Signals and Systems - DR.Akhavan

CA2 - Matin Bazrafshan

Part 1 - English Number Plate Detection:

1.1 Read Date

```
[filename, filepath] = uigetfile({'*.jpg;*.png;*.bmp', 'Image Files (*.jpg, *.png, *.bmp)'}, 'S
imagepath = fullfile(filepath, filename);
image = imread(imagepath);
```

1.2 Resize

At the beginning we need to resize images to a specific size, so we can do preprocessing better and in a more efficient way.

```
image = imresize(image, [300,500]);
figure;
imshow(image);
```

1.3 To desaturate image

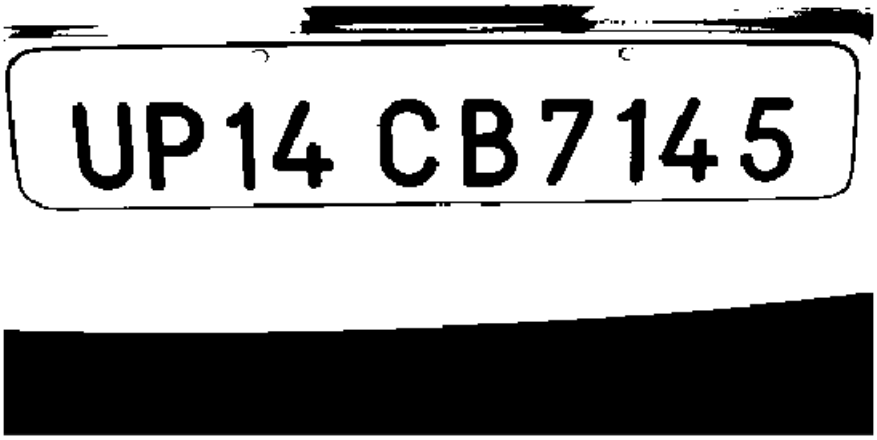
Colors can not help to detect chrachter so we convert our image to grayscale.

```
gray_image = mygrayfun(image);
figure;
imshow(gray_image);
```

1.4 To binarize image

To reduce our model complexity, by setting a threshold, we map our image's pixels to 0 and1.

```
bianry_image = mybinaryfun(gray_image, 100);
figure;
imshow(bianry_image);
```

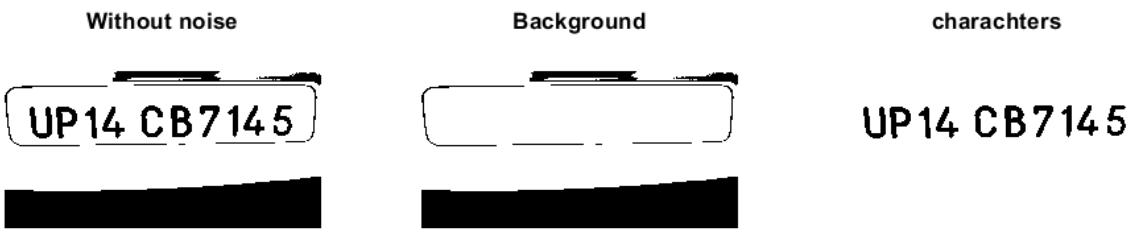


1.5 Remove noises and background

The idea of isolating charachters comes up with filtering our image two times.

1. The first filtering, removes noises(the components with less than pixel comparing to a specific threshold)
2. The second filtering removes all components with larger size, but it keeps very large components that we expect them as background elements.

```
filtered = myremovecom(~bianry_image, 300);
background = myremovecom(~bianry_image, 2300);
charachters = (filtered - background);

figure('Position', [0 0 900 400]);

subplot(1,3,1);
imshow(~filtered);
title('Without noise');

subplot(1,3,2);
imshow(~background);
title('Background');

subplot(1,3,3);
imshow(~charachters);
title('charachters');
```



