گزارش تمرین کامپیوتری2

سیگنال ها و سیستم ها (دکتر اخوان)

سامان دوچی طوسی 810101420

پریسا محمدی 810101509

گزارش تمرین کامپیوتری1 سیگنال ها و سیستم ها (دکتر اخوان)

سوال 1

کد بخش 1 و 2:

```
clc
close all;
clear;

% SELECTING THE TEST DATA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');
s=[path,file];
picture=imread(s);
figure
subplot(1,2,1)
imshow(picture)
picture=imresize(picture,[300 500]);
subplot(1,2,2)
imshow(picture)

```

تصویر 1 – کد بخش اول

توضیح خط به خط کد تصویر 1:

1: پاک کردن کامند ویندوز

2: پاک کردن همه تصاویر

3: پاک کردن همه متغیر ها

7: باز کردن انتخاب فایل و نشان دادن متن مورد نیاز

8: ترکیب کردن اسم تصویر با آدرس تصویر

9: خواندن تصویر و ریختن آن داخل ماتریس picture

10: باز کردن پنجره جدید تصویر

11: ساختن ساب پلات داخل پوزیشن 1

12: نشان دادن تصویر اصلی

13: تصویر با اندازه خواسته شده

14: ساختن ساب پلات داخل پوزیشن 2

15: نشان دادن تصویر تغییر یافته

تصویر 2 – خروجی حاصل از اجرای کد

کد بخش 3:

```
function grayImage = mygrayfun(rgbImage)
    % Extract the red, green, and blue channels from the RGB image
    redChannel = rgbImage(:,:,1);
    greenChannel = rgbImage(:,:,2);
    blueChannel = rgbImage(:,:,3);

    % Calculate the grayscale image using the given formula
    grayImage = 0.299 * redChannel + 0.578 * greenChannel + 0.114 * blueChannel;

    % Ensure the output is of the same class as the input
    grayImage = cast(grayImage, 'like', rgbImage);
end

```

تصویر 3 – کد برای خاکستری کردن تصویر

```
%RGB2GRAY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
picture=mygrayfun(picture);
figure
subplot(1,3,1)
imshow(picture)
```

تصویر 4 – استفاده از تابع در کد اصلی

تصویر 5 – خروجی حاصل از اجرای بخش 3

کد بخش 4:

```
function binaryImage = mybinaryfun(grayImage, threshold)
    % Convert grayscale image to binary image using the specified threshold
    binaryImage = grayImage <= threshold;

    % Ensure the output is of the same class as the input
    binaryImage = cast(binaryImage, 'like', grayImage);
end
```

تصویر 6 – کد بخش 4

```
25  % THRESHOLDIG and CONVERSION TO A BINARY IMAGE
26  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
27  threshold = 128; % Example threshold value
28  picture = mybinaryfun(picture,threshold);
29  picture=~picture;
30  subplot(1,3,3)
31  imshow(picture)
32
```

تصویر 7 – استفاده از تابع تصویر 6 در کد اصلی

تصویر 8 – تست کردن تابع بخش 4

کد بخش 5:

```
function cleanedImage = myremovecom(binaryImage, n)
    % Label connected components in the binary image
    [labeledImage, numObjects] = bwlabel(binaryImage);

    % Get properties of connected components
    componentStats = regionprops(labeledImage, 'Area', 'PixelIdxList');

    % Initialize cleanedImage
    cleanedImage = binaryImage;

    % Loop through each component and check its size
    for i = 1:numObjects
        if componentStats(i).Area < n
            % If the component has fewer pixels than n, remove it
            cleanedImage(componentStats(i).PixelIdxList) = 0;
        end
    end
end

```

تصویر 9 – کد تابع حذف اشکال ریز اضافه

توضیح خط به خطی کد:

برای لیبل زدن اجزای به هم چسبیده - Labeled image نگهداری لیبل های هر کامپوننت - 3: استفده از تابع bwlabel

Numobjects نگهداری تعداد کامپوننت های پیدا شده

6: آرایه ای از استراکچر ها

9: انیشیتال کردن کامپوننت ها

12-17: جدا کردن کامپوننت هایی که کمتر از N هستند.

```
% Removing the small objects and background
threshold = 300;
cleanedImage = myremovecom(~picture, threshold);
subplot(1,3,3);
imshow(cleanedImage);
```

تصویر 10 – استفاده از تابع در کد اصلی

تصویر 11 – نتیجه حاصل از اجرا

کد بخش 6

```
function [L, Ne] = mysegmentation(picc)
    flag = 1;
    L = zeros(300, 500);
    for i = 1:500
        for j = 1:300
            if picc(j, i) == 1 && L(j, i) == 0
                S = [j, i];
                L(j, i) = flag;
                while ~isempty(S)
                    current_pixel = S(1, :);
                    S(1, :) = [];
                    for k = -1:1
                        for l = -1:1
                            new_x = current_pixel(1) + k;
                            new_y = current_pixel(2) + l;
                            if new_x >= 1 && new_x <= 300 && new_y >= 1 && new_y <= 500 && picc(new_x, new_y) == 1 && L(new_x, new_y) == 0
                                S = [S; new_x, new_y];
                                L(new_x, new_y) = flag;
                            end
                        end
                    end
                end
                Ne = flag;
                flag = flag + 1;
            end
        end
    end
end
```

تصویر 12 – کد بخش segmentat

بررسی نحوه اجرای خط به خط کد:

این تابع در ابتدا یک عکس به عنوان ورودی گرفته و خروجی آن تعداد ابجکت ها و خود ابجکت ها می باشد

