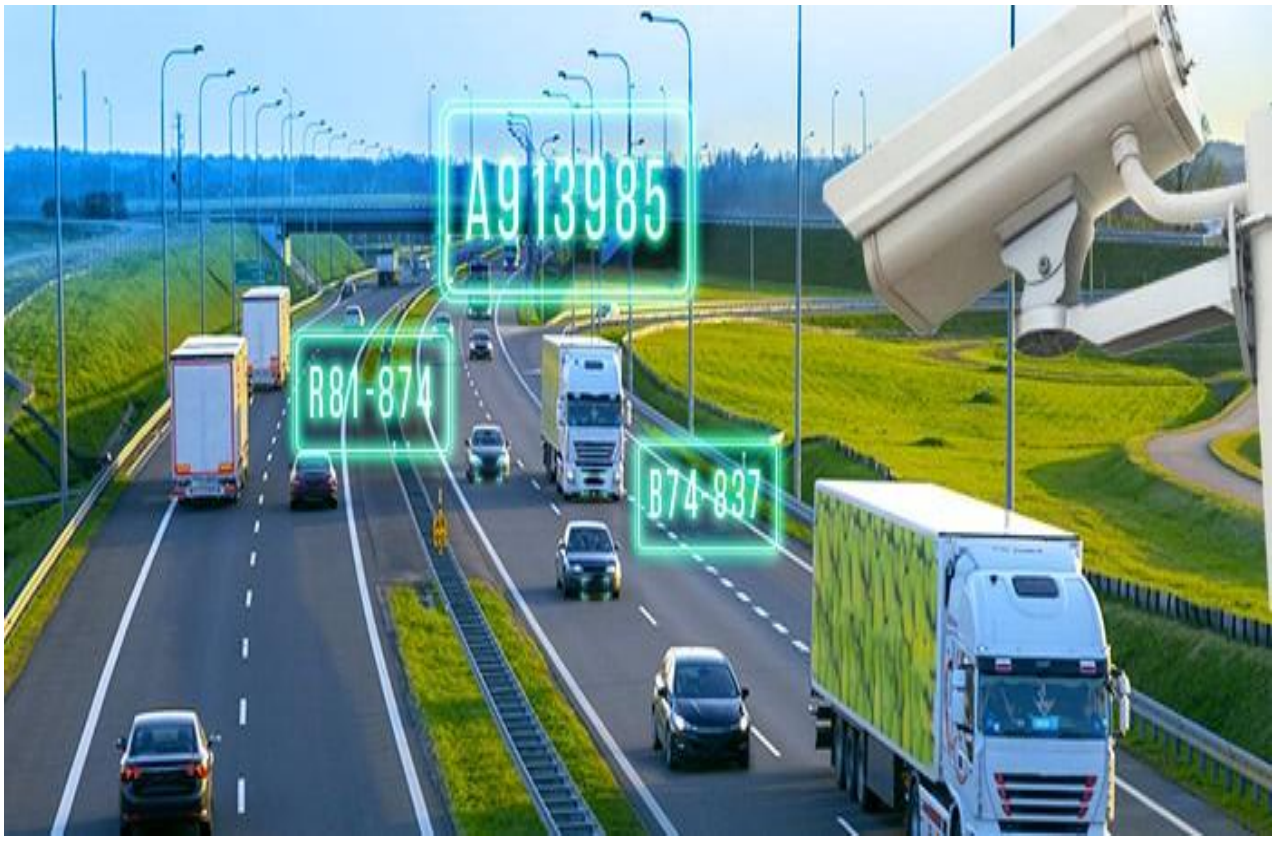
# Automated License Plate Recognition

Author – Durgesh Mishra




In this article, we're going to use OpenCV to detect license plates.

OpenCV (Open Source Computer Vision Library) is the huge open-source library for Computer Vision, Machine learning, and Image processing, now it plays a major role in real-time operations. OpenCV focuses on image processing, video capture, and feature analysis like face detection and object detection.

OpenCV makes use of Numpy in python, which is a highly optimized library for numerical operations. All the OpenCV array structures are converted to and from Numpy arrays, therefore making it easier to integrate with other libraries that use Numpy such as Matplotlib and SciPy.

Note that we will be using a Python Jupyter notebook to build our project. To work with OpenCV, we need to install OpenCV in a Jupyter notebook using the command ‘pip install opencv-python. Before we proceed further, here are the steps we will be following to build our project.

First, we import the image as an input that we want to work with. But Computers do not recognize or look at images the way we do, so to store an image on a computer, the image is first broken down into tiny elements called Pixels. The smallest element in a picture or image is called Pixel (Picture-Element = Pixel). A computer does not understand pixels as dots of color, it only understands numbers given in the form matrices. The computer uses various color models to convert colors to numbers. Broadly, we can classify images as grayscale or color images.

- In grayscale images (Black & White), each pixel is a single number stored in a matrix, representing the amount of light, or intensity, it carries. The Intensity range is from 0 (Black color) to 255 (White color). Everything between 0 to 255 is various shades of gray.

