

گزارش تمرین کامپیوتری 2 درس سیگنال‌ها و سیستم‌ها

بابک حسینی محتشم 810101408

محمدسینا پرویزی مطلق 810101394

1403/7

بخش اول (

۱- با کمک تابع `uigetfile` پنجره جدیدی باز می‌کنیم تا کاربر تصویر مورد نظر را انتخاب کند.

```
1 clc;close all;clear;
2 % 1. Load the target image
3 [file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');
4 target=imread([path file]);
5 figure;
6 imshow(target);
7
```

تصویر یکی از پلاک‌های استفاده شده برای تست:



۲- با کمک تابع `imresize` ابعاد تصویر را 300 در 500 پیکسل می‌کنیم.

```
8 % 2. Resize the target image
9 WIDTH=300;
10 HEIGHT=500;
11 target=imresize(target,[WIDTH,HEIGHT]);
12 imshow(target);
```

تصویر پلاک پس از تغییر ابعاد آن:



۳- با استفاده از رابطه داده شده عکس رنگی را به عکس سیاه و سفید تبدیل می‌کنیم.

```
1 function grayed_image=mygrayfun(colored_image)
2   grayed_image=0.299*colored_image(:,:,1)+0.578*colored_image(:,:,2)+0.114*colored_image(:,:,3);
3 end
```

تصویر سیاه و سفید شده:



۴- با تست کردن مقادیر مختلف thresh روی ۳ داده تست، متوجه شدیم مقدار ۹۰ به خوبی برای مانع عکس را باینری می‌کند.

```

18 % 4. Binarizing the target image
19 thresh=90;
20 target=mybinaryfun(target,thresh);
21 imshow(target*255);
22 target=~target;
23 imshow(target*255);

```

در تابع mybinaryfun ابتدا پیکسل‌هایی که بیشتر از thresh و هستند را ۱ و پیکسل‌های کمتر از آن را ۰ می‌کنیم و سپس برای نزدیک‌تر بودن تعداد پیکسل‌های سیاه و سفید، اگر پیکسلی دقیقاً برابر thresh باشد آن را برابر مقداری که تعداد کمتری از آن داریم می‌گذاریم.

```

1 function out_image=mybinaryfun(in_image,thresh)
2     out_image=in_image;
3     w_ind=out_image>thresh;
4     b_ind=out_image<thresh;
5     ind=out_image==thresh;
6     out_image(b_ind)=0;
7     out_image(w_ind)=1;
8     if(sum(w_ind,'all')>=sum(b_ind,'all'))
9         out_image(ind)=0;
10    else
11        out_image(ind)=1;
12    end
13 end

```

تصویر پلاک باینری شده:



۵- حال در تابع `myremovecom` سعی می‌کنیم هم از نویز عکس بکاهیم و هم بعضی قسمت‌های کوچک خارج از پلاک را حذف کنیم.

```
25 % 5. Removing clusters with less than n pixels from the target image
26 target=myremovecom(target,300);
27 imshow(target*255);
```

خروجی بدین شکل می‌شود:



تابع `myremovecom` خوشه‌ها را با تابع `myfindclusters` خوشه‌های موجود در تصویر باینری را می‌یابد و مقدار تمام خوشه‌هایی که حداکثر n دارند را صفر می‌کند.

```

1 function out_image=myremovecom(in_image,n)
2     out_image=in_image;
3     clusters_ind=myfindclusters(in_image);
4     for i=1:length(clusters_ind)
5         if(size(clusters_ind{i},1)<=n)
6             ind=sub2ind(size(out_image),clusters_ind{i}(:,1),clusters_ind{i}(:,2));
7             out_image(ind)=0;
8         end
9     end
10 end

```

تابع `myfindclusters` پیکسل‌های تصویر را پیمایش می‌کند تا وقتی که به یک عدد ۱ برسد و سپس تابع `mybfs` را روی آن یک صدا می‌زند تا خوشه مربوط به آن یک را پیدا کند و آن خوشه را کلاً از تصویر حذف می‌کند و به همین ترتیب تمام خوشه‌ها را پیدا می‌کند.

```

1 function clusters=myfindclusters(in_image)
2     bfs_image=in_image;
3     clusters={};
4     num_obj=1;
5     [WIDTH,HEIGHT]=size(in_image);
6     for j=1:HEIGHT
7         for i=1:WIDTH
8             if(bfs_image(i,j)==1)
9                 [new_obj_ind,bfs_image]=mybfs(bfs_image,i,j);
10                clusters{num_obj}=new_obj_ind;
11                num_obj=num_obj+1;
12            end
13        end
14    end
15 end

