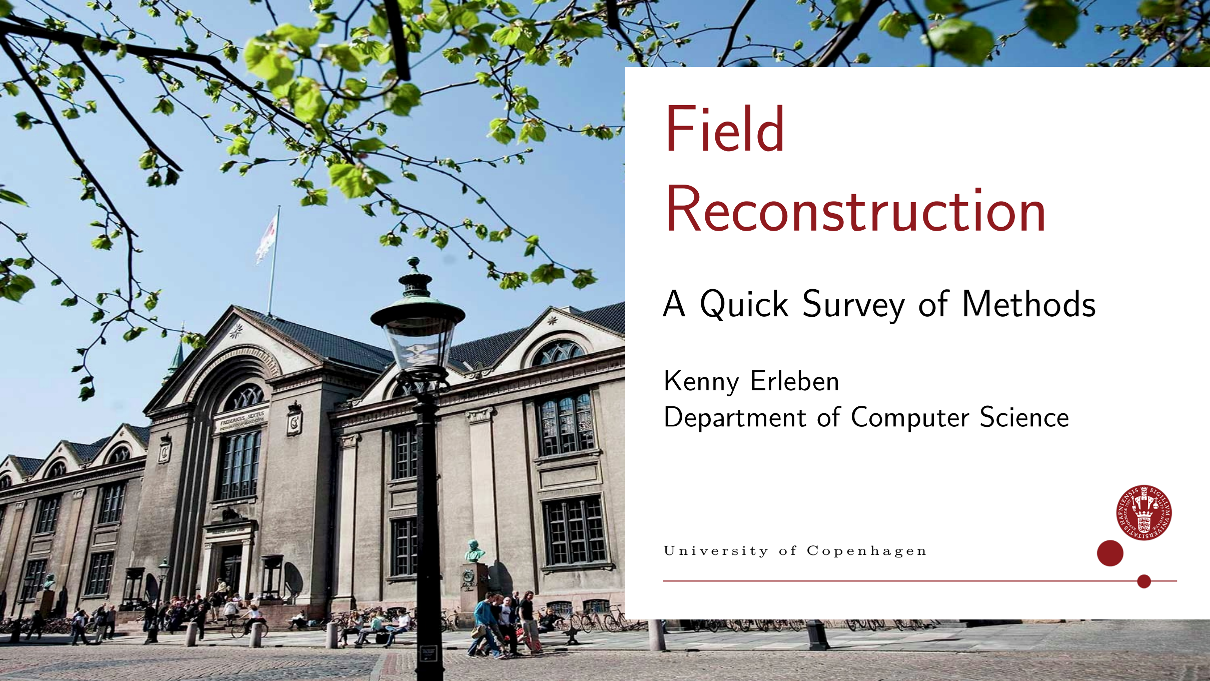


Field Reconstruction

A Quick Survey of Methods

Kenny Erleben
Department of Computer Science

University of Copenhagen

The General Problem

Given

- N points $\mathbf{p}_i \in \mathbb{R}^D$ for $i = 1..N$ (Usually $D = 2$ or $D = 3$)
- For each point we know some value $\phi_i \in \mathbb{R}$

We wish

- to find some function $f(\mathbf{x}) : \mathbb{R}^D \mapsto \mathbb{R}$. We write $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_D \end{bmatrix}^T$
- such that $f(\mathbf{x})$ interpolates all ϕ_i's, $f(\mathbf{p}_i) = \phi_i$ for all i
- and to know the value of $f(\mathbf{x})$ at some location $\mathbf{x} \neq \mathbf{p}_i$ for all i

Observe $f(\mathbf{x})$ as an approximation to some unknown function $\phi(\mathbf{x})$ that we only know some sample values of.

Method of Least Squares (1/5)

Assume a linear basis function $\mathbf{b} : \mathbb{R}^D \mapsto \mathbb{R}^{D+1}$

$$\mathbf{b}(\mathbf{x}) \equiv \begin{bmatrix} 1 & x_1 & \dots & x_D \end{bmatrix}^T$$

then we find optimal coefficients $\mathbf{c} \in \mathbb{R}^{D+1}$ of interpolating function

$$f_{\text{LS}}(\mathbf{x}) \equiv \mathbf{b}(\mathbf{x})^T \mathbf{c}$$

such that

$$\begin{gathered} \mathbf{b}(\mathbf{p}_1)^T \mathbf{c} = \phi_1 \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \mathbf{c} = \phi_N \end{gathered}$$

