**به نام خدا**

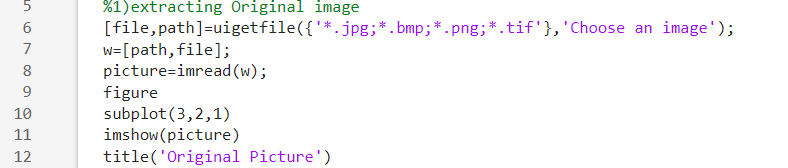
**سید محمد حسین مظهری 810101520 پروژه 2 درس سیگنال و سیستم**

**پارسا دقیق 810101419**

**بخش اول:**

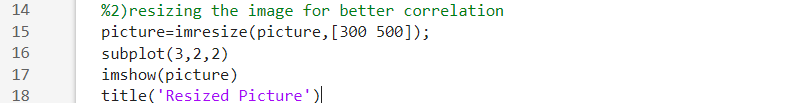
1. با استفاده از تابع uigetfile از کاربر تصویر مورد نظرش را درخواست می کنیم. سپس با استفاده از تابع imread عکس انتخاب شده را در قالب یک ماتریس لود می کنیم.

اسکریپت استفاده شده:



1. در این قسمت از تابع imresize استفاده کرده تا ابعاد تصویر را به x500300 تغییر دهیم.

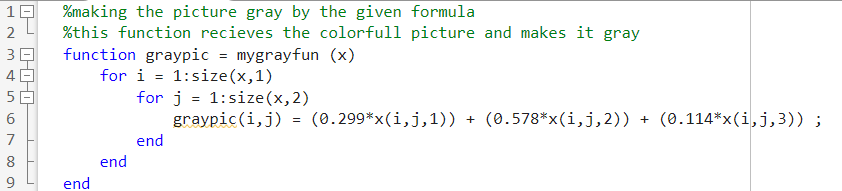
اسکریپت مورد استفاده:



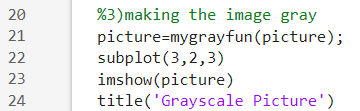
1. برای استخراج پلاک و اعداد و حروف ، رنگ تصویر اهمیتی ندارد . پس برای پردازش راحت تر ، با استفاده از فرمول داده شده رنگ را از تصویر حذف کرده و آن را خاکستری می کنیم.

تابع این بخش را خودمان پیاده سازی میکنیم.

اسکریپت تابع mygrayfun :

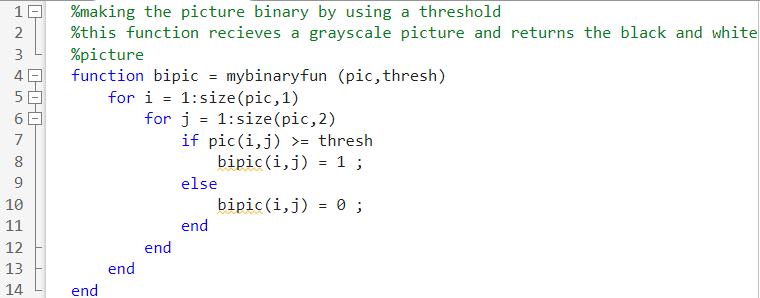


استفاده از تابع در کد اصلی:

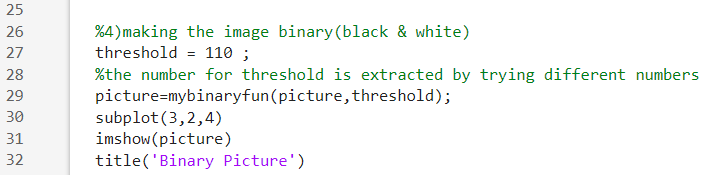


1. با بررسی مقدار هر سلول و مقایسه ی آن با مقدار threshold تصویر را دودویی می کنیم.

