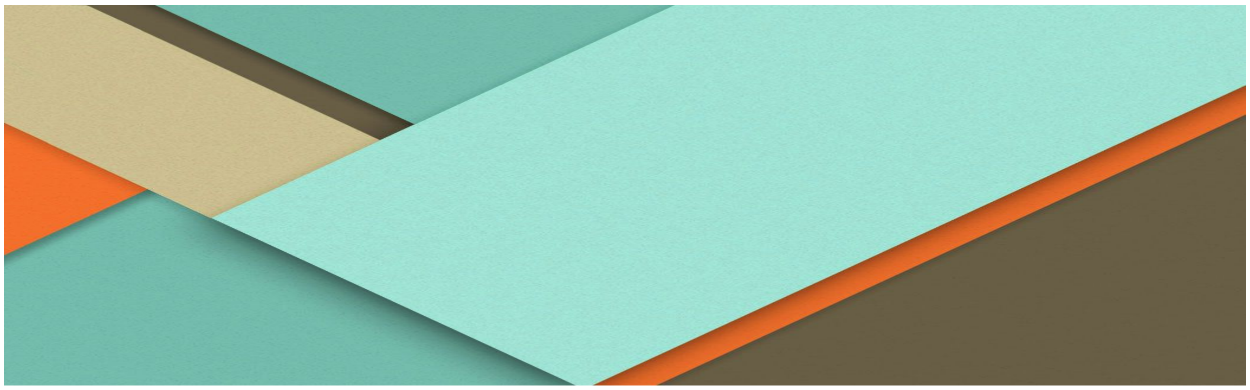

ECE 6782 Image Processing Experiment - 1

Group ORANGE

Jie Wang (jw4hr)

Akshat Verma (av2zf)

Damien Labier (dal2we)

Goal :

To read Tiff stack, process individual frames, separate in color, label and count neuroblasts and nuclei and estimate the surface area of membranes

Methodology and Results:

There are three broad steps in our approach:

1. We read individual frames of the stack one by one , color segmented them using k-means clustering and created three different stacks for red, blue and green respectively.
2. We read segmented stacks separately frame by frame, performed morphological operations and then created 3d image from the individual processed frames. In the 3d image we counted connected components using “bwlabeln”.
3. We estimated the surface area with two methods using the morphological skeleton of the membrane.

The different steps and the outputs produced are discussed in details below, using SampleStack.tif:

1. Reading Stack and colour segmentation

In order to deal with individual frames of the complete stack we read the stack frame by frame using a loop . Inside the loop, colour segmentation is realized by four steps. The basic idea for the first three steps have been taken from [2] and developed further.

i) Converting image from RGB Color Space to L*a*b* Color Space by using makecform and applycform. The idea for using L*a*b* Color Space is that this color space can predict which spectral power distributions will be perceived as the same color. For example, similar red will all be classified to red space. Command ‘rgb2lab(rgb)’ just helps us to convert RGB values to CIE 1976 L*a*b* values.

ii) Classifying the colors in 'a*b*' Space using K-means Clustering with Euclidean distance metric. Clustering is a way to separate groups of objects; and K-means clustering is a method of vector quantization which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

iii) Labelling every pixel in the image using the results from K-means with its cluster_index.

iv) Creating images that segment the image by color by using pixel_label. Here, we observed that the cluster indices returned by k-means clustering are not consistent across different frames. So, to identify whether the segmented images belonged to the red, green or blue cluster, we attempt to detect a majority in the RGB pixels population. For each cluster, pixels above an intensity threshold and with highly differing RGB intensities are scanned. The brightest color is identified for each pixel and stored in a map, from which the most recurring color is assumed to be the image's color. However, due to high noise in some segments, this can lead to false attributions. Hence in the case of double attributions, another method is applied. The latter consists in counting the pixels above the previous threshold and RGB range for each color. The largest population will be attributed to the corresponding color and image. This is iterated for all images and enables an accurate identification with the exception of about 5% of outliers, which were manually corrected.

