cs512 Assignment 2

Jiranun Jiratrakanvong
Department of Computer Science
Illinois Institute of Technology

October 20, 2015

Abstract

The objective of this assignment is to learn about feature detection with OpenCV by conducting a program using python as a programming language. The program of this assignment will perform ellipse detection and ellipse fitting.

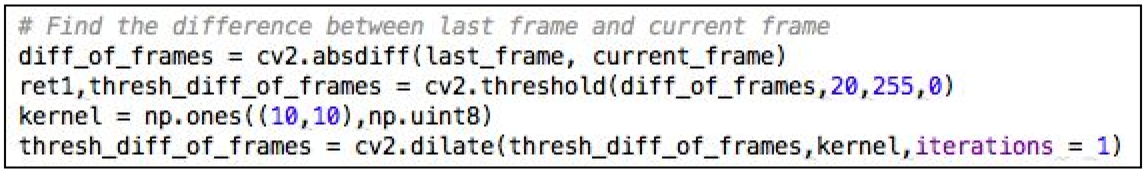
Problem Statement

The program from this assignment need to perform ellipse detection and ellipse fitting using OpenCV**.** The program capture images directly from the camera. The program will detect a user's hand, and try to fit the ellipses on fingers

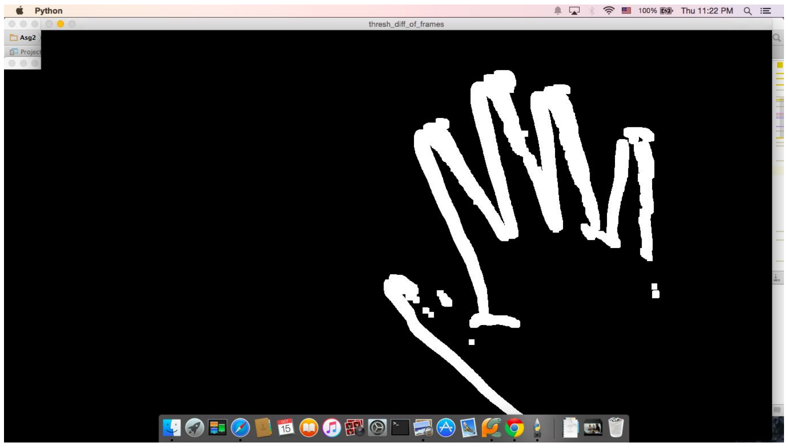
Purpose Solution

The program was separated into 2 parts, hand moving detection, and ellipse fitting. The first part is **Asg2_part1.py**, and the second part is **Asg2_Part2.py.**

For the first part, **Asg2_part1.py**, this program will continuously capture frames directly from the camera, and find the edge of moving hand. Every captured frame will be flipped, converted to grayscale, and smoothed by gaussian filter. The difference between last frame and current frame will be performed, and thresholded. After that, it will be dilated by 10x10 kernel in order to make the location larger.



```
# Find the difference between last frame and current frame
diff_of_frames = cv2.absdiff(last_frame, current_frame)
ret1,thresh_diff_of_frames = cv2.threshold(diff_of_frames,20,255,0)
kernel = np.ones((10,10),np.uint8)
thresh_diff_of_frames = cv2.dilate(thresh_diff_of_frames,kernel,iterations = 1)
```

