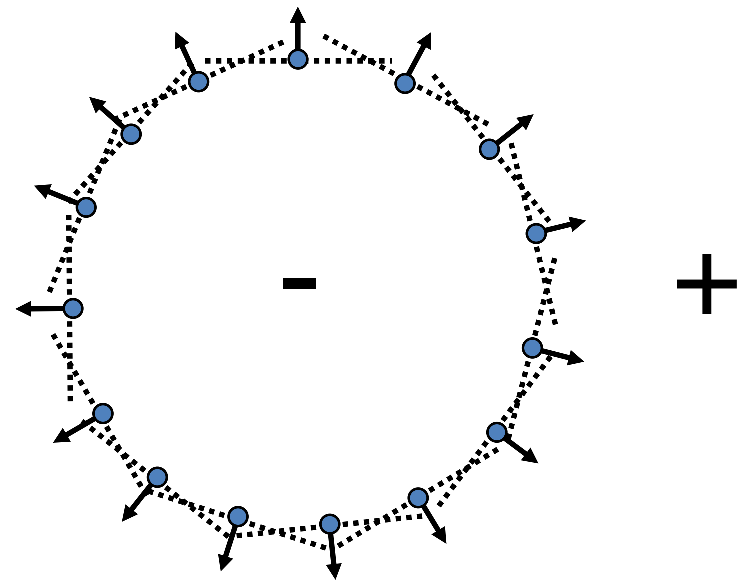


Input Explicit Implicit

SDF from Points and Normals

Compute signed distance to the tangent plane of the closest point

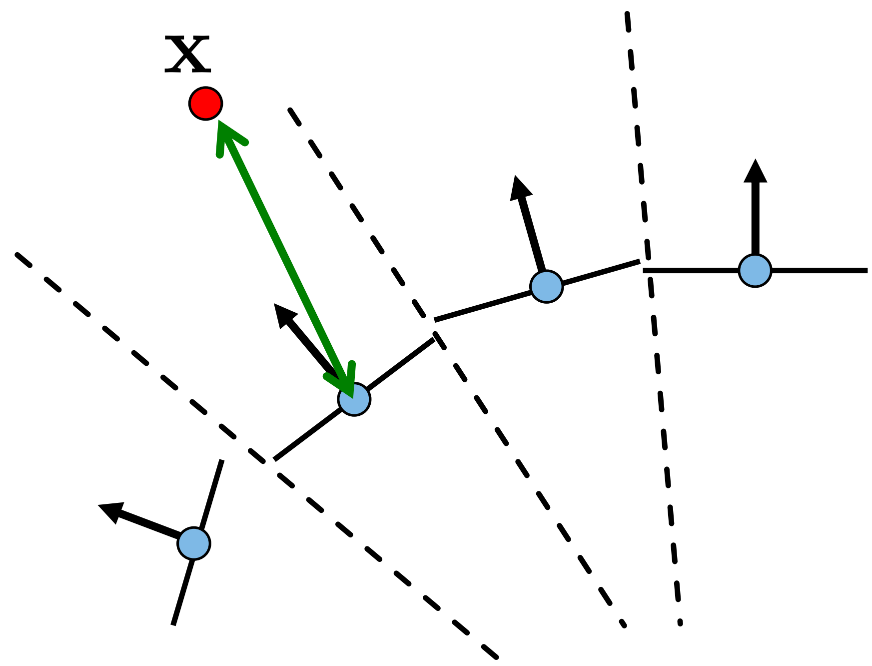
Normals will help to distinguish between inside and outside



“Surface reconstruction from unorganized points”, Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

SDF from Points and Normals

Compute signed distance to the tangent plane of the closest point

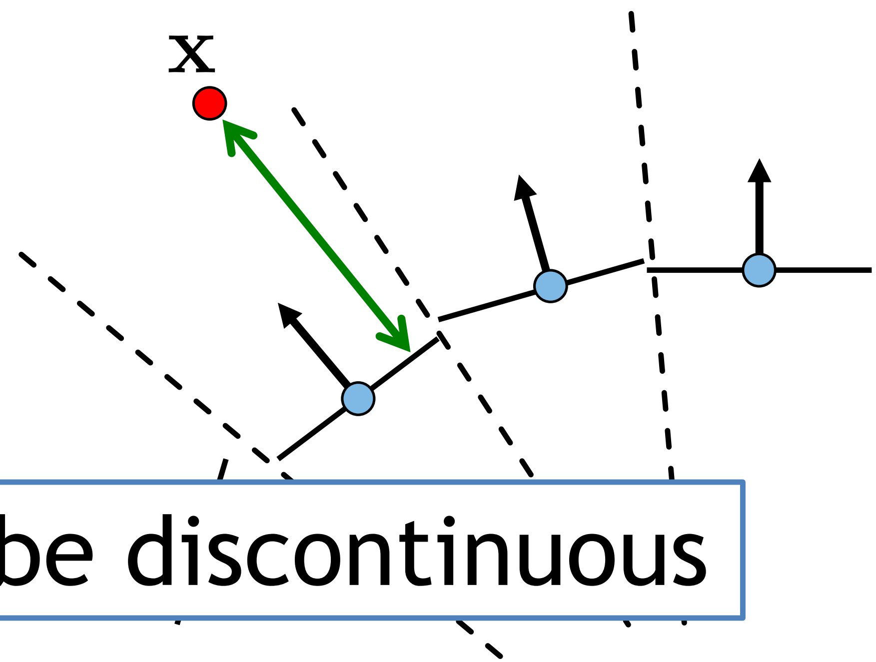


ETH zürich

SDF from Points and Normals

Compute signed distance to the tangent plane of the closest point

Problem?



The function will be discontinuous

Note: The Hoppe92 paper computes the tangent planes slightly differently (by PCA on k-nearest-neighbors of each data point, see next class), but the consequences are still the same.

Approximate SDF

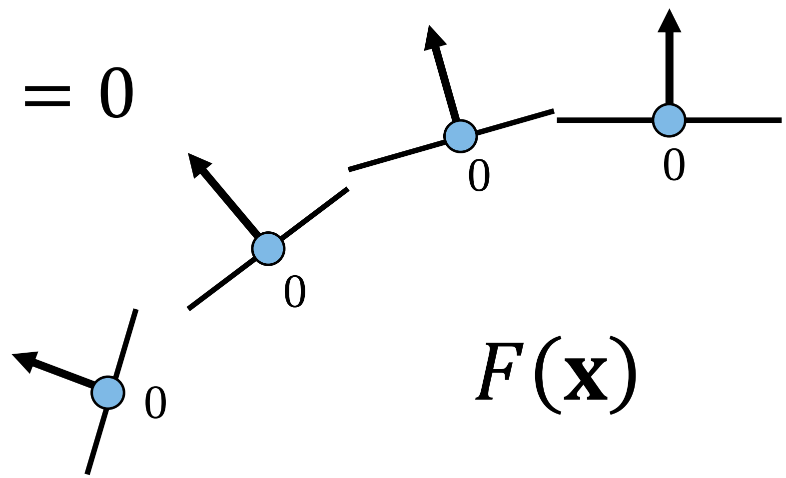
Pose problem as scattered data interpolation

Find a smooth F

$F(\mathbf{p}_i) = 0$

Avoid trivial solution $F(\mathbf{x}) = 0$

Add more constraints



"Reconstruction and representation of 3D objects with radial basis functions", Carr et al., ACM SIGGRAPH 2001

Approximate SDF

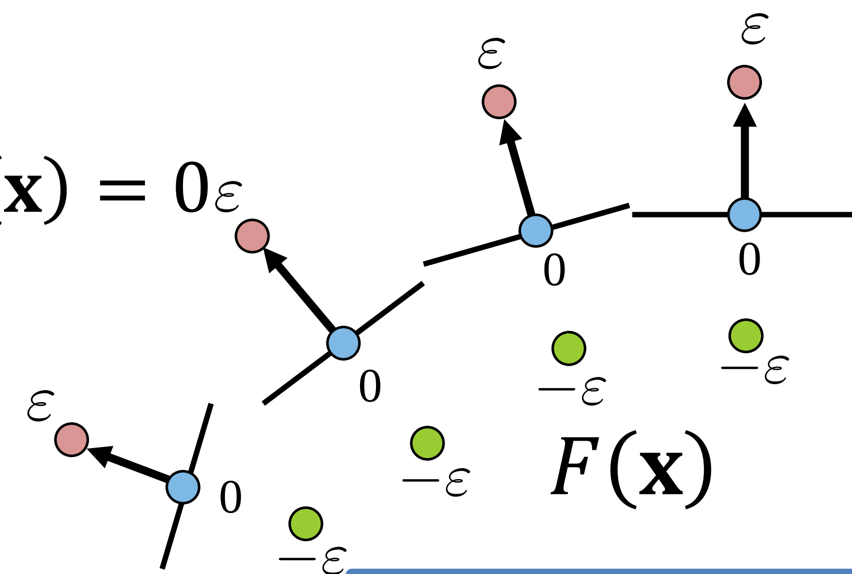
Pose problem as scattered data interpolation

Fina a smooth F

$F(\mathbf{p}_i) = 0$

Avoid trivial solution $F(\mathbf{x}) = 0$

Add more constraints



$$F(\mathbf{p}_i + \varepsilon \mathbf{n}_i) = \varepsilon$$

$$F(\mathbf{p}_i - \varepsilon \mathbf{n}_i) = -\varepsilon$$

"Reconstruction and representation of 3D objects with radial basis functions", Carr et al., ACM SIGGRAPH 2001

Radial Basis Function Interpolation

RBF

Weighted sum of shifted kernels

$$F(\mathbf{x}) = \sum_{m=0}^{N-1} w_m \, \varphi(\|\mathbf{x} - \mathbf{c}_m\|)$$

Centers

Kernel /basis function

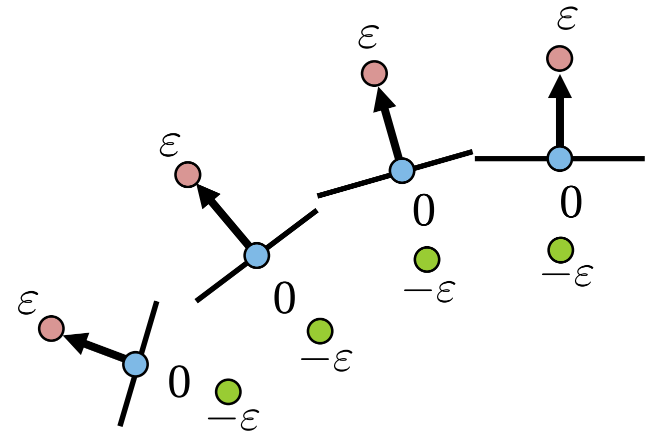
$\varphi(r) = r^3$

Scalar weights
Unknowns

Radial Basis Function Interpolation

Interpolate the constraints:

$$F(\mathbf{p}_i) = 0$$
$$F(\mathbf{p}_i + \varepsilon \mathbf{n}_i) = \varepsilon$$
$$F(\mathbf{p}_i - \varepsilon \mathbf{n}_i) = -\varepsilon$$



Set centers at:

$$\{\mathbf{c}_{3i}, \mathbf{c}_{3i+1}, \mathbf{c}_{3i+2}\} = \{\mathbf{p}_i,\ \mathbf{p}_i + \varepsilon \mathbf{n}_i,\ \mathbf{p}_i - \varepsilon \mathbf{n}_i\}$$

Find w_m $\sum_{m=0}^{N-1} w_m \, \varphi(\|\mathbf{c}_j - \mathbf{c}_m\|) = d_j \quad \forall j = 0, \ldots, N-1$

where

$$d_j = \begin{cases} 0 & j = 3i \\ \varepsilon & j = 3i+1 \\ -\varepsilon & j = 3i+2 \end{cases}$$

ETH zürich

Radial Basis Function Interpolation

Solve linear system to get the weights:

$$\begin{pmatrix} \varphi(\|\mathbf{c}_0 - \mathbf{c}_0\|) & \dots & \varphi(\|\mathbf{c}_0 - \mathbf{c}_{N-1}\| \\ \vdots & \ddots & \vdots \\ \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_0\|) & \dots & \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_{N-1}\|) \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_{N-1} \end{pmatrix} = \begin{pmatrix} d_0 \\ \vdots \\ d_{N-1} \end{pmatrix}$$

RBF Kernels

Triharmonic: $\varphi(r) = r^3$

- Globally supported
- Leads to dense symmetric linear system
- C^2 smoothness
- Works well for highly irregular sampling

RBF Kernels

Thin plate spline (polyharmonic)

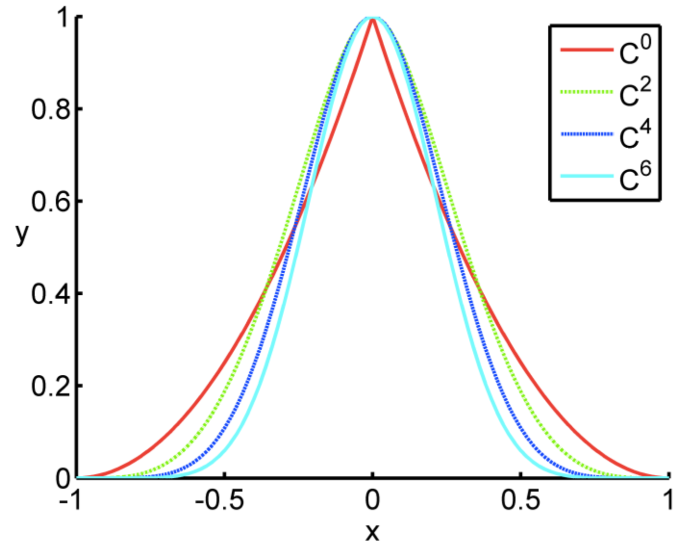
$$\varphi(r) = r^k \log(r),\ \ k = 2, 4, 6 \ldots$$
$$\varphi(r) = r^k,\ \ k = 1, 3, 5 \ldots$$

Multiquadratic

$$\varphi(r) = \sqrt{r^2 + \beta^2}$$

Gaussian

$$\varphi(r) = e^{-\beta r^2}$$



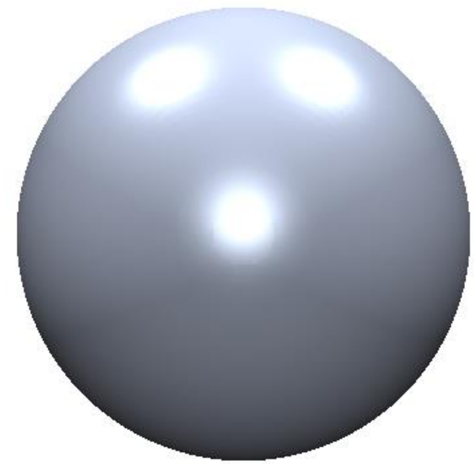
B-Spline (compact support)

$$\varphi(r) = \text{piecewise-polynomial}(r)$$

Comparison of the various SDFs so far

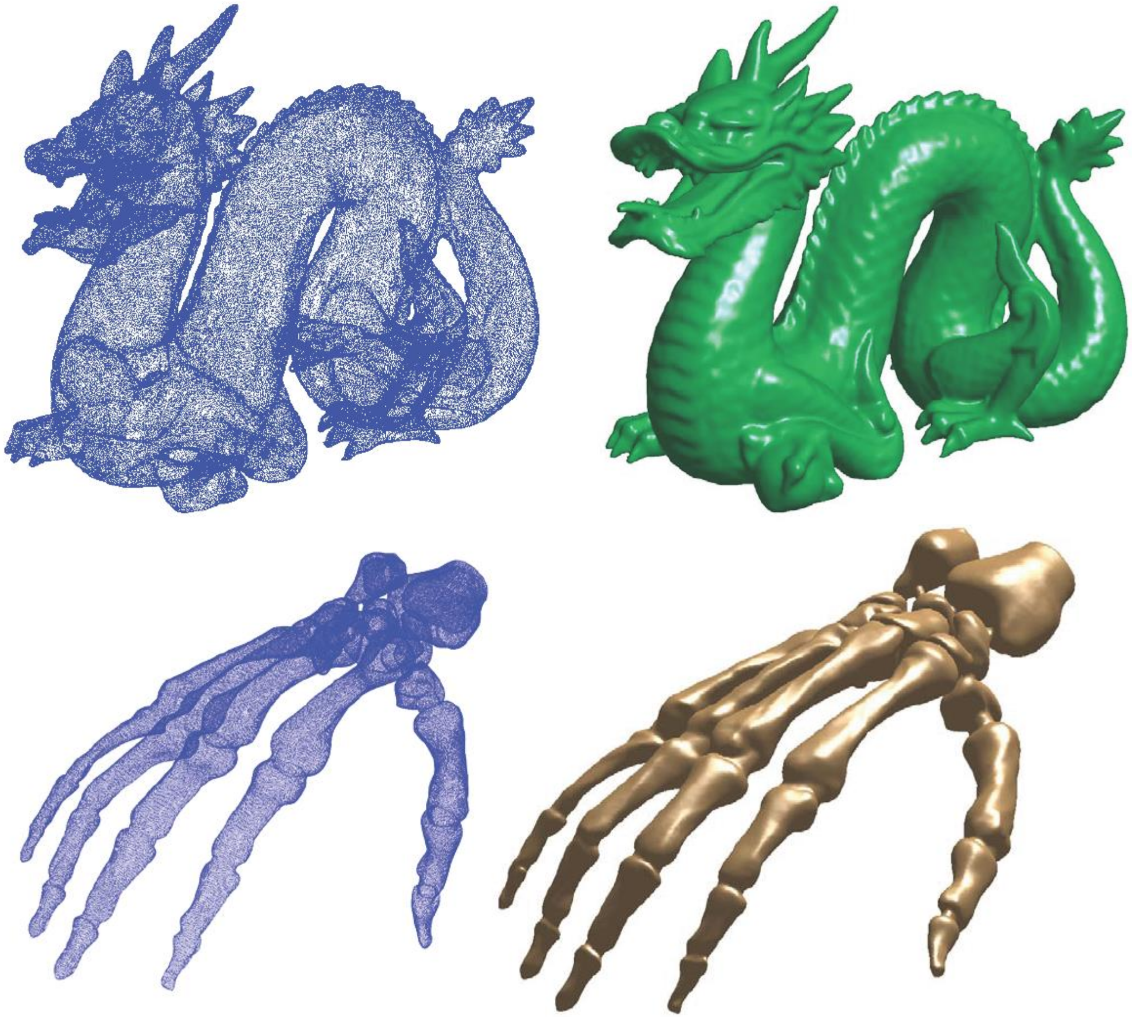
Distance
to plane

Local RBF



Global RBF
Triharmonic

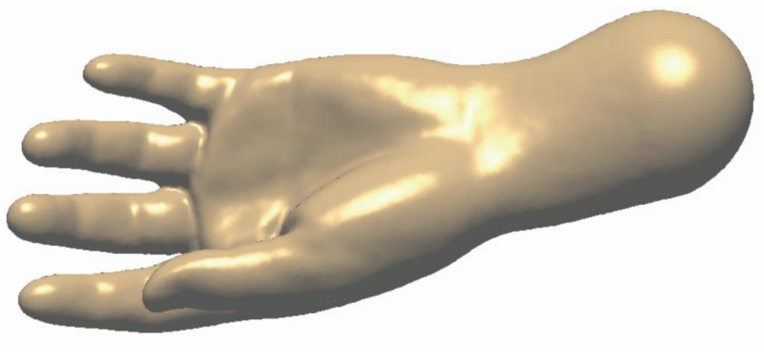
RBF Reconstruction Examples



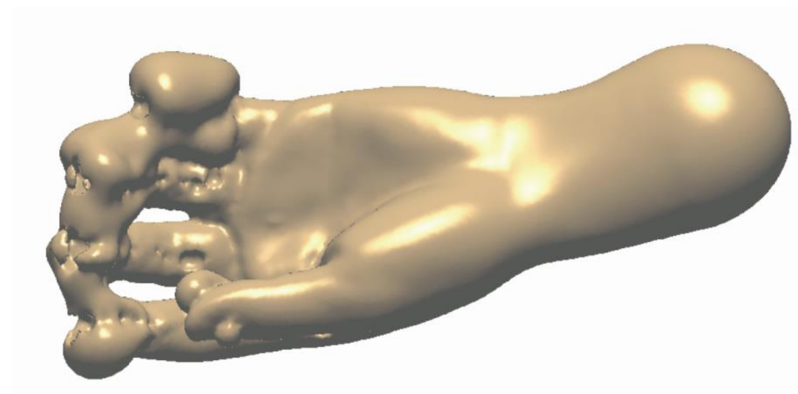
“Reconstruction and representation of 3D objects with radial basis functions”, Carr et al., ACM SIGGRAPH 2001

Off-Surface Points

Must pick the correct ε



Properly chosen off-surface points



Insufficient number/
badly placed off-surface points

“Reconstruction and representation of 3D objects with radial basis functions”, Carr et al., ACM SIGGRAPH 2001

RBF – Discussion

Pros: Global definition

- Single function
- Globally optimal

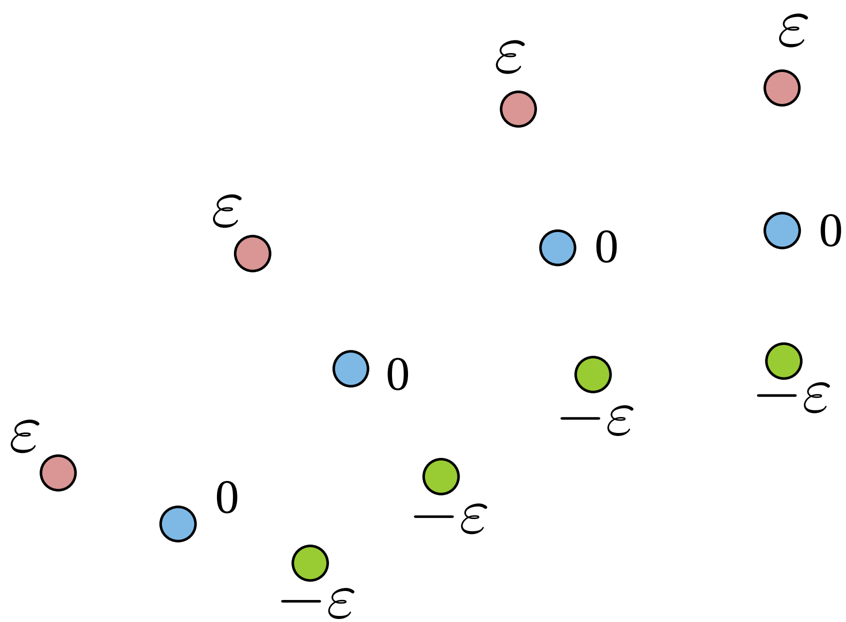
Cons: Global definition

- Global optimization – slow
- Why is global better?

