Chapter 3

Ridge Flow Segmentation

3.1 Introduction

Segmentation of images into meaningful regions has been an important area in medical image analysis. Rapid, accurate, automated identification and measurement of anatomical objects in medical images, such as organs and tumors, is a desirable goal. Three typical applications requiring segmentation are display from regions of interest, measurement of object information, and atlas identification. A 3–dimensional image contains a large amount of data, so rendering of the complete data set can be time consuming. Selecting regions of interest will reduce the amount of data to process and will reduce the rendering time. Also, objects that the user wants to display may be obscured by other irrelevant objects in the data. To display a single unobscured object, the selection of a region of interest containing only the object becomes an important issue. A good segmentation algorithm should produce a region just slightly larger than the object. Display of an object gives a good qualitative description, but quantitative measurements might also be required, for example, the volume of a tumor. Current clinical tools allow radiologists to "hand–segment" objects in images by drawing contours around the objects of interest on a slice–by–slice basis. Volume estimates can be made by calculating the enclosed area on each slice and then adding up the areas. This process is slow and tedious. A good segmentation algorithm should allow the radiologists to make more accurate measurements with far less cost of user time. Finally, if the objects in an image are precisely located and represented by an abstract structure such as a tree or

graph, then the objects can be identified by matching their representations against an atlas of representations stored as a data base of previously segmented objects. The resulting naming can be fed back to allow a model to improve the segmentation. It can allow object–specific measurements to be made, including deviation from normality.

Automated segmentation processes are not as adept as humans at recognizing objects. The chances are small that the segmentation process will correctly associate with each object a single primitive region. Realizing this limitation, a good design goal is to create many small primitive regions and attempt to semantically link them into a hierarchy. The semantic rules can be based on such constructs as pyramids (Burt, Hong & Rosenfeld 1981, Burt & Adleson 1983), the difference–of–low–pass transform (Crowley & Parker 1984), tree matching (Beaulieu & Goldberg 1989), or multiscale methods (Gauch, Oliver & Pizer 1988, Gauch & Pizer 1988, Lifshitz 1990, Pizer, Cullip & Fredericksen 1990) where the scale space is constructed via Gaussian blurring. A goal of this chapter is to show how such hierarchies can be used for interactive object definition. The hierarchies also have the potential for object recognition using tree matching algorithms.

Hierarchy construction via multiscale methods focuses on annihilation of certain geometric features as a means of identifying both primitive regions and parent–child links. Primitive regions consist of points which are similar with respect to a selected geometric feature. The boundaries (and nearby points) of primitive regions are points which are not similar to the interior points. For example, one might choose to group points based on magnitude of gradient intensity, as is so often the case in edge detection algorithms. A primitive region contains points for which the gradient magnitude is small. At the boundaries of the regions, the gradient magnitude is relatively larger. Each level of a hierarchy corresponds to viewing the initial image at a larger scale. The parent–child links are established by observing how the geometric features are annihilated through increasing scale. In the example of gradient magnitude, if two regions have part of their boundaries (points of large gradient magnitude) close together at one scale, and if at a larger scale part of the common boundary blurs together, then the common boundary was annihilated and the two objects blur into a single object. The links in the hierarchy will capture this annihilation information.

In Lifshitz (1990), the initial image $I(\vec{x})$ is blurred using Gaussian kernels of increasing

standard deviation σ (scale) to form the function $B(\vec{x}, \sigma)$, with $B(\vec{x}, 0) = I(x)$. The hierarchy construction is based on the nesting of light and dark regions at varying scales, each such region containing a local extremum of intensity. The local extrema are tracked through increasing scale until annihilation, producing extremum paths in scale space. The paths are necessarily solutions to $\nabla_{\vec{x}} B(x, \sigma) \equiv 0$, where the gradient is taken only in the spatial components.

I give a brief description of the algorithm and its intended result. As noted in the next paragraph, the experimental results showed that the algorithm did not always produce what was expected. At a point $(\vec{x}_0, \sigma_0)$ on the extremum path with $B(\vec{x}_0, \sigma_0) = c$, there is an isointensity surface of the blurred function, implicitly defined by $B(\vec{x}, \sigma) \equiv c$, which contains the point. At zero scale, the isointensity contour implicitly defined by $I(\vec{x}) \equiv c$ is associated with the given extremum path. In particular, at the point of annihilation, the associated isointensity contour of I forms the boundary of an extremal region which is assigned to the extremum path. At annihilation, another region's isointensity contour (for B at the scale of annihilation) encloses the region associated with the annihilating extremum. This induces a containment relationship among extremal regions which can be represented as a hierarchy.

The above construction has some drawbacks. One problem is that pixels which visually all belong to one region may be linked to different extremum paths; that is, only portions of level curves were mapped to single extremum paths. This is contrary to the intuition one has about nesting of regions. Another problem is that the algorithm places too much emphasis on intensity. The links in the hierarchy are created based on the intensity of annihilation. A better algorithm would be one which emphasizes higher order geometric effects.

In Gauch & Pizer (1988), the initial image $I(\vec{x})$ is also blurred to construct a scale space image $B(\vec{x}, \sigma)$. Shape of objects in the image is described in terms of the level sets of the image, thus providing a description which is invariant to monotonic rescaling of intensities. At zero scale, for each intensity c a level disk is $D_c = \{\vec{x} : I(\vec{x}) \geq c\}$. The Blum medial axis (Blum & Nagel 1978) was constructed for each level disk, call it M_c. The totality of all such axes is called the *intensity axis of symmetry* (IAS), given by $A = \cup\{(\vec{x}, c) : c \in \text{range}(I), \vec{x} \in M_c\}$. The axis consists of branching and looping sheets which fit under the graph of intensity. The shape and placement of the sheets are affected most by the vertices

of the level curves. Visually, the sheets fit under the graph at places one would call ridges of the surface.

The IAS is computed for the blurred image $B(\vec{x}, \sigma)$ through increasing scale. At each scale, primitive regions are constructed based on the decomposition of the IAS sheets into simpler structures. For each simple structure the boundary of the region is built using watershed methods. The hierarchy relating these regions is produced by tracking the IAS through scale space. The links between scales are constructed based on how branches of the IAS are annihilated. The algorithms are based on active surfaces, a generalization of active contours (Kass, Witkin & Terzopoulos 1987).

The IAS construction is computationally expensive and requires careful handling of the topological problems that can arise in the active surface algorithm. Numerical difficulties in tracking the level curve vertices through scale can be a problem. A drawback to both the hierarchy constructions of Lifshitz (1990) and Gauch & Pizer (1988) is that object boundaries are created as level sets of intensity. Typically object boundaries are not level sets, especially in images with noise, and they depend on higher–order geometric information.

In Fredericksen, Coggins, Cullip & Pizer (1990) and Pizer et al. (1990), a segmentation algorithm is developed using a *peak flow model*. In the model the local maxima of the intensity function are identified. The primitive regions are built using a reverse watershed method. A local maximum point is assigned a region of points, each point being the initial value for a flow line on the graph of intensity such that the flow terminates at the local maximum. The flow line directions are determined by the gradient of intensity. This process segments the pixels into a disjoint union of primitive regions. The main problem with the peak flow model is that visually important regions may not correspond to watershed regions. For example, consider the subgraph of intensity over a watershed region. The subgraph contains a single peak (by definition) but may also contain ridge structures. A flank associated with a ridge may itself contain subridge structures. Visually, the subridge and an associated region containing it are important, but such a region is never identified in the peak flow process.

In Section 3.2 I present a general method for segmenting medical images which is based on the ridge definitions discussed in Chapter 2. Ridges of an image appear to be associated with regions of perceptual importance. The hierarchies I construct are also based on

annihilation, but now the selected geometric features are ridges. The segmented image is represented by a hierarchy which is built using multiscale and differential geometric methods. The leaf nodes of the hierarchy represent small, primitive regions of an image. These regions are constructed by using a *ridge flow model* which is a generalization of the peak flow model in the sense that peaks are local maxima of intensity and ridges are a generalization of local maxima. The ridges are segmented into curvilinear segments. A ridge segment is assigned a region of points, each point being the initial value for a flow line in the image plane such that the flow terminates at the ridge segment. The flow process is a generalization of the reverse watershed method, but in our case the flow line directions are determined not by the gradient of intensity, but by the gradient of a function which is in some sense natural to the ridge construction. A refinement of the ridge flow model which uses both ridges and valleys to form primitive regions is the *ridge–valley flow model*, discussed in Section 3.3.

The interior nodes of the hierarchy relate the primitive regions in a way that reflects the natural object structure of the image. Each level of nodes in the hierarchy represents primitive regions of an image which has been blurred to a selected scale using *variable conductance diffusion* (Grossberg 1984, Perona & Malik 1987, Whitaker 1993). As compared to the regions obtained in Lifshitz (1990) and Gauch & Pizer (1988), our region boundaries are not level sets; rather they delimit those points which form the flanks associated with the ridge segment. The blurred image is also segmented using a ridge flow model. The idea is that small regions at one scale are blurred into a single larger region at a larger scale. Such small regions will be linked to a parent node at the next level in the hierarchy. As the scale increases without bound, the blurring process yields a constant image. Thus, at a large scale, all nodes will eventually have a single ancestor–the root of the tree. Anatomical objects are obtained as unions of subtrees of the hierarchy. A detailed description of the hierarchy construction is given in Section 3.4.

I have implemented the algorithms for any dimension $n \geq 1$, not just for the usual 2 and 3 dimensions. The image and corresponding hierarchy are used as input to two object definition and visualization tools, called the Magic Crayon (Beard, Faith, Eberly, Pizer, Kurak & Johnston 1993, Beard, Eberly, Faith, Kurak, Paramasivam & Pizer 1994) and the Interactive Hierarchy Viewer (Fredericksen et al. 1990, Pizer et al. 1990), which allow the

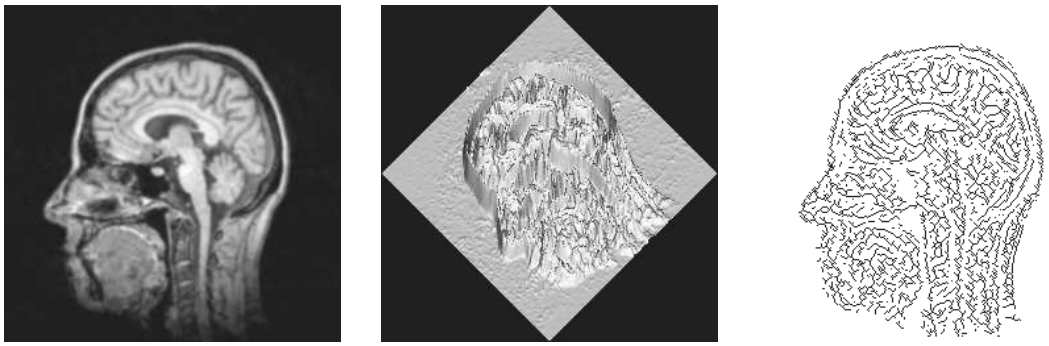


Figure 3.1: MR image, MR intensity surface, and ridges of intensity

user to interactively define objects by rapidly traversing the hierarchy and displaying the regions of interest with color overlays. A discussion of these tools is given in Section 3.5. The results of the segmentation applied to 2– and 3–dimensional images are also provided.

