Medical Image Analysis
Sixth hand-in

Mariuta Nicolae (rqt629)

October 20, 2015

K-means clustering

For the k-means clustering I implemented the algorithm very much as it was described at the lecture and applied it for k=2, k=5, k=9 and then displayed the results.

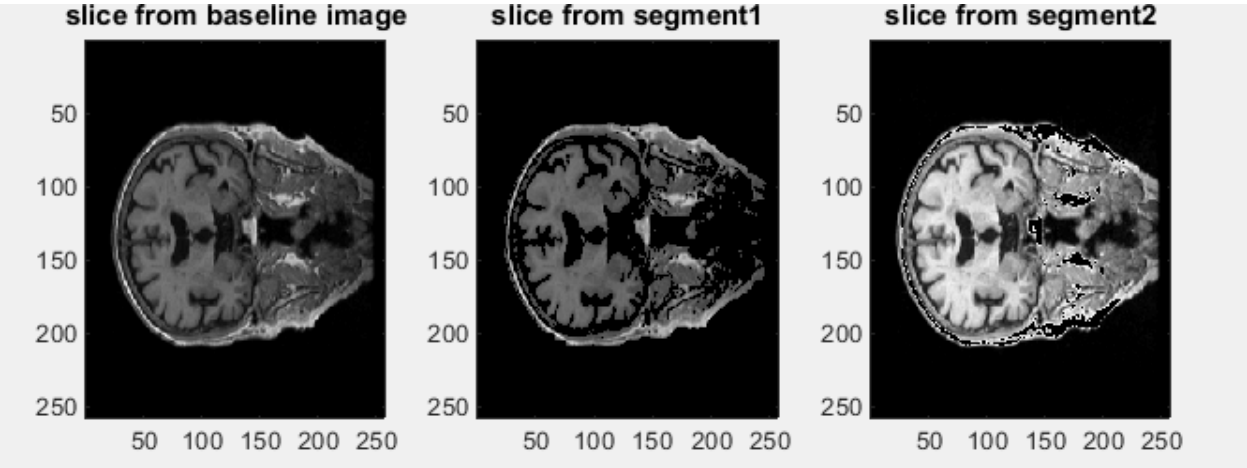
Unfortunately, I did not make the code to be very optimal and it contains much redundant code and I have created separate code for each value of k that I tested. It was mostly because it was hard to have a flexible number of arrays(clusters) without using some more advanced data structures that I do not know how to use them in Matlab.

I started by calculating the histogram and then applied normalization for both parameters used in k-means: the intensity values and number of pixels so I had values between 0 and 1 for both.

In the next step I implemented the standard k-means clustering algorithm for the data I have and after obtaining the final clusters, I used the values of pixels in each of these to segment the image by taking only the values of pixels with intensities within those clusters.

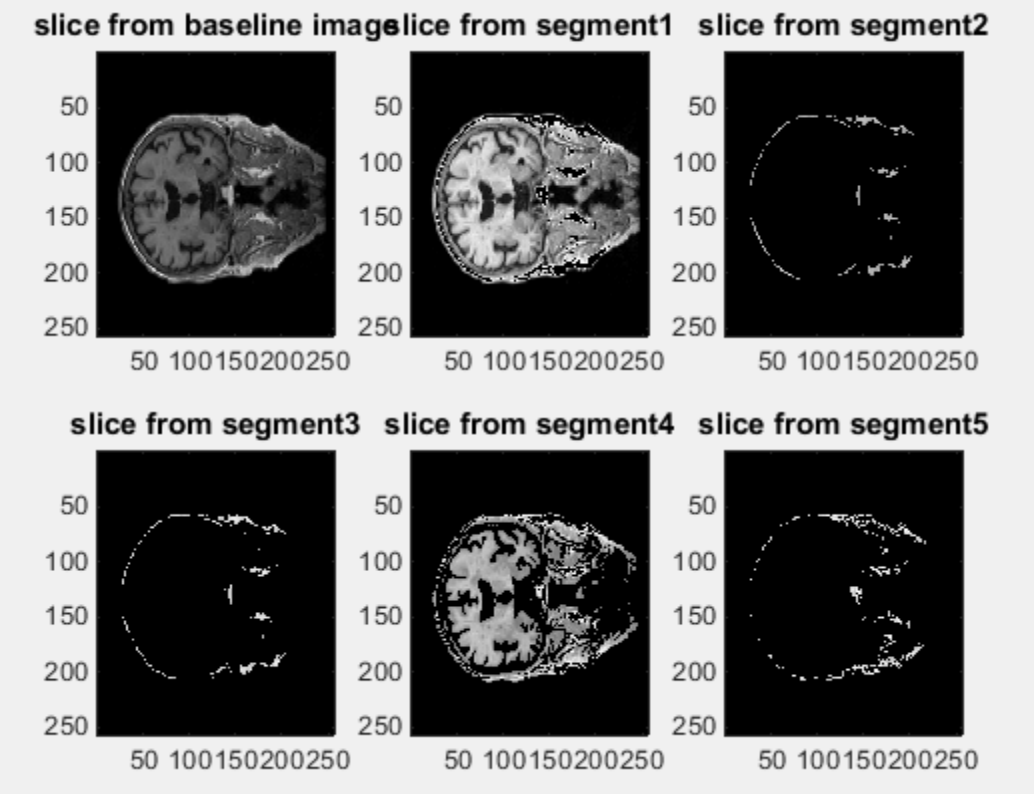
I used the MRI image of the brain and displayed slice 128 sagittal for each segmentation I obtained.

For k=2 I obtained the following result:

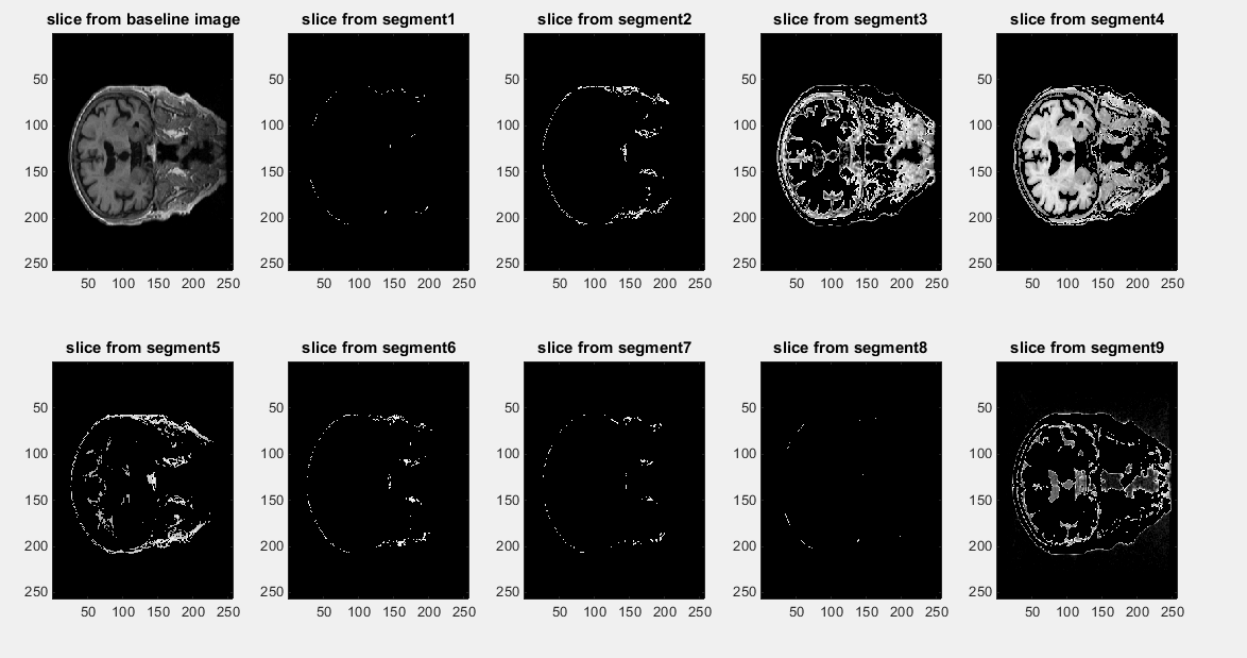


It is visible that in the first segmentation there are darker pixels while in the second segmentation the pixels are very bright. I guess it is mostly showing different types of tissue that have different colors.

For k = 5 I obtained the following results:



And for k = 9 I obtained



For k = 7 I see almost the same thing as for k =2, one segmentation contains brighter pixels, another segmentation contains brighter pixels but the other segmentations contain mostly pixels on the edge of

the head and I guess that is category of pixels with low probability distribution in comparison to segment 1 and 4.

But for k = 9 I see a lot more split into parts of the brain. There it is still a segment that contains the brightest pixels, I guess the one that has highest probability distribution and 3 segmentations contain different components of the brain, I think it is a better split into types of tissues. There are also few segments that contain only some pixels on the margins of the head.

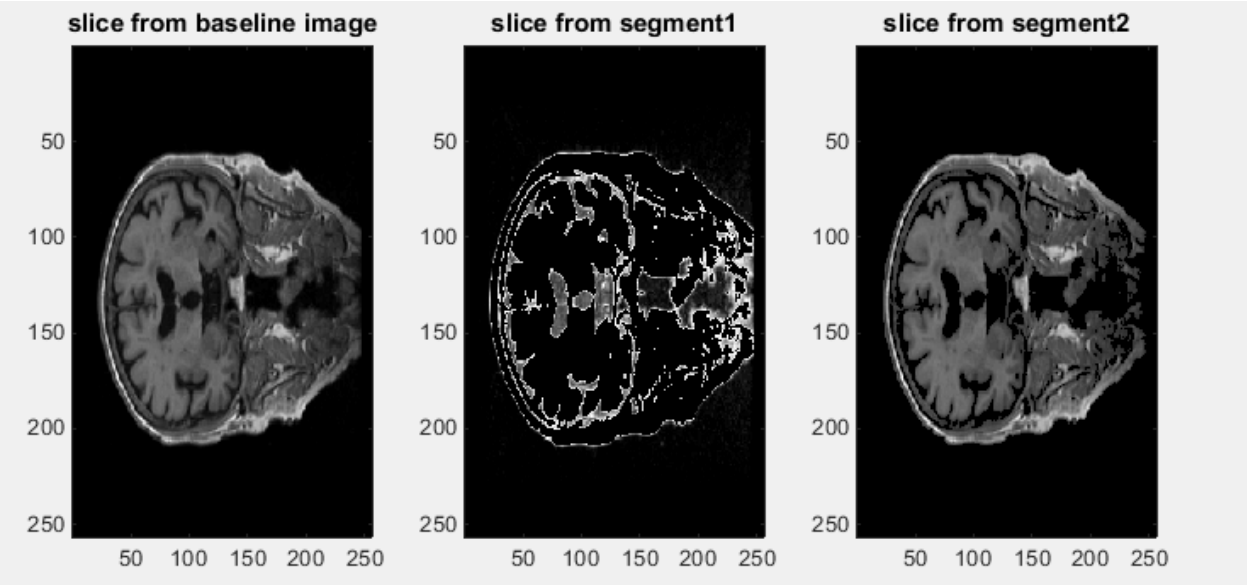
I noticed that for some runs at k=5 or 9 I get error because at different steps some clusters are not getting filled which I guess it is because of the random choice for centroids in the first stage but it happened quite rarely.

