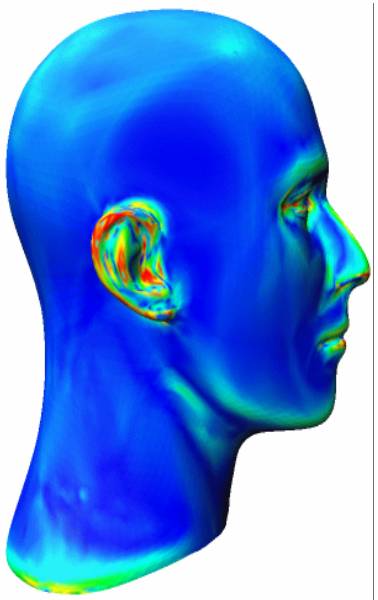
15-864 Assignment 2: Mesh Smoothing



Overview

In this short assignment you will get some final practice performing operations on polygon meshes. You may also find these techniques useful for creating meshes for future projects.

Details

1. Implement an explicit solver for both diffusion- (using the Laplacian operator) and curvature-based smoothing on meshes, as described in [Desbrun et al. 1999]. You may use an explicit forward Euler method to solve the equation (as described in the paper); be cogent of the timestep restriction, however.
2. Provide the ability to change both the lambda value and the number of iterations via either command-line arguments or interactive keystrokes. Document this in your README file.
3. You should already have a viewer capable of displaying polygon meshes from the first project, as well as an .obj reader. Present the results of the smoothing in this viewer. Also, as an intuitive way to aid in debugging, display update values (e.g., scaled Laplacian or curvature) at the vertices. For example, the head at right uses the Matlab Jet colormap to visualize curvature values. The color scheme you use can be anything you want, provided that it is clear where the high/low values are.
4. As a final step, **compare** the two smoothing approaches (curvature flow and laplacian smoothing) by showing a side-by-side comparison after N iterations for a few values of N (a simple jpeg image is sufficient here). For extra coolness factor, optionally turn in a side-by-side *video* of the two approaches iterating.
5. At a minimum, your program should work with the following meshes:
6. As usual, turn in your full source code, plus a README summarizing build instructions, interface details, etc.

Additional tips

- If you can't recall how to control materials in OpenGL, Nat Robins has a refresher.
- Take a look at this page for some useful library pointers.
- If you're (still?) having trouble logging into the lab machines, please see this page.

References

1. Gabriel Taubin. *A Signal Processing Approach to Fair Surface Design*. *Proceedings of SIGGRAPH 95*. pp. 351-358, 1995.
2. Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. *Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow*. *Proceedings of SIGGRAPH 99*. pp. 317-324, 1999.

Implicit Extras

Distance Fields + Sharp Features

- Problem: Distance fields (DFs) sampled at finite resolution can not recover sharp features
- Consider two related solutions:
  - Adaptively Sampled Distance Fields (ADFs)
    - [Frisken et al., 2000]
  - Extended Marching Cubes
    - [Kobbelt et al, 2001]

Adaptively Sampled Distance Fields (ADFs)

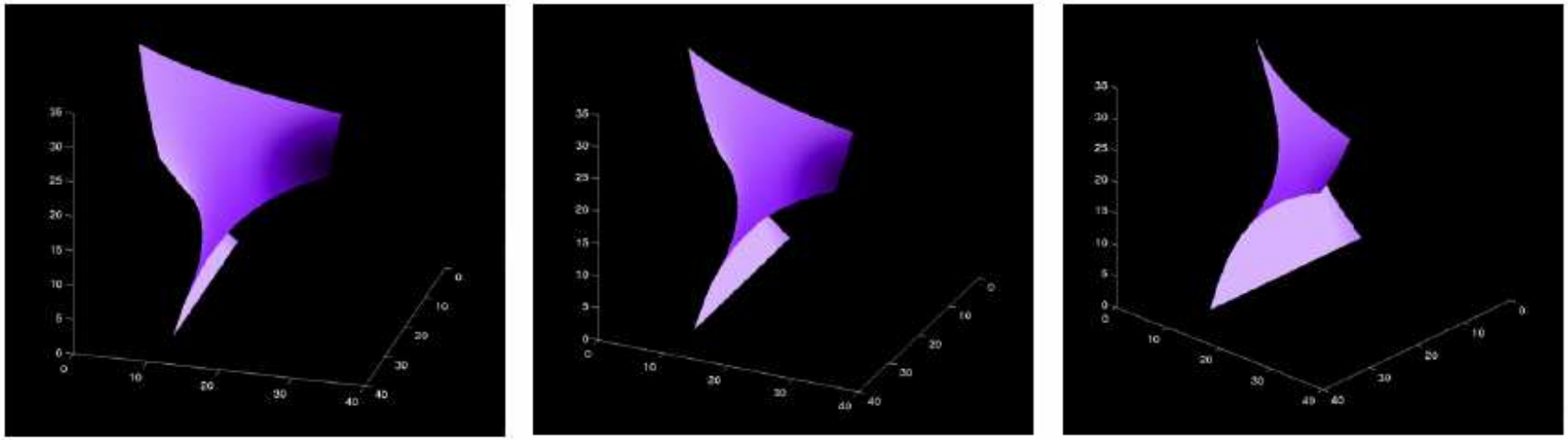
Sarah Frisken, Ronald Perry, Alyn Rockwood and Thouis Jones,

SIGGRAPH 2000

Adaptively Sampled Distance Fields (ADFs)

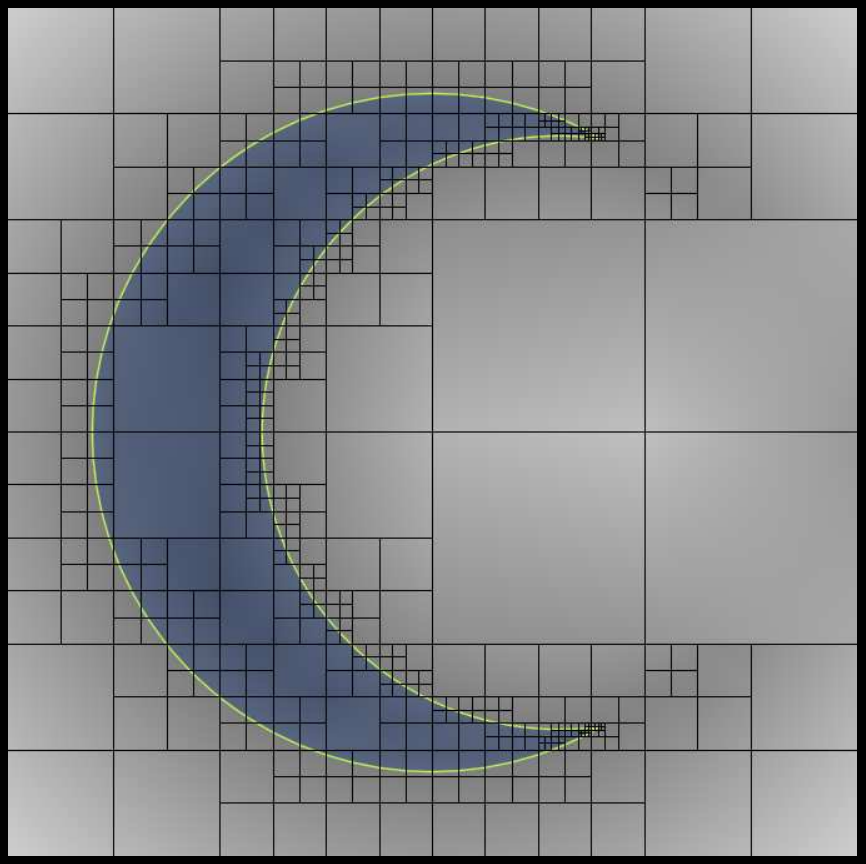
- Detail-directed sampling
  - High sampling rates only where needed
- Spatial data structure (e.g., an octree)
  - Fast localization for efficient processing
- Reconstruction method (e.g., trilinear interpolation)
  - For reconstructing the distance field and its gradient from the sampled distance values

Reconstruction



A single trilinear field can represent highly curved surfaces

Advantages of ADFs



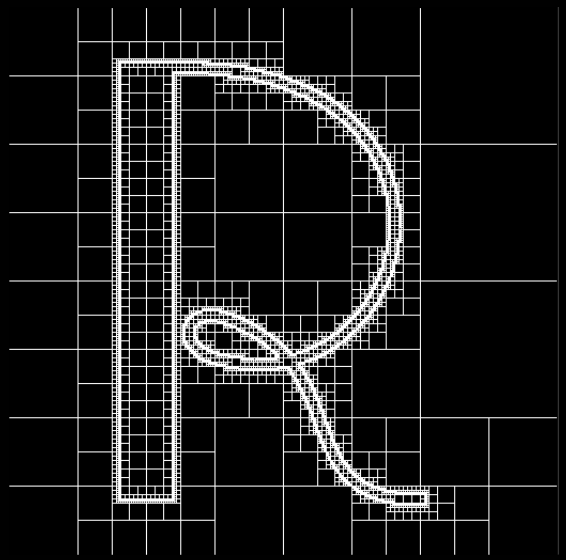
ADFs consolidate the data needed to represent complex objects

