[SLAM] Bundle Adjustment Concept Review

> **Note:** This post is written for study purposes. If there are any errors or areas for improvement, please let me know.

In this post, we explain the process of performing **Bundle Adjustment (BA)** by minimizing the **reprojection error**, based on feature-based Visual Odometry (VO).

1. Introduction

When matching point pairs between two images are given, we can estimate the **Fundamental Matrix F** using the 8-point algorithm. If the camera is calibrated, we can obtain the **Essential Matrix E** from **F**.

$$\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K}$$

Where $\mathbf{K}$ is the camera intrinsic parameter matrix:

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

- f_x, f_y: Focal lengths.
- c_x, c_y: Principal point coordinates.

From the Essential Matrix, the relative pose between two cameras can be decomposed. Using the camera poses, we can calculate the 3D points for the matching pairs using **triangulation**.

Bundle Adjustment (BA) is the method of finding the optimal positions for **all camera poses** and **all 3D world points** simultaneously when given a sequence of camera images and matching pairs. In the context of feature-based VO, this is done by minimizing the **reprojection error**.

---

2. Reprojection Error

The core of BA is to minimize the difference between the *observed* pixel coordinates and the *projected* pixel coordinates of the 3D points.

Let: - $\mathbf{T}_i$: The pose of the i-th camera. - $\mathbf{X}_j$: The position of the j-th 3D world point. - $\mathbf{z}_{ij}$: The observed 2D feature point (pixel) of $\mathbf{X}_j$ in camera $\mathbf{T}_i$.

The **projection model** maps a 3D point $\mathbf{X}_j$ to the image plane of camera $\mathbf{T}_i$:

$$\pi(\mathbf{T}_i, \mathbf{X}_j) = \mathbf{K} \cdot [\mathbf{R}_i | \mathbf{t}_i] \cdot \mathbf{X}_j$$

(Note: In practice, we normalize the coordinates and apply distortion models, but the basic form is the perspective projection).

The **Reprojection Error** $\mathbf{e}_{ij}$ is defined as:

$$\mathbf{e}_{ij} = \mathbf{z}_{ij} - \pi(\mathbf{T}_i, \mathbf{X}_j)$$

Our goal is to find the optimal parameters $\mathbf{X}^*$ (all poses and points) that minimize the sum of squared errors:

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmin}} \sum_{i,j} \|\mathbf{e}_{ij}(\mathbf{X})\|_{\Sigma}^2$$

---

3. Optimization (Non-linear Least Squares)

Since the projection function is non-linear, we use **Non-linear Least Squares** methods, such as **Gauss-Newton** or **Levenberg-Marquardt (LM)**.

We iteratively update the state vector $\mathbf{x}$ by a small increment $\Delta\mathbf{x}$:

$$\mathbf{x}_{new} = \mathbf{x}_{old} \oplus \Delta\mathbf{x}$$

Using the first-order Taylor expansion, we approximate the error function:

$$\mathbf{e}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{e}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x}$$

Where $\mathbf{J}$ is the **Jacobian matrix** (partial derivatives of the error with respect to the parameters). To find the optimal $\Delta\mathbf{x}$, we solve the **Normal Equation**:

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$$

Where: - $\mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J}$ (Hessian Matrix approximation) - $\mathbf{b} = \mathbf{J}^T\mathbf{W}\mathbf{e}$

---

4. Jacobian and Hessian Sparsity

The state vector consists of camera poses ($\mathbf{x}_c$) and 3D points ($\mathbf{x}_p$). Because a specific 3D point is only observed by a subset of cameras, the Jacobian $\mathbf{J}$ is very sparse. Consequently, the Hessian $\mathbf{H}$ has a specific **arrowhead** or **block-sparse** structure:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{cc} & \mathbf{H}_{cp} \\ \mathbf{H}_{pc} & \mathbf{H}_{pp} \end{bmatrix}$$

- $\mathbf{H}_{cc}$: Camera-Camera block (usually diagonal-like if cameras are not directly linked, but dense in the reduced system).
- $\mathbf{H}_{pp}$: Point-Point block (Diagonal, because points typically do not depend on each other).
- $\mathbf{H}_{cp}, \mathbf{H}_{pc}$: Camera-Point off-diagonal blocks.

---

5. Schur Complement (Marginalization)

Solving the large linear system $\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$ directly is computationally expensive due to the large number of 3D points. However, we can exploit the structure of $\mathbf{H}$ using the **Schur Complement**.

We define the linear system as:

$$\begin{bmatrix} \mathbf{H}_{cc} & \mathbf{H}_{cp} \\ \mathbf{H}_{cp}^T & \mathbf{H}_{pp} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_c \\ \Delta\mathbf{x}_p \end{bmatrix} = \begin{bmatrix} -\mathbf{b}_c \\ -\mathbf{b}_p \end{bmatrix}$$

Since $\mathbf{H}_{pp}$ is diagonal (and easy to invert), we can eliminate $\Delta\mathbf{x}_p$ first. We multiply the second row by $\mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}$ and subtract it from the first row:

$$\underbrace{(\mathbf{H}_{cc} - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{H}_{cp}^T)}_{\mathbf{H}_{schur}} \Delta\mathbf{x}_c = \underbrace{-\mathbf{b}_c + \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{b}_p}_{\mathbf{b}_{schur}}$$

1. **Solve for camera poses** $\Delta\mathbf{x}_c$ using the reduced system (size depends only on the number of cameras).
2. **Back-substitute** to find point updates $\Delta\mathbf{x}_p$:

$$\Delta\mathbf{x}_p = \mathbf{H}_{pp}^{-1}(-\mathbf{b}_p - \mathbf{H}_{cp}^T\Delta\mathbf{x}_c)$$

This trick, often called **Marginalization**, significantly speeds up the Bundle Adjustment process.

> **References** - *Detailed derivations for Jacobians can be found in related SLAM literature. - Figures representing the sparsity pattern and the graph structure are omitted in this text format.*