From Aligned Range Maps to Triangle Meshes
Input & Output

- Input: a set of aligned range maps
  - Variable density
  - Noisy
- Output: a triangulated mesh
  - Manifold
  - Good aspect ratio
  - Possibly watertight (may not be possible)
Mesh Zippering [Turk94]

- **Input**: triangulated ranges maps (not just point clouds)
- **Works in pairs**:
  - Remove overlapping portions
  - Clip one RM against the other
  - Remove small triangles
Mesh Zippering

- Input: triangulated ranges maps (not just point clouds)
- Works in pairs:
  - Remove overlapping portions
  - Clip one RM against the other
  - Remove small triangles
Mesh Zippering

- Input: triangulated ranges maps (not just point clouds)
- Works in pairs:
  - Remove overlapping portions
  - **Clip one RM against the other**
  - Remove small triangles
Mesh Zippering

- Input: triangulated range maps (not just point clouds)
- Works in pairs:
  - Remove overlapping portions
  - Clip one RM against the other
  - Remove small triangles
Mesh Zippering

- Input: triangulated ranges maps (not just point clouds)

- Works in pairs:
  - Remove overlapping portions
  - Clip one RM against the other
  - Remove small triangles
Mesh Zippering

- Input: triangulated ranges maps (not just point clouds)
- Works in pairs:
  - Remove overlapping portions
  - Clip one RM against the other
  - Remove small triangles

![Diagram of Final triangles and small triangles removed]
Mesh Zippering

- Not so trivial to implement…for example..
  - **remove overlapping regions**: «a face of mesh A overlaps if its 3 vertices project on mesh B»
  - Hole may appear, to be fixed later…
Mesh Zippering

- Not so trivial to implement...for example..
  - remove overlapping regions: criterion?
Mesh Zippering [marras10]

- Not so trivial to implement... for example...
  - remove overlapping regions: criterion?
    - Preserve faces from left
    - Preserve faces from right
    - Halfway (distance from the border)
Ball Pivoting [Bernardini99]

- **Input:** a uniform point cloud and normals per-vertex
- **Output:** a triangulation of the point cloud
- **How?** With a rolling sphere:

(a)  
(b)  
(c)
Ball Pivoting: Alpha Shapes [Edelsbrunner83]

Question: what is the surface of a point cloud?
Possible answer: the Convex Hull

**Def:** Convex hull of \( \{p_0,p_1,\ldots,p_h\} \)

\[
CH \in \mathbb{R}^n \\
CH = HS_i, \forall p_j \in CH \\
\forall CH = HS_i, \forall p_j \in CH \Rightarrow CH \subseteq CH
\]

**Def:** Empty half-space EHS: Half space not including points in S

**Def:** Convex hull of S

\[
CH = \mathbb{R}^n \setminus \bigcup_i EHS
\]
**Alpha Shapes**

*Def: empty $\alpha$-ball*  
Sphere with radius $\alpha$ **not** including point of S

*Def: $\alpha$-Hull*  
Complemento dell’unione di tutte le empty $\alpha$-ball

An **alpha-shape** is a polytope, not necessarily convex or connected

\[ \alpha H = R^n \setminus \bigcup_i E\alpha-balls \]
Alpha Shapes Examples
Alpha Shapes: generalizations

- **Scale density alpha-shape**
  \( \alpha \) is a function of point density

- **Anisotropic alpha-shape**
  ellipsoid instead of spheres i.e. the metrics is defined with an ellipsoid

alpha-shape  
scaled  
Scaled & anisotropic
Normals on a point cloud

Question: what are the normals on a point cloud?
Answer: the gradient of the surface, which we don’t know yet 😞

- Use the assumption that the points are **locally** on a plane
  - Take the K neighbors (easy with range maps)
  - Compute the best fitting plane
Neighborhood used for normal computation

Accurately computed normals in locally planar regions
Neighborhood used for normal computation

Two possible directions [towards outside]
Neighborhood used for normal computation

Two possible directions [towards inside]
Neighborhood used for normal computation

Consistent normal direction for all points
Neighborhood used for normal computation

Consistent normal direction for all points

Choose outside orientation
Neighborhood used for normal computation

Inaccurately computed normals close to boundaries
Ball Pivoting Algorithm [Input]

