Surface Reconstruction
FFT vs. Poisson

M. Kazhdan, M. Bolitho, R. Burns, Ming Chuang, H. Hoppe, K. Dalal, and A. Klein

Multilevel Streaming for Out-of-Core Surface Reconstruction. Bolitho et al. (2007)
Unconstrained Isosurface Extraction on Arbitrary Octrees. Kazhdan et al. (2007)
Parallel Poisson Surface Reconstruction. Bolitho et al. (2009)
Screened Poisson Surface Reconstruction. Kazhdan et al. (2013)
Accurate Isosurface Interpolation with Hermite Data. Fuhrmann et al. (2015)
Poisson Surface Reconstruction with Boundary Constraints. Kazhdan et al. (2020)
Outline

• Introduction
• Related Work
• Math Review
• Two Reconstruction Algorithms
• Results
Surface Reconstruction

Given a set of *oriented* points from the surface of a model, generate a water-tight surface that passes through/near the samples.
Applications

- Surface Blending

Disjoint Model  “Zippered” Model
Applications

- Surface Blending
- Hole-Filling
Applications

- Surface Blending
- Hole-Filling
- Compression

Geometry + Topology Representation ➔ Geometry Representation
Applications

- Surface Blending
- Hole-Filling
- Compression
- Simplification

Original Model
871,000 Triangles

Simplified Model
95,000 Triangles
Applications

- Surface Blending
- Hole-Filling
- Compression
- Simplification
- Scan Reconstruction

http://createdigitalmotion.com/
http://www.xbox.com/
http://graphics.stanford.edu/projects/mich/
Related Work

Three general approaches:

1. **Computational Geometry**
   - Boissonnat, 1984
   - Edelsbrunner, 1984
   - Amenta et al., 1998
   - Dey et al., 2003

2. **Surface Fitting**
   - Terzopoulos et al., 1991
   - Chen et al., 1995

3. **Implicit Function Fitting**
   - Hoppe et al., 1992
   - Curless et al., 1996
   - Whitaker, 1998
   - Carr et al., 2001
   - Davis et al., 2002
   - Ohtake et al., 2004
   - Turk et al., 2004
   - Shen et al., 2004
Related Work

3. Implicit Function Fitting
   - Use the (oriented) points to define a function that is negative inside the shape and positive outside.
Related Work

3. Implicit Function Fitting
   – Use the (oriented) points to define a function that is negative inside the shape and positive outside.
   – Extract the zero-crossing iso-surface.
Goal: Reconstruct the simplest implicit function – the *indicator function*:

\[
\chi_D (x, y, z) = \begin{cases} 
1 & \text{if } (x, y, z) \in D \\
0 & \text{otherwise}
\end{cases}
\]
Outline

• Introduction
• Related Work
• **Math Review**
  – FFT: What can we compute?
  – Poisson: What are we representing?
• Results
Math Review (Gradient)

Given a real-valued function in three-space:

\[ F: \mathbb{R}^3 \rightarrow \mathbb{R} \]

we compute the gradient of the function by taking the vector of partial derivatives:

\[ \nabla F = \left( \frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z} \right) \]
Math Review (Gradient)

Given a real-valued function in three-space:
\[ F: \mathbb{R}^3 \rightarrow \mathbb{R} \]
we compute the \textit{gradient} of the function by taking the vector of partial derivatives:
\[
\nabla F = \left( \begin{array}{c}
\frac{\partial F}{\partial x} \\
\frac{\partial F}{\partial y} \\
\frac{\partial F}{\partial z} 
\end{array} \right)
\]

Intuitively:
At every point, this gives the direction of (steepest) change of \( F \).
Math Review (Gradient)

Example:
The gradient of \( F(x, y, z) = e^{ax+by+cz} \)
is:

\[
\nabla F = \begin{pmatrix}
a \cdot e^{ax+by+cz} \\
b \cdot e^{ax+by+cz} \\
c \cdot e^{ax+by+cz}
\end{pmatrix}
\]
Math Review (Divergence)

Given a vector-valued function in three-space:

\[ \vec{V} = \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \]

we compute the divergence of the function by taking the sum of the partial derivatives:

\[ \nabla \cdot \vec{V} = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z} \]
Math Review (Divergence)