Moving Least Squares (MLS)

Do purely **local** approximation of the SDF

Weights change depending on where we are evaluating

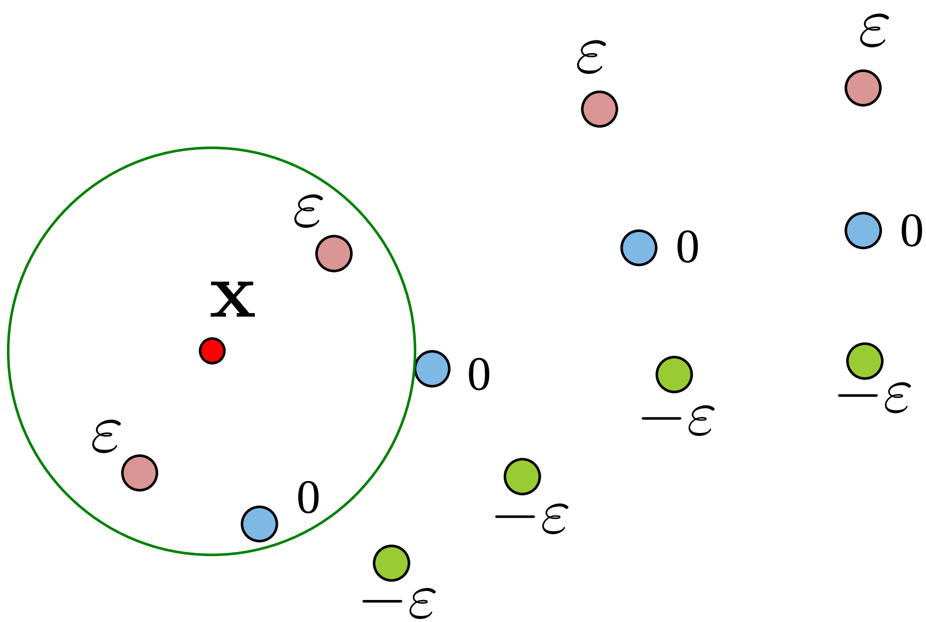


"Interpolating and Approximating Implicit Surfaces from Polygon Soup", Shen et al., ACM SIGGRAPH 2004
http://graphics.berkeley.edu/papers/Shen-IAI-2004-08/index.html

Moving Least Squares (MLS)

Do purely **local** approximation of the SDF

Weights change depending on where we are evaluating

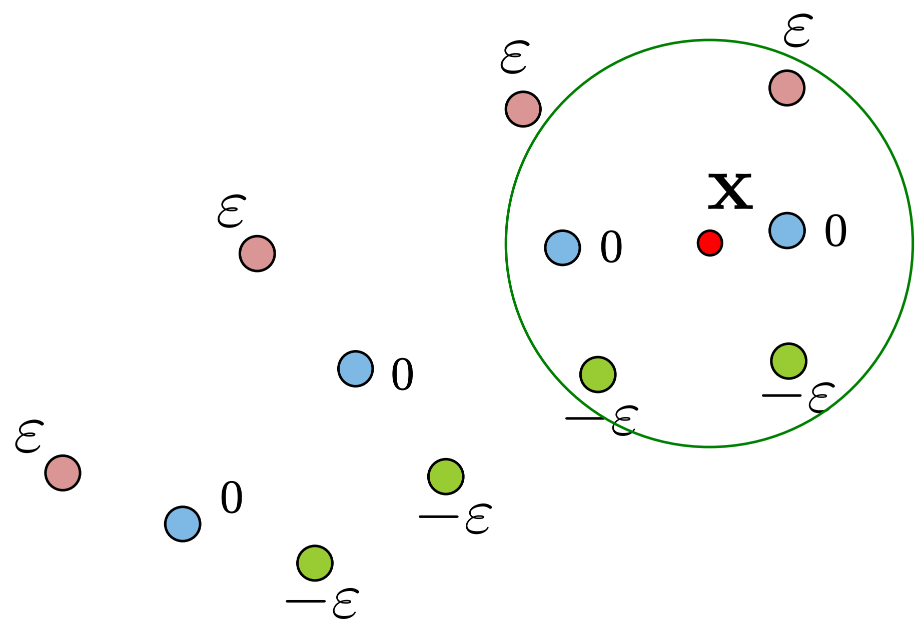


"Interpolating and Approximating Implicit Surfaces from Polygon Soup", Shen et al., ACM SIGGRAPH 2004
http://graphics.berkeley.edu/papers/Shen-IAI-2004-08/index.html

Moving Least Squares (MLS)

Do purely **local** approximation of the SDF

Weights change depending on where we are evaluating

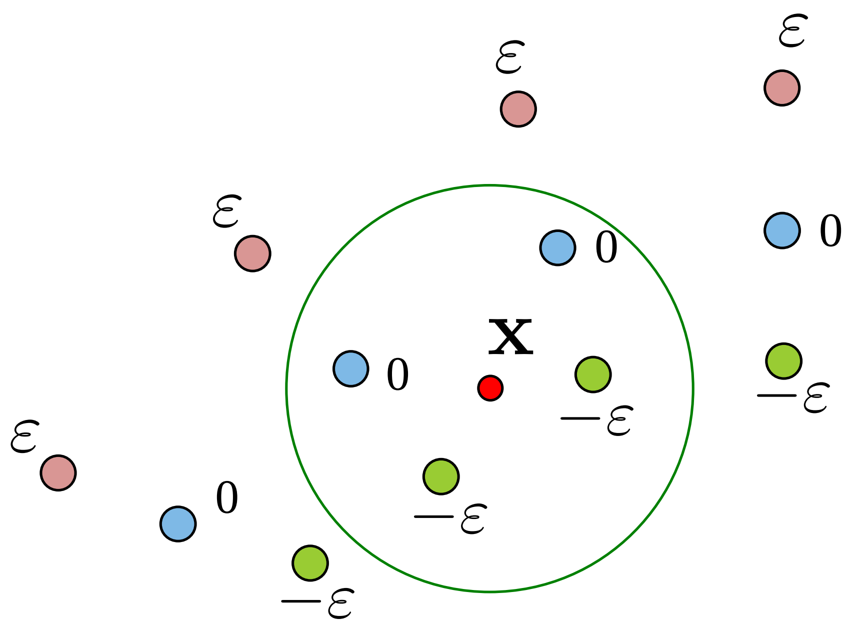


"Interpolating and Approximating Implicit Surfaces from Polygon Soup", Shen et al., ACM SIGGRAPH 2004
http://graphics.berkeley.edu/papers/Shen-IAI-2004-08/index.html

Moving Least Squares (MLS)

Do purely **local** approximation of the SDF

Weights change depending on where we are evaluating

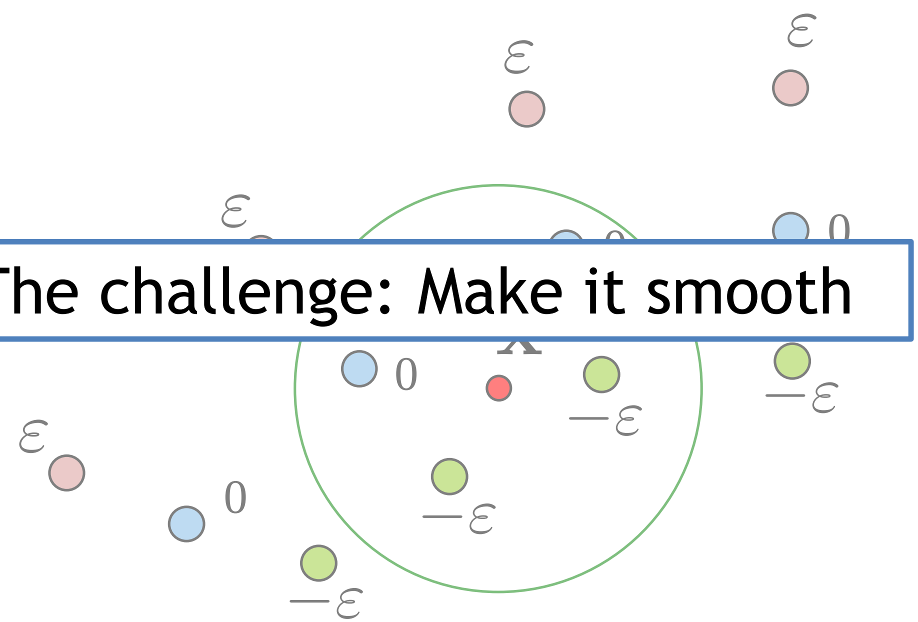


"Interpolating and Approximating Implicit Surfaces from Polygon Soup", Shen et al., ACM SIGGRAPH 2004
http://graphics.berkeley.edu/papers/Shen-IAI-2004-08/index.html

Moving Least Squares (MLS)

Do purely **local** approximation of the SDF

Weights change depending on where we are evaluating



The challenge: Make it smooth

"Interpolating and Approximating Implicit Surfaces from Polygon Soup", Shen et al., ACM SIGGRAPH 2004
http://graphics.berkeley.edu/papers/Shen-IAI-2004-08/index.html

Least-Squares Approximation

Polynomial least-squares approximation

$$f \in \Pi_k^3 : f(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 xy + \ldots + a_* z^k$$

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{a}$$

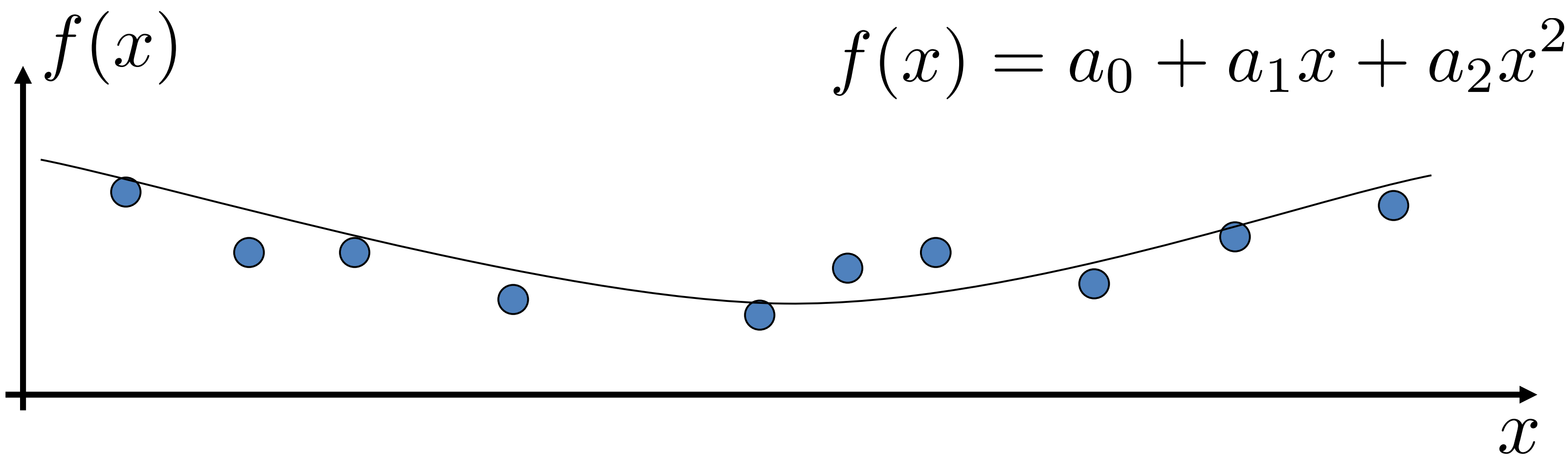
$$\mathbf{a} = (a_1, a_2, \ldots, a_*)^T,\ \mathbf{b}(\mathbf{x})^T = (1, x, y, z, x^2, xy, \ldots, z^k)$$

Find $\mathbf{a}$ that minimizes sum of squared differences

$$\operatorname*{argmin}_{\mathbf{a}} \sum_{m=0}^{N-1} \left(\mathbf{b}(\mathbf{c}_m)^T \mathbf{a} - d_m\right)^2$$

MLS – 1D Example

- Global approximation in Π_2^1



$$\underset{\mathbf{a}}{\operatorname{argmin}} \sum_{m=0}^{N-1} \left(\mathbf{b}(\mathbf{c}_m)^T \mathbf{a} - d_m\right)^2$$

Moving Least-Squares Approximation

Polynomial least-squares approximation

$$f \in \Pi_k^3 : f(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 xy + \ldots + a_* z^k$$

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{a}$$

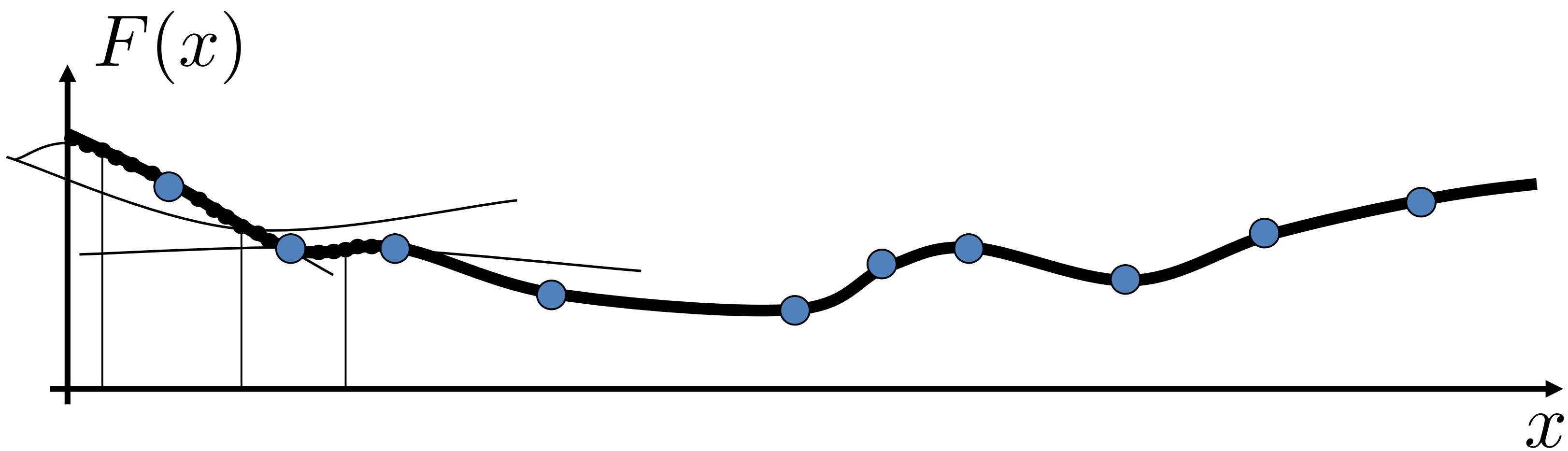
$$\mathbf{a} = (a_1, a_2, \ldots, a_*)^T, \ \mathbf{b}(\mathbf{x})^T = (1, x, y, z, x^2, xy, \ldots, z^k)$$

Find $\mathbf{a}$ that minimizes weighted sum of squared differences

$$\mathbf{a_x} = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{m=0}^{N-1} \theta(\|\mathbf{x} - \mathbf{c}_m\|) \left(\mathbf{b}(\mathbf{c}_m)^T \mathbf{a} - d_m\right)^2$$

MLS – 1D Example

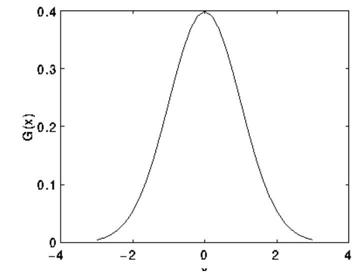
- MLS approximation using functions in Π_2^1



$$F(x) = f_x(x), \quad f_x = \underset{f \in \Pi_2^1}{\operatorname{argmin}} \sum_{m=0}^{N-1} \theta(\|c_m - x\|) \left(f(c_m) - d_m\right)^2$$

Weight Functions

Gaussian $\theta(r) = e^{-\frac{r^2}{h^2}}$



h is a smoothing parameter

Wendland function $\theta(r) = (1 - r/h)^4(4r/h + 1)$

Defined in $[0, h]$ and

$$\theta(0) = 1,\ \theta(h) = 0,\ \theta'(h) = 0,\ \theta''(h) = 0$$

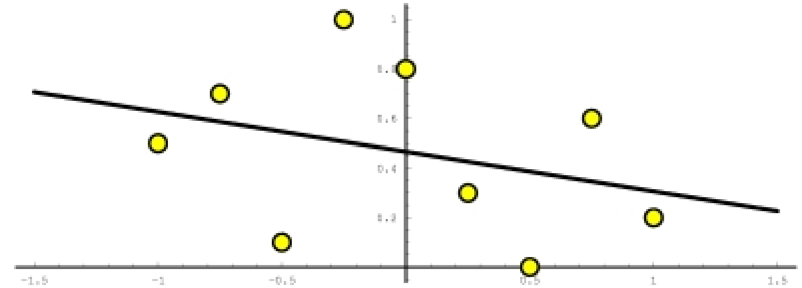
Singular function

$$\theta(r) = \frac{1}{r^2 + \varepsilon^2}$$

For small ε, weights are large near r=0 (interpolation)

