Implicit Surface Reconstruction

Avinash Sharma
Center for Visual IT, KCIS, IIIT Hyderabad

Content is largely borrowed from Authors and other publicly available resources

Input Data

- Set of range scans
  - Scans registered in common coordinate system

- Set of irregular sample points
  - Typically with normal vectors
  - More general

- Surface Extraction

This week

Point Cloud

- In how many ways can we connect the points?
    - Many candidate shapes
    - Ill Posed Problem!!!

Ill Posed Problem

Many candidate shapes for the reconstruction problem

Priors

Exhaustive Connection

Smooth

Piece-wise Smooth

Triangulation Issues

- Triangulation is sensitive to noise, sampling pattern, omissions etc
- Can we more robustly recover the underlying surface?

Slides adapted from Kazhdan, Bolitho and Hoppe

Priors

Exhaustive Connection

Smooth

Piece-wise Smooth

Surfaces: Representations

- Parametric Surfaces:
  - Vector-valued parameterization ‘f’
  - Maps 2D parameter domain $\Omega \subset \mathbb{R}^2$ to the surface $\mathcal{S} = \mathbf{f}(\Omega) \subset \mathbb{R}^3$

$$\mathbf{f} : \Omega \to \mathcal{S}$$

- Implicit Surfaces:
  - Defined by the zero level set of a scalar-valued function

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid F(\mathbf{x}) = 0\}$$

Shape Representations

- Explicit/parametric representation
  - Image of parametrization

$$F(x,y) = 0$$

$$f(t) = (x(t), y(t)) = (r\cos(t), r\sin(t))$$

$$f([0,2\pi))$$

- Implicit representation
  - Zero set of distance function

F(x,y) < 0

$$F(x,y) = \sqrt{x^2 + y^2} - r$$

Implicit Function Approach

Define a function with positive values inside the model and negative values outside

Level Set

- ***c*-Level set**: The set of points where a function takes a constant value *c*

Level Set

- ***c*-Level set**: The set of points where a function takes a constant value *c*
  - **Isocontour**: Level set of a 2D function
  - **Isosurface**: Level set of a 3D function

Toby Breckon

Implicit Function Approach

Define a function with positive values inside the model and negative values outside

Extract the zero-set

Key Idea

- Reconstruct the surface of the model by solving for the indicator function of the shape

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

Indicator function

c

M

In practice, we define the indicator function to be -1/2 outside the shape and 1/2 inside, so that the surface is the zero level set. We also smooth the function a little, so that the zero set is well defined.

Challenge

- How to construct the indicator function?

Gradient Relationship

- There is a relationship between the normal field at the shape boundary, and the gradient of the (smoothed) indicator function

Oriented points

Indicator gradient

$\nabla \chi_M$

Poisson Surface Reconstruction

- Represent the point normals by a vector field *V*
- Find the function χ whose gradient best approximates *V*

$$\min_{\chi} \|\nabla \chi - V\|^2$$

- Applying the divergence operator,

$$\nabla \cdot (\nabla \chi) = \nabla \cdot V \quad \Rightarrow \quad \Delta \chi = \nabla \cdot V$$

Laplacian $\Delta \chi = \frac{\partial^2 \chi}{\partial x^2} + \frac{\partial^2 \chi}{\partial y^2} + \frac{\partial^2 \chi}{\partial z^2}$

Divergence $\nabla \cdot V = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z}$

Linear Least Squares

- Need to solve set of equations $A\mathbf{x} = \mathbf{b}$ in a least squares
- Minimize $||\mathbf{r}|| = ||\mathbf{b} - A\mathbf{x}||$^2
- The minimum is achieved when

$$A^T A\mathbf{x} = A^T\mathbf{b} \quad \text{or} \quad \mathbf{x} = (A^T A)^{-1} A^T \mathbf{b}$$

