Group Assignment: Incremental SFM

Zhou Ruoheng
G2303498K
zh0121ng@e.ntu.edu.sg

Zou Xinan
G2303947L
s230120@e.ntu.edu.sg

Li Jiali
G2406167G
JIALI004@e.ntu.edu.sg

Bao Wenrui
G2402610C
C240103@e.ntu.edu.sg

Hao Xiuran
G2406180B
xiuran001@e.ntu.edu.sg

Abstract

This paper describes our implementation of Incremental Structure-from-Motion (SfM) to reconstruct a 3D model of a campus landmark.

We began by calibrating the camera using a chessboard pattern to estimate intrinsic parameters and correct lens distortions. Preprocessing steps included undistorting images, cropping, resizing, and normalization to ensure consistency. For the initial reconstruction, we selected an image pair with significant viewpoint differences, extracted SIFT features, and performed feature matching using a FLANN-based matcher with a ratio test. We computed the essential matrix, recovered relative camera poses, and triangulated matched points to generate an initial sparse 3D point cloud. In the incremental phase, we added new images by selecting those with the highest overlap with the existing point cloud. We estimated their poses using the Perspective-n-Point (PnP) algorithm with RANSAC and triangulated new points to expand the model.

Our method demonstrated robustness and scalability across datasets in different sizes. We also discussed limitations of our approach and suggested future improvements in image organization, initial image pair selection, next view selection strategies, correspondence search methods, and dense reconstruction implementation.

1 Introduction

Structure-from-Motion (SFM) is a technique in computer vision that aims to infer the 3D geometric structure of a scene from a series of 2D images while simultaneously estimating the camera poses used to capture these images. In simple terms, SFM is a task of reconstructing 3D point clouds and camera trajectories from multiple images.

SFM has numerous practical applications. For instance, it can be used to digitally preserve historical sites. By taking a series of photos from different angles around a heritage site, SFM can reconstruct the 3D structure of the building, which can then be used for display, cultural preservation, or providing 3D data for further detailed modeling.

There are several frameworks and methodologies to solve SFM. Incremental Structure from Motion[8] (SFM) operates through sequential frame reconstruction, while global SFM[5] performs simultaneous optimization across all images. In contrast, optical flow[9] methods estimate motion and structure based on dense pixel-wise correspondence relationships.

In this project, we employ incremental SFM to reconstruct a 3D view of a landmark on campus. Incremental SFM builds the 3D reconstruction by starting with a small set of images and gradually adding new ones. It processes the image sequence in a step-by-step manner, dynamically expanding and optimizing the model from initial geometric relationships to a complete 3D structure.

This method begins by estimating the initial camera poses and sparse 3D point clouds using two or a few images with sufficient viewpoint differences. Subsequently, the method matches features from new images to the existing point cloud to estimate the camera poses for the new images. The observations from these new images are then used to triangulate additional 3D points, thereby extending the point cloud. In this project, a global Bundle Adjustment optimization is performed every time five new images are added to ensure the accuracy and consistency of the reconstructed 3D model.

This incremental expansion process is well-suited for handling the geometric relationships within an image sequence and can manage large datasets by completing the reconstruction in stages.

2 Technical Foundations

2.1 Four Coordinate Systems and their Transformations

2.1.1 World Coordinate System. [3]The world coordinate system is a special coordinate system that establishes a reference framework necessary for describing other coordinate systems. It allows the positions of other coordinate systems to be expressed in terms of the world coordinate system but cannot itself be described using a larger external coordinate system. From a non-technical perspective, the world coordinate system represents the largest coordinate system of interest, without necessarily encompassing the entire physical world. It is typically denoted as (X_w, Y_w, Z_w).

2.1.2 Camera Coordinate System. The camera coordinate system is defined with the geometric center of the camera lens (optical center) as the origin. This coordinate system follows the right-hand rule and is denoted as (X_cam, Y_cam, Z_cam). The camera optical axis serves as the Z-axis, the X-axis is horizontal, and the Y-axis is vertical. Figure 1 shows the Euclidean transformation between the world and camera coordinate frames.

2.1.3 Physical Image Coordinate System. The physical image coordinate system uses the center of the CCD image plane as the origin, with coordinates represented as (x, y). The unit of this coordinate system is typically millimeters. The origin corresponds to the intersection of the camera optical axis and the imaging plane, which is generally close to the geometric center of the image.

2.1.4 Pixel Image Coordinate System. The pixel image coordinate system is what is typically referred to when discussing an image.

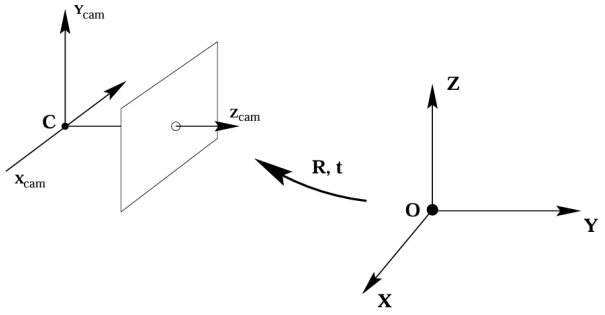


Figure 1: The Euclidean transformation between the world and camera coordinate frames

Its origin is located at the top-left corner of the image, and the unit is pixels, just like figure2.

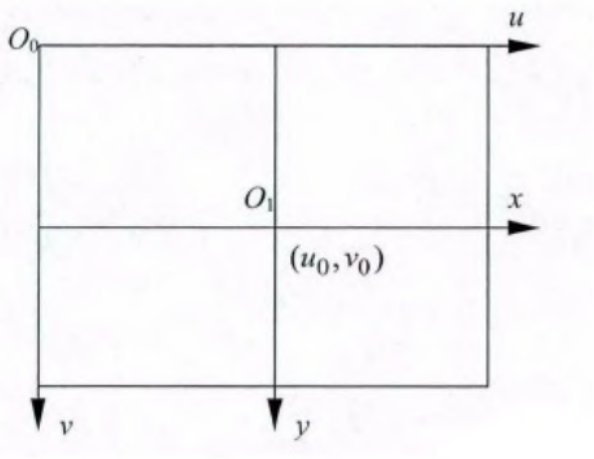


Figure 2: Image (x,y) and pixel image (u,v) coordinate systems

2.1.5 Coordinate Transformation. The relationships among the four coordinate systems can be expressed mathematically as follows:

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{M1: intrinsic parameter}} \underbrace{\begin{bmatrix} R_{3\times3} & T_{3\times1} \end{bmatrix}}_{\text{M2: extrinsic parameter}} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (1)$$

where $[u, v, 1]^\top$ is the pixel coordination, while the $[X_w, Y_w, Z_w, 1]^\top$ is the world coordination.

2.2 Camera Model

2.2.1 Pinhole Camera Model. The pinhole camera model is a fundamental camera model widely used in computer vision and image processing for geometric modeling. It assumes that light rays pass through a small hole, forming an inverted image on the camera's image plane. This model allows mapping 3D points in space to the 2D image plane.

Imaging Process. In the pinhole camera model, a 3D point in space $P = (X, Y, Z)$ passes through the camera's pinhole and projects onto the image plane, forming a 2D image point $p = (u, v)$. This projection process follows the principles of perspective projection.

Projection Relation. Through the intrinsic matrix K, a 3D point $P = (X, Y, Z)$ can be projected onto the image point $p = (u, v)$. The projection formula is[6]:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (2)$$

where s is a scaling factor that projects the 3D point onto 2D image coordinates.

The pinhole camera model provides the projection relationship from 3D points to the 2D image plane, enabling 3D structure recovery from multiple views in Incremental SfM.

2.2.2 Intrinsic and Extrinsic Parameters.

Intrinsic Parameters. The intrinsic parameters describe the optical and geometric characteristics of the camera, the imaging process can be represented by an intrinsic matrix that maps 3D points to the image coordinates.The intrinsic matrix K is a 3×3 matrix formed from these parameters, used to map 3D coordinates to the image plane:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where:

- f_x and f_y are the focal lengths in the x- and y-directions, usually calculated based on the camera's sensor and lens focal length, as well as pixel dimensions.
- c_x and c_y are the coordinates of the principal point, usually located at the center of the image.

Extrinsic Parameters. The extrinsic parameters define the position and orientation of the camera relative to the world coordinate system. The extrinsic parameters include a rotation matrix R and a translation vector t, which describe the relationship between the camera coordinate system and the world coordinate system.
If a 3D point in the world coordinate system is represented as $P_w = (X_w, Y_w, Z_w)$, it can be transformed to the camera coordinate system by:

$$P_c = R \cdot P_w + t \quad (4)$$

where P_c is the point coordinate in the camera coordinate system.

By combining the intrinsic and extrinsic parameters, we obtain the complete formula for projecting a 3D world point onto the 2D image plane:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot [R \mid t] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (5)$$

where $[R \mid t]$ is the 3×4 extrinsic matrix, which maps points from the world coordinate system to the camera coordinate system.

Intrinsic parameters define the camera's optical characteristics, ensuring consistency in feature point projections across different views, while extrinsic parameters describe each image's camera position and orientation, forming the foundation for building and expanding the 3D model.

2.3 Distortion Model and Correction Principles

The camera lens often introduces *distortion* during imaging, causing straight lines or points in the image to shift from their true projections. Distortion mainly includes **radial distortion** and **tangential distortion**, which can affect accurate geometric calculations. Therefore, in many computer vision tasks (e.g., 3D reconstruction or multi-view geometry), it is necessary to model and correct distortion[6].

