Image processing- OpenCV

OpenCV is an open-source computer vision and machine learning software library

OpenCV was started at Intel 1999 by Gary Bradsky and the first release come out in 2000.

OpenCV-Python is the python library for OpenCV, combining the best qualities of the OpenCV C++ API and the python language.

OpenCV-Python is a library of python bindings designed to solve computer vision problems.

Computer vision, the field that allows computers gaining high-level understanding from digital images or videos. CV is everywhere, whether you realize it or not.

OpenCV was built to provide a common infrastructure for computer vision applications.

The goal of computer vision is to emulate human vision using digital images through three main processing components, executed one after the other:

1- Image acquisition
2- Image processing
3- Image analysis and understanding

As our human visual understanding of world is reflected in our ability to make decisions through what we see.

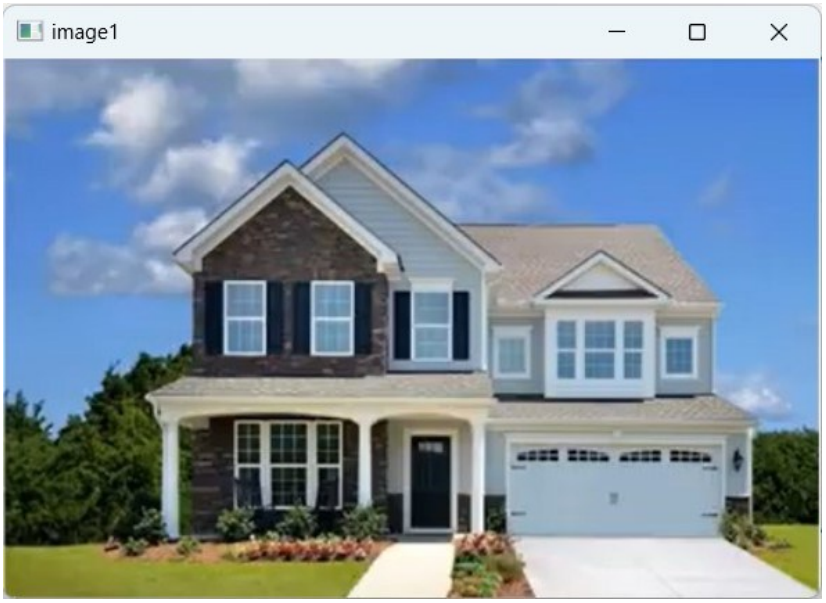
Some of applications:

1- Face recognition.
2- Character recognition.
3- Medical image analysis.
4- Remote sensing.
5- Security.

- **Reading & writing an image**
  - cv.imread(filename,flag) to read an image from a storage.
  - cv.imshow(window name,Image) to display an image in a window.
  - cv.waitkey(delay) is a keyboard binding function. Waits for a pressed key.
  - cv.imwrite(filename,Image) to write and save an image to the storage.
  - cv.destroyAllwindows() to destroy the current running window(GUI).

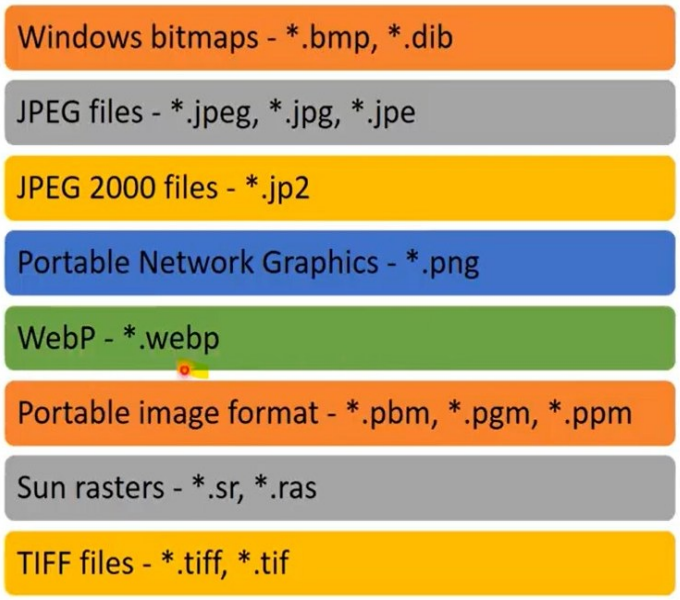
```
import cv2
img=cv2.imread("images/home.jpg") #read an image
#print(img)
cv2.imshow("image1", img) #display an image
cv2.waitKey(0)
cv2.imwrite("images/home.png", img) #save an image
cv2.destroyAllWindows()
```

Output :



- **Supporting file types**

Currently, the following file formats are supported



- **Types of images**

1. binary image: takes only two values black and white (0 and1) and requires 1 bit/pixel
2. Grayscale: contain only brightness information. No color information

   Contain 8 bits/pixel data (0 to 255)

3. Color: three band monochrome image data.

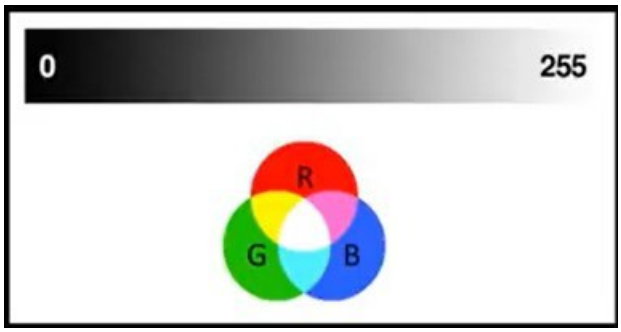
   8 bits for each of the three-color bands (red, green, blue)

4. Multispectral: contain information outside normal human perceptual range.

- **Understanding Images**

Top: grayscale gradient where brighter pixels are closer to 255 and darker pixels are closer to 0.

Bottom: RGB diagram where brighter pixels are closer to the center.



All images consist of pixels in a grid. A 640 * 480 has 640 rows and 480 columns. There are 640*480=307200 pixels.

In OpenCV color images in the RGB (Red, Green, Blue) color space have a 3-tuple associated with each pixel: (B, G, R). Each value in the BGR 3-tuple has a range of [0,255].

#access the RGB pixel located at y=100, x=50 , keeping in mind that

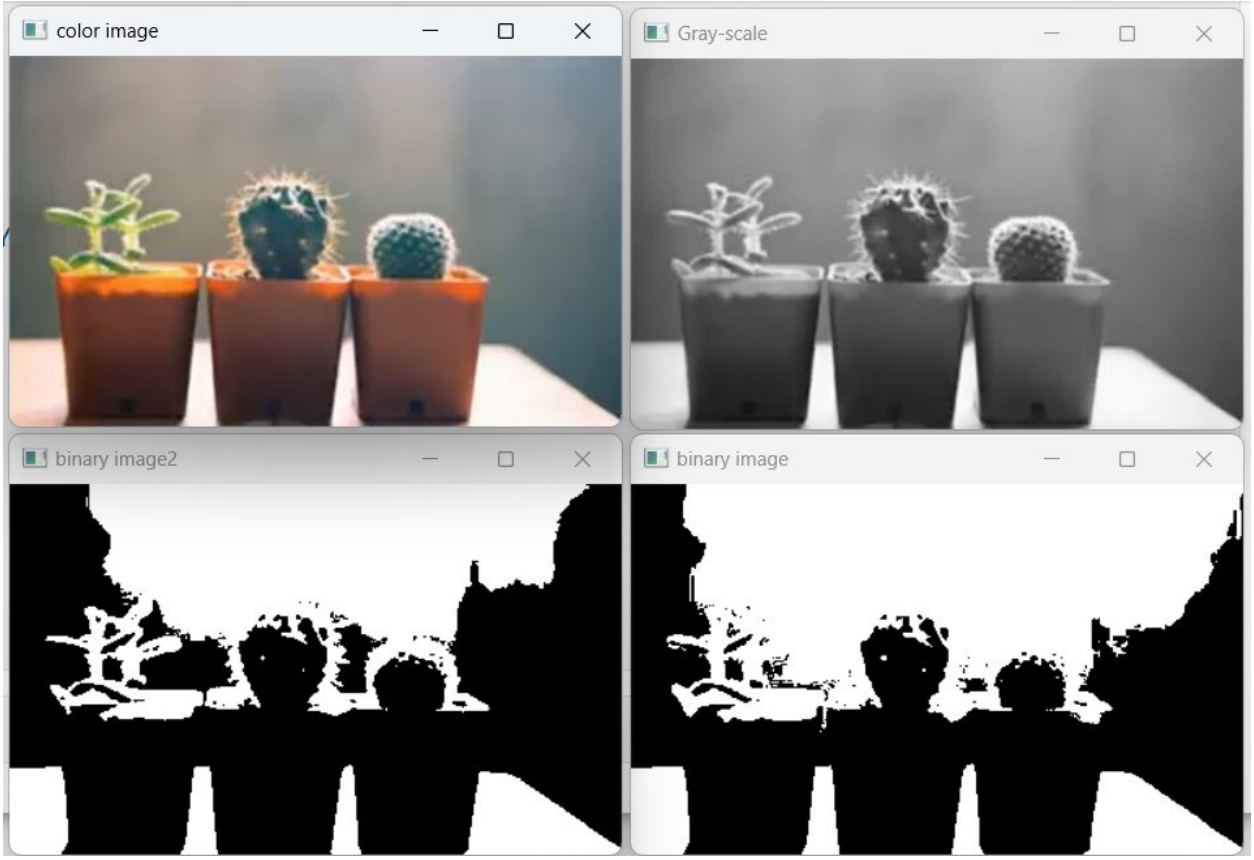
```
import cv2 as c
img=c.imread("images/home.jpg")
px=img[100,100]
print(px)
(B,G,R)=img[100,50]
print("R={} , G={} , B={}".format(R,G,B))
```

```
[208 189 182]
R=182 , G=189 , B=208
```

Example: convert to grayscale and binary

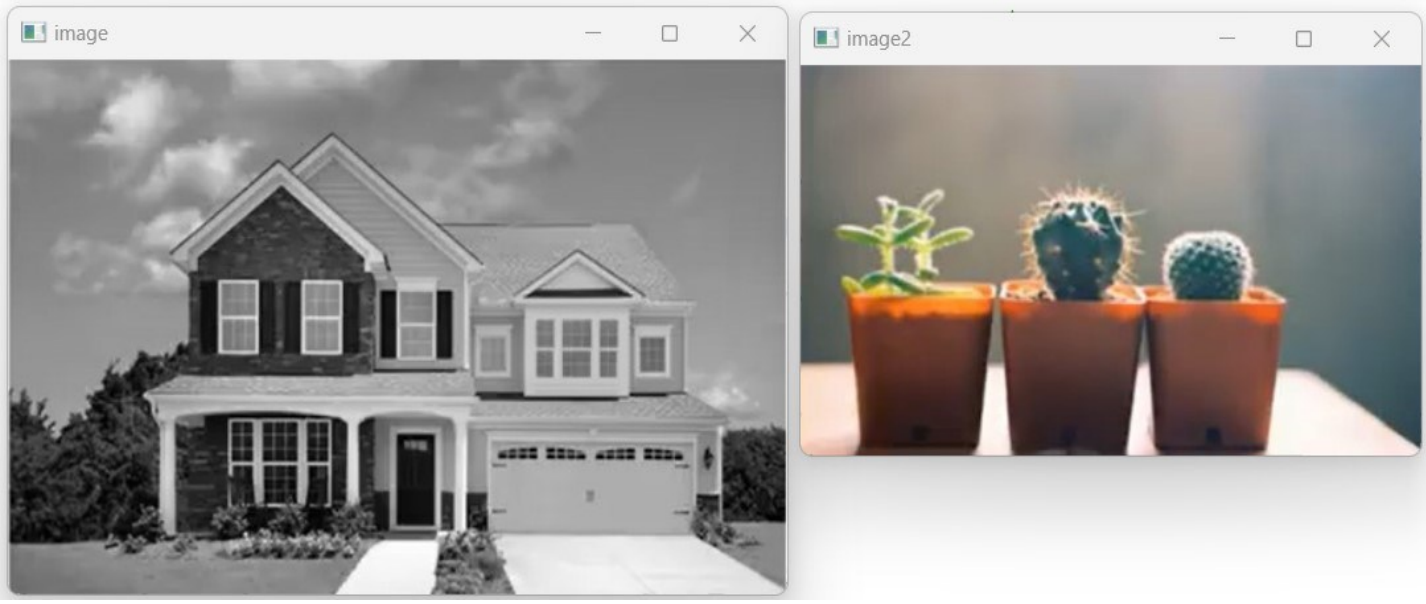
```
import cv2 as c
import numpy as np
img=c.imread("images/trees.jpg")
#convert the image to Gray-scale
grayimg=c.cvtColor(img,c.COLOR_BGR2GRAY)
#convert the image to binary image by using threshold
ret,binaryimg=c.threshold(grayimg,127,255,c.THRESH_BINARY)
#convert the image to binary image by using threshold
bwimg=np.zeros_like(grayimg)
bwimg[grayimg>140]=255 #threshold is 140
c.imshow("color image", img)
c.imshow("Gray-scale", grayimg)
c.imshow("binary image", binaryimg)
c.imshow("binary image2", bwimg)
c.waitKey(0)
c.destroyAllWindows()
```

Output:



- **Image shape and size**

  Img.shape. it returns a tuple of number of rows, columns and channels(if image is color)



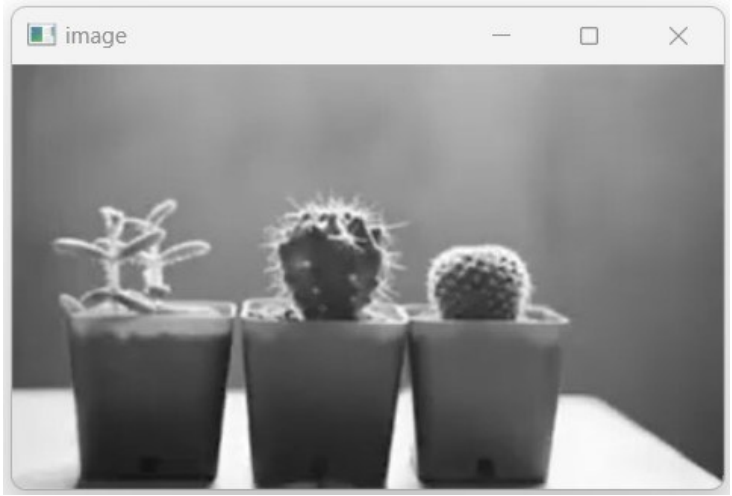
```python
import cv2 as c
img=c.imread("images/home.jpg",0)
print("shape->dimensions: ",img.shape)
print("size->numbers of pixels: ",img.size)
print("Type: ",img.dtype)
c.imshow("image", img)
img2=c.imread("images/trees.jpg")
print('image2')
print("shape->dimensions: ",img2.shape)
print("size->numbers of pixels: ",img2.size)
print("Type: ",img2.dtype)
c.imshow("image2", img2)
c.waitKey(0)
c.destroyAllWindows()
```

Convert to Grayscale Image

```
shape->dimensions:  (311, 472)
size->numbers of pixels:  146792
Type:  uint8
image2
shape->dimensions:  (227, 376, 3)
size->numbers of pixels:  256056
Type:  uint8
```

- **Controlling the keyboard**

Output :----------→



```python
import cv2 as c
import sys
img=c.imread("images/trees.jpg",0)
if img is None:
   print('Failed to read image from file')
   sys.exit(1)
c.imshow("image", img)
# if wait for any key to be pressed to excute next step
k=c.waitKey(0)
if k==27: # wait for ESC key to exit
   c.destroyAllWindows()
elif k==ord('s'): # wait for 's' key to save and exit
     c.imwrite("images/treesgray.png", img)
     c.destroyAllWindows()
```

- **Reading and Writing Video with OpenCV**

```python
import cv2
# Open the input video file
videoCapture = cv2.VideoCapture('1.mp4')
# Check if video opened successfully
if not videoCapture.isOpened():
    print("Error: Could not open video file.")
    exit()
# Get FPS and frame size from input video
fps = videoCapture.get(cv2.CAP_PROP_FPS)
size = (int(videoCapture.get(cv2.CAP_PROP_FRAME_WIDTH)),
        int(videoCapture.get(cv2.CAP_PROP_FRAME_HEIGHT)))
# Create VideoWriter object for output video
videoWriter = cv2.VideoWriter(
    '2.mp4', cv2.VideoWriter_fourcc(*'mp4v'), fps, size)
```

```
# Read and write frames until video ends
success, frame = videoCapture.read()
while success:
    videoWriter.write(frame)       # Write frame to output video
    cv2.imshow('Video Preview', frame)  # Display the frame
    # Wait for 1 ms and check if 'q' key pressed to quit early
    if cv2.waitKey(1)  == ord('q'):
        break
    success, frame = videoCapture.read()
# Release resources and close windows
videoCapture.release()
videoWriter.release()
cv2.destroyAllWindows()
print("Video copying completed successfully.")
```

- **Display Live Camera and Exit on Click or Key Press**

```
import cv2
# Global variable to track mouse click
clicked = False
# Mouse callback function to detect left mouse button release
def onMouse(event, x, y, flags, param):
    global clicked
    if event == cv2.EVENT_LBUTTONUP:
        clicked = True  # Set flag when user clicks
# Open default camera (index 0)
cameraCapture = cv2.VideoCapture(0)
# Create a window to display the video
cv2.namedWindow('MyWindow')
# Set the mouse callback function for the window
cv2.setMouseCallback('MyWindow', onMouse)
print('Showing camera feed. Click window or press any key to stop.')
# Read the first frame from the camera
success, frame = cameraCapture.read()
# Loop to display frames until a key is pressed or the mouse is clicked
while success and cv2.waitKey(1) == -1 and not clicked:
    cv2.imshow('MyWindow', frame)           # Show the current frame
    success, frame = cameraCapture.read()   # Read the next frame
cv2.destroyWindow('MyWindow')
```