2017年5月17日 星期三

HOG

主要思想：在一幅图像中，局部目标的形状能够被梯度或边缘的方向密度分布很好地描述。其本质是梯度的统计信息，而梯度主要存在于边缘所在的地方。

实现过程：简单来说，首先需要将图像分成小的cell。然后采集cell单元中各像素点的梯度或边缘的方向直方图。最后把这些直方图组合起来就可以构成特征描述器。

流程：

1、读入所需要的检测目标即输入的image

2、将图像进行灰度化（将输入的彩色的图像的r,g,b值通过特定公式转换为灰度值）

3、采用Gamma校正法对输入图像进行颜色空间的标准化（归一化）

4、计算图像每个像素的梯度（包括大小和方向），捕获轮廓信息

5、统计每个cell的梯度直方图（不同梯度的个数），形成每个cell的descriptor

6、将每几个cell组成一个block（以3*3为例），一个block内所有cell的特征串联起来得到该block的HOG特征descriptor

7、将图像image内所有block的HOG特征descriptor串联起来得到该image（检测目标）的HOG特征descriptor，这就是最终分类的特征向量

HOG参数设置是：2*2cell / 区间、8*8像素 / cell、8个直方图通道,步长为1。

数据准备

读入彩色图像，并转换为灰度值图像, 获得图像的宽和高。采用Gamma校正法对输入图像进行颜色空间的标准化（归一化），目的是调节图像的对比度，降低图像局部的阴影和光照变化所造成的影响，同时可以抑制噪音。采用的gamma值为0.5。

In [1]:
```
import cv2
import numpy as np
img = cv2.imread('/Users/houluanxuan/Desktop/cameraman.tif', cv2.IMREAD_GRAYSCALE)
```

In [2]:
```
cv2.imshow('Image', img)
cv2.imwrite("Image-test.jpg", img)
k=cv2.waitKey(0)
if k==ord('s'):#如果输入s
    cv2.destroyAllWindows()
img = np.sqrt(img / float(np.max(img)))
cv2.imshow('Image', img)
cv2.imwrite("Image-test2.jpg", img)
k=cv2.waitKey(0)
if k==27:#如果输入ESC退出
    cv2.destroyAllWindows()
```

输入图像 

输出图像（gamma） 

计算每个像素的梯度

计算图像横坐标和纵坐标方向的梯度，并据此计算每个像素位置的梯度方向值；求导操作不仅能够捕获轮廓，人影和一些纹理信息，还能进一步弱化光照的影响。在求出输入图像中像素点（x,y）处的水平方向梯度、垂直方向梯度和像素值，从而求出梯度幅值和方向。

首先用[-1,0,1]梯度算子对原图像做卷积运算，得到x方向（水平方向，以向右为正方向）的梯度分量gradscalx，然后用[1,0,-1]T梯度算子对原图像做卷积运算，得到y方向（竖直方向，以向上为正方向）的梯度分量gradscaly。

```
In [3]: height, width = img.shape
        gradient_values_x = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)
        gradient_values_y = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)
        gradient_magnitude = cv2.addWeighted(gradient_values_x, 0.5, gradient_values_y, 0.5, 0)
        gradient_angle = cv2.phase(gradient_values_x, gradient_values_y, angleInDegrees=True)
        print gradient_magnitude.shape, gradient_angle.shape
```

out

(256, 256) (256, 256)

为每个细胞单元构建梯度方向直方图

我们将图像分成若干个“单元格cell”，默认我们将cell设为8*8个像素。假设我们采用8个bin的直方图来统计这6*6个像素的梯度信息。也就是将cell的梯度方向360度分成8个方向块，

```
cell_size = 8
bin_size = 8
angle_unit = 360 / bin_size
gradient_magnitude = abs(gradient_magnitude)
cell_gradient_vector = np.zeros((height / cell_size, width / cell_size, bin_size))

print cell_gradient_vector.shape

def cell_gradient(cell_magnitude, cell_angle):
    orientation_centers = [0] * bin_size
    for k in range(cell_magnitude.shape[0]):
        for l in range(cell_magnitude.shape[1]):
            gradient_strength = cell_magnitude[k][l]
            gradient_angle = cell_angle[k][l]
            min_angle = int(gradient_angle / angle_unit)%8
            max_angle = (min_angle + 1) % bin_size
            mod = gradient_angle % angle_unit
            orientation_centers[min_angle] += (gradient_strength * (1 - (mod / angle_unit)))
            orientation_centers[max_angle] += (gradient_strength * (mod / angle_unit))
    return orientation_centers


for i in range(cell_gradient_vector.shape[0]):
    for j in range(cell_gradient_vector.shape[1]):
        cell_magnitude = gradient_magnitude[i * cell_size:(i + 1) * cell_size,
                         j * cell_size:(j + 1) * cell_size]
        cell_angle = gradient_angle[i * cell_size:(i + 1) * cell_size,
                     j * cell_size:(j + 1) * cell_size]
        print cell_angle.max()
        cell_gradient_vector[i][j] = cell_gradient(cell_magnitude, cell_angle)
```

