Lp Centroidal Voronoi Tesselations

Bruno Lévy and Yang Liu
Overview

• 1. Motivations
• 2. Blowing Square Bubbles
• 3. Algorithm
• 4. Applications and Results
• 5. Conclusions and Future Work
1. Motivations – Hex meshing
1. Motivations – Why Hexes?

**Tet Meshing**
1. Fully Automated
2. Millions of elements in minutes/seconds
3. Adequate for some analysis
4. Inaccurate for other Analysis

**Hex Meshing**
1. Partially Automated, some Manual
2. Millions of elements in days/weeks/months
3. Preferred by some analysts for solution quality

[Matt Staten] (Sandia Labs)
2. Blowing Square Bubbles

$p=2$

$p=4$

$p=8$

$\ldots$
2. Blowing Square Bubbles

Disclaimer:
This presentations contains live demos.
Crashes may occur.
The presenter assumes no liability.
3. Algorithm

Standard CVT: \[ F = \sum_{i} \int_{\text{Vor}(i)} \| (x_i - x) \|^2 \, dx \]
3. Algorithm

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

Lp CVT:
\[ F = \sum \int_{\text{Vor(i)}} \left\| M(x) (x_i - x) \right\|^p \, dx \]
3. Algorithm

Lp CVT:

\[ F = \sum_i \int_{\text{Vor}(i)} \left\| \mathbf{M}(\mathbf{x})(\mathbf{x}_i - \mathbf{x}) \right\|^p \mathbf{dx} \]

Anisotropy, encodes desired orientation
Riemannian metric \( \mathbf{G} = \mathbf{M}^t \mathbf{M} \)
3. Algorithm

Lp CVT: $F = \sum_i \int_{\text{Vor}(i)} \left\| M(x)(x_i - x) \right\|^p_p \text{d}x$

Lp norm: $\| x \|^p_p = x^p + y^p + z^p$

If $p$ is even: $\| x \|^p_p = x^p + y^p + z^p$
3. Algorithm

Lp CVT: \[
F = \sum_i \int_{\text{Vor}(i)} \| \mathbf{M}(x) (x_i - x) \|_p^p \, dx
\]

Optimization with LBFGS (quasi-Newton)

For each iterate \(X^{(k)}\):
- Compute \(F(X^{(k)})\) and \(\nabla F(X^{(k)})\)
3. Algorithm

\[ F = \sum_i \int_{\text{Vor}(i)} \left\| \mathbf{M}(x) (x_i - x) \right\|_p^p \, dx \]

Computing \( F(X^{(k)}) \) and \( \nabla F(X^{(k)}) \)
3. Algorithm

\[ F = \sum_i \int_{Vor(i)} \left\| M(x) (x_i - x) \right\|_p^p \, dx \]

Computing \( F(X^{(k)}) \) and \( \nabla F(X^{(k)}) \)
3. Algorithm

\[ F = \sum \int_{\text{Vor}(i)} M(x) (x_i - x)^p \, dx \]

Computing \( F(X^{(k)}) \) and \( \nabla F(X^{(k)}) \)

\( \text{Vor}(x_i) \) clipped by Domain.
3. Algorithm

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

Computing \( F(X^{(k)}) \) and \( \nabla F(X^{(k)}) \)

An example in 3D (cross-section)

Computing the intersection between a 3D Voronoi diagram and the interior of a polygonal mesh - DEMO
3. Algorithm

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

Computing \( F(X^{(k)}) \) and \( \nabla F(X^{(k)}) \)
3. Algorithm

Computing $F(X^{(k)})$ and $\nabla F(X^{(k)})$

\[ \int_{T} \left\| M(x) (x_i - x) \right\|_p^p \, dx \]

Problem #1: $F_T(x_i, C_1, C_2)$
3. Algorithm

Computing $F(X^{(k)})$ and $\nabla F(X^{(k)})$

$$\int T \left\| M(x) (x_i - x) \right\|_p^p dx$$

Problem #1: $F_T(x_i, C_1, C_2)$

Problem #2: $C_1(x_i, x_j, x_k)$
3. Algorithm

The formula is terrible...

* When in doubt, use brute force
* I’d rather write programs that write programs than write programs
  [Tom Duff, Bell Labs]

Lp-CVT: the making of.... (not in the paper)

Programming Pearls  John Bentley
3. Algorithm

\[ \int T \left\| M(x) (x_i - x) \right\|_p^p dx \]

Lp-CVT: the making of…. (not in the paper)

template <class T> struct vec3g {
    vec3g(T x_in, T y_in, T z_in) :
        x(x_in), y(y_in), z(z_in) {
    }

    T length2() const {
        return x*x+y*y+z*z ;
    }

    T length() const {
        return sqrt(length()) ;
    }

    T x,y,z ;
} ;

typedef vec3g<double> vec3 ;
typedef vec3g<Ginac::Expression> vec3s ;
Lp-CVT: the making of….

