Bundle Adjustment – A Report

JesseyGe

Bundle adjustment is the problem of refining view reconstruction to produce jointly optimal 3D structure and camera parameter(pose and/or calibration) estimates. The optimal results can be found by minimizing some cost function that satisfies a model with respect to both structure and camera parameter. The word 'bundle' refers to bundle of light rays projecting each 3D feature onto each images and converging on each camera center, which are 'adjusted' to optimal about both feature and camera parameters. This article is a report of the theory and the implement method of bundle adjustment, aimed at 3D reconstruction using multiple cameras.

The rest of the report is organized as follow. Section 2 provides a brief introduction to basic theory concering camera projection model and bundle parameterization. Section 3 explains the entire process of bundle adjustment in detail. Section 4 gives a implementation detail during the process of bundle adjustment. Experiment results based on simulation are presented in Section 6 and our findings is concluded in Section 5.

1 Basic

1.1 Camera Model

I begin the development of bundle adjustment by considering the basic image projection model. The camera imaging process is a mapping from the three-dimensional world to the two-dimensional image, which can be described by a geometric model such as pinhole camera model. Pinhole camera model is a basic and common projection model, which describe the geometric relationship of a bundle of light passing through a small pinhole and imaging on the back of the pinhole. Firstly, four related coordinate systems are introduced:

- **World Coordinate System**: It is used to represent 3D features and camera pose in the real world.
- **Camera Coordinate System**: This is a coordinate system that uses the camera's optical center as the origin. The camera's optical axis is the **Z** axis, and the **X** and **Y** axes are parallel to the imaging plane.
- **Image Coordinate System**: The image coordinate system is a coordinate system in physical dimensions (unit: mm) in the imaging plane. The principle point is the coordinate origin, is the intersection of the optical axis and the imaging plane. The **x** axis is parallel to the horizontal axis of the imaging plane, and the **y** axis is parallel to the vertical axis of the imaging plane.
- **Pixel Coordinate System**: The pixel coordinate system is a coordinate system in pixels. The origin of the coordinates is the upper left corner of the image. The **u** axis is parallel to the **x** axis of the image coordinate system, and the **v** axis is parallel to the **y** axis of the image coordinate system.

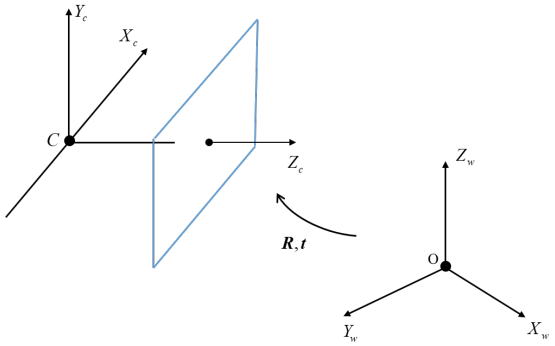


Figure 1: The transformation from the world coordinate system to the camera coordinate system.

The transformation between the world coordinate system and the camera coordinate system is a rigid transformation. The world coordinate system can be transformed into the camera coordinate system by rotation and translation. This is illustrated in Figure 1. Assuming that both world and camera points are represented by homogeneous vectors, the rigid transformation is wirtten as

$$\begin{pmatrix} X_{\mathrm{c}} \\ Y_{\mathrm{c}} \\ Z_{\mathrm{c}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} X_{\mathrm{w}} \\ Y_{\mathrm{w}} \\ Z_{\mathrm{w}} \\ 1 \end{pmatrix}, \tag{1}$$

where $\mathbf{P_w} = (X_{\mathrm{w}}\ Y_{\mathrm{w}}\ Z_{\mathrm{w}}\ 1)^{\mathrm{T}}$ is a 3D point in the world frame, and $\mathbf{P_c} = (X_{\mathrm{c}}\ Y_{\mathrm{c}}\ Z_{\mathrm{c}}\ 1)^{\mathrm{T}}$ is the point in the camera frame. $\mathbf{R}$ is an orthogonal 3×3 matrix and $\mathbf{t}$ is a 3×1 vector, which also known as the extrinsic parameter of camera. $\mathbf{R}$ and $\mathbf{t}$ represents the pose and position of camera relative to the world coordinate system.

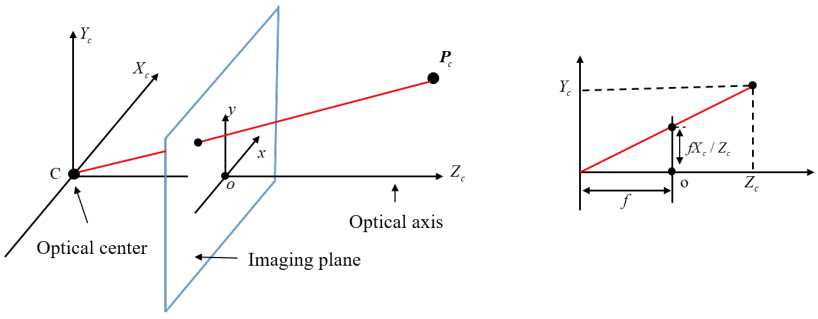


Figure 2: The transformation from the camera coordinate system to the image coordinate system.

As is illustrated in Figure 2, a 3D point $\mathbf{P_c}$ in the camera coordinate system projecting on an image point represented by a 3×1 vector $\mathbf{p}_c$. Assuming $z = f$ is the imaging plane, this mappin is written compactly as

$$\begin{cases} x_{\mathrm{c}} = f\frac{X_{\mathrm{c}}}{Z_{\mathrm{c}}} \\ y_{\mathrm{c}} = f\frac{Y_{\mathrm{c}}}{Z_{\mathrm{c}}} \end{cases}. \tag{2}$$

However, the images captured by the camera are expressed in pixels, so the imaging plane needs to be sampled and quantized. The difference between the pixel coordinate system and the image plane coordinate system is an offset between the scaling and the principal point. Assuming α and β are the scale factor of u axis and v axis respectively, and $(c_x\ c_y)^{\mathrm{T}}$ is the translation of the origin.

The coordinate of $\mathbf{p}_c$ in the pixel coordinate system is

$$\begin{cases} u = \alpha x_{\mathrm{c}} + c_x \\ v = \beta y_{\mathrm{c}} + c_y. \end{cases} \tag{3}$$

Let $f_x = \alpha f$, and $f_y = \beta f$. Thus, the Equation (3) can now rewritten as

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{Z_{\mathrm{c}}} \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & f_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_{\mathrm{c}} \\ Y_{\mathrm{c}} \\ Z_{\mathrm{c}} \\ 1 \end{pmatrix} \overset{\text{def}}{=} \frac{1}{Z_{\mathrm{c}}} \mathbf{K}\mathbf{P}_c. \tag{4}$$

where $\mathbf{K}$ is the 3×3 upper triangular intrinsic calibration matrix which encodes the camera's optical properties(i.e., focal length, aspect ratio, and principal point).

Consequently, the camera projection model can be defined as

$$Z_{\mathrm{c}} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & f_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{t} \end{pmatrix} \begin{pmatrix} X_{\mathrm{w}} \\ Y_{\mathrm{w}} \\ Z_{\mathrm{w}} \\ 1 \end{pmatrix}, \tag{5}$$

Equation (5) can expressed by another form:

$$\lambda \mathbf{u} = \mathbf{M}\mathbf{P}, \tag{6}$$

where $\mathbf{u}$ is a homogeneous 3×1 vector representing an image point and $\mathbf{P}$ is a 3D point represented by a 4×1 vector. λ is an an arbitrary scale factor. $\mathbf{M}$, known as projection matrix, is a 3×4 matrix.

