# Automatic Feature-based Surface Mapping for Brain Cortices

Fabien Vivodtzev[1] David F. Wiley[2] Lars Linsen[3] James Jones[2] Nina Amenta[2]
Bernd Hamann[2] Kenneth I. Joy[2]

[1] Laboratoire GRAVIR (UJF, CNRS, INPG, INRIA)
Grenoble, France
[2] Institute for Data Analysis and Visualization (IDAV)
Department of Computer Science
University of California, Davis
Davis, CA 95616, U.S.A.
[3] Department of Mathematics and Computer Science
Ernst-Moritz-Arndt-Universität Greifswald
Greifswald, Germany

## ABSTRACT

We present a method that maps a complex surface geometry to an equally complicated, similar surface. One main objective of our effort is to develop technology for automatically transferring surface annotations from an atlas brain to a subject brain. While macroscopic regions of brain surfaces often correspond, the detailed surface geometry of corresponding areas can vary greatly. We have developed a method that simplifies a subject brain's surface forming an abstract yet spatially descriptive point cloud representation, which we can match to the abstract point cloud representation of the atlas brain using an approach that iteratively improves the correspondence of points. The generation of the point cloud from the original surface is based on surface smoothing, surface simplification, surface classification with respect to curvature estimates, and clustering of uniformly classified regions. Segment mapping is based on spatial partitioning, principal component analysis, rigid affine transformation, and warping based on the thin-plate spline (TPS) method. The result is a mapping between topological components of the input surfaces allowing for transfer of annotations.

**Keywords:** Surface mapping, surface segmentation, discrete curvature, PCA, thin-plate splines.

## 1. INTRODUCTION

The mapping of one surface to another has been the subject of research in computer graphics for a long time. Many automated, semi-automated, and interactive approaches exist. Depending on the driving application, different approaches are favorable. We approach the problem from a neuroscientific background.

Surfaces representing the boundary of a mammalian and, in particular, human brain cortex exhibit a complex geometry with many deep meandering folds representing a two-manifold surface. Neuroscientists refer to the deep folds in a brain cortex as *sulci*. The bumps between the folds are called *gyri*. Except for the cerebellum region, sulci and gyri determine the global shape of a brain. Sulcal and gyral regions are of interest for mapping purposes since they delineate functional brain areas.

When mapping annotations from an atlas brain surface to a subject brain surface, the goal is to detect the individual gyral and sulcal regions, as well as the cerebellum, and find a one-to-one correspondence between regions of the atlas and those of the subject. Due to inter-subject variation, the detection of corresponding brain regions is not trivial. The same brain regions for two individuals, as identified by a neuroscientist, may vary in size, location, orientation, and shape. Thus, a reliable detection of functional brain regions as a first processing step is essential for the success of any mapping process. The subject becomes particularly challenging when considering diseased brains, where some functional regions can be heavily deformed or not present at all. We do not attempt to solve the problem of automatically mapping heavily distorted diseased brain surfaces. However, we believe our method can be supplemented by user guidance to correctly map diseased brain surfaces.

More generally, we address the problem of surface mapping for surfaces with similar global geometry. Our algorithms are not applicable to mapping of any two surfaces. The surfaces need to exhibit common geometric structures. We detect

[1]fabien.vivodtzev@imag.fr [2]{wiley, jonesj, amenta, hamann, joy}@cs.ucdavis.edu [3]linsen@uni-greifswald.de

features of the surfaces and base our mapping on correspondences of matching features. We do not require other brain-specific properties such as symmetry or being of genus zero, as long as all such geometric properties are/are not co-existent in both surfaces.

Our approach consists of two steps, the detection of functional brain regions and the mapping of these regions. In the context of surface mapping, we refer to the brain regions as *segments* or *surface segments*. The two steps of our approach are referred to as surface segmentation and segment mapping.

The surface segmentation part, described in Section 3, consists of several processing steps. For example, starting with a surface obtained by isosurface extraction from 3D medical imaging data (usually computed tomography (CT) or magnetic resonance imaging (MRI) data), the surface representation can be heavily influenced by noise and scanning artifacts, which we treat by applying a Laplacian filter. The resulting surface contains both high- and low-frequency features, which we simplify to eliminate high-frequency details since we are primarily interested in “true” shape features and not high-frequency perturbations. The simplified surface is segmented based upon curvature. We choose a threshold that separates concave from convex surface regions, leading to a separation of sulci from gyri. Separated regions are clustered (or merged) to form regions representing functional areas of a brain, which we call segments (sulci). We use these segments to define a data abstraction in the form of a point cloud positioned by decomposing the segments into several subsegments and replacing these subsegments by a vertex in the point cloud at the centroid of each subsegment. Vertices for two abstract brain point clouds are matched to determine the mapping between segments. Figure 1 illustrates these steps.

