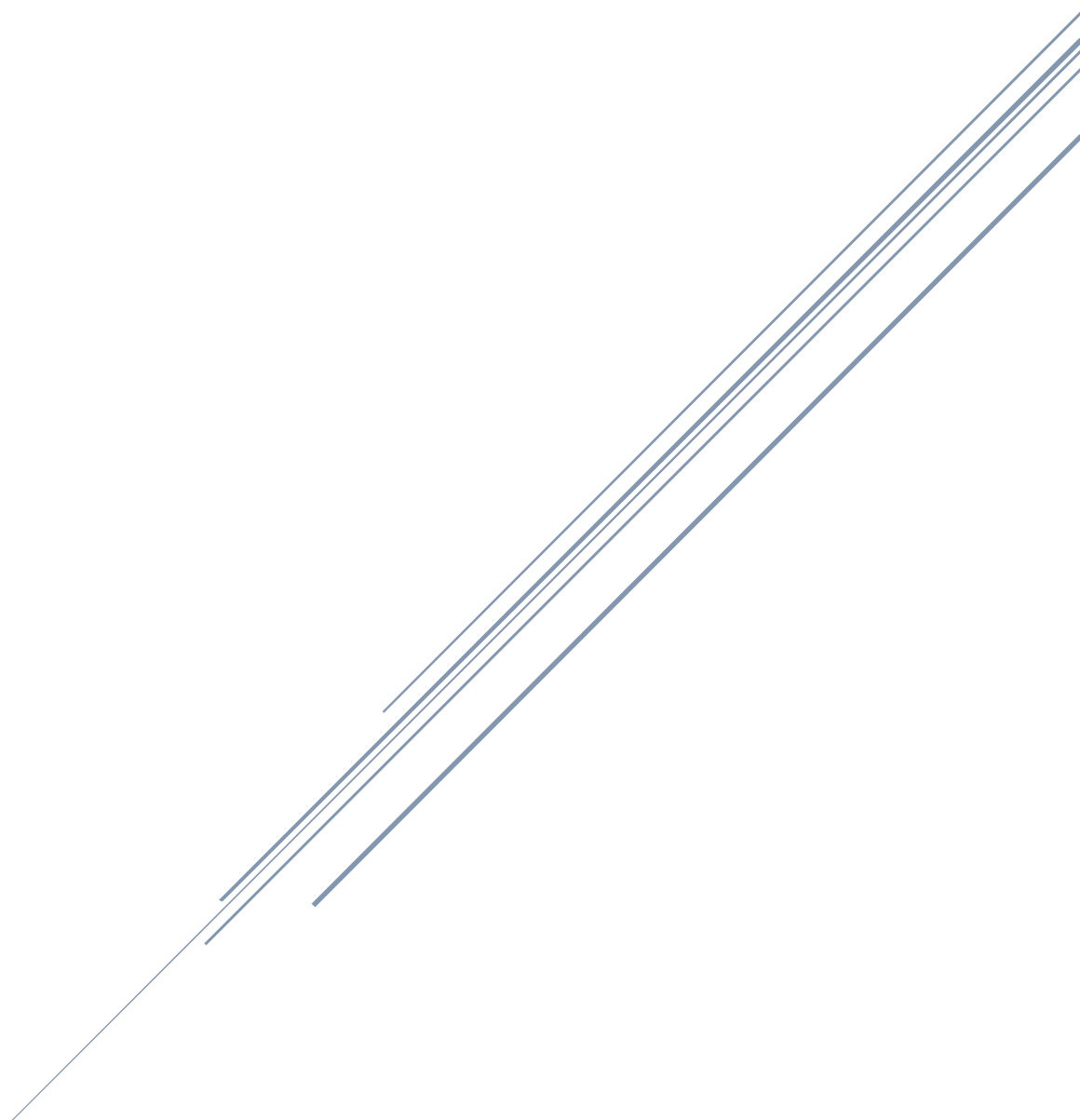


SIGNALS AND SYSTEMS – COMPUTER ASSIGNMENT 2

Rouja Aghajani - 810101380



بخش اول)

-1

با دستور `uigetfile` فایل و آدرس آن ذخیره می شود. سپس این دو در متغیری ذخیره شده و آنگاه به کمک `imread` و آن متغیر عکس مد نظر به صورت یک عکس `rgb` به شکل `uint8` یعنی (8-bit unsigned integer) ذخیره می شود.

```
3 %%  
4 % Question 1  
5 [file , path] = uigetfile({'*.jpg; *.bmp; *.png; *.tif'}, 'please enter the photo');  
6 sourceFile = [path, file];  
7 picture = imread(sourceFile);  
8 figure  
9 subplot(1,2,1)  
10 imshow(picture);
```



-2

با دستور `resize` عکس مدنظر به سائز دلخواه ما که در اینجا `300*500` پیکسل است، در می آید.

```
11 %%  
12 picture = imresize(picture, [300,500]);  
13 subplot(1,2,2)  
14 imshow(picture);
```



-3

با استفاده از تابع `mygrayfun` عکسی که داریم را با ضرایب معین برای بخش قرمز، آبی و سبز آن، تبدیل به یک عکس خاکستری می‌کنیم.

```
16 %%  
17 % Question 3  
18 picture = mygrayfun(picture);  
19 figure  
20 imshow(picture);
```

```
1 function [pictureInGray] = mygrayfun(inputFile)  
2     pictureInGray = 0.299*inputFile(:,:,1) + 0.578* inputFile(:,:,2) + 0.114*inputFile(:,:,3);  
3 end
```