Given a vector-valued function in three-space:

\[ \mathbf{V} = \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} : \mathbb{R}^3 \to \mathbb{R}^3 \]

we compute the **divergence** of the function by taking the sum of the partial derivatives:

\[ \nabla \cdot \mathbf{V} = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z} \]

Intuitively:
At every point, this gives the difference between the amount of flow into the point and the amount of flow out of it.
Math Review (Divergence)

Example:
The divergence of

\[ \vec{V}(x, y, z) = \begin{pmatrix} a \cdot e^{ax+by+cz} \\ b \cdot e^{ax+by+cz} \\ c \cdot e^{ax+by+cz} \end{pmatrix} \]

is:

\[ \nabla \cdot \vec{V} = (a^2 + b^2 + c^2) \cdot e^{ax+by+cz} \]
Math Review (Laplacian)

Given a real-valued function in three-space:
\[ F: \mathbb{R}^3 \rightarrow \mathbb{R} \]

the **Laplacian** of the function is the divergence of its gradient:
\[ \Delta F = \nabla \cdot \nabla F \]
Math Review (Laplacian)

Given a real-valued function in three-space:

\[ F: \mathbb{R}^3 \rightarrow \mathbb{R} \]

the \textit{Laplacian} of the function is the divergence of its gradient:

\[ \Delta F = \nabla \cdot \nabla F \]

\[ = \left( \begin{array}{c} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial z} \end{array} \right) \cdot \left( \begin{array}{c} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial z} \end{array} \right) \]
Math Review (Laplacian)

Given a real-valued function in three-space:

\[ F: \mathbb{R}^3 \rightarrow \mathbb{R} \]

the Laplacian of the function is the divergence of its gradient:

\[ \Delta F = \nabla \cdot \nabla F \]

\[
= \nabla \cdot \left( \begin{array}{c} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial z} \end{array} \right) = \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + \frac{\partial^2 F}{\partial z^2}
\]
Math Review (Laplacian)

Given a real-valued function in three-space:

\[ F: \mathbb{R}^3 \rightarrow \mathbb{R} \]

the **Laplacian** of the function is the divergence of its gradient:

\[ \Delta F = \nabla \cdot \nabla F \]

\[
\begin{align*}
\left( \frac{\partial F}{\partial x} \right) & = \nabla \cdot \left( \frac{\partial F}{\partial x} \right) = \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + \frac{\partial^2 F}{\partial z^2} \\
\end{align*}
\]

Intuitively:
At every point, this measures the difference between the value of \( F \) at the point and the average value of the neighbors.
Example:
The Laplacian of
\[ F(x, y, z) = e^{ax+by+cz} \]
is:
\[ \Delta F = (a^2 + b^2 + c^2) \cdot e^{ax+by+cz} \]
Math Review (Divergence Theorem)

Given a solid $D \subset \mathbb{R}^3$, the boundary of $D$ is denoted by $\partial D$. 
Math Review (Divergence Theorem)

Given a vector-valued function in three-space:
\[ \mathbf{V} : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \]
and given a solid \( D \subset \mathbb{R}^3 \), we can integrate the divergence of \( \mathbf{V} \) over the interior of \( D \):
\[ \int_D (\nabla \cdot \mathbf{V}) (p) \, dp. \]

This equals the integral of \( \mathbf{V} \) across \( \partial D \):
\[ \int_D (\nabla \cdot \mathbf{V}) (p) \, dp = \int_{\partial D} \langle \mathbf{V}(p), \mathbf{n}(p) \rangle \, dp \]
where \( \mathbf{n}(p) \) is the normal at \( p \in \partial P \).
Given integers $l$, $m$, and $n$ we can define a (complex-valued) function in three-space:

$$\zeta_{lmn} : \mathbb{R}^3 \rightarrow \mathbb{C}$$

$$(x, y, z) \mapsto e^{i(lx+my+nz)}$$

The set of functions $\{\zeta_{lmn}\}_{l,m,n \in \mathbb{Z}}$ is called the Fourier basis.
Math Review (Fourier Transform)

If $F: \mathbb{R}^3 \to \mathbb{R}$ is a function in three-space, we can write it out as as the linear combination of the Fourier basis functions:

$$F(x, y, z) = \sum_{l,m,n \in \mathbb{Z}} \hat{F}(l,m,n) \cdot \zeta_{lmn}(x, y, z)$$

