PA1. Shape detection report - tanukiShapes

010-5136-6003, tlsgusghq@kaist.ac.kr, 신현호(Hyeonho Shin), 20203344, Electrical Engineering

Abstract

My answer to PA1, tanukiShapes is detecting shape with features such as area, perimeter, and the number of vertices. It does not need more library except for numpy, opencv. Also, extremely fast because it exploits on Rule-based method only.

One more thing. For convenience in scoring, I added TA code which can operate my classifier off-the-shelf. There is no trick. It modified some arguments because my classifier does not need some arguments. Mine is not learning-based.

Tested Environment

- Python 3.7.9
- OpenCV 3.4.2
- Numpy 1.19.2

How to use

Same as main.py you gave us.

When checking accuracy with training images (you gave)

1. Pose the main.py with './shapes' folder which has images.
2. In terminal, type 'python main.py'

When checking accuracy with test images (you only have)

1. Pose the main.py with './../ForTA' folder which has images.
2. Un-comment the lines under the line, " Pre-made TA code which I made for convenience."
3. In terminal, type 'python main.py'

Design

Summary

My shape classifier can be decomposed into two steps. In the first step, it extracts features, such as area, periment, number of vertices. In Classify step, using them, it classifies shapes based on pre-determined rule. Let me explain details in next sections.

Preprocess

In this step, features are extracted from images. The extracted features are area, periment of shape in image and the number of vertices.

The number of vertices is calculated using approximation. When an image comes, it finds contours, approximates to a polygon. And return the number of the polygon's vertices.

If given image is ellipse, features of given images are much similar to fitted elipse. So, approximate given shape into ellipse using fitEllipse() function. Because Ellipse fitting needs the sample points of boundary, I applied canny filter in prior. And for removal of noise, I also apply bilateral filter in prior to canny filter.

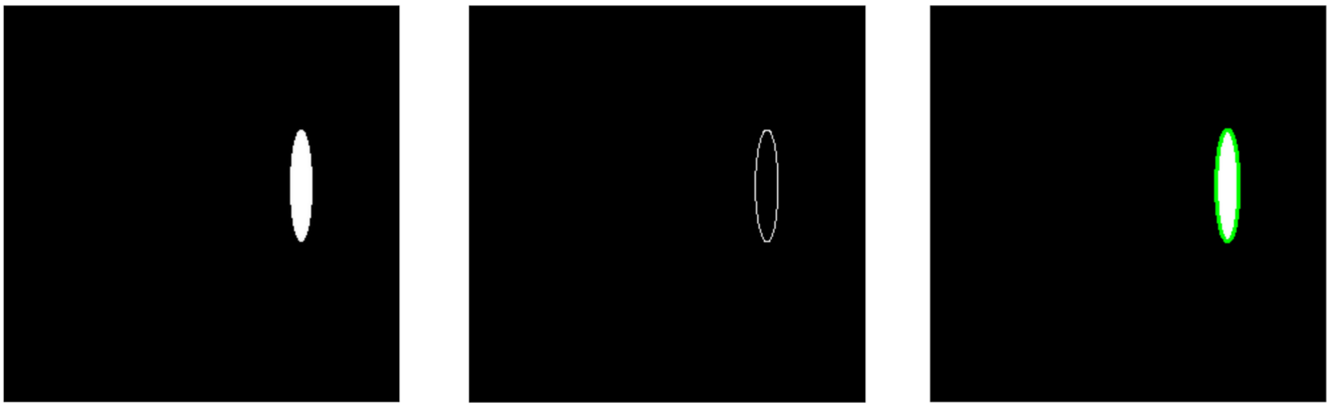


Figure 1 Given image, Boundary of image detected by Canny filter, Fitted ellipse (Green)

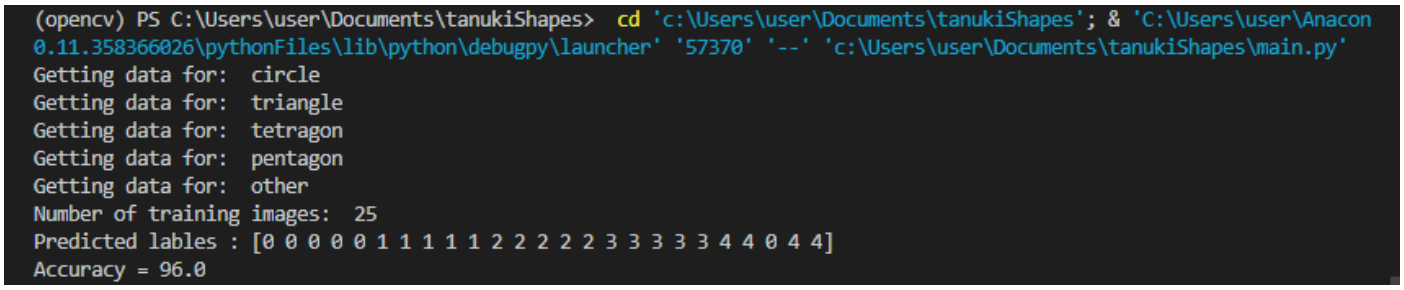
Classify

Using features, we got from the prior steps, we are able to determine classification rule.

At first, classfier checks how many vertices the shape has. According to it, it classifies from triangles to pentagon. However, circles and others cannot be discrimminated with this feature only. So, use other features.

Let assume given image is an ellipse. Then, when fitted the shape as ellipse, it will have much small errors. More specifically, the area and perimeter of given shape is same with them of fitted ellipse. Because we calcuated it the prior step, we can compare it by ratio. Using ratio, my algorithm discrimminates an ellipse from other shapes.

Result in train images



```
(opencv) PS C:\Users\user\Documents\tanukiShapes>  cd 'c:\Users\user\Documents\tanukiShapes'; & 'C:\Users\user\Anacon
0.11.358366026\pythonFiles\lib\python\debugpy\launcher' '57370' '--' 'c:\Users\user\Documents\tanukiShapes\main.py'
Getting data for:  circle
Getting data for:  triangle
Getting data for:  tetragon
Getting data for:  pentagon
Getting data for:  other
Number of training images:  25
Predicted lables : [0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 0 4 4]
Accuracy = 96.0
```

Conclusion

It shows 96.0% accuracy in the given dataset. The mis-classified sample was polygon with many vertices. I guessed the reason of it would be the area and permeters are closed to the ellipses as vertices increasing.

By trial-and-error, I determined the threshold of them as 0.992 < Ratio of area < 1.015, 0.91 < Ratio of perimeter < 1.25. It fits to our trainset with a little margin for robustness.

References

i. 옥수별, Oct-14-2015, “[11편] 이미지 변환 – 리사이징, 이동, 회전, 원근효과”, https://m.blog.naver.com/samsjang/220504966397

ii. OpenCV document, “Contour Features”, https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html

iii. Sergio Canu, Sep-25-2018, “Simple shape detection – Opencv with Python 3”, https://pysource.com/2018/09/25/simple-shape-detection-opencv-with-python-3/

iv. Jrosebr1, “imutils”, https://github.com/jrosebr1/imutils/blob/master/imutils/convenience.py