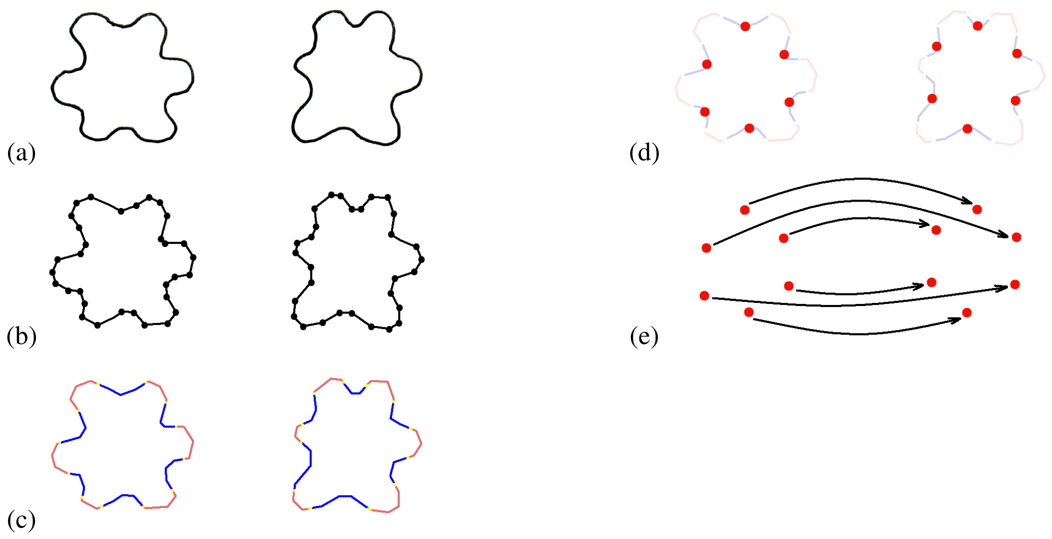

**Figure 1.** Matching process. (a) Original high-resolution surfaces are smoothed and simplified to produce a low-resolution representation (b). (c) Curvature information is used to segment the surface into sulci (blue) and gyri (red) regions. (d) Sulcal regions are converted to points (red dots) which are iteratively matched (e).

The segment mapping part, described in Section 4, is based on spatial examination of the point cloud using principal component analysis (PCA), combined with a non-linear global warping between two point sets based upon thin-plate spline (TPS) warps. Based on the point cloud warping, vertices are matched to one another, defining a mapping of the associated segments. This step includes a voting system to obtain a robust and error-tolerant segment mapping. The surface mapping is defined as the combination of all segment mappings.

We have chosen a hybrid mesh/point-cloud approach, since a mesh representation is favorable for surface segmentation, where we can exploit connectivity information, whereas the point cloud representation is favorable for unconstrained mapping. We intentionally decouple the two steps to obtain independent results. The voting system combines them again while providing the possibility to handle and compensate errors resulting in both steps, i. e., surface-segmentation errors due to noise and misclassification errors during matching.

## 2. RELATED WORK

Several different approaches exist for mapping two surfaces to each other, e. g.,.[1,2] A survey of existing mesh parameterization algorithms to map one mesh to another is given in.[3] These methods are general and can operate on any two surfaces. In our case, we are dealing with surfaces, which tend to be structured similarly at a macroscopical level and only differ in fine detail. Thus, we propose to exploit our constrained environment by automatically finding homologous points that the user would normally have to specify manually.

A general mapping algorithm would not assure that a specific anatomic region of the atlas brain is mapped to the corresponding anatomic region of the subject brain. Since we know that the surfaces to be mapped have similar geometrical features, we can use this *a priori* knowledge for a correct mapping of corresponding features. Mapping is still not straight forward, as shape and position of these feature may vary significantly due to inter-subject variation. However, the features will have a certain relative position to each other that does not vary too much. We can exploit this fact by extracting features and defining the mapping based on feature matching. Kraevoy and Sheffer[4] propose the concept of cross-paramterization, which allows to find constraint paramterizations. The constraints could impose the mapping of homologous points. They do not propose any automatic detection of homologous points. Also, their approach cannot compensate for errors made during the assignment of homologous points, which is necessary when dealing with automatic approaches.

The method described in[5] outlines a process that deforms subject or test brain volumes (obtained via MRI) using B-splines into the space of a volumetric atlas which has annotation information for each voxel. A voting system is employed to determine the most likely annotation for each voxel in the subject brain volume. The drawback of using B-splines in this context is that the B-spline deformation function is an approximation to the required warp and results in inaccurate warping when using a low-density grid to define the B-splines. This method requires the specification of feature points within the two volumes to set up the deformation, which may be selected manually, semi-automatically, or automatically. The classification method described in[6] uses nearest-neighbor information to distinguish the sulcal regions as opposed to attempting to match the topological structure of the brain surface.

We have extended the method described in[7] and provide an overview of the general process in the following. Given two surfaces, we first segment each based on curvature. This step finds the sulci and gyri of the brain surface as well as connecting regions in-between. Next, we construct a hierarchical graph representation for both surfaces, where graph nodes correspond to sulci and gyri triangle-patches, and graph edges represent how the sulci and gyri are connected to each other over the surface. Beginning with a coarse representation of each hierarchical graph, we then match graph nodes based on spatial locality and propagate those correspondences upwards in the hierarchy (from coarse to fine levels) to finally obtain a complete mapping.

Our contribution is the improvement of the mapping by increasing the likelihood of correctly matching nodes of the graph by using alignment and warping techniques. One can imagine a spectrum in terms of number of points used for matching surface features. One could use high-resolution meshes (consisting of several thousand vertices) of both surfaces and try to match them. Or, one could simplify such a high-resolution mesh to a "low-resolution mesh" and use that. The method we have proposed in[7] converts a decimated low-resolution mesh to an abstract graph by converting each segment to a single point and forming connecting edges by analyzing segment connectivity. Ideally, an algorithm would match the high-resolution meshes together. This goal, however, turned out to be not feasible due to complexity and performance. Our proposed method converts each segment into several vertices leading to a result "in-between." We also improve some of the other steps used in this approach—primarily, filtering input surfaces and refining curvature estimates. We do not use the graph structure; rather, we divide sulcal segments into several pieces in order to obtain spatial information regarding the segment.

## 3. SURFACE SEGMENTATION

We perform a number of steps to find the sulcal and gyral regions. These steps are described in the following sections.

### 3.1. Laplacian Filter

An isosurface extracted from 3D medical imaging data often suffers from noise caused by various sources during the imaging and extraction process. Since surfaces of anatomical objects tend to be smooth, a low-pass filter can be employed to reduce the high-frequency noise (while maintaining the actual data). Iterative application of a simple Laplacian filter has proven to be effective in many application areas. For triangular surface smoothing, a discrete version of the Laplacian filter applied to a vertex $\mathbf{p}$ is defined by

$$L(\mathbf{p}) = \mathbf{p} - \lambda \cdot (\overline{\mathbf{p}} - \mathbf{p}) \ ,$$

where $\overline{\mathbf{p}}$ is the barycenter of the edge-connected neighbor vertices of $\mathbf{p}$ (1-ring), and the constant $\lambda$ regulates the step size.