2. Labelling and counting neuroblasts and nuclei

We read the segmented blue and red stacks separately frame by frame and performed the following operations in sequence in the loop:

i) Cropping the image with a rectangular dimension using "imcrop" to get rid of the other cells

ii) Extracting corresponding channels for red and blue in the RGB image i.e. I(:,:,1) for red and I(:,:,3) for blue

iii) Applying median filter on the extracted image matrix to remove noise as far as possible

iii) Thresholding the image based on Otsu's algorithm

iv) Using "imfill" to fill any holes

v) Performing morphological operation open using "imopen" with a disk of radius 3 as the structural element

vi) Still there seemed to be a lot of noise especially in the blue stack, so filtered out the connected components having area greater than a experimentally determined number as they constituted the noise.

Finally, all the processed images were stacked to form a 3D-image and we used "bwlabeln" to count the number of connected components in 3D image.

Visualisation of labels was done individually in all the frames of the stack, by overlaying the area obtained after passing the labelled matrix output from bwlabel to regionprops with the corresponding labels. “NB” label was used for neuroblasts and “N” label was used for nuclei

Results:

Count of neuroblasts (red) : **116**

Count of nuclei (blue): **96**

The attached videos show the frames labelled with location of nuclei and neuroblasts.

A couple of sample labelled images are shown below (the video contains all the frames):

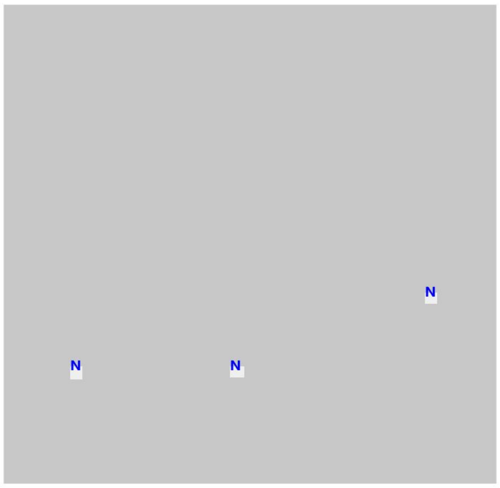

Figure 1 : Sample labelled output of nuclei

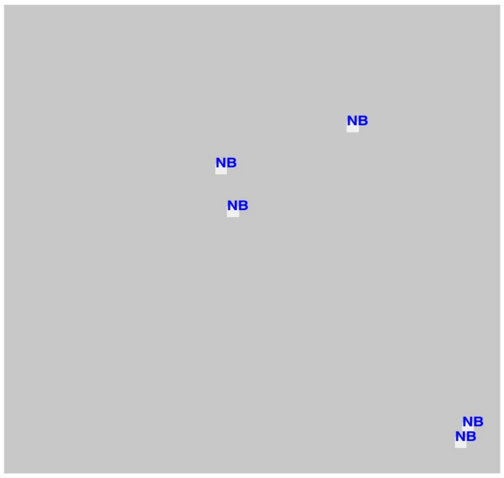

Figure 2 : Sample labelled output of neuroblasts

3. Estimate the surface area of glia membranes :

A. We start by extracting the green intensities of the frame and binarize it at an intensity level of 50.
B. We then perform a skeletonization on the image to remove pixels on the boundaries of the membrane without allowing it to break apart. Isolated pixels are cleaned from the image. The resulting image is shown below.

