基于颜色直方图的图像检索与匹配

2019.10.24

王泽鹏 2017301510036 网安二班

概要

该实验通过对图像的RGB三个通道进行划分，形成特征向量，进而对给定图像进行相关特征值计算来与图库中其他图像进行相关性计算，以达到相似图片的检索与匹配的目的。

目录

1.简介

图像检索自20世纪70年代始便成为一个非常活跃的研究领域，其推动力来源于两大研究团体：数据库系统和计算机视觉，它们从基于文本以及基于内容这两个不同的角度，对图像检索进行了研究。20世纪90年代早期，由于大规模图像数据库的出现，传统的通过手工标记和索引图像的方法（基于文本的图像检索）已经无法满足人们的需求，因为图像数据不断增大，而人们对图像的理解有着不同的侧重点，不同的人对同一幅图像的认识可能存在着巨大的差异，导致无法准确描述图像信息，因为存在着这种问题，基于内容的图像检索应运而生。

在基于内容的图像检索中，如何根据图像的视觉特征快速准确地检索图像成为了关键问题。其中，颜色作为彩色图像中最显著的视觉特征，被广泛应用与图像检索中。颜色特征对图像本身尺寸、视角等依赖性较小，可以更为容易地得到抽象模型，以图像颜色特征为核心的基于颜色直方图的图像检索方法成为了一个有效途径。

2.基本原理

1.颜色直方图

颜色直方图所描述的是不同色彩在整幅图像中所占的比例，而并不关心每种色彩所处的空间位置，即无法描述图像中的对象或物体。颜色直方图也别适

于描述那些难以进行自动分割的图像。直方图中的数值都是统计得来的，描述了该图像中关于颜色的数量特征，可以反应图像颜色的统计分布和基本色调；包含了该图像中某一颜色值出现的频数，而丢失了某像素所在的空间位置信息。

颜色直方图可以是基于不同的颜色空间和坐标系，最常用的颜色空间是RGB颜色空间，本实验也以RGB颜色空间作为颜色直方图的选取空间进行统计。

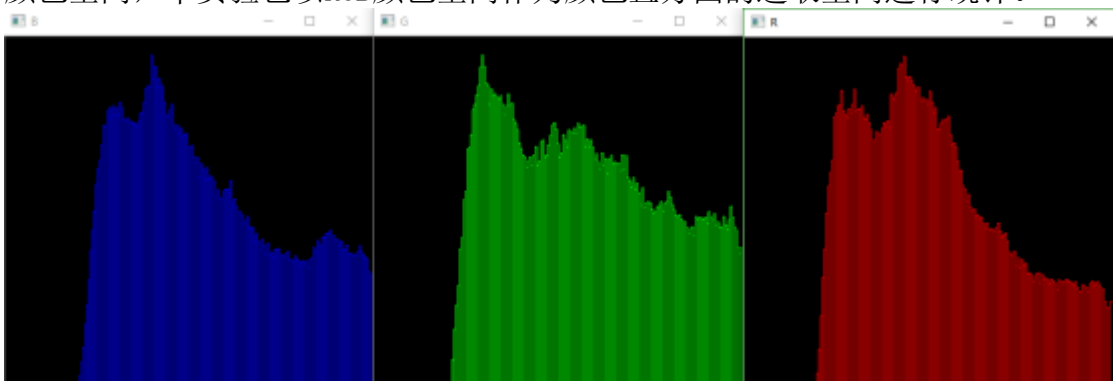

图1 RGB空间颜色直方图

彩色图像的直方图可以用R、G、B三个单色的直方图来表示，每个直方图表示的是图像在该颜色分量的统计分布，共三个不同的概率分布。

2.基于内容的图像检索

基于内容的图像检索是指先使用图像分析软件对输入图像进行分析，根据图像中物体或区域的颜色、形状或纹理等特征及特征的组合，选取合适的特征，连同输入图像一起存入图像数据库中。在之后的图像检索过程中，对于每一幅给定的图像，先进行图像分析，提取图像的特征向量，再将该图像的特征向量与图像数据库中的特征向量进行匹配，根据匹配的结果便可以得到图像库中与输入图像最相近的一张（或多张）图像，从而实现图像检索的目标。

