

3D Scanning & Motion Capture

Surface Representations

Dr. Justus Thies, Dr. Angela Dai

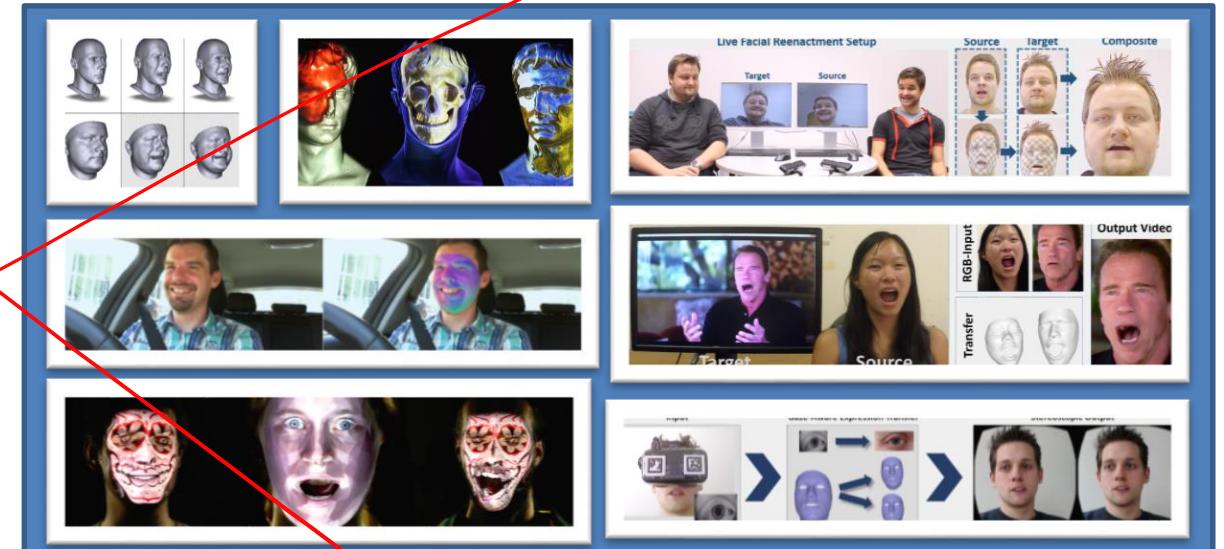


Visual Computing @ TUM

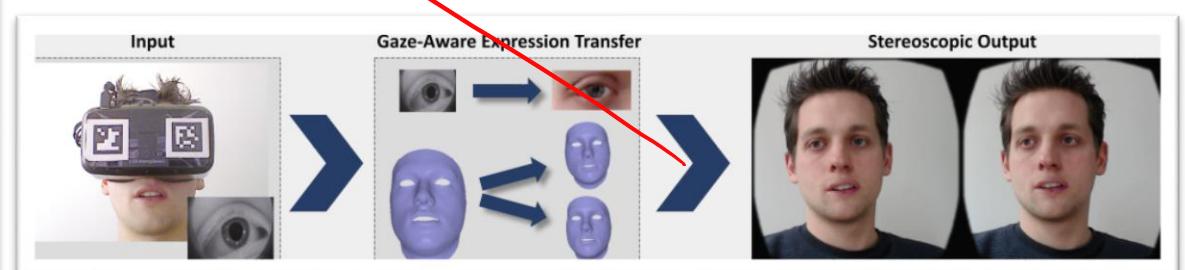
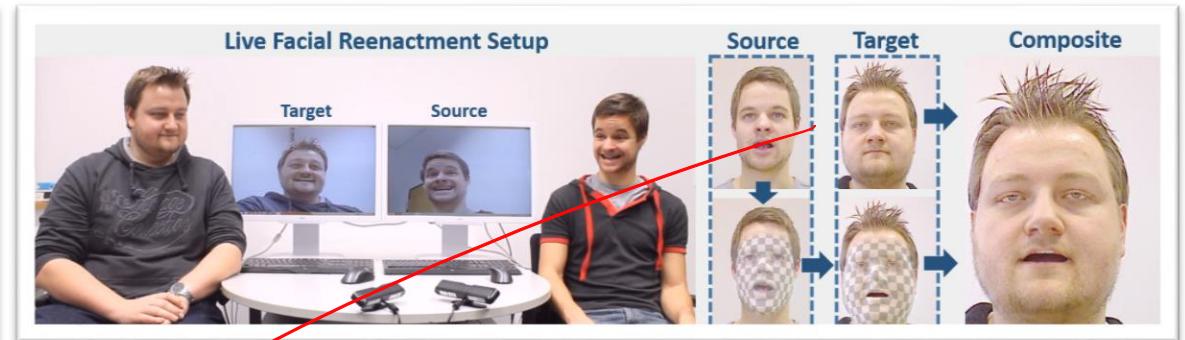
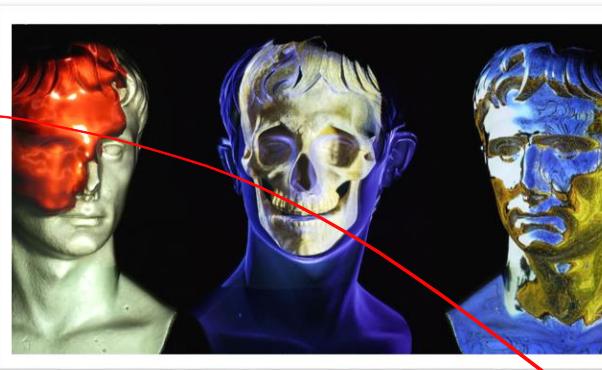
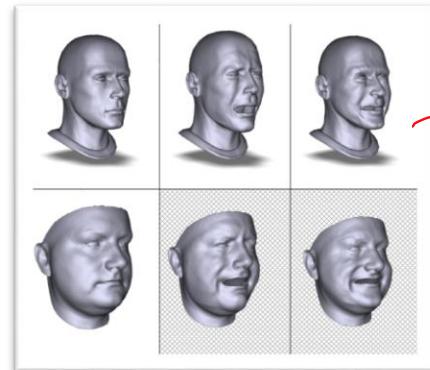
Scene Reconstruction & Understanding



Human Face & Body Tracking



Research Projects



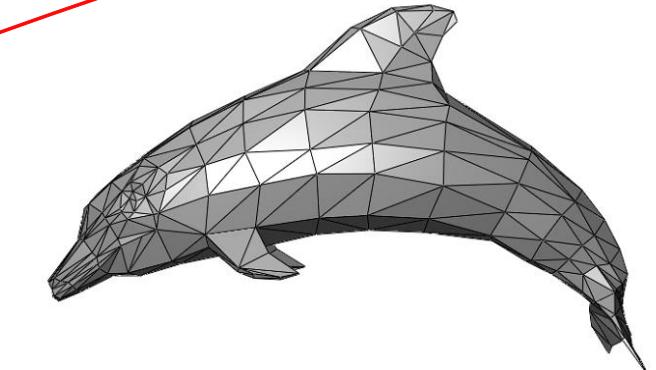
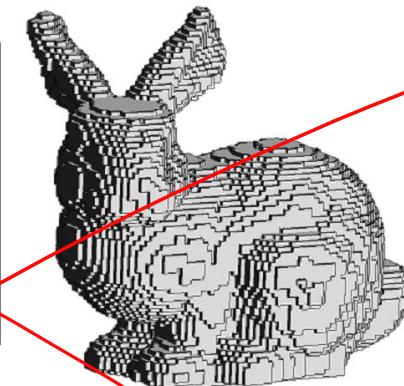
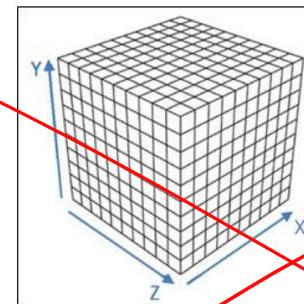
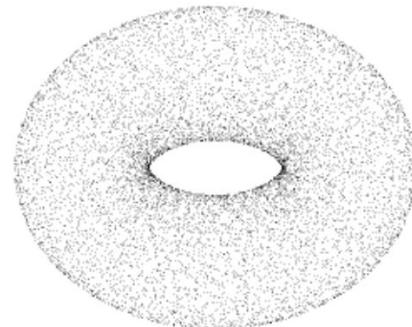
Research Projects



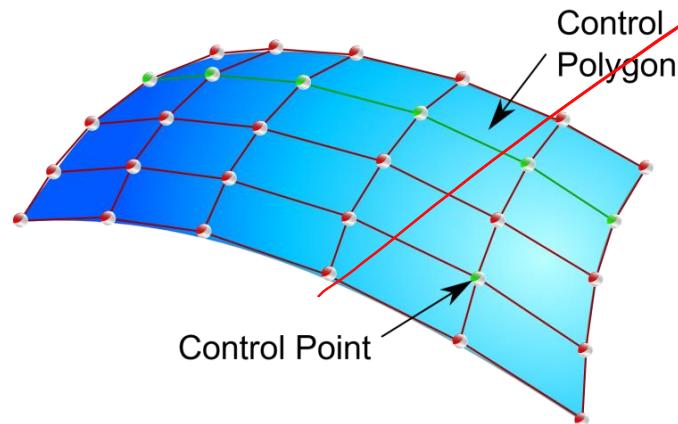
3D Scanning & Motion Capture
Thies, Dai

Last Lecture: What is “3D”?

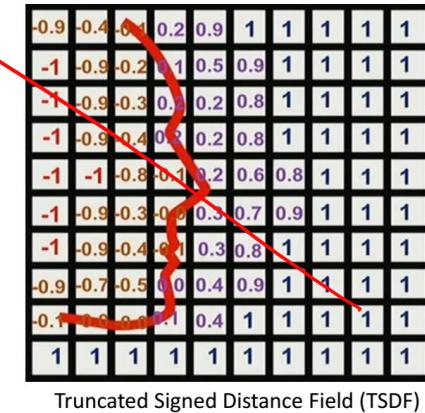
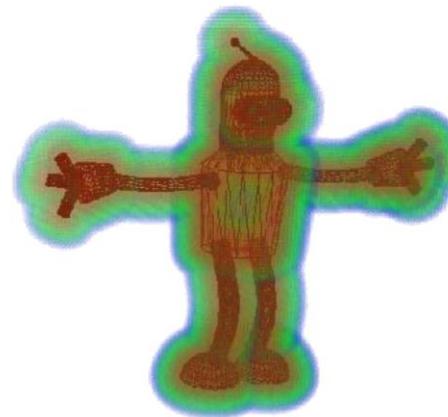
- Point Clouds
- Voxels
- Polygonal Meshes



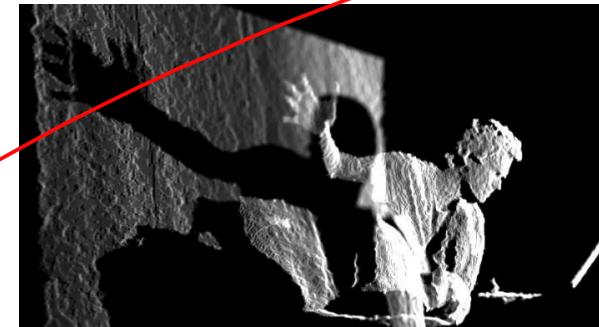
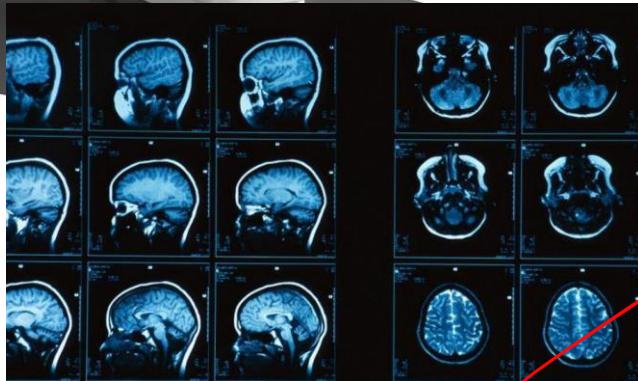
- Parametric Surfaces

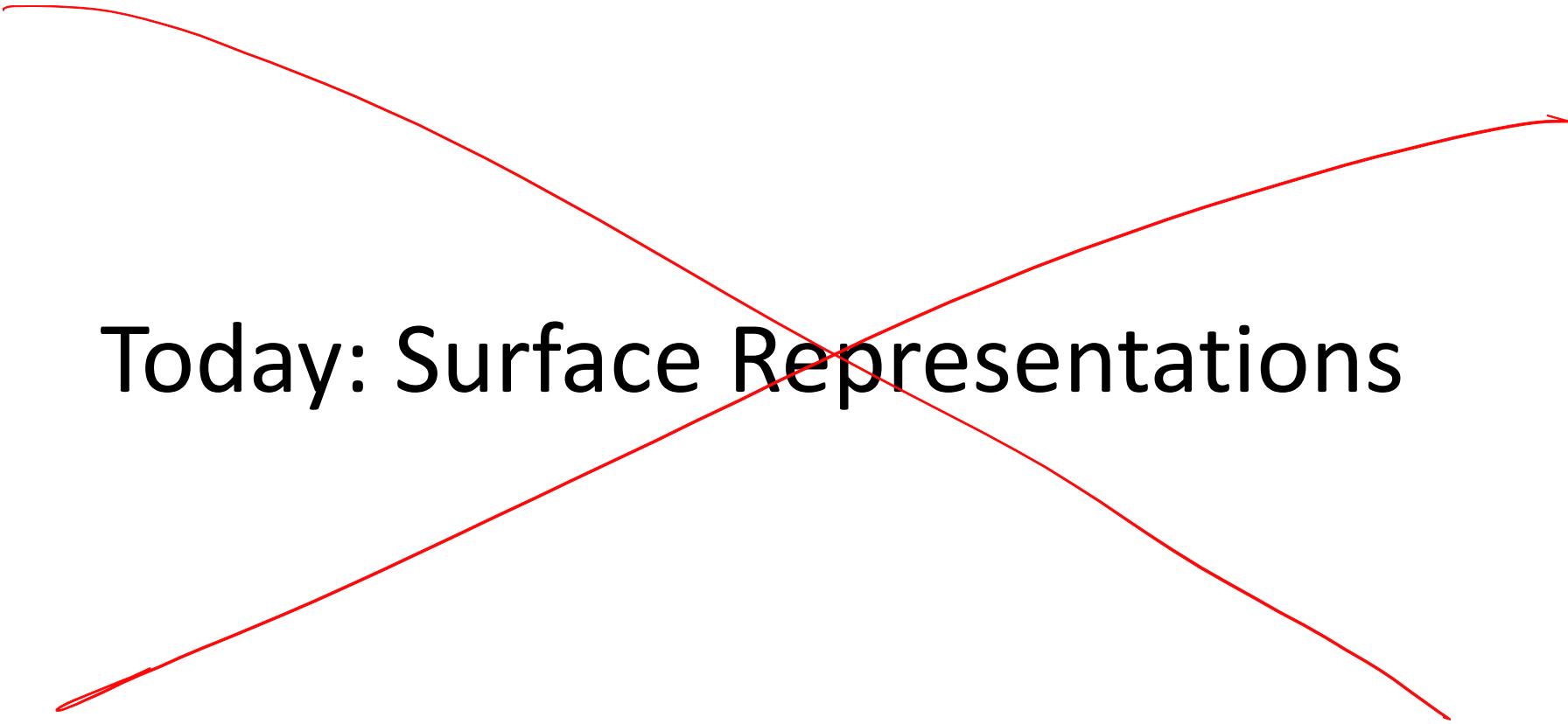


- Implicit Surfaces



Last Lecture: How to obtain “3D”?