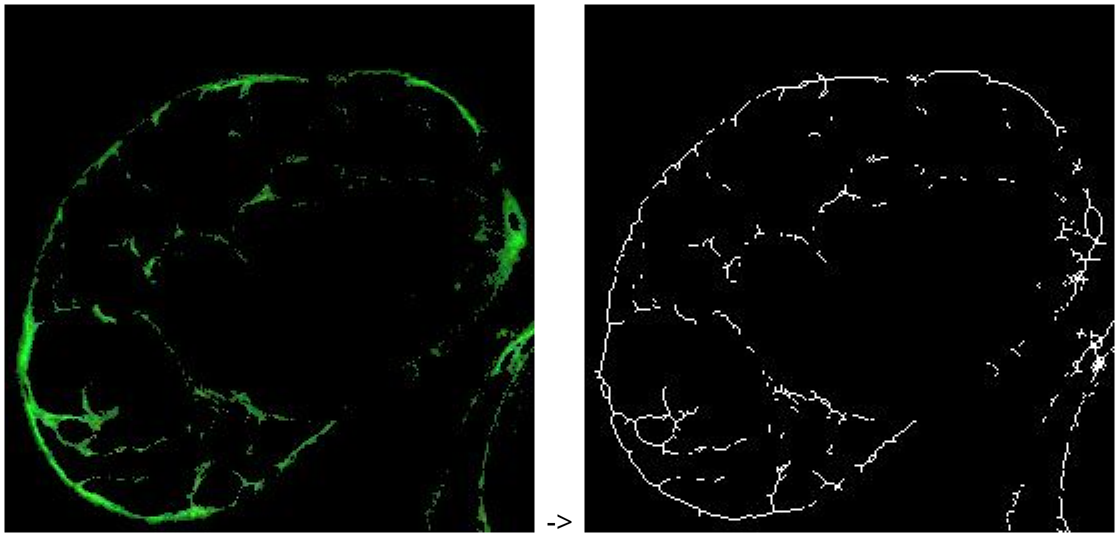

Figure 3: output of the skeletonization process - Image 45

C. Since the unit length of a pixel is the real FOV size over the image size, i.e. 200 um/265 px and the curvature radii are usually much larger than a pixel, we can approximate the perimeter of the membrane by counting the number of pixels forming that skeleton and multiplying it by the image resolution. Finally, we can approximate the surface area by multiplying the perimeter by the height difference between slices, i.e. 1 um. This provides an estimate of .177 mm^2.
D. Another method is to link each layers with a triangulated surface, before computing the area of the resulting mesh. To ease computation, the original image is scaled down to 200x200px, resulting in a resolution of 1um/px. Here the method of [1] is used. A visualization of the volume is shown below.

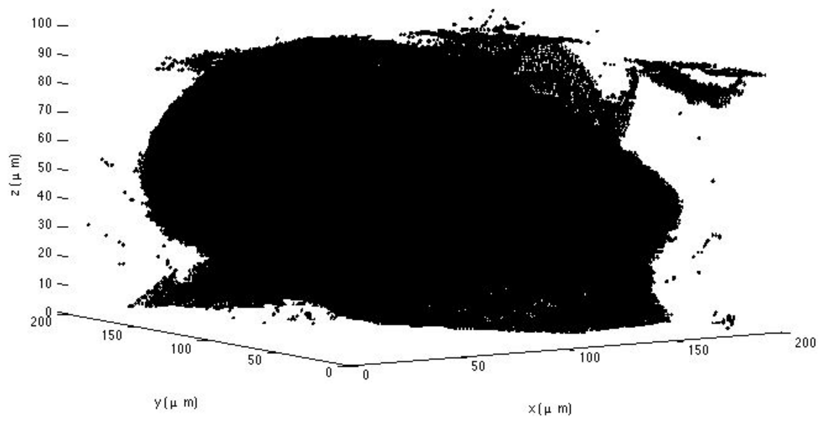

Figure 4: Output of the triangulation process

This method resulted in an estimate of a total area of .412 mm^2.

Conclusion: both methods estimated a glia membranes surface area of the order of .1 mm^2. Note: these frames included elements outside the main volume of interest, which were estimated to need a manual removal.

[1]http://stackoverflow.com/questions/25747918/how-to-calculate-the-area-by-triangulating-a-3d-object-in-matlab

[2]http://www.mathworks.com/help/images/examples/color-based-segmentation-using-k-means-clustering.html

