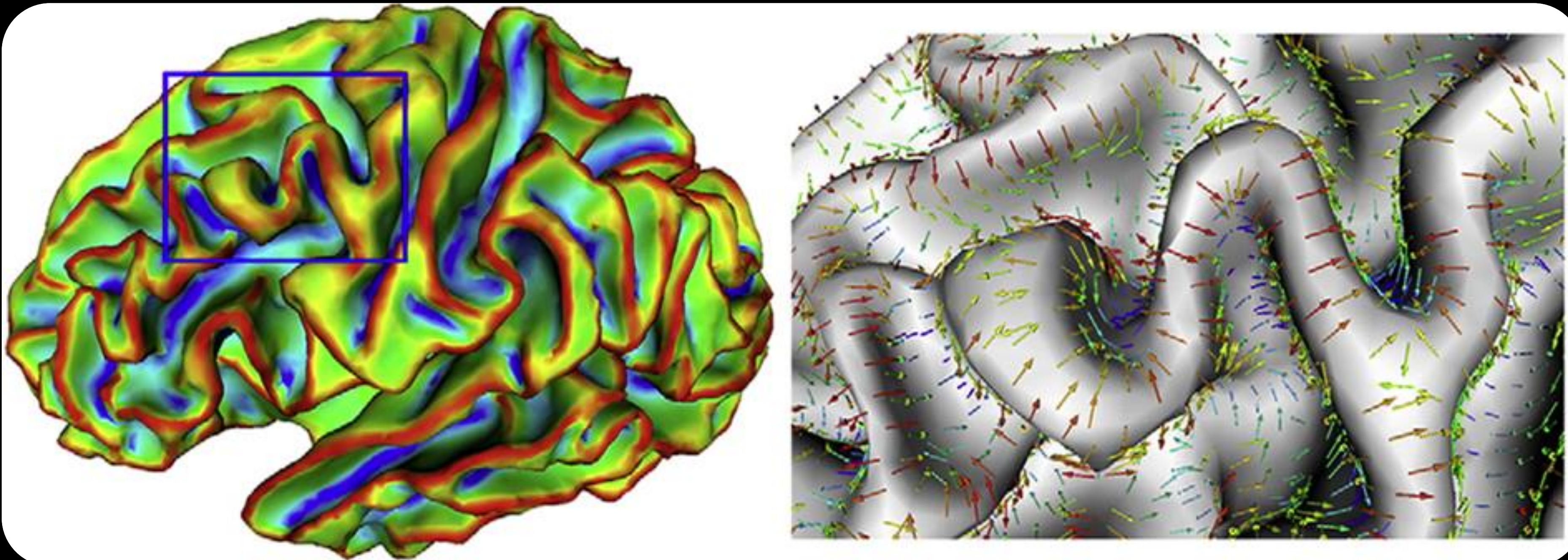
http://www.sciencedirect.com/science/article/pii/S1053811909002663



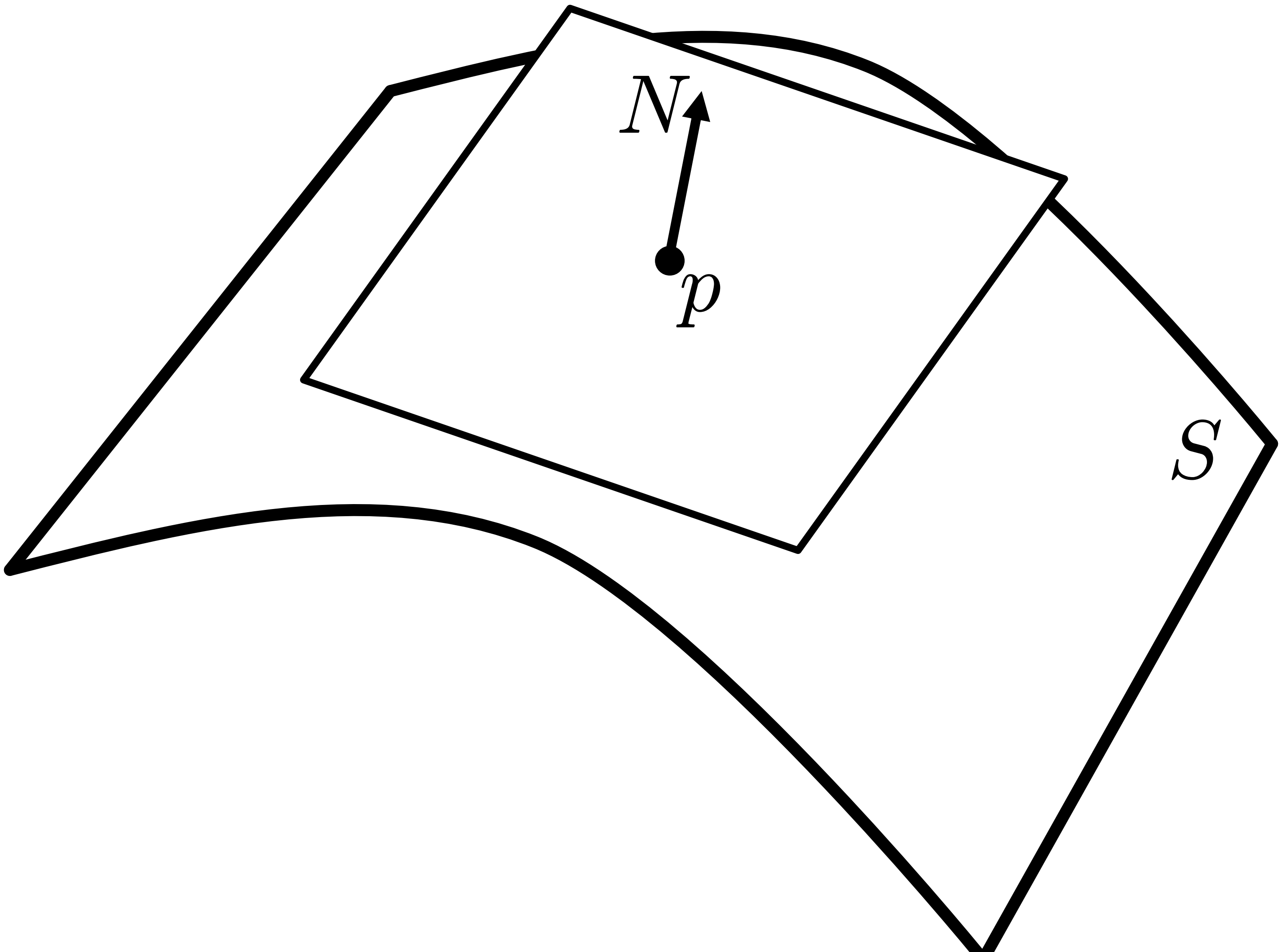
Computing Curvature

CS 468, Spring 2013
Differential Geometry for Computer Science

Justin Solomon and Adrian Butscher

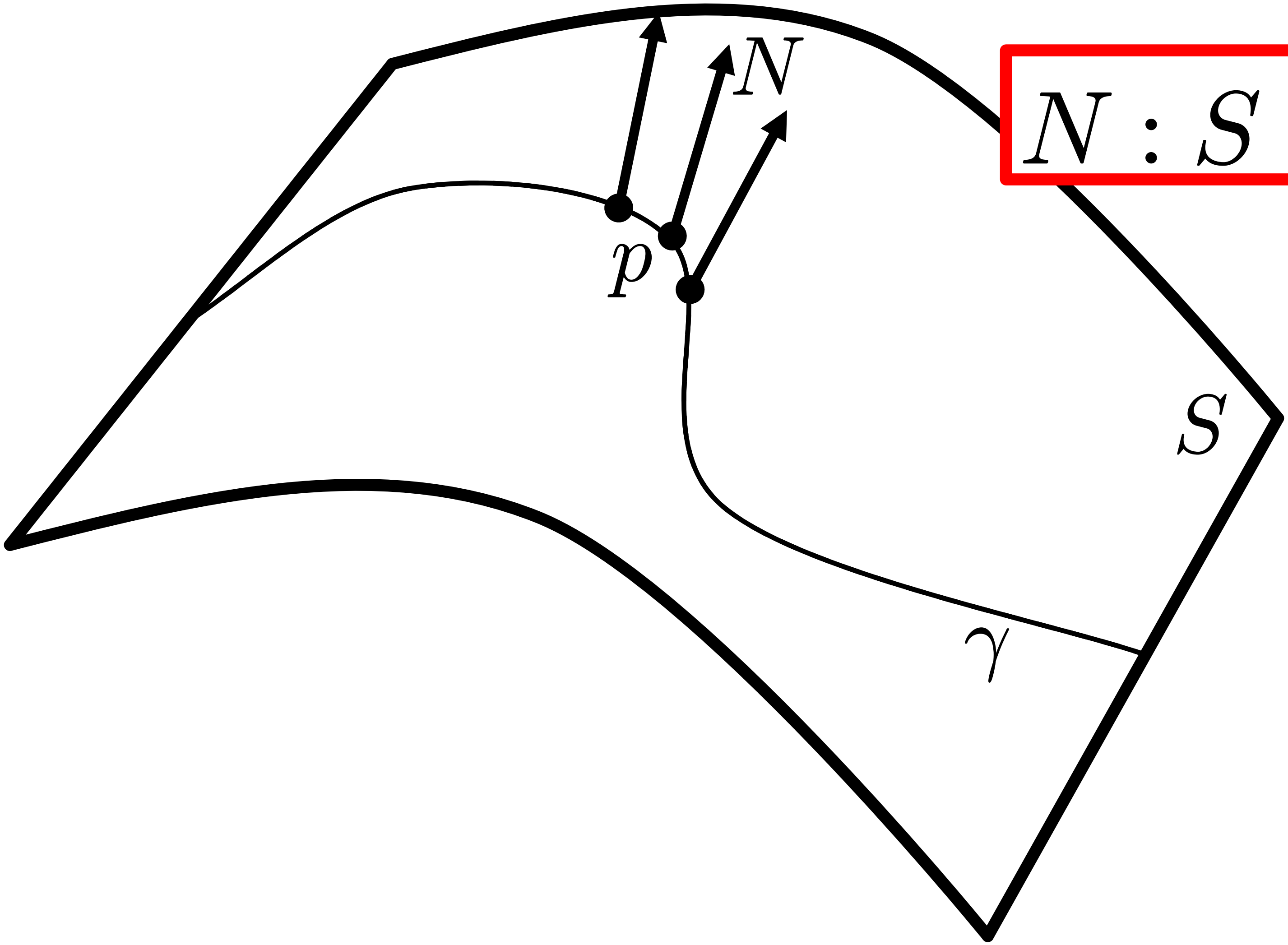
Recall:

Surface Theory



Recall:

Gauss Map



$$N : S \to S^2$$

Recall:

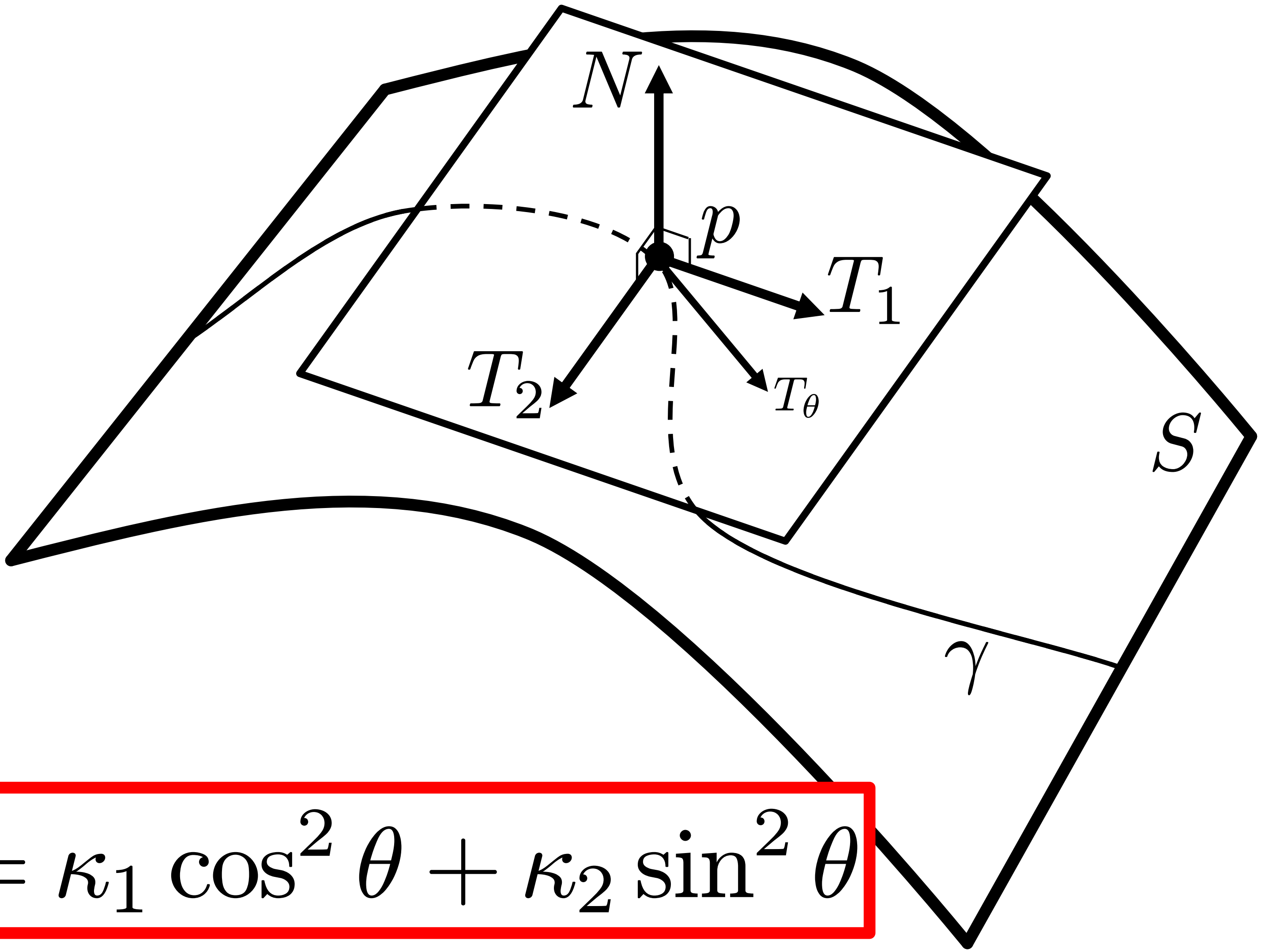
Second Fundamental Form

$$DN_p : T_pS \to T_pS$$

$$A_p(V, W) = -\langle DN_p(V), W \rangle$$

Recall:

Principal Directions and Curvatures

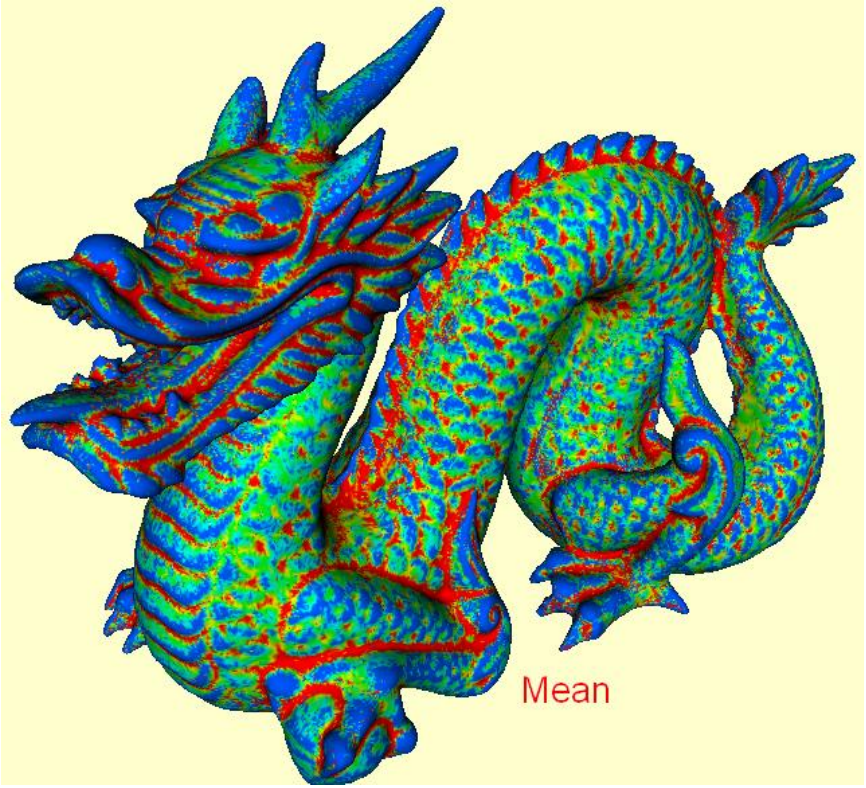


$$\kappa_\theta = \kappa_1 \cos^2\theta + \kappa_2 \sin^2\theta$$

Who Cares?

Curvature completely determines local surface geometry.

Use as a Descriptor

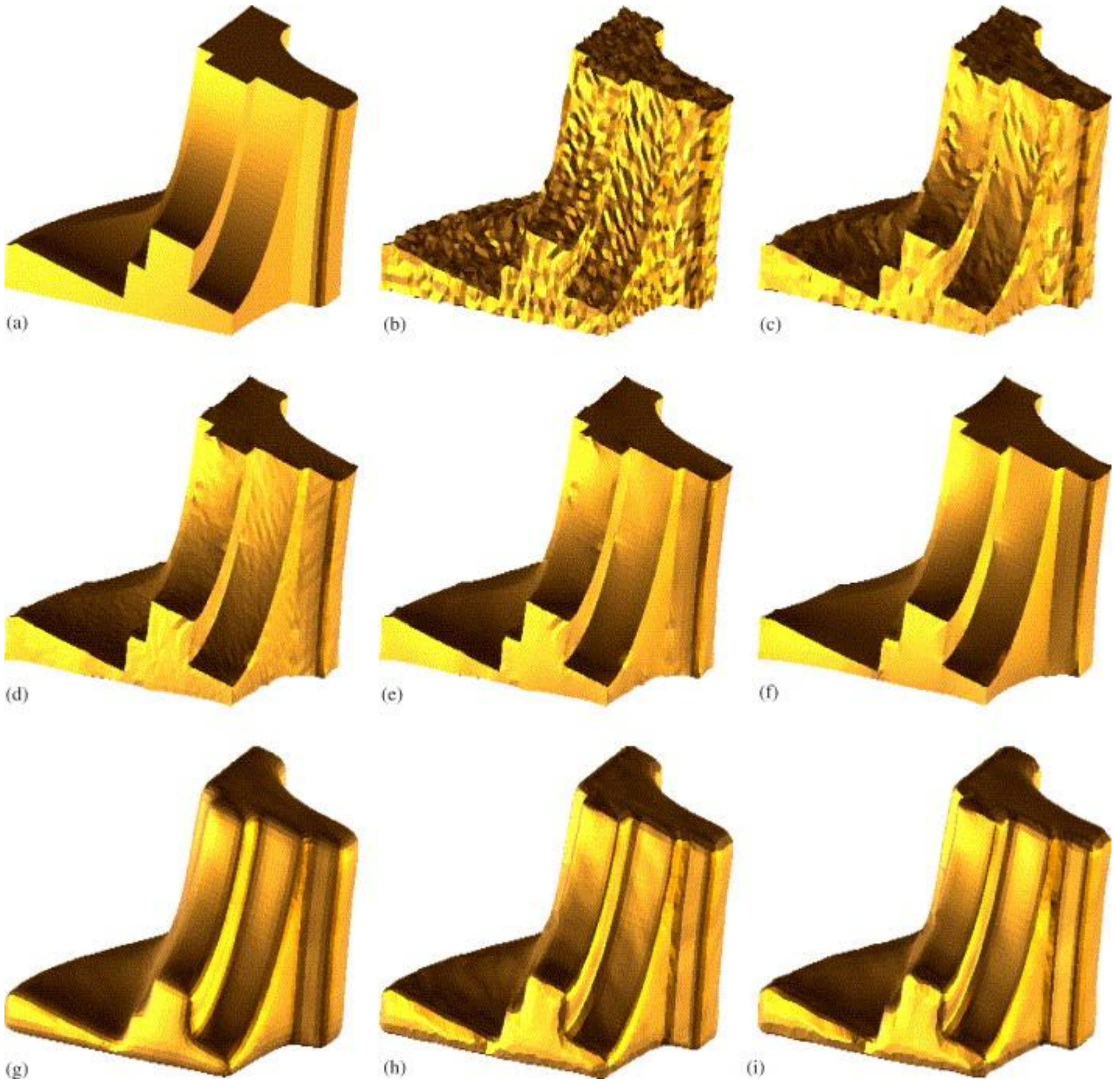


http://graphics.ucsd.edu/~iman/Curvature/

Smoothing and Reconstruction

Linear Surface Reconstruction from Discrete Fundamental Forms on Triangle Meshes
Wang, Liu, and Tong
Computer Graphics Forum 31.8 (2012)

Fairness Measure



Triangular Surface Mesh Fairing via Gaussian Curvature Flow
Zhao, Xu
Journal of Computational and Applied Mathematics 195.1-2 (2006)

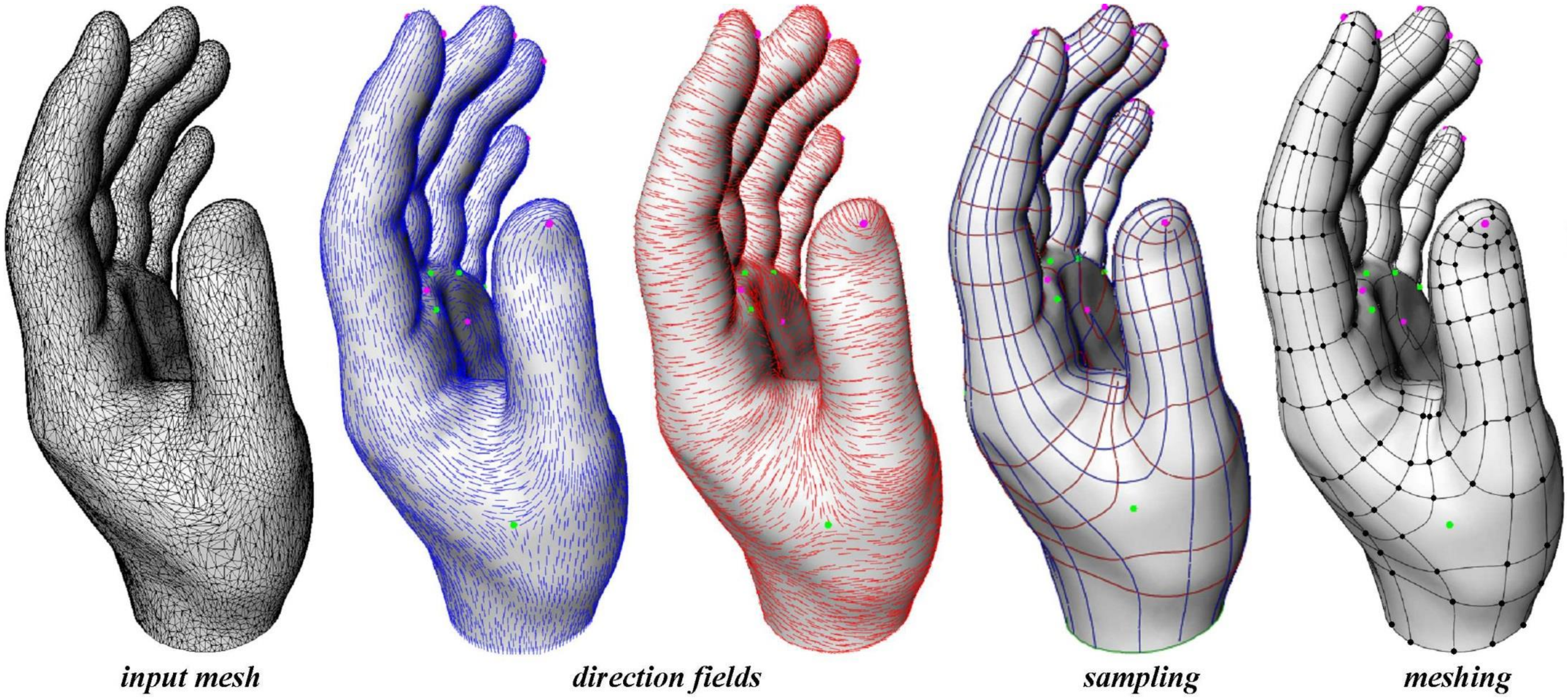
...and many more

Guiding Rendering

Highlight Lines for Conveying Shape
DeCarlo, Rusinkiewicz
NPAR (2007)

http://www.cs.rutgers.edu/~decarlo/pubs/npar07.pdf

Guiding Meshing



Anisotropic Polygonal Remeshing
Alliez et al.
SIGGRAPH (2003)

Special Topic for Me…

US 20090244082A1

(19) **United States**
(12) **Patent Application Publication**
Livingston et al.

(10) **Pub. No.: US 2009/0244082 A1**
(43) **Pub. Date: Oct. 1, 2009**

(54) **METHODS AND SYSTEMS OF COMPARING FACE MODELS FOR RECOGNITION**

(76) Inventors: **Mark A. Livingston**, Alexandria, VA (US); **Justin Solomon**, Oakton, VA (US)

Correspondence Address:
NAVAL RESEARCH LABORATORY
ASSOCIATE COUNSEL (PATENTS)
CODE 1008.2, 4555 OVERLOOK AVENUE, S.W.
WASHINGTON, DC 20375-5320 (US)

(21) Appl. No.: **12/416,716**

(22) Filed: **Apr. 1, 2009**

Related U.S. Application Data

(60) Provisional application No. 61/041,305, filed on Apr. 1, 2008.

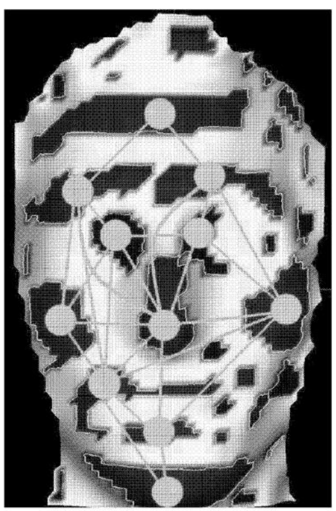
Publication Classification

(51) **Int. Cl.**
G09G 5/00 (2006.01)
G06K 9/46 (2006.01)

(52) **U.S. Cl.** **345/581**; 382/203

(57) **ABSTRACT**

Methods and systems of representation and manipulation of surfaces with perceptual geometric features, using a computer graphics rendering system, include executing algorithmic instructions to compute a plurality of vertices, edges and surfaces in a mesh for the purpose of defining representations of surfaces on grids. Normals and distances are determined for triangular surfaces to be considered. Additionally, height fields of a function are defined. A set of feature curves and a set of feature points are derived, based on the defined function. Infinitesimal movements along the representations of the surfaces are determined, along with derivations of properties of representations of continuous surfaces. Additional determinations of perceptual geometric features include determinations such as zero crossings, parabolic curves, flecnodes, ruffles, gutterpoints, conical points and biflecnodes in a given mesh. After these determinations are made, visual representation are rendered which captures perceptually important features for smoothly varying shapes.



