NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

College of Computing and Data Science

Master of Science in Artificial Intelligence

AI6121 Computer Vision

Project Report: Structure From Motion

Tan Jia Wei (Matric No.: G2403382B)

Tiffany Tan Ge Ru (Matric No.: G2404908D)

Fung Kwok Pong (Matric No.: G2405595A)

Zhou Chunxi (Matric No.: G2405212J)

Jin Zitong (Matric No.: G2403513F)

This document consists of **10** pages, including the cover page.

Contents

1 Introduction

Structure-from-Motion (SfM) is a widely used computer vision technique that reconstructs 3D structures from 2D images. This project aims to apply SfM principles to create detailed 3D reconstructions of selected scenes from the NTU campus, leveraging the power of modern tools and algorithms. Among the available options, we chose COLMAP, a state-of-the-art photogrammetry software known for its high-resolution image performance and robust 3D reconstruction capabilities. By utilizing PyCOLMAP, a Python binding for COLMAP, we streamlined the reconstruction process to handle camera calibration, feature matching, and 3D point cloud generation efficiently.

This report details our process, including the rationale for tool selection, data collection methods, implementation steps, and evaluation of the results. We tested the pipeline on three distinct scenes within NTU to analyze its effectiveness and limitations, offering insights into potential improvements and applications of the method in diverse domains.

2 Motivation to Use COLMAP

We considered several tools for this task, including OpenCV, COLMAP and Meshroom. Ultimately, we decided to use COLMAP based on the following considerations.

OpenCV, while convenient, requires camera calibration, which can be challenging to perform accurately given the limitations of our equipment, as set out in the instructions. The calibration process is also time-consuming. Additionally, OpenCV struggles with solving certain ill-posed problems such as mismatched features between images.

We attempted to convert the OpenCV scene reconstruction code from Python 2 to Python 3. However, we encountered compatibility issues during the process and the output did not meet our expectations. We also explored the Structure from Motion (SfM) module in OpenCV, which is implemented in C++. Unfortunately, the module lacked adequate Python bindings and we were unable to successfully convert the relevant code in Python. Additionally, we experimented with OpenMVG but its output did not meet our desired quality standards. This was perhaps due to its simpler feature matching and outlier rejection algorithms, which can struggle with complex datasets involving significant variations in lighting, scale, or viewpoint. Additionally, its optimization processes are less advanced, potentially leading to incomplete or inaccurate reconstructions compared to more robust tools like COLMAP.

On the other hand, COLMAP offers a significant advantage by enabling camera calibration through automatic feature detection and matching between images, simplifying the workflow and ensuring more reliable results.

Compared to Meshroom, COLMAP excels in handling high-resolution images and generating detailed 3D reconstructions. It is also better optimized for CUDA, which is crucial for our setup. While Meshroom may be more suitable for large-scale projects, it faces challenges in recognizing smooth or transparent objects and preserving intricate details. Since our project is on a smaller scale, COLMAP's capabilities align more closely with our requirements and therefore is the most suitable pipeline.

3 How PyCOLMAP Works and Reconstruction Pipeline

3.1 About PyCOLMAP

In this project, we leveraged PyCOLMAP - which is a Python binding for COLMAP. Broadly, it works by identifying overlapping images to identify common points, estimating the positions of the cameras when the images were taken, and building a 3D representation of the scene.

Its pipeline comprises of feature extraction, feature matching, camera pose estimation, and incremental 3D reconstruction and will be explained further in section 3.2.

Inputs required are: 1) 2D Image dataset, taken from multiple perspectives with significant overlap between adjacent views. 2) Camera intrinsic parameters like focal length, principal point (optical center and distortion coefficients

The output of PyCOLMAP contains the 3D reconstruction of the scene as a point cloud (in a .ply file). This file stores the spatial coordinates of the reconstructed points, along with optional information like color or intensity. To visualize the reconstruction, we used MeshLab for 3D view of the point clouds.

3.2 How COLMAP works

The following describes the operation of COLMAP:

1. **Feature Extraction**: Detects keypoints using SIFT, with parameters like `max_num_features` controlling the number of features and `num_threads` enabling parallel processing.
2. **Feature Matching**: Matches descriptors between images using nearest-neighbor search, with geometric verification through RANSAC (`ransac_max_error`) to filter outliers.
3. **Camera Pose Estimation**: Computes camera positions and orientations using the essential or fundamental matrix, ensuring robust poses with parameters like `min_num_inliers`.
4. **Sparse Reconstruction**: Triangulates 3D points from matched keypoints to form a sparse point cloud, capturing the basic structure of the scene.
5. **Bundle Adjustment**: Optimizes 3D points, camera poses, and intrinsic parameters by minimizing reprojection error, controlled by settings like `max_iterations`.
6. **Dense Reconstruction**: Computes depth maps and fuses them into a dense point cloud or mesh, with parameters like `stereo_window_size` and `stereo_max_depth`.

3.3 Code Implementation

In this section, we will demonstrate how the code was implemented to do a 3D reconstruction of a scene in NTU. We ensured to take images from multiple angles and heights, on a clear day to ensure consistent lighting. We chose to reconstruct three scenes in NTU. The first scene is the CCDS signpost, located in front of the CCDS office. The second scene is the Former Nanyang University Memorial in Yunnan Garden, which commemorates the completion of the first phase of the university's construction. The third one is *tète-à-tète*, a bespoke piece of artwork located in the School of Humanities, which is a simple object but effective in reconstruction practice. Their appearances are shown in Section 4. We chose them because they serve as prominent and symbolic features of the campus, making them meaningful subjects for our project. Their sizes and complexity provide an opportunity for us to try capture intricate details from multiple angles.

The Python code to complete the 3D point cloud reconstruction is shown below, whose inputs are a set of images taken from the chosen scenes.

```python
import pathlib
import pycolmap

IMAGE_PATH = pathlib.Path.cwd() / "image"
OUTPUT_PATH = pathlib.Path.cwd() / "output"

OUTPUT_PATH.mkdir()
mvs_path = OUTPUT_PATH / "mvs"
database_path = OUTPUT_PATH / "database.db"

pycolmap.extract_features(database_path=database_path, image_path=IMAGE_PATH,
↪   camera_model='OPENCV')
pycolmap.match_exhaustive(database_path=database_path)
maps = pycolmap.incremental_mapping(database_path=database_path, image_path=IMAGE_PATH,
↪   output_path=OUTPUT_PATH)
maps[0].write(OUTPUT_PATH)
maps[0].write_text(OUTPUT_PATH)
maps[0].export_PLY(OUTPUT_PATH / "sparse.ply")

pycolmap.undistort_images(output_path=mvs_path, input_path=OUTPUT_PATH, image_path=IMAGE_PATH)
pycolmap.patch_match_stereo(workspace_path=mvs_path)  # requires compilation with CUDA

pycolmap.stereo_fusion(output_path=mvs_path / "dense.ply", workspace_path=mvs_path)
pycolmap.poisson_meshing(input_path=mvs_path / "dense.ply", output_path=mvs_path /
↪   "meshed-poisson.ply")
```

To install PyCOLMAP on Windows with CUDA support, as the instruction for readers, we created an Anaconda environment with Python 3.12 and install `pycolmap` via

```
conda create -n COLMAP python=3.12
conda activate COLMAP
conda install pycolmap=3.10=cuda120py312hd04e3e1_2
```

The execution of the code follows the pipeline below.

3.4 Pipeline of Structure-from-Motion

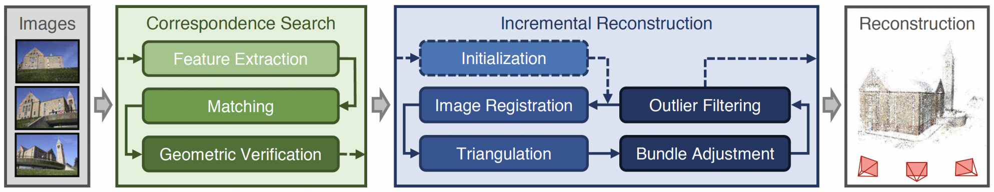

Figure 1: COLMAP's incremental Structure-from-Motion pipeline (Schönberger and Frahm, 2016).

Structure-from-Motion (SfM) is the process of reconstructing 3D structure from its projections into a series of images (Schönberger and Frahm, 2016). The input is a set of overlapping images $\mathcal{I} = \{I_i | i = 1, \ldots, |\mathcal{I}|\}$ of the same object, taken from different viewpoints. The output is a 3-D reconstruction of the object, and the reconstructed intrinsic and extrinsic camera parameters of all images. Typically, Structure-from-Motion systems divide this process into three stages, which are discussed respectively below.

3.4.1 Feature Detection and Extraction

In the first step, COLMAP finds sparse feature points in the input images and describes their appearances using a numerical descriptor. For each image I_i, SfM detects sets $\mathcal{F}_i = \{(\boldsymbol{x}_j, \boldsymbol{f}_j) | j = 1, \ldots, |\mathcal{F}_i|\}$ of local features at location $\boldsymbol{x}_j \in \mathbb{R}^2$ represented by a SIFT (Lowe, 2004) descriptor $\boldsymbol{f}_j$. The features should be invariant under radiometric and geometric changes so that SfM can uniquely recognize them in multiple images (Slama et al., 1980). All the images in our project are shot on mobile phones, which are inexpensive but substantial devices compared to professional cameras exclusive for reconstruction. To obtain intrinsic parameters of camera, the focal length is automatically extracted from the embedded EXIF information. Since our images were captured by the same physical camera with identical zoom factor, the intrinsics share between all the images. The detected features are then extracted to files on disk drive.

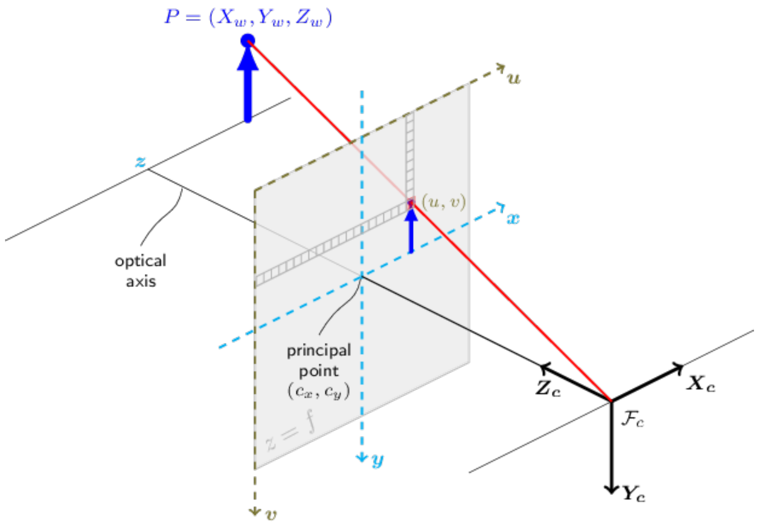

Figure 2: Pinhole camera model. (Bradski, 2000).

During this process, the estimators of COLMAP discovers the best camera intrinsic parameters to minimize mismatch. Real lenses usually have some distortion, mostly radial distortion, and slight tangential distortion.

Table 1: Estimated camera intrinsic parameters at the three scenes in NTU.

	"CCDS Signpost"	"Former Nanyang University Memorial"	"*tète-à-tète*"
Image Width	4032	4000	4000
Image Height	3024	3000	3000
f_x	1605.8358805349317	2716.9311253749129	1564.6930778694507
f_y	1606.9931627964017	2784.0175109990728	1575.950263423983
c_x	2016	2000	2000
c_y	1512	1500	1500
k_1	−0.0031229605821424403	0.088110280337931432	0.0016967444510417535
k_2	0.00047945720106026813	−0.10932857438621135	−0.00050255445787063366
p_1	−0.00018596153928465524	0.00061034732758530634	0.00042817254921631044
p_2	−0.00016600507865841953	−0.0013172975510685936	−0.0010509205590455621

The pinhole camera model is shown in Figure 2 (Bradski, 2000), where the model is extended as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x x'' + c_x \\ f_y y'' + c_y \end{bmatrix}, \tag{1}$$

where

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4) \begin{bmatrix} x' \\ y' \end{bmatrix} + p_1 \begin{bmatrix} 2x'y' \\ r^2 + 2y'^2 \end{bmatrix} + p_2 \begin{bmatrix} r^2 + 2x'^2 \\ 2x'y' \end{bmatrix} \tag{2}$$

with $r = x'^2 + y'^2$ and $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix}$. In our experiments, the parameters are estimated as in Table 1.

3.4.2 Feature Matching and Geometric Verification

In the second step, COLMAP finds correspondences between the feature points $\mathcal{F}_i$ as an appearance description of the images. We use Exhaustive Matching, the naïve approach that tests every image pair for scene overlap; it searches for feature correspondences by finding the most similar feature in image I_a for every feature in image I_b, using a similarity metric comparing the appearance $\boldsymbol{f}_j$ of the features. This approach has computational complexity $O(|\mathcal{I}|^2|\mathcal{F}_i|^2)$ (Schönberger and Frahm, 2016) and is prohibitive for large image collections. However, because the number of images in our dataset is relatively low (less than 100), and this method leads to the best reconstruction results, we still stick to this method. Here, every image is matched against every other image, while the block size is 50 which determines how many images are loaded from disk into memory at the same time. The outputs are a set of potentially overlapping image pairs $\mathcal{C} = \{\{I_a, I_b\}|I_a, I_b \in \mathcal{I}, a < b\}$ and their associated feature correspondences $\mathcal{M}_{ab} \in \mathcal{F}_a \times \mathcal{F}_b$.

In the third step, SfM verifies the potentially overlapping image pairs $\mathcal{C}$ by trying to estimate a transformation that maps feature points between images using epipolar geometry (Schönberger and Frahm, 2016). If a valid transformation maps a sufficient number of features between the images, they are considered geometrically verified The outputs of this stage are a set of geometrically verified image pairs $\bar{\mathcal{C}}$ and their associated inlier correspondences $\bar{\mathcal{M}}_{ab}$.

3.4.3 Sparse Reconstruction

After producing the scene graph in the previous three steps, the incremental reconstruction process seeds the reconstruction from an initial image pair. Then, the scene is incrementally extended by registering new images and triangulating new points. This incremental mapping process does non-linear refinement of structure and motion and minimizes reprojection error (Triggs et al., 2000)

$$\min_{\boldsymbol{P},\boldsymbol{X}} ||\boldsymbol{x} - \boldsymbol{\pi}(\boldsymbol{P}, \boldsymbol{X})||, \tag{3}$$

where $\boldsymbol{x}$ are points of the target plane, $\boldsymbol{\pi}$ are object surfaces or planes, $\boldsymbol{P}$ are intrinsic calibration parameters and $\boldsymbol{X}$ are points in the original plane.

The outputs are pose estimates $\mathcal{P} = \{\boldsymbol{P}_c \in \boldsymbol{SE}(3)|c = 1, \ldots, |\mathcal{P}|\}$ for registered images and the reconstructed scene structure as a set of points $\mathcal{X} = \{\boldsymbol{X}_k \in \mathbb{R}^3|k = 1, \ldots, |\mathcal{X}|\}$. These points $\mathcal{X}$ are called a point cloud.

3.4.4 Dense Reconstruction

Multi-View Stereo (MVS) takes the output of SfM to compute depth and normal information for every pixel in an image. Fusion of the depth and normal maps of multiple images in 3D then produces a dense point cloud of the scene. After reconstructing a sparse representation of the scene and the camera poses of the input images, MVS can now recover denser scene geometry (Schönberger et al., 2016). COLMAP has an integrated dense reconstruction pipeline to produce depth and normal maps for all registered images, to fuse the depth and normal maps into a dense point cloud with normal information.

4 Evaluation and Future Work

Using images from the previously described structures, we ran our implementation and the results are generally good.

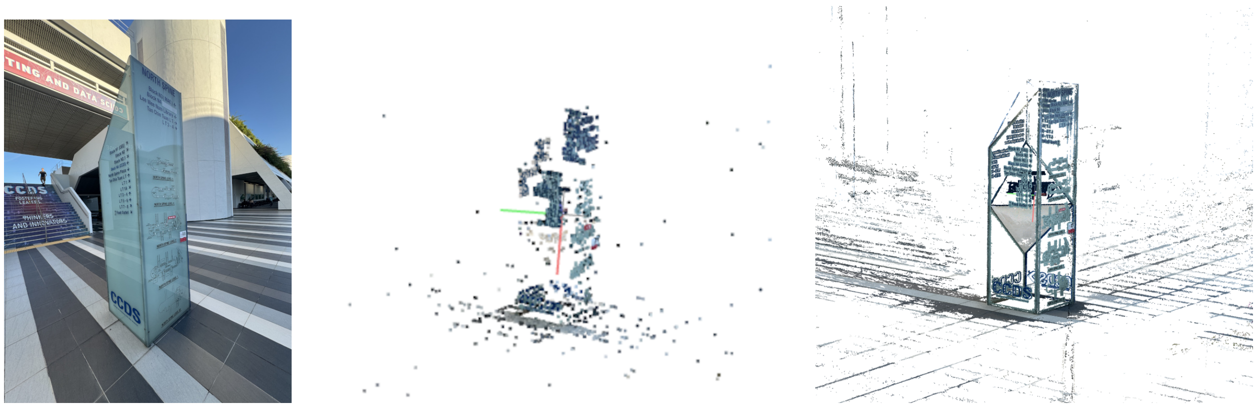

Figure 3: Comparison between actual image, sparse and dense reconstruction on the CCDS Signpost

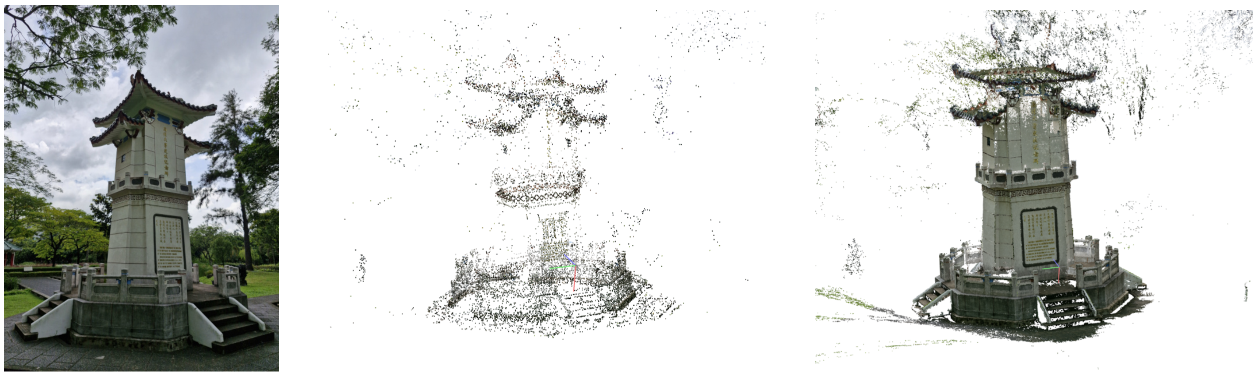

Figure 4: Comparison between actual image, sparse and dense reconstruction on the Former Nanyang University Memorial

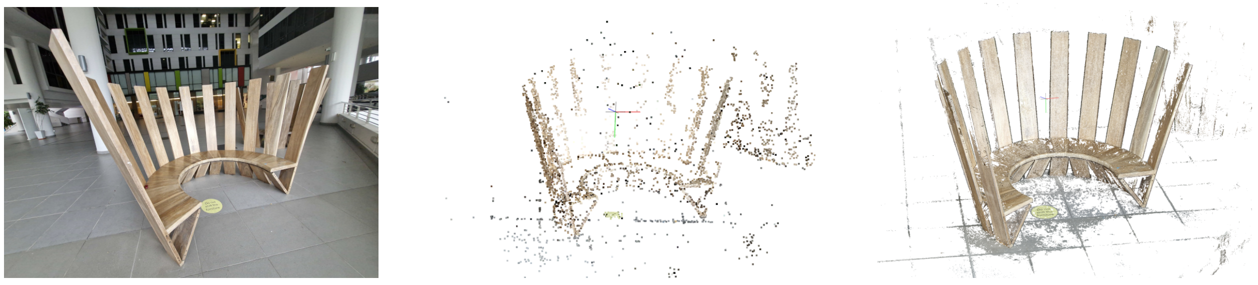

Figure 5: Comparison between actual image, sparse and dense reconstruction on *tète-à-tète*

For each structure, we executed two runs. The first run used sparse reconstruction, while the second run used dense reconstruction. Sparse reconstruction typically takes 20 to 50 minutes to create a point cloud, depending on the total number of input images. However, the output 3D point cloud shows only a hollow structure of the underlying objects. In comparison, dense reconstruction takes 20 to 40 minutes to construct, and it shows a nicer solid shape with better details. For the reconstruction, we have used a powerful NVIDIA GeForce RTX 4060 GPU.

It is worth noting that for the Former Nanyang University Memorial in Figure 4, the top of the structure could not be properly reconstructed because it is too tall, and we were unable to capture its top view. Nevertheless, the details on its body can be clearly seen in the dense point cloud.

Besides the main structures, parts of the background are loosely constructed. Such reconstruction is particular useful in applications like autonomous driving. As a vehicle camera footage is sampled at intervals, existing objects, and newly discovered ones (which initially forms as part of the background), and their distances can be computed, and then feed into an autonomous driving algorithm. However, SfM on COLMAP is time consuming and resource intensive. Therefore, further explorations could point to newer algorithms such as Hierarchical Localization (Paul-Edouard et al., 2019) and CF-3DGS (Yang et al., 2024), which may achieve quicker computations for real-time purposes.

5 Conclusion

In this project, we successfully implemented a Structure-from-Motion pipeline using PyCOLMAP to reconstruct 3D models of selected scenes from the NTU campus. Our results demonstrated the capabilities of COLMAP in generating sparse and dense reconstructions with considerable detail. Despite challenges, such as incomplete reconstruction of tall structures due to limited image perspectives, the dense reconstruction pipeline provided a robust representation of the scenes.

The project highlighted the trade-offs between sparse and dense reconstruction methods, with sparse reconstructions being computationally efficient and dense reconstructions providing richer details. However, the time and resources required for dense reconstruction underscore the need for exploring more efficient algorithms in future work. Emerging approaches, like Hierarchical Localization and CF-3DGS, offer promising avenues for real-time applications.

Overall, this project demonstrates the feasibility of SfM for practical use cases, such as autonomous driving and virtual environment creation, while underscoring the potential for optimization in scalability and computational efficiency.

References

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools.*

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

Paul-Edouard, S., Cesar, C., Roland, S., & Marcin, D. (2019). From coarse to fine: Robust hierarchical localization at large scale. *arXiv:1812.03506*.

Schönberger, J. L., & Frahm, J.-M. (2016). Structure-from-motion revisited. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4104–4113. https://doi.org/10.1109/CVPR.2016.445

Schönberger, J. L., Zheng, E., Frahm, J.-M., & Pollefeys, M. (2016). Pixelwise view selection for unstructured multi-view stereo. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision – eccv 2016* (pp. 501–518). Springer International Publishing.

Slama, C. C., Theurer, C., & Henriksen, S. W. (1980). *Manual of photogrammetry.*

Triggs, B., McLauchlan, P. F., Hartley, R. I., & Fitzgibbon, A. W. (2000). Bundle adjustment — a modern synthesis. In B. Triggs, A. Zisserman, & R. Szeliski (Eds.), *Vision algorithms: Theory and practice* (pp. 298–372). Springer Berlin Heidelberg.

Yang, F., Sifei, L., Amey, K., Jan, K., Alexei A., E., & Xiaolong, W. (2024). Colmap-free 3d gaussian splatting. *arXiv:2312.07504*.