به نام خدا

سیگنال ها و سیستم

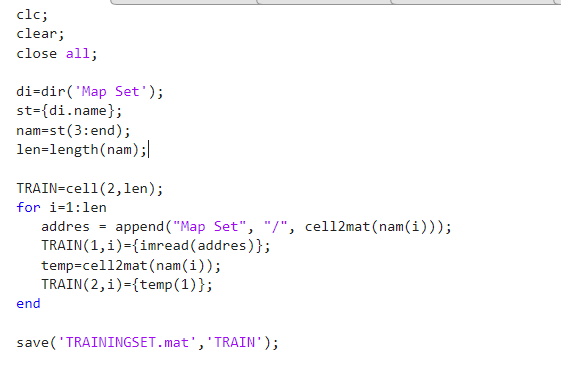
پروژه کامپیوتری 2

مهراد لیویان 810101501

مرضیه موسوی 810101526

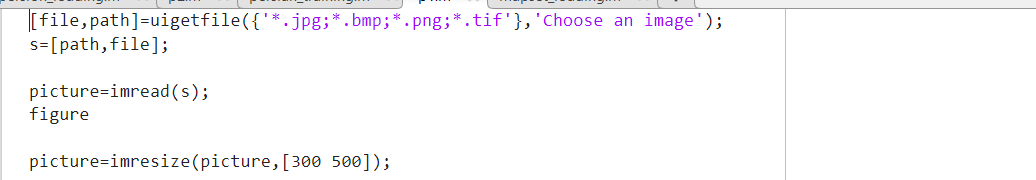
بخش اول

ابتدا mapset انگلیسی را لود می کنیم:

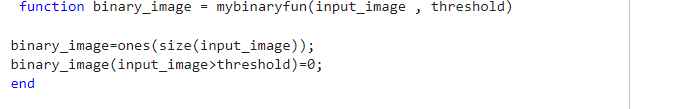


در یک بخش از train عکس مپ ست و در بخش دیگر نام عضو مپ ست را ذخیره می کنیم.

حال در p1.1 عکس را از کاربر گرفته و آن را لود می کنیم و سپس resize می کنیم.

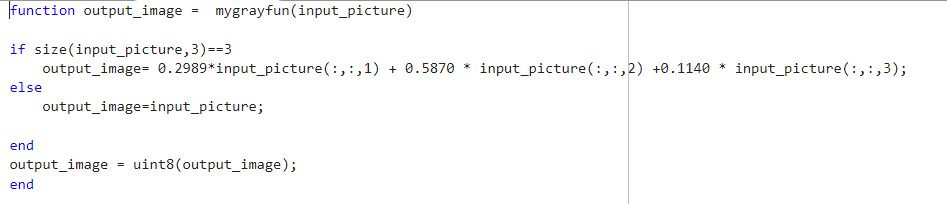


مشابه تابع imbinarize تابع mybinaryfun را می نویسیم:

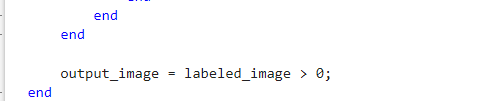
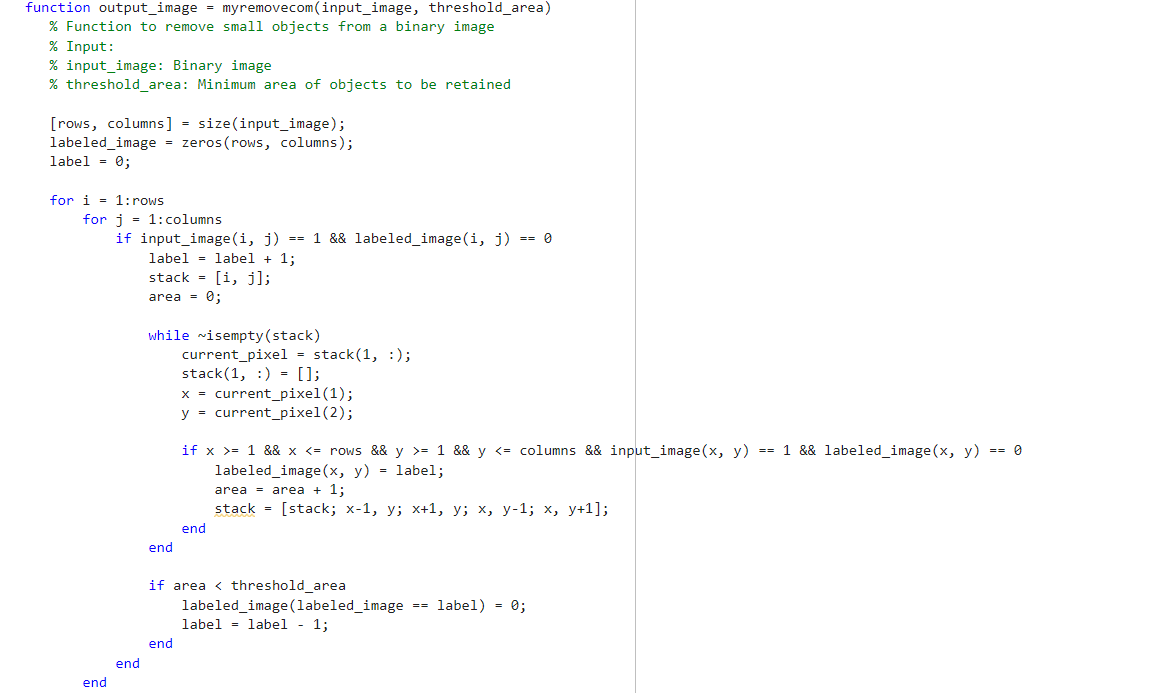


اگر عدد از threshold بالاتر بود یک و در غیر این صورت 0 بگذار.

تابع mygrayfun را برای برعکس کردن رنگ سیاه و سفید عکس می گذاریم:

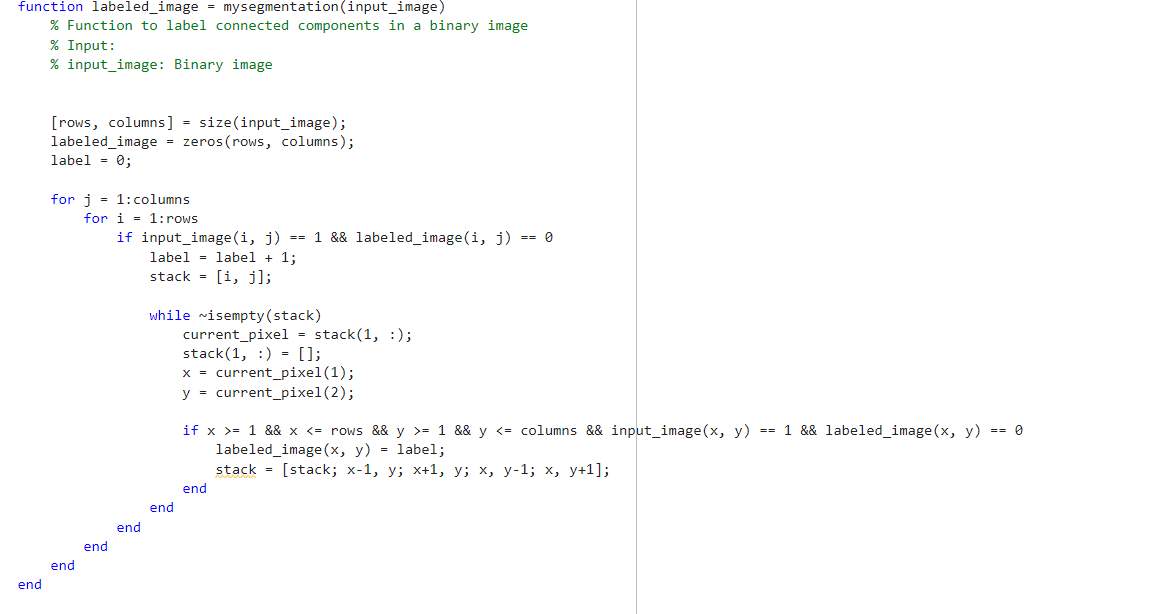


برای تابع myremovecom مانند الگوریتم bfs عمل می کنیم:



ابتدا طول و عرض نقاطی که سفید هستند را جدا می کنیم و نقاط نزدیک آن ها را پیدا می کنیم و.سپس همسایه های نقاط جدیدی را که پیدا کردیم را پیدا می کنیم تا وقتی که همسایه ی جدیدی یافت نشود.وقتی همسایه جدیدی یافت نشد مساحت بخش را که قبلا حساب کرده ایم اگر از آستانه کمتر بود حذف می کنیم.

