**《计算机视觉》实验报告**

**姓名：刘远航 学号：22121883**

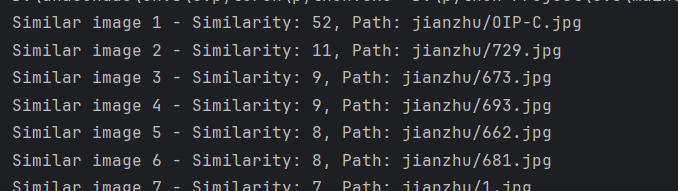
实验八

1. **采用SIFT特征实现图像检索功能，即输入一张图片，在数据集中检索出相似的图片，数据集自选。**
   1. **核心代码：**

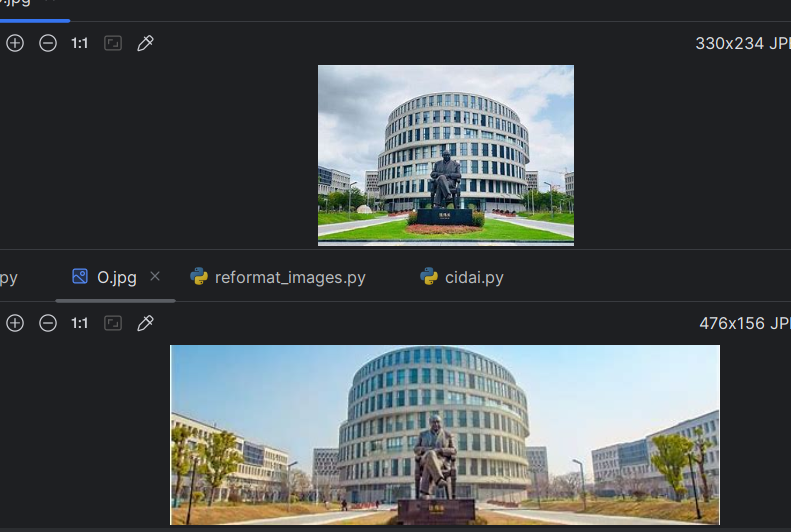
import cv2  
import numpy as np  
from os import listdir  
def extract\_sift\_features(image\_path):  
 # 读取图像并转换为灰度图  
 image = cv2.imread(image\_path)  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 # 创建SIFT对象  
 sift = cv2.SIFT\_create()  
 # 检测关键点和计算描述符  
 keypoints, descriptors = sift.detectAndCompute(gray, None)  
 return keypoints, descriptors  
def find\_similar\_images(query\_image\_path, dataset\_folder):  
 # 提取待检索图片的SIFT特征  
 query\_keypoints, query\_descriptors = extract\_sift\_features(query\_image\_path)  
  
 # 遍历数据集中的每张图片，并计算相似度  
 similarities = []  
 for image\_file in listdir(dataset\_folder):  
 image\_path = dataset\_folder + "/" + image\_file  
 keypoints, descriptors = extract\_sift\_features(image\_path)  
  
 # 创建FLANN匹配器  
 FLANN\_INDEX\_KDTREE = 0  
 index\_params = dict(algorithm=FLANN\_INDEX\_KDTREE, trees=5)  
 search\_params = dict(checks=50)  
 flann = cv2.FlannBasedMatcher(index\_params, search\_params)  
  
 # 使用KNN匹配器计算特征相似度  
 matches = flann.knnMatch(query\_descriptors, descriptors, k=2)  
  
 # 计算相似度  
 good\_matches = []  
 for m, n in matches:  
 if m.distance < 0.7 \* n.distance:  
 good\_matches.append(m)  
  
 similarity = len(good\_matches)  
 similarities.append((image\_path, similarity))  
  
 # 根据相似度排序  
 similarities.sort(key=lambda x: x[1], reverse=True)  
  
 return similarities  
# 测试  
query\_image\_path = "O.jpg"  
dataset\_folder = "jianzhu"  
similar\_images = find\_similar\_images(query\_image\_path, dataset\_folder)  
# 输出相似图片  
for i, (image\_path, similarity) in enumerate(similar\_images):  
 print("Similar image {} - Similarity: {}, Path: {}".format(i + 1, similarity, image\_path))

