GAMES
Surface Reconstruction

Qixing Huang
July 23th 2021
Goal

Captured point cloud

Reconstructed model
Two approaches

• **Implicit**
  – Signed distance function estimation
  – Mesh approximation
  – Fast and efficient

• **Explicit**
  – Local surface connectivity estimation
  – Computation geometry based
 Implicit-Based Methods
Surface Reconstruction from Unorganized Points

[H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle
SIGGRAPH 1992]
Method Pipeline

• Input:
  – Cloud of points
  – Orientation not required

• Output:
  – Triangular mesh
  – Possible boundary edges

• Guarantees:
  – Manifold
Implicit Surfaces: Regular value

- We would like to represent a function as the zero set of a function \( f : \mathbb{R}^3 \to \mathbb{R} \).

- We say that zero is a regular value of \( f \) if \( \nabla f(p) \neq 0 \) for all points such that \( f(p) = 0 \).
Implicit Function Theorem

$f : \mathbb{R}^3 \to \mathbb{R}$

$g : \mathbb{R}^2 \to \mathbb{R}$

$f(x, y, g(x, y)) = 0$

$\nabla f(\bar{p}) = (f_x, f_y, f_z \neq 0)$

$(p_x, p_y, p_z) = (p_x, p_y, g(p_x, p_y))$
Computation of the Signed Distance Function

• For each sample fit a tangent plane using its k-Nearest Neighbours

• Define a coherent orientation for the tangent plane of all sample points

• For any $p \in R^3$ the signed distance function is given by its closest (oriented) tangent plane.
Tangent Plane Fitting

Find a plane that fits, in the Least Squares sense, to its K-Nearest Neighbours:

$$\min_{\vec{n} \in S^2, a \in \mathbb{R}} \sum_{i=1}^{k} (x_i \cdot \vec{n} - a)^2$$

Properties:

1) This plane passes through the baricenter of the neighbours
   $$o = \frac{1}{k} \sum_{i=1}^{k} x_i$$

2) The normal direction is given an eigenvector of smallest eigenvalue of the covariance matrix
   $$\sum_{i=1}^{k} (x_i - o)(x_i - o)^T$$
Computation of the Signed Distance Function

- For each sample fit a tangent plane using its k-Nearest Neighbours

- Define a coherent orientation for the tangent plane of all sample points

- For any $p \in R^3$ the signed distance function is given by its closest (oriented) tangent plane.
Global Optimization

**Riemannian Graph**

K- Nearest Neighbours

\[ w_{ij} = n_i \cdot n_j \]

**Energy Function**

\[ \Phi : V \to \{-1, 1\} \]

\[ \max \sum_{(i,j) \in E} w_{ij} \Phi_i \Phi_j \]

NP-hard!
Normal Propagation By Geometric Proximity

Euclidean Minimum Spanning Tree

Take $n_j = -n_j$ if $n_i \cdot n_j < 0$
Normal Propagation By Geometric Proximity

Geometric proximity is not a good criteria for normal propagation.
EMST is connected but not dense enough in edges.
Normal Propagation by Plane Parallelism

1) Construct a Riemmanian graph over the plane centers \((o_i's)\) and edge weights \(w_{ij} = 1 - |n_i \cdot n_j|\).

2) Propagate normals along the Minimum Spanning Tree of this graph.
Normal Propagation by Plane Parallelism

Favors normal propagation along low curvature regions.
Normal Propagation by Plane Parallelism
Computation of the Signed Distance Function

• For each sample fit a tangent plane using its k-Nearest Neighbours

• Define a coherent orientation for the tangent plane of all sample points

• For any $p \in \mathbb{R}^3$ the signed distance function is given by its closest (oriented) tangent plane
Sampling Assumptions

$Y \subset M$ is $\rho$-dense

$X \subset \mathbb{R}^3$ is $\rho$-dense $\delta$-noisy

For any point in the surface the closest sample point in $X$ is at most $\rho + \delta$ apart.
Signed Distance Function

Compute $f(p)$:

1) Find the closest center to $p$.

2) Computed the signed distance to the plane.

3) If $d(z, X) < \rho + \delta$ then $f(p) = ((p - o_i) \cdot \vec{n}) \vec{n}$.
   Otherwise, $f(p)$ is undefined.
Remark

Still it provides a good approximation to reconstruct the surface.
Surface Reconstruction: Marching Cubes

1) Define a cube partition of the space. The edge of each cube should be less than $\rho + \delta$.