The (complex-valued) coefficients $\hat{F}(l, m, n)$ are called the Fourier coefficients of $F$. 
Math Review (Fourier Transform)

\[ F(x, y, z) = \sum_{l,m,n \in \mathbb{Z}} \hat{F}(l,m,n) \cdot \zeta_{lmn}(x, y, z) \]

The Fourier coefficients are obtained by integrating against the (conjugate) Fourier basis functions:

\[ \hat{F}(l, m, n) = \int F(x, y, z) \cdot \overline{\zeta_{lmn}}(x, y, z) \]

- Doing this over an \( N \times N \times N \) grid requires \( O(N^3) \) time.
- Computing for all \( O(N^3) \) Fourier coefficients takes \( O(N^6) \) time.
- Similarly, given the coefficients, evaluating at \( N \times N \times N \) grid points also takes \( O(N^6) \) time.
Math Review (Fourier Transform)

\[ F(x, y, z) = \sum_{l,m,n \in \mathbb{Z}} \hat{F}(l, m, n) \cdot \zeta_{lmn}(x, y, z) \]

The Fourier coefficients are obtained by integrating against the (conjugate) Fourier basis functions:

\[ \hat{F}(l, m, n) = \int F(x, y, z) \cdot \overline{\zeta_{lmn}(x, y, z)} \]

✓ Using the **Fast Fourier Transform** all \( O(N^3) \) coefficients can be computed in \( O(N^3 \log N) \) time.

✓ Similarly, given the Fourier coefficients, the values of the function can be obtained in \( O(N^3 \log N) \) time.
Math Review (Fourier Transform)

Given the \((l, m, n)\)-th Fourier basis function:
\[
\zeta_{lmn} = e^{i(lx+my+nz)}
\]

The gradient of \(\zeta_{lmn}\) is:
\[
\nabla \zeta_{lmn} = \begin{pmatrix}
il \cdot \zeta_{lmn} \\
im \cdot \zeta_{lmn} \\
in \cdot \zeta_{lmn}
\end{pmatrix}
\]

The Laplacian of \(\zeta_{lmn}\) is:
\[
\Delta \zeta_{lmn} = \nabla \cdot \nabla \zeta_{lmn} = -(l^2 + m^2 + n^2) \zeta_{lmn}
\]
Math Review (Fourier Transform)

\[ \Delta \zeta_{lmn} = -(l^2 + m^2 + n^2) \zeta_{lmn} \]

⇒ Computing the Laplacian of a function \( F \) is the same as multiplying the Fourier coefficients:

\[ \hat{F}(l, m, n) \rightarrow -(l^2 + m^2 + n^2) \hat{F}(l, m, n) \]

⇒ Computing the inverse of the Laplacian of a function \( F \) is the same as multiplying the Fourier coefficients:

\[ \hat{F}(l, m, n) \rightarrow -\frac{\hat{F}(l, m, n)}{(l^2 + m^2 + n^2)} \]
Outline

• Introduction
• Related Work
• Math Review
• Two Reconstruction Algorithms
  – FFT: What can we compute?
  – Poisson: What are we representing?
• Results
Motivation

What can we do with a set of oriented points?
Motivation

What can we do with a set of oriented points?

Divergence Theorem:
Given a vector field $\vec{V}$ and a domain $D$:

$$\int_D \nabla \cdot \vec{V} = \int_{\partial D} \langle \vec{V}, \vec{n} \rangle$$
Motivation

What can we do with a set of oriented points?

**Divergence Theorem:**

Given a vector field $\vec{V}$ and a domain $D$:

$$\int_D \nabla \cdot \vec{V} = \int_{\partial D} \langle \vec{V}, \hat{n} \rangle$$

$$\int_D \nabla \cdot \vec{V} \approx \sum_{i=1}^{A} \frac{1}{n} \langle \vec{V}(p_i), \hat{n}_i \rangle$$

$$\int_D \nabla \cdot \vec{V}$$
Motivation

What can we do with a set of oriented points?

Divergence Theorem:

Without explicitly knowing what the interior is, with a set of oriented points we can approximate integrals over the interior.