Code

Segmentation (segmentation.m)

```
%%%% ECE 6782 Experiment 1
%%%% Jie Wang jw4hr, Akshat Verma av2zf, Damien Lieber dal2we

function segmentation(img,img_dir)
close all
clear all

%% Obtains images path and count, saving directory
data_dir= img_dir;
fname = img;
info = imfinfo(fname);
num_images = numel(info);
```

```
save_dir=sprintf('output\\%s',data_dir);
if (exist(save_dir)~=7)
    mkdir(save_dir);
    mkdir(sprintf('%s\\red',save_dir));
    mkdir(sprintf('%s\\green',save_dir));
    mkdir(sprintf('%s\\blue',save_dir));
end

%% Defines C form and initiates variables
cform = makecform('srgb2lab');
nColors = 3;%# of colors
occurences=zeros(nColors,nColors);

%% Loop through all images
for k = 24:num_images

    clearvars -except fname k cform nColors occurences save_dir num_images
    display(sprintf('frame %d of %d',k,num_images));

    %Reads image
    A = imread(fname, k);

    %Applies cform to image
    lab_A = applycform(A,cform);
    ab = double(lab_A(:,:,2:3));
    nrows = size(ab,1);
    ncols = size(ab,2);
    ab = reshape(ab,nrows*ncols,2);


    %Clustering; repeats it 5 times to avoid local minima
    [cluster_idx, cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean', 'Replicates',5);
    pixel_labels = reshape(cluster_idx,nrows,ncols);%Reshapes into image-form
    rgb_label = repmat(pixel_labels,[1 1 3]);

    %% Loops through color clusters
    for i = 1:nColors

        %Deletes all pixels out of the cluster
        color_img = A;
        color_img(rgb_label ~= i) = 0;

        indices_mat=zeros(size(A,1),size(A,2));

        %Threshold is define at mu-2sigma in total intensity of non-zero
        %pixels
        threshold=mean(mean(nonzeros(sum(double(color_img(:,:,:)),3))))-std(std(nonzeros(sum(double(color_img(:,:,:)),3))))*2;

        %Scans each pixel to find maximum RGB brightness for pixels above
        %threshold and where RGB ratio is higher than 2
        for ii=1:size(color_img,1)
            for jj=1:size(color_img,2)
                tmp=find((color_img(ii,jj,:)==max(color_img(ii,jj,:)) & sum(double(color_img(ii,jj,:)))>threshold &
min(double(color_img(ii,jj,:)))/max(double(color_img(ii,jj,:)))<.5),1);
                if isempty(tmp)==0
                    indices_mat(ii,jj)=tmp;
            end
         end
      end

      %Stores the most recurrent brightest color in  array rgb_index, stores
      %number of reccurences for each color for each cluster in array
      %occurences
      rgb_index(i)=mode(mode(indices_mat(find(indices_mat(:,:)~=0))));
      for j=1:nColors
         occurences(i,j)=size(find(indices_mat==j),1);
      end
   end

   %% Detects if the array rgb_index has repeated values
   trigger=false;
   for i=1:nColors
      for ii=1:nColors
         if i~=ii && rgb_index(i)==rgb_index(ii)
            trigger=true;
         end
      end
   end

   %If so, attributes the frame with the sharpest difference between color populations to
   %its brightest color, and iteratively.
   if trigger==true
      display(occurences);
      occurences=occurences+1;
      for i=1:nColors
         [tmp,frame_index]=max(max(occurences,[],2)./mean(occurences,2));%
         [tmp,color_index]=max(occurences(frame_index,:));
         rgb_index(frame_index)=color_index;
         occurences(frame_index,:)=[-1 -1 -1];
         occurences(:,color_index)=[-1; -1; -1];

      end
      display(rgb_index);
   end

   %% Sorts and saves images in folders
   for i=1:nColors
      color_img = A;
      color_img(rgb_label ~= i) = 0;
      rgb_str={'red','green','blue'};
      imwrite(color_img, sprintf('%s\\%s.tif',save_dir,char(rgb_str(rgb_index(i)))),'tif',’writemode’,’append’);
   end
end
```