2) Compute the signed distance function on the cube vertices.

3) Interpolate zero values (i.e., surface intersections) at changing sign edges.

4) Find a triangulation with vertices at zero values.
Surface Reconstruction: Marching Cubes
Surface Reconstruction: Marching Cubes
Data Structure

Group samples by voxels

Constant number of samples per voxel if sampling is uniform.

Complexities:
1) Riemmanian Graph construction: $O(nk)$
2) MST: $O(n \log n)$
3) Normal Propagation: $O(n)$
4) Distance Function Evaluation: $O(1)$
Results
A Volumetric Method for Building Complex Models from Range Images

[Brian Curless and Marc Levoy, SIGGRAPH 1996]

https://graphics.stanford.edu/papers/volrange/slides/
Volumetric method

- For a set of range images, $R_1, R_2, ..., R_N$, we construct signed distance functions $d_1(x), d_2(x), ..., d_N(x)$.

- We combine these functions to generate the cumulative function, $D(x)$.

- We extract the desired manifold as the isosurface, $D(x) = 0$. 

Least squares solution

Range surface #1

Range surface #2

$f(x)$
Error per point

\[ E(f) = \sum_{i=1}^{N} \int d_i^2(x, f) \, dx \]

Error per range surface

Finding the \( f(x) \) that minimizes \( E \) yields the optimal surface.

This \( f(x) \) is exactly the zero-crossing of the combined signed distance functions.
Hole Filling Using Sensor Information

• It is possible to fill holes in the polygonal mesh directly, but such methods:
  – Are hard to make robust
  – Do not use all available information
Without space carving
With space carving

Hole fill

Sensor

- Unseen
- Empty
- Near surface
Drill bit

1.6 mm

Side view of drill bit

Plan view with sensing directions
Plan view

Unorganized points

Range surfaces

Zippered mesh

Volumetric mesh
Side view

Photograph of painted drill bit

Zippered mesh

Volumetric mesh
...to hardcopy

Before hole filling  After hole filling  Hardcopy
Other Print-Based Fitting Methods
Least Squares

- Fit a primitive to the data
- Minimizes squared distances between the points and the primitive

\[ g(x) = a + bx + cx^2 \]

\[ \min_g \sum_i (f_i - g(p_i))^2 \]
Least Squares-- Example

• Primitive is a (univariate) polynomial

\[
g(x) = \left(1, x, x^2, \ldots \right) \cdot c^T
\]

\[
\min \sum_i \left( f_i - \left(1, p_i, p_i^2, \ldots \right) c^T \right)^2 \Rightarrow
\]

\[
0 = \sum_i 2p_i^j \left( f_i - \left(1, p_i, p_i^2, \ldots \right) c^T \right)
\]

• Linear system of equations
Least Squares-- Example

• Resulting system

\[
0 = \sum_i 2p_i^j \left( f_i - \left(1, p_i, p_i^2, \ldots \right)c^T \right) \iff \\
\sum_i \begin{pmatrix}
1 & p_i & p_i^2 & \cdots \\
p_i & p_i^2 & p_i^3 & \cdots \\
p_i^2 & p_i^3 & p_i^4 & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{pmatrix}
\begin{pmatrix}
c_0 \\
c_1 \\
c_2 \\
\vdots
\end{pmatrix}
= 2\sum_i f_i \begin{pmatrix}
1 \\
p_i \\
p_i^2 \\
\vdots
\end{pmatrix}
\]
Radial Basis Functions

• Represent approximating function as
  – Sum of radial functions $r$
  – Centered at the data points $p_i$

\[
f(x) = \sum_{i} w_i r(\|p_i - x\|)
\]
Radial Basis Functions

- Solve \( f_j = \sum w_i r(\|p_i - p_j\|) \)

  to compute weights \( w_i \)

- Linear system of equations

\[
\begin{pmatrix}
    r(0) & r(\|p_0 - p_1\|) & r(\|p_0 - p_2\|) & \cdots \\
    r(\|p_1 - p_0\|) & r(0) & r(\|p_1 - p_2\|) \\
    r(\|p_2 - p_0\|) & r(\|p_2 - p_1\|) & r(0) \\
    \vdots & \vdots & \vdots & \ddots \\
\end{pmatrix}
\begin{pmatrix}
    w_0 \\
    w_1 \\
    w_2 \\
    \vdots
\end{pmatrix}
= \begin{pmatrix}
    f_0 \\
    f_1 \\
    f_2 \\
    \vdots
\end{pmatrix}
\]
Radial Basis Functions

