NeuroImage **16,** 295–316 (2002)
doi:10.1006/nimg.2002.1093, available online at http://www.idealibrary.com on IDEAL®

Shape-Based Cortical Surface Segmentation for Visualization Brain Mapping

Kevin P. Hinshaw, Andrew V. Poliakov, Eider B. Moore, Richard F. Martin, Linda G. Shapiro, and James F. Brinkley

Structural Informatics Group, Department of Biological Structure, University of Washington, Seattle, Washington 98195

Received September 19, 2001

We describe a knowledge-based approach to cortical surface segmentation that uses learned knowledge of the overall shape and range of variation of the cortex (excluding the detailed gyri and sulci) to guide the search for the grey-CSF boundary in a structural MRI image volume. The shape knowledge is represented by a radial surface model, which is a type of geometric constraint network (GCN) that we hypothesize can represent shape by networks of locally interacting constraints. The shape model is used in a protocol for visualization-based mapping of cortical stimulation mapping (CSM) sites onto the brain surface, prior to integration with other mapping modalities or as input to existing surface analysis and reconfiguration programs. Example results are presented for CSM data related to language organization in the cortex, but the methods should be applicable to other situations where a realistic visualization of the brain surface, as seen at neurosurgery, is desired. © 2002 Elsevier Science (USA)

1. INTRODUCTION

A common step in many brain mapping and surgical planning techniques is reconstructing the cortical surface from a structural MRI dataset. Such reconstructions are essential for surgical planning, for visualization of functional data such as fMRI, and for surface reconfiguration methods such as inflation or flattening.

In particular, accurate representation of the cortical surface is essential for a technique we have developed called visualization-based brain mapping (Modayur *et al.*, 1997). This technique was developed to recover 3-D locations of cortical stimulation mapping (CSM) language maps, which are gathered during neurosurgery for intractable epilepsy. The primary record of each map is a photograph, taken during surgery, in which sterile numbered tags on the patient's brain mark the locations of mapped sites. Using the photograph as a reference, an operator must locate these sites on a 3-D, MRI-derived graphical rendering of the patient's brain. Prominent landmarks on the rendering help the human operator to match positions in the 2-D photograph with the 3-D reconstruction.

The primary requirement for this method is a surface reconstruction that accurately represents the cortical surface as seen during neurosurgery. In recent years several excellent algorithms have been described for automatically or semi-automatically extracting the cortical surface. For example, MacDonald *et al.* describe an automatic multi-resolution surface deformation technique called ASP (Anatomic Segmentation using Proximities), in which an inner and outer surface are progressively deformed to fit the image, where the cost function includes image terms, model-based terms, and proximity terms (MacDonald *et al.*, 2000). Dale *et al.* describe an automated approach that is implemented in the freely-available FreeSurfer program (Dale *et al.*, 1999; Fischl *et al.*, 1999) This method initially finds the grey-white boundary, then fits smooth grey-white and pial surfaces using deformable models. Van Essen *et al.* describe the SureFit program (Van Essen *et al.*, 2001), which finds the cortical surface running midway between the grey-white (inner) boundary and the grey-CSF (outer) boundary. This mid-level surface is created from probabilistic representations of both inner and outer boundaries that are determined using image intensity, intensity gradients, and knowledge of cortical topography.

The surfaces generated by these programs are exquisite, and because they are all topologically correct, can be used as the basis for surface reconfiguration techniques such as inflation, mapping to a sphere, or flattening (Carman *et al.*, 1999; Fischl *et al.*, 1999; Van Essen *et al.*, 2001), which can in turn be used to visualize functional data, or to normalize different brains. In our own work we have begun to utilize some of these programs for further surface analysis.

However, in our experience these methods generally do not represent the surface of the living brain, as seen during neurosurgery, because they were not designed to do so. In our experience, the most reliable reconstruction approach for the exposed brain surface (other

1053-8119/02 $35.00
© 2002 Elsevier Science (USA)
All rights reserved.

than purely manual contour tracing) is simple isosurface following (Lorensen and Cline, 1987) on an image volume in which the cortical volume has been cleanly segmented from the surrounding tissues, since the isosurface presumably extracts what is actually visible. On its own the extracted isosurface cannot be the basis for further surface analysis, since the extracted surface mesh generally is not topologically correct and the deep sulci are not visible. However, once the CSM sites are located within the 3-D image volume they can be input to programs such as FreeSurfer for further analysis.

The key to an accurate isosurface is a cleanly-segmented cortex, in which the skull and related tissues have been stripped away. This "skull-stripping" step can be done by hand (Lancaster *et al.*, 1999), but the task is tedious and time-consuming. Automating the process is nontrivial, though. Simple thresholding fails because the voxel intensity range for the cortex overlaps with those for surrounding tissues. Region-growing methods have difficulty because the gap between the cortex and the inner scalp is frequently bridged by small areas of bright voxels. This problem is often addressed by combining 3-D region growing with mathematical morphology (Davatzikos and Bryan, 1996; Sandor and Leahy, 1997). Our previous system used this combination of methods, but it required considerable hand-tuning of its cost function parameters—a significant hurdle for most users. Furthermore, errors are difficult to correct in region-based methods; generally the user must adjust parameters that have global effects and hope for the best.

The other common method for skull stripping is the deformable model. Deformable models pose segmentation as an optimization problem, finding a surface that fits the underlying image data while maintaining a certain degree of local smoothness (Kass *et al.*, 1987). Among the examples that use a coarse deformable model to isolate the cortex prior to more detailed segmentation are FreeSurfer, the Brain Extraction Tool (BET) (Smith, 2000) and ASP. However, these methods suffer the same drawbacks as region-based methods: global parameters that are difficult to tune and a lack of facility for correcting errors locally. The latter problem is particularly relevant for our application, because abnormal brain anatomy is common among the surgical patients for which the system is used.

In this paper we describe a semi-automatic skull stripping technique that employs a different paradigm than region growing or deformable models. This approach uses learned shape knowledge of the cortical "envelope" (the general shape of the cortex, disregarding the detailed gyri and sulci (Toga, 2001)) to guide the search for the grey-CSF boundary, then combines boundary information with the shape knowledge to constrain the search for boundaries in other parts of the image volume. When the automatic algorithm makes mistakes, the user can correct the errors by interactively adjusting the boundary. The approach thus combines automatic methods with intuitive user controls to give an accurate cortical envelope. This envelope is then used to remove nonbrain structures so that isosurface following can be used to reconstruct the cortical surface, veins, and arteries. The skull-stripping algorithm is implemented within a workflow-based software architecture for visualization-based mapping.

The integrated system, which we call the Visual Brain Mapper (VBM), is used routinely in the UW Human Brain Project to integrate CSM data with nonsurgical language measures such as fMRI. To-date over 40 neurosurgical patient datasets have been mapped, most by a nonprogrammer neuroscientist. The techniques and software architecture, while developed initially for CSM of language areas in the cortex, should be applicable to other brain mapping and visualization problems as well.

2. METHODS

2.1. Representing the Cortical Envelope

2.1.1. Geometric constraint networks. Our approach to skull stripping uses a representation we call the radial surface model (RSM) to represent the shape and range of variation of the cortical envelope. The RSM is an example of a Geometric Constraint Network (GCN), which we have proposed as a potential representation for spatial knowledge of anatomical shape and shape variation (Brinkley, 1992). The central hypothesis of the GCN representation is that local constraints between structures or parts of structures, when interacting together in a constraint satisfaction process, can capture the global shape and range of variation of a particular shape class without requiring an explicit global shape model.

A GCN is an instance of a constraint network, which can be defined as a graph, in which the nodes V_1–V_n represent variables and the arcs C_{ij} represent constraints between pairs of variables (Mackworth, 1977). The range of possible values for a variable V may be described as a discrete list of scalars or vectors, or in the case of continuous ranges, by intervals or probability distributions. A constraint C_{ij} between two variables V_i and V_j specifies which possible values of the two variables are compatible with each other. A solution to a constraint network is one possible value per variable such that all the pairwise constraints are simultaneously satisfiable. We call such a solution a "coherent instance." The set of all coherent instances represents the complete solution to the constraint satisfaction problem specified by the constraint network.

A geometric constraint network (Fig. 1) is a constraint network in which the variables V in the network represent physical objects or parts of objects, the

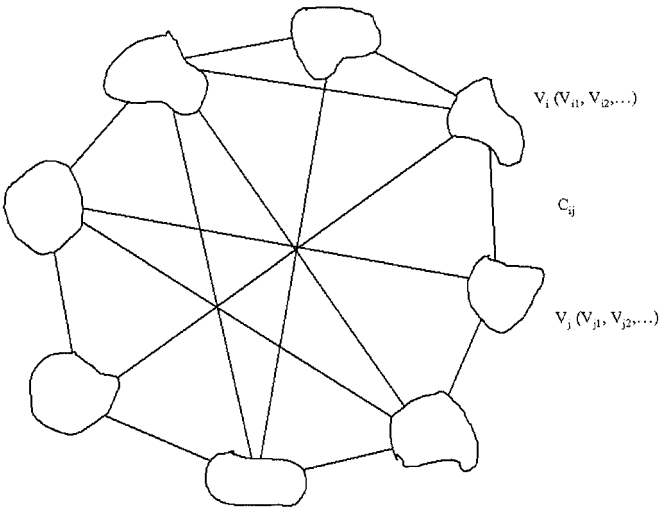


FIG. 1. A geometric constraint network is a constraint network in which the variables *V* represent objects in space or points on the surface of an object, the possible values for each variable represent the possible locations of the object (shown as "clouds" in the figure), and the constraints *C* represent geometric constraints (such as distance ranges) between pairs of objects.

possible values for each variable represent the possible locations of each object in space, also called the accessible volume, and the arcs represent geometric constraints between the objects. If the variables represent locations on the surface of an object such as an organ, and the constraints represent allowed spatial relationships such as distance ranges, then we hypothesize that the set of all coherent instances of the associated constraint network represents the set of instances of the shape class defined by the GCN. If this hypothesis is true or even partially true, then the GCN representation should be of use in shape matching, in which an unknown shape is classified by determining how well it satisfies the GCNs representing various reference shape classes.

The GCN representation should also be of use in model-based image segmentation, in which the accessible volumes of the nodes in the GCN can provide search regions for low-level image processing operators, the results of which can be folded back as additional constraints in order to further reduce the accessible volumes. When the accessible volumes are small enough the remaining set of coherent instances represents those instances of the shape class that are compatible with the image data. It is this application of GCN's that we use for shape-based skull stripping.

This approach to model-based image segmentation has many similarities to the currently popular deformable models, in which a single instance is deformed according to a cost function that includes both image-based and (usually) global shape constraints such as smoothness. The difference is that in the GCN approach all instances of a given shape class that are compatible with any additional image-related constraints are implicitly represented, whereas in the deformable model only a single instance is represented at any one time. The inclusion of all possible instances allows for an exclusion paradigm for structure determination (Altman, 1990), in which instances are gradually excluded as image-related constraints are found, and the set of instances at any given time can be used to constrain the search for additional image-related features. In contrast, the deformable model approach uses an adjustment paradigm, in which a single instance is gradually adjusted in order to minimize a cost function. Because the representation does not maintain explicit shape variation, the adjustment approach does not permit model-guided search for additional image features of interest, nor does it permit shape-based matching. In addition, without adequate starting structures, the adjustment approach can become caught in local minima that represent incorrect shapes.

The main problem with the GCN approach is that without efficient representations it is computationally intractable to precisely maintain all instances of a given shape class. In previous work we developed specializations of GCNs that are computationally tractable and have been shown to be *potentially* useful for specific cases, including organ volume determination from 3-D ultrasound (Brinkley, 1985), protein structure determination from constraints given by nuclear magnetic resonance spectroscopy (Brinkley *et al.*, 1988), 2-D image segmentation of structures critical to find for radiation treatment planning (Brinkley, 1993; Hinshaw *et al.*, 1995), 3-D structure determination from MRI (Hinshaw and Brinkley, 1997) and 2-D shape-based matching (Brinkley, 1993). The current paper describes the first application of GCNs that has been shown to be actually useful in a working application that is used by nondevelopers.

2.1.2. The radial contour model. The computationally tractable representation we use in the Visual Brain Mapper (VBM) is called the radial surface model (RSM, section 2.1.3), which is itself built from a set of parallel radial contour models (RCMs, Fig. 2). Each of these models comprises a structural component that defines the variables (or nodes) *V* in the GCN and a shape component that defines the constraints *C*.

The structural component of the RCM is defined by a radial contour. A radial contour is a closed polygon parameterized by polar coordinates $r(\theta)$. The points forming the contour can be thought of as the tips of spokes, or *radials*, emanating from a central point inside the contour.