اسکریپت تابع mybinaryfun :



استفاده از تابع در کد اصلی:

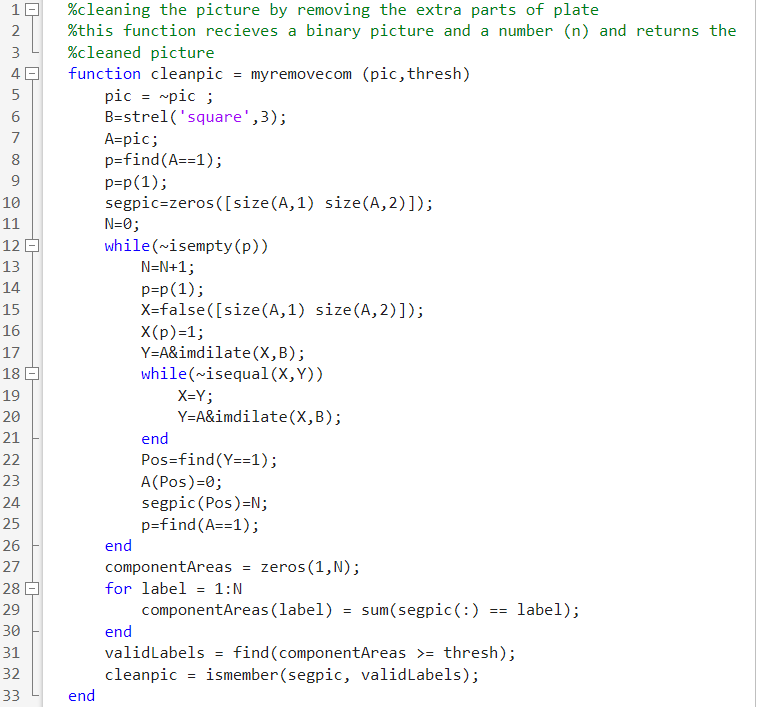


1. با گرفتن عکس باینری و آستانه تعداد پیکسل های متصل به هم ، تابعی را می نویسیم که بخش هایی که تعداد پیکسل های پیوسته آن از مقدار آستانه کمتر است را حذف کند.

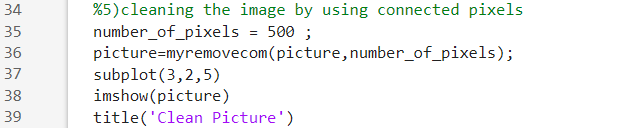
ابتدا تصویر را نقیض می کنیم(تبدیل سیاه به سفید و سفید به سیاه).سپس همانند الگوریتمی که برای قسمت بعد یعنی mysegmentation نوشته ایم تصویر ورودی را بخش بندی می کنیم. اکنون برداری به طول تعداد قسمت ها ایجاد می کنیم و تعداد پیکسل های بر قسمت را پیدا کرده و در بردار مورد نظر ذخیره می کنیم.

سپس با دستور find قسمت هایی که مساحت آن ها از مقدار آستانه بیشتر است را استخراج می کنیم و در برداری دیگر ذخیره می کنیم. با دستور ismember این بردار را با ماتریس segment بندی شده مقایسه می کنیم. نواحی که لیبل آنها در بردار وجود دارند، عدد یک را اتخاذ می کنند و دیگر نواحی عدد صفر. در حقیقت خروجی این تابع یک تصویر دودویی است که شرط مساحت در نواحی آن اعمال شده است

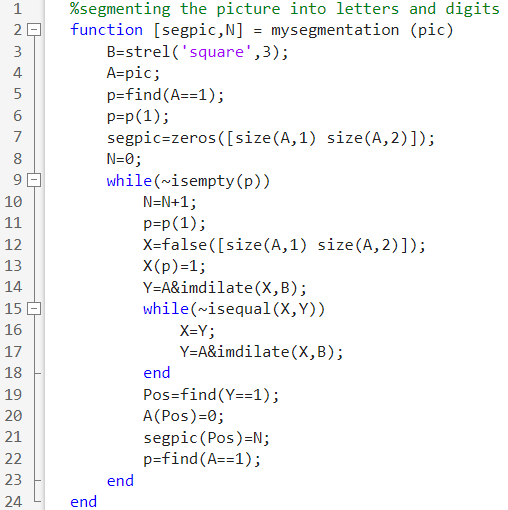
اسکریپت تابع myremovecom :



