机器视觉课程设计报告

成员

云川 李云川 2020212234

文卿 周文卿 2020212248

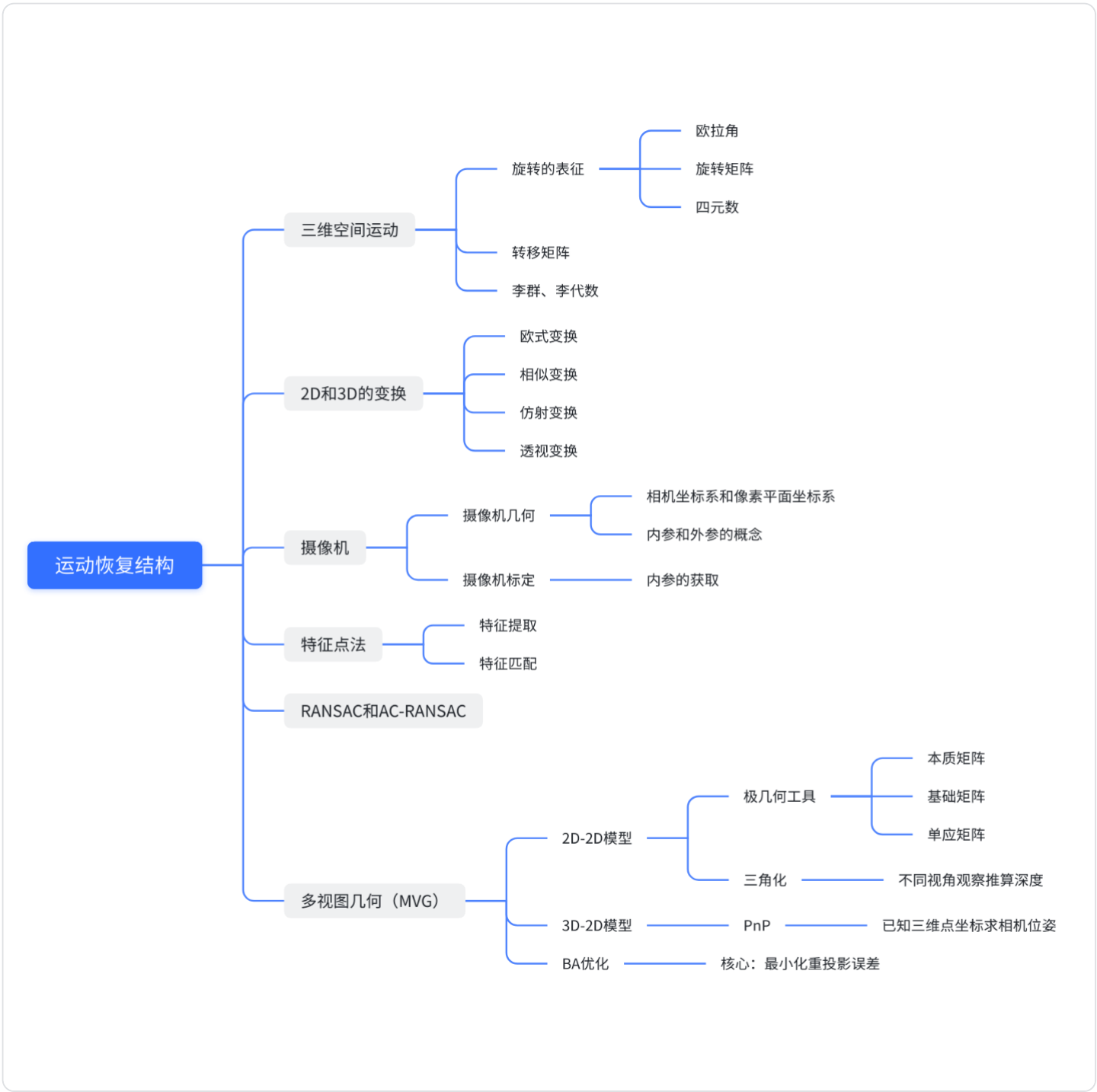
玮俊 罗玮俊 2020212236

项目简介

实现一个**运动恢复结构系统：**假设摄像机内参数已知，构建一套三维重建系统，界面输入多张图像，界面展示重建点云。

任务初步分解

可先从“输入--内部处理--输出”范畴认识这个系统：输入是一组图像（或一段视频）和对应的内参数据，它们既可以来自现有的数据集，也可以自行拍摄和标定得到；输出是点云文件或者一个点云显示窗口；而内部处理需要运用相关的机器视觉理论知识进行进一步的详细的拆分和设计。



详细设计

工程的明确定位

根据课设要求和论文《Adaptive structure from motion with a contrario model estimation》的内容，本次课设我们需要实现的基础部分是一个增量式SFM系统，并在此基础上实现AC-RANSAC以达成自适应阈值功能。选用增量式SFM系统给的主要原因是，目前主流的SFM（Structure from Motion，运动结构恢复）可以分为两大类型，一种是全局式的，一种是增量式的。全局式（Global）能够一次性得出所有的相机姿态和场景点结构并且全局式SFM只需要在最后进行一次BA（Bundle Adjustment），因此效率较高，但是其鲁棒性差，很容易受到outlier的影响而导致重建失败。增量式

（Incremental）SFM则是一边三角化（triangulation）和pnp（perspective-n-points），一边进行局部BA。这类方法在每次添加图像后都要进行一次BA，效率较低，而且由于误差累积，容易出现漂移问题；但是增量式sfm的鲁棒性较高。

