```
clc; clear; close all
cd 'G:\Jiaxu Song\test_for_length_recording\test6'
addpath 'G:\Jiaxu Flashdrive Backup\code\functions'
```

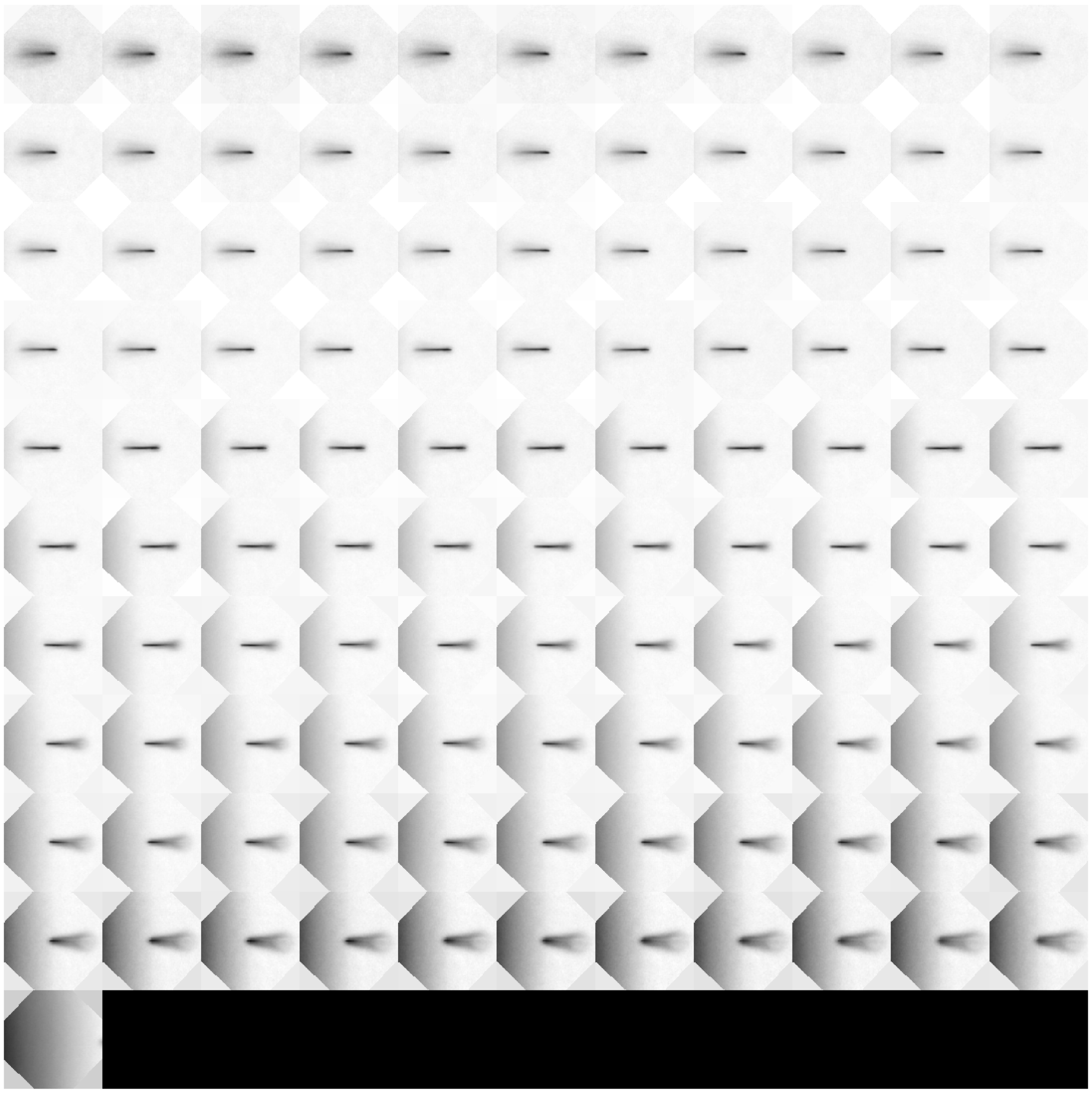
```
imds = imageDatastore("G:\Jiaxu Song\test_for_length_recording\test6");
```

```
for i = 1:length(imds.Files)
    I(:,:,i) = readimage(imds,i);
    [filepath,name,ext] = fileparts(imds.Files{i});
    numbers = regexp(name,'\d*','Match');
    numbers = str2double(numbers);
    leng(i) = numbers(2)/10; angle(i) = numbers(4); width(i) = numbers(3)/10; height(i) = numbe
end
```

```
for i = 1:length(imds.Files)
    if angle(i) > 90
        angle(i) = angle(i) - 180;
    end
end
lens = 40; %20 40
if lens == 10
    pixeltoum = 0.65; %um/pixal
elseif lens == 40
    pixeltoum = 0.1625; %um/pixal
end
max_length = max(leng);
window_size = round(max_length);
```

```
I1 = readimage(imds,1);
figure(1)
imshow(I1);
[x,y] = ginput(1);
offset = 100;
close all
x = round(x); y = round(y);
```

```
I_half_width = 150; I__half_height = 150;
for i = 1:length(imds.Files)
    clear centroid_x centroid_y max_tensity float_I scale_I reverse_scale_I
    image_of_interested = chopimage(I(:,:,i),x,y,I_half_width,I__half_height);
    x_left_top = x-I_half_width;
    y_left_top = y-I__half_height;
    left_top = [x_left_top,y_left_top];
    max_tensity = min(min(image_of_interested));
    [cen_rows,cen_col] = find(image_of_interested <= max_tensity + offset);
    centroid_x = mean(cen_col);
    centroid_y = mean(cen_rows);
    center_point= [centroid_x,centroid_y];
    centroid(i,:) = round(center_point)+left_top;
    %     new_image_of_interested(:,:,i) = chopimage(I(:,:,i),centroid(i,1),centroid(i,2),75,75
