



**ORMEDIAN RESEARCH INSTITUTE**

TOPIC: : **OpenCV Tutorials**  
**(part three)**

By:

**ADEKOLA OLUWASEUN O.**

**February 17, 2023**

# TABLE OF CONTENT

- ❑ IMAGE TRANSFORMATION
- ❑ IMAGE TRANSLATION
- ❑ IMAGE ROTATION
- ❑ IMAGE FLIPPING
- ❑ IMAGE RESIZING
- ❑ IMAGE CROPPING
- ❑ READING OF IMAGE/VIDEO FROM WEBCAM

# IMAGE TRANSFORMATION IN OPENCV

## Image Transformation

In OpenCV, we regard image transformation as the process of transforming image data for further reprocessing. It works on the principle of the Euclidean transformation in which the image is subjected to some changes in angles and dimensions without a change in its structure-like area. Specifically, the transformation of this form involves the mapping of some points in one coordinate to some other points in another coordinate.

Some common basic image transformation techniques are:Image Translation

- ❖ Image Rotation
- ❖ Image Cropping
- ❖ Image Resizing
- ❖ Image Flipping

# IMAGE TRANSFORMATION IN OPENCV

## Cont'd

### Image Translation

Image Translation simply refers to the shifting of an image along the x- and y-axis direction i.e. the shifting of an image in any direction be it left, right, up, or down or possibly by combining these options. While translating an image along the x and y-axis direction, it is important to understand that the values of x or y matters. The effects of the possible values of x or y and their implications on the image translation are listed below:

Negative value of x implies shifting of an image to the left

Negative value of y implies shifting of an image up

Positive value of x implies shifting of an image to the right

Positive value of y implies shifting of the image down

# IMAGE TRANSFORMATION IN OPENCV

## Cont'd

### Algorithm for image translation

```
import cv2 as cv
import numpy as np
img = cv.imread("image path")
def translate(img, x, y):
    TransMat = np.float32 ([[1, 0, x], [0, 1, y]])
    dimensions = [img.shape[0], img.shape[1]]
    return cv.warpAffine(img, TransMat, dimensions)
Translated_img = translate (img, x-axis value, y-axis value )
cv.imshow("Original Image", img)
cv.imshow("Translated Image", Translated_img)
cv.waitKey(0)
```

# IMAGE TRANSFORMATION IN OPENCV

## Cont'd

### Image Transformation Code Interpretation

**Line 1:** Import OpenCV module i.e. cv

**Line 2:** Import the NumPy module

**Line 3:** Specify the image path and read the image

**Line 4:** Create a function for the translation which will take the input image, x-axis, and y-axis values as parameters.

**Line 5:** Specify the translational matrix

**Line 6:** To get the image dimensions

**Line 7:** Apply cv.warpAffine() to perform the remapping routine for the translation process

**Line 8:** Finally call the function [previously defined and pass in the necessary arguments

**Lines 9 & 10:** Display the old image (Optional) and display the translated image respectively to see the results.

**NB:** Although the **cv.warpAffine()** method takes in a few parameters such as the **src:** input image, **dst:** output image that has the same size and type as the input image, **M:** Transformation matrix, **dsize:** the size of the output image, **flags:** the combination of interpolation method. Others are **borderMode** and **borderValue**. But in this example, we have only used three parameters and made the rest optional.

# IMAGE TRANSFORMATION IN OPENCV

## Cont'd

**Example 1:** Translate the image named “seun.jpg” by shifting the image to right by 70 pixels and down by 50 pixels.

```
import cv2 as cv
import numpy as np
img = cv.imread("Images/Seun.jpg")
def translate(img, x, y):
    TransMat = np.float32 ([[1, 0, x], [0, 1, y]])
    dimensions = [img.shape[0], img.shape[1]]
    return cv.warpAffine(img, TransMat, dimensions)
Translated_img = translate (img, 170, 120)
cv.imshow("Original Image", img)
cv.imshow("Translated Translated", Translated_img)
cv.waitKey(0)
```



# OUTPUT

Visual Studio Code interface showing the execution of a Python script for image translation.

**EXPLORER:** The file explorer on the left shows the project structure. The **CV** folder contains an **Images** subfolder with **seun.jpg**. Other files include **edge\_cascade.py**, **blur.py**, **more\_blur.py**, **dilated.py**, **eroded.py**, **image\_resize.py**, **cropped\_image.py**, **crop\_image.py**, and **img\_translate.py**.

