**图片检索**

**一．实验内容：**

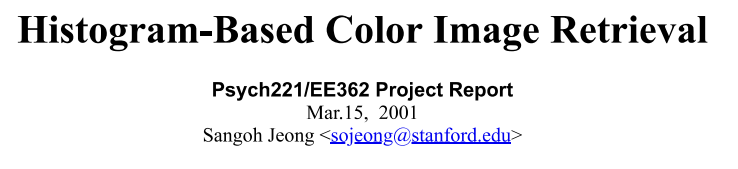
在指定的图片数据库内，完成图片检索。图像数据按照文件名的次序大概100幅为1类，以颜色直方图为依据，用最邻近方法或其他方法，随机抽取一张图片，利用模板匹配方法在图片库中找出100幅最相似的图片，计算查全率和查准率。

报告最后部分展示10幅来自不同类别的图片，计算查全率和查准率，展示最接近的5幅图片，并分析结果。

**数据集为**

http://www.vision.caltech.edu/Image\_Datasets/Caltech101/Caltech101.html

**参考文献** Histogram-Based Color Image Retrieval



**检索方法：**

采用图像的RGB三通道直方图与hsv直方图，利用欧几里得距离（Euclid），卡方比较（Chi-Square），Bhattacharyya distance（巴氏距离），相关性比较（Correlation），计算与模板的距离，找到最相近的图片。

**影响性能的参数：**

rgb的直方图bins，hsv直方图的h，s的bins，以及采用的检索模式。

**利用API：**

cv2. calcHist()

cv2.compareHist()

**二．实验环境：**

主机配置: CPU :intel core i5-7300 2.50GHZ RAM :8.0GB

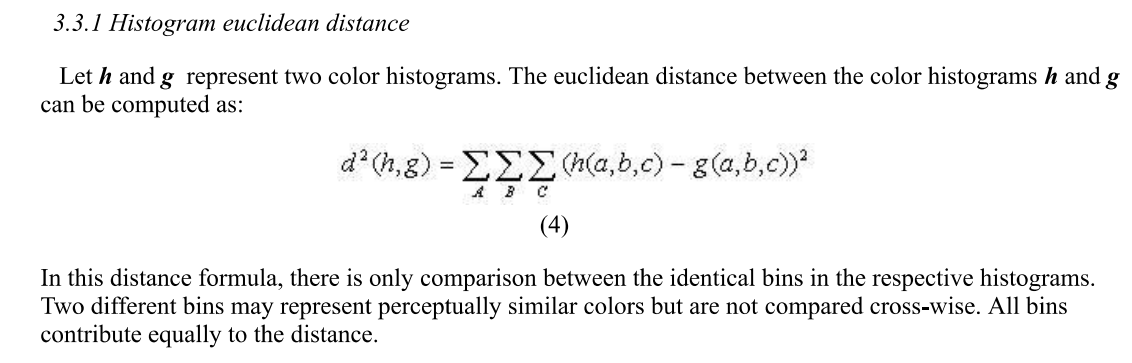
运行环境：win10 64位操作系统

开发环境：python3.7

1. **核心算法原理：**

**采用以下四种方式计算图片与标准图片的距离，选择最接近的标准，认为图片属于该组。**

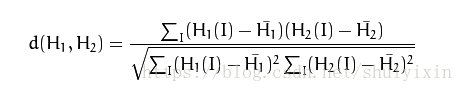
**1.欧几里得距离（Euclid distance）**



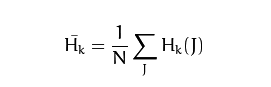
**两幅图的直方图欧几里得距离范围为0-∞。越接近0，图像越相似。**

1. **相关性比较（Correlation）**

相关性比较公式如下：

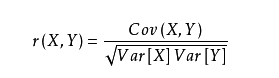


        其中

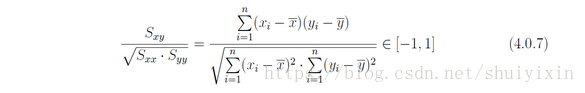


        如果H1 = H2，即两个图的直方图一样，分子等于分母，值为1，所以在不严格的情况下，当值为1时，可以认为两个图是一样的。但是也有可能会出现两个图不一样，但是两个图的直方图是一样的情况。因为直方图计算的是像素点个数的分布情况，但是不会显示像素点的位置，所以有可能会出现两幅图片不一样，但是相同像素的个数完全一样，那他们的直方图也是一样的，不过这种情况，不常有。

