

# 《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
二：详细处理步骤解析 .....	2
三：实验结果分析 .....	4
四：附录（python 完整代码） .....	13

## 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  $8 \times 8 \times 8 = 512$  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。

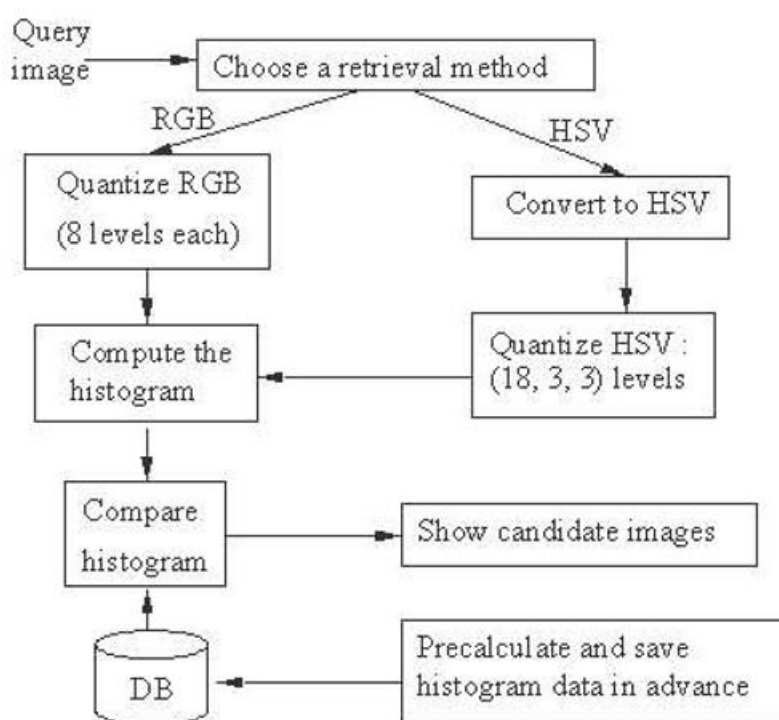
最后将这 512 维向量代表一个图片的特征向量存储到 excel 表中的一行。

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  $8 \times 8 \times 8 = 512$  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片：

```

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：

```
('7087.jpg', 0.455292538787019), ('8767.jpg', 0.4561318769341558), ('3281.jpg', 0.4586095819629014), ('9288.jpg', 0.4660982733833462), ('6826.jpg', 0.47274729080601065), ('4903.jpg', 0.473972639126),
('6101.jpg', 0.474736528792488), ('7330.jpg', 0.48061262470395627), ('8771.jpg', 0.4820992028058285), ('6102.jpg', 0.485731783869312), ('6816.jpg', 0.48591341245177316), ('6167.jpg', 0.491515392023),
('5153.jpg', 0.4922770707114474), ('8760.jpg', 0.4946558339837323), ('9405.jpg', 0.5108811525300717), ('6604.jpg', 0.5240041603659035), ('3235.jpg', 0.524396052129133), ('6104.jpg', 0.5265354314946),
('4874.jpg', 0.5275843777043214), ('4900.jpg', 0.5292671469400452), ('4877.jpg', 0.5345663012219173), ('4901.jpg', 0.5363261543264192), ('8965.jpg', 0.5424993768838169), ('9335.jpg', 0.5465245728964),
('609.jpg', 0.5532334094772061), ('6103.jpg', 0.5542293728396762), ('9404.jpg', 0.5720701789101234), ('8750.jpg', 0.573206992326332), ('5217.jpg', 0.5764425526332198), ('4902.jpg', 0.57764755096021),
('9321.jpg', 0.5804357305395013), ('6075.jpg', 0.5826385261498539), ('8763.jpg', 0.5884223365986079), ('4905.jpg', 0.6106039177251736), ('6603.jpg', 0.6132169906997684), ('5395.jpg', 0.6147097856691),
('5930.jpg', 0.6168243053132921), ('5487.jpg', 0.6256605386322644), ('5486.jpg', 0.6437714656926345), ('5453.jpg', 0.6455365495066928), ('5448.jpg', 0.6469717209451952), ('9169.jpg', 0.6469800479455),
('4899.jpg', 0.6471838054727359), ('6100.jpg', 0.6725421011014829), ('8751.jpg', 0.6813456331985188), ('7250.jpg', 0.6903421760976040), ('5484.jpg', 0.6968014164050951), ('4907.jpg', 0.7059833156076),
('8963.jpg', 0.7075078704034061), ('5485.jpg', 0.7194887495792633), ('8964.jpg', 0.7445610994050388), ('7301.jpg', 0.8864644546427463), ('7262.jpg', 0.8887359681705117), ('7295.jpg', 0.8895014309386),
('7259.jpg', 0.8920247444006153), ('7296.jpg', 0.8943107573304173), ('7298.jpg', 0.9005276323003309), ('7269.jpg', 0.9021748089112377), ('7268.jpg', 0.904786984584023), ('7305.jpg', 0.90521639946673),
('7289.jpg', 0.9053710110424497), ('7258.jpg', 0.9068909236358109), ('7297.jpg', 0.9070146820412499), ('7284.jpg', 0.907294759913123377), ('7263.jpg', 0.9087018069921539), ('7300.jpg', 0.9097448575417),
