上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

计算机视觉课程大作业报告

学院 电子信息与电气工程学院

班级 F2103001

学号 521030910030

姓名 周恂初

2024 年 5 月 12 日

摘　要

本次大作业，我们基于 SLAM 技术实现了一个简单的三维重建。利用 11 张建筑物图像，我们实现了从特征提取到三维点云生成的完整重建流程。首先，通过尺度不变特征变换（SIFT）算法从图像中提取关键点并进行匹配。接着，应用对极几何和透视 n 点（PnP）方法计算相机位姿。此后，通过三角定位技术从成对图像中计算三维坐标点。最终，通过 Bundle Adjustment 优化方法，实现了建筑物三维模型的重建与优化。

目录

1 引言

近年来，随着计算机视觉和机器学习技术的迅速发展，三维重建成为了数字化建筑与环境分析领域的重要工具之一。通过从二维图像中重建出场景的三维结构，我们能够更好地理解和分析现实世界中的各种复杂场景。SLAM 技术作为一种综合性的方法，不仅能够在无 GPS 信号环境下实现定位，还能够实现场景的实时三维重建。

在本次大作业中，我们以 11 张图像中的建筑物为例，通过一系列的步骤实现了从特征提取到三维点云生成的完整流程。具体而言，我们采用了尺度不变特征变换（SIFT）算法进行特征提取和匹配，基于对极几何和 pnp 技术计算相机位姿，利用三角定位法生成三维坐标点，并最终应用 Bundle Adjustment 进行全局优化，以获得高质量的建筑物三维点云模型。

1. **特征提取与匹配**：首先，我们使用尺度不变特征变换（SIFT）算法从各张图像中提取特征点。SIFT 算法能够有效地识别并描述图像中的关键点，且对旋转、尺度缩放、亮度变化保持良好的不变性。提取的特征点之后，利用 SIFT 描述子在多幅图像间进行精确的特征匹配。

2. **基于对极几何的相机位姿估计**：在获取特征匹配信息后，我们应用对极几何原理计算相机的初始位姿。对极几何通过分析成对图像中对应点的几何约束关系来估计相机的运动参数及其三维空间中的位置关系。

3. **利用 pnp 技术计算相机位姿**：随着更多 3D 点的生成，我们采用了透视 n 点（PnP）方法来精细调整相机位姿。PnP 算法利用已知的 3D 点和其在图像中的 2D 投影来求解相机的位置和姿态，这对于在已部分重建的场景中添加新图像特别有效。

4. **三角定位**：通过配对的图像特征点，我们运用三角测量法生成三维坐标点。这一过程中，我们解决了图像之间的对应点位置关系，利用它们在各自相机坐标系下的位置，计算出空间中的三维点位置。

5. **Bundle Adjustment（BA）**：为了进一步优化重建的准确性和稳定性，我们在最终步骤中应用了 Bundle Adjustment。这是一种全局优化技术，通过调整所有相机的位置和姿态以及 3D 点的坐标来最小化重投影误差，从而提高整体重建质量。

2 方法

本次作业我们以第 1 张图（相机）为基准，分别估计其余图片（相机）相对于第一张图的位姿，随后通过三角定位完成点云的生成。值得注意的是，只有相机 2 的位姿是通过对极几何方法生成的，其余的相机位姿都是利用 pnp 方法计算的，这是因为相对于对极几何方法，pnp 更加准确高效。整体流程设计可见算法 1。

接下来我们将详细说明每个部分的实现。

Algorithm 1 三维重建流程

对第一张图和第二张图应用特征点匹配
利用匹配得到的特征点和对极几何求出两图的相机位姿
使用三角定位根据两图的位姿和匹配点生成初步点云
for $i = 3$ to 11 **do**
    对第 i 张图和第 $(i-1)$ 张图应用特征点匹配
    利用匹配到的特征点中已有的 3D 坐标，应用 PnP 技术求出第 i 张图的位姿
    for $j = 1$ to $i-1$ **do**
        与第 i, j 张图进行特征点匹配
        利用 i, j 的位姿使用三角定位生成新的点云
    end for
    应用 Bundle Adjustment (BA) 优化所有点云和相机位姿
end for

