بخش اول – پلاک انگلیسی

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

1. سلول Initialization

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

2. سلول Load the Dataset

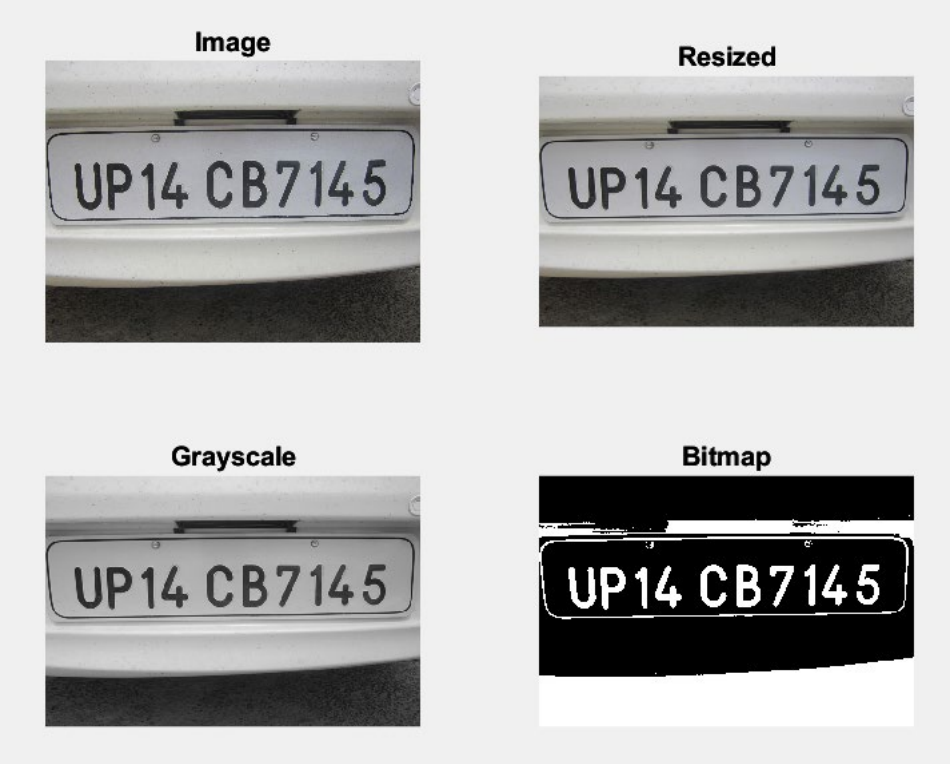
این سلول ابتدا بررسی می‌کند که فایل LICENSE_LETTERS.mat که حاوی دیتاست برنامه است، وجود دارد یا خیر. اگر وجود داشت، فایل را لود کرده و در متغیر letters ذخیره می‌کند. اما اگر این فایل وجود نداشت، تابع `make_letterset` را صدا می‌زند و این تابع تصاویری که در فولدر Map Set قرار دارند را می‌خواند و متغیر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

3. سلول Input Image

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

4. 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.