- Solvability depends on radial function
- Several choices assure solvability

\[ r(d) = d^2 \log d \] (thin plate spline)

\[ r(d) = e^{-d^2/h^2} \] (Gaussian)

- \( h \) is a data parameter
- \( h \) reflects the feature size or anticipated spacing among points
Function Spaces!

• Monomial, Lagrange, RBF share the same principle
  – Choose basis of a function space
  – Find weight vector for base elements by solving linear system defined by data points
  – Compute values as linear combinations

• Properties
  – One costly preprocessing step
  – Simple evaluation of function in any point
Function Spaces?

• Problems
  – Many points lead to larger linear systems
  – Evaluation requires global solutions

• Solutions
  – RBF with compact support
    • Matrix is sparse
    • Still: solution depends on every data point, though drop-off is exponential with distance
  – Local approximation approaches
Partition of Unity
Shepard Interpolation

- Approach \( f(x) = \sum_i \phi_i(x) f_i \)

with basis functions \( \phi_i(x) = \frac{\|x - x_i\|^{-p}}{\sum_j \|x - x_j\|^{-p}} \)

- Define \( f(p_i) = f_i = \lim_{x \to p_i} f(x) \)
Shepard Interpolation

- $f(x)$ is a convex combination of $\phi_i$, because all $\phi_i \in [0,1]$ and $\sum \phi_i(x) \equiv 1$
- $f(x)$ is contained in the convex hull of data points
- $|\{p_i\}| > 1 \Rightarrow f(x) \in C^\infty$ and $\nabla f(p_i) = 0$
  - Data points are saddles
- global interpolation
  - every $f(x)$ depends on all data points
- Only constant precision, i.e. only constant functions are reproduced exactly
Shepard Interpolation

Localization:

• Set  \[ f(x) = \sum_i \mu_i(x) \phi_i(x) f_i \]

• with  \[ \mu_i(x) = \begin{cases} \left(1 - \frac{\|x - p_i\|}{R_i}\right)\nu & \text{if } \|x - p_i\| < R_i \\ 0 & \text{else} \end{cases} \]

for reasonable \( R_i \) and \( \nu > 1 \)

\( \Rightarrow \) no constant precision because of possible holes in the data
Partition of Unity Methods
Partial of Unity Methods
Partial of Unity Methods

Subdivide domain into cells
Partial of Unity Methods

Compute local interpolation per cell
Partial of Unity Methods

Blend local interpolations?
Partial of Unity Methods

Subdivide domain into overlapping cells
Partial of Unity Methods

Compute local interpolations
Partial of Unity Methods

Blend local interpolations
Partition of Unity Methods

• Weights should
  – Have the (local) support of the cell
Partition of Unity Methods

• Weights should
  – Sum up to one everywhere (Shepard weights)
  – Have the (local) support of the cell
Moving Least Squares

- Compute a local LS approximation at \( \mathbf{x} \)
- Weight data points based on distance to \( \mathbf{x} \)

\[
\min \sum_i (f_i - g(p_i))^2 \theta(\|\mathbf{x} - p_i\|)
\]
Moving Least Squares

- The set
  \[ f(x) = g_x(x), g_x : \min_{g} \sum_{i} (f_i - g(p_i))^2 \theta(\|x - p_i\|) \]

  is a smooth curve, iff \( \theta \) is smooth.
Moving Least Squares

- Typical choices for $\theta$:
  - $\theta(d) = d^{-r}$
  - $\theta(d) = e^{-d^2/h^2}$

- Note: $\theta_i = \theta(\|x - p_i\|)$ is fixed

- For each $x$
  - Standard weighted LS problem
  - Linear iff corresponding LS is linear
Fitting
Implicits

- Each orientable 2-manifold can be embedded in 3-space
- Idea: Represent 2-manifold as zero-set of a scalar function in 3-space
  - Inside: \( f(x) < 0 \)
  - On the manifold: \( f(x) = 0 \)
  - Outside: \( f(x) > 0 \)
Implicits from point samples

