◆ Human Brain Mapping 6:329–333(1998) ◆

Curve Matching on Brain Surfaces Using Frenet Distances

M. Bakırcioğlu,[1] U. Grenander,[2] N. Khaneja,[3] and M.I. Miller[1,4]*

[1]*Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, Maryland*
[2]*Division of Applied Mathematics, Brown University, Providence, Rhode Island*
[3]*Division of Applied Science, Harvard University, Cambridge, Massachusetts*
[4]*Department of Biomedical Engineering, The Johns Hopkins University, Baltimore, Maryland*

Abstract: This paper describes methods for diffeomorphic matching of curves on brain surfaces. Distances between curves are defined by Frenet representation via speed, curvature, and torsion. The curve-matching algorithm is based on bipartite graph matching, with weights defined by the Frenet distance over diffeomorphic maps of one curve onto the other (Sedgewick [1983]: Algorithms). We follow Khaneja ([1996]: Statistics and Geometry of Cortical Features) and define fundus curves on the brain surfaces as extremal curvature lines generated using dynamic programming. Examples are shown for fundus curve matchings on macaque brain surfaces. *Hum. Brain Mapping 6:329–333, 1998.*

Key words: computational anatomy; image matching; deformable templates; computer vision; medical imaging

INTRODUCTION

The most striking gross morphological features of the cerebral hemisphere in mammals are the diverse and complex arrangement of the sulcal fissures and gyral prominences visible on the cortical surface of a mammalian brain. Despite their anatomic and functional significance, even the gyri and sulci that consistently appear in all normal anatomies exhibit pronounced variability in size and configuration [Welker, 1990]. Methods are beginning to appear for characterizing their variation [Thompson et al., 1996]. A quantitative study of these anatomical features requires developing mathematical models for characterizing their shape and geometry, and for accommodating the variability present across an anatomic population. Anatomical features such as sulci and gyri are being defined precisely in terms of the geometrical properties of the cortical surface, using the notions of ridge curves and crest lines corresponding to extreme points of curvature. From two-dimensional surface representations of the neocortex [Joshi et al., 1995a], automated algorithms now exist for generating such geometric features automatically. Such features as gyral crowns and sulcal curves define natural high-dimensional landmark correspondences between anatomies. They have been proposed by our group as defining landmark correspondences which are used in a heirarchical mapping procedure, bringing brain anatomy into register first at a coarse level defined through the dimensions of the landmarks and successively refined by the high-dimensional information provided by the images themselves [Joshi et al., 1995b; Miller et al., 1997]. To fully automate such a procedure, automated methods

Contract grant sponsor: NSF; Contract grant numbers: BIR 9424264, R01 NS35368-02; Contract grant sponsor: NIH; Contract grant number: 50567.

*Correspondence to: M.I. Miller, Department of Biomedical Engineering, The Johns Hopkins University, 3400 N. Charles Street, Baltimore, MD 21218. E-mail:mim@cis.jhu.edu