برای لیبل زدن به کامپوننت های متصل به هم هست. و ماتریس l هم برای ذخیره کردن این لیبل ها متغیر flag

روی هر پیکس عکس ورودی حرکت میکنیم. چک میکنیم اگه ولیو پیکسل در تصویر اولیه 1 بوده و هنوز لیبل نخورده ان را داخل میریزیم و این پیکسل را لیبل میزنیم. (در واقع متغییر اس مانند استک عمل میکند و تا متغییر sزمانی که خالی نشده ادامه میدهد)

را از استک پاپ کردن و ان را داخل Current_pixel میریزیم. سپس سراغ 8 همسایه کناری اش میرویم. سپس متغییر S

و چک میکنیم که قبلا لیبل نخورده اند و آن ها را داخی استک پوش کرده و لیبل فلگ را به آن ها میزنیم. زمانی که استک خالی شد فلگ را یکی زیاد کرده و سراغ پیکسل بعدی میرویم و همین الگوریم را تکرار میکنیم.

```
% Segmentation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Labeling connected components
[L, Ne] = mysegmentation(cleanedImage);
propied = regionprops(L, 'BoundingBox');
hold on
for n = 1:size(propied, 1)
    rectangle('Position', propied(n).BoundingBox, 'EdgeColor', 'g', 'LineWidth', 2)
end
hold off
```

تصویر 13 – پیدا کردن نقطه های نزدیک هم

در ابتدا تابع را بر روی تصویر خود صدا میزنیم. خروجی های تابع به ترتیب همه آبجکت های شناسایی شده و تعداد ابجکت های شناسایی شده می باید.

و regionprops استفاده میکنیم که کار این 2 تابع به ترتیب پیدا کردن کوچک ترین کادری که سپس از توابع اماده توانایی فیت شدن به دور ها آبجکت را دارد و کشیدن مستطیلی سبز رنگ به Rectangle دور هر کدام از کادر های پیدا شده می باشد.

تصویر 14 – خروجی حاصل از کد تصاویر 12 و 13

کد بخش 7

```
clc;
clear;
close all;
files=dir('Map Set');
len=length(files)-2;
TRAIN=cell(2,len);

for i=1:len
   TRAIN{1,i}=imread([files(i+2).folder,'\',files(i+2).name]);
   TRAIN{2,i}=files(i+2).name(1);
end

save TRAININGSET TRAIN;

```

تصویر 15 – لود کردن مپ ست

با استفاده از این فایل، مپ ست مورد نیاز برای تشخیص حروف پلاک را لود میکنیم.

در به تعداد حروفی که داخل این دایرکتوری قرار دارد آرایه ای تشکیل میدهیم. سپس هر کدام از تصاویر را میخوانیم

در ادامه هر کدام از حروفی که از پلاک خونده ایم را با هر کدام از ابجکت های مپ ست کوریلیشن میگیریم...

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Loading the mapset
load TRAININGSET;
totalLetters=size(TRAIN,2);

figure
final_output=[];
t=[];
for n=1:Ne

    [r,c]=find(L==n);
    Y=cleanedImage(min(r):max(r),min(c):max(c));
    imshow(Y)
    Y=imresize(Y,[42,24]);
    imshow(Y)
    pause(0.2)


    ro=zeros(1,totalLetters);
    for k=1:totalLetters
        ro(k)=corr2(TRAIN{1,k},Y);
    end

