# Classical Surface Reconstruction

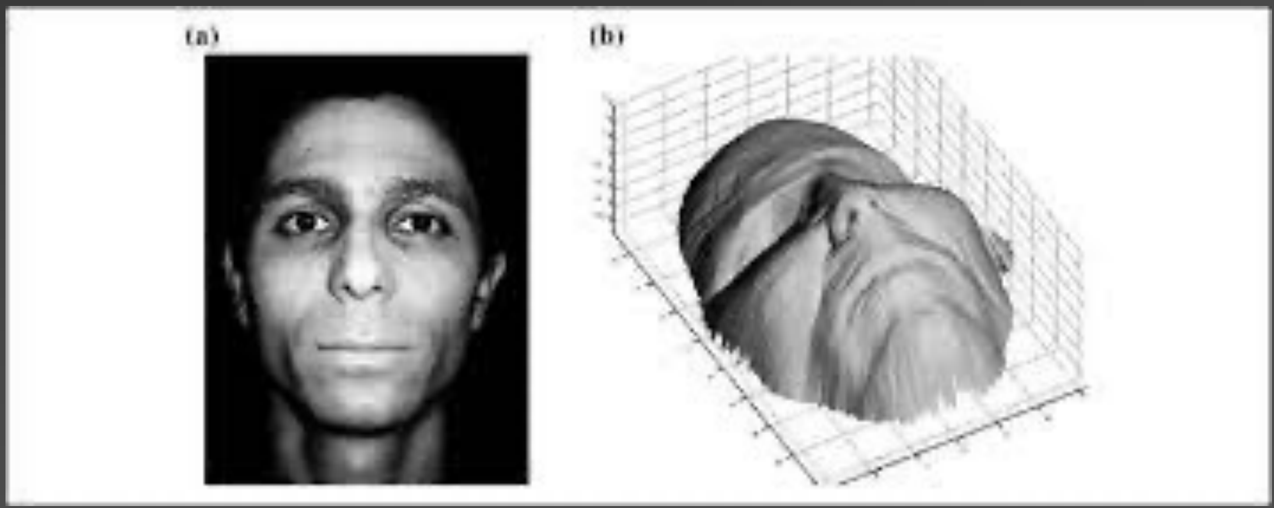
Richard Zhang

CMPT 464/764: Geometric Modeling in Computer Graphics

Lecture 8

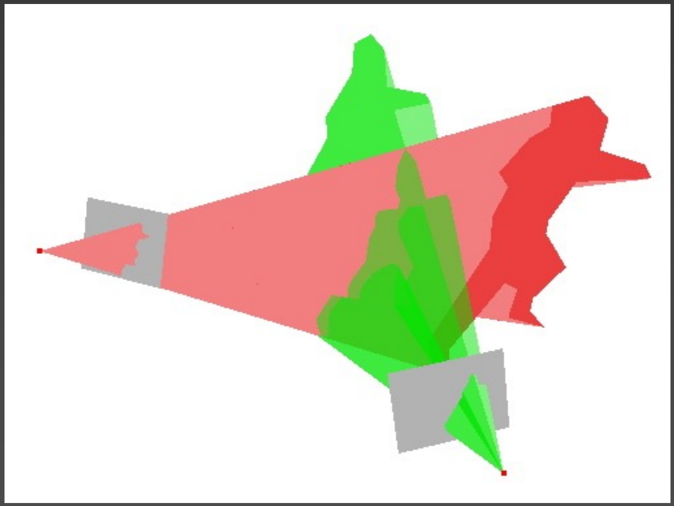
# Single-view input

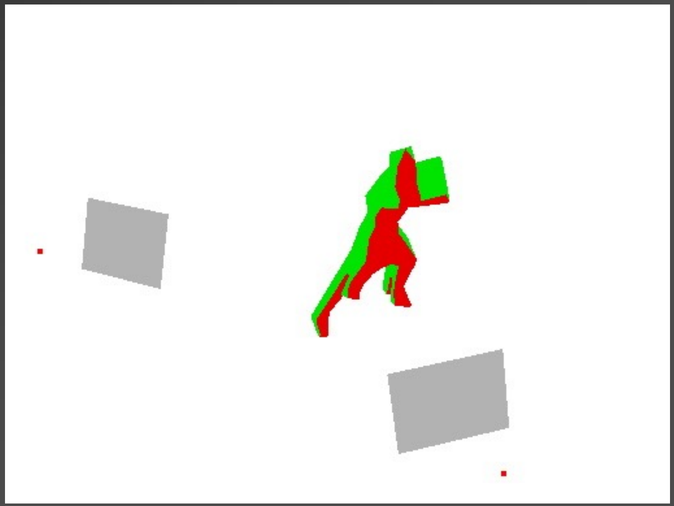
- A prototypical computer vision problem: 3D geometry/surface reconstruction **from single or multiple view sensors (images)**
  - Classical "shape from shading" result



# Multi-view input

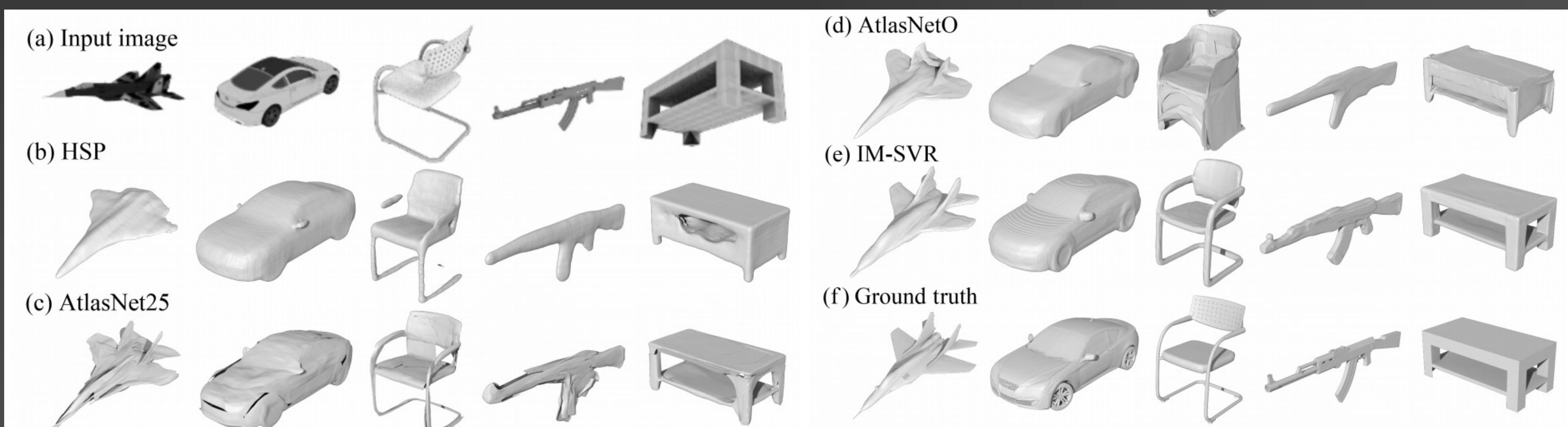
- A prototypical computer vision problem: 3D geometry/surface reconstruction **from single or multiple view sensors (images)**
  - **Visual hull:** shape from **multi-view silhouettes**





# Learning-based: single-view

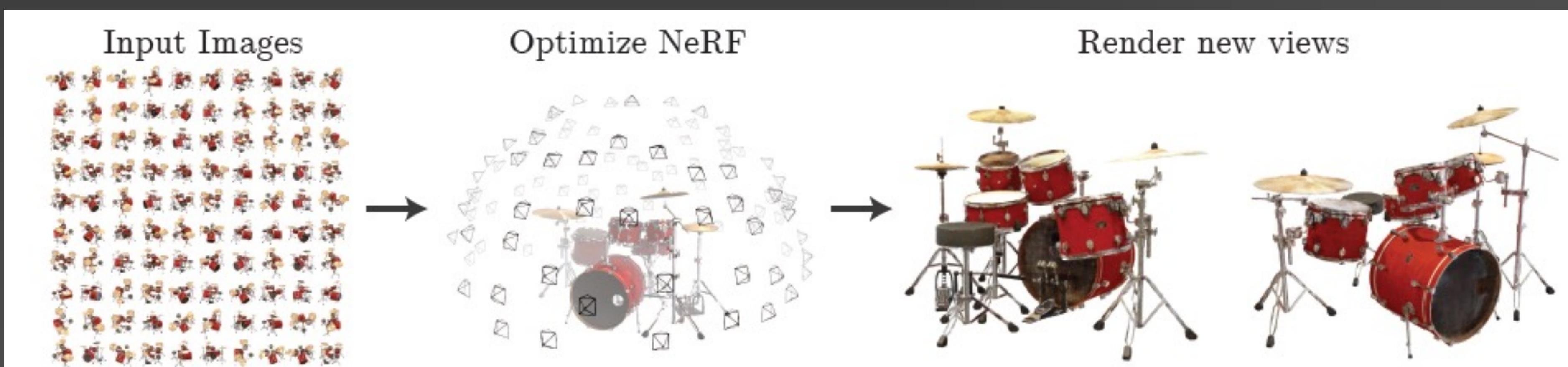
- A prototypical computer vision problem: 3D geometry/surface reconstruction **from a single or multiple view sensors (images)**
  - One of the most intensely studied problems in geometric deep learning




[IM-Net: Chen and Zhang 2019]

# Learning-based: multi-view

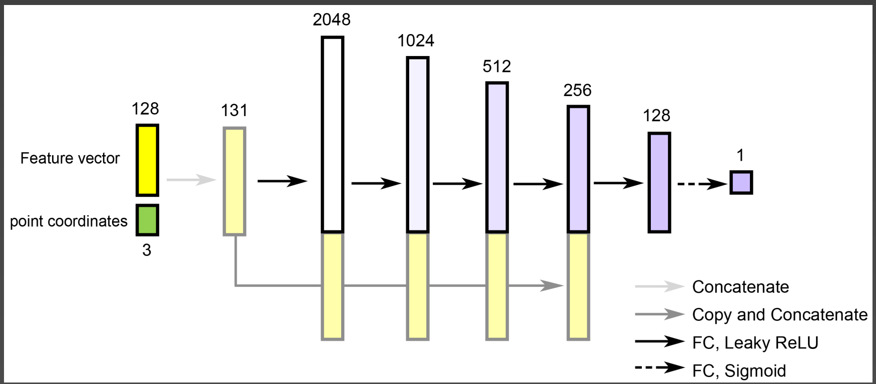
- A prototypical computer vision problem: 3D geometry/surface reconstruction **from a single or multiple view sensors (images)**
  - NeRF (2020): Neural Radiance Field, from multi-view images
  - **Novel view synthesis** (NVS): need **many images and long training**



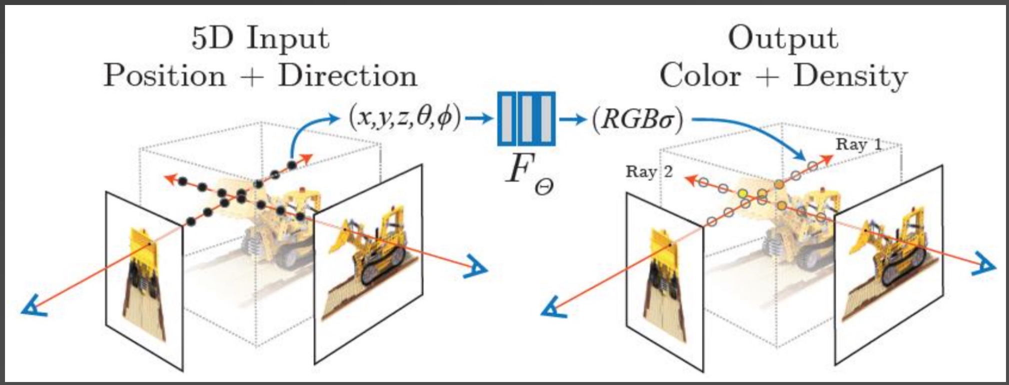

[Middenhall et al. 2020]

# Learning-based: multi-view

- A prototypical computer vision problem: 3D geometry/surface reconstruction **from a single or multiple view sensors (images)**
  - NeRF (2020): Neural Radiance Fields from multi-view images
  - **Connections between IM-Net and NeRF**

