Multi-level Partition of Unity Implicit

SIGGRAPH 2003

Yutaka Ohtake

mpi INFORMATIK

Alexander Belyaev

mpi INFORMATIK

Marc Alexa

TU Darmstadt

Greg Turk

Georgia Tech

Hans-Peter Seidel

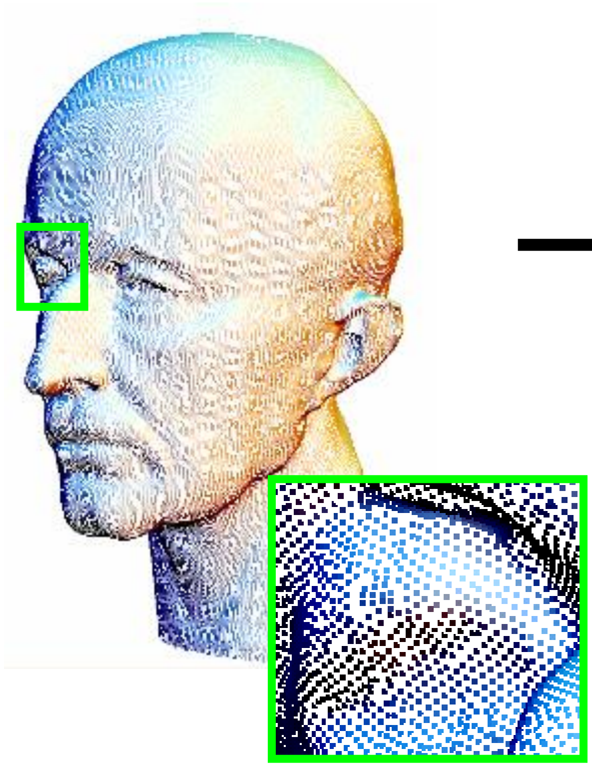
mpi INFORMATIK

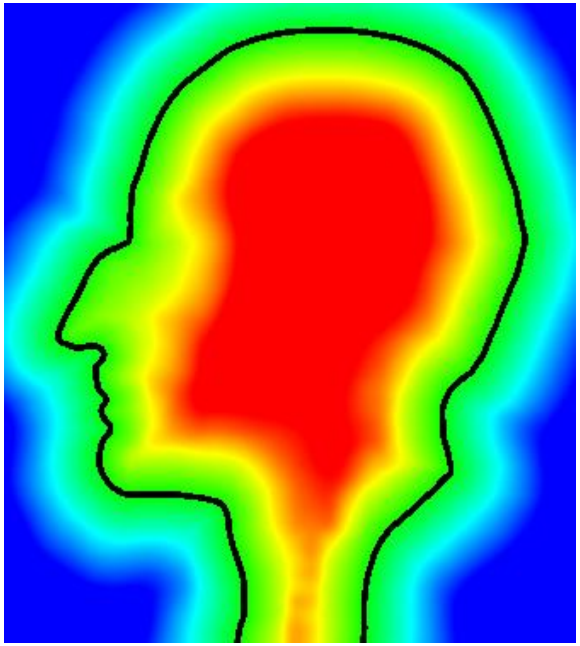
Outline

- Introduction
  - Problem
  - Issues
- Algorithm
  - Partition of Unity
  - Adaptive Octree Subdivision
  - Estimating Local Shape Functions
- Applications
- Performance
- Conclusion

Introduction - Problem

- Goal: Representing implicit solid as a function *f*
- Input: Points with Normals (typical output of range scanners)





f(x,y,z) > *0* inside
f(x,y,z) < *0* outside
f(x,y,z) = *0* approximates points

Signed distance.

Introduction - Problem

- Issues
  - Local vs. Global
    - (N/k of k x k) vs. (N x N)
  - Sharp Features
  - Error Control

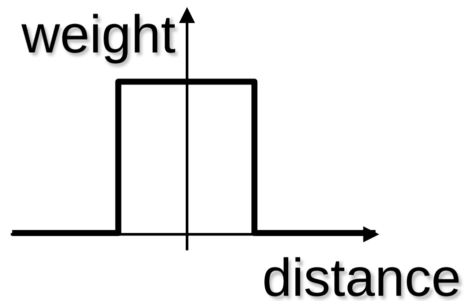
[Turk and O'brien 2002.]

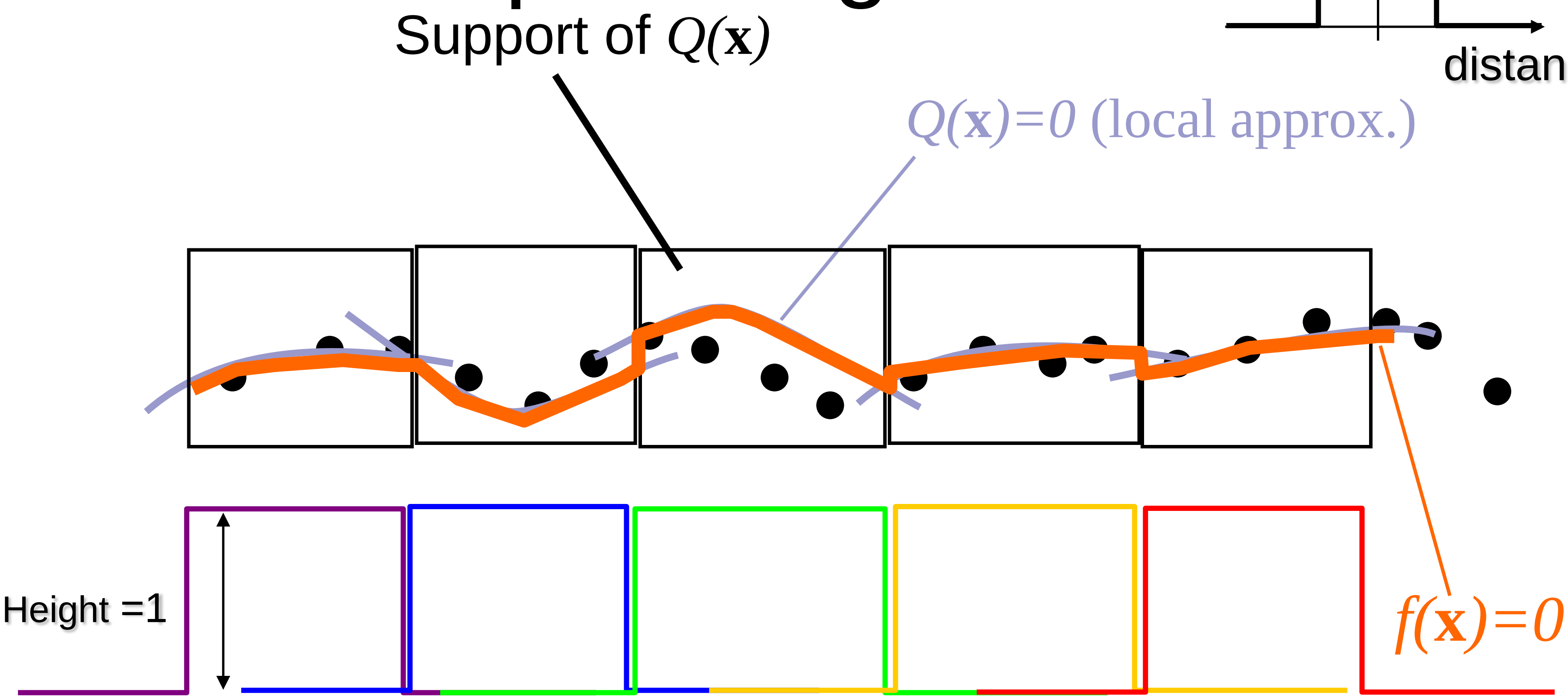
$$f(\mathbf{x}) = \sum_{j=1}^{k} w_j \phi(\mathbf{x} - \mathbf{c}_j) + P(\mathbf{x})$$

weight (w_j) basis ($\phi(\mathbf{x} - \mathbf{c}_j)$)