数据集介绍

本次任务当中，根据所需要完成的项目需求，我们选取了如下的一些数据集。包括了一些基督教建筑、雕塑和行人等，但是所使用的数据中又不仅仅包含这些类(此处不再一一列举)。每个类别的数据集合中包含了用于三维重构输入的图片、以及一系列的相机参数文件，每张图片都对应一个相机参数文件，每个camera参数文件都有形如下面格式的数据：

```
2759.48 0 1520.69
0 2764.16 1006.81
0 0 1
0 0 0
0.14139 0.153155 0.978035
0.989608 -0.0479961 -0.135547
0.0261821 0.987036 -0.15835
-17.6081 -3.12802 0.014307
3072 2048
```

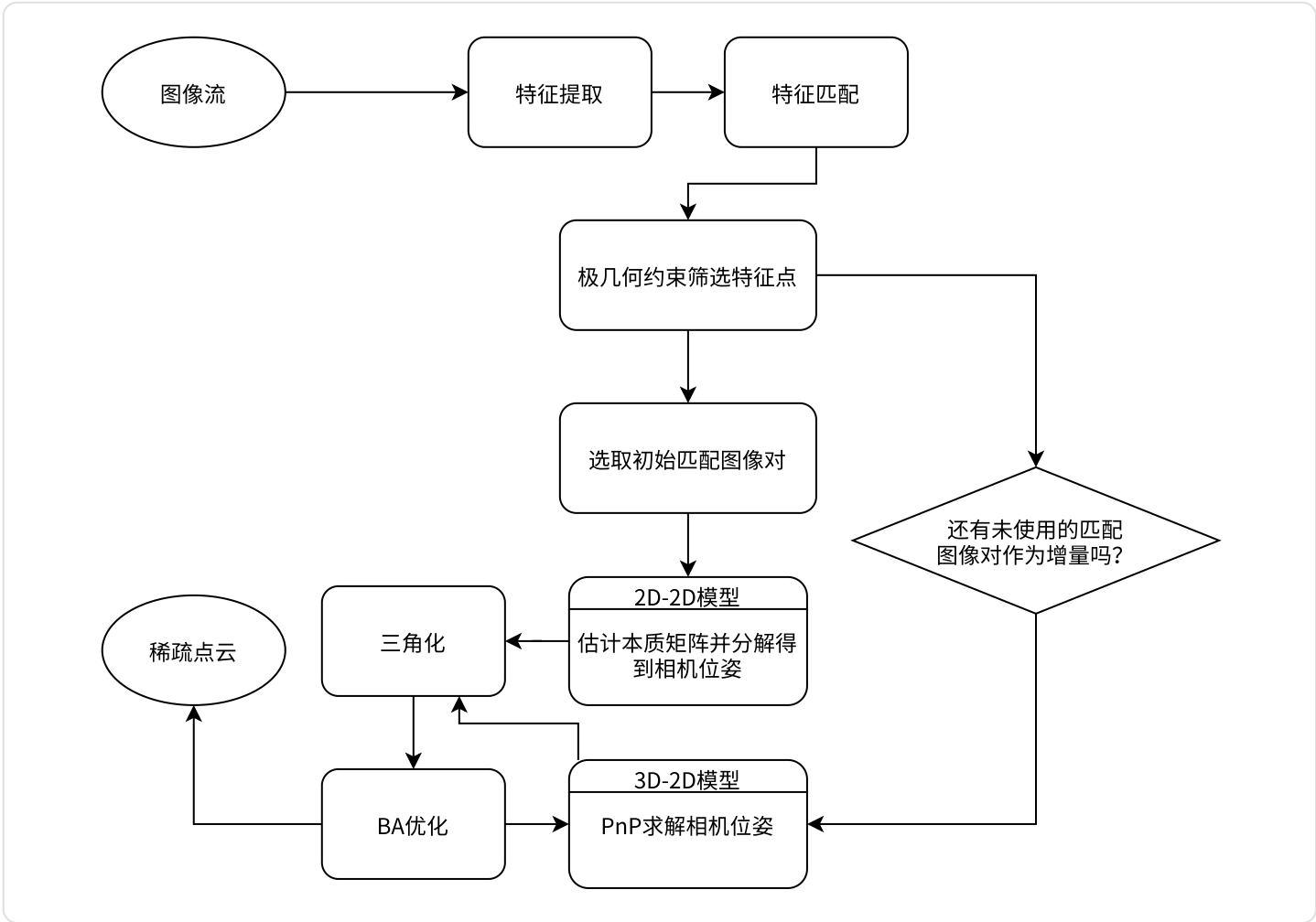
这些参数分别代表着如下的含义：

1. 内部参数（Intrinsic Parameters）
   - f_x , f_y : 相机的焦距，表示相机的水平和垂直像素尺度。
   - c_x , c_x : 相机的光学中心，表示图像平面上的原点位置。
2. 畸变参数（Distortion Parameters）
   - k_1 , k_2 , k_3 : 径向畸变参数，用于描述透视相机镜头的径向畸变。
   - p_1 , p_2 : 切向畸变参数，用于描述透视相机镜头的切向畸变。
3. 外部参数（Extrinsic Parameters）
   - r_{11} , r_{12} , r_{13} : 相机的旋转矩阵的第一行。
   - r_{21} , r_{22} , r_{23} : 相机的旋转矩阵的第二行。
   - r_{31} , r_{32} , r_{33} : 相机的旋转矩阵的第三行。
   - t_1 , t_2 , t_3 : 相机的平移向量，表示相机的位置。
4. 图像尺寸
   - width: 图像的宽度（像素）。
   - height: 图像的高度（像素）。