```

تابع `mybfs` با اجرای الگوریتم `bfs` با شروع از یک مقدار ۱ به عنوان یکی از اعضای خوشه، بقیه اعضای خوشه را پیدا می‌کند و سپس مقدار اعضای آن خوشه را صفر می‌کند و تصویر بدون آن خوشه و جایگاه عناصر خوشه را برمی‌گرداند.

```

1 function [cluster,out_image]=mybfs(in_image,i,j)
2     out_image=in_image;
3     [WIDTH, HEIGHT]=size(in_image);
4     out_image(i,j)=0;
5     cluster=[i j];
6     ind=1;
7     while(ind<=size(cluster,1))
8         i=cluster(ind,1);
9         j=cluster(ind,2);
10        ind=ind+1;
11        for a=-1:1
12            if(i+a>WIDTH || i+a<=0)
13                continue;
14            end
15            for b=-1:1
16                if(j+b>HEIGHT || j+b<=0)
17                    continue;
18                end
19                if(a==0 && b==0)
20                    continue;
21                end
22                if(out_image(i+a,j+b)==1)
23                    out_image(i+a,j+b)=0;
24                    cluster=[cluster;i+a,b+j];
25                end
26            end
27        end
28    end
29 end

```

۶- با کمک تابع mysegmentation تصویر را segment بندی می‌کنیم.

```

29 % 6. Segmenting the clusters
30 target=mysegmentation(target);
31 cluster_target=zeros(size(target));

```

این تابع مقدار هر یک از خوشه‌های موجود را مقدار منحصر به فردی از ۱ تا تعداد خوشه‌ها شماره گذاری می‌کند.

```

1 function out_image=mysegmentation(in_image)
2     out_image=zeros(size(in_image));
3     clusters_ind=myfindclusters(in_image);
4     for i=1:length(clusters_ind)
5         ind=sub2ind(size(out_image),clusters_ind{i}(:,1),clusters_ind{i}(:,2));
6         out_image(ind)=i;
7     end
8 end

```

۷- حال اعداد موجود در mapset را می‌خوانیم و با کمک تابع mydecisionmaker شماره پلاک را به دست می‌آوریم.

```
33 % 7. Extract letters and numbers
34 files=dir('MapSetEng');
35 len=length(files)-2;
36 TRAIN=cell(2,len);
37
38 for i=1:len
39     TRAIN{1,i}=imread([files(i+2).folder,'\ ',files(i+2).name]);
40     TRAIN{2,i}=files(i+2).name(1);
41 end
42 save TRAINSET TRAIN;
43
44 decision=mydecisionmaker(target);
```

تابع mydecisionmaker به ازای هر خوشه، همبستگی آن را با تمام حروف mapset مقایسه می‌کند و هر حرفی که بیشترین همبستگی را داشت به عنوان حرف آن خانه از پلاک ذخیره می‌کند.

```
1 function decision=mydecisionmaker(in_image)
2     load TRAINSET TRAIN;
3     n_seg=max(in_image,[],"all");
4     decision=[];
5     for n=1:n_seg
6         [row,col]=find(in_image==n);
7         seg_image=zeros(size(in_image));
8         seg_image(sub2ind(size(seg_image),row,col))=1;
9         Y=seg_image(min(row):max(row),min(col):max(col));
10        imshow(Y);
11        Y=imresize(Y,[42,24]);
12        imshow(Y);
13        ro=zeros(1,length(TRAIN));
14        for k=1:length(TRAIN)
15            ro(k)=corr2(TRAIN{1,k},Y);
16        end
17        [MAXRO,pos]=max(ro);
18        if MAXRO>.47
19            out=TRAIN{2,pos};
20            decision=[decision out];
21        end
22    end
23    end
```

۸- در آخر شمارخ پلاکی که mydecisionmaker به دست آورد را در فایلی ذخیره کرده و نمایش می‌دهیم.

```
46 % 8. Report decision
47 display(decision);
48 file = fopen('number_plate.txt', 'wt');
49 fprintf(file,'%s\n',decision);
50 fclose(file);
51 winopen('number_Plate.txt')
```

برای پلاک اول به درستی شماره پلاک در می‌آید.



همچنین برای پلاک سوم نیز شماره پلاک به درستی در می‌آید.



```
decision =
```

```
'UP14CB7145'
```

