Aim: We are given a set of 10 'axial' images of MRI of human brain subject, the aim is to segment out 5 tissue layers. There are 3 tasks we need to perform as follows:

- Task 1: Develop and apply different segmentation algorithms, for each slice of the MRI data.
- Task 2: Compare the segmented results for each algorithm to the ground-truth label, evaluate the metrics used to access accuracy and highlight the best algorithm to be used.
- Task 3: Implement a 3D segmentation algorithm to apply it to all slices simultaneously, discussing proposed algorithm is better/worse using evaluation metrics used above.

Methods:

Task 1: -

Active contour [3]: - It is an image segmenting algorithm also called snakes, which runs iteratively region growing, where we specify the initial curves for region of interest, and the algorithm evolves the curves towards object boundaries.
Multi Thresholding [4]: - A segmentation algorithm based on Otsu's method, that is used to separate pixels of image into different classes, according to their gray scale intensities. Performs well.
Image Segmenter App [2]: - This image processing tool gives multiple options to segment better, where we can generate custom function directly into main code. We have used the flood fill which is an automatic technique where initial points are specified, and it segments areas with similar intensities.

Experiment 1:- We first loaded "brain.mat" file which yields two variables even which is the actual images and label which has the ground truth. By looking at the original image we could see That we could separate the air, skull and the inner part which could be segmented easily using active contour method.

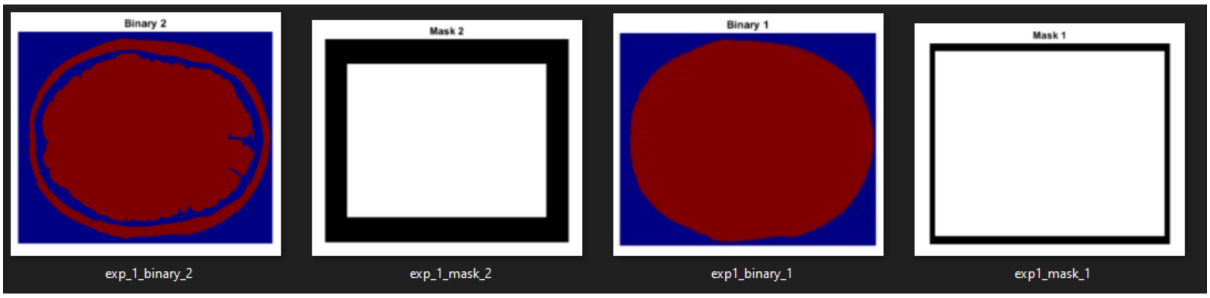
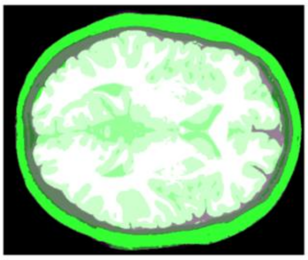


Figure 1: Masks and binaries obtained via Active Contour

This proved to be insufficient, and we decided to approach the problem in a new route.



```
similarity_dice = 5x1
    0.2132
         0
         0
         0
         0
```

Figure 2: Experiment 1 Results

Experiment 2: - We then investigated multi thresholding to get all the thresholding at once This proved to be a good result, but we wanted to improve it further.

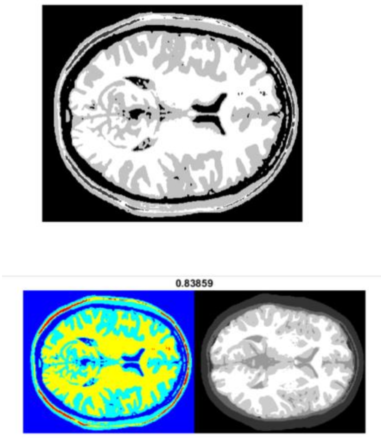


Figure 3: MultiThresh segmentation image and its dice results (0.83859)

<u>Experiment 3</u>: -So then finally we decided to use combinations of algorithms to manually segment each of the parts.