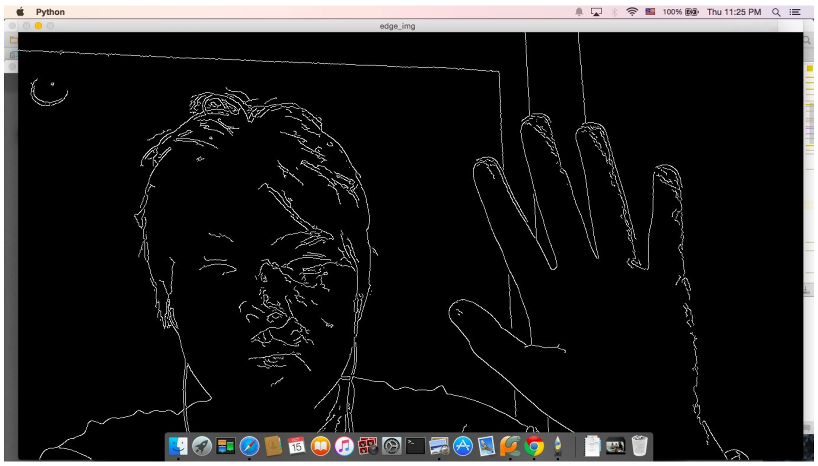


Difference between two frames

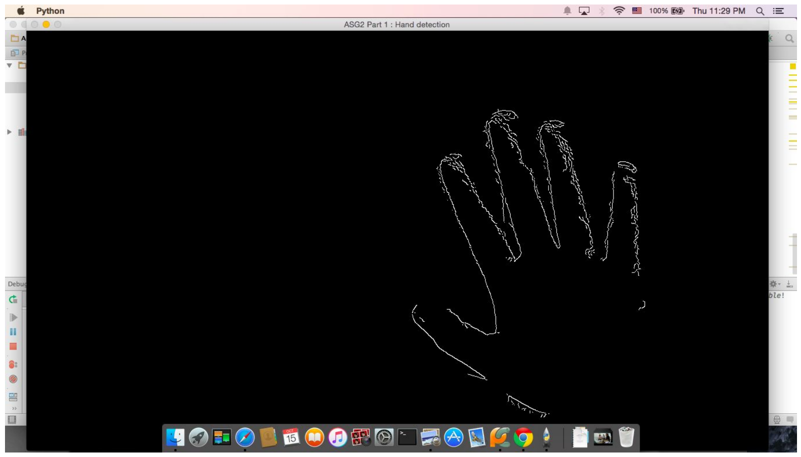
Then, the program will perform canny edge detection on the current frame by using high and low threshold as 40 and 10. The intersection of the the difference between frames and the canny edge image, marked_edge_img will be performed by cv2.bitwise_and.

```
# Find the edge of image by using canny detection
edge_img = cv2.Canny(current_frame, 40, 10)

# Find the intersection of the difference between frames and the edge (marked_edge_img)
marked_edge_img = cv2.bitwise_and(thresh_diff_of_frames, edge_img)
```

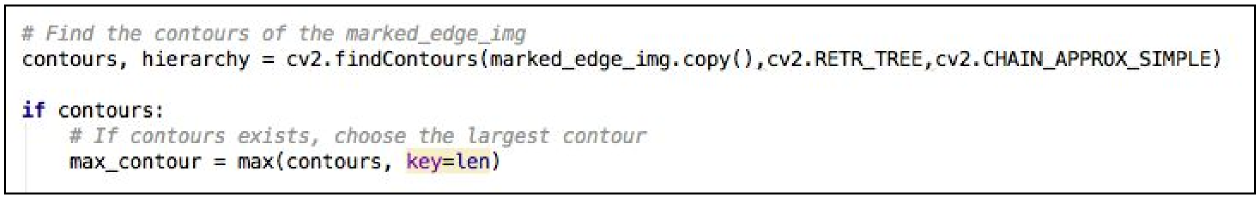


Canny Edge Detection



Marked Edge Image

After the marked_edge_img performed, the program will find the contours of it by using cv2.findContours. The return value of this function will be list of contours in the image. Sometimes there are many contours are found, so the program will choose only the largest contour in the list.



```
# Find the contours of the marked_edge_img
contours, hierarchy = cv2.findContours(marked_edge_img.copy(),cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

if contours:
    # If contours exists, choose the largest contour
    max_contour = max(contours, key=len)
```