- Function should be zero in data points:
  \[ f(p_i) = 0 \]

- Use standard approximation techniques to find \( f \)

- Trivial solution: \( f = 0 \)

- Additional constraints are needed
Implicits from point samples

- Constraints define inside and outside
- Simple approach (Turk, O’Brien)
  - Sprinkle additional information manually
  - Make additional information soft constraints
Implicits from point samples

- Use normal information
- Normals could be computed from scan
- Or, normals have to be estimated
Estimating normals

- Normal orientation (Implicits are signed)
  - Use inside/outside information from scan
- Normal direction by fitting a tangent
  - LS fit to nearest neighbors
  - Weighted LS fit
  - MLS fit
Estimating normals

- General fitting problem
  \[ \min_{\|n\|=1} \sum_i \langle q - p_i, n \rangle^2 \theta(\|q - p_i\|) \]

- Problem is non-linear because \( n \) is constrained to unit sphere
Estimating normals

• The constrained minimization problem

\[
\min_{\|n\|=1} \sum_i \left\langle q - p_i, n \right\rangle^2 \theta_i
\]

is solved by the eigenvector corresponding to the smallest eigenvalue of the following covariance matrix

\[
\sum_i (q_p_i - q_p_i)^T \theta_i
\]

which is constructed as a sum of weighted outer products.
Implicits from point samples

- Compute non-zero anchors in the distance field
- Use normal information directly as constraints
  \[ f(p_i + n_i) = 1 \]
Implicits from point samples

- Compute non-zero anchors in the distance field
- Compute distances at specific points
  - Vertices, mid-points, etc. in a spatial subdivision
Computing Implicits

- Given N points and normals $\mathbf{p}_i, \mathbf{n}_i$ and constraints $f(\mathbf{p}_i) = 0, f(\mathbf{c}_i) = d_i$
- Let $\mathbf{p}_{i+N} = \mathbf{c}_i$
- An RBF approximation
  \[ f(\mathbf{x}) = \sum_{i} w_i \theta(\|\mathbf{p}_i - \mathbf{x}\|) \]

leads to a system of linear equations
Computing Implicits

- Practical problems: $N > 10000$
- Matrix solution becomes difficult
- Two solutions
  - Sparse matrices allow iterative solution
  - Smaller number of RBFs
Computing Implicits

- Sparse matrices
  \[
  \begin{pmatrix}
  \theta(0) & \theta(\|p_0 - p_1\|) & \theta(\|p_0 - p_2\|) & \cdots \\
  \theta(\|p_1 - p_0\|) & \theta(0) & \theta(\|p_1 - p_2\|) \\
  \theta(\|p - p_0\|) & \theta(\|p_2 - p_1\|) & \theta(0) \\
  \vdots & & & \ddots
  \end{pmatrix}
  \]

- Needed: \( d > c \rightarrow r(d) = 0, r'(c) = 0 \)

- Compactely supported RBFs
Computing Implicits

- Smaller number of RBFs
- Greedy approach (Carr et al.)
  - Start with random small subset
  - Add RBFs where approximation quality is not sufficient
RBF Implicits - Results

Image courtesy: Greg Turk
Multi-Level Partition of Unity
Overview

• Goal:
  – Use multi-level partition of unity (MPU) implicit surface to construct surface models

• 3 Key Concepts:
  – Piecewise quadratic functions used as local estimates
  – Weighing functions that blend these local shape functions.
  – Octree subdivision that adapts based on shape complexity.

• Flexibility
  – Accurate representation of sharp features (edges, corners)

• Adaptive approximation based on required accuracy
  – Determines space/time complexity
Advantages of Implicit Functions

- Edit surfaces using standard implicit modeling operations: shape blending, offsets, deformations
Method Summary: Setup

• Given: set of points with normals to indicate surface orientation

• Partition of unity: set of weighing functions that sum to one at all points in the domain

• MPU implicit: adaptive error-controlled approximation of signed distance function from surface
  – Surface is zero-level of the distance function.
Method Summary: Algorithm

• To create implicit representation:
  – Octree-based subdivision of bounding box for entire point set
  – At each cell, fit a piecewise quadratic function (local shape function)
    • Signed distance function: 0 near points, positive inside, negative outside
  – If shape function isn’t accurate enough, subdivide further until desired accuracy is achieved
  – In common boundary between cells, shape functions are blended together according to weights from partition of unity functions