**img\_translate.py:** The code in the editor defines a `translate` function that uses `cv2.warpAffine` to shift an image by `x` and `y` pixels. The script reads `Images/Seun.jpg`, translates it by (70, 50), and displays both the original and translated images.

```
1 import cv2 as cv
2 import numpy as np
3 img = cv.imread("Images/Seun.jpg")
4 def translate(img, x, y):
5     TransMat = np.float32 ([[1, 0, x], [0, 1, y]])
6     dimensions = [img.shape[0], img.shape[1]]
7     return cv.warpAffine(img, TransMat, dimensions)
8 Translated_img = translate (img, 70, 50)
9 cv.imshow("Original Image", img)
10 cv.imshow("Translated Image", Translated_img)
11 cv.waitKey(0)
12
```

**Terminal:** The terminal at the bottom shows the command `python img_translate.py` being executed successfully in the directory `C:\Users\SHOPINVERSE\Desktop\CV`.

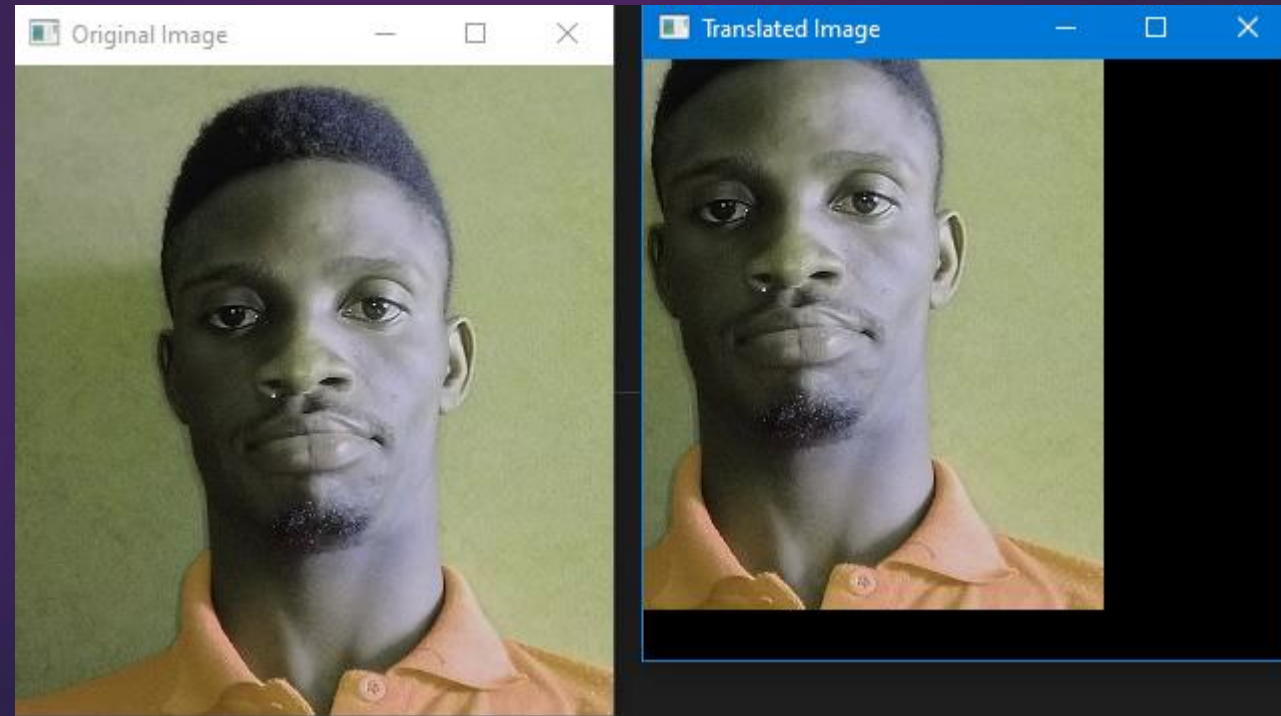
**Output:** Two windows are displayed side-by-side. The **Original Image** window shows the original portrait of a man. The **Translated Image** window shows the same portrait shifted to the right and slightly down, demonstrating the effect of the `translate` function.



# IMAGE TRANSFORMATION IN OPENCV Cont'd

**Example 2:** Translate the image named “seun.jpg” by shifting the image to left by 70 pixels and up by 50 pixels.

## OUTPUT



## Image Rotation in OpenCV

Image Rotation in openCV refers to the image processing technique in which the image is correctly displayed from different angles. Image rotation involves the computation of the inverse transformation for every destination pixel. Simply put, it is the process of rotating an image. OpenCV has a method to simply rotate an image. But if there is a need for some arbitrary angles for rotation then there is a need to use a more complex one. This means that the `cv2.rotate()` method is used for the rotation of a 2D array of an image in multiple of  $90^\circ$  whereas the `cv2.getRotationMatrix2D()` is used in the case of arbitrary angles when transformation matrix will be required for the rotation of an image.

