

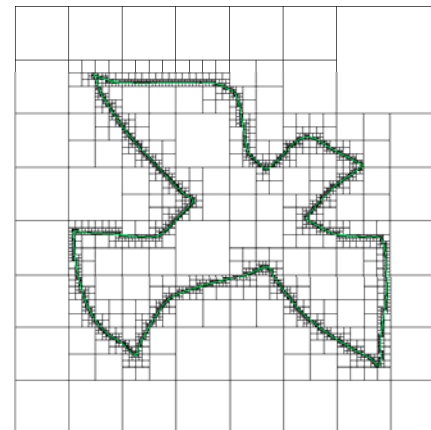
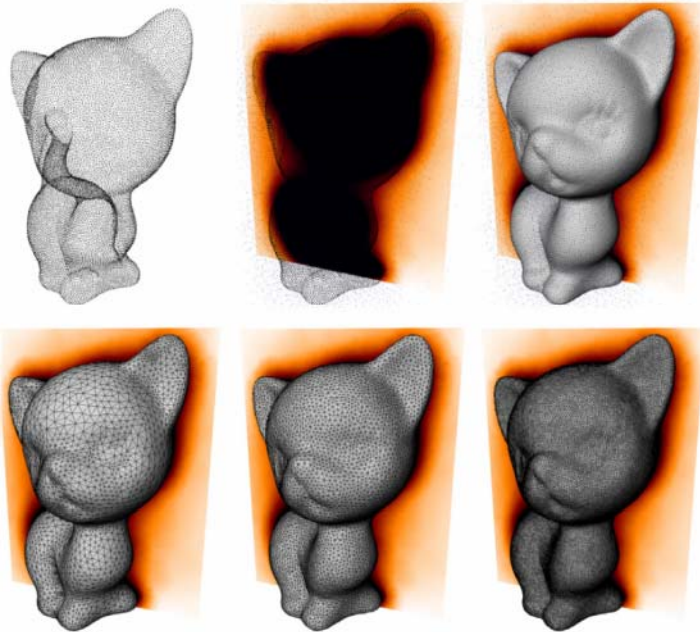
G22.3033-004, Spring 2009

Interactive Shape Modeling

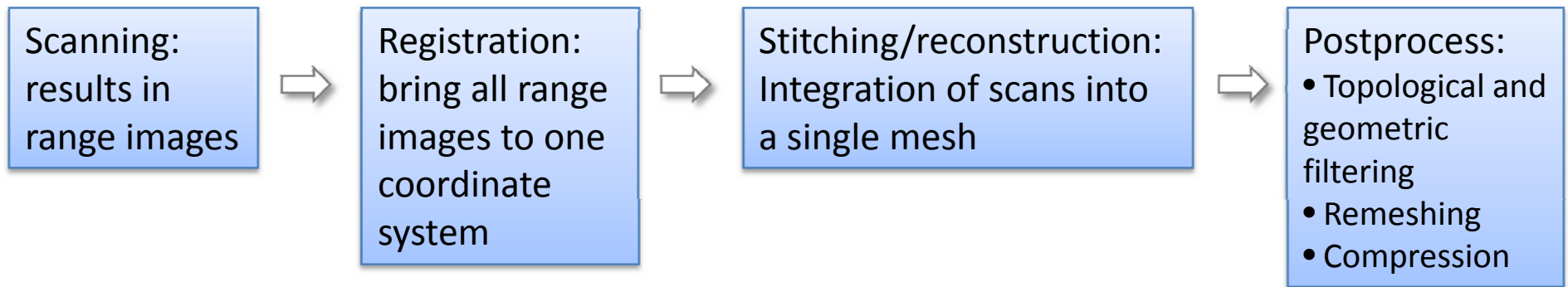
Shape Reconstruction

Course Topics

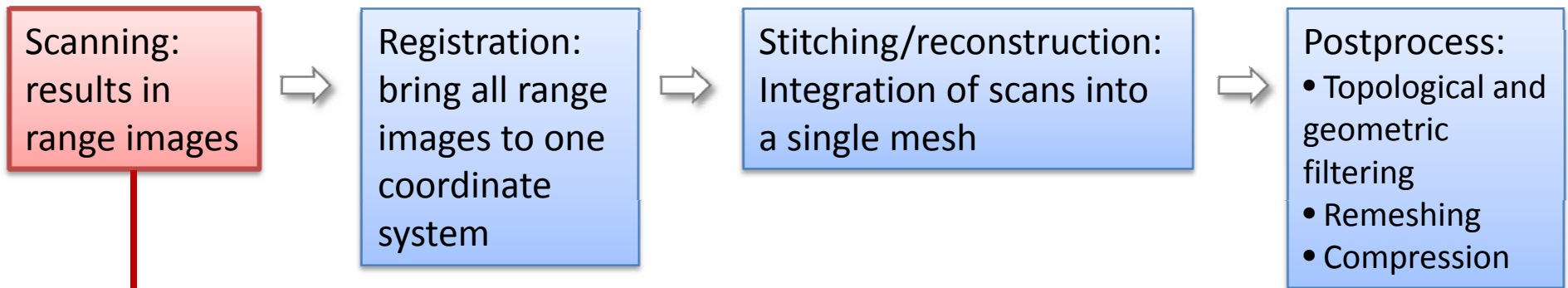
- Shape acquisition
 - Scanning/imaging
 - Reconstruction



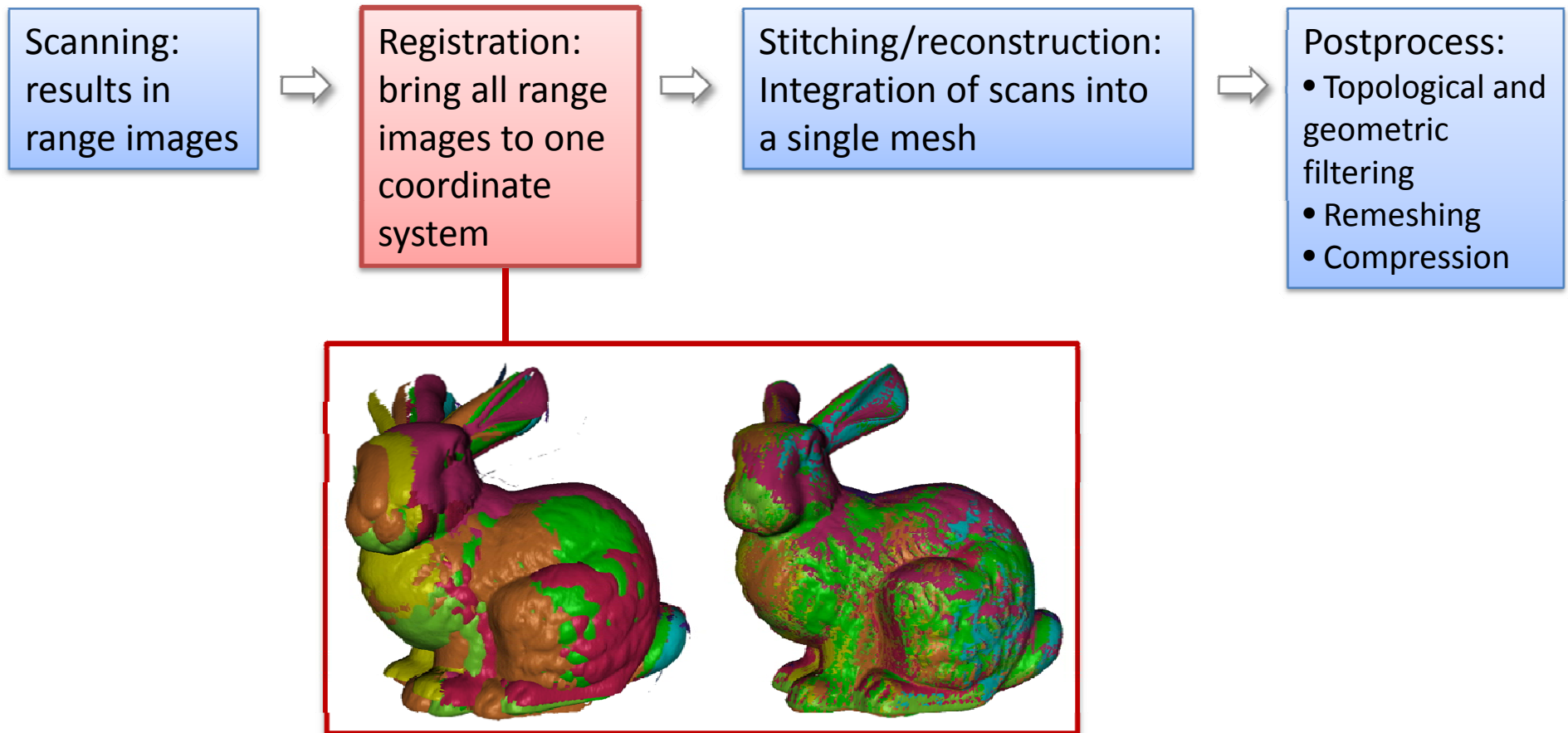
Data Acquisition Pipeline



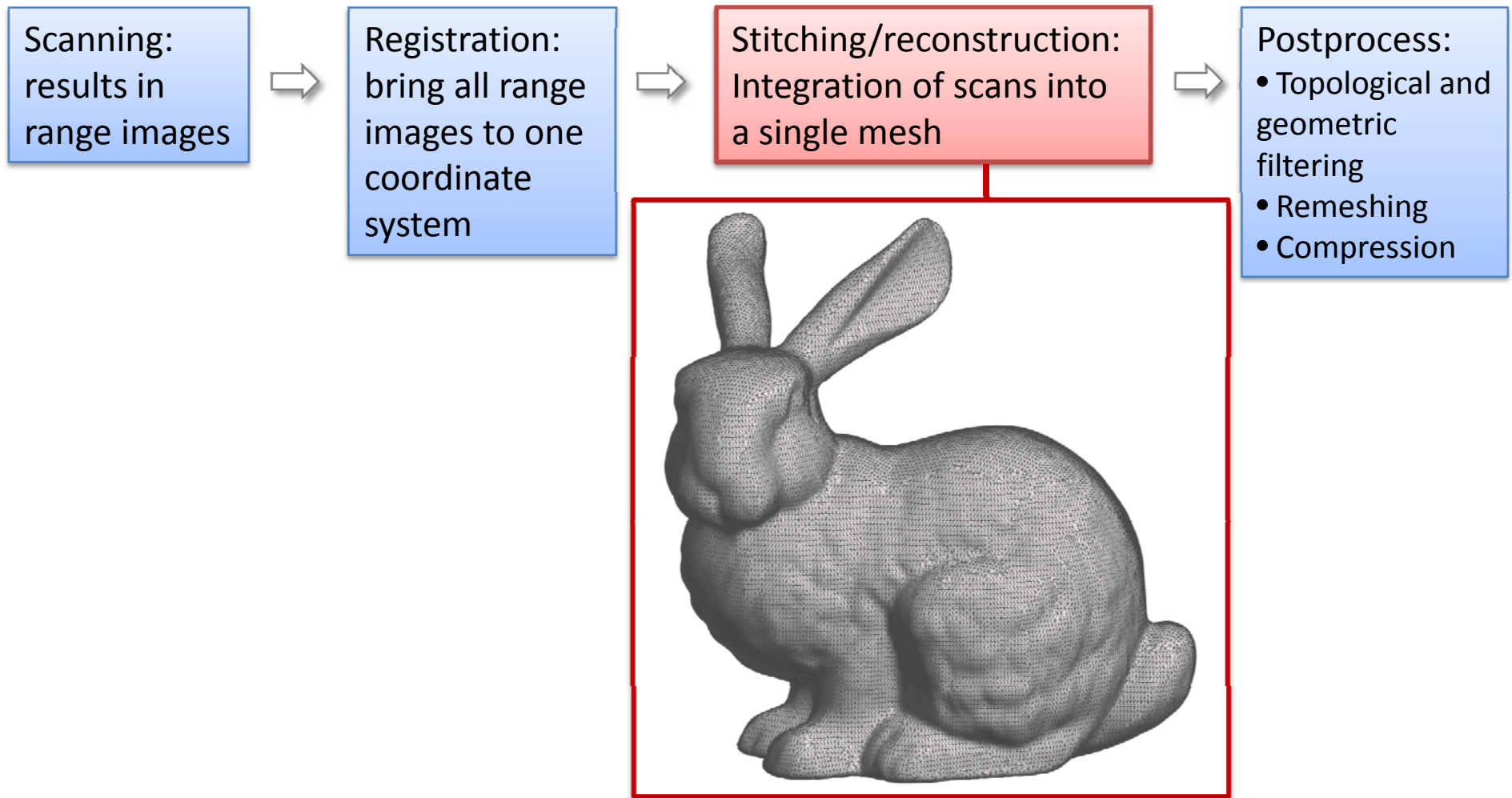
Data Acquisition Pipeline



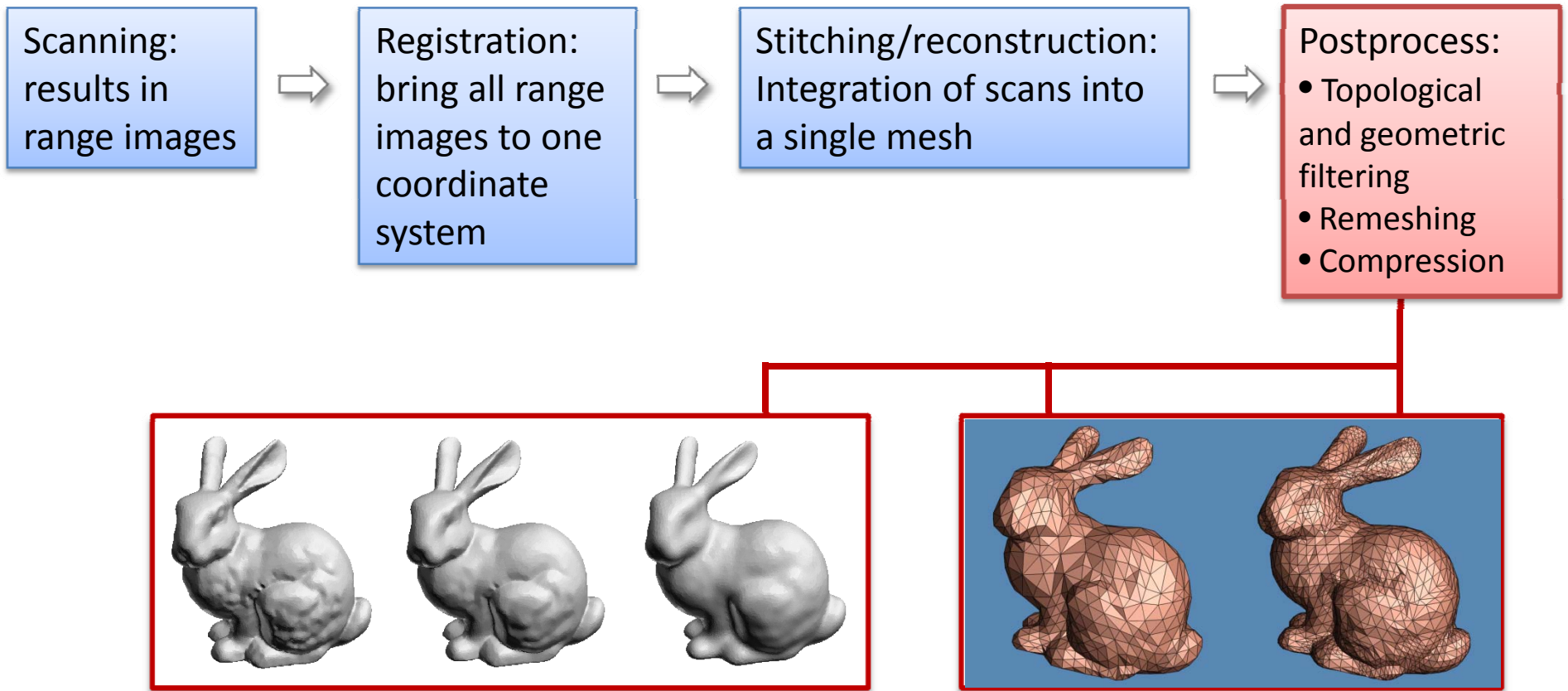
Data Acquisition Pipeline



Data Acquisition Pipeline



Data Acquisition Pipeline



Touch probes

- Physical contact with the object
- Manual or computer-guided
- Advantages:
 - Can be very precise
 - Can scan any solid surface
- Disadvantages:
 - Slow, small scale
 - Can't use on fragile objects



Optical scanning

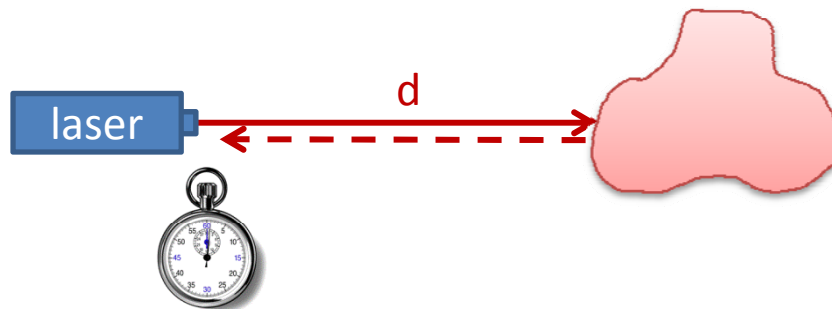
- Infer the geometry from light reflectance
- Advantages:
 - Less invasive than touch
 - Fast, large scale possible
- Disadvantages:
 - Difficulty with transparent and shiny objects



Optical scanning – active lighting

Time of flight laser

- Laser rangefinder (lidar)
- Measures the time it takes the laser beam to hit the object and come back
- Scans one point at a time; mirrors used to change beam direction

