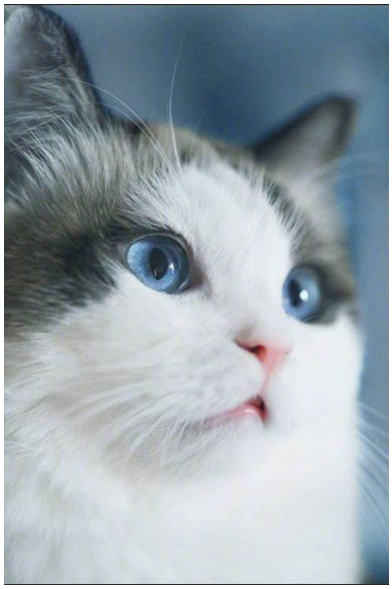
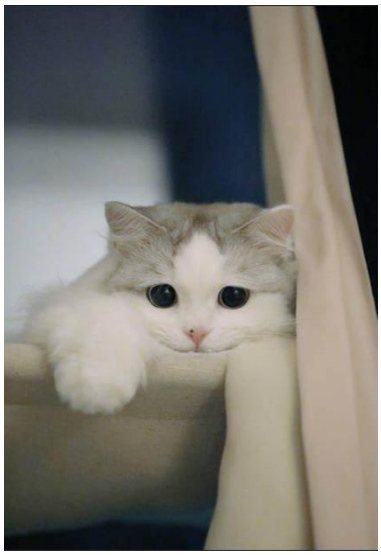
《计算机视觉》（本科，2025）作业 3

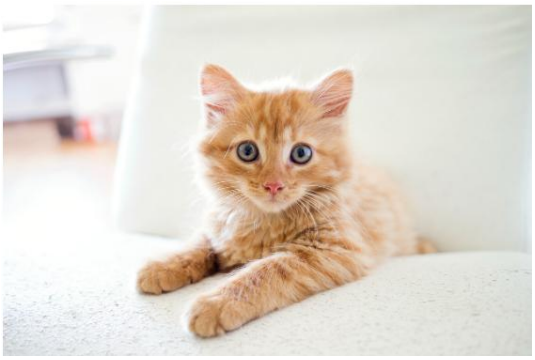
1. 在搜索引擎上按照某一关键词，搜索 50 张不同的图像，从中选出 5 张作为检索请求，另 45 张作为被检索图像。

【5 张检索请求图像贴这里】





【45 张被检索图像贴这里】



2. 以全局 RGB 颜色直方图（每通道 bin 的数量为 8）作为特征，进行图像检索。展示每个检索请求及对应的前 3 个结果。

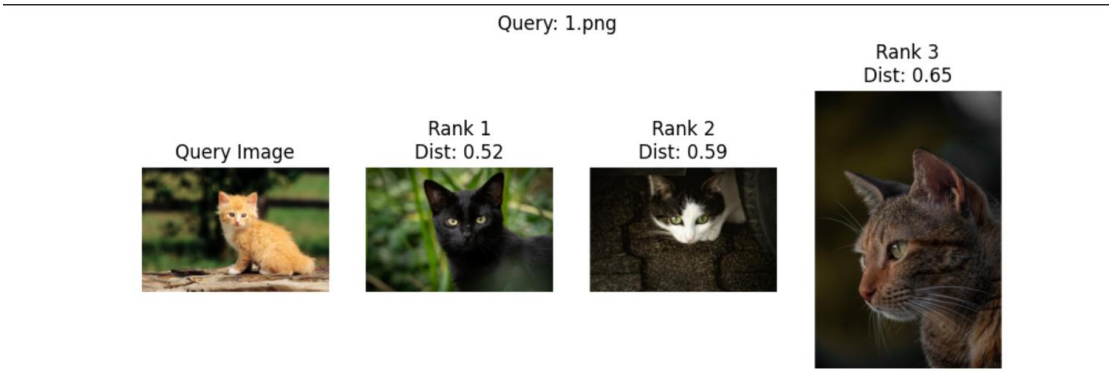
【特征抽取代码贴这里】

```
# 计算RGB直方图（8 bins/通道）
2 usages
def compute_histogram(image_path):
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # 转为RGB格式
    hist = cv2.calcHist( images: [image], channels: [0, 1, 2], mask: None, histSize: [8, 8, 8], ranges: [0, 256, 0, 256, 0, 256])
    hist = cv2.normalize(hist, hist).flatten()  # 归一化并展平为1D向量
    return hist
```

