《多媒体技术》第 3 次作业

姓名：张半佛　　完成日期：2019 年 10 月

题目要求：

在指定的图片数据库内，完成图片检索。图像数据按照文件名的次序，大概 100 幅为一类，以颜色直方图为依据，用最近邻方法或者其他方法，为每一类建立模板。随机抽取一张图片，利用模板匹配方法，在图片库中找到 100 幅最相思的图片，计算查全率和查准率。
报告最后部分展示不少于 10 幅图片。这 10 幅图片来自不同类别，计算其查全率和查准率，报告中给出这 10 幅图片最接近的 5 幅图片。并分析结果。
（提交完整的实验报告，具体要求有：核心算法原理、处理步骤、每一步骤的结果分析，还包括使用不同参数的性能分析。不能直接调用库，必须一步一步走，核心算法可以调库。）
解：

目录

一：核心算法原理

1. 存储各个颜色直方图特征向量到 excel 表内存储：

   将 RGB 图像的每一个通道的颜色值 0~255 分为八个部分：①0~31 为第一部分。②32~63 为第二部分。③64~95 为第三部分。④96~127 为第四部分。。。以此类推。这意味着红绿蓝分别有 8 个部分，总共可以构成 8*8*8=512 个组合。

然后统计在这 512 个组合中每一个组合里面有多少个像素值。

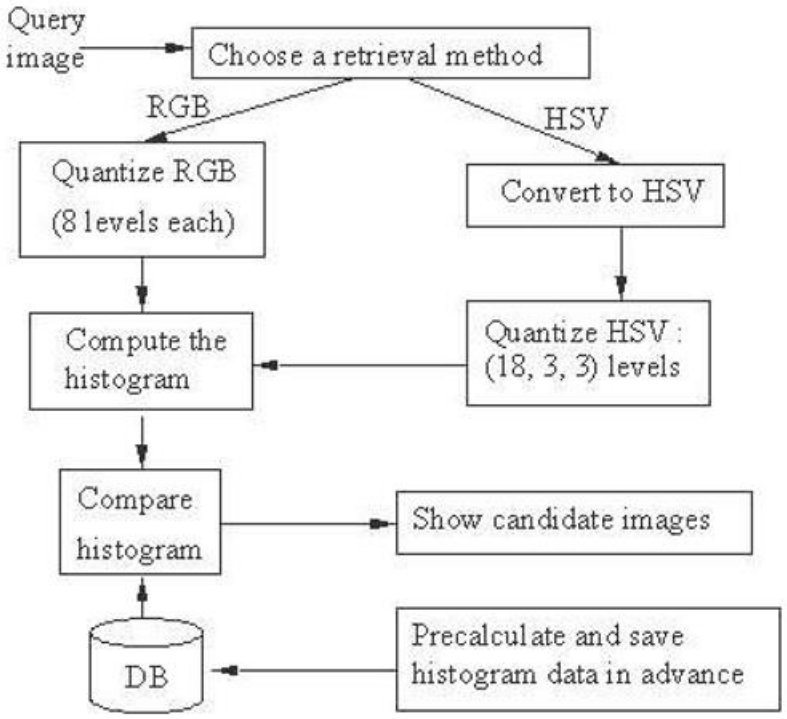
最后将这 512 维向量代表一个图片的特征向量存储到 excel 表中的一行。

例如：一个像素的 RGB 为（0，0，35），在 R 的第一部分，G 的第一部分，B 的第二部分，那么它就属于 512 个组合中的第二个组合，这个组合的值加一。

2. 计算进行匹配图像的颜色直方图特征向量，与 excel 表内的各向量进行匹配：

匹配相似度可以用巴氏距离距离来表示，因为经过大量的实验证明，巴氏距离在颜色直方图的匹配中表现是最好的一种度量手段。因此本次实验也采用巴氏距离作为测量手段。

逐个计算图片与图库中的巴氏距离，最后进行排序，效果最好的前 100 个进行观察。



二：详细处理步骤解析

1. 存储各个颜色直方图特征向量到 excel 表内存储：

```
if __name__ == '__main__':
    COLOR_DEGREE = 8
    for bigi in range(0,9908):
        img = cv2.imread('../image/' + str(bigi) + '.jpg')
        hist=getColorVec(img)
        with open('test8.csv', 'a+',newline='')as f:
            f_csv = csv.writer(f)
            f_csv.writerow(hist)
```