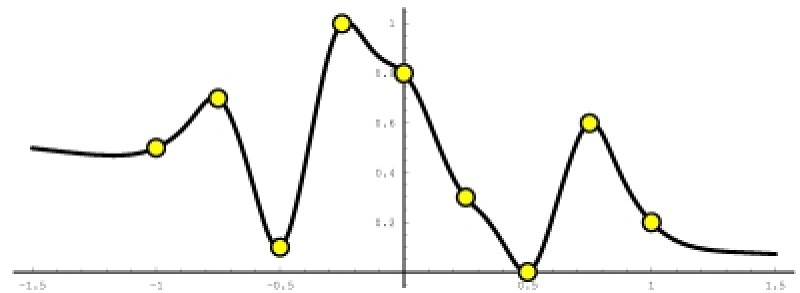
Dependence on Weight Function

Global least squares with linear basis



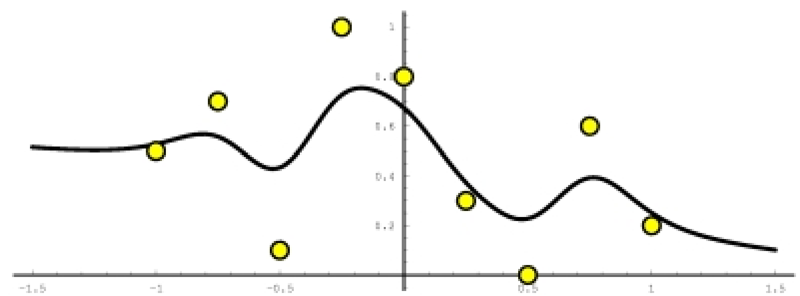
MLS with (nearly) singular weight function

$$\theta(r) = \frac{1}{r^2 + \varepsilon^2}$$



MLS with approximating weight function

$$\theta(r) = e^{-\frac{r^2}{h^2}}$$



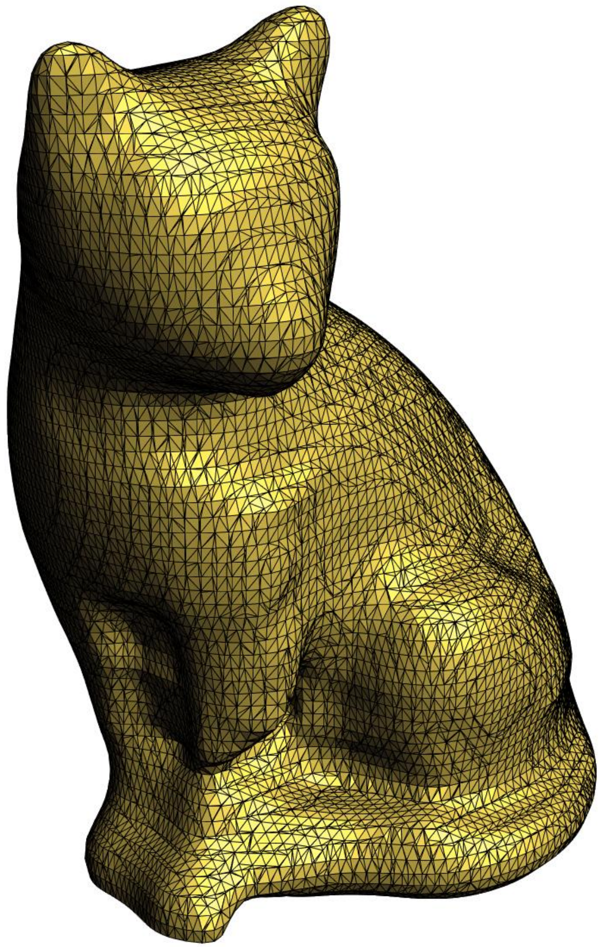
Dependence on Weight Function

The MLS function F is continuously differentiable if and only if the weight function θ is continuously differentiable

In general, F is as smooth as θ

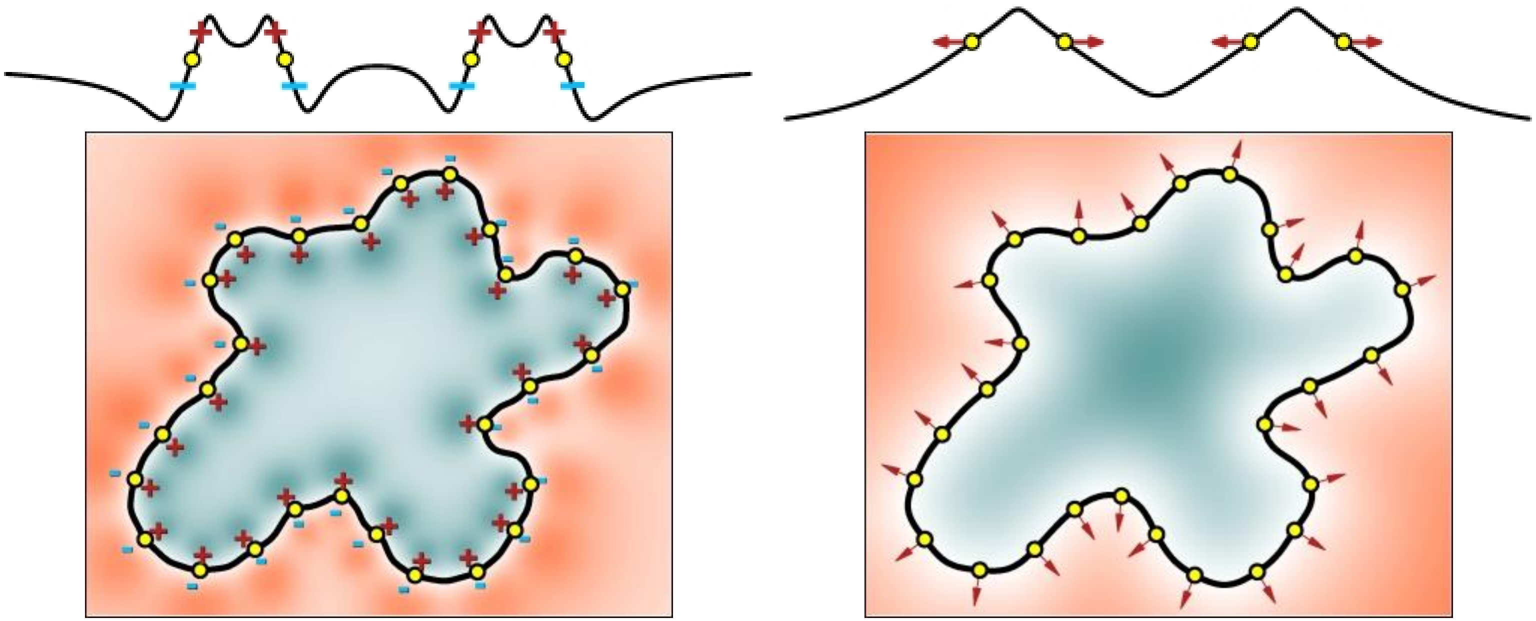
$$F(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}), \quad f_{\mathbf{x}} = \underset{f \in \Pi_k^d}{\operatorname{argmin}} \sum_{m=0}^{N-1} \theta(\|\mathbf{c}_m - \mathbf{x}\|) \left(f(\mathbf{c}_m) - d_m\right)^2$$

Example: Reconstruction



MLS SDF – Possible Improvement

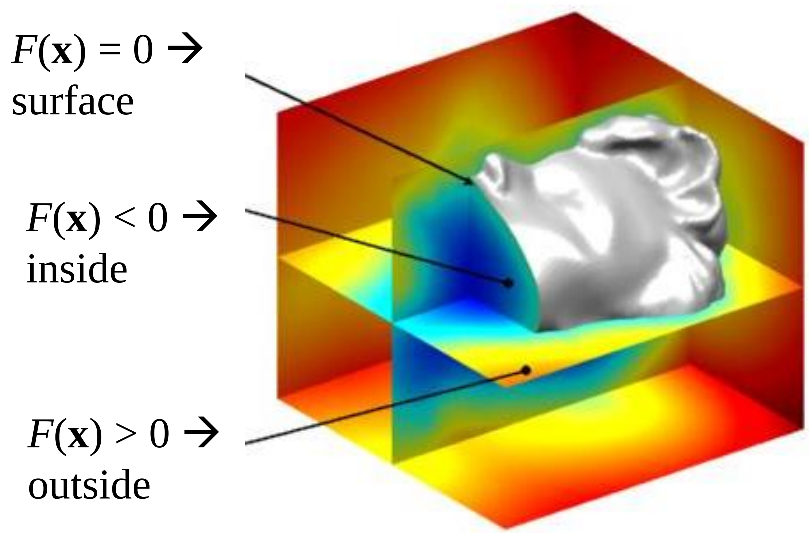
- Point constraints vs. true normal constraints



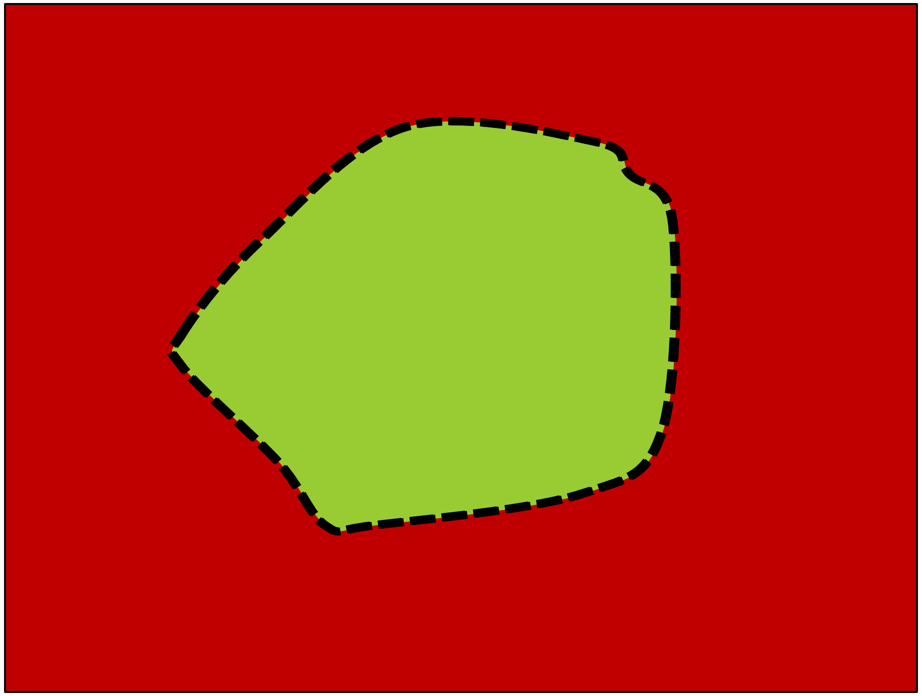
- Details: see [Shen et al. SIGGRAPH 2004] and the bonus assignment in Ex2

Extracting the Surface

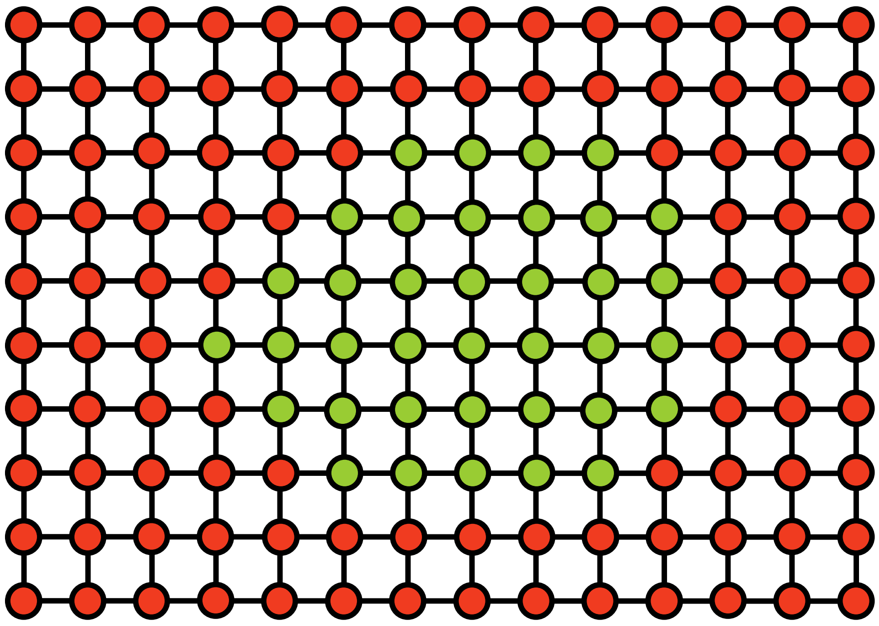
How to find a mesh of the level set?

