EM Segmentation: Automatic Tissue Class Intensity Initialization Using K-means

P. Srinivasan, M. E. Shenton and S. Bouix

July 2011

ABSTRACT

Brain tissue segmentation is important in many medical image applications. We augmented the Expectation-Maximization segmentation algorithm in Slicer3 (www.spl.harvard.edu) . Currently, in the EM Segmenter module in Slicer3 user input is necessary to set tissue-class (Gray Matter, White Matter etc.) intensity values. Our contribution to the current pipeline is to automatically compute such values using k-means clustering. Our method can be applied to scans of varying intensity profiles and thus we obviate the need for a normalization step. We applied this pipeline on multiple datasets and our method was able to accurately classify tissue-classes. The implementation was done as a standalone utility in the Python programming language (www.python.org) and work is underway to incorporate it in the EM processing pipeline.

INTRODUCTION

Brain tissue segmentation is an important step in many medical image applications for instance, in determining tissue volume quantification in various pathology such as in Schizophrenia [1], in lesion location and quantification etc. Many segmentation methods based on region growing, clustering, watershed transformation and level sets have been developed. The segmentation algorithm in Slicer3 developed at the Surgical Planning Laboratory (www.spl.harvard.edu) is based on the Expectation-Maximization (EM) Algorithm [2].

The pipeline of the EM segmentation module initially implemented in Slicer3 (www.slicer.org) [2] is summarized below:

1. Creation of a hierarchical tree: This defines the way segmentation should proceed. For example, the entire image can be first segmented into Background (BG), Gray Matter (GM), White Matter (WM) and Cerebro-spinal Fluid (CSF) . Then GM can be further divided in to Left GM and Right GM and so on. The final structures to be segmented are the leaves of the tree.
2. Choosing Atlas Probability Priors: In this step the user has to input a prior atlas-probability map for each leaf node i.e. the final structures to be segmented in the hierarchical tree created.
3. Multi-channel segmentation: This step allows the user to enter multiple contrast input images. For instance CSF appears brighter in a T2-weighted image and thus this information can be used for segmentation.
4. Intensity Normalization: This step is useful when 3D scans of different intensity profiles have to be segmented but by only using common pre-determined template parameters like the global priors, atlas weights, tissue class intensity distribution etc. for all the scans. This is to avoid choosing parameters, say, the tissue-class intensity distribution manually for each 3D scan. For a chosen atlas image, the mean value of pixels with the background excluded is chosen as the normalization value and all target images are normalized to this value.
5. Selecting tissue-class intensity distributions: Users have two options, either selecting the tissue-class intensity distribution manually by clicking on a few representative voxels for each structure to be segmented, or using the template parameters computed from the intensity normalization step.
6. Hierarchical parameters: This step allows the user to select hierarchical parameters such as the global prior, atlas weight and the weights to be given to each image in the multi-channel segmentation.
   a) The global prior ranges from 0 to 1 and gives an idea of the size of the structure to be segmented.
   b) The atlas weight also ranging from 0 to 1 determines if a particular spatial prior atlas needs to be used when segmenting a structure. For instance, GM, WM and CSF are well delineated just by using intensity parameters and so the atlas weight can be set to 0. On the other hand, atlas weight is typically set to 1 for segmenting Left GM and Right GM.
   c) The weight for the multi-channel images determines how much a particular image should be used in

segmenting a structure. For instance, T2-weighted images can be given a weight of 1 while segmenting CSF.

7. Registration: The step is done to align the atlas priors to the images provided in the multi-channel segmentation step.

Improvements were done to the above processing pipeline as described in [3]. Only those that are part of the processing pipeline are mentioned here excluding others like Bias Field correction. Briefly, the following were incorporated:

1. In Step 5 above, for selecting the tissue-class distributions, a label map illustrating the relation between color and tissue class is setup by the user; the mean and covariance values are then automatically computed. This label map can be saved for further use. Also, a tissue-class intensity distribution visualization tool was developed that allows the user to see the similarity or dissimilarity of the distributions of the classes to be segmented.
2. A normalization tool allows the selection of a normalization value based on the image histogram that appears in the Slicer3 GUI. Background excluded, the mean is calculated based on the threshold the user chooses and the maximum intensity of the scan.
3. Global Prior Weights Calculation: A tool is provided that helps the user calculate global prior weights or an estimate of the size of the tissue class to be segmented. Visual feedback is provided for each tissue class as the user chooses global priors.

