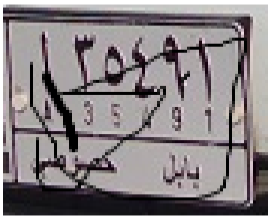
Detecting an object (words) in an image

Asked 10 years, 8 months ago Modified 10 years ago Viewed 9k times

20

I want to implement object detection in license plate (the city name) . I have an image:



and I want to detect if the image contains the word "بابل":

I have tried using a template matching method using OpenCV and also using MATLAB but the result is poor when tested with other images.

I have also read this page, but I was not able to get a good understanding of what to do from that.

Can anyone help me or give me a step by step way to solve that? I have a project to recognize the license plate and we can recognize and detect the numbers but I need to detect and recognize the words (it is the same words with more cars) java c++ matlab opencv image-processing

Share

Improve this question

Follow

edited Apr 21, 2014 at 10:57

asked Apr 12, 2014 at 8:24

Maadh

633 ● 4 ● 24

2 Template matching and SIFT/SURF detectors aren't the best ways to tackle character recognition; they are better fit to recognition of more detailed objects. Try looking at the answers given here, or at this example. You can always opt to go the neural network way, but this is not recommended as a first approach. – buzjwa Apr 12, 2014 at 9:01

Thanks @Naveh , can I know why my question has voted down please , it is important issue – Maadh Apr 12, 2014 at 10:04

I didn't cast the vote. I actually can't downvote, newbie that I am :). Read about how to ask good questions, and then you can rephrase yours and perhaps have the downvote removed. – buzjwa Apr 12, 2014 at 10:13

I didn't cast the downvote either but I suspect it is related to the fact that your question is theoretical and thus not about programming. You might want to post in the signal processing site. – Cape Code Apr 21, 2014 at 14:07

My guess is that since you are dealing with arabic characters most OCR software and libraries won't work for you. – Alceu Costa Apr 24, 2014 at 21:58

2 Answers

Sorted by: Highest score (default)

47

✓

+50

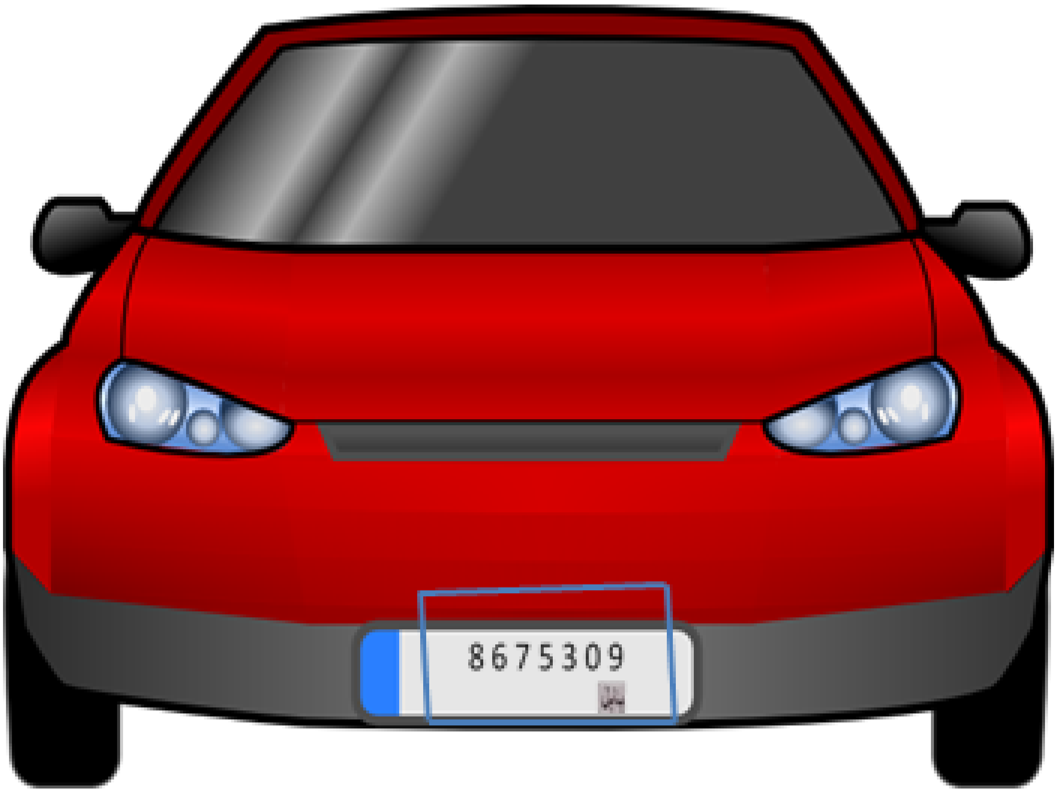
Your question is very broad, but I will do my best to explain optical character recognition (OCR) in a programmatic context and give you a general project workflow followed by successful OCR algorithms.

