MRI SEGMENTATION USING KMEANS AND CANNY EDGE DETECTOR ALGORITHM

Anu Sharma[1], Ashish Oberoi[2] and Rajeev Kumar[1]

[1] Computer Science Department College of Engineering
Teerthanker Mahaveer University Moradabad.
[2]Lecturer M.M.University, Mullana Ambala
E-Mail : rajeev2009mca@gmail.com

Abstract: In this paper, two algorithms for MRI segmentation are studied. K-means and canny edge detector. The objective of this paper is to perform a segmentation process on MR images of the human brain, using K-means Algorithm and canny Edge detection algorithm. K-means Clustering algorithm gives us the segmented image of an MRI having the same intensity regions. K-means Clustering segments all the three matters of the brain i.e. Grey matter, White matter and Dark matter. Also the edge detection algorithm is implemented that gives us the boundaries of the various regions of the MRI depending on scale and threshold values used for the segmentation. Implementation of each algorithm is then discussed. Finally, the experimental results of each algorithm are presented and discussed.
[Anu Sharma, Ashish Oberoi and Rajeev kumar.MRI segmentation using k-means and canny edge detector algorithm. New York Science Journal 2011;4(6):53-60]. (ISSN: 1554-0200). http://www.sciencepub.net/newyork.

Keywords: Clustering Analysis, Medical Imaging, Thresholding Techniques.

1. Introduction:

Segmentation refers to the process of partitioning a digital image into multiple segments (sets of pixels, also known as super pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze[1]. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. Medical imaging is performed in various modalities, such as MRI, CT, ultrasound, positron emission tomography (PET), etc. In the present review, we are focusing primarily on the segmentation of MR and CT images only. MR is generally more sensitive in detecting brain abnormalities during the early stages of disease, and is excellent in early detection of cases of brain tumors, or infections. MR is particularly useful in detecting white matter disease. In contrast, CT scan fails to detect white matter abnormalities. In this, we look at two algorithms namely K Means clustering, And canny edge detector and compare them for image segmentation.

The main advantages of MR imaging system are:.

- It has an excellent capability for soft tissue imaging
- It has very high resolution of the order of 1mm cubic voxels
- It has high signal to noise ratio
- Multi channel images with variable contrast can be achieved by using different pulse sequences; this can be further utilized for segmenting and classifying different structures[2].

2. MRI SEGMENTATION ALGORITHM:

Images can be segmented into regions by the following algorithms:

2.1 K-means Clustering Algorithm: K-means (Macqueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids shoud be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more[3].

K-Means algorithm is an unsupervised clustering algorithm that classifies the input data points into multiple classes based on their inherent distance from each other. The algorithm assumes that the data

features form a vector space and tries to find natural clustering in them. The points are clustered around centroid

$\mu_i \forall_i = 1 \ldots k$ which are obtained by minimizing the objective

$V=\sum_{i=1}^{k} \sum_{x_j \in s_i} (x_j - \mu_i)^2$

where there are k clusters Si, i = 1; 2; : : : ; k and μ_i is the centroid or mean point of all the points Various steps in the algorithm are as follows:

Compute the intensity distribution(also called the histogram) of the intensities.

1. Initialize the centroids with k random intensities.
2. Repeat the following steps until the cluster labels of the image does not change anymore.
3. Cluster the points based on distance of their intensities from the centroid intensities.

$c^{(i)} := \arg\min_j \| x^i - \mu_j \|^2$

4. Compute the new centroid for each of the clusters.

$\mu_i = \sum_{i=1}^{m} 1\{c_{(i)} = j\} x^{(i)} / \sum_{i=1}^{m} 1\{c_{(i)} = j\}$

where k is a parameter of the algorithm (the number of clusters to be found), i iterates over all the intensities, j iterates over all the centroids and μi are the centroid intensities[4].

