

电类工程导论C 实验报告

——图像特征抽取

目录页

1. 引言

2. 实验环境

3. 预备知识

4. 实验过程

4.1. 颜色直方图

4.2. 灰度直方图

4.3. 灰度梯度直方图

5. 困难和解决方案

6. 总结

7. 参考

1. 引言

在这次实验里，我首先完成了在 OS X 的 opencv 部署，然后对两个图片分别进行了颜色直方图、灰度直方图和灰度梯度直方图。制作图的过程主要是由 Python 打印数据，然后将数据复制到 excel 中进行构图。（代码在同一个压缩包内。）

2. 实验环境

OS X 10.11.1 + opencv 2.4.12 + numpy1.10.1 + Python 2.7.10 (64bit)+Excel 2016

3. 预备知识

预备知识主要包含了颜色直方图、灰度直方图、灰度梯度直方图的概念以及 opencv 还有 numpy 的基本了解。

3.1.颜色直方图

注意，此实验中的颜色直方图并不是指在某一个分色上一个图片在各个色度上的分部图像，而指的是不同颜色的能量比例分布。体现了一个图像在各个颜色分量上的分部。

3.2.灰度直方图

灰度直方图定位为各个灰度值对应的像素点的数目的相对比例。体现了一个图像的明亮。

3.3.梯度直方图

灰度梯度定义为各个灰度梯度强度区间所包含的像素点的数目比例。（注意，边界的像素点不计入）。体现了一个图像纹理的复杂度。

3.4.opencv

opencv是由 Intel 最先开始主办的开源计算机视觉工具库，如果要在 python 中使用，需要结合 numpy 的应用。

实验过程中 opencv 的文档非常有帮助。比如这一段话，非常简洁地让我理解了 imread 函数的几个参数的含义。

Python: `cv2.imread(filename[, flags])` → retval

C: `IplImage* cvLoadImage(const char* filename, int iscolor=CV_LOAD_IMAGE_COLOR)`

C: `CvMat* cvLoadImageM(const char* filename, int iscolor=CV_LOAD_IMAGE_COLOR)`

Parameters:

- **filename** – Name of file to be loaded.
- **flags** –

Flags specifying the color type of a loaded image:

- **CV_LOAD_IMAGE_ANYDEPTH** - If set, return 16-bit/32-bit image and convert it to 8-bit.
- **CV_LOAD_IMAGE_COLOR** - If set, always convert image to 3 channels.
- **CV_LOAD_IMAGE_GRAYSCALE** - If set, always convert image to grayscale.
- **>0** Return a 3-channel color image.

Note: In the current implementation the alpha channel value is ignored if you need the alpha channel.

- **=0** Return a grayscale image.
- **<0** Return the loaded image as is (with alpha channel).

3.5.numpy

numpy 可以用来处理大型的矩阵。这里我们的图片由于是一个一个像素点组成的矩阵，所以 numpy 可以帮助我们进行非常有效简洁的计算。比如4.1中会提到的 num.py，还有梯度计算中的矩阵思想。

4. 实验过程

4.1. 颜色直方图

在计算颜色直方图时，我的思路是，先将一个图片按照 rgb 模式读入，然后 split 成三个分量矩阵，然后对每个矩阵求出所有像素点在这个矩阵内部的值的和。这样就得到了三种颜色的能量，然后再算出能量总和，这样就可以分别算出能量的比例了。

代码如下：

```
def CalcEnergy():  
    img1_rgb = cv2.imread('img1.png',1) #读入图像  
    b,g,r = cv2.split(img1_rgb) #分散三种颜色，注意顺序  
    #分别计算三种颜色的能量  
    energy_b = np.sum(b)  
    energy_g = np.sum(g)  
    energy_r = np.sum(r)  
    energy_all = energy_b + energy_g + energy_r #计算总能量  
    #输出三个能量的比例  
    print (energy_b+0.0) / energy_all  
    print (energy_g+0.0) / energy_all  
    print (energy_r+0.0) / energy_all
```