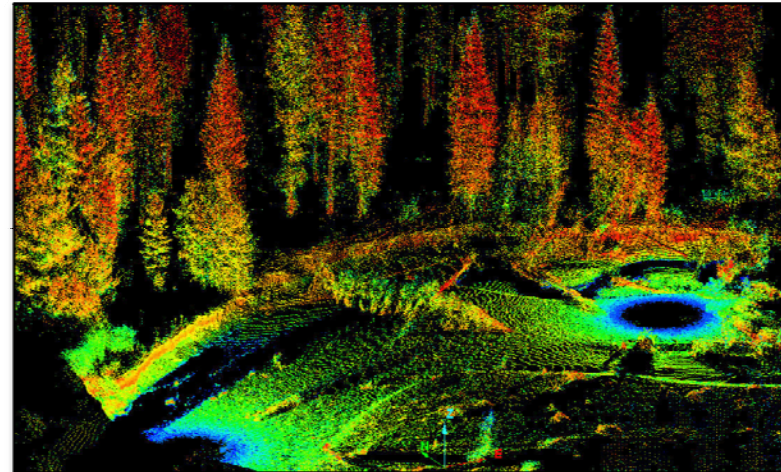
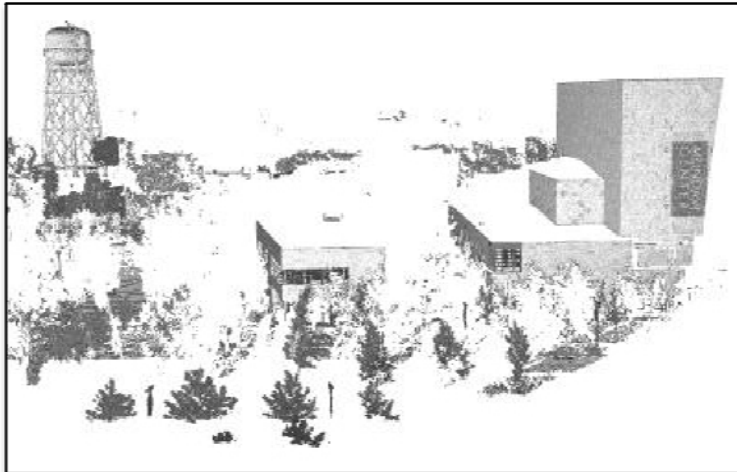


$$d = 0.5 t \cdot c$$

Optical scanning – active lighting

Time of flight laser

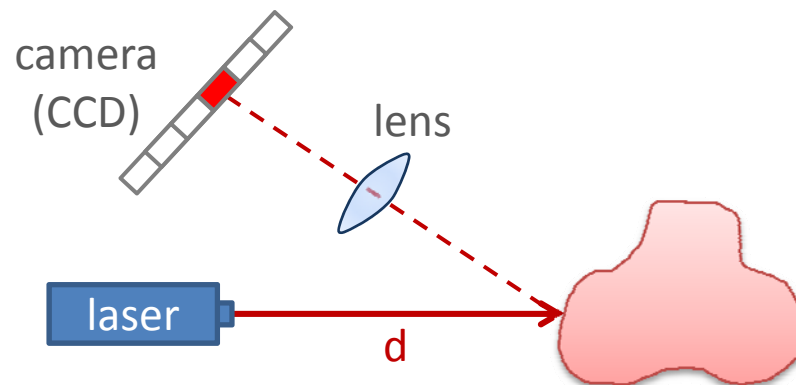
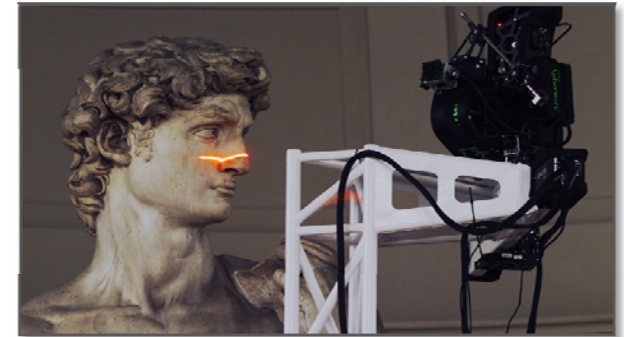
- Accommodates large range – up to several miles (suitable for buildings, rocks)
- Lower accuracy (light travels really fast)



Optical scanning – active lighting

Triangulation laser

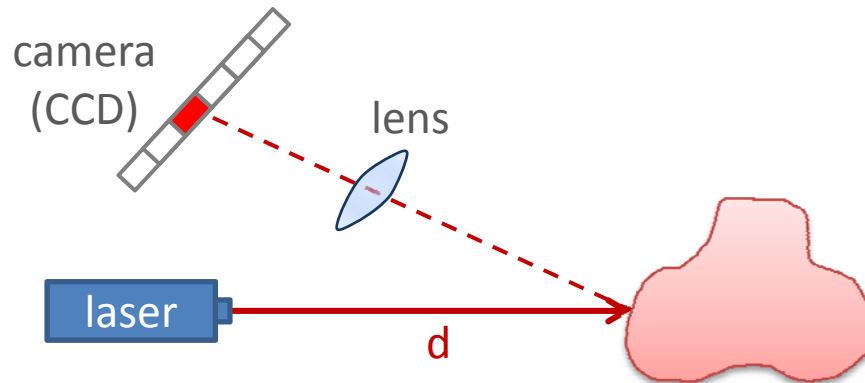
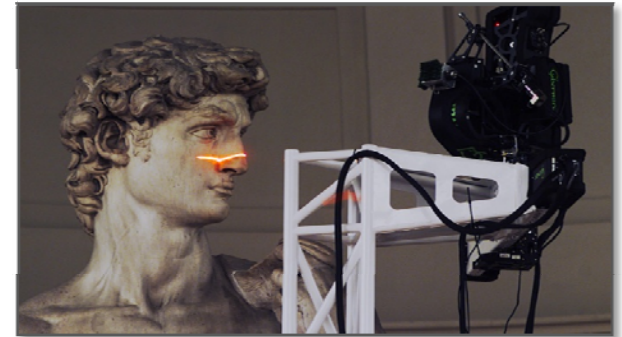
- Laser beam and camera
- Laser dot is photographed
- The location of the dot in the image allows triangulation – so we get the distance to the object



Optical scanning – active lighting

Triangulation laser

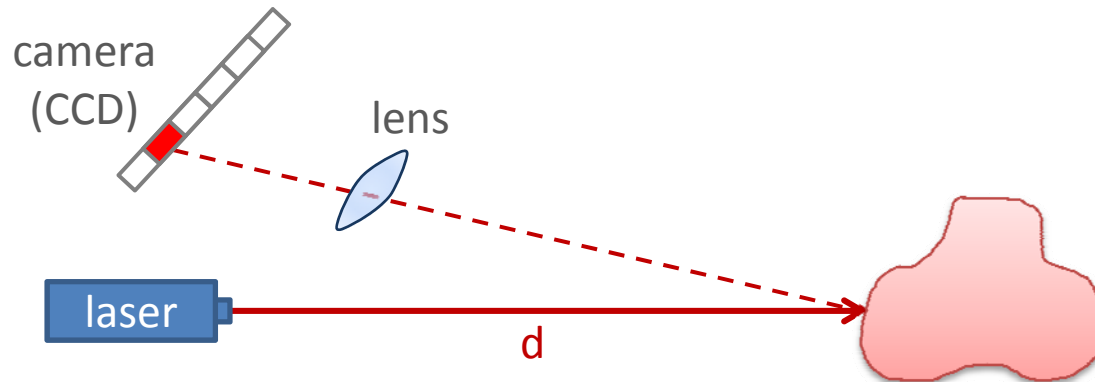
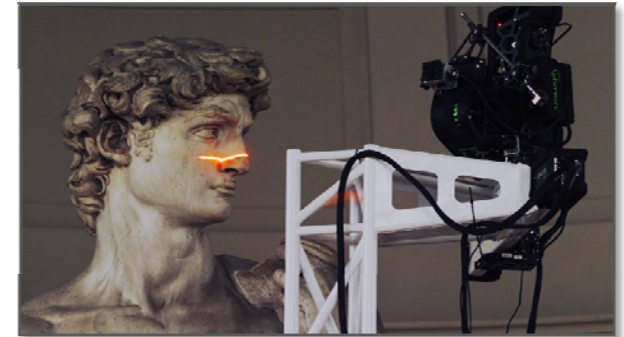
- Laser beam and camera
- Laser dot is photographed
- The location of the dot in the image allows triangulation – so we get the distance to the object



Optical scanning – active lighting

Triangulation laser

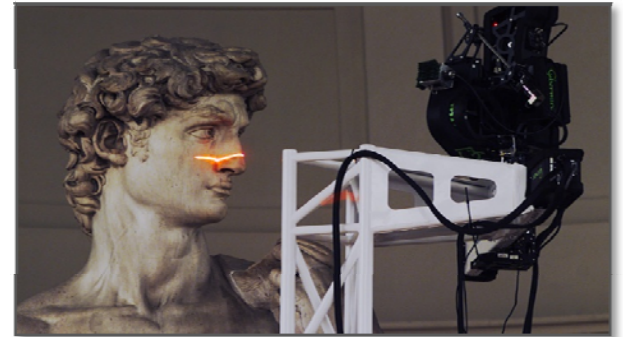
- Laser beam and camera
- Laser dot is photographed
- The location of the dot in the image allows triangulation – so we get the distance to the object



Optical scanning – active lighting

Triangulation laser

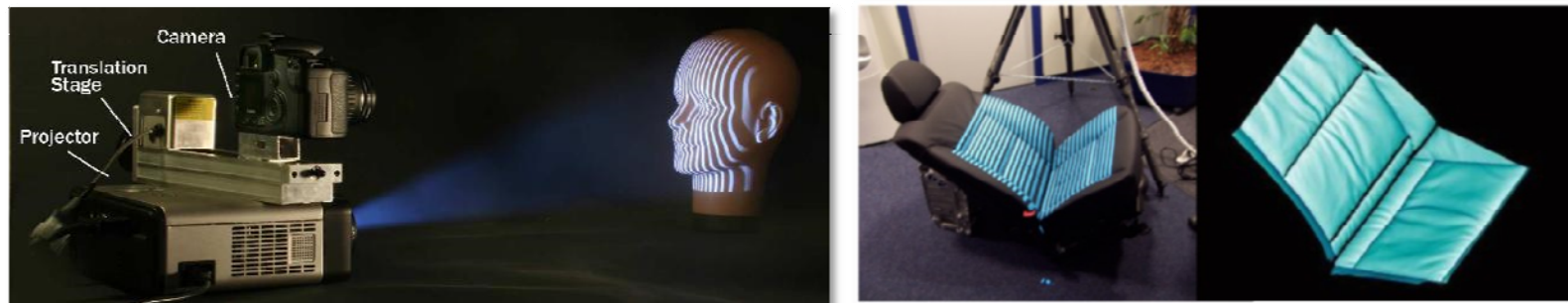
- Very precise (tens of microns)
- Small distances (meters)



Optical scanning – active lighting

Structured light

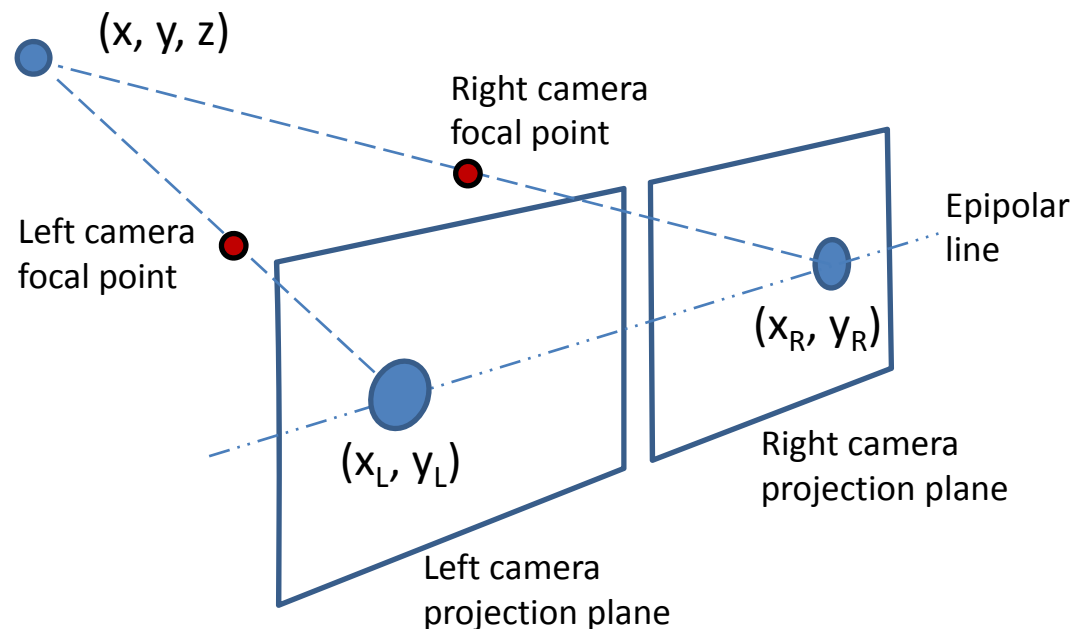
- Pattern of visible light is projected onto the object
- The distortion of the pattern, recorded by the camera, provides geometric information
- Very fast – 2D pattern at once, not single dots/lines
 - Even in real time
- Complex distance calculation, prone to noise



Optical scanning – passive

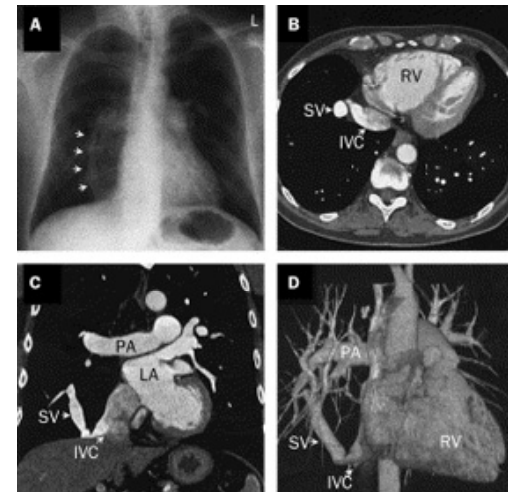
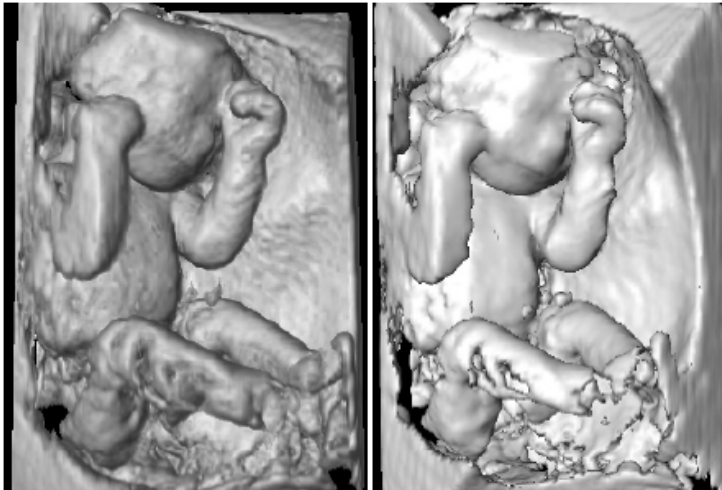
Stereo

- No need for special lighting/radiation
- Two (or more) cameras
- Feature matching and triangulation



Imaging

- Ultrasound, CT, MRI
- Discrete volume of density data
- First need to segment the desired object (contouring)



Surface reconstruction

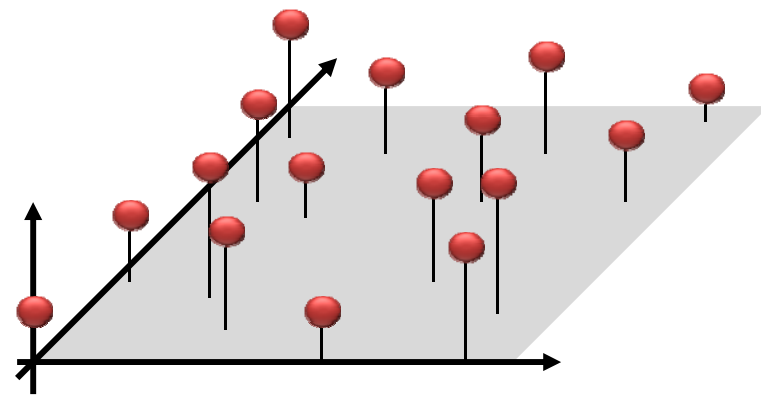
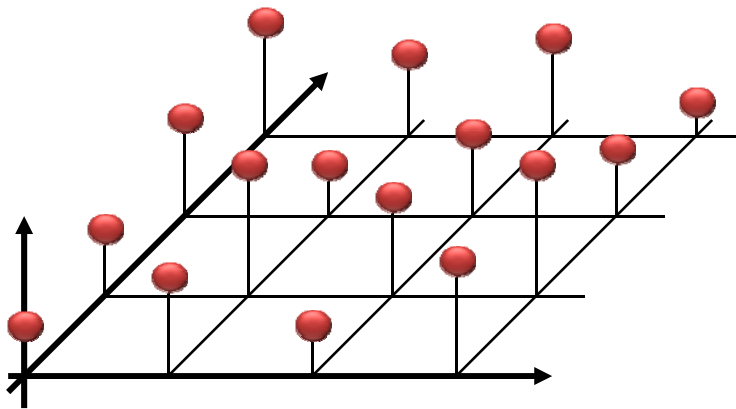
- How to create a single mesh?
 - Surface topology?
 - Smoothness?
 - How to connect the dots?



