12212859 黎思婕

Part 1

Task: By quantifying the targt protein content in axons and myelin, we can know **where the POI express**.

I first **determine the domain** of axons nad myelin respectively and create a binary mask. Then administrate the **mask** to the protein image by multiply (0 removed and 1 maintain the original values) and **sum** the values of pixels after masking. Also, to eliminate the effect of the original abundance of axons and myelin, the result is normalized by areas of axons and myelin.

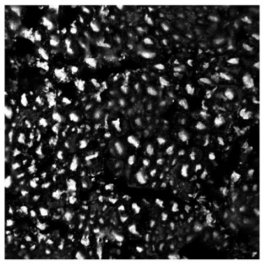
In this process, the abundance of protein is quantify by the **area (number of pixel)** and **intensity (value of pixel).**

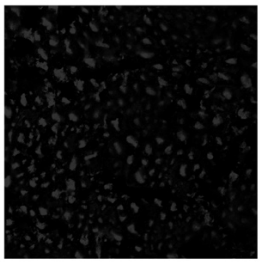
Decompose

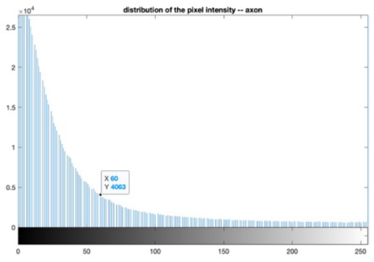
To better handle the images, I tried to understand the **composition** of them at first.

```
clc,clear;
axon_img = imread('green-axon.tif');
axonR = axon_img(:,:,1);
imshow(axonR);
axonG = axon_img(:,:,2);
imshow(axonG);
axonB = axon_img(:,:,3);
imshow(axonB);
axon = axonB + axonG + axonR;
figure;
imhist(axon);title('distribution of the pixel intensity -- axon')
% take lower threshold for the grey should be counted due to the lower
intensity
```

axonR


axonG


axonB

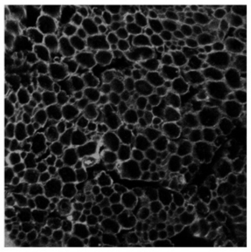


Axon_hist

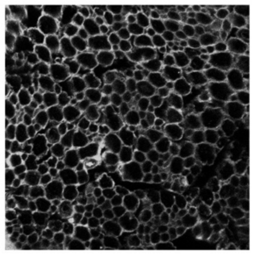
Same for the myelin.

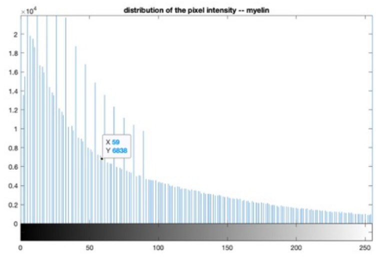
```
myelin_img = imread('purple-myelin.tif');
myelinR = myelin_img(:,:,1);
imshow(myelinR);
myelinG = myelin_img(:,:,2);
imshow(myelinG);
myelinB = myelin_img(:,:,3);
```

```
imshow(myelinB);
myelin = myelinR + myelinG + myelinB;
imhist(myelin);title('distribution of the pixel intensity -- myelin')
```


