**License Plate Detection And Recognition Using OpenCv And Pytesseract**

License plate detection is identifying the part of the car that is predicted to be the number plate. Recognition is identifying the values that make up the license plate.

License plate detection and recognition is the technology that uses computer vision to detect and recognize a license plate from an input image of a car.

This technology applies in many areas. On roads, it is used to identify the cars that are breaking the traffic rules. In security, it is used to capture the license plates of the vehicles getting into and out of certain premises. In parking lots, it is used to capture the license plates of the cars being parked. The list of its applications goes on and on.

Introduction

**Python** gives us the ability to create our license plate detection and recognition program. We achieve this by using three of its libraries; pytesseract, imutils, and OpenCv.

In this tutorial, we are going to learn the processes software passes to detect and recognize a number plate, how to use the three libraries we did mention above to create our program, and finally how to create a program that is capable of detecting and recognizing a license plate. We will use the **pycharm community edition** as our IDE since it is freely available on the internet. A person using Windows or Linux can follow through.

Prerequisites

To follow through this tutorial, the reader needs to:

* Be familiar with the **python** programming language
* Have **pycharm** installed on their computer.

Processes a software undergoes to detect and recognize a license plate

For software to detect and recognize a license plate, it undergoes three major processes.

* **Taking an image of a car as input** - The program takes in the input of the car in which the license plate is to be detected.
* **Processing the input** - The image taken as the input undergoes processing to detect the part of the car that is the license plate.
* **Recognizing the number plate** - The values of the detected license plate are extracted from the number plate image.

##### **importing the libraries we need**

**import cv2**

**import imutils**

**import pytesseract**

In the code above, we are importing each library we discussed. This will enable us to use some of the functions we will require from them.

##### **Specifying the path to which tesseract is installed**

**pytesseract.pytesseract.tesseract\_cmd = 'C:\Program Files\Tesseract-OCR\\tesseract'**

In the above code: 'C:\Program Files\Tesseract-OCR\\tesseract' is the path to which we installed tesseract. Replace this part with your path.

##### **Taking in our image input and resizing its width to 300 pixels**

**image = cv2.imread('test.jpg')**

**image = imutils.resize(image, width=300 )**

**cv2.imshow("original image", image)**

**cv2.waitKey(0)**

* image = cv2.imread('test.jpg'): We are taking in the image as our input. test.jpg is the name of the image. Feel free to replace it with your own.
* image = imutils.resize(image, width=300 ): We are resizing our image. image is the image we took as input. For width=300 we are resizing the width of that image to 300 pixels.
* cv2.imshow("original image", image): We are displaying the image after setting its width to 300 pixels. original image is the name of the window that displays the image. Feel free to give it your name.
* cv2.waitKey(0): We are waiting for any key on the keyboard to be pressed to continue executing the code that follows.

##### **Converting the input image to greyscale**

**gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)**

**cv2.imshow("greyed image", gray\_image)**

**cv2.waitKey(0)**

* gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY): We are creating a variable gray\_image. We are then passing our input image to cv2.cvtColor. cv2.COLOR\_BGR2GRAY specifies that the image should be converted to grey image.
* cv2.imshow("greyed image", gray\_image): We are displaying the image which is already converted to grey.

##### **Reducing the noise in the greyscale image**

**gray\_image = cv2.bilateralFilter(gray\_image, 11, 17, 17)**

**cv2.imshow("smoothened image", gray\_image)**

**cv2.waitKey(0)**

* gray\_image = cv2.bilateralFilter(gray\_image, 11, 17, 17): We are reducing the noise in the grey image hence smoothening it.
* cv2.imshow("smoothened image", gray\_image): We are displaying the already smoothened image.

##### **Detecting the edges of the smoothened image**

**edged = cv2.Canny(gray\_image, 30, 200)**

**cv2.imshow("edged image", edged)**

**cv2.waitKey(0)**

* edged = cv2.Canny(gray\_image, 30, 200): We are creating variable edged. We are then passing our smoothened image to cv2.canny to detect the edges in it.
* cv2.imshow("edged image", edged): We are displaying the image with the detected edges.

