REFERENCES


function drop_gui(input_str)

clear all

close all

% Define some global variables so information is not destroyed when
% the function finishes. Need to hold all the gui handles cause the findobj
% function in matlab is just way too slow

% Variables to hold the image information
global drop1_org_image drop1_threshold_image drop1_clean_image
global drop2_org_image drop2_threshold_image drop2_clean_image

% Handles for all the image boxes
global h_drop1_org h_drop1_threshold h_drop1_clean h_drop1_final
global h_drop2_org h_drop2_threshold h_drop2_clean h_drop2_final

% Handles for the text boxes etc in the drop panels
global h_drop1_invert_check h_drop1_threshold_slider h_drop1_threshold_slider_textbox
h_drop1_dropdown

% Handles for the objects used in the calculation panel
global h_calc_diameter_textbox h_calc_calibration_editbox h_drop1_remove_size_editbox
global stats1 stats2

% Some variables to handle file names
global drop1_file_name path previous_path

% PROGRAM DESCRIPTION
% This purpose of this matlab program is to easily and quickly calculate
% the diameter and velocity of falling drops. The drops are extracted from
% a source image by applying a threshold value to a grayscale image to
% convert it into a series of black or white pixels. The image is then
% cleaned up and analysed. The user gets IMMEDIATE feedback on how
% different parameters effect the final result.
%
% CURRENT PROBLEMS
% - Lots of error catching bug fixes required
% - Program flow is a bit all over the place, needs to be tidied up by
% - combining functions into 1 instead of having 2 seperate ones
% - Have to use global variables for everything otherwise they get deleted.
% - Not sure there is anyway around this
%
% IMPROVEMENTS REQUIRED
% - Write out the results to file, so they can be easily viewed in excel
% - Implement other more accurate edge detection methods
% - Implement the autothreshold feature
% - Implement a more accurate method of diameter determination. Use the
% profile of the drop to revolve it and get a volume out, then relate it
% back to a diameter
global boundary
global stats1 stats2

% Set the way images will be views
iptsetpref('ImshowBorder','tight');
iptsetpref('ImshowAxesVisible','off');
iptsetpref('ImviewInitialMagnification','fit');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CREATE THE GUI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The portion of the program creates all the figures, text boxes etc. etc
% used during the program. The GUI is structured as follows Main Figure -->
% 3 uipanel objects --> Buttons, Axes and Boxes children of the Uipanel
% objects. This allows the positioning of everything to be set relative to
% the panels not the main figure.
% ALL SPACING IS IN NORMALISED OR PERCENT TERMS
%
% CURRENT PROBLEMS
% - Stupid MATLAB won't allow me to have the axes as children of the
% uipanel object when imshow is executed. Bug in MATLAB. Only work around is
% to set the positions of the Axes relative to the main figure
%
% IMPROVEMENTS REQUIRED
% - Organise things a bit better, i.e. the naming of things need to follow
% a better convention then what I use now
%
% Extract screen size and dimensions required for the figure
figure_width = 1024;
figure_height = 768;
screen_size = get(0,'ScreenSize');

% Create main figure and add all the gui elements to it. Also centre
% the GUI
h_fig = figure('position',[screen_size(3)/2-figure_width/2 screen_size(4)/2-figure_height/2 figure_width figure_height ],'
' 'menubar','figure','
' 'color','white','
' 'name','Drop Velocity and Diameter Calculator','
' 'resize','off','
' 'numbertitle','off');

% Split the screen up into 3 panels for clarity. First two panels
% display all the image and the third one does the calculations

% Define a spacing height in percent
panel_height = 0.4;
h_drop1_panel = uipanel('position', [0 1-panel_height 1 panel_height],
  'title','Drop 1 Analysis Panel',
  'parent',h_fig);

h_drop2_panel = uipanel('position', [0 1-2*panel_height 1 panel_height],
  'title','Drop 2 Analysis Panel');
h_calc_panel = uipanel('position', [0 0 1-2*panel_height],...
    'title','Calculation Panel');