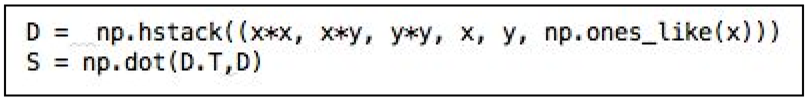
After the largest contour found, the program was supposed to do the ellipse detection and ellipse fitting on the contour, but the result will not be accurate because the largest contour is not the contour of the whole hand. It happened because the marked edge was disconnected by the process before. Therefore, the program from ASG2_Part1.py will perform only hand moving detection by marked edge detection, and largest contour. After program run, there are two window shown, **ASG2 Part 1 : Hand detection** and **ASG2 Part 1 : Largest contour**

The second part of the program is **Asg2_Part2.py**. This part will perform ellipse detection and ellipse fitting on the specified image. The image name will be specified via command line. If the file name has not been specified, or the file does not exist, the program will read “ellipse.jpg” instead by default.

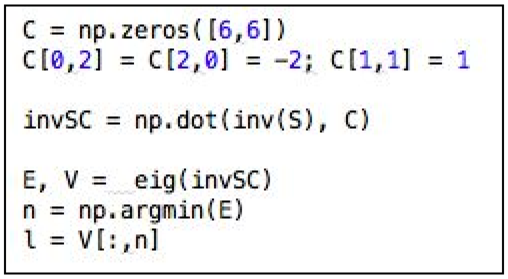
After the program run, image will be read, converted to grayscale and thresholded. Then, the program will find the contour of the image by cv2.findContours. The contour points will be used to create a new data matrix (D). Let $S = D^TD$, find the eigenvalues and eigenvectors of $S^{-1}C$

$$\mathbf{D} = \begin{pmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i^2 & x_iy_i & y_i^2 & x_i & y_i & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^2 & x_Ny_N & y_N^2 & x_N & y_N & 1 \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

From the conic formula Ax2+Bxy+Cy2+Dx+Ey+F = 0 ,and constraint of ellipse that B2-4AC < 0, the program will find [A,B,C,D,E,F] from the eigenvector belonging to the smallest eigenvalue.



```
D =  np.hstack((x*x, x*y, y*y, x, y, np.ones_like(x)))
S = np.dot(D.T,D)
```

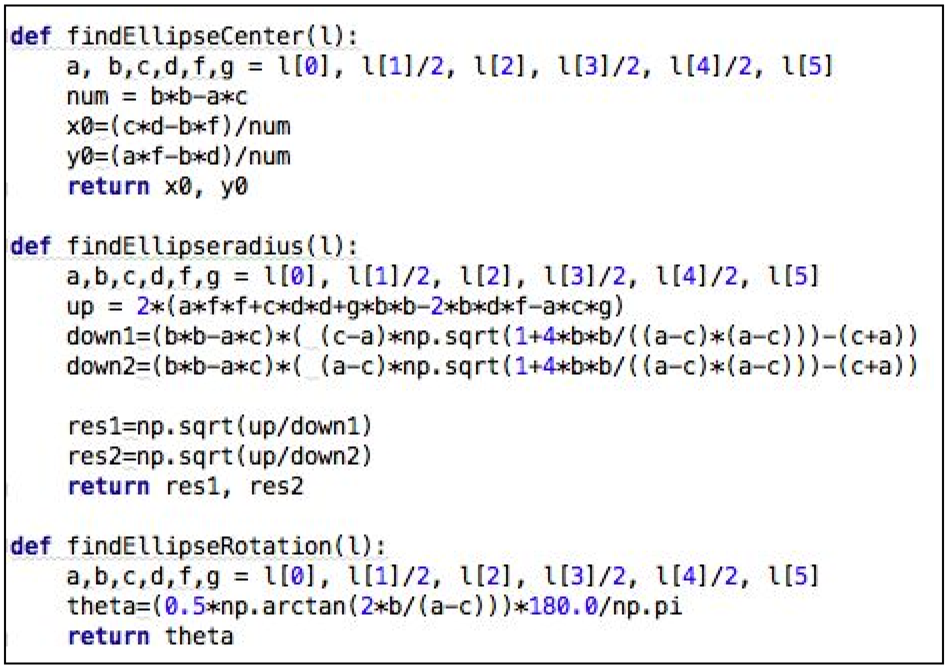


```
C = np.zeros([6,6])
C[0,2] = C[2,0] = -2; C[1,1] = 1

invSC = np.dot(inv(S), C)

E, V =  eig(invSC)
n = np.argmin(E)
l = V[:,n]
```

