Automatic Neurosurgery Planning by Finding the Safest Path in 3D Images

Yasmine El-Glaly
yasmineg@vt.edu

Abstract

In this paper, we investigate the problem of automatically finding the safest path from an incision point to a target point in order to help the surgeon plan and practice before performing the neurosurgery in real. The input to our proposed system is a segmented 3D image that the incision point, the region of interest, and the vital functional areas are marked on it. Our proposed method will convert the 3D image into weighted graph. Then we will find the shortest path using Dijkstra algorithm.

1. Introduction

With a wide range of applications of computer in the graphics, medical imaging technology has also developed a lot. It helps surgeons to complete operations more intuitively and accurately, simplifying the complex process of surgery. Planning of neurosurgery path is an important aspect and application of medical image processing. It could find the best path through scientific basis to reduce the difficulty and risk of surgery and to improve surgical precision. Planning brain surgery together with other medical image processing, needs to reconstruct three-dimensional structure of the brain based on accurate data of two-dimensional image by equipments like CT and MRT, and then to do the analog image denoising processing through mathematical algorithms as well as the final three-dimensional display.

Functional magnetic resonance imaging (fMRI) has become a versatile noninvasive tool for studying the functionality of the brain and for localizing cognitive functions. Nowadays, fMRI is not only used in the neurosciences but also in applications such as presurgical planning. For example, in neuro-oncologic brain surgery, the goal is to maximize tumor resection or to perform epilepsy surgery while preserving important brain functions. Presurgical fMRI can be used to localize motor, sensory, and language-control areas, and has been used to study cerebral reorganization in tumor patients. In general, activation in the brain is indirectly measured by utilizing the blood oxygenation level dependent (BOLD) effect as a natural contrast sensitive to neural activity [1].

Diffusion tensor imaging (*DTI*) is a magnetic resonance imaging (MRI) technique that enables the measurement of the restricted diffusion of water in tissue in order to produce neural tract images instead of using this data solely for the purpose of assigning contrast or colors to pixels in a cross sectional image. It also provides useful structural information about muscle—including heart muscle, as well as other tissues such as the prostate. In DTI, each voxel therefore has one or more pairs of parameters: a rate of diffusion and a preferred direction of diffusion—described in terms of three dimensional space—for which that parameter is valid. The properties of each voxel of a single DTI image is usually calculated by vector or tensor math from six or more different diffusion weighted acquisitions, each obtained with a different orientation of the diffusion sensitizing gradients [2].

It is becoming normal to register images from fMRI with anatomical MR images to analyze and identify important regions pertaining to the neurosurgery, see figure 1. Recently fiber tracking, the reconstruction of microstructural characteristics in the brain and central nervous system are achieved using diffusion tensor imaging (DTI) [2].

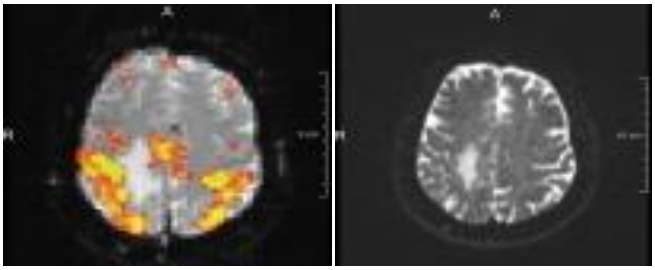


Figure 1 (a) fMRI data (b) DTI data

2. Related Work

Brain Surgery Path Planning System is the most important part of The Computer-Assisted Surgery. The relevance for automated path planning becomes clear in the work of Brunenberg et al. [3]. Their partnering neurosurgeons estimate a time gain of 30 minutes during pre-operative planning when using the authors' method for automated trajectory proposition, which effectively means that the planning time is halved. A segmentation of anatomy is performed in terms of blood vessels, ventricles and sulci of the cortex. An Euclidean distance map allows the calculation of paths with maximum distance to critical structures and thus minimal risk.

