2021 春计算机视觉综合课程报告

功能实现（50）	完成相机的标定，得到内参数 K（10）	
	利用 OpenMVG 实现 Structure from Motion 算法，完成图像特征匹配、相机姿态估计、Bundle Adjustment 等功能，得到相机的外参数 R、t 和稀疏三维点云（25）	
	在真实数据集上测试结果（10）	
	对所得到的三维模型进行展示（5）	
报告格式（50）	报告是否按照规定模板撰写（5）	
	是否符合科技文献撰写的格式规范（5）	
	是否对算法原理有清楚的介绍（10）	
	是否有充分的测试样例和量化结果（10）	
	实验部分是否采用图、表相结合的方式进行展示（5）	
	是否有对实验结果的分析、归纳、总结（10）	
	有核心源代码（5）	
综合得分		

小组成员

学号	班级	姓名	承担主要工作	成绩
20181001775	191181	高天翔	COLMAP 调试运行 Windows openMVG, openMVS 调试运行 报告润色修改	
20181001095	191181	常文瀚	资料搜集 协助程序调试运行 报告撰写	
20181003174	191181	刘宇鹏	Ubuntu 下 openMVG, openMVS 调试运行 报告润色修改	
20181001448	191181	苏浩文	Ubuntu 下 openMVG, openMVS 调试运行 报告润色修改	

任课教师：孙 琨

原创性声明：

本人声明报告者中的内容和程序为本人独立完成，引用他人的文献、数据、图件、资料均已明确标注出。除标注内容外，不包含任何形式的他人成果，无侵权和抄袭行为，并愿意承担由此而产生后果。

作者签字：　　时间：2021.05.21

要求及评分标准

1. 功能（50 分）

主要考查：是否按要求实现了主要功能，即：（1）完成相机的标定，得到内参数 K；（2）利用 OpenMVG 实现 Structure from Motion 算法，完成图像特征匹配、相机姿态估计、Bundle Adjustment 等功能，得到相机的外参数 R、t 和稀疏三维点云；（3）可选项，利用 OpenMVS，根据前述步骤得到的内、外参数，进一步估计稠密三维点云；（4）在真实数据集上测试结果，并对所得到的三维模型进行展示。

2. 报告内容格式（50 分）

主要考查：（1）报告是否按照规定模板撰写；（2）是否符合科技文献撰写的格式规范，有摘要、关键词、目录、正文、参考文献等；（3）是否对算法原理有清楚的介绍，可附图说明；（4）是否有充分的测试样例和量化结果；（5）实验部分是否采用图、表相结合的方式进行展示；（6）是否有对实验结果的分析、归纳、总结（7）附核心源代码。

3. 每个小组成员的得分，在报告综合得分的基础上，根据个人承担工作量大小、任务难易程度上下浮动。

计算机视觉实习报告

院（系）：计算机学院

专　　业：计算机科学与技术

姓　　名：高天翔、常文瀚

刘宇鹏、苏浩文

班　　级：191181

指导教师：孙琨

2021 年 5 月 2 日

目录

从多个角度拍摄同一目标的多张图像

进行多视图三维重建

高天翔　　常文瀚　　刘宇鹏　　苏浩文

（中国地质大学，武汉，2021）

摘要：三维重建是计算机图形学、图像处理和计算机视觉研究中的一个重要课题,其包括基于三维点云的重建和基于图像的重建.基于图像的三维重建又包括多幅图像和单幅图像重建.本文侧重于多幅图像重建技术,在对大量有关多幅图像三维重建技术的文献理解和综合基础上,归纳总结了团队成员对课程知识及一些计算机视觉算法的理解，并介绍了团队对于课程最终结课作业的实践成果。

关键词：三维重建；点云；多视图；openMVG；openMVS

三维重建经过数十年的发展,已经取得巨大的成功，基于视觉的三维重建在计算机领域是一个重要的研究内容，主要通过使用相关仪器来获取物体的二维图像数据信息,然后,再对获取的数据信息进行分析处理，最后，利用三维重建的相关理论重建出真实环境中物体表面的轮廓信息。基于视觉的三维重建具有速度快、实时性好等优点，能够广泛应用于人工智能、机器人、无人驾驶、虚拟显示和 3D 打印等领域，具有重要的研究价值，也是未来发展的重要研究方向。

课题“从多个角度拍摄同一目标的多张图像-进行多视图三维重建”是使用单个数码相机对被重建物体多方位拍摄来获取多幅图像,利用所得图像进行该物体的三维点云重建,以达到对实际物体的形状恢复的目的。课题的实践过程中完成了：相机的标定(张氏标定法)、利用 openMVG 实现 Structure from Motion 算法等任务，最终在真实数据集上进行测试并展示了重建后的三维模型。

一、 张氏标定法

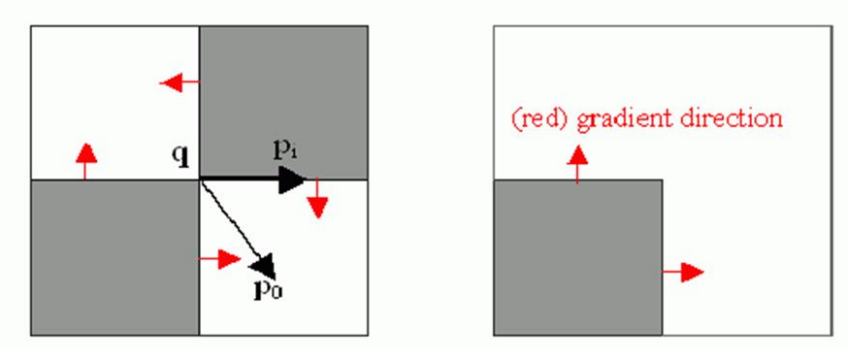
张正友针对径向畸变问题提出了一个新的求解摄像机内外参数的方法，即张氏标定法，该方法是介于传统标定和自标定之间的一种方法，它只需要摄像机对某个标定板从不同方向拍摄多幅图片，通过标定板上每个特征点和其像平面的像点间的对应关系，即每一幅图像的单应矩阵来进行摄像机的标定，由于该方法模板制作容易，使用方便，成本低，鲁棒性好，准确率高，因此得到了较为广泛的应用。该算法也属于两步法，摄像机和模板可以自由的移动，不需要知道运动参数。本文即采用张氏摄像机标定的方法。

1.1 目标

光学透镜具有固有透视失真，是相机畸变的主要部分，桶形畸变和枕形畸变，会使实际物体的直线在图像中产生弯曲，在手持相机模型中表现为径向畸变、切向畸变，这是由于透镜与成像平面不可能绝对平行造成的。这种畸变会造成图像中的某些点看上去的位置会比我们认为的位置要近一些。修正以上畸变。

1.2 方法

获取成像平面角点坐标，采用黑白棋盘标定板作为拍摄对象，寻找棋盘板所在平面与相机成像平面之间的关系，通过角点检测可以找到棋盘各个角点在图像中的像素位置，又已知每个角点在实际世界空间中的位置关系，依次寻找两平面的单应关系，优化亚像素坐标。



图（1） 畸变与修正

对于一角点 q，在其邻域区间内取任一点 p，向量 qp 一定与当前点 q 所在邻域内 p 点的灰度梯度方向的转置正交，于是有待最小化的误差式，其中(DI_p)^T 是 p 点的梯度方向转置