    [MAXRO,pos]=max(ro);
    if MAXRO>.45
        out=TRAIN{2,pos};
        final_output=[final_output out];
    end
end

```

تصویر 16 – پیدا کردن هر کدام از حروف پلاک

توضیح کد:

در ابتدا مپ ست را لود میکنیم.

داخل متغییر Totalletter به تعداد کل متغییر ها را ذخیره میکنیم.

سپس یک تصویر باز میکنیم و متغییر Final_output برای نگه داری کاراکتر های تشخیص داده شده است.

سپس سطر و ستون پیکسلی که متعلق به ان امین ابجکت هست را ذخیره میکنیم.

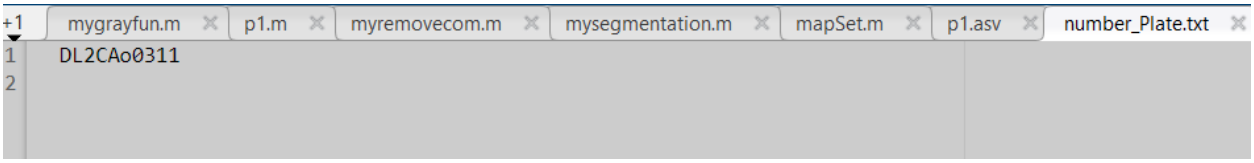
سپس داخل متغیر Y ماکسیمم و مینیمم سطر و ستون را ذخیره میکنیم.

تصویر ان و تغییر سایز پیدا کرده آن را نشان میدهیم. و 0.2 ثانیه صبر میکنیم.

سپس روی تمام کاراکتر های مپ ست حلقه میزنیم و کوریلیشن را محاسبه میکنیم.

سپس ماکسیمم کوریلیشن را برداشته و ایندکس ان را ذخیره میکنیم.

بعد ترش هولدی تعریف میکنیم تا اگر ماکسیمم کوریلیشن از یک حدی کمتر بود آن را اگنور کند.



تصویر 17 – خروجی ذخیره شده فایل تکست

همان طور که مشخص است D باقی حروف به درستی تشخیص داده شده اند. با بررسی کوریلشن ها مشخص یه جز یک حرف

میشود که حرف D هم جز ولیو های بالا هست.

بخش 8 : چک کردن با پلاکی دیگر

تصویر 18 – بررسی درستی کارکرد با پلاکی دیگر

سوال 2

بخش 1) انتخاب تصویر تست

از کاربر خواسته می‌شود تا یک فایل تصویری از طریق یک دیالوگ انتخاب فایل انتخاب کند. تصویر بارگذاری شده (`imread`) و با استفاده از `imshow`نمایش داده می‌شود. سپس تصویر به اندازه‌ی ثابت ۸۰ در ۲۵۰ پیکسل تغییر اندازه داده می‌شود .تغییر اندازه به استانداردسازی ورودی کمک می‌کند تا در مراحل بعدی پردازش آسان‌تر شود.

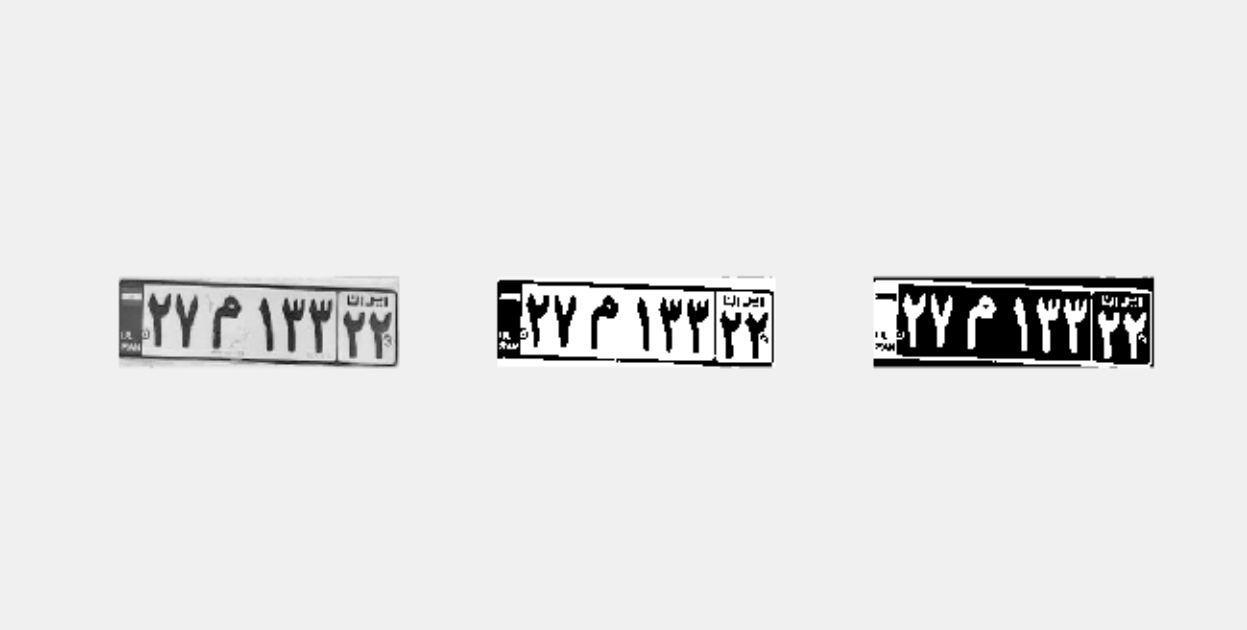
```
% SELECTING THE TEST DATA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[file,path]=uigetfile({'*.jpg;*.bmp;*.png;*.tif'},'Choose an image');
s=[path,file];
picture=imread(s);
figure
subplot(1,2,1)
imshow(picture)
picture=imresize(picture,[80 250]);
subplot(1,2,2)
imshow(picture)
```

تصویر 19 _ انتخاب تصویر پلاک

بخش 2) آستانه‌گذاری و تبدیل به تصویر **خاکستری** و سپس باینری

تصویر رنگی به تصویر خاکستری تبدیل می‌شود با استفاده از تابع .rgb2grayاین مرحله پیچیدگی محاسباتی را کاهش می‌دهد و تنها به شدت نور پیکسل‌ها تمرکز می‌کند .بسیاری از الگوریتم‌های پردازش تصویر، به‌ویژه آن‌هایی که به بخش‌بندی یا آستانه‌گذاری مربوط می‌شوند، با تصاویر خاکستری بهتر کار می‌کنند.

تابع graythreshمقدار آستانه‌ی بهینه‌ای را بر اساس هیستوگرام شدت نور تصویر محاسبه می‌کند. سپس تصویر با استفاده از imbinarizeبه صورت باینری (سیاه و سفید) تبدیل می‌شود، به طوری که پیکسل‌های کمتر از آستانه به ۰ (سیاه) و پیکسل‌های بیشتر به ۱ (سفید) تبدیل می‌شوند. در نهایت تصویر با picture~معکوس می‌شود تا نواحی مورد نظر (مثل حروف) سفید و پس‌زمینه سیاه باشد.این مرحله به جدا کردن نواحی مورد نظر (کاراکترها) از پس‌زمینه کمک می‌کند و فرآیند بخش‌بندی را ساده‌تر می‌سازد.



تصویر 20 _ تصویر تبدیل شده به خاکستری و سپس باینری و مکمل ان

بخش 3) حذف اشیاء کوچک و پس‌زمینه

تابع bwareaopenاشیاء کوچک متصل در تصویر باینری را که کمتر از یک اندازه‌ی مشخص پیکسل (مثلاً ۲۰۰ یا ۴۵۰۰ پیکسل) دارند حذف می‌کند. این کار نویز را کاهش داده و پس‌زمینه را تمیز می‌کند .حذف نویزها و اشیاء نامربوطی که ممکن است در تصویر باینری ظاهر شده باشند، به طوری که فقط نواحی مربوط به کاراکترها باقی بمانند.

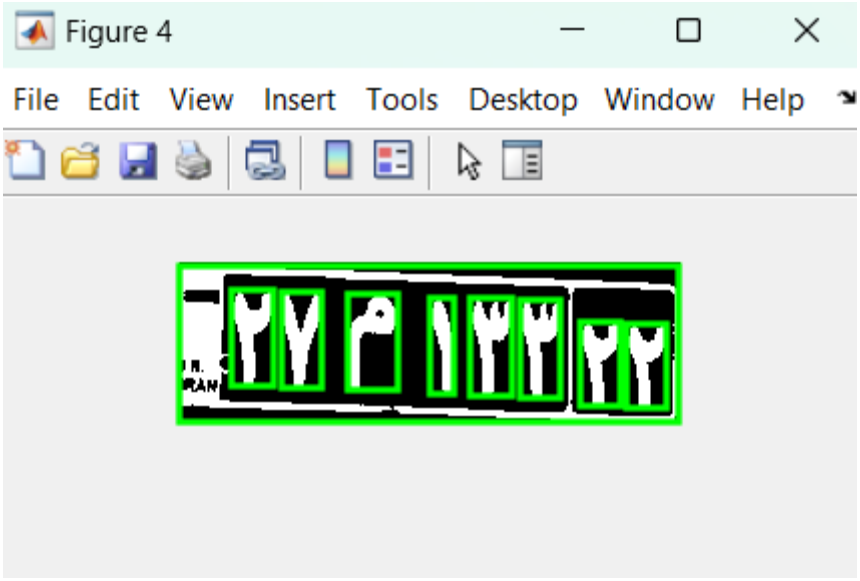
تصویر 21 _ حذف اشیاء کوچک و پس‌زمینه

بخش 4)برچسب‌گذاری اشیاء متصل

تابع bwlabel اجزاء متصل در تصویر باینری را برچسب‌گذاری می‌کند و به هر بخش (مثل حروف یا اعداد) یک برچسب یکتا اختصاص می‌دهد regionprops .جعبه‌های محدود کننده‌ی هر بخش را استخراج می‌کند و rectangleجعبه‌هایی سبز رنگ دور هر ناحیه‌ی برچسب‌گذاری شده رسم می‌کند.این مرحله به شناسایی و جداسازی اجزاء جداگانه در تصویر کمک می‌کند که حروف یا اعداد هستند و هر یک را برای پردازش جداگانه آماده می‌کند.

