Computer Vision and Imaging Extended [06 30241]
Assessed Assignment

- Kevin Bupalam Sudhir (2356685)

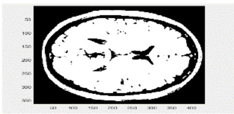
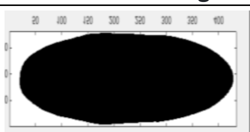
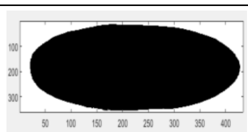
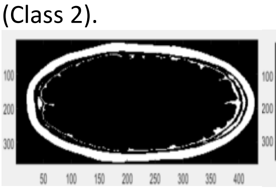
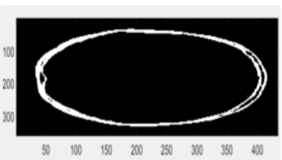
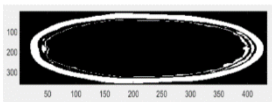
Task 1: 2D tissue segmentation

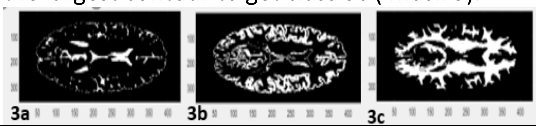
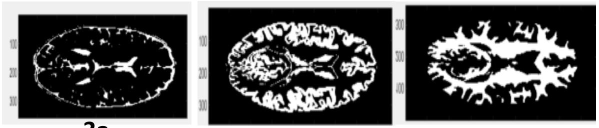
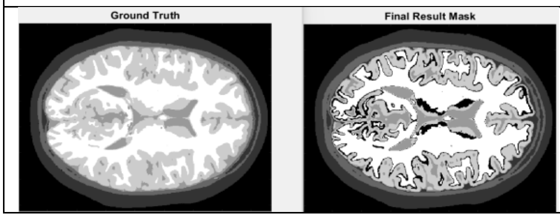
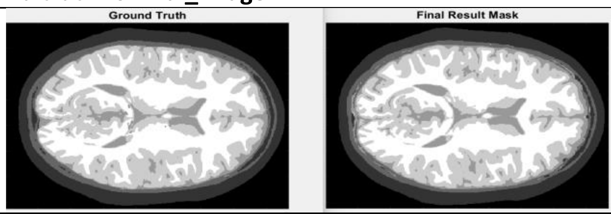
Aim: Develop and apply different segmentation algorithms, based on any technique you have learnt to each slice of the MRI data. You need to apply exactly the same algorithm to every slice.

Methodology: 2D tissue segmentation can be implemented using a wide variety of image processing techniques based on the threshold, clustering, edge, region, etc. In this assignment, the image processing technique used is based on the threshold image processing technique. The algorithms used are Manual thresholding and Multi class Otsu thresholding.

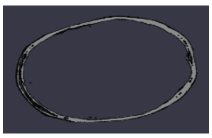
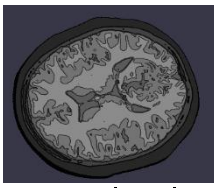
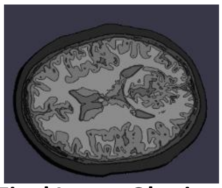
Environment used: MATLAB R2021b **Code:** MATLAB code

Implementation adapted for the manual and multi class Otsu thresholding:

<table>
<tr><th>Manual Thresholding</th><th>Multi Class Otsu thresholding</th></tr>
<tr><td>Manual thresholding algorithm manually calculates, and classifies the pixels in an input image and can be divided into ‘n’ number of classes according to the intensity of grey levels within that image.</td><td>Multi Class Otsu thresholding is a thresholding method that divides pixels in an input image into ‘n’ number of classes according to the intensity of grey levels within that image.</td></tr>
<tr><td colspan="2">1. Load the given Brain.mat data set.
2. Iterate through each image and label pair using for loop.
3. Assign Class 1 as outer portion and Class 3 to 5 as inner portions.
</td></tr>
<tr><td>4. Manually binarize image using mat2gray() and imbinarize() function by using grey threshold value.
5. Then use manual thresholding to segment these classes.</td><td>4. Use multithresh() function to automatically create binary image.
5. Then use Multi Class Otsu thresholding to segment these classes.</td></tr>
<tr><td colspan="2">6 . Get the two biggest contours and sort the connected components by area. Classes 3 to 5 has the biggest contour and second biggest is the outer ring.</td></tr>
<tr><td colspan="2">Class 0 – Background class implementation</td></tr>
<tr><td>7. Manually creating background mask image from the outer mask value obtained through outer ring mask and complementing it.
</td><td>7. Use active contour to create image background mask, fill that mask with ‘holes’ and create component for that image.
</td></tr>
<tr><td colspan="2">Class 1 and 2 – Outer portion implementation</td></tr>
<tr><td>8. Manually find the closest value for the outer ring mask and update the same(Class 1).
9. From the obtained outer ring mask, complement it.
10. Then subtract the filled inner mask and the background mask to get the inner ring mask (Class 2).
</td><td>8. Close all the holes in inner potion by using imfill() and extract the mask.
9. Using this image, create another image by replacing all the pixels of value 1 (in step 8) to value 0.
10. Extract the mask of Class 1 and complement it.

Extract class 2 then subtract inner filled mask and background mask by using multithresh() and imquantize() to apply multi class thresholding and further divide image mask into 3 masks.</td></tr>
</table>

<table>
<tr><th>Manual Thresholding (contd..)</th><th>Multi Class Otsu thresholding (contd..)</th></tr>
<tr><td colspan="2">Classes 3-5 – Inner portion implementation</td></tr>
<tr><td colspan="2">11. Using imerode() function get the first biggest contour from the step 6.</td></tr>
<tr><td>12. Segment the largest contour into 3 classes by manually getting the value of the mask 3a and subtract it with background mask, outer ring mask and inner ring mask to get Class 3a. Similarly, get the value of mask 3b, then complement it and subtract it with class 3a to get class 3b.
13. Finally, get the mask 3c value manually from the largest contour to get class 3c (Mask 5).

</td><td>12. Segment the largest contour into 3 classes by applying Multi Class Otsu.
13. Finally, Class 3a (Mask 3), 3b (Mask 4), 3c (Mask 5) of the biggest contour are extracted depending on the image indexes.

3a 3b 3c</td></tr>
<tr><td colspan="2">14. Final result Ground truth vs Final_Image</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

Task 3: Advanced 2D tissue segmentation (3D Segmentation)

<table>
<tr><th colspan="2">Multi Class Otsu 3D thresholding</th></tr>
<tr><td>1. Load the given Brain.mat data set.
2. Using 3D input scan and Multi Class thresholding segment Class 1 as outer portion and Class 3 to 5 as inner portions.</td><td>Classes 3 to 5: 9. Using imerode() function get the first biggest contour from the step 6.
10. Segment the largest contour into 3 classes by applying Multi Class Otsu.</td></tr>
<tr><td>3 . Get the two biggest contours and sort the connected components by area. Classes 3 to 5 has the biggest component and the second biggest is the outer ring.
</td><td>11. Finally, Class 3a (Mask 3), 3b (Mask 4), 3c (Mask 5) of the biggest contour are extracted depending on the image indexes.</td></tr>
<tr><td>Background Class: 4. Use active contour to create background mask, fill the mask using imfill3d() function and create component for that image.
</td><td>
3a 3b 3c</td></tr>
<tr><td>Class 1 and 2: 5. Close all the holes in inner potion by using imfill3d() and extract the 3D mask.
6. Using this image create another image by replacing all the pixels of value 1 (step 5) to value 0.
7. Apply multi class thresholding to further divide image mask into 3 masks and extract the mask of Class 1 and complement it.
8. Extract class 2 mask and then subtract inner filled mask and background mask.
Class 1   Class 2</td><td>12. Result: Ground truth vs Final Image Obtained
 
Ground Truth Final Image Obtained
 </td></tr>
</table>

Task 2: Result evaluation

The results are calculated using the following metrics:
Structural Similarity Measure(SSIM): It is a measure for evaluating the visual effect of three image characteristics: brightness, contrast, and structure.
Similarity/Intersection over Union (IOU): It is a metric to calculate the extent of overlap between two areas. (In our case ground truth and the final label were obtained)
Dice Score: This is another metric that is similar to IOU where the dice coefficient is used to get the similarity of the given two samples.
Mean Score: This is the score that is obtained by calculating the average of the above-mentioned scores. This will give us the overall value to compare which algorithm suits the best.