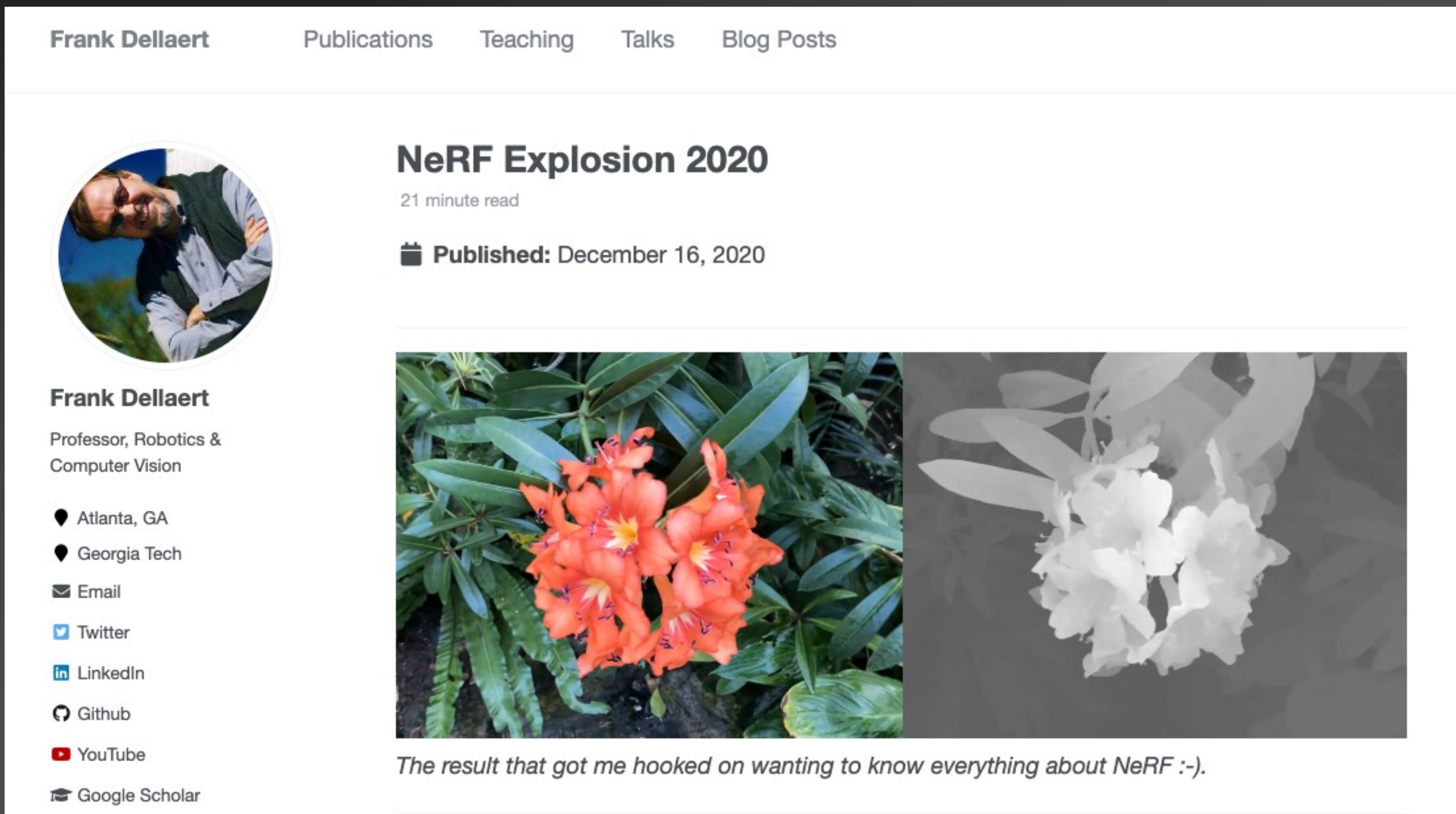



[IM-Net: Chen and Zhang 2019]




[Middenhall et al. 2020]

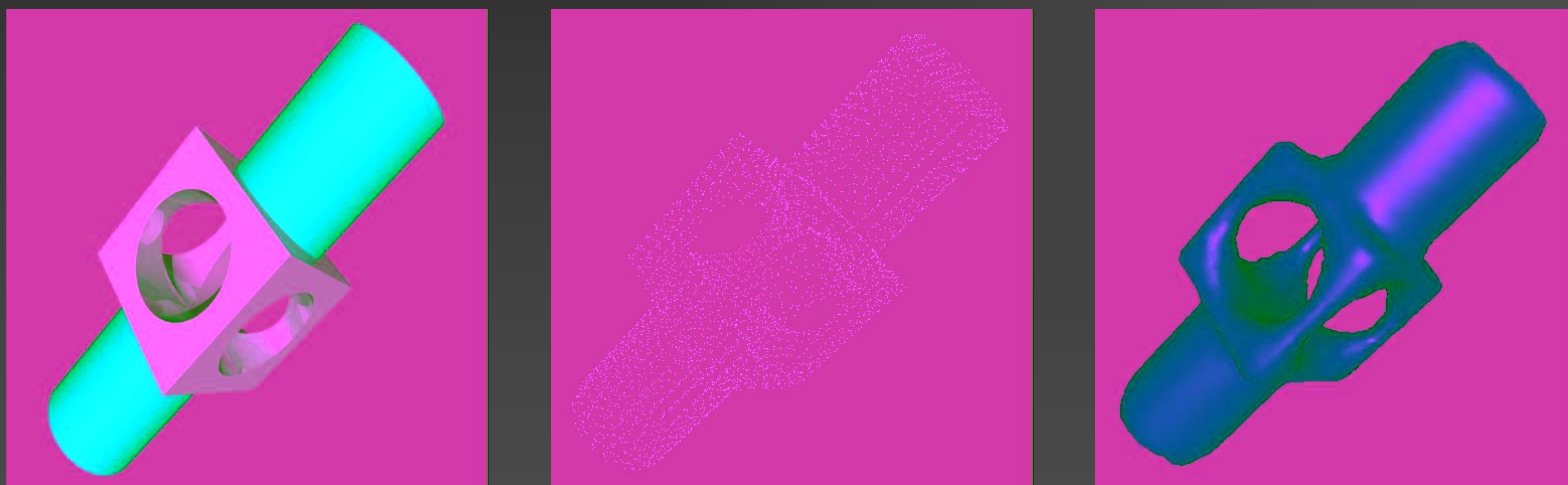
# See: Account from "NeRF Explosion"




https://dellaert.github.io/NeRF/

# This week: from a “graphics origin”

Given a set of **unorganized 3D points** $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ sampled from an unknown surface $M$, construct a surface $M'$ that approximates $M$.



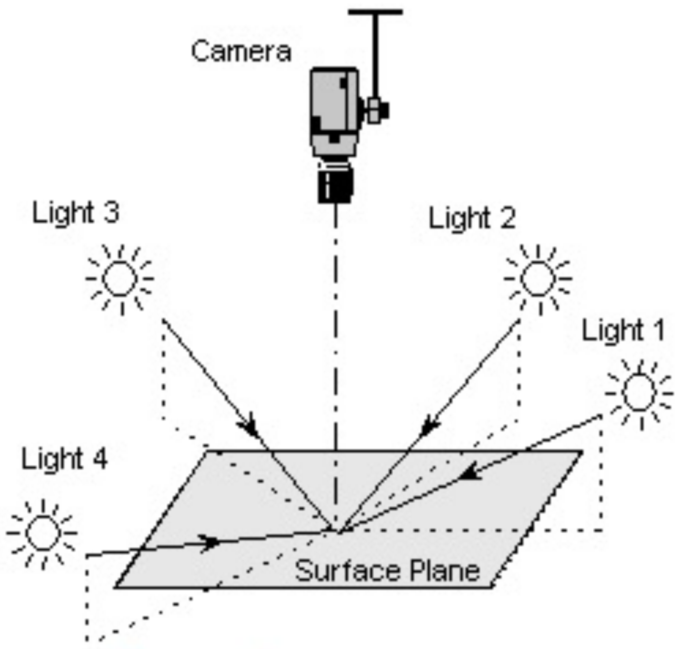
Surface reconstruction from **unorganized point cloud** data

# Background

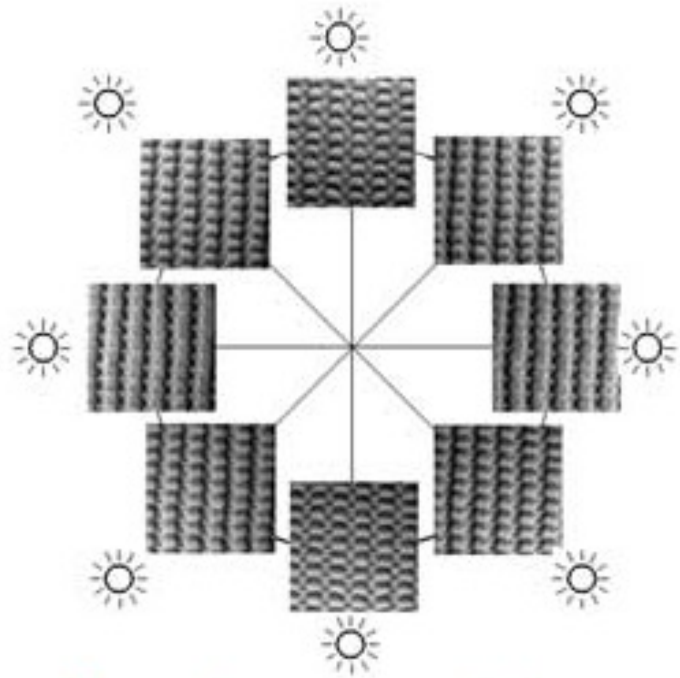
- Input: point cloud obtained via laser scanning with **no normal information**
- Output: a triangle mesh
- Surface *M'* can either **interpolate** or **approximate** *X*
- Solve a general problem: no structure or organization of points assumed …
  - Here **structural information** refers to specific knowledge about the arrangement of the point samples, e.g., contours on **parallel slices in MRI**
  - Some info about the device specs can be known, e.g., **scanning accuracy**
  - Normal information may be available via **photometric stereo** [Woodham 80]

# Photometric stereo

- Estimate surface orientation from different images



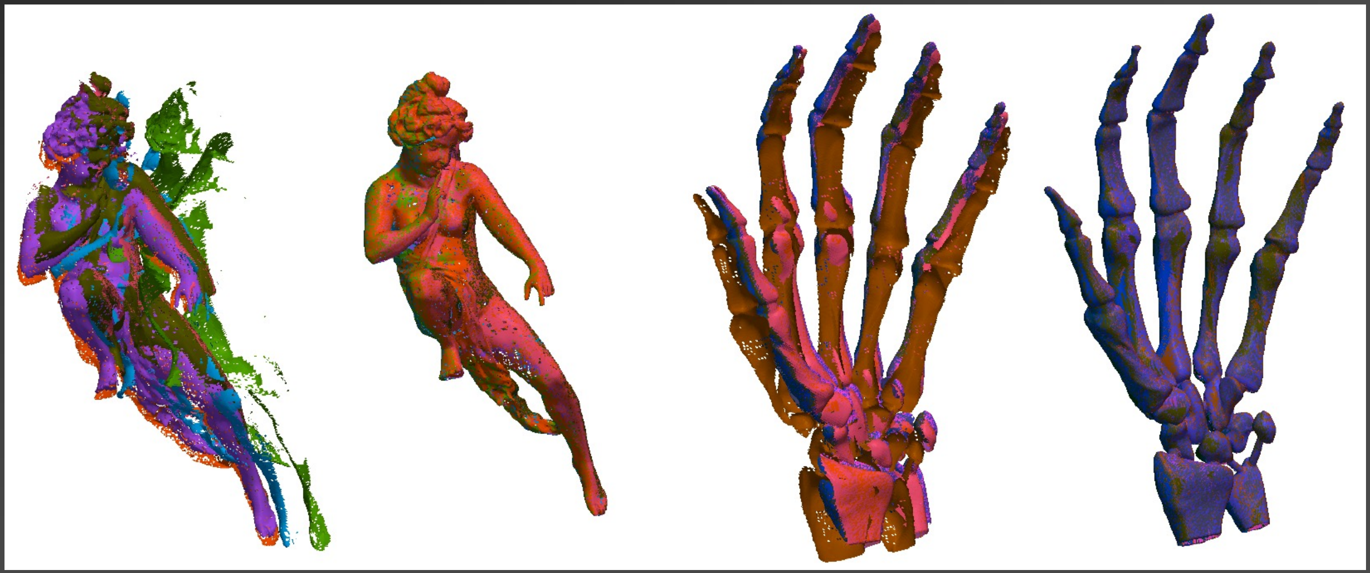

Photometric stereo geometry



Photometric stereo example images

# Many related problems

- (Static) surface registration: bring several partial scans to **alignment**



- Key: point or region **correspondence** - a topic we cover later

# Many related problems

- **Multi-view** geometry reconstruction, e.g., Microsoft photosynth

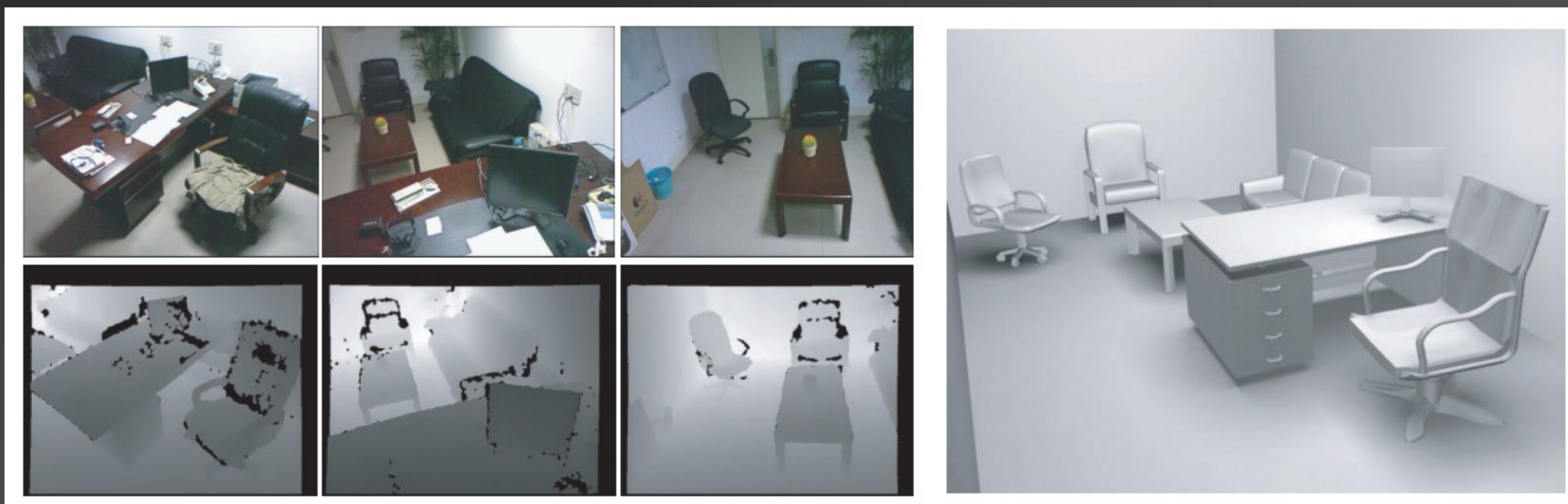
- Sub-problems: **shape-from-shading**, e.g., photo to point clouds, and (multi-view) point cloud **registration**

# Many related problems

- **Time-varying** surface **tracking**, e.g., for deformation or animation

# Problem scales

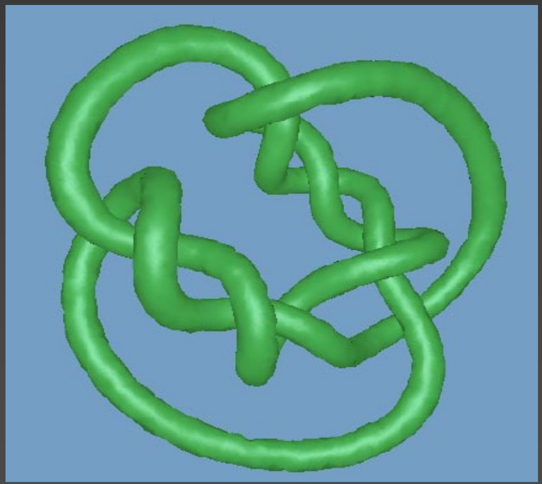
- From single objects (our focus) to scenes to buildings and cities!