Method of Least Squares (2/5)

Define

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}(\mathbf{p}_1)^T \\ \mathbf{b}(\mathbf{p}_2)^T \\ \vdots \\ \mathbf{b}(\mathbf{p}_N)^T \end{bmatrix} \quad \text{and} \quad \mathbf{V} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{bmatrix}$$

Then we have

$$\mathbf{Bc} = \mathbf{V}$$

Method of Least Squares (3/5)

This linear system of equations

$$\mathbf{Bc} = \mathbf{V}$$

can be solved with the method of normal equations (aka. pseudo inverse).

$$\underbrace{\mathbf{B}^T\mathbf{B}}_{\mathbf{H}_{\mathsf{LS}}}\mathbf{c} = \mathbf{B}^T\mathbf{V}$$

$$\mathbf{c} = \mathbf{H}_{\mathsf{LS}}^{-1}\mathbf{B}^T\mathbf{V}$$

Method of Least Squares (4/5)

Let us explore the connection to unconstrained optimization

$$\mathbf{Bc} = \mathbf{V}$$

Let us define the error as

$$\varepsilon = \mathbf{Bc} - \mathbf{V}$$

Then we wish to solve for the solution $\mathbf{c}^*$

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \underbrace{\frac{1}{2}\,\varepsilon(\mathbf{c})^2}_{\equiv \psi_{\mathsf{LS}}(\mathbf{c})}$$

Method of Least Squares (5/5)

The first-order solution is given by

$$\begin{aligned} \nabla \psi_{\mathsf{LS}}(\mathbf{c}) = \frac{\partial}{\partial \mathbf{c}} \left(\frac{1}{2} \varepsilon(\mathbf{c})^2 \right) &= \mathbf{0} \\ \mathbf{B}^T \left(\mathbf{B}\mathbf{c} - \mathbf{V} \right) &= \mathbf{0} \end{aligned}$$

Hence, we arrive at the solution

$$\mathbf{c} = \mathbf{H}_{\mathsf{LS}}^{-1} \mathbf{B}^T \mathbf{V}$$

Method of Moving Least Squares (1/3)

The main idea in the moving least squares method is to let the coefficients **c** change depending on **x**.

The trick is to use a weighting function $w(r) = w(\| \mathbf{x} - \mathbf{p}_i \|) = w(\mathbf{x}, \mathbf{p}_i) : \mathbb{R}_0^+ \mapsto \mathbb{R}^+$

$$\begin{aligned} w(\mathbf{x}, \mathbf{p}_1)\mathbf{b}(\mathbf{p}_1)^T\mathbf{c} &= w(\mathbf{x}, \mathbf{p}_1)\phi_1 \\ w(\mathbf{x}, \mathbf{p}_2)\mathbf{b}(\mathbf{p}_2)^T\mathbf{c} &= w(\mathbf{x}, \mathbf{p}_2)\phi_2 \\ &\vdots \\ w(\mathbf{x}, \mathbf{p}_N)\mathbf{b}(\mathbf{p}_N)^T\mathbf{c} &= w(\mathbf{x}, \mathbf{p}_N)\phi_N \end{aligned}$$

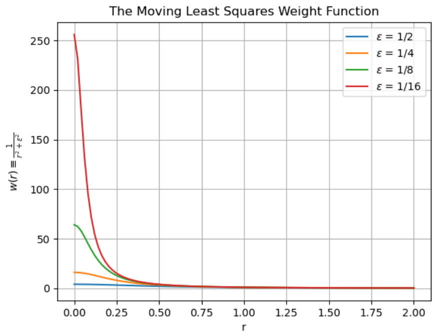
Observe this is just a scaling of each equation that we had in the least squares method.

Method of Moving Least Squares (2/3)

The weight function can be chosen as

$$w(r) \equiv \frac{1}{r^2 + \varepsilon^2}$$

where $r = \| \mathbf{x} - \mathbf{p}_i \|$ for some i and an user-specified $\varepsilon > 0$. One should be careful in selecting parameters as Bumps and dimples occur unless ε is substantially larger than the spacing between points.