可以看出我计算的方法是利用了 **np.sum** 函数，直接对整个矩阵的各个元素求和，非常简便。从而得到能量，公式如下：

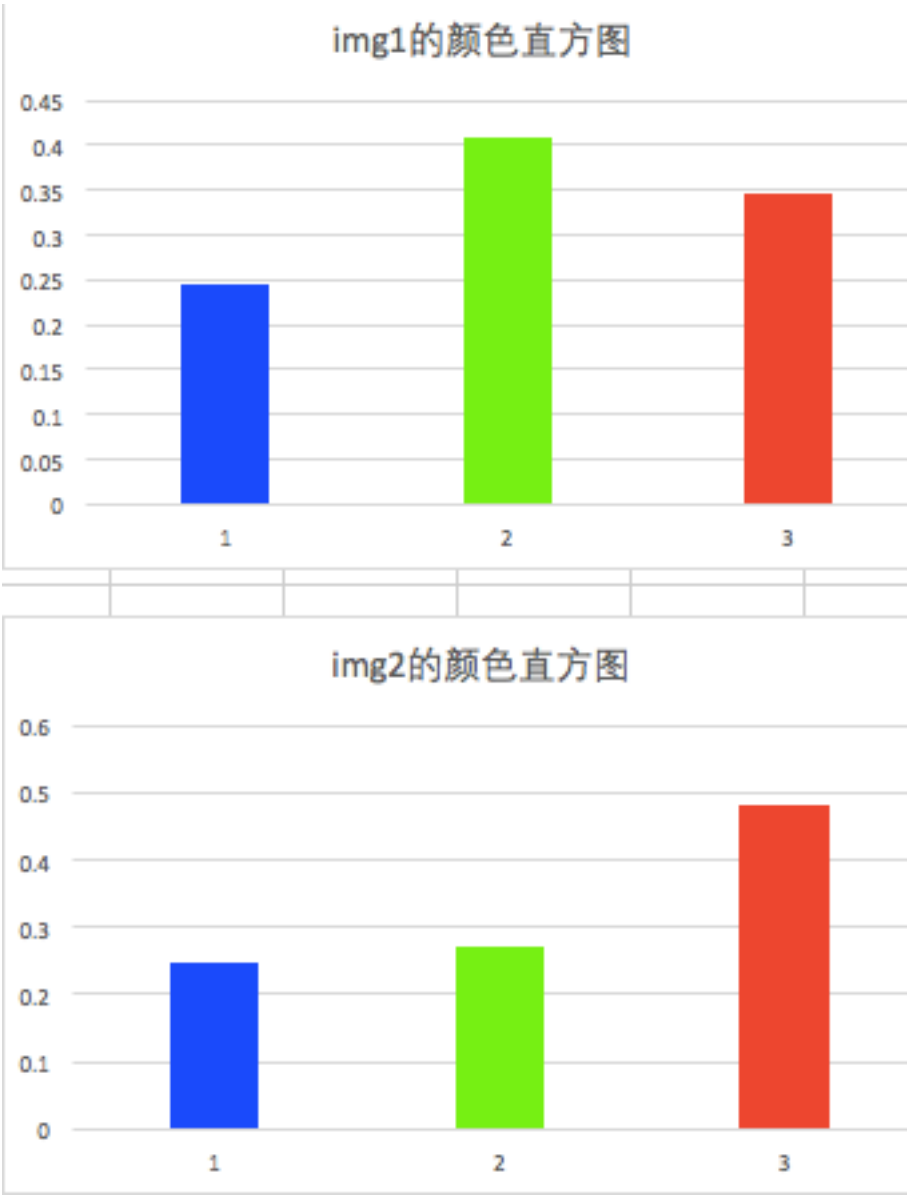
$$E(c) = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} I(x, y, c)$$