2.3.1 *Types of Distortion.*

Radial Distortion. Radial distortion is caused by the curvature of the lens and is related to the distance of light rays from the optical axis. Radial distortion is typically represented by the following formulas:

$$x_{\text{distorted}} = x \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{6}$$

$$y_{\text{distorted}} = y \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{7}$$

where:

- (x, y): The normalized coordinates of an undistorted image point.
- $(x_{\text{distorted}}, y_{\text{distorted}})$: The normalized coordinates of the distorted image point.
- $r = \sqrt{x^2 + y^2}$: The radial distance from the image center.
- k_1, k_2, k_3: Radial distortion coefficients.

Radial distortion causes straight lines to appear curved. Radial distortion becomes larger the farther points are from the center of the image. Figure 3 is shown below in which two edges of a chess board are marked with red lines.

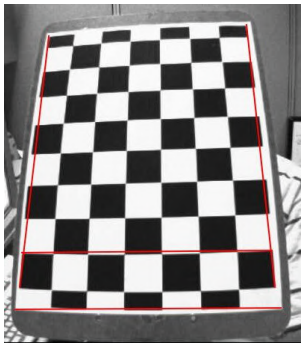


Figure 3: Chessboard with Detected Boundaries for Calibration

Tangential Distortion. Tangential distortion is caused by misalignment between the lens and the image sensor, resulting in points shifting in non-radial directions.

Tangential distortion is typically represented by the following formulas:

$$x_{\text{distorted}} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{8}$$

$$y_{\text{distorted}} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{9}$$

where:

- p_1, p_2: Tangential distortion coefficients.

In short, we need to find five parameters, known as distortion coefficients, given by:

$$\textit{Distortion coefficients} = (k_1,\ k_2,\ p_1,\ p_2,\ k_3) \tag{10}$$

Distortion Correction Model. The goal of distortion correction is to establish a distortion model that allows calculating the undistorted pixel coordinates.

2.3.2 *Correction Formula.*

The core of correction is to compute the undistorted point (x, y) from the distorted image point $(x_{\text{distorted}}, y_{\text{distorted}})$. Typically, the correction process is solved iteratively:

1. Compute an initial estimate (x_0, y_0) of the undistorted point.
2. Substitute into the distortion model to estimate its position in the distorted image.
3. Adjust (x_0, y_0) so that its projected point gradually approaches $(x_{\text{distorted}}, y_{\text{distorted}})$.

2.3.3 *Correction Process.*

The correction process typically includes the following steps:

1. **Camera Calibration**: Capture images of a calibration pattern, such as a checkerboard or circular pattern, to extract the relationship between image points and known 3D points.
2. **Solving Distortion Parameters**: Optimize the radial and tangential distortion coefficients by minimizing the reprojection error.
3. **Image Correction**: Use the distortion parameters to remap each pixel's coordinates in the image, generating the corrected image.

The Figure 4 is an example of feature point detection and connection, commonly used in feature point calibration during the camera calibration process. In this figure, each corner of the checkerboard is marked as a feature point (indicated by colored circles), and different colored lines connect these feature points.

During camera calibration, each corner point of the checkerboard is identified and located to calculate the camera's intrinsic parameters and distortion coefficients. The colored connecting lines visually illustrate the relative positions of the feature points, helping to observe if any distortions cause shifts. Ideally, these lines should be straight. However, due to tangential and radial distortions, the lines may appear curved or offset.

Figure 5 shows a distortion-corrected checkerboard pattern. The correction successfully removed radial and tangential distortions, restoring the edges and internal lines of the checkerboard to straight lines, with uniform square proportions and no geometric distortion. Such correction improves the accuracy of feature matching and pose estimation, providing a precise foundation for subsequent 3D reconstruction tasks.

Distortion correction removes lens-induced geometric distortions,

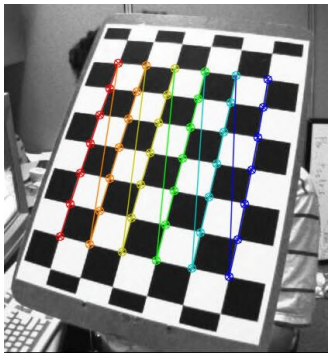


Figure 4: Feature Point Detection and Connection on Chessboard

Figure 5: Rectified Chessboard After Calibration

improving feature matching accuracy and ensuring precise geometry in each image, which enhances the accuracy of the 3D reconstruction process in SfM.

2.4 Feature Extraction and Matching

We need to find as many possible matches between two images to find the fundamental matrix. For this, we use SIFT descriptors with FLANN based matcher and ratio test.

(1) **SIFT**[4] extracts keypoints from images and computes high-dimensional descriptors.These descriptors encode local gradient information around each keypoint and are scale-invariant and rotation-invariant, making them robust for matching across images with different viewpoints or lighting.
(2) **FLANN**[1] is an optimized algorithm for finding the nearest neighbors in high-dimensional spaces. It uses efficient data structures to speed up the matching process.
(3) **Ratio test** filters out unreliable matches. For each keypoint, it evaluates the ratio of the distance between the nearest neighbor and the second-nearest neighbor. A match is considered reliable if the ratio is below a predefined threshold (commonly 0.75).

2.5 Epipolar Constraint

Epipolar geometry[7] describes the geometric relationship between two camera views of the same 3D scene. It defines constraints, known as epipolar constraints, that limit the possible locations of corresponding points in each view, making feature matching more efficient. This is foundational in 3D reconstruction tasks, as it enables more accurate and computationally efficient identification of matching points across multiple images.

Epipoles. The epipole is the projection of one camera's optical center onto the other camera's image plane.

Epipolar Line Constraint. Given a point in the first image, its corresponding point in the second image must lie on the epipolar line. This constraint reduces the feature matching search from a 2D area to a 1D line, improving efficiency and robustness.

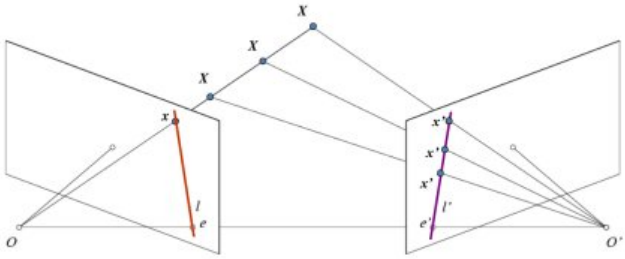


Figure 6: Epipolar Geometry and Epipolar Lines

- **Epipoles**: Points O and O' are the optical centers of two cameras. The projection of each camera center onto the other camera's image plane is called the epipole, labeled e and e'.
- **Epipolar Lines**: The 3D point X projects to points x and x' in the two images. Points x and x' lie on the epipolar lines l and l', respectively.
- **Epipolar Constraint**: Given a point x in the first image, its corresponding point x' in the second image must lie on the corresponding epipolar line l'. This constraint reduces the search area for feature matching from a 2D region to a 1D line, enhancing matching efficiency.

The epipolar geometry constraint restricts the search space for feature matching in multi-view geometry. In SfM, especially incremental SfM, each new viewpoint introduces numerous feature points that need to be matched across images. The epipolar constraint uses epipolar lines to reduce the feature matching problem from a 2D area to a 1D line search, significantly improving matching efficiency and reducing mismatches. This is essential for accurate camera pose estimation and 3D point reconstruction.

2.5.1 Fundamental Matrix and Essential Matrix. In Figure 6, we can see a mapping from a point in one image to the corresponding epipolar line in the other image, represented as $x \rightarrow l'$. The Fundamental Matrix represents this projective mapping relationship from a point to a line.

Fundamental Matrix.
Definition. For any pair of corresponding points $\mathbf{x}$ in the plane to the left of Figure 4 and $\mathbf{x}'$ in the right, the following equation holds:

$$\mathbf{x}'^{\top} F\mathbf{x} = 0 \tag{11}$$

where:

- $\mathbf{x}$ and $\mathbf{x}'$ are points in homogeneous coordinates (3×1 vectors).
- F is the fundamental matrix with 7 degrees of freedom.

Properties. The fundamental matrix is determined by the intrinsic and extrinsic geometry between two images.

Essential Matrix. The **Essential Matrix** E is a special case of the fundamental matrix when the intrinsic parameters of the cameras are known.
Definition. The essential matrix satisfies the same relationship as the fundamental matrix:

$$\mathbf{x}'^{\top} E \mathbf{x} = 0 \tag{12}$$

However, it is related to the fundamental matrix through the camera intrinsic matrices:

$$E = K'^{\top} F K \tag{13}$$

where:

- K and K' are the intrinsic matrices of the two cameras.
- F is the fundamental matrix.

Properties. The essential matrix is determined only by the relative rotation and translation (extrinsic parameters) of the cameras.
Figure 7 illustrates the geometric relationships between two camera views in a stereo vision system.

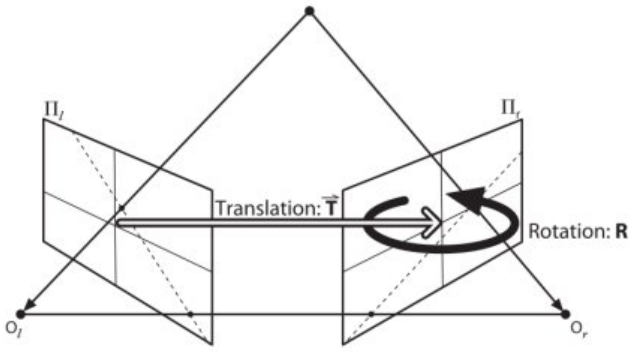


Figure 7: Camera Transformation with Rotation and Translation

- **Camera Centers (O_l and O_r)**: The optical centers of the left and right cameras are labeled O_l and O_r, respectively. The two cameras have a certain distance in 3D space, forming a baseline.
- **Rotation (R)**: The right camera is rotated relative to the left camera. The rotation matrix R represents the change in direction from the left camera's coordinate system to the right camera's coordinate system, indicated by a circular arrow in the figure.
- **Translation (T)**: The translation vector T between the two camera centers describes the displacement from the left camera to the right camera, representing the spatial distance between them.
- **Image Planes (Π_l and Π_r)**: The rectangles in the figure represent the image planes of the left and right cameras, Π_l and Π_r. Points on each image plane can correspond to points on the other plane through rotation and translation.