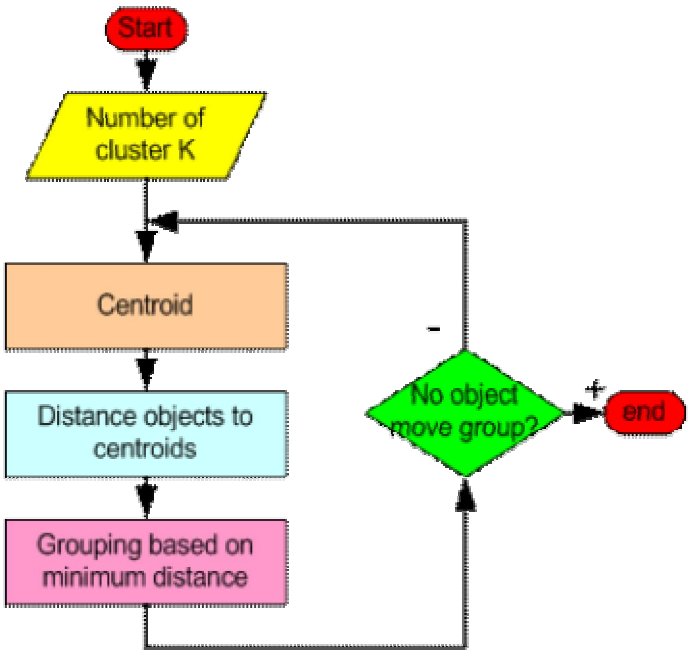

Figure 1.

The K-Means clustering algorithm was developed by J. MacQueen (1967) and then by J. A. Hartigan and M. A. Wong around 1975. *Simply speaking, K-Means clustering is an algorithm to classify or to group your objects based on attributes/features, into K number of groups. K is a positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. Thus, the purpose of K-means clustering is to classify the data*[5].

2.2 Canny's Edge Detection Algorithm: The Canny edge detection algorithm is known as the optimal edge detector. Canny's intentions were to enhance the many edge detectors already out at the time he started his work. He was very successful in achieving his goal and his ideas and methods can be found in his paper, "*A Computational*

Approach to Edge Detection". In his paper, he followed a list of criteria to improve current methods of edge detection. The first and most obvious is low error rate. It is important that edges occurring in images should not be missed and that there be NO responses to non-edges. The second criterion is that the edge points be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum. A third criterion is to have only one response to a single edge. Based on these criteria, the canny edge detector first smoothes the image to eliminate and noise. It then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (no maximum suppression). The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds and if the magnitude is below the first threshold, it is set to zero (made a no edge). If the magnitude is above the high threshold, it is made an edge. And if the magnitude is between the 2 thresholds, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above T2.

Step 1: In order to implement the canny edge detector algorithm, a series of steps must be followed. The first step is to filter out any noise in the original image before trying to locate and detect any edges. And because the Gaussian filter can be computed using a simple mask, it is used exclusively in the Canny algorithm. Once a suitable mask has been calculated, the Gaussian smoothing can be performed using standard convolution methods. A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. **The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise**. The localization error in the detected edges also increases slightly as the Gaussian width is increased. The Gaussian mask used in my implementation is shown below.

1/15

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Fig 2: Discrete approximation to Gaussian function with σ =1.4

Step 2: After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on an image. Then, the approximate absolute gradient magnitude (edge strength) at each point can be found. The Sobel operator uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows). They are shown below:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 3.

The magnitude, or edge strength, of the gradient is then approximated using the formula:

$$|G| = |Gx| + |Gy|$$

Step 3: The direction of the edge is computed using the gradient in the x and y directions. However, an error will be generated when sumX is equal to zero. So in the code there has to be a restriction set whenever this takes place. Whenever the gradient in the x direction is equal to zero, the edge direction has to be equal to 90 degrees or 0 degrees, depending on what the value of the gradient in the y-direction is equal to. If GY has a value of zero, the

edge direction will equal 0 degrees. Otherwise the edge direction will equal 90 degrees. The formula for finding the edge direction is just:

Theta = invtan (Gy / Gx)

Step 4: Once the edge direction is known, the next step is to relate the edge direction to a direction that can be traced in an image. So if the pixels of a 5x5 image are aligned as follows:

x x x x x
x x x x x
x x a x x
x x x x x
x x x x x

Figure 4.

Then, it can be seen by looking at pixel "**a**", there are only four possible directions when describing the surrounding pixels - **0 degrees** (in the horizontal direction), **45 degrees** (along the positive diagonal), **90 degrees** (in the vertical direction), or **135 degrees** (along the negative diagonal). So now the edge orientation has to be resolved into one of these four directions depending on which direction it is closest to (e.g. if the orientation angle is found to be 3 degrees, make it zero degrees). Think of this as taking a semicircle and dividing it into 5 regions.

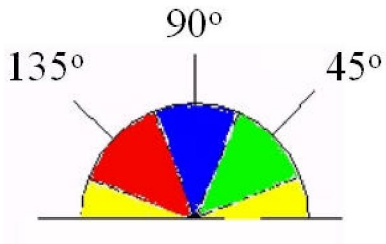

Figure 5.

