Signals and Systems
CA 2

Mohammad Javad Pesarakloo
810100103

October 23, 2023

Phase 1

The overall approach in this phase is as follows:

- Get the input image using **uigetfile** built in function of matlab
- To reduce complexity of data,we make this image binary.To do so we need to determine a thereshold. To find this thereshold I wrote the function **find_ther**.
- Then we have to remove noises.To do so I wrote the function **myremove-com**
- Then we have to divide image into diffrent segments.I wrote the function **mysegmentation** for this porpuse.
- Now using the dataset of english alphabet and numbers,we get correalation of segmented image,we find the corresponding number or alphabet.

mygrayfun

This function has the duty to change RGB picture to gray picture.To do so we have the following formula:

$$GRAY = 0.299 * RED + 0.578 * GREEN + 0.114 * BLUE$$

The script of the function is as follows:

```
function gray_image = mygrayfun(image)
    red_channel = image(:, :, 1);
    green_channel = image(:, :, 2);
    blue_channel = image(:, :, 3);
    gray_image = 0.299 * red_channel + 0.578 * green_channel + 0.114 * blue_channel;
end
```

Figure 1: mygrayfun

find ther

To find the appropraite thereshold,I wrote this function which returns average of minimum and maximum of image:

```
function thereshold = find_ther(image)
    thereshold = (min(image) + max(image)) / 2;
end
```

Figure 2: find there

mybinaryfun

This function takes the gray image and the thereshold we calculated using the above function and returns the binary image:

```
function binary_image = mybinaryfun(gray_image, threshold)
    binary_image = gray_image > threshold;
end
```

Figure 3: mybinaryfun

myremovecom

This function is an equivalent of the matlab built in function **bwareaopen**.To implement this function,I used the Idea of breadth first search;which explores adjacent bits of a single bit and then does this procedure for the adjacents again and then ommits the bits whose adjacents have all been visited.the body of this function is as follows:

```matlab
function [out_picture,ther] = myremovecom(in_picture, n)
    [row, col] = find(in_picture == 1);
    points = [row';col'];
    obj_count = 1;
    points_count = size(points, 2);
    while points_count > 0
        initial_point = points(:, 1);
        points(:, 1) = [];
        [points, new_points] = close_points(initial_point, points);
        cur_obj = [initial_point new_points];
        new_points_len = size(new_points, 2);
        while new_points_len > 0
            new_points2 = [];
            for i = 1:new_points_len
                [points, new_points1] = close_points(new_points(:,i), points);
                new_points2 = [new_points2 new_points1];
            end
            cur_obj = [cur_obj new_points2];
            new_points = new_points2;
            new_points_len = size(new_points, 2);
        end
        obj{obj_count} = cur_obj;
        obj_count = obj_count + 1;
        points_count = size(points, 2);
    end
    obj_count = obj_count - 1;
    t = 1;
    for i = 1:obj_count
        if size(obj{i}, 2) > n
            new_obj{t} = obj{i};
            t = t + 1;
        end
    end
    ther = max(obj{i},2);
    out_picture = zeros(size(in_picture));
    for i = 1:t-1
        ind=sub2ind(size(in_picture),new_obj{i}(1,:),new_obj{i}(2,:));
        out_picture(ind) = 1;
    end
end
```

Figure 4: myremovecom

mysegmentation

This function is an equivalent of the matlab built-int function **bwlabel**.The idea of implementation is just the same as *myremovecom* with the diffrence that after finding the objects,we label them with a number in a for loop;the body of this function is as follows:

```
function [out_picture, num_objects] = mysegmentation(in_picture)
    [row, col] = find(in_picture == 1);
    points = [row'; col'];
    obj_count = 1;
    points_count = size(points, 2);
    while points_count > 0
        initial_point = points(:, 1);
        points(:, 1) = [];
        [points, new_points] = close_points(initial_point, points);
        cur_obj = [initial_point new_points];
        new_points_len = size(new_points, 2);
        while new_points_len > 0
            new_points2 = [];
            for i = 1:new_points_len
                [points, new_points1] = close_points(new_points(:, i), points);
                new_points2 = [new_points2 new_points1];
            end
            cur_obj = [cur_obj new_points2];
            new_points = new_points2;
            new_points_len = size(new_points, 2);
        end
        obj{obj_count} = cur_obj;
        obj_count = obj_count + 1;
        points_count = size(points, 2);
    end
    num_objects = obj_count - 1;
    out_picture = zeros(size(in_picture));
    for i = 1:num_objects
        ind = sub2ind(size(in_picture), obj{i}(1,:), obj{i}(2,:));
        out_picture(ind) = i;
    end
end
```

