Computer Assignment #2

Ali Hamzehpour 810100129

Part one

1-0

Before starting, I load the dataset using a function I wrote (*load_dataset)* and I also set the constants I use in the program:

```
WIDTH_RESIZE = 300;
HEIGHT_RESIZE = 500;
THRESHOLD = 80;
MIN_SEGMENT_SIZE = 300;
MAX_SEGMENT_SIZE = 2500;
MIN_CORR = 0.45;
dataset = load_dataset("./p1/Map Set");
```

load_dataset function:

```
function dataset = load_dataset(folder_path)
    file_names = dir(fullfile(folder_path, "*.bmp" ));
    file_names = [file_names; dir(fullfile(folder_path, '*.png'))];
    num_files = length(file_names);

    dataset = cell(num_files, 2);
    for i = 1 : num_files
        image_path = fullfile(folder_path, file_names(i).name);
        image = imread(image_path);
        [~, file_name, ~] = fileparts(file_names(i).name);
        dataset{i, 1} = image;
        dataset{i, 2} = file_name;
    end
end
```

1-1

First we load the we get the plate's image from the user using *uigetfile* function and load it in a 3D matrix:

```
[file, path] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');
img = imread([path, file]);
```

1-2

Now we resize our image using *imresize* command:

```
img = imresize(img, [WIDTH_RESIZE HEIGHT_RESIZE]);
```

1-3

Now we convert our image to grayscale by using this equation:

$$Gray_{channel} = 0.299 \times Red_{channel} + 0.578 \times Green_{channel} + 0.114 \times Blue_{channel}$$

```
gray = mygrayfun(img);
```

mygrayfun function:

```
function gray_img = mygrayfun(img)
    red_values = img(:, :, 1);
    green_values = img(:, :, 2);
    blue_values = img(:, :, 3);
    gray_img = 0.2989 * red_values + 0.5870 * green_values + 0.1140 * blue_values;
end
```

1-4

Now we we convert our image to black and white:

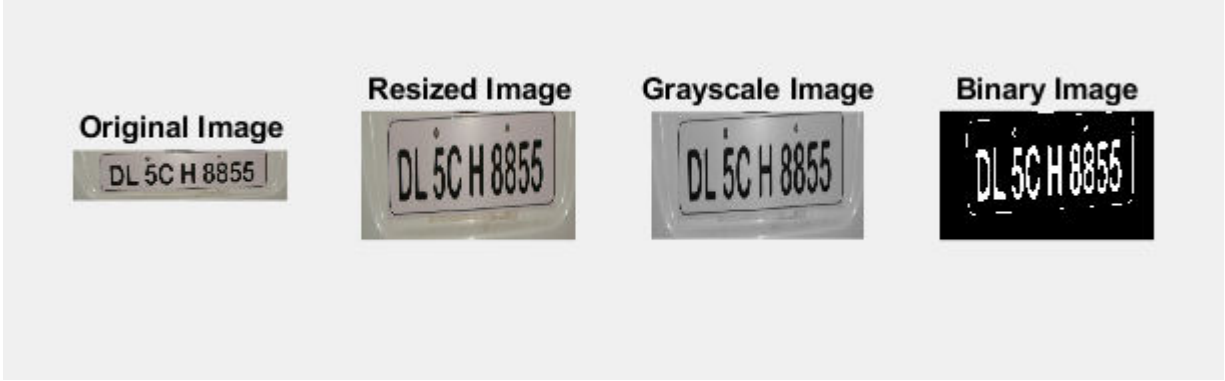
```
binary_img = mybinaryfun(gray, THRESHOLD);
```

mybinaryfun function:

```
function binary_img = mybinaryfun(gray_img, threshold)
    binary_img = gray_img < threshold;
end
```

After performing these changes we plot all of the images we've had:

```
figure
subplot(1,4,1)
imshow(img)
title('Original Image')

subplot(1,4,2)
imshow(img)
title('Resized Image')

subplot(1,4,3)
imshow(gray)
title('Grayscale Image')

subplot(1,4,4)
imshow(binary_img)
title('Binary Image')
```



1-5, 1-6

First we are asked to write the *bewaraopen* function from scratch. This function removes the small white components in the image in order to reduce noise. Then we are asked to rewrite the *bwlabel* function from scratch. This function gets an images and give the white pixels of a component a label. I Implemented this function before *bewareaopen,* so it can be used in *bewareaopen.* The way I Implemented it is by performing BFS algorithm on each white pixel that I see and giving a specific label to the other white pixels that we see in our search.