Mathematical Relationship. The **Essential Matrix** E is an extension of the **Fundamental Matrix** F when the intrinsic parameters of the cameras are known. Their relationship is expressed as:

$$E = K'^{\top} F K \tag{14}$$

where:

- K and K' are the intrinsic matrices of the two cameras.

Conversely, if the essential matrix E and the intrinsic parameters are known, the fundamental matrix can be computed as:

$$F = K'^{-\top} E K^{-1} \tag{15}$$

The essential matrix depends on the extrinsic parameters of the cameras (rotation and translation) and their intrinsic parameters, while the fundamental matrix is entirely determined by the relationships between points in the images, independent of the intrinsic parameters.

Geometric Meaning and Degrees of Freedom. The **Fundamental Matrix** F is defined in the pixel coordinate system and describes the geometric relationship between corresponding points in two images. It is applicable when the intrinsic parameters of the cameras are unknown. The fundamental matrix does not rely on the intrinsic parameters and has **7 degrees of freedom** (as a 3×3 matrix with rank 2, constrained by normalization).
The **Essential Matrix** E, on the other hand, is defined in the normalized image coordinate system, which removes the influence of the camera's intrinsic parameters. It directly reflects the relative rotation and translation between the cameras. The essential matrix has **5 degrees of freedom**, derived from the 3 degrees of freedom of rotation and 2 degrees of freedom of translation (since scale is not recoverable).
Process

(1) Extract SIFT descriptors for both images.
(2) Use FLANN to find the two closest matches in descriptor space for each keypoint in the first image.
(3) Apply the ratio test to the nearest and second-nearest matches:

$$\text{ratio} = \frac{\text{distance to nearest neighbor}}{\text{distance to second-nearest neighbor}} \tag{16}$$

Retain matches where the ratio is below the threshold, ensuring only reliable matches are kept.

Figure 8 illustrates the results of feature matching and epipolar geometry visualization between two images.
The colored dots in both images represent the detected feature points. The colorful lines in the image are epipolar lines, which were computed using the fundamental matrix. The lines connecting the two images represent the matches between feature points in the two views.

2.6 Triangulation Principle

The triangulation principle[2] plays a core role in this assignment by determining the 3D positions of points in the scene from their projections in multiple views.

2.6.1 *Key Steps of Triangulation.*

(1) **Extracting Key Points from Multi-view Images:** Use computer vision techniques (e.g., feature detection and matching) to identify corresponding key points across 2D

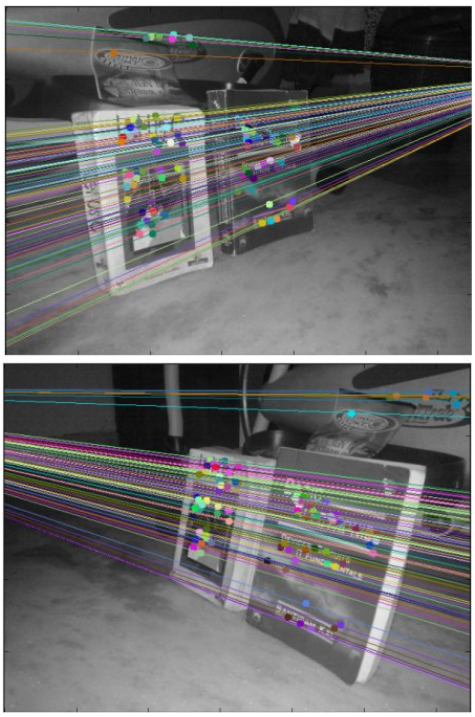


Figure 8: Feature Matching and Epipolar Geometry Visualization

images from different viewpoints. These key points represent projections of the same physical point in 3D space.

(2) **Estimating Camera Poses:** Leverage the geometric relationships between images and use camera intrinsics (e.g., focal length, principal point) to estimate the position and orientation (pose) of each image.

(3) **Reconstructing 3D Points via Triangulation:** For each pair of matched key points, use their 2D pixel coordinates as input. Apply the triangulation principle, combining the pixel coordinates and camera poses to compute the 3D coordinates of corresponding points in the scene.

2.6.2 Steps for Implementation.

(1) **Feature Matching:** Extract key points using algorithms like SIFT or ORB.
Match the key points across different images.

(2) **Essential/Fundamental Matrix Calculation:** Compute the essential or fundamental matrix to describe the geometric relationships between images.

(3) **Camera Projection Matrix Calculation:** Solve for the camera projection matrix to determine the pose of each camera view.

(4) **3D Point Reconstruction:** Use the projection matrices to back-project the 2D points into 3D space. Solve the triangulation equations to compute the 3D point coordinates.

(5) **Point Cloud Generation:** The reconstructed 3D points form a point cloud, representing the 3D structure of the scene.

2.6.3 Mathematical formulas of Triangulation. Assume two images I_1 and I_2 with respective projection matrices P_1 and P_2. If the 2D pixel coordinates are x_1 and x_2, the goal is to solve for the 3D point X. The relationship is given by:

$$x_1 = P_1 X, \quad x_2 = P_2 X \tag{17}$$

By minimizing the reprojection error (e.g., using linear algebra techniques), the 3D point X can be computed accurately.

Using triangulation and Structure-from-Motion (SfM) techniques, the task aims to reconstruct the 3D structure of a scene, typically represented as a point cloud, while also evaluating the generalization capability of the algorithm to ensure its applicability across different scenes.

2.7 3D-2D: PnP

Perspective-n-Point(PnP)[2] is for the problem of determining the pose of a calibrated camera from n correspondences between 3D reference points and their 2D projections. The PnP problem has many solution methods, such as P3P, which estimates the pose using three-point correspondences, and Direct Linear Transformation (DLT).

- P3P (Perspective-Three-Point): P3P requires the use of the geometric relationships of the given three points. Its input data consists of three pairs of 3D-2D matching points. Let the 3D points be A, B, and C, and the 2D points be a, b, and c, where the lowercase letters represent the projections of the uppercase points onto the camera's image plane, as shown in Figure9 . In addition, P3P also requires the use of a verification point pair to select the correct solution from the possible ones (similar to the case in epipolar geometry). Let the verification point pair be D - d, with the camera's optical center denoted as O. It is important to note that what we know are the coordinates of A, B, and C in the world coordinate system, not in the camera coordinate system.

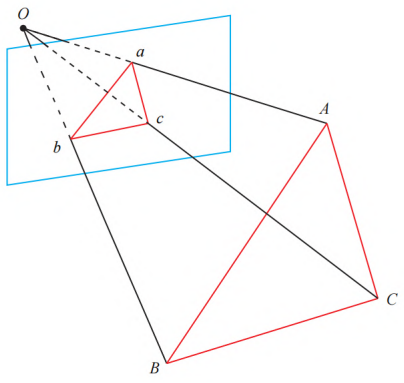


Figure 9: Illustration for P3P

- DLT (Direct Linear Transformation) is a linear algebra-based method used for solving the PnP problem, where 3D-2D correspondences are used to compute the camera's intrinsic and extrinsic parameters. DLT works by setting

up a system of linear equations that relate the 3D world coordinates and their 2D projections in the image.

Subsequently, many other methods have been proposed, such as EPnP and UPnP. These methods leverage additional information and use iterative approaches to optimize camera poses, aiming to minimize the impact of noise as much as possible. However, compared to P3P, their principles are more complex.

RANSAC (Random Sample Consensus) can help solve the PnP problem by providing a robust method for estimating the camera pose, especially in the presence of outliers in the data. In the PnP problem, the goal is to estimate the camera's position and orientation based on a set of 3D world points and their corresponding 2D projections in an image. However, due to noise or incorrect feature matches, some correspondences may be outliers, which can distort the pose estimation.

RANSAC addresses this by iteratively selecting a random subset of point correspondences (typically three pairs for P3P) to estimate the camera pose. The algorithm then checks how many of the remaining correspondences fit the estimated model, identifying inliers that are consistent with the estimated pose. This process is repeated multiple times, and the solution that results in the highest number of inliers is chosen as the final estimate for the camera pose.

In this way, RANSAC improves the PnP solution by filtering out erroneous correspondences and ensuring that the camera pose is determined using only the most reliable points, leading to a more accurate and robust result.

2.8 Bundle Adjustment

Bundle Adjustment can be divided in to Global Bundle Adjustment (GBA) and Local Bundle Adjustment (LBA). GBA performs a global optimization of all camera parameters and 3D point positions in the entire system. It considers the observation relationships across all image viewpoints to minimize the overall reprojection error. In contrast, LBA only optimizes the current frame and its neighboring keyframes (typically within a sliding window), focusing on local map accuracy to reduce real-time computational errors. In this task, the GBA method is adopted.

