**Segmentation**

Our segmentation algorithm for the tumor dataset can be configured to use either K-means or GMM algorithms as its base classifier. The results are then post-processed using Graph Cut, random walker, or ICM. The hippocampus dataset was segmented using a Multi-Altas Segmentation method where we registered each of the training images to each test image and then projected the labeling for the hippocampus from the training image onto the test image. The results from the multi-atlas method were post processed using the same Graph Cut, random walker and ICM methods used for the tumor dataset. We found that the GMM method with minor Graph Cut produced highly accurate results.

**Multi-Atlas Resolution**

For the hippocampus dataset we found that the Multi-Atlas Resolution method worked very well. We experimented with different methods for defining the cutoff for the labels and eventually settled on a probabilistic model. For each training image, we assigned a probability distribution for the mask and then multiplied each of these to get the joint probability. To avoid underflow, we added logs rather than use straight multiplication. We found that this method, while successful, provided marginal benefit over averaging the resolved masks, normalizing the intensity and then using a cutoff of .5 for the hippocampus.

**GMM and K-means**

For a base classifier, we implemented GMM and K-means algorithms. We used the labeled seed points from the tumor dataset to set initial values for the clusters and then assigned the remaining points to the appropriate cluster (tumor, edema, brain and empty) based on the. distance measure for that point to each of the four clusters. We found that GMM was far superior to K-means, consistently achieving high scores on each of the four measures. As can be seen in Table 1. Across the board, we found GMM to be preferable to K-means for an initial segmentation.

Table 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Dice (Label 1) | Dice (Label 2) | Avg. Dist. (Label 1) | Avg. Dist. (Label 2) | Hausdorff Dist. (Label 1) | Hausdorff Dist. (Label 2) |
| K-means | 0.6024 | 0.2233 | 64.0473 | 53.1909 | 121.8504 | 109.5293 |
| GMM | 0.8179 | 0.6161 | 76.7084 | 69.0498 | 144.1643 | 134.4436 |

**Graph Cut**

The Graph Cut algorithm is implemented in graph\_cut.m in the attached files. Using this algorithm we found that we were able to consistently achieve good results. The Graph Cut was very successful at reducing the variance in labeling within a local region of the image. We can see in figure 1 the effect of the Graph Cut algorithm on an otherwise noisy segmentation.

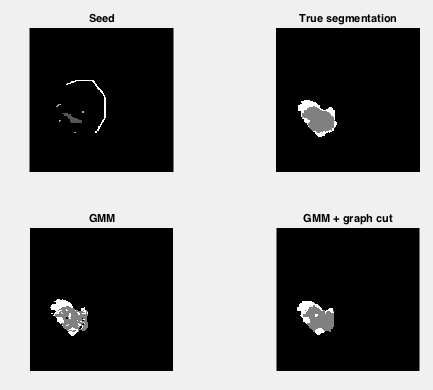


Figure 1

While the Graph Cut algorithm was successful at removing variance from the labeling, we did not see a significant increase in accuracy. We can see in table 2 that the accuracy measures for GMM were in some cases better than for GMM + Graph Cut.

Table 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Dice (Label 1) | Dice (Label 2) | Avg. Dist. (Label 1) | Avg. Dist. (Label 2) | Hausdorff Dist. (Label 1) | Hausdorff Dist. (Label 2) |
| GMM | 0.8179 | 0.6161 | 76.7084 | 69.0498 | 144.1643 | 134.4436 |
| GMM + Graph Cut | 0.7954 | 0.6713 | 80.8471 | 72.9769 | 149.0450 | 140.3605 |

**Random Walker Algorithm**

*Implementation*:

This method was implemented using the Random Walker method for segmentation described by Grady1,2. The various sections of this algorithm are implemented in RandomWalkerSeg.m in the attached files. This then displays the segmentation, the seeds, and the original image for easy comparison.

There were two primary parameters which needed to be tuned to optimize this algorithm. The first one was the selection of the initial points. Because the initial data came with a set of seeds and their locations, this offered a number of possible locations for initial seeds, and it was readily apparent from testing that the segmentation was highly dependent on the seeds chosen.

Because the algorithm ran quickly on its own (t=0.22s on a 4.5GHz Intel i5-4690k processor for tumor image 001), it was trivial to calculate the segmentation for every possible seed combination.

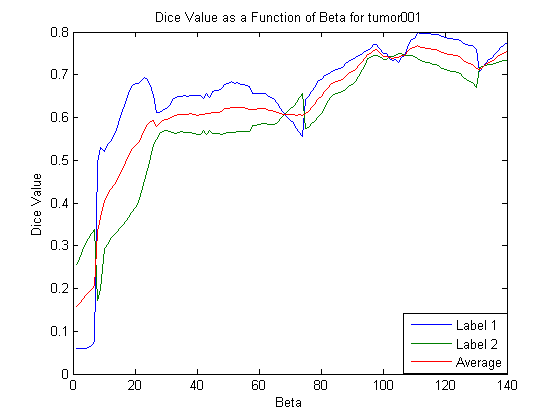
Two methods were considered, an average Method and a selection Method. To demonstrate the efficiency of this despite the large number of calculations, the Average Method took 9.04s to run and the selection method took 11.34s to run on the same processor as above for tumor image 001. The average method averaged the probabilities for all of the images in the hopes that the stronger segmentations would predominate over the weaker ones, and that they would contribute to different areas to provide a stronger segmentation. The selection method instead determined the average Dice value for each label as each segmentation was calculated, and compared it to the average Dice value for the previous segmentation. The segmentation with higher average Dice was saved, such that the strongest segmentation was selected.

