ORMEDIAN RESEARCH INSTITUTE

TOPIC: : OpenCV Tutorials (part two)

By:

ADEKOLA OLUWASEUN O.

February 15, 2023

Table of Content

- Essential Functions in OpenCV
- Conversion of Image from RGB to Gray Scale
- Edge Cascade
- Blurring an Image
- Reducing Edges in an Image
- Image Dilation
- Image Erosion
- Image Resizing (Enlarging and Shrinking)
- Image Cropping

Essential Functions in OpenCV

(1) Conversion of Image from RGB to Gray Scale

This is an OpenCV function that is very useful in the concept of image processing. Gray Scaling is needed for some reasons:

- ❑ It helps compress an image to its barest minimum pixels.
- ❑ It makes the simplification of the algorithm easy.
- ❑ It also eliminates the complexities related to computational requirements.

It is important to understand that RGB images contain a lot of information within them that may not be needed for processing. By Gray scaling an image, you have automatically discarded some of the information in such an image.

Essential Functions in OpenCV Cont’d

Simple Algorithm for Conversion of RGB Images to Gray Scale images

```
import cv2 as cv
img = cv.imread(‘path to file')
gray_image = cv.cvtColor(img,
cv.COLOR_RGB2GRAY)
cv.imshow('Original Image', img)
cv.imshow('GRAY', gray)
```

Essential Functions in OpenCV Cont’d

Interpretation of the Code

Line 1: Import the OpenCV module

Line 2: Specify the file path by calling the imread() method to read the image.

Line 3: Perform the RGB-to-Gray conversion by using the OpenCV inbuilt module.

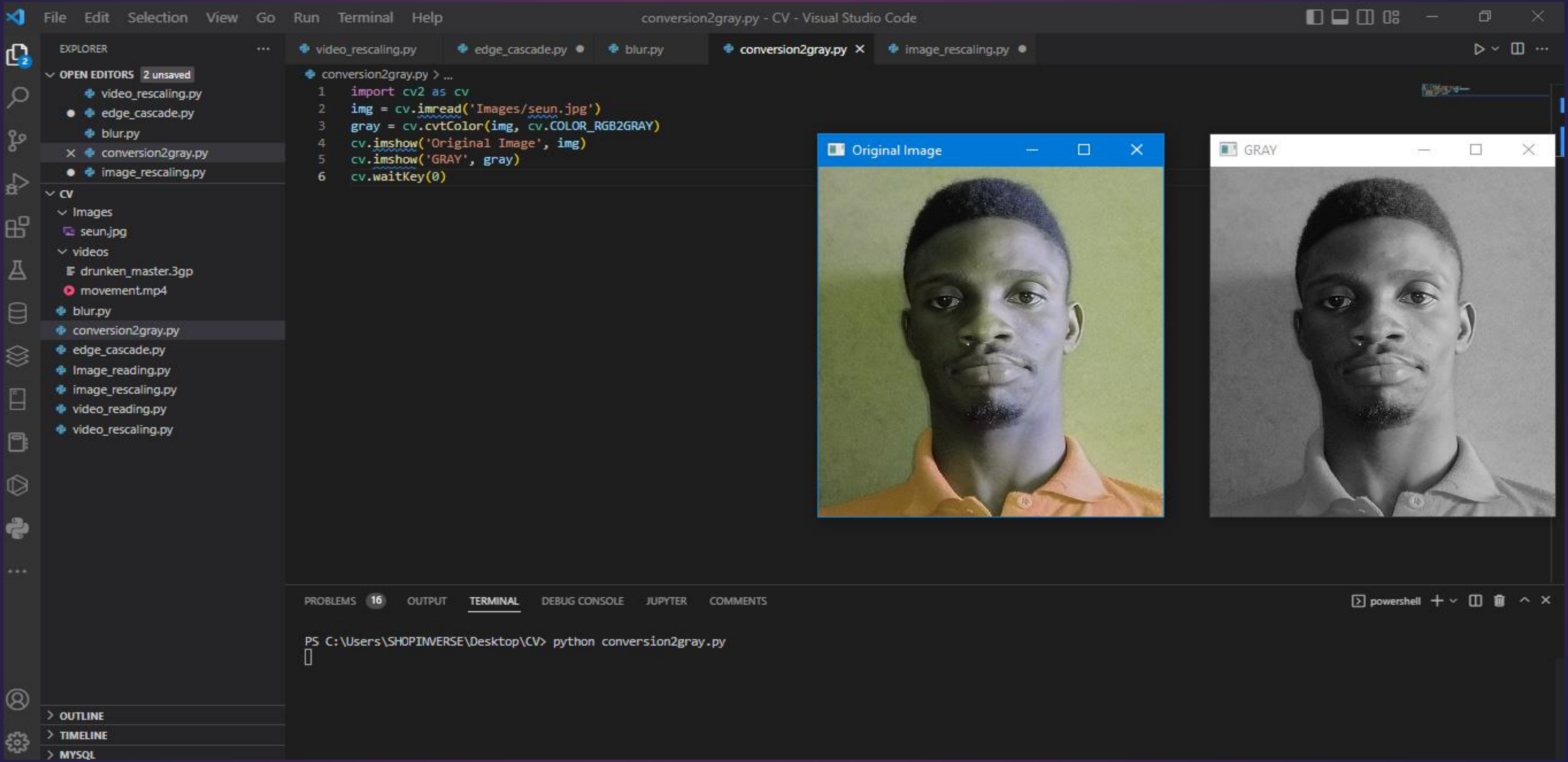
Lines 4 & 5: Call the imshow() method on the image to display the result of both original and the gray images.