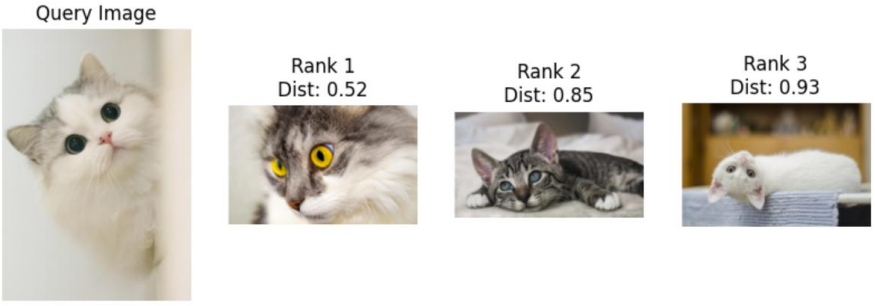
【检索（特征匹配）代码贴这里】

```
# 加载所有图像直方图
database = {}
for img_name in os.listdir(db_dir):
    img_path = os.path.join(db_dir, img_name)
    database[img_name] = compute_histogram(img_path)

# 处理每个检索请求
for query_name in os.listdir(query_dir):
    query_path = os.path.join(query_dir, query_name)
    query_hist = compute_histogram(query_path)

    # 计算与所有被检索图像的相似度（欧氏距离）
    distances = {}
    for db_name, db_hist in database.items():
        distance = np.linalg.norm(query_hist - db_hist)
        distances[db_name] = distance

    # 按距离排序，取前3名
    sorted_results = sorted(distances.items(), key=lambda x: x[1])[:3]

    # 可视化结果
    fig = plt.figure(figsize=(10, 4))
    plt.subplot( *args: 1, 4, 1)
    query_img = cv2.cvtColor(cv2.imread(query_path), cv2.COLOR_BGR2RGB)
    plt.imshow(query_img)
    plt.title("Query Image")
    plt.axis('off')
```

【结果（检索请求+前 3 个结果）贴这里】

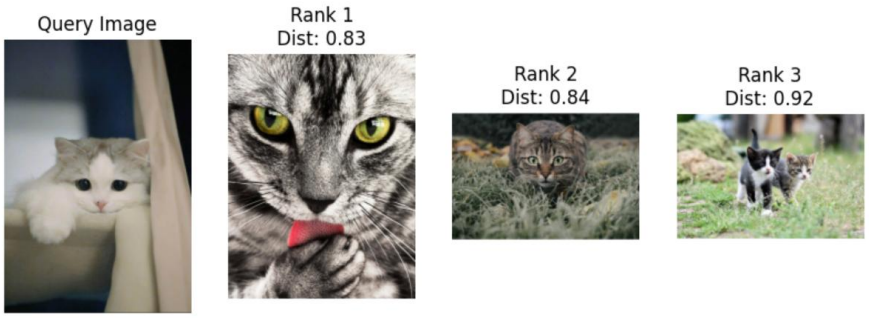


Query: 2.png



Query: 3.png

Query: 4.png



Query: 5.png

3. 选择 SIFT 特征，重复问题 2。

【SIFT 特征抽取代码的来源及说明贴这里】

```python
# 配置路径
query_dir = "5_query_images"  # 检索请求图像路径
db_dir = "45_database_images"  # 被检索图像路径
output_dir = "sift_results"  # 结果保存路径
os.makedirs(output_dir, exist_ok=True)

# 初始化SIFT检测器和匹配器
sift = cv2.SIFT_create()
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=False)  # 或使用FLANN匹配器

# 预提取数据库图像的SIFT特征
database_features = {}
for img_name in os.listdir(db_dir):
    img_path = os.path.join(db_dir, img_name)
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    kp, des = sift.detectAndCompute(img, None)
    if des is not None:
        database_features[img_name] = des  # 存储描述子
```