• Points/normals of all range images in data structure
• Connectivity of points in range images not needed anymore
• Radius $\rho$ selected by the user.
Voxel-based data structure

Fast search for points within circle of radius $\rho$
Voxel-based data structure

Fast search for points within circle of radius $\rho$

Need to search only in adjacent voxels

[9 in this picture – 27 in 3D space]
Voxel-based data structure

Fast search for points within circle of radius \( \rho \)
Need to search only in adjacent voxels
[9 in this picture – 27 in 3D space]
A sequence of ball-pivoting operations. From left to right: A seed triangle is found; pivoting around an edge of the current front adds a new triangles to the mesh; after a number of pivoting operations, the active front closes on itself; a final ball-pivoting completes the mesh.

Closely related to alpha-shapes, Edelsbrunner 94
Seed triangles

A number of seed triangles with their associated spheres shown
Pivoting in 2D

(a) Circle of radius $\rho$ pivots from point to point, connecting them with edges.

(b) When sampling density is low, some of the edges will not be created, leaving holes.

(c) When the curvature of the manifold is larger than $1/\rho$, some of the points will not be reached by the pivoting ball, and features will be missed.
The algorithm [Edge representation]

- Edge \((s_i, s_j)\)
  - Opposite point \(s_0\), center of empty ball \(c\)
  - Edge: “Active”, “Boundary”, or “Frozen”
Pivoting example

Initial seed triangle:
Empty ball of radius $\rho$ passes through the three points

Active edge

Point on front
Pivoting example

Ball pivoting around active edge

Active edge

Point on front
Pivoting example

Ball pivoting around active edge

Active edge
- Point on front
Pivoting example

Ball pivoting around active edge

Active edge

Point on front
Pivoting example

Ball pivoting around active edge

Active edge

Point on front
Pivoting example

Ball pivoting around active edge

Active edge
- Point on front
- Internal point
Pivoting example

Boundary edge

Ball pivoting around active edge
No pivot found

Active edge
- Point on front
- Internal point
Pivoting example

Boundary edge

Ball pivoting around active edge

Active edge
- Point on front
- Internal point
Pivoting example

Boundary edge

Ball pivoting around active edge
No pivot found

Active edge
- Point on front
- Internal point
Pivoting example

Boundary edge

Ball pivoting around active edge

Active edge
- Point on front
- Internal point
Pivoting example

Boundary edge

Frozen edge

Points in frozen region

Ball pivoting around active edge

Active edge
- Point on front
- Internal point
Pivoting example

Boundary edge

Frozen edge

Ball pivoting around active edge

Points in frozen region

Active edge
  • Point on front
  • Internal point
Pivoting example

Boundary edge

Frozen edge

Points in frozen region

Ball pivoting around active edge

Active edge
- Point on front
- Internal point
Pivoting example

Boundary edge

Frozen edge

Points in frozen region

Ball pivoting around active edge

Active edge
- Point on front
- Internal point
(a) Points “below” surface level are not touched by the pivoting ball and remain isolated (and are discarded by the algorithm).

(b) Due to missing data, the ball pivots around an edge until it touches a sample that belongs to a different part of the surface. By checking that triangle and data point normals are consistently oriented, we avoid generating a triangle in this case.

(c) Noisy samples form two layers, distant enough to allow the ball to “walk” on both layers. A spurious small component is created.
Ball Pivoting Algorithm

Overlapping scans

3D mesh
Ball Pivoting Algorithm

3D mesh detail

3D mesh detail
Out of core implementation

• Process data in slices that fit in memory
• No limit in size of input
• BPA’s active front provides natural implementation
• Of major importance for large scale scenes

run-out-of-core()

for (i=0; i<k; ++i){
    current_slice=i;
    if (i>0) unload_slice(i-1);
    load_slice(i);
    if (i>0) move_frozen_to_active();
    run_bpa();
    save_current_mesh();
}

end run-out-of-core()
Out of core implementation

Frozen region

Mesh for first slice

First slice in memory  Bounding box of scene
Out of core implementation

Frozen region

Second slice in memory
Out of core implementation

Third slice in memory, etc.
Out of core results

Part of the Great Hall mesh. The different colors correspond to meshes of different slices produced by the out-of-core implementation.
Limits of direct tessellation