```
% Labeling connected components
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
imshow(picture2)
[L,Nseg]=bwlabel(picture2);
Cordinate=regionprops(L,'BoundingBox');
hold on
for n=1:Nseg
    rectangle('Position',Cordinate(n).BoundingBox,'EdgeColor','g','LineWidth',2)
end
hold off
```

تصویر 22 _کد برچسب‌گذاری اشیاء متصل

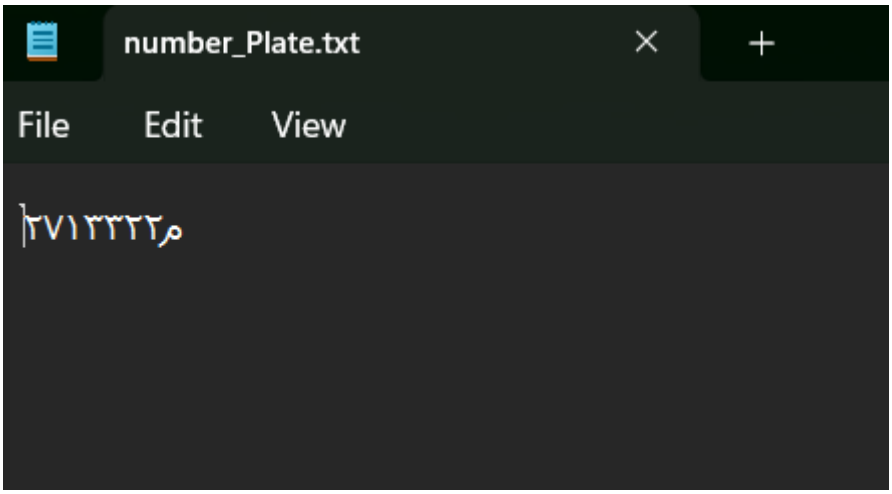


تصویر 23 _ خروجی برچسب‌گذاری اشیاء متصل

بخش 5) شناسایی کاراکتر

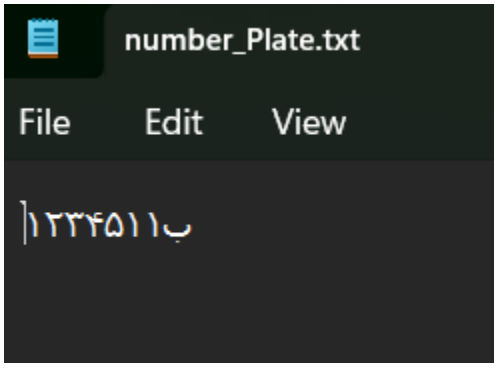
فرآیند شناسایی کاراکترها با بارگذاری مجموعه داده‌ی آموزشی (TRAININGSET) آغاز می‌شود. برای هر بخش برچسب‌گذاری شده(Y) ، سیستم آن را به اندازه‌ی تصاویر آموزشی (۴۲ در ۲۴) تغییر اندازه می‌دهد و سپس همبستگی (corr2) بین بخش و هر کاراکتر در مجموعه‌ی آموزشی را محاسبه می‌کند. کاراکتری که بالاترین همبستگی را دارد (در صورتی که از یک آستانه خاص بزرگ‌تر باشد، مثلاً 0.45) به عنوان خروجی انتخاب می‌شود .همبستگی به عنوان یک روش ساده برای تطابق الگو استفاده می‌شود، که در آن هر بخش با کاراکترهای از پیش ذخیره شده مقایسه شده و مشابه‌ترین کاراکتر انتخاب می‌شود.

کاراکترهای شناسایی شده به یک فایل متنی (number_Plate.txt) با استفاده از توابع fopen, fprintf, وfclose ذخیره می‌شوند. سپس فایل با winopenباز می‌شود. این مرحله نهایی نتیجه‌ی استخراج شده (شماره پلاک) را ذخیره کرده و به کاربر نمایش می‌دهد.



تصویر 24 _ کاراکترهای شناسایی شده

خروجی کد برای نمونه عکس ورودی دیگر:



سوال 3

بخش 1) گام ۱: انتخاب تصویر بزرگتر و انتخاب تصویر کوچکتر

کاربر یک تصویر بزرگتر را انتخاب می‌کند که برای همسان‌سازی اندازه آن تغییر می‌کند. سپس بررسی می‌شود که آیا تصویر RGB است یا نه. کانال‌های رنگی (قرمز، سبز، آبی) استخراج می‌شوند. و هدف از این کار نیر آماده‌سازی تصویر برای پردازش بیشتر، با تمرکز بر استخراج نوار آبی روی پلاک است.

کاربر یک تصویر کوچکتر (مانند یک الگوی از پیش تعیین‌شده) انتخاب می‌کند. اگر تصویر RGB باشد، به خاکستری و سپس به باینری (سیاه و سفید) تبدیل می‌شود .تصویر کوچکتر به عنوان یک الگوی مرجع استفاده می‌شود، مانند بخشی از پلاک که قرار است با تصویر بزرگتر مطابقت داده شود.