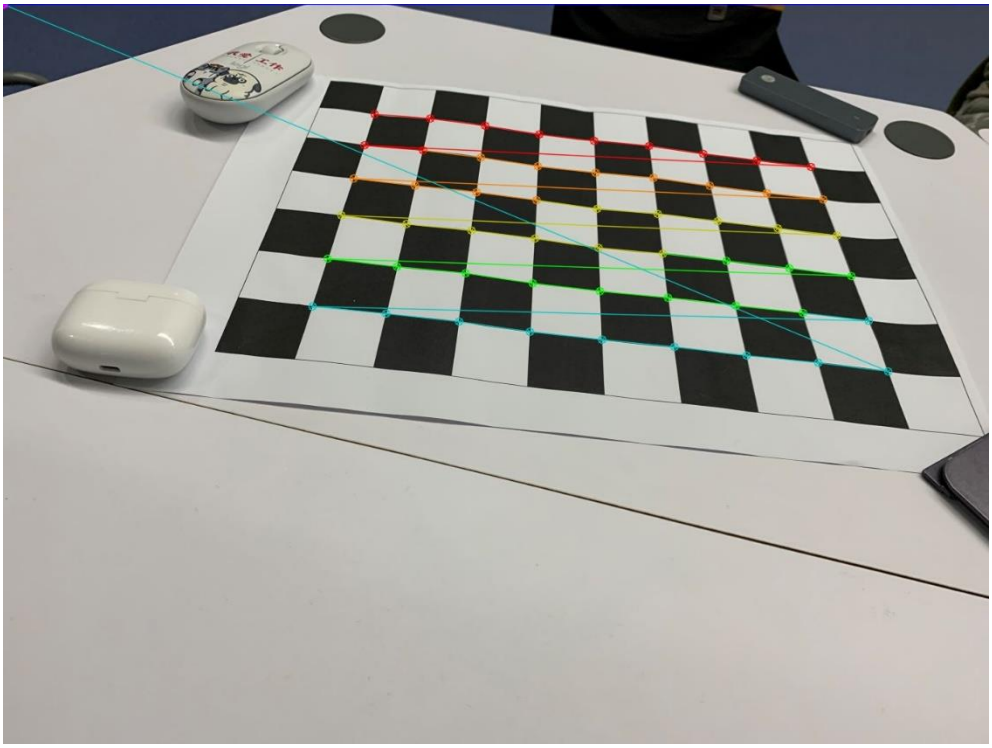
$$\epsilon_i = {DI_{p_i}}^T \cdot (q - p_i)$$

根据两平面单应关系和内参约束求解相机内参数：

$$\begin{cases} h_1^T K^{-T} K^{-1} h_2 = 0 \\ h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2 = 1 \end{cases} \Rightarrow \begin{cases} v_{22}^T b = 0 \\ v_{11} b = v_{12} b \end{cases}$$

$$H = [h_1 \quad h_2 \quad h_3] = \lambda K [r_1 \quad r_2 \quad t]$$

图像像素坐标系点 p 与棋盘平面坐标系对应点 P 关系是 p=K[R|t]P 令 H=K[R|t]，称 H 为单应矩阵，即就是两个平面对应点的坐标变换关系。最后根据等式求解得到相机各个内参数。



图（2）张氏标定法实践效果

1.3 标定结果

输入：通过打印棋盘图，使用手机拍摄 10 张不同角度棋盘图照片，进行张氏标定

输出：手机相机内参数矩阵

$$K=\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} 2941.5 & 0 & 1506.3 \\ 0 & 2941.5 & 1920 \\ 0 & 0 & 1 \end{bmatrix}$$

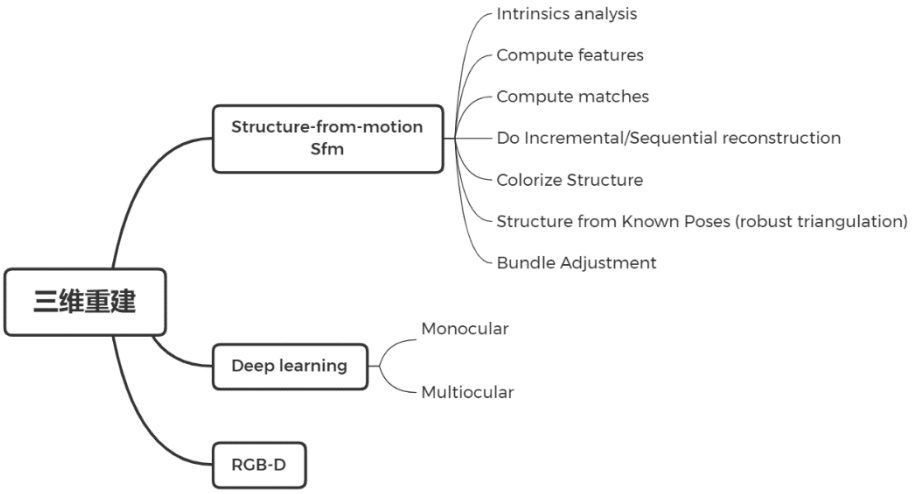
K 有 4 个未知数和相机的构造相关，*fx,fy* 和相机的焦距，像素的大小有关；*cx,cy* 是平移的距离，和相机成像平面大小有关。

二、 三维重建方法

从整体上来看，三维重建技术主要通过视觉传感器来获取外界的真实信息，然后，再通过信息处理技术或者投影模型得到物体的三维信息，也就是说，三维重建是一种利用二维投影恢复三维信息的计算机技术。

三维重建指对三维物体建立适合计算机表示和处理的数学模型，经过十多年的发展，已经取得了巨大的成就．随着科学技术的发展，利用计算机构建出真实景物的三维模型，已经成为很多领域开始深层研究之前必不可少的一部分．如在医学治疗中，利用三维 CT 图像判断病情; 在文物保护中，将文物进行三维复原，供游客参观．除此之外，在游戏开发、工业设计、航天航海等诸多领域，三维重建发挥着不可或缺的作用．目前，研究者们主要通过三种方式来获取三维模型: 一是直接根据传统的几何建模技术; 二是基于多幅不同角度拍摄的图像，运用计算机视觉理论进行三维建模，该方式为基于图像的三维重建; 三是利用三维扫描设备对目标进行扫描，然后重建目标的三维模型，该方式为基于点云的三维重建．在上述三种方法中，传统的几何建模技术发展最为成熟，利用该方法可以得到非常精确的三维模型，因此该方法已经广泛应用于诸多领域．然而采用这种建模技术周期长、操作复杂，对于许多不规则的自然或人为物体，建模效果与真实场景仍存在差异．因此基于图像与基于点云的三维重建成为了三维重建领域中的

热点.



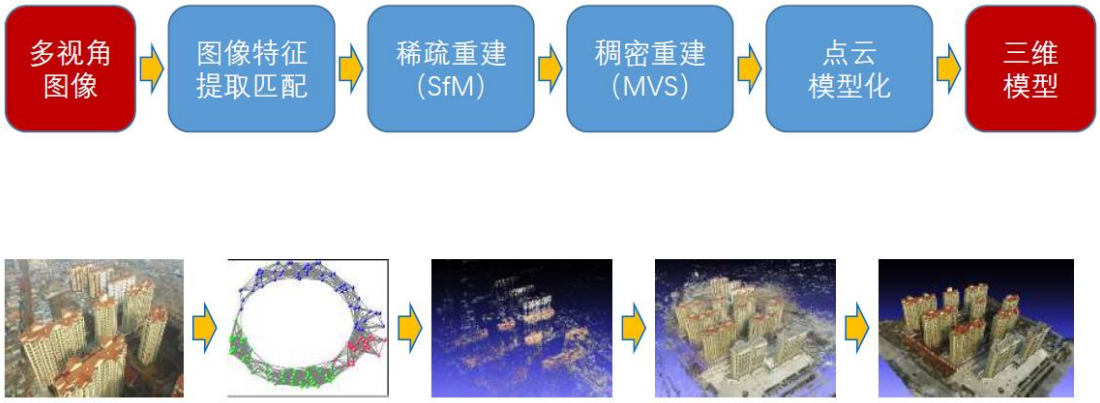
图（3）三维重建的几种方法

三维重建包含三个方面，基于 SFM 的运动恢复结构，基于 Deep learning 的深度估计和结构重建，以及基于 RGB-D 深度摄像头的三维重建。

基于图像的三维重建主要是将二维图像恢复成三维模型，其分为基于单幅图像的三维重建与基于多幅图像的三维重建. 基于单幅图像的三维重建不能应用于大规模场景，因此本文主要介绍了基于多幅图像的三维重建。

基于点云的三维重建主要是将点云数据恢复成图形、图像，然后通过计算机处理形成三维模型．其主要流程有: 数据配准、点云数据预处理、分割、三角网格化、网格渲染．由于点云获取设备获取点云角度单一，因此如何对多个点云数据进行配准拼接一直是研究的重点和难点。

我们可以利用 OpenMVG 实现 Structure from Motion 算法，根据多个视图实现图像特征提取和匹配、相机姿态估计、Bundle Adjustment 等功能，最后得到相机的外参数 R、T 和稀疏三维点云。



图（4）三维建模的步骤

2.1 SFM 算法原理初简介

稀疏点云重建是通过相机的移动来确定目标的空间和几何关系的. 首先输入一个场景的多视角图像，检测每张图像中的特征点，并对特征点进行匹配，然后运用 SFM 重构算法，恢复出相机的参数、相机之间的相对位置关系及场景的稀疏点云，最后捆绑调整优化相机位姿、相机参数、场景结构。