Mysegmentation function(=*bwlabel)*:

```
function [labeled_img, num_labels] = mysegmentation(binary_img)
    num_labels = 0;
    labeled_img = zeros(size(binary_img));
    [rows, cols] = size(binary_img);

    for j = 1 : cols
        for i = 1 : rows
            if (binary_img(i, j) == 0 || labeled_img(i, j) ~= 0)
                continue;
            end
            queue = [i, j];
            num_labels = num_labels + 1;
            labeled_img(i, j) = num_labels;
            while size(queue, 1) ~= 0
                current_pixel = queue(1, :);
                queue(1, :) = [];
                for x = -1:1
                    for y = -1:1
                        if (x == 0 && y == 0)
                            continue;
                        end
                        neighbor_pixel = current_pixel + [x, y];
                        if (neighbor_pixel(1) < 1 || neighbor_pixel(1) > rows || neighbor_pixel
                            continue;
                        end
                        if (binary_img(neighbor_pixel(1), neighbor_pixel(2)) == 0 || labeled_im
                            continue;
```

```
                        end
                        queue = [queue; neighbor_pixel];
                        labeled_img(neighbor_pixel(1), neighbor_pixel(2)) = num_labels;
                    end
                end
            end
        end
    end
end
```

Now we can implement *myremovecom*(=*bewareaopen*) easily. This function gets the image and a minimum and maximum for number of pixels of a component the clear those unwanted components. It has three outputs:

- clean_img: is the the image matrix after removing those components.
- labeled_img: is the image matrix where the pixels of a component have the same label.
- num_clean_labels: is the number of components after cleaning the image.

```
function [clean_binary_img, clean_labeled_img, num_clean_labels] = myremovecom(binary_img, min_
    [labeled_img, num_labels] = mysegmentation(binary_img);

    pixel_counts = zeros(num_labels, 1);
    clean_labeled_img = zeros(size(labeled_img));
    for label = 1:num_labels
        pixel_counts(label) = sum(labeled_img(:) == label);
    end

    clean_binary_img = zeros(size(binary_img));
    num_clean_labels = 0;
    for label = 1:num_labels
        if pixel_counts(label) >= min_segment_size && pixel_counts(label) < max_segement_size
            clean_binary_img(labeled_img == label) = 1;
            num_clean_labels = num_clean_labels + 1;
            clean_labeled_img(labeled_img == label) = num_clean_labels;
        end
    end
end
```

So in our main program, we can use them:

```
[clean_img, labeled_img, num_clean_labels] = myremovecom(binary_img, MIN_SEGMENT_SIZE, MAX_SEGM
figure
imshow(clean_img)
title('Clean Image')
```

1-7

Now that we have all the components seperated from each other, We can use correlation to determine what character it is. The only point is to use a threshold that if the maximum correlation is less than threshold, We can conclude that component is not a chracter from the plate and we ignore it. In the end, We save our result in output.txt file:

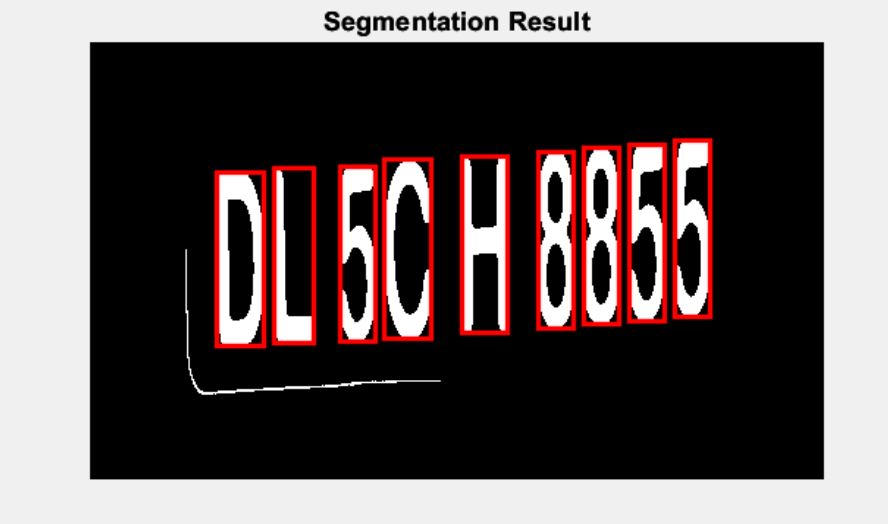
```
license_plate = '';
dataset_img_size = size(dataset{1, 1});
total_letters = size(dataset, 1);

figure
imshow(clean_img)
title('Segmentation Result')
hold on;

for label = 1 : num_clean_labels
    current_obj = labeled_img == label;
    [rows, cols] = find(current_obj);
    xmin = min(cols);
    xmax = max(cols);
    ymin = min(rows);
    ymax = max(rows);

    cropped_image = clean_img(ymin:ymax, xmin:xmax);
    cropped_image = imresize(cropped_image, dataset_img_size);
    corrs = zeros(1,total_letters);
    for k = 1 : total_letters
        corrs(k) = corr2(dataset{k, 1}, cropped_image);
    end
    [maxcor, max_pos] = max(corrs);
    if maxcor < MIN_CORR
        continue;
    end
    cur_character = dataset{max_pos, 2};
    license_plate = [license_plate; cur_character];
```

```
    rectangle('Position', [xmin, ymin, xmax - xmin, ymax - ymin], 'EdgeColor', 'r', 'LineWidth
end

file = fopen('output.txt', 'w');
fprintf(file, license_plate);
fclose(file);
```

Testing

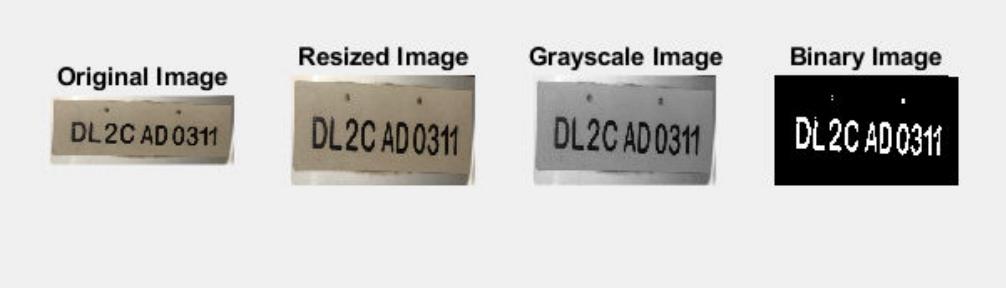
We test our code with 3 images:

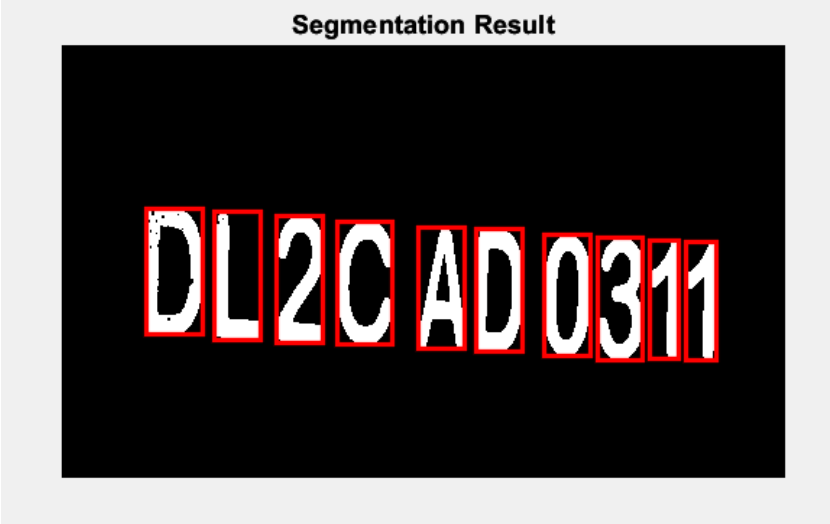
1



The final output is DL5CH8855 which is correct.