% Need to figure out where the image boxes and labels will lie relative to
% the border. Also need to find how big the boxes will be if the resolution
% is changed
side_buffer = 0.1;
bottom_buffer = 0.2;
top_buffer = 0.1;

image_box_spacing = 0.01;
image_box_width = (1 - 2*side_buffer - 3*image_box_spacing)/4;
image_box_height = (1 - top_buffer - bottom_buffer);
left_centre_offset = side_buffer+image_box_width/2;

% Create all the axes objects now
% DROP1 CALCULATION BOX
h_drop1_org = axes('units','normalized',....
    'position', [0.01 0.67 0.25 0.3],....
    'parent',h_fig,...
    'visible','off',....
    'box','on',....
    'activepositionproperty','position');
%'PlotBoxAspectRatio',[1 1 1],....
%'DataAspectRatio',[1 1 1],....
%'DataAspectRatioMode','manual');

h_drop1_threshold = axes('units','normalized',....
    'position', [0.26 0.67 0.23 0.3],....
    'parent',h_fig,...
    'visible','off',....
    'box','on',....
    'PlotBoxAspectRatio',[1 1 1]);

h_drop1_clean = axes('units','normalized',....
    'position', [0.51 0.67 0.23 0.3],....
    'parent',h_fig,...
    'visible','off',....
    'box','on',....
    'PlotBoxAspectRatio',[1 1 1]);

% DROP2 CALCULATION BOX
h_drop2_org = axes('units','normalized',....
    'position', [0.01 0.27 0.23 0.3],....
    'parent',h_fig,...
    'visible','off',....
    'box','off',....
    'activepositionproperty','position',....
    'PlotBoxAspectRatio',[1 1 1]);
h_drop2_threshold = axes('units','normalized',... 'position', [0.26 0.27 0.23 0.3],... 'parent',h_fig,... 'visible','off',... 'box','on',... 'PlotBoxAspectRatio',[1 1 1]);

h_drop2_clean = axes('units','normalized',... 'position', [0.51 0.27 0.23 0.3],... 'parent',h_fig,... 'visible','off',... 'box','on',... 'PlotBoxAspectRatio',[1 1 1]);

h_drop2_final = axes('units','normalized',... 'position', [0.76 0.27 0.23 0.3],... 'parent',h_fig,... 'visible','off',... 'box','on',... 'PlotBoxAspectRatio',[1 1 1]);

%% Create all the buttons and text boxes. ALL THESE CONTROLS ARE INTERACTIVE
% DROP1 PANEL ITEMS
h_drop1_open_button = uicontrol('units','normalized',... 'style','pushbutton',... 'string','Open',... 'parent',h_drop1_panel,... 'position',[0.075 0.05 0.1 0.1],... 'callback',@open_image1);

h_drop1_invert_check = uicontrol('units','normalized',... 'style','checkbox',... 'value',0,... 'string','Invert',... 'position',[0.275 0.05 0.1 0.05],... 'parent',h_drop1_panel,... 'callback',@threshold_image1);

h_drop1_threshold_slider = uicontrol('units','normalized',... 'style','slider',... 'position',[0.325 0.08 0.15 0.05],... 'max',1,... 'min',0,... 'sliderstep',[0.025 0.05],... 'value',0.4,... 'callback',@threshold_image1,... 'parent',h_drop1_panel);

h_drop1_threshold_2_slider = uicontrol('units','normalized',... 'style','slider',... 'position',[0.325 0.03 0.15 0.05],... 'max',1,... 'min',0,... 'sliderstep',[0.025 0.05],... 'value',0.3,... 'callback',@threshold_image1,... 'parent',h_drop1_panel);

