



ORMEDIAN RESEARCH INSTITUTE

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

By:

ADEKOLA OLUWASEUN O.

February 22, 2023

TABLE OF CONTENT

- ❑ WHAT COUNTOURS ARE
- ❑ FINDING NUMBER OF CONTOURS IN AN IMAGE
- ❑ USING CANNY EDGE DETECTOR APPROACH IN IDENTIFYING CONTOURS
- ❑ REDUCING NUMBER OF CONTOURS FOUND
- ❑ USING THRESHOLDING APPROACH IN IDENTIFYING CONTOURS.
- ❑ DRAWING OF CONTOURS ON A BLANK IMAGE.

CONTOURS

WHAT CONTOURS ARE

Contours could be defined as the lines joining all points along the boundary of an image that have the same intensity and colour. This means that when we draw all the points on the boundary of an object, we will get a contour. OpenCV has a functionality that makes it easy to find contours present in any image. It generally provides a function to find a contour, and a function to draw such a contour, i.e. `findContours()` and `drawContours()` methods, respectively.

CONTOURS

Cont'd

APPLICATIONS OF CONTOURS IN COMPUTER VISION

There are several applications of contours in computer vision. Some are listed below.

- ▶ Motion detection
- ▶ Background/Foreground Segmentation
- ▶ Unattended Object Detection
- ▶ Object Recognition
- ▶ Shape Analysis

CONTOURS

Cont'd

In the process of detecting contours, the algorithm to use is usually specified. There are two different approximation algorithms commonly used for contour detection in computer vision because we need to know the kind of contours that we want. These approximation algorithms are:

- ▶ **CHAIN_APPROX_NONE:** This returns all the contours that are found in an image. It does not do any compression. It returns all the connects between the points of lines in that image.
- ▶ **CHAIN_APPROX_SIMPLE:** This basically compresses all the contours that are returned. Here, lines are compressed into two end points.

These approximation method uses the Ramer-Douglas-Peucker algorithm, which aims at reducing the number of points in a curve that is approximated by a series of points. Generally, the CHAIN_APPROX_SIMPLE is mostly preferred.

CONTOURS

Cont'd

STEPS REQUIRED FOR DETECTING AND DRAWING CONTOURS IN OpenCV

- i. Read and Convert the Image to Grayscale Format
- ii. Apply Binary Thresholding or Canny Edge Detection to the Grayscale image obtained in step (i)
- iii. Find the Contours
- iv. Finally, Draw the Contours on the original RGB image.

CONTOURS

Cont'd

METHOD 1: Using Canny Edge detection in the process of finding contours

```
import cv2 as cv
img = cv.imread('input image path')
gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
canny = cv.Canny(gray, min_intensity, max_intensity)
contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_NONE)
print ("There are { } contours in this image".format(len(contours)))
```

CONTOURS

Cont'd

Code interpretation

Line 1: Import the OpenCV module

Line 2: Read the input image

Line 3: Convert the image from RGB to grayscale format.

Line 4: Apply Canny Edge detector to the Grayscale image

Line 5: Find the contours and hierarchies by specifying the canny edges, retrieval mode, and the approximation algorithm. What the findContours() method does is that it essentially returns the contours and the hierarchies.

Line 6: Print the number of contours (since the contours in this case is a list, we can apply len() method on it)

NB: There are different retrieval modes: `cv.RETR_TREE` retrieves all hierarchical contours, `cv.RETR_EXTERNAL` retrieves all external contours i.e. contours outside, `cv.RETR_LIST` retrieves all the contours in an image.

NB: APPROXIMATION algorithm could be set to `cv.CHAIN_APPROX_SIMPLE` which compresses all the contours that are returned as endpoints only While `cv.CHAIN_APPROX_NONE` does nothing but returns all the contours in the image.