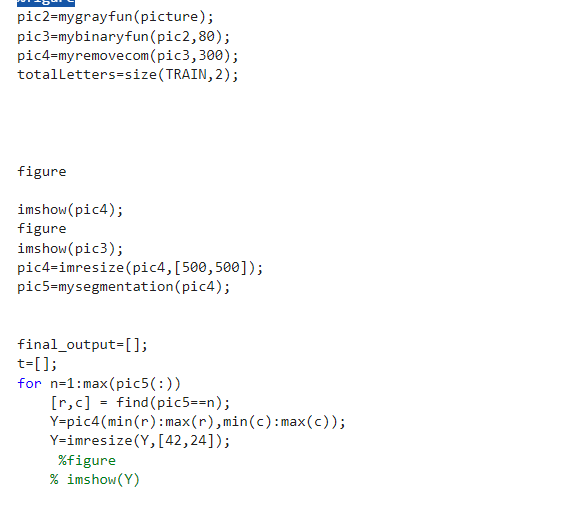
تابع mySegmentation به صورت زیر است:

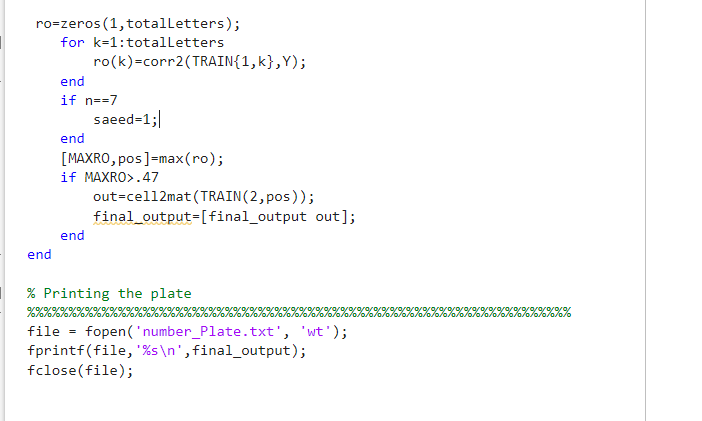


در این تابع نیز مانند قبل با الگوریتمbfs مانند قسمت قبل نقاط همسایه را که 1 هستند می یابیم و هر وقت نقطه ی جدید پیدا نشد لیبل را یکی اضافه می کنیم.

به ترتیب توابع را اجرا می کنیم.دوبار myremovecom را اجرا می کنیم.یکبار برای حذف نقاط کوچک و بار دوم همه ی نقاط را به جز قاب حذف می کنیم و در یک تصویر حاصل را ذخیره می کنیم.قاب را از تصویر اولیه کم می کنیم تا فقط حروف بماند.

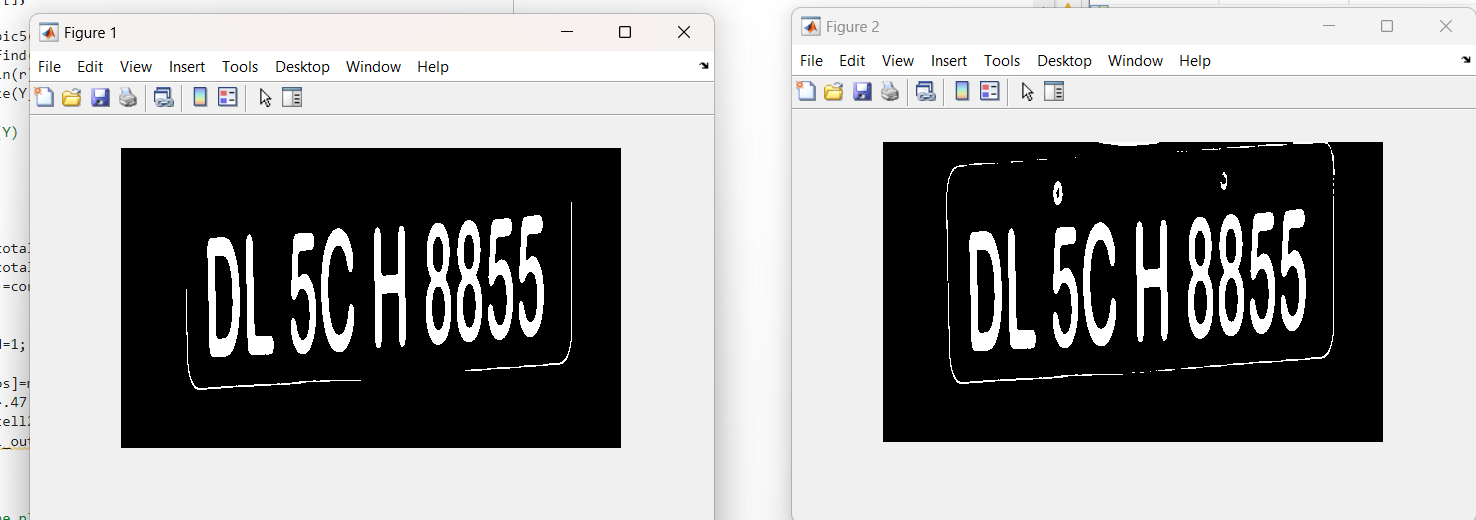
تصاویر را برای چک کردن نشان می دهیم.





