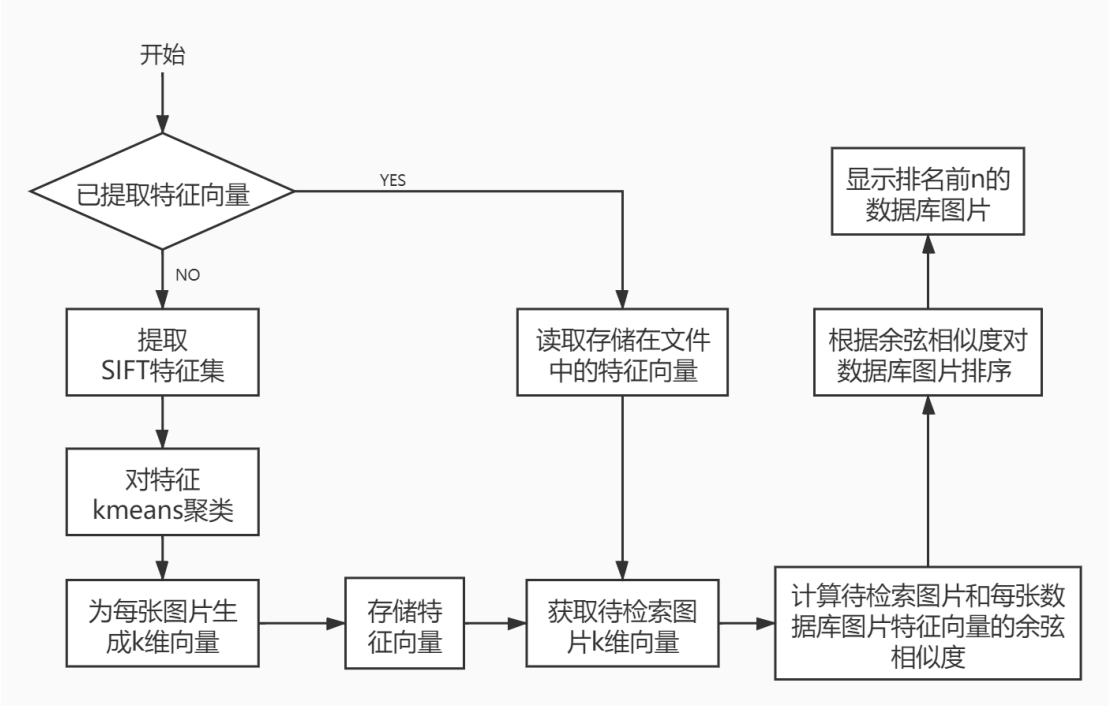
系统功能描述

将待检索图片放入./aims 文件夹下，运行 code.py 检索每张图片在./oxbuild_images 文件夹下最匹配的前 n 张图片并进行展示。

系统设计



核心算法设计

总体设计

总体想法和第二次作业很类似，先将数据集中每张图片转换为 k 维向量；将待检索图片同样转换为 k 维向量，求数据集中每张图片的向量和待检索图片向量的余弦相似度，对余弦相似度进行降序排序，显示前 n 张图片作为结果。但是经过试验检索效果其实很一般，同时我注意到人在观察图片时注意力一般会放在图片中央附近，因此我在生成特征向量时引入了高斯核加权机制，即越靠近中间的特征对特征向量贡献越大。

生成高斯核加权特征向量集

对数据集中每张图片提取 sift 特征，把所有提取到的 sift 特征 kmeans 聚类，得到 k 个中心点；对数据集中的每张 x*y 的图片，生成一个 x*y 的高斯核，对于此图片中提取到的每个 sift 特征，根据特征位置(s_x,s_y)在高斯核中的取值加权。但由于每张图片的 sift 特征数相同，因此尺寸较大的图片对应的大尺寸高斯核会导致特征向量被衰减，因此经过考虑，我将高斯核取值乘以 x*y（图片面积）得到最终的权值。

计算余弦相似度集

同样的方法求得待检索图片的加权特征向量，计算待检索图片和数据集中每张图片特征向量的余弦相似度，这样得到了一个余弦相似度集。权值引入在余弦相似度计算中表现为中央部分的差异会被放大，使余弦相似度变小，这样能够区分内容不同的图片。