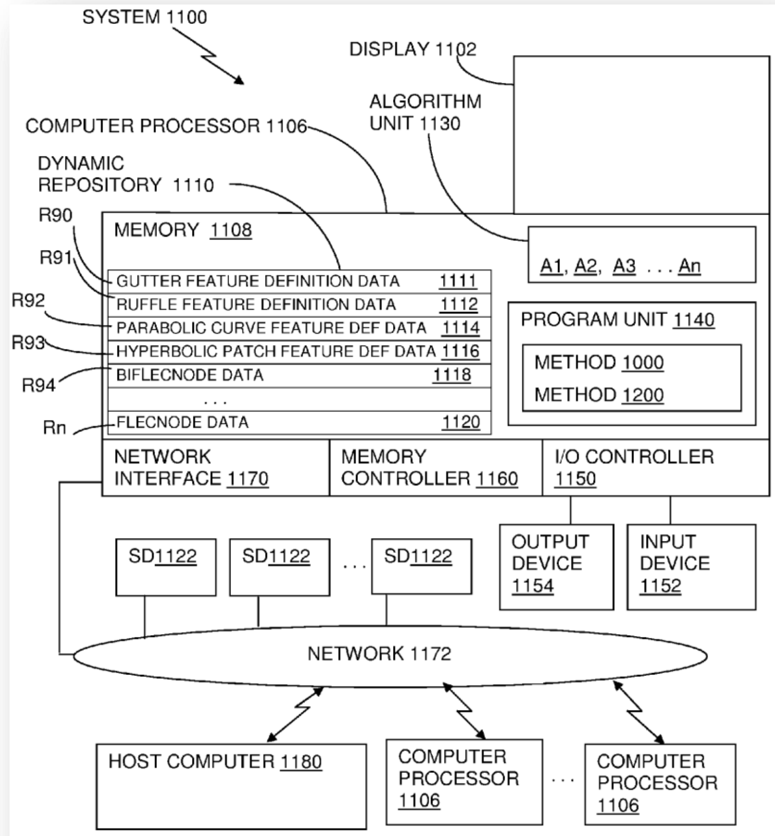


FIG. 11

Standard Citation

ESTIMATING THE TENSOR OF CURVATURE OF A SURFACE FROM A POLYHEDRAL APPROXIMATION

ICCV 1995

Gabriel Taubin

IBM T.J.Watson Research Center
P.O.Box 704, Yorktown Heights, NY 10598
taubin@watson.ibm.com

Abstract

Estimating principal curvatures and principal directions of a surface from a polyhedral approximation with a large number of small faces, such as those produced by iso-surface construction algorithms, has become a basic step in many computer vision algorithms. Particularly in those targeted at medical applications. In this paper we describe a method to estimate the tensor of curvature of a surface at the vertices of a polyhedral approximation. Principal curvatures and principal directions are obtained by computing in closed form the eigenvalues and eigenvectors of certain 3×3 symmetric matrices defined by integral formulas, and

mate principal curvatures at the vertices of a triangulated surface. Both this algorithm and ours are based on constructing a quadratic form at each vertex of the polyhedral surface and then computing eigenvalues (and eigenvectors) of the resulting form, but the quadratic forms are different. In our algorithm the quadratic form associated with a vertex is expressed as an integral, and is constructed in time proportional to the number of neighboring vertices. In the algorithm of Chen and Schmitt, it is the least-squares solution of an overdetermined linear system, and the complexity of constructing it is quadratic in the number of neighbors.

2 The Tensor of Curvature

Taubin Matrix

$$M = \frac{1}{2\pi} \int_{-\pi}^{\pi} \kappa_\theta T_\theta T_\theta^\top \, d\theta$$

$$\kappa_\theta = \kappa_1 \cos^2\theta + \kappa_2 \sin^2\theta$$

$$T_\theta = T_1 \cos\theta + T_2 \sin\theta$$

Taubin Matrix

$$M = \frac{1}{2\pi} \int_{-\pi}^{\pi} \kappa_\theta T_\theta T_\theta^\top \, d\theta$$

- Eigenvectors are N, T_1, and T_2
- Eigenvalues are $\frac{3}{8}\kappa_1 + \frac{1}{8}\kappa_2$ and $\frac{1}{8}\kappa_1 + \frac{3}{8}\kappa_2$

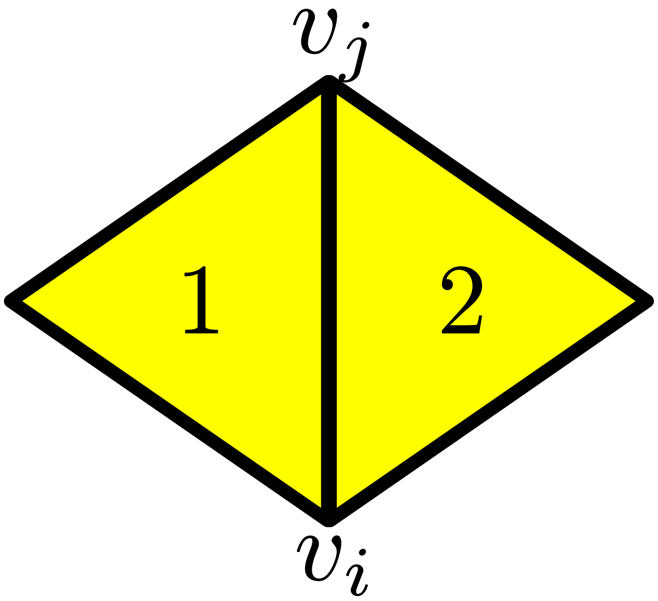
Prove to yourself!

Taubin's Approximation

$$M = \frac{1}{2\pi} \int_{-\pi}^{\pi} \kappa_\theta T_\theta T_\theta^\top \, d\theta$$

$$\tilde{M}_{v_i} = \sum_{v_j \sim v_i} w_{ij} \kappa_{ij} T_{ij} T_{ij}^\top$$

Taubin's Approximation



$$w_{ij} \propto A_1 + A_2$$

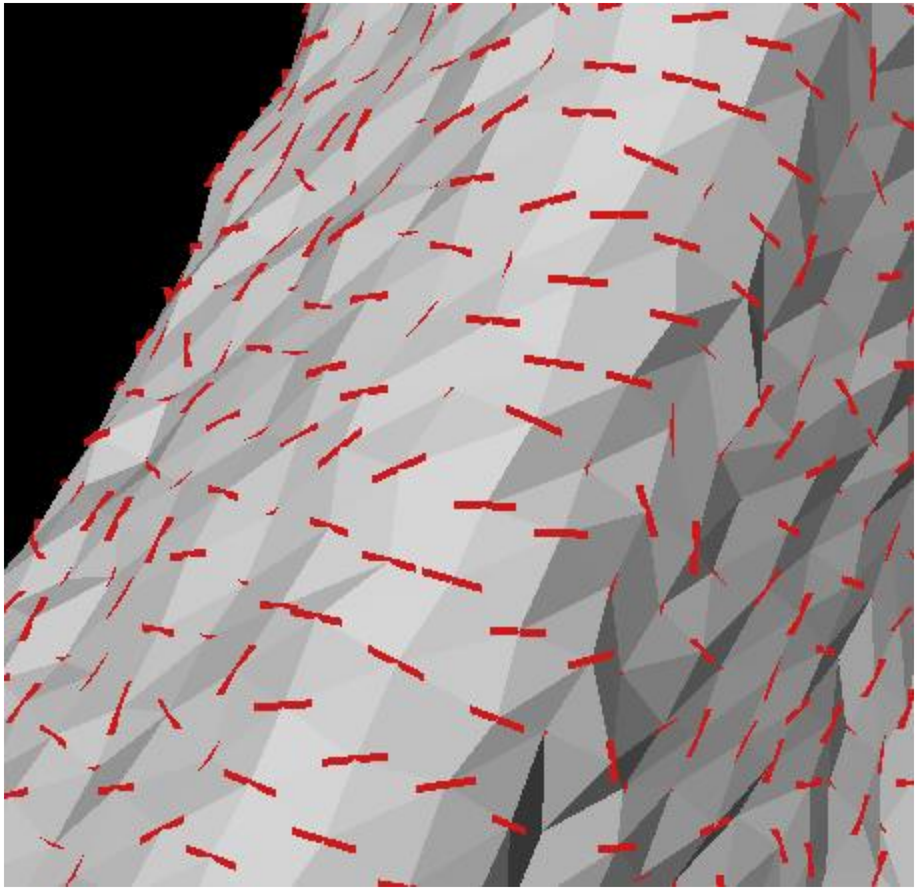
$$\tilde{M}_{v_i} = \sum_{v_j \sim v_i} w_{ij} \kappa_{ij} T_{ij} T_{ij}^\top$$

Taubin's Approximation

Divided difference approximation

$$\tilde{M}_{v_i} = \sum_{v_j \sim v_i} w_{ij} \kappa_{ij} T_{ij} T_{ij}^{\top}$$

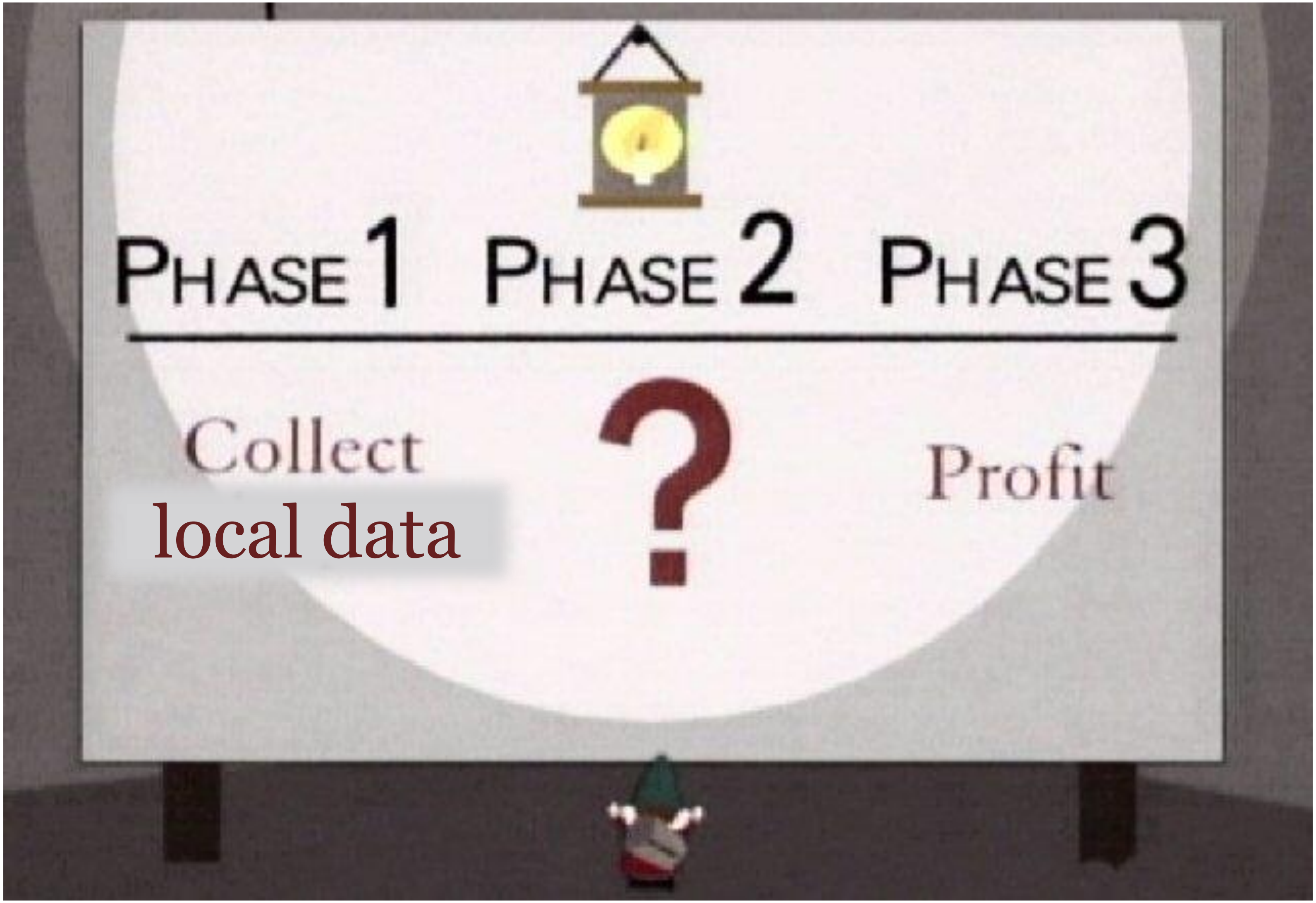
Problem

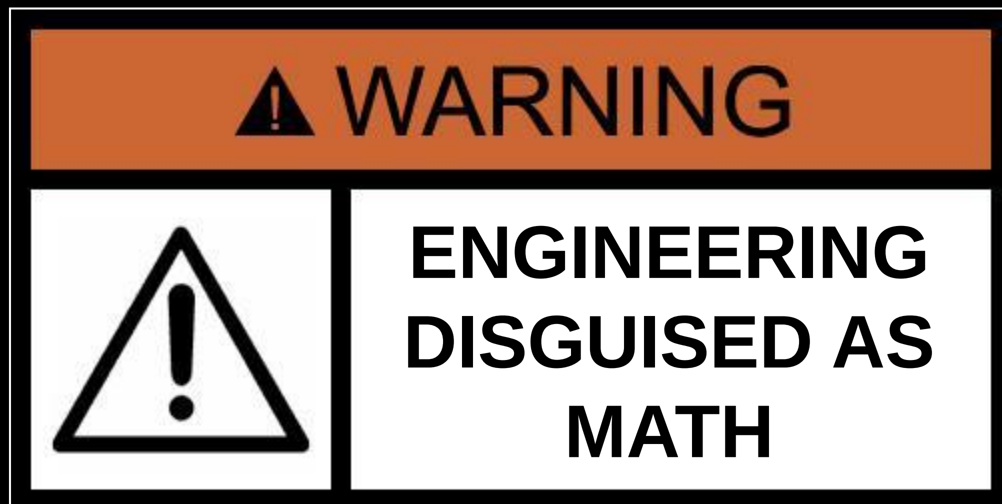


http://iristown.engr.utk.edu/~koschan/paper/CVPR01.pdf

Local estimates are noisy

General Strategy




WARNING
ENGINEERING
DISGUISED AS
MATH

Main Take-Away

Use application to motivate choice of curvature.

