Locating number plates in cars — OpenCV & Python

Keyne Oei Follow
Mar 24, 2019 · 6 min read

- assignment of UQ-ELEC4630 (all resources from UQ)
- Code available in https://github.com/keynekassapa13/num-plate-track

Locating number plates are challenging specifically because each image has different characteristic and features. Creating only one system that satisfies every image character would be difficult without any initial classification. From 26 number plate images, I divided them to 3 major types which are images that have white number plates, images that have yellow number plates and images that have white number plates with white background car behind the plate. I used Python and OpenCV environment to complete this task. At the end of the task, the number plates that are successfully identified is 21 out of 26 (~ 80%).

Main steps to locate the number plates are:

1. Filtering image by threshold
2. Detect edges using Canny method
3. Contour the highlighted area
4. Draw convex hull on the biggest contour area
5. Cropped the picked area

Furthermore, I tried to use pattern matching method by OpenCV to detect the number in the numberplates. However, this has issue regarding the different sizes of template images and the detected number plates cropped image.

Filter Image by Threshold

There is two threshold method that I used in order to filter the image. The first one is simple thresholding. It is used to calculate if the pixel value is greater than a threshold value. The method types used for this is **cv.THRESH_BINARY + cv.THRESH_OTSU**. THRESH_BINARY is an enumerator that would assign white value when the input value is bigger than the threshold and THRESH_OTSU is an enumerator that use Otsu

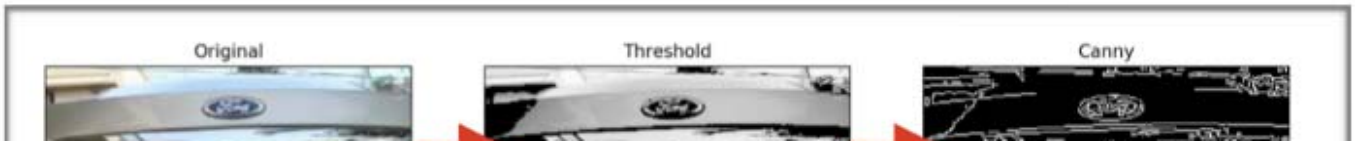
Algorithm to choose optimal threshold value. [1] Otsu algorithm works based on the clustering based image which will calculate the optimum threshold that divide the pixels in the image. [2]

Threshold method is very useful for images that have white number plates but does not work for number plates that have color other than white or number plates that have white background. This is due to similar density value if the image is converted to black and white image. The result will not differentiate between the plate and the background. To minimize this error, another threshold function from OpenCV could extract pixels that are in the range between the specified boundary. Fundamentally, this is to check if the element or input belongs between elements of two array. [1] To do this, **cv.inRange(HSV, lower_boundary, upper_boundary)** is performed. The

image is required to be converted from RGB to HSV beforehand inRange function applied because the function only supports HSV color space.

Canny Edge

One of the difficulties in locating the plate is when the plate has the exact same color as its background. In this case, the plate is white and the background is also white. Threshold function and color extraction function doesn't work as both numberplate and background color areas belong to the same area above the chosen threshold. One solution to solve this issue is to use Edge Detection to check if there are any edge that is similar to numberplate characteristic which is a rectangular (4 sides). In this case, I used canny edge detection because it automatically removes noises first unlike Sobel and Laplacian. The simplification method in canny is very useful since the image characteristic varies between each image. Canny edge detection in OpenCVsmooth the image using 5x5 Gaussian filter to reduce noise and using non-maximum suppression to remove unwanted pixels. [1] After the image edge is detected, the next step is to acknowledge the biggest area and find the edge that has 4 sides.



Contour

After filtering image features, the image is simply put to contour in order to analyse any shape of the detected area. The purpose of this method is to detect an object which is number plate that has 4 sides (rectangle). To generate contours, OpenCV provide a **cv.findContours(img, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)**. cv2.RETR_TREE is a retrieval mode and cv2.CHAIN_APPROX_SIMPLE is contour approximation method. The output of this method would consist of a list of areas that need to be filtered again. The number plate is detected if the side is equal to 4. Another method that I tried to improve better accuracy is to filter the contour using convex hull. OpenCV is using Sklansky algorithm to perform the convex hull operation from **cv.convexHull(contour).** [1]

From these 2 filtered methods, convex hull method is proven to be more efficient method specifically for number plates that have different color than its background type. On another side, if the number plate has the same

color with its background, then deciding if the area has 4 sides method would be more efficient since the area is already reduced from canny edge detection.

Pattern Matching to detect numbers

Out of curiosity, I tried pattern matching to detect numbers in number plate. The tutors mention about 2d cross correlation method which is the same method of cv2.matchTemplate in OpenCV. The difficulties from achieving this are the different size of the template and the cropped image of number plates. The cross correlation method only works if the size of the number in plate number and template image is exactly the same. Some number plates are cropped within the size of templates and some are bigger or smaller than expected. This is operated using **cv2.matchTemplate(crop_img, template_img, cv2.TM_CCOEFF_NORMED)**. Consequently, this method ended with a result that has a high error rate and doesn't work as expected. However, these are some result of the number detected that is close to its expected result.

Performance

Out of 26 images that provided from tutors, the codes are able to detect **21** images correctly. There are 5 images that has number plates and couldn't be detected. These are:

There are some reasons why the codes are not able to detect the number plate. The foremost reason for this issue is the inability of the algorithm to identify the different characteristic between the number plate and its background that has the same color. Even with canny edge detection algorithm, it also doesn't work because some of the backgrounds have the same 4 sides which then it'll detect the background instead of the number plate. Another reason is because of the unfamiliar number plate of the car.

Thus, this needs more conditions to specify if the image has a number plate or not.

Solutions to solve this that I can suggest is hard-coded each undetectable image. For example, in the yellow car image, it detects the back window because it has the same density with the number plate. To solve this, possible area of number plate positions could be included in the system parameters. Thus, the system will crop the image to the middle of the image.

Or maybe Machine Learning/Deep Learning :)

References

1. OpenCV Documentation. Retrieved from docs.opencv.org
2. Otsu Method. Retrieved from https://en.wikipedia.org/wiki/Otsu%27s_method
3. SauceCat. (2017, April 30) *Boosting algorithm: AdaBoost*. Retrieved from https://towardsdatascience.com/boosting-algorithm-adaboost-b6737a9ee60c

Opencv Computer Vision Number Plate Cars