- In color images, pixels are often represented in the RGB (Red Green Blue) color model. Each pixel in a color image is a mix of these three colors. To store a color image, the computer stores three separate matrices corresponding to the red, green, and blue color channels of this image.

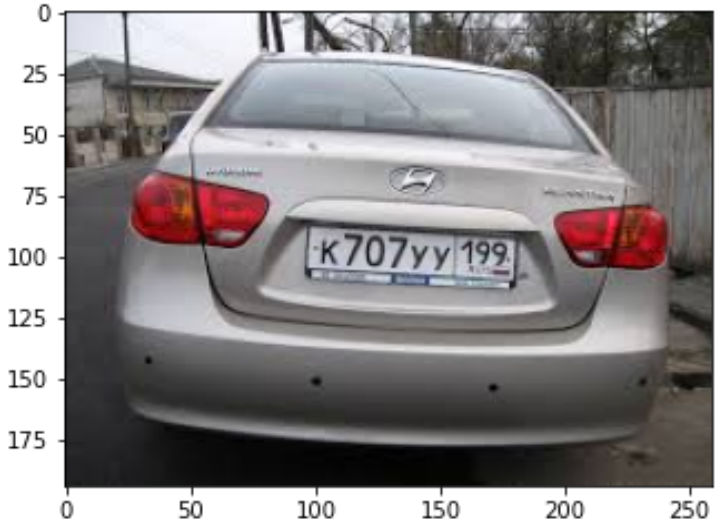
Note in OpenCv we use BGR format instead of RGB format for color images. The reason the early developers chose the BGR format instead of RGB format at OpenCV is that back then BGR color format was popular among software providers and camera manufacturers. We will need to run the cv2.cvtColor method to switch it to RGB format before we ask matplotlib to display the image.

As an example, if our input image is 64 x 64 pixels, we would have

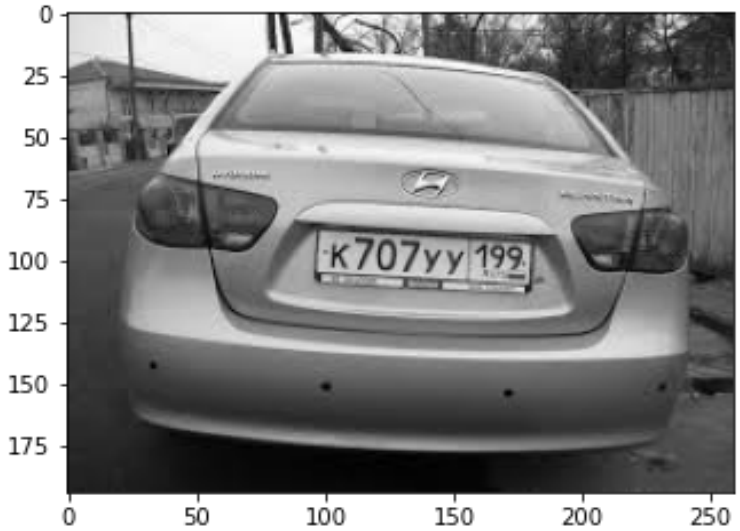
3 x (64 x 64) total number of pixels corresponding to the red, green, and blue pixel intensity values for our images.

A simple call to the cv2.imread() method loads our image as a multi-dimensional NumPy array (BGR in OpenCv) and cv2.imshow() displays our image to our screen.

The image we are going to use for our license plate detection.




It is easier to detect the license plate by converting the color image to a grayscale image first. cv2.cvtColor() method is used to convert an image from one color to another color space. There are more than 150 color-space conversion methods available in OpenCV Library.




OpenCV already contains many pre-trained classifiers, besides the installation of the OpenCV library, another important thing to retrieve is the Haar Cascade XML file containing the features for Russian car plates.

We will be using '*haarcascade_russian_plate_number*' detection in OpenCV. Object Detection using Haar feature-based cascade classifiers in OpenCV is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, 'Rapid Object Detection using a Boosted Cascade of Simple Features' in 2001.

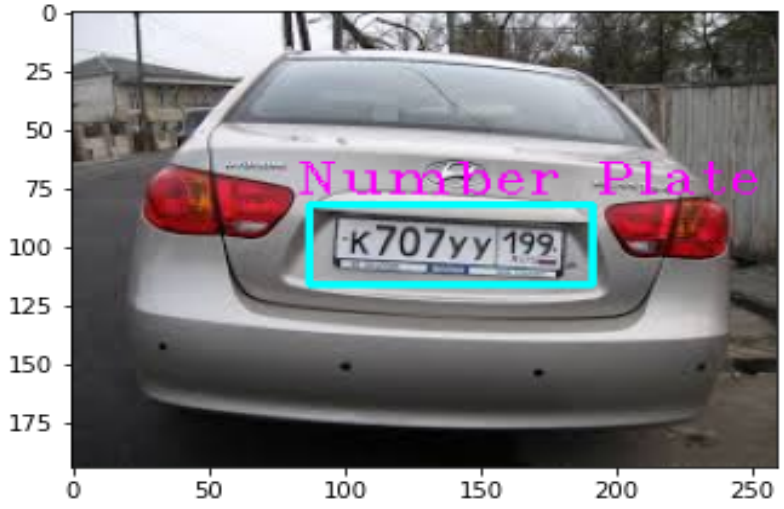
Haar feature-based cascade classifiers is a machine learning based approach where a cascade function is trained from a lot of positive images and negative images. It is then used to detect objects in other input images.