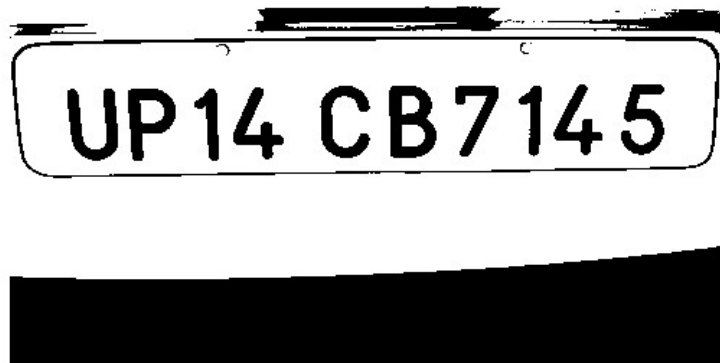


-4

با تعیین یک `threshold` (که می‌توانیم آن را با استفاده از تابع `graythresh(picture)` نیز بیابیم)، آن را به همراه عکس به تابع `mybinaryfun` می‌دهیم. سپس در این تابع، با داشتن `threshold` عکس خاکستری به عکس سیاه و سفید تبدیل می‌شود. روش کار آن به این صورت است که برای هر پیکسل ورودی، بررسی می‌کند که آیا مقدار آن از `threshold` داده شده بیشتر یا کمتر است؛ و آنهایی که بزرگتر است را با 1 و مابقی را با 0 ذخیره می‌کند. در نهایت با 1 ها را به 0 و 0 ها را به 1 تبدیل کرده تا عکس سیاه و سفیدی که ما نیاز داریم بدست آید. (در واقع قبل از تبدیل 0 ها به 1 و برعکس، عکس سیاه و سفید شده بود. تبدیل نهایی برای سایر بخش های برنامه، در واقع بخش `correlation`، نیاز بود.)

```
21 %%  
22 % Question 4  
23 threshold = 100;  
24 picture = mybinaryfun(picture,threshold);  
25 figure  
26 imshow(picture);
```

```
1 ☐ function [binaryPic] = mybinaryfun(grayPicture, threshold)
2     binaryPic = grayPicture > threshold;
3     imshow(binaryPic);
4     binaryPic = ~binaryPic;
5     end
```



-5

در اینجا باید بخش های کوچک اضافی و همچنین فریم دور پلاک از عکس جدا شوند. این کار به کمک تابع `myremovecom` صورت میگیرد.

روش کار تابع:

تابع `myremovecom` اجزای متصل را در یک تصویر باینری لیبل گذاری می کند و اشیاء کوچک را با صفر کردن پیکسل های مربوطه آنها حذف می کند. این تابع در درون خود از یک تابع بازگشتی `labelObject` برای برچسب زدن اجزای متصل بهره می برد و این فرآیند اجسام کوچک را از تصویر باینری بر اساس (`minArea`) فیلتر می کند؛ که `minArea` حداقل مساحتی است که برای اجزای اضافی در نظر گرفته شده است.

نحوه لیبل کردن اجزای متصل به هم:

برای هر پیکسل از تصویر باینری یک حلقه (`loop`) انجام میشود. اگر با یک پیکسل سفید (`binaryImage(i, j) == 1`) مواجه شد و هنوز لیبل نشده است (`labeledImage(i, j) == 0`)، مولفه متصل را با استفاده از تابع `labelObject` لیبل گذاری می شود. پس از لیبل گذاری هر جزء متصل، `Label` فعلی را افزایش می یابد.

محاسبه مساحت هر بخش:

مساحت هر جزء متصل لیبل گذاری شده محاسبه می شود. سپس آن مساحت در آرایه مساحت ها ذخیره می شود.

پیدا کردن اجسام کوچک:

اجسام کوچکی که مساحت آنها کمتر از `minArea` است شناسایی می شوند.

حذف اجسام کوچک:

پیکسل های مربوط به اشیاء کوچک را در تصویر لیبل زده شده صفر می شود.

تصویر باینری نهایی:

تصویر لیبل گذاری شده به یک تصویر باینری تبدیل می شود که در آن پیکسل های متعلق به هر شی روی سفید (1) و بقیه پیکسل ها روی سیاه (0) تنظیم می شوند.

تابع `labelObject`:

این تابع به صورت بازگشتی اجزای متصل را لیبل گذاری می کند.

با یک `if` بررسی می شود که آیا پیکسل فعلی در محدوده های تصویر است، پیکسل سفید است و قبلاً لیبل نشده است. در صورت عدم رعایت هر یک از این شرایط، از تابع خارج می شود.

لیبل گذاری و بازگشت:

پیکسل فعلی با لیبل داده شده لیبل گذاری می شود. تابع `labelObject` را به صورت بازگشتی برای پیکسل های همسایه خود فراخوانی می شود (8-پیکسل).

```
27 %%
28 %Question 5
29 noise = 350;
30 frameSize = 2300;
31 withoutNoise = myremovecom(picture, noise);
32 frame = myremovecom(picture, frameSize);
33 picture = withoutNoise - frame;
34 figure
35 imshow(picture);
```

```

1 function binaryImage = myremovecom(binaryImage, minArea)
2     labeledImage = zeros(size(binaryImage));
3     currentLabel = 1;
4     [rows, cols] = size(binaryImage);
5     for i = 1:rows
6         for j = 1:cols
7             if binaryImage(i, j) == 1 && labeledImage(i, j) == 0
8                 labeledImage = labelObject(binaryImage, labeledImage, i, j, currentLabel);
9                 currentLabel = currentLabel + 1;
10            end
11        end
12    end
13
14    areas = zeros(1, currentLabel - 1);
15    for label = 1:(currentLabel - 1)
16        areas(label) = sum(sum(labeledImage == label));
17    end
18
19    smallObjectsIdx = find(areas < minArea);
20
21    for i = 1:numel(smallObjectsIdx)
22        labeledImage(labeledImage == smallObjectsIdx(i)) = 0;
23    end
24
25    binaryImage = labeledImage > 0;
26 end
27
28 function labeledImage = labelObject(binaryImage, labeledImage, row, col, label)
29     [rows, cols] = size(binaryImage);
30
31     if row < 1 || row > rows || col < 1 || col > cols || binaryImage(row, col) == 0 || labeledImage(row, col) > 0
32         return;
33     end
34
35     labeledImage(row, col) = label;
36     for i=-1:1
37         for j=-1:1
38             labeledImage = labelObject(binaryImage, labeledImage, row+i, col+j, label);
39         end
40     end
41 end