Next, when the program gains [A,B,C,D,E,F] from the process above, the center, radius, and angle of the ellipse will be calculated as below



```
def findEllipseCenter(l):
    a, b,c,d,f,g = l[0], l[1]/2, l[2], l[3]/2, l[4]/2, l[5]
    num = b*b-a*c
    x0=(c*d-b*f)/num
    y0=(a*f-b*d)/num
    return x0, y0

def findEllipseradius(l):
    a,b,c,d,f,g = l[0], l[1]/2, l[2], l[3]/2, l[4]/2, l[5]
    up = 2*(a*f*f+c*d*d+g*b*b-2*b*d*f-a*c*g)
    down1=(b*b-a*c)*( (c-a)*np.sqrt(1+4*b*b/((a-c)*(a-c)))-(c+a))
    down2=(b*b-a*c)*( (a-c)*np.sqrt(1+4*b*b/((a-c)*(a-c)))-(c+a))

    res1=np.sqrt(up/down1)
    res2=np.sqrt(up/down2)
    return res1, res2

def findEllipseRotation(l):
    a,b,c,d,f,g = l[0], l[1]/2, l[2], l[3]/2, l[4]/2, l[5]
    theta=(0.5*np.arctan(2*b/(a-c)))*180.0/np.pi
    return theta
```

Implementation Details

Problem and Solution

- The contour of the hand was disconnected because of the edge detection and the difference between frames, so it's hard to find the whole contour of the hand. The solution is dilating the difference between frames. It solved the problem, but the problem still occurs sometimes especially when the hand and background color are quite similar.
- The curvature is calculated by | Pi+1 - 2Pi + Pi-1 |. After the calculation the curvature of each point will be positive value only, so the program cannot distinguish between finger tips and finger bridges. Consequently, the program cannot separate fingers contour into groups. The solution is separating program into two parts. First is the hand detection, and the second part is ellipse fitting.
- After [A,B,C,D,E,F] of the conic function found, the OpenCV cannot draw the ellipse by these parameters directly because the OpenCV function requires center, axes, and angle. The solution is trying to find the center, axes, and angle by those parameter first.
- [A,B,C,D,E,F] of the conic function was incorrect because the points of contour were very large compared to the variation between the values. The solution is dividing all points by a factor and translate all points by mean value of all points before doing the ellipse fitting. After doing the ellipse fitting, all contour points, center, and axes are multiplied by the factor and translate back to the original location again.

