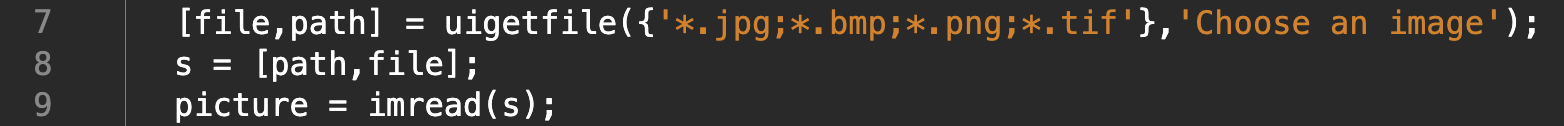
گزارش‌کار پروژه دوم سیگنال ها و سیستم ها

فاطمه کرمی محمدی | ۸۱۰۱۰۰۲۵۶

**بخش اول)**

۱- با استفاده از تابع uigetfile و imread تصویر مورد نظر کاربر در یک ماتریس سه بعدی به نام picture لود می‌شود.



۲- با تابع imersize ابعاد تصویر به ۳۰۰ در ۵۰۰ پیکسل تغییر میابد.

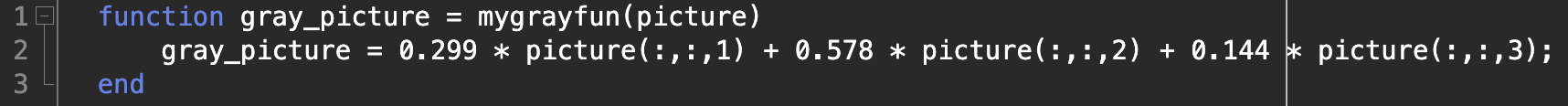


نمایش نمونه ای از picture با تابع imshow به صورت زیر است:



۳- با تابع mygrayfun رنگ تصویر به طوسی تبدیل شده و تصویر طوسی در ماتریسی دو بعدی با نام gray\_picture ذخیره می‌شود. ماتریس دو بعدی است به این دلیل که با انجام محاسباتی روی سه رنگ هر پیکسل، یک عدد نهایی به جای سه عدد RGB در ماتریس ذخیره می‌شود که آن را طوسی می‌کند.

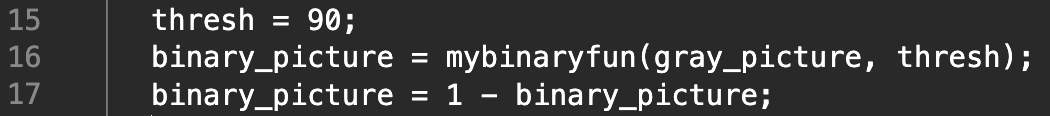




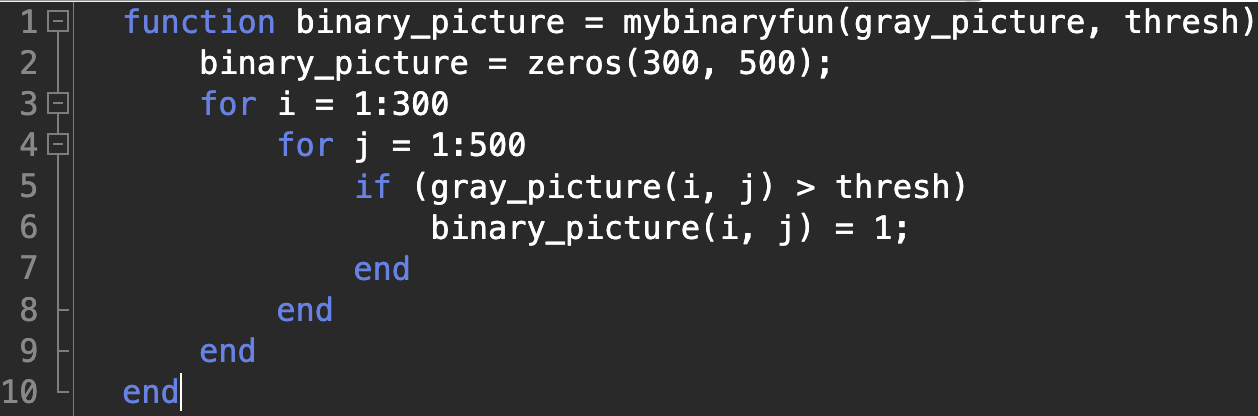
نتیجه gray\_picture به صورت زیر است:



۴- با تعریف یک آستانه دلخواه برای هر پیکسل (برای مثال ۹۰) مقدار پیکسل هایی که مقدار بیشتر از آستانه دارند به ۱ و مقدار پیکسل هایی که مقدار کمتر از آستانه دارند به ۰ تغییر داده شده و در ماتریس دو بعدی جدیدی به نام binary\_picture ذخیره می‌شود. با این کار تصویر تبدیل به تصویری کاملا سیاه و سفید می‌شود. در نهایت با not کردن همه پیکسل های تصویر سیاه و سفید رنگ های سیاه و سفید در آن جابجا شده تا در ادامه اعداد و حروف پلاک سفید باشند.



تابع mybinaryfun به این صورت این کار را انجام می‌دهد:

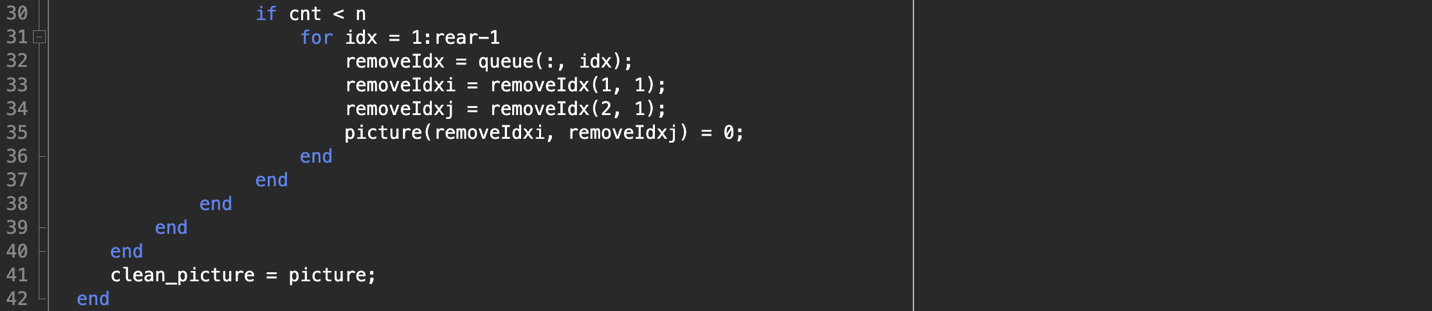
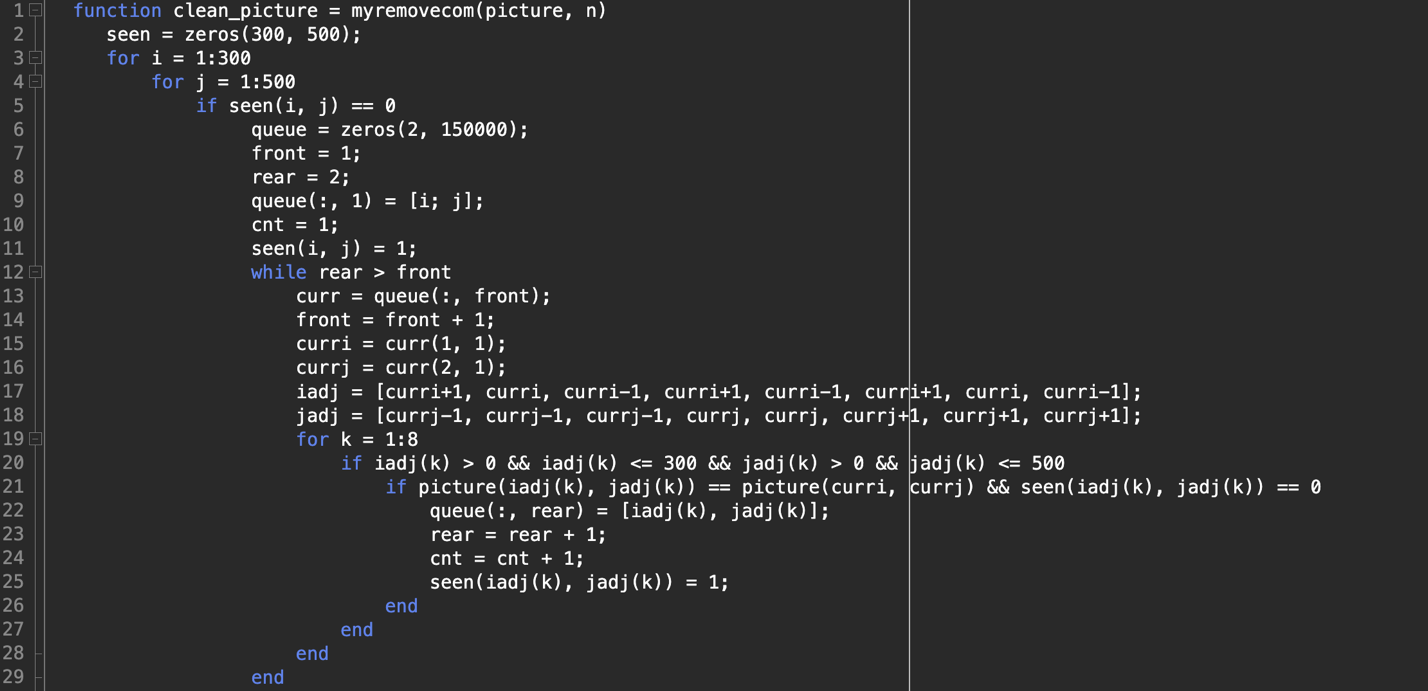