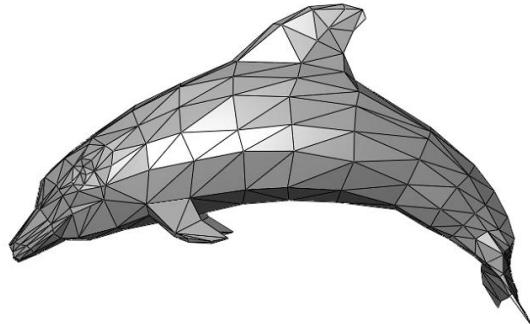




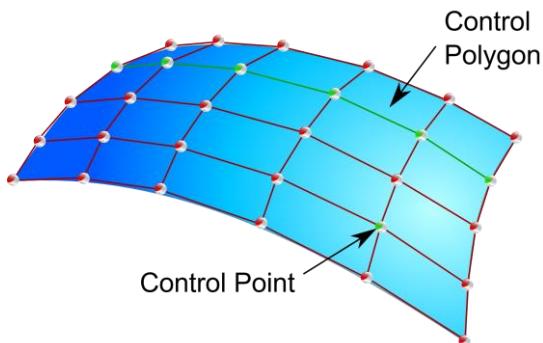
Today: Surface Representations

Overview

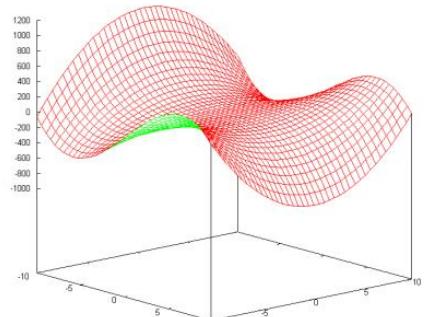
- Polygonal Meshes



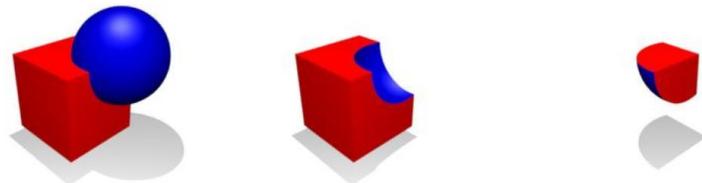
- Parametric Surfaces



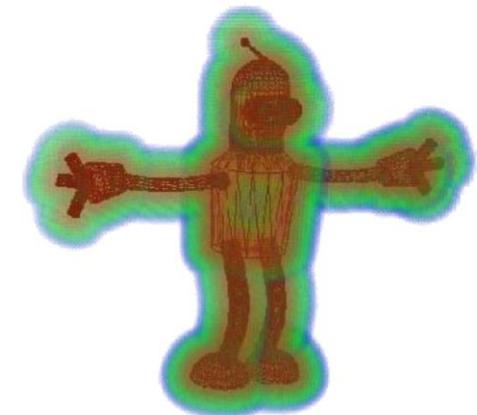
- Explicit Surfaces



- Constructive Solid Geometry

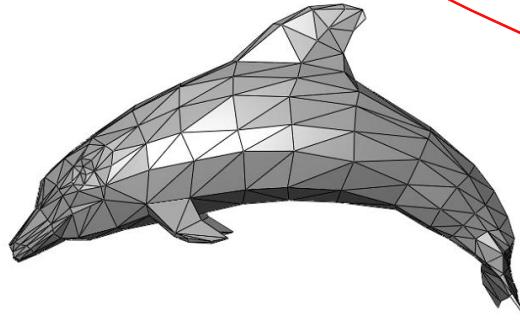


- Implicit Surfaces

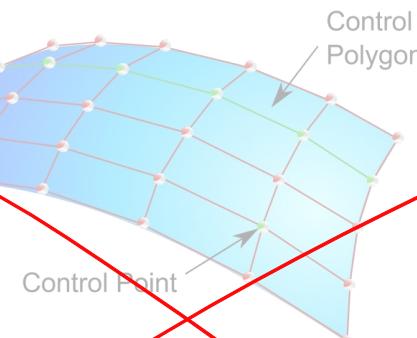


Overview

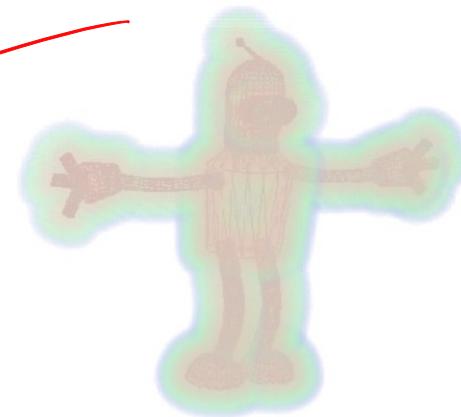
- Polygonal Meshes



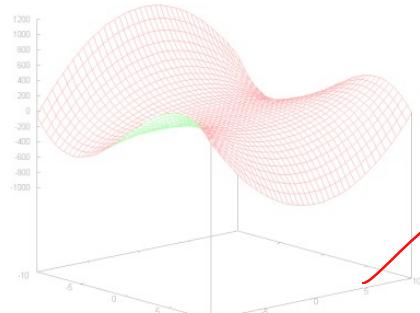
- Parametric Surfaces



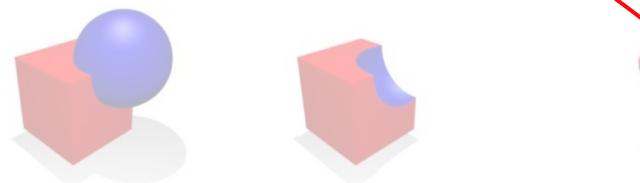
- Implicit Surfaces



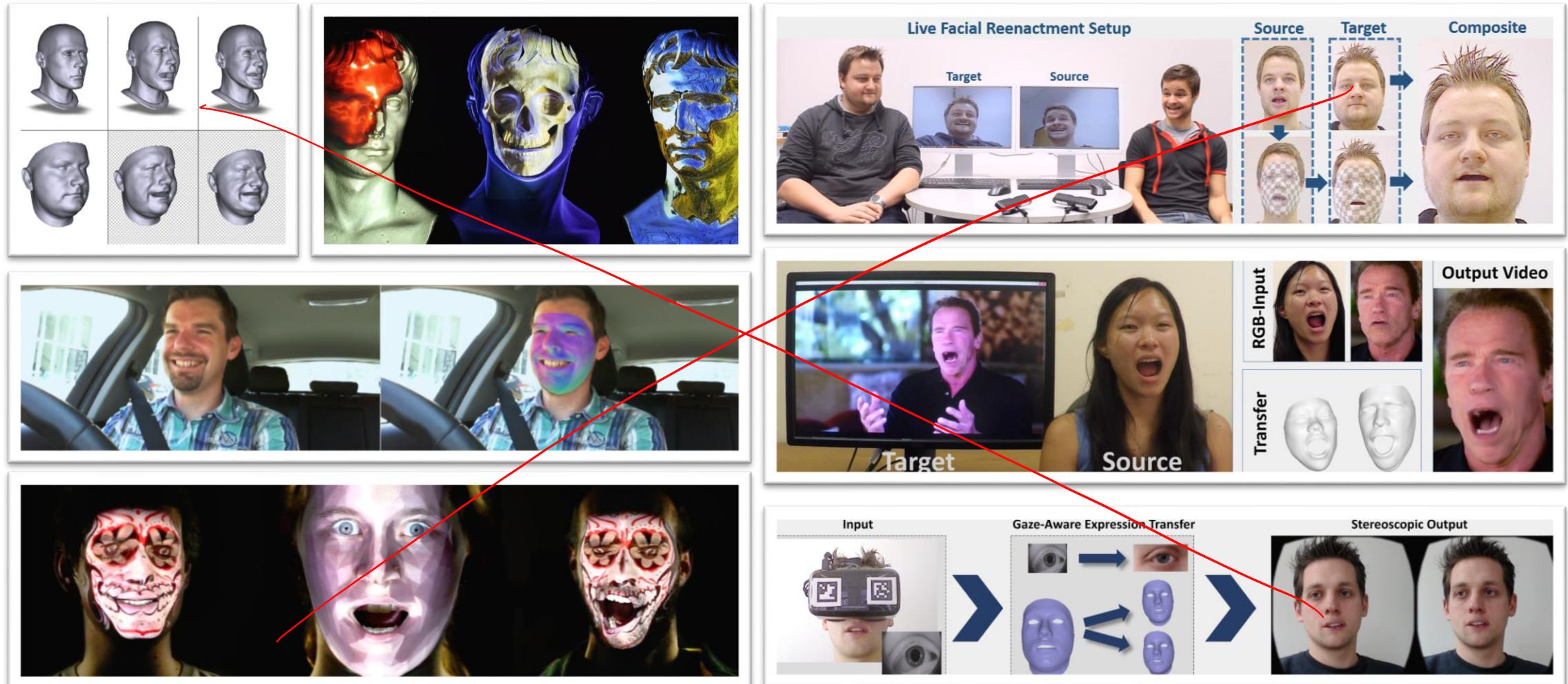
- Explicit Surfaces



- Constructive Solid Geometry

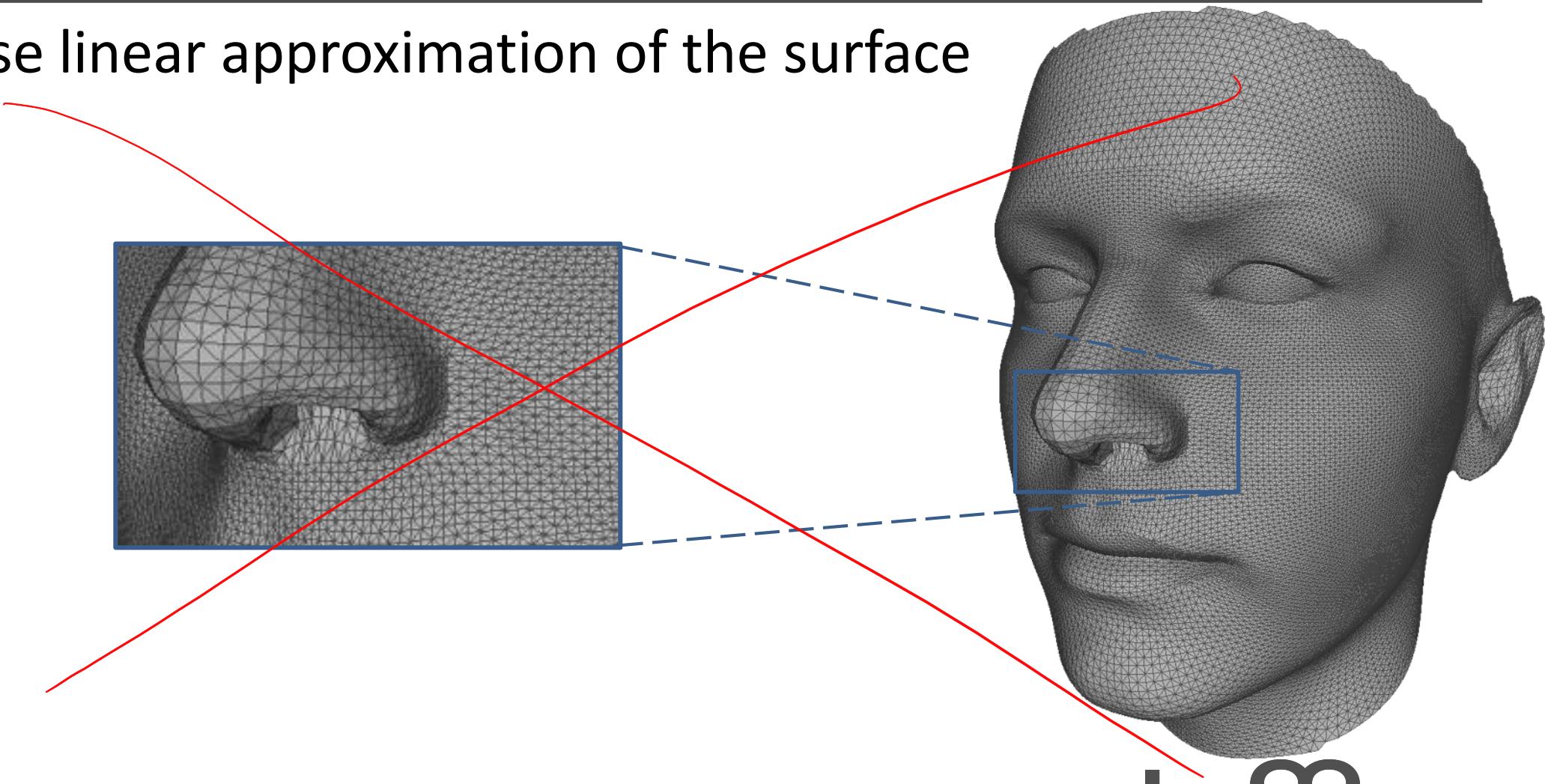


Polygonal Meshes



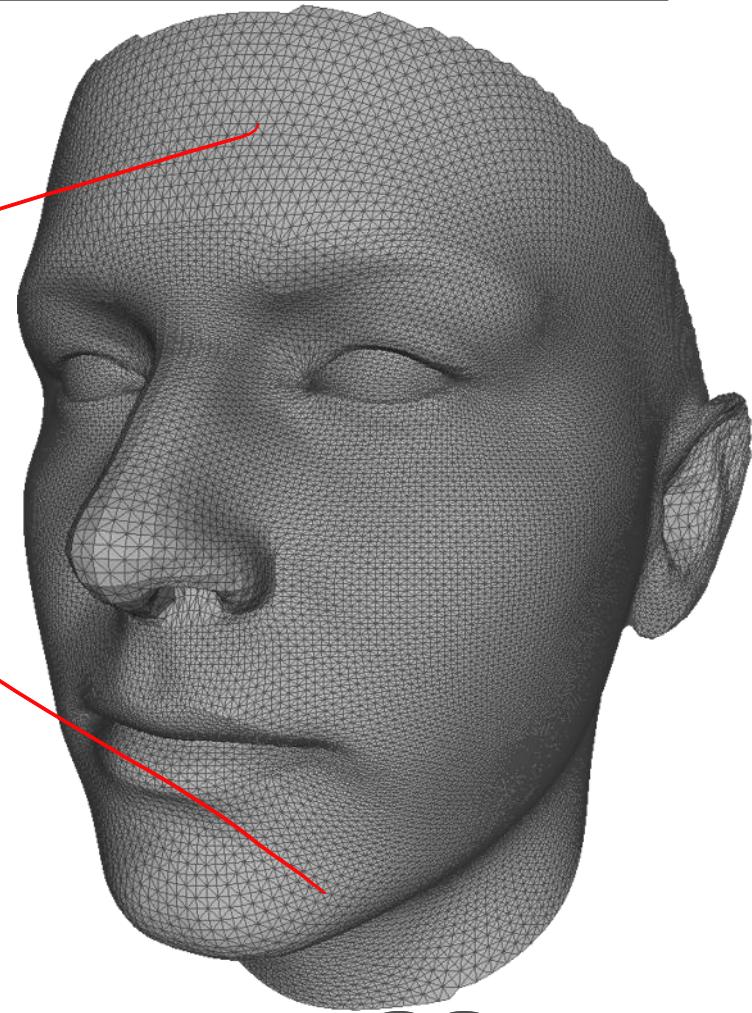
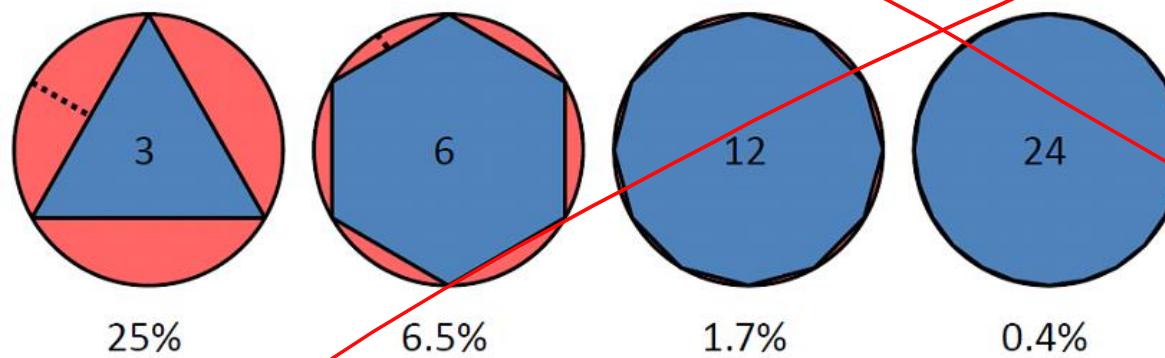
Polygonal Meshes

- Piecewise linear approximation of the surface



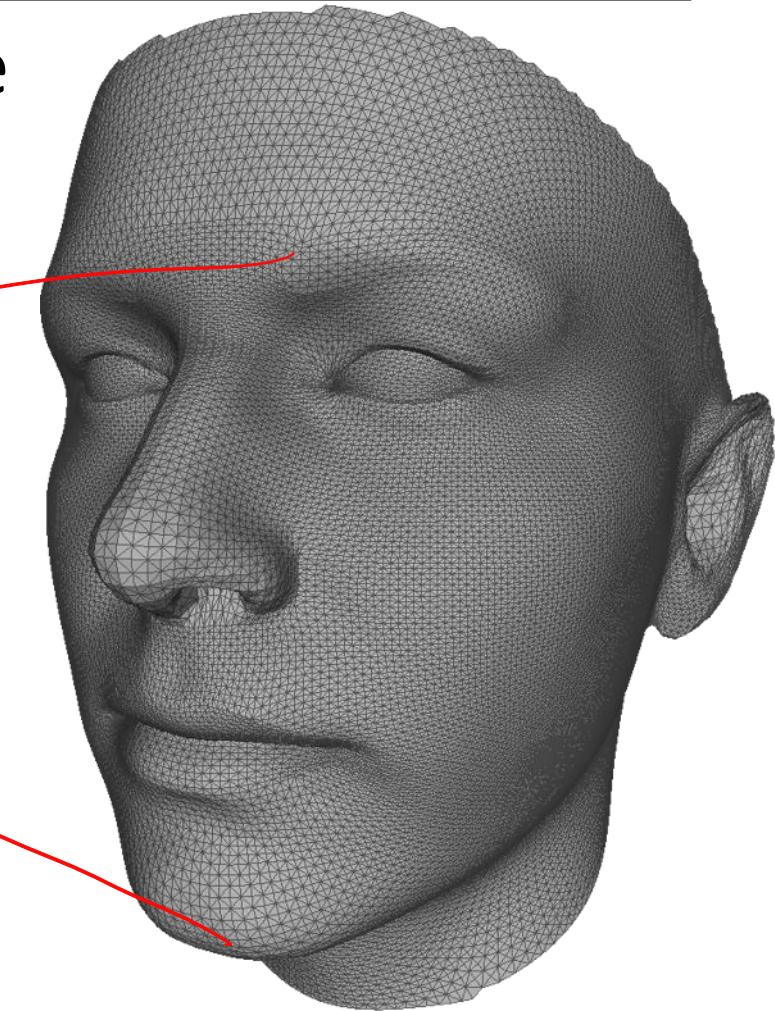
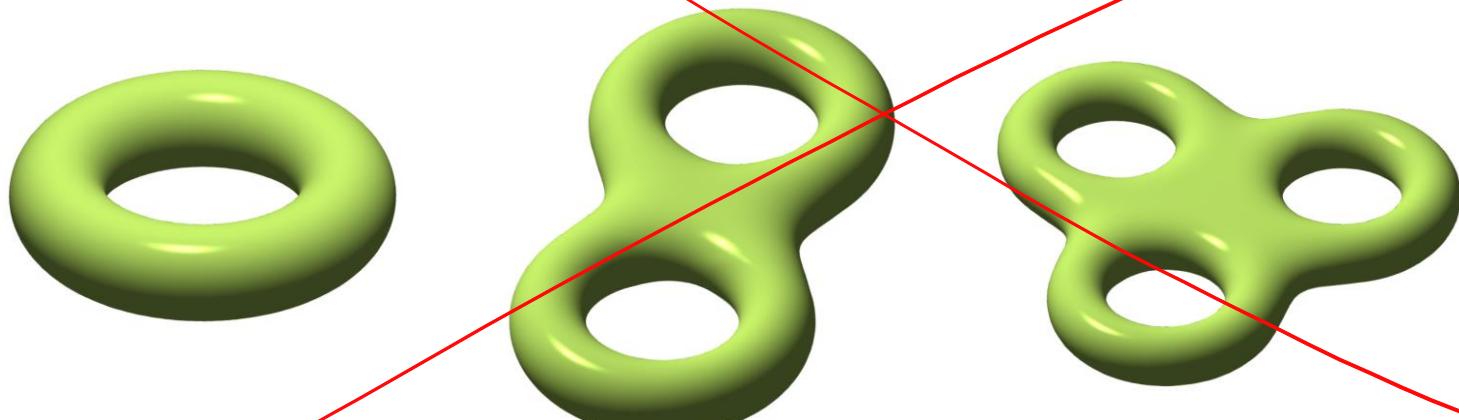
Polygonal Meshes

- Piecewise linear approximation of the surface
 - Error $O(h^2)$



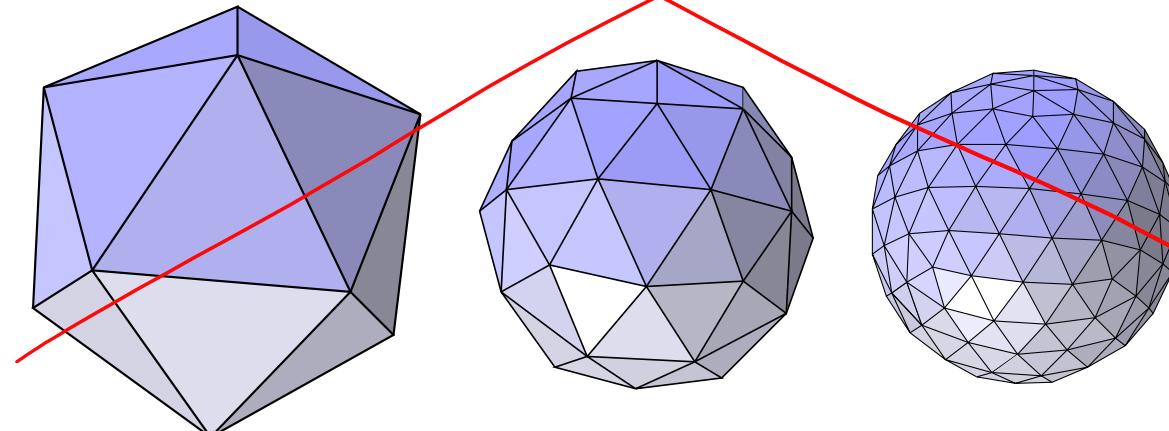
Polygonal Meshes

- Piecewise linear approximation of the surface
 - Error $O(h^2)$
 - Arbitrary topology

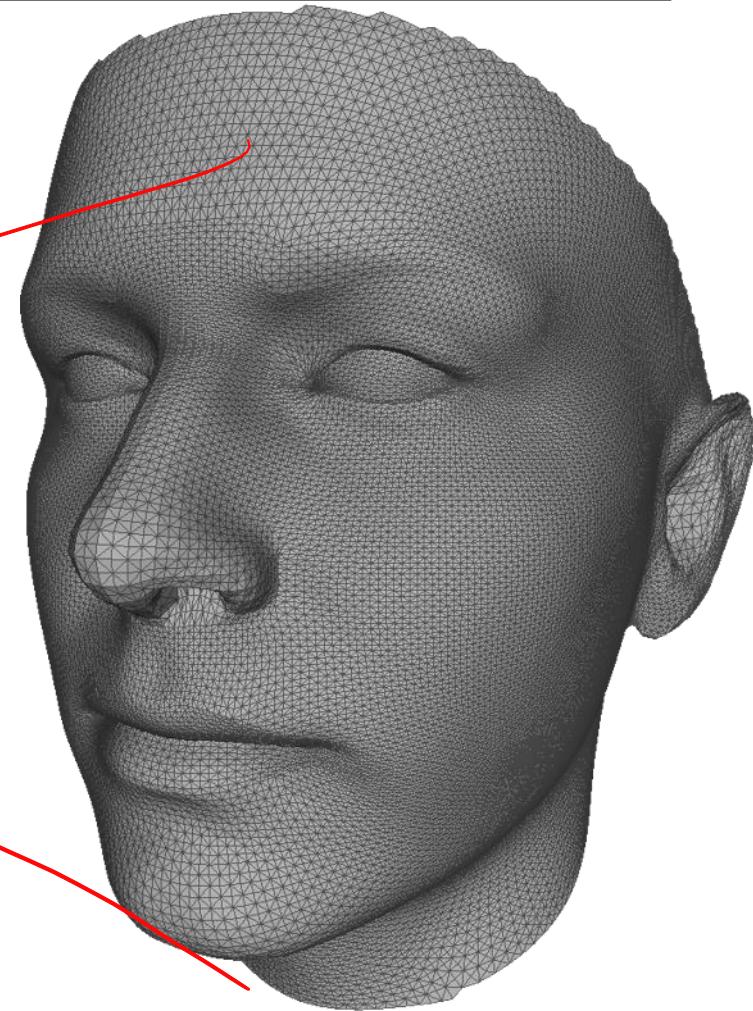


Polygonal Meshes

- Piecewise linear approximation of the surface
 - Error $O(h^2)$
 - Arbitrary topology
 - Adaptive refinement (subdivision)

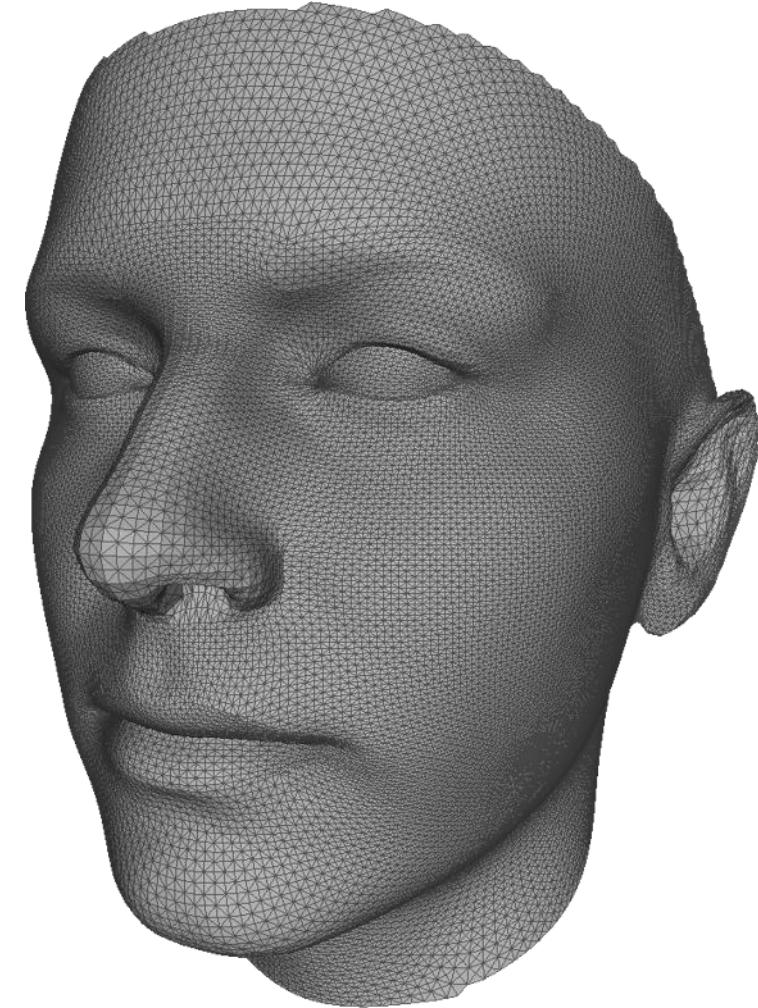


Loop Subdivision Surface



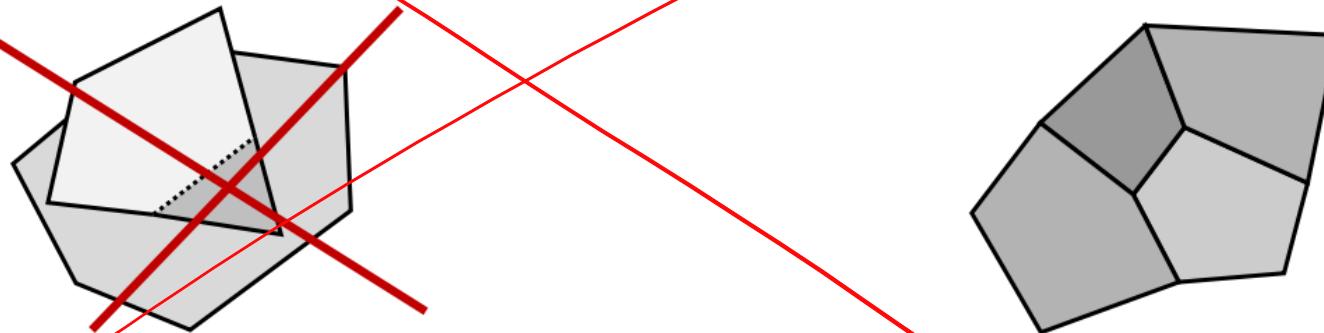
Polygonal Meshes

- Piecewise linear approximation of the surface
 - Error $O(h^2)$
 - Arbitrary topology
 - Adaptive refinement (subdivision)
 - Efficient rendering



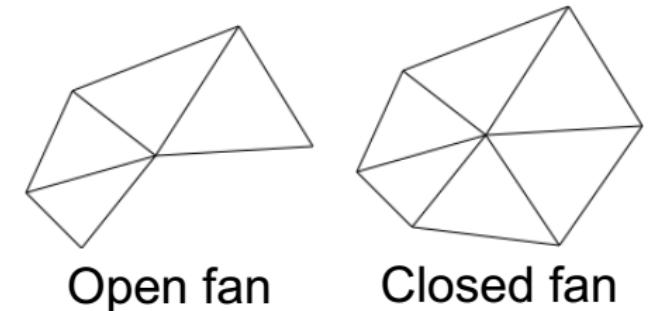
Polygonal Meshes

- A polygonal mesh is a collection of polygons satisfying certain restrictions



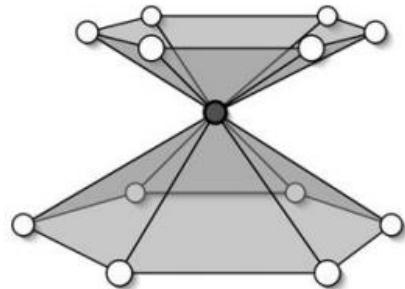
Polygonal Meshes

- A triangle mesh is called **manifold** if:
 - The intersection of two triangles is:
 - Empty
 - A common vertex
 - A common edge
 - Edges have
 - One adjacent triangle → border edge
 - Two adjacent triangles → inner edge
 - For a vertex the adjacent triangles
 - Build a single open fan → border vertex
 - Build a single closed fan → inner vertex