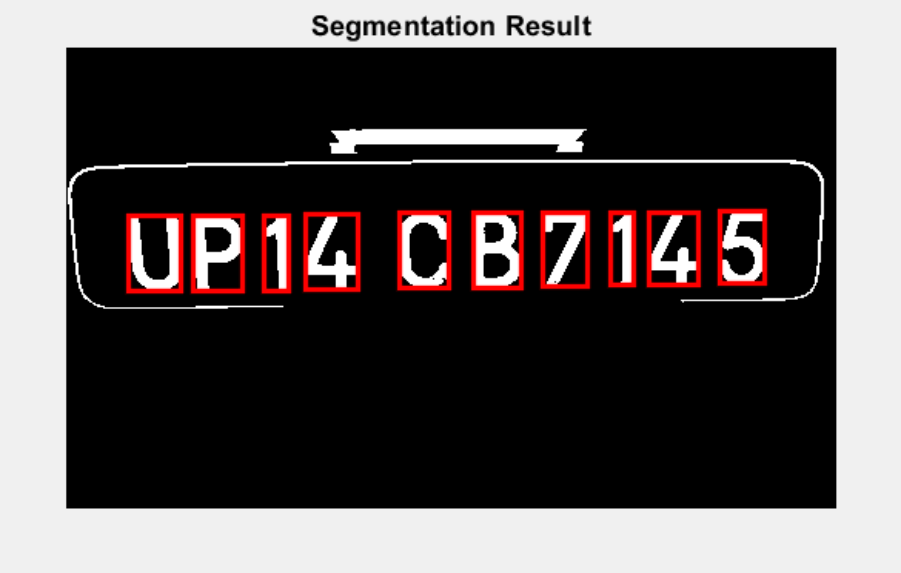
2





The output is DL2CADo311 which o is correct because our dataset is not good enough. If we delete 'o' in our dataset the answer will be correct.(which makes sense because we don't have small letters in license plates)

3



The output is UP14CB7145 which is correct.

