**DIP PROJECT REPORT**

**Recognition of Vehicle Plate**

**Group No: 4**

Shwet Khatri 19dcs001

Amit Gupta 19ucs011

Gunjan Paghdar 19ucs101

Manan Gandhi 19ucs130

Kush Gandhi 19ucs131

**Index**

* Objectives
* Description
* Solution
* Libraries Used
* Concepts Of Image Processing Used
* Code Snips
* Output
* References

**Objective:**

The main objective of number plate recognition is to ensure that rules are not violated because in today’s world with the increasing number of vehicles day by day it’s not possible to manually keep a record of the entire vehicle. Our software will be helpful to keep track of people who violate rules in cases of not wearing masks in current case scenarios, driving without helmet and overloading of passengers in public transport vehicles by tracking the number plates and charging them accordingly so as to avoid any such mishaps in future. It will be really useful in automating the world and reducing pressure on the human workforce on putting manual efforts to strain their eyes and recognise the number plate in order to penalise people to make them follow the traffic rules. It can also be used for charging parking tickets in malls, restaurants or for municipal corporations. As already the fastag has been made mandatory by the government so slowly parking charges can also be added to the fastags for making it a big time saver for the people. We can also use it to identify the stolen vehicles recognising their number plates making it less hectic for the police force and reducing their workload.

**Description:**

The VPR (Vehicle Number plate Recognition) system is based on image processing technology. We design an efficient automatic vehicle identification system by using vehicle number plates. The system first would capture the vehicle's image as soon as the vehicle reaches the security checking area. The captured images are then extracted by using the segmentation process. Optical character recognition is used to identify the characters. The obtained data is then compared with the data stored in their database. The system is implemented using Python(OpenCV) and performance is tested on real images. This system is mainly designed for the purpose of security system and video surveillance system is used for security purpose as well as monitoring systems. But detection of moving objects is a challenging part of video surveillance.

**Solution:**

Procedure For Vehicle Plate Recognition

1)Image Sharpening using unsharp masking.

2)Conversion to gray scale using cv2.

3)Applying bilateral filter for smoothing.

4)Canny Edge Detection for edge detection.

5)Now start finding contours,also we will sort the counters that have been detected and will look a counter having four sides and closed figure.

6)Now we will read characters from image using ‘pytesseract’ library.

7)We can find and print the text extracted from vehicle plate.

**Libraries Used**

* **OpenCV**

Opencv is an open source library which is very useful for computer vision applications such as video analysis, CCTV footage analysis and image analysis. OpenCV is written by C++ and has more than 2,500 optimized algorithms. When we create applications for computer vision that we don’t want to build from scratch we can use this library to start focusing on real world problems. There are many companies using this library today such as Google, Amazon, Microsoft and Toyota. Many researchers and developers contribute. We can easily install it in any OS like Windows, Ubuntu and MacOS.

* **Matplotlib**

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays.

It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code.

* **Pytesseract**

Tesseract is an open-source OCR (Optical character recognition) engine that can recognize more than 100 languages with Unicode support.

OCR = Optical Character Recognition. In other words, OCR systems transform a two-dimensional image of text, that could contain machine printed or handwritten text from its image representation into machine-readable text.

OCR as a process generally consists of several sub-processes to perform as accurately as possible. The subprocesses are:

Preprocessing of the Image

Text Localization

Character Segmentation

Character Recognition

Post Processing

Also, it can be trained to recognize other languages.

An OCR engine can save time by digitizing documents rather than manually typing the content of the document.

* **NumPy**

NumPy is a Python package that stands for ‘Numerical Python’. It is the core library for scientific computing, which contains a powerful n-dimensional array object.PythonNumPy arrays provide tools for integrating C, C++, etc. It is also useful in linear algebra, random number capability etc.

NumPy array can also be used as an efficient multi-dimensional container for generic data.Numpy array is a powerful N-dimensional array object which is in the form of rows and columns. We can initialize NumPy arrays from nested Python lists and access it elements.