number_plate.txt - Notepad

File Edit Format View Help

UP14CB7145

ولی برای پلاک دوم، به جای عدد 0، حرف o تشخیص داده می شود.



```
decision =
```

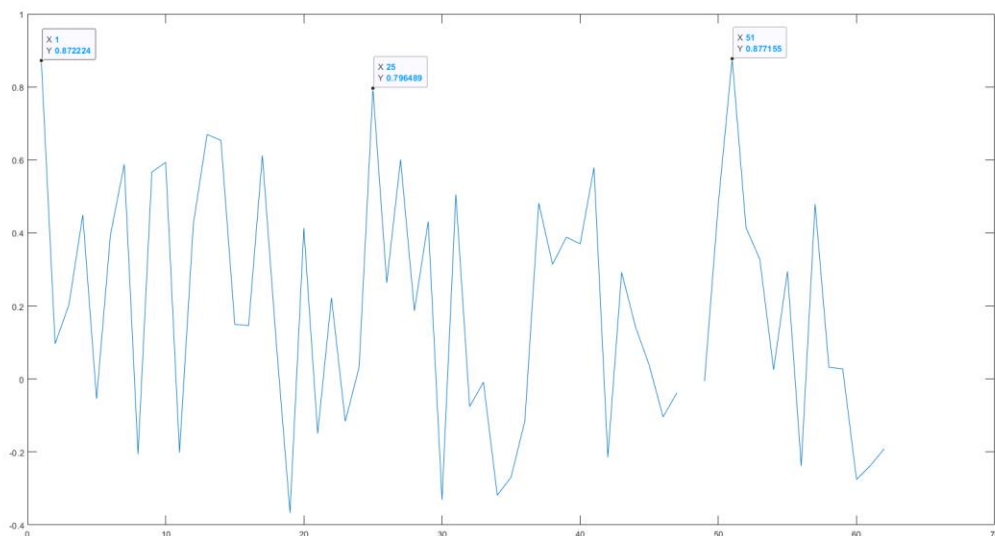
```
'DL2CAdo311'
```

number_plate.txt - Notepad

File Edit Format View Help

DL2CAdo311

البته می‌توان در نمودار همبستگی دید که مقدار همبستگی برای سه حرف مقدار قابل توجهی است.



با بررسی mapset متوجه می‌شویم که این سه مقدار برای حروف 0,o,O است که به دلیل شباهت بسیار بالای این سه حرف با ممکن است آن‌ها را اشتباه تشخیص دهیم ولی چون یکی از آن‌ها عدد، یکی حرف کوچک و دیگری حرف بزرگ است، می‌توانیم با بررسی فرمت پلاک در کشور مورد نظر، حرف درست را تشخیص دهیم.

بخش دوم)

در این بخش از کدهای قسمت قبل کردیم و تنها از تابع آماده graythresh متلب به جای عدد دلخواه استفاده کردیم که نتایج بهتری بگیریم. همچنین مقدار threshold برای حذف نویز را به 1000 افزایش دادیم.

```

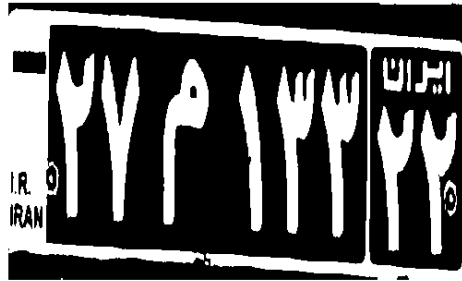
6      imshow(target);
7
8      % 2. Resize the target image
9      WIDTH=300;
10     HEIGHT=500;
11     target=imresize(target,[WIDTH,HEIGHT]);
12     imshow(target);
13
14     % 3. RGB2Gray the target image
15     target=mygrayfun(target);
16     imshow(target);
17
18     % 4. Binarizing the target image
19     thresh=graythresh(target)*255;
20     target=mybinaryfun(target,thresh);
21     imshow(target*255);
22     target=~target;
23     imshow(target*255);
24
25     % 5. Removing clusters with less than n pixels from the target image
26     target=myremovecom(target,1000);
27     imshow(target*255);
28
29     % 6. Segmenting the clusters
30     target=mysegmentation(target);
31     cluster_target=zeros(size(target));
32
33     % 7. Extract letters and numbers
34     files=dir('MapSetFa2');
35     len=length(files)-2;
36     TRAIN=cell(2,len);
37
38     for i=1:len
39         TRAIN{1,i}=imread([files(i+2).folder,'\ ',files(i+2).name]);
40         TRAIN{2,i}=files(i+2).name(1);
41     end
42     save TRAININGSET TRAIN;
43
44     decision=mydecisionmaker2(target);
45
46     % 8. Report decision
47     display(decision);
48     file = fopen('number_plate.txt', 'wt');
49     fprintf(file,'%s\n',decision);
50     fclose(file);
51     winopen('number_Plate.txt')

```

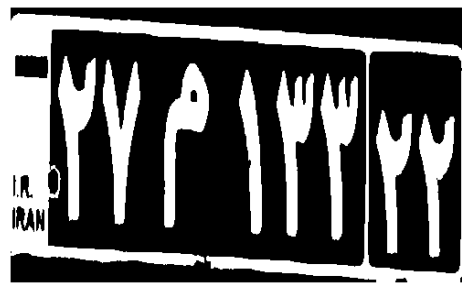
تصویر اصلی:



تصویر باینری شده:



تصویر پس از حذف قسمت‌های کوچک:



همچنین لازم بود تفاوت‌هایی در تابع mydecisionmaker ایجاد کنیم پس تابع mydecisionmaker2 را ساختیم که تغییر یافته تابع mydecisionmaker است. اولاً مپ ستی که ساختیم شامل عکس‌های ۳۲ در ۳۲ پیکسل بود پس طول و عرض تصویر را از Trainset به دست می‌آوریم و تصاویر را به آن سایز تغییر می‌دهیم. همچنین پس‌زمینه تصاویر مپ ست مشکی سفید است پس تصویر پلاک را بیت به بیت نقیض می‌کنیم تا مشابه تصاویر مپ ست شود.

```

1 function decision=mydecisionmaker2(in_image)
2     load TRAININGSET TRAIN;
3     n_seg=max(in_image,[],"all");
4     decision=[];
5     [width,height]=size(TRAIN{1,1});
6     for n=1:n_seg
7         [row,col]=find(in_image==n);
8         seg_image=zeros(size(in_image));
9         seg_image(sub2ind(size(seg_image),row,col))=1;
10        seg_image=~seg_image;
11        Y=seg_image(min(row):max(row),min(col):max(col));
12        imshow(Y);
13        Y=imresize(Y,[width,height]);
14        imshow(Y);
15        ro=zeros(1,length(TRAIN));