Since simultaneous application of the Laplacian filter to all vertices of the surface leads to shrinking, we alternate filtering steps with positive and negative $\lambda$, as proposed in.[8] This approach keeps the volume enclosed by the surface nearly constant. A few iteration steps suffice to obtain pleasing results. Other smoothing operators have been developed to only smooth along the direction of the surface normal,[9,10] but in our case a relaxation of the triangles is desired, as more uniform triangles lead to better results during further processing. Figure 2 illustrates smoothing using the Laplacian filter.

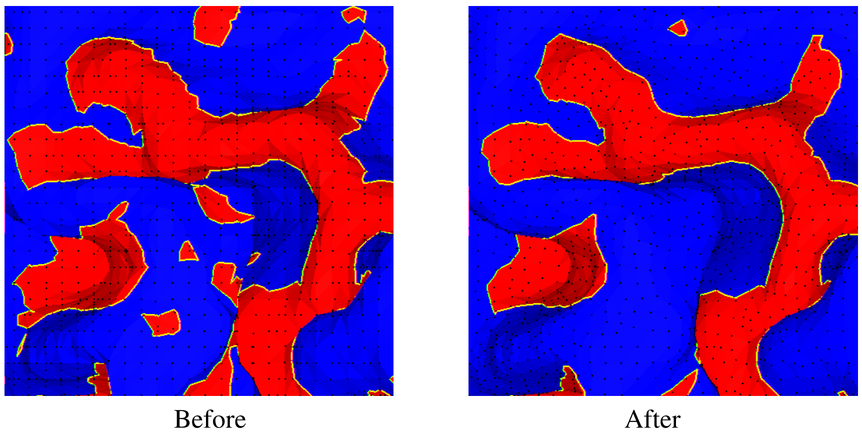

**Figure 2.** Laplacian filter application.

### 3.2. Surface simplification

To simplify a high-resolution triangular mesh we use a simplification algorithm based on progressive meshes.[11] We iteratively apply edge-collapse operations. Although collapsing an edge is a simple operation, it can modify topology and geometry. To ensure consistency of our mesh, we use consistency checks as described in,[12] based on topological analysis in the neighborhood of a collapse region.

For each edge of the mesh, an error corresponding to the cost of its collapse is computed and stored. According to this value an ordered heap of edges is created. During mesh simplification, the method identifies the top edge, checks for consistency, and, if possible, collapses it. This process is highly dependent on the error metric used to decide which edge to collapse. Many metrics have been proposed for edge collapse algorithms over the past decade.[11,13–15] Most of these metrics attempt to preserve sharp edges and details. Our objective is to remove detail even in regions of high curvature. Thus, our error metric is only based on edge length, and our main goal is to create an even distribution of vertices on the surface. After a valid collapse, the affected neighborhood is updated accordingly.

Topology (i. e., adjacency information of triangles) and geometry (i. e., position of the resulting "collapse" vertices) are modified by an edge collapse. An edge collapses to its midpoint. (We decided not to optimize the position to keep computation costs low.) Using midpoints also reduces the risk of self-intersections.

### 3.3. Curvature-based Classification

The method we have presented in[7] uses a discrete curvature estimate to distinguish between gyri and sulci, but a method that provides "smooth" boundaries between these different regions is clearly more desirable. By applying a continuous curvature approximation for each vertex in the mesh, we can find smooth boundaries between gyral and sulcal regions. The resulting smooth boundaries lie on the surface triangulation, but do not necessarily follow triangle edges, as they do in the method discussed in.[7]

Several approximation schemes have been proposed to estimate the curvature over a piecewise linear domain.[16–18] More recently, curvature tensors based on concepts of differential geometry have been used to compute a piecewise linear curvature tensor field. As described in,[19] the curvature tensor has a local minimum along an edge and a local maximum across the edge and can be estimated for any point on an edge. By integrating the contributions of all edges inside a certain domain centered at a vertex, we obtain the curvature tensor for a specific vertex. The two highest eigenvalues of this tensor are the principal curvature estimates of the vertex.

### 3.4. Clustering

Based on the continuous curvature estimates we can identify similar regions subject to a properly chosen threshold. For example, the curvature estimate may provide a positive value for a bump and a negative value for a fold. Thus, a threshold of zero would allow one to find a meaningful smooth boundary between the two types of features. This step detects and separates sulci and gyri of the brain surface. Figure 3 shows the result, a segmented brain cortex.

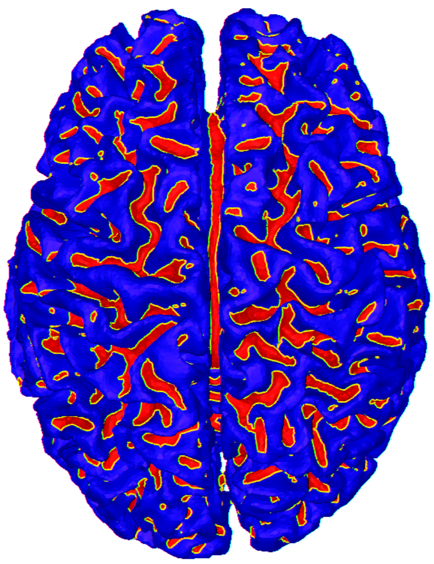

**Figure 3.** Segmented brain cortex using curvature-based classification and clustering after surface smoothing and simplification. Detected sulci are shown in red.

## 4. SEGMENT MAPPING

We convert the segments representing the sulci into an abstract point cloud that represents the segments spatially for each brain surface. This step is performed by converting the groups of triangles representing each segment into a number of representative vertices in the point cloud. Once this point cloud is constructed, we use an iterative matching scheme to establish correspondences for the vertices of the atlas point cloud and those of the subject point cloud. Once matched, we employ a voting system to determine the correspondence of atlas regions to those in the subject brain. These steps are described in more detail in the following sections.