然后分别算出例，公式如下：

$$H(c) = \frac{E(c)}{\sum_{i=0}^2 E(i)}$$

将三个数据画出直方图即可。

如下：



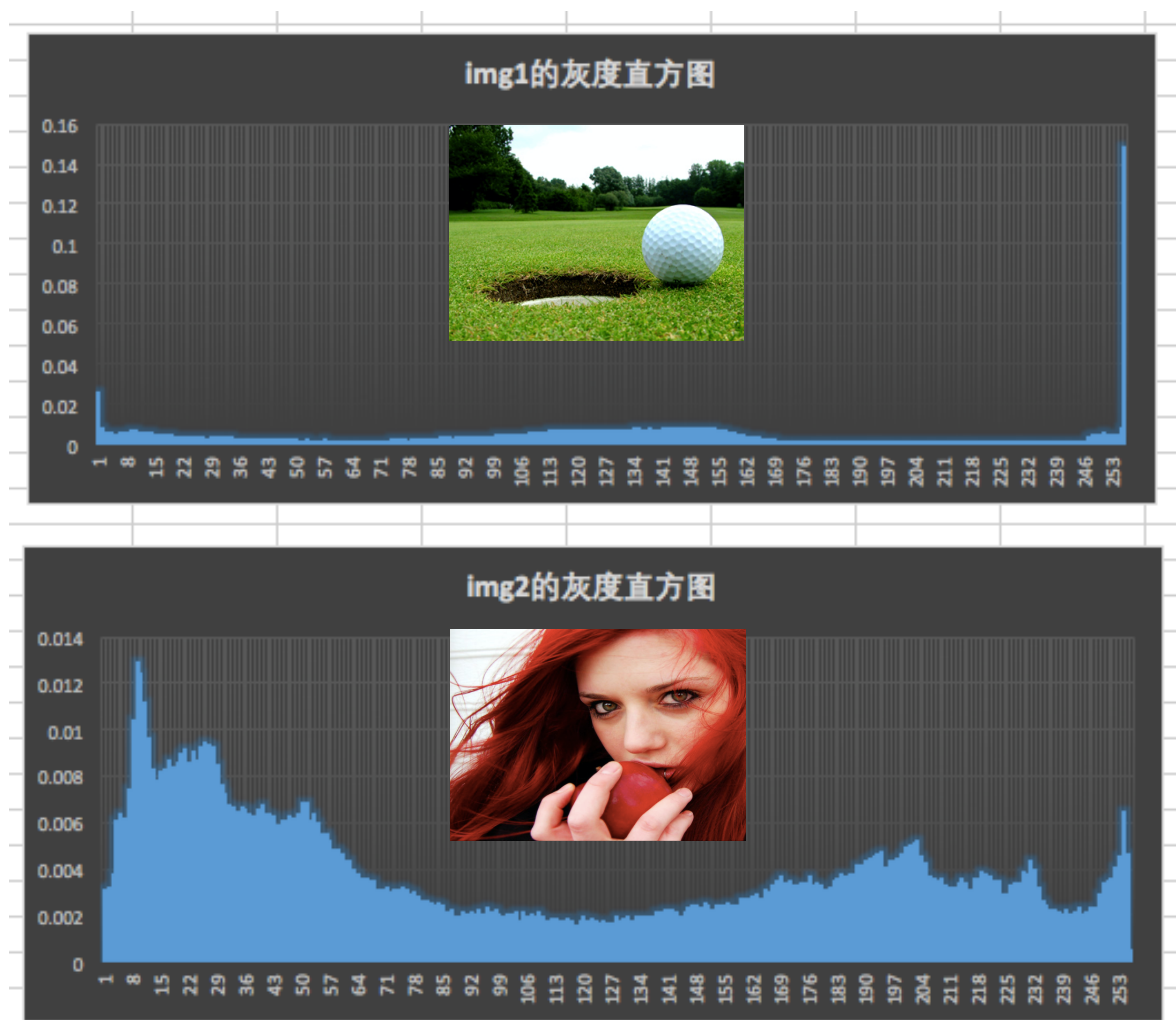
可以看出来，第一张图片是高尔夫球场，确实绿色占了绝大多数。第二张图片是一个红发女孩吃红苹果，确实红色的比例最多。

