《计算机视觉》实验报告

姓名：刘远航 学号：22121883

实验八

一. 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。

a) 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)
```

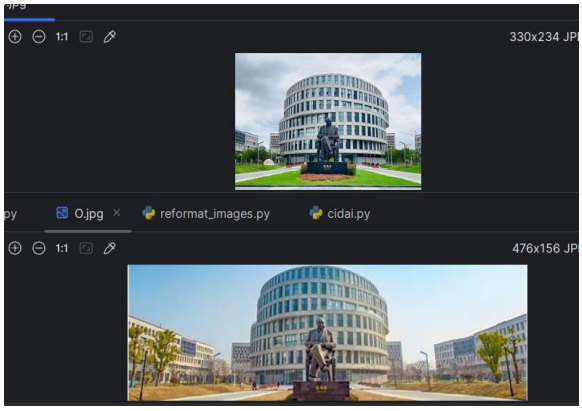
```
            # 使用 KNN 匹配器计算特征相似度
            matches = flann.knnMatch(query_descriptors, descriptors, k=2)

            # 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))
```

b) 实验结果截图

检所结果会按照相似度进行排序：

```
Similar image 1 - Similarity: 52, Path: jianzhu/OIP-C.jpg
Similar image 2 - Similarity: 11, Path: jianzhu/729.jpg
Similar image 3 - Similarity: 9, Path: jianzhu/673.jpg
Similar image 4 - Similarity: 9, Path: jianzhu/693.jpg
Similar image 5 - Similarity: 8, Path: jianzhu/662.jpg
Similar image 6 - Similarity: 8, Path: jianzhu/681.jpg
Similar image 7 - Similarity: 7, Path: jianzhu/1.jpg
```

取最相似的与原图对比：



c) 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。

二. **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**

a) 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：

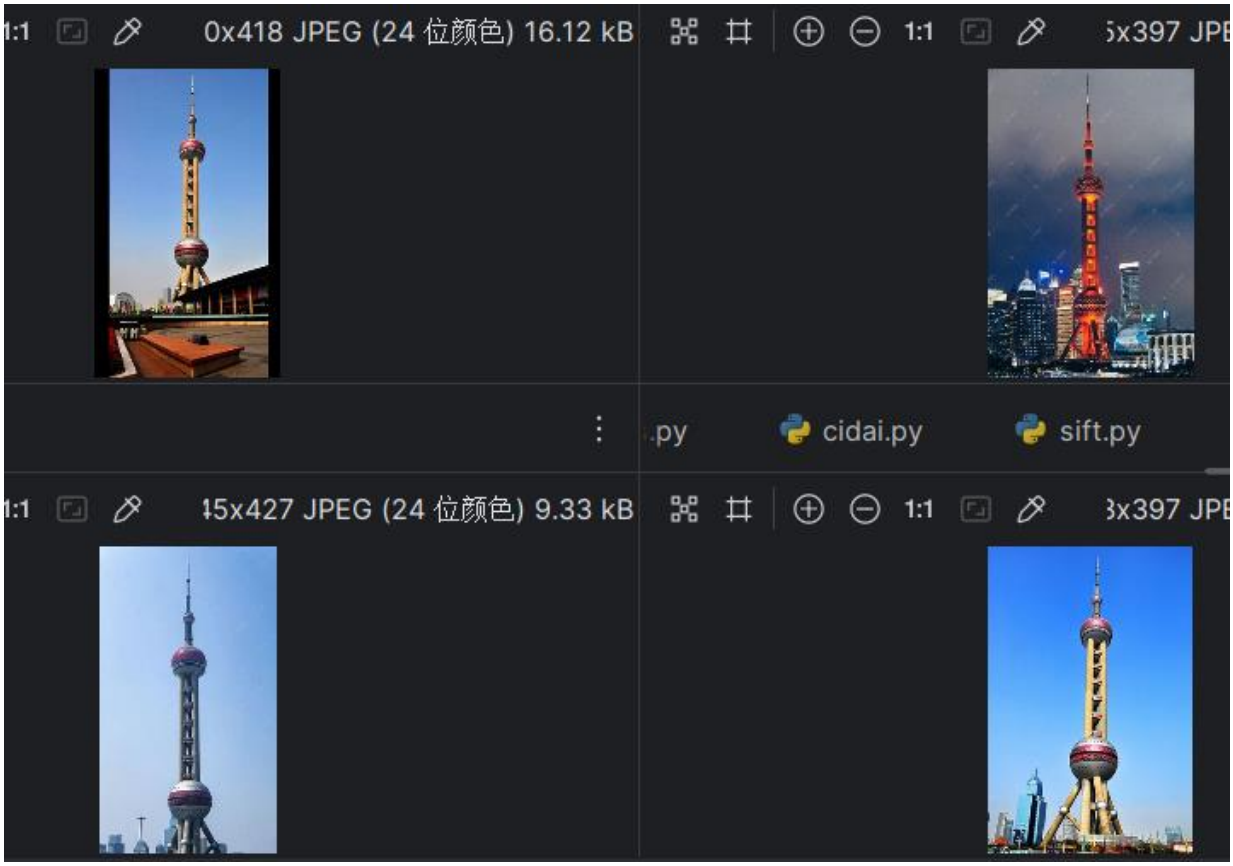
```
def extract_sift_features(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
    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)
```

b) 实验结果截图

```
Similarity 1: 0.7721919990759786, Image Path: jianzhu\650.jpg
Similarity 2: 0.7383493482086279, Image Path: jianzhu\709.jpg
Similarity 3: 0.7285151689943505, Image Path: jianzhu\666.jpg
Similarity 4: 0.7211514275951261, Image Path: jianzhu\708.jpg
Similarity 5: 0.7205335976608933, Image Path: jianzhu\725.jpg
```



c) 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。