ADFs provide

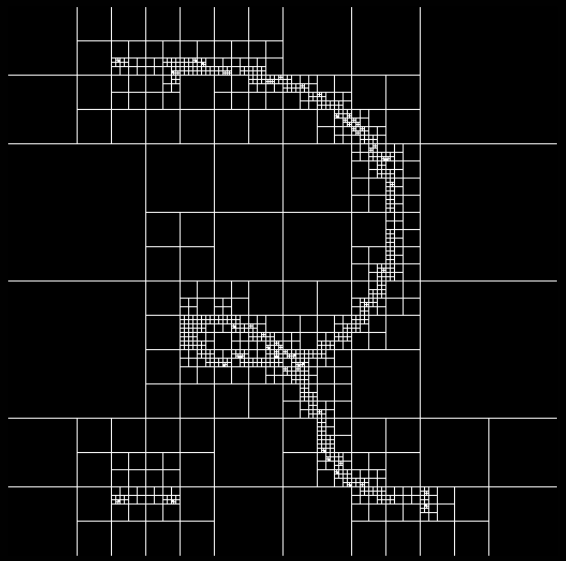
- Spatial hierarchy
- Distance field
- Object surface
- Object interior
- Object exterior
- Surface normal (gradient at surface)
- Direction to closest surface point (gradient off surface)

Comparison of 3-color Quadtrees and ADFs

- Fewer distance computations
- Smaller memory footprint



23,573 cells (3-color)



1713 cells (ADF)

ADF Examples

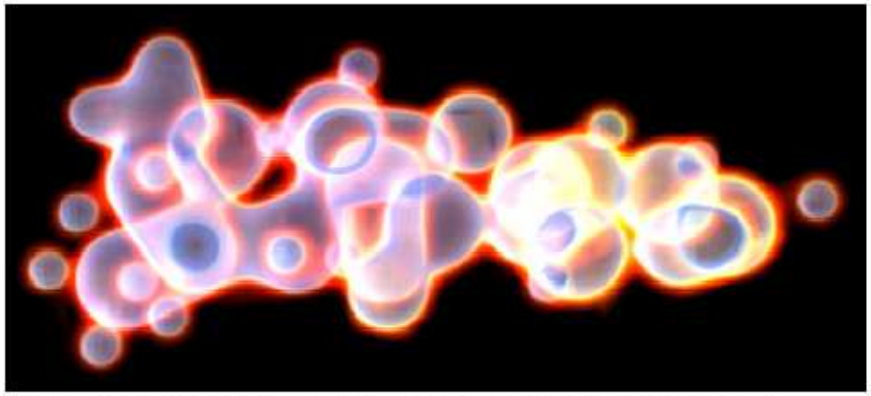
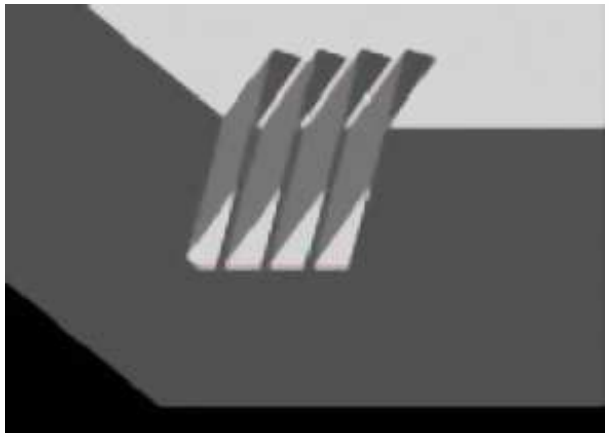


Figure 7. An ADF cocaine molecule volume rendered in a haze of turbulent mist. The mist was generated using a color function dependent on distance from the molecule surface.



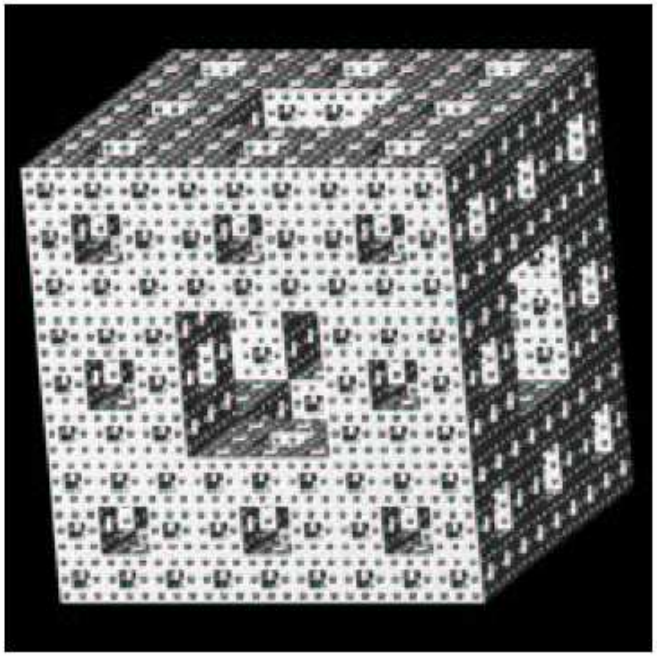


Figure 8. An ADF of the Menger Sponge, a fractal created recursively by subtracting smaller and smaller crossing cuboids from an initial cube. Four levels of recursion are shown.

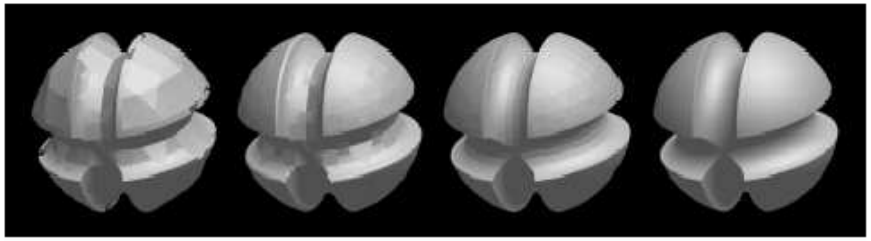
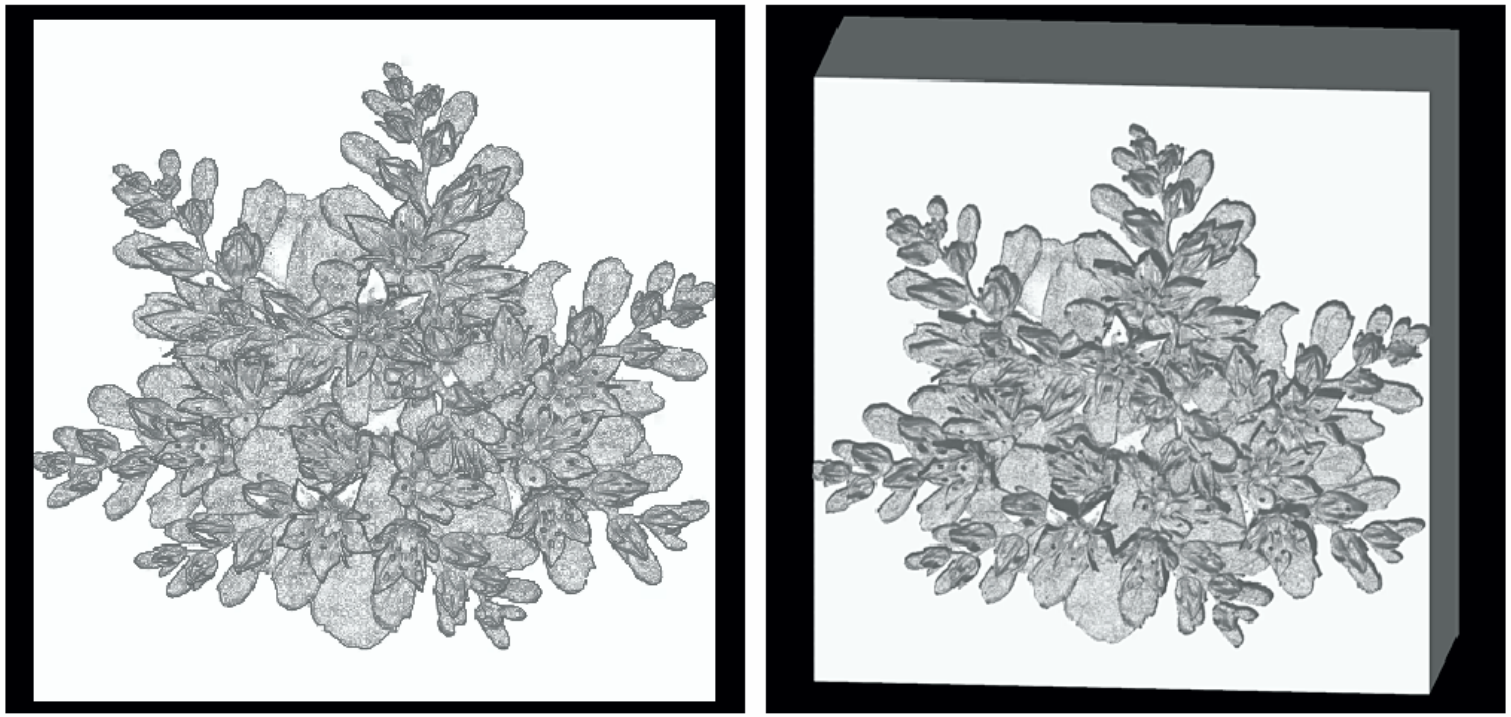


Figure 9. Four LOD models with varying amounts of error rendered from an ADF octree.