Therefore, any edge direction falling within the **yellow range** (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the **green range** (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the **blue range** (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the **red range** (112.5 to 157.5 degrees) is set to 135 degrees.

Step 5: After the edge directions are known, nonmaximum suppression now has to be applied. Nonmaximum suppression is used to trace along the edge in the edge direction and suppress any pixel value (sets it equal to 0) that is not considered to be an edge. This will give a thin line in the output image.

Step 6: Finally, hysteresis is used as a means of eliminating streaking. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. To avoid this, hysteresis uses 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to this edge pixel and that have a value greater than T2 are also selected as edge pixels. If you think of following an edge, you need a gradient of T2 to start but you don't stop till you hit a gradient below T1[5,6,7]

Canny specified three issues that an edge detector must address. They are:-

1. **Error rate:-**The edge detector should respond only to edges, and should find all of them; no edges should be missed.
2. **Localization:-** The distance between the edge pixels as found by the edge detector and the actual edge should be as small as possible.
3. **Response:-** The edge detector should not identify multiple edge pixels where only a single edge exists**.**

Steps:

1. Apply derivative of Gaussian
2. Non-maximum suppression

- Thin multi-pixel wide “ridges” down to single pixel width

3. Linking and thresholding

- Low, high edge-strength thresholds

- Accept all edges over low threshold that are connected to edge over high threshold

To improve current methods of edge detectors we must follow

1. The first and most obvious is low error rate. It is important that edges occurring in images should not be missed and that there be no responses to non-edges.
2. The second criterion is that the edge points be well localized i.e., the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum.
3. A third criterion is to have only one response to a single edge

Many of the existing segmentation techniques, such as clustering methods in colour space work well on homogeneous colour regions. Natural scenes are rich in colour and texture. Many texture segmentation algorithms require the estimation of texture model parameters. Parameter estimation is a difficult problem and often requires a good homogeneous region for robust estimation[8].