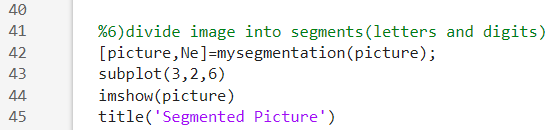
استفاده از تابع در کد اصلی:



1. اسکریپت تابع mysegmentation :



استفاده از تابع در کد اصلی:

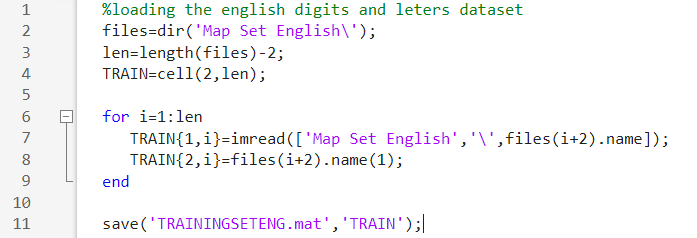


در این قسمت قرار است تابعی بنویسیم که تصویر را segment بندی بکند و تعداد segment ها را نیز برگرداند. در ابتدا یک ماتریس با درایه های صفر به ابعاد تصویر ایجاد می کنیم . سپس با استفاده از دستور find آدرس سلول هایی که مقدار آنها برابر با یک است را در بردار p ذخیره می کنیم و مقدار اولین سلول آن را باز میگردانیم.سپس به تعداد segment ها حلقه می زنیم.(چون تعداد آنها مشخص نیست از while استفاده می کنیم.) سپس یک ماتریس با ابعاد تصویر و مقادیر صفر ایجاد می کنیم. و مقدار سلول p ام آن را یک قرار می دهیم. یعنی اولین خانه ای که تصویر ورودی در آن یک است، در این ماتریس نیز یک است. حال از این سلول شروع کرده و کل ناحیه را می پیماییم.از تابع imdilate استفاده می کنیم.عملکرد این تابع به این صورت است که یک تصویر را بر حسب ناحیه ای که قبلا تعریف کرده ایم گسترش می دهد. ناحیه ی تعریف شده در این بخش مربعی به ضلع 3 پیکسل است.

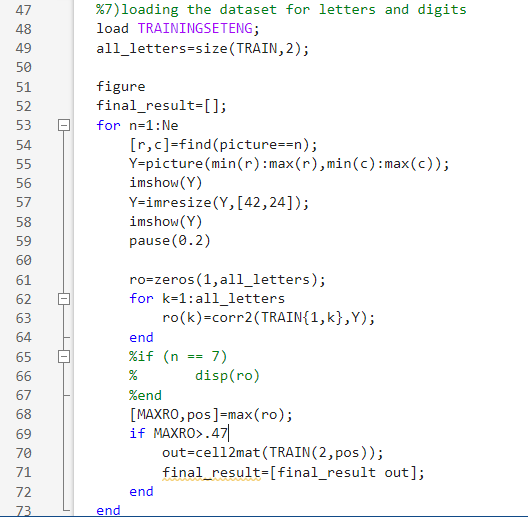
حال حلقه اي ديگر ايجاد مي كنيم. درون اين حلقه از نقطه اي كه قبلا پيدا كرديم ، شروع كرده و ناحيه را همانطور كه توضيح داده شد ، گسترش مي دهيم. پس از هر گسترش ناحيه گسترش يافته را با تصوير اصلي and می کنیم تا نقاطی که در تصویر اصلی صفر هستند در ناحیه ی گسترش یافته نیز صفر شوند و تنها نقاط موجود در تصویر اصلی گسترش پیدا کنند. این حلقه تا زمانی ادامه پیدا میکند که تمامی ناحیه تصویر اصلی را در بر گرفته باشیم.

سپس این نقاط را در ماتریسی که در ابتدا به عنوان تصویر segment بندی شده تعریف کرده بودیم ، با N جایگزین می کنیم. بعد این ناحیه را از تصویر اصلی حذف می کنیم و حلقه از ابتدا شروع به اجرا شدن می کند. این حلقه تا زمانی اجرا می شود که ناحیه ای در تصویر اصلی باقی نمانده باشد.هنگامی که تصویر اصلی خالی شد و تمامی نواحی segment بندی شدند، از حلقه خارج می شویم و تابع تصویر segment بندی شده و تعداد segment ها (N) را بر میگرداند.