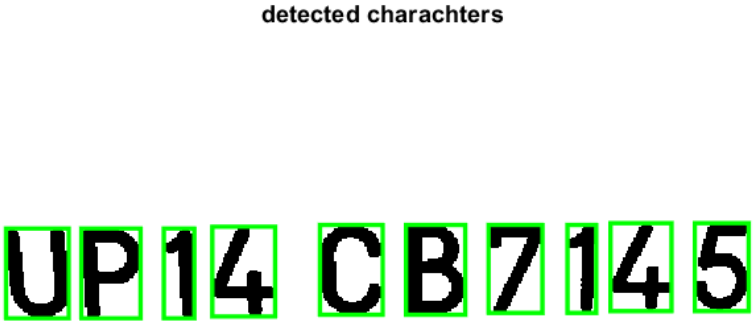
1.6 Detect segments

Now that we extract charchters, we need to label them correctly, there is a little trick that we need to hover over image in way of colomn inrease, so we need to check a column, lavel the pixels correctly and then go to the next colimn, this approach will result in label segements from left to write, so in the next part when we want to template match those pixels, we have the segements in the correct order.

```
[n, labeled] = mysegmentation(charachters);
fprintf("Number of segments: %d", n);
```

```
Number of segments: 10
```

```
figure('Position', [0 0 900 400]);
propied=regionprops(labeled,'BoundingBox');
imshow(~charachters);
title('detected charachters')
hold on
for m=1:size(propied,1)
    rectangle('Position',propied(m).BoundingBox,'EdgeColor','g','LineWidth',2)
end
hold off
```



1.7 To import our dataset

```
folderPath = './EnglishMapSet';
fileNames = dir(fullfile(folderPath, '*.bmp'));
numFiles = length(fileNames);

imageData = cell(numFiles, 2);

for i = 1:numFiles
    imagePath = fullfile(folderPath, fileNames(i).name);
    image = imread(imagePath);

    [~, fileName, ~] = fileparts(fileNames(i).name);
    imageData{i, 1} = image;
    imageData{i, 2} = fileName;
end
```

1.8 Template matching using correlation

now that we have extracted segments correclty, we need to check what is its label, to do this we first need to resize the segment to the standard size of our dataset -which in this case is [42,24], after that wee need to calculate cross-correlation of these two pictures. The formula is written, but built-in MATLAB function is used.

$$corr2(A,B) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(A(i,j)-\bar{A})(B(i,j)-\bar{B})}{\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(A(i,j)-\bar{A})^2}\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(B(i,j)-\bar{B})^2}}$$

$(\bar{A})and(\bar{B})$ are means of A and B respectively.

The correlation measures the similarity between the template and the corresponding region in the input image. By comparing the correlation measures we can find what is the label of our picture.

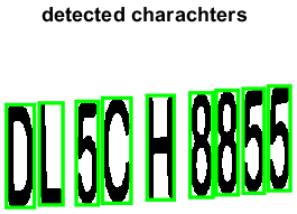
```
res = '';
for i = 1:n
    [row, col] = find(labeled == i);
    y = charachters(min(row):max(row), min(col):max(col));
    y = imresize(y,[42,24]);
    corr_max = -2;
    i_max = 0;
    for j = 1:numFiles
        corr_j = corr2(y,imageData{j,1});
        if(corr_j > corr_max)
            corr_max = corr_j;
            i_max = j;
        end
    end
    if(corr_max > 0.3)
        res = strcat(res,imageData{i_max,2});
    end
end
fprintf("Estimated plate number: %s", res);
```

```
Estimated plate number: UP14CB7145
```