Part Two

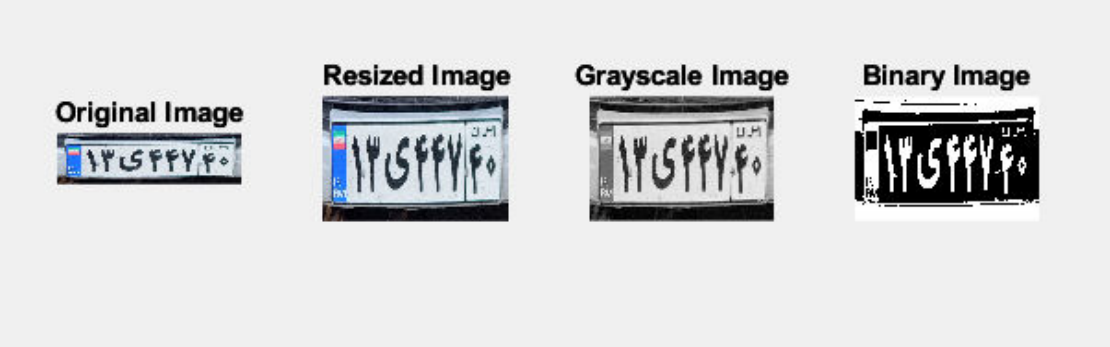
The code is like the previous part but we use another datset which is for persian charcters. We also use MATLAB's built-in functions for grayscale and binarizing:

```
dataset = load_dataset("./p2/Persian Map Set")
[file, path] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');
img = imread([path, file]);

WIDTH_RESIZE = 800;
HEIGHT_RESIZE = 1200;
MIN_SEGMENT_SIZE = 1500;
MAX_SEGEMENT_SIZE = 50000;
MIN_CORR = 0.50;

figure
subplot(1,4,1)
imshow(img)

img = imresize(img, [WIDTH_RESIZE HEIGHT_RESIZE]);
subplot(1,4,2)
imshow(img)

gray = rgb2gray(img);
subplot(1,4,3)
imshow(gray)

threshold = graythresh(gray);
binary_img = ~imbinarize(gray, threshold);
subplot(1,4,4)
imshow(binary_img)
```

```
[clean_img, labeled_img, num_clean_labels] = myremovecom(binary_img, MIN_SEGMENT_SIZE, MAX_SEGE

license_plate = '';
dataset_img_size = size(dataset{1, 1});
total_letters = size(dataset, 1);

figure
imshow(clean_img)
title('Segmentation Result')
hold on;

for label = 1 : num_clean_labels
    current_obj = labeled_img == label;
    [rows, cols] = find(current_obj);
    xmin = min(cols);
    xmax = max(cols);
    ymin = min(rows);
    ymax = max(rows);

    cropped_image = clean_img(ymin:ymax, xmin:xmax);
    cropped_image = imresize(cropped_image, dataset_img_size);
    corrs = zeros(1,total_letters);
    for k = 1 : total_letters
        corrs(k) = corr2(dataset{k, 1}, cropped_image);
    end
    [maxcor, max_pos] = max(corrs);
    if maxcor < MIN_CORR
        continue;
    end
    cur_character = dataset{max_pos, 2};
    license_plate = [license_plate; cur_character];
    rectangle('Position', [xmin, ymin, xmax - xmin, ymax - ymin], 'EdgeColor', 'r', 'LineWidth
end

file = fopen('output.txt', 'w');
fprintf(file, license_plate);
fclose(file);
```

