```
clear; clc; close all;

%% Read the images and assign them to the respective variables

folder1 = 'screw images top and front/';
file1 = 'first screw image';
extension1 = '.bmp';
filepath1 = [folder1,file1,extension1];
OriginalImg = imread(filepath1);
%figure,imshow(OriginalImg)

folder2 = 'screw images top and front/';
file2 = 'Blank_Img';
extension2 = '.bmp';
filepath2 = [folder2,file2,extension2];
BlankImg = imread(filepath2);
%figure,imshow(BlankImg)

ScrewImg = BlankImg-OriginalImg;
figure,imshow(ScrewImg)

%% binarizing the image:

[pixelCount,grayLevels]= imhist(ScrewImg);
Threshold = otsuthresh(pixelCount);
BrightImg = imbinarize(ScrewImg,Threshold);
%figure, imshow(BrightImg);

%% Connected Components

% conn=8;
% labeledImg = bwlabel(BrightImg, conn);
% colLabels = label2rgb (labeledImg, 'hsv', 'k', 'shuffle');
% figure,imshow(colLabels);

%% Regional properties:

Out_BrightImg = BrightImg;

% Remove portions of the image that touch an outside edge.
%Out_BrightImg = imclearborder(Out_BrightImg);

% Fill holes in regions.
Out_BrightImg = imfill(Out_BrightImg, 'holes');

% Filter image based on image properties.
Out_BrightImg = bwpropfilt(Out_BrightImg,'Area',[5000 + eps(5000), Inf]);

% Get properties.
properties = regionprops(Out_BrightImg, {'Area', 'Eccentricity', 'EquivDiameter',↙
'EulerNumber', 'MajorAxisLength', 'MinorAxisLength', 'Orientation', 'Perimeter'});

% Show the output image
%figure, imshow(Out_BrightImg);
```

```
%% Morphological Operations

r = 13;
SE = strel('diamond',r);

OpenedRegion = imopen(Out_BrightImg,SE);
figure,imshow(OpenedRegion);

%% Getting the tiny removed thread areas from the above morphological operations

I1 = Out_BrightImg-OpenedRegion;
%figure,imshow(I1);
[pixelCount,grayLevels]= imhist(I1);
Threshold = otsuthresh(pixelCount);
BI1 = imbinarize(I1,Threshold);
%figure, imshow(BI1);

%% Coverting the above image into a binary image and remove any small areas
BW_out = BI1;

BW_out = imclearborder(BW_out);


BW_out = imfill(BW_out, 'holes');


BW_out = bwpropfilt(BW_out,'Area',[-Inf, 250 - eps(250)]);

properties = regionprops(BW_out, {'Area', 'Eccentricity', 'EquivDiameter',↙
'EulerNumber', 'MajorAxisLength', 'MinorAxisLength', 'Orientation', 'Perimeter'});

figure,imshow(BW_out);

%% Getting the clear image without any noise  and obtaining only the threaded↙
region.

%figure,imshow(OpenedRegion);
CloseImg = OpenedRegion+BW_out;
%figure,imshow(CloseImg);

[pixelCount,grayLevels]= imhist(CloseImg);
Threshold = otsuthresh(pixelCount);
CloseLogicImg = imbinarize(CloseImg,Threshold);
figure,imshow(CloseLogicImg);

ClearImg = CloseLogicImg;
ClearImg = imclearborder(ClearImg);
ClearImg= imfill(ClearImg, 'holes');
ClearImg = bwpropfilt(ClearImg,'Area',[112295 + eps(112295), Inf]);
properties = regionprops(ClearImg, {'Area', 'Eccentricity', 'EquivDiameter',↙
'EulerNumber', 'MajorAxisLength', 'MinorAxisLength', 'Orientation', 'Perimeter'});
%figure,imshow(ClearImg);

%% Remove small areas using Regional properties
```

```
SeperateImg = ClearImg;

% Remove portions of the image that touch an outside edge.