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1k} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \dots & \phi_{2k} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \dots & \phi_{kk} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \dots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \dots & c_k^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \dots & c_k^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \dots & c_k^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Function patching

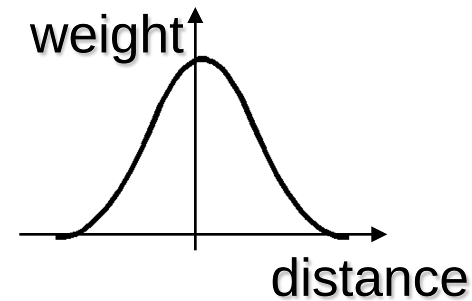


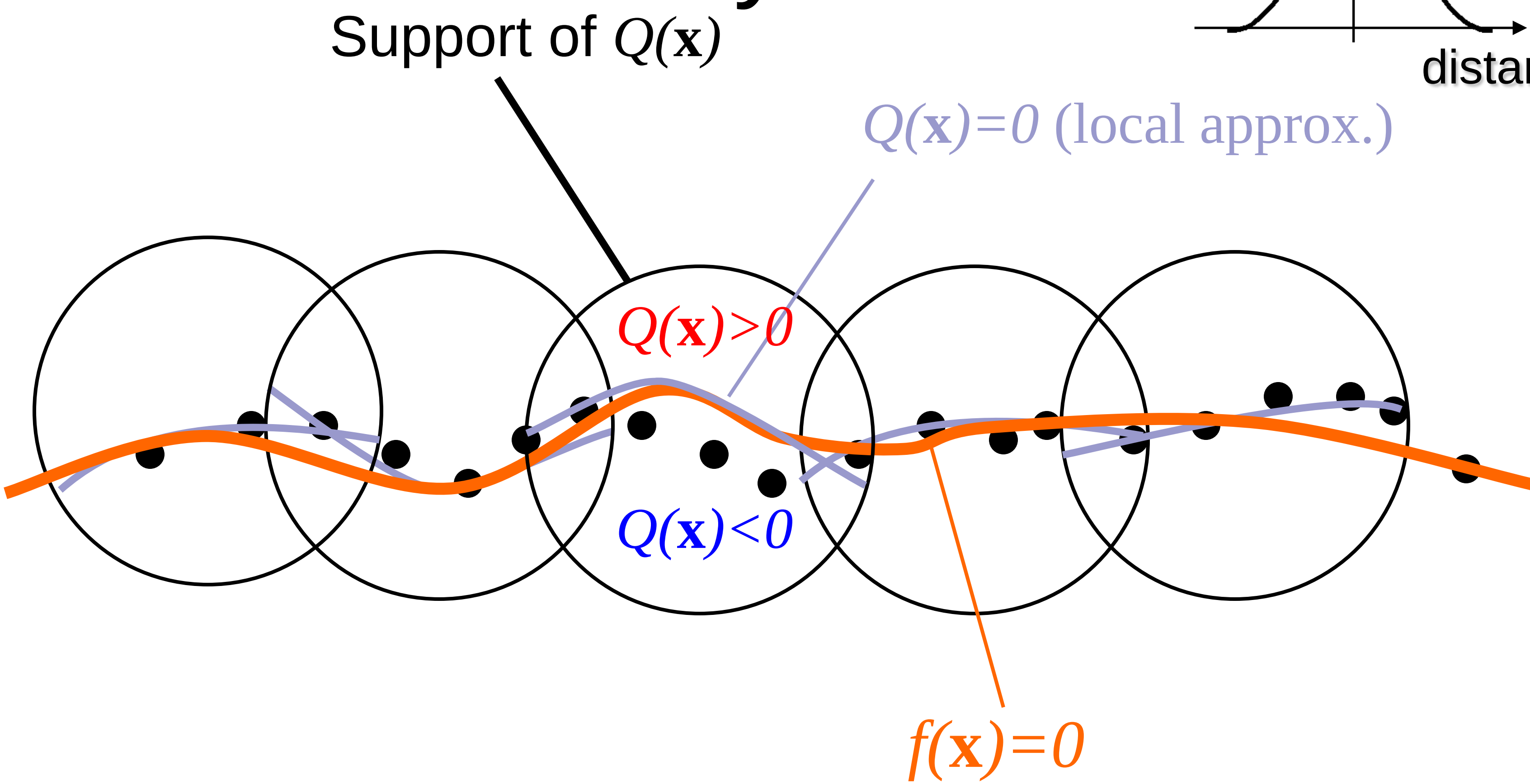


Weighted average of local approximations

$$f(\mathbf{x}) = \frac{\sum w_i(\mathbf{x}) Q_i(\mathbf{x})}{\sum w_i(\mathbf{x})}$$

Partition of Unity



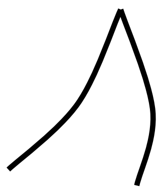


Weighted average of
local approximations

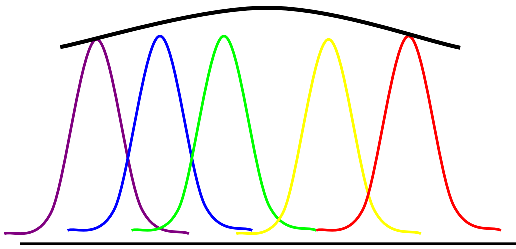
$$f(\mathbf{x}) = \frac{\sum w_i(\mathbf{x}) Q_i(\mathbf{x})}{\sum w_i(\mathbf{x})}$$

Approach of this paper

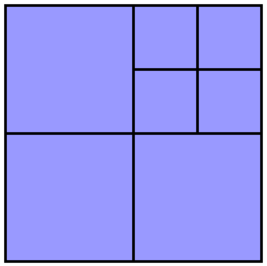
$$f(\mathbf{x}) = \sum w_i(\mathbf{x}) Q_i(\mathbf{x}) \Big/ \sum w_i(\mathbf{x})$$



- Piecewise quadratic local approximations $Q_i(\mathbf{x})$.

