# بخش اول – پلاک انگلیسی

کد این بخش از تعدادی سلول تشکیل شده که به صورت جداگانه در مورد هر کدام از آن‌ها توضیح می‌دهیم.

## سلول Initialization

این سلول شامل بستن تمام نمودارها، پاک کردن تمامی متغیرها و تعریف تعدادی متغیر constant می‌شود.

## سلول Load the Dataset

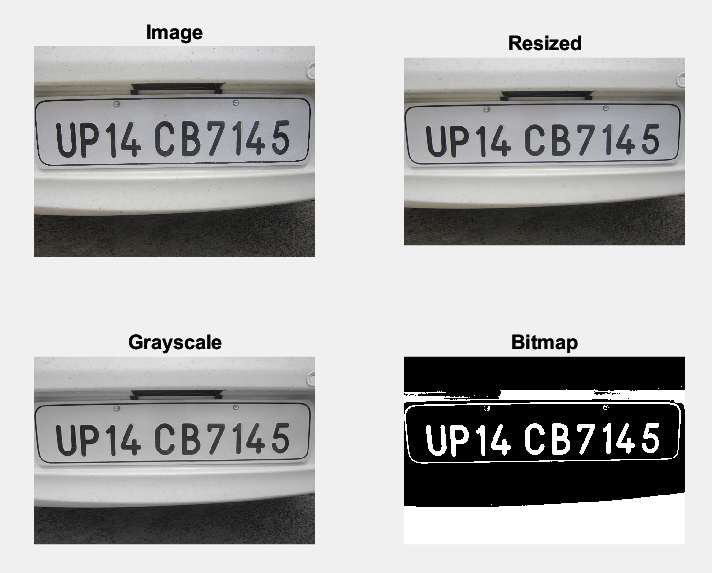
این سلول ابتدا بررسی می‌کند که فایل LICENSE\_LETTERS.mat که حاوی دیتاست برنامه است، وجود دارد یا خیر. اگر وجود داشت، فایل را لود کرده و در متغیر letters ذخیره می‌کند. اما اگر این فایل وجود نداشت، تابع make\_letterset را صدا می‌زند و این تابع تصاویری که در فولدر Map Set قرار دارند را می‌خواند و متغیر letters را برمی‌گرداند.

## سلول Input Image

در این سلول فقط تصویر پلاک مورد نظر انتخاب شده و ماتریس این تصویر در متلب لود می‌شود.

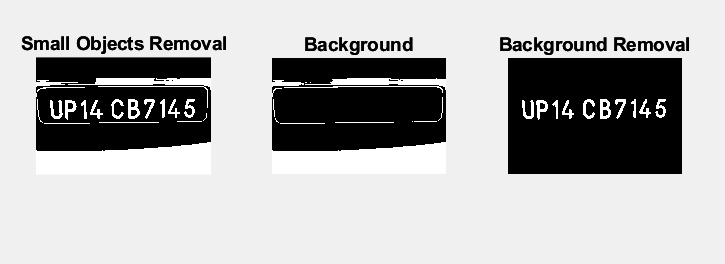
## سلول Preprocessing

در این سلول ابتدا تصویر را به اندازه‌ای که در متغیر IMAGE\_SIZE مشخص شده resize می‌کنیم. سپس، تصویر را به grayscale تبدیل کرده و در نهایت آن را به صورت سیاه و سفید (binary) تبدیل می‌کنیم. لازم به ذکر است که رنگ‌های تصویر سیاه و سفید معکوس می‌شوند که رنگ حروف پلاک، سفید باشد. نمونه‌ای از این مرحله در تصویر زیر قابل مشاهده است.



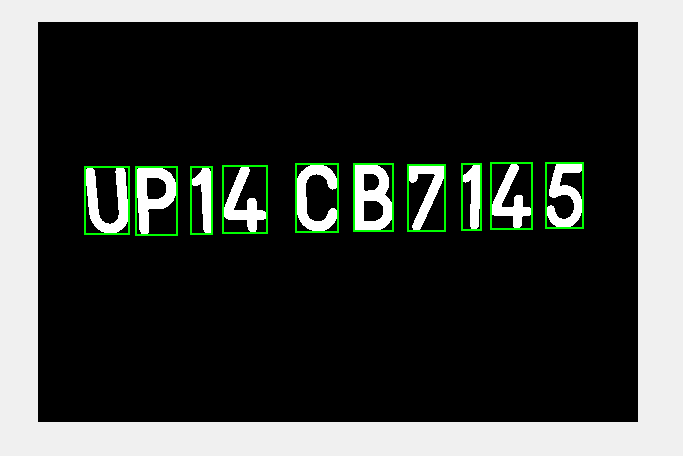
## سلول Remove Background and Small Objects