Discrete Derivative

Continuous

- Function: f
- Operator: $\frac{d}{dx}$
- Applying operator:

$$\frac{df}{dx} = f'$$

Discrete

- Vector: $\mathbf{f} = [f(x_1), f(x_2) \ldots f(x_n)]$
- Matrix:

$$A = \frac{1}{h}\begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ & 0 & -1 & \cdots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & & & -1 & 1 \\ 0 & 0 & \cdots & & 0 & -1 \end{bmatrix}$$

- Applying matrix:

$$A\mathbf{f} = \mathbf{f'}$$

Example: Discrete Derivative

Continuous

Discrete

Discrete 2nd Derivative

Continuous

- Function: f
- Operator: $\dfrac{d^2}{dx^2}$
- Applying operator:

$$\frac{d^2 f}{dx^2} = f''$$

Discrete

- Vector: $\mathbf{f} = [\, f(x_1), f(x_2) \ldots f(x_n)]$
- Matrix:

$$L = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 & 0 \\ & 1 & -2 & \cdots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & & & -2 & 1 \\ 0 & 0 & \cdots & & 1 & -2 \end{bmatrix}$$

- Applying matrix:

$$L\mathbf{f} = \mathbf{f}''$$

Discrete 2D Laplacian

- The Laplacian is computed via differences of a cell from its neighbours

Operators in higher dimensions

- The underlying function space can have a higher-dimensional domain

Continuous function

Discrete approximation

$$\begin{bmatrix} -4 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & -4 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & -4 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & -4 & 1 & \cdot & 1 & \cdot & \cdot \\ \cdot & 1 & \cdot & 1 & -4 & 1 & \cdot & 1 & \cdot \\ \cdot & \cdot & 1 & \cdot & 1 & -4 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & -4 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & -4 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & -4 \end{bmatrix}$$

2D discrete Laplace operator

Operators in higher dimensions

Laplacian (of scalar-valued function) maps scalar field to scalar field

Original function

After applying Laplacian

Operators in Non-Euclidean Domain

$$\nabla = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & +1 \end{bmatrix}$$

$$(\nabla \boldsymbol{f})(e_{ij}) = f(v_j) - f(v_i)$$

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & +1 \end{bmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \end{pmatrix} = \begin{pmatrix} f(2) - f(1) \\ f(1) - f(3) \\ f(3) - f(2) \\ f(4) - f(2) \end{pmatrix}$$

Operators in Non-Euclidean Domain

$$\nabla = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & +1 \end{bmatrix}$$

$$\mathbf{L} = \nabla^\top \nabla$$

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$(\mathbf{L}\boldsymbol{f})(v_i) = \sum_{v_j \sim v_i} w_{ij}(f(v_i) - f(v_j))$$

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

First step of Poisson reconstruction

- Define function space for χ and V
- Accurate solution only needed near surface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - Compute vector field
  - Compute indicator function
  - Extract isosurface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - **Set Octree**
  - Compute vector field
  - Compute indicator function
  - Extract isosurface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - **Compute vector field**
  - Compute indicator function
  - Extract isosurface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - **Compute vector field**
  - Compute indicator function
  - Extract isosurface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - **Compute vector field**
  - Compute indicator function
  - Extract isosurface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - **Compute vector field**
  - Compute indicator function
  - Extract isosurface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - **Compute vector field**
  - Compute indicator function
  - Extract isosurface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - **Compute vector field**
  - Compute indicator function
  - Extract isosurface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - **Compute vector field**
  - Compute indicator function
  - Extract isosurface

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - **Compute vector field**
  - Compute indicator function
  - Extract isosurface

$$\vec{V}(q) \equiv \sum_{s \in S} \sum_{o \in \mathrm{Ngbr}_D(s)} \alpha_{o,s} F_o(q) s.\vec{N}$$

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - Compute vector field
  - **Compute indicator function**
  - Extract isosurface

$$\Delta \chi_M = \nabla \cdot V$$

Kazhdan, Bolitho and Hoppe

Implementation of Poisson reconstruction

- Given Input points:
  - Set Octree
  - Compute vector field
  - Compute indicator function
  - **Extract isosurface**

Kazhdan, Bolitho and Hoppe

Isocontours in 2D grid

- **Data:** 2D structured grid of scalar values

Slides adapted from Toby Breckon

Isocontours

- The 5-level set:

Splits edge asymetrically, since 5 is closer to 6 than to 2

Bisects the edge, since 5 is equidistant from 3 and 7

Slides adapted from Toby Breckon

Where is the intersection?

