Anisotropic Surface Remeshing

Scaling up with Voronoi Parallel Linear Enumeration

Bruno Lévy
ALICE Géométrie & Lumière
CENTRE INRIA Nancy Grand-Est
OVERVIEW

Part. 1. Goals and Motivations
Part. 2. Centroidal Voronoi Tesselations
Part. 3. Tweaking the Definition of Distance
Part. 4. Constraints & Protecting Balls
Goals and Motivations
Part. 1. Goals: a “Flexible” mesh generator

“wish list”

• Tolerant meshing (Scan2FEA, STL2FEA)

• Steerable (orientation, anisotropy, quads/hex)

• Beauty (… of the mesh, … of the approach)
Part. 1. Goals: a “Flexible” mesh generator

Tolerant meshing (Scan2FEA, STL2FEA)
Part. 1. Goals: a “Flexible” mesh generator
Part. 1. Goals: a “Flexible” mesh generator

Steerable meshing (orientation, anisotropy, quads/hex)
Part. 1. Goals: a “Flexible” mesh generator

Steerable meshing (orientation, anisotropy, quads/hex)

Hex-dominant meshing of the larynx
(data courtesy Dan Einstein)
**Part. 1. Goals:** a “Flexible” mesh generator

Goal: beauty (… of the mesh, … of the approach)

Input: raw scanned mesh (courtesy XYZRGB)
Part. 1. Goals: a “Flexible” mesh generator

Goal: beauty (… of the mesh, … of the approach)

Hope for faster and more accurate computations
Part. 1. Goals: a “Flexible” mesh generator

Goal: beauty
... of the mesh,
... of the approach
Part. 1. Goals: a “Flexible” mesh generator

Goal: beauty
... of the mesh,
... of the approach
Part. 1. Goals: a "Flexible" mesh generator

Part. 1. Goals and Motivations
Part. 2. Centroidal Voronoi Tesselations
Part. 3. Tweaking the Definition of Distance
Part. 4. Constraints & Protecting Balls
Part. 1. Goals: a “Flexible” mesh generator

Part. 1. Goals and Motivations
Part. 2. Centroidal Voronoi Tessellations
Part. 3. Tweaking the Definition of Distance
Part. 4. Constraints & Protecting Balls
Part. 1. Goals: a “Flexible” mesh generator

Part. 1. Goals and Motivations
Part. 2. Centroidal Voronoi Tesselations
Part. 3. Tweaking the Definition of Distance
Part. 4. Tweaking the Definition of Distance
Centroidal Voronoi Tessellations
Part. 2. Centroidal Voronoi Tessellation

Optimize a Voronoi diagram from the point of view of sampling regularity (quantization noise power)

[Lloyd] least squares quantization in PCM ; [Du], [Iri], [Okabe]
Part. 2. Centroidal Voronoi Tessellation

Optimize a Voronoi diagram from the point of view of sampling regularity (quantization noise power)

\[ F = \sum_{i} \int_{\text{Vor}(i)} \left\| x_i - x \right\|^2 \, dx \]

[Lloyd] least squares quantization in PCM
Part. 2. Centroidal Voronoi Tessellation

Optimize a Voronoi diagram from the point of view of sampling regularity (quantization noise power)

\[ F = \sum \int_{\text{Vor}(i)} \left\| x_i - x \right\|^2 \, dx \]

[Lloyd] least squares quantization in PCM
Part. 2. Centroidal Voronoi Tessellation

Optimize a Voronoi diagram from the point of view of sampling regularity (quantization noise power)

\[ F = \sum_i \int_{\text{Vor}(i)} \| x_i - x \|^2 \, dx \]

[Lloyd] least squares quantization in PCM
Part. 2.
Part. 2. Centroidal Voronoi Tesselation

\[ F = \sum_i \int_{\text{Vor}(i)} \left\| x_i - x \right\|^2 dx \]

\( F \): Quantization noise power (measures the quality of the sampling)