这里的 COLOR_DEGREE 就是分为 8 部分，之后对 9908 张图片依次用函数 getColorVec()计算每张图片的特征向量，得到 8*8*8=512 维的向量，存储到 test8.csv 的文件中。

getColorVec()函数如下：

```
def getColorVec(img):

    hei, width, channel=img.shape
    colorVec=[0 for e in range(0, int(pow(COLOR_DEGREE, 3)))]
    i=0
    while(i<hei):
        j=0
        while(j<width):
            pixel=img[i][j]
            grade=getPixelGrade(pixel)
            index=grade[0]*COLOR_DEGREE*COLOR_DEGREE+grade[1]*COLOR_DEGREE+grade[2]
            colorVec[index]+=1
            j+=1
        i+=1
    return colorVec


def getPixelGrade(pixel):
    grade=[]
    base=int(256/COLOR_DEGREE)+1
    for one in np.array(pixel):
        grade.append(int(one/base))
    return grade
```

2. 匹配：
首先读取需要匹配的图像：

```
imgpath='../image/7256.jpg'
img = cv2.imread(imgpath)
```

之后获得该图像的特征向量：

```
hist1 = getColorVec(img)
```

之后读取 csv 文件使用巴氏距离判断两个向量之间的相关性：（这里共比较了 9908 次，即与每一张图片均进行了比较）

```
gray_dict = {}
jpgi=0
with open('test8.csv')as f:
    f_csv = csv.reader(f)
    for csvrow in f_csv:
        hist2 = [int(row) for row in csvrow]  # 将数据从string形式转换为float形式
        gray_dict[str(jpgi) + '.jpg'] = Bdistance(hist1, hist2)
        jpgi=jpgi+1
```

之后进行排序，截取最好的 100 个进行输出：

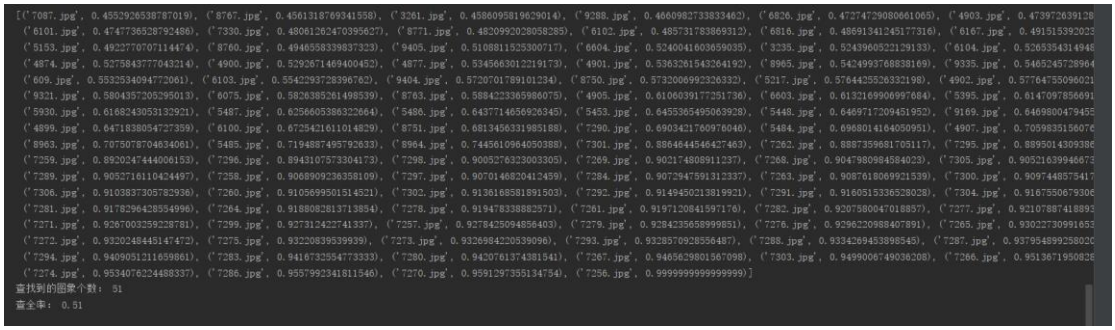
```
gray_dict = sorted(gray_dict.items(),key=operator.itemgetter(1))

dic100=gray_dict[-100:]
print(dic100)
```