ابتدا binary\_picture به صورت ماتریسی همه صفر در نظر گرفته شده و سپس به ازای هر پیکسل gray\_picture که مقدار آن از آستانه ورودی بیشتر است مقدار binary\_picture پیکسل متناظر برابر ۱ قرار داده می‌شود.

نتیجه به این صورت است:

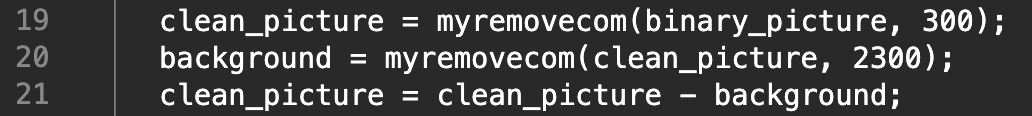


۵- برای حذف نویزها و قطعه های سفید باقی مانده در binary\_picture که اطلاعات مفید نیستند باید قطعه های سفید متصل به هم که از سایز مشخصی کوچکتر هستند از شکل حذف شوند. تابع myremovecom به این شکل این کار را انجام می‌دهد:



دو ورودی picture و عدد n که المان های با سایز کوچکتر از آن باید حذف شوند را می‌گیرد. با اجرای الگوریتم bfs (جستجوی اول سطحی) روی عکس ورودی، المان های کنار هم را پیدا کرده و به ازای هر المان مختصات پیکسل های آن المان را در آرایه queue نگه داشته و تعداد پیکسل های آن المان را می‌شمرد. در نهایت اگر سایز المان کمتر از مقدار ورودی n باشد در عکس باینری ورودی، مقدار پیکسل های موجود در آن المان را برابر ۰ می‌گذارد تا آن المان پاک شود. در نهایت ماتریس دو بعدی clean\_picture را برابر ماتریس picture بدون نویز گذاشته و خروجی می‌دهد.