Rieder et al. [4], proposed a technique to suggest the safest path to the damaged area in the brain without putting the functional areas at risk. After visualization phase is completed as discussed in section 4, the surgeon now can search for the optimal path between the point of incision and the ROI point. In the internal view, the path is represented by a line connecting the two points, while in the external view it is represented by a cylinder. The path is very flexible, the view is updated accordingly, and the path thickness (cylinder radius) and direction can be changed.

The beauty in their method is that the surgeon can explore the path while observing the functional areas around the path at the same time. After the path is computed and visualized as shown in figure 2, they add landmarks to the brain and the skull to give more support for the surgeons.

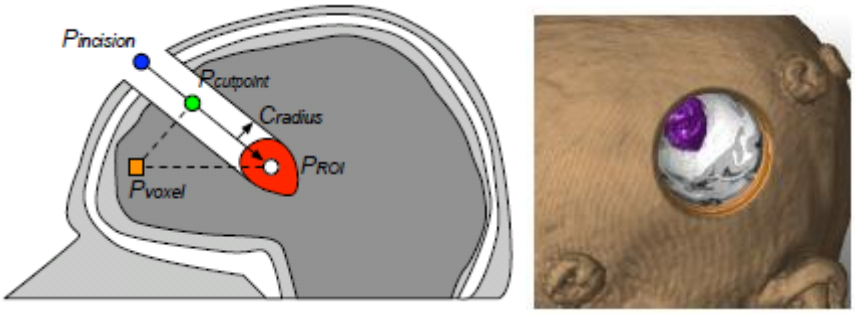


Figure 2. (a) Computation of the cylindrical path from entry point to region of interest. (b) The cylindrical cut geometry [4].

Regarding the visualization methods, the paper didn't actually present a new technique although they combine previously known enhancement methods in one integrating system. The authors assumed that the path to the region of interest should be a straight line. Their system needs to be more generalized to make it possible for

the user to explore more complicated forms of paths, putting into consideration that a tumor can exist in a deep difficult area in the brain and the surgeon must take more than straight line move to reach it. Also, it will be more convenient if the system enable the user to compare between different paths at the same time to choose the safest path in case there is no optimal path.

3. Specific Aims

We suppose that the volumetric data obtained from medical scanners to patient's brain e.g. fMRI and DTI are used to construct 3D images. After visualizing the patient's data, further processing is conducted to mark the functional areas in the brain. We obtain segmented images that detect the important parts within the visualized data.

The objective of the research is to automatically planning the path for the neurosurgery with minimal interaction from the surgeon. Thus we want to explore the different possible paths from the incision point to the target point without passing critical areas and then rank the safest path which minimizes the risk factor.

4. Methodology

The inputs to the system we built which are defined by the user are as follows:

1. The "Can't be touched" points; which are the points that the surgeon must not go through it because of its vital importance for the patient.
2. The entry point; which is the point the user will select to begin the surgery from it i.e. the incision point.
3. The target point; which is the point the surgeon wants to achieve to remove a tumor or the damaged part of the brain.

The main steps we followed to compute the safest path between the incision point and the target point can be summarized as follows:

1. Convert the input segmented image into graph. Each voxel in the segmented image will represent a node in the graph.
2. Assign weights over the edges according to the vitality of the area that the edge is crossing.
3. Compute the shortest path between the input node and the target node as given by the user.

From this new perspective, we have transformed our problem into a single source shortest path problem.

In order to perform the ***first step*** in our designed algorithm efficiently, we tried to decrease the number of nodes in the to-be constructed graph to save the computational power.
So, instead of naively convert every voxel in the image into a node, we will represent every group of voxels with one node in the corresponding graph based on a similarity measurement. This problem can be solved as an optimization problem [5]. For every two voxels in the input image, they will be connected into one group if the change in the density between the two voxels is minimized, under the constraint that they are 26-neighborhooded connectivity as shown in figure 3.
The similarity measurement can be calculated using the following optimization function:

$$\int \left[(\frac{\partial^2 F}{\partial x^2})^2 + (\frac{\partial^2 F}{\partial y^2})^2 + (\frac{\partial^2 F}{\partial z^2})^2 \right] dF \rightarrow \min .$$

Where F is the given image, x,y, and z are the three dimensions of F.