        相关性比较公式来源于统计学中的相关系数，最早由统计学家卡尔·皮尔逊设计的统计指标，是研究变量之间线性相关程度的量，一般用字母 r 表示。



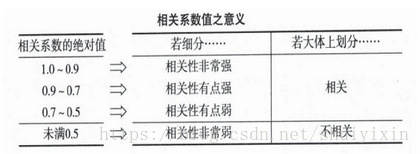
        其中，Cov(X,Y)为X与Y的协方差，Var[X]为X的方差，Var[Y]为Y的方差。



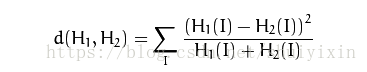
        下面是该公式更加详细的取值范围分析：并且如果两个变量的相关性越强，相关系数就会越接近±1，相关性越弱，相关系数越接近0。

        相关系数的值若为正值，称为正相关；相关系数的值若为负值，称为负相关；相关系数的值为0，称为不相关。

**故两幅图的直方图相关系数范围-1-1，越接近1越相似。**



**3.Chi-Square（卡方比较）**



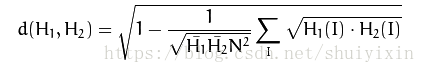
    通过这个公式我们能够发现，卡方比较和相关性比较恰恰相反，相关性比较的值为0，相似度最低，越趋近于1，相似度越低；卡方比较则是，值为0时说明H1= H2，这个时候相似度最高。

        卡方比较来源于卡方检验，卡方检验就是统计样本的实际观测值与理论推断值之间的偏离程度，实际观测值与理论推断值之间的偏离程度就决定卡方值的大小，卡方值越大，越不符合；卡方值越小，偏差越小，越趋于符合，若两个值完全相等时，卡方值就为0，表明理论值完全符合。卡方检验的公式如下，其中fi是观测频率，npi是期望频率，X²是卡方值。

IMG_256

**两幅图的卡方比较范围0-1，越接近0越相似。**

**4.Bhattacharyya distance 巴氏距离**



 在直方图相似度计算时，巴氏距离获得的效果最好，但计算是最为复杂的。巴氏距离的计算结果，其值完全匹配为1，完全不匹配则为0。

        在统计学中，巴氏距离（巴塔恰里雅距离 / Bhattacharyya distance）用于测量两离散概率分布。它常在分类中测量类之间的可分离性。在同一定义域X中，概率分布p和q的巴氏距离定义如下：离散概率分布；连续概率分布。

**两幅图的直方图巴氏距离范围0-1，越接近0越相似。**

1. **处理步骤：**

Step1:计算图片库内图片的RGB三通道直方图，hsv直方图

Step2:选取10组图像，每组约100幅，每组取其中一幅图作为标准，共10个标准

Step3:遍历图片库图片，每幅图片与10个标准对比。

对比方法为分别对图片的RGB直方图，hsv直方图，按照欧几里得距离（Euclid），卡方比较（Chi-Square），Bhattacharyya distance（巴氏距离），相关性比较（Correlation）方法，计算与10个标准的距离，选择最接近的标准，判定图片属于该组。并记录距离

Step4:利用Step3得到的记录数据，对判定为属于某组的图片排序，选取距离最近的100幅图，认为这100幅图为最终结果，即判定为属于该组的检索结果。

Step5：对选取的10组图片，计算查全率，查准率，并在每组的检索结果中选择距离最近的5幅图片做展示。

Step6：计算10组图片的平均查全率，查准率，取值最高的检索方法及参数为最优方法。

**具体步骤分析见 六. 步骤分析**

**五．核心代码：**

'''

从下面10组中随机各选取一幅图片,

直接根据其颜色直方图在9908(0-9907)幅图片中按照最邻近方法找出最接近的图片(若超出100幅取最接近的100幅。若不足100幅则不足)

计算查全率查准率

并展示最接近的五福图片

0-99 butterfly 100 mark 1,第一组图片,选取19.bmp为模板

100-199 mountain 100 mark 2 125.bmp

300-399 mark3 376.bmp

700-799 luori 100 mark 4 747.bmp

800-899 花 mark5 850.bmp

899-998 tree 100 mark 6 940.bmp

1106-1205 mark 7 1177.bmp

1593-1692 mark 8 1596.bmp

8641 8740 mark 9 8655.bmp

9029 9128 mark 10 9037.bmp

'''

import cv2

import numpy as np

#转bmp

def jpg2bmp():

for i in range(0,9908):

nam='D:/image/'+str(i)+'.jpg'

nam1='D:/image/'+str(i)+'.bmp'

img = cv2.imread(nam, 1)

cv2.imwrite(nam1, img)