With the above definitions, the problem of multiview 3D reconstruction can now be defined more precisely as follows. Assuming n 3D points are observed in m images and denote by $\mathbf{u}_{ij}$(the jth point on the ith image). The task of multiview 3D reconstruction becomes to find the m camera matrix $\mathbf{M}_i$ and n 3D points $\mathbf{P}_j$, which approximately satisfy the Equation (6). It should be pointed out that each $\mathbf{M}_i$ and $\mathbf{P}_j$ are both the outcome of a measurement process by some multiview geometry theory. Therefore, they are subject to noise. At this point, some solution serve as the initial value to seek for more accurate estimates based on BA.

1.2 Bundle Parameterization

Bundle adjustment, essentially, is a parameter estimaton problem. The parameter to be estimated, including 3D points and camera parameter, can be put together into a state vector $\mathbf{x}$.

In this article, camera calibration parameter is given.Therefore, I consider 3D points and camera pose(rotation and translation). The bundle parameterization requres a minimum parameterization form, otherwise it will cause over-parameter problems.

3D points: It is well kown that the degree of freedom for 3D points are 3. Obviously, a 3×1 vector can represent a 3D point in the world coordinate.

Camera poses: Although rotation and translation are usually expressed in the form of a matrix, since the degree of freedom of the camera pose is 6, we need a 6-parameterized form to represent the camera pose. Also, the rotation matrix does not have a well-defined addition operation. Therefore, the Jacobian of the rotation matrix during the optimization process cannot be found. $\mathfrak{se}(3)$ is the Lie algebra corresponding to the Euclidean transformation, and its form is shown below:

$$\mathfrak{se}(3) = \{\boldsymbol{\xi} = \begin{pmatrix} \rho \\ \boldsymbol{\phi} \end{pmatrix} \in \mathbb{R}^6, \boldsymbol{\rho} \in \mathbb{R}^3, \boldsymbol{\phi} \in \mathfrak{se}(3)\}, \tag{7}$$

where $\boldsymbol{\xi}$ represents a $\mathfrak{se}(3)$ element. The first three dimensions represent translation and the last three dimensions represent rotation. In $\mathfrak{se}(3)$, a six-dimensional vector can be converted into a four-dimensional matrix through operations:

$$\boldsymbol{\xi}^{\wedge} = \begin{pmatrix} \boldsymbol{\phi}^{\wedge} & \boldsymbol{\rho} \\ \mathbf{0}^{\mathrm{T}} & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4}. \tag{8}$$

In $\mathfrak{se}(3)$, the perturbation model can be used to calculate the derivative of the objective function with respect to the transformation matrix to adjust the current estimate.

2 Bundle Adjustment

In section 1.1, the problem that both camera poses and 3D points to be estimated are noisy is mentioned. The noise come from two reasons: *localization* errors arise when the cameras are not detected in the correct locations and *matching* errors that image matching points are not correctly associated.

To decrease the noise, we can define a prediction model based on the state parameter. Hence, bundle adjustment can be viewed as optimizing a complicated nonlinear cost function(**the total prediction error**) with respcet to a large state space(**3D points and the camera parameter**). The entire process of bundle adjustment includes:

1) Error Modelling: According to the projection model of the camera, a loss function can be constructed, and it is a nonlinear least squares model.

2) Normal Equation Solving: The iterative method is used to linearize the non-linear least squares model and transform it into the solution of normal equations. According to the sparse nature of bundle adjustment, the updating state vector is solved.

3) Optimization: The non-linear model is fitted by the non-linear optimization method (Gauss-Newton, Levenberg-Marquardt), and the state amount is iteratively updated according to the update amount obtained by the normal equation, thereby obtaining the optimal state estimate.

2.1 Error Modelling

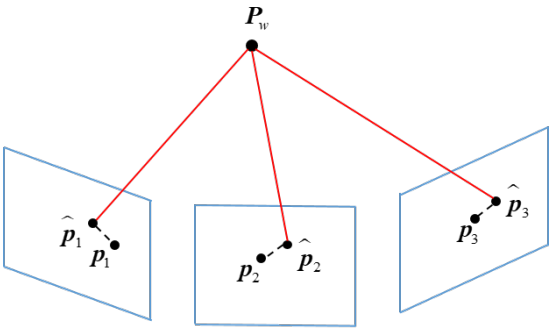


Figure 3: Reprojection errors.

As is illustrated in Figure 3, the projection position of a 3D point is different from the observation position because of noise, which we called is **reprojection errors**. Therefore, we hope to estimate the optimal state parameters through a parameter estimation method. One of the most basic parameter estimation methods is nonlinear leaset squares.

Assuming that n 3D points are all visible in m views and let $\mathbf{x}_{ij}$ denote projection of the ith point on jth image. The task of bundle adjustment is to jointly refine a set of given camera and structure parameter estimates for finding the most accurately estimations of n points and m cameras, which can be called maximum likelihood estimation (MLE). Assuming that each camera j is parameterized by a $\mathfrak{se}(3)$ element $\mathbf{a}_j$ and each 3D point i by a 3×1 vector $\mathbf{b}_i$, we can put them into a state vector $\mathbf{x}$. Since the dimensions of each $\mathbf{a}_j$ and $\mathbf{b}_i$ is 6 and 3, the total number of $\mathbf{x}$ equals $6m + 3n$. For example, in the case of 1000 points projection on 4 cameras, the size of $\mathbf{x}$ is 3024.

According the camera model in section 1.1, we can have vectors of observations $\mathbf{z}_{ij}$ predicted by the projection model $\hat{\mathbf{z}}_{ij} = \mathbf{Q}(\mathbf{a}_i, \mathbf{b}_j)$. Here ,we can define the reprojection error:

$$\mathbf{e}(\mathbf{x}_{ij}) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij} \tag{9}$$

where $\mathbf{e}(\mathbf{x}_{ij})$ represent the error of the ith point on the image j. BA minimize the reprojection error with respect to all 3D points and cameras parameters:

$$\mathbf{x}^* = \arg\min_{\mathbf{a}_i, \mathbf{b}_j} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} d(\mathbf{Q}(\mathbf{a}_i, \mathbf{b}_j), \mathbf{z}_{ij}) \tag{10}$$

where $\mathbf{x}$ is the state vector containing the camera pose and 3D points location to be estimated. And $d(\mathbf{x}, \mathbf{y})$ denotes the distance between $\mathbf{x}$ and $\mathbf{y}$. Equation (10) can be rewritten as another form:

$$\mathbf{x}^* = \arg\min \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{m} \mathbf{e}(\mathbf{x}_{ij})^{\mathrm{T}} \mathbf{W}_{ij} \mathbf{e}(\mathbf{x}_{ij}) \tag{11}$$

Obviously, Equation (11) is a nonlinear least square model. We hope to find the optimal $\mathbf{x}$ by adjusting each camera pose $\mathbf{a}_j$ and the position of 3D point $\mathbf{b}_i$.

2.2 Solution for Nolinear Least Square

Assuming a least square model:

$$F(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{m} (f_i(\mathbf{x}))^2, \tag{12}$$

where f_i is the ith residual function, which is the difference between predictions and observations. $\mathbf{x}$, its dimension is n, is the model parameter. We can find a local minimum value x^*:

$$F(\mathbf{x}^*) < F(\mathbf{x}) \qquad \text{s.t.} \quad ||\mathbf{x} - \mathbf{x}^*|| < \delta, \tag{13}$$

where δ represents a sufficiently small value. Supposing $F(\mathbf{x})$ is smooth and derivable, $F(\mathbf{x})$ can be expressed a second-order Taylor expansion:

$$F(\mathbf{x} + \Delta\mathbf{x}) \approx F(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^{\mathrm{T}} \mathbf{H} \Delta\mathbf{x}, \tag{14}$$

where $\mathbf{J}$ and $\mathbf{H}$ represent the first and second derivative matrices of the loss function $F(\mathbf{x})$ for the variable $\mathbf{x}$ respectively.