Sampling quality and reconstruction issues:

- Ideal sampling
- Uneven sampling: holes?
- Noisy sampling: interpolation?
- Solid object with thin section?
- Solid object with small features?
Volumetric methods

- Use the RM (or the point clouds) to define a distance field from the surface

- return the isosurface for a small distance
Marching Cube: isosurfaces from volume data [Lorensen87]:

**Input:**
- a regular 3d grid where each node is associated with a scalar value $f$ (i.e. a scalar field)
- a scalar value $\alpha$

**Output:** a surface with scalar value $\alpha$ and non null gradient (the isosurface)

The value at $p$ is obtained by trilinear interpolation of the values of the vertices of the grid cell contained in

$$f(x, y) = \sum_{i,j=0}^{i,j=1} f_{ij} \cdot (1 - x)^{1-i} x^i (1 - y)^{1-j} y^j$$

$$\alpha = 7.5$$

$$f = 7.5$$
Marching Cube: configurations

- All configurations: $2^8 = 256$, but only 14 considering rotations, mirroring and complement
For each combination of field value respect to the threshold, store the corresponding triangulation.
Marching Cubes: Implementation

• “Marching”. The algorithm proceeds cell by cell, row by row, slice by slice. Each cell produces a triangulation

• Except than for the border cells, a new cell only requires the evaluation of three edges

• Store “border” vertices to avoid duplication
  - Topology problems: result made by a lot of tiny meshes
  - Care must be taken in unifying vertices: numerical problems cause degenerate configurations
Marching Cubes: pros/issues

• Pros:
  – Quite easy to implement
  – Fast and non memory consuming
  – Very robust
• ..then why from 87 zillions papers where published?

Issues:
• **Consistency**: Guarantee a C0 and manifold result: ambiguous cases
• **Correctness**: return a good approximation of the “real” surface
• **Mesh complexity**: the number of triangles does not depend on the shape of the isosurface
• **Mesh quality**: arbitrarily ugly triangles
Marching Cubes: ambiguous cases
Marching Cubes: ambiguous cases

?
Saddle points

Field value on a cell’s face

\[ f(0, x, y) = cyz + dy + gz + h \]

\[
\frac{\partial f(0, x', y')}{\partial y} = cz' + d = 0 \Rightarrow z' = -\frac{d}{c}
\]

\[
\frac{\partial f(0, x', y')}{\partial x} = cy' + g = 0 \Rightarrow y' = -\frac{g}{c}
\]
ELUT: Exhaustive LUT [Cignoni00]

For each ambiguous configuration determines the coherent internal triangulation looking at the saddle points
Marching Tetrahedra

• Tetrahedral cells (instead of cubical)
• Only 3 configurations (from the $2^4$ permutation of grid values)
• No ambiguities but it may be “less” correct
Marching Tetrahedra

• Original approach []: cubic cells are partitioned in 5 (or 6) tetrahedra.
  – Subdivision determines topology

• Body centered cubic lattice: one more sample in the cubic cell
  – Unique subdivision
  – Equal tetrahedra
  – Better surface (better triangles)
Adaptive triangulation []

- Refine for better approximation (re-evaluate scalar field)
Extended MC \cite{Kobbelt01}

\begin{align*}
D2Y &> 0 \\
D1Y &< 0 \\
D1X &< 0 \\
D3X &> 0 \\
\end{align*}

Surface

Exact intersection point
MC

Reconstructed surface

Normal

Tangent element
Extended MC

Marching Cubes

Extended Marching Cubes
Dual Marching Cubes [Schaefer04]

• one vertex for each patch generated by MC
• One quad for each intersected edge (the 4 vertices associated to the patches of the cells sharing the edge)
• Tends to improve triangles quality
From point cloud to a scalar field...