We had a look at the image histogram of one of the figures (image 9), where we observed 4 clear spikes.

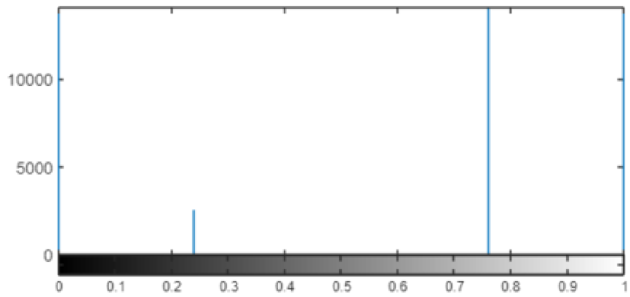


Figure 4: Histogram of Image 9 in Brain.mat

Then we begin isolating the intensities. clearly this was again not enough so then we applied active counter by setting the mask intensities and then to make it better we used image segmented tool provided in the MATLAB to make it clearer.

Then we decided to normalise the obtain slices on this grace core value by applying each intensity in multiples of 51 so that after 5 segments we obtain an image which ranges from zero to 255.

Figure 5 (Above): All five segments visualized

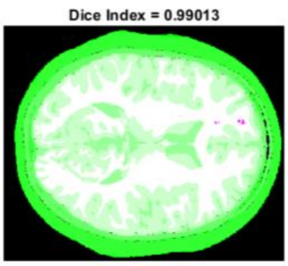


Figure 6 (Left): Final dice accuracy of 99.01% visualized

Task 2

Dice Similarity coefficient aka Dice Score [1] is a commonly used statistic to check similarity between two sample images. It is commonly used in image segmentation evaluation in medical fields. The third experiment yielded best results for the dice score.

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \quad DSC = \frac{2TP}{2TP + FP + FN}.$$

Figure 7: Dice score formulae

Task 3

From the MATLAB documentation of active contours, it was possible to apply to 3D images. within created empty 3D masks. We then used the existing final variables obtained from the 2D segmentation and pass them as masks to the active contour. Each counter was run for around 300 iterations. We finally plotted each segment to a 3D map. The final 3D image obtained by adding all the 5 binaries after performing segmentation. We had to manually adjust the alpha map of this image using the rendering editor and parameter settings as "maximum intensity projection" in the volume viewer app[5]. We then export this config to a file in the workspace and display it using the volshow function passing the final 3D image and config as parameters.

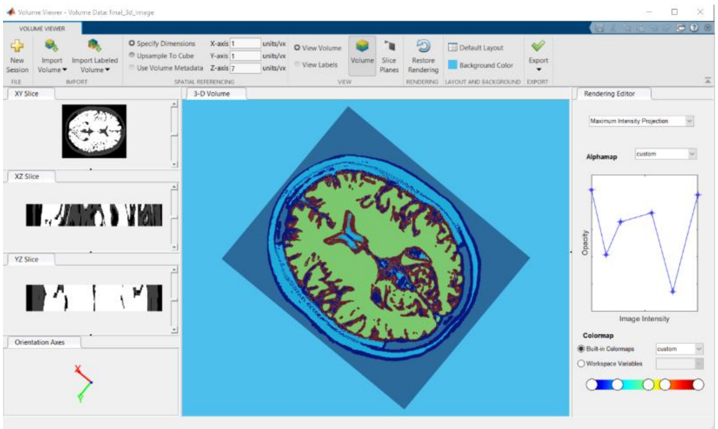


Figure 8: Final 3D segmented image after adjustments, viewed in Volume Viewer

Results:

<u>Task 1 & 2:</u>

For Experiment 1, After segmenting the air, skull, and the inner part, we decided to run the dice core just to be sure that the code works and can be further improved. the dice result was a disappointing 24% accuracy.
For Experiment 2, For MultiThresholding we, applied dice score and got an accuracy of 85%.
For Experiment 3, After the final 2D segmentation and adding all the segments and running it through the dice score, we finally got an accuracy of 99.01%.