3.具体实现

1.提取颜色直方图

任何一种颜色都是由红绿蓝（RGB）三原色组成的，每种原色都可以取256个值，则整个颜色空间共有256*256*256中颜色，要实现所有颜色的直方图，所需的计算时间及存储空间都过大，因此我们可以作适当简化。将0~255分为四个分区，0~63为第0区，64~127为第1区，128~191为第2区，192~255为第3区。按照这样划分后，每种原色都只有4个分区，三种原色总计可构成64种组合，即只需64种组合便可简略表示整个颜色空间中任意一种颜色。

如果我们使用RGB三个通道各自四个分区上的像素点的个数作为特征量，那么对于每一张图片，我们只需要12个值便可以得到一组描述该图像的特征向量。如下表，是部分图片经过特征提取后得到的特征向量表。

name	BGR
0.jpg	2414.0,6995.0,1410.0,61.0,1153.0,6650.0,2072.0,979.0,1194.0,6002.0,2626.0,801.0
1.jpg	8845.0,1641.0,379.0,15.0,4012.0,5019.0,1607.0,240.0,6413.0,3388.0,970.0,109.0
2.jpg	8872.0,1124.0,733.0,151.0,7990.0,1566.0,704.0,613.0,8109.0,1049.0,911.0,719.0
3.jpg	6819.0,2331.0,1609.0,121.0,5062.0,1653.0,1912.0,2127.0,4972.0,1811.0,2406.0,1592.0
4.jpg	10021.0,765.0,92.0,2.0,7860.0,2757.0,261.0,2.0,8939.0,1715.0,223.0,3.0
5.jpg	9280.0,873.0,716.0,11.0,8927.0,652.0,1003.0,298.0,8950.0,563.0,961.0,406.0
6.jpg	9751.0,740.0,389.0,0.0,8041.0,1897.0,742.0,200.0,8525.0,1148.0,710.0,497.0
7.jpg	10037.0,487.0,342.0,14.0,7730.0,2555.0,365.0,230.0,9414.0,784.0,384.0,296.0
8.jpg	5000.0,3117.0,2563.0,200.0,3853.0,1714.0,2951.0,2340.0,3783.0,1358.0,2293.0,3162.0
9.jpg	9301.0,1431.0,142.0,6.0,4619.0,4258.0,1763.0,240.0,5391.0,2943.0,1443.0,977.0
10.jpg	7689.0,2387.0,770.0,34.0,7020.0,1115.0,2275.0,470.0,6876.0,755.0,1499.0,1679.0
11.jpg	10143.0,559.0,176.0,2.0,3797.0,6632.0,401.0,50.0,5247.0,5115.0,435.0,83.0
12.jpg	7830.0,2853.0,197.0,0.0,2249.0,7656.0,940.0,35.0,3257.0,6508.0,1070.0,45.0
13.jpg	9176.0,1225.0,461.0,18.0,7406.0,2676.0,671.0,127.0,7417.0,2239.0,820.0,397.0
14.jpg	6993.0,3303.0,571.0,13.0,4834.0,4805.0,1187.0,54.0,4499.0,4871.0,1420.0,90.0
15.jpg	9705.0,1034.0,141.0,0.0,2724.0,7316.0,745.0,95.0,3304.0,6719.0,679.0,178.0
16.jpg	7898.0,2520.0,460.0,2.0,2770.0,5487.0,2156.0,467.0,1676.0,5138.0,2957.0,1050.0
17.jpg	8467.0,1894.0,517.0,2.0,5796.0,3162.0,852.0,1070.0,6668.0,2228.0,410.0,1566.0
18.jpg	3262.0,4396.0,2933.0,289.0,1670.0,3366.0,4495.0,1337.0,1634.0,3314.0,4391.0,1533.0
19.jpg	7896.0,2510.0,465.0,9.0,3470.0,3671.0,1804.0,1922.0,4392.0,2465.0,2513.0,1281.0
20.jpg	8634.0,1612.0,454.0,163.0,6963.0,2714.0,734.0,468.0,7339.0,2364.0,833.0,343.0

图2 部分图片的特征向量表

使用到提取每个通道每个分区像素点数目的关键函数为cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])，该函数有5个参数：

①image输入图像，传入时应该用中括号[]括起来

②channels:：传入图像的通道，如果是灰度图像，那就不用说了，只有一个通道，值为0，如果是彩色图像（有3个通道），那么值为0,1,2,中选择一个，对应着BGR各个通道。这个值也得用[]传入。