The exact position and orientation of these radials in an image are determined by a *reference axis* for the contour. This axis is typically aligned either to an axis of symmetry or to the direction in which a shape is

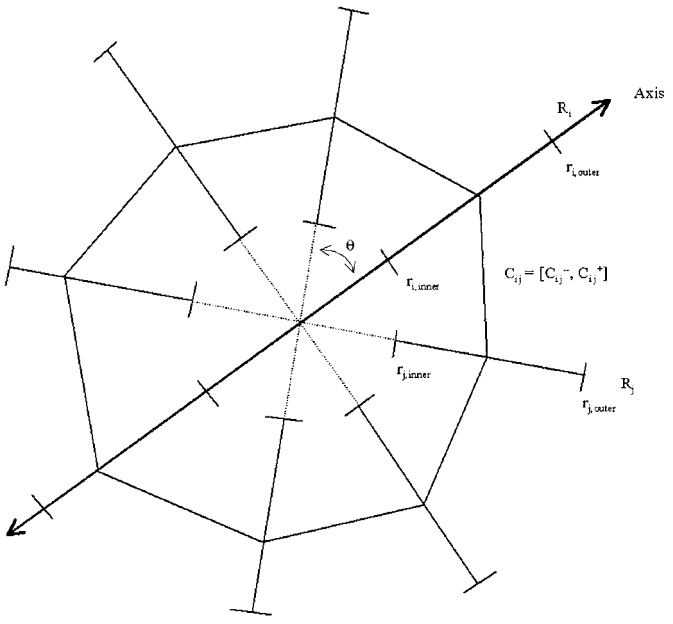


FIG. 2. A radial contour model, or RCM. A RCM is a GCN in which the variables are points on the cross-sectional contour of a structure, the possible values of the variables are constrained to lie in intervals along a set of fixed radials R emanating from the origin of a local coordinate system defined by a reference axis, and the constraints C delimit the range of ratios of neighboring radials as observed in a training set.

most elongated. The endpoints of the reference axis touch the object's boundary, and its midpoint provides the central point from which other radials extend. Since radials are specified relative to it, the reference axis provides a local coordinate system for the radial contour.

Some additional limits increase the tractability of the radial contour representation. Radials are restricted to be evenly spaced, which means the angle between any two adjacent radials in a contour is the same. It is also assumed that each radial will intersect the contour exactly once; contours meeting this constraint are sometimes called *star-shaped*. These restrictions eliminate self-intersections. In addition, the connectivity of the polygon becomes implicit, with its edges formed by proceeding sequentially along the endpoints of the radials. The primary disadvantage of these simplifications is that they restrict the range of shapes that the radial contour can represent. Yet for shapes such as coronal sections of the cortical envelope, star-shaped approximations will frequently suffice.

By sacrificing the ability to represent every possible shape, we gain the critical ability to compare multiple shapes. By using a predetermined layout of radials relative to the reference axis, this representation provides automatic registration between radial contours. In effect, the reference axis factors out issues of a structure's location and orientation, leaving only issues of scale and shape variation to be accounted for.

Once the reference axis has been determined for two structures, the correspondences between all of their radials will also be known, thereby eliminating θ as an unknown. Therefore the variables V defined by the structural component of the RCM are the radials, and the possible values (accessible "volumes") of each variable are given by the possible radial distances r_i.

Given the structural component, the shape component of the RCM becomes tractable because the registration of radial position and orientation permits concise comparison between corresponding boundary regions. The shape component defines the constraints C between radials, and is given by the range of ratios of neighboring radial lengths, as observed on a training set. (In other work we looked at alternative representations for the constraints, with no major improvement to the results, but at a larger computational cost (Altman and Brinkley, 1993)).

For a pair of neighboring radials i and j with lengths r_i and r_j, the ratio is simply $s_{ij} = r_i/r_j$. The definition of "neighbor" varies. In this application, again for computational tractability in the 3-D case, we define neighboring radials to be those that are adjacent to each other.

The constraint C_{ij} between the pair of neighboring radials i and j is defined by a lower bound C_{ij}^- and an upper bound, C_{ij}^+, which correspond to the minimum and maximum values observed in the training set for ratio s_{ij}.

The constraints of the RCM capture information about relationships between radials, rather than information about the radials in isolation. This makes it difficult to draw a picture of the constraints. An instantiated RCM, shown in Fig. 3, is easier to visualize. An instance of an RCM can be derived from a reference axis; the two halves of the axis provide lengths for two of the contour's radials. These lengths can be used in conjunction with the RCM's shape constraints to derive *uncertainty bounds* (shown shaded in the figure) for the remaining radials. These bounds, which are the RCM version of the accessible volumes for the nodes in the GCN, delineate the range of lengths that each radial can have while still satisfying the shape constraints of the model. The uncertainty bounds, which gradually decrease as new constraints are introduced, can be used to limit the search for edges in a segmentation algorithm.

The uncertainty bounds are determined by a constraint propagation algorithm. Suppose that radial i is initialized to have length r_i. For each of its neighbors j, the constraint $C_{ij}^- \le s_{ij} \le C_{ij}^+$ can be used to infer that $C_{ij}^- r_i \le r_j \le C_{ij}^+ r_i$. The neighbors of i are now bounded, because the constraints have been used to rule out values that were not observed in the model's training set. The updated neighbors can be used in turn to bound *their* neighbors, creating a wave of updates that propagates through the entire contour. Essentially, the

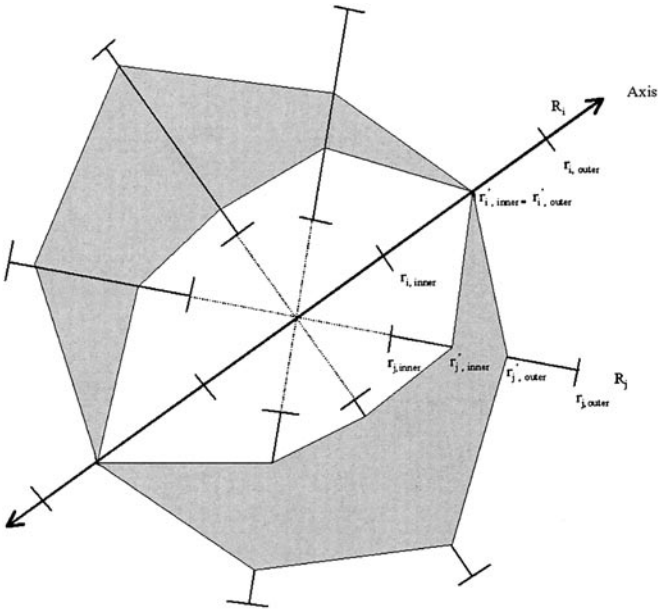


FIG. 3. Constraint propagation in the radial contour model. Data (either from image edges or input by the user) allows the uncertainty along some radial R_i to be set to 0 ($r'_{i,inner} = r'_{i,outer}$) from its initial interval [$r_{i,inner}$, $r_{i,outer}$]. The constraint C_{ij} between radial i and radial j can then be used to reduce the uncertainty interval along radial R_j from its initial value [$r_{j,inner}$, $r_{j,outer}$] to a new interval [$r'_{j,inner}$, $r'_{j,outer}$]. This reduced interval can then be used to reduce the uncertainty for its neighbor, but by a lesser amount. The result is a series of waves of propagation from data sources (in this case the two pole radials) that cancel each other out when they meet. Once the process is complete, the further away a radial is from a radial defined by data, the less is known about the location of the contour boundary, but the shape constraints ensure that at least something is known. The remaining uncertainty can be used to constrain the search for edges for those radials which have not yet been examined.

further away from data, the less is known about the contour, but something is still known because of the shape knowledge encoded in the constraints.

Figure 4 shows pseudocode for the propagation algorithm. This code is an example of a network consistency algorithm (Mackworth, 1977) that is often used in constraint satisfaction problems to reduce the number of solutions compatible with the constraints prior to exhaustive search for all possible solutions. In previous work (Brinkley, 1985) we showed that this algorithm has an average-case complexity of *O(n)*. In other words, when information is acquired about a radial's position, all necessary interval updates can be made in time proportional to the number of radials in the model. This fast execution time permits the use of the algorithm in interactive applications.

2.1.3. The radial surface model. To handle full 3-D objects, the radial contour model is generalized to a Radial Surface Model (RSM). As Fig. 5 illustrates, a radial surface is a stack of slices. The center points for the slices are collinear, forming an axis that runs perpendicular to all the slices. At given intervals along the

```
procedure propagate(float length, int k, Shape Constraints C,
                    Radial Contour R) {
    Q = new(Queue)
    R[k].lo = R[k].hi = length
    enqueue(Q,k)
    while notEmpty(Q)
        i = dequeue(Q)
        for each j in neighbors(i)
            d.lo = R[i].lo * C[i][j].lo
            d.hi = R[i].hi * C[i][j].hi
            e = intersect(R[j], d)
            if (e != R[j])
                R[j] = e
                enqueue(Q,j)
}
```

FIG. 4. The RCM-model's propagation algorithm. Given a measurement `length` for radial `k` of radial contour `R`, the algorithm uses the shape constraints `C` of the RCM model to tighten the uncertainty bounds for contour `R`. `Q` is a queue of radials to be processed, `R[k].lo` and `R[k].hi` define the upper and lower bounds for the interval along radial `k` in which the boundary of radial contour `R` is assumed to lie. The upper and lower bounds for radial `k` are set equal to the input `length`, and the queue is initialized with radial `k`. The queue is then processed until it is empty. For each radial `i` removed from the queue, each of its neighbors is considered in turn. For each neighbor `j` of radial `i`, a candidate interval `d` is computed from the interval at radial `i` and the constraint `C` between radials `i` and `j`. The intersection `e` between the original interval `R[j]` and the candidate interval `d` is then computed. If this intersection is not the same as the original interval at radial `j`, the interval at radial `j` is set to the intersection, and radial `j` added to the end of the `Q`. The algorithm terminates when no more radials have changed and the queue is therefore empty.

axis, radials are extended outward in the slice plane to the surface boundary. Each surface also has a local coordinate system that describes its orientation within a 3-D space. The coordinate system is derived from landmarks that provide translation, scale, and rotation information. As with the reference axis for the 2-D model, these landmarks can be chosen arbitrarily, but they generally correspond to extremal points or axes of symmetry. For the cortical envelope, the landmarks of the Talairach coordinate system (Talairach and Tournoux, 1988) are used. Structurally, the radial surface is equivalent to a stack of radial contours, with two "pole" radials defining the endpoints along the reference axis.

As with the radial contour, certain features of the

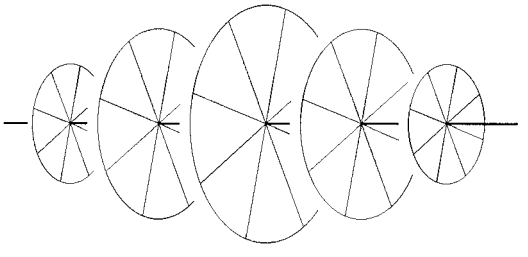


FIG. 5. The radial surface model. A series of parallel slices perpendicular to a reference axis, with evenly-spaced radials extending out from the center of each slice to the surface boundary.

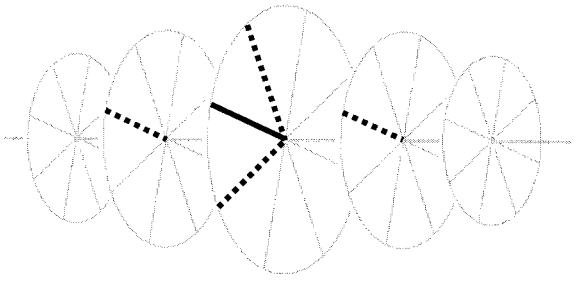


FIG. 6. A radial and its four neighbors (shown with solid and dotted lines, respectively).

radial surface are simplified to ease the problem of registration. The angle between radials in a slice is kept constant, and slices themselves are evenly spaced. It is also assumed that each radial will intersect the surface only once. Despite these limits on the class of describable shapes, the radial surface representation is flexible enough to be used for shape comparison and for guidance of low-level segmentation methods.

User interface considerations strongly influenced this stack-of-slices layout. User-assisted reconstruction of volume data is essentially an exercise in 3-D sculpting, a task which poses significant interface challenges and remains an open research problem (Markosian *et al.*, 1999). Yet the reconstruction task is even more difficult than free-form sculpting—the goal, after all, is getting a close match between the model and the volume data. But how do you display a surface embedded in 3-D data in a meaningful way? By using a stack of slices, this difficult 3-D interface question was sidestepped, because each surface slice can be superimposed over its corresponding image slice and manipulated as a standard 2-D contour.