ADF Examples



Figures 5B. A bas-relief carving on a slab. The 3D geometry was generated from a black and white image of a flower (photograph courtesy of John Arnold) using the algorithm described in Section 7.1 for converting range data to an ADF. The highly detailed carving is represented as a level-10 ADF requiring 186 MB of storage.

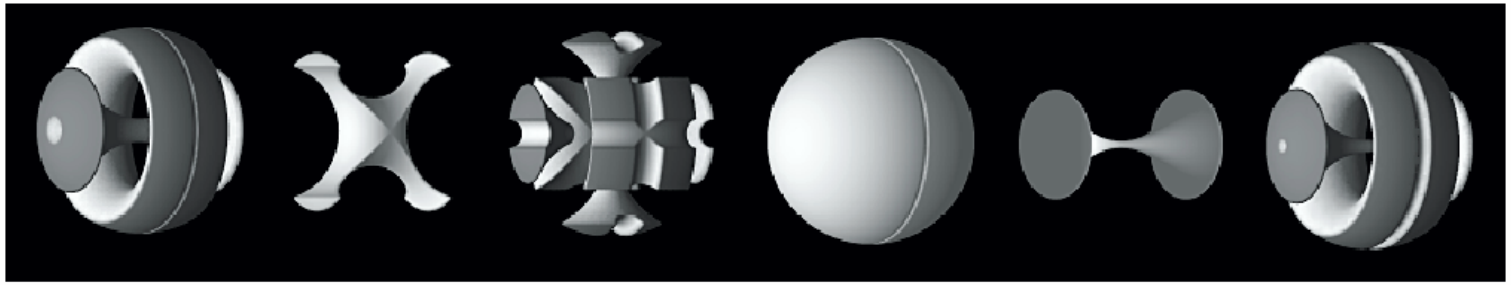


Figure 5C. Several sculpted level-8 ADFs showing how well ADFs represent both smooth surfaces and fine detail.

Feature Sensitive Surface Extraction from Volume Data

Leif Kobbelt et al.,

SIGGRAPH 2001

Limitations of Marching Cubes

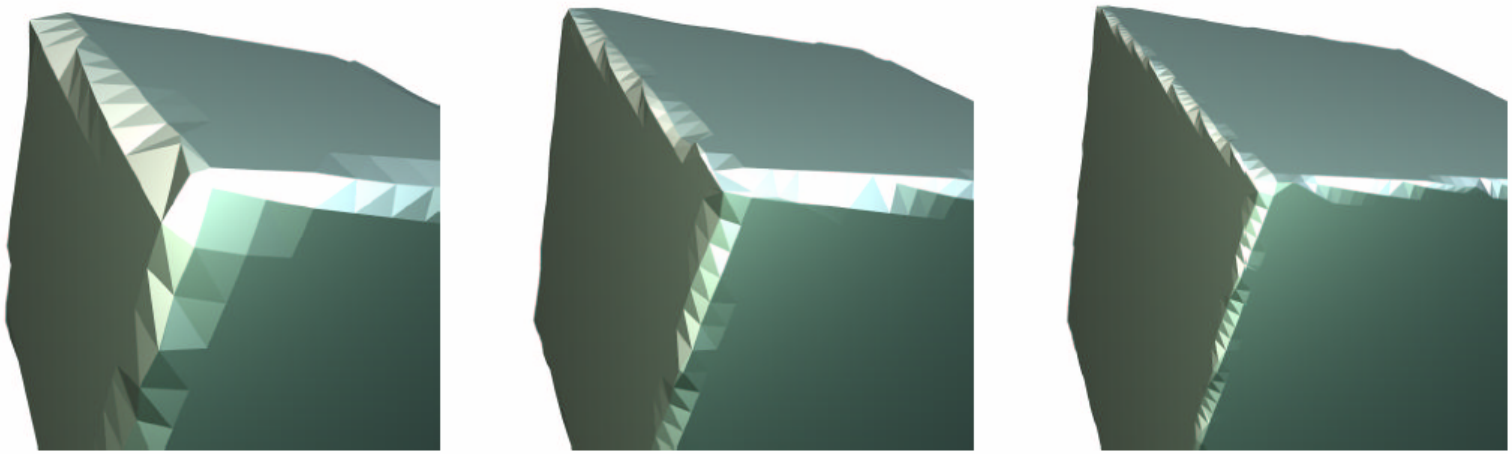


Figure 6: Alias errors in surfaces generated by the Marching Cubes algorithm are due to the fixed sampling grid. By decreasing the grid size, the effect becomes less and less visible due to the convergence of S^* to S but the problem is not really solved since the normal vectors of S^* do not converge to the normals of S.

Extending Marching Cubes

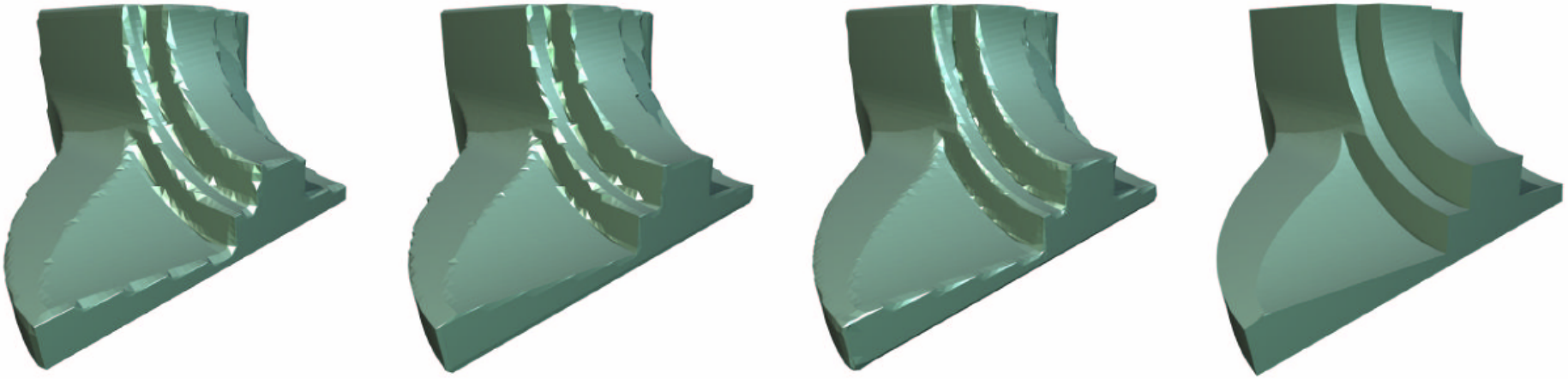
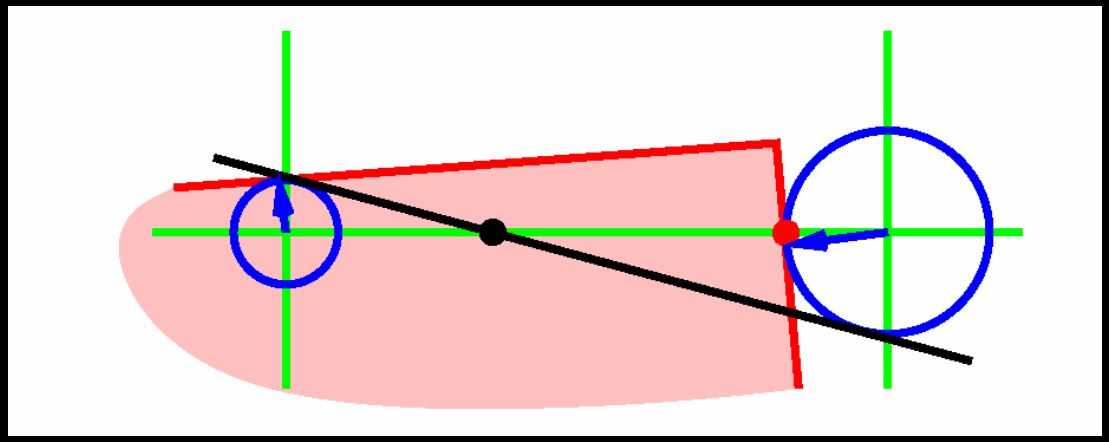


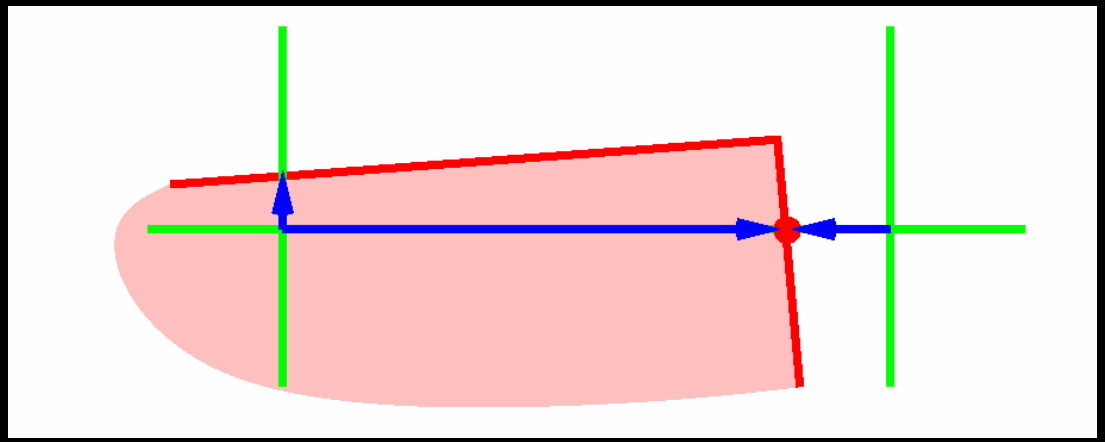
Figure 1: We present a new technique to extract high quality triangle meshes from volume representations of geometric objects. The two main contributions are an *enhanced distance field representation* and an *extended Marching Cubes algorithm*. The above figures show reconstructions of the well-known "fandisk" dataset from its distance field representation. The distance field has been sampled on a uniform $65 \times 65 \times 65$ grid. The far left image shows the standard Marching Cubes reconstruction, center left is the reconstruction by the same algorithm but applied to the enhanced distance field with the same resolution. Center right shows the result of our new extended Marching Cubes algorithm applied to the original volume data, and finally on the far right we show the reconstruction by our new algorithm applied to the enhanced distance field. The approximation error to the original polygonal model is below 0.25 %.