<u>Task 3:</u> Then used the dice score to calculate its accuracy which came up to be 94.004% yielding good results.

Conclusion: To achieve our aims set for the tasks, no particular algorithm gave the best segmentation results. However, a combination of several algorithms and functions i.e., active contour, histogram, MultiThresh, flood fill image segmentation. This usually depends on the image data and ground truths provided. We see that our accuracy has decreased for 3D segmentation compared to 2D segmentation. This maybe due to few data samples provided.

References:

1. https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient
2. https://uk.mathworks.com/help/images/image-segmentation-using-the-image-segmenter-app.html
3. https://uk.mathworks.com/help/images/ref/activecontour.html
4. https://uk.mathworks.com/help/images/ref/multithresh.html
5. https://uk.mathworks.com/help/images/ref/volumeviewer-app.html

MATLAB Code used to process and obtain results

Task 1&2

Experiment 1 Code

```matlab
load("Brain.mat");
```

```matlab
figure();
for i = 1 : 1 : 10
    subplot(2,5,i)
    imagesc(label(:,:,i));
    title(['Image Number = ' num2str(i)]);
    %axis equal; axis tight;
    colormap jet
end
figure();

for j = 1 : 1 : 10
    subplot(2,5,j)
    imagesc(T1(:,:,j));
    title(['Image Number = ' num2str(j)]);
    %axis equal; axis tight;
    colormap gray
end
```

```matlab
image_1 = T1(:,:,1);
figure();
imshow(image_1,[])
title("Original")
```

```matlab
mask_1 = zeros(size(image_1));
mask_1(15:end-15, 10:end-10)=1;
bw_1=activecontour(image_1, mask_1,300);
figure();
imshow(mask_1,[]);
title("Mask 1")
figure();
imshow(bw_1,[], colormap=jet)
title("Binary 1")
```

```matlab
mask_2 = zeros(size(image_1));
mask_2(45:end-45, 40:end-40)=1;
bw_2=activecontour(image_1, mask_2,300);
figure();
imshow(mask_2,[]);
title("Mask 2")
figure();
imshow(bw_2,[], colormap=jet)
title("Binary 2")
```

Experiment 2 Code

```matlab
% Direct segmentation based on Multi_thresholding
load("Brain.mat")
[result_img,multi_dice_score,multi_test_variable] = Multi_Thres_Score(T1(:,:,9),label(:,:,9));
imshowpair(result_img,label(:,:,1),"montage");
title(score);
```

```matlab
%%
function [result_img,multi_dice_score,multi_test_variable] = Multi_Thres_Score(img,label)
    % Converts the matrix to a grayscale image
    image1_mat_gray = mat2gray(img);
    % Main Multi Thresholding
    main_threshold_img = multithresh(image1_mat_gray,3);
    % Quantize based on output values
    main_segmented_im = imquantize(image1_mat_gray,main_threshold_img);
    % Need to convert label image, into an RGB color image for the purpose of visualizing the labeled regions
    main_seg_rgb = label2rgb(main_segmented_im);
    % Convert RGB image to grayscale
    multi_test_variable = im2gray(main_seg_rgb);
    % Convert
    multi_test_variable = mat2gray(multi_test_variable);
    imshow(multi_test_variable);
    figure;
    %Convert to logical to make it compatible with Dice function
    label_1 = cast(label, "logical");
    result_img = cast(main_segmented_im,"logical");
    multi_dice_score = dice(result_img, label_1);
    result_img = main_seg_rgb;
end
%%
```