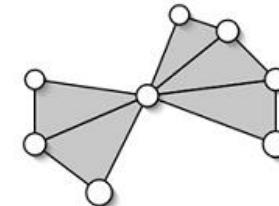


Polygonal Meshes

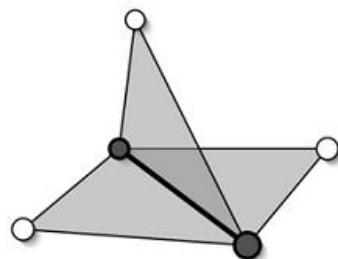
- Example of **non-manifold** meshes:



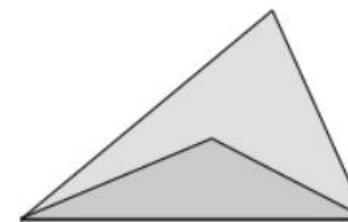
Two closed fans
at one vertex



Two open fans
at one vertex



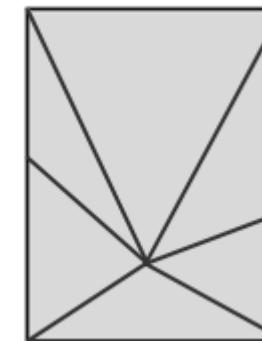
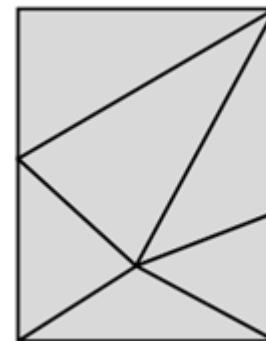
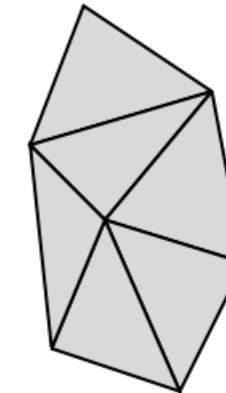
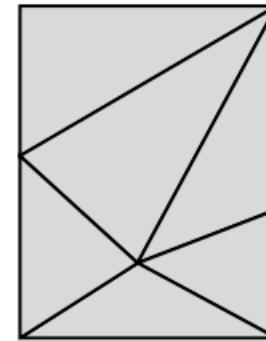
Three triangles
At one edge



Intersection of triangles
Is an entire triangle

Polygonal Meshes

- Topology vs. Geometry
 - Topologically equivalent:
 - But different geometry
 - Geometrically equivalent:
 - But different topology



Polygonal Meshes

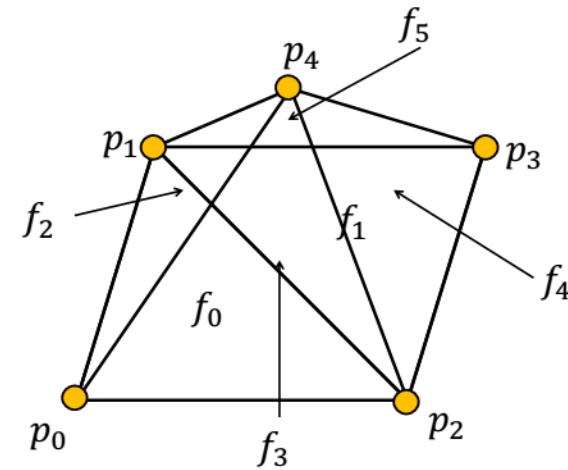
- Shared Vertex Data Structure
 - De facto standard for file formats (e.g., OFF, OBJ, STL, PLY, ...)

vertex list

```
p0: x0, y0, z0;  
p1: x1, y1, z1;  
p2: x2, y2, z2;  
p3: x3, y3, z3;  
p4: x4, y4, z4;
```

face list

```
f0: 0, 1, 2;  
f1: 2, 1, 3;  
f2: 1, 0, 4;  
f3: 4, 0, 2;  
f4: 4, 2, 3;  
f5: 4, 3, 1;
```



Polygonal Meshes

- Half-Edge Data Structure

- Easy geometric queries → widely used in geometric computations
- Only manifolds $\stackrel{?}{=}$ Linked list of edges

$v_{end}, v_{end} + e_{opp}/$

$v_{end} + e_{next}$

\Rightarrow Vertices
 \Rightarrow Normals...

Klausur!

Edge

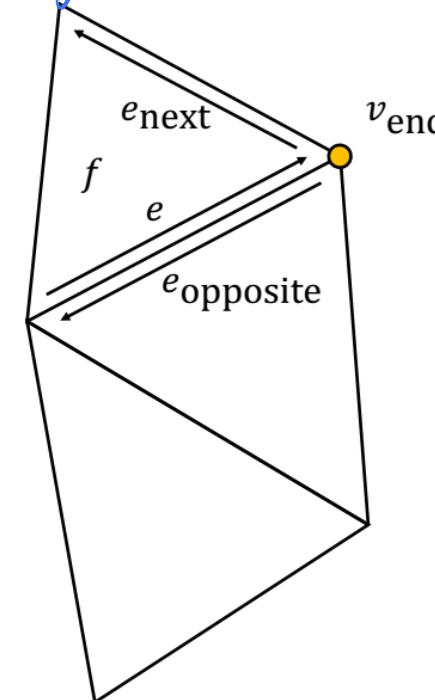
- Edge ID e
- End vertex v_{end}
- Next edge e_{next}
- Opposite edge $e_{opposite}$
- Face f

Vertex

- Vertex ID
- Index of one incident edge

Face

- Face ID
- Index of one incident edge



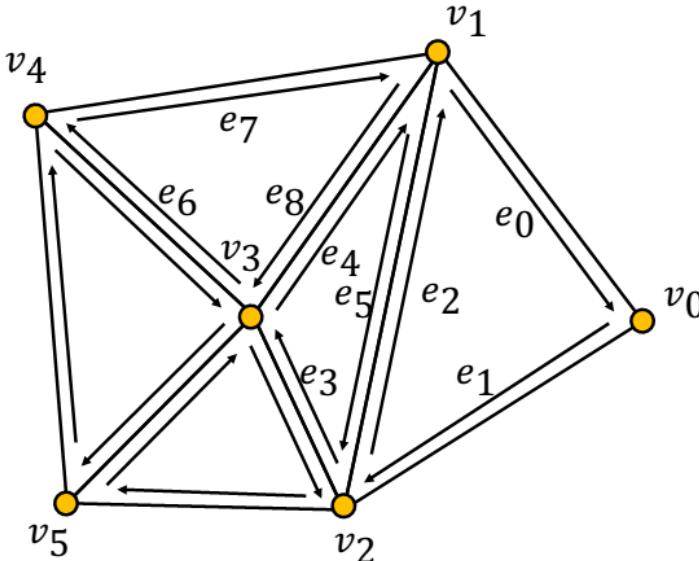
OpenMesh

Polygonal Meshes

- Half-Edge Data Structure
 - Easy geometric queries → widely used in geometric computations
 - Only manifolds
 - Triangular Meshes → Directed Edge Data Structure

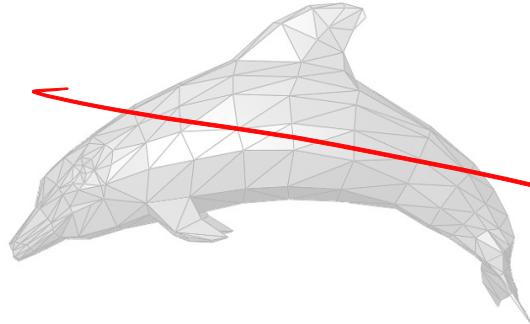
vertex list
p₀: x₀, y₀, z₀;
p₁: x₁, y₁, z₁;
p₂: x₂, y₂, z₂;
p₃: x₃, y₃, z₃;
p₄: x₄, y₄, z₄;
p₅: x₅, y₅, z₅;

edge list
e₀: 1, -1;
e₁: 0, -1; } t₀
e₂: 2, 5;
e₃: 2, X; } t₁
e₄: 3, 8;
e₅: 1, 2;
e₆: 3, X;
e₇: 4, -1;
...

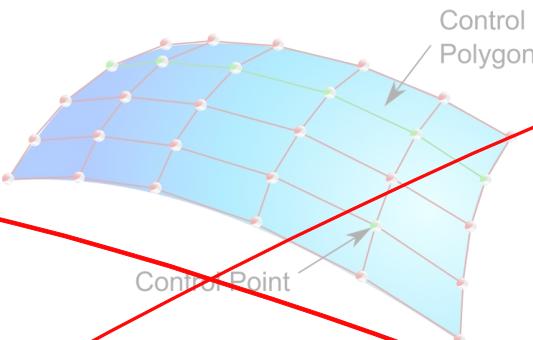


Overview

- Polygonal Meshes



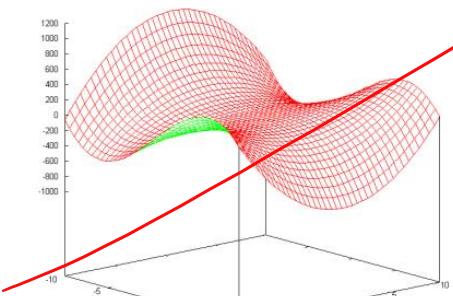
- Parametric Surfaces



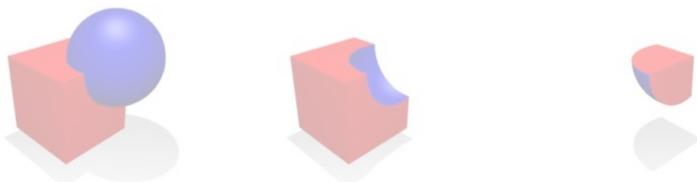
- Implicit Surfaces



- Explicit Surfaces

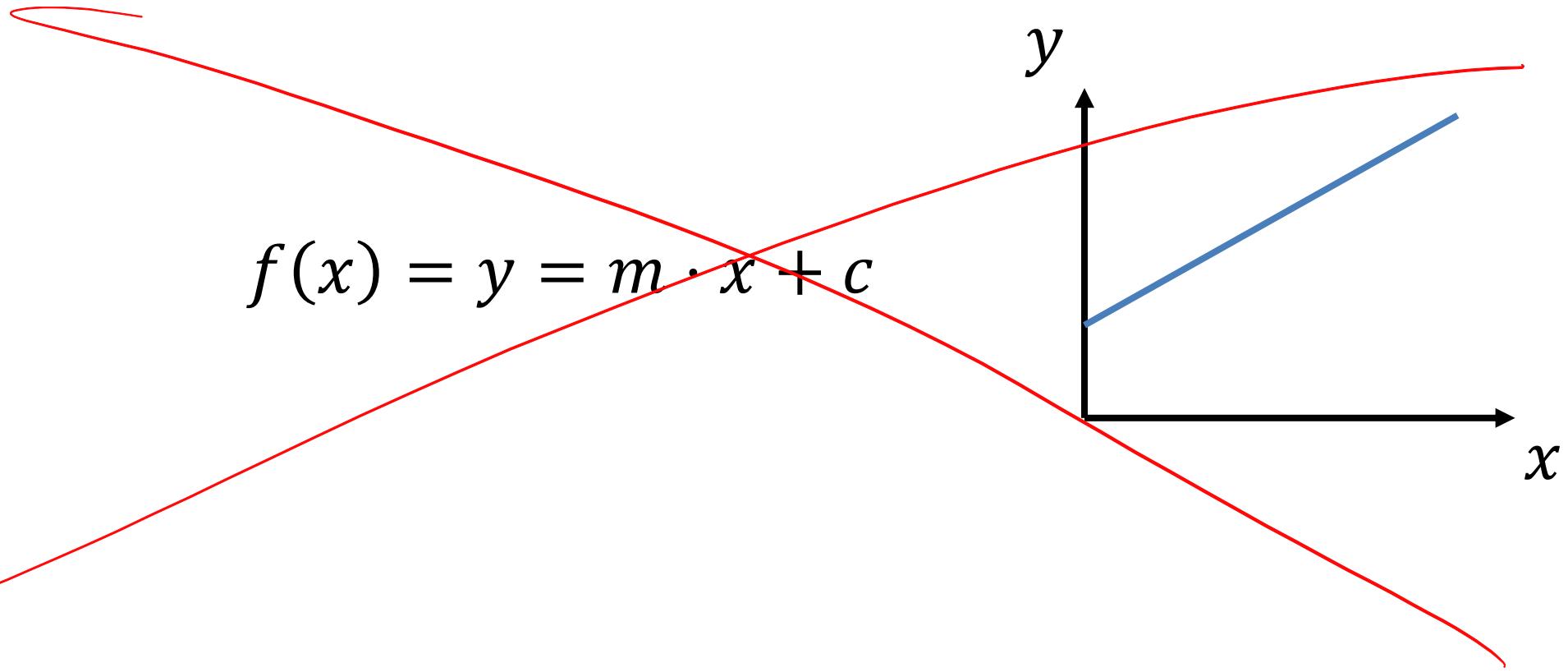


- Constructive Solid Geometry



Explicit Surfaces

- Explicit form:



Explicit Surfaces

- Explicit form:

$$f(x, y): \mathbb{R}^2 \rightarrow \mathbb{R}$$

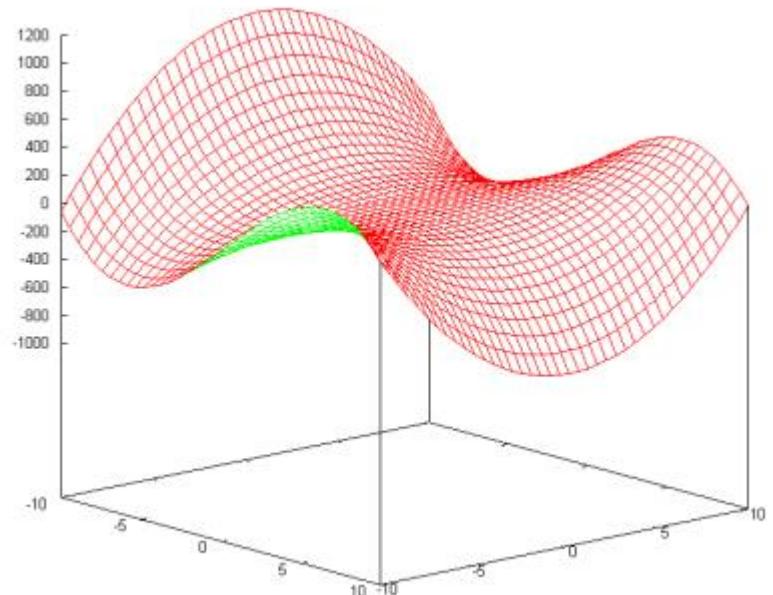
- Surface is defined by:

$$S(x, y) = (x, y, f(x, y))$$

- Disadvantages

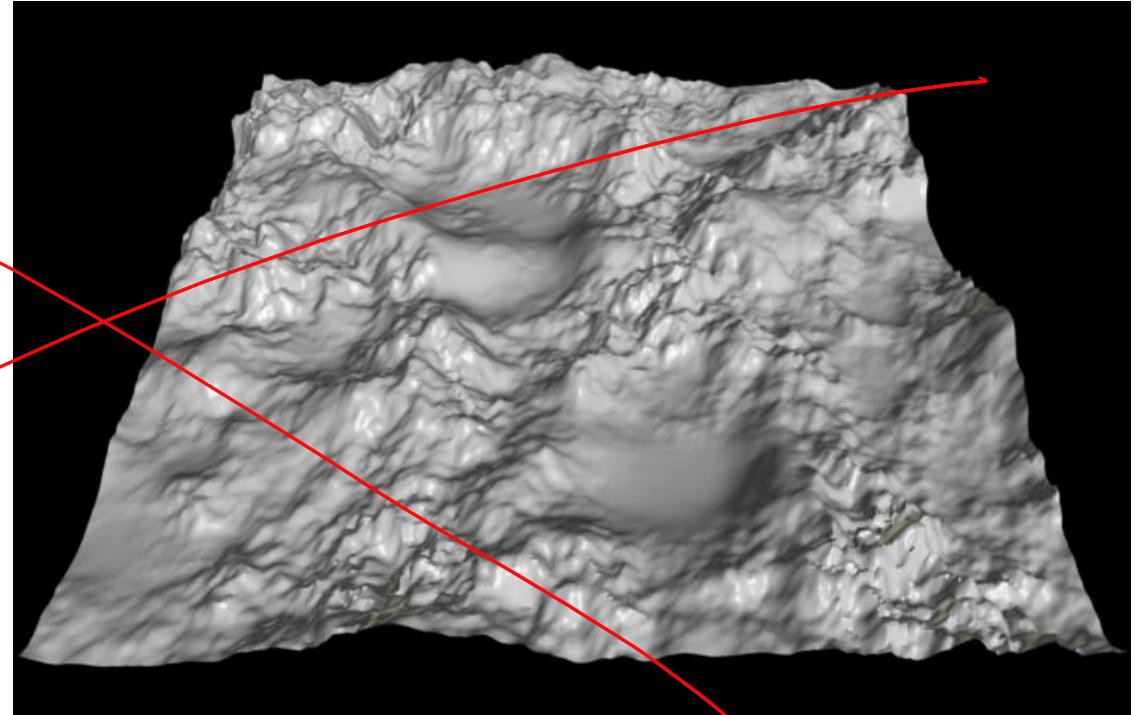
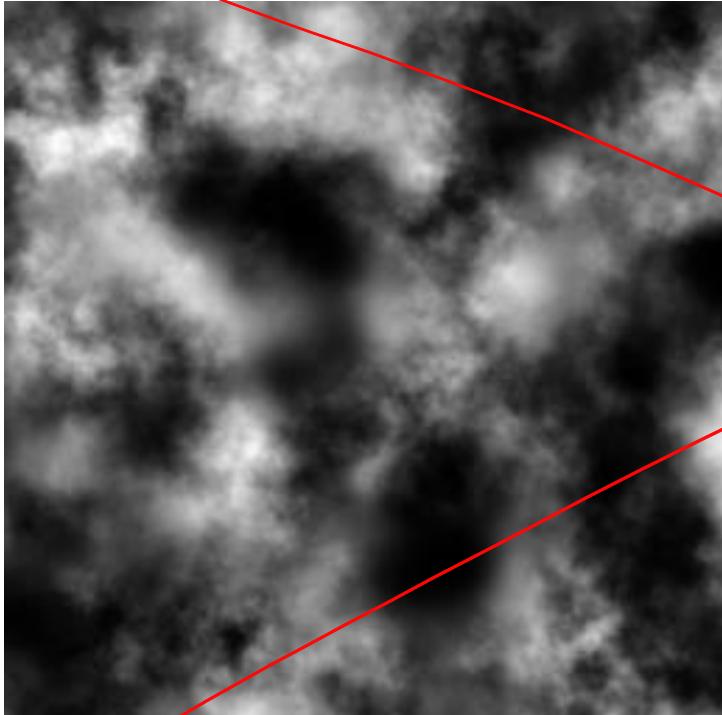
- Restricted shapes, e.g. only one height value per (x,y) pair