```

UP14 CB7145

-6

اکنون باید به شماره گذاری اجزای تصویر (component) های آن بپردازیم. الگوریتم این قسمت بدین صورت است که ابتدا جمع مقادیر پیکسل های هر ستون بدست می آید، سپس ستون هایی که مقدار همه ی آن ها صفر نیست نظر گرفته می شوند حال باید به دنبال زیر آرایه ای از این آرایه باشیم تا بتوانیم بگویم هر جز تصویر از چه ستونی تا ستونی است، در نتیجه ستون های ابتدایی و انتهایی هر جزء بدست می آید. بعد از آنکه ستون ابتدایی و انتهایی هر جز بدست آمدند، کافیهست که مقدار غیر صفر در این ستون هارا به شماره ایندکس هر جز مقدار دهی کنیم

```
36 %%  
37 %Question 6  
38 [L,Ne] = mysegmentation(picture);  
39 propied=regionprops(L,'BoundingBox');  
40 hold on  
41 for n=1:size(propied,1)  
42     rectangle('Position',propied(n).BoundingBox,'EdgeColor','g','LineWidth',2)  
43 end  
44 hold off
```

```

1 function [LabeledPic , Ne]=mysegmentation(picture)
2     arr = sum(picture);
3     subStart = 0;
4     insideSubarray = false;
5     foundSubarrays = [];
6     for i = 1:length(arr)
7         if arr(i) == 0
8             if insideSubarray
9                 subEnd = i - 1;
10                foundSubarrays = [foundSubarrays; subStart, subEnd];
11                insideSubarray = false;
12            end
13        else
14            if ~insideSubarray
15                subStart = i;
16                insideSubarray = true;
17            end
18        end
19    end
20
21    if insideSubarray
22        subEnd = length(arr);
23        foundSubarrays = [foundSubarrays; subStart, subEnd];
24    end
25    Ne = size(foundSubarrays , 1);
26    LabeledPic=zeros(size(picture));
27
28    for i=1 :Ne
29        theStart = foundSubarrays(i , 1 );
30        theEnd = foundSubarrays(i , 2);
31        [row,column]=find(picture(: , theStart:theEnd)==1);
32        indicated=sub2ind(size(picture),row',(column+theStart-1)');
33        LabeledPic(indicated)=i;
34
35    end
36 end

```



-7

ابتدا با استفاده از training loading دیتابیس مربوط به اعداد و حروف انگلیسی را تبدیل به ماتریس میکنیم. سپس در اسکرینت اصلی برنامه، یعنی همان p1، ماتریس ساخته شده را لود میکنیم.

تصویر را به سایز 42*24 resize کرده، آنگاه correlation هر جزء را با دیتاست محاسبه میکنیم، و حداکثر مقدار correlation یافت شده به همراه ایندکس آن ذخیره می کنیم. در نهایت در دیتاست با استفاده از ایندکس یافت شده، حرف یا رقم مربوط به آن جزء تصویر را یافته و در ماتریس مربوط به خروجی نهایی ذخیره می کنیم.

```
1      clc;
2      clear;
3      close all;
4
5      files=dir('Map Set');
6      len=length(files)-2;
7      TRAIN=cell(2,len);
8
9      for i=1:len
10         TRAIN{1,i}=imread(['Map Set','\ ',files(i+2).name]);
11         TRAIN{2,i}=files(i+2).name(1);
12     end
13
14     save('TRAININGSET.mat','TRAIN');
15     clear;
```

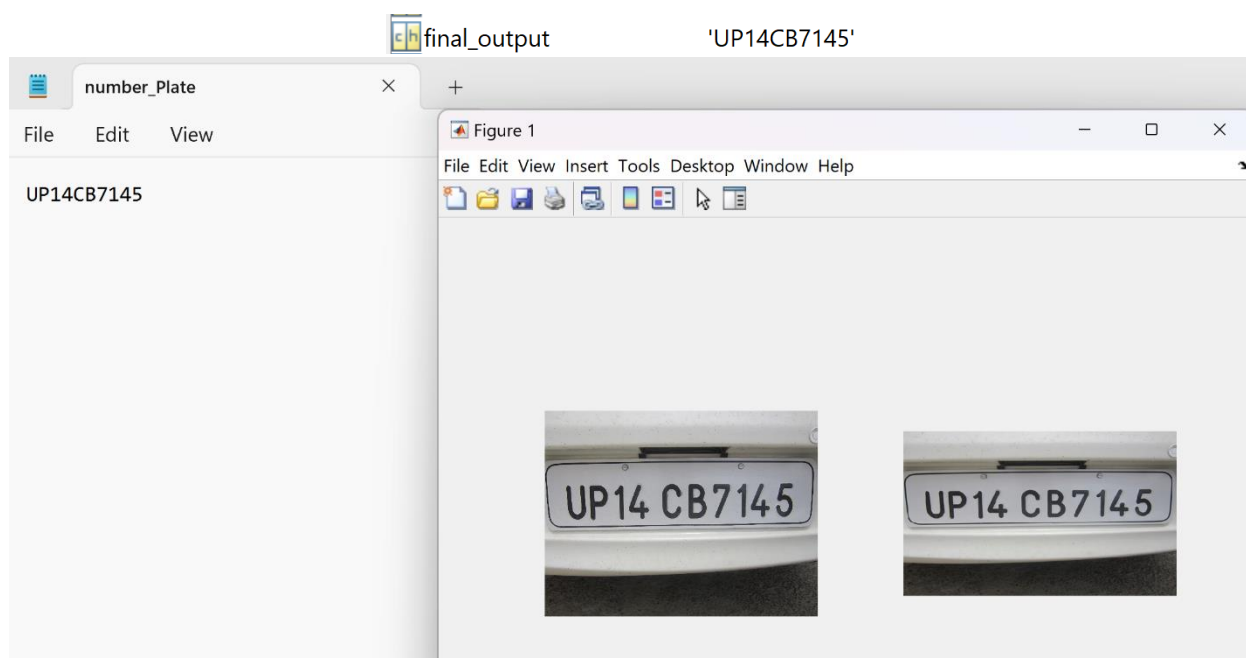
```
45     %%
46     %Question 7
47     load TRAININGSET;
48     totalLetters=size(TRAIN,2);
49     figure
50     final_output=[];
51     t=[];
52     for n=1:Ne
53         [r,c]=find(L==n);
54         Y=picture(min(r):max(r),min(c):max(c));
55         imshow(Y)
56         Y=imresize(Y,[42,24]);
57         imshow(Y)
58         pause(0.2)
59
60         ro=zeros(1,totalLetters);
61
62         for k=1:totalLetters
63             ro(k)=corr2(TRAIN{1,k},Y);
64         end
65         [MAXRO,pos]=max(ro);
66         if MAXRO>.45
67             out=cell2mat(TRAIN(2,pos));
68             final_output=[final_output out];
69         end
70     end
```