③mask：掩膜图像。如果统计整幅图，那么为none。主要是如果要统计部分图的直方图，就得构造相应的掩膜来计算。

④histSize：灰度级的个数，需要中括号，比如[256]

⑤ranges:像素值的范围，通常[0,256]，有的图像如果不是0-256，比如说你来回各种变换导致像素值负值、很大，则需要调整后才可以。

以提取B通道的直方图参数为例，使用代码为：

```
histB = cv2.calcHist([img], [0], None, [4], [0.0, 255.0])
```

其中，[0]表示对B通道进行计算，[4]将区间分为四部分，[0.0,255.0]表示区间的范围。如此对三个通道进行计算后，便可得到一幅图像的特征向量，相关代码如下：

```
for i in range(0,number): #因为从0开始，同时range右值不取
     img = cv2.imread('D:/file/image.vary.jpg/%s.jpg'%i)
     histB = cv2.calcHist([img], [0], None, [4], [0.0, 255.0])
     histG = cv2.calcHist([img], [1], None, [4], [0.0, 255.0])
     histR = cv2.calcHist([img], [2], None, [4], [0.0, 255.0])

     arr = [0 for x in range(0,12)]
     for j in range(0,4):
         arr[j]=histB[j][0]
```

```
    for j in range(4,8):
        arr[j]=histG[j-4][0]
    for j in range(8,12):
        arr[j]=histR[j-8][0]

    strRGB = ','.join(str(m) for m in arr)
    value = [["%s.jpg"%i, strRGB],]

    write_excel_xls_append(book_name_xls, value)
```

2. 将数据写入数据库或excel表

我们可以将所有图片按照上述方法得到的特征向量集存在一个数据库的表中，也可以利用python对excel表操作的方法将特征向量集存在一张excel表中，这里我们使用了后者。

对excel表进行操作的基本函数如下：

```
def write_excel_xls(path, sheet_name, value):
    index = len(value)  # 获取需要写入数据的行数
    workbook = xlwt.Workbook()  # 新建一个工作簿
    sheet = workbook.add_sheet(sheet_name)  # 在工作簿中新建一个表格
    for i in range(0, index):
        for j in range(0, len(value[i])):
            sheet.write(i, j, value[i][j])  # 像表格中写入数据（对应的行和列）
    workbook.save(path)  # 保存工作簿
    print("xls格式表格写入数据成功！")

def write_excel_xls_append(path, value):
    index = len(value)  # 获取需要写入数据的行数
    workbook = xlrd.open_workbook(path)  # 打开工作簿
    sheets = workbook.sheet_names()  # 获取工作簿中的所有表格
    worksheet = workbook.sheet_by_name(sheets[0])  # 获取工作簿中所有表格中的的第一个表格
    rows_old = worksheet.nrows  # 获取表格中已存在的数据的行数
    new_workbook = copy(workbook)  # 将xlrd对象拷贝转化为xlwt对象
    new_worksheet = new_workbook.get_sheet(0)  # 获取转化后工作簿中的第一个表格
    for i in range(0, index):
        for j in range(0, len(value[i])):
            new_worksheet.write(i + rows_old, j, value[i][j])  # 追加写入数据，注意是从i+rows_old行开始写入
    new_workbook.save(path)  # 保存工作簿
    # print("xls格式表格【追加】写入数据成功！")


def read_excel_xls(path, row, col):
    workbook = xlrd.open_workbook(path)  # 打开工作簿
```

```
    sheets = workbook.sheet_names()  # 获取工作簿中的所有表格
    worksheet = workbook.sheet_by_name(sheets[0])  # 获取工作簿中所有表格中的的第一个表格
    temp = worksheet.cell_value(row, col)
    return temp
```

三个函数分别为：创建excel表、向excel表中添加一行数据、读excel表中某个单元格的数据。

3. 图像检索匹配

在建立图像特征向量信息表后，就可以对任意输入图像进行匹配检索了。以同样方法获取到输入图像的特征向量后，将其与表中的每一项进行相关性度量，找出相关性最大的100张（或5张）图像，即完成了相似图像的检索匹配。