```
% Step 1: Select the larger image
[file1, path1] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'}, 'Choose the larger image');
s1 = [path1, file1];
largerImage = imread(s1);
largerImage = imresize(largerImage, [700 600]);

% Ensure that the larger image is indeed RGB by checking its dimensions
if size(largerImage, 3) == 3  % Check if the image has 3 channels (RGB)
    % Extract red, green, and blue channels
    redChannel = largerImage(:, :, 1);
    greenChannel = largerImage(:, :, 2);
    blueChannel = largerImage(:, :, 3);

    % Define color masks based on your criteria for each channel
    red_mask = redChannel < 50;
    green_mask = greenChannel < 140 & greenChannel > 30;
    blue_mask = blueChannel > 110;

    % Combine the masks to create a final color-based mask
    color_mask = red_mask & green_mask & blue_mask;
else
    error('The selected larger image is not an RGB image.');
end
```

تصویر 25 _ انتخاب عکس جلو خودرو

```
% Select the smaller image
[file2, path2] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'}, 'Choose the smaller image');
s2 = [path2, file2];
smallerImage = imread(s2);
smallerImage = imresize(smallerImage, [70 25]);  % Resize for consistency

% Convert the smaller image to binary if it is RGB
if size(smallerImage, 3) == 3  % If the smaller image is RGB, convert to grayscale and then to binary
    smallerImage = rgb2gray(smallerImage);
    threshold = graythresh(smallerImage);
    smallerImage = imbinarize(smallerImage, threshold);
end
```

تصویر 26 _ انتخاب عکس نوار ابی پلاک

بخش 2) تعریف فیلتر مکانی و رنگ ای

یک ماسک مکانی تعریف می‌شود تا فضای جستجو به نواحی خاصی از تصویر بزرگتر محدود شود، با تمرکز بر نواحی وسط و پایین تصویر که احتمال وجود پلاک بیشتر است. این گام فضای جستجو را کاهش می‌دهد و فرآیند همبستگی را سریع‌تر می‌کند با تمرکز بر نواحی محتمل.

```
% Step 3: Define spatial filtering to search both middle and bottom regions
spatial_mask = false(size(largerImage, 1), size(largerImage, 2));  % Initialize mask with all false
quarter_row = floor(size(largerImage, 1) / 4);