('7306.jpg', 0.9103837305782936), ('7260.jpg', 0.9105699501514521), ('7302.jpg', 0.9136168581891503), ('7292.jpg', 0.9149450213819921), ('7291.jpg', 0.9160515336528028), ('7304.jpg', 0.9167550679306),
('7281.jpg', 0.9178296423854996), ('7264.jpg', 0.9188082813713854), ('7278.jpg', 0.919478338825711), ('7261.jpg', 0.9197120841597176), ('7282.jpg', 0.9207880647018857), ('7277.jpg', 0.92107887418893),
('7271.jpg', 0.9267003495238761), ('7299.jpg', 0.9273124227413377), ('7257.jpg', 0.9276425094896403), ('7279.jpg', 0.9284235658999651), ('7276.jpg', 0.9296220868407891), ('7265.jpg', 0.93022730991653),
('7272.jpg', 0.9320248445147472), ('7275.jpg', 0.932208395399391), ('7273.jpg', 0.9326984220539960), ('7293.jpg', 0.9328570928556487), ('7288.jpg', 0.9334269453898545), ('7287.jpg', 0.937954599258020),
('7294.jpg', 0.9409051211698661), ('7283.jpg', 0.9416732554773333), ('7280.jpg', 0.9420761374381541), ('7267.jpg', 0.9465629801567099), ('7303.jpg', 0.9499006749036208), ('7266.jpg', 0.9513671950826),
('7274.jpg', 0.9534076224488337), ('7268.jpg', 0.9557952341811546), ('7270.jpg', 0.9591297355134754), ('7256.jpg', 0.9999999999999999)]
查找到的图像个数： 51
查全率： 0.51
```

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



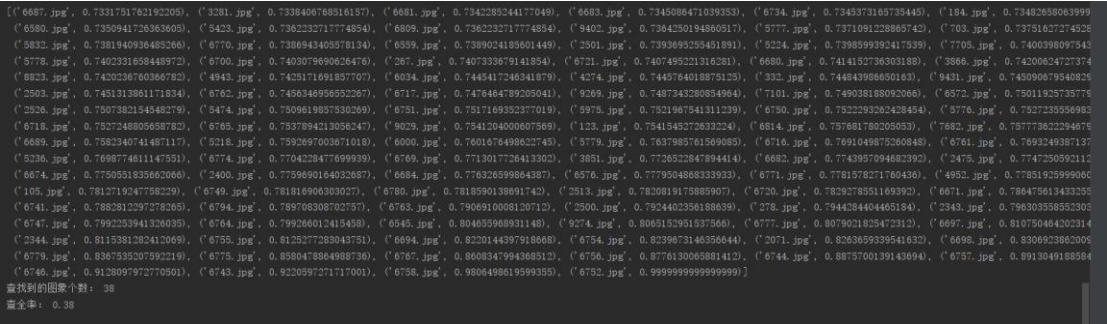
查找到 51 张相似的，查全率为 51%。

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





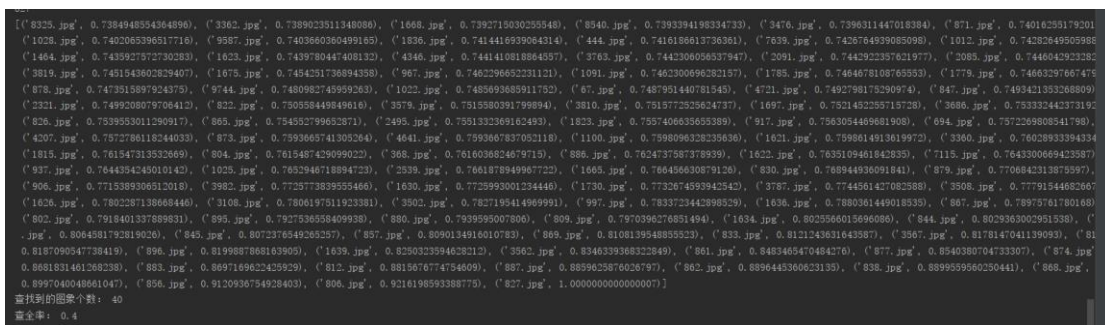
查找到 38 张相似的，查全率为 38%。

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%。

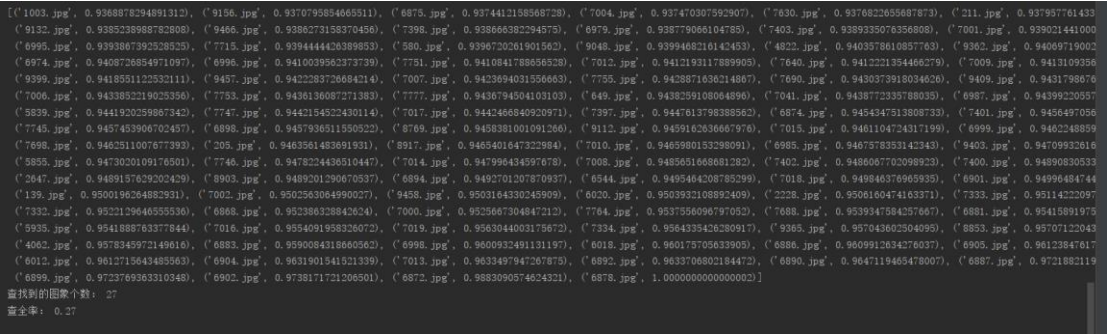
该类共有 99 张图（800-898），查准率为 40%





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：

[('5826.jpg', 0.9610067079999922), ('5825.jpg', 0.9613989732133624), ('4890.jpg', 0.9614323749959067), ('4891.jpg', 0.961502990568997), ('6047.jpg', 0.9615977824096161), ('9035.jpg', 0.9617912091), ('4893.jpg', 0.9623691040298537), ('9040.jpg', 0.9624682574348023), ('7731.jpg', 0.9626710975321747), ('7659.jpg', 0.9628832365079106), ('5152.jpg', 0.9629399075413742), ('6161.jpg', 0.9633945102), ('9080.jpg', 0.9634997535141239), ('6668.jpg', 0.9635533867024319), ('4892.jpg', 0.9642529736826199), ('4938.jpg', 0.9645485789687671), ('6006.jpg', 0.9649509978266414), ('5893.jpg', 0.965205776), ('5901.jpg', 0.9654881512194311), ('5945.jpg', 0.965526765767776), ('9108.jpg', 0.9658988200647138), ('9048.jpg', 0.9662515809588371), ('9064.jpg', 0.966756052115434), ('9013.jpg', 0.9668835879), ('7658.jpg', 0.9670121411506609), ('9032.jpg', 0.9671812653348925), ('7688.jpg', 0.9672414243750139), ('4940.jpg', 0.9677231969765102), ('9045.jpg', 0.9679983815556239), ('7764.jpg', 0.968712132313), ('9136.jpg', 0.9687401256649437), ('8891.jpg', 