Simulation, smoothing, analysis, meshing, nonphotorealistic rendering, …

Another Example

Estimating Curvatures and Their Derivatives on Triangle Meshes

Szymon Rusinkiewicz
Princeton University

3DPVT '04

Abstract

The computation of curvature and other differential properties of surfaces is essential for many techniques in analysis and rendering. We present a finite-differences approach for estimating curvatures on irregular triangle meshes that may be thought of as an extension of a common method for estimating per-vertex normals. The technique is efficient in space and time, and results in significantly fewer outlier estimates while more broadly offering accuracy comparable to existing methods. It generalizes naturally to computing derivatives of curvature and higher-order surface differentials.

1 Introduction

As the acquisition and use of sampled 3D geometry become more widespread, 3D models are increasingly becoming the focus of analysis and signal processing techniques previously applied to data types such as audio, images, and video. A key component of algorithms such as feature detection, filtering, and indexing, when applied to both geometry and other data

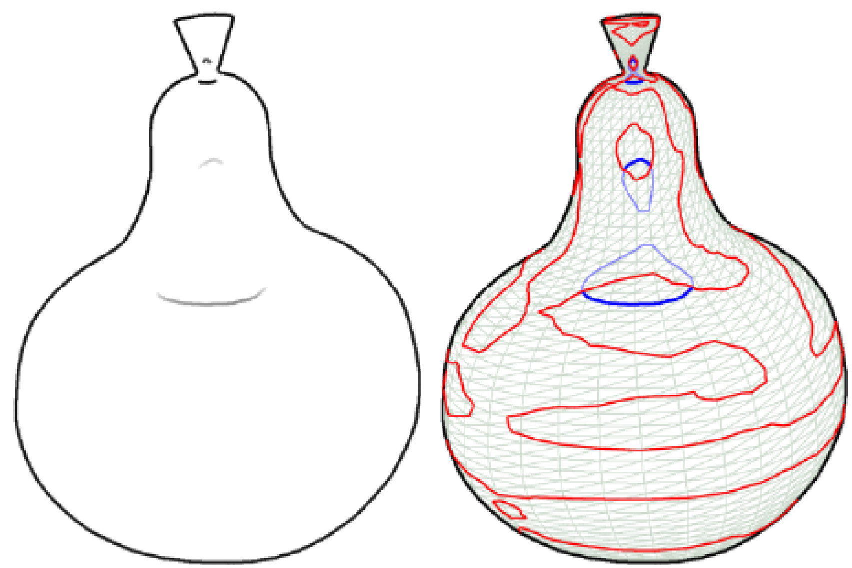


Figure 1: ***Left:*** *suggestive contours for line drawings* [DeCarlo et al. 2003] *are a recent example of a driving application for the estimation of curvatures and derivatives of curvature.* ***Right:*** *suggestive contours are drawn along the zeros of curvature in the view direction, shown here in blue, but only where the derivative of curvature in the view direction is positive (the curvature deriva-*

Second Fundamental Form Matrix

$$\mathrm{II} = \begin{pmatrix} D_u n & D_v n \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial n}{\partial u} \cdot \vec{u} & \frac{\partial n}{\partial v} \cdot \vec{u} \\ \frac{\partial n}{\partial u} \cdot \vec{v} & \frac{\partial n}{\partial v} \cdot \vec{v} \end{pmatrix}$$

Second Fundamental Form Matrix

$$\mathrm{I\!I} = \begin{pmatrix} D_u n & D_v n \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial n}{\partial u} \cdot \vec{u} & \frac{\partial n}{\partial v} \cdot \vec{u} \\ \frac{\partial n}{\partial u} \cdot \vec{v} & \frac{\partial n}{\partial v} \cdot \vec{v} \end{pmatrix}$$

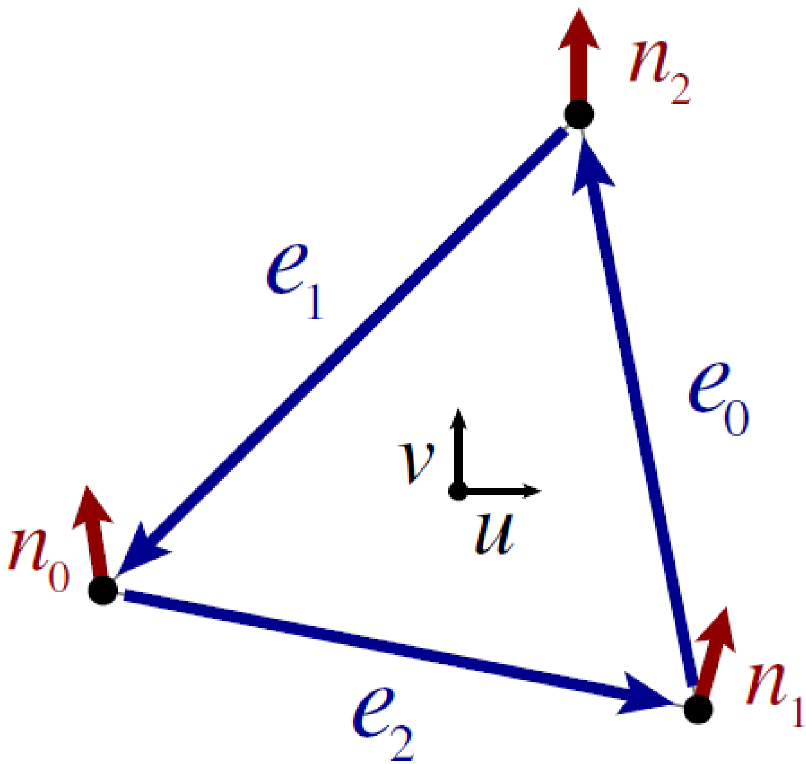
Assume *u*, *v* are orthogonal

Second Fundamental Form Matrix

$$s = c_1\vec{u} + c_2\vec{v}$$

$$\mathrm{II}\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = D_s n$$

Finite Difference Per-Face



$$\mathbf{II}\begin{pmatrix} e_0 \cdot u \\ e_0 \cdot v \end{pmatrix} = \begin{pmatrix} (n_2 - n_1) \cdot u \\ (n_2 - n_1) \cdot v \end{pmatrix}$$

$$\mathbf{II}\begin{pmatrix} e_1 \cdot u \\ e_1 \cdot v \end{pmatrix} = \begin{pmatrix} (n_0 - n_2) \cdot u \\ (n_0 - n_2) \cdot v \end{pmatrix}$$

$$\mathbf{II}\begin{pmatrix} e_2 \cdot u \\ e_2 \cdot v \end{pmatrix} = \begin{pmatrix} (n_1 - n_0) \cdot u \\ (n_1 - n_0) \cdot v \end{pmatrix}$$

Figure from the paper

Least-squares

Average for Per-Vertex

- Rotate tangent plane about cross product of normals
- Average using Voronoi weights

Completely Different Formula

Consistent Computation of First- and Second-Order Differential Quantities for Surface Meshes

Xiangmin Jiao*
Dept. of Applied Mathematics & Statistics
Stony Brook University

Hongyuan Zha†
College of Computing
Georgia Institute of Technology

Abstract

Differential quantities, including normals, curvatures, principal directions, and associated matrices, play a fundamental role in geometric processing and physics-based modeling. Computing these differential quantities *consistently* on surface meshes is important and challenging, and some existing methods often produce inconsistent results and require *ad hoc* fixes. In this paper, we show that the computation of the gradient and Hessian of a height function provides the foundation for consistently computing the differential quantities. We derive simple, *explicit* formulas for the transformations between the first- and second-order differential quantities (i.e., normal vector and curvature matrix) of a smooth surface and the first- and second-order derivatives (i.e., gradient and Hessian) of its corresponding height function. We then investigate a general, flexible numerical framework to estimate the derivatives of the height function based on local polynomial fittings formulated as weighted least squares approximations. We also propose an iterative fitting

often require *ad hoc* fixes to avoid crashing of the code, and their effects on the accuracy of the applications are difficult to analyze.

The ultimate goal of this work is to investigate a mathematically sound framework that can compute the differential quantities *consistently* (i.e., satisfying the intrinsic constraints) with provable *convergence* on general surface meshes, while being flexible and easy to implement. This is undoubtly an ambitious goal. Although we may have not fully achieved the goal, we make some contributions toward it. First, using the singular value decomposition [Golub and Van Loan 1996] of the Jacobian matrix, we derive explicit formulas for the transformations between the first- and second-order differential quantities of a smooth surface (i.e., normal vector and curvature matrix) and the first- and second-order derivatives of its corresponding height function (i.e., gradient and Hessian). We also give the explicit formulas for the transformations of the gradient and Hessian under a rotation of the coordinate system. These transformations can be obtained without forming the shape operator and the associated computation of its eigenvalues or eigenvectors. We

Completely Different Formula

Consistent Computation of First- and Second-Order Differential Quantities for Surface Meshes

Xiangmin Jiao*
Dept. of Applied Mathematics & Statistics
Stony Brook University

Hongyuan Zha†
College of Computing
Georgia Institute of Technology

Abstract

Differential quantities, including normals, curvatures, principal directions, and associated matrices, play a fundamental role in geo-

...

ible numerical framework to estimate the derivatives of the height function based on local polynomial fittings formulated as weighted least squares approximations. We also propose an iterative fitting

often require *ad hoc* fixes to avoid crashing of the code, and their effects on the accuracy of the applications are difficult to analyze.

The ultimate goal of this work is to investigate a mathematically

...

give the explicit formulas for the transformations of the gradient and Hessian under a rotation of the coordinate system. These transformations can be obtained without forming the shape operator and the associated computation of its eigenvalues or eigenvectors. We

Theorem 3 *The mean and Gaussian curvature of the height function $f(\boldsymbol{u}) : \mathbb{R}^2 \to \mathbb{R}$ are*

$$\kappa_H = \frac{tr(\boldsymbol{H})}{2\ell} - \frac{(\nabla f)^T \boldsymbol{H} (\nabla f)}{2\ell^3}, \text{ and } \kappa_G = \frac{det(\boldsymbol{H})}{\ell^4}. \quad (16)$$

Conserved Quantity Approach

Discrete Differential-Geometry Operators for Triangulated 2-Manifolds

Mark Meyer[1], Mathieu Desbrun[1,2], Peter Schröder[1], and Alan H. Barr[1]

[1] Caltech
[2] USC

Visualization and Math. !!!

Summary. This paper proposes a unified and consistent set of flexible tools to approximate important geometric attributes, including normal vectors and curvatures on arbitrary triangle meshes. We present a consistent derivation of these first and second order differential properties using *averaging Voronoi cells* and the mixed Finite-Element/Finite-Volume method, and compare them to existing formulations. Building upon previous work in discrete geometry, these operators are closely related to the continuous case, guaranteeing an appropriate extension from the continuous to the discrete setting: they respect most intrinsic properties of the continuous differential operators. We show that these estimates are optimal in accuracy under mild smoothness conditions, and demonstrate their numerical quality. We also present applications of these operators, such as mesh smoothing, enhancement, and quality checking, and show results of denoising in higher dimensions, such as for tensor images.

Recall:

Structure preservation

[**struhk**-cher pre-zur-**vey**-sh*uh*n]**:**

Keeping properties from the continuous abstraction exactly true in a discretization.