RESULT SCORES:

Manual thresholding 2D Scores (Table 1)			
Mean Score = 0.829 Time = 2.17s			
Metric	SSIM	DICE	IOU
M0	0.670	0.890	0.802
M1	0.899	0.919	0.851
M2	0.974	0.957	0.917
M3	0.814	0.673	0.507
M4	0.700	0.789	0.651
M5	0.976	0.989	0.979

Multi class Otsu thresholding 2D Scores (Table 2)			
Mean Score = 0.916 Time = 5.16s			
Metric	SSIM	DICE	IOU
M0	0.987	0.997	0.994
M1	0.960	0.977	0.939
M2	0.955	0.922	0.845
M3	0.879	0.849	0.725
M4	0.888	0.958	0.896
M5	0.936	0.979	0.931

Multi class Otsu thresholding 3D Scores (Table 3)			
Mean Score = 0.917 Time = 11.83s			
Metric	SSIM	DICE	IOU
M0	0.987	0.997	0.995
M1	0.948	0.972	0.945
M2	0.954	0.916	0.846
M3	0.859	0.851	0.742
M4	0.874	0.945	0.897
M5	0.892	0.963	0.930

CONCLUSION:

1. **Result Evaluation:**
   We can observe from Tables 1, 2, and 3 that all the algorithms perform the best for the masks/classes 0, 1, 2, and 5 as these classes do not have a complex structure and are also easily differentiable. We can also observe that the mask 4 performance is average, this is because the mask 4 has a lot of vacant space and is complex to get the right values. Additionally, we can observe that mask 3 performs the worst because there is a slight overlapping of class boundaries which also have a thin mask thus leading to huge error margins. Finally, we can observe that the mean values of Multi class Otsu thresholding 2D and 3D are almost the same whereas the Manual thresholding mean value is very less compared to Multi class. To summarize, Multi class Otsu thresholding is preferred over Manual thresholding.
   Note: The time taken by Multiclass Otsu thresholding 2D and 3D is very high because of the active contour implementation, if active contour implementation is removed then the time taken to execute decreases significantly.

2. **Selected algorithm for Task 3 and its reasoning**
   We can clearly observe from tables 1 and 2 that the performance of Manual Thresholding 2D scores per each mask/class score suffers when compared to the Multi class Otsu thresholding 2D. This behaviour is because the values of the masks in the manual thresholding are very vague and implemented manually. One more observation is that the scores suffer a lot for the masks 3 and 4 in manual thresholding as the area has a very complex structure with a lot of holes. The mean scores of the manual and multi Otsu are 0.829 and 0.916 i.e. 82.9 and 91.6 percent respectively. There is an approximate drop of 9 percent which concludes that the manual thresholding performance is bad and hence the Multi class Otsu is selected for the task 3 3D segmentation as its mean score is very good.

3. **Comparing Multi class Otsu 2D and 3D approaches:**
   When the Multi class Otsu 2D is compared to Multi class Otsu 3D, from tables 2 and 3 we can see one major difference in the time where the 3D implementation takes almost twice the time than 2D algorithm. This shows that 3D algorithm is slower than 2D algorithm. We can also observe that the performance of both the methods are very similar and there is a negligible difference between scores. The performance is expected to be similar as the internal functioning of algorithms are pretty similar.