• Global implicit of function is given by blending of local shape functions at the leaves of the octree
Partition of Unity

• Generate weight functions:
  – For approximation: use quadratic B-spline \( b(t) \)
    \[
    w_i(x) = b \left( \frac{3 |x - c_i|}{2R_i} \right)
    \]
  – For interpolation: use inverse-distance singular weights
    \[
    w_i(x) = \left[ \frac{(R_i - |x - c_i|)}{R_i |x - c_i|} \right]^2, \text{ where } (a)_+ = \begin{cases} a & a > 0 \\ 0 & \text{else} \end{cases}
    \]

• For interpolation: use inverse-distance singular weights
  \[
  \Omega \subset \bigcup_i \text{ supp}(w_i)
  \]
Partition of Unity

- Blend local functions using smooth, local weights that add up to 1
  - Partition of unity functions
    \[
    \sum_i \phi_i = 1 \text{ on } \Omega \quad \phi_i(x) = \frac{w_i(x)}{\sum_{j=1}^{n} w_j(x)}
    \]
- Define set of local shape functions \( V_i \)
- Approximation of a function defined on domain
  \[
  Q_i \in V_i \quad f(x) \approx \sum_i \phi_i(x)Q_i(x)
  \]
Adaptive Octree

- Octree-based subdivision. Each cell has center $c$ and diagonal length $d$
- Define the support radius for the cell’s weight function: $R = \alpha d$
  - Bigger alpha $\rightarrow$ smoother interpolation/approximation, slower computation
  - Time complexity quadratic on $\alpha$
- Must have at least $N_{\text{min}}$ points in the sphere to estimate shape function
  - If not enough, iteratively increase radius

$$\hat{R} = \hat{R} + \lambda R$$

$\alpha = 0.75 \quad \lambda = 0.1$
Adaptive Octree

• Local max-norm approximation error estimated based on Taubin distance

\[ \epsilon = \max_{|p_i - c| < R} \frac{|Q(p_i)|}{|\nabla Q(p_i)|} \]

• If error is larger than a threshold \( \varepsilon_0 \), subdivide the cell
Algorithm: Pseudocode

\[ S_{wQ} = \sum w_i(x)Q_i(x) \]
\[ S_w = \sum w_i(x) \]
\[ f(x) \approx \frac{S_{wQ}}{S_w} = \frac{\sum w_i(x)Q_i(x)}{\sum w_i(x)} \]

MPUapprox(x, \varepsilon_0)

\[ d = \|c - x\|; \]
if \( d > R \) then
  return;
end if
if \( Q \) is not created yet then
  Create \( Q \) and compute \( \varepsilon \);
end if
if \( \varepsilon > \varepsilon_0 \) then
  if (No childs) then
    Create childs;
  end if
  for each child
    child->MPUapprox(x, \varepsilon_0);
  end for
else
  \[ S_{wQ} = S_{wQ} + w(d, R) \cdot Q(x) \]
  \[ S_w = S_w + w(d, R) \]
end if
Local Shape Functions – Details in the paper

• General 3D quadric
  – Larger parts of the surface: unbounded, more than one sheet B

• Bivariate quadratic polynomial in local coordinates
  – Local smooth patch C

• Piecewise quadric surface to fit sharp features
  – Edges, corners
Representation power

Eye from Stanford’s reconstruction of Michalangelo’s David (scanned at 1mm resolution). Right: The eye is reconstructed as an MPU implicit with relative accuracy $10^{-4}$. 
Robustness

Offsetting of a knot model. The distance function to the knot is approximated by $w = f(x,y,z)$.
Projection-based Approaches
Projection

• Idea: Map space to surface
• Surface is defined as fixpoints of mapping
Surface definition

• Projection procedure (Levin)
  – Local polynomial approximation
    • Inspired by differential geometry
  – “Implicit” surface definition
  – Infinitely smooth &
  – Manifold surface
Surface definition

- Projection procedure (Levin)
  - Local polynomial approximation
    - Inspired by differential geometry
  - “Implicit” surface definition
- Infinitely smooth &
- Manifold surface
Surface definition