Gauss-Bonnet Theorem

$$\int_M K \, dA + \int_{\partial M} k_g \, ds = 2\pi\chi(M)$$

Gaussian curvature

Geodesic curvature (curvature projected on tangent plane)

2-2g

Gauss-Bonnet Theorem

$$\int_M K \, dA + \int_{\partial M} k_g \, ds = 2\pi\chi(M)$$

Gaussian curvature

2-2g

Geodesic curvature (curvature projected on tangent plane)

\omit{proof}

For Polygonal Cells

$$\int_V K \, dA = 2\pi - \sum_j \varepsilon_j$$

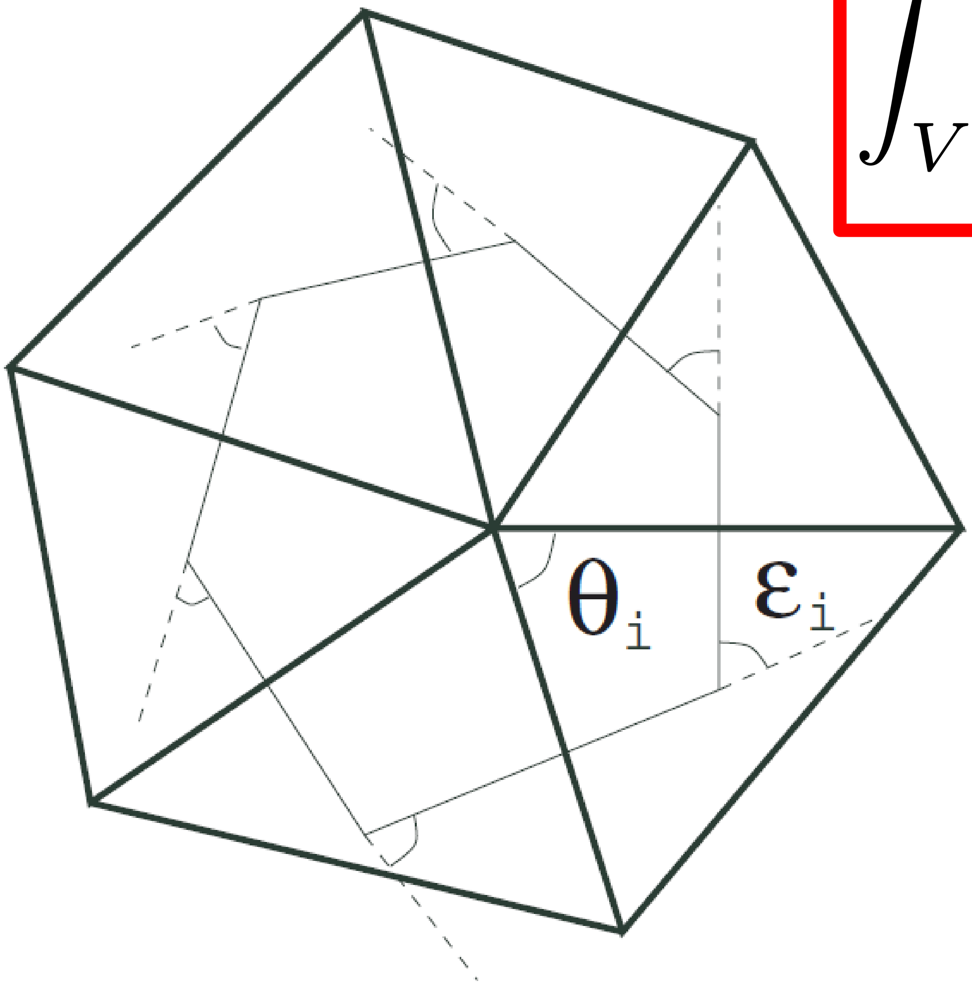
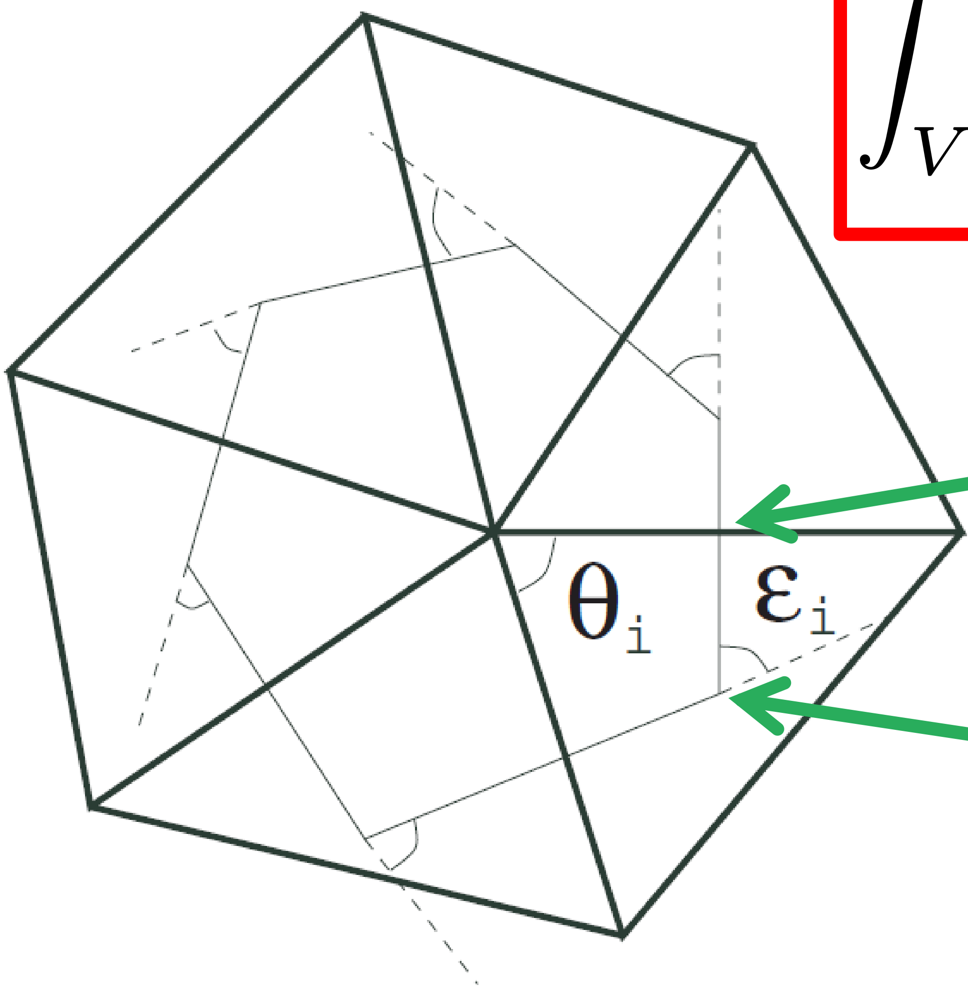


Figure from the paper

For Polygonal Cells

$$\int_V K \, dA = 2\pi - \sum_j \varepsilon_j$$

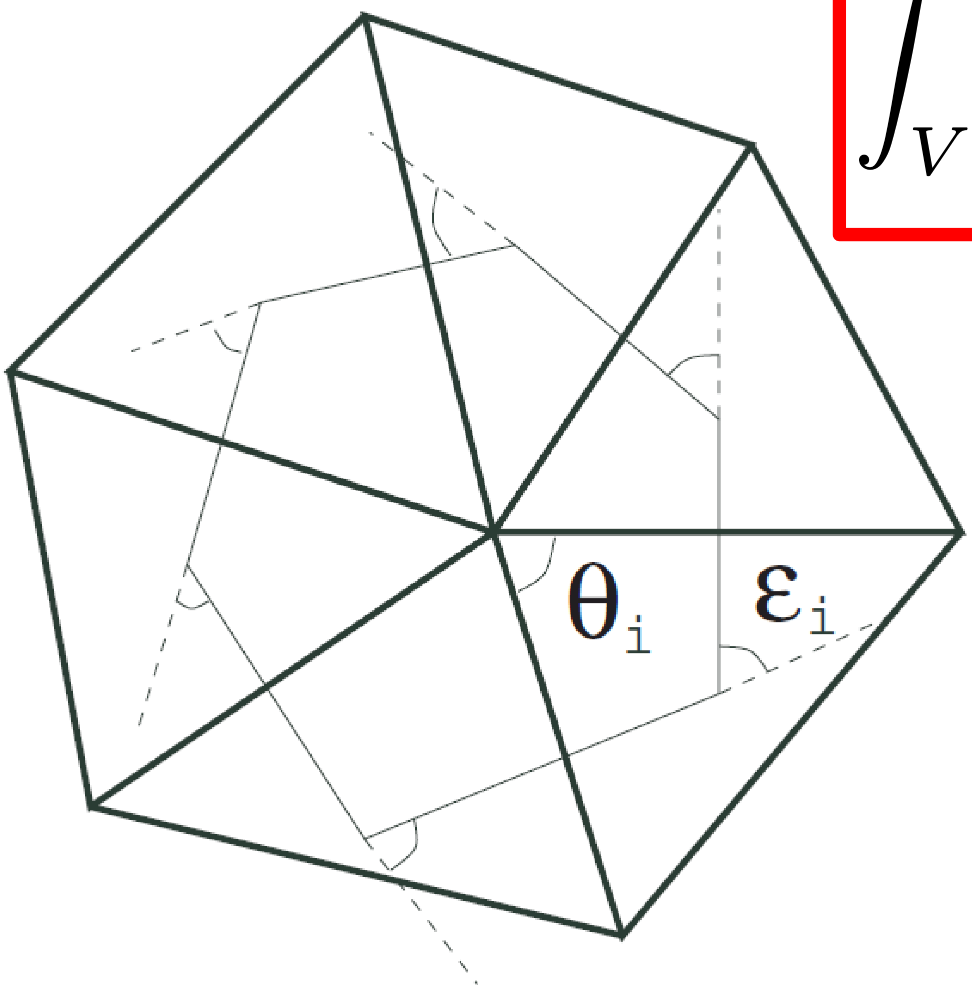


Change is in normal direction

Turning angle integrated curvature

Figure from the paper

For Voronoi Cells



$$\int_V K \, dA = 2\pi - \sum_j \theta_j$$

Homework!

Figure from the paper

Flip Things Backward

DEFINITION:

Gaussian curvature integrated over region *V* is given by

$$\int_V K \; dA = 2\pi - \sum_j \theta_j$$

Divide by area for curvature estimate

Recall: Euler Characteristic

$$V - E + F = \chi$$

$$\chi = 2 - 2g$$

$g = 0$

$g = 1$

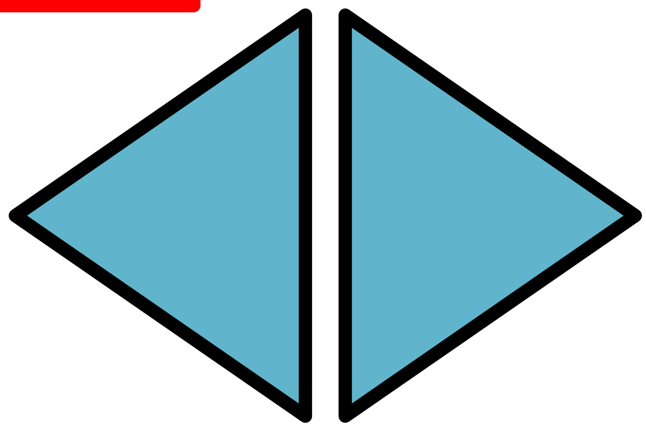
$g = 2$

Recall:

Consequences for Triangle Meshes

$$V - \frac{1}{2}F = \chi$$

"Each edge is adjacent to two faces. Each face has three edges."



$$2E = 3F$$

Closed mesh: Easy estimates!

Discrete Gauss-Bonnet

$$\int_M K \, dA = \sum_i \int_{V_i} K \, dA$$

Partition the surface

Discrete Gauss-Bonnet

$$\int_M K\ dA = \sum_i \int_{V_i} K\ dA$$

$$= \sum_i \left(2\pi - \sum_j \theta_{ij} \right)$$

Apply our definition

Discrete Gauss-Bonnet

$$\int_M K \, dA = \sum_i \int_{V_i} K \, dA$$

$$= \sum_i \left(2\pi - \sum_j \theta_{ij} \right)$$

$$= 2\pi V - \sum_{ij} \theta_{ij}$$

Pull out constants

Discrete Gauss-Bonnet

$$\int_M K\ dA = \sum_i \int_{V_i} K\ dA$$

$$= \sum_i \left(2\pi - \sum_j \theta_{ij} \right)$$

$$= 2\pi V - \sum_{ij} \theta_{ij}$$

$$= 2\pi V - \pi F$$

Consider sum over triangles

Discrete Gauss-Bonnet

$$\begin{aligned}
\int_M K\ dA &= \sum_i \int_{V_i} K\ dA \\
&= \sum_i \left(2\pi - \sum_j \theta_{ij} \right) \\
&= 2\pi V - \sum_{ij} \theta_{ij} \\
&= 2\pi V - \pi F \\
&= \pi(2V - F) \\
&= 2\pi\chi
\end{aligned}$$

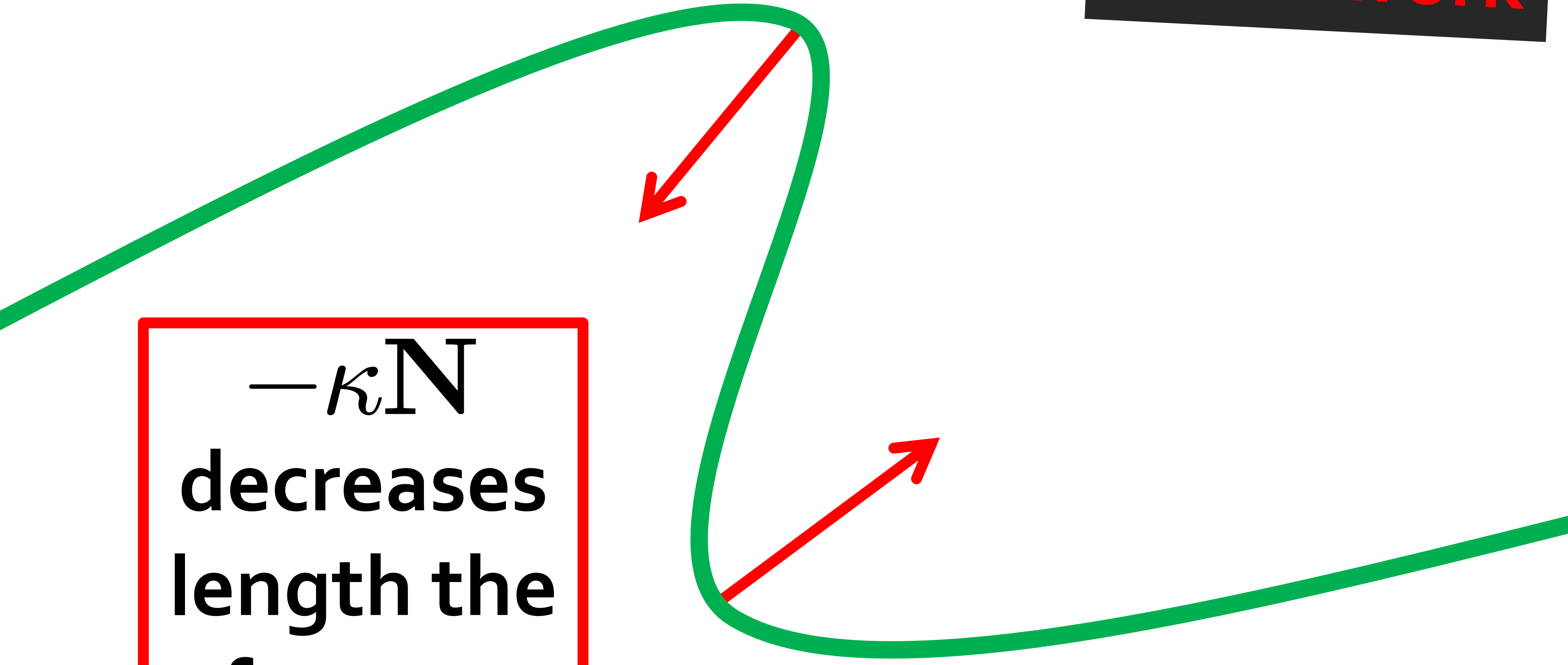
By definition

<qed/>

Recall:

Curvature Vector

Homework



$$-\kappa \mathbf{N}$$

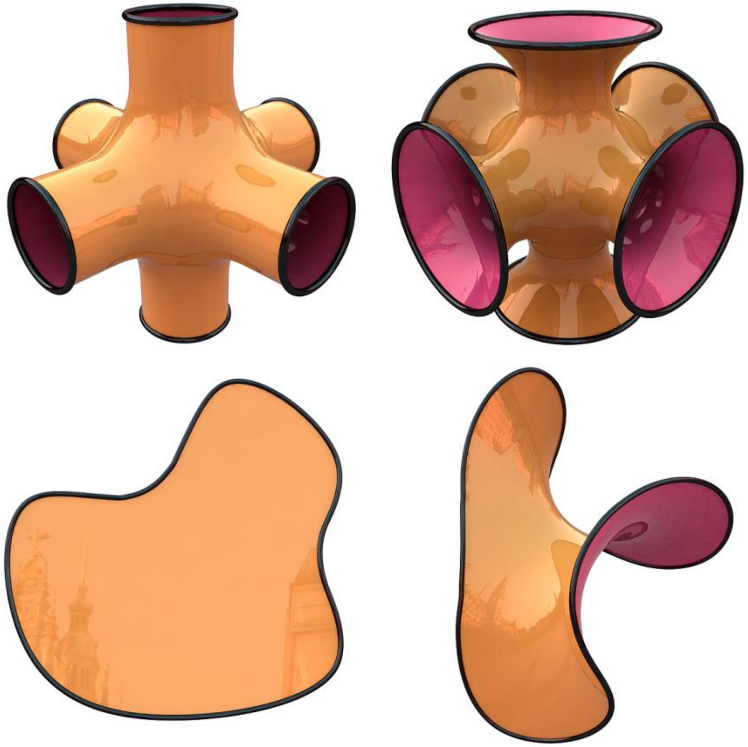
decreases length the fastest.

Foreshadow: Mean Curvature Normal

$$E(M) = \text{Area}(M)$$

$$\nabla E(p) = H\vec{n}$$

"Variational derivative"

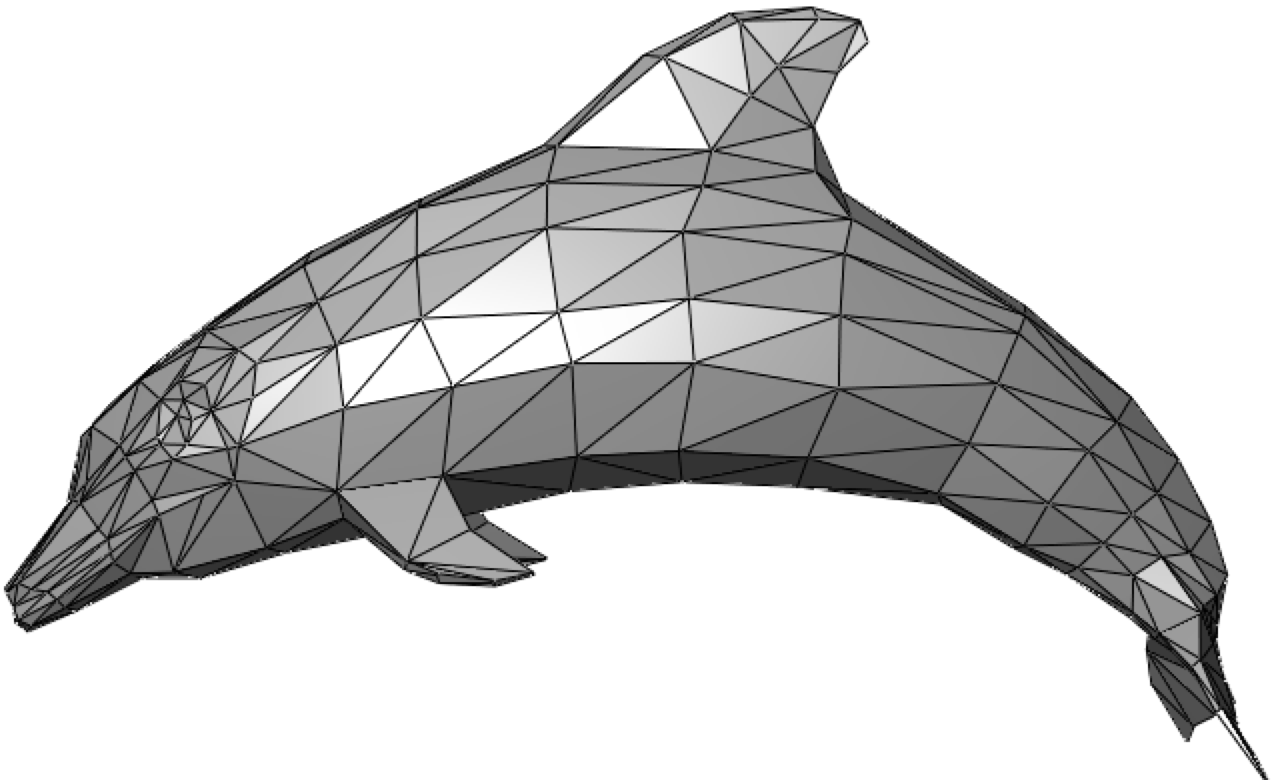


$$\nabla E(p) \equiv 0 \;\forall p \in \text{int } M$$

Minimal surfaces

http://multires.caltech.edu/~keenan/pdf/spinxform.pdf

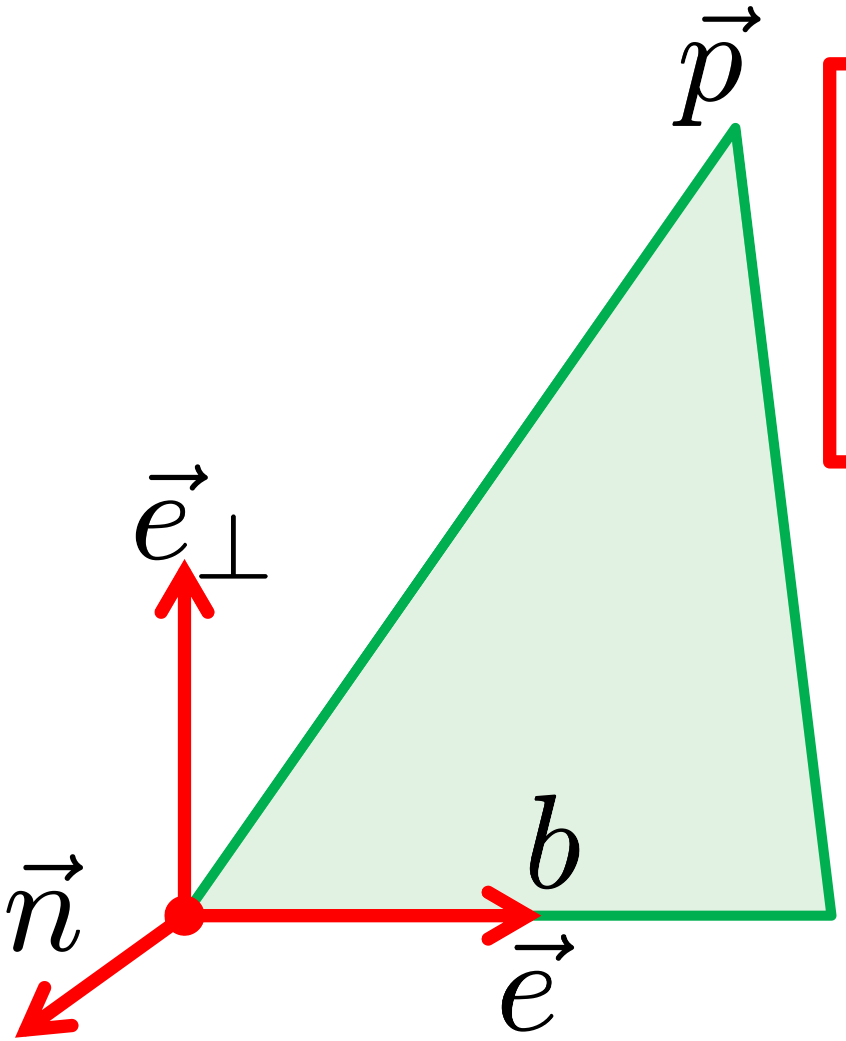
Area Functional for Meshes



$$\text{Area} : \mathbb{R}^{3V} \to \mathbb{R}$$

http://upload.wikimedia.org/wikipedia/commons/f/fb/Dolphin_triangle_mesh.png

Single Triangle



$$\vec{p} = p_n \vec{n} + p_e \vec{e} + p_\perp \vec{e}_\perp$$

$$A = \frac{1}{2} b \sqrt{p_n^2 + p_\perp^2}$$

Single Triangle: Derivatives

$$\vec{p} = p_n \vec{n} + p_e \vec{e} + p_\perp \vec{e}_\perp$$

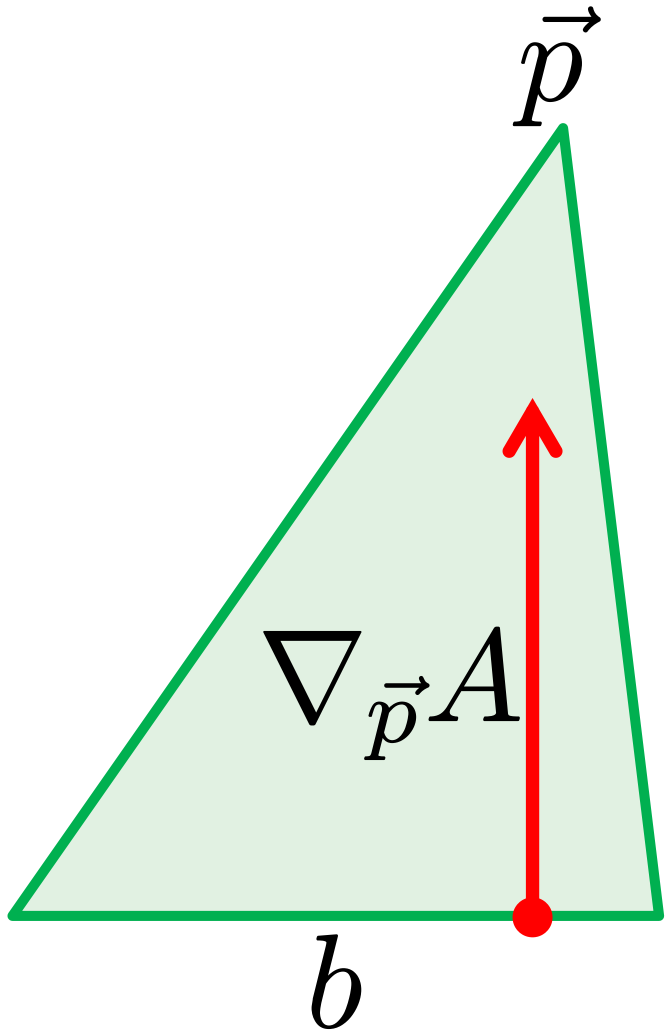
$$A = \frac{1}{2} b \sqrt{p_n^2 + p_\perp^2}$$

$$\frac{\partial A}{\partial p_e} = 0$$

$$\frac{\partial A}{\partial p_n} = \frac{b p_n \to 0}{2\sqrt{p_n^2 + p_\perp^2}} = 0 \Longrightarrow \nabla_{\vec{p}} A = \frac{1}{2} b \vec{e}_\perp$$

$$\frac{\partial A}{\partial p_\perp} = \frac{b p_\perp}{2\sqrt{p_n^2 + p_\perp^2}}$$