With enough integrals, we should be able to figure out the interior.
Computing $\chi_D$ (Step 1)

Computing the indicator function is equivalent to computing its Fourier coefficients:

$$\chi_D(x, y, z) = \sum_{l,m,n} \hat{\chi}_D(l, m, n) \cdot \zeta_{lmn}(x, y, z)$$
Computing $\chi_D$ (Step 2)

Computing the Fourier coefficients is equivalent to computing a set of volume integrals:

$$\hat{\chi}_D(l, m, n) = \int_{[0,1]^3} \chi_D(x, y, z) \cdot \zeta_{lmn}(x, y, z) \, dx \, dy \, dz$$

$$= \int_D \zeta_{lmn}(x, y, z) \, dx \, dy \, dz$$

since the indicator function is one inside of $D$ and zero outside.
Computing $\chi_D$ (Step 3)

Surface Integration
Let $\vec{V}_{lmn}(x, y, z)$ be a vector field whose divergence is the $(l, m, n)$-th complex exponential:

$$\left( \nabla \cdot \vec{V}_{lmn} \right) (x, y, z) = \zeta_{lmn}(x, y, z).$$

Applying the Divergence Theorem, the volume integral can be expressed as a surface integral:

$$\int_D \zeta_{lmn}(x, y, z) dx dy dz = \int_{\partial D} \langle \vec{V}_{lmn}, \vec{n} \rangle dp$$
Reconstruction Algorithm

Given a set of oriented point samples $\{(p_j, n_j)\}$

• We approximate the Fourier coefficients:

$$\hat{\chi}_D (l, m, n) \approx \sum_{j=1}^{k} \langle \vec{V}_{lmn}(p_k), n_k \rangle$$

• Use that to define the indicator function:

$$\chi_D(x, y, z) = \sum_{l,m,n} \hat{\chi}_D (l, m, n) \cdot \zeta_{lmn}(x, y, z)$$

• And extract the iso-surface:

$$\partial D = \{ p | \chi_D(p) = 0.5 \}$$
Implementation

We need to find vectors field whose divergences are the complex exponentials:

\[
\nabla \cdot \mathbf{V}_{lmn}(x, y, z) = \zeta_{lmn}(x, y, z) = e^{-i(lx+my+nz)}
\]

There are many of these:

\[
\mathbf{V}_{lmn}(x, y, z) = \begin{pmatrix}
\frac{i \cdot e^{-i(lx+my+nz)}}{l} \\
0 \\
0
\end{pmatrix}
\]

\[
\mathbf{V}_{lmn}(x, y, z) = \begin{pmatrix}
0 \\
\frac{i \cdot e^{-i(lx+my+nz)}}{m} \\
0
\end{pmatrix}
\]

\[
\mathbf{V}_{lmn}(x, y, z) = \begin{pmatrix}
0 \\
0 \\
\frac{i \cdot e^{-i(lx+my+nz)}}{n}
\end{pmatrix}
\]

\[
\mathbf{V}_{lmn}(x, y, z) = \frac{i}{3} \begin{pmatrix}
e^{-i(lx+my+nz)} \\
e^{-i(lx+my+nz)} \\
e^{-i(lx+my+nz)} \\
\end{pmatrix}
\]

\[
\mathbf{V}_{lmn}(x, y, z) = \frac{i}{l + m + n} \begin{pmatrix}
e^{-i(lx+my+nz)} \\
e^{-i(lx+my+nz)} \\
e^{-i(lx+my+nz)} \\
\end{pmatrix}
\]
Implementation

We need to find vectors field whose divergences are the complex exponentials:

\[
(\nabla \cdot \vec{V}_{lmn})(x, y, z) = \overline{\zeta_{lmn}}(x, y, z) = e^{-i(lx+my+nz)}
\]

There are many of these:

\[
\vec{V}_{lmn}(x, y, z) = \begin{pmatrix}
i \cdot e^{-i(lx+my+nz)} \\
l \\
0
\end{pmatrix}
\]

All of these commute with translation:

Translating the points and computing the indicator function.

\[
\uparrow
\]

Computing the indicator function and translating.
Implementation

We need to find vectors field whose divergences are the complex exponentials:

\[ \nabla \cdot \vec{V}_{lmn}(x, y, z) = \zeta_{lmn}(x, y, z) = e^{-i(lx + my + nz)} \]

But there is only one vector field that commutes with both translation and rotation:

\[ \vec{V}_{lmn}(x, y, z) = \frac{1}{l^2 + m^2 + n^2} \begin{pmatrix} il \cdot e^{-i(lx + my + nz)} \\ im \cdot e^{-i(lx + my + nz)} \\ in \cdot e^{-i(lx + my + nz)} \end{pmatrix} \]
## Implementation