Directed Distance Fields

- Scalar DFs



- Directed DFs



Using Tangent Information

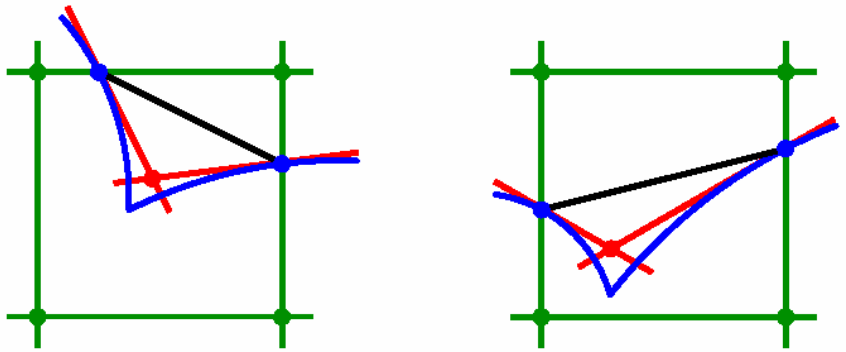


Figure 7: By using point and normal information on both sides of the sharp feature one can find a good estimate for the feature point at the intersection of the tangent elements.

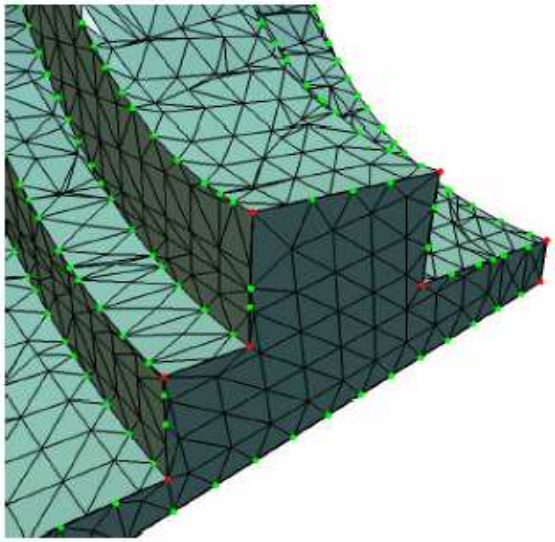


Figure 8: When inserting additional feature samples in some cells during the extended Marching Cubes we distinguish between different types of feature configurations: *edge features* are shown in green and *corner features* in red.

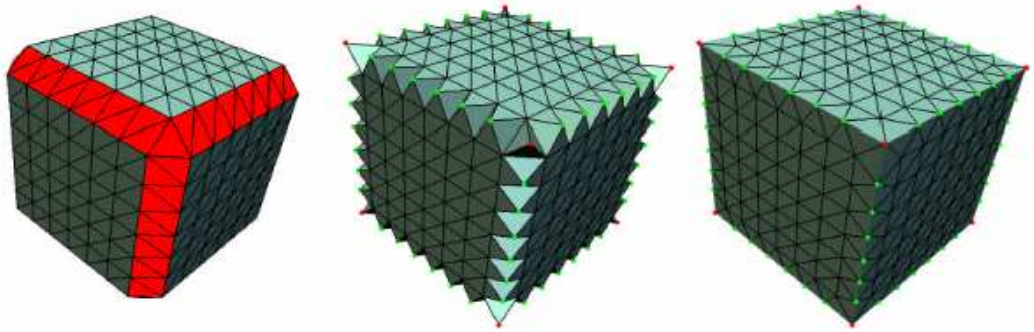
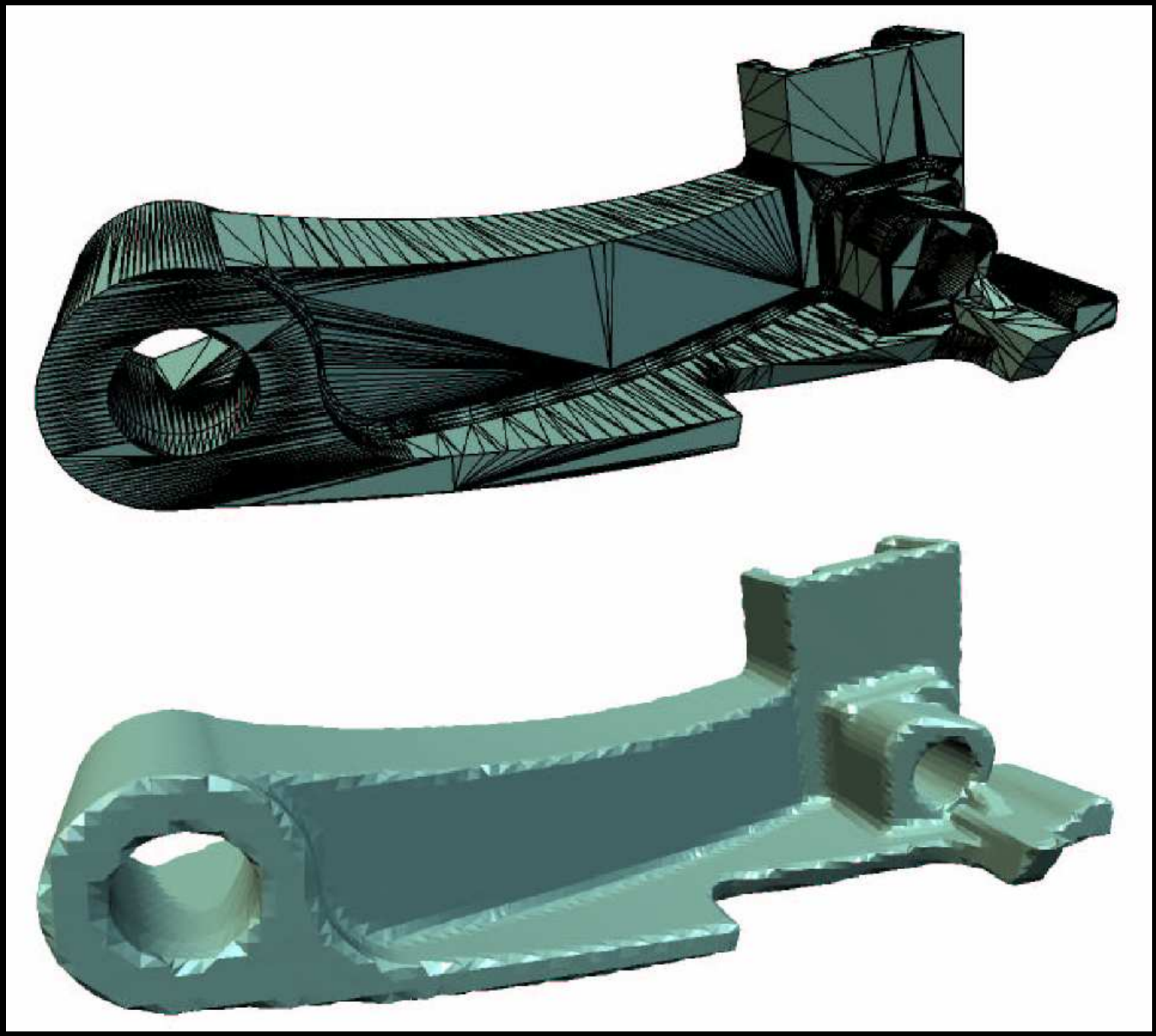
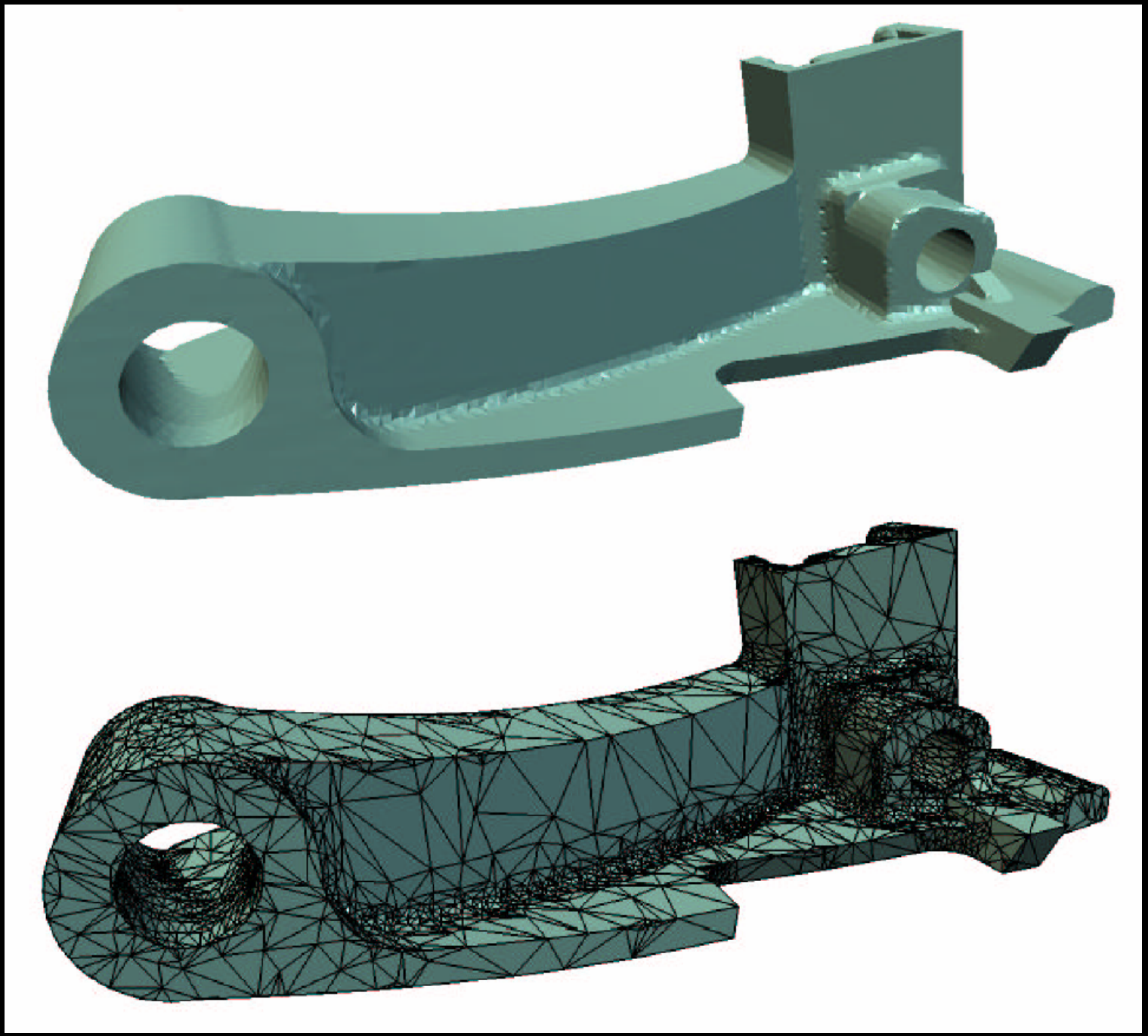