0.9689037515020468), ('9184.jpg', 0.968924972226839), ('8902.jpg', 0.970017136312205), ('7399.jpg', 0.9702701966046738), ('5849.jpg', 0.9708459139423), ('9060.jpg', 0.9709989926479844), ('7676.jpg', 0.971559550630275), ('8827.jpg', 0.971559550630275), ('6117.jpg', 0.9718490143439942), ('5978.jpg', 0.9726536182244727), ('5977.jpg', 0.9730714080), ('5946.jpg', 0.9735239437272724), ('5870.jpg', 0.9736612691589359), ('7398.jpg', 0.9740102073188477), ('5871.jpg', 0.975438114970328), ('8903.jpg', 0.975464093161887), ('5806.jpg', 0.975794932514), ('580.jpg', 0.975809899603142), ('5808.jpg', 0.9771147533402242), ('7403.jpg', 0.9773061883434881), ('7765.jpg', 0.9782683018600586), ('7397.jpg', 0.9783586724396699), ('7402.jpg', 0.97849726543), ('5895.jpg', 0.9785760053174817), ('7690.jpg', 0.9789522727138928), ('7687.jpg', 0.9799110691725951), ('8116.jpg', 0.9803437282313096), ('6963.jpg', 0.9803548666797697), ('6162.jpg', 0.9809221415), ('7401.jpg', 0.9815802263065353), ('6964.jpg', 0.9833530107502754), ('6966.jpg', 0.9834688768948848), ('7400.jpg', 0.9838174413859877), ('6965.jpg', 0.9843752227479584), ('6115.jpg', 0.9846838222), ('6969.jpg', 0.9854316419440112), ('6996.jpg', 0.9857721256454111), ('6987.jpg', 0.9861658316745288), ('6989.jpg', 0.9863166073469276), ('6994.jpg', 0.9869618426167849), ('6979.jpg', 0.9870522045), ('6985.jpg', 0.9877218851584372), ('6980.jpg', 0.9879670444716224), ('6984.jpg', 0.9885472440950814), ('6986.jpg', 0.9891316197618871), ('6975.jpg', 0.9891602933695726), ('6991.jpg', 0.9894251810), ('6976.jpg', 0.9895969893497475), ('6993.jpg', 0.9897470254368389), ('6988.jpg', 0.9900370237911265), ('6982.jpg', 0.9908785701188024), ('6967.jpg', 0.9911439294306703), ('6962.jpg', 0.9915696047), ('6972.jpg', 0.991663580787513), ('6993.jpg', 0.9918447335612126), ('6961.jpg', 0.9918969478993169), ('6973.jpg', 0.9921197997598465), ('6983.jpg', 0.9923871202417374), ('6992.jpg', 0.99242956368), ('6997.jpg', 0.99306413889624), ('6970.jpg', 0.9930776476651589), ('6971.jpg', 0.9931748852637089), ('6968.jpg', 0.9932512034708391), ('6990.jpg', 0.9934293325032328), ('6977.jpg', 0.993511269684), ('6974.jpg', 0.9943792476095933), ('6978.jpg', 0.9949135077914392), ('6960.jpg', 0.9951899031944202), ('6959.jpg', 1.0)]  
查找到图像个数: 38  
查全率: 0.38

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:

[('7100.jpg', 0.715762786925166), ('6165.jpg', 0.7167991159397495), ('9134.jpg', 0.7177493844450394), ('7098.jpg', 0.7178513931079145), ('7095.jpg', 0.718951630791964), ('8919.jpg', 0.71976870109), ('6840.jpg', 0.7198769779071152), ('6110.jpg', 0.7209497601616136), ('7449.jpg', 0.7214954219355446), ('7096.jpg', 0.7237750344049265), ('8866.jpg', 0.7238745334114348), ('9476.jpg', 0.7240673279), ('7092.jpg', 0.7244421142286325), ('7741.jpg', 0.7254103594685458), ('7091.jpg', 0.7264112817968743), ('9461.jpg', 0.7266533453611231), ('8878.jpg', 0.7266627392449208), ('9471.jpg', 0.7271376884), ('6086.jpg', 0.7276999253863718), ('5440.jpg', 0.7287499491571051), ('7458.jpg', 0.729137250207132), ('4942.jpg', 0.7295832736379956), ('7099.jpg', 0.7303184588736659), ('9482.jpg', 0.731453884315), ('7102.jpg', 0.7321197193114729), ('4878.jpg', 0.7321556664778561), ('7106.jpg', 0.7323649067637443), ('7742.jpg', 0.7325233752408798), ('5465.jpg', 0.7339200656835279), ('7093.jpg', 0.7339371459), ('8864.jpg', 0.7359031123785559), ('4944.jpg', 0.7365806045455273), ('5394.jpg', 0.7366663439636136), ('7466.jpg', 0.7405190417233528), ('8771.jpg', 0.7419937852778126), ('6090.jpg', 0.7421785655), ('7108.jpg', 0.7430068324629774), ('7107.jpg', 0.7430189198844428), ('9408.jpg', 0.7449501444121092), ('6830.jpg', 0.745621088992331), ('8912.jpg', 0.7479406882430245), ('7104.jpg', 0.7481811497), ('7090.jpg', 0.7487361103289539), ('5466.jpg', 0.7488990872767698), ('7103.jpg', 0.7491308784180902), ('7105.jpg', 0.7513513993559143), ('9477.jpg', 0.7540738176504647), ('8767.jpg', 0.75445338853), ('4945.jpg', 0.7569301423384891), ('5386.jpg', 0.7573689063504262), ('8894.jpg', 0.7646447879941298), ('8747.jpg', 0.7670337132784339), ('8922.jpg', 0.770119259323296), ('8905.jpg', 0.770455277396), ('6887.jpg', 0.7709022934067975), ('6837.jpg', 0.7744336602238189), ('6832.jpg', 0.7772651573680747), ('9479.jpg', 0.781749665552957), ('9407.jpg', 0.797326815290162), ('7728.jpg', 0.799208968148), ('9473.jpg', 0.8369801048819744), ('7727.jpg', 0.8427833141177032), ('8943.jpg', 0.88155232161232579), ('7098.jpg', 0.889717908897964), ('7023.jpg', 0.907407671824351), ('7035.jpg', 0.909743262507), ('7032.jpg', 0.9197653381950856), ('7039.jpg', 0.931834614067546), ('7032.jpg', 0.936337722445878), ('7030.jpg', 0.93732091794071472), ('7059.jpg', 0.937223681964674), ('7033.jpg', 0.94072318413), ('7027.jpg', 0.9417797291029092), ('7049.jpg', 0.9432081740685740), ('7050.jpg', 0.9442388073986801), ('7036.jpg', 0.9463785530292615), ('7047.jpg', 0.9470010166897687), ('7024.jpg', 0.9471133888), ('7021.jpg', 0.9520913052665619), ('7020.jpg', 0.9565505187695832), ('7035.jpg', 0.9576005053666673), ('7040.jpg', 0.9580767723111484), ('7023.jpg', 0.9601685258354259), ('7029.jpg', 0.9607665147), ('7054.jpg', 0.9610747793648380), ('7044.jpg', 0.961179099091159), ('7046.jpg', 0.9667203569675380), ('7045.jpg', 0.9672159513031521), ('7043.jpg', 0.9685072050612805), ('7025.jpg', 0.9690268920), ('7031.jpg', 0.9718792769100972), ('7051.jpg', 0.9730246832976935), ('7057.jpg', 0.9734722908301984), ('7038.jpg', 0.973674402671029), ('7026.jpg', 0.97376033987002), ('7042.jpg', 0.97433100756), ('7048.jpg', 0.9754094961900543), ('7060.jpg', 0.9866744098232076), ('7028.jpg', 0.9885923167806472), ('7022.jpg', 1.0000000000000004)]  