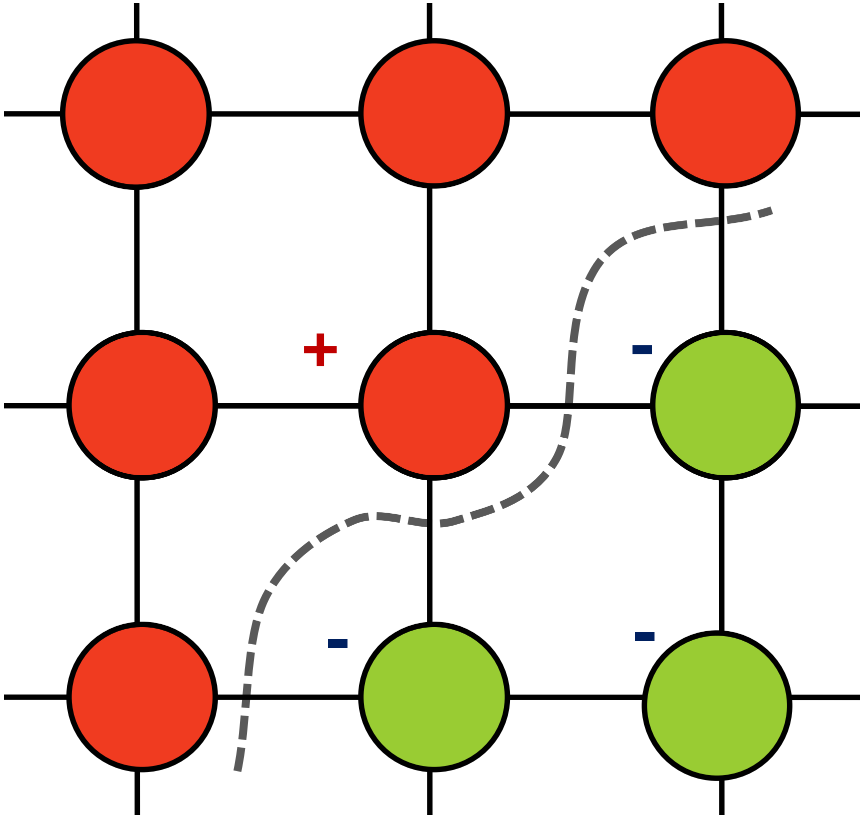


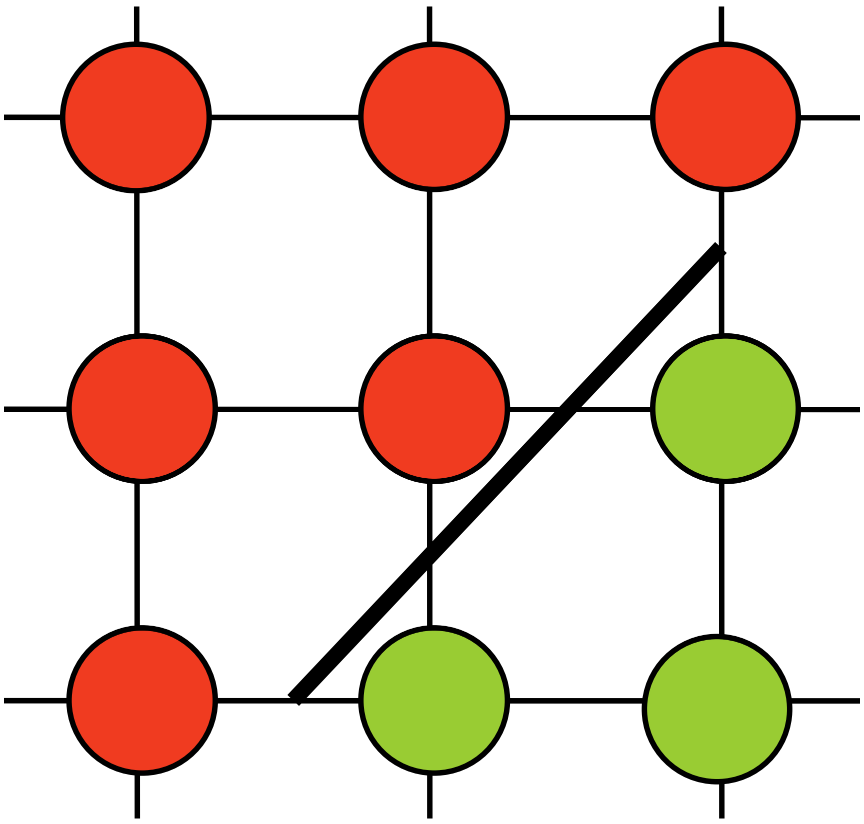
Sample the SDF



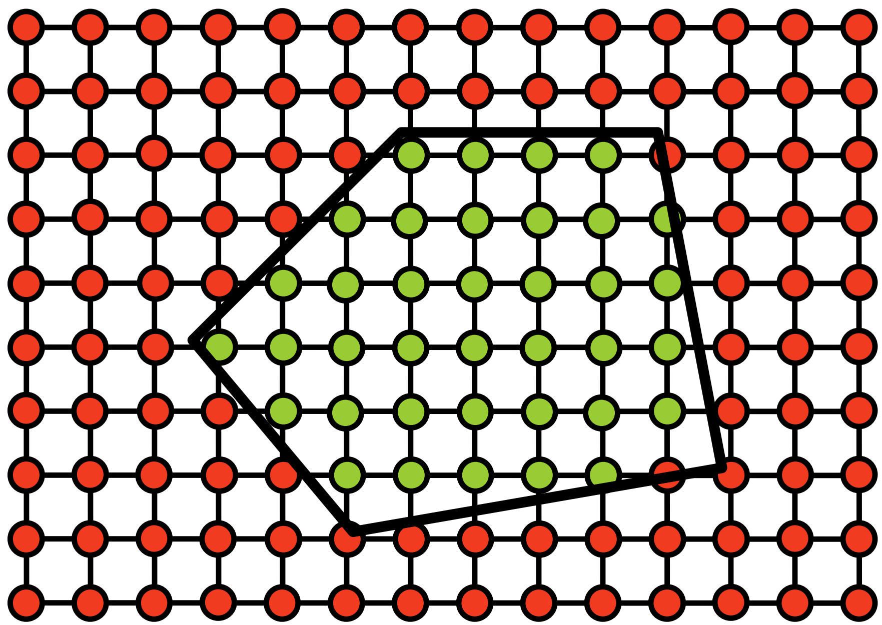
Sample the SDF







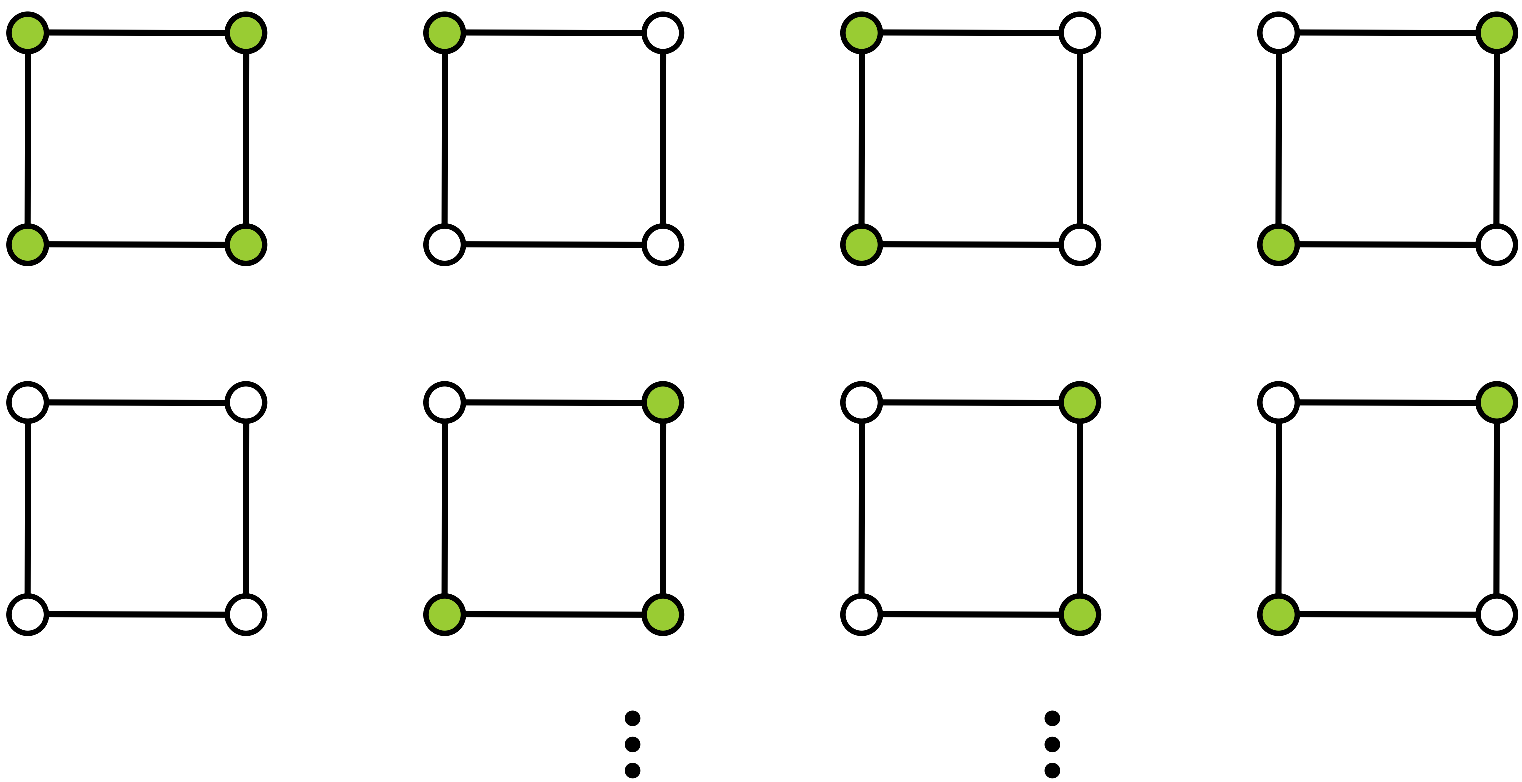
Sample the SDF



Marching Squares

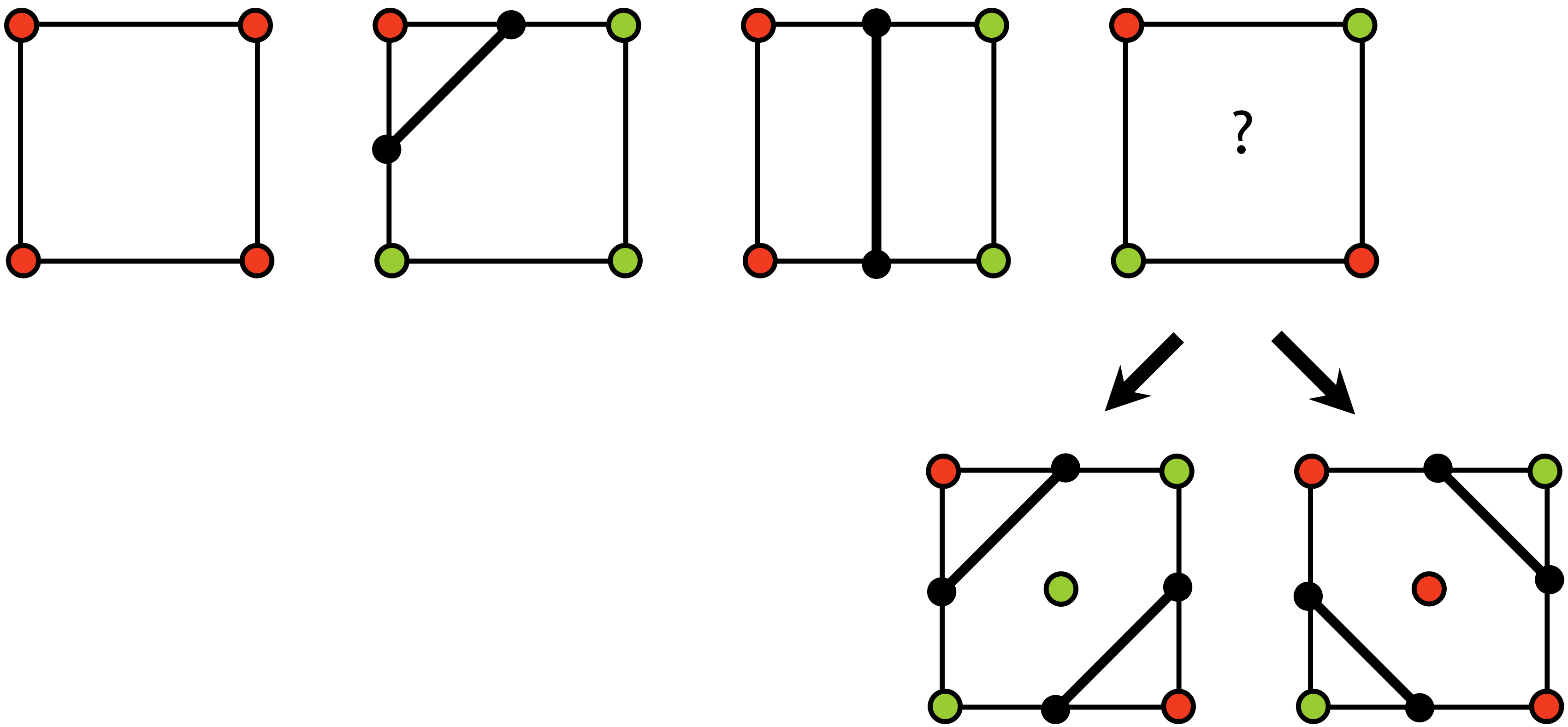
16 different configurations in 2D

4 classes (rotation, reflection, negation)



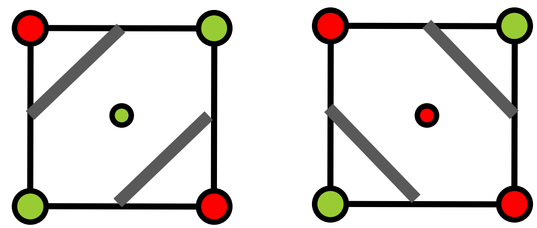
Tessellation in 2D

4 classes (rotation, reflection, negation)

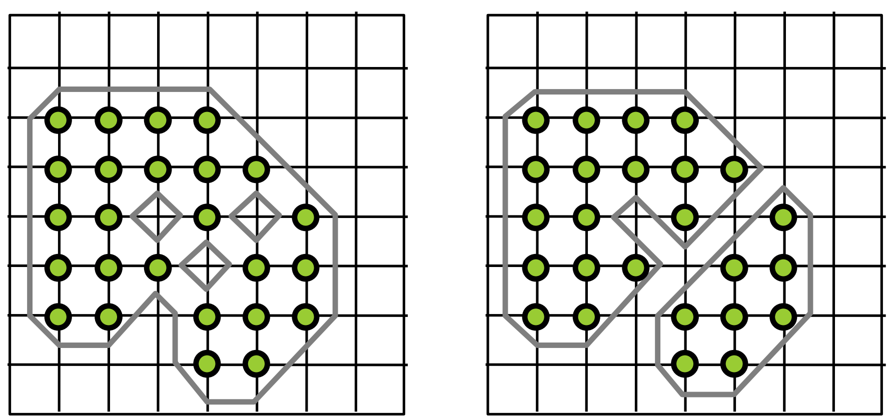


Tessellation in 2D

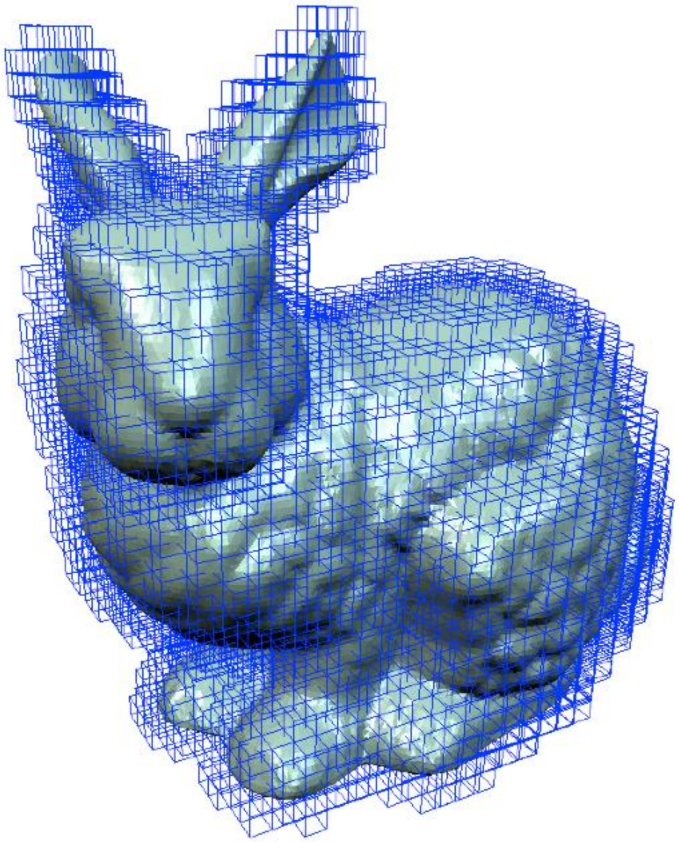
Case 4 is ambiguious:

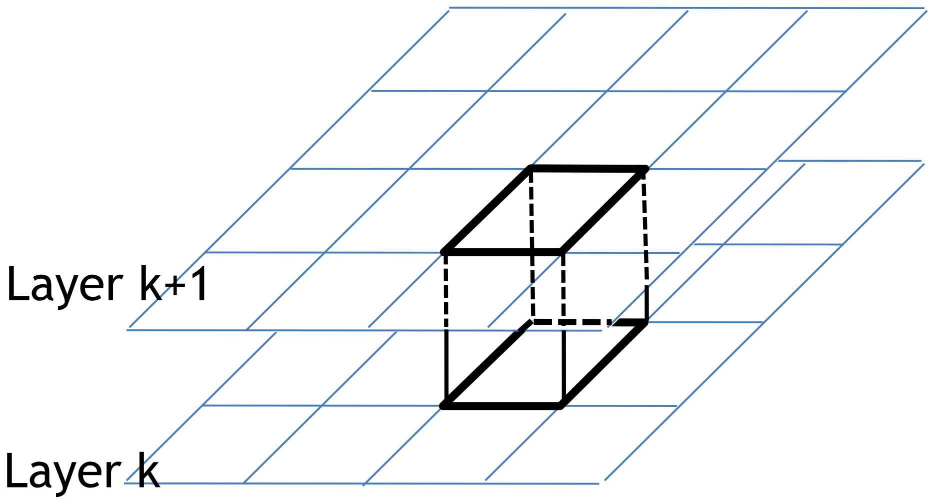


Always pick consistently



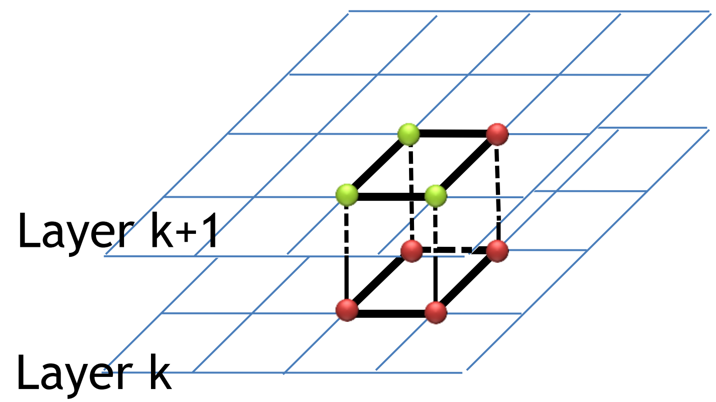
3D: Marching Cubes





Marching Cubes

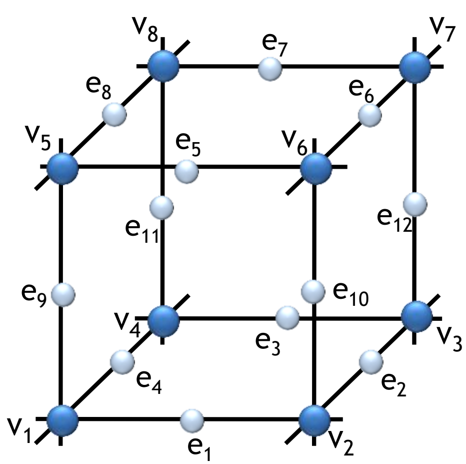
- Marching Cubes (Lorensen and Cline 1987)
  1. Load 4 layers of the grid into memory
  2. Create a cube whose vertices lie on the two middle layers
  3. Classify the vertices of the cube according to the implicit function (inside, outside or on the surface)

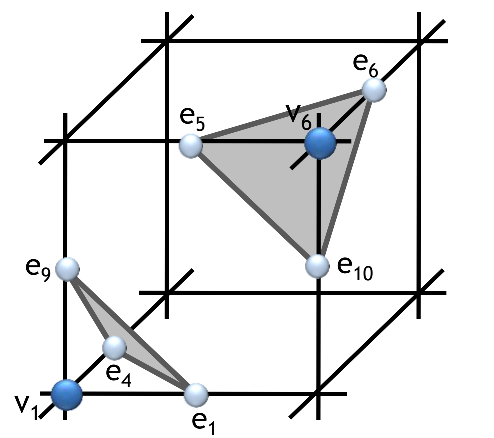


ETH zürich

Marching Cubes

4. Compute case index. We have $2^8 = 256$ cases (0/1 for each of the eight vertices) – can store as 8 bit (1 byte) index.



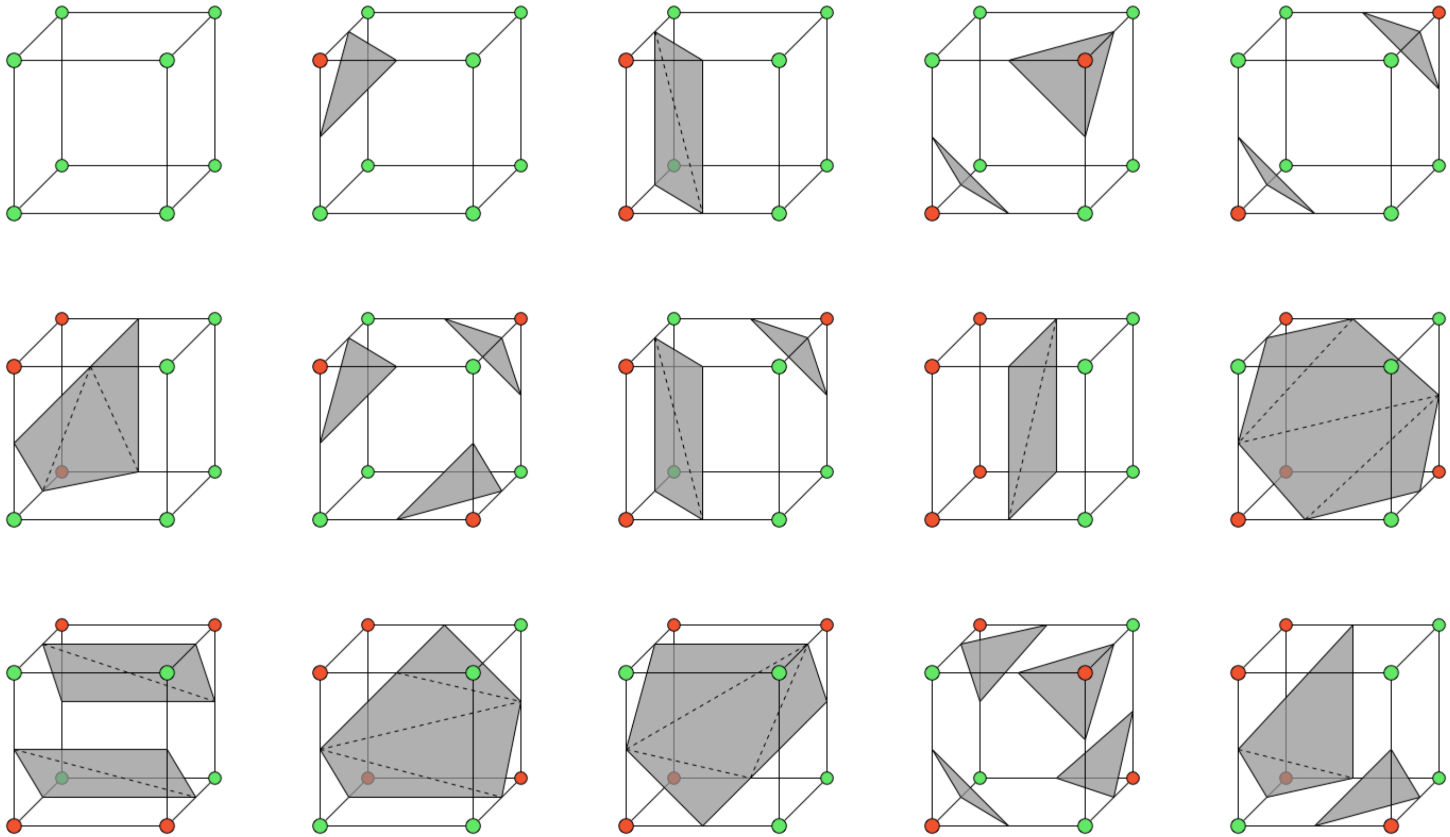


index = | v_1 | v_2 | v_3 | v_4 | v_5 | v_6 | v_7 | v_8 |

index = | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | = 33

Marching Cubes

- Unique cases (by rotation, reflection and negation)

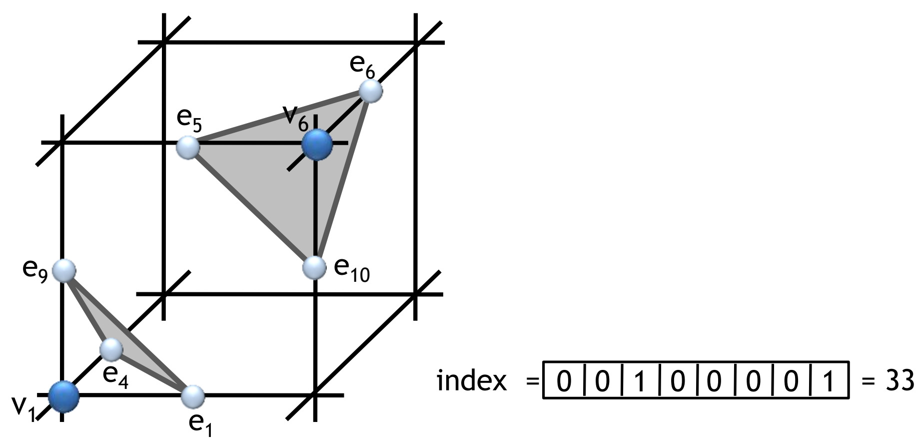


Tessellation

3D – Marching Cubes

5. Using the case index, retrieve the connectivity in the look-up table

- Example: the entry for index 33 in the look-up table indicates that the cut edges are e_1; e_4; e_5; e_6; e_9 and e_{10} ; the output triangles are (e_1; e_9; e_4) and (e_5; e_{10}; e_6).



