用Python和OpenCV创建一个图片搜索引擎的完整指南

2015/01/15 · 实践项目, 系列教程 · 21 评论 · OpenCV, Python, 图片搜索引擎

大家都知道，**通过文本或标签来搜索图片的体验非常糟糕。**

无论你是将个人照片贴标签并分类，或是在公司的网站上搜索一堆照片，还是在为下一篇博客寻找合适的图片。在用文本和关键字来描述图片是非常痛苦的事。

我就遇到了这样的痛苦的事情，上周二我打开了一个很老的家庭相册，其中的照片是9年前扫描成电子档的。

我想找到我家在夏威夷海滩拍的照片。我用iPhoto打开相册，慢慢的浏览。这个过程非常辛苦，每个JPEG图像的元信息中的日期都是错的。我已经不记得文件夹中的图片是如何排列的，我绝望的搜索海滩的照片，但还是找不到。

也许是运气，我跌跌撞撞的找到了其中一幅海滩上的照片。多美的一幅照片啊。蓝天中飘着棉花糖般的白云。晶莹透彻的海水，像丝绸一样掠过在金色的沙滩上。我几乎可以感觉微风轻抚着面庞，呼吸着海边湿润的空气。

找到这幅照片后，我停止了手动搜索，打开一个代码编辑器。

虽然iPhoto这样的应用能让你将相片分组，甚至可以检测人脸，但我们可以做的更多。

注意，我并不是介绍如何手动给图片添加标签。我是在介绍更强大的东西。比如通过一幅图片来搜索一组相似的图片。

这是不是很酷？只需鼠标点击一次就可以可视化搜索图片。

这就是我的工作内容。我用半个小时写好代码，完成了一个针对家庭假期相册的图片搜索引擎。

然后用上面找到的那张海滩图片作为搜索源。几秒后我就找到了相册中其他的海滩图片，其中没有任何为某张图片添加标签的动作。

感兴趣吗，我们继续。

在本文的其他部分，我将介绍如何自己创建一个图像搜索引擎。

想要文本中的代码?

直接跳到文本中的最后的“下载”一节。

什么是图像搜索引擎?

读者也许会问，什么才是一个真正的图像搜素引擎?

我的意思是，我们都熟悉基于文本的搜索引擎，如Google、Bing、Baidu等。用户只需输入几个与内容相关的关键字，接着就会获得搜索结果。但对于图像搜索引擎，其工作方式就有点区别。搜索时使用的不是文字，而是图片。

听起来很困难，我的意思是，如何量化图像的内容，让其可搜索呢?

本文将逐步回答这个问题。首先，先了解一下图像搜索引擎的内容。

一般来说，有三种类型的图像搜索引擎：基于元数据、基于例子、混合模式。

基于元数据

图1：基于元数据的图像搜索引擎的例子。注意其中关键字和标签是手动关联到图像上的。

通过元数据搜索与标准的关键字搜索引擎没有本质的不同。这种方式很少检测图像本身的内容。而是用相关的文本信息：如手动注释或添加标签；以及自动上下文提示（如网页中该图片附近的文字信息）。

当用户在基于元数据的系统上进行搜索时，与传统的文本搜索引擎其实差点不多的。得到的是含有类似标签或注释的图片。

再次说明，使用基于元数据系统的工具进行搜索，基本上是不查找图像本身的。

用基于元数据进行搜索的应用中，一个比较好的例子就是Flickr。将图像上传到Flickr后，输入一些文本作为标签描述这幅图像。接着Flickr会使用这些关键字进行搜索，查找并推荐其他相关的图像。

基于例子搜索

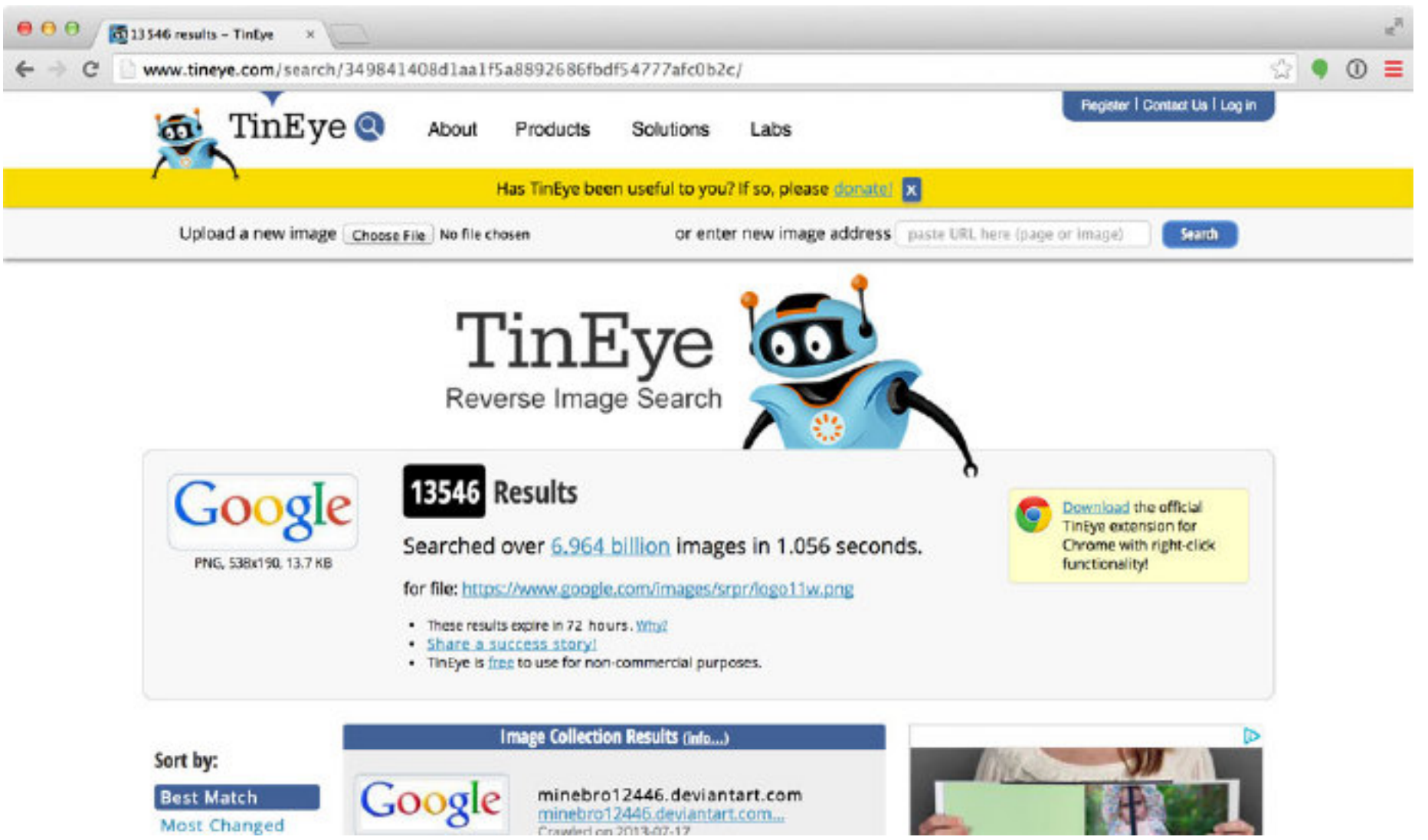


图2：TinEye就是一个基于例子的图像搜索引擎。该搜索引擎会使用图像本身的内容进行搜索，而不是使用文本搜索。

另一方面，基于例子搜索仅仅依赖于图像的内容，不需要提供关键字。引擎会分析、量化并存储图像，然后返回其他相关的图像。

图像搜索引擎量化图像内容的过程称为**基于内容的图像信息获取（Content-Based Image Retrieval，CBIR）**系统。术语CBIR通常用在学术文献中，但在实际上，这是“图像搜索引擎”的另一种表述，并特意指明该搜索引擎是严格基于图像的内容的，没有任何关于图像的文本的信息。

基于例子系统的一个比较好的例子就是TinEye。向TinEyet提交待查找图像时，TinEye实际上是一个逆向图像搜索引擎，TinEye返回该图像最相近的匹配，以及该图像位于的原始网页地址。

看下本节刚开始的示例图像。我上传了一个Google logo图像。TinEye检测了图像的内容，在搜索了超过60亿幅图片后，返回了1.3万个含有Google logo图片的网页。

所以仔细想一下：你需要为TinEye中的60亿幅图像收到添加标签吗？**当然不需要，**为这么多图片手动添加标签需要庞大的人力物力。

取而代之，使用某些算法从图像本身中提取“特征”（如用一组数字来量化并抽象表示图像）。接着，当用户提交了需要查找的图像，从这幅图像中提取特征，将其与数据库中的特征进行比较，尝试返回相似的图像。

同样，依然需要强调，基于例子的搜索系统非常依赖图像的内容。这种类型的系统很难构建并扩展，但可以用算法全自动的搜索，无需人工干预。

混合方式

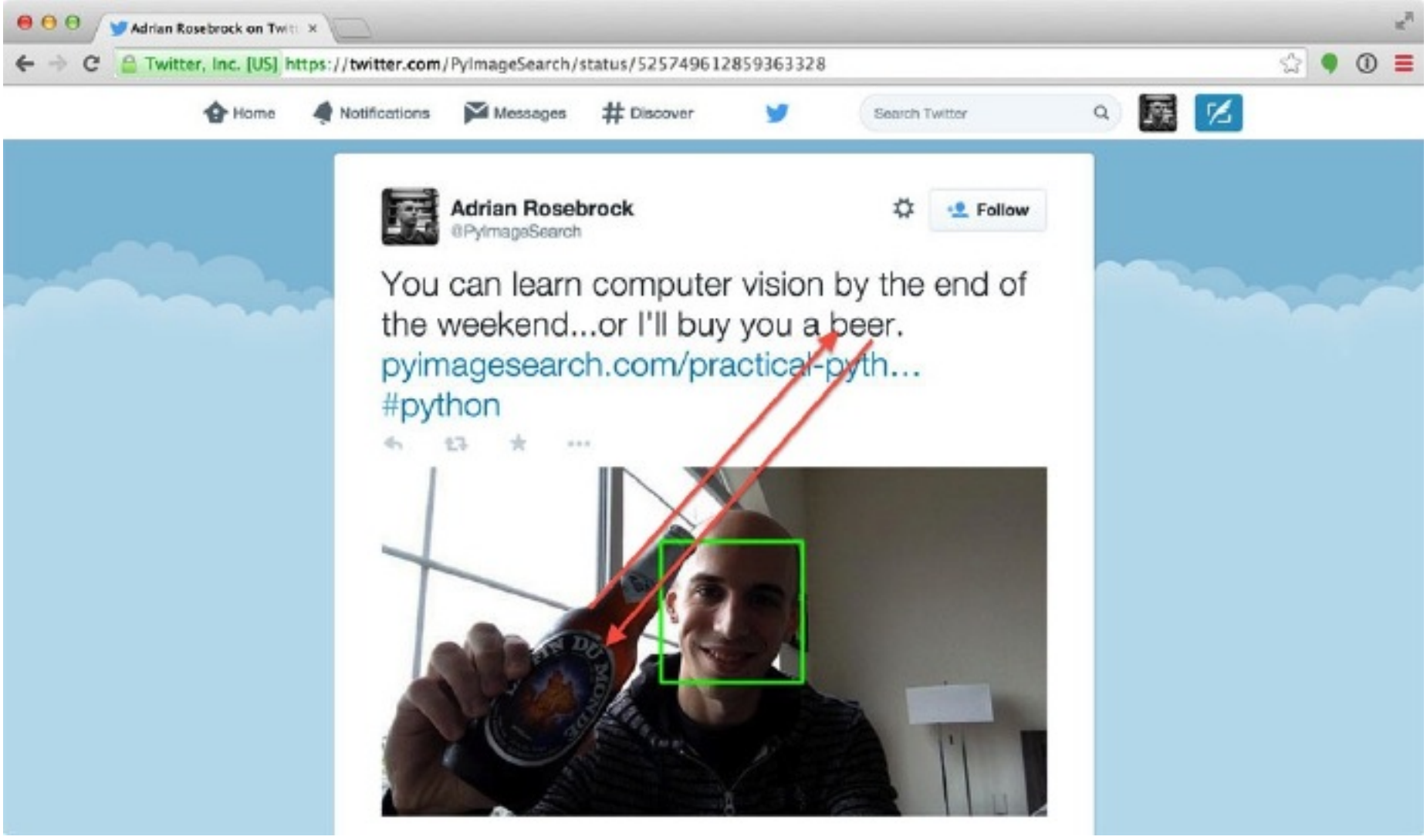


图3：混合式图像搜索引擎可以同时基于图像和文本描述搜索。

当然，除了前面介绍的两种方式，还有一种介于两者之间的方式，如Twitter使用的。

在Twitter上，可以与推文一起上传图像。这样就可以即使用提取图像本身的特征，也可以使用推文中的文本，从而诞生一种混合方式。基于这种方式可以可以构建一个即使用上下文关系，又使用基于例子搜索的策略的图像搜索引擎。

提示：有兴趣阅读更多关于不同类型的图像搜索引擎的资料？我有一篇完整的博客介绍比较这些搜索引擎的，链接在此。

在进一步描述和构建图像搜索引擎之前，让我们先来了解一些重要的术语。

一些重要的术语

在深入了解之前，先花点时间了解一些重要的术语。

在构建图像搜索引擎时，首先要对**数据集编列索引**。索引化数据集是量化数据集的过程，即通过**图像描述符**（image descriptor，也称描述子）提取每幅图像的**特征**。