h_drop1_threshold_slider_textbox = uicontrol('units','normalized',... 'style','text',...
h_drop1_threshold_2_slider_textbox = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.4 0.13 0.05 0.05],...
    'string',get(h_drop1_threshold_slider,'value'),...
    'parent',h_drop1_panel);

h_drop1_remove_size_editbox = uicontrol('units','normalized',...
    'style','edit',...
    'position',[0.65 0.05 0.075 0.075],...
    'string','10',...
    'backgroundcolor',[1 1 1],...
    'parent',h_drop1_panel,...
    'callback',@clean_image1);

h_drop1_dropdown = uicontrol('units','normalized',...
    'style','popupmenu',...
    'position',[0.9 0.05 0.075 0.075],...
    'string','2',...
    'backgroundcolor',[1 1 1],...
    'parent',h_drop1_panel,...
    'callback',@calculate_data);

% DROP2 PANEL ITEMS
h_drop2_open_button = uicontrol('units','normalized',...
    'style','pushbutton',...
    'string','2nd Image',...
    'parent',h_drop2_panel,...
    'position',[0.075 0.05 0.1 0.1],...
    'callback',@open_image2,...
    'enable','off');

h_drop2_invert_check = uicontrol('units','normalized',...
    'style','checkbox',...
    'value',0,...
    'string','Invert',...
    'position',[0.275 0.05 0.1 0.05],...
    'parent',h_drop2_panel,...
    'callback',@threshold_image2);

h_drop2_threshold_slider = uicontrol('units','normalized',...
    'style','slider',...
    'position',[0.325 0.08 0.15 0.05],...
    'max',1,...
    'min',0,...
    'sliderstep',[0.025 0.05],...
    'value',0.4,...
    'callback',@threshold_image2,...
    'parent',h_drop2_panel);

h_drop2_threshold_2_slider = uicontrol('units','normalized',...
    'style','slider',...
    'position',[0.325 0.03 0.15 0.05],...
    'max',1,...
    'min',0,...
    'sliderstep',[0.025 0.05],...
    'value',0.3,...
% CALCULATION PANEL ITEMS
h_calc_velocity_textbox = uicontrol('units','normalized',...'
    'style','text',....
    'position',[0.815 0.1 0.075 0.1],....
    'string',' ',....
    'parent',h_calc_panel,...
    'backgroundcolor',[1 1 1]);

h_calc_diameter1_textbox = uicontrol('units','normalized',....
    'style','text',....
    'position',[0.815 0.7 0.075 0.1],....
    'string',' ',....
    'parent',h_calc_panel,...
    'backgroundcolor',[1 1 1]);

h_calc_diameter2_textbox = uicontrol('units','normalized',....
    'style','text',....
    'position',[0.815 0.5 0.075 0.1],....
    'string',' ',....
    'parent',h_calc_panel,...
    'backgroundcolor',[1 1 1]);

h_calc_average_diameter_textbox = uicontrol('units','normalized',....
    'style','text',....
    'position',[0.815 0.3 0.075 0.1],....
    'string',' ',....
    'parent',h_calc_panel,...
    'backgroundcolor',[1 1 1]);
h_calc_eccentricity1_textbox = uicontrol('units','normalized','
    'style','text',
    'position',[0.9 0.7 0.075 0.1],
    'string','','
    'parent',h_calc_panel,
    'backgroundcolor',[1 1 1]);

h_calc_eccentricity2_textbox = uicontrol('units','normalized','
    'style','text',
    'position',[0.9 0.5 0.075 0.1],
    'string','','
    'parent',h_calc_panel,
    'backgroundcolor',[1 1 1]);

h_calc_average_eccentricity_textbox = uicontrol('units','normalized','
    'style','text',
    'position',[0.9 0.3 0.075 0.1],
    'string','','
    'parent',h_calc_panel,
    'backgroundcolor',[1 1 1]);

h_calc_framerate_editbox = uicontrol('units','normalized','
    'style','edit',
    'position',[0.5 0.5 0.1 0.1],
    'string',num2str(2000),
    'parent',h_calc_panel,
    'backgroundcolor',[1 1 1],
    'callback',@calculate_data);

h_calc_sigma_editbox = uicontrol('units','normalized','
    'style','edit',
    'position',[0.5 0.3 0.1 0.1],
    'string','1',
    'parent',h_calc_panel,
    'backgroundcolor',[1 1 1],
    'callback',@threshold_image1);

h_calc_calibration_editbox = uicontrol('units','normalized','
    'style','edit',
    'position',[0.35 0.5 0.15 0.1],
    'string',num2str(0.022),
    'parent',h_calc_panel,
    'backgroundcolor',[1 1 1],
    'callback',@calculate_data);

h_calc_location_editbox = uicontrol('units','normalized','
    'style','text',
    'position',[0.01 0.8 0.3 0.1],
    'string','','
    'parent',h_calc_panel,
    'backgroundcolor',[1 1 1]);

h_calc_analysis_type = uicontrol('units','normalized','
    'style','popupmenu',
    'position',[0.01 0.6 0.1 0.1],
    'string','Canny|Threshold|Sobel|Prewitt|Laplacian|Zero-Cross',
    'parent',h_calc_panel);

%% Create all the NON-INTERACTIVE text boxes
h_drop1_remove_size_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.5 0.03 0.15 0.1],...
    'string','Size in pixels of the smallest object to be removed',...
    'parent',h_drop1_panel);

h_drop1_regionpick_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.75 0.03 0.15 0.1],...
    'string','Which regions properties do you wish to analyse?',...
    'parent',h_drop1_panel);

h_drop2_remove_size_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.5 0.03 0.15 0.1],...
    'string','Size in pixels of the smallest object to be removed',...
    'parent',h_drop2_panel);

h_drop2_regionpick_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.75 0.03 0.15 0.1],...
    'string','Which regions properties do you wish to analyse?',...
    'parent',h_drop2_panel);

h_calc_velocity_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.7 0.1 0.1 0.1],...
    'string','Velocity (m/s)',...
    'parent',h_calc_panel);

h_calc_diameter_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.8 0.8 0.1 0.1],...
    'string','Diameter (mm)',...
    'parent',h_calc_panel);

h_calc_eccentricty_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.9 0.8 0.075 0.1],...
    'string','Eccentricity',...
    'parent',h_calc_panel);

h_calc_drop1_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.7 0.7 0.1 0.1],...
    'string','Drop 1',...
    'parent',h_calc_panel);

h_calc_drop2_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.7 0.5 0.1 0.1],...
    'string','Drop 2',...
    'parent',h_calc_panel);

h_calc_average_label = uicontrol('units','normalized',...
    'style','text',...
    'position',[0.7 0.3 0.1 0.1],...
    'string','Average',...
% This function opens up the image after the user has clicked the open
% button. A UI for getting the file is displayed.

% OPEN THE INPUT IMAGE/IMAGES

function open_image1(obj, event)

% obj = 0;
% event = 0;
%

previous_path = '*.tif';

[drop1_file_name, path] = uigetfile(previous_path, 'Open Image 1');
drop1_full_name = strcat(path, drop1_file_name);
drop1_org_image = imread(drop1_full_name);

%structure_ele = strel('disk', 2);
%drop1_org_image = imtophat(drop1_org_image, structure_ele);
%drop1_org_image = imadjust(drop1_org_image, [0 0.01], [0 0]);

previous_path = strcat(path, '*.tif');
set(h_calc_location_editbox,'string', strcat(path, drop1_file_name));

% Find the figure to put the original image in and show it
set(gcf,'CurrentAxes',h_drop1_org);
imshow(drop1_org_image);

% Start the threshold function which will create the binary image
threshold_image1(obj, event);
% Need to find what the next image file will be called so we can call
% it straight up
filename_length = length(drop1_file_name);

