上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

# 计算机视觉课程大作业报告


题目：室内场景的三维重建

评分：

学生姓名：宋柏彰
学生学号：522031910809
专　　业：自动化
学院(系)：自感学院

# 摘 要

本次项目是计算机视觉课程的大作业，主要基于 SLAM 技术完成了一个较简单场景的三维重建任务。利用 62 张围绕恐龙化石采集到的图像，实现从特征提取到三维点云生成的流程，完成该场景的三维重建任务。整体流程大致为：首先对图像进行 SIFT 特征提取和匹配，然后利用对极几何计算相机位姿并完成场景初始化。随后利用 PnP 算法完成场景重建，最后利用 Bundle Adjustment 对重建场景进行优化。最后进行消融实验，分析各环节作用。

**关键词**: 三维重建，计算机视觉，SIFT，SLAM

# 第一章 引言

在计算机视觉领域中，三维重建是指根据单视角或多视角的图像重建三维信息的过程。多视角的三维重建相对于单视角更加容易，因为其可以类比人的双目视觉定位。随着计算机视觉和机器学习技术的发展，三维重建技术在许多领域均已经得到广泛应用，帮助研究人员更好的理解和分析现实世界中的各种复杂场景。

本次大作业期望实现的是一个室内场景的三维重建。通过 62 张围绕恐龙化石采集到的图片，采取一系列的步骤完成对该场景的三维重建任务。整体流程包含了对所有图像进行 SIFT(尺度不变特征变换算法)特征提取，获得后续匹配稳定且具备区分度的关键点及其描述子。随后进行相邻两张图像的特征匹配，在不同图像之间建立关键点的一一对应关系。再利用对极几何对场景进行初始化，从第一对图像恢复相对位并三角化出初始稀疏点云。再基于 PnP 算法进行三维的场景重建。最后，基于 Bundle Adjustment 方法对模型进行进一步的优化，获得点云和相机外参文件。

具体而言，各个步骤基本实现方法可以分解如下：

1. SIFT 即具有尺度不变性的特征变换，用于检测和描述图像局部特征。在这一步骤中，目标是获得图像的稳定且具备区分度的关键点和描述子用于后续稳定匹配。一般实现步骤为先在不同尺度上检测极值点，为每个关键点分配主方向以保证旋转不变性，最后计算关键点邻域的梯度直方图，形成 128 维描述子。
2. 图像特征匹配，首先需要对两幅图像的描述子集合计算其欧氏距离。对每个描述子取第 1、2 最近邻距离，通过设定比值剔除大量外点。基于 RANSAC 与对极几何约束，随机采样 8 对匹配点，估计基础矩阵 F。检验所有匹配对的 Sampson 距离，最后取内点最多的 F 并返回对应的内点集合。
3. 场景初始化的目标是根据第一对图像恢复出相对位姿并三角化初始点云，位姿信息包含旋转 R 和平移 t。由于已知相机内参矩阵 K，可以计算出本质矩阵 E。再通过 SVD 提取四种解，利用三角化后点的正深度约束确定正确解。
4. 增量式场景重建，首先建立 2D-3D 对应，在步骤一和二的基础上，利用已建点云对每个 3D 点投影到之前某帧，若与当前特征点匹配，则形成匹配对。然后基于 PnP 算法从 n 个 3D–2D 对应直接解出相机位姿 (R,t)。并在 RANSAC 里随机采样，迭代选取最优的位姿，剔除误差过大的外点。
5. 为了进一步优化重建的准确性和稳定性，采用 Bundle Adjustment 方法，在全局范围内同时优化所有的相机位姿和所有 3D 点，使重投影误差整体最小。

# 第二章 方法

基于上述核心流程，下面将逐个步骤详解其理论和代码实现的细节。

## 2.1 特征提取与匹配

SIFT（Scale Invariant Feature Transform）于 1999 年被 David G.Lowe 教授提出，SIFT 特征对图像的旋转、尺度缩放、亮度变化表现出高度鲁棒性，且信息量丰富，适用于在海量特征数据中进行快速，准确的匹配。少数物体也能产生大量的 SIFT 特征，使其成为进行图像匹配和对象识别的理想选择。

