```
close all
clear all
% Read the input image
img = imread('image1.jpeg');


% Convert the image to grayscale
img_gray = rgb2gray(img);
histogram = hist(img_gray);

figure, imshow(img_gray);
title('Gray Scale Image');

%getting the size of an image
sz = size(img_gray);

%PAD THE MATRIX WITH ZEROS ON ALL SIDES
padded_image=zeros(size(img_gray)+2);

%COPY THE ORIGINAL IMAGE MATRIX TO THE PADDED MATRIX
for x=1:size(img_gray,1)
    for y=1:size(img_gray,2)
        padded_image(x+1,y+1)=img_gray(x,y);
    end
end
%% Calling either mean, median or Gaussian filter
% %calling median filter
% output_med_image = med_filter(padded_image);
% figure, imshow(output_med_image);
% title('IMAGE AFTER MEDIAN FILTERING');

% %calling mean filter
% output_mean_image = meanfilter(padded_image);
% figure, imshow(output_mean_image);
% title('Image After Mean Filtering')

%calling gausian filter
output_gaus_image = gausFiltering(img_gray);
figure, imshow(output_gaus_image);
title('Image after Gausian Filtering');
histogram = hist(output_gaus_image);
%% Calling Thresholding
% binarized_image = binimage(output_gaus_image);
% figure, imshow(binarized_image);
% title('Binarized image')

%% Calling opening and closing
% %calling closing for mean image
% I_Final_closing = closing (binarized_image);
% figure, imshow(I_Final_closing);
% title('Closing Image')
```

```
% %calling closing for median image
% I_Final = closing (output_gaus_image);
% figure, imshow(I_Final);
% title('Closing Image')

%calling closing for mean image
I_Final_opening = opening (output_gaus_image);
figure, imshow(I_Final_opening);
title('Opening Image');
histogram = hist(I_Final_opening);
%% Calling Thresholding
binarized_image = binimage(I_Final_opening);
figure, imshow(binarized_image);
title('Binarized image')
histogram = hist(binarized_image);
%% Calling Edge Detection Function
% edge_detected_image = edi(binarized_image);
% figure, imshow(edge_detected_image);
% title('Edge Detected Of The Image')

% Apply the Canny edge detection algorithm to detect edges
edge_detected_image = edge(binarized_image, 'Canny');
figure, imshow(edge_detected_image);
title('Edge Detected Of The Image')
histogram = hist(edge_detected_image);
%%
% Find the boundaries of the license plate
boundaries = bwboundaries(edge_detected_image);

% Choose the boundary with the largest area
max_area = 0;
max_idx = 0;
for i = 1:length(boundaries)
    boundary = boundaries{i};
    area = polyarea(boundary(:, 2), boundary(:, 1));
    if area > max_area
        max_area = area;
        max_idx = i;
    end
end
license_plate_boundary = boundaries{max_idx};

% Display the license plate with the boundary overlaid
figure, imshow(binarized_image);
hold on;
plot(license_plate_boundary(:, 2), license_plate_boundary(:, 1), 'g','LineWidth', 4);
title('Boundary detected of license plate with green color')

% Define the bounding box for the license plate
x_min = min(license_plate_boundary(:, 2));
```

```
x_max = max(license_plate_boundary(:, 2));
y_min = min(license_plate_boundary(:, 1));
y_max = max(license_plate_boundary(:, 1));

% Crop the license plate region
license_plate = binarized_image(y_min:y_max, x_min:x_max, :);

% Display the cropped license plate
figure, imshow(license_plate);
title('License Plate')

%%
% % %segemetating the characters in license plate
% chars = segment_license_plate_characters(license_plate);
%
% % Display the first character image
% imshow(chars{100});
%%
%
% Create a larger structuring element
se = strel('square', 4);

% Perform erosion with the larger structuring element
eroded_license_plate = imdilate(license_plate, se);
