# Contents



# Installation of OpenCV

- Python

```
$pip install opencv-python
```

# Read and Save an image

### Mat Variable in CPP

Mat: It is a matrix variable( n-dimensional array) created by OpenCV to handle the images.

Eg: Mat img; //if we include OpenCV CPP library

cv::Mat

### How to Read an image and video in OpenCV

- **Read the image in the original color format**

**Syntax** : Mat =cv::imread("path of the image")

Eg: Mat img= cv::imread("noise1.jpg");  //If image is present in current directory

1. If the Image is present in the mentioned path, it returns the Mat Structure with an image.

- **Read the image in the different color formats**

**Syntax** : Mat =cv::imread("path of the image",color format)

The color formats will be IMREAD_GRAYSCALE, IMREAD_UNCHANGED, IMREAD_COLOR( with 3-channel color format)

Mat img= cv::imread("noise1.jpg",IMREAD_GRAYSCALE);

Mat img= cv::imread("noise1.jpg",IMREAD_UNCHANGED); // Returns original image

- **Capture the Video from the file**

```
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
 int main()
{
        //read an image//
        //Mat img= cv::imread("noise1.jpg",IMREAD_GRAYSCALE);//grayscale formt
       //  Mat img= cv::imread("noise1.jpg",IMREAD_UNCHANGED);//orginal format
        cv::VideoCapture cap("walking.mp4");// capture the file from file
      if(cap.isOpened()!=true)
          printf("unable to open the video file\n");
        while(true)
          {
                  Mat img;
                  cap.read(img); //read frame by frame from cap
        imshow("output",img);//display the each frame with the gap of 20ms
        waitKey(20);//Wait for 20ms fro next frame
          }
 }
```

- **Capture the video from the Camera/Webcam**

**Syntax:** cv::VideoCapture cap(VideoIndex);// capture the file from file

```
cv::VideoCapture cap(0);// capture the file from file
```

## Basic Operations on Image

- **Color Conversion**

**Syntax:** cvtColor(in_image,Output_img,COLOR_FORMAT);

```
cvtColor(img,img,COLOR_BGR2GRAY); //Converts BGR color format to Gray color format
```

- **Blur an image**

To blur an image we mostly use a gaussian blur filter, In CPP we have one function to blur an image as follows

**Syntax:** GaussianBlur(input_img,Out_img,Kernel_size,sigmax, sigmay);

Where sigmax will be the standard deviation in the X-direction, and Sigma will be the Standard deviation in Y-direction.

```
GaussianBlur(reimg,Blurimag,Size(7,7),3,0);
```

Note: As Kernel size increases, Blurness also increases.

- **Edge Detection**

To detect the edges on the image, the pre-requirement is having a blur on the image.

**Syntax:** Canny(input image,output image,min threshold value,max_threshold value)

```
Canny(Blurimag,edge_det_img,100,200);
```

Any edges with an intensity gradient more than maxVal are sure to edge and those below the min threshold value are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified as edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of the edges. Otherwise, they are also discarded. So As increasing the maximum values, the number of edges in the output image will be reduced.

- **Dilation and Erosion on edges**

Dilation means increasing the thickness of the edge, the thickness depends on the given kernel size, the more kernel size, the thickness of the edge will be more. In Contrast way, erosion means reducing the thickness of the edges.

```
Mat kenel_5=getStructuringElement(MORPH_RECT,Size(5,5));
Mat kenel_3=getStructuringElement(MORPH_RECT,Size(3,3));
dilate(edge_det_img,Dil_img5,kernel_5);
imshow("Dilated image_kerenl5*5",Dil_img5);
dilate(edge_det_img,Dil_img3,kernel_3);
imshow("Dilated image_kerenl3*3",Dil_img3);
erode(Dil_img3,erode_img,kernel_3);
imshow("Eroded image_kernel3*3",erode_img);
waitKey(0);
```

- **Resize**

Reduce the Number of pixels in an image. So that the memory requirement to store the image is less.

Syntax:  resize(input_img,Out_img,size(width, height));

```
Mat img=imread("Tree.jpg");
Mat reimg,edge_det_img,Dil_img5,Dil_img3,erode_img;
cout<<img.size()<<endl;  //[4944 x 3263]
resize(img,reimg,Size(640,480));
imshow("OriginalImage",img);
imshow("ResizedImage",reimg);
waitKey(0);
```

- **Scaling**

Scaling is also a similar kind of resize, but we will give a scaling factor instead of giving width and height parameters in the Size argument.

Syntax: resize(input_img,out_img,Size(),scalar1, scalar2);

```
resize(img,reimg,Size(),0.2,0.1);//scalling
```

- **Crop an Image**

To Crop an image in OpenCV we have the ROI( Region Of Intrest) function. It takes the different arguments like starting index of the row and column the required cropped image, and the required size in the cropped image.