To discretize the previous equation, the central finite difference is used as shown in the following equation:

$$\sum_{ijk} \begin{aligned} & [(f_{i+1jk} + f_{i-1jk} - 2f_{ijk})^2 + \\ & (f_{ij+1k} + f_{ij-1k} - 2f_{ijk})^2 + \\ & (f_{ijk+1} + f_{ijk-1} - 2f_{ijk})^2] \rightarrow \min . \end{aligned}$$

The equation is solved such that the following constraint:

The voxels are connected by 26 neighborhood connectivity

Each group will then be represented by one node.

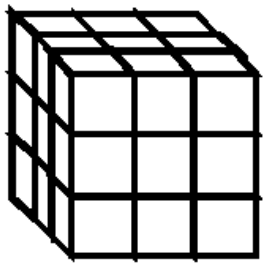


Figure 3. 26 connectivity

Moving to the ***second step***, it could be done manually by the user or automatically by the system. Following the first option, the user can mark the region(s) of interest and give an arbitrary value for each region.

If the user chooses the automatic building for the weight matrix, the system assigns large scalar value for the edges crossing the vital areas in the brain. While the edges that cross safe areas with minimal risk are assigned to very small scalar values. By such procedure, the algorithm is encouraged to select the safest path for the in planning neurosurgery.

The system definitions for the safe/critical regions in the image are based on the voxels value. This is done by thresholding function to determine which regions belong to the different parts that can be detected from the MRI images such as the white matter, gray matter, bone, tissues, nerves, and lesion. More details about implementing these two steps can be found in the appendix.

Finally, the ***third step*** of the proposed algorithm is implemented using Dijkstra algorithm [6]. We will: 1) Add a label for each node, 2) calculate the distance function to get the shortest path which is the safest path in our problem.

As we got from the previous steps the positions of the nodes and the weights of the edges, we use the index of the participating pixel as its label. This is very important in our case so that we can traverse the safest path on the medical image after we define it on the graph. The Dijkstra algorithm can be simply summarized as follows:

1. Assign to every node a distance value. Set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes as unvisited. Set initial node as current.
3. For current node, consider all its unvisited neighbors and calculate their distance (from the initial node). If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.
4. When we are done considering all neighbors of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.
5. If all nodes have been visited, finish. Otherwise, set the unvisited node with the smallest distance (from the initial node) as the next "current node" and continue from step 3.

5. Results and Discussion

All the images used during this research are for anonymous patients. The images are in DICOM (Digital Imaging & Communications in Medicine) format. It is a standard format for medical images and widely used by the different systems, printers, scanners and workstations [7]. Sample data can be found in the medical dataset in [8]. The developed system is implemented using MATLAB 7.10.0 and Image Processing Toolbox 4.1. The system is experimented on 2.0 GHz computer with 3.0 GB RAM under Windows Vista operating system.

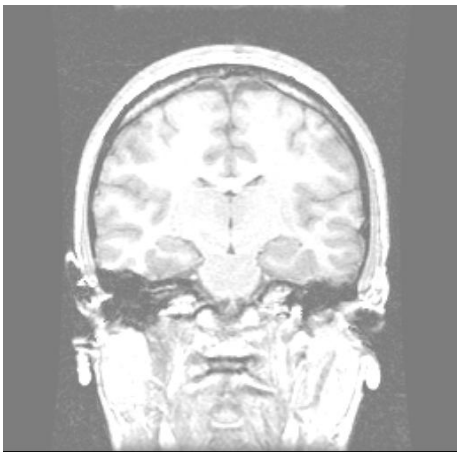
Beginning by applying the first step in our algorithm, we find that grouping the voxels with very similar values into one region has their mean value, made huge difference in the performance of the shortest path algorithm.