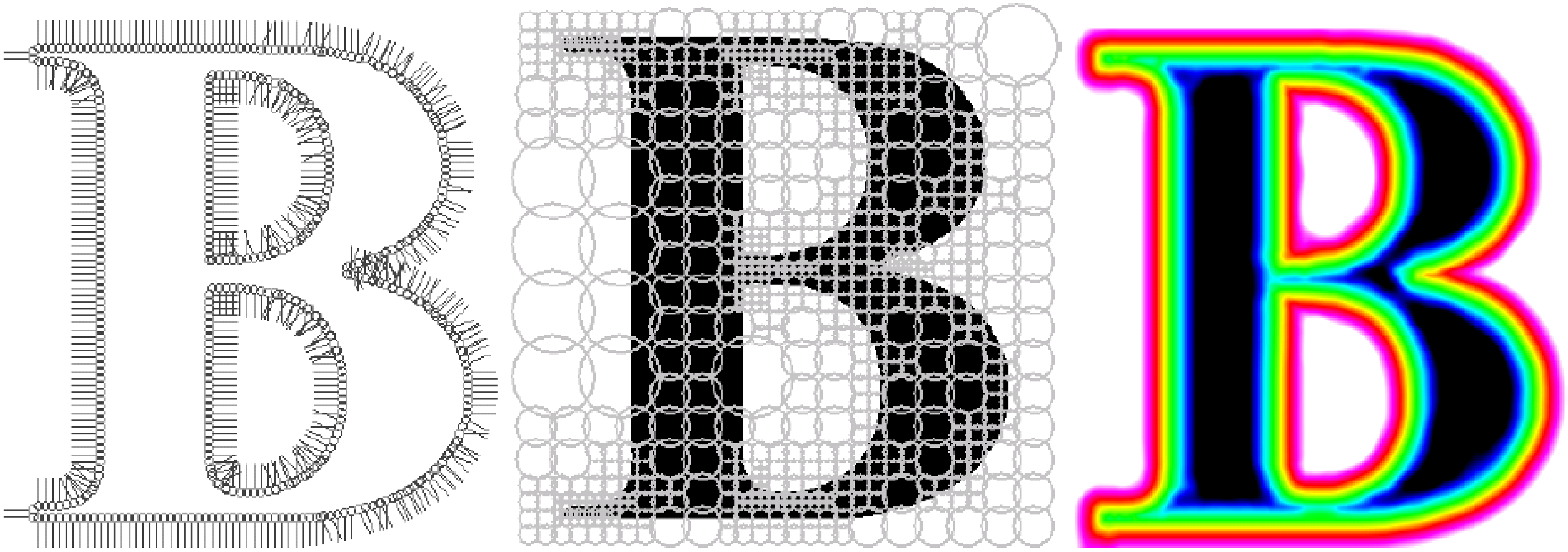


- Partition of unity $\left\{ w_i(\mathbf{x}) \big/ \sum w_i(\mathbf{x}) \right\}$
  - Used to blend local approximations
- Octree based multi-level structure
  - Adapted to geometrical complexity
  - Delivers an adaptive approximation of the distance-function
  - Allows a user to specify approximation accuracy



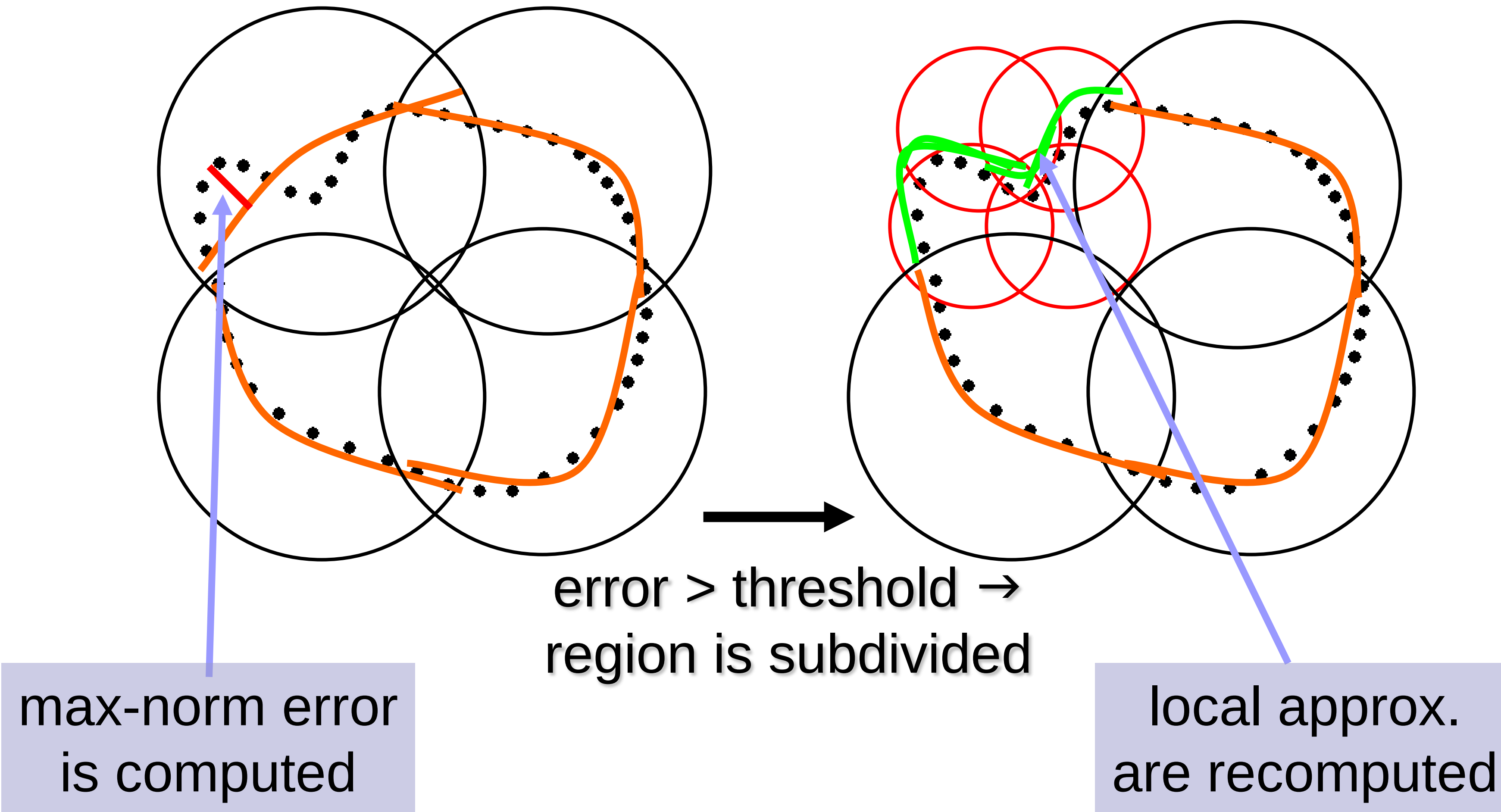
Octree and ball

- Balls proportional to cell size, center at c_i



Multi-level Partition of Unity

- Adaptive octree subdivision



Algorithm of MPU

- For cell i,
  - Fit Q_i
  - Calculate ε_i
  - If ($\varepsilon_i > \varepsilon_0$) subdivide the cell and re-compute
- Blending (assembling) all leaf Q_i 's using w_i's
  - $f(\mathbf{x}) = \sum w_i(\mathbf{x}) Q_i(\mathbf{x}) \Big/ \sum w_i(\mathbf{x})$
- Remaining problem: choices of Q_i and w_i

Weight functions

- For approximation
  - B-spline b(t)

$$w_i(\mathbf{x}) = b\left(\frac{3\left|\mathbf{x}-\mathbf{c}_i\right|}{2R_i}\right)$$

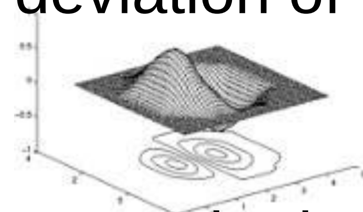
- For interpolation
  - Inverse-distance singular weights

$$w_i(\mathbf{x}) = \left[\frac{\left(R_i - \left|\mathbf{x}-\mathbf{c}_i\right|\right)_+}{R_i\left|\mathbf{x}-\mathbf{c}_i\right|}\right]^2, \text{ where } (a)_+ = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases}$$