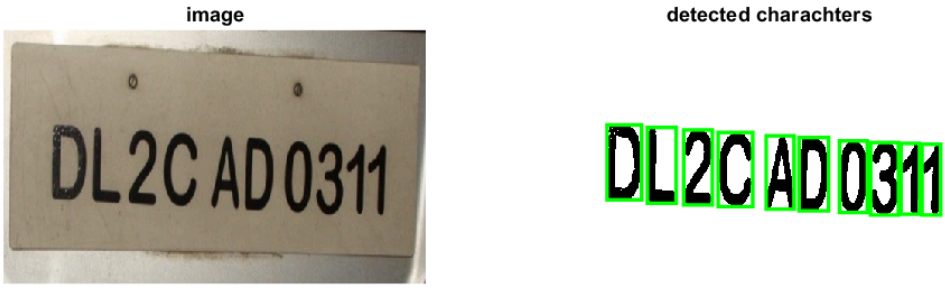
1.9 Test our model

```
tests = ['EnglishTest/1.jpg'; 'EnglishTest/2.jpg'; 'EnglishTest/3.jpg'];
for k = 1:3

    image = imread(tests(k,:));
    image = imresize(image, [300,500]);
    figure('Position', [0 0 900 400]);
    subplot(1,2,1);
    imshow(image);
    title('image');

    gray_image = mygrayfun(image);
    bianry_image = mybinaryfun(gray_image, 100);
    filtered = myremovecom(~bianry_image, 300);
    background = myremovecom(~bianry_image, 2300);
    charachters = (filtered - background);
    [n, labeled] = mysegmentation(charachters);

    propied=regionprops(labeled,'BoundingBox');
    subplot(1,2,2);
    imshow(~charachters);
    title('detected charachters');
    hold on
    for m=1:size(propied,1)
        rectangle('Position',propied(m).BoundingBox,'EdgeColor','g','LineWidth',2)
    end
    hold off
```

```
    res = '';
    for i = 1:n
        [row, col] = find(labeled == i);
        y = charachters(min(row):max(row), min(col):max(col));
        y = imresize(y,[42,24]);
        corr_max = -inf;
        i_max = 0;
        for j = 1:numFiles
            corr_j = corr2(y,imageData{j,1});
            if(corr_j > corr_max)
                corr_max = corr_j;
                i_max = j;
            end
        end
        if(corr_max > 0.3)
            res = strcat(res,imageData{i_max,2});
        end
    end
    fprintf("Estimated plate number: %s", res);
end
```




```
Estimated plate number: DL5CH8855
```




```
Estimated plate number: DL2CAD0311
```


```
Estimated plate number: UP14CB7145
```

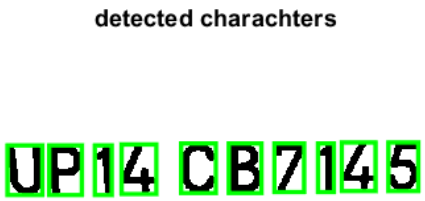
1.10 Additional Testing using Naive Bayes

Naive bayes is a simple machine learning algorithm which can detect and classify data using bayes theorem, the idea of naive bayes is to add a simpification to the problem that every pixel is independent to other by knowing the label of image, which is a complexity downgrade of what is a real pixel works. According to the bayes theorem and conditional probabality, we can implement it in MATLAB, look at the formula:

$$P(label|pixels) = \frac{P(label) \cdot P(pixels|label)}{P(pixels)}$$

```
res = '';
for i = 1:n
    [row, col] = find(labeled == i);
    y = charachters(min(row):max(row), min(col):max(col));
    y = imresize(y,[42,24]);
    p_max = -inf;
    i_max = 0;
```

```
    for j = 1:numFiles
        p_j = naive_bayes(imageData{j,1}, y);
        if(p_j > p_max)
            p_max = p_j;
            i_max = j;
        end
    end
    res = strcat(res,imageData{i_max,2});
end
figure('Position', [0 0 900 400]);
subplot(1,2,1);
imshow(image);
propied=regionprops(labeled,'BoundingBox');
subplot(1,2,2);
imshow(~charachters);
title('detected charachters')
hold on
for m=1:size(propied,1)
    rectangle('Position',propied(m).BoundingBox,'EdgeColor','g','LineWidth',2)
end
hold off
```




```
fprintf("Estimated plate number using naive bayes: %s", res);
```

```
Estimated plate number using naive bayes: UP14BB7145
```

Part 2: Persian Number Plate Detection

Please note that every step in part 2 is same as part 1, but some hyperparameters, like resizing ratio, threshold, etc is different.