3.2 Ridge Flow Models

For many types of medical images, the intensity values tend to be large at pixels which are centrally located in objects, and tend to be small near boundaries of objects. For example, in the MR image of Figure 3.1, the scalp tends to show up as bright along the center of the scalp, but darker as you move towards the scalp boundaries. Similarly, the brain stem tends to show up as bright along its center, but darker near its boundaries. Other objects appear dark on a light background, but since these can be thought of as light objects on a dark background in a "negative" of the image, I will consider for now only light objects on a dark background. The graph of the intensity is shown in the center image. Notice how the bright centers of objects show up as *ridges* on the surface. Let the image be represented by a smooth function $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}$, where D is a rectangular solid. A *ridge flow model* is a method for segmenting D into a disjoint union of primitive regions which are determined by the ridge structure of the graph of f. There are numerous definitions for ridges, but the one I use to illustrate the ideas is based on the height definition, a generalization of local maximum points for real–valued functions. The right–most image shows the ridge structure for the MR image.

3.2.1 Height Ridges

As indicated in Chapter 2, there are a number of choices for ridge definitions. Each of the definitions has its own invariance properties. Invariance with respect to rigid motions is desirable in that our construction of primitive regions should not depend on the orientation of the objects in the image. For example, if an object is the union of primitive regions and if that object is rotated by 45°, then the primitive regions of the rotated object should be the rotated primitive regions for the original object. Invariance with respect to monotonic changes in intensity is also desirable since the construction of primitive regions should not depend on the actual intensities. For example, if the intensity of every pixel of an image is doubled, the primitive regions should not change. Only the level definition was shown to have invariance with respect to monotonic changes in intensity. However, my experimental results seemed to show that the height definition produces a qualitatively better set of ridges than does the level definition. The results in this section will therefore be based on the height definition for ridges.

I give a brief summary of the *height definition for ridges* found in Section 2.3. The partial derivatives of f are denoted by subscripting f with the appropriate variable. If $x = (x_1, \ldots, x_n)$, then f_{x_i} is the first partial derivative of f with respect to x_i. The *gradient* of f is the vector $\nabla f = (f_{x_1}, \ldots, f_{x_n})$. The matrix of second partial derivatives of f, called the *Hessian* of f, is denoted by $\mathrm{H}(f) = [f_{x_i x_j}]$. The function f is said to be C^k if its partial derivatives through order k are continuous functions.

The following definitions for local maxima and local minima are from standard calculus. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a C^2 function. A point $x_0 \in \mathbb{R}^n$ is a *critical point* for f if $\nabla f(x_0) = 0$. The function has a *minimum (maximum)* at x_0 if $\mathrm{H}(f(x_0))$ is positive (negative) definite. Any such point is called an *extreme point* and the value $f(x_0)$ is called an *extreme value*.

The concept of local extrema for f can be generalized. Let $v_1, \ldots, v_n$ be a set of n linearly independent vectors in $\mathbb{R}^n$. Let V be the $n \times n$ matrix whose columns are the v_k. The set of solutions to $\nabla f(x) = 0$ are identical to those of $V^t \nabla f(x) = 0$ since V is invertible. Moreover, the matrices $\mathrm{H}(f(x))$ and $V^t \mathrm{H}(f(x)) V$ have the same definiteness by Sylvestor's Theorem (Horn & Johnson 1991). Therefore, the tests for local extrema can be rephrased as follows. A point x_0 is a minimum (maximum) point if $V^t \nabla f(x_0) = 0$ and $V^t \mathrm{H}(f(x_0)) V$

is positive (negative) definite.

Now consider a set of d linearly independent vectors $v_1, \ldots, v_d$ where $1 \leq d < n$. I want points for which f has extreme values, but *only* with respect to the specified directions. Let V be the $n \times d$ matrix whose columns are the vectors v_1 through v_d. Say that f has a *relative minimum (maximum) of type* $n-d$ at x_0 if $V^t \nabla f(x_0) = 0$ and $V^t \mathrm{H}(f(x_0))V$ is positive (negative) definite. Such points x_0 are called *relative extreme points of type* $n-d$ *for* f *with respect to* V. The classification of extreme points has the same form as the case $d = n$, but the x_0 are now solutions to d equations in n unknowns. The solution sets are usually $(n-d)$–dimensional manifolds, hence the use of "type $n-d$" in the definition.

Note that the extrema depend on choice of vectors v_k, which hopefully are natural to the application. In my segmentation application I require 1–dimensional ridge structures, so $d = n-1$. Let $\lambda_i(x)$ and $v_i(x)$, $1 \leq i \leq n$, be the eigenvalues and eigenvectors of $\mathrm{H}(f(x))$ such that $\lambda_1 \leq \cdots \leq \lambda_n$. Let v_1 through v_{n-1} be the vectors used in testing for relative maxima of f. A point x is a *ridge point* (of type 1) if $\lambda_{n-1}(x) < 0$ and x is a relative maximum of type 1 for f with respect to $V = [v_1 \cdots v_{n-1}]$. Since $V^t \mathrm{H}(f)V = \mathrm{diag}\{\lambda_1 |v_1|^2, \ldots, \lambda_{n-1}|v_{n-1}|^2\}$, and since the eigenvalues are ordered, the test for a ridge point reduces to $V^t \nabla f(x) = 0$ and $\lambda_{n-1}(x) < 0$. Additionally, x is a *strong ridge point* if $|\lambda_{n-1}| > |\lambda_n|$, otherwise it is a *weak ridge point*. If all the eigenvalues are negative at ridge point x, it is automatically a strong ridge point. But if $\lambda_n > 0$, the definition intuitively says that a strong ridge point occurs in a region where the convexity of the graph dominates the concavity.

As a mathematical example, consider the function $f(x, y) = x^2 y$. It can be shown that the points $(0, y)$ for $y < 0$ are strong ridge points, whereas the points $(\pm y\sqrt{2}, y)$ for $y > 0$ are weak ridge points. (See Example 2.1 in Section 2.3.4.) As indicated earlier, Figure 3.1 shows the ridge points for the MR image of the head.

3.2.2 Flow Regions

The construction of primitive regions is analogous to the reverse gravity watershed methods found in the peak flow models (Fredericksen et al. 1990, Pizer et al. 1990) and in the network models (Colchester 1990, Griffin, Colchester & Robinson 1991). In the peak flow model, the region points are labeled by starting at a point and following the flow line, whose direction

is the gradient of intensity, to a local maximum. Since the gradient points in the direction of maximum increase for f, the flows are in the "uphill" direction. In the network models, the boundaries of regions are constructed by following flow lines in the gradient direction from local minima to saddles (course lines) and from saddles to local maxima (ridge lines). The points contained in such a boundary lie on flows which are also in the uphill direction.

In my application, following the gradient of intensity direction is not helpful in constructing primitive regions. Generally, a curve consisting of height ridge points is not an integral curve of $\nabla f(x)$. It is possible for a flow line corresponding to $\nabla f(x)$ to transversely cross a height ridge and continue to flow *away* from the ridge (*cf.* the examples in Chapter 2). Even if ridges were flow lines, it must be that the flows can only intersect at a critical point for f (for sufficiently smooth f). The ridges would never take part in the segmentation and the process reduces to just peak flow.

Associated with our ridge construction is a flow vector field whose flows terminate orthogonally at the ridges. The function

$$Q_r(x) = \sqrt{\sum_{i=1}^{n-1} [v_i(x) \cdot \nabla f(x)]^2}$$

has the property that $Q_r(x) = 0$ at ridges. The vectors $-\nabla Q_r^2(x) = -2Q_r(x)\nabla Q_r(x)$ are orthogonal to the level sets of Q_r, so such flows will meet the ridges as indicated. However, I want the flows to terminate rather than originate at the ridges; that is, the flows should generally be in uphill direction on the graph of intensity. To force the flows to terminate at ridges, I adjust the sign on $-\nabla Q_r(x)$ so that the flow has positive component in the $\nabla f(x)$ direction, thereby forcing the flows to move in the uphill direction. In the continuous model, the flows are determined by

$$\frac{dx}{dt} = \text{sign}[\nabla f(x) \cdot \nabla Q_r^2(x)]\nabla Q_r^2(x),\ t > 0,\ \ x(0) = x_0,$$

where x_0 is the initial position for the flow. Note that the minus signs from the two occurrences of ∇Q_r^2 cancel. In the discrete model where the computations are on a rectangular lattice, there are only $3^n - 1$ discrete directions to flow in. The direction selected is the one which minimizes the angle between the true and discrete directions.

The ridge structures are initially segmented into curvilinear segments which are then assigned distinct labels. In the discrete implementation, candidate ridge points are represented

as pixels in a binary image. The sets are not necessarily 1–pixel thick, a requirement for identifying the curvilinear segments. I designed a thinning algorithm which takes the candidate ridges and thins them so that the resulting set can be easily segmented into curvilinear segments. The thinning algorithm preserves the topology of the original set in that the number of holes is invariant. The algorithm also thins sets from outside to inside, thereby preserving the general shapes of the initial sets. A detailed discussion of the thinning algorithm can be found in Section 5.4.

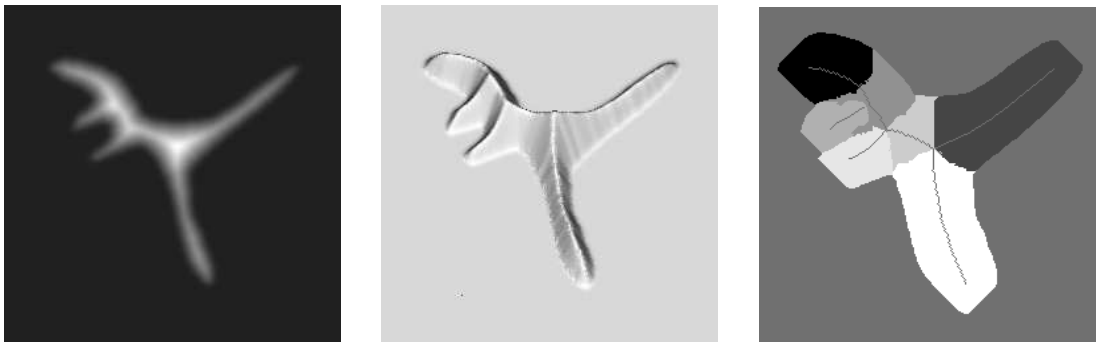
After the ridges are segmented and labeled, the flow algorithm is executed. At each pixel in the image the flow line is followed until the flow intersects a ridge. Every pixel along the path is assigned the label of the terminal ridge point. In this way every pixel obtains a label and the image is segmented into primitive regions. Each region contains the pixels which flow to the ridge of the same label.