Essential Functions in OpenCV Cont’d

Example: Convert an RGB image named “seun.jpg” in the images folder to a gray scale image.

```
import cv2 as cv
img = cv.imread('Images/seun.jpg')
gray = cv.cvtColor(img, cv.COLOR_RGB2GRAY)
cv.imshow('Original Image', img)
cv.imshow('GRAY', gray)
```

OUTPUT



```python
import cv2 as cv
img = cv.imread('Images/seun.jpg')
gray = cv.cvtColor(img, cv.COLOR_RGB2GRAY)
cv.imshow('Original Image', img)
cv.imshow('GRAY', gray)
cv.waitKey(0)
```

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python conversion2gray.py
```

Essential Functions in OpenCV Cont’d

(2) Edge Cascade

This is used for finding edges that are present in an image and objects. Although there are several packages in OpenCV for finding the edges but Canny edge detector will be used in this tutorial for finding the edge in an image. The operation of the edge detector is based on the discontinuities in the brightness of an image. Edge detection is very important because it helps you find the boundary within an image. The Canny edge detector uses a multistage algorithm from the initial stage to the final stage to detect a wide range of edges.

The steps that the Canny edge detector is composed of are:

- Noise reduction
- Gradient calculation
- Non-maximum Suppression
- Double threshold
- Edge Tracking by Hysteresis

Essential Functions in OpenCV Cont’d

Simple Algorithm to find Edges of an Image

```
import cv2 as cv
img = cv.imread(‘image path')
edges = cv.Canny(‘img’, minVal, maxVal, aperture, L2gradient)
cv.imshow('Original Image', img)
cv.imshow(“Detected edges”, edges)
```

Essential Functions in OpenCV Cont’d

Code interpretation

Line 1: Import the OpenCV module

Line 2: Specify the path to the image in the imread() method to read such a file

Line 3: Call the Canny edge detector syntax on the image and pass necessary parameters such as the file, minimum value of the intensity gradient (minVal), the maximum value of intensity gradient (maxVal), aperture (OPTIONAL), L2gradient is usually set to false for canny but in the case where computationally expensive equations are required, it is set to true for more accuracy.

Lines 4 & 5: To output the original image and the detected edges.

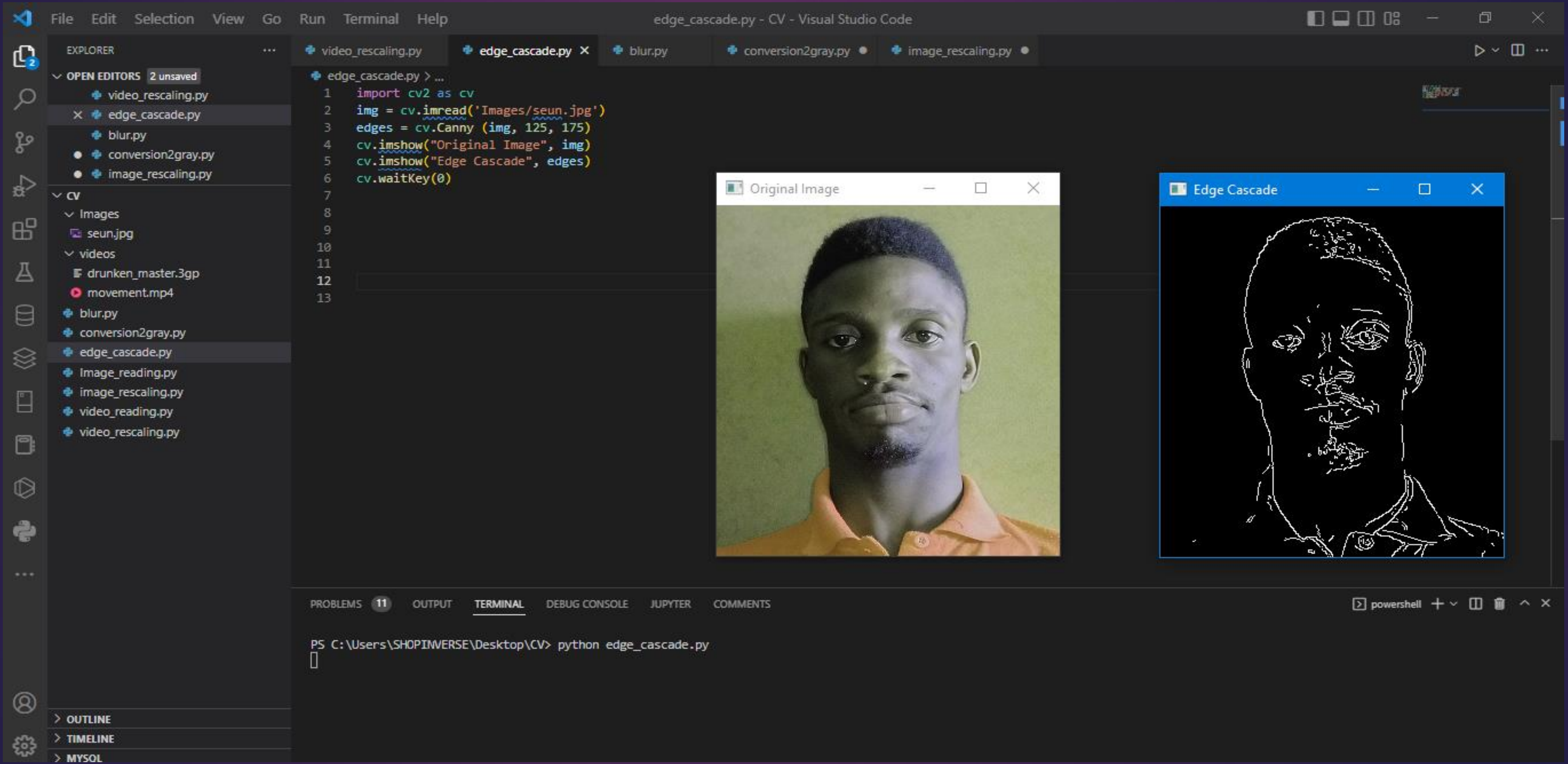
Essential Functions in OpenCV Cont’d

Example: Find the edges of the original image “seun.jpg”

SOLUTION

```
import cv2 as cv
img = cv.imread(‘image path')
edges = cv.Canny(‘img’, minVal, maxVal, aperture, L2gradient)
cv.imshow('Original Image', img)
cv.imshow(“Detected edges”, edges)
```

OUTPUT



```python
import cv2 as cv
img = cv.imread('Images/seun.jpg')
edges = cv.Canny (img, 125, 175)
cv.imshow("Original Image", img)
cv.imshow("Edge Cascade", edges)
cv.waitKey(0)
```

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python edge_cascade.py
```

Essential Functions in OpenCV Cont’d

(3) Blur an Image

This is a function used to reduce noise from an image by also removing the high-frequency content from an image. Convolution with the use of a low-pass filter i.e. kernel of M x N size is performed on an image to make the image blur. One of the techniques of obtaining a blur version of an image is by applying the GaussianBlur to it. When applying the GaussianBlur method on an image, some parameters such as the source file, kernel size, SigmaX, borderType, etc. To increase the image blurriness, the kernel size must be increased.