2.1 Read Date

```
[filename, filepath] = uigetfile({'*.jpg;*.png;*.bmp', 'Image Files (*.jpg, *.png, *.bmp)'}, 'S
imagepath = fullfile(filepath, filename);
image = imread(imagepath);
```

2.2 Resize

```
image = imresize(image, [100,500]);
figure;
imshow(image);
```

2.3 To desaturate image

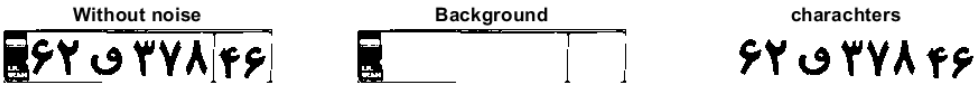
```
gray_image = mygrayfun(image);
figure;
imshow(gray_image);
```

2.4 To binarize image

```
bianry_image = mybinaryfun(gray_image, 100);
figure;
imshow(bianry_image);
```

2.5 Remove noises and background

```
filtered = myremovecom(~bianry_image, 300);
background = myremovecom(~bianry_image, 2300);
charachters = (filtered - background);

figure('Position', [0 0 900 400]);

subplot(1,3,1);
imshow(~filtered);
title('Without noise');

subplot(1,3,2);
imshow(~background);
title('Background');

subplot(1,3,3);
imshow(~charachters);
title('charachters');
```

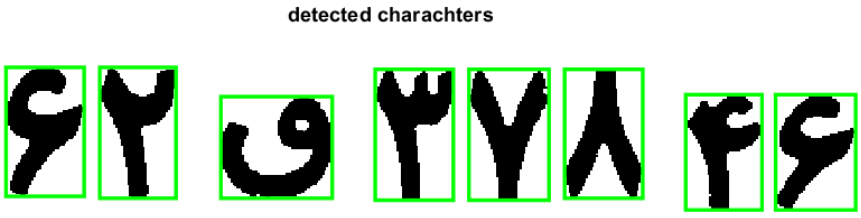


2.6 Detect segments

```
[n, labeled] = mysegmentation(charachters);
fprintf("Number of segments: %d", n);
```

```
Number of segments: 8
```

```
figure('Position', [0 0 900 400]);
propied=regionprops(labeled,'BoundingBox');
imshow(~charachters);
title('detected charachters')
hold on
for m=1:size(propied,1)
    rectangle('Position',propied(m).BoundingBox,'EdgeColor','g','LineWidth',2)
end
hold off
```



2.7 To import our dataset

```
folderPath = './PersianMapSet';
fileNames = dir(fullfile(folderPath, '*.png'));
numFiles = length(fileNames);

imageData = cell(numFiles, 2);

for i = 1:numFiles
    imagePath = fullfile(folderPath, fileNames(i).name);
    image = imread(imagePath);

    [~, fileName, ~] = fileparts(fileNames(i).name);
    imageData{i, 1} = image;
    imageData{i, 2} = fileName;
end
```

2.8 Template matching using correlation

```
res = '';
for i = 1:n
    [row, col] = find(labeled == i);
    y = charachters(min(row):max(row), min(col):max(col));
    y = imresize(y,[100,80]);
    corr_max = -2;
    i_max = 0;
    for j = 1:numFiles
        corr_j = corr2(y,imageData{j,1});
        if(corr_j > corr_max)
            corr_max = corr_j;
            i_max = j;
        end
    end
    if(corr_max > 0.3)
        res = strcat(res,imageData{i_max,2});
    end
end
fprintf("Estimated plate number: %s", res);
```

```
Estimated plate number: 62GH37846
```