* 1. **实验结果截图**

检所结果会按照相似度进行排序：

****

取最相似的与原图对比：



* 1. **实验小结**

这次实验是用sift特征进行图像检索，搜集到数据集后进行对比，发现确实能检索到相似的图片，很好的完成了实验任务。

1. **选做： （1）基于词袋模型实现（2）检索结果按照相似度进行排序**
   1. **核心代码：**

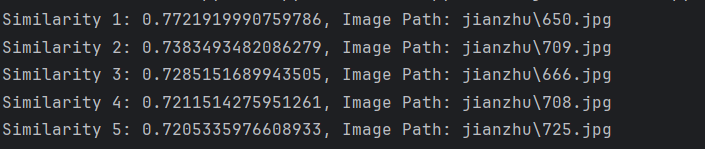
构建词袋模型：

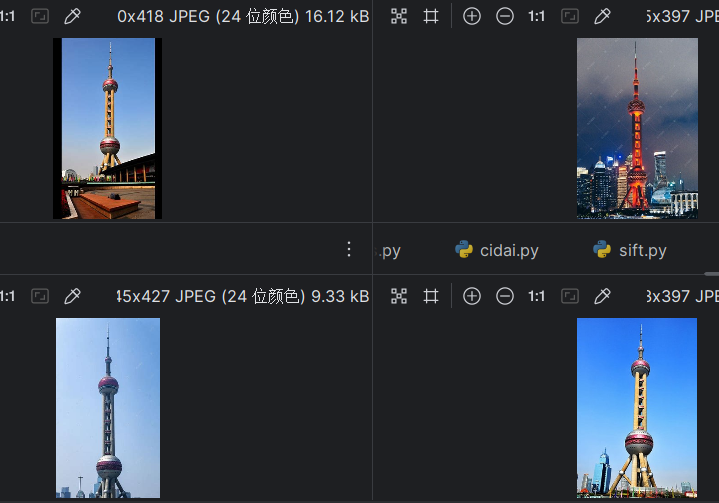
def extract\_sift\_features(image):  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 sift = cv2.SIFT\_create()  
 keypoints, descriptors = sift.detectAndCompute(gray, None)  
 return descriptors  
  
# Step 2: 从数据集中提取所有特征  
def extract\_features\_from\_dataset(dataset\_path):  
 all\_descriptors = []  
 for filename in os.listdir(dataset\_path):  
 if filename.endswith(".jpg") or filename.endswith(".png"):  
 image\_path = os.path.join(dataset\_path, filename)  
 image = cv2.imread(image\_path)  
 descriptors = extract\_sift\_features(image)  
 if descriptors is not None:  
 all\_descriptors.extend(descriptors)  
 return np.array(all\_descriptors)  
  
# Step 3: 使用K均值聚类构建词袋模型  
def build\_vocabulary(features, k):  
 kmeans = KMeans(n\_clusters=k)  
 kmeans.fit(features)  
 return kmeans.cluster\_centers\_  
  
# 示例用法  
dataset\_path = "jianzhu"  
k = 100 # 选择聚类中心的数量  
all\_features = extract\_features\_from\_dataset(dataset\_path)  
vocabulary = build\_vocabulary(all\_features, k)  
  
# 将词袋模型保存为文件  
np.save("vocabulary.npy", vocabulary)

待测图像检验：

def extract\_sift\_features(image):  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 sift = cv2.SIFT\_create()  
 keypoints, descriptors = sift.detectAndCompute(gray, None)  
 return keypoints, descriptors  
# 2. 构建词袋模型  
def build\_vocabulary(descriptors, k):  
 kmeans = KMeans(n\_clusters=k)  
 kmeans.fit(descriptors)  
 return kmeans.cluster\_centers\_  
# 3. 计算图像的特征向量  
def calculate\_feature\_vector(descriptors, vocabulary):  
 feature\_vector = np.zeros(len(vocabulary))  
 if descriptors is None:  
 return feature\_vector  
 distances = cosine\_similarity(descriptors, vocabulary)  
 nearest\_word = np.argmax(distances, axis=1)  
 for word in nearest\_word:  
 feature\_vector[word] += 1  
 return feature\_vector  
# 4. 计算相似度  
def calculate\_similarity(query\_vector, dataset\_vectors):  
 similarities = cosine\_similarity([query\_vector], dataset\_vectors)  
 return similarities[0]  
# 5. 图像检索  
def image\_retrieval(query\_image\_path, dataset\_path, k):  
 query\_image = cv2.imread(query\_image\_path)  
 query\_keypoints, query\_descriptors = extract\_sift\_features(query\_image)  
 vocabulary = np.load("vocabulary.npy") # 加载预先计算好的词袋模型  
 query\_vector = calculate\_feature\_vector(query\_descriptors, vocabulary)  
 dataset\_vectors = []  
 image\_paths = []  
 for filename in os.listdir(dataset\_path):  
 if filename.endswith(".jpg") or filename.endswith(".png"):  
 image\_path = os.path.join(dataset\_path, filename)  
 image = cv2.imread(image\_path)  
 \_, descriptors = extract\_sift\_features(image)  
 vector = calculate\_feature\_vector(descriptors, vocabulary)  
 dataset\_vectors.append(vector)  
 image\_paths.append(image\_path)  
  
 similarities = calculate\_similarity(query\_vector, dataset\_vectors)  
 sorted\_indices = np.argsort(similarities)[::-1]  
 # 输出相似度最高的前k张图片路径  
 for i in range(k):  
 print(f"Similarity {i+1}: {similarities[sorted\_indices[i]]}, Image Path: {image\_paths[sorted\_indices[i]]}")  
# 示例用法  
query\_image\_path = "d.jpg"  
dataset\_path = "jianzhu"  
k = 5  
image\_retrieval(query\_image\_path, dataset\_path, k)

* 1. **实验结果截图**

****

****

* 1. **实验小结**

经过这次实验，我明白了词袋模型的具体实现步骤，也明白了实现图像检索所需要的步骤，很好的完成了实验任务。