For the `cv2.rotate()` method, the simplest syntax will be:

```
rotated_image = cv.rotate(src, rotationCode)
```

Where `src` implies the input image while `rotationCode` could be `cv2.ROTATE_90_CLOCKWISE`, `cv2.ROTATE_90_COUNTERCLOCKWISE` or `cv2.ROTATE_180`. These are the three basic rotation codes available in OpenCV.

# IMAGE TRANSFORMATION IN OPENCV

Cont'd

Using the `cv2.rotate()` method

## SYNTAX

```
import cv2 as cv
src = cv.imread("path to the image file")
rotated_image = cv.rotate(src, rotationCode)
cv.imshow("Original Image", src)
cv.imshow("Rotated Image", rotated_image)
cv.waitKey(0)
```

# IMAGE TRANSFORMATION IN OPENCV

# Cont'd

## Using the `cv2.rotate()` method

**Example 1:** Rotate the original image “seun.jpg” by (i) 90° clockwise (ii) 90° counterclockwise and, (iii) 180° clockwise.

## SOLUTION

(i)

```
import cv2 as cv
img = cv.imread("Image/seun.jpg")
rotated_image = cv.rotate(img,
cv.ROTATE_90_CLOCKWISE)
cv.imshow("Original Image", img)
cv.imshow("Rotated Image",
rotated_image)
cv.waitKey(0)
```

(ii)

```
import cv2 as cv
img = cv.imread("Image/seun.jpg")
rotated_image = cv.rotate(img,
cv.ROTATE_90_COUNTERCLOCK
WISE)
cv.imshow("Original Image", img)
cv.imshow("Rotated Image",
rotated_image)
cv.waitKey(0)
```

(iii)

```
import cv2 as cv
img = cv.imread("Image/seun.jpg")
rotated_image = cv.rotate(img,
cv.ROTATE_180)
cv.imshow("Original Image", img)
cv.imshow("Rotated Image",
rotated_image)
cv.waitKey(0)
```



# OUTPUT

Visual Studio Code interface showing the execution of a Python script for image rotation.

**EXPLORER**

- OPEN EDITORS 3 unsaved
  - blur.py
  - more\_blur.py
  - dilated.py
  - eroded.py
  - image\_resize.py
  - cropped\_image.py
  - crop\_image.py
  - img\_translate.py
  - img\_rotate.py
- cv
  - blur.py
  - conversion2gray.py
  - crop\_image.py
  - cropped\_image.py
  - dilated.py
  - edge\_cascade.py
  - eroded.py
  - Image\_reading.py
  - image\_rescaling.py
  - image\_resize.py
  - img\_rotate.py
  - img\_translate.py
  - more\_blur.py
  - video\_reading.py
  - video\_rescaling.py

**img\_rotate.py**

```
1 import cv2 as cv
2 img = cv.imread("Images/seun.jpg")
3 rotated_image = cv.rotate(img, cv.ROTATE_90_CLOCKWISE)
4 cv.imshow("Original Image", img)
5 cv.imshow("Rotated Image", rotated_image)
6 cv.waitKey(0)
7
```