    x_center = centroid(i,1);
    y_center = centroid(i,2);
    centered_image_of_interested = chopimage(I(:,:,i),x_center,y_center,window_size,window_size
    resizedimage = imresize(centered_image_of_interested, [100 100]);
    ang = angle(i);
    float_I = double(resizedimage);
    scale_I = scale_image(float_I);
    reverse_scale_I = reverse_img(round(scale_I*255));
    I_grayscale = reverse_scale_I;
    max_value = max(max(I_grayscale));
    avg_value = mean(mean(I_grayscale))+min(min(I_grayscale));
    pad_value = (double(avg_value)+double(max_value))/2+randi([0,10]);
    if ang > 0
        img_r = imrotate_white(I_grayscale,-ang,pad_value);
    elseif ang <= 0
        img_r = imrotate_white(I_grayscale,ang,pad_value);
    end

    newrotateImg= img_r;

%     threshold = graythresh(newrotateImg);
%     max_sharp_img_bw = imbinarize(newrotateImg,threshold);
%     [B,~] = bwboundaries(max_sharp_img_bw);
%     image_thresholded = uint8(max_sharp_img_bw);
%     B{1} = [];
%     s = size(B);
%     if s(1) ==0
%         continue
%     end
%     for j = 1:s(1)
%         [b_s(j),~] = size(B{j});
%     end
%     if b_s == 0
%         continue
%     end
%     max_b = max(b_s);
%     NW_b = find(b_s == max_b);
%     boundary = B{NW_b};
%     boundary_x = B{NW_b}(:,2);
%     boundary_y = B{NW_b}(:,1);
%     left_most_point_x = round(mean(min(boundary_x)));
%     left_most_point_y = round(mean(boundary_y(left_most_point_x)));

    rotate_ROI(:,:,i) = newrotateImg;
end
montage(rotate_ROI)
```



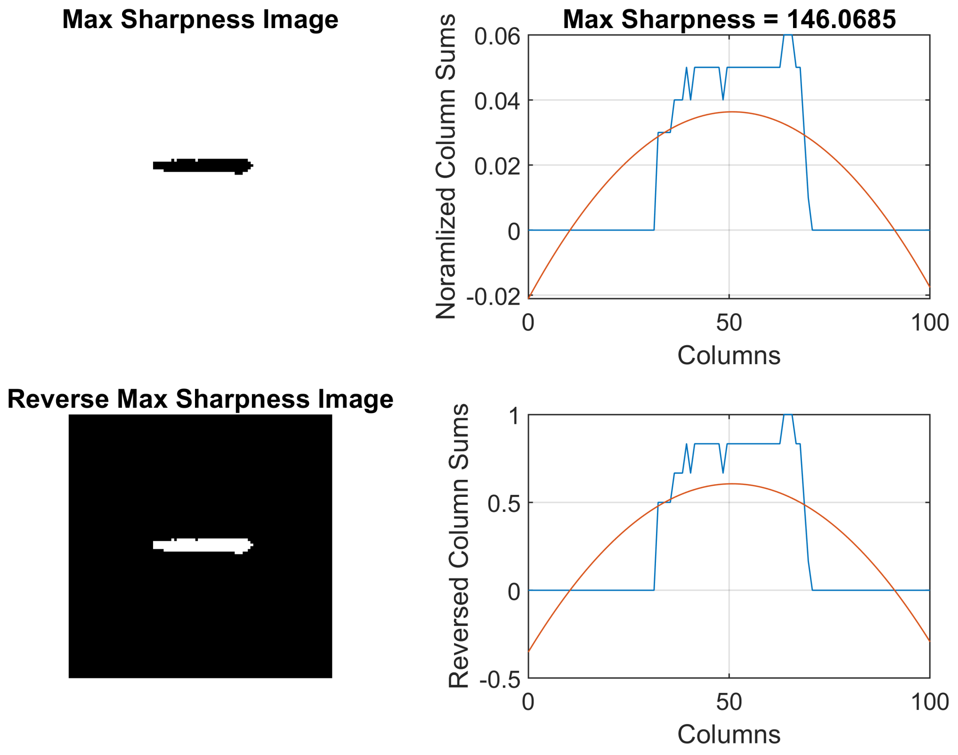
```
for i = 1:length(imds.Files)
    sharpness(i) = brenner(rotate_ROI(:,:,i))*mean2(rotate_ROI(:,:,i));
