**Machine Vision System Practical 4**

**Programming for Image Enhancement & Image Compression**

**in OpenCV using Python**

In this Practical, we will learn

1. How to Enhance an Image i.e. Blurring(Noise reduction) and Sharpening.

&

1. How to Compress an Image in 3D plane i.e. Perspective transformation.

**1] Image Enhancement:**

* **Importing OpenCV & Numpy libraries:**

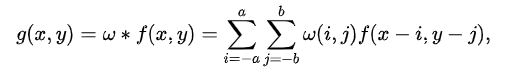
The code starts by importing the OpenCV library i.e. cv2 as cv, which is used for computer vision tasks, including reading and processing images. We also import numpy library as np for matrix calculations for transformation.

* **Reading the Original Image:**

Here we read our image with the help of cv.imread() function by giving it the proper path where the image is stored. And then we display the image with any name suitable using cv.imshow() function.

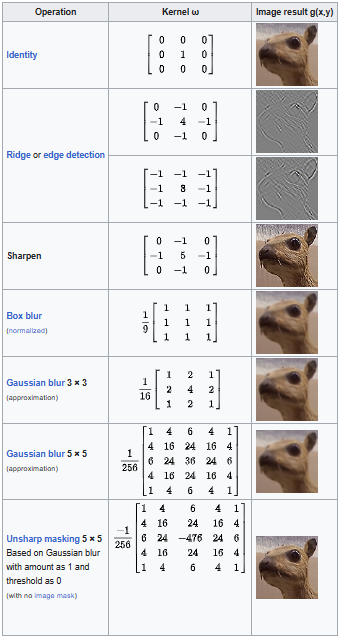
* **Image Enhancement:**

In [image processing](https://en.wikipedia.org/wiki/Image_processing), a kernel, convolution matrix, or mask is a small [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) used for blurring, sharpening, embossing, [edge detection](https://en.wikipedia.org/wiki/Edge_detection), and more. This is accomplished by doing a [convolution](https://en.wikipedia.org/wiki/Convolution) between the kernel and an [image](https://en.wikipedia.org/wiki/Bitmap_image). Or more simply, when each pixel in the output image is a function of the nearby pixels (including itself) in the input image, the kernel is that function.

The general expression of a convolution is





****

* **Blurring:**

**Code:**

import cv2 as cv

import numpy as np

#reading a image from computer and taking dimensions

img = cv.imread('Photos/noisy.jpg')

rows, cols = img.shape[:2]

#gaussian Blur

output\_gaus = cv.GaussianBlur(img, (5,5), 0)

#median Bur (reduction of noise)

output\_med = cv.medianBlur(img, 5)

#Bilateral filtering (Reduction of noise + Preserving of edges)

output\_bil = cv.bilateralFilter(img, 5, 6, 6)

cv.imshow('Gaussian', output\_gaus)