Method of Moving Least Squares (3/3)

The weighted equations can now be written in matrix form as

$$\mathbf{W}(\mathbf{x})\mathbf{B}\,\mathbf{c} = \mathbf{W}(\mathbf{x})\mathbf{V}$$

where $\mathbf{W}(\mathbf{x})$ is a diagonal matrix of the weight functions. The resulting normal equations are

$$\left(\underbrace{\mathbf{B}^{T}\left(\mathbf{W}(\mathbf{x})\right)^{2}\mathbf{B}}_{\mathbf{H}_{\mathrm{MLS}}(\mathbf{x})}\right)\mathbf{c} = \mathbf{B}^{T}\left(\mathbf{W}(\mathbf{x})\right)^{2}\mathbf{V}$$

Now, we can evaluate the interpolating function using

$$f_{\mathrm{MLS}}(\mathbf{x}) = \mathbf{b}^{T}(\mathbf{x})\,\underbrace{\left(\mathbf{H}_{\mathrm{MLS}}(\mathbf{x})\right)^{-1}\mathbf{B}^{T}(\mathbf{W}(\mathbf{x}))^{2}\mathbf{V}}_{\mathbf{c}(\mathbf{x})}$$

Augmented Radial Basis Function Method (1/3)

Using the radial basis function, $w : \mathbb{R}^D \mapsto \mathbb{R}$,

$$w(\mathbf{y}) = \| \mathbf{y} \|^2 \log \| \mathbf{y} \|$$

This is the thin plate spline (TPS) radial basis function. The interpolation function is now defined as

$$f_{\text{RBF}}(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{c} + \sum_{j=1}^{N} \lambda_j w(\mathbf{x} - \mathbf{p}_j)$$

where $b(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & \cdots & x_D \end{bmatrix}^T$ and the coefficients are $\mathbf{c} = \begin{bmatrix} c_0 & c_1 & \cdots & c_D \end{bmatrix}^T$. The λ_j's are Lagrange multipliers

Augmented Radial Basis Function Method (2/3)

We have the constraints

$$\begin{aligned}\phi_1 &= f(\mathbf{p}_1)\\ &\vdots\\ \phi_N &= f(\mathbf{p}_N)\end{aligned}$$

and the augmented/orthogonality constraints

$$\sum_{j=1}^{N} \lambda_j \mathbf{b}(\mathbf{p}_j) = \mathbf{0}$$

are also imposed.

Augmented Radial Basis Function Method (3/3)

Defining $w_{ij} = w(\mathbf{p}_i - \mathbf{p}_j)$ and for $D = 3$ this can be formulated as a linear system

$$\underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} & 1 & \mathbf{p}_{1,1} & \mathbf{p}_{1,2} & \mathbf{p}_{1,3} \\ w_{21} & w_{22} & \cdots & w_{2N} & 1 & \mathbf{p}_{2,1} & \mathbf{p}_{2,2} & \mathbf{p}_{2,3} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} & 1 & \mathbf{p}_{N,1} & \mathbf{p}_{N,2} & \mathbf{p}_{N,3} \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ \mathbf{p}_{1,1} & \mathbf{p}_{2,1} & \cdots & \mathbf{p}_{N,1} & 0 & 0 & 0 & 0 \\ \mathbf{p}_{1,2} & \mathbf{p}_{2,2} & \cdots & \mathbf{p}_{N,2} & 0 & 0 & 0 & 0 \\ \mathbf{p}_{1,3} & \mathbf{p}_{2,3} & \cdots & \mathbf{p}_{N,3} & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{H}_{\text{RBF}}} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \\ c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This is a symmetric positive definite system and will always have a unique solution.

Field Reconstruction Problems

Typical for 2D FVM on a triangle mesh

- Normal velocities $v = \mathbf{n} \cdot \mathbf{u}$ are at face centers (edge midpoints in 2D)
- Pressure values are at cell centers (triangle circumcenters in 2D)

Typical Problems:

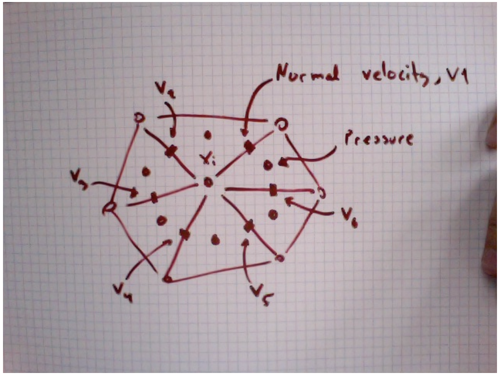
- We want to know the full velocity vector at i^{th} vertex
- We want to know the pressure gradient at the i^{th} vertex

Solution:

- Use moving least squares on the neighborhood of i^{th} vertex

The position of the i^{th} vertex will be given by $\mathbf{x}_i$

Vertex Neighborhood 2D Example



Observe i^{th} vertex has N cell (triangles) neighbors and N face (edges) neighbors.