CONTOURS

Cont'd

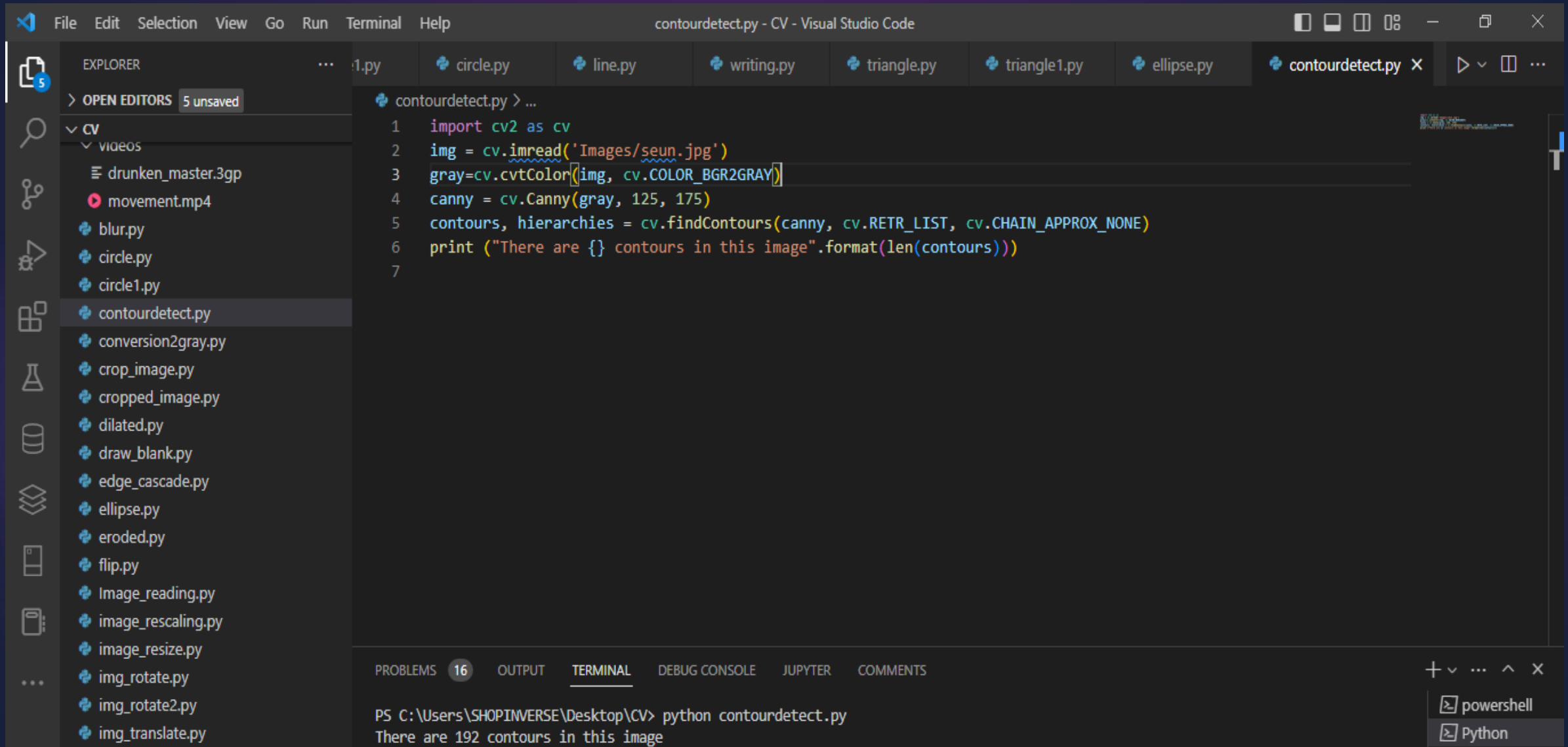
Example 1: Find all the contours in the image “seun.jpg” by using the two different approximation algorithms.

SOLUTION

```
import cv2 as cv
img = cv.imread('Images/seun.jpg')
gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
canny = cv.Canny(gray, 125, 175)
contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_NONE)
print ("There are { } contours in this image".format(len(contours)))
```

From the output shown on the next slide, we could see that the number of contours in this image is 192.

OUTPUT (With cv.CHAIN_APPROX_NONE)



The screenshot displays the Visual Studio Code interface with a Python script named `contourdetect.py` open. The script uses OpenCV to read an image, convert it to grayscale, and detect contours using `cv.CHAIN_APPROX_NONE`. The terminal output shows the execution of the script, resulting in the message: "There are 192 contours in this image".

EXPLORER

- CV
 - videos
 - drunken_master.3gp
 - movement.mp4
 - blur.py
 - circle.py
 - circle1.py
 - contourdetect.py
 - conversion2gray.py
 - crop_image.py
 - cropped_image.py
 - dilated.py
 - draw_blank.py
 - edge_cascade.py
 - ellipse.py
 - eroded.py
 - flip.py
 - Image_reading.py
 - image_rescaling.py
 - image_resize.py
 - img_rotate.py
 - img_rotate2.py
 - img_translate.py

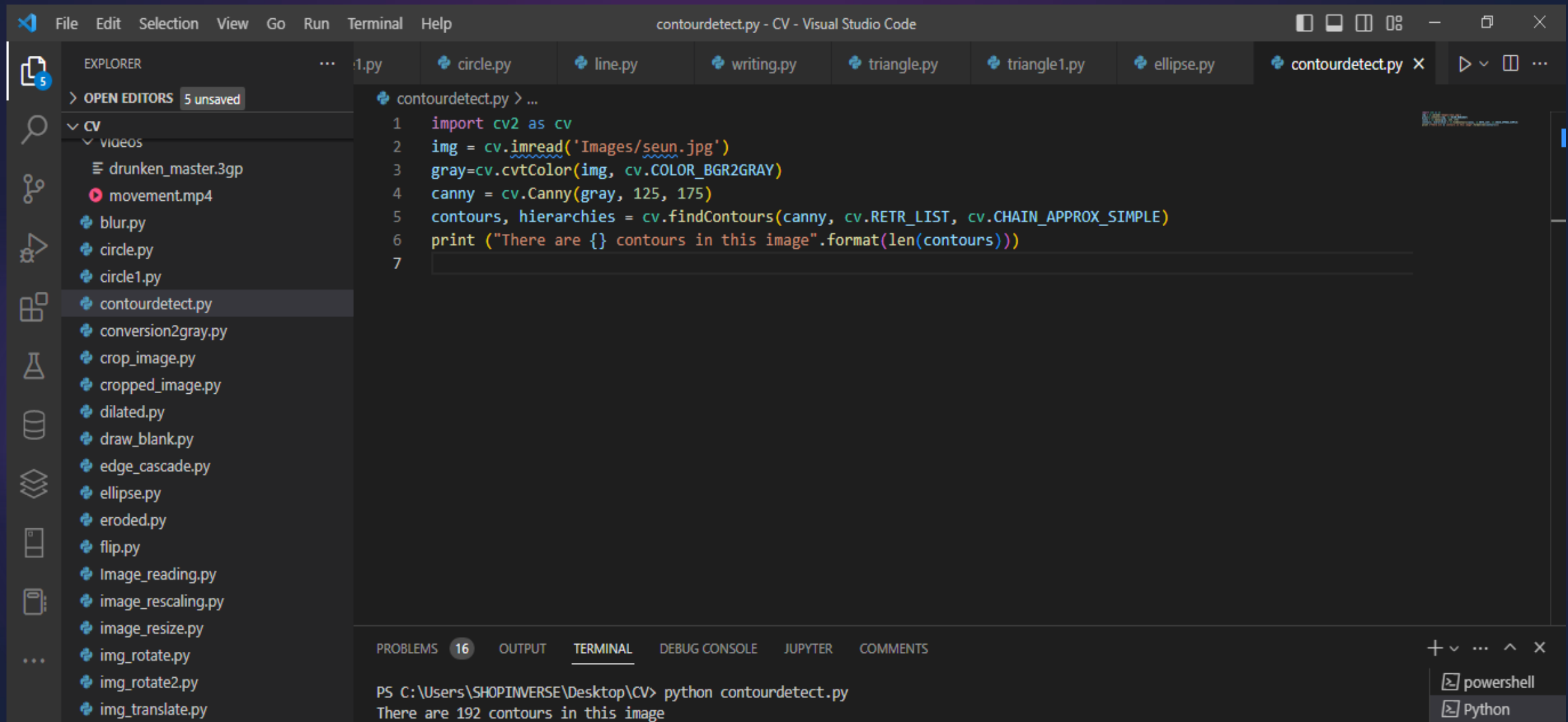
contourdetect.py

```
1 import cv2 as cv
2 img = cv.imread('Images/seun.jpg')
3 gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
4 canny = cv.Canny(gray, 125, 175)
5 contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_NONE)
6 print ("There are {} contours in this image".format(len(contours)))
7
```

TERMINAL

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python contourdetect.py
There are 192 contours in this image
```

OUTPUT (With cv.CHAIN_APPROX_SIMPLE)



The screenshot displays the Visual Studio Code interface with a Python script named `contourdetect.py` open. The script uses OpenCV to read an image, convert it to grayscale, apply Canny edge detection, and find contours using `cv.CHAIN_APPROX_SIMPLE`. The terminal output shows that 192 contours were found in the image.

```
contourdetect.py > ...
1  import cv2 as cv
2  img = cv.imread('Images/seun.jpg')
3  gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
4  canny = cv.Canny(gray, 125, 175)
5  contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
6  print ("There are {} contours in this image".format(len(contours)))
7
```

PROBLEMS 16 OUTPUT **TERMINAL** DEBUG CONSOLE JUPYTER COMMENTS

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python contourdetect.py
There are 192 contours in this image_
```

Terminal session options: powershell, Python

CONTOURS

Cont'd

Reducing the number of the contours

Although there might be cases where we feel the number of contours obtained is very high and we may want to have a more reduced number. In this scenario, it is pretty good to blur the image before finding edges and contour of such an image. To do this with the previous result, further modification is required.

Result shown on the next slide implies that there is a significant reduction in the number of contours after the application of the blurring effect.

The number of contours has been reduced to 55 from 192.

OUTPUT

The image shows a Visual Studio Code window with the file `contreduced.py` open. The Explorer sidebar on the left shows a project structure with folders `Images` and `videos`, and a list of Python files including `seun.jpg`, `drunken_master.3gp`, `movement.mp4`, and several image processing scripts like `blur.py`, `circle.py`, `circle1.py`, `contourdetect.py`, `contreduced.py`, `conversion2gray.py`, `crop_image.py`, `cropped_image.py`, `dilated.py`, `draw_blank.py`, `edge_cascade.py`, `ellipse.py`, `eroded.py`, `flip.py`, `Image_reading.py`, `image_rescaling.py`, and `image_resize.py`.

The `contreduced.py` file contains the following Python code:

```
1 import cv2 as cv
2 img = cv.imread('Images/seun.jpg')
3 gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
4 blur = cv.GaussianBlur(gray, (3, 3), cv.BORDER_DEFAULT)
5 canny = cv.Canny(blur, 125, 175)
6 contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
7 print ("There are {} contours in this image".format(len(contours)))
```

The bottom of the window shows the **TERMINAL** panel with the command `python contreduced.py` executed in a PowerShell session, resulting in the output: `There are 55 contours in this image`.

CONTOURS

Cont'd

Contour Detection Using the Simple Thresholding Approach (Binarization)

This approach uses some threshold value to find the contours from the gray image simply by binarizing the image. The intensity values set in this method are binarized; for example, if the intensity is less than the minimum intensity defined in the syntax, such intensity is set to black, but if it is above 125, it is set to white. Meaning, binarizing of image has been performed. But thresholding works well for images with few variations in colours.

SYNTAX

```
import cv2 as cv
img = cv.imread('input image path')
gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
retr, thresh = cv.threshold(gray, min_intensity, max_intensity, cv.THRESH_BINARY)
contours, hierarchies = cv.findContours(thresh, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
print ("There are { } contours in this image".format(len(contours)))
```

CONTOURS

Cont'd

Example 1: Use the thresholding method to find contours in the image “seun.jpg”

SOLUTION

```
import cv2 as cv
img = cv.imread('Images/seun.jpg')
gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
retr, thresh = cv.threshold(gray, 125, 255, cv.THRESH_BINARY)
contours, hierarchies = cv.findContours(thresh, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
print ("There are { } contours in this image".format(len(contours)))
```

The output shows that there are 183 contours in this image.

OUTPUT

The image shows a Visual Studio Code window with the title "Threshcont.py - CV - Visual Studio Code". The Explorer sidebar on the left shows a folder named "CV" containing various Python files, with "Threshcont.py" selected. The main editor displays the code for "Threshcont.py":

```
1 import cv2 as cv
2 img = cv.imread('Images/seun.jpg')
3 gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
4 retr, thresh = cv.threshold(gray, 125, 255, cv.THRESH_BINARY)
5 contours, hierarchies = cv.findContours(thresh, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
6 print ("There are {} contours in this image".format(len(contours)))
```

The bottom status bar shows the "TERMINAL" tab with the following output:

```
PS C:\Users\SHOPINVERSE\Desktop\CV> python threshcont.py
There are 183 contours in this image
```

On the right side of the terminal, there are buttons for "powershell" and "Python".

CONTOURS

Cont'd

DRAWING OF CONTOURS

In the computer vision concept, OpenCV has the functionality that allows us to draw out the contours of an image on a blank image for ease of visualization. By drawing out the contours, it enables us to know the kind of contours that an image has.

The `drawContours()` method has to be used to get this done.

SYNTAX

```
cv2.drawContours(blank, contours, contourIndex, colour, thickness)
```

CONTOURS

Cont'd

Example 1: Draw all the contours of the image named “seun.jpg” on a blank image of the same size as the original image.

SOLUTION

```
import cv2 as cv
import numpy as np
img = cv.imread('Images/seun.jpg')
gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
canny = cv.Canny(gray, 125, 175)
contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
blank = np.zeros(img.shape, dtype = 'uint8')
cv.drawContours(blank, contours, -1, (0, 0, 225), 1)
cv.imshow('Canny edges', canny)
cv.imshow("Contours Drawn.", blank)
cv.waitKey(0)
```

OUTPUT

Visual Studio Code interface showing the execution of a Python script for contour detection.

EXPLORER

- CV
 - blur.py
 - circle.py
 - circle1.py
 - contourdetect.py
 - contreduced.py
 - conversion2gray.py
 - crop_image.py
 - cropped_image.py
 - dilated.py
 - draw_blank.py
 - drawContours.py
 - edge_cascade.py
 - ellipse.py
 - eroded.py
 - flip.py
 - Image_reading.py
 - image_rescaling.py
 - image_resize.py
 - img_rotate.py
 - img_rotate2.py
 - img_translate.py
 - line.py
 - more_blur.py
 - opencv_frame_10.png

drawContours.py

```
1 import cv2 as cv
2 import numpy as np
3 img = cv.imread('Images/seun.jpg')
4 gray=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
5 canny = cv.Canny(gray, 125, 175)
6 contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
7 blank = np.zeros(img.shape, dtype = 'uint8')
8 cv.drawContours(blank, contours, -1, (0, 0, 225), 1)
9 cv.imshow('Canny edges', canny)
10 cv.imshow("Contours Drawn.", blank)
11 cv.waitKey(0)
```

TERMINAL

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

Output Windows:

- Canny edges:** Displays the detected edges of the image in white on a black background.
- Contours Drawn.:** Displays the detected contours of the image in red on a black background.



THANKS FOR VIEWING

More tutorials will be covered in part six