查找到图像个数: 53  
查全率: 0.53

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：

```
[('6666.jpg', 0.9730718996758416), ('6607.jpg', 0.9732399117559495), ('9107.jpg', 0.9732784630738651), ('5387.jpg', 0.9735194689644142), ('9327.jpg', 0.9741636489826686), ('4937.jpg', 0.9744286908), ('4887.jpg', 0.9744689180726543), ('4885.jpg', 0.9746893883308905), ('9266.jpg', 0.9749486390283304), ('9109.jpg', 0.975416165559527), ('6844.jpg', 0.9756799389073092), ('5845.jpg', 0.975944899606), ('9081.jpg', 0.976086060573299), ('7725.jpg', 0.9761382741966571), ('5232.jpg', 0.9763634762713225), ('7719.jpg', 0.9764644693840753), ('5844.jpg', 0.9766697967732698), ('5847.jpg', 0.97675726004), ('6845.jpg', 0.9768151136478492), ('4884.jpg', 0.9768888411002042), ('9061.jpg', 0.9769279241559008), ('5803.jpg', 0.9771484185212028), ('9016.jpg', 0.9771613411181863), ('9267.jpg', 0.9771638057), ('5799.jpg', 0.9782000425392892), ('6825.jpg', 0.9783207320995132), ('9503.jpg', 0.9785755349381638), ('5909.jpg', 0.9788985671786647), ('9066.jpg', 0.979064457455997), ('9053.jpg', 0.97909048357), ('9328.jpg', 0.9792393578339834), ('6609.jpg', 0.9793390940003041), ('5233.jpg', 0.9802349920421861), ('5342.jpg', 0.98029932430754), ('9077.jpg', 0.9807011084826339), ('5454.jpg', 0.98073552064), ('5785.jpg', 0.9807333861116329), ('9161.jpg', 0.9808989418744182), ('9062.jpg', 0.9810089017869448), ('9105.jpg', 0.9815636868322436), ('5397.jpg', 0.9816648332593132), ('4870.jpg', 0.9820699328), ('5789.jpg', 0.9829097500608661), ('9312.jpg', 0.9834023525648171), ('7618.jpg', 0.9855466838372838), ('9311.jpg', 0.986591778083698), ('9310.jpg', 0.9869334576040405), ('7344.jpg', 0.98703027919), ('7381.jpg', 0.9910182292858322), ('4872.jpg', 0.9915576127590091), ('7347.jpg', 0.9919445581873441), ('7372.jpg', 0.9934347009094147), ('7384.jpg', 0.9936376935358671), ('7365.jpg', 0.9939816095), ('7388.jpg', 0.9940881890584409), ('7362.jpg', 0.9942751737966190), ('7382.jpg', 0.9943823046183408), ('7374.jpg', 0.9945492718160962), ('7387.jpg', 0.9945633775372086), ('7391.jpg', 0.9947731027), ('9265.jpg', 0.9949522149388899), ('7375.jpg', 0.9950587822338084), ('7378.jpg', 0.9952562828542419), ('7367.jpg', 0.9956322346131004), ('7393.jpg', 0.9956990227539787), ('7360.jpg', 0.9957422616), ('7373.jpg', 0.995742913431079), ('7392.jpg', 0.9958589661806), ('7377.jpg', 0.9959047161990516), ('7356.jpg', 0.996004203414792), ('7366.jpg', 0.9960869230639392), ('7354.jpg', 