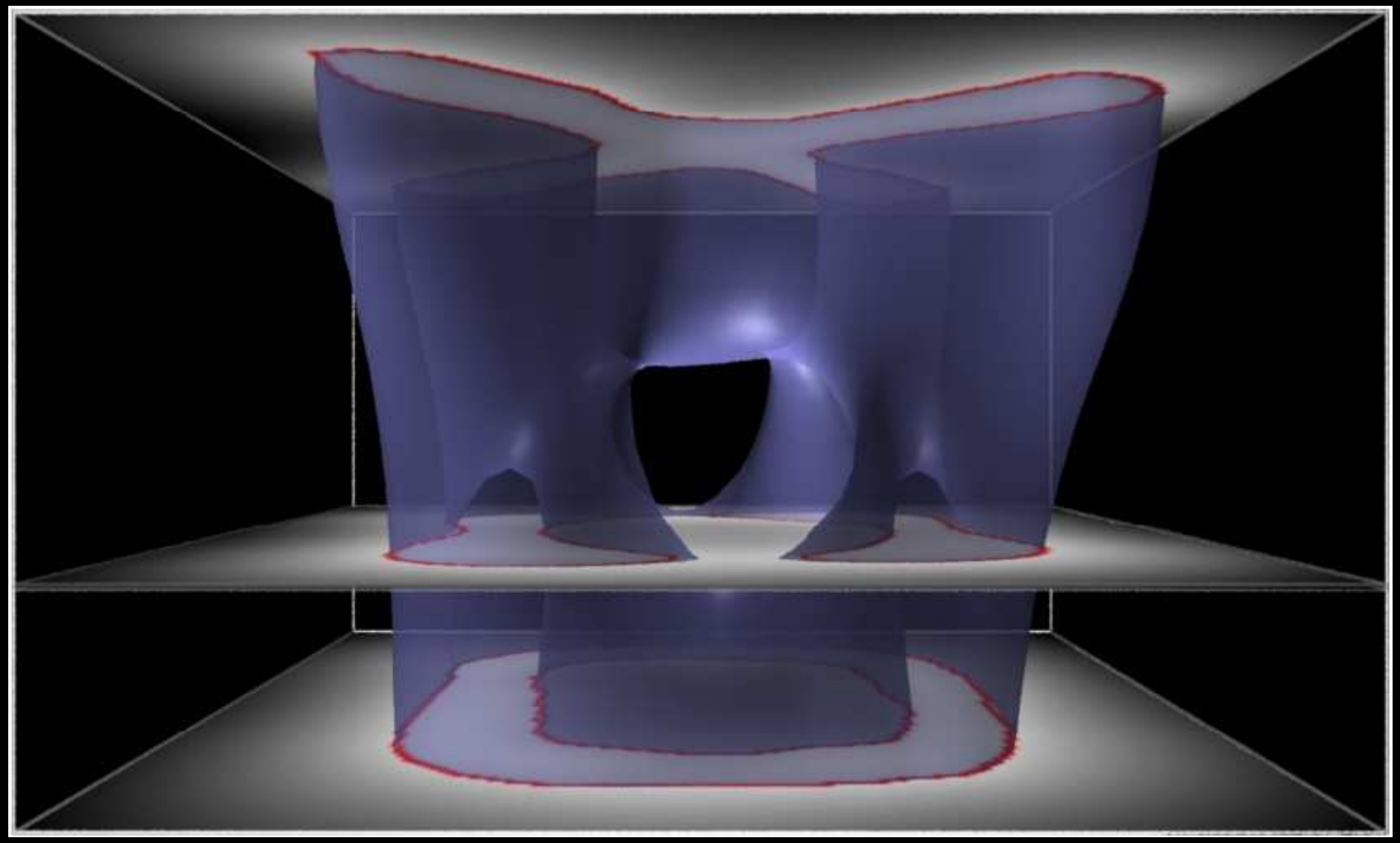
Figure 9: The feature sensitive sampling in the extended Marching Cubes algorithm works in three steps. First, the cells/patches that contain a feature are identified (left). Then *one* new sample is included per cell (center) and finally one round of edge flipping reconstructs the feature edges.





Shape Transformation Using Variational Implicit Functions

Turk + O’Brien, SIGGRAPH 99



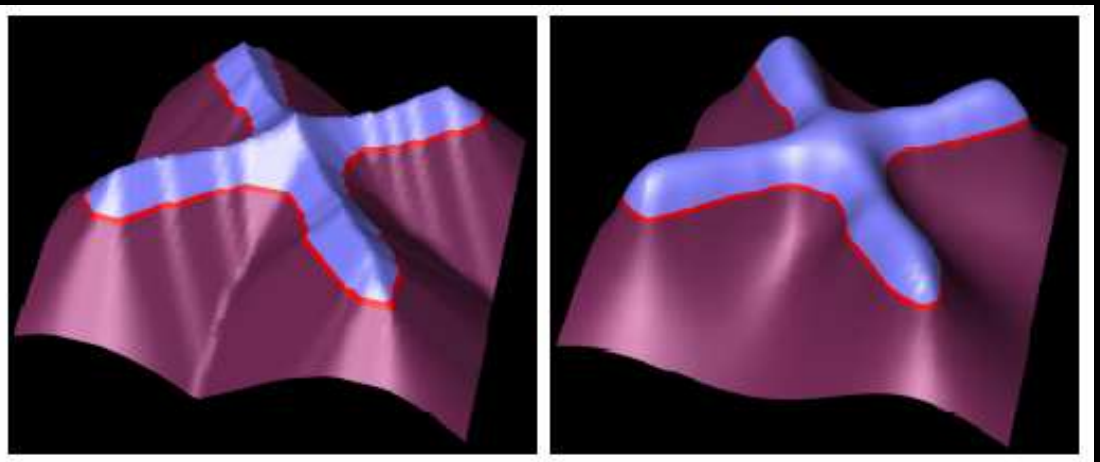


Figure 2: Implicit functions for an X shape. Left shows the signed distance function, and right shows the smoother variational implicit function.