The primary motivation for the work described in this article is that there is currently no existing method in the EM Segmenter module in Slicer3 that is fully automatic in setting the intensity profiles for the different tissue classes. The method described in [3] of setting the label map intensities provides visual feedback still requires manual user input and is fairly involved. Our contribution in the EM Segmentation pipeline is to automatically compute tissue intensity values using k-means clustering. A key advantage of our approach is that in addition to obviating manual user input, it can be applied to scans with varying intensity profiles since an explicit normalization step is unnecessary.

Our intensity initialization algorithm is currently implemented as a standalone utility in the Python programming language (www.python.org) and work is underway to incorporate this in the EM processing pipeline inside Slicer3.

BACKGROUND THEORY

Expectation-Maximization

In this section, we briefly review the EM algorithm and refer the interested reader to [2] , [3] and [4] for more details on the theory and implementation.

The EM algorithm iterates alternatively between an Expectation (E) step and a Maximization (M) step until convergence is reached:

The E step produces a soft segmentation map of the hidden data by using the current estimates of the parameters. In our case, the parameters are the mean, covariance and initial prior probability that a voxel belongs to a tissue class (provided by the probabilistic atlas) and the hidden data are the labels of the final segmentation.

The M step uses the soft segmentation map to refine the parameters.

In Slicer3, a novel contribution includes incorporating the EM algorithm via a hierarchical tree [2] that describes the relationship between the anatomical structures. For example, the image is divided into Background (BG) and Intra-Cranial Content (ICC) which is further divided into Gray Matter (GM) , White Matter (WM) and Cerebro-Spinal Fluid (CSF). By changing the tree any desired segmentation of the brain can be achieved. For example, the GM can be further divided into Left GM and Right GM etc. as long as an accurate atlas prior exists for each

structure to be segmented. Information such as tissue class intensity distributions and label probabilities for each structure to be segmented are stored at the leaves and this is propagated upwards towards the respective parent nodes of the leaves. The segmentation process proceeds downwards from the parent nodes to the leaves. This framework is very flexible and at each sub-segmentation level, the problem is simplified.
The main limitation of the algorithm described in [2] is that any error made at the previous segmentation step cannot be corrected and thus propagates to the rest of the tree.

K-means algorithm

K-means is a popular clustering algorithm in which the goal is to classify the data points in a sample into a given number of clusters fixed *a priori*. Briefly, this algorithm accepts centroids one for each cluster to be classified. The centroids for k-means can be chosen Then the distance between the each data point and all the centroids are calculated and the data point is assigned to the centroid to which it is closest to. When all the points have been assigned, the cluster centroids are then recalculated and this process is repeated. The goal of the algorithm is to minimize the sum of the squares of the distances of each data point to its nearest centroid [5] .

METHODS

This section describes the steps followed in our approach. There are two steps viz. pre-processing and computing final intensity parameters using k-means clustering.

Step 1 : Pre-Processing

1. Atlas probability priors are generated using the registration algorithm in [6].
2. Intensity inhomogeneity can be problematic in determining accurate tissue class intensities and also in registration of atlas priors to the T1 structural image. Thus bias field correction of the T1 structural image is done using FSL FAST [7].
3. The input atlas probability priors have to be pre-aligned with the T1 structural image before inputting them in the Python pipeline. This is because the binarized atlas priors are multiplied with the structural T1 image for k-means initialization (explained in detail below). The prior atlases are aligned using rigid registration.
4. A template Slicer3 EM Segmenter scene with the required parameters such as global priors, atlas weights and tree hierarchy is created using the EM Segmenter pipeline. Intensity values need not be determined at this point. This scene is created only once and used as a template for further segmentation of all other cases, assuming the same tree structure.

The Python command line implementation can be used once the above pre-processing steps are completed. The algorithm used to initialize tissue class intensities is described in the following section.