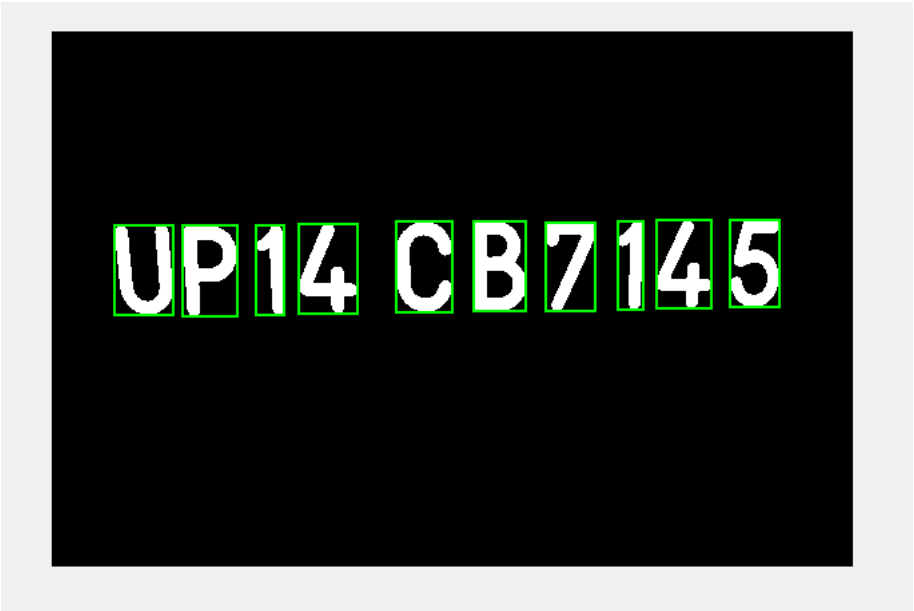


5. سلول Remove Background and Small Objects

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.

6. 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.

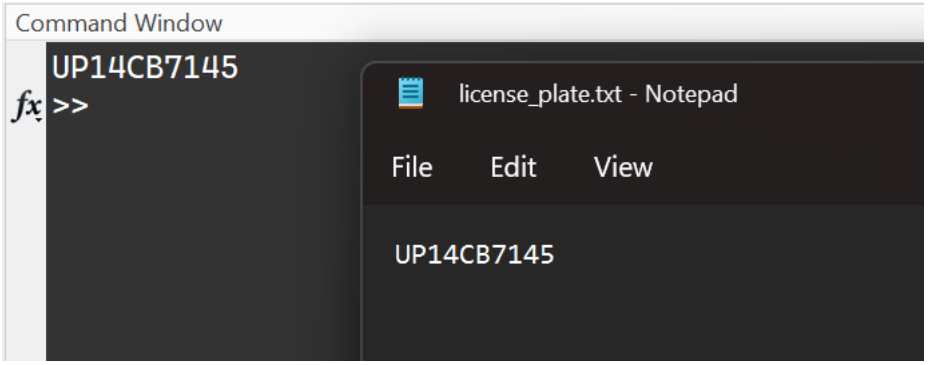


7. سلول Recognition

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.

8. 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.

1. سلول Initialization

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

2. 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.

3. سلول Input Image

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

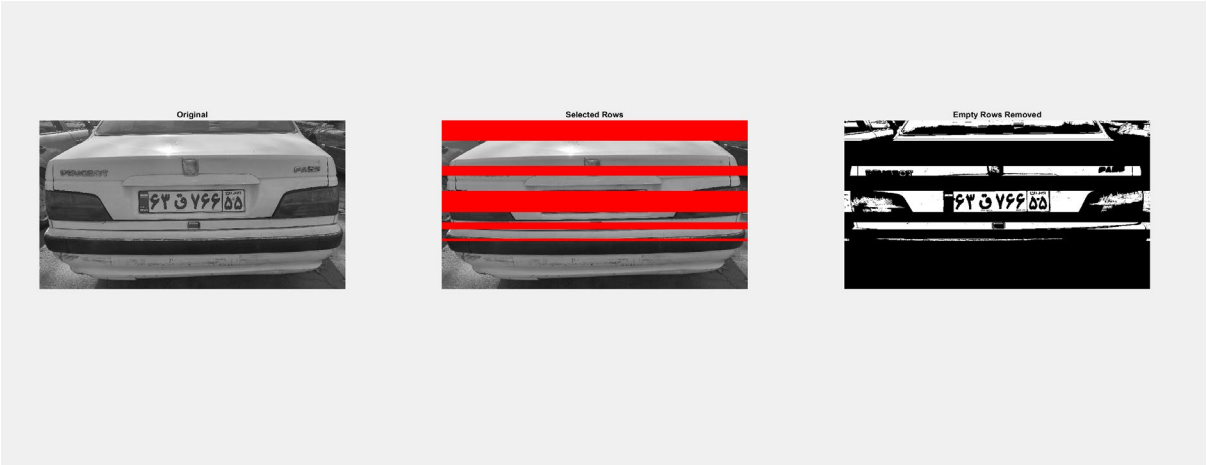
4. سلول License Plate Detection

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.

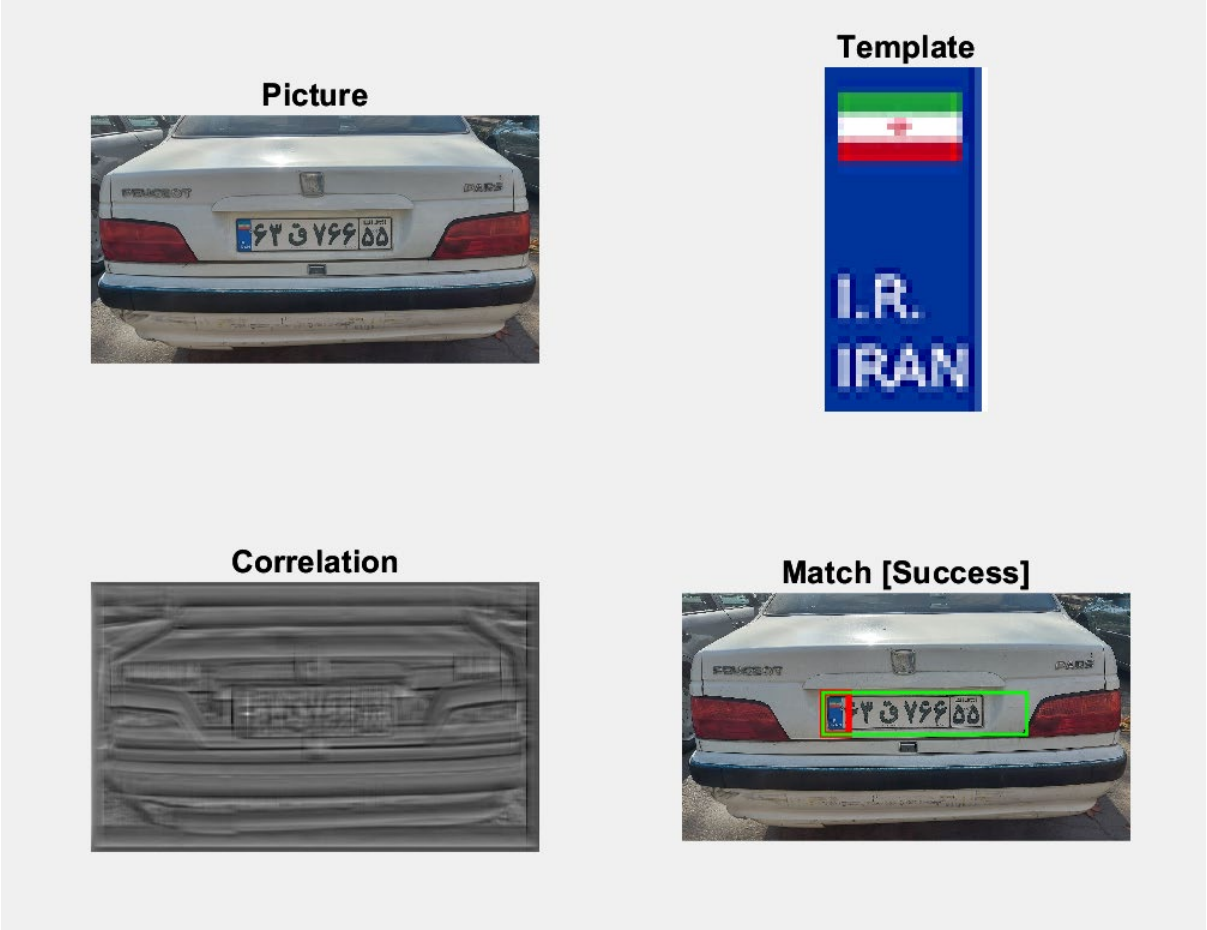
همانطور که مشاهده می‌شود، پلاک خودرو نیز در یکی از نواحی یافت شده است. در نهایت تمامی این نواحی را به عنوان پاسخ بازمی‌گردانیم تا پس از پردازش نهایی، ناحیه صحیح را انتخاب کنیم.

روش bluestrip

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.

از آنجا که نوار دینامیک تغییر اندازه نمی‌دهد و تطابق با عکس، بر اساس اندازه اصلی نوار است، دو عکس نوار با اندازه های متفاوت در نظر گرفته شده است و عکس های ورودی به اندازه ثابتی resize می‌شوند. یک عکس نوار کوچک‌تر بوده و برای عکس هایی که ماشین کامل است استفاده می‌شود و یک عکس هم بزرگ تر بوده و وقتی عکس ورودی خود پلاک است استفاده می‌شود.