در آخر correlation می گیریم و هر حرفی ماکسیمم correlation را داشت به عنوان جواب برداشته و در خروجی می نویسیم.

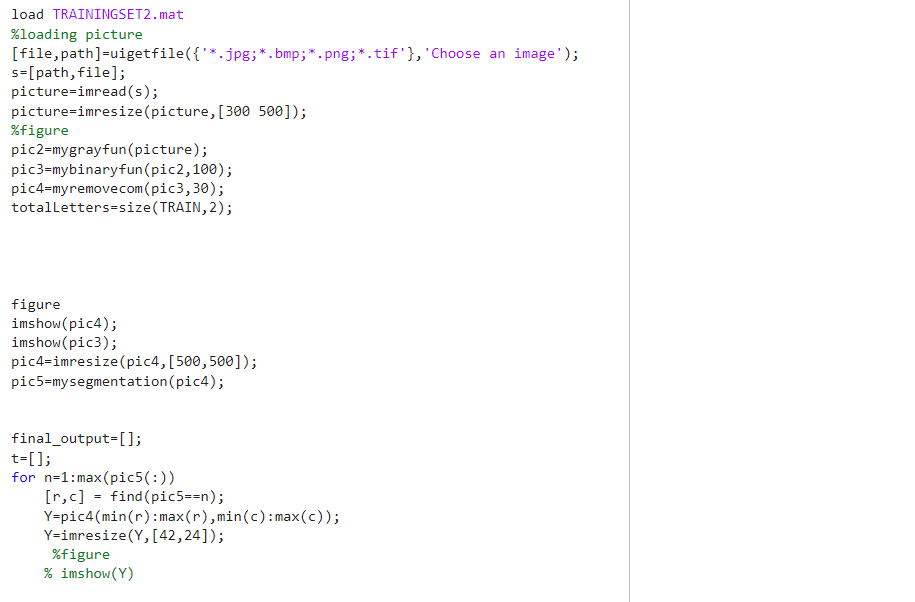
کورلیشن را حتما از عددی بیشتر باید در نظر بگیریم که مثلا اجزای اضافی و نویز را با حرف اشتباه نگیرد.

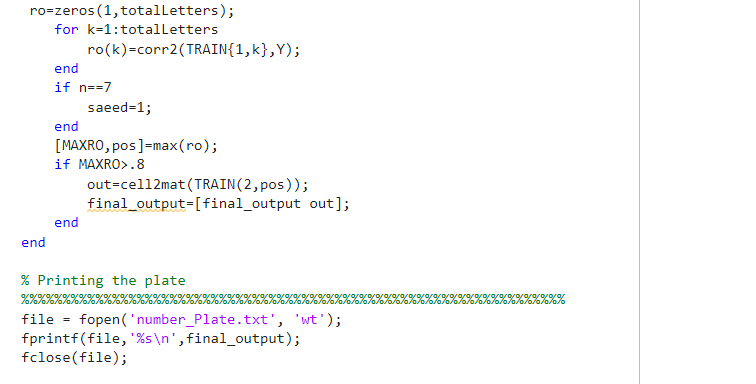




بخش دوم:

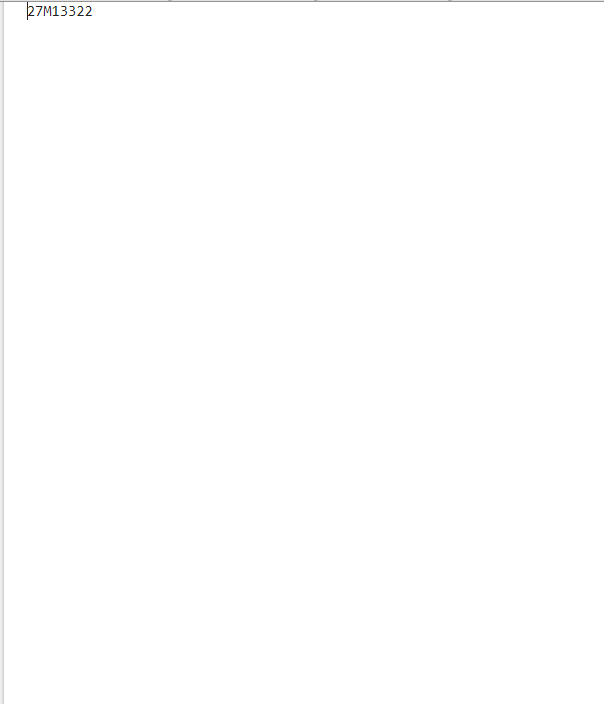
ابتدا mapset فارسی را لود می کنیم.

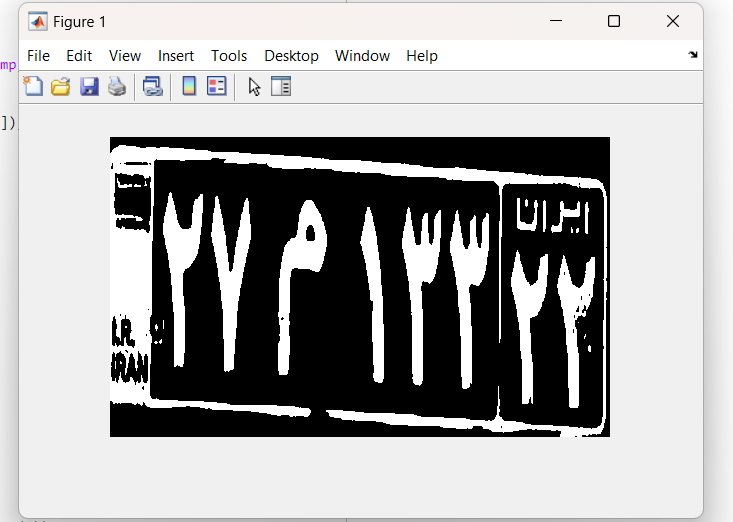


چ

پروسه ی بخش دوم دقیقا مانند بخش اول است.برای trainingset از ماتریکس ست ترم قبل استفاده شده و خودمان کدی برای لود کردن مپ ست نزدیم.