The problem you face is easier than most, because instead of having to recognize/differentiate between different characters, you only have to recognize a single image (assuming this is the only city you want to recognize). You are, however, subject to many of the limitations of any image recognition algorithm (quality, lighting, image variation).

Things you need to do:

1) **Image isolation**

You'll have to isolate your image from a noisy background:



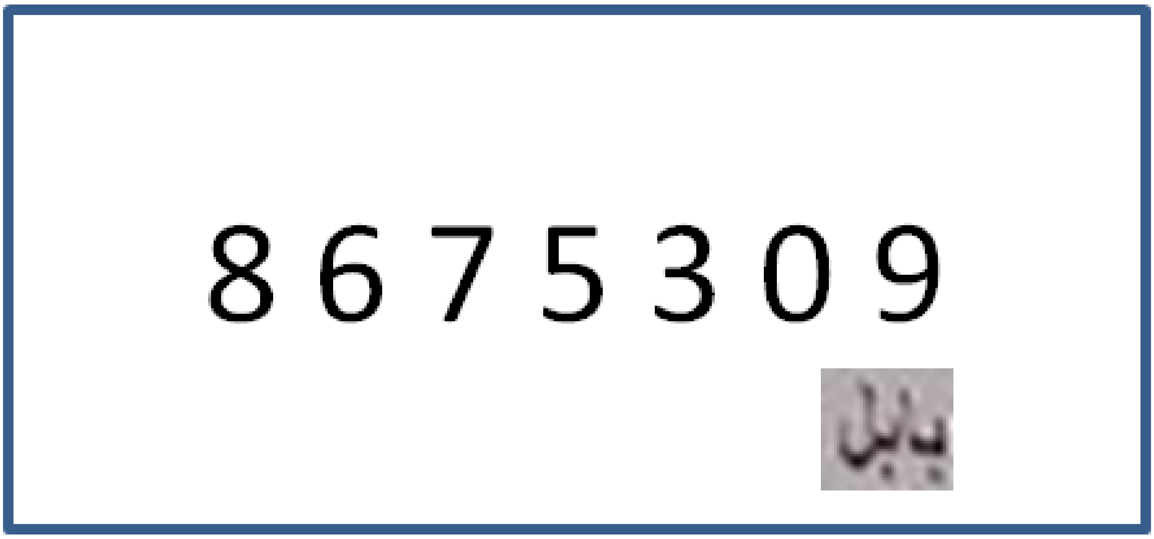
I think that the best isolation technique would be to first isolate the license plate, and then isolate the specific characters you're looking for. Important things to keep in mind during this step:

- Does the license plate always appear in the same place on the car?
- Are cars always in the same position when the image is taken?
- Is the word you are looking for always in the same spot on the license plate?