imshow(eroded_license_plate);
% performing OCR in the extracted license plate
ocr_results = ocr(eroded_license_plate,↙
CharacterSet='ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789');
%% median filtering starts here
function output_med_image = med_filter(padded_image)
output_med_image = zeros(size(padded_image));
    for i= 1:size(padded_image,1)-2
        for j=1:size(padded_image,2)-2
            window=zeros(9,1);
            inc=1;
            for x=1:3
                for y=1:3
                    window(inc)=padded_image(i+x-1,j+y-1);
                    inc=inc+1;
                end
            end

            med=sort(window);
            %PLACE THE MEDIAN ELEMENT IN THE OUTPUT MATRIX
            output_med_image(i+1,j+1)=med(5);

        end
    end

    %CONVERT THE OUTPUT MATRIX TO 0-255 RANGE IMAGE TYPE
    output_med_image=uint8(output_med_image);
```

```
%     title('IMAGE AFTER MEDIAN FILTERING');
%     figure,imshow(output_image);
  end
%% mean filtering starts here
% function for meanfiltering
function output_mean_image = meanfilter(padded_image)
output_mean_image = zeros(size(padded_image));
for i= 1:size(padded_image,1)-2
        for j=1:size(padded_image,2)-2
            window=zeros(9,1);
            inc=1;
            for x=1:3
                for y=1:3
                    window(inc)=padded_image(i+x-1,j+y-1);
                    inc=inc+1;
                end
            end
            men = mean(window);
            %PLACE THE MEAN ELEMENT IN THE OUTPUT MATRIX
            output_mean_image(i+1,j+1)= men;
        end
        output_mean_image=uint8(output_mean_image);
end
end
%% Gaussian filtering
function output_gaus_image = gausFiltering(img_gray)
I = double(img_gray);
sigma = sqrt(3);

sz = 3;
[x,y]=meshgrid(-sz:sz,-sz:sz);

M = size(x,1)-1;
N = size(y,1)-1;

Exp_comp = -(x.^2+y.^2)/(2*sigma*sigma);

kernal_mat = exp(Exp_comp);
min_val = min(min(kernal_mat));

kernal_mat = round(kernal_mat / min_val);
sok = (size(kernal_mat,1)*size(kernal_mat,2));

window = zeros(sok, 1);
inc = 1;

for i = 1:size(kernal_mat, 1)
    for j = 1:size(kernal_mat, 2)
        window(inc) = kernal_mat(i,j);
        inc = inc+1;
    end
```

```
end

denom = sum(window);

output_gaus_image=zeros(size(I));

padded_image=zeros(size(I)+6);

%COPY THE ORIGINAL IMAGE MATRIX TO THE PADDED MATRIX
for x=1:size(I,1)
    for y=1:size(I,2)
        padded_image(x+3,y+3)=I(x,y);
    end
end

for i = 1:size(padded_image,1)-M
    for j =1:size(padded_image,2)-N
        Temp = padded_image(i:i+M,j:j+M).*kernal_mat;
        output_gaus_image(i,j)=sum(Temp(:)) / denom ;
    end
end
%
output_gaus_image = uint8(output_gaus_image);
end
%% Thresholding
function binarized_image = binimage(output_gaus_image)
binarized_image = zeros(size(output_gaus_image));

for i = 1:size(output_gaus_image,1)
    for j = 1:size(output_gaus_image,2)
        if output_gaus_image(i,j) <= 127
            binarized_image(i,j) = 0;
        else
            binarized_image(i,j) = 255;
        end
    end
end

end

%% Closing
function I_final_closing = closing(output_gaus_image)
I = output_gaus_image;
S = [1 1 1; 1 1 1;1 1 1];

I_Temp = zeros(size(I));

I_final_closing = zeros(size(I));

h = size(I,1);
w = size(I,2);
```

```
for i = 1:h
    for j = 1:w
        if i == 1
            if j == 1
                I_Temp(i,j) = max(max(I(i:i+1,j:j+1)));
            elseif j == w
                I_Temp(i,j) = max(max(I(i:i+1,j-1:j)));
            else
                I_Temp(i,j) = max(max(I(i:i+1,j-1:j+1)));
            end
        elseif i == h
            if j == 1