Single Triangle: Complete

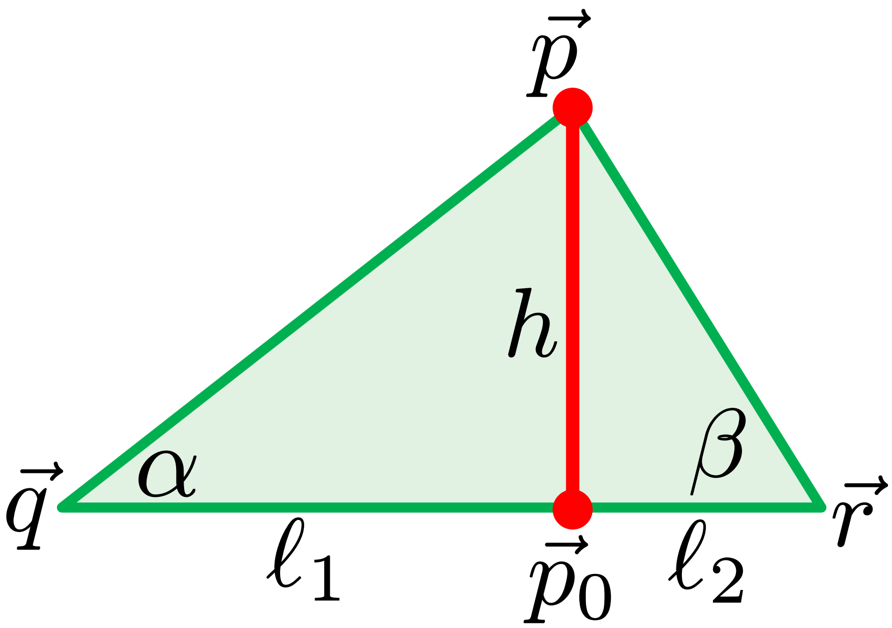


$$\vec{p} = p_n \vec{n} + p_e \vec{e} + p_\perp \vec{e}_\perp$$

$$A = \frac{1}{2} b \sqrt{p_n^2 + p_\perp^2}$$

$$\nabla_{\vec{p}} A = \frac{1}{2} b \vec{e}_\perp$$

Cotangent for Single Triangle



$$h = \ell_1 \tan\alpha = \ell_2 \tan\beta$$

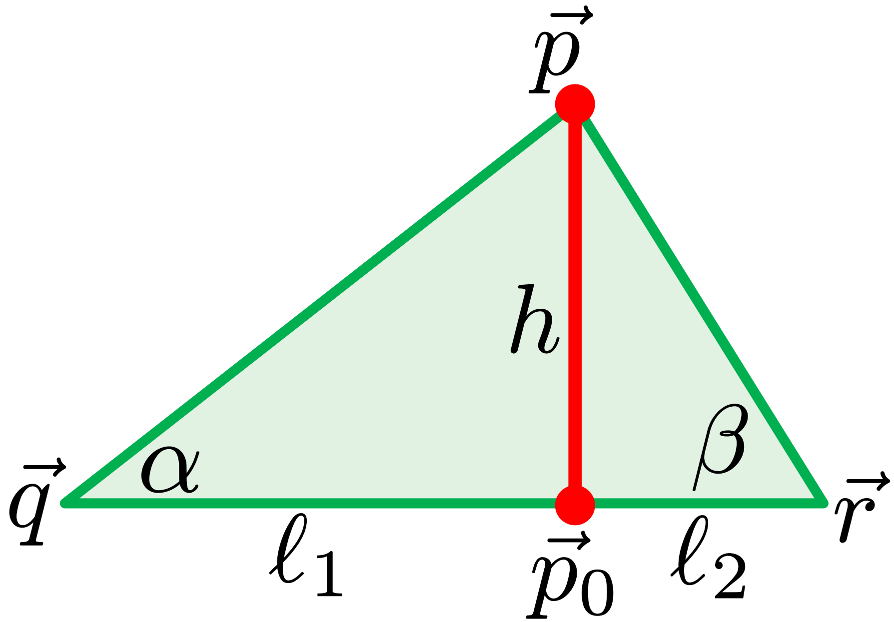
$$\vec{p}_0 = t\vec{r} + (1-t)\vec{q}$$

$$t = \frac{\ell_1}{\ell_1 + \ell_2}$$

$$= \frac{\ell_1}{\ell_1 + \ell_1 \frac{\tan\alpha}{\tan\beta}}$$

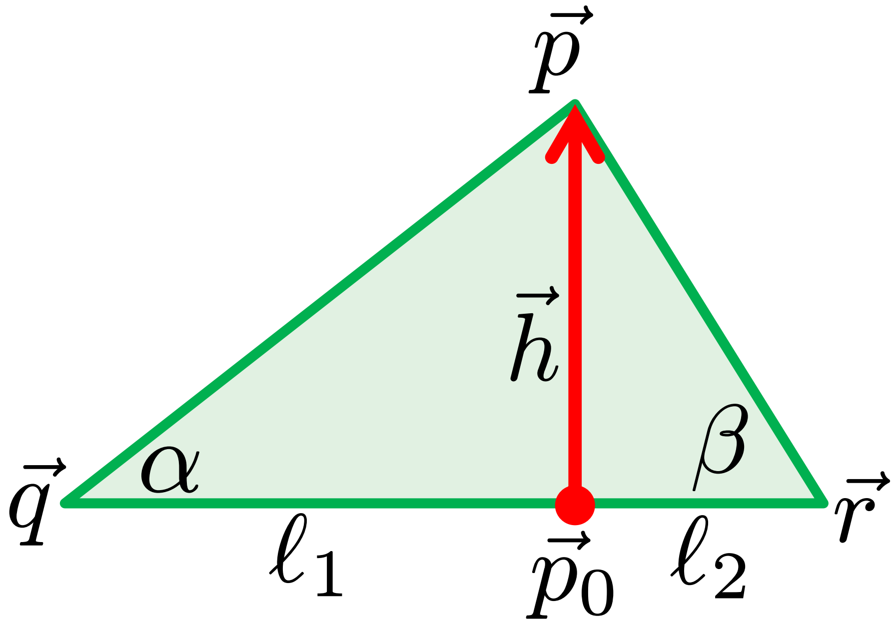
$$= \frac{\tan\beta}{\tan\alpha + \tan\beta}$$

Cotangent for Single Triangle



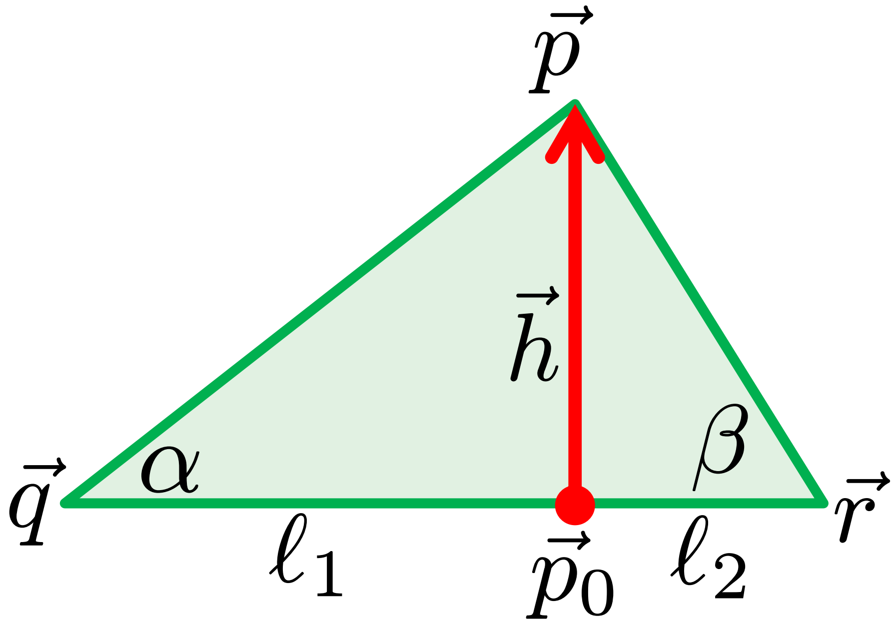
$$\vec{p}_0 = t\vec{r} + (1-t)\vec{q}$$
$$= \frac{1}{\tan\alpha + \tan\beta}(\vec{r}\tan\beta + \vec{q}\tan\alpha)$$

Cotangent for Single Triangle



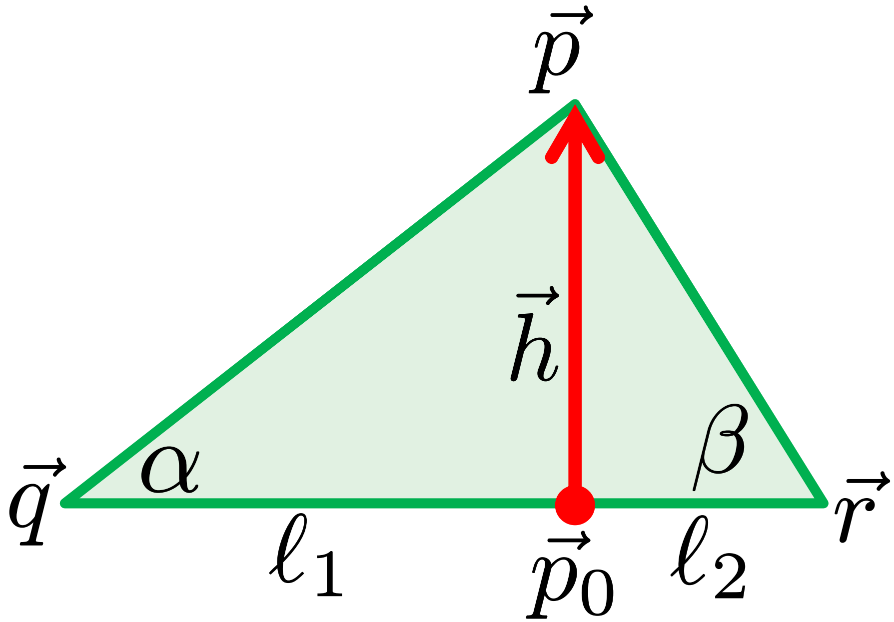
$$\vec{h} = \vec{p} - \vec{p}_0 = \frac{1}{\tan\alpha + \tan\beta}((\tan\alpha + \tan\beta)\vec{p} - \vec{r}\tan\beta - \vec{q}\tan\alpha)$$

Cotangent for Single Triangle



$$\frac{\ell_1 + \ell_2}{\|\vec{h}\|} = \frac{\ell_1 + \frac{\tan\alpha}{\tan\beta}\ell_1}{\ell_1 \tan\alpha} = \frac{\tan\alpha + \tan\beta}{\tan\alpha\tan\beta}$$

Cotangent for Single Triangle



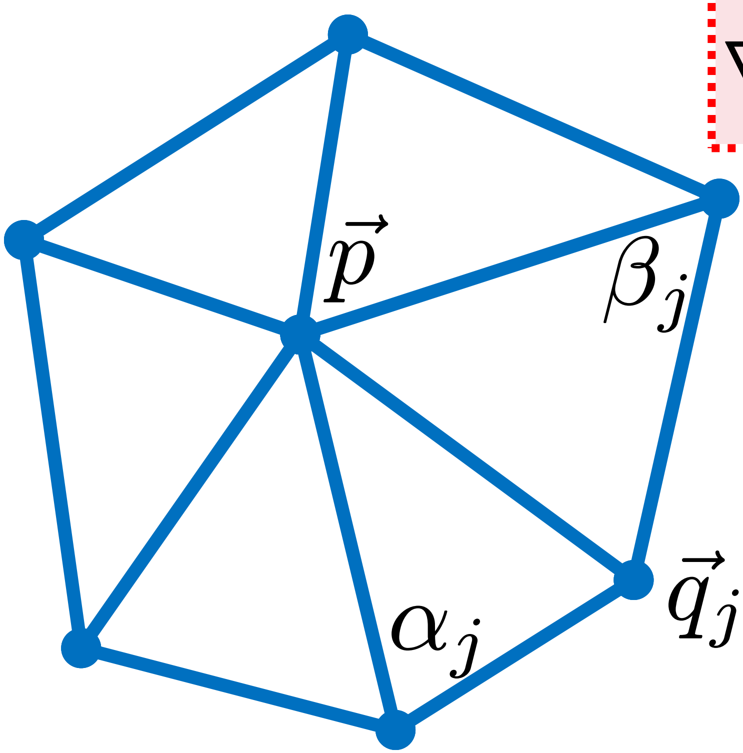
$$\nabla_{\vec{p}} A = \frac{1}{2} \cdot \frac{\ell_1 + \ell_2}{\|\vec{h}\|} \cdot \vec{h}$$

$$= \frac{1}{2}((\vec{p} - \vec{r}) \cot \alpha + (\vec{p} - \vec{q}) \cot \beta)$$

Summing Around a Vertex

$$\nabla_{\vec{p}} A = \frac{1}{2} \sum_j (\cot \alpha_j + \cot \beta_j)(\vec{p} - \vec{q}_j)$$

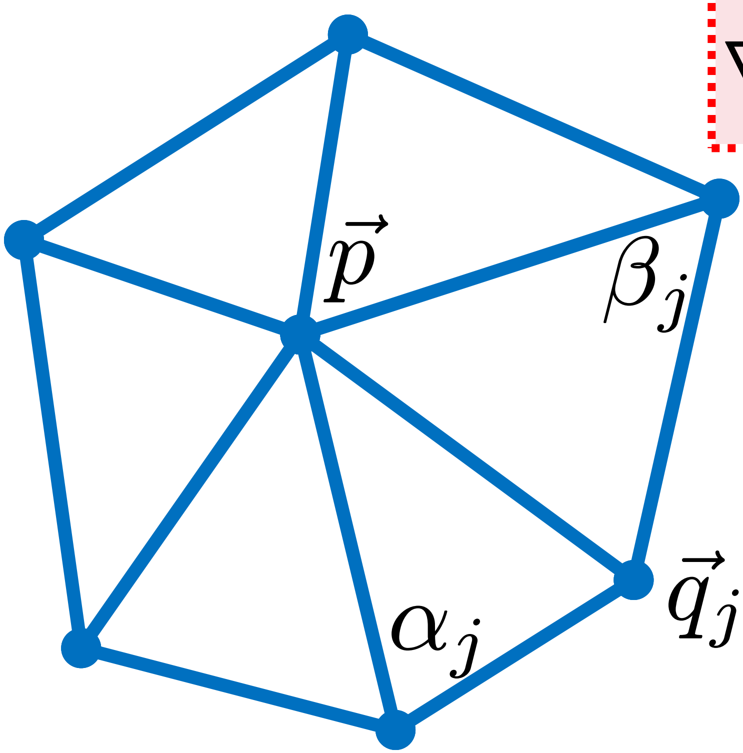
$$\nabla_{\vec{p}} A = \frac{1}{2}((\vec{p} - \vec{r}) \cot \alpha + (\vec{p} - \vec{q}) \cot \beta)$$



Summing Around a Vertex

$$\nabla_{\vec{p}} A = \frac{1}{2} \sum_j (\cot \alpha_j + \cot \beta_j)(\vec{p} - \vec{q}_j)$$

$$\nabla_{\vec{p}} A = \frac{1}{2}((\vec{p} - \vec{r}) \cot \alpha + (\vec{p} - \vec{q}) \cot \beta)$$



Vanishes as you refine the mesh

Integrated Mean Curvature Normal

DEFINITION:

The mean curvature normal integrated over region *V* is given by

$$\int_V H\vec{n}\, dA = \frac{1}{2}\sum_j (\cot\alpha_j + \cot\beta_j)(\vec{p} - \vec{q}_j)$$

Divide by area for curvature estimate

Pipeline

- Compute integrated *H*, *K*
- Divide by **area of cell** for estimated value

To Obtain Principal Directions

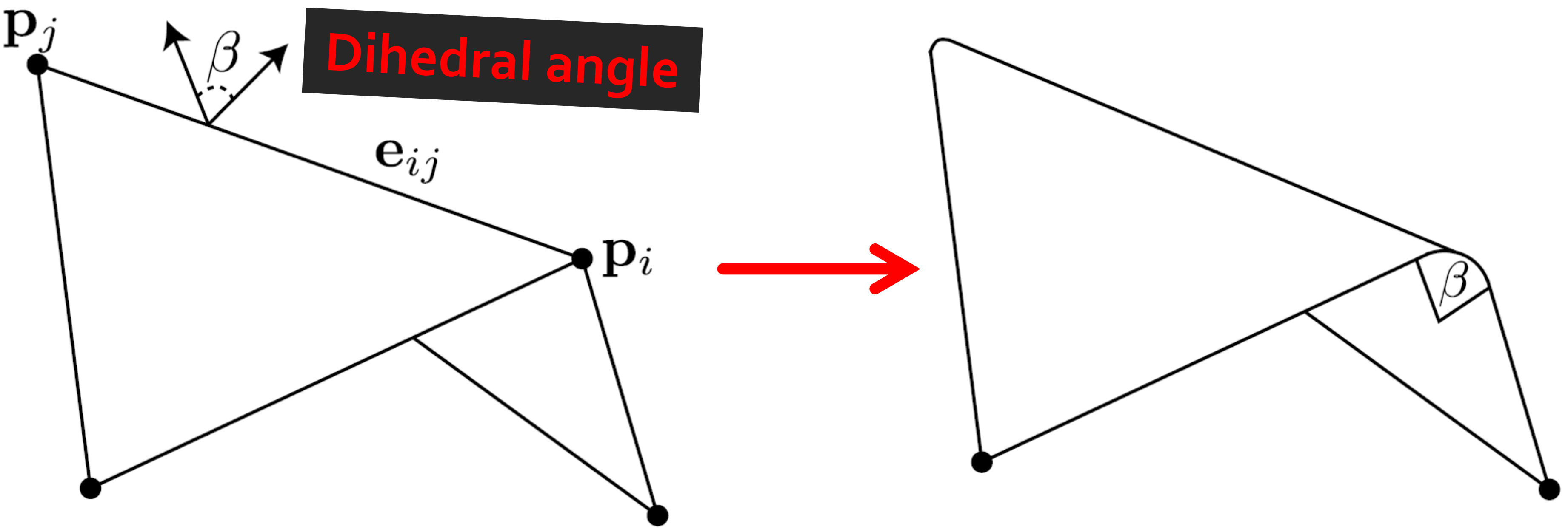
Taubin's method with different weights.

See paper.

$$H = \frac{1}{2}(\kappa_1 + \kappa_2)$$

$$K = \kappa_1 \kappa_2$$

Another Mean Curvature

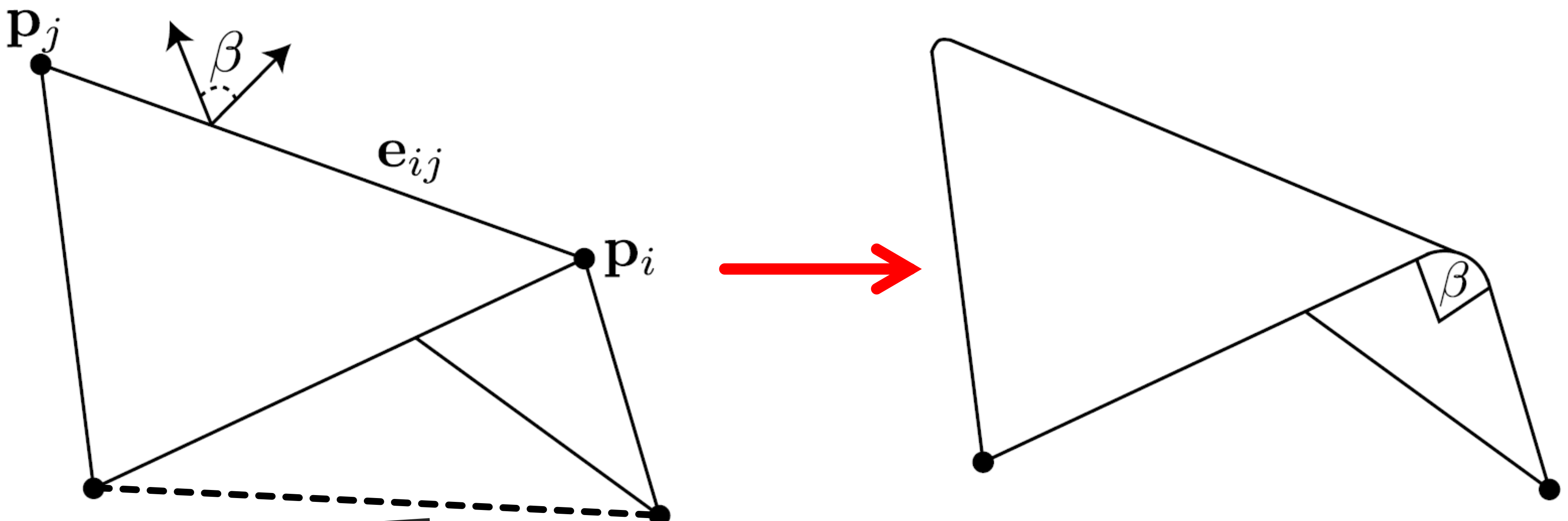


$$\int_B H = \frac{1}{2}\beta\|\mathbf{e}\|$$

J.A. Bærentzen et al., *Guide to Computational Geometry Processing* (2012)

Used for triangulation applications

Another Mean Curvature



Same vertices, lower curvature

$$\int_B H = \frac{1}{2}\beta\|\mathbf{e}\|$$

J.A. Bærentzen et al., *Guide to Computational Geometry Processing* (2012)

Used for triangulation applications

Tuned for Variational Applications

Computing discrete shape operators on general meshes

Eitan Grinspun
Columbia University
eitan@cs.columbia.edu

Yotam Gingold
New York University
gingold@mrl.nyu.edu

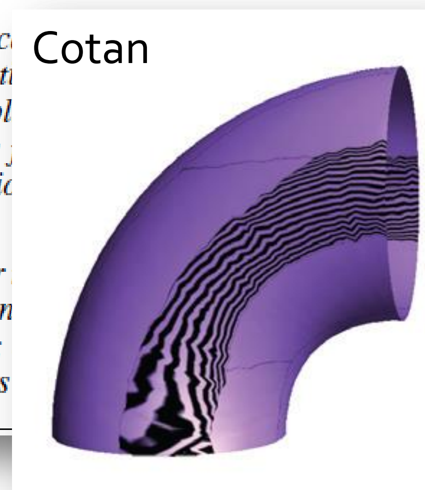
Jason Reisman
New York University
jasonr@mrl.nyu.edu

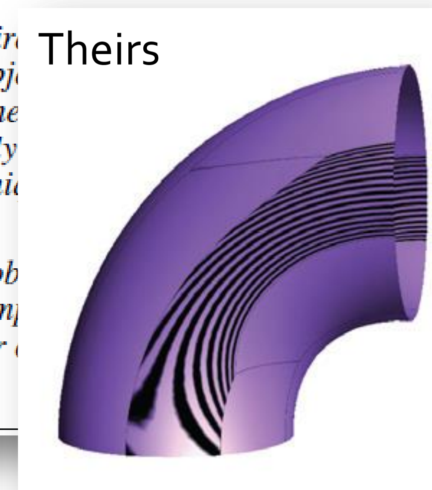
Denis Zorin
New York University
dzorin@mrl.nyu.edu

Abstract

Discrete curvature and shape operators, which c… out dir… are essential in a variety of applications: simulat… nal obj… geometric data processing. In many of these appl… l by me… approaches for formulating curvature operators … highly expensive methods used in engineering applicati… techni… computer graphics.

We propose a simple and efficient formulation for … al prob… degrees of freedom associated with normals. On … its sim… curvature operators commonly used in graphics; … umber … and produces consistent results for different types …





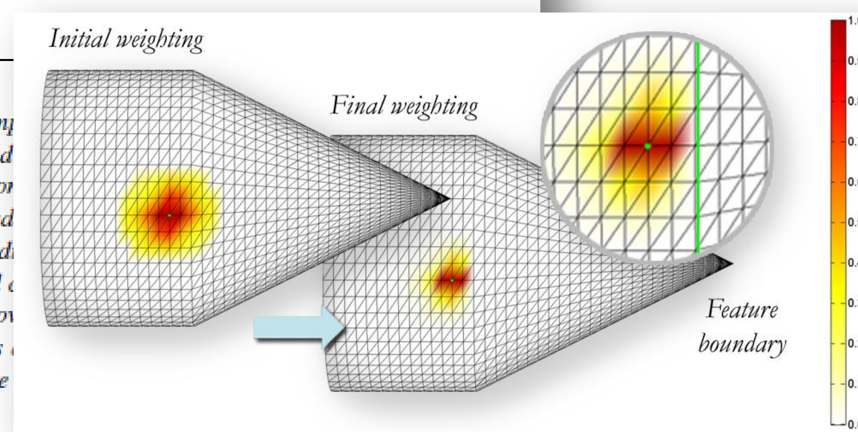
Tuned for Robustness

Eurographics Symposium on Geometry Processing (2007)
Alexander Belyaev, Michael Garland (Editors)

Robust statistical estimation of curvature on discretized surfaces

Evangelos Kalogerakis, Patricio Simari, Derek Nowrouzezahrai and Karan Singh

Dynamic Graphics Project, Computer Science Department, University of Toronto



Abstract

A robust statistics approach to curvature estimation on discretely samp point clouds, is presented. The method exhibits accuracy, stability and sampled surfaces with irregular configurations. Within an M-estimatio noise and structured outliers by sampling normal variations in an ad each point. The algorithm can be used to reliably derive higher order d surface normals while preserving the fine features of the normal and with state-of-the-art curvature estimation methods and shown to impro across ground truth test surfaces under varying tessellation densities noise. Finally, the benefits of a robust statistical estimation of curvature applications of mesh segmentation and suggestive contour rendering.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computational Geometry and Object Modeling]: Geometric algorithms, languages, and systems; curve, surface, solid, and object representations.

Alternative Strategies

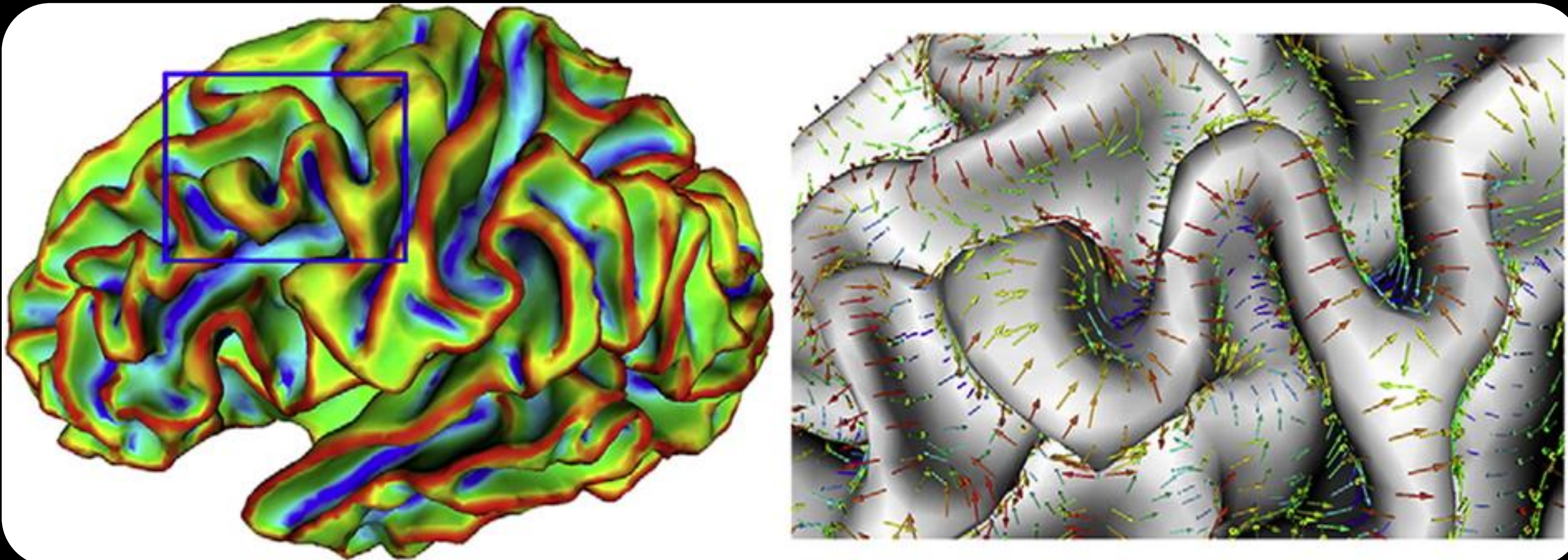
- **Locally fit** a smooth surface
  What type of surface? How to fit?
- Different **formula**
  Function of curvature? Where on mesh?
  Convergence of approximation?
- **Learn** curvature computation
  Tune for application? Training data?

Practical Advice

Try as many as you can.

Most are easy to implement!

http://www.sciencedirect.com/science/article/pii/S1053811909002663



Computing Curvature

CS 468, Spring 2013
Differential Geometry for Computer Science

Justin Solomon and Adrian Butscher