Naturally, a radial surface contains significantly more radials than a radial contour. For example, the cortical envelope can be sampled reasonably with 600 radials—20 slices of 30 radials each. Given this increase, performance was a concern when deciding which neighborhood model to use. Due to its lower computational complexity, the RSM model implemented in VBM uses a local neighborhood model. Each radial has four neighbors—the two next to it within its slice, plus the nearest radial on each of the adjacent slices. In this way, shape information can propagate both within a slice and between slices. Figure 6 shows the arrangement. Parallel implementations of this algorithm could use a more extensive neighborhood model, in which every radial is a neighbor of every other radial.

2.2. Finding the Cortical Envelope

The RSM is implemented in a module called 3-D Scanner, which in turn is implemented as part of the cortex segmentation component of VBM (Hinshaw and Brinkley, 1997). Figure 7 shows a screen shot of the 3-D Scanner interface. The interface displays three orthogonal slices through the MRI volume dataset: a coronal view on the left, an axial view in the middle, and a sagittal view on the right. On each view, draggable crosshairs control the position of the other two slices. Using these crosshairs, the user can browse quickly along any axis of the volume.

A primary component of the 3-D Scanner interface is a surface editor for building radial surfaces. This editor is used both for manual surface creation (to generate training examples) and for correcting mistakes made during the automatic segmentation stage. During surface editing, the current model slice is superimposed over its corresponding coronal image (top-left corner in Fig. 7), where the user may adjust radial lengths by dragging along given radials. In a separate viewport, Scanner also shows a 3-D view of the surface, along with the current image slice and a box representing the bounds of the volume dataset. Camera controls allow users to adjust the view of the surface.

The RSM used by 3-D Scanner has evolved over time. At first, it was built from four radial surfaces that were drawn by hand with the surface editor. Each time a new data set is segmented with the system, its cortical envelope can be added to the available training set. The RSM currently in use was derived from sixteen examples with normal shape, as determined by the system's primary neuroanatomist user (author R.M.).

Given an image volume and RSM, the primary steps involved in finding the cortical envelope prior to skull stripping are: (1) define landmarks, and (2) search and propagate.

2.2.1. Define landmarks. Landmarks are used to define the reference axis for the RSM, thereby allowing corresponding boundary points to be compared in multiple datasets. For brain segmentation, Scanner uses the landmarks of the Talairach coordinate system (Talairach and Tournoux, 1988). A special interface was designed to help the user enter these landmarks for a volume dataset.

1. The midsagittal plane. (See Fig. 8a.) Using the MRI crosshairs, the user selects the sagittal slice that is closest to the midsagittal plane. This method assumes that the midsagittal plane aligns with the sagittal slices of the MR scan. This assumption has been a reasonable one thus far, because a head restraint holds the patient's head in alignment with the scanning equipment. (If needed, though, a more flexible interface could be built to accommodate a tilted midsagittal plane.)

2. The anterior and posterior commissures (AC and PC). (See Fig. 8b.) The user drags the MRI crosshairs until one of the commissures is visible on the midsagittal plane, then clicks on the location to mark it. The points can be moved on the MRI slices to fine-tune

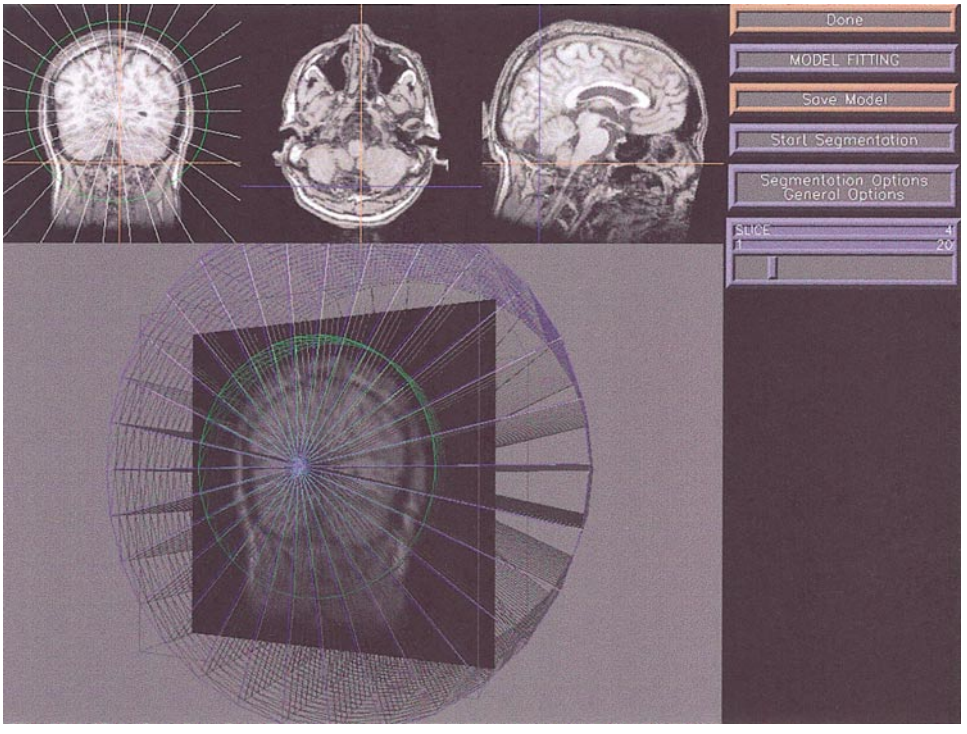


FIG. 7. 3-D Scanner's interface for brain segmentation. Three orthogonal slices through the volume data appear at the top. Crosshairs on the images mark the locations of the other two slices. The 3-D view in the lower left displays the current radial surface model instance within the image volume data's bounding box. In this case the radials are set to their original bounds before any search or constraint propagation has taken place. The RSM slice currently being viewed is superimposed on the coronal plane at the top left, and the corresponding image slice is displayed in the surface view for reference. The menu, which controls the workflow, appears at the right.

their positions; constraints are used to keep them on the midsagittal slice. These two points determine the AC–PC line, which is used as the reference axis for the radial shape model of the brain.

3. A bounding box. (See Fig. 8c.) The user positions a bounding box around the cortex, which just includes the top of the parietal lobe, the bottom of the temporal lobe, the back of the occipital lobe, the front of the frontal lobe, and the side of the parietotemporal lobe. The box has one axis aligned to the AC–PC line, the second perpendicular to AC–PC in the midsagittal plane, and the third perpendicular to the midsagittal plane. The user specifies the first four of these bounds using a rectangle on the sagittal MRI cross-section. The lateral bounds are indicated by adjusting lines on the coronal and/or axial cross-sections. It is worth noting that these positions cannot be found with any single MRI slice; the user must scroll through the different cross-sectional views to find the true extremal points. The interface makes it easy to scan quickly through the slices and verify that the true extremal points have been selected.

Once these landmarks have been specified, they are used to initialize the RSM, as shown in Figs. 7 and 9. The AC–PC line provides the model's reference axis. The back-to-front dimension of the bounding box is used to compute both the endpoints of the reference axis, which define the "pole" radials, and the proper spacing between model slices.

2.2.2. Search and propagate. At this point the user could initiate the automated segmentation algorithm by pressing the "Start Segmentation" button. For illustration purposes, Fig. 10 shows just the first step in that process: the user has accessed the "Segmentation Options" menu to turn off automated segmentation, and has clicked the "Start Segmentation" button. This action initiates only constraint propagation from the pole radials, whose distances are given by the bounding box. As shown in the figure, the resulting uncertainty for the non-pole radials is minimal in the vicinity of the poles, but becomes larger the further away a radial is from data. The greatest uncertainty is therefore at the midcoronal slice.

Once propagation from the poles has completed the user can manually indicate one or more "hint" radials (upper left of Fig. 10). These hint radials provide additional constraints which help guide the search for additional edges.

Following input of the hint radials, the user can turn automatic segmentation back on and click again on the "Start Segmentation" button. At this point the automatic search and propagate algorithm is executed. The

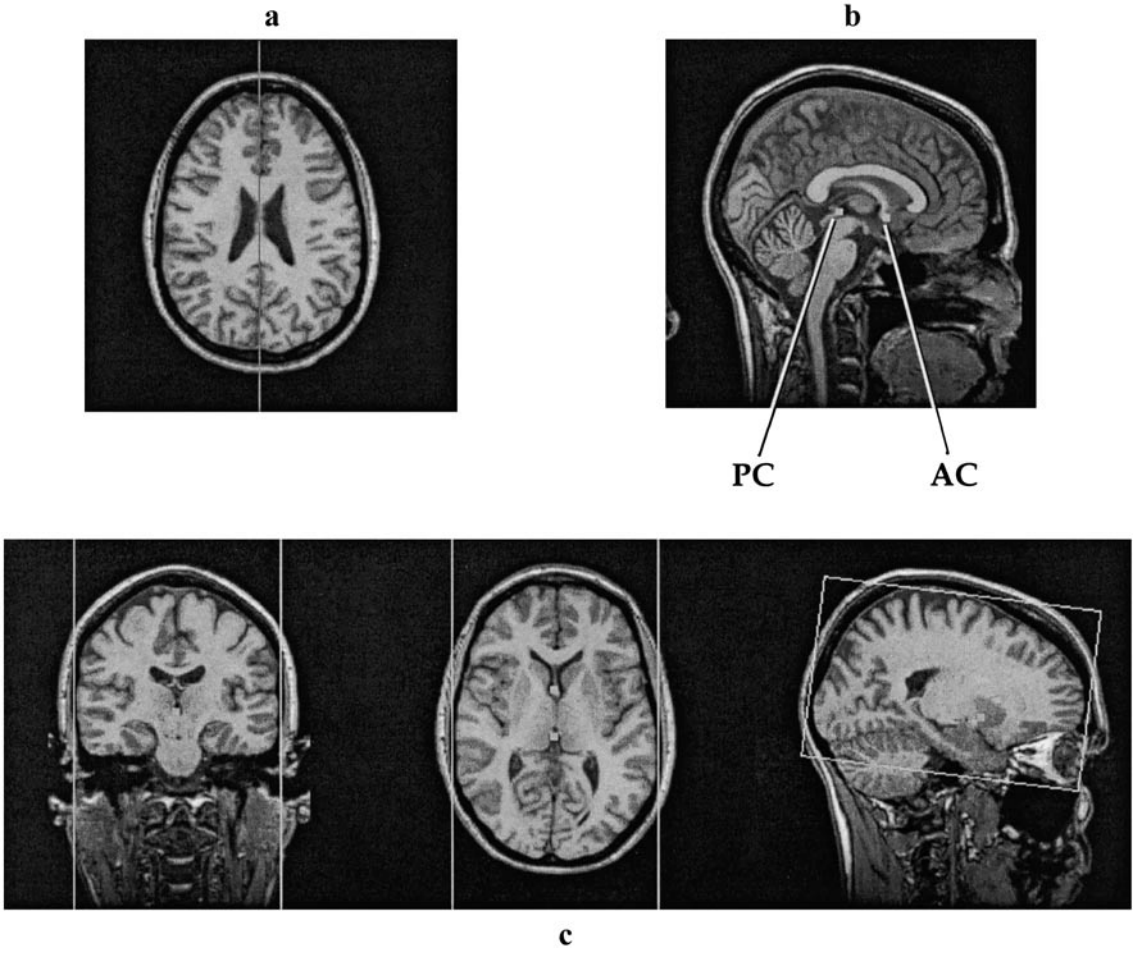
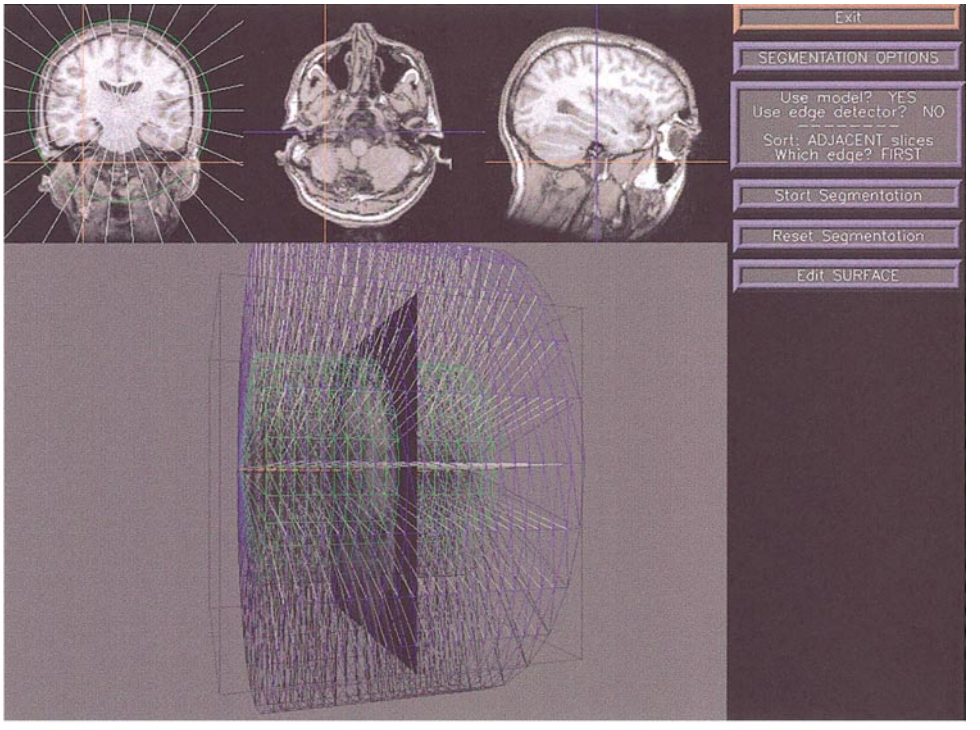


FIG. 8. The landmarks for the Talairach coordinate system. (a) The midsagittal plane. (b) The anterior and posterior commissure points. (c) The bounding box.