                I_Temp(i,j) = max(max(I(i-1:i,j:j+1)));
            elseif j == w
                I_Temp(i,j) = max(max(I(i-1:i,j-1:j)));
            else
                I_Temp(i,j) = max(max(I(i-1:i,j-1:j+1)));
            end
        else
            if j == 1
                I_Temp(i,j) = max(max(I(i-1:i+1,j:j+1)));
            elseif j == w
                I_Temp(i,j) = max(max(I(i-1:i+1,j-1:j)));
            else
                I_Temp(i,j) = max(max(I(i-1:i+1,j-1:j+1)));
            end
        end
    end
end

for i = 1:h
    for j = 1:w
        if i == 1
            if j == 1
                I_final_closing(i,j) = min(min(I_Temp(i:i+1,j:j+1)));
            elseif j == w
                I_final_closing(i,j) = min(min(I_Temp(i:i+1,j-1:j)));
            else
                I_final_closing(i,j) = min(min(I_Temp(i:i+1,j-1:j+1)));
            end
        elseif i == h
            if j == 1
                I_final_closing(i,j) = min(min(I_Temp(i-1:i,j:j+1)));
            elseif j == w
                I_final_closing(i,j) = min(min(I_Temp(i-1:i,j-1:j)));
            else
                I_final_closing(i,j) = min(min(I_Temp(i-1:i,j-1:j+1)));
            end
        else
            if j == 1
```

```
                I_final_closing(i,j) = min(min(I_Temp(i-1:i+1,j:j+1)));
            elseif j == w
                I_final_closing(i,j) = min(min(I_Temp(i-1:i+1,j-1:j)));
            else
                I_final_closing(i,j) = min(min(I_Temp(i-1:i+1,j-1:j+1)));
            end
        end
    end
end


I_final_closing = uint8(I_final_closing);
% figure, imshow(I_Final);
% title('Erosion')
end
%% Opening the image

function I_final_opening = opening(image)
I = image;
S = [1 1 1; 1 1 1;1 1 1];

I_Temp = zeros(size(I));

I_final_opening = zeros(size(I));

h = size(I,1);
w = size(I,2);

for i = 1:h
    for j = 1:w
        if i == 1
            if j == 1
                I_Temp(i,j) = min(min(I(i:i+1,j:j+1)));
            elseif j == w
                I_Temp(i,j) = min(min(I(i:i+1,j-1:j)));
            else
                I_Temp(i,j) = min(min(I(i:i+1,j-1:j+1)));
            end
        elseif i == h
            if j == 1
                I_Temp(i,j) = min(min(I(i-1:i,j:j+1)));
            elseif j == w
                I_Temp(i,j) = min(min(I(i-1:i,j-1:j)));
            else
                I_Temp(i,j) = min(min(I(i-1:i,j-1:j+1)));
            end
        else
            if j == 1
                I_Temp(i,j) = min(min(I(i-1:i+1,j:j+1)));
            elseif j == w
                I_Temp(i,j) = min(min(I(i-1:i+1,j-1:j)));
```

```
            else
                I_Temp(i,j) = min(min(I(i-1:i+1,j-1:j+1)));
            end
        end
    end
end

for i = 1:h
    for j = 1:w
        if i == 1
            if j == 1
                I_final_opening(i,j) = max(max(I_Temp(i:i+1,j:j+1)));
            elseif j == w
                I_final_opening(i,j) = max(max(I_Temp(i:i+1,j-1:j)));
            else
                I_final_opening(i,j) = max(max(I_Temp(i:i+1,j-1:j+1)));
            end
        elseif i == h
            if j == 1
                I_final_opening(i,j) = max(max(I_Temp(i-1:i,j:j+1)));
            elseif j == w
                I_final_opening(i,j) = max(max(I_Temp(i-1:i,j-1:j)));
            else
                I_final_opening(i,j) = max(max(I_Temp(i-1:i,j-1:j+1)));
            end
        else
            if j == 1
                I_final_opening(i,j) = max(max(I_Temp(i-1:i+1,j:j+1)));
            elseif j == w
                I_final_opening(i,j) = max(max(I_Temp(i-1:i+1,j-1:j)));
            else