E.g. Height Fields



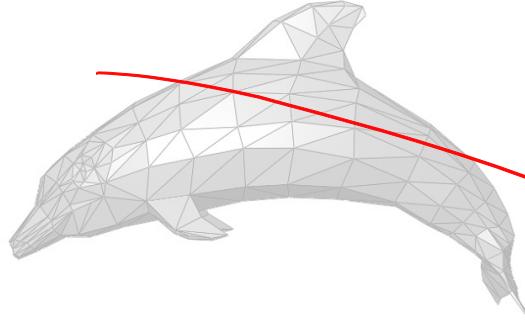
Explicit Surfaces

- E.g., height fields

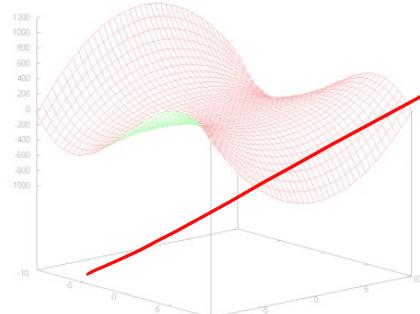


Overview

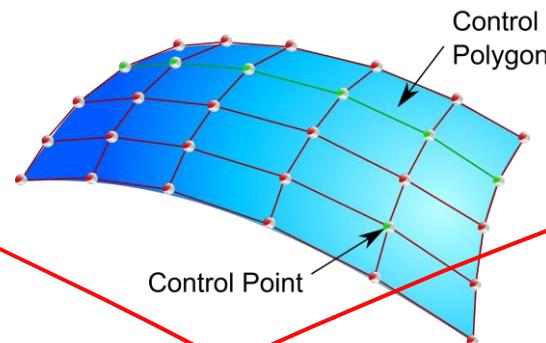
- Polygonal Meshes



- Explicit Surfaces



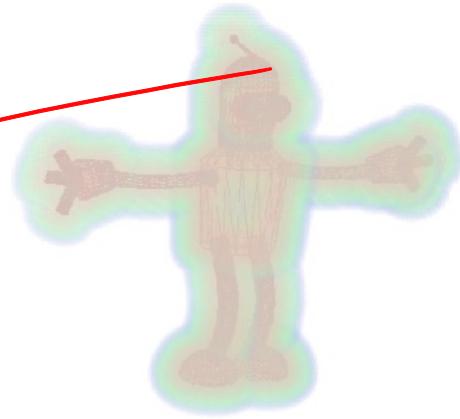
- Parametric Surfaces



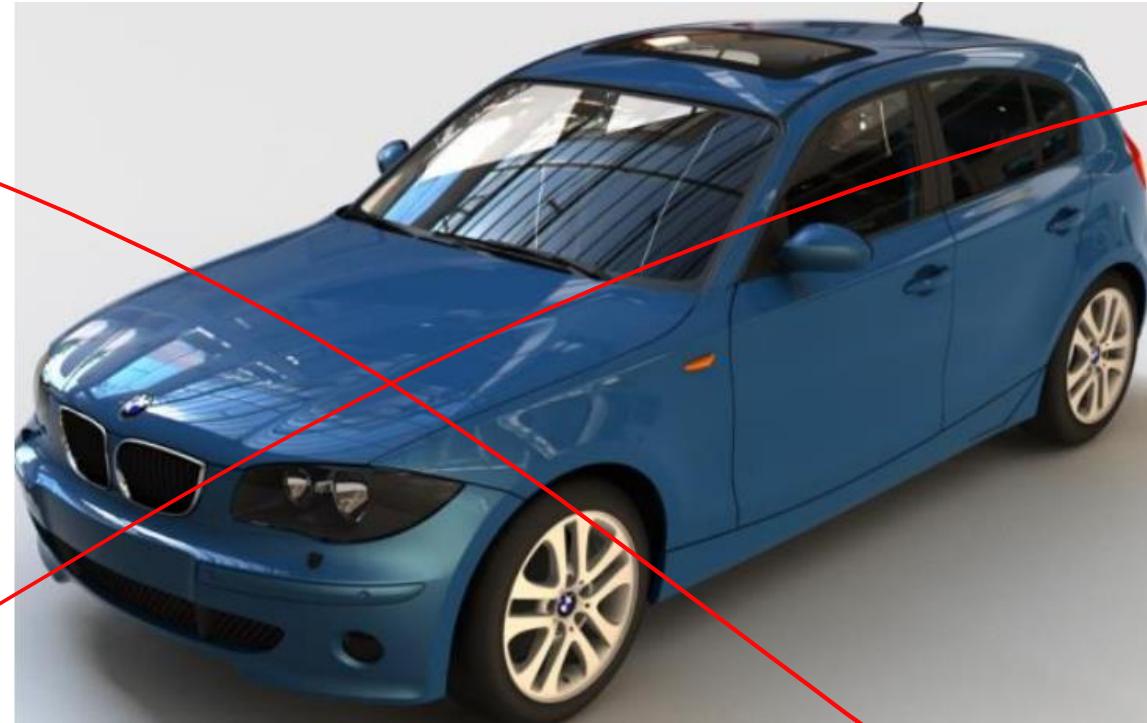
- Constructive Solid Geometry



- Implicit Surfaces



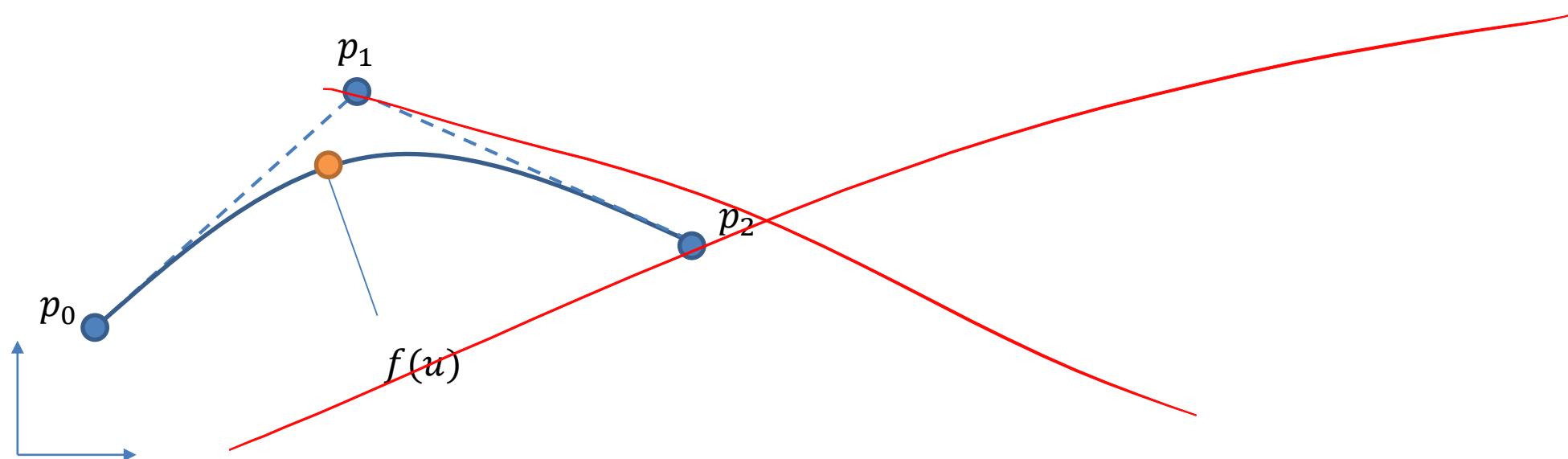
Parametric Surfaces



– Used in design and construction → freeform surfaces

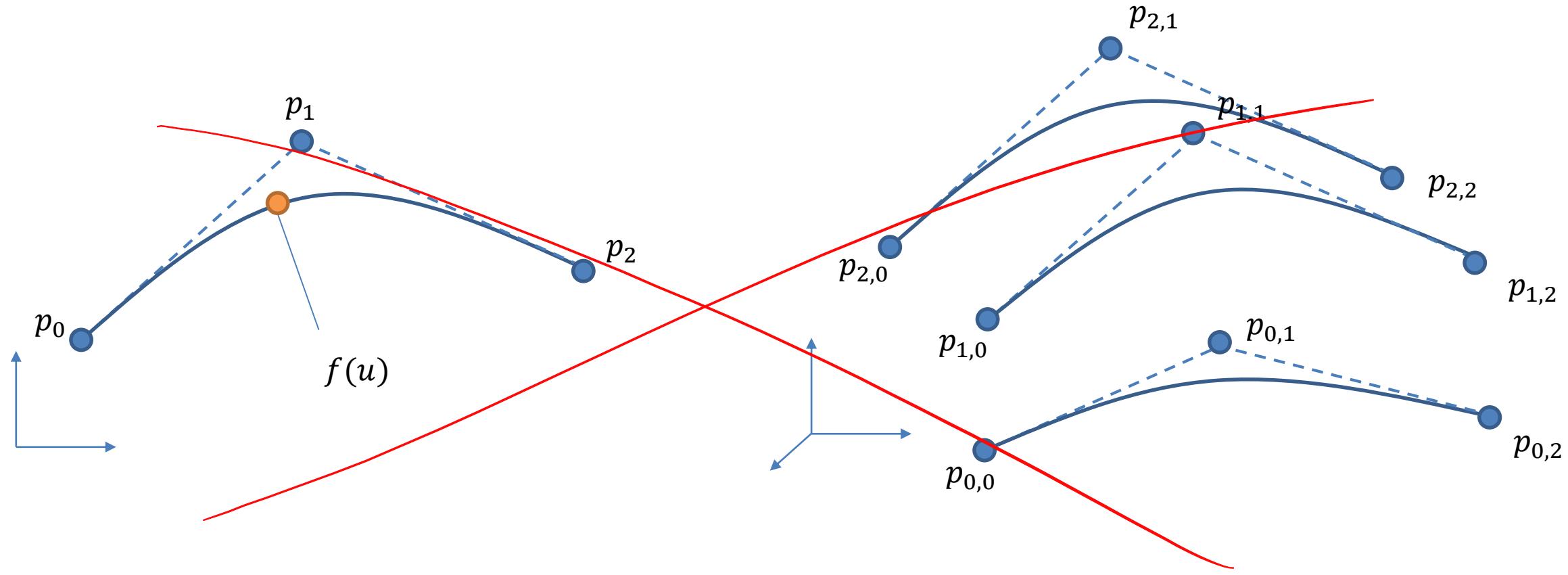
Parametric Surfaces

- Example: Bézier Curves & Tensor Product Surface



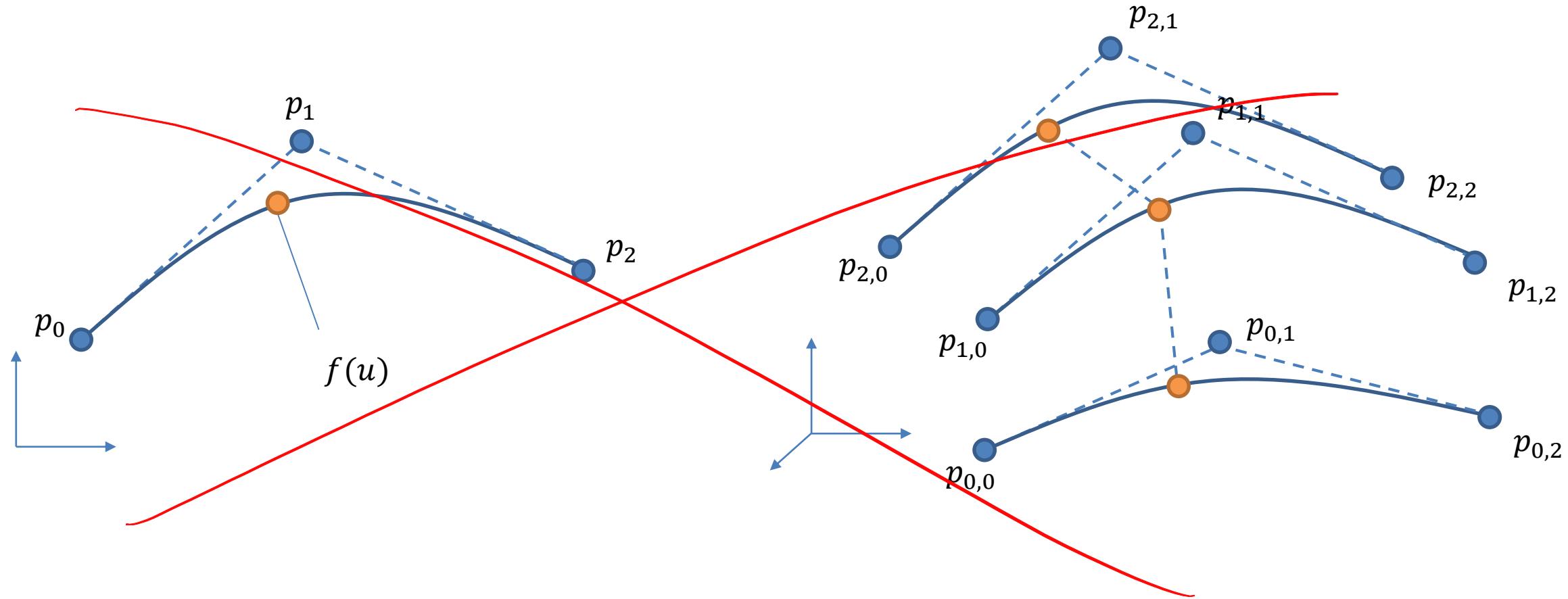
Parametric Surfaces

- Example: Bézier Curves & Tensor Product Surface



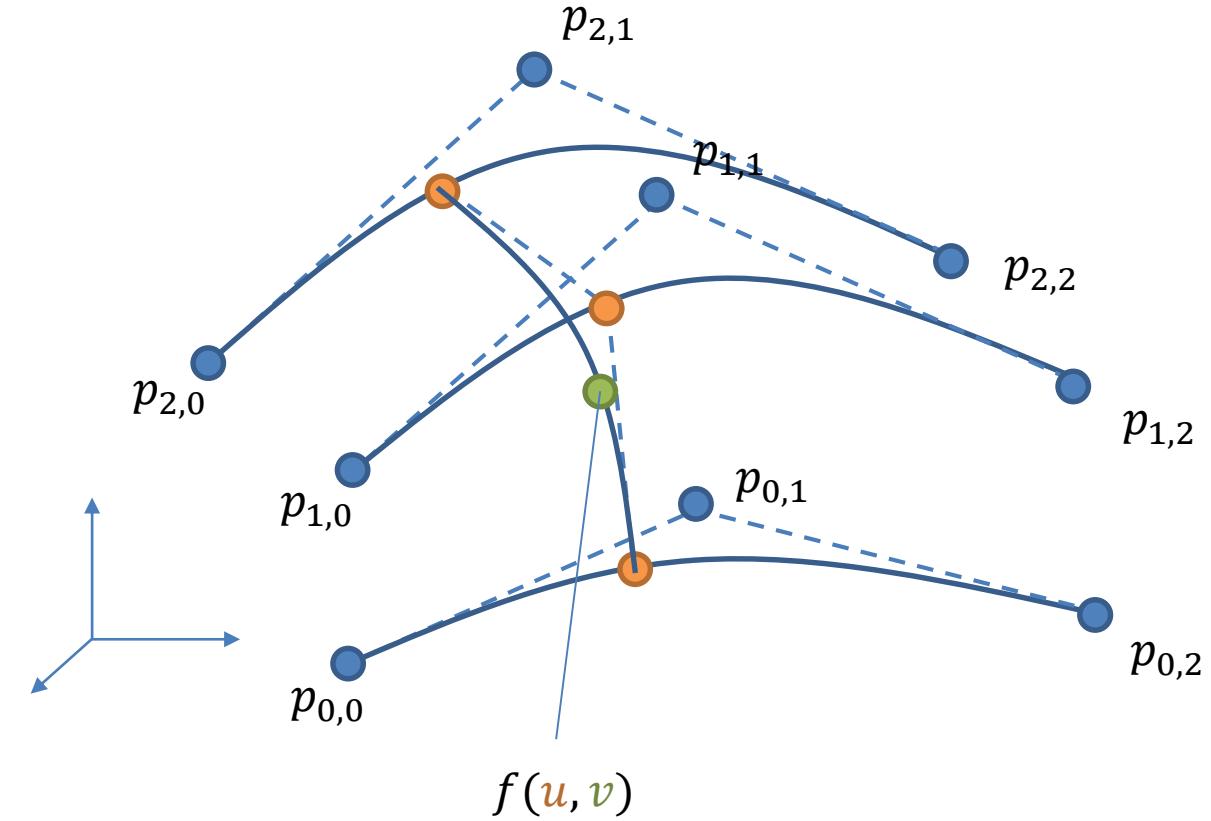
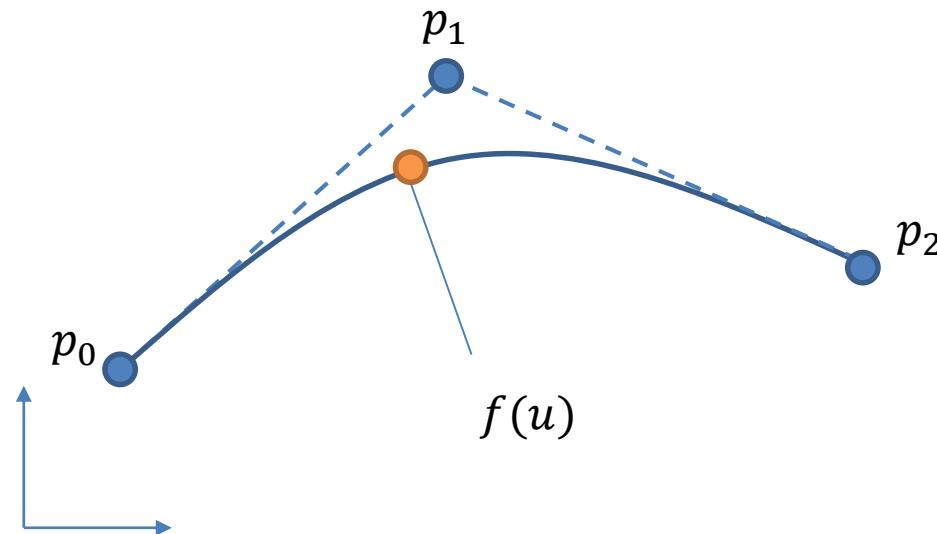
Parametric Surfaces

- Example: Bézier Curves & Tensor Product Surface



Parametric Surfaces

- Example: Bézier Curves & Tensor Product Surface

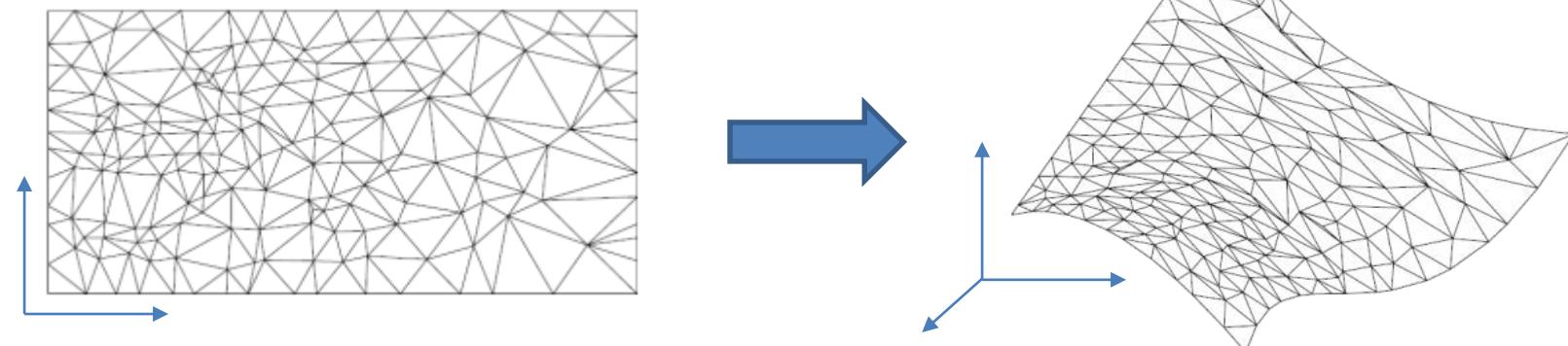


Parametric Surfaces

- Parametric form:

$$f(u, v): \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

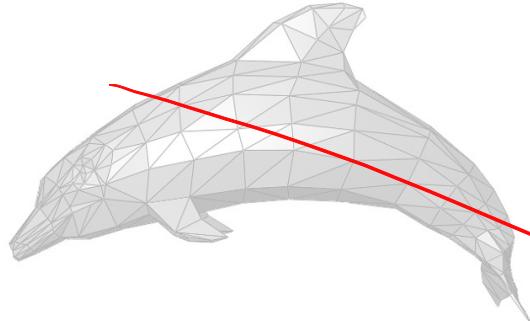
- Maps a bounded 2D domain to 3D
 - Bézier-, B-Spline, or NURBS surfaces



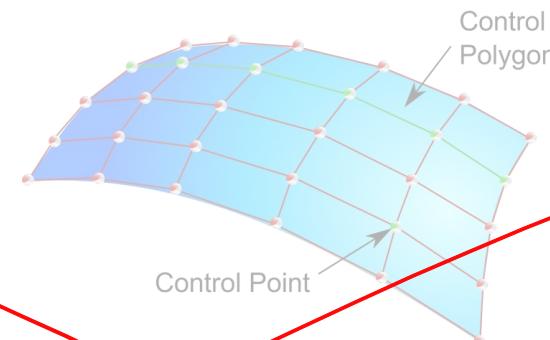
- Used in design and construction → freeform surfaces

Overview

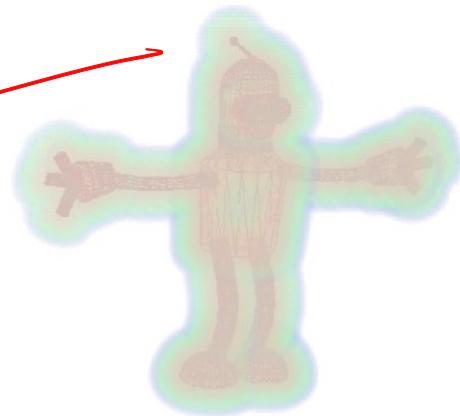
- Polygonal Meshes



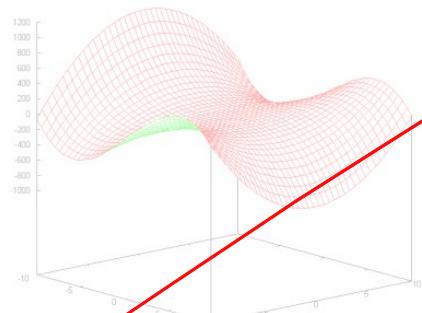
- Parametric Surfaces



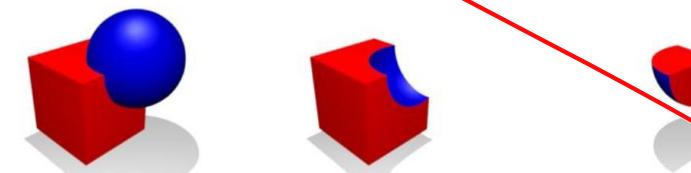
- Implicit Surfaces



- Explicit Surfaces

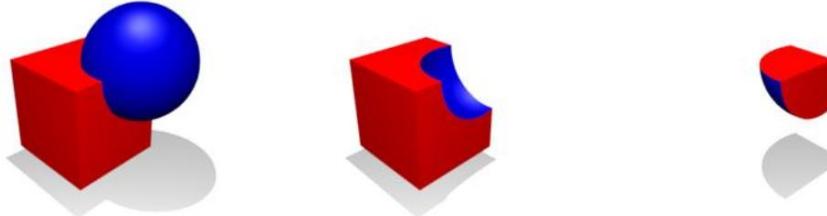


- Constructive Solid Geometry



Constructive Solid Geometry

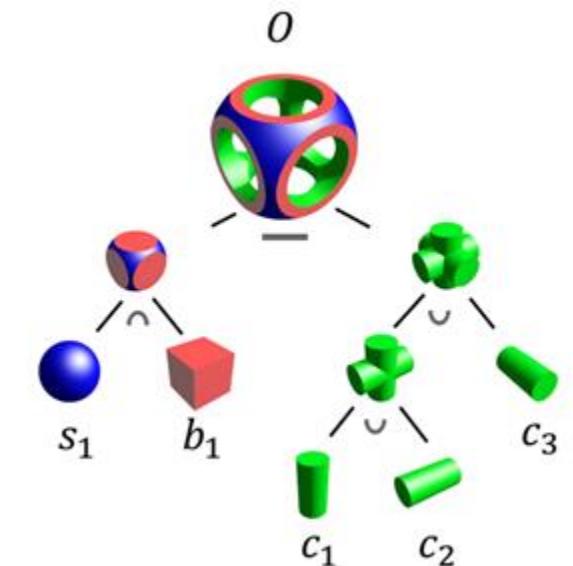
- Surface is defined as the boundary of a solid object that was created by Boolean operations on primitive solids