SeperateImg = imclearborder(SeperateImg);

% Fill holes in regions.
SeperateImg = imfill(SeperateImg, 'holes');

% Filter image based on image properties.
SeperateImg = bwpropfilt(SeperateImg,'Area',[2000 + eps(2000), Inf]);

% Get properties.
properties = regionprops(SeperateImg, {'Area', 'Eccentricity', 'EquivDiameter',↙
'EulerNumber', 'MajorAxisLength', 'MinorAxisLength', 'Orientation', 'Perimeter'});

%figure, imshow(SeperateImg);

[Boundary,Location] = bwboundaries(SeperateImg);


%% Edge detection

% Threshold = 0.4;
% %BW = edge(SeperateImg,"canny", Threshold,"both","nothinning");
% EdgeScrewImg = edge(SeperateImg,'canny',Threshold);
% [r, c] = find(EdgeScrewImg); % Edge point coordinates


%% Removing the top head of the screw image

ThreadImg = SeperateImg;
ThreadImg = imclearborder(ThreadImg);
ThreadImg = imfill(ThreadImg, 'holes');
ThreadImg = bwpropfilt(ThreadImg,'Area',[112600 + eps(112600), Inf]);
properties = regionprops(ThreadImg, {'Area', 'Eccentricity', 'EquivDiameter',↙
'EulerNumber', 'MajorAxisLength', 'MinorAxisLength', 'Orientation', 'Perimeter'});

%% Apply Median Filter to filer the noise in the image
FThreadImg = medfilt2(ThreadImg,[8,8]);
%figure;imshow(FThreadImg)


%% Hugh circle detection

[centersIn,radiiIn] = imfindcircles(FThreadImg,[5 10], "Method","TwoStage","↙
EdgeThreshold",0.4,"Sensitivity",0.85);
[centersOut,radiiOut] = imfindcircles(FThreadImg,[5 10], "Method","TwoStage","↙
EdgeThreshold",0.4,"Sensitivity",0.85,"ObjectPolarity","dark");
figure,imshow(FThreadImg), hold on
viscircles(centersIn, radiiIn);
viscircles (centersOut, radiiOut);
len1 = size(centersIn,1);
len2 = size(centersOut,1);
```

```
for k = 1:len1
    plot(centersIn(k,1), centersIn(k,2), 'r+', 'MarkerSize', 1, 'LineWidth', 2);
end

for k = 1:len2
    plot(centersOut(k,1), centersOut(k,2), 'w+', 'MarkerSize', 1, 'LineWidth', 2);
end

%% Hugh line detection
%
sv = strel('diamond',5);
OpenThreadImg = imopen(ThreadImg,sv);
 Threadedge = edge(ThreadImg,'sobel',0.04,'both');

[H, Theta, Rho] = hough(Threadedge, 'Theta', -90.00:6.00:84.00, 'RhoResolution',↙
1.00);
Peaks = houghpeaks(H, 200, 'threshold', 0.50, 'NHoodSize', [11 11]);
lines = houghlines(Threadedge, Theta, Rho, Peaks, 'FillGap', 8, 'MinLength', 15);

% VISUALIZATION:
figure;
imshow(Threadedge);
hold on;
for ii = 1:length(lines)
    xy = [lines(ii).point1; lines(ii).point2];
    plot(xy(:, 1), xy(:, 2), 'LineWidth', 2, 'Color', 'green');
end

for id = 1:length(lines)
dist_space(id,1) =sqrt((lines(id).point1(1)-lines(id).point2(1))^2+(lines(id).point1↙
(2)-lines(id).point2(2))^2);
dist_space(id,2)= abs(lines(id).theta);
    if (dist_space(id,1)<20 || dist_space(id,1)>45)||(dist_space(id,2)<50 ||↙
dist_space(id,2)>65)
        dist_space(id,3)=0;
    else
        dist_space(id,3)= 1;
    end
end
c2 = 1;
for id = 1:length(lines)
if dist_space(id,3)==1
    meanAngle(c2,1)= dist_space(id,2);
    c2 = c2+1;
end
end
modeAngle = mode(meanAngle);
% if modeAngle<57.5
%     fprintf("Damaged Screw : %s " ,file1);
%
% end

%% Properties of the image and know the scaling factor
```

```
%Uncomment this so that you could measure in pixels on the image
 %my_pixinfo_tool(SeperateImg);

 %% Scale factor for Telecentric lens

%  by using this code the scaling factor from the original object to image
%  formed from the telecentric lens is approx 0.03  mm/pixel
% Uncomment this section to measure the screw thread in pixels
% Note: Please run this following code in the end because it was having a
% function in it.