3 Implementation of kmeans clustering Algorithm.

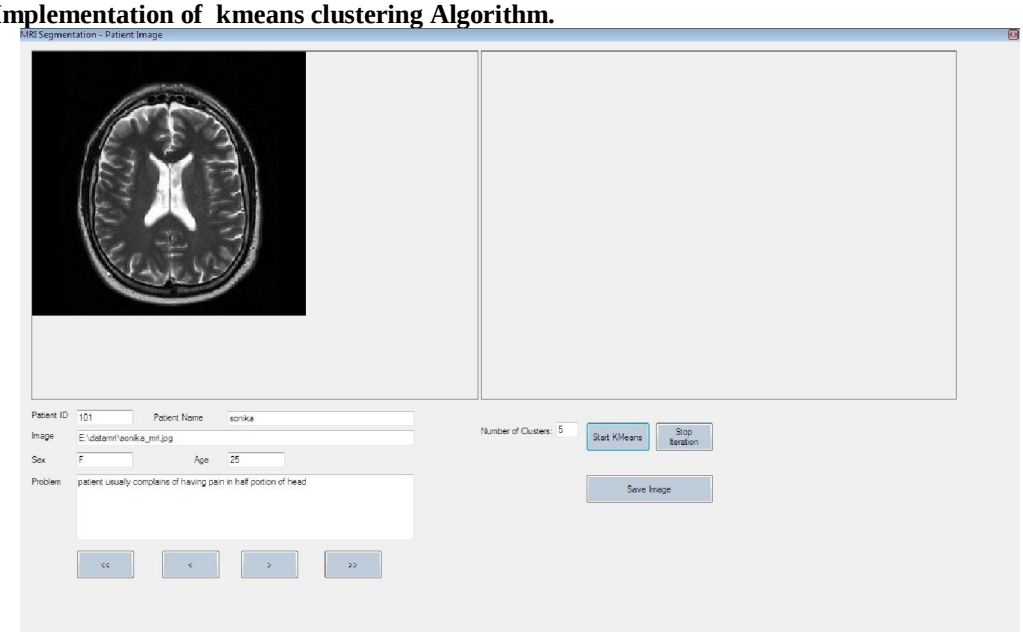

Fig 6: Snapshot of front page of kmeans clustering algorithm

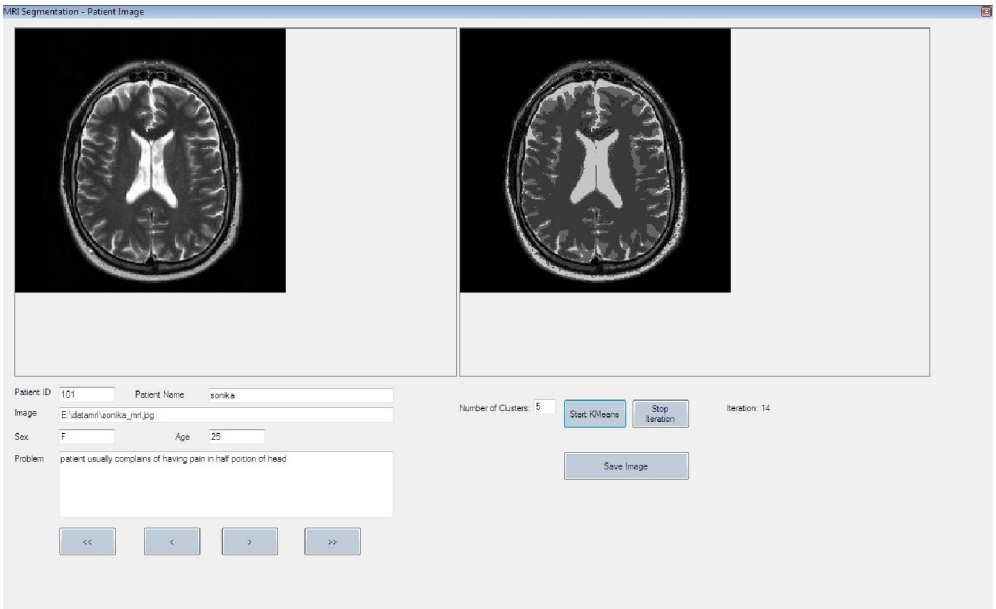

Figure 7. Snapshot of MRI being segmented using kmeans algorithm k=5

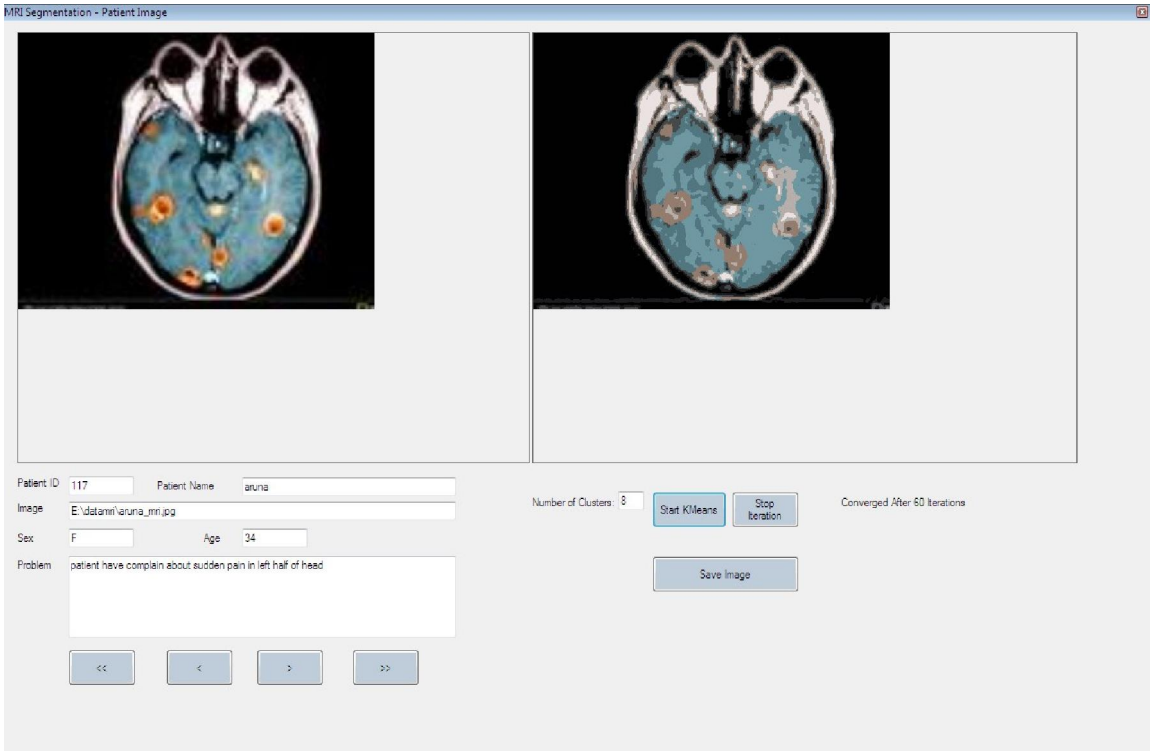

Figure 8: Snapshot of colored MRI segmented by kmeans algorithm with k=8

3.2 Implementation of MRI Segmentation using canny edge detection algorithm.

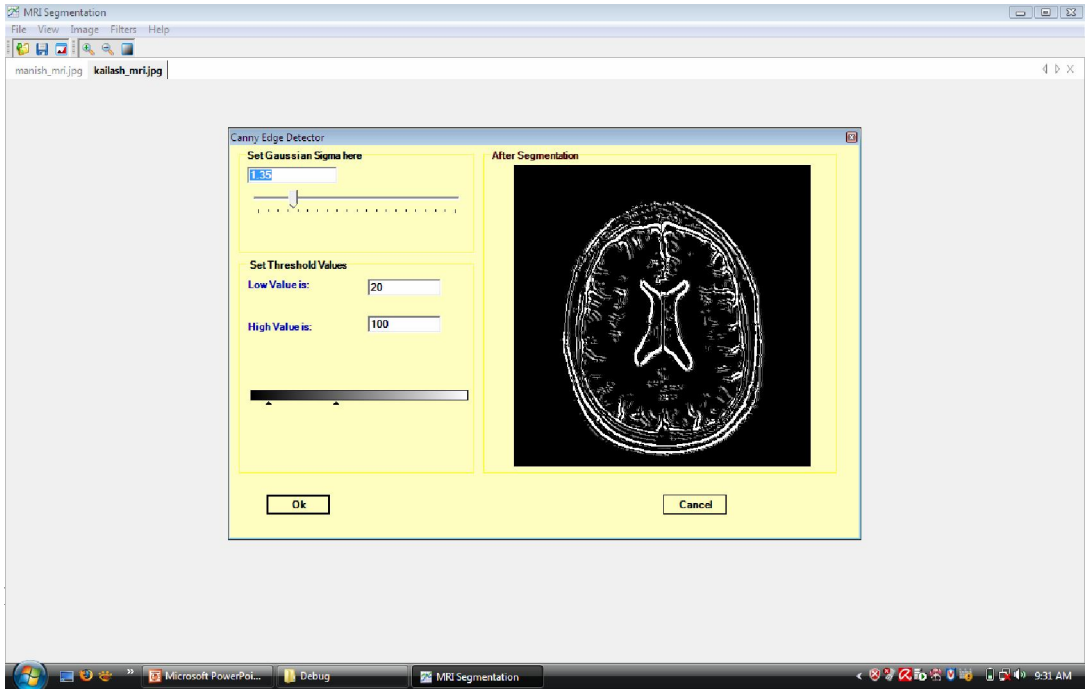

Figure 9. Snapshot of canny edge detection algorithm with T1= 20 and T2=100

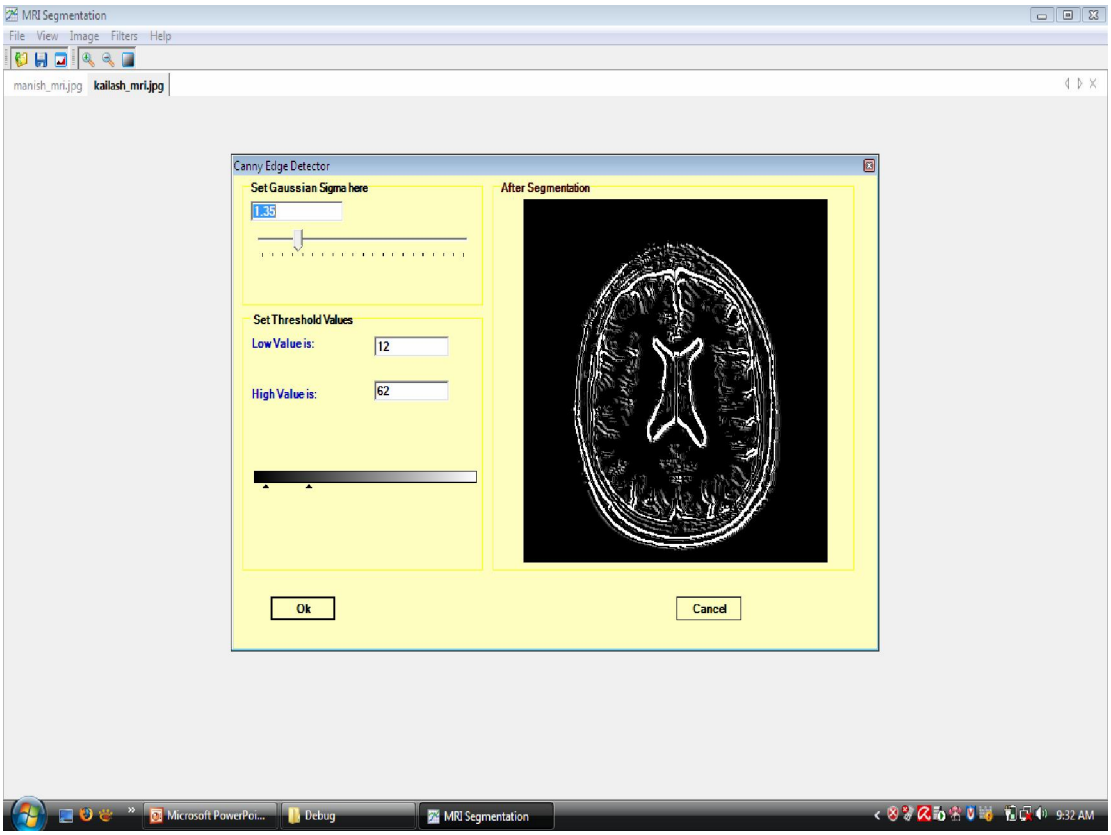

Figure 10: Second Snapshot of canny edge detection algorithm with T1= 12 and T2=62

4. Conclusion and Future scope:

4. 1 **Conclusion and Future scope of kmeans algorithms:** K-means algorithm segments the MRI on the basis of region intensity into K clusters. we note that quality of segmentation increases with increase in the number of clusters but here the value of K is decided by us according to the nature of the MRI and the according to the application used. For example we can decide by seeing the image that how many clusters should be there in the particular image. For example by seeing the number of colors in a colored MRI we can easily guess the value of k. In the snapshot a MRI is segmented with different value of k which will give different results each time.