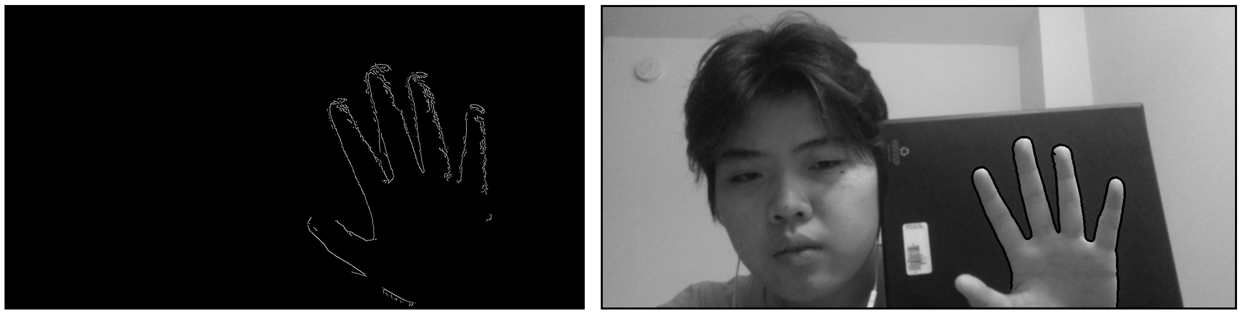
Instructions for using the program

- **Asg2_part1.py**
  - Running via commandline : *python Asg2_part1.py*
  - The program will show two windows , Movement detection and Largest contour
  - Press <ESC> to quit the program
- **Asg2_Part2.py**
  - Running via commandline : *python Asg2_part2.py [filename.jpg]*
  - Press any key to show the next ellipse if it exists
  - Press <ESC> to quit the program

Results and discussion

- **Asg2_part1.py**

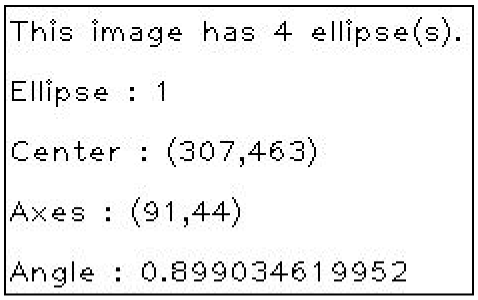
  The program will launch two windows. The first window will show Movement detection while the second window will show Largest contour (black line) as below