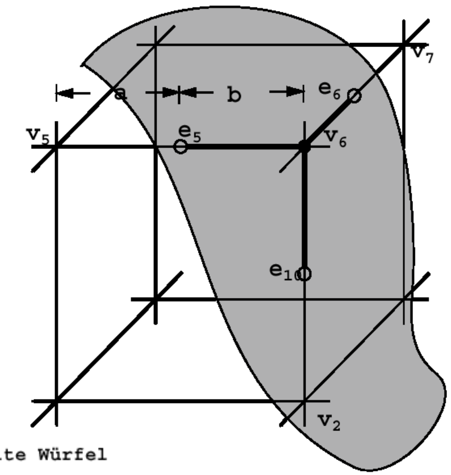
Marching Cubes

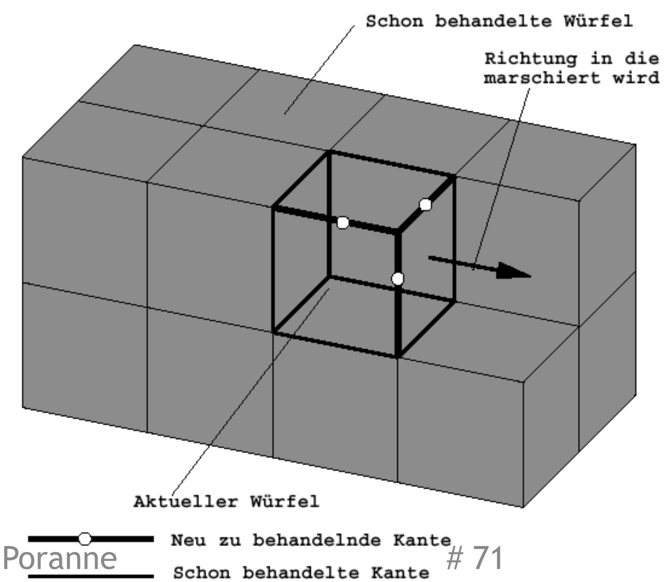
6. Compute the position of the cut vertices by linear interpolation:

$$\mathbf{v}_s = t\mathbf{v}_a + (1-t)\mathbf{v}_b$$

$$t = \frac{F(\mathbf{v}_b)}{F(\mathbf{v}_b) - F(\mathbf{v}_a)}$$

7. Move to the next cube

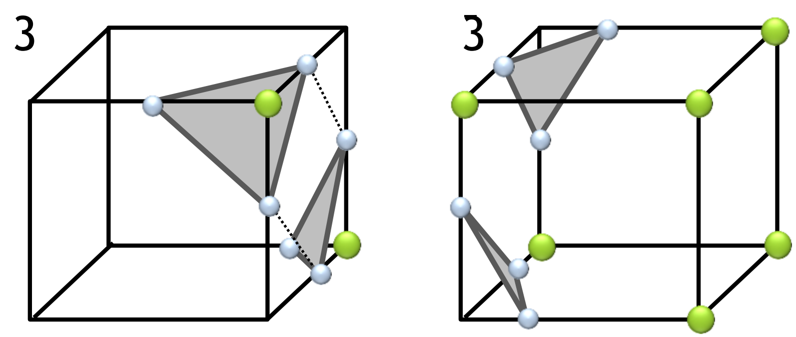


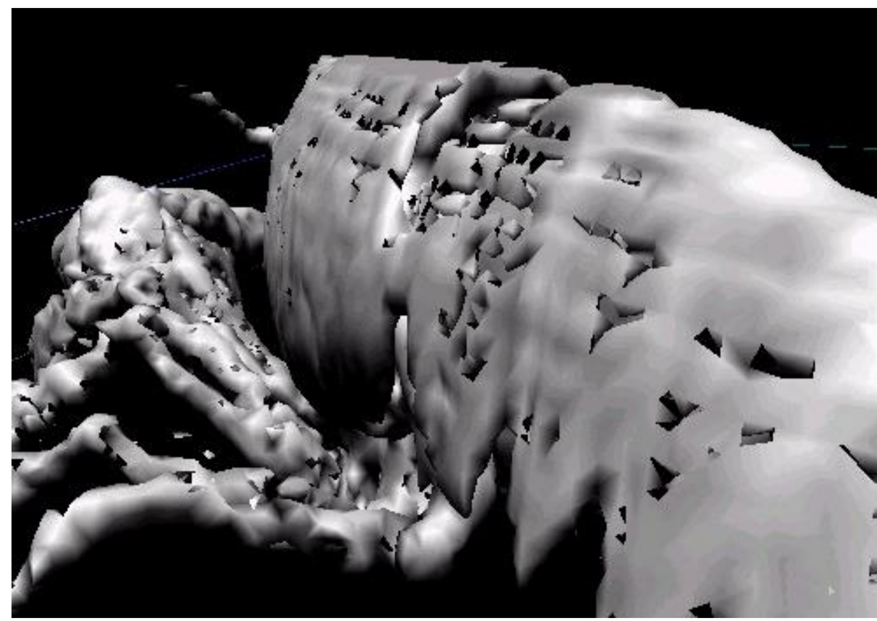


ETH zürich

Marching Cubes – Problems

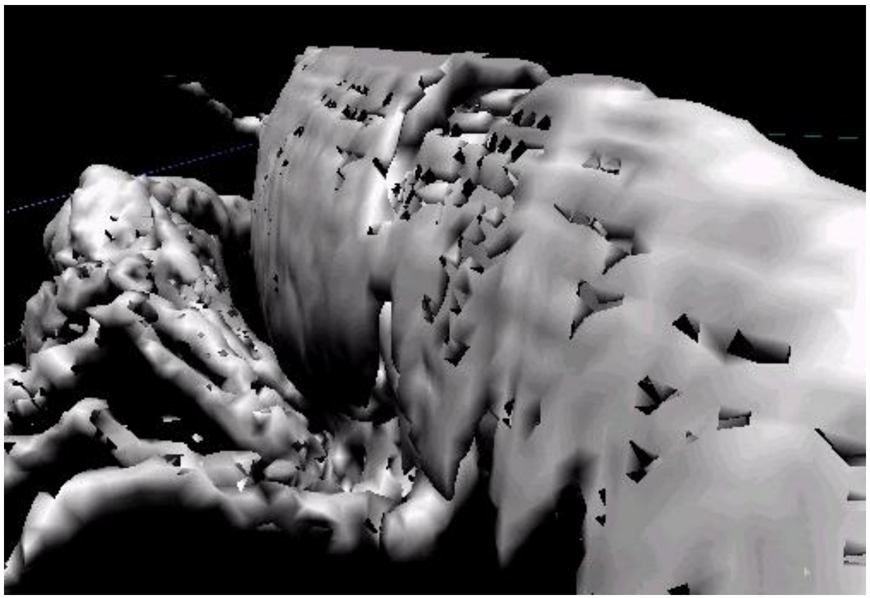
- Have to make consistent choices for neighboring cubes – otherwise get holes



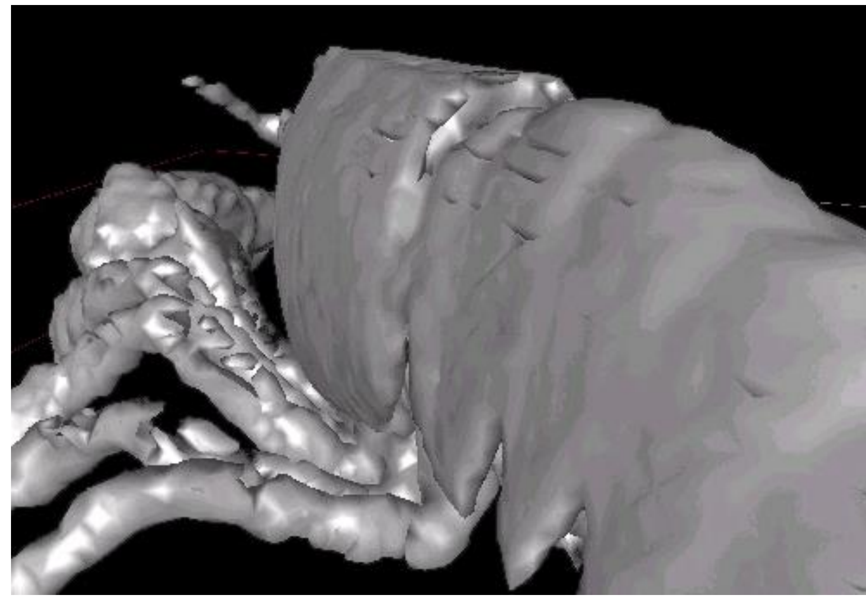


Marching Cubes – Problems

- Resolving ambiguities



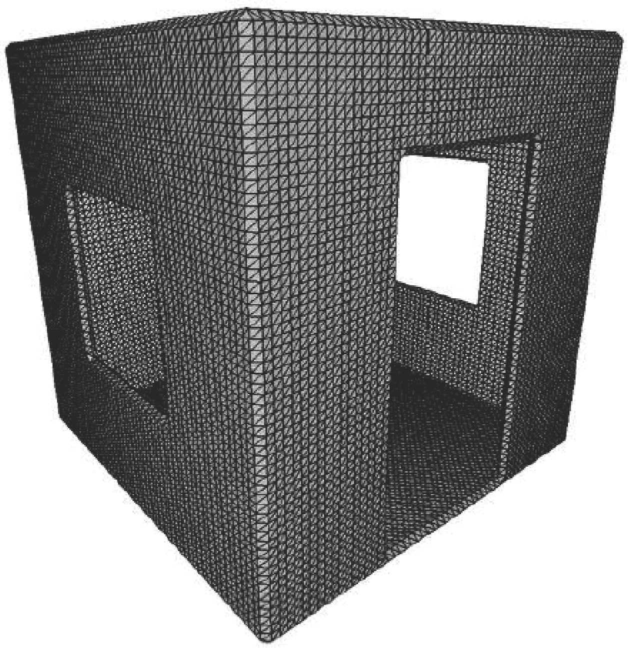
Ambiguity



No Ambiguity

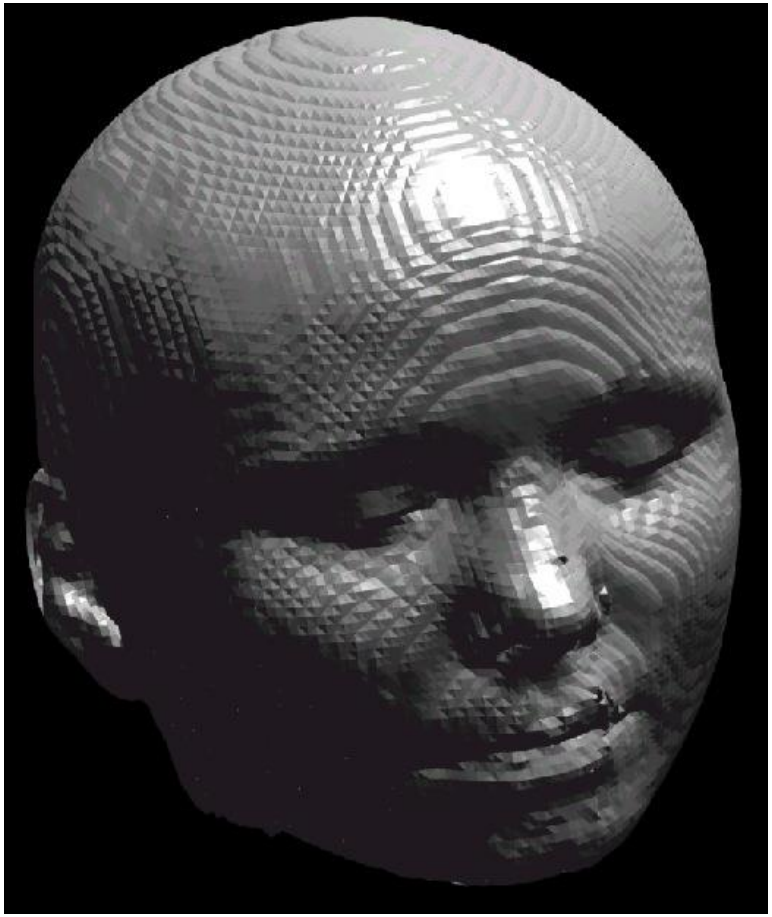
Marching Cubes – Problems

- Grid not adaptive
- Many polygons required to represent small features



Images from: “Dual Marching Cubes: Primal Contouring of Dual Grids” by Schaeffer et al.

Marching Cubes – Problems



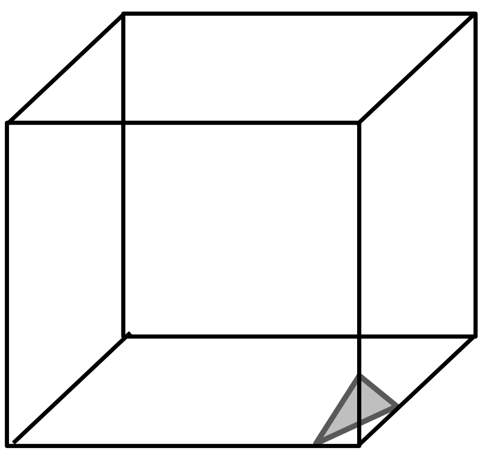
ETH zürich

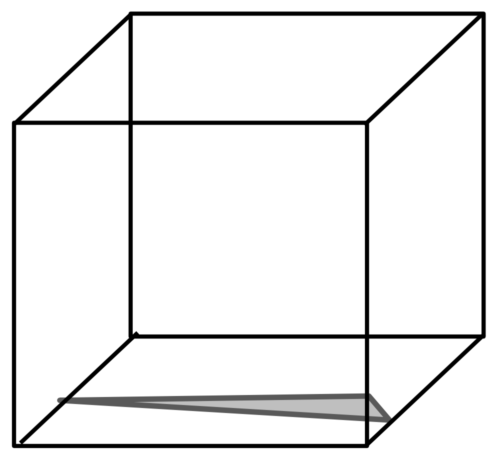
Marching Cubes – Problems

Problems with short triangle edges

- When the surface intersects the cube close to a corner, the resulting tiny triangle doesn‘t contribute much area to the mesh
- When the intersection is close to an edge of the cube, we get skinny triangles (bad aspect ratio)

Triangles with short edges waste resources but don‘t contribute to the surface mesh representation



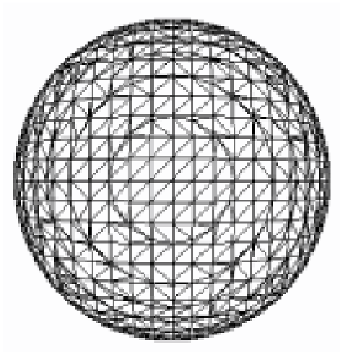


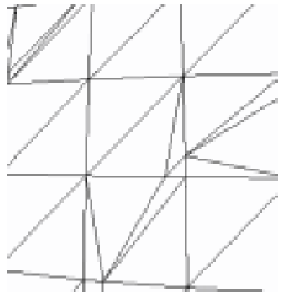
Grid Snapping

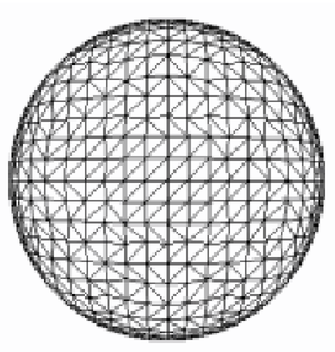
Solution: threshold the distances between the created vertices and the cube corners

When the distance is smaller than d_{snap} we snap the vertex to the cube corner

If more than one vertex of a triangle is snapped to the same point, we discard that triangle altogether







Grid Snapping

With grid snapping one can obtain significant reduction of space consumption

d_{snap}	0	0,1	0,2	0,3	0,4	0,46	0,495
Vertices	1446	1398	1254	1182	1074	830	830
Reduction (%)	0	3,3	13,3	18,3	25,7	42,6	42,6

Global RBF vs. Local MLS

RBF:

- sees the whole data set, can make for very smooth surfaces
- global (dense) system to solve – expensive

MLS:

- sees only a small part of the dataset, can get confused by noise
- local linear solves – cheap

Poisson Surface Reconstruction

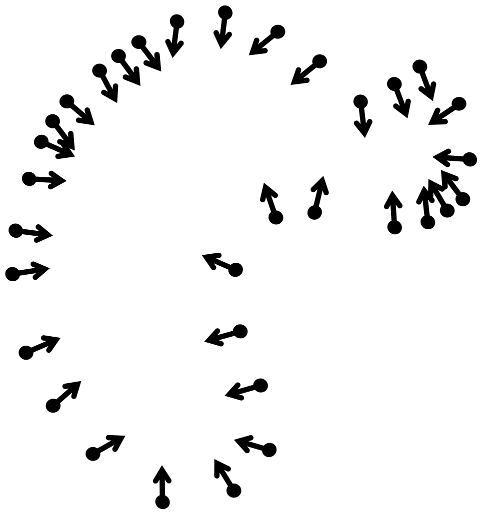
Very popular modern method, code available: M. Kazhdan, M. Bolitho and H. Hoppe, Symposium on Geometry Processing 2006
http://www.cs.jhu.edu/~misha/Code/PoissonRecon/

Global fitting of an *indicator function* using PDE

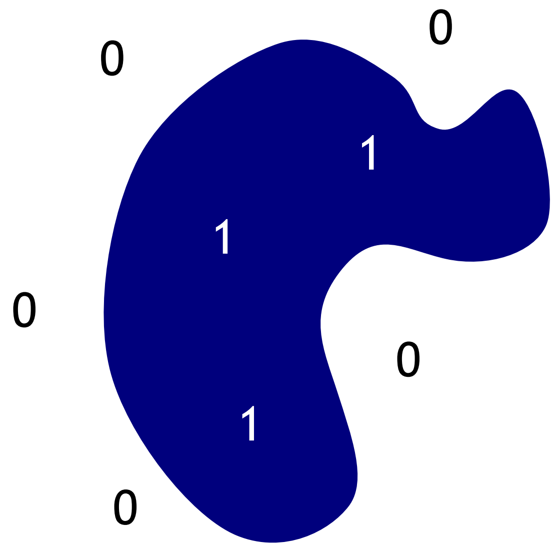
- Robust to noise, sparse, computationally tractable