EXAMPLE 3.1: In Figure 3.2, the left image is a simulated gray scale image which was constructed by taking a binary blob object, computing the Euclidean distance transform of it, and smoothing the transform by blurring with a circular Gaussian kernel of standard deviation 2 pixel widths. The middle image shows a shaded rendering of the graph of the image intensities. The right image shows the height ridges and associated ridge flow regions. Small scale ridges due to the discretization of the algorithm were eliminated by keeping only those ridge points x for which the largest eigenvalue of the Hessian of the image was suitably large. □

This example gives an idea of the primitive regions for a simple image. The ridge flow algorithm applied to the image in Figure 3.1 yields approximately 1500 flow regions, so it is difficult to get a feeling for the distribution of regions. On a color workstation an idea of the distribution can be obtained by using a random coloring of the regions.

3.3 Ridge–Valley Flow Models

Segmentation using only ridges produces primitive regions which typically overestimate where object boundaries are. This is to be expected since the flow regions tend to have boundaries at valleys on the graph of intensity, whereas the object boundaries tend to occur



Gray scale image | Intensity surface | Ridges and flow regions

Figure 3.2: Ridge segmentation of gray scale image

somewhere between ridges and valleys. As an illustration in one dimension, consider the graphs in Figure 3.3.

Rather than segmenting an image based only on its ridge structures, segment it using both ridges and valleys. In Section 3.2.1 I gave a definition for height ridges. A similar definition for height valleys is the following, and uses the same notation as in the ridge definition. Let λ_1 through λ_n be the eigenvectors of $\mathrm{H}(f(x))$ with corresponding eigenvectors v_1 through v_n. Let v_2 through v_n be the vectors used in testing for relative minima of f. A point x is a *valley point* (of type 1) if $\lambda_2 > 0$ and x is a relative minimum of type 1 for f with respect to $V = [v_2 \cdots v_n]$. The test for a valley point reduces to $V^t \nabla f(x) = 0$ and $\lambda_2(x) > 0$. Additionally, x is a *strong valley point* if $|\lambda_2| > |\lambda_1|$; otherwise it is a *weak valley point*. If all the eigenvalues are positive at valley point x, then it is automatically a strong valley point.

For each ridge or valley segment I want to assign a primitive region which is built by following flows, but now the flows should terminate orthogonally at ridges and at valleys. The flow region boundaries will occur between ridges and valleys in positions which are natural to the underlying flow field. In Section 3.2.2 I showed that ridges are zeros of the function $Q_r(x)$. Valleys are zeros of a different function,

$$Q_v(x) = \sqrt{\sum_{i=2}^{n} [v_i(x) \cdot \nabla f(x)]^2}.$$

Therefore, ridges and valleys can be located simultaneously by finding the zeros to $Q(x) =$

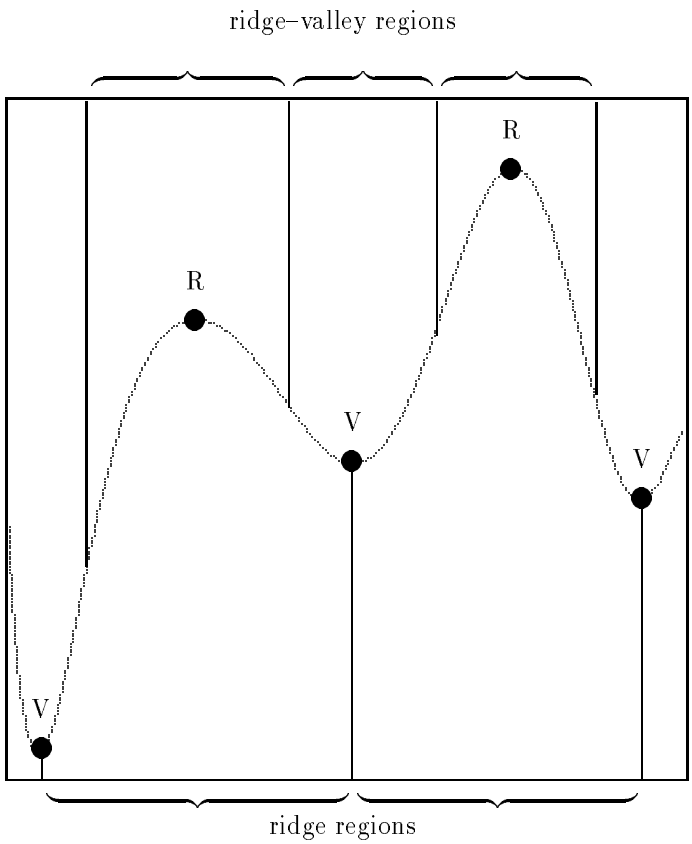


Figure 3.3: Comparison of flow regions

$Q_r(x)Q_v(x)$. The flow field vectors are given by $-\nabla Q^2(x)$, and the flows are determined by

$$\frac{dx}{dt} = -\nabla Q^2(x),\ t > 0,\ \ x(0) = x_0,$$

where x_0 is the initial position for the flow.

Similar to the ridge flow model, the ridge and valley structures are initially segmented into curvilinear segments which are assigned distinct labels. At each pixel, the flow line is followed until it intersects either a ridge or a valley. Every pixel along the path is assigned the label of the terminal ridge or valley point. Every pixel of the image obtains a label by this method, so the image is segmented into primitive regions. Generally, the number of regions obtained by ridge–valley flow is larger than the number obtained by only ridge flow.

3.4 Image Hierarchy

The image hierarchy for image $f : \mathbb{R}^n \to \mathbb{R}$ is constructed as a multiscale process. The scale is introduced by blurring the initial image via variable conductance diffusion (Grossberg 1984, Perona & Malik 1987, Whitaker 1993). As a continuous process, the blurred image $B(x, \sigma)$ at position x and scale σ is the solution to the initial value problem

$$\begin{aligned} B_\sigma(x,\sigma) &= \sigma\nabla\cdot(C(x,\sigma,B,\nabla B,\ldots)\nabla B(x,\sigma)),\ x \in \mathbb{R}^n,\ \sigma > \sigma_0 \\ B(x,\sigma_0) &= f(x),\ x \in \mathbb{R}^n, \end{aligned}$$

where C is a conductance function (treating the blurring process as a heat transfer process). The initial scale σ_0 is called the *inner scale* of the process. If $C \equiv 1$, then the blurred image is simply the convolution of a Gaussian kernel of standard deviation σ with the input image, $B(x, \sigma) = G(x, \sigma) \oplus f(x)$. I used $C \equiv 1$ in my experiments with images.

In practice, images are specified on a compact set $D \subset \mathbb{R}^n$. The blurring must be computed on D with initial data $f(x)$, but conditions at the boundary ∂D must also be imposed. I choose Neumann conditions $\partial B(x, \sigma)/\partial\eta = 0$ for $x \in \partial D$ and $\sigma > \sigma_0$, where the derivative above indicates a directional derivative taken in the direction normal to the boundary of D. As a heat transfer process, this condition says that the boundary is insulated.

Initially $f(x)$ has small scale geometric details. As σ increases, the geometric details are annihilated, including ridge structures. Since the equation has no heat source, one expects

that as scale becomes infinite, $B(x,\sigma)$ tends to a constant; that is, all details are eventually annihilated. In contrast to the interest of physicists in the steady–state behavior of such differential equations, I am concerned with short–time properties which will directly affect how the hierarchy is built.

The algorithm to construct the hierarchy is as follows. Select a geometric sequence of N scales, say $\sigma_j = \sigma_0 b^j$ for some $b > 1$ and for $1 \leq j \leq N$. The final scale σ_N is called the *outer scale* for the process. For each σ_j, $j \geq 0$, compute the blurred image $B(x,\sigma_j)$ and segment it using the ridge flow model to obtain $D = \cup_{i=1}^{M_j} R_i^{(j)}$ where $R_i^{(j)}$ is a primitive ridge flow region and $M_j \geq 1$ is the number of primitive regions at scale σ_j. Identify each primitive region $R_i^{(j)}$ with a node at height j in a general tree. The leaf nodes of the tree correspond to the primitive regions occurring at inner scale σ_0.

The links in the tree are inserted based on how primitive regions at one scale become blurred into single regions at the next scale. Two primitive regions at one scale may blur together to form part of a single primitive region at the next scale. This single primitive region is considered to be the parent of the original two regions since it overlaps those two regions more than any other region at the current scale. More specifically, to link the nodes (regions), the parent of region $R_i^{(j)}$ will be a region $R_m^{(j+1)}$ which overlaps $R_i^{(j)}$ more than any other region $R_\ell^{(j+1)}$. At the outer scale σ_N, one expects very few regions in the segmentation. The root node of the tree is the parent for all primitive regions at the outer scale. Thee root node essentially corresponds to a constant image with a single region (infinite scale).

The implementation of the tree construction is a discrete algorithm which is an approximation to the continuous model described here. Two technical problems can arise as a result of the discretization. First, it is possible in the tree construction that blurring from one scale to the next does not annihilate features (in at least one subregion). Thus, the tree may contain nodes which have exactly one child. Second, even though the variable conductance diffusion process is *causal* (no new features are introduced as scale increases), the discretized process may introduce regions whose corresponding tree nodes have no children. These nodes are not essential to the hierarchy and can be removed; consequently all leaf nodes of the tree correspond to primitive regions only at the initial scale.

EXAMPLE 3.2: Consider a hypothetical hierarchy, shown in Figure 3.4, produced by the ridge

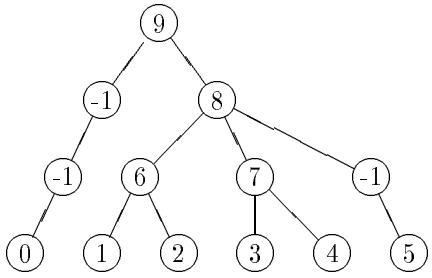


Figure 3.4: Hierarchy for an image

flow segmentation algorithm. The leaf nodes and internal nodes with 2 or more children are labeled with distinct positive integers. The -1 labels indicate that the node has a single child, so the regions of the subtree have not blurred together with any other regions of the hierarchy.