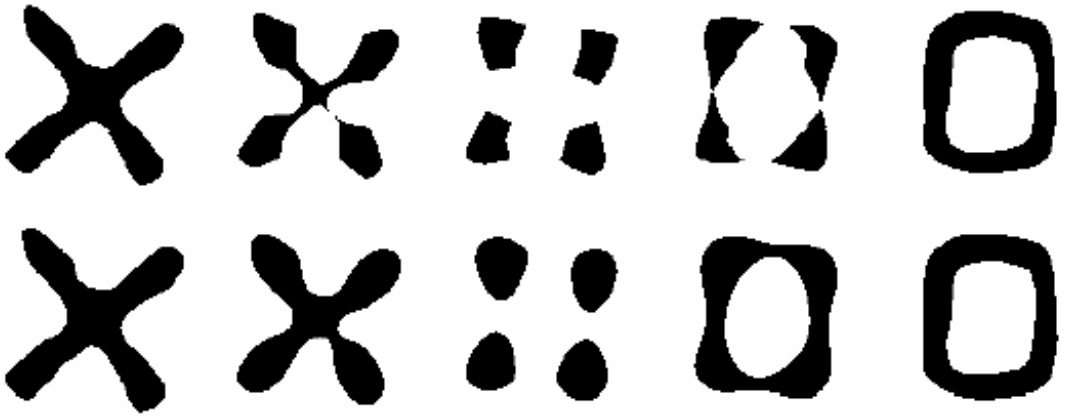
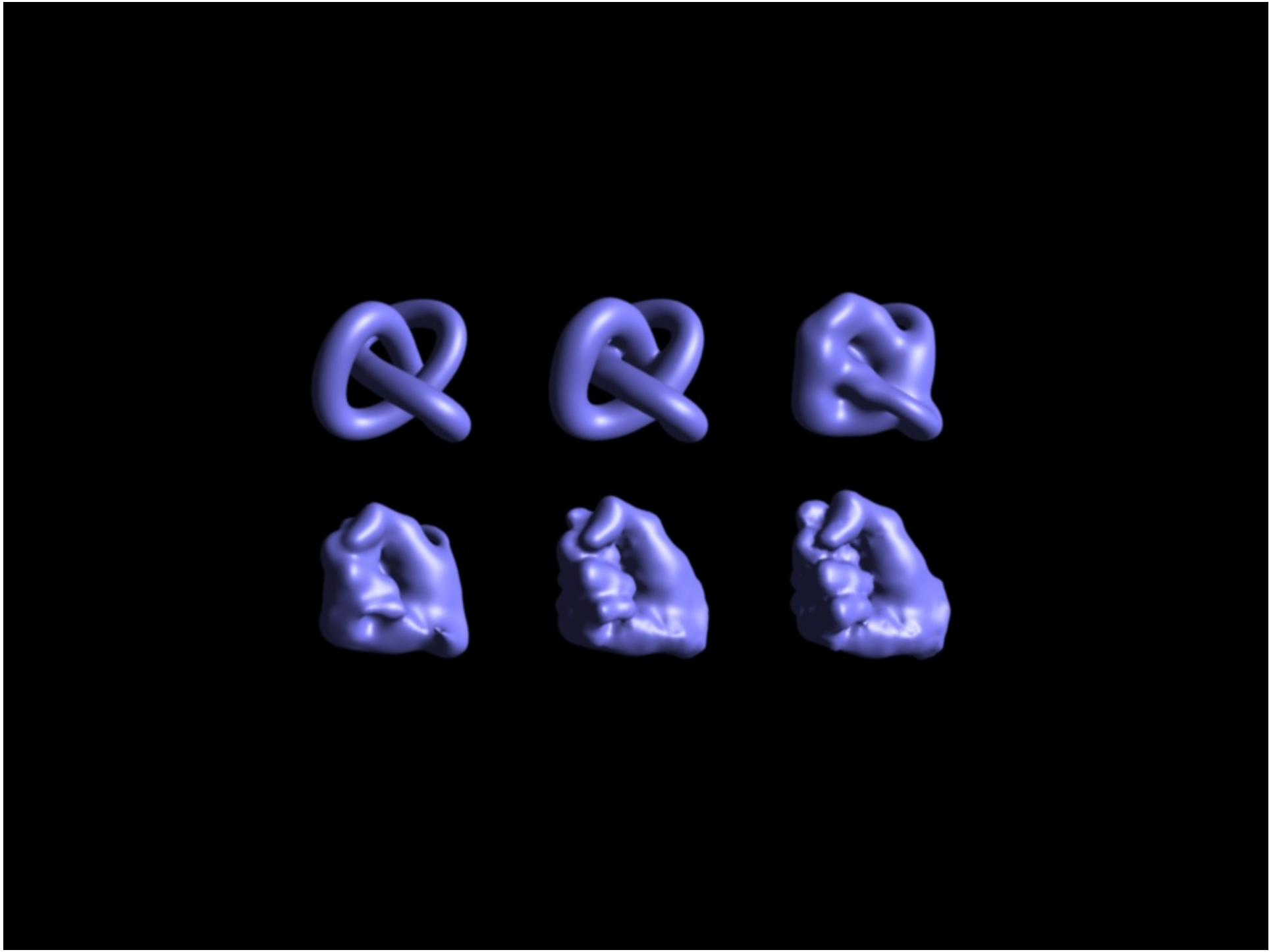
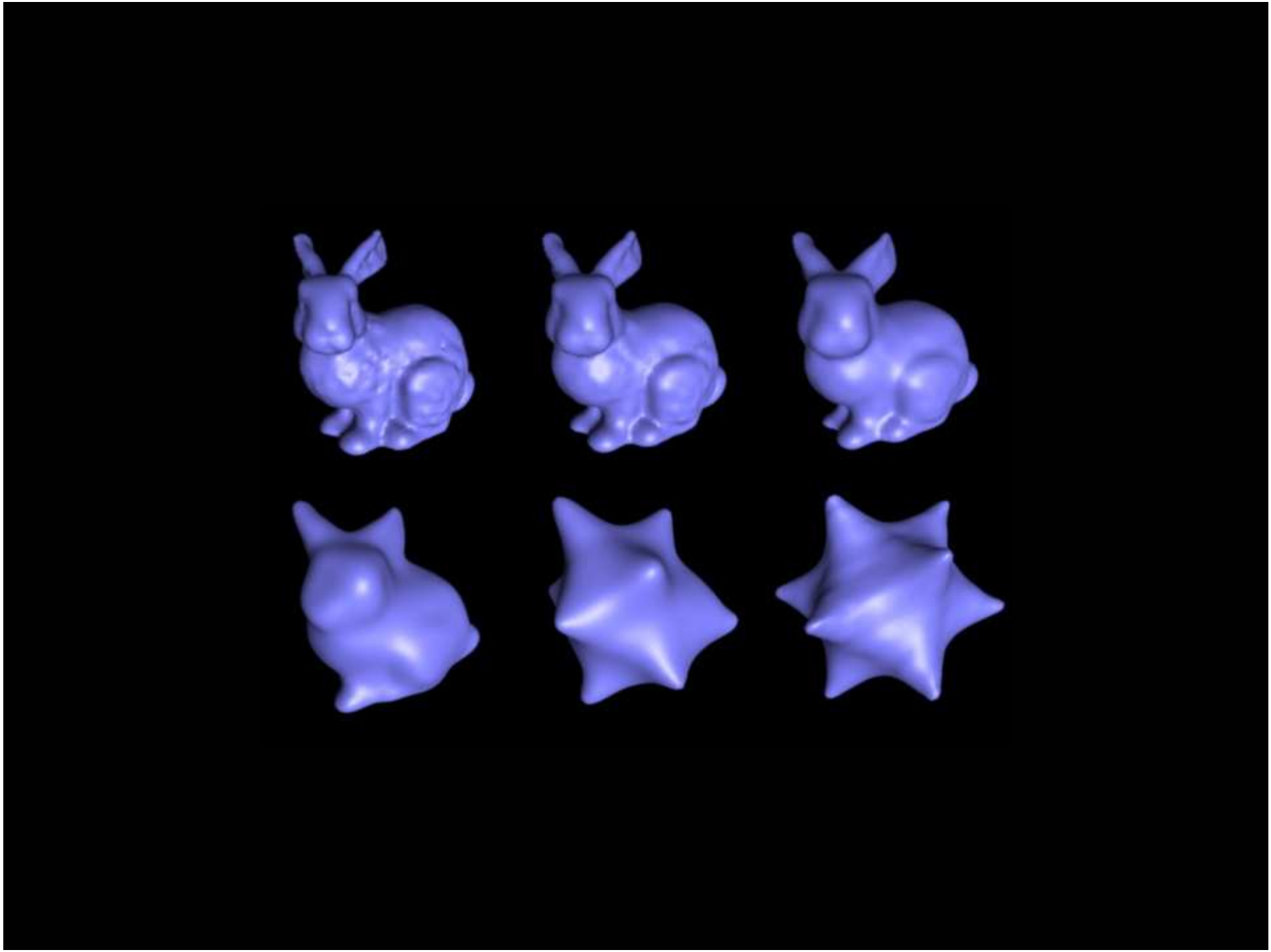


Figure 3: Upper row is a shape transformation created using the signed distance transform. Lower row is the sequence generated using a single variational implicit function.