Step 2: Computation of Final Intensities Using K-means in Python

1. From within the Slicer Python console, the module *replace_fileName.py* for customizing the EM Segmenter scene created in Step 4 of the pre-processing described above is called. This module takes in paths to the T1 image and atlas priors for a new case and replaces the corresponding existing paths in the template scene.
2. Next the module *kmeans_start.py* for performing k-means is called. In addition to the two input paths to the T1 image and atlas priors, this takes in a text file and the template EM Segmenter scene. This module automatically performs the following tasks:
    a) Determines the number of partitions in the k-means clustering. This is automatically calculated based on the number of rows in the user-defined input text file *class_leaf_text* (explained below). The number of rows can be set intuitively based on the data. For example, the brain clearly has four different intensity profiles i.e. Background (BG), Cerebro-spinal Fluid (CSF), Gray Matter (GM) and White Matter (WM). Although GM, say, is further split into left, right, sub etc. in the final EM Segmentation in our case, each of the substructure is assumed to have

the same intensity. Thus individual atlas priors of different lobes of GM are added to give one single GM prior atlas [Figure 1] (see also *class1* in *class_leaf_text* below) making up one row in *class_leaf_text*. This can be easily extended to any data and the number of desired partitions can be easily increased as the user sees fit simply by adding more rows in the text file and by giving a corresponding prior atlas. The entries corresponding to each row must correspond to leaf nodes in the template scene. An example of the text file used for our brain parcellation is shown below:

class0: BG
class1: ltgm1, ltgm2, ltgm3, ltgm4, rtgm1, rtgm2, rtgm3, rtgm4, subgm
class2: ltwm1, ltwm2, ltwm3, ltwm4, rtwm1, rtwm2, rtwm3, rtwm4, subwm
class3: CSF

where, lt=left, rt=right, subgm=subcortical gray matter, subwm=subcortical white matter.

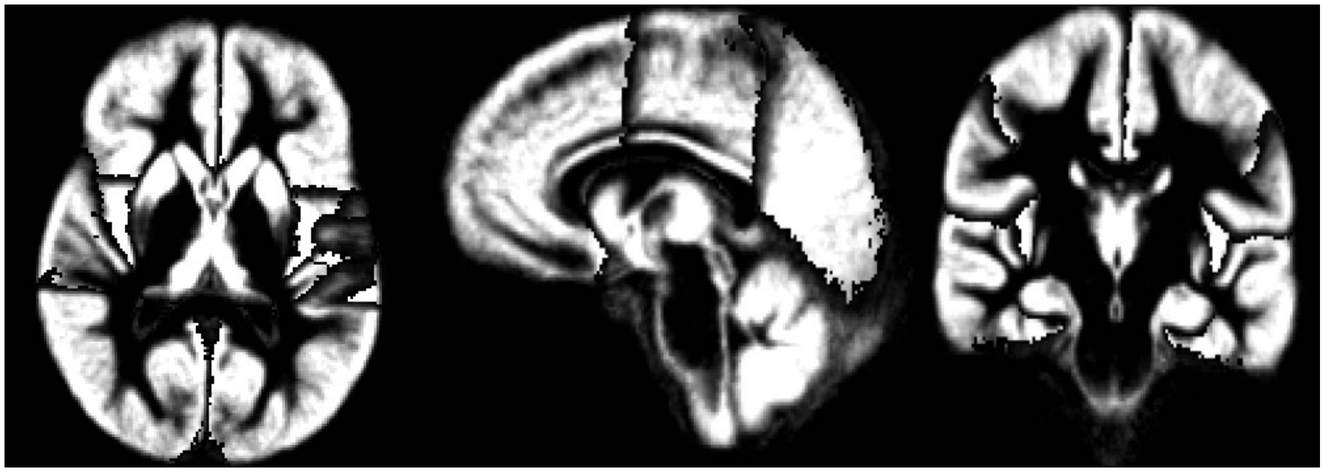
Figure 1: The figure above shows the summed GM atlas prior consisting of atlas priors of the individual lobes for one 2D slice.

b) Initializes the centroid values for k-means. In this step, the atlas is first binarized by thresholding the summed atlas probability priors (scaled between 0 - 100) at a value of 70. This ensures that only those pixels of a particular structure whose probability is greater than 70 is considered for centroid initialization for that structure. The threshold value was chosen after experimenting with other values and was found to be best given the atlas priors. Figure 2 shows the thresholded binarized atlas priors. Finally, the thresholded atlas priors are multiplied with the structural T1 image and the mean of the resulting 3D structure is calculated. This mean is provided as the initial centroid value for k-means clustering. Figure 3 shows an example for the GM structure.

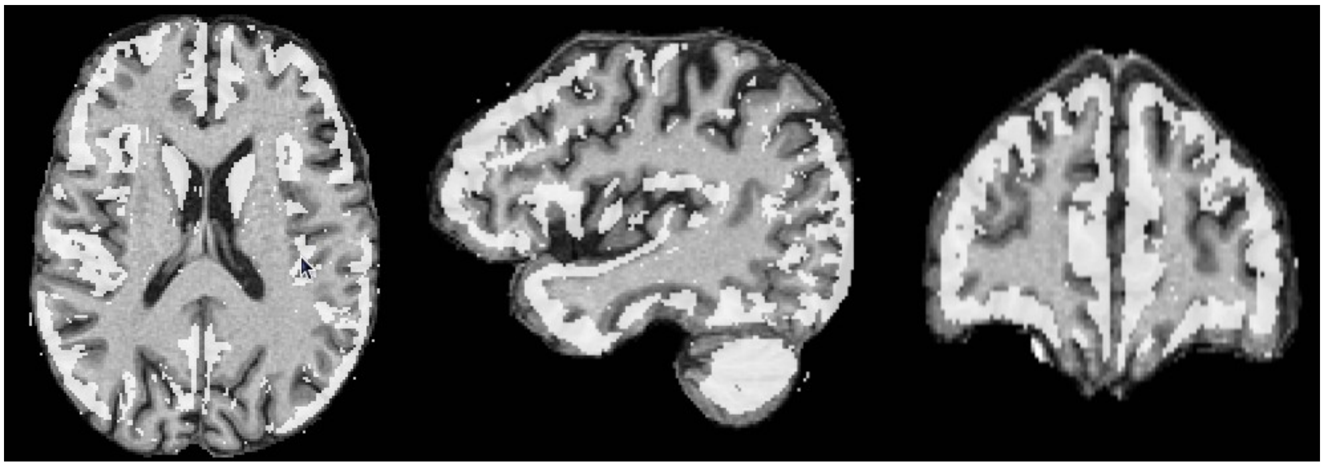
Figure 2: Thresholded atlas prior for the GM overlayed on the T1 structural image for one 2D slice.

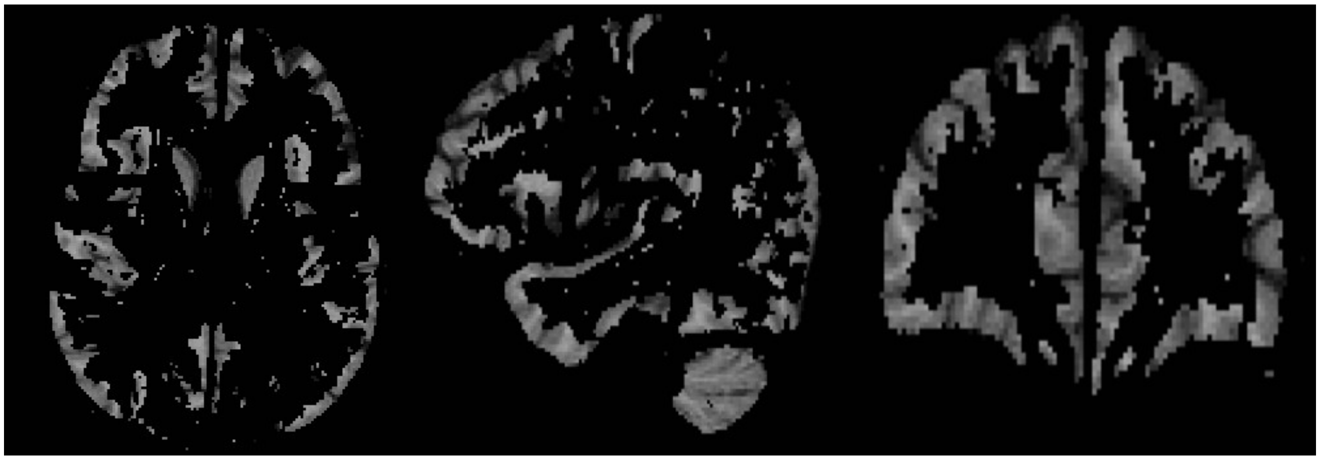
Figure 3: Resulting structure after multiplying thresholded atlas priors with T1 image for the GM of one slice. The mean for the whole 3D GM similar to the above is calculated and provided as the initialization value for GM in k-means clustering.