Experiment 3 Code

```
load ("Brain.mat")
```

```
for i = 1 : 1 : 10
    [result_img, multi_dice_score] = Final_segment_2D(T1(:,:,i), label(:,:,i))
    figure()
end
```

```
function [result_img, multi_dice_score] = Final_segment_2D(image, label)
        [result_img,multi_dice_score,multi_test_variable] = Multi_Thres_Score(image,label);
        imshowpair(result_img,label,"montage");
        title(multi_dice_score);
        % Histogram approach
        imhist(multi_test_variable);
        figure;

        %Create a filter for air region
        air_mask_filter = grayconnected(mat2gray(image),1,1);

        %% Skull + Grey and seperating Skull Label 2
        multi_test_variable1 = multi_test_variable >= 0.7 & multi_test_variable <= 0.8;
        imshow(multi_test_variable1);
        % Create a mask
        mask_1 = zeros(size(multi_test_variable1));
        % Define contour edges
        mask_1(43:end-45, 41:end-40)=1;
        % Active contour applied
        binary_weighted_1=activecontour(multi_test_variable, mask_1,300);
        % Calculate the complement
        binary_weighted_1 = imcomplement(binary_weighted_1);
        % Cast to double for next step
        binary_weighted_1 = cast(binary_weighted_1,"double");
        % Function from Image segmentor with Flood fill
        skull_variable = FloodFill_segmentImage_1(binary_weighted_1);
        imshow(multi_test_variable1);
        imshow(skull_variable);
        % Remove skull to get it segmented
        imshow(multi_test_variable1 - skull_variable);
        final_skull= multi_test_variable1 - skull_variable;
        %final_skull = final_skull > 0;
```

```
        % Normalising
        aa_1 = zeros(size(final_skull));
        aa_1(final_skull>0)=51;
        imshow(aa_1);
        imshow(skull_variable);
        title("Skull New 55 values");

        %% Skin Label 1
        multi_test_variable2 = multi_test_variable1 - skull_variable;
        imshow(multi_test_variable2);
        mask_2 = zeros(size(multi_test_variable2));
        % Selecting the co-ordinates from original image
        mask_2(77:end-77, 88:end-88)=1;
        % Active contour
        binary_weighted_2 = activecontour(multi_test_variable, mask_2, 300);
        binary_weighted_2 = imcomplement(binary_weighted_2);
        binary_weighted_2 = cast(binary_weighted_2,"double");
        % Flood Fill via Image segmenter
        skin_variable = FloodFill_segmentImage_1(binary_weighted_2);
        imshow(skin_variable);
        % Removing air and skull region
        imshow(skin_variable - air_mask_filter);
        final_skin = skin_variable - air_mask_filter -skull_variable;
        final_skin = final_skin > 0;
        aa_3 = zeros(size(final_skin));
        aa_3(final_skin>0)=153;
        imshow(aa_3);
        title("Skin New 153 values");

        %% CSF Label 3
        % Selecting the first visible peak from Histogram
        multi_test_variable4 = multi_test_variable1 >= 0.0 & multi_test_variable1 <= 0.1;
        imshow(multi_test_variable4);
        mask_3 = zeros(size(multi_test_variable4));
        % Selecting the co-ordinates from original image
        mask_3(106:end-106, 114:end-114)=1;
        % Active contour
        bw_3=activecontour(multi_test_variable4, mask_3, 300);
        bw_3 = imcomplement(bw_3);
        bw_3 = cast(bw_3,"double");
        csfvar = FloodFill_segmentImage_1(bw_3);
        imshow(csfvar);
```

```
% Finally removing grey, white matter, skin and skull to obtain CSF
imshow(multi_test_variable4 - csfvar);
final_csfvar = multi_test_variable4 - csfvar;
final_csfvar = final_csfvar > 0;
aa_5 = zeros(size(final_skin));
% Normalisation
aa_5(final_skin>0)=255;
imshow(final_csfvar);
imshow(aa_5);
title("CSF New 255 values");


%% Grey Matter label 4
grey_variable = FloodFill_segmentImage_2(binary_weighted_1);
grey_variable = multi_test_variable1 - skull_variable - grey_variable;
imshow(grey_variable);
mask_4 = zeros(size(multi_test_variable1));
mask_4(15:end-15, 10:end-10)=1;
bw_4=activecontour(multi_test_variable1, mask_4,300);
figure();
imshow(mask_4,[]);
title("Skull")
figure();
imshow(bw_4,[], colormap=jet)
title("Skull Binary")
imshow(bw_4);
title("Air New 0 Values")
bw_4 = cast(bw_4,'double');
skull_1=bw_4-grey_variable;
imshow(skull_1);
skull_1 = imcomplement(skull_1);
skull_1 = skull_1 - csfvar;
imshow(skull_1)