Vertex Pressure Gradients

- Use either f_{RBF}, f_{LS}, or f_{MLS} to interpolate $p(\mathbf{x}_i) = f(\mathbf{x}_i)$
- Take the analytical derivative of f_{RBF}, f_{LS}, or f_{MLS} to get the pressure gradient

$$\nabla p(\mathbf{x}_i)^T = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_i}$$

Normal velocities at face centers to full velocities at vertices

Vertex Velocity Field Reconstruction

Assume that the velocity field is sufficiently linear

$$\mathbf{u}(\mathbf{x}) \equiv \mathbf{f}_{\mathrm{MLS}}(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{c} = \mathbf{c}_0 + \sum_{j=1}^{D} x_j \mathbf{c}_j$$

where $\mathbf{c}_0, \mathbf{c}_1, ..\mathbf{c}_D \in \mathbb{R}^D$ are coefficients and we have the constraints

$$\mathbf{n}_1 \cdot \mathbf{u}(\mathbf{p}_1) = v_1$$
$$\vdots$$
$$\mathbf{n}_N \cdot \mathbf{u}(\mathbf{p}_N) = v_N$$

where $\mathbf{p}_j$ are the edge midpoints and v_j the corresponding normal velocity at $\mathbf{p}_j$

Reverse Thinking

Imagine we know the full velocities at the edges $\mathbf{u}_1, \ldots, \mathbf{u}_n$, then linear basis functions implies

$$\underbrace{\begin{bmatrix} 1 & x_1 & y_1 \\ \vdots & & \vdots \\ 1 & x_N & y_N \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} c_{0,x} \\ c_{1,x} \\ c_{2,x} \end{bmatrix}}_{\mathbf{c}_x} = \underbrace{\begin{bmatrix} \mathbf{u}_{1,x} \\ \vdots \\ \mathbf{u}_{N,x} \end{bmatrix}}_{\mathbf{v}_x} \quad \text{and} \quad \underbrace{\begin{bmatrix} 1 & x_1 & y_1 \\ \vdots & & \vdots \\ 1 & x_N & y_N \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} c_{0,y} \\ c_{1,y} \\ c_{2,y} \end{bmatrix}}_{\mathbf{c}_y} = \underbrace{\begin{bmatrix} \mathbf{u}_{1,y} \\ \vdots \\ \mathbf{u}_{N,y} \end{bmatrix}}_{\mathbf{v}_y}$$

where $\mathbf{p}_i = (x_i, y_i)^T$. We have omitted the weight $\mathbf{W}$ for readability.

Projecting onto Normals

Or more compactly

$$\begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \end{bmatrix}}_{\mathbf{c}} = \underbrace{\begin{bmatrix} \mathbf{V}_x \\ \mathbf{V}_y \end{bmatrix}}_{\mathbf{V}}$$

We do not know the $\mathbf{u}_i$'s stored in the $\mathbf{V}$-vectors. Instead we know the $v_i = \mathbf{n}_i \cdot \mathbf{u}_i$'s. So we have to retry the problem setup using the projection of velocities onto normals instead.

Projecting onto Normals

To write up the projected velocities we define the diagonal matrices

$$\mathbf{N}_x = \begin{bmatrix} \mathbf{n}_{1,x} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathbf{n}_{N,x} \end{bmatrix} \quad \text{and} \quad \mathbf{N}_y = \begin{bmatrix} \mathbf{n}_{1,y} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathbf{n}_{N,y} \end{bmatrix}$$

Then we may now write

$$\underbrace{\begin{bmatrix} \mathbf{N}_x\mathbf{B} & \mathbf{N}_y\mathbf{B} \end{bmatrix}}_{\mathbf{B}'} \underbrace{\begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \end{bmatrix}}_{\mathbf{c}} = \underbrace{\begin{bmatrix} \mathbf{N}_x\mathbf{V}_x + \mathbf{N}_y\mathbf{V}_y \end{bmatrix}}_{\mathbf{V}'}$$