در این سلول ابتدا بخش‌های کوچک تصویر (بخش‌هایی تعداد پیکسل تشکیل دهنده آن‌ها کمتر از مقداری است که در متغیر SMALL\_OBJECT\_AREA مشخص شده) از تصویر حذف می‌شوند. در نهایت پس‌زمینه تصویر (بخش‌هایی که تعداد پیکسل تشکیل دهنده آن‌ها کمتر از مقداری است که در متغیر BACKGROUND\_AREA مشخص شده) از تصویر حذف می‌شود. نمونه‌ای از این مرحله در تصویر زیر نشان داده شده است.



## سلول Segmentation

در این سلول بخش‌های باقی مانده از تصویر جداسازی می‌شوند که بتوانیم آن‌ها را در سلول بعدی با استفاده از دیتاست تشخیص دهیم. نمونه‌ای از این مرحله در تصویر زیر مشخص است.

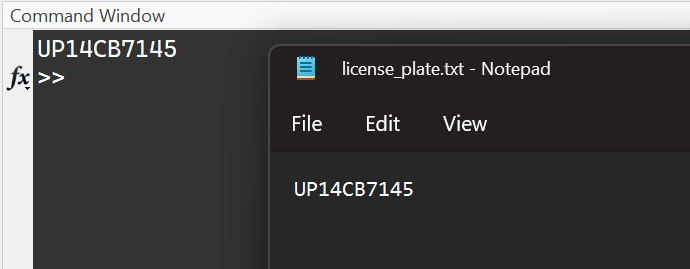


## سلول Recognition

در این سلول، مقدار correlation هر کدام از بخش‌های جداسازی شده توسط سلول قبلی را با تمام حروف دیتاست در نظر می‌گیریم و مقدار بیشینه آن را پیدا می‌کنیم. اگر این مقدار بیشینه از حدی که در متغیر SEGMENT\_THRESHOLD مشخص شده، بیشتر باشد، آن را معادل با حرف (یا عددی) که این مقدار بیشینه را ساخته در نظر می‌گیریم.

## سلول Output

در این سلول، تمام کاراکترهایی که در بخش قبل با یک کاراکتر دیتاست تطبیق پیدا کرده‌اند را ابتدا در کنسول چاپ می‌کنیم و سپس آن را در فایل license\_plate.txt نیز می‌نویسیم. نمونه‌ای از این مرحله در تصویر زیر قابل مشاهده است.



# بخش دوم – پلاک فارسی

این بخش نیز از تعدادی سلول تشکیل شده که هر کدام را به تفکیک ذکر می‌کنیم.

## سلول Initialization

این سلول شامل بستن تمام نمودارها، پاک کردن تمامی متغیرها و تعریف تعدادی متغیر constant می‌شود.

## سلول Load the Dataset

این سلول ابتدا بررسی می‌کند که فایل LICENSE\_LETTERS.mat که حاوی دیتاست برنامه است، وجود دارد یا خیر. اگر وجود داشت، فایل را لود کرده و در متغیر letters ذخیره می‌کند. اما اگر این فایل وجود نداشت، تابع make\_letterset را صدا می‌زند و این تابع تصاویری که در فولدر Map Set قرار دارند را می‌خواند و متغیر letters را برمی‌گرداند.

## سلول Input Image

در این سلول فقط تصویر پلاک مورد نظر انتخاب شده و ماتریس این تصویر در متلب لود می‌شود.

## سلول License Plate Detection

در این سلول هدف ما پیدا کردن محل پلاک در تصویر است. برای این کار از تعدادی روش استفاده شده که هر کدام را جداگانه توضیح می‌دهیم. در این بخش با توجه به اینکه دقیق‌ترین روش bluestrip است، ابتدا این روش را انجام می‌دهیم. اگر پلاک با استفاده از این روش یافت شد، نتیجه را برای پردازش نهایی در نظر می‌گیریم. در غیر این صورت، خروجی دو روش دیگر را در کنار همدیگر برای پردازش نهایی در نظر می‌گیریم.