图像描述符就是用来描述图像的算法。

例如：

- R、G、B三色通道的均值和标准差。
- 图像特征形状的统计矩.
- 形状和纹理的梯度和朝向。

这里最重要的是图像描述符确定了图像是如何量化的。

另一方面，**特征**是图像描述符的输出。当将一幅图像放入图像描述符中时，就会获得这幅图像的特征。

以基本的术语来说。**特征**（或**特征向量**）仅仅是一个用来抽象表示或量化的图像的数字列表。

来看下面这幅示例图像：

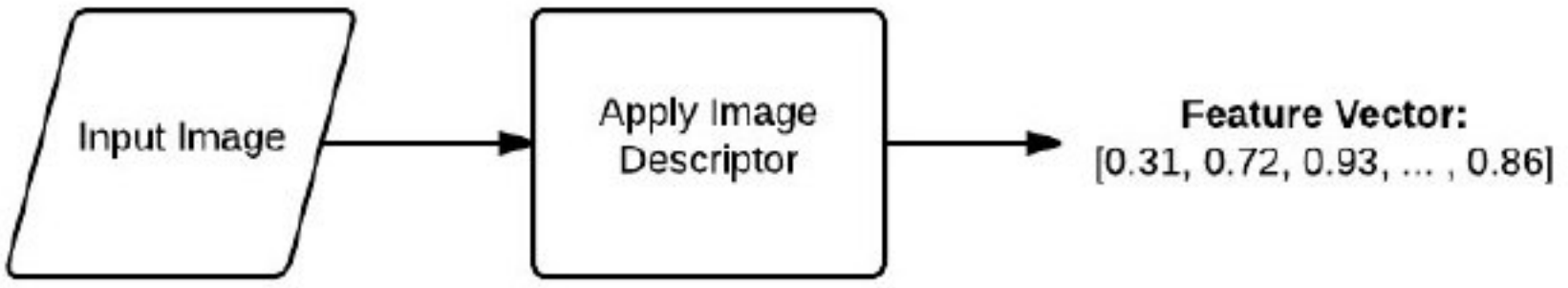


图4：图像描述符的管道。描述符中有一幅输入图像，使用图像描述符会返回一个特征向量（一个数字列表），

这里对一幅输入图像使用图像描述符，输出是一组用来量化图像的数字。

通过距离量测或其他相似度比较函数，特征向量可以用来表示比较的相似度。**距离量测和相似度函数**采用两个特征向量作为输入，返回一个数值来描述着两个特征向量的相似度。

下图以可视化的方式比较了两幅图的比较过程：

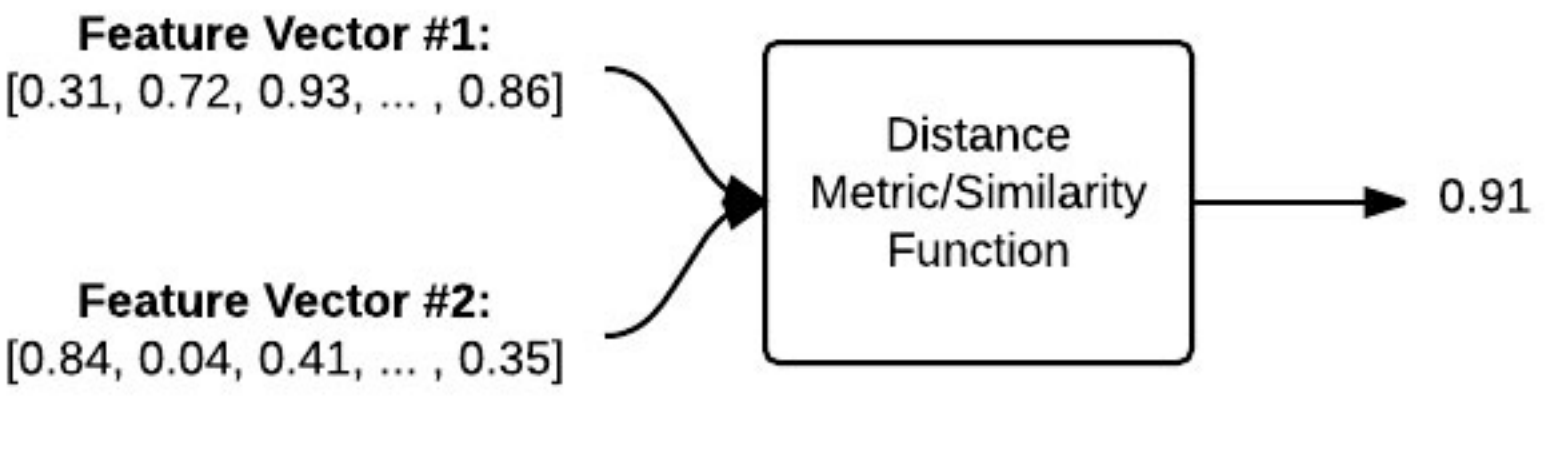


图5：为了比较两幅图，必须将对应的特征向量输入进距离量测/相似度比较函数。输出结果是一个数值，量化地描述两幅图下的相似度。

给定两个特征向量，使用距离函数来确定这两个特征向量的相似度。距离函数的输出是一个浮点数，用来描述两幅图像的相似度。

CBIR系统的4个步骤

无论构建的是什么样的CBIR系统，最终都可以分解成4个不同的步骤。

1. **定义图像描述符**：在这一阶段，需要决定描述图像的哪一方面。是关注图像的颜色，还是图像中的物体形状，或是图像中的纹理?
2. **索引化数据集**：现在有了图像描述符，接着就是将这个图像描述符应用得到数据集中的每幅图像，提取这些图像的特征，将其存储起来（如CSV文件、RDBMS、Redis数据库中，等），这样后续步骤就能使用以便比较。
3. **定义相似矩阵**：很好，现在有了许多特征向量。但如何比较这些特征向量呢？流行的方式是比较欧几里德距离、余弦距离、或卡方距离。但实际中取决于两点：1、数据集；2、提取的特征类型。
4. **搜索**：最后一步是进行实际的搜索。用户会向系统提交一幅需要搜索的图片（例如从上传窗口或通过移动App提交），而你的任务是：1、提取这幅图像的特征；2、使用相似度函数将这幅图像的特征与已经索引化的特征进行比较。这样，只需根据相似度函数的结果，返回相关的图像就可以了。

再次强调，这是所有CBIR系统中最基本的4步。如果使用的特征表示不同，则步骤数会增加，也会为每个步骤增加一定数量的子步骤。就目前而言，让我们关注并使用这4步。

下面通过图像来具体了解这4个大步骤。下图表述的是步骤1和2：

图6：处理并提取数据集中的每幅图像的流程图。

首先提取数据集中每幅图像的特征，将这些特征存入一个数据库。

接着可以执行搜索（步骤3和4）：

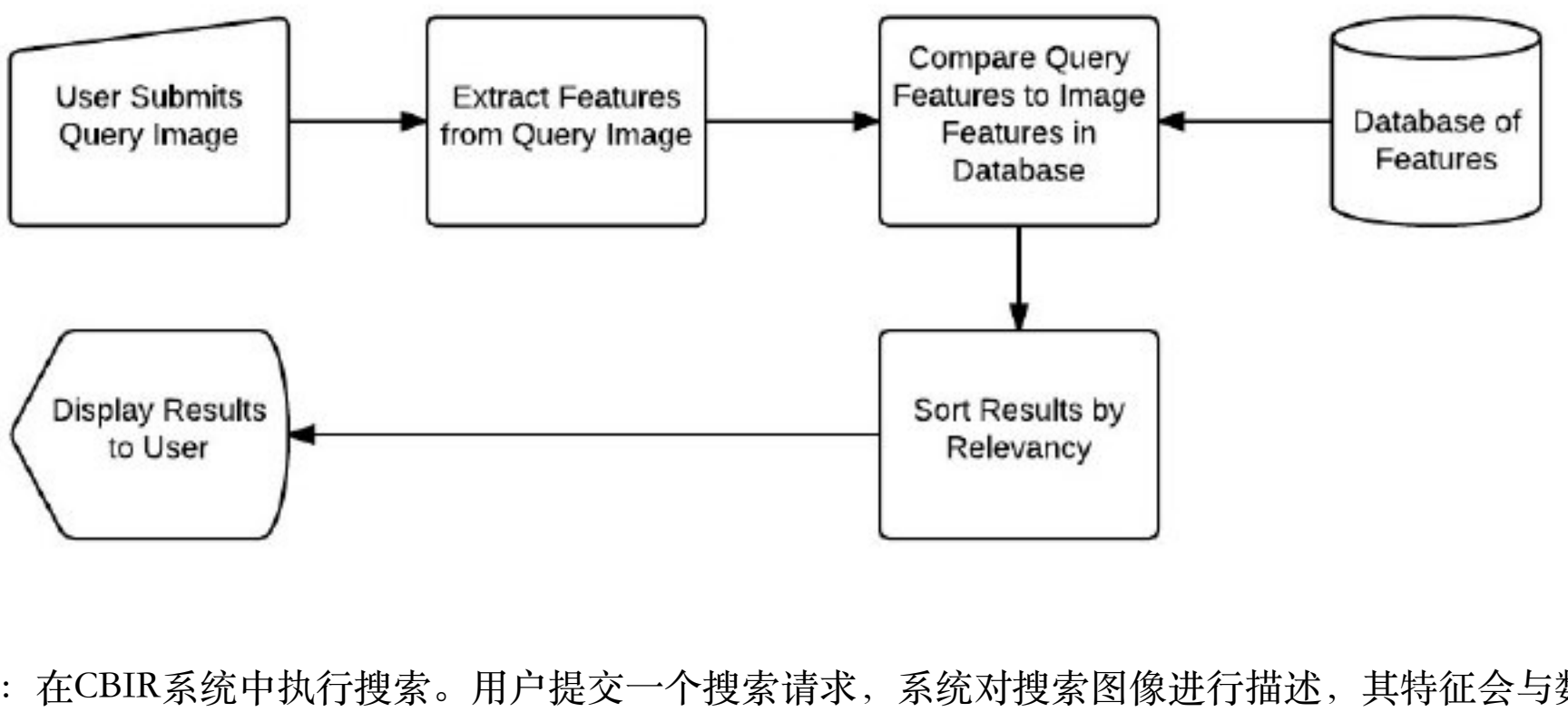


图7：在CBIR系统中执行搜索。用户提交一个搜索请求，系统对搜索图像进行描述，其特征会与数据集中已有的特征进行比较，并对结果根据相关度进行排序，返回给用户。

首先，用户必须像搜索引擎提交一幅需要查找的图像。接着对这幅图像提取特征信息。将这些特征信息与数据集中已有的图像的特征信息进行比较。最后，对结果根据相关度进行排序并返回给用户。

数据集——假期相册

这里将INRIA假期数据集作为图像搜索的数据集。

这个数据集含有全世界许多地方的假期旅行，包括埃及金字塔、潜水、山区的森林、餐桌上的瓶子和盘子、游艇、海面上的日落。

下面是数据集中的一些图片：

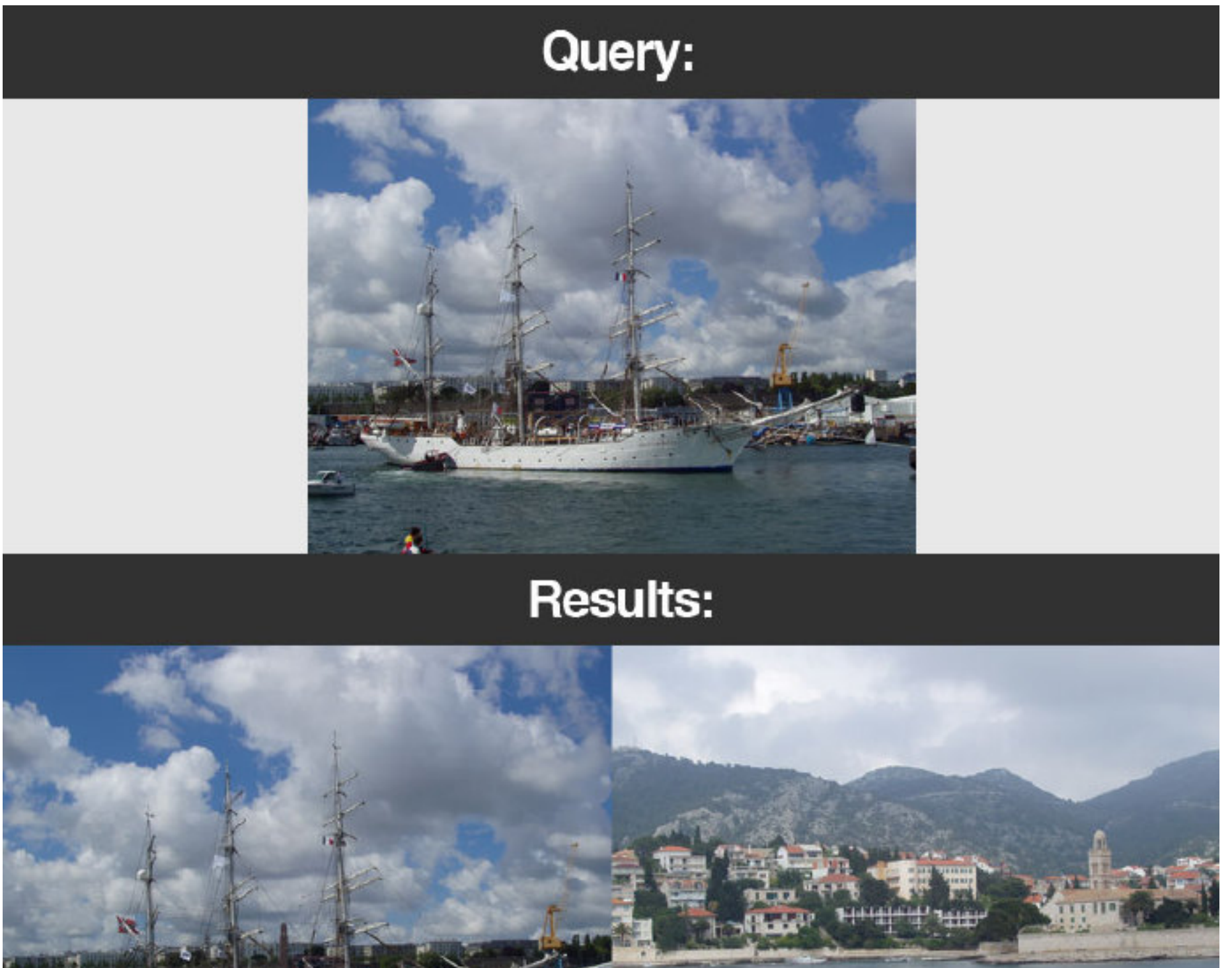
图8：数据集中的示例图像。我们将使用这些图像构建自己的图像搜索引擎。

在本例中，对于我们希望从旅行相册中找到某种景色的相片来说，用这幅数据集作为示例来说非常好。

目标

我们的目标是构建一个个人图像搜索引擎。将假期照片作为数据集，我们希望将这个数据集变成可搜索的，即一个“基于例子”的图像搜索引擎。例如，如果我提交了一幅在河中航行的帆船的照片，图像搜索引擎应该能找到并返回相册中码头和船坞拍摄的照片。