Problem #1: \( F_T(x_i, C_1, C_2) \)

3. Algorithm

**Problem #1:**

\[
\int_T \left\| M(x) (x_i - x) \right\|_p^p \, dx
\]

**Inline vec3s tri_point():**

```cpp
inline vec3s tri_point(
    const vec3s& p1, const vec3s& p2, const vec3s& p3, const vec3s& p4,
    const Expression& u, const Expression& v, const Expression& w
) {
    Expression t = 1 - u - v - w ;
    return u*p1 + v*p2 + w*p3 + t*p4 ;
}
```

**Inline Expression tri_dist():**

```cpp
inline Expression tri_dist(
    const vec3s& p1, const vec3s& p2, const vec3s& p3, const vec3s& p4,
    const Expression& u, const Expression& v, const Expression& w,
    const vec3s& q
) {
    return length_Lp(M, q - tri_point(p1,p2,p3,p4,u,v,w)) ;
}
```
3. Algorithm

Problem #1: $F_T(x_i, C_1, C_2)$

$$\int_T \left\| M(x) (x_i - x) \right\|_p^p \, dx$$
3. Algorithm

Intersections of bisectors

Lp-CVT: the making of...

Problem #2: $C(x_i, x_j, x_k)$

```cpp
vec3g<T> three_planes_intersection(
    const Plane<T>& P1,
    const Plane<T>& P2, const Plane<T>& P3
) {
    T b00, b01, b02, b10, b11, b12, b20, b21, b22 ;
    // Note: b is transposed
    comatrix3x3(
        P1.a, P1.b, P1.c,
        P2.a, P2.b, P2.c,
        P3.a, P3.b, P3.c,
        b00, b10, b20,
        b01, b11, b21,
        b02, b12, b22
    ) ;
    return -T(1) / det3x3(
        P1.a, P1.b, P1.c,
        P2.a, P2.b, P2.c,
        P3.a, P3.b, P3.c
    ) * vec3g<T>(
        P1.d * b00 + P2.d * b01 + P3.d * b02,
        P1.d * b10 + P2.d * b11 + P3.d * b12,
        P1.d * b20 + P2.d * b21 + P3.d * b22
    ) ;
}
```
3. Algorithm

Lp-CVT: the making of...
Assembling $F$ and $\nabla F$

$F(p_i, p_1, p_2)$

$x_i$  \quad $C(x_i, x_j, x_k)$  \quad $C(x_i, x_l, x_m)$

**FunctionCompose** class
+ chain rule to compute derivatives
3. Algorithm

\[ \int_{T} \left\| M(x) (x_i - x) \right\|_p^p \, dx \]

Written in C++ + Ginac

It works! (the 2D demo is done with this «symbolic» code), but too slow for 3D
3. Algorithm

Lp-CVT: the making of....

\[ \int_{T} \left\| M(x) (x_i - x) \right\|_{p}^{p} dx \]

Written in C++ + Ginac,
Instanciation with Exp class.

It works ! (the 2D demo is done with this « symbolic » code), but too slow for 3D

void eval_F_Lp(
    unsigned int n,
    const double* X,
    double& F_Lp,
    double* grad_F_Lp
) ;
3. Algorithm

Can we do better?

Do not let the computer think for us,

We need to crank the algebra…
3. Algorithm: chalk and board

\[ \int_{a}^{b} x^q \, dx = \frac{b^{q+1} - a^{q+1}}{1 + q} = \frac{b - a}{1 + q} \left[ a^q + a^{q-1}b + \cdots + ab^{q-1} + b^q \right]. \]

univariate, 1D version

*can we have a nD version on simplices?*

[Lasserre and Avrachenkov 2000]
3. Algorithm: chalk and board

Homogeneous polynomial in polar form $H()$:

eample: $x^4 = H(x,x,x,x)$

$H(x,x,x,x) = x_1 \cdot x_2 \cdot x_3 \cdot x_4$

replace degree $q$ with multilinear form of $q$ variables
3. Algorithm: chalk and board

Homogeneous polynomial in polar form $H()$:

example: $x^4 = H(x,x,x,x)$

$H(x,x,x,x) = x_1 \times x_2 \times x_3 \times x_4$

replace degree $q$ with multilinear form of $q$ variables

$$
\int_{\Delta_n} H(x,x,\ldots,x) \, dx = \frac{\text{vol}(\Delta_n)}{\binom{n+q}{q}} \left[ \sum_{0\leq i_1 \leq i_2, \ldots, \leq i_q \leq n} H(x_{i_1}, x_{i_2}, \ldots, x_{i_q}) \right]
$$

nD version (n: dimension; q: degree of the polynomial)