Continuous reconstruction

2D Example

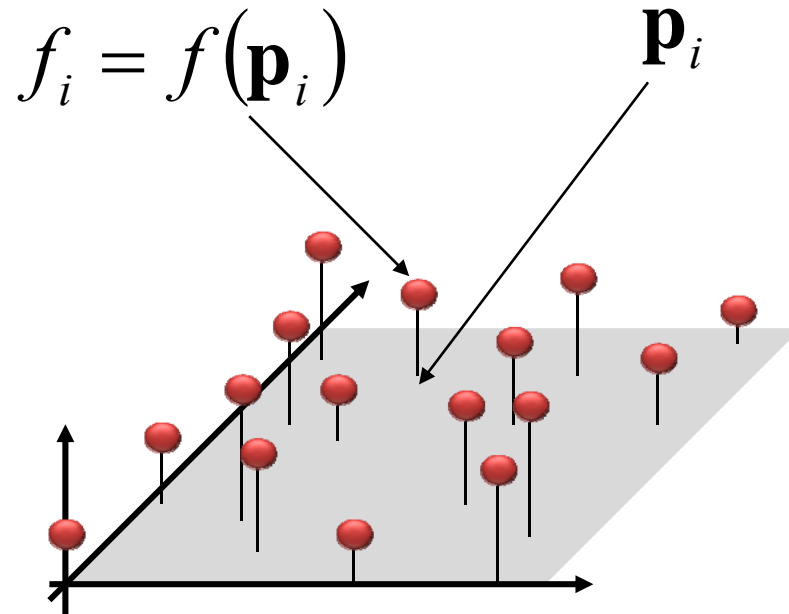
- Given a set of scattered (scalar) data points f_i at positions \mathbf{p}_i in a 2D parameter domain
- The principles are applicable to arbitrary parameter domain dimensions



Continuous reconstruction

2D Example

- Goal: approximate function f from f_i, \mathbf{p}_i

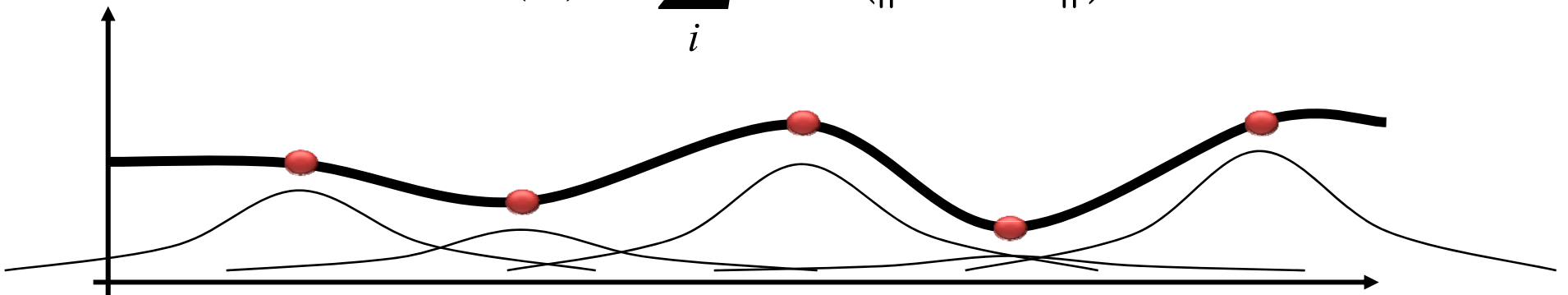


Radial Basis Functions

1D Example

- Independent of parameter domain dimension
- Function f represented as
 - Weighted sum of radial functions r
 - In the parameter domain positions \mathbf{p}_i

$$f(\mathbf{x}) = \sum_i w_i r(\|\mathbf{p}_i - \mathbf{x}\|)$$



Radial Basis Functions

Computing the coefficients

- Set

$$f_j = \sum_i w_i r(\|t_i - t_j\|)$$

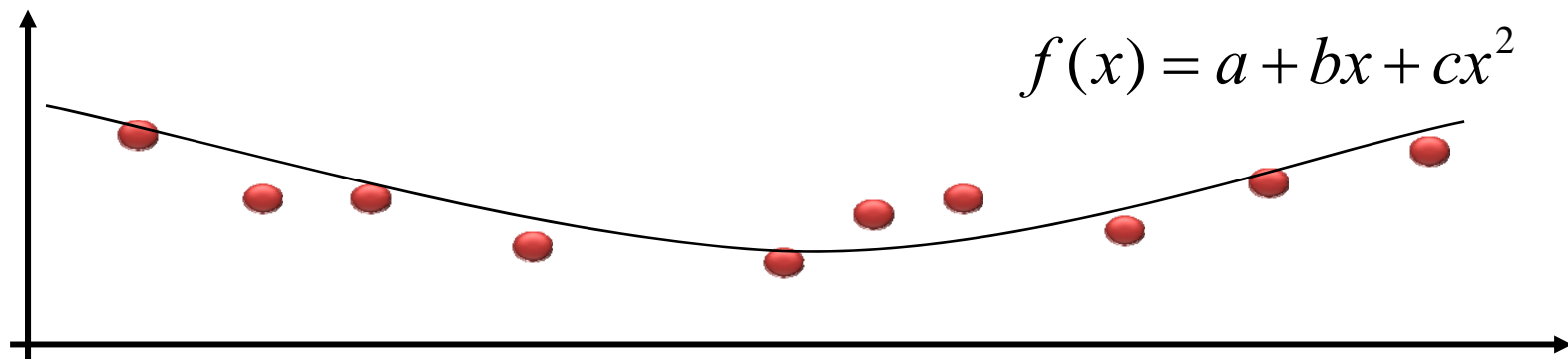
to compute the weights/coefficients w_i

- Linear system of equations

$$\begin{pmatrix} r(0) & r(\|t_0 - t_1\|) & r(\|t_0 - t_2\|) & \cdots \\ r(\|t_1 - t_0\|) & r(0) & r(\|t_1 - t_2\|) & \cdots \\ r(\|t_2 - t_0\|) & r(\|t_2 - t_1\|) & r(0) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \end{pmatrix}$$

Global Approximation

- Given $\mathbf{p}_i \in \mathbb{R}^d$, $f_i \in \mathbb{R}$, $i = 0, \dots, n$
 - \mathbf{p}_i - parameter domain positions
 - f_i - function values
- Compute polynomial curve $f(\mathbf{p}_i) \approx f_i$, $i = 0, \dots, n$



Least Squares Approximation

- Error functional

$$J_{LS} = \sum_i \|f(\mathbf{x}_i) - f_i\|^2$$

- Polynomial basis of degree m in d dimensions

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

$$\mathbf{b}(\mathbf{x}) = [b_1(\mathbf{x}), \dots, b_k(\mathbf{x})]^T \quad \mathbf{c} = [c_1, \dots, c_k]^T$$

$$\mathbf{b}(\mathbf{x}) = [1, x, y, x^2, xy, y^2]^T$$

- Previous 1D quadratic Example $f(\mathbf{x}) = c_1 + c_2x + c_3x^2$

Least Squares Approximation

- Solve for \mathbf{c} by taking (partial) derivatives of J_{LS} w.r.t. the unknowns and setting to zero

$$\partial J_{LS} / \partial c_1 = 0 : \quad \sum_i 2b_1(\mathbf{x}_i) [\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - f_i] = 0$$

$$\partial J_{LS} / \partial c_2 = 0 : \quad \sum_i 2b_2(\mathbf{x}_i) [\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - f_i] = 0$$

⋮

$$\partial J_{LS} / \partial c_k = 0 : \quad \sum_i 2b_k(\mathbf{x}_i) [\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - f_i] = 0.$$

Least Squares Approximation

- In matrix-vector notation

$$\sum_i 2\mathbf{b}(\mathbf{x}_i) [\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - f_i] =$$

$$2 \sum_i [\mathbf{b}(\mathbf{x}_i) \mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - \mathbf{b}(\mathbf{x}_i) f_i] = \mathbf{0}.$$

$$\sum_i \mathbf{b}(\mathbf{x}_i) \mathbf{b}(\mathbf{x}_i)^T \mathbf{c} = \sum_i \mathbf{b}(\mathbf{x}_i) f_i$$

- Solve for $\mathbf{c} = \left[\sum_i \mathbf{b}(\mathbf{x}_i) \mathbf{b}(\mathbf{x}_i)^T \right]^{-1} \sum_i \mathbf{b}(\mathbf{x}_i) f_i$

Least Squares Approximation

2D quadratic example

- Error functional and partial derivatives

$$f(\mathbf{x}) = a + b_u u + b_v v + c_{uu} u^2 + c_{uv} uv + c_{vv} v^2$$

$$\min_{(a,b,C)} \sum_i (f(u_i, v_i) - f_i)^2 = \min_{(a,b,C)} \sum_i (a + b_u u_i + b_v v_i + c_{uu} u_i^2 + c_{uv} u_i v_i + c_{vv} v_i^2 - f_i)^2$$

$$\frac{\partial \sum_i (f(u_i, v_i) - f_i)^2}{\partial a} = \sum_i 2(a + b_u u_i + b_v v_i + c_{uu} u_i^2 + c_{uv} u_i v_i + c_{vv} v_i^2 - f_i) = 0$$

⋮

$$\frac{\partial \sum_i (f(u_i, v_i) - f_i)^2}{\partial c_{vv}} = \sum_i 2v_i^2 (a + b_u u_i + b_v v_i + c_{uu} u_i^2 + c_{uv} u_i v_i + c_{vv} v_i^2 - f_i) = 0$$

Least Squares Approximation

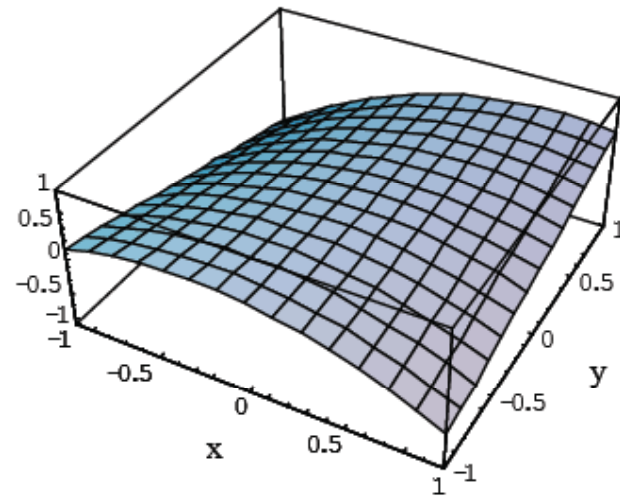
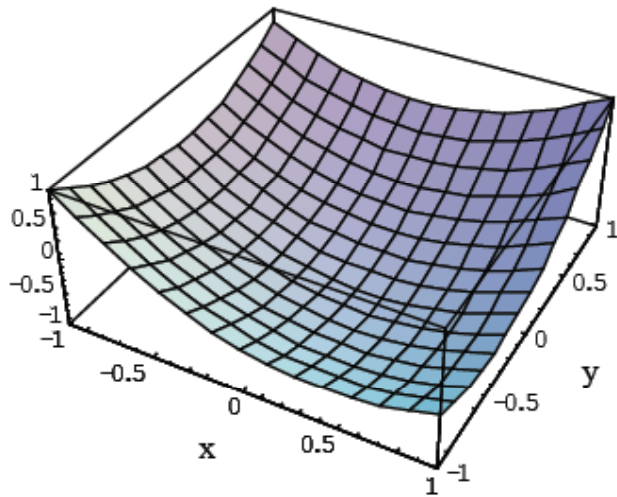
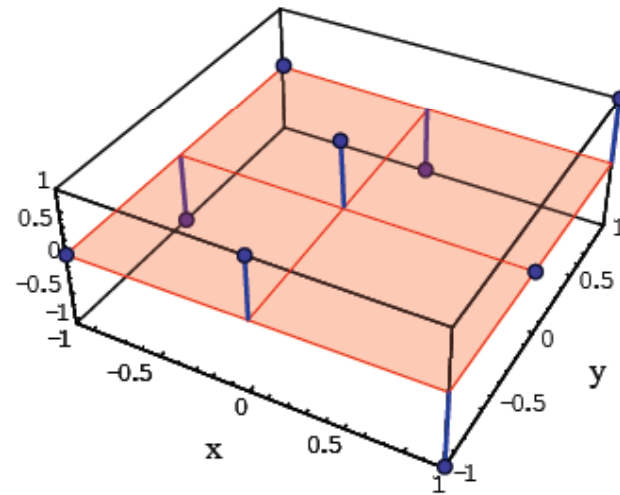
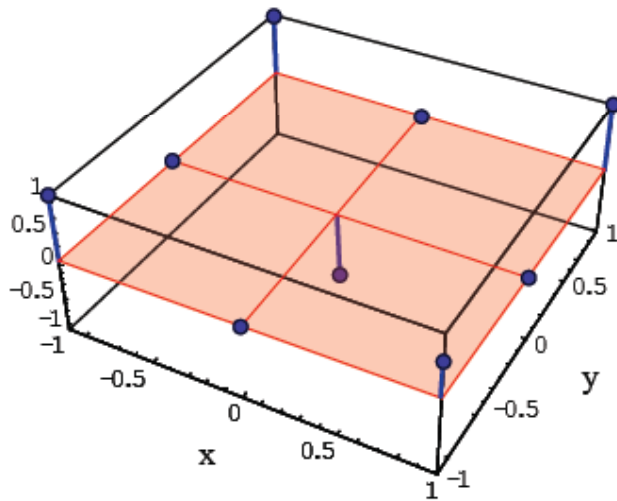
2D quadratic example

- Linear system of equations

$$\sum_i \begin{pmatrix} 1 & u_i & v_i & u_i^2 & u_i v_i & v_i^2 \\ u_i & u_i^2 & u_i v_i & u_i^3 & u_i^2 v_i & u_i v_i^2 \\ v_i & u_i v_i & v_i^2 & u_i^2 v_i & u_i v_i^2 & v_i^3 \\ u_i^2 & u_i^3 & u_i^2 v_i & u_i^4 & u_i^3 v_i & v_i^2 u_i^2 \\ u_i v_i & u_i^2 v_i & u_i v_i^2 & u_i^3 v_i & u_i^2 v_i^2 & u_i v_i^3 \\ v_i^2 & v_i^2 u_i & v_i^3 & v_i^2 u_i^2 & u_i v_i^3 & v_i^4 \end{pmatrix} \begin{pmatrix} a \\ b_u \\ b_v \\ c_{uu} \\ c_{uv} \\ c_{vv} \end{pmatrix} = \sum_i f_i \begin{pmatrix} 1 \\ u_i \\ v_i \\ u_i^2 \\ u_i v_i \\ v_i^2 \end{pmatrix}$$

Least Squares Approximation

Results



Least Squares Approximation

Normal equations

Method of Normal Equations. For a different but also very common notation, note that the solution for \mathbf{c} solves the following (generally over-constrained) LSE ($\mathbf{B}\mathbf{c} = \mathbf{f}$) in the least-squares sense

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

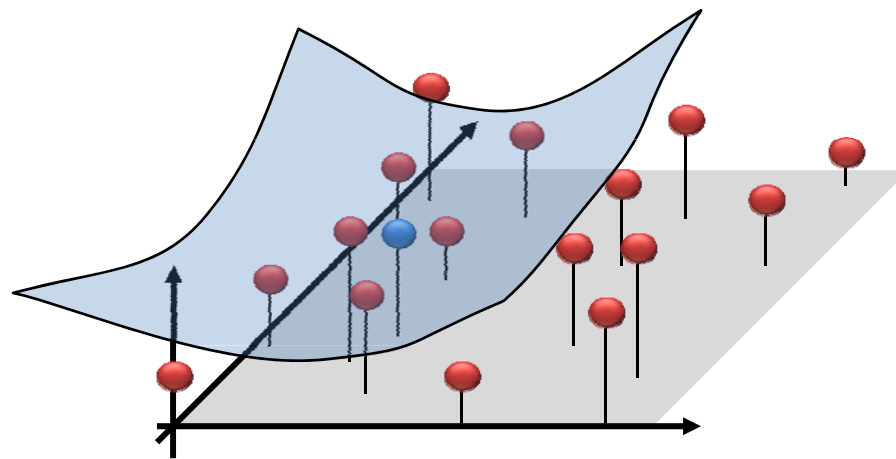
using the method of normal equations

$$\begin{aligned} \mathbf{B}^T \mathbf{B} \mathbf{c} &= \mathbf{B}^T \mathbf{f} \\ \mathbf{c} &= (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T \mathbf{f}. \end{aligned}$$

Weighted Least Squares

- Principle: local approximation at $\bar{\mathbf{x}}$ by weighting the squared errors based on proximity in the parameter domain

$$\min_{f_{\mathbf{x}} \in \Pi_k^d} \sum_{i=0}^n \|f(\mathbf{p}_i) - f_i\|^2 \theta(\|\mathbf{p}_i - \bar{\mathbf{x}}\|)$$



Weighted Least Squares

Weighting functions

- Gaussian $\theta(d) = e^{-\frac{d^2}{h^2}}$
 - h is a smoothing parameter

- Wendland function

$$\theta(d) = (1 - d/h)^4 (4d/h + 1)$$

- Defined in $[0, h]$ and

$$\theta(0) = 1, \theta(h) = 0, \theta'(h) = 0 \text{ and } \theta''(h) = 0$$

- Singular function

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

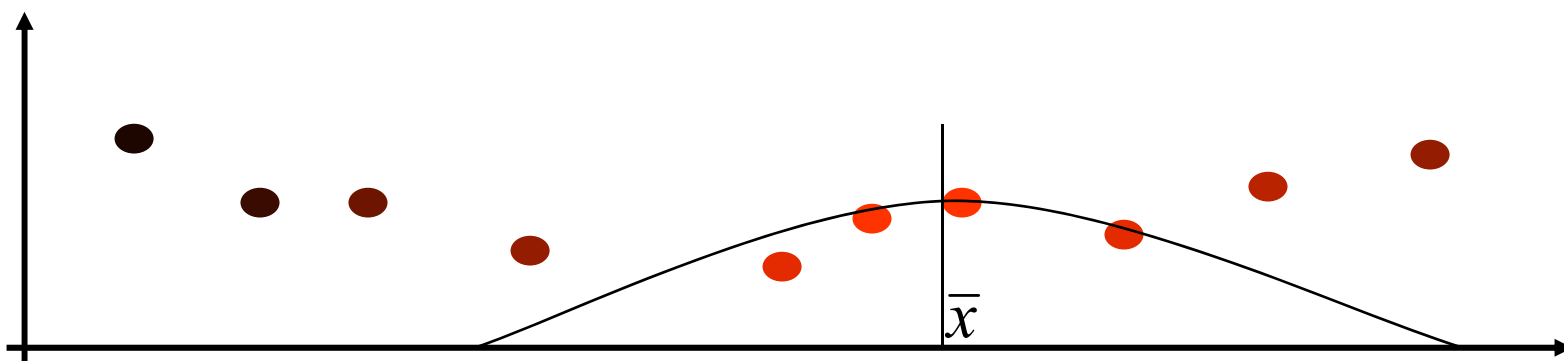
- For small ε , weights large near $d=0$ (interpolation)

Moving Least Squares

Parametric 1D example

- Principle: “construct” a global function from infinitely many locally weighted functions

$$f(\mathbf{x}) = f_{\bar{\mathbf{x}}}(\mathbf{x}), \quad \min_{f_{\mathbf{x}} \in \Pi_k^d} \sum_{i=0}^n \|f(\mathbf{p}_i) - f_i\|^2 \theta(\|\mathbf{p}_i - \bar{\mathbf{x}}\|)$$