通过观察计算出查全率和查准率。

三：实验结果分析

1. 图像一（7256.jpg）：  ，

100 张图像如下：



最好的五张为：7270.jpg　7286.jpg　7274.jpg　7266.jpg　7303.jpg



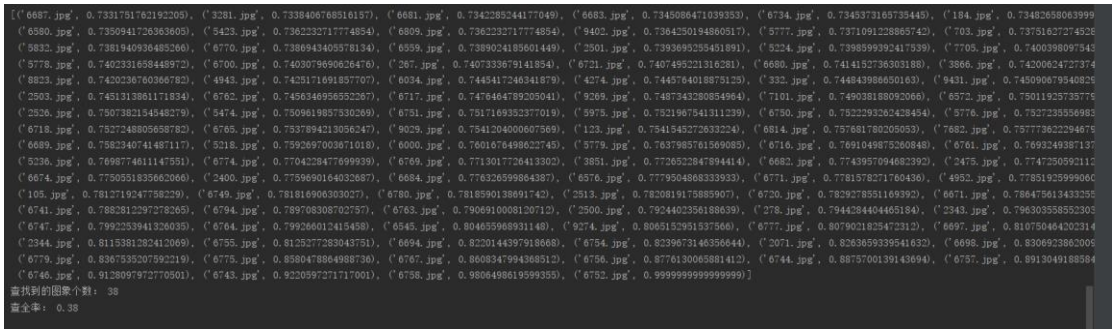


查找到 51 张相似的，查全率为 51%。

该类图像总共有 51 张（7256-7306），查准率为 100%。

2. 图像二（6752.jpg）： 

100 张图片如下：



最好的 5 张为：6758　6743　6746　6757　6744

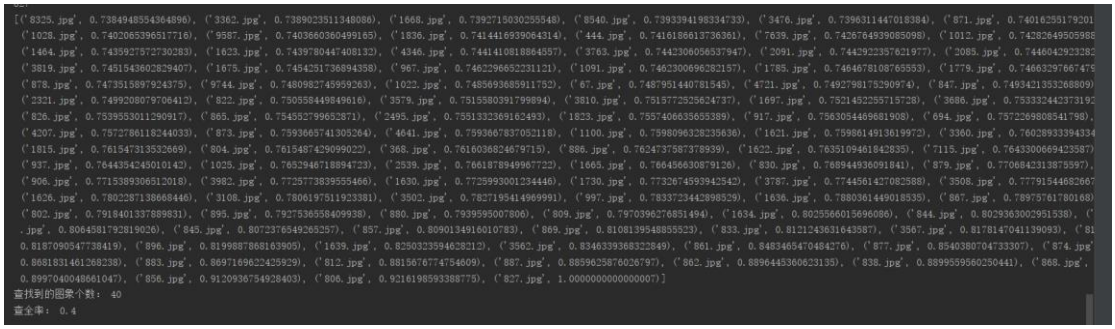



查找到 38 张相似的，查全率为 38%。

该类共有 40 张图（6741-6780），查准率为 95%

3. 图像三（827）： 

100 张图片如下：



最好的 5 张为：806　856　868　838　862



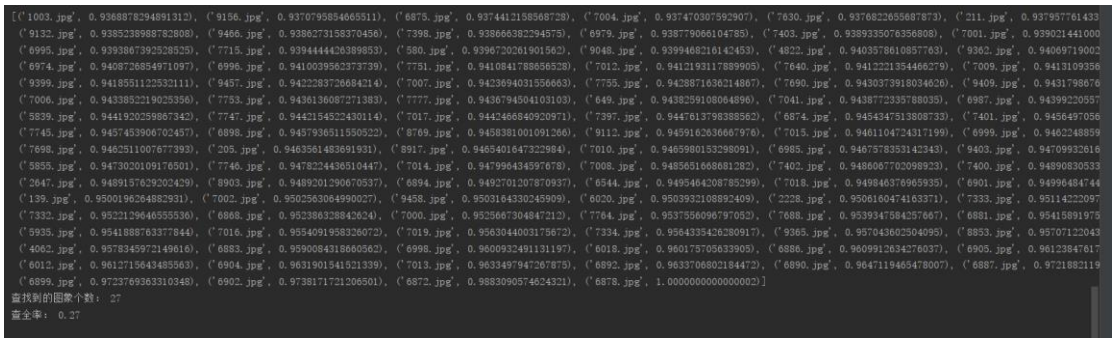


查找到张相似的 40，查全率为 40%。

该类共有 99 张图（800-898），查准率为 40%

4. 图像四（6878）： 