-8

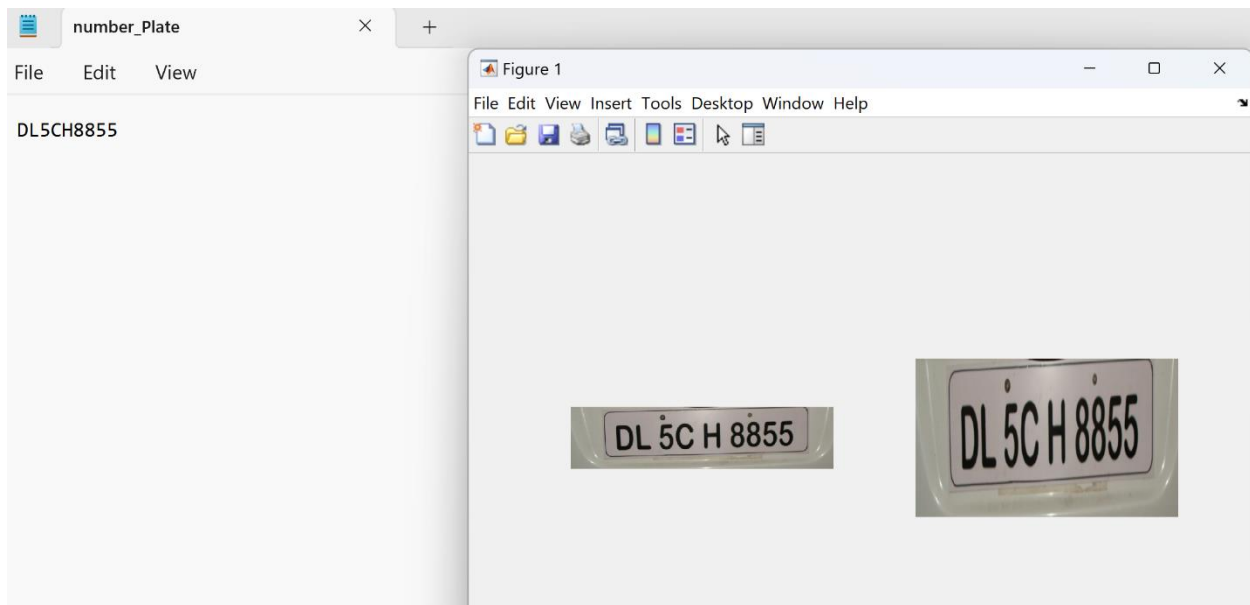
با دستور `fopen` فایل را برای ذخیره پلاک باز میکنیم. سپس با `fprintf` مقدار ماتریس مربوط به پلاک، که حاوی ارقام و حروف پلاک بود و در بخش قبل پیدا کردیم، را در آن فایل مینویسیم. بدین ترتیب ارقام و حروف پلاک ماشین با همان ترتیب در یک فایل ذخیره می شوند.

```
71 %%  
72 %Question 8  
73 file = fopen('number_Plate.txt', 'wt');  
74 fprintf(file, '%s\n', final_output);  
75 fclose(file);  
76 winopen('number_Plate.txt')
```

خروجی برای نمونه پلاک ها:



final_output 'DL5CH8855'



مشاهده می شود دو پلاک با دقت 100% تشخیص داده شده اند.

بخش دوم)

در اینجا روش کار مشابه بالاست. با دستور `uigetfile` و در نهایت با `imread` عکس مدنظر خوانده می شود. سپس با تابع `rgb2gray` عکس رنگی به عکس خاکستری تبدیل می شود. در ادامه با تابع `graythresh` یک `threshold` برای تبدیل عکس خاکستری به سیاه و سفید بدست آمده، آنگاه با تابع `imbinarize` عکس مربوطه تبدیل به عکس سیاه و سفید می شود.

قدم بعدی حذف اجزای کوچک اضافه و همچنین فریم دور پلاک است. با در نظر گرفتن یک `threshold` برای حذف اجزای کوچک و یک `threshold` دیگر برای حذف فریم دور قاب که بزرگتر است، دو عکس بدست می آید؛ یکی حاوی کل اجزای عکس اولیه به جز اجزای کوچکتر، و دیگری حاوی فریم دور قاب. با کم کردن فریم از آن عکس، تصویر نهایی شامل تنها اعداد و حروف پلاک بدست می آید.

در گام بعدی با تابع `bwlabel` بخش های عکس مدنظر لیبل گذاری می شود؛ یعنی هر حرف و هر رقم در یک لیبل (به شکل یک مربع سبز) قرار میگیرند. علت اینکار آن است که بتوانیم با استفاده از `correlation` برای هر رقم و هر حرف، مقدار یا حرف مربوطه را شناسایی کنیم.