As we know, for a linear least square model, the derivative of the function can be found to find the direction of the gradient descent, so as to find the local minimum. However, it is hard to find the derivative of the nonlinear model. Therefore, we turn to think of an iterative method to find a

descent direction so that the nonlinear loss function $F(\mathbf{x})$ gradually decreases with the iteration of $\mathbf{x}$ until converging to x^*:

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k). \tag{15}$$

For simplicty, we first consider iterative methods for solving **linear models**. The iterative method includes two step: finding the unit vector $\mathbf{d}$ of descent direction and determining the step α of decline. Supposing α is small enough, we can perform a first-order taylor expansion on the loss function $F(\mathbf{x})$:

$$F(\mathbf{x} + \alpha\mathbf{d}) \approx F(\mathbf{x}) + \alpha\mathbf{J}\mathbf{d}. \tag{16}$$

The descend direction olny meet:

$$\mathbf{J}\mathbf{d} < 0, \tag{17}$$

and the decreasing step can be found through the line search:

$$\alpha^* = \arg\min_{\alpha>0}\{F(\mathbf{x} + \alpha\mathbf{d})\}. \tag{18}$$

There are two basic iteration methods: *steepest descent* and *Newton method*. The steepest descent method applies to the beginning of the iteration. According to the condition of descent direction:

$$\mathbf{J}\mathbf{d} = ||\mathbf{J}||cos\theta, \tag{19}$$

where θ represents the angle between the descent direction and the gradient direction. If $\theta = \pi$, Equation (16) becomes

$$\mathbf{d} = -\frac{\mathbf{J}^{\mathrm{T}}}{||\mathbf{J}||}. \tag{20}$$

The disadvantages of the steepest descent method are oscillation near the optimal value, slow convergence. The Newton method is applied when near the optimal vaule. In the neighbour of the local optimal vector $\mathbf{x}^*$, if $\mathbf{x} + \Delta\mathbf{x}$ is the optimal vaule, the derivative of the loss function for $\Delta\mathbf{x}$ is equal to 0. Take the first derivative of Equation (14):

$$\frac{\partial(F(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}}\mathbf{H}\Delta\mathbf{x})}{\partial\Delta\mathbf{x}} = \mathbf{J}^{\mathrm{T}} + \mathrm{H}\Delta\mathbf{x} = 0, \tag{21}$$

Therefore,

$$\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{J}^{\mathrm{T}}. \tag{22}$$

However, the calculation of the second derivative matrix $\mathbf{H}$ is complicated.

We now consider iterative methods for a **nonlinear model**. Here, we turn to analyis a nonlinear vector function f instead of $F(\mathbf{x})$. We want to minimize $||\mathbf{f}(\mathbf{x})||$, or equivalently to find

$$\mathbf{x}^* = \arg\min_{\mathbf{x}}\{F(\mathbf{x})\}, \tag{23}$$

where

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}(f_i(\mathbf{x}))^2 = \frac{1}{2}||\mathbf{f}(\mathbf{x})||^2 = \frac{1}{2}\mathbf{f}(\mathbf{x})^{\mathrm{T}}\mathbf{f}(\mathbf{x}) \tag{24}$$

Provided that $\mathbf{f}$ has continuous second partial derivatives, its first-order Taylor approximation can be written as

$$\mathbf{f}(\mathbf{x}+\Delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x}, \tag{25}$$

where $\mathbf{J}$ is the jacobian ontaining the first partial derivatives of the function components. If $\mathbf{J}_i = \frac{\partial f_i(\mathbf{x})}{\partial \mathbf{x}}$, then we have:

$$\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}(\mathbf{x}) \tag{26}$$

As regards $F(\mathbf{x})$:

$$\frac{\partial F}{\partial x_j}(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}) \frac{\partial f_i}{\partial x_j}(\mathbf{x}). \tag{27}$$

Thus the gradient is

$$F^{'}(\mathbf{x}) = \mathbf{J}^{\mathrm{T}}\mathbf{f}(\mathbf{x}), \tag{28}$$

We shall also need the Hessian of $F(\mathbf{x})$. From Equation (27) we see that the element in position (j, k) is

$$\frac{\partial^2 F}{\partial x_j \partial x_k}(\mathbf{x}) = \sum_{i=1}^{m} (\frac{\partial f_i}{\partial x_j}(\mathbf{x}) \frac{\partial f_i}{\partial x_k}(\mathbf{x}) + f_i(\mathbf{x}) \frac{\partial^2 f_i}{\partial x_j \partial x_k}(\mathbf{x})), \tag{29}$$

showing that

$$F^{''}(\mathbf{x}) = \mathbf{J}^{\mathrm{J}}\mathbf{J} + \sum_{i=1}^{m} f_i(\mathbf{x}) f^{''}(\mathbf{x}) \approx \mathbf{J}^{\mathrm{J}}\mathbf{J}. \tag{30}$$

There are two basis of the very efficient methods for nonlinear least sauqre are *the Gauss-Newton method* and *the Lenvenberg-Marquardt method*. Next we describe them specifically.

The Gauss-Newton method is based on a linear approximation to the com- ponents of $f(\mathbf{x})$ (a linear model of $f(\mathbf{x})$) in the neighbourhood of $\mathbf{x}$. Inserting Equation (25) in the Equation (24), we can see that

$$\begin{aligned} F(\mathbf{x}+\Delta\mathbf{x}) \approx L(\mathbf{x}) &= \frac{1}{2}(\mathbf{f}(\mathbf{x}) + \mathbf{J}\mathbf{x})^{\mathrm{T}}(\mathbf{f}(\mathbf{x}) + \mathbf{J}\mathbf{x}) \\ &= \frac{1}{2}\mathbf{f}(\mathbf{x})^{\mathrm{T}}\mathbf{f}(\mathbf{x}) + \Delta\mathbf{x}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{f}(\mathbf{x}) + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{J}\Delta\mathbf{x} \\ &= F(\mathbf{x}) + \Delta\mathbf{x}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{f}(\mathbf{x}) + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{J}\Delta\mathbf{x}. \end{aligned} \tag{31}$$

It is easily seen that the Jacobian and Hessian of $F(\mathbf{x})$

$$L^{'}(\mathbf{x}) = \mathbf{J}^{\mathrm{T}}\mathbf{f}(\mathbf{x}), \qquad L^{''}(\mathbf{x}) = \mathbf{J}^{\mathrm{T}}\mathbf{J}. \tag{32}$$

$F^{''}(\mathbf{x})$ is symmetric and if $\mathbf{J}$ has full rank. Also, if the columns are linearly independent, then $L^{''}(\mathbf{x})$ is positive definite. Equation (31) implies that $F(\mathbf{x})$ has a unique minimizer, which can be found by solving normal equation:

$$(\mathbf{J}^{\mathrm{T}}\mathbf{J})\Delta\mathbf{x}_{\mathrm{gn}} = -\mathbf{J}^{\mathrm{T}}\mathbf{f}, \tag{33}$$

with $\mathbf{f} = \mathbf{f}(\mathbf{x})$. Thus, we can solve Equation (33) and then the iterative step is

$$\mathbf{x} = \mathbf{x} + \alpha \Delta \mathbf{x}_{\text{gn}} \tag{34}$$

where α is found by line search. The classical Gauss-Newton method uses $\alpha = 1$ in all steps.