                I_final_opening(i,j) = max(max(I_Temp(i-1:i+1,j-1:j+1)));
            end
        end
    end
end


I_final_opening = uint8(I_final_opening);
% figure, imshow(I_Final);
% title('Erosion')
end

%% Edge Detection
function edge_detected_image = edi (image)
I = double(image);
In = I;
% m1=[1, 0, -1;1, 0, -1;1, 0, -1];
% m2=[1, 1, 1;0, 0, 0;-1, -1, -1];
% m3=[0, -1, -1;1, 0, -1;1, 1, 0];
% m4=[1, 1, 0;1, 0, -1;0, -1, -1];
```

```
m1=[-1, 0, 1;-1, 0, 1;-1, 0, 1];
m2=[-1, -1, -1;0, 0, 0;1, 1, 1];
m3=[-1, 0, 1;-1, 0, 1;-1, 0, 1];
m4=[-1, -1, -1;0, 0, 0;1, 1, 1];

% Flipping the mask horizontally & Vertically
m1=flipud(m1);
m1=fliplr(m1);
m2=flipud(m2);
m2=fliplr(m2);
m3=flipud(m3);
m3=fliplr(m3);
m4=flipud(m4);
m4=fliplr(m4);

for i=2:size(I, 1)-1
    for j=2:size(I, 2)-1
        neigh_matrix1=m1.*In(i-1:i+1, j-1:j+1);
        avg_val1=sum(neigh_matrix1(:));

        neigh_matrix2=m2.*In(i-1:i+1, j-1:j+1);
        avg_val2=sum(neigh_matrix2(:));

        neigh_matrix3=m3.*In(i-1:i+1, j-1:j+1);
        avg_val3=sum(neigh_matrix3(:));

        neigh_matrix4=m4.*In(i-1:i+1, j-1:j+1);
        avg_val4=sum(neigh_matrix4(:));

        %using max function for detection of final edges
        I(i, j)=max([avg_val1, avg_val2, avg_val3, avg_val4]);
    end
end
edge_detected_image = uint8(I);
end
%%
%Segmentation
function chars = segment_license_plate_characters(license_plate)
% This function takes in a binary license plate image and returns a cell array
% containing each segmented character as a separate image.

% Get the license plate dimensions
[~, width] = size(license_plate);

% Initialize variables for character bounding boxes and labels
char_bboxes = zeros(6, 4);
char_labels = cell(6, 1);

% Initialize a variable to keep track of the current character being segmented
char_idx = 1;
```

```
% Loop through each column of the license plate
for col = 1:width
    % Get the column of the license plate
    col_data = license_plate(:, col);

    % Get the regions in the column where the license plate is black (i.e., where a character is located)
    regions = regionprops(col_data, 'BoundingBox', 'Area');

    % If there are no regions in the column, move on to the next column
    if isempty(regions)
        continue
    end

    % Sort the regions by area, so that the largest regions (i.e., the characters) are first
    [~, idx] = sort([regions.Area], 'descend');
    regions = regions(idx);

    % Loop through each region in the column
    for j = 1:numel(regions)
        % Get the bounding box for the region
        bbox = regions(j).BoundingBox;

        % Check if the bounding box is taller than it is wide (i.e., it is a character)
        if bbox(4) > bbox(3)
            % Add the character's bounding box and label to the list of characters
            char_bboxes(char_idx, :) = bbox;
            char_labels{char_idx} = '';

            % Increment the character index
            char_idx = char_idx + 1;
        end
    end
end

% Crop each character from the license plate and save it as a separate image
chars = cell(size(char_bboxes, 1), 1);
for i = 1:size(char_bboxes, 1)
    % Get the bounding box for the current character
    bbox = char_bboxes(i, :);

    % Crop the character from the license plate
    char_img = imcrop(license_plate, bbox);

    % Add the character image to the cell array
    chars{i} = char_img;
end
end
%%
```

```
%Creating histogram
% Compute the histogram
function histogram = hist(image)
histogram = imhist(image);

% Display the histogram
figure, bar(histogram);
xlabel('Intensity Value');
ylabel('Count');
title('Histogram of Image');
end
```