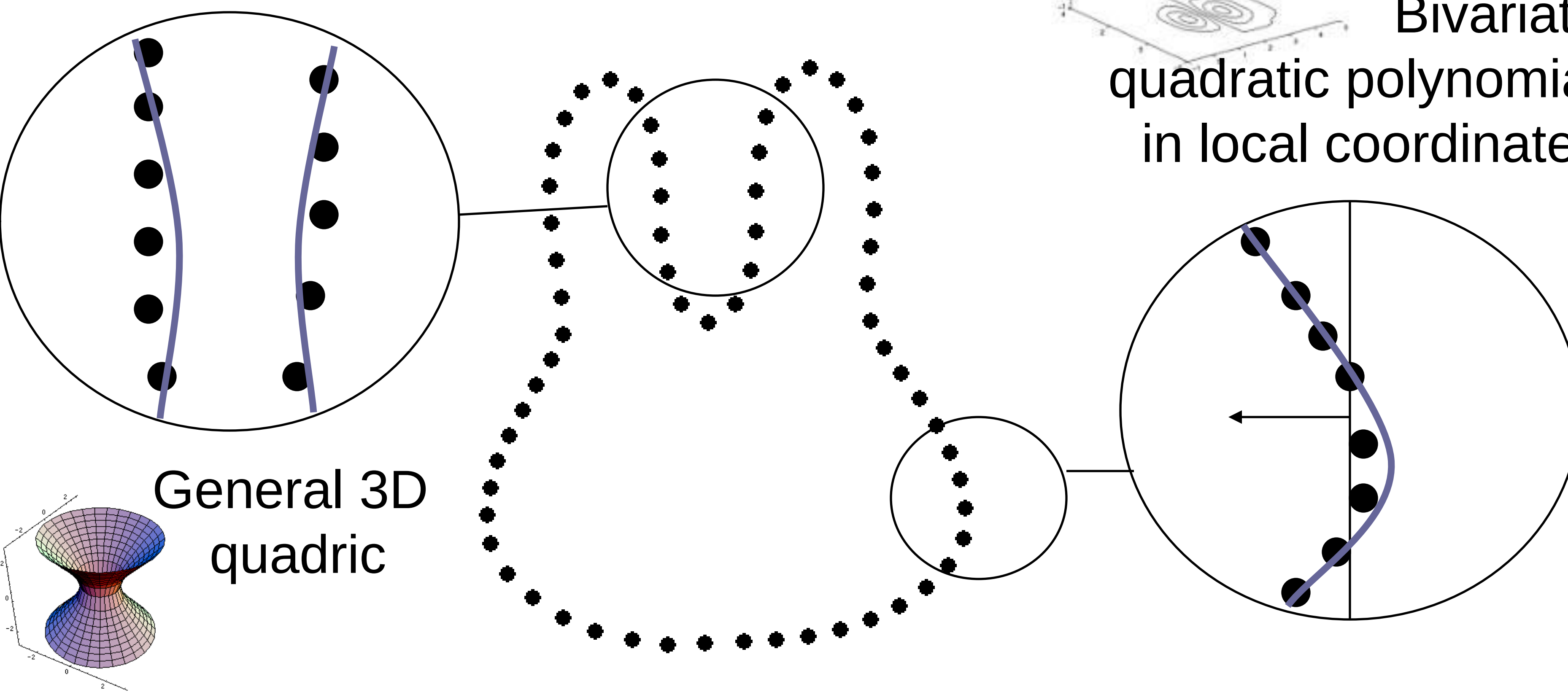
→infinity near $\mathbf{c}_i$

Local Shape Function (Q_i)

- Second-order polynomial approx. by least square fitting
- Approximation type: according to the deviation of normals.



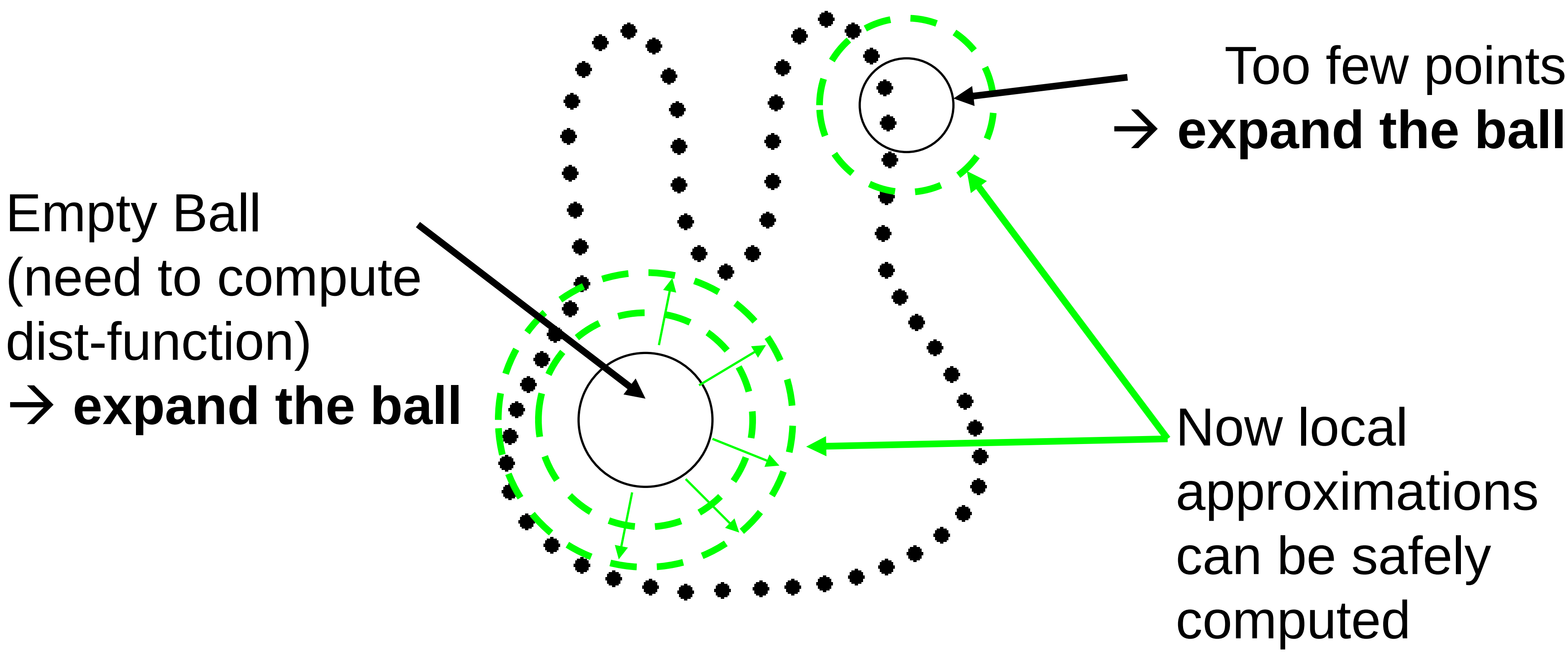
Bivariate quadratic polynomial in local coordinates