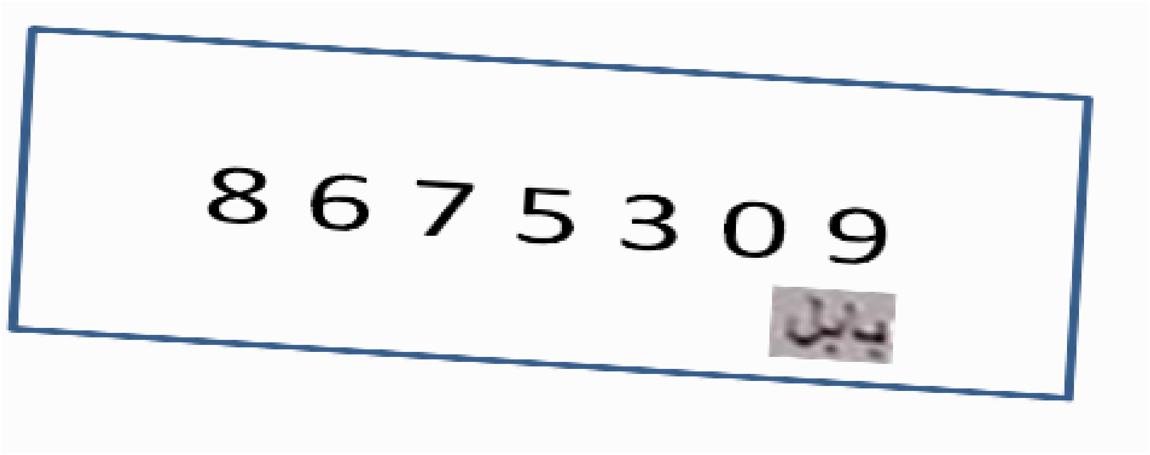
The difficulty/implementation of the task depends greatly on the answers to these three questions.

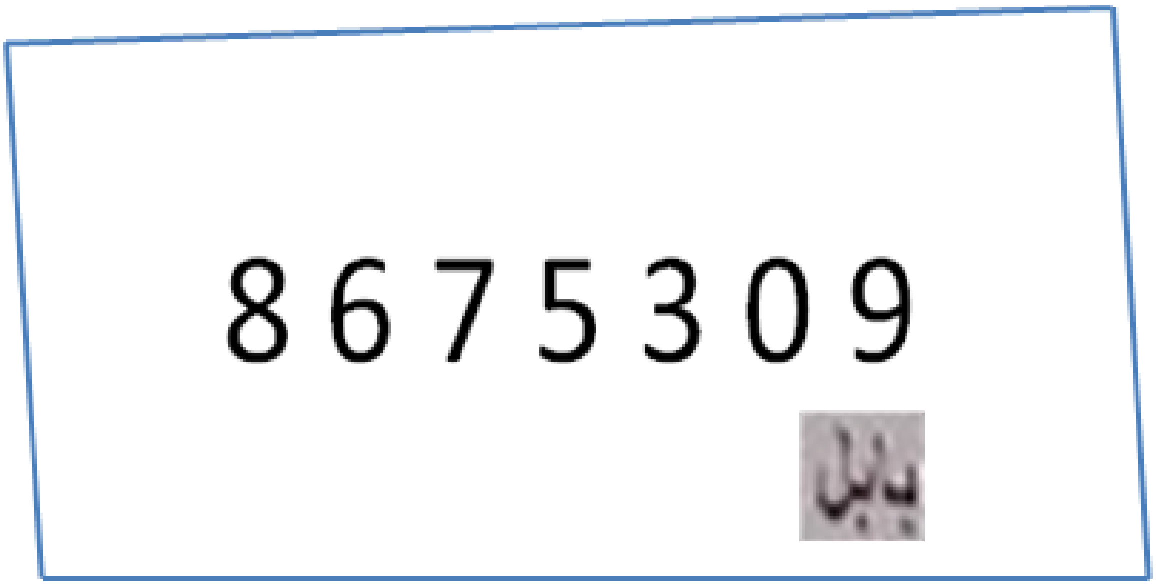
2) **Image capture/preprocessing**

This is a very important step for your particular implementation. Although possible, it is highly unlikely that your image will look like this:



as your camera would have to be directly in front of the license plate. More likely, your image may look like one of these:





depending on the perspective where the image is taken from. Ideally, all of your images will be taken from the same vantage point and you'll simply be able to apply a single transform so that they all look similar (or not apply one at all). If you have photos taken from different vantage points, you *need* to manipulate them or else you will be comparing two different images. Also, *especially* if you are taking images from only one vantage point and decide not to do a transform, make sure that the text your algorithm is looking for is transformed to be from the same point-of-view. If you don't, you'll have an not-so-great success rate that's difficult to debug/figure out.