CODES/SCRIPTS:

Below are the codes which are used to implement the 2D and 3D segmentation. The codes are also in the following GitHub repository:
[CV_FinalAssignment_GitHubKevin]

Code 1: manualThreshold2D.m

```
% Loading the provided data set
load Brain.mat

% Initilization of score arrays
similarity_score = zeros(6,1,'double');
ssim_array = zeros(6, 1, 'double');
dice_score = zeros(6, 1, 'double');

% Display set for image output
figure(); colormap gray; axis equal; axis off;

%Traverse over all the images and labels from data
set
for i=1:10
    image = T1(:,:,i); % Reading the ith image
    l = label(:,:,i);  % Reading the ith label

   % Creating final mask to store each segmented
mask
    final_mask = zeros(size(l));

    % Use manual thresholding to create binary
image
    image = mat2gray(image);
    level = graythresh(image);
    outer = imbinarize(image,level);
    % Sort connected components by size
    [v, n] = bwlabel(outer, 8);
    flag = sum(bsxfun(@eq,v(:),1:n));
    % Finding and assigning the 2 biggest contours
    [v2, n2] = maxk(flag, 2);
    t1 = n2(1);
    t2 = n2(2);
    inval = v == t1;
    % Filling the holes in inner mask
    fill_imask = imfill(inval, 'holes');

    % Outer ring mask creation
    out_imask = image;
    outval = fill_imask == 1;
    out_imask(outval) = 0;

  % Find the manual value for the outer ring mask
  % and update outer ring and outer mask to final
mask
    out_imask = imadjust(out_imask);
    out_rmask = out_imask > 2.352900e-01; % Value
for the outer ring mask
    outer_vals = out_rmask == 1;
    final_mask(outer_vals)=1;
    subplot(6,1,2); % Plotting mask 1
    imagesc(out_rmask);
    caption = sprintf('Mask 1');
    title(caption, 'FontSize', 8);



% Manually creating background mask from outer ring
mask
    bg_mask = imfill(out_rmask, 'holes');
    bg_mask = imcomplement(bg_mask);
    l_values = bg_mask == 1;
    final_mask(l_values)=0;
    lindex=2;
    subplot(6,1,1); % Plotting mask 0
    imagesc(bg_mask);
    caption = sprintf('Mask 0 - Background mask');
    title(caption, 'FontSize', 8);

    % Manually creating inner ring and inner mask
from outer ring mask
    inner_rmask = imcomplement(out_rmask);
    inner_rmask =
imsubtract(inner_rmask,fill_imask);
    inner_rmask =
imsubtract(inner_rmask,double(bg_mask));
    inner_vals = inner_rmask == 1;
    final_mask(inner_vals)=lindex;
    lindex = lindex+1;
    subplot(6,1,3); % Plotting mask 2
    imagesc(inner_rmask);
    caption = sprintf('Mask 2');
    title(caption, 'FontSize', 8);

    % Creating 3 different masks for inner
components
    f = strel('disk', 5);
    in_imask = image;
    in_vals = imerode(fill_imask == 0, f);
    in_imask(in_vals) = 0;

  % Manual calculation for generating mask class
3(A)
    Xmin = min(in_imask(:));
    Xmax = max(in_imask(:));
    if isequal(Xmax,Xmin)
        in_imask = 0*in_imask;
    else
       in_imask = (in_imask - Xmin) ./ (Xmax -
Xmin);