Essential Functions in OpenCV Cont’d

Simple Algorithm to Blur an Image

```
import cv2 as cv
img = cv.imread(‘image path')
Blur = cv.GaussianBlur(‘img’, kernelsize, cv.BORDER_DEFAULT)
cv.imshow('Original Image', img)
cv.imshow(“Blurred Image”, Blur)
```

Essential Functions in OpenCV Cont’d

Code interpretation

Line 1: Import the OpenCV module

Line 2: Specify the path to the image in the imread() method to read such a file.

Line 3: Call the GaussianBlur() method on the image, and specify the image source, kernel size, and other parameters.

Lines 4 & 5: To output the original image and the blurred image.

Essential Functions in OpenCV Cont'd

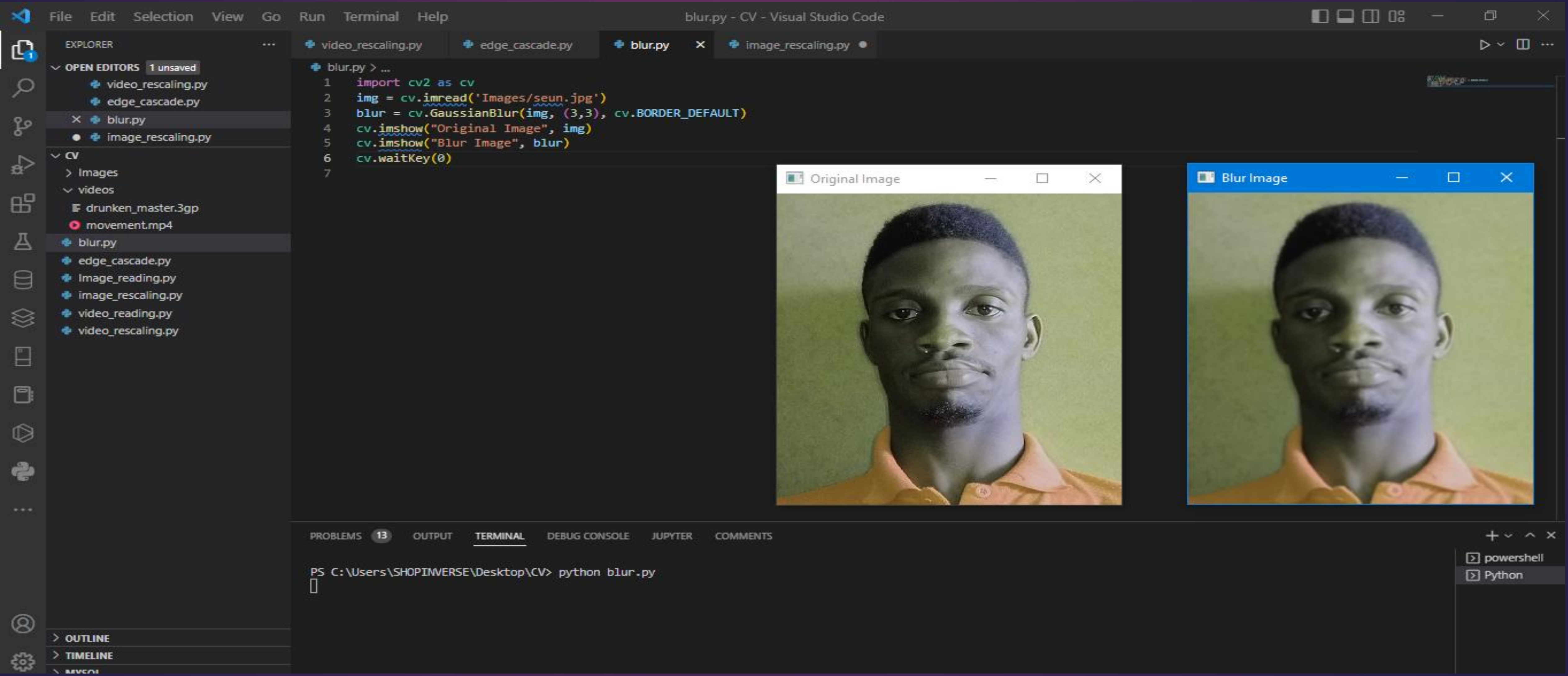
Example: Convert the original image “seun.jpg” to a blurry image by first applying 3x3 kernel size, then 5x5 kernel size.

SOLUTION

```
import cv2 as cv
img = cv.imread(“Images/seun.jpg”)
blur = cv.GaussianBlur(‘img’,(3, 3),  cv.BORDER_DEFAULT)
cv.imshow('Original Image', img)
cv.imshow(“Blur Image”, Blur)
```