看下面的图，其中有我提交的照片，即一幅在水里的船。得到了假期照片集合中相关的图像。











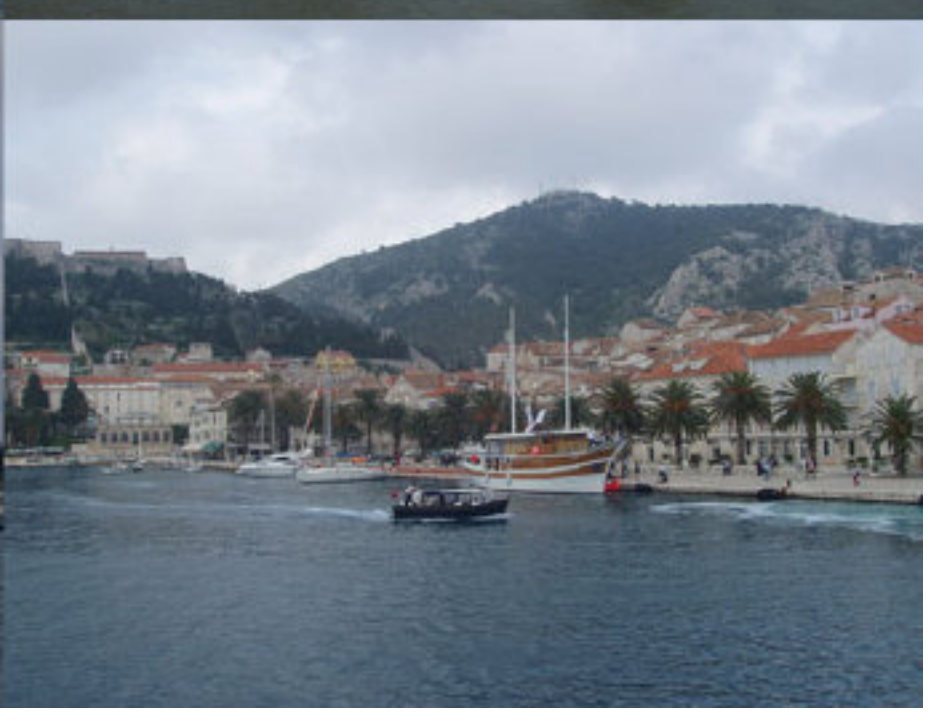



图9：图像搜索引擎的例子。提交了一幅含有海中船只的图像。返回相关的图像，这些图像都是在海中的船。

为了构建这个系统，将使用一个简单且有效的图像描述符：**颜色直方图**。

通过将颜色直方图作为我们的图像描述符，可以根据图像的色彩分布提取特征。由于这一点，我们可以对我们的图像搜索引擎做个重要的假设：

假设：如果图像含有相似的色彩分布，那么这两幅图像就认为是相似的。即使图像的内容差别非常大，依然会根据色彩分布而被认为是相近的。

这个假设非常重要，在使用颜色直方图作为图像描述符时，这是个公平且合理的假设。

第一步1：定义图像描述符

这里不使用标准的颜色直方图，而是对其进行一些修改，使其更加健壮和强大。

这个图像描述符是HSV颜色空间的3D颜色直方图（色相、饱和度、明度）。一般来说，图像由RGB构成的元组表示。通常将RGB色彩空间想象成一个立方体，如下图所示。

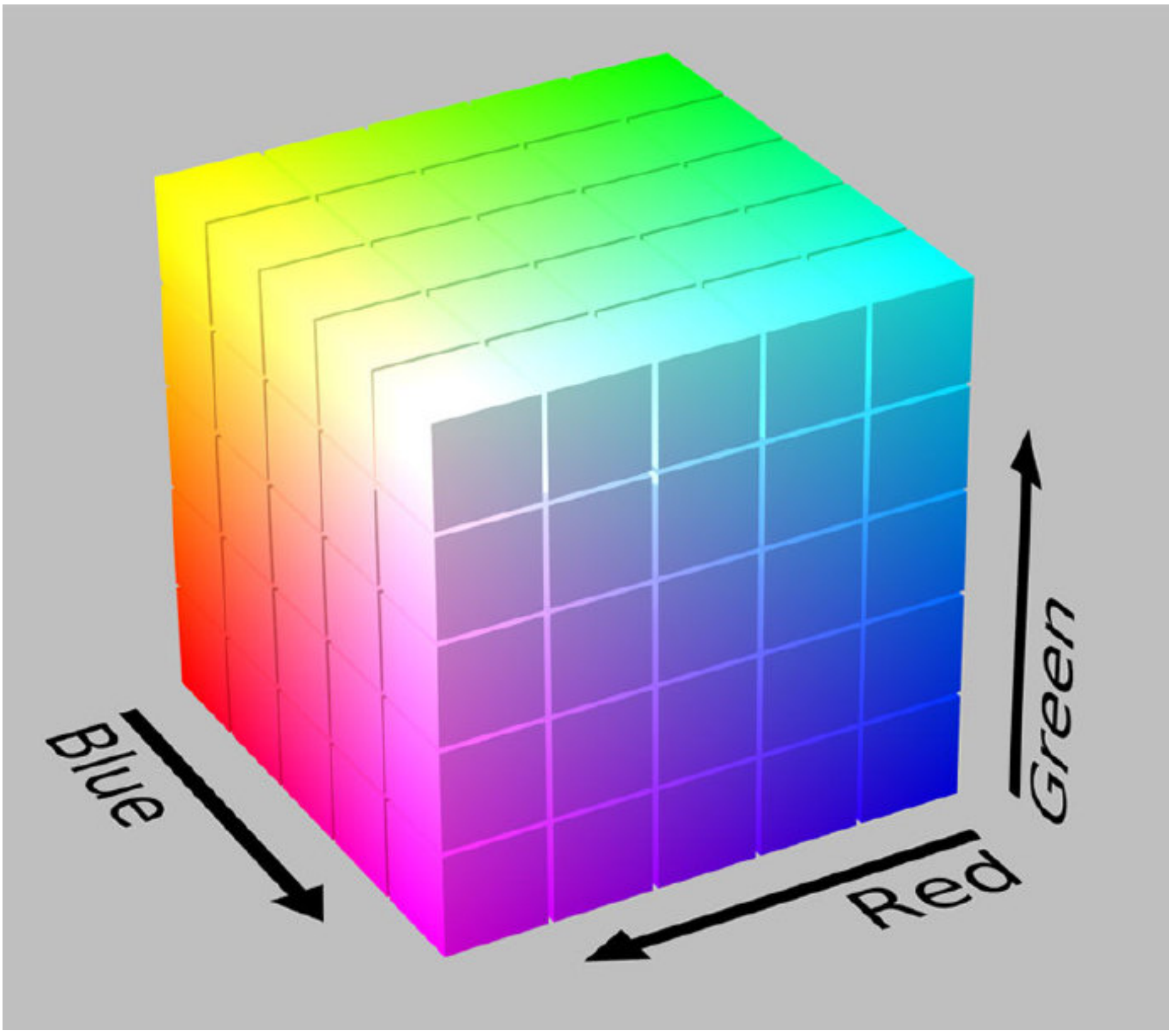


图10：RGB立方体的例子。

然而，虽然RGB值很容易理解，但RGB色彩空间无法模拟人眼接受到的色彩。取而代之，我们使用HSV色彩空间将像素点的映射到圆bin体上。

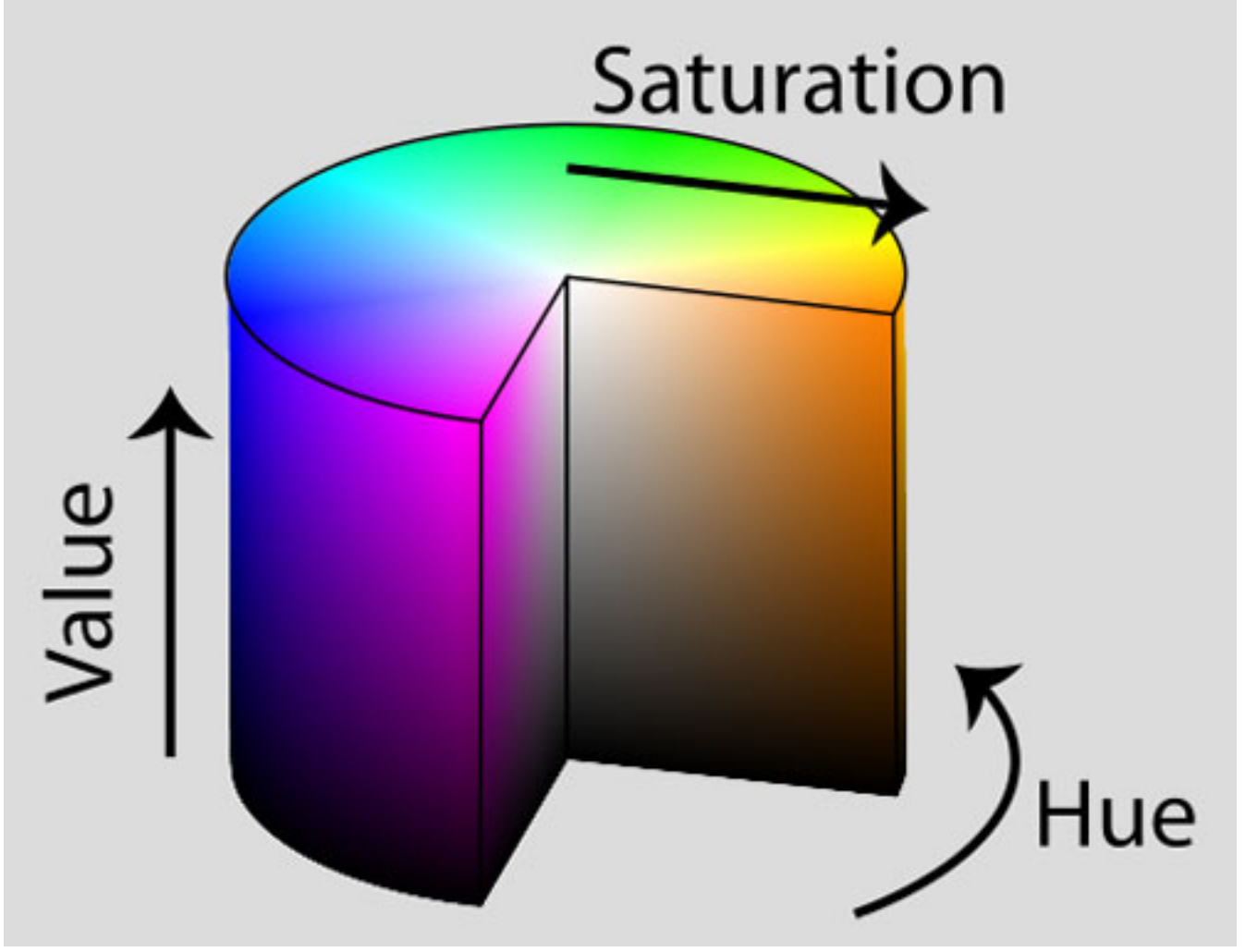


图11：HSV圆bin体的例子。

还有其他颜色空间能够更好的模拟人眼接收的颜色，如CIE L*a*b*和CIE XYZ颜色空间，但作为第一个图像搜索引擎的实现，先简化使用的色彩模型。

现在选定了颜色空间，接着需要定义直方图中bin的数量。直方图用来粗略的表示图像中各强度像素的密度。本质上，直方图会估计底层函数的概率密度。在本例中，P是图像I中像素色彩C出现的概率。

主要注意的是，为直方图选取bin的数目需要不断的权衡。如果选择的bin数目过少，那么直方图含有的数据量就不够，无法区分某些不同颜色分布的图像。反之，如果直方图选取的bin的数目过多，那么其中的组件就过多，导致内容很相近的图片也会判断成不相似。