end
max_sharp = max(sharpness);
max_sharp_img =find(sharpness==max_sharp);
max_image = rotate_ROI(:,:,max_sharp_img);
```

```
threshold = graythresh(max_image);
max_sharp_img_bw = imbinarize(max_image,threshold);
[B,L] = bwboundaries(max_sharp_img_bw);
B{1} = [];
s = size(B);
for j = 1:s(1)
    [b_s(j),~] = size(B{j});
end
max_b = max(b_s);
NW_b = find(b_s == max_b);
boundary = B{NW_b};
boundary_x = B{NW_b}(:,2);
boundary_y = B{NW_b}(:,1);
nw_boundary_plot = max_image(min(boundary_y):max(boundary_y),min(boundary_x):max(boundary_x));
white_background = uint8(ones(100,100)*255);
white_background(min(boundary_y):max(boundary_y),min(boundary_x):max(boundary_x)) = nw_boundar
```

```
% BW = poly2mask(boundary_x,boundary_y,100,100);
BW_NW_only = uint8(abs(double(max_sharp_img_bw)-1));
BW_White_BG = uint8(abs(double(max_sharp_img_bw)));
BW_NW = (double(BW_NW_only) .* double(max_image));
BW_BG = double(BW_White_BG).* ones(100,100)*255;
BW_out = uint8(BW_NW + BW_BG);
% BW_new_reverse = reverse_img((BW_new));
figure()
imshow(BW_BG)
```

```
column_value = sum(BW_BG);
norm_column_value = column_value / max(column_value);
x_axis = linspace(0,100,100);
y_axis = abs(1-norm_column_value);
p = polyfit(x_axis,y_axis,2);
y1 = polyval(p,x_axis);
reverse_image = uint8(abs(double(BW_BG) - 255));
reverse_column_value = sum(reverse_image);
reverse_norm_column_value = reverse_column_value / max(reverse_column_value);
reverse_y_axis = reverse_norm_column_value;
p_r = polyfit(x_axis,reverse_y_axis,2);
r_y1 = polyval(p_r,x_axis);
```

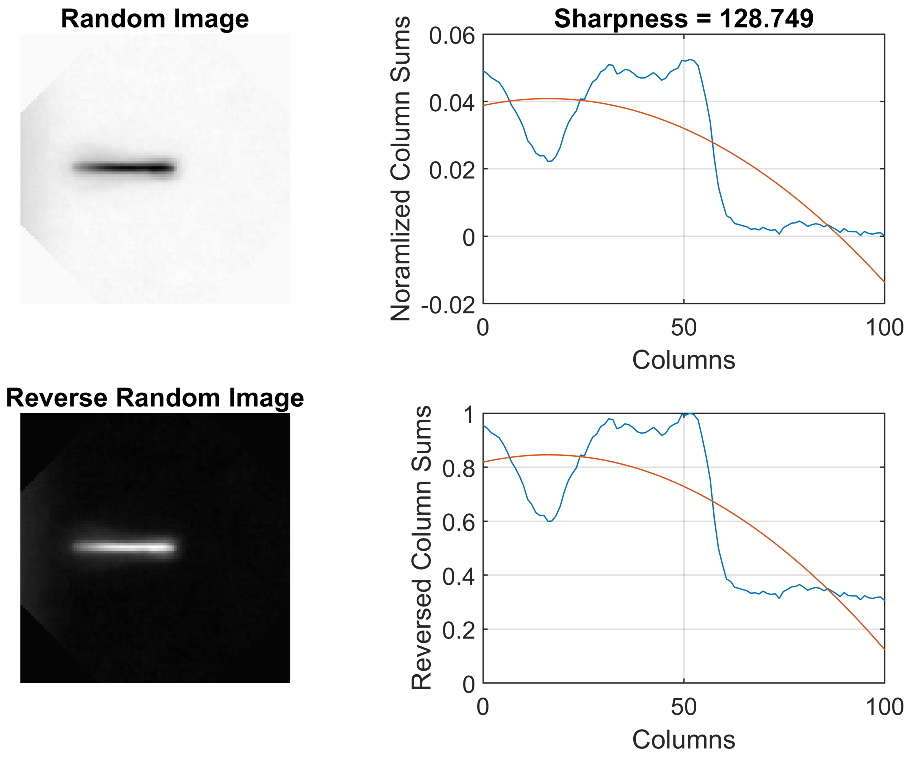
```
figure()
subplot(2,2,1)
imshow(BW_BG)
title('Max Sharpness Image')
subplot(2,2,2)