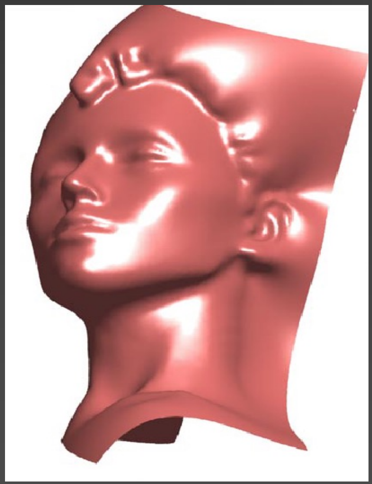


Scaling up from objects to scenes [Shao et al. *Siggraph Asia* 2012]

# Our problem: challenges

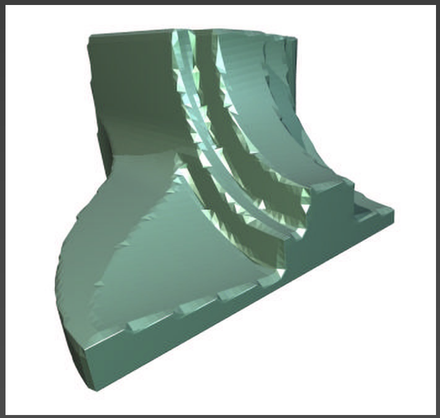
- Reconstruction should cover a range of shapes
  - Arbitrary topology, even if manifold, and arbitrary details
  - Shapes with **boundaries, holes, missing data**, etc.

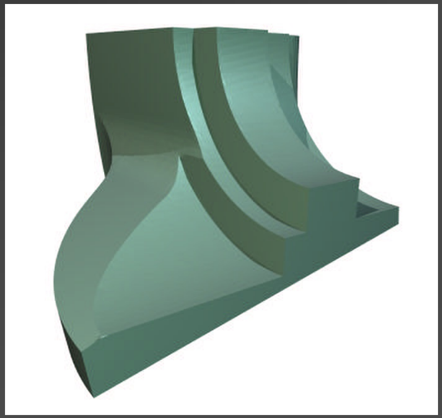




# Challenges

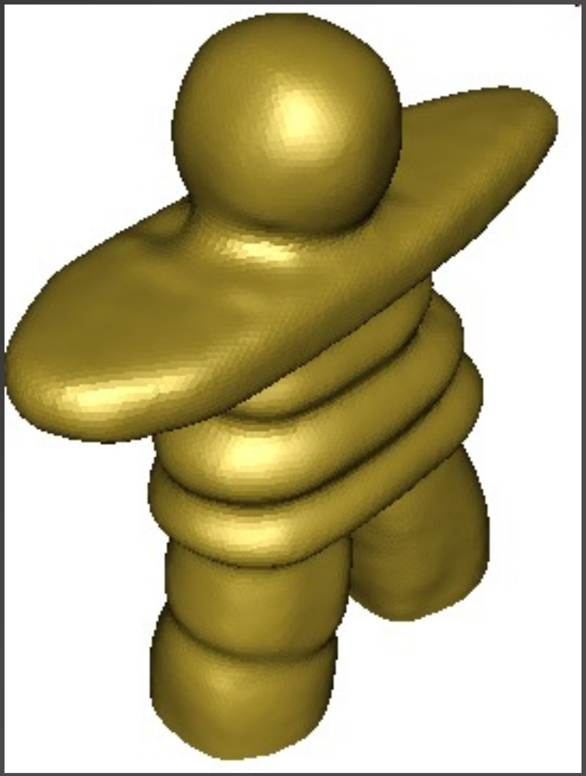
- Ensure consistent surface orientation
- Deal with **noise** in the data
- Recover **sharp features**: not easy if points are not on edges



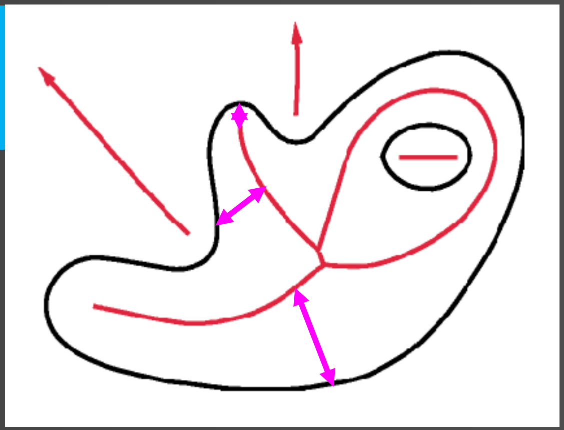


Feature-sensitive reconstruction [Kobbelt et al. 2001]

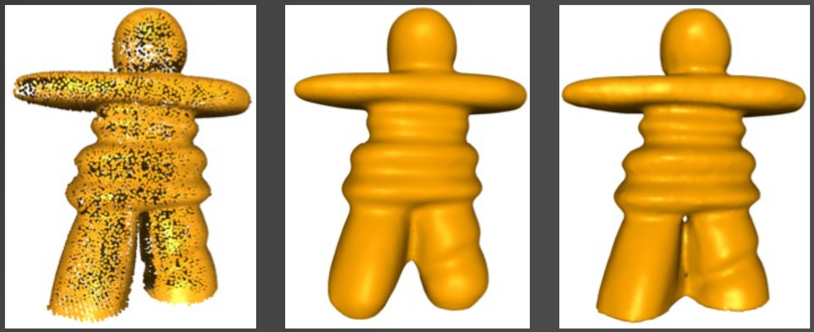
# Missing data and noise



# Theoretical challenge (aside)



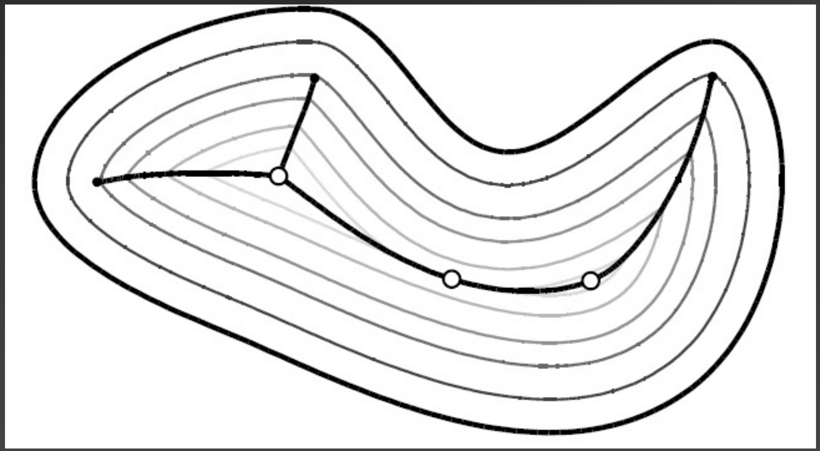
- Ensure “correctness” of reconstruction, meaning
  - **Topology** correctness
  - **Geometry** precision: as sampling density increases, reconstruction approaches the original surface

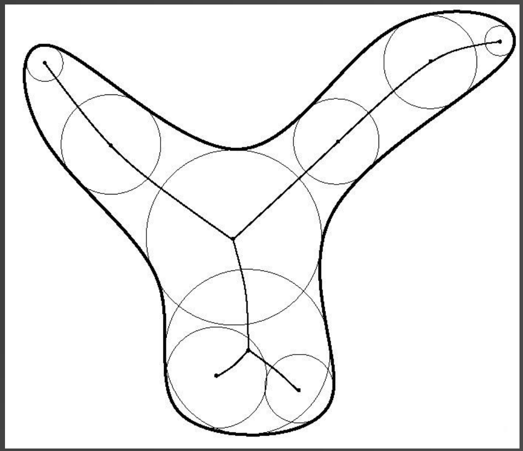


- Correctness guarantees possible if sampling is sufficiently “good” – not easy to achieve or define “goodness” [Amenta et al. 98]
  - Related to **local feature size: distance to medial axis**

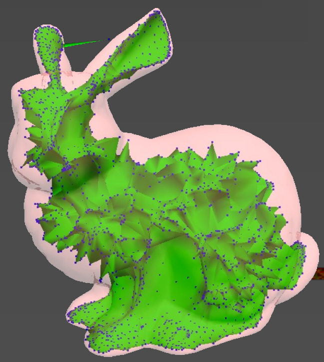
# Medial axis (aside)

- **Singularities or meeting fronts** of a “grass-fire flow”





- Set of all points that have **at least two closest points** to the boundary
- Medial axes for 3D shapes have **sheets** rather than curves



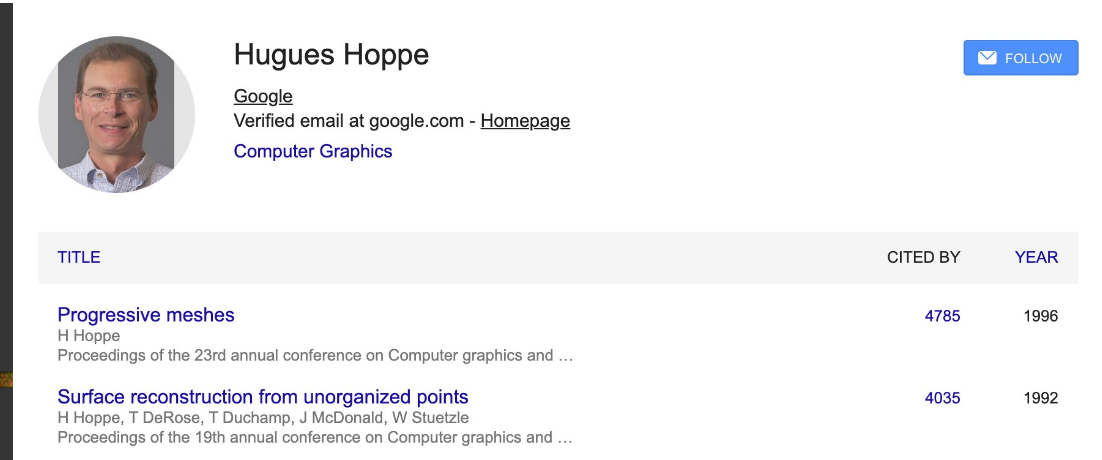
# Classical main approaches

- Reconstruct **zero-set of a 3D scalar field**, e.g., via **marching cubes**
  - Use of tangent plane estimators – [Hoppe et al. 92]
  - Use of radial basis functions – [Carr et al. 01]
- Utilizing **Voronoi diagrams or Delaunay Tetrahedralizations** – [Amenta et al. 01, Boissonnat 84, Dey & Goswami 03, Kolluri et al. 04]
  - Power crust algorithm – [Amenta et al. 01]
- **Deform-to-fit** with energy minimization
  - e.g., inflating a balloon from inside the object – [Terzopoulos, Witkin, and Kass 88 & 91, Miller 91]

# Classical main approaches

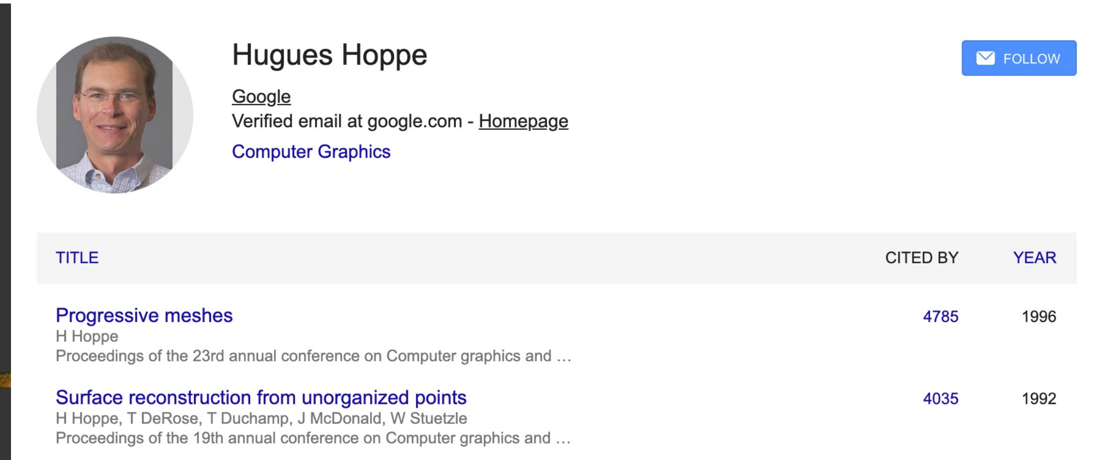
- Reconstruct **zero-set of a 3D scalar field**, e.g., via **marching cubes**
  - Use of tangent plane estimators – [Hoppe et al. 92]
  - Use of radial basis functions – [Carr et al. 01]
- Utilizing **Voronoi diagrams or Delaunay Tetrahedralizations** – [Amenta et al. 01, Boissonnat 84, Dey & Goswami 03, Kolluri et al. 04]
  - Power crust algorithm – [Amenta et al. 01]
- **Deform-to-fit** with energy minimization
  - e.g., inflating a balloon from inside the object – [Terzopoulos, Witkin, and Kass 88 & 91, Miller 91]