سپس `map set` برای زبان فارسی لود می شود. از اسکریپت `persian_training_loading` استفاده میکنیم، که در آن با استفاده از مپ ست فارسی تهیه شده یک ماتریس دارای حروف و ارقام فارسی مورد نیاز ایجاد می شود. سپس این ماتریس با نام `PTRAININGSET` ذخیره می شود تا از آن در `correlation` استفاده کنیم.

حال ماتریس حاوی ارقام و حروف فارسی در اسکریپت اصلی برنامه لود می شود. سپس هر جزء از پلاک که در بخش قبل لیبل گذاری کردیم، با هر یک از اجزای این ماتریس مقایسه و `correlation` برایشان محاسبه می شود. در نهایت حداکثر مقدار `correlation` بدست آمده و ایندکس مربوط به آن را پیدا میکنیم. با استفاده از این ایندکس و مپ ست فارسی، رقم یا حرف مربوط به آن جزء از پلاک که بیشترین همخوانی با آن راداشت پیدا کرده، و به عنوان مقدار آن جزء در یک ماتریس ذخیره می کنیم.

در نهایت فایل تکستی با نام `number_plate` باز شده و در آن مقدار ماتریسی که در آن پلاک را ذخیره کردیم مینویسیم. بدین صورت پلاک ماشین خوانده شده و در یک فایل `txt` ذخیره می شود.

نکته حائز اهمیت آن است که به علت عدم وجود فونت فارسی در متلب و در notes، در بخش variables در متلب پلاک به صورت خلاف جهت (یعنی راست ترین رقم در چپ ترین جایگاه) ذخیره شده است؛ و در نوت پد نیز حرف فارسی سمت راست ارقام ذخیره می شود.

```
1  clc;clear;close all;
2
3  di=dir('Persian Map Set');
4  st={di.name};
5  nam=st(3:end);
6  len=length(nam);
7
8
9  PTRAIN=cell(2,len);
10 for i=1:len
11     pic = imread(['Persian Map Set','/',cell2mat(nam(i))]);
12     pic = im2gray(pic);
13     threshold = graythresh(pic);
14     pic = im2bw(pic,threshold);
15     PTRAIN(1,i)={pic};
16     temp=cell2mat(nam(i));
17     PTRAIN(2,i)={temp(1)};
18 end
19
20 save('PTRAININGSET.mat','PTRAIN');
21 clear;
```

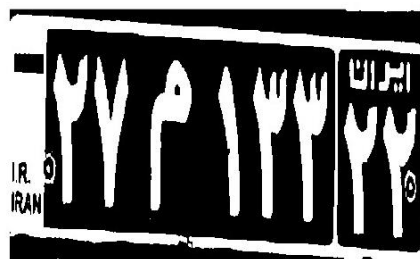
1	%%	
2	% part1: image input	
3	[file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'}, 'Choose an image');	
4	s=[path,file];	
5	picture=imread(s);	
6	figure	
7	subplot(1,2,1)	
8	imshow(picture)	
9	picture=imresize(picture,[300 500]);	
10	subplot(1,2,2)	
11	imshow(picture)	
12	%%	
13	%part2: RGB to Gray	
14	picture=rgb2gray(picture);	
15	figure	
16	subplot(1,2,1)	
17	imshow(picture)	
18		
19	%%	
20	% part3: Gray to Binary	
21	threshold = graythresh(picture);	
22	picture = ~imbinarize(picture,threshold);	
23	subplot(1,2,2)	
24	imshow(picture)	
25		
26	%%	
27	%part4: removing small objects + background	
28	figure	
29	picture = bwareaopen(picture,500);	
30	subplot(1,3,1)	
31	imshow(picture)	
32	background=bwareaopen(picture,4100);	
33	subplot(1,3,2)	
34	imshow(background)	
35	picture2=picture-background;	
36	subplot(1,3,3)	
37	imshow(picture2)	
38		

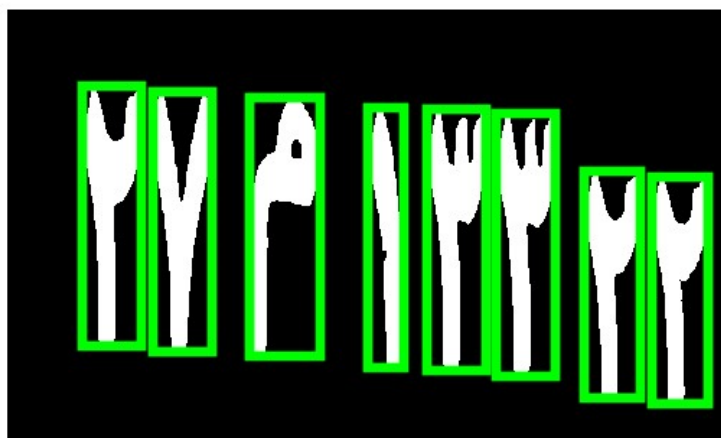
```


39 %%
40 %part5: labeling picture
41 figure
42 imshow(picture2)
43 [L,Ne]=bwlabel(picture2);
44 propied=regionprops(L,'BoundingBox');
45 hold on
46 for n=1:size(propied,1)
47     rectangle('Position',propied(n).BoundingBox,'EdgeColor','g','Linewidth',2)
48 end
49 hold off
50 %%
51 % part6: Loading the mapset
52 load PTRAININGSET;
53 totalLetters=size(PTRAIN,2);
54
55 %%
56 % part7: correlation of each part with the mapset
57 final_output=[];
58 t=[];
59 for n=1:Ne
60     [r,c]=find(L==n);
61     Y=picture(min(r):max(r),min(c):max(c));
62     Y=imresize(Y,[60,50]);
63     ro=zeros(1,totalLetters);
64     for k=1:totalLetters
65         ro(k)=corr2(PTRAIN{1,k},Y);
66     end
67     [MAXRO,pos]=max(ro);
68     if MAXRO>.50
69         out=cell2mat(PTRAIN(2,pos));
70         final_output=[final_output , out];
71     end
72 end
73 %%
74 % part8: Printing the plate
75 file = fopen('number_Plate.txt', 'wt');
76 fprintf(file,'%s\n',final_output);
77 fclose(file);
78 winopen('number_Plate.txt')