[Lasserre and Avrachenkov 2000]
3. Algorithm: chalk and board
3. Algorithm: chalk and board

\[ F = \int \| x - x_1 \|_p \, dx \]

\[ T(x_1, C_2, C_3, C_4) \]
3. Algorithm : chalk and board

\[ F = \int \left\| x - x_1 \right\|^p \, dx \]

\[ T(x_1, C_2, C_3, C_4) \]

Translate \( x_1 \) at origin :

\[ F = \int \left\| x \right\|^p \, dx \]

\[ T(0, C_2-x_1, C_3-x_1, C_4-x_1) \]
3. Algorithm: chalk and board

\[ \int_{\Delta_n} H(x, x, \ldots, x) \, dx = \frac{\text{vol}(\Delta_n)}{\binom{n+q}{q}} \left[ \sum_{0 \leq i_1 \leq i_2 \leq \cdots \leq i_q \leq n} H(x_{i_1}, x_{i_2}, \ldots, x_{i_q}) \right] \]

nD version (n: dimension; q: degree of the polynomial)

\[ F = \int \left\| x \right\|^p d x \]

\[ T(0, C_2-x_1, C_3-x_1, C_4-x_1) \]
3. Algorithm: chalk and board

\[ \int_{\Delta_n} H(x, x, \ldots, x) \, dx = \frac{\text{vol}(\Delta_n)}{\binom{n+q}{q}} \left[ \sum_{0 \leq i_1 \leq i_2, \ldots, i_q \leq n} H(x_{i_1}, x_{i_2}, \ldots, x_{i_q}) \right] \]

nD version (n: dimension; q: degree of the polynomial)

\[ F = \int \| x \|_{p}^{p} \, dx \]

\[ T(0, C_2-x_1, C_3-x_1, C_4-x_1) \]

\[ H(x, x) = x^p \quad H(x_1, \ldots, x_p) = x_1 \cdots x_p \]

\[ U_1 = C_2-x_1; \quad U_2 = C_3-x_1; \quad U_3 = C_4-x_1 \]
3. Algorithm: chalk and board

\[ \int_{\Delta_n} H(x, x, \ldots, x) \, dx = \frac{\text{vol}(\Delta_n)}{(n+q)} \left[ \sum_{0 \leq i_1 \leq i_2 \ldots \leq i_q \leq n} H(x_{i_1}, x_{i_2}, \ldots, x_{i_q}) \right] \]

nD version (n: dimension; q: degree of the polynomial)

\[ F = \int T(0, C_2-x_1, C_3-x_1, C_4-x_1) \]

\[ F = \frac{\text{Vol}(T)}{10 \, p!} \sum_{i_1, i_2, \ldots, i_p \text{ in } (1, 2, 3)} (U_{i_1} \times U_{i_2} \times U_{i_3} \ldots \times U_{i_p}) \]

H(x, x) = x^p \quad H(x_1, \ldots, x_p) = x_1 \times \ldots \times x_p

U_1 = C_2-x_1;
U_2 = C_3-x_1;
U_3 = C_4-x_1

This solves problem #1
3. Algorithm: chalk and board

\[ \int_{\Delta_n} H(x, x, \ldots, x) \, dx = \frac{\text{vol}(\Delta_n)}{(n+q)} \left[ \sum_{0 \leq i_1 \leq i_2, \ldots, \leq i_q \leq n} H(x_{i_1}, x_{i_2}, \ldots, x_{i_q}) \right] \]

nD version (n: dimension; q: degree of the polynomial)

\[ F = \int_{T(0, C_2-x_1, C_3-x_1, C_4-x_1)} x^p \, dx \]

\[ F = \frac{\text{Vol}(T)}{10 \, p!} \sum_{i_1, i_2, \ldots, i_p \text{ in } (1,2,3)} (U_{i_1} \cdot U_{i_2} \cdot U_{i_3} \ldots \cdot U_{i_p}) \]

This solves problem #1
Problem #2

[Minka] Matrix computation
3. Algorithm: chalk and board

Problem #2: circumcenters and their derivatives

[Minka] Matrix computation
3. Algorithm: chalk and board

Problem #2: circumcenters and their derivatives

[Minka] Matrix computation

C = intersection of 3 bisector planes
Problem #2 : circumcenters and their derivatives