#计算图片的hsv histogram h,s为Hue, Saturation的bins

def calchsv(start,end,h,s):

histoh=[]

for i in range(start,end+1):

img = cv2.imread('D:/image/'+str(i)+'.bmp')

hsv = cv2.cvtColor(img,cv2.COLOR\_BGR2HSV)

hist1 = cv2.calcHist([hsv], [0, 1], None, [h, s], [0, 180, 0, 256])

#cv2.normalize(hist1, hist1, 0, 1, cv2.NORM\_MINMAX, -1); #归一化，后面数据处理的方便，其次是保证程序运行时收敛加快

histoh.append(hist1)

return histoh

#计算图片的rgb histogram 。分三个通道RGB计算，每个通道bins为bins，而后拼接为一个histogram.

def calcRGBhisto(start,end,bins):

histor=[] #histo storage all histogram nparrays

for i in range(start,end+1):

img = cv2.imread('D:/image/'+str(i)+'.bmp')

#histc = cv2.calcHist([img],[0,1,2],None,[bins,bins,bins],[0,255,0,255,0,255])

hist2 = cv2.calcHist([img],[2],None,[bins],[0,255]) #np.array RGB三通道

hist0 = cv2.calcHist([img],[0],None,[bins],[0,255]) #np.array

hist1 = cv2.calcHist([img],[1],None,[bins],[0,255]) #np.array

histf=np.vstack((hist0,hist1,hist2)) #histf,当前图片的直方图的nparray

#cv2.normalize(histf, histf, 0, 1, cv2.NORM\_MINMAX, -1);

histor.append(histf)

return histor

#imggroup为记录搜索数据的array。形如[[19,0,99],[125,100,199]] 意味第一组图片范围为0-99，选取19号图片为标准搜索

#hisstart,hisend为要做检索的图片范围，如0,9907

#method为采用的检索方式。如Euclid为根据图片的直方图的欧几里得距离，越短认为越接近

def search(imggroup,gsize,hisstart,hisend,method):

rgbeqpre=0 #平均查全率查准率

hsveqrecall=0

rgbeqrecall=0

hsveqpre=0

if (method=="Euclid"):

calmethod="Euclid"

elif (method=="Correlation"):

calmethod=cv2.HISTCMP\_CORREL

elif (method=="ChiSquare"):

calmethod=cv2.HISTCMP\_CHISQR

else:

calmethod=cv2.HISTCMP\_BHATTACHARYYA

rgbdis=[] #rgbdis记录认为属于某组的图片的距离该组标准的距离;如dis[1]记录所有认为属于第二组的图片的距离该组标准150.bmp的距离

hsvdis=[]

rgbresult=[] #rgbresult记录结果,为gsize维数组。从0开始index越小越接近标准。如rgbresult[0]数组记录判定为属于1组的图片index。如rgbresult[0][0]=19，即为19.bmp

hsvresult=[]

rgbindex=[] #rgbindex[0]数组记录判定为属于1组的图片index。

hsvindex=[]

for i in range(0,gsize+1):

rgbdis.append([])

hsvdis.append([])

rgbindex.append([])

hsvindex.append([])

ghisrgb=[] #选取为标准的图片的histogram数组。1维数组。如ghisrgb[0]为19.bmp的histogram

ghishsv=[]

gstart=[] #记录每组开始图片index

gend=[]

for i in range(0,gsize): #ghisrgb,ghishsv 记录各个选取图片的histogram gstart记录图片所属组的开始index，

ghisrgb.append(historgb[imggroup[i][0]]) #如第一组0-99，查找19.bmp，gstart[0]=0，gend[0]=99，ghisrgb[0]=histo[19]

ghishsv.append(histohsv[imggroup[i][0]])

gstart.append(imggroup[i][1])

gend.append(imggroup[i][2])

for i in range(hisstart,hisend+1): #在hisstart-hisend的范围内，计算图片距离所选取的每幅图片距离，选择最近的认为属于该组;cnt记录距离各组距离

tprgbdis=[]

tphsvdis=[] #记录当前图片采用rgb和hsvhistogram,距离选取出的图片的距离

for j in range(0,gsize):

if (calmethod=="Euclid"):

tempr=historgb[i]-ghisrgb[j]

tempr=np.power(tempr,2)

tempdis=np.sum(tempr)

tprgbdis.append(tempdis)

temph=histohsv[i]-ghishsv[j]

temph=np.power(temph,2)

tempdis=np.sum(temph)

tphsvdis.append(tempdis)

else:

match = cv2.compareHist(historgb[i],ghisrgb[j], calmethod)

tprgbdis.append(match)

match = cv2.compareHist(histohsv[i],ghishsv[j], calmethod)

tphsvdis.append(match)

#np.argsort()用于给数组排序，返回值为从小到大元素index的值.