In our experiments, we examined two cases. In the first case we convert the input image to a graph where every pixel in the image became a node. That means that if an input image with low resolution e.g. 500*500, we will need to build a matrix of size 250000*250000 to get the sparse matrix of this new graph or at least 250000*8 to indicate the full connectivity between each pixel and its neighbors.

In the second case, we apply the grouping procedure as described in section 4. Using the same image with the same resolution, it is converted into graph with only 100 nodes. Its sparse matrix is of size 100*100, and the full connectivity matrix is 100*8.

Downsizing the number of nodes in the graph with this rate leads to great enhancement in the performance of the system.

In figure 4, it is shown a sample MRI scan for an anonymous patient that is displayed using DICOM viewer [9] in part (a) and its computed segmentation in part (b) using Matlab code.



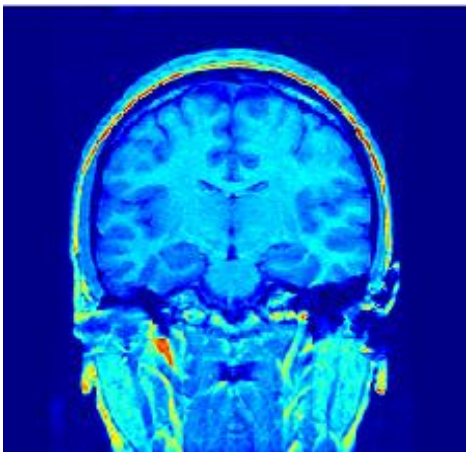


Figure 4. a)MRI scan b) Computed MRI scan

With respect to the second step in our algorithm, we should note that there is a compromise between the two options in establishing the weight matrix. The first option where the user has to input manually the values for the regions of interest is exhaustive for the user. This is because he is supposed to precisely mark every separate region and choose a suitable value for it, where the regions are interleaved in almost all images. Regarding the second option, it is very fast and does not need user interaction. But the thresholding function is mainly tailored for the MRI dataset we used in our research; according to the DICOM format. Also, if there exist noise in the MRI data, it could mislead the thresholding function and thus the weighting function.
From our experiments, we find that the weighting matrix is more accurate when specified by the user though it is much more time consuming.
In figure 5, we show a shot screen of our program where the user is enabled to choose specific regions using different tools such as the ellipse, rectangle, and line.

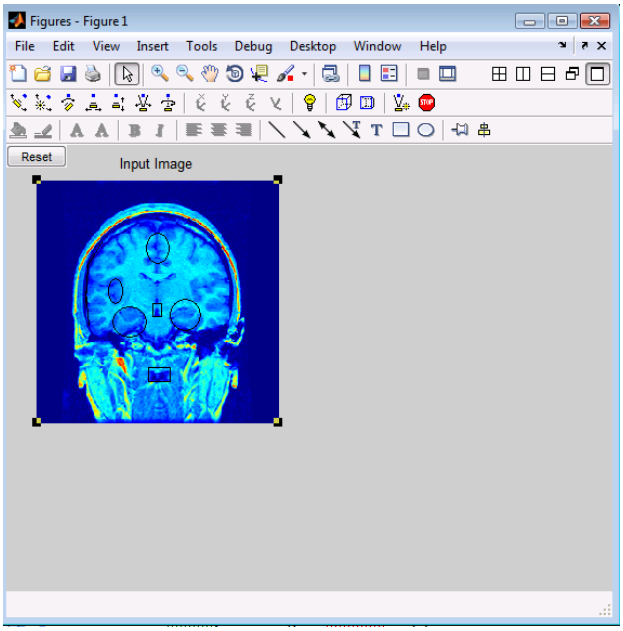


Figure 5. Manual selection for regions of interest.

In order to navigate through the different slices of the MRI scans, we use a sliceomatic which is a volume slice visualization GUI. It can be downloaded from Matlab Central official website. This tool was of a great importance to us as it was the

main tool we use to visualize our input and computed data. A screenshot for Sliceomatic is shown in figure 6.