Movement detection and largest contour

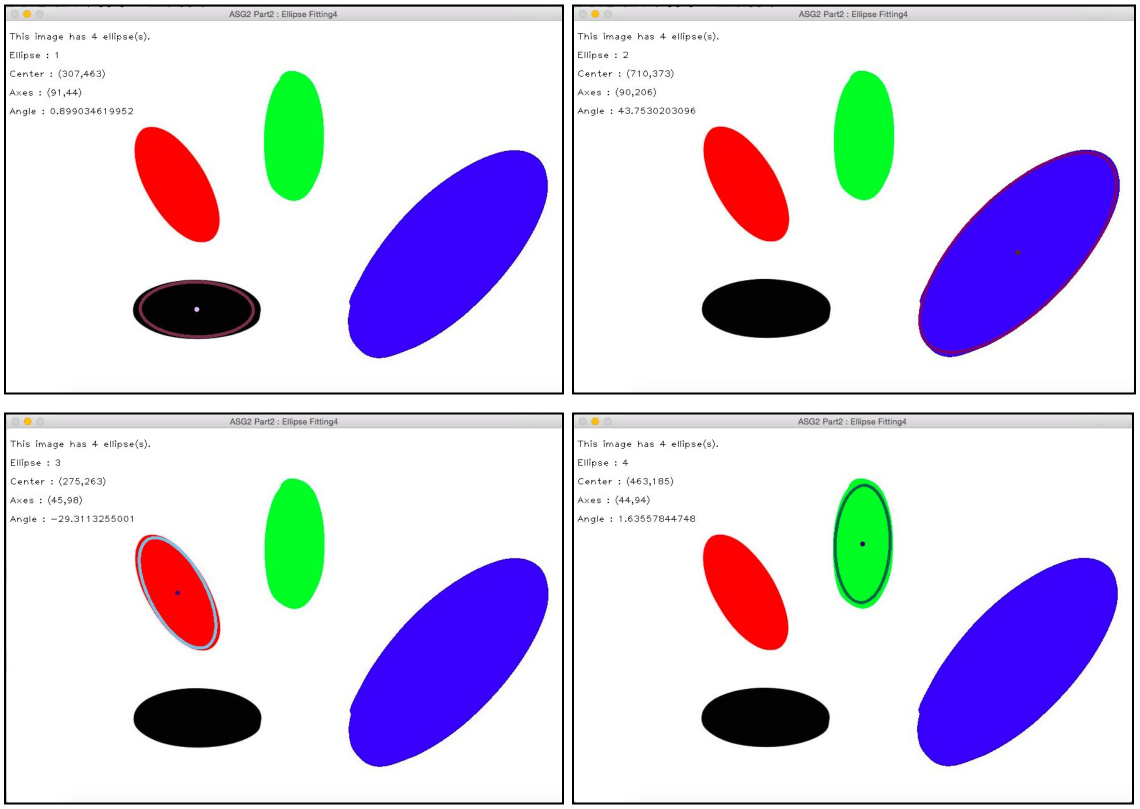
- **Asg2_part2.py**

  The program will do the ellipse fitting in the specified image. It can find multiple ellipses. The program will draw the line of each ellipse and its center by random color, and show the ellipse information including number of ellipses, center, axes, and angle. (Press any key to show next ellipse information.)



```
This image has 4 ellipse(s).
Ellipse : 1
Center : (307,463)
Axes : (91,44)
Angle : 0.899034619952
```

Example of ellipse information



```
This image has 4 ellipse(s).
Ellipse : 1
Center : (307,463)
Axes : (91,44)
Angle : 0.899034619952
```

```
This image has 4 ellipse(s).
Ellipse : 2
Center : (710,373)
Axes : (90,206)
Angle : 43.7530203096
```

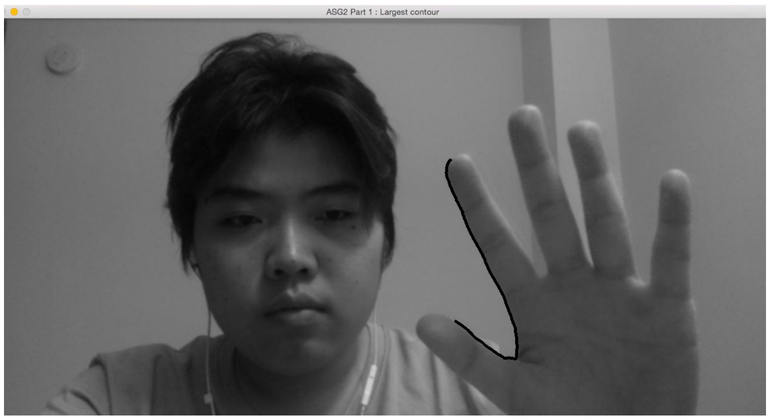
```
This image has 4 ellipse(s).
Ellipse : 3
Center : (275,263)
Axes : (45,98)
Angle : -29.3113255001
```

```
This image has 4 ellipse(s).
Ellipse : 4
Center : (463,185)
Axes : (44,94)
Angle : 1.63557844748
```

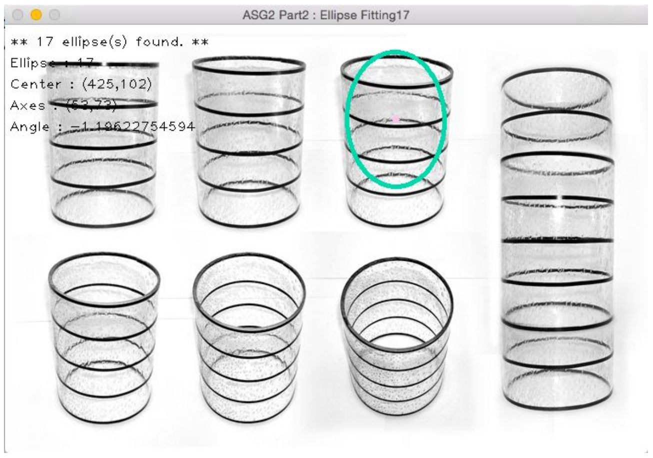
Example of ellipse fitting

Strength and weaknesses

For the first part, if the background (wall) and the hand colors are similar, the contour will be separated into many small contours. As a result, the largest contour will be only small part of hand.



