哈尔滨工业大学（深圳）

《计算机视觉》课程设计报告

题目：基于 Xfeat 特征匹配的多图像融合拼接算法实现

学　　生：陈宇琛

指导教师：吴晓军教授

学　　号：24S153005

院　（系）：机器人与先进制造学院

专　　业：控制工程

☐博士研究生　　☑ 硕士研究生

哈尔滨工业大学研究生院

二〇二四年十二月

1 引言

图像拼接技术作为计算机视觉和图像处理领域的重要研究方向，旨在将多幅具有重叠区域的图像无缝合成为一幅整体连贯的全景图像或更大视野的复合图像。随着虚拟现实、全景摄影、遥感测绘及医学影像等应用场景的快速发展，对高效、精确的多图像拼接算法需求日益增加。然而，随着待拼接图像数量的增加，传统图像拼接方法在计算复杂度和处理速度上面临显著挑战，尤其是在特征点检测与匹配阶段。

本项目针对 15 张具有特定相对位置关系的图像，设计了一种高效的图像拼接算法。通过充分利用图像之间的邻接关系，如图 1 所示，算法能够在特征点检测与匹配过程中实现加速，从而显著提升整体处理速度。项目将结合传统方法与深度学习方法的优势，探索在多图像拼接任务中的应用与优化。

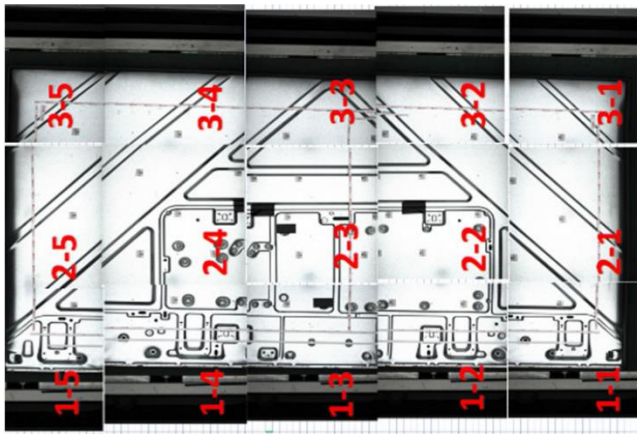


图 1

传统图像拼接方法通常依赖于 SIFT、SURF 等特征点检测与描述算法，通过匹配特征点来估计图像间的几何变换关系，再进行图像配准与融合。这些方法在精度上具有较好的表现，但在处理大规模图像时，计算量和时间成本较高。近年来，深度学习技术的兴起为图像拼接提供了新的解决思路。基于深度学习的方法通过训练卷积神经网络，可以在特征提取和匹配阶段实现更高的效率和鲁棒性，尤其是在处理复杂场景和遮挡情况下表现优异。

本项目将综合利用图像的邻接关系，优化特征点检测与匹配流程，结合传统与深度学习方法的优势，设计出一种适用于快速多图像拼接的算法。具体而言，算法将首先根据图像的相对位置关系确定图像之间的邻接关系，从而限制特征点匹配的搜索范围，减少计算量。同时，采用高效的特征点检测与描述方法，提升匹配的准确性与速度。最终，通过优化的图像配准与融合策略，实现 15 张图像的高质量拼接。

通过本项目的研究与实现，期望在多图像拼接领域提出一种高效、快速且精确的算法解决方案，满足实际应用中对大规模图像拼接的需求，为相关领域的发展提供技术支持。

2 方法论

2.1 扫描图拼接定义

多图像拼接技术可以根据应用场景的不同，分为**全景图拼接**和**扫描图拼接**两大类。尽管两者都旨在将多幅图像合成为一幅完整的图像，但在处理方法和技术细节上存在显著差异，本文主要介绍扫描图拼接的实现。

扫描图拼接主要用于将多个扫描得到的图像合成为一幅完整的目标图像，常见于工业检测、条码识别和物体表面扫描等领域。

2.2 扫描图拼接的挑战

在实际应用中，扫描图拼接面临诸多挑战，包括但不限于：

1. 特征匹配的准确性：不同图像间的重叠区域可能较小，导致特征点分布不均匀，特别是在图像边界处集中；高分辨率图像中，特征点数量庞大，容易产生错配。
2. 几何变换的计算复杂度：高效且准确地估计图像之间的几何变换（如仿射变换、单应性变换）对于拼接质量至关重要；复杂的变换计算在处理大量图像时会增加计算资源的消耗。
3. 图像融合的无缝性：拼接后的图像需要在颜色、亮度和细节上保持一致，避免出现明显的拼接痕迹；传统的图像融合方法在处理曝光差异和细节保留方面效果有限。
4. 资源利用与处理效率：高分辨率图像拼接对计算资源要求高，尤其在实时应用场景中，如何在有限的资源下实现高效处理是一个重要问题。

2.3 图像预处理

2.3.1 图像锐化

图像锐化是一种图像增强技术，旨在提高图像的边缘和细节，使图像看起来更加清晰和鲜明。通过增强图像中的高频信息，锐化能够突出物体的边界和纹理细节，改善视觉效果

本文中使用的锐化核为：

$$\text{kernel} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

2.3.2 对比度增强

本文采用对比度限制自适应直方图均衡化方法(CLAHE)，这是一种改进的直方图均衡化方法，旨在增强图像的对比度，同时避免过度增强导致的噪声放大和图像失真,如图 2 所示。CLAHE 通过将图像分割为多个小区域（称为“块”或“窗口”），对每个区域分别进行直方图均衡化，并限制直方图的对比度增强，最终将各区域融合，生成对比度均衡的图像。

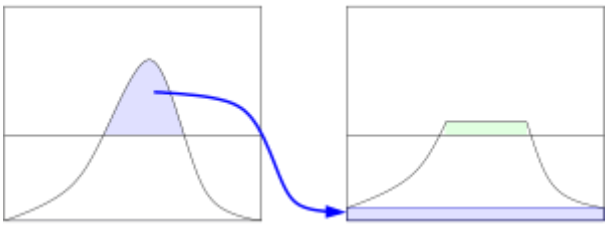


图 2

2.4 特征点检测与匹配

2.4.1 传统的特征点匹配方法

常用的传统特征检测算法包括 SIFT（尺度不变特征变换）、SURF（加速稳健特征）和 ORB（Oriented FAST and Rotated BRIEF）等。

优点：成熟稳定，经过多年研究和应用，算法稳定性和可靠性较高；计算资源需求相对较低，适用于资源有限的设备。

缺点：匹配准确性受限，在处理特征点分布不均或重叠区域有限的情况下，匹配效果不佳；对光照和视角变化敏感，在光照变化或视角剧烈变化的情况下，特征匹配效果下降。

2.4.2 基于深度学习的特征点匹配方法

随着深度学习技术的发展，基于神经网络的特征检测与匹配方法逐渐应用于多图像拼接领域。XFeat（Accelerated Features）作为一种轻量级且高效的特征匹配架构，通过重新设计卷积神经网络的基本结构，实现了在资源受限设备上的快速且鲁棒的特征匹配。

该方法提取了一个关键点热图 **K**、一个紧凑的 64 维密集描述符图 **F** 以及一个可靠性热图 **R**，如图 3 所示。通过早期下采样和浅层卷积，随后在后续编码器中使用更深的卷积以增强鲁棒性，XFeat 实现了无与伦比的速度。与典型方法不同，XFeat 将关键点检测分离到一个独立的分支，使用在 8×8 张量块转换图像上的 1×1 卷积进行快速处理。这使得 XFeat 成为当前少数能够将检测与描述解耦并独立处理的学习方法之一。

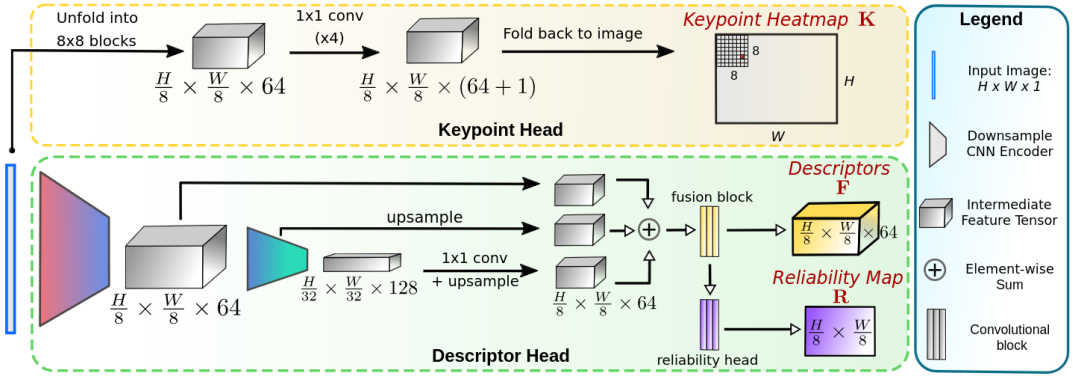


图 3

优点：高效性，相比传统深度学习方法，XFeat 的特征提取和匹配速度提升了约 5 倍；资源利用优化，无需依赖 GPU 资源，适用于 CPU 密集型的应用场景；高匹配准确性，在姿态估计和视觉定位任务中表现出与或优于现有深度学习特征的准确性。

缺点：依赖预训练模型，需要大量的数据进行训练，以确保模型的鲁棒性

2.5 几何变换

2.5.1 透视变换

透视变换（又称单应变换）是一种能够处理图像中透视畸变的非线性变换。它通过调整图像的视角，使得不同视角下的图像能够精确对齐。透视变换可以用以下矩阵表示：

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

其中，(x, y)为原始图像的坐标，(x', y')为变换后的坐标，w'为缩放因子。通过归一化，可得到最终的坐标：

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$
$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

然而，观察到在本任务中相机拍摄视角几乎在同一平面，因此并不需要处理图像的畸变，故采用仿射变换，使算法更高效准确。

2.5.2 仿射变换

仿射变换是一种线性变换，能够保持图像中的直线和平行关系，但不保持角度和长度。它通过旋转、缩放、剪切和平移等操作对图像进行变换。仿射变换可以用以下矩阵表示：

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

其中， (x, y)为原始图像的坐标，(x', y')为变换后的坐标，a, b, c, d 为线性变换参数，t_x，t_y为平移参数。

2.6 增益补偿

增益补偿（Gain Compensation）是一种图像处理技术，旨在调整多幅图像之间的亮度和颜色差异，以实现拼接区域的亮度和色彩一致性。

本文定义一个全局误差函数来描述不同图片之间亮度与颜色的不一致性：

$$e = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij}\left(\frac{(g_i\bar{I}_{ij} - g_j\bar{I}_{ji})^2}{\sigma_N^2} + \frac{(1-g_i)^2}{\sigma_g^2}\right)$$

其中，N_{ij}是图像 i 在与图像 j 重叠区域的像素数，$\bar{I}_{ij}$是图像 i 在与图像 j 重叠区域的平均像素值，g_i表示图像 i 的增益，σ_N和σ_g分别是标准化强度误差与增益的标准差，本文中分别取 10 和 0.1。

对上式对g_i求一阶导并令为 0 可得:

$$g_j\left(\frac{2}{\sigma_N^2}\sum_{i=1}^{n} N_{ji}\bar{I}_{ji}^2 + \frac{\sum_{i=1}^{n} N_{ki}}{\sigma_g^2}\right) - \frac{2}{\sigma_N^2}\sum_{i=1}^{n} N_{ki}\bar{I}_{ki}\bar{I}_{ik}g_i - \frac{\sum_{i=1}^{n} N_{ki}}{\sigma_g^2} = 0$$

最后可通过简单的线性求解器求解得到各图像的增益。

2.7 多频段融合

多频段融合（Multi-Band Blending）是一种高级图像融合技术，旨在将多幅图像在不同频率层次上进行融合，以实现无缝的拼接效果。通过将图像分解为多个频带（通常使用高斯金字塔和拉普拉斯金字塔），多频段融合能够在低频层次处理颜色和亮度的一致性，在高频层次处理细节和纹理的保留，从而有效减少拼接痕迹，提升最终全景图像的视觉质量。

2.7.1 构建高斯金字塔

高斯金字塔是一种多分辨率表示方法，通过逐层对图像进行高斯滤波和下采样，生成一系列分辨率逐渐降低的图像层次。每一层的图像代表原始图像在不同尺度上的低频信息。

$$G_{l+1} = \text{Gaussian}(G_l)$$

其中，G_l表示第l层的高斯金字塔图像。

2.7.2 构建拉普拉斯金字塔

拉普拉斯金字塔通过高斯金字塔构建，每一层的图像由当前高斯层减去上采样后下一层的高斯图像得到，代表图像的高频细节。

$$L_l = G_l - \text{Expand}(G_{l+1})$$

其中，L_l表示第l层的高斯金字塔图像。

2.7.3 在各频带上进行图像融合

在每一层的拉普拉斯金字塔上，使用一个对应的高斯金字塔掩膜进行加权融合：

$$B_l = M_l \cdot L_{1,l} + (1 - M_l) \cdot L_{2,l}$$

其中：B_l为第 1 层的融合拉普拉斯金字塔图像,M_l为第 1 层的高斯金字塔掩膜,$L_{1,l}$和$L_{2,l}$分别为两幅待融合图像的第l层拉普拉斯金字塔图像

2.7.4 重建融合后的图像

从最底层开始，逐层上采样并与当前层的融合拉普拉斯金字塔图像相加，最终重建出融合后的全景图像：

$$I = B_0 + \text{Expand}(B_1 + \text{Expand}(B_2 + \ldots))$$

3 算法设计与实现

3.1 环境

3.1.1 依赖

- python 3.8
- pytorch 1.10.1
- numpy 1.24.4
- opencv-python 4.10
- matplotlib 3.7.5

3.1.2 实验机

该算法在 Ubuntu 20.04, Intel Core i9, NVIDIA Geforce RTX 4060(8G)上实现。

3.2 图像编号

将 15 张图片编号以方便编程实现，如下表所示：

编号	图像名称	编号	图像名称	编号	图像名称	编号	图像名称	编号	图像名称
0	image_1_1.jpg	1	image_1_2.jpg	2	image_1_3.jpg	3	image_1_4.jpg	4	image_1_5.jpg
5	image_2_1.jpg	6	image_2_2.jpg	7	image_2_3.jpg	8	image_2_4.jpg	9	image_2_5.jpg
10	image_3_1.jpg	11	image_3_2.jpg	12	image_3_3.jpg	13	image_3_4.jpg	14	image_3_5.jpg

位置关系如图 4 所示。

4	9	14
3	8	13
2	7	12
1	6	11
0	5	10

图 4

3.3 层级构造

以编号 7 的图像，即 image_2_3.jpg 为基准图像，并令为第 0 层。

后有第 1 层：图像 2，6，8，12；第 2 层：1，3，11，13；第三层：5，9；第四层：0，4，10，14。如图 5 所示。

4	3	4
2	1	2
1	0	1
2	1	2
4	3	4

图 5

3.4 特征匹配

3.4.1 从中心逐层匹配算法设计

从第 1 层开始逐层遍历，将该层的每张图片都与前一层中存在邻接关系的图片进行 Xfeat 特征点匹配，如第 1 层的 2 与第 0 层的 7 匹配，第 2 层的 3 与第 1 层的 2 和 8 匹配，第 3 层的 9 与第 2 层的 8 匹配，第 4 层的 14 与第三层的 9 和 12 匹配，以此类推，共进行 22 次特征匹配。

算法时间复杂度为：$O(n)$

3.4.2 利用邻接信息控制匹配区域

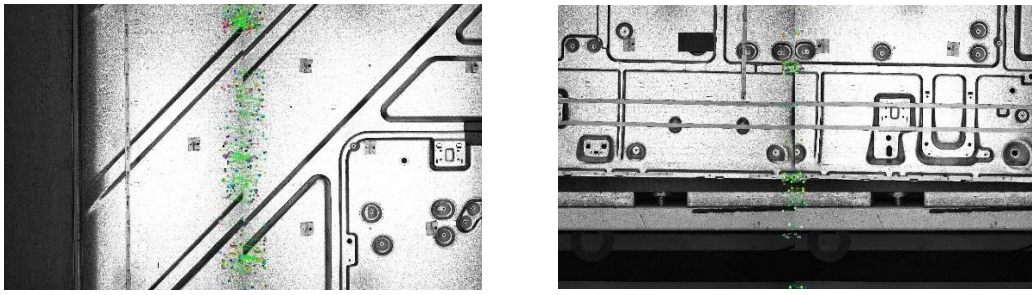
观察到本任务中图片重叠区域较少，因此可以基于邻接的位置信息来限制匹配的区域，

减少计算负担并减少误匹配。例如在 2 与 7 的匹配中，仅需考虑 2 的右侧边界一定比例区域与 7 左侧边界相应比例区域的特征匹配。

3.4.3 利用坐标差值筛选匹配点

观察到本任务中原始图片的规则性，可以发现特征点的横坐标或纵坐标一定不会偏差特别大，因此可设定阈值以剔除误匹配点。例如在 6 与 7 的匹配中，若存在有匹配点对的 x 坐标差距超过某一设定阈值，则可将其视为误匹配点并剔除。

3.4.4 匹配结果



(a) (b)

图 6

3.5 仿射变换矩阵计算

3.5.1 逐层计算与储存

与特征匹配同一层循环中，紧接着特征点匹配逐层进行仿射变换的计算，然后用以这对变换图片的索引构成的元组为键储存在字典中，如图 7 展示的程序段所示。

```
kpts1, kpts2 = match_image_using_xfeat(center_index, idx, ratio, top_k, threshold)
transformations[(idx, center_index)] = get_trans(kpts1, kpts2, warp_mode)
```

图 7

3.5.2 多路径加权平均

例如计算图片 3 到基准图片 7 的仿射变换时，首先计算 3 到 2 以及 3 到 8 的仿射变换矩阵，然后利用先前已有的 2 到 7 和 8 到 7 的仿射变换矩阵，通过扩行为 3*3 矩阵后相乘再取前两行，可得到 3→2→7 以及 3→8→7 两条变换路径获得的仿射变换矩阵，如图 8 所示，蓝色为路径 1，橙色为路径 2。之后以 3 和 2 以及 3 和 8 特征匹配点的维数为权重，将上述两个仿射变换矩阵加权平均，得到最终 3 到 7 的仿射变换矩阵。其他可依此类推。

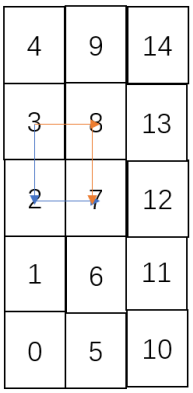


图 8

3.6 最终画布尺寸计算

利用求得的仿射变换矩阵对每幅图像仅对上下左右四个角点进行变换，然后将 15 幅图像角点坐标的最大值减去最小值，可得最终画布的宽度与高度。如图 9 所示。

```python
def compute_output_size(images, transformations, base_index=7):
    """
    计算所有图像变换后所需的画布大小，确保所有图像能够在同一个画布上完整显示。
    :param images: 图像列表
    :param transformations: 仿射变换字典，存储从其他图像到基准图像7的变换矩阵
    :param base_index: 基准图像的索引，默认为7
    :return: 输出图像的宽度和高度
    """
    h,w=images[base_index].shape[:2]
    min_x, min_y, max_x, max_y = 0, 0, w, h
    for idx, img in enumerate(images):
        if idx == base_index:
            continue
        # 获取该图像的变换矩阵
        trans = transformations.get((idx, base_index))
        if trans is not None:
            h, w = img.shape[:2]
            # 对图像的4个角点进行变换，计算新的边界框
            corners = np.array([[0, 0], [0, h - 1], [w - 1, 0], [w - 1, h - 1]], dtype=np.float32)
            transformed_corners = cv2.transform(np.array([corners]), trans)[0]
            min_x = min(min_x, transformed_corners[:, 0].min())
            min_y = min(min_y, transformed_corners[:, 1].min())
            max_x = max(max_x, transformed_corners[:, 0].max())
            max_y = max(max_y, transformed_corners[:, 1].max())

    # 输出图像的宽度和高度
    return int(max_x - min_x), int(max_y - min_y),-int(min_x),-int(min_y)
```

图 9

3.7 增益补偿

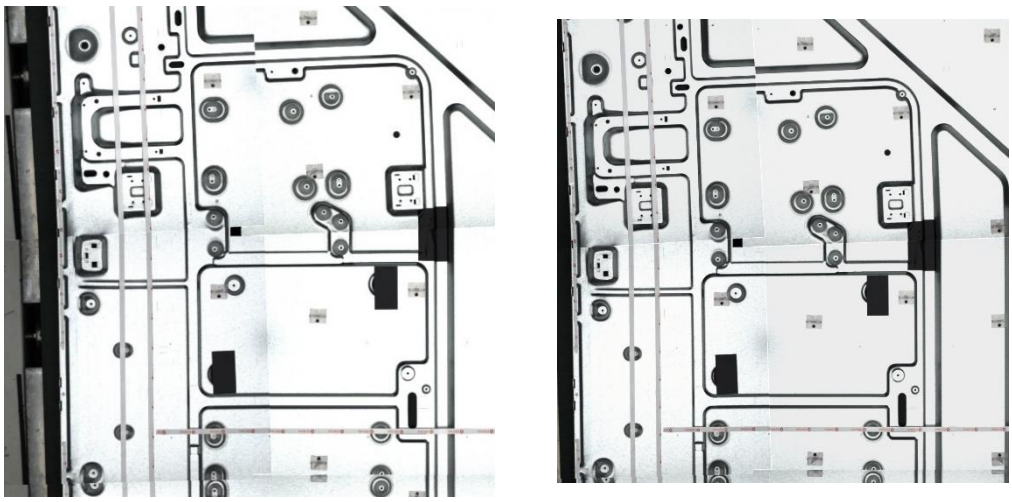
3.7.1 重叠区域像素计算

同样采用逐层计算的方法，对每一对有重叠区域的图片，计算重叠区域像素总数、图像 a 在重叠区域的像素值和、图像 b 在重叠区域的像素值和，如图 10 所示。

```python
for k, image in enumerate(images):
    coefs = [np.zeros(3) for _ in range(len(images))]
    result = np.zeros(3)
    for (a, b), match in pair_matches.items():
        overlap_area = match["overlap_area"]
        Iab = match["Iab"]  # Shape: (3,)
        Iba = match["Iba"]  # Shape: (3,)
        if a == k:
            coefs[k] +=  overlap_area* (
                (2 * Iab ** 2 / sigma_n ** 2) + (1 / sigma_g ** 2)
            )

            coefs[b] -= (
                (2 / sigma_n ** 2) * overlap_area * Iab * Iba
            )

            result += overlap_area / sigma_g ** 2
        elif b == k:
            coefs[k] += overlap_area * (
                (2 * Iba ** 2 / sigma_n ** 2) + (1 / sigma_g ** 2)
            )

            coefs[a] -= (
                (2 / sigma_n ** 2) * overlap_area * Iab *Iba
            )

            result += overlap_area / sigma_g ** 2
    coefficients.append(coefs)
    results.append(result)
coefficients = np.array(coefficients)
results = np.array(results)
gains = np.zeros_like(results)
```

图 10

3.7.2 增益补偿效果



(a)补偿前　　(b)补偿后

图 11

3.8 多频段融合拼接

3.8.1 仿射变换应用与掩膜构造

同样采用逐层拼接顺序，如图 12 所示，将前一次的结果作为 tar_img，将当前层的图片作为 src_img，对当前层的图片构造掩膜 mask，之后进行多频段融合。

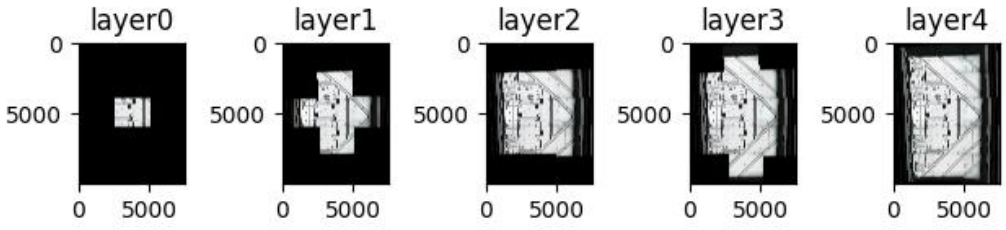


图 12

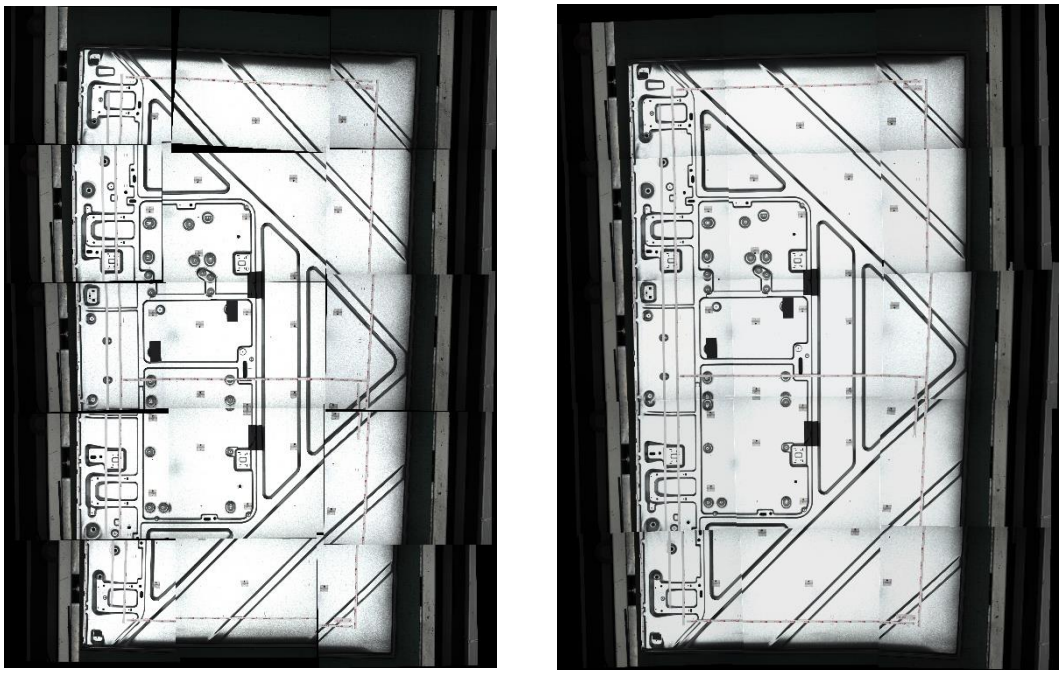
3.8.2 多频段融合实现

对 tar_img, src_img, mask 构造高斯金字塔，对 tar_img, src_img 构造拉普拉斯金字塔，然后逐层按 mask 权重融合，最后重建形成结果图像。如图 13 所示。

```
def pyramid_blend(dst_img,src_img,mask,scale):

    gpA = cv_pyramid(dst_img.copy(), scale)
    gpB = cv_pyramid(src_img.copy(), scale)
    gpM = cv_pyramid(mask.copy(), scale)
    lpA = cv_laplacian(gpA, scale)
    lpB = cv_laplacian(gpB, scale)
    blended_pyramid = cv_multiresolution_blend(gpM, lpA, lpB)
    blended_image = cv_reconstruct_laplacian(blended_pyramid)
    return blended_image
```

图 13

3.8.3 融合效果



(a)融合前　　(b)融合后

图 14

3.9 运行程序

输入命令：

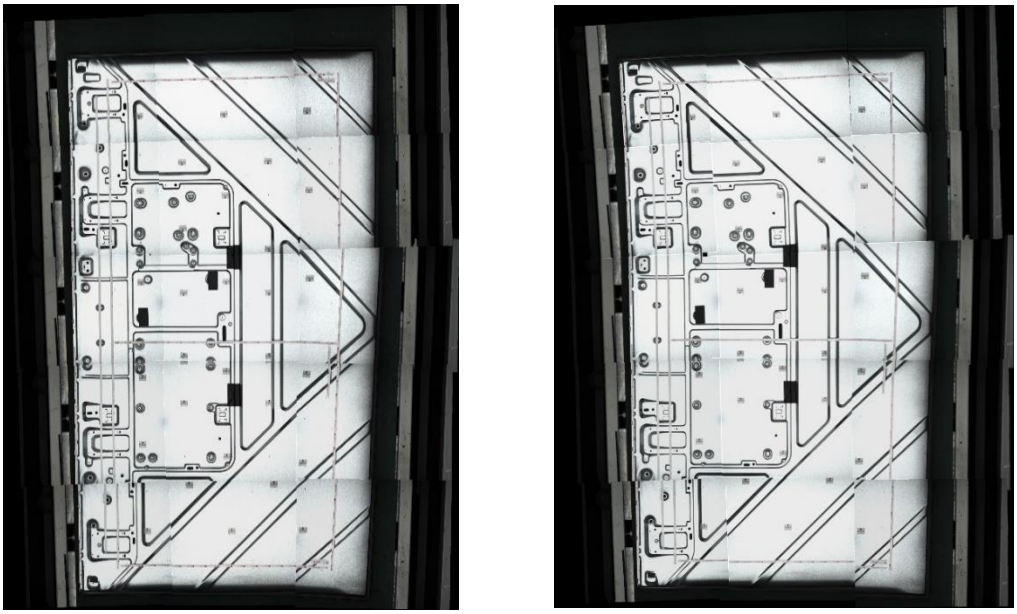
- conda activate xfeat
- python main.py

运行结果如图 15 所示。

```
(base) yc-chen@ycchen-Legion-Y9000P-IRX9:~/PycharmProjects/CV/project3$ conda activate xfeat
(xfeat) yc-chen@ycchen-Legion-Y9000P-IRX9:~/PycharmProjects/CV/project3$ python main.py
loading weights from: /home/yc-chen/PycharmProjects/CV/project3/modules/../weights/xfeat.pt
num of matched points: 181
num of matched points: 101
num of matched points: 52
num of matched points: 43
num of matched points: 112
num of matched points: 221
num of matched points: 177
num of matched points: 217
num of matched points: 47
num of matched points: 126
num of matched points: 273
num of matched points: 70
num of matched points: 17
num of matched points: 408
num of matched points: 62
num of matched points: 883
num of matched points: 368
num of matched points: 198
num of matched points: 76
num of matched points: 202
num of matched points: 91
num of matched points: 278
gain for image0 is :[0.88245078 0.88472941 0.87861517]
gain for image1 is :[0.89134577 0.89176022 0.88463681]
gain for image2 is :[0.939742   0.94608172 0.94678847]
gain for image3 is :[0.9302483  0.93181981 0.92930864]
gain for image4 is :[0.93992534 0.94086248 0.94271893]
gain for image5 is :[0.92341276 0.92105319 0.9248734 ]
gain for image6 is :[0.92430677 0.92127482 0.92219238]
gain for image7 is :[0.9376078  0.93334116 0.93108652]
gain for image8 is :[0.95135739 0.94721655 0.94815781]
gain for image9 is :[0.96970787 0.96536625 0.96625153]
gain for image10 is :[0.8870141  0.88957818 0.89101819]
gain for image11 is :[0.91476728 0.91652146 0.9160686 ]
gain for image12 is :[0.98786391 0.99278212 0.9943978 ]
gain for image13 is :[0.90898693 0.90967485 0.91172604]
gain for image14 is :[0.99756653 0.99984044 1.        ]
(9892, 7442, 3) - (9892, 7442, 3)
(9892, 7442, 3) - (9892, 7442, 3)
(9892, 7442, 3) - (9892, 7442, 3)
(9892, 7442, 3) - (9892, 7442, 3)
Stitching complete, saved as /results/final_image_compensation.jpg
running time: 32.0754609079712 s
(xfeat) yc-chen@ycchen-Legion-Y9000P-IRX9:~/PycharmProjects/CV/project3$
```

图 15

3.10 实验结果



(a)两层融合　　(b)三层融合

图 16

4 总结与展望

4.1 总结

本项目结合了深度学习方法(XFeat)与传统计算机视觉方法，通过自设计的算法实现了15 张高分辨率图片的高质量扫描拼接，具有快速、高效、可扩展性强、鲁棒性好的特点。

4.2 遇到的困难与解决办法

4.2.1 误匹配特征点

- 困难：本任务与全景图拼接不同的是，重叠区域极少，仅占 10%不到，导致直接进行特征点匹配会产生大量误匹配点。
- 解决办法：仅对感兴趣区域匹配；利用坐标信息筛选。

4.2.2 透视变换效果差

- 困难：同样地，由于重叠区域少，同时本任务中几乎所有图片都位于同一平面中，采用透射变换会对图片产生畸变，拼接效果很差
- 解决办法：改用仿射变换，所需特征点相对更少，同时能保留线的平行特征。

4.2.3 无监督深度学习图像拼接

- 困难： 复现无监督深度学习拼接项目时，效果差，同时由于图片分辨率较高，出现爆显存的问题。
- 解决办法：改用传统方法，或后续可根据本任务构造特定的数据集训练提高效果。

4.3 展望

✧ 可利用 GPU 加速并实现图像的并行处理提高运算速度

✧ 在适配于本任务的训练集上训练可提高 Xfeat 特征点匹配效果

参考文献

[1] G. Potje, F. Cadar, A. Araujo, R. Martins, and E. R. Nascimento, “XFeat: Accelerated Features for Lightweight Image Matching,” in *2024 IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*, 2024.

[2] M. Brown and D. G. Lowe, “Automatic Panoramic Image Stitching using Invariant Features,” *International Journal of Computer Vision*, vol. 59, no. 1, pp. 2–18, 2007.

[3] L. Nie, C. Lin, K. Liao, S. Liu, and Y. Zhao, “Parallax-Tolerant Unsupervised Deep Image Stitching,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 7399–7408.