OUTPUT

With 3x3 Kernel size

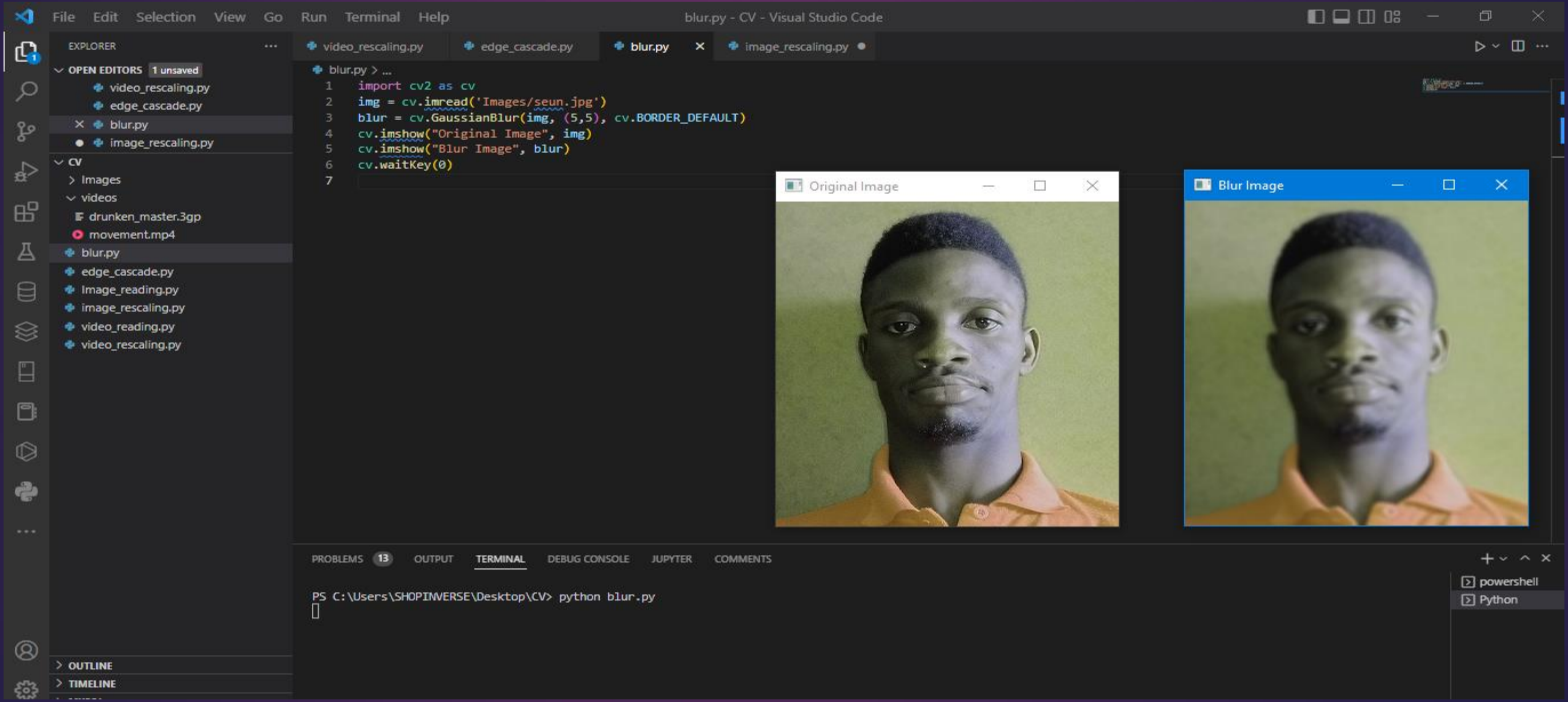


```python
import cv2 as cv
img = cv.imread('Images/seun.jpg')
blur = cv.GaussianBlur(img, (3,3), cv.BORDER_DEFAULT)
cv.imshow("Original Image", img)
cv.imshow("Blur Image", blur)
cv.waitKey(0)
```

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python blur.py
▯
```

OUTPUT

With 5x5 Kernel size



```python
import cv2 as cv
img = cv.imread('Images/seun.jpg')
blur = cv.GaussianBlur(img, (5,5), cv.BORDER_DEFAULT)
cv.imshow("Original Image", img)
cv.imshow("Blur Image", blur)
cv.waitKey(0)
```

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python blur.py
```

Essential Functions in OpenCV Cont'd

Reducing edges in an image

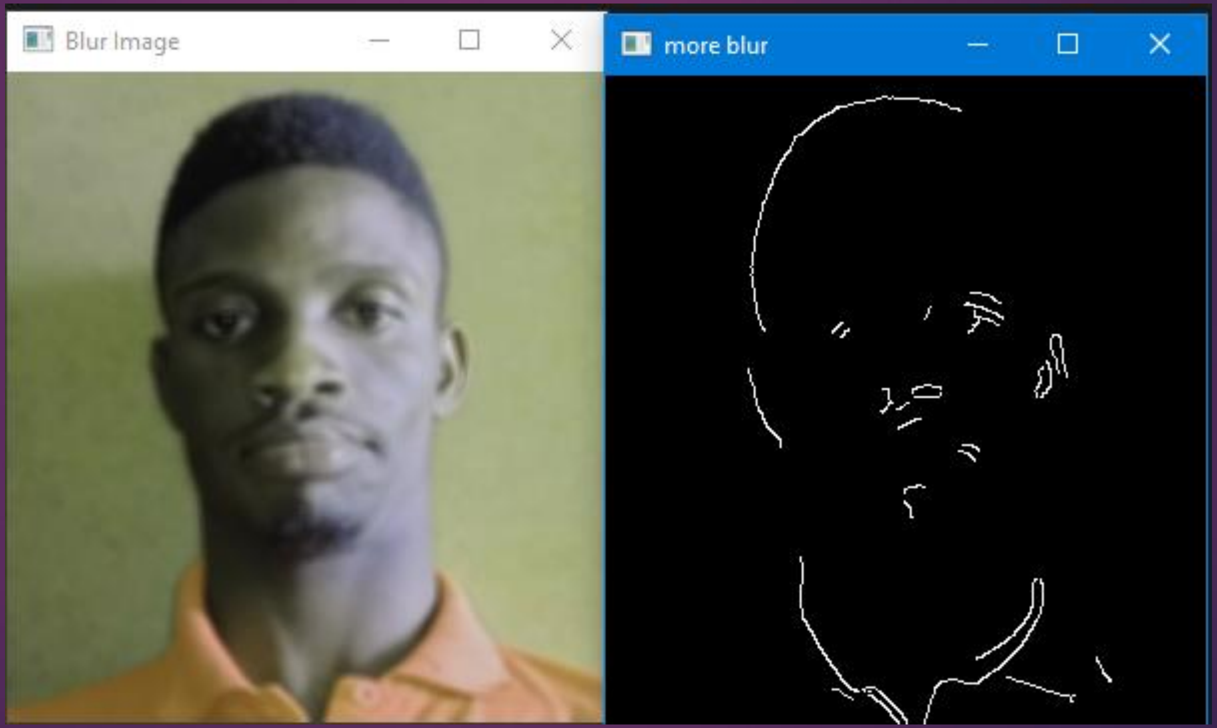
One of the ways to reduce the edges in an image is to simply blur such an image, then apply an edge detector like Canny edge detector to it. The final output of this process will produce reduced edges of the original image. By carefully examining the images and edges displayed on the next slide, we could see the edges are more reduced in the blurred image than in the original image. This is because the blurred image has less noise in it compared to the original image.