GBA begins with an initial estimate of the 3D structure of the scene, as well as the camera poses, which include their positions and orientations. Once the initial setup is in place, GBA formulates the problem as a non-linear least squares optimization. The goal is to minimize the reprojection error, which measures the difference between the observed 2D image points and the 2D projections of the estimated 3D points onto the image planes. And this error item is written as formula 18

$$\xi^* = \arg\min_{\xi} \frac{1}{2} \sum_{i=1}^{n} \left\| \mathbf{u}_i - \frac{1}{s_i} \mathbf{K}[R \mid K]\mathbf{P}_i \right\|_2^2. \tag{18}$$

where the computed camera pose is represented by R, t, and its Lie algebra representation is ξ, the coordinates of a certain spatial point are P_i, and its projected pixel coordinates are u_i. The optimization involves iteratively adjusting the parameters to reduce the reprojection error. During each iteration, the algorithm computes how changes in the parameters affect the reprojection error and updates them accordingly. Throughout this process, GBA simultaneously

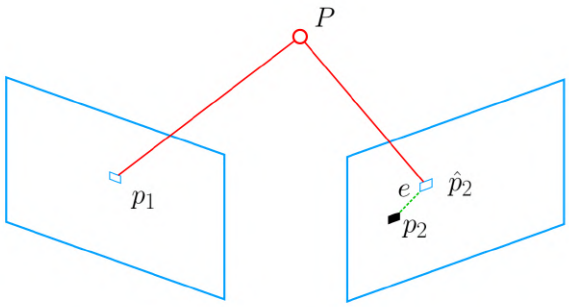


Figure 10: Illustration for Reprojection Error

considers all views and 3D points, ensuring that the adjustments are consistent globally. By the end of the process, GBA outputs the refined camera poses, 3D point positions, and optionally the camera intrinsics, resulting in a globally optimized reconstruction with minimal reprojection error.

3 Pipeline and Experiments

Figure 11 shows the whole pipeline of our designed reconstruction procedure.

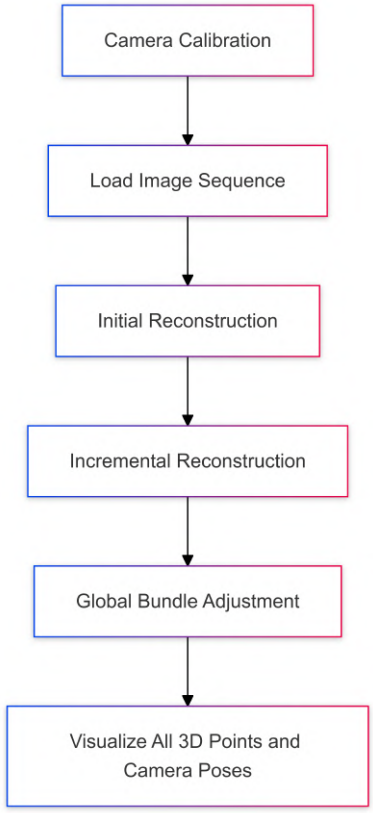


Figure 11: Our Workflow

3.1 Camera Calibration

The camera calibration process ensures accurate estimation of intrinsic and extrinsic parameters, enabling precise projection of 3D points into 2D image space[9].

3.1.1 Initialization. The calibration begins with defining the setup and parameters for processing:

- **Chessboard Pattern Setup**: The calibration uses a 9×6 chessboard, where the internal corners form a structured grid. The 3D world coordinates of these corners are defined on a flat plane ($Z = 0$).
- **Image Preparation**: A set of images capturing the chessboard from different angles and positions is collected. These images ensure a robust dataset for accurate calibration.
- **Optimization Criteria**: Iterative optimization criteria are established. These include a maximum iteration limit and a convergence threshold for refining corner positions.

3.1.2 Corner Detection. Accurate detection of chessboard corners is essential for effective calibration:

- **Preprocessing**: Calibration images are converted to grayscale to simplify processing and improve corner detection accuracy.
- **Feature Detection**:
  - The 9×6 chessboard corners are located in each grayscale image.
  - Predefined 3D world coordinates of the chessboard corners are appended to a list for calibration.
  - Detected 2D image points are refined to sub-pixel accuracy for enhanced precision.
- **Visualization**:Annotated images showing the detected corners are saved for quality control and validation.

3.1.3 Parameter Estimation. The calibration process estimates intrinsic and extrinsic parameters:

- **Intrinsic Parameters**: The intrinsic matrix, including focal lengths (f_x, f_y) and principal point coordinates (c_x, c_y), is computed from the 3D-2D point correspondences.
- **Extrinsic Parameters**:The camera's rotation matrix (R) and translation vector (t) are calculated. These describe the position and orientation of the camera relative to the world coordinate system.
- **Distortion Coefficients**: Radial and tangential distortion coefficients are estimated.These correct optical aberrations and ensure accurate mapping between 3D world points and the image plane.

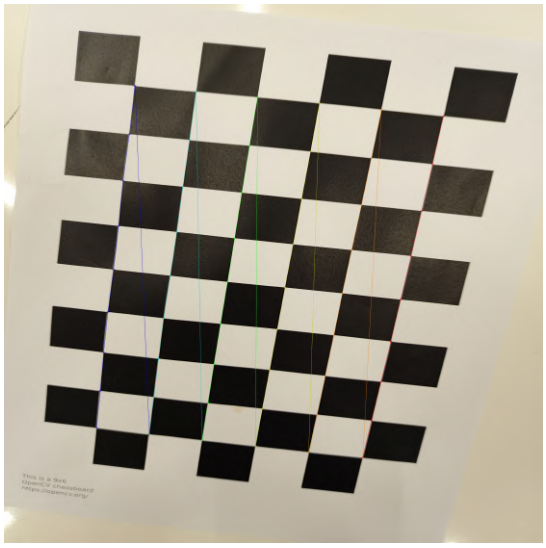


Figure 12: Chessboard after camera calibration

3.1.4 Validation. Validation ensures the accuracy of the results:

- **Reprojection Error Analysis**: The reprojection error is computed by projecting the 3D chessboard corners back into the 2D image plane. The error is compared with the detected 2D points. Lower errors indicate better accuracy.
- **Visual Inspection**: Annotated images are reviewed to confirm the alignment of detected corners with the actual chessboard pattern.

```
Camera Intrinsic Matrix:
 [[2.88343889e+03 0.00000000e+00 1.50153423e+03]
 [0.00000000e+00 2.91468589e+03 1.99828861e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
Camera Distortion Coefficients:
 [[ 5.90267031e-02 -4.64560315e-01  1.69330902e-03  1.61081073e-04
   7.08018543e-01]]
```

Figure 13: Result of camera calibration

3.1.5 Saving and Reusing Calibration Results. Efficient storage and reuse of results ensure a streamlined process for future applications:

- **Saving Parameters**: The intrinsic matrix, distortion coefficients, and extrinsic parameters are stored in a structured file format that ensures easy access and reusability.
- **Result Validation**: The saved parameters are checked to confirm successful storage.
- **Loading Existing Parameters**: Before recalibrating, the workflow includes a check for previously saved calibration results. If available, these parameters are directly loaded to avoid redundant computation.

3.2 Image Preprocessing

Image preprocessing ensures consistency and quality in input images, preparing them for subsequent tasks by addressing issues such as distortion, cropping, and feature quality.

3.2.1 Distortion Correction. Removing lens distortion is the first step in preprocessing to ensure geometric accuracy in the images:

- **Undistortion Process**: Lens distortion, including radial and tangential aberrations, is corrected using the intrinsic matrix K and distortion coefficients obtained during camera calibration. This removes the "barrel" or "pincushion" effects, ensuring straight lines in the real world are also straight in the image.
- **Edge Cropping**: Distortion correction may cause irregular edges in the image.And cropping is applied to remove distorted edges, retaining only the valid region of the image. This improves the consistency of the dataset and ensures that only useful pixels are preserved for further processing.

Figure 14: Left: Original. Right: After distortion correction

3.2.2 Resizing and Normalization. Standardizing image size and intensity:

- **Image Resizing**: All images are resized to a consistent resolution, such as 640×480 pixels, ensuring uniform scale and aspect ratio. This standardization improves computational efficiency in subsequent processing steps.
- **Pixel Normalization**: Pixel intensity values are normalized to a fixed range (e.g., 0-1) or standardized with zero mean and unit variance. Normalization reduces sensitivity to lighting variations and improves the reliability of feature extraction.

3.2.3 Feature Enhancement. Enhancing the quality of image features is critical for robust keypoint detection and matching:

- **Noise Reduction**: Noise is removed using filters such as Gaussian blur or bilateral filtering, which preserve edges while reducing random variations. This enhances the clarity of features and reduces false detections during keypoint extraction.
- **Contrast Adjustment**: Methods like histogram equalization are applied to improve image contrast to enhance contrast highlights features such as edges and corners, improving the visibility of keypoints.

3.2.4 Validation. To ensure the quality of preprocessed images before they are used in furthur tasks:

- **Visual Inspection**: Processed images are reviewed to confirm that distortions are removed, and the images are properly cropped and normalized. Consistency in resolution and intensity across all images is checked to ensure uniformity in the dataset.
- **Reprojection Consistency**: Keypoints detected in the preprocessed images are validated by projecting them back into the camera's view. The alignment of the projected points with the real-world features is checked to verify accuracy.

3.3 Initial Reconstruction

The initial reconstruction phase generates a sparse 3D point cloud from a carefully selected image pair, forming the basis for further 3D scene reconstruction.