c) Calls the module cal*c_means_covariance.py*. Briefly, this calculates the k-means clusters using *scipy.cluster.vq.kmeans2*. The number of clusters is determined by the number of rows provided in *class_leaf_text* as described above.

d) Calls the module *replace_mrml.py*. Briefly, this overwrites the existing log mean and log covariance values in the template scene for each of the leaf names e.g. ltgm1, ltgm2 etc. provided in *class_leaf_text* as above.

RESULTS

1. The results of segmentation for the brain structures in the above mentioned class_leaf_text example for a volume scan from a 3 Tesla scanner is shown in Figure 4.

2. Intensity normalization is not required across scans of varying intensity profile. This is because for each scan regardless of the intensity profile, k-means computes a suitable starting cluster initialization based on the multiplication of the atlas priors and the respective T1 structural scan. The segmentation of scans for the same subject but different intensity profiles is shown in Figure 5.

3. A comparison of segmentation from k-means initialized intensity values with an algorithm that uses

aligned atlases and corresponding T1 image sampling to obtain intensity values is shown in Figure 6. The differences in white matter estimation can be clearly seen in row 1 (k-means) and row 2 (T1 sampling) below. Also CSF is underestimated throughout the volume with just T1 sampling.

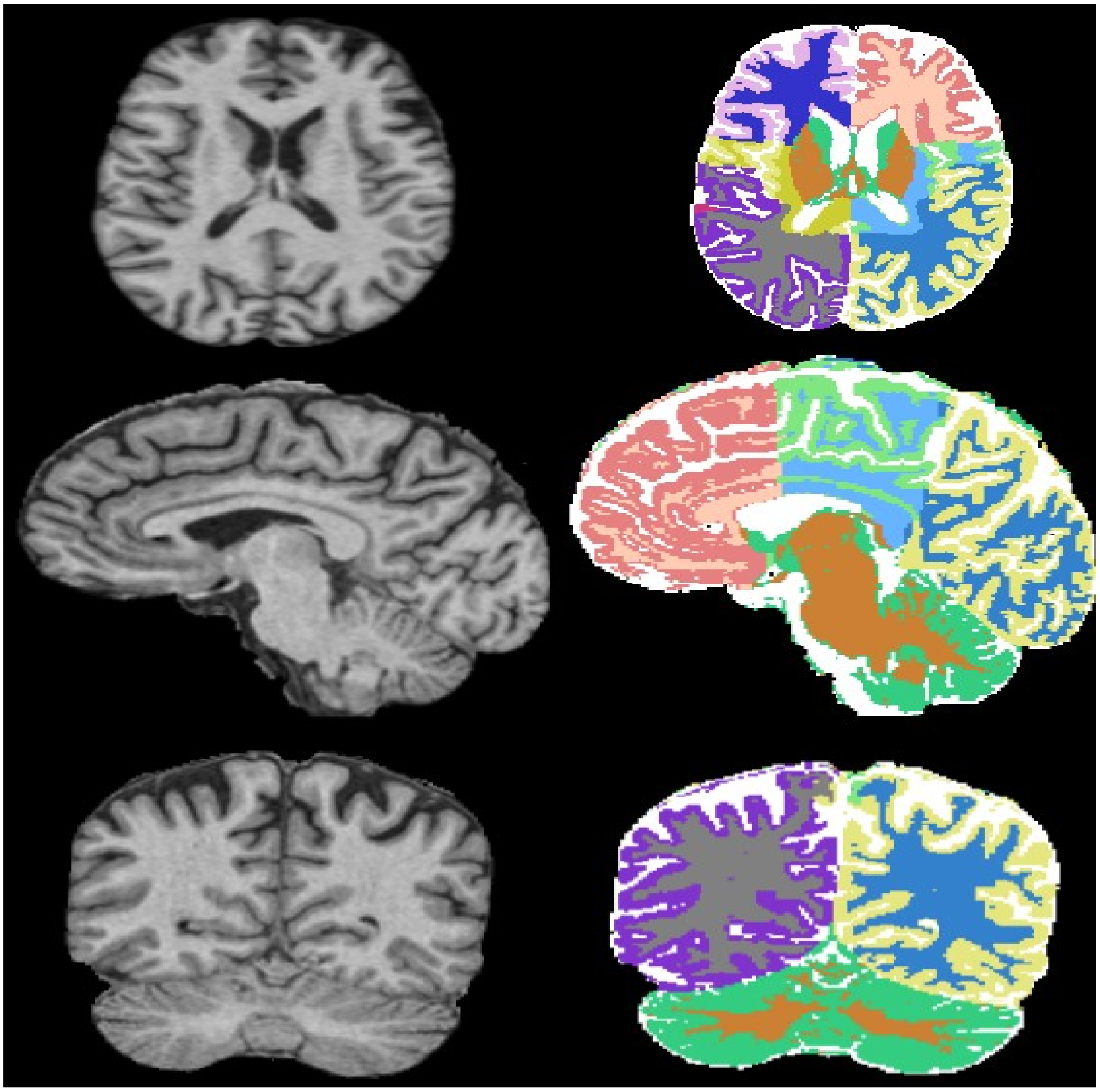

Figure 4: 3 Tesla T1 structural scans and the respective segmented scans.