Otsu's optimal Threshold

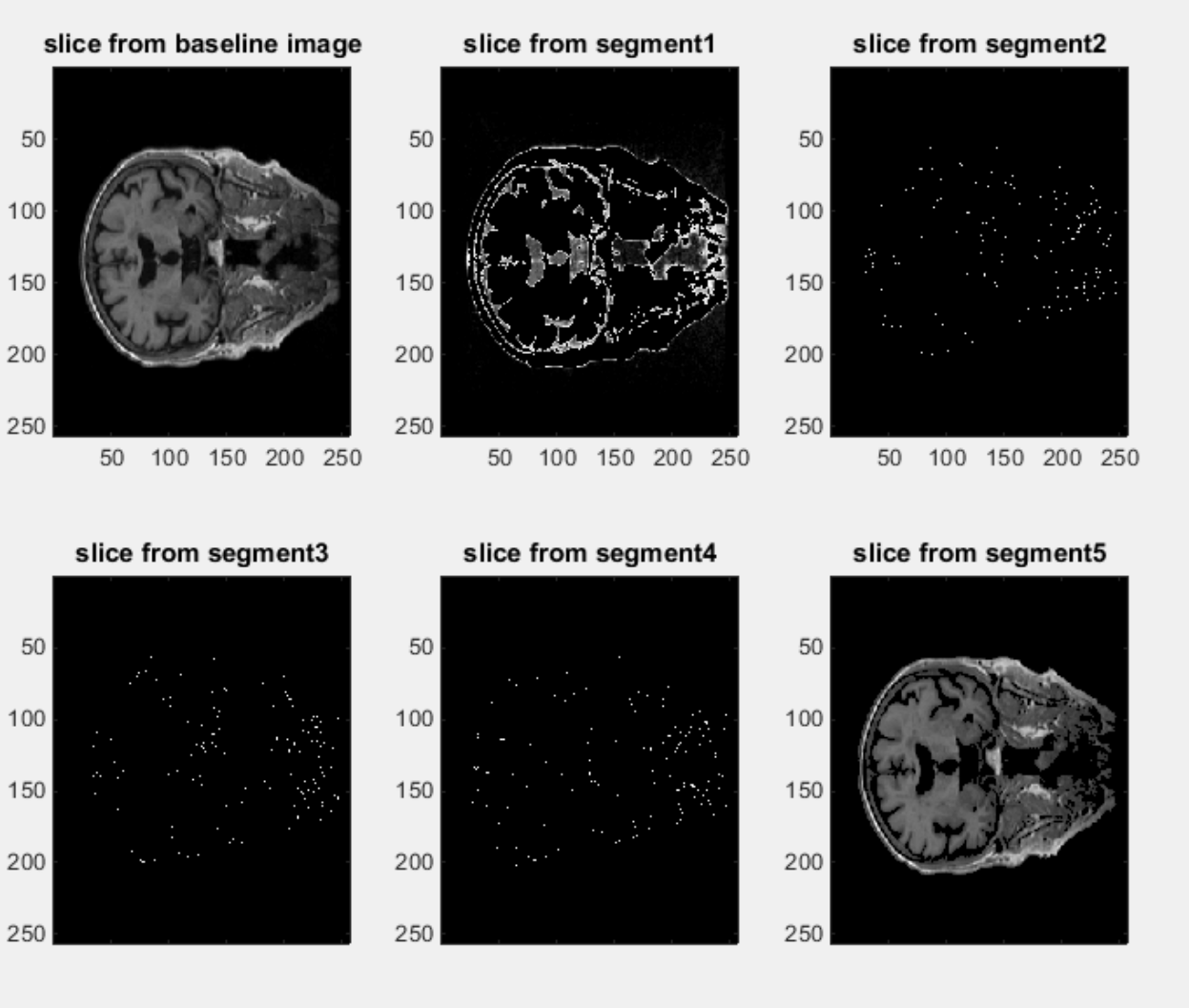
For the implementation of optimal threshold, I followed the algorithm from the paper that was suggested. I started by calculating the normalized histogram, the cumulative sums, the cumulative means and the global intensity mean. Then, I computed the between-class variance and took the optimal value for threshold.

I tried for k=2 by only taking one threshold corresponding to pixel intensity for the largest between-class variance and then also for k=5 by taking the first 5 values. For k=2 also calculated the similarity measure to see how good the results are.

For k=2 I obtained:



And for k=9 I obtained:



It appears that there is not much difference for taking larger number of threshold values, I only get some more segmentations that only contain few pixels without any meaning. But for both cases I obtain 1 segment that looks very much like segment 9 obtained by k-means clustering for k=9,and another segment that looks very much like the first segment obtained or 2 means clustering. I think both algorithms detected a parts of brain with same properties. For optimal threshold I obtained similarity measure 0.80 and I think it is a good value, I expect to get 1 for perfect similarity.

I can conclude that both methods obtain a segmentation looking the same and the 9-means clustering has split the image into more areas with different properties which makes me think it has better performance than the optimal threshold. I think that for the optimal threshold there is a very big impact from the fact that there is a very large number of pixels with value close to 0(the background) and that could make it less efficient for a large number of k and I think I could obtain more meaningful segmentations if I did somehow to reduce the number of pixels surrounding the head that are considered by the algorithm.

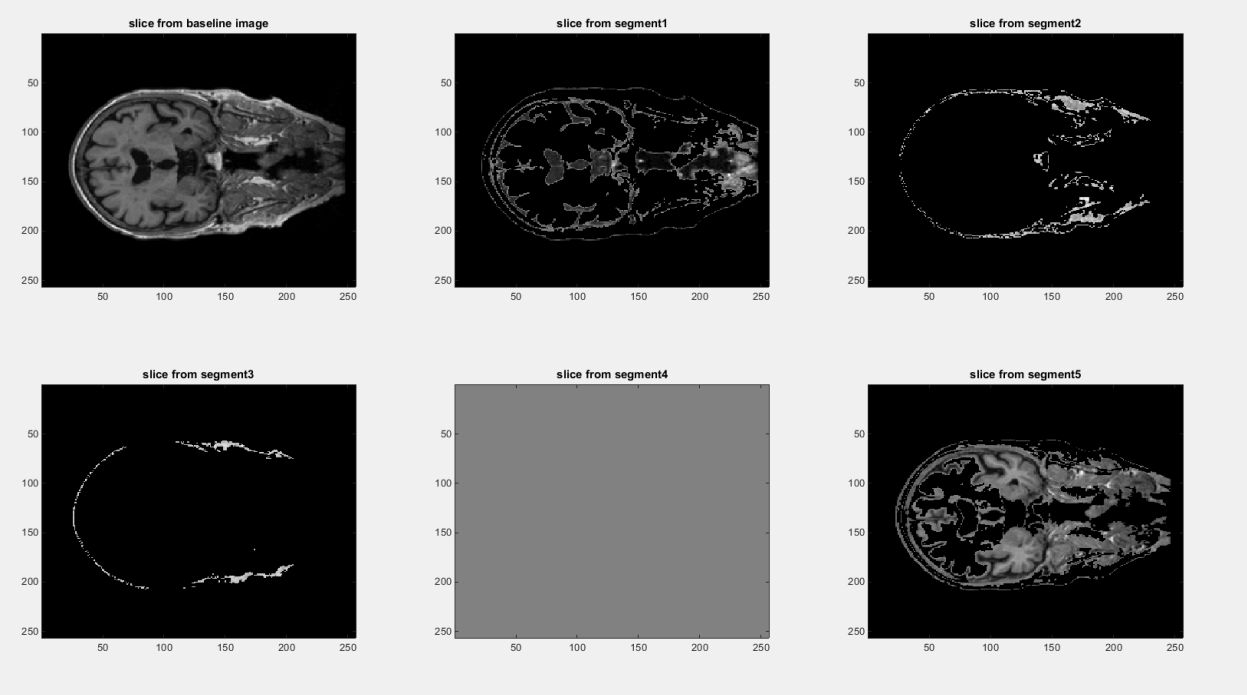
Segmentation using Region Growing

For region growing, I have chosen to use an implementation I found on the Internet because even if the method is quite simple, I did not manage to find a time-efficient algorithm that can process a 3d image and my attempts led to very long computation times by using for loops.

I understood very well that the idea of region growing is to define seeds for regions and then read pixels surrounding those seeds until we find pixels with properties different from the seed, in our case we take the image intensity but I read in documentations that there are also other criteria to decide the region growing threshold, like using properties of texture.