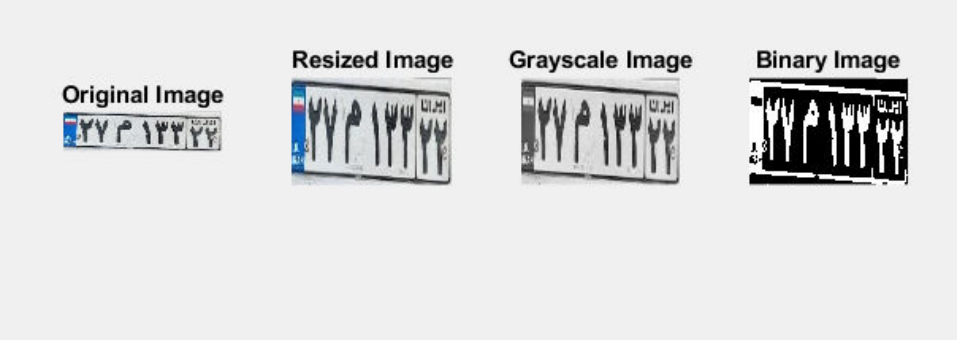
Testing

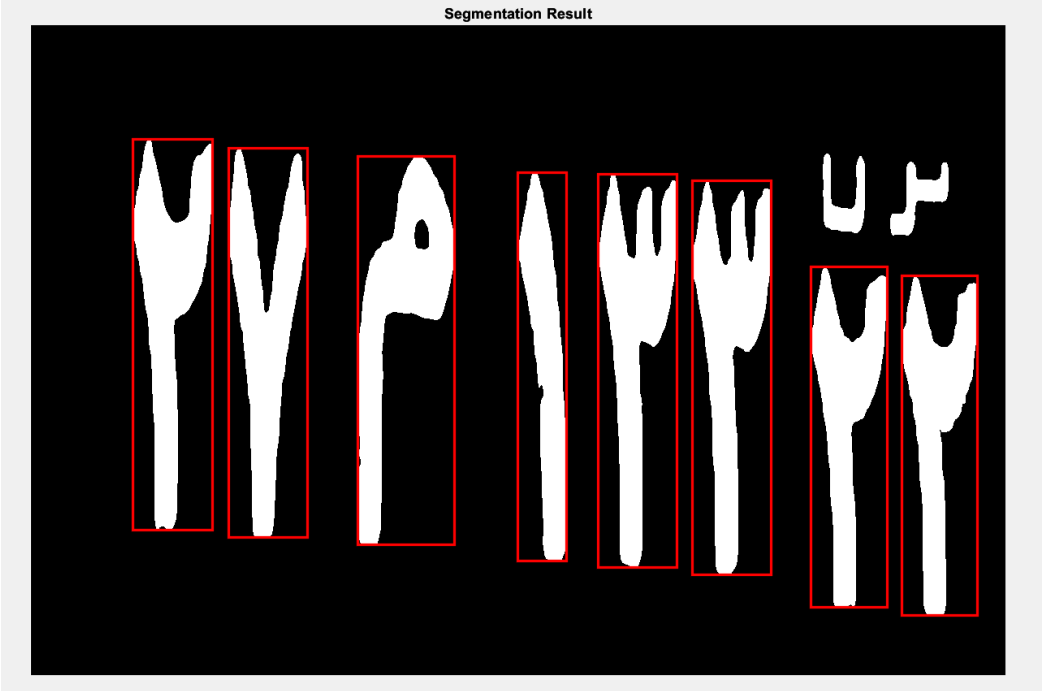
1



The asnwer is 1344740ی which is correct.

2





The answer is 2713322ع which is correct.

Part Three

In this part We are given the image of the front of the car or the rear of the car and we have to detect the plate and then detect the characters. For plate detection I used a bluestrip with different sizes that is in all persian plates. So I calculate the correlation of those bluestrips with all parts of the image and the maximum correlation is where the plate is. Then I find the width and height of the plate with the size of the blustrip. The rest is like part two.