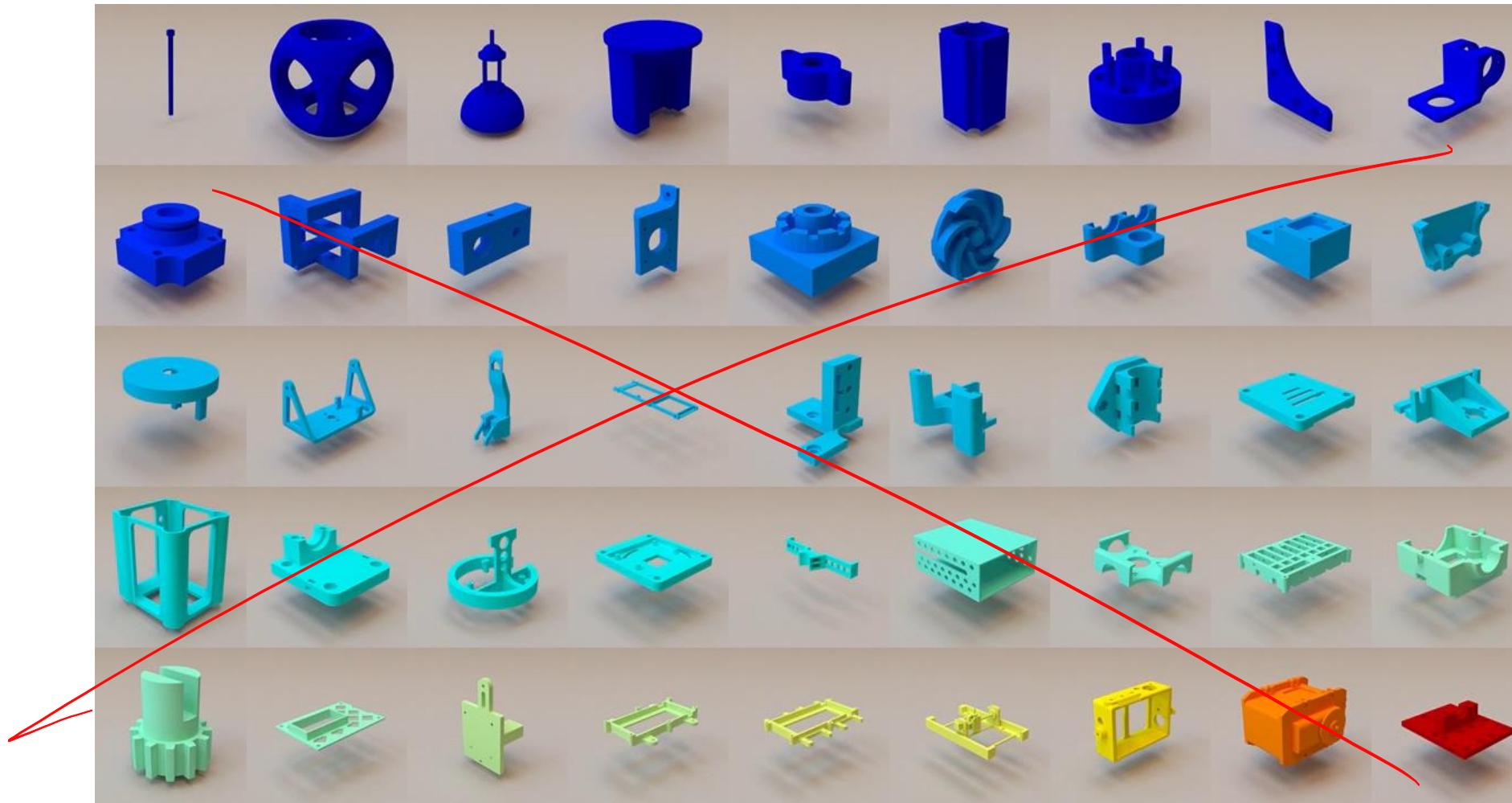
- Example:

$$O = (s_1 \cap b_1) - (c_3 \cup (c_1 \cup c_2))$$

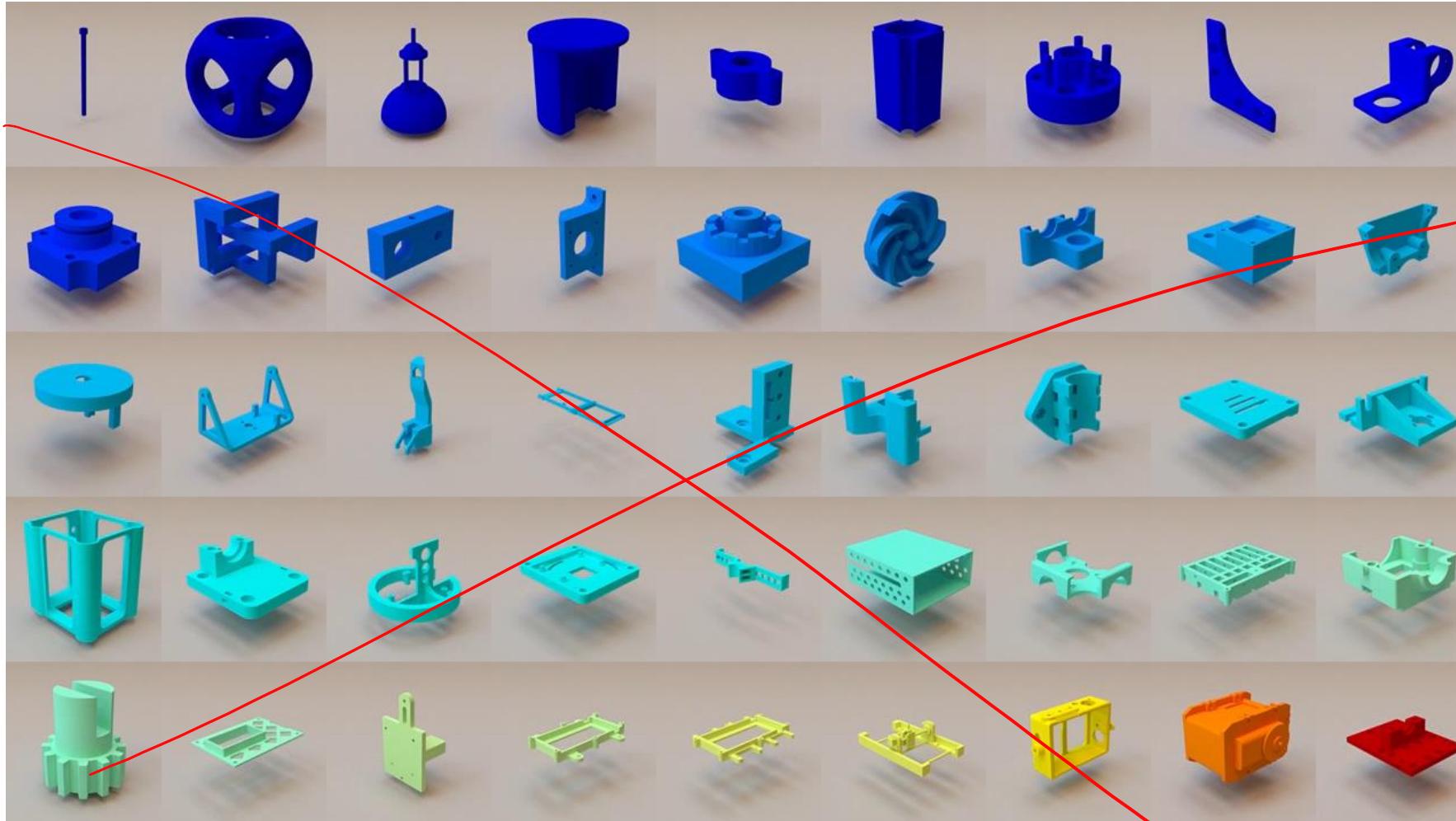
- Used in design and construction
 - Hard to model “organic” shapes



Constructive Solid Geometry

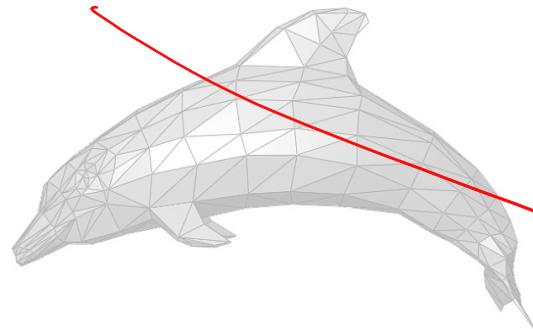


Constructive Solid Geometry

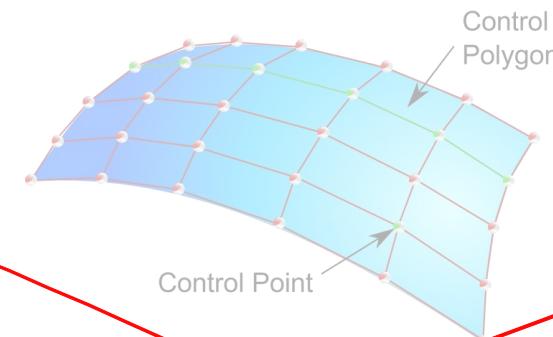


Overview

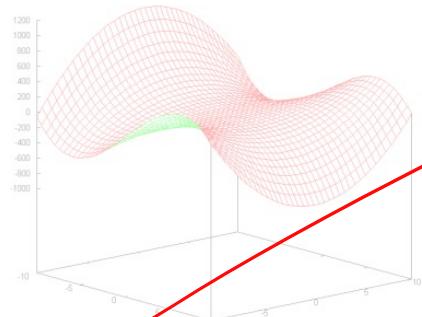
- Polygonal Meshes



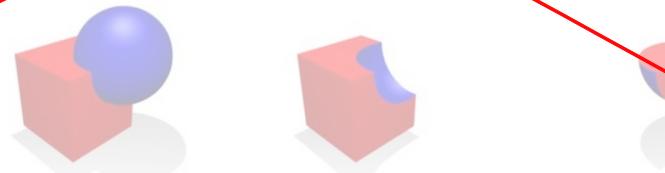
- Parametric Surfaces



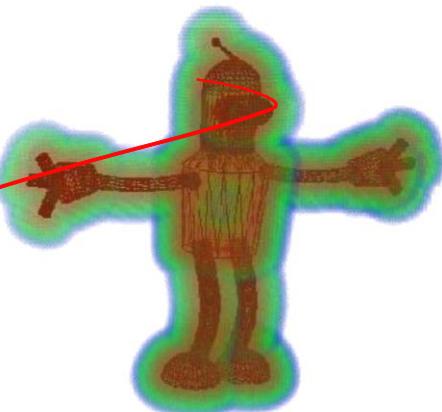
- Explicit Surfaces



- Constructive Solid Geometry



- Implicit Surfaces



Implicit Surfaces

- [Curless and Levoy], KinectFusion, VoxelHashing, ...



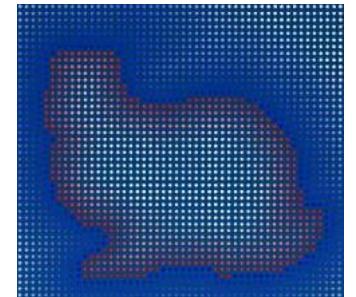
Implicit Surfaces

- ~~Implicit form:~~

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

- Surface is defined by the level set of the tri-variate scalar function

$$f(x, y, z) = c$$



Implicit Surfaces

- Implicit form:

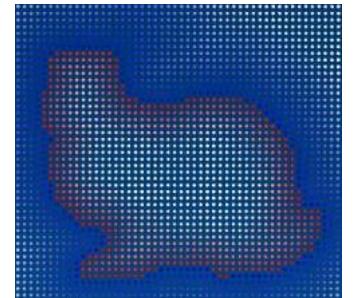
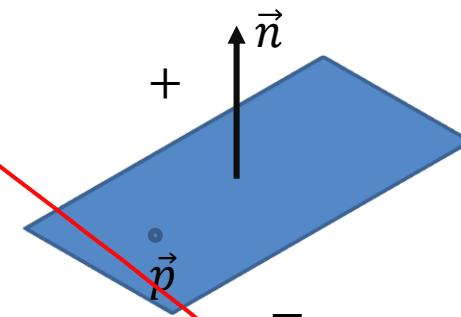
$$f(x, y, z) : \mathbb{R}^3 \rightarrow \mathbb{R}$$

- Surface is defined by the level set of the tri-variate scalar function

$$f(x, y, z) = c$$

- Example: Hesse normal form

$$f(x, y, z) = \left(\begin{pmatrix} x \\ y \\ z \end{pmatrix} - \vec{p} \right) \cdot \vec{n} = 0$$



Implicit Surfaces

- Implicit form:

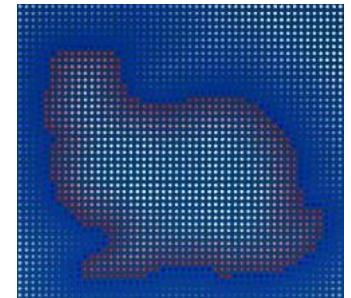
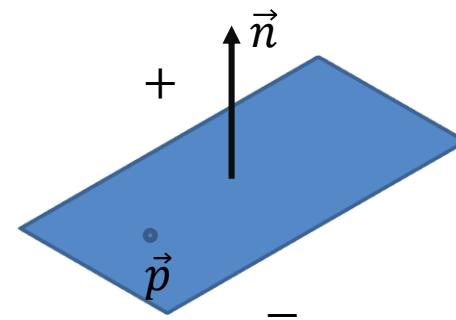
$$f(x, y, z): \mathbb{R}^3 \rightarrow \mathbb{R}$$

- Surface is defined by the level set of the tri-variate scalar function

$$f(x, y, z) = c$$

- Example: Hesse normal form

$$f(x, y, z) = \left(\begin{pmatrix} x \\ y \\ z \end{pmatrix} - \vec{p} \right) \cdot \vec{n} = 0$$

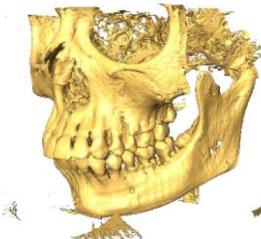
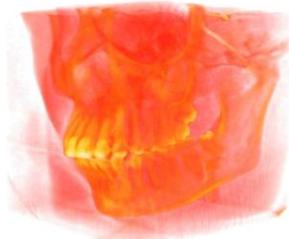


f is also called **Signed Distance Function (SDF)**

Implicit Surfaces

- Examples of such scalar functions $f(x, y, z)$

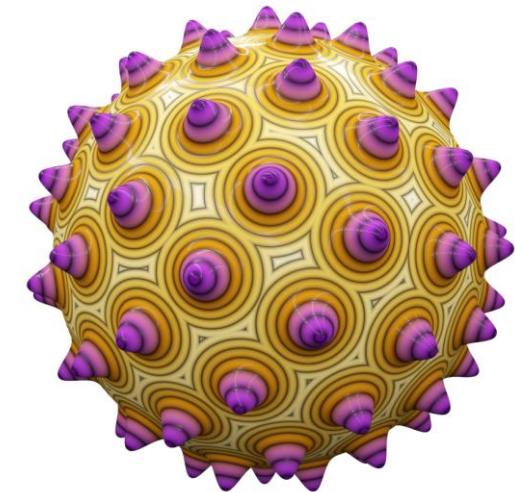
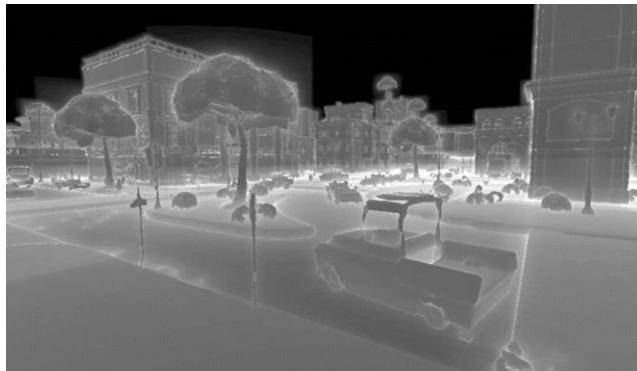
- Density



- Heat



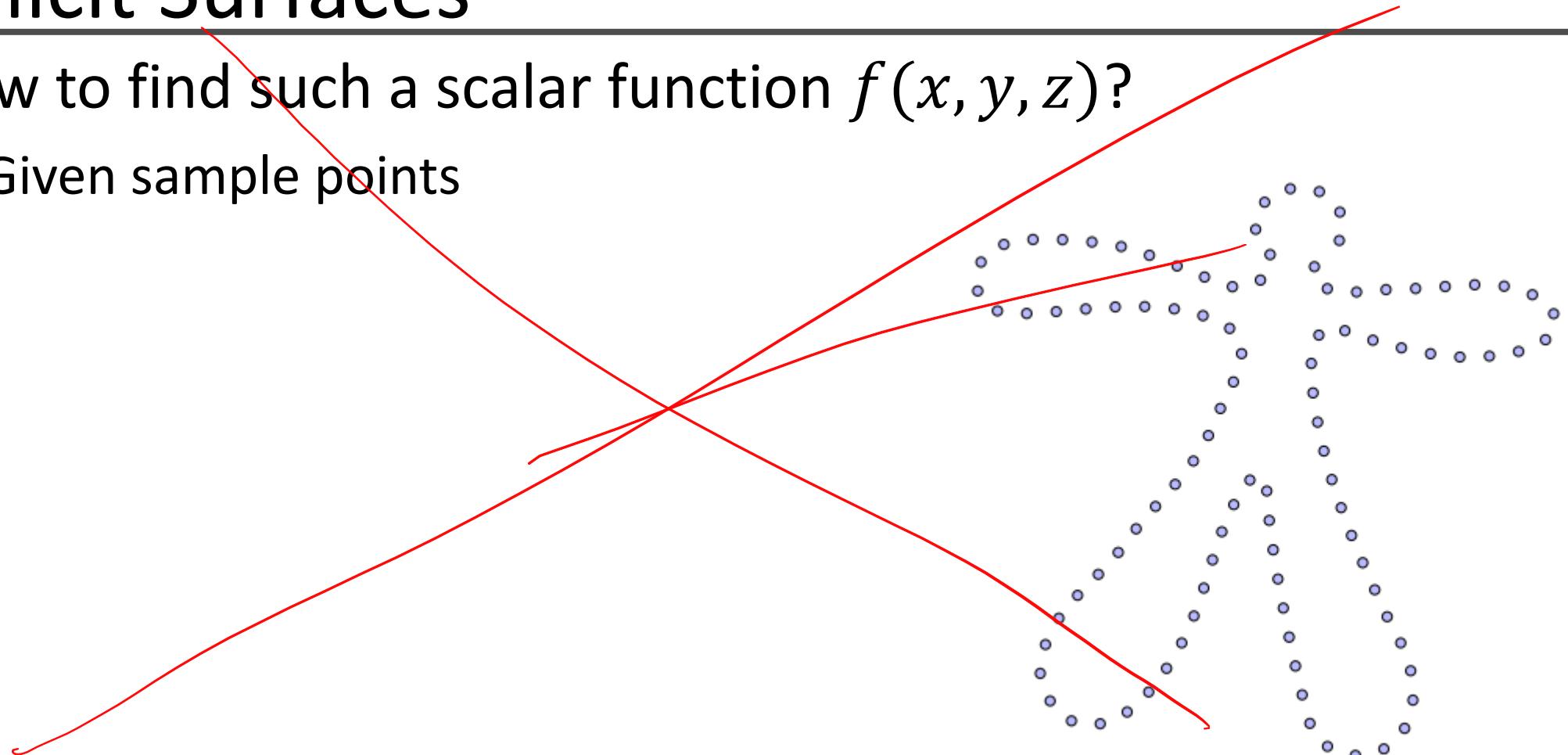
- Distance



Used for soft shadows in game engines (e.g., Unreal Engine)

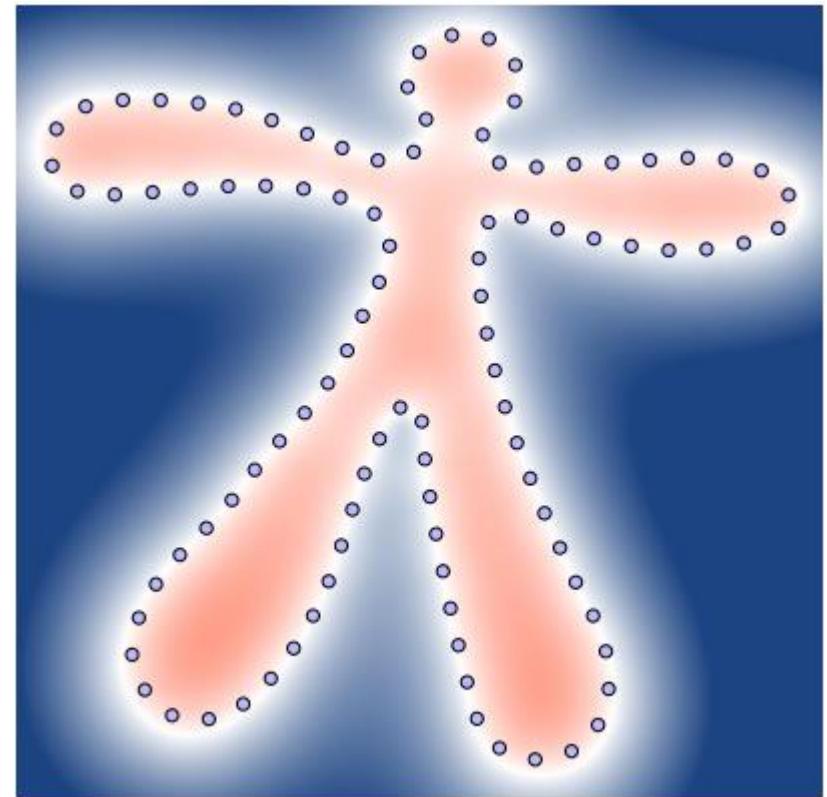
Implicit Surfaces

- How to find such a scalar function $f(x, y, z)$?
 - Given sample points



Implicit Surfaces

- How to find such a scalar function $f(x, y, z)$?
 - Given sample points
 - Find a scalar function that fulfils:
 - $f = 0$ at the sample points
 - $f < 0$ inside the object
 - $f > 0$ outside the object

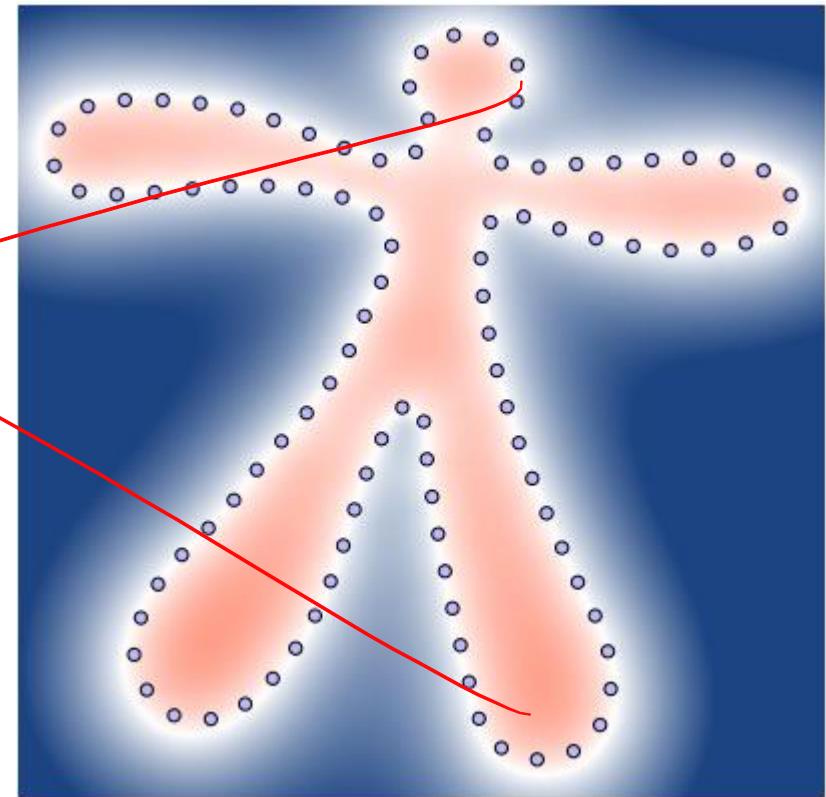
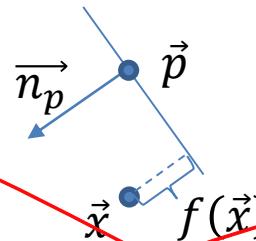