# Our coverage




- H. Hoppe et al., “Surface Reconstruction from Unorganized Points.” *SIGGRAPH* 92
- W. Lorensen and H. Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm,” *SIGGRAPH* 87
  - A sub-algorithm of the surface reconstruction algorithm
  - One of most fundamental surface reconstruction algorithms itself
  - Input is volumeric data or scalar field of signed distances to surfacee
  - Algorithm constructs approximation of the **zero-set** of the scalar field

# Our coverage




- H. Hoppe et al., “Surface Reconstruction from Unorganized Points.” *SIGGRAPH 92*
- W. Lorensen and H. Cline, “Marching Cubes: A High Resolution 3D


https://www.computer.org/csdl/magazine/cg/2020/02/09020249/1hS2S5b2V6E

# Assumptions

- … on **data noise (measurement error)**
  - The samples $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ are **$\delta$-noisy**, i.e., each sample is no farther than $\delta$ away from its true position
  - Features of size less < $\delta$ cannot be recovered reliably
- … on **sampling density**
  - **$\rho$-dense**: within each sphere centered at a point on surface $M$ having radius $\rho$, at least one sample is drawn
  - This assumption is necessary in order to distinguish between holes in surface (boundary) and holes in the sampling
  - If there is an empty sphere with radius ($\delta + \rho$) embedded in the sampling, then it is a hole in the model

# Overview of Hoppe's approach

- Input: set $X$ of unorganized 3D points ($\delta$-noisy; $\rho$-dense) sampled near surface $M$
- Algorithm in two stages
  1. Obtain an **implicit function** $f : D \rightarrow \mathbf{R}$, where $D \subseteq \mathbf{R}^3$, is a **region near true surface** $M$, and $f(\mathbf{p})$ estimates the **signed distance** from $\mathbf{p}$ to $M$
  2. The **zero-set Z(*f*)** of $f$ is an estimate of $M$. A **contouring** or **marching-cube** algorithm approximates $Z(f)$ by a triangle mesh
- Output: a connected, consistently oriented 2-manifold triangle mesh
- A general paradigm: implicit function $f$ can be obtained in various ways, Hoppe paper uses a set of approximate **tangent planes**

# Learning of implicit/signed distance functions

- Generates surfaces with best visual quality so far

**Learning Implicit Fields for Generative Shape Modeling**

Zhiqin Chen, Hao Zhang

*(Submitted on 6 Dec 2018 (v1), last revised 5 Apr 2019 (this version, v3))*

**Occupancy Networks: Learning 3D Reconstruction in Function Space**

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, Andreas Geiger

*(Submitted on 10 Dec 2018)*

**DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation**

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, Steven Lovegrove

*(Submitted on 16 Jan 2019)*

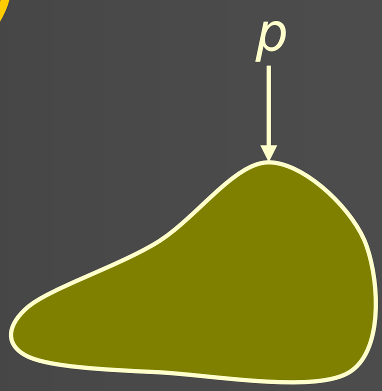
**Deep Level Sets: Implicit Surface Representations for 3D Shape Inference**

Mateusz Michalkiewicz, Jhony K. Pontes, Dominic Jack, Mahsa Baktashmotlagh, Anders Eriksson

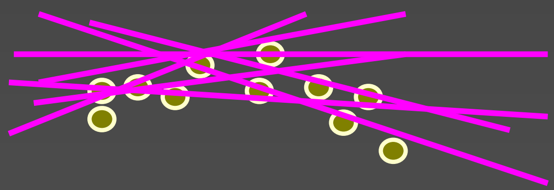
*(Submitted on 21 Jan 2019)*

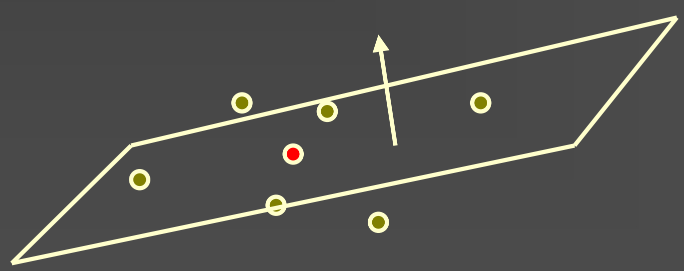
Awesome implicit neural representations:
https://github.com/vsitzmann/awesome-implicit-representations

# Signed distance function (SDF)

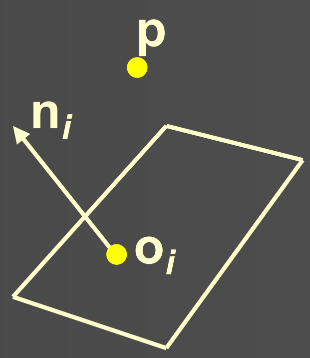



- Distance from a point **p** to a surface *M* is the distance from **p** to a **closest point** on *M*
- Sign depends on which side of *M* point **p** lies
- Since *M* is unknown, it is approximated by a set of **oriented tangent planes – one per data point**
- Tangent plane for $\mathbf{x}_i$ is defined by a center $\mathbf{o}_i$ and a unit normal $\mathbf{n}_i$





# Computing SDF, given tangent planes




- Determine **region *D* close to surface *M***
- If $\mathbf{p} \in D$, the signed distance from **p** to *M* is a projection

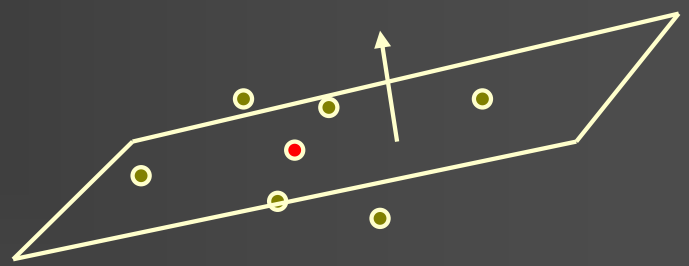
$$f(\mathbf{p}) = (\mathbf{p} - \mathbf{o}_i) \cdot \mathbf{n}_i$$

where **$\mathbf{o}_i$ is the tangent plane center that is closest to p**

- If the shortest distance from a point **p** to the point set *X* is > $(\delta + \rho)$, then **p** cannot be on the surface *M*
  - Otherwise, the sphere centered at **p** with radius $\delta + \rho$ must contain a point from *X*, since the samples are $\delta$-noisy and $\rho$-dense
  - **p** is possibly near a hole on the surface → ***f*(p) is undefined**
  - The remaining set of **p** define *D*

# Tangent plane estimation – key!

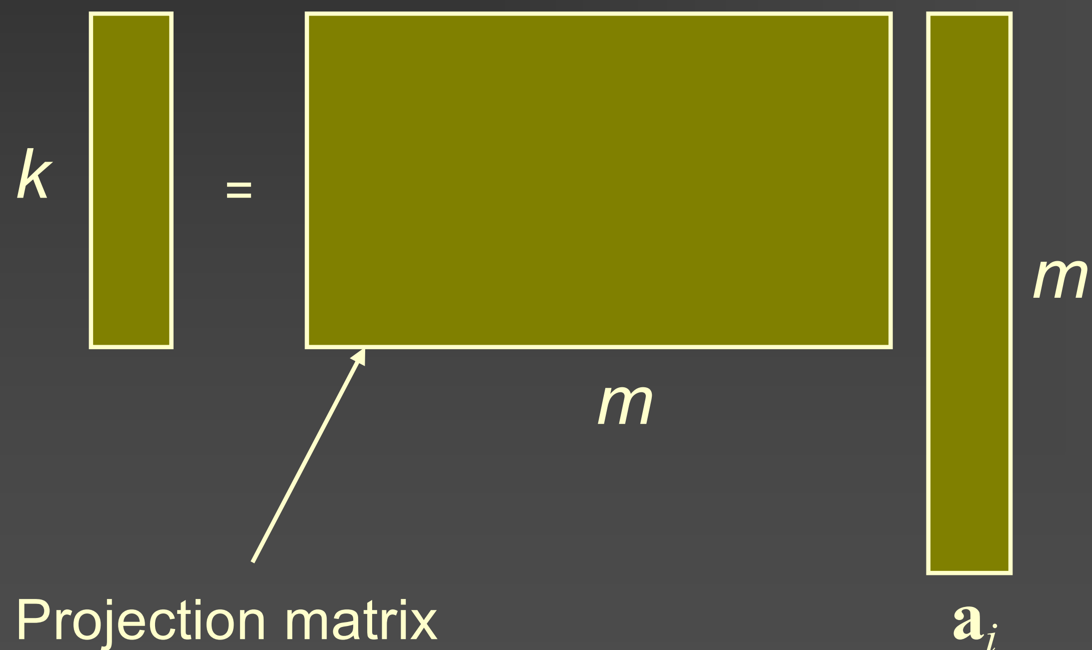
- How to define a tangent plane associated with a sample $\mathbf{x}_i$?
- Define: $Nbr(\mathbf{x}_i, k)$ = the set of ***k* nearest neighbors** (*k*NN) of a data point $\mathbf{x}_i$, where $k$ is a user input value
- Center $\mathbf{o}_i$ is the **centroid** of $Nbr(\mathbf{x}_i, k)$



- Normal $\mathbf{n}_i$ is determined by **principal component analysis (PCA)**
- The oriented plane passing through $\mathbf{o}_i$ having normal +/– $\mathbf{n}_i$ provides the **least squares best fit** to points in $Nbr(\mathbf{x}_i, k)$

# PCA: Linear dimensionality reduction

- **Linearly map** a set of $m$-dimensional vectors $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$, to an $k$-dimensional subspace, $k < m$, so as to minimize the approximation error in the least square sense

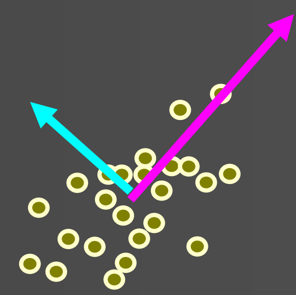


# Principal component analysis (PCA)

- Project data points $\mathbf{a}_i$ onto the leading $k$ eigenvectors (for the $k$ **largest eigenvalues**) of the **covariance matrix** $\Sigma$ for the original data set $\mathbf{a}$

$$\Sigma = (\mathbf{a} - \bar{\mathbf{a}}\mathbf{1}^T)(\mathbf{a} - \bar{\mathbf{a}}\mathbf{1}^T)^T = \Sigma_{j=1..n}(\mathbf{a}_i - \bar{\mathbf{a}}) \cdot (\mathbf{a}_i - \bar{\mathbf{a}})^T \in \mathbf{R}^{m \times m}$$

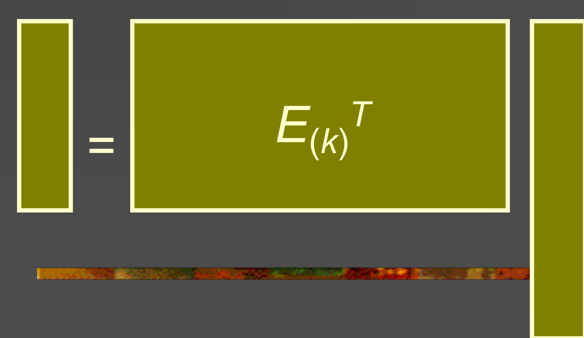
where $\bar{\mathbf{a}}$ is the (uniform) mean of data points in $\mathbf{a}$.



- Eigenvectors: orthogonal and **major modes of variations**
- A ***k*-dimensional embedding** is obtained by

$$\hat{\mathbf{a}}_{(k)} = E_{(k)}{}^T\mathbf{a},$$

where $E_{(k)} \in \mathbf{R}^{m \times k}$ has $k$ columns of leading eigenvectors of $\Sigma$.



# PCA

- Project data points $\mathbf{a}_i$ onto the leading $k$ eigenvectors (for the $k$ **largest eigenvalues**) of the **covariance matrix** $\Sigma$ for the original data set $\mathbf{a}$

$$\Sigma = (\mathbf{a} - \bar{\mathbf{a}}\mathbf{1}^T)(\mathbf{a} - \bar{\mathbf{a}}\mathbf{1}^T)^T = \Sigma_{j=1..n}(\mathbf{a}_i - \bar{\mathbf{a}}) \cdot (\mathbf{a}_i - \bar{\mathbf{a}})^T \in \mathbf{R}^{m \times m}$$

where $\bar{\mathbf{a}}$ is the (uniform) mean of data points in $\mathbf{a}$.