General 3D quadric

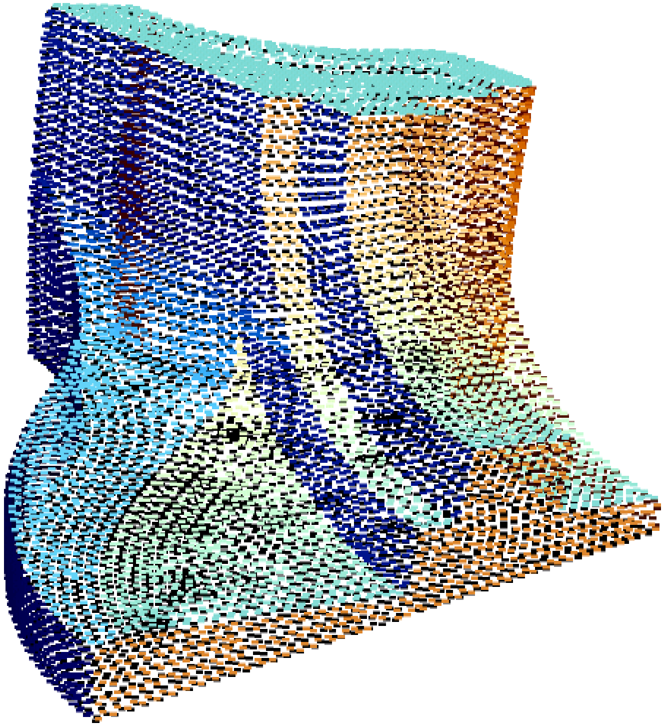
Local Shape Function (Q_i)

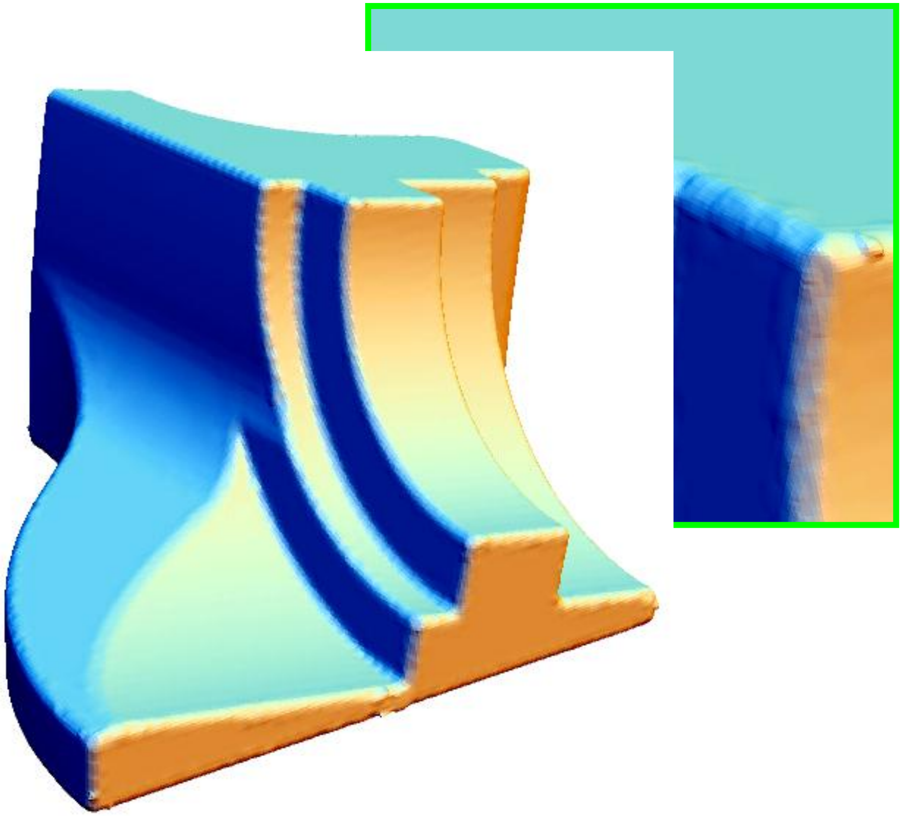
- Expand balls to include sufficient number of points.



Sharp Features

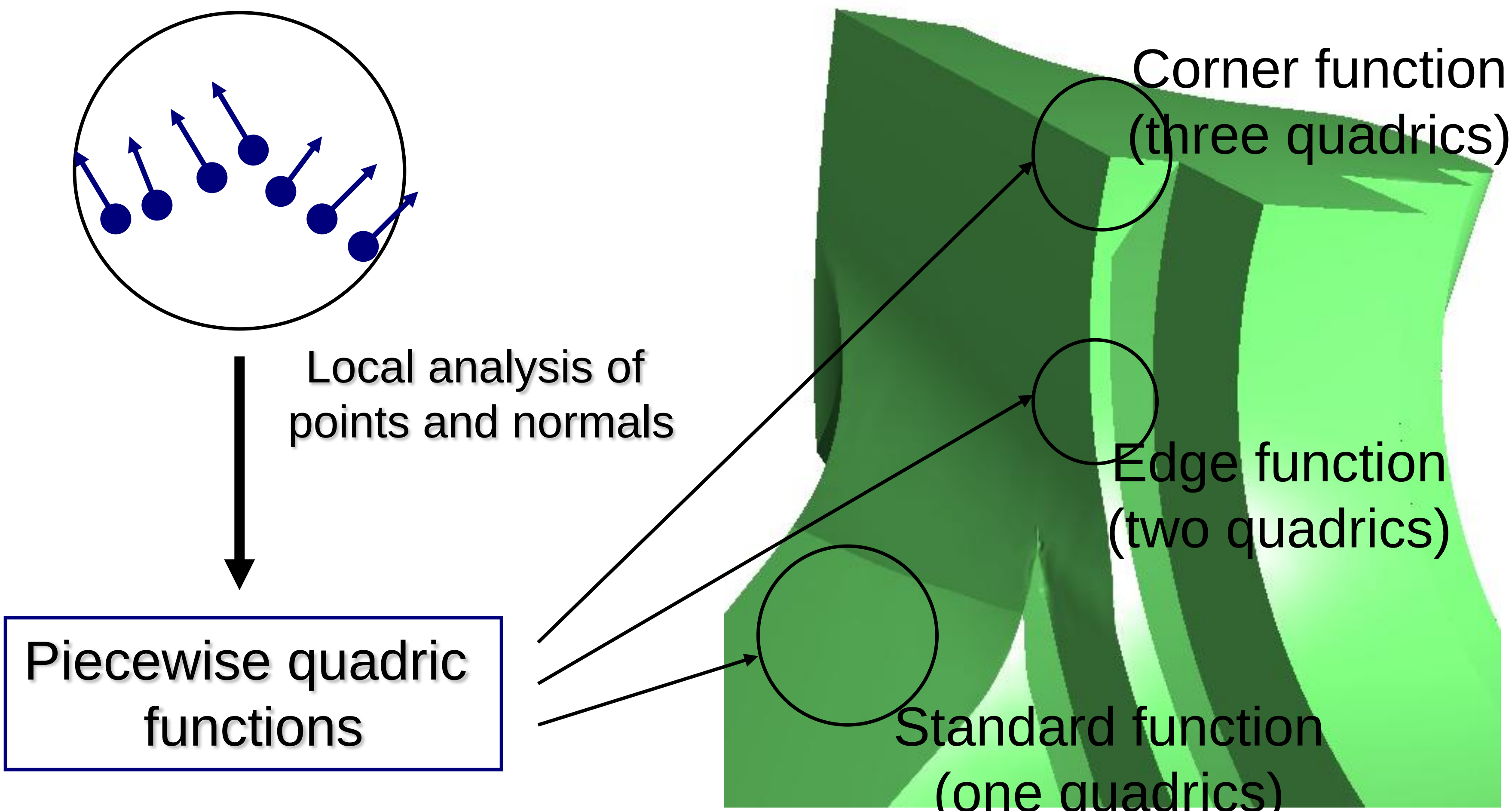
- Quadrics → Impossible to represent sharp features