- Find location of contour intersection with edge by interpolating vertex values

The value 5 splits the edge in a 1:1 ratio

The value 5 splits the edge in a 1:3 ratio

Isocontours: Ambiguity

- Where is the contour?

Triangular cell:
No ambiguities

"Split" green (inner) region

or

Square cell:
2 ambiguous cases

"Join" green (inner) region

Slides adapted from Toby Breckon

Isocontours: Ambiguity

- Where is the contour?

Join

Split

Slides adapted from Toby Breckon

Isocontours: Cell Confgurations

No intersections

1 vertex diferent

2 vertices diferent

Ambiguous case

2^4 = 16 different possibilities, reducible to just 6 distinct cases after factoring out symmetries

Slides adapted from Toby Breckon

Marching Squares Algorithm

- Select a starting cell
- Calculate inside/outside state for each vertex
- Classify cell confguration
  - Determine which edges are intersected
- Find exact locations of edge intersections
- Link up intersections to produce contour segment(s)
- Move (or “march”) into next cell and repeat
  - … until all cells have been visited

Example: Marching Squares

Find 5-contour of function represented by its values at vertices of a uniform grid

Slides adapted from Toby Breckon

Step 1: Classify vertices

Green: inside
Red: outside

Step 2: Classify cells

No intersections

Adjacent edges

Opposite edges

Ambiguous

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

No intersections

Adjacent edges

Opposite edges

Ambiguous

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

No intersections

Adjacent edges

Opposite edges

Ambiguous

Arbitrarily choose to split here, instead of join. We could also have gone the other way.

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Step 3: Interpolate contour intersections

Slides adapted from Toby Breckon

Resolving ambiguities

No intersections

Adjacent edges

Opposite edges

Ambiguous

Choosing to join instead

Slides adapted from Toby Breckon

In 3D: Marching Cubes

Exactly the same algorithm, but cells are now cubes (15 distinct confgurations) and output is triangles (or a polygon mix)

Wikipedia

In 3D: Marching Cubes

Koen Samyn, https://www.youtube.com/watch?v=LfttaAepYJ8

Marching Cubes: Estimating Normals

- We could estimate normals from the generated mesh, but the density function has more information
- Recall: The normal to the surface is the gradient of the density function

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$

- We will estimate the gradient from the grid of values

Normals at Cube Vertices

Discrete approximation to the gradient at the blue cube vertex

$$n_x = \frac{f(i+1, j, k) - f(i-1, j, k)}{2\,\Delta x}$$

$$n_x = \frac{f(i, j+1, k) - f(i, j-1, k)}{2\,\Delta y}$$

$$n_y = \frac{f(i, j, k+1) - f(i, j, k-1)}{2\,\Delta z}$$

(Better approximations are possible)

Normals at Mesh Vertices

Example: Diferent level sets of CT scan

Bone surface

Soft tissue surface

Lorensen and Cline, “Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm”, SIGGRAPH '87

Marching Cubes: Pros and Cons

- **Pros:**
  - Local computations only, so needs very little working memory and has good cache coherence
  - Works well with grid-structured input
    - E.g. medical scans
  - Simple to implement

- **Cons:**
  - No adaptive resolution, produces lots of triangles
  - Telltale patterned artifacts, since cells are cubes and output triangles are generated from a uniform grid.
  - No principled approach to resolve ambiguities