

Stat 992: Lecture 25

Curve modeling I.

Moo K. Chung mchung@stat.wisc.edu

March 29, 2004

1. *Curve data.* Both curve and surface modelling follows the same mathematical framework of spatially correlated noise with the *geometrically* constructed covariance function. See Chung *et al.* (2003) NeuroImage paper to find out the major issues in the closed surface modelling. For closed curve data see Hoffmann *et al.* (2004). This lecture gives estimation part of the solution to **Problem 25**.

Let $\mathbf{p}_i = (x_i, y_i)$ be the i -th pixel of a corpus callosum $\partial\Omega$ segmented by the level set algorithm (Read Sethian and Osher). The level set segmentation gives the binary representation of the boundary of the corpus callosum and all pixels are connected by either vertically, horizontally or diagonally. We assume that there are m pixels and indexed in clockwise direction such that $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are neighboring pixels. Then $(\mathbf{p}_1, \dots, \mathbf{p}_m)$ form a discrete chain of a closed curve topologically equivalent to a circle (Figure 1). In Figure 1, the gap in the rostrum of CC indicates starting point $\mathbf{p}_1$ and ending point $\mathbf{p}_m$.

One of the major problem is the curvature estimation that has been used as a morphometric measure (Hoffmann *et al.*, 2004). There are many different ways to estimate the curvatures. Let us review some basic curve geometry.

2. *Parameterization.* The simplest way to parameterize $\partial\Omega$ is to map the whole closed discrete chain to a unit circle S^1 such that

$$\mathbf{f} : \mathbf{p}_i \rightarrow \left(\cos \frac{2\pi i}{m}, \sin \frac{2\pi i}{m} \right).$$

This mapping maps m pixels that define the corpus callosum boundary to m equal spaced points in a unit circle. Let us denote $\mathcal{K}\mathbf{f}$, a smooth mapping of $\mathbf{f}$ such that the inverse map $(\mathcal{K}\mathbf{f})^{-1}$ is well defined. There are many ways to smooth the parameterization. Use either kernel or spline smoothing. Define a new mapping $\mathbf{g}$ such that

$$\mathbf{g} : \theta \rightarrow (\cos \theta, \sin \theta).$$

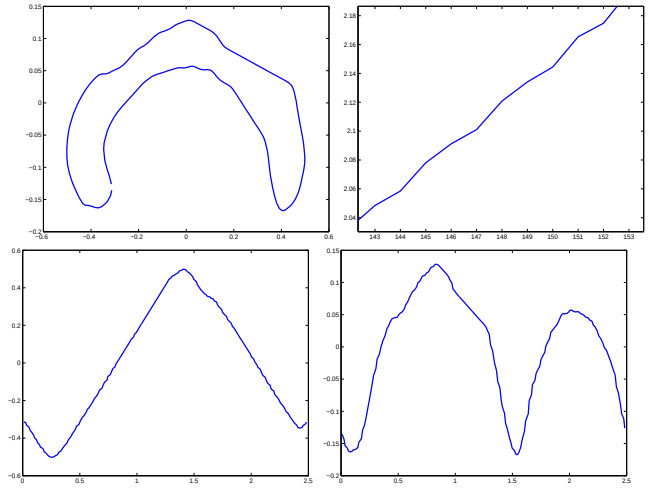


Figure 1: Top left: normalized corpus callosum closed boundary curve based on $m = 171$ points. Top right: arc length function λ given as a function of the vertex index. Bottom left: arc length parameterization of x -component. Bottom right: arc length parameterization of y -component.

Then the parameterization of $\partial\Omega$ by parameter $\theta \in [0, 2\pi)$ is given by

$$\mathbf{X}(\theta) = (\mathcal{K}\mathbf{f})^{-1} \circ \mathbf{g}(\theta).$$

The arc-length element is given by $(\mathbf{X}' \cdot \mathbf{X}')^{1/2}$. It measures the length of the corpus callosum boundary with respect to the unit length in parameter space $[0, 2\pi)$. Note that the arc length of the corpus callosum $\partial\Omega$ is given by

$$\lambda(\theta) = \int_0^\theta (\mathbf{X}' \cdot \mathbf{X}')^{1/2} d\theta$$

and the total arc length is $\lambda(2\pi)$. It will be used as a global morphometric just like the total surface area has been used as a global morphometric in Chung *et al.* (2004). Another important local metric other than the arc length element is the curvature.