% Middle region
spatial_mask(quarter_row:3*quarter_row, :) = true;
% Bottom half
spatial_mask(floor(size(largerImage, 1) / 2):end, :) = true;
```

تصویر 27 _ تعریف فیلتر مکانی

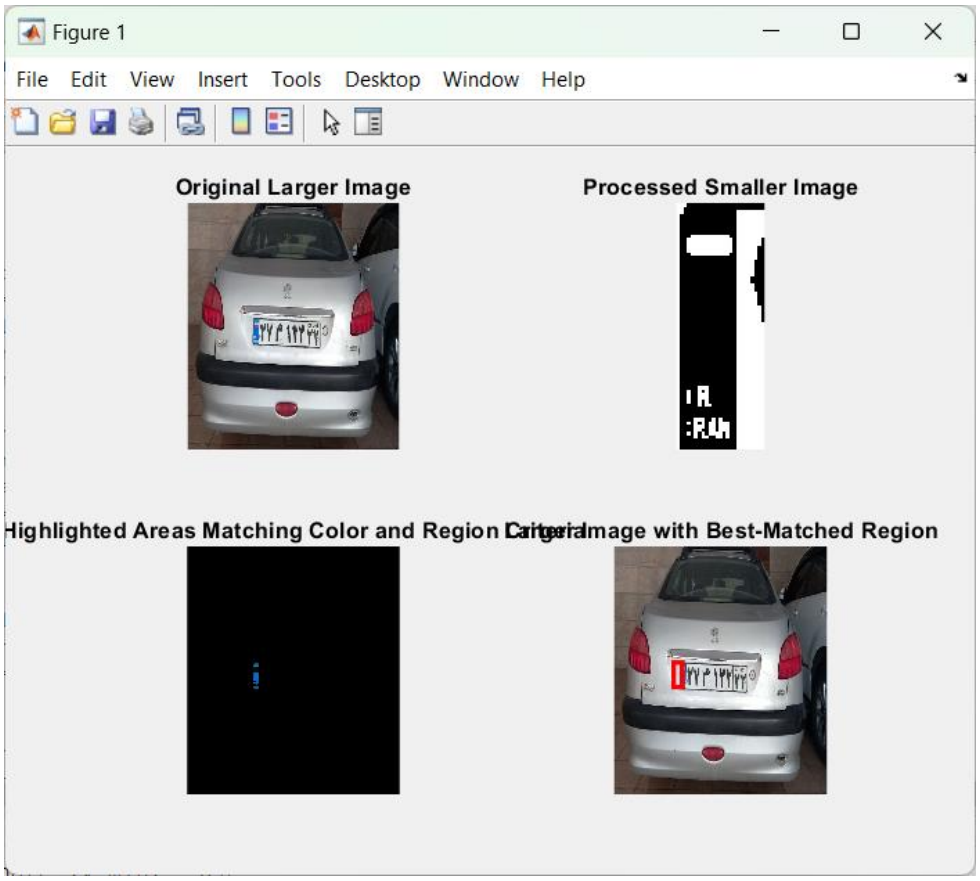
ماسک رنگ (که نوار آبی را تشخیص می‌دهد) با ماسک مکانی ترکیب می‌شود. این ماسک نهایی نواحی‌ای را در تصویر بزرگتر برجسته می‌کند که معیار رنگ و مکان را مطابقت دارند .هدف از این فیلتر ها، جداسازی و برجسته کردن بخشی از تصویر که شامل نوار آبی روی پلاک است.

```
% Combine the spatial mask with the color mask
final_mask = color_mask & spatial_mask;

