MATLAB script

**Location analysis**

%This is the main script, which calls all the other functions.

%"locationAnalysis" is what needs to be put in the command window to get

%this to run.

%Any variables currently stored are cleared at the start of each run of the

%analysis script

clear

%PARAMETERS:

initial\_threshold = 30; %This is an intensity threshold against which the

%cell membrane image is compared. Any pixels in the image with an

%intensity (ranging from 0-255) above this threshold are classed as

%corresponding to the cell membrane and are used as a mask

filter\_threshold = 30; %This intensity threshold determines which pixels

%are classed as corresponding to the protein being analysed.

bulk = 0; %This defines the amount by which the width of cell membranes in

%the cell membrane mask are increased. Setting this to "0" will not

%increase the width. Changing this will increase the area around the cell

%membranes in which proteins are classed as being colocalised with the cell

%membrane

plot\_condition = 1; %This determines whether the system will display

%images. Setting this to "1" will show images and setting to "0" will

%surpress images

%MAIN SCRIPT:

%Asking the user to enter the root filename of the two images to be

%analysed. This name doesn't contain the "0000.tif" part of the name. It

%is a "string" variable type (word), so is written in single inverted

%commas

root\_filename = input('Enter filename root: ', 's');

%Displaying the current filename (as input by the user) in the command

%window

disp(['Analysing "',root\_filename, '"']);

%Loading the two images for analysis. "base\_im" is the image showing the

%cell membranes and "top\_im" is the image showing the proteins of interest

base\_im = imread(strcat([root\_filename,'0001.tif']));

top\_im = imread(strcat([root\_filename,'0000.tif']));

%Displaying the cell membrane image ("base\_im") if the user set

%"plot\_condition" to "1"

if plot\_condition == 1

%Opening a new figure window

figure();

%Populating the new figure window with "base\_im" and telling it to

%display in grayscale (the default colourmap is a visible colour

%spectrum)

image(base\_im), colormap(gray);

end

%Calling the main function, which generates and applies the image mask.

%This takes certain input variables (listed in brackets) and output other

%ones (in square brackets). The variables are listed in detail at the

%beginning of the function. The function is for analysing proteins

%colocalised with the cell membrane

[cell\_membrane\_im filt\_cell\_membrane\_im cell\_membrane\_table] = ...

cellMembraneAnalysis(base\_im, top\_im, initial\_threshold, bulk, ...

filter\_threshold);

%If the user specified "plot\_condition = 1" this will execute and display a

%series of images. In each case a new figure window is opened.

if plot\_condition == 1

%The protein image with only the cell membrane region displayed

figure();

image(cell\_membrane\_im), colormap(gray);

%The protein image with only the cell membrane region displayed and

%thresholded, so only pixels with an intensity above "filter\_threshold"

%are displayed

figure();

image(filt\_cell\_membrane\_im), colormap(gray);

end

%ANALYSIS:

%Calculating the percentage of pixels in the cell membrane region, which

%are above "filter\_threshold".

%"cell\_membrane\_table" contains all the pixels, which were determined

%previously to lie on a cell membrane. Measuring the number of rows in

%this table gives the total number of pixels on the cell membrane.

cell\_membrane\_num\_total = size(cell\_membrane\_table);

%Here, the number of pixels in the protein image, which lie on the cell

%membrane and have an intensity above "filter\_threshold" are counted.

cell\_membrane\_num\_populated = size(find(cell\_membrane\_table(:,4)==255));

%This calculates the percentage of pixels that correspond with protein

%(populated) in the total number of possible pixels (area in pixels of the

%cell membrane)

cell\_membrane\_percentage = (cell\_membrane\_num\_populated(1,1)/...

cell\_membrane\_num\_total(1,1))\*100;

%The percentage is displayed in the command window

disp([num2str(cell\_membrane\_percentage), '% of cell membrane populated']);

%A blank line is output to the command line to make the output look tidier

disp(' ');

**Make mask**

%This function takes a greyscale image and applies a pixel intensity

%threshold. Pixels above (or equal to) this threshold are set to "255" and

%those below are set to "0". There is also the option to increase the area

%surrounding the pixels above "255". With this "bulking" option all the

%pixels within a square region around each pixel above the threshold is set

%to "255" regardless of their initial intensity.

%INPUTS:

%"input\_im" is the input image, which is to be thresholded and turned into

%a mask

%"threshold" is the threshold to be applied to the input image. It has to

%have a value between "0" and "255"

%"bulk" is a distance in pixels, which determines the area around all

%pixels with initial intensity above the threshold in which all pixels will

%be set to "255". Setting this to "0" effectively turns the feature off.

%OUTPUTS:

%"mask" is the modified version of "input\_im", which displays all regions

%above the threshold with intensity "255" and those below with intensity

%"0"

function [mask] = makeMask(input\_im, threshold, bulk)

%Initialising the array "mask". This will be the same size as "input\_im",

%so mask is initially just set to be a copy of "input\_im"

mask = input\_im;

%Applying the threshold to "input\_im". All pixels in "mask" corresponding

%to pixels in "input\_im" below the threshold are set to "0"

mask(find(input\_im(:,:) < threshold)) = 0;

%Applying the other part of the threshold. All pixels above "threshold"

%are set to "255"

mask(find(input\_im(:,:) >= threshold)) = 255;

%Applying the bulking feature. This will always run, but if "bulk" is set

%to "0" the image will be unaffected.