myelinR

myelinG


myelinB

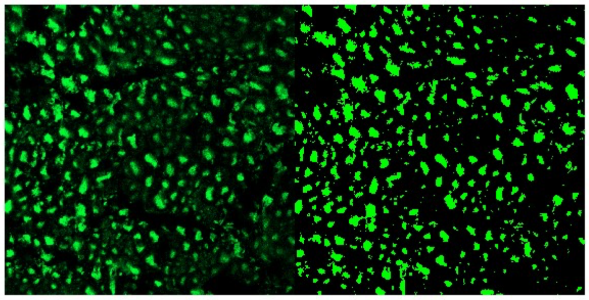


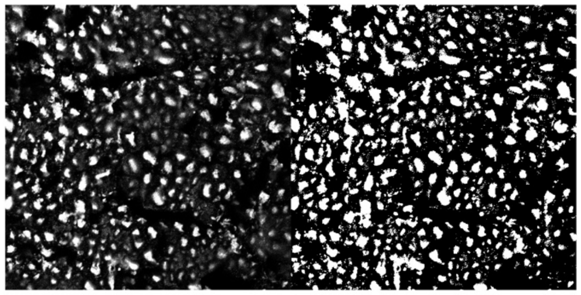
Myelin_hist

Create Binary masking

A **threshold** should be assigned to determine the domain. Here from the histogram above, I choose 60 (axon) and 59 (myelin) and compare with the default Otsu's method.

```
bi_axon_60 = imbinarize(axon,60/255);
figure;
imshowpair(axon,bi_axon_60,'montage');
bi_axon_default = imbinarize(axon_img);
figure;
imshowpair(axon_img,sum(bi_axon_default,3),'montage');
```

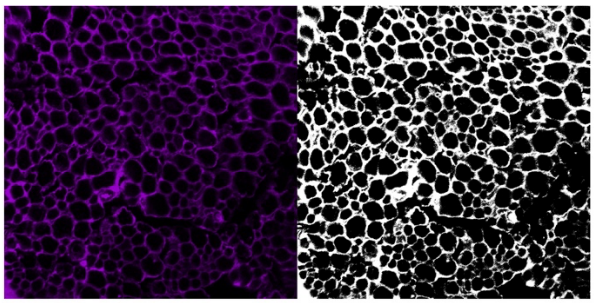

Axon_default (left origin, right binarized)

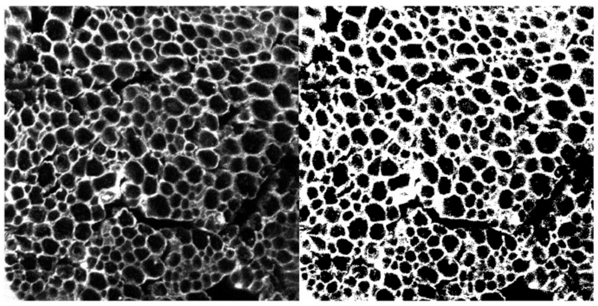

Axon_60_threshold (left origin, right binarized

From the result, we can see with lower threshold, axon can be caught better (there are some vague axons at the peripheral neglected by the default binarization method.)

Same for the myelin.

```
bi_myelin_59 = imbinarize(myelin,59/255);
figure;
imshowpair(myelin,bi_myelin_59,'montage');
bi_myelin_default = imbinarize(myelin_img);
figure;
imshowpair(myelin_img,sum(bi_myelin_default,3),'montage');
```

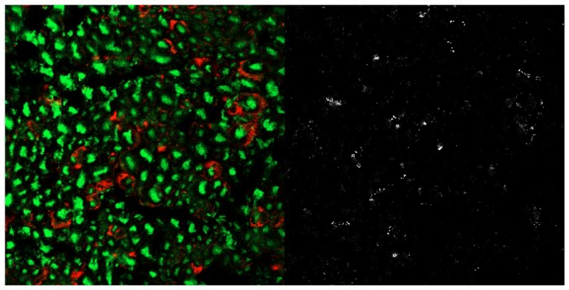

myelin_default (left origin, right binarized)

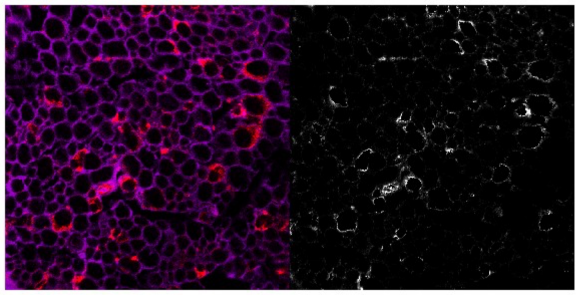

myelin_59_threshold (left origin, right binarized

It seems that the default one is better.