- Constructive definition
  - Input point $\mathbf{r}$
  - Compute a local reference plane $H_r = \langle q, n \rangle$
  - Compute a local polynomial over the plane $G_r$
  - Project point $\mathbf{r}' = G_r(0)$
  - Estimate normal
Local reference surface

- Find plane \( H_r = \langle q, n \rangle + D \)
- \( \min_{q, \| n \| = 1} \sum_i \langle q - p_i, n \rangle^2 \theta\left(\|q - p_i\|\right) \)
- \( \theta(d) = e^{d^2/h^2} \)
  - \( h \) is feature size/point spacing
  - \( H_r \) is independent of \( r \)'s distance
  - Manifold property
Local Reference Plane

- Computing reference plane
  - Non-linear optimization problem
- Minimize independent variables:
  - Over $n$ for fixed distance $\|r - q\|$  
  - Along $n$ for fixed direction $n$
  - $q$ changes $\rightarrow$ the weights change
  - Only iterative solutions possible
Local reference plane

- Practical computation
  - Minimize over $n$ for fixed $q$
    - Eigenvalue problem
  - Translate $q$ so that $r = q + \|r - q\|n$
    - Effectively changes $\|r - q\|$
  - Minimize along $n$ for fixed direction $n$
    - Exploit partial derivative
Rejecting the point

- MLS polynomial over $H_r$

\[- \min_{G \in \Pi_d} \sum_i \left( \langle q - p_i, n \rangle - G(p_i | H_r) \right)^2 \theta(\|q - p_i\|) \]

- LS problem
- $r' = G_r(0)$

- Estimate normal
Spatial data structure

- Regular grid based on support of $\theta$
  - Each point influences only 8 cells
- Each cell is an octree
  - Distant octree cells are approximated by one point in center of mass
Defining point-set surfaces

[Amenta and Kil 05]
Poisson Surface Reconstruction
Poisson surface reconstruction

- Michael Kazhdan, M. Bolitho, and H. Hoppe, SGP 2006
- Implementation included in Meshlab
- Relevant works
  - Poisson mesh editing [SIGGRAPH 2004, SGP 2004]
  - Poisson image editing [SIGGRAPH 2003]
Poisson surface reconstruction

• Indicator Function
  – reconstruct the surface by solving for the indicator function of the shape
  – Assume normal as inputs

\[ \chi_M(p) = \begin{cases} 
1 & \text{if } p \in M \\
0 & \text{if } p \notin M 
\end{cases} \]
Problem

• Fit the indicator function to a set of oriented normal
  – Fitting should be robust

Oriented points $\xrightarrow{\text{}}$ Indicator function $\chi_M$
Gradient Relationship

• There is a relationship between the normal field and gradient of indicator function

\[ \nabla \chi_M \]
Integration

- Represent the points by a vector field
- Find the function $\chi$ whose gradient best approximates $\vec{V}$:

$$ \min_{\chi} \| \nabla \chi - \vec{V} \| $$
Integration

• Represent the points by a vector field
• Find the function $\chi$ whose gradient best approximates $\vec{V}$:

$$\min_{\chi} \left\| \nabla \chi - \vec{V} \right\|$$

• Applying the divergence operator, we can transform this into a Poisson problem:

$$\nabla \cdot (\nabla \chi) = \nabla \cdot \vec{V} \iff \Delta \chi = \nabla \cdot \vec{V}$$
Implementation

- Given the Points:
  - Set octree
  - Compute vector field
  - Compute indicator function
  - Extract iso-surface
Implementation: Adapted Octree

• Given the Points:
  – Set octree
  – Compute vector field
  – Compute indicator function
  – Extract iso-surface
Implementation: Vector Field

- **Given the Points:**
  - Set octree
  - Compute vector field
    - Define a function space
    - Splat the samples
  - Compute indicator function
  - Extract iso-surface
Implementation: Vector Field

• Given the Points:
  – Set octree
  – Compute vector field
    • Define a function space
    • Splat the samples
  – Compute indicator function
  – Extract iso-surface
Implementation: Vector Field

• Given the Points:
  – Set octree
  – Compute vector field
    • Define a function space
    • Splat the samples
  – Compute indicator function
  – Extract iso-surface
Implementation: Vector Field

- Given the Points:
  - Set octree
  - Compute vector field
    - Define a function space
    - Splat the samples
  - Compute indicator function
  - Extract iso-surface
Implementation: Vector Field