استفاده از این تابع در main به این گونه است:



ابتدا تابع myremovecom روی عکس سیاه و سفید و با n مساوی ۳۰۰ صدا شده و خروجی آن در ماتریس clean\_picture ریخته می‌شود. سپس یک بار دیگر این تابع روی عکس تمیز شده و با n مساوی ۲۳۰۰ صدا زده می‌شود تا المان های با سایز بیشتر از ۲۳۰۰ در ماتریس background ذخیره شوند. در آخر clean\_picture منهای background می‌شود تا المان های سفید با سایز بزرگتر از ۲۳۰۰ از clean\_picture حذف شوند.

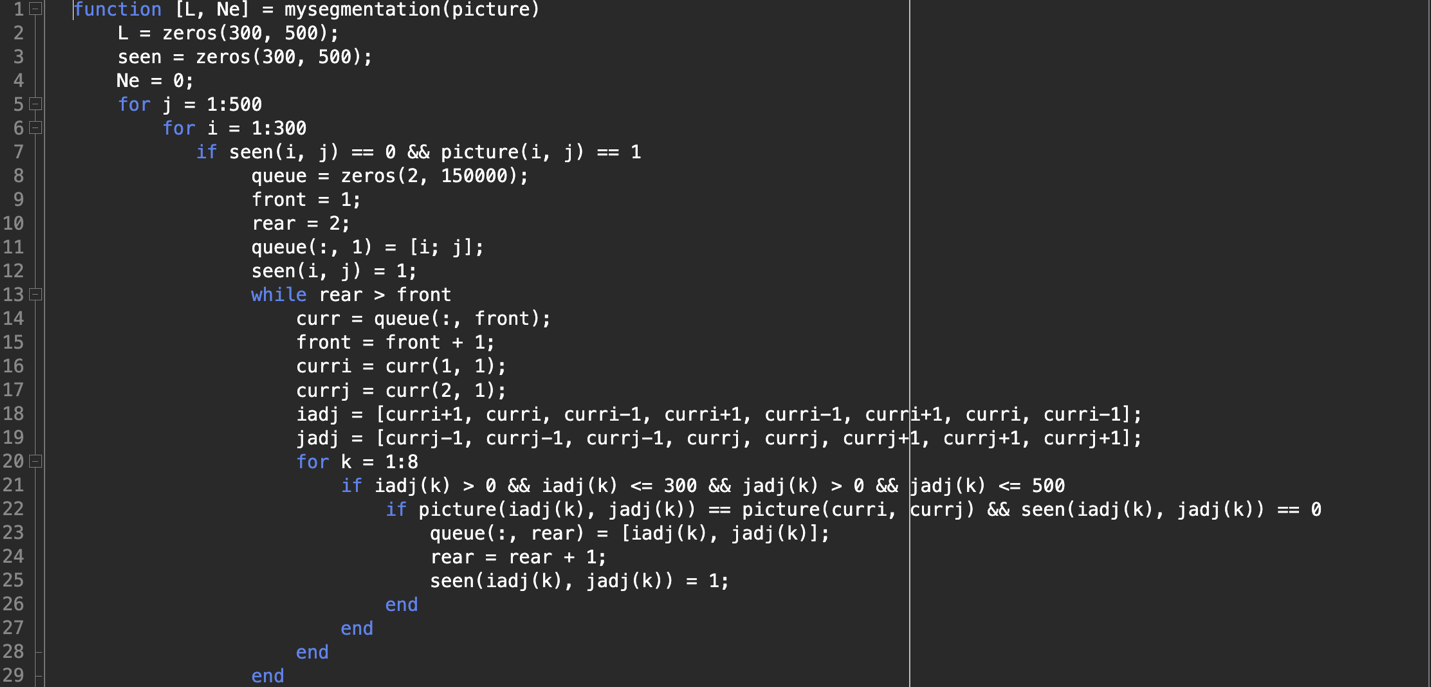
تصویر نتیجه به این شکل است:



در تصویر بالا اکثر نویز ها و اطلاعات غیرمفید حذف شده است.