4.2 Conclusion and Future scope of canny edge Detection algorithm: By Edge detection method we are able to get the boundaries of the different regions in the MRI and the increasing the scale decreases the quality of the segmentation while decreasing the scale improves the quality of the segmentation. Also decreasing the threshold gives us finer segmentation though it detects noise as edge pixels as two threshold values have been used. If the difference between two threshold value will be lesser than we will get fine details of the image otherwise only main boundary of image i.e. the edges which has higher intensity will be visible only.

Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to this edge pixel and that have a value greater than T2 are also selected as edge pixels.

The performance of the Canny algorithm depends heavily on the adjustable parameters, σ, which is the standard deviation for the Gaussian filter, and the threshold values, 'T1' and 'T2'. σ also controls the size of the Gaussian filter. The bigger the value for σ, the larger the size of the Gaussian filter becomes. This implies more blurring, necessary for noisy images, as well as detecting larger edges. As expected, however, the larger the scale of the Gaussian, the less accurate is the localization of the **edge**. Smaller values of σ imply a smaller

Gaussian filter which limits the amount of blurring, maintaining finer edges in the image. The user can tailor the algorithm by adjusting these parameters to adapt to different environments.

References:

1. http://en.wikipedia.org/wiki/K-means_clustering
2. Neeraj Sharma and Lalit M. Aggarwal[1] "Automated medical image segmentation techniques" *J Med Phys.* 2010 Jan–Mar; 35(1): 3–14. doi: 10.4103/0971-6203.58777 PMCID: PMC2825001
3. http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html
4. Suman Tatiraju, Avi Mehta "Image Segmentation using k-means clustering,EM and Normalized Cuts"
5. http://www.codeproject.com/KB/recipes/K-Mean_Clustering.aspx
6. Canny Edge Detection Tutorial (http://www.pages.drexel.edu/~weg22/can_tut.html)
7. Raman Maini, Dr. Himanshu Aggarwal "Study and Comparison of Various Image Edge Detection Techniques" International Journal of Image Processing (IJIP), Volume (3) : Issue (1)
8. H.S.Prasantha, Dr. Shashidhara. H. L, Dr. K. N. B. Murthy, Madhavi Lata.G" MEDICAL IMAGE SEGMENTATION" H.S. Prasantha et. al. / (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 2010, 1209-1218.

5/11/2011