Aum Amriteshwaryai Namah

19AIE303 - Signal and Image Processing

Assignment 5

Q1. Try out the illustrations in the video (https://www.youtube.com/watch?v=IcBzsP-fvPo) yourself using opencv. For each operation, try to achieve the result as shown in the video. The sample images to be used as input are uploaded:

1) Fig09_5.tif for erosion, dilation, opening and closing;
2) Improve the characters of the text in Fig09_7.tif;
3) Remove the unwanted elements from Fig09_11.tif;
4) Extract the boundary in Fig09_16.tif

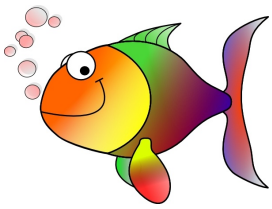
Submit the .py file along with output images.

Q2. **Contours:** The boundary tracing and chain code creation method described in the lecture has the following counterpart in opencv: **cv2.findContours()**. This is based on an algorithm described in:

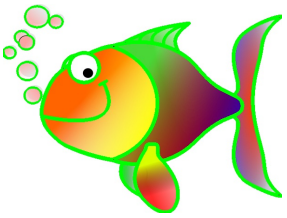
> "Suzuki, S., and Be, K. (1985). Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing 30, 32–46. doi:10.1016/0734-189X(85)90016-7."

Exercise:

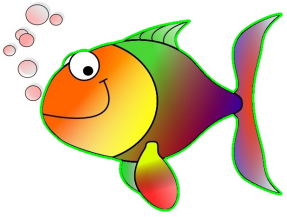
a) Find all the contours in the given image 'bubblingFish.jpg' (use: **cv2.findContours()**) and draw them on the image using **cv2.drawContours().** For better accuracy, apply canny edge detection and use the edge map image to find contours. Save the resulting image



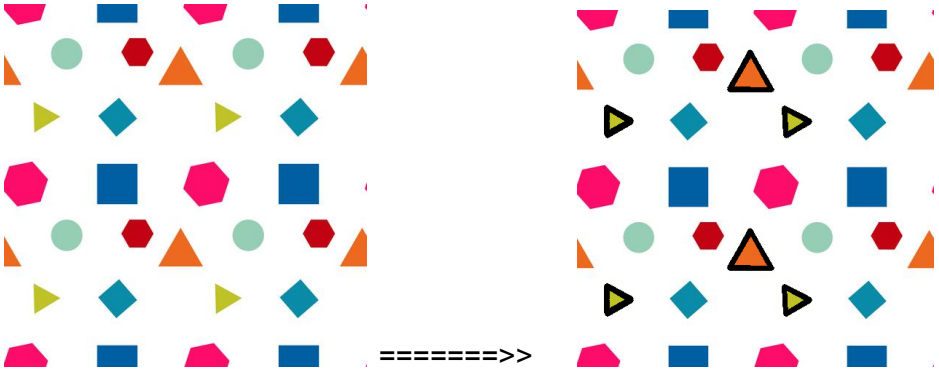
=======>>



b) The above shows all the contours found in the image. In this, determine the **largest contour** (contour with max area) and draw that contour alone to get an outline of the fish. Save this image.

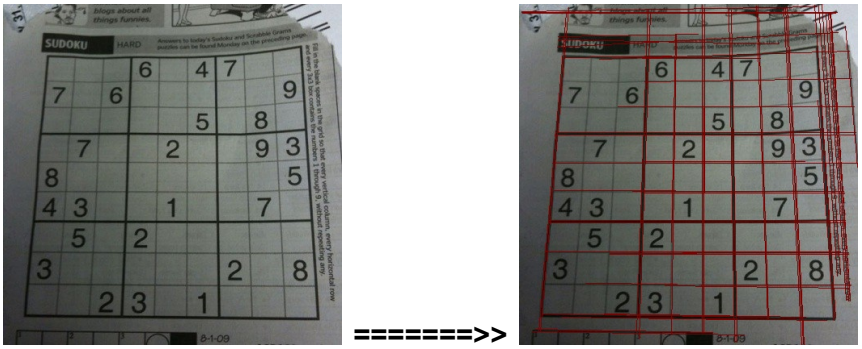


Q3) **Fitting a Polygon**: The contour shapes can be approximated to a polygon with the call **cv2.approxPolyDP()**. Use this to find all the complete triangles in the image 'polygons.png'. Adjust the parameters of this function to get the correct result. **Count** the number of triangles and display the number. Also, **outline the triangles** and save the resulting image.



Q4) **Hough Lines**: The opencv calls **cv2.HoughLines()** and **cv2.HoughLinesP()** both are used to determine hough lines in an image. The later call denotes Probabilistic Hough Transform and is an optimization of Hough Transform for easy computation. Try these calls on the image 'sudoku.jpg', and adjust the parameter to detect maximum possible correct lines. Save the resulting image:

Sample shown below:



What to submit:

1. **A single python file** containing **code and comments** for all the questions. Demarcate both questions using comments
2. **Output images** pasted in a document (word or pdf)

Pls avoid submitting .zip, .tar.gz formats etc

Some references:
https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_features/py_contour_features.html

https://theailearner.com/tag/suzuki-contour-algorithm-opencv/
https://www.geeksforgeeks.org/python-detect-polygons-in-an-image-using-opencv/
https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/