At inner scale σ_0, the number of primitive regions is $M_0 = 6$. Each region was assigned to a leaf node in the tree. After blurring the image at scale σ_1, the ridge flow segmentation produced $M_1 = 4$ primitive regions. Leaf regions 1 and 2 blurred together so that region 6 (at the next scale) overlapped them the most. Similarly regions 3 and 4 blurred together and were overlapped the most by region 7. However, regions 0 and 5 did not significantly blur together with other regions, so they are not linked to other regions. Intuitively, small adjacent and similar regions should be linked at small scales, but large adjacent and similar regions should be linked at large scales. The number of primitive regions at scale σ_2 is $M_2 = 2$. Regions 6, 7, and the one corresponding to the parent of 5 blurred together, so in effect leaf regions 1 through 5 blurred together at this scale. Region 0 is still dissimilar to the regions represented by the other subtrees at height 2, so its subtree is again an only child. Finally, no more blurring was performed, so the root node is added and corresponds to a constant image (single region) at infinite scale. □

3.5 Object Definition and Visualization

The image and corresponding ridge flow hierarchy are used as input to two experimental tools designed for object definition and visualization, the Interactive Hierarchy Viewer (Freder-

icksen et al. 1990, Pizer et al. 1990) and the Magic Crayon (Beard et al. 1993, Beard et al. 1994). Both tools provide a graphical user interface which allows the user to traverse the hierarchy and quickly "color" objects of interest. The Interactive Hierarchy Viewer was developed to run both on workstations and on Pixel Planes, an extremely fast graphics computer designed and built at the University of North Carolina. One of its major functions is to allow the user to interact with the hierarchy to rapidly define and extract anatomical objects from the data set, then to quickly volume render the data with Pixel Planes. The Magic Crayon was developed to run on workstations in a clinical setting. In addition to allowing the user to rapidly define and extract anatomical objects, the tool provides the means for making accurate volume and surface area measurements of the extracted objects. Both tools work on the principle that objects of interest can be identified as unions and differences of subtrees of the hierarchy. Of course the details of the data structures of the tree and the subtree identification are hidden from the user. The remainder of the section illustrates how one uses the Magic Crayon to define objects by manipulating the hierarchy in a way that identifies the relevant subtrees.

As a simple example, consider the hierarchy shown in Figure 3.4. The original image is displayed by the Magic Crayon, and the hierarchy is already loaded into memory. The user moves the mouse cursor to a region of interest in the image and clicks the left button. Suppose that the pixel at the cursor is in region 1. All pixels in region 1 are subsequently colored. The user has the option of coloring nearby related regions by clicking on an "add more" button. This corresponds to moving up the hierarchy from node 1 to its parent, node 6, and then traversing to all children of node 6. All regions corresponding to the children are drawn. In my example, only region 2 is newly visited, and all pixels in region 2 are colored. The left tree of Figure 3.5 shows the hierarchy after the initial click on a pixel in region 1. A shaded node indicates that the corresponding region has been colored. The right tree of Figure 3.5 shows the state of the hierarchy after the "add more" operation. If instead the original colored region is 5, an "add more" operation does not produce any newly colored regions. At height 1 in the tree, region 5 did not merge (through blurring) with any other regions. However, a second "add more" operation will merge region 5 with regions 1 through 4.

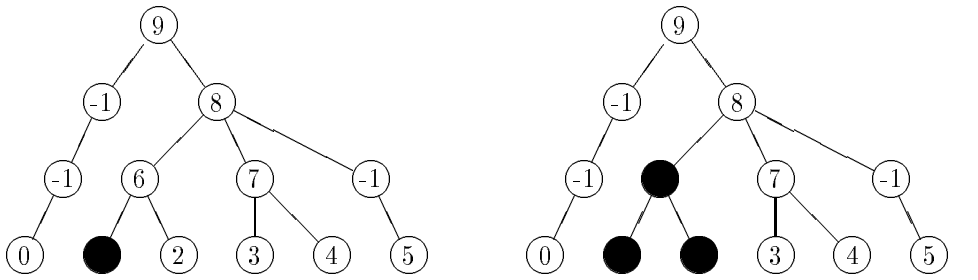


Figure 3.5: Hierarchy traversal and coloring

More generally, given that the user is at height H in the hierarchy, an "add more" operation corresponds to moving up the hierarchy from the current node to its parent at height $H+1$, followed by traversal of all subtrees of the parent. Any leaf nodes visited during the traversal, for which the corresponding regions are not colored, are marked as visited and the regions are all colored. If the user is far up the hierarchy, sometimes unwanted objects are colored. The Magic Crayon provides "undo" operations and also allows for user–initiated editing of the current colored objects.

For some anatomical objects, moving up the hierarchy will not completely color the object. For example, if the user attempts to color the brain stem in the image of Figure 3.1, a single click on a region at height 0 followed by "add more" operations will not completely color the stem. The user must start the process again, at height 0, in an uncolored portion of the object. The Magic Crayon provides a more efficient way of drawing elongated objects like the brain stem. A operation is provided for moving up the hierarchy from many regions colored initially at height 0, called the "add more all" operation. The user can press the left mouse button and drag the cursor through the object. All encountered regions are colored, so quite a few leaf nodes are visited in the hierarchy. By pressing the "add more all" button, the user moves up the hierarchy to parent nodes for all of the currently colored regions; that is, the "add more" operation is applied at multiple regions.

As an illustration, suppose that regions 1 and 3 were initially colored. After an "add more all" operation, regions 1 through 4 will be colored. The left tree of Figure 3.6 shows the hierarchy after the initial clicks on pixels in regions 1 and 3. The right tree of Figure 3.6

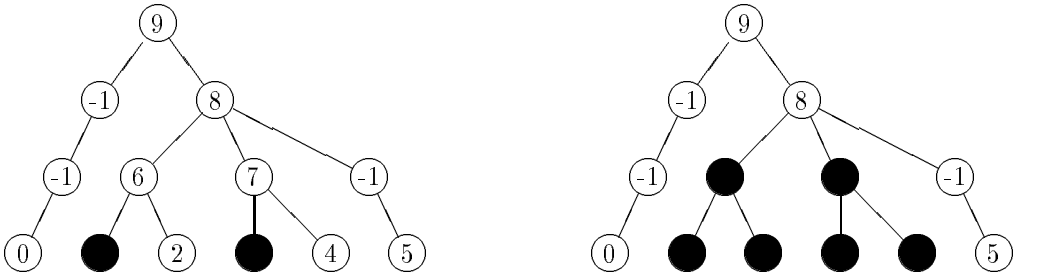


Figure 3.6: More hierarchy traversal and coloring

shows the state of the hierarchy after the "add more all" operation.

In addition to the "add more" controls, the user can delete subtrees using a Magic Eraser. For each of the addition operations there corresponds an analogous deletion operation. By using both sets of controls, the user in effect can move freely through the hierarchy marking the appropriate subtrees that make up the objects of interest. The final marked nodes yield unions and/or differences of subtrees of the original hierarchy.

The Magic Crayon currently does not support coloring multiple objects, but the Interactive Hierarchy Viewer does. (The Interactive Hierarchy Viewer, however, does not have an "add more all" operation.) Figure 3.7 shows some colored objects from the 2–dimensional image of Figure 3.1. The image was segmented using the ridge flow model with Gaussian blurring. The number of primitive regions at initial scale $\sigma = 1$ is 1450 for ridge flow and 2698 for ridge–valley flow. I used a geometric sequence of 8 scales with multiplier $b = 1.1$. Each object was colored by starting at various places, clicking on a primitive region, and adding or deleting regions by traversing the hierarchy. The total object coloring required about 2 minutes of user interaction.

The same image was segmented using the ridge–valley flow model with Gaussian blurring. Figure 3.8 shows an attempt at defining the objects shown in Figure 3.7. The total time for coloring was about 5 minutes. The primitive regions were smaller, so it took more time to try to fill in the objects close to their boundaries.

I also segmented a 3–dimensional image containing 20 slices, each slice being 256×256. The number of primitive regions at initial scale $\sigma = 1$ is 31679. I used a geometric sequence

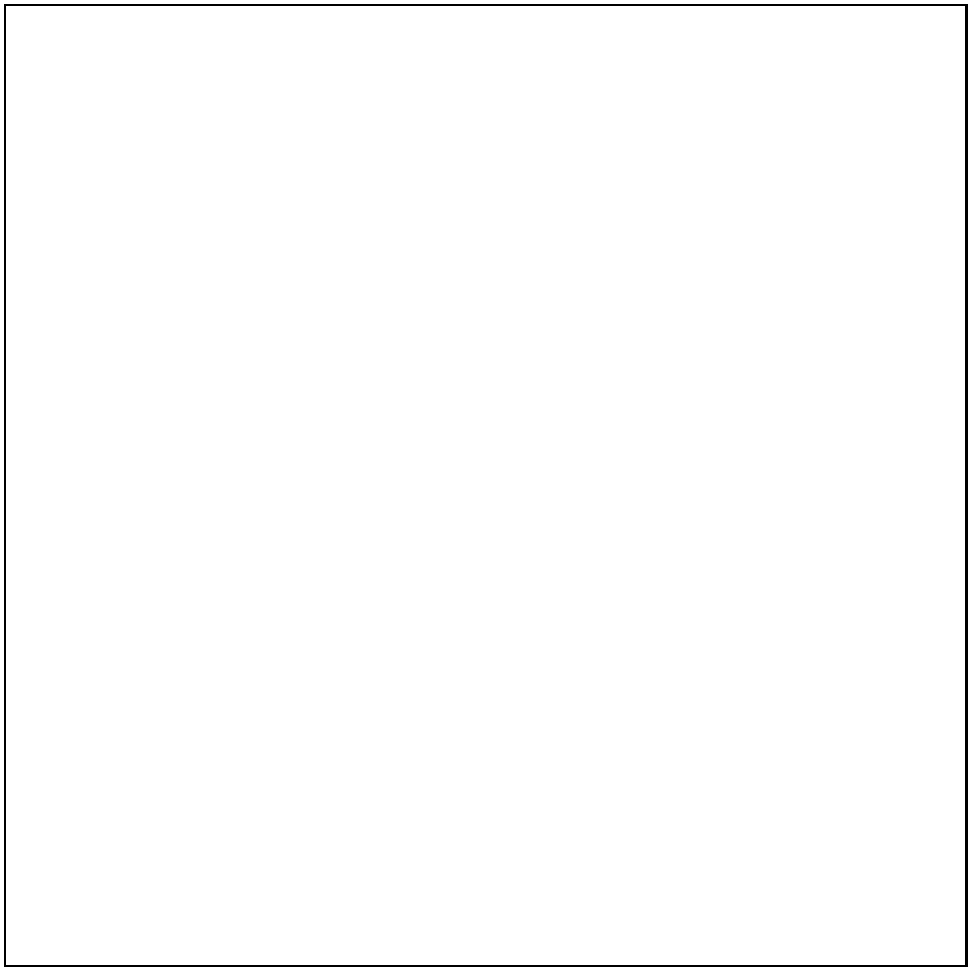


Figure 3.7: Visualization of hierarchy subtrees (ridge flow)