Problem: given a set of points \( \{x_0, ..., x_n\} \), define

\[
f(x) = \varphi(\{x_0, ..., x_n\})
\]

\[
S = \{x \mid f(x) = \alpha\}
\]

so that \( S \) interpolates/approximates the point cloud
Metaballs [Blinn92, Wyvill86]

- $f$ is the sum of functions that have maximum in the points and decay with the distance

$$f(x) = \sum_i \frac{1}{\|x - x_i\|^2}$$

$$f(x) = \sum_i \left(2 \frac{r^3}{R^3} - 3 \frac{r^2}{R^2} + 1\right), r = \|x - x_i\|, \text{ } R= \text{support radius}$$
Radial Basis Functions (RBF)

Solutions that follow the general scheme:

\[
f(x) = p(x) + \sum_i \omega_i \varphi(\|x - x_i\|)
\]

\[
f(x_i) = f_i
\]

weights: \(\omega_i \in \mathbb{R}\)

RBF: \(\varphi: \mathbb{R} \to \mathbb{R}\)

\(p\) a polynome
Radial Basis Functions (RBF) [Carr01]

\[ f(x) = p(x) + \sum_i \omega_i \varphi(||x - x_i||), \]

\[ \omega_i \in \mathbb{R} \]
\[ \varphi: \mathbb{R} \to \mathbb{R} \]
\[ p \text{ a polynome} \]

\[
\begin{bmatrix}
A \\
p^T \\
0
\end{bmatrix}
\begin{bmatrix}
\omega \\
c
\end{bmatrix} =
\begin{bmatrix}
F \\
0
\end{bmatrix}
\]

\[ A_{ij} = \varphi(||x_j - x_i||) \]
\[ p: \text{basis for all polynomials of degree } k \]
\[ P_{ij} = p_j(x_i) \]

Examples of polynomial basis:
\[ p = \{1, x, y, z\} \quad d=1, \ m=3 \]
\[ p = \{1, x, y, x^2, xy, y^2\} \quad d=2, \ m=2 \]
\[ p = \{1, x, x^2, x^3\} \quad d=1, \ m=3 \]
Example

\[ P = \{1, x\} \]

\[ \varphi(d) = d \]

\[
\begin{bmatrix}
\varphi(x_1, x_1) & \varphi(x_1, x_2) & \varphi(x_1, x_3) & p_1(x_1) & p_2(x_1) \\
\varphi(x_2, x_1) & \varphi(x_2, x_2) & \varphi(x_2, x_3) & p_1(x_2) & p_2(x_2) \\
\varphi(x_3, x_1) & \varphi(x_3, x_2) & \varphi(x_3, x_3) & p_1(x_3) & p_2(x_3) \\
p_1(x_1) & p_1(x_2) & p_1(x_3) & 0 & 0 \\
p_2(x_1) & p_2(x_2) & p_2(x_3) & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\omega_1 \\
\omega_2 \\
\omega_3 \\
c_1 \\
c_2
\end{bmatrix}
= 
\begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
0 \\
0
\end{bmatrix}
\]

\[
\omega_1 = -2 \quad f_1 = -2 \\
\omega_2 = -1 \quad f_2 = -1 \\
\omega_3 = 1 \quad f_3 = 1
\]

Polynomial basis: \{1, x\}
Example

\[
\begin{bmatrix}
0 & 1 & 4 & 1 & -2 \\
1 & 0 & 3 & 1 & -1 \\
4 & 3 & 0 & 1 & 2 \\
1 & 1 & 1 & 0 & 0 \\
-2 & -1 & 2 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
\omega_1 \\
\omega_2 \\
\omega_3 \\
\omega_1 \\
\omega_1 \\
\end{bmatrix}
= 
\begin{bmatrix}
-2 \\
-1 \\
1 \\
0 \\
0 \\
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
\omega_1 \\
\omega_2 \\
\omega_3 \\
\omega_1 \\
\omega_1 \\
\end{bmatrix}
= 
\begin{bmatrix}
0.166 \\
-0.166 \\
0 \\
-0.5 \\
0.66 \\
\end{bmatrix}
\]

\[
f(x) = -0.5 + 0.66x + 0.166(|x - (-2)|) - 0.166(|x - (-1)|) + 0(x - 2) = \\
= -0.334 + 0.66x + 0.166(|x + 2|) - 0.166(|x + 1|)
\]
Example

\[ f(x) = -0.334 + 0.66x + 0.166(|x + 2|) - 0.166(|x + 1|) \]
Radial Basis Functions (RBF)

• Several possible choices for \( \varphi \) and \( p \):
  – \( \varphi(d) = d \), linear polynomial
  – \( \varphi(d) = d^2 \), linear polynomial
  – \( \varphi(d) = d^3 \), linear/quadratic polynomial
  – \( \varphi(d) = d^2 \log(d) \), linear/quadratic polynomial
  – ….  