You will try out the code in Ex2 and compare with MLS results

Poisson Surface Reconstruction



Oriented points

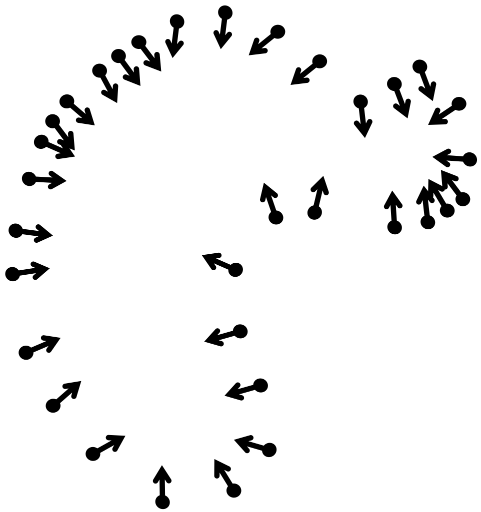


Indicator function

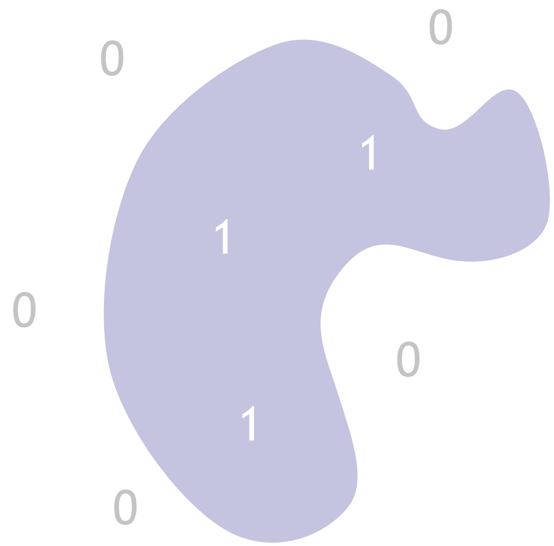
$\chi_{\mathcal{M}}$

ETH zürich

Poisson Surface Reconstruction



Oriented points

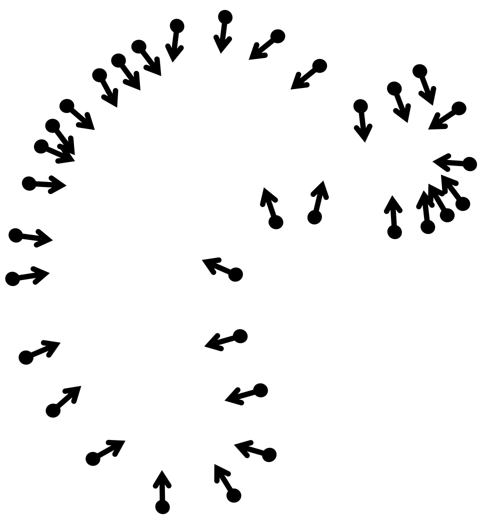


Indicator function

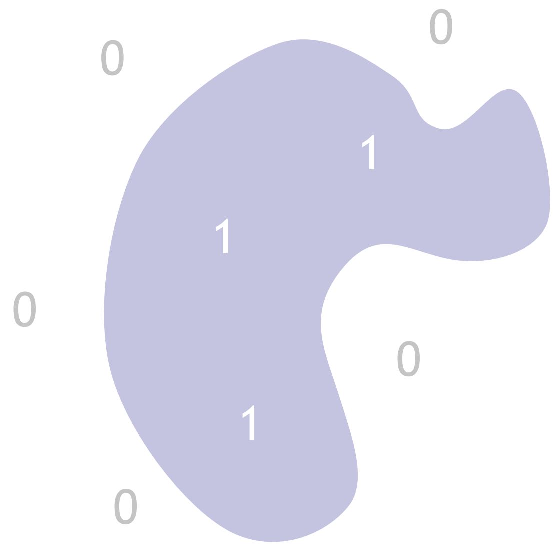
$\chi_{\mathcal{M}}$

We don’t know the indicator function ☹

Poisson Surface Reconstruction

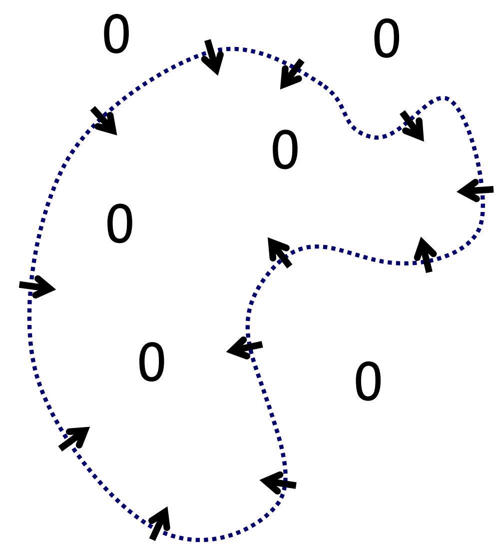


Oriented points



Indicator function

$\chi_{\mathcal{M}}$

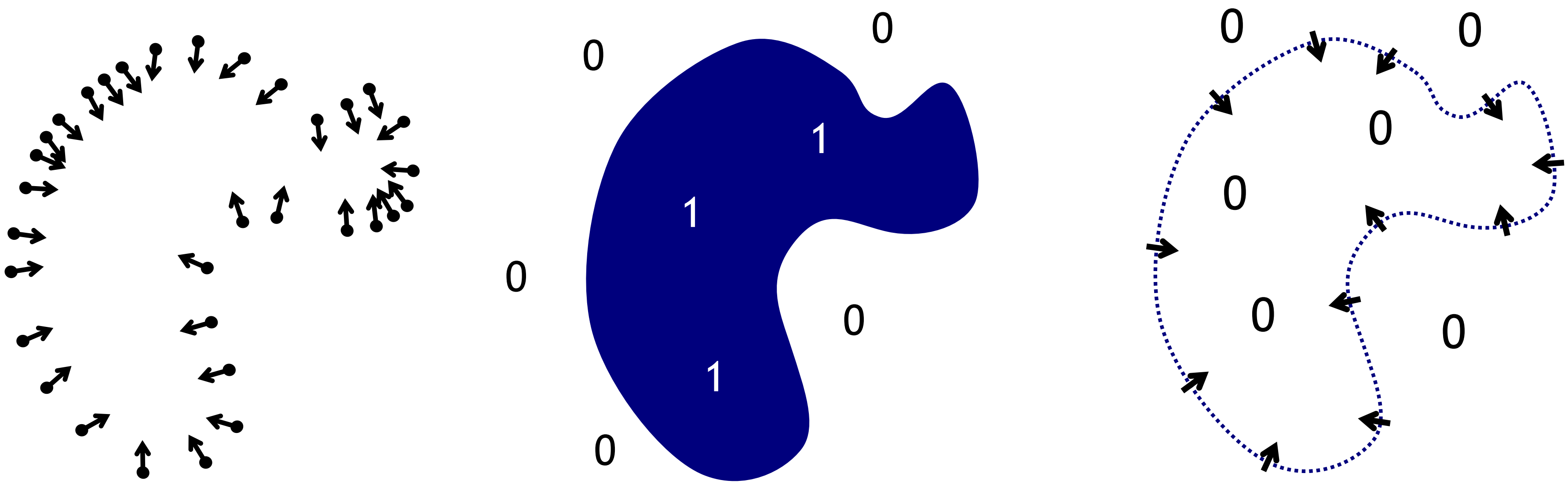


Indicator gradient

$\nabla \chi_{\mathcal{M}}$

But we can estimate its gradient! ☺

Poisson Surface Reconstruction



Oriented points

Indicator function $\chi_{\mathcal{M}}$

Indicator gradient $\nabla\chi_{\mathcal{M}}$

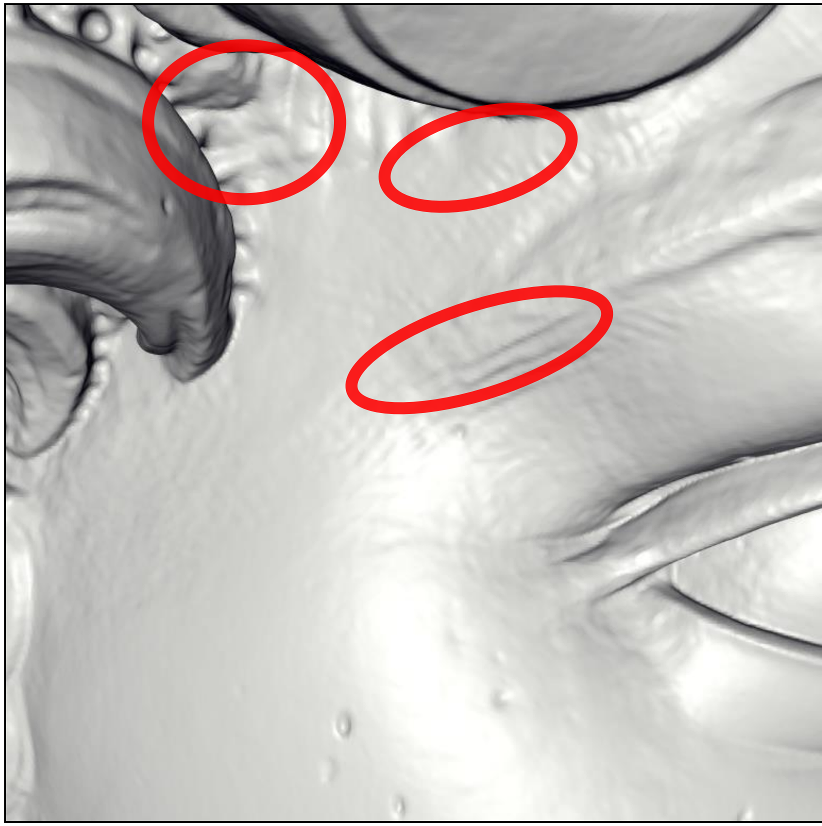
Reconstruct χ by solving the Poisson equation

$$\Delta\chi_{\mathcal{M}} = \mathrm{div}\nabla\chi_{\mathcal{M}}$$

Michelangelo's David

- 215 million data points from 1000 scans
- 22 million triangle reconstruction
- Compute time: 2.1 hours (this was in year 2006)
- Peak Memory: 6600MB

David – Chisel marks



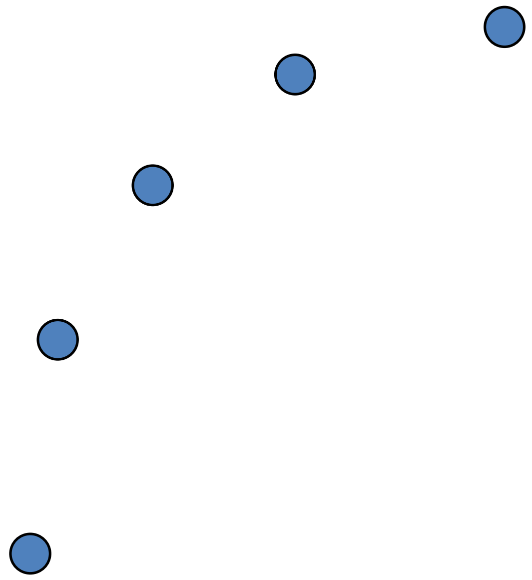
David – Drill Marks

David – Eye

Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

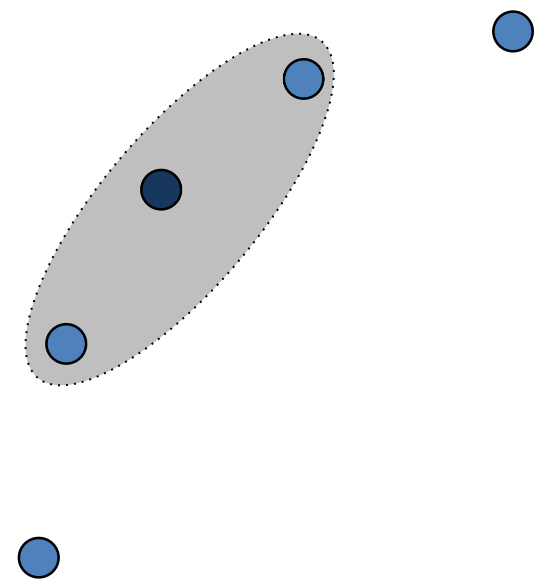
- Estimate the direction by fitting a local plane



Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

- Estimate the direction by fitting a local plane

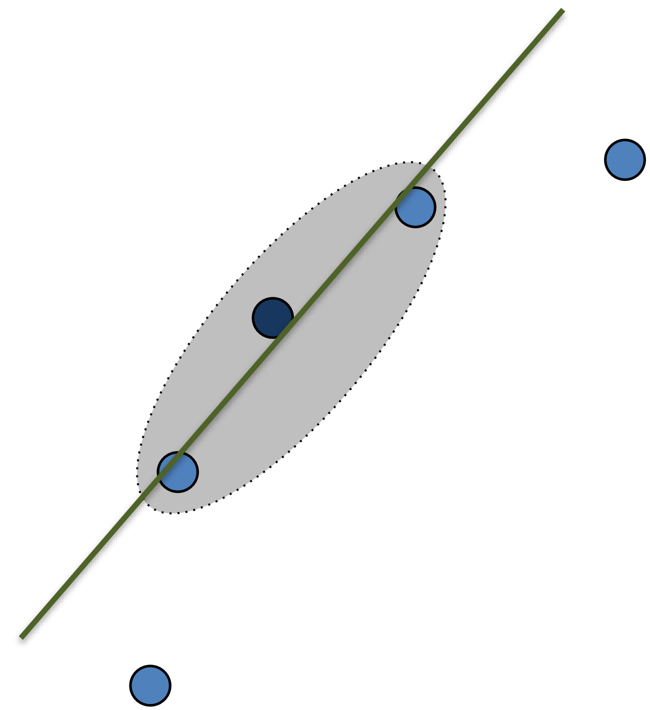


ETH zürich

Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

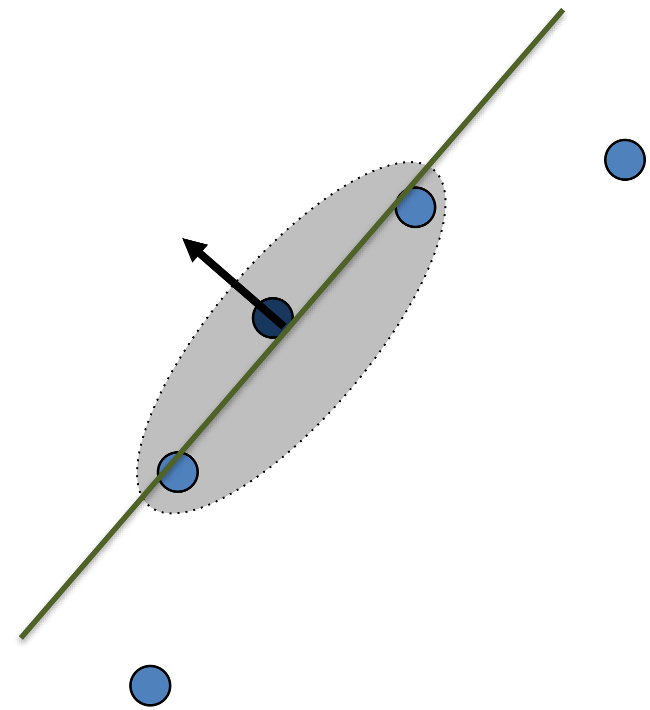
- Estimate the direction by fitting a local plane



Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

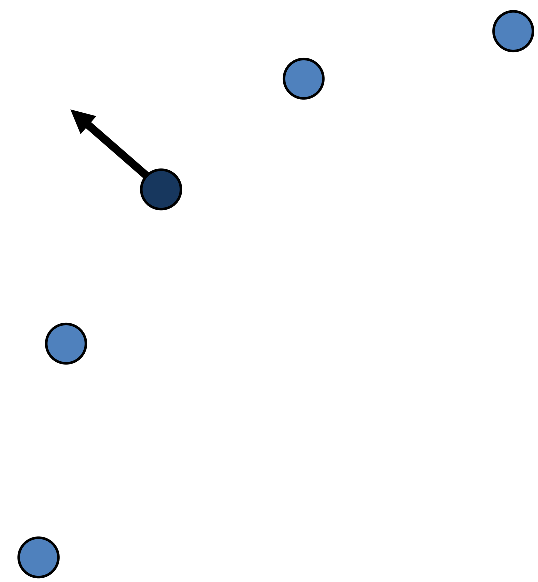
- Estimate the direction by fitting a local plane



Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

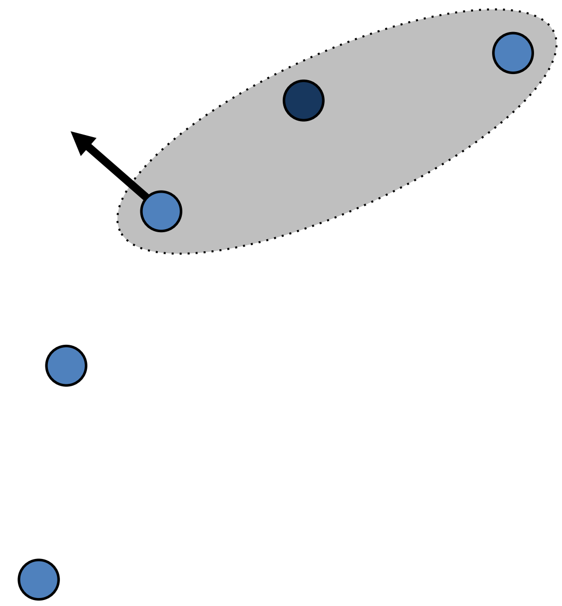
- Estimate the direction by fitting a local plane



Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

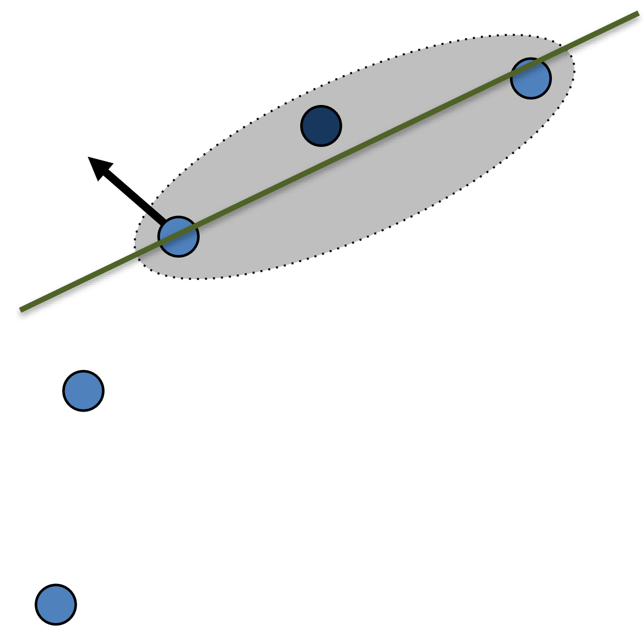
- Estimate the direction by fitting a local plane



Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

- Estimate the direction by fitting a local plane

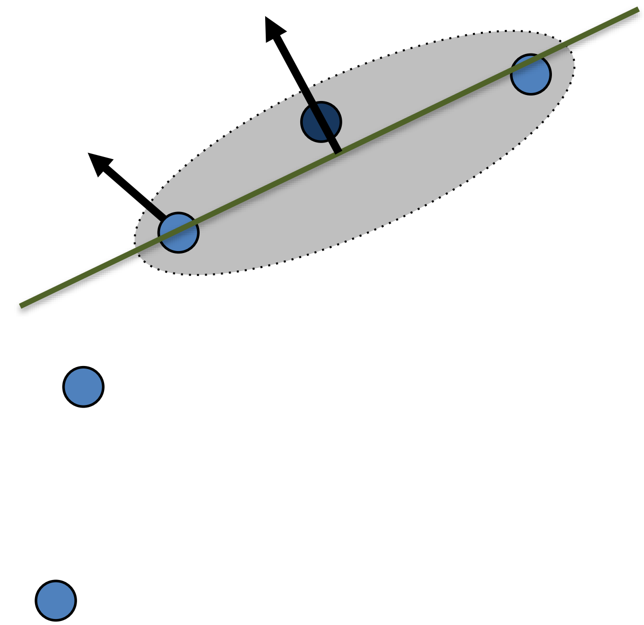


ETH zürich

Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

- Estimate the direction by fitting a local plane

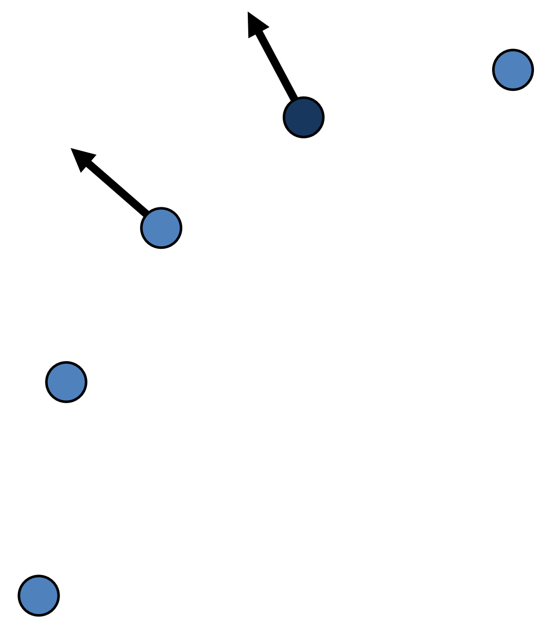


ETH zürich

Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

- Estimate the direction by fitting a local plane

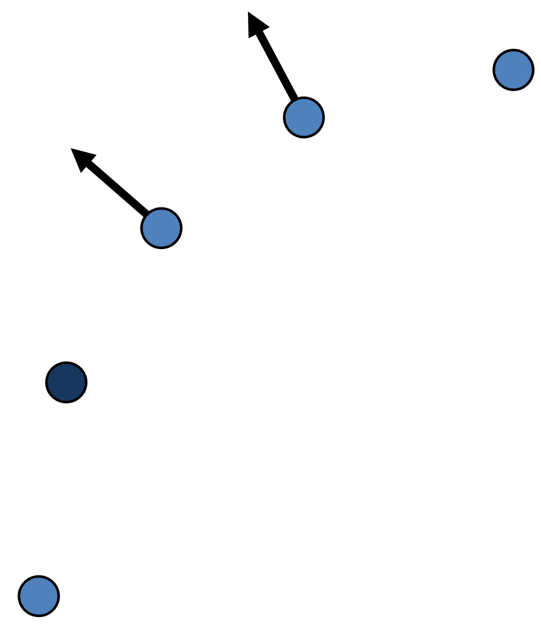


ETH zürich

Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

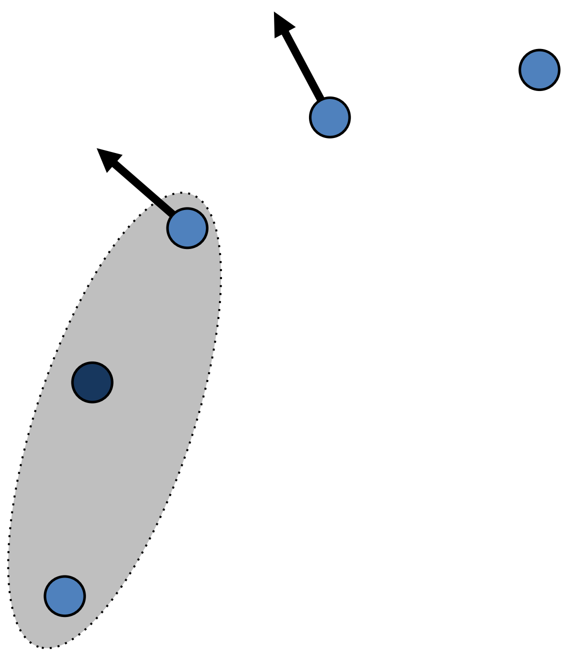
- Estimate the direction by fitting a local plane



Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

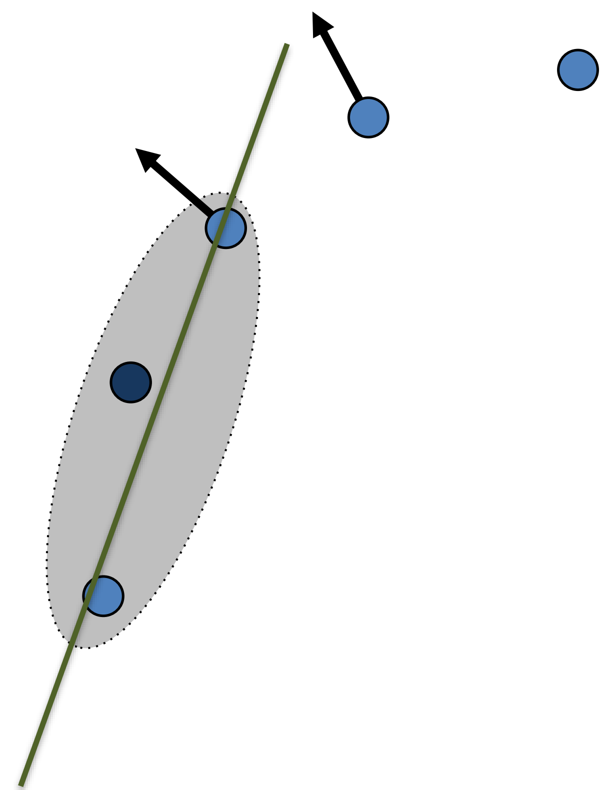
- Estimate the direction by fitting a local plane



Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

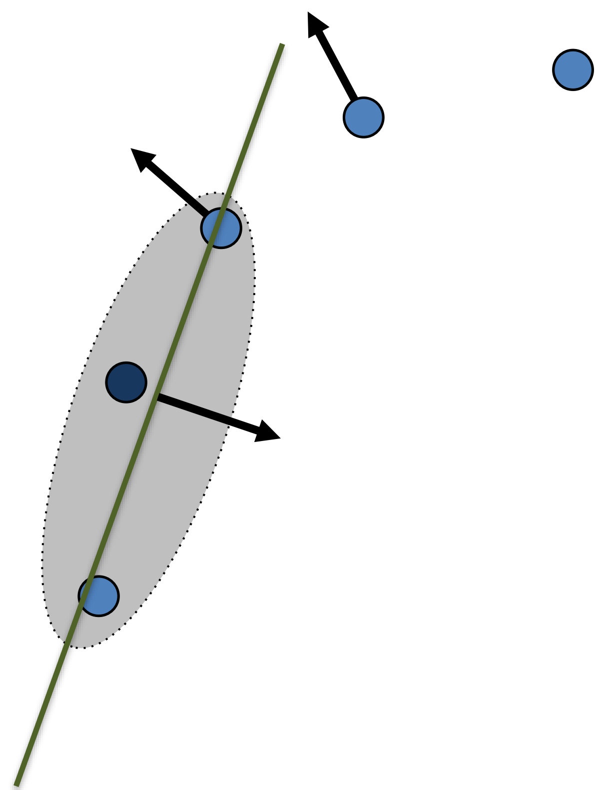
- Estimate the direction by fitting a local plane



Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

- Estimate the direction by fitting a local plane



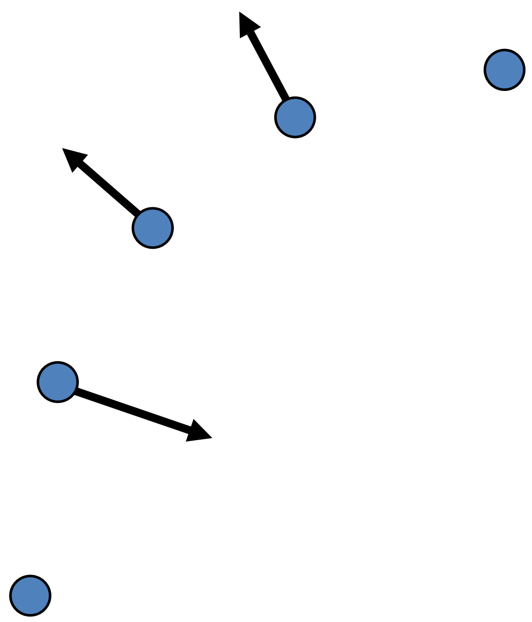
ETH zürich

Normal Estimation

Assign a normal vector **n** at each point cloud point **x**

- Estimate the direction by fitting a local plane

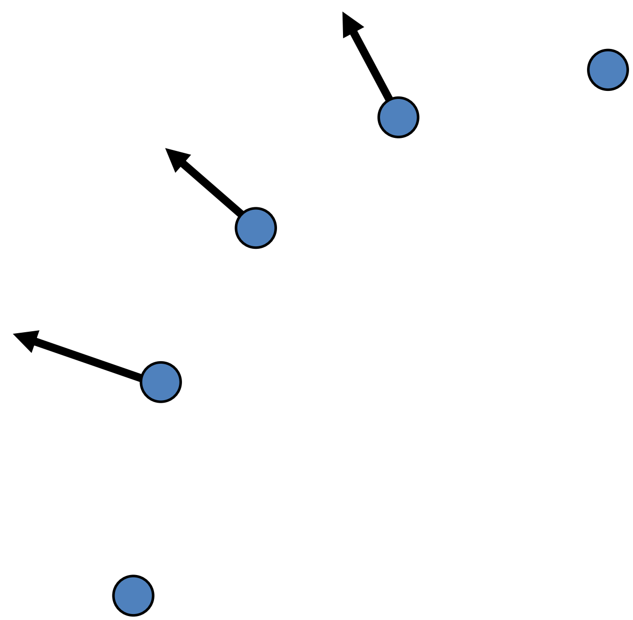
- Find consistent global orientation by propagation (spanning tree)



Normal Estimation

Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$

- Estimate the direction by fitting a local plane
- Find consistent global orientation by propagation (spanning tree)



ETH zürich

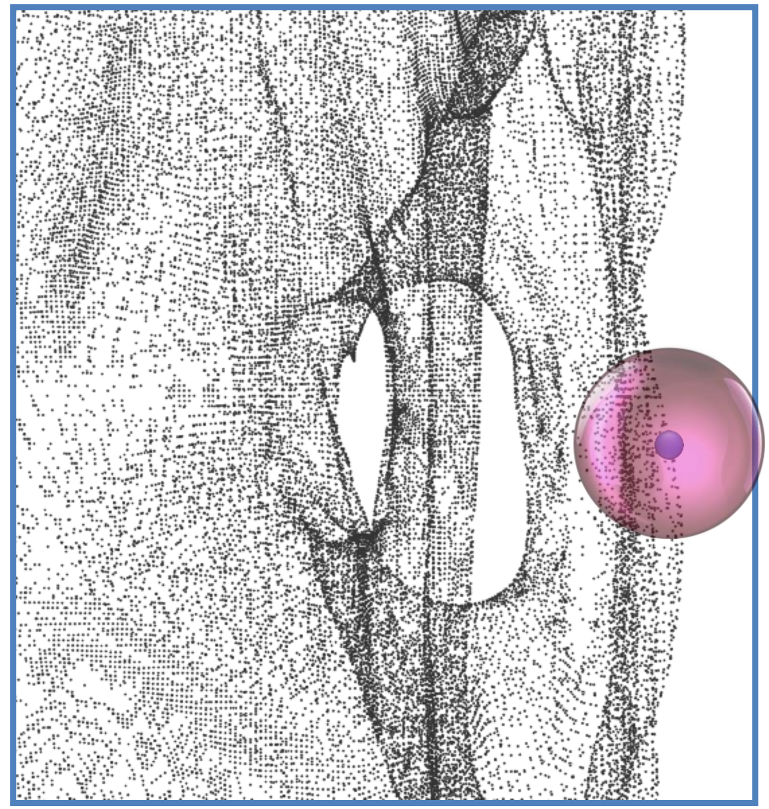
Local Plane Fitting

- For each point $\mathbf{x}$ in the cloud, pick k nearest neighbors or all points in r-ball: $\{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{x}\| < r\}$

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$$

- Find a plane Π that minimizes the sum of square distances:

$$\min \sum_{i=1}^{n} \operatorname{dist}(\mathbf{x}_i, \Pi)^2$$



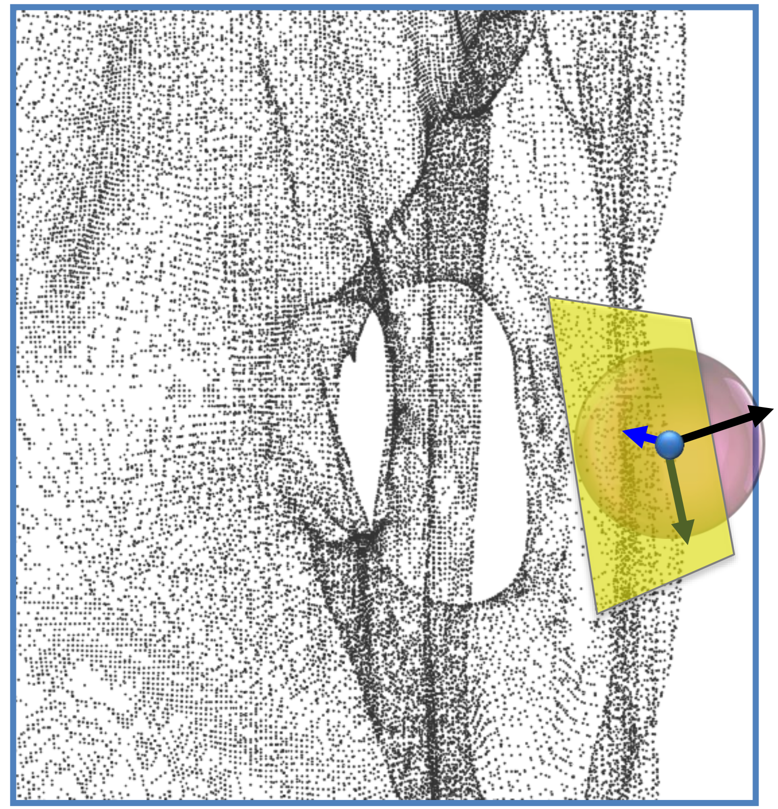
Local Plane Fitting

- For each point $\mathbf{x}$ in the cloud, pick k nearest neighbors or all points in r-ball: $\{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{x}\| < r\}$

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$$

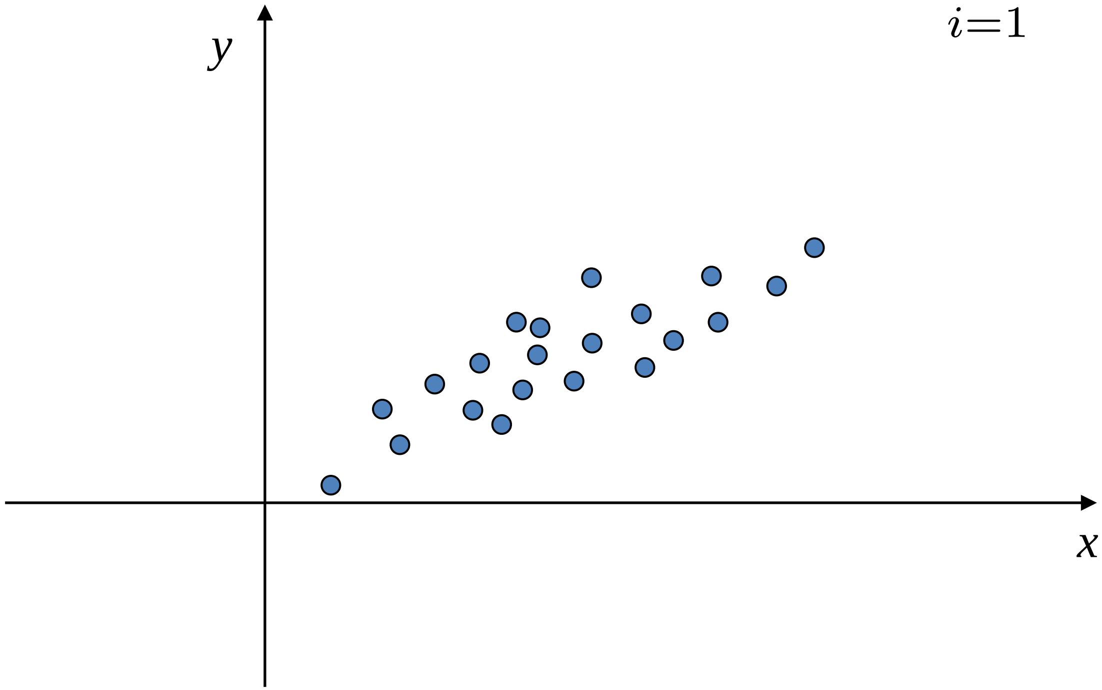
- Find a plane Π that minimizes the sum of square distances:

$$\min \sum_{i=1}^{n} \operatorname{dist}(\mathbf{x}_i, \Pi)^2$$



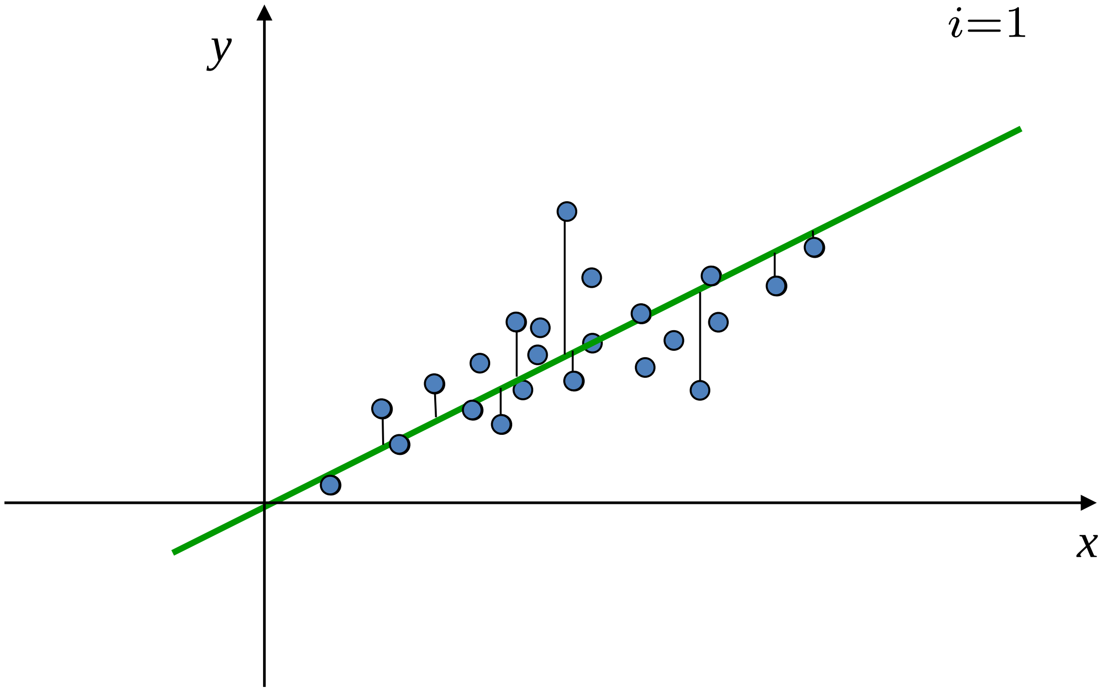
Linear Least Squares?

Find a line $y = ax + b$ s.t. $\min \sum_{i=1}^{n} (y_i - (ax_i + b))^2$



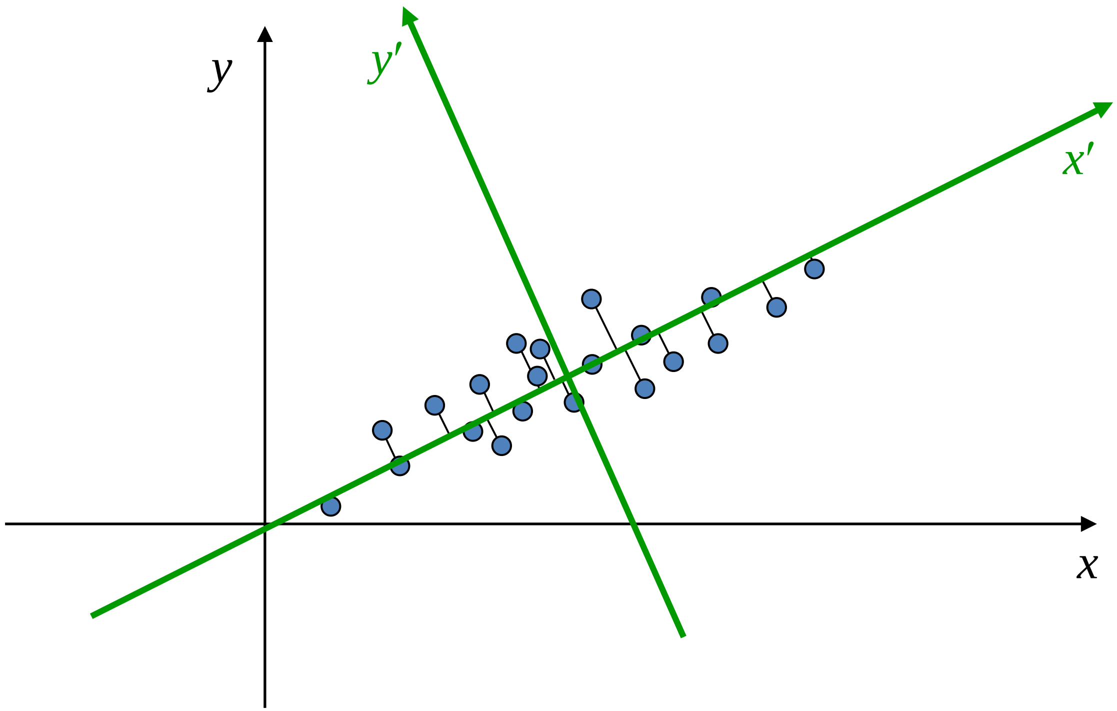
Linear Least Squares?

Find a line $y = ax + b$ s.t. $\min \sum_{i=1}^{n} (y_i - (ax_i + b))^2$



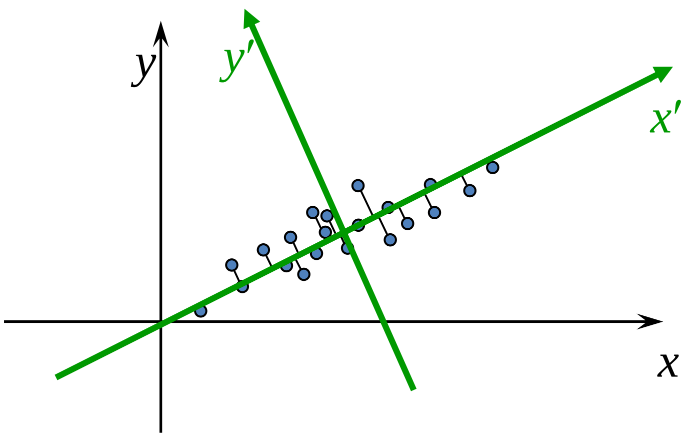
But we would like true orthogonal distances!