Essential Functions in OpenCV Cont’d

Edges Reduction Implementation Algorithm

```
import cv2 as cv
img = cv.imread('Images/seun.jpg')
blur = cv.GaussianBlur(img, (5,5), cv.BORDER_DEFAULT)
Reduced_edges= cv.Canny(blur, 125, 175)
cv.imshow("Blur Image", blur)
cv.imshow(“reduced edges",reduced_edges.)
cv.waitKey(0)
```

OUTPUT



Essential Functions in OpenCV Cont’d

(4) Dilation of an Image in OpenCV

Dilation refers to the increase in the object area of an image by simply adding pixels to the object boundaries. It specifically increases the white region in an image. It is a very useful function used for joining the broken part of an image together. While using the dilation function of OpenCV, a few parameters have to be set such as image file, kernel, and iteration.

Image file: This is the image on which dilation is to be applied

Kernel: This is used to specify the kernel size

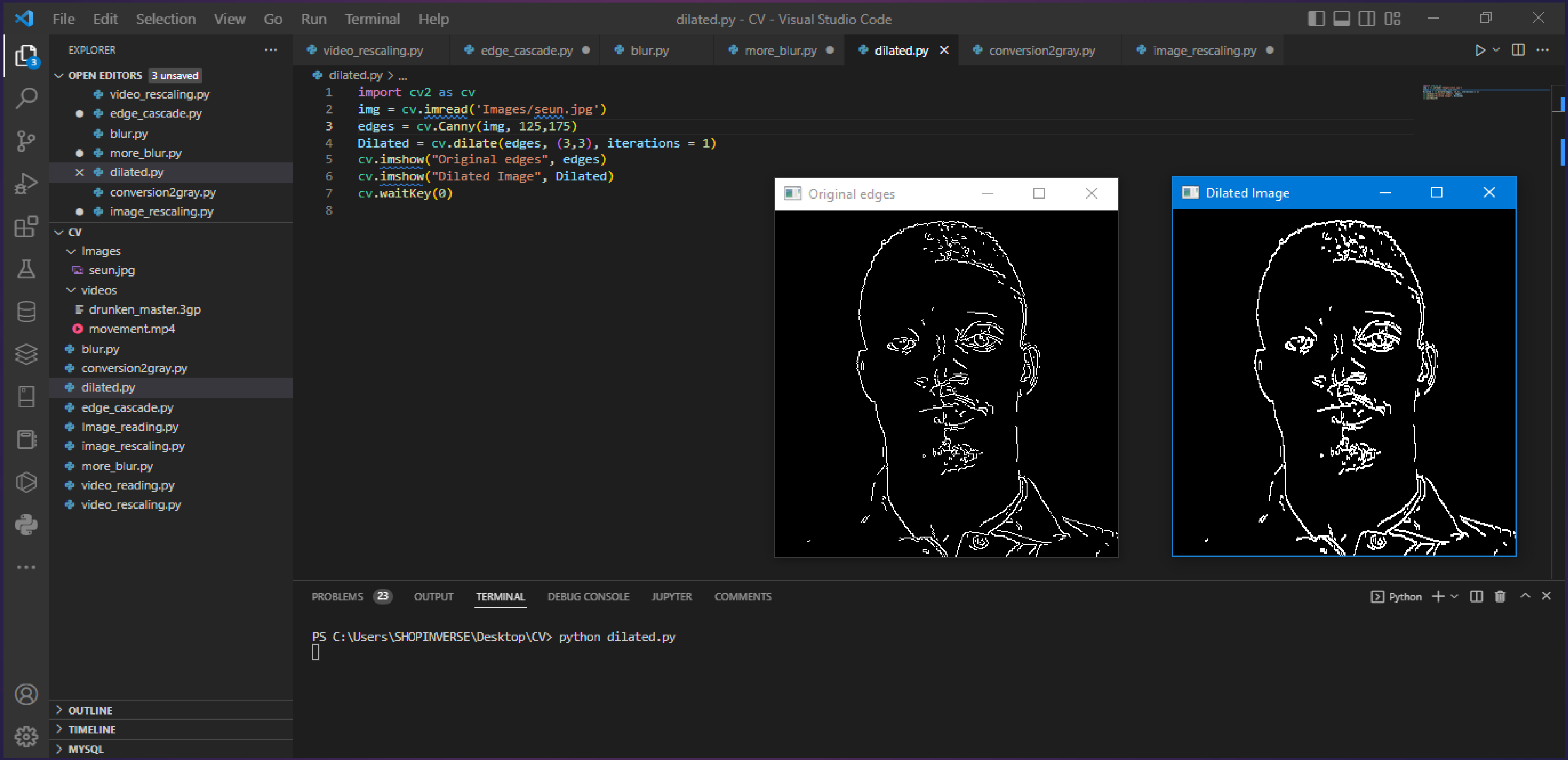
Iterations: The number of iterations of dilations to be performed.

Essential Functions in OpenCV Cont’d

Dilation Implementation Algorithm

```
import cv2 as cv
img = cv.imread('Images/seun.jpg')
edges = cv.Canny(img, 125,175)
Dilated = cv.dilate(edges, (3,3), iterations = 1)
cv.imshow("Original edges", edges)
cv.imshow("Dilated Image", Dilated)
cv.waitKey(0)
```