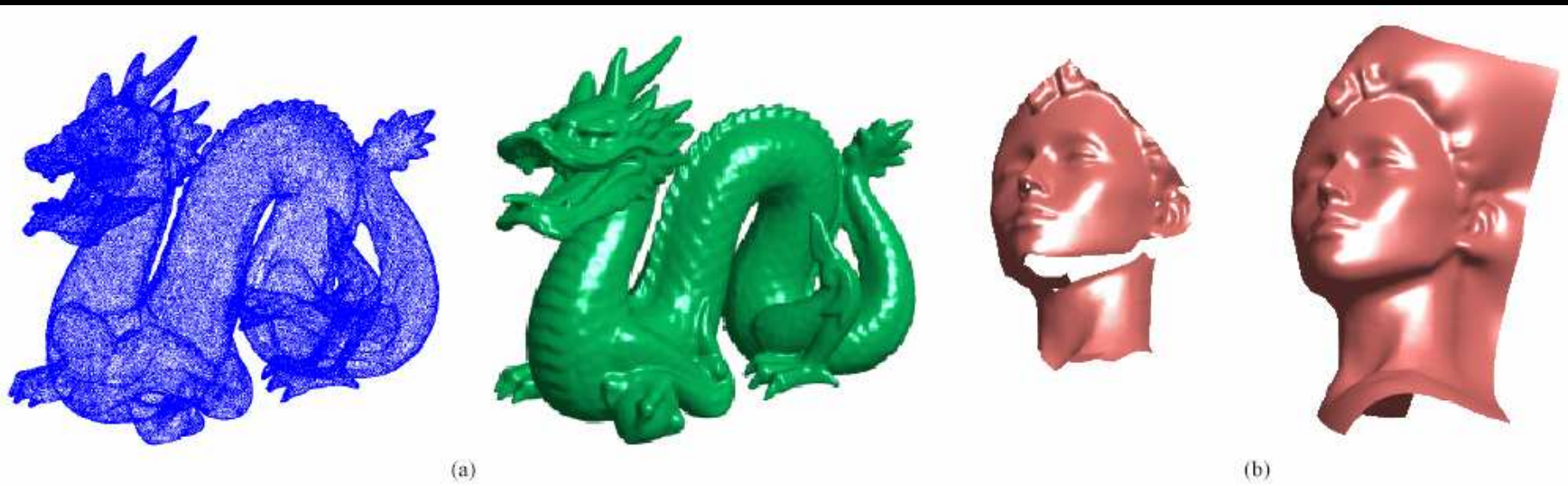


Variational Implicit Functions; Radial Basis Function Surfaces

A few more details…

Reference:

- Reconstruction and Representation of 3D Objects with Radial Basis Functions, Carr et al., SIGGRAPH 2001.



(a) (b)

Figure 1: (a) Fitting a Radial Basis Function (RBF) to a 438,000 point-cloud. (b) Automatic mesh repair using the biharmonic RBF.

Avoiding f(x)=0: Off-surface Points

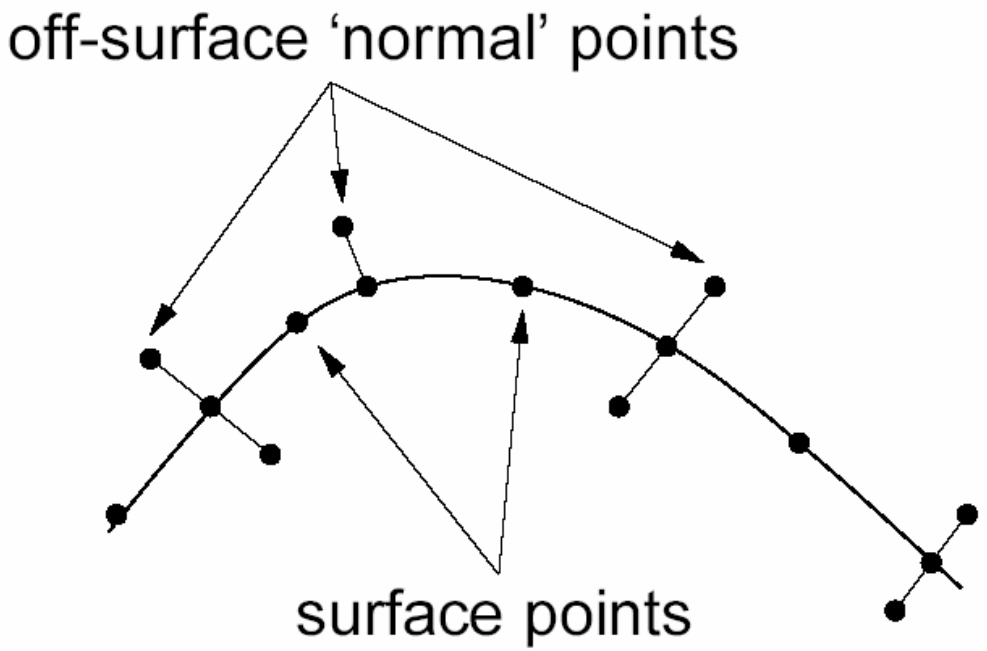


Figure 2: A signed-distance function is constructed from the surface data by specifying off-surface points along surface normals. These points may be specified on either or both sides of the surface, or not at all.

Off-surface Points

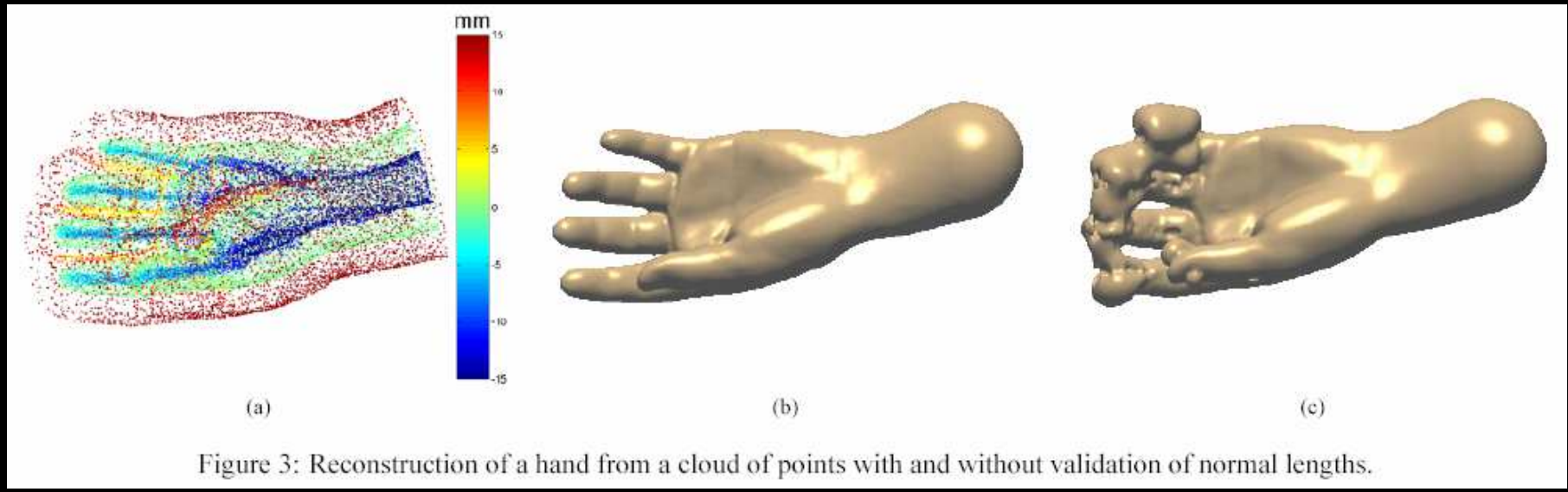


Figure 3: Reconstruction of a hand from a cloud of points with and without validation of normal lengths.

RBF Function

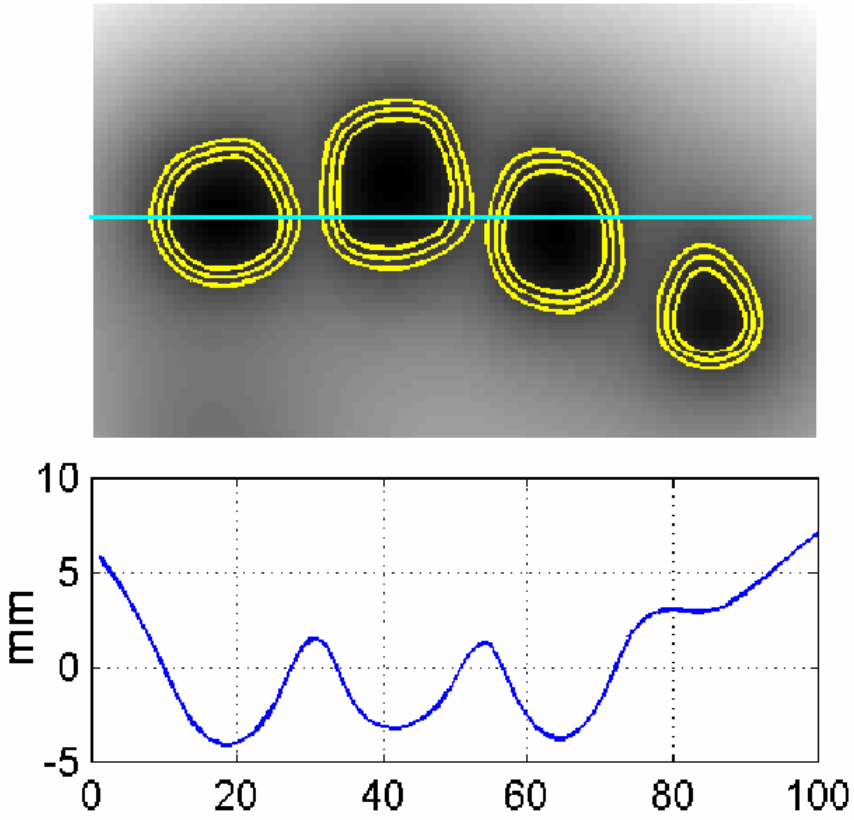


Figure 4: Cross section through the fingers of a hand reconstructed from the point-cloud in Fig. 3. The iso-contours corresponding to +1, 0 and -1 are shown (top) along with a cross sectional profile of the RBF (bottom) along the line shown.