我们进行相关性度量而使用的是皮尔逊相关系数。皮尔逊相关系数广泛用于度量两个变量之间的相关程度，其值介于-1与1之间，取值为1时，表示两个随机变量之间呈完全正相关关系；取值为-1时，表示两个随机变量之间呈完全负相关关系；取值为0时，表示两个随机变量之间线性无关。

两个变量之间的皮尔逊相关系数定义为两个变量之间的协方差和标准差的商：

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y},$$

估算样本的协方差和标准差，可以得到皮尔逊相关系数，用r代表：

$$r = \frac{\sum_{i=1}^{n}(X_i-\overline{X})(Y_i-\overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\overline{Y})^2}}.$$

因此，我们可以对输入图像与图像库中每一张图像的特征向量进行皮尔逊相关系数计算，将结果存储到数组中，最后对数组进行最大值的选取，便可得到前100张（或前5张）最相似图像的序号。相关关键代码如下：

```
pearson_max = []
for j in range(0,9908):
    print(j)
    strDb = read_excel_xls(book_name_xls, j+1, 1)#读取字符串
    arrDb = strDb.split(',')  # 字符串转char型数组
    histDb = list(map(float, arrDb))  # char型数组转float型
    #print(pearsonr(arr,histDb)[0])
    pearson_max.append(pearsonr(arr,histDb)[0])

max_pearson_index_list = map(pearson_max.index, heapq.nlargest(100, pearson_max))
max_pearson_index_list=list(max_pearson_index_list)
print(max_pearson_index_list)
print("与输入图片%s.jpg最相近的100张图片为：" %i)
for i in range(0,100):
```

```
    x = max_pearson_index_list[i]
    print("%s.jpg" %x)
```

