CS 455 Term Project

License Plate Detection

By: Jianqiu Huang

Taylor Foxhall

Introduction

Automatic license recognition system is an important research field in the study of intelligent transportation and surveillance systems. These systems have many real world applications such as ticketless parking management systems, access control, border control and monitoring, and traffic law enforcement. There are many algorithms for license plate recognition systems. However, choosing the correct algorithm is totally based on criteria such as algorithm time complexity, lighting situation, and camera orientation. The majority of the license plate recognition systems are used in real-time systems; thus, the implementation should focus on providing both high accuracy and fast response time. License plate recognition systems consist of three basic sections: image acquisition, license plate detection, and optical character recognition. The operations for the license plate recognition system are illustrated as follows:

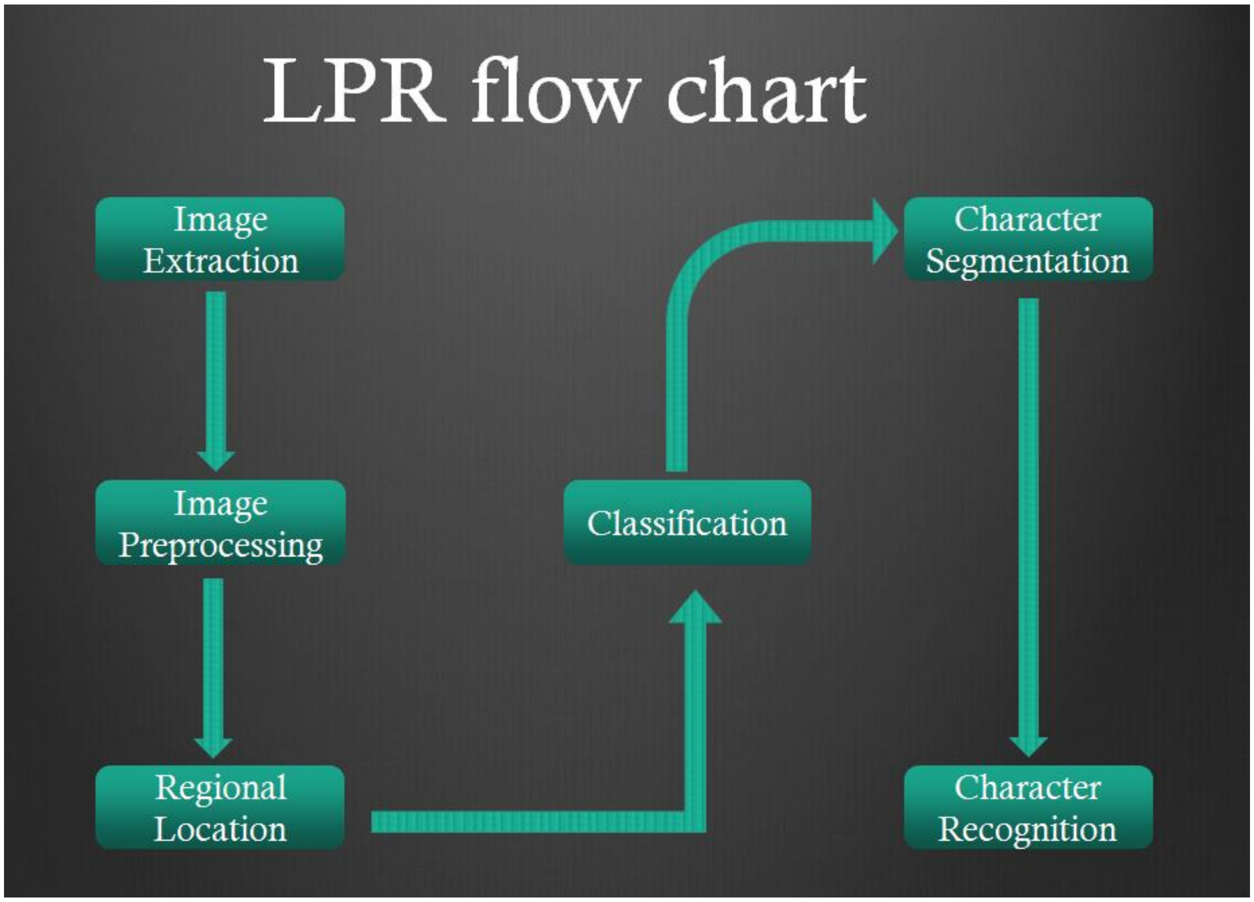
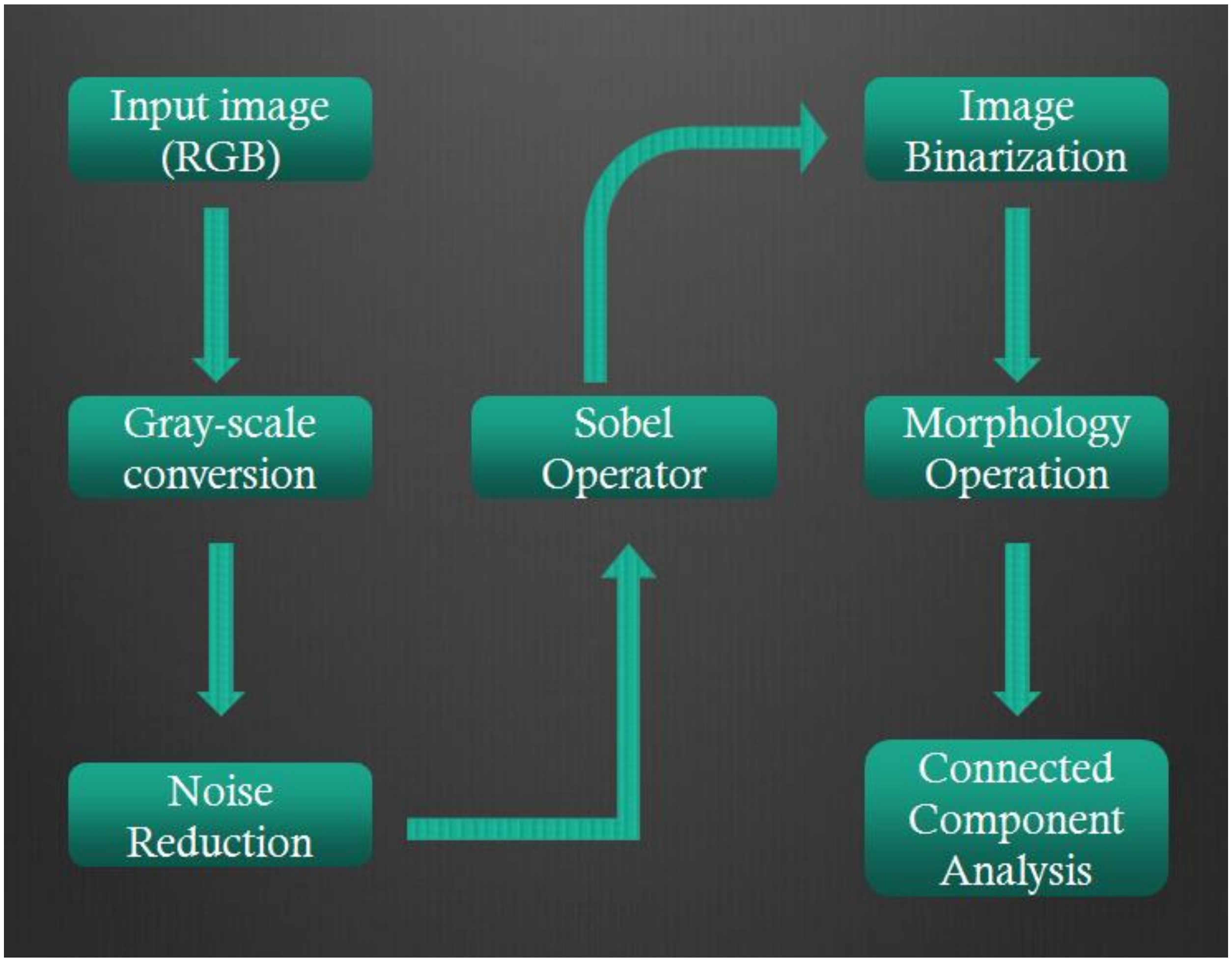


Image Acquisition

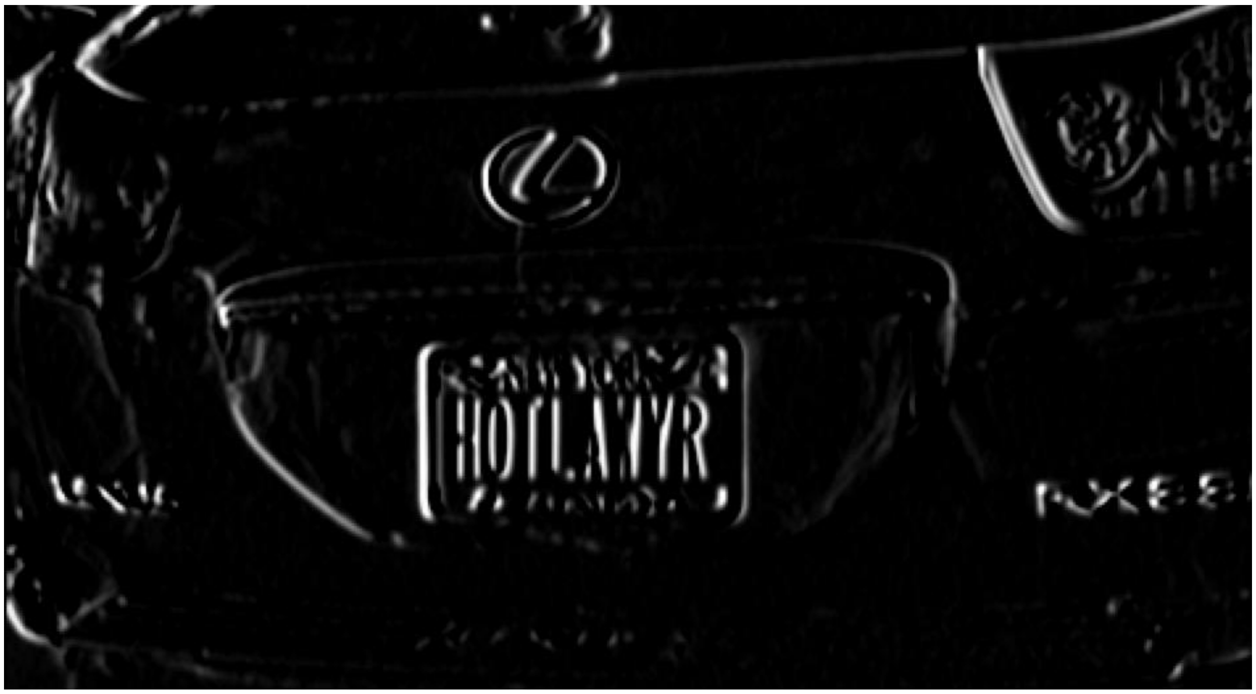
Image acquisition is always the first step in image processing work flow. It is defined as the action of retrieving an image containing visible license plate from a security camera. The image acquired is completely unprocessed. One of the ultimate goals of this process is to have the input frame to have some measured guidelines in image quality. In our project, we mainly focus on license detection and license number recognition, so we assume the image acquired has the basic standard for further processing.

License Plate Detection

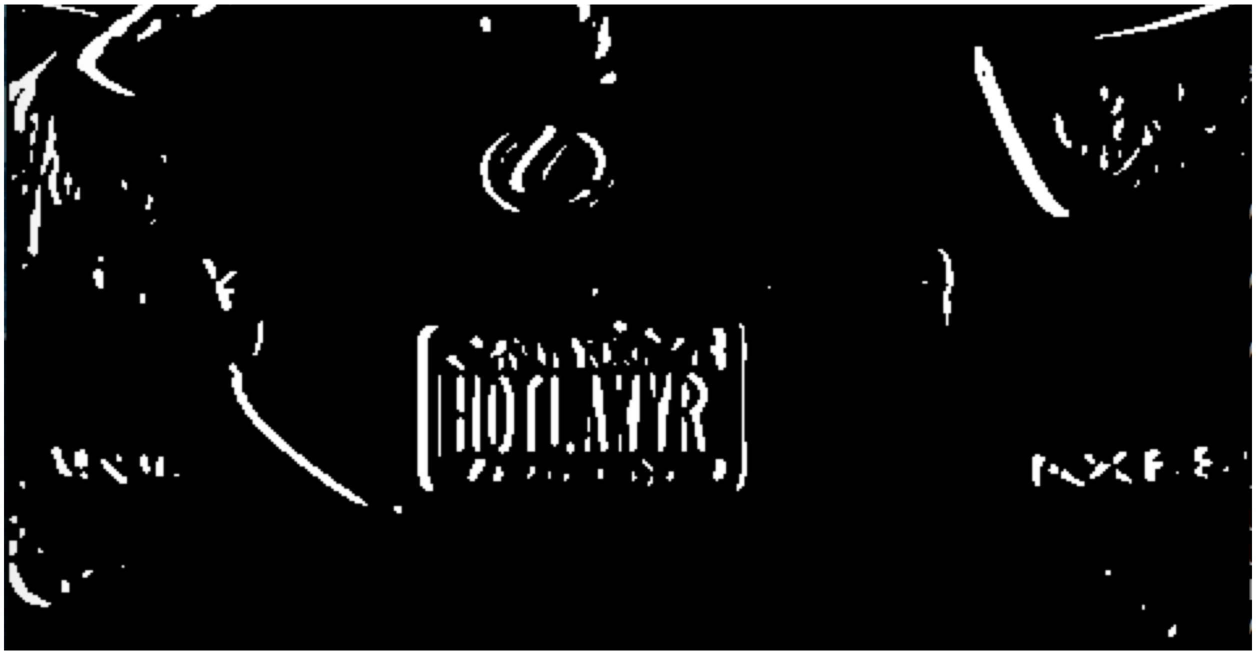
In license plate detection, we are to detect all the plates in the current camera frame. To further explain our approach, we divide it into three main steps: image preprocessing, detect potential license plate locations, and classification. The steps are shown in the diagram below.



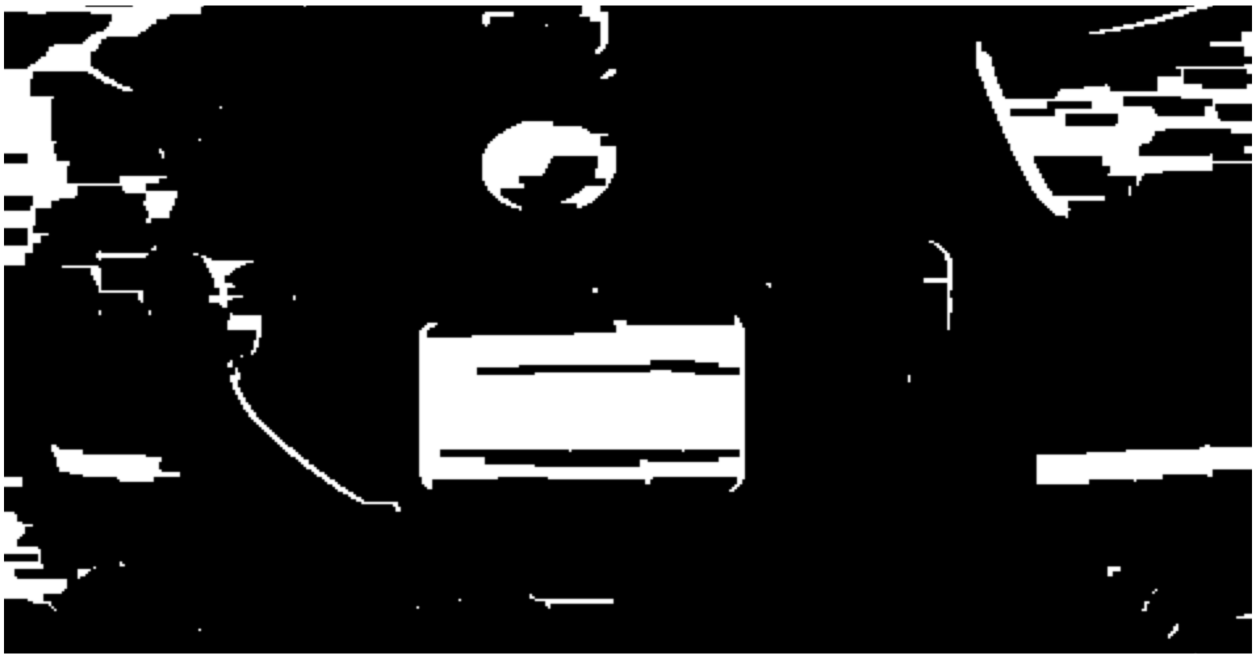
As for most security cameras, the image quality is often low, and edge information cannot easily be calculated. Hence, our first step is to compute all potential edges from the input video frame. We used a combination of Gaussian blur and Sobel operator to detect the edges in the image.



We then perform covert the image into binary image.



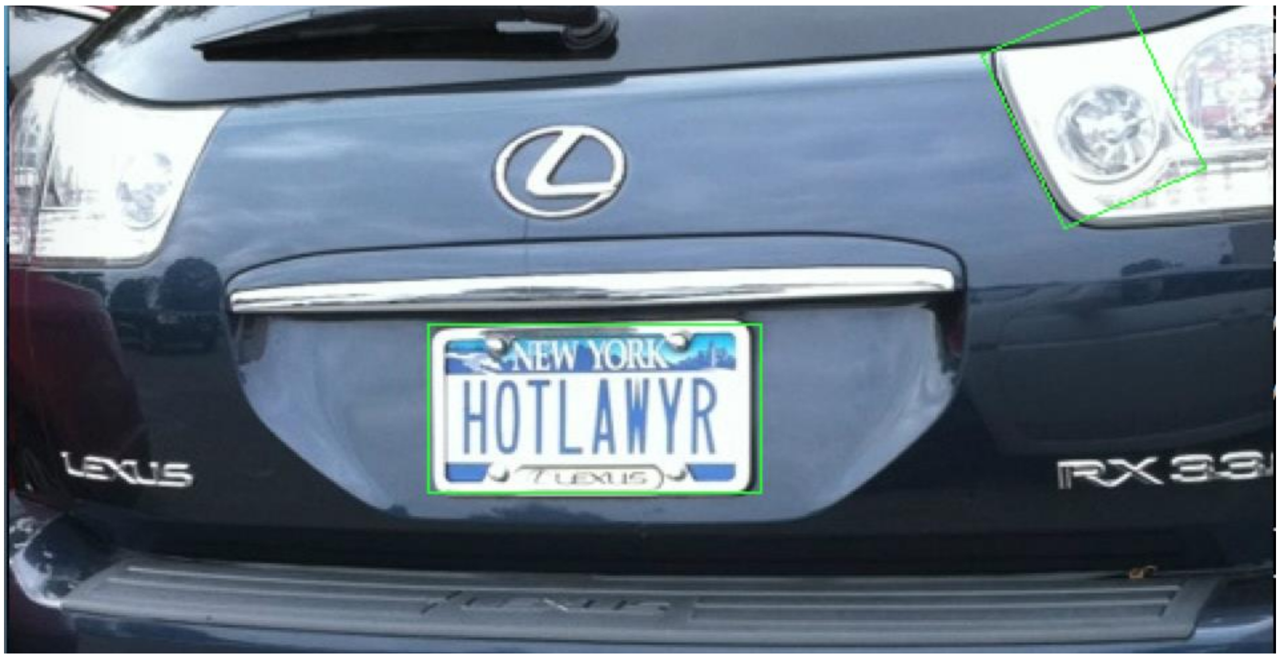
Now, we use closed morphological operation to fill the potential plate regions.



At the stage, it is obvious to human vision that the location of the license plate is near the center region of the image. To segment the interest regions from the above image, we used connected component analysis by using the findContour() method in OpenCV. For all contour detected, we only consider the regions with area that fall within our threshold.

```
float error=0.4;
//NY plate width: 32 height: 17 aspect ratio (width/height): 32/17
float aspect= 32 / 17.0;
//Set a min and max area. All other patchs are discarded
int min= 30*aspect*30; // minimum area
int max= 125*aspect*125; // maximum area
//Get only patchs that match to a respect ratio.
float rmin= aspect-aspect*error;
float rmax= aspect+aspect*error;
```

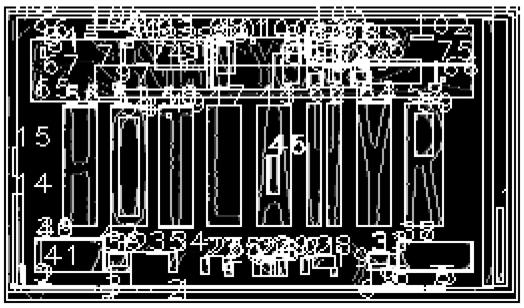
The remaining regions are shown as follow:



Even though thresholding has eliminated the majority of the false plate regions, usually, there will still be some false regions survive the test. To further examine the true plate region, we are to perform classification to eliminate all other false interest regions. The basic idea is that the true license plate region contains more contours than other false regions due to the existence of characters. Therefore, it is sufficient to simply count the number of contours in each interest region and return the region with maximum contours as the resulting true license location. Note that the contour in each region is also thresholded to ignore false contours. We obtain the cropped license plate.

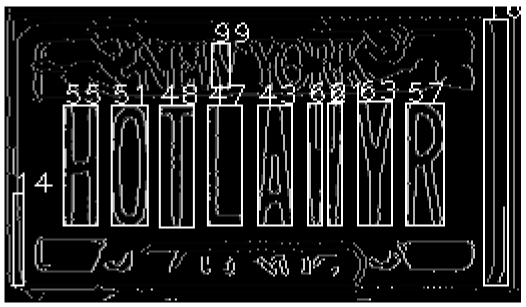
Binarization

After selecting the appropriate region of interest, we then need to perform some processing to help with text extraction. Based of the work of Kasar, we implement a text binarization algorithm to extract the license plate numbers. First we run a Canny edge detector on each of the RGB channels and merge each channel together into a single binary edge image. We then need to find the minimal area that each edge is contained in, its "bounding box."

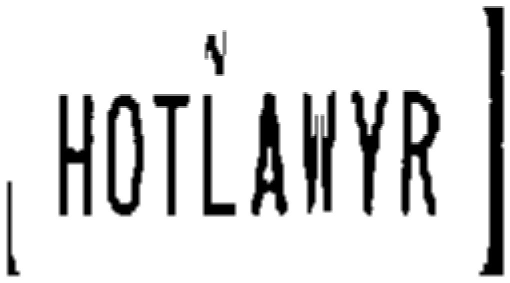


We can then begin the filtering of each edge. Since we're looking for letters and numbers, we can filter out contours whose bounding box has an aspect ratio greater than 1. We can make this assumption because the characters on license plates (at least in the US) are taller than they are wide. A contour can also be reasonably rejected if its bounding box has a total area less than 100 pixels, as we expect the characters on this license plate to be a relatively large portion of the image.

As can be seen in the image above, some characters consist of more than one contour. For instance the 'A' and 'R' letters both contain holes in them, so the initial contour detection picks up those holes. We won't want to consider those in our thresholding, so we must also eliminate them. To do this, we can search for parent contours, and if we find one, eliminate it only if the currently processed contour has no children. Since no character has more than two holes, we can also pick contours that have two or fewer children as candidates for filtering.

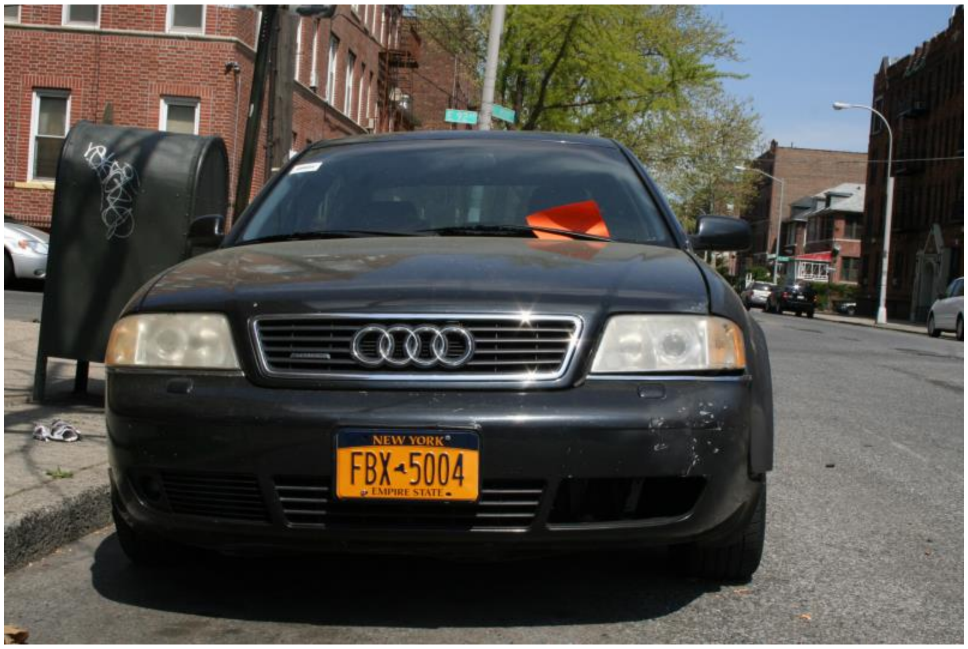


Now every contour with a box will be considered from here on out. To threshold, we need to determine the relative brightness of the foreground to the background. For each contour we find the mean intensity of the pixels in the original image at the location of the edge and use it as the foreground intensity. Then we examine the corners of the bounding box, and use the median value of intensities as the background intensity. Since we want black text on a white background, if the foreground is brighter than the background, then whatever pixels in the bounding box that are brighter than the foreground will be colored black and anything below that will be colored white. The coloring scheme is reversed if the background is brighter than the foreground. The image is then blurred for better results when run through an OCR.

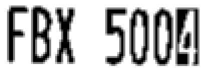


Results

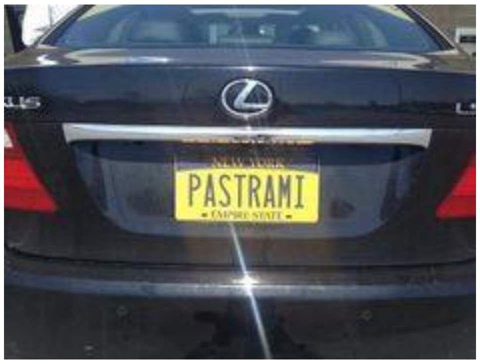
Unprocessed image:



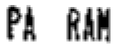
Result:



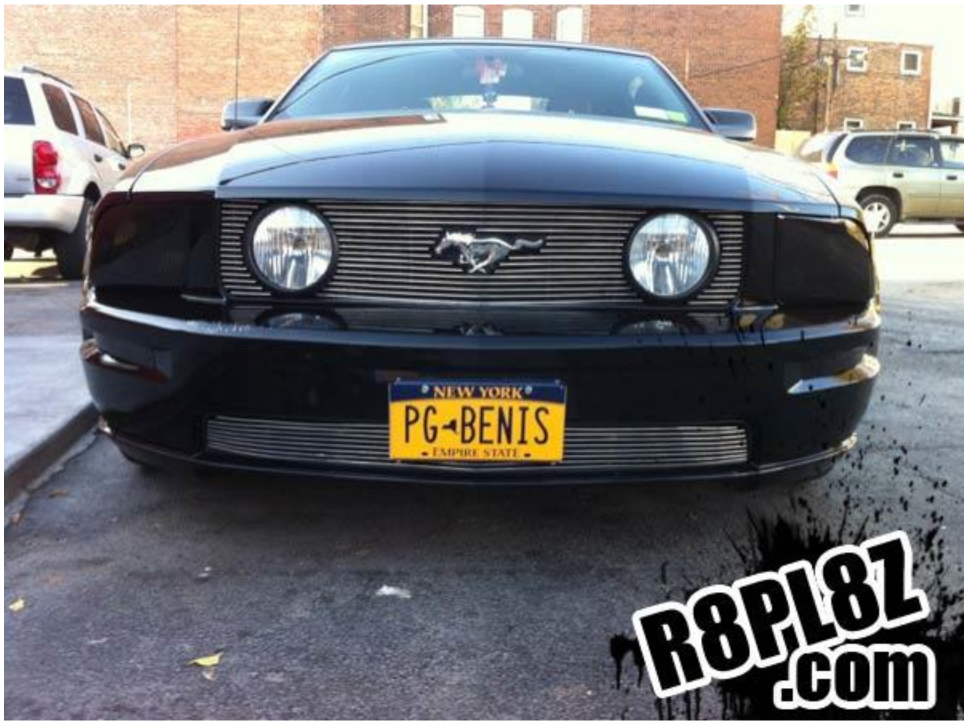
Unprocessed image:



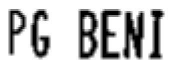
Result:



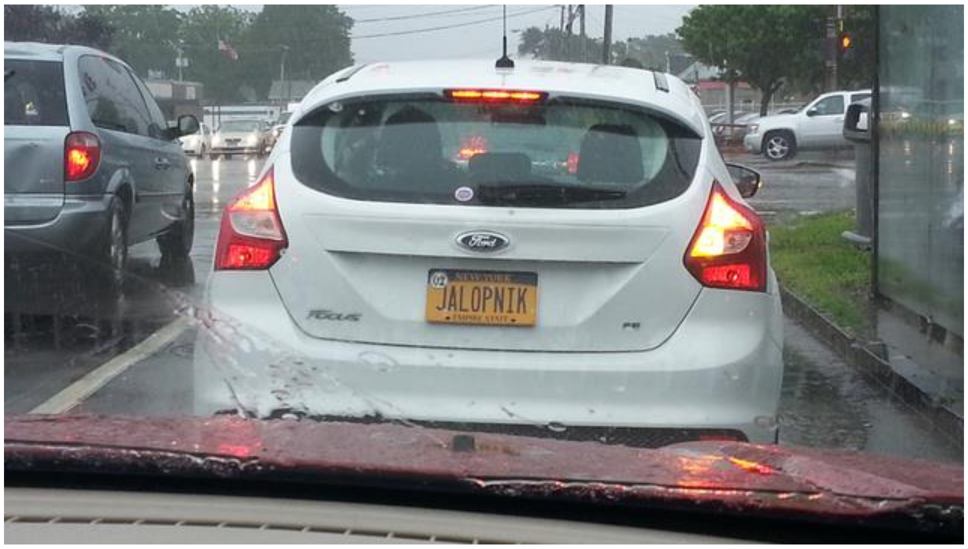
Unprocessed image:



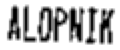
Result:



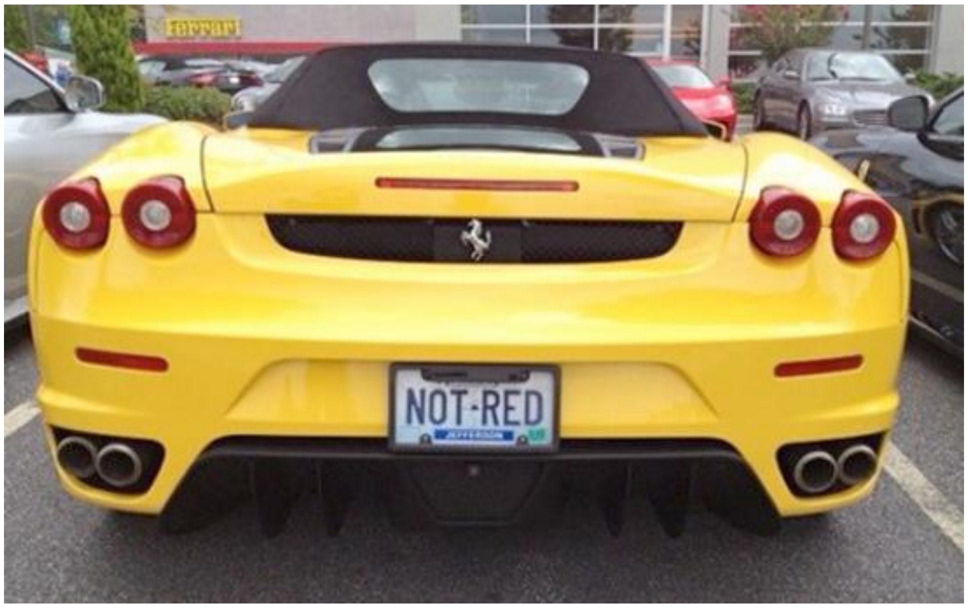
Unprocessed image:



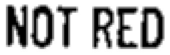
Result:


ALOPNIK

Unprocessed image:



Result:


NOT RED

Conclusion

The results are fairly accurate, while some characters on the edge of the license plates get clipped off. It appears that while the algorithm works in some good cases, certain lighting conditions and angles can really throw off the accuracy of detection. Clearly optimizations could be made in these cases. This process hinges on whether the right plate region is detected, and the accuracy could always be improved. Also, because it appears characters can be clipped off, there may be a desire to add a border to the image for processing to avoid contours being clipped at the edge of the image. Steps beyond this may also include taking steps to improve accuracy in when the extracted binary image is sent through an OCR software.

References

Dey, Suprokash, Amitava Choudhury, and Joydeep Mukherjee. "An Efficient Technique to Recognize License Plate Using Morphological Edge Detection and Character Matching Algorithm." International Journal of Computer Applications, 1 Sept. 2014. http://research.ijcaonline.org/volume101/number15/pxc3898883.pdf

T Kasar, J Kumar and A G Ramakrishnan. Font and Background Color Independent Text Binarization. Second International Workshop on Camera-Based Document Analysis and Recognition. 2007. http://www.m.cs.osakafu-u.ac.jp/cbdar2007/proceedings/papers/O1-1.pdf

"Welcome World!" : LICENSE PLATE DETECTION AND IDENTIFICATION USING OPENCV. Web. 15 Dec. 2015. http://visakh017.blogspot.com/2014/05/license-plate-detection-and.html