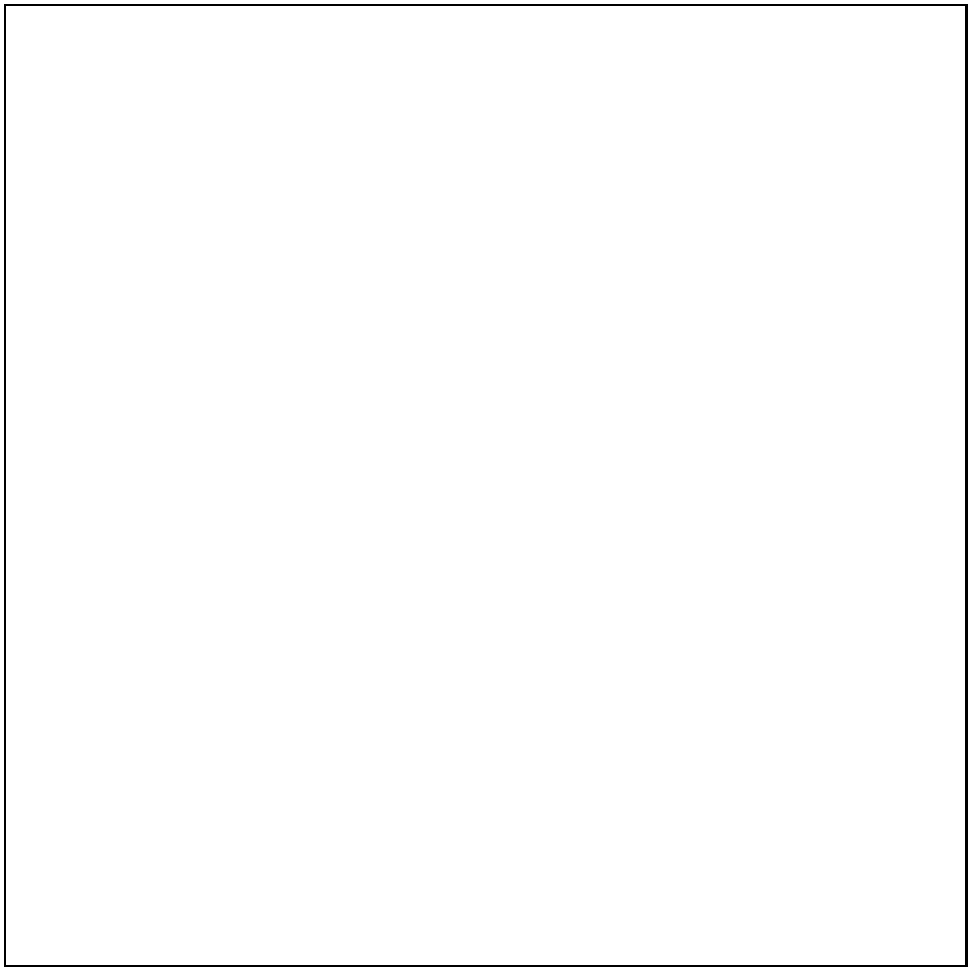


Figure 3.8: Visualization of hierarchy subtrees (ridge–valley flow)

of 16 scales using multiplier $b = 1.1$. Therefore, the hierarchy is a tree with 17 levels and 31679 leaf nodes. Figure 3.9 shows four consecutive slices (8,9,10,11) of the original image. The panel of buttons is part of the Magic Crayon interface.

The mouse was initially dragged down the middle of the brain stem in slice 9, coloring primitive regions along the way. Since the primitive regions are 3–dimensional, the corresponding regions in the other slices have also been colored, including the slices that are not currently displayed (but buffered in memory). Figure 3.10 shows the colored primitive regions.

The next sequence of 5 figures corresponds to clicking the "add more all" button. The colored regions correspond to the visited subtrees of the hierarchy. Note that on the last click the coloring appears to be fairly solid.

The final two figures show the colored regions on slices that were not currently displayed. Figure 3.16 shows slices 12 through 15 and Figure 3.17 shows slices 4 through 7 for the current status after clicking 5 "add more all" operations.

The Magic Crayon was run on a DECstation 5000/125 running at 25 MHz and rated at 21 MIPS (machine name is "phong"). The machine has 224 Megabytes of physical memory and 1 Gigabyte of virtual memory. The initialization of the Magic Crayon takes about 2 minutes. During this time the hierarchy and primitive region image are read from disk files and converted to internal representations. Bounding boxes are built for each primitive region to help speed up access time during later stages of program execution.

Once the program is loaded and the X–Window interface appears, the user is free to explore the hierarchy. The time to color the brain stem region as shown in Figure 3.15 took about 10 seconds. Of course more time will be taken to completely segment the brain stem. The user can color the regions in the currently displayed slices, then move to other slices and start from new leaf nodes, traverse the hierarchy, and fill in the object on those slices. For this particular data set, it took me about 2 minutes to identify (what I think is) the brain stem.