下面是直方图bin过少的例子。

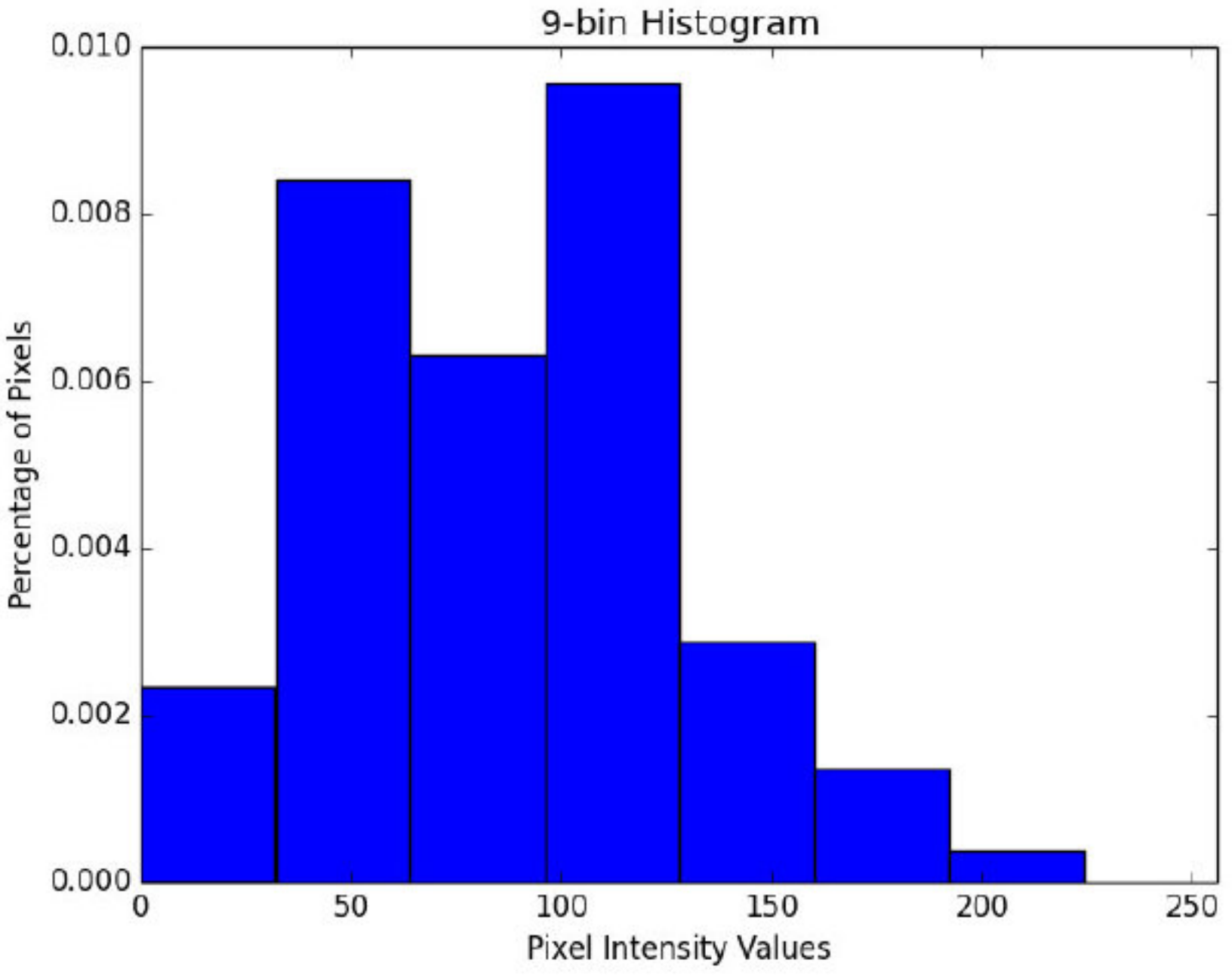


图12：9个bin直方图的例子。注意其中bin的数量很少，只有少数给定的像素值位列其中。

注意其中只有少数几个bin及相应的像素值。

下面是直方图bin过多的例子。

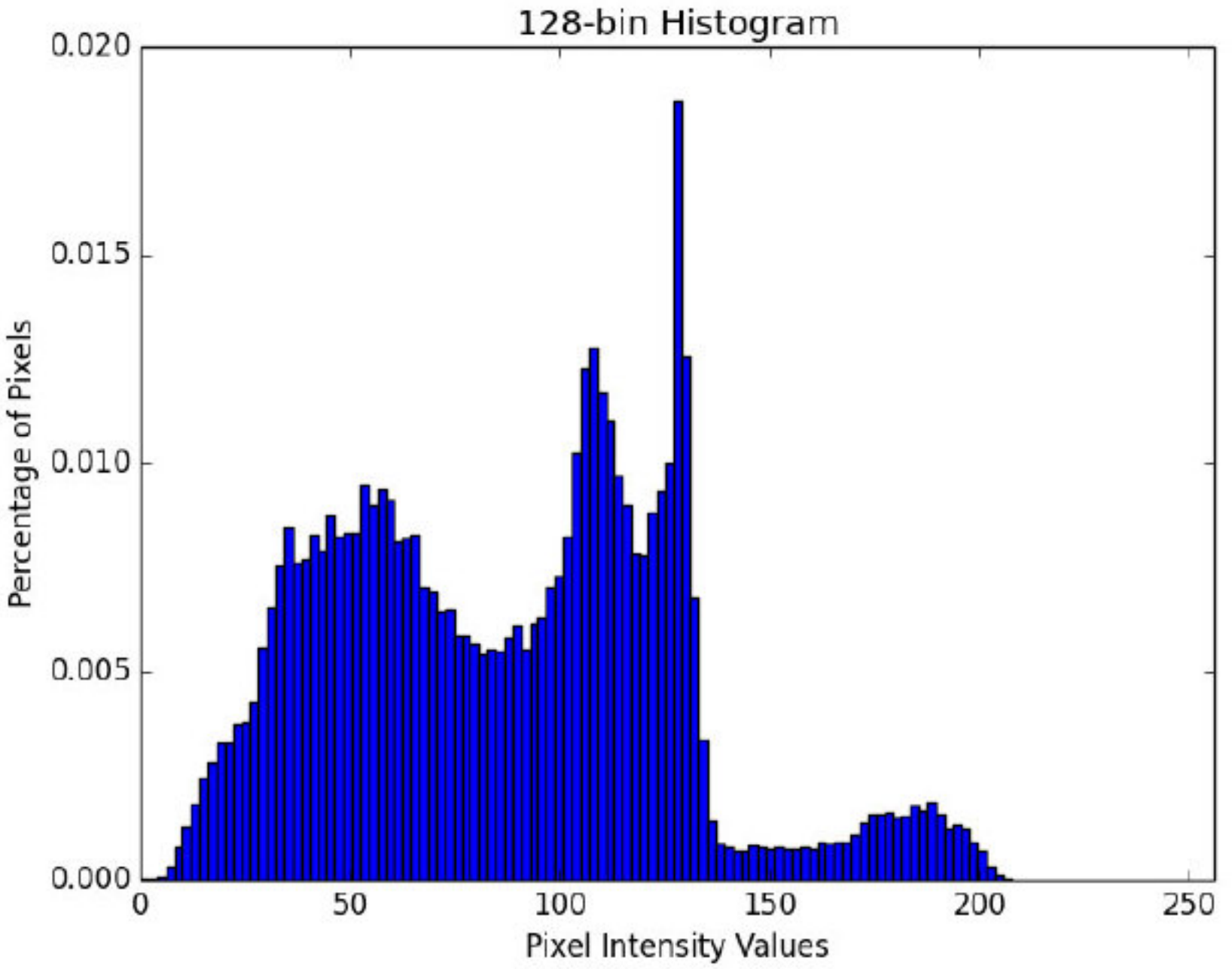


图13：128 bin直方图的例子。注意其中含有许多的柱和相应的像素值。

在上面的的例子中使用了许多bin，bin的数目过多，由于需要直方图中每个“山峰”和“山谷”都需要匹配才能认为图像是“相似的”，所以就失去了“概括”图像的能力。

就我个人而言，我喜欢用迭代、实验性的方式来调整bin的数目。迭代方法一般基于数据集的大小调整。数据集越小，使用的bin的数目就越少。如果数据集非常大，则会使用更多的bin，这样可以让直方图更大，更能区分图像。

一般来说，读者需要为颜色直方图描述符实验bin的个数，具体取决于数据集的大小和数据集中图像之间色彩分布的差异。

对于我们的假期照片图像搜索引擎，将在HSV色彩空间中使用3D颜色直方图，8个bin用于色相通道、12个bin用于饱和度通道、3个bin用于明度通道，总共的特征向量有$8 \times 12 \times 3=288$。

这意味着数据集中的每幅图像，无论其像素数目是36×36，还是2000×1800。最终都会用288个浮点数构成的列表抽象并量化表示。

我认为解释3D直方图最好的方式是用连接词AND。一个3D HSV颜色描述符将查找指定图像中1号bin有多少像素含有**色相**值，AND有多少像素有**饱和度**值，AND有多少像素有**明度**值。计算出符合条件的像素值。虽然需要对每个bin重复这个操作，但可以非常高效的完成这个任务。

很酷，是吧！

理论讲解的够多了，下面来开始编码。

用你最喜欢的编辑器打开一个新文件，命名为colordescriptor.py。加入下面代码：

```
Python
```

```python
# import the necessary packages
import numpy as np
import cv2

class ColorDescriptor:
    def __init__(self, bins):
        # store the number of bins for the 3D histogram
        self.bins = bins

    def describe(self, image):
        # convert the image to the HSV color space and initialize
        # the features used to quantify the image
        image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        features = []

        # grab the dimensions and compute the center of the image
        (h, w) = image.shape[:2]
        (cX, cY) = (int(w * 0.5), int(h * 0.5))
```

首先导入所需的Python模块。用NumPy进行数值处理，用cv2使用OpenCV的Python绑定。

在第五行定义了ColorDescriptor类。该类用来封装所有用于提取图像中3D HSV颜色直方图的逻辑。

ColorDescriptor的__init__方法只有一个参数——bins，即颜色直方图中bin的数目。

在第10行定义describe方法，用于描述指定的图像。

在describe方法中，将图像从RGB颜色空间（或是BGR颜色空间，OpenCV以NumPy数组的形式反序表示RGB图像）转成HSV颜色空间。接着初始化用于量化图像的特征列表features。

17和18行获取图像的维度，并计算图像中心(x, y)的位置。

现在遇到难点。

这里不计算整个图像的3D HSV颜色直方图，而是计算图像中不同区域的3D HSV颜色直方图。

使用基于区域的直方图，而不是全局直方图的好处是：这样我们可以模拟各个区域的颜色分布。例如看下面的这幅图像：

图14：待搜索的图像。

在这幅图像中，很明显，蓝天在图像的上部，而沙滩在底部。使用全局搜索的话，就无法确定图像中“蓝色”区域和“棕色”沙子区域的位置。而是仅仅知道图像中有多少比例是蓝色，有多少比例是棕色。

为了消除这个问题，可以对图像中的不同区域计算颜色直方图：

图15：将图像分为5个不同区域的例子。

对于我们的图像描述符，将图像分为5个不停的区域：1、左上角；2、右上角；3、右下角；4、左下角；以及图像的中央。

使用这些区域，可以粗略模拟出不同的区域。能够表示出蓝天在左上角和右上角，沙滩在左下角和右下角。图像的中央是沙滩和蓝天的结合处。

下面的代码是创建基于区域的颜色描述符：

Python

```python
# import the necessary packages
import numpy as np
import cv2

class ColorDescriptor:
    def __init__(self, bins):
        # store the number of bins for the 3D histogram
        self.bins = bins

    def describe(self, image):
        # convert the image to the HSV color space and initialize
        # the features used to quantify the image
        image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        features = []

        # grab the dimensions and compute the center of the image
        (h, w) = image.shape[:2]
        (cX, cY) = (int(w * 0.5), int(h * 0.5))

        # divide the image into four rectangles/segments (top-left,
        # top-right, bottom-right, bottom-left)
        segments = [(0, cX, 0, cY), (cX, w, 0, cY), (cX, w, cY, h),
            (0, cX, cY, h)]

        # construct an elliptical mask representing the center of the
        # image
        (axesX, axesY) = (int(w * 0.75) / 2, int(h * 0.75) / 2)
        ellipMask = np.zeros(image.shape[:2], dtype = "uint8")
        cv2.ellipse(ellipMask, (cX, cY), (axesX, axesY), 0, 0, 360, 255, -1)

        # loop over the segments
        for (startX, endX, startY, endY) in segments:
            # construct a mask for each corner of the image, subtracting
            # the elliptical center from it
            cornerMask = np.zeros(image.shape[:2], dtype = "uint8")
            cv2.rectangle(cornerMask, (startX, startY), (endX, endY), 255, -1)
```

```
37            cornerMask = cv2.subtract(cornerMask, ellipMask)
38
39            # extract a color histogram from the image, then update the
40            # feature vector
41            hist = self.histogram(image, cornerMask)
42            features.extend(hist)
43
44        # extract a color histogram from the elliptical region and
45        # update the feature vector
46        hist = self.histogram(image, ellipMask)
47        features.extend(hist)
48
49        # return the feature vector
50        return features
```

22和23行用于分别定义左上、右上、右下、和左下区域。

这里，我们需要构建一个椭圆用来表示图像的中央区域。在代码的27行，定义一个长短轴分别为图像长宽75%的椭圆。

接着初始化一个空白图像（将图像填充0，表示黑色的背景），该图像与需要描述的图像大小相同，见28行。

最后，在29行使用cv2.ellipse函数绘制实际的椭圆。该函数需要8个不同的参数：

1. 需要绘制椭圆的图像。这里使用了下面会介绍的“掩模”的概念。
2. 两个元素的元组，用来表示图像的中心坐标。
3. 两个元组的元组，表示椭圆的两个轴。在这里，椭圆的长短轴长度为图像长宽的75%。
4. 椭圆的旋转角度。在本例中，椭圆无需旋转，所以值为0。
5. 椭圆的起始角。
6. 椭圆的终止角。看上一个参数，这意味着绘制的是完整的椭圆。
7. 椭圆的颜色，255表示的绘制的是白色椭圆。
8. 椭圆边框的大小。传递正数比会以相应的像素数目绘制椭圆边框。负数表示椭圆是填充模式。