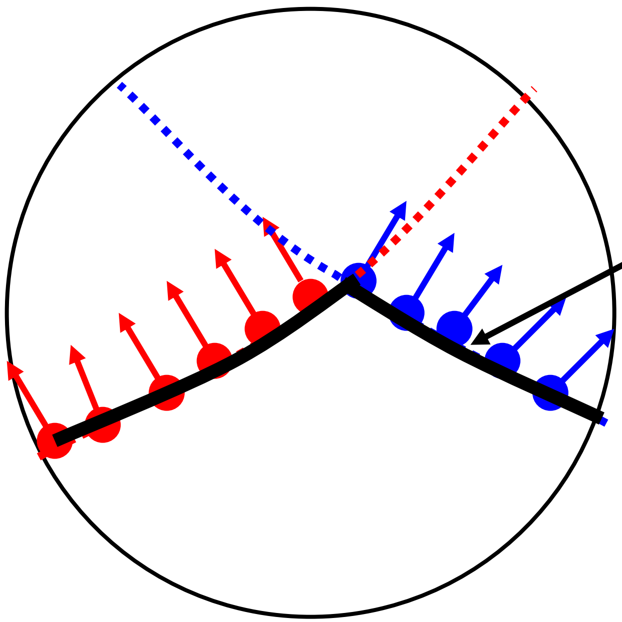
Sharp Features

- Use piecewise smooth local approximations



Sharp Features

- Edge: most deviated n_1, n_2
- Corner: highly deviated from $n_3 = n_1 \times n_2$



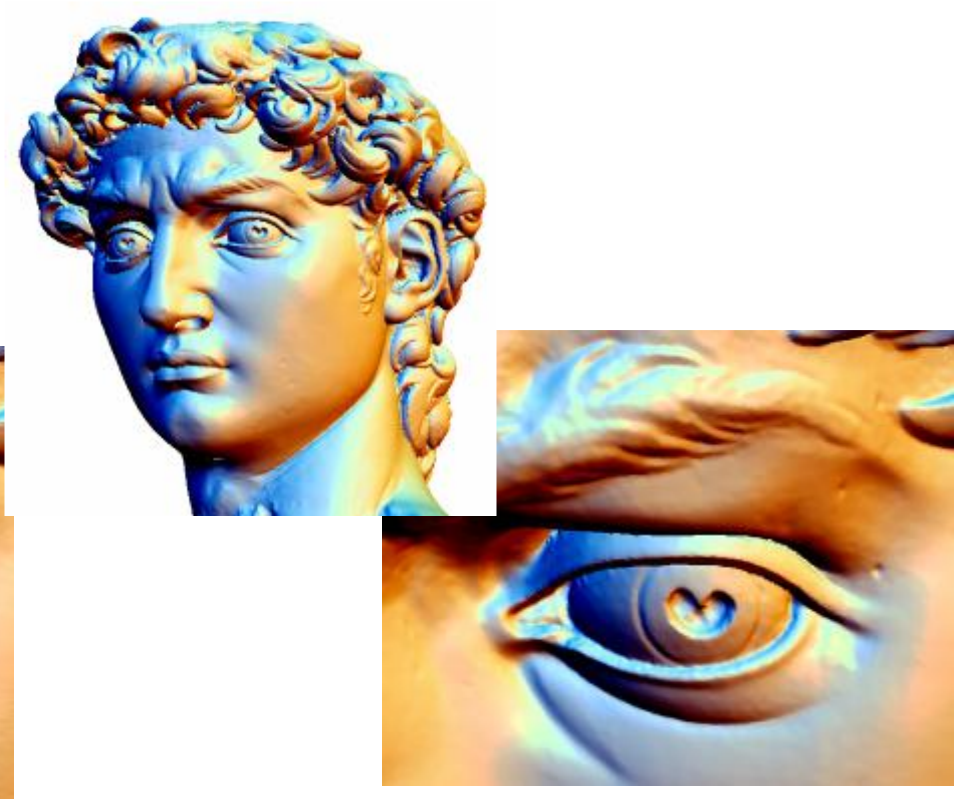
max/min Boolean operations
→ piecewise smooth local approximations

Clustering Normals
→ Clustring Points

Accuracy Control

- For visualization purposes 0.01% accuracy is sufficient.

Original mesh
(David head 1mm)