稀疏点云重建的流程为: 特征点检测，特征点匹配，相机位姿（旋转矩阵 R、平移矩阵 T)初始值的求解，捆绑调整优化相机位姿、相机参数、场景结构。

算法的关键是获得两张图片中的对应点，然后估计基础矩阵，再估计本征矩阵，再通过 SVD 分解求得较好的，得到物体的三维点，最后将多个稀疏点云融合在一起，这里一个常用的算法是 Bundle Adjustment。

在针孔相机模型下，由世界坐标系投影到像素坐标系的数学模型。图像中的像素坐标点和它在真实世界下的世界坐标点的对应关系为：

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

用矩阵形式表示：

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{X}$$

对于同一个世界坐标系下的点，在多个相机坐标系下成像，即为：

$$\mathbf{x}_1 = \mathbf{K}[\mathbf{R}_1 \mid \mathbf{t}_1]\mathbf{X}$$
$$\mathbf{x}_2 = \mathbf{K}[\mathbf{R}_2 \mid \mathbf{t}_2]\mathbf{X}$$
$$\mathbf{x}_3 = \mathbf{K}[\mathbf{R}_3 \mid \mathbf{t}_3]\mathbf{X}$$

如果能准确找到这些对应点，就可以准确计算出各相机的 R,t，但事实上只能用估计的方法求得较好的对应点。一个思路是提取各图像中物体的特征点，常用的算法是 SIFT，因其具有尺度和旋转不变性，再去做匹配，再用 RANSC 算法优化改善匹配对。之后利用 F 矩阵和 E 矩阵可以算出相机的 R,t，再通过三角化得到稀疏点云。

在 openMVG 管道中处理图像数据集的第一个任务是创建一个描述所用图像数据集的 sfm_data.json 文件。该结构化文件为每个图像列出了一个称为“视图”的对象。每个视图都与一个摄影机姿势索引和一个固有摄影机参数组（如果有）相关联。 分组表示可以在某些视图之间共享相机固有参数（这将导致更稳定的参数估计）。计算完数据集描述后，即可计算图像特征。

2.2 特征提取

2.2.1 特征点检测

局部特征点，主要分为斑点和角点两类特征点，斑点是指图像中像素或灰度值大的区域，角点是指图像中物体的拐角或线条交叉的部分．斑点检测主要包括二阶的拉普拉斯高斯边缘提取算法(Laplacian Of Gaussian，LOG) 、尺度不变特征变换算法(Scale Invariant Feature Transform，SIFT)、加速稳健特征算法(Speeded Up Robust Features，SURF)等．角点检测主要有 Harris 角点特征提取算子、加速分割测试特征提取(Features from Accelerated Segment Test，FAST) 、特征点检测算子(Oriented FAST and Rotated BRIEF，ORB) 等。1999 年 SIFT 算子被 Lowe 提出，并于 2004 年得到完善，该算法主要是寻找多尺度空间下的关键点，计算关键点的位置、方向、尺度信息，以生成特征点描述子．SIFT 算子提取的关键点精度高、不易受光照噪声的影响，但是该算子也存在计算量大、实时性能不高的缺点．Bay 提出了 SURF 算子，与 SIFT 算法不同的是，SURF 特征点主方向的分配不是通过梯度计算，而是利用了 Haar 小波，SURF 的计算速度是 SIFT 的 3 至 7 倍。在 Susan 角点检测的基础上，Rosten 等提出了 FAST 算子，随后

很多学者在该算子的基础上进行改进。

2.2.2 特征点匹配

特征点匹配即通过寻找两幅图像中相似的特征点，来确定两幅图像的位置关系。SIFT 和 SURF 算子在对特征点匹配时都是通过两特征点间的欧式距离计算出匹配度的。SURF 算子中加入了 Hessian 矩阵，能够判断出两个特征点的方向，相比于 SIFT 算子，SURF 算子匹配的精确度更高. SURF 算子与 SIFT 算子需要对 A 视角图像的某个特征点与 B 视角图像上的每个特征点进行匹配，因此耗时较长。

然而初选的匹配对可能还是不可靠，需要用几何约束去检测。这个测试是基于事实的，假设一个静止场景，不是所有的匹配特征点在实际场景中是符合物理规律的。那么就需要计算对极几何，F 矩阵可以把两张图片之间的像素坐标联系起来，并包含相机的内参信息。每一个符合的匹配对像素坐标都需要满足：

$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$$

然而像这种 F 矩阵计算出有很多噪声数据，需要用 RANSAC(随机抽样一致性)算法进行滤波，用直接线性变换(DLT)(需要八组对应点)来进行 RANSACA 假设，剔除不满足基础矩阵的匹配对。

用 RANSAC 去估计基础矩阵 F 的思路是：多次迭代以下流程：选 8 个点、用 DLT 算法计算 F、记录内点的数目，在这之后选取内点最多的对应 F。

openMVG 中具有视觉重叠的二进制计算图像使用由 openMVG_main_ComputeFeatures 计算的图像描述，我们建立了相应的推定光度匹配，并使用一些健壮的几何滤镜来过滤结果对应关系。一旦计算出匹配项，就可以选择将检测到的匹配项显示为 SVG 文件。

2.3 SFM 重构

SFM 重构的思想是利用相机运动轨迹来估算相机参数。相机在不同视点摄取多幅图像，利用这些图像计算出相机的位置信息以及运动轨迹，从而在空间坐

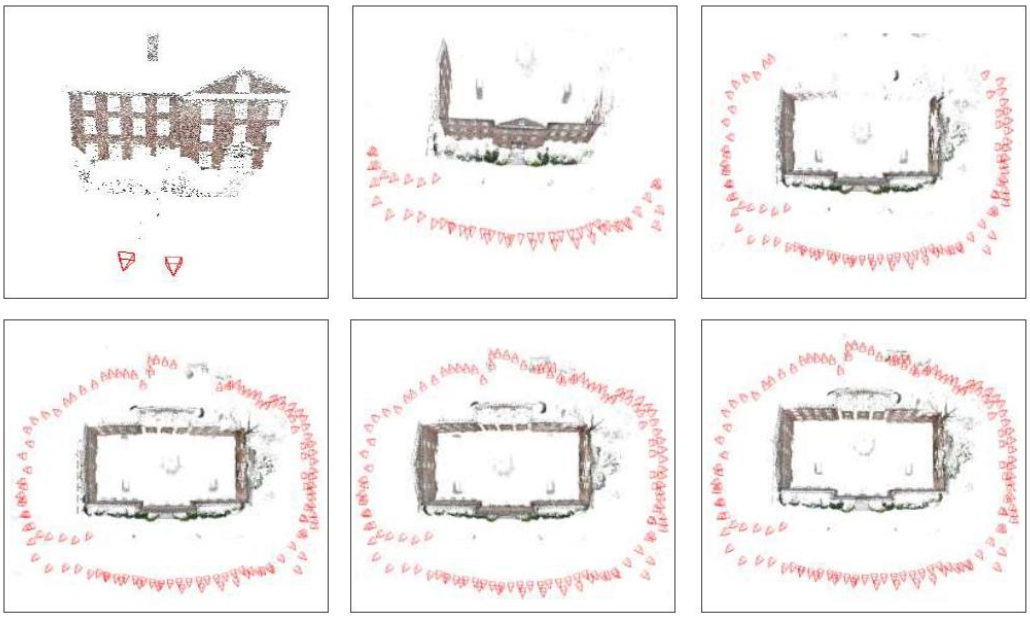
标系下生成三维点云，恢复出物体的空间结构。由于初步匹配过程形成的特征点较为粗糙，因此在估算相机参数之前，常用几何约束删除匹配错误的特征点对。SFM 重构的方法主要有增量式和全局式两种，我们进行试验使用的是增量式重构，因此下文我们主要介绍增量式重构。

增量式重构是从两张图像中重建出部分场景以及两相机的 R（旋转矩阵）、T（平移矩阵），然后添加图像，利用 PnP 计算出第三幅图像的 R、T，通过三角化重建出更多的场景．由于在求位姿与三维点时会有大量噪声干扰，后续计算结果会发生漂移，因此引入 BA(Bundle Adjustment) 算法进行优化。在每次添加图像完成三角化后都需要进行优化。

增量式重构的步骤为: 初始化、PnP 解算、三角化、BA 算法优化。其中较为重要的步骤是利用 PnP 算法求解相机位姿。PnP 有很多种类，如 P3P、EPnP、DLT、UPnP 和 Mre。