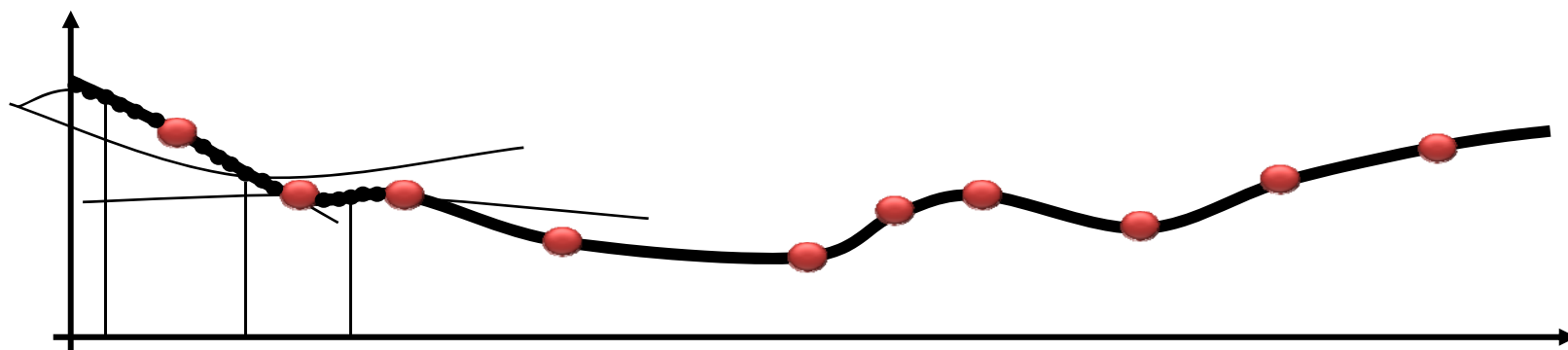
Moving Least Squares

Parametric 1D example

- The infinite set

$$f(\mathbf{x}) = f_{\bar{\mathbf{x}}}(\mathbf{x}), \quad \min_{f_{\mathbf{x}} \in \Pi_k^d} \sum_{i=0}^n \|f(\mathbf{p}_i) - f_i\|^2 \theta(\|\mathbf{p}_i - \bar{\mathbf{x}}\|)$$

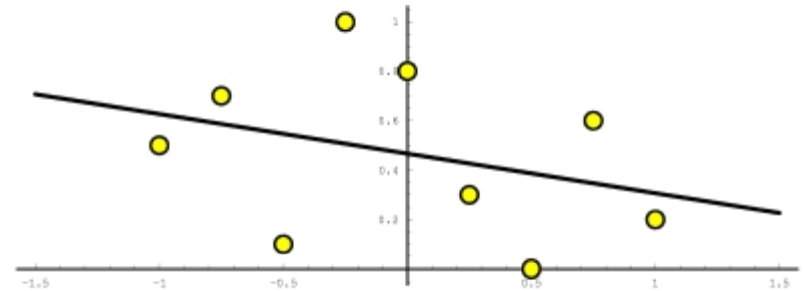
is continuously differentiable if and only if θ is continuously differentiable



LS, MLS and Weight Functions

Linear polynomial fit

- Global least squares



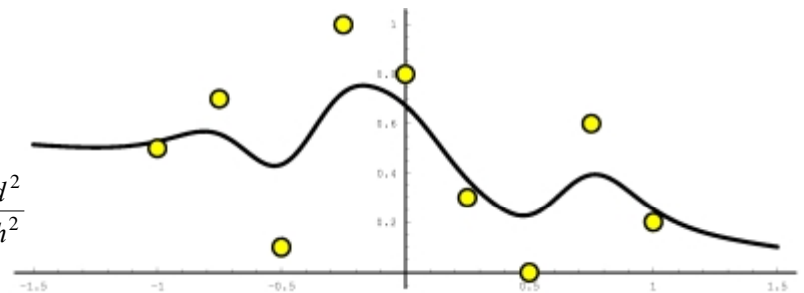
- MLS with (near) singular weight function

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



- MLS with approximating weight function

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

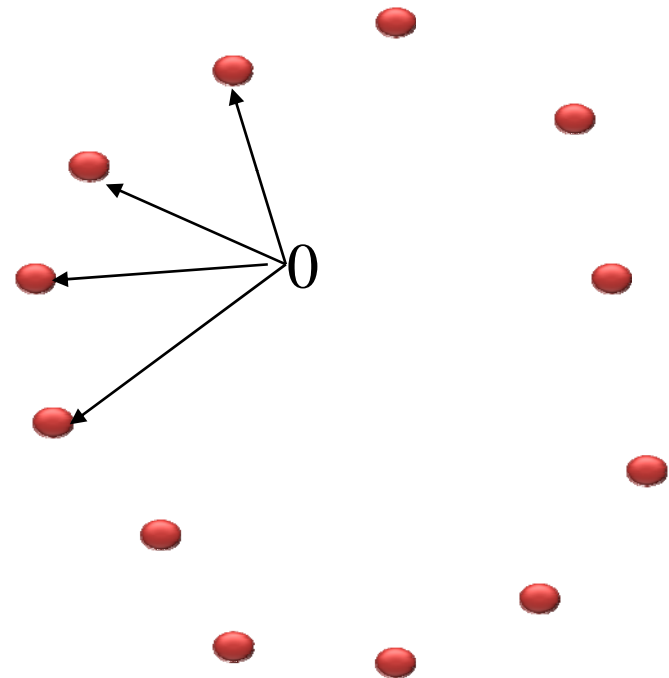


Implicit Surface Reconstruction

Distance Field Reconstruction

2D example

- Idea: construct a distance field on the points
- Implicit function
$$f(\mathbf{p}_i) = 0$$
for the points \mathbf{p}_i
- Trivial solution $f = 0$
- Requires additional constraints



Distance Field Reconstruction

[Hoppe et al. 1992]

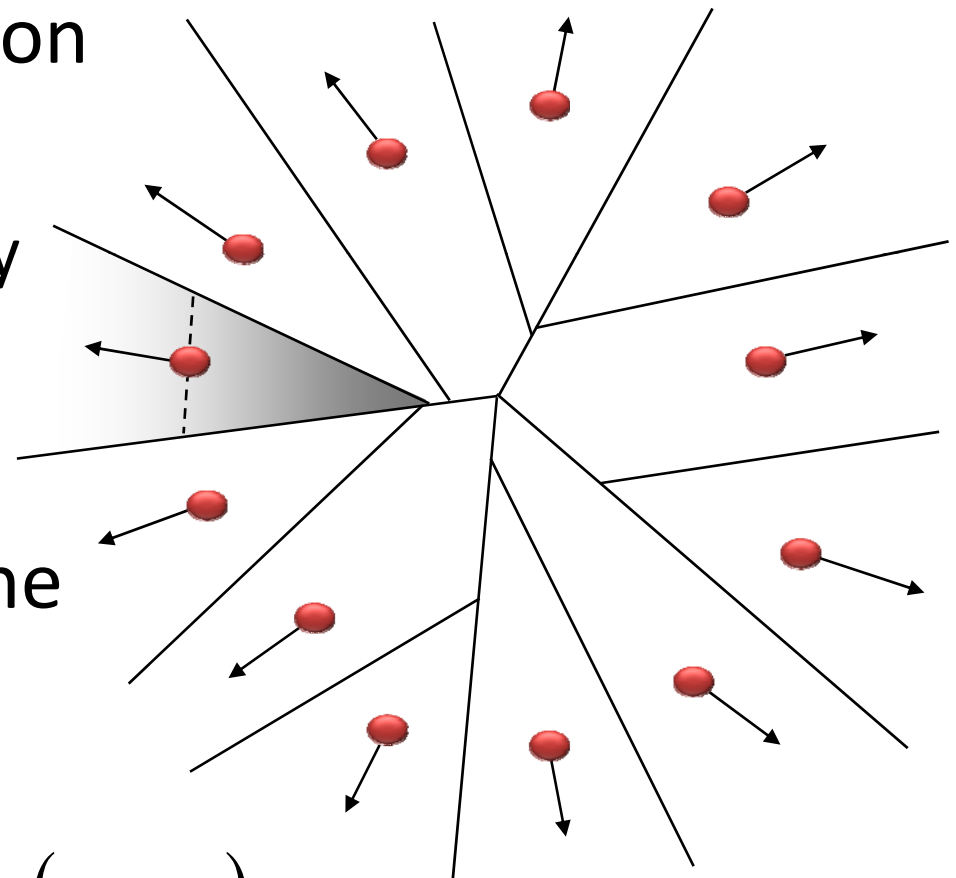
- Linear distance function per point

- Direction is defined by surface normal

$$f_i(\mathbf{x}) = \mathbf{n}_i \cdot (\mathbf{x} - \mathbf{p}_i)$$

- Distance in space is the minimum of all local distance functions

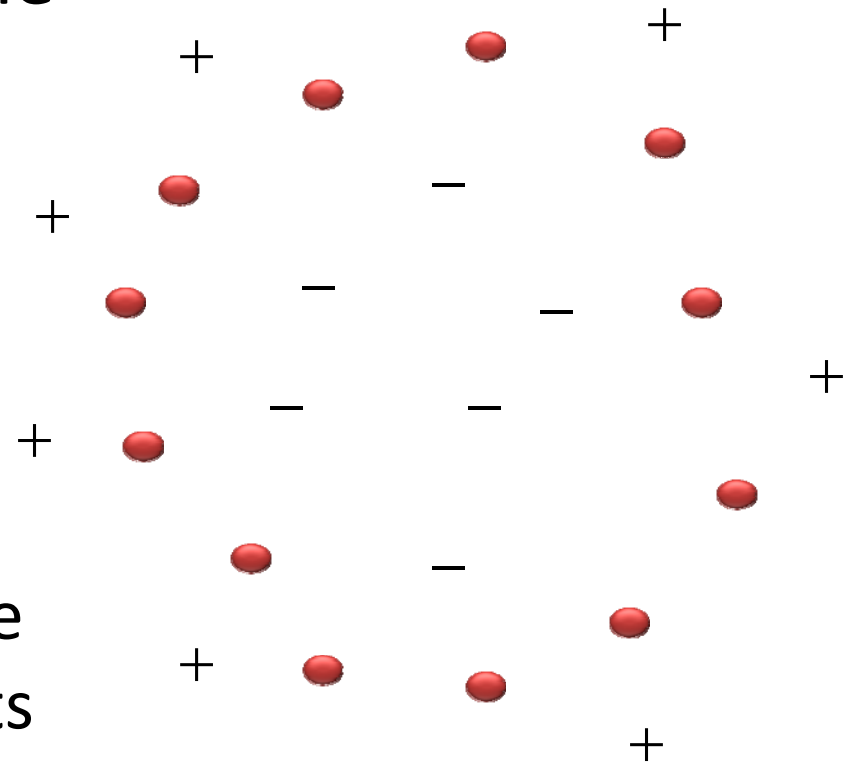
$$f(\mathbf{x}) = \min_i f_i(\mathbf{x}) = \min_i \mathbf{n}_i \cdot (\mathbf{x} - \mathbf{p}_i)$$



Distance Field Reconstruction

Inside + outside point constraints

- Additional data to define inside and outside
- Basic idea [Turk and O'Brien 1999]
 - Insert additional value constraints manually
 - These constraints can be added as soft constraints with low(er) weight



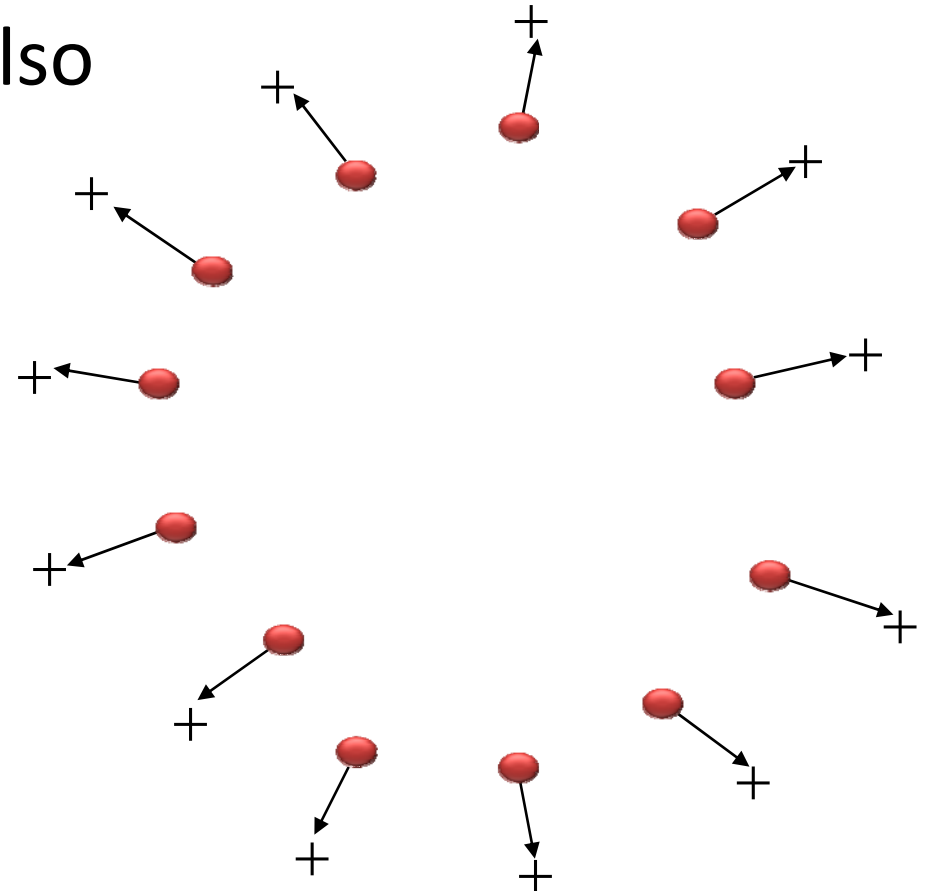
Distance Field Reconstruction

Inside + outside point constraints

- This information can also be obtained from surface normals

$$f(\mathbf{p}_i + \alpha \mathbf{n}_i) = \alpha$$

- Some acquisition devices provide normals
- If not, they must be locally approximated



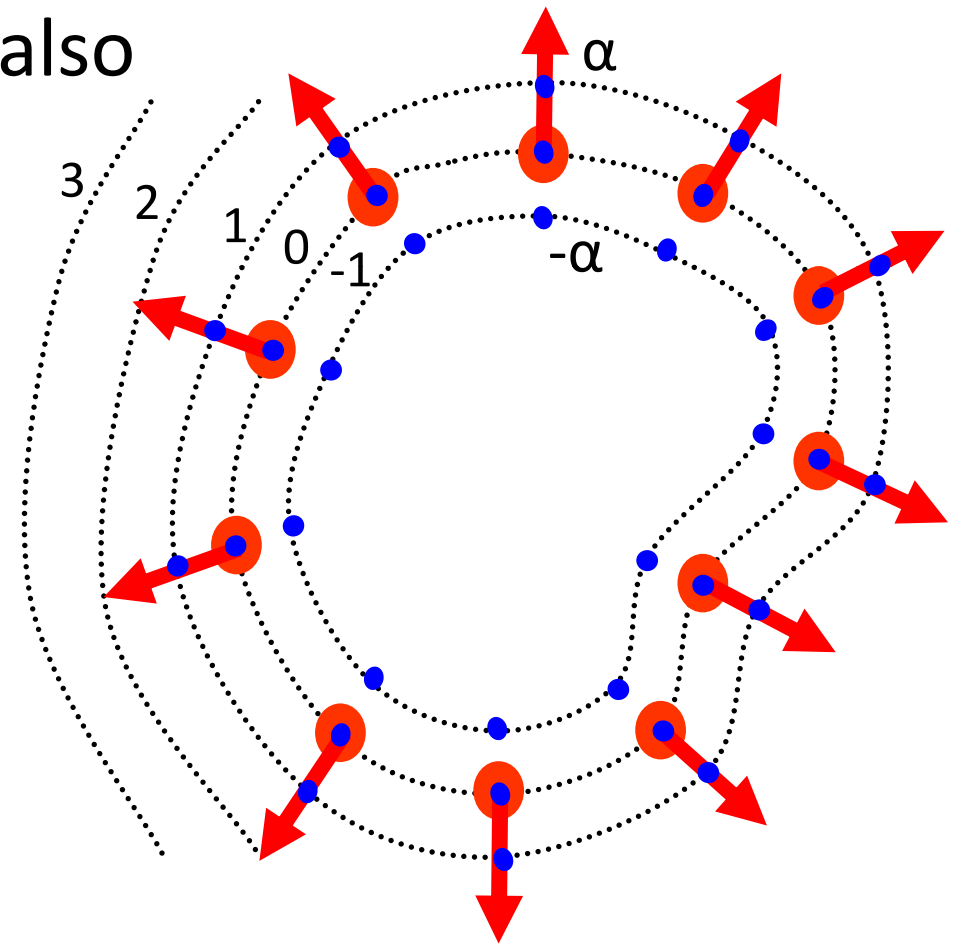
Distance Field Reconstruction

Inside + outside point constraints

- This information can also be obtained from surface normals

$$f(\mathbf{p}_i + \alpha \mathbf{n}_i) = \alpha$$

- Some acquisition devices provide normals
- If not, they must be locally approximated



Distance Field Reconstruction

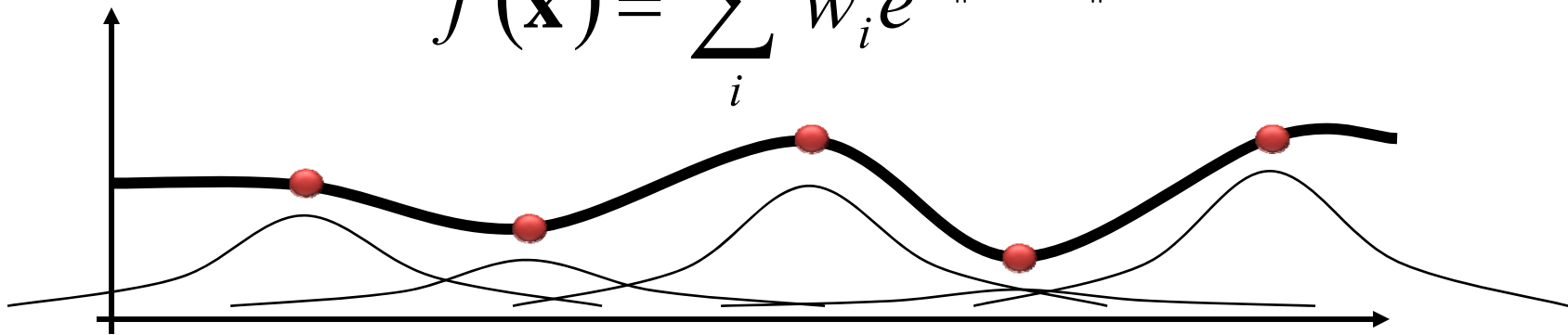
Radial basis functions (RBFs)