$BL^{(2)}(R^3)$, the Beppo-Levi space of distributions in R^3

$$S = \{s \in \mathrm{BL}^{(2)}(\mathbb{R}^3) : s(x_i) = f_i, \quad i = 1, \ldots, N\}. \tag{3}$$

The space $\mathrm{BL}^{(2)}(\mathbb{R}^3)$ is equipped with the rotation invariant semi-norm defined by

$$\begin{aligned} \|s\|^2 = \int_{\mathbb{R}^3} & \left(\frac{\partial^2 s(x)}{\partial x^2}\right)^2 + \left(\frac{\partial^2 s(x)}{\partial y^2}\right)^2 + \left(\frac{\partial^2 s(x)}{\partial z^2}\right)^2 \\ & + 2\left(\frac{\partial^2 s(x)}{\partial x \partial y}\right)^2 + 2\left(\frac{\partial^2 s(x)}{\partial x \partial z}\right)^2 + 2\left(\frac{\partial^2 s(x)}{\partial y \partial z}\right)^2 dx. \end{aligned} \tag{4}$$

This semi-norm is a measure of the energy or "smoothness" of functions: functions with a small semi-norm are smoother than those with a large semi-norm.

Greedy RBF Algorithm

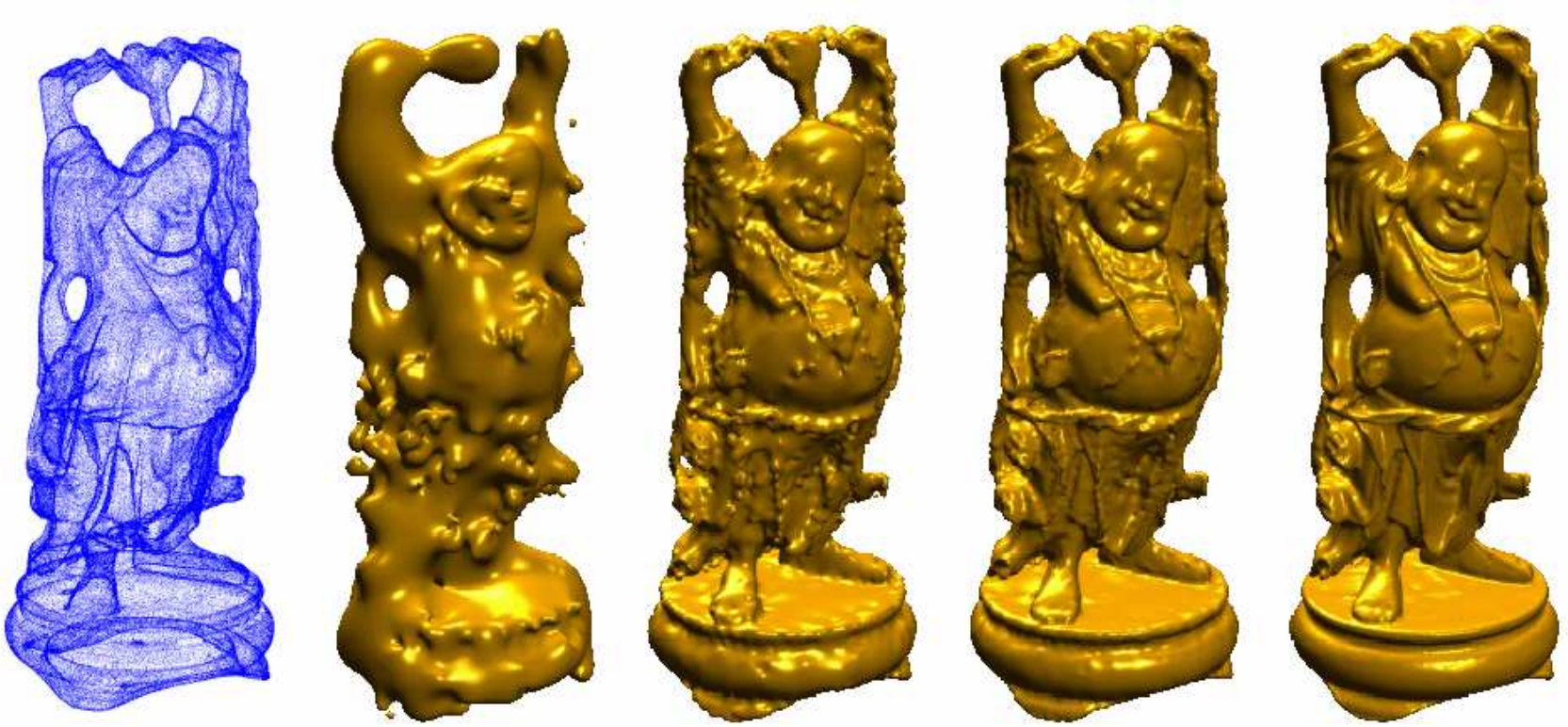


Figure 6: A greedy algorithm iteratively fits an RBF to a point cloud resulting in fewer centers in the final function. In this case the 544,000 point cloud is represented by 80,000 centres to a relative accuracy of 5×10^{-4} in the final frame.

Greedy RBF Algorithm

A simple greedy algorithm consists of the following steps:

1. Choose a subset from the interpolation nodes x_i and fit an RBF only to these.
2. Evaluate the residual, $\varepsilon_i = f_i - s(x_i)$, at all nodes.
3. If $\max\{|\varepsilon_i|\} <$ *fitting accuracy* then stop.
4. Else append new centers where ε_i is large.
5. Re-fit RBF and goto 2.

RBF approximation of noisy data

(a) (b) (c)

Figure 9: (a) Exact fit, (b) medium amount of smoothing applied (the RBF approximates at data points), (c) increased smoothing.

Fast Multipole Method for Complex Models

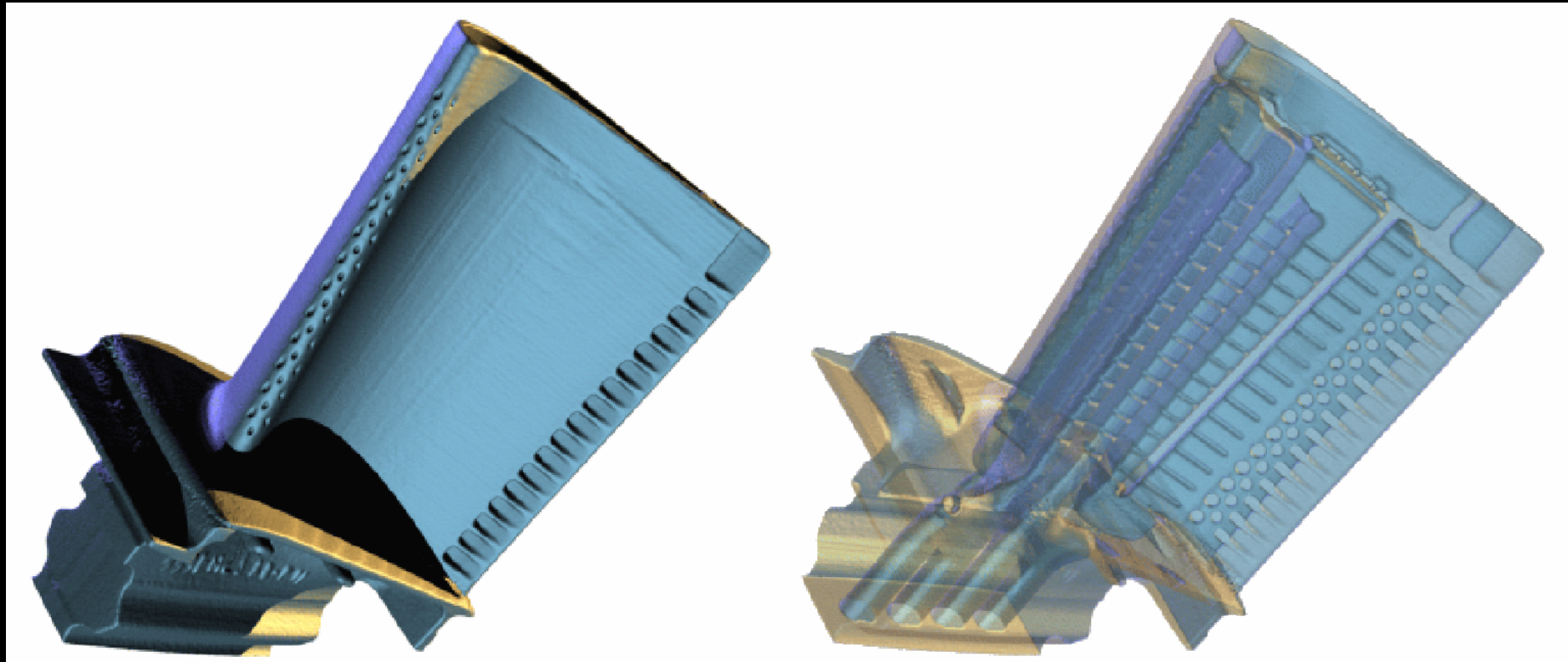


Figure 14: Solid and semi-transparent renderings of an RBF model of a turbine blade containing intricate internal structure. The RBF has 594,000 centers.