```

چون می‌دانیم در پلاک‌های ایران ابتدا دو عدد می‌آید سپس یک حرف و دوباره پنج عدد، پس می‌توانیم برای افزایش دقت، در ابتدا همبستگی را تنها با اعداد حساب کنیم و بعد از اینکه دو عدد را تشخیص دادیم، تا تشخیص یک حرف همبستگی را با حروف حساب کنیم و از آن جا به بعد هم دوباره همبستگی را تنها با اعداد حساب کنیم. در `mapset` ما تصویر ابتدایی حرف هستند و بقیه تصاویر عدد پس کد را به این شکل تغییر دادیم.

```

16        k=13:length(TRAIN);
17        len_decision=length(decision);
18        if(len_decision==2)
19            k=1:12;
20        end
21        for i=k
22            ro(i)=corr2(TRAIN{1,i},Y);
23        end

```

متلب حروف فارسی را از چپ به راست چاپ می‌کند پس نیاز بود تغییراتی در بخش آخر تابع نیز ایجاد کنیم.

```

24        [MAXRO,pos]=max(ro);
25        if MAXRO>.47
26            out=TRAIN{2,pos};
27            if(length(decision)<2)
28                decision=[decision,out];
29            elseif(length(decision)==2)
30                decision=[out,decision];
31            else
32                decision=[decision(1:len_decision-3),out,decision(len_decision-2:len_decision)];
33            end
34        end
35    end
36 end

```

برای ساخت mapset فونت‌های زیادی را بررسی کردیم و نزدیک‌ترین فونتی که به فونت پلاک خودروها یافتیم، فونت B Roya Bold بود. پس ابتدا MapSetFa را با نوشتن هر حرف در برنامه فتوشاپ با اندازه یکسان، این mapset را امتحان کردیم ولی مقدار همبستگی پایین تر مقدار threshold بود و اگر threshold را از 47 به 3 کاهش می‌دادیم، خروجی بدین شکل می‌شد:



پس برای افزایش دقت، MapSetFa0 را ایجاد کردیم. در این مپ ست برخلاف حالت قبل اندازه فونت حروف را یکسان قرار ندادیم و تا حد امکان بزرگ کردیم که قسمت بیشتری از تصویر شامل حرف باشد و در این حالت مقدار همبستگی‌ها کمی بیشتر شد ولی باز هم باید مقدار threshold را 0.3 می‌گذاشتیم و خروجی بدین شکل شد:



سپس MapSetFa1 را تشکیل دادیم که در آن علاوه بر افزایش فونت تا حد امکان، طول و عرض حرف را هم به صورت دستی تغییر دادیم تا بخش بیشتری از تصاویر را بپوشاند. بااینکه با این کار باعث شدیم که تناسب حروف به هم بریزد ولی چون با تغییر اندازه در متلب هم این اتفاق رخ می‌داد نتایج خیلی بهتری گرفتیم و حتی با $\text{threshold}=0.7$ خروجی بدین شکل بود:



با مشاهده نمدار همبستگی متوجه شدیم که همبستگی دو عدد ۲ و ۳ بسیار به هم نزدیک بود پس برای رفع تشخیص نادرست، ۲ و ۳ را از تصویر پلاک استخراج کرده و جایگزین ۲ و ۳ موجود در MapSetFa1 کردیم تا MapSetFa2 بوجود بیاید و برنامه با این مپ ست با $\text{threshold}=0.7$ به درستی پلاک را تشخیص می‌دهد:



بخش سوم)

تصویر ابتدایی خودرو:



ابتدا مانند قبل تصویر را خوانده و سپس اندازه آن را تغییر می‌دهیم. با بررسی سه تصویر داده شده، چون ابعاد کوچکترین تصویر حدود ۶۰۰ در ۹۰۰ پیکسل بود، ابعاد تصویر ورودی را به این اعداد تغییر می‌دهیم. سپس تصویر را سیاه‌وسفید می‌کنیم و بعد باینری می‌کنیم. در ابتدا مقدار **threshold** را با کمک تابع **graythresh** مطلب به دست می‌آوریم ولی در تصویر اول باعث می‌شد ۵ به پلاک بچسبد و قابل تشخیص نباشد پس دوباره از عدد ثابت ۹۰ استفاده کردیم.


```

1      clc;close all;clear;
2      % 1. Load the target image
3      [file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');
4      target=imread([path file]);
5      figure;
6      imshow(target);
7
8      % 2. Resize the target image
9      WIDTH=600;
10     HEIGHT=900;
11     target=imresize(target,[WIDTH,HEIGHT]);
12     imshow(target);
13
14     % 3. RGB2Gray the target image
15     target=mygrayfun(target);
16     imshow(target);
17
18     % 4. Binarizing the target image
19     thresh=90;
20     target=mybinaryfun(target,thresh);
21     imshow(target*255);
22     target=~target;
23     imshow(target*255);
24     picture=target;
25