%  figure,imshow(SeperateImg)
%  roi = drawline('Color','r');
%
% addlistener(roi,'MovingROI',@allevents);
% addlistener(roi,'ROIMoved',@allevents);
%
% function allevents(src,evt)
%     evname = evt.EventName;
%     switch(evname)
%         case{'MovingROI'}
%             disp(['ROI moving previous position: ' mat2str(evt.↙
PreviousPosition)]);
%             disp(['ROI moving current position: ' mat2str(evt.CurrentPosition)]);
%         case{'ROIMoved'}
%             disp(['ROI moved previous position: ' mat2str(evt.PreviousPosition)]);
%             disp(['ROI moved current position: ' mat2str(evt.CurrentPosition)]);
%     end
% end

%% Detection of screw Depth parameters

%finding depth in the right side of the screw

len1 = size(centersIn,1);
len2 = size(centersOut,1);
t1 = 1;
t2 = 1;

for k = 1 : len1          % Loop through all blobs.
    % Place the blob label number at the centroid of the blob.
    text(centersIn(k,1), centersIn(k,2), num2str(k), 'FontSize', 10, 'FontWeight',↙
'Bold', 'HorizontalAlignment', 'right', 'VerticalAlignment', 'cap','Color','k');
end
for k = 1 : len2         % Loop through all blobs.
    % Place the blob label number at the centroid of the blob.
    text(centersOut(k,1), centersOut(k,2), num2str(k), 'FontSize', 10, 'FontWeight',↙
'Bold', 'HorizontalAlignment', 'left', 'VerticalAlignment', 'cap','Color','w');
end
p_min = min(centersIn(:,1));
p_max = max(centersIn(:,1));
p_mean = (p_max+p_min)/2;

% for right side of the screw thread
u1 = 1;
```

```
for k = 1:len2
     if centersOut(k,1)>p_mean
        xOut(u1,1) = centersOut(k,1);
        yOut(u1,1) = centersOut(k,2);
        u1=u1+1;
     end
end
u2 = 1;
for k = 1:len1
     if centersIn(k,1)>p_mean
        xIn(u2,1) = centersIn(k,1);
        yIn(u2,1) = centersIn(k,2);
        u2=u2+1;
     end
end

for k = 1:size(yOut,1)
    for i1 = 1:size(yIn,1)
    s_y(i1,k)= yIn(i1,1)-yOut(k,1);
    end
end

for m = 1:size(s_y,2)
e1 = 1;
d_y = [];
for k = 1:size(s_y,1)
    if s_y(k,m)>0
        d_y(e1,1) = (s_y(k,m));
         e1 = e1+1;
    end
end
short_dist1(m,1) =min(d_y);
end

nearest_pt = yOut+short_dist1;

for i  = 1:size(nearest_pt,1)
nearest_pt(i,2) = find(centersIn(:,2) == nearest_pt(i,1));
sn = nearest_pt(i,2);
nearest_pt(i,3) = centersIn(sn,1);
nearest_pt(i,4) = radiiIn(sn,1);
end

for i  = 1:size(nearest_pt,1)
nearest_pt(i,5) = yOut(i,1);
end

for i  = 1:size(nearest_pt,1)
nearest_pt(i,6) = find(centersOut(:,2) == nearest_pt(i,5));
 sn = nearest_pt(i,6);
 nearest_pt(i,7) = centersOut(sn,1);
 nearest_pt(i,8) = radiiOut(sn,1);
end
```

```
% screw depth at the right side of the image is

depth_right = nearest_pt(:,4)+nearest_pt(:,8)+(nearest_pt(:,3)-nearest_pt(:,7));


%finding depth in the left side of the screw

%for left side of the screw

u1 = 1;
for k = 1:len2
     if centersOut(k,1)<p_mean
        xOut2(u1,1) = centersOut(k,1);
        yOut2(u1,1) = centersOut(k,2);
        u1=u1+1;
     end