2.1 特征提取与匹配

特征提取是三维重建流程的首要步骤，为此我们采用了尺度不变特征变换（SIFT）算法。SIFT 是一种广泛应用于计算机视觉中的特征提取工具，它由 David Lowe 于 1999 年提出，并在 2004 年完善。SIFT 特征对图像的旋转、尺度缩放、亮度变化表现出高度的稳健性，使其成为进行图像匹配和对象识别的理想选择。

2.1.1 SIFT 关键步骤

1. **尺度空间极值检测：** SIFT 算法首先构建尺度空间，这是通过对原始图像应用不同尺度（即不同模糊程度）的高斯模糊来完成的。每个尺度的图像都用来寻找局部极值点，这些极值点是潜在的不变特征位置。

2. **关键点定位：**在每个候选的特征位置，SIFT 算法进一步精确关键点的位置和尺度。这一步涉及对尺度空间的 Taylor 展开，以精确定位关键点的位置。

3. **方向赋值：**为了使特征具有旋转不变性，算法为每个关键点分配一个或多个方向，基于其局部图像梯度的方向。这确保了无论图像如何旋转，相同的特征点可以被匹配到。

4. **关键点描述：**在每个关键点周围，SIFT 算法提取局部图像特征，并将它们转换成一个稳健的特征向量，即 SIFT 描述子。这些描述子在后续的图像匹配过程中用来比较不同图像中的特征点。

2.1.2 特征匹配过程

提取特征的目的本质上是要定位两幅图中的同一个点，从而可以生成 3d 坐标。因此在特征提取完成后，还需要进行特征匹配，这是通过比较不同图像中的 SIFT 描述子来实现的。常用的匹配策略包括最近邻匹配，其中通常采用 KD 树（K-dimensional tree）来快速实现。为了提高匹配的准确性，还可以采用最近邻距离比测试（Lowe's ratio test），该测试可以有效地剔除错误匹配，增强匹配结果的鲁棒性。本次大作业我们选择的是 KD 树方法结合 knn 实现。

通过这些方法，我们能够从多视角图像中提取并匹配稳健的特征点，为后续的相机位姿估计和三维点云构建打下坚实的基础。

具体实现的时候，我们调用 cv2 的 SIFT 实现，结合 FlannBasedMatcher 方法完成匹配。

2.2 基于对极几何的相机位姿估计

在三维重建的过程中，估计相机位姿是通过对极几何来实现的。对极几何描述两个视图之间的几何关系，为从成对图像中推导相机的相对运动提供了一个强大的理论框架。对极几何的基本元素包括：

1. **基本矩阵（Fundamental Matrix）F**：基本矩阵 $\mathbf{F}$ 用于未校准的相机，它描述两幅图像中对应特征点之间的对极约束。对于匹配点 $\mathbf{x}$ 在第一幅图像中和 $\mathbf{x}'$ 在第二幅图像中的位置，基本矩阵满足关系 $\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$。

2. **本质矩阵（Essential Matrix）E**：对于已校准的相机，本质矩阵 $\mathbf{E}$ 是使用的核心矩阵，它可以表示为 $\mathbf{E} = \mathbf{K}'^{\top}\mathbf{F}\mathbf{K}$，其中 $\mathbf{K}$ 和 $\mathbf{K}'$ 是两个相机的内参矩阵。本质矩阵直接关联到相机的旋转和平移，因此对于位姿恢复尤其重要。

3. **极线（Epipolar Lines）**：极线是图像中的直线，对于图像 1 中的任意点，其在图像 2 中的对应点必须位于该点的极线上。极线有助于减少特征匹配过程中的搜索空间，提高匹配的效率和准确性。