100 张图片如下：



最好的 5 张为：6872　6902　6899　6887　6890

查找到 27 张相似的，查全率为 27%。

该类共有 43 张图（6867-6909），查准率为 62.8%

5. 图像五（6959）： 

100 张图片如下：

最好的 5 张为：6960　6978　6974　6977　6990

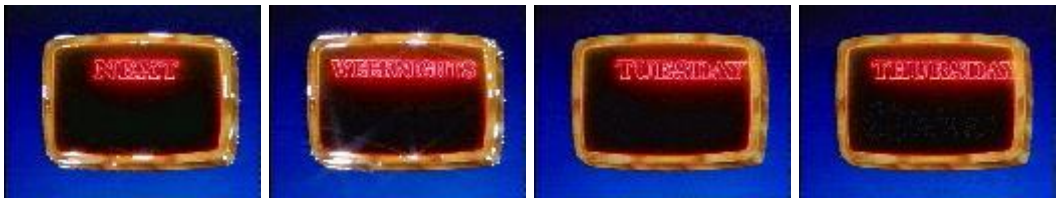
查找到 38 张相似的，查全率为 38%。

该类共有 39 张图（6959-6997），查准率为 97.4%

6. 图像六（7022）： 

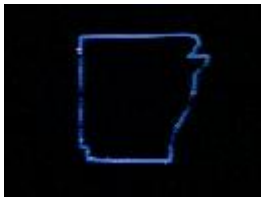
100 张图片如下：

最好的 5 张为：7028　7060　7048　7042　7026

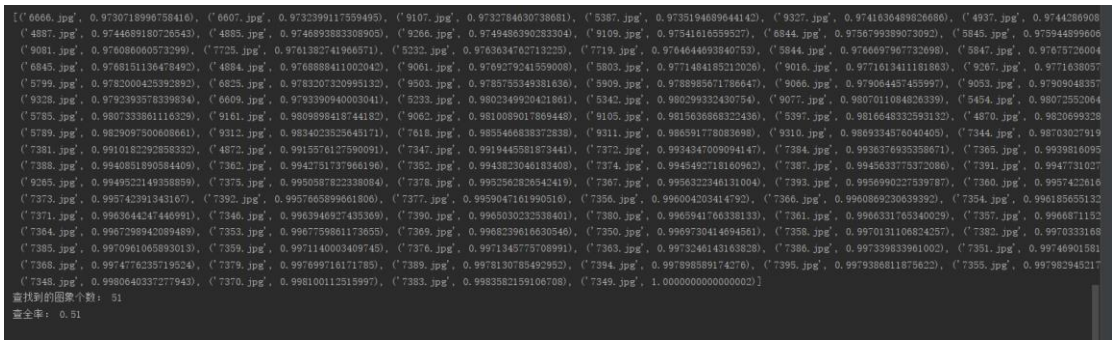



查找到 53 张相似的，查全率为 53%。

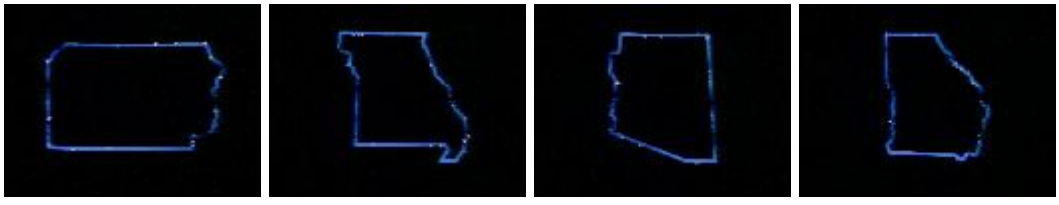
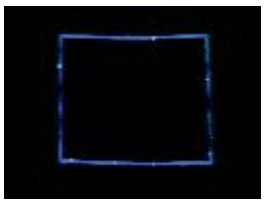
该类共有 70 张图（7020-7089），查准率为 75.7%