### 2.1.1 图像特征提取（SIFT）

SIFT 特征提取的主要步骤包括：

1. **构建高斯尺度空间**。对每张图像$I(x,y)$，按若干倍(通常取 4–5 个倍频)频程和每个倍频程内的尺用不同标准差的高斯核平滑：
$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$
2. **差分高斯极值检测。**相邻尺度做差分，构造 DoG 金字塔：
$$D(x,y,\sigma_{o,s}) = L(x,y,\sigma_{o,s+1}) - L(x,y,\sigma_{o,s})$$
在每个像素$(x,y,\sigma_{o,s})$处，与其在同一尺度及上下相邻尺度的 3*3*3 的邻域内比较，若为极大值或极小值，则视为初始关键点候选。
3. **关键点精确定位与剔除。**对 DoG 函数在$x+\Delta x$做泰勒展开，精确估计亚像素位置。同时剔除对比度低（$|D(x+\Delta x)| < 6$）或$\|x\|_\infty > 0.5$的点。同时剔除边缘响应，通过计算 Hessian 矩阵：
$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{D_{xx}} & \boldsymbol{D_{xy}} \\ \boldsymbol{D_{yx}} & \boldsymbol{D_{yy}} \end{pmatrix}$$
剔除位于边缘的点。
4. **方向分配。**对每个精炼后的关键点，在其所在尺度的高斯图$L(x,y,\sigma)$邻域内计算每个像素的梯度幅值$M(x,y)$与方向$\theta(x,y)$. 将这些$(M,\theta)$按方向 θ 分入 36 个等宽 bin 的直方图, 取最高峰及高于峰值 80% 的次峰，为关键点赋予主方向。
5. **生成描述子。** 在关键点位置围绕主方向对齐后，取 16*16 的领域，分为 4*4 个子块，每块计算 8-bin 梯度直方图，长度 128。得到 128 维特征向量。

具体实现于 feature_extraction.py，调用 cv2 库中的 cv2.SIFT_create 函数，同时通过 sift.detectAndCompute 可以直接得到图像关键点和描述子。

```
MAX_SIFT = 50000
sift = cv2.SIFT_create(nfeatures=MAX_SIFT)
keypoints, descriptors = sift.detectAndCompute(gray_image, None)
```

最后输出带有关键点特征的图像并展示。

### 2.1.2 图像特征匹配

图像特征匹配理论实现步骤如下：

1. **描述子距离度量。**对于两幅图像$I$和$I'$,分别提取到关键点集合及其 SIFT 描述子。度量任意一对描述子的相似度，采用欧氏距离计算：
$$\boldsymbol{dist}(\boldsymbol{d_i}, \boldsymbol{d'_j}) = \sqrt{\sum_{\boldsymbol{k=1}}^{\boldsymbol{128}} (\boldsymbol{d_i} - \boldsymbol{d'_j})^2}$$
2. **最近邻与 Ratio Test。**对每个描述子$d_i$，在$\{d'_j\}$中找到最小距离的两个邻居$j_1$和$j_2$。若第一最近邻与第二最近邻距离比小于一个设定值$\tau$，则认为$(\boldsymbol{i}, \boldsymbol{j_1})$是一个稳定匹配。通过这样的判断，可以有效剔除平坦或重复纹理区域的错误对应。

3. **基于几何约束的剔除**。即便通过 Ratio Test 仍可能有少量错误点。下一步使用对极几何约束在 RANSAC 框架下剔除外点。无畸变时，采用基础矩阵 F，满足：

$$x'^T F x = 0$$

同时计算本质矩阵 E：

$$E = K^T F K$$

其中，$x = [u, v, 1]^T$，$x' = [u', v', 1]^T$

随后计算 Sampson 距离，对每一个$(x, x')$，计算近似的几何重投影误差：

$$d_{\text{Samp}}(\mathbf{x}, \mathbf{x}') = \frac{(\mathbf{x}'^T F \mathbf{x})^2}{(F\mathbf{x})_1^2 + (F\mathbf{x})_2^2 + (F^T\mathbf{x}')_1^2 + (F^T\mathbf{x}')_2^2}.$$

若$d_{samp} < \tau_{geo}$，则视为内点。

最终筛选，只保留属于最佳模型的内点匹配对，剔除几何不一致的外点。

具体实现于 feature_matching.py 中，$\tau_{ratio}$设为 0.75，RANSAC 内点阈值设为 1.5。调用 cv2.findFundamentalMat 以直接计算得到基础矩阵 F 并剔除外点：

```
F, mask = cv2.findFundamentalMat(
    pts1_np, pts2_np, cv2.FM_RANSAC,
    ransacReprojThreshold=ransac_thresh,
    confidence=0.999
)
```