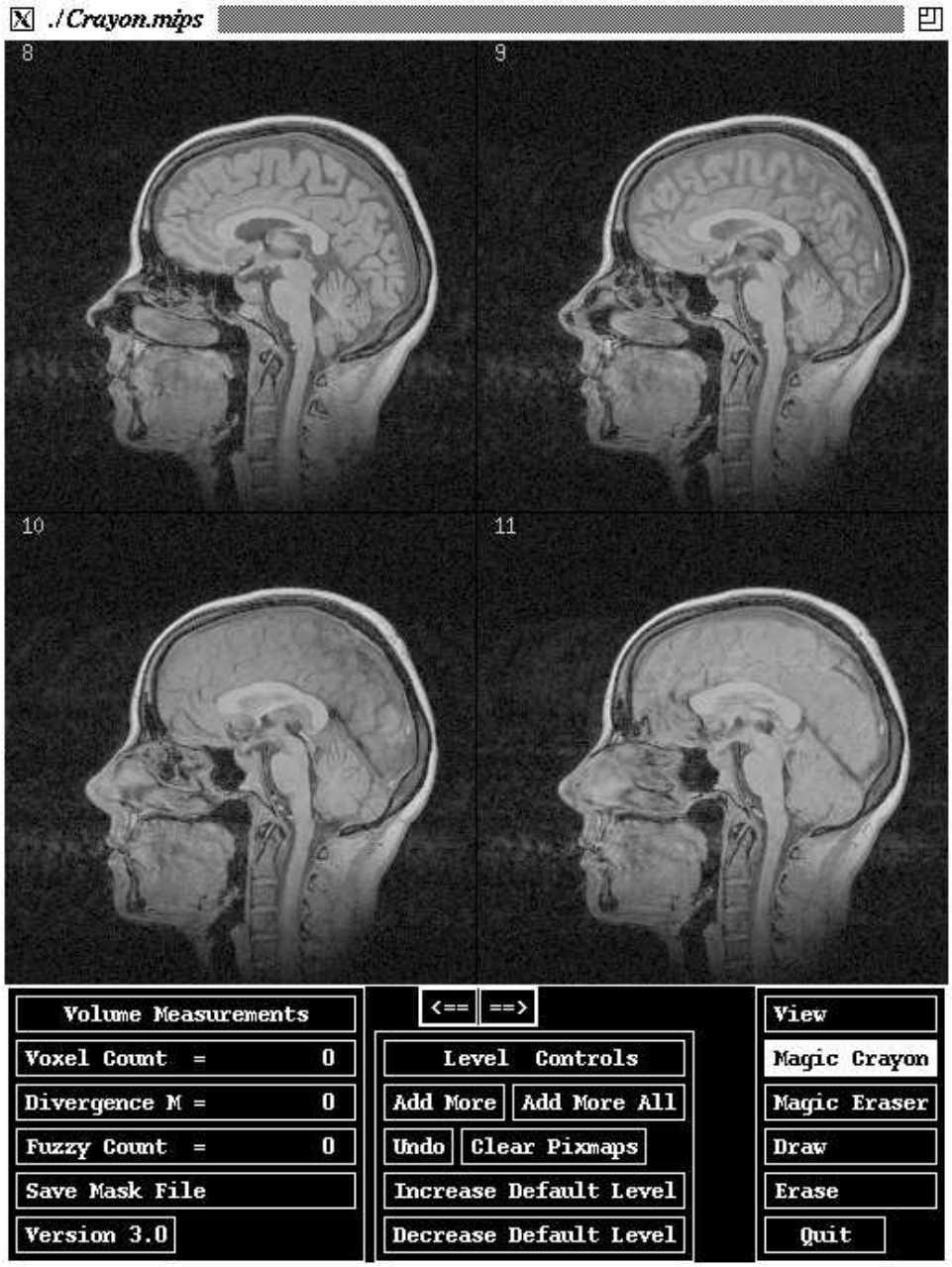


Figure 3.9: Magic Crayon Interface with 3D head image

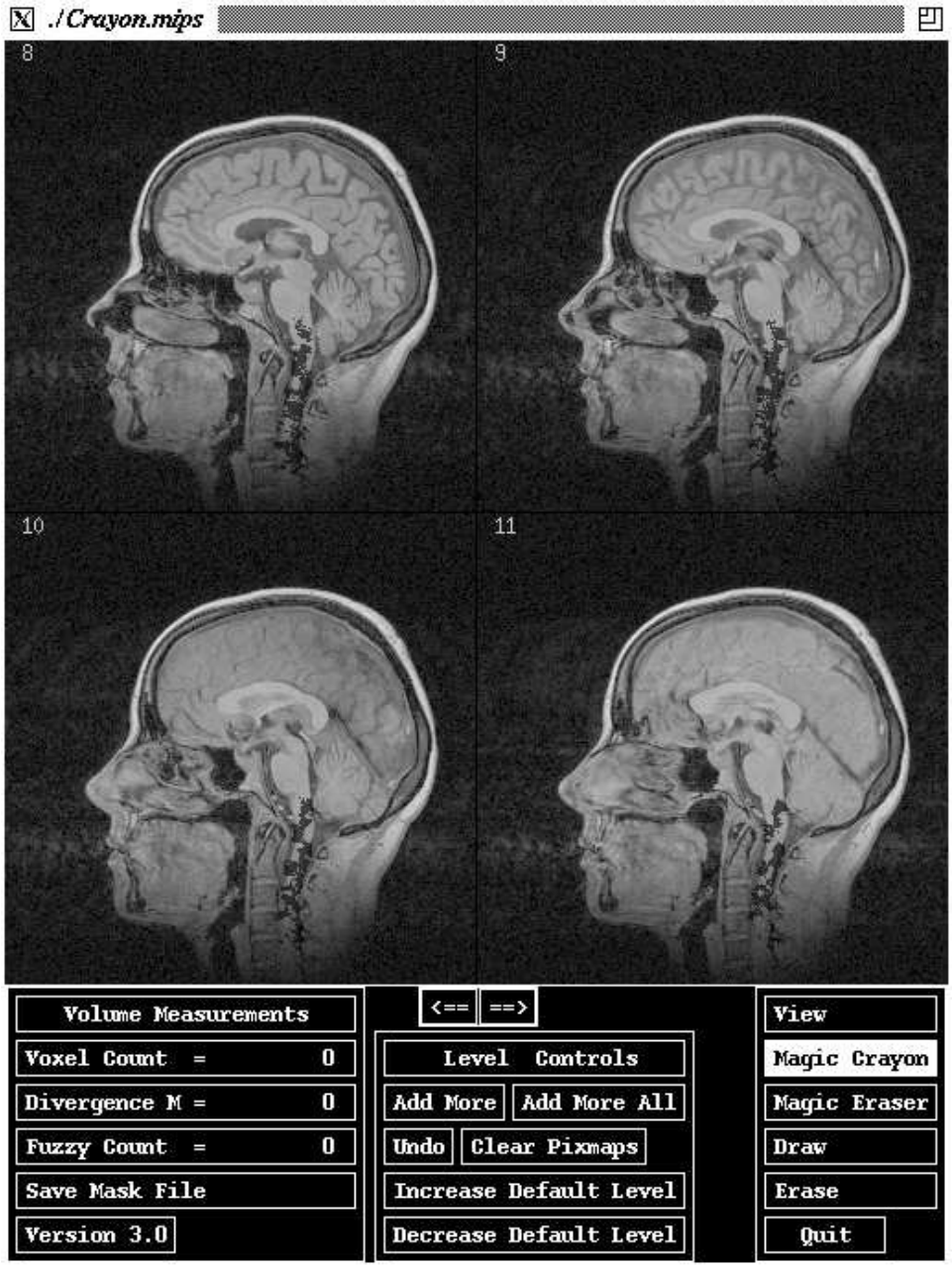


Figure 3.10: Colored regions after brain stem partially colored

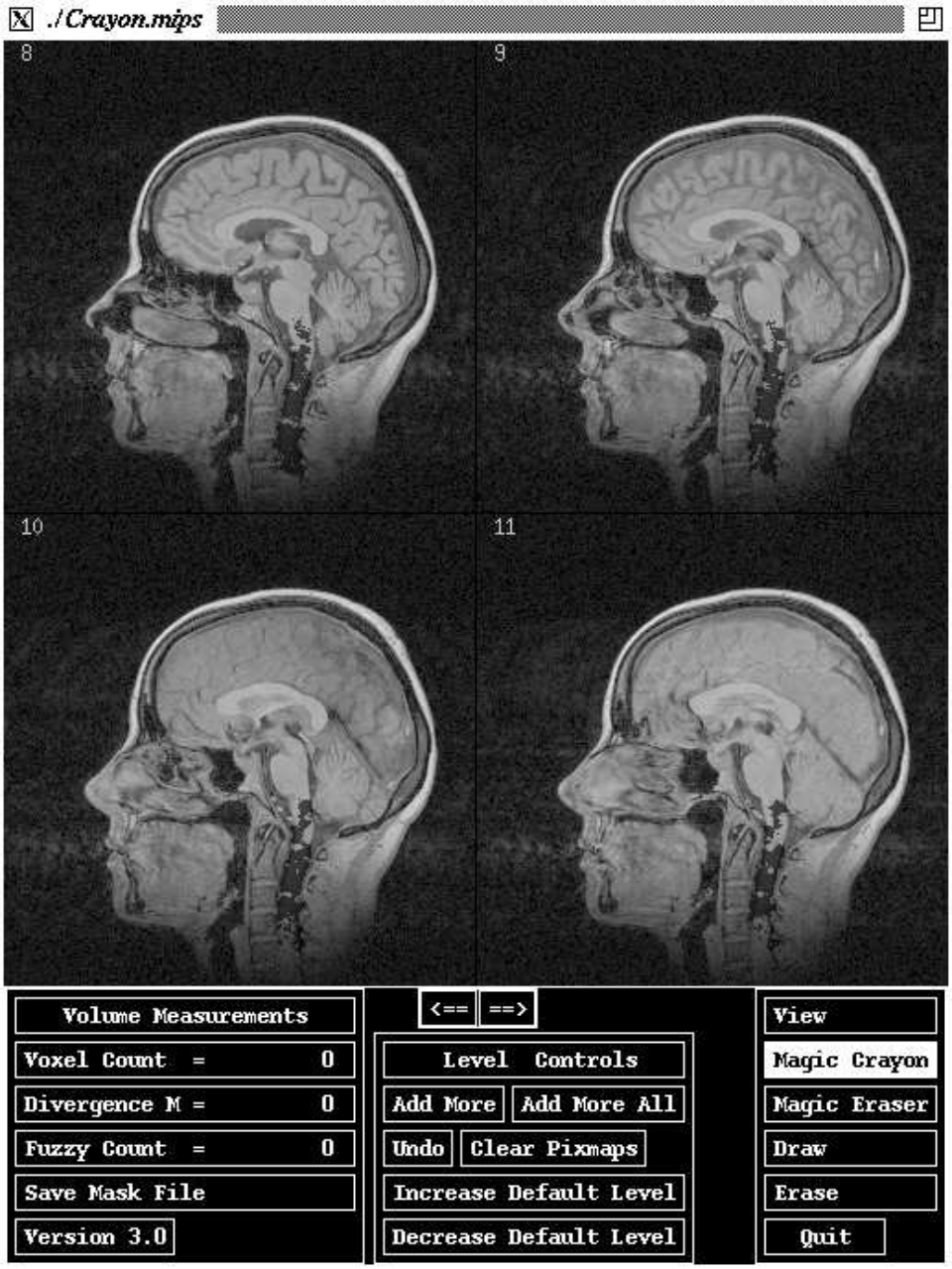


Figure 3.11: Regions after add–more–all 1

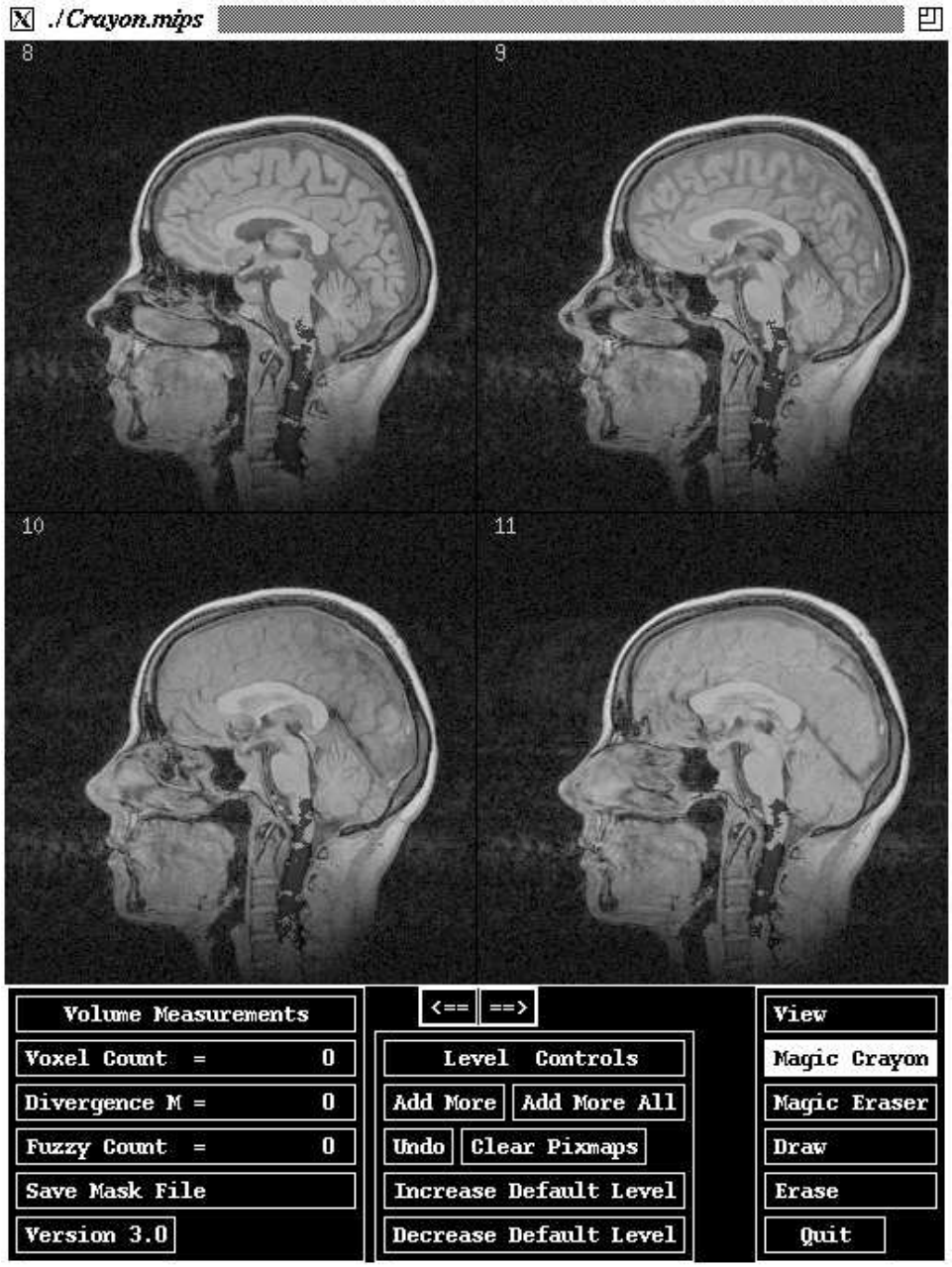


Figure 3.12: Regions after add–more–all 2

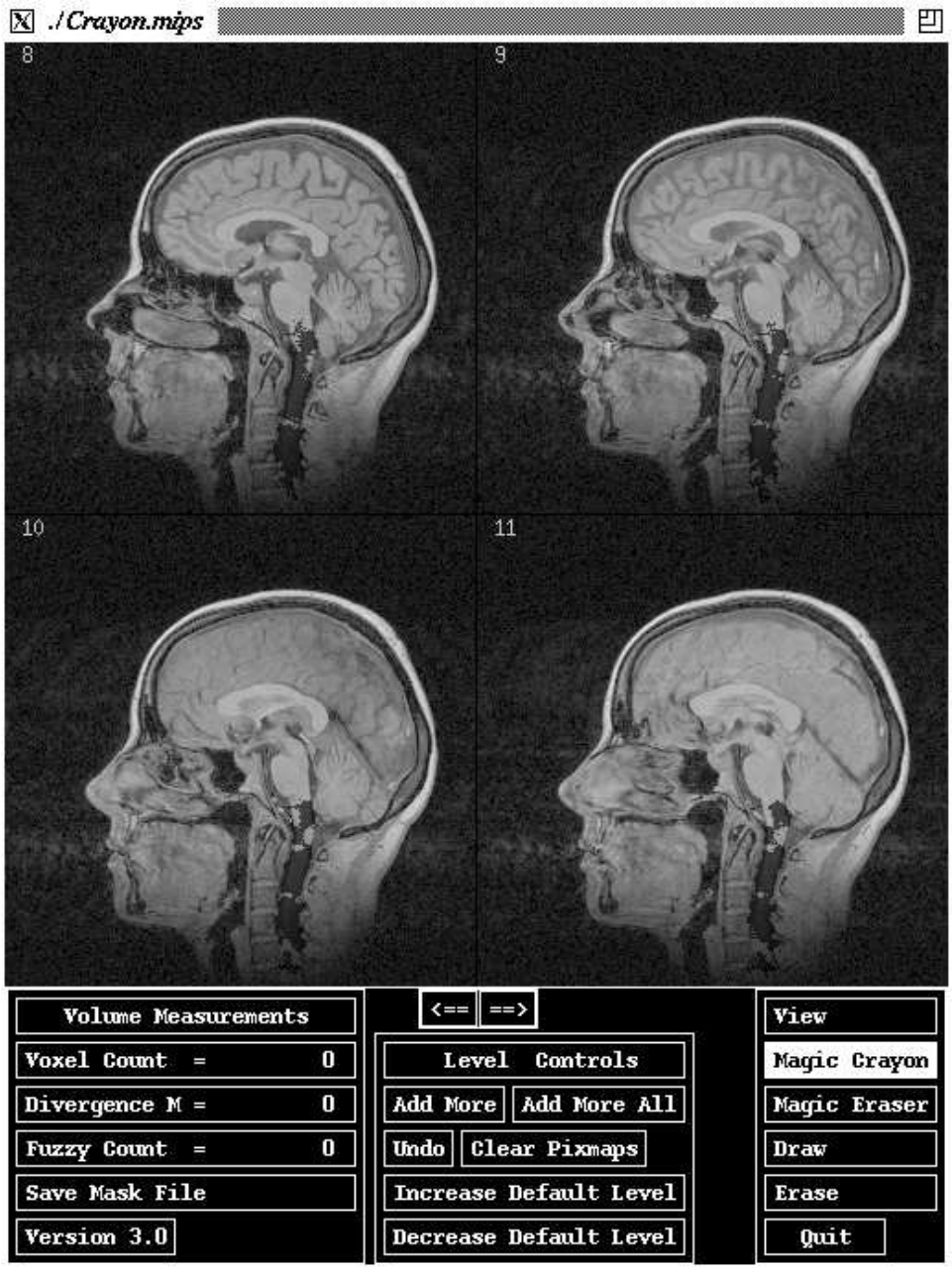


Figure 3.13: Regions after add–more–all 3

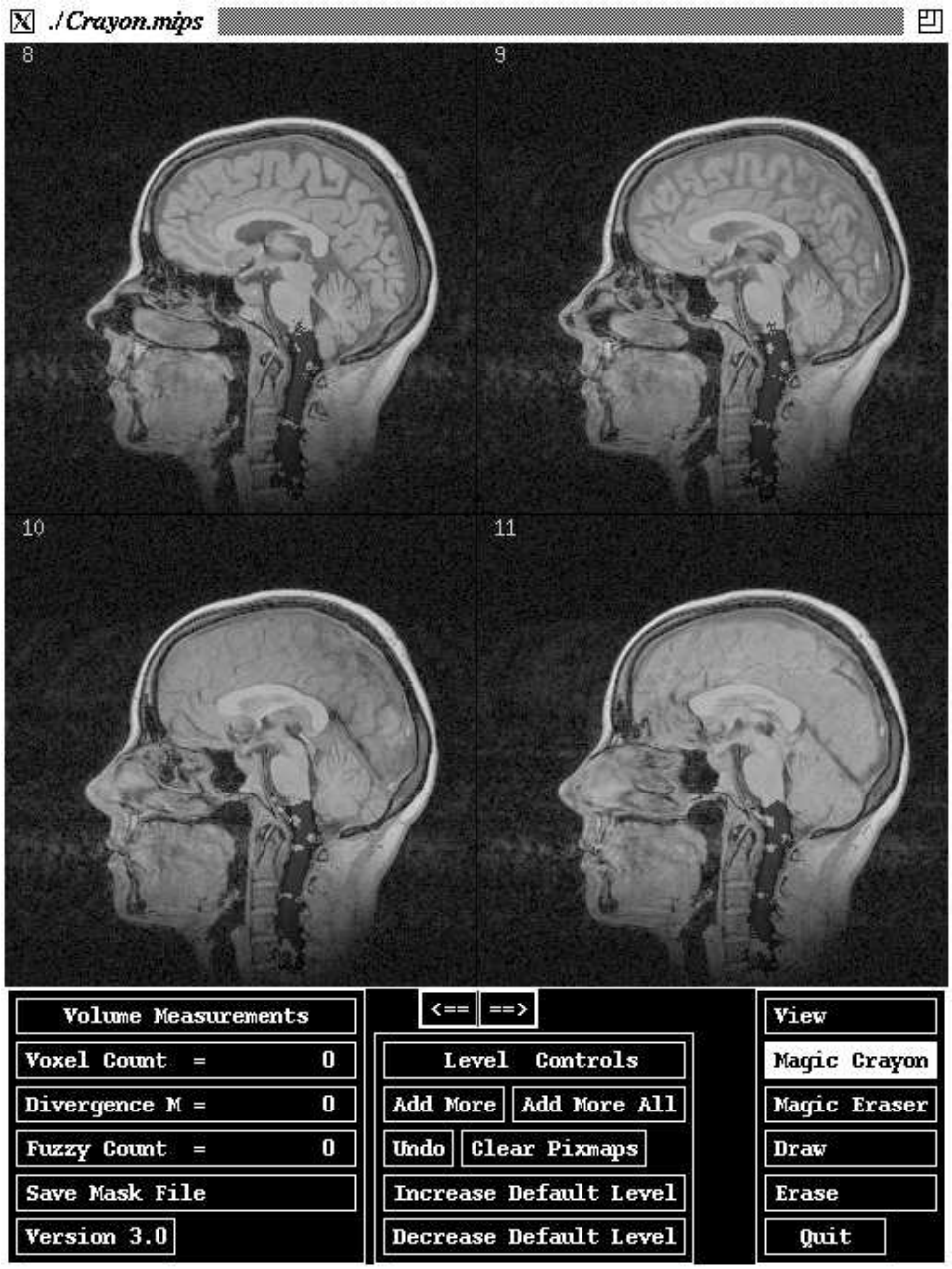


Figure 3.14: Regions after add–more–all 4

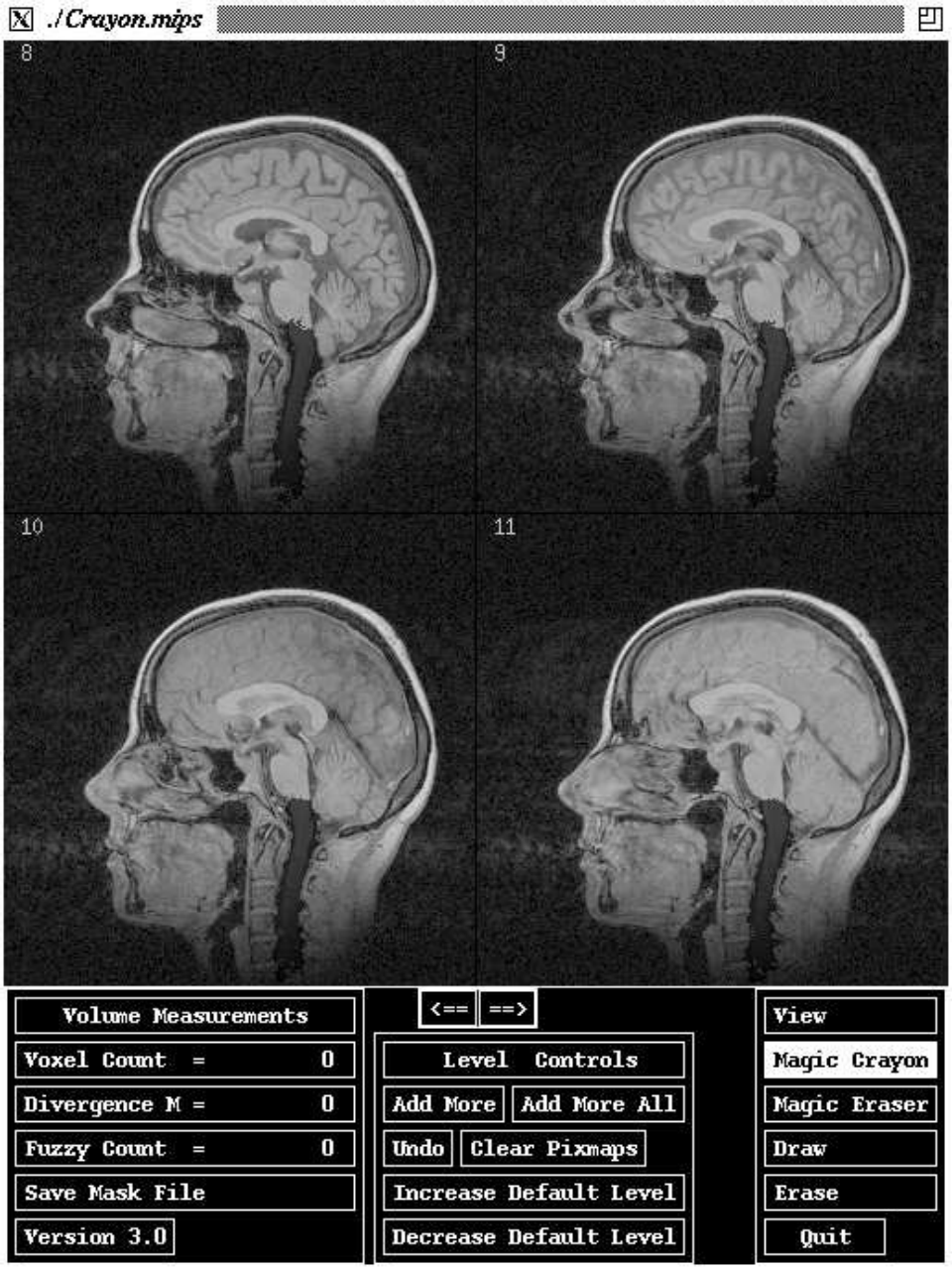


Figure 3.15: Regions after add–more–all 5

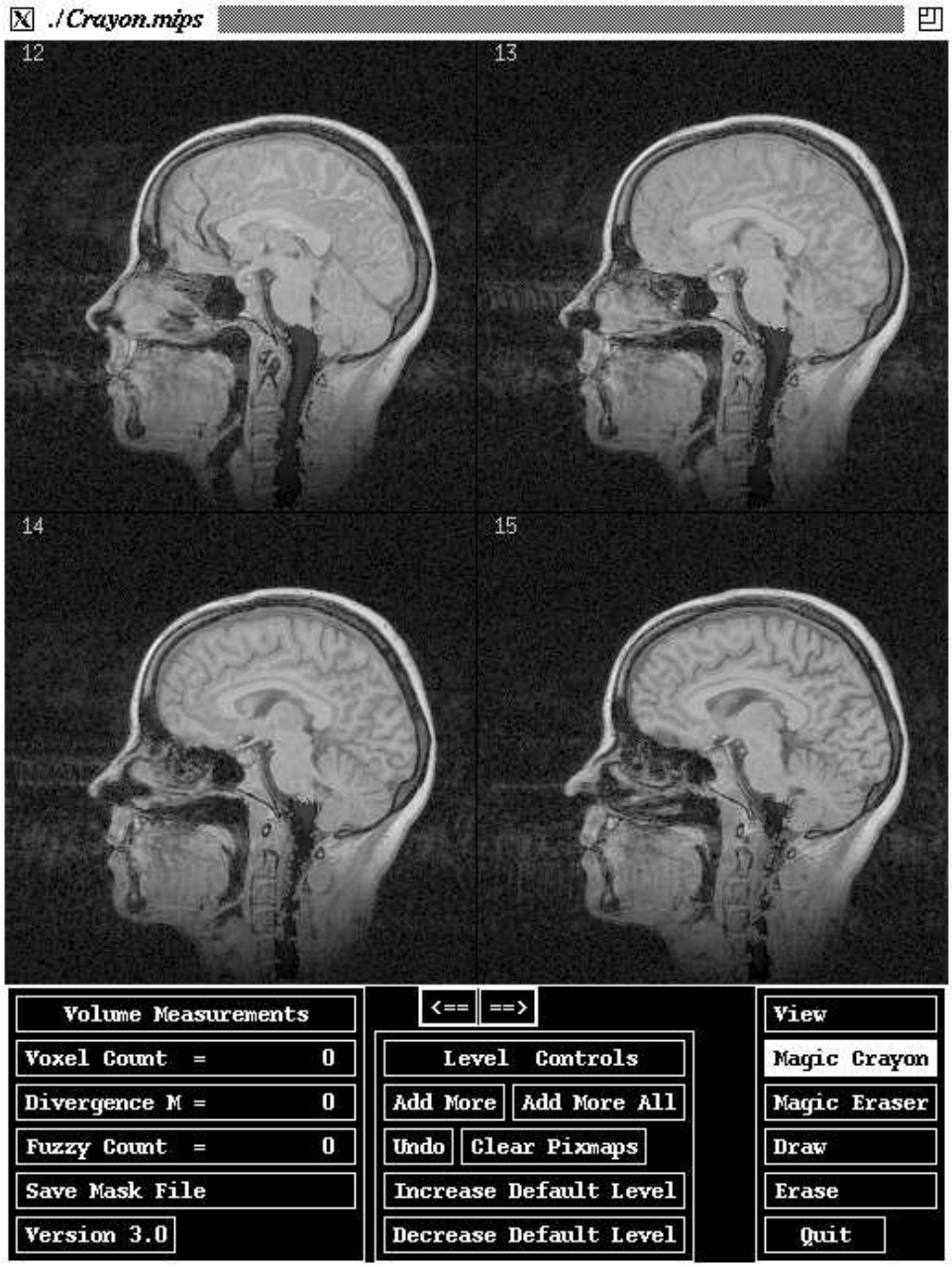


Figure 3.16: Slices 12–15 after add–more–all 5

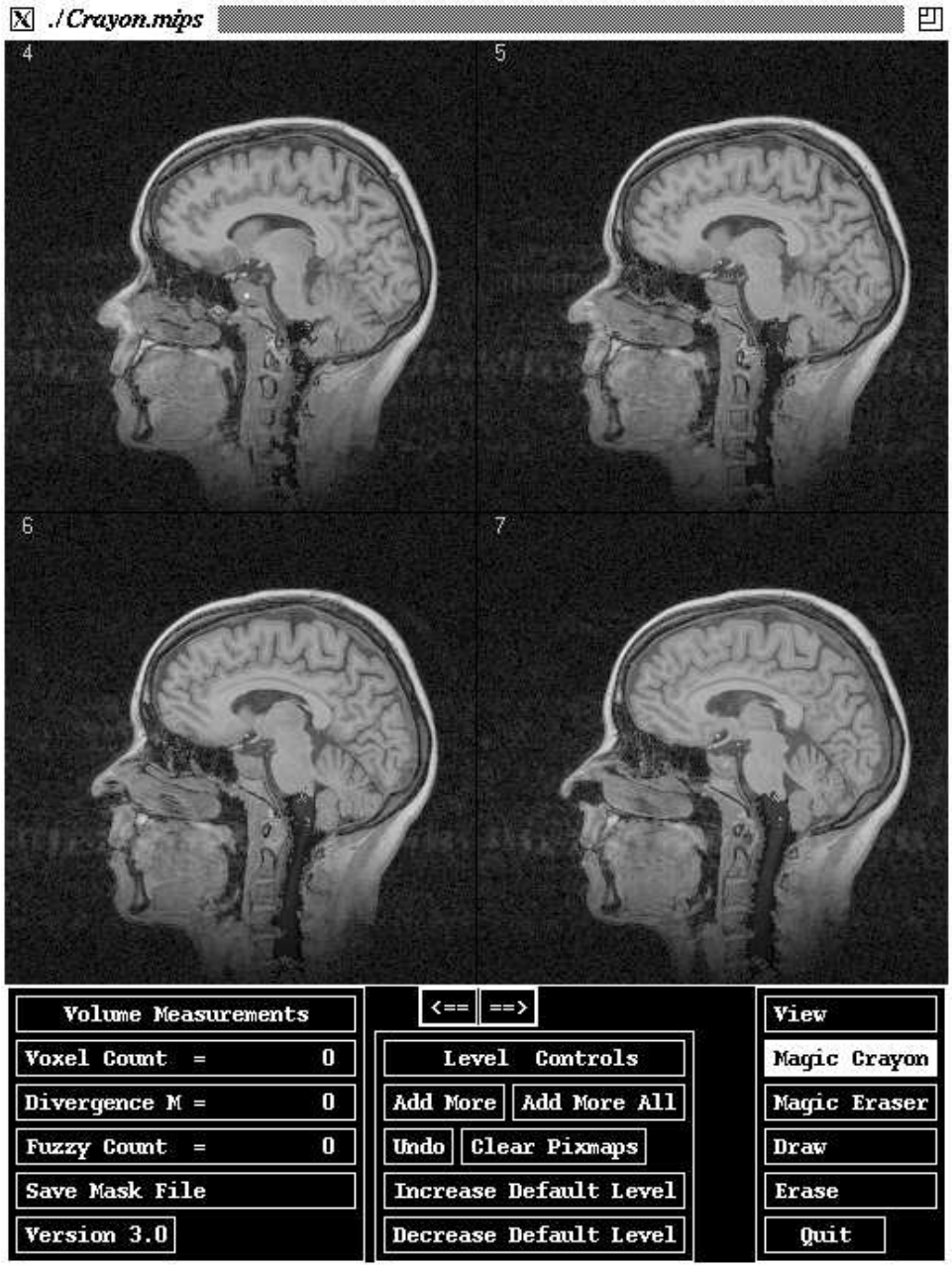


Figure 3.17: Slices 4–7 after add–more–all 5

3.6 Discussion