1. آپلود مپ ست :



استفاده از کد اصلی:



برای decision making ، ابتدا باید mapset را در متلب لود کنیم. از دستور dir استفاده می کنیم که با گرفتن اسم فایل شامل mapset ، اطلاعات فایل های درون آن را در struct ذخیره می کند. حال ماتریسی با ابعاد تعداد فایل های mapset در 2 ایجاد می کنیم. در یک ردیف تصویر ها و در ردیف دیگر کاراکتر مربوط به هر تصویر را قرار می دهیم.

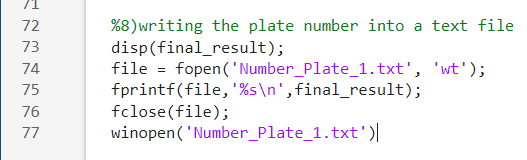
هم اکنون کاراکتر های پلاک را با تصویر های ماتریس به دست آمده از mapset مقایسه می کنیم تا کاراکتر ها را تشخیص دهیم.

تصویر از قبل لیبل گذاری شده است. یک حلقه به تعداد لیبل ها می زنیم و با استفاده از دستور find آدرس نقاطی از تصویر که مقداری برابر با لیبل مربوط به حلقه را دارند به دست می آوریم. حال این نقاط را جدا کرده و در ماتریس دیگری دخیره می کنیم. ( کوچکترین و بزرگترین ردیف و ستون را با استفاده از دستور های min , max به دست آورده و سلول های بین این چهار مقدار را در ماتریسی دیگر ذخیره می کنیم. )

حال اندازه این تصویر را با دستور imresize تغییر می دهیم تا هم اندازه تصویر موجود در mapset شود . سپس به اندازه ی تعداد کاراکتر های موجود در mapset مقدار correlation ها را محاسبه کرده و در یک بردار ذخیره می کنیم. پس از پایان کار حلقه ، بیشترین مقدار correlation و موقعیت آن (یعنی کاراکتر مورد نظر ) را بر می گردانیم .( توجه شود که مقدار آستانه ای برای correlation در نظر گرفته شده است (0.47) ).

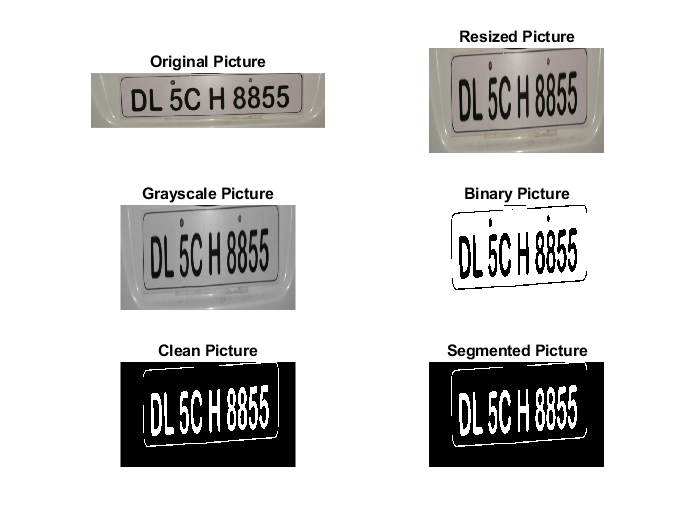
اگر matching به درستی رخ داد ، با استفاده از دستور celltmat و موقعیت بزرگترین عدد بردار، کاراکتر مربوطه را از TRAIN استخراج کرده و به یک ماتریس تبدیل می کنیم. و این ماتریس را به ماتریس نهایی اضافه می کنیم. با تمام شدن حلقه ، ماتریس نهایی نیز کامل می شود .