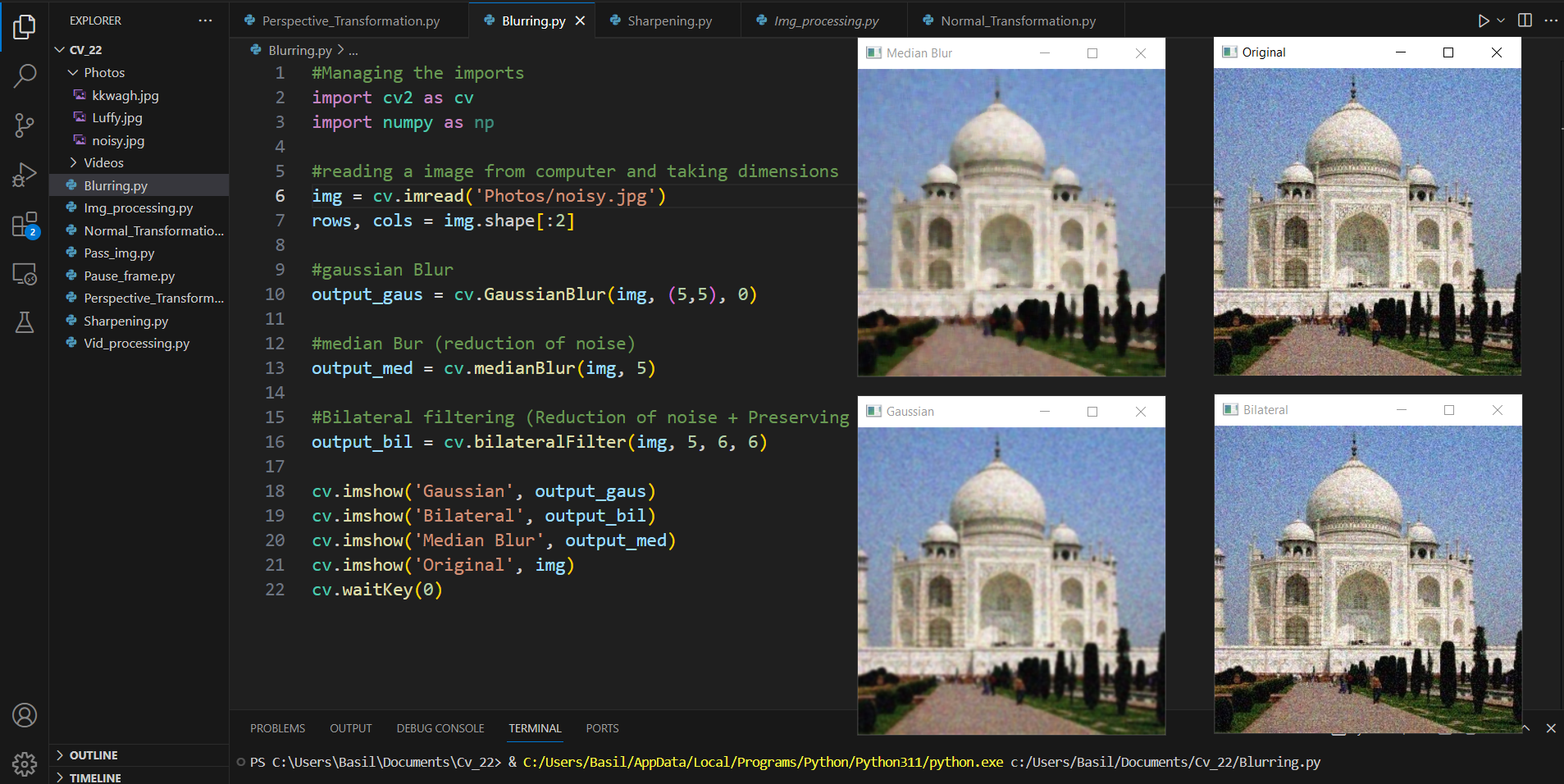
cv.imshow('Bilateral', output\_bil)

cv.imshow('Median Blur', output\_med)

cv.imshow('Original', img)

cv.waitKey(0)

**Output:**

****

* **Sharpening:**

**Code:**

import cv2 as cv

import numpy as np

#Reading the image

img = cv.imread('Photos/kkwagh.jpg')

#Gauusian kernel for sharpening

gaussian\_blur = cv.GaussianBlur(img, (7,7), 2)

#Sharpening using addweighted()

sharpened1 = cv.addWeighted(img,1.5, gaussian\_blur, -0.5, 0)

sharpened2 = cv.addWeighted(img,3.5, gaussian\_blur, -2.5, 0)

sharpened3 = cv.addWeighted(img,7.5, gaussian\_blur, -6.5, 0)

#Showing the sharpened Images

cv.imshow('Sharpened 3', sharpened3)