Vertex Velocity Field Reconstruction

For $D = 2$ and $\mathbf{p}_i = (x_i, y_i)^T$ we write

$$\mathbf{B}' = \begin{bmatrix} \mathbf{n}_1^T & x_1\mathbf{n}_1^T & y_1\mathbf{n}_1^T \\ \vdots & & \vdots \\ \mathbf{n}_N^T & x_N\mathbf{n}_N^T & y_N\mathbf{n}_N^T \end{bmatrix}, \mathbf{V}' = \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix}, \text{ and } \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix},$$

then

$$\mathbf{B}'\,\mathbf{c} = \mathbf{V}'$$

This is a well-known form and we can extend it to the moving least squares formalism

$$\mathbf{W}(\mathbf{x})\mathbf{B}'\,\mathbf{c}(\mathbf{x}) = \mathbf{W}(\mathbf{x})'$$

Once we know $\mathbf{c}_j$'s we can compute $u(\mathbf{x}_i)$.

Normal velocities at face centers to full velocities at cell centers

Cell Center Velocity Field (1/2)

Assume constant cell velocity $\mathbf{u}(\mathbf{x}) = \mathbf{c}_0$ for a triangle i, j and k then

$$\mathbf{n}_i^T \mathbf{u} = v_i$$
$$\mathbf{n}_j^T \mathbf{u} = v_j$$
$$\mathbf{n}_k^T \mathbf{u} = v_k$$

where **n**'s are outward unit normals.

Cell Center Velocity Field (2/2)

Now

$$\mathbf{B} = \begin{bmatrix} \mathbf{n}_i^T \\ \mathbf{n}_j^T \\ \mathbf{n}_k^T \end{bmatrix}, \mathbf{V} = \begin{bmatrix} v_i \\ v_j \\ v_k \end{bmatrix}, \text{and } \mathbf{c} = \mathbf{c}_0$$

Then we solve

$$\mathbf{B}\,\mathbf{c} =$$

This is just the least squares method.

Full velocities at cell centers to full velocity at vertices

Some Convex Polytope Terminology

- A convex polytope $\mathcal{P}$ in $\mathbb{R}^D$ is the convex hull of N affinely independent points $\mathbf{p}_i$ where $N > D$.
- The polytope is bounded by a set of $(D-1)$ dimensional faces with unit outward normals $\mathbf{n}_a$ where a is the index of a face.
- Let $\mathcal{N}_i$ be the face index set of the i^{th} vertex such that if $a \in \mathcal{N}_i$ then $i \in a$.
- If $|\mathcal{N}_i| = D$ we say i^{th} is a simple vertex
- If all vertices of the polytope is simple then we say we have a simple polytope

Observations

- All 2D convex polytopes are simple
- A tetrahedron is a simple polytope
- A square pyramid is not a simple polytope

Weight Function Definition (1/2)

We define the weight function

$$w_i(\mathbf{x}) = \frac{\mathbf{vol}(\mathcal{N}_i)}{\Pi_{a\in\mathcal{N}_i}\left(\mathbf{n}_a^T\left(\mathbf{p}_i - \mathbf{x}\right)\right)}$$

where $\mathbf{vol}(\mathcal{N}_i)$ is the “volume” spanned by the face normals of the faces in $\mathcal{N}_i$.

Weight Function Definition (2/2)

In 2D if $\mathcal{N}_i = \{a, b\}$ then

$$\mathbf{vol}(\mathcal{N}_i) = \det\left(\begin{bmatrix}\mathbf{n}_a^T \\ \mathbf{n}_b^T\end{bmatrix}\right)$$

In 3D if $\mathcal{N}_i = \{a, b, c\}$ then

$$\mathbf{vol}(\mathcal{N}_i) = \det\left(\begin{bmatrix}\mathbf{n}_a^T \\ \mathbf{n}_b^T \\ \mathbf{n}_c^T\end{bmatrix}\right)$$

General Barycentric Coordinate Definition

The barycentric coordinate is defined as

$$\omega_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_{j=1}^{N} w_j(\mathbf{x})}$$

For non-simple vertices

- Infinitesimal perturbation faces in $\mathcal{N}_i$ apart (splitting of vertex i)
- Continue to perturbation until all split vertices are simple
- Compute barycentric coordinates of each split vertex (which is simple) and add up the results

Primary Delaunay Mesh and Dual Circumcenter Voronoi Cell

Observations

- The dual cells are convex
- The dual cells are non-self-intersecting
- The dual edges are orthogonal to primal edges