```

تست با پلاک داده شده:

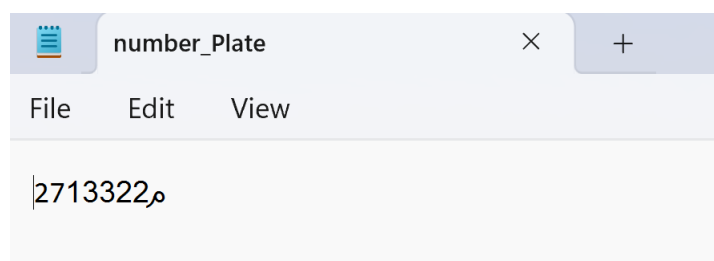




 final_output

'27م13322'

به علت عدم وجود فونت فارسی در متلب، پلاک از چپ به راست ذخیره شده است (یعنی راست ترین عدد به صورت چپ ترین عدد ذخیره شده است و ارقام بعدی نیز همینطورند).



به علت عدم وجود فونت فارسی در Microsoft notes، حرف فارسی در سمت راست پلاک و باقی اعداد در سمت چپ تکست ذخیره شده اند.

بخش سوم)

روش کار کلی در اینجا مانند دو بخش دیگر است؛ تنها تفاوت آن است که باید ابتدا پلاک ماشین در تصویر شناسایی گردد. روش کلی شناسایی:



همانطور که مشاهده می شود، پلاک های فارسی در سمت راست خود، بخش آبی رنگی دارند. در نتیجه با استفاده از یک تصویر از بخش آبی رنگ، و محاسبه correlation آن با کل تصویر، پیدا میکنیم که حداکثر correlation کجا رخ داده است. سپس محل آن را پیدا کرده، و در نتیجه پلاک شناسایی می شود. محاسبه همبستگی متقابل:

correlation 'normxcorr2 نرمال شده بین هر چنل RGB بخش آبی و تصویر ورودی را به طور جداگانه محاسبه می کند. سه ماتریس correlation (corrBlu, corrGrn, corrRed) محاسبه می شود که هر کدام مربوط به چنل های قرمز، سبز و آبی است.

ماتریس های correlation مجزا برای به دست آوردن یک ماتریس correlation ترکیبی (corr_mix) که نشان دهنده شباهت بین الگو و تصویر ورودی در تمام کانال های RGB است، میانگین می شوند.

یافتن حداکثر correlation:

حداکثر مقدار همبستگی (corr_max) و شاخص آن (corrIndex) در ماتریس correlation ترکیبی یافت می شود. تبدیل ایندکس به مختصات:

شاخص حداکثر مقدار correlation با استفاده از ind2sub به مختصات مربوط به آن (x,y) در ماتریس corr_mix تبدیل می شود. این مختصات نشان دهنده افست تطابق الگو در تصویر ورودی است.

محاسبه مختصات تشخیص داده شده:

افست برای نشان دادن مکان الگوی شناسایی شده در تصویر ورودی تنظیم می شود. اندازه الگو برای تشکیل یک کادر محدود در اطراف ناحیه شناسایی شده استفاده می شود. متغیر تشخیص حاصل شامل مختصات و ابعاد منطقه شناسایی شده است.

خروجی:

تابع سه مقدار را برمی گرداند:

corr_mix: ماتریس correlation ترکیبی.

corr_max: حداکثر مقدار correlation.

تشخیص: مختصات و ابعاد منطقه شناسایی شده.

حال در تابع اصلی:

fullSizeBound یک بردار چهار عنصری است که کادر مرزی اطراف نوار آبی شناسایی شده را نشان می دهد.

detection(1) - ERR_MARGIN) * ratio

با کم کردن حاشیه خطا از مختصات x شناسایی شده و مقیاس آن با نسبت تغییر اندازه، مختصات x سمت چپ کادر محدود را محاسبه می کند.

(detection(2) - ERR_MARGIN) * ratio

مختصات y بالای کادر محدود را با کم کردن حاشیه خطا از مختصات y شناسایی شده و مقیاس آن با نسبت تغییر اندازه محاسبه می کند.

(detection(3) + 2 * ERR_MARGIN) * ratio

با افزودن دوبرابر حاشیه خطا به عرض شناسایی شده و مقیاس آن بر اساس نسبت تغییر اندازه، عرض کادر محدود کننده را محاسبه می کند.

(detection(4) + 2 * ERR_MARGIN) * ratio

ارتفاع کادر محدود را با افزودن دو برابر حاشیه خطا به ارتفاع شناسایی شده و مقیاس آن با نسبت تغییر اندازه محاسبه می کند.
تابع round برای گرد کردن مقادیر محاسبه شده به نزدیکترین عدد صحیح استفاده می شود زیرا مختصات جعبه مرزی معمولاً اعداد صحیح هستند.

plateSizeBound در ابتدا همان مقادیر fullSizeBound را دارد.

عرض جعبه مرزی را برای در نظر گرفتن اندازه صفحه تنظیم می کند. سومین عنصر plateSizeBound را با ضرب پهنای شناسایی شده (در detection(3) و BLUE2PLATE_RATIO و با نسبت تغییر اندازه آپدیت می کند.

این adjustment با فرض یک نسبت مشخص بین عرض نوار آبی و عرض پلاک، به تنظیم باکس مرزی برای تقریب اندازه پلاک خودرو کمک می کند.