以图片案例为例，增量式重构的具体流程是: 首先输入第 1、第 2 张图像计算出相机位姿并重建出部分场景，然后输入第 3 张图像进一步调整相机位姿并重建场景，直到输入所有不同视角的图像，得到所有相机的初始位姿和初始三维点．如图 2 所示，通过输入第 1、第 2 张图像，重建出第 1 个场景，由于 2 张图重建出的场景效果不是很好，图 2 中没有将场景 1 展示出来; 再添加第 n 张图像与之前重建的场景 N-1 融合获得下一步的新场景，直到所有图像添加结束，重建出全部场景。

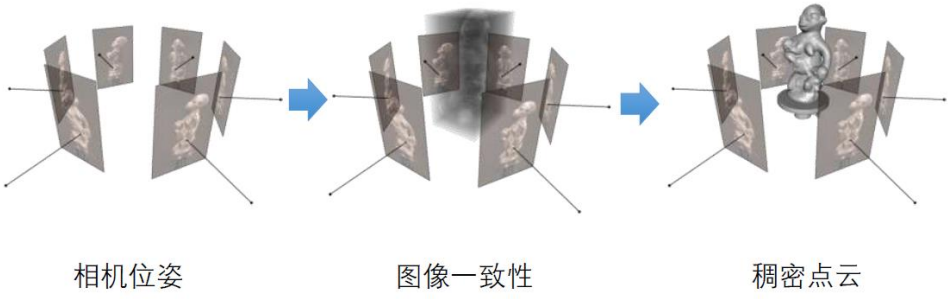
增量式 SFM 的缺点是：在对大规模场景进行三维重建时，由于误差累积会造成场景漂移以及反复地捆绑调整会耗费大量的时间。



图（5） 增量式重构案例

三、 稠密点云重建

由于 SFM 用来做重建的点是由特征匹配提供的，导致匹配点稀疏，重建效果并不是很好，所以需要 MVS 进行完善。稠密点云重建(MVS)的目的是通过最佳方法对不同图像上的同一个点进行匹配，增强重建场景的稠密性，我们的课题选择了 openMVS 进行实验。下图为 MVS 的基本思路。

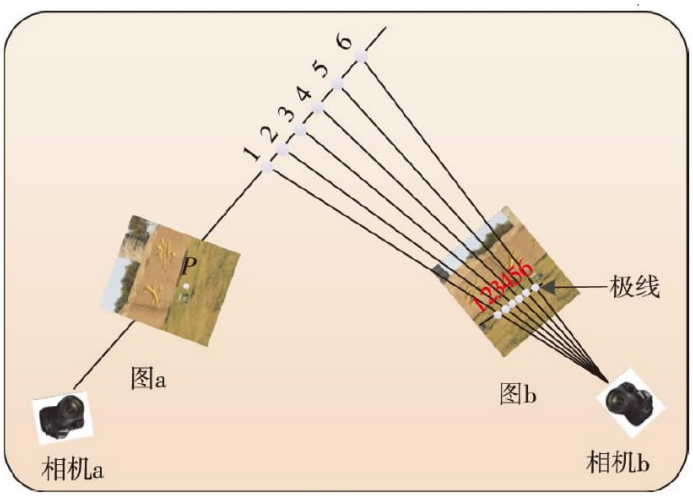


图（6）MVS 基本思路

3.1 一致性度量

如图 6 所示，假设图 b 中存在一点 P'与图 a 中点 P 在不同视角的同一点，由于几何约束，点 P 与相机 a 的返回射线和点 P' 与相机 b 的返回射线一定相交

于一点，点 P 所对应的点 P'一定位于极线上．MVS 主要任务是通过一种最佳搜索的方法，匹配不同图像的同一个点．接下来工作就是在图片上的一条线上进行探测，寻找两张图片上的同一点，主要方法为逐像素判断，由此提出了一致性度量函数。



图（7） 匹配点的几何约束示意图

两视图图像一致性的度量方法有：平均绝对差算法(Mean Absolute Differences，MAD)、差值平方和(Sum of Squared Differences，SSD)、绝对差值和(Sum of Absolute Differences，SAD)、归一化互相关性(Normalized Cross-Correlation，NCC) 等。这三个算法大同小异，采用的方法都是遍历整个搜索图，寻找到与模块图最为相似的部分，不同的只是计算方法不一样。

3.2 深度图的计算

深度图是指将相机到场景中各点距离作为像素值的图像．在 MVS 中对于深度图的计算是一大重点。深度图的计算方法有很多，在将深度图的计算这一技术应用到三维重建时，主要思想是将深度范围内

分割一个一个平面，通过观察投影到摄像机上的点的颜色是否一致来判断这些点是否位于可见物体的表面上．但是该算法难以应对遮挡问题，另外，当纹理弱、光照变化大时还会重建出很多杂点。

四、 实验流程

4.1 使用 COLMAP 重建

如图，当前正在新增第 11 个视角，当前影像可以看到已有点云的 2421 个，进行姿态估计(Pose Refinement Report)。

```
================================================================================
Registering image #11 (11)
================================================================================

  => Image sees 2421 / 3572 points

Pose refinement report
----------------------
    Residuals : 3478
   Parameters : 8
   Iterations : 9
         Time : 0.0177128 [s]
 Initial cost : 0.921852 [px]
   Final cost : 0.476509 [px]
  Termination : Convergence

  => Continued observations: 1736
  => Added observations: 732
```

图（8） 姿态估计

再进行 BA 优化，整体稀疏点云融合测量点 21 个，滤除测量点 53 个。

```
Bundle adjustment report
------------------------
    Residuals : 61540
   Parameters : 11606
   Iterations : 25
         Time : 1.21164 [s]
 Initial cost : 0.32689 [px]
   Final cost : 0.30443 [px]
  Termination : Convergence

  => Merged observations: 21
  => Completed observations: 19
  => Filtered observations: 53
  => Changed observations: 0.003022
```