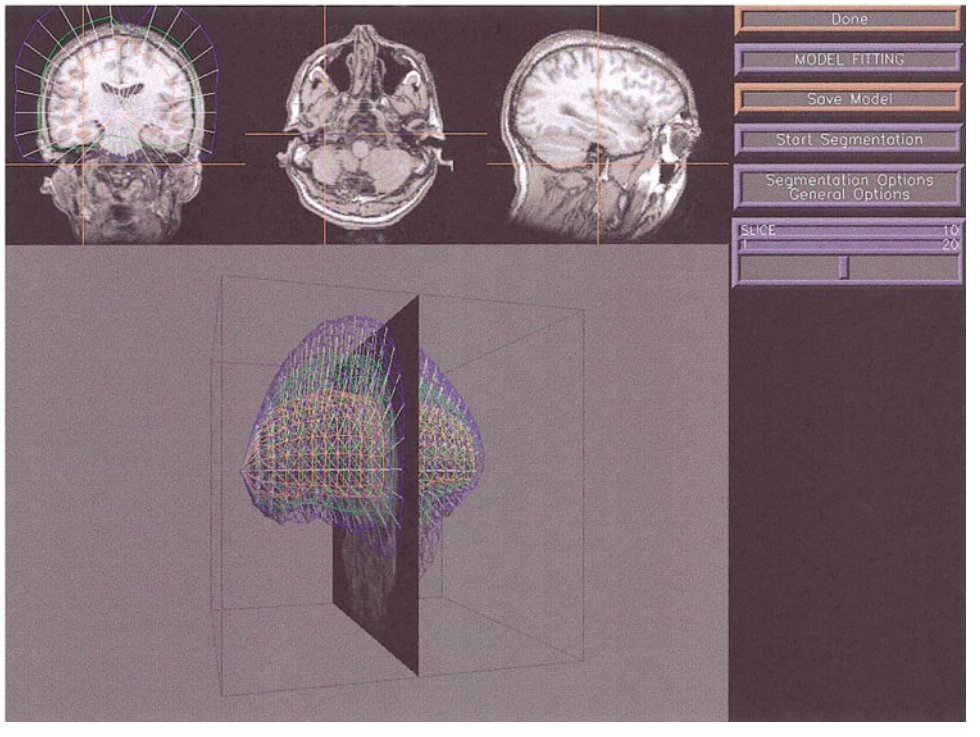
hint radials are propagated, after which the system chooses one or more radials, finds the best edge along each of the selected radials, uses the found edges to set the selected radial lengths, and reruns the constraint propagation procedure with the updated radials as input. This alternating search-and-propagate loop continues until all edges have been found or until the remaining uncertainty bounds are below a given threshold.

The decisions as to which radials to search next, and which edge to choose along a given radial, are made using heuristics that include additional knowledge of the problem domain. Different heuristics are encoded in different edge detectors that may be swapped in and out to evaluate alternate search strategies. We have experimented and are continuing to experiment with many different heuristics. The current strategy is as follows:

1. Place the slices for which the user has indicated hint radials on a queue.
2. Select the first slice in the queue or quit if there are no more slices.
3. Select the first radial in the slice that is not already set by the user and is not indicated as a radial to be ignored. Ignored radials are heuristically set to be those for which edges are difficult to find and which are not needed for our application. For the Visual Brain Mapper we ignore radials in the inferior part of the brain since we are mainly interested in the left temporal lobe, letting the shape constraints set the edge for these radials to be the middle of the search region.
4. For the selected radial, find the distance along the radial to all local image gradient extrema within the one-dimensional search region defined by the shape constraints. This list constitutes the potential edges along the radial.
5. Prune the list of possible edges using a set of additional constraints. These constraints include distance from the outer skull, size of the gradient at the candidate edge, and similarity of the intensity at the candidate edge to edge intensities already found at the radial neighbors.
6. If only one candidate edge remains set the radial distance to the value found at this edge, and add the



9



10

FIG. 9. Rotation of the surface view. Clicking the General Options menu item (Fig. 7) brings up a menu which controls options such as viewpoint, rendering style, colors, and objects to display. In this screenshot the user has accessed the General Options menu to rotate the initial RSM instance.

FIG. 10. Initial propagation. To illustrate the effect of constraint propagation without any edge detection the user has accessed the Segmentation Options menu to turn off automatic edge detection, then hit the Start Segmentation button to initiate constraint propagation from the pole radials, whose values were given by defining the bounding box (Fig. 8c). With data only from the landmarks and the pole radials the propagation process has generated a reasonable guess as to the location of the cortical envelope, with greatest uncertainty in the midcoronal section, which is furthest from the pole radials. The user has also indicated the location of the surface on a single radial (upper left) prior to starting the automatic search and propagate procedure.

radial to the queue of radials to be propagated. If no edges remain or if more than one remain ignore this radial until others have been searched.

7. Repeat steps 3–6 for all radials within the selected slice, then propagate the edges throughout the model using the shape constraints.

8. Place slices adjacent to the current slice on the slice queue if they are not already there.

9. Go to step 2 to select the next slice.

Figure 11 shows the result after the automatic search and propagate loop has completed. The uncertainty in the radial surface has been greatly reduced, and now mostly matches the cortical envelope. The system will occasionally make errors if an incorrect edge is found, as shown the upper left of the figure. In this case the user simply drags any radials with errors to the correct locations, resulting in the corrected surface shown in Fig. 12.

2.3. Integration in a Visualization-Mapping Protocol

The shape-based procedure for finding the cortical envelope is integrated within the visualization mapping protocol described earlier for mapping CSM language data onto a 3-D anatomical model (Modayur *et al.,* 1997). The current protocol differs from the previous protocol not only in its use of the shape-based model rather than 3-D region growing, but also because all steps in the protocol are implemented within a workflow architecture, as described in section 2.4.

The current protocol is shown in Fig. 13. The steps in the protocol are shown in the boxes, with arrows defining the flow of data between steps. Most of the intermediate data are stored as files. Several steps have substeps, which are shown as expansions to the right of the parent step. For example, the steps in finding the cortical envelope—3a1 (Define Landmarks) and 3a2 (Search-and-Propagate)—are substeps of step 3a (Find Cortical Envelope), which in turn is a substep of step 3 (Segment Cortex). (Step numbers are only shown for clarity of discussion, they are not used in the actual program.) The following sections describe these steps, with emphasis on those that are different than our previous work (Modayur *et al.,* 1997).

2.3.1. Input data. The input to the visualization mapping protocol consists of CSM data and image data.

The CSM data are acquired during neurosurgery for temporal lobe tumors or intractable focal epilepsy (Ojemann *et al.,* 1989). Since language function is located primarily in the temporal lobe, and since the location of language areas in the temporal lobe varies among individuals, it is necessary to map these language areas in order to avoid them during the surgical resection. The resulting CSM maps, in addition to their clinical use in planning the resection, are a valuable source of data for understanding language organization in the brain.

The mapping proceeds as follows. After a portion of the skull has been removed to expose the cortical surface, the patient is awakened (but kept under local anesthesia). The patient is then asked to perform a simple, language-related task. An example task is object naming, where the patient is shown pictures of common objects—a chair or a dog, for example—and asked to name them. For each repetition of the task, a different site on the surface of the brain is stimulated with a mild electrical current. Sites where the stimulation disrupts the task are deemed essential for language function, and therefore must be avoided during resection to prevent permanent language impairment. The locations of the stimulation sites are determined by placing small numbered tags on the exposed cortical surface. A photograph of these tags (Fig. 14) records their locations. The task of the visualization mapping procedure is to determine the 3-D locations of these tags with respect to the patient's brain anatomy.

The image data provide the information needed to reconstruct the patient's brain anatomy. During the week prior to a patient's surgery, three image series are acquired using a whole body 1.5 Tesla MR scanner: one for surface anatomy, one for veins, and one for arteries. Figure 15 shows samples of each type of scan, along with details about the imaging parameters used.

After acquisition, the three image datasets, together with the CSM data, are transferred to a local database, where they form the input to the visualization mapping protocol.

2.3.2. Steps 1 and 2: Select patient and align image volumes. The first step in the protocol shown in Fig. 13 is to select the patient dataset to work with. Following patient selection the three sets of MR image volumes (cortex, veins and anatomy) are aligned and resampled to contain 256^3 uniform-sized voxels, such that a given voxel in all three datasets corresponds to the same location in MR machine coordinate space. When patient movement is not an issue (the usual case, since patients are immobilized during image acquisition) the image headers are used to do the alignment. When patient motion cannot be ignored, the operator performs a manual alignment of the datasets, using the Register tool developed at the Montreal Neurological Institute (MacDonald, 1993). This tool lets the user place landmarks in the datasets by hand in order to specify correspondence points. The tool then performs a linear transformation in order to bring the datasets into alignment.

2.3.3. Step 3: Segment cortex. Extraction of the cortical surface from the aligned cortical MRI dataset begins by using the shape-based procedures described in section 2.2 to find the cortical envelope (step 3a and substeps 3a1 and 3a2 in Fig. 13), then using the result-

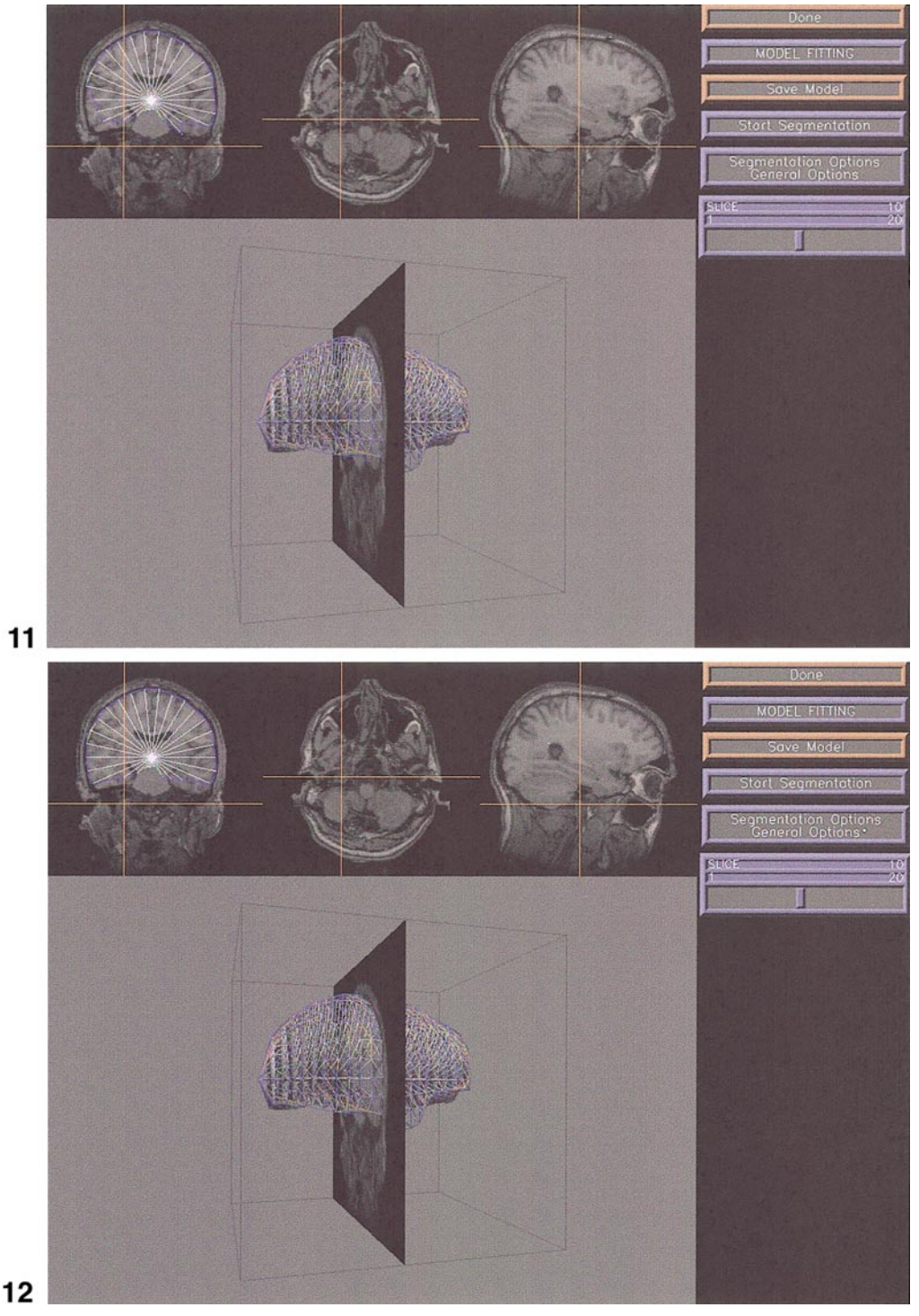


FIG. 11. Final result of automatic processing, showing a segmentation error. The user turned on the automatic segmentation option (using the Segmentation Options menu) after constraint propagation from the poles and input of a single hint radial, as shown in Fig. 10. The user then clicked on the Start Segmentation button, which initiated the search and propagate algorithm: the system chooses a radial, finds an edge along that radial, propagates that edge throughout the constraint network, and chooses another edge to search. The process completes when no more radials are left to search, or when the remaining uncertainty is below a threshold. The procedure will sometimes results in errors, as shown in the upper left of the figure.