**Theorem:** \( F \) is \( C^2 \) almost everywhere

[Liu, Wang, L, Sun, Yan, Lu and Yang 09]
3
Tweaking the definition of distances
Part. 3. Tweaking distances - Anisotropy

The input: anisotropy field

\[ G(x,y) = \begin{bmatrix} a(x,y) & b(x,y) \\ b(x,y) & c(x,y) \end{bmatrix} \]
Part. 3. Tweaking distances - Anisotropy

*The input:* anisotropy field

\[ G(x,y) = \begin{bmatrix} a(x,y) & b(x,y) \\ b(x,y) & c(x,y) \end{bmatrix} \]

\[ \{ q \mid d_G(p,q) = 1 \} \]
Part. 3. Tweaking distances - Anisotropy

The input: anisotropy field

\[ G(x,y) = \begin{bmatrix} a(x,y) & b(x,y) \\ b(x,y) & c(x,y) \end{bmatrix} \]

\[ <v,w>_G = v^t G(p) w \]

\[ l_G(C) = \int_{t=0}^{1} \sqrt{v(t)^t G(t) v(t)} \, dt \]
Part. 3. Tweaking distances - Anisotropy

**The result:** triangles are “deformed” by the anisotropy.
Part. 3. Tweaking distances - Anisotropy

The result: triangles are “deformed” by the anisotropy.

Q: How to compute an Anisotropic Centroidal Voronoi Tessellation?
Part. 3 **Tweaking distances – Anisotropy**

Tweak 1/3

Standard CVT: \[ F = \sum_i \int_{\text{Vor}(i)} \left\| (x_i - x) \right\|^2 dx \]

Anisotropic CVT: \[ F = \sum_i \int_{\text{Vor}(i)} \left\| (x_i - x) \right\|^2 dx \]

[Qiang Du]

[L and Bonneel 2012]
Part. 3 Tweaking distances - Anisotropy

The key idea

This example:

Anisotropic mesh in 2d  ↔  Isotropic mesh in 3d
This example:

Anisotropic mesh in 2d  ↔  Isotropic mesh in 3d

Replace anisotropy with additional dimensions
Part. 3 Tweaking distances - Anisotropy

The key idea

Replace **anisotropy** with **additional dimensions**

*Note: more dimensions may be needed*
Part. 3 Tweaking distances - Anisotropy

The key idea

Replace **anisotropy** with **additional dimensions**

*Note: more dimensions may be needed*

*How many?*
Part. 3 Tweaking distances - Anisotropy

The key idea

Replace anisotropy with additional dimensions

Note: more dimensions may be needed

How many?
John Nash’s isometric embedding theorem:

Maximum: depending on desired smoothness

$C^1 : 2n$ [Nash-Kuiper]

$C^k : \text{bounded by } n(3n+11)/2$ [Nash, Nash-Moser]
Part. 3 Tweaking distances - Anisotropy

A 6d embedding for curvature-adapted meshing
Part. 3  Anisotropy - the algorithm

Lloyd relaxation in $\mathbb{R}^6$ (Naïve version)

(1) Embed the surface $S$ into $\mathbb{R}^6$
Part. 3  Anisotropy - the algorithm

Lloyd relaxation in IR$^6$ (Naïve version)

(1) Embed the surface $S$ into IR$^6$
(2) Compute initial point distrib. $X$
Part. 3  Anisotropy - the algorithm

Lloyd relaxation in $\mathbb{R}^6$ (Naïve version)

(1) Embed the surface $S$ into $\mathbb{R}^6$
(2) Compute initial point distrib. $X$
While convergence is not reached
Part. 3 Anisotropy - the algorithm

Lloyd relaxation in IR\(^6\) (Naïve version)

1. Embed the surface \(S\) into IR\(^6\)
2. Compute initial point distrib. \(X\)
While convergence is not reached
3. Compute Vor(\(X\))
Lloyd relaxation in $\mathbb{R}^6$ (Naïve version)