final_grey_matter = skull_1;
final_grey_matter = final_grey_matter > 0;
aa_2 = zeros(size(final_grey_matter));
aa_2(final_grey_matter>0)=102;
imshow(aa_2);
title("Grey Matter New 102 values");
```

```
%% White Matter Label 5
% Selecting the next peak from Histogram
multi_test_variable3 = multi_test_variable1 >= 0.9 & multi_test_variable1 <= 1;
imshow(multi_test_variable3);
% Removing skin and others
multi_test_variable3 = multi_test_variable3 - skin_variable;
imshow(multi_test_variable3);
final_white_matter = multi_test_variable3;
final_white_matter = final_white_matter >0;
aa_4 = zeros(size(final_white_matter));
% Normalisation
aa_4(final_white_matter>0)=204;
imshow(aa_4);
title("White Matter New 204 values");

%% Combining all Binaries
%ASd
final_image = (final_skin)+(final_skull)+(final_csfvar)+(final_grey_matter)+(final_white_matter);
%final_image = aa_1 + aa_2 + aa_3 + aa_4 + aa_5;
title("Segmented and Combined");
imshow(final_image);
label_new = cast(label, "logical");
final_image = cast(final_image,"logical");
similarity_dice_new = dice(final_image, label_new);
figure();
imshowpair(final_image, label);
title(['Dice Index = ' num2str(similarity_dice_new)]);

function [BW,maskedImage] = FloodFill_segmentImage_2(X)
%segmentImage Segment image using auto-generated code from imageSegmenter app
%  [BW,MASKEDIMAGE] = segmentImage(X) segments image X using auto-generated
%  code from the imageSegmenter app. The final segmentation is returned in
%  BW, and a masked image is returned in MASKEDIMAGE.

% Auto-generated by imageSegmenter app on 11-May-2022
%----------------------------------------------------


% Normalize input data to range in [0,1].
Xmin = min(X(:));
Xmax = max(X(:));
if isequal(Xmax,Xmin)
```

```
        if isequal(Xmax,Xmin)
            X = 0*X;
        else
            X = (X - Xmin) ./ (Xmax - Xmin);
        end

        % Create empty mask.
        BW = false(size(X,1),size(X,2));

        % Flood fill
        row = 30;
        column = 185;
        tolerance = 5.000000e-02;
        addedRegion = grayconnected(X, row, column, tolerance);
        BW = BW | addedRegion;

        % Create masked image.
        maskedImage = X;
        maskedImage(~BW) = 0;
        end

        function [BW,maskedImage] = FloodFill_segmentImage_1(X)
        %segmentImage Segment image using auto-generated code from imageSegmenter app
        %  [BW,MASKEDIMAGE] = segmentImage(X) segments image X using auto-generated
        %  code from the imageSegmenter app. The final segmentation is returned in
        %  BW, and a masked image is returned in MASKEDIMAGE.

        % Auto-generated by imageSegmenter app on 11-May-2022
        %----------------------------------------------------


        % Adjust data to span data range.
        X = imadjust(X);

        % Create empty mask.
        BW = false(size(X,1),size(X,2));

        % Flood fill
        row = 75;
        column = 87;
        tolerance = 5.000000e-02;
        addedRegion = grayconnected(X, row, column, tolerance);
        BW = BW | addedRegion;
```

```
        % Create masked image.
        maskedImage = X;
        maskedImage(~BW) = 0;
        end


        %%
        function [result_img,multi_dice_score,multi_test_variable] = Multi_Thres_Score(img,label)
            % Converts the matrix to a grayscale image
            image1_mat_gray = mat2gray(img);
            % Main Multi Thresholding
            main_threshold_img = multithresh(image1_mat_gray,3);
            % Quantize based on output values
            main_segmented_im = imquantize(image1_mat_gray,main_threshold_img);
            % Need to convert label image, into an RGB color image for the purpose of visualizing the labeled regions
            main_seg_rgb = label2rgb(main_segmented_im);
            % Convert RGB image to grayscale
            multi_test_variable = im2gray(main_seg_rgb);
            % Convert
            multi_test_variable = mat2gray(multi_test_variable);
            imshow(multi_test_variable);
            figure;
            %Convert to logical to make it compatible with Dice function
            label_1 = cast(label, "logical");
            result_img = cast(main_segmented_im,"logical");
            multi_dice_score = dice(result_img, label_1);
            result_img = main_seg_rgb;
        end
        %%
    end
```