FIG. 12. Fixing an error. The user fixes errors in the automatic segmentation process, like that shown in Fig. 11, by simply clicking the mouse on the radial with the error, and dragging to the desired location. The radial is flagged as being set by the user, and is therefore not changed during further invocations of the search and propagate algorithm.

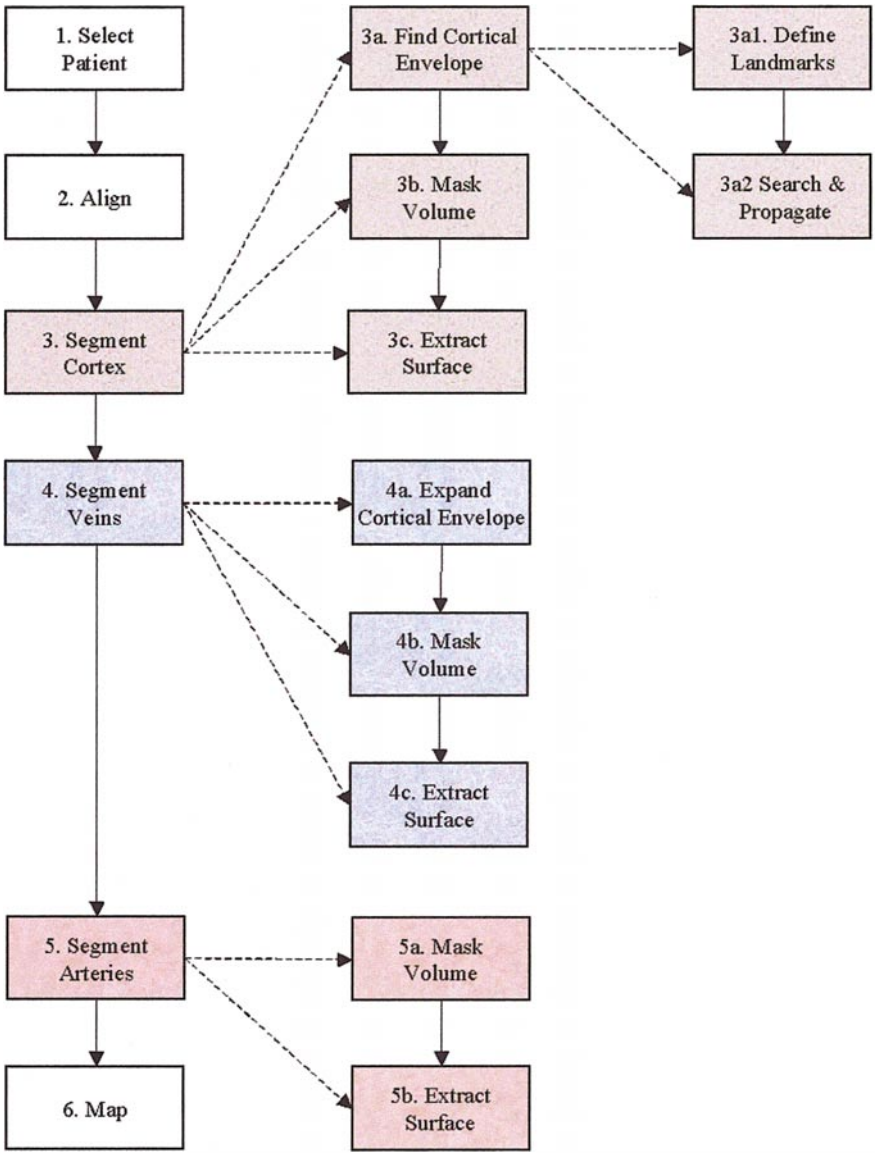


FIG. 13. Visualization-based mapping workflow. Overall workflow is in the left-hand column, and consists of 6 worksteps (Select Patient–Map). Middle columns are sub-workflows, each of which consists of several worksteps. Step 3a, Find Cortical Envelope, has a sub-sub-workflow shown in the right-hand column.

ing radial surface model instance to mask out the volume data that are outside the cortical envelope (step 3b), thereby providing a cleanly segmented image volume for isosurface extraction (step 3c).

Step 3b: Create the voxel mask. The radial surface model instance is surface-based, so it must be converted into a region-based voxel mask. This is accomplished in two steps. First, a shell is constructed by locating all voxels within a certain distance, ε, of at least one facet of the model. Using $\varepsilon = 0.5$ produces a strict voxel rasterization of the radial surface; increasing the tolerance gives the shell thickness and rounds out its corners. Second, a flood-fill algorithm is used to add all voxels inside the shell to the mask. Since the model is closed by construction, this filling operation will not spill into the region outside the shell. Figure 16 shows the computed mask.

Step 3c: Extract the isosurface. After the voxel mask has been computed, the detailed surface can be extracted. First a new volume dataset is created by setting the intensity of all voxels outside the masked region to zero. This masked dataset is then passed to an isosurface extraction algorithm to convert its voxel-based representation of the brain into a surface-based one, Fig. 17. The technique is based on the marching cubes algorithm (Lorensen and Cline, 1987), using the method described by Bloomenthal to resolve topological ambiguities (Bloomenthal, 1988).

FIG. 14. Intraoperative photograph taken during cortical stimulation mapping. Sterile number tags mark positions where stimulation tests were performed. The scale bar in the lower right-hand corner is 1 cm long.

With the exception of the cortical envelope fitting, the stages of segmentation run very quickly (on the order of seconds because both the volume masking step and the isosurface extraction step have been highly optimized). Thus, it is easy for the user to adjust the segmentation results interactively. By looking at the masked volume or the extracted isosurface, it is fairly easy to locate errors in the fitted RSM. Fixing such

Surface anatomy: T1-weighted SPGR (29/5/1/45°) [TR/TE/NEX/flip angle], 22 cm field-of-view (FOV), 256 × 192, 1.2 mm sagittal slices.

Veins: 2D time-of-flight venogram (45/9/1/60°), 22 cm FOV, 256 × 192, 1.5 mm axial slices.

Arteries: 3D MOTSA MR angiogram, 4 overlapping slabs of 16 partitions each (36/6.9/1/25°), flow compensation, 22 cm FOV, 256 × 256, 0.9 mm axial slices.

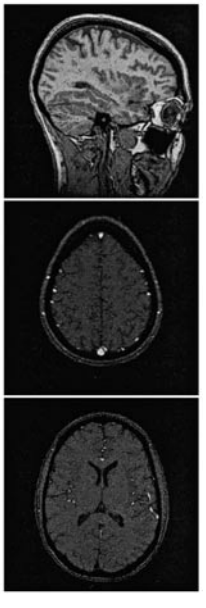
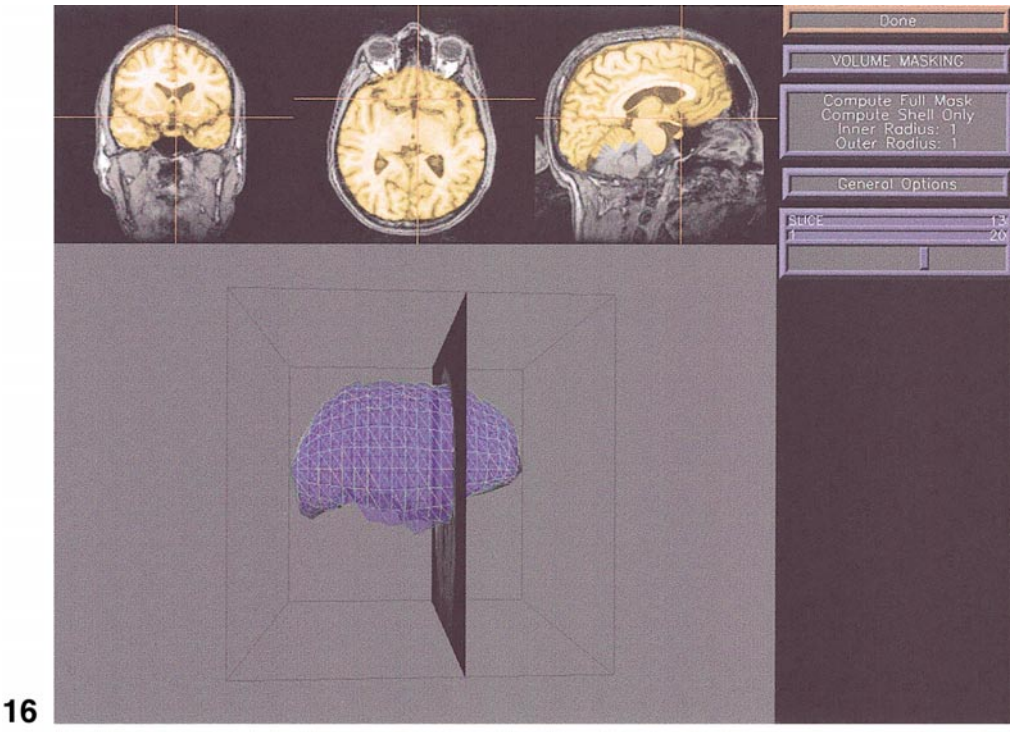
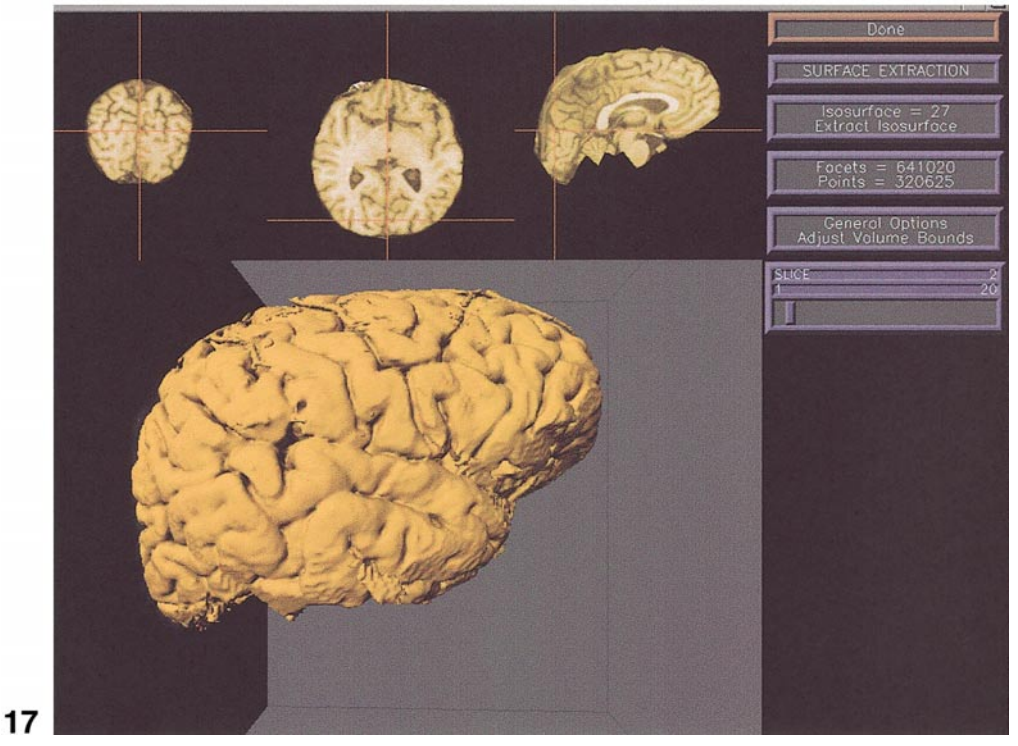


FIG. 15. The three MRI datasets used for reconstruction.