2.9 Test our model

```
filedest = "PersianTestPlate/";
for k = 2:8

    image = imread(filedest + k + ".jpg");
    image = imresize(image, [100,500]);
    figure('Position', [0 0 900 400]);
    subplot(1,2,1);
    imshow(image);
    title('image');
```

```
    gray_image = mygrayfun(image);
    bianry_image = mybinaryfun(gray_image, 100);
    filtered = myremovecom(~bianry_image, 300);
    background = myremovecom(~bianry_image, 2300);
    charachters = (filtered - background);
    [n, labeled] = mysegmentation(charachters);

    propied=regionprops(labeled,'BoundingBox');
    subplot(1,2,2);
    imshow(~charachters);
    title('detected charachters');
    hold on
    for m=1:size(propied,1)
        rectangle('Position',propied(m).BoundingBox,'EdgeColor','g','LineWidth',2)
    end
    hold off

    res = '';
    for i = 1:n
        [row, col] = find(labeled == i);
        y = charachters(min(row):max(row), min(col):max(col));
        y = imresize(y,[100,80]);
        corr_max = -inf;
        i_max = 0;
        for j = 1:numFiles
            corr_j = corr2(y,imageData{j,1});
            if(corr_j > corr_max)
                corr_max = corr_j;
                i_max = j;
            end
        end
        if(corr_max > 0.3)
            res = strcat(res,imageData{i_max,2});
        end
    end
    fprintf("Estimated plate number: %s", res);
end
```

image

detected charachters

Estimated plate number: 57DD91366

image

detected charachters

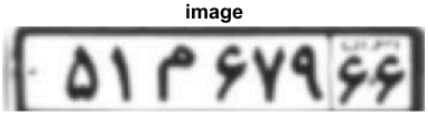
Estimated plate number: 47VV67457

image

detected charachters

Estimated plate number: 29BB5829

image



detected charachters

Estimated plate number: 51MM67966

image

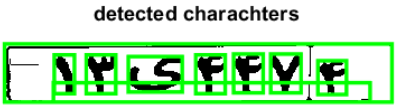
detected charachters

Estimated plate number: 59YY84410

image

detected charachters

Estimated plate number: 62GH37846

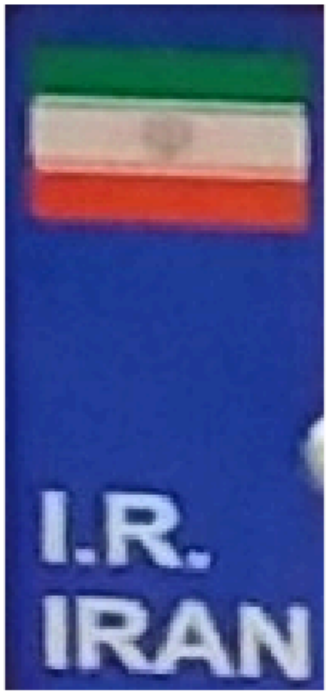



```
Estimated plate number: 13YY4474
```

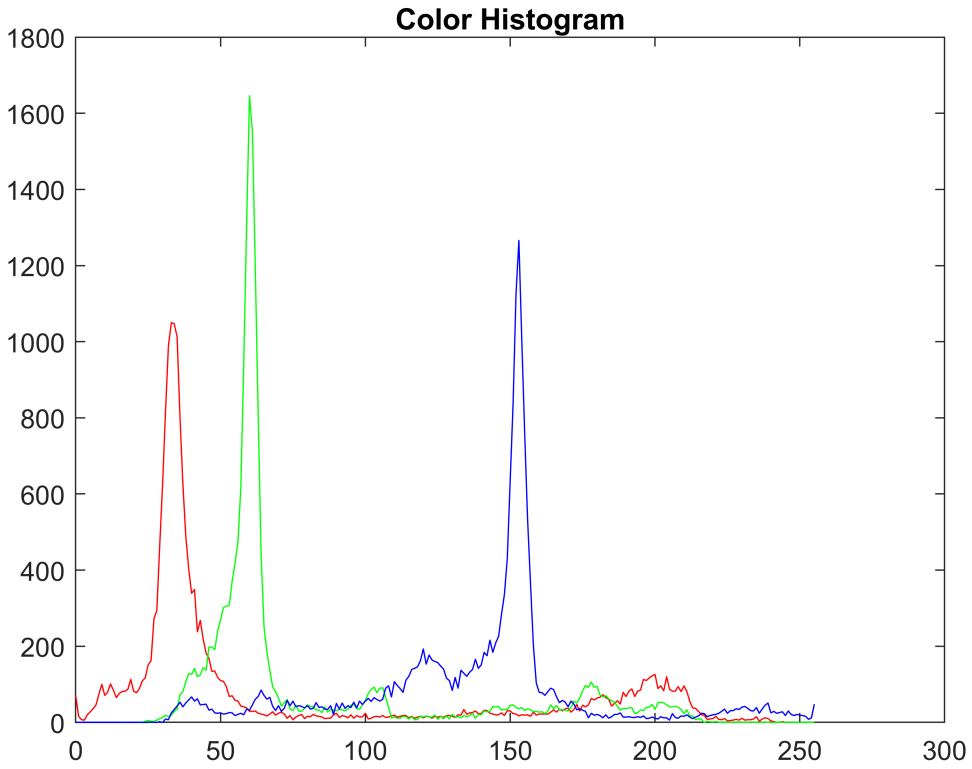
Part 3: Persian Licsense Plate Detection