The Lenvenberg-Marquart method, suggested by Levenberg and Marquardt, is an iterative method that find a local minimum of nolinear function that is expressed the sum of square. It improves the Gauss-Newton method, and introduces a damping factor during the solution

$$(\mathbf{J}^{\text{T}}\mathbf{J} + \mu\mathbf{I})\Delta \mathbf{x}_{\text{lm}} = -\mathbf{J}^{\text{T}}\mathbf{f}, \qquad \text{with} \quad \mu \geq 0, \tag{35}$$

where μ is the damping factor. The damping parameter μ has several effects:

- For $\mu > 0$, $(\mathbf{J}^{\text{T}}\mathbf{J} + \mu\mathbf{I})$ is positive definite, and this ensures that $\Delta \mathbf{x}_{\text{lm}}$ is a descent direction.
- If μ is very large, we can get

  $$\Delta \mathbf{x}_{lm} = \frac{1}{\mu}\mathbf{J}^{\text{T}}\mathbf{f}, \tag{36}$$

  and the step is similar to the steepest descent direction. This is good if the current iterate is far from the solution.
- If μ is very small, we can get

  $$\Delta \mathbf{x}_{lm} \approx \Delta \mathbf{x}_{gn}, \tag{37}$$

  and it is similar to the step in the Gauss-Newton method. This is good if the current iterate is close to the solution.

The vaule μ need to be set at the initial step of the iteration, and continuously updated during the iteration. The choice of initial μ-value is related to the size of the elements in $\mathbf{J}^{\text{T}}\mathbf{J}$. The eigenvalues of semi-definite coefficient matrix $\mathbf{J}^{\text{T}}\mathbf{J}$ are $\{\lambda_j\}$, and the eigenvectors are $\{v_j\}$. After Eigen Decomposition of $\mathbf{J}^{\text{T}}\mathbf{J}$: $\mathbf{J}^{\text{T}}\mathbf{J} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\text{T}}$, we can get(The process of proof is given in Appendix A.)

$$\Delta \mathbf{x}_{\text{lm}} = -\sum_{j=1}^{n} \frac{\mathbf{v}_j^{\text{T}}\mathbf{F}'^{\text{T}}}{\lambda_j + \mu}\mathbf{v}_j. \tag{38}$$

Thus, a simple strategy for the initial damping factor μ_0 is

$$\mu_0 = \tau \cdot \max\{(\mathbf{J}^{\text{T}}\mathbf{J})_{ii}\}, \tag{39}$$

where τ is set $[10^{-8}, 1]$ up to users. In the process of iterations:

- If the value loss function $F(\mathbf{x})$ increase after adopting the current step $\Delta\mathbf{x}$, the current iteration should be rejected. Then, we should increase μ-value and reduce the step $\Delta\mathbf{x}$.
- If the value loss function $F(\mathbf{x})$ decrease after updating the step, we should decrease the damping value μ and increase $\Delta\mathbf{x}$.

The specific updating is controlled by the ratio

$$\rho = \frac{F(\mathbf{x}) - F(\mathbf{x} + \Delta\mathbf{x}_{lm})}{L(\mathbf{0}) - L(\Delta\mathbf{x}_{\mathrm{lm}})}, \tag{40}$$

Algorithm 1 LevenbergMarquardt method

Input: A n-dimension nonlinear residual vector function f, a n-dimension measurement vector $\mathbf{z}$ and a m-dimension initial parameters estimation $\mathbf{x}_0$.

Output: A optimal estimation vector $\mathbf{x}^*$.

Algorithm:

$k := 0$; $v := 2$; $\mathbf{x} := \mathbf{x}_0$
$\mathbf{H} := \mathbf{J}^{\mathrm{T}}\mathbf{J}$; $\epsilon_{\mathbf{x}} = \mathbf{z} - f(\mathbf{x})$; $\mathbf{b} := \mathbf{J}^{\mathrm{T}}\epsilon_{\mathbf{x}}$;
stop := false; $\mu := \tau \cdot \max\{(\mathbf{J}^{\mathrm{T}}\mathbf{J})_{ii}\}$
while (not *stop*) and ($k < k_{\max}$) **do**
  $k := k + 1$; $cnt := 0$
  repeat
    Solve $(\mathbf{H} + \mu\mathbf{I})\Delta\mathbf{x}_{\mathrm{lm}} = \mathbf{b}$;
    if (($||\Delta\mathbf{x}_{\mathrm{lm}}|| \leq \epsilon_1$) and ($cnt < cnt_{\max}$)) **then**
      $\mathbf{x}_{\mathrm{new}} := \mathbf{x} + \Delta\mathbf{x}_{\mathrm{lm}}$;
      $\rho := (||\epsilon_{\mathbf{x}}||^2 - ||\mathbf{z} - f(\mathbf{x}_{\mathrm{new}})||)^2/(\Delta_{\mathrm{lm}}^{\mathrm{T}}(\mu\Delta_{\mathrm{lm}} + \mathbf{b}))$
      if $\rho > 0$ **then**
        $\mu := \mu * \max\{\frac{1}{3}, 1 - (2\rho - 1)^3\}$; $v := 2$;
        $\mathbf{x} = \mathbf{x}_{\mathrm{new}}$;
        $cnt := 0$;
        $\mathbf{H} := \mathbf{J}^T\mathbf{J}$; $\mathbf{b} = -\mathbf{J}^{\mathrm{T}}\mathbf{f}$;
      else
        $\mathbf{x}_{\mathrm{new}} := \mathbf{x} - \Delta\mathbf{x}_{\mathrm{lm}}$;
        $\mu := \mu * v$; $v := 2 * v$; $cnt := cnt + 1$;
      end if
    end if
  until $\rho > 0$ or *stop*
  stop := ($||\epsilon||_{\mathbf{x}} < \epsilon_2$); $k := k + 1$;
end while
$\mathbf{x}^* := \mathbf{x}$

where the denominator is the predicted by the linear model

$$\begin{aligned} L(\mathbf{0}) - L(\Delta\mathbf{x}_{\mathrm{lm}}) &= -\Delta\mathbf{x}_{\mathrm{lm}}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{f} - \frac{1}{2}\Delta\mathbf{x}_{\mathrm{lm}}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{J}\Delta\mathbf{x}_{\mathrm{lm}} \\ &\overset{\mathbf{b}=-\mathbf{J}^{\mathrm{T}}\mathbf{f}}{=} -\frac{1}{2}\Delta\mathbf{x}_{\mathrm{lm}}^{\mathrm{T}}(-2\mathbf{b} + (\mathbf{J}^{\mathrm{T}}\mathbf{J} + \mu\mathbf{I} - \mu\mathbf{I})\Delta\mathbf{x}_{\mathrm{lm}}) \\ &= \frac{1}{2}\Delta\mathbf{x}_{\mathrm{lm}}^{\mathrm{T}}(\mu\Delta\mathbf{x}_{\mathrm{lm}} + \mathbf{b}). \end{aligned} \tag{41}$$

Note that both $\Delta\mathbf{x}_{\mathrm{lm}}^{\mathrm{T}}\Delta\mathbf{x}_{\mathrm{lm}}$ and $\Delta\mathbf{x}_{\mathrm{lm}}^{\mathrm{T}}\mathbf{b}$ are positive, so $L(\mathbf{0}) - L(\Delta\mathbf{x}_{\mathrm{lm}})$ is guaranteed to be positive. And, The effect of ρ on the updating step is as follows:

- If ρ is a large value, we should reduce μ so that the next Levenberg-Marquardt step is closer to the Gauss-Newton step and makes the model converge faster.