本质矩阵 $\mathbf{E}$ 的恢复通常通过奇异值分解（SVD）来完成，即 $\mathbf{E} = \mathbf{U}\,\mathbf{V}^{\top}$。理论上，对于一个理想的本质矩阵，中间的奇异值矩阵 应该有两个相同的非零奇异值和一个零奇异值。分解结果中的 $\mathbf{U}$ 和 $\mathbf{V}$ 矩阵用于构建旋转 $\mathbf{R}$ 和平移 $\mathbf{t}$：

$$\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^{\top} \quad \text{或} \quad \mathbf{U}\mathbf{W}^{\top}\mathbf{V}^{\top}$$

$$\mathbf{t} = \mathbf{U}[:, 3] \quad (\text{即 } \mathbf{U} \text{ 的最后一列})$$

其中，$\mathbf{W}$ 是一个特定的旋转矩阵，用于确保 $\mathbf{R}$ 是一个合法的旋转矩阵。由于奇异值分解的不唯一性，从 $\mathbf{E}$ 中恢复的 $\mathbf{R}$ 和 $\mathbf{t}$ 有四种可能的组合。这四种解分别对应于 $\mathbf{t}$ 和 $-\mathbf{t}$ 以及由 $\mathbf{W}$ 和 $\mathbf{W}^{\top}$ 产生的两种旋转。为了确定正确的解，我们需要利用用于估算位姿的

匹配点，判断它们在给定位姿下是否大部分都位于相机前方。由于只有一组解能够满足所有的匹配点都在相机前方（深度大于 0），因此通过这种方式，可以排除不符合物理实际的解，确保恢复的位姿是正确的。

具体实现的时候，我们采用 cv2 的 findFundamentalMat 求解基础矩阵，再结合 numpy 的基础工具来进行 SVD 分解求解位姿。

2.3 基于 PnP 技术的相机位姿估计

如果我们拥有一组图像中一些特征点的 3D 坐标，那么我们还可以利用 PnP 来估算相机的位姿。PnP 是指透视 n 点问题（Perspective-n-Point, PnP）来实现，其目标是确定在给定 n 个 3D 点及其在图像中的 2D 投影的情况下相机的位置和方向。

2.3.1 PnP 问题的数学描述

给定一组 3D 点 $\{\mathbf{X}_i\}_{i=1}^n$ 和它们在图像上的对应 2D 投影 $\{\mathbf{x}_i\}_{i=1}^n$，PnP 问题旨在找到一个旋转矩阵 $\mathbf{R}$ 和一个平移向量 $\mathbf{t}$，使得投影关系

$$\mathbf{x}_i \approx \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}_i$$

得到最佳满足，其中 $\mathbf{K}$ 是相机的内参矩阵。

2.3.2 PnP 问题解决方法

解决 PnP 问题的一种常用方法是直接线性变换（DLT）。DLT 方法是解决 PnP 问题的一种线性方法，它不需要初始的参数估计。给定一组 3D 点 $\{\mathbf{X}_i\}_{i=1}^n$ 和它们在图像上的对应 2D 投影 $\{\mathbf{x}_i\}_{i=1}^n$，目标是找到一个旋转矩阵 $\mathbf{R}$ 和一个平移向量 $\mathbf{t}$。每个 3D 点 $\mathbf{X}_i$ 通过相机的投影矩阵 $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ 映射到其对应的 2D 点 $\mathbf{x}_i$。在齐次坐标下，这可以表示为：

$$\mathbf{x}_i \sim \mathbf{P}\mathbf{X}_i$$

其中 $\sim$ 表示等比例关系。

为了构建 DLT 算法，首先将每个 2D 点 $\mathbf{x}_i$ 转换为归一化平面上的点，去除内参影响。然后，对于每个点对，构建两个方程来描述 x 和 y 坐标的映射关系。通过这样的处理，可以为所有点构建一个线性方程组：

$$\mathbf{A}\mathbf{p} = \mathbf{0}$$

其中，$\mathbf{A}$ 是一个由所有点方程构成的矩阵，$\mathbf{p}$ 是包含投影矩阵参数的向量。

解这个方程系统通常通过奇异值分解（SVD）来找到最小奇异值对应的奇异向量，该向量即为解向量 $\mathbf{p}$，它可以被重塑成投影矩阵 $\mathbf{P}$。之后，可以通过分解投影矩阵 $\mathbf{P}$ 来恢复 $\mathbf{R}$ 和 $\mathbf{t}$。