The idea of this part is matching small blue part of IR Licsence plates with a template, first lets take a look at color histogram of this small but useful template:

```
template = imread('template.jpg');

figure();
imshow(template);
```




```
Red = template(:,:,1);
Green = template(:,:,2);
Blue = template(:,:,3);

[yRed, xr] = imhist(Red);
[yGreen, xg] = imhist(Green);
[yBlue, xb] = imhist(Blue);

figure;
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue, 'Blue')
title('Color Histogram')
```

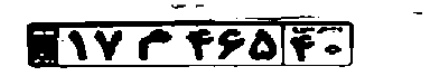


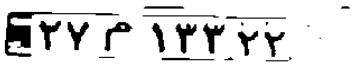
As you can see, color histogram of this template is also a unique histogram, we can also see this histogram pattern in a real picture:

now by having knowledge to the fact that this template is a unique pattern, we can use it a key of detection, used by template matching algorithm.

```
template = imread('origin.jpg');
filedest = "PersianTestCar/";
for k = 1:9
    image = imread(filedest + k + ".jpg");

    image = imresize(image, [200, 300]);
    template = imresize(template, [20, 10]);

    template = im2double(template);
    image = im2double(image);

    correlationMapR = normxcorr2(template(:,:,1), image(:,:,1));
    correlationMapG = normxcorr2(template(:,:,2), image(:,:,2));
    correlationMapB = normxcorr2(template(:,:,3), image(:,:,3));
    wR = 1;
    wG = 1;
    wB = 1;
    correlationMap = (wR * correlationMapR + wG * correlationMapG + wB * correlationMapB) / (wR

    [maxCorrelationValue, maxIndex] = max(correlationMap(:));
    [maxRow, maxCol] = ind2sub(size(correlationMap), maxIndex);
    templateHeight = size(template, 1);
    templateWidth = size(template, 2);
    matchedRegionRow = maxRow - templateHeight + 1;
    matchedRegionCol = maxCol - templateWidth + 1;


    figure('Position', [0 0 900 400]);
    subplot(1,2,1);
    imshow(image);
    hold on;
    rectangle('Position', [matchedRegionCol, matchedRegionRow, templateWidth, templateHeight],
    rectangle('Position', [matchedRegionCol-10, matchedRegionRow-10, 7 * templateHeight, 2 * te
    hold off;
    plate = imcrop(image, [matchedRegionCol-10, matchedRegionRow-10, 7 * templateHeight, 2 * te
    subplot(1,2,2);
    imshow(mybinaryfun(mygrayfun(imresize(plate, [100, 500])), 0.4));
%     fprintf("Estimated plate number: %s", detect_plate_persian(plate, imageData));
end
```