- Similar to parametric case
- Given points and normals $\mathbf{p}_i, \mathbf{n}_i$ construct a function with

$$f(\mathbf{p}_i) = 0, \quad f(\mathbf{p}_i + \alpha \mathbf{n}_i) = \alpha$$

- Possible solution: Gaussian RBFs

$$f(\mathbf{x}) = \sum_i w_i e^{-\|p_i - \mathbf{x}\|^2}$$



Distance Field Reconstruction

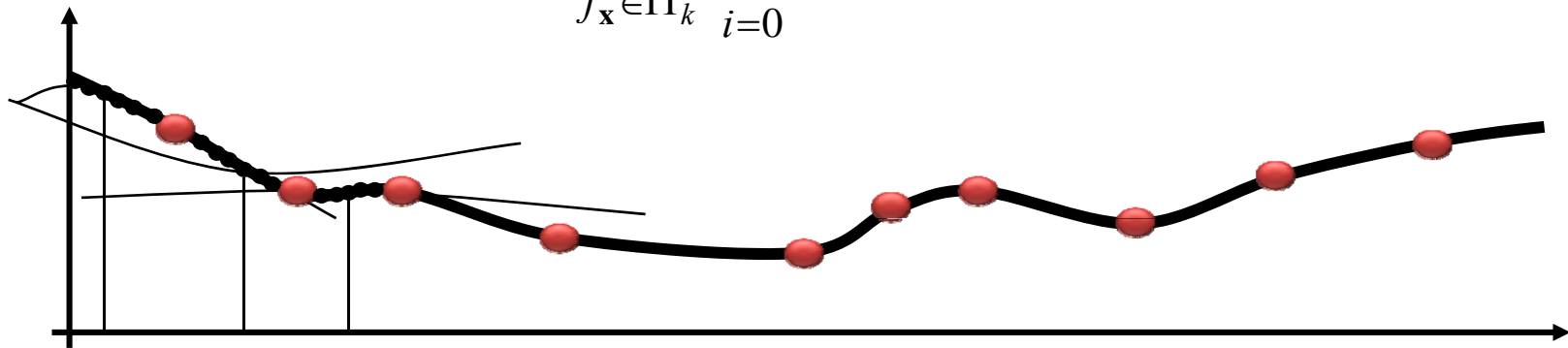
Moving least squares (MLS)

- Given points and normals $\mathbf{p}_i, \mathbf{n}_i$
construct a function with

$$f(\mathbf{p}_i) = 0, \quad f(\mathbf{p}_i + \alpha \mathbf{n}_i) = \alpha$$

using the moving least squares technique

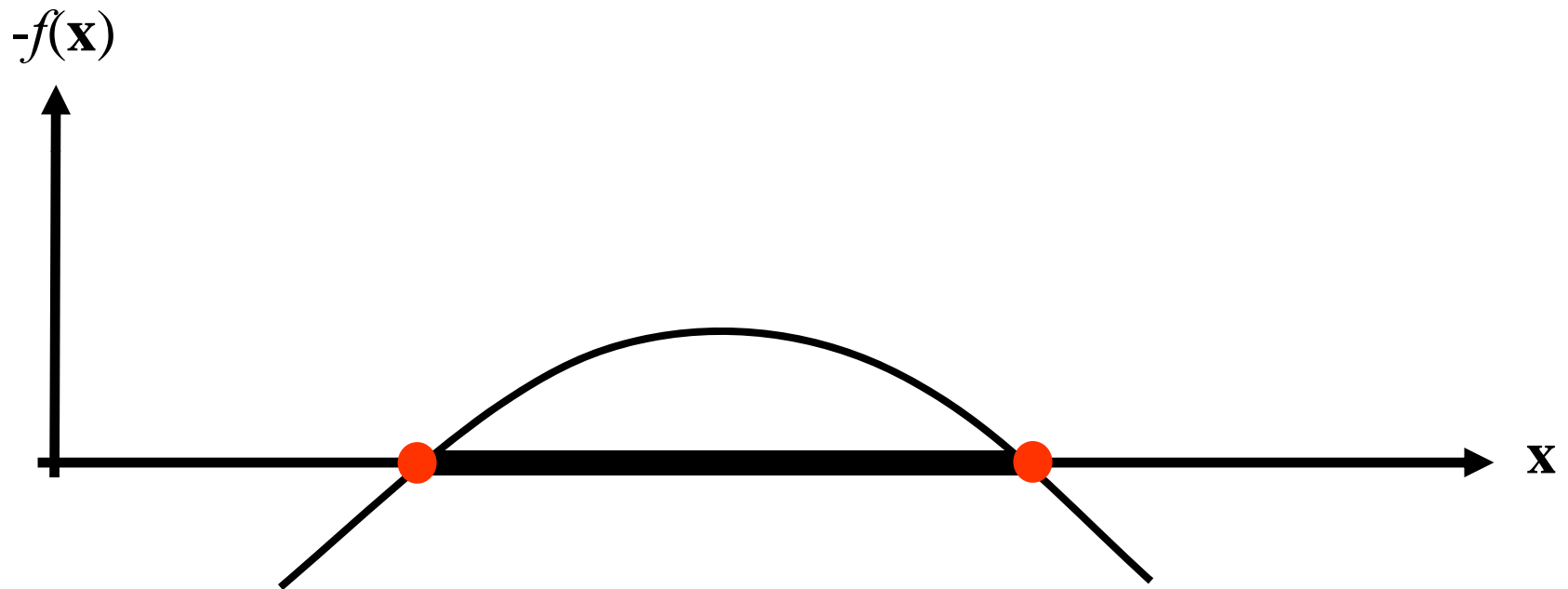
$$f(\mathbf{x}) = f_{\bar{\mathbf{x}}}(\mathbf{x}), \quad \min_{f_{\mathbf{x}} \in \Pi_k^d} \sum_{i=0}^n \|f(\mathbf{p}_i) - f_i\|^2 \theta(\|\mathbf{p}_i - \bar{\mathbf{x}}\|)$$



MLS Distance Field

1D example

- One dimensional Implicit function



\mathbf{p}_i



\mathbf{n}_i



Approximation



Constraint



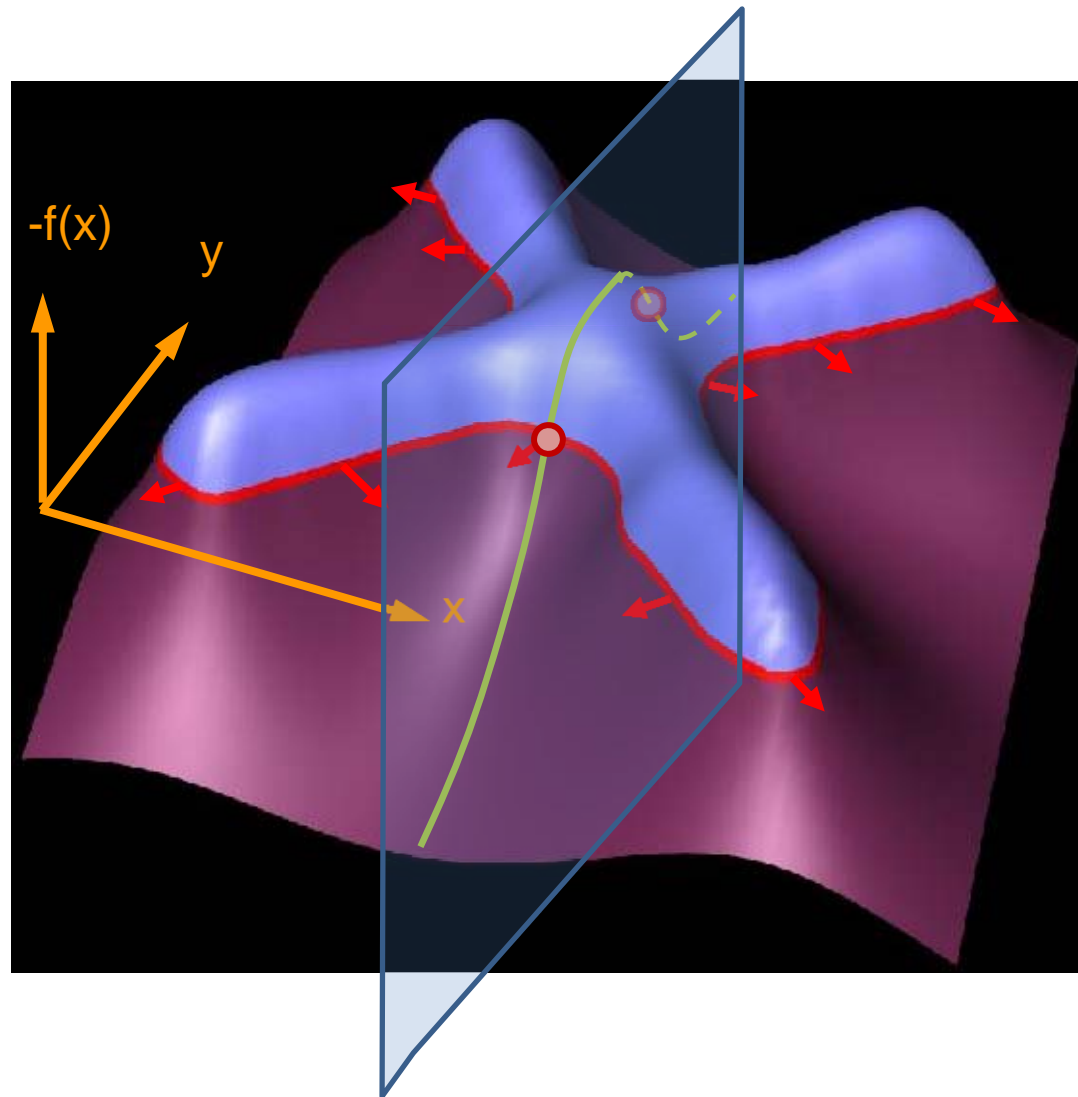
$f(\mathbf{x})$



Weighting

MLS Distance Field

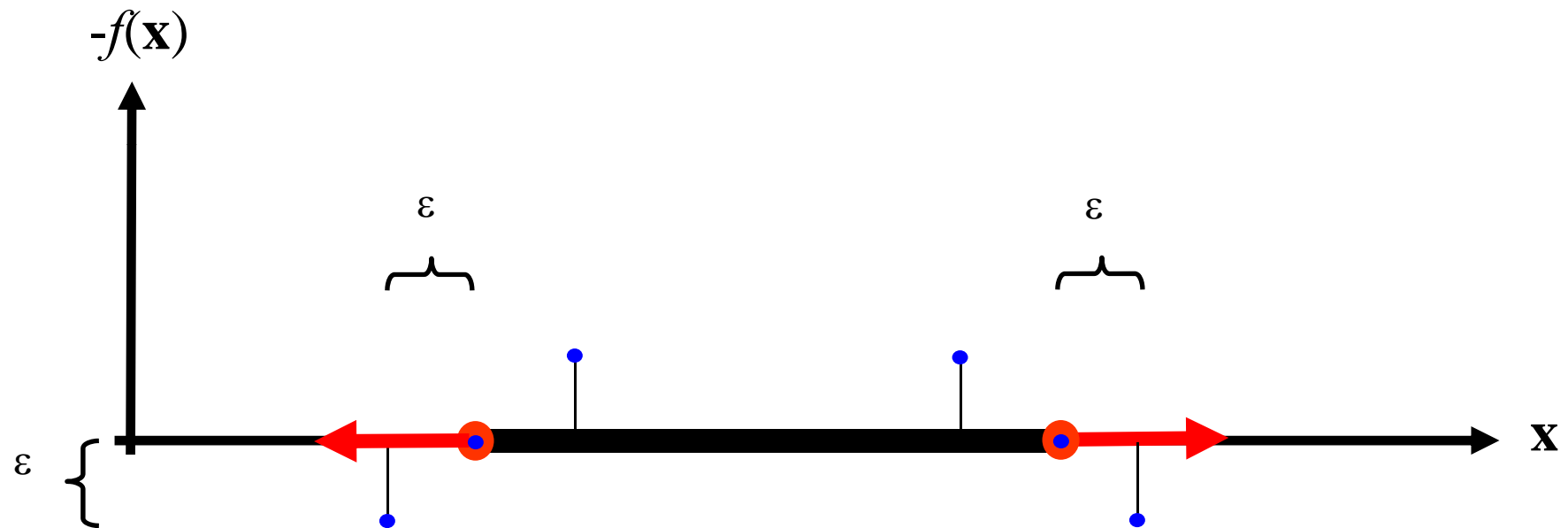
1D slice of a 2D height field



MLS Distance Field

1D example

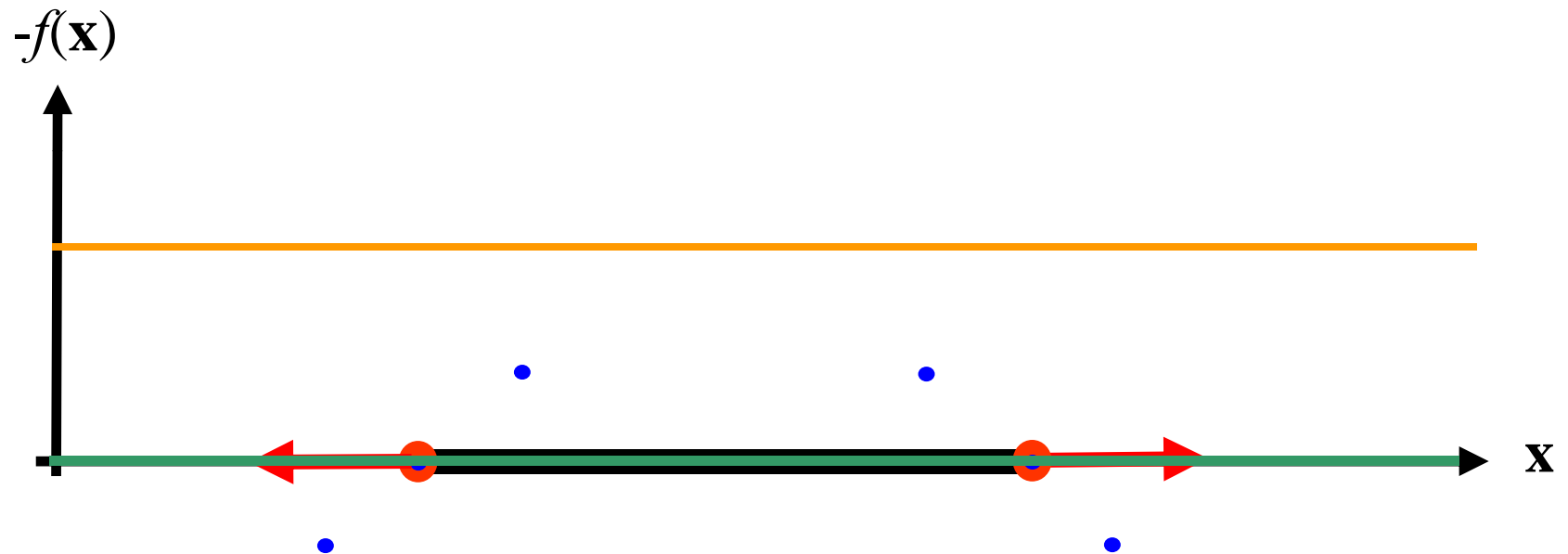
- Adding inside + outside constraints



MLS Distance Field

1D example

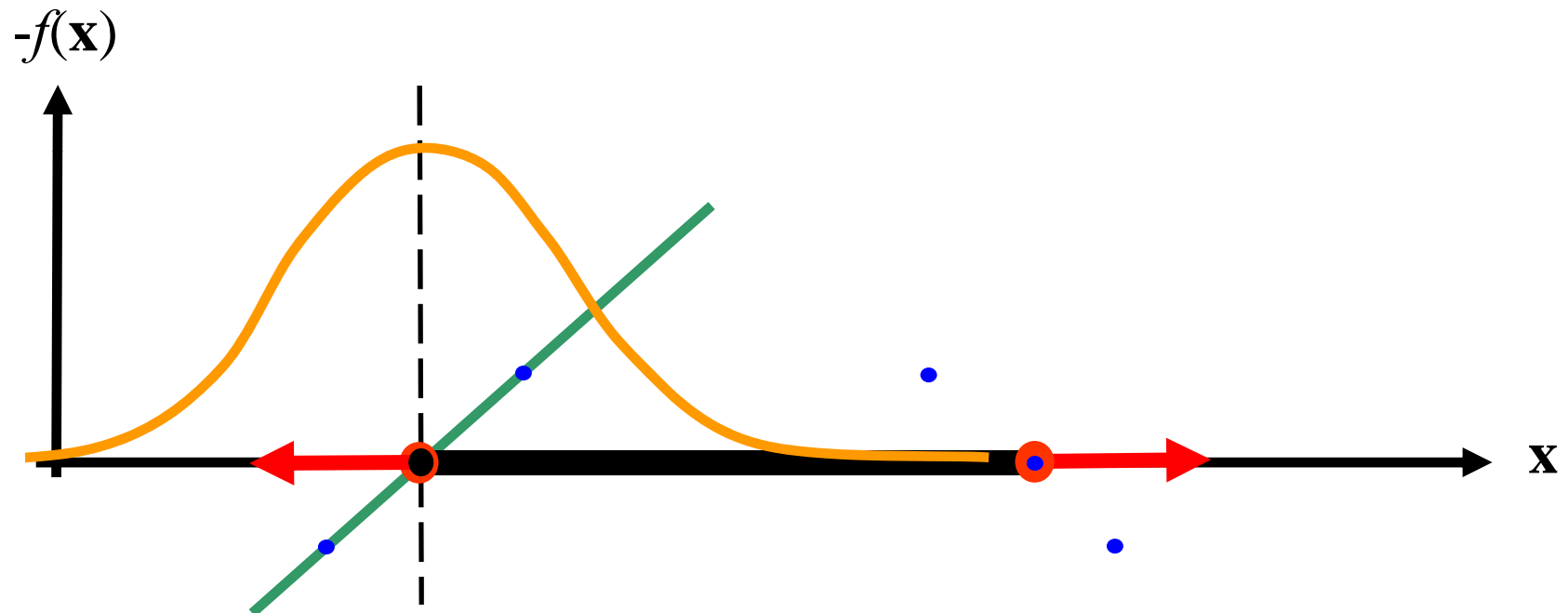
- Linear polynomial fit (uniform weights)