% Step 4: Apply the final mask to highlight the regions in the larger image
highlightedPicture = largerImage;
highlightedPicture(repmat(~final_mask, [1 1 3])) = 0;  % Black out non-matching areas
```

تصویر 28 _ ادغام فیلتر مکانی و رنگ ای

بخش 3) انجام همبستگی عرضی(Cross-Correlation)

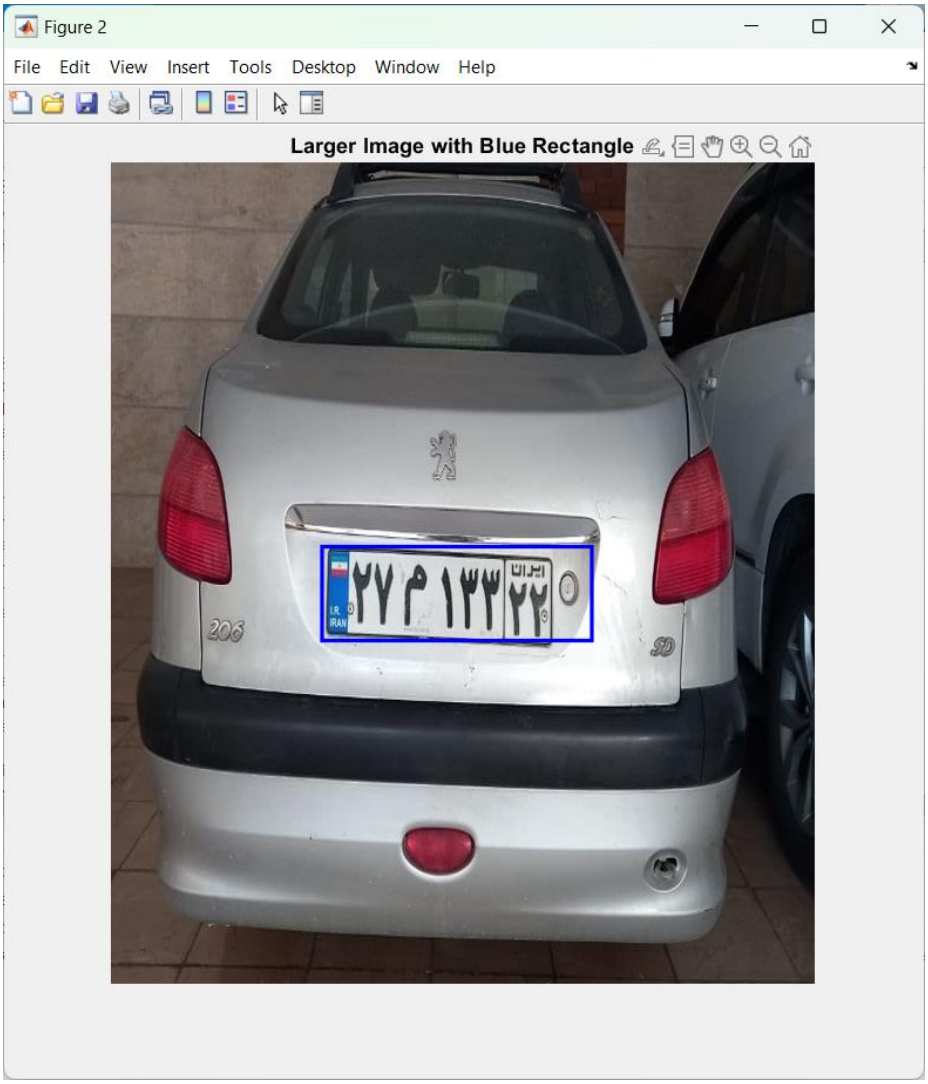
همبستگی عرضی بین تصویر کوچکتر (الگو) و تصویر ماسک‌دار بزرگتر انجام می‌شود. مقدار اوج همبستگی پیدا می‌شود که نشان‌دهنده بهترین تطابق الگو با تصویر بزرگتر است .یافتن ناحیه‌ای در تصویر بزرگتر که بهترین تطابق را با الگو دارد و احتمالاً شامل پلاک است. ناحیه‌ای از تصویر بزرگتر که بالاترین همبستگی با تصویر کوچکتر دارد با یک مستطیل قرمز برجسته می‌شود. شناسایی بصری ناحیه‌ای در تصویر بزرگتر که احتمال وجود پلاک در آن بیشتر است.



تصویر 29 _ پیدا کردن نوار ابی پلاک

بخش 4) برش ناحیه تعریف‌شده توسط مستطیل آبی

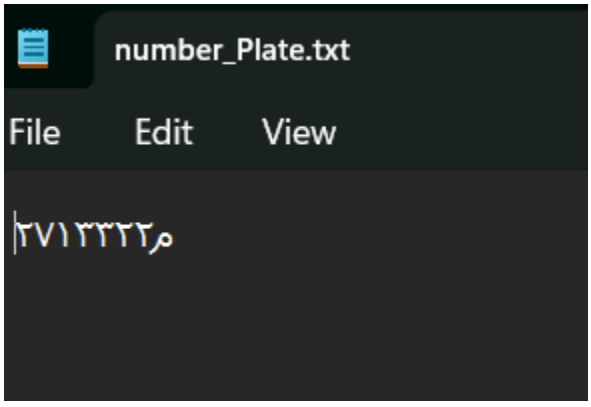
یک تابع (`drawBlueRectangle`) برای ترسیم یک مستطیل آبی در اطراف محل تخمینی پلاک استفاده می‌شود که براساس مختصات محاسبه‌شده از مستطیل قرمز تنظیم می‌شود .این تابع به دقت ناحیه پلاک را تعریف و برجسته می‌کند تا برای برش آماده شود. با استفاده از مختصات مستطیل آبی، برنامه ناحیه‌ای از تصویر بزرگتر که شامل پلاک است را برش داده و ذخیره می کند.



تصویر 30 _ پیدا کردن پلاک تخمینی

تصویر 31 _ جدا کردن پلاک به عنوان یک عکس

حال تصویر پلاک به دست امده را به کد سوال 2 داده تا ارقام و حرف پلاک را تشخیص دهد و طبق تصاویر زیر خروجی صحیح بوده است.



تصویر 32 _ تشخیص کاراکتر های پلاک

سوال 4

کد متلب برای لود کردن یک ویدیو و انتخاب 2 فریم از آن:

```
videoFile = 'car1.mp4';
videoObj = VideoReader(videoFile);

% Initialize frame counter
frameCounter = 1;

% Specify the frames you want
frame1 = 30;
frame2 = 50;

% Initialize variables for storing the frames
chosenFrame1 = [];
chosenFrame2 = [];

while hasFrame(videoObj)
    frame = readFrame(videoObj);
    if frameCounter == frame1
        chosenFrame1 = frame;
    elseif frameCounter == frame2
        chosenFrame2 = frame;
        break;
    end
    frameCounter = frameCounter + 1;
end
```

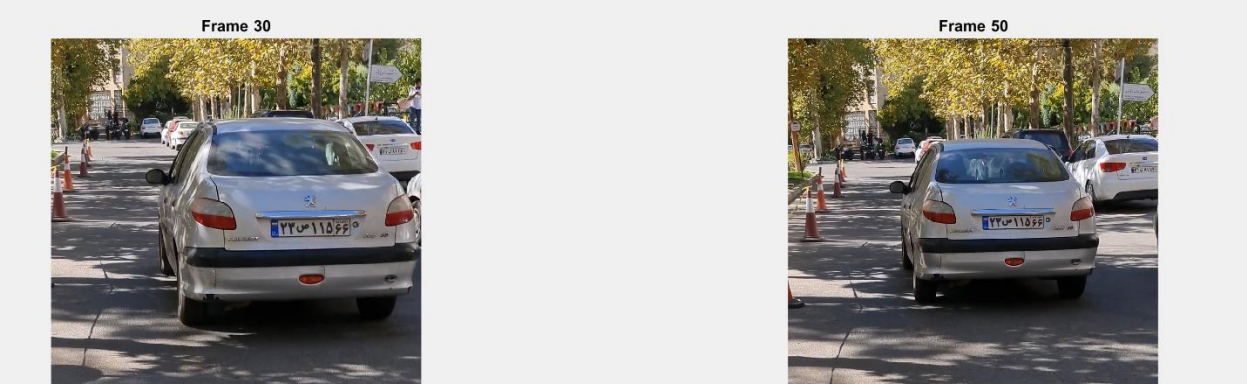
```
% Display the chosen frames
figure;
subplot(1, 2, 1);
imshow(chosenFrame1);
title(['Frame ', num2str(frame1)]);