Received for publication 17 February 1998; accepted 17 June 1998

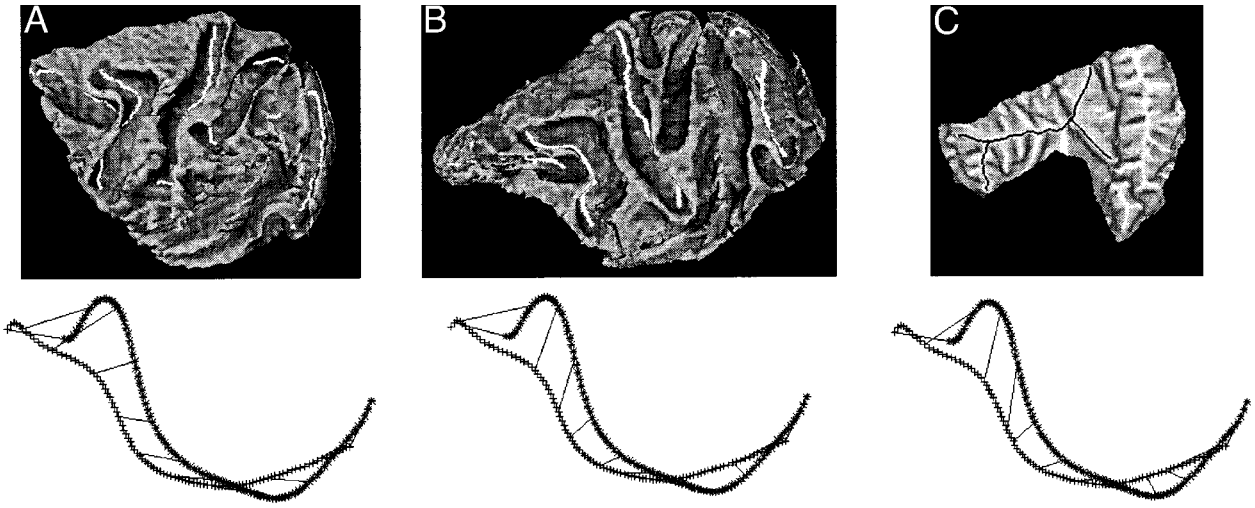

Figure 1.

A,B: Two macaque brains (87A and 93I) taken from David Van Essen's laboratory with the fundus curves generated using dynamic programming. [Khaneja, 1996] **C:** The bifuracating sylvian fissure in the Visible Human, illustrating how the start and end points control the solution of dynamic programming. **Bottom:** Superior temporal sulcus of the target macaque brain (87A) matched to the superior temporal sulcus of the template brain (93I) based on speed (**A**), curvature (**B**), and torsion (**C**).

for matching such curves must be established. This paper describes methods for matching curves on brain surfaces. Distances between curves are defined by Frenet representation via speed, curvature, and torsion. The particular matching algorithm used is bipartite matching [Sedgewick, 1983]. We follow Khaneja [1996] to define fundus curves on the brain surfaces as extremal curvature lines (crest lines) generated using dynamic programming. These fundus curves are then matched across brain surfaces. The results are presented in Figure 1.

MATCHING CURVES ACROSS DIFFERENT BRAINS

Given a curve $\alpha(t)$, $t \in [0, 1]$, the Frenet equations provide the differential geometric characterization of α in three dimensions (3-D) based on its speed, curvature, torsion, and the orthogonal frame (T, N, B):

$$\begin{bmatrix} \frac{dT(t)}{dt} \\ \frac{dN(t)}{dt} \\ \frac{dB(t)}{dt} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \nu\kappa(t) & 0 \\ -\nu\kappa(t) & 0 & \nu\kappa(t) \\ 0 & -\nu\tau(t) & 0 \end{bmatrix}}_{F(t)} \begin{bmatrix} T(t) \\ N(t) \\ B(t) \end{bmatrix}.$$

Here T is the unit tangent vector field, N is the unit normal vector field, B is the unit binormal field on α, and the speed, curvature, and torsion parameters are given by: $\nu = \|\alpha'\|$, $\kappa = \|\alpha' \times \alpha''\| / \|\alpha'\|^3$, $\tau = (\alpha' \times \alpha'') \cdot \alpha''' / \|\alpha' \times \alpha''\|$. F(t) describes the flow of the orthogonal frame through its tangent space. See O'Neill [1966] for a detailed discussion on the Frenet representation.

Given two simple curves $\alpha(t)$ and $\beta(t)$ and their orthogonal frames $F_\alpha(t)$, $F_\beta(t)$ parameterized on the unit interval [0, 1], define the set of diffeomorphisms the index set [0, 1] to itself as

$$\Phi = [\phi: [0, 1] \leftrightarrow [0, 1]]. \quad (1)$$

We define the distance $\rho(\alpha, \beta; \phi)$ by using the Frobenius norm between any 2 3×3 matrices A and B by: trace $(A - B)(A - B)^T$.