3) **Image optimization**

You'll probably want to (a) convert your images to black-and-white and (b) reduce the noise of your images. These two processes are called binarization and despeckling, respectively. There are many implementations of these algorithms available in many different languages, most accessible by a Google search. You can batch process your images using any language /free tool if you want, or find an implementation that works with whatever language you decide to work in.

4) **Pattern recognition**

If you only want to search for the name of this one city (only one word *ever*), you'll most likely want to implement a matrix matching strategy. Many people also refer to matrix matching *as* pattern recognition so you may have heard it in this context before. Here is an excellent paper detailing an algorithmic implementation that should help you immensely should you choose to use matrix matching. The other algorithm available is feature extraction, which attempts to identify words based on patterns within letters (i.e. loops, curves, lines). You might use this if the font style of the word on the license plate ever changes, but if the same font will always be used, I think matrix matching will have the best results.

5) **Algorithm training**

Depending on the approach you take (if you use a learning algorithm), you may need to train your algorithm with data that is tagged. What this means is that you have a series of images that you've identified as `True` (contains city name) or `False` (does not). Here's a psuedocode example of how this works:

```
train = [(img1, True), (img2, True), (img3, False), (i

img_recognizer = algorithm(train)
```

Then, you apply your trained algorithm to identify untagged images.

```
test_untagged = [img5, img6, img7]

for image in test_untagged:
    img_recognizer(image)
```

Your training sets should be much larger than four data points; in general, the bigger the better. Just make sure, as I said before, that all the images are of an identical transformation.

Here is a very, very high-level code flow that may be helpful in implementing your algorithm:

```
img_in = capture_image()

cropped_img = isolate(img_in)

scaled_img = normalize_scale(cropped_img)

img_desp = despeckle(scaled_img)

img_final = binarize(img_desp)

#train
match() = train_match(training_set)

boolCity = match(img_final)
```

The processes above have been implemented many times and are thoroughly documented in many languages. Below are some implementations in the languages tagged in your question.

- Pure Java
- cvBlob in OpenCV (check out this tutorial and this blog post too)
- tesseract-ocr in C++
- Matlab OCR

Good luck!