Syntax:  Rect ROI(Start_index_row,Start_index_col, width,height)

```
Rect ROI(100,100,340,250);
imgcrop=reimg(ROI);
imshow("Croped image",imgcrop);
```

Colour Formats

Scalar(B, G, R)

Scalar(255,0,0)-Blue colour

Scalar(0,255,0)= Green colour

Scalar(0,0,255)= Red colour

- Drawing Shapes and Text on Images
- **Shapes**
- **Circle**

Step1: Create a blank image

Mat img(width,height,colour_format,Colour type)

Step2: use the circle function to draw a circle on the image

circle(Inpt_image,Central_piont,Radius,Color_format,Thickness);

It Thickness is -1 / FILLED, which means the Circle will be filled with the given color.

```
Mat img(512,512,CV_8UC3,Scalar(255,255,255);
circle(img,Point(256,256),155,Scalar(0,0,0),3);
circle(img,Point(256,256),155,Scalar(0,0,0),-1);//Circle Fill with black color
```

- **Rectangle**

**Syntax:** rectangle(input_img,Top_left_corner_index_point,Bottom_right_corner_index_point,color_type,Thickness);

```
rectangle(img2,Point(130,226),Point(382,286),Scalar(0,70,255),4);
```

- **Put the Text On the Image**

Syntax: putText(inpt_img,String/Text, Starting_index_point, FONT_FORMAT,Color_type,Thickness);

- **Draw a line on the Image**

Syntax: line(inpt_img,Start_point,End_point,Color_type,Thickness);

```
putText(img2,"DEMO",Point(170,260),cv::FONT_HERSHEY_PLAIN,2,Scalar(0,69,255),2);
line(img2,Point(170,270),Point(270,270),Scalar(0,70,255),2);
 imshow("Textimage",img2);
```

## Warp an Image

Wrapping an image means, Select the portion of the image and copied to another image. The working of the warp function can be observed in the following picture.

https://i0.wp.com/theailearner.com/wp-content/uploads/2020/11/perspective2-1.jpg?resize=625%2C395&ssl=1

The example code for warping an image is mentioned below

```
#include<iostream>
#include<opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main()
{
        Mat card_img,Warp_img;
        int width=520, height=620;
        card_img=imread("Warp_cards.jpg");
        cout<<card_img.size()<<endl;
        Point2f src[4]={ {87,368}, {298,343}, {99,693}, {354,649}};
        Point2f dest[4]={ {0,0},{width,0}, {0,height}, {width,height} };
        for(int i=0;i<4;i++)
        {
           circle(card_img,src[i],10,Scalar(255,0,0),-1);
        }
```

```
        imshow("cards",card_img);
        Mat matrix=getPerspectiveTransform(src,dest);
        warpPerspective(card_img,Warp_img,matrix,Point(width,height));
        imshow("Warped image",Warp_img);
        waitKey(0);
}
```

Note down the coordinate values of ROI, And pass them to the getPerspectiveTransform function to get the mapped matrix( Mat) and use the warpPerspective function.

## Face Detection Using OpenCV

There are open-source trained models to detect the models. One of them is the Haar cascade classifier. Before going to detect the face in the image, there are a few requirements needed.

Step 1: Read an image to detect the face on it.

Step 2: Download the Haar cascade-trained model from GitHub by using the following link.

https://github.com/opencv/opencv/tree/master/data/haarcascades

Step 3: Declare the CascadeClasifier variable and load the trained model.

```
CascadeClassifier faceCascade;
faceCascade.load("haarcascade_frontalface_default.xml");
```

Step 4: Declare the vector for the rectangle

Step 5: Using the Cascade Classifier- detectuiltiScale function find out the face in an image.

Step 6: Draw the rectangle on the image.

```
#include<iostream>
#include<opencv2/opencv.hpp>
#include<opencv2/objdetect.hpp>
#include<opencv2/imgproc.hpp>
using namespace std;
using namespace cv;
int main()

{
    Mat img,reimg;
    //Step 1: Read an image to detect the face on it.
    img=imread("Human.jpg");
    resize(img,reimg,Size(640,480));
    //Step 3: Load the Harcascade classifier Model
    CascadeClassifier faceCascade;
    faceCascade.load("haarcascade_frontalface_default.xml");
```

```
    if(faceCascade.empty())
    {
          cout << "XML file not loaded" << endl;
    }
//Step 4: Declare the vector for the rectangle
     vector<Rect>face;

     faceCascade.detectMultiScale(reimg,face,1.1,10);
//Step 5: Draw the rectangle on detected face.
     for(int i=0;i<face.size();i++)
     {
       rectangle(reimg,face[i].tl(),face[i].br(),Scalar(255,0,255),3);

     }
      imshow("Image",reimg);
        waitKey(0);
}
```