subplot(1, 2, 2);
imshow(chosenFrame2);
title(['Frame ', num2str(frame2)]);
```

تصویر 33 – کد لود کردن ویدیو و جدا کردن 2 فریم از ویدئو

توضیحات کد :

در ابتدا نام ویدئو را انیشیال کرده و سپس آن را میخوانیم، سپس 2 فریم مورد نظر را انتخاب میکنیم. سپس ویدئو را فریم به فریم می خوانیم، هر گاه به 2 فریم انتخاب شده رسیدیم آن ها را داخل 2 آرایه میریزم.

سپس 2 فریم انتخاب شده را نمایش میدهیم.



تصویر 34 – خروجی حاصل از اجرای کد

تصویر 35 – پیدا کردن پلاک در فریم اول

با استفاده از کد بخش های قبل پلاک را جدا میکنیم.

```
The x-start of the palte: 318
The y-start of the palte: 384
```

تصویر 36 – به دست آوردن x y

تصویر 37 – پیدا کردن پلاک در فریم 2

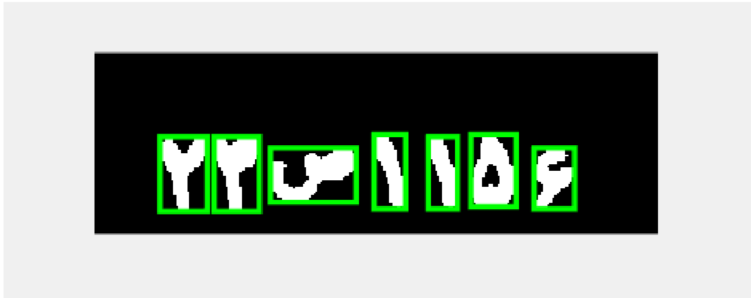
با استفاده از کد بخش های قبل پلاک را جدا میکنیم.

```
The x-start of the palte: 282
The y-start of the palte: 353
```

تصویر 38 – به درست آوردن x , y

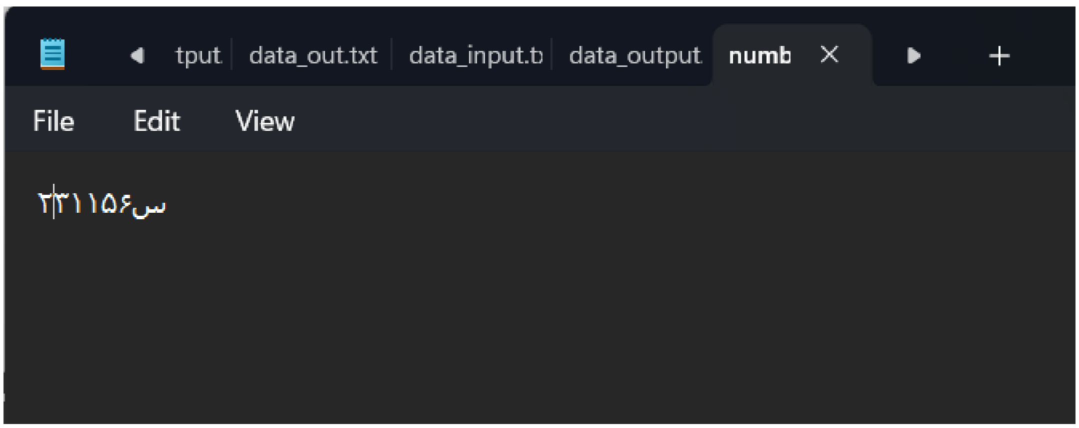
تصویر 39 – به دست آوردن پلاک ها

جدا کردن پلاک ها برای تشخیص کاراکتر ها



تصویر 40 – در آوردن کاراکتر های پلاک

با استفاده از کد قسمت های قبل کاراکتر ها را پیدا میکنیم.



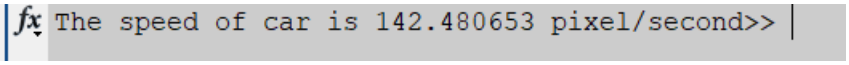
تصویر 41 – به دست آوردن شماره پلاک

شماره پلاک را به دست آورده و آن را داخل فایل تکست ذخیره میکنیم.

```
% Calculating speed
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_frame_1 = 318;
y_frame_1 = 384;
x_frame_2 = 282;
y_frame_2 = 353;

frameRate = videoObj.FrameRate;
deltaTime = (frame2 - frame1) / frameRate;
deltaDistance = sqrt((x_frame_1 - x_frame_2)^2 + (y_frame_1 -y_frame_2)^2);
pixelSpeed = deltaDistance / deltaTime;
fprintf('The speed of car is %f pixel/second', pixelSpeed);
```

تصویر 42 – به دست آوردن سرعت

توضیحات کد بالا: فاصله میان 2 تا پیکسل را محاسبه کرده و آن را تقسیم بر فاصله زمانی میکنیم، بدین ترتیب سرعت متوسط محاسبه میشود(سرعت متوسط = زمان / جا به جایی)



تصویر 43 – خروجی حاصل از اجرا کردن کد