One may also consider the abstract points as data points, since they capture more information than merely position. In particular, they store references to the segments that were used for their generation. This additional information is exploited by the voting system to evaluate the “votes.”

### 4.1. Segment Partitioning

Segments are broken into manageable pieces by partitioning them using a binary space partition (BSP). Large meandering segments are converted to several nodes to spatially track the segment. Smaller segments may be converted to one node.

Care must be taken when converting segments to vertices in the point cloud. Consider, for example, a segment that is shaped like a tuning fork, see Figure 4. The space partitioning needs to be carefully tuned to accurately represent complex segment topologies. Although we are dealing with surfaces and not volumes, we have found that a BSP approach (when taking into account the problem shown in Figure 4) works well for our brain data sets. For other data sets, it may be beneficial to replace the BSP step with a surface-based space partitioning method.

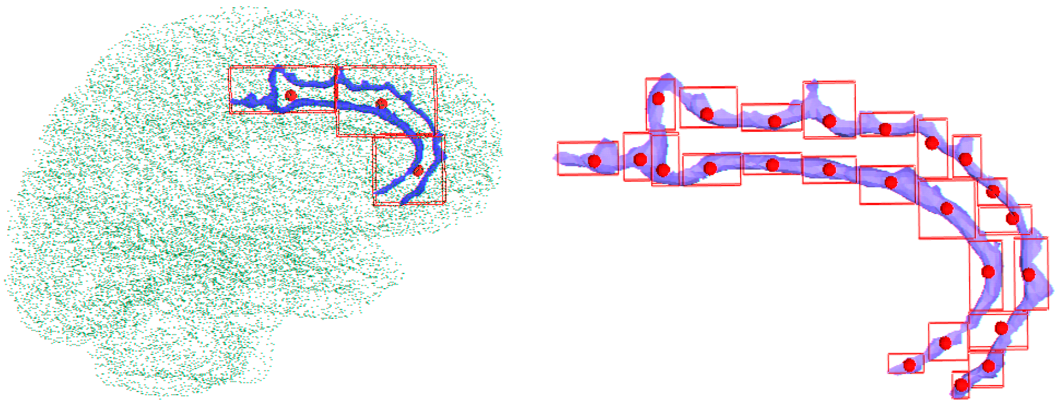
**Figure 4.** The "tuning fork" problem that occurs when converting segments to nodes ina point cloud. Left image shows "tongs" that are not represented sufficiently by the point cloud vertices (red dots). Right image shows a sufficient representation of the fork.

It is crucial to capture sufficient spatial information for each segment. However, one must also be aware of the tradeoff between having too many vertices and not having enough. Too many vertices can cause a bottleneck due to time complexity. However, not having enough vertices to represent segments spatially can result in poor segment matching.

## 4.2. Initial Homology

We use an iterative process to align the vertices in the abstract point clouds. We take advantage of symmetry and automatically find several homologous points in the two brains being mapped. We use a sequence of PCA steps, applied to subsets of points to find homologous points in space. The process is described for one brain. First, we apply PCA to the entire set of points to find the centroid (homologous point $\mathbf{h}_1$) and principal components of the data. Then, we partition the brain into two halves using the sagital plane (i. e., the vertical plane separating left and right halves) determined by the principal axes. Operating on each half separately, we apply PCA to each half, finding the centroid of the halves (homologous points $\mathbf{h}_2$ and $\mathbf{h}_3$ for the left and right halves, respectively). Next, we partition each half into four subregions, based on the PCA axes, with orthogonal horizontal and vertical planes passing through each halves' centroid. Finding the centroid of each of the subregions yields homologous points $\mathbf{h}_4, \mathbf{h}_5, \mathbf{h}_6$, and $\mathbf{h}_7$ for the left half, and $\mathbf{h}_8, \mathbf{h}_9, \mathbf{h}_{10}$, and $\mathbf{h}_{11}$ for the right half. Figure 5 illustrates this process.

## 4.3. Matching

After identifying a few key homologous points, we begin an iterative process that aligns and warps the subject brain point cloud into the space of the atlas point cloud. We assume that, after warping into the space of the atlas, vertices near the "correctly" matched vertices match up, as well. We use a tight tolerance that only permits the matching of a few vertices at each iteration. Using these new vertices to re-define the warp, we iteratively improve the warping of space to better line-up the vertices in the point clouds.

We draw upon two methods that are often used in the context of biological morphometrics. The first is *Procrustes method* (called *Horn's method* in computer vision); and the second is the TPS warp. We briefly describe both methods.

### 4.3.1. Horn's Method

Given are two point sets $P = \{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n\}$ and $Q = \{\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_n\}$, where points with the same indices correspond to each other. There exists an affine transformation consisting of a translation, scaling operation, and rotation that minimizes $\sum_{i=0}^{n} \parallel \mathbf{p}_i - \mathbf{q}_i \parallel$. Details for determining this transformation are provided in.[20]

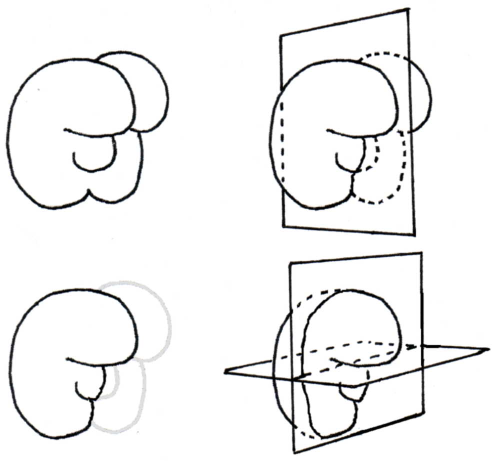

**Figure 5.** Finding homologous points by partitioning each brain based on PCA axes. Upper-left image shows input brain surface. Upper-right image shows partitioning into two halves. Lower-left image shows right half of brain. Lower-right image shows partitioning of right half into four subregions.

This transformation allows for an initial alignment of the point sets so that they are as close to each other as possible using an optimal rigid affine transformation. However, the corresponding point pairs $(\mathbf{p}_i, \mathbf{q}_i)$ are not necessarily near enough (spatially) to imply correspondences between neighboring vertices. Thus, we include the additional step of warping one point set into the space of the other to further improve correspondence.

### 4.3.2. Thin-plate Spline Method

The affine transformation only provides an initial, rough alignment. If two surfaces are substantially different, in terms of features and their distribution, it is impossible to map them using only a rigid transformation. A warping technique commonly used in biological morphometics for quantitative measurement as well as warping and morphing purposes, called the TPS method, is briefly described in the following.

Given are two point sets $P = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$ and $Q = \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_n\}$, where points with the same index correspond to each other. To construct a one-way warp from $P$ to $Q$, we first compute the difference vector set between the two point sets, i. e., $D = \{\mathbf{q}_0 - \mathbf{p}_0, \mathbf{q}_1 - \mathbf{p}_1, \ldots, \mathbf{q}_n - \mathbf{p}_n\}$. For 3D difference vectors, we construct three component functions (or "height fields") for the $x, y$, and $z$ components of $D$ so that the value of the component function is the "height" positioned at the locations in $P$. Next, we construct interpolating functions $X, Y$ and $Z$ through each height field, such that $D_i = (X(\mathbf{p}_i), Y(\mathbf{p}_i), Z(\mathbf{p}_i))$.

Given a point $\mathbf{p} = \{x, y, z\}$ in the space of $P$ that we want to warp to a point in the space of $Q$, yielding $\mathbf{p}'$, we sample $X, Y$ and $Z$ at $\mathbf{p}$ to determine the offsets needed to translate (or warp) the point, i. e., $\mathbf{p}' = (x + X(\mathbf{p}), y + Y(\mathbf{p}), z + Z(\mathbf{p}))$. Figure 6 shows a 2D example of this process.

The standard method for constructing the interpolating surfaces for $X, Y$ and $Z$ in biological morphometrics is based on the TPS approach. The TPS minimizes the "bending energy" defined by the second derivative, constrains the second derivative to zero at infinity, and provides a smooth surface. Other surface approximation or interpolation schemes, such as those discussed in[21] and,[22] could be used, but the TPS method is the one that is generally accepted in biological sciences. A more detailed description of TPS-based warping in biological sciences is provided in.[23,24]

The advantage of computing interpolating (or approximating) component functions (height fields) is the fact that they can be sampled at any location, not just at those in $P$. We take advantage of this fact since we want to use only a few points to define the warp from an unknown brain $P$ to an atlas brain $Q$. We can transform all vertices in the subject brain into the space of the atlas brain. Consequently, these transformed vertices should lie more closely to their counterpart (or corresponding) vertices in the atlas brain. Finding a few points that are close to each other at this point, adding them to $P$ and $Q$, re-computing the warp from $P$ to $Q$, and transforming the unknown points again, iteratively improves their positions.

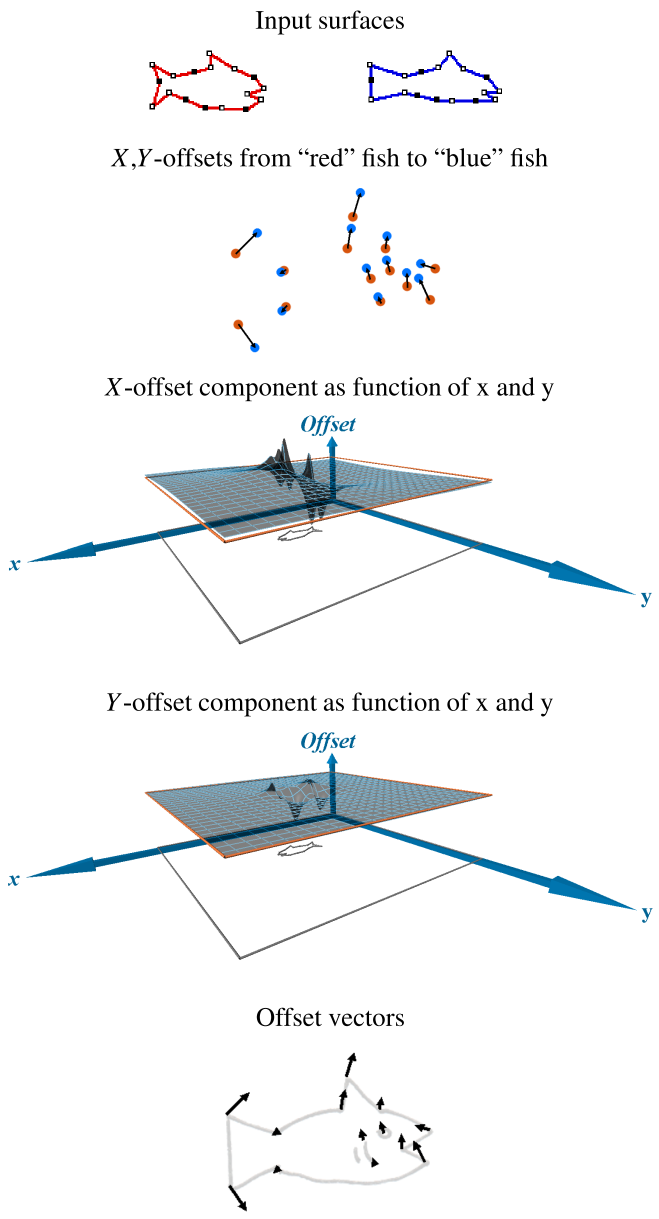
Input surfaces
X,Y-offsets from "red" fish to "blue" fish
X-offset component as function of x and y
Offset
x
y
Y-offset component as function of x and y
Offset
x
y
Offset vectors

**Figure 6.** Warping process. Top image shows input surfaces with homologous points marked as white boxes. The next images show the difference vectors that are used to compute the component functions ($X$ and $Y$). Bottom image shows the difference vectors that are obtained when sampling the component functions.

### 4.3.3. Iteration

Figure 7 shows the initial alignment of two brain point clouds after being aligned and warped using Horn's method and TPS warping, respectively. Blue dots indicate the atlas point cloud, and red dots indicate the subject point cloud. At each iteration, we must choose a small number of points to match in preparation for the next iteration. This is done in three steps. First, for each unmatched point $\mathbf{a}_i$ in the atlas point cloud, we find the closest unmatched point $\mathbf{s}_j$ in the subject point cloud. Second, to compute a selection "weight" $w_{i,j}$, we weigh each pair based on how far away the two points are from the nearest matched point $\mathbf{m}$ ($\mathbf{m}$ representing one point from the atlas and one from the subject, being exactly the same). Thus, the weight $w_{i,j}$ is computed as $w_{i,j} = d_{i,j}d_id_j$, where $d_{i,j}$ is the distance between $\mathbf{a}_i$ and $\mathbf{s}_j$, $d_i$ is the distance between $\mathbf{a}_i$

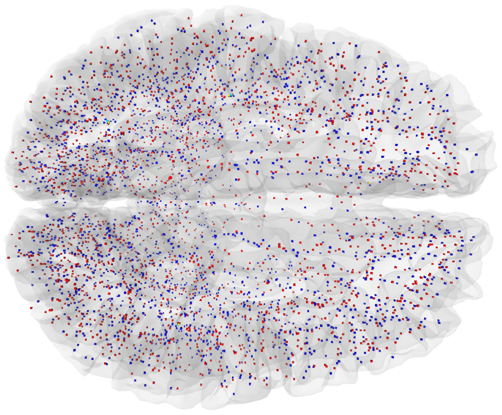

**Figure 7.** Initial warp based on automatically chosen homologous points. Blue dots indicate the atlas point cloud and the red dots indicate the subject point cloud.

and $\mathbf{m}$, and $d_j$ is the distance between $\mathbf{s}_j$ and $\mathbf{m}$. Finally, we choose $M$ pairs having the smallest values for their respective weights $w_{i,j}$. Typically, we select $M$ to be between 4 and 6 so that we have to match only a small number of points at each iteration, ensuring that the warped space does not change dramatically between iterations.

The iteration process is terminated by exhausting all possible point pairs, either by matching all points or by not having a point pair that is within a specified tolerance. A maximum distance requirement for point pairs is needed towards the end of the iterative process when nearly all points are matched. Without the maximum-distance requirement, points that are quite distant from one another are matched causing problems with the deformed space and incorrectly matched segments.

### 4.4. Voting System

Once the iterative process terminates, we classify the subject segments by examining the ratios of atlas segments that have been matched to each subject segment. Keeping in mind that each sulcal region may be represented by several points in each point cloud, we introduce a voting system to evaluate segment matching. Each atlas segment is mapped to the subject segment, with whose points the majority of the atlas segment's points have been matched. Using a voting system, we can account for some mismatches and we can give and assure reliability measures for our mapping. Moreover, large meandering atlas segments may map to several segments in the subject brain. Our algorithm allows for such a one-to-many correspondence. However, several small atlas segments can also be mapped to large meandering segments in the subject brain (many-to-one correspondence). The correct approach would be to split the subject segment and map each atlas segment to a corresponding subdivided subject segment. Our current implementation simply chooses the atlas segment that matches more points and we leave this splitting technique as well as many-to-many correspondences as a topic of future work.

## 5. RESULTS AND DISCUSSION

Our first example demonstrates the segment matching for a simple surface obtained using a laser range scan of clay brain models. Figure 8 shows the transference of colored sulcal segments onto the subject brain. This visualization nicely demonstrates that our method is capable of matching corresponding features, even if their location varies significantly from one brain to the other.

Figure 8 also indicates that it is important to include many-to-one and many-to-many correspondences. The segment colored in magenta on the left is matched with a "Y-shaped" segment on the right. The Y-shaped sulcus is represented by one segment for the subject brain but by more than one segment for the atlas brain. In our current implementation, only one of the segments of the atlas brain is matched. The atlas brain segment representing the second "branch" of the Y-shaped sulcus has not been matched and, thus, is not displayed. A many-to-one or many-to-many correspondence could be established by applying the mapping in both directions and combining the two results.

Our second example demonstrates our method when applied to a human brain cortex. Figure 9 shows a top view and Figure 10 shows an image viewing the right side of the brain. This example required about 45 minutes to complete the

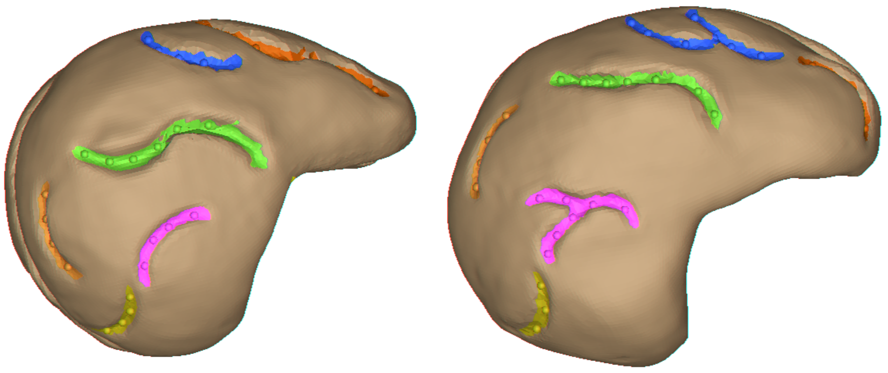
**Figure 8.** Mapping of colored sulcal segments from clay models from atlas brain (left) to subject brain (right).

matching process on a PC with a 1GHz Pentium III processor and 512MB RAM. There were approximately 2100 vertices in the point clouds, and we matched only six vertices per iteration.