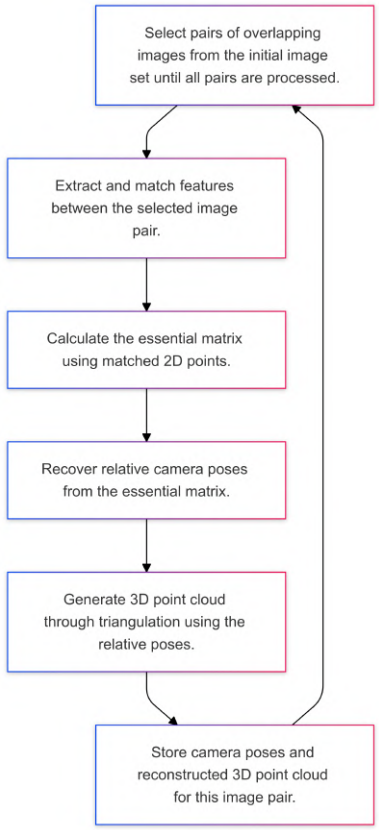


Figure 15: Initial Reconstruction Workflow

3.3.1 Initial Image Pair Selection.

- Select the image pair with the highest number of feature matches. Ensure sufficient overlap between the two images to provide stable reconstruction.
- Further improvements, as detailed in the subsequent improvement subsection, have been implemented.

3.3.2 Essential Matrix Estimation. Compute the essential matrix E using feature matches and intrinsic parameters to encode the geometric relationship between the two cameras:

Figure 16: Feature Extraction (Red dot: Keypoints)

Figure 17: Matched feature points between two images

- **Calculation of the Essential Matrix**: Use matched feature points to compute the essential matrix. Incorporate the camera's intrinsic parameters during the computation.
- **Robust Estimation**: Apply RANSAC to eliminate outliers and ensure accurate estimation of E. Validate the essential matrix against geometric constraints to confirm correctness.

3.3.3 Relative Pose Recovery. Relative camera poses are recovered and recursively propagated to establish a consistent world coordinate system:

- **Camera 0 as World Coordinate System**:
  - Define Camera 0 as the reference frame with:

$$\mathbf{R}_0 = \mathbf{I}, \quad \mathbf{t}_0 = \mathbf{0}$$

    Use Camera 0's coordinate system as the global reference for all subsequent calculations.
- **Relative Pose of Camera 1**:
  - Recover the relative rotation $\mathbf{R}_{1\to0}$ and translation $\mathbf{t}_{1\to0}$ from E.
  - Transform Camera 1 into the world coordinate system:

$$\mathbf{R}_1 = \mathbf{R}_{1\to0}, \quad \mathbf{t}_1 = \mathbf{t}_{1\to0}$$

- **Recursive Pose Recovery for Additional Cameras**:
  - For Camera $i+1$, compute its pose relative to Camera i:

$$\mathbf{R}_{i+1} = \mathbf{R}_{i+1\to i} \cdot \mathbf{R}_i$$

$$\mathbf{t}_{i+1} = \mathbf{R}_{i+1\to i} \cdot \mathbf{t}_i + \mathbf{t}_{i+1\to i}$$

  - Repeat this process for all additional cameras to maintain a consistent global coordinate system.

3.3.4 Triangulation of 3D Points. Matched feature points are triangulated to generate a sparse 3D point cloud representing the scene:

- **3D Point Reconstruction**: Use the relative camera poses and corresponding projection matrices to triangulate 3D points. Solve for 3D coordinates of each matched feature pair using linear least-squares methods.
- **Sparse Point Cloud Generation**: Aggregate the triangulated 3D points to form the initial sparse point cloud. This ensure the point cloud covers key features of the scene for further reconstruction.

3.3.5 Validation. The sparse 3D point cloud is validated through visualization and reprojection error analysis to ensure reconstruction accuracy:

- **Point Cloud Visualization**: Visualize the sparse 3D point cloud to inspect coverage and consistency. Identify and correct any visible anomalies in the reconstruction.
- **Reprojection Error Analysis**: Compare reprojected 2D points with original to calculate the reprojection error. Lower errors indicate higher reconstruction accuracy.

```
@ camera pose:  [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]] [[0.]
 [0.]
 [0.]]
1 camera pose:  [[ 9.99997561e-01  7.39027180e-04  2.08132243e-03]
 [-7.37531513e-04  9.99999469e-01 -7.19289644e-04]
 [-2.08185290e-03  7.17752849e-04  9.99997575e-01]] [[ 0.32985137]
 [ 0.45759601]
 [-0.82571422]]
After initial reconstruction, number of 3D points:  2015
After initial reconstruction, number of camera poses:  2
```

Figure 18: Result of Initial Reconstruction

3.4 Incremental Reconstruction

Incremental Reconstruction aims to extend the initial 3D model by incorporating additional images one by one. This approach allows for building a comprehensive and accurate 3D representation of the scene by sequentially adding new information.

3.4.1 ***Candidate Image Selection.*** Selecting the next best image to incorporate is critical for a stable and accurate reconstruction.

- **Overlap Evaluation**: For each image not yet included in the reconstruction, calculate the number of shared features with the existing 3D points.
- **Selection Criteria**: Choose the image with the highest number of correspondences to the existing model. This ensures sufficient overlap, which is essential for accurate pose estimation.
- **Quality Considerations**: Take into account image quality factors such as focus, exposure, and motion blur to prioritize images that will contribute positively to the reconstruction.

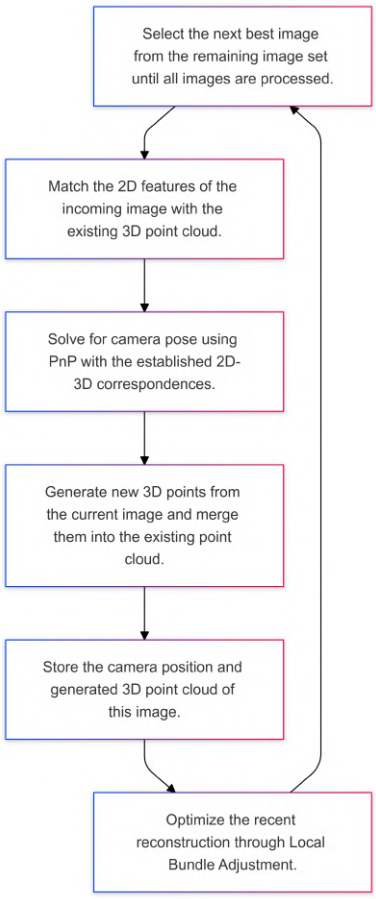


Figure 19: Incremental Reconstruction Workflow

3.4.2 *Feature Extraction and Match*.

- Extract distinctive features from images using algorithms like SIFT.
- Adopt FLANN as mentioned above to Match features between image pairs using descriptors, ensuring robustness to changes in scale, rotation, and illumination.
- Apply ratio test (comparing distances of nearest and second-nearest neighbors) with threshold 0.75 to filter out ambiguous matches.
- Use techniques such as RANSAC to eliminate outliers and improve matching accuracy.

3.4.3 *Correspondent 3D Points Selection*.

- For a selected image, use feature matching to find corresponding points in the previous image.
- Identify 2D matching points that have corresponding 3D points by taking the intersection of feature matches with existing 3D point projections.
- Recursively search the current 3D point cloud to find and verify the specific 3D points corresponding to the matched 2D points.

The specific Implementations of above steps are abstracted by following Algorithms 1, 2, 3 and 4.

The *unique_points3D_dict* implementation utilizes a nested dictionary structure, where the outer dictionary employs image indices as keys, and the inner dictionary maps 2D point coordinates to their corresponding 3D points. This structure represents the unique 3D points generated from each respective image.

In addition, The *good_matches_dict* is structured as a nested dictionary, where the outer dictionary keys represent the mapping from image index i to image index j, while the inner dictionary establishes correspondences between 2D points in image i to their matching 2D points in image j.

Algorithm 1 RecursiveIndexing(pt, idx, target_idx)

if $idx = target\_idx$ **then**
  return pt
else
  $good\_matches \leftarrow good\_matches\_dict[idx]$
  if $pt \in good\_matches$ **then**
    $indexed\_pt \leftarrow good\_matches[pt]$
    $next\_idx \leftarrow idx - 1$
    return RecursiveIndexing($indexed\_pt, next\_idx, target\_idx$)
  else
    return null
  end if
end if

Algorithm 2 IndexStartPointOnTargetIdx (src_pt, src_idx, target_idx)

return RecursiveIndexing($src\_pt, src\_idx, target\_idx$)

Algorithm 3 MatchOnTargetIdx(target_idx, src_pts, matched_mask, incremental_mask, total_points3D)

$unique\_point3D\_dict \leftarrow unique\_points3D\_dict[target\_idx]$
for $i = 0$ to $|src\_pts| - 1$ **do**
  if not $matched\_mask[i]$ **then**
    $indexed\_pt \leftarrow$ IndexStartPointOnTargetIdx ($src\_pts[i], src\_idx, target\_idx$)
    if $indexed\_pt \neq$ null AND $indexed\_ptunique\_point3D\_dict$ **then**
      $matched\_mask[i] \leftarrow$ true
      $incremental\_mask[i] \leftarrow$ false
      $total\_points3D[i] \leftarrow unique\_point3D\_dict[indexed\_pt]$
    end if
  end if
end for

Algorithm 4 Match3DBy2D(unique_points3D_dict, good_matches_dict, src_idx, src_pts)

// Initialize arrays
matched_mask ← *zeros*(|*src_pts*|)
incremental_mask ← *ones*(|*src_pts*|)
total_points3D ← *zeros*(|*src_pts*| × 3)
cur_idx ← *src_idx*
end_idx ← 1
while *cur_idx* ≥ *end_idx* **do**
  MatchOnTargetIdx
    (*cur_idx*, *src_pts*, *matched_mask*, *incremental_mask*, *total_points3D*)
  cur_idx ← *cur_idx* − 1
end while
return *total_points3D*, *matched_mask*, *incremental_mask*

3.4.4 ***Camera Pose Estimation using PnP.*** Once a candidate image is selected, its position and orientation relative to the existing model must be determined.