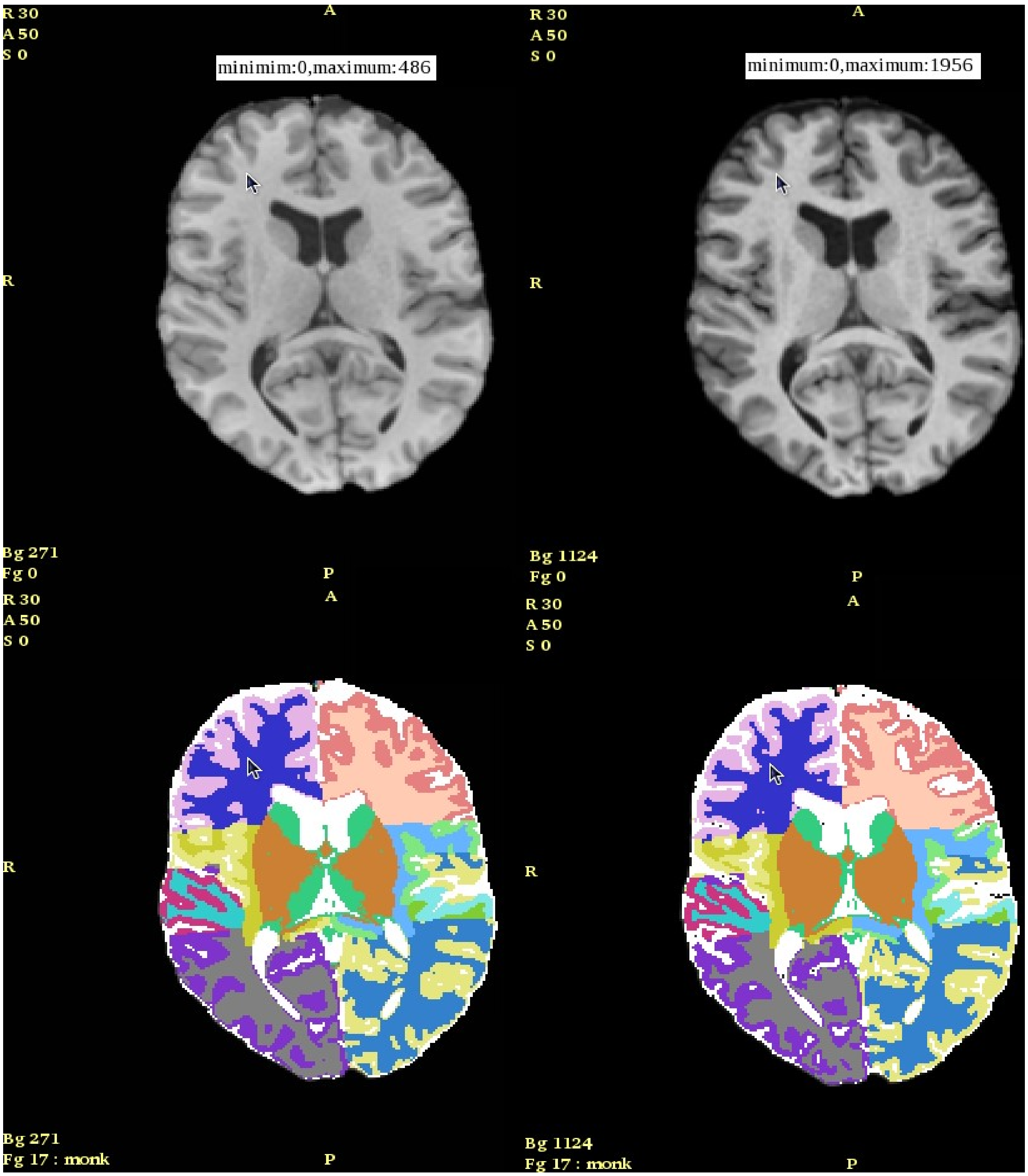

Figure 5: The figure above shows scans for the same subject but with different intensity profiles. The minimum and maximum values for the whole 3D scan respectively is shown in the figure. The value at a single voxel location indicated by the mouse pointer.