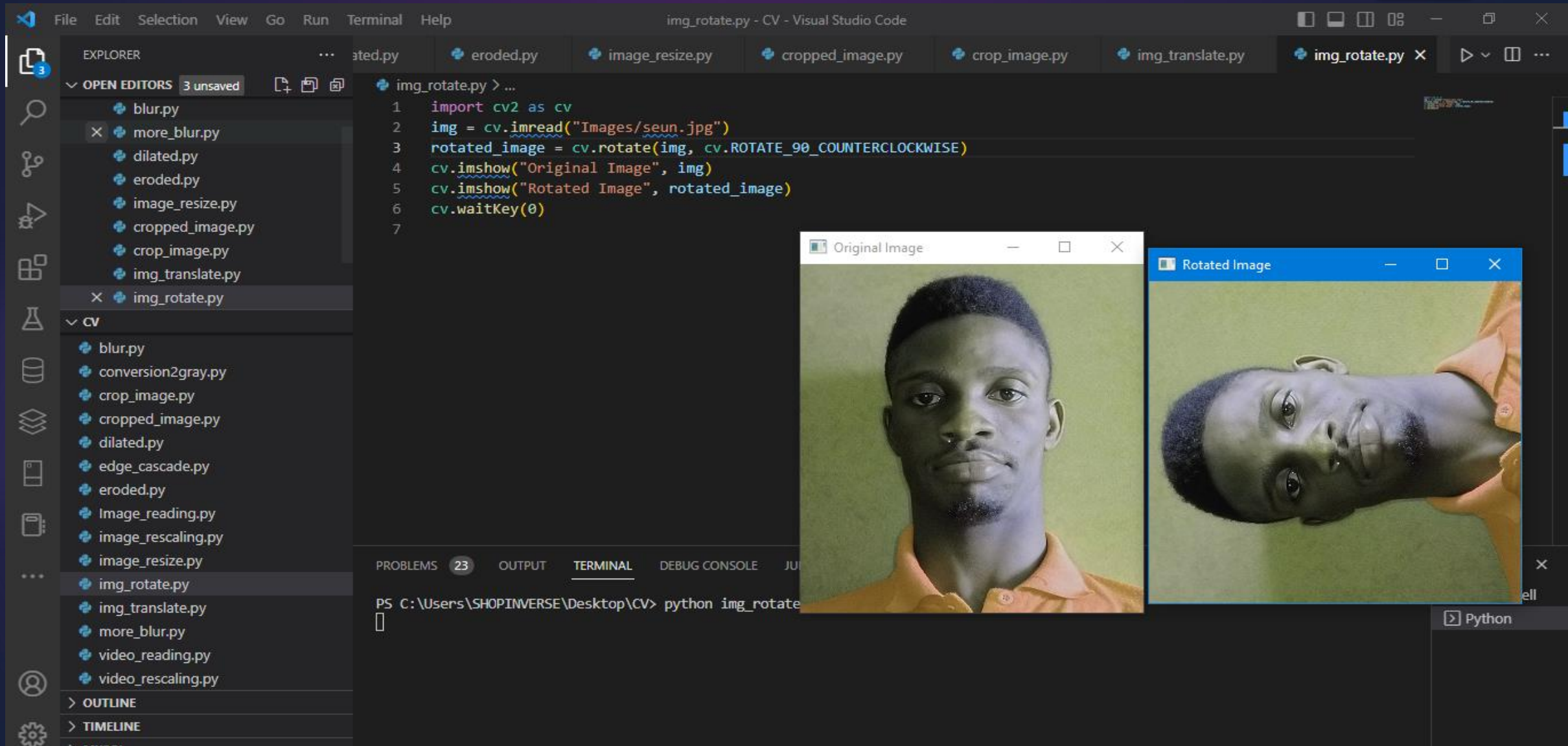
**Terminal**

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

**Output Windows:**

- Original Image: Shows the original portrait of a man.
- Rotated Image: Shows the original portrait rotated 90 degrees clockwise.

# OUTPUT





# OUTPUT

Visual Studio Code interface showing the execution of a Python script for image rotation.

**EXPLORER**

- OPEN EDITORS (3 unsaved)
  - blur.py
  - more\_blur.py
  - dilated.py
  - eroded.py
  - image\_resize.py
  - cropped\_image.py
  - crop\_image.py
  - img\_translate.py
  - img\_rotate.py
- CV
  - blur.py
  - conversion2gray.py
  - crop\_image.py
  - cropped\_image.py
  - dilated.py
  - edge\_cascade.py
  - eroded.py
  - Image\_reading.py
  - image\_rescaling.py
  - image\_resize.py
  - img\_rotate.py
  - img\_translate.py
  - more\_blur.py
  - video\_reading.py
  - video\_rescaling.py

**img\_rotate.py**

```
1 import cv2 as cv
2 img = cv.imread("Images/seun.jpg")
3 rotated_image = cv.rotate(img, cv.ROTATE_180)
4 cv.imshow("Original Image", img)
5 cv.imshow("Rotated Image", rotated_image)
6 cv.waitKey(0)
7
```

**Terminal**

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

**Output Windows**

- Original Image: Shows the original portrait of a man.
- Rotated Image: Shows the original portrait rotated 180 degrees.