# Normal of the tangent plane

- Covariance matrix $\Sigma$ of 3D points in $Nbr(\mathbf{x}_i, k)$ is a symmetric (positive semi-definite) $3 \times 3$ matrix
- The normal chosen for $Nbr(\mathbf{x}_i, k)$ is +/– of the eigenvector of $\Sigma$ corresponding to the **smallest eigenvalue** of $\Sigma$
- The 2-dimensional subspace, i.e., the plane, is spanned by the other two eigenvectors
- The exact **sign** of the normal is chosen so that nearby tangent planes are **consistently oriented**

# Derivation of PCA (aside)

- Given a set of 3D points $x_1, \ldots, x_k$, find a **best fitting plane** ($o$, $n$) in the **least squares sense**, where $o$ is a point on the plane and $n$ is the **unit** plane normal
- The minimization problem: $$\min \sum_{i=1}^{k} [(x_i - o)^T \cdot n]^2 \text{ subject to } n^T n = 1$$
- Use **Lagrange Multiplier**: $$\min \; F(o, n, \lambda) = \sum_{i=1}^{k} [(x_i - o)^T \cdot n]^2 - \lambda(n^T n - 1)$$
- We assume that $n = (n_x, n_y, n_z)^T \neq 0$.
- Differentiate $F$ with respect to $o$, we have $$\left[\sum_{i=1}^{k} (x_i - o)^T\right] \cdot n = 0$$
- Differentiate $F$ with respect to $n_x$, $n_y$, $n_z$ and then combine into matrix form, we have $$\left[\sum_{i=1}^{k} (x_i - o)(x_i - o)^T\right] \cdot n = \lambda n$$

# Derivation of PCA (aside)

$$\left[\sum_{i=1}^{k}(x_i - o)(x_i - o)^T\right] \cdot n = \lambda n$$

- So the normal *n* is an eigenvector of the covariance matrix and there are three local minima corresponding to three eigenvectors
- Alternatively, the minimization problem is really

$$\min \sum_{i=1}^{k}[(x_i - o)^T \cdot n]^2 = n^T \Sigma n, \text{ subject to } n^T n = 1$$

- By Courant-Fischer Theorem, this is just an eigenvalue problem

# Consistent normal orientation

- A harder part of the algorithm – it tells **topological information**
- One can model it as a global graph optimization problem
  - One node $N_i$ per tangent plane
  - Two nodes connected if the corresponding centers are sufficiently close (where consistent orientation is enforced)
  - Cost of edge $(N_i, N_j)$ is $\mathbf{n}_i \cdot \mathbf{n}_j$ (maximum if coplanar)
  - Problem: **Find orientation to maximize the total cost in graph**
- But this optimization problem is NP-hard (i.e., its decision version is NP-complete)

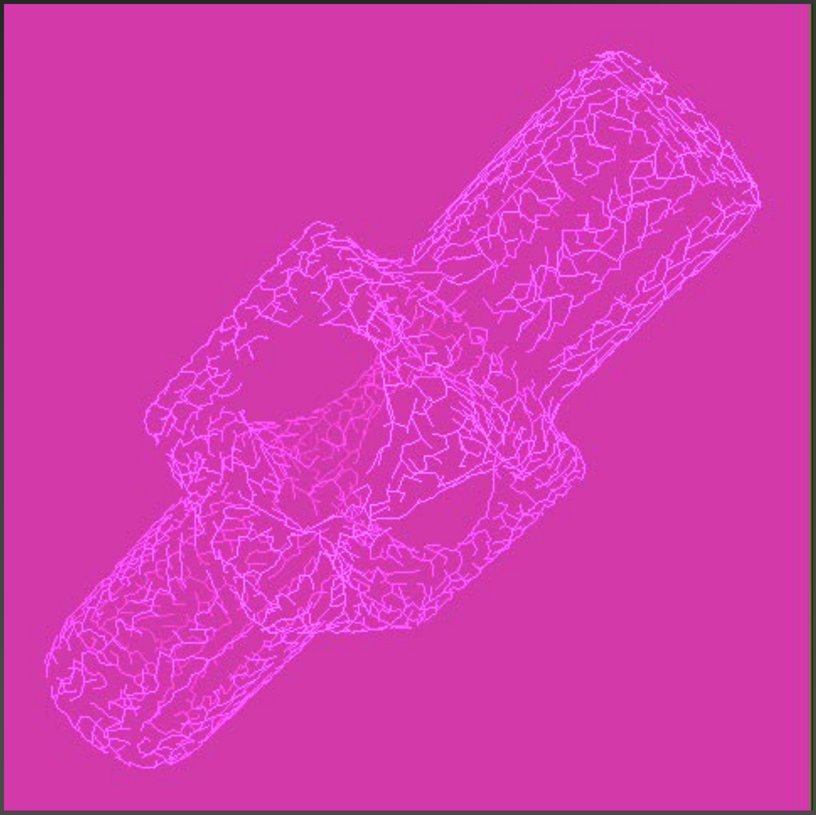
# Approximate solution

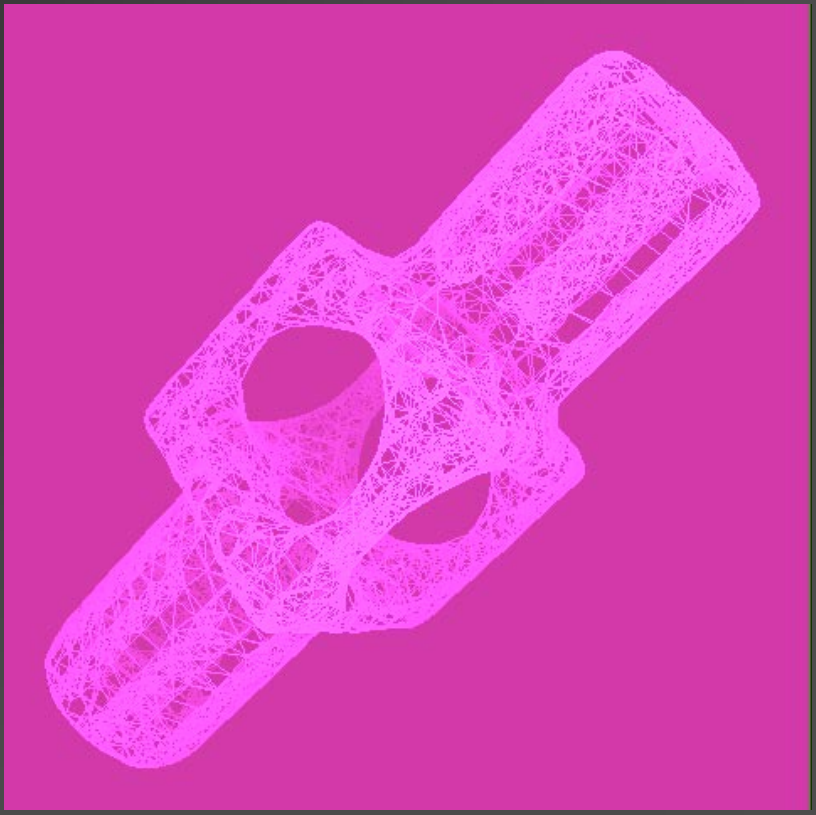
- First, build a **Riemannian graph** on tangent plane centers
  - Riemannian graph: encodes the geometric proximity of the tangent plane centers
  - Riemannian graph is built upon the **Euclidean minimum spanning tree** (EMST) – connected, tends to connect near neighbors, but there are not enough edges
  - Add an edge ($N_i$, $N_j$) to EMST if $\mathbf{o}_i$ is one of the *k* closest neighbors of $\mathbf{o}_j$ or vice versa

# Recall: EMST

- Given a set of points *L*, an EMST is a spanning tree of *L* with the **minimum total cost** (edge cost measured by Euclidean distance)
- Can be obtained via **Kruskal's minimum spanning tree algorithm**
  - Conceptually consider complete graph on *L* with Euclidean distances as edge weights
  - Greedily add shortest edges that do not form a cycle
  - Stop when no edges can be added any more

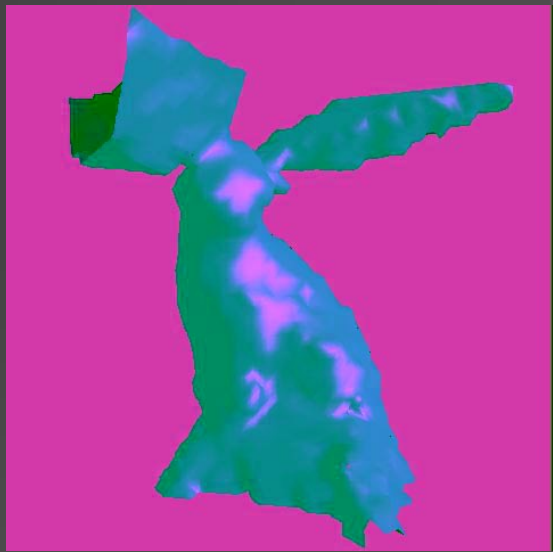
# EMST and Riemannian graph





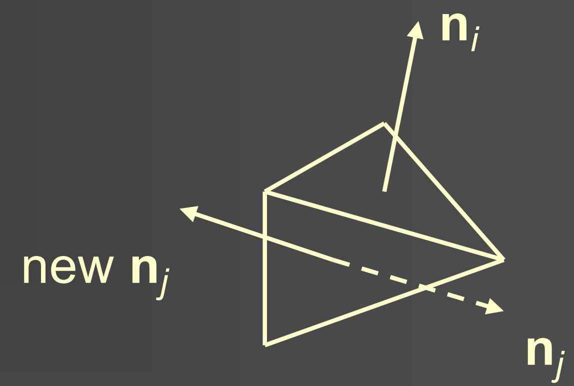
# Orientation propagation

- To start propagation, choose orientation for an initial plane
- Propagate this orientation to its nearby planes by traversing the Riemannian graph
- **Traversal order is important**
- A heuristic: propagate along **low curvature** directions –
  - favor propagation from plane *i* to *j* if they are almost parallel
  - less likely to be a mistake

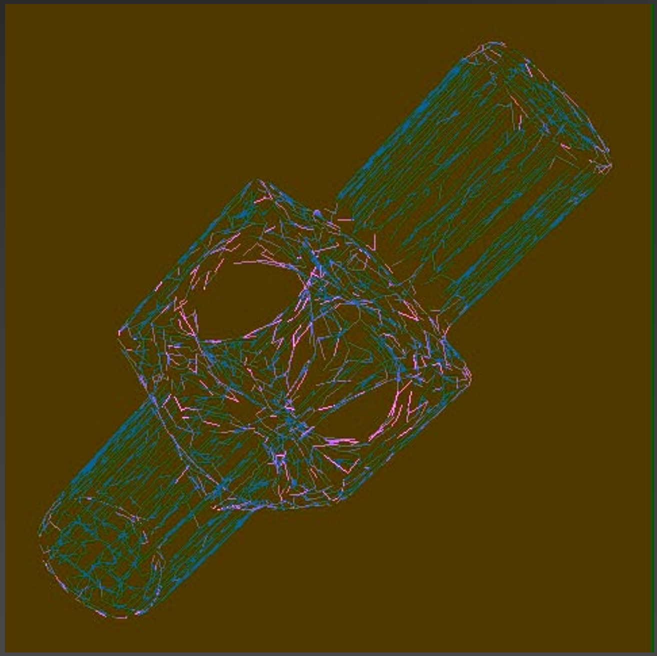


# Algorithm

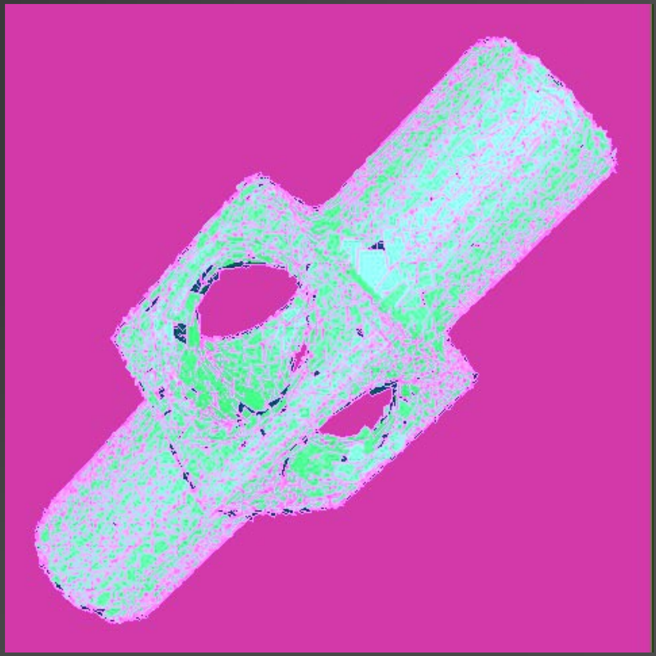
- Assign weight $(1 - |\mathbf{n}_i \cdot \mathbf{n}_j|)$ to edge $(N_i, N_j)$
- Propagate along edges of **minimum spanning tree** of the resulting graph (**depth-first search**)
- How to propagate from $\mathbf{n}_i$ to next plane $j$ ?
  - If $\mathbf{n}_i \cdot \mathbf{n}_j < 0$, $\mathbf{n}_j = -\mathbf{n}_j$
- How to choose an initial orientation?
  - Normal of plane whose center has largest $z$ value is forced to point to $+z$ direction



# Result



MST of normal variation graph with edge costs colored



Oriented tangent planes as shaded triangles

# Recall SDF

- $f(\mathbf{p})$ is signed distance from **p** to "closest tangent plane"
- Since sampling is $\delta$-noisy and $\rho$-dense, if $f(\mathbf{p}) > \delta + \rho$, then **p** cannot be on the surface $M$

  $\rightarrow f(\mathbf{p})$ is undefined in this case

- Otherwise, the signed distance from **p** to $M$ is a projection

$$f(\mathbf{p}) = (\mathbf{p} - \mathbf{o}_i) \cdot \mathbf{n}_i$$

where $\mathbf{o}_i$ is the tangent plane center that is closest to **p**

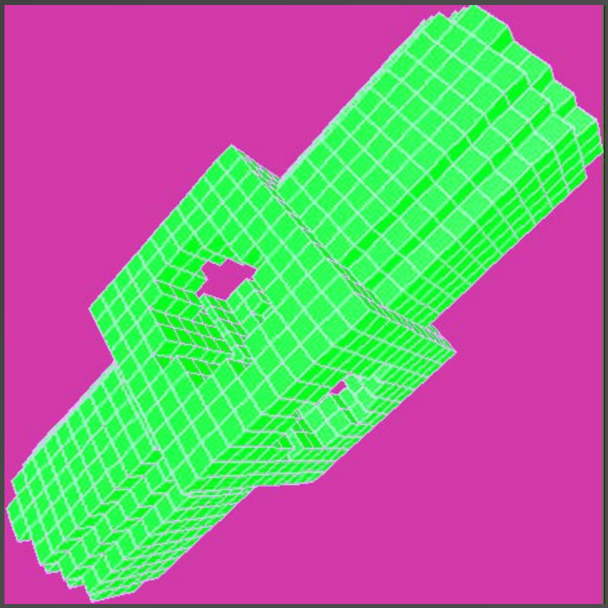
Why is $f(\mathbf{p})$ not the closest distance from p to any tangent plane?