Mask 是长度为 N 的二值数组，1 代表被视为内点。置信度取 0.999。最后输出带有特征匹配的图像对并展示。

## 2.2 基于对极几何的相机位姿估计

在上一步中已经得到了经过 RANSAC 筛选后的内点匹配对：

$$\{(x_i, x_i')\}_{i=1}^M$$

将其转到归一化平面得到$\tilde{x_i} = K_i^{-1}x$，$\tilde{x_i'} = K^{-1}x_i$。接着估计本质矩阵 E，构造线性方程组满足$\tilde{x_i}E\tilde{x_i'} = 0$，将 M 条约束叠成矩阵 A，求解最小二乘$Ae = 0$，重组成 3*3 矩阵后再做奇异值约束：

$$E = U\,diag(1,1,0)V^T$$

在 RANSAC 框架下多次用估计 E，并按 Sampson 距离剔除外点，最终得到最优本质矩阵 E。

对得到的 E 做奇异值分解：

$$E = U\sum U^T,\ \sum = diag(\sigma_1, \sigma_2)$$

构造固定矩阵$W$, $Z$得到两种可能的旋转矩阵和两种平移向量方向：

$$R_1 = UWV^T,\ R_2 = UW^TV^T$$

$$t = UZU^T$$

共得四组解:

$$(R_1, t), (R_1, -t), (R_2, t), (R_2, -t)$$

再对每一组解进行 Cheirality 检验，选出物理可行解，最后输出两台相机的位姿分别为：

$$(I, 0), (R^*, t^*)$$

$(R^*, t^*)$即为选出的可行解。

具体代码实现于 init_recon.py, 先调用 cv2.findEssentialMat 函数计算基本矩阵 E：

```
E, mask = cv2.findEssentialMat(
    pts1, pts2, K, method=cv2.RANSAC, prob=0.999, threshold=1.0
)
```

再调用 cv2.recoverPose 函数将基本矩阵 E 分解为 R 和 t，同时返回一个新的二值掩码 mask_pose：

```
_, R, t, mask_pose = cv2.recoverPose(E, pts1, pts2, K, mask=mask)
```

mask_pose 表示哪些点对在分解出的$(R,t)$下满足 Cheirality 检验，是物理可行解。

然后构造 triangulate_points 函数，利用两个相机的投影矩阵和一组匹配点，做线性三角化，恢复对应的三维点坐标。首先构造两台摄像机的外参，相机 1 即是世界坐标系：

```
P1 = K @ np.hstack((np.eye(3), np.zeros((3, 1))))
P2 = K @ np.hstack((R, t))
```

然后对 pts1 和 pts2 分别做一次转置，将其作为参数和 P1,P2 一同传入：

```
pts4d = cv2.triangulatePoints(P1, P2, pts1_h, pts2_h)
pts3d = (pts4d[:3] / pts4d[3]).T
```

调用 OpenCV 的线性三角化函数 *cv2.triangulatePoints*，将其前三行$(X,Y,Z)$分别除以第四行 W，最终得到三角化后的三维点在世界坐标系下的$(x,y,z)$坐标。

## 2.3 基于 PnP 技术的场景重建

上面的步骤已经利用前两张图通过特征点匹配和对极几何求出两图的相机位姿，并用三角化方法生成了初步点云，下面介绍基于 PnP 算法批量处理余下图像的理论方法和具体实现。其核心思想分为三步：

1. 在窗口内与历史帧匹配。
2. 构造 2D–3D 对，并用 PnP-RANSAC 求位姿。
3. 与窗口内帧三角化新特征。

其中第一步的原理在上文特征匹配部分已经讲解。具体实现时，设定仅与前 6 帧做匹配，保证后面三角化时不会膨胀到整个历史集合。调用上面构造的 match_feature 函数，返回一个列表，内含所有通过 Ratio Test 和 RANSAC 基础矩阵筛选后的 2D–2D 匹配对。

第二步需构造 2D–3D 串并用 PnP 求解。首先建立相机投影模型。对第 i 帧中第 k 个观测，三维点投影至像素$(u_k,v_k)$需要满足：

$$s_k\begin{pmatrix}u_k\\v_k\\1\end{pmatrix}=K[R_i|t_i]\begin{pmatrix}X_k\\1\end{pmatrix}$$