#假设一个数组a为[0,1,2,20,67,3],使用numpy.argsort(a),返回值应为[0,1,2,5,3,4]

if(method=="Correlation"): #Correlation -1-1,越大越接近

tpp=np.argsort(tprgbdis)

final=tpp[gsize-1] #img i 与final组correlation最接近1，认为属于final组

rgbdis[final].append(tprgbdis[final]) #将i.img距离final组标准的距离记录到rgbdis[final]数组，同时其index i记录到rgbindex[final]数组。

rgbindex[final].append(i)

tpp=np.argsort(tphsvdis)

final=tpp[0] #img i 与final组euclid距离最短，认为属于final组

hsvdis[final].append(tphsvdis[final])

hsvindex[final].append(i)

else: #其余的三种方法越小越接近

tpp=np.argsort(tprgbdis)

final=tpp[0] #img i 与final组距离最短，认为属于final组

rgbdis[final].append(tprgbdis[final])

rgbindex[final].append(i)

tpp=np.argsort(tphsvdis)

final=tpp[0]

hsvdis[final].append(tphsvdis[final])

hsvindex[final].append(i)

#gstart[i],第i组起始index。如第0组0-99，gstart[0]=0,gend[0]=99;

#index[i][result[i][j]] index[i]为认为属于[i]组的图片的index数组.result[i]为认为属于i组的图片的dis升序排序后的索引序列

#例如 index[0]={0,24,5} 那么result[0][2]=5

for i in range(0,gsize):

rgbcorr=0 #corr 认为属于i组并且确实属于i组

hsvcorr=0

if(method=="Correlation"): #越大越好

#print(type(rgbdis))

nprd=np.array(rgbdis[i])

nphd=np.array(hsvdis[i])

rgbresult.append(np.argsort(-nprd)) #argsort 得到的result是dis降序排序后的索引序列，数组

hsvresult.append(np.argsort(-nphd)) #result数组中增加一个数组，如[]-->[[0,1,2]],判定属于第0组的图片index为0,1,2

else: #越小越好

rgbresult.append(np.argsort(rgbdis[i]))

hsvresult.append(np.argsort(hsvdis[i]))

#print(rgbresult[i].size,hsvresult[i].size)

if rgbresult[i].size<=100: #若判定为i组的图片不足100幅

rgbsize=rgbresult[i].size

else: #若判定为i组的图片超过100幅,取最近的100幅

rgbsize=100

if hsvresult[i].size<=100: #若判定为i组的图片不足100幅

hsvsize=hsvresult[i].size

else: #若判定为i组的图片超过100幅,取最近的100幅

hsvsize=100

for j in range(0,rgbsize): #result[i][0]理论上是被search的图片自身，imgnum

if (gstart[i]<=rgbindex[i][rgbresult[i][j]])&(rgbindex[i][rgbresult[i][j]]<=gend[i]):

rgbcorr=rgbcorr+1

for j in range(0,hsvsize): #result[i][0]理论上是被search的图片自身，imgnum

if (gstart[i]<=hsvindex[i][hsvresult[i][j]])&(hsvindex[i][hsvresult[i][j]]<=gend[i]):

hsvcorr=hsvcorr+1

rgbclose=[]

hsvclose=[] #取最接近的5幅图片展示

for k in range(0,6):

rgbclose.append(rgbindex[i][rgbresult[i][k]])

hsvclose.append(hsvindex[i][hsvresult[i][k]])

rgbpre=rgbcorr/(rgbsize)

rgbeqpre=rgbeqpre+rgbpre

rgbrecall=rgbcorr/100

rgbeqrecall=rgbeqrecall+rgbrecall

hsvpre=hsvcorr/(hsvsize)

hsveqpre=hsveqpre+hsvpre

hsvrecall=hsvcorr/100

hsveqrecall=hsveqrecall+hsvrecall

print("imgnumber %d\n"%(imggroup[i][0]))

print("using rgb histogram %s method 's precision:%.5f ,recall:%.5f\n"%(method,rgbpre,rgbrecall))

print("using hsv histogram %s method 's precision:%.5f ,recall:%.5f\n"%(method,hsvpre,hsvrecall))

print("rgb close 5\n")

print(rgbclose)

print("hsv close 5\n")

print(hsvclose)

print("rgbeqpre: %.5f,rgbeqrecall: %.5f,hsveqpre: %.5f,hsveqrecall: %.5f\n"%(rgbeqpre/gsize,rgbeqrecall/gsize,hsveqpre/gsize,hsveqrecall/gsize))

if \_\_name\_\_ == '\_\_main\_\_':