算法流程和细节说明

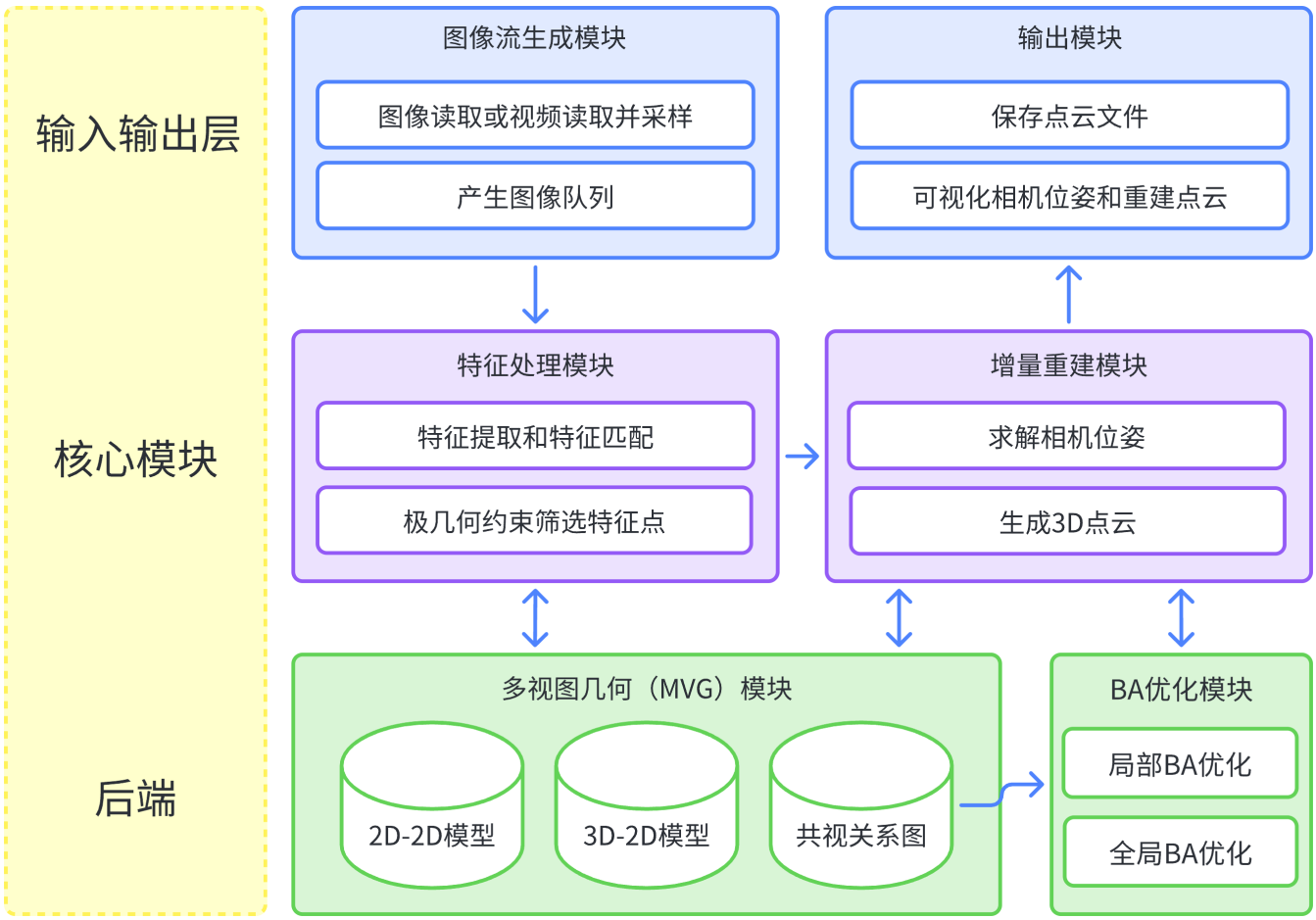
算法的概要流程图

以增量式运动恢复结构系统（the Incremental Structure from Motion System）的常见算法为基础，我们组设计了如下核心算法概要流程图：



代码实现

整体代码模块设计



模块简介

本次代码按照开源式框架进行编写，主要包括数据预处理、特征提取和匹配、点云生成和优化、以及可视化和评估等模块。这些模块共同工作，以实现对三维场景的重建和表示。

首先，我们有数据预处理模块`stream.py`，负责对输入的图像或视频数据进行处理和准备。这包括图像去噪、图像对齐、相机标定等步骤，以提高后续处理的精确性和稳定性。

其次，特征提取和匹配模块是三维重建的核心步骤之一。在这个模块中，我们使用了最为熟悉的计算机视觉算法，如SIFT特征匹配，从图像序列中提取出特征点，并通过特征匹配算法来计算相机之间的对应关系。这为后续的三维点云生成提供了关键的输入。

接下来，是点云生成模块，它将通过特征匹配和相机参数估计，将二维图像转换为三维点云。这个过程可以通过三角测量、立体视觉或者深度学习方法来实现。生成的点云数据将被用于表示物体的三维形状和结构。