Approximation by MPU
with 0.01% accuracy

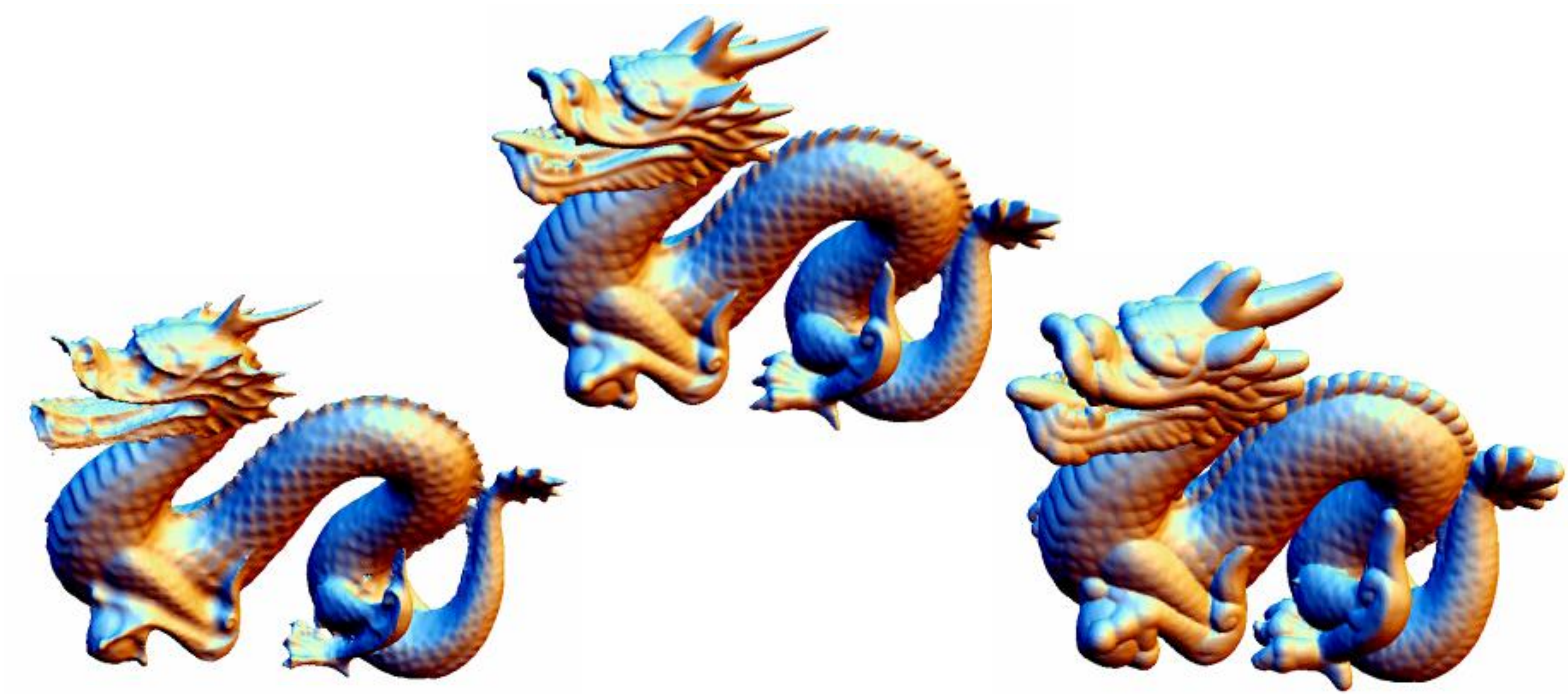
Applications – Geometric operations

Boolean operations

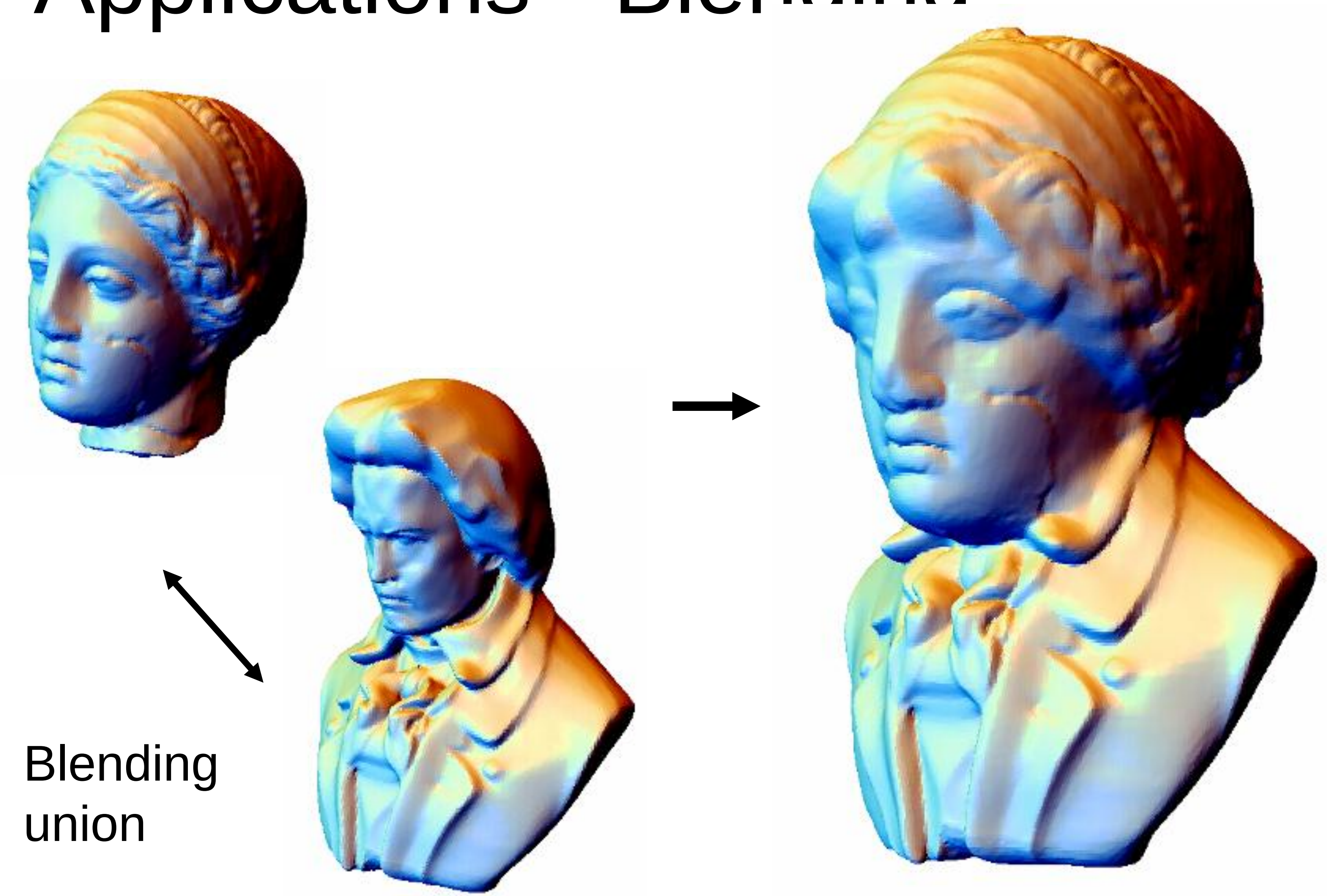
Space transformation

Applications - offsetting

- If f is a good approx. of signed distance.