Definition

The distance ρ between the curves α and β is:

$$\begin{aligned} \rho(\alpha, \beta; \phi) &= \int_0^1 \text{trace}\,(F_\alpha(t) - F_\beta(t))(F_\alpha(t) - F_\beta(t))^T \\ &= 2\int_0^1 (\nu_\alpha(t)\kappa_\alpha(t) - \nu_\beta(\phi(t))\kappa_\beta(\phi(t)))^2 dt \\ &\quad + 2\int_0^1 (\nu_\alpha(t)T_\alpha(t) - \nu_\beta(\phi(t))T_\beta(\phi(t)))^2 dt \end{aligned}$$

Due to the properties of the Frenet representation, the distance ρ is inherently invariant to spatial position and orientation. The problem of matching curves across different brains is now defined as one of finding the particular diffeomorphism that minimizes the above distance. This is akin to the work of Younes [1998].

Problem statement

Given two curves $\alpha(t)$, $\beta(t)$, $t \in [0, 1]$ and the set of diffeomorphisms $\Phi = \{\phi : \phi(\alpha(t)) = \beta(t)\}$, matching the two curves is equivalent to finding the diffeomorphism that minimizes the above distance with a penalty on the velocity field:

$$\hat{\phi}(\alpha, \beta) \equiv \operatorname{argmin}_{\phi \in \Phi}\left(\int_0^1 ||v_\beta(\phi(t)) - v_\alpha(t)||^2 dt + \rho(\alpha, \beta; \phi)\right). \quad (2)$$

Introducing the simple norm-square function $\|v_{\beta_\phi} - v_\alpha\|^2$ in the matching gives the anatomist control over local scale. For generating the lowest cost diffeomorphism, we define the cost function ρ^Δ as the discrete approximation to the distance and then use the bipartite matching algorithm to reduce the complexity of the problem. Representing the curve as a linear array of straight line segments and assuming that curvature and torsion are piecewise constant, the distance assigned to the correspondence becomes:

$$\rho^\Delta(\alpha, \beta; \phi) = a \sum_j [v_\alpha(j) - v_\beta(\phi(j))]^2 \Delta_j + b \sum_j [v_\alpha(j)\kappa_\alpha(j) - v_\beta(\phi(j))\kappa_\beta(\phi(j))]^2 + c \sum_j [v_\alpha(j)\,\tau_\alpha(j) - v_\beta(\phi(j))\tau_\beta(\phi(j))]^2\,\Delta_j$$

where a, b, and c are coefficients picked by the anatomist to adjust the weight of the matching based on the speed, curvature, or torsion terms. Choosing to match based on speed ($a \neq 0$, $b = c = 0$) emphasizes uniform streching of the curves. Matching based on curvature ($b \neq 0$, $a = c = 0$) emphasizes the turning of the curves in the plane. Matching based on torsion ($c \neq 0$, $a = b = 0$) emphasizes twisting of the curves out of the plane. Matching based on a weighted combination of these criteria is also possible. The bipartite matching algorithm is presented in the Appendix.

GEOMETRIC REPRESENTATION AND GENERATION OF FUNDUS CURVES

We follow Khaneja [1996] for generating fundus curves. The deepest valleys of sulcal beds (fundus) are analogous to curves of extremal positive curvature resembling crest lines. We define the problem of tracking the fundus as a control / optimization problem of searching for a curve that passes through regions of highest maximal curvature and joins the manually-specified start and end points.