$s_k$为比例因子。

然后需建立最小化重投影误差，理想情况下$(R_i,t_i)$由下式给出：

$$min_{R_i,t_i}\sum_{k=1}^{n}\left\|u_k-\pi\left(R_i,t_i;X_{j_k}\right)\right\|^2$$

其中，$u_k=(u_k,v_k)^T$,$X_{j_k}$是其对应的三维点。再用 EPnP 线性解算，将所有三维点表示为四个控制点的仿射组合：

$$X_k=\sum_{j=1}^{4}\alpha_k jC_j,\qquad \sum_{j=1}^{4}\alpha_k j=1$$

带入投影方程可一步解出相机位姿的近似值。

为了保证以置信度 p 至少出现一次全内点采样，需要：

$$N\geq\frac{ln(1-p)}{ln(1-(1-\varepsilon)^s)}$$

最后进行内点判定，对每个观测计算重投影误差，若误差小于一个额定值$(\tau\approx 3px)$则视为内点。

$$e_k=\left\|u_k-\pi\left(R_c,t_c;X_{j_k}\right)\right\|$$

具体实现时，调用 cv2.solvePnPRansac 函数求解：

```
ok, rvec, tvec, inl_pnp = cv2.solvePnPRansac(
    np.asarray(Pw, np.float32), np.asarray(uv, np.float32),
    K, None,
    flags=cv2.SOLVEPNP_EPNP,
    reprojectionError=3.0, confidence=0.999, iterationsCount=150
)
```

输入 objectPoints，imagePoints，相机内参矩阵，畸变参数，指定 PnP 算法，设定重投影误差阈值，置信度和最大迭代次数，通过函数便可以自动求解三维旋转向量，三维平移向量，内点索引列表以及返回是否成功找到满足内点数要求的位姿解。这一步保证了每帧相机都能在同一个世界坐标系下对齐，为后续三角化打下基础。

第三步用于增量扩充 3D 点云，利用当前帧与若干参考帧的几何关系重新三角化，生成新的 3D 点，使得稀疏点云不断增大并覆盖越来越多的场景。且只靠相邻两帧三角化可能基线很小，通过选取与当前帧视差角≥某个阈值的多帧，在更宽的阈值下进行三角化，可以提高点云的精度和鲁棒性。

给定参考帧 r 和当前帧 i 的位姿$(R_r,t_r),(R_i,t_i)$,其相对变换为：

$$R_{rel}=R_iR_r^T,\quad t_{rel}=t_i-R_{rel}t_r$$

对一对未映射的匹配$(u_r,u_i)$，构造两相机投影矩阵：

$$P_r=K\,[\,I\mid 0\,],\quad P\_i=K\,[\,R\_rel\mid t\_rel\,]$$

解得相机坐标系下的点$X_{cam}$,再变换到全局：$X_w=R_r^T(X_{cam}-t_r)$。

得到$X_w$后，分别在 r 和 i 上做重投影并计算误差：

$$\widehat{u_r}=\pi(I,0;X_{cam}),\quad \widehat{u_i}=\pi\left(R_{\{rel\}},t_{\{rel\}};X_{cam}\right)$$

$$e=\|\widehat{u_r}-u_r\|+\|\widehat{u_i}-u_i\|$$

若误差大于一个阈值$(\tau\approx 1px)$，则舍弃。两相机基线夹角由下式给出：

$$\theta=\arccos\left(\frac{\left((R_r^Te_z)\cdot\left(R_i^Te_z\right)\right)}{\left(\|R_r^Te_z\|\|R_i^Te_z\|\right)}\right)$$

具体实现时，用 calc_parallax 函数先计算两帧光轴夹角，若过小则跳过，避免基线不足导致深度不稳定。

```
if calc_parallax(R_ref, R_i) < PARALLAX_DEG:
continue
```

再由公式$R_{rel} = R_i R_r^T$，$t_{rel} = t_i - R_{rel} t_r$计算出相对位姿：

```
R_rel = R_i @ R_ref.T
t_rel = t_i - R_rel @ t_ref
```

调用 triangulate_points 函数得到相机坐标下的点$X_{cam}$，再用 world_from_cam 函数转到全局坐标$X_w$，最后追加到 pts3d。

```
X_cam = triangulate_points(R_rel, t_rel,
                           np.asarray(uv_ref), np.asarray(uv_cur), K)
X_w   = world_from_cam(X_cam, R_ref, t_ref)
```

总结而言，批量处理余下帧的处理步骤就是先从历史帧匹配相似特征，然后用已有三维点构造 2D–3D 对，并用 PnP 算法估计相机位姿，最后用新位姿和若干参考帧做三角化，不断在点云中扩充新点。

## 2.4 Bundle Adjustment

BA 用于优化重建出的三维场景，确保模型拥有更高的精度和一致性。

给定 n 台相机的外参和 m 个三维点位置：

$$(R_i, t_i)_{i=1}^{n},\ (x_j)_{j=1}^{m}$$

以及每个观测$(i,j)$对应的像素坐标$(u_{ij}, v_{ij})$，定义第 i 台相机对第 j 个点的重投影为：

$$\widehat{u_{ij}} = pi(K, R_i, t_i; X_j) = \frac{1}{z_{ij}} * \begin{pmatrix} fx & 0 & cx; \\ 0 & fy & cy \end{pmatrix} * \left( R_i X_j + t_i \right)$$

$$z_{ij} = \left( R_i X_j + t_i \right)_3$$

BA 需要解决如下最小二乘问题：

$$min_{\{\{R_i, t_i\}, \{X_j\}\}} \sum_{(i,j)} \left\| u_{ij} - \widehat{u_{ij}} \right\|^2$$

其中$(i,j)$属于所有可观测集合，求解策略一般是通过构造残差向量 r，用稀疏 Levenberg–Marquardt 算法最小化$\|r\|^2$，并根据 Schur 补高效消去三维点未知量。

具体实现于 bundle_adjustment.py 中，在 rotate 函数中用 Rodrigues 公式对每个点用对应的旋转向量 rot_vecs 进行旋转，得到每行点应用对应旋转后得到新的点。再于 project 函数得到归一化图像坐标 (N,2)，将 3D 点先旋转平移到相机坐标，再做透视除法。Fun 函数用于构造残差向量。bundle_adjustment_sparsity 函数用于构建稀疏模式矩阵 A，指示残差 $r_i$ 对哪个参数有偏导，显著加速雅可比矩阵构造与求解。bundle_adjustment 函数先把像素坐标

转到归一化平面，再与投影函数返回结果对比。调用 scipy.optimize.least_squares 函数自动求解，返回残差 res。在 3D_recon.py 中调用 bundle_adjustment 函数实现 BA。

此外，在点云渲染中还加入体素下采样，统计离群滤波和球体滤波，以进一步去除杂点。统计滤波对每个点计算它与其 **K** 个最近邻点之间的平均距离；然后基于全局所有点的平均距离分布，剔除杂点。球体裁剪基于与点云质心的距离来裁剪边缘杂散点，除去与场景距离过远的点。

全部参数设置如下：

```
PAIR_WIN          = 6          # 每帧向前可用于三角化的参考帧数
PARALLAX_DEG      = 1.0        # 最小视差角 (deg) — 小于该值不三角化
REPROJ_ERR_TH     = 1.0        # 三角化后重投影阈值 (px)
MIN_PNP_PTS       = 20         # 运行 PnP 的最小 2D-3D 对
VOXEL_SIZE        = 0.01       # 体素下采样大小 (世界单位)
STAT_NB           = 30         # 统计滤波：每点邻居数
STAT_RATIO        = 1.5        # 统计滤波：标准差倍数
SPHERE_SCALE      = 0.1        # 基于最大簇半径裁剪比率
```