% Find where the first digit of the incrementing number starts
start_number = filename_length - 4 - 5;
new_frame_number = str2num(drop1_file_name(start_number:start_number+5)) + 1;

% Convert the number into a string and add leading zeros to the start
new_filename = num2str(new_frame_number);
while length(new_filename) ~= 6
    new_filename = strcat('0',new_filename);
end

% Put it all together all read in the next frame
final_filename = strcat(drop1_file_name(1:start_number-1),new_filename,'.tif');
don2_full_name = strcat(path, final_filename);
don2_org_image = imread(don2_full_name);

% Find the figure to put the original image in and show it
set(gcf,'CurrentAxes',h_don2_org);
imshow(don2_org_image);

% Start the threshold function which will create the binary image
threshold_image2(obj,event);

end

%function open_image2(obj,event)
  % drop2_file = uigetfile('*.*','Open Image 2')
  % don2_org_image = imread(don2_file);
  
  % Find the figure to put the original image in and show it
  % set(gcf,'CurrentAxes',h_don2_org);
  % imshow(don2_org_image);
  
  % Start the threshold function which will create the binary image
  % threshold_image2(obj,event);

%end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THRESHOLD THE IMAGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function essentially takes the raw image and turns every pixel into
% either a fully black or fully white pixel. The cut off limit for the
% transition between black and white is controlled by THRESHOLD_VALUE when
% the user inputs via the slider
%
% IMPROVEMENTS REQUIRED

% - Allow the changing of the threshold hold so that different methods can
% be used i.e. sobel, canning etc. etc. This would probably require the
% algorithm to be re-written
% - Allow the user to enter in a threshold value into a text box

function threshold_image1(obj, event)

    % Get the threshold value from the slider. Also output the value from
    % the slider into the text box above it
    threshold_value = get(h_drop1_threshold_slider,'value');
    threshold_2_value = get(h_drop1_threshold_2_slider,'value');
    set(h_drop1_threshold_slider_textbox,'string',num2str(threshold_value));
    set(h_drop1_threshold_2_slider_textbox,'string',num2str(threshold_2_value));
    sigma_value = str2num(get(h_calc_sigma_editbox,'string'));

    % Do the edge detection of the image based on which analysis type is
    % selected
    switch analysis_type
        case 1
            drop1_threshold_image = edge(drop1_org_image,'canny',[threshold_2_value
            threshold_value],sigma_value);
        case 2
            drop1_threshold_image = im2bw(drop1_org_image,threshold_value);
        case 3
            drop1_threshold_image = edge(drop1_org_image,'sobel',threshold_value);
        case 4
            drop1_threshold_image = edge(drop1_org_image,'prewitt',threshold_value);
        case 5
            drop1_threshold_image = edge(drop1_org_image,'log',threshold_value);
        case 6
            drop1_threshold_image = edge(drop1_org_image,'zerocross',threshold_value);
    end

    % Now read off the state of the invert check box and invert the image if it
    % is selected
    invert_box_state=get(h_drop1_invert_check,'value');
    if invert_box_state == 1
        drop1_threshold_image=~drop1_threshold_image;
    end

    % Display the thresholded image
    set(gcf,'CurrentAxes',h_drop1_threshold);
    imshow(drop1_threshold_image);

    clean_image1(obj, event);
end

function threshold_image2(obj, event)

    % Get the threshold value from the slider. Also output the value from
    % the slider into the text box above it

end
threshold_value = get(h_drop2_threshold_slider,'value');
threshold_2_value = get(h_drop2_threshold_2_slider,'value');
set(h_drop2_threshold_slider_textbox,'string',num2str(threshold_value));
set(h_drop2_threshold_2_slider_textbox,'string',num2str(threshold_2_value));

analysis_type = get(h_calc_analysis_type,'value');

switch analysis_type
    case 1
        drop2_threshold_image = edge(drop2_org_image,'canny',threshold_value);
        drop2_threshold_image = edge(drop2_org_image,'canny',threshold_value);
    case 2
        drop2_threshold_image = im2bw(drop2_org_image,threshold_value);
end