The curvature is given by

$$\kappa = \frac{|\mathbf{X}' \times \mathbf{X}''|}{|\mathbf{X}'|^3}. \quad (1)$$

Note that it will not be invariant under parameterization. To make it invariant, it is necessary to have arc length parameterization, i.e. $\mathbf{X}(\lambda)$ where $\lambda \in [0, \lambda(2\pi))$. In this case, equation (1) simplifies to $\kappa = \sqrt{\mathbf{X}'' \cdot \mathbf{X}''}$.

This measures the amount of bending of the corpus callosum boundary. Usually this curvature is positive but we can give signs to indicate the convexity. However this additional information is redundant since curvature function κ can completely characterize the original curve itself. The curvature will be positive if the corpus callosum boundary is locally concave and negative if convex. In this way this curvature function can give more information about the geometry of the corpus callosum boundary. The global measure of total bending can be measured by computing so called *bending energy*

$$K = \int_0^{2\pi} k^2(\theta) d\theta.$$

Similar approach has been taken in measuring the amount of bending of white fibers in diffusion tensor images (Poupon *et al.*, 2000). In Chung *et al.* (2003), the thin-plate spline energy functional which is the sum of the squared principal curvatures of the cortex is used for measuring the amount of surface bending. For the detailed treatment of the geometry of curves and surfaces, read Kreyszig, Differential Geometry (1964).

3. *Curvature estimation* We show how to estimate it without the use of the parametric representation $\mathbf{X}$. By definition the curvature at point $\mathbf{p}_i \in \partial\Omega$ is the reciprocal of the radius of the largest circle that touches $\mathbf{p}_i$ without crossing the boundary $\partial\Omega$. Three-points estimation uses the fact that a unique circle can pass through given three non collinear points. For vertices $\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}$, the curvature is given by

$$\kappa = \frac{4A(\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1})}{|\mathbf{p}_{i-1} - \mathbf{p}_i| |\mathbf{p}_i - \mathbf{p}_{i+1}| |\mathbf{p}_{i+1} - \mathbf{p}_{i-1}|}$$

where $A(\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1})$ is the area of triangle with vertices $\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}$ (Shubing Wang). However this estimation is based on very constrained geometric shape and will cause a lot of singularities.

Due the partial volume effects, we relax the definition a little bit and estimate a circle that pass through neighboring vertices around $\mathbf{p}_i = (x_i, y_i)$ in the least squares sense. For the vertices $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{p}_{i+2}\}$, we solve the equation of a circle

$$(x - a)^2 + (y - b)^2 = (x_i - a)^2 + (y_i - b)^2.$$

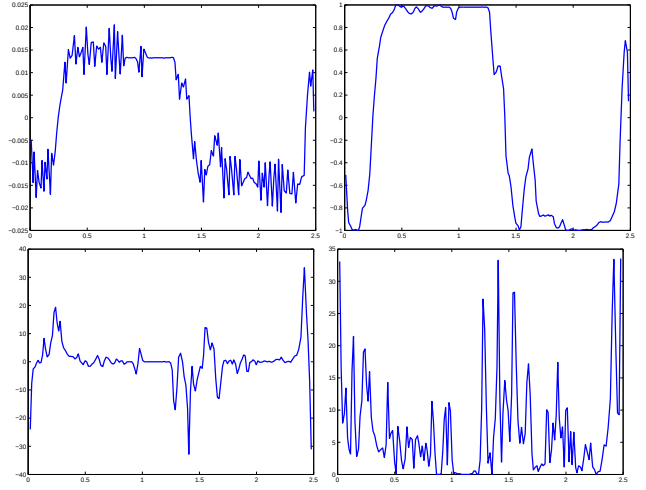


Figure 2: Top left: The horizontal component of $\mathbf{X}'$ via the finite difference without arclength parameterization. Top right: The horizontal component of $\mathbf{X}'$ via the finite difference with arclength parameterization. Bottom left: The horizontal component of $\mathbf{X}''$ via arclength parameterization. Bottom right: Curvature function $\kappa(\lambda)$. Shubing's method gives the almost identical result.