Also, for the second part, if the input image is quite complex image, not all of ellipses will be found. Moreover, some ellipse fittings are incorrect.

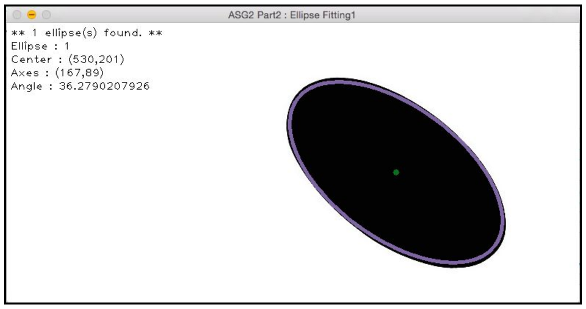


```
** 17 ellipse(s) found. **
Ellipse : 17
Center : (425,102)
Axes : (53,73)
Angle : -1.19622754594
```

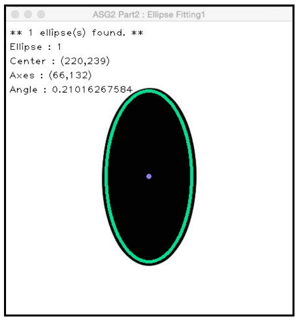
Evaluation

Every operation was evaluated, and tested by a various kind of images.

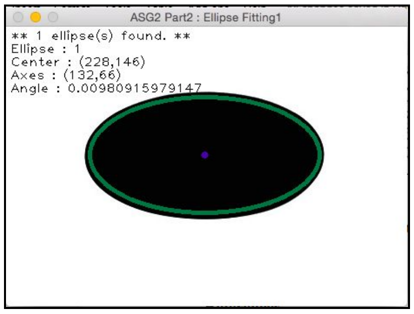
1. Different rotations



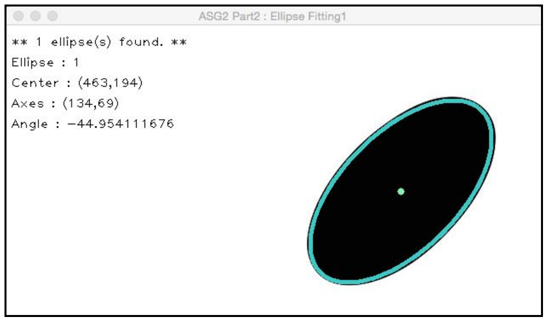
```
** 1 ellipse(s) found. **
Ellipse : 1
Center : (530,201)
Axes : (167,89)
Angle : 36.2790207926
```



```
** 1 ellipse(s) found. **
Ellipse : 1
Center : (220,239)
Axes : (66,132)
Angle : 0.21016267584
```

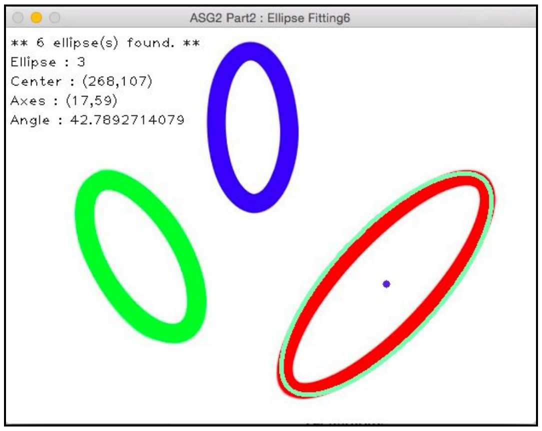


```
** 1 ellipse(s) found. **
Ellipse : 1
Center : (228,146)
Axes : (132,66)
Angle : 0.00980915979147
```