##### **Finding the contours from the edged image**

**cnts,new = cv2.findContours(edged.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)**

**image1=image.copy()**

**cv2.drawContours(image1,cnts,-1,(0,255,0),3)**

**cv2.imshow("contours",image1)**

**cv2.waitKey(0)**

* cnts: This represents the contours.
* RETR\_LIST: It retrieves all the contours but does not create any parent-child relationship.
* CHAIN\_APPROX\_SIMPLE: Removes all the redundant points on the contours detected.
* image1=image.copy(): We are making a copy of the original input image. This is because we do not want to change the original image.
* cv2.drawContours(image1,cnts,-1,(0,255,0),3): We are drawing the identified contours on our image. Input the values as they are.
* cv2.imshow("contours",image1): We are displaying the image with the identified contours drawn around it.

##### **Sorting the identified contours**

**cnts = sorted(cnts, key = cv2.contourArea, reverse = True) [:30]**

**screenCnt = None**

**image2 = image.copy()**

**cv2.drawContours(image2,cnts,-1,(0,255,0),3)**

**cv2.imshow("Top 30 contours",image2)**

**cv2.waitKey(0)**

* cnts = sorted(cnts, key = cv2.contourArea, reverse = True) [:30]: We are sorting contours based on the minimum area 30 and ignoring the ones below that.
* screenCnt = None: Stores the number plate contour.
* cv2.drawContours(image2,cnts,-1,(0,255,0),3): Draws the sorted contours on the image.
* cv2.imshow("Top 30 contours",image2): Displays the image which contains the top 30 contours drawn around it.

##### **Finding the contour with four sides**

**i=7**

**for c in cnts:**

**perimeter = cv2.arcLength(c, True)**

**approx = cv2.approxPolyDP(c, 0.018 \* perimeter, True)**

**if len(approx) == 4:**

**screenCnt = approx**

* for c in cnts: : We are creating a for loop over the contours we did sort. This is to find the best contour of our expected number plate.
* perimeter = cv2.arcLength(c, True): Perimeter is also referred to as arclength. We are using the arclength function to find it.
* cv2.approxPolyDP(c, 0.018 \* perimeter, True): ApproxPolyDP approximates the curve of polygon with precision.
* if len(approx) == 4:: chooses the contours with four sides as this will probably be our number plate.

##### **Cropping the rectangular part identified as license plate**

**x,y,w,h = cv2.boundingRect(c)**

**new\_img=image[y:y+h,x:x+w]**

**cv2.imwrite('./'+str(i)+'.png',new\_img)**

**i+=1**

**break**

* x,y,w,h = cv2.boundingRect(c) : This finds the coordinates of the part identified as the license plate.
* cv2.imwrite('./'+str(i)+'.png',new\_img: Stores the new image of the cropped number plate.
* break: breaks the for loop.

##### **Drawing the selected contour on the original image**

**cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 3)**

**cv2.imshow("image with detected license plate", image)**

**cv2.waitKey(0)**

* cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 3): This draws the contour selected to be the number plate on our original image.
* cv2.imshow("image with detected license plate", image): Displaying the final image that has a contour drawn over the number plate.

##### **Extracting text from the image of the cropped license plate**

**Cropped\_loc = './7.png'**

**cv2.imshow("cropped", cv2.imread(Cropped\_loc))**

**plate = pytesseract.image\_to\_string(Cropped\_loc, lang='eng')**

**print("Number plate is:", plate)**

**cv2.waitKey(0)**

**cv2.destroyAllWindows()**

* Cropped\_loc = './7.png': This is the file name of the cropped image of the license plate.
* cv2.imshow("cropped", cv2.imread(Cropped\_loc)): We are displaying the image of the cropped license plate part.
* plate = pytesseract.image\_to\_string(Cropped\_loc, lang='eng'): We are passing the image of the cropped part of the license plate. We are then calling on pytesseract to extract the text on the image.
* print("Number plate is:", plate): We are printing out the extracted text.
* cv2.destroyAllWindows(): We are closing all the open windows.