具体实现时，我们调用 cv2 库的 solvePnP 函数来完成。

2.4 三角定位

三角定位是三维重建中的一个关键过程，它利用两幅或多幅图像中的匹配特征点来恢复场景中点的三维坐标。当两个相机（或同一相机在不同位置）拍摄同一场景时，从各自的视角观察到的场景点会在各自的图像平面上有不同的投影。通过这些投影点和相机的几何关系，可以计算出场景点的精确三维位置。

考虑两个相机的投影矩阵分别为 $\mathbf{P}_1 = \mathbf{K}[\mathbf{R}_1|\mathbf{t}_1]$ 和 $\mathbf{P}_2 = \mathbf{K}[\mathbf{R}_2|\mathbf{t}_2]$，其中 $\mathbf{K}$ 是相机的内参矩阵，$\mathbf{R}_1, \mathbf{R}_2$ 和 $\mathbf{t}_1, \mathbf{t}_2$ 分别是旋转矩阵和平移向量。给定两个相机中相对应的点的图像坐标 $\mathbf{x}_1$ 和 $\mathbf{x}_2$，三角定位的目标是找到一个三维点 $\mathbf{X}$，使得其在两个相机中的投影与观测到的图像坐标尽可能一致。

通常情况下，直接求解 $\mathbf{X}$ 通常会遇到不一致的问题，因为实际观测可能存在误差。因此，三角定位通常转化为一个优化问题，即最小化重投影误差：

$$\min_{\mathbf{X}} \left(\|\mathbf{x}_1 - \mathbf{P}_1\mathbf{X}\|_2^2 + \|\mathbf{x}_2 - \mathbf{P}_2\mathbf{X}\|_2^2 \right)$$

这个优化问题可以通过线性方法、迭代方法或其他数值优化技术解决。

三角定位的实现包括以下步骤：

1. 确定匹配的特征点对 $\mathbf{x}_1$ 和 $\mathbf{x}_2$。
2. 使用相机的几何信息（投影矩阵或本质矩阵以及相机内参）构建约束方程。
3. 通过求解优化问题来估计三维坐标 $\mathbf{X}$。
4. 验证三维点 $\mathbf{X}$ 是否位于两个相机前方，以确保解的有效性。

具体实现时，我们调用 cv2 的 triangulatePoints 方法。

2.5 Bundle Adjustment

Bundle Adjustment (BA) 是三维重建中的关键优化技术，用于通过最小化重投影误差来同时精确调整相机的位姿和场景中的三维点坐标。它通常在三维重建的末尾阶段执行，以确保高精度和一致性。

Bundle Adjustment 的目标： Bundle Adjustment 的核心目标是通过调整相机的位姿（旋转 $\mathbf{R}$ 和平移 $\mathbf{t}$）以及场景点的坐标 $\mathbf{X}$，最小化所有视角下所有点的重投影误差。这个目标可以形式化为一个非线性最小二乘问题：

$$\min_{\mathbf{C},\mathbf{X}} \sum_{i=1}^{m} \sum_{j=1}^{n} \|\mathbf{x}_{ij} - \pi(\mathbf{P}_i, \mathbf{X}_j)\|_2^2$$

其中，$\mathbf{C}$ 表示相机参数，$\mathbf{X}$ 表示场景中的三维点坐标，$\mathbf{x}_{ij}$ 是第 i 个相机观测到的第 j 个点的图像坐标，$\pi(\cdot)$ 表示投影函数。

解决 Bundle Adjustment 的最小二乘方法：为了解决这一优化问题，通常采用 Levenberg-Marquardt 算法，这是一种特别适用于解决非线性最小二乘问题的方法。该算法结合了梯度下降法和高斯牛顿法的优点，以快速逼近最优解。在实现中，还需要考虑问题的稀疏性，即大多数场景点仅在少数视图中可见，这可以通过使用稀疏矩阵技术来显著减少计算和存储需求。

具体实现时,我们参考了https://scipy-cookbook.readthedocs.io/items/bundle_adjustment.html中的实现代码。具体的，我们需要建立 3D 的点集合，每个 3D 点的若干个视角以及每个视角所对应的相机。

2.6 实现细节

参考算法1，我们需要按照算法的流程来实现我们的算法。在具体实现的时候，虽然每一个部分都已经实现，基本上是按照框架一步一步实现即可，但是仍然有许多值得注意的细节。首先是在做特征匹配时，对特征点匹配判定的阈值设计十分重要，一旦阈值设置的过高，就会出现大量的误匹配点。因此，合适的阈值选择十分重要。另外，在每一次利用一个相机对图像添加完 3D 点后，我们都会计算每个 3D 坐标在每个相机（当前已添加）的重投影误差，删除掉其中欧氏距离误差超过某个阈值的。由于图像的分辨率是 3072×2048 的，因此我们设置的阈值是 50 像素（距离）。另外一处值得注意的细节便是，在添加新的 3D 点时，我们需要先判断当前点云中是否已经存在已有的点。而这个过程的判断并不是直接利用 3D 坐标判断的，因为实际计算时很难保证没有误差，无法准确判断。注意到新加入的 3D 点都是由新的相机视角和旧的相机视角的匹配点生成的，而这些匹配点如果曾经还被用于添加过 3D 点，那么就说明之前匹配的 3D 点和本次要加入的点是同一个点，因此直接认为是那个点即可。至于误差，可以利用后续的 BA 优化来进行消除。为了避免累计误差逐步叠加到无法挽回的地步，我们每加入一个新的视角，就会进行一次 BA 优化，这样就能保证在进行下一步时当前的位姿是最优的。具体的实现细节可见代码以及注释。

3 实验结果

3.1 关键点展示

见图1。

3.2 关键点匹配

见图2。

图 1: SIFT 关键点提取结果

3.3 场景初始化结果

第二张图的位姿（以第一张图为标准参考系）:

$$R = \begin{pmatrix} 0.98788298 & -0.02116456 & -0.15375072 \\ 0.02370814 & 0.99961042 & 0.01472875 \\ 0.15337909 & -0.01819542 & 0.98799989 \end{pmatrix}, t = \begin{pmatrix} 0.99281062 \\ 0.03573469 \\ -0.11423708 \end{pmatrix}.$$

3.4 最终结果

见图3

3.5 消融实验

首先，我们移除了 BA 优化，并只用对极几何来重建点云，结果可见图4。可以发现，结果非常不理想。为此，我们再加入 BA 优化，来观察得到的结果（图5）。可以看到结果明显有了巨大的提升，但是和使用 pnp 相比点云密度会稀疏很多。接下来，我们再验证 BA 的作用。我们去掉 BA（图6）。可以看到虽然点云重建大致没有问题，但是出现了很多离群点（最后几张图的误匹配）。最后，我们单独验证一下只执行一次 BA 优化，也就是在所有相机的视角都加入后（图7）。可以发现仍然有误匹配点的存在，不过比完全不要 BA 会更少一点。这说明我们每一步都做一次 BA 是更加有效合理的。

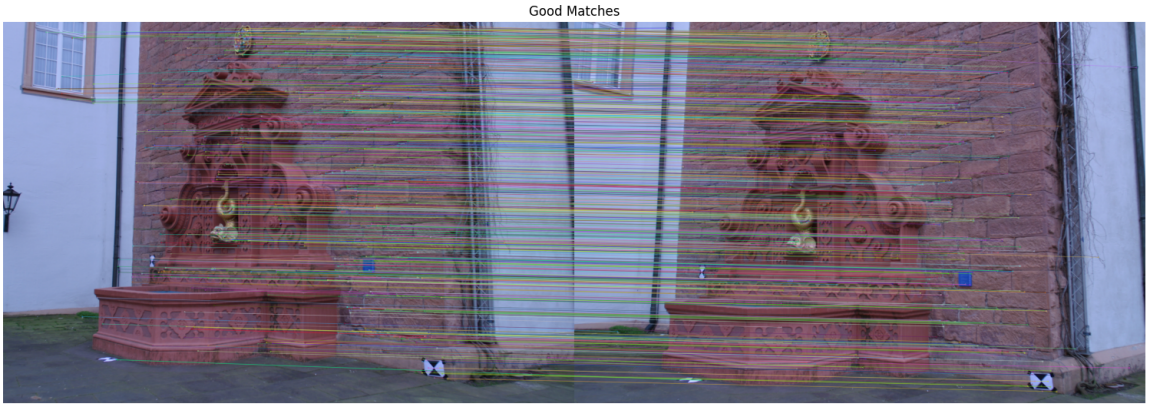

图 2: SIFT 关键点匹配结果

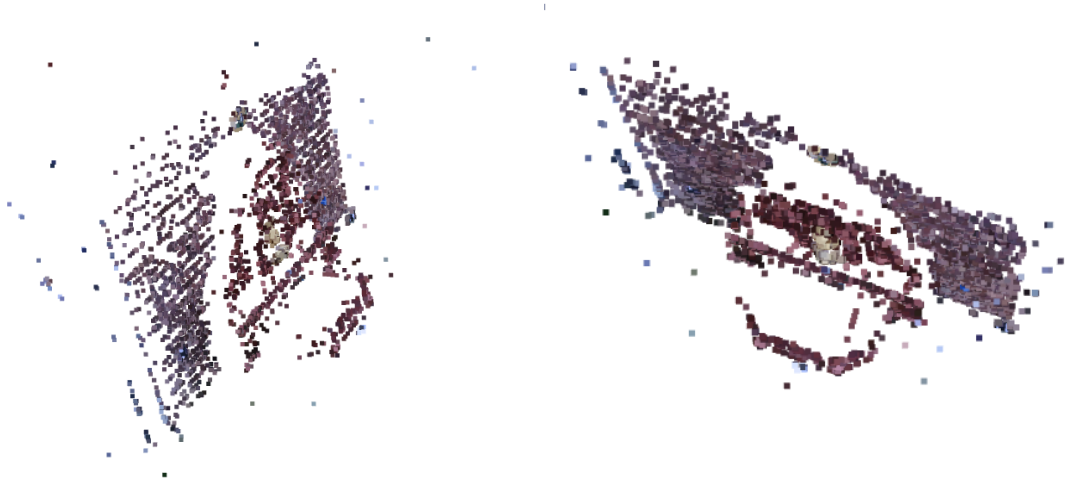

图 3: 重建结果（完整）