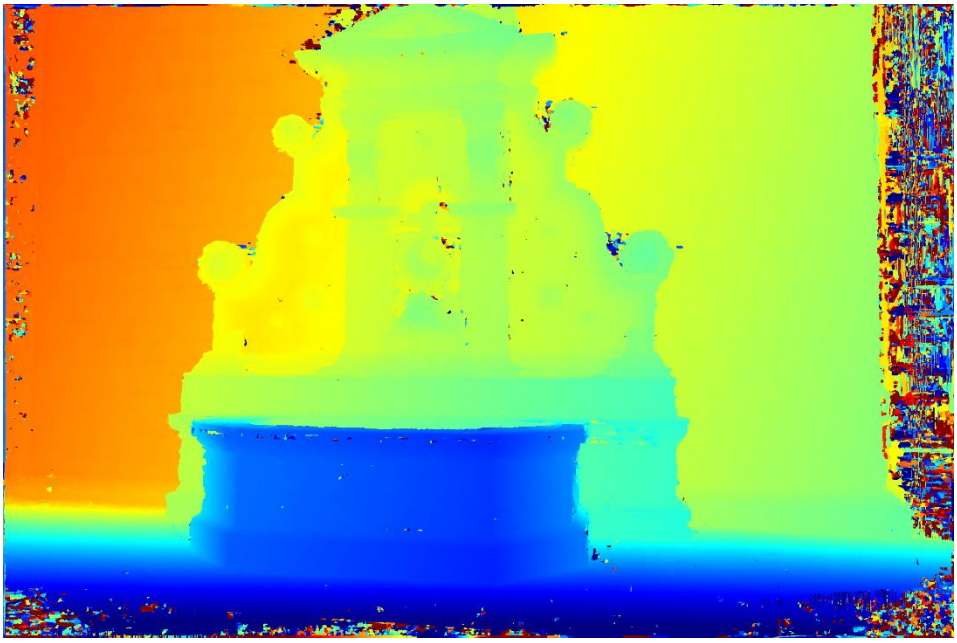
图（9） BA 优化

最后进行三角测量(Retriangulation)。上述过程结束后，进行迭代全局的 BA 优化，优化已有相机的姿态和三维稀疏点云坐标。

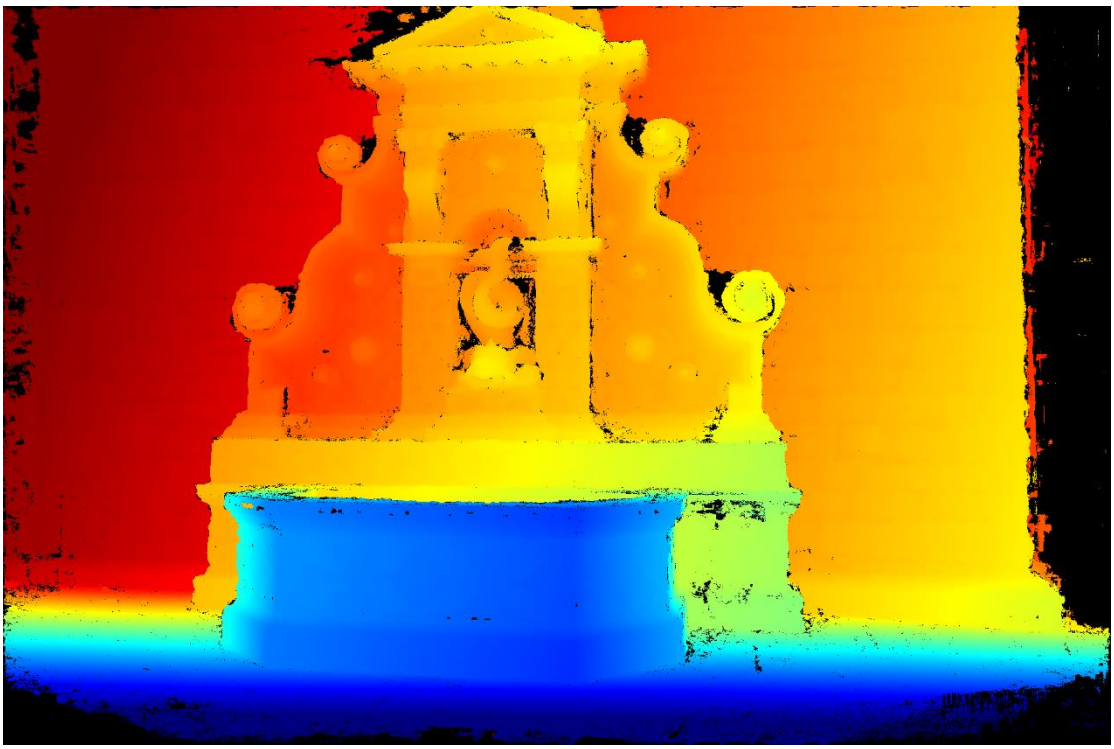
```
Bundle adjustment report
------------------------
    Residuals : 123812
   Parameters : 40176
   Iterations : 7
         Time : 0.871796 [s]
 Initial cost : 0.3215 [px]
   Final cost : 0.321452 [px]
  Termination : Convergence

  => Completed observations: 0
  => Merged observations: 0
  => Filtered observations: 0
  => Changed observations: 0.000000
  => Filtered images: 0

Elapsed time: 0.595 [minutes]
```

图（10） 三角测量

去除图像畸变（undistortion）后进行深度估计，可以得到优化前后的深度图，见图 11 优化前，图 12 优化后：



图（11） 优化前



图（12） 优化后

之后进行基于深度图融合的稠密重建。其原理为对估计出来的深度图，首先通过配准进行深度图融合，然后按照投影方法进行点云恢复。结果图 13 如下：

图（13） 点云恢复结果

COLMAP 可以用于快速三维重建，代码封装良好。但缺点为深度图估计速度过慢，且精度较低。

4.2 openMVG

4.2.1. 从图片数据集中生成场景描述文件

执行 openMVG_main_SfMInit_ImageListing，程序输入参数包括：[-i] 图像目录、[-d] 传感器宽度数据库、[-o] 输出目录；可选参数：[-f] 焦点、[-k] 内参数、[-c] 相机型号。

在 openMVG 管道中处理图像数据集的第一个任务是创建一个描述所用图像数据集的 sfm_data.json 文件。该结构化文件为每个图像列出了一个称为“视图”的对象。该视图存储图像信息和列表：图片名称、图片大小、内部摄像机校准信息（固有参数）。每个视图都与一个摄影机姿势索引和一个摄影机固有参数组（如果有）相关联。分组表示可以在某些视图之间共享相机固有参数（这将导致更稳定的参数估计）。

内部图像分析是基于 JPEG EXIF 元数据和相机传感器宽度的数据库进行的。如果图像中没有元数据，则可以直接指定已知的像素焦距值，或者让 SfM 进程自动找到它（必须定义至少两个共享公共关键点并且具有有效内在组的图像）像素的焦点计算如下：（如果在提供的传感器宽度数据库中找到了 EXIF 相机型号和制造商）

$$\text{focal}_{pix} = \frac{\max(w_{\text{pix}}, h_{\text{pix}}) * \text{focal}_{\text{mm}}}{\text{ccdw}_{\text{mm}}}$$

4.2.2. 计算图像特征

执行 `openMVG_main_ComputeFeatures`，程序参数包括：[-i] 输入文件、[-o] 输出文件路径

计算给定 sfm_data.json 文件的图像描述。 对于每个视图，它都会计算图像描述（本地区域）并将其存储在磁盘上有时您可能想要仅在图像的某些部分上计算特征/区域。对于这种需求，您可以使用遮罩。

遮罩只是具有与目标图像相同大小（宽度和高度）的二进制图像。 遮罩上的黑色区域表示“不良部分”，即要遮罩且未计算描述符的区域。 如果点位置处的遮罩值不等于 0，则保留一个点。在 openMVG_main_ComputeFeatures 中，遮罩和图像的关联是隐式的。

4.2.3. 计算几何匹配

执行 `openMVG_main_ComputeMatches`，程序参数包括：[-i] 输入文件、[-o] 输出文件路径。

具有视觉重叠的此二进制计算图像。使openMVG_main_ComputeFeatures计算的图像描述，我们建立了相应的推定光度匹配，并使用一些健壮的几何滤镜来过滤结果对应关系。

4.2.4. 执行增量三维重建

执行 openMVG_main_IncrementalSfM，程序参数包括：[-i | –input_file] 一个 SfM_Data 文件、 [-m | –matchdir] 存储几何匹配项的路径、 [-o | –outdir] 输出数据将被存储的路径

该步旨在在 sfm_data.json 文件和一些预先计算的匹配项上运行。openMVG_main_IncrementalSfM 显示一些初始对，它们共享许多重要的公共点。选择两个会聚的图像索引，然后将开始 3D 重建。 必须以多种对应关系选择初始对，同时保持足够宽的基线。

4.2.5. 计算场景结构颜色

执行 openMVG_main_ComputeSfM_DataColor，程序参数：[-i] 输入文件、[-o] 输出文件路径。

计算 SFM_DATA 场景的结构的颜色。该应用程序在 sfm_data.json 文件上运行。sfm_data 文件应包含： 具有定义好的内在函数和摄影机姿势的有效视图。

4.2.6. 测量稳健三角

执行 openMVG_main_ComputeStructureFromKnownPoses，程序参数：[-i | –input_file] 一个 SfM_Data 文件、[-m | –matchdir] 存储几何匹配项的路径、[-o | –outdir] 输出数据将被存储的路径。

此应用程序计算相应的特征，并根据已知的相机固有和姿势的几何形状对它们进行稳健的三角测量。该链在 sfm_data.json 文件和一些预先计算的匹配项上运行。 sfm_data 文件应包含：具有某些已定义的内部函数和照相机姿势的有效视图。

4.3 openMVS

校准并缝合所有摄像机视图后，默认情况下，OpenMVG 将生成 sfm_data.bin 文件，其中包含摄像机姿势和稀疏点云。 使用 OpenMVG 提供的导出器工具，将其转换为 OpenMVS 项目 scene.mvs.