%Generating a table called "mask\_table", which lists the pixel locations of

%all pixels, which were initially above the threshold (i.e. have thus far

%been assigned an intensity of "255")

[mask\_table(:,1), mask\_table(:,2)] = find(mask == 255);

%Measuring the size of "mask\_table". This will tell how many rows are in

%"mask\_table" and thus is used to determine how many times the bulking loop

%needs to run

size\_mask\_table = size(mask\_table);

%Once for each row in "mask\_table" this loop will run. At each iteration

%the value "i" will increase by "1". This is used to access the current

%row in "mask\_table"

for i = 1:size\_mask\_table(1,1)

%It will try to access some pixels outside the image when bulking,

%which would cause the system to crash. By using a "try" statement

%the system will skip any attempts at assigning a new intensity to

%a pixel where it fails.

try

%This section accesses a square region of "mask" around each pixel,

%which was initially above the threshold. The square region has the

%width (2\*bulk)+1 (i.e. "bulk" either side of the current pixel and

%the pixel itself) and the same height.

mask((mask\_table(i,1)-bulk):(mask\_table(i,1)+bulk),...

(mask\_table(i,2)-bulk):(mask\_table(i,2)+bulk)) = 255;

end

end

%Along two edges of the image (the edges with pixel locations not equal to

%"1") "mask" will be "bulk" wider than "input\_im". This can cause problems

%when applying the mask (trying to multiple matrices of different

%dimensions is not possible), so those extra rows and columns are deleted.

%To know which rows and columns to remove the size of "mask" must be known.

%The number of rows and columns is set to "rows" and "cols".

[rows, cols] = size(mask);

%This takes the extra rows and columns and sets them to "[]", which has the

%effect of removing those rows or columns.

mask(rows-bulk+1:rows,:) = [];

mask(:,cols-bulk+1:cols) = [];

**Cell membrane analysis**

%This function calls the functions responsible for generating and applying

%the mask, such that only regions of the top image ("top\_im") are

%accessible (all other regions being set to an intensity of "0"). The

%locations and intensities of all pixels classed as being in the cell

%membrane are stored in "cell\_membrane\_table" for analysis outside of the

%function.

%INPUTS:

%"base\_im" is the image showing the cell membranes, which will be turned

%into a mask for "top\_im"

%"top\_im" is the image showing the proteins to be analysed

%"initial\_threshold" is an intensity threshold, which determines which

%regions of "base\_im" will be classed as being cell membranes or membrane

%"bulk" is the pixel width added to the cell membranes to increase the area

%in which proteins are classed as being co-localised with the cell membrane

%"filter\_threshold" is an intensity threshold, which distinguishes between

%proteins and the background

%OUTPUTS:

%"cell\_membrane\_im" is "top\_im" with the cell membrane mask applied

%"filt\_cell\_membrane\_im" is "top\_im" with the cell membrane mask applied

%and all pixels with intensities above "filter\_threshold" set to 255 and

%those below set to "0"

%"cell\_membrane\_table" is a 4 column table containing information for all

%pixels identified as being in the cell membrane. The columns are

%horizontal position, vertical position, intensity in "top\_im" and

%post-filter intensity

function [cell\_membrane\_im filt\_cell\_membrane\_im cell\_membrane\_table] = ...

cellMembraneAnalysis(base\_im, top\_im, initial\_threshold, bulk, ...

filter\_threshold)

%This function generates the mask outlining the cell membranes. It also

%applies any cell membrane bulking as defined by the user (in "bulk")

[cell\_membrane\_mask] = makeMask(base\_im, initial\_threshold, bulk);

%Applying the mask to "top\_im". The mask is set as either "0" or "255", so

%for the purpose of applying the mask all values are divided by "255"

cell\_membrane\_im = top\_im.\*(cell\_membrane\_mask/255);

%Applying the filter to "cell\_membrane\_im", which sets all pixels above

%"filter\_threshold" to "255" and all those below it to "0". The filtered

%image is set to "filt\_cell\_membrane\_im".

[filt\_cell\_membrane\_im] = makeMask(cell\_membrane\_im, filter\_threshold, 0);

%Setting the first two columns of "cell\_membrane\_table" to the row and

%column of each pixel in the cell membrane. These are identified as all

%locations in "cell\_membrane\_mask" with an intensity equal to 255

[cell\_membrane\_table(:,1), cell\_membrane\_table(:,2)] = ...

find(cell\_membrane\_mask == 255);

%The number of pixels in the cell membrane is given by the size of

%"cell\_membrane\_table"

num\_px = size(cell\_membrane\_table);

%For each pixel listed by its coordinates in "cell\_membrane\_table" the

%pre-filtered intensity (actual intensity in "top\_im") and the

%post-filtered intensity ("0" or "255") are added to columns 3 and 4 of

%"cell\_membrane\_table" respectively.

for i = 1:num\_px(1,1)

cell\_membrane\_table(i,3) = cell\_membrane\_im(...

cell\_membrane\_table(i,1),cell\_membrane\_table(i,2));

cell\_membrane\_table(i,4) = filt\_cell\_membrane\_im(...

cell\_membrane\_table(i,1),cell\_membrane\_table(i,2));

end