# Contour tracing

- Given the set of **oriented** tangent planes, SDF from points to these planes can be computed
- Next, need to extract the iso-surface corresponding to the **zero-set** of the signed distance function
- This can be done using a **Marching Cubes** (**contour tracing**) algorithm or one of its variants

# Preparation for cubes marching

- Divide 3D space into cubical grids
- **Sample signed distance values at cube vertices**
- Only choose cubes that intersect the zero iso-surface for efficiency
- Size of cube $d \approx \delta + \rho$, why?
  - if $d >> \delta + \rho$, may fill holes or join boundaries
  - if $d$ too small, complexity too high
- No intersection between zero-surface and cube if a vertex has undefined $f(\mathbf{p})$

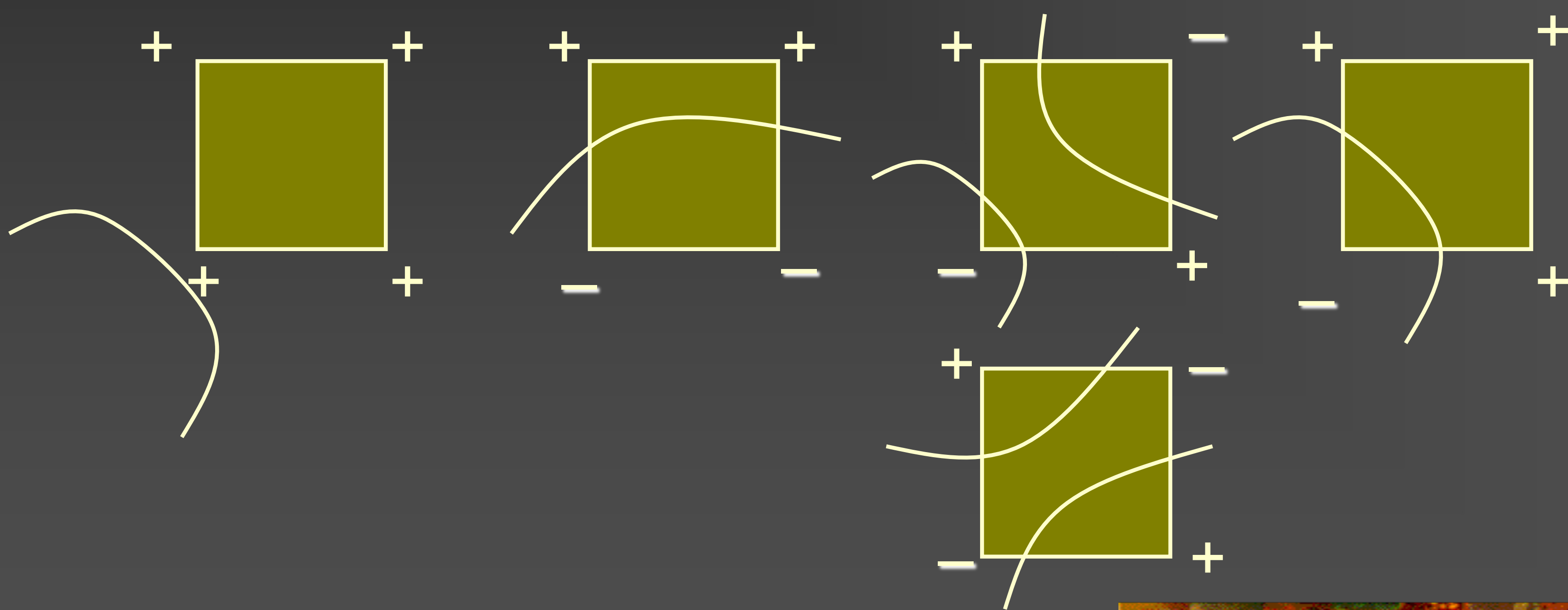


# Marching cubes algorithm

- Input: a scalar field sampled over the vertices of a cubical grid
- Output: a set of triangles approximating the zero iso-surface of the scalar field
- Basic idea:
  - Process (march) cubes one at a time
  - Look at scalar values at vertices to decide how the iso-surface intersects the cube
  - Generate triangles reflecting these intersections

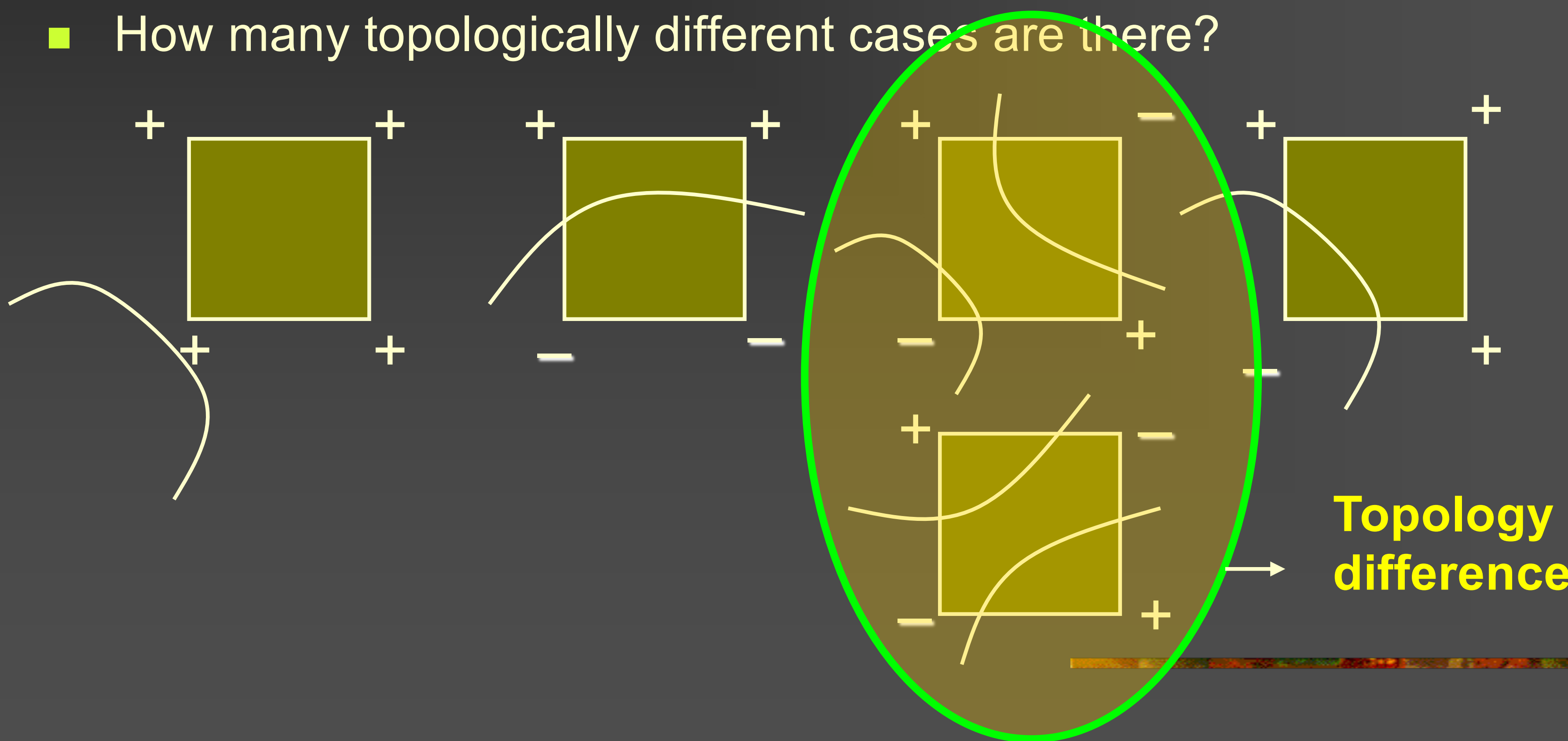
# 2D case: iso-contouring

- Inside iso-curve ≡ < and iso-value ≡ –
- Outside iso-curve ≡ > and iso-value ≡ +
- How many topologically different cases are there?

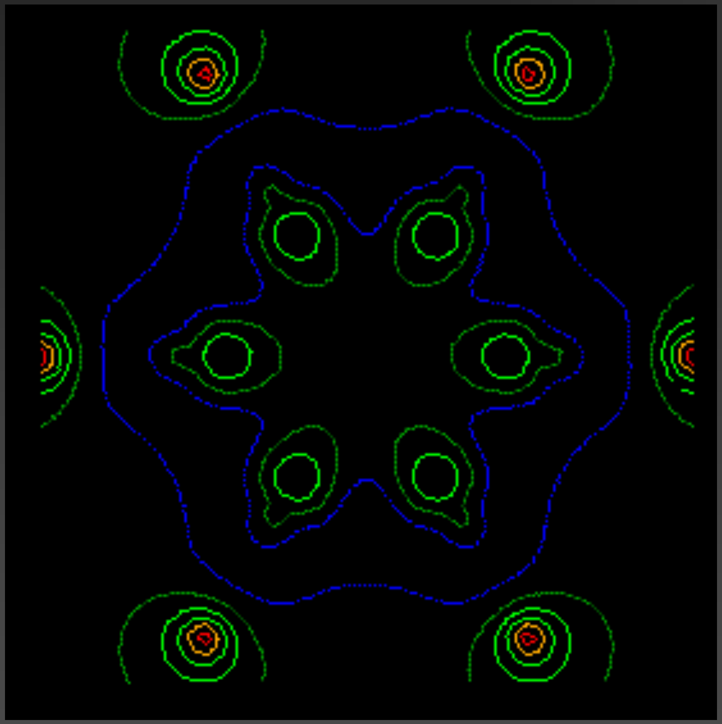


# 2D case: iso-contouring

- Inside iso-curve ≡ <   and   iso-value ≡ –
- Outside iso-curve ≡ >   and   iso-value ≡ +
- How many topologically different cases are there?



# Iso-contouring algorithm sketch

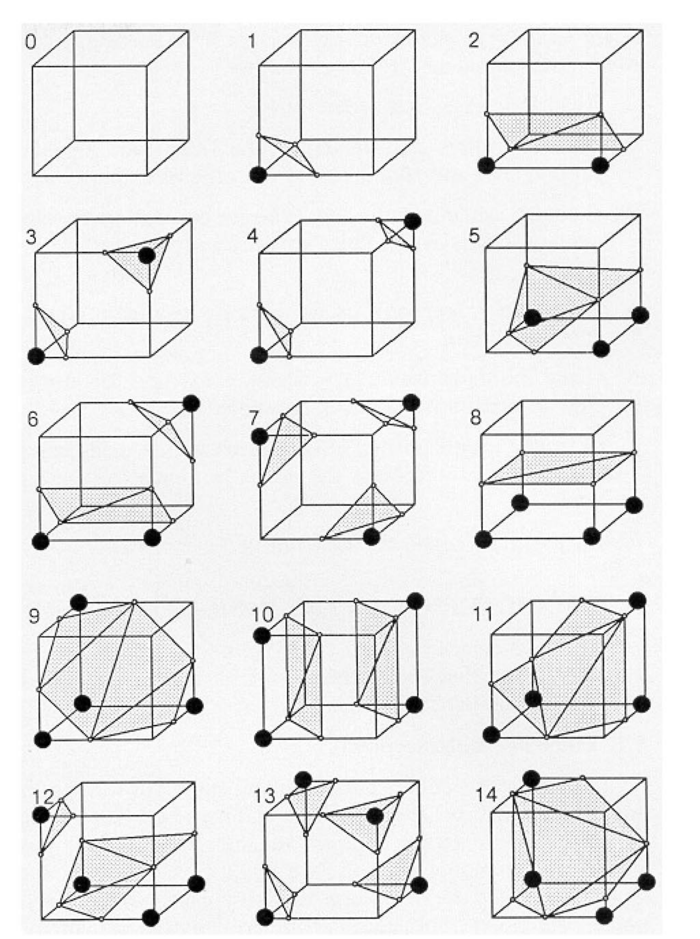


Divide-and-conquer

1. Look at (march) one cell at a time
2. Compare the values at 4 corners with iso-value
3. **Linear interpolate** along edges for intersection points
4. Connect interpolated points together

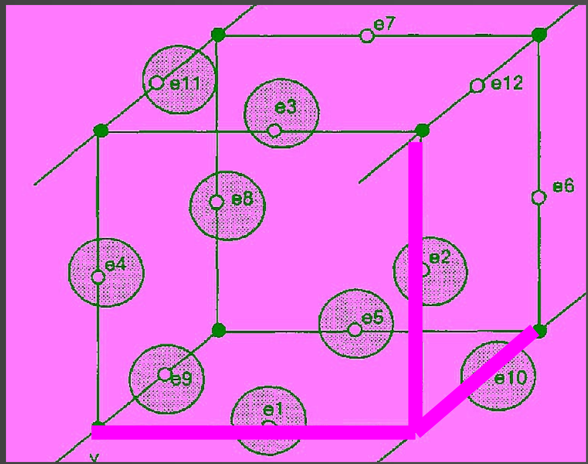
# Marching cubes

- Generalize iso-contour algorithm to 3D
- March cubes one at a time
- Linear interpolation again
- There are more cases:
  - Total of $2^8 = 256$ cases
  - Reduce to 15 topological cases relying on value and rotational symmetry