<table>
<thead>
<tr>
<th>Does not Commute:</th>
<th>Commutes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{V}_{lmn}(x, y, z) = \frac{i}{l + m + n} \begin{pmatrix} e^{-i(lx + my + nz)} \ e^{-i(lx + my + nz)} \ e^{-i(lx + my + nz)} \end{pmatrix}$</td>
<td>$\vec{V}_{lmn}(x, y, z) = \frac{i}{l^2 + m^2 + n^2} \begin{pmatrix} l \cdot e^{-i(lx + my + nz)} \ m \cdot e^{-i(lx + my + nz)} \ n \cdot e^{-i(lx + my + nz)} \end{pmatrix}$</td>
</tr>
</tbody>
</table>

### Examples:

- **0°**
- **30°**
- **45°**
## Implementation

<table>
<thead>
<tr>
<th>Does not Commute:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{V}_{l,m,n}(x, y, z) = \frac{i}{l + m + n} \begin{pmatrix} e^{-i(lx + my + nz)} \ e^{-i(lx + my + nz)} \ e^{-i(lx + my + nz)} \end{pmatrix}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commutes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{V}_{l,m,n}(x, y, z) = \frac{i}{l^2 + m^2 + n^2} \begin{pmatrix} l \cdot e^{-i(lx + my + nz)} \ m \cdot e^{-i(lx + my + nz)} \ n \cdot e^{-i(lx + my + nz)} \end{pmatrix}$</td>
</tr>
</tbody>
</table>
Efficient Implementation

Explicitly summing over all the points to find each Fourier coefficient would be too slow:

\[ \hat{\chi}_V(l, m, n) = \sum_{j=1}^{K} \langle \vec{V}_{lmn}(p_j), n_j \rangle \]

Assuming \( O(N^2) \) points, computing the \( O(N^3) \) Fourier coefficients takes \( O(N^5) \) time.

Using the Fast Fourier Transform, this can be reduced to a convolution:

\[ O(N^5) \rightarrow O(N^3 \log N) \]
Properties

Advantages:

– Mathematical Correctness:
  • For a sufficiently dense and uniform sampling, the indicator function is guaranteed to be accurate
Properties

Advantages:
– Computational Simplicity:
  • Splat
  • Convolve
  • Extract

\[ S = \{ p | \chi_D(p) = 0.5 \} \]

\[ \chi_D(p) = \vec{V}(p) \ast \vec{F}(p) \]

\[ \vec{V}(p) \]
Properties

Disadvantages

• For an $O(N^2)$ reconstruction:
  ✖ Temporal Complexity: $O(N^3 \log N)$
  ✖ Spatial Complexity: $O(N^3)$

On a machine with 16GB of RAM, reconstruction resolutions will be limited to $1024^3$ voxel grids.
Properties

Disadvantages

• For an $O(N^2)$ reconstruction:
  × Temporal Complexity: $O(N^3 \log N)$
  × Spatial Complexity: $O(N^3)$
× Resolution is not adaptive
Properties

Disadvantages

• For an $O(N^2)$ reconstruction:

• Temporal Complexity: $O(N^3 \log N)$

• Spatial Complexity: $O(N^3)$

• Resolution is not adaptive

The fixed resolution precludes an adaptively smoothed reconstruction.

To adapt to the sampling density:

1. Weight the samples’ contribution.
2. Reconstruct more coarsely/smoothly in regions of sparser sampling.
Outline

• Introduction
• Related Work
• Math Review
• Two Approaches
  – FFT: What can we compute?
  – Poisson: What are we representing?
• Results
Motivation

What information about the indicator function does a set of oriented points represent?
Motivation

What information about the indicator function does a set of oriented points represent?

**Indicator Function Gradient:**

Because $\chi_D$ is piecewise constant, its gradient will be zero almost everywhere...
Motivation

What information about the indicator function does a set of oriented points represent?