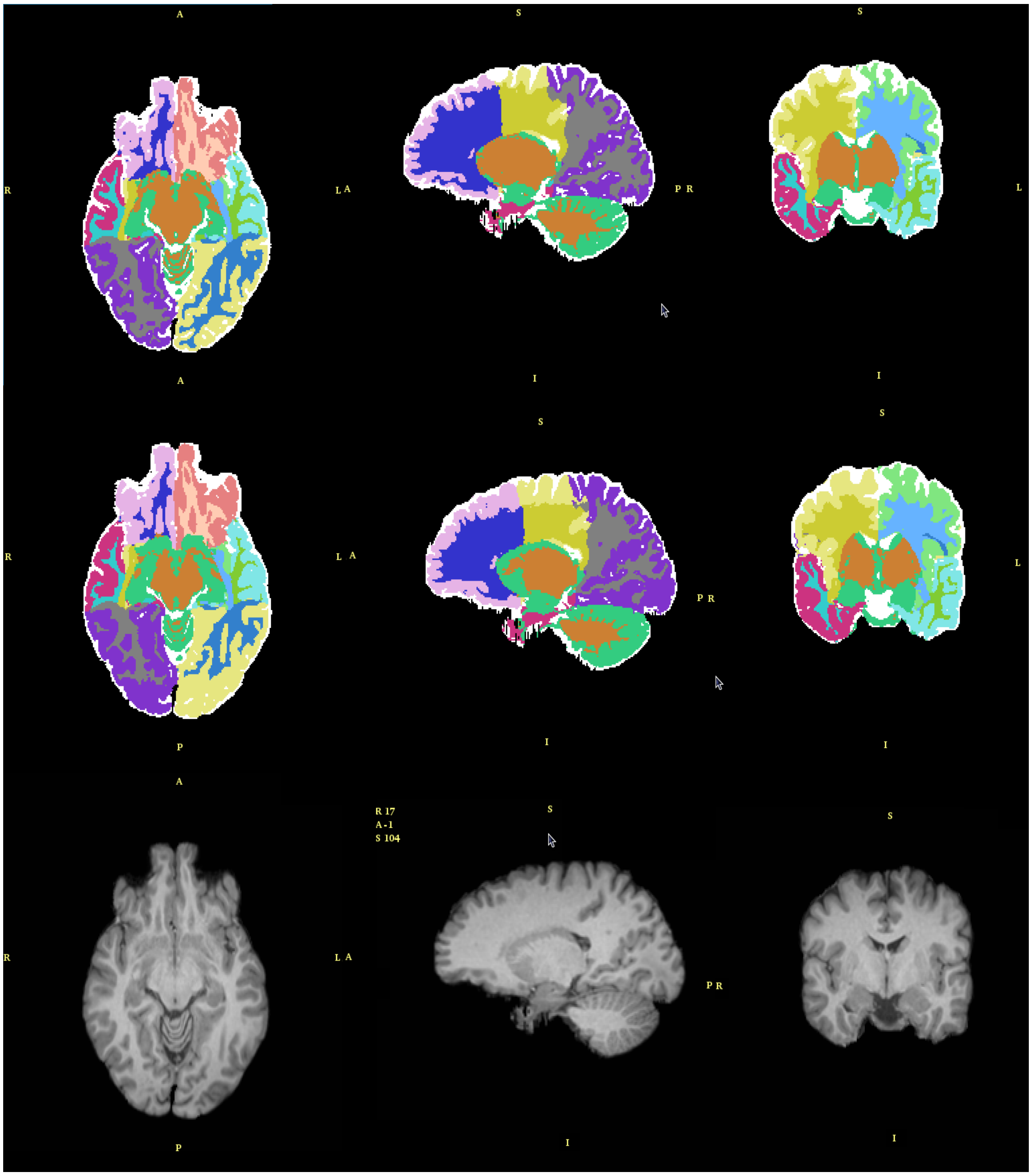

Figure 6: Row 1 -Segmentation obtained using k-means initialization. Row 2 – Segmentation obtained using T1 sampling. Row 3 – Original T1 image.

CONCLUSION

As observed in the experimental results, k-means clustering method is well suited in our effort to automate tissue class intensities. In addition to obviating user input, the other advantages are that an explicit normalization step for scans of different intensity profile is not necessary and the results obtained are reproducible because the starting initializations for the clusters remain the same. Thus different users will get the same results for a given subject. The current disadvantage of this method is that the atlas priors have to be aligned prior to giving them as inputs to the k-means pipeline. However, work is in progress to incorporate our approach in the EM pipeline such that the atlas priors will need not be aligned beforehand.

REFERENCES