1. Embed the surface $S$ into $\mathbb{R}^6$
2. Compute initial point distrib. $X$
3. While convergence is not reached
   3. Compute $\text{Vor}(X)$
   4. Compute $\text{Vor}(X) \cap S$
Part. 3 Anisotropy - the algorithm

Lloyd relaxation in $\mathbb{R}^6$ (Naïve version)

(1) Embed the surface $S$ into $\mathbb{R}^6$
(2) Compute initial point distrib. $X$
While convergence is not reached
    (3) Compute $\text{Vor}(X)$
    (4) Compute $\text{Vor}(X) \cap S$
    (5) Move each $x_i$ to the centroid of $\text{Vor}(x_i) \cap S$
Part. 3  Anisotropy - the algorithm

Lloyd relaxation in $\mathbb{R}^6$ (Naïve version)

1. Embed the surface $S$ into $\mathbb{R}^6$
2. Compute initial point distrib. $X$
While convergence is not reached
   3. Compute $\text{Vor}(X)$
   4. Compute $\text{Vor}(X) \cap S$
   5. Move each $x_i$ to the centroid of $\text{Vor}(x_i) \cap S$

Costs $d!$ for dimension $d$
Part. 3  Anisotropy - the algorithm

Lloyd relaxation in $\mathbb{IR}^6$ (Naïve version)

(1) Embed the surface $S$ into $\mathbb{IR}^6$
(2) Compute initial point distrib. $X$
While convergence is not reached
(3) Compute $\text{Vor}(X)$
(4) Compute $\text{Vor}(X) \cap S$
(5) Move each $x_i$ to the centroid of $\text{Vor}(x_i) \cap S$

Costs $d!$ for dimension $d$
$d = 6 ; d! = 720$
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
Part. 3 Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
Neighbors in increasing (6d) distance from $x_i$
Part. 3 Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
Bisector of \( x_i, x_1 \)
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping

Half-space clipping

This side: $\Pi^-(i, 1)$

The other side: $\Pi^+(i, 1)$
Part. 3 Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
Half-space clipping

This side: $\Pi^-(i, 1)$

The other side: $\Pi^+(i, 1)$

Remove $\Pi^-(i, 1)$
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
Half-space clipping

Then remove $\Pi^{-}(i,2)$
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
Half-space clipping

... then remove $\Pi^-(i,3)$
Part. 3 Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
Half-space clipping

... then remove $\Pi^{-}(i,4)$
Part. 3 Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
Half-space clipping

... then remove $\Pi^-(i,5)$
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
When should I stop?
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping

When should I stop?  \( R_k \)
Voronoi cells as iterative convex clipping
When should I stop?
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping

When should I stop? \[ d(x_i, x_k) > 2 R_k \]
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping

Theorem:

\[ d(x_i, x_{k+1}) > 2R_k \rightarrow \bigcap \Pi^+(i,k) = \text{Vor}(x_i) \]

[Li and Bonneel 2012]
Part. 3 Anisotropy - the algorithm

Voronoi cells as iterative convex clipping

When should I stop? \[ d(x_i, x_k) > 2 R_k \]

“Radius of security” is reached

[L and Bonneel 2012]
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
When should I stop?  \( d(x_i, x_k) > 2 \, R_k \)

“Radius of security” is reached

Note: \( R_k \) decreases and \( d(x_i, x_k) \) increases

[L and Bonneel 2012]
Part. 3 Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
When should I stop? \( d(x_i, x_k) > 2R_k \)

“Radius of security” is reached
Note: \( R_k \) decreases and \( d(x_i, x_k) \) increases

Advantages:

[Laurent and Bonneel 2012]
Voronoi cells as iterative convex clipping
When should I stop? \( d(x_i, x_k) > 2R_k \)

“Radius of security” is reached
Note: \( R_k \) decreases and \( d(x_i, x_k) \) increases

Advantages:
(1) Compute \( \text{Vor}(X) \cap S \) directly (start with \( f \) and clip)