# IMAGE TRANSFORMATION IN OPENCV

## Cont'd

### Image Rotation using the cv2.getRotationMatrix2D() Method

```
import cv2 as cv
img = cv.imread("Images/seun.jpg")
def rotate(img, angle, RotPt=None):
    (width, height) = img.shape[:2]
    if RotPt is None:
        rotPt = (width//2, height//2)
        rotMat = cv.getRotationMatrix2D(rotPt, angle, 1.0)
        dimensions = (width, height)
        return cv.warpAffine(img, rotMat, dimensions)
rotated_image = rotate(img, angle_value)
cv.imshow("Original Image", img)
cv.imshow("Rotated Image", rotated_image)
cv.waitKey(0)
```

# IMAGE TRANSFORMATION IN OPENCV

# Cont'd

## Image Transformation Code Interpretation

**Line 1:** Import OpenCV module i.e. cv

**Line 2:** Specify the image path and read the image

**Line 3:** Create a function for the rotation which will take the input image, angle, and rotation point as parameters.

**Line 4:** Grab the image dimension and calculate the image center point.

**Line 5:** To get the image dimensions

**Line 6:** Grab the rotation matrix by specifying the angle and scaling.

**Line 7:** Set the dimension variable

**Line 8:** Perform the remapping routing using the cv.warpAffine() method.

**Lines 9 & 10:** Display the old image (Optional) and display the translated image respectively to see the results.

# IMAGE TRANSFORMATION IN OPENCV

# Cont'd

**Example:** Rotate the original image “seun.jpg” by 150°, then by 320°.

## SOLUTION

### Rotation by 150°

```
import cv2 as cv
img = cv.imread("Images/seun.jpg")
def rotate(img, angle, RotPt=None):
    (width, height) = img.shape[:2]
    if RotPt is None:
        rotPt = (width//2, height//2)
        rotMat = cv.getRotationMatrix2D(rotPt, angle, 1.0)
        dimensions = (width, height)
        return cv.warpAffine(img, rotMat, dimensions)
rotated_image = rotate(img, 150)
cv.imshow("Original Image", img)
cv.imshow("Rotated Image", rotated_image)
cv.waitKey(0)
```

### Rotation by 320°

```
import cv2 as cv
img = cv.imread("Images/seun.jpg")
def rotate(img, angle, RotPt=None):
    (width, height) = img.shape[:2]
    if RotPt is None:
        rotPt = (width//2, height//2)
        rotMat = cv.getRotationMatrix2D(rotPt, angle, 1.0)
        dimensions = (width, height)
        return cv.warpAffine(img, rotMat, dimensions)
rotated_image = rotate(img, 320)
cv.imshow("Original Image", img)
cv.imshow("Rotated Image", rotated_image)
cv.waitKey(0)
```



# OUTPUT

Visual Studio Code interface showing the output of a Python script for image rotation.

**EXPLORER**

- more\_blur.py
- dilated.py
- eroded.py
- image\_resize.py
- cropped\_image.py
- crop\_image.py
- img\_translate.py
- img\_rotate.py
- img\_rotate2.py

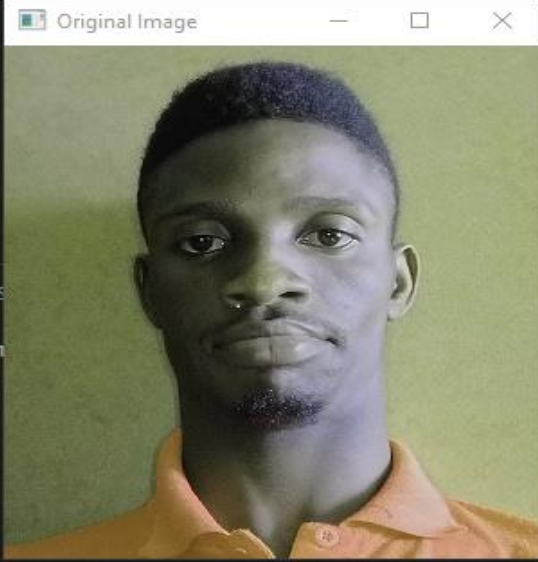
**CV**

- blur.py
- conversion2gray.py
- crop\_image.py
- cropped\_image.py
- dilated.py
- edge\_cascade.py
- eroded.py
- Image\_reading.py
- image\_rescaling.py
- image\_resize.py
- img\_rotate.py
- img\_rotate2.py
- img\_translate.py
- more\_blur.py
- video\_reading.py