• Issue 1: if functions have unbounded support, i.e. nonzero everywhere, the matrix will always be dense
  – Expensive to solve when \( n \) increase…

• Issue 2: the whole surface is influenced by each single input point
Bounded RBD [Morse01]

\[
\varphi(d) = \begin{cases} 
(1 - d)^p P(d), & d < 1 \\
0, & d \geq 1 
\end{cases}
\]

\[P(d) = \text{polynome with degree 6}\]

- The value of \(f\) is determined only locally (within the radius 1)
  - Use \(\varphi(d/R)\) to adapt to the point cloud resolution
- The resulting matrix is \textbf{sparse}
- The fitting is local
Bounded RBD

\[ \varphi(d) = f(x) = \begin{cases} 
(1 - d)^p P(d), & d < 1 \\
0, & d \geq 1 
\end{cases} \]

\( P(d) = \text{polynome with degree 6} \)

8000-point model  Interpolated to 41,864 points
Bounded RBF

\[ \varphi(d) = \begin{cases} (1 - d)^p P(d), & d < 1 \\ 0, & d \geq 1 \end{cases} \]

\[ P(d) = \text{polynome with degree 6} \]

- The value of \( f \) is determined only locally (within the radius 1)
  - Use \( \varphi(d/R) \) to adapt to the point cloud resolution
- The resulting matrix is **sparse**
- The *fitting* is local

More issues:
- Still hard to represent sharp features, anisotropic basis functions may be used [Dinh01]
Partition of Unit [Ohtake03]

\[ f(x) = \sum_i \omega_i(x) Q_i(x) \]

- \( f \) is defined globally as the weighted sum of local functions that describe (implicitly) the surface.
- Each \( i \) corresponds to a region of \( \mathbb{R}^3 \) where the function is described by \( f_i \).
- **The sum of the weights is 1 everywhere:**

\[ \sum_i \omega_i(x) = 1 \]

- Which is obtained by normalization

\[ \omega_i(x) = \frac{W_i(x)}{\sum_i W_i(x)} \]
Multilevel PoU

- Subdivide the domain with an octree
- Fit the points within each cell with a function $Q_i(x)$, either:
  - A quadric (for noisy and unbounded regions)
  - A bivariate (u,v) quadratic polynomial in a local coordinate system (for smooth patch)
  - A piecewise quadratic surface (for sharp features)
- Blending PU:

$$\omega_i(x) = b \left( \frac{3|x - c_i|}{2R_i} \right)$$

$$R_i = 0.75 \times diag$$
Results

Distance field from range maps [Levoy]  MPU implicits
Moving Least Square Reconstruction

**LS**
Least square

\[ \min_{f \in \prod_{m}^{d}} \sum_{i} \| f(x_i) - f_i \| \quad \prod_{m}^{d} : \text{polynomials degree m in d-dimension} \]

**WLS**
Weighted Least square

\[ \min_{f, \bar{x} \in \prod_{m}^{d}} \sum_{i} \theta(\|x_i - \bar{x}\|) \| f(x_i) - f_i \| \quad \bar{x}: \text{fixed point} \]

**MLS**
Moving Least square

\[ \min_{f, x \in \prod_{m}^{d}} \sum_{i} \theta(\|x_i - x\|) \| f(x_i) - f_i \| \]
Moving Least Square Reconstruction

[Alexa01]

- Iterative approach: project the points near the surface onto the surface (??)

1. \[ \min_{n,D} \sum_{i=1}^{n} (\langle n, p_i \rangle - D)^2 \theta(\|p_i - q\|) \]

2. \[ \min_g \sum_{i=1}^{n} (g(x_i, y_i) - f)^2 \theta(\|p_i - q\|) \]

3. Move \( r \) to \( q + g(0,0) n \)
Moving Least Square Reconstruction

[Alexa01]

- Iterative approach: project the points near the surface onto the surface (??)