OUTPUT

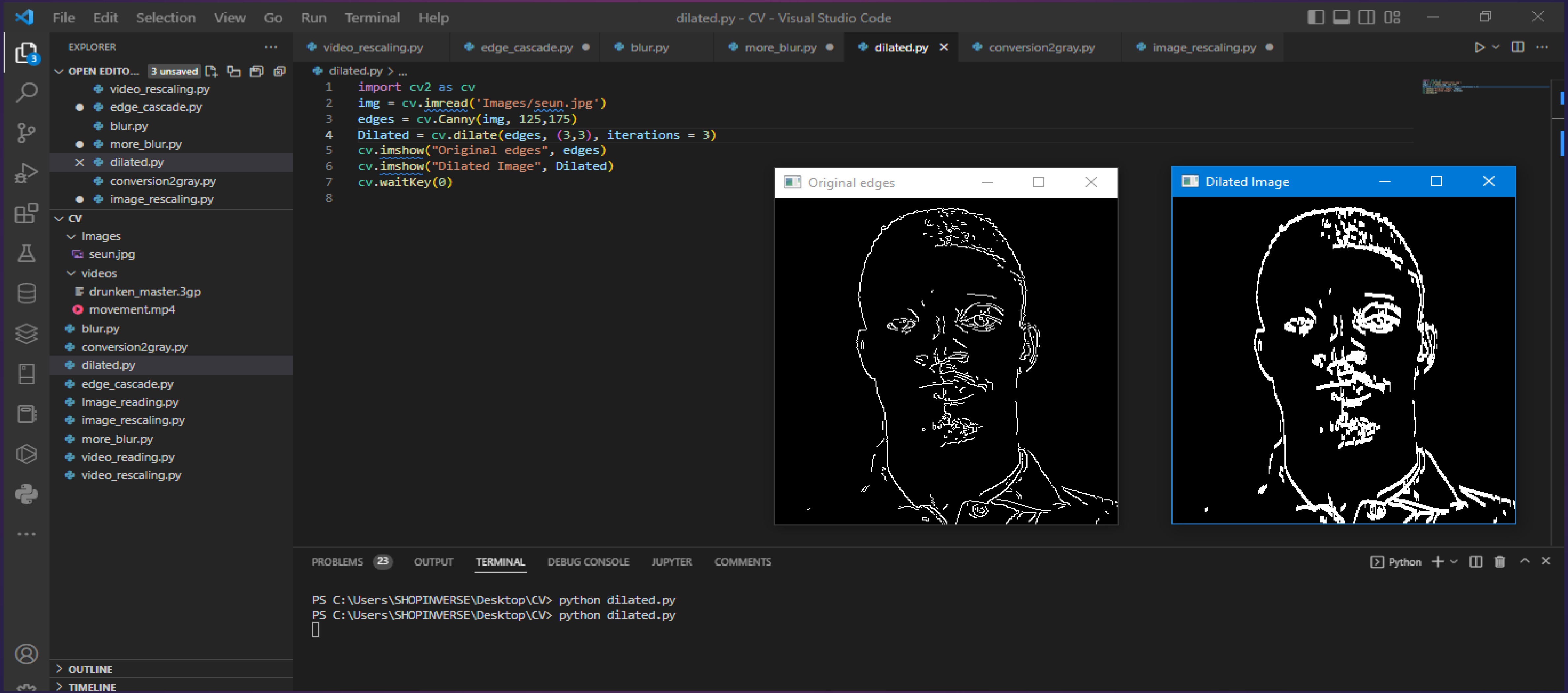


```python
import cv2 as cv
img = cv.imread('Images/seun.jpg')
edges = cv.Canny(img, 125,175)
Dilated = cv.dilate(edges, (3,3), iterations = 1)
cv.imshow("Original edges", edges)
cv.imshow("Dilated Image", Dilated)
cv.waitKey(0)
```

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python dilated.py
```

OUTPUT

With iterations increased from 1 to 3, the white area increased, more broken parts are joined together.



```python
import cv2 as cv
img = cv.imread('Images/seun.jpg')
edges = cv.Canny(img, 125,175)
Dilated = cv.dilate(edges, (3,3), iterations = 3)
cv.imshow("Original edges", edges)
cv.imshow("Dilated Image", Dilated)
cv.waitKey(0)
```

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python dilated.py
PS C:\Users\SHOPINVERSE\Desktop\CV> python dilated.py
```

Essential Functions in OpenCV Cont’d

(5) Erosion of Image

This is just the opposite of the dilation process. It simply removes pixels from the object boundaries. Simply put, erosion is the counter process of dilation i.e. erosion shrinks an image that has been enlarged by the dilation process. Hence dilated element is reversed to its structuring element upon the application of erosion on such an image.

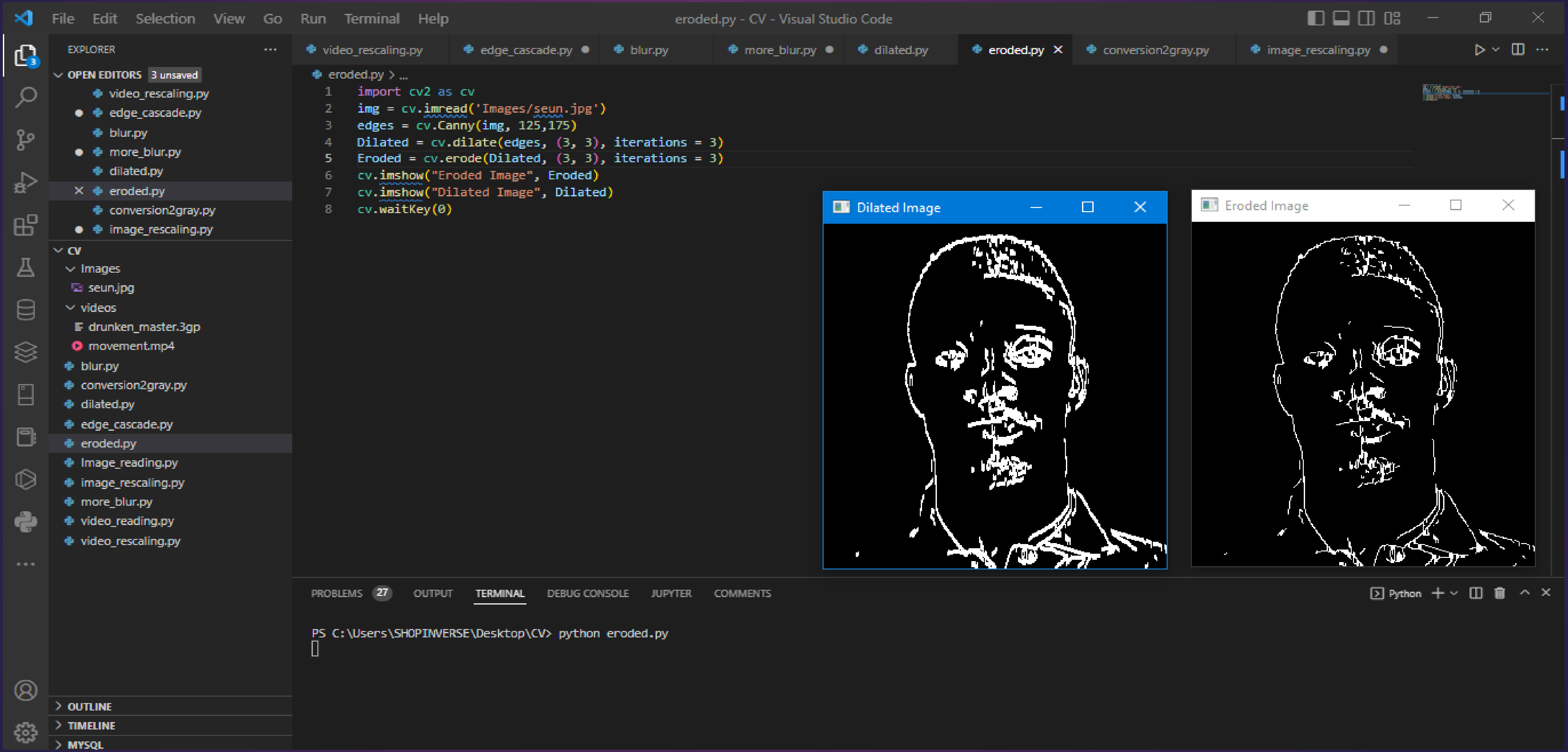
The possibility of getting a perfect structuring element depends solely on the setting of the parameters used. If the same steps and arguments passed in during the dilation are the same as those passed in during erosion, there is a greater chance of getting the perfect result which will counter the dilated result. If this rule is not well followed, the counter effect will never be realized.