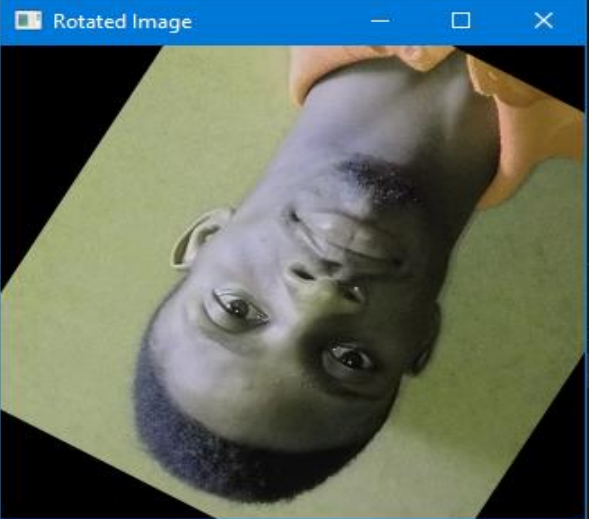
**img\_rotate2.py**

```
1 import cv2 as cv
2 img = cv.imread("Images/seun.jpg")
3 def rotate(img, angle, RotPt=None):
4     (width, height) = img.shape[:2]
5     if RotPt is None:
6         rotPt = (width//2, height//2)
7         rotMat = cv.getRotationMatrix2D(rotPt, angle, 1.0)
8         dimensions = (width, height)
9         return cv.warpAffine(img, rotMat, dimensions)
10 rotated_image = rotate(img, 150)
11 cv.imshow("Original Image", img)
12 cv.imshow("Rotated Image", rotated_image)
13 cv.waitKey(0)
14
15
```

**Original Image**



**Rotated Image**



**TERMINAL**

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python im
```

# OUTPUT

Visual Studio Code interface showing the execution of a Python script for image rotation.

**EXPLORER:** The file explorer on the left shows the project structure. The file `img_rotate2.py` is selected.

**CODE EDITOR:** The main editor displays the code for `img_rotate2.py`:

```
1 import cv2 as cv
2 img = cv.imread("Images/seun.jpg")
3 def rotate(img, angle, RotPt=None):
4     (width, height) = img.shape[:2]
5     if RotPt is None:
6         rotPt = (width//2, height//2)
7         rotMat = cv.getRotationMatrix2D(rotPt, angle, 1.0)
8         dimensions = (width, height)
9         return cv.warpAffine(img, rotMat, dimensions)
10 rotated_image = rotate(img, 320)
11 cv.imshow("Original Image", img)
12 cv.imshow("Rotated Image", rotated_image)
13 cv.waitKey(0)
14
15
```

**OUTPUT:** The output pane at the bottom shows the execution of the script, displaying the command prompt prompt `PS C:\Users\SHOPINVERSE\Desktop\CV> python in`.

**Image Windows:** Two windows are open, showing the original image and the rotated image. The "Original Image" window shows a portrait of a man. The "Rotated Image" window shows the same portrait rotated 320 degrees.

**Terminal:** The terminal at the bottom shows the command prompt prompt `PS C:\Users\SHOPINVERSE\Desktop\CV> python in`.



# IMAGE TRANSFORMATION IN OPENCV **Cont'd**

## Image Flipping in OpenCV

This could be regarded as the process of turning an image horizontally or vertically. In the process of flipping the right side of an image is made to become the left, and the top becomes the bottom. In a nutshell, flipping is simply the mirror image of the original image.

In the process of flipping an image, there is no need to define a function. `cv2.flip()` method is applied to such an image. The `cv2.flip()` method takes in two arguments i.e. the input image and the flipCode. The flipCode takes in value 0 or 1 or -1. 0 specifies vertical flipping, 1 specifies horizontal flipping while -1 specifies both horizontal and vertical flipping.

## SYNTAX

```
Flipped_image = cv.flip(img, flipCode)
```

# IMAGE TRANSFORMATION IN OPENCV **Cont'd**

**Example:** Flip the original image “seun.jpg” vertically, horizontally, and both horizontally and vertically.

## Vertical Flipping

```
import cv2 as cv
img = cv.imread("Images/seun.jpg")
flipped_image = cv.flip(img, 0)
cv.imshow("Original Image", img)
cv.imshow("Flipped Image", flipped_image)
cv.waitKey(0)
```

## Horizontal Flipping

```
import cv2 as cv
img = cv.imread("Images/seun.jpg")
flipped_image = cv.flip(img, 1)
cv.imshow("Original Image", img)
cv.imshow("Flipped Image", flipped_image)
cv.waitKey(0)
```

## Horizontal and Vertical Clipping