**Abstract**

The project is designed to detect and capture the license plate of vehicles parked in no parking areas using machine learning algorithms. The captured license plate is then processed and sent to a web application where authorities can access the record and impose a fine or penalty on the car owner. The project also includes a feature to detect fake license plates and capture the car photo for record-keeping.

**Introduction**

The problem of illegal parking in no-parking areas is a common issue faced by most cities. It not only causes inconvenience to other drivers but also poses a threat to pedestrians and causes traffic congestion. This project aims to address this issue by automating the process of identifying parked cars in no-parking areas and issuing fines or penalties to car owners.

**Working Principle**

The project uses a camera to capture images of parked cars. The captured images are processed using machine learning algorithms to detect and extract the license plate. The extracted license plate is then sent to a web application where the authorities can access the record and impose a fine or penalty on the car owner. The project also includes a feature to detect fake license plates by analyzing the characteristics of the license plates.

**Software Requirement**

* Python 3.x
* OpenCV
* Pytesseract
* Requests

**Hardware Requirement**

* Camera

**Methodology**

The project follows the following methodology:

Start the camera and capture images of parked cars.

Process the captured images using OpenCV to detect and extract the license plate.

Send the extracted license plate to a web application where authorities can access the record and impose a fine or penalty on the car owner.

If the license plate is fake, capture the car photo for record-keeping.

Send an SMS to the car owner about the fine penalty.

**Future Scope**

The project can be further improved by adding the following features:

Real-time monitoring of no-parking areas using multiple cameras.

Integration with an online payment gateway for immediate payment of fines.

Integration with GPS to track the location of the car and find the car owner accordingly.

Integration with a database to store the records of fined car owners for future reference.

**Conclusion**

The project successfully addresses the issue of illegal parking in no-parking areas by automating the process of identifying parked cars and issuing fines or penalties to car owners. The project can be further improved by adding more features and integrating it with other technologies.

**Code**

import cv2

import imutils

import requests

import pytesseract

URL = "https://www.techvegan.in/license-detector/script.php"

cameraPort = 0

pytesseract.pytesseract.tesseract\_cmd = 'C:\Program Files\Tesseract-OCR\\tesseract'

key = cv2. waitKey(1)

webcam = cv2.VideoCapture(cameraPort)

while True:

try:

check, frame = webcam.read()

print(check) #prints true as long as the webcam is running

print(frame) #prints matrix values of each framecd

cv2.imshow("Capturing", frame)

key = cv2.waitKey(1)

if key == ord('s'):

cv2.imwrite(filename='detect.jpg', img=frame)

webcam.release()

# img\_new = cv2.imread('detect.jpg', cv2.IMREAD\_GRAYSCALE)

# img\_new = cv2.imshow("Captured Image", img\_new)

# cv2.waitKey(1650)

cv2.destroyAllWindows()

# print("Processing image...")

# img\_ = cv2.imread('detect.jpg', cv2.IMREAD\_ANYCOLOR)

# print("Converting RGB image to grayscale...")

# gray = cv2.cvtColor(img\_, cv2.COLOR\_BGR2GRAY)

# print("Converted RGB image to grayscale...")

# print("Resizing image to 28x28 scale...")

# img\_ = cv2.resize(gray,(28,28))

# print("Resized...")

# img\_resized = cv2.imwrite(filename='detect-grey.png', img=img\_)

# print("Image saved!")

break

elif key == ord('q'):

print("Turning off camera.")

webcam.release()

print("Camera off.")

print("Program ended.")

cv2.destroyAllWindows()

break

except(KeyboardInterrupt):

print("Turning off camera.")

webcam.release()

print("Camera off.")

print("Program ended.")

cv2.destroyAllWindows()

break

image = cv2.imread('detect.jpg')

image = imutils.resize(image, width=300 )

#cv2.imshow("original image", image)

#cv2.waitKey(0)

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

#cv2.imshow("greyed image", gray\_image)

#cv2.waitKey(0)

gray\_image = cv2.bilateralFilter(gray\_image, 11, 17, 17)

#cv2.imshow("smoothened image", gray\_image)

#cv2.waitKey(0)

edged = cv2.Canny(gray\_image, 30, 200)

#cv2.imshow("edged image", edged)

#cv2.waitKey(0)

cnts,new = cv2.findContours(edged.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

image1=image.copy()

cv2.drawContours(image1,cnts,-1,(0,255,0),3)

#cv2.imshow("contours",image1)

cv2.waitKey(0)

cnts = sorted(cnts, key = cv2.contourArea, reverse = True) [:30]

screenCnt = None

image2 = image.copy()

cv2.drawContours(image2,cnts,-1,(0,255,0),3)

#cv2.imshow("Top 30 contours",image2)

cv2.waitKey(0)

i=7

for c in cnts:

perimeter = cv2.arcLength(c, True)

approx = cv2.approxPolyDP(c, 0.018 \* perimeter, True)

if len(approx) == 4:

screenCnt = approx

x,y,w,h = cv2.boundingRect(c)

new\_img=image[y:y+h,x:x+w]

cv2.imwrite('./'+str(i)+'.png',new\_img)

i+=1

break

cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 3)

#cv2.imshow("image with detected license plate", image)

cv2.waitKey(0)

Cropped\_loc = './7.png'

#cv2.imshow("cropped", cv2.imread(Cropped\_loc))

plate = pytesseract.image\_to\_string(Cropped\_loc, lang='eng')

plateClean = plate.replace("\n","")

# HTTPS Request

print("Number plate is:", plate)

PlateImage = {'file': open('detect.jpg', 'rb')}

# r = requests.post(url, files=files)

DATA = {'license':plateClean}

requests.post(url = URL, data = DATA, files=PlateImage)

print("Number plate is:", plate)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Web Application**

**Bootstrap CSS**

Bootstrap is a popular front-end development framework used to build responsive and mobile-first websites. It provides a set of CSS, JavaScript, and HTML templates that help developers to create web pages and applications quickly and easily. Bootstrap was created by Twitter and is now an open-source project with a large community of contributors.

One of the main features of Bootstrap is its CSS framework. Bootstrap CSS is a collection of CSS classes that can be used to style HTML elements. These classes provide a consistent and unified look and feel to the web pages built with Bootstrap. They are designed to be easy to use and can be added to any HTML element without having to write custom CSS.

Bootstrap CSS provides a number of benefits for developers, including:

Responsive Design: Bootstrap CSS is designed to be mobile-first, which means it's optimized for viewing on smaller screens. This ensures that websites built with Bootstrap are responsive and accessible on all devices.

Easy Customization: Bootstrap CSS is highly customizable, allowing developers to create unique designs that suit their needs. It includes a number of customization options, such as color schemes, typography, and layout options.

Consistency: Bootstrap CSS provides a consistent and unified look and feel to websites, making them more professional and polished. This consistency also helps to improve user experience and makes websites easier to use.

Cross-Browser Compatibility: Bootstrap CSS is compatible with all modern browsers, including Chrome, Firefox, Safari, and Internet Explorer.

Accessibility: Bootstrap CSS is designed to be accessible to all users, including those with disabilities. It includes support for screen readers and other assistive technologies, making websites built with Bootstrap more inclusive.

Overall, Bootstrap CSS is a powerful and versatile tool for front-end developers. It provides a solid foundation for building responsive, mobile-first websites with a consistent and professional look and feel.

**HTML**

HTML (Hypertext Markup Language) is a markup language used to create the structure and content of web pages. It defines the structure of a document by using tags to represent elements such as headings, paragraphs, images, and hyperlinks. These tags allow web browsers to interpret and display the content of a web page.

HTML has evolved over the years, with the latest version being HTML5. Some of the key features of HTML5 include improved support for multimedia content, new structural elements such as <article> and <section>, and new form controls such as <input type="date"> and <input type="range">.

HTML is used in conjunction with other web technologies such as CSS (Cascading Style Sheets) and JavaScript to create interactive and dynamic web pages. CSS is used to style the appearance of HTML elements, while JavaScript is used to add interactivity and dynamic functionality to web pages.

HTML can be written using any text editor, but there are also many specialized HTML editors available that provide features such as syntax highlighting, auto-completion, and WYSIWYG (What You See Is What You Get) editing. Once an HTML document has been created, it can be viewed in a web browser to see how it will appear to users.

Overall, HTML is a fundamental technology for the web and is used extensively in web development. It provides a simple yet powerful way to create structured content and is an essential skill for anyone looking to create web pages or web applications.

**PHP**

PHP (Hypertext Preprocessor) is a widely-used server-side scripting language designed for web development. It was initially released in 1995 and is now one of the most popular programming languages on the web, powering millions of websites and web applications. PHP is open-source and free to use, making it an accessible and cost-effective option for developers.

**Here are some key features and characteristics of PHP:**

* Server-side scripting language: PHP is executed on the server side, meaning that it runs on the web server before the HTML is sent to the client's web browser. This allows for dynamic content generation and database interactions.
* Cross-platform compatibility: PHP can run on various operating systems, including Windows, Linux, macOS, and Unix.
* Easy to learn: PHP has a relatively simple and straightforward syntax, making it accessible to beginners. It also has extensive documentation and a large online community that offers support.
* Supports multiple databases: PHP supports a wide range of databases such as MySQL, Oracle, PostgreSQL, and more.
* Object-oriented programming support: PHP has support for object-oriented programming (OOP), allowing developers to create reusable code and improve code organization and maintenance.
* Extensible: PHP has a vast library of extensions and add-ons that can be easily integrated into projects, making it highly extensible.
* Security: PHP has built-in security features such as input validation and output escaping to help prevent attacks such as SQL injection and cross-site scripting (XSS).
* Integration with HTML: PHP can be integrated with HTML, allowing developers to create dynamic and interactive web pages.
* PHP can be used for a wide range of web development tasks, such as creating dynamic web pages, developing e-commerce websites, building content management systems (CMS), creating web-based applications, and more. It can also be used in combination with other technologies such as HTML, CSS, JavaScript, and various web frameworks like Laravel, CodeIgniter, and Symfony.
* Overall, PHP's simplicity, flexibility, and robustness have contributed to its popularity among web developers worldwide.

**MySQLi**

MySQLi stands for MySQL improved extension and is a PHP extension used for accessing MySQL databases. It is an improved version of the older MySQL extension and provides a number of additional features and improvements over its predecessor.

**Some of the key features of MySQLi include:**

* Object-oriented interface: MySQLi provides an object-oriented interface for accessing MySQL databases, which makes it easier to use and more flexible.
* Prepared statements: Prepared statements are a feature of MySQLi that allows you to execute SQL statements with placeholders for input parameters. This can help prevent SQL injection attacks by separating the SQL code from user-supplied input.
* Improved error handling: MySQLi provides improved error handling compared to the older MySQL extension. This includes more detailed error messages and the ability to handle errors more gracefully.
* Support for multiple statements: MySQLi allows you to execute multiple SQL statements with a single function call, which can help improve performance in some situations.
* Support for transactions: MySQLi provides support for transactions, which allow you to group multiple SQL statements into a single unit of work that can be committed or rolled back as a single operation.
* Support for stored procedures: MySQLi allows you to call stored procedures in a MySQL database, which can help improve performance and simplify your code.

To use MySQLi, you first need to establish a connection to a MySQL database using the mysqli\_connect() function. Once you have a connection, you can execute SQL statements using functions like mysqli\_query() and mysqli\_fetch\_array(). You can also use prepared statements to execute SQL statements with placeholders, as well as transactions to group multiple SQL statements into a single unit of work.

Overall, MySQLi is a powerful and flexible extension that provides a wide range of features for working with MySQL databases in PHP.

**APACHE HTTP SERVER**

Apache HTTP Server, commonly referred to as simply Apache, is an open-source web server software that is widely used to serve web content across the internet. It was first released in 1995 and has since become the most popular web server software on the internet, running on nearly 40% of all web servers worldwide.

**Some of the key features of Apache include:**

* Open-source: Apache is open-source software, which means that its source code is freely available to anyone and can be modified and redistributed as needed.
* Cross-platform compatibility: Apache is compatible with most popular operating systems including Windows, Linux, and macOS.
* Modular design: Apache is designed to be highly modular, which means that it can be easily extended and customized with third-party modules.
* Security: Apache has a range of built-in security features that help protect web servers and websites from attacks and unauthorized access.
* Performance: Apache is designed to be highly scalable and efficient, allowing it to handle a large number of concurrent connections and serve web content quickly.

Apache uses a number of configuration files to control its behavior, including httpd.conf, which contains the main configuration settings, and .htaccess, which allows users to override settings on a per-directory basis.

Overall, Apache HTTP Server is a highly reliable, customizable, and secure web server software that is widely used by developers and website administrators around the world.

**Pytesseract**

The tesseract package is designed to compute concentrations of simulated dark matter halos from volume info for particles generated using Voronoi tesselation. This technique is advantageous as it is non-parametric, does not assume spherical symmetry, and allows for the presence of substructure. For a more complete description of this technique including a comparison to other techniques for calculating concentration, please see the accompanying paper Lang et al. (2015).

This package allows users to:

perform Voronoi tessellation through access to qhull routines

measure particle distribution properties like concentration using different techniques including tessellation

replicate the performance test presented in Lang et al. (2015)

esseRACt can be installed from either PyPI or from the source distribution.

Installing from PyPI

The easiest way to install TesseRACt is using pip. If you have administrative privleges on the target machine, this is done using:

$ pip install tesseract

If you do not have admin privleges, simply install it locally using:

$ pip install tesseract --user

The TesseRACt package can then be updated to the most recent stable release using:

$ pip install tesseract --upgrade

Installing from the Source Distribution

The most recent TesseRACt source distribution can be obtained by either downloading or cloning the repository from Bitbucket. Using Mercurial this is done by issuing the following command:

$ hg clone https://[username]@bitbucket.org/[username]/tesseract

where [username] should be replaced with your Bitbucket username. The Bitbucket repository is currently private. If you would like access to this repository, please contact Meagan Lang.

Once you have the TesseRACt source distribution, move into the distribution directory:

$ cd tesseract

and use the standard Distutils command to build and install the distribution:

$ python setup.py install

If you do not have administrative privleges, this can be done using:

$ python setup.py install --user

Testing the Install

To test that everything was installed propertly. From the python prompt, import TesseRACt:

>>> import tesseract

and try to access the documentation:

>>> help(tesseract)

Additional tests can be found in Tutorials.

The First Import

The first time you run import tesseract, a few things will happen. First, a user config file .tessrc will be created in your home directory. This file is used to control different aspects of TesseRACt which are explained in The Config File. Second, you will be prompted to enter a directory in which qhull will be installed. This directory will be added to the user configuration file, which can be changed at any time if you move qhull.

**OPEN CV**

What is Computer Vision?

Computer vision is an approach to understanding how photos and movies are stored, as well as manipulating and extracting information from them. Artificial Intelligence depends on or is mostly based on computer vision. Self-driving cars, robotics, and picture editing apps all rely heavily on computer vision

Human vision has a resemblance to that of computer vision. Human vision learns from the various life experiences and deploys them to distinguish objects and interpret the distance between various objects and estimate the relative position.



Installing and Importing the OpenCV Image Preprocessing Package

OpenCV in deep learning is an extremely important important aspect of many Machine Learning algorithms. OpenCV is an open-source library (package) for computer vision, machine learning, and image processing applications that run on the CPU exclusively. It works with many different programming languages, including Python. It can be imported with single line command as being depicted below

pip install opencv-python

A package in Python is a collection of modules that contain pre-written programmes. These packages allow you to import modules separately or in their whole. Importing the package is as simple as calling the “cv2” module as seen below:

import cv2 as cv

Reading an Input Image

Colour photographs, grayscale photographs, binary photographs, and multispectral photographs are all examples of digital images. In a colour image, each pixel contains its colour information. Binary images have only two colours, usually black and white pixels, and grayscale images have only shades of grey as their only colour. Multispectral pictures gather image data spanning the electromagnetic spectrum within a specific wavelength.