Masking

```
protein_img = imread("red-target.tif");
proteinR = protein_img(:,:,1);
imshow(proteinR);
proteinG = protein_img(:,:,2);
imshow(proteinG);
proteinB = protein_img(:,:,3);
imshow(proteinB);
protein = proteinR + proteinG + proteinB;
imshow(protein);
protein_in_axon = double(protein) .* bi_axon_60; % multiply to realize
masking!
figure;
imshowpair(protein_img+axon_img,protein_in_axon,'montage');
protein_in_myelin = double(protein) .* sum(bi_myelin_default,3);
figure;
imshowpair(protein_img+myelin_img,protein_in_myelin,'montage');
```

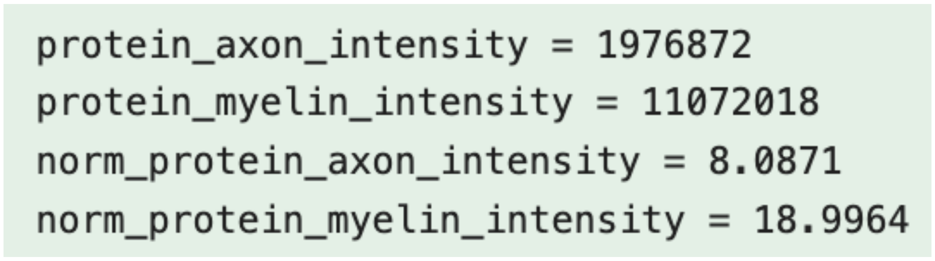

Protein in axons


Protein in myelin

Quantifying

```
protein_axon_intensity = sum(protein_in_axon(:)) % add up the intensity
after masking
protein_myelin_intensity = sum(protein_in_myelin(:))
norm_protein_axon_intensity = protein_axon_intensity/sum(bi_axon_60(:)) %
sum(bi_axon_60(:)) is the area of axon
```

```
bi_myelin_default = sum(bi_myelin_default,3);
norm_protein_myelin_intensity =
protein_myelin_intensity/sum(bi_myelin_default(:))
```



```
protein_axon_intensity = 1976872
protein_myelin_intensity = 11072018
norm_protein_axon_intensity = 8.0871
norm_protein_myelin_intensity = 18.9964
```

The result is in line with intuition. There are **much more target protein in myelin than in axons,** which means POI expressed more in the cell myelin.

Part 2

Step 1: register each slice repectively

Motivation

We want to take multiple images of one slice and take the average to remove the random noise generating during imaging. However, these **multiple images should be transformed into the same coordinate system before caluculating the average**. So, registration is first taken.

bUnwarpJ

The deformation of microglia is **subtle** and **elastic** within slice, so I choose **bUnwarpJ** plugin for its good performance in handling nonlinear deformation and fine structure.

I first use Fiji GUI and **macro record** to take down the parameters I used.

Macro script

```
run("bUnwarpJ", "source_image=XYTZ_512_512_300um_0.51s_045350_Z36UM_2.bmp
target_image=XYTZ_512_512_300um_0.51s_045350_Z36UM_1.bmp registration=Mono
image_subsample_factor=0 initial_deformation=[Very Coarse]
final_deformation=Fine divergence_weight=0 curl_weight=0 landmark_weight=0
image_weight=1 consistency_weight=10 stop_threshold=0.01");
```

*Then using **MATLAB** to process (for further automatic processing).*

- Initialize Fiji in MATLAB using `javaaddpath`, `addpath`, and `Miji`
- Use for loop to process each image
- Convert macro script with MIJ.run()
- Save the registered result as .bmp

(The whole script for MATLAB is attached below)

Step 2: average to improve SNR

Motivation

Noise is random, thus can be largely lower by averaging among multiple imaging.

Process

- Create an matrix with the image size sum_img = zeros(size(size_img), 'double');
- Add up the pixel intensity of all the images sum_img = sum_img + im2double(imread(img_path));
- Divided by the image number of the corresponding slice avg_img = sum_img / length(file_list)

(The whole script for MATLAB is attached below)

Step 3: Z projection

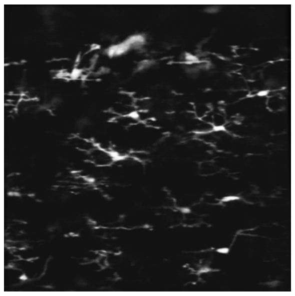
Motivation

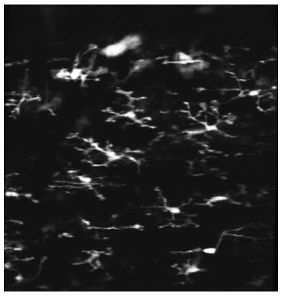
Can observe 3D information in 2D image. Sometimes, through projection, we can highlight key structure and reduce noise.