Share Improve this answer Follow

edited Dec 8, 2014 at 23:52

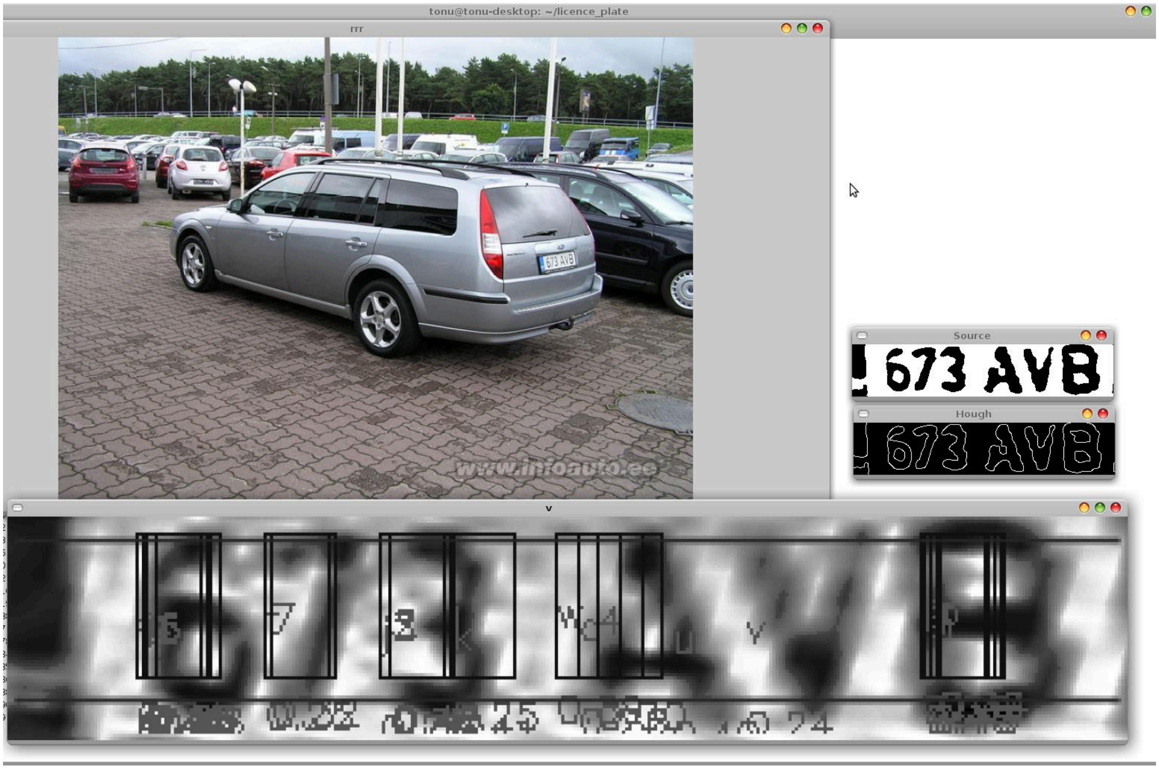
answered Apr 21, 2014 at 17:27

Luigi
4,129 ● 6 ● 38 ● 57

13

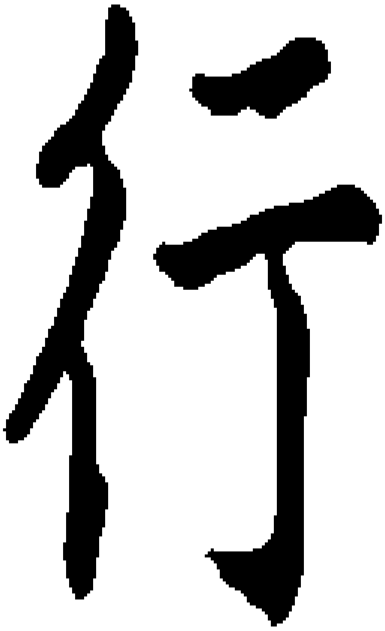
If you ask "I want to detect if the image contains the word "بابل" - this is classic problem which is solved using http://code.opencv.org/projects/opencv/wiki/FaceDetection like classifier.

But I assume you still want more. Years ago I tried to solve simiar problems and I provide example image to show how good/bad it was:



To detected licence plate I used very basic rectangle detection which is included in every OpenCV samples folder. And then used perspective transform to fix layout and size. It was important to implement multiple checks to see if rectangle looks good enough to be licence plate. For example if rectangle is 500px tall and 2px wide, then probably this is not what I want and was rejected.

Use https://code.google.com/p/cvblob/ to extract arabic text and other components on detected plate. I just had similar need yesterday on other project. I had to extract Japanese kanji symbols from page:



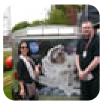
CvBlob does a lot of work for you.

Next step use technique explained http://blog.damiles.com/2008/11/basic-ocr-in-opencv/ to match city name. Just teach algorithm with example images of different city names and soon it will tell 99% of them just out of box. I have used similar approaches on different projects and quite sure they work

Share Improve this answer

Follow

answered Apr 25, 2014 at 2:25



Tõnu Samuel

2,896 ● 3 ● 22 ● 31

Please help me with same problem stackoverflow.com/questions/28495824/… – Dmitrij Holkin Feb 21, 2015 at 15:44