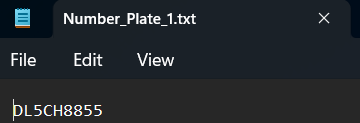
1. اسکریپت مربوط به چاپ پلاک:



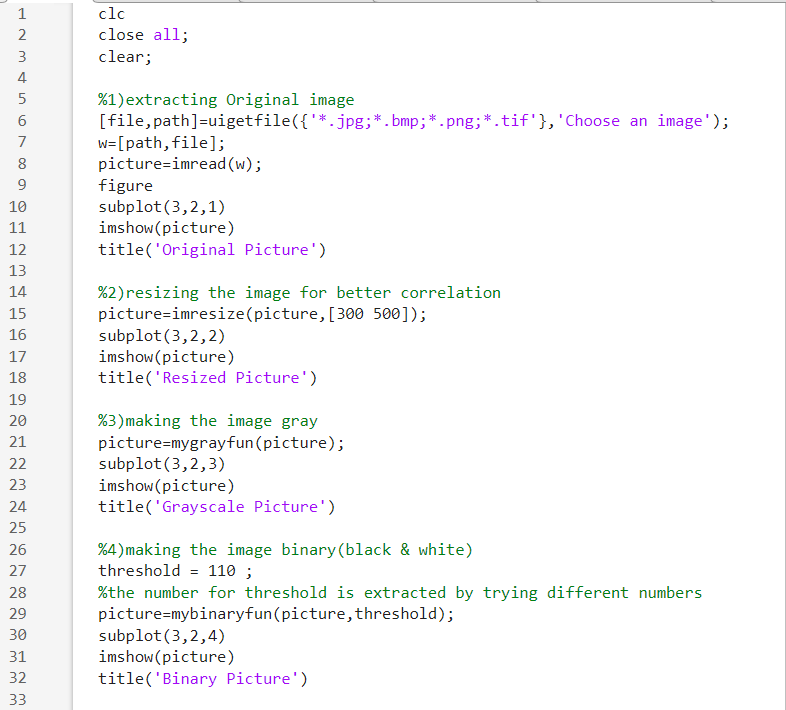
برای چاپ پلاک، از دستور disp استفاده می کنیم. برای ذخیره کردن پلاک در فایل ، ابتدا با دستور fopen یک فایل .txt ایجاد میکنیم. سپس با دستور fprintf کاراکتر های ماتریس نهایی را به شکل استرینگ در فایل می نویسیم. در نهایت هم با دستور fclose به کار خاتمه می دهیم. بدین ترتیب هم پلاک در یک فایل .txt ذخیره می شود .

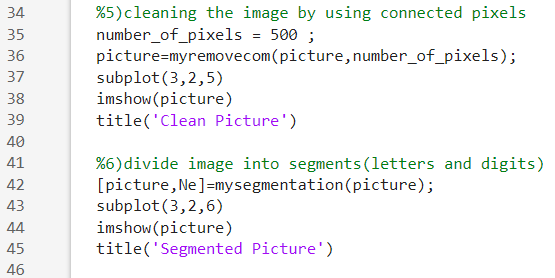
نتیجه ی کار برای یک پلاک ورودی :

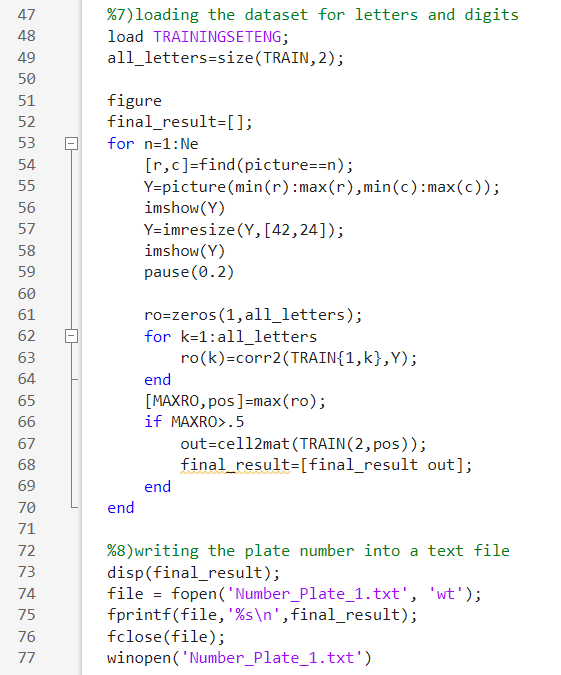




اسکریپت p1.m :