I followed the suggestion from the question in book and used first k-means clustering and took the centroids from those clusters to use as seeds. For me it was a bit confusing that text because if I calculate the spatial mean of the pixels in each cluster, I obtain points outside the regions found by k-means clustering segmentation. I just took mostly random a pixel with intensity equal with the intensity of the centroid in each cluster from the histogram and used those pixels as seeds for each region.

By using this method, with threshold of 30 (there can be difference of 30 in intensity between seed and the pixels within a region) I obtained:



Even with my more random choice of seeds I obtained the first and last segmentation very much like the first and last segmentation obtained from Otsu-s optimal threshold and also very much like some of the segments obtained by k-means clustering. The other regions are also edges of the head with specific intensity of pixels that were also obtained by k-means clustering.

This shows that the region growing is also quite good method and perhaps I could obtain more interesting results by changing the threshold of the region growing but even with this more optimal implementation I found, it still takes long time to compute on my computer so I could not experiment very much.

However, I think this method can obtain more accurately the shape for different parts of the brain by using some interaction with user, by selecting the seed from the image and then apply region growing with lower threshold. Like that, the region growing will not spread much and I think it will segment more accurately different parts of organs. I think by doing like this, region growing may be more efficient than k-means where we cannot have much control over the size of clusters.

Maximum Likelihood segmentation

Unfortunately, I think I did not understand very well this method. I understood that we have to first split the image into regions and then to calculate the likelihood that each pixel in the region to be in that region by using the Bayes' formula. I tried to use the regions obtained by k-means clustering for k -2 and calculate the maximum likelihood for those regions. But I did not understand how to model the function for calculating the likelihood and display the results.

Fuzzy c-means clustering method

For this I had a look on Wikipedia description of this method since it had more information and it was easier to understand. The main difference from k-means is the fact that for fuzzy c-means clustering, we assign each point in several clusters with weights for each cluster indicating its membership level. It usually means that if the membership is high, then the point is closer to the centroid of a specific cluster.

The algorithm starts in the same way as for the k-means by assigning random centroids, computing distances and means to recalculate until we clusters are no longer changing. And then, for each data point we calculate the weight for how much does it belong to each cluster. By doing this, we will have high weight for points close to centroid of a cluster. In order to do segmentation for this, I guess we can set a limit for the weight to decide if we assign a point into segmentation or not. Like this, we can have a pixel in several segments that we obtain. I think this can make the clustering look very much like region growing.

So the main difference is that for k-means we assign each data point to a single cluster, while for fuzzy c-means clustering we have a weight for how much the point belongs to each cluster. Like this, it will be fairer because if a point is around the edge between two clusters, for a classification, we cannot be very certain if that point belongs to a centroid or another.

Hippocampus segmentation in MR images using atlas registration, voxel classification and graph cuts

The suggested paper is presenting a segmentation method based on the minimization of an energy functional with intensity and prior terms, which are derived from manually labeled training images.

The method for selecting the hippocampus in MR images is similar to assigning a label from {0,1} to every voxel p in the image. Hippocampus voxels are assigned label 1 and the background voxels by label 0. The segmentation is modeled by computing a maximum posteriori estimate using the Bayes' theorem.

Finding the MAP is resulting in finding the minimum of an energy function. By taking the negative logarithm of MAP, the segmentation becomes an energy minimization problem.

The prior energy describes any prior knowledge on the class labels. In the work presented in the article, it contains information on the spatial distribution of the labels and their neighbors which is derived from registered training images.

The intensity energy is also calculated by making the product of the individual voxels p. Since the hippocampus consists mostly of grey matter, the foreground likelihood is approximated by a Gaussian density function. Then, the background likelihood is estimated using a Parzen window estimator.

The prior energy is modeled by assuming that the prior probability that the voxel p has the label fp only depends on the position and its direct neighbors. Prior probability is approximated using Markov random field with cliques consisting of one or two voxel sites.

After these calculations, optimization is applied by converting the functional to a directional graph and computing the minimum s-t cut. The graph has a set of nodes, one for every voxel in the image and two terminal nodes(source and sink). Edges have non-negative weights assigned based on energy functional. Finding the minimum energy corresponds to finding the minimum cut and there are several algorithms to solve this in polynomial time.

This method has proven to perform usually better than multi-atlas-based approach in the tests that were made.

