ECE5554 SU22 - Prof. Jones – HW 4

Due Wednesday, August 3, 2022 – 11:59 PM via Canvas

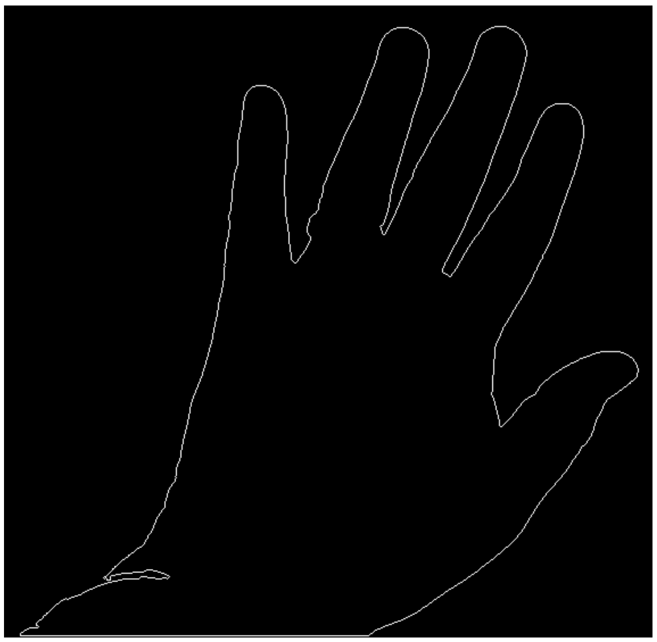
In this assignment you will implement and demonstrate three algorithms we have discussed: the Pavlidis contour extractor, the Gauss area estimation method and Discrete Curve Evolution. I am giving you two images to operate on: "hand0.png" and "US.png". They are in the "Image Files" section of the "Files" page in Canvas.

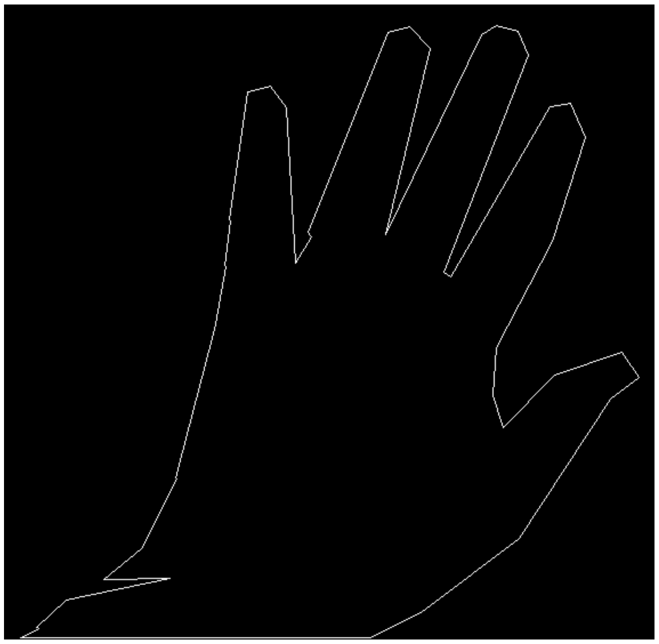
You are to write and test a Python/OpenCV program that will do the following:

1. For each of the images I have supplied:
    a. Load the image and convert to grayscale.
    b. Convert to a binary image using Otsu's thresholding method (you may use cv2.thresold()). Note that I want the hand and the continental US to be foreground (white).
    c. Since the object may, in general, touch the edge of the binary image and this can be complicated to handle in the contour tracing, fill the first and last rows and the first and last columns of the binary image with black. Thus, we have put a one-pixel black boundary around the remainder of the image.
    d. Write out the binarized image to a file.
    e. Find a point on the edge of the object. You can assume that the edge is the first white pixel on the line halfway down the image.
    f. Call your own Pavlidis function to extract the contour of the object.
    g. Call the function that I have supplied to calculate the area of the object from the image.
    h. Call your own Gauss area estimation function to compute the estimated area from the contour points (not the image).
    i. Print (to the console) the filename, the number of points in the contour, the actual area and the Gauss' estimate of the area.
    j. Do the following eight times:
        i. Np is the number of points currently in the contour; do the following np/2 times:
            1. Call your function OnePassDCE (see below) to remove one point from the contour.
               (the result is to remove half of the points from the contour).
        ii. Create an image consisting of the remaining contour points, connected by lines, in WHITE on a BLACK background.
        iii. Save this image to a file.
        iv. Call your own Gauss area estimation function to compute the estimated area from the current contour points (not the image).
        v. Print (to the console) the filename, the number of points in the current contour and the Gauss' estimate of the area.

Below is an example of console output (on a different image; these are not the right numbers for you) and an example of two of the images:

```
File hand2.png: Contour points = 2696, area = 133656.0, Gauss area = 135343.0
File hand2.png: Contour points = 1348, Gauss area = 135343.0
File hand2.png: Contour points = 674, Gauss area = 135665.0
File hand2.png: Contour points = 337, Gauss area = 135891.5
File hand2.png: Contour points = 169, Gauss area = 135825.0
File hand2.png: Contour points = 85, Gauss area = 138355.0
File hand2.png: Contour points = 43, Gauss area = 142746.5
File hand2.png: Contour points = 22, Gauss area = 152350.5
File hand2.png: Contour points = 11, Gauss area = 128974.0
```





Be sure and obey proper practice for loading, converting and using image files.

You must write a function to implement the Pavlidis contour extraction; its inputs should be the binary image and the start point (x and y). This function should return the contour as an ordered set of (x,y) points. This function cannot use any other OpenCV or other functions (numpy functions for basic arithmetic are fine).

You must write a function to implement the Gauss polygon area estimation method; its input should be a set of contour points. This function should return the area. This function cannot use any other OpenCV or other functions (numpy functions for basic arithmetic are fine).

You must write a function to implement one pass of the Discrete Curve Evolution algorithm; its input should be a set of contour points. This function should return a new contour from which the least relevant point has been removed. This function cannot use any other OpenCV or other functions (numpy functions for basic arithmetic are fine).

Your submission will consist of a Word or pdf file and a .py file (no .ipnyb files) containing your code, as well as eighteen image files (for each input image, the binary image and the eight images of the successively trimmed contours). Do NOT put all of your files into a zip file; attach them separately. Your Word file should contain:

- your complete Python code (pasted in as plain text, no screenshots or dark mode); and
- the console output of your program as described above, pasted in as plain text.

```python
# this little function calculates the area of an object from its contour
# using simple pixel counting
# holes in the object are not considered (they count as if they are filled in)
# it's really klugey - not great code
def fillArea(ctr):
    maxx = np.max(ctr[:, 0]) + 1
    maxy = np.max(ctr[:, 1]) + 1
    contourImage = np.zeros( (maxy, maxx) )
    length = ctr.shape[0]
    for count in range(length):
        contourImage[ctr[count, 1], ctr[count, 0]] = 255
        cv2.line(contourImage, (ctr[count, 0], ctr[count, 1]), \
                 (ctr[(count + 1) % length, 0], ctr[(count + 1) % length, 1]), \
                 (255, 0, 255), 1)
    fillMask = cv2.copyMakeBorder(contourImage, 1, 1, 1, 1, \
                    cv2.BORDER_CONSTANT, 0).astype(np.uint8)
    areaImage = np.zeros((maxy, maxx), np.uint8)
    startPoint = (int(maxy/2), int(maxx/2))
    cv2.floodFill(areaImage, fillMask, startPoint, 128)
    area = np.sum(areaImage)/128
    return area
```