The selection method proved to be more effective, even though it took slightly longer because of the calculation for Dice as it was run. The average method was dominated by the more abundant weak segmentations, and so the result was inferior to the average method. The comparison for the segmentation metrics for tumor image 001 for the two methods is shown in table 1. While there is not a significant difference between the average distance and Hausdorff distance metrics between the two, there is a significant improvement in the Dice score. A hybrid method adding only high-dice values above a certain threshold was also inferior to the selection method. It appears that smearing during averaging leads to an overall poor segmentation.

Table 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Beta=99 | Dice (Label 1) | Dice (Label 2) | Avg. Dist. (Label 1) | Avg. Dist. (Label 2) | Hausdorff Dist. (Label 1) | Hausdorff Dist. (Label 2) |
| Average Method | 0.4639 | 0.5968 | 47.68 | 49.30 | 55.37 | 56.45 |
| Selection Method | 0.7617 | 0.7420 | 52.77 | 44.66 | 59.17 | 51.93 |

The second free parameter that had to be adjusted was the value for Beta. This was made somewhat difficult by the dramatic variability in response to this value. The Dice score changed dramatically depending on the value for Beta and the image used. Figure 1 shows the change in the Dice Score as Beta changes for tumor image 001.



The beta provided a maximum for the average Dice score at Beta=111; however the maximums for the individual segmentations for the tumor and edema occur at different points. At values over 140 the dice scores begin to diverge again, leading to poor segmentations. The average dice score is chosen because it represents a compromise that gives good segmentations for both components. Based on this, a value of Beta=111 was chose; however, this was no guarantee that it would be optimal for other images, and this turned out to be the case.

The papers suggested a value of 90 for Beta, so to compromise we added another layer to the selection algorithm that tested for several values of Beta ranging from 90 to 110. This proved to be more robust when calculating the segmentation, while keeping time reasonable (t= 47.31s for tumor image 001). Figure 2 shows segmentation for tumor image 003 with Beta=111, while Figure 3 shows the segmentation with multiple Betas. Even though Beta=111 works well for a number of the images, the multiple Betas method gives better segmentations in situations where the image is a poor candidate for segmentation. Figure 4 shows the segmentation for a well behaved image (tumor image 001).

Figure 2

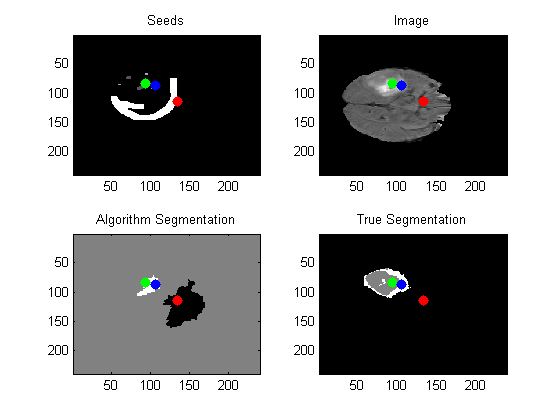


Figure 3

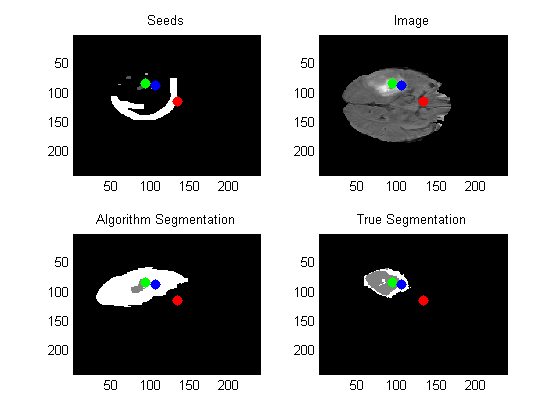
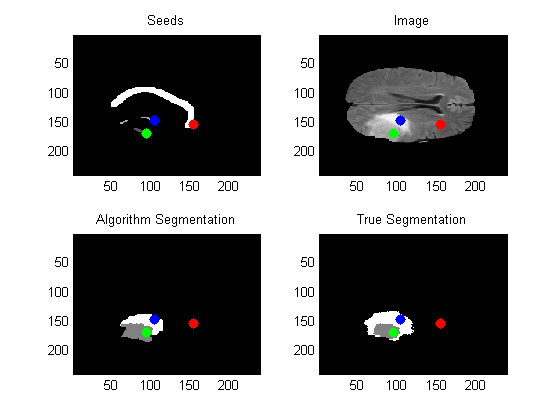


Figure 4



Overall the Random Walker algorithm provides an elegant solution to a complicated problem. While very sensitive to the tuning parameters, the algorithm’s efficiency, obtained by using linear equations to determine segmentation, allows for brute-force approaches to be used without taking an unreasonable amount of time. The advantage over other algorithms is that is less likely to get stuck on corners or edges; however, it does require good choice or knowledge a priori of starting locations for the random walker, or a brute-force approach is needed. Fortunately it is versatile enough that a single algorithm can be used for both cases.