Implicit Surfaces

- How to find such a scalar function $f(x, y, z)$?
 - Hoppe'92:

$$f(\vec{x}) = (\vec{x} - \vec{p}) \cdot \vec{n}_p$$

\vec{p} is the closest point to \vec{x}



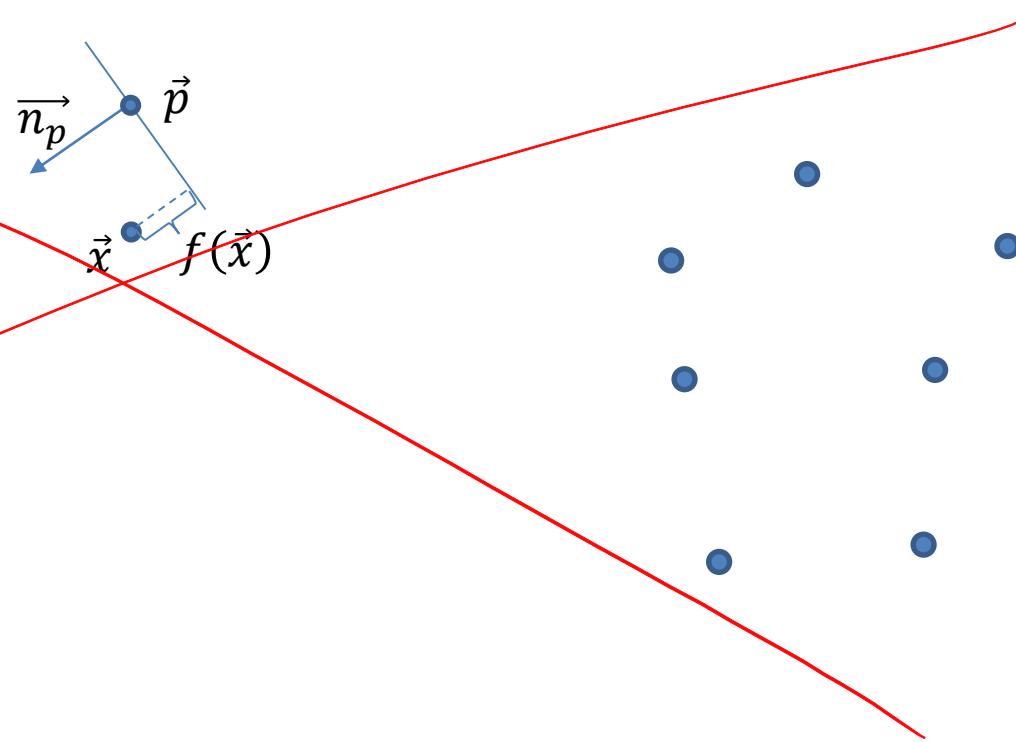
[Hoppe'92] H.Hoppe et al., "Surface reconstruction from unorganized points" (SIGGRAPH 92)

Implicit Surfaces

- How to find such a scalar function $f(x, y, z)$?
 - Hoppe'92:

$$f(\vec{x}) = (\vec{x} - \vec{p}) \cdot \vec{n}_p$$

\vec{p} is the closest point to \vec{x}



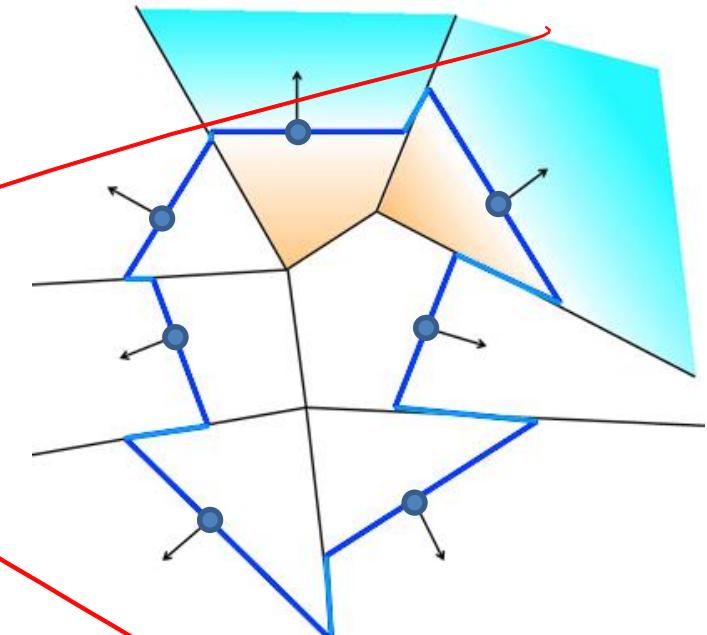
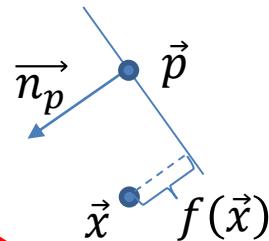
[Hoppe'92] H.Hoppe et al., "Surface reconstruction from unorganized points" (SIGGRAPH 92)

Implicit Surfaces

- How to find such a scalar function $f(x, y, z)$?
 - Hoppe'92:

$$f(\vec{x}) = (\vec{x} - \vec{p}) \cdot \vec{n}_p$$

\vec{p} is the closest point to \vec{x}



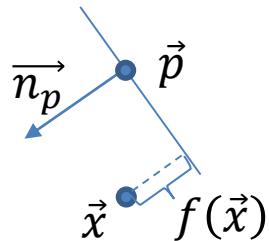
[Hoppe'92] H.Hoppe et al., "Surface reconstruction from unorganized points" (SIGGRAPH 92)

Implicit Surfaces

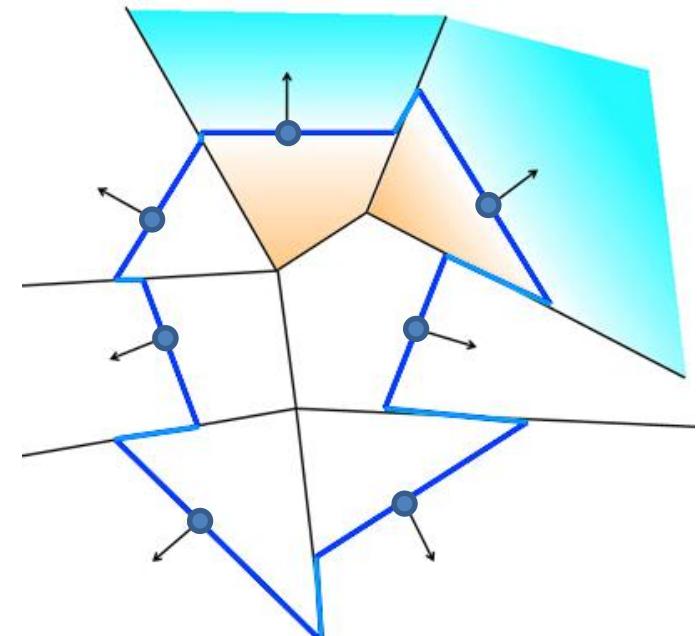
- How to find such a scalar function $f(x, y, z)$?
 - Hoppe'92:

$$f(\vec{x}) = (\vec{x} - \vec{p}) \cdot \vec{n}_p$$

\vec{p} is the closest point to \vec{x}

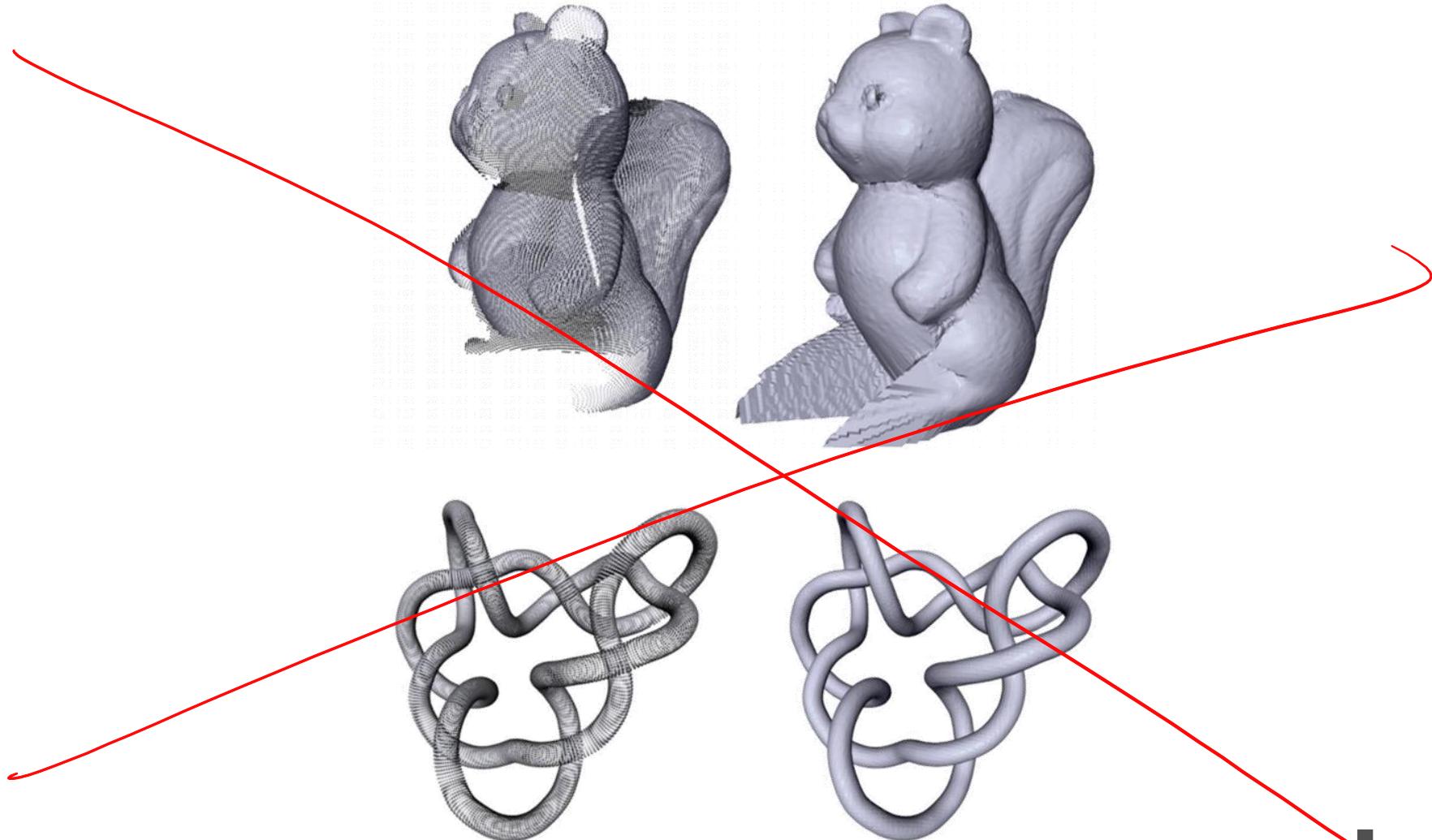


- Piecewise linear, defined on the Voronoi diagram of the input
- Discontinuous along Voronoi edges
- Dependent on the input density

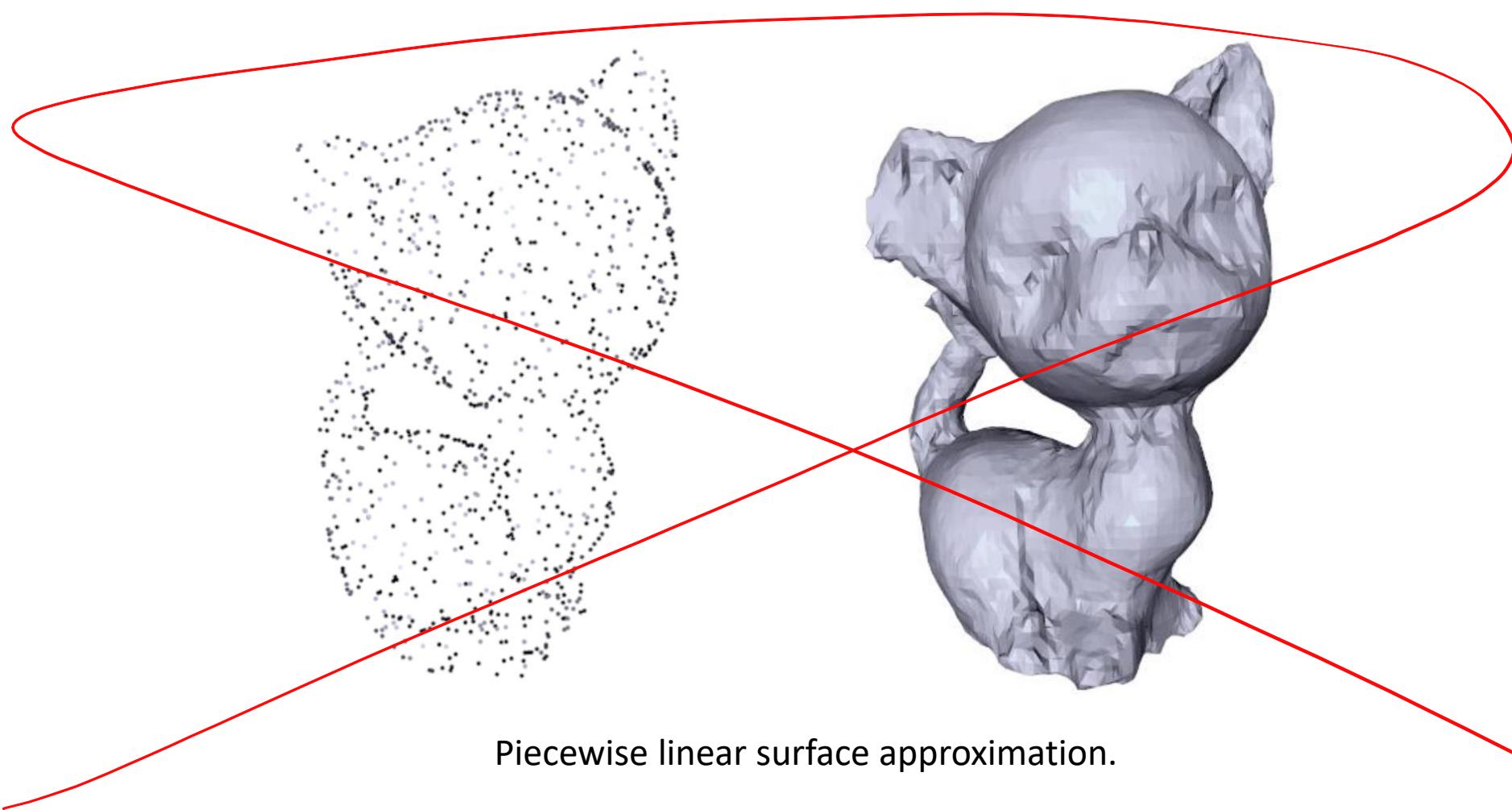


[Hoppe'92] H.Hoppe et al., "Surface reconstruction from unorganized points" (SIGGRAPH 92)

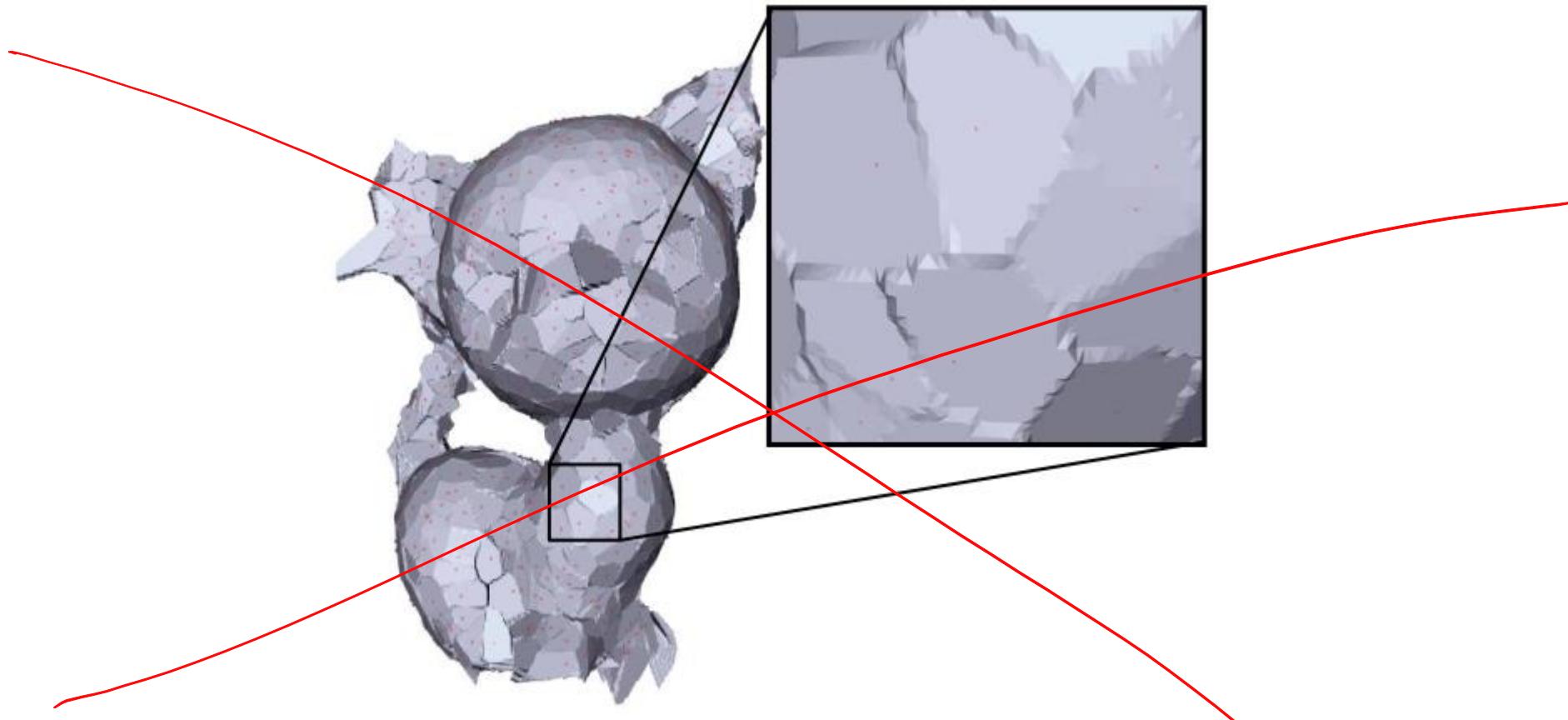
Implicit Surfaces



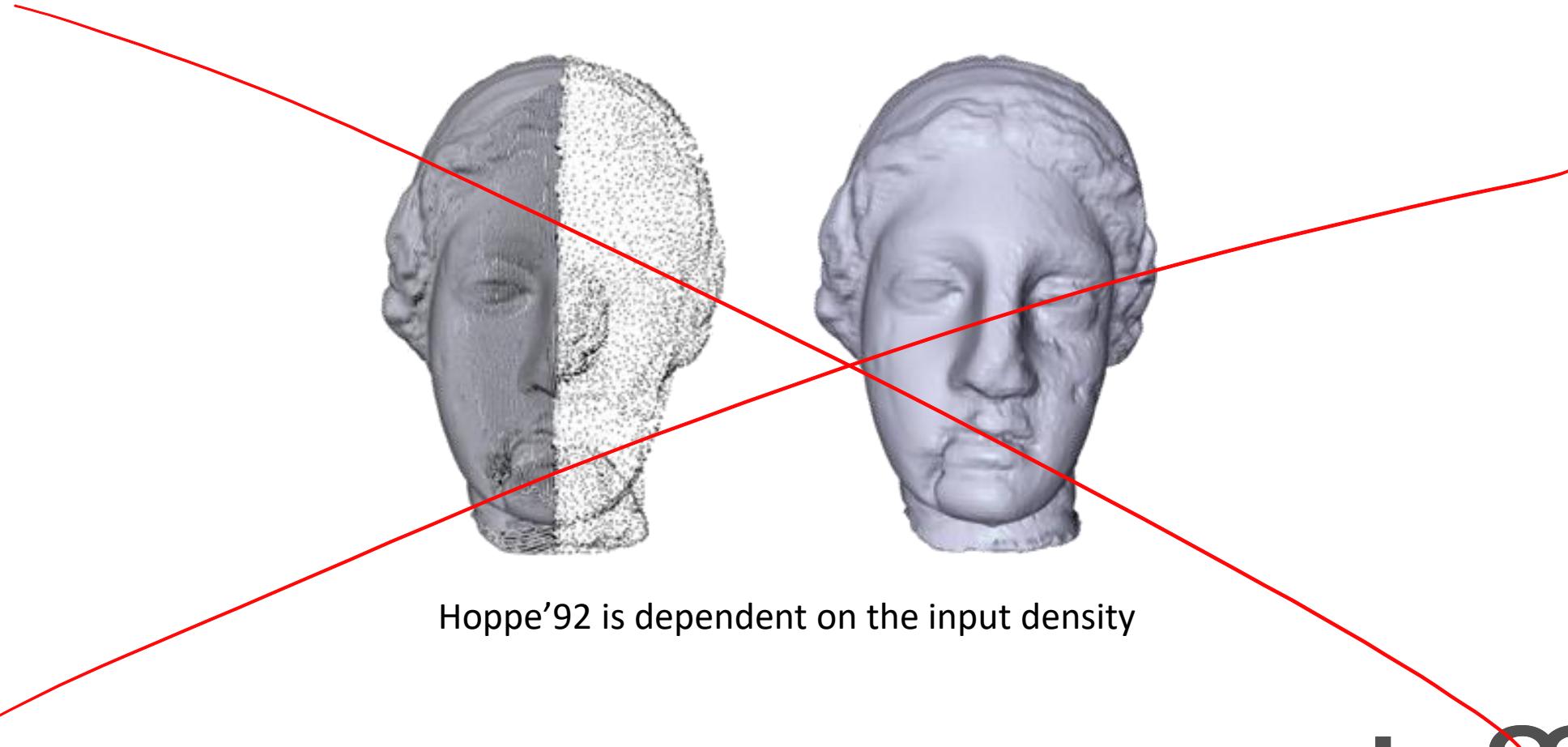
Implicit Surfaces



Implicit Surfaces



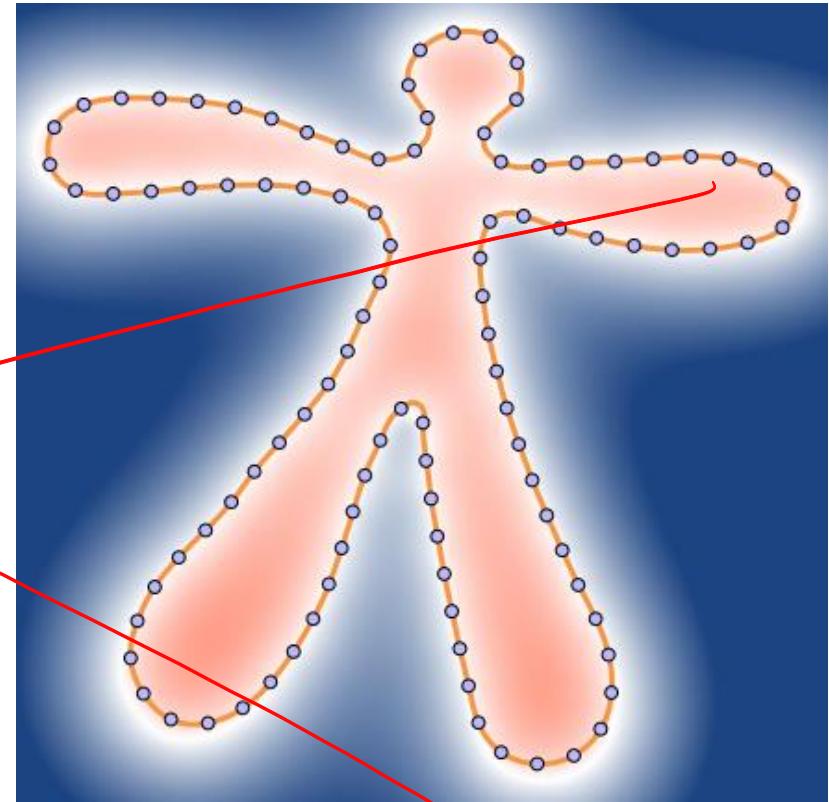
Implicit Surfaces



Implicit Surfaces

- How to find such a scalar function $f(x, y, z)$?