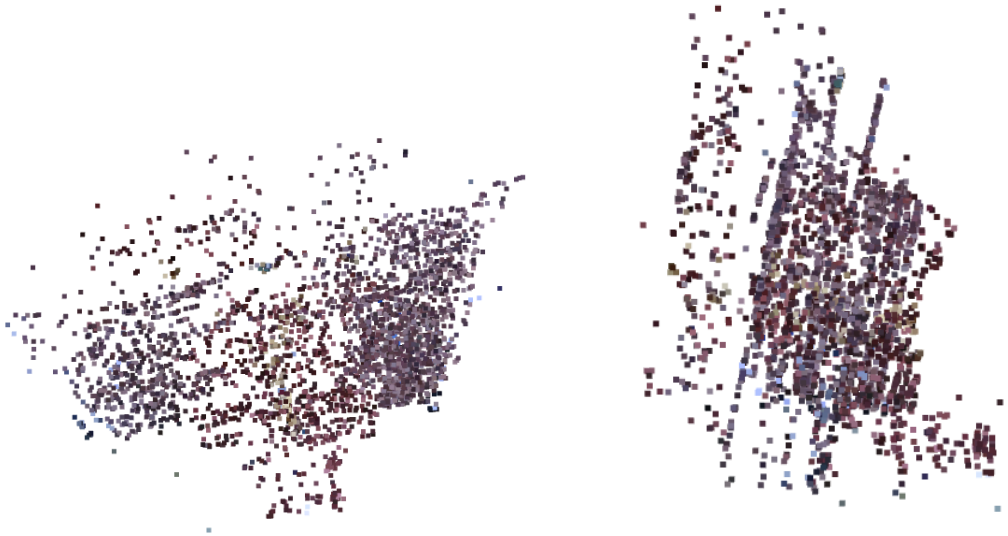

图 4: 重建结果（无 BA，无 PNP）

图 5: 重建结果（无 PNP）

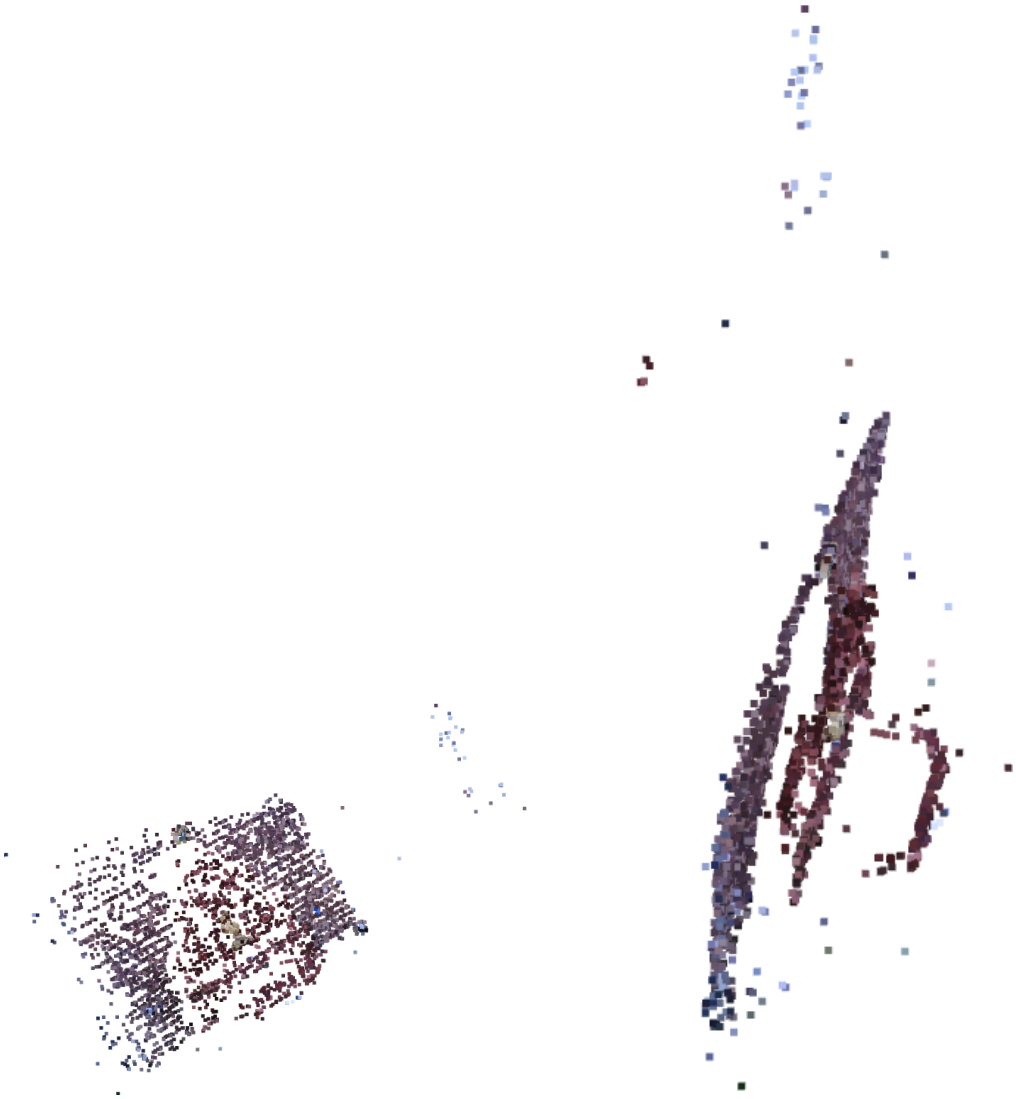

图 6: 重建结果（无 BA）

图 7: 重建结果（一次 BA）