همه ی پروسه ی تشخیص پلاک مانند گذشته است



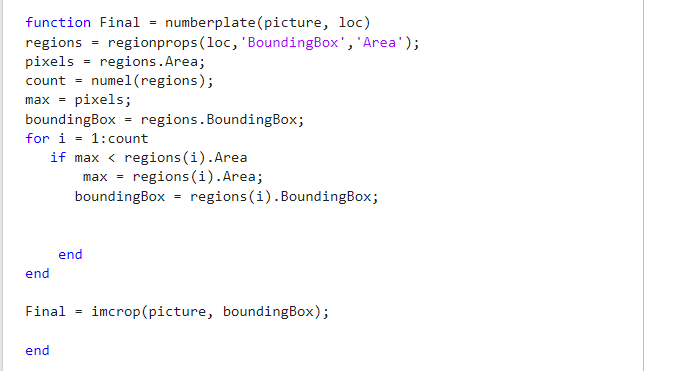


بخش سوم

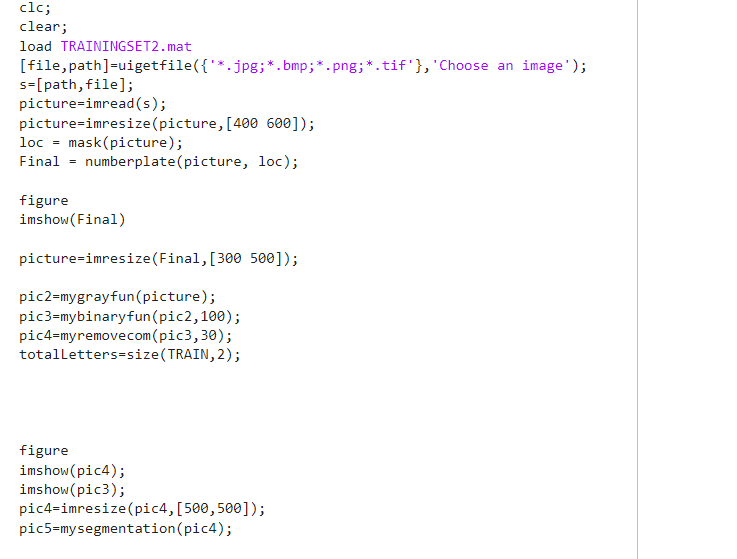
ابتدا تابع mask را معرفی می کنیم:



ابتدا در تابع mask ،edge های تصویر پلاک تشخیص داده می شود و برای بریدن بخش پلاک استفاده شود.



سپس با استفاده از تابع numberplate ، تصویر شماره های پلاک را استخراج می کنیم.





ابتدا لوکیشن پلاک با استفاده از تابع mask استخراج می کنیم و سپس شماره ها را تشخیص می دهیم.سپس همان پروسه را تکرار می کنیم.تبدیل به تصویر خاکستری می کنیم.باینریش می کنیم.نقاط اضافی را حذف می کنیم.سگمنت بندی می کنیم.کورلیشن می گیریم و حروف را چاپ می کنیم.

خروجی تصویر:





بخش چهارم :

در این بخش ابتدا یک تابع به نام calculate\_velocity می نویسیم که به این صورت عمل می کند که نام ویدیو را در ورودی میگیرد سپس ویدیو را می خواند .

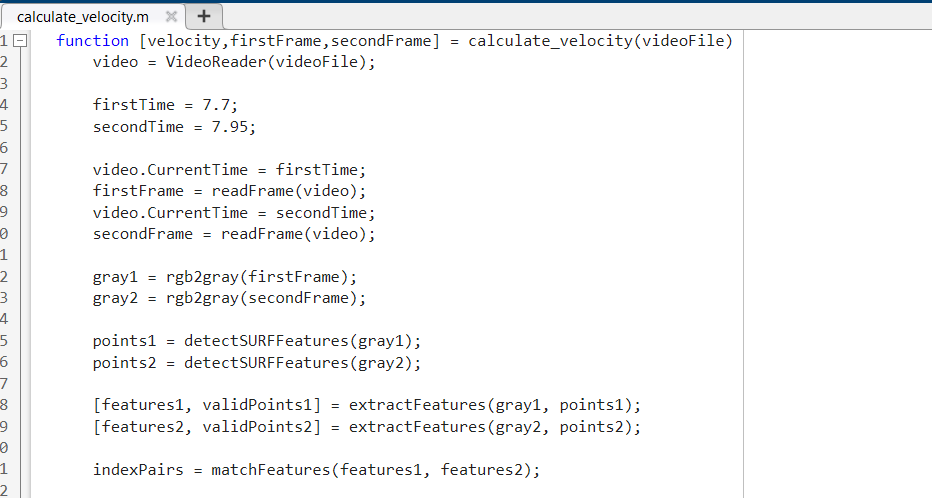
سپس بعد از آن دو فریم از آن را میگیریم و آن را با تابع rgb2gray به صورت خاکستری در می آوریم سپس آن ها را به تابع detectSURFFeatures می دهیم تا این تابع نقاط مهم را برای محاسبه سرعت پیدا کند .

1. این فراخوانی تابع تصاویر خاکستری را تحلیل می‌کند تا نقاط جالب را پیدا کرده و برگرداند - این‌ها نقاط کلیدی هستند که در آن‌ها تصویر دارای بافت‌ها یا گوشه‌های متمایز است.
2. **خروجی:** خروجی، points1 و points2، اشیایی هستند که اطلاعات مربوط به نقاط کلیدی شناسایی شده را شامل می‌شوند. هر نقطه کلیدی دارای مکان، مقیاس و جهت است. این نقاط در مراحل بعدی برای استخراج و تطبیق ویژگی‌ها بین تصاویر استفاده می‌شوند.