7. 图像七（7349）： 

100 张图片如下：



最好的 5 张为：7383　7370　7348　7355　7395

查找到 51 张相似的，查全率为 51%。

该类共有 52 张图（7344-7395），查准率为 98%

8. 图像八（9571）： 

100 张图片如下：

最好的 5 张为：9580　9581　9583　8329　9565

查找到 31 张相似的，查全率为 31%。

该类共有 89 张图（9505-9593），查准率为 34.8%

9. 图像九（19）： 

100 张图片如下：

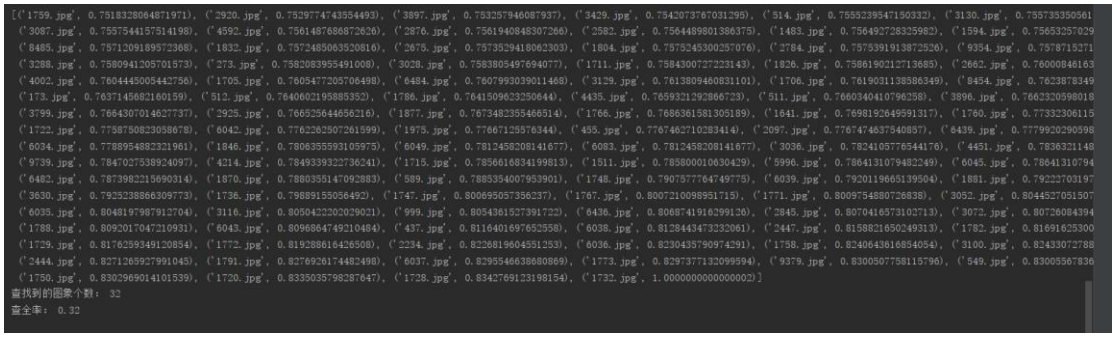
最好的 5 张为：9　571　17　4178　3604

查找到 19 张相似的，查全率为 19%。

该类共有 100 张图（0-99），查准率为 19%

10. 图像十（1732）： 

100 张图片如下：



最好的 5 张为：1728　1720　1750　549　9379

查找到 32 张相似的，查全率为 32%。

该类共有 98 张图（1695-1792），查准率为 32.6%

结果分析：

经过多次实验可以看出，颜色直方图用于匹配效果并不是太好。但还是能匹配出非常相近的图像，不过这仅限于两幅图片的颜色差不多，对图像物体的特征没有考虑到。

单纯的依靠颜色直方图的检索得到的结果并不如人意，如果加上针对图像本身属性的附加处理对其进行改进，效果应该会好很多。

可以采用 SIFT 特征提取加以辅助，效果应该会好一点。

四：附录（python 完整代码）

```
#imgstore.py,图像存储代码
import cv2
import csv
import numpy as np

def getColorVec(img):

    hei, width, channel=img.shape
    colorVec=[0 for e in range(0, int(pow(COLOR_DEGREE, 3)))]
    i=0
    while(i<hei):
        j=0
        while(j<width):
            pixel=img[i][j]
            grade=getPixelGrade(pixel)

index=grade[0]*COLOR_DEGREE*COLOR_DEGREE+grade[1]*COLOR_DEGREE+grade[2]
            colorVec[index]+=1
            j+=1
        i+=1
    return colorVec


def getPixelGrade(pixel):
    grade=[]
    base=int(256/COLOR_DEGREE)+1
    for one in np.array(pixel):
        grade.append(int(one/base))
    return grade

if __name__ == '__main__':
    COLOR_DEGREE = 8
    for bigi in range(0,9908):
        img = cv2.imread('../image/' + str(bigi) + '.jpg')
        hist=getColorVec(img)
        with open('test8.csv', 'a+',newline='')as f:
            f_csv = csv.writer(f)
            f_csv.writerow(hist)
```

```
#imgmatch.py,图像匹配代码
import cv2
import numpy as np
import csv
import re
from math import sqrt

def getColorVec(img):

    hei, width, channel=img.shape
    colorVec=[0 for e in range(0, int(pow(COLOR_DEGREE, 3)))]
    i=0
    while(i<hei):
        j=0
        while(j<width):
            pixel=img[i][j]
            grade=getPixelGrade(pixel)

index=grade[0]*COLOR_DEGREE*COLOR_DEGREE+grade[1]*COLOR_DEGREE+grade[2]
            colorVec[index]+=1
            j+=1
        i+=1
    return colorVec


def getPixelGrade(pixel):
    grade=[]
    base=int(256/COLOR_DEGREE)+1
    for one in np.array(pixel):
        grade.append(int(one/base))
    return grade


def Bdistance(l1, l2):
    if(len(l1)!=len(l2)):
        raise RuntimeError("计算巴氏距离时，引入长度不相等的向量")
    s1=sum(l1)
    s2=sum(l2)
    BD=0
    for ind in range(0, len(l1)):
        BD+=sqrt((l1[ind]/s1)*(l2[ind]/s2))
    return BD

if __name__ == '__main__':
```

```
    import operator
    imgpath='../image/1732.jpg'
    img = cv2.imread(imgpath)
    imgnum=(int(re.sub("\D", "", imgpath)))//100
    COLOR_DEGREE=8
    hist1 = getColorVec(img)
    gray_dict = {}
    jpgi=0
    with open('test8.csv')as f:
        f_csv = csv.reader(f)
        for csvrow in f_csv:
            hist2 = [int(row) for row in csvrow]  # 将数据从 string 形式转换
为 float 形式
            gray_dict[str(jpgi) + '.jpg'] = Bdistance(hist1, hist2)
            jpgi=jpgi+1

    gray_dict = sorted(gray_dict.items(),key=operator.itemgetter(1))

    dic100=gray_dict[-100:]
    totalnum=0
    for i in range(0,len(dic100)):
        num=int(re.sub("\D", "", dic100[i][0]))
        print(num)
        if num>=imgnum*100 and num<=imgnum*100+200:
            totalnum=totalnum+1
    print(dic100)
    print('查找到的图象个数：',totalnum)
    print('查全率：',totalnum/100.0)
```