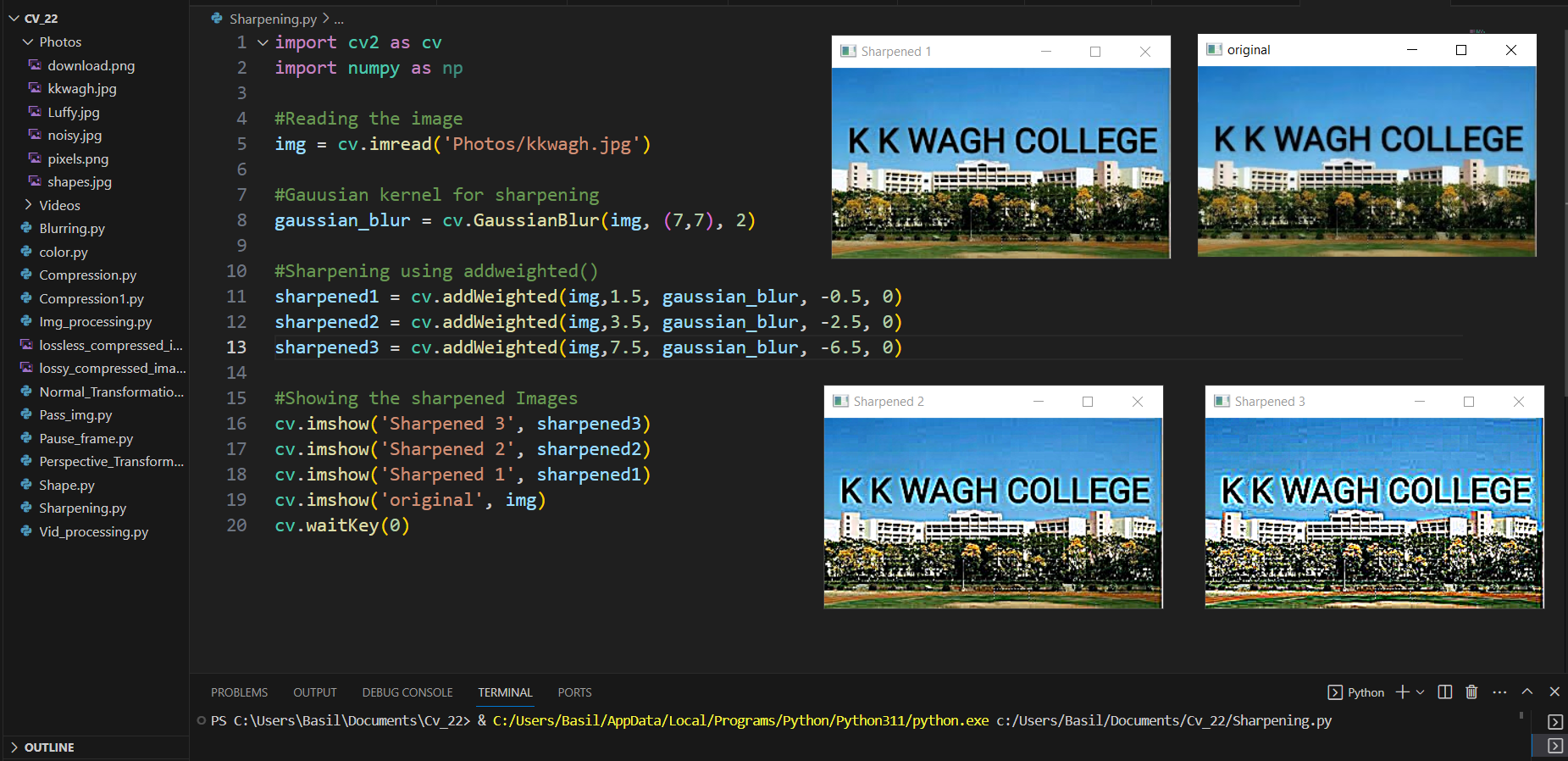
cv.imshow('Sharpened 2', sharpened2)

cv.imshow('Sharpened 1', sharpened1)

cv.imshow('original', img)

cv.waitKey(0)

**Output:**

****

**2] Image Compression:**

* **Importing OpenCV & Numpy libraries:**

The code starts by importing the OpenCV library i.e. cv2 as cv, which is used for computer vision tasks, including reading and processing images. We also import numpy library as np for matrix calculations for transformation.

* **Reading the Original Image:**

Here we read our image with the help of cv.imread() function by giving it the proper path where the image is stored. And then we display the image with any name suitable using cv.imshow() function.

* **Image Compression:**

Image compression is the process of reducing the size of an image file without significantly affecting its visual quality. This is achieved by removing redundant data or approximating parts of the image, making it more storage and transmission-friendly.

There are two main types of image compression: [lossless compression](https://en.wikipedia.org/wiki/Lossless_compression) and [lossy compression](https://en.wikipedia.org/wiki/Lossy_compression). Lossless compression retains the original image data, while lossy compression discards some information to achieve a smaller file size, potentially reducing the image quality.

Image compression is essential for many reasons:

* **Reduced storage space**: Compressed images occupy less storage space, allowing you to store more images or use the saved space for other purposes.
* **Faster transmission**: Smaller file sizes enable faster transmission of images over networks, which is particularly important in applications like video streaming and remote sensing.
* **Reduced bandwidth usage**: Compressed images consume less bandwidth, which is crucial for data transfer on limited or metered connections.

**Code:**

import cv2 as cv

import os

img = cv.imread('Photos/pixels.png')

cv.imwrite('lossless\_compressed\_image.png', img)

# A value between 0 and 100 (higher means better quality, but larger file size)

jpeg\_quality = 90

cv.imwrite('lossy\_compressed\_image.jpg', img, [cv.IMWRITE\_JPEG\_QUALITY, jpeg\_quality])

original\_size = os.path.getsize('Photos/pixels.png')

lossless\_size = os.path.getsize('lossless\_compressed\_image.png')

lossy\_size = os.path.getsize('lossy\_compressed\_image.jpg')

print(f'Original image size: {original\_size} bytes')

print(f'Lossless compressed image size: {lossless\_size} bytes')

print(f'Lossy compressed image size: {lossy\_size} bytes')

lossless\_img = cv.imread('lossless\_compressed\_image.png')

lossy\_img = cv.imread('lossy\_compressed\_image.jpg')

cv.imshow('Original Image', img)

cv.imshow('Lossless Compressed Image', lossless\_img)

cv.imshow('Lossy Compressed Image', lossy\_img)

cv.waitKey(0)

cv.destroyAllWindows()

**Output:**