# Improvements

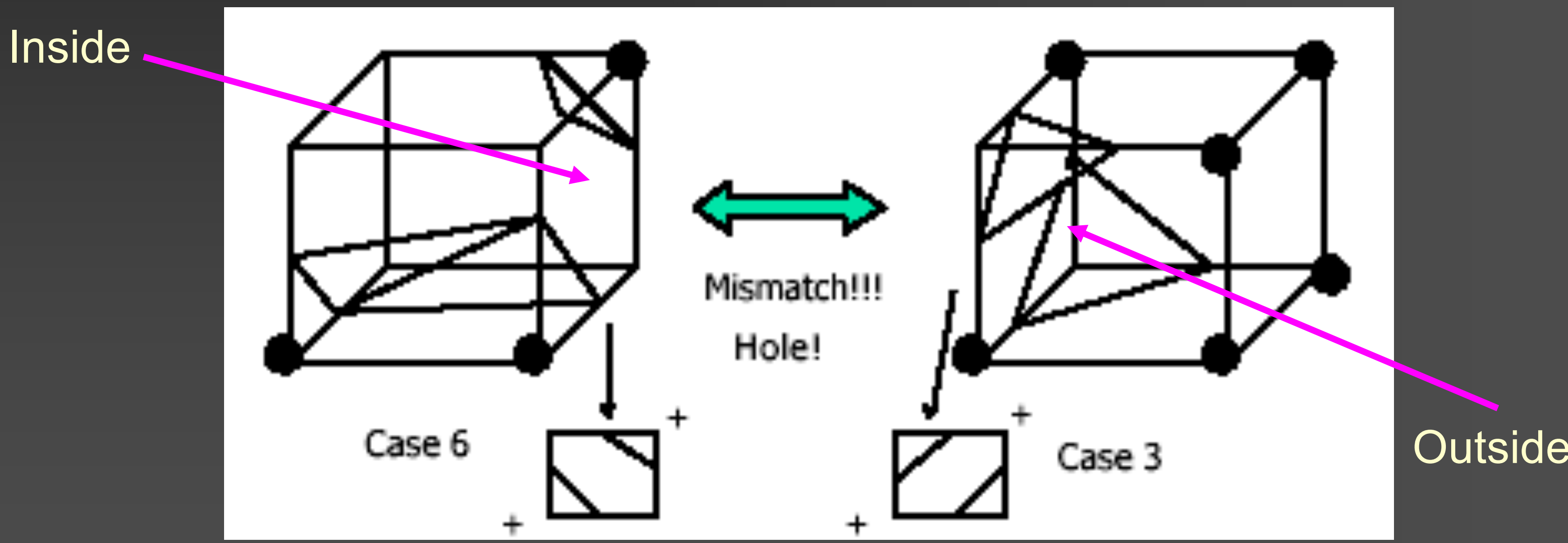
- Exploit spatial coherence
  - e.g., for an interior cube, only three new linear interpolations are needed, if cubes are visited in scan-line order




- Need to find efficient ways for cube traversal
  - Typically, roughly $n^2$ cubes intersect an iso-surface in $n^3$ cube grid
  - e.g., can use an octree to skip empty regions – a great deal of research along this line

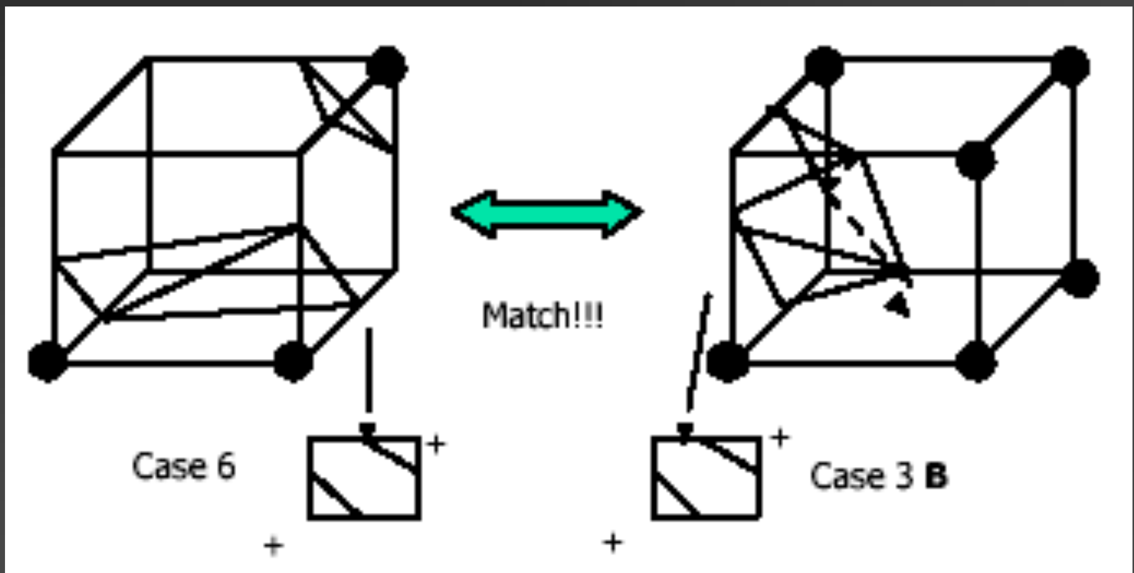
# The ambiguity problem

- Certain marching cube cases have more than one possible triangulations – may create a hole mistakenly



# Fixing the ambiguity problem

- One consistent way to do it



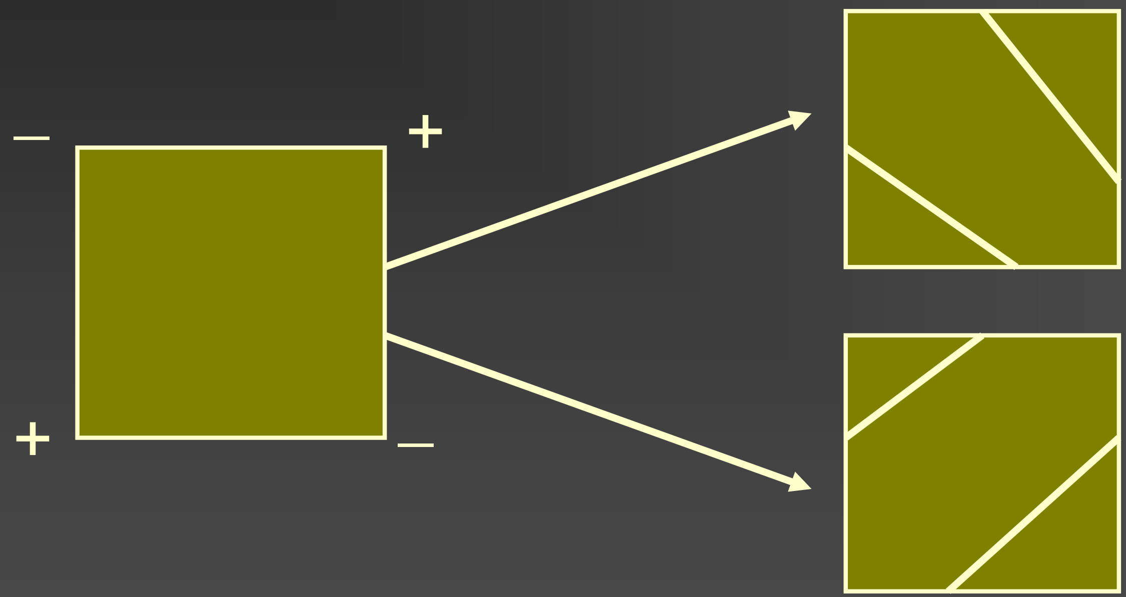

There is another **opposite case:**

keep case 3 and change case 6 to 6A

- Need to come up with these **consistent triangulations**

# Ambiguous faces

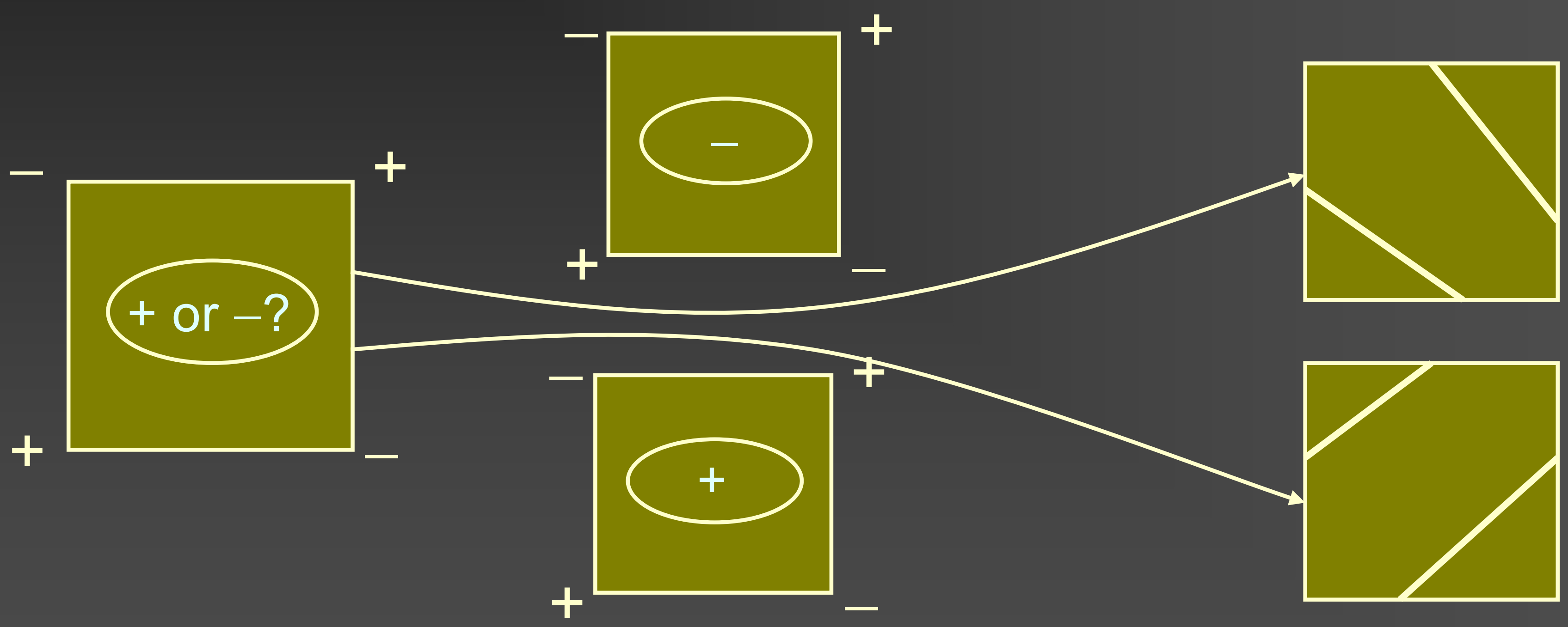
- A face with two opposite vertices having the same sign




- How to resolve this ambiguity? — use the **asymptotic decider** [Nielson & Hamman 91] — somewhat complex and adds cases to original marching cubes

# Asymptotic decider: rough idea

- Need to **examine iso-values inside the face**

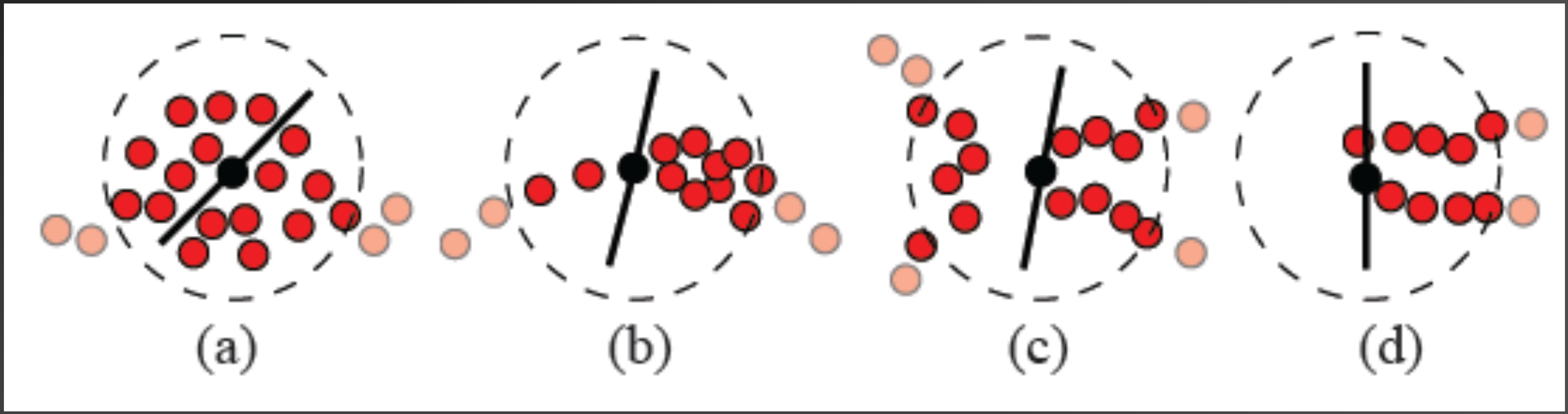



- Inside values are unknown, approximate via **bi-linear interpolation**

# Summary of Hoppe's approach

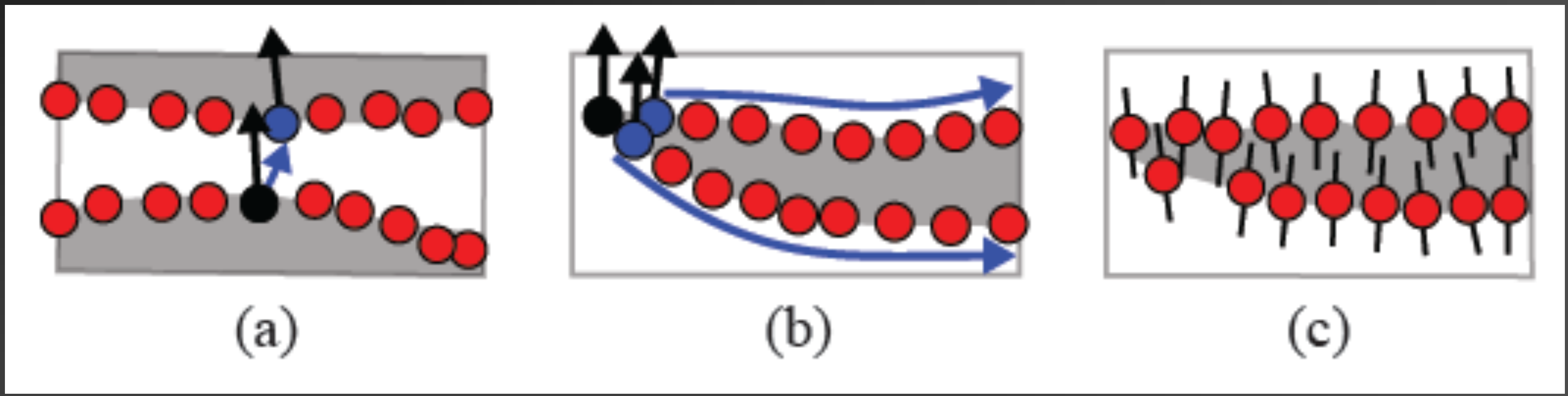
- Surface reconstruction from unorganized points through iso-surface extraction over a signed distance field computed with respect to a set of oriented tangent planes approximating the surface
- Space subdivision helps speed up algorithm (empty cube skipping)
- Constructed surface **approximates** point cloud
- No theoretical guarantee that the surface is correct
- No mechanism for feature preservation