PEUGEOT
PARIS



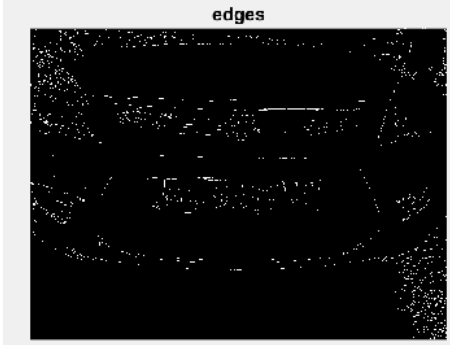


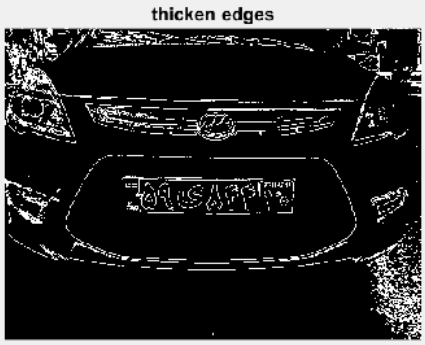
4. <u>Additional template detecton</u>: using edge detection

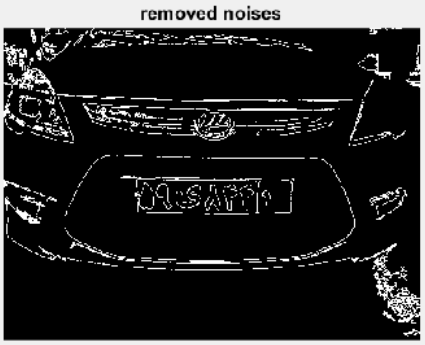
The algorithm has the following steps:

1. Preprocess the image (grayscale - binorize - remove noises)
2. Find edges of image
3. Scale the edges to have thicker edges
4. Remove noises
5. Find edges which makes a rectangle
6. Crop the rectangle
7. If the rectangle is too small(less then 400 pixels) or has not-plate ratio (I conside a plate has width / height ranges between [3,6])
8. Extract the elemets, if elements are more than 5, that is a plate(hopefully!)

- The code can be found in `edgeDetection.m`








```
TestDist = "PersianTestCar/";
Tests = [1, 7, 8, 9];
se = strel('rectangle', [5, 5]);

for k = 1:length(Tests)

    picture = imread(TestDist + Tests(k) + ".jpg");
    gray = rgb2gray(picture);
    bin = ~imbinarize(gray);

    edgeImage = edge(bin, 'Canny');

    dilatedImage = imdilate(edgeImage,se);

    filtered = bwareaopen(dilatedImage, 200);

    regions = regionprops(filtered, 'BoundingBox');

    for i = 1:length(regions)
        region = regions(i);
        picture_region = imcrop(picture, region.BoundingBox);
        w = round(region.BoundingBox(3));
        h = round(region.BoundingBox(4));
```

```
        if (w * h < 400 || w / h < 3 || w / h > 6)
            continue
        end
        image = imresize(picture_region,[100,500]);
        gray_image = mygrayfun(image);
        bianry_image = mybinaryfun(gray_image, 100);
        filtered = myremovecom(~bianry_image, 300);
        background = myremovecom(~bianry_image, 2300);
        charachters = (filtered - background);
        [n, labeled] = mysegmentation(charachters);
        if(n < 6)
            continue
        end
        figure('Position', [0 0 900 400]);
        subplot(1,2,1);
        imshow(picture);
        title('Car');
        subplot(1,2,2);
        imshow(image);
        title('detected plate');
    end
end
```