#jpg2bmp()

histostart = 0

histoend = 9907

histobins=32

h= 30 #180

s= 128 #256

historgb = calcRGBhisto(histostart,histoend,histobins)

histohsv = calchsv(histostart,histoend,h,s)

group=[[19,0,99],[125,100,199],[376,300,399],[747,700,799],[850,800,899],[940,899,998],

[1177,1106,1205],[1596,1593,1692],[8655,8641,8740],[9037,9029,9128]] #group=[[50,0,99],[137,100,199],[350,300,399]]

groupsize =10

search(group,groupsize,histostart,histoend,"Euclid")

#search(group,groupsize,histostart,histoend,"Correlation")

#search(group,groupsize,histostart,histoend,"ChiSquare")

#search(group,groupsize,histostart,histoend,"Bhattach")

**六．步骤分析：**

**概述：**

Main函数调用calRGBhisto计算图片库内图片的RGB三通道直方图，保存到historgb。

然后调用calchsv计算图片库内的图片的hsv直方图，保存到histohsv。

Group为检索数据。如[50,0,99]表示一组图片范围为0.img-99.img，选取50.img作为标准检索。

之后调用search，遍历图片库，检索图片并展示。

**函数分析：**

**calRGBhisto**

**参数**

histostart,histoend,histobins。Histostart，histoend为需要计算直方图的图片index范围。Histobins为计算的直方图的bins。

**返回值**

historgb，为记录了图片RGB直方图数据的list。

调用cv2.calcHist计算直方图。

**Calchsv**

**参数**

histostart,histoend,h,s。H，s为计算直方图时Hue, Saturation的bins。

**返回值**

Histohsv，为记录了图片hsv直方图数据的list。

调用api，cv2.calcHist计算直方图。

**Search**

**参数**

imggroup,gsize,hisstart,hisend,method。

Imggroup为检索数据，形如[[50,0,99],[150,100,199]]。表示检索的第一组index范围0-99，选取50.bmp作为标准。

gsize为imggroup 的组数。

hisstart,hisend为检索范围。如整个图片库为0,9907.

Method为采取的检索方法，包括Euclid，Correlation， CHI-Square，巴氏距离。

**输出**

**首先输出10组，每组各自的查全率，查准率，以及最接近的5幅图片。**

print("imgnumber %d\n"%(imggroup[i][0])) 表示当前展示图片的index。

print("using rgb histogram %s method 'sprecision:%.5f ,recall:%.5f\n"%(method,rgbpre,rgbrecall))

采用rgb直方图，method “method”的查全率和查准率。

print("using hsv histogram %s method 'sprecision:%.5f ,recall:%.5f\n"%(method,hsvpre,hsvrecall))

采用hsv直方图，method “method”的查全率和查准率。

print("rgb close 5\n") print(rgbclose) 采用rgb直方图展示的最接近的5幅图片。

print("hsv close 5\n") print(hsvclose) 采用hsv直方图展示的最接近的5幅图片。

**最后输出十组图片平均查全查准率rgbeqpre，rgbeqrecall，hsveqrecall，hsveqpre**

**功能实现**

采用遍历图片数据库的方式，对每幅图片，计算与imggroup中每组的标准的距离，计算方法由method规定。选取最接近的组，判定当前图片属于该组，记录到hsvdis[final].append(tphsvdis[final]) hsvindex[final].append(i)。 final为判定所属组，如final=0判定属于imggroup中第1组。

Hsvdis[i]数组记录判定为属于iD组的图片距离i组的标准的距离，hsvindex记录对应的图片index。

如hsvdis[0][2]=50，hsv[0][2]=5.表示index为5的图片属于0组，并且距离0组标准距离为50.