To read the image, we use the “imread” method from the cv2 package, where the first parameter is the image’s path, including filename and extension, and the second parameter is a flag that determines how to read in the image.

By changing the absolute path of the image here, you can test reading it from your local computer or even the internet! If the image is already in your current working directory, you only need to specify the picture name and extension type. Set the second parameter to 0 to read it as a grayscale image, -1 to read it as unmodified (reads the image as alpha or transparency channel if it exists), and 1 to read it as a colour image if you want to read it as a colour image.

The features of a picture that is being utilised as an input

import cv2

# To read image cv2.imread function,

img = cv2.imread("pythonlogo.png", cv2.IMREAD\_COLOR)

# Creating GUI window to display an image on screen

cv2.imshow("Cute Kitens", img)

Image Data Type

To discover the image’s type, use the “dtype” technique. This strategy enables us to comprehend the representation of visual data and the pixel value.

in addition to the image kind, It’s a multidimensional container for things of comparable shape and size.

Pixel values for the image

A collection of small samples can be thought of as an image. These samples are referred to as pixels. To have a better understanding of an image, try zooming in as much as possible. Divided into several squares, the same can be seen. These are pixels, and when all of them are combined, they form an image. One of the simplest methods to represent an image is via a matrix.

Code:

print("The data type of the image is",image.dtype)

Output:

The data type of the image is uint8

uint8 is representing each pixel value being an Unsigned Integer of

8 bits. This data type ranges between 0 to 255

Image Resolution

Image resolution is defined as the number of pixels in an image. As the number of pixels rises, the image quality improves. As we saw before, the image’s shape determines the number of rows and columns. Pixel values in images: 320 x 240 pixels (mostly suitable for small screen devices), 1024 x 768 pixels (appropriate for viewing on standard computer monitors), 720 x 576 pixels (good for viewing on standard definition TV sets with 4:3 aspect ratio), 1280 x 720 pixels (for viewing on widescreen monitors), 1280 x 1024 pixels (for viewing on full-screen monitors) Pixel values in images.

Image Pixel Values

A collection of small samples can be thought of as an image. The unit of measurement for these samples is pixels. For improved comprehension, try zooming in on a picture as much as possible. The same can be divided into several different squares. These are pixels that, when combined, make up an image.

The quality of an image decreases as the number of pixels in the image increases. The image’s shape, which we saw earlier, determines the number of rows and columns.

**Viewing the Images**

Let’s have a look at how to make the image appear in a window. We’ll need to create a graphical user interface (GUI) window to display the image on the screen to do so. The title of the GUI window screen must be the first parameter, and it must be specified in string format. The image can be displayed in a pop-up window using the cv2.imshow() method. However, if you try to close it, you can get stuck with its window. We can use the “waitKey” method to mitigate this.

The “waitKey” parameter has been set to ‘0’ to keep the window open until we close it. (You can specify the time in milliseconds instead of 0, indicating how long it should be open for.)

# To read image from disk, we use

# cv2.imread function, in below method,

img = cv2.imread("python logo.png", cv2.IMREAD\_COLOR)

# Creating GUI window to display an image on screen

# first Parameter is windows title (should be in string format)

# Second Parameter is image array

cv2.imshow("The Logo", img)

# To hold the window on screen, we use cv2.waitKey method,

If 0 pass an parameter, then it will

# hold the screen until user close it.

cv2.waitKey(0)

# for removing/deleting created GUI window from screen

# and memory

cv2.destroyAllWindows()

**OpenCV Applications**

• The concept of OpenCV in Deep Learning is applied for recognition of faces.

• Counting the number of people (foot traffic in a mall, for example)

• Counting the number of automobiles on motorways and their speeds

• Interaction-based art installations

• Anomalies (defects) are detected during the production process (the odd defective products)

• Stitching an image from a street view

• Street view image stitching

• Video/image search and retrieval

• Robot and autonomous car navigation and control

• object recognition

• Medical image analysis

• Movies – 3D structure from motion

**Functionality of OpenCV**

• I/O, processing and display of images and videos

• Detection of objects and features

• Computer vision based on geometry

• Computer-assisted photography

**Summary**

So in this article, we covered the basic Introduction about OpenCV Library and its application in real-time scenarios. We also covered other key terminologies and fields where OpenCV in deep learning is being deployed(Computer Vision) as well as implemented python code for performing some of the basic image operations(dilation, erosion, and changing image colours) with the help of the OpenCV library. Apart from that OpenCV in deep learning would also find application in a variety of industries.

**WebCAM**

****

|  |  |
| --- | --- |
| Brand | Logitech |
| Connector Type | USB |
| Colour | Black |
| Special Feature | Low Light |
| Model Name | C270 |
| Mounting Hardware | Webcam with 6-foot cable and User documentation |

**Technical Details**

|  |  |
| --- | --- |
| Brand | ‎Logitech |
| Manufacturer | ‎Logitech G, ‎Logitech, Logitech Asia Pacific Ltd, Unit no. 1003, Tower 1, Cheung Sha Wan Plaza, Cheung Sha Wan Road, Kowloon, Hong Kong |
| Model | ‎960-000999 |
| Model Name | ‎C270 |
| Product Dimensions | ‎13.6 x 8.1 x 13.6 cm; 130 Grams |
| Item model number | ‎960-000999 |
| Hardware Platform | ‎Laptop, PC, Tablet |
| Special Features | ‎Low Light |
| Mounting Hardware | ‎Webcam with 6-foot cable and User documentation |
| Number of items | ‎1 |
| Display Technology | ‎LCD |
| Has Image Stabilisation | ‎No |
| Digital zoom | ‎5 x |
| Optical Sensor Resolution | ‎3 MP |
| Min Focal Length | ‎30 |
| Microphone format | ‎Built-In |
| Video Capture Resolution | ‎720p |
| Voltage | ‎240 Volts |
| Batteries Included | ‎No |
| Batteries Required | ‎No |
| Connector Type | ‎USB |
| Media type | ‎HDD |
| Is there a timer? | ‎No |
| External memory included | ‎No |
| Manufacturer | ‎Logitech G |
| Country of Origin | ‎China |
| Item Weight | ‎130 g |

**Other Information**

**Python-tesseract:**  
Py-tesseract is an optical character recognition (OCR) tool for python. That is, it’ll recognize and “read” the text embedded in images. Python-tesseract is a wrapper for Google’s Tesseract-OCR Engine. It is also used as an individual script, because it can read all image types like jpeg, png, gif, bmp, tiff, etc. Additionally, if used as a script, Python-tesseract will print the recognized text rather than writing it to a file. It has ability to recognize more than 100 languages.

**Installation:**

pip install pytesseract

**OpenCV:**  
OpenCV is an open source computer vision library. The library has more than 2500 optimized algorithms. These algorithms are often used to search and recognize faces, identify objects, recognize scenery and generate markers to overlay images using augmented reality, etc.

**References**

[*https://pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/*](https://pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/)

[*https://www.geeksforgeeks.org/license-plate-recognition-with-opencv-and-tesseract-ocr/*](https://www.geeksforgeeks.org/license-plate-recognition-with-opencv-and-tesseract-ocr/)

[*https://pytesseract.readthedocs.io/en/latest/*](https://pytesseract.readthedocs.io/en/latest/)

[*https://devdocs.io/*](https://devdocs.io/)

[*https://www.analyticsvidhya.com/blog/2022/02/a-basic-introduction-to-opencv-in-deep-learning/#:~:text=OpenCV%20is%20a%20massive%20open,are%20fundamental%20in%20today's%20systems*](https://www.analyticsvidhya.com/blog/2022/02/a-basic-introduction-to-opencv-in-deep-learning/#:~:text=OpenCV%20is%20a%20massive%20open,are%20fundamental%20in%20today's%20systems)*.*

**START**

**STOP**

SEND SMS TO THE OWNER

UPDATE RECORD IN MySQL DATABASE

SEND LICENSE NO. TO WEB SERVER

CREATE HTTP REQUEST

DETECT LICENSE NO.

CAPTURE LICENSE PLATE

INITIALIZE CAMERA