4. 结果演示

1. 随机抽取一张图片，在图片库中找出100幅最相似的图片。

我们以1. jpg作为输入图片进行检索匹配，得到输出图片序列如下图：

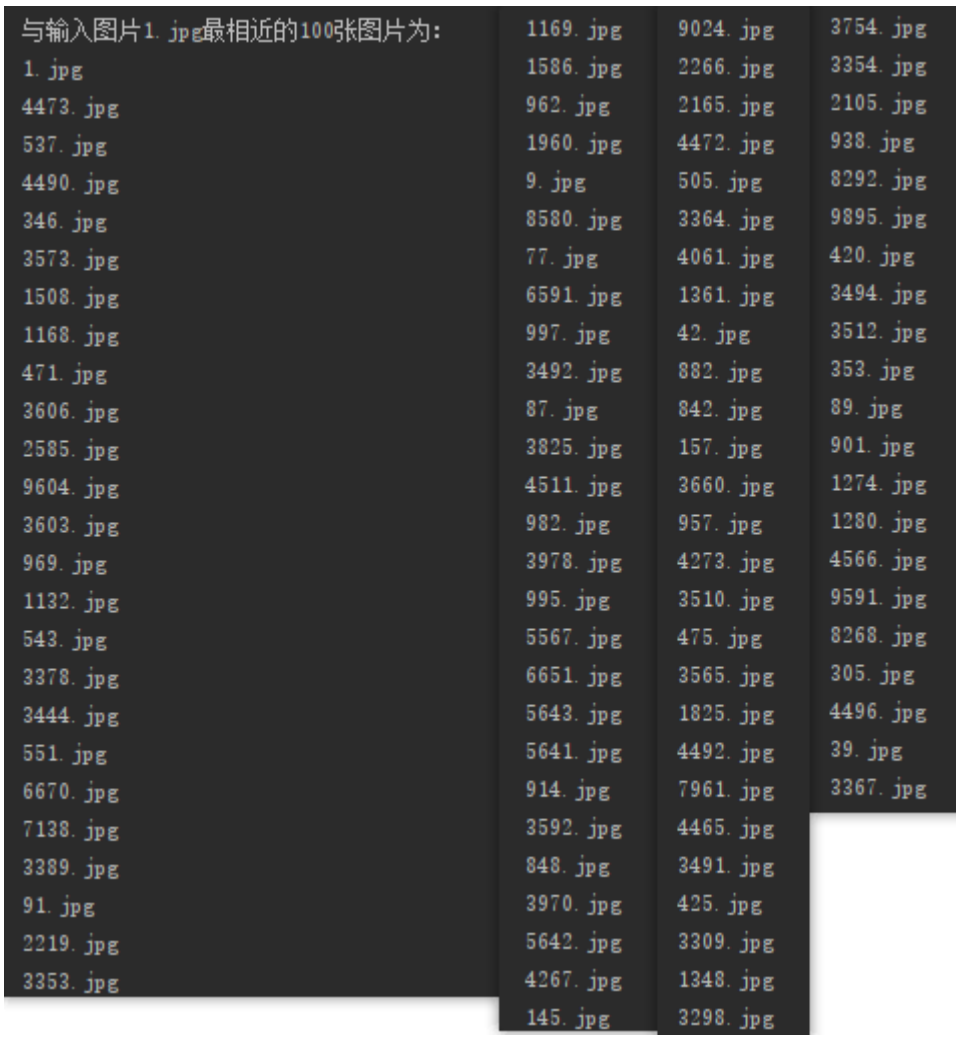

图3　与1. jpg相似度最高的100张图片检索结果

在图片库中查找序号对应的部分图片，结果如下：

图4　1. jpg

图5 与1.jpg相似度最高的部分图像

2.分别从不同类别（大致每100张为一类）输入共10幅图片，给出每一幅图片最接近的5幅图片。

10幅图片输入分别为736.jpg、4895.jpg、5303.jpg、948.jpg、3722.jpg、2370.jpg、883.jpg、6286.jpg、6767.jpg、8228.jpg，输出结果如下（五幅图片中的第一幅图片为输入原图）：

图6 与736.jpg相似度最高的5张图像

图7 与4895.jpg相似度最高的5张图像

图8 与5303.jpg相似度最高的5张图像

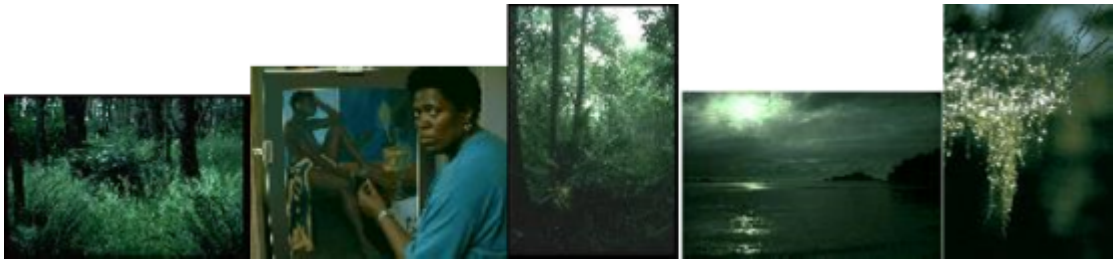

图9 与948.jpg相似度最高的5张图像

图10 与3722.jpg相似度最高的5张图像

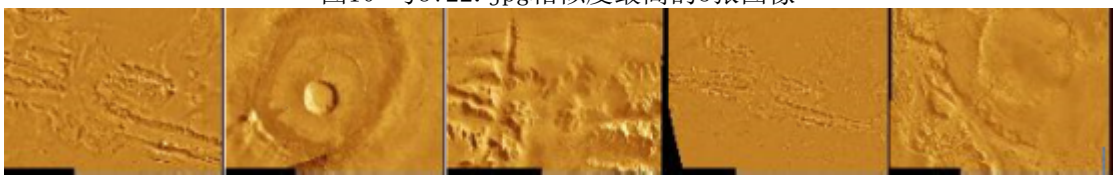

图11 与2370.jpg相似度最高的5张图像

图12 与883.jpg相似度最高的5张图像

图13 与6286.jpg相似度最高的5张图像

图14 与6767.jpg相似度最高的5张图像

图15 与8228.jpg相似度最高的5张图像

5.结论

1.任意输入图像，输出100幅最相近图像的结果

输入图像1.jpg类别号为1，同属于该类别的输出结果数为8，故查准率为8%，查全率为8%。

2.不同类别的10幅图像，输出5幅最相近图像的结果

输入图像 匹配结果		736.jpg	4895.jpg	5303.jpg	948.jpg	3722.jpg
结果1	文件名	736.jpg	4895.jpg	5303.jpg	948.jpg	3722.jpg
	相关度	1	0.999	0.999	0.999	0.999
结果2	文件名	2535.jpg	4896.jpg	5304.jpg	1143.jpg	4498.jpg
	相关度	0.952	0.999	0.999	0.995	0.991