遍历完成后，对hsvdis[i]进行argsort排序，返回的是按升序排列的数据index。对越大越好的Correlation先取负。然后选择最小的100幅图的index，判定这100幅图属于i组。(hsv[i]若超过100幅，去掉超过部分，若不足不补全。)保存到hsvresult[i]数组。

之后计算查全率查准率，先根据i组的范围判定hsvresult[i]中图片是否判定正确。如group[0]范围0-99，hsvresult[0]数组中图片index若属于0-99，判定正确，否则错误，记录判定正确的个数为hsvcorr，hsvcorr/hsvresultize=precision，hsvcorr/100=recall。

之后选择hsvresult[i]中前5幅图进行展示。

最后输出十组图片平均查全查准率rgbeqpre，rgbeqrecall，hsveqrecall，hsveqpre

**Main**

调用calRGBhisto，calhsv，search。

其中histostart = 0，histoend = 500为检索范围。

histobins=32为RGB直方图的bins。 h= 30 s= 128 为hsv直方图的bins

group=[[50,0,99],[137,100,199],[350,300,399]]为检索数据。

search(group,groupsize,histostart,histoend,"Euclid") 为规定检索方法并检索。

search(group,groupsize,histostart,histoend,"Correlation")

search(group,groupsize,histostart,histoend,"ChiSquare")

search(group,groupsize,histostart,histoend,"Bhattach")

1. **性能分析:**

**时间性能：**

程序主要运行时间为计算RGB直方图和HSV直方图的时间。之后遍历图片库进行检索的时间很短。

**影响计算时间的参数为histobins（RGB直方图bins），h，s（hsv直方图bins）。以及histoend，histostart（检索范围）。**

****检索性能：****

****检索性能判定依据为precision查准率和recall查全率。****

****影响检索性能的参数****

**histobins（RGB直方图bins），h，s（hsv直方图bins），以及采取的检索方式（Euclid，ChiSquare，Correlation，BHATTACHARYYA distance）。以及采用的直方图是RGB直方图还是HSV直方图。**

****最优参数：****

**采用实验方法，根据计算结果的recall和precision判定最优的histobins，h，s，以及采取的检索方式（Euclid，ChiSquare，Correlation，BHATTACHARYYA distance），以及RGB与HSV直方图。**

****实验方法：****

**Histobins范围为16,32,64,128,256**

**h范围为30,60,90,180**

**s范围为32,64,128,256**

**直方图计算方法范围为hsv直方图与rgb直方图**

**Method范围为Euclid distance，ChiSquare,Correlation，BHATTACHARYYA Distance**

**直方图范围 hsv直方图与rgb直方图**

**分不同method进行检索。每个method下采取不同的Histobins，直方图计算方法等参数，计算10组图片的平均查全率查准率，使得二者最高的参数为此method下的最优参数。之后再比较四种method，表现最好的method为最优method。**