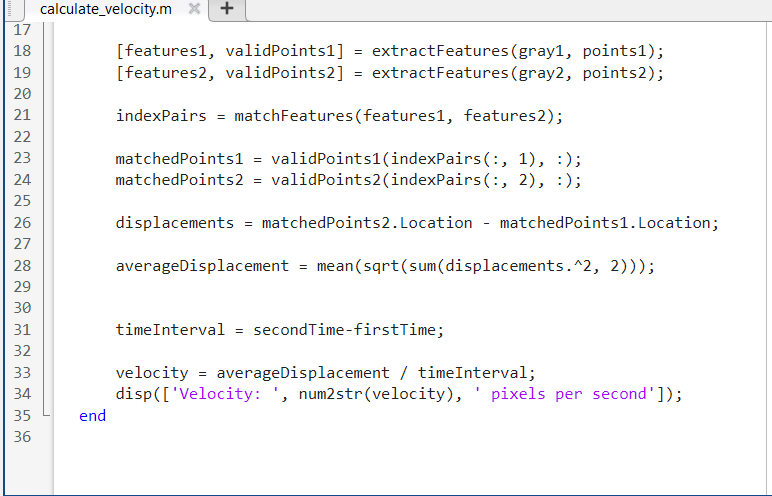
با تشخیص این ویژگی‌های SURF، ما پایه‌ای را برای مقایسه نقاط کلیدی بین دو فریم ایجاد می‌کنید که برای محاسبه جابجایی و در  نتیجه سرعت اجسام در ویدئو ضروری است.

سپس ایندکس های این تصاویر را پیدا می کنیم سپس به روش MSE میانگین مجموع مربعات تفاضل این نقاط مهم در دو فریم را حساب می کنیم و تقسیم بر تفاوت زمان این دو تابع می کنیم مانند فرمول سرعت تا مقدار سرعت بدست بیاید.

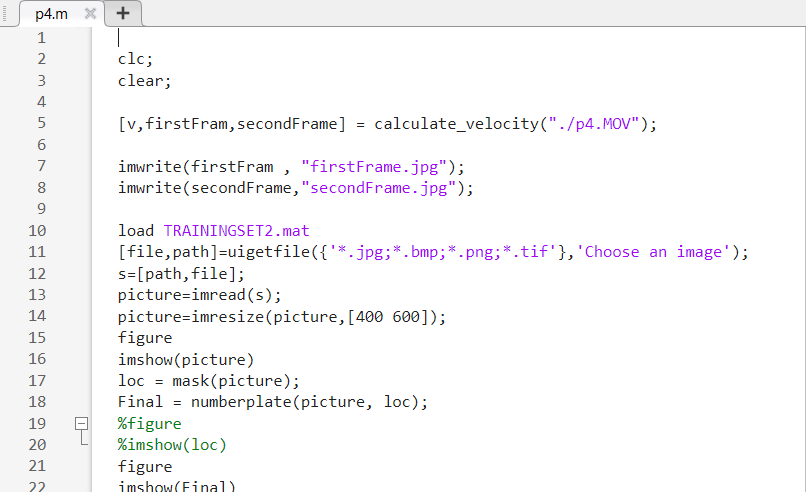
اسکریپتی که نوشته شده :



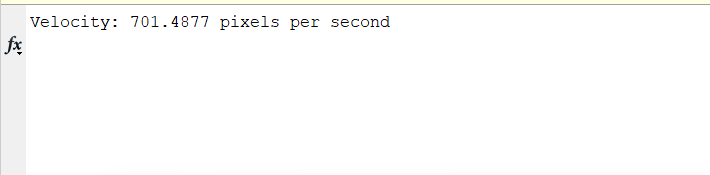
ادامه :



حالا از این تابع در p4.m استفاده می کنیم تا مقدار سرعت را چاپ کند سپس این دو فریم را در دو فایل firstFrame.jpg و secondFrme.jpg می ریزیم تا بتوانیم آن ها را در ادامه کد که مانند بخش p3.m هست استفاده کنیم.



نتیجه آن برای سرعت :



تصویر اول آن :