از آنجا که cross correlation فقط روی یک چنل عکس انجام می‌شود، سه بار انجام شده و از میانگین آن استفاده می‌شود. سپس max آن به عنوان نوار آبی پلاک در نظر گرفته می‌شود. از آنجا با توجه به نسبت نوار آبی به پلاک، کل پلاک پیدا می‌شود.

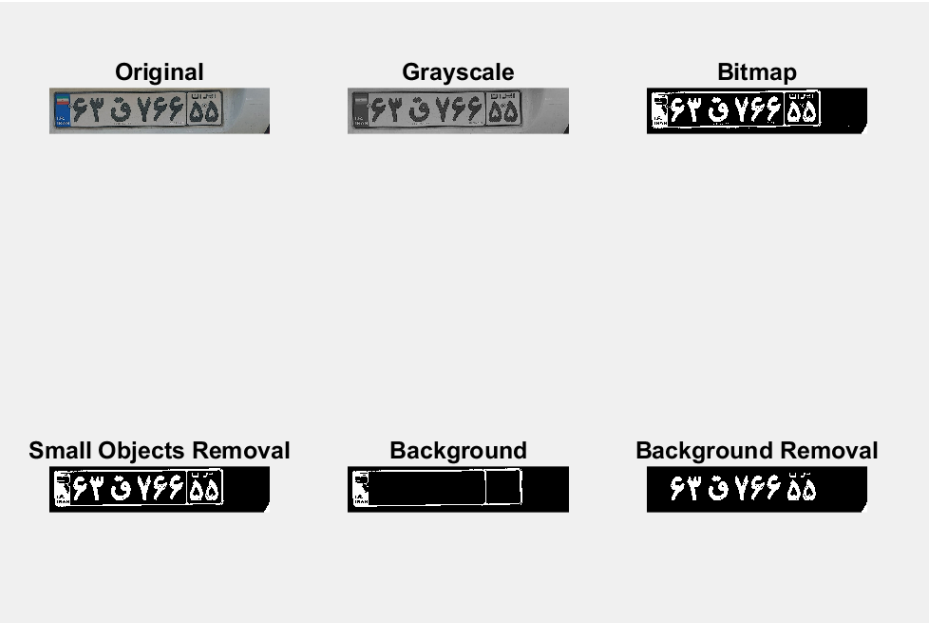


روش aspect

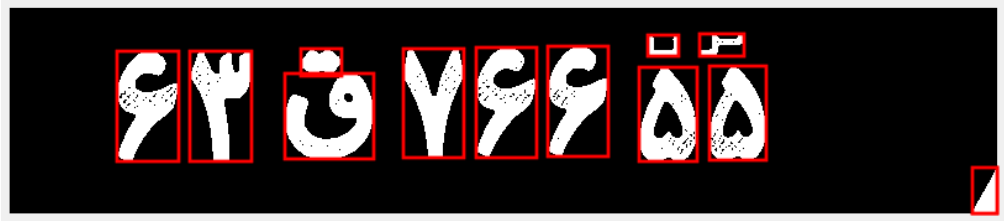
در این روش روی همه bounding box های تصویر شرط aspect ratio پلاک که حدود 4 است بررسی می‌شود و سپس بین آنهایی که این شرط را دارند، چک می‌شود که 20درصد سمت چپ آن رنگ آبی وجود داشته باشد و بقیه سمت راست نداشته باشد. اینطوری یک مرحله بیشتر فیلتر می‌شوند.

5. 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. به ترتیب از چپ منطقه های مورد نظر را بررسی می‌کنیم که باید به ترتیب 2 تا عدد، یک حرف و در نهایت 5 عدد بیاید. پس وقتی به حرف رسیدیم، از آنجا که ممکن است نقطه داشته باشد، منطقه بعدی را بررسی میکنیم که آیا از محور x در داخل ناحیه حرف قرار دارد یا خیر. اگر داشت آن دو را با هم merge می‌کنیم و دوباره correlation را با حروف بررسی می‌کنیم.



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.

6. 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: