计算机视觉实践——练习 5_图像视差匹配报告

目录

1 实验目的

- 通过立体匹配计算两张图片的视差图。

2 实验原理

使用 Sum of Absolute Differences（SAD）计算视差图。

2.1 原理

该算法通过比较立体图像对（一般为左右图）中的光照亮度差异来寻找匹配点对，进而计算视差。

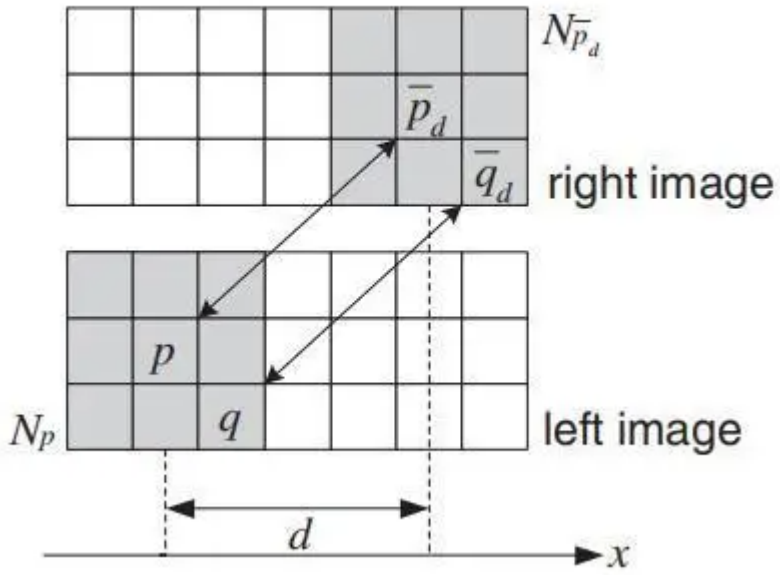

图 1 SAD 匹配

2.1.2 执行步骤

1. 构造一个小窗口，类似于卷积核。
2. 用窗口覆盖左边的图像，选择出窗口覆盖区域内的所有像素点。
3. 同样用窗口覆盖右边的图像并选择出覆盖区域的像素点。
4. 左边覆盖区域减去右边覆盖区域，并求出所有像素点灰度差的绝对值之和。
5. 移动右边图像的窗口，重复（3）-（4）的处理（这里有个搜索范围,超过这个范围跳出）。
6. 找到这个范围内 SAD 值最小的窗口,即找到了左图锚点的最佳匹配的像素块。

3 实验

3.1 实验细节

本次实验使用的语言为 Python，使用的库有 opencv、numpy、matplotlib。

3.2 实验结果

输入图如下：

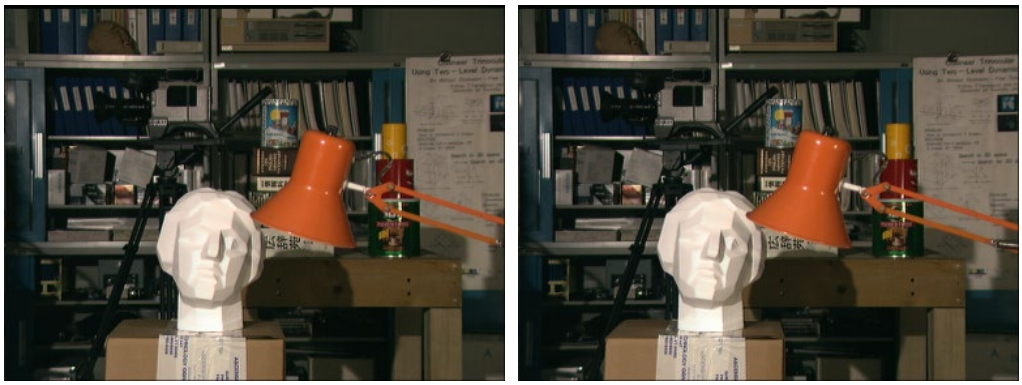

图 2 输入图

视差图如下：

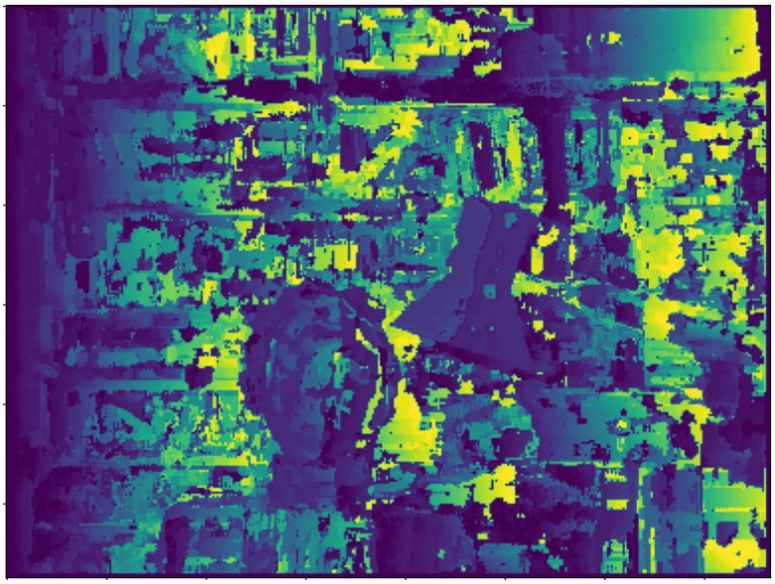

图 3 视差图

3.3 实验分析

SAD 算法虽然简单高效，但是容易受光照、曝光等噪声以及滑动窗口等参数设置影响，视差图的效果往往不是特别鲁棒。

附录——代码展示

仅展示主要代码。

```
def stereo_matching(img_left, img_right, window_size=5, disparity_range=100):
    # 转换为灰度图像
    if len(img_left.shape) > 2:
        img_left = cv2.cvtColor(img_left, cv2.COLOR_BGR2GRAY)
    if len(img_right.shape) > 2:
        img_right = cv2.cvtColor(img_right, cv2.COLOR_BGR2GRAY)
    height, width = img_left.shape

    # 定义一个空的视差图像
    disparity = np.zeros_like(img_left)

    # 计算每个像素点的视差
    half_window = window_size // 2
    for y in range(half_window, height - half_window):
        for x in range(half_window, width - half_window):
            # 选择左图像中的一个窗口
            window_left = img_left[y - half_window : y + half_window + 1, x - half_window : x + half_window + 1]

            # 初始化最小匹配成本和最佳视差
            min_cost = np.inf
            best_disparity = 0

            # 在一定的视差范围内搜索右图像
            for d in range(disparity_range):
                # 确定右图像中对应窗口的位置
                x_right = x - d
                if x_right < half_window:
                    break

                # 选择右图像中的一个窗口
                window_right = img_right[y - half_window : y + half_window + 1, x_right - half_window : x_right + half_window + 1]

                # 计算匹配成本
                cost = np.sum(np.abs(window_left - window_right))

                # 如果匹配成本更小，则更新最小匹配成本和最佳视差
                if cost < min_cost:
                    min_cost = cost
```

```
                # 如果匹配成本更小，则更新最小匹配成本和最佳视差
                if cost < min_cost:
                    min_cost = cost
                    best_disparity = d

            # 将最佳视差存储在视差图像中
            disparity[y, x] = best_disparity

    return disparity
```