[Minka] Matrix computation

\[ C = \text{intersection of 3 bisector planes} \]

\[ C = A^{-1} B \quad \text{where} \quad A = \begin{pmatrix} [x_1 - x_0]^t \\ [x_2 - x_0]^t \\ [x_3 - x_0]^t \end{pmatrix} \quad ; \quad B = \frac{1}{2} \begin{pmatrix} x_1^2 - x_0^2 \\ x_2^2 - x_0^2 \\ x_3^2 - x_0^2 \end{pmatrix} \]
Problem #2: circumcenters and their derivatives

[Minka] Matrix computation

\[ C = A^{-1} B \quad \text{where} \quad A = \begin{pmatrix} [x_1 - x_0]^t \\ [x_2 - x_0]^t \\ [x_3 - x_0]^t \end{pmatrix} \quad ; \quad B = \frac{1}{2} \begin{pmatrix} x_1^2 - x_0^2 \\ x_2^2 - x_0^2 \\ x_3^2 - x_0^2 \end{pmatrix} \]
3. Algorithm: chalk and board

Problem #2: circumcenters and their derivatives

[Minka] Matrix computation

\[ C = A^{-1}B \quad \text{where} \quad A = \begin{pmatrix} [x_1 - x_0]^t \\ [x_2 - x_0]^t \\ [x_3 - x_0]^t \end{pmatrix} \quad ; \quad B = \frac{1}{2} \begin{pmatrix} x_1^2 - x_0^2 \\ x_2^2 - x_0^2 \\ x_3^2 - x_0^2 \end{pmatrix} \]

(1) \quad d(AB) = dAB + AdB

(2) \quad d(A^{-1}) = -A^{-1}(dA)A^{-1}
3. Algorithm: chalk and board

Problem #2: circumcenters and their derivatives

[Minka] Matrix computation

\[ dC = \begin{pmatrix} [x_1 - x_0]^t \\ [x_2 - x_0]^t \\ [x_3 - x_0]^t \end{pmatrix}^{-1} \begin{pmatrix} [C - x_0]^t & [x_1 - C]^t & 0 & 0 \\ [C - x_0]^t & 0 & [x_2 - C]^t & 0 \\ [C - x_0]^t & 0 & 0 & [x_3 - C]^t \end{pmatrix} \]
3. Algorithm: chalk and board

\[ dC = \left( \begin{bmatrix} [x_1 - x_0]^t \\ [x_2 - x_0]^t \\ [x_3 - x_0]^t \end{bmatrix} \right)^{-1} \begin{bmatrix} (C - x_0)^t & 0 & 0 \\ 0 & (C - x_1)^t & 0 \\ 0 & 0 & (C - x_2)^t \end{bmatrix} \]

Can the results be reproduced by a grad student?

\[
\frac{dE_{L_p}}{dU_1U_2U_3} = \begin{bmatrix} \sum_{\alpha+\beta+\gamma=p;\alpha \geq 1} \alpha U_1^{\alpha-1} U_2^\beta U_3^\gamma \\ \sum_{\alpha+\beta+\gamma=p;\beta \geq 1} \beta U_1^\alpha U_2^{\beta-1} U_3^\gamma \\ \sum_{\alpha+\beta+\gamma=p;\gamma \geq 1} \gamma U_1^\alpha U_2^\beta U_3^{\gamma-1} \end{bmatrix}
\]
3. Algorithm: chalk and board

\[ dC = \left( \begin{bmatrix} [x_1-x_0]^t \\ [x_2-x_0]^t \\ [x_3-x_0]^t \end{bmatrix} \right)^{-1} \left( \begin{bmatrix} [C-x_0]^t \\ [C-x_0]^t \\ [C-x_0]^t \end{bmatrix} \right) \left( \begin{bmatrix} 0 \\ [x_1-C]^t \\ 0 \end{bmatrix} \right) \]

Source code on the DVD
4. Applications and Results

\[ p = 2 \]

\[ M(x) = \text{ppal dir. of curvature}. \]
4. Applications and results

\[ p = 2 \]
\[ M(x) = \text{Normal anisotropy.} \]
4. Applications and results

p = 2
M(x) = Normal anisotropy.

CSG-Remeshing - DEMO

Feature-sensitive meshing
4. Applications and results

\[ p = 8 \]

\[ M(x) = \text{ppal dir. of curvature.} \]
4. Applications and results

+ many other examples in paper and supplemental material.
4. Applications and results
5. Future Work

To $L_\infty$ and beyond!
5. Future Work

- CVT for line segment and graphs (Accepted pending rev.)
- CVT, structured meshing
Aknowledgements

- Bytes : CGAL (Delaunay triangulation)
- Triangles: AIM@Shape, Digital Michelangelo
- Euros : European Research Council
  GOODSHAPE ERC-StG-205693

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