MLS Distance Field

1D example

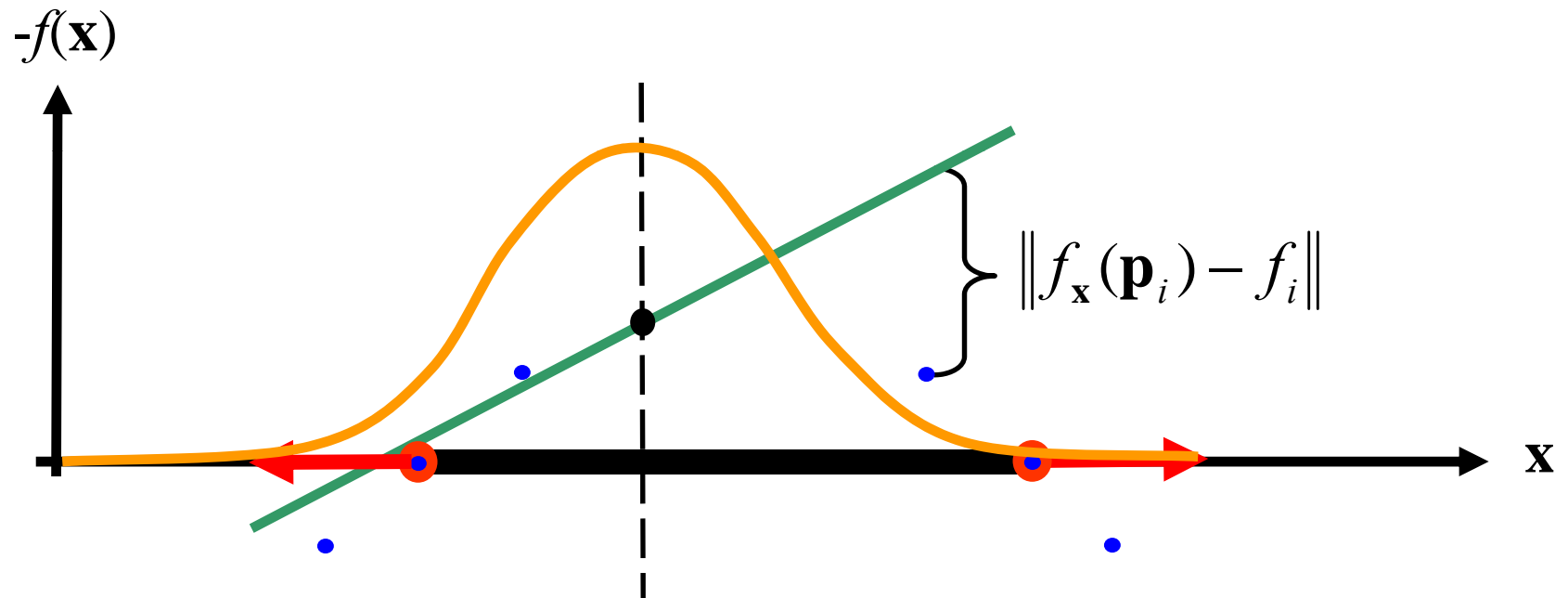
- Linear polynomial fit (Gaussian weights)



MLS Distance Field

1D example

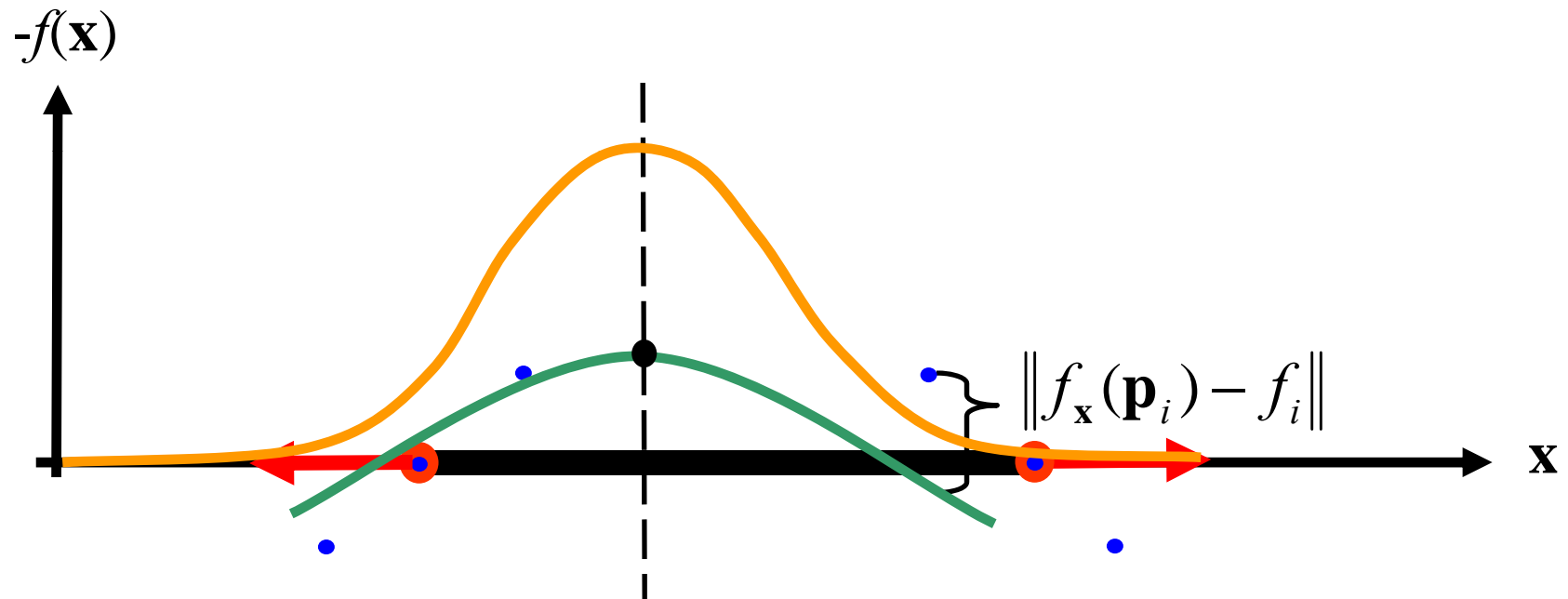
- Linear polynomial fit (Gaussian weights)



MLS Distance Field

1D example

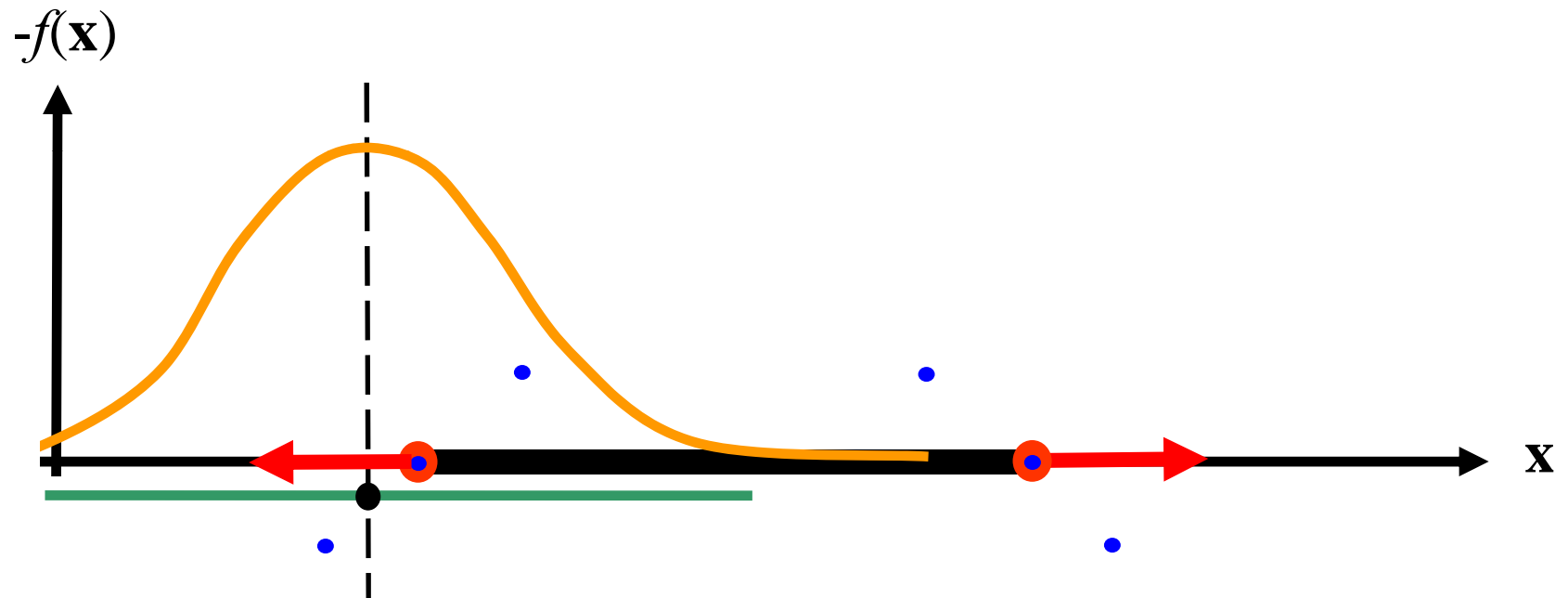
- Quadratic polynomial fit (Gaussian weights)



MLS Distance Field

1D example

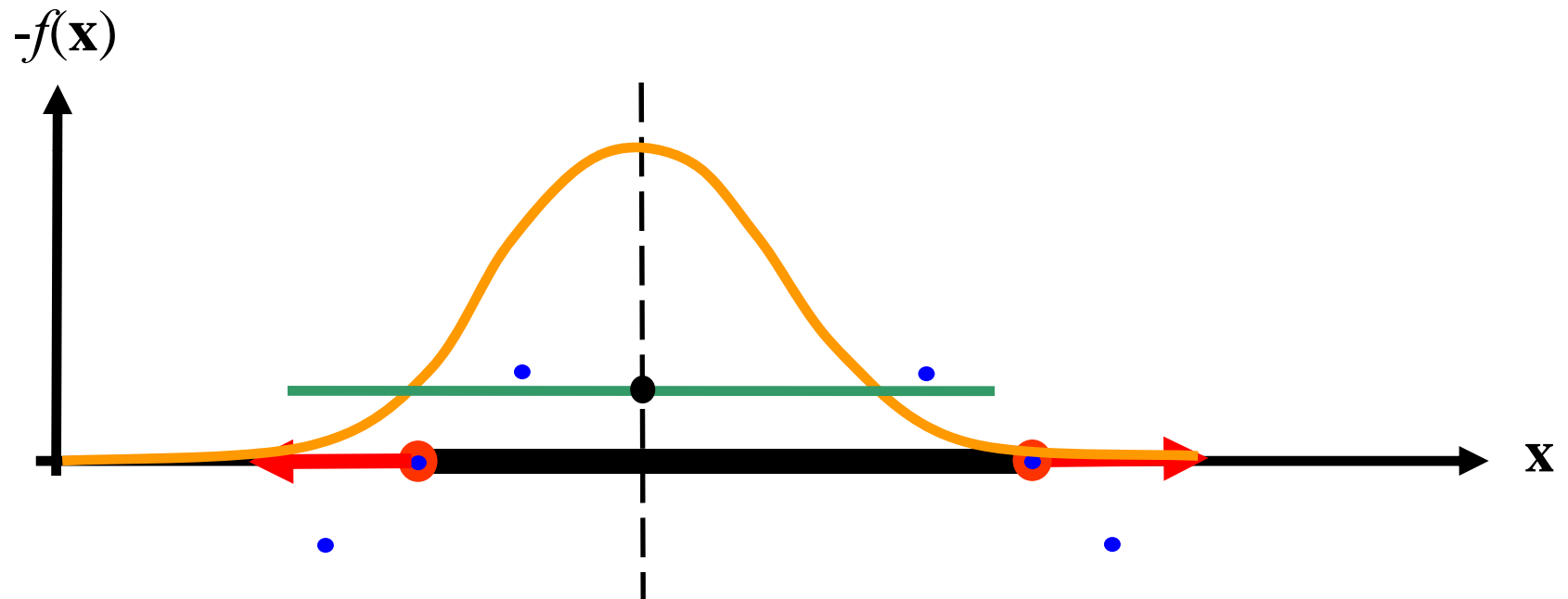
- Constant polynomial fit (Gaussian weights)



MLS Distance Field

1D example

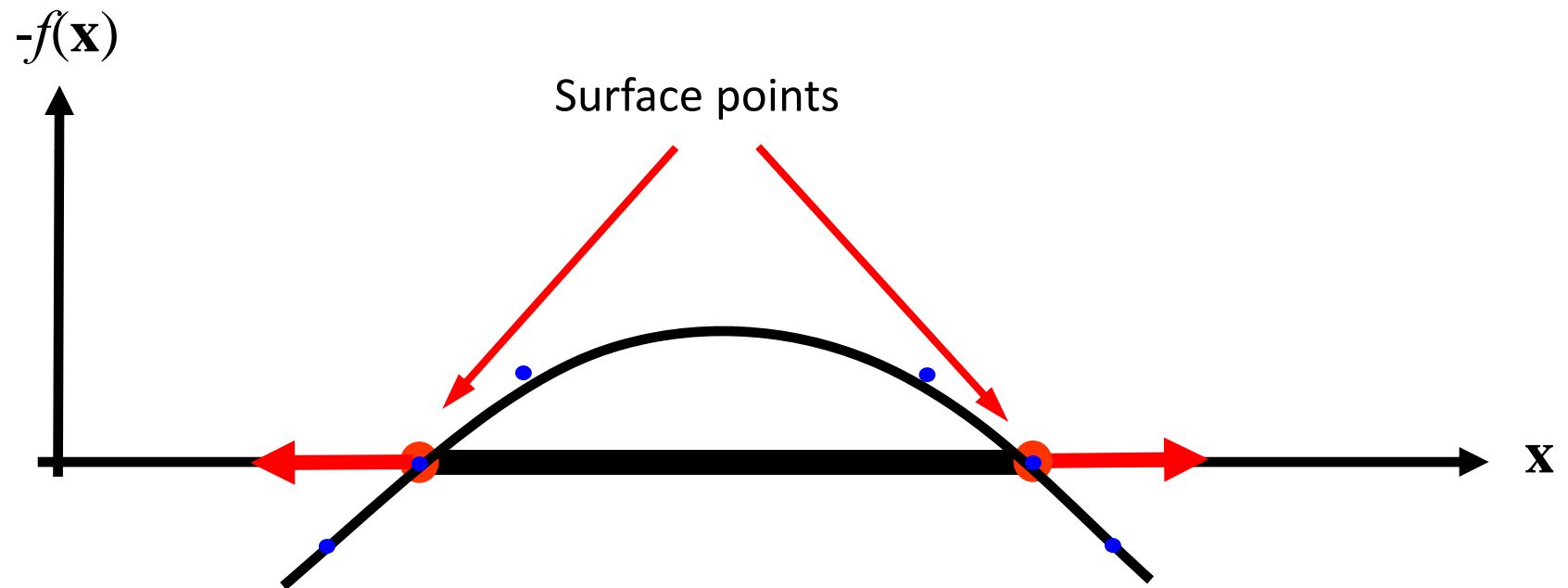
- Constant polynomial fit (Gaussian weights)



MLS Distance Field

1D example

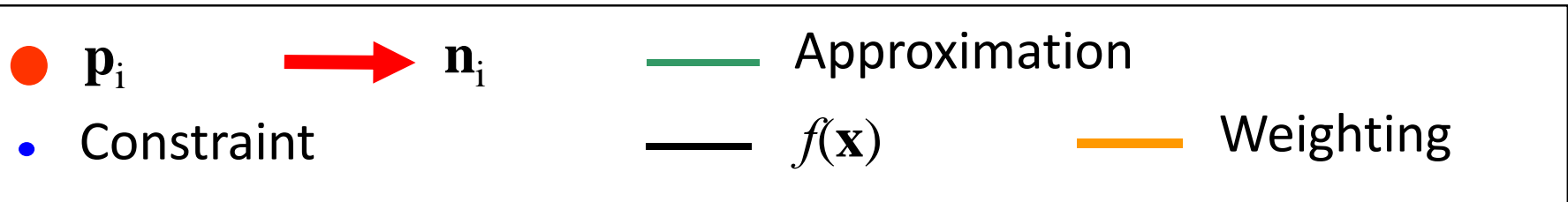
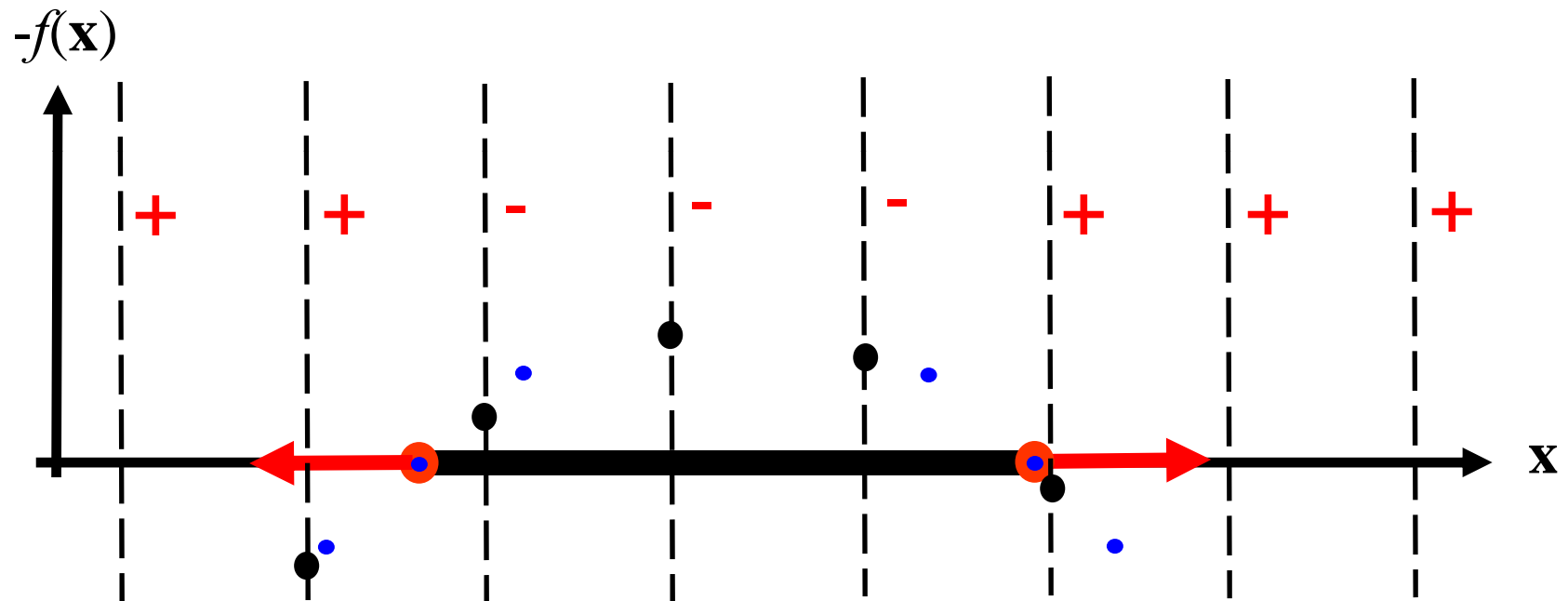
■ MLS approximation results