**Concepts Of Image Processing Used**

* **Image Sharpening**

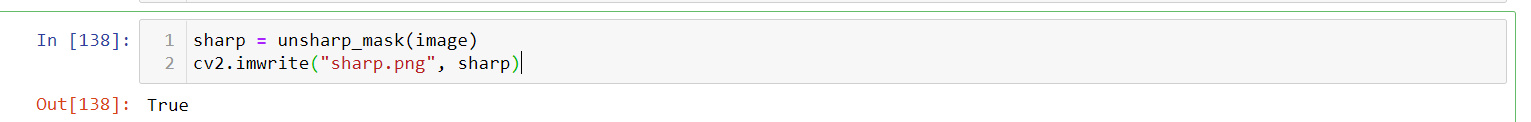
Function Used.: unsharp\_mask(image)

Image sharpening is an effect applied to digital images to give them a sharper appearance. Almost all lenses can benefit from at least a small amount of sharpening. Sharpening is applied in-camera to JPEG images at a level specified by the photographer or at the default set by the camera manufacturer.

Lightroom automatically applies some sharpening to images unless it is instructed not to. There are Photoshop techniques and filters like Unsharp Mask and Smart Sharpen to do the job. Sharpening works best on images whose blur did not stem from camera shake or drastically missed focus, though minor camera shake or slightly out-of-focus shots can also be fixed with sharpening.

Unsharp masking (USM) is an [image sharpening](https://en.wikipedia.org/wiki/Image_sharpening) technique, often available in [digital image processing](https://en.wikipedia.org/wiki/Digital_image_processing) software. Its name derives from the fact that the technique uses a [blurred](https://en.wikipedia.org/wiki/Gaussian_blur), or "unsharp", negative image to create a [mask](https://en.wikipedia.org/wiki/Mask_(computing)) of the original image. The unsharp mask is then combined with the original positive image, creating an image that is less blurry than the original.

Sample Execution(Screenshot)





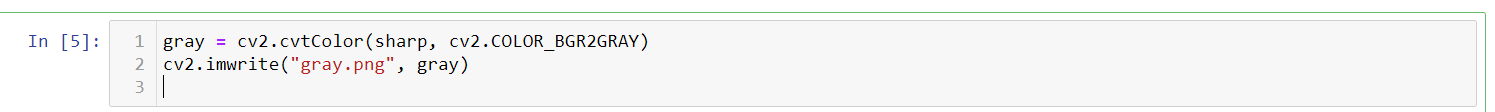
* **Grayscale Conversion**

Function Used.: cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

A grayscale (or gray level) image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a `gray' color is one in which the red, green and blue components all have equal intensity in [RGB space](https://homepages.inf.ed.ac.uk/rbf/HIPR2/rgb.htm), and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a [full color image](https://homepages.inf.ed.ac.uk/rbf/HIPR2/colimage.htm).

Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive gray levels is significantly better than the gray level resolving power of the human eye.

Grayscale images are very common, in part because much of today's display and image capture hardware can only support 8-bit images. In addition, grayscale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process color images.

Sample Execution(Screenshot)



* **Image Blurring**

Function Used .: cv2.bilateralFilter(gray,11,90,90)

Blurring an image is make the image less sharp. This can be done by smoothing the color transition between the pixels.

It reduce the noises in the image. A random brightness spot or incorrect color spot, depends on the type of noise, could be reduced by blurring the image with suitable type of blur.

The value of the filtered image at a given location is a function of the values of the input image in a small neighborhood of the same location. For example, Gaussian low-pass filtering computes a weighted average of pixel values in the neighborhood, in which the weights decrease with distance from the neighborhood center.

The intuition is that images typically vary slowly over space, so near pixels are likely to have similar values, and it is therefore appropriate to average them together. The noise values that corrupt these nearby pixels are mutually less correlated than the signal values, so noise is averaged away while signal is preserved.

Sample Execution(Screenshot)





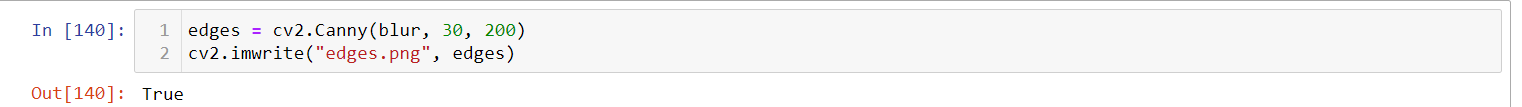
* **Edge Detection**

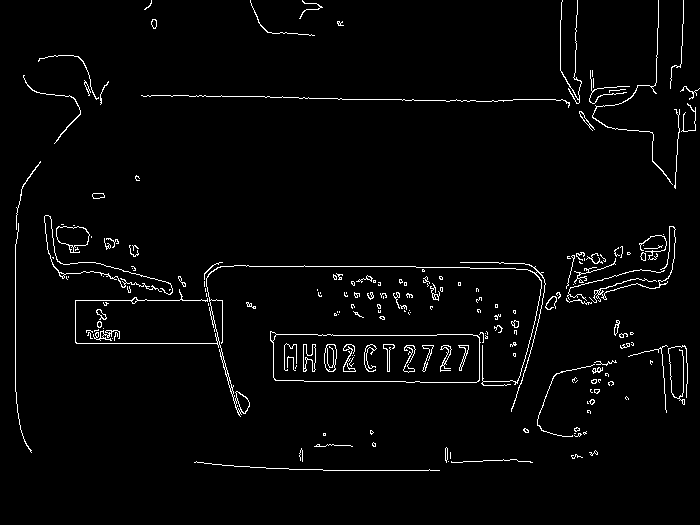
Function Used .: cv2.Canny(blur,30,200)

Canny edge detection, created by John Canny in 1986.

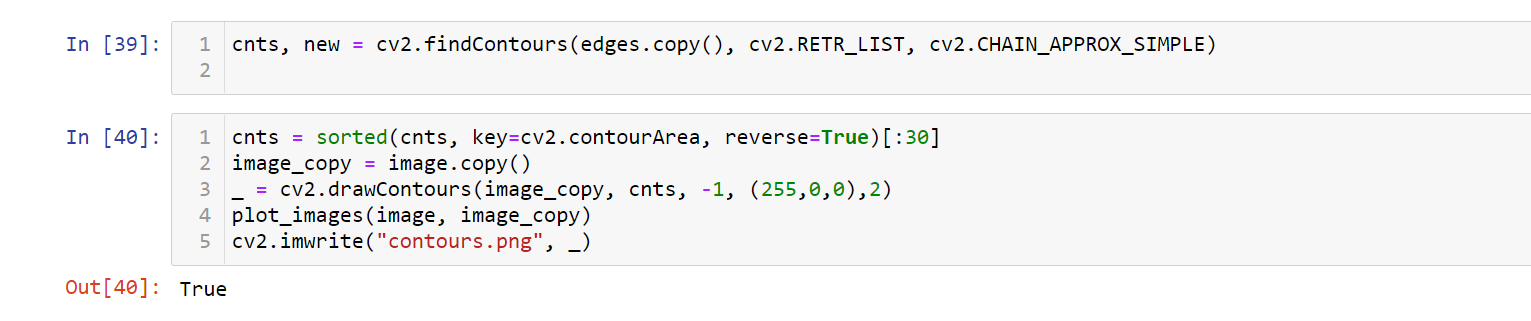
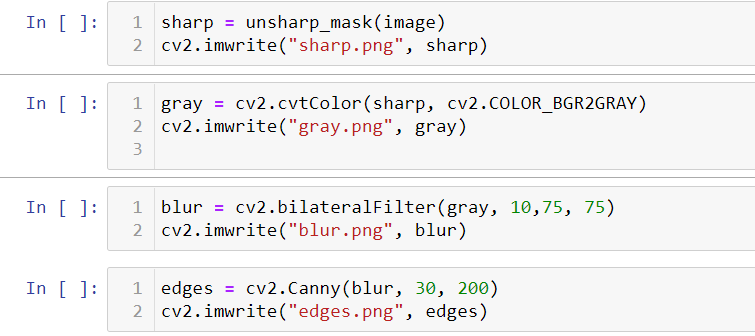
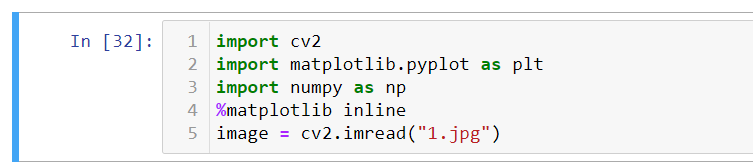
1. A Gaussian blur is applied to remove noise from the image.
2. Sobel edge detection is performed on both the x and y dimensions, to find the intensity gradients of the edges in the image. Sobel edge detection computes the derivative of a curve fitting the gradient between light and dark areas in an image, and then finds the peak of the derivative, which is interpreted as the location of an edge pixel.
3. Pixels that would be highlighted, but seem too far from any edge, are removed. This is called non-maximum suppression, and the result is edge lines that are thinner than those produced by other methods.
4. A double threshold is applied to determine potential edges. Here extraneous pixels caused by noise or milder color variation than desired are eliminated. If a pixel’s gradient value – based on the Sobel differential – is above the high threshold value, it is considered a strong candidate for an edge. If the gradient is below the low threshold value, it is turned off. If the gradient is in between, the pixel is considered a weak candidate for an edge pixel.
5. Final detection of edges is performed using hysteresis. Here, weak candidate pixels are examined, and if they are connected to strong candidate pixels, they are considered to be edge pixels; the remaining, non-connected weak candidates are turned off.