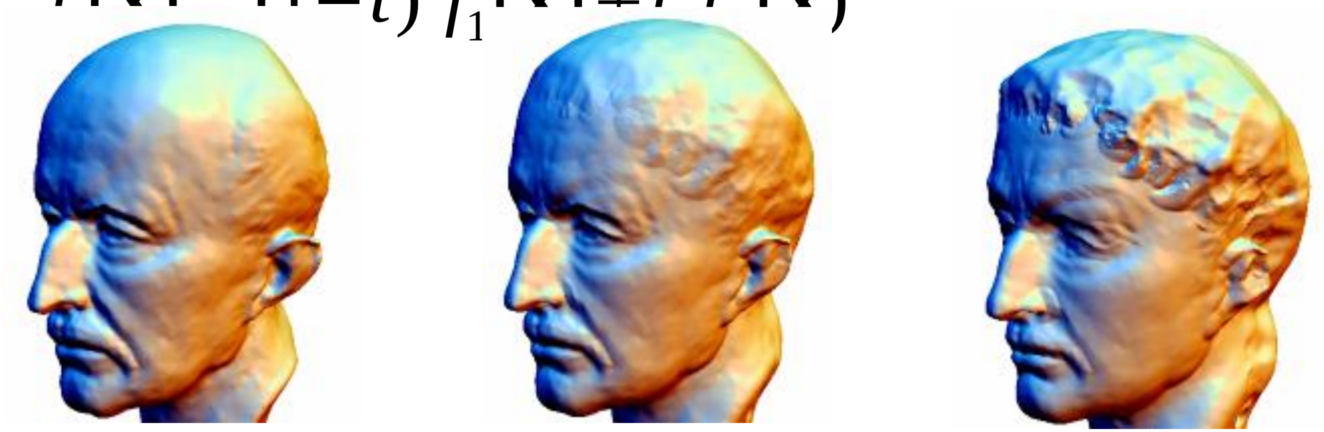
Applications - Blending



Blending union

Applications - Morphing

$$f(\mathbf{x}) = (1-t)\, f_1(\mathbf{x}) + t\, f_2(\mathbf{x})$$

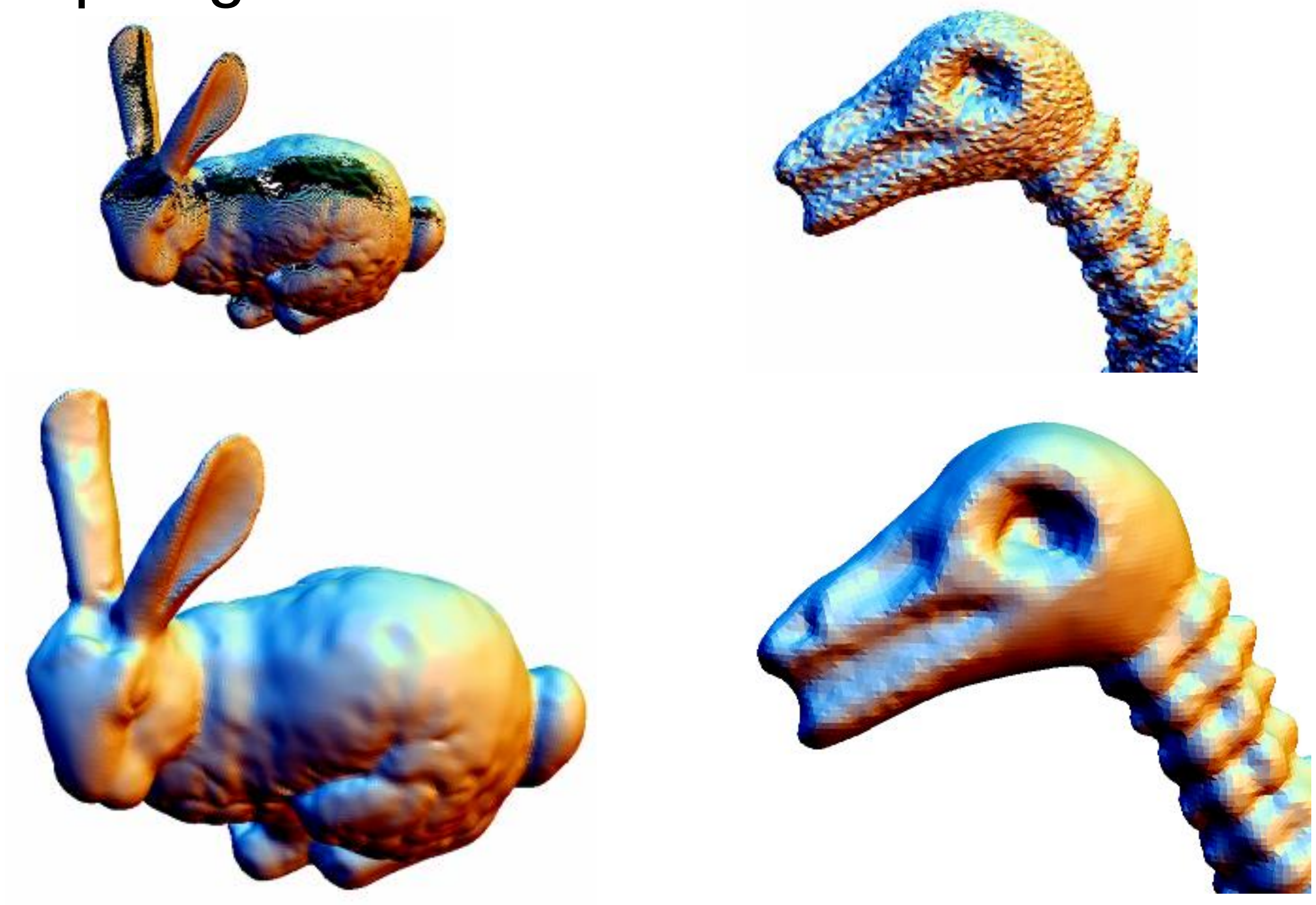


$f_1(\mathbf{x}) = 0$

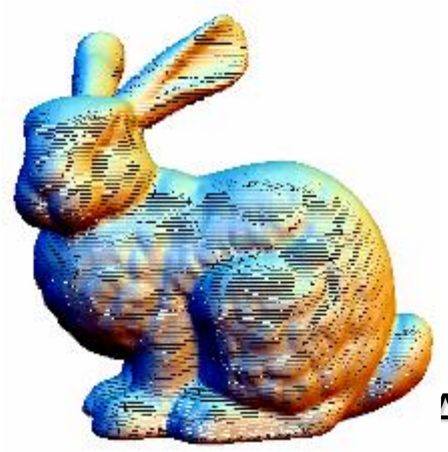
$f_2(\mathbf{x}) = 0$

Applications – Filling, Smoothing

- No topological restrictions



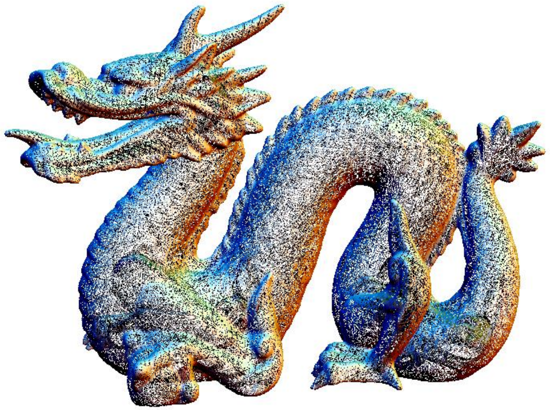
Performance



RAM: 34 MB
Time: 7 sec.
(1.6 GHz P4)
Accuracy: 0.25%

35K points

91K triangles



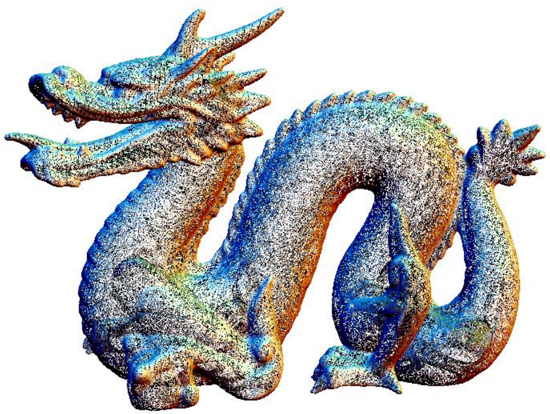
RAM: 195 MB
Time: 99 sec.
(1.6 GHz P4)
Accuracy: 0.08%

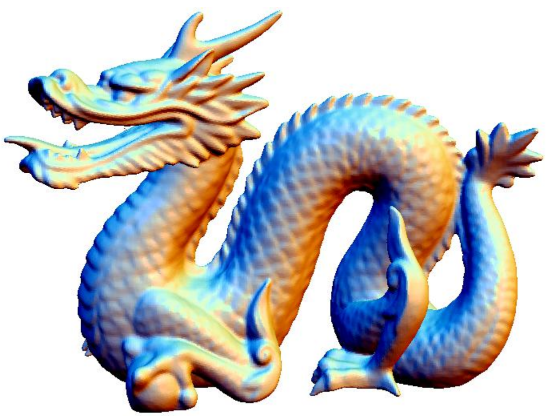
433K points

820K triangles

Performance

Reconstruction
with 0.08% accuracy





MPU

[Carr et al. SIG01]

RAM: 195 MB
Time: 99 sec.
(Pentium4 1.6 GHz)

$\times 100$

$\times 3$

RAM: 306MB
Time: 170 min.
(Pentium3 550 MHz)

Conclusion

- A new implicit representation for 3D scattered point data
  - Easy to implement
  - Fast reconstruction
  - Can handle a very large data
  - Can represent sharp features
  - Good for function-based modeling