Simplifying the equation, we get

$$2a(x_i - x) + 2b(y_i - y) = x_i^2 - x^2 + y_i^2 - y^2.$$

Substituting the coordinates for the neighboring vertices into the above equation, we get a matrix equation of the form $AB = C$:

$$2 \begin{pmatrix} x_i - x_{i-2} & y_i - y_{i-2} \\ x_i - x_{i-1} & y_i - y_{i-1} \\ x_i - x_{i+1} & y_i - y_{i+1} \\ x_i - x_{i+2} & y_i - y_{i+2} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} x_i^2 - x_{i-2}^2 + y_i^2 - y_{i-2}^2 \\ x_i^2 - x_{i-1}^2 + y_i^2 - y_{i-1}^2 \\ x_i^2 - x_{i+1}^2 + y_i^2 - y_{i+1}^2 \\ x_i^2 - x_{i+2}^2 + y_i^2 - y_{i+2}^2 \end{pmatrix}$$

The least squares estimations of a and b are given by the generalized inverse $\tilde{B} = (A'A)^{-1}A'C$. Then the curvature κ at each vertex $\mathbf{p}_i$ is

$$\kappa(\mathbf{p}_i) = \left((x_i - \tilde{a})^2 + (y_i - \tilde{b})^2 \right)^{-1/2}.$$

A slightly different method is to do the least squares fit of polynomial $y = \alpha_0 + \alpha_1 x + \alpha_2 x^2$ locally for horizontal curve sections and $x = \alpha_0 + \alpha_1 y + \alpha_2 y^2$ for vertical curve sections on 5 vertices (Tulaya Limpiti).

4. *Curvature estimation via acrlength parameterization*. This is somewhat trivial. The first derivatives $\mathbf{X}'(\lambda)$ are estimated via the finite difference and the second derivative follows similarly. Finite difference for non regular grid is

$$f'(\lambda) = \frac{f(\lambda + \Delta\lambda_1) - f(\lambda)}{\Delta\lambda_1}$$

and

$$f''(\lambda) = \frac{\frac{f(\lambda+\Delta\lambda_1)-f(\lambda)}{\Delta\lambda_1} - \frac{f(\lambda)-f(\lambda-\Delta\lambda_2)}{\Delta\lambda_2}}{\frac{\Delta\lambda_1+\Delta\lambda_2}{2}}$$

```
load CCcurve.data
CCcurve=[CCcurve;CCcurve(1,:)];
x=CCcurve(:,1);
y=CCcurve(:,2);

dx=diff(x); dy=diff(y);
dl=sqrt(dx.^2 + dy.^2);

%arclength function
for i=1:171
    l(i)=sum(dl(1:i))
end;

%fist difference
Dx=dx./dl(1:171);
figure;plot(l,Dx,'linewidth',2);
Dy=dy./dl(1:171);

%second difference
DDL= (dl(1:170)+dl(2:171))/2;
DDx = diff(Dx)./DDL;
DDy = diff(Dy)./DDL;
L= (l(1:170)+l(2:171))/2;

kappa=sqrt(DDx.^2 + DDy.^2)
figure;plot(L,kappa,'linewidth',2);
```

Problem 38. For triangular mesh data of the brain, first estimate the Riemannian metric tensor g via the parameterization to unit sphere S^2 . The principal curvatures are defined as the eigenvalues of g . Estimate the principal curvatures. Read lecture 6 MATLAB code for reading data. Hint: There are a couple of ways of doing this. One suggestion, if you have no clue, is to fit local polynomials $z = \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 xy + \alpha_4 x^2 + \alpha_5 y^3$. In two of the previous lectures, Riemannian metric tensors are defined.

Homework 5. Solve problems 33-40 and any other previous problems where complete solutions are not given. Due April 14 Wednesday 11:00am.