Indicator Function Gradient:

... Which looks distinctively like the oriented (inward-pointing) surface samples.
Approach

We reconstruct by solving for the indicator function whose gradient is most similar to the vector field $\vec{V}$ represented by the samples:

$$\chi_D = \arg \min_F \int \left\| \nabla F - \vec{V} \right\|^2 dp$$

This can be done by:

1. Computing the divergence of $\vec{V}$
2. Solving the Poisson equation:

$$\chi_D = \Delta^{-1} (\nabla \cdot \vec{V})$$
Implementation: Adapted Octree

Given the Points:

- Set up an octree
- Compute vector field
- Compute indicator function
- Extract iso-surface
Implementation: Vector Field

Given the Points:

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

Given the Points:

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

Given the Points:

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

Given the Points:

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

Given the Points:

• Set up an octree
• Compute vector field
  – Define a function basis
  – Splat the samples
• Compute indicator function
• Extract iso-surface
Implementation: Vector Field

Given the Points:

• Set up an octree
• Compute vector field
  – Define a function basis
  – Splat the samples
• Compute indicator function
• Extract iso-surface
Implementation: Vector Field

Given the Points:

- Set up an octree
- Compute vector field
  - Define a function basis
  - Splat the samples
- Compute indicator function
- Extract iso-surface
Implementation: Vector Field

Given the Points:

• Set up an octree
• Compute vector field
  – Define a function basis
  – Splat the samples
• Compute indicator function
• Extract iso-surface
Implementation: Indicator Function

Given the Points:

• Set up an octree
• Compute vector field
• Compute indicator function
  – Compute divergence
  – Solve Poisson equation
• Extract iso-surface
Implementation: Indicator Function

Given the Points:

• Set up an octree
• Compute vector field
• **Compute indicator function**
  – Compute divergence
  – Solve Poisson equation
• Extract iso-surface
Implementation: Surface Extraction

Given the Points:

- Set up an octree
- Compute vector field
- Compute indicator function
- Extract iso-surface
Cascadic Multigrid

Efficiency:
– We can leverage the hierarchical structure of the octree to solve the linear system using a multigrid solver:
  • Solve at coarser resolutions
  • Up-sample the coarse solution to get a good initial guess for the next level.
Properties

Advantages:
– Solving over an octree, reduces both the space and time complexity of the reconstruction algorithm:
  • Space: $O(N^3) \rightarrow O(N^2)$
  • Time: $O(N^3 \log N) \rightarrow O(N^2)$
Properties

Advantages:
– We can adapt the octree to the sampling density to better handle non-uniform samples
Advantages:
– Solving a screened-Poisson equation:

\[ \chi_D = \arg \min_F \left[ \alpha \sum_{p \in P} (F(p) - 0.5)^2 + \int \| \nabla F - \vec{V} \|^2 \right] \]

we can constrain the reconstructed surface to stay close to the input samples.
Properties

Advantages:

– Solving a screened-Poisson equation:

\[ \chi_D = \arg \min_F \left[ \alpha \sum_{p \in P} (F(p) - 0.5)^2 + \int \left\| \nabla F - \vec{V} \right\|^2 \right] \]

we can constrain the reconstructed surface to stay close to the input samples.
Properties

Advantages:
- Given information about where the surface cannot be, we can add constraints that the solution to the Poisson equation should be zero (outside).
Properties

Advantages:
- Given information about where the surface cannot be, we can add constraints that the solution to the Poisson equation should be zero (outside) there.
Properties

Advantages:
– Sorting the octree nodes by $x$-index, we can design a streaming and parallel implementation.

Pajarola, 2005
Properties

Advantages:
– Sorting the octree nodes by $x$-index, we can design a streaming and parallel implementation.
Advantages:
- Sorting the octree nodes by $x$-index, we can design a streaming and parallel implementation.
Properties

What is the difference between the FFT and Poisson solvers?
Properties

What is the difference between the FFT and Poisson solvers?

- The “splatting” operator computes the divergence and then applies the inverse of the Laplacian.

\[ \left\langle \vec{V}_{lmn}(x, y, z), \vec{n} \right\rangle \]

\[ = \left( \frac{-1}{l^2 + m^2 + n^2} \right) \left( \begin{pmatrix} -il \cdot e^{-i(lx+my+nz)} \\ -im \cdot e^{-i(lx+my+nz)} \\ -in \cdot e^{-i(lx+my+nz)} \end{pmatrix}, \vec{n} \right) \]

\[ = \left( \frac{-1}{l^2 + m^2 + n^2} \right) \left\langle i(l, m, n), \vec{n} \cdot e^{-i(lx+my+nz)} \right\rangle \]

Inverse Laplacian  Divergence
Properties

What is the difference between the FFT and Poisson solvers?