## 2.5 完整流程实现

完整流程实现于 3D_recon.py 中。整个流程综合了上述讲到的所有步骤，用伪代码的形式可以表示如下：

```
for 每张图像 Ik, k = 1...N do
    提取 SIFT 关键点 Kk 和描述子 Dk
end for

匹配特征：good, pts1, pts2 ← match_features(I1, I2)
估计本质矩阵 E ← findEssentialMat(pts1, pts2, K)
R2, t2 ← recoverPose(E, pts1, pts2, K)
将内点对 {pts1_inl, pts2_inl} 三角化得到 X_cam
将相机坐标点 X_cam 转到世界坐标 X_w
初始化
    poses ← { (I,0), (R2,t2) }
    pts3d ← X_w
    kp_map ← 为每个图像保存“关键点→全局点 ID”的映射

for i = 3 to N do
    for ref = max(1, i–PAIR_WIN) to i–1 do
        matches_dict[ref,i] ← match_features(I_ref, I_i)
    end for

    if |(Pw,uv)| < MIN_PNP_PTS then
        poses[i] ← poses[i–1]
        continue
```

```
        end if
        ok, rvec, tvec, inl ← solvePnPRansac(Pw, uv, K)
        R_i ← Rodrigues(rvec);   t_i ← tvec
        poses[i] ← (R_i, t_i)

        for ref = max(1, i–PAIR_WIN) to i–1 do
            if calc_parallax(poses[ref].R, R_i) < PARALLAX_DEG then
                continue
            end if
            收集在 ref 与 i 中均未映射的 matches → (uv_ref, uv_i)
            if |uv_ref| < MIN_TRI_MATCH then
                continue
            end if
            R_rel ← R_i · R_ref^T;   t_rel ← t_i – R_rel·t_ref
            X_cam_new ← triangulate_points(R_rel, t_rel, uv_ref, uv_i, K)
            X_w_new, keep ← filter_triangulation(...)
            将 X_w_new[keep] 加入 pts3d，更新 kp_map[ref,i] 的点 ID
        end for
    end for

    构造 BA 优化变量：
        相机位姿 { (R1,t1), ..., (R_N,t_N) }
        三维点集 { X1, ..., X_M }
    定义重投影误差函数
        E(poses, pts3d) = Σ_{i,j} ‖x_{ij} – π(K,[R_i|t_i];X_j)‖²
    求解
        (poses*, pts3d*) ← argmin E(poses, pts3d)

    构建 Open3D 点云 pc ← pts3d*
    下采样与裁剪
    保存 pc 为 PLY，并可视化
```

同时，最后返回以最后一张图片为世界坐标系的 62 行，每行 16 个数（外参矩阵拉平）的外参文件，首先提取最后一张图片的旋转和平移分量$R_{base}$和$t_{base}$作为世界坐标系的基准。首先写出第一行的外参为单位阵，对余下 61 张图片做相应的变换：

$$R_{rel} = R_i R_{base}^T$$
$$t_{rel} = t_i - R_{rel} t_{base}$$

最后全部写入文件并导出。

# 第三章 结果

下面按步骤分别展示结果。

## 3.1 SIFT 关键点提取

由于篇幅限制，仅以一张处理结果为例展示：


## 3.2 关键点匹配

所有结果存于 matches 文件夹中。下图为图 1 和图 2 的 SIFT 特征点匹配结果。可以看出过滤效果较好，误匹配的情况很少。

## 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）为：

$$R = \begin{pmatrix} 0.99939 & -0.00663 & 0.03424 \\ 0.00686 & 0.99995 & -0.00677 \\ -0.03419 & 0.00700 & 0.99939 \end{pmatrix}, \quad t = \begin{pmatrix} -0.92938 \\ 0.21482 \\ -0.30015 \end{pmatrix}$$

代码位于 init_recon.py 中。

### 3.4 PnP 初始重建结果

运行 PnP_recon.py 文件，会自动将结果保存至 model_pnp.ply 中，并弹出 Open3D 预览窗口，初步重建结果（不加入 BA）为：

点云总数为 302740。

### 3.5 最终重建结果

运行整体流程 3D_recon.py，在上述流程中加入 BA 优化，将结果自动保存至 model.ply 中，同时按要求以最后一张图（DJI_20200223_163225_243.jpg）的相机坐标系为世界坐标系输出相机外参文件 camera_extrinsics.txt。

最终重建结果如下：

| Iteration | Total nfev | Cost | Cost reduction | Step norm | Optimality |
|---|---|---|---|---|---|
| 0 | 1 | 1.8426e+04 | | | 4.13e+06 |
| 1 | 3 | 1.6201e+04 | 2.22e+03 | 3.92e+02 | 8.79e+05 |
| 2 | 5 | 1.5205e+04 | 9.96e+02 | 1.57e+02 | 8.04e+05 |
| 3 | 6 | 1.4324e+04 | 8.81e+02 | 3.14e+02 | 8.63e+05 |
| 4 | 8 | 1.3893e+04 | 4.31e+02 | 1.54e+02 | 5.20e+05 |
| 5 | 9 | 1.3627e+04 | 2.66e+02 | 3.43e+02 | 4.21e+05 |
| 6 | 10 | 1.3474e+04 | 1.54e+02 | 3.57e+02 | 1.70e+05 |
| 7 | 11 | 1.3459e+04 | 1.46e+01 | 1.19e+02 | 5.35e+04 |
| 8 | 12 | 1.3454e+04 | 5.10e+00 | 1.22e+02 | 2.19e+04 |
| 9 | 13 | 1.3451e+04 | 2.82e+00 | 5.06e+01 | 1.36e+04 |
| 10 | 14 | 1.3450e+04 | 1.45e+00 | 4.91e+01 | 5.71e+03 |
| 11 | 15 | 1.3448e+04 | 1.14e+00 | 3.27e+01 | 6.25e+03 |