OpenMVS 可以处理由任何“运动结构”求解器校准的场景，只要它接收到摄

像机的姿势，稀疏点云和相应的未失真图像作为输入即可。 所有要做的就是按照 Interface.h 标头文件中所述的 MVS 文件格式存储此信息。 该文件是独立的，可以按 SfM 求解器代码中的原样复制并直接使用它以 MVS 格式导出数据。

4.3.1.密集点云重构，以获得尽可能完整，准确的点云

执行命令： `DensifyPointCloud scene.mvs`

如果缺少场景部分，则密集重建模块可以通过估计密集点云来恢复它们，默认情况下采用 Patch-Match 方法, 致密化模块沿着 MVS 格式的密集场景还存储了 DMAP 格式的每个已处理图像的深度图。 查看器模块可用于可视化 DMAP 文件并将其导出为 PLY 点云。

4.3.2.网格重建，用于估计最能解释输入点云的网格表面

执行命令： `ReconstructMesh scene_dense.mvs`

前面步骤中获得的稀疏或密集点云用作网格重建模块的输入，构建粗网格。

4.3.3.网格细化可恢复所有细微的细节

执行命令： `RefineMesh scene_mesh.mvs`

从稀疏或密集点云中获得的网格可以进一步细化，以恢复所有细微的细节甚至更大的缺失部分。 接下来，对仅从稀疏点云获得的粗糙网格进行细化。

4.3.4.网格纹理，用于计算清晰准确的纹理以对网格着色

执行命令： `TextureMesh scene_dense_mesh_refine.mvs`

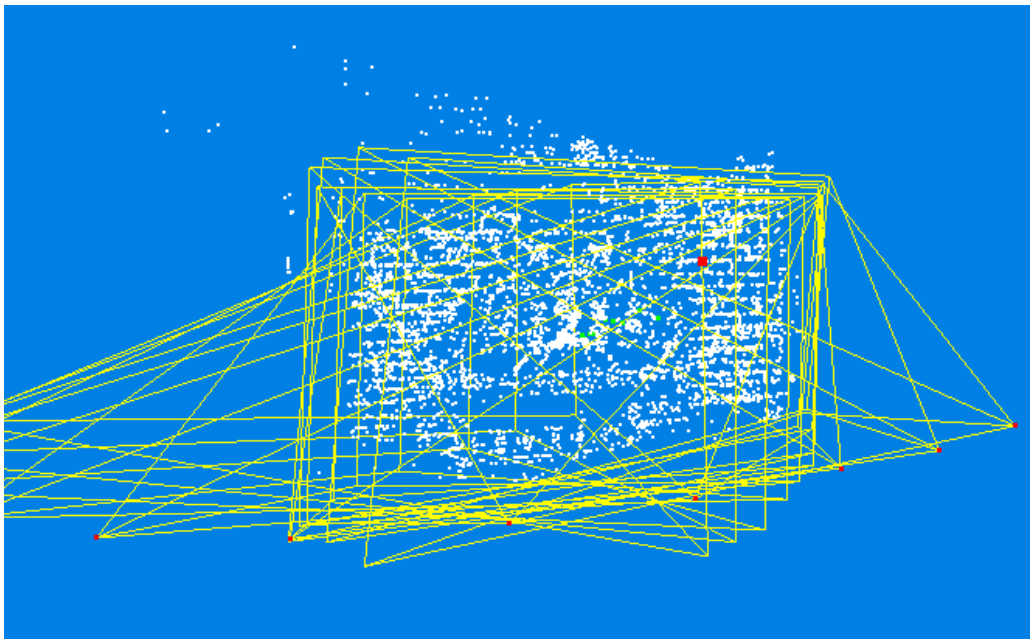
前面步骤中获得的网格用作网格纹理化模块的输入。橙色（默认）纹理的三角形在任何输入图像中都不可见，并且可以使用不同的颜色或将其删除。