% Now read off the state of the invert check box and invert the image if it
% is selected
if invert_box_state == 1
    drop2_threshold_image = ~drop2_threshold_image;
end

% Display the thresholded image
set(gcf,'CurrentAxes',h_drop2_threshold);
imshow(drop2_threshold_image);
clean_image2(obj, event);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CLEANUP THE IMAGE%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function essentially fills in any gaps in the region and gets rid of%
% any small objects
%
% CURRENT PROBLEMS
%
% IMPROVEMENTS REQUIRED
%
function clean_image1(obj, event)

% Read the minimum size value from the text box
min_size = str2num(get(h_drop1_remove_size_editbox,'string'));

% Remove all object containing fewer than the specified number in pixels
drop1_clean_image = bwareaopen(drop1_threshold_image,min_size);

% Fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
drop1_clean_image = imfill(drop1_clean_image,4,'holes');
function clean_image2(obj, event)

% Read the minimum size value from the text box
min_size = str2num(get(h_drop2_remove_size_editbox,'string')); %*
% Remove all object containing fewer than the specified number in pixels
drop2_clean_image = bwareaopen(drop2_threshold_image,min_size);

% Fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
drop2_clean_image = imfill(drop2_clean_image,'holes');

% Fill in any gaps in the image
% se = strel('disk',2);
% drop1_clean_image = imclose(drop1_clean_image,se);

% Output the cleaned up image for the user to see
set(gcf,'CurrentAxes',h_drop2_clean);
imshow(drop2_clean_image);

final_image2(obj, event)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
% DISPLAY THE FINAL IMAGE WITH REGIONS
%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The original image is displayed in box with the outline and number of the
% regions on it.
%
% CURRENT PROBLEMS
% - The text from other regions is kept on screen after the regions have
%    changed. I think this will require a screen redraw
%
% IMPROVEMENTS REQUIRED
% - Would be nice if you could click on the axes and it puts the image in a
%    large figure so you can investigate it a bit better.

function final_image1(obj, event)

drop1_popup_string = '';

% Find boundaries of all objects in image. If one region has points inside
% it, remove those points.
[B,L] = bwboundaries(drop1_clean_image,'noholes');

% Draw the original image on the axes ready for the region outlines
set(gcf,'CurrentAxes',h_drop1_final);
imshow(drop1_org_image);

% Draw the region outlines on the original image
hold on
for k = 1:length(B)
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end

% Extract the centroids so they can be displayed
% Create statistics of the region properties for analysis
stats1 = regionprops(L,'Area','Centroid','EquivDiameter','Eccentricity','Orientation','MajorAxisLength','MinorAxisLength');

% Calculate the diameter of each droplet in the image
for k = 1:length(B)
    diameter = stats1(k).EquivDiameter;
    % Find the centroid of the image for velocity measurement
    centroid = stats1(k).Centroid;
    angle = stats1(k).Orientation;
    major_length = stats1(k).MajorAxisLength;
    minor_length = stats1(k).MinorAxisLength;
    text(centroid(1),centroid(2),num2str(k),'color',[1 1 1]);
end

% Populate the drop down box with all the available regions
drop1_popup_string = strcat(drop1_popup_string, num2str(k), '|');
set(h_drop1_dropdown,'string',drop1_popup_string);

function final_image2(obj, event)

    drop2_popup_string = '';

    % Find boundaries of all objects in image. If one region has points inside
    % it, remove those points.
    [B,L] = bwboundaries(drop2_clean_image,'noholes');

    % Draw the original image on the axes ready for the region outlines
    set(gcf,'CurrentAxes',h_drop2_final);
imshow(drop2_org_image);