结果3	文件名	1115.jpg	4894.jpg	5306.jpg	3285.jpg	4665.jpg
	相关度	0.945	0.998	0.999	0.995	0.989
结果4	文件名	801.jpg	8965.jpg	5305.jpg	2194.jpg	1855.jpg
	相关度	0.944	0.991	0.999	0.992	0.988
结果5	文件名	821.jpg	6589.jpg	6837.jpg	478.jpg	1302.jpg
	相关度	0.944	0.991	0.998	0.992	0.987
查全率		1%	3%	4%	1%	1%
查准率		20%	60%	80%	20%	20%

匹配结果	输入图像	2370.jpg	883.jpg	6286.jpg	6767.jpg	8228.jpg
结果1	文件名	2370.jpg	883.jpg	6286.jpg	6767.jpg	8228.jpg
	相关度	0.999	1.000	0.999	1	1
结果2	文件名	2372.jpg	806.jpg	8213.jpg	6770.jpg	8229.jpg
	相关度	0.999	0.987	0.957	0.969	1
结果3	文件名	2375.jpg	861.jpg	6403.jpg	6765.jpg	8238.jpg
	相关度	0.995	0.979	0.950	0.963	0.997
结果4	文件名	2369.jpg	4000.jpg	6141.jpg	6769.jpg	8242.jpg
	相关度	0.995	0.977	0.949	0.948	0.996
结果5	文件名	2373.jpg	1675.jpg	8230.jpg	6763.jpg	8236.jpg
	相关度	0.992	0.976	0.940	0.940	0.996
查全率		5%	3%	3%	5%	5%
查准率		100%	60%	60%	100%	100%

3.结论

我们由上述实验结果可以发现，基于颜色直方图的图像检索因为只关注不同色彩在图像中所占的比例，而忽略了每种色彩所处的空间位置，对图像物体的形状、纹理等特征也没有进行描述，导致图像检索匹配的效果并不理想。对于颜色分布集中的图像，匹配检索的效果较好，但对于颜色分布分散、形状纹理复杂的图像，在图像库信息众多的情况下，效果较差。

6.改进思路

针对上述问题，我们有两种改进思路：更换特征值、更换相关系数。

1.更换特征值

之前提到，基于颜色直方图的图像检索因忽略了图像物体形状、纹理等特征，无法较全面地反映图像的真实属性，因此，在特征值的选取上，我们可以尝试不同的算法将颜色直方图换为其他效果更好的特征值。

一种可以尝试的算法是“感知哈希算法”，它的作用是对每张图片生成一个"指纹"（fingerprint）字符串，然后比较不同图片的指纹。结果越接近，就说明图片越相似。其基本实现步骤为：缩小尺寸、简化色彩、计算平均值、比较像素的灰度、计算哈希值。得到指纹后，通过比较数据位的差异便可判断图片的相似性。

这种算法的优点是简单快速，不受图片大小缩放的影响，缺点是图片的内容不能变更。如果在图片上加几个文字，它就认不出来了。所以，它的最佳用途是根据缩略图，找出原图。在实际应用中，往往使用原理类似但更强大的pHash算法和SIFT算法，它们能够识别图片的变形。本实验将来也会采用这种方法进行优化。

2.更换相关系数

在本实验中，我们采用了皮尔逊相关系数来度量图像之间的相关性。实际

上，度量相关性的方法有很多，我们可以更换使用其他相关系数来进行对比实验，选择最适合图像相似性度量的相关系数。

统计学中有三大相关系数，包括pearson相关系数、Kendall Tau相关系数和spearman秩相关。本实验使用了pearson相关系数，为了更精确的结果，我们将来可以尝试另外两个相关系数进行实验，从而得到最适于比较图像相关性的相关系数。

7.参考文档

[1] 百度百科，皮尔逊相关系数
[2] 百度百科，查全率与查准率
[3] 机器学习进阶-直方图与傅里叶变换-图像直方图 1.cv2.calc(生成图像的像素频数分布(直方图))
[4] 相似图片搜索的原理
[5] python基于scipy模块实现统计学中三大相关系数的计算

附：本次实验代码已上传至https://github.com/xiaopeng-whu/Image_match