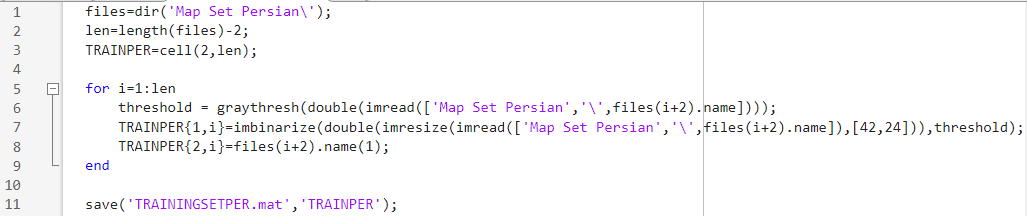




**بخش دوم:**

برای تهیه ی mapset از پلاک ها عکس گرفته و با سیاه سفید کردن عکس و جدا کردن کاراکتر ها به mapset فارسی می رسیم. سپس با استفاده از تابع imbinarize تصاویر را دودویی می کنیم. جهت انجام عملیات correlation گیری نیز اندازه ی تصاویر را به 42x24 تغییر می دهیم .

اسکریپت این بخش :



Main :

گرفتن فایل از کاربر مانند بخش قبل است .

تغییر دادن اندازه تصاویر مانند بخش قبل است .

برای خاکستری کردن تصاویر از تابع rgb2gray استفاده شده است .

برای باینری کردن عکس از graythresh برای پیدا کردن آستانه استفاده می کنیم و سپس با دستور imbinarize تصویر را باینری می کنیم.

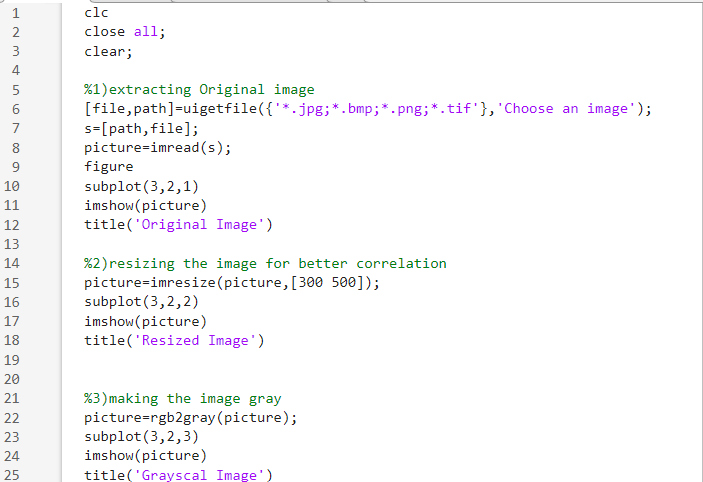
در قسمت کاهش نویز از تابع bwareaopen استفاده می کنیم و آستانه تعداد پیکسل ها را 500 در نظر می گیریم .

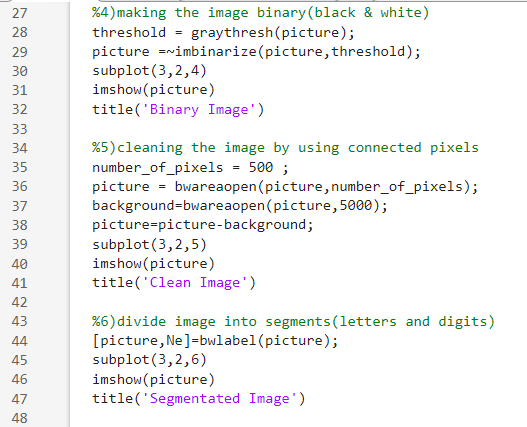
در قسمت segmentation از تابع bwlabel استفاده می کنیم .

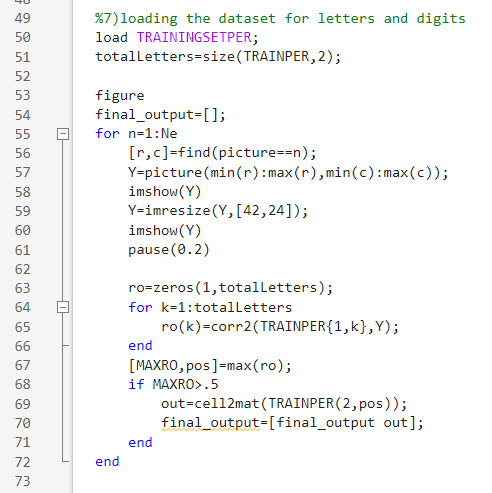
Decision making مانند بخش قبل است .