从 BA 处理日志可以看到，迭代 11 次，最初几次 cost 有显著下降，到第八次左右 cost 基本维持在 1.345e+04 的水平，可见 BA 有效减少了重投影误差。

同时返回了相机外参文件 camera_extrinsics.txt。

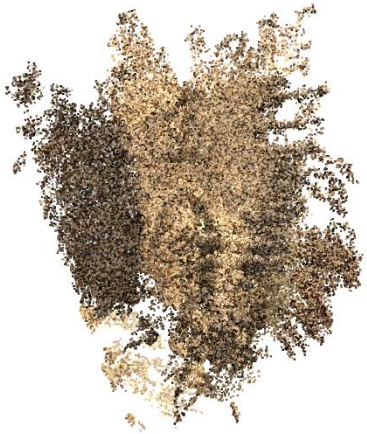
# 第四章 讨论

下面进行消融实验部分，将着重讨论各个环节的作用：

1. 只用对极几何重建 Pose
2. 不加入 BA 和加入 BA 的区别。
3. 图像数量影响和原因分析。

## 4.1 只用对极几何重建 Pose

修改 PnP_recon.py，去除其中 PnP 部分，增量处理采用连续帧 Essential→recoverPose→三角化方法，得到重建结果：



可见重建效果很差，噪声明显，且点云扩散严重。主要原因是每一帧的相对位姿都是通过上一帧到这一帧的本质矩阵分解为 R 和 t 得到的，因此任何一次恢复都有噪声。后面三角化又把这些小误差当作正确的几何约束去构建新点，误差因此不断叠加。由此可见，没有 PnP 就无法把新帧与整个全局点云做 2D–3D 对应来修正位姿，只有局部的两帧对。没有 Bundle Adjustment 则重投影误差不能被最小化调整。且连续帧之间基线往往很小，因此建模呈现深度不稳定的现象。

## 4.2 不加入 BA 和加入 BA 对比

在上述方法下加入 BA，得到无 PnP 但有 BA 优化的结果：

从结果上来看，相较于上面没有 BA 优化的结果，中心部分点云更加紧凑，但仍然总体而言效果较差。从 BA 日志上可以看到迭代次数为 102 次，cost 水平从 4.3929e+08 下降至 8.8641e+06，虽然已有显著降低，但和之前的 1.345e+04 相比仍过大。这可能是因为在没有 PnP 的情况下，初始位姿漂移依然很严重，每一帧的位姿都是基于与前一帧的一次性 RANSAC-Essential 估计，没有全局约束。BA 虽能均衡误差，但其优化对象仅有基于稀疏、漂移很大的初始位姿，并不能消除累计误差。因此整体点云依然没有呈现一个清晰的形状。

## 4.3 图像数量影响

当输入的图像数量发生变化时，重建结果也必然会受到影响，尤其是点云数量上。下面将讨论重建前 20 张图片重建和前 40 张图片重建的结果并分析。（均不加入 BA）

以 PnP_recon.py 为模板脚本，修改为重建前 20 张图片，结果如下：

重建前 40 张图片，结果如下：

使用全部图像重建结果：

：

图像数量逐步增多的过程可以看出，重建 20 张图时点云较为稀疏，且主要集中在视角最集中的区域。由于图像数量较少，意味着很多场景细节未能覆盖。重建 40 张图时覆盖了更多视角，点云在更多方向上都有新的三角化点，且点云更密集，细节更丰富，但也带来了更多杂点。具体原因在于，每增加一张图，都会在它和之前已定位帧的匹配对上三角化出新点，因而覆盖范围扩大，点云数量更多，细节更丰富。但同时随着图像数量增多，匹配数量和错误累积都上升，导致散点更多，需要加入相应的过滤等。因此若想取得场景丰富性和噪声的平衡，需要加入 BA，并加入更加严格的滤波和关键帧处理。