# Unreliability of PCA




- Thick point cloud – need thinning
- Non-uniform point distribution
- Close-by surface sheets

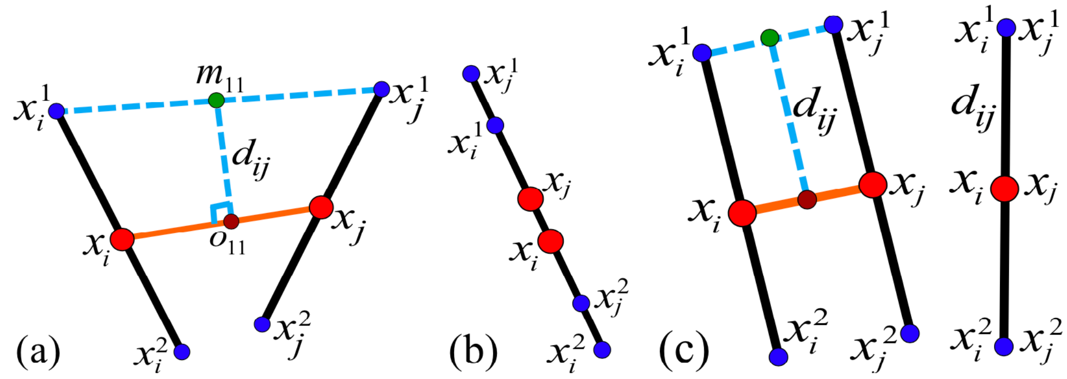
# New propagation cost (aside)




- Again, the close-by surface sheets problem
- Possible solution: also look at the **propagation direction**
- Sharp feature detection: should prevent propagation there

Hui Huang, Dan Li, **Hao Zhang**, Uri Ascher, and Daniel Cohen-Or, "**Consolidation of Unorganized Point Clouds for Surface Reconstruction**," *ACM Trans. on Graphics (Proceeding of SIGGRAPH Asia 2009)*, Article 176.

# New propagation cost (aside)




(a) (b) (c)

$$\mathcal{D}_{ij} = 1 - |\langle \mathbf{v}_i, \mathbf{v}_j \rangle| \cdot \frac{\max_{r,s\in\{1,2\}} \|m_{rs} - o_{rs}\|}{1 + \|x_i - x_j\|}. \qquad (4)$$

Note that $\mathcal{D}_{ij} \in [0,1]$; it combines Euclidean distance (the denominator), angular distance $|\langle \mathbf{v}_i, \mathbf{v}_j \rangle|$, and a third term $d_{ij} = \max_{r,s\in\{1,2\}} \|m_{rs} - o_{rs}\|$, which is designed to weigh in propagation direction.

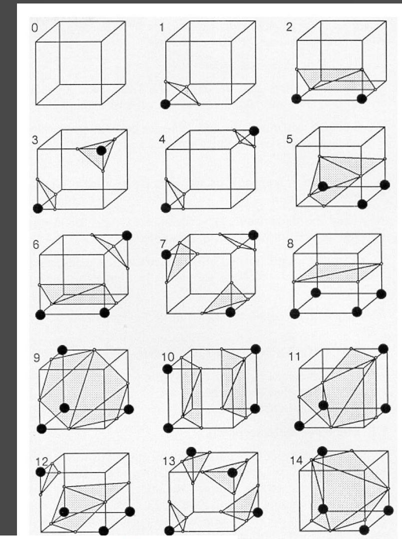
Hui Huang, Dan Li, **Hao Zhang**, Uri Ascher, and Daniel Cohen-Or, "**Consolidation of Unorganized Point Clouds for Surface Reconstruction**," *ACM Trans. on Graphics (Proceeding of SIGGRAPH Asia 2009)*, Article 176.

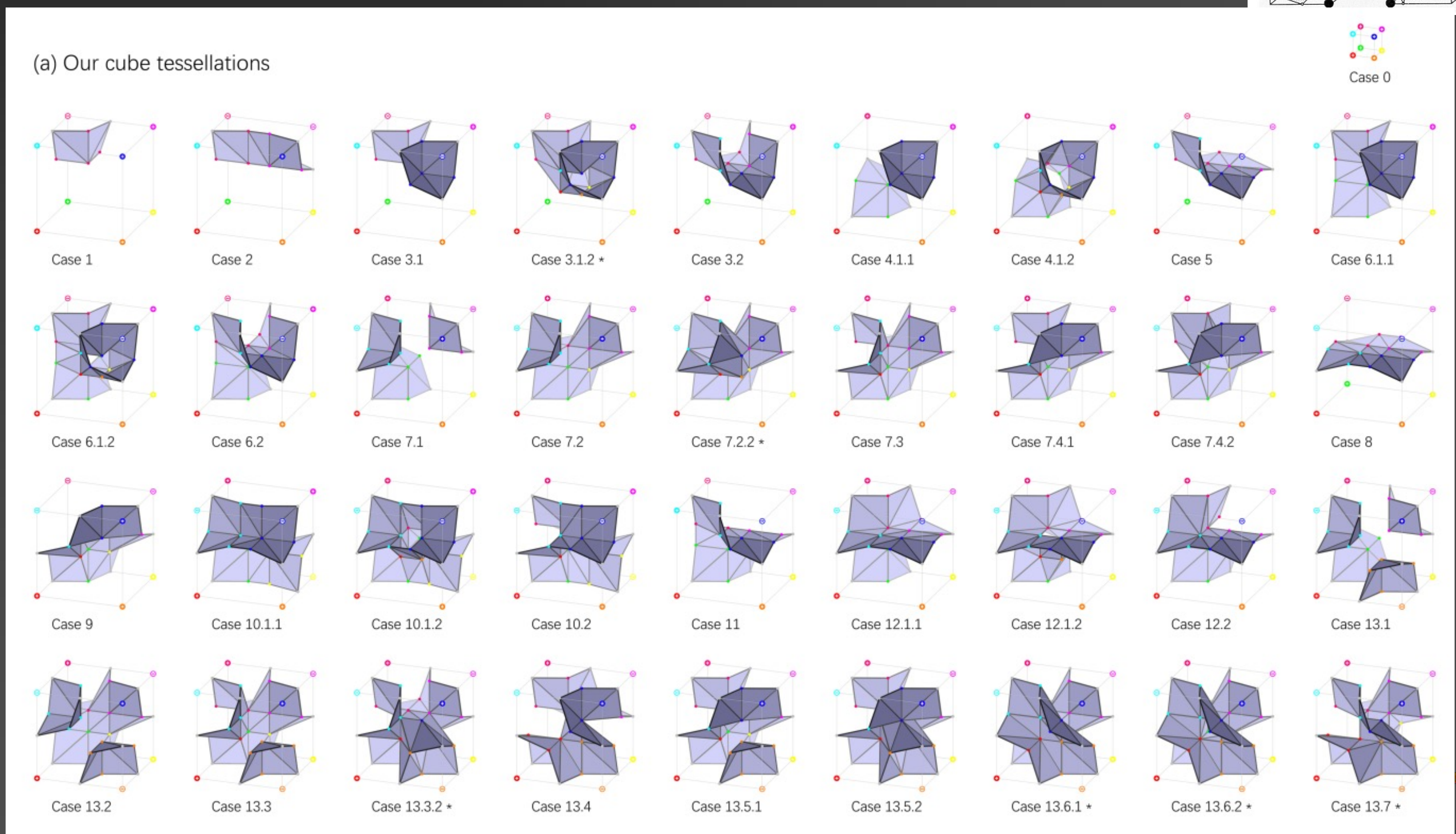
# Other classical approaches

- Voronoi-based with **theoretical guarantee** – by N. Amenta et al., “A New Voronoi-based Surface Reconstruction Algorithm,” *SIGGRAPH* 98
- $\alpha$-shape based approaches – [Bajaj, Bernardini 95]
- **Deform-to-fit** with energy minimization (e.g., inflating a balloon in the object) – [Terzopoulos, Witkin, and Kass 88 & 91, Miller 91]
- Use of **radial basis functions** – [Carr et al. 01, Iske 02]
- Use of **Poisson reconstruction** – [Kazhdan et al. 06]
- Definition of point set surfaces, e.g., **MLS = Moving Least Squares** – [Levin et al. 01, Alexa et al. 02]

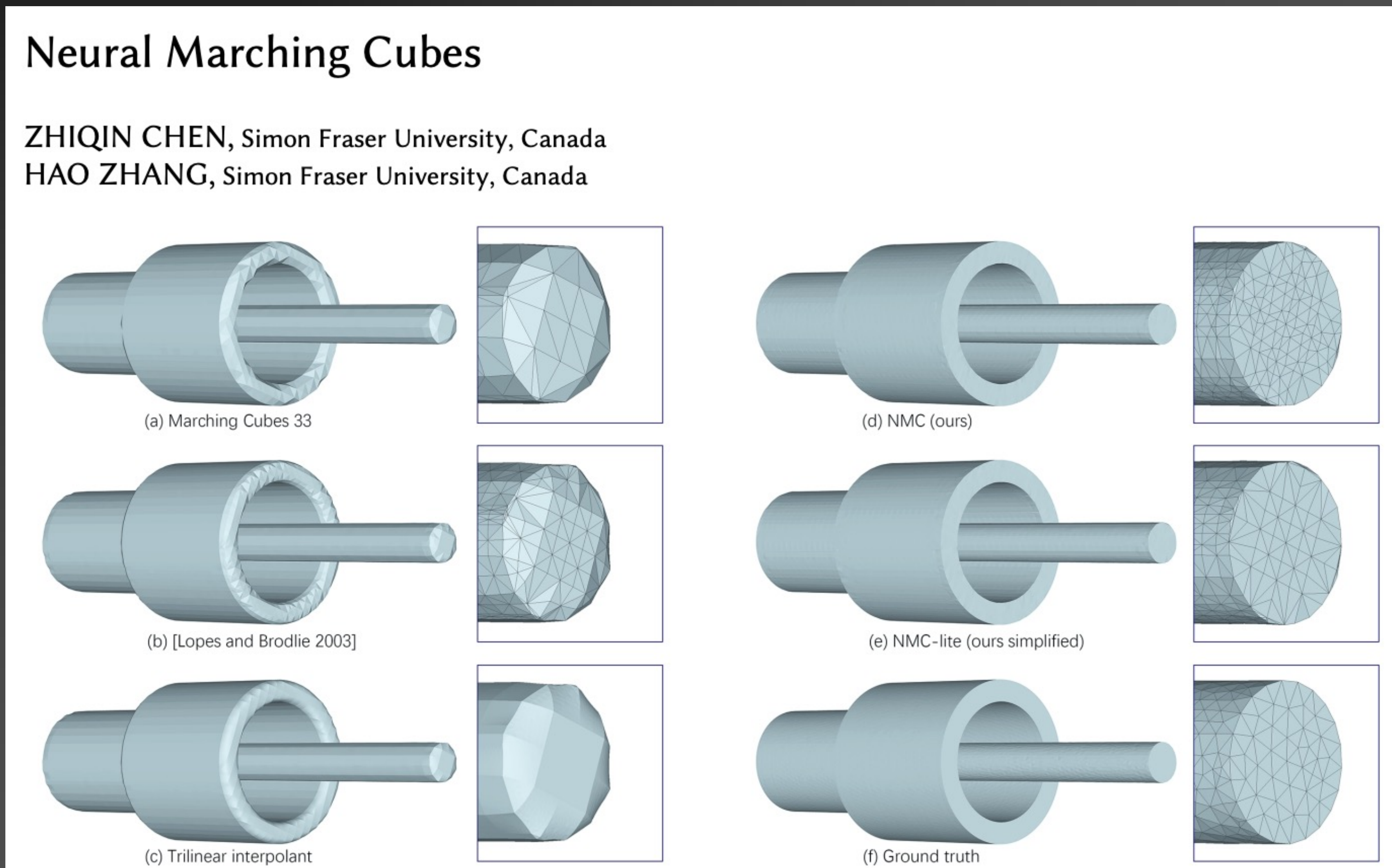
# From MC to NMC

- Use **machine learning** to improve iso-surfacing



# Neural Marching Cubes (NMC) – next week



Neural Marching Cubes

ZHIQIN CHEN, Simon Fraser University, Canada
HAO ZHANG, Simon Fraser University, Canada

(a) Marching Cubes 33
(d) NMC (ours)
(b) [Lopes and Brodlie 2003]
(e) NMC-lite (ours simplified)
(c) Trilinear interpolant
(f) Ground truth