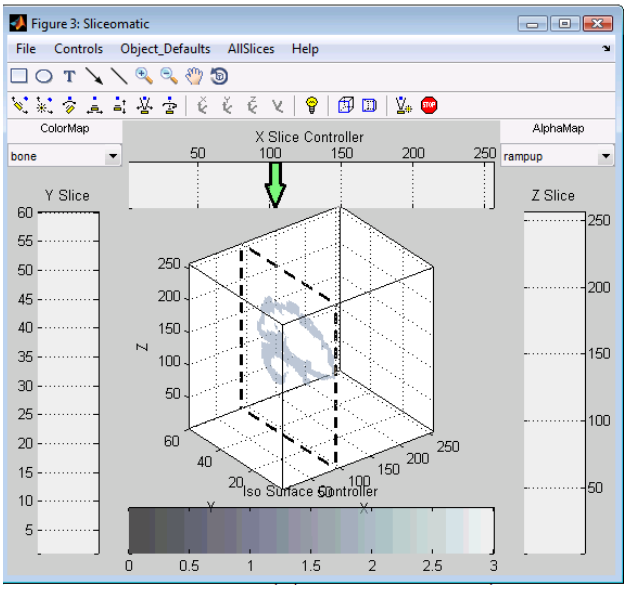


Figure 6. Sliceomatic GUI

Finally, we examine different incision and target points to find the safest path between them among all other possibilities. We applied Dijkstra algorithm directly to the graph obtained from the previous steps. It succeeds to find the safest path as it is apparently does not cross the critical regions. Though visualizing the shortest path between 800 nodes on a graph can be somewhat clear, it is not the case when it is mapped to the medical image. We find that copying the shortest path and plotting it to the original image will result a total mess. Rather, we repeat the first module in our algorithm to get more smoothness in the image and so bigger areas for the regions. This further processing was needed for visualization aspects only. Figure 7 shows that problem clearly. We should also note that in part (b) we choose an incision point that is relatively close to the target point and that what makes that figure neat.

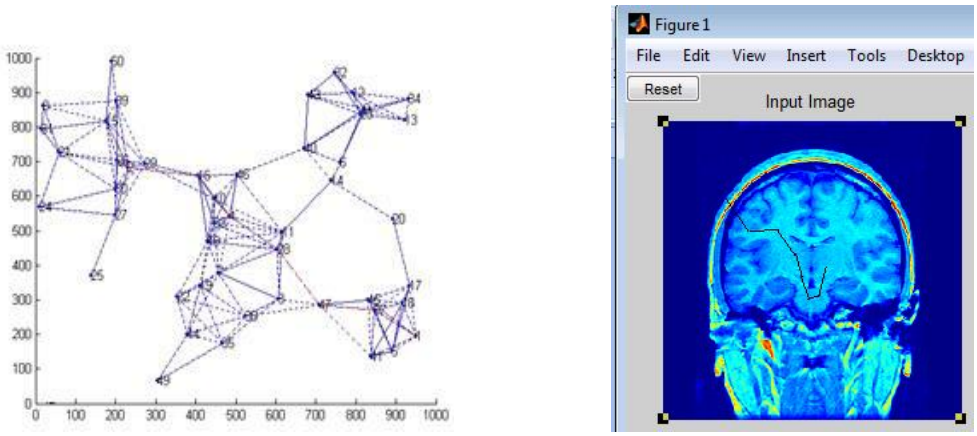


Figure 7. Shortest path using Dijkstra algorithm a) on 1000 nodes and b) on 100 nodes

6. Conclusions

In this paper, we present a new system in the field of computer aided neurosurgery planning. The objective the system achieved is to find the safest path the

neurosurgeon can access during the surgery with minimum risk to the patient. The presented system enhances the overall performance of searching procedure; by first minimize the image nodes to a defined threshold, and then applying one of the most powerful single source shortest path algorithms which is Dijkstra.