最后，可视化和评估模块对生成的点云进行可视化展示和性能评估。通过可视化，我们可以直观地观察点云的形状和结构，并对结果进行直观的判断。评估模块可以通过与标准答案比对或其他定量指标来评估重建结果的准确性和精度。

```
1 CloudView.py -- 提供了用于可视化点云数据的功能。
2 config.py -- 项目的配置信息和参数，这些参数包含图像的尺寸、特征匹配所需的最小匹配
3              数量、特征检测过程中的各种阈值和参数设置等。
```

```
context.py -- 用于处理不同版本的opencv，主要目的是兼容不同版本的OpenCV库，针对不同
              的版本进行适配和处理。
feature.py -- 特征提取，模块中提供了重塑图片、提取sift特征等相关函数.
graph.py -- 提供了运动恢复结构中至关重要的两个图结构类，主要是用于在三维重建中处理特
            征和视觉连通性。
Initializer.py -- 三维重建的初始化，为后续的相机位姿估计和三维点云增量化建立初始基础
PySBA.py -- 实现了一个简单的 Bundle Adjustment 算法，可用于对相机位姿和三维点进行联
            合优化，以提高点云重建的精度和鲁棒性。
reconstruction.py -- 增量重建
sfm_main.py -- 实现了 SfM 流程的主要逻辑，包括特征提取、共特征点图构建、共视关系图构建、
               增量式重建等步骤，用于将多张图片转换为三维点云模型。
stream.py -- 实现了图像和视频数据的读取和处理功能，用于将图像和视频数据转换为神经网络的
             输入格式。
utils.py -- 实现了将点云和颜色保存为PLY文件的功能,可以被多种三维软件读取和处理。
webui.py -- 实现了前端的构建，获得可供用户交互和展示3维图像的界面
```

模块详解

在这一部分，我们将详细介绍CloudView、feature、graph、PySBA、reconstruction和sfm_main等模块。所介绍的模块都会是整个运动恢复结构中最重要的模块部分。

CloudView模块

在最终的UI中会有点云显示，不必启用此模块。如果从demo.py运行程序或正在进行debug建议启用。此模块在CloudView模块中, 定义了一个名为 `CloudView` 的类，该类用于管理点云和相机位姿的可视化，并提供了更新点云和相机姿态的方法。在初始化时，根据配置信息和图像大小创建了相应的属性。其主要的函数功能如下所示：

1. `__init__(self, cfg: Dict[str, Any])`：类的构造函数。接受一个字典类型的配置信息 `cfg`，进行图像的高度 `ht` 和宽度 `wd`，以及用于指示是否启用可视化布尔值 `viz` 等的设置。
2. `start_viewer(self)`：启动可视化窗口。
   - 该函数使用之前定义的 `image_`、`poses_`、`points_` 和 `colors_` 等属性创建一个 `Viewer` 对象，用于显示图像、相机姿态、点云和颜色。
3. `update(self, image: np.ndarray, pose: np.ndarray, intrinsics: np.ndarray, points: np.ndarray, colors: np.ndarray)`：更新可视化窗口的内容。
   - `image`：输入的图像数组。
   - `pose`：相机的姿态（位姿）数组。
   - `intrinsics`：相机的内参矩阵数组。
   - `points`：点云的坐标数组。
   - `colors`：点云的颜色数组。

feature模块

在本模块中，实现了一个SIFT特征匹配器，用于在图像中检测和匹配特征点。模块提供了一些类以及函数接口，包括提取SIFT特征、重塑图片但不限于这些接口函数。

1. `FeaturesData` 类

- 功能：存储特征数据，包括特征点、特征描述符和颜色信息。
- 属性：
  - `points`：特征点的坐标，类型为 `np.ndarray`。
  - `descriptors`：特征描述符，类型为 `np.ndarray`，可选参数。
  - `colors`：特征点对应的颜色信息，类型为 `np.ndarray`。
- 方法：
  - `mask(mask: np.ndarray) -> "FeaturesData"`：根据掩码对特征数据进行过滤，返回新的 `FeaturesData` 对象。

类: MatchedFeatures

- 功能：存储匹配的特征数据，包括特征点、特征描述符、颜色信息和索引。
- 继承：继承自 `FeaturesData` 类。
- 属性：
  - `index`：特征点的索引列表，类型为 `List[int]`。
- 方法：
  - `mask(mask: np.ndarray) -> "MatchedFeatures"`：根据掩码对匹配特征数据进行过滤，返回新的 `MatchedFeatures` 对象。

`resized_image(image: np.ndarray, max_size: int)` 函数

- 功能：调整图像尺寸，使其最大边的长度等于 `max_size`。
- 参数：
  - `image`：输入的图像，类型为 `np.ndarray`。
  - `max_size`：目标尺寸，类型为 `int`。
- 返回值：调整尺寸后的图像，类型为 `np.ndarray`。

函数 `root_feature(desc: np.ndarray, l2_normalization: bool = False) -> np.ndarray`

- 功能：对特征描述符进行根归一化。
- 参数：
  - `desc`：输入的特征描述符，类型为 `np.ndarray`。

    - `l2_normalization`：是否进行L2归一化，类型为 `bool`，默认为 `False`。
- 返回值：根归一化后的特征描述符，类型为 `np.ndarray`。

函数: extract_features(image: np.ndarray, config: Dict[str, Any], is_panorama: bool) -> FeaturesData

- 功能：提取图像的特征数据。
- 参数：
    - `image`：输入的图像，类型为 `np.ndarray`。
    - `config`：配置参数字典，类型为 `Dict[str, Any]`。
    - `is_panorama`：是否为全景图像，类型为 `bool`。
- 返回值：提取的特征数据，类型为 `FeaturesData`。

函数 `extract_features_sift(image: np.ndarray, config: Dict[str, Any], features_count: int) -> Tuple[np.ndarray, np.ndarray]`

- 功能：使用SIFT算法提取图像的特征点和特征描述符。
- 参数：
    - `image`：输入的图像，类型为 `np.ndarray`。
    - `config`：配置参数字典，类型为 `Dict[str, Any]`。
    - `features_count`：提取的特征点数量，类型为 `int`。
- 返回值：提取的特征点和特征描述符，类型为元组 `(np.ndarray, np.ndarray)`。

类: `SIFTmatcher`

- 功能：进行SIFT特征点匹配。
- 属性：
    - `matcher`：SIFT特征点匹配器，类型为 `cv2.BFMatcher`。
- 方法：
    - `match_by_images(cfg: Dict[str, Any], image1: np.ndarray, image2: np.ndarray) -> Tuple[MatchedFeatures, MatchedFeatures]`：
        - 功能：根据图像进行特征点匹配。
        - 参数：
            - `cfg`：配置参数字典，类型为 `Dict[str, Any]`。
            - `image1`：图像1，类型为 `np.ndarray`。
            - `image2`：图像2，类型为 `np.ndarray`。
        - 返回值：匹配的特征数据，类型为元组 `(MatchedFeatures, MatchedFeatures)`。

- `match_by_features(cfg: Dict[str, Any], feature1: FeaturesData, feature2: FeaturesData) -> Tuple[MatchedFeatures, MatchedFeatures]`:
  - 功能：根据特征数据进行特征点匹配。
  - 参数：
    - `cfg`：配置参数字典，类型为 `Dict[str, Any]`。
    - `feature1`：特征数据1，类型为 `FeaturesData`。
    - `feature2`：特征数据2，类型为 `FeaturesData`。
  - 返回值：匹配的特征数据，类型为元组 `(MatchedFeatures, MatchedFeatures)`。
- `search_by_Initialization(cfg: Dict[str, Any], feature1: FeaturesData, feature2: FeaturesData, K: np.ndarray) -> Tuple[MatchedFeatures, MatchedFeatures, np.ndarray]`:
  - 功能：基于初始化信息进行特征点匹配。
  - 参数：
    - `cfg`：配置参数字典，类型为 `Dict[str, Any]`。
    - `feature1`：特征数据1，类型为 `FeaturesData`。
    - `feature2`：特征数据2，类型为 `FeaturesData`。
    - `K`：相机内参矩阵，类型为 `np.ndarray`。
  - 返回值：匹配的特征数据、基础矩阵和掩码，类型为元组 `(MatchedFeatures, MatchedFeatures, np.ndarray)`。

---

graph模块

`graph` 模块在运动恢复结构三维重建中的主要作用是管理特征点和视觉信息，并构建图形表示来表示相机之间的关系和特征点之间的匹配。它提供了特征提取、特征匹配、共视图和特征图的构建、节点和边的管理以及图形可视化等功能。通过这些功能，该模块可以帮助恢复相机的轨迹和场景的三维结构，并在三维重建过程中进行优化和更新。

1. `CoFeaturesNode` 类：表示特征图中的节点。具有以下属性：
   - `node_id`：节点的唯一标识符。
   - `image`：节点对应的图像数据。
   - `intrinsics`：节点对应图像的内参矩阵。
   - `all_features`：节点包含的所有特征数据。
2. `CoFeaturesEdge` 类：表示特征图中的边。具有以下属性：
   - `node_id1` 和 `node_id2`：连接的两个节点的标识符。

- `feature1` 和 `feature2`：两个节点之间匹配的特征数据。
- `weight`：边的权重。
- `EssentialMatrix`：基础矩阵。

3. `CoFeaturesGraph` 类：继承自 `networkx.Graph`，表示特征图。具有以下功能和属性：
   - 添加节点和边：可以添加节点和边，并保存相应的特征数据和权重信息。
   - 查询节点和边：可以获取特定节点和边的详细信息。
   - 获取邻居节点和边：可以获取指定节点的所有邻居节点和边的列表。
   - 获取最大权重边：可以获取具有最大权重的边。
   - 重置图：可以清除所有节点和边，并重置图的状态。
   - 绘制图：可以绘制图形表示。
4. `CovisibilityNode` 类：表示共视图中的节点。具有以下属性：
   - `node_id`：节点的唯一标识符。
   - `image`：节点对应的图像数据。
   - `intrinsics`：节点对应图像的内参矩阵。
   - `all_features`：节点包含的所有特征数据。
   - `pose`：节点的姿态信息。
5. `CovisibilityEdge` 类：表示共视图中的边。具有以下属性：
   - `node_id1` 和 `node_id2`：连接的两个节点的标识符。
   - `feature1` 和 `feature2`：两个节点之间匹配的特征数据。
   - `cloud`：表示边连接的点云数据。
   - `weight`：边的权重。
6. `CovisibilityGraph` 类：继承自 `networkx.Graph`，表示共视图。具有类似于 `CoFeaturesGraph` 的功能，用于添加节点和边、查询节点和边、获取邻居节点和边、获取最大权重边、重置图和绘制图。
7. `_co_features_graph`：`CoFeaturesGraph` 类的实例，用于保存特征图。
8. `_covisibility_graph`：`CovisibilityGraph` 类的实例，用于保存共视图。

PySBA模块

这个模块来自于github：https://github.com/jahdiel/pySBA。

这段代码实现了一个快速的稀疏捆绑优化Bundle Adjustment）算法，用于优化相机参数和三维点的估计值。下面是代码的主要部分的解释：

- `PySBA` 类是一个用于简单束调整的Python类，其中包含了初始化函数和一些辅助函数。
- `fun` 函数计算残差（residuals）。输入参数 `params` 包含相机参数和三维点坐标，该函数根据这些参数使用cv2计算投影点并与观测到的二维坐标进行比较，返回残差。
- `bundle_adjustment_sparsity` 函数生成稀疏矩阵A，用于优化过程中的雅可比矩阵的计算。
- `optimizedParams` 函数从优化后的参数中提取相机参数和三维点坐标。
- `bundleAdjust` 函数是主要的调用函数。它首先根据初始参数调用 `fun` 函数计算初始残差。然后使用 `bundle_adjustment_sparsity` 生成稀疏矩阵A。最后，使用 `least_squares` 函数进行优化，得到优化后的相机参数和三维点坐标。

