王　兆東

マルチメディア信号解析

2020年6月2日

Report of First Assignment

In this report I will discuss how I design my program to complete the assigned task, it include how to read an image by program and several image transferring operation like enlargement, reduction and sampling.

1. Environment Introduction

· Programming language: Python3.6

· library would be used: OpenCV [1]

· text book：jupyter [2]

2. Implementation

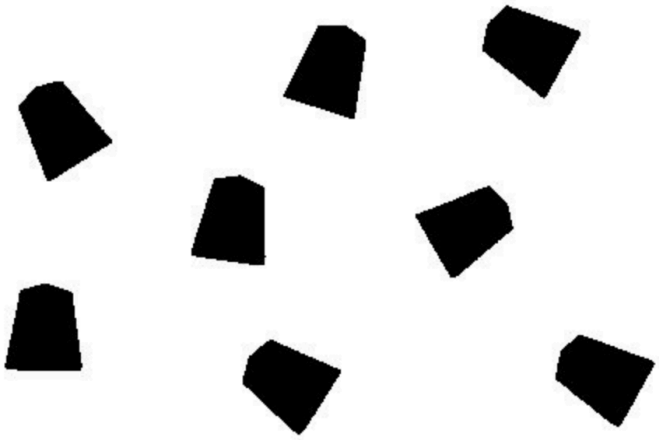
First we discuss the Otsu image segmentation，It divides an image into background and target by the grayscale characteristics of the image. The interclass variance between the background and the target，For the image I(x,y), the segmentation threshold of the foreground (i.e. target) and background is denoted as T, and the number of pixels belonging to the foreground accounts for the entire image，as it is ω0, and its average grayscale is μ0. The proportion of background pixels to the whole image is ω1, and its average grayscale is μ1. We calculate the g as:

$g = \omega 0 \omega 1 (\mu 0 - \mu 1)^2$

Traverse the entire image, the maximum value g is the threshold.

```
for i in range(0,256):

    w0 = sum_fromi2j(count_ratio, 0, i)
    u0 = mean_fromi2j(count, 0, i)

    w1 = sum_fromi2j(count_ratio, i, 255)
    u1 = mean_fromi2j(count, i, 255)

    g[i] = w0*w1*(np.power(u0-u1, 2))
```

The threshold been calculated is 152, the image segregated by the threshold are as follows:



For the iteration method, it goes as follows:

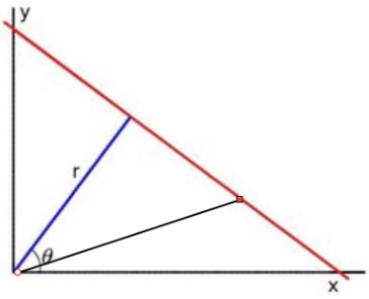
First, let the initialized threshold T be the average of grayscale. Using T to separate the image into 2 parts, and calculate the means of grayscale separately as ZO and ZB. Then recalculate the $T = ZO + ZB/2$, Until the T doesn't change anymore. Finally the T would be the threshold of this image. The main part of code is as follows.

```
while(T_changed):
    oldT = T
    zo = mean_fromi2j(count, 0, T)
    zb = mean_fromi2j(count, T, 255)
    T = int((zo + zb)/2)
    if oldT == T:
        T_changed = False
```

Then we have the Hough transform. The Hough transform can be used to detect lines circles or

The Hough transform can be used to detect lines, circles or other parametric curves.

Every line in the image can be described as (r, θ), and the distance of r is limited(Does not exceed the length of the diagonal)，so we can have an accumulator which could calculate the vote from every pixel, and choose the max (r, θ) as the line we want.



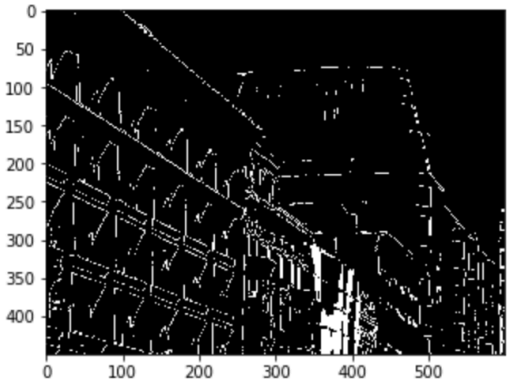
$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

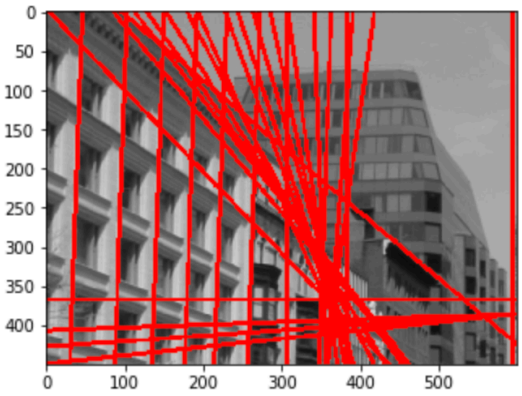
$$r = x\cos\theta + y\sin\theta$$

The main code are as follows:

```
def linear_detector_hough(img):
    imgsize = img.shape
    ThetaDim = 90
    DistStep = 1
    MaxDist = np.sqrt(imgsize[0]**2 + imgsize[1]**2)

    DistDim = int(np.ceil(MaxDist/DistStep))

    halfThetaWindowSize = 2
    halfDistWindowSize = int(DistDim/50)
    accumulator = np.zeros((ThetaDim,DistDim))

    sinTheta = [np.sin(t*np.pi/ThetaDim) for t in range(ThetaDim)]
    cosTheta = [np.cos(t*np.pi/ThetaDim) for t in range(ThetaDim)]

    for i in range(imgsize[0]):
        for j in range(imgsize[1]):
            if not edge[i,j] == 0:
                for k in range(ThetaDim):
                    accumulator[k][int(round((i*cosTheta[k]+j*sinTheta[k])*DistDim/MaxDist))] += 1

    M = accumulator.max()
```

The result of Hough Transform.





2. Consideration

To analysis the cluster of lines on the top, I think it is because the uncompleted of threshold segregation, which leaves big part of white to disturb the line's recognition.

[1] OpenCV, https://opencv.org/

[2] Jupiter, https://jupyter.org/