Task 3

```
load ("Brain.mat")

%% Create empty masks
mask_11 = zeros(size(T1));
mask_12 = zeros(size(T1));
mask_13 = zeros(size(T1));
mask_14 = zeros(size(T1));
mask_15 = zeros(size(T1));

%% Mask values are assigned with previous 2D segmentaion binaries
mask_11(:,:,9) = final_skin;
mask_12(:,:,9) = final_skull;
mask_13(:,:,9) = final_csfvar;
mask_14(:,:,9) = final_grey_matter;
mask_15(:,:,9) = final_white_matter;

%% Applying 3D active contour to entire image set
bw_11 = activecontour(T1,mask_11,300);
bw_12 = activecontour(T1,mask_12,300);
bw_13 = activecontour(T1,mask_13,300);
bw_14 = activecontour(T1,mask_14,300);
bw_15 = activecontour(T1,mask_15,300);

% Display the Skin 3D segment
figure;
p_1 = patch(isosurface(double(bw_11)));
p_1.FaceColor = 'red';
p_1.EdgeColor = 'none';
daspect([1 1 27/128]);
camlight;
lighting phong

% Display Skull 3D Segment
figure;
p_2 = patch(isosurface(double(bw_12)));
p_2.FaceColor = 'blue';
p_2.EdgeColor = 'none';
daspect([1 1 27/128]);
camlight;
lighting phong

% Display CSF 3D Segment
figure;
p_3 = patch(isosurface(double(bw_13)));
p_3.FaceColor = 'green';
p_3.EdgeColor = 'none';
daspect([1 1 27/128]);
camlight;
lighting phong

% Display Grey Matter 3D Segment
figure;
p_4 = patch(isosurface(double(bw_14)));
p_4.FaceColor = 'yellow';
p_4.EdgeColor = 'none';
daspect([1 1 27/128]);
camlight;
lighting phong

% Display White Matter 3D Segment
figure;
p_5 = patch(isosurface(double(bw_15)));
p_5.FaceColor = 'black';
p_5.EdgeColor = 'none';
daspect([1 1 27/128]);
camlight;
lighting phong

% Adding final binaries
final_3d_image = bw_11+bw_12+bw_13+bw_14+bw_15;

% Calculating final 3D Dice score
final_3d_image_1 = cast(final_3d_image,'logical');
final_3d_groundTruth = cast(label, 'logical');
sim_dice_3d = dice(final_3d_image_1, final_3d_groundTruth);
sim_dice_3d

% 3D image display from Volume viewer
volshow(final_3d_image, config1);
```