总体来说，这段代码实现了一个基于最小二乘优化的简单束调整算法，用于优化相机参数和三维点的估计值，以使它们与观测到的二维坐标更加吻合。

---

reconstruction模块

这个模块实现了增量重建（incremental reconstruction）的功能。

1. `incremental_reconstruction(cloud_viewer: CloudView, cfg: Dict[str, Any]) -> Tuple[np.ndarray, np.ndarray]`：这个函数是整个模块的入口函数也是整个模块唯一的接口函数，它实现了增量重建的主要逻辑。它接受一个 `CloudView` 对象作为输入，以及一些配置参数 `cfg`。根据传入的参数和初始化的图结构，它逐步进行增量重建，生成点云和颜色信息。最后，它返回生成的点云和颜色数组。函数的主要迭代步骤如下所示：
   a. 导入配置参数并初始化一些变量和数据结构。
   b. **根据参数选择初始化函数进行初始化，得到初始的图像ID、点云和颜色数据。根据是否启用更鲁棒重建，函数会采用不同的初始化策略：如果启用，则计算共特征点图中匹配点最多的一对图像用以初始化；否则直接使用第一、二幅图像进行初始化。**
   c. 将初始点云和颜色数据添加到总的点云和颜色数组中，并记录已使用的节点。
   d. 当存在待处理的边时，执行以下循环：
   e. 获取与已使用节点相邻的节点和边的信息。
   f. **从相邻的边中选择与已重建对应点最多的边作为基准边和待增量重建的边，这涉及计算待增量重建的边的匹配特征点对和基准边重建的点云的索引匹配；**
   g. 求解PnP问题，通过三角化计算待增量重建边对应的点云，并更新相机位姿。
   h. 根据参数选择是否进行局部BA优化，如果是，则执行局部BA优化，优化相机参数和点云数据。
   i. 将生成的点云和颜色数据添加到总的点云和颜色数组中，并更新使用的节点和边的关系。
   j. 更新点云视图，并输出日志信息。
   k. 返回最终生成的总的点云和颜色数组。

sfm_main模块

模块定义了一个名为 `sfm_runner` 的类，用于运行SfM算法。模块中会根据是否使用鲁棒模式，通过匹配算法搜索两个节点的共视特征点。在鲁棒模式下，会遍历所有节点对，并计算每对节点之间的共视特征点；在非鲁棒模式下，只会计算每个节点与之后节点之间的共视特征点。共视特征点数量小于一定阈值的节点对会被忽略。

然后，将共特征点图传递给增量式重建函数 `incremental_reconstruction` ，执行增量式重建过程。该函数会逐步处理图像和特征点，生成三维点云。同时，会根据参数选择是否进行局部BA优化。

在增量式重建完成后，将生成的点云和颜色数据保存到文件，并可视化共视关系图。最后，如果云视图对象存在，将等待可视化窗口的关闭。

整个模块的作用是从队列中获取图像数据，执行增量式重建算法，并可视化重建结果。通过该模块，可以实现自动化的三维重建过程。

其具体实现如下所示：

1. `sfm_runner` 类：
    - `__init__(self, queue: Queue, cfg: Dict[str, Any])` ：初始化函数，接受一个队列和配置字典作为参数，并初始化 `CloudView` 对象。
    - `__call__(self) -> Tuple[np.ndarray, np.ndarray]` ：主要的执行函数，将被调用来执行SFM流程。
    - `reset(self)` ：重置函数，当前为空。
2. `__call__(self) -> Tuple[np.ndarray, np.ndarray]` 函数的执行流程：
    - 进入无限循环，直到从队列中获取到结束标志为止。
    - 从队列中获取图像、帧数和内参。
    - 如果帧数小于0，表示图像序列结束，退出循环。
    - 调用 `extract_features` 函数提取图像的特征点。
    - 向共特征点图和共视关系图中添加节点，节点包含图像、内参和特征点。
    - 检查共特征点图中的节点数量是否小于2，如果是则抛出异常。
    - 获取共特征点图中的节点列表。
    - 如果配置中设置了鲁棒重建，则进行鲁棒匹配：
        - 对于每一对节点，通过SIFT匹配器计算它们的共视特征点。
        - 如果共视特征点数量小于配置中的最小匹配点数要求，则跳过。
        - 向共特征点图中添加边，包含特征点、权重和本质矩阵。
    - 如果没有设置鲁棒重建，则进行逐帧匹配：

        - 对于每一对相邻的节点，通过SIFT匹配器计算它们的共视特征点。
        - 如果共视特征点数量小于配置中的最小匹配点数要求，则跳过。
        - 向共特征点图中添加边，包含特征点、权重和本质矩阵。
    - 增量式重建：
        - 调用 `incremental_reconstruction` 函数进行增量式重建，并传入 `CloudView` 对象和配置字典。
        - 输出增量式重建的时间。
    - 如果配置中设置了保存重建结果，则调用 `save_ply` 函数保存点云模型。
    - 输出完成的消息。
    - 如果存在 `CloudView` 对象的可视化器，等待可视化器关闭。
    - 返回总的点云和颜色数据。