    end

 % Find the manual value for the 3(A) inner ring
mask
 % Generating Mask class 3(A) using inner mask and
outer mask
    temp_imask = imcomplement(in_imask);
    l3a_vals = temp_imask > 4.588200e-01;
img_temp = imsubtract(l3a_vals,bg_mask);
    temp_mask = zeros((size(l)));
    vals1 = img_temp==1;
    temp_mask(vals1) = 1;
    vals2 = img_temp == -1;
    temp_mask(vals2) = 0;
    l3a_vals = temp_mask == 1;
    img_temp2 = imsubtract(l3a_vals,out_rmask);
    temp_mask2 = zeros((size(l)));
    vals11 = img_temp2==1;
    temp_mask2(vals11) = 1;
    vals22 = img_temp2 == -1;
    temp_mask2(vals22) = 0;
    l3a_vals = temp_mask2 == 1;
    img_temp3 = imsubtract(l3a_vals,inner_vals);
    temp_mask3 = zeros((size(l)));
    vals111 = img_temp3==1;
    temp_mask3(vals111) = 1;
    vals222 = img_temp3 == -1;
    temp_mask3(vals222) = 0;
    l3a_vals = temp_mask3 == 1;
    final_mask(l3a_vals) = lindex;
    lindex = lindex+1;
    subplot(6,1,4); % Plotting mask 3
```

```
imagesc(l3a_vals);
    caption = sprintf('Mask 3');
    title(caption, 'FontSize', 8);

 % Find the manual value for the 3(B) inner ring
mask
    % Generating mask class 3(B)
    l3b_vals = in_imask > 7.137300e-01;
l3b_vals = imcomplement(l3b_vals);
    img_temp =
imsubtract(l3b_vals,imcomplement(fill_imask));
    temp_mask = zeros((size(l)));
    vals1 = img_temp==1;
    temp_mask(vals1) = 1;
    vals2 = img_temp == -1;
    temp_mask(vals2) = 1;
    temp_mask =
imsubtract(temp_mask,double(l3a_vals));
    l3b_vals = temp_mask == 1;
    final_mask(l3b_vals) = lindex;
    lindex = lindex+1;
    subplot(6,1,5); % Plotting mask 4
    imagesc(l3b_vals);
    caption = sprintf('Mask 4');
    title(caption, 'FontSize', 8);

 % Find the manual value for the 3(C) inner ring
mask
    % Generating Mask class 3(C)
    l3c_vals = in_imask > 7.764700e-01;
final_mask(l3c_vals) = lindex;
    lindex = lindex+1;
    subplot(6,1,6); % Plotting mask 5
    imagesc(l3c_vals);
    caption = sprintf('Mask 5');
    title(caption, 'FontSize', 8);

    % Calculate metrics for each mask
    similarity = jaccard(categorical(l),
categorical(final_mask));
    similarity_score = similarity_score +
similarity;
    dice_val = dice(categorical(l),
categorical(final_mask));
    dice_score = dice_score + dice_val;
    ssim_score = get_ssim_scores(l, final_mask);
    ssim_array = ssim_array + ssim_score;
end

similarity_score = similarity_score / 10;
ssim_array = ssim_array / 10;
dice_score = dice_score / 10;
% Compute mean of all three calculated scores
mean_score = (similarity_score + ssim_array +
dice_score) / 3;
meanval = mean(mean_score);

% Plotting final result mask
figure();colormap gray; axis equal; axis off;
imagesc(final_mask);
caption = sprintf('Final Result Mask');
title(caption, 'FontSize', 14);
% Plotting ground truth
figure();colormap gray; axis equal; axis off;
imagesc(l);
caption = sprintf('Ground Truth');
title(caption, 'FontSize', 14);
```