Squared distance between \( p_i \) and the plane \( n, D \)

1. \( \min_{n,D} \sum_{i=1}^{n} ((n, p_i) - D)^2 \theta(\|p_i - q\|) \)  No linear problem

2. \( \min_{g} \sum_{i=1}^{n} (g(x_i, y_i) - f)^2 \theta(\|p_i - q\|) \)

3. Move \( r \) to \( q + g(0,0) n \)
Moving Least Square Reconstruction
[Alexa01]

- Iterative approach: project the points near the surface onto the surface (??)

1. \( \min_{n,D} \sum_{i=1}^{n} ((n, p_i) - D)^2 \theta(||p_i - q||) \)

   \( f_i = n \cdot (p_i - q) \)

2. \( \min_g \sum_{i=1}^{n} (g(x_i, y_i) - f_i)^2 \theta(||p_i - q||) \) \hspace{1cm} \text{Known from 1.}

   \( g: \mathbb{R}^2 \rightarrow \mathbb{R} \) approximates point set in the local reference system centered in q

3. Move r to \( q + g(0,0) n \)
Moving Least Square Reconstruction

\[ \theta(d) = e^{-\frac{d^2}{h^2}} \]  

\( h \) is related to the spacing between samples
Poisson Reconstruction [kazhdan06]

- reconstruct the surface of the model by solving for the indicator function of the shape.

  - ...the indicator function is unknown!

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

Indicator function $\chi_M$
Challenge

- How to construct the indicator function?

Oriented points $\rightarrow$ Indicator function $\chi_M$
Gradient Relationship

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

Oriented points

Indicator gradient \( \nabla \chi_M \)
Integration as a Poisson Problem

- Represent the points by a vector field $\vec{V}$
- Find the function $\chi$ whose gradient best approximates $\vec{V}$:
  \[
  \min_{\chi} \left\| \nabla \chi - \vec{V} \right\|
  \]
- Applying the divergence operator it becomes a Poisson problem:
  \[
  \nabla \cdot (\nabla \chi) = \nabla \cdot \vec{V} \quad \Leftrightarrow \quad \Delta \chi = \nabla \cdot \vec{V}
  \]
Implementation

Given the Points:
- Set \texttt{octree}
- Compute vector field
- Compute indicator function
- Extract iso-surface
Vector field

- Use a basis $F: \mathbb{R}^3 \rightarrow \mathbb{R}$ (another RBF)
  - One base for each node containing some original point
  - Express the vector field as the interpolation of neighbor cells

$$
\vec{V}(\vec{q}) = \sum_{s \in \mathcal{N}^s} \alpha_s F(\vec{q} - \vec{s})
$$

For each octree level
For each neighbor
Compute indicator function

- $\chi$ is also expressed on the same basis $F$

$$\chi = \sum_o x_o F_o, \quad x \in \mathcal{R}^{\mid o\mid}$$

$$\min_{x \in \mathcal{R}^{\mid o\mid}} \| Lx - v \|^2$$

Divergence of $V$ at node $o$

$L_o = \frac{\partial F}{\partial x} F + \frac{\partial F}{\partial y} \frac{\partial F}{\partial x}$

So that $Lx$ is the laplacian of $\chi$
References


[Marras10] Controlled and adaptive mesh zippering S.Marras, F. Ganovelli, P. Cignoni, R. Scateni and R. Scopigno, GRAPP 2010


[Cignoni00] Reconstruction of topologically correct and adaptive trilinear isosurfaces P Cignoni, F Ganovelli, C Montani, R Scopigno Computers and graphics 24 (3), 399-418

[Kobbelt01] Feature sensitive surface extraction from volume dataL.P. Kobbelt, M. Botsch, U. Schwanecke, Hans-Peter Seidel SIGGRAPH '01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques Pages 57-66


Credits:

the slides for the Ball Pivoting Algorithm are taken as they were from the original presentation. Part of the slides on Poisson reconstruction were also taken from the original presentation.

All images are taken from the papers.