```

سپس سعی می‌کنیم تعدادی از قسمت‌هایی که پلاک نیستند را حذف کنیم. پس از تابع myremovecom2 استفاده می‌کنیم که شبیه myremovecom است با این تفاوت که خوشه‌هایی که بیشتر از m پیشکسل دارند را حذف می‌کند.

```

% 6. Remove noise from the edges
target=myremovecom2(target,5,900);
imshow(target*255);

```

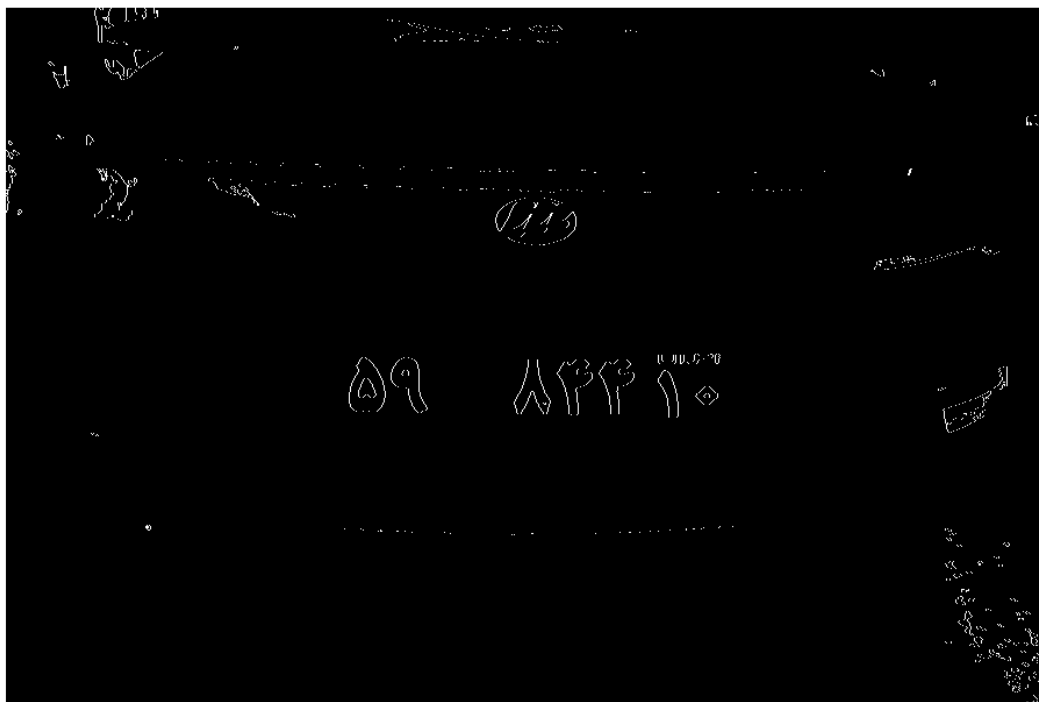
پس از استفاده از این تابع به تصویر زیر می‌رسیم. می‌توان دید که حرف ی هم از پلاک حذف شده که می‌توان با تغییر پارامترهای تابع مانع حذف آن شد ولی چون برای یافتن ناحیه پلاک تنها عدد اول و آخر کافی است، با همین تصویر ادامه می‌دهیم.



ما هر دو روش همبستگی با پرچم ایران و به دست آوردن لبه‌ها را امتحان کردیم و روش دوم بهتر جواب داد. همچنین لبه‌های عمودی، افقی و هر دو باهم را امتحان کردیم و لبه‌های عمودی بهتر جواب دادند که قابل انتظار است چون اکثر اعداد فارسی لبه‌های عمودی بیشتری دارند در حالی که در تصاویر جاهای خارج از پلاک لبه‌های افقی بیشتر بودند. برای به دست آوردن لبه‌های افقی تصویر را از شیف‌ت یافته آن در بعد دوم کم کردیم.

```
26 % 5. Find the vertical edges in the target image
27 target2=target;
28 target2(1:size(target,1),1:size(target,2)-1)=target(1:size(target,1),2:size(target,2));
29 imshow(target2);
30 dif_target=abs(target2-target);
31 imshow(dif_target);
```

تصویر خودرو بعد از تغییرات ذکر شده:



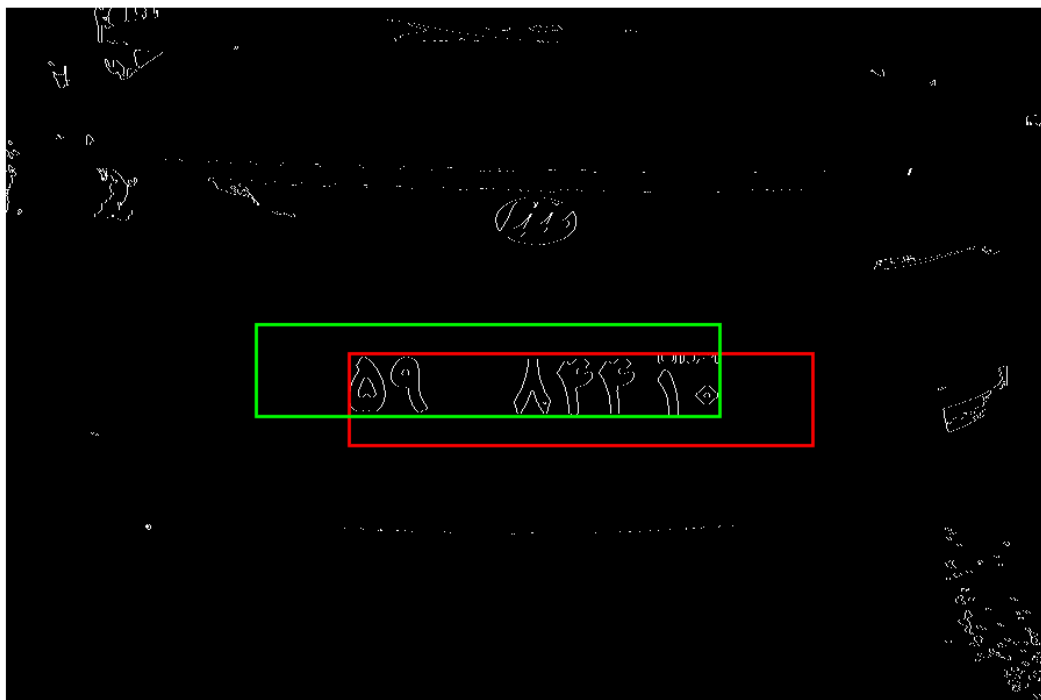
ما با بررسی سه تصاویر داده شده متوجه شدیم ابعاد پلاک ۲های خودرو در تصاویر کمتر از ۸۰ در ۴۰۰ پیکسل است. مستطیلی به ابعاد ۸۰ در ۴۰۰ را در تصویر حرکت می‌دهیم و مجموع لبه‌ها را حساب می‌کنیم و قسمتی که بیشتری مجموع را داشت به عنوان پلاک در نظر می‌گیریم. برای به دست آوردن انتهای مستطیل نیز همین کار را بار دیگر با حرکت مستطیل از انتها به ابتدای تصویر تکرار می‌کنیم و دو نقطه به دست آمده را به عنوان ابتدا و انتهای پلاک در نظر می‌گیریم.

```

37 % 7. Find area with most edges
38 box_width=80;
39 box_height=400;
40 max_i1=1;
41 max_j1=1;
42 max_dif=0;
43 for i = 1:(WIDTH - box_width + 1)
44     for j = 1:(HEIGHT - box_height + 1)
45         submatrix = dif_target(i:i + box_width - 1, j:j + box_height - 1);
46         dif = sum(submatrix(:));
47         if dif >= max_dif
48             max_dif = dif;
49             max_i1 = i;
50             max_j1 = j;
51         end
52     end
53 end
54 max_i2=WIDTH - box_width + 1;
55 max_j2=HEIGHT - box_height + 1;
56 max_dif=0;
57 for i = fliplr(box_width:WIDTH)
58     for j = fliplr(box_height:HEIGHT)
59         submatrix = dif_target(i - box_width + 1:i, j - box_height + 1:j);
60         dif = sum(submatrix(:));
61         if dif >= max_dif
62             max_dif = dif;
63             max_i2 = i;
64             max_j2 = j;
65         end
66     end
67 end
68 imshow(dif_target);
69 hold on;
70 rectangle('Position', [max_j1,max_i1,box_height-1,box_width-1], 'EdgeColor', 'r', 'LineWidth', 2);
71 rectangle('Position', [max_j2-box_height+1,max_i2-box_width+1,box_height-1,box_width-1], 'EdgeColor', 'g', 'LineWidth', 2);
72 hold off;

```

تصویر پلاک یافت شده:



بقیه مراحل مشابه قبل است:

```
61 % 8. Resize the plate
62 target=picture(max_i:max_i+box_width-1,max_j:max_j+box_height-1);
63 WIDTH=300;
64 HEIGHT=500;
65 target=imresize(target,[WIDTH,HEIGHT]);
66 imshow(target);
67
68 % 9. Removing clusters with less than n pixels from the plate
69 target=myremovecom(target,500);
70 imshow(target*255);
71
72 % 10. Segmenting the clusters
73 target=mysegmentation(target);
74 cluster_target=zeros(size(target));
75
76 % 11. Extract letters and numbers
77 files=dir('MapSetFa2');
78 len=length(files)-2;
79 TRAIN=cell(2,len);
80
81 for i=1:len
82     TRAIN{1,i}=imread([files(i+2).folder,'\ ',files(i+2).name]);
83     TRAIN{2,i}=files(i+2).name(1);
84 end
85 save TRAININGSET TRAIN;
86
87 decision=mydecisionmaker3(target);
88
89 % 12. Report decision
90 display(decision);
91 file = fopen('number_plate.txt', 'wt');
92 fprintf(file, '%s\n', decision);
93 fclose(file);
94 winopen('number_Plate.txt')
```

با اینکه پلاک را به درستی یافتیم ولی حروف به درستی استخراج نمی‌شدند که برای رفع این مشکل چند تغییر ایجاد کردیم. تصاویر عدد ۵ و حرف ه که در mapset مشابه هم بودند را تغییر دادیم.