Anexes with the code

```
% kmeans clusttering k = 2
clear;
%load the images and rescale so I can compute
B = my_load_mgh('nu1.mgz');

I = imresize3d(B,1,[],'linear','bound');
I(isnan(I)) = 0;

%calculate histogram
pixels = floor(double(I(:)))+1;
histogram = accumarray(pixels,1);
data = [(1:256)'/256  histogram/max(histogram)];

%get initial random centroids
nrCentroids = 2;
centroids = [];
for i = 1:nrCentroids
    r = floor(rand*256)+1;
    centroids = [centroids ; data(r,1)  data(r,2)];
end

%initialize the clusters
cluster1 = [];
cluster2 = [];

for i = 1:100
    %fill the clusters
    for j = 1 : size(data,1)
        point = squeeze(data(j,:));
        d1 = sqrt((centroids(1,1) - point(1))^2 + (centroids(1,2) -
point(2))^2 );
        d2 = sqrt((centroids(2,1) - point(1))^2 + (centroids(2,2) -
point(2))^2 );

        if(d1<d2)
            cluster1 = [cluster1 ; point];
        else
            cluster2 = [cluster2 ; point];
        end
     end

    %calculate new means
    %calcualte mean for each cluster and find next centroids
    m1 = mean(cluster1);
    m2 = mean(cluster2);

    distances1 = sqrt(sum((cluster1 - ones(size(cluster1))*diag(m1)),2).^ 2);
    distances2 = sqrt(sum((cluster2 - ones(size(cluster2))*diag(m2)),2).^ 2);

    c1 = find(distances1 == min(distances1(:)));
    c2 = find(distances2 == min(distances2(:)));
```

```
    centroids(1,:) = cluster1(c1(1),:);
    centroids(2,:) = cluster2(c2(1),:);

    disp(i);

end

%find actual values of pixels in each cluster
[sharedVals1,idxs1] = intersect(data(:,1),cluster1(:,1));
[sharedVals2,idxs2] = intersect(data(:,1),cluster2(:,1));

%segment the image
segment1 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs1)));
segment1(lind) = I(lind);

segment2 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs2)));
segment2(lind) = I(lind);;

subplot(1,3,1),imagesc((squeeze(I(:,:,128)))),colormap('gray'),title('slice
from baseline image');
subplot(1,3,2),imagesc((squeeze(segment1(:,:,128)))),colormap('gray'),title('
slice from segment1');
subplot(1,3,3),imagesc((squeeze(segment2(:,:,128)))),colormap('gray'),title('
slice from segment2');



%%
% kmeans clusttering k = 5
clear;
%load the images and rescale so I can compute
B = my_load_mgh('nu1.mgz');

I = imresize3d(B,1,[],'linear','bound');
I(isnan(I)) = 0;

%calculate histogram
pixels = floor(double(I(:)))+1;
histogram = accumarray(pixels,1);
data = [(1:256)'/256  histogram/max(histogram)];

%get initial random centroids
nrCentroids = 5;
centroids = [];
for i = 1:nrCentroids
    r = floor(rand*256)+1;
    centroids = [centroids ; data(r,1)  data(r,2)];
end

%initialize the clusters
cluster1 = [];
cluster2 = [];
```

```
cluster3 = [];
cluster4 = [];
cluster5 = [];

for i = 1:100
    %fill the clusters
    for j = 1 : size(data,1)
        point = squeeze(data(j,:));
        d = [];
        d = [d sqrt((centroids(1,1) - point(1))^2 + (centroids(1,2) -
point(2))^2 )];
        d = [d sqrt((centroids(2,1) - point(1))^2 + (centroids(2,2) -
point(2))^2 )];
        d = [d sqrt((centroids(3,1) - point(1))^2 + (centroids(3,2) -
point(2))^2 )];
        d = [d sqrt((centroids(4,1) - point(1))^2 + (centroids(4,2) -
point(2))^2 )];
        d = [d sqrt((centroids(5,1) - point(1))^2 + (centroids(5,2) -
point(2))^2 )];
        [m,im] = min(d);

        if(im == 1)
            cluster1 = [cluster1 ; point];
        elseif(im == 2)
            cluster2 = [cluster2 ; point];
        elseif(im == 3)
            cluster3 = [cluster3 ; point];
        elseif(im == 4)
            cluster4 = [cluster4 ; point];
        else
            cluster5 = [cluster5 ; point];
        end
     end

    %calculate new means
    %calcualte mean for each cluster and find next centroids
    m1 = mean(cluster1);
    m2 = mean(cluster2);
    m3 = mean(cluster3);
    m4 = mean(cluster4);
    m5 = mean(cluster5);

    distances1 = sqrt(sum((cluster1 - ones(size(cluster1))*diag(m1)),2).^ 2);
    distances2 = sqrt(sum((cluster2 - ones(size(cluster2))*diag(m2)),2).^ 2);
    distances3 = sqrt(sum((cluster3 - ones(size(cluster3))*diag(m3)),2).^ 2);
    distances4 = sqrt(sum((cluster4 - ones(size(cluster4))*diag(m4)),2).^ 2);
    distances5 = sqrt(sum((cluster5 - ones(size(cluster5))*diag(m5)),2).^ 2);

    c1 = find(distances1 == min(distances1(:)));
    c2 = find(distances2 == min(distances2(:)));
    c3 = find(distances3 == min(distances3(:)));
    c4 = find(distances4 == min(distances4(:)));
    c5 = find(distances5 == min(distances5(:)));

    centroids(1,:) = cluster1(c1(1),:);
    centroids(2,:) = cluster2(c2(1),:);
```

```
    centroids(3,:) = cluster3(c3(1),:);
    centroids(4,:) = cluster4(c4(1),:);
    centroids(5,:) = cluster5(c5(1),:);

    disp(i);

end

%find actual values of pixels in each cluster
[sharedVals1,idxs1] = intersect(data(:,1),cluster1(:,1));
[sharedVals2,idxs2] = intersect(data(:,1),cluster2(:,1));
[sharedVals3,idxs3] = intersect(data(:,1),cluster3(:,1));
[sharedVals4,idxs4] = intersect(data(:,1),cluster4(:,1));
[sharedVals5,idxs5] = intersect(data(:,1),cluster5(:,1));

%segment the image
segment1 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs1)));
segment1(lind) = I(lind);

segment2 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs2)));
segment2(lind) = I(lind);

segment3 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs3)));
segment3(lind) = I(lind);

segment4 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs4)));
segment4(lind) = I(lind);

segment5 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs5)));
segment5(lind) = I(lind);

subplot(2,3,1),imagesc((squeeze(I(:,:,128)))),colormap('gray'),title('slice
from baseline image');
subplot(2,3,2),imagesc((squeeze(segment1(:,:,128)))),colormap('gray'),title('
slice from segment1');
subplot(2,3,3),imagesc((squeeze(segment2(:,:,128)))),colormap('gray'),title('
slice from segment2');