Figure 5: mysegmentation

Phase 2

This phase is just the same as previous phase with the diffrence that when writing in a text file,the charachter which is persian goes to write of the file;so it does not look the same.to solve this problem I write the charachters and numbers linewise.The script of this phase is as follows:

```
clc
close all;
clear;
[file, path] = uigetfile('*.jpg;*.png;*.jpeg;*.bmp');
picture = imread([path, file]);
picture = rgb2gray(picture);
ther = graythresh(picture);
picture = ~imbinarize(picture, ther);
picture = imresize(picture, [600, 800]);
picture = bwareaopen(picture, 6000);
background = bwareaopen(picture, 20000);
picture = picture - background;
[L, Ne] = bwlabel(picture);
load trainingset;
file = fopen('number_Plate_Persian.txt', 'wt');
output = [];
numOfLetters = size(train, 2);
for n=1:Ne
    [r, c] = find(L == n);
    Y = picture(min(r):max(r), min(c):max(c));
    ro = zeros(1, numOfLetters);
    for k = 1:numOfLetters
        [row, col] = size(train{1,k});
        Y = imresize(Y, [row, col]);
        ro(k) = corr2(train{1,k},Y);
    end
    [MAXRO, pos] = max(ro);
    if MAXRO>.45
        out = cell2mat(train(2,pos));
        output = [output out];
        fprintf(file,'%s\n', out);
    end
end
fclose(file);
winopen('number_Plate_Persian.txt');
```

Figure 6: Detecting with persian dataset

Phase 3

In this phase we have to detect the plate from the image.The Idea comes from the fact that at position of plate,we have several switches from black to white;using this idea we can detect the up,left,right and down bound of the plate and crop it from image.The procedure is as follows:

- Convert image to binary
- To detect the top and bottom of the license plate, we look for the first row of the plate that has the most changes from the top of the image to the row that has the most changes and is close to the maximum changes in a row.
- Finding the left and right bounds is just the same as above

The script of this algorithm is as follows:

```
for i=1: width
    changes_count = 0;
    for j=1: length - 1
        if picture(i, j + 1) ~= picture(i, j)
            changes_count = changes_count + 1;
        end
    end
    horizental_changes_count(i) = changes_count;
    if changes_count > maximum_horizental_changes && i > 300 && i < 500
        maximum_horizental_changes = changes_count;
        y_max_changes = i;
    end
end

down_bound = width - 100;
up_bound = 100;

right_bound = length - 100;
left_bound = 100;

for i=100: y_max_changes
    if abs(horizental_changes_count(i) - maximum_horizental_changes) < 20 && y_max_changes - i < 50
        up_bound = i;
        break;
    end
end

for i=width - 100:-1: y_max_changes
    if abs(horizental_changes_count(i) - maximum_horizental_changes) < 20 && i - y_max_changes < 50
        down_bound = i;
        break;
    end
end
```

Figure 7: plate detecting 1

```
for j=1: length
    changes_count = 0;
    for i=1: width - 1
        if picture(i + 1, j) ~= picture(i, j)
            changes_count = changes_count + 1;
        end
    end
    vertical_changes_count(i) = changes_count;
    if changes_count > maximum_vertical_changes && j > 300 && j < 500
        maximum_vertical_changes = changes_count;
        x_max_changes = j;
    end
end

for j=220: x_max_changes
    if abs(vertical_changes_count(j) - maximum_vertical_changes) < 40 && x_max_changes - j < 300
        left_bound = j;
        break;
    end
end

for j=length - 200:-1: x_max_changes
    if abs(vertical_changes_count(j) - maximum_vertical_changes) < 40 && j - x_max_changes < 300
        right_bound = j;
        break;
    end
end

width_of_picture = down_bound - up_bound;

if width_of_picture < 50
    down_bound = down_bound + 90 - width_of_picture;
end
```

Figure 8: plate detecting 2

After the bounds are found, we are done finding the plate and the rest of the procedure is the same as phase 2