1. Neocortical Gray Matter Volume in First-Episode Schizophrenia and First-Episode Affective Psychosis: A Cross-Sectional and Longitudinal MRI Study. M. Nakamura, D.F. Salisbury, Y. Hirayasu, S. Bouix, K. Pohl, T. Yoshida, M. Koo, M. Koo, R. McCarley. Biological Psychiatry. 2007
2. A Hierarchical Algorithm for MR Brain Image Parcellation. K. Pohl, S. Bouix, M. Nakamura, T. Rohlfing, R. McCarley, R. Kikinis, W. Grimson, M. E._ Shenton, W. Wells. IEEE Transactions on Medical Imaging Volume 26, Number 9, Pages 1201-1212, 2007
3. New Expectation Maximization segmentation pipeline in Slicer 3. Rannou N., Jaume S., Pieper S., Kikinis R. Insight Journal. 2009
4. Tutorial on Expectation-Maximization: Application to Segmentation of Brain MRI. Maria Murgasova. May 6, 2007
5. Pattern Recognition and Machine Learning. Christopher Bishop. 2006
6. Asymmetric Image-Template Registration. Mert R. Sabuncu , B.T. Thomas Yeo , Koen Van Leemput, Tom Vercauteren, and Polina Golland . Med Image Comput Comput Assist Interv. 2009;12(Pt 1):565-73.
7. Segmentation of brain MR images through a hidden Markov random field model and the expectation maximization algorithm. Zhang, M. Brady, and S. Smith. *IEEE Trans. on Medical Imaging*, 20(1):45-57, 2001.