可视化Cell梯度直方图

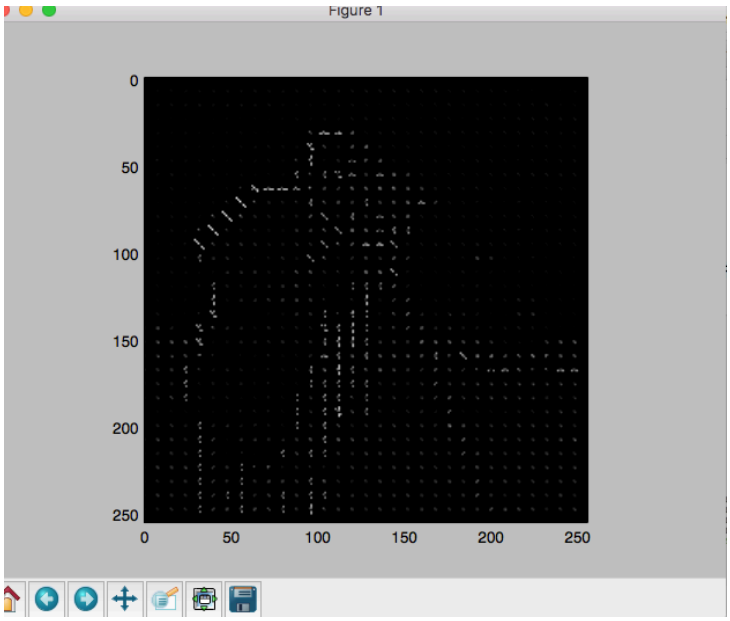
将得到的每个cell的梯度方向直方图绘出，得到特征图

```
In [*]: # fourth part

        import math
        import matplotlib.pyplot as plt

        hog_image= np.zeros([height, width])
        cell_gradient = cell_gradient_vector
        cell_width = cell_size / 2
        max_mag = np.array(cell_gradient).max()
        for x in range(cell_gradient.shape[0]):
            for y in range(cell_gradient.shape[1]):
                cell_grad = cell_gradient[x][y]
                cell_grad /= max_mag
                angle = 0
                angle_gap = angle_unit
                for magnitude in cell_grad:
                    angle_radian = math.radians(angle)
                    x1 = int(x * cell_size + magnitude * cell_width * math.cos(angle_radian))
                    y1 = int(y * cell_size + magnitude * cell_width * math.sin(angle_radian))
                    x2 = int(x * cell_size - magnitude * cell_width * math.cos(angle_radian))
                    y2 = int(y * cell_size - magnitude * cell_width * math.sin(angle_radian))
                    cv2.line(hog_image, (y1, x1), (y2, x2), int(255 * math.sqrt(magnitude)))
                    angle += angle_gap

        plt.imshow(hog_image, cmap=plt.cm.gray)
        plt.show()
```



统计Block的梯度信息

把细胞单元组合成大的块(block），块内归一化梯度直方图

本次采用的是矩阵形区间，它可以有三个参数来表征：每个区间中cell单元的数目、每个cell单元中像素点的数目、每个cell的直方图通道数目。

本次实验中我们采用的参数设置是：2*2cell / 区间、8*8像素 / cell、8个直方图通道,步长为1。则一块的特征数为2*2*8

In [6]:
```
# fifth part
hog_vector = []
for i in range(cell_gradient_vector.shape[0] - 1):
    for j in range(cell_gradient_vector.shape[1] - 1):
        block_vector = []
        block_vector.extend(cell_gradient_vector[i][j])
        block_vector.extend(cell_gradient_vector[i][j + 1])
        block_vector.extend(cell_gradient_vector[i + 1][j])
        block_vector.extend(cell_gradient_vector[i + 1][j + 1])
        mag = lambda vector: math.sqrt(sum(i ** 2 for i in vector))
        magnitude = mag(block_vector)
        if magnitude != 0:
            normalize = lambda block_vector, magnitude: [element / magnitude for element in block_vector]
            block_vector = normalize(block_vector, magnitude)
        hog_vector.append(block_vector)

print np.array(hog_vector).shape
```

out

（961，32）