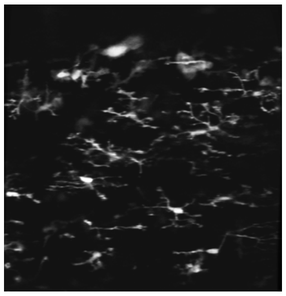
Process

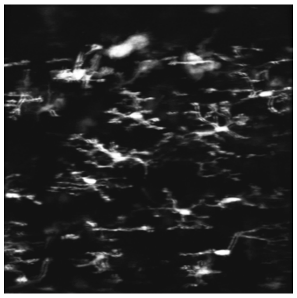
Take the maximun pixel value from the 3 slice and combine them together max_projection = max(stack, [], 3);

Result


Slice1_avg.bmp


Slice2_avg.bmp


Slice3_avg.bmp


Max_Z_Projection.bmp

It is obvious that due to different depth, there is a **slight shift between different slice**. Thus, the projection appears shifting effect.

Step 4: automize the whole precess

Input

the **slice number** (here 3), **the path of the directory** containing the slice folder (with .bmp to be process inside), and the path Fiji/scripts.

Output

A directory containing the average and registered result (in their folder repectively).

Setting

mij.jar and ij.jar file should be parallel to the input directory

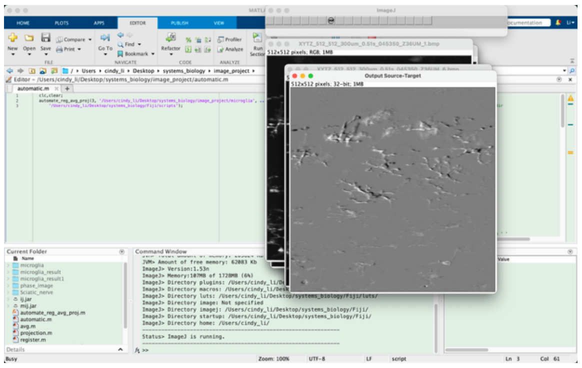
running script

```
clc,clear;
automate_reg_avg_proj(3,
'/Users/cindy_li/Desktop/systems_biology/image_project/microglia', ...
    '/Users/cindy_li/Desktop/systems_biology/Fiji/scripts');
```

function script

```
function automate_reg_avg_proj(slice_number, input_dir,fiji_script_dir)
    % input: slice number and directory wrap slice directory
    % file setting: mij.jar and ij.jar should be at the same level as
input_dir
    % output a directory contain registered, average, and projection
result
    % should be run in the directory containing this function


    % initialize path
    [parent_dir, content] = fileparts(input_dir);
    output_dir = fullfile(parent_dir, [content '_result1']);
    registered_dir = fullfile(output_dir, 'registered');
    average_dir = fullfile(output_dir, 'average');
    mkdir(registered_dir);
    mkdir(average_dir);


    % initialize Fiji
    javaaddpath(fullfile(parent_dir, 'mij.jar')); % must have
    javaaddpath(fullfile(parent_dir, 'ij.jar'));
    addpath(fiji_script_dir); % for Fiji, orlese ImageJ only
    Miji;


    % step 1: registration
    for slice_num = 1:slice_number
        slice_path = fullfile(input_dir, sprintf('slice%d', slice_num));
        image_files = dir(fullfile(slice_path, '*.bmp'));


        % set the first image to the reference
        target_img = fullfile(slice_path, image_files(1).name);
        MIJ.run('Open...', ['path=' target_img]); % variable should not be
in ' '


        for i = 2:numel(image_files)
            % open source image and select objects
            source_img = fullfile(slice_path, image_files(i).name);
            MIJ.run('Open...', ['path=' source_img]);
            MIJ.selectWindow(image_files(1).name);
            MIJ.selectWindow(image_files(i).name);


            % I get the parameter from Fiji GUI using macro record
function
```

```
            % Mono for I only need the registered source
            % Mind: there should be a blank after every parameters
            cmd = ['source_image=' image_files(i).name ' target_image='
image_files(1).name ' registration=Mono '];
            cmd = [cmd, 'image_subsample_factor=0
initial_deformation=[Very Coarse] '];
            cmd = [cmd, 'final_deformation=Fine divergence_weight=0
curl_weight=0 '];
            cmd = [cmd, 'landmark_weight=0 image_weight=1
consistency_weight=10 stop_threshold=0.01'];
            MIJ.run('bUnwarpJ', cmd);


            % save the registered result
            MIJ.selectWindow('Registered Source Image');
            output = fullfile(registered_dir, sprintf('Slice%d_%d.bmp',
slice_num, i));
            MIJ.run('Save', ['save=' output]);
            MIJ.run('Close', image_files(i).name);
        end
        MIJ.selectWindow(image_files(1).name);
        output = fullfile(registered_dir, sprintf('Slice%d_1.bmp',
slice_num));
        MIJ.run('Save', ['save=' output]); % this willl change the name of
the window, should be at the end
        MIJ.run('Close', image_files(1).name);
    end
    MIJ.run('Close All');
    MIJ.exit;


    % Step 2: average
    for slice_num = 1:slice_number
        file_pattern = sprintf('Slice%d_*.bmp', slice_num);
        file_list = dir(fullfile(registered_dir, file_pattern));
        % initialize the size of summation matrix
        first_img = im2double(imread(fullfile(registered_dir,
file_list(1).name)));
        sum_img = zeros(size(first_img), 'double');

        for i = 1:length(file_list)
            img_path = fullfile(registered_dir, file_list(i).name);
            sum_img = sum_img + im2double(imread(img_path));
        end
        avg_img = sum_img / length(file_list);
        imwrite(avg_img, fullfile(average_dir, sprintf('Slice%d_avg.bmp',
slice_num)));
    end
    % Step 3: projection (take max intensity of every pixel from the
slices)
    avg_images = cell(1, slice_number);
    for i = 1:slice_number
        avg_images{i} = im2double(imread(fullfile(average_dir,
sprintf('Slice%d_avg.bmp', i))));
    end
    stack = cat(3, avg_images{:});
    max_projection = max(stack, [], 3);
    imwrite(max_projection, fullfile(average_dir,
'Max_Z_Projection.bmp'));
    disp('Done! You can view the result in the "result" directory.');
end
```



```
clc,clear;
automate_reg_avg_proj(3, '/Users/cindy_li/Desktop/systems_biology/image_project/microglia', ...
    '/Users/cindy_li/Desktop/systems_biology/Fiji/scripts');
```

```
JVM> Amount of free memory: 62083 Kb
ImageJ> Version:1.53n
ImageJ> Memory:107MB of 1728MB (6%)
ImageJ> Directory plugins: /Users/cindy_li/De
ImageJ> Directory macros: /Users/cindy_li/Desk
ImageJ> Directory luts: /Users/cindy_li/Desktop/systems_biology/Fiji/luts/
ImageJ> Directory image: Not specified
ImageJ> Directory imagej: /Users/cindy_li/Desktop/systems_biology/Fiji/
ImageJ> Directory startup: /Users/cindy_li/Desktop/systems_biology/Fiji/
ImageJ> Directory home: /Users/cindy_li/
--------------------------------------------------------------
Status> ImageJ is running.
--------------------------------------------------------------
>>
```

Running the script and you can see the continuously popping out window showing the registration process. Just wait for 2 minutes or so and you will get the registration, average, and projection result in corresponding folders.

Harvest

- Image is **not ideal** due to diffraction, artecfect, etc.! The intensity of pixel is continuous so it is hard to tell the **exact edge** of the substances we are interested in. Commonly, a threshold should be assigned. Thus, it is hard to obtain **precise quantitive** information form image.
- Refer to documentation when confusing.
    - https://imagej.net/plugins/bunwarpj/ documentation for registration plugin--bUnwarpJ
    - https://imagej.net/plugins/miji tutorial to access Fiji instead of the legacy ImageJ (can not use bUnwarpJ plugin with ImageJ!)--recommend to add semicolon to avoid wasting time printing out the image result
- Get the parameter useing macro record function in the GUI and use MIJ.run() to realized batch and automatic processing
    - I use Mono mode: only results from the source to the target image
- …