• Given the Points:
  – Set octree
  – Compute vector field
    • Define a function space
    • Splat the samples
  – Compute indicator function
  – Extract iso-surface
Implementation: Vector Field

• Given the Points:
  – Set octree
  – Compute vector field
    • Define a function space
    • Splat the samples
  – Compute indicator function
  – Extract iso-surface
Implementation: Vector Field

- **Given the Points:**
  - Set octree
  - Compute vector field
    - Define a function space
    - Splat the samples
  - Compute indicator function
  - Extract iso-surface
Implementation: Vector Field

• Given the Points:
  – Set octree
  – Compute vector field
    • Define a function space
    • Splat the samples
  – Compute indicator function
  – Extract iso-surface
Implementation: Vector Field

• Given the Points:
  – Set octree
  – Compute vector field
  – Compute indicator function
    • Compute divergence
    • Solve Poisson equation
  – Extract iso-surface
Implementation: Vector Field

- Given the Points:
  - Set octree
  - Compute vector field
  - Compute indicator function
    - Compute divergence
    - Solve Poisson equation
  - Extract iso-surface
Implementation: Vector Field

• Given the Points:
  – Set octree
  – Compute vector field
  – Compute indicator function
    • Compute divergence
    • Solve Poisson equation
  – Extract iso-surface
Summary

Oriented points $\vec{V}$
Indicator gradient $\nabla \chi_M$
Indicator function $\chi_M$
Surface $\partial M$
Michelangelo’s David

- 215 million data points from 1000 scans
- 22 million triangle reconstruction
- Compute Time: 2.1 hours
- Peak Memory: 6600MB
David – Chisel marks
David – Drill Marks
David – Eye
Scalability – Buddha Model

The graph illustrates the relationship between the number of triangles and the time taken and peak memory usage. As the number of triangles increases, both the time taken and peak memory usage increase linearly.

- **Time Taken**: The time taken to process the triangles, represented by the blue line.
- **Peak Memory Usage**: The maximum memory usage, represented by the purple line.

The x-axis represents the number of triangles, while the y-axis shows the time taken in milliseconds and peak memory usage in megabytes.
Stanford Bunny

Power Crust

FastRBF

MPU

VRIP

FFT Reconstruction

Poisson Reconstruction
VRIP Comparison

VRIP

Poisson Reconstruction
Neural Implicits

- We will cover this later
Meshing
Algorithms

- Primal methods
  - Marching Squares (2D), Marching Cubes (3D)
  - Placing vertices on grid edges

- Dual methods
  - Dual Contouring (2D,3D)
  - Placing vertices in grid cells
Dual Contouring (2D)

- For each grid cell with a sign change
  - Create one vertex
- For each grid edge with a sign change
  - Connect the two vertices in the adjacent cells with a line segment
Dual Contouring (2D)

- For each grid cell with a sign change
  - Create one vertex
- For each grid edge with a sign change
  - Connect the two vertices in the adjacent cells with a line segment
Dual Contouring (2D)

• Creating the vertex within a cell
  – Compute one point on each grid edge with a sign change (by linear interpolation)
    • There could be more than two sign-changing edges, so >2 points possible
  – Take the centroid of these points
Dual Contouring (3D)

- For each grid cell with a sign change
  - Create one vertex (same way as 2D)
- For each grid edge with a sign change
  - Create a quad (or two triangles) connecting the four vertices in the adjacent grid cubes
  - No look-up table is needed!
• The two outputs have a dual structure
  – Vertices and quads of Dual Contouring correspond (roughly) to un-triangulated polygons and vertices produced by Marching Cubes
Primal vs. Dual

- **Marching Cubes**
  - ✓ Always manifold
  - ✗ Requires look-up table in 3D
  - ✗ Often generates thin and tiny poly

- **Dual Contouring**
  - ✗ Can be non-manifold
  - ✓ No look-up table needed
  - ✓ Generates better-shaped polygon
Primal vs. Dual

- **Marching Cubes**
  - ✓ Always manifold
  - ✗ Requires look-up table in 3D
  - ✗ Often generates thin and tiny polygons
  - ✗ Restricted to uniform grids

- **Dual Contouring**
  - ✗ Can be non-manifold
  - ✓ No look-up table needed
  - ✓ Generates better-shaped polygons
  - ✓ Can be applied to any type of grid