Near a point p in the surface, we express the surface as the graph of a function $z = f(x, y)$, such that $(x, y) \rightarrow f(x, y)$ is locally quadratically approximated by $f(p) = 0$, $f_x(p) = f_y(p) = 0$, with $f(x, y) = \frac{1}{2}(f_{xx}x^2 + 2f_{xy}xy + f_{yy}y^2)$. Define the 2×2 shape operator $S_p = \begin{pmatrix} fxx & fxy \\ fxy & fyy \end{pmatrix}$; the maximum and minimum eigenvalues κ_1, κ_2 of S_p are called the principal curvatures at p. The unit vector directions $\vec{t_1}$ and $\vec{t_2}$ in which these extreme values occur are called the principal directions. Then we define the fundus as the curve $\alpha(t)$, $\theta \in [0, 1]$ that minimizes $\int_\alpha (\kappa_m(t) - K)^2\, dt$, where K is the largest maximal curvature of the surface (the largest of the maximum eigenvalues of S_p evaluated over the entire surface) and κ_m is the principal curvature with the highest absolute value at each point on the curve (with its sign retained). This choice of κ_m steers the algorithm away from points of high negative curvature such as those on gyral curves, for which both the principal curvatures are negative. Using the corrected trapezoid approximation to this integral, the problem of extracting the fundus is reduced to finding the curve that minimizes:

$$H(\alpha) = \sum_{j=1}^{N-1}\left(\frac{(\kappa_m(x_j) - K)^2 + (\kappa_m(x_{j+1}) - K)^2}{2} + \frac{(\kappa_m(x_{j+1}) - \kappa_m(x_j))^2}{6}\right)\|x_j - x_{j+1}\|$$

over the triangulated surface where $x_0 = s$ and $x_N = t$ are the predefined start and end points of the fundus. For nodes s and t on the triangulated surface, consider the collection of all curves $\alpha(s, t)$ connecting (s, t), and define the discrete fundus between (s, t) as $\hat{\alpha}(s, t) \equiv \operatorname{argmin}_{\alpha(s,t)} H(\alpha)$. Assuming the triangulated surface is a finite state space of size N and the optimal path has no more than K nodes, there are N^K paths between the points x_0 and x_N. The brute-force algorithm would require generating all the N^K paths to find the optimal one, but since the cost is additive over the length of the path, we use dynamic programming to reduce the complexity to the order of KN^2. See Khaneja [1996] for details of the dynamic programming algorithm.

RESULTS AND DISCUSSION

We extracted various fundus curves from the macaque brains labeled 87A and 93I using the dynamic programming algorithm, and then matched these fundus curves using the induced Frenet distances. Shown in the top row of Figure 1 are the different fundus beds on the two brains 87A and 93I generated by the dynamic programming algorithm (Fig. 1A, B). Shown in Figure 1C is a demonstration of the dynamic programming algorithm handling the bifurification of the Sylvian fissure in the Visible Human with different sets of start and end points. The bottom row of Figure 1 shows the matching for superior temporal sulcus between the brains 87A and 93I. The bipartite graph matching algorithm (Appendix) was used to minimize the distance ρ^A. The parameters were chosen so that matching was based on speed for Figure 1A ($a = 1$, $b = 0$, $c = 0$), curvature for Figure 1B ($a = 0$, $b = 1$, $c = 0$), and torsion for Figure 1C ($a = 0$, $b = 0$, $c = 1$).

ACKNOWLEDGMENTS

This work was supported by NSF grant BIR 9424264 to M.I.M. NSF, grant R01 NS35368-02 to M.V. and M.I.M., and NIH grant 50567 to D.V.E. and M.I.M.

REFERENCES

Joshi SC, Wang J, Miller MI, Essen DV, Grenander U (1995a): On the differential geometry of the cortical surface. In: Meiter RA (editor) Proceedings of SPIE's 1995 Geometric Methods in Applied Imaging (San Diego, CA.) 9–14 July, 304–311 1995.

Joshi SC, Miller MI, Christensen GE, Banerjee A, Coogan TA, Grenander U (1995b): Hierarchical brain mapping via a generalized Dirichlet solution for mapping brain manifolds. In: Proceedings of SPIE's 1995 International Symposium on Optical Science, Engineering, and Instrumentation, Volume 2573, August, 1995, pp 278–289.

Khaneja N (1996): Statistics and Geometry of Cortical Features. M.S. Thesis. St. Louis: Department of Electrical Engineering, Sever Institute of Technology, Washington University.