五、 Fountain 数据实验结果

5.1 样例

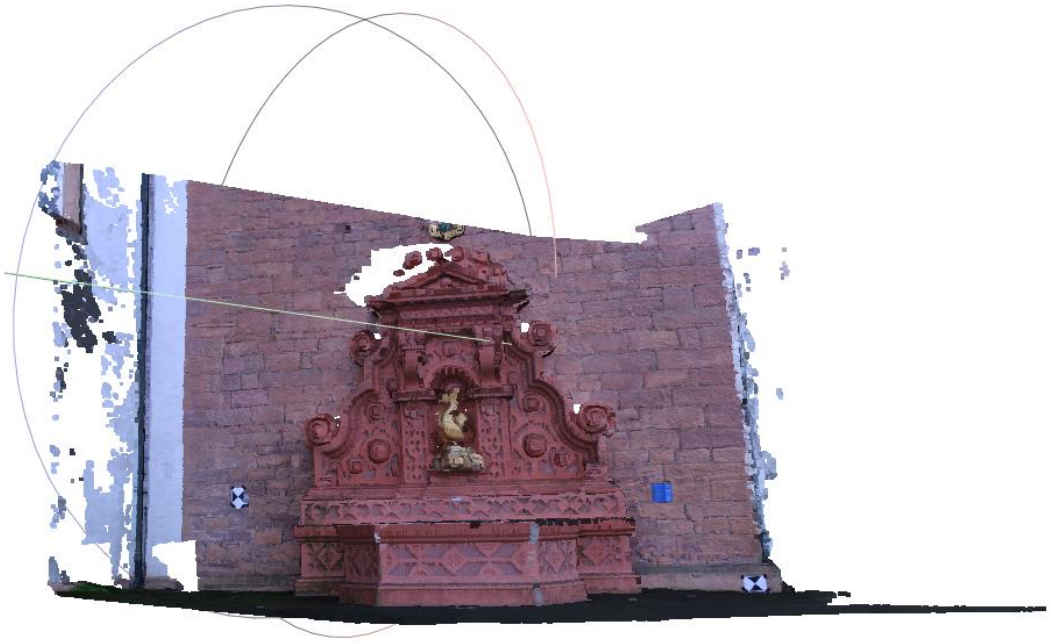
图（14） 原始图片

5.2 稀疏点云实验结果



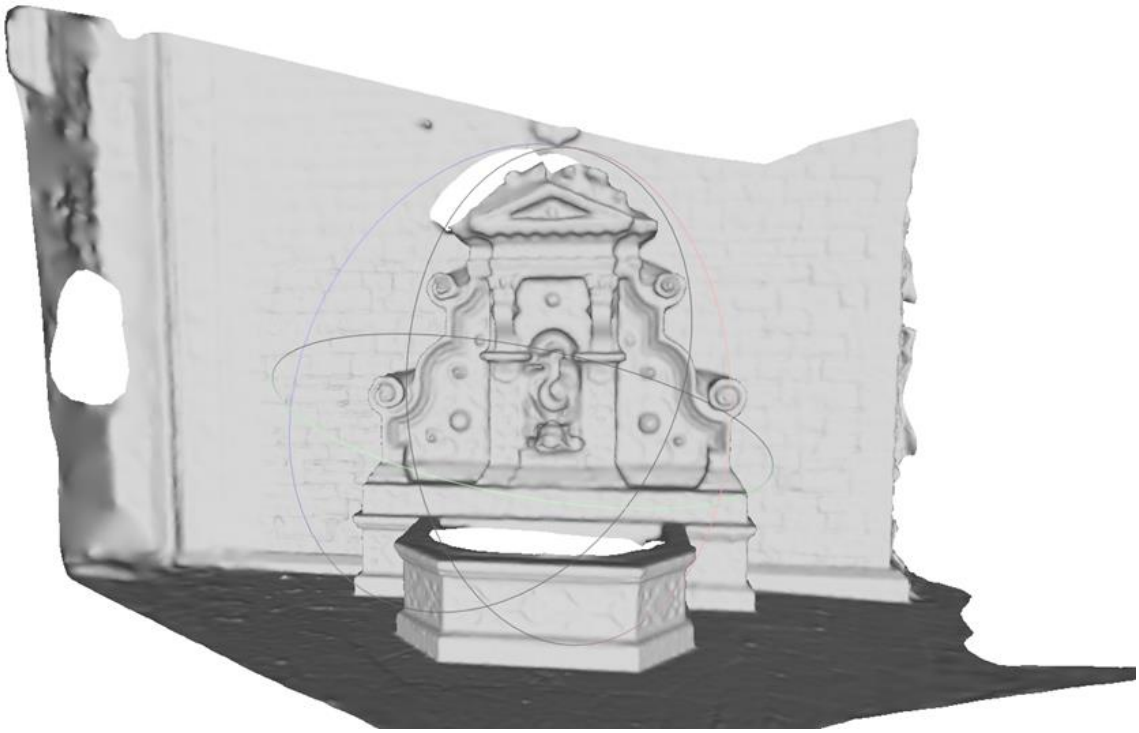
图（15） 稀疏点云

5.3 稠密点云实验结果



图（16） 稠密点云

5.4 细网格重建实验结果



图（17） 细网格

5.5 混合网格重建实验结果

图（18） 混合网格重建

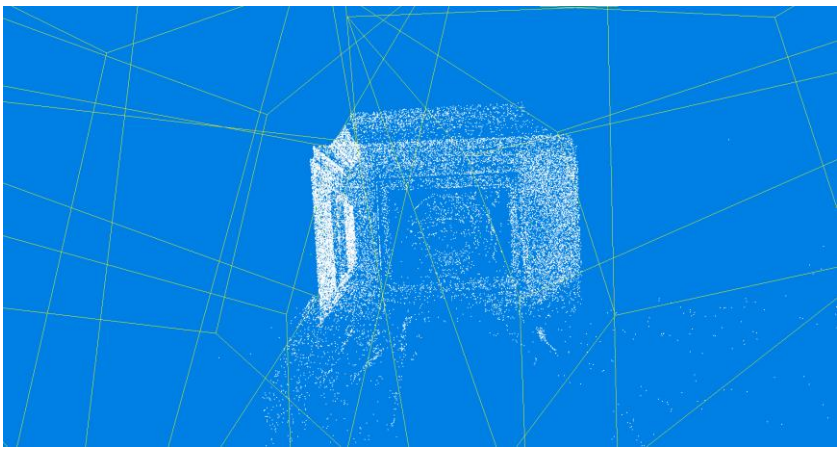
六、 真实数据集重建结果

实验结果部分展示了团队使用 openMVG 和 openMVS 对成员亲自收集的数据照片进行三维重建的稀疏点云和稠密点果图，并在最后对实验结果进行了分析。

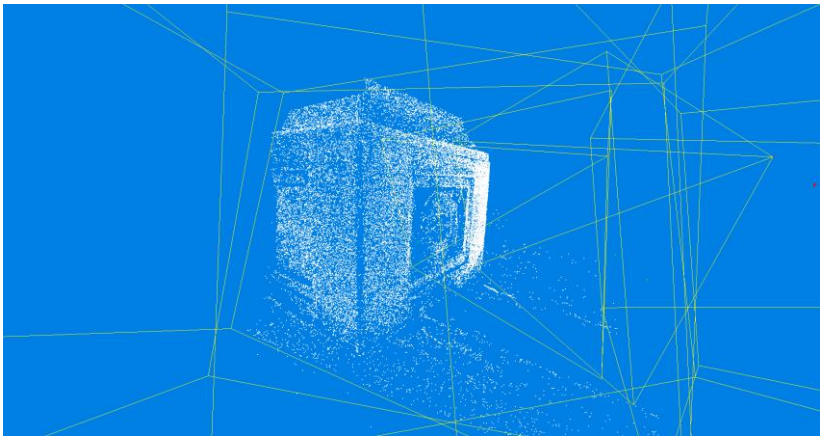
6.1 样例

图（19） 原始图片

6.2 稀疏点云实验结果

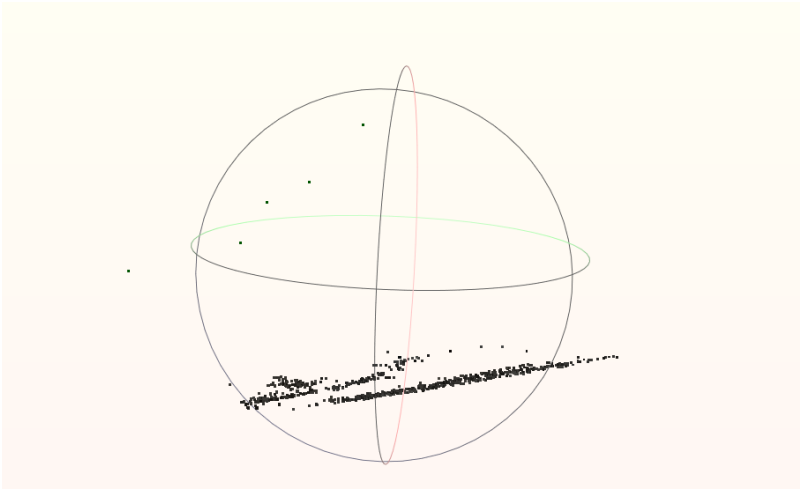


图（20） 稀疏点云效果



图（21） 稀疏点云效果图 2

为验证输入内参数准确性，使用错误内参数进行稀疏点云构建，结果如下：



图（22）错误内参数稀疏点云效果图

6.3 相机外参数结果

对于相机外参数 R，t,由七张图片得到七组数据，openmvg 重建稀疏点云第五步执行 `openMVG_main_ComputeSfM_DataColor` 命令，输出二进制文件 `sfm_data.bin`,其中包含相机姿态估计，其中旋转矩阵 R 为 3x3 矩阵，平移距离 t 使用现实坐标与相机坐标中心点相对位置表示，为 1x3 矩阵。部分数据如下：

第一张图片相机姿态估计：

R：

0.560817154	-0.054239428	0.826161125
0.265574916	0.956908709	-0.11745504
-0.784190081	0.285278472	0.55105545