همچنین تغییراتی در تابع mydecisionmaker2 ایجاد کردیم و تابع mydecisionmaker3 را تشکیل دادیم. یکی از مشکلاتی که متوجه شدیم این بود که با حساب کردن ضرب داخلی یا همان همبستگی چون عکس‌ها باینری صفر و یک هستند تصویری از مپ ست که اشتراک بیشتری با تصویر داشته باشد به عنوان حرف انتخاب

می‌شود و بابت ناحیه‌هایی که با هم اختلاف دارند از ضرب داخلی آن کم نمی‌شود پس برای رفع این مشکل، ابتدا تصاویر را دو برابر کرده و منهای یک می‌کنیم تا باینری منفی یک و مثبت یک شوند و حال اگر ضرب داخلی‌شان را حساب کنیم، تفاوت‌ها باعث کاهش ضرب داخلی می‌شوند.

```

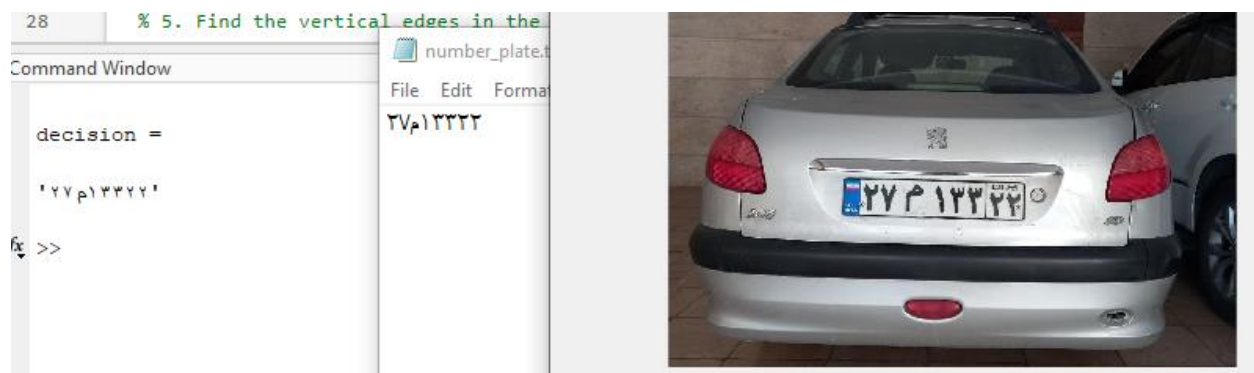
1 function decision=mydecisionmaker3(in_image)
2     load TRAININGSET TRAIN;
3     n_seg=max(in_image,[],"all");
4     decision=[];
5     [width,height]=size(TRAIN{1,1});
6     for n=1:n_seg
7         [row,col]=find(in_image==n);
8         seg_image=zeros(size(in_image));
9         seg_image(sub2ind(size(seg_image),row,col))=1;
10        seg_image=~seg_image;
11        Y=seg_image(min(row):max(row),min(col):max(col));
12        imshow(Y);
13        Y=imresize(Y,[width,height]);
14        imshow(Y);
15        ro=zeros(1,length(TRAIN));
16        k=13:length(TRAIN);
17        len_decision=length(decision);
18        if(len_decision==2)
19            k=1:12;
20        end
21        for i=k
22            ro(i)=sum((im2double(TRAIN{1,i})*2-1) .* (im2double(Y)*2-1),'all');
23        end
24        [MAXRO,pos]=max(ro);
25        if MAXRO>.5*1024
26            out=TRAIN{2,pos};
27            if(length(decision)<2)
28                decision=[decision,out];
29            elseif(length(decision)==2)
30                decision=[out,decision];
31            else
32                decision=[decision(1:len_decision-3),out,decision(len_decision-2:len_decision)];
33            end
34        end
35    end
36 end

```

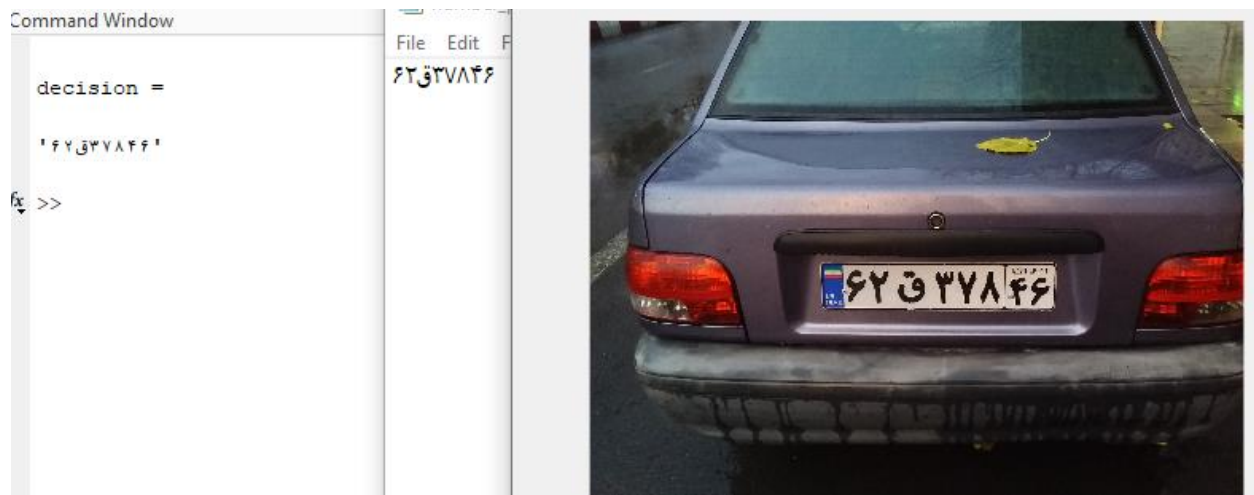
خروجی تصویر اول:



خروجی تصویر دوم:



خروجی تصویر سوم:



بخش چهارم)

در این بخش، ایده ما محاسبه سرعت متوسط یک خودرو از روی دو فریم انتخاب شده از یک فیلم است. به کمک مقایسه دو فریم، تعداد پیکسل‌هایی که بین دو فریم جابه‌جا شده‌اند را محاسبه می‌کنیم و بر اساس فاصله زمانی بین این دو فریم، سرعت خودرو را به دست می‌آوریم.

در ابتدا یک ویدیو را انتخاب می‌کنیم. سپس دو فریم با فاصله زمانی مشخص (در اینجا فریم‌های ۴۰ و ۶۰) از ویدئو استخراج می‌شود. هدف این است که جابه‌جایی تصویر خودرو بین این دو فریم محاسبه شود (در واقع تفاضل دو فریم).

فریم‌ها به اندازه‌های 600×900 تغییر سایز می‌دهیم و سپس آنها را به (grayscale) تبدیل می‌کنیم تا پردازش تصویر ساده‌تر شود. برای استخراج بهتر لبه‌های خودرو تصاویر باینری می‌شوند. آستانه‌ای برای تبدیل پیکسل‌ها به سیاه و سفید تنظیم می‌شود (در اینجا مقدار ۹۰) و سپس تصاویر معکوس می‌شوند تا سیاه و سفید با هم جابه‌جا شوند. سپس با منهای کردن بردار دو فریم، جابه‌جایی لبه‌ها را در دو فریم به دست می‌آوریم. به دلیل لرزش دست، جاهایی غیر از خودرو هم جابه‌جایی وجود دارد پس با تابع `myremovecom` این نویزها را حذف می‌کنیم. پس از شناسایی ناحیه مربوط به خودرو، جابه‌جایی دقیق پیکسل‌ها در آن ناحیه بین دو فریم محاسبه می‌شود. برای این کار، مانند بخش ۳ یک مستطیل با طول و عرض مشخص را در تصویر جابه‌جا می‌کنیم تا ناحیه‌ای که بیشترین تغییرات را داشته پیدا کنیم. سپس با روش بخش ۳ با شیف‌ت دادن تصویر لبه‌ها را پیدا می‌کنیم. با این تفاوت که در این بخش تصویر را i بار شیف‌ت می‌دهیم. با هر بار شیف‌ت دادن جمع لبه‌های ناحیه‌ای که در مرحله قبل به دست آوردیم را با آن مقایسه می‌کنیم. مقدار i ‌ای که به ازای آن اختلاف لبه در تصویر اول که اختلاف دو فریم بود و تصویر دوم که اختلاف عکس و شیف‌ت یافته آن بود کمینه می‌شود را برمی‌گردانیم. این i تعداد تقریبی پیکسلی است که خودرو بین این دو فریم جابه‌جا شده. در نهایت با توجه به جابه‌جایی پیکسل‌ها، سرعت خودرو بر حسب پیکسل بر ثانیه (فریم) محاسبه می‌شود.

```

hold off;

n=100;
min_i=1;
min_val=100000;
sub_dif=sum(dif_target(max_i1:max_i2,max_j1:max_j2),"all");
target=target(max_i1:max_i2,max_j1:max_j2);
for i=1:n
    target3=target;
    target3(1:size(target,1),1:size(target,2)-i)=target(1:size(target,1),i+1:size(target,2));
    dif_target2=abs(target3-target);
    dif_target2=myremovecom(dif_target2,50);
    sub_dif2=sum(dif_target2,"all");
    val=abs(sub_dif2-sub_dif);
    if(val<min_val)
        min_val=val;
        min_i=i;
    end
end
end
end

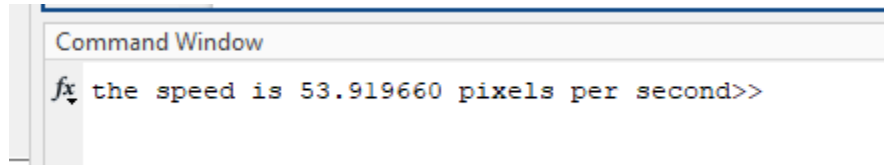
```

```

[min_shift_dif_i,edited_first_frame]=find_plate(first_frame,after_frame);

speed=(min_shift_dif_i)*vid.FrameRate/(last_ind-first_ind);
fprintf("the speed is %f pixels per second", speed);

```



The screenshot shows a MATLAB Command Window with the title "Command Window". The output text is "the speed is 53.919660 pixels per second>>".

باقی کد های این بخش مانند بخش های قبل است.