在35行为每个角的掩模分配内存，在36行为图像的每个角绘制白色矩形，接着在37行将矩形减去中间的椭圆。

如果将这个过程用图像动态表示，看上去应该是这样的：

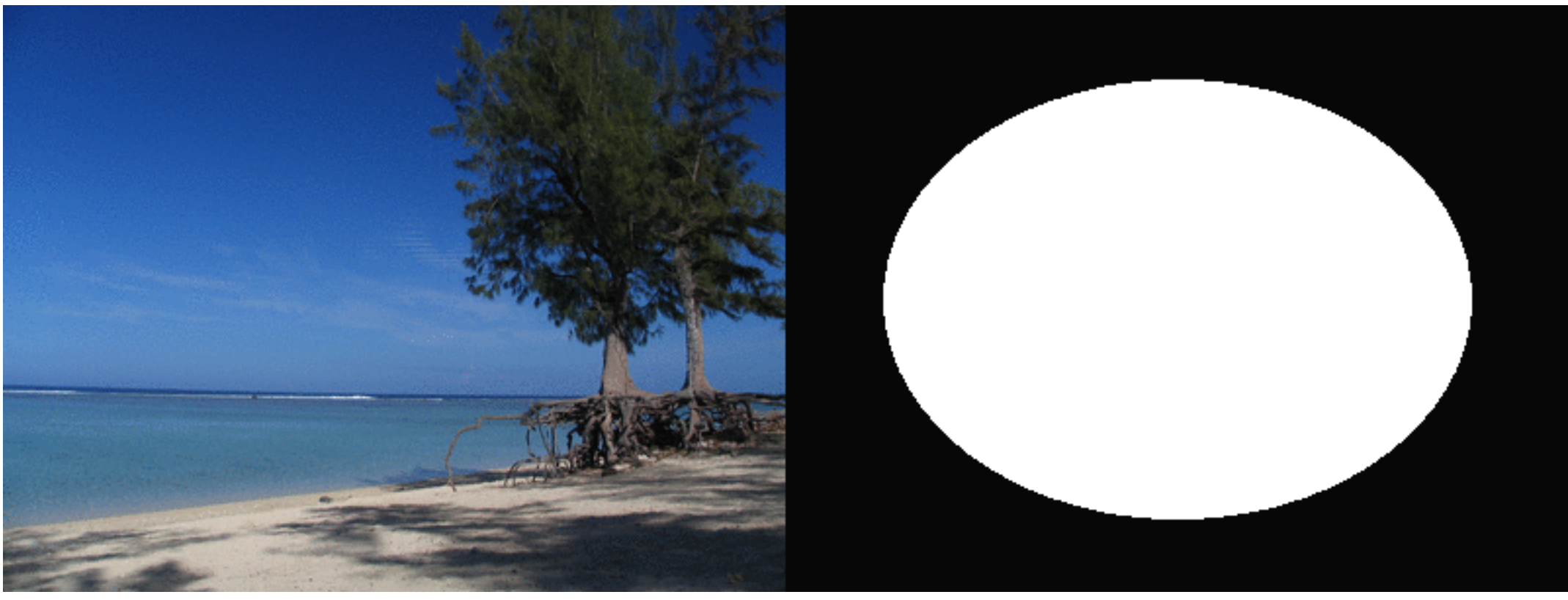


图16：为图像中每个需要提取特征的区域构建掩模。

如这个动画所示，我们独立检测每块区域，在迭代中移除每个矩形与图像中间的椭圆重叠的部分。

读者也许会奇怪，“我们不是要提取图像的颜色直方图吗？为什么要做这些掩模的事情？”

问得好！

原因是因为我们需要告诉OpenCV直方图函数我们要提取的颜色直方图的区域。

记住，我们的目标是分开描述图像的每个区域。表述不同区域最高效的方法是使用掩模。对于图像中某个点(x, y)，只有掩模中该点位白色(255)时，该像素点才会用于计算直方图。如果该像素点对应的位置在掩模中是黑色(0)，将会被忽略。

通过下图可以更加深刻的了解这个概念。

图17：对图像使用掩模。注意左图中只有右图掩模中对应的区域为白色才会显示。

可以看到，只有掩模的区域才会用于直方图的计算中。

很合理，是吧。

所以现在在41行针对每个区域都调用直方图方法。第一个参数是需要提取特征的图像，第二个参数是掩模区域，这样来提取颜色直方图。

histogram方法会返回当前区域的颜色直方图表示，我们将其添加到特征列表中。

46和47行提取图像中间（椭圆）区域的颜色直方图并更新features列表。

最后，在50行像调用函数返回特征向量。

现在来快速浏览下实际的histogram方法：

```python
def histogram(self, image, mask):
    # extract a 3D color histogram from the masked region of the
    # image, using the supplied number of bins per channel; then
    # normalize the histogram
    hist = cv2.calcHist([image], [0, 1, 2], mask, self.bins,
        [0, 180, 0, 256, 0, 256])
    hist = cv2.normalize(hist).flatten()

    # return the histogram
    return hist.
```

这里的histogram方法需要两个参数，第一个是需要描述的图像，第二个是mask，描述需要描述的图像区域。

在5和6行，通过调用cv2.calcHist计算图像掩模区域的直方图，使用构造器中的bin数目作为参数。

在7行对直方图归一化。这意味着如果我们计算两幅相同的图像，其中一幅比另一幅大50%，直方图会是相同的。对直方图进行归一化非常重要，这样每个直方图表示的就是图像中每个bin的所占的比例，而不是每个bin的个数。同样，归一化能保证不同尺寸但内容近似的图像也会在比较函数中认为是相似的。

最后，在10行向调用函数返回归一化后的3D HSV颜色直方图。

步骤2：从数据集提取特征

现在有了定义好的图像描述符，进入第二步，对数据集中的每幅图像提取特征（如颜色直方图）。提取特征并将其持久保存起来的过程一般称为“索引化”。

继续来看对假期照片数据集进行索引化的代码。创建一个新文件，命名为index.py，添加索引化所需的代码：

```python
# import the necessary packages
from pyimagesearch.colordescriptor import ColorDescriptor
import argparse
import glob
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required = True,
    help = "Path to the directory that contains the images to be indexed")
ap.add_argument("-i", "--index", required = True,
    help = "Path to where the computed index will be stored")
args = vars(ap.parse_args())

# initialize the color descriptor
cd = ColorDescriptor((8, 12, 3))
```

首先导入所需的模块。注意第一步中的ColorDescriptor类，这里将其放入pyimagesearch模块，以便更好的组织代码。

还需要argparse模块来处理命令行参数、glob来获取图像的文件路径，以及cv2来使用OpenCV的接口。

7-12行用来处理命令行指令。这里需要两个指令，–dataset，表示假期相册的路径。–index，表示输出的CSV文件含有图像文件名和对应的特征。

最后，在16行初始化ColorDescriptor，8 bin拥有色相、12 bin用于饱和度、3 bin用于明度。

现在所有内容都初始化了，可以从数据集提取特征了：

```python
# open the output index file for writing
output = open(args["index"], "w")

# use glob to grab the image paths and loop over them
for imagePath in glob.glob(args["dataset"] + "/*.png"):
    # extract the image ID (i.e. the unique filename) from the image
    # path and load the image itself
    imageID = imagePath[imagePath.rfind("/") + 1:]
    image = cv2.imread(imagePath)

    # describe the image
    features = cd.describe(image)

    # write the features to file
    features = [str(f) for f in features]
    output.write("%s,%sn" % (imageID, ",".join(features)))

# close the index file
output.close()
```

在2行代开输出文件，在5行遍历数据集中的所有图像。

对于每幅图像，可以提取一个imageID，即图像的文件名。对于这个作为示例的搜索引擎，我们假定每个文件名都是唯一的，也可以针对每幅图像生产一个UUID。在9行将从磁盘上读取图像。

现在图像载入内存了，在12行对图像使用图像描述符并提取特征。ColorDescriptor的describe方法返回由浮点数构成的列表，用来量化并表示图像。

这个数字列表，或者说特征向量，含有第一步中图像的5个区域的描述。每个区域由一个直方图表示，含有8 × 12 × 3 = 288项。5个区域总共有5 × 288 = 1440维度。。。因此每个图像使用1440个数字量化并表示。

15和16行简单的将图像的文件名和管理的特征向量写入文件。

为了索引化我们的相册数据集，打开一个命令行输入下面的命令：

```Python
$ python index.py --dataset dataset --index index.csv
```

这个脚本运行的很快，完成后将会获得一个名为index.csv的新文件。

使用你最喜欢的文本编辑器打开并查看该文件。

可以看到在.csv文件的每一行，第一项是文件名，第二项是一个数字列表。这个数字列表就是用来表示并量化图像的特征向量。

对index文件运行wc命令，可以看到已经成功对数据集中805幅图像索引化了：

```Python
$ wc -l index.csv
    805 index.csv
```

第3步：搜索

现在已经从数据集提取了特征了，接下来需要一个方法来比较这些特征，获取相似度。这就是第三步的内容，创建一个类来定义两幅图像的相似矩阵。

创建一个新文件，命名为searcher.py，让我们在这里做点神奇的事情：

```Python
# import the necessary packages
import numpy as np
import csv

class Searcher:
    def __init__(self, indexPath):
        # store our index path
        self.indexPath = indexPath

    def search(self, queryFeatures, limit = 10):
        # initialize our dictionary of results
        results = {}
```

首先先导入NumPy用于数值计算，csv用于方便的处理index.csv文件。

在第5行定义Searcher类。Searcher类的构造器只需一个参数，indexPath，用于表示index.csv文件在磁盘上的路径。

要实际执行搜索，需要在第10行调用search方法。该方法需要两个参数，queryFeatures是提取自待搜索图像（如向CBIR系统提交并请求返回相似图像的图像），和返回图像的数目的最大值。

最后，在12行初始化results字典。在这里，字典有很用的用途，每个图像有唯一的imageID，可以作为字典的键，而相似度作为字典的值。

好了，现在将注意力放在这里。这里是发生神奇的地方：

```python
        # open the index file for reading
        with open(self.indexPath) as f:
            # initialize the CSV reader
            reader = csv.reader(f)

            # loop over the rows in the index
            for row in reader:
                # parse out the image ID and features, then compute the
                # chi-squared distance between the features in our index
                # and our query features
                features = [float(x) for x in row[1:]]
                d = self.chi2_distance(features, queryFeatures)

                # now that we have the distance between the two feature
                # vectors, we can udpate the results dictionary -- the
                # key is the current image ID in the index and the
                # value is the distance we just computed, representing
                # how 'similar' the image in the index is to our query
                results[row[0]] = d

            # close the reader
            f.close()

        # sort our results, so that the smaller distances (i.e. the
        # more relevant images are at the front of the list)
        results = sorted([(v, k) for (k, v) in results.items()])

        # return our (limited) results
        return results[:limit]
```

在1行打开index.csv文件，在3行获取CSV读取器的句柄，接着在6行循环读取index.csv文件的每一行。

对于每一行，提取出索引化后的图像的颜色直方图，用11行的chi2_distance函数将其与待搜索的图像特征进行比较，该函数在下面介绍。

在32行使用唯一的图像文件名作为键，用与待查找图像的与索引后的图像的相似读作为值来更新results字典。

最后，将results字典根据相似读升序排序。

卡方相似度为零的图片表示完全相同。相似度数值越高，表示两幅图像差别越大。

说到卡方相似读，看下面的源码：

```python
def chi2_distance(self, histA, histB, eps = 1e-10):
    # compute the chi-squared distance
    d = 0.5 * np.sum([((a - b) ** 2) / (a + b + eps)
        for (a, b) in zip(histA, histB)])

    # return the chi-squared distance
    return d
```

chi2_distance函数需要两个参数，即用来进行比较的两个直方图。可选的eps值用来预防除零错误。

这个函数的名称来自皮尔森的卡方测试统计，用来比较离散概率分布。

由于比较的是颜色直方图，根据概率分布的定义，卡方函数是个完美的选择。

一般来说，直方图两端的值的差别并不重要，可以使用权重对其进行处理，卡方距离函数就是这么做的。

还能跟的上吗？我保证，最后一步是最简单的，仅仅需要将前面的各部分组合在一起。

第四步：执行搜索

如果我告诉你，执行搜索是最简单的一步，你信吗？实际上，只需一个驱动程序导入前面定义的所有的模块，将其依次组合成具有完整功能的CBIR系统。

所以新建最后一个文件，命名为search.py，这样我们的例子就能完成了：

```python
# import the necessary packages
from pyimagesearch.colordescriptor import ColorDescriptor
from pyimagesearch.searcher import Searcher
import argparse
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--index", required = True,
    help = "Path to where the computed index will be stored")
ap.add_argument("-q", "--query", required = True,
    help = "Path to the query image")
ap.add_argument("-r", "--result-path", required = True,
    help = "Path to the result path")
args = vars(ap.parse_args())

# initialize the image descriptor
cd = ColorDescriptor((8, 12, 3))
```

首先导入所需的包，导入第一步的ColorDescriptor来提取待查找图像的特征；导入第三步定义的Searcher类，用于执行执行实际的搜索。

argparse和cv2模块一直会导入。

在8-15行处理命令行参数。我们需要用一个 – index来表示index.csv文件的位置。

还需要 – query来表示带搜索图像的存储路径。该图像将与数据集中的每幅图像进行比较。目标是找到数据集中欧给你与待搜索图像相似的图像。

想象一下，使用Google搜索并输入“Python OpenCV tutorials”，会希望获得与Python和OpenCV相关的信息。

与之相同，如果针对相册构建一个图像搜索引擎，提交了一副关于云、大海上的帆船的图像，希望通过图像搜索引擎获得相似的图像。

接着需要一个 – result-path，用来表示相册数据集的路径。通过这个命令可以选择不同的数据集，向用户显示他们所需要的最终结果。

最后，在18行使用图像描述符提取相同的参数，就如同在索引化那一步做的一样。如果我们是为了比较图像的相似度（事实也正是如此），就无需改变数据集中颜色直方图的bin的数目。