۶ و ۷- برای جدا کردن حروف و اعداد پلاک از هم و segment بندی از تابع mysegmentation استفاده می‌شود. این تابع مانند تابع myremovecon عمل می‌کند با این تفاوت که بعنوان ورودی فقط clean\_picture را گرفته و ماتریسی همه صفر با ابعاد عکس و نام L نگه می‌دارد. پس از انجام bfs روی تصویر ورودی، مقدار همه پیکسل های متناظر هر المان را در ماتریس L برابر با شماره المان می‌گذارد. در انتها L شامل اعداد ۰ و ۱ تا Ne (تعداد المان ها) خواهد بود که هر المان با یکی از اعداد ۱ تا Ne از دیگر المان ها متمایز شده است. در نهایت تابع، ماتریس L و عدد Ne را برمی‌گرداند.

پیاده سازی تابع به شکل زیر است:



صدا کردن تابع در main به این شکل است:



برای decision making نیاز است که train data که حاوی تصاویر حروف و اعداد انگلیسی است در متلب لود شود:

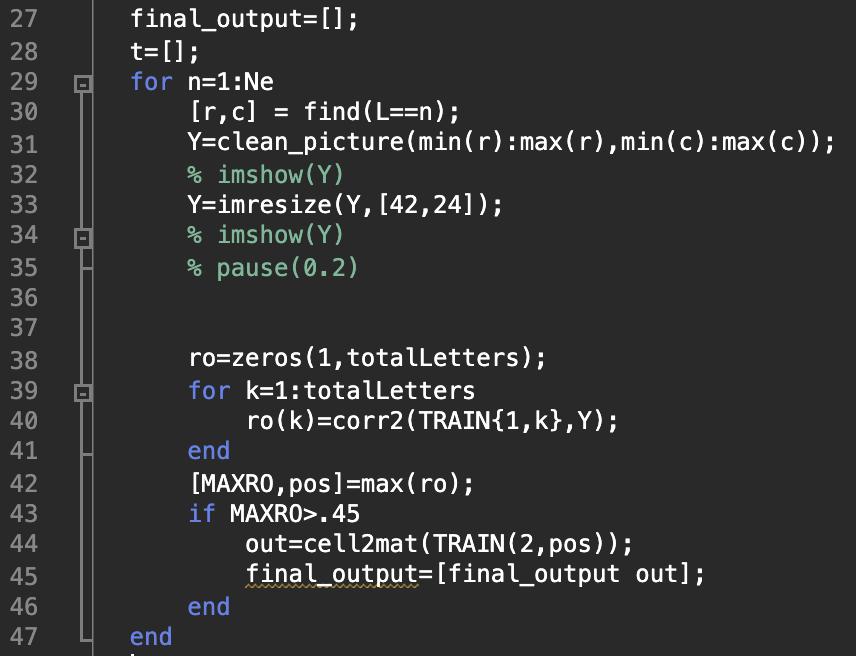


در این فایل ادرس فولدر train data ذخیره و این فولدر باز شده و تصاویر داخل آن به همراه عدد یا حرفی که آن تصویر نشان دهنده آن است در متغیر TRAIN داخل فایل TRAININGSET ذخیره می‌شود.

برای استفاده این دیتاست در main باید لود شود:



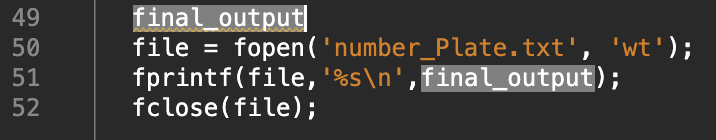
سپس در یک حلقه با استفاده از تابع find المان هایclean\_picture بر حسب عددی که با آن در L نشان داده شده اند از هم جدا شده و تغییر سایز به سایز استاندارد [42, 24] می‌دهند (عکس های دیتاست هم همین ابعاد را دارند). سپس با استفاده از تابع corr2، correlation هر حرف از پلاک با همه اعداد و حروف دیتاست گرفته می‌شود و حرف یا عددی که بیشتری correlation را با حرف مورد نظر در پلاک دارد و مقدار correlation آنها از threshold خاصی بیشتر است بعنوان برنده در نظر گرفته شده و حرف یا عدد متناظر با آن تصویر به final output اضافه می‌شود.