[\text{L and Bonneel 2012}]
Part. 3  Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
When should I stop?  \( d(x_i, x_k) > 2 R_k \)

“Radius of security” is reached

Note: \( R_k \) decreases and \( d(x_i, x_k) \) increases

Advantages:
(1) Compute \( \text{Vor}(X) \cap S \) directly (start with \( f \) and clip)
(2) Replace Delaunay with ANN! (no \( d! \) factor)

[L and Bonneel 2012]
Part. 3   Anisotropy - the algorithm

Voronoi cells as iterative convex clipping
When should I stop?  \( d(x_i, x_k) > 2R_k \)

“Radius of security” is reached
Note:  \( R_k \) decreases and \( d(x_i, x_k) \) increases

Advantages:
(1) Compute \( \text{Vor}(X) \cap S \) directly (start with \( f \) and clip)
(2) Replace Delaunay with ANN! (no \( d! \) factor)
(3) Parallelization is trivial (partition \( S \) and // in parts)

[L and Bonneel 2012]
Part. 3 Tweaking distances - Anisotropy

A 6d embedding for curvature-adapted meshing

David Lopez
Part. 3  Tweaking distances – Lp norm
Part. 3  Tweaking distances – Lp norm

\[ L_p \text{ norm: } \| x \|_p = \sqrt[p]{|x|^p + |y|^p + |z|^p} \]
Part. 3
Part. 3 Tweaking distances – Lp norm

Tweak 2/3

Standard CVT: \( F = \sum_{i} \int_{\text{Vor}(i)} \left\| (x_i - x) \right\|^2 \, dx \)

Lp CVT: \( F = \sum_{i} \int_{\text{Vor}(i)} \left\| M(x) (x_i - x) \right\|^p \, dx \)

[Li and Liu 2010]

Anisotropy and desired orientation

Lp norm: \( \left\| x \right\|_p = \sqrt[p]{|x|^p + |y|^p + |z|^p} \)
Part. 3  Tweaking distances – $L^p$ norm

$L_p$ CVT

81K hexes
11K tets
13K prisms
12 minutes

+ many other examples in paper and supplemental material.
Preserving sharp features with protecting balls
Part. 4 Tweaking distances – Protecting balls
Voronoï diagram: \( \text{Vor}(x_i) = \{ x \mid d^2(x,x_i) < d^2(x,x_j) \} \)
Part. 4 Tweaking distances – Protecting balls

Voronoi diagram: \( \text{Vor}(x_i) = \{ x \mid d^2(x,x_i) < d^2(x,x_j) \} \)

Power diagram: \( \text{Pow}(x_i) = \{ x \mid d^2(x,x_i) - w_i^2 < d^2(x,x_j) - w_j^2 \} \)
Part. 4 Tweaking distances – Protecting balls

Voronoi diagram: \( \text{Vor}(x_i) = \{ x \mid d^2(x,x_i) < d^2(x,x_j) \} \)

Power diagram: \( \text{Pow}(x_i) = \{ x \mid d^2(x,x_i) - w_i^2 < d^2(x,x_j) - w_j^2 \} \)

Thm: [Dey et.al] if \( B(x_i, w_i) \) and \( B(x_j, w_j) \) intersect and are empty of other \( x_k \)'s, then \([x_i,x_j]\) appears in \( \text{Pow}(X) \)

[Dey et.al, Boltcheva et.al]
Voronoi diagram: \( \text{Vor}(x_i) = \{ x \mid d^2(x,x_i) < d^2(x,x_j) \} \)

Power diagram: \( \text{Pow}(x_i) = \{ x \mid d^2(x,x_i) - w_i^2 < d^2(x,x_j) - w_j^2 \} \)

Thm: [Dey et.al] if \( B(x_i, w_i) \) and \( B(x_j, w_j) \) intersect and are empty of other \( x_k \)'s, then \( [x_i, x_j] \) appears in \( \text{Pow}(X) \)