```
import cv2 as cv
img = cv.imread("Images/seun.jpg")
flipped_image = cv.flip(img, -1)
cv.imshow("Original Image", img)
cv.imshow("Flipped Image", flipped_image)
cv.waitKey(0)
```

# OUTPUT

Visual Studio Code interface showing the execution of a Python script using OpenCV to flip an image.

**EXPLORER**

- OPEN EDITORS (3 unsaved)
  - dilated.py
  - eroded.py
  - image\_resize.py
  - cropped\_image.py
  - crop\_image.py
  - img\_translate.py
  - img\_rotate.py
  - img\_rotate2.py
  - flip.py
- CV
  - blur.py
  - conversion2gray.py
  - crop\_image.py
  - cropped\_image.py
  - dilated.py
  - edge\_cascade.py
  - eroded.py
  - flip.py
  - Image\_reading.py
  - image\_rescaling.py
  - image\_resize.py
  - img\_rotate.py
  - img\_rotate2.py
  - img\_translate.py
  - more\_blur.py

**flip.py**

```
1 import cv2 as cv
2 img = cv.imread("Images/seun.jpg")
3 flipped_image = cv.flip(img, 0)
4 cv.imshow("Original Image", img)
5 cv.imshow("Flipped Image", flipped_image)
6 cv.waitKey(0)
```

**OUTPUT**

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

The output shows two side-by-side windows:

- Original Image**: A portrait of a man with short dark hair and a goatee, wearing an orange shirt, against a green background.
- Flipped Image**: The same portrait, but vertically flipped (rotated 180 degrees).

# OUTPUT

The screenshot displays the Visual Studio Code interface with a Python script named `flip.py` open. The script uses the `cv2` library to read an image, flip it horizontally, and display both the original and the flipped versions. The Explorer sidebar on the left shows a project structure with various image processing scripts. The Output window at the bottom right shows the execution of the script, and two image windows, 'Original Image' and 'Flipped Image', are open side-by-side.

**EXPLORER**

- OPEN EDITORS 3 unsaved
  - dilated.py
  - eroded.py
  - image\_resize.py
  - cropped\_image.py
  - crop\_image.py
  - img\_translate.py
  - img\_rotate.py
  - img\_rotate2.py
  - flip.py
- cv
  - blur.py
  - conversion2gray.py
  - crop\_image.py
  - cropped\_image.py
  - dilated.py
  - edge\_cascade.py
  - eroded.py
  - flip.py
  - Image\_reading.py
  - image\_rescaling.py
  - image\_resize.py
  - img\_rotate.py
  - img\_rotate2.py
  - img\_translate.py
  - more\_blur.py
- OUTLINE
- TIMELINE

**flip.py**

```
1 import cv2 as cv
2 img = cv.imread("Images/seun.jpg")
3 flipped_image = cv.flip(img, 1)
4 cv.imshow("Original Image", img)
5 cv.imshow("Flipped Image", flipped_image)
6 cv.waitKey(0)
```

**Original Image**

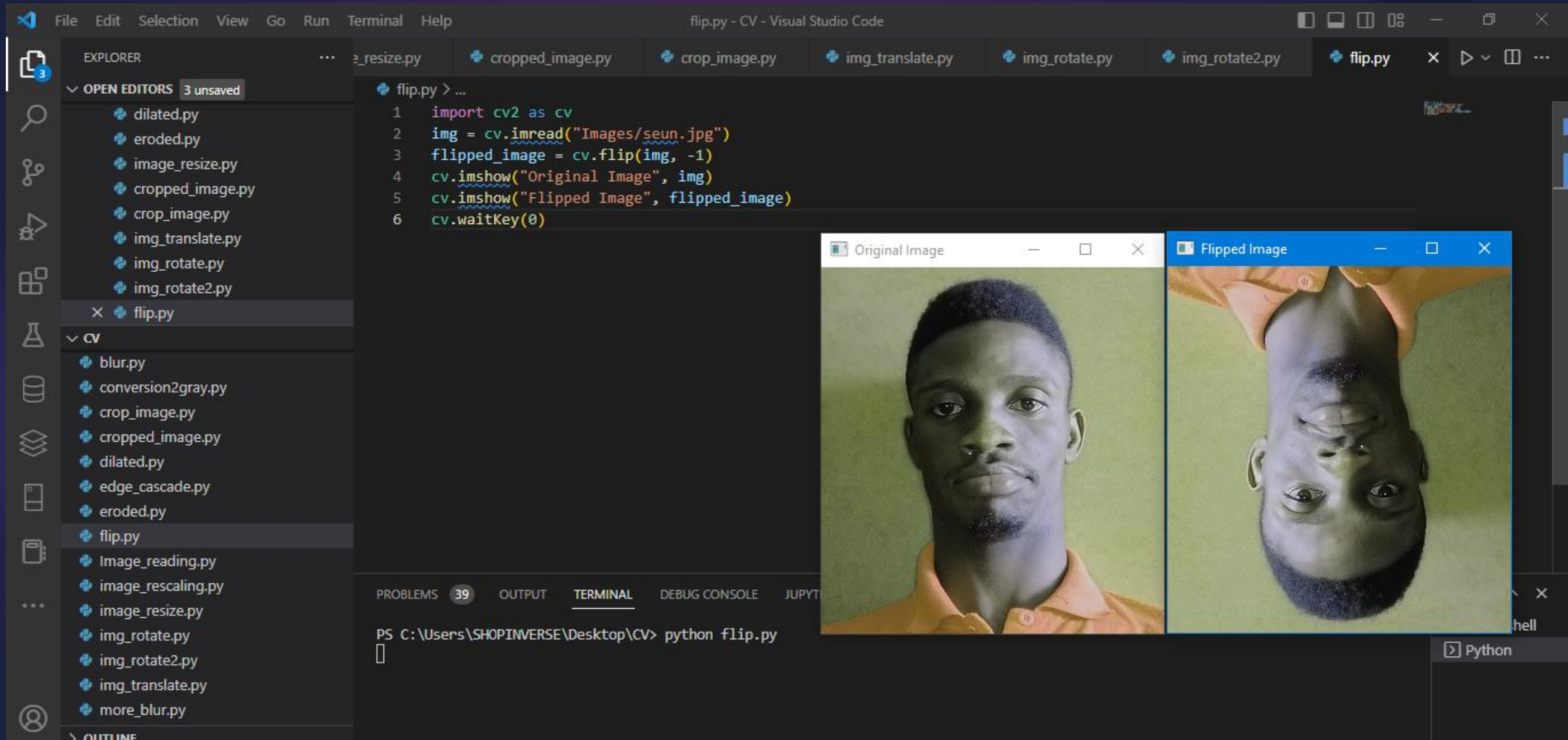
**Flipped Image**

**TERMINAL**

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



# OUTPUT



# IMAGE TRANSFORMATION IN OPENCV

Cont'd

**NOTE:** Image cropping and Image resizing were explained in the previous lecture (part two)



# READING OF IMAGE/VIDEO FROM WEBCAM

## Syntax for reading images/videos from webcam