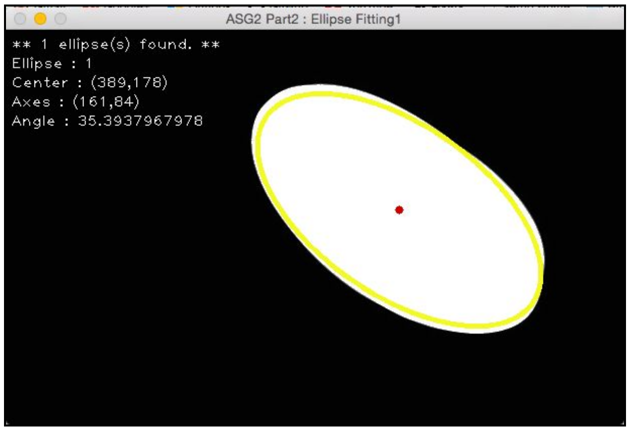


```
** 1 ellipse(s) found. **
Ellipse : 1
Center : (463,194)
Axes : (134,69)
Angle : -44.954111676
```

2. Different color

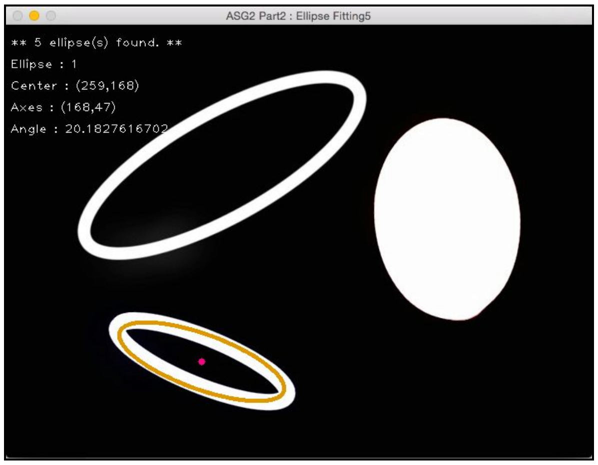


```
** 6 ellipse(s) found. **
Ellipse : 3
Center : (268,107)
Axes : (17,59)
Angle : 42.7892714079
```

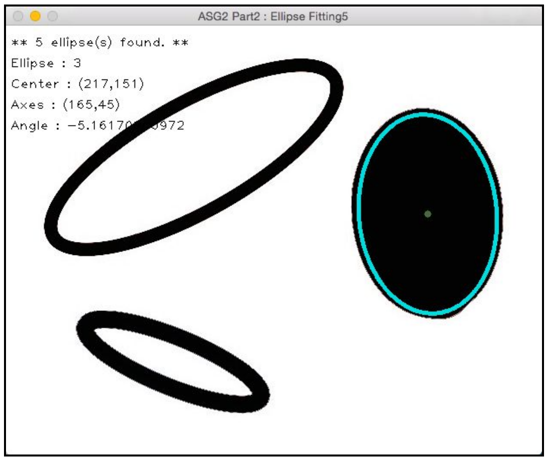


```
** 1 ellipse(s) found. **
Ellipse : 1
Center : (389,178)
Axes : (161,84)
Angle : 35.3937967978
```

3. Multiple ellipses



```
** 5 ellipse(s) found. **
Ellipse : 1
Center : (259,168)
Axes : (168,47)
Angle : 20.1827616702
```



```
** 5 ellipse(s) found. **
Ellipse : 3
Center : (217,151)
Axes : (165,45)
Angle : −5.16170[illegible]0972
```

References

- Gady Agam - *CS512: Computer Vision, Introduction to Python, and Using Python with OpenCV*, Department of Computer Science, Illinois Institute of Technology
- Nicky van Foreest - *Fitting an Ellipse to a Set of Data Points*, http://nicky.vanforeest.com/misc/fitEllipse/fitEllipse.html
- Wolfram Research, Inc. - *Ellipse*, http://mathworld.wolfram.com/Ellipse.html
- Charles F. Van Loan - *Using the Ellipse to Fit and Enclose Data Points*, Department of Computer Science, Cornell University https://www.cs.cornell.edu/cv/OtherPdf/Ellipse.pdf