Miller M, Banerjee A, Christensen G, Joshi SC, Khaneja N, Grenander U, Matejic L (1997): Statistical methods in computational anatomy. Stat Methods Med Res 6:267–299.

O'Neill B (1966): Elementary Differential Geometry. San Diego: Academic Press, Inc.

Sedgewick R (1983): Algorithms. Boston: Addison-Wesley Publishing Company, Inc.

Thompson P, Schwartz C, Lin R, Khan A, Toga A (1996): Three-dimensional statistical analysis of sulcal variability in the human brain. J Neurosci 16:4261–4274.

Welker W (1990): Why does cerebral cortex fissure and fold? Cereb Cortex 83:3–136.

Younes L (1998): Computable elastic distances between shapes. Siam J Appl Math (in press).

APPENDIX

BIPARTITE MATCHING

For the implementation, the target curve α is sampled with n equally spaced points and the template curve β is sampled with m points (m is approximately $N(n-1)$, where N is the spacing between the samples in the target). The neighborhood of each point i in the target curve is defined to be all points $j \in \beta$ such that $|i - j| < N$. We now have a bipartite weighted graph in which there are two distinct set of nodes (samples), and all edges in the graph connect two samples $i \in \alpha$, $j \in \beta$ where j is defined to be a neighbor of i. The weights on the edges are defined by:

$$w(i, j) = a[\nu_\alpha(i) - \nu_\beta(j)]^2 + b[\nu_\alpha(i)\kappa_\alpha(i) - \nu_\beta(j)\kappa_\beta(j)]^2 + c[\nu_\alpha(i)\tau_\alpha(i) - \nu_\beta(j)\tau_\beta(j)]^2.$$

Let i represent one of the n samples in the target. For each such sample, there is an associated cost array, where cost[i, j] = w(i, j) if the edge exists (j is in the neighborhood of i). cost[i, j] = ∞ if the edge does not exist. There is also a prefer[i, k] array for each sample i in the target, which contains the indices of the template samples j sorted by their costs in ascending order. For example, if $\min_j$cost[i, j] = j′ then prefer[i, 1] = j′, i.e., if j′ is the template point (among all the template points j) for which the cost function for the target point i is minimized, then j′ becomes the first point on point i's preference list. We also need to keep track of how far down each point in the target has progressed in its preference list. This is handled by the index[i] array, initialized to 1. The current match in the template assigned to point i is stored in the array match[i]. The algorithm proceeds as follows:

1. Initialize: index[i] = 1 $\forall i \in 1, \ldots n$, match[1] = prefer[1, 1]
2. for i = 2 to n do
   repeat
     isLegal = FALSE
     match[i] = prefer[i, index[i]]
     if match[i] > match[i − 1]
       isLegal = TRUE
     else
       find all p and q such that
         *p ≥ index[i − 1] and q ≥ index[i] where (p, q) ≠ (match [i − 1], match[i])
         *match[i] ≤ prefer[i − 1, p], prefer[i, q] ≤ match[i − 1] and
       $\delta_{i-1}(p)$ = cost[i − 1, match[i − 1]] − cost[i − 1, prefer[i − 1, p]

$\delta_i(q)$ = cost[i, match[i]] − cost[i, prefer[i, q]])
$\Delta(p, q) = \delta_{i-1}(p) + \delta_i(q)$
$p^* = \text{argmin}_p \Delta(p, q)$
$q^* = \text{argmin}_q \Delta(p, q)$
index[i − 1] = p*
index[i] = q*
if i ≠ 1 i = i − 1
until isLegal = TRUE.

The algorithm assigns each sample in the target the match with the lowest possible cost. If the match does not violate the diffeomorphism, the inner loop terminates. If it does, the algorithm finds the next match that does not violate the diffeomorphism for the last two points and maintains the lowest cost constraint. It then steps back to check if the new match violates the diffeomorphism for the previous points.