4.2. 灰度直方图

对于灰度的计算，我们可以利用 cv2 中的 `calcHist` 函数来生成灰度表，然后利用这个数据进行计算和输出。

```
def CalcGray():  
    #以灰度的形式读入  
    img2_gray = cv2.imread('img2.png',0)  
    #生成灰度表格，注意最大值最小值的范围  
    histGray_2 = cv2.calcHist([img2_gray],[0],None,[256],[0.0,256.0])  
    #用list 来处理更好的输出格式  
    h_list = []  
    for h in histGray_2:h_list.append(h[0])  
    #求出所有灰度所含像素点数目之后，当然也可以直接利用图片的长宽乘积  
    h_sum = sum(h_list)  
    #输出每个灰度所占的比例  
    for i in h_list:  
        print (i+0.0)/h_sum
```

放到 excel 中之后形成的图像如下：



由于灰度有0~255，一共256个值，所以它的直方图看起来很像一个曲线与x轴围成的面积。

可以看出img1的高峰在最右侧，也就是明亮的地方，所以整体看起来很明亮，正如高尔夫球场的明亮一样。然而第二个图片的灰度分部集中在较小的地方，说明图片相对偏暗。

所用公式是：

$$N(i) = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} I(x, y) == i ? 1 : 0$$
$$H(i) = \frac{N(i)}{\sum_{j=0}^{255} N(j)}, i = 0, \dots, 255$$

4.3. 灰度梯度直方图

要计算灰度梯度直方图（梯度直方图），我们首先要算出每一个像素点（除了边界点以外）的梯度强度，然后根据梯度强度的分部率画出图像。那么如何计算每一个像素点的灰度梯度呢？

首先我们可以创建矩阵算子，由于我们的公式如下：


$$I_x(x, y) = \frac{\partial I(x, y)}{\partial x} = I(x+1, y) - I(x-1, y)$$
$$I_y(x, y) = \frac{\partial I(x, y)}{\partial y} = I(x, y+1) - I(x, y-1)$$

所以，我们可以分别创建两个算子矩阵，分别是 $[-1,0,1]$ 和 $[-1,0,1]$ 的转置矩阵，有了这两个矩阵，我们直接调用 `cv2.filter2D` 函数即可实现在某一个方向的梯度的计算，这里要注意我们的图像必须转变成为一个 `numpy` 的对象才可以直接进行计算，所以需要 `astype` 函数，还有 `np.array` 函数的转化。

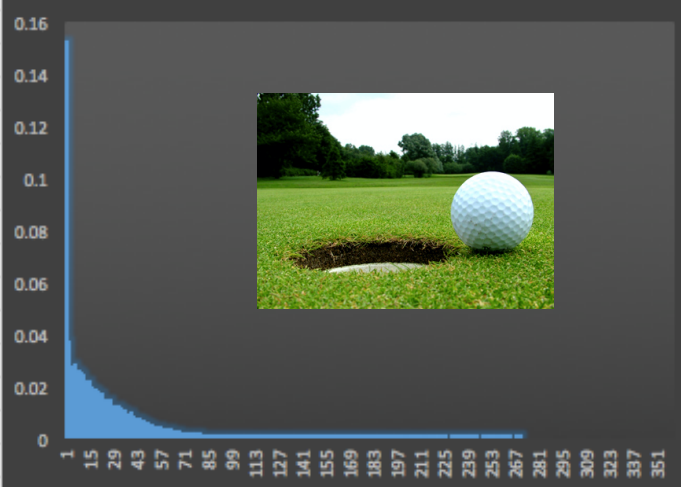
有了两个方向的梯度矩阵之后，我们还需要计算梯度强度矩阵，所以，我们要对所有的像素点进行遍历，然后分别根据梯度强度所在的整数区间值来进行计数，最后输出比例即可。