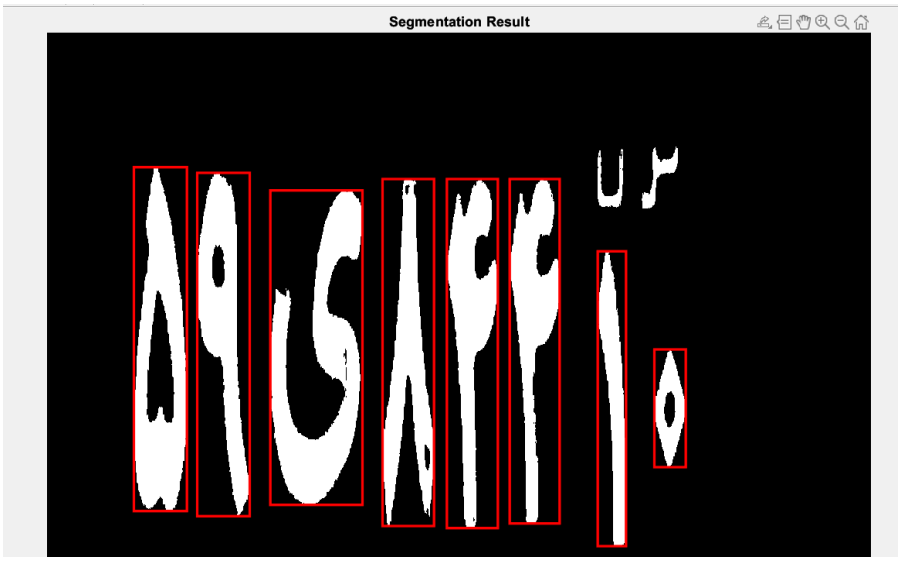
```
dataset = load_dataset("./p2/Persian Map Set");
[file, path] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');

img = imread([path, file]);


plate_box = detect_plate(img);
plate = imcrop(img, plate_box);
figure
imshow(plate);

img = plate;

WIDTH_RESIZE = 1200;
HEIGHT_RESIZE = 1600;
MIN_SEGMENT_SIZE = 1500;
MAX_SEGEMENT_SIZE = 50000;
```

```
MIN_CORR = 0.60;

figure
subplot(1,4,1)
imshow(img)

img = imresize(img, [WIDTH_RESIZE HEIGHT_RESIZE]);
subplot(1,4,2)
imshow(img)

gray = rgb2gray(img);
subplot(1,4,3)
imshow(gray)

threshold = graythresh(gray);
binary_img = ~imbinarize(gray, threshold - 0.1);
subplot(1,4,4)
imshow(binary_img);

[clean_img, labeled_img, num_clean_labels] = myremovecom(binary_img, MIN_SEGMENT_SIZE, MAX_SEGE

license_plate = '';
dataset_img_size = size(dataset{1, 1});
total_letters = size(dataset, 1);

figure
imshow(clean_img)
title('Segmentation Result')
hold on;

for label = 1 : num_clean_labels
    current_obj = labeled_img == label;
    [rows, cols] = find(current_obj);
    xmin = min(cols);
    xmax = max(cols);
    ymin = min(rows);
    ymax = max(rows);

    cropped_image = clean_img(ymin:ymax, xmin:xmax);
    cropped_image = imresize(cropped_image, dataset_img_size);
    corrs = zeros(1,total_letters);
    for k = 1 : total_letters
        corrs(k) = corr2(dataset{k, 1}, cropped_image);
    end
    [maxcor, max_pos] = max(corrs);
    if maxcor < MIN_CORR
        continue;
    end
    cur_character = dataset{max_pos, 2};
    license_plate = [license_plate; cur_character];
```

```
    rectangle('Position', [xmin, ymin, xmax - xmin, ymax - ymin], 'EdgeColor', 'r', 'LineWidth
end

file = fopen('output.txt', 'w');
fprintf(file, license_plate);
fclose(file);

function bounding_box = detect_plate(img)
    RESIZE_WIDTH = 800;
    ERR_MARGIN = 10;
    BLUE2PLATE_RATIO = 14;


    template = imread("template_big.png");
    resized_img = imresize(img, [NaN, RESIZE_WIDTH]);
    ratio = size(img, 1) / size(resized_img, 1);
    corr_max = 0;

    for i = 1 : 50
        new_template = imresize(template, [NaN, size(template, 2) - i + 1]);
        [corr_mixB, corr_maxB, bboxB] = template_match(new_template, resized_img);
        if corr_maxB > corr_max
            [corr_mix, corr_max, bbox] = deal(corr_mixB, corr_maxB, bboxB);
        end
    end

    bbox_full = [round((bbox(1) - ERR_MARGIN) * ratio), ...
                 round((bbox(2) - ERR_MARGIN) * ratio), ...
                 round((bbox(3) + 2 * ERR_MARGIN) * ratio), ...
                 round((bbox(4) + 2 * ERR_MARGIN) * ratio)];

    bounding_box = bbox_full;
    bounding_box(3) = BLUE2PLATE_RATIO * bbox(3) * ratio;


end

function [corr_mix, corr_max, bbox] = template_match(template, pic)
    corr_red = normxcorr2(template(:, :, 1), pic(:, :, 1));
    corr_green = normxcorr2(template(:, :, 2), pic(:, :, 2));
    corr_blue = normxcorr2(template(:, :, 3), pic(:, :, 3));
    corr_mix = (corr_red + corr_green + corr_blue ) / 3;

    [corr_max, color_idx] = max(abs(corr_mix(:)));
    [y, x] = ind2sub(size(corr_mix), color_idx(1));
    corr_offset = [x - size(template, 2), y - size(template, 1)];
    bbox = [corr_offset(1), corr_offset(2), size(template, 2), size(template, 1)];
end
```