subplot(2,3,4),imagesc((squeeze(segment3(:,:,128)))),colormap('gray'),title('
slice from segment3');
subplot(2,3,5),imagesc((squeeze(segment4(:,:,128)))),colormap('gray'),title('
slice from segment4');
subplot(2,3,6),imagesc((squeeze(segment5(:,:,128)))),colormap('gray'),title('
slice from segment5');

%%
% kmeans clusttering k = 9
clear;
%load the images and rescale so I can compute
B = my_load_mgh('nu1.mgz');
```

```
I = imresize3d(B,1,[],'linear','bound');
I(isnan(I)) = 0;

%calculate histogram
pixels = floor(double(I(:)))+1;
histogram = accumarray(pixels,1);
data = [(1:256)'/256  histogram/max(histogram)];

%get initial random centroids
nrCentroids = 9;
centroids = [];
for i = 1:nrCentroids
    r = floor(rand*256)+1;
    centroids = [centroids ; data(r,1)  data(r,2)];
end

%initialize the clusters
cluster1 = [];
cluster2 = [];
cluster3 = [];
cluster4 = [];
cluster5 = [];
cluster6 = [];
cluster7 = [];
cluster8 = [];
cluster9 = [];

for i = 1:100
    %fill the clusters
    for j = 1 : size(data,1)
        point = squeeze(data(j,:));
        d = [];
        d = [d sqrt((centroids(1,1) - point(1))^2 + (centroids(1,2) -
point(2))^2 )];
        d = [d sqrt((centroids(2,1) - point(1))^2 + (centroids(2,2) -
point(2))^2 )];
        d = [d sqrt((centroids(3,1) - point(1))^2 + (centroids(3,2) -
point(2))^2 )];
        d = [d sqrt((centroids(4,1) - point(1))^2 + (centroids(4,2) -
point(2))^2 )];
        d = [d sqrt((centroids(5,1) - point(1))^2 + (centroids(5,2) -
point(2))^2 )];
        d = [d sqrt((centroids(6,1) - point(1))^2 + (centroids(6,2) -
point(2))^2 )];
        d = [d sqrt((centroids(7,1) - point(1))^2 + (centroids(7,2) -
point(2))^2 )];
        d = [d sqrt((centroids(8,1) - point(1))^2 + (centroids(8,2) -
point(2))^2 )];
        d = [d sqrt((centroids(9,1) - point(1))^2 + (centroids(9,2) -
point(2))^2 )];
        [m,im] = min(d);

        if(im == 1)
            cluster1 = [cluster1 ; point];
        elseif(im == 2)
            cluster2 = [cluster2 ; point];
```

```
    elseif(im == 3)
        cluster3 = [cluster3 ; point];
    elseif(im == 4)
        cluster4 = [cluster4 ; point];
    elseif(im == 5)
        cluster5 = [cluster5 ; point];
    elseif(im == 6)
        cluster6 = [cluster6 ; point];
    elseif(im == 7)
        cluster7 = [cluster7 ; point];
    elseif(im == 8)
        cluster8 = [cluster8 ; point];
    else
        cluster9 = [cluster9 ; point];
    end
 end

%calculate new means
%calcualte mean for each cluster and find next centroids
m1 = mean(cluster1);
m2 = mean(cluster2);
m3 = mean(cluster3);
m4 = mean(cluster4);
m5 = mean(cluster5);
m6 = mean(cluster6);
m7 = mean(cluster7);
m8 = mean(cluster8);
m9 = mean(cluster9);

distances1 = sqrt(sum((cluster1 - ones(size(cluster1))*diag(m1)),2).^ 2);
distances2 = sqrt(sum((cluster2 - ones(size(cluster2))*diag(m2)),2).^ 2);
distances3 = sqrt(sum((cluster3 - ones(size(cluster3))*diag(m3)),2).^ 2);
distances4 = sqrt(sum((cluster4 - ones(size(cluster4))*diag(m4)),2).^ 2);
distances5 = sqrt(sum((cluster5 - ones(size(cluster5))*diag(m5)),2).^ 2);
distances6 = sqrt(sum((cluster6 - ones(size(cluster6))*diag(m6)),2).^ 2);
distances7 = sqrt(sum((cluster7 - ones(size(cluster7))*diag(m7)),2).^ 2);
distances8 = sqrt(sum((cluster8 - ones(size(cluster8))*diag(m8)),2).^ 2);
distances9 = sqrt(sum((cluster9 - ones(size(cluster9))*diag(m9)),2).^ 2);

c1 = find(distances1 == min(distances1(:)));
c2 = find(distances2 == min(distances2(:)));
c3 = find(distances3 == min(distances3(:)));
c4 = find(distances4 == min(distances4(:)));
c5 = find(distances5 == min(distances5(:)));
c6 = find(distances6 == min(distances6(:)));
c7 = find(distances7 == min(distances7(:)));
c8 = find(distances8 == min(distances8(:)));
c9 = find(distances9 == min(distances9(:)));

centroids(1,:) = cluster1(c1(1),:);
centroids(2,:) = cluster2(c2(1),:);
centroids(3,:) = cluster3(c3(1),:);
centroids(4,:) = cluster4(c4(1),:);
centroids(5,:) = cluster5(c5(1),:);
centroids(6,:) = cluster6(c6(1),:);
centroids(7,:) = cluster7(c7(1),:);
```

```
    centroids(8,:) = cluster8(c8(1),:);
    centroids(9,:) = cluster9(c9(1),:);

    disp(i);

end

%find actual values of pixels in each cluster
[sharedVals1,idxs1] = intersect(data(:,1),cluster1(:,1));
[sharedVals2,idxs2] = intersect(data(:,1),cluster2(:,1));
[sharedVals3,idxs3] = intersect(data(:,1),cluster3(:,1));
[sharedVals4,idxs4] = intersect(data(:,1),cluster4(:,1));
[sharedVals5,idxs5] = intersect(data(:,1),cluster5(:,1));
[sharedVals6,idxs6] = intersect(data(:,1),cluster6(:,1));
[sharedVals7,idxs7] = intersect(data(:,1),cluster7(:,1));
[sharedVals8,idxs8] = intersect(data(:,1),cluster8(:,1));
[sharedVals9,idxs9] = intersect(data(:,1),cluster9(:,1));

%segment the image
segment1 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs1)));
segment1(lind) = I(lind);

segment2 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs2)));
segment2(lind) = I(lind);

segment3 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs3)));
segment3(lind) = I(lind);

segment4 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs4)));
segment4(lind) = I(lind);

segment5 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs5)));
segment5(lind) = I(lind);

segment6 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs6)));
segment6(lind) = I(lind);

segment7 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs7)));
segment7(lind) = I(lind);

segment8 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs8)));
segment8(lind) = I(lind);

segment9 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs9)));
segment9(lind) = I(lind);
```

```
%
subplot(2,5,1),imagesc((squeeze(I(:,:,128)))),colormap('gray'),title('slice
from baseline image');
subplot(2,5,2),imagesc((squeeze(segment1(:,:,128)))),colormap('gray'),title('
slice from segment1');
subplot(2,5,3),imagesc((squeeze(segment2(:,:,128)))),colormap('gray'),title('