plot(x_axis,y_axis); hold on
plot(x_axis,y1); hold off
set(gcf,'position',[20,20,600,200]);
grid on
xlabel('Columns');
ylabel('Noramlized Column Sums')
title(['Max Sharpness = ' num2str(sharpness(max_sharp_img))])
subplot(2,2,3)
imshow(reverse_image)
title('Reverse Max Sharpness Image')
subplot(2,2,4)
plot(x_axis,reverse_y_axis); hold on
plot(x_axis,r_y1); hold off
set(gcf,'position',[20,20,600,400]);
grid on
xlabel('Columns');
ylabel('Reversed Column Sums')
```



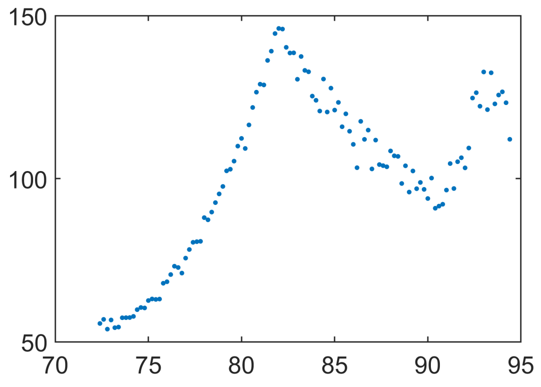
```
clear B b_s
random_number = randi([1,length(imds.Files)]);
random_image = rotate_ROI(:,:,random_number);
reverse_random_image = uint8(abs(double(random_image) - 255));
rand_reverse_column_value = sum(reverse_random_image);
rand_reverse_norm_column_value = rand_reverse_column_value / max(rand_reverse_column_value);
```

```
rand_reverse_y_axis = rand_reverse_norm_column_value;
rand_p_r = polyfit(x_axis,rand_reverse_y_axis,2);
rand_r_y1 = polyval(rand_p_r,x_axis);

threshold = graythresh(random_image);
rand_sharp_img_bw = imbinarize(random_image,threshold);
BW_NW_only = uint8(abs(double(rand_sharp_img_bw)-1));
BW_White_BG = uint8(abs(double(rand_sharp_img_bw)));
BW_NW = (double(BW_NW_only) .* double(random_image));
BW_BG = double(BW_White_BG).* ones(100,100)*255;
BW_out = uint8(BW_NW + BW_BG);
[B,L] = bwboundaries(rand_sharp_img_bw);
B{1} = [];
s = size(B);
for j = 1:s(1)
    [b_s(j),~] = size(B{j});
end
max_b = max(b_s);
NW_b = find(b_s == max_b);
boundary = B{NW_b};
boundary_x = boundary(:,2);
boundary_y = boundary(:,1);
randi_img_centroid = centroid(random_number,:);
mean_x = round(mean(boundary_x));
if mean_x > 50
    correction = mean_x - 50;