Count and Label nuclei and neuroblasts (labelcount.m and vislabel.m)

```
% Function to label and count nuclei and nseuroblasts

function labelcount(img,outdir)
```

```
% Create output directory if it doesn't exist
if (exist(out_dir)~=7)
   mkdir(out_dir);

fname = img;
info = imfinfo(fname);
num_images = numel(info);

% Preallocate Image Matrix to store 3D image from cropped 2D slices
[x,y,z] = size(imcrop(imread(fname,1),[28.5 29.5 206 220]));
allImgs = uint8(zeros(x, y, num_images));

% Define a structural element disk of radius 3
se_disk = strel('disk',3);



% Read the input stack frame by frame
for k = 1:num_images
   A = imread(fname,k);
   if isempty(strfind(fname, 'blue')) % if the stack is not blue i.e red
      A = A(:,:,1); % Extract red channel from the image
      label = 'NB'; % Define the label "NB" for neuroblasts
   else
      A = A(:,:,3); % Extract blue channel from the image
      label = 'N';  % Define the label "N" for nuclei
   end

   A = imcrop(A,[28.5 29.5 206 220]) % Cropping the iage to get rid of other cell parts
   A = medfilt2(A); % Applying median filter
   A = im2bw(A,graythresh(A)); % Thresholding using graythresh
   A = imfill(A,'holes'); % fill holes
   A = imopen(A,se_disk); % open operation using disk as structural element

   %  Finding connected components which are greater than 30 and removing
   %  them, as they constitute noise
   CC = bwconncomp(A,8);
   S = regionprops(CC, 'Area');
   L = labelmatrix(CC);
   A = ismember(L, find([S.Area] <= 30));

   % Visualizing and printing the labelled 2D image, it is just for the
```

```
    % purpose of visualisation, actual count is calculated on the 3D image
    % using bwlabeln below
    [L2,N] = bwlabel(A);
    vislabels(L2,label);
    print(strcat(outdir,'//',num2str(k),'.png'),'-dpng')

    % Creating 3D image from the 2D slices by stacking them
    allImgs(:,:,k) = A;
end

% Calling bwlabeln on 3D image and get the count
[L, NUM] = bwlabeln(allImgs)
function vislabels(L,label)
%   VISLABELS Vi=sualize labels of connected components
%   Originally written by Steven L. Eddins
%   Copyright 2008 The MathWorks, Inc
%   Modified by us for this experiment

% Form a grayscale image such that both the background and the
% object pixels are light shades of gray.  This is done so that the
% black text will be visible against both background and foreground
% pixels.

background_shade = 200;
foreground_shade = 240;
I = zeros(size(L), 'uint8');
I(L == 0) = background_shade;
I(L ~= 0) = foreground_shade;

% Display the image, fitting it to the size of the figure.
imageHandle = imshow(I, 'InitialMagnification', 'fit');

% Get the axes handle containing the image.  Use this handle in the
% remaining code instead of relying on gca.
axesHandle = ancestor(imageHandle, 'axes');

% Get the extrema points for each labeled object.
s = regionprops(L, 'Extrema');

% Superimpose the text label at the left-most top extremum location
% for each object.  Turn clipping on so that the text doesn't
% display past the edge of the image when zooming.

hold(axesHandle, 'on');

for k = 1:numel(s)

   e = s(k).Extrema;

   text(e(1,1), e(1,2), sprintf('%s', label), ...

      'Parent', axesHandle, ...

      'Clipping', 'on', ...

      'Color', 'b', ...

      'FontWeight', 'bold');

end

hold(axesHandle, 'off');
 `
```

Glia membrane surface area computation

```
%% Import parameters of sorted clusters
data_dir='output/green_2/';
res_xy=2.7834;%um
dz=1;%um

listOfTiffs = dir(sprintf('%s*.tif',data_dir));
    numberOfTiffs = numel(listOfTiffs);

C=zeros(200,200,95);

%% Loop through green clusters
for i=1:95%numberOfTiffs
    A = imread(sprintf('%s%02d.tif',data_dir,i));%Imports image
    A=A(:,:,2);%Keeps only green pixels
    A=imresize(A,[200 200]);%Resizes image to fit FOV size of 200um at 1um/px

    %% Morphological operations: skeletonization + isolated pixels cleaning
    B = bwmorph(A,'skel',Inf);
    C(:,:,i) = bwmorph(B,'clean');

    %% Saves images in stack
    if i==1
        imwrite(C(:,:,i), sprintf('output/test3.tif'),'tif');
    else
            imwrite(C(:,:,i), sprintf('output/test3.tif'),'tif','writemode','append');
    end

    %% Approximates perimeter of frame i by counting pixelss
    perimeter(i)=size(nonzeros(C(:,:,i)),1);%*res_xy/size(C(:,:,i),1);%um
end

%% Approximates membrane area by integrating each pixel over the vertical inter-frame distance dz
membrane_area=sum(perimeter)*(dz-1);


%% Method 2 - mesh plot
% Creates an isosurface for all pixels=1
FV = isosurface(C, 0.5);
hiso = patch(FV);
V = FV.vertices;%Extracts vertices info
F = FV.faces;%Extracts faces info
%Computes triangles edges vectors
AA = V(F(:, 2), :) - V(F(:, 1), :);
```

```
BB = V(F(:, 3), :) - V(F(:, 1), :);
CC = cross(AA, BB, 2);
%Computes membrane area by summing the areas of triangles
membrane_area2 = 1/2 * sum(sqrt(sum(CC.^2, 2)));
```