- If ρ is a small value(may be negative), we should increase μ so that Levenberg-Marquardt is getting closer to the steepest descent method.

Concerning to the strategy of the damping, Marquardt proposed a updating method

$$\begin{aligned} &\text{if} \quad \rho < 0.25 \\ &\quad \mu := \mu * 2 \\ &\text{elseif} \quad \rho > 0.75 \\ &\quad \mu := \mu/3. \end{aligned} \tag{42}$$

Besides, Nielsen has proposed another strategy:

$$\begin{aligned} &\text{if} \quad \rho > 0 \\ &\quad \mu := \mu * \max\{\frac{1}{3}, 1 - (2\rho - 1)^3\}; \quad v := 2 \\ &\text{else} \\ &\quad \mu := \mu * v; \quad v := 2 * v. \end{aligned} \tag{43}$$

The LM algorithm using Nielsen's damping update strategy is summarized in Algorithm 1, and it terminates when any of the following conditions is meet:

- a maximum number of iterations $k_{\max}$ is reached,
- a maximum number of false count $cnt_{\max}$ is reached,
- the magnitude of the updating step drops below a threshold ϵ_1;
- the magnitude of the residual ϵ drops below a threshold ϵ_2

Algorithm 2 The Kwak's damping parameter strategy

```
μ := 0.1;   β := 0.1
if (F(x_new) < F(x)) then
    wip := (dot(x_new, x))/(||x_new|| · ||x||);
    μ := μ · β^wip;
else
    D := H;
    if (D is positive) then
        μ := μ/β;
    else
        d_ij := D,   d_ii := Σ_{i≠j} |d_ij|   for all i;
        μ := min{d_ii}   for all i;
    end if
end if
```

In addition, I have tried a new damping update strategy proposed by Kwak. It use the inner product of the updated state and the state before the updating to reduce the damping factor, and apply a diagonally dominant matrix to increase the damping factor. The inner product of the weighted matrix can prevent the oscillation of the state vector at the beginning of the iteration. In addition, the damping parameter reduction μ is adaptively determined according to the direction of the weighted matrix $\mathbf{D}$. The decay rate β^{wip} of μ depends on the inner product of state vector,

where β is set initially, usually 0.1, and wip is the inner product of updated state vector $\mathbf{x}_{\text{new}}$ and the state vector without updating $\mathbf{x}$. A large value of $wip(-1 \leq wip \leq 1)$ indicates that $F(\Delta\mathbf{x}_{\text{lm}})$ is a good approximation $F(\mathbf{x}+\Delta\mathbf{x}_{lm})$, and we can decrease μ through wip so that the next Levenberg-Marquardt step is closer to the Gauss-Newton step. If wip is small, usually at the beginning of iterations, then $F(\Delta\mathbf{x}_{\text{lm}})$ is a poor approximation, and we should increase μ to get closer to the steepest descent direction. The whole strategy is shown in the Algorithm 2.

2.3 Sparse Bundle Adjustment

According to the error model in section 2.1, BA can be cast as a nonlinear least square problem as follows:

$$\min_{\mathbf{a}_i,\mathbf{b}_j} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} d(\mathbf{Q}(\mathbf{a}_i,\mathbf{b}_j),\mathbf{z}_{ij}), \tag{44}$$

where $d(\mathbf{x},\mathbf{y})$ denotes the distance (Euclidean distance or Mahalanobis distance) between $\mathbf{x}$ and $\mathbf{y}$.

For simplicty, we now assume that $n = 4$ points are visible in $m = 3$ cameras, and the calibration parameters are given, which can be represented by a graph model in Figure 4.

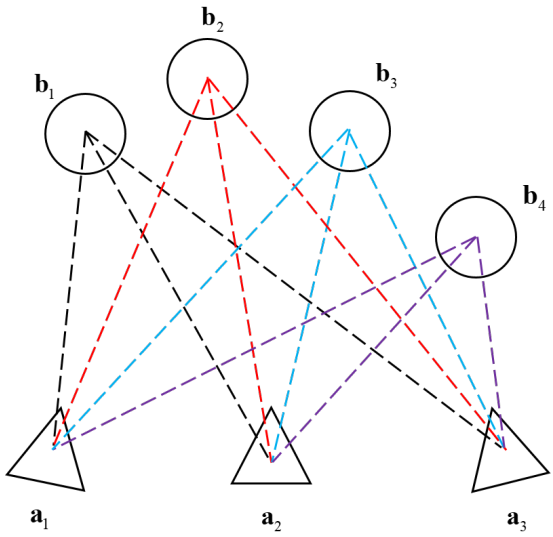


Figure 4: The graph model of projection model.

A parameter vector $\mathbf{P}$ is defined by all parameters describing the $m = 3$ camera matrix(rotation and translation) and the $n = 4$ 3D ponts in Equation (10), namely, $\mathbf{x} = (\mathbf{a}_1^{\mathrm{T}}, \mathbf{a}_2^{\mathrm{T}}, \mathbf{a}_3^{\mathrm{T}}, \mathbf{b}_1^{\mathrm{T}}, \mathbf{b}_2^{\mathrm{T}}, \mathbf{b}_3^{\mathrm{T}}, \mathbf{b}_4^{\mathrm{T}})^{\mathrm{T}}$. Assuming all of 3D points can be visible in 3 images, a observation vector $\mathbf{Z}$ is made up of the observed image point coordinates across all cameras:

$$\mathbf{Z} = (\mathbf{z}_{11}^{\mathrm{T}}, \mathbf{z}_{12}^{\mathrm{T}}, \mathbf{z}_{13}^{\mathrm{T}}, \mathbf{z}_{21}^{\mathrm{T}}, \mathbf{z}_{22}^{\mathrm{T}}, \mathbf{z}_{23}^{\mathrm{T}}, \mathbf{z}_{31}^{\mathrm{T}}, \mathbf{z}_{32}^{\mathrm{T}}, \mathbf{z}_{33}^{\mathrm{T}}, \mathbf{z}_{41}^{\mathrm{T}}, \mathbf{z}_{42}^{\mathrm{T}}, \mathbf{z}_{43}^{\mathrm{T}})^{\mathrm{T}}. \tag{45}$$

According the loss function in Equation (44), an estimated vector $\hat{\mathbf{Z}}$ is defined as

$$\hat{\mathbf{Z}} = (\hat{\mathbf{z}}_{11}^{\mathrm{T}}, \hat{\mathbf{z}}_{12}^{\mathrm{T}}, \hat{\mathbf{z}}_{13}^{\mathrm{T}}, \hat{\mathbf{z}}_{21}^{\mathrm{T}}, \hat{\mathbf{z}}_{22}^{\mathrm{T}}, \hat{\mathbf{z}}_{23}^{\mathrm{T}}, \hat{\mathbf{z}}_{31}^{\mathrm{T}}, \hat{\mathbf{z}}_{32}^{\mathrm{T}}, \hat{\mathbf{z}}_{33}^{\mathrm{T}}, \hat{\mathbf{z}}_{41}^{\mathrm{T}}, \hat{\mathbf{z}}_{42}^{\mathrm{T}}, \hat{\mathbf{z}}_{43}^{\mathrm{T}})^{\mathrm{T}}, \tag{46}$$

with $\hat{\mathbf{Z}} = \mathbf{Q}(\mathbf{a}_i,\mathbf{b}_j)$. The initial parameter estimation $\mathbf{x}_0$ can be measured through image processing and multiple geometry theroy. Besides, let $\Sigma_{\mathbf{Z}}$ represents the covariance matrix expressing the

uncertainty the observation vector $\mathbf{M}$. If there is no prior knowledge, $\Sigma_{\mathbf{Z}}$ is equal to the identiy matrix.