Essential Functions in OpenCV Cont’d

Example: In the example shown below, I tried to reverse the result of the dilation to the original structuring element. So I simply applied the erode method to the dilated image to counter the possible changes it has experienced.

```
import cv2 as cv
img = cv.imread('Images/seun.jpg')
edges = cv.Canny(img, 125,175)
Dilated = cv.dilate(edges, (3, 3), iterations = 3)
Eroded = cv.erode(Dilated, (3, 3), iterations = 3)
cv.imshow("Eroded Image", Eroded)
cv.imshow("Dilated Image", Dilated)
cv.waitKey(0)
```

OUTPUT



```python
import cv2 as cv
img = cv.imread('Images/seun.jpg')
edges = cv.Canny(img, 125,175)
Dilated = cv.dilate(edges, (3, 3), iterations = 3)
Eroded = cv.erode(Dilated, (3, 3), iterations = 3)
cv.imshow("Eroded Image", Eroded)
cv.imshow("Dilated Image", Dilated)
cv.waitKey(0)
```

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python eroded.py
```

Essential Functions in OpenCV Cont'd

(6) Image Resizing

Image resizing simply refers to the process of either enlarging or reducing the original size of an image. in openCV image can either be downscaled or upscaled. this depends completely on the need of the user.The simple syntax could be used for resizing an image. And this requires the need for specification of one or more parameters to give a precise or accurate description of what the result should look like. By default, a simple syntax required for the image resizing is given as:

```
resized_image = cv.resize(img, (width, height))
```

But if further operation such as image enlargement, image shrinking etc. is required, then there is a need to call an interpolation parameter for the implementation of further operation. The argument passed into the interpolation parameter could be cv.INTER_AREA (in the case of shrinking), cv.INTER_LINEAR (In the case of enlarging), or cv.INTER_CUBIC (in the case of enlarging but slower in operation). The code is modified to:

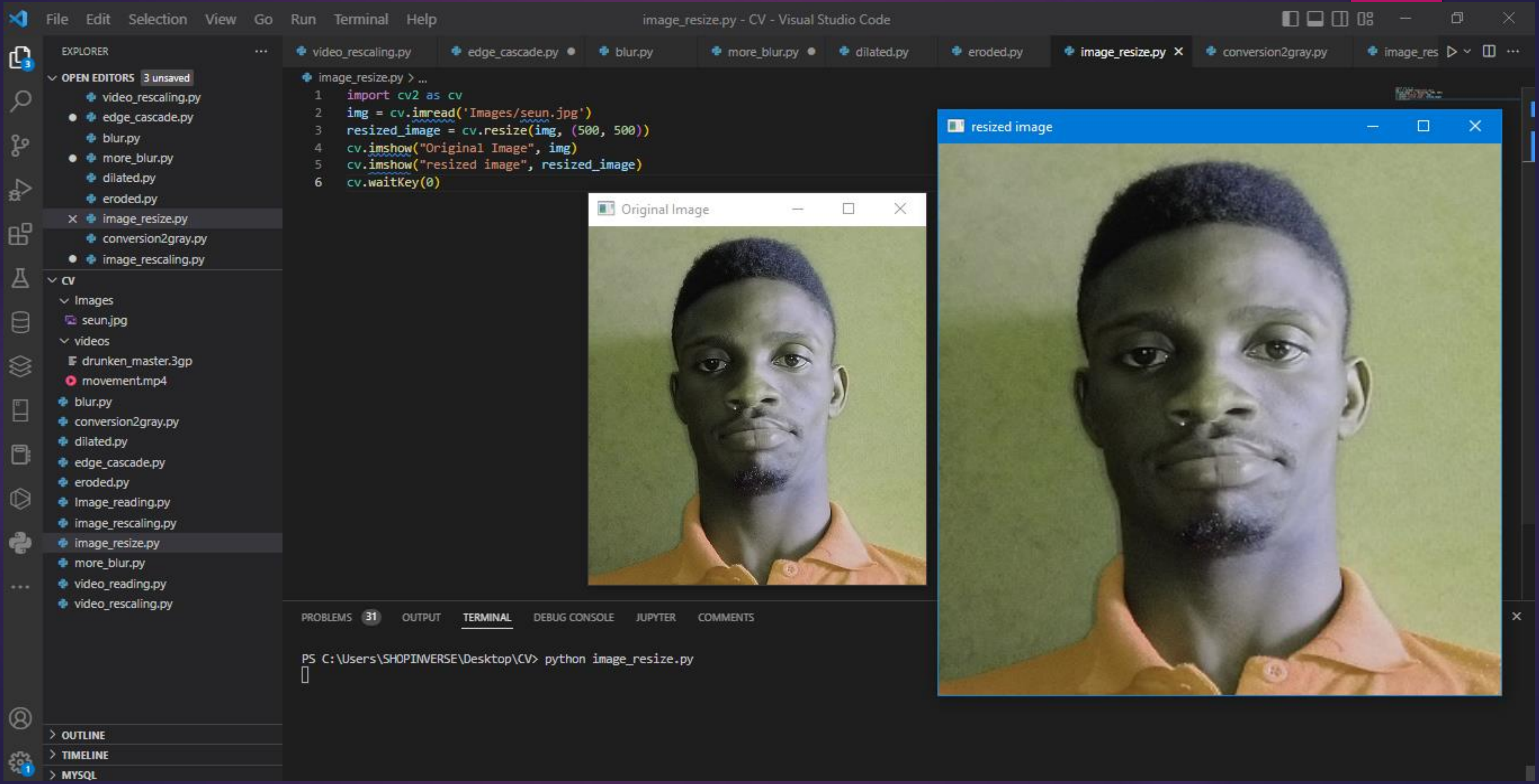
```
resized_image = cv.resize(img, (width, height), interpolation = cv.INTER_LINEAR)
```

Essential Functions in OpenCV Cont'd

EXAMPLE: Resize the original image “seun.jpg” to 500 x 500 pixels