MLS Distance Field

1D example

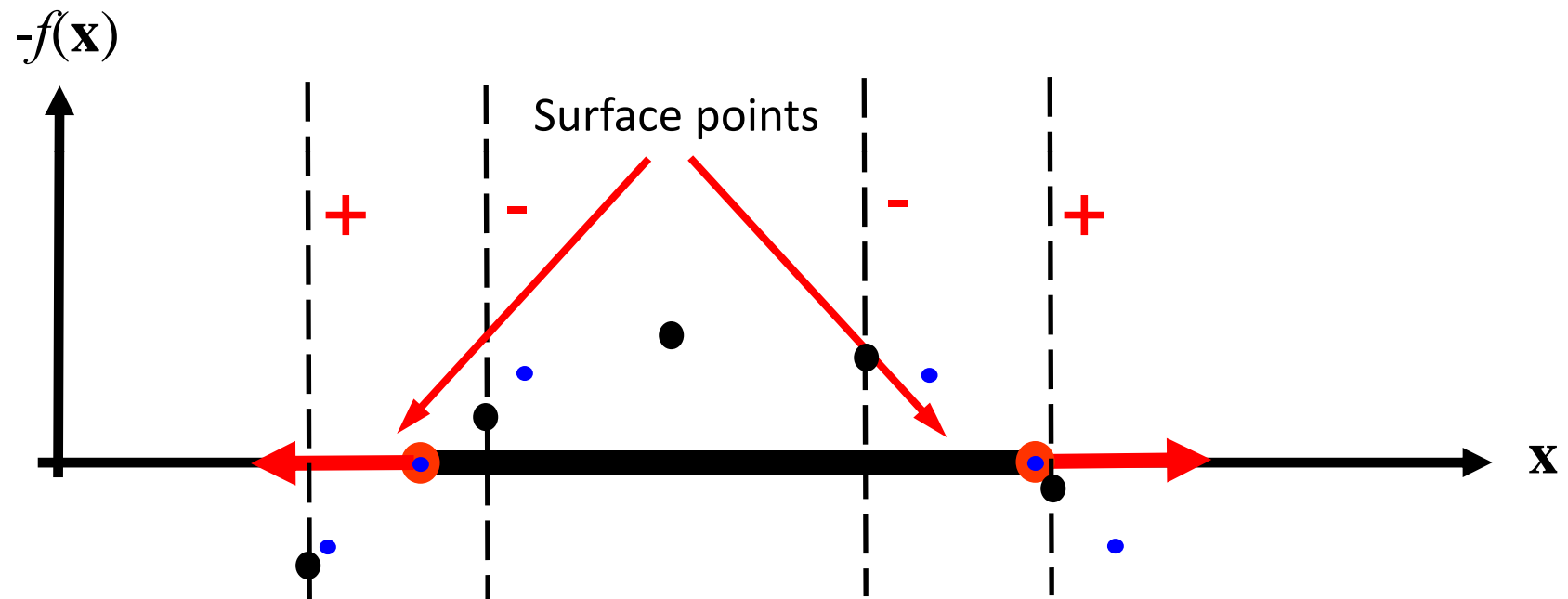
- Discrete evaluation with marching cubes (3D)



MLS Distance Field

1D example

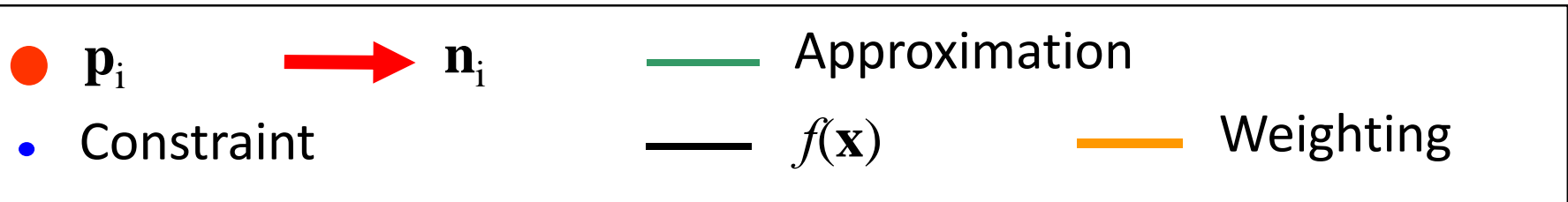
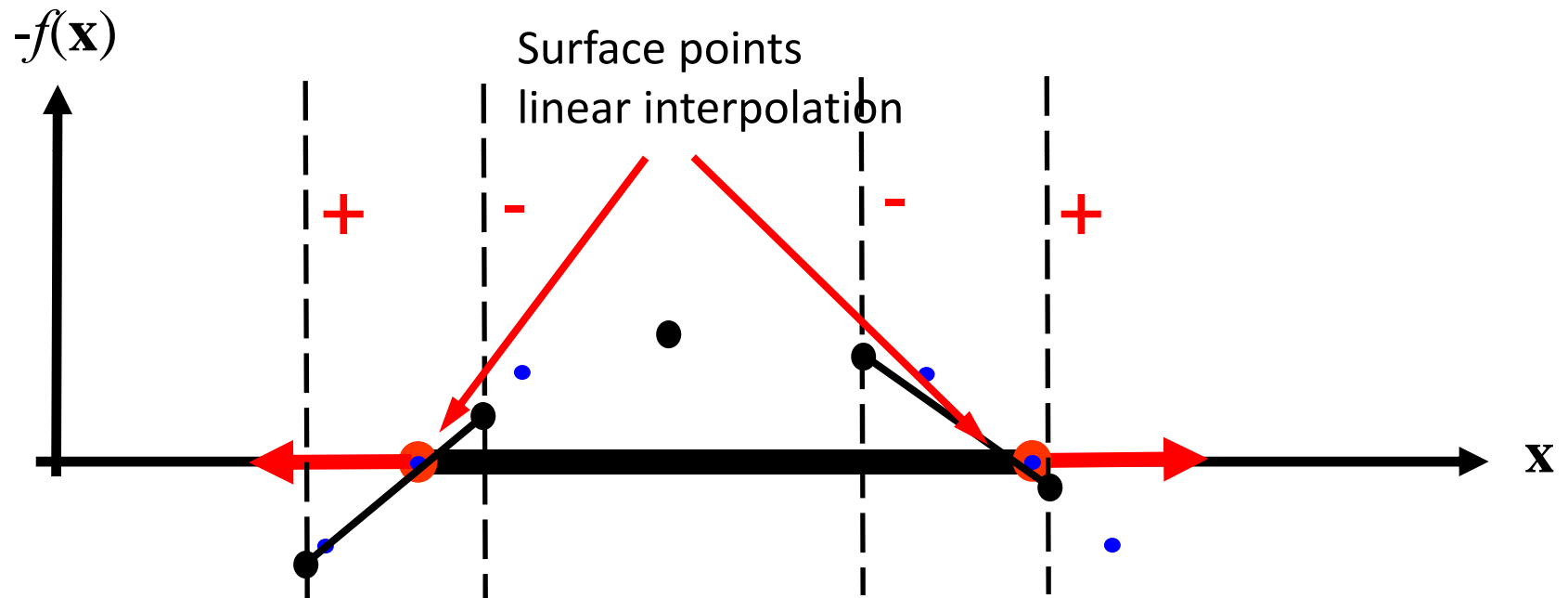
- Discrete evaluation with marching cubes (3D)



MLS Distance Field

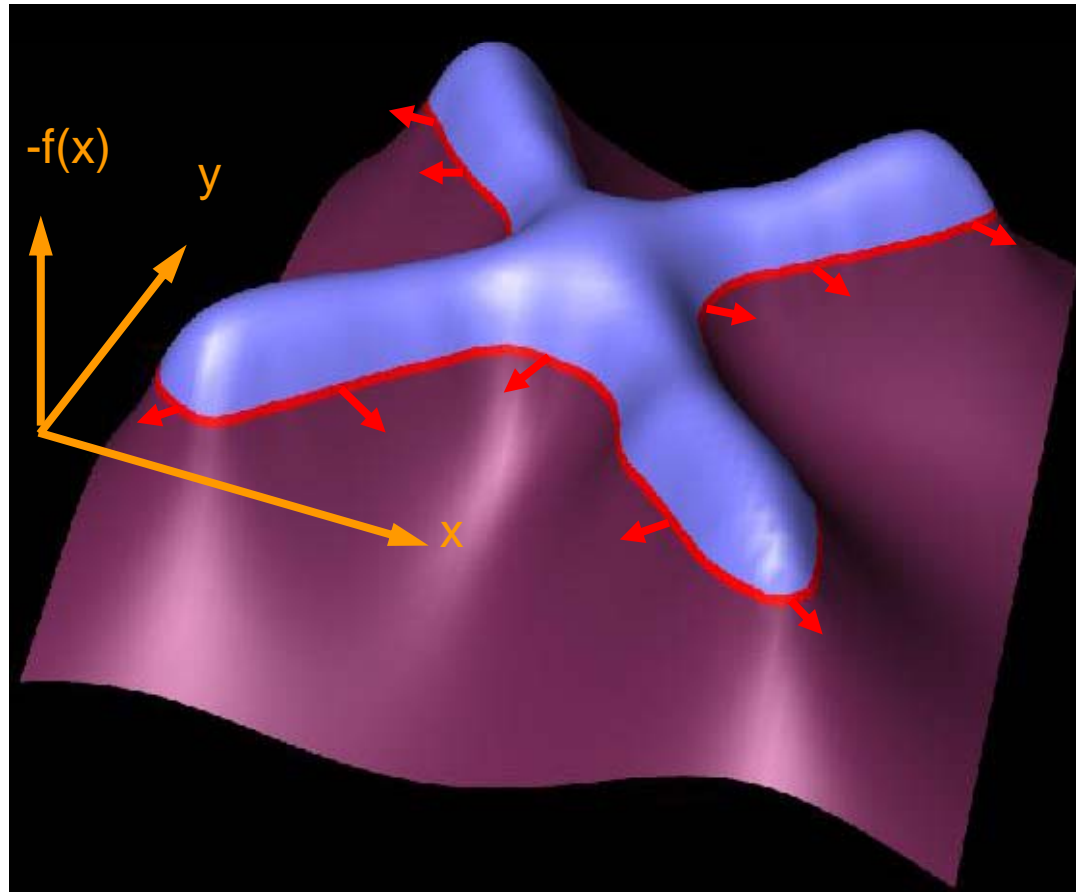
1D example

- Discrete evaluation with marching cubes (3D)



MLS Distance Field

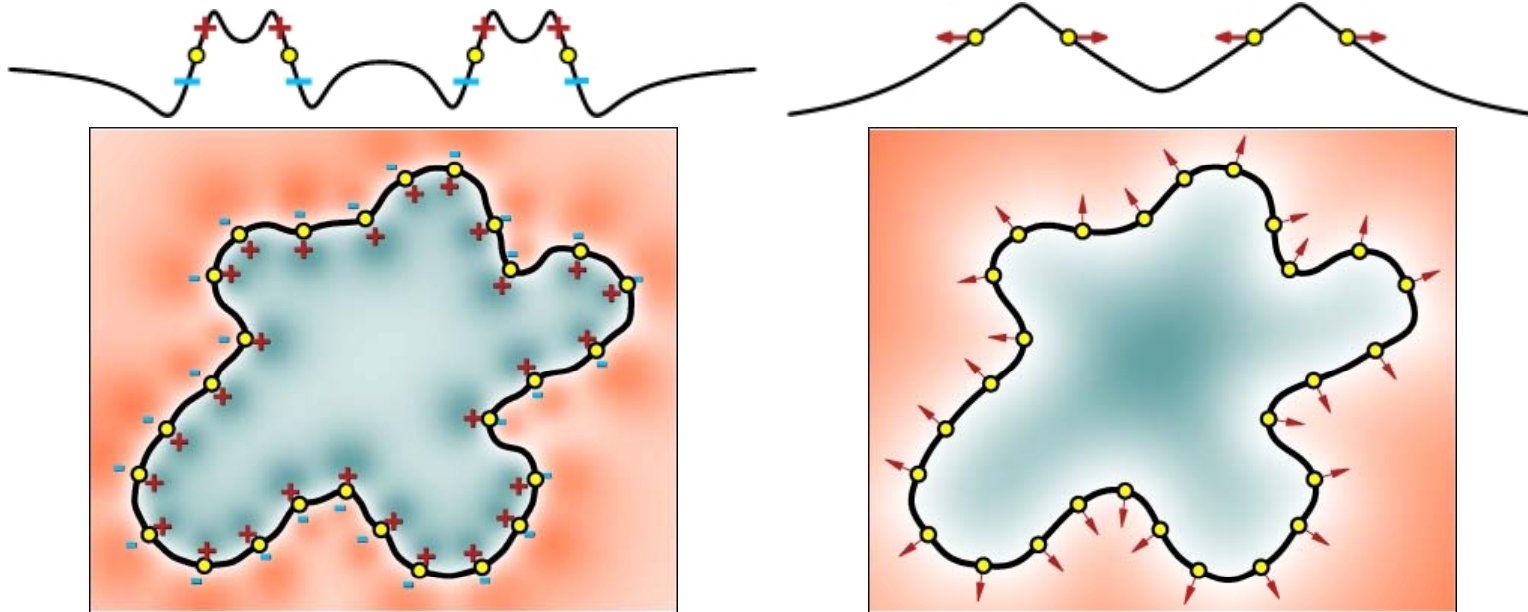
2D Illustration



MLS Distance Field

Extensions

- Point constraints vs. true normal constraints

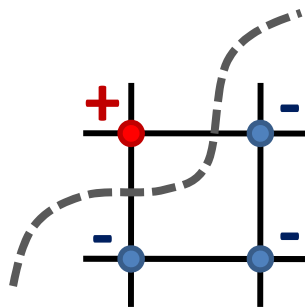


- **Details:** Shen, C., O'Brien, J. F., Shewchuk J. R., "**Interpolating and Approximating Implicit Surfaces from Polygon Soup.**" *Proceedings of ACM SIGGRAPH 2004*, Los Angeles, California, August 8-12.

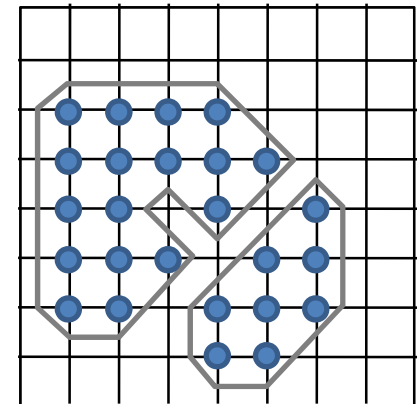
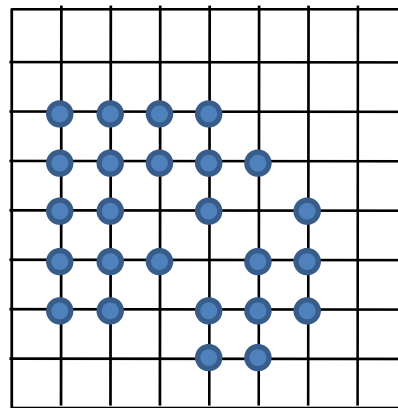
Tessellation of implicit surfaces

Tessellation

- Want to approximate an implicit surface with a mesh
 - For rendering, further processing
- Can't explicitly compute all the roots
 - Infinite amount (the whole surface)
 - The expression of the implicit function may be complicated
- Solution: find approximate roots by trapping the implicit surface in a grid (lattice)



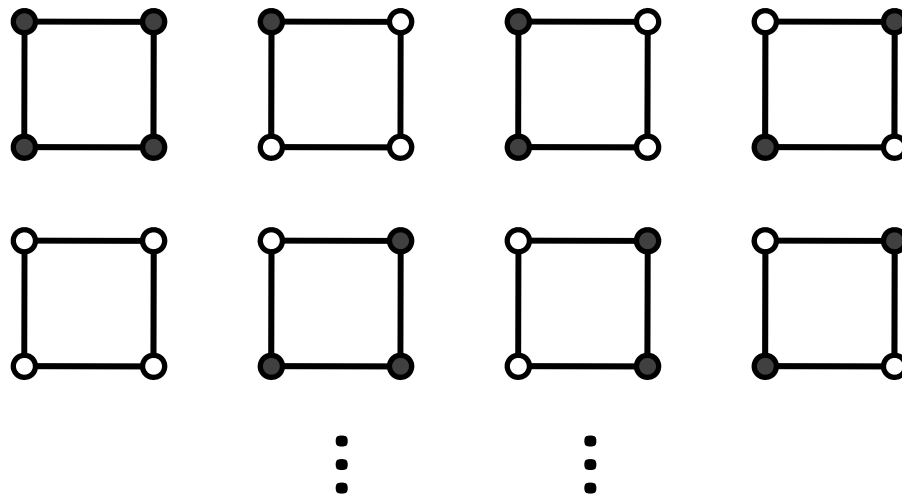
● $f(\mathbf{p}) < 0$



Tessellation

2D grid

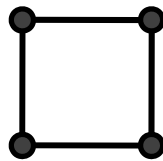
- 16 different configurations in 2D
- 4 equivalence classes (up to rotational and reflection symmetry + complement)



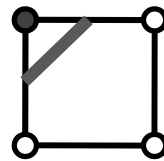
Tessellation

2D grid

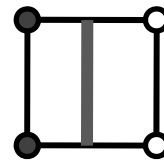
- 16 different configurations in 2D
- 4 equivalence classes (up to rotational and reflection symmetry + complement)