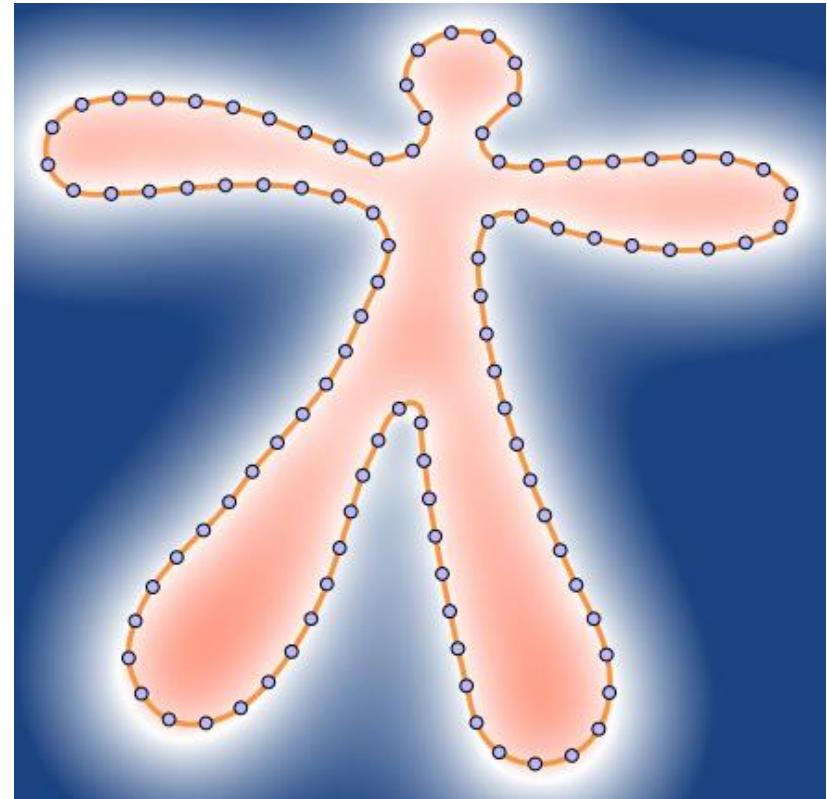
- Given sample points
- Find a scalar function that fulfils:
 - $f = 0$ at the sample points
 - $f < 0$ inside the object
 - $f > 0$ outside the object
 - **f should be smooth**



Implicit Surfaces

- How to find such a scalar function $f(x, y, z)$?
 - Given sample points
 - Find a scalar function that fulfils:
 - $f = 0$ at the sample points
 - $f < 0$ inside the object
 - $f > 0$ outside the object
 - **f should be smooth**

→ Radial Basis Functions (RBF)

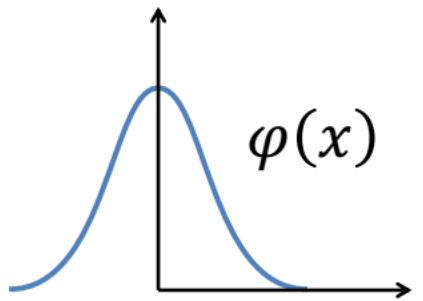
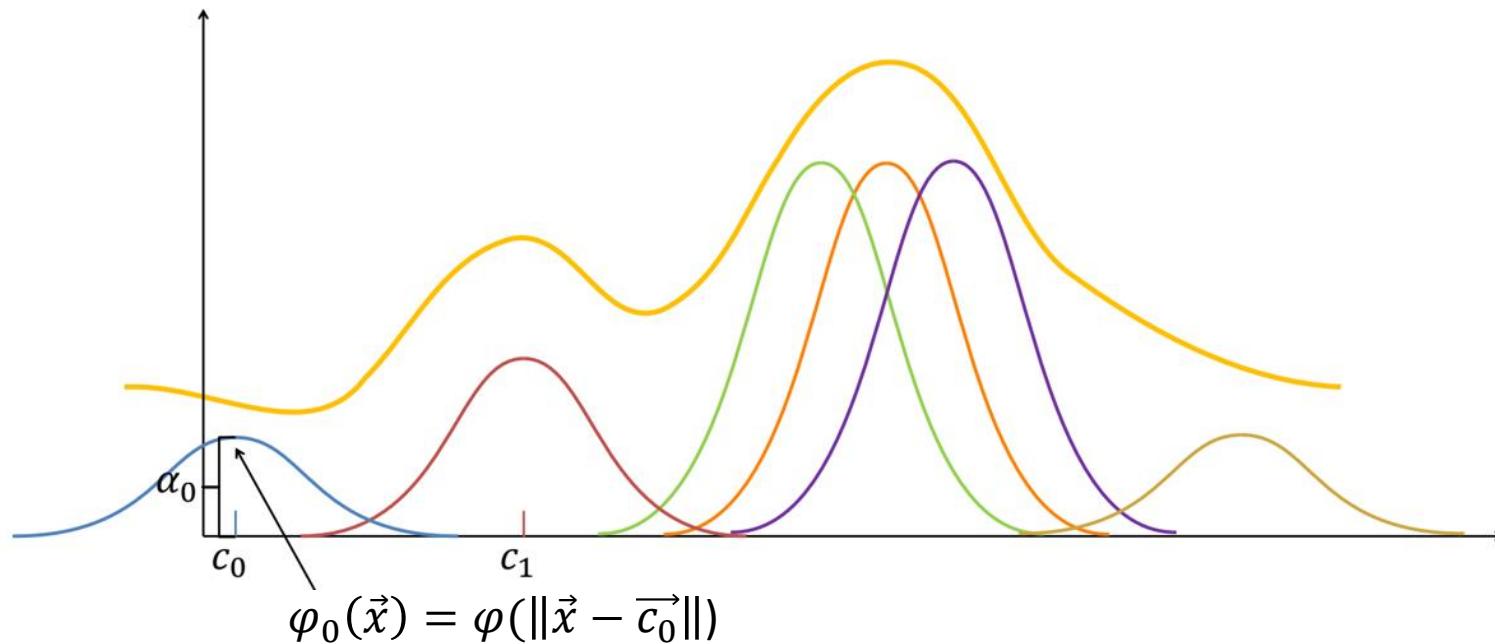


[Turk'99] G.Turk et al., "Variational Implicit Surfaces"

Implicit Surfaces – Radial Basis Functions

- Idea: each complex function can be approximated as the sum of simple scaled and translated kernel functions $\varphi(x)$:

$$g(\vec{x}) \approx f(\vec{x}) = \sum_i \alpha_i \cdot \varphi_i(\vec{x}) = \sum_i \alpha_i \cdot \varphi(\|\vec{x} - \vec{c}_i\|)$$



Implicit Surfaces – Radial Basis Functions

- A **Radial Basis Function** (RBF), also called a **Radial Basis Function Network** is defined as sum of translated and scaled kernels and a linear polynomial:

$$f(\vec{x}) = \sum_i \alpha_i \cdot \varphi_i(\vec{x}) + \underbrace{\vec{b} \cdot \vec{x}}_{\text{linear term}} + d$$

- Allow for functions of arbitrary complexity!
 - complexity increases with the number of used kernel functions
- If basis functions $\varphi_i(x)$ are smooth, $f(\vec{x})$ is smooth as well.

Implicit Surfaces – Radial Basis Functions

- Radial Basis Functions in 3D:

$$f(\vec{x}) = \sum_i \alpha_i \cdot \varphi_i(\vec{x}) + \vec{b} \cdot \vec{x} + d$$

- Use the biharmonic radial basis function:

$$\varphi_i(\vec{x}) = \|\vec{p}_i - \vec{x}\|^3$$

- Where \vec{p}_i are the input sample points

$$f(\vec{x}) = \sum_i \alpha_i \cdot \|\vec{p}_i - \vec{x}\|^3 + \vec{b} \cdot \vec{x} + d$$

[Turk'99] G.Turk et al., "Variational Implicit Surfaces"

Implicit Surfaces – Radial Basis Functions

- Radial Basis Functions in 3D:

$$f(\vec{x}) = \sum_i \alpha_i \cdot \varphi_i(\vec{x}) + \vec{b} \cdot \vec{x} + d$$

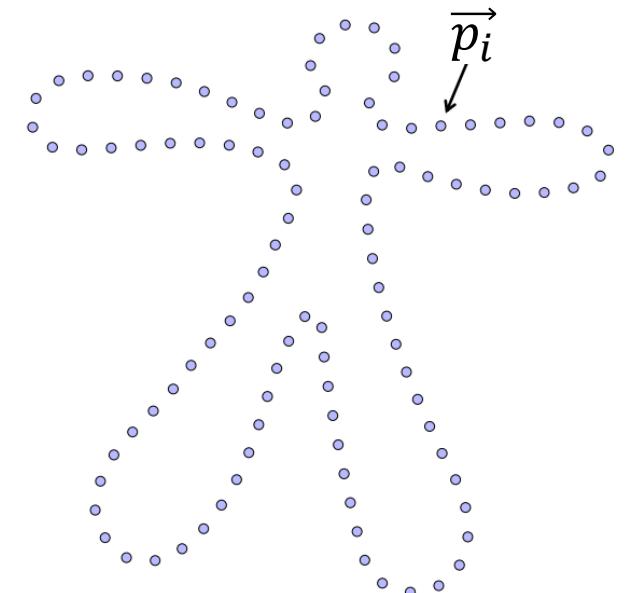
- Use the biharmonic radial basis function:

$$\varphi_i(\vec{x}) = \|\vec{p}_i - \vec{x}\|^3$$

- Where \vec{p}_i are the input sample points

$$f(\vec{x}) = \sum_i \color{red}{\alpha_i} \cdot \|\vec{p}_i - \vec{x}\|^3 + \color{red}{\vec{b}} \cdot \vec{x} + \color{red}{d}$$

unknowns



Implicit Surfaces – Radial Basis Functions

$$f(\vec{x}) = \sum_i \alpha_i \cdot \|\vec{p}_i - \vec{x}\|^3 + \vec{b} \cdot \vec{x} + d$$

unknowns

- $f(\vec{x})$ should be zero at the sample points:
 - $f(\vec{p}_i) = 0 \rightarrow n$ equations (n : number of sample points)

→ Solve system of linear equations: $A\vec{x} = \vec{b}$

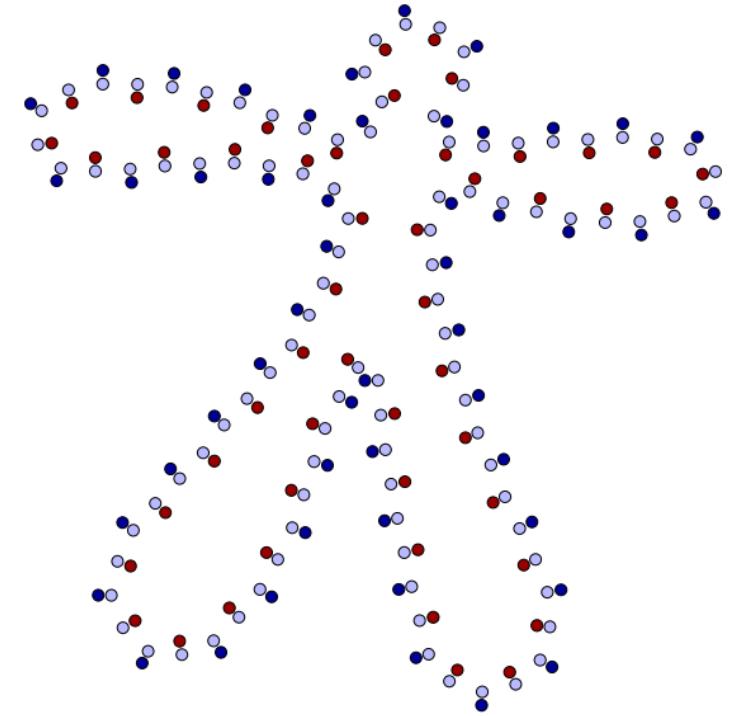
- Note: system is **underdetermined** since we have $n + 4$ unknowns
- Solving the system will result in the trivial result ($\vec{x} = \vec{0}$)

Implicit Surfaces – Radial Basis Functions

$$f(\vec{x}) = \sum_i \alpha_i \cdot \|\vec{p}_i - \vec{x}\|^3 + \vec{b} \cdot \vec{x} + d$$

unknowns

- $f(\vec{x})$ should be zero at the sample points:
 - $f(\vec{p}_i) = 0 \rightarrow n$ equations
- Additional constraints where f is non-zero
 - Normal constraints:
 - For each sample point add off-surface points by moving the points a little in \pm normal direction.
 - Set the target distance value of these points to $\pm\epsilon$.



Implicit Surfaces – Radial Basis Functions

$$f(\vec{x}) = \sum_i \alpha_i \cdot \|\vec{p}_i - \vec{x}\|^3 + \vec{b} \cdot \vec{x} + d$$

$$\begin{array}{l} \text{on surface points} \\ \left[\begin{array}{ccccccc} \varphi_{1,1} & \cdots & \varphi_{1,n} & p_{1,x} & p_{1,y} & p_{1,z} & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \varphi_{n,1} & \cdots & \varphi_{n,n} & p_{n,x} & p_{n,y} & p_{n,z} & 1 \end{array} \right] \cdot \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ b_1 \\ b_2 \\ b_3 \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \pm \varepsilon \\ \vdots \\ \pm \varepsilon \end{bmatrix} \\ \text{off surface points} \\ \left[\begin{array}{ccccccc} \hat{\varphi}_{1,1} & \cdots & \hat{\varphi}_{1,n} & q_{1,x} & q_{1,y} & q_{1,z} & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{\varphi}_{n,1} & \cdots & \hat{\varphi}_{n,n} & q_{n,x} & q_{n,y} & q_{n,z} & 1 \end{array} \right] \cdot \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ b_1 \\ b_2 \\ b_3 \\ d \end{bmatrix} = \begin{bmatrix} \pm \varepsilon \\ \vdots \\ \pm \varepsilon \\ \vdots \\ \pm \varepsilon \\ \vdots \\ \pm \varepsilon \end{bmatrix} \end{array}$$

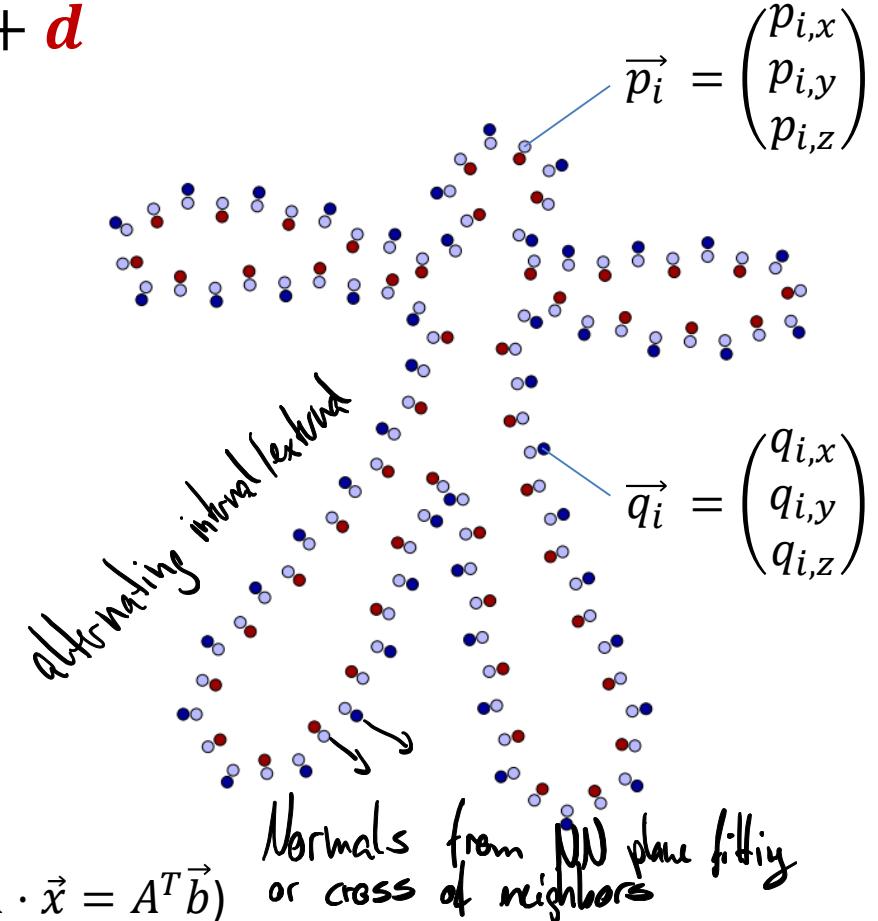
$\cdot \vec{x} = \vec{b}$

$$\varphi_{i,j} = \|\vec{p}_i - \vec{p}_j\|^3$$

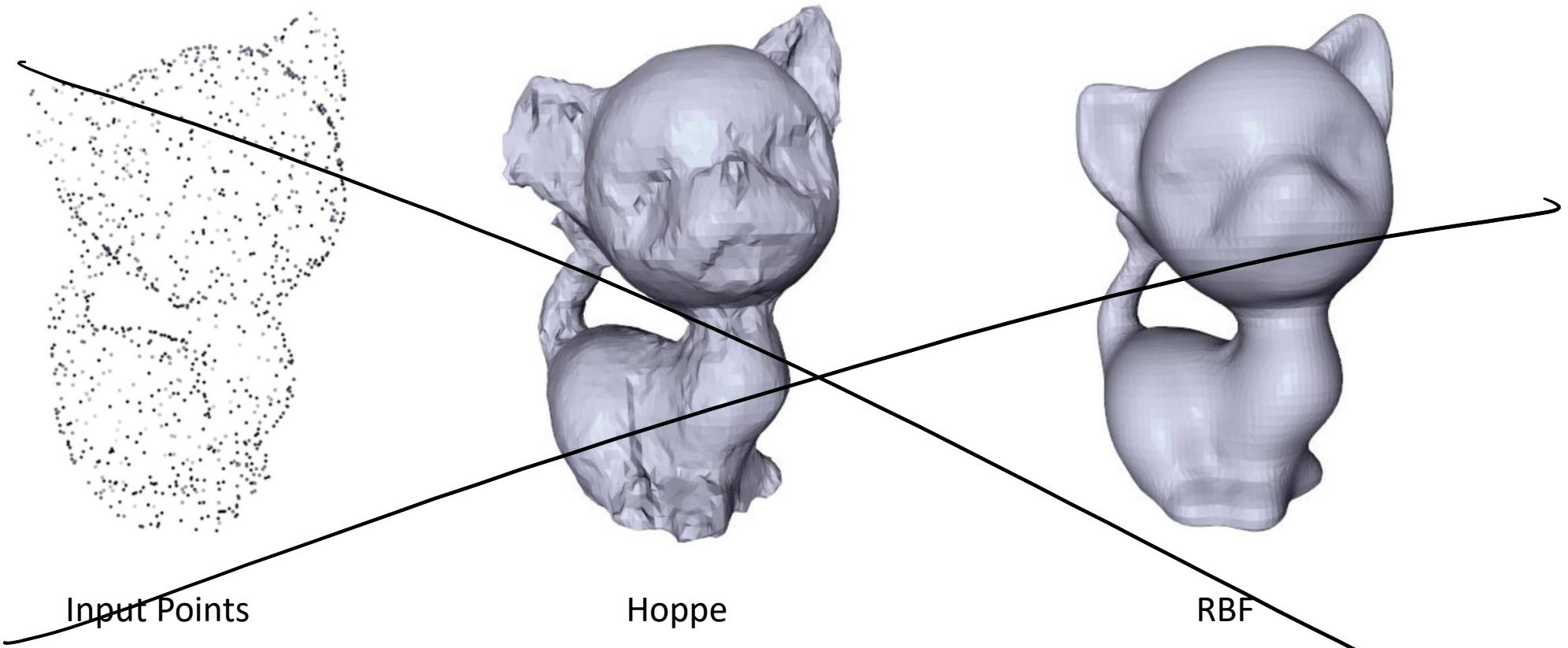
A

$$\hat{\varphi}_{i,j} = \|\vec{q}_i - \vec{p}_j\|^3$$

Note: System is **overdetermined!** \rightarrow use least squares solution ($A^T A \cdot \vec{x} = A^T \vec{b}$)



Implicit Surfaces – Radial Basis Functions

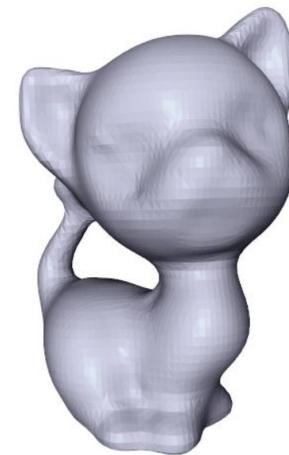
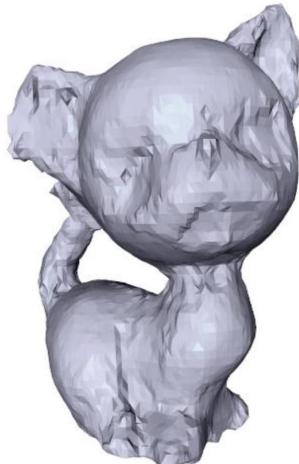


Implicit Surfaces – Radial Basis Functions



Hoppe

- Local method
- Fast and easy to implement
- Cannot handle noise, outliers, large holes



RBF

- Global method
- Requires solving a linear system (slow)
- Can only handle small point sets
- Can handle noise, outliers



- Find a compromise between local and global fitting
 - Fit point cloud in a coarse-to-fine manner
 - Segment point cloud into parts, fit individually

