lab3-SIFT　实验报告

电院 2303 徐恺阳 523030910085

November 2024

1　实验概览

本实验内容如下：

1. 图像关键点的提取；
2. SIFT 描述子的计算；
3. 图像特征匹配。

SIFT 特征匹配算法可以处理两幅图像之间发生平移、旋转、仿射变换情况下的匹配问题，具有很强的匹配能力。

SIFT 算法的实质是在不同的尺度空间上查找关键点 (特征点)，并计算出关键点的方向。SIFT 所查找到的关键点是一些十分突出，不会因光照，仿射变换和噪音等因素而变化的点，如角点、边缘点、暗区的亮点及亮区的暗点等。

SIFT 算法主要步骤如下：

1. 检测尺度空间关键点
2. 精确定位关键点
3. 为每个关键点指定方向参数
4. 关键点描述子的生成

2　实验环境

本实验基于 Python 3.10，用到的库有 OpenCV、Numpy 和 Matplotlib。

3　练习一

3.1　题目描述

在 dataset 文件夹中的所有图片中搜索 target.jpg 图片所示物体，并绘制程序认为的好的匹配。

注意：高斯差分金字塔和特征点提取可用简化方法，SIFT 描述子需要自己计算。

3.2 解题思路

Step1: 构建尺度空间

构建尺度空间的方法有两种，分别是：

- 用高斯差分构建高斯金字塔；
- 直接缩放图像构建图像金字塔。

这里，我们选择较为简单的后者。代码如下：

```python
# 构建图像金字塔
def build_pyramid(image, levels=3):
    pyramid = [image]
    for i in range(1, levels):
        scaled_image = cv2.pyrDown(pyramid[-1])  # 图像降采样
        pyramid.append(scaled_image)
    return pyramid
```

图 1: 构建图像金字塔

这里我们选用 OpenCV 自带的 **pyrDown 函数**进行图像的缩放，理由如下：

- 缩放前会对图像进行高斯模糊，平滑图像，以减少降采样带来的混叠效应；
- 只需传入图像，不需要手动计算目标尺寸。

Step2: 提取关键点

提取关键点的方法也有两种，分别是：

- 在高斯金字塔中提取**极值点**再精确定位和消除边缘响应，作为关键点；
- 在图像金字塔提取**角点**（灰度变化最显著的点），作为关键点。

这里，我们选择较为简单的后者。

OpenCV 提供了两个角点检测的函数，分别为 cornerHarris 函数和 goodFeaturesToTrack 函数。这里我们选择基于 $Shi-Tomasi$ 算法的 **goodFeaturesToTrack 函数**，理由如下：

- 可以直接选择质量最好的角点，减少无关点；
- 可以通过参数控制角点数量、质量和分布间距；
- 自带非极大值抑制，避免过于密集的角点。

代码如下：

```python
# Harris角点检测
def detect_keypoints(image, max_corners=100, quality_level=0.01, min_distance=10):
    corners = cv2.goodFeaturesToTrack(
        image,
        maxCorners=max_corners,
        qualityLevel=quality_level,
        minDistance=min_distance
    )

    # 转换角点为 OpenCV 的 KeyPoint 对象
    keypoints = [cv2.KeyPoint(pt[0][0], pt[0][1], 1) for pt in corners]
    return keypoints
```

图 2: Harris 角检测

Step3: 计算 SIFT 描述子

SIFT 描述子计算的原理如下：

1. 梯度幅值与方向计算

在某关键点的 $m \times m$ 邻域内，计算梯度幅值 $m(x, y)$ 和梯度方向 $\theta(x, y)$。

- 梯度幅值公式:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

- 梯度方向公式:

$$\theta(x, y) = \arctan\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right)$$