代码如下：

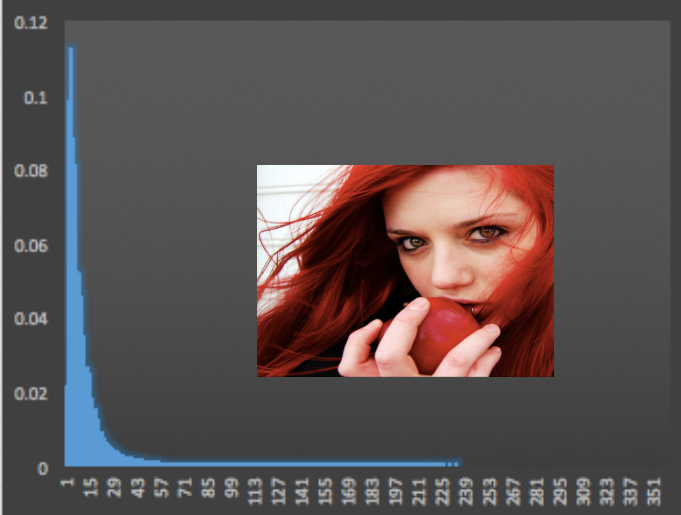
```
def CalcGridient():  
    #创建算子向量：  
  
    #mx：对每个元素的左面一个乘以-1，自己乘以0，右边乘以1 然后相加  
    #my：对每个元素的上面一个乘以-1，自己乘以0，下面乘以1 然后相加  
    mx = np.array([[ -1, 0, 1]])  
    my = np.array([[ -1, 0, 1]]).T #转置  
  
    #读入灰度图片，并且将其转成一个 numpy 对象，从而可以更好的计算  
    img1_gray = cv2.imread('img1.png',0)  
    im = np.array(img1_gray).astype(np.uint8)  
  
    #分别计算x 方向， y 方向的梯度  
    gx = cv2.filter2D(im, cv2.CV_32F, mx)  
    gy = cv2.filter2D(im, cv2.CV_32F, my)  
  
    #存储梯度强度  
    M = [0]*360  
  
    #二维遍历所有的像素点，计算他们的梯度强度  
    for x in range(1,gx.shape[0]):  
        for y in range(1,gx.shape[1]):  
            #计数  
            M[math.trunc( math.sqrt((gx[x][y]**2+gy[x][y]**2)) ) ]+=1;  
            -----  
  
    #输出所有的灰度强度分布  
    s = sum(M)  
    for m in M: print (m+0.0)/s
```


$$M(x, y) = \sqrt{I_x(x, y)^2 + I_y(x, y)^2}$$

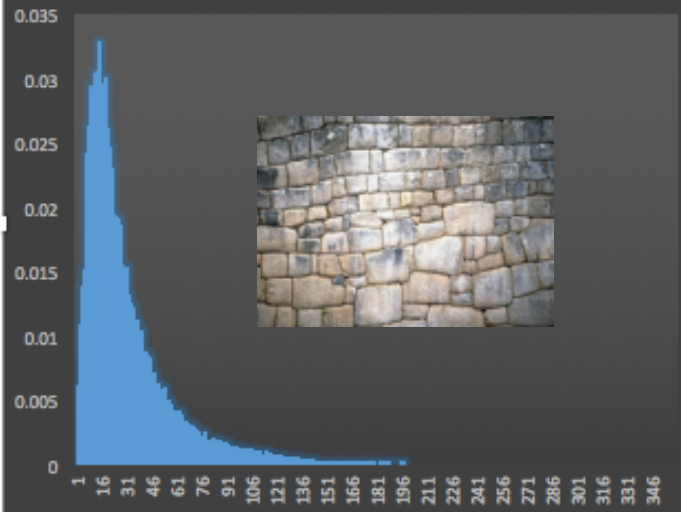
img1的灰度梯度直方图



img2的灰度梯度直方图



img3的灰度梯度直方图



本来只需要做两个图片，但是我觉得这两个图片的梯度直方图分别不太明显，所以右选了 PPT 中的一个图片进行计算，果然可以看出从img1到 img3，纹理复杂之后，灰度梯度直方图在渐渐改变。