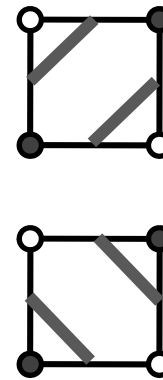
case 1



case 2



case 3

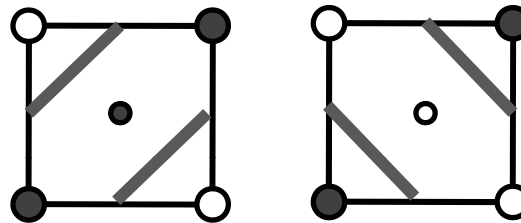


case 4

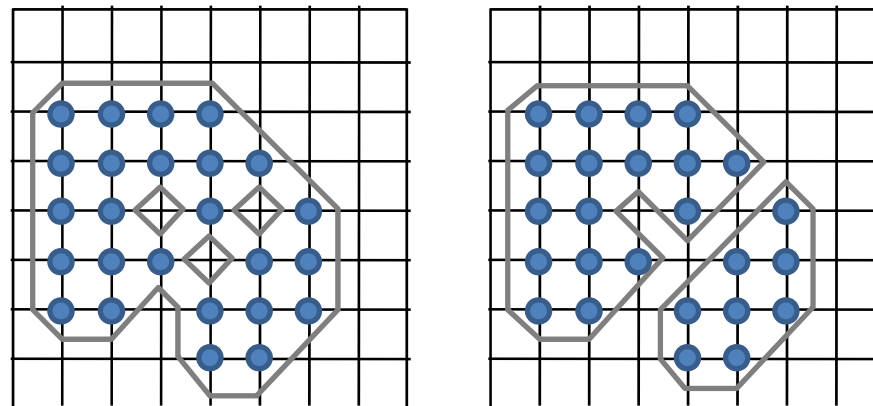
Tessellation

2D grid, consistency

- Case 4 is ambiguous:



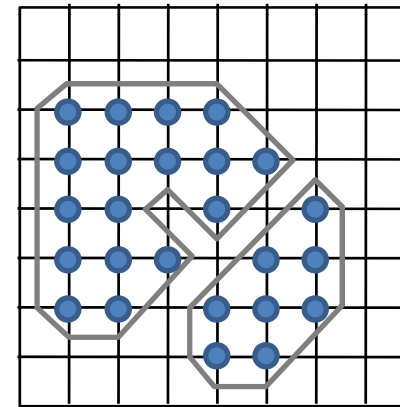
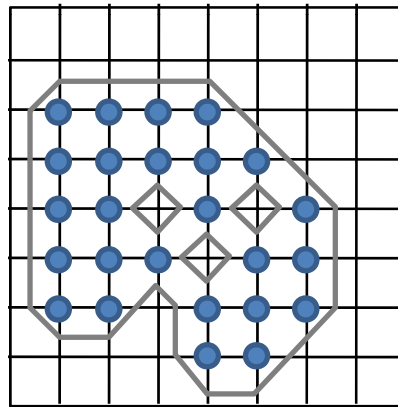
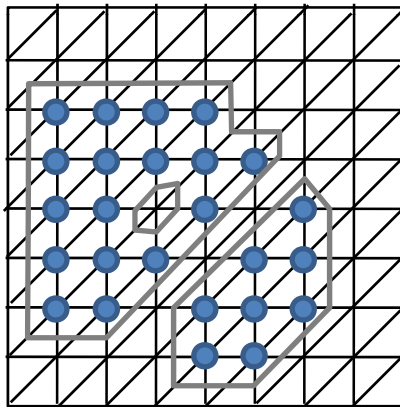
- Always pick consistently to avoid problems with the resulting mesh



Tessellation

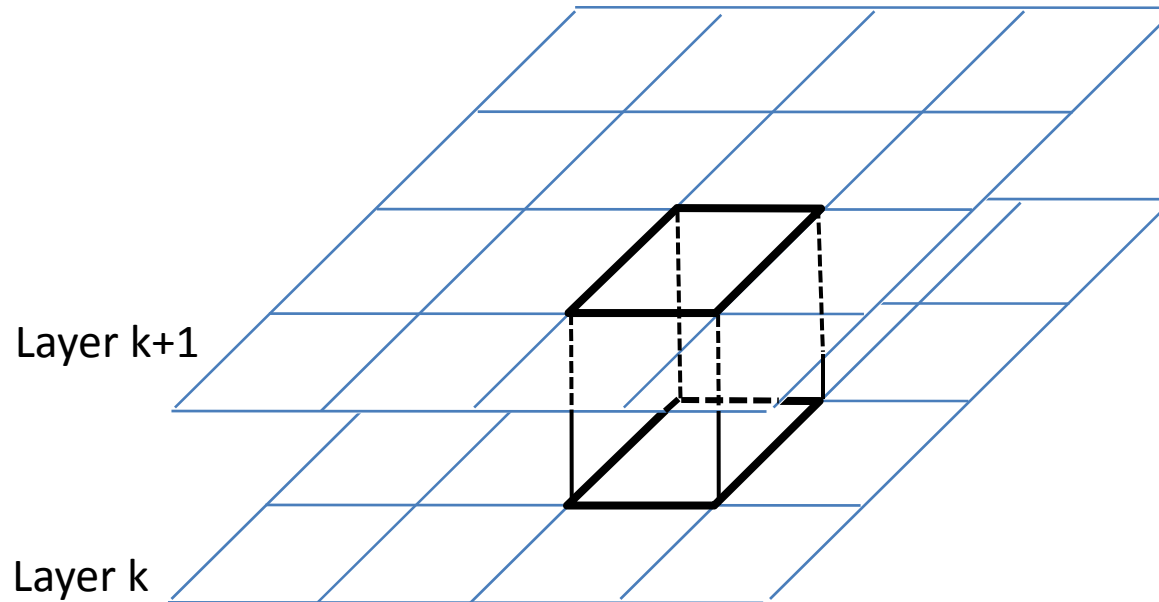
2D triangle grid

- No ambiguity if we have triangles instead of squares
- However, it is still unknown what the true surface is!



Tessellation

3D – Marching Cubes

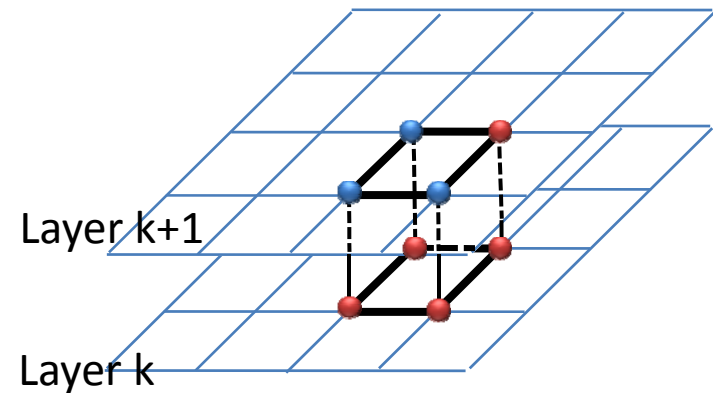


Tessellation

3D – Marching Cubes

- Marching Cubes (Lorensen and Cline 1987)

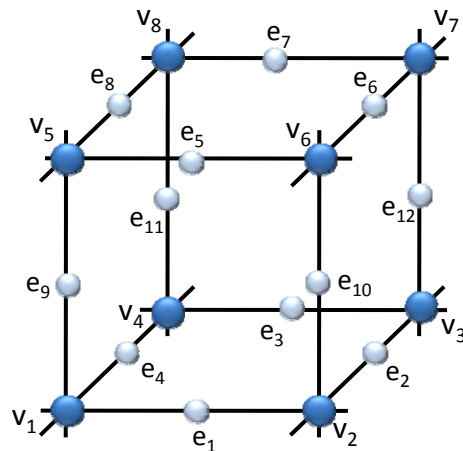
1. Load 4 layers of the grid into memory
2. Create a cube whose vertices lie on the two middle layers
3. Classify the vertices of the cube according to the implicit function (inside, outside or on the surface)



Tessellation

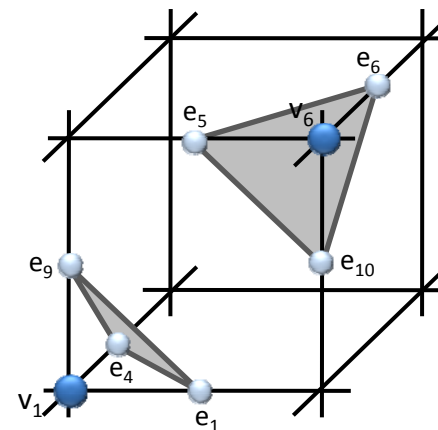
3D – Marching Cubes

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 =

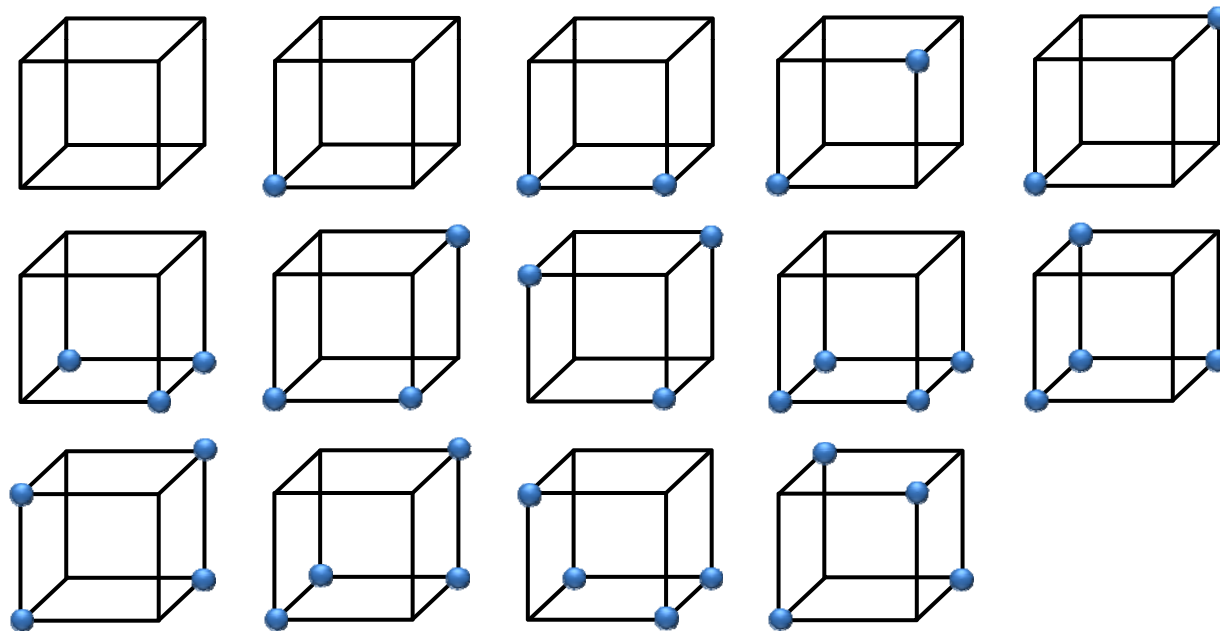
0	0	1	0	0	0	0	1
---	---	---	---	---	---	---	---

 = 33

Tessellation

3D – configurations

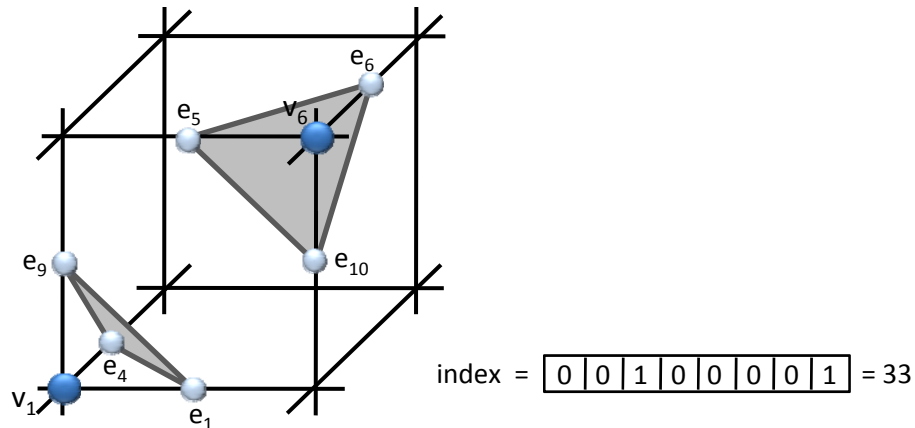
- We have 14 equivalence classes (by rotation, reflection and complement)



Tessellation

3D – Marching Cubes

- 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 $(e_5; e_{10}; e_6)$.



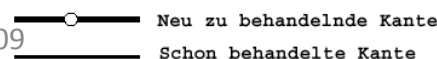
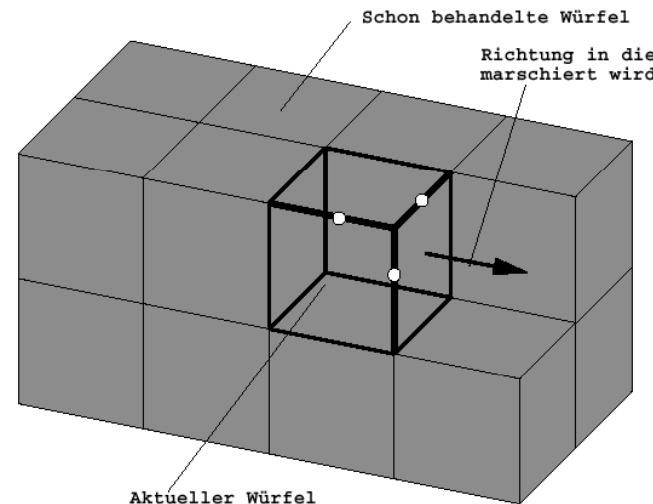
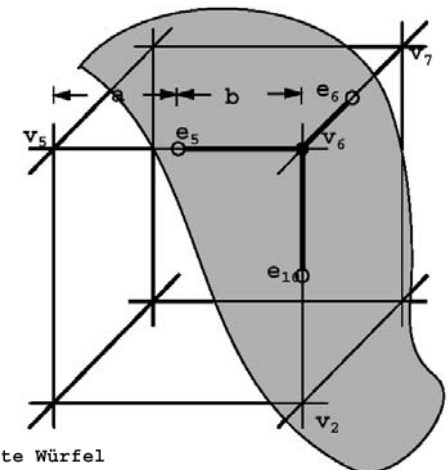
Tessellation

3D – Marching Cubes

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

$$\mathbf{v}_s = \alpha \mathbf{v}_a + (1 - \alpha) \mathbf{v}_b$$
$$\alpha = \frac{f(\mathbf{v}_b)}{f(\mathbf{v}_b) - f(\mathbf{v}_a)}$$

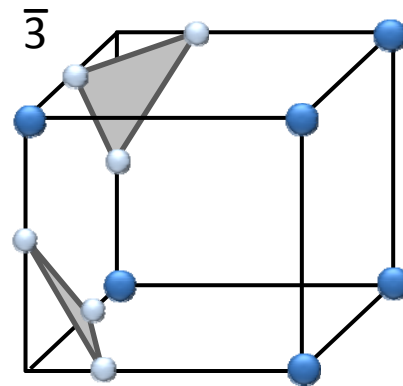
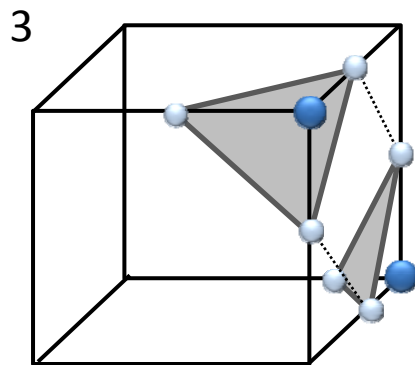
7. Compute the vertex normals
8. Move to the next cube



Tessellation

3D – configurations, consistency

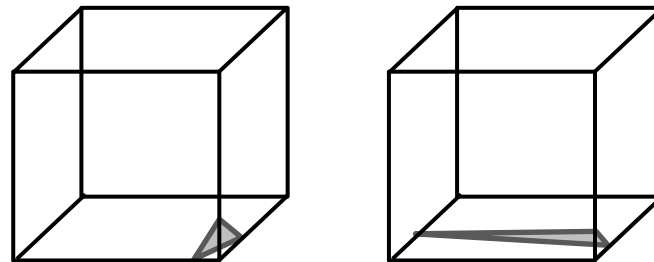
- Have to make consistent choices for neighboring cubes
- Prevent “holes” in the triangulation



Tessellation

Grid-Snapping

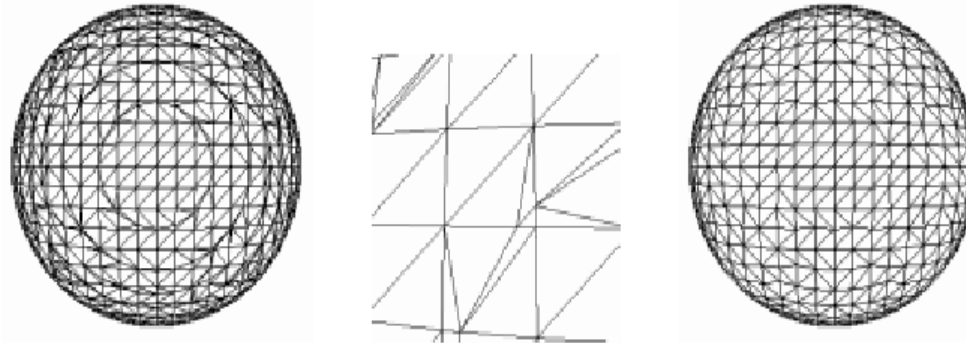
- Problems with short triangle edges
 - When the surface intersects the cube close to a corner, the resulting tiny triangle doesn't contribute much area to the mesh
 - When the intersection is close to an edge of the cube, we get skinny triangles (bad aspect ratio)
- Triangles with short edges waste resources but don't contribute to the surface mesh representation



Tessellation

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



Tessellation

Grid-Snapping

- With Grid-Snapping one can obtain significant reduction of space consumption

Parameter	0	0,1	0,2	0,3	0,4	0,46	0,49 5
Vertices	1446	1398	1254	1182	1074	830	830
Reduction	0	3,3	13,3	18,3	25,7	42,6	42,6

Tessellation

Sharp corners and sharp edges

- (Kobbelt et al. 2001):
 - Evaluate the normals
 - When they significantly differ, create additional vertex