16



17

FIG. 16. Volume mask before skull stripping. The RSM instance is used to label all voxels within a user settable distance of the shell defined by the instance. As shown in the color figure, the system tints the labeled voxels for easy identification. If errors occur, the user can go back and manually adjust the radials giving rise to the error and then recompute the mask.

FIG. 17. Isosurface extraction. The skull-stripped volume is input to a simple isosurface extraction algorithm, which creates a surface from voxels forming the boundary of the cortical region.

errors is straightforward—the user simply adjusts radials in the affected region, remasks the volume, and extracts a new isosurface.

2.3.4. Steps 4 and 5: Segmentation of the veins and arteries. The process for reconstructing the veins and arteries is nearly identical to that for the cortical sur-

face. The RSM instance obtained from segmenting the cortical envelope provides an obvious starting point. One minor complication is that the vessels of interest lie just above the cortical surface, and therefore the larger ones may lie partially outside the fitted model.

To address this problem, a companion RSM instance is used for the veins and arteries. This model is initialized by copying the landmarks and radial lengths from the first RSM instance, avoiding the overhead of a second round of constraint propagation and edge detection. The copied model instance is then "inflated" slightly (step 4a) by lengthening all of its radials by a fixed percentage. Since this inflation may not have the desired effect everywhere, the user can edit the result to make sure that surface vessels are included, but that radials do not extend so far as to include scalp vessels.

The rest of the segmentation then proceeds as it did with the cortical surface. The model is used to remove structures outside the brain (steps 4b, 5a), and isosurface extraction is used to reconstruct the veins (step 4c) and arteries (step 5b) from their respective datasets. The user can experiment with different thresholds until a satisfactory level is found, and the resulting surfaces are saved to the database. Figure 18 shows the extracted veins; Fig. 19 shows the extracted arteries; Fig. 20 shows combined veins and arteries; and Fig. 21 shows combined veins, arteries, and cortical surface, ready for input to the visualization mapping procedure.

2.3.5. Step 6: Visual mapping. The final stage of the mapping process is nearly identical to that described in our earlier work (Modayur *et al.*, 1997), except that the procedure is integrated with the other steps, and all the data are automatically loaded. The user interface, which is shown in Fig. 22, depicts the intraoperative photograph, a rendering of the cortical, vein and artery surfaces as generated in steps 3–5, coronal and transverse sections through the aligned MR volumes as generated in step 2, a control menu, and a palette of draggable numbers. Using camera, material and lighting controls on the menu (which become available when the various menu options are clicked) the user positions and colors the 3-D rendering to match as closely as possible the view of the cortex as seen in the photograph. The user then selects numbers from the palette and drags them onto the rendering such that they match as closely as possible the corresponding locations of the numbered tags. The reconstructed blood vessels provide key landmarks for this visual matching process. Those tags that were found during neurosurgery to be critical for language can be noted, at which point a box is drawn around them. Figure 23 shows a closeup of the final map generated using the visual mapping interface.

The output of the mapping procedure is a file containing the 3-D MR machine coordinates of each numbered site, together with an indication of whether the site was critical for language. These 3-D points are in the same coordinate system as the MR images, so are therefore comparable with image-based functional mapping methods such as fMRI. They can also be input to our Web-based experiment management system for further analysis (Jakobovits and Brinkley, 1997), which provides access to an applet for remote visualization over the web (Poliakov *et al.*, 2001) or can be given as input to surface based analysis programs such as FreeSurfer (Dale *et al.*, 1999) or Caret (Van Essen *et al.*, 2001).

2.4. Implementation

The individual modules of the VBM are implemented using an in-house software toolkit called *Skandha4*, which combines C-based image and graphics operations with a Lisp scripting language for rapid program development.

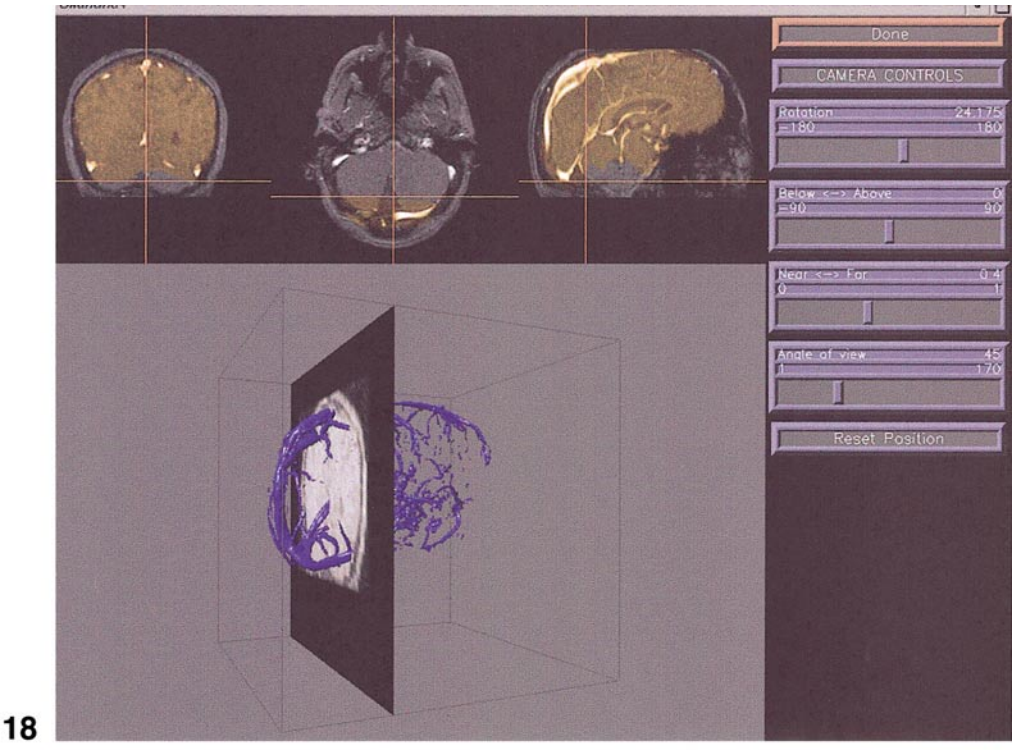
In our previous work the visualization mapping protocol was implemented with an assortment of software tools, but getting them all working successfully required considerable expertise. Different pieces were handled by different tools and scripts, making it difficult for the uninitiated to learn to use the system. The segmentation tools in particular required a solid understanding of rather unintuitive parameters to control a 3-D region grower.

To overcome these difficulties, VBM was designed to present all of the steps involved in visualization mapping within a single, unified workflow. The protocol shown in Fig. 13 naturally lends itself to a workflow description: the larger problem can be partitioned into smaller ones, each of which takes certain inputs, manipulates them in some way, and generates outputs for the next stage. Consequently a workflow specification language (Ailamaki *et al.*, 1998) was built in order to tie individual modules together into an integrated application for brain mapping. The language uses three simple building blocks:

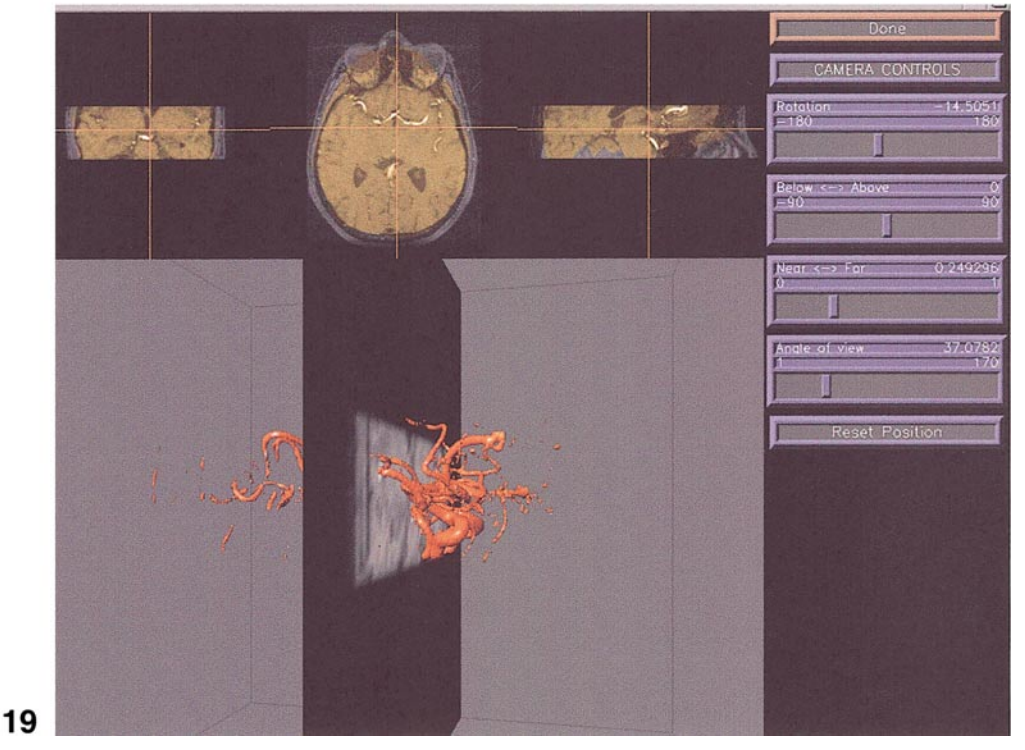
- a *workflow*, which groups related worksteps and datapaths into a task sequence.
- a *workstep*, which performs a particular task, manipulating a set of inputs to generate a set of outputs. A workstep can include a subsidiary workflow to specify subtasks.
- a *datapath*, which transfers data between worksteps.

In Fig. 13 the leftmost column represents the primary workflow, which consists of individual worksteps such as Align and Segment Cortex. The middle and right columns show subsidiary workflows, each of which consists of worksteps defining the subtasks of the parent step. The datapaths are represented by the arrows between worksteps.

This formalized workflow mechanism provides several advantages. First, it simplifies the process of