Car

detected plate

Car

detected plate

Car

detected plate

```
function [res] =  detect_plate_persian(image, imageData)
    image = imresize(image, [100, 500]);
    gray_image = mygrayfun(image);
    bianry_image = mybinaryfun(gray_image, 100/256);
    filtered = myremovecom(~bianry_image, 300);
    background = myremovecom(~bianry_image, 2300);
    charachters = (filtered - background);
    [n, labeled] = mysegmentation(charachters);

    res = '';
    for i = 1:n
        [row, col] = find(labeled == i);
        y = charachters(min(row):max(row), min(col):max(col));
        y = imresize(y,[100,80]);
        corr_max = -2;
        i_max = 0;
        for j = 1:size(imageData,1)
            corr_j = corr2(y,imageData{j,1});
            if(corr_j > corr_max)
                corr_max = corr_j;
                i_max = j;
            end
        end
        if(corr_max > 0.5)
            res = strcat(res,imageData{i_max,2});
        end
    end
end

function[ans] = naive_bayes(p, x)
    a = log((p + 1e-9) .^ x) .* ((1 - p + 1e-9) .^ (1 - x));
    ans = sum(a(:));
end
```

```
function[i,j] = find_any(matrix)
    for j = 1:size(matrix,2)
        for i = 1:size(matrix,1)
            if(matrix(i,j) == 1)
                return;
            end
        end
    end
    i = -1;
    j = -1;
end

function[binary_pic] = mybinaryfun(pic, threshold)
    binary_pic = pic > threshold;
end

function[grayscale_pic] = mygrayfun(pic)
    grayscale_pic = 0.299 .* pic(:,:,1) + 0.578 .* pic(:,:,2) + 0.114 .* pic(:,:,3);
end

function[parts] = detect_parts(pic, threshold)
    height = size(pic,1);
    width = size(pic,2);
    parts = {};
    queue = [];
    pairs = [];
    while(true)
        if(isempty(queue))
            if(size(pairs,1) >= threshold)
                parts{end+1} = pairs;
            end
            pairs = [];
            [i,j] = find_any(pic);
            if(i == -1)
                break;
            end
            queue = [i,j];
            pic(i,j) = 0;
        else
            row = queue(1, 1);
            col = queue(1, 2);
            pairs = [pairs; queue(1, :)];
            queue(1, :) = [];
            if(row > 1 && pic(row - 1, col) == 1)
                pic(row - 1, col) = 0;
                queue = [queue;[row - 1, col]];
            end
            if(row < height && pic(row + 1, col) == 1)
                pic(row + 1, col) = 0;
                queue = [queue;[row + 1, col]];
            end
            if(col > 1 && pic(row, col - 1) == 1)
                pic(row, col - 1) = 0;
```

```
                queue = [queue;[row, col - 1]];
            end
            if(col < width && pic(row, col + 1) == 1)
                pic(row, col + 1) = 0;
                queue = [queue;[row, col + 1]];
            end
            if(row > 1 && col > 1 && pic(row - 1, col - 1) == 1)
                pic(row - 1, col - 1) = 0;
                queue = [queue;[row - 1, col - 1]];
            end
            if(row < height && col > 1 && pic(row + 1, col - 1) == 1)
                pic(row + 1, col - 1) = 0;
                queue = [queue;[row + 1, col - 1]];
            end
            if(row > 1 && col < width && pic(row - 1, col + 1) == 1)
                pic(row - 1, col + 1) = 0;
                queue = [queue;[row - 1, col + 1]];
            end
            if(row < height && col < width && pic(row + 1, col + 1) == 1)
                pic(row + 1, col + 1) = 0;
                queue = [queue;[row + 1, col + 1]];
            end

        end

    end

    pairs = [pairs;queue];
    if(size(pairs,1) > threshold)
        parts{end+1} = pairs;
    end



end


function[filtered_pic] = myremovecom(pic, threshold)
    parts = detect_parts(pic, threshold);
    filtered_pic = zeros(size(pic));
    for i = 1:numel(parts)
        part = parts{i};
        for j = 1:size(part,1)
            filtered_pic(part(j,1),part(j,2)) = 1;
        end
    end
end

function[n, labeled_pic] = mysegmentation(pic)
    parts = detect_parts(pic, 1);
    labeled_pic = zeros(size(pic));
    n = numel(parts);
    for i = n:-1:1
        part = parts{i};
```

```
        for j = 1:size(part,1)
            labeled_pic(part(j,1),part(j,2)) = i;
        end
    end
end
```