### روش color\_changes

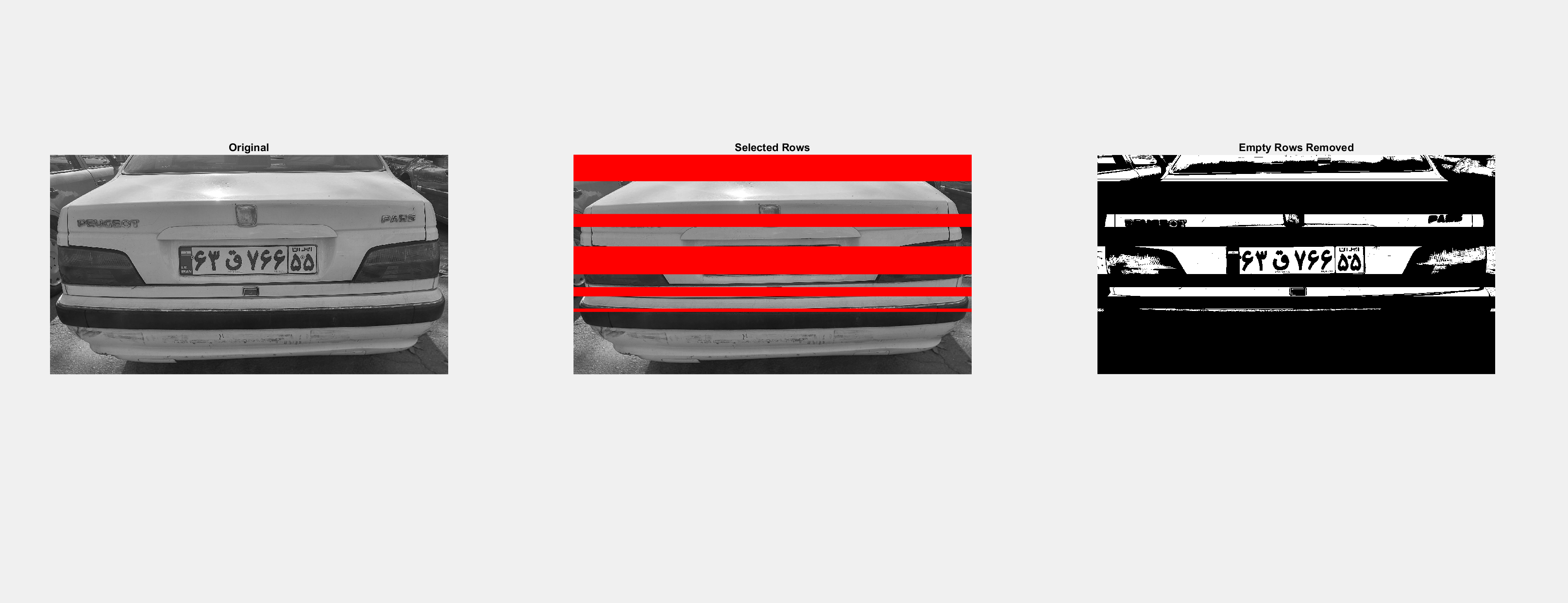
در این روش این حقیقت را در نظر می‌گیریم که پلاک اتومبیل سفید است و تعدادی حرف و عدد سیاه در آن قرار دارد. در نتیجه وجود پلاک باعث ایجاد تغییرات رنگی زیاد در آن بخش از تصویر می‌شود. در نتیجه هدف ما پیدا کردن بخش‌هایی از تصویر است که تعداد و مقدار تفاوت رنگ در آن بخش زیاد باشد. برای این کار ابتدا تصویر را به grayscale تبدیل می‌کنیم. سپس روی تمام سطرهای تصویر پیمایش می‌کنیم و سطرهایی که تعداد و مقدار تفاوت رنگ در آن‌ها از میانگین تمام سطرها بالاتر باشد را انتخاب می‌کنیم. در اینجا لازم است تعدادی اصطلاح را با توجه به نحوه استفاده در این بخش تعریف کنیم:

تفاوت رنگ: بین دو پیکسل مجاور در یک سطر تفاوت رنگ وجود دارد اگر تفاوت رنگ این 2 پیکسل (در تصویر grayscale) بیشتر از مقدار مشخص شده در متغیر GRAYSCALE\_COLOR\_CHANGE\_THRESHOLD (40 واحد) باشد.

تعداد تفاوت رنگ: تعداد جفت پیکسل‌های مجاوری که در یک سطر با هم تفاوت رنگ دارند.

مقدار تفاوت رنگ: مجموع اختلاف رنگ تمام جفت پیکسل‌های مجاور در یک سطر که با هم اختلاف رنگ دارند.