Code 2: threshold2D.m

```
% Loading the provided data set
load Brain.mat

% Display set for image output
figure(); colormap gray; axis equal; axis off;

% Initilization of score arrays
similarity_score = zeros(6,1,'double');
ssim_array = zeros(6, 1, 'double');
dice_score = zeros(6, 1, 'double');

%Traverse over all the images and labels from data
set
for i=1:10

    image = T1(:,:,i); % Reading the ith image
    l = label(:,:,i); % Reading the ith label

  % Creating final mask to store each segmented
mask
    final_mask = zeros(size(l));
    lindex=0;

    % Use Multi thresholding to create binary image
    t = multithresh(image, 1);
    outer = imquantize(image,t);
    portion_mask = outer == 2;
    % Sort connected components by size
    [v, n] = bwlabel(portion_mask, 8);
    flag = sum(bsxfun(@eq,v(:),1:n));
    % Finding and assigning the 2 biggest contours
    [v2, n2] = maxk(flag, 2);
    t1 = n2(1);
    t2 = n2(2);
    % Creating active contour image for outer mask
    mask = zeros(size(l));
    mask(25:end-25,25:end-25) = 1;
    img_ac = activecontour(image,mask,100);
    inval = v == t1;
    % Deriving the background mask by using image
from active contour
    mask_background = imcomplement(imfill(img_ac,
'holes'));
    f = strel('disk', 9);
    % Filling the remaining holes in inner mask
    fill_imask = imclose(inval, f);
    subplot(6,1,1); % Plotting mask 0
    imagesc(mask_background);
    caption = sprintf('Mask 0 - Background mask');
    title(caption, 'FontSize', 8);

    % Background mask creation and updating it to
final mask
    l_values = mask_background == 1;
    final_mask(l_values)=lindex;
    lindex=lindex+1;

    % Outer ring mask creation
    out_imask = image;
    outval = fill_imask == 1;
    out_imask(outval) = 0;
    t = multithresh(out_imask, 2); % Image mask
conversion into 2 classes
    v = imquantize(out_imask, t);

   % Updating outer ring and outer mask to final
mask
    out_rmask = v == 1;
    out_rmask = imcomplement(out_rmask);
    outer_vals = out_rmask == 1;
    final_mask(outer_vals)=lindex;
```

```
    lindex = lindex+1;
    subplot(6,1,2); % Plotting mask 1
    imagesc(out_rmask);
    caption = sprintf('Mask 1');
    title(caption, 'FontSize', 8);
   % Updating inner ring and inner mask to final
mask
    inner_rmask = zeros((size(l)));
    inner_ring_vals = v == 2;
    inner_rmask(v == 1) = 1;
    bg_vals = mask_background == 1;
    inner_rmask(bg_vals) = 0;
    inner_vals = fill_imask ==1;
    inner_rmask(inner_vals) = 0;
    inner_mask_vals = inner_rmask == 1;
    final_mask(inner_mask_vals)=lindex;
    lindex = lindex+1;
    subplot(6,1,3); % Plotting mask 2
    imagesc(inner_rmask);
    caption = sprintf('Mask 2');
    title(caption, 'FontSize', 8);

    % Creating masks for inner components
    f = strel('disk', 8);
    in_imask = image;
    in_vals = imerode(fill_imask == 0, f);
    in_imask(in_vals) = 0;
    % Segmenting inner mask using Multi class Otsu
Thresholding
    t = multithresh(in_imask, 3); % Into 3 classes
    v_in = imquantize(in_imask, t);

    % Mask class 3(A)
    l3a_vals = v_in == 2;
    final_mask(l3a_vals) = lindex;
    lindex = lindex+1;
    subplot(6,1,4); % Plotting mask 3
    imagesc(l3a_vals);
    caption = sprintf('Mask 3');
    title(caption, 'FontSize', 8);

    % Mask class 3(B)
    l3b_vals = v_in == 3;
    final_mask(l3b_vals) = lindex;
    lindex = lindex+1;
    subplot(6,1,5); % Plotting mask 4
    imagesc(l3b_vals);
    caption = sprintf('Mask 4');
    title(caption, 'FontSize', 8);

    % Mask class 3(C)
    l3c_vals = v_in == 4;
    final_mask(l3c_vals) = lindex;
    lindex = lindex+1;
    subplot(6,1,6); % Plotting mask 5
    imagesc(l3c_vals);
    caption = sprintf('Mask 5');
    title(caption, 'FontSize', 8);

    % Calculate metrics for each score
    similarity = jaccard(categorical(l),
categorical(final_mask));
    similarity_score = similarity_score +
similarity;
    dice_val = dice(categorical(l),
categorical(final_mask));
    dice_score = dice_score + dice_val;
    ssim_score = get_ssim_scores(l, final_mask);
    ssim_array = ssim_array + ssim_score;
end
    similarity_score = similarity_score / 10;
ssim_array = ssim_array / 10;
```

```
dice_score = dice_score / 10;
%Compute mean of all three calculated scores
mean_score = (similarity_score + ssim_array +
dice_score) / 3;
mean_val = mean(mean_score);

% Plotting final result mask
figure();colormap gray; axis equal; axis off;
imagesc(final_mask);
caption = sprintf('Final Result Mask');
title(caption, 'FontSize', 14);
% Plotting ground truth
figure();colormap gray; axis equal; axis off;
imagesc(l);
caption = sprintf('Ground Truth');
title(caption, 'FontSize', 14);
```