end
u2 = 1;
for k = 1:len1
     if centersIn(k,1)<p_mean
        xIn2(u2,1) = centersIn(k,1);
        yIn2(u2,1) = centersIn(k,2);
        u2=u2+1;
     end
end

for k = 1:size(yOut2,1)
    for i1 = 1:size(yIn2,1)
    s_y2(i1,k)= yIn2(i1,1)-yOut2(k,1);
    end
end

for m = 1:size(s_y2,2)
e1 = 1;
d_y2 = [];
for k = 1:size(s_y2,1)
    if s_y2(k,m)>0
        d_y2(e1,1) = (s_y2(k,m));
         e1 = e1+1;
    end
end
short_dist2(m,1) =min(d_y2);
end

nearest_pt2 = yOut2+short_dist2;

for i  = 1:size(nearest_pt2,1)
nearest_pt2(i,2) = find(centersIn(:,2) == nearest_pt2(i,1));
sn = nearest_pt2(i,2);
nearest_pt2(i,3) = centersIn(sn,1);
nearest_pt2(i,4) = radiiIn(sn,1);
end

for i  = 1:size(nearest_pt2,1)
```

```
nearest_pt2(i,5) = yOut2(i,1);
end

for i  = 1:size(nearest_pt2,1)
nearest_pt2(i,6) = find(centersOut(:,2) == nearest_pt2(i,5));
 sn = nearest_pt2(i,6);
 nearest_pt2(i,7) = centersOut(sn,1);
 nearest_pt2(i,8) = radiiOut(sn,1);
end

% screw depth at the leftt side of the image is

depth_left = nearest_pt2(:,4)+nearest_pt2(:,8)+(nearest_pt2(:,7)-nearest_pt2(:,3));


%% Pitch of a screw

% finding pitch on the right side of the image

right_sort = sort(nearest_pt(:,1));
for k = 1:size(nearest_pt,1)-1
 rightPitch(k,1) = right_sort(k+1,1)-right_sort(k,1);
end

% finding pitch on the left side of the image

left_sort = sort(nearest_pt(:,1));

for k = 1:size(nearest_pt,1)-1
leftPitch(k,1) = left_sort(k+1,1)-left_sort(k,1);
end

MeanPitch_right = mean(rightPitch);
MeanPitch_left = mean(leftPitch);

% As we can deduce from the table that rightPitch is equal to leftPitch for
% a good bolt and may not be equal for a bad one


%% Screw length

FSeperateImg = medfilt2(CloseLogicImg,[15,15]);
[Threadedge1, threshOut] = edge(FSeperateImg, 'Canny', [], 1.00);

%figure, imshow(Threadedge1);

[H1, T1, R1] = hough(Threadedge1, 'Theta', -90.00:11.00:79.00, 'RhoResolution',↙
9.88);
P1 = houghpeaks(H1, 15, 'threshold', 0.50, 'NHoodSize', [5 1]);
lines1 = houghlines(Threadedge1, T1, R1, P1, 'FillGap', 20, 'MinLength', 5);

figure;
imshow(Threadedge1);
hold on;
```

```
for jj = 1:length(lines1)
    xy1 = [lines1(jj).point1; lines1(jj).point2];
    plot(xy1(:, 1), xy1(:, 2), 'LineWidth', 2, 'Color', 'green');
end

TotalLength = transpose({lines1(:).point1}) ;
for k = 1:size(TotalLength,1)
yValue(k,1) = TotalLength{k}(2);
end

yTop = min(yValue);
yBottom = max(yValue);
screwLength = abs(yTop-yBottom);


%% Pitch Diameter of a Screw

% to find the distance form the crest to trough of a right side of a screw
%length_right(:,1) = nearest_pt(:,3)+nearest_pt(:,4)+(abs(nearest_pt(:,3)-nearest_pt↙
(:,7)))+nearest_pt(:,7)-nearest_pt(:,8);
length_right(:,1)= depth_right(:,1);
% using the data sheet of metric screw mentioned that 5H/8 = teeth distance
%  and 3H/8  from the top therefore for getting the pitch diameter point we could↙
just multiply with
% 3/5 to the respective depth to get the pitch point position.