直接将第三步中使用的直方图bin的数目作为参数在第四步使用。

这样会保证图像的描述是连续且可比较的。

现在到了进行真正比较的时候：

```python
# load the query image and describe it
```

```python
query = cv2.imread(args["query"])
features = cd.describe(query)

# perform the search
searcher = Searcher(args["index"])
results = searcher.search(features)

# display the query
cv2.imshow("Query", query)

# loop over the results
for (score, resultID) in results:
    # load the result image and display it
    result = cv2.imread(args["result_path"] + "/" + resultID)
    cv2.imshow("Result", result)
    cv2.waitKey(0)
```

在2行从磁盘读取待搜索图像，在3行提取该图像的特征。

在6和7行使用提取到的特征进行搜索，返回经过排序后的结果列表。

到此，所需做的就是将结果显示给用户。

在9行显示出待搜索的图像。接着在13-17行遍历搜索结果，将相应的图像显示在屏幕上。

所有这些工作完成后，就可以实际操作了。

继续阅读，看最终效果如何。

CBIR系统实战

打开终端，切换到代码所在的目录，执行下面的命令：

```
Python
$ python search.py --index index.csv --query queries/108100.png --result-path dataset
```







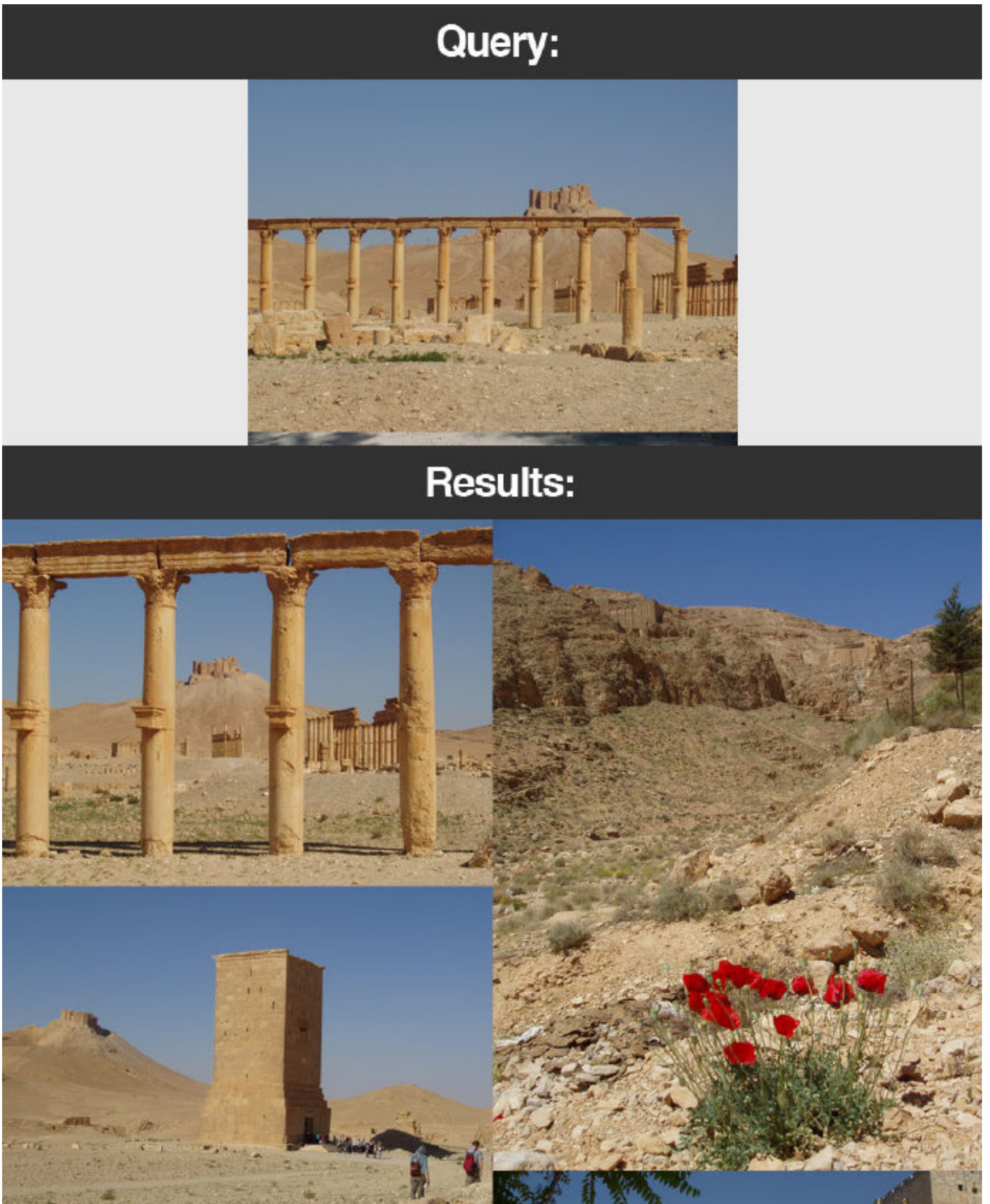



图18：在相册中搜索含有埃及金字塔的图像。

第一幅图像是待搜索的埃及金字塔。我们的目标是在相册中找到相似的图像。可以看到，在相册中找到了去金字塔游玩拍摄的照片。

我们还游览的埃及其他地方，所以用其他照片搜索试试：

```python
$ python search.py --index index.csv --query queries/115100.png --result-path dataset
```



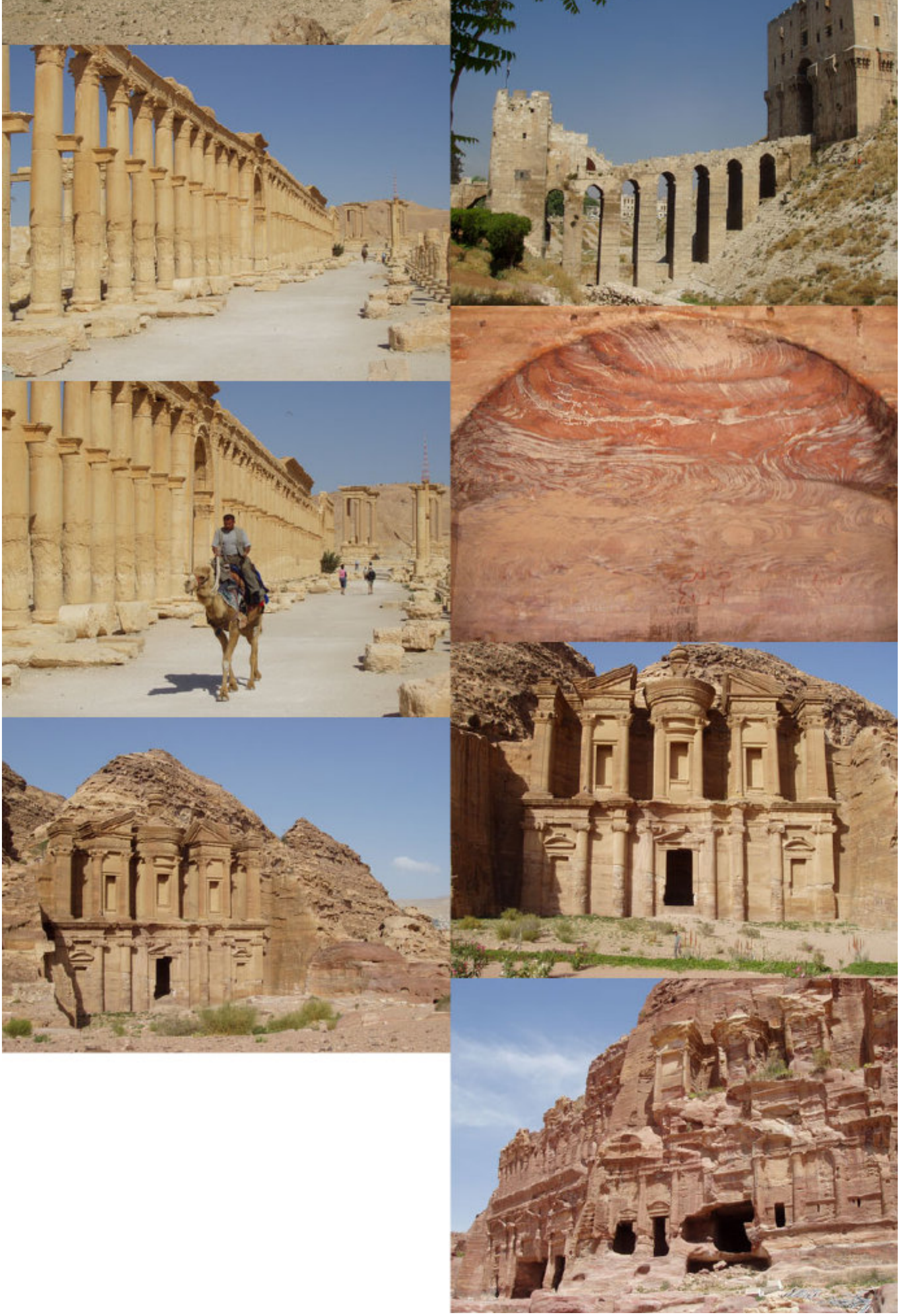


图19：使用搜索引起搜索埃及其他地方的图片，注意图中蓝天的位置。

注意我们的搜索图像中，上半部分是蓝天。中间和下半部分是褐色的建筑和土地。

可以肯定，图像搜索引擎会返回上半部分是蓝天，下半部分是棕褐色建筑和沙子的图像。

这是因为我们使用了本文开头介绍的基于区域的颜色直方图描述符。使用这种图像描述符可以粗略的针对每个区域执行，最后的结果中会含有图像每个区域的像素的密度。

旅途的最后一站是海滩，用下面的命令搜索海滩上的图像：

```python
$ python search.py --index index.csv --query queries/103300.png --result-path dataset
```

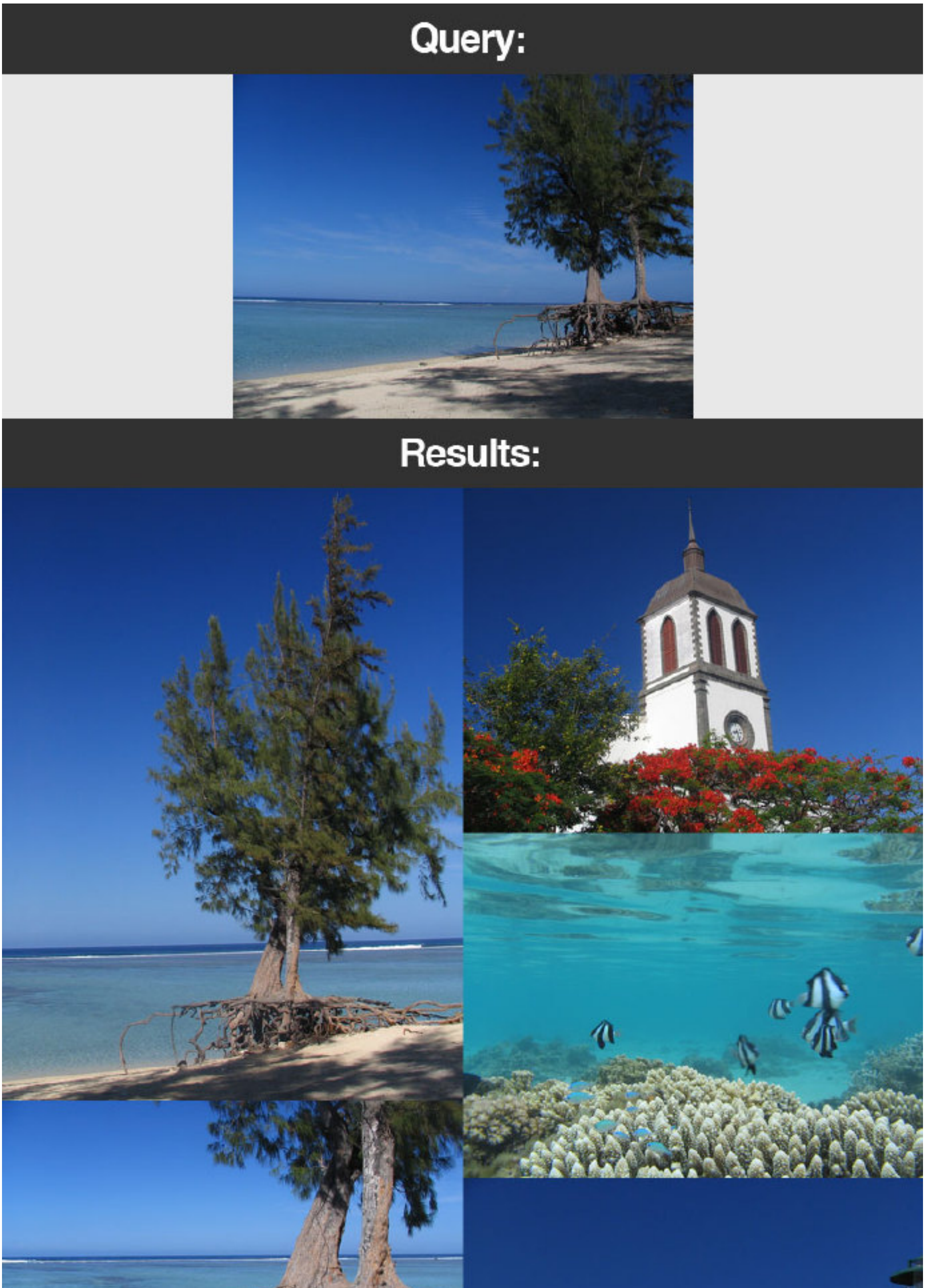


图20：使用OpenCV构建CBIR系统来搜索相册。

注意，前3个搜索结果是在相同地点拍摄到的图像。其他图像都含有蓝色的区域。

当然，没有潜水的海滩之旅是不完整的。

```
Python
1 $ python search.py --index index.csv --query queries/103100.png --result-path dataset
```