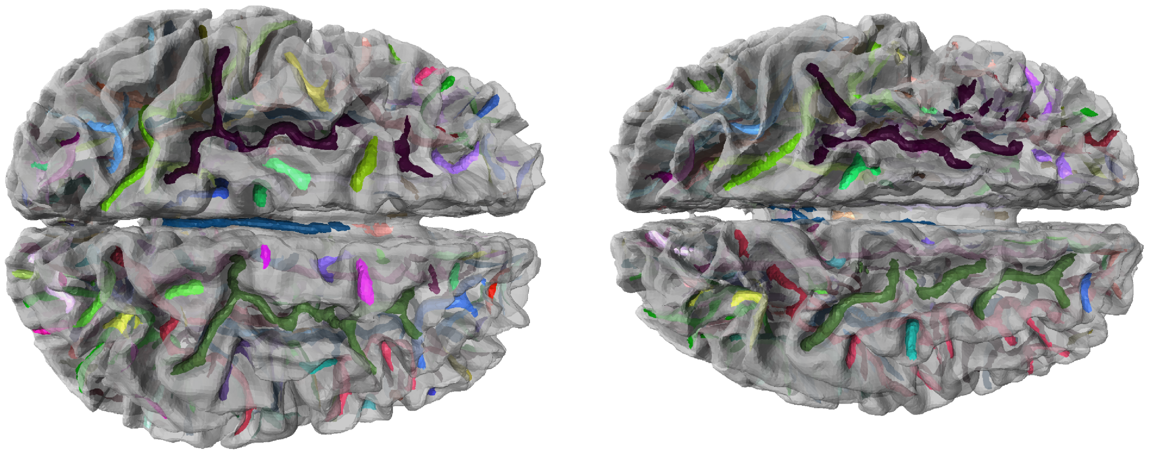
**Figure 9.** Top view of brain cortex. Left image shows the atlas brain and right image shows the mapping of atlas sulci onto the subject brain.

Performance considerations are simple. There is a tradeoff between matching reliability and time complexity. For brain surfaces, we have found that between 1000 and 2000 points provide sufficient spatial information for the sulcal regions, and one can obtain reasonable mappings. The more points one uses, the more the voting system becomes statistically reliable. However, there are two bottlenecks: Finding which vertices to match at each iteration, and computing the TPS warp. Finding points can be optimized by space partitioning, though the TPS computation will always have time complexity $O(N^2)$, where $N$ is the number of matched vertices used to set up the TPS transformation. In our implementation, for the images computed in Figures 9 and 10, we have found that each iteration requires about eight seconds for approximately 320 iterations on a standard Pentium workstation. Memory requirements were minimal.

The applicability of our approach is limited to surfaces with similar (global) geometric structures. Our algorithm is based on detecting such structures (or features) and matching them to derive a surface mapping. We believe that many surface mapping or surface morphing applications deal with surfaces with similar geometric properties. Thus, there should be many other applications for our approach. Our algorithm should also be able to handle surfaces of high genus and even surfaces with non-connected parts, but we have not tried it on such data sets yet.

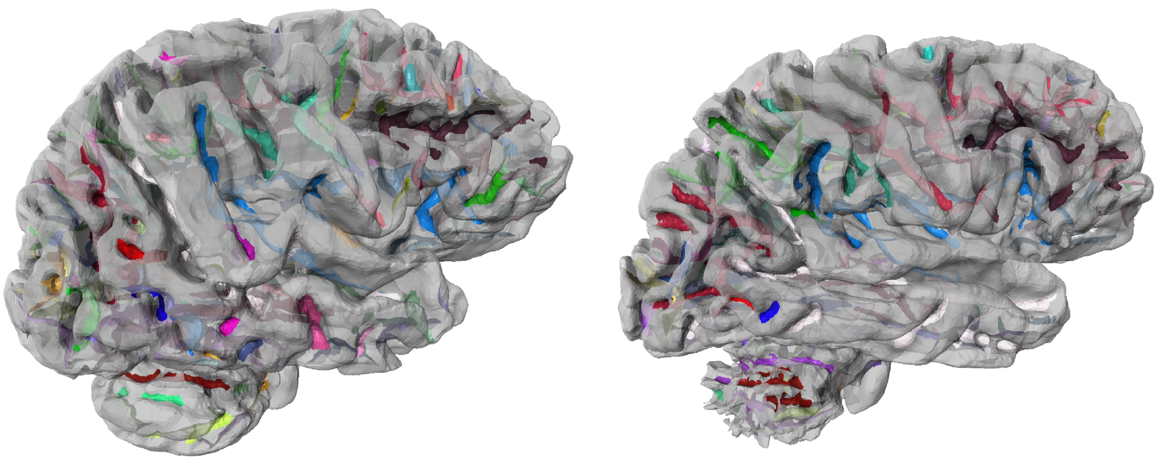

**Figure 10.** Right view of brain cortex. Left image shows the atlas brain and right image shows the mapping of atlas sulci onto the subject brain.

## 6. CONCLUSIONS AND FUTURE WORK

We have presented a method for automatic feature-based surface mapping. Our approach is applicable to any surfaces with similar geometric structure. By detecting features and using them to establish correspondences between the surface, our method is capable of matching complex surface geometries such as brain cortices.

Our approach consists of two steps, a surface segmentation and a segment matching. The surface segmentation makes use of discrete curvature estimates to classify the surface into regions of well-defined geometric regions. The segment matching is based on space warping applied to a point cloud that has been derived from the segmentation. These two decoupled steps are brought together by a voting system that determines the best matches. The voting system is tolerant to errors made in both preceding steps and can assure reliability measures.