Problem for 6d remeshing:

“Security Radius” theorem does not work with power diagrams
**Part. 4 Tweaking distances – Protecting balls**

**Voronoi diagram:** \( \text{Vor}(x_i) = \{ x \mid d^2(x, x_i) < d^2(x, x_j) \} \)

**Power diagram:** \( \text{Pow}(x_i) = \{ x \mid d^2(x, x_i) - w_i^2 < d^2(x, x_j) - w_j^2 \} \)

**Thm:** [Dey et.al] if \( B(x_i, w_i) \) and \( B(x_j, w_j) \) intersect and are empty of other \( x_k \)'s, then \( [x_i, x_j] \) appears in \( \text{Pow}(X) \)

Problem for 6d remeshing:

“Security Radius” theorem does not work with power diagrams

Idea: use 7d embedding: \( x_i \rightarrow [x_i; \sqrt{(W^2 - w_i^2)}] \) where \( W = \text{Max}(w_i) \)
Tweak 3/3

Voronoi diagram: \[ \text{Vor}(x_i) = \{ x \mid d^2(x,x_i) < d^2(x,x_j) \} \]

Power diagram: \[ \text{Pow}(x_i) = \{ x \mid d^2(x,x_i) - w_i^2 < d^2(x,x_j) - w_j^2 \} \]

Thm: [Dey et.al] if \( B(x_i, w_i) \) and \( B(x_j, w_j) \) intersect and are empty of other \( x_k \)'s, then \([x_i, x_j]\) appears in \( \text{Pow}(X) \)

Problem for 6d remeshing:

“Security Radius” theorem does not work with power diagrams

Idea: use 7d embedding: \( x_i \rightarrow [x_i; \sqrt{(W^2 - w_i^2)}] \); \( x \rightarrow [x; 0] \)

where \( W = \text{Max}(w_i) \)

\[ d(x,x_i)_{(\text{Voronoi 7d})} = d(x,x_i)_{(\text{Power 6d})} + W \]
Part. 4 Constraints – Protecting balls
Part. 4 Constraints – Protecting balls

6d power diagram (7d Voronoi diagram)
Part. 4 Constraints – Protecting balls
Part. 4 Constraints – Protecting balls
Part. 4 Constraints – Protecting balls

6d power diagram (7d Voronoi diagram)
6d power diagram (7d Voronoi diagram)
Part. 4 Constraints – Protecting balls
Part. 4 Constraints – Protecting balls

6d power diagram (7d Voronoi diagram)
Part. 4 Constraints – Protecting balls
Summary

Tweaking distances:

**Tweak #1**: anisotropic distance (through higher-d embedding)

*Readily available  
“Vorpaline” software*

**Tweak #2**: hex/quads, $L_p$ distance

*Prototype*

**Tweak #3**: constraints, power diagrams (through $d+1$ dim. embedding)

*Prototype (note: we can do simpler)*

*almost everywhere*
Software roadmap

Vorpalone 1.0: (current version)
Isotropic surface meshing
Anisotropic surface meshing
Support for mesh gradation

Vorpalone 2.0: (planned Q1 2014)
Constrained surface meshing
Structural model meshing
Quad-dominant surface meshing

Vorpalone 3.0: (planned Q4 2014)
Anisotropic volumetric meshing
Hex-dominant volumetric meshing
Acknowledgements

- **Bytes**: ANN [Mount et.al], libmesh5 [Marechal], tetgen [Si], CGAL
- **Triangles**: AIM@Shape, Digital Michelangelo, GrabCAD, XYZRGB
- **Euros**: European Research Council
  GOODSHAPE ERC-StG-205693
  VORPALINE ERC-PoC-334829
  ANR MORPHO, ANR BECASIM

Rhaleb Zayer, Wenping Wang, Jean-Francois Remacle,
Nicolas Saugnier, DongMing Yan, Loic Maréchal,
Tamal Dey, Pierre Alliez, David Bommes, Leif Kobbelt