– Alternatively, the equivalence of the normals with the gradient of the indicator function derives from the Divergence Theorem.

Indicator function $\chi_D$

Indicator gradient $\nabla \chi_D$
Outline

• Introduction
• Related Work
• Math Review
• Two Approaches
  – FFT: What can we compute?
  – Poisson: What are we representing?
• Results
FFT Results (Resolution)

100,000 Points

100,000 Points

100,000 Points

res=64^3
tris=11,900
time=0:01

res=128^3
tris=49,556
time=0:03

res=256^3
tris=200,692
time=0:17
Memory Usage: FFT vs. Poisson

The graph compares the peak memory usage of the Fast Fourier Transform (FFT) and Poisson methods as a function of the number of triangles. The y-axis represents the peak memory usage in MB, while the x-axis represents the number of triangles. The FFT method shows a steeper increase in memory usage compared to the Poisson method, indicating higher memory demands for the FFT.
Michelangelo’s David

Effective Resolution: $2048^3$

Projected FFT Recon:
  Time: ~6hrs
  Memory: ~90GB

Poisson Recon:
  Time: ~2.5hrs
  Memory: 4.4GB
  Triangles: 22 million

216x10^6 points (4.8 GB)
Michelangelo’s David

216x10^6 points (4.8 GB)
Michelangelo’s David

216x10^6 points (4.8 GB)
Michelangelo’s David

216x10^6 points (4.8 GB)
### Streaming Results

<table>
<thead>
<tr>
<th>Res.</th>
<th>Octree Memory</th>
<th>Peak Memory</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>48 way, 49 way</td>
<td>521 way, 309 way</td>
<td>0.53, 0.50</td>
</tr>
<tr>
<td>512</td>
<td>168 way, 188 way</td>
<td>278 way, 442 way</td>
<td>0.68, 0.65</td>
</tr>
<tr>
<td>1024</td>
<td>702 way, 818 way</td>
<td>213 way, 1,285 way</td>
<td>1.20, 1.05</td>
</tr>
<tr>
<td>2048</td>
<td>3,070 way, 3,695 way</td>
<td>212 way, 4,442 way</td>
<td>3.33, 2.65</td>
</tr>
<tr>
<td>4096</td>
<td>13,367 way, N/A</td>
<td>427 way, N/A</td>
<td>12.6, N/A</td>
</tr>
<tr>
<td>8192</td>
<td>39,452 way, N/A</td>
<td>780 way, N/A</td>
<td>32.3, N/A</td>
</tr>
</tbody>
</table>

Out-of-Core (Streaming) Reconstruction

In-Core Reconstruction

216x10^6 points (4.8 GB)
Streaming Poisson Results

In-Core Reconstruction
Peak Mem: 4.4 GB (Depth 11)

Out-of-Core (Streaming) Reconstruction
Peak Mem: 0.8 GB (Depth 13)
Parallel Poisson Results

- Ideal Scaling
- Distributed Lucy (1 Machine)
- Distributed Lucy (3 Machines)
- Distributed David (3 Machines)

Observed Speedup vs. Number of Processors

94x10^6 points
1x10^9 points
Conclusion

We can robustly reconstruct surfaces in a memory footprint smaller than:

– The size of the input point set
– The size of the output surface

using a streaming and parallel, linear-time algorithm.

<table>
<thead>
<tr>
<th>Points (1x10^9)</th>
<th>24 GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangles (1x10^9)</td>
<td>18 GB</td>
</tr>
<tr>
<td>Total Memory</td>
<td>242 GB</td>
</tr>
<tr>
<td>Working Memory</td>
<td>2 GB</td>
</tr>
<tr>
<td>Time (12 cores)</td>
<td>886 min</td>
</tr>
</tbody>
</table>
We can reconstruct watertight surfaces in a memory footprint smaller than:

- The total memory used
- The size of the input point set
- The size of the output surface

Using a streaming and parallel, linear-time algorithm.
Midterm

Content:
Everything that we have covered since Spring break:
  – Subdivision Surfaces
  – Spline Curves/Surfaces
  – Procedural Models
  – Solid Models
  – 3D Scanning
  – Surface Reconstruction
  – Animation
  – Radiosity
  – Image Stitching
  – Shape Matching
Midterm

Format:
• Short answer questions only
• No essays
• No True/False
• No multiple choice