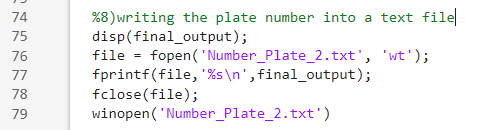
چاپ و ذخیره پلاک مانند بخش قبل است .

اسکریپت p2.m :

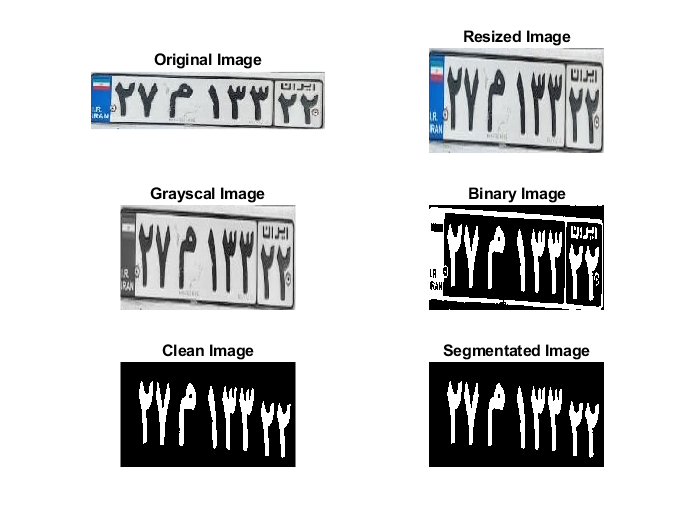


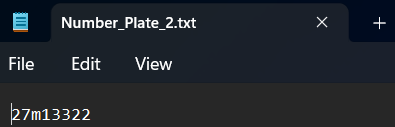
****

****

****

نتیجه ی کار برای پلاک دلخواه:



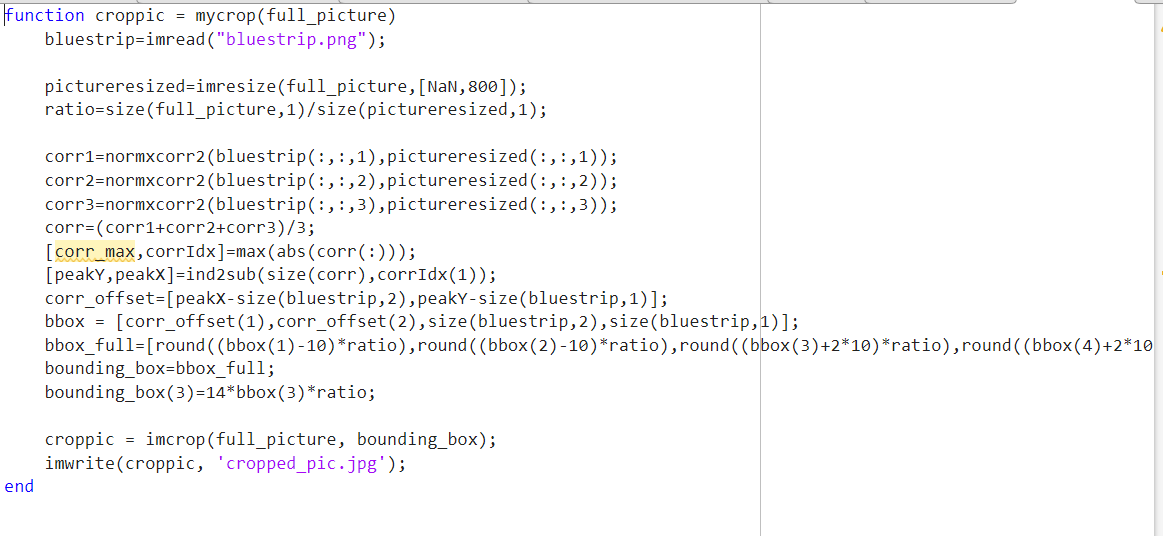


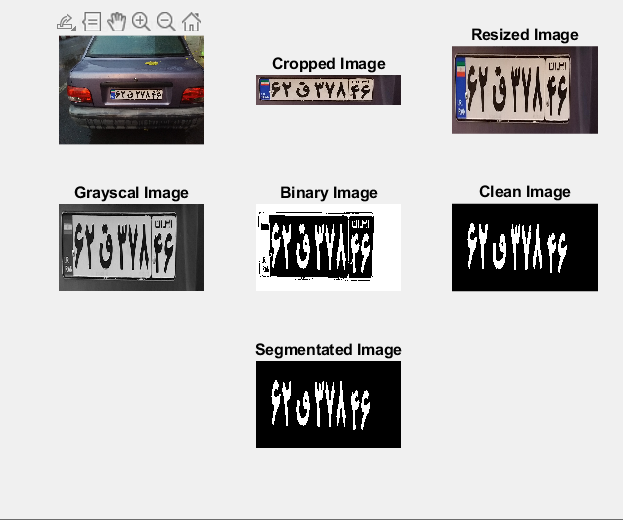
**بخش سوم:**

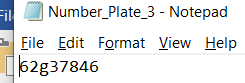
### روش bluestrip

در این روش یک template matching روی نوار آبی سمت چپ که همه پلاک‌های فارسی دارند انجام می‌دهیم.

این کار را با استفاده از cross correlation نوار و عکس انجام می‌دهیم. با استفاده از تابع mycrop می توانیم که پلاک را جدا کنیم.

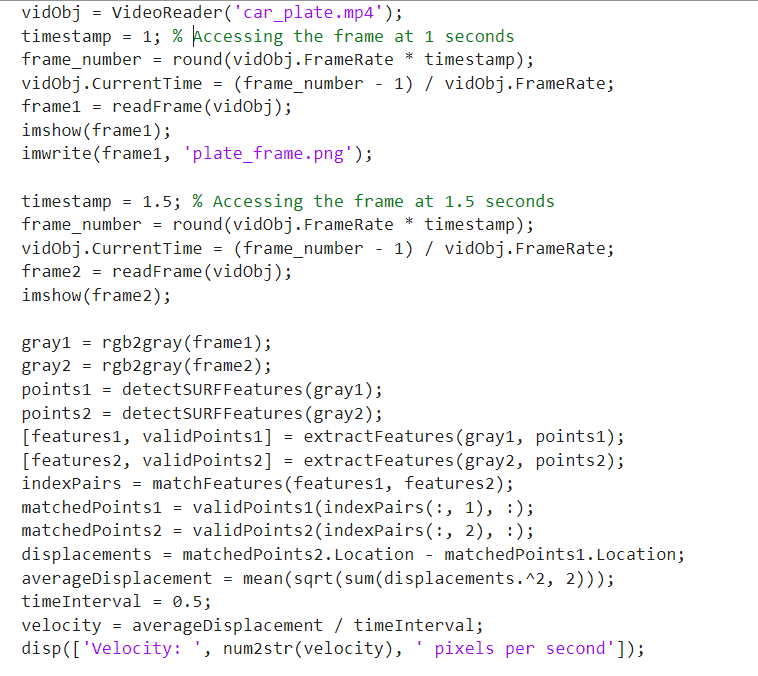






**بخش چهارم:**

در دو لحظه t=1sو t=1.5s دو فریم را بررسی می‌کنیم. فریم‌ها را پردازش می‌کند تا ویژگی‌ها شناسایی شوند، ویژگی‌ها را بین دو فریم مطابقت می‌دهد، جابجایی نقاط مطابقت داده شده را محاسبه می‌کند و سپس سرعت را بر اساس فاصله زمانی بین فریم‌ها محاسبه می‌کند. نتیجه در واحد پیکسل بر ثانیه نمایش داده می‌شود.

که با استفاده از کد قسمت سوم هم پلاک را از فریم اول t=1s تشخیص می دهیم.

با استفاده از محاسبات بالا میبینیم که نتایج زیر بدست می آید:

