Incremental Structure from Motion (SfM) Project

3D Scene Reconstruction from a Dataset of 2D Images

Ghaith Chamaa & Rafay Aamir
Under Supervision of Prof. David Fofi & PhD. Candidate Zaar Ali
Université Bourgogne Europe

May 23, 2025

Overview

Introduction

- Structure from Motion (SfM) is a photogrammetric technique used to reconstruct 3D scenes from a set of 2D images.
- The incremental SfM approach builds the 3D model progressively by initializing reconstruction with a pair of images and incrementally adding more images and 3D points.
- This project uses a dataset of **N images** (configurable).
- Independent of camera calibration matrix.
- Performs feature extraction, matching, triangulation, and bundle adjustment for accurate 3D reconstruction.
- Final reconstruction has standard SFM visualization (Camera Poses, Trajectory, Point Cloud and Colored Pixels).

Theory: Structure from Motion Overview

SfM estimates 3D structure and camera motion from a sequence of 2D images by exploiting the geometric relationships between images. The pipeline involves:

- **Feature Extraction and Matching**: Identifying and matching keypoints across images to establish correspondences.
- **Epipolar Geometry**: Using the fundamental matrix to filter outliers and build connectivity between images.
- **Triangulation**: Computing 3D points from 2D correspondences.
- **Perspective-n-Point (PnP)**: Estimating camera poses for new images.
- **Bundle Adjustment (BA)**: Optimizing camera parameters and 3D points to minimize reprojection errors.

Theory: Key Concepts

1. **Lowe's Ratio Test**: For keypoint matching, the ratio of distances between the best and second-best matches is computed. A match is reliable if the ratio is below a threshold (e.g., 0.75).
2. **Fundamental Matrix (F)**: A 3x3 matrix (rank 2) that enforces epipolar constraints to filter outlier matches. It relates corresponding points in two images.
3. **Epipolar Graph**: An adjacency matrix representing image pairs with sufficient good matches, where edges are weighted by the number of matches.
4. **Triangulation**: Computes 3D points from 2D correspondences using camera poses and the calibration matrix.
5. **PnP**: Estimates the extrinsic camera parameters (rotation and translation) of a new image given 2D-3D correspondences and the calibration matrix.
6. **Bundle Adjustment (BA)**: A non-linear least squares optimization that minimizes reprojection errors by refining camera poses and 3D point coordinates.

RANSAC: Robust Model Fitting in a Nutshell

Goal: Estimate model parameters from data containing significant outliers.

Algorithm Overview

Repeat ‘T‘ times:

1. **Sample**: Randomly pick minimal data points (‘s‘) to fit model.
2. **Compute**: Fit a candidate model to these ‘s‘ points.
3. **Score**: Count data points (inliers) fitting this model within error ‘δ‘.

Best model is the one with the most inliers after ‘T‘ trials.

Key Parameters:

- ‘s‘: Min. points for model (e.g., 2 for line, 8 for Fundamental Matrix, 3+1 for PnP).
- ‘δ‘: Inlier error/distance threshold.
- ‘T‘: Iterations, chosen for high success probability ‘p‘, given outlier ratio ‘e‘: $T = \frac{\log(1-p)}{\log(1-(1-e)^s)}$.

SfM Uses: Essential for robust Fundamental/Essential Matrix estimation, PnP.

Theory: Mathematical Formulation (Bundle Adjustment)

Bundle Adjustment Objective

Minimize reprojection error:

$$E(\{R_i, T_i\}_{i=1}^{m}, \{X_j\}_{j=1}^{N}) = \sum_{i=1}^{m} \sum_{j=1}^{N} \theta_{ij} \left\| \tilde{x}_{ji} - \pi(R_i, T_i, X_j) \right\|^2$$

Where:

- (R_i, T_i): Camera i rotation and translation.
- X_j: 3D point j.
- $\tilde{x}_{ji}$: Observed 2D keypoint.
- π: Perspective projection.
- θ_{ij}: Visibility indicator.

The Jacobian matrix for BA's non-linear LS is sparse, with non-zero blocks corresponding to camera parameters (A_{ijk}) and 3D points (B_{ijk}).

Methodology: Overview

The SfM pipeline is divided into three main stages:

1. **Matching**
2. **Reconstruction**
3. **Bundle Adjustment**

Methodology: 1. Dataset and Calibration

- **Dataset**: Processes a dataset of **N images** (configurable).
- **Calibration Matrix (K)**: Camera intrinsic parameters are assumed known:

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

Methodology: 2. Matching

The matching script establishes correspondences between image pairs:

1. **Feature Extraction**: Extract keypoints and descriptors (e.g., SIFT) from all images.
2. **All-to-All Matching**: Perform k-nearest neighbor (k-NN) matching between descriptors of all image pairs. For each pair (i, j), compute a list of good matches based on Lowe's ratio test.
3. **Outlier Filtering**: Use the fundamental matrix F (rank 2) to filter outliers based on epipolar constraints.
4. **Epipolar Graph Construction**: Build an adjacency matrix (upper triangular) where edges are weighted by the number of good matches [Fusiello, , P131].

Output: A 3D matrix ($n \times n \times m$) where n is number of images, m is number of good matches for a pair.

Methodology: 3. Reconstruction (1/2)

Performs incremental 3D reconstruction [Fusiello, , P144]:

1. **Initial Pair Selection**: Based on:
   - Sufficient Translation (ensures triangulation stability [Geiger, , P33]).
   - High Number of Matches (normalized to max as percentages to avoid hardcoding thresholds, resuling in flexible Initialization by percentage-based pair selection.
2. **Initial Triangulation**: Use selected pair to compute initial 3D points.
3. **Incremental Reconstruction**:
   - For each unresected image, estimate camera pose via PnP [Stachniss,] using resected images. (PnP needs ≥ 3, 2D-3D correspondences).
   - Add new 3D points via triangulation.

Output: Points from PnP and identifies points for new triangulation between resected/unresected pair.

Methodology: 3. Reconstruction (2/3)

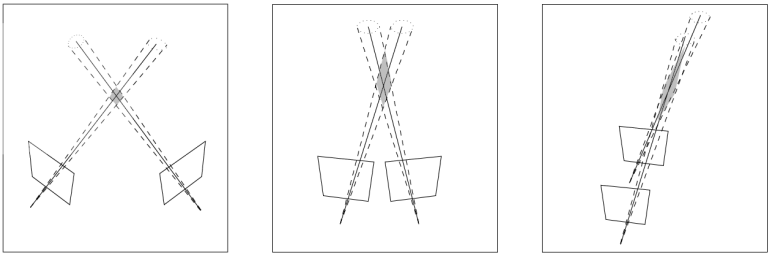


Figure: Triangulation stability: wider baseline (B) and precise disparity improve depth (z) accuracy.

$$\sigma_z \approx \frac{z^2 \sigma_p}{fB}$$

Where:

- σ_z: error in depth estimation
- z: actual estimated depth of the point
- σ_p: error in disparity (pixels)
- f: focal length (pixels)
- B: baseline (world units)

Methodology: 3. Reconstruction (3/3)

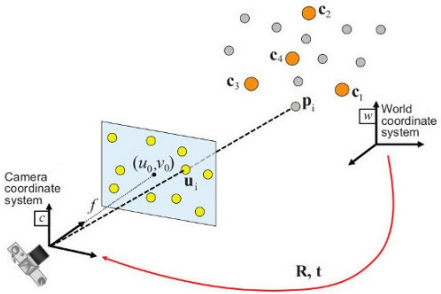


Figure: PnP, estimate exterior pose of camera

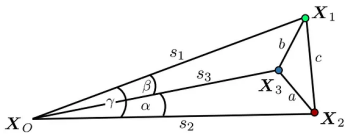


Figure: PnP, estimate angles and distance between 3D points (objective is s1, s2, s3)

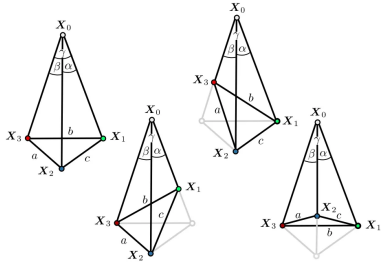


Figure: Ambiguity (same angles, different lengths)

Methodology: 4. Bundle Adjustment (1/2)

BA refines reconstruction by minimizing reprojection errors [Cremers, , P05], [Fusiello, , P167]:

1. **Monitoring**: Track PnP inlier percentage and reprojection error. Trigger BA if too low.
2. **Optimization**: Solve non-linear least squares problem using solvers leveraging sparse Jacobian.

The Jacobian consists of:

- A_{ijk}: Partial derivatives of residual of point j in frame i w.r.t. camera orientation k.

$$A_{ijk} = \frac{\partial \tilde{\eta}(P_i \mathbf{M}^j)}{\partial \mathbf{g}_k^\top}$$

- B_{ijk}: Partial derivatives of residual of point j in frame i w.r.t. coordinates of point k.

$$B_{ijk} = \frac{\partial \tilde{\eta}(P_i \mathbf{M}^j)}{\partial \tilde{\mathbf{M}}_k^\top}$$

Methodology: 4. Bundle Adjustment (2/2)

3. **Sparsity Exploitation**:
   - $A_{ijk} = 0$ for all $i \neq k$.
   - $B_{ijk} = 0$ for all $j \neq k$.
   - Implementation uses camera observation indices to activate only the specific camera for an observation, reducing cost.

Jacobian Matrix Structure:

$$\left(\begin{array}{ccc|ccc}
A_{111} & & & B_{111} & & \\
A_{121} & & & & B_{122} & \\
\vdots & & & & & \ddots \quad \vdots \\
A_{1n_1} & & & & & B_{1nn} \\
 & A_{212} & & B_{211} & & \\
 & A_{222} & & & B_{222} & \\
 & \vdots & & & & \ddots \quad \vdots \\
 & A_{2n_2} & & & & B_{2nn} \\
\cdots & \cdots & \ddots & \cdots & \cdots & \ddots \quad \cdots \\
 & & A_{m1m} & B_{m11} & & \\
 & & A_{m2m} & & B_{m22} & \\
 & & \vdots & & & \ddots \\
 & & A_{mn_mm} & & & B_{mnn}
\end{array}\right)$$

Optimization: Non-Linear Least Squares (NLS)

In Bundle Adjustment (BA) The goal is to find parameters $\mathbf{p}$ that minimize a sum of squared residuals which are reprojection errors:

$$\min_{\mathbf{p}} \sum_{k} \|r_k(\mathbf{p})\|^2$$

The Gauss-Newton Algorithm for solving NLS problems

1. **Initialize**: Start with an initial estimate of parameters $\mathbf{p}_t$.
2. **Linearize**: Approximate the residuals around the current estimate

$$r(\mathbf{p}_t + \delta\mathbf{p}) \approx r(\mathbf{p}_t) + J(\mathbf{p}_t)\delta\mathbf{p}$$

3. **Solve Linear System**: Find the update step $\delta\mathbf{p}$ that minimizes the linearized sum of squares $\|r(\mathbf{p}_t) + J(\mathbf{p}_t)\delta\mathbf{p}\|^2$. This leads to:

$$\delta\mathbf{p} = -\left(J(\mathbf{p}_t)^T J(\mathbf{p}_t)\right)^{-1} J(\mathbf{p}_t)^T r(\mathbf{p}_t)$$

4. **Update**: Update the parameters: $\mathbf{p}_{t+1} = \mathbf{p}_t + \delta\mathbf{p}$.
5. **Iterate**: Repeat steps 2-4 until convergence.

Optimization: Gauss-Newton for Bundle Adjustment

Applying Gauss-Newton to Bundle Adjustment:

- **Parameters p**: The camera poses $\{R_i, T_i\}$ and 3D point coordinates $\{X_j\}$.
- **Residuals** $r_k(\mathbf{p})$: The reprojection errors $r_{ji}(\cdot) = \tilde{x}_{ji} - \pi(R_i, T_i, X_j)$ for each visible point j in image i.

The Jacobian in Bundle Adjustment

- Its rows correspond to individual residual components (e.g., x and y error for a reprojection), its columns correspond to the parameters being optimized.

Key Aspects for BA:

- The term $J^T J$ is an approximation of the Hessian matrix (avoiding explicit computation of 2nd-order derivatives) of the BA cost function. It is symmetric and positive semi-definite (gives curvature information at current iter).
- The **sparsity** of J (and thus $J^T J$) is critical. Specialized solvers exploit this sparsity to efficiently solve specific (known thanks to jacobian sparsity) normal equations (error functionals set to zero), especially for large-scale BA problems.

Implementation Side:

- **scipy.optimize.least_squares** was used, often employing Trust Region Reflective (TRF) method which uses the Gauss-Newton for NLS.
- The **sparsity** of the Jacobian J is provided to the solver to efficiently handle large-scale BA problems

Optimization: Gradient Descent and Adam (1/2)

Gradient Descent (GD) methods offer an alternative iterative approach for optimization. They are **first-order methods**, using only the gradient (first derivative) $\nabla L(\mathbf{p})$ of the loss function $L(\mathbf{p})$.

$$\mathbf{p}_{t+1} = \mathbf{p}_t - \alpha \nabla L(\mathbf{p}_t)$$

where α is the learning rate.

The Adam (Adaptive Moment Estimation) Optimizer

Adam is an advanced GD variant that adapts the learning rate for each parameter individually.

- It computes adaptive learning rates using estimates of **first and second moments** of the gradients.
- It keeps an exponentially decaying average of past gradients (1st moment).
- Like RMSProp, it stores an exponentially decaying average of past squared gradients (2nd moment).
- Can be seen as a combination of RMSProp and momentum with bias correction.

Optimization: Gradient Descent and Adam (2/2)

The Adam Algorithm

(Hyperparameters: step size α, decay rates ρ_1, ρ_2, stability constant β. Parameters to optimize: θ.)

1. **Initialize**: Parameters θ_0, 1st moment $s_0 = 0$, 2nd moment $r_0 = 0$, time step $t = 0$.
2. **Iterate** for $t = 1, 2, \ldots$ until convergence:
   - 2.1 Compute gradient: $\hat{g}_t = \nabla_\theta L(\theta_{t-1})$ (on current minibatch).
   - 2.2 Update biased 1st moment: $s_t \leftarrow \rho_1 s_{t-1} + (1 - \rho_1)\hat{g}_t$.
   - 2.3 Update biased 2nd moment: $r_t \leftarrow \rho_2 r_{t-1} + (1 - \rho_2)\hat{g}_t \odot \hat{g}_t$.
   - 2.4 Correct bias in 1st moment: $\hat{s}_t \leftarrow s_t / (1 - \rho_1^t)$.
   - 2.5 Correct bias in 2nd moment: $\hat{r}_t \leftarrow r_t / (1 - \rho_2^t)$.
   - 2.6 Update parameters: $\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{s}_t}{\sqrt{\hat{r}_t} + \beta}$.

(Note: β here is the stability constant, $\odot$ denotes element-wise product.)

Optimization: Gradient Descent with Adam & PyTorch

Similar to Gauss-Newton based methods, **Gradient Descent (GD)** offers another iterative approach:

- **First-Order Method**: Relies only on the gradient of the loss function, not requiring explicit computation or approximation of the Hessian (like $J^T J$), Hessian nor matrix inversion.
- **Adaptive Step Size**: Adam adjusts learning rates per parameter, which can be beneficial for complex loss landscapes like BA.

PyTorch for BA with Adam:

- **Automatic Differentiation**: PyTorch's 'autograd' engine automatically computes the necessary gradients $\nabla L(\mathbf{p})$ for the BA loss.
- **Project Implementation**: We are using PyTorch with the Adam optimizer to directly minimize the BA reprojection error,
  which allowed a 4X decrease in computational cost (from 8min to 2min in the case of Middlebury).

Adam v.s. Gauss Newton:

- **Adam**: 1st/2nd moments control update speed, low grad $\rightarrow$ more speed, high grad $\rightarrow$ less speed.
- **Gauss Newton**: takes Hessian for curvature info to control update speed, **Levenberg** has λ that switches between GN and GD, **Marquadt** has adaptivity $\lambda.diag(J^T J)$.

Methodology: Point Cloud Colorization

To enhance the realism of the 3D reconstruction, each point in the cloud is assigned an RGB color. This is done during the export process to COLMAP format (to visualize camera trajectory, color, camera poses and 3D-Camera rays), also visualizing color and camera poses in **open3D**.

Color Source

- For each 3D point, we identify all 2D keypoints in the original images that correspond to it.
- The color is sampled from the original image at the location of these 2D keypoints.

Color Assignment Strategy A 3D point may be visible in multiple images, leading to multiple color candidates. The project implements the mean approach to determine the final color for each 3D point:

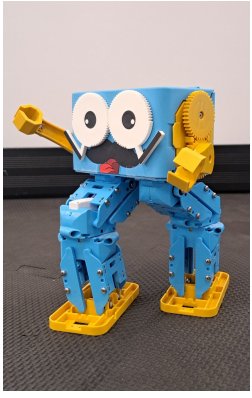
1. Calculates the average RGB values from all valid 2D observations.
2. Can produce a smoother, more blended appearance.
3. BGR pixel values are collected, averaged per channel (R, G, B), then rounded.

Results (1/10)

The pipeline produces:

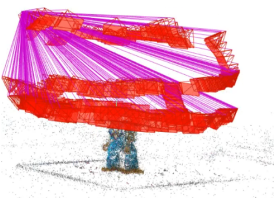
- **3D Point Cloud**: A sparse set of 3D points representing the scene.
- **Camera Poses**: Extrinsic parameters (rotation and translation) for each image.
- **Reprojection Error**: Typically reduced to below 1 pixel after bundle adjustment.
- **Performance**: Successfully reconstructs scenes with 46 images in reasonable time (hardware/dataset dependent).

Results (2/10)



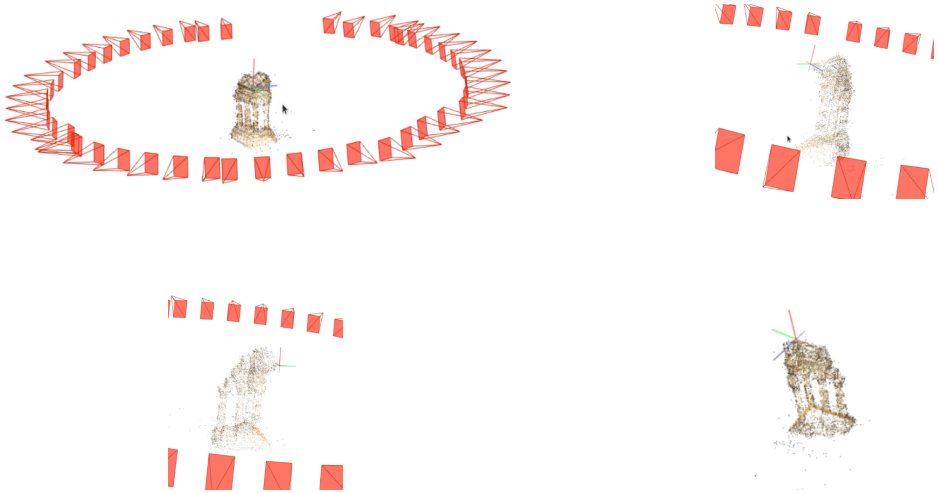
The used object in custom data **open3D**.

Results (3/10)



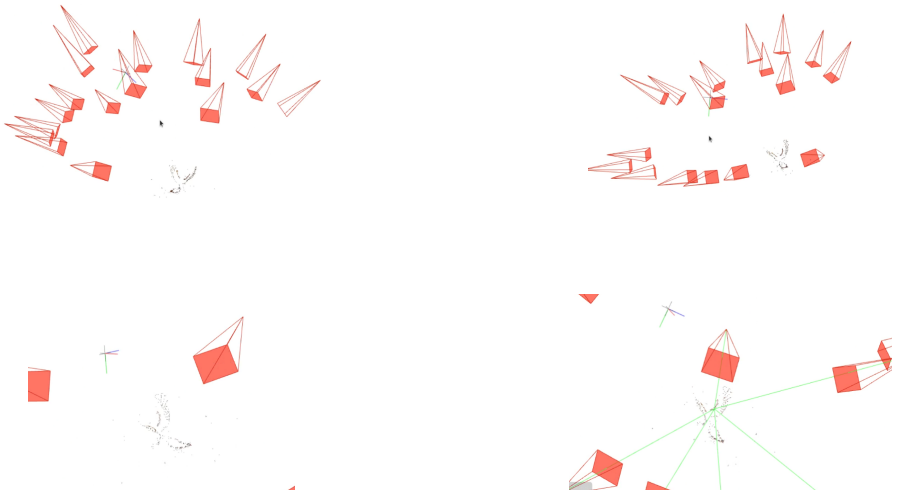
Some views of the object in the collected custom dataset (345 images).

Results (4/10)



Some views of the ruin in the Middlebury dataset (46 images).

Results (5/10)



Some views of an object captured with few images (21 images) to see the performance of the algorithm.

Results (6/10)

A view of the object in the Middlebury dataset visualized with **open3D**.

Results (7/10)

(a) t = 1

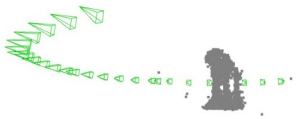
(b) t = 2

(c) t = 3

(d) t = 4

Some Views of the Incremental Reconstruction (1/2).

Results (8/10)



(a) t = 5

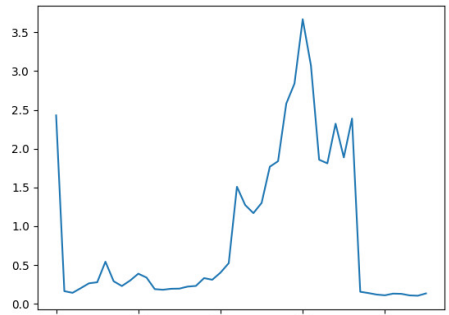
(b) t = 6

(c) t = 7

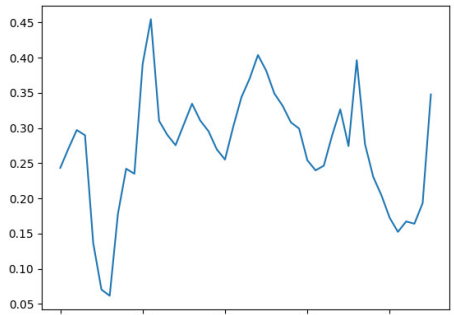
(d) t = 8

Some Views of the Incremental Reconstruction (2/2).

Results (9/10)



(a) Scipy's NLS Gauss Newton (average ≈ 0.82)



(b) Pytorch's Adam (average ≈ 0.27)

Figure: Comparison for Reprojection Error for Middlebury dataset.

Results (10/10)

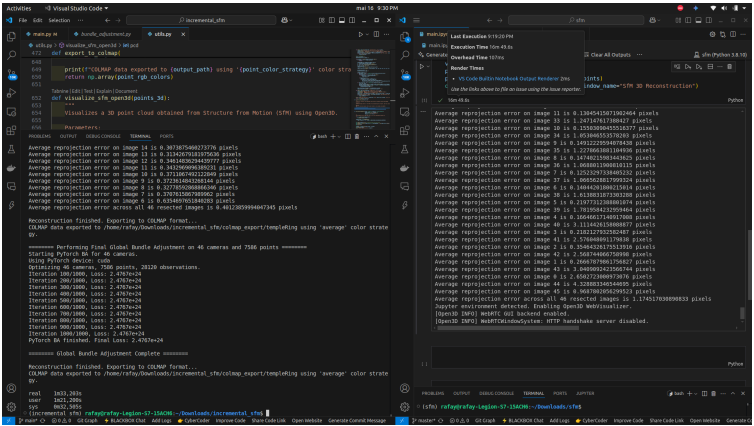


Figure: Pytorch's Adam v.s. Scipy's Gauss Newton NLS execution runtime/reprojection error on Middlebury (1m33s v.s. 16m / 0.4pix v.s. 1.17pix).

Discussion: Strengths

- **Robust Matching**: Lowe's ratio and fundamental matrix filtering ensure high-quality correspondences.
- **Incremental Approach**: Scales well by adding one image at a time.
- **Bundle Adjustment**: Significantly improves accuracy by joint optimization.
- **Flexible Initialization**: Percentage-based pair selection avoids hardcoding thresholds.
- **Calibration Independent**: Requires running calibration code to get calibration matrix **K**.
- **Pixel Colorization**: Thanks to pixel coloring the reconstruction is more realistic.
- **Camera Poses**: Having the camera poses helps in seeing which image contributed their 3D points to the overall point cloud.
- **Rapid Execution**: Thanks to parallel processing and simplified optimization.

Discussion: Limitations

- **Computational Cost**: All-to-all matching and least squares BA are expensive for large datasets.
- **Initialization Sensitivity**: Poor possible initial pair selection can lead to unstable triangulation, due to instable feature matching.
- **Outlier Sensitivity**: Some outliers may persist despite filtering.

Discussion: Future Improvements

- Implement parallel processing for feature extraction & matching.
- Integrate Bag of Visual Words for better feature matching.
- Improve the Least Squares approach by including dedicated linear solvers (ex: Ceres, g2o, etc...) .
- Explore deep learning-based feature matching for better robustness.

References

Cremers, D.

Lecture Notes on Computer Vision II: Multiple View Geometry.

Technical University of Munich.

Fusiello, A.

Lecture Notes on Computer Vision: 3D Reconstruction Techniques.

University of Udine, IT.

Geiger, A.

Lecture Notes on Computer Vision, Lecture 3 – Structure-from-Motion.

Autonomous Vision Group, University of Tübingen.

Stachniss, C.

Lecture Notes on Projective 3-Point (P3P) Algorithm / Spatial Resection.

University of Bonn.

Thank You