The ridge flow algorithm was applied to two slices from different 3–dimensional MR images of heads, to two full 3–dimensional MR images of heads, to a 3–dimensional abdominal image, and to a 3–dimensional angiogram. In all these images my experiments with the ridge flow algorithm indicate that it successfully identifies objects whose boundaries and background have a moderate–to–good contrast. Objects whose boundaries contrast poorly with background are not as precisely constructed. The problem is that primitive regions sometimes "bleed" outside one object into the background, or into a neighboring object. This bleeding is a function of the current blurring method, Gaussian blurring. It is a linear process which typically averages object information with background data when applied near an object boundary. A better blurring method is needed which is sensitive to boundaries and will not average across them. If an object boundary is approximately a level curve of intensity, then a good conductance function is one which will force the level curves to propagate in the directions normal to the curves with speed proportional to the local curvature. A consequence is that object shapes tend to be preserved throughout the blurring process. The diffusion equation, $B_\sigma = \sigma|\nabla B|\nabla \cdot (\nabla B/|\nabla B|)$, does have the desired properties. I propose this as a better alternative blurring method for ridge flow or for ridge–valley flow. A long term goal is to use variable conductance diffusion of higher–order geometric image information (Whitaker 1993). Additionally, the image could be preprocessed with local contrast enhancement algorithms such as adaptive histogram equalization (Pizer, Amburn, Austin, Cromartie, Geselowitz, Greer, ter Haar Romeny & Zimmerman 1987) or generalized order–statistic filters (Longbotham & Eberly 1992, Longbotham & Eberly 1993).