elseif mean_x < 50
    correction = 50 - mean_x;
else
    correction = 0;
end

figure()
subplot(2,2,1)
imshow(random_image)
title('Random Image')
column_value = sum(random_image);
norm_column_value = column_value / max(column_value);
x_axis = linspace(0,100,100);
y_axis = abs(1-norm_column_value);
p = polyfit(x_axis,y_axis,2);
y1 = polyval(p,x_axis);
subplot(2,2,2)
plot(x_axis,y_axis); hold on
plot(x_axis,y1); hold off
set(gcf,'position',[20,20,600,200]);
grid on
xlabel('Columns');
ylabel('Noramlized Column Sums')
title(['Sharpness = ' num2str(sharpness(random_number))])
subplot(2,2,3)
imshow(reverse_random_image)
title('Reverse Random Image')
subplot(2,2,4)
```

```
plot(x_axis,rand_reverse_y_axis); hold on
plot(x_axis,rand_r_y1); hold off
set(gcf,'position',[20,20,600,400]);
grid on
xlabel('Columns');
ylabel('Reversed Column Sums')
```



```
figure()
plot(height,sharpness,'.')
set(gcf,'position',[20,20,300,200]);
```



```
function rotated_image = imrotate_white(image, rot_angle_degree,pad_value)
RA = imref2d(size(image));
tform = affine2d([cosd(rot_angle_degree)    -sind(rot_angle_degree)     0; ...
    sind(rot_angle_degree)     cosd(rot_angle_degree)     0; ...
    0                          0                          1]);
Rout = images.spatialref.internal.applyGeometricTransformToSpatialRef(RA,tform);
Rout.ImageSize = RA.ImageSize;
xTrans = mean(Rout.XWorldLimits) - mean(RA.XWorldLimits);
yTrans = mean(Rout.YWorldLimits) - mean(RA.YWorldLimits);
Rout.XWorldLimits = RA.XWorldLimits+xTrans;
Rout.YWorldLimits = RA.YWorldLimits+yTrans;
rotated_image = imwarp(image, tform, 'OutputView', Rout, 'interp', 'cubic', 'fillvalues', pad_v
end
```