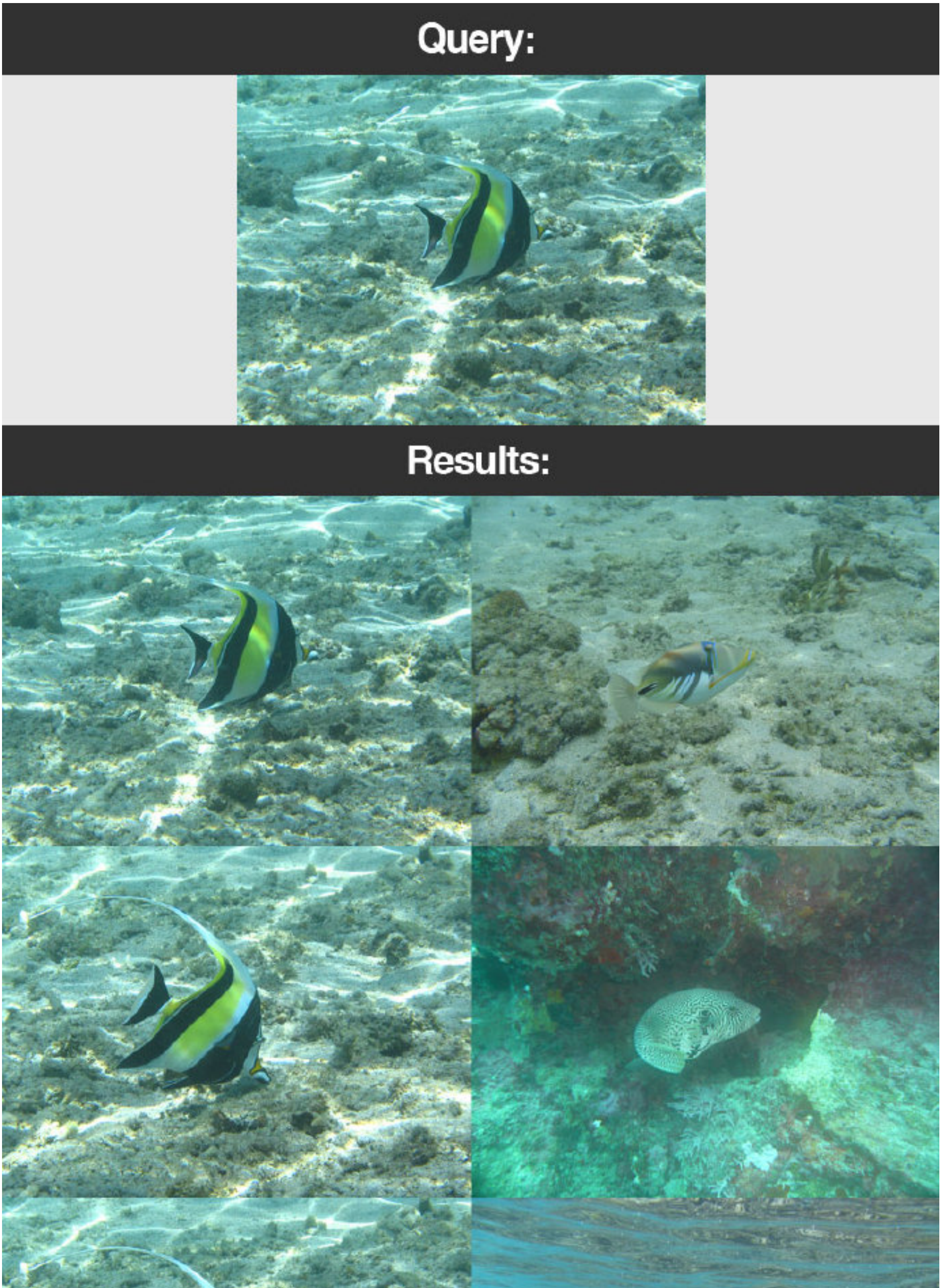
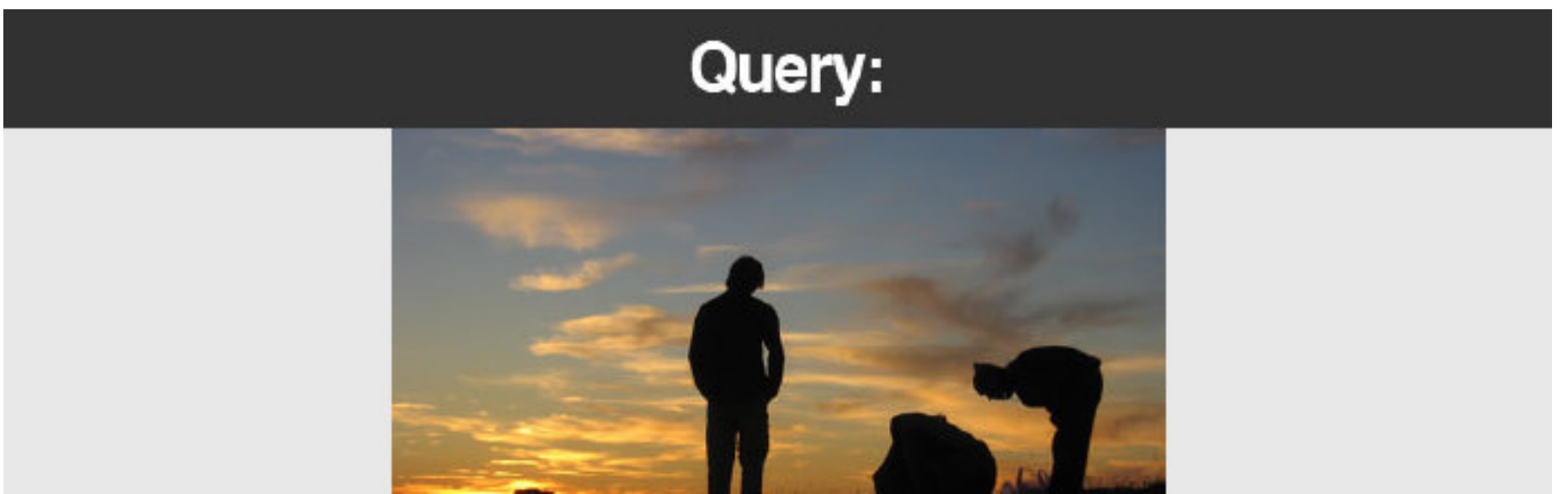


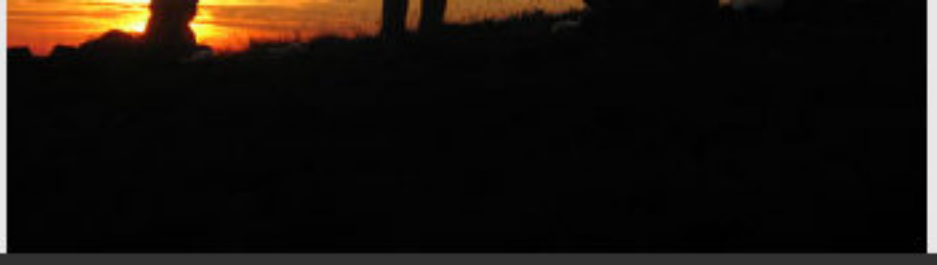
图21：图像搜索引起再次返回相关的结果。这次是水下冒险。

结果非常棒。前5个结果是同一条鱼，前10幅有9幅是水下探险。

最后，一天的旅途结束了，到了观看夕阳的时候：

```
Python
1 $ python search.py --index index.csv --query queries/127502.png --result-path dataset
```





Results:

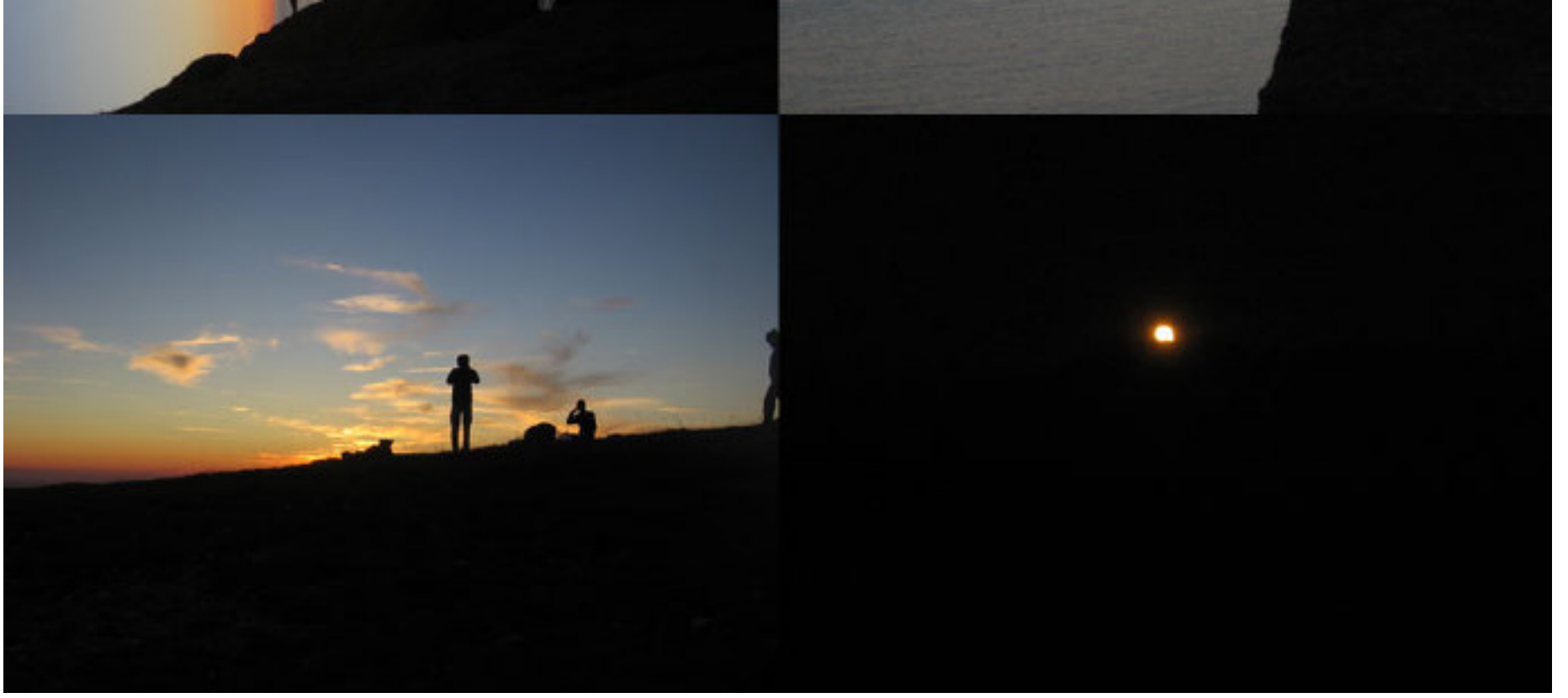


图22：这个OpenCV图像搜索引起可以查找到相册集中含有夕阳的相片。

搜索结果非常棒，所有的结果都含有夕阳。

这样，你就有了第一个图像搜索引擎：

总结

本文介绍了如何构建一个图像搜索引擎，来查找相册中的图像。

使用颜色直方图对相册中的图像的颜色部分进行分类。接着，使用颜色描述符索引化相册，提取相册中每一副图像的颜色直方图。

使用卡方距离比较图像，这是比较离散概率分布最常见的选择。

接着，实现了提交待搜索图像和返回查找结果的逻辑。

下一步

接下来该干什么？

可以看到，使用命令行是与这个图像搜索引擎交互的唯一方式。这样还不是太吸引人

下一篇文章将探索如何将这个图像搜索引擎封装进一个Python网络框架中，让其更易于使用。

下载：

代码和数据集总共有约200mb。如果读者想要下载文中用到的代码和图像。请在原文中输入你的邮箱地址，我会给你一个下载代码和数据集的链接。这个链接不仅能下载到代码的zip压缩包，还会收到我送给你的11页关于计算机视觉和图像搜索引擎的资源指南，其中含有文中没有介绍到的一些技术！是不是很不错？别犹豫，赶快在下面输入你的邮箱地址，我会立即给你发送代码的！ 在原文填写邮箱 （**要查看两次邮箱，还得翻墙**。不想来回倒腾的童鞋，请戳链接：http://pan.baidu.com/s/1ntsoamP）

¥ 打赏译者

3 赞　9 收藏

关于作者：Daetalus

Pyston核心开源开发者。熟悉CPython实现，关注Python科学计算。 个人主页 · 我的文章 · 27 ·

相关文章

- 使用Python和OpenCV在视频中实时监测条形码
- 利用Python和OpenCV将URL直接转换成OpenCV格式
- 在Raspberry Pi 2/B+上安装Python和OpenCV
- 用 Python 和 OpenCV 检测和跟踪运动对象
- 用树莓派 + Python + OpenCV 实现家庭监控和移动目标探测（下）
- 用 Opencv 和 Python 对汪星人做模糊检测
- 用Python和OpenCV来测量相机到目标的距离
- Python-OpenCV 处理图像（六）：对象识别
- Python-OpenCV 处理图像（五）：图像中边界和轮廓检测
- Python-OpenCV 处理图像（三）：图像像素点操作