Implicit Surfaces

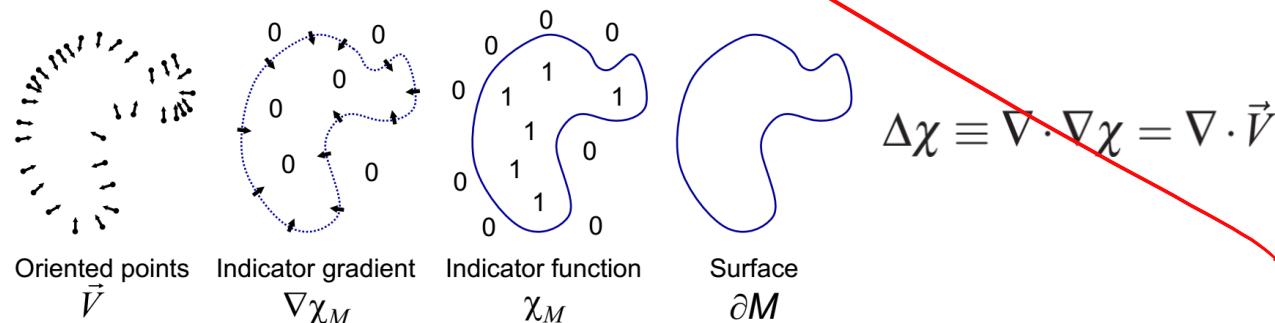
- Examples of further methods:

- RBF with floating centers

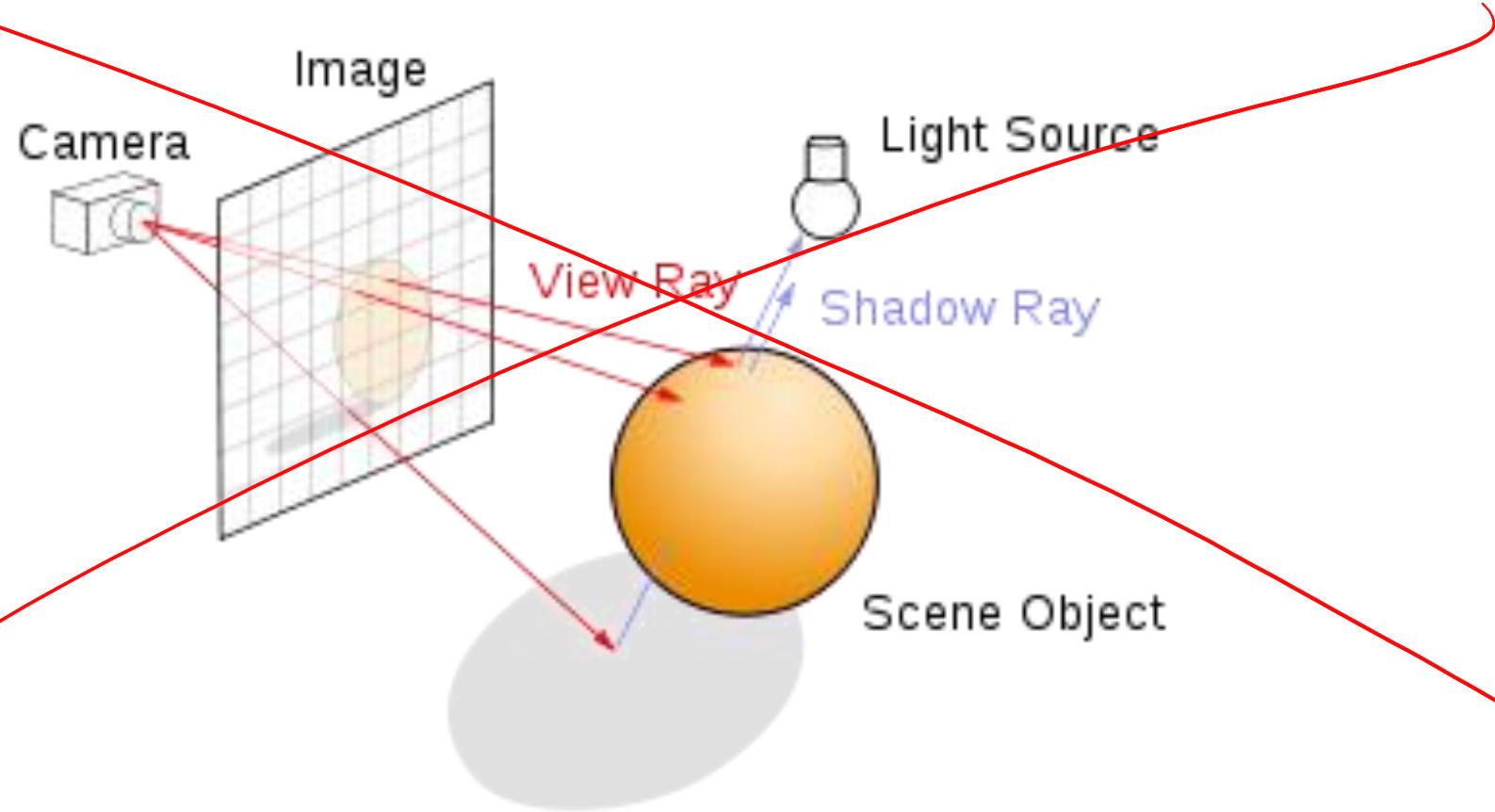
- Reduce number of RBF centers
 - Also optimize for the center positions → non-linear
 - [Süßmuth'10] J.Süßmuth "Surface Reconstruction based on Hierarchical Floating Radial Basis Functions"

- Poisson Reconstruction

- [Kazhdan'06] M.Kazhdan "Poisson Surface Reconstruction"

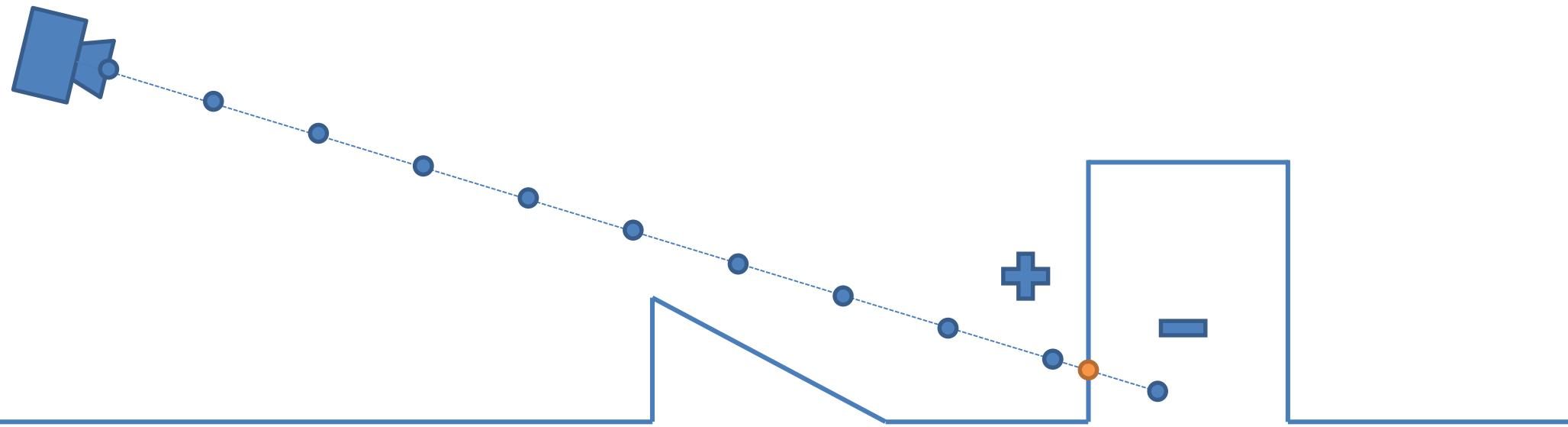


How to render Signed Distance Functions?



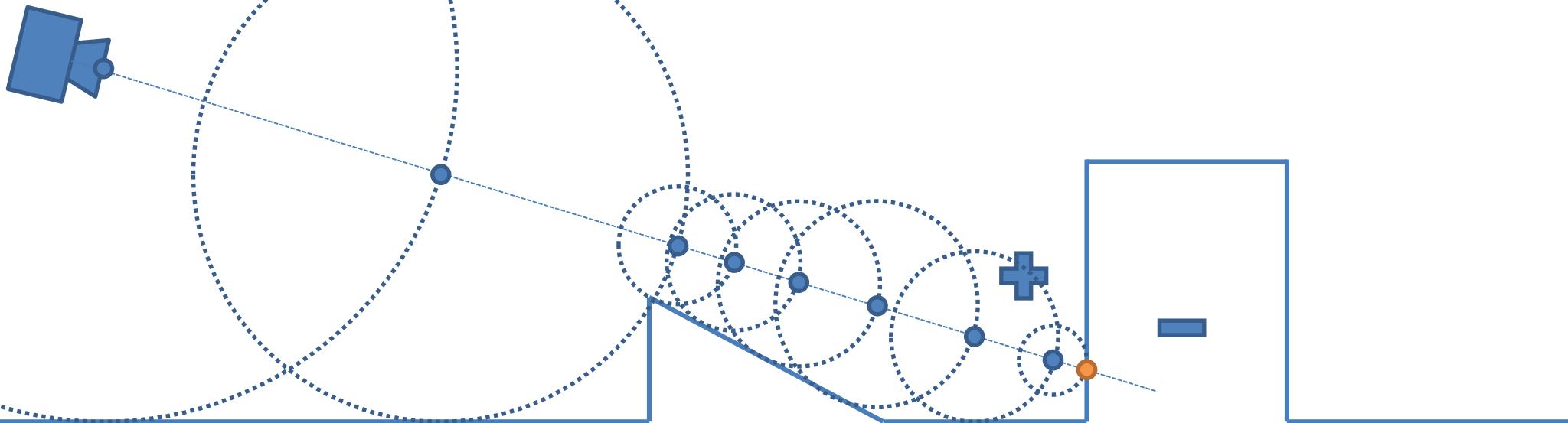
How to render Signed Distance Functions?

- Ray Marching
 - Fixed step length
 - Linear Interpolation, if zero crossing occurs



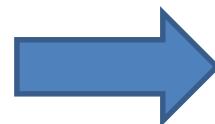
How to render Signed Distance Functions?

- Sphere Tracing
 - Dynamic step length
 - Stop if distance is below a threshold



How to render Signed Distance Functions?

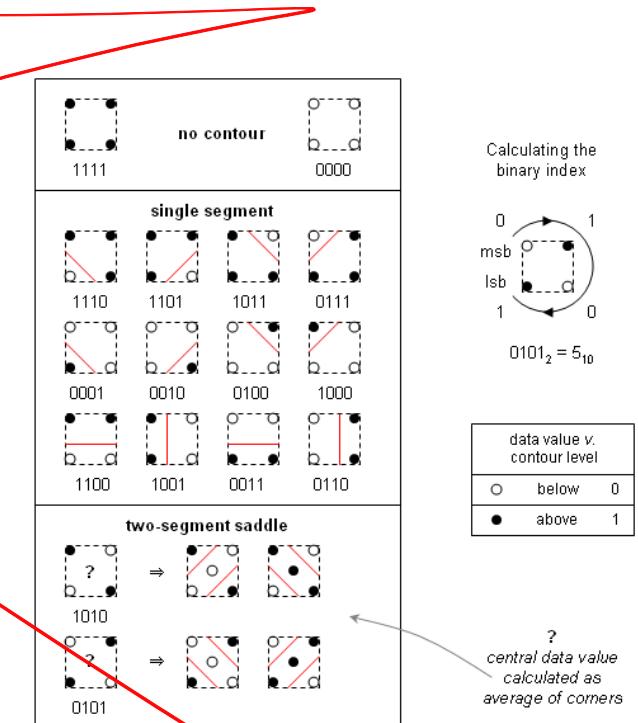
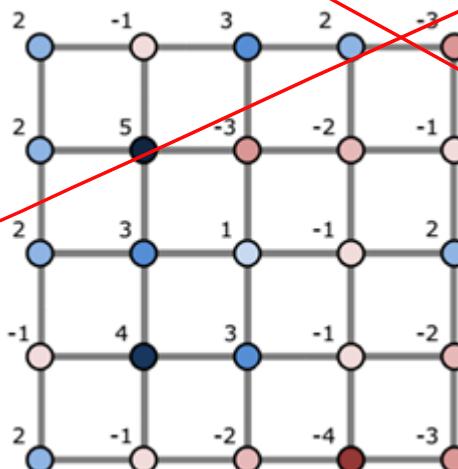
- Convert the iso-surface to polygonal mesh
 - Easy to render
 - Can be used by other post processing pipelines



Marching Cubes

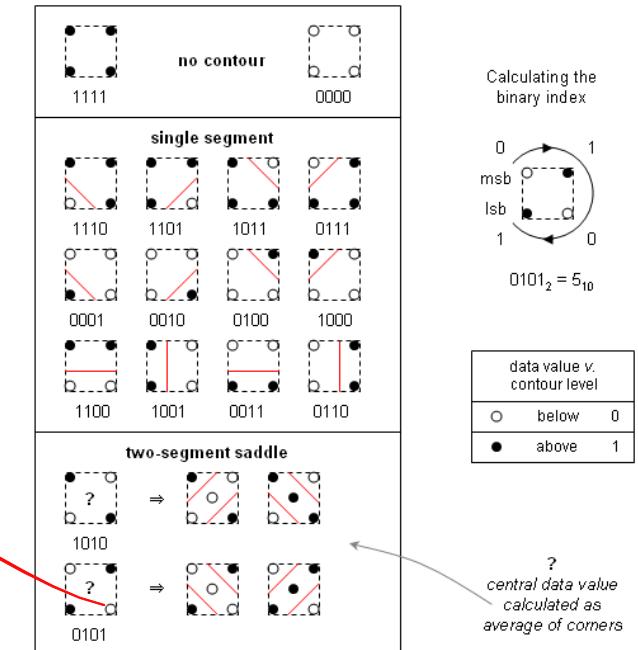
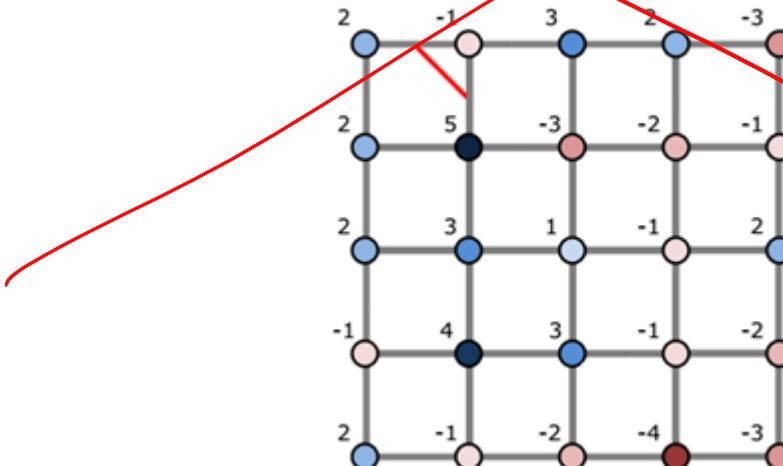
Marching Squares (2D)

- Converts an iso-line of a bi-variat scalar function to a polygon
 - Given: A uniform sampling (on a 2D grid) of the implicit function
 - For every grid cell determine the zero crossings
 - There are $2^4 = 16$ possible combinations



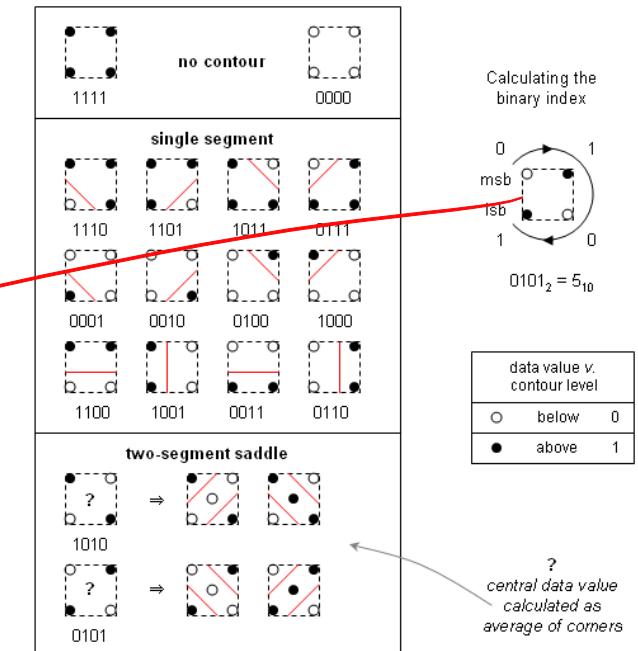
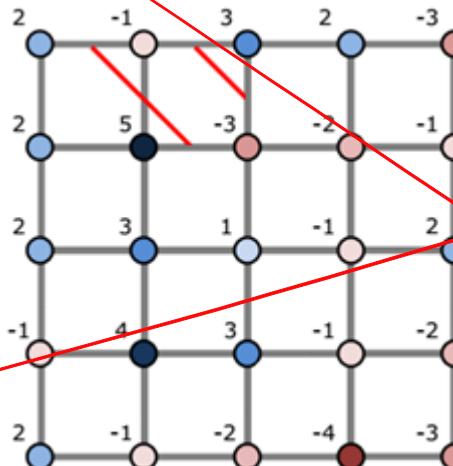
Marching Squares (2D)

- Converts an iso-line of a bi-variat scalar function to a polygon
 - Given: A uniform sampling (on a 2D grid) of the implicit function
 - For every grid cell determine the zero crossings
 - There are $2^4 = 16$ possible combinations



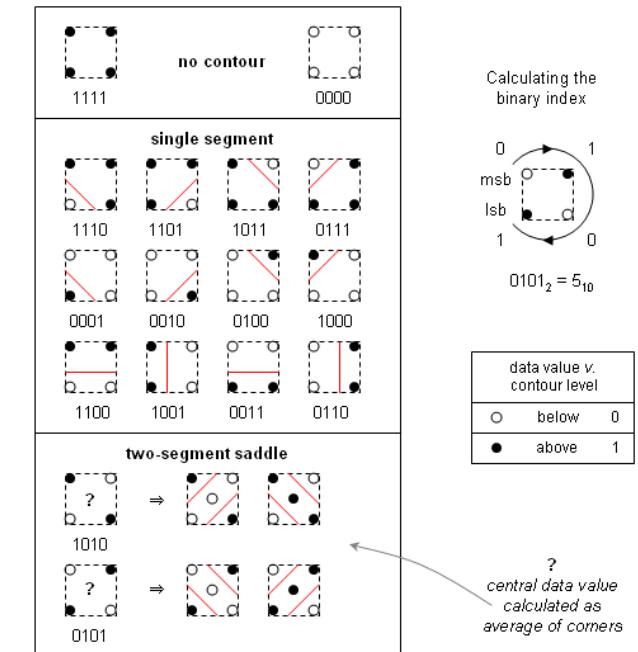
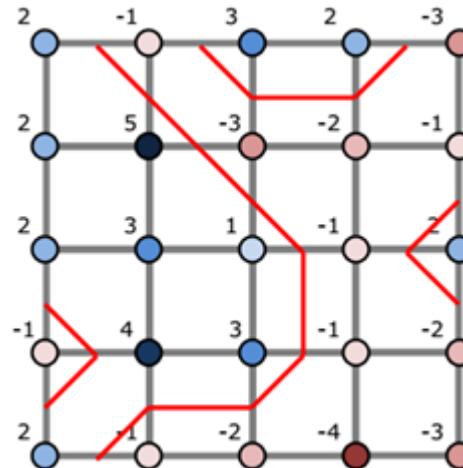
Marching Squares (2D)

- Converts an iso-line of a bi-variat scalar function to a polygon
 - Given: A uniform sampling (on a 2D grid) of the implicit function
 - For every grid cell determine the zero crossings
 - There are $2^4 = 16$ possible combinations



Marching Squares (2D)

- Converts an iso-line of a bi-variat scalar function to a polygon
 - Given: A uniform sampling (on a 2D grid) of the implicit function
 - For every grid cell determine the zero crossings
 - There are $2^4 = 16$ possible combinations



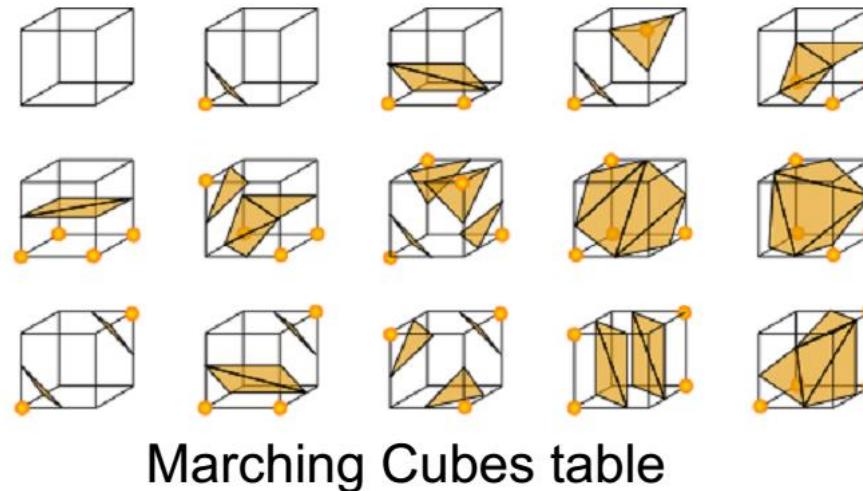
Marching Cubes (3D)

- Converts an iso-surface of a tri-variate scalar function to a polygonal mesh
 - Given: A uniform sampling (on a 3D grid) of the implicit function
 - For every grid cell determine the zero crossings
 - There are $2^8 = 256$ possible combinations
 - Use lookup table to find triangulation
 - Adjust vertex position according to linear interpolation

[Lorensen'87] W.Lorensen et al., "Marching cubes: A high resolution 3D surface construction algorithm"

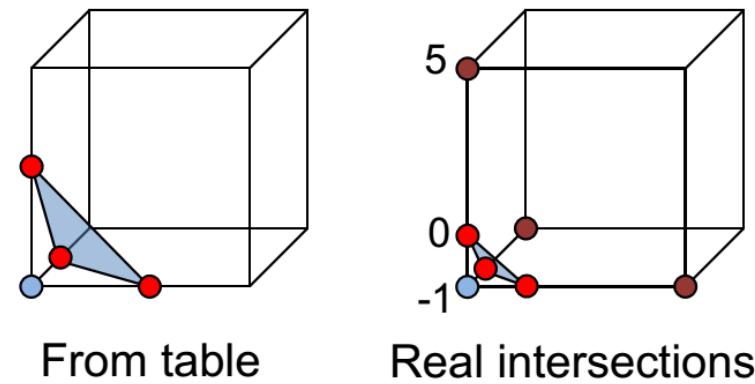
Marching Cubes (3D)

- Lookup table:



Marching Cubes table

- Linear interpolation:



Administrative

Next week:

- Overview of 3D reconstruction methods