In future work, we plan to modify the proposed algorithm so that it can be run in parallel using more advanced hardware. Parallel processing of the data will give us the facility to use images with higher resolution and so more accurate results will be obtained. That is in addition to speed and reliability that parallelism inherently ensures. Also, we are working on more visualization facilities to be added to our system such as traversing the safest path along the slices of the MRI scans so that it can be visualized in 3D.

References

[1] K. Tabelow, J. Polzehl, A. M. Uluˇg,J. P. Dyke, R. Watts, L. A. Heier, and H. U. Voss. 2008. Accurate Localization of Brain Activity in Presurgical fMRI by Structure Adaptive Smoothing. IEEE Transactions On Medical Imaging. Vol. 27, No. 4, April 2008.

[2] T. Schultz, N. Sauber, A. Anwander, H. Theisel, and H.P. Seidel. 2008. Virtual Klingler dissection: Putting fibers into context". Computer Graphics Forum(Proceedings of EuroVis. 27(3):1063–1070.

[3] Ellen J. L. Brunenberg and Anna Vilanova and Veerle Visser-Vandewalle and Yasin Temel and Linda Ackermans and Bram Platel and Bart M. ter Haar Romeny, Automatic Trajectory Planning for Deep Brain Stimulation: A Feasibility Study, Medical Image Computing and Computer-Assisted Intervention (MICCAI), vol. 4791, 2007, pp. 584-592

[4] Christian Rieder, Felix Ritter, Matthias Raspe, and Heinz-Otto Peitgen. 2008. Interactive Visualization of Multimodal Volume Data for Neurosurgical Tumor Treatmen. Eurographics/ IEEE-VGTC Symposium on Visualization.

[5] Victor Lempitsky. 2008. Surface Extraction from Binary Volumes with Higher-Order Smoothness. Technical Report in Microsoft Research.

[6] K.H. Sharkawi, M.U. Ujang and A. Abdul-Rahman. 2008. 3D Navigation System For Virtual Reality Based On 3d Game Engine. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. Part B4. Beijing 2008.

[7] Wahle, A.; Builtjes, J.H.; Oswald, H.; Fleck, E.; , "DICOM-integration in a heterogeneous environment," *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE* , vol.3, no., pp.1228-1229 vol.3, 31 Oct-3 Nov 1996.