Next, we make use of the cv2.detectMultiScale() method of the CascadeClassifier to detect the license plates. The detected objects are returned as a list of bounding boxes (*x*, *y*, *w*, *h*) representing the location of each plate the classifier found.

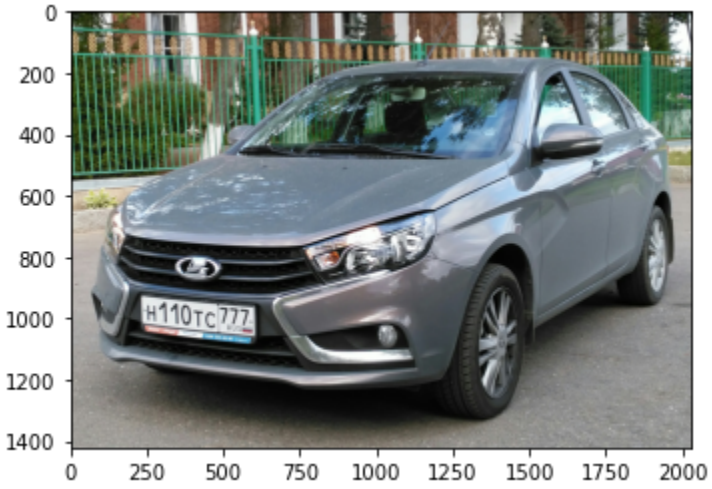
- X-coordinate of the bottom-left corner of rectangle (x)
- Y-coordinate of the bottom-left corner of rectangle (y)
- Width of the rectangle (w)
- Height of the rectangle (h)

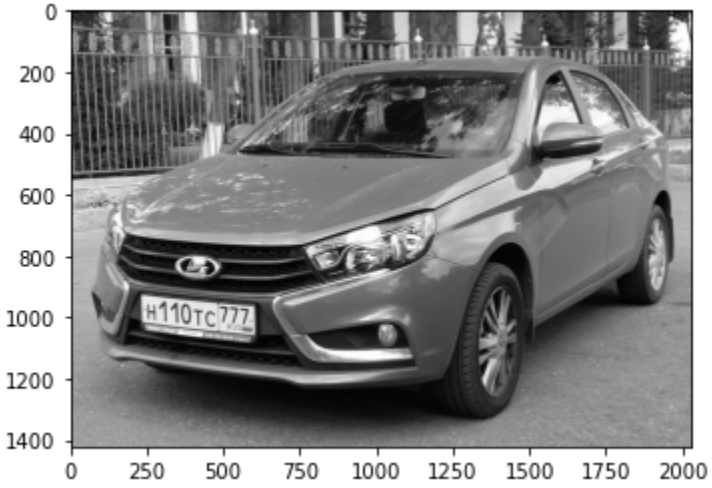
We now have the four endpoints of the license plate. We will draw a rectangle given the vertices of the rectangle are known, using the cv2.rectangle() method in OpenCV and use cv2.putText() method to draw a text string "Number Plate" on the input color image.

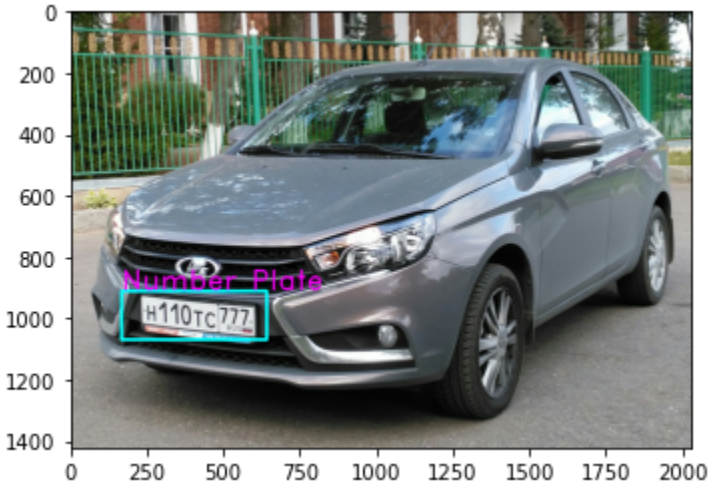
Here's the final image after detecting the license plate,



Here's another image used for the license plate detection.



## Conclusion:

In this article, we have covered how to set up OpenCV in Python, and how to make use of the powerful in-built functions of OpenCV to detect car license plates (Russian cars).

Github Link:
https://github.com/durg3sh10/Automated_License_Plate_Recognition

References:
https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

https://www.geeksforgeeks.org/license-plate-recognition-with-opencv-and-tesseract-ocr/

https://towardsdatascience.com/russian-car-plate-detection-with-opencv-and-tesseractocr-dce3d3f9ff5c