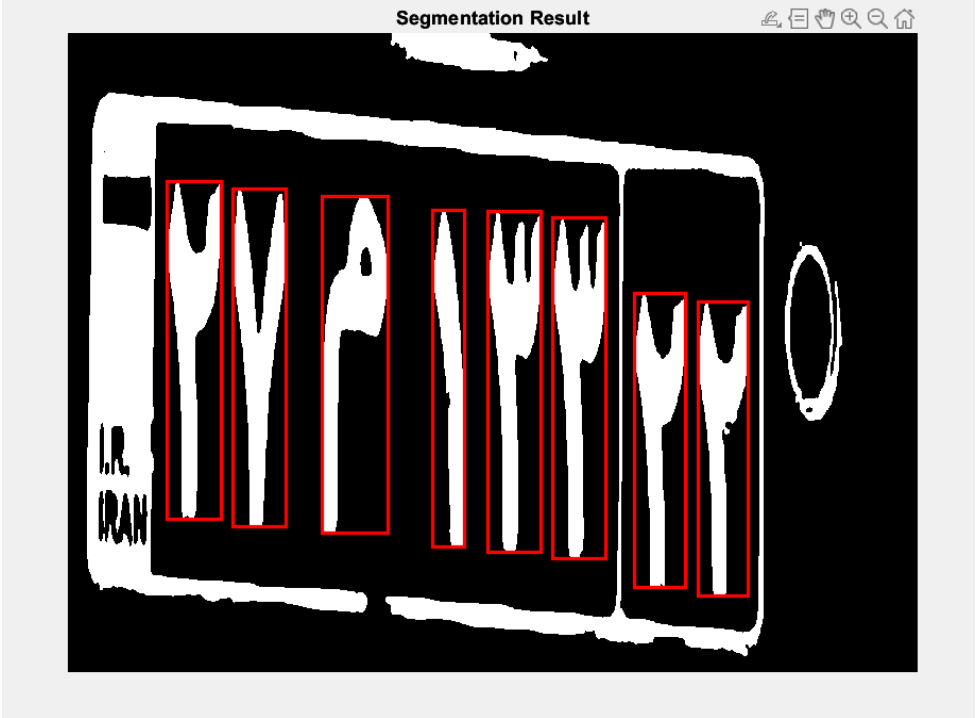
Testing

1



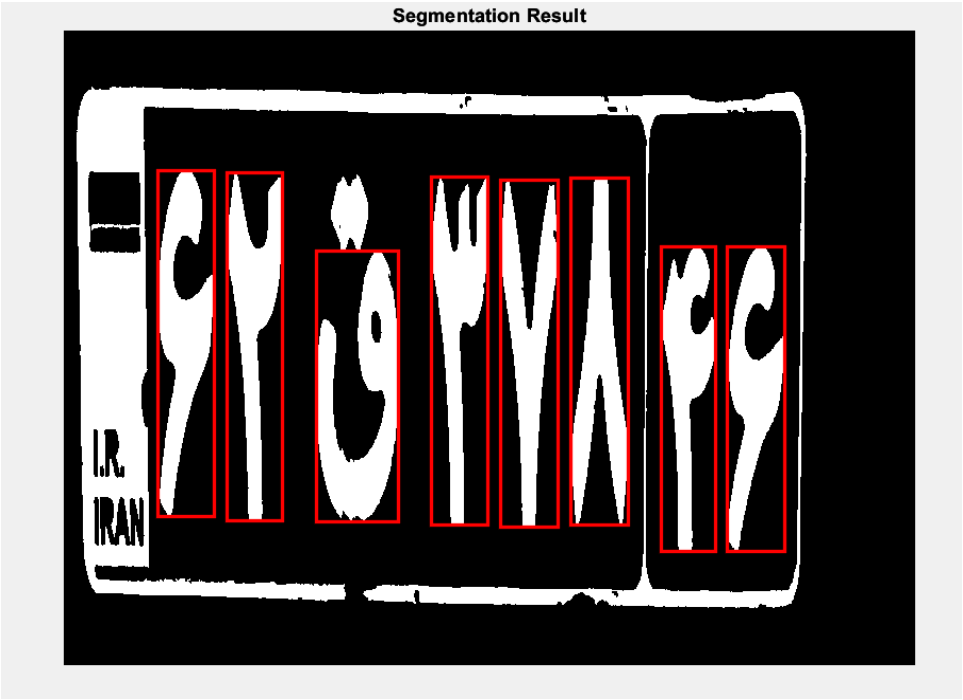
it detects ۵۹۸۴۴۱۰ی which is correct.

2



It detects ۲۷۱۳۳۲۲م which is correct.

3



It detects ۶ ۲۳۷۸۴۶ق which is correct.