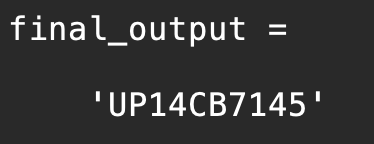
المنت‌های جدا شده عکس به این شکل خواهد بود:



۸- در نهایت نتیجه اعداد و حروف پلاک در یک فایل txt ذخیره شده و چاپ می‌شود.



نتیجه پلاک نهایی برای این مثال به این شکل است:



**بخش دوم)**

پس از تهیه دیتابیس اعداد و حروف فارسی و لود کردن آنها همانند قبل در main این بخش، باقی کار مانند بخش اول است به جز کمی تفاوت در threshold و ورودی n تابع myremovecome که بر حسب آزمایش روی عکس های مختلف با پلاک فارسی بهینه سازی می‌شود.

نمونه ای از تشخیص پلاک فارسی در این بخش به این شکل است:

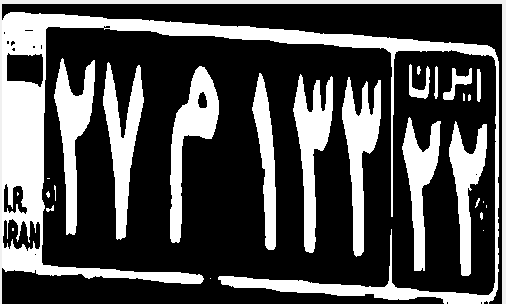
عکس خام:



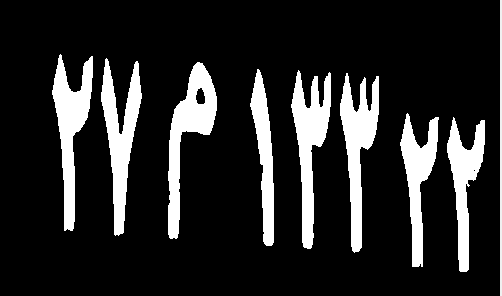
عکس طوسی:



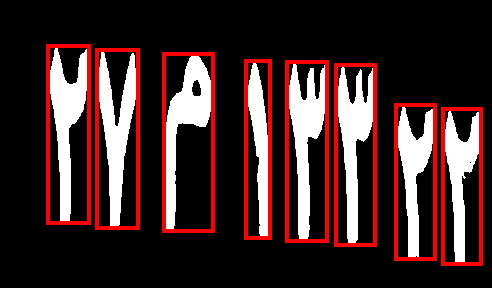
عکس سیاه و سفید:



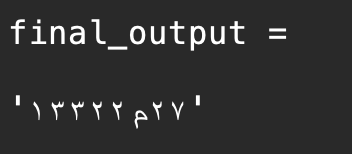
عکس سیاه و سفید پس از حذف نویز و المان های غیرمفید:



بخش بندی شده عکس سیاه و سفید:

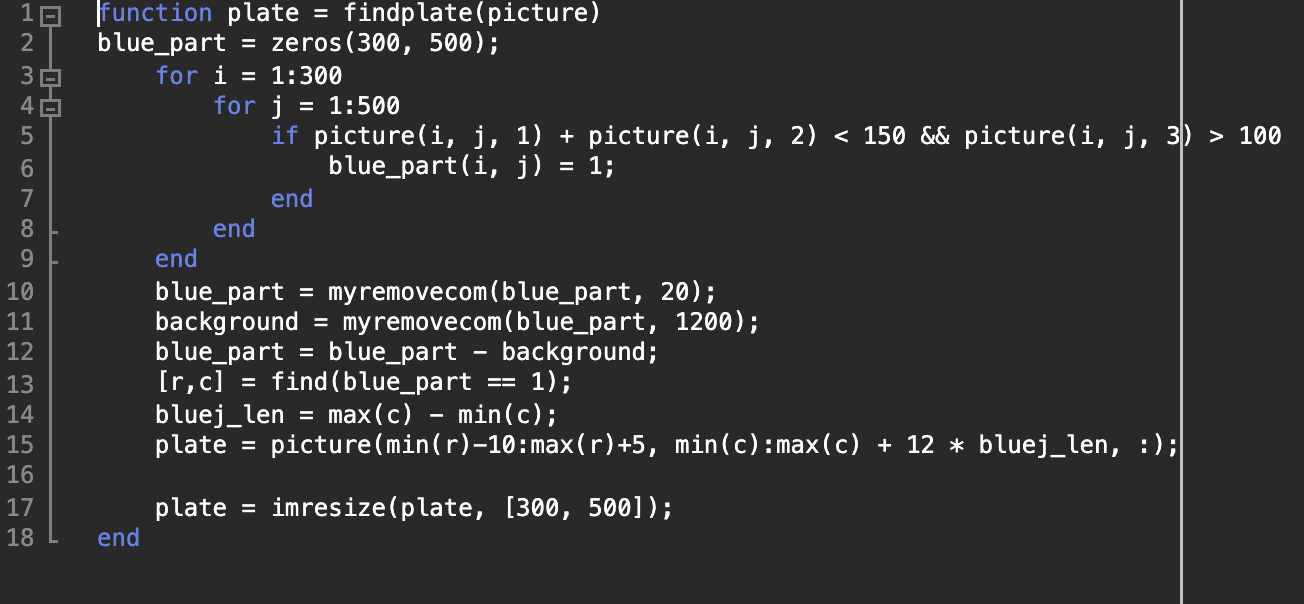


نتیجه نهایی برای این پلاک:



**بخش سوم)**

ایده این قسمت این است که مستطیل آبی سمت چپ پلاک پیدا شده و سپس متناسب با ابعاد بخش آبی پیدا شده، بقیه پلاک از تصویر جدا شود و باقی کار مانند بخش دوم انجام شود. برای پیدا کردن بخش آبی و جدا کردن پلاک طبق تابع زیر عمل می‌شود:



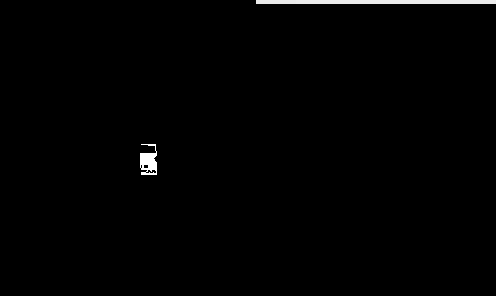
در این تابع پیکسل هایی که در تصویر اصلی رنگ نزدیک به آبی دارند با فرمول نوشته شده پیدا شده و در ماتریس blue\_part مقدار پیکسل های متناظر با آنها ۱ می‌شود. سپس با کمک تابع myremovecom قسمت های بسیار کوچک و بسیار بزرگ که آبی هستند در ماتریس blue\_part حذف شده تا فقط بخش متناسب با مستطیل آبی سمت چپ پلاک باقی بماند. در نهایت مختصات اضلاع این مستطیل با کمک تابع find پیدا شده و با کمک ابعاد و مختصات آن بخش متناظر با پلاک کامل ماشین مشخص شده و از عکس اصلی جدا خواهد شد. این قسمت در ماتریس plate ذخیره شده، به ابعاد ۳۰۰ در ۵۰۰ resize می‌شود و خروجی داده می‌شود.

مثالی از این قسمت به شکل زیر است:

تصویر جلوی ماشین:



تصویر ماتریس blue\_part:



تصویر پلاک پیدا شده نهایی:



باقی کار مانند قبل است.