OpenCV

Module – 1

Reading and rescaling Images and Videos

# Import OpenCV

*import cv2 as cv*

# Reading an Image

*cv.imread(‘path to the image’)*

you can store it in a variable like this:

*img = cv.imread(‘path to the image’)*

you can display this image by:

*cv.imshow(“Name of the window(You can name whatever you want)”, variable containing the image)*

**“waitKey() function**of Python OpenCV allows users to display a window for given milliseconds or until any key is pressed. It takes time in milliseconds as a parameter and waits for the given time to destroy the window, if 0 is passed in the argument it waits till any key is pressed.”  Geeksforgeeks

# Reading a Video

Reading videos is a little different than reading images. The first thing you need to know is that videos are captured and displayed frame by frame. Think of it like multiple pictures running in a reel.

First create a pointer, you can name it whatever you want:

*capture = cv.VideoCapture(integer or path to the video)*

When to use integer:

“We use the VideoCapture() class of OpenCV to create the camera object which is used to capture photos or videos. The constructor of this class takes an integer argument which denotes the **hardware camera index**. Suppose the devices has two cameras, then we can capture photos from first camera by using VideoCapture(0) and if we want to use second camera, we should use VideoCapture(1).” Geeksforgeeks

Since the video is actually a series of images, we read it in a loop:

*while True:*

*isTrue, frame = capture.read() # A Boolean statement that stores frames of a video in a variable*

*cv.imshow(“Name of the window”, frame)*

*if cv.waitkey(20) & 0xFF == ord(‘d’): # Terminates when a key ‘(d’ in this case) is pressed*

*break # the loop is broken and the video stops*

*capture.release() # Release the ptr*

*cv.destroyAllWindows() # Destroy all windows*

# Resizing an image

Firstly, read the image:

*Img = cv.imread(‘path to the image’)*

*cv.imshow(“Original Image”, img)*

Then write the **rescale function** as:

*def rescale\_frame(frame, scale = 0.75): # the default scale value is 0.75,*

*width = int(frame.shape[1] \* scale) # width is at 1st index of the shape tuple*

*height = int(frame.shape[0] \* scale) # height is at 0th index*

*dimensions = (width,height) # create a tuple of dimensions*

*return cv.resize(frame, dimensions, interpolation = cv.INTER\_AREA) # return resized img*

*resized\_img = rescale\_frame(img)*

*cv.imshow(“Resized Image”, resized\_img)*

*cv.waitKey(0)*

“**Choice of Interpolation Method for Resizing:**

* cv2.INTER\_AREA: This is used when we need to shrink an image.
* cv2.INTER\_CUBIC: This is slow but more efficient.
* cv2.INTER\_LINEAR: This is primarily used when zooming is required. This is the default interpolation technique in OpenCV.” Geeksforgeeks

## def rescale\_frame():

It takes a frame(image) and scales it with the scale value. The default value of scale parameter is 0.75, but you change it. The function resizes the frame according to the new dimensions obtained by multiplying the width and height of the frame with the scale value. The code converts the results into integer values, this is for obtaining non-floating values as dimensions of the image. It returns the cv.resize() function which takes in arguments frame, dimensions, and interpolation. This function works for images, existing videos and live videos.

# Resizing a video

Plug this line inside the loop after the frame is read:

*resized\_frame = rescale\_frame(frame)*

*cv.imshow(‘Resized Video’, resized\_frame)*

The rest of the code remains the same as video’s reading code.

Note: rescale\_frame() works for both videos and images.