One enhancement we plan to devise is a step that improves the matching process by arbitrarily removing some of the matched vertices so that they can be rematched in a future iteration. The idea is that, if there was an error early in the matching process, the mismatch can be fixed by removing the matched pair and allowing them to re-match to possibly another segment. To determine which matched pairs to repeat this process for could also be driven by the segment matching percentages where segments that have a low matching ratio are un-matched and could be considered for a re-matching step. If they match back to the original segment, then it most likely was a correct match.

In order to adapt this method to highly distorted (e. g., diseased) brains, it is necessary to manually specify the homologous points used to begin the first iteration. Once this homology is defined, the iterative process can automatically match segments.

### Acknowledgements

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); and the National Institute of Mental Health and the National Science Foundation under contract NIMH 2 P20 MH60975-06A2. We thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis.

## REFERENCES

1. M. S. Floater, “Parameterization and smooth approximation of surface triangulations,” *Computer Aided Geometric Design* **14**, pp. 231–250, 1997.

2. J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe, “Inter-surface mapping,” in *Proceedings of ACM SIGGRAPH*, pp. 870–877, 2004.
3. M. S. Floater and K. Hormann, “Surface parameterization: a tutorial and survey,” in *Advances in Multiresolution for Geometric Modelling*, N. A. Dodgson, M. S. Floater, and M. A. Sabin, eds., pp. 157–186, Springer-Verlag, Heidelberg, 2004.
4. V. Kraevoy and A. Sheffer, “Cross-parameterization and compatible remeshing of 3d models,” *ACM Trans. Graph.* **23**(3), pp. 861–869, 2004.
5. D. Shen and C. Davatzikos, “Hammer: Heirarchical attribute matching mechanism for elastic registration.,” *IEEE Trans. Med. Imaging* **21**(11), pp. 1421–1439, 2002.
6. K. J. Behnke, M. E. Rettmann, D. L. Pham, D. Shen, S. M. Resnick, C. Davatzikos, and J. L. Prince, “Automatic classification of sulcal regions of the human brain cortex using pattern recognition,” in *Medical Imaging 2003: Image Processing. Edited by Sonka, Milan; Fitzpatrick, J. Michael. Proceedings of the SPIE, Volume 5032, pp. 1499-1510 (2003).*, pp. 1499–1510, May 2003.
7. F. Vivodtzev, L. Linsen, G.-P. Bonneau, B. Hamann, K. I. Joy, and B. A. Olshausen, “Hierachical isosurface segmentation based on discrete curvature,” in *Data Visualization 2003, Proceedings of VisSym 2003*, G.-P. Bonneau, S. Hahmann, and C. D. Hansen, eds., pp. 249–258, 2003.
8. G. Taubin, “A signal processing approach to fair surface design,” in *SIGGRAPH ’95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 351–358, ACM Press, 1995.
9. I. Guskov, W. Sweldens, and P. Schröder, “Multiresolution signal processing for meshes,” in *SIGGRAPH ’99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 325–334, ACM Press/Addison-Wesley Publishing Co., 1999.
10. L. Linsen and H. Prautzsch, “Global versus local triangulations,” in *Proceedings of Eurographics 2001, Short Presentations*, J. Roberts, ed., pp. 257–263, 2001.
11. H. Hoppe, “Progressive meshes,” in *Proceedings of SIGGRAPH 1996*, pp. 99–108, ACM Press, 1996.
12. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, “Mesh optimization,” in *Proceedings of SIGGRAPH 1993*, pp. 19–26, ACM Press, 1993.
13. M. Garland and P. Heckbert, “Surface simplification using quadric error metrics,” *Computer Graphics, 31st Annual Conference Series* , pp. 209–216, 1997.
14. P. Lindstrom and G. Turk, “Fast and efficient polygonal simplification,” in *Proceedings of IEEE Conference on Visualization 1998*, pp. 279–286, IEEE Computer Society Press, 1998.
15. R. Ronfard and J. Rossignac, “Full-range approximation of triangulated polyhedra,” *Computer Graphics Forum, Proceedings of Eurographics 1996* **15**(3), 1996.
16. B. Hamann, “Curvature approximation for triangulated surfaces,” in *Geometric modelling, Computing Suppl. 8*, G. Farin, H. Hagen, and H. Noltemeier, eds., pp. 139–153, Springer-Verlag, 1993.
17. G. Taubin, “Estimating the tensor of curvature of a surface from a polyhedral approximation,” in *ICCV ’95: Proceedings of the Fifth International Conference on Computer Vision*, p. 902, IEEE Computer Society, 1995.
18. M. Meyer, M. Desbrun, P. Schroder, and A. Barr, “Discrete differential geometry operators for triangulated 2-manifolds,” 2002.
19. D. Cohen-Steiner and J.-M. Morvan, “Restricted delaunay triangulations and normal cycle,” in *SCG ’03: Proceedings of the nineteenth annual symposium on Computational geometry*, pp. 312–321, ACM Press, 2003.
20. B. K. P. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *Jounral of the Optical Society of America A* **4**, pp. 629–642, 1987.
21. D. R. Forsey and R. H. Bartels, “Hierarchical b-spline refinement,” in *Proceedings of ACM SIGGRAPH*, pp. 205–212, 1988.
22. R. Franke and G. M. Nielson, “Scattered data interpolation and applications: A tutorial and survey,” in *Geometric Modeling: Methods and Their Applications*, H. Hagen and D. Roller, eds., pp. 131–160, Springer, 1990.
23. F. L. Bookstein, *Morphometric tools for landmark data: Geometry and Biology*, Cambridge Univ. Press, New York, 1991.
24. F. Bookstein, “Applying landmark methods to biological outline data,” in *Proceedings in Image Fusion and Shape Variability Techniques*, K. Mardia, C. Gill, and I. Dryden, eds., pp. 59–70, University of Leeds Press, Leeds, 1996.