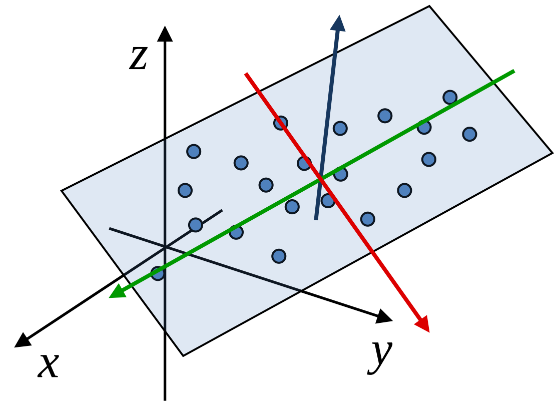
Principle Component Analysis (PCA)



Principle Component Analysis (PCA)

PCA finds an orthogonal basis that best represents a given data set





PCA finds the best approximating line/plane/orientation… (in terms of $\Sigma_{distances}{}^2$)

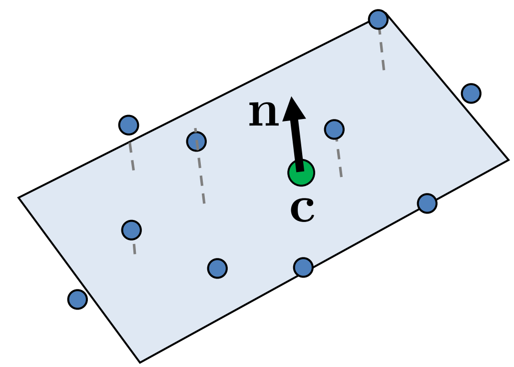
Notations

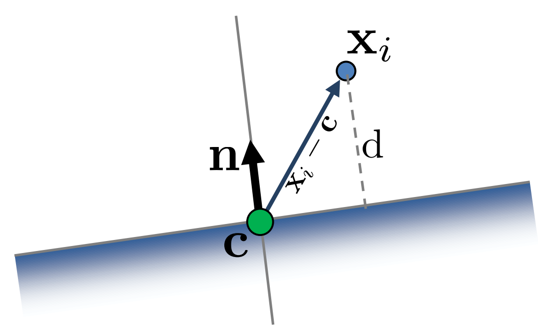
Input points: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

Looking for a (hyper) plane passing through $\mathbf{c}$ with normal $\mathbf{n}$ s.t.

$$\min_{\mathbf{c},\mathbf{n},\|\mathbf{n}\|=1} \sum_{i=1}^{n} \left((\mathbf{x}_i - \mathbf{c})^T \mathbf{n}\right)^2$$

$\mathbf{c}$ and $\mathbf{n}$ are variables



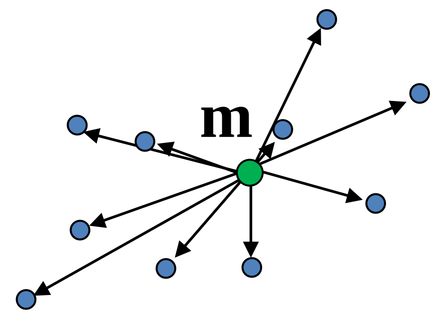


Notations

Input points: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

Centroid:

$$\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$



Vectors from the centroid:

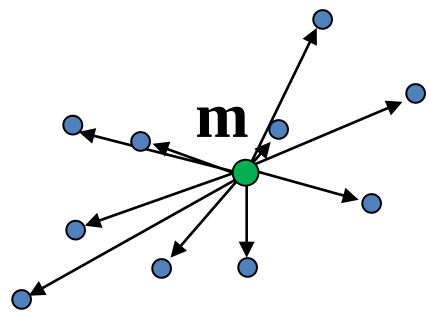
$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}$$

Centroid: 0-dim Approximation

It can be shown that:

$$\mathbf{m} = \underset{\mathbf{c}}{\operatorname{argmin}} \sum_{i=1}^{n} \left((\mathbf{x}_i - \mathbf{c})^T \mathbf{n} \right)^2$$

$$\mathbf{m} = \underset{\mathbf{c}}{\operatorname{argmin}} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{c}\|^2$$



$$\mathbf{m} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$$

$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}$$

$\mathbf{m}$ will be the origin of the (hyper)-plane

Our problem becomes:

$$\min_{\|\mathbf{n}\|=1} \sum_{i=1}^{n} \left(\mathbf{y}_i^T \mathbf{n} \right)^2$$

ETH zürich

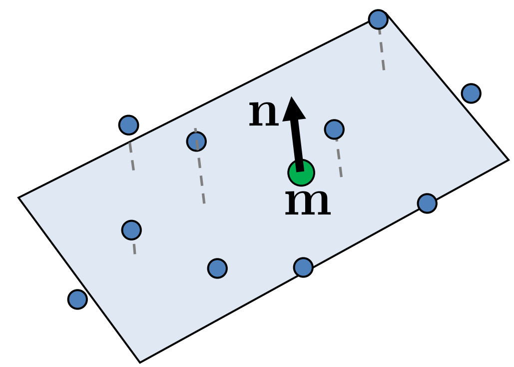
Hyperplane Normal

Minimize!

$$\min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \left(\mathbf{y}_i^T \mathbf{n}\right)^2 = \min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \mathbf{n}^T \mathbf{y}_i \mathbf{y}_i^T \mathbf{n} =$$

$$\min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^T\right) \mathbf{n} = \min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\mathbf{Y}\mathbf{Y}^T\right) \mathbf{n}$$

$$\mathbf{Y} = \begin{pmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_n \\ | & | & & | \end{pmatrix}$$

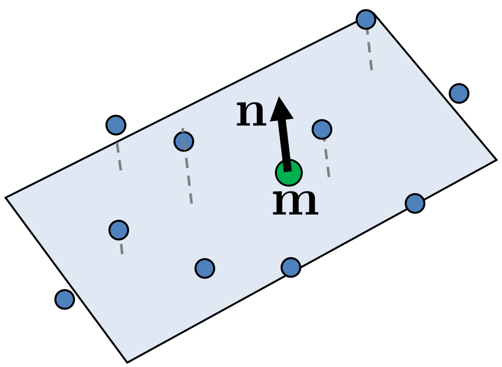


Hyperplane Normal

Minimize!

$$\min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \left(\mathbf{y}_i^T \mathbf{n}\right)^2 = \min_{\mathbf{n}^T\mathbf{n}=1} \sum_{i=1}^{n} \mathbf{n}^T \mathbf{y}_i \mathbf{y}_i^T \mathbf{n} =$$

$$\min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^T \right) \mathbf{n} = \min_{\mathbf{n}^T\mathbf{n}=1} \mathbf{n}^T \left(\mathbf{Y}\mathbf{Y}^T\right) \mathbf{n}$$



$$\mathbf{Y} = \begin{pmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_n \\ | & | & & | \end{pmatrix}$$

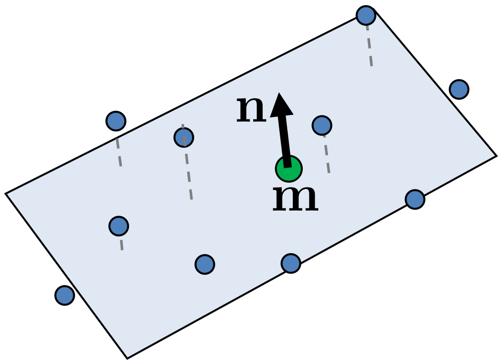
$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$

$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T\mathbf{n} = 1$$

Hyperplane Normal

Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T\mathbf{S}\mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$
$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T\mathbf{n} = 1$$



$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T\mathbf{n} - 1)$$
$$\nabla\mathcal{L} = 0$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{n}} = \frac{\partial}{\partial\mathbf{n}} f(\mathbf{n}) - \lambda\frac{\partial}{\partial\mathbf{n}}(\mathbf{n}^T\mathbf{n} - 1)$$
$$\frac{\partial\mathcal{L}}{\partial\lambda} = \mathbf{n}^T\mathbf{n} - 1$$

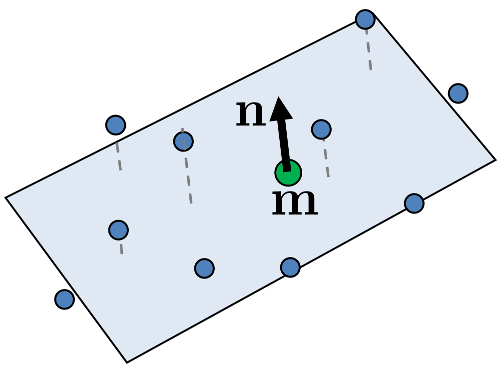
$$\frac{\partial}{\partial\mathbf{n}} f(\mathbf{n}) - \lambda\frac{\partial}{\partial\mathbf{n}}(\mathbf{n}^T\mathbf{n} - 1) = (\mathbf{S} + \mathbf{S}^T)\mathbf{n} - \lambda(\mathbf{I} + \mathbf{I}^T)\mathbf{n} = 2\mathbf{S}\mathbf{n} - 2\lambda\mathbf{n}$$

Hyperplane Normal

Constrained minimization - Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$

$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T \mathbf{n} = 1$$



$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T \mathbf{n} - 1)$$

$$\nabla \mathcal{L} = 0$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}} = 0 \iff \mathbf{S}\mathbf{n} = \lambda \mathbf{n}$$

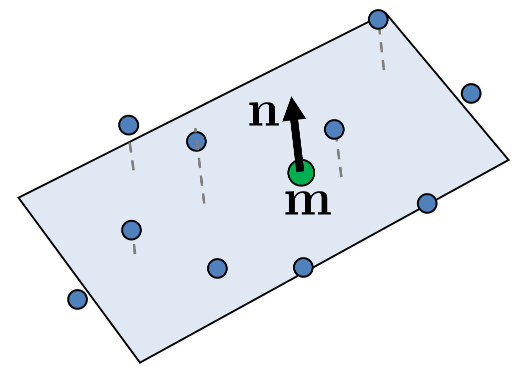
$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff \mathbf{n}^T \mathbf{n} = 1$$

Hyperplane Normal

Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T\mathbf{S}\mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$
$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T\mathbf{n} = 1$$

$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T\mathbf{n} - 1)$$
$$\nabla\mathcal{L} = 0$$



$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}} = 0 \iff \mathbf{S}\mathbf{n} = \lambda\mathbf{n}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff \mathbf{n}^T\mathbf{n} = 1$$

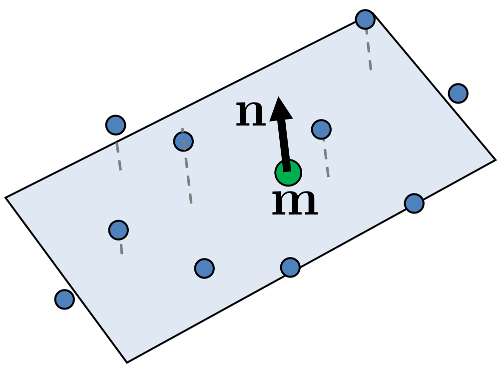
What can be said about $\mathbf{n}$??

ETH zürich

Hyperplane Normal

Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \qquad (\mathbf{S} = \mathbf{Y}\mathbf{Y}^T)$$
$$\min f(\mathbf{n}) \quad s.t.\ \mathbf{n}^T \mathbf{n} = 1$$



$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T \mathbf{n} - 1)$$
$$\nabla \mathcal{L} = 0$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}} = 0 \iff \mathbf{S}\mathbf{n} = \lambda \mathbf{n}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff \mathbf{n}^T \mathbf{n} = 1$$

$\mathbf{n}$ is the eigenvector of $\mathbf{S}$ with the smallest eigenvalue

Summary – Best Fitting Plane Recipe

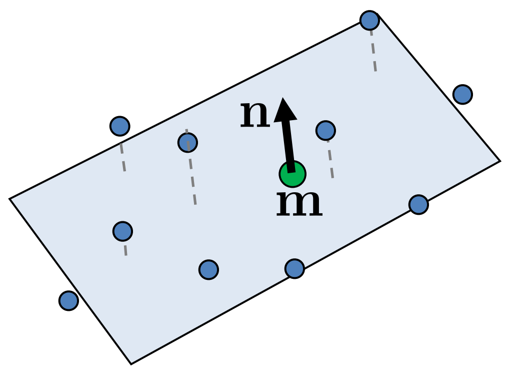
- Input: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$
- Compute centroid = plane origin $\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$
- Compute scatter matrix $\mathbf{S} = \mathbf{Y}\mathbf{Y}^T$

$$\mathbf{Y} = (\mathbf{y}_1 \ \mathbf{y}_2 \ \ldots \ \mathbf{y}_n)$$

$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}$$

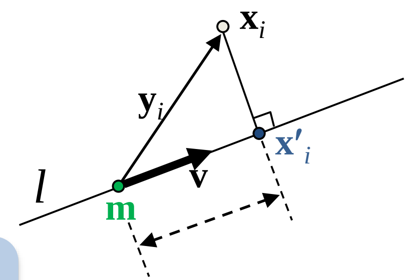
- The plane normal $\mathbf{n}$ is the eigenvector of $\mathbf{S}$ with the smallest eigenvalue

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix} \mathbf{V}^T$$



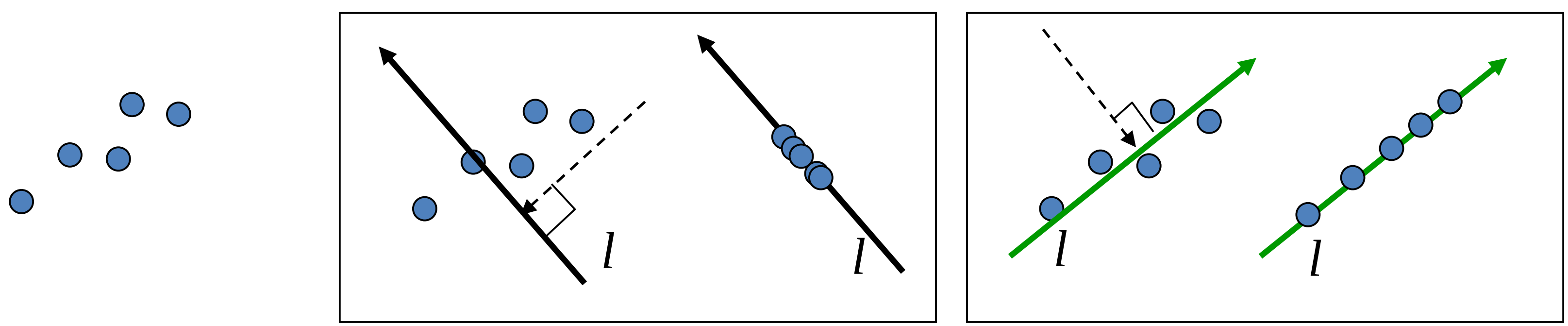
What does Scatter Matrix do?

- Let's look at a line l through the center of mass $\mathbf{m}$ with direction vector $\mathbf{v}$, and project our points $\mathbf{x}_i$ onto it. The variance of the projected points $\mathbf{x}'_i$ is:



$$\mathrm{var}(\mathbf{x}_1, \ldots \mathbf{x}_n; \mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}'_i - \mathbf{m}\|^2 =$$

$$= \frac{1}{n}\sum_{i=1}^{n} \|(\mathbf{m} + \mathbf{v}^T\mathbf{y}_i) - \mathbf{m}\|^2 = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{y}_i^T\mathbf{v})^2 = \frac{1}{n}\mathbf{v}^T\mathbf{S}\mathbf{v}$$



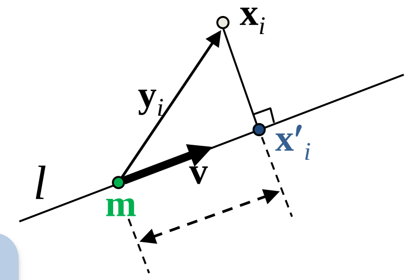
Original set | Small variance | Large variance

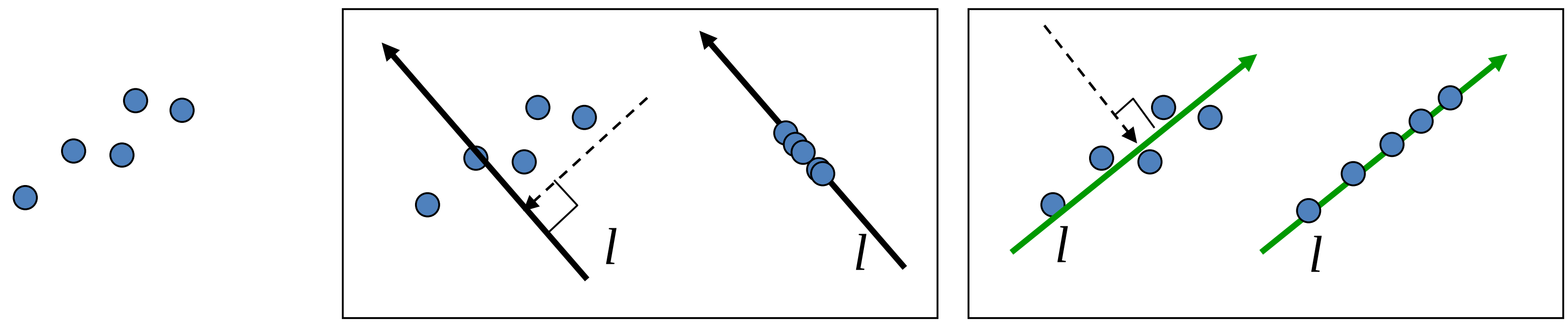
What does Scatter Matrix do?

- The scatter matrix measures the variance of our data points along the direction $\mathbf{v}$

$$\mathrm{var}(\mathbf{x}_1, \ldots \mathbf{x}_n; \mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}'_i - \mathbf{m}\|^2 =$$

$$= \frac{1}{n}\sum_{i=1}^{n} \|(\mathbf{m} + \mathbf{v}^T\mathbf{y}_i) - \mathbf{m}\|^2 = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{y}_i^T\mathbf{v})^2 = \frac{1}{n}\mathbf{v}^T\mathbf{S}\mathbf{v}$$





Original set | Small variance | Large variance

ETH zürich

Principal Components

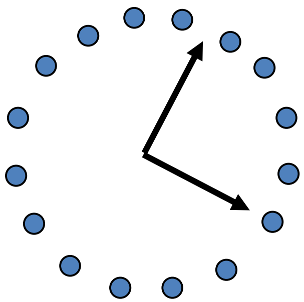
The scatter matrix measures the variance of the data points along the direction $\mathbf{v}$

Eigenvectors of $\mathbf{S}$ that correspond to big eigenvalues are the directions in which the data has strong components (= large variance).

If the eigenvalues are more or less the same, there is no preferable direction.

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix} \mathbf{V}^T$$

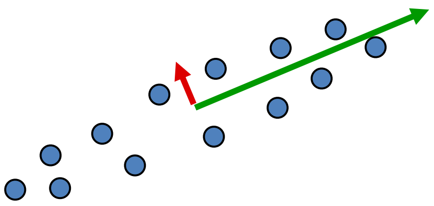
Principal Components



- There's no preferable direction
- S looks like this:

$$\mathbf{S} = \mathbf{V}\begin{pmatrix}\lambda & \\ & \lambda\end{pmatrix}\mathbf{V}^T$$

- Any vector is an eigenvector



- There's a clear preferable direction
- S looks like this:

$$\mathbf{S} = \mathbf{V}\begin{pmatrix}\lambda & \\ & \mu\end{pmatrix}\mathbf{V}^T$$

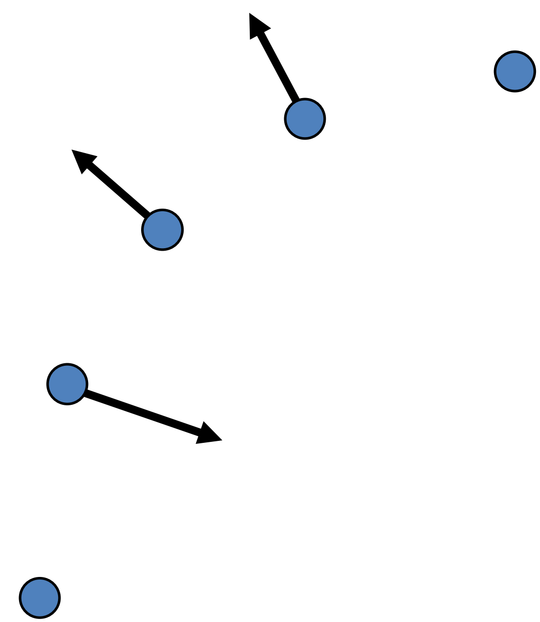
- μ is close to zero, much smaller than λ

Normal Orientation

PCA may return arbitrarily oriented eigenvectors

Need to orient consistently

Neighboring points should have similar normals



Normal Orientation

Build graph connecting neighboring points

- Edge (i,j) exists if $\mathbf{x}_i \in \mathrm{kNN}(\mathbf{x}_j)$ or $\mathbf{x}_j \in \mathrm{kNN}(\mathbf{x}_i)$

Propagate normal orientation through graph

- For neighbors $\mathbf{x}_i$, $\mathbf{x}_j$: Flip $\mathbf{n}_j$ if $\mathbf{n}_i^T \mathbf{n}_j < 0$

"Surface reconstruction from unorganized points", Hoppe et al., SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf

Normal Orientation

Build graph connecting neighboring points

- Edge (i,j) exists if $\mathbf{x}_i \in \mathrm{kNN}(\mathbf{x}_j)$ or $\mathbf{x}_j \in \mathrm{kNN}(\mathbf{x}_i)$

Propagate normal orientation through graph

- For neighbors $\mathbf{x}_i, \mathbf{x}_j$: Flip $\mathbf{n}_j$ if $\mathbf{n}_i^{\mathrm{T}}\mathbf{n}_j < 0$
- Fails at sharp edges/corners

Propagate along "safe" paths (parallel tangent planes)

- Minimum spanning tree with angle-based edge weights $w_{ij} = 1 - |\mathbf{n}_i^T \mathbf{n}_j|$

http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf