opencv-demo

January 8, 2026

1 OpenCV - Open Computer Vision Library

Handling Images - PIL, OpenCV, Matplot

Basics of Image Formation and Image Formats Reading, Display and Writing Images in OpenCV Resizing, cropping, annotating, creating a Region of Interest

[ ]:
```
!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.12/dist-
packages (4.12.0.88)
Requirement already satisfied: numpy<2.3.0,>=2 in
/usr/local/lib/python3.12/dist-packages (from opencv-python) (2.0.2)
```

[ ]:
```
import cv2 as cv
image = cv.imread('large.png',cv.COLOR_BGR2GRAY)
```

[ ]:
```
image.size
```

[ ]:
```
486720
```

[ ]:
```
image
```

[ ]:
```
array([[[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],
```

```
        [[255, 255, 255],
         [255, 255, 255],
         [255, 255, 255],
         …,
         [255, 255, 255],
         [255, 255, 255],
         [255, 255, 255]],

        …,

        [[255, 255, 255],
         [255, 255, 255],
         [255, 255, 255],
         …,
         [255, 255, 255],
         [255, 255, 255],
         [255, 255, 255]],

        [[255, 255, 255],
         [255, 255, 255],
         [255, 255, 255],
         …,
         [255, 255, 255],
         [255, 255, 255],
         [255, 255, 255]],

        [[255, 255, 255],
         [255, 255, 255],
         [255, 255, 255],
         …,
         [255, 255, 255],
         [255, 255, 255],
         [255, 255, 255]]], dtype=uint8)
```

[ ]:
```
image2 = cv.imread('/content/pexels-magda-ehlers-pexels-1279813.jpg')
image2
```

[ ]:
```
array([[[  3, 179,  65],
        [  2, 178,  64],
        [  0, 175,  61],
        …,
        [139,  97,   0],
        [139,  97,   0],
        [145, 103,   2]],

       [[  1, 177,  63],
        [  0, 176,  62],
```

```
        [  0, 174,  60],
        …,
        [142, 101,   0],
        [141, 101,   0],
        [144, 103,   0]],

       [[  0, 175,  61],
        [  0, 174,  60],
        [  0, 172,  58],
        …,
        [146, 107,   0],
        [146, 108,   0],
        [146, 107,   0]],

       …,

       [[  0, 182,  64],
        [  0, 181,  63],
        [  0, 180,  62],
        …,
        [147, 105,   0],
        [145, 103,   0],
        [144, 102,   0]],

       [[  0, 184,  66],
        [  0, 184,  66],
        [  0, 183,  65],
        …,
        [151, 106,   1],
        [150, 105,   0],
        [147, 102,   0]],

       [[  0, 184,  66],
        [  0, 184,  66],
        [  0, 183,  65],
        …,
        [151, 106,   1],
        [150, 105,   0],
        [147, 102,   0]]], dtype=uint8)
```

[ ]:
```
gray = cv.cvtColor(image,cv.COLOR_BGR2GRAY)
print(gray)
```

```
[[255 255 255 … 255 255 255]
 [255 255 255 … 255 255 255]
 [255 255 255 … 255 255 255]
 …
```

```
[255 255 255 … 255 255 255]
[255 255 255 … 255 255 255]
[255 255 255 … 255 255 255]]
```

[ ]:
```
resize_image = cv.resize(image,(64,64))
resize_image
```

[ ]:
```
array([[[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       …,

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
```

```
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]], dtype=uint8)
```

[ ]:
```
cropped_image = image[50:200,200:400]
cropped_image
```

[ ]:
```
array([[[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [  0,   0,   0],
        [ 17,  17,  17],
        [153, 153, 153]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       …,

       [[  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],
```

```
        [[  0,   0,   0],
         [  0,   0,   0],
         [  0,   0,   0],
         …,
         [255, 255, 255],
         [255, 255, 255],
         [255, 255, 255]],

        [[  0,   0,   0],
         [  0,   0,   0],
         [  0,   0,   0],
         …,
         [255, 255, 255],
         [255, 255, 255],
         [255, 255, 255]]], dtype=uint8)
```

[ ]:
```
X, Y, W, H = 150,200,100,50
 cv.rectangle(image,(X,Y),(X+W,Y+H),(0,255,0),2)
```

[ ]:
```
array([[[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       …,

       [[255, 255, 255],
        [255, 255, 255],
```

```
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        …,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]], dtype=uint8)
```

[ ]:
```
# cv.imshow('image',image)
# cv.waitKey(0)
```

[ ]:
```
import cv2

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    cv2.imshow("Webcam", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

2 color spaces

RGB BGR Gray Scale Binary

1. HSV

Hue: The pure color (0-360° angle on the color wheel: Red, Yellow, Green, etc.).

Saturation: The purity/intensity of the hue (0% gray, 100% pure color).

Value (Brightness): How light or dark the color is (0% black, 100% brightest).

2. HSL
3. CMYK
4. LAB

[ ]:
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
```

Image thresholding, Bitwise operations and Masking Creating Digital signatures using Alpha Blending Color space conversion and different Color Spaces

Filtering (Noise Removal)

1. Gaussian Blur
2. Median Blur (Best for Salt & Pepper)
3. Bilateral (Edge preserving)

[ ]:
```
gaussian = cv2.GaussianBlur(img, (5,5), 0)
```

[ ]:
```
median = cv2.medianBlur(img, 5)
```

[ ]:
```
bilateral = cv2.bilateralFilter(img, 9, 75, 75)
```

Edge Detection

Sobel

Canny

[ ]:
```
sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0)
sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1)
```

[ ]:
```
edges = cv2.Canny(gray, 100, 200)
```

Morphological Operations

Dilation

Erosion

Opening

Closing

[ ]:
```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5,5))
```

```
erosion = cv2.erode(gray, kernel, iterations=1)
dilation = cv2.dilate(gray, kernel, iterations=1)
opening = cv2.morphologyEx(gray, cv2.MORPH_OPEN, kernel)
closing = cv2.morphologyEx(gray, cv2.MORPH_CLOSE, kernel)
```

Thresholding

Global Threshold

Adaptive Threshold

Otsu

Contours

[ ]:
```
contours, _ = cv2.findContours(thresh,
                               cv2.RETR_EXTERNAL,
                               cv2.CHAIN_APPROX_SIMPLE)

for cnt in contours:
    area = cv2.contourArea(cnt)
    if area > 500:
        x,y,w,h = cv2.boundingRect(cnt)
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

Feature Detection (ORB)

[ ]:
```
orb = cv2.ORB_create()
kp, des = orb.detectAndCompute(gray, None)

img_kp = cv2.drawKeypoints(img, kp, None, color=(0,255,0))
```

SEGMENTATION

Watershed Algorithm

GrabCut

Mask-Based Segmentation (HSV)

[ ]:
```

```

Image Histograms and enhancement using Histogram Equalization

[ ]:
```
import cv2
import matplotlib.pyplot as plt

# Read image
img = cv2.imread("image.jpg", 0)  # grayscale

# Calculate histogram
hist = cv2.calcHist([img], [0], None, [256], [0, 256])
```

```
# Plot histogram
plt.figure()
plt.title("Grayscale Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.plot(hist)
plt.xlim([0, 256])
plt.show()
```

[ ]:
```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread("image.jpg")
colors = ('b', 'g', 'r')

plt.figure()
plt.title("Color Histogram")

for i, col in enumerate(colors):
    hist = cv2.calcHist([img], [i], None, [256], [0,256])
    plt.plot(hist, color=col)

plt.xlim([0,256])
plt.show()
```

[ ]:
```
equalized = cv2.equalizeHist(img)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.imshow(img, cmap='gray')
plt.title("Original")

plt.subplot(1,2,2)
plt.imshow(equalized, cmap='gray')
plt.title("Equalized")

plt.show()
```

Reading and Writing videos using OpenCV Motion Detection using Background Subtraction Build an Intruder Detection System

[ ]:
```

```

[ ]:
```

```

Image Anotator Tool - LabelIMG - pip install labelImg

VGG Anotator -