slice from segment2');
subplot(2,5,4),imagesc((squeeze(segment3(:,:,128)))),colormap('gray'),title('
slice from segment3');
subplot(2,5,5),imagesc((squeeze(segment4(:,:,128)))),colormap('gray'),title('
slice from segment4');
subplot(2,5,6),imagesc((squeeze(segment5(:,:,128)))),colormap('gray'),title('
slice from segment5');
subplot(2,5,7),imagesc((squeeze(segment6(:,:,128)))),colormap('gray'),title('
slice from segment6');
subplot(2,5,8),imagesc((squeeze(segment7(:,:,128)))),colormap('gray'),title('
slice from segment7');
subplot(2,5,9),imagesc((squeeze(segment8(:,:,128)))),colormap('gray'),title('
slice from segment8');
subplot(2,5,10),imagesc((squeeze(segment9(:,:,128)))),colormap('gray'),title(
'slice from segment9');

%% Otsu's optimal value threshold for 2 segments
clear;
% load the images and rescale so I can compute
B = my_load_mgh('nu1.mgz');

I = imresize3d(B,1,[],'linear','bound');
I(isnan(I)) = 0;

% calculate normalized histogram
pixels = floor(double(I(:)))+1;
histogram = accumarray(pixels,1);
data = [(0:255)'  histogram/numel(I)];

%compute the cumulative sums
cumSums = zeros(size(data));
for i = 0:255
    cumSums(i+1,:) = [i sum(data(1:(i+1),2))];
end

%compute the cumulative means
cumMeans = zeros(size(data));
for i = 0:255
    cumMeans(i+1,:) = [i sum(data(1:(i+1),2).*(0:i)')];
end

%compute global intensity mean
mG = sum(data(:,2).*(0:255)');

%compute between-class variance
sigmab = ((mG*cumSums(:,2) - cumMeans(:,2)).^2)./(cumSums(:,2).*(1-
cumSums(:,2)));
sigmab(isnan(sigmab)) = 0;
```

```
%optimal threshold
[ksigmab, kOptimal] = max(sigmab);

%apply segmentation
segment1 = zeros(size(I));
lind = sub2ind(size(I),find(I<kOptimal));
segment1(lind) = I(lind);


segment2 = zeros(size(I));
lind = sub2ind(size(I),find(I>=kOptimal));
segment2(lind) = I(lind);

%evaluate separability measure
sigmag = sum(data(:,2).*(((0:255) - mG)').^2);
measure = ksigmab/sigmag;



%display results
subplot(1,3,1),imagesc((squeeze(I(:,:,128)))),colormap('gray'),title('slice
from baseline image');
subplot(1,3,2),imagesc((squeeze(segment1(:,:,128)))),colormap('gray'),title('
slice from segment1');
subplot(1,3,3),imagesc((squeeze(segment2(:,:,128)))),colormap('gray'),title('
slice from segment2');

%% Otsu's optimal value threshold for 5 segments
clear;
% load the images and rescale so I can compute
B = my_load_mgh('nu1.mgz');

I = imresize3d(B,1,[],'linear','bound');
I(isnan(I)) = 0;

% calculate normalized histogram
pixels = floor(double(I(:)))+1;
histogram = accumarray(pixels,1);
data = [(0:255)'  histogram/numel(I)];

%compute the cumulative sums
cumSums = zeros(size(data));
for i = 0:255
    cumSums(i+1,:) = [i sum(data(1:(i+1),2))];
end

%compute the cumulative means
cumMeans = zeros(size(data));
for i = 0:255
    cumMeans(i+1,:) = [i sum(data(1:(i+1),2).*(0:i)')];
end

%compute global intensity mean
mG = sum(data(:,2).*(0:255)');
```

```
%compute between-class variance
sigmab = ((mG*cumSums(:,2) - cumMeans(:,2)).^2)./(cumSums(:,2).*(1-
cumSums(:,2)));
sigmab(isnan(sigmab)) = 0;
%optimal threshold
[ksigmab, kOptimal] = sort(sigmab);

kOptimal = sort(kOptimal(252:256,:));
%apply segmentation
segment1 = zeros(size(I));
lind = sub2ind(size(I),find(I<kOptimal(1)));
segment1(lind) = I(lind);


segment2 = zeros(size(I));
lind = sub2ind(size(I),find(I>=kOptimal(1) & I<kOptimal(2)));
segment2(lind) = I(lind);

segment3 = zeros(size(I));
lind = sub2ind(size(I),find(I>=kOptimal(2) & I<kOptimal(3)));
segment3(lind) = I(lind);

segment4 = zeros(size(I));
lind = sub2ind(size(I),find(I>=kOptimal(4) & I<kOptimal(5)));
segment4(lind) = I(lind);

segment5 = zeros(size(I));
lind = sub2ind(size(I),find(I>=kOptimal(5)));
segment5(lind) = I(lind);

%display results
subplot(2,3,1),imagesc((squeeze(I(:,:,128)))),colormap('gray'),title('slice
from baseline image');
subplot(2,3,2),imagesc((squeeze(segment1(:,:,128)))),colormap('gray'),title('
slice from segment1');
subplot(2,3,3),imagesc((squeeze(segment2(:,:,128)))),colormap('gray'),title('
slice from segment2');
subplot(2,3,4),imagesc((squeeze(segment3(:,:,128)))),colormap('gray'),title('
slice from segment3');
subplot(2,3,5),imagesc((squeeze(segment4(:,:,128)))),colormap('gray'),title('
slice from segment4');
subplot(2,3,6),imagesc((squeeze(segment5(:,:,128)))),colormap('gray'),title('
slice from segment5');

%% Region growing
clear;
%load the images and rescale so I can compute
B = my_load_mgh('nu1.mgz');

I = imresize3d(B,1,[],'linear','bound');
I(isnan(I)) = 0;

%calculate histogram
pixels = floor(double(I(:)))+1;
```

```
histogram = accumarray(pixels,1);
data = [(1:256)'/256  histogram/max(histogram)];

%get initial random centroids
nrCentroids = 5;
centroids = [];
for i = 1:nrCentroids
    r = floor(rand*256)+1;
    centroids = [centroids ; data(r,1)  data(r,2)];
end

%initialize the clusters
cluster1 = [];
cluster2 = [];
cluster3 = [];
cluster4 = [];
cluster5 = [];

for i = 1:100
    %fill the clusters
    for j = 1 : size(data,1)
        point = squeeze(data(j,:));
        d = [];
        d = [d sqrt((centroids(1,1) - point(1))^2 + (centroids(1,2) -
point(2))^2 )];
        d = [d sqrt((centroids(2,1) - point(1))^2 + (centroids(2,2) -
point(2))^2 )];
        d = [d sqrt((centroids(3,1) - point(1))^2 + (centroids(3,2) -
point(2))^2 )];
        d = [d sqrt((centroids(4,1) - point(1))^2 + (centroids(4,2) -
point(2))^2 )];
        d = [d sqrt((centroids(5,1) - point(1))^2 + (centroids(5,2) -
point(2))^2 )];
        [m,im] = min(d);

        if(im == 1)
            cluster1 = [cluster1 ; point];
        elseif(im == 2)
            cluster2 = [cluster2 ; point];
        elseif(im == 3)
            cluster3 = [cluster3 ; point];
        elseif(im == 4)
            cluster4 = [cluster4 ; point];
        else
            cluster5 = [cluster5 ; point];
        end
     end

    %calculate new means