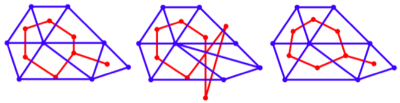


Figure 3: *Different choices of triangulations (blue) and dual meshes (red) in 2D. From left to right: 1) Delaunay triangulation with circumcentric dual. 2) Non-Delaunay triangulation with circumcentric dual. The dual mesh is self-intersecting. 3) Delaunay triangulation with barycentric dual. Primal/dual edge pairs lack orthogonality. Based on figures by Perot & Subramaniam [PS07].*

Taken from Batty and Xenos et. al. 2010

Voronoi Barycentric Coordinate Simplification

Trick:

- Want to rewrite the weight function so we do not have to explicitly work with the dual Voronoi mesh but can use the primary Delaunay mesh

Observations for Voronoi cell of primary vertex i^{th}

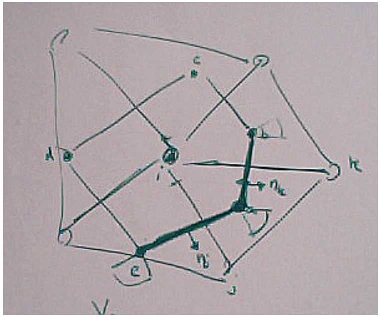
- The Voronoi cell center vertex is shared by the one-ring neighborhood of primary triangles/tetrahedra $\mathcal{V}_i \equiv \{a, b, c, \ldots\}$
- The Voronoi cell is always convex (and a simple polytope)
- The corners of the Voronoi cell are the triangle/tetrahedron circumcenters $\mathbf{p}_a$, $\mathbf{p}_b$, $\mathbf{p}_c$, $\ldots$

Simplification in 2D

We will

- interpolate over the Voronoi cell of the i^{th} vertex
- simplify the weight function of the a^{th} corner point $\mathbf{p}_a$
- assume a^{th} triangle has vertices i, j and k

Our notation



Simplification in 2D

Due to the orthogonality of primary and dual edges, we have

$$\mathbf{vol}(\mathcal{N}_a) = 2\frac{A_a}{l_{ki}l_{ji}}$$

where

- A_a is the area of the a^{th} triangle
- l_{ki} is the length of the edge from i to k and similar for l_{ji}

We have $\mathcal{N}_a \equiv \{j, k\}$

$$\Pi_{b\in\{j,k\}}\left(\mathbf{n}_b^T\left(\mathbf{p}_a - \mathbf{x}\right)\right) = \Pi_{b\in\{j,k\}}\frac{\left(\mathbf{x}_b - \mathbf{x}_i\right)^T}{l_{bi}}\left(\mathbf{p}_a - \mathbf{x}\right)$$

Simplification in 2D

Putting it together we have

$$w_a(\mathbf{x}) = \frac{2A_a}{\Pi_{b\in\{j,k\}}\left(\mathbf{x}_b - \mathbf{x}_i\right)^T\left(\mathbf{p}_a - \mathbf{x}\right)}$$

3D Voronoi Barycentric Coordinate Definition

Repeating the steps in 3D one finds

$$w_a(\mathbf{x}) = \frac{6V_a}{\Pi_{b\in\{j,k,m\}}\left(\mathbf{x}_b - \mathbf{x}_i\right)^T\left(\mathbf{p}_a - \mathbf{x}\right)}$$

where

- V_a is the volume of the tetrahedron associated with corner point a
- The corner point $\mathbf{p}_a$ is the circumcenter of the a^{th} tetrahedron
- The a^{th} tetrahedron has nodes i, j, k, and m
- The i^{th} vertex is the Voronoi cell center

Further Reading

- Greg Turk and James F. O'Brien: Shape Transformation Using Variational Implicit Functions, Siggraph 1999.
- Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk: Interpolating and Approximating Implicit Surfaces from Polygon Soup, Siggraph 2004.
- Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner: Animating Gases with Hybrid Meshes, Siggraph 2005
- Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien: Fluid Animation with Dynamic Meshes, Siggraph 2006.
- Joe Warren, Scott Schaefer, Anil N. Hirani, and Mathieu Desbrun: Barycentric Coordinates for Convex Sets, (unpublished manuscript?) 2005

Assignment

Recall the method of least squares from Page 3 to Page 7.