1. **结果展示：**

**以表格形式展示采用不同方法与参数得到的平均查全率，查准率。**

**选择表现最好的方法参数，展示10组图片分别的查全率，查准率，以及最接近的5幅图片。**

**部分情况下采用hsv直方图计算得出的检索图片少于5幅，未列入表中**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **计算模式method** | **直方图计算方法** | **RGB直方图bins** | **HSV直方图h，s bins** | **平均查全率** | **平均查准率** |
| **Euclid** | **Rgb** | **256** |  | **0.066** | **0.066** |
| **Euclid** | **Rgb** | **32** |  | **0.067** | **0.067** |
| **Euclid** | **Rgb** | **16** |  | **0.073** | **0.073** |
| **Euclid** | **Rgb** | **64** |  | **0.061** | **0.061** |
| **Euclid** | **Rgb** | **128** |  | **0.061** | **0.061** |
| **Euclid** | **Hsv** |  | **30,128** | **0.07** | **0.08117** |
| **Euclid** | **Hsv** |  | **60,128** | **0.068** | **0.08375** |
| **Euclid** | **Hsv** |  | **90,128** | **0.064** | **0.09388** |
| **Euclid** | **Hsv** |  | **180,128** | **0.064** | **0.10252** |
| **Euclid** | **Hsv** |  | **30,32** | **0.065** | **0.07784** |
| **Euclid** | **Hsv** |  | **60,32** | **0.059** | **0.07694** |
| **Euclid** | **Hsv** |  | **90,32** | **0.058** | **0.08303** |
| **Euclid** | **Hsv** |  | **30,64** | **0.07** | **0.082** |
| **Euclid** | **Hsv** |  | **60,64** | **0.061** | **0.0763** |
| **Euclid** | **Hsv** |  | **90,64** | **0.058** | **0.08** |
| **Euclid** | **Hsv** |  | **180,64** | **0.061** | **0.09841** |
| **Euclid** | **Hsv** |  | **180,256** | **0.066** | **0.0966** |
| **Euclid** | **Hsv** |  | **90,256** | **0.07** | **0.09183** |
| **Euclid** | **Hsv** |  | **60,256** | **0.075** | **0.086** |
| ****Correlation**** | **Rgb** | **256** |  | **0.073** | **0.073** |
| ****Correlation**** | **Hsv** |  | **30,256** | **0.006** | **0.006** |
| ****Correlation**** | **Rgb** | **128** |  | **0.065** | **0.065** |
| ****Correlation**** | **Hsv** |  | **90,128** | **0.003** | **0.003** |
| ****Correlation**** | **Hsv** |  | **180,64** | **0.005** | **0.005** |
| ****Correlation**** | **Rgb** | **64** |  | **0.065** | **0.065** |
| ****Correlation**** | **Rgb** | **32** |  | **0.069** | **0.069** |
| **ChiSquare** | **Rgb** | **32** |  | **0.056** | **0.056** |
| **ChiSquare** | **Hsv** |  | **30,32** | **0.026** | **0.026** |
| **ChiSquare** | **Rgb** | **64** |  | **0.058** | **0.058** |
| **ChiSquare** | **Hsv** |  | **60,32** | **0.023** | **0.023** |
| **ChiSquare** | **Rgb** | **128** |  | **0.059** | **0.059** |
| **ChiSquare** | **Hsv** |  | **90,32** | **0.025** | **0.025** |
| **ChiSquare** | **Rgb** | **256** |  | **0.06** | **0.06** |
| **ChiSquare** | **Hsv** |  | **180,32** | **0.024** | **0.024** |
| **ChiSquare** | **Hsv** |  | **180,64** | **0.024** | **0.02418** |
| **ChiSquare** | **Hsv** |  | **90,64** | **0.029** | **0.029** |
| **ChiSquare** | **Hsv** |  | **60,64** | **0.026** | **0.026** |
| **ChiSquare** | **Hsv** |  | **30,64** | **0.026** | **0.026** |
| **ChiSquare** | **Hsv** |  | **30,128** | **0.031** | **0.031** |
| **ChiSquare** | **Hsv** |  | **60,128** | **0.029** | **0.02911** |
| **ChiSquare** | **Hsv** |  | **90,128** | **0.029** | **0.02928** |
| **ChiSquare** | **Hsv** |  | **180,128** | **0.023** | **0.02341** |
| **ChiSquare** | **Hsv** |  | **180,256** | **0.025** | **0.02596** |
| **ChiSquare** | **Hsv** |  | **90,256** | **0.026** | **0.02661** |
| **ChiSquare** | **Hsv** |  | **60,256** | **0.03** | **0.03037** |
| **ChiSquare** | **Hsv** |  | **30,256** | **0.035** | **0.03512** |
| **Bhattach** | **Rgb** | **256** |  | **0.071** | **0.017** |
| **Bhattach** | **Hsv** |  | **30,256** | **0.124** | **0.124** |
| **Bhattach** | **Rgb** | **128** |  | **0.072** | **0.072** |
| **Bhattach** | **Hsv** |  | **60,256** | **0.131** | **0.131** |
| **Bhattach** | **Rgb** | **64** |  | **0.073** | **0.073** |
| **Bhattach** | **Hsv** |  | **90,256** | **0.133** | **0.133** |
| **Bhattach** | **Rgb** | **32** |  | **0.08** | **0.08** |
| **Bhattach** | **Hsv** |  | **180,256** | **0.125** | **0.125** |
| **Bhattach** | **Hsv** |  | **180,128** | **0.132** | **0.132** |
| **Bhattach** | **Hsv** |  | **90,128** | **0.128** | **0.128** |
| **Bhattach** | **Hsv** |  | **60,128** | **0.128** | **0.128** |
| **Bhattach** | **Hsv** |  | **30,128** | **0.123** | **0.123** |
| **Bhattach** | **Hsv** |  | **30,64** | **0.119** | **0.119** |
| **Bhattach** | **Hsv** |  | **60,64** | **0.128** | **0.128** |
| **Bhattach** | **Hsv** |  | **90,64** | **0.129** | **0.129** |
| **Bhattach** | **Hsv** |  | **180,64** | **0.131** | **0.131** |
| **Bhattach** | **Hsv** |  | **180,32** | **0.13** | **0.13** |
| **Bhattach** | **Hsv** |  | **90,32** | **0.123** | **0.123** |
| **Bhattach** | **Hsv** |  | **60,32** | **0.124** | **0.124** |
| **Bhattach** | **Hsv** |  | **30,32** | **0.121** | **0.121** |