2. 主方向确定

将梯度方向分为 36 个 bins，统计每个 bin 的权值（权值为梯度幅值）。选取权值最大的方向为主方向。

3. 描述子生成

以关键点为中心，建立 16×16 的邻域，将其划分为 16 个 4×4 的子区域。

每个子区域计算 8 个方向的梯度直方图，最终得到 $4 \times 4 \times 8 = 128$ 维的描述子。

4. 插值与归一化

- 使用双线性插值计算梯度方向对应的直方图权值。
- 对 128 维的描述子进行归一化处理。

代码过长，见 HW_3_1.py。

Step4: 图像特征匹配

注意： 这里曾尝试用 BFMatcher（暴力匹配器）的普通特征匹配进行特征匹配，但是匹配出来的关键点几幅图区分度太小，效果不佳，故这里采用另一种特征匹配方法——KNN 匹配，代码如下：

```python
# KNN特征匹配
matches = bf.knnMatch(target_des, des, k=2)

# 筛选匹配
good_matches = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        good_matches.append(m)

# 绘制匹配结果
result_img = cv2.drawMatches(target_img, target_kp, img, kp, good_matches, None,
                             flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```

图 3: KNN 特征匹配

3.3 代码运行结果

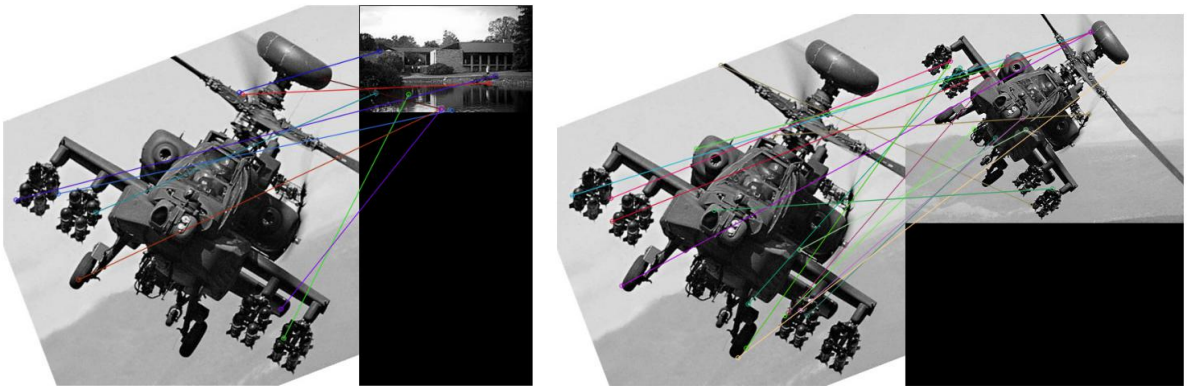

图 4: 部分匹配结果

3.4 分析与思考

可以看出，虽然 KNN 特征匹配下相匹配的关键点较少，但是几幅图匹配结果区分度很大，效果较好。

并且，相匹配的关键点大多为黑白交界处的点，因为此处的梯度变化较大，更容易成为关键点。因此，SIFT 特征匹配对于色彩对比鲜明、形状分布较多的图片可能有较好的匹配效果。

4 练习二

4.1 题目描述

利用 OpenCV 自带的 SIFT 相关函数，在 dataset 文件夹中的所有图片中搜索 target.jpg 图片所示物体，并绘制程序认为的好的匹配。

4.2 解题思路

Step1: 实例化 SIFT 对象

利用 OpenCV 自带的 SIFT_create 函数，创建 SIFT 对象。

Step2: 提取目标图像和数据集图像的关键点和描述子

利用 sift.detectAndCompute 函数，代码如下：

```
# 提取目标图像的关键点和描述子
kp_target, des_target = sift.detectAndCompute(target_img, None)

# 提取数据集图像的关键点和描述子
kp_dataset, des_dataset = sift.detectAndCompute(dataset_img, None)
```

图 5: 提取目标图像和数据集图像的关键点和描述子

Step3: 特征匹配，并绘制结果

注意： 这里曾尝试用 BFMatcher（暴力匹配器）的普通匹配进行特征匹配，但是匹配出来的关键点个数太少，效果不佳，故这里采用另一种特征匹配方法——KNN 匹配，代码如下：

```python
# 通过比值测试筛选好的匹配
good_matches = []
for m, n in knn_matches:
    if m.distance < 0.75 * n.distance:  # 比值测试，保留好的匹配
        good_matches.append(m)

# 绘制好的匹配结果
img_matches = cv2.drawMatches(target_img, kp_target, dataset_img, kp_dataset,
```

图 6: KNN 特征匹配

4.3 代码运行结果

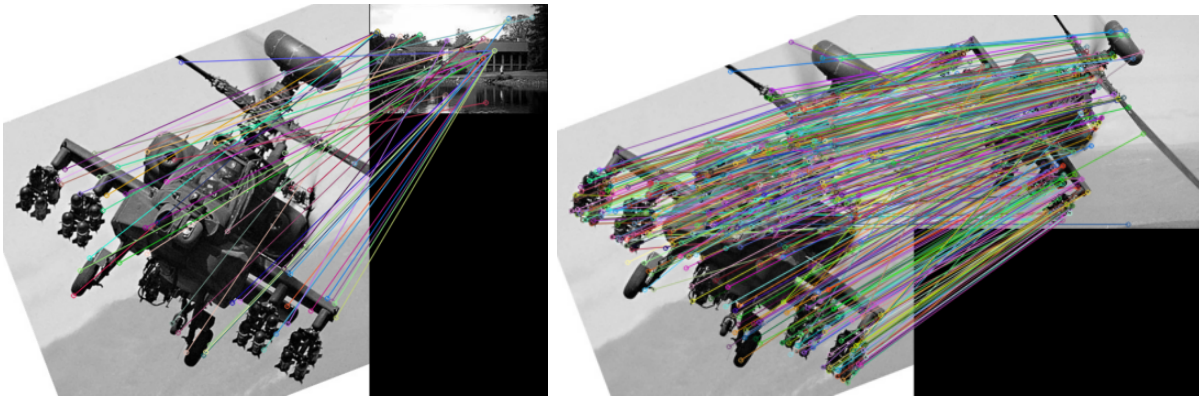

图 7: 部分匹配结果

4.4 分析与思考

与手搓的 SIFT 算法相比，OpenCV 自带的 SIFT 函数在同样的匹配方式（KNN 匹配）下，相匹配的关键点远远多于前者，可能是因为其并不是通过简单的图像金字塔中检测角点的方法来提取关键点，而是通过更严格的高斯金字塔中检测极值点的方法，所以对关键点的提取更加准确。

同时，OpenCV 自带的 SIFT 函数的匹配结果在关键点非常非常多的情况下，仍然具有很好的区分度与辨识度，我们可以很轻松的看出目标图像与数据集图像匹配结果的好坏。

5 实验感想

本次实验，我通过实现 SIFT 算法进行图像特征点匹配，深入理解了图像关键点的提取、描述子计算和匹配的过程。在不借助 OpenCV 中的 SIFT 实例的情况下，我一步一步计算 SIFT 描述子，并据此绘制了目标图像与数据集图像的特征匹配结果。

这次实验提升了我在图像处理和计算机视觉中的实际操作能力，并为解决更复杂的匹配问题积累了经验。