- What are the dimensions of $\mathbf{B}$?
- Under what conditions are the system $\mathbf{Bc} = \mathbf{V}$ over/under constrained?
- What condition must $\mathbf{B}$ fulfill for the pseudo inverse to work? (Hint consider rank)
- What is the gradient of $f_{\mathrm{LS}}(\mathbf{x})$? (Hint: use analytical differentiation)

Assignment

Recall the moving least squares method on Pages 8 to 10.

- Define the error as $\varepsilon = \mathbf{W}(\mathbf{x})\mathbf{B}\,\mathbf{c} - \mathbf{W}(\mathbf{x})\mathbf{V}$, use this to define the unconstrained minimizer ψ_{MLS} and show the connection to optimization problems.
- What is the dimensions of $\mathbf{H}_{\mathsf{MLS}}$?
- Under what conditions are $\mathbf{H}_{\mathsf{MLS}}$ non-singular?
- What is the gradient of $f_{\mathsf{MLS}}(\mathbf{x})$? (Hint: use analytical differentiation)
- Make a sampling of the function $\phi(\mathbf{x}) \equiv (x_1 + x_2)^n$ at some 100 uniform random points $\mathbf{p}_i \in \mathbb{R}^2$ inside the box $[0..1] \times [0..1]$. Compare the accuracy of using least squares f_{LS} and moving least squares f_{MLS} at the same 10 random locations inside the box. Try for different values of $n = 1, 2, 3, \ldots$.
- For what values of ε does moving least squares work best? (Hint: use ψ_{MLS} to measure the overall error, can you explain why you should use this?)

Assignment

Consider the augmented Radial basis function from Pages 11-13.

- Discuss the role of the extra orthogonality constraints.
- What is the dimensions of the $\mathbf{H}_{\mathrm{RBF}}$?
- What is the gradient of $f_{\mathrm{RBF}}(\mathbf{x})$? (Hint: use analytical differentiation)

Assignment

Consider the velocity reconstruction from Page 22. Assume we have a 2D case and that all points $\mathbf{p}_i \in \mathbb{R}^2$ for $i = 1..n$ and $n \geq 6$ is such that

- $\mathbf{B}$ has full column rank

Now Consider what “obvious” conditions one must require of $\mathbf{n}_i$ such that $\mathbf{B}'$ does not lose full column rank.

Assignment

- Create a 2D triangle mesh. Let the total number of vertices be given by K
- Initialize a ground truth full velocity field at the vertices. Let $\mathbf{m} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k$ then $\mathbf{u}_k = \widehat{\mathbf{x}_k - \mathbf{m}}$. The hat operation is defined as $\widehat{(x_1, x_2)}^T = (-x_2, x_1)^T$ (Hint: Consider if this velocity field is a linear mapping or not?).
- Use the ground truth velocity field to compute normal velocities at face centers. (Hint if vertex i and j are both incident to the edge a then $v_a = \mathbf{n}_a^T (\mathbf{u}_i + \mathbf{u}_j)/2$)
- From the normal velocities reconstruct a full velocity field at the vertices.
- Compare the reconstructed velocity field against the ground truth velocity field.

Assignment

- Extend the 2D triangle example from Page 43 to reconstruct cell center (circumcenter) velocities
- Compare the reconstructed cell center velocities with the ground truth vertex velocity field, are the velocities meaningful? (Hint, remember you have a closed-form solution for the ground truth velocity at the cell centers)
- Consider how good the cell-centered reconstructed velocity field is when the normal velocity face field is not divergence-free with respect to the cells.

Assignment

Recall the formula on Page 36

- Create a Delaunay mesh of hexagon shape, let corner points be $\mathbf{x}_j$,$\mathbf{x}_k$,. . ., $\mathbf{x}_p$ and center point be $\mathbf{x}_i$.
- Assign ground truth values ϕ_a to the circumcenters of triangles
    - Try a constant $\phi_a = 1$ field
    - Try a linear $\phi_a = p_{a,1} + p_{a,2}$ field
- Interpolate the ground truth fields on the spiral points

$$\mathbf{s}_n = \frac{r}{n}\begin{bmatrix}\cos(\Delta\theta n)\\ \sin(\Delta\theta n)\end{bmatrix} \quad \text{for } n = 1 \text{ to } N$$

  where $r = \parallel \frac{\mathbf{p}_a+\mathbf{p}_b}{2} - \mathbf{x}_i \parallel$ and $\Delta\theta = \frac{2\pi}{N}$ and $N = 50$.

- Discuss your interpolation results.

Assignment

- (Advanced) Prove the simplification of the weight function for the 3D case from Page 37