结果选取和显示

对余弦相似度集进行降序排序，选取前 n 张图片进行排版展示。

系统实现

算法实现详见 code.py

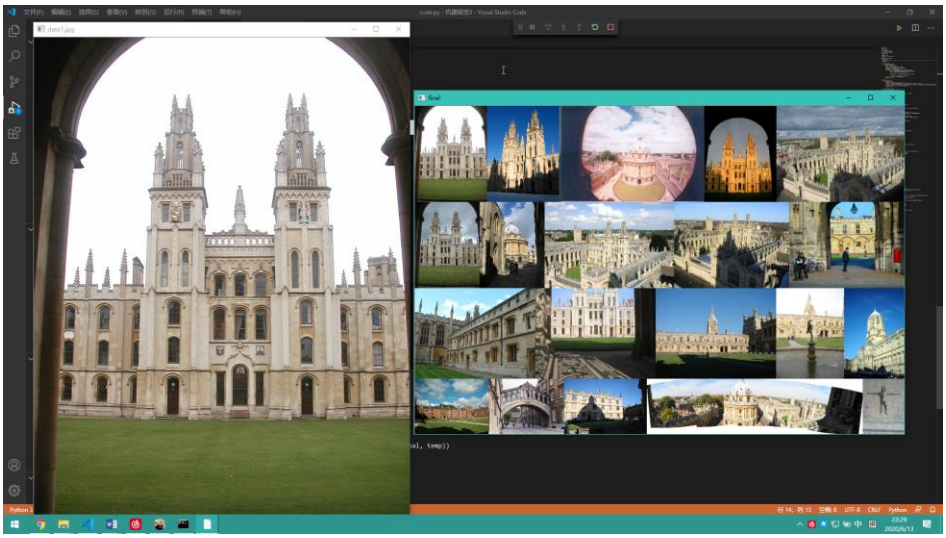
sift 提取和 kmeans 均调用 cv2 库实现。

二维高斯核用大小为 x 的一维高斯核乘大小为 y 的一维高斯的逆核得到。

界面实现

利用 cv2.resize、np.hatack、np.vstack，两层循环将多个检索结果无缝合并成一张大图片展示。

显示效果大致如下（参数 x=5,y=4,height=200,length=1000）。



核心算法评价

实验数据集

oxford building 5k 牛津建筑物数据集

(http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/)

实验参数设置

实验中尝试了以下参数的不同取值：

SIFT_create 的参数，提取的 sift 特征数目

sift_num=300

kmeans 的 k

wordCnt = 150

性能分析

时间效率方面，余弦计算速度非常快，检索时间基本可以忽略不计。主要时间花费

在特征向量集生成，其中 sift 特征数影响特征提取步骤和特征向量生成的速度，（取消 SIFT_create 的参数会让 featureSet 过长，进行了两次尝试均引起了存溢出），影响速度的同时，sift 数量也能影响检索结果准确度。

k 值显然会影响 kmeans 的速度，同时影响检索结果准确度。

鉴于大量时间花费在特征向量集的生成上，我保存了不同参数下的特征向量集和聚类结果，程序第二次运行可以直接读取，进行快速检索。

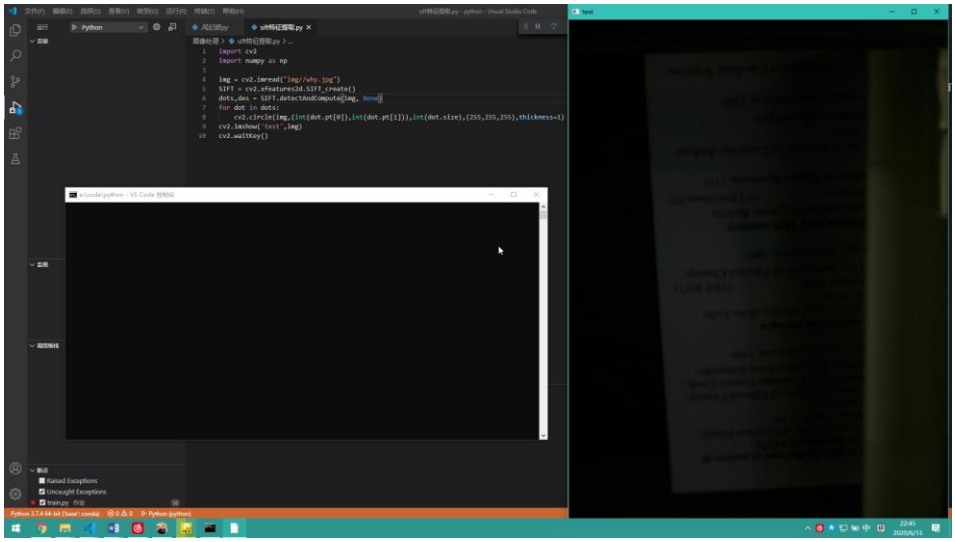
准确率方面，由于找不到好的准确率描述方法，只能人眼判断大致准确率。高斯加权机制的引入肉眼可见的提高了大部分图片的检索准确率。

但同时，我注意到，中心加权机制其实本质上是利用了对称构图的规律，对于非对称构图的图片检索效果并不是很好；对此我想到如果能采取较大物体上的特征权值更大的加权机制，应该会使检索准确度更上一层。很可惜目前自己学艺不精，没能成功实现。

系统测试

测试集中放入 1 张牛津大学图片网图(来自网络)，1 张数据集中的建筑图片，1 张无关图片(来自网络)

实验中发现有一张图片(ashmolean_000214.jpg)总是提取不到 sift 特征(提取结果是 None)，推断可能是因为拍摄光线太暗而且对焦不准……。



对于这张图片我不再过多研究，直接忽略。详细实验过程及检索结果详见展示视频。

结论

本系统对非对称构图的图片和尺寸过小的图片检索准确率较低。

优势：

突出图片中心位置特征对图片检索的贡献。

余弦相似度计算速度可观，检索速度快。

二次运行可以直接读取文件进行检索。

缺点：

生成特征向量集的速度很慢。

非对称构图的图像检索效果较差。