0.996185655132), ('7371.jpg', 0.9963644247446991), ('7346.jpg', 0.9963946927435369), ('7390.jpg', 0.9965030232538401), ('7380.jpg', 0.9965941766338133), ('7361.jpg', 0.9966331765340029), ('7397.jpg', 0.9966871152), ('7364.jpg', 0.9967298942089489), ('7353.jpg', 0.9967759881173655), ('7369.jpg', 0.9968239616630546), ('7350.jpg', 0.9969730414694561), ('7358.jpg', 0.9970131108924257), ('7382.jpg', 0.9970333168), ('7385.jpg', 0.9970961065893013), ('7359.jpg', 0.9971140003409745), ('7376.jpg', 0.9971345777878991), ('7363.jpg', 0.9973246143163828), ('7386.jpg', 0.997339833961002), ('7351.jpg', 0.99746901581), ('7368.jpg', 0.9974776235719524), ('7379.jpg', 0.997699716171785), ('7389.jpg', 0.9978130785492952), ('7394.jpg', 0.997898589174276), ('7395.jpg', 0.9979386811875622), ('7355.jpg', 0.997982945217), ('7348.jpg', 0.9980640337277943), ('7370.jpg', 0.998100112515997), ('7383.jpg', 0.9983582159106708), ('7349.jpg', 1.0000000000000002)]
```

查找到的图像个数：51  
查全率：0.51

最好的 5 张为：7383 7370 7348 7355 7395



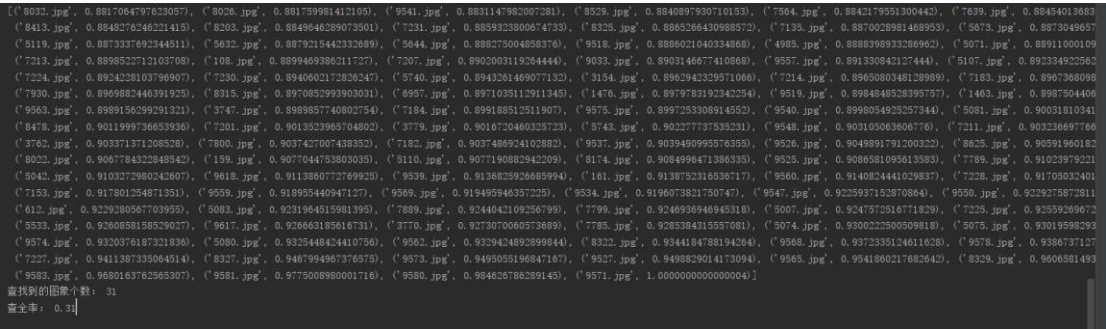
查找到 51 张相似的，查全率为 51%。

该类共有 52 张图（7344-7395），查准率为 98%

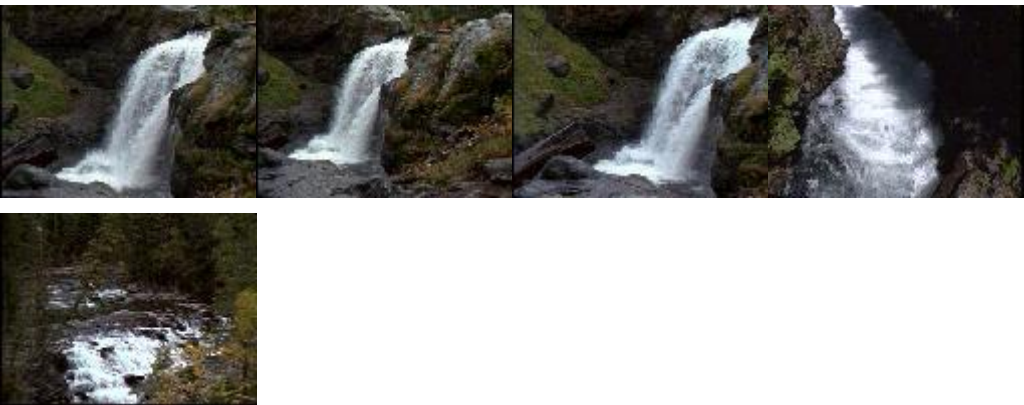


8. 图像八（9571）:

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%

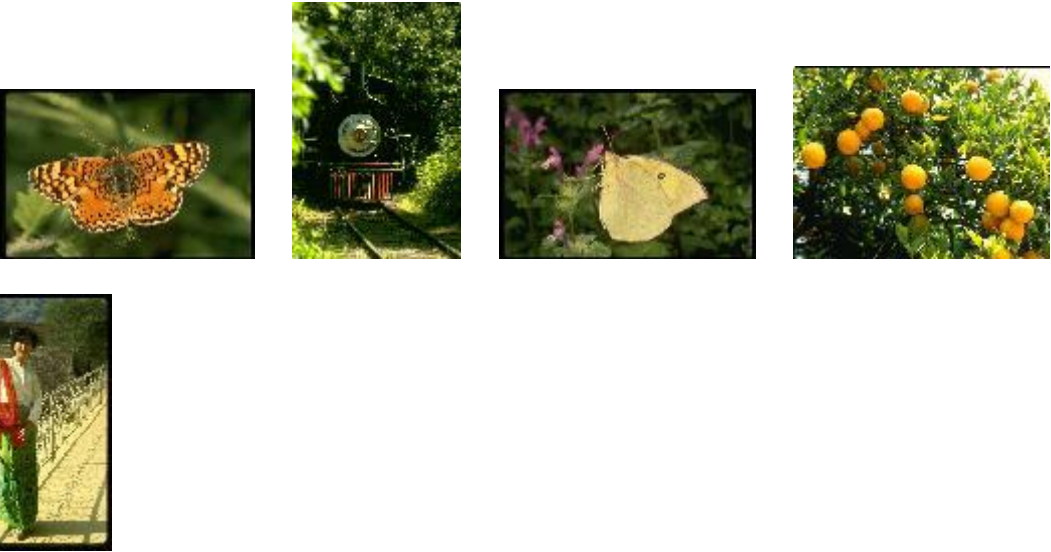


9. 图像九（19）:

100 张图片如下:

```
[('4250.jpg', 0.75499927612360549), ('1880.jpg', 0.7550122064079811), ('8577.jpg', 0.7569330813645738), ('3468.jpg', 0.7571115885939765), ('204.jpg', 0.7573459301139218), ('490.jpg', 0.757677151529  
( '69.jpg', 0.7582565738450489), ('4061.jpg', 0.7583581062199412), ('3355.jpg', 0.758917579277841), ('8291.jpg', 0.759112191114833), ('3675.jpg', 0.7593019880428050), ('91.jpg', 0.76005997586591  
( '3486.jpg', 0.7604884737360247), ('893.jpg', 0.7605237851476327), ('3588.jpg', 0.760738296170465), ('370.jpg', 0.7609924091994863), ('8508.jpg', 0.7616091574432272), ('4623.jpg', 0.7616322565027  
( '4661.jpg', 0.7618927670251612), ('8268.jpg', 0.7622483384187675), ('9651.jpg', 0.7623875451497644), ('4621.jpg', 0.7633304215974888), ('3600.jpg', 0.7638004157524878), ('4692.jpg', 0.7639536051  
( '145.jpg', 0.7639883293574661), ('3641.jpg', 0.7645885460217784), ('425.jpg', 0.7653248317716148), ('2282.jpg', 0.7653912073050064), ('4652.jpg', 0.7665470120246288), ('88.jpg', 0.76665635405039  
( '99.jpg', 0.76753334477469037), ('6.jpg', 0.768278887320079), ('4347.jpg', 0.7683683895196131), ('3435.jpg', 0.7700712229440643), ('4273.jpg', 0.770737580237981), ('1313.jpg', 0.7708301855814653  
( '5030.jpg', 0.7728772716347444), ('1427.jpg', 0.7730125059567888), ('4174.jpg', 0.7732944816821908), ('1580.jpg', 0.7738413256737889), ('3660.jpg', 0.773751722168384), ('239.jpg', 0.774339735691  
