Incremental Structure from Motion (SfM) for 3D Reconstruction

Adel KANSO - Mariam Janbein

Abstract

This project implements an **Incremental Structure from Motion (SfM)** pipeline for 3D scene reconstruction using monocular image sequences. It supports both pre-calibrated datasets and custom datasets (with calibration via a checkerboard). The system progressively reconstructs a sparse 3D point cloud, recovers camera trajectories, and performs bundle adjustment and colorization of points. The project combines computer vision techniques including feature detection, epipolar geometry, triangulation, PnP pose estimation, and nonlinear optimization.

Introduction

Structure from Motion (SfM) is a core technique in computer vision for reconstructing 3D structure from 2D image sequences. This project focuses on an **incremental** SfM approach, adding images one-by-one to expand the reconstruction. The pipeline handles:

- Camera calibration (for custom datasets),
- SIFT feature detection and matching,
- Initial 3D reconstruction from image pairs,
- Progressive registration of additional views,
- Bundle adjustment,
- Visualization of 3D point clouds and camera trajectories.

The project is implemented in Python using OpenCV, SciPy, and Open3D.

About Precalibrated Dataset

The dataset comprises several scenes designed to evaluate the performance of Structure-from-Motion (SfM) algorithms. Each scene includes pre-calibrated multi-view image sequences. These scenes vary in complexity and texture, encompassing both indoor and outdoor environments, to provide a comprehensive benchmark for assessing the accuracy and robustness of SfM pipelines. A K file that contains the matrix is included with the dataset.

Theory

Camera Calibration

Camera calibration estimates the intrinsic matrix (K) to model the camera's internal parameters. For custom datasets, calibration is performed using 8–10 checkerboard images with OpenCV's `cv2.calibrateCamera` and `cv2.findChessboardCorners`. The intrinsic matrix is scaled by a downscaling factor (2.0) to account for image resizing. Note that lens distortion correction is not applied in this implementation for simplicity, assuming minimal distortion in custom datasets.

Feature Detection & Matching

Features are detected using SIFT and matched via brute-force matching with Lowe's ratio test (threshold 0.7). The fundamental matrix (F) is estimated using RANSAC to reject outliers, and the essential matrix (E) is computed as: $$ E = K^T F K $$

Initial Reconstruction

Two views are used to recover relative rotation and translation from (E) using `cv2.recoverPose`, which validates solutions via cheirality (ensuring positive depth). Triangulation is performed using: $$ x_i = P_i X, \quad x_j = P_j X $$

Incremental Camera Registration (PnP)

Each new image is registered via Perspective-n-Point (PnP) solving using `cv2.solvePnPRansac` with the iterative method: $$ \min_{R,t} \sum_i | x_i - \pi(K(RX_i + t)) |^2 $$

Triangulation & Bundle Adjustment

New points are triangulated between registered views using `cv2.triangulatePoints`. Bundle adjustment, which is optional and controlled via the GUI, minimizes reprojection errors using `scipy.optimize.least_squares`: $$ \min_{{R_i, t_i, X_j, K}} \sum_{i,j} | x_{ij} - \pi(K(R_i X_j + t_i)) |^2 $$ In this implementation, the intrinsic matrix (K) is included in the optimization, allowing slight adjustments to focal length and principal point. This may improve accuracy for custom datasets with noisy calibration but could introduce inconsistencies for pre-calibrated datasets where (K) is assumed fixed. Future improvements could include an option to fix (K) for pre-calibrated datasets.

Colorization

3D points are colored by sampling RGB values from the 2D image points in the current image that correspond to the triangulated 3D points. These 2D points, obtained during triangulation, are used directly to sample pixel values from the current image, avoiding the need to re-project 3D points with `cv2.projectPoints`. Colors are sampled in BGR format (as loaded by OpenCV) and normalized for visualization in Open3D, though BGR-to-RGB conversion is not explicitly performed, which may affect color accuracy in visualizations. Colors are not averaged across multiple views, simplifying the process but potentially leading to inconsistencies under varying lighting conditions.

Methodology

The *Structure from Motion (SfM)* pipeline implemented in this study reconstructs a 3D scene from a set of unordered images. The methodology consists of the following key stages:

1. Initialization and Camera Calibration

Choose between pre-calibrated or custom dataset. If custom, intrinsic matrix (K) is estimated using checkerboard images by detecting corners findChessboardCorners and then calibrating. Case of pre-calibrated, the key is given. In order to reduce resources consumption and computation time, images are being down scaled by 2.0 factor

2. Initial Image Pair Selection and Feature Matching

Two images are selected to initiate the reconstruction. These images are downscaled to reduce computational load.
SIFT is used to detect and match keypoints between the two images, Brut Force Matcher to match descriptors, The *Fundamental Matrix* is estimated using RANSAC to eliminate outliers, and the *Essential Matrix* is computed and decomposed to recover the relative rotation and translation between the first two views.

3. Initial Triangulation

With the camera intrinsics and relative pose, 3D points are triangulated from the matched 2D keypoints. These 3D points are then evaluated using *re-projection error* to ensure geometric consistency.

4. Pose Estimation using PnP

The *Perspective-n-Point (PnP)* algorithm is applied to estimate the pose of the second camera using the initial 3D points and corresponding 2D features.
This serves as a base for subsequent incremental camera pose estimation.

5. Incremental Camera Registration

The remaining images are processed iteratively. For each new image:

- Keypoints are matched with the previous image
- Common feature points are identified using cross-matching
- The camera pose is estimated using PnP
- 3D points are triangulated with the newly estimated pose
- Reprojection error is computed to validate the new points

6. Bundle Adjustment

If enabled, *bundle adjustment* is performed after each new pose registration.
It refines both camera poses and 3D point positions by minimizing the re-projection error using least_squares.
This optimization is crucial for improving the accuracy and consistency of the reconstructed structure.

7. Color Accumulation and Visualization Preparation

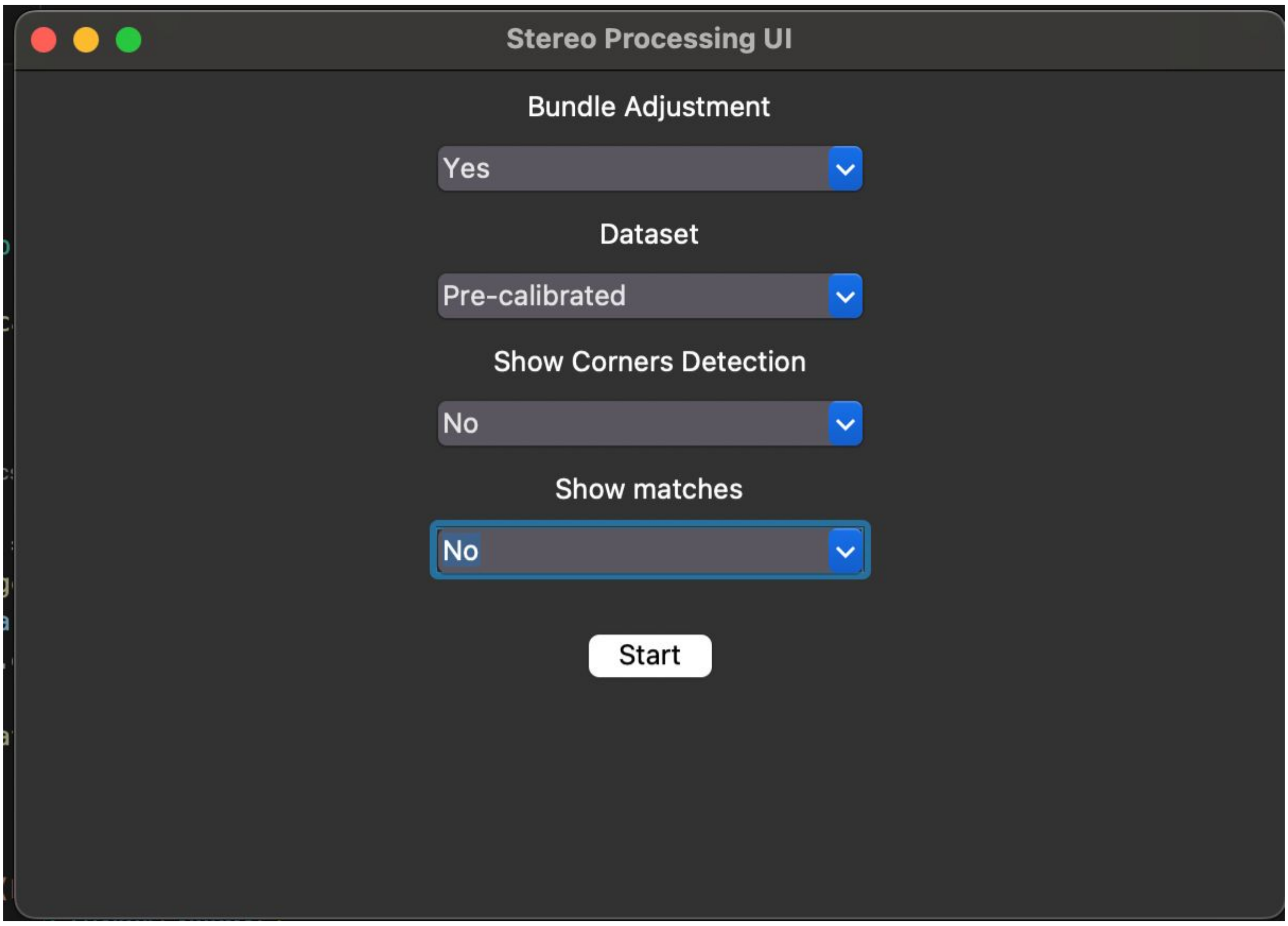
For each valid 3D point, color is extracted from the image to facilitate visualization.
The camera extrinsic matrices are stored in a format compatible with *Open3D* for later rendering.
The error at each step is logged for analysis.

8. Result Visualization

Once all images have been processed:

- The reconstruction results (3D points, colors, camera poses, and re-projection errors) are visualized and analyzed and saved in .ply format.

GUI allows interactive selection of dataset type, bundle adjustment, visualization of checkerboard corners, and feature matches.


Stereo Processing UI

Bundle Adjustment: Yes

Dataset: Pre-calibrated

Show Corners Detection: No

Show matches: No

Results

A. **Pre-Calibrated Dataset Results**

- **Camera Matrix (K):**

```
[[1.37974e+03 0.00000e+00 7.60345e+02]
 [0.00000e+00 1.38208e+03 5.03405e+02]
 [0.00000e+00 0.00000e+00 1.00000e+00]]
```

- **Fundamental Matrix (RANSAC):**

```
[[-1.62834568 -6.7726362  -0.40500172]
 [ 8.29667134 -0.36149299 -1.74263939]
 [-0.19144145 -0.91671512 -0.06076527]]
```

- **Rotation Matrix:**

```
[[ 0.93405847 -0.10643337 -0.34089105]
 [ 0.1400502   0.98726166  0.07550064]
 [ 0.32851288 -0.11826388  0.93706614]]
```

- **Translation Vector:**

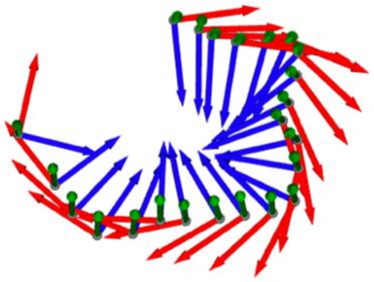
```
[[-0.13395233]
 [-0.00342375]
 [ 0.99098186]]
```

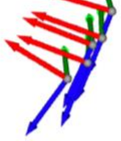
- **Reprojection Errors (Samples):**

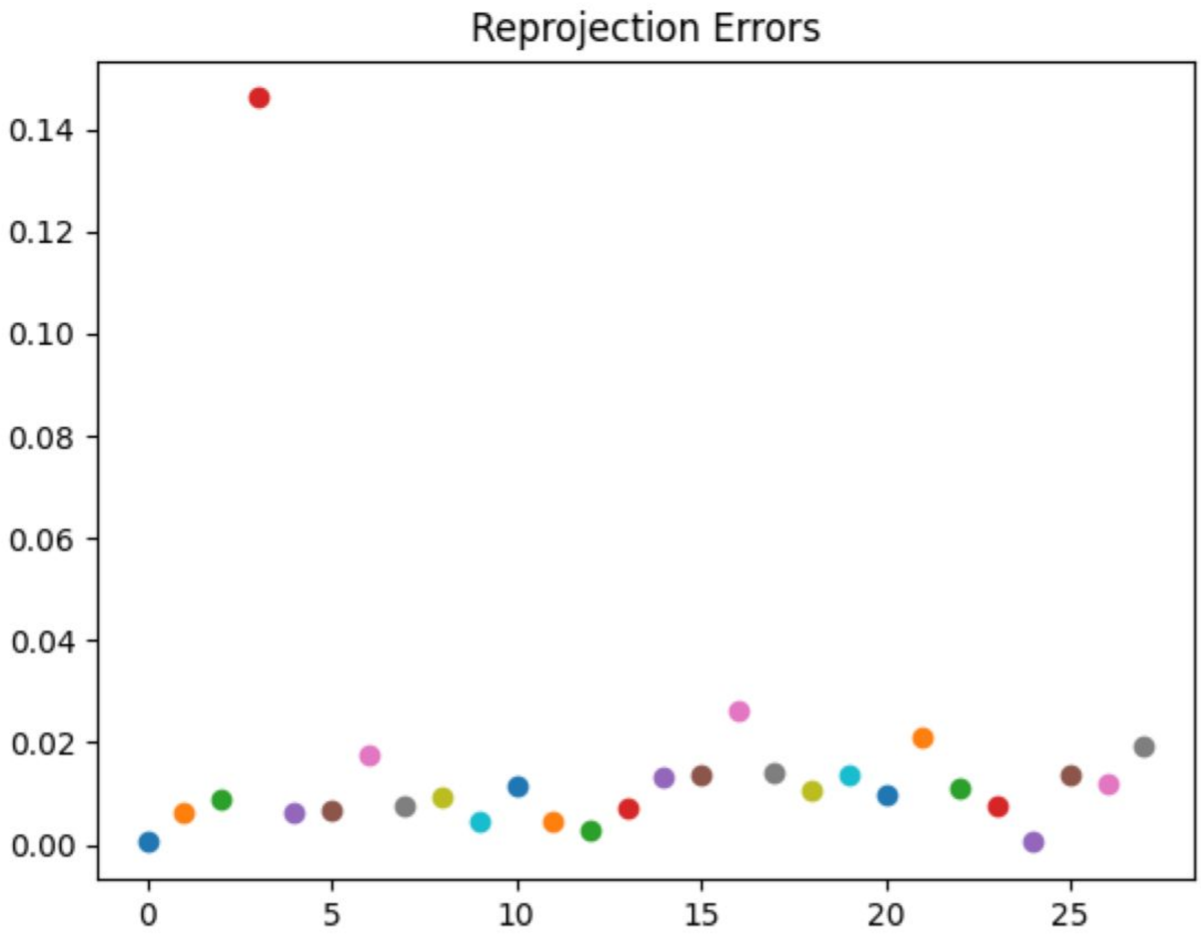
```
Initial: 0.36041085714306625
Intermediate: 0.749, 0.518, 3.99, 1.09, 0.47 ...
Final Bundle Error (min): ~0.0006
```

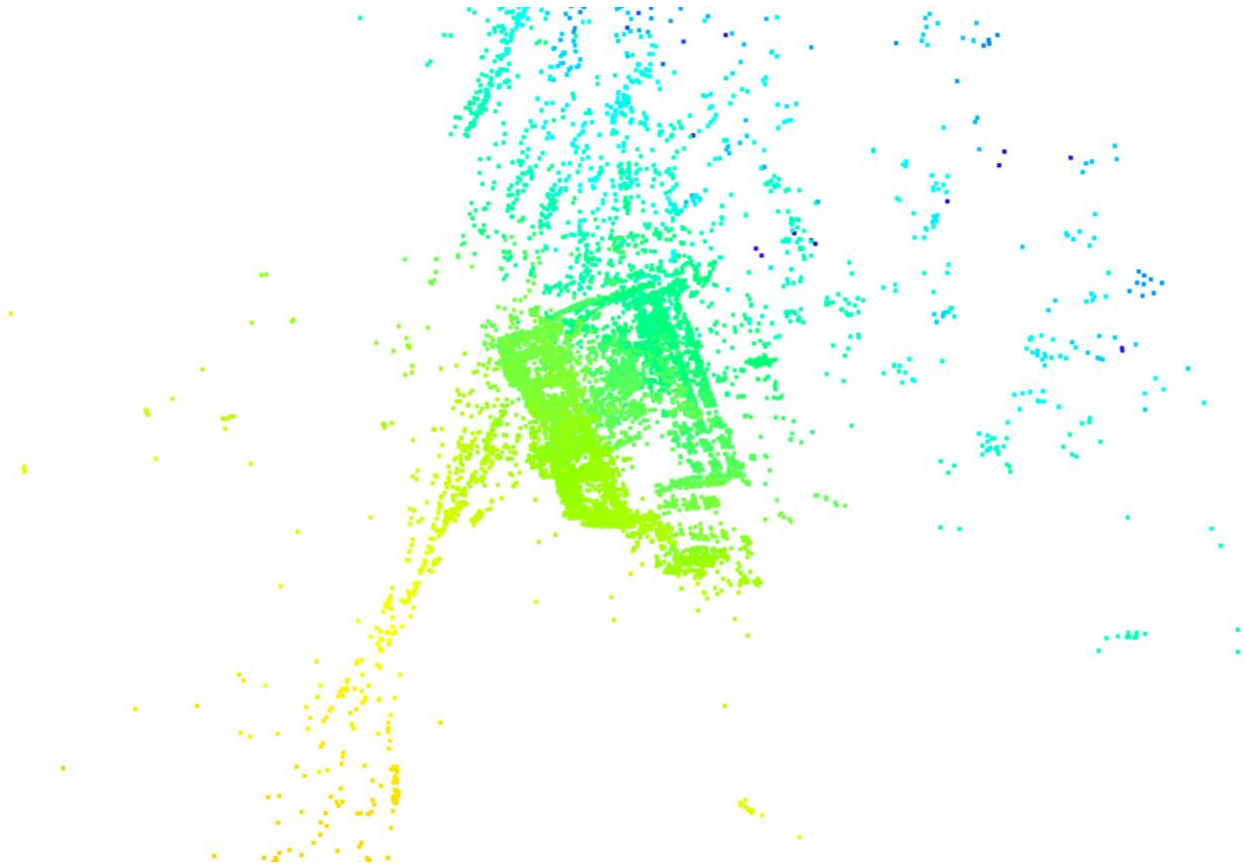
- **Visualizations**:
    - Uncolored and colored point clouds saved as `no_colors.ply` and `colors.ply`.
    - Camera trajectory rendered via Open3D, showing smooth camera motion but minor drift due to lack of loop closure.
    - Point clouds are sparse but capture the scene structure effectively, with colorization enhancing visual interpretability despite single-view sampling and potential BGR color display.

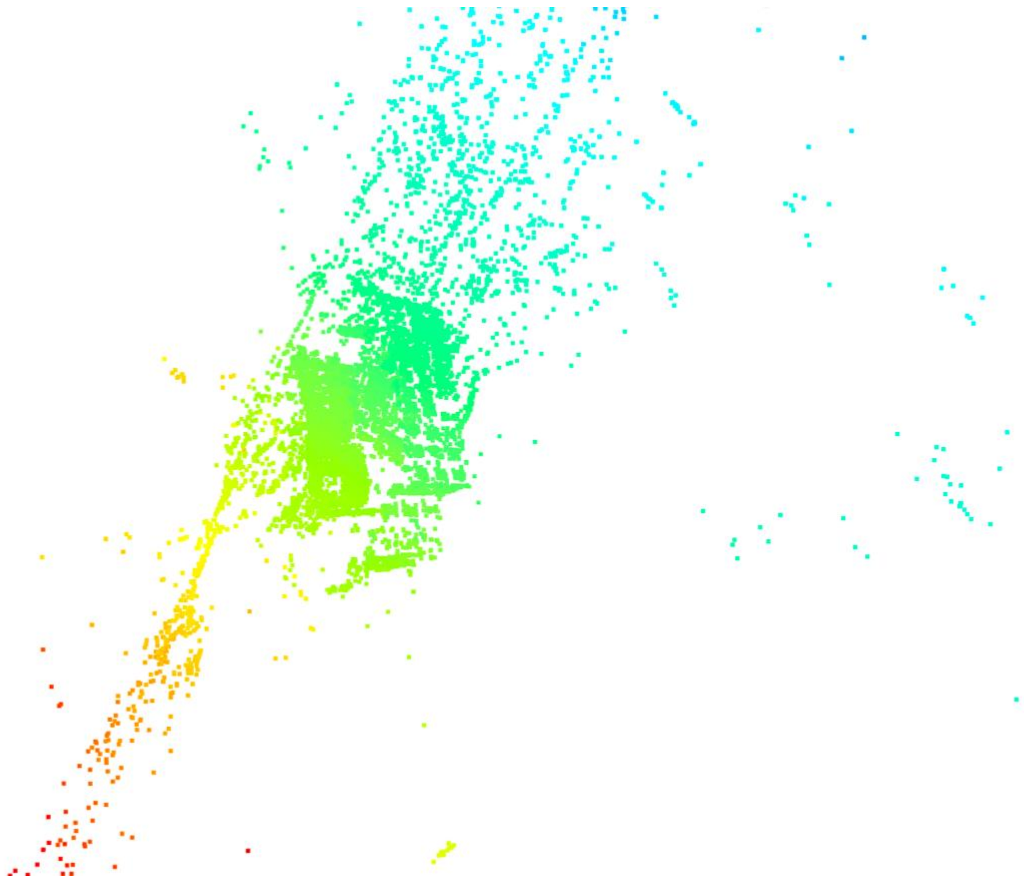
- **Visualization Results:**






Reprojection Errors
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25





B. **Calibrated Dataset Results**

- **Camera Matrix (K):**

```
[[1.560e+03 0.000e+00 8.368e+02]
 [0.000e+00 1.594e+03 9.506e+02]
 [0.00000e+00 0.000e+00 1.000e+00]]
```

- **Fundamental Matrix (RANSAC):**

```
[[ 0.38844368 -1.90189844  2.59917697]
 [ 3.43310505  0.20238321  9.56764704]
 [-2.5450532  -9.58769692  0.34359227]]
```

- **Rotation Matrix:**

```
[[ 9.90106055e-01 -9.26306852e-04 -1.40318002e-01]
 [-5.83416714e-04  9.99942393e-01 -1.07177831e-02]
 [ 1.40319846e-01  1.06936059e-02  9.90048477e-01]]
```

- **Translation Vector:**

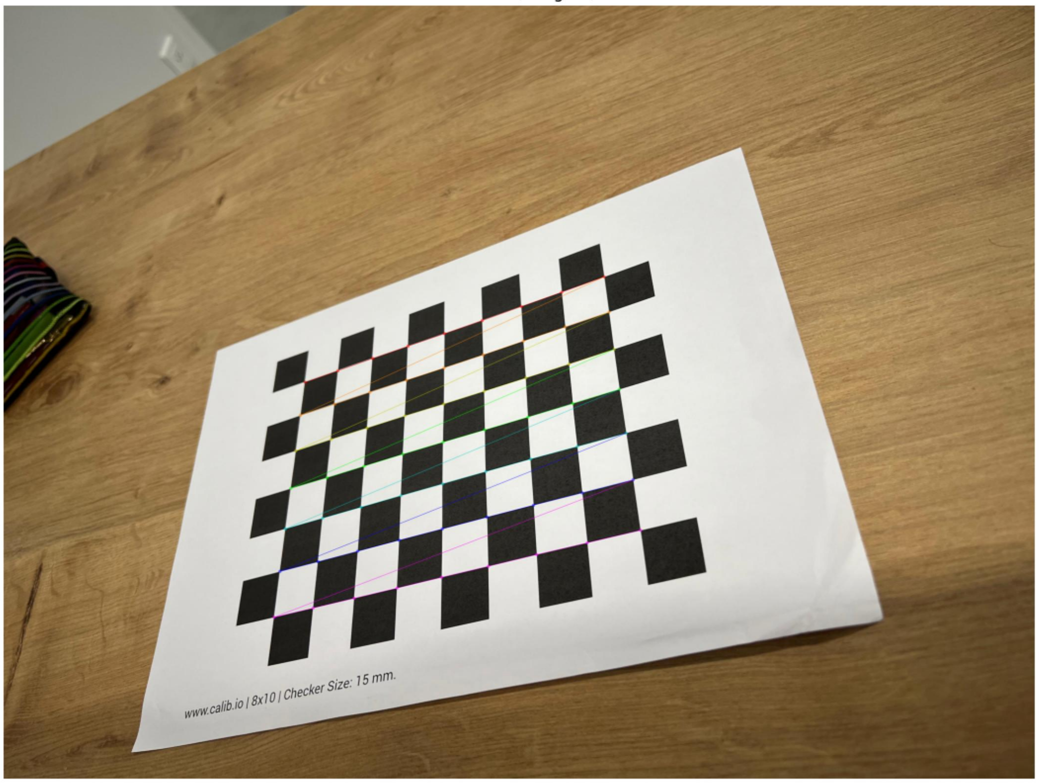
```
[[ 0.94853009]
 [-0.25073346]
 [-0.19345127]]
```

- **Reprojection Errors (Samples):**

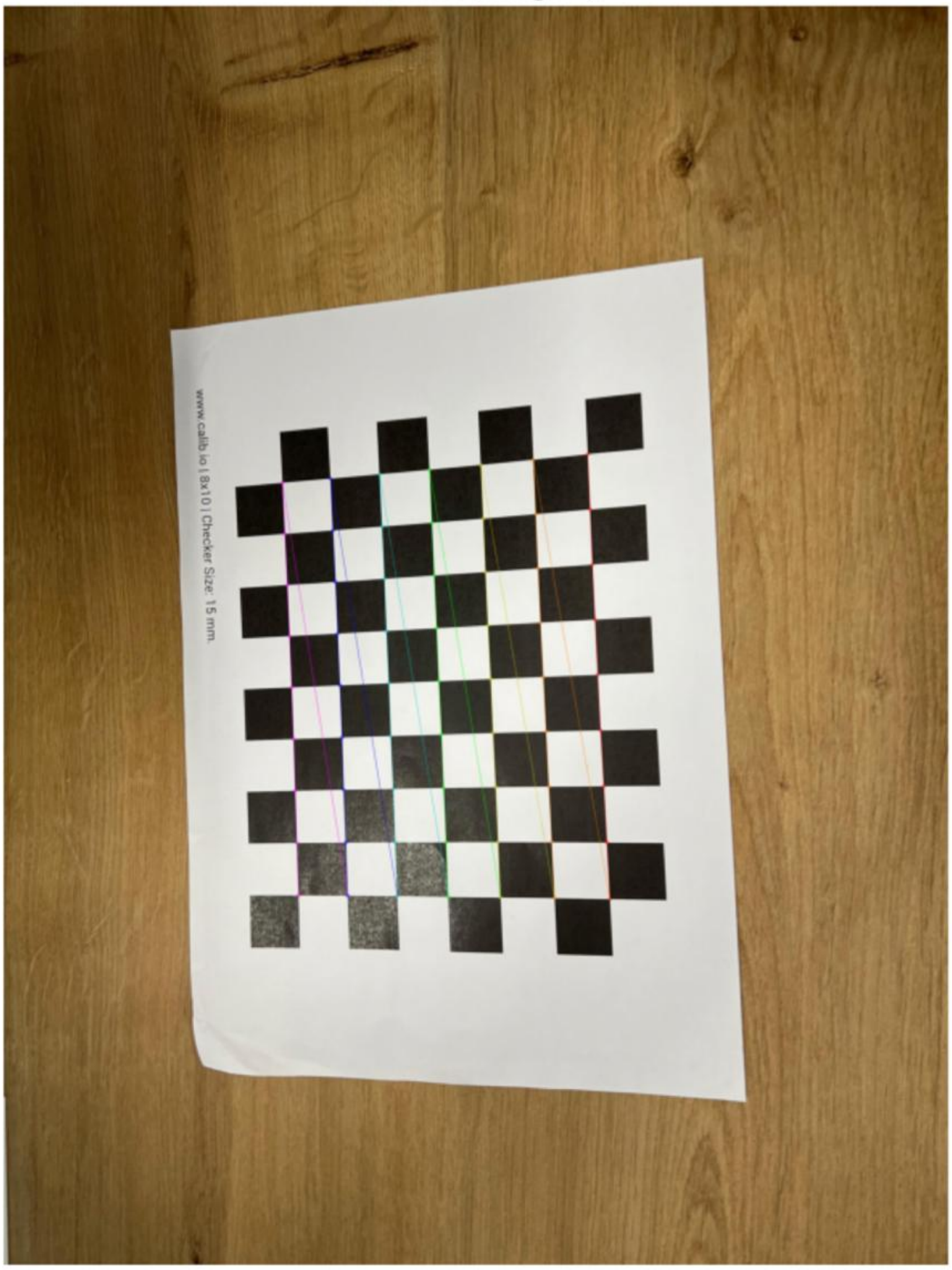
```
Initial: 0.17731041580129453
Intermediate: 0.381, 0.166, 0.022 ...
Final Bundle Error (min): ~0.002
```

- **Visualization Results**:
  - Similar to pre-calibrated results, point clouds are sparse but meaningful, with minor drift in trajectories. Colorization improves visual quality but may show inconsistencies due to single-view sampling and potential BGR color display.

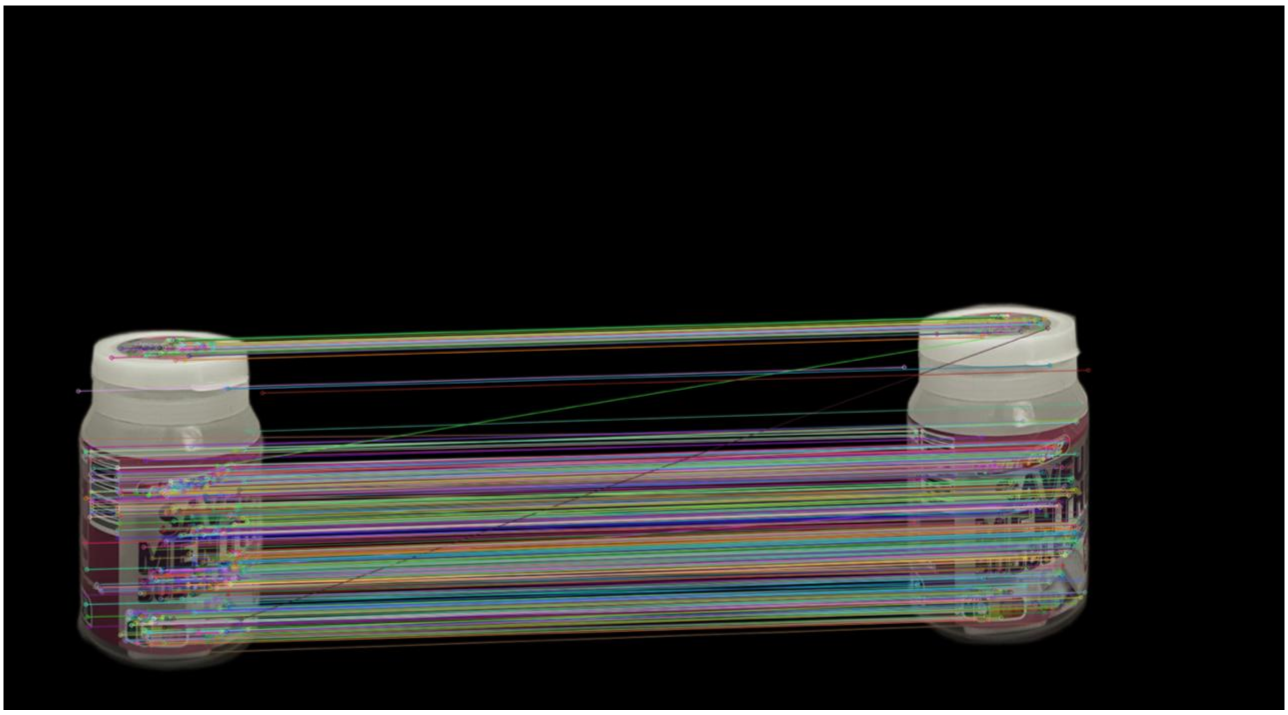
Corners (Image 3)

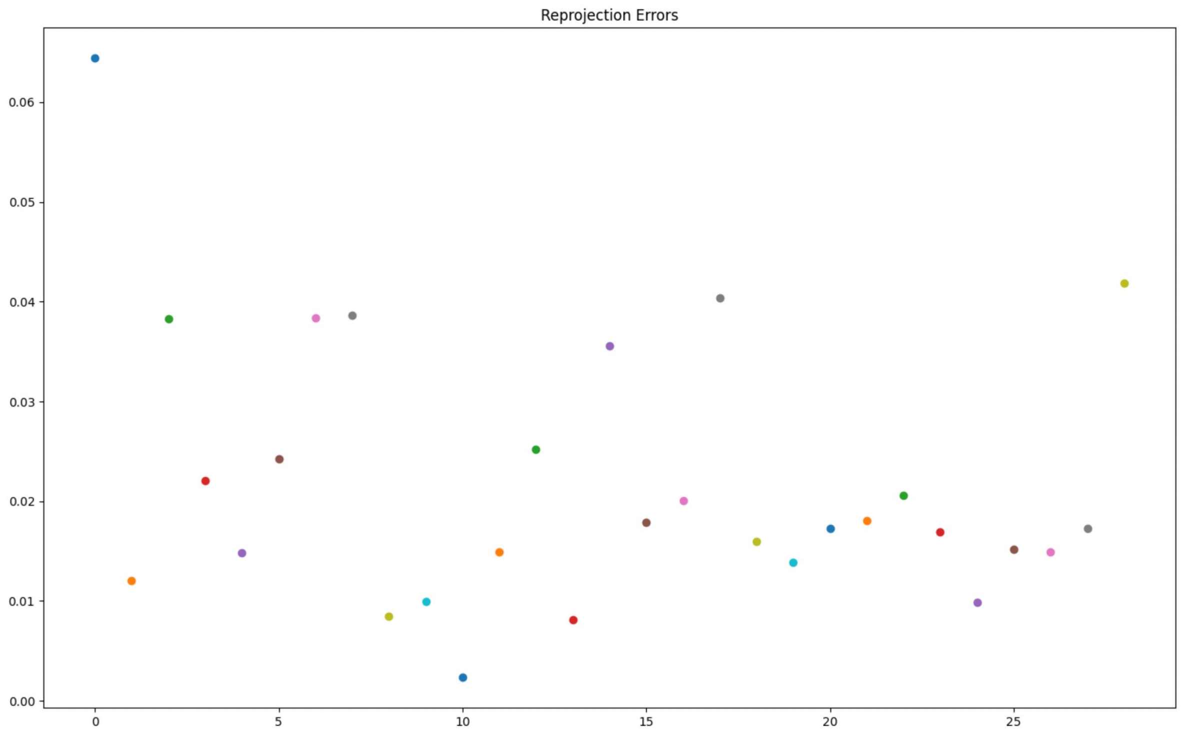


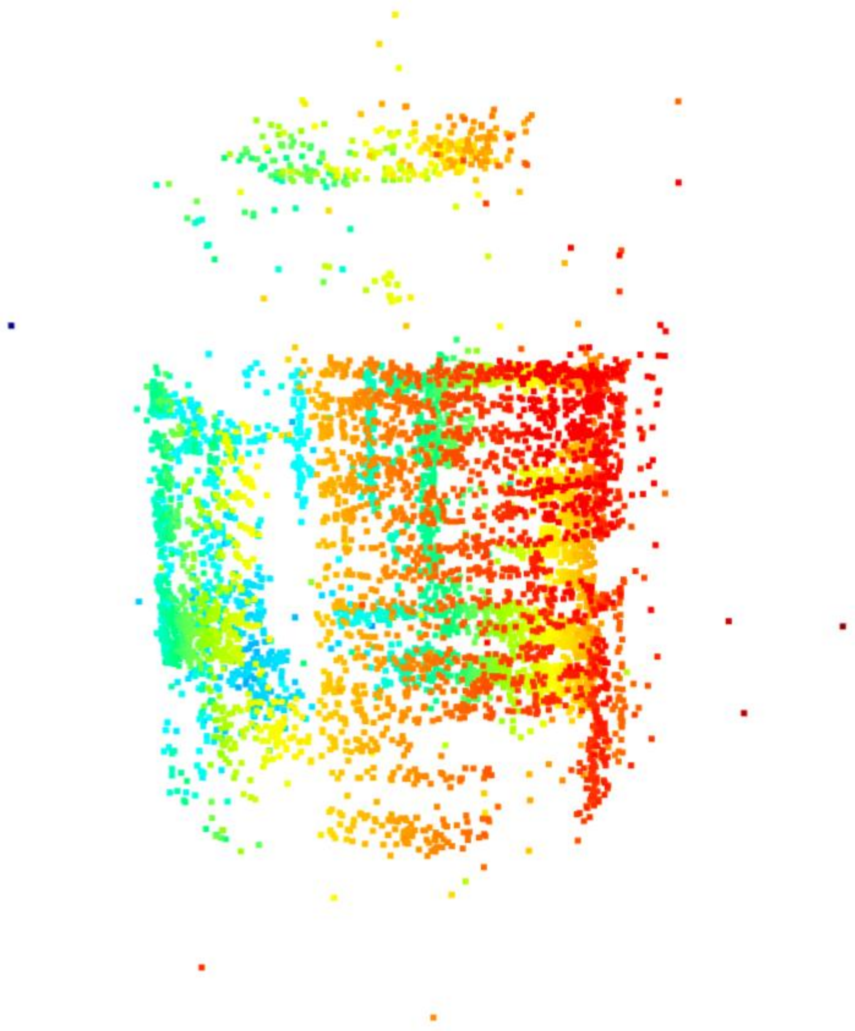
Corners (Image 2)

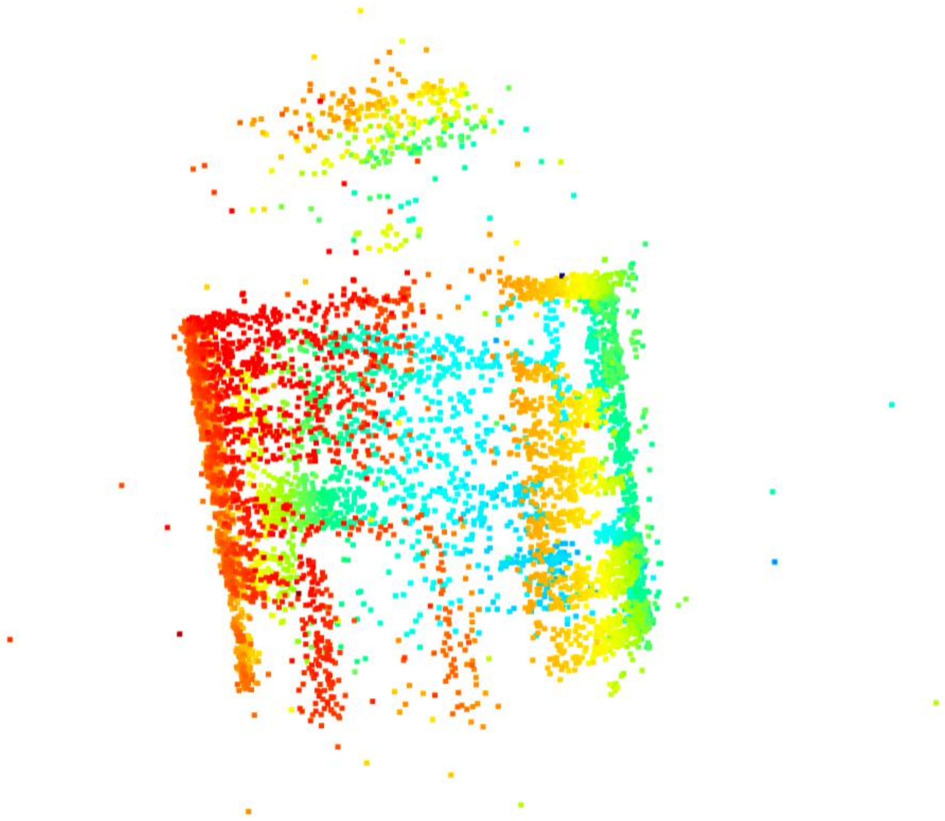


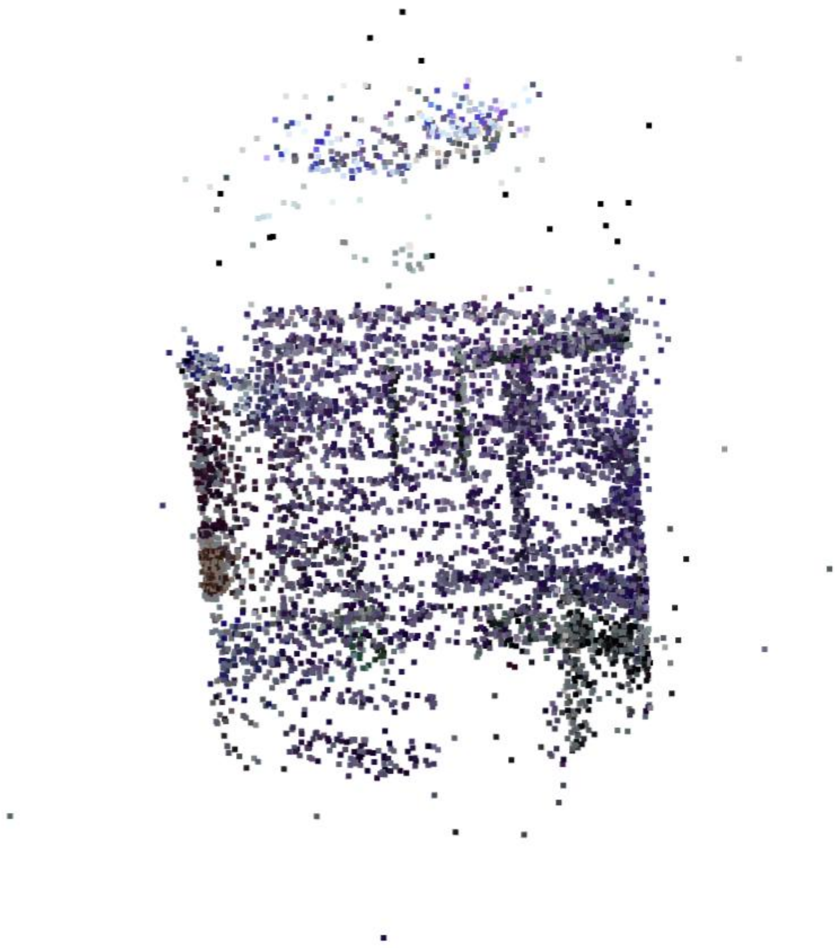
FEEL FREE
SAVEUR
MENTHE
BUBBLE
aux micro granules
SANS SUCRES




Reprojection Errors
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
5
10
15
20
25







Discussion

Mathematical Formulations

- Used SIFT with Lowe's ratio test (threshold 0.7) for robust keypoint matching.
- Computed (F), (E), and pose from matched features, with cheirality validation ensuring positive depth.
- Applied triangulation and reprojection for structure recovery.
- Optimized camera parameters, 3D points, and (K) via bundle adjustment using least squares minimization. For pre-calibrated datasets, fixing (K) is standard to maintain consistency, but this implementation optimizes (K) to potentially refine noisy calibrations in custom datasets.
- Colorization uses existing 2D points from triangulation to sample BGR values, simplifying the process by avoiding re-projection.

Calibration Analysis

- For custom datasets, camera intrinsics (K) are estimated via checkerboard calibration using `cv2.findChessboardCorners` and `cv2.calibrateCamera`, with 10 images recommended for robust results.
- Intrinsics are scaled by a factor of 2.0 to match image downscaling.

Reprojection & Reconstruction Quality

- Bundle adjustment significantly reduces reprojection errors (e.g., from 0.36 to ~0.0006 for pre-calibrated datasets), but some intermediate errors exceed 1.0 due to noisy matches or suboptimal baseline selection.
- Downscaling images improves processing speed but may reduce feature detection accuracy, impacting point cloud density.
- Point clouds capture scene structure well, though sparsity limits detail. Camera trajectories are smooth but exhibit minor drift without global loop closure, visible in trajectory visualizations.
- **Quantitative Analysis**: Bundle adjustment computation time increases with the number of points and cameras . Match quality varies, with ~50–70% of SIFT matches retained after RANSAC, depending on scene texture and baseline.

Challenges

- Bundle adjustment is computationally expensive, requiring significant memory and time .
- Image matching quality varies due to scene texture and lighting, with ~50–70% of matches retained after filtering.
- Some reprojection errors exceed 1.0, particularly for images with poor feature matches or small baselines.
- Camera trajectories deviate slightly due to noisy matches, with drift accumulating over many views (quantifiable by trajectory misalignment in Open3D visualizations).
- Downscaling (factor 2.0) balances speed and accuracy but reduces feature resolution. Higher resolution increases computation time significantly (e.g., 2–3x slower without downscaling).
- Didn't find any easy to use package to implement ceres or g2o to enhance bundle adjustment
- Poor lightening affected the accuracy of the sfm process
- Point clouds are sparse but effectively represent the scene, with colorization enhancing interpretability despite using single-view sampling and potential BGR color display, which may cause color channel mismatches (e.g., red and blue swapped).
- Camera trajectories are visualized clearly, showing the camera path, but minor deviations occur due to accumulated errors, measurable by comparing to ground truth (if available).

Deliverables

Code Implementation

- **Feature Matching**: Detect SIFT features and filter outliers using Lowe’s ratio test and RANSAC.
- **Initial Reconstruction**: Compute essential matrix, recover (R) and (t), triangulate points with cheirality validation.
- **Incremental SfM**: Register at least 5 new cameras via PnP and triangulate new points.
- **Colorization**: Assign BGR values to 3D points using single-view sampling from triangulation points.
- **Camera Calibration**: Perform calibration for custom datasets using checkerboard images.

Visualization

- Generate 3D point clouds in PLY format (`no_colors.ply`, `colors.ply`).
- Visualize camera poses and trajectories using Open3D.

- Display reprojection error plots using Matplotlib.
- Visualize feature matchings.
- Visualize corner detections for camera calibration.

Files

- 📂 `main.py`, `controller.py`,`plot.py`: Full implementation.
- 📂 `Results/`: Contains `.ply` point cloud files.
- 📂 `Results/`: Contains `OurData/` and `PrecalibratedData/` directories that contains visualizations.
- 📂 `Dataset/`: Contains `Calibrated/` and `Images/` directories that contains datasets and checkerboard images.

Resources

- Liu, S., Gao, Y., Zhang, T., Pautrat, R., Schönberger, J. L., Larsson, V., & Pollefeys, M. (2024). Robust Incremental Structure-from-Motion with Hybrid Features. arXiv preprint arXiv:2409.16719. Submitted on September 29, 2024. https://arxiv.org/abs/2409.19811
- Stack overflow
- AI language models
- Computer Vision course
- Multiple View Geometry in Computer Vision (Hartley & Zisserman).
- Dataset: https://github.com/openMVG/SfM_quality_evaluation

Installation

Install Python:
Go to the official website: https://www.python.org/downloads/
Make sure Python is added to environment variables.

Then install the required libraries:

```
pip install --upgrade pip
pip install numpy opencv-python opencv-contrib-python open3d matplotlib
tqdm scipy
```

How it Works

1. **Bundle Adjustment**: Choose whether to use it or not
2. **Dataset**: Select either a pre-calibrated dataset or calibrate your own
3. **Show Corners Detection**: For calibration, this option displays checkerboard corners on 5 images
4. **Show Matches**: Visualizes feature matches between image pairs
5. **Start**: It start the sfm technique

Acknowledgements

This project was developed by Mariam JANBEIN and Adel KANSO as part of the Computer Vision semester under the guidance of Profs:

David Fofi

Yohan Fougerolle

Zaar Khizar