پیش‌تر سطرهایی که ممکن است شامل پلاک باشند را پیدا کردیم. حال تصویر را به باینری (سیاه و سفید یا Bit Map) تبدیل می‌کنیم و تمام سطرهایی که انتخاب نشده بودند (تفاوت رنگ در آن‌ها کم بود) را در تصویر جدید سیاه می‌کنیم. همچنین، با توجه به اینکه سمت چپ پلاک یک بخش آبی رنگ وجود دارد، اگر هر کدام از سطر‌های انتخاب شده در تصویر اصلی شامل پیکسلی با حداقل مقدار آبی BLUE\_VALUE\_THRESHOLD نبود، این سطر نیز حذف می‌شود. در تصویر باینری، اشیا کوچک را نیز حذف می‌کنیم. این مراحل را با یک مثال نشان می‌دهیم.



سپس با استفاده از تابع bwlabel، componentهای تصویر باینری را پیدا می‌کنیم. پس از آن از یک فیلتر دیگر استفاده می‌کنیم که کامپوننت‌هایی که نسبت طول به عرض آن‌ها خیلی پایین و یا خیلی بالا باشد را حذف می‌کنیم. حد بالا و پایین این فیلتر در متغیر REGION\_ASPECT\_RATIO\_THRESHOLDS ذخیره شده است. انجام دادن و یا انجام ندادن این فیلتر توسط متغیر filter\_by\_aspect\_ratio که این تابع به عنوان ورودی می‌گیرد، قابل تنظیم است.

پس از انجام مراحل ذکر شده، تعدادی کامپوننت در تصویر پیدا کردیم که ممکن است تعدادی از آن‌ها با هم overlap داشته باشند و یا اینکه به هم چسبیده باشند. برای رفع این مشکل، تمام نواحی‌ای که این مورد در آن‌ها صادق است را با هم merge می‌کنیم. نواحی باقی‌مانده برای تصویر بالا به صورت زیر خواهند بود.



همانطور که مشاهده می‌شود، پلاک خودرو نیز در یکی از نواحی یافت شده است. در نهایت تمامی این نواحی را به عنوان پاسخ بازمی‌گردانیم تا پس از پردازش نهایی، ناحیه صحیح را انتخاب کنیم.

### روش bluestrip

### روش aspect

## سلول Recongition

در این بخش ممکن است یک یا تعدادی ناحیه برای پلاک پیدا کرده باشیم. به ازای هر کدام از نواحی، عکس را crop می‌کنیم تا فقط ناحیه مورد نظر در تصویر باشد. سپس هر کدام از این تصاویر را به تابع recognize\_characters می‌دهیم تا پردازش نهایی را انجام دهد. این تابع ابتدا تصویر را resize و آن را باینری می‌کند (به صورت معکوس)، سپس اشیا کوچک (کمتر از SMALL\_OBJECT\_AREA) را از آن حذف می‌کند و پس از آن پس‌زمینه تصویر (بیشتر از BACKGROUND\_AREA) را حذف می‌کند. در نهایت، با استفاده از تابع bwlabel، کامپوننت‌های تصویر را پیدا می‌کنیم. حال 2 فیلتر را بر روی کامپوننت‌ها اعمال می‌کنیم. ابتدا کامپوننت‌هایی که نسبت طول به عرضشان و یا نسبت عرض به طولشان بیشتر از LONG\_ASPECT باشد، را حذف می‌کنیم. سپس، تمام کامپوننت‌هایی که تعداد پیکسل تشکیل دهنده آن‌ها کمتر از مقدار SMALL\_AREA باشد و کمتر از 30 درصد کامپوننت سفید باشد را نیز حذف می‌کنیم. شرط دوم برای جلوگیری از حذف شدن نقطه حروف و عدد 0 است.

حال باید کاراکترهای موجود در پلاک را تشخیص دهیم.

در نهایت خروجی بدست آمده را بازمی‌گردانیم. همانطور که پیش‌تر ذکر شد، ممکن است تعدادی ناحیه برای پلاک بدست آورده باشیم. در این صورت ناحیه‌ای را انتخاب می‌کنیم که تعداد حروف شناخته شده در آن، بیشتر از بقیه نواحی باشد.

## سلول Output

در این سلول مقدار خوانده شده از پلاک را ابتدا در کنسول و سپس در فایل license\_plate.txt می‌نویسیم. خروجی نهایی در تصویر زیر قابل مشاهده است.