    % Draw the region outlines on the original image
    hold on
    for k = 1:length(B)
        boundary = B{k};
    end

end
plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end

% Extract the centroids so they can be displayed
% Create statistics of the region properties for analysis
stats2 = regionprops(L,'Area','Centroid','EquivDiameter','Eccentricity');

% Calculate the diameter of each droplet in the image
for k = 1:length(B)
    % Find the equivalent diameter from region properties
diameter = stats2(k).EquivDiameter;
    % Find the centroid of the image for velocity measurement
    centroid = stats2(k).Centroid;
    % Populate the drop down box with all the available regions
    drop2_popup_string = strcat(drop2_popup_string, num2str(k), '|');
    set(h_drop2_dropdown,'string',drop2_popup_string);
end

calculate_data(obj, event)

% CURRENT PROBLEMS
% -
% % IMPROVEMENTS REQUIRED
% - Allow the user to select whether they use the XML file or not
% - Results should be output to a text file for import in another program.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
% CALCULATE VELOCITY AND DIAMETER
%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This function does all the important calculations. Initially, the frame
% rate from the XML file is extracted and used to calculate the velocity.
% Then the regions selected by the user are used to pull out all the
% diameter and centroid data
%
% CURRENT PROBLEMS
% -
% % IMPROVEMENTS REQUIRED
% - Allow the user to select whether they use the XML file or not
% - Results should be output to a text file for import in another program.

function calculate_data(obj, event)

% Extract the framerate from the XML file
filename_length = length(drop1_file_name);
xml_file = strcat(path, drop1_file_name(1:filename_length - 4 - 7), '.xml');
file_id = fopen(xml_file, 'r');
xml_data = fscanf(file_id, '%c', 1000);
frame_rate_start = findstr(xml_data, '<FrameRate>') + 11;
frame_rate_end = findstr(xml_data, '</FrameRate>') - 1;

% Fill in the frame rate box with the extracted number
frame_rate = str2num(xml_data(frame_rate_start:frame_rate_end));
set(h_calc_framerate_editbox,'string',num2str(frame_rate));
% Extract the data for the regions based on what is picked in the popup menu
dropdown1_value = get(h_drop1_dropdown,'value');
dropdown2_value = get(h_drop2_dropdown,'value');

% Extract info from calibration and framerate boxes
calibration_factor = str2num(get(h_calc_calibration_editbox,'string'));
frame_rate = str2num(get(h_calc_framerate_editbox,'string'));

% Calculate the diameter of the drop from the first drop and apply the calibration factor
diameter1 = stats1(dropdown1_value).EquivDiameter * calibration_factor;
diameter2 = stats2(dropdown2_value).EquivDiameter * calibration_factor;
average_diameter = (diameter1 + diameter2)/2;
set(h_calc_diameter1_textbox,'string',num2str(diameter1));
set(h_calc_diameter2_textbox,'string',num2str(diameter2));
set(h_calc_average_diameter_textbox,'string',num2str(average_diameter));

% Find the eccentricity of the drops and display it
eccentricity1 = stats1(dropdown1_value).Eccentricity;
eccentricity2 = stats2(dropdown2_value).Eccentricity;
average_eccentricity = (eccentricity1 + eccentricity2) / 2;
set(h_calc_eccentricity1_textbox,'string',num2str(eccentricity1));
set(h_calc_eccentricity2_textbox,'string',num2str(eccentricity2));
set(h_calc_average_eccentricity_textbox,'string',num2str(average_eccentricity));

% Figure out the centroid and display the velocity
centroid1 = stats1(dropdown1_value).Centroid;
centroid2 = stats2(dropdown2_value).Centroid;

displacement = ((centroid1(1)-centroid2(1))^2+(centroid1(2)-centroid2(2))^2)^0.5;
final_velocity = displacement * calibration_factor * frame_rate / 1000;
set(h_calc_velocity_textbox,'string',num2str(final_velocity));

end

function export_data()
end

end