---

stream模块

stream主要是用于进行数据预处理，针对输入的图像(视频)进行预处理，生成序列，方便与后续的三维重构任务进行。

1. `image_stream(cfg, queue)`：这个函数是一个图像生成器，用于生成图像序列。它接受一个队列作为参数，将生成的图像放入队列中。该函数的作用如下：
    - 从配置文件中读取图像目录、相机标定参数、步长、跳过帧数等信息。
    - 加载相机标定参数，并计算相机的内参矩阵K。
    - 构建图像扩展名列表，根据图像目录和扩展名列表获取图像文件列表。
    - 遍历图像文件列表，读取每个图像文件。
    - 如果相机标定参数包含畸变参数，对图像进行去畸变处理。
    - 根据配置文件中的图像下采样比例，对图像进行下采样。
    - 将图像、帧数和内参放入队列中，循环直到遍历完所有图像文件。
    - 最后将帧数设为-1，表示图像序列的结束。
2. `ros_topic_stream` 类：实现订阅ros的相机话题和相机参数话题，同时在UI中进行ros图像的截取，输入SFM系统，实现实时连接相机进行图像获取并重建。
3. `img_downscale(img, downscale)`：这个函数用于对图像进行下采样。它接受一个图像和下采样比例作为参数，并返回下采样后的图像。该函数的作用如下：
    - 根据指定的下采样次数，多次调用 `cv2.pyrDown` 函数对图像进行（二取一）下采样。
    - 返回下采样后的图像。

这些函数一起构成了一个图像或视频生成器的模块，用于生成用于三维重建任务的图像序列或视频序列。这些生成器将生成的图像或视频帧放入队列中，以供后续的三维重建算法使用。

---

webui模块

`webui` 模块主要使用 `gradio` 进行前端的架构，获得一个可供用户进行交互的简单界面。

1. 文件夹位置输入：由于 `gradio` 中没有针对文件夹的输入选项，选择使用多个 `Textbox` 来获得文本输入作为文件夹的位置进行之后的三维重建。
2. 参数控制：使用到 `Checkbox` ， `Number` ， `Slider` 获得参数的输入，对不同的参数使用不同的方法。
3. 三维重建：使用前端获得的参数进行三维重建获得三维点云模型。
4. 点云可视化： `gradio` 的输出中使用 `Plot` 类型，可以获得三维点云的展示。
    a. 借助 `pyvista` 打开算法获得 `ply` 文件，在测试中 `pyvista` 借助 `Plotter` 可以实现可视化网格数据，测试过程中考虑使用 `pyvista` 的弹出窗口来展现点云图像，但希望可以在同一个界面获得展示。
    b. 经过探索，选择了 `gradio` 自带的输出类型 `Plot` 来展示点云文件，由于UI界面展示的三维图像的特殊性，借助 `plotly.graph_objs` 下的 `feature` 和 `Scatter3d` 来对读取到的点云文件的位置和颜色信息进行绘制点云图像，以获得 `UI` 界面交互式的3D图像，从而展示出算法的三维重建结果。

使用 `Blocks` 构建界面，除了 `Slider` 的输入，其他的输入每四个为一行， `launch` 获得界面展示，我们在界面中输入相应的参数，点击 `submit` 后传入后端，进行三维重建过程之后得到界面下方的输出点云展示。

在上面的部分中，我们着重介绍了一些比较重要的模块，而对其他一些模块没有进行详细介绍的原因是因为我们希望聚焦于系统中最为关键和核心的部分。在一个复杂的轻量级三维重建系统中，通常会包含多个模块，每个模块都有其独特的功能和作用。

然而，为了确保叙述的清晰度和可读性，我们选择着重介绍那些对系统整体性能和结果产生最直接影响的模块。这些模块通常是构成三维重建流程的核心环节，涉及到关键的计算和算法。

在这种情况下，我们选择介绍数据预处理模块、特征提取和匹配模块、点云生成模块以及点云滤波和优化模块。这些模块涵盖了从图像数据处理到三维点云生成的关键步骤，对于系统的整体性能和结果产生直接影响。

同时，我们意识到在一个完整的三维重建系统中，还可能存在其他一些模块，比如相机参数估计、纹理映射、模型配准等等。这些模块同样具有重要的作用，但由于篇幅限制和为了突出重点，我们在此未对其进行详细介绍。

请注意，对于一个轻量级的三维重建系统，往往需要在设计上进行一定的取舍和权衡，以达到较好的性能与效果。因此，我们着重介绍了那些最为核心和关键的模块，以期提供一个全面而又简明的概览。