Code 3: threshold3D.m

```
% Loading the provided data set
load Brain.mat

image = T1; % Reading the image
l = label; % Reading the label
% Creating final mask to store each segmented mask
final_mask = zeros(size(l));
lindex = 0;

% Use Multi thresholding to create binary image
t = multithresh(image, 1);
outer = imquantize(image, t);
p_mask = zeros(size(outer));
t_vals = outer == 2;
p_mask(t_vals) = 1;
v = bwlabeln(p_mask); % Get components that are
connected
in_rmask = zeros(size(outer));
invals = v == 3;
in_rmask(invals) = 1;
% Creating active contour image for outer mask
mask = zeros(size(l));
mask(25:end-25,25:end-25) = 1;
img_ac = activecontour(image,mask,100);
% Deriving the background mask by using image from
active contour in 3D
mask_background = imfill3d(img_ac);
% Filling the remaining holes in inner mask in 3D
fill_imask = imfill3d(in_rmask);

% Background mask creation and updating it to final
mask
maskbg = zeros((size(l)));
l_values = mask_background == 0;
maskbg(l_values) = 1;
final_mask(l_values) = lindex;
lindex = lindex+1;

% Outer ring 3D mask creation
out_mask = image;
outvals = fill_imask == 1;
out_mask(outvals) = 0;
t = multithresh(out_mask, 2); % Image mask
conversion into 2 classes
v = imquantize(out_mask, t);

% Updating outer ring and outer mask to final mask
out_rmask = zeros((size(l)));
outval = v == 1;
out_rmask(outval) = 1;
out_rmask = imcomplement(out_rmask);
outer_rvals = out_rmask == 1;
```

```
final_mask(outer_rvals)=lindex;
lindex = lindex+1;

% Updating inner ring and inner mask to final mask
in_rmask = zeros((size(l)));
in_rvals = v == 2;
in_rmask(outval) = 1;
l_values = maskbg == 1;
in_rmask(l_values) = 0;
invals = fill_imask ==1;
in_rmask(invals) = 0;
in_maskvals = in_rmask == 1;
final_mask(in_maskvals)=lindex;
lindex = lindex+1;

% Creating masks for inner components in 3D
f = strel('disk', 8);
in_imask = image;
invals = imdilate(fill_imask, f)==0;
in_imask(invals) = 0;
% Segmenting inner mask using Multi class Otsu
Thresholding
t = multithresh(in_imask, 3); % Into 3 classes
v_in = imquantize(in_imask, t);

% Mask class 3(A)
l3a_mask = zeros((size(l)));
l3a_vals = v_in == 2;
l3a_mask(l3a_vals) = 1;
final_mask(l3a_vals) = lindex;
lindex = lindex+1;

% Mask class 3(B)
l3b_mask = zeros((size(l)));
l3b_vals = v_in == 3;
l3b_mask(l3b_vals) = 1;
final_mask(l3b_vals) = lindex;
lindex = lindex+1;

% Mask class 3(C)
l3c_mask = zeros((size(l)));
l3c_vals = v_in == 4;
l3c_mask(l3c_vals) = 1;
l3c_vals = l3c_mask == 1;
final_mask(l3c_vals) = lindex;
lindex = lindex+1;

% Calculate metrics for each score
similarity = jaccard(categorical(l),
categorical(final_mask));
ssim_score = get_ssim_scores(l, final_mask);
dice_score = dice(categorical(l),
categorical(final_mask));
%Compute mean of all three calculated scores
mean_score = (similarity + dice_score + ssim_score)
/ 3;
mean_val = mean(mean_score);

% Plotting final result mask
figure();
volshow(final_mask);
figure();
volshow(l);
% Function to fill image along all the three
dimensions
function f3=imfill3d(f3)
    for i=1:3
        for j=1:size(f3,3)
            f3(:,:,j)=imfill(f3(:,:,j),'holes');
        end
    end
end
```

Code 4: get_ssim_scores.m

```
% Function to calculate ssim scores (Structural
Similarity)
function values=get_ssim_scores(grount_truth, l)
    values = zeros(6,1,'double');
    for i=0:5
        grount_truth_mask =
zeros((size(grount_truth)));
        grount_truth_vals = grount_truth == i;
        grount_truth_mask(grount_truth_vals) = 1;

        mask = zeros((size(grount_truth)));
        vals = l == i;
        mask(vals) = 1;
        [ssim_values, ~] = ssim(grount_truth_mask,
mask);
        values(i+1) = values(i+1) + ssim_values;
    end
end
```