Thus, BA is now equal to minimize the distance $\mathbf{e}(\mathbf{z})^{\mathrm{T}}\mathbf{e}(\mathbf{z})$, $\mathbf{e}(\mathbf{z}) = \mathbf{Z} - \hat{\mathbf{Z}}$ over parameters $\mathbf{X}$. We here use Mahalanobis distance to calculate the distance between the observed image points and the estimated points, and then equtiaon (44) becomes:

$$\min_{\mathbf{a}_i,\mathbf{b}_j} \frac{1}{2} \sum_{j=1}^{4} \sum_{i=1}^{3} \mathbf{e}_{ij}(\mathbf{z})^{\mathrm{T}} \Sigma_{\mathbf{z}_{ij}}^{-1} \mathbf{e}_{ij}(\mathbf{z}). \tag{47}$$

Obviously, the optimal estimaton of Equation (47) can be found by LM algorithm described in section 2.2, which calls for repeated solving the weighted normal equatons

$$(\underbrace{\mathbf{J}^{\mathrm{T}}\Sigma_{\mathbf{z}}^{-1}\mathbf{J}}_{\mathbf{H}} + \mu\mathbf{I})\Delta\mathbf{x} = \underbrace{-\mathbf{J}^{\mathrm{T}}\Sigma_{\mathbf{z}}^{-1}\mathbf{e}}_{\mathbf{b}}, \tag{48}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{e}$ and $\Delta\mathbf{x}$:

$$\mathbf{J} = \frac{\partial \mathbf{e}}{\partial \Delta\mathbf{x}} = \begin{pmatrix} \frac{\partial \mathbf{e}_{11}}{\partial \boldsymbol{\Delta}\mathbf{x}} \\ \frac{\partial \mathbf{e}_{21}}{\partial \boldsymbol{\Delta}\mathbf{x}} \\ \frac{\partial \mathbf{e}_{31}}{\partial \boldsymbol{\Delta}\mathbf{x}} \\ \frac{\partial \mathbf{e}_{12}}{\partial \boldsymbol{\Delta}\mathbf{x}} \\ \vdots \\ \frac{\partial \mathbf{e}_{34}}{\partial \boldsymbol{\Delta}\mathbf{x}} \end{pmatrix} = \begin{pmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \mathbf{J}_3 \\ \mathbf{J}_4 \\ \vdots \\ \mathbf{J}_{12} \end{pmatrix}. \tag{49}$$

Thus, Equation (48) can be rewritten int the form of cumulative:

$$\sum_{k=1}^{12} (\mathbf{J}_k^{\mathrm{T}}\Sigma_{\mathbf{z}}^{-1}\mathbf{J}_k + \mu\mathbf{I})\boldsymbol{\Delta}\mathbf{x} = -\mathbf{J}_k^{\mathrm{T}}\Sigma_{\mathbf{z}}^{-1}\mathbf{e}. \tag{50}$$

Because each residual is only related to several state quantities, the Jacobian of the irrelevant terms is 0 when the Jacobian matrix is derived. Specifically, $\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{a}_t} = \mathbf{0}$, $\forall j \neq t$ and $\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{b}_t} = \mathbf{0}$, $\forall i \neq t$. Thus, jacobian matrix $\mathbf{J}_1$ is

$$\mathbf{J}_1 = \frac{\partial \mathbf{e}_{11}}{\partial \boldsymbol{\Delta}\mathbf{x}} = \begin{pmatrix} \frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{51}$$

and,

$$\mathbf{H}_1 = \mathbf{J}_1\Sigma_1^{-1}\mathbf{J}_1 = \begin{pmatrix} (\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1})^{\mathrm{T}}\Sigma_1^{-1}\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1})^{\mathrm{T}}\Sigma_1^{-1}\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1})^{\mathrm{T}}\Sigma_1^{-1}\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1})^{\mathrm{T}}\Sigma_1^{-1}\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{52}$$

Note that the matrix $\mathbf{H}$ is not a real second-order derivative matrix (Hessian matrix), but a second-order derivative matrix that is approximately expressed in the form of $\mathbf{J}^{\mathrm{T}}\mathbf{J}$. In the same way, we can calculate $\mathbf{H}_k, \forall k = 1, \ldots, 12$, see Appendix B. Finally, by accumulating all $\mathbf{H}_k$, we can get the $\mathbf{H}$ in

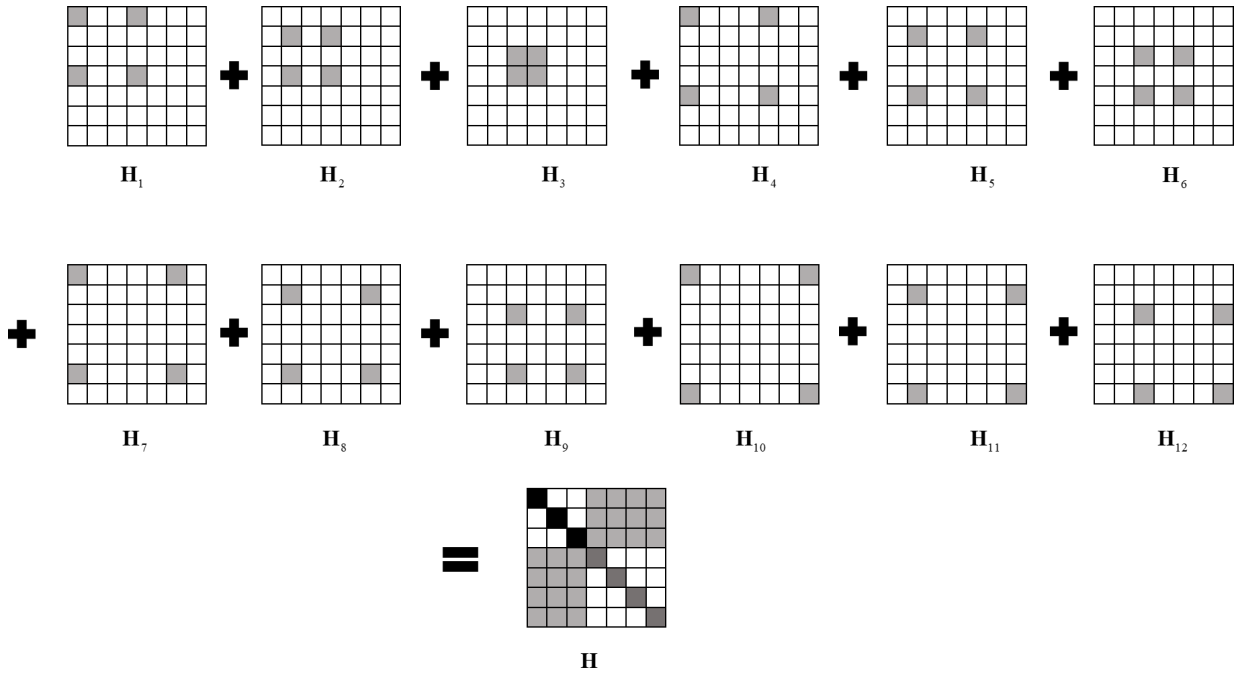


Figure 5: The process of accumulation of $\mathbf{H}$, which we can the sparse nature.