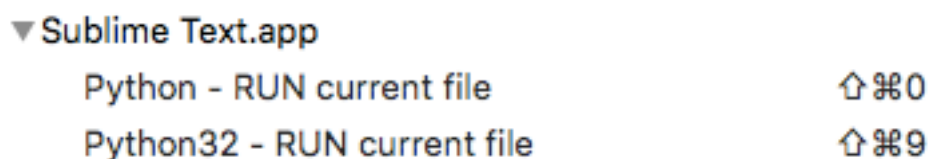
5. 遇到的困难和解决方案

在这次实验中遇到的第一个困难是在 OS X 下安装 opencv 和 numpy。个人习惯是喜欢用自己重新编译源码的方式来安装新框架，所以这次选择了 homebrew 来部署 opencv 和 numpy 这两个框架。在安装时一直出现了报错，提示有关于 Xcode 的应用没有授权，但是我一直没有找到如何授权的命令。最后偶然发现，只要打开 Xcode，它会自动弹出一个提示框，然后点击 Yes 授权就可以了。。。。

第二个困难就是因为前半学期我们都是用的 lucene 必须在32bit 的 python 中才可以，而我这次装的 opencv 是64bit 的，会提示：

```
ImportError: dlopen(/usr/local/lib/python2.7/site-packages/numpy/core/multiarray.so, 2): no suitable image found. Did find:  
  /usr/local/lib/python2.7/site-packages/numpy/core/multiarray.so: mach-o, but wrong architecture
```

所以，我又重新装了64bit 的 python，然后重新配置 Sublime Text 的 REPL 让两个版本的 python 可以共存，调用不同的快捷键来分别，从而实现了非常方便的切换 python 版本的功能。



```

{"command": "repl_open",
 "caption": "Python - RUN current file",
 "id": "repl_python_run",
 "mnemonic": "R",
 "args": {
  "type": "subprocess",
  "encoding": "utf8",
  "cmd": ["python", "-u", "$file_basename"],
  "cwd": "$file_path",
  "syntax": "Packages/Python/Python.tmLanguage",
  "external_id": "python",
  "extend_env": {"PYTHONIOENCODING": "utf-8"}
 }
},
{"command": "repl_open",
 "caption": "Python32 - RUN current file",
 "id": "repl_python32_run",
 "mnemonic": "R",
 "args": {
  "type": "subprocess",
  "encoding": "utf8",
  "cmd": ["/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7-32", "-u", "$file_basename"],
  "cwd": "$file_path",
  "syntax": "Packages/Python/Python.tmLanguage",
  "external_id": "python",
  "extend_env": {"PYTHONIOENCODING": "utf-8"}
 }
},

```

第三个困难是想用 opencv 自带的画图模块来画图，虽然代码结合度比较高，但是奇丑无比，就放弃了。还是 Excel 的图片更加现代化，至少能看。

第四个困难就是梯度的计算，一开始想用是一个非常 naive 的方法，不过觉得麻烦，而且不适用于以后的应用，所以在阅读了很多 opencv 的代码之后找到了这个方法来完成我想要的计算。

6. 总结

在这次实验中，学到了很多关于 opencv 和 numpy 的简单操作，也让我对图像的基本特征的抽取有了很多理解。

7. 参考

<http://www.tuicool.com/articles/r2yyei>

<http://www.numpy.org>

<http://stackoverflow.com/questions/21233043/non-sobel-discrete-gradients-in-python-opencv-or-numpy>

最后，衷心感谢师姐和何老师的指导。

林禹臣

5140309507

2015年11月21日