حال در اسکرپت اصلی با داشتن مختصات پلاک، باقی اجزای عکس حذف می شود. نتیجه آن است که گویا به ابتدای سال بخش قبل رسیدیم، یعنی حال یک عکس تنها از پلاک داریم که باید آن را بخوانیم؛ لذا مراحل از این به بعد، دقیقاً همان مراحل اجرا شده در بالا هستند.

مراحل بعدی:

با تابع rgb2gray عکس رنگی به عکس خاکستری تبدیل می شود. در ادامه با تابع graythresh یک threshold برای تبدیل عکس خاکستری به سیاه و سفید بدست آمده، آنگاه با تابع imbinarize عکس مربوطه تبدیل به عکس سیاه و سفید می شود.

قدم بعدی حذف اجزای کوچک اضافه و همچنین فریم دور پلاک است. با در نظر گرفتن یک threshold برای حذف اجزای کوچک و یک threshold دیگر برای حذف فریم دور قاب که بزرگتر است، دو عکس بدست می آید؛ یکی حاوی کل اجزای عکس اولیه به جز اجزای کوچکتر، و دیگری حاوی فریم دور قاب. با کم کردن فریم از آن عکس، تصویر نهایی شامل تنها اعداد و حروف پلاک بدست می آید.

در گام بعدی با تابع bwlable بخش های عکس مدنظر لیبل گذاری می شود؛ یعنی هر حرف و هر رقم در یک لیبل (به شکل یک مربع سبز) قرار میگیرند. علت اینکار آن است که بتوانیم با استفاده از correlation برای هر رقم و هر حرف، مقدار یا حرف مربوطه را شناسایی کنیم.

سپس map set برای زبان فارسی لود می شود. از اسکرپت persian_training_loading استفاده میکنیم، که در آن با استفاده از مپ ست فارسی تهیه شده یک ماتریس دارای حروف و ارقام فارسی مورد نیاز ایجاد می شود. سپس این ماتریس با نام PTRAININGSET ذخیره می شود تا از آن در correlation استفاده کنیم.

حال ماتریس حاوی ارقام و حروف فارسی در اسکرپت اصلی برنامه لود می شود. سپس هر جزء از پلاک که در بخش قبل لیبل گذاری کردیم، با هر یک از اجزای این ماتریس مقایسه و correlation برایشان محاسبه می شود. در نهایت حداکثر مقدار

correlation بدست آمده و ایندکس مربوط به آن را پیدا میکنیم. با استفاده از این ایندکس و مپ ست فارسی، رقم یا حرف مربوط به آن جزء از پلاک که بیشترین همخوانی با آن راداشت پیدا کرده، و به عنوان مقدار آن جزء در یک ماتریس ذخیره می کنیم.

در نهایت فایل تکستی با نام **number_plate** باز شده و در آن مقدار ماتریسی که در آن پلاک را ذخیره کردیم مینویسیم. بدین صورت پلاک ماشین خوانده شده و در یک فایل **txt**. ذخیره می شود.

نکته حائز اهمیت آن است که به علت عدم وجود فونت فارسی در متلب و در بخش **variables** در متلب پلاک به صورت خلاف جهت (یعنی راست ترین رقم در چپ ترین جایگاه) ذخیره شده است؛ و در نوت پد نیز حرف فارسی سمت راست ارقام ذخیره می شود.

1	<code>clear</code>	
2	<code>clc</code>	
3	<code>%%</code>	
4	<code>% part1: image input</code>	
5	<code>[file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'}, 'Choose an image');</code>	
6	<code>s=[path,file];</code>	
7	<code>picture=imread(s);</code>	
8	<code>figure</code>	
9	<code>subplot(1,2,1)</code>	
10	<code>imshow(picture)</code>	
11	<code>%%</code>	
12	<code>% part2: detecting the plate</code>	
13	<code>boundingBoxes = detectplate_bluestrip(picture);</code>	
14	<code>result = [];</code>	
15	<code>for i = 1:size(boundingBoxes, 1)</code>	
16	<code> picture = imcrop(picture, boundingBoxes(i, :));</code>	
17	<code> figure</code>	
18	<code> imshow(picture);</code>	
19	<code> [a,b,~] = size(picture);</code>	
20	<code>end</code>	
21	<code>%%</code>	
22	<code>%part3: RGB to Gray</code>	
23	<code>figure</code>	
24	<code>picture=rgb2gray(picture);</code>	
25	<code>figure</code>	
26	<code>subplot(1,2,1)</code>	
27	<code>imshow(picture)</code>	
28		
29	<code>%%</code>	
30	<code>% part4: Gray to Binary</code>	
31	<code>threshold = graythresh(picture);</code>	
32	<code>picture = ~imbinarize(picture,threshold);</code>	
33	<code>subplot(1,2,2)</code>	
34	<code>imshow(picture)</code>	
35		

37	%part5: removing small objects + background	
38	picture=imresize(picture,[300 1450]);	
39	figure	
40	[row,column,~] = size(picture);	
41	picture = bwareaopen(picture,2000);	
42	subplot(1,3,1)	
43	imshow(picture)	
44	background=bwareaopen(picture,18*column);	
45	subplot(1,3,2)	
46	imshow(background)	
47	picture2=picture-background;	
48	subplot(1,3,3)	
49	imshow(picture2)	
50	%%	
51	%part6: labeling picture	
52	figure	
53	imshow(picture2)	
54	[L,Ne]=bwlabel(picture2);	
55	propied=regionprops(L, 'BoundingBox');	
56	hold on	
57	for n=1:size(propied,1)	
58	rectangle('Position',propied(n).BoundingBox,'EdgeColor','g','LineWidth',2)	
59	end	
60	hold off	
61	%%	
62	% part7: Loading the mapset	
63	load PTRAININGSET;	
64	totalLetters=size(PTRAIN,2);	
65	%%	
66	% part8: correlation of each part with the mapset	
67	final_output=[];	
68	t=[];	
69	for n=1:Ne	
70	[r,c]=find(L==n);	
71	Y=picture2(min(r):max(r),min(c):max(c));	
72	Y=imresize(Y,[60,50]);	
73	ro=zeros(1,totalLetters);	
74	for k=1:totalLetters	
75	ro(k)=corr2(PTRAIN{1,k},Y);	
76	end	
77	[MAXRO,pos]=max(ro);	
78	if MAXRO>.50	
79	out=cell2mat(PTRAIN(2,pos));	
80	final_output=[final_output , out];	
81	end	
82	end	
83	%%	
84	% part8: Printing the plate	
85	file = fopen('number_Plate.txt', 'wt');	
86	fprintf(file,'%s\n',final_output);	
87	fclose(file);	
88	winopen('number_Plate.txt')	