the normal equation. The Visualization of accumulation is shown in Figure 5. Supplementally, the sparse nature of Jacobian $\mathbf{J}$ is shown below:

$$\mathbf{J} = \begin{pmatrix} \frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{e}_{21}}{\partial \mathbf{a}_2} & \mathbf{0} & \frac{\partial \mathbf{e}_{21}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{31}}{\partial \mathbf{a}_3} & \frac{\partial \mathbf{e}_{31}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \frac{\partial \mathbf{e}_{12}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{12}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{32}}{\partial \mathbf{a}_3} & \mathbf{0} & \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \\ \frac{\partial \mathbf{e}_{13}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{13}}{\partial \mathbf{b}_3} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{e}_{23}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{23}}{\partial \mathbf{b}_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{33}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{33}}{\partial \mathbf{b}_3} & \mathbf{0} \\ \frac{\partial \mathbf{e}_{14}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{14}}{\partial \mathbf{b}_2} \\ \mathbf{0} & \frac{\partial \mathbf{e}_{24}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{24}}{\partial \mathbf{b}_3} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{34}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{34}}{\partial \mathbf{b}_4} \end{pmatrix}. \tag{53}$$

As is shown in Figure 5, we can express the matrix $\mathbf{H}$ in the left-hand side of Equation (48) as

$$\mathbf{H} = \begin{pmatrix} \mathbf{U}_1 & 0 & 0 & \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} & \mathbf{W}_{14} \\ 0 & \mathbf{U}_2 & 0 & \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} & \mathbf{W}_{24} \\ 0 & 0 & \mathbf{U}_3 & \mathbf{W}_{31} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{34} \\ \mathbf{W}_{11}^{\mathrm{T}} & \mathbf{W}_{21}^{\mathrm{T}} & \mathbf{W}_{31}^{\mathrm{T}} & \mathbf{V}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{12}^{\mathrm{T}} & \mathbf{W}_{22}^{\mathrm{T}} & \mathbf{W}_{32}^{\mathrm{T}} & \mathbf{0} & \mathbf{V}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{13}^{\mathrm{T}} & \mathbf{W}_{23}^{\mathrm{T}} & \mathbf{W}_{33}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & \mathbf{V}_3 & \mathbf{0} \\ \mathbf{W}_{14}^{\mathrm{T}} & \mathbf{W}_{24}^{\mathrm{T}} & \mathbf{W}_{34}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{V}_4 \end{pmatrix}. \tag{54}$$

where

$$\mathbf{U}_i = \sum_{j=1}^{4} (\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{a}_i})^{\mathrm{T}} \Sigma_{\mathbf{z}_{ij}}^{-1} \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{a}_i}, \quad \mathbf{V}_j = \sum_{i=1}^{3} (\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{b}_j})^{\mathrm{T}} \Sigma_{\mathbf{z}_{ij}}^{-1} \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{b}_j}, \quad \mathbf{W}_{ij} = (\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{a}_i})^{\mathrm{T}} \Sigma_{\mathbf{z}_{ij}}^{-1} \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{b}_j}. \tag{55}$$

we can block the matrix $\mathbf{H}$

$$\mathbf{H} = \begin{pmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{pmatrix}. \tag{56}$$

Also, using Equations (53) and (55), we can expand the right-hand side of Equation (48) as

$$\begin{aligned} \mathbf{b} = (& \sum_{j=1}^{4} (\frac{\partial \mathbf{e}_{1j}}{\partial \mathbf{a}_i})^{\mathrm{T}} \Sigma_{\mathbf{z}_{1j}}^{-1} \mathbf{e}_{1j} \quad \sum_{j=1}^{4} (\frac{\partial \mathbf{e}_{2j}}{\partial \mathbf{a}_i})^{\mathrm{T}} \Sigma_{\mathbf{z}_{1j}}^{-1} \mathbf{e}_{2j} \quad \sum_{j=1}^{4} (\frac{\partial \mathbf{e}_{3j}}{\partial \mathbf{a}_i})^{\mathrm{T}} \Sigma_{\mathbf{z}_{1j}}^{-1} \mathbf{e}_{3j} \\ & \sum_{i=1}^{3} (\frac{\partial \mathbf{e}_{i1}}{\partial \mathbf{b}_j})^{\mathrm{T}} \Sigma_{\mathbf{z}_{i1}}^{-1} \mathbf{e}_{i1} \quad \sum_{i=1}^{3} (\frac{\partial \mathbf{e}_{i2}}{\partial \mathbf{b}_j})^{\mathrm{T}} \Sigma_{\mathbf{z}_{i2}}^{-1} \mathbf{e}_{i2} \quad \sum_{i=1}^{3} (\frac{\partial \mathbf{e}_{i3}}{\partial \mathbf{b}_j})^{\mathrm{T}} \Sigma_{\mathbf{z}_{i3}}^{-1} \mathbf{e}_{i3} \quad \sum_{i=1}^{3} (\frac{\partial \mathbf{e}_{i4}}{\partial \mathbf{b}_j})^{\mathrm{T}} \Sigma_{\mathbf{z}_{i4}}^{-1} \mathbf{e}_{i4} \ )^{\mathrm{T}}. \end{aligned} \tag{57}$$

Letting

$$\mathbf{e}_{\mathbf{a}_i} = \sum_{j=1}^{4} (\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{a}_i})^{\mathrm{T}} \Sigma_{\mathbf{z}_{ij}}^{-1} \mathbf{e}_{ij}, \quad \mathbf{e}_{\mathbf{b}_j} = \sum_{i=1}^{3} (\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{b}_i})^{\mathrm{T}} \Sigma_{\mathbf{z}_{ij}}^{-1} \mathbf{e}_{ij}, \quad \text{with } \mathbf{e}_{ij} = \mathbf{z}_{ij} - \hat{\mathbf{z}_{ij}} \ \forall i, j, \tag{58}$$

we can rewrite the vector in Equation (57) to

$$\mathbf{b} = \begin{pmatrix} \mathbf{e}_{\mathbf{a}_1}^{\mathrm{T}} & \mathbf{e}_{\mathbf{a}_2}^{\mathrm{T}} & \mathbf{e}_{\mathbf{a}_3}^{\mathrm{T}} & \mathbf{e}_{\mathbf{b}_1}^{\mathrm{T}} & \mathbf{e}_{\mathbf{b}_2}^{\mathrm{T}} & \mathbf{e}_{\mathbf{b}_3}^{\mathrm{T}} & \mathbf{e}_{\mathbf{b}_4}^{\mathrm{T}} \end{pmatrix}^{\mathrm{T}}. \tag{59}$$

After denoting the Euqation (48), we can further compact the normal eqation to

$$\begin{pmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{x}_{\mathbf{a}} \\ \Delta \mathbf{x}_{\mathbf{b}} \end{pmatrix} = \begin{pmatrix} \mathbf{e}_{\mathbf{a}} \\ \mathbf{e}_{\mathbf{b}} \end{pmatrix}, \tag{60}$$

where,

- the dimension of $\mathbf{U}$ is 18×18 ($6m \times 6m$),
- the dimension of $\mathbf{V}$ is 12×12 ($3n \times 3n$),
- the dimension of $\mathbf{W}$ is 18×12 ($6m \times 3n$),
- the dimension of $\mathbf{W}^T$ is 12×18 ($3n \times 6m$),
- the whole dimension of $\mathbf{H}$ is 30×30 ($(6m + 3n) \times (6m + 3n)$).

Obviously, (60) is the linear equations $\mathbf{H}\Delta\mathbf{x} = \mathbf{b}$. There are many ways to accelerate the solution of this equation by using the sparse nature of $\mathbf{H}$. Here we use a common method: **Schur elimination**, which can be seen in Appendix C.

In formula (60), $\mathbf{U}$ is a diagonal block matrix, and the dimensions of each diagonal block are the same as the dimensions of the camera parameters. Since the number of three-dimensional points is much larger than the number of camera variables, the dimension of $\mathbf{V}$ is often much larger than $\mathbf{U}$. Since the complexity of finding the inverse of a diagonal block matrix is much less than the complexity of a general matrix, we only need to inverse the diagonal matrix blocks separately. Considering this special characteristic, we can use Schur elimination on the linear equations. The goal is to eliminate the non-diagonal part $\mathbf{W}$ in the upper right corner. Left multiplication of Equation (18) by the block matrix

$$\begin{pmatrix} \mathbf{I} & -\mathbf{W}\mathbf{V}^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{x}_{\mathbf{a}} \\ \Delta \mathbf{x}_{\mathbf{b}} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & -\mathbf{W}\mathbf{V}^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{e}_{\mathbf{a}} \\ \mathbf{e}_{\mathbf{b}} \end{pmatrix}, \tag{61}$$

results in

$$\begin{pmatrix} \mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x_a} \\ \Delta\mathbf{x_b} \end{pmatrix} = \begin{pmatrix} \mathbf{e_a} - \mathbf{W}\mathbf{V}^{-1}\mathbf{e_b} \\ \mathbf{e_b} \end{pmatrix}, \tag{62}$$

Since the top right block of the above left hand matrix is zero, the dependence of the camera parameters on the structure parameters has been eliminated in Equation (62). Therefore, we can get the incremental equation about camera pose, which is

$$(\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}})\Delta\mathbf{x_a} = \mathbf{e_a} - \mathbf{W}\mathbf{V}^{-1}\mathbf{e_b}. \tag{63}$$

This linear equation has the same dimensions as $\mathbf{U}$, and can be effiiently solved using the CHolesky factorization of $\mathbf{S} = \mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}$, which is the Schur complement of $\mathbf{V}$. And then, we can substitute the solution $\Delta\mathbf{x_a}$ into the original equation to solve $\Delta\mathbf{x_b}$:

$$\mathbf{V}\Delta\mathbf{x_b} = \mathbf{e_b} - \mathbf{W}^{\mathrm{T}}\Delta\mathbf{x_a}. \tag{64}$$

The choice of solving first for $\Delta\mathbf{x_a}$ and then for $\Delta\mathbf{x_b}$ is justified by the fact that the total number of camera parameters is in general much smaller than the total number of 3D points parameters.

2.4 The Complexity of One Bundle Adjustment Iteration

The Lenvenberg-Marquardt algorithm is well-known iterative method, and has computational complexity of $O((m+n)^3)$ per iteration, where m is the number of cameras and n is the number of 3D points. However, the bundle adjustment has the nature of spase, and LM algorithm can take advantage of this characteristic to reduce comlexity.

According to the section 2.3, the block structure of matrix $\mathbf{H}$ is illustrated in Figure 6. This matrix is composed of three matrix $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$, $\mathbf{W}^{\mathrm{T}}$, where $\mathbf{U}$, $\mathbf{V}$ are block-diagnoal.

- $\mathbf{U}$ is made of diagonal 6×6 blocks representative of the relations between measurements of the ith image and associated cameras parameters.
- $\mathbf{V}$ is made of diagonal 3×3 blocks, and represents the dependence between 3D point j and measurement asscociated it.
- $\mathbf{W}$ is made of a number of 6×3 blocks equal to the number of 2D reprojections. This matrix represents the correlation between camera and 3D points.
- $\mathbf{W}^{\mathrm{T}}$ is made of a number of 3×6 blocks, and is the transpose of $\mathbf{W}$.

According to Equations (63) and (64), the problem is solved by two steps. Let m and n represent the number of cameras and 3D points respectively. Let k be the number of projecting points in each camera. After $\mathbf{H}$ is calculated, the two time consuming are

- The matrix product $\mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}$
- The resolution of linear equation (63).

For matrix product $\mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}$, we usually calculate the matrix product $\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}$ first. Since $\mathbf{V}^{-1}$ is block diagnoal and its number is equal to the number of reprojections k, the time complexity of $\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}$ is equal to km. Then, in the product $\mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}$, each not-null 3×6 block of $\mathbf{W}$ is used once in the calculation of each block column of $\mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}$. Therefore, the time complexity of $\mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}$

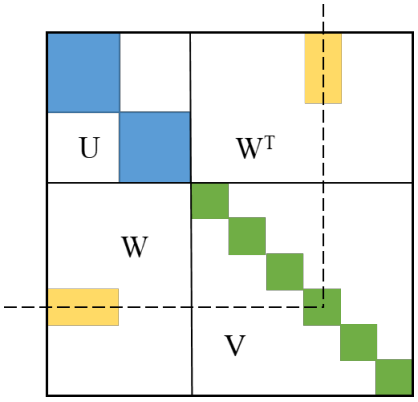


Figure 6: The structure of **H**.

is $O(km^2)$. The time complexity of the traditional resolution of the linear system is $O(m^3)$. Thus, the time complexity of bundle adjustment per iteration is

$$O(m^3 + km^2). \tag{65}$$

In particular, if all 3D points are visible in all cameras, the time complexity of one bundle adjustment iteration is

$$O(m^3 + mn) \tag{66}$$

3 Implemention Details

3.1 Camera Pose

The geometric relations between two images I_i and I_{i-1} are described by the so-called fundmental matrix **F**. **F** contains the camera motion parameters the intrinsic parameters in the following form:

$$\mathbf{F} \simeq \mathbf{K}^{-\mathrm{T}}\mathbf{t}^{\wedge}\mathbf{R}\mathbf{K}^{-1}, \tag{67}$$

where $\mathbf{t} = \begin{pmatrix} t_x & t_y & t_z \end{pmatrix}^{\mathrm{T}}$ and

$$\mathbf{t}^{\wedge} = \begin{pmatrix} 0 & -t_z & t-y \\ t_z 0 & -t_x & \\ -t_y & t_x & 0 \end{pmatrix}. \tag{68}$$

The symbol $\simeq$ is used to denote that the equivalence is valid up to a multiplicative scalar. For estimating motion between the calibrated cameras, the so-called essential matrix **E** only contains the camera motion parameters up to an unknown scale factor for the translation in the following form:

$$\mathbf{E} \simeq \mathbf{t}^{\wedge}\mathbf{R}. \tag{69}$$

and

$$\mathbf{F} \simeq \mathbf{K}^{-\mathrm{T}}\mathbf{E}\mathbf{K}^{-1}. \tag{70}$$

The specific deduction process of **F** and **E** is shown in Appendix D.

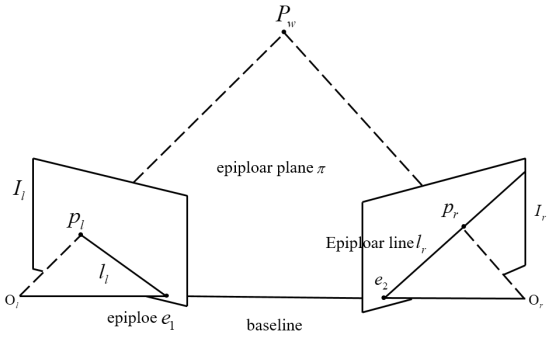


Figure 7: An illustration of the epipolar constraint.

The fundmental matrix can be computed from 2D-to