【SIFT 特征匹配代码的来源及说明贴这里】

```python
# 预提取数据库图像的SIFT特征
database_features = {}
for img_name in os.listdir(db_dir):
    img_path = os.path.join(db_dir, img_name)
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    kp, des = sift.detectAndCompute(img, None)
    if des is not None:
        database_features[img_name] = des  # 存储描述子

# 处理每个查询图像
for query_name in os.listdir(query_dir):
    query_path = os.path.join(query_dir, query_name)
    query_img = cv2.imread(query_path, cv2.IMREAD_GRAYSCALE)
    kp_query, des_query = sift.detectAndCompute(query_img, None)

    if des_query is None:
        print(f"查询图像 {query_name} 未检测到SIFT特征，跳过。")
        continue

    # 存储匹配结果（格式：{图像名: 匹配点数量}）
    matches_count = {}

    # 与所有数据库图像匹配
    for db_name, db_des in database_features.items():
        if db_des is None:
            continue
        # 使用k-NN匹配（k=2）并应用比率测试
        matches = bf.knnMatch(des_query, db_des, k=2)
        good_matches = []
        for m, n in matches:
            if m.distance < 0.7 * n.distance:  # 比率测试阈值
                good_matches.append(m)
        matches_count[db_name] = len(good_matches)  # 记录匹配点数量

    # 按匹配点数量排序，取前3名
    sorted_results = sorted(matches_count.items(), key=lambda x: -x[1])[:3]
```

【结果（检索请求+前 3 个结果）贴这里】

Query: 1.png

Query Image

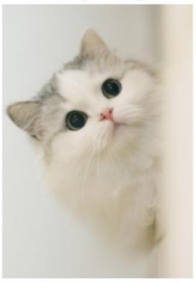
Rank 1
Matches: 47

Rank 2
Matches: 33

Rank 3
Matches: 21

Query: 2.png

Query Image



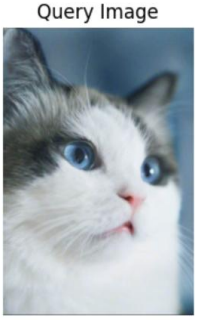
Rank 1
Matches: 6

Rank 2
Matches: 4

Rank 3
Matches: 3

Query: 3.png

Query Image



Rank 1
Matches: 8

Rank 2
Matches: 7

Rank 3
Matches: 7

4. 将问题 2 和问题 3 的结果进行比较和分析。

【比较和分析结果贴这里】

1. 原理差异

RGB：RGB 色彩模式是通过红（R）、绿（G）、蓝（B）三个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色。RGB 特征提取通常是统计图像的颜色直方图，以此来描述图像的整体颜色分布。

SIFT：尺度不变特征变换（SIFT）是一种局部特征提取算法。它在不同尺度空间上查找关键点，并计算出关键点的方向和描述子。这些描述子对图像的旋转、尺度缩放、亮度变化等具有不变性。

2. 相似度衡量

RGB：一般通过比较颜色直方图之间的距离（如欧氏距离、巴氏距离等）来衡量图像的相似度。颜色分布相似的图像，其直方图距离较小。

SIFT：通过匹配关键点的描述子来衡量图像的相似度。通常使用最近邻匹配（如暴力匹配或 FLANN 匹配），匹配到的关键点对越多，图像越相似。

3. 优缺点比较

RGB

优点

计算简单：颜色直方图的计算相对简单，计算速度快，对硬件要求低。

全局信息：能很好地反映图像的整体颜色分布，适用于对颜色敏感的场景，如颜色分类、图像检索等。

缺点

缺乏空间信息：只考虑颜色的分布，不考虑颜色的空间位置，对于颜色分布相似但内容不同的图像容易误判。

受光照影响大：光照变化会导致图像颜色发生改变，从而影响颜色直方图的准确性。

SIFT

优点

尺度、旋转不变性：能在不同尺度和旋转角度下检测到相同的关键点，对图像的几何变换具有很强的鲁棒性。

局部特征：关注图像的局部特征，对遮挡、背景变化等有较好的适应性。

缺点

计算复杂：SIFT 算法的计算量较大，处理速度相对较慢，对硬件性能要求较高。

全局信息不足：主要关注局部特征，可能忽略图像的整体信息。

4. 应用场景比较

RGB：适用于对颜色要求较高、对计算速度要求较快的场景，如基于颜色的图像检索、图像分类、色彩校正等。

SIFT：适用于需要处理图像几何变换、遮挡等复杂情况的场景，如物体识别、图像拼接、目标跟踪等。

5. 结果分析示例

RGB 结果：由于旋转可能会导致颜色分布略有变化，但整体颜色分布相似，RGB 可能会检索出一些颜色相似但物体不同的图像，存在较多误检。

SIFT 结果：SIFT 能检测到旋转后物体的关键点，并准确匹配到数据库中相同物体的图像，检索结果较为准确。