**由实验结果可知，采用**BHATTACHARYYA Distance，hsv直方图，H bins=90，S bins=256时检索效果最好****

**最优参数：**

**Method = **BHATTACHARYYA Distance，H = 90,S=256, hsv histogram****

****下面按照上述最佳参数，对10组图片分组展示其查全率，查准率，最接近的5幅图片。****

****0-99 butterfly 100 mark 1,第一组图片,选取19.bmp为模板****

****100-199 mountain 100 mark 2 125.bmp****

****300-399 mark3 376.bmp****

****700-799 luori 100 mark 4 747.bmp****

****800-899 花 mark5 850.bmp****

****899-998 tree 100 mark 6 940.bmp****

****1106-1205 mark 7 1177.bmp****

****1593-1692 mark 8 1596.bmp****

****8641 8740 mark 9 8655.bmp****

****9029 9128 mark 10 9037.bmp****

****Group1****

****0-99****

****standard 19.bmp****

********

****Precision = 21%****

****Recall =21%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****15**** | ****893**** | ****1344**** | ****9605**** | ****9**** |
| ****15.bmp**** | ****893**** | ****1344**** | ****9605**** | ****9**** |

****Group2****

****100-199****

****standard 125.bmp****

********

****Precision = 11%****

****Recall =11%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****9205**** | ****2347**** | ****642**** | ****3271**** | ****8328**** |
| ****9205**** | ****2347**** | ****642**** | ****3271**** | ****8328**** |

****Group3****

****300-399****

****standard 376.bmp****

********

****Precision = 9%****

****Recall =9%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****3124**** | ****3886**** | ****927**** | ****257**** | ****3948**** |
| ****3124**** | ****3886**** | ****927**** | ****257**** | ****3948**** |

****Group4****

****700-799****

****standard 747.bmp****

********

****Precision = 7%****

****Recall =7%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****739**** | ****784**** | ****3723**** | ****1858**** | ****7644**** |
| ****739**** | ****784**** | ****3723**** | ****1858**** | ****7644**** |

****Group5****

****800-899****

****standard 850.bmp****

********

****Precision = 42%****

****Recall =42%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****8157**** | ****807**** | ****8407**** | ****847**** | ****856**** |
| ****8157**** | ****807**** | ****8407**** | ****847**** | ****856**** |

****Group6****

****899-998****

****standard 940.bmp****

********

****Precision = 7%****

****Recall =7%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****2566**** | ****3413**** | ****932**** | ****3437**** | ****3466**** |
| ****2566**** | ****3413**** | ****932**** | ****3437**** | ****3466**** |

****Group7****

****1106-1205****

****standard 1177.bmp****

********

****Precision = 4%****

****Recall =4%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****4093**** | ****8093**** | ****4623**** | ****8094**** | ****4068**** |
| ****4093**** | ****8093**** | ****4623**** | ****8094**** | ****4068**** |

****Group8****

****1593-1692****

****standard 1596.bmp****

********

****Precision = 14%****

****Recall =14%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****3092**** | ****388**** | ****992**** | ****3067**** | ****3397**** |
| ****3092**** | ****388**** | ****992**** | ****3067**** | ****3397**** |

****Group9****

****8641-8740****

****standard 8655.bmp****

********

****Precision =10%****

****Recall =10%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****8656**** | ****5685**** | ****8494**** | ****8516**** | ****8493**** |
| ****8656**** | ****5685**** | ****8494**** | ****8516**** | ****8493**** |

****Group10****

****9029-9128****

****standard 9037.bmp****

********

****Precision = 8%****

****Recall =8%****

****Closest 5****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ****7334**** | ****9032**** | ****4849**** | ****7332**** | ****7747**** |
| ****7334**** | ****9032**** | ****4849**** | ****7332**** | ****7747**** |

****结果分析：****

****在10组中，第5组表现最好，查全率42% 查准率42%****

****第7组表现最差，查全率为4%，查准率为4%****

****原因为采用的检索方法为直方图距离检索，而直方图只关心颜色在图像中所占的比例，主要考虑颜色，不考虑形状，且不考虑颜色分布，无法描述图像内的物体。****

****而数据库为人眼归类，分类依据为物体的现实属性，比如花，赛车，等等，这些物体属性相近但颜色，颜色分布，拍摄角度导致的形状等等很可能不同。****

****由上述展示的最接近的5幅图片可知，所采用的检索方法对颜色确实十分敏感，检索得到的图像与原图颜色比例十分接近。****

****当遇到同一类物体颜色形状都十分接近时检索效果好，比如第5组的粉色的花；当遇到物体颜色，分布，照片角度不同时，检索效果很差，比如第7组的人像。****