[8] http://www.barre.nom.fr/medical/samples/

[9] DicomWorks, Freeware available at http://dicom.online.fr

Appendix

Code of Part 1 of the algorithm: compute and visualize the nodes of the graph to be constructed and its corresponding weight matrix.

```
%filename convention used in image series
```

```
prefix = 'Series 8\I0000';
fnum = 417:476;
ext = '_anon.dcm';

%first filename in series
fname = [prefix num2str(fnum(1)) ext];

%examine file header
info = dicominfo(fname)

%extract size info from metadata voxel_size = [info.PixelSpacing;
info.SliceThickness]'

%read slice images; populate XYZ matrix
hWaitBar = waitbar(0,'Reading DICOM files');
for i=length(fnum):-1:1
  fname = [prefix num2str(fnum(i)) ext];
  D(:,:,i) = uint16(dicomread(fname));
  waitbar((length(fnum)-i)/length(fnum))
end
delete(hWaitBar)
whos D

%% Visualization
%-------------------------------------------------------------------------
 %explore image data using Image Viewer GUI tool
i = 30;  %middle slice
im = squeeze(D(:,:,i));
max_level = double(max(D(:)));
imtool(im,[0 max_level])

%custom display - image data
fig1 = figure;
max_level = double(max(D(:)));
imshow(im,[0 max_level])
title('Coronal Slice #30')
set(fig1,'position',[601 58 392 314])
%imview close all
imtool close all

%add intensity legend
colorbar

%change colormap
colormap jet

%3D visualization
%explore 3D volumetric data using Slice-O-Matic GUI tool
addpath('D:\work\Demos\others\sliceomatic')
sliceomatic(double(D))
%ref: submission #780 @ www.mathworks.com/matlabcentral
hSlico1 = gcf;
daspect(1./voxel_size)
movegui('northwest')

%explore rotated 3D volume (new Slice-O-Matic viwer)
if ishandle(hSlico1), delete(hSlico1), end
sliceomatic(double(D))
daspect(1./voxel_size)
```

```
hSlico2 = gcf;
set(hSlico2,'position',[455 63 560 420])

%intensity distribution
%max_level = double(max(D(:)));
my_map = jet(max_level);
fig2 = figure;

%intensity distribution - top 2/3
subplot(3,1,1:2)
hist(double(im(:)),max_level)
axis([0 max_level 0 900])
title('Distribution')

%color scale - bottom 1/3
subplot(3,1,3)
imagesc(1:max_level)
colormap(my_map)
xlim([0 max_level])
set(gca,'ytick',[])
ylabel('Color Map')
xlabel('Intensity')
set(fig2,'position',[22 60 560 300],'render','zbuffer')
set(fig1,'position',[601 68 392 314])
figure(fig1)

%% Segmentation
im = imrotate(squeeze(D(30,:,:)),90);
figure(hSlico2)
thresh_tool(im)
D1 = D;

%apply thresholding rules
D(D<=40) = 0;
D(D>=100) = 0;
D(:,:,1:60) = 0;
update_sliceomatic(double(D),hSlico2)

%erode away thick layer (dissolve thin surrounding tissues)
blk = ones([3 7 7]);
D = imerode(D,blk);
update_sliceomatic(double(D),hSlico2)

%isolate brain mass
lev = graythresh(double(im)/max_level) * max_level;
bw = (D>=lev);
L = bwlabeln(bw);

%connected region properties - how many, how big?
stats = regionprops(L,'Area')
A = [stats.Area];
biggest = find(A==max(A))

%remove noise
D(L~=biggest) = 0;
update_sliceomatic(double(D),hSlico2)

%grow back main region
D = imdilate(D,blk);
```

```
update_sliceomatic(double(D),hSlico2)

%separate white vs. gray matter
im = imrotate(squeeze(D(30,:,:)),90);
figure(hSlico2)
lev2 = thresh_tool(im,'gray')

%partition brain mass
lev2 = 67;
L = zeros(size(D));      %0=outside brain (head/air)
L(D<lev2 & D>0) = 2;     %2=gray matter
L(D>=lev2) = 3;          %3=white matter

%new Slice-O-Matic viewer
sliceomatic(L)
hSlico3 = gcf;
daspect(1./voxel_size)
set(hSlico3,'position',[455 63 560 420])

%% Volumetric Measurements (voxel counting)
%-----------------------------------------------------------------------
%total volume of brain
brain_voxels = length(find(L(:)>1));
brain_volume = brain_voxels*prod(voxel_size)/1e6

%volume of gray matter
gray_voxels = length(find(L(:)==2));
gray_volume = gray_voxels*prod(voxel_size)/1e6

%volume of white matter
white_voxels = length(find(L(:)==3));
white_volume = white_voxels*prod(voxel_size)/1e6

%density calculations
gray_fraction = gray_volume/brain_volume
white_fraction = white_volume/brain_volume

return

%% Separate head from background for visualization (advanced
maneuver)
%-----------------------------------------------------------------------

%duplicate data
L1 = L;

%exterior of head (connected inside through ears)
%new Slice-O-Matic viewer (binary data)
BW = (D1<lev1);

%melt away layer (separate small interior volumes)
BW = imerode(BW,blk);
%update_sliceomatic(double(BW),hSlico3)

%how many connected regions?
L = bwlabeln(BW);
stats = regionprops(L,'Area')
```

```
%which one biggest?
A = [stats.Area];
biggest = find(A==max(A))
BW(L~=biggest) = 0;
%update_sliceomatic(double(BW),hSlico3)

%grow layer back
BW = imdilate(BW,blk);

%label head voxels
L = L1;
L(BW<1 & L1<2) = 1;  %1=head (0=air, 2=gray, 3=white)

%update final display
update_sliceomatic(L,hSlico3)
```

Apply_dijkstra.m