t：

1.168722224	-0.099954669	0.656859743

6.4 稠密点云实验结果

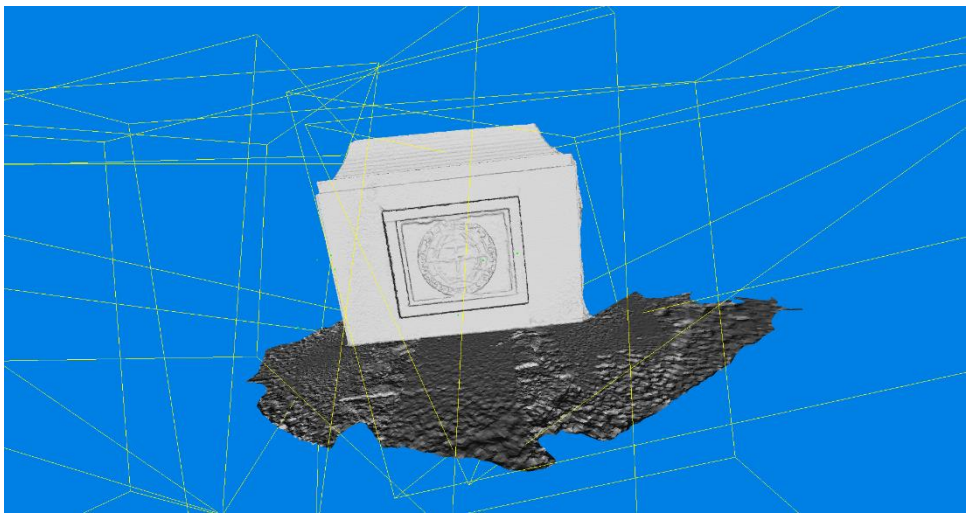
图（23） 稠密点云效果图 1

图（24） 稠密点云效果图 2

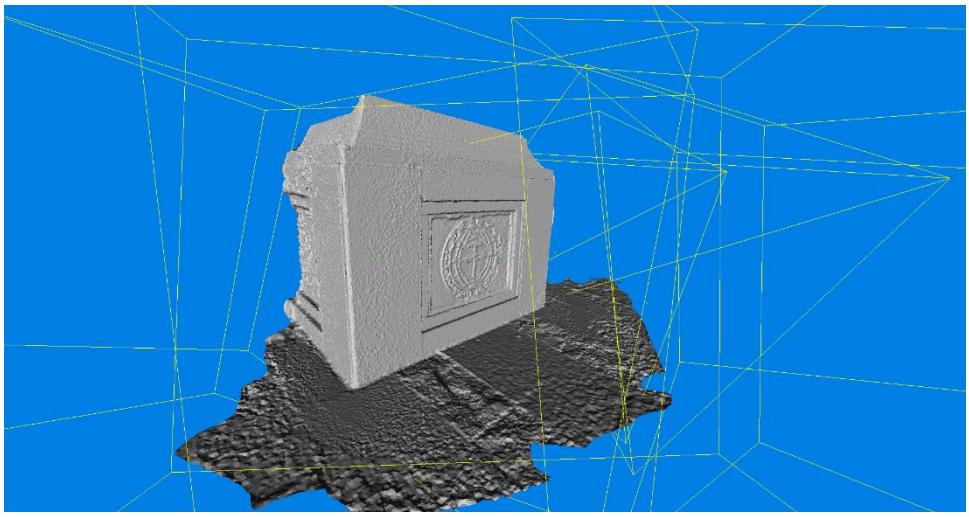
6.5 导出为点云文件后显示效果

图（25） 点云 ply 文件

6.6 细网格重建实验结果

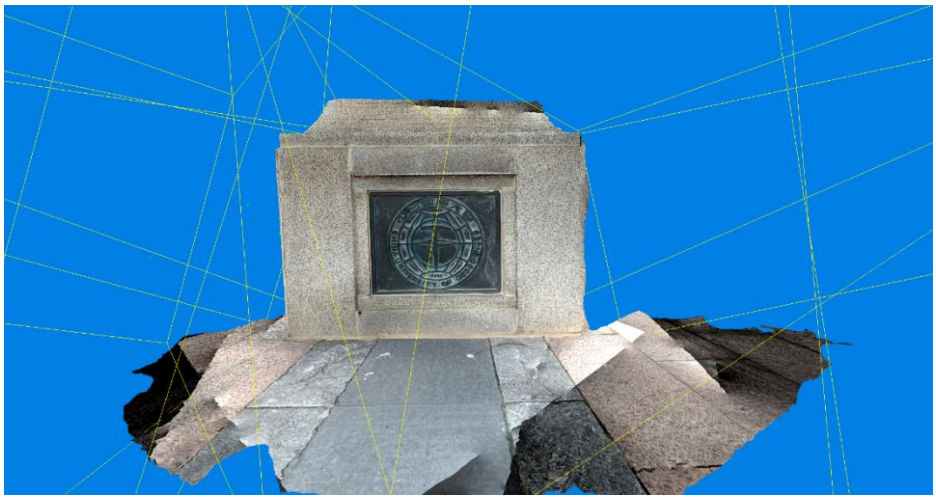


图（26） 细网格结果 1

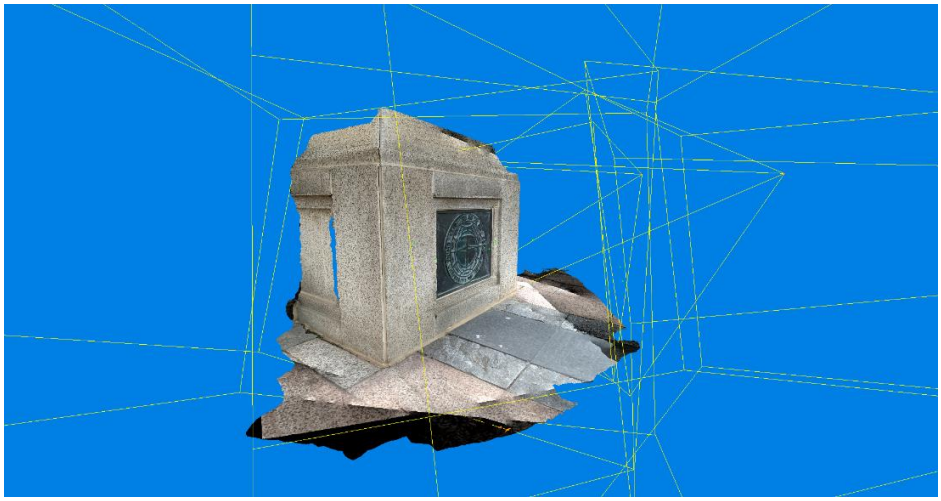


图（27） 细网格结果 2

6.7 混合网格重建实验结果



图（28） 混合网格结果 1



图（29） 混合网格结果 2

图（30） 混合网格结果 3

七、 实验感受

7.1 问题归纳与反思

①openmvg 从图片数据集中生成场景描述文件时，缺失 rubust.bin 文件，无法正确运行，但是程序示例图片可以正常生成稀疏点云。

解决方法：通过网上查询，最终在 GitHub Issues 找到解决方法，对于非示例图片集，在运行 openMVG_main_SfMInit_ImageListing 命令时，要指定焦距数值为自定义图片集文件的 1.2*max{宽，高}

②openmvs 无法正常运行编译后文件命令

解决方法：需要将编译后生成的可执行文件添加到 Ubuntu 系统的/usr/bin 文件夹内，才可以通过终端执行 openmvs 命令

7.2 总结

计算机视觉是一个要求动手能力很强的一门实践课程，在课程设计期间小组成员努力将自己以前所学的理论知识向实践方面转化，尽量做到理论与实践相结合， 认真完成老师布置的任务。

此次的结课作业，是从多个角度拍摄同一目标的多张图像-进行多视图三维重建 ，在设计过程中总是会遇到一些很小的问题，虽然不明显，却可以影响到整个程序的正常运作和结果。这样一个工程，却是通过一次次的搭建，修改之后的结果，真是令人感慨万千。

在设计与完善的过程中，第一个问题就是对三维重建的操作工具不熟悉。虽然看过了一些参考视频，但是到了开始编写的时候还是会出现不熟练的问题，但是随着一次次的操作，成员也开始越来越熟练了，虽然在这个过程中出了很多的差错，但随着修改也会逐步完善并产生许多新的想法。

第一次的三维重建并将结果进行分析虽然有所困难，却也使我们更加深入的了解了“张氏标定法”、SfM 三维重建方法的流程、OpenMVG 和 OpenMVS 的基

本思路以及原理。并将平时所学的知识第一次融会贯通。也明白了对稀疏点云重建和稠密点云重建是需要花费很多精力去构思的，其间收获的财富是任何时候的上课实验所不可比拟的。

计算机视觉是每一个计算机专业大学生在大学生涯中都不可或缺的课程，它使我们在实践中了巩固了所学的知识、在实践中锻炼自己的动手能力；它又是对每一位大学生所学专业知识的一种拓展手段，它让我们学到了很多在课堂上根本就学不到的知识,不仅开阔了自己的视野，增战了自己的见识，也为我们以后进一步走向社会打下了坚实的基础，是我们走向以后走向工作岗位的奠基石。

八、 参考文献

[1] 成祺. 单目多视角三维重建算法设计与实现[D]. 内蒙古大学, 2019.
[2] 王刚. 基于多视角立体视觉的三维重建研究[D]. 哈尔滨工业大学, 2019.
[3] 王明珠. 基于单目多视点图像的铸件三维重建方法研究[D]. 哈尔滨理工大学, 2018.
[4] 张彦雯,胡凯,王鹏盛.三维重建算法研究综述[J].南京信息工程大学学报(自然科学版),2020,12(05):591-602.
[5] 缪君, 储珺, 张桂梅, 王璐. 基于稀疏点云的多平面场景稠密重建. 自动化学报, 2015, 41(4): 813-822.

九、 附录

①Openmvg 核心代码：

SfM_SequentialPipeline.py:

```
print ("1. Intrinsics analysis")
pIntrisics = subprocess.Popen( [os.path.join(OPENMVG_SFM_BIN, "openMVG_main_SfMInit_ImageListing"), "-i", input_dir, "-o", matches_dir,"-d", camera_file_params, "-f", "4838.4"],"-k","2941.50591,0.0,1506.3,0.0,2925.4,1920.0,0.0,0.0,1.0" )
pIntrisics.wait()

print ("2. Compute features")
pFeatures = subprocess.Popen( [os.path.join(OPENMVG_SFM_BIN, "openMVG_main_ComputeFeatures"), "-i", matches_dir+"/sfm_data.json", "-o", matches_dir, "-m", "SIFT"] )
```

```
pFeatures.wait()

print ("3. Compute matches")
pMatches = subprocess.Popen( [os.path.join(OPENMVG_SFM_BIN, "openMVG_main_ComputeMatches"), "-i", matches_dir+"/sfm_data.json", "-o", matches_dir] )
pMatches.wait()

# Create the reconstruction if not present
if not os.path.exists(reconstruction_dir):
    os.mkdir(reconstruction_dir)

print ("4. Do Sequential/Incremental reconstruction")
pRecons = subprocess.Popen( [os.path.join(OPENMVG_SFM_BIN, "openMVG_main_IncrementalSfM"), "-i", matches_dir+"/sfm_data.json", "-m", matches_dir, "-o", reconstruction_dir] )
pRecons.wait()

print ("5. Colorize Structure")
pRecons = subprocess.Popen( [os.path.join(OPENMVG_SFM_BIN, "openMVG_main_ComputeSfM_DataColor"), "-i", reconstruction_dir+"/sfm_data.bin", "-o", os.path.join(reconstruction_dir,"colorized.ply")] )
pRecons.wait()

# optional, compute final valid structure from the known camera poses
print ("6. Structure from Known Poses (robust triangulation)")
pRecons = subprocess.Popen( [os.path.join(OPENMVG_SFM_BIN, "openMVG_main_ComputeStructureFromKnownPoses"), "-i", reconstruction_dir+"/sfm_data.bin", "-m", matches_dir, "-f", os.path.join(matches_dir, "matches.f.bin"), "-o", os.path.join(reconstruction_dir,"robust.bin")] )
pRecons.wait()

pRecons = subprocess.Popen( [os.path.join(OPENMVG_SFM_BIN, "openMVG_main_ComputeSfM_DataColor"), "-i", reconstruction_dir+"/robust.bin", "-o", os.path.join(reconstruction_dir,"robust_colorized.ply")] )
pRecons.wait()
```

②openmvs 核心代码：

```
DensifyPointCloud scene.mvs
ReconstructMesh scene_dense.mvs
RefineMesh scene_mesh.mvs
TextureMesh scene_dense_mesh_refine.mvs
```