- **Feature Matching**: Detect keypoints in the new image using methods like SIFT or ORB. Match these keypoints to the existing 3D points based on descriptor similarity.
- **PnP Algorithm**: Use the Perspective-n-Point (PnP) algorithm to estimate the camera pose. Employ RANSAC to handle outliers and improve robustness.
- **Implementation**:
  - Use `cv2.solvePnPRansac()` from OpenCV:

```
retval, rvec, tvec, inliers = cv2.
    solvePnPRansac(
    objectPoints,  # 3D points in the
        world coordinate system
    imagePoints,   # 2D points in the
        image plane
    cameraMatrix,  # Camera intrinsic
        parameters
    distCoeffs     # Distortion
        coefficients
)
```

  - `rvec` and `tvec` represent the rotation and translation vectors, respectively.
- **Pose Refinement**: Apply optimization algorithms like Levenberg-Marquardt to refine the estimated pose by minimizing the reprojection error.

3.4.5 ***New 3D Points Generation.*** To enrich the 3D model, new points are triangulated using the newly added image.

- **Identify Unmatched Keypoints**: Find keypoints in the new image that have not been matched to existing 3D points.
- **Find Correspondences**: Match these keypoints with keypoints in other images that have overlapping views but have not yet contributed to 3D points.
- **Triangulation**:
  - Use multiple views to triangulate the 3D positions of new points.
  - Implement using `cv2.triangulatePoints()`:

```
points4D = cv2.triangulatePoints(
    projMat1, projMat2,   # Projection
        matrices of the cameras
    points1, points2      #
        Corresponding 2D points in each
        image
)
```

  - Convert homogeneous coordinates to 3D coordinates.
- **Update the Model**: Add the new 3D points to the existing point cloud.

3.4.6 ***Result of Update 3D points cloud.*** In our implementation, for a selected image, we first find the corresponding exsiting 3D points. Then we find the 3D points we need to add. Lastly, we find the unique incremental 3D points that we need to add exactly. We show our results in red bounding boxes.

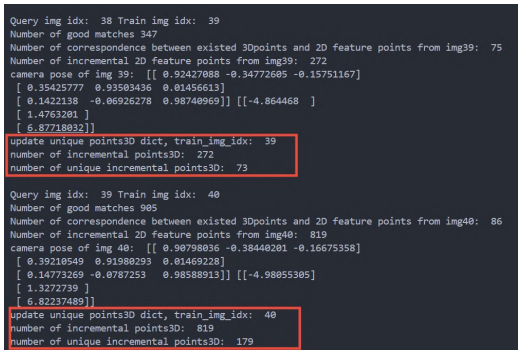


Figure 20: Our Incremental 3D points

3.4.7 ***Local Bundle Adjustment.*** To refine the accuracy of the added points and camera poses, a local optimization is performed.

- **Define the Local Subset**: Include the newly added camera and its neighboring cameras, as well as the 3D points observed by these cameras.
- **Optimization Goal**: Minimize the reprojection error for the local subset.
- **Optimization Technique**:
  - Use optimization libraries such as Ceres Solver or g2o.
  - Adjust camera parameters (intrinsic if necessary, and extrinsic) and 3D point positions.
  - Employ Levenberg-Marquardt algorithm for robust convergence.
- **Iterative Process**: Repeat the incremental addition of images followed by local bundle adjustment until all images are processed.

3.5 Global Bundle Adjustment

After all images have been incorporated, a global optimization ensures that the overall reconstruction is consistent and accurate.

3.5.1 ***Minimizing Reprojection Error.*** The global bundle adjustment considers all cameras and 3D points simultaneously.

- **Objective Function**:

$$\min_{\{\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j\}} \sum_{i=1}^{N} \sum_{j=1}^{M} \left\| \mathbf{u}_{ij} - \pi(\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j) \right\|^2$$

  where:
  - $\mathbf{u}_{ij}$ is the observed 2D point in image i corresponding to 3D point j.
  - $\pi(\cdot)$ is the projection function from 3D to 2D.
  - $\mathbf{R}_i, \mathbf{t}_i$ are the rotation and translation of camera i.
  - $\mathbf{X}_j$ is the position of 3D point j.
- **Inclusion of Intrinsics**:
  - If camera intrinsics vary or need refinement, include them in the optimization variables.

3.5.2 *Refining Camera Parameters and 3D Structure*.

- **Optimization Algorithms**:
  - Use sparse bundle adjustment techniques to handle the large number of variables efficiently.
  - Utilize the sparsity of the Jacobian matrix due to the fact that each 3D point is observed in a subset of images.
- **Outlier Rejection**:
  - Implement robust loss functions (e.g., Huber loss) to reduce the influence of outliers.
- **Convergence Criteria**:
  - Set thresholds for parameter changes and error reduction to determine when to stop the optimization.
- **Result**:
  - Obtain refined estimates of all camera parameters and 3D point positions, leading to a consistent and accurate model.

3.6 Visualization

Visualization is crucial for evaluating the quality of the reconstruction and for presenting the results.

3.6.1 *3D Point Cloud Visualization*.

- **Visualization Tools**:
  - Use libraries such as Mayavi or matplotlib for rendering.
- **Rendering the Point Cloud**:
  - Display the 3D points in a coordinate system.
  - Apply coloring based on intensity, depth, or other attributes if available.
- **Interactivity**:
  - Allow rotation, zooming, and panning to inspect different parts of the model.

3.6.2 *Camera Trajectory Visualization*.

- **Visualizing Camera Positions**:
  - Represent each camera as a frustum or a simple symbol (e.g., pyramid) indicating position and orientation.
- **Trajectory Lines**:
  - Connect camera centers in the order of image capture to show the path.
- **Combined View**:
  - Overlay the camera trajectory on the 3D point cloud to assess coverage and movement.

3.6.3 *Error Analysis*.

- **Reprojection Error Visualization**:
  - Plot the reprojection error for 2D-3D correspondences to identify problematic areas.
- **Density Evaluation**:
  - Assess the distribution and density of the 3D points to detect holes or sparse regions.
- **Quality Metrics**:
  - Compute and visualize metrics such as mean error, standard deviation, and error histograms.

4 Result Analysis

4.1 Basic Results

We demonstrate the effectiveness of our method across different scales of input data using three distinct test cases.

4.1.1 ***Rock Formation Reconstruction***. Using a small dataset of 21 images, we successfully reconstructed detailed rock formations, , as shown in Figure 21. Despite the limited number of input images, our method captured fine surface textures and structural details, demonstrating robust performance with minimal data.

Figure 21: Rock formation reconstruction from 21 images.

4.1.2 ***Sculpture Reconstruction***. The medium-scale test involved reconstructing a Sculpture using 65 images, as shown in Figure 22. This case showcases our method's capability to handle increased complexity while maintaining accuracy in capturing both large-scale features and subtle artistic details.

Figure 22: Sculpture formation reconstruction from 65 images.

4.1.3 ***Monument Reconstruction.*** Our largest test case involved reconstructing a monument using 143 images, as shown in Figure 23. This comprehensive dataset allowed for highly detailed reconstruction, preserving intricate inscriptions and surface characteristics. The results demonstrate our method's scalability to larger datasets without compromising quality.

Figure 23: Monument formation reconstruction from 143 images.

These three cases validate our method's consistent performance across varying dataset sizes, from small-scale (21 images) to large-scale (143 images) reconstructions. The quality of results remains robust regardless of the input scale, indicating the method's versatility and reliability.

4.2 Bundle Adjustment (BA) Analysis

In the following part, we present the results obtained from applying BA to our dataset. These results demonstrate improvements in key metrics such as point accuracy, observation consistency, and overall model reliability. By comparing models with and without BA, we highlight the significant impact of this process on enhancing the precision of our 3D reconstructions.

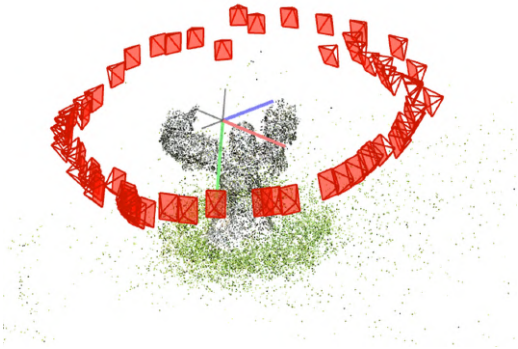
4.2.1 ***Statistic Analysis.*** Both models start with the same number of cameras and images, ensuring a fair comparison. The model with BA demonstrates an increase in the number of reconstructed points and observations, suggesting that BA enhances the density and detail of the 3D model. Additionally, the mean track length is improved, indicating that points are observed across more images, which enhances the robustness of the reconstruction. More observations per image in the BA model reflect better feature coverage. Interestingly, while the reprojection error is slightly higher with BA, this could indicate a focus on overall structural consistency and better outlier management.

Metric	With BA	No BA
Cameras	65	65
Images	65	65
Registered Images	65	65
Points	33,339	32,659
Observations	129,434	109,305
Mean Track Length	3.88	3.35
Mean Observations per Image	1,991.29	1,681.62
Mean Reprojection Error	1.31px	1.21px

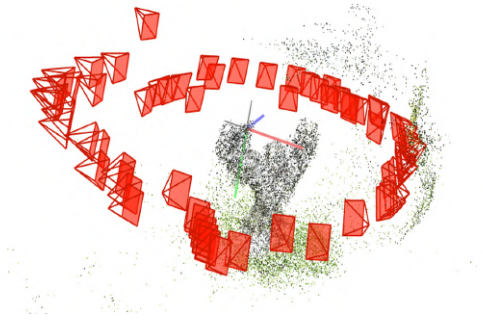
Table 1: Comparison of models with and without BA

4.2.2 ***Image Result.*** Figure 24 illustrate the impact of Bundle Adjustment (BA) on 3D reconstruction of a Sculpture formation. In the reconstruction with BA, the camera positions and the structure appear more consistent and well-defined, with a denser and more accurate point cloud representing the Sculpture. This suggests that BA effectively refines the camera parameters and point positions, leading to improved alignment and detail capture.

In contrast, the reconstruction without BA shows less uniformity in the camera positions and a sparser point cloud. This indicates potential inaccuracies and less precise alignment, which can result in a less reliable representation of the structure. Overall, BA enhances the accuracy and clarity of the 3D reconstruction, providing a more coherent and detailed model.



(a) Sculpture formation reconstruction with BA



(b) Sculpture formation reconstruction without BA

Figure 24: Comparison of images with and without BA

4.2.3 ***Benefits of Bundle Adjustment (BA) in Reconstruction.*** Bundle Adjustment (BA) significantly enhances the reconstruction process by optimizing various metrics, resulting in a more complete and reliable model.

- **Retention of 3D Points:** The model retains more 3D points, contributing to a denser and more detailed reconstruction.
- **Increased Observations:** A higher number of observations ensures a robust dataset, improving overall model accuracy.

- **Longer Feature Track:** Extended feature tracking results in improved alignment and consistency across frames.
- **More Reliable Structure:** On average, each 3D point is observed by more images, enhancing the structural reliability.
- **Higher Observations per Image:** Each image contributes a larger number of observations, strengthening geometric constraints.
- **Slight Increase in Reprojection Error:** A 0.1px increase in reprojection error is a reasonable trade-off for the benefits gained.
- **Enhanced Completeness and Reliability:** The trade-off results in a balance between quality and completeness, yielding a more robust reconstruction.

5 Future Direction

Through comprehensive literature review[8] and examination of relevant research materials[2], we have successfully implemented our own incremental Structure from Motion (SFM) pipeline as mentioned above. However, there remain multiple areas with substantial potential for optimization and enhancement.

5.1 Image Organization Structure

The constraints and drawbacks for our framework are mainly caused by current organization structure of images. In some state-of-art incremental SFM frameworks *e.g.* openMVG or COLMAP, the image relationships are structured and organized through graph-based architecture, enabling efficient querying and traversal capabilities. One of graph-based architectures adopted by openMVG is undirected connected graph to organize image relationships. Vertices in this graph represents the individual images and Edges denote pairwise image matching relationship, of which weights are computed based on feature matching quantity and geometric verification quality.

In contrast, initially our framework did not incorporate this architectural consideration, relying solely on temporal sequence for image management. The absence of a robust architectural foundation subsequently imposed significant constraints on development progression and system scalability. For example, when the pipeline goes on selecting initial pairwise images for initial reconstruction, it's difficult and high-cost for us to find the proper ones which is executed by searching on double loops for computing scores of every pair of images. In addition, the rudimentary view organizations also cause lots of troubles when selecting the next best view in incremental reconstruction phase which have similar causes and effects with finding good initial pairwise images.

More unfortunately, another significant architectural limitation of our system means its restricted capacity to store only one matching image per reference image, in contrast to the comprehensive matching capabilities of openMVG and COLMAP, which utilize undirected graph structures to record all viable image matches.

5.2 Finding good initial pair

The selection of initial image pairs is crucial for incremental Structure from Motion (SfM), as these pairs serve as the baseline reference for reconstruction and directly impact subsequent camera pose estimation and triangulation accuracy. Any errors introduced at this stage propagate and accumulate throughout the incremental process, ultimately determining both the success rate and final precision of the entire reconstruction.

However, not only our rudimentary image organizations lead to constraints, but also our evaluation criterion appears comparatively primitive and lacks sufficient robustness, where numbers of matches between pairwise images are directly set as scores. Compared to our evaluation method, openMVG employs a more reliable and robust mechanism based on the fundamental matrix (F) scoring through geometric verification, prioritizing image pairs with abundant feature matches and uniform spatial distribution. Besides, COLMAP utilizes another scoring criterion which depends on the inlier ratio during relative pose estimation while simultaneously evaluating both the quantity and spatial distribution of triangulated points.

5.3 Selecting next best view

In incremental Structure from Motion (SFM) reconstruction, the selection of optimal next views is critical for ensuring robustness and precision. By prioritizing views with superior co-visibility relationships and geometric configurations, this selection process minimizes error accumulation and enhances the quality of triangulated points, thereby ensuring reconstruction continuity and overall effectiveness.

Compared to our primitive selection method mentioned above, for example, the selection employed in COLMAP is implemented through a priority queue system based on co-visibility relationships. This queue determines the registration sequence by synthesizing multiple evaluation metrics, including: the quantity of feature matches with already reconstructed views, the number of potential new triangulated points, baseline distances between views, and anticipated reprojection errors. The robust mechanisms in those state-of-art frameworks can be viewed as exemplars for reference and implementation guidance.

5.4 Correspondences Search

As discussed in previous section 3.4.3, we use tracking on different levels of indexing to facilitate collecting all correspondent 3D points for current view in incremental reconstructions. This searching algorithm can work well but still exhibits several limitations that warrant further refinement.

What COLMAP or openMVG have achieved for correspondent search can be paradigmatic model for reference, which depends on their image organization structures. Following the selection of the next best view, the system establishes correspondences between this new view and the existing 3D point cloud. This correspondence establishment leverages existing feature matching relationships as intermediaries - specifically, when a feature point in the new view matches with a feature point in an already reconstructed view, and that reconstructed feature point is associated with a 3D point, a correspondence between the new view's feature point and the 3D point can be established. The system evaluates the reliability of these 2D-3D correspondences through multiple criteria, including reprojection error, observation angles, and descriptor distances. It can be indicated that the critical point is the well-designed image

organization structure which contributes to robust methods of representations and storage paradigm of matches. Robust methods of matches enable enhanced mining more information between images, facilitating more efficient and effective correspondent search of 3D points.

5.5 Next Stage: Dense Reconstruction

The aforementioned work focuses exclusively on sparse reconstruction methodologies. The dense reconstruction phase was implemented through the integration of COLMAP's existing framework, rather than through independent development, serving as the basis for subsequent improvements.

Sparse reconstruction primarily focuses on determining 3D positions of feature points (such as SIFT keypoints) and camera pose estimation, yielding a skeletal structure of the scene. In contrast, dense reconstruction endeavors to reconstruct every pixel within the images, employing multi-view stereo matching to estimate depth values for each pixel based on known camera poses, ultimately producing a more complete and detail-rich three-dimensional scene model. In essence, dense reconstruction builds upon the foundation established by sparse reconstruction, utilizing intensive pixel-level matching to populate scene details.

Dense reconstruction pipeline of COLMAP builds upon the sparse reconstruction foundation, incorporating several additional critical stages:

- The process initiates with normal and depth map estimation: A reference image is selected, followed by depth value and normal vector estimation for each pixel through plane sweeping and photometric consistency verification across multiple adjacent views. This estimation process evaluates matching quality across different depth hypotheses utilizing similarity metrics such as Normalized Cross Correlation (NCC).
- Following the completion of single-view depth and normal map estimation, the pipeline proceeds to multi-view depth map fusion, integrating depth maps estimated from different viewpoints into a coherent dense point cloud.
- This fusion process incorporates both depth consistency and geometric consistency checks to eliminate noise and erroneous reconstructions. Optionally, the point cloud can undergo Poisson surface reconstruction to generate a complete mesh model.
-

This comprehensive process effectively utilizes camera poses obtained from sparse reconstruction as known parameters, implementing multi-view stereo matching to reconstruct every pixel within the scene, thereby achieving a more complete scene reconstruction. The following Figures 25,26 and 27 display dense reconstructions of the rock, the sculpture and the obelisk with the help of COLMAP.

Figure 25: Dense rock formation reconstruction.

Figure 26: Dense rock formation reconstruction.

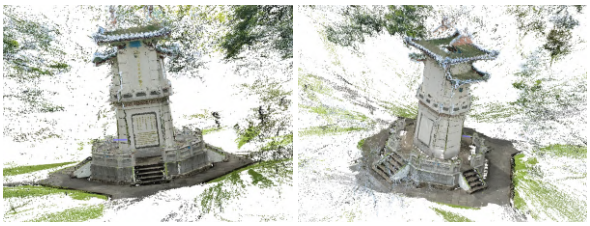


Figure 27: Dense rock formation reconstruction.

6 Summary

Through the whole process of research and implementations, we learn a lot of mechanisms and trick points in pipeline of Incremental SFM. The process of implementing incremental reconstruction methodologies has deepened our understanding of conventional computer vision techniques and their foundational mathematical principles. The achievement of satisfactory sparse point cloud reconstruction results substantiates our methodological experiment, encouraging continued research endeavors in this field.

References

[1] Chum, O., and Matas, J. Matching with prosac-progressive sample consensus. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (2005), vol. 1, IEEE, pp. 220–226.

[2] Gao, X., Zhang, T., Liu, Y., and Yan, Q. *14 Lectures on Visual SLAM: From Theory to Practice.* Publishing House of Electronics Industry, 2017.

[3] Hartley, R., and Zisserman, A. *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[4] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision 60* (2004), 91–110.

[5] Moulon, P., Monasse, P., and Marlet, R. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the IEEE international conference on computer vision* (2013), pp. 3248–3255.

[6] OpenCV Documentation. Camera calibration and 3d reconstruction, 2024. Accessed: 2024-11-20.

[7] OpenCV Documentation. Epipolar geometry and stereo images, 2024. Accessed: 2024-11-20.

[8] Schonberger, J. L., and Frahm, J.-M. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 4104–4113.

[9] Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence 22*, 11 (2000), 1330–1334.

A Appendix. Code Snippets

Complete Code Repository locates on:
https://github.com/ryougishiki1999/NTU_AI6121_Project_SFM_2024fall

A.1 Camera Calibration

Listing 1: Content of `camera_calibration.py`

```python
def _run_calibration_and_save_result(self):
    print("Running calibration and saving result...")
    for pattern_img_path in self.pattern_img_paths:
        pattern_img = cv.imread(pattern_img_path)
        gray_pattern_img = cv.cvtColor(pattern_img, cv.COLOR_BGR2GRAY)

        ret, corners = cv.findChessboardCorners(gray_pattern_img, self.pattern_size, None)

        if ret == True:
            self.objps_list.append(self.objps)

            corners2 = cv.cornerSubPix(gray_pattern_img, corners, (11, 11), (-1, -1), self.
                criteria)
            self.imgps_list.append(corners2)

            cv.drawChessboardCorners(pattern_img, self.pattern_size, corners2, ret)
            calibration_pattern_img = "calibration_" + os.path.basename(pattern_img_path)
            calibration_pattern_img_path = os.path.join(DATA_CALIBRATION_DIR_PATH,
                calibration_pattern_img)
            cv.imwrite(calibration_pattern_img_path, pattern_img)
```

A.2 Initial Reconstruction

Listing 2: Content of `init_reconstruction.py`

```python
def _reconstruct_by_consecutive_pair(self, consecutive_pair):
    img1, img2, img1_idx, img2_idx = consecutive_pair

    kp1, desc1 = extract_SIFT_features(img1)
    kp2, desc2 = extract_SIFT_features(img2)

    pts1, pts2, _ = match_SIFT_features(desc1, desc2, kp1, kp2)
    E, _ = cv.findEssentialMat(pts1, pts2, self.K, cv.RANSAC, 0.99, 1.0)

    update_good_matches_dict(self.good_mathces_dict, img2_idx, pts2, pts1)
    ...

    _, R, tvec, _ = cv.recoverPose(E, pts1, pts2, self.K)
    rvec1, tvec1 = recover_camera_pose(self.camera_poses[img1_idx])
    R1 = cv.Rodrigues(rvec1)[0]
    R2, tvec2 = R @ R1, R @ tvec1 + tvec
    rvec2 = cv.Rodrigues(R2)[0]
    self.camera_poses[img2_idx] = generate_camera_pose(rvec2, tvec2)
    ...

    P1 = generate_projection_matrix(rvec1, tvec1, self.K)
    P2 = generate_projection_matrix(rvec2, tvec2, self.K)

    points4D = cv.triangulatePoints(P1, P2, pts1.T, pts2.T)
    points3D = points4D[:3] / points4D[3]
```

```
    points3D = points3D.T
    ...

    return points3D, pts2
```

A.3 Incremental Reconstruction

Listing 3: Content of `sfm_engine.py`

```
def _incremental_reconstruction(self, query_idx, train_idx):
        query_img = self.imgs[query_idx]
        train_img = self.imgs[train_idx]

        query_kpts, query_pts_desc = extract_SIFT_features(query_img)
        train_kpts, train_pts_desc = extract_SIFT_features(train_img)

        query_pts, train_pts, good_matches = match_SIFT_features(query_pts_desc,
            train_pts_desc, query_kpts, train_kpts)
        update_good_matches_dict(self.good_matches_dict, train_idx, train_pts, query_pts)
        print("Number of good matches", len(good_matches))

        total_points3D, matched_mask, incremental_mask =\
            match_3d_by_2d(self.unqiue_points3D_dict, self.good_matches_dict, query_idx,
                query_pts)

        matched_points3D = total_points3D[matched_mask]
        matched_train_pts = train_pts[matched_mask]

        print(f"Number of correspondence between existed 3Dpoints and 2D feature points from
            img{train_idx}: ", np.count_nonzero(matched_mask))
        print(f"Number of incremental 2D feature points from img{train_idx}: ", np.
            count_nonzero(incremental_mask))


        incremental_query_pts, incremental_train_pts = query_pts[incremental_mask], train_pts
            [incremental_mask]

        _, train_rvec, train_tvec, _ = cv.solvePnPRansac(matched_points3D, matched_train_pts,
             self.K, None, confidence=0.999)
        query_rvec, query_tvec = recover_camera_pose(self.camera_poses[query_idx])
        ...

        self.camera_poses[train_idx] = generate_camera_pose(train_rvec, train_tvec)
        print(f"camera pose of img {train_idx}: ", cv.Rodrigues(train_rvec)[0], train_tvec)


        P_query = generate_projection_matrix(query_rvec, query_tvec, self.K)
        P_train = generate_projection_matrix(train_rvec, train_tvec, self.K)

        new_Points4D = cv.triangulatePoints(P_query, P_train, incremental_query_pts.T,
            incremental_train_pts.T)
        new_Points3D = new_Points4D[:3] / new_Points4D[3]
        new_Points3D = new_Points3D.T

        unique_mask = update_unique_points3D_dict(self.unqiue_points3D_dict, train_idx,
            new_Points3D, incremental_train_pts)

```

```
        print("number of incremental points3D: ", new_Points3D.shape[0])
        print("number of unique incremental points3D: ", new_Points3D[unique_mask].shape[0])
```

A.4 Image Organization Structure

Listing 4: Content of `points3D_update.py`

```

def update_good_matches_dict(good_matches_dict, train_img_idx, train_pts, query_pts):
    """_summary_

    Args:
        good_matches_dict (_type_): key:trainImg idx, value: dict{key: trainImg pt, value:
            queryImg pt},
        trainImg idx from 1 to N_SFM_IMGS-1
    """
    if train_img_idx not in good_matches_dict:
        good_matches_dict[train_img_idx] = dict()

    for train_pt, query_pt in zip(train_pts, query_pts):
        good_matches_dict[train_img_idx][tuple(train_pt)] = tuple(query_pt)

...

def update_unique_points3D_dict(unqiue_points3D_dict, train_img_idx, new_points3D, new_pts,
    distance_threshold=1e-2):
    print("update unique points3D dict, train_img_idx: ", train_img_idx)
    all_unique_points3D = get_all_unique_points3D(unqiue_points3D_dict)

    unique_mask = np.ones(new_points3D.shape[0], dtype=bool)

    all_unique_points3D = np.ascontiguousarray(all_unique_points3D)
    new_points3D = np.ascontiguousarray(new_points3D)
    tree = cKDTree(all_unique_points3D)
    distances, _ = tree.query(new_points3D, k=1)
    unique_mask = distances > distance_threshold


    # for i, new_point3D in enumerate(new_points3D):
    #     for _, unique_point3D in enumerate(all_unique_points3D):
    #         if np.array_equal(new_point3D, unique_point3D):
    #             unique_mask[i] = False

    new_unqiue_points3D = new_points3D[unique_mask]
    new_unqiue_pts = new_pts[unique_mask]

    if train_img_idx not in unqiue_points3D_dict:
        unqiue_points3D_dict[train_img_idx] = dict()

    for unique_pt, unique_point3D in zip(new_unqiue_pts, new_unqiue_points3D):
        unqiue_points3D_dict[train_img_idx][tuple(unique_pt)] = np.array(unique_point3D,
            dtype=np.float64)

    return unique_mask

def match_3d_by_2d(unique_points3D_dict, good_matches_dict, src_idx, src_pts):

```

```
    def index_start_pt_on_target_idx(src_pt, target_idx):

        def recursive_indexing(pt, idx):

            if idx == target_idx:
                indexed_pt = pt
                return indexed_pt
            else:
                good_matches = good_matches_dict[idx]
                if tuple(pt) in good_matches:
                    indexed_pt = good_matches[tuple(pt)]
                    nxt_idx = idx - 1
                    return recursive_indexing(indexed_pt, nxt_idx)
                else:
                    return None

        return recursive_indexing(src_pt, src_idx)

    def match_on_target_idx(target_idx):

        unique_point3D_dict = unique_points3D_dict[target_idx]
        for i, start_pt in enumerate(src_pts):
            if matched_mask[i]:
                continue
            else:
                indexed_pt = index_start_pt_on_target_idx(start_pt, target_idx)
                if indexed_pt is not None and tuple(indexed_pt) in unique_point3D_dict:
                    matched_mask[i] = True
                    incremental_mask[i] = False
                    total_points3D[i] = unique_point3D_dict[tuple(indexed_pt)]

    matched_mask = np.zeros(src_pts.shape[0], dtype=bool)
    incremental_mask = np.ones(src_pts.shape[0], dtype=bool)
    total_points3D = np.zeros((src_pts.shape[0], 3), dtype=np.float64)

    # Correspondent Search from image src_idx to 1, 1 is the first image.
    cur_idx, end_idx = src_idx, 1
    while cur_idx >= end_idx:
        match_on_target_idx(cur_idx)
        cur_idx -= 1

    return total_points3D, matched_mask, incremental_mask
```

B Appendix. Original Image Examples

B.1 Rock

The rock is located in front of The Hive.

B.2 Sculpture

This stone sculpture is located in the Yunnan Garden.

B.3 Monument

This Founding Monument is located in the center of Yunnan Garden, which symbolizes the institution's transformation from its roots in Nanyang University (Nantah), founded in 1955, to its official establishment as NTU in 1991.

Figure 28: Original Images of Rock.

Figure 29: Original Images of Sculpture.

Figure 30: Original Images of Sculpture.