Sample Execution (Screenshot)

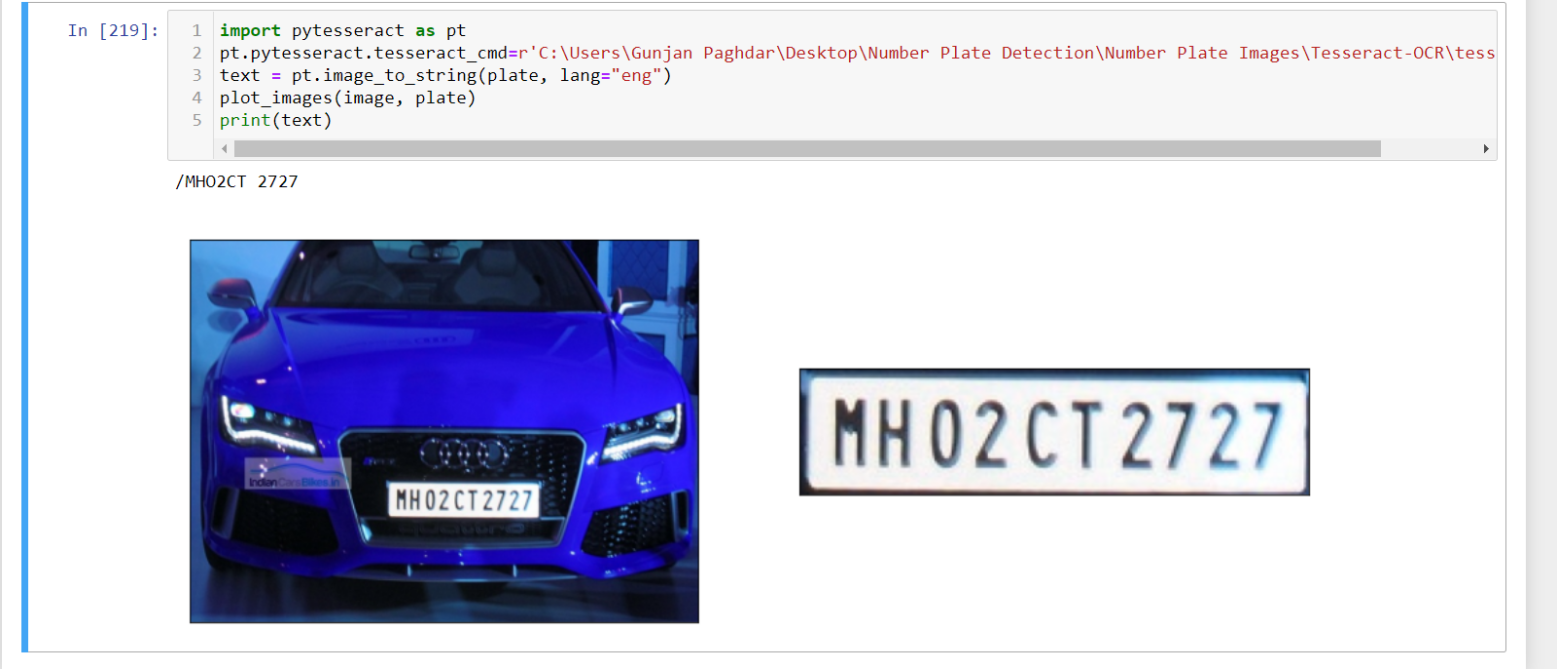


****

**Code Snips**

****

**Output**





**References**

* https://en.wikipedia.org/wiki/OpenCV
* https://en.wikipedia.org/wiki/NumPy
* https://en.wikipedia.org/wiki/Matplotlib
* https://pypi.org/project/pytesseract/
* https://www.youtube.com/watch?v=oXlwWbU8l2o
* https://www.youtube.com/watch?v=rfYLeH9vT9o