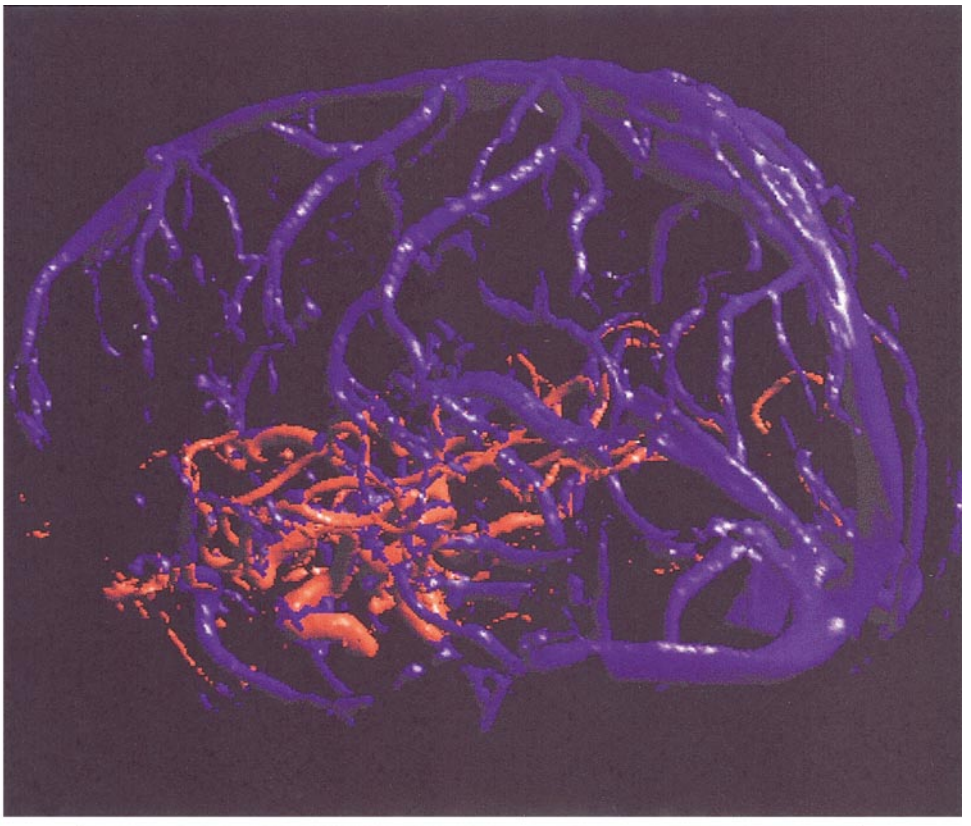
18



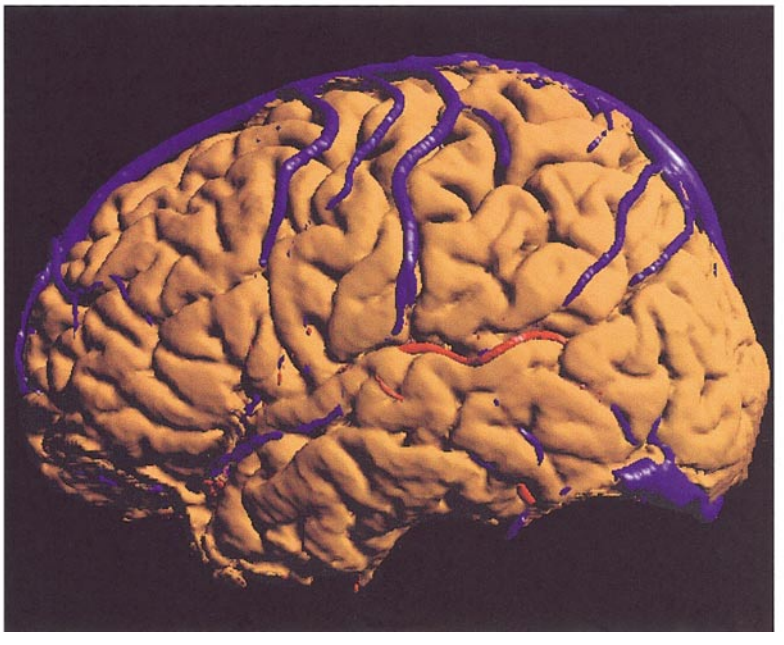
19

FIG. 18. Vein segmentation. The RSM instance is semiautomatically inflated to create a mask that excludes scalp veins. The isosurface algorithm is applied to extract the high-contrast veins from the magnetic resonance venography (MRV) images. Menus such as these camera control widgets allow the veins and other surfaces to be rotated or zoomed in and out.

FIG. 19. Artery segmentation. The same inflated RSM instance used to extract the veins is used to create a mask that excludes scalp arteries from the magnetic resonance arteriography (MRA) image volumes (only a subvolume is imaged in order to save scanning time). The isosurface algorithm is then applied to extract the high-contrast arteries.



20



21

FIG. 20. Combined vessels. Extracted veins and arteries can be simultaneously displayed.
FIG. 21. Reconstructed cortex with veins (blue) and arteries (red).

changing the mapping protocol. Modules can be added or removed with minimal effort. Second, the datapaths between worksteps provide useful data dependency information. For example, VBM presents menus that track the user's progress through the mapping protocol. In these menus, data dependencies are used to distinguish between steps that have been finished, steps that are ready to run, and steps that cannot yet be started. Third, the workflow system can simplify the user's workload by automatically managing intermedi-

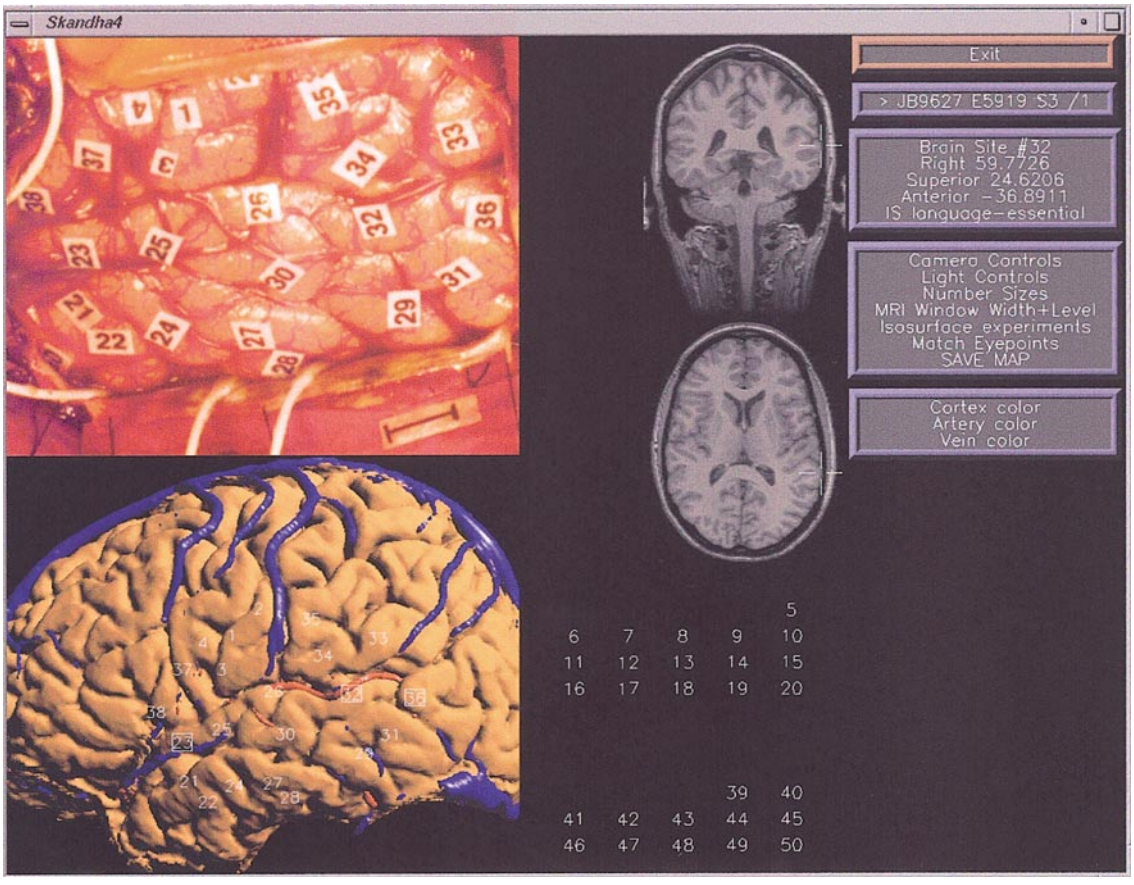


FIG. 22. The visual mapping interface. The reconstructed surface and vessels are shown below a photograph taken during surgery. The user drags tags from a number palette and drops them onto their correct positions on the 3-D surface. After a tag has been placed, the system can also display its position on the original MRI slices (as shown in this example for site 32). Tags with boxes around them indicate language-critical sites.

ate data files. Additional details about the workflow implementation are presented elsewhere (Hinshaw, 2000).

3. RESULTS

In our previous work, repeatability studies showed that with the visualization mapping method, CSM sites could be localized within about 5 mm, which is less than the approximately 1 cm error in localization of the stimulating electrode (Modayur *et al.*, 1997). In this new version of VBM, the goal was to produce an application that is not only accurate, as shown in the previous study, but is also fast and easy to use. Our evaluations in the present study therefore concentrated on these two factors.

3.1. Speed

To gauge the speed of the system, estimates were made of the clock time required to run a patient through the entire mapping procedure. The primary neuroscience user (author R.M.) has employed the system to generate brain models and maps for some 40 patients. For all these patients, mapping was performed on a Silicon Graphics Octane workstation with an IP30 processor and 384 megabytes of memory. On average the entire mapping procedure required just under one hour to complete. Estimated average time expenditures for each stage of the procedure are shown in Table 1. The time for image acquisition and transfer has been omitted.

As shown in Table 1 the primary bottleneck is the Segment Cortex step (step 3 in Fig. 13, of which the slowest substep is Find Cortical Envelope (step 3a)). The other substeps (Mask Volume and Extract Surface) run on the order of seconds.

Find Cortical Envelope is the interactive shape-based method that uses the radial surface model 2described in section 2.1.3. The clock time for this step is proportional to the number of radials in the model instance that must be corrected by the user, since the time to input the Talairach landmarks is more or less constant, and the edge detection and constraint propagation procedures take minimal time. We therefore

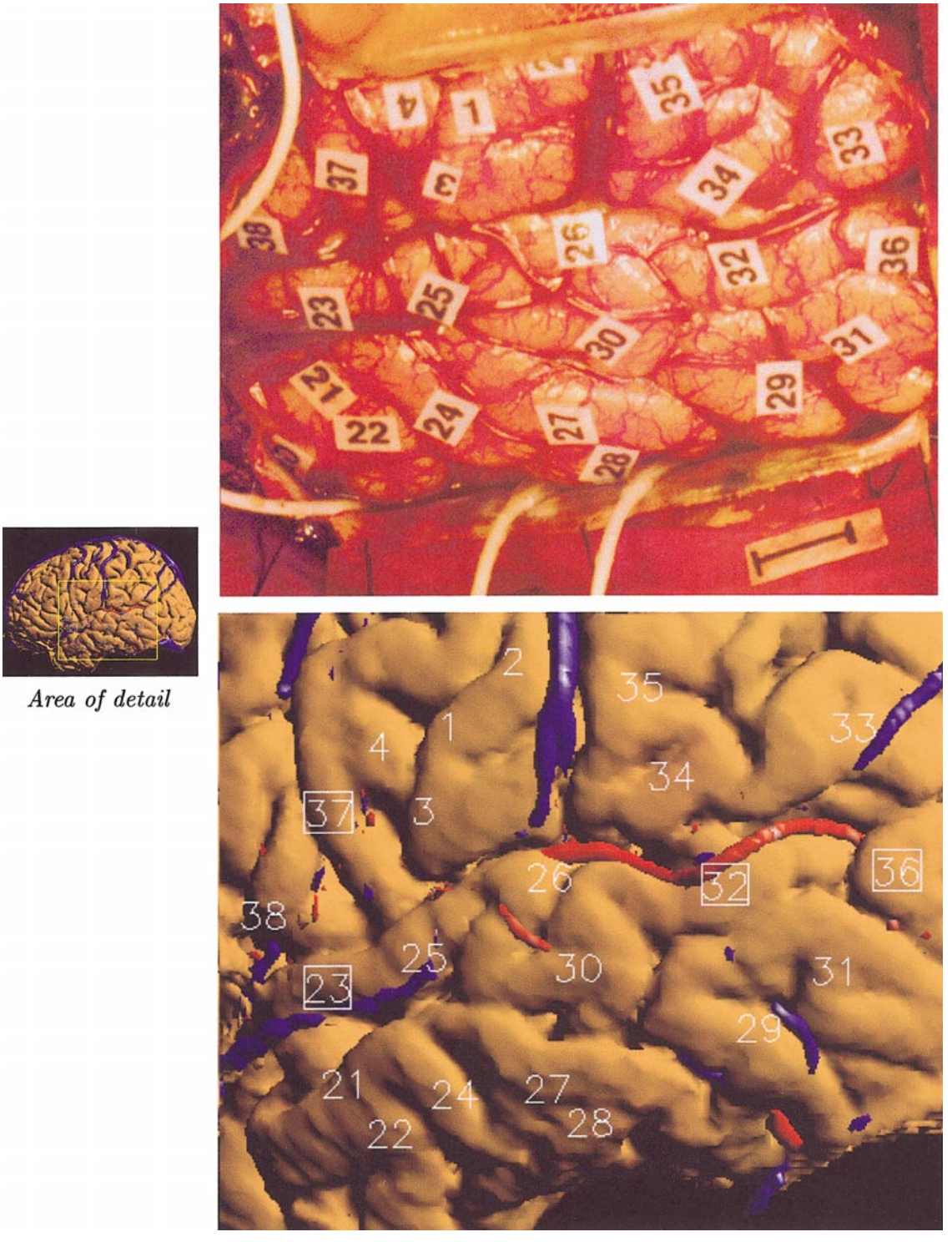


FIG. 23. A completed brain map.

examined the number of radial corrections in more detail by simulating user interaction on 16 patient datasets.

The 16 datasets were selected from approximately 40 in the database as all having more or less the same "normal" overall cortical shape. For each of these datasets, a radial surface model (RSM) instance was generated by using Scanner's surface editor to define the radials for the 600 radials in the surface (20 slices times 30 radials per slice). These instances were then

TABLE 1

Estimated Timings for the Mapping Pipeline Stages (h:mm)

	Average (h:mm)
Alignment	0:05
Segment cortex	0:30
Segment veins	0:06
Segment arteries	0:06
Visual mapping	0:12
Total	0:59

used for a series of leave-one-out experiments similar to those we performed in earlier work for the 2-D case (Brinkley, 1993).