    %calcualte mean for each cluster and find next centroids
    m1 = mean(cluster1);
    m2 = mean(cluster2);
    m3 = mean(cluster3);
    m4 = mean(cluster4);
    m5 = mean(cluster5);
```

```
    distances1 = sqrt(sum((cluster1 - ones(size(cluster1))*diag(m1)),2).^ 2);
    distances2 = sqrt(sum((cluster2 - ones(size(cluster2))*diag(m2)),2).^ 2);
    distances3 = sqrt(sum((cluster3 - ones(size(cluster3))*diag(m3)),2).^ 2);
    distances4 = sqrt(sum((cluster4 - ones(size(cluster4))*diag(m4)),2).^ 2);
    distances5 = sqrt(sum((cluster5 - ones(size(cluster5))*diag(m5)),2).^ 2);

    c1 = find(distances1 == min(distances1(:)));
    c2 = find(distances2 == min(distances2(:)));
    c3 = find(distances3 == min(distances3(:)));
    c4 = find(distances4 == min(distances4(:)));
    c5 = find(distances5 == min(distances5(:)));

    centroids(1,:) = cluster1(c1(1),:);
    centroids(2,:) = cluster2(c2(1),:);
    centroids(3,:) = cluster3(c3(1),:);
    centroids(4,:) = cluster4(c4(1),:);
    centroids(5,:) = cluster5(c5(1),:);

    disp(i);

end

%find actual values of pixels in each cluster
[sharedVals1,idxs1] = intersect(data(:,1),cluster1(:,1));
[sharedVals2,idxs2] = intersect(data(:,1),cluster2(:,1));
[sharedVals3,idxs3] = intersect(data(:,1),cluster3(:,1));
[sharedVals4,idxs4] = intersect(data(:,1),cluster4(:,1));
[sharedVals5,idxs5] = intersect(data(:,1),cluster5(:,1));

%segment the image
segment1 = zeros(size(I));
lind1 = sub2ind(size(I),find(ismember(I,idxs1)));
segment1(lind1) = I(lind1);

segment2 = zeros(size(I));
lind2 = sub2ind(size(I),find(ismember(I,idxs2)));
segment2(lind2) = I(lind2);

segment3 = zeros(size(I));
lind3 = sub2ind(size(I),find(ismember(I,idxs3)));
segment3(lind3) = I(lind3);

segment4 = zeros(size(I));
lind4 = sub2ind(size(I),find(ismember(I,idxs4)));
segment4(lind4) = I(lind4);

segment5 = zeros(size(I));
lind5 = sub2ind(size(I),find(ismember(I,idxs5)));
segment5(lind5) = I(lind5);

%calculating the seeds by taking the mean of each segmentation
c1 = find(ismember(data,centroids(1,1)));
[x1,y1,z1] = ind2sub(size(I),find(ismember(I,c1)));
seed1 =[x1(100),y1(100),z1(100)];
```

```
c2 = find(ismember(data,centroids(2,1)));
[x2,y2,z2] = ind2sub(size(I),find(ismember(I,c2)));
seed2 = [x2(100),y2(100),z2(100)];

c3 = find(ismember(data,centroids(3,1)));
[x3,y3,z3] = ind2sub(size(I),find(ismember(I,c3)));
seed3 = [x3(100),y3(100),z3(100)];

c4 = find(ismember(data,centroids(4,1)));
[x4,y4,z4] = ind2sub(size(I),find(ismember(I,c4)));
seed4 = [x4(100),y4(100),z4(100)];

c5 = find(ismember(data,centroids(5,1)));
[x5,y5,z5] = ind2sub(size(I),find(ismember(I,c5)));
seed5 = [x5(100),y5(100),z5(100)];

[~,m1] = regionGrowing(squeeze(I),seed1,20, Inf, [], true, false);
[~,m2] = regionGrowing(squeeze(I),seed2,20, Inf, [], true, false);
[~,m3] = regionGrowing(squeeze(I),seed3,20, Inf, [], true, false);
[~,m4] = regionGrowing(squeeze(I),seed4,20, Inf, [], true, false);
[~,m5] = regionGrowing(squeeze(I),seed5,20, Inf, [], true, false);

%apply obtained masks to see the segmented regions
I1 = I.*m1;
I2 = I.*m2;
I3 = I.*m3;
I4 = I.*m4;
I5 = I.*m5;

%display results
subplot(2,3,1),imagesc((squeeze(I(:,:,128)))),colormap('gray'),title('slice
from baseline image');
subplot(2,3,2),imagesc((squeeze(I1(:,:,128)))),colormap('gray'),title('slice
from segment1');
subplot(2,3,3),imagesc((squeeze(I2(:,:,128)))),colormap('gray'),title('slice
from segment2');
subplot(2,3,4),imagesc((squeeze(I3(:,:,128)))),colormap('gray'),title('slice
from segment3');
subplot(2,3,5),imagesc((squeeze(I4(:,:,128)))),colormap('gray'),title('slice
from segment4');
subplot(2,3,6),imagesc((squeeze(I5(:,:,128)))),colormap('gray'),title('slice
from segment5');



%% Maximum likelihood segmentation

clear;
%load the images and rescale so I can compute
B = my_load_mgh('nu1.mgz');

I = imresize3d(B,1,[],'linear','bound');
I(isnan(I)) = 0;
```

```
%calculate histogram
pixels = floor(double(I(:)))+1;
histogram = accumarray(pixels,1);
data = [(1:256)'/256  histogram/max(histogram)];

%get initial random centroids
nrCentroids = 2;
centroids = [];
for i = 1:nrCentroids
    r = floor(rand*256)+1;
    centroids = [centroids ; data(r,1)  data(r,2)];
end

%initialize the clusters
cluster1 = [];
cluster2 = [];

for i = 1:100
    %fill the clusters
    for j = 1 : size(data,1)
        point = squeeze(data(j,:));
        d1 = sqrt((centroids(1,1) - point(1))^2 + (centroids(1,2) -
point(2))^2 );
        d2 = sqrt((centroids(2,1) - point(1))^2 + (centroids(2,2) -
point(2))^2 );

        if(d1<d2)
            cluster1 = [cluster1 ; point];
        else
            cluster2 = [cluster2 ; point];
        end
     end

    %calculate new means
    %calcualte mean for each cluster and find next centroids
    m1 = mean(cluster1);
    m2 = mean(cluster2);

    distances1 = sqrt(sum((cluster1 - ones(size(cluster1))*diag(m1)),2).^ 2);
    distances2 = sqrt(sum((cluster2 - ones(size(cluster2))*diag(m2)),2).^ 2);

    c1 = find(distances1 == min(distances1(:)));
    c2 = find(distances2 == min(distances2(:)));

    centroids(1,:) = cluster1(c1(1),:);
    centroids(2,:) = cluster2(c2(1),:);

    disp(i);

end

%find actual values of pixels in each cluster
[sharedVals1,idxs1] = intersect(data(:,1),cluster1(:,1));
```

```
[sharedVals2,idxs2] = intersect(data(:,1),cluster2(:,1));

%segment the image
segment1 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs1)));
segment1(lind) = I(lind);

segment2 = zeros(size(I));
lind = sub2ind(size(I),find(ismember(I,idxs2)));
segment2(lind) = I(lind);;

%apply maximum likelihood on the resulting regions

%number of elements in each region
n1 = numel(segment1);
n2 = numel(segment2);

histogram1 = accumarray(floor(double(n1(:)))+1,1);
data1 = [(0:255)'  histogram1/numel(n1)];

histogram2 = accumarray(floor(double(n2(:)))+1,1);
data2 = [(0:255)'  histogram2/numel(n2)];
```