و فانکشن که با آن پلاک تشخیص داده می شود:

```
1 function plateSizeBound = detectplate_bluestrip(picture_full, open_figure)
2     blueStripFile = 'bluestrip.png';
3     RESIZE_WIDTH = 800;
4     RESIZE_LENGTH = 600;
5     ERR_MARGIN = 10;
6     BLUE2PLATE_RATIO = 14;
7     threshold = 0.5;
8
9     if nargin < 2
10         open_figure = true;
11     end
12
13     bluestrip = imread(blueStripFile);
14     picture = imresize(picture_full, [RESIZE_LENGTH, RESIZE_WIDTH]);
15     ratio = size(picture_full, 1) / size(picture, 1);
16
17     [~, corr_max, detection] = rgb_corr2(bluestrip, picture);
18
19     fullSizeBound = [round((detection(1) - ERR_MARGIN) * ratio), ...
20                     round((detection(2) - ERR_MARGIN) * ratio), ...
21                     round((detection(3) + 2 * ERR_MARGIN) * ratio), ...
22                     round((detection(4) + 2 * ERR_MARGIN) * ratio)];
23
24     plateSizeBound = fullSizeBound;
25     plateSizeBound(3) = BLUE2PLATE_RATIO * detection(3) * ratio;
26
27     if open_figure
28         figure('Name', 'Blue Strip Search');
29         subplot(2, 2, 1)
30         imshow(picture)
31         title('Picture')
32         subplot(2, 2, 2)
33         imshow(bluestrip)
34         title('Template')
35         subplot(2, 2, 3)
36         imshow(picture_full)
37         hold on
38         rectangle('Position', fullSizeBound, 'edgecolor', 'r', 'linewidth', 2);
39         rectangle('Position', plateSizeBound, 'edgecolor', 'g', 'linewidth', 1);
40         if corr_max < threshold
41             title('Match [Failed]')
42         else
43             title('Match [Success]')
44         end
45     end
46
47     if corr_max < threshold
48         plateSizeBound = [];
49     end
50 end
```

```

52 function [corr_mix, corr_max, detection] = rgb_corr2(template, pic)
53     corrRed = normxcorr2(template(:, :, 1), pic(:, :, 1));
54     corrGrn = normxcorr2(template(:, :, 2), pic(:, :, 2));
55     corrBlu = normxcorr2(template(:, :, 3), pic(:, :, 3));
56     corr_mix = (corrRed + corrGrn + corrBlu) / 3;
57     [corr_max, corrIndex] = max(abs(corr_mix(:)));
58     [peakY, peakX] = ind2sub(size(corr_mix), corrIndex(1));
59     corr_offset = [peakX - size(template, 2), peakY - size(template, 1)];
60     detection = [corr_offset(1), corr_offset(2), size(template, 2), size(template, 1)];
61 end

```

عملکرد با پلاک داده شده:

Picture



Template



Match [Success]

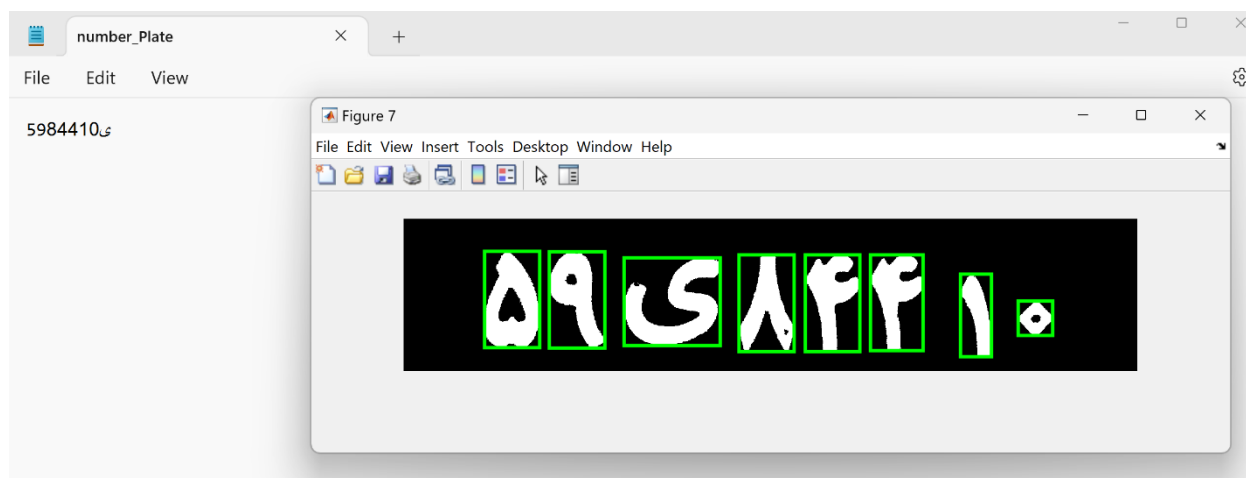




final_output

'84410ی59'

به علت نبود فونت فارسی در متلب، پلاک از راست به چپ (یعنی خلاف جهت اعداد آن) ذخیره شده است.



به علت نبودن فونت فارسی در notes، حرف فارسی سمت راست کل اعداد پلاک نوشته شده است.