For each leave-one-out experiment 15 of the 16 RSM instances were used as a training set to create a RSM. The RSM was then used to segment the 16th dataset using the procedures described in section 2.2, and the resulting automatically determined RSM instance was compared with the manually determined RSM instance. To simulate user initialization of the segmentation process, the Talairach landmarks for each dataset were taken from the corresponding hand-drawn RSM instance. The segmenter was initialized with a single radial from the hand-drawn RSM instance, constraints were propagated to shrink the uncertainty bounds, then alternating phases of edge detection and constraint propagation were run until all radials had been searched.

The result for each trial was an automatically-segmented cortical envelope that could be compared with the hand-drawn "gold standard" cortical envelope. The comparison was made by counting the number of radials in the automatically-determined surface that differed from the hand-drawn surface by more than 4 voxels. The value 4 was chosen empirically. The number of radials counted is presumably a good indicator of the number that would need to be corrected by the user in an interactive system. Radials that were specified as "ignored" in the RSM were counted as being correct, since presumably the user would not need to interact with these radials.

Of the 600 radials in the segmented surface, the simulations showed that the user would have needed to correct an average of 79 (13%) using the search strategy described in section 2.2.2. When compared to a purely manual system, in which the user would need to interact with all 600 radials to set their lengths, this number represents a potential reduction in workload of $100 - 13 = 87\%$.

A more conservative comparison takes into account the 220 ignored radials (11 radials per slice times 20 slices) that are not searched but whose values are set purely by the shape constraints. A system that did not do any automatic edge detection but still used shape constraints or some other method to define the ignored radials would require the user to set the values for all nonignored radials, that is $600 - 220 = 380$ radials. Therefore, in the above simulations the user would have needed to correct 79 of 380 radials (21%) or a potential reduction in workload of $100 - 21 = 79\%$.

Thus, depending on how the comparison is made, the simulations show that the shape constraints and edge heuristics should theoretically reduce the user's workload by about 80–90%. As improvements in the skull stripping method are developed, this automatic evaluation strategy will allow us to quickly determine the expected speed-up in the procedure, without requiring tedious manual testing.

3.2. Usability

The second performance measure, ease of use, was assessed by interviewing the primary neuroscience user (author R.M.), who has no programming experience. Compared to other programs, the primary user finds the VBM protocol much easier to perform because the workflow architecture ensures that the next step is automatically indicated on the menu, and steps that do not yet have all their needed inputs are shown as blocked. Furthermore, files representing the input or output of individual steps do not need to be explicitly loaded or saved—those procedures are done automatically by the workstep.

The primary user reports that the skull stripping procedure saves time over a manual method, and he believes that by having to redraw only selected radials, the quality of his manual corrections is better due to reduced eye and hand fatigue. When segmentation errors do occur he can quickly locate them, adjust the fitted model in that region, and extract a new surface. From a user's standpoint, this is a significant improvement over our earlier region growing method, which offered no local control over the segmentation results.

But the best evidence that a "useful system" has been produced is that the system is actually used. To date, VBM has been used by four different individuals to reconstruct and map over 40 patients. Furthermore, it is playing a significant role in neuroscientific research. For example, the system provided key support for a rare case study involving a deaf signer undergoing cortical stimulation mapping (Corina *et al.*, 1999). In addition, the system has been expanded to support comparisons between CSM results and fMRI (Poliakov *et al.*, 1999), and other researchers have expressed interest in using it for their own projects.

4. DISCUSSION

This paper has described a novel skull-stripping technique that uses learned shape knowledge of the cortical

envelope to guide low-level image processing operators. The shape-based technique is integrated within a workflow architecture that permits simple and intuitive operation of the visualization mapping protocol by nonprogrammer users.

Evaluation results show that the system is fast and easy to use, both because of the workflow architecture and because of the intuitive control of the segmentation process. Although the surface-finding algorithms are not completely automatic, as are those in some of the other surface-finding methods, the procedure is less time-consuming than a manual method, and produces surfaces that are more realistic for the visualization mapping protocol.

The shape-based skull stripping procedure could be improved in several ways that would greatly reduce the amount of user interaction. For example, the specification of Talairach landmarks (Step 3a1) could most likely be eliminated if the image volumes were first transformed to the average brain coordinate system by the Montreal Auto-Register program (Collins *et al.*, 1994) (as we have done for some of our later analysis, and as is done by FreeSurfer and ASP). In this case the radial surface model could simply be initialized once for all patients within the coordinate system of the average brain. Other possible improvements include additional heuristics for edge detection along radials, backtracking if an incorrect edge is found, and incorporation of the shape constraints into the cost function for an optimization procedure. These and other improvements should gradually lead to an automated or near-automated procedure that produces much more accurate skull strippings than are available from nonmanual methods.

The surface-finding procedures we have described are most likely to find use in applications that need realistic visualizations of the cortical surface. Thus, CSM-based mapping protocols for modalities other than language could benefit from these techniques, as could educational applications, and potentially surgical planning.

The surfaces produced by our approach are not in a form that can be directly input to programs for surface analysis or surface reconfiguration. For these applications the automatic programs such as FreeSurfer, ASP or SureFit are better suited because they produce topologically correct surfaces that can be analyzed and manipulated. However, since our mapping procedures generate MR machine coordinates of the CSM sites, it should be possible to provide these coordinates as input to the automatic procedures, which can then carry the CSM sites along as they warp or unfold the brain surface. In addition, once our skull-stripping technique becomes more automated, it may be useful as a replacement for the skull-stripping methods that are embedded in these programs.

All the software tools described in this paper are being developed under an open source license through the auspices of the Human Brain Project. As these and other tools sponsored by the Human Brain Project become more widely available to the neuroscience community, they should greatly speed progress in neuroscientific research.

ACKNOWLEDGMENTS

This work was funded by Human Brain Project Grant MH/DC02310. We thank other members of the University of Washington Human Brain Project for their help and encouragement on various aspects of this work: David Corina, George Ojemann, Ken Maravilla, Cornelius Rosse, Etore Lettich, Jeff Prothero, Julia Byer, and Karen Kinbar.

REFERENCES

Altman, R. B., and Brinkley, J. F. 1993. Probabilistic constraint satisfaction with structural models: application to organ modelling by radial contours. In *17th Symposium on Computer Applications in Medical Care,* Washington, DC, pp. 492–497.

Ailamaki, A., Ioannidis, Y. E., and Livny, M. 1998. Scientific workflow management by database management. In *Proceedings of 10th International Conference on Scientific and Statistical Database Management,* pp. 190–199.

Altman, R. B. 1990. *Exclusion Methods for the Determination of Protein Structure from Experimental Data,* PhD thesis. Technical Report STAN-CS-90-1306, Stanford Univ. Departments of Computer Science and Medicine.

Brinkley, J. F., Altman, R. B., Duncan, B. S., Buchanan, B. G., and Jardetzky, O. 1988. Heuristic refinement method for the derivation of protein solution structures: Validation on cytochrome b562. *J. Chem. Inf. Comput. Sci.* **28**(4): 194–210.

Bloomenthal, J. 1988. Polygonization of implicit surfaces. *IEEE Comput. Graphics Appl.* **5**(4): 341–355.

Brinkley, J. F. 1985. Knowledge-driven ultrasonic three-dimensional organ modelling. *PAMI,* **7**(4): 431–441. [Also available as Stanford Technical Report KSL-85-33]

Brinkley, J. F. 1992. Hierarchical geometric constraint networks as a representation for spatial structural knowledge. In *Proceedings of 16th Annual Symposium on Computer Applications in Medical Care,* pp. 140–144.

Brinkley, J. F. 1993. A flexible, generic model for anatomic shape: Application to interactive two-dimensional medical image segmentation and matching. *Comput. Biomed. Res.* **26**: 121–142.

Carman, G. J., Drury, H. A., and Van Essen, D. C. 1995. Computational methods for reconstructing and unfolding the cortex. *Cerebral Cortex* **5**(6): 506–517.

Corina, D. P., McBurney, S. L., Dodrill, C., Hinshaw, K., Brinkley, J., and Ojemann, G. 1999. Functional roles of Broca's area and SMG: Evidence from cortical stimulation mapping in a deaf signer. *NeuroImage* **10**: 570–581.

Collins, D. L., Neelin, P., Peters, T. M., and Evans, A. C. 1994. Automatic 3-D intersubject registration of MR volumetric data in standardized Talairach space. *J. Comput. Assist. Tomogr.* **18**(2): 192–205.

Davatzikos, C., and Bryan, R. N. 1996. Using a deformable surface model to obtain a shape representation of the cortex. *IEEE Trans. Med. Imag.* **15**(6): 785–795.

Dale, A. M., Fischl, B., and Sereno, M. I. 1999. Cortical surface-based analysis. i. Segmentation and surface reconstruction. *NeuroImage* **9**(2): 179–94.

Fischl, B., Sereno, M. I., and Dale, A. M. 1999. Cortical surface-based analysis. ii: Inflation, flattening, and a surface-based coordinate system. *NeuroImage* **9**(2): 195–207.

Hinshaw, K. P., Altman, R. B., and Brinkley, J. F. 1995. Shape-based models for interactive segmentation of medical images. In *SPIE Medical Imaging 1995: Image Processing*, pp. 771–780. San Diego.

Hinshaw, K. P., and Brinkley, J. F. 1997. Shape-based interactive three-dimensional medical image segmentation. In *SPIE Medical Imaging: Image Processing* (K. M. Hanson, Ed.), Vol. 3034, pp. 236–242. Newport Beach, CA.

Hinshaw, K. P. 2000. *Seeing Structure: Using Knowledge to Reconstruct and Illustrate Anatomy.* PhD thesis, University of Washington.

Jakobovits, R. M., and Brinkley, J. F. 1997. Managing medical research data with a web-interfacing repository manager. In *Proceedings of American Medical Informatics Association Fall Symposium*, pp. 454–458, Nashville.

Kass, M., Witkin, A., and Terzopoulos, D. 1987. Snakes: Active contour models. *Int. J. Comput. Vis.* **1**(4): 321–331.

Lorensen, W. E., and Cline, H. E. 1987. Marching cubes: A high resolution 3-D surface construction algorithm. *ACM Comput. Graphics* **21**(4): 163–169.

Lancaster, J. L., Fox, P. T., Downs, H., Nickerson, D. S., Hander, T. A., El Mallah, M., Kochunov, P. V., and Zamarripa, F. 1999. Global spatial normalization of human brain using convex hulls. *J. Nucl. Med.* **40**(6): 942–955.

Mackworth, A. K. 1977. Consistency in networks of relations. *Artific. Intell.* **8**: 99–118.

MacDonald, D. 1993. Register, Montreal Neurological Institute.

Markosian, L. Cohen, J. M., Crulli, T., and Hughes, J. F. 1999. Skin: A constructive approach to modelling free-form shapes. In *Proceedings SIGGRAPH*, pp. 393–400.

MacDonald, D., Kabani, N., Avis, D., and Evans, A. C. 2000. Automated 3-D extraction of inner and outer surfaces of cerebral cortex from mri. *NeuroImage* **12**(3): 340–356.

Modayur, B., Prothero, J., Ojemann, G., Maravilla, K., and Brinkley, J. F. 1997. Visualization-based mapping of language function in the brain. *NeuroImage* **6**: 245–258.

Ojemann, G., Ojemann, J., Lettich, E., and Berger, M. 1989. Cortical language localization in left, dominant hemisphere. *J. Neurosurg.* **71**: 316–326.

Poliakov, A. V., Albright, E., Corina, D., Ojemann, G., Martin, R. F., and Brinkley, J. F. 2001. Server-based approach to web visualization of integrated 3-D medical image data. In *Proc. AMIA Fall Symposium*, pp. 533–537.

Poliakov, A. V., Hinshaw, K. P., Rosse, C., and Brinkley, J. F. 1999. Integration and visualization of multimodality brain data for language mapping. In *Proceedings of American Medical Informatics Association Fall Symposium*, pp. 349–353. Washington, DC.

Sandor, S., and Leahy, R. 1997. Surface-based labeling of cortical anatomy using a deformable atlas. *IEEE Trans. Med. Imag.* **16**(1): 41–54.

Smith, S. M. 2000. *BET: Brain Extraction Tool*, Technical Report TR00SMS2a. Oxford Centre for Functional Magnetic Resonance Imaging of the Brain, Oxford University.

Toga, A. W. 2001. The cortical envelope. Personal communication.

Talairach, J., and Tournoux, P. 1988. *Co-planar Stereotaxic Atlas of the Human Brain.* Thieme Medical, New York.

Van Essen, D. C., Drury, H. A., Dickson, J., Harwell, J., Hanlon, D., and Anderson, C. H. 2001. An integrated software suite for surface-based analysis of cerebral cortex. *J. Am. Med. Assoc.* **8**(5): 443–459.