OpenCV

What is OpenCV library?

OpenCV, short for Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by a community of developers under the OpenCV Foundation.

Introduction

Opencv is a huge open-source library for computer vision, machine learning, and image processing. Now, it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human.

When it is integrated with various libraries, such as **NumPy**, **python** is capable of processing the opencv array structure for analysis. To Identify an image pattern and its various features we use vector space and perform mathematical operations on these features.

Example of OpenCV

OpenCV allows you to perform various operations in the image.

Read the Image : OpenCV helps you to read the image fro file or directly from camera to make it accessible for further processing.

Image Enhacncement : You will be able to enhance image by adjusting the brightness , sharpness or contract of the image. This is helpful to visualize quality of the image.

Object detection: Image objects can also be detected by using OpenCV , Bracelet , watch , patterns, and faces can be detected. This can also include recognizing faces , shapes or even objects.

Image Filtering: You can change the image by applying various filters such as blurring or Sharpening.

Draw the Image: OpenCV allows you to draw text, lines, and any shapes in the images.
Saving the Changed Images: After processing, You can save images that are being modified for future analysis.

Applications of OpenCV

Following are some applications of face recognition

- Face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channel's advertisement recognition

Image-Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.
If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality”.

Digital Image
Image is nothing more than a two-dimensional matrix (3-D in the case of colored images) which is defined by the mathematical function f(x, y) at any point giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what color it should be.
Image processing is basically signal processing in which the input is an image and the output is an image or characteristics according to requirements associated with that image.

How does a computer read an image?

Computers don’t “see” images in the way humans do. Instead, they interpret images as arrays of numerical values. The basic process of how a computer reads and processes an image are:

Pixel Values: An image is made up of pixels, which are the smallest units of information in an image. Each pixel has a value that represents its color and intensity. In the case of an RGB image, there are three values for each pixel corresponding to the Red, Green, and Blue channels.

Digital Representation: The RGB values are usually represented as integers ranging from 0 to 255. 0 represents the absence of color (black), and 255 represents the maximum intensity of that color (full brightness).

Image Matrix: The computer reads the image as a matrix of numbers, where each element in the matrix corresponds to the pixel value at that location. For a color image, there are typically three matrices, one for each RGB channel.

Image Processing: Image processing algorithms are applied to manipulate these numerical representations. Common operations include resizing, cropping, filtering, and more.

Functions of cv2

Read the image

Syntax: **cv2.imread(path, flag)**
Parameters:
path: A string representing the path of the image to be read.
flag: It specifies the way in which image should be read. It's default value is cv2.IMREAD_COLOR

flags -
cv2.IMREAD_COLOR: It specifies to load a color image. Any transparency of image will be neglected. It is the default flag. Alternatively, we can pass integer value 1 for this flag.
cv2.IMREAD_GRAYSCALE: It specifies to load an image in grayscale mode. Alternatively, we can pass integer value 0 for this flag.
cv2.IMREAD_UNCHANGED: It specifies to load an image as such including alpha channel. Alternatively, we can pass integer value -1 for this flag.

Return Value: This method returns an image that is loaded from the specified file.

Display the image

Syntax: **cv2.imshow("Image", image_variable_name)**

Change the color format of image

Syntax = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

Write Image

cv2.imwrite()

method is used to save an image to any storage device. This will save the image according to the specified format in current working directory.

Syntax: cv2.imwrite(filename, image)
Parameters:filename: A string representing the file name. The filename must include image format like .jpg, .png, etc. image: It is the image that is to be saved. Return Value: It returns true if image is saved successfully.

OpenCV Functions Overview

1. Core Functionality
Basic Operations
- `cv2.imread()`: Read an image from file
- `cv2.imwrite()`: Write an image to file
- `cv2.imshow()`: Display an image in a window
- `cv2.waitKey()`: Wait for a key event
- `cv2.destroyAllWindows()`: Close all OpenCV windows

Matrix Operations
- `cv2.add()`, `cv2.subtract()`, `cv2.multiply()`, `cv2.divide()`: Basic arithmetic operations
- `cv2.addWeighted()`: Blend two images
- `cv2.bitwise_and()`, `cv2.bitwise_or()`, `cv2.bitwise_not()`, `cv2.bitwise_xor()`: Bitwise operations

2. Image Processing

Color Space Conversions
- `cv2.cvtColor()`: Convert image from one color space to another (e.g., BGR to HSV)

Image Filtering
- `cv2.blur()`: Apply average blurring
- `cv2.GaussianBlur()`: Apply Gaussian blurring
- `cv2.medianBlur()`: Apply median blurring
- `cv2.bilateralFilter()`: Apply bilateral filtering

Morphological Operations
- `cv2.erode()`: Erode an image
- `cv2.dilate()`: Dilate an image
- `cv2.morphologyEx()`: More advanced morphological transformations

Thresholding
- `cv2.threshold()`: Apply fixed-level thresholding
- `cv2.adaptiveThreshold()`: Apply adaptive thresholding

3. Geometric Transformations

- `cv2.resize()`: Resize an image
- `cv2.rotate()`: Rotate an image
- `cv2.warpAffine()`: Apply affine transformations
- `cv2.warpPerspective()`: Apply perspective transformations

4. Feature Detection and Description

- `cv2.Canny()`: Canny edge detection
- `cv2.cornerHarris()`: Harris corner detection
- `cv2.goodFeaturesToTrack()`: Shi-Tomasi corner detection
- `cv2.SIFT()`: Scale-Invariant Feature Transform
- `cv2.SURF()`: Speeded Up Robust Features

5. Video Analysis

- `cv2.VideoCapture()`: Capture video from a camera or file
- `cv2.VideoWriter()`: Write video to file
- `cv2.calcOpticalFlowPyrLK()`: Calculate optical flow

6. Object Detection

- `cv2.CascadeClassifier()`: Haar Cascade object detection
- `cv2.HOGDescriptor()`: Histogram of Oriented Gradients (HOG) detection

7. Machine Learning

- `cv2.ml.KNearest_create()`: K-Nearest Neighbors
- `cv2.ml.SVM_create()`: Support Vector Machines
- `cv2.ml.ANN_MLP_create()`: Artificial Neural Networks

8. Computational Photography

- `cv2.inpaint()`: Inpainting
- `cv2.denoising()`: Denoising
- `cv2.createTonemapDurand()`: HDR Tonemapping

9. 3D Reconstruction

- `cv2.stereoRectify()`: Stereo rectification
- `cv2.reprojectImageTo3D()`: Reproject image to 3D

10. GPU Acceleration

- Many functions have GPU-accelerated versions prefixed with `cv2.cuda`