```
import cv2 as cv
captured_image = cv.VideoCapture(0)
while True:
    isTrue, frame = captured_image.read()
    cv.imshow("frame", frame)
    if cv.waitKey(1) == ord('r'):
        break
captured_image.release()
cv.destroyAllWindows()
```

## CODE INTERPRETATION

**Line 1:** Import the OpenCV module

**Line 2:** Create an instance of the VideoCapture to read the camera event

**Line 3:** Check if the frame can be read

**Line 4:** Displays the image

**Line 5:** To specify the wait key and key to close the frame.

**Line 6:** Used to release the camera so that other resources can use it.

**Line 7:** Used terminate all windows

# OUTPUT

The image shows a Visual Studio Code interface with a Python script named `read2.py` open. The script uses OpenCV to capture video frames. A separate window titled `frame` displays the captured image of a person's face. The terminal at the bottom shows the command to run the script.

**Visual Studio Code Explorer:**

- OPEN EDITORS (4 unsaved)
  - `img_translate.py`
  - `img_rotate.py`
  - `img_rotate2.py`
  - `flip.py`
  - `read.py`
  - `read2.py` (selected)
- CV
  - `eroded.py`
  - `flip.py`
  - `Image_reading.py`
  - `image_rescaling.py`
  - `image_resize.py`
  - `img_rotate.py`
  - `img_rotate2.py`
  - `img_translate.py`
  - `more_blur.py`
  - `opencv_frame_10.png`
  - `read.py`
  - `read2.py` (selected)
  - `video_reading.py`
  - `video_rescaling.py`

**read2.py Code:**

```
1 import cv2 as cv
2 captured_image = cv.VideoCapture(0)
3 while True:
4     isTrue, frame = captured_image.read()
5     cv.imshow("frame", frame)
6     if cv.waitKey(1) == ord('r'):
7         break
8
9 captured_image.release()
10 cv.destroyAllWindows()
```

**Terminal Output:**

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python read2.py
```

**frame Window:** Displays a video frame of a person's face.

**Status Bar:** 0 0 46 tabnine starter Partial support, click to resolve Ln 10, Col 23 Spaces: 4

**Taskbar:** Windows taskbar with search bar and various application icons. System tray shows 27°C, ENG INTL, and 03:19 17/02/2023.



**THANKS FOR VIEWING**

**More tutorials will be covered in part four**