可能感兴趣的话题

- 求助：关于 AngularJS 和 AJAX · 2
- 除了码代码，还有什么想做又还没有去做的事呢？ · 10
- 为什么那些很成功很有钱的人还是那么的努力，比之前还要的努力 · 21
- 工作上（或学习上）太追求完美，是一种怎样的体验？真会累了自己，还可能累... · 1
- 指令中的scope理解
- 天云山那个坑我的地方：（PS）节假日千万表要去 · 3
- 2017华为java工程师在线评测题 · 8
- 请问有什么职位是跟编程、英语、写作都相关的吗？ · 57
- 网易2017校园招聘算法题(找数字问题) · 12
- 2016美团java工程师在线笔试题(JVM)

« 使用Python写一个小小的项目监控
用Python在地图上模拟疫情扩散 »

登录后评论　新用户注册

直接登录 

最新评论

李俊雄 2015/02/02

看得很开心，谢谢！

赞 回复

奕克 2015/02/14

这样的stepbystep的教程写的很不错啊，但是在windows底下pycharm里运行，index.py里两处需要改动，imageID = imagePath[imagePath.rfind("\\") + 1:]和 output.write("%s,%s\n" % (imageID, ",".join(features)))，要不然生成的index.csv里面文件名带有路径，而且每幅图片之后不会回车换行。。。而且看几幅图片之后怎么还会出错，回头再看看哪有问题。。

1 赞 回复

sakura_mekhi （ 1 ） 2015/06/04

是空格换

1 赞 回复

刘娜 2015/03/23

写的很不错，收获很多。望楼主再多分享一些

赞 回复

刘娜 2015/03/23

写的很详细，收获很多，望楼主多分享些图像方面的文章，幸苦拉

赞 回复

Leon 2015/04/21

有个疑问，我看了pearson chi-square的wiki，http://en.wikipedia.org/wiki/Pearson%27s_chi-squared_test
里面只有pearson的chi-square test，和你代码中不同的地方是，它的分母只有b，而不是a+b，请问你这样的定义是怎么得到的么？

赞 回复

selfim 2015/05/05

很详细的教程，楼主辛苦了，，
但是 有个问题 想问一下 就是 查找时 是不是很慢呀

赞 回复

james 2015/05/13

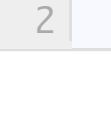
总共的特征向量不是8 × 12 × 3=288，而应该是8+12+3吧

赞 回复

sakura_mekhi （ 1 ） 2015/06/04

为了评论，特意注册个账号
colordescriptor.py这个文件中，这个 def histogram() 方法的hist = cv2.normalize(hist).flatten()这行，执行不通过，so，我小改了一下，顺便参考了一下官文档
hist = cv2.normalize(hist, hist).flatten()，这样就能执行了，之前就是不能执行会报一个参数错误。

2 赞 回复



小编辑 （ 10 · ） 2015/06/04
编辑

请 @Daetalus @sunbiaobiao 核实一下，谢谢~

赞 回复

Daetalus （ 27 · ） 2015/06/05
精华评论

感谢细心的读者，回复如下：

简单来说，尽量用hist = cv2.normalize(hist, hist)这样的形式！

测试结果：

- 在Mint 64位，OpenCV 2.4.8版本中，hist = cv2.normalize(hist).flatten()可以通过。
- 在Windows 8.1 64位，OpenCV 3.0.0beta版本中，hist = cv2.normalize(hist).flatten()无法通过。必须像你这样修改才可以。

建议：

在Python中使用cv2.normalize(hist, hist)时，最好一直把第一个参数（原图像）和第二参数归一化后的图像，必须和原图像“大小”相同）都填上，不要像原文作者中那样使用。

研究：

在OpenCV的官方文档中，关于cv.normalize()的解释。 无论是3.x版本，见此： http://docs.opencv.org/3.0-last-rst/modules/core/doc/operations_on_arrays.html#normalize 还是2.x版本，见此： http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#normalize。都是以下形式：

Python

```
cv2.normalize(src[, dst[, alpha[, beta[, norm_type[, dtype[, mask]]]]]]) → dst

```

这里的方括号表示第二个参数是可选的，该函数会返回处理后的结果。

所以理论上作者的那种写法是正确的！！

验证：

Python

```
        hist1 = hist.copy()
        # 复制hist来新建一个hist1，通过下面的语句可以验证是与原图相同
        print("Does hist1 equal hist befor normalize?" + str(np.array_equal(hist, hist1)))
        hist2 = cv2.normalize(hist, hist1)
```

```
        # 处理后就会发现，hist1和hist不再相同
        print("Does hist1 equal hist after normalize?" + str(np.array_equal(hist, hist1)))
        # 将hist1作为第二个参数传给normalize，用来存储处理结果。
        # 同时用hist2作为normalize返回的结果。
        # 通过下面的语句会发现，hist2和hist1其实是同一个东西
        print("hist1 id is: " + str(id(hist1)))
        print("hist2 id is: " + str(id(hist2)))
        print("Does hist1 is hist2 after normalize?" + str(hist2 is hist1))
        print("Does hist1 equal hist2 after normalize?" + str(np.array_equal(hist2, hist1)))
```

这就相当于

dst = cv2.normalize(hist, dst)

等同于

cv2.normalize(hist, dst)

如果不是用其他可选参数的话，这样做其实没多少意义，仅仅是符合OpenCV C++的调用方式。前者是多此一举，后者也变得不Pythonic了。

另一方面，在可以使用dst = cv2.normalize(hist)这种形式时，笔者猜测其实在函数中会自动创建了一个ndarray来持有处理结果，而现在则必须手动创建一个与原图“大小”相同的变量（也可以使用原图hist）来持有处理结果。

关于最后一点的猜想，有待进一步的验证。如果遇到cv2.normalize的话，就把第二个参数一定加上吧。

2 赞 回复

sakura_mekhi （ 1 ） 2015/06/05

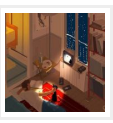
感谢你细心的解答，我也是实验加官方文档得出的结论。我用的是mac + opecv3.0 的环境，所以就会报第二个参数的错误。谢谢您，相互交流，学习。

1 赞 回复

JackPy （ 1 · ） 2015/06/17

写得很不错，评论也精彩，有时间就玩一下呗

赞 回复



v_胡桃 （ 1 · ） 2015/08/14

imagePath[imagePath.rfind("\\") + 1:] 这段语法不太明白。特别是1:这个地方，求指教。

赞 回复

zoewxx （ 1 ） 2015/08/21

请问 下我在linux下运行这个程序的时候为什么显示该命令不存在?

zoegreen@zoegreen-Lenovo-IdeaPad-Y470:~/桌面/vacation-image-search-engine$ $ python search.py --index index.csv --query queries/115100.png --result-path dataset
$: 未找到命令

赞 回复

笨兔 （ 1 · ） 2015/10/14

请问一下，我在运行100100.png为什么显示了一下错误，是系统的问题吗?
PoohmatoMacBook-Air:vacation-image-search-engine rabbity$ python search.py --index index.csv --query queries/100100.png --result-path dataset
OpenCV Error: Assertion failed (size.width>0 && size.height>0) in imshow, file /tmp/opencv20151002-13652-vgf2gw/opencv-2.4.12/modules/highgui/src/window.cpp, line 261
Traceback (most recent call last):
File "search.py", line 39, in
cv2.imshow("Result", result)
cv2.error: /tmp/opencv20151002-13652-vgf2gw/opencv-2.4.12/modules/highgui/src/window.cpp:261: error: (-215) size.width>0 && size.height>0 in function imshow
求指教，多谢啦！

赞 回复

雪前小雨 （ 1 ）
学生

2015/12/13

你好，我自己搭建了一个python3.4.3+opencv3.0.0的环境，但是在用python3 search.py --index index.csv --query queries/127502.png --result-path dataset运行你的代码时，出现了这个错误： cv2.ellipse(ellipMask, (cX, cY), (axesX, axesY), 0, 0, 360, 255, -1)
TypeError: ellipse() takes at most 5 arguments (8 given)
后来我去官方看文档，我觉得你的函数调用也没错，在网上查找时，有人碰到类似问题，但是也没解决。我自己照官方例子来写，ellipse函数时可以这样调用并且画出一个椭圆。希望您能帮忙解答下。还有我想知道您这个例子是什么版本的python和opencv呢?

赞 回复

雪前小雨 （ 1 ）
学生

2015/12/13

您好，我把您的代码跑起来了，我觉得有一个错误，不知道是不是，我改了后能 跑了。在您代码文件colordescriptor.py中的代码:
(axesX, axesY) = (int(w * 0.75) / 2, int(h * 0.75) / 2)
ellipMask = np.zeros(image.shape[:2], dtype = "uint8")
cv2.ellipse(ellipMask, (cX, cY), (axesX, axesY), 0, 0, 360, 255, -1)
ellipse函数传递的参数应该是只能有int型，但是您的axesX和axexY有可能是double型的。我把(axesX, axesY) = (int(w * 0.75) / 2, int(h * 0.75) / 2)改为：(axesX, axesY) = (int(w * 0.375） , int(h * 0.375）)就能跑起来了。如果不是这个错误，打扰了。我刚学python，可能问的问题会比较低级。

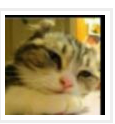
赞 回复

阿芳92 （ 1 · ）

03/16

同出现报错，这个问题参见这个解答 http://stackoverflow.com/questions/18595099/python-opencv-how-i-can-use-cv2-ellipse
具体原因是由于python3对于除法的使用和python2不同导致的，估计原代码的运行环境是python2，所以把/改为//应该是正确解决方法。

赞 回复



雪前小雨 （ 1 ）
学生

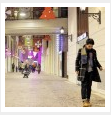
2015/12/13

您好，我把您的代码跑起来了，我觉得有一个错误，不知道是不是，我改了后能 跑了。在您代码文件colordescriptor.py中的代码:
(axesX, axesY) = (int(w * 0.75) / 2, int(h * 0.75) / 2)
ellipMask = np.zeros(image.shape[:2], dtype = "uint8")
cv2.ellipse(ellipMask, (cX, cY), (axesX, axesY), 0, 0, 360, 255, -1)
ellipse函数传递的参数应该是只能有int型，但是您的axesX和axexY有可能是double型的。我把(axesX, axesY) = (int(w * 0.75) / 2, int(h * 0.75) / 2)改为：(axesX, axesY) = (int(w * 0.375） , int(h * 0.375）)就能跑起来了。如果不是这个错误，打扰了。我刚学python，可能问的问题会比较低级。

赞 回复

Superman （ 1 ）

2015/12/30



为什么我执行 index.py的时候 显示的是SyntaxError: invalid syntax 呢?

赞 回复

Python

输入搜索关键字

搜索

Python小组话题

我有新话题

在大公司做测试，Python 要达到怎...
池浅 发起 • 5 回复

可控制调度执行的celery小插件 celery-tas...
BaoEr 发起

已经有了python基础，正在学习数据分...
Michael_翔_ 发起 • 4 回复

零基础自学Python感觉很难，不像大...
keepcalm 发起 • 42 回复

Jpype效率如何？又有哪些问题呢
成融融融融融 发起

轨迹点数据坐标经纬度可视化的时候用...
nature 发起 • 2 回复

- 本周热门Python文章
- 本月热门
- 热门标签

0 Python 框架介绍

1 一些实用的 python 小建议

2 十行代码实现牛顿方法

3 用Python创建Excel高级工作表

4 Virtualenv + SublimeText 的Pyt...

5 R语言和 Python —— 一个错误...

6 python中的魔术方法__

7 Python 多进程实践

8 Python map, reduce, filter和sor...

9 Python数据结构——树的实现

Python工具资源 **更多资源 »**

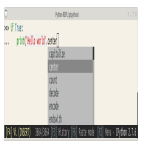
SimpleCV

SimpleCV：开源计算机视觉框架
Python, 计算机视觉

SciPy库：Python的科学计算工具集
科学计算和数据分析

[NumPy：Python科学计算的基石]
Python, 科学计算和数据分析

[webssh：基于tornado的web linux终端]
Python



[Python Prompt Toolkit：构建强大交互式命令行的 Pyt...]
Python, 开发库 · 1

最新评论

- 
  Re: scrapy入门教程2：建立一个简单的爬...
  start_urls = [\"http://news.njupt.edu.cn/s/22...
-
  Re: scrapy入门教程2：建立一个简单的爬...
  第一个parse 里面 返回的item　　不是会多次执行吗　这样
-
  Re: Python源码阅读——list
  请问一下作者，我现在在用python，对于python的垃圾回收，魔术方法，元类，装饰器等都有了一定...
-
  Re: Python爬虫入门（8）：Beautiful...
  这一节确实太多了
-
  Re: Pandas透视表（pivot_table）...
  很赞！
-
  Re: Python下用Scrapy和MongoDB...
  DOWNLOAD_DELAY = 5BOT_NAME = \'stack\'SPIDER_MODULES...
-
  Re: 使用flask开发RESTful架构的api服务器端...
  代码好像未经测试就发布了啊，很多地方报错：(
-

Re: python-文本处理和正则表达式
import resplit=re.compile(\'d座|客车\')try: result=s...

关于 Python 频道

Python频道分享 Python 开发技术、相关的行业动态。

快速链接
问题反馈与求助 »
Python工具资源 »
Python技术话题 »

关注我们

新浪微博：@Python开发者
RSS：订阅地址
推荐微信号

Python开发者　Linux爱好者　数据库开发

合作联系
Email：bd@Jobbole.com
QQ：2302462408 （加好友请注明来意）

更多频道

小组 – 好的话题、有启发的回复、值得信赖的圈子
头条 – 分享和发现有价值的内容与观点
相亲 – 为IT单身男女服务的征婚传播平台
资源 – 优秀的工具资源导航
翻译 – 翻译传播优秀的外文文章
文章 – 国内外的精选文章
设计 – UI,网页，交互和用户体验
iOS – 专注iOS技术分享
安卓 – 专注Android技术分享
前端 – JavaScript, HTML5, CSS
Java – 专注Java技术分享
Python – 专注Python技术分享