```
import cv2 as cv
img = cv.imread('Images/seun.jpg')
resized_image = cv.resize(img, (500, 500))
cv.imshow("Original Image", img)
cv.imshow("resized image", resized_image)
cv.waitKey(0)
```

OUTPUT



```python
import cv2 as cv
img = cv.imread('Images/seun.jpg')
resized_image = cv.resize(img, (500, 500))
cv.imshow("Original Image", img)
cv.imshow("resized image", resized_image)
cv.waitKey(0)
```

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python image_resize.py
```

Essential Functions in OpenCV Cont'd

(7) Image Cropping

Image Cropping could be regarded as an image processing technique that involves the cutting off or the trimming off of the edges of an image to have a more meaningful image. Cropping of an image is done specifically to remove some information from the original image. This could also be done to remove unwanted information from an image. This could also be done to improve image framing.

Simple syntax for cropping of image:

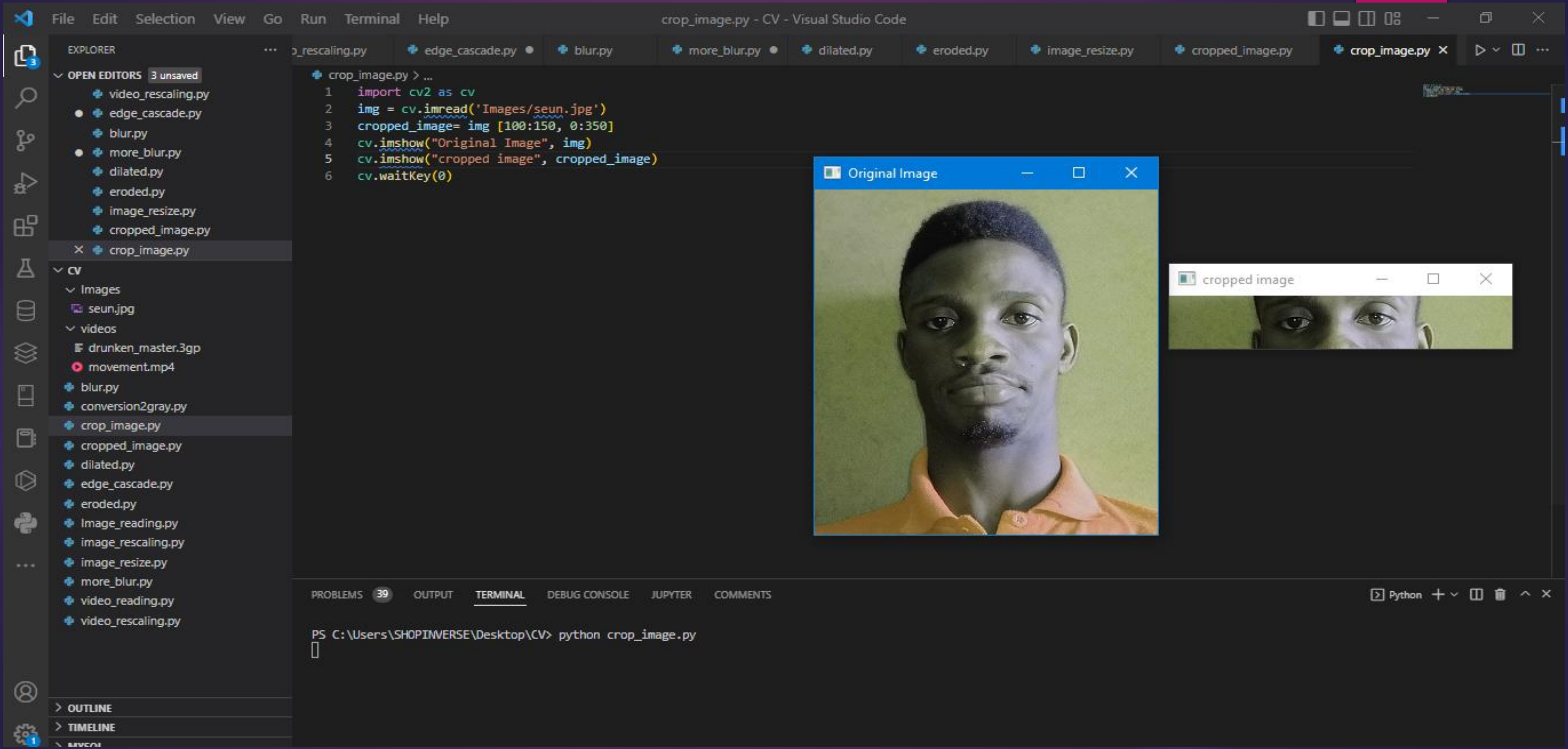
Cropped_image = img [slicing option values for width and height]

Essential Functions in OpenCV Cont’d

Example: Perform cropping operation on the image “seun.jpg” by applying the slicing option [100:150, 0:350]

```
import cv2 as cv
img = cv.imread('Images/seun.jpg')
cropped_image= img [100:150, 0:350]
cv.imshow("Original Image", img)
cv.imshow("cropped image", cropped_image)
cv.waitKey(0)
```

OUTPUT



```python
import cv2 as cv
img = cv.imread('Images/seun.jpg')
cropped_image= img [100:150, 0:350]
cv.imshow("Original Image", img)
cv.imshow("cropped image", cropped_image)
cv.waitKey(0)
```

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python crop_image.py
```

THANK YOU FOR VIEWING

PROCEED TO PART 3 OF THE Opencv TUTORIALS.