length_right(:,2) = (3*length_right(:,1))/5;

% determining the pitch point from the right end corner point of the screw

length_right(:,3) = (nearest_pt(:,3)+nearest_pt(:,4))-length_right(:,2);

% to find the distance from the crest to trough of a left side of a screw
length_left(:,1) = depth_left(:,1);
% using the data sheet of metric screw mentioned that 5H/8 = teeth distance
% therefor for getting the pitch diameter point we could just multiply with
% 3/5 to the respective depth to get the pitch point position.
length_left(:,2) = (3*length_left(:,1))/5;

% determining the pitch point from the right end corner point of the screw

length_left(:,3) = (nearest_pt2(:,3)+nearest_pt2(:,4))-length_left(:,2);

%The pitch diameter of the screw
pitchDiameter = abs(mean(length_left(:,3))-mean(length_right(:,3)));

%% Final results:

% here the obtained values are conved from pixels to mm using the scale
% factor  that is approxmately 0.03 mm/pixel

scaleFactor = 0.03 ;

% finding the actual thread angle parameters in "degrees"
```

```
if modeAngle<57.5
    fprintf("Damaged Screw : %s " ,file1);
end
Actual_Threadangle = modeAngle;

% finding the actual thread depth parameters in "mm"
totalDepth = [depth_right;depth_left];
da = 0;
q1 = 0;
for i = 1:length(totalDepth)
    if totalDepth(i,1)>26 % make sure that the depth is greather than this value
        q1 = q1+1;
        Actual_ThreadDepth(q1,1) = totalDepth(i,1)*scaleFactor;
    else
        da = da+1;
    end
end

if da > 7
 fprintf("Damaged Screw : %s " ,file1);
end

% finding the screw thread pitch parameter in "mm"

total_pitch = [rightPitch;leftPitch];
da = 0;
q1 = 0;
for i = 1:length(total_pitch)
    if 40<total_pitch(i,1)<62 % make sure that the pitch is in between these pixel↙
values
        q1 = q1+1;
        Actual_ThreadPitch(q1,1) = total_pitch(i,1)*scaleFactor;
    else
        da = da+1;
    end
end

if da > 1
 fprintf("Damaged Screw : %s " ,file1);
end

%finding the Total screw length in "mm"

Actual_TotalScrewlength = screwLength*scaleFactor;

%finding the Pitch diameter of the screw in "mm"

Actual_TotalPitchDiameter = pitchDiameter*scaleFactor;


% Creating a structure for the Screw parameters
% Note: all the parameters are in mm only except thread angle which is in
% degrees
Screw_threadParam.Note = "All the values are in milli meters except thread angles↙
```

```
those are in degrees";
Screw_threadParam.Thread_Angles = Actual_Threadangle;
Screw_threadParam.Thread_Depth = Actual_ThreadDepth;
Screw_threadParam.Thread_Pitch= Actual_ThreadPitch;
Screw_threadParam.Total_ScrewLength = Actual_TotalScrewlength;
Screw_threadParam.Total_PitchDiameter = Actual_TotalPitchDiameter;
```