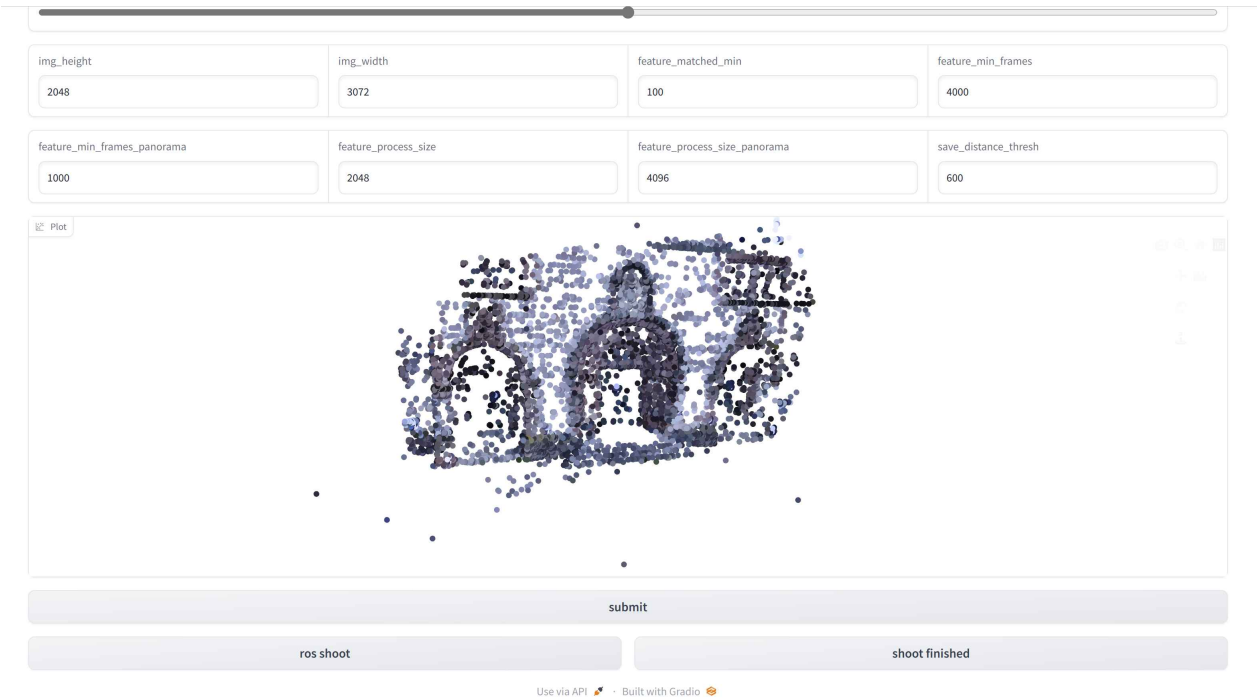
总结而言，我们选择介绍了一些比较重要的模块，以强调它们对整个系统的关键性。其他未进行详细介绍的模块同样具有重要作用，但在本文中未予以展开。这样做是为了聚焦于核心部分，突出系统的关键步骤和主要特点

运行和结果展示

UI参数配置界面如下：

在Herz-Jesus-P8数据集上运行程序结果如下：

数据集的一张图像



UI展示点云

如果启用pangolin可视界面，可以看到相机位姿和点云：

尽管由于时间原因，我们无法将结果与数据集的标准答案进行比对，但是该系统在三维重建方面取得了令人满意的成果。它成功生成了点云数据，这是表示三维物体表面的集合点。

点云是一种用于描述三维几何结构的数据表示形式，它由大量的点组成，每个点都有其在三维空间中的坐标信息。这些点可以捕捉到物体的形状、表面细节和空间位置等信息，从而实现对物体的可视化、分析和处理。

在该系统的结果中，我们可以观察到点云的形状和结构。通过可视化点云，我们可以更好地理解场景的几何特征和物体之间的空间关系。这对于许多应用领域都是非常有价值的，例如计算机辅助设计、虚拟现实、机器人导航和三维打印等。

尽管存在一些离群点，这并不意味着整个点云数据失去了价值。离群点可能是由于图像噪声、运动模糊或者遮挡等因素引起的。然而，这些离群点通常只占整个点云数据的一小部分，对于整体形状的表示并没有太大影响。

此外，该系统作为一个轻量级三维重建系统，具有一些优势。首先，它具有较低的计算资源需求，可以在较低配置的计算设备上运行，这使得它具备更广泛的应用潜力。其次，该系统的运行速度相对较快，可以在较短的时间内完成三维重建任务，提高了效率。

然而，由于我们无法将结果与数据集的标准答案进行比对，我们无法得知该系统的重建准确度和精度。与标准答案进行比对是评估三维重建系统性能的重要手段，它可以帮助我们了解系统的偏差和误差。因此，在进一步应用该系统的结果时，我们需要对其进行进一步的评估和验证。

总结而言，该轻量级三维重建系统在时间限制下取得了令人满意的成果，成功生成了点云数据。尽管存在一些离群点，但它们并不影响整体的点云表示。该系统的轻量级特性使其具备广泛的应用潜力，但在应用之前，我们需要进行更多的评估和验证工作，以确保结果的准确性和可靠性。

实验总结

作为一名学生，在大学的学习生涯中，我们小组选择了三维重建这项具有挑战性的课程设计任务。这门课程设计任务对于我们小组的专业发展和个人能力提升起到了极大的推动作用。在完成这个任务的过程中，我们经历了很多的挑战和成长，积累了宝贵的经验和知识。

首先，三维重建是一项具有挑战性的任务。它要求我们从二维图像或点云数据中还原出三维物体的形状和结构。这需要我们掌握多个领域的知识，包括计算机图形学、计算机视觉、数学和数据结构等。在开始任务之前，尽管在课程上听了老师的讲解，但是我们对这些领域的知识仍然了解有限(特别是在实际应用当中)，因此需要通过自学和查阅大量的资料来弥补不足。这个过程中，我们小组同学遇到了许多困难和障碍，但都坚持不懈地克服了它们。

其次，三维重建任务培养了解决问题的能力。在完成任务的过程中，需要运用所学的理论知识，结合实际情况，找到适合的方法和算法来解决问题。有时候，需要进行参数调整、数据预处理和模型优化等工作，以获得更好的重建效果。这锻炼了我们的分析和判断能力，培养了我们的问题解决思维。同时我们也学会了从多个角度思考问题，并寻找最佳解决方案。

最后，完成三维重建任务让我们对专业某些领域的发展和未来有了更清晰的认识。通过实践，可以深入的了解到三维重建技术的应用和潜力。也意识到三维重建在许多领域中都有广阔的应用前景，如虚拟现实、游戏开发、建筑设计等，我们感觉现在对这些领域的兴趣进一步加深。

总的来说，完成三维重建的课程设计任务是一次难忘的经历。在这个过程中，我们小组不仅学到了专业知识，还培养了解决问题的能力、团队合作能力。这门任务让我更加坚定了自己的学习目标。我们将把这次经历视为人生的财富，继续不断地努力学习和成长，为实现自己的梦想而努力奋斗！