( '1179.jpg', 0.7746756935886384), ('1190.jpg', 0.7750818433044069), ('982.jpg', 0.7757808743299889), ('82.jpg', 0.7764426413463934), ('9293.jpg', 0.7782689507063786), ('3497.jpg', 0.7785240178569  
( '39.jpg', 0.7786844649620343), ('5641.jpg', 0.7792173739358084), ('1204.jpg', 0.7799423717253617), ('3637.jpg', 0.7803039447493461), ('1294.jpg', 0.7809861341688271), ('3592.jpg', 0.781062840112  
( '1111.jpg', 0.78123715277986650), ('473.jpg', 0.781262494174791), ('8582.jpg', 0.7819141982083899), ('5036.jpg', 0.7820800459162715), ('70.jpg', 0.7824226497830047), ('3297.jpg', 0.78292464033666  
( '227.jpg', 0.7834612677229336), ('4520.jpg', 0.7839887691450046), ('15.jpg', 0.7843278583414973), ('1174.jpg', 0.7847737940299652), ('3494.jpg', 0.7854799484521257), ('4657.jpg', 0.7854965991072  
( '1226.jpg', 0.7856194751615754), ('4561.jpg', 0.788510515871897), ('3608.jpg', 0.7885897434421419), ('3510.jpg', 0.7890677546849069), ('8580.jpg', 0.7936458970813588), ('42.jpg', 0.7938783599406  
( '1172.jpg', 0.7943890587870207), ('3444.jpg', 0.7952516785612904), ('2105.jpg', 0.7965313201007155), ('3671.jpg', 0.7970909817465094), ('1146.jpg', 0.7982674210185544), ('1132.jpg', 0.7992657153  
( '357.jpg', 0.8016871647148432), ('34.jpg', 0.8019977784468176), ('62.jpg', 0.803868674164812), ('1421.jpg', 0.8046193720798091), ('4565.jpg', 0.8052129934476507), ('1516.jpg', 0.8059278669161808  
( '8384.jpg', 0.8069355740398112), ('90.jpg', 0.8083171513094068), ('8581.jpg', 0.8085954231539831), ('44.jpg', 0.8127552192388184), ('3.jpg', 0.8183626102692944), ('2219.jpg', 0.819195838282592)  
( '1280.jpg', 0.8198853300333614), ('1274.jpg', 0.8212659872957753), ('3756.jpg', 0.8313245661177372), ('5643.jpg', 0.8314043281907894), ('3604.jpg', 0.8342547643712201), ('4178.jpg', 0.8391203379  
( '17.jpg', 0.8431749164712473), ('571.jpg', 0.8462659899327123), ('9.jpg', 0.8594219611838178), ('19.jpg', 1.0000000000000002)]  
查找到的图像个数：19  
查全率：0.19
```

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：

```

('1799.jpg', 0.7918328094871971), ('2920.jpg', 0.7829774743594493), ('3897.jpg', 0.753257946087937), ('3429.jpg', 0.7542073767031290), ('514.jpg', 0.7588299547190332), ('3130.jpg', 0.758733359561),
('3087.jpg', 0.7557544157514198), ('4592.jpg', 0.7561487686872626), ('2876.jpg', 0.7561940848307266), ('2582.jpg', 0.7564489801386375), ('1483.jpg', 0.756492738325982), ('1594.jpg', 0.75651257029),
('8485.jpg', 0.7571209189572368), ('1832.jpg', 0.7572485063520816), ('2675.jpg', 0.7573529418062303), ('1804.jpg', 0.7575245300237076), ('2784.jpg', 0.7575391913872320), ('9354.jpg', 0.7578715271),
('3288.jpg', 0.7580941205701573), ('273.jpg', 0.7582083955491008), ('3028.jpg', 0.7583805497694077), ('1711.jpg', 0.758430072723143), ('1826.jpg', 0.7586190212713689), ('2662.jpg', 0.76000848163),
('4002.jpg', 0.7604445005443758), ('1705.jpg', 0.7605477205706498), ('6484.jpg', 0.7607993039011468), ('3129.jpg', 0.7613809460831101), ('1706.jpg', 0.7619031138586349), ('8454.jpg', 0.7623878349),
('173.jpg', 0.7637145682160159), ('512.jpg', 0.7640402195885352), ('1786.jpg', 0.7641509623250644), ('4435.jpg', 0.7659321292866729), ('811.jpg', 0.7660340410796258), ('3896.jpg', 0.7663320598018),
('3789.jpg', 0.766430701462737), ('2925.jpg', 0.766525644656216), ('1877.jpg', 0.7673482355466514), ('1766.jpg', 0.7686361581305159), ('1841.jpg', 0.7698192649591317), ('1760.jpg', 0.77332306115),
('1722.jpg', 0.77387508232058678), ('6042.jpg', 0.77623262507261599), ('1975.jpg', 0.77667128576344), ('435.jpg', 0.7767463710283414), ('2097.jpg', 0.7767474637540837), ('6439.jpg', 0.7779920290998),
('6034.jpg', 0.7789954892231961), ('1846.jpg', 0.7806355992105675), ('6049.jpg', 0.7812458208141677), ('6083.jpg', 0.7812458208141677), ('3028.jpg', 0.7824105776544176), ('4451.jpg', 0.7836321148),
('9739.jpg', 0.7847027539924097), ('4214.jpg', 0.784933922736241), ('1715.jpg', 0.7856616834199813), ('1511.jpg', 0.785800010630429), ('5996.jpg', 0.7864131079482249), ('6045.jpg', 0.78641310794),
('6482.jpg', 0.7873982219690314), ('1870.jpg', 0.7880355147092883), ('889.jpg', 0.7885354007933901), ('1748.jpg', 0.7907577764749775), ('6039.jpg', 0.7920119668139564), ('1881.jpg', 0.79222703197),
('3830.jpg', 0.7923238866309773), ('1738.jpg', 0.79689155056492), ('1747.jpg', 0.800595057356237), ('1757.jpg', 0.8007210089561715), ('1771.jpg', 0.8008754880736838), ('3052.jpg', 0.8044527051507),
('6035.jpg', 0.8048197987912704), ('3116.jpg', 0.8059432202020021), ('999.jpg', 0.8054361527391722), ('8428.jpg', 0.80657419167599128), ('3845.jpg', 0.8070418573102713), ('3072.jpg', 0.80726084394),
('1788.jpg', 0.8092017047210931), ('6043.jpg', 0.8096864749210484), ('437.jpg', 0.811640167625358), ('6038.jpg', 0.8128445473232061), ('2447.jpg', 0.8158821650249313), ('1782.jpg', 0.81691623300),
('1729.jpg', 0.8176259349110834), ('1772.jpg', 0.819258616425508), ('2234.jpg', 0.8226819604591133), ('6036.jpg', 0.8230435790974591), ('1738.jpg', 0.8240643616854054), ('3100.jpg', 0.82433072788),
('2444.jpg', 0.8271265927991049), ('1791.jpg', 0.8276926174482498), ('6037.jpg', 0.8295546638688069), ('1773.jpg', 0.8297377132099594), ('9379.jpg', 0.8300507758115798), ('549.jpg', 0.83005567836),
('1750.jpg', 0.8302969014101539), ('1720.jpg', 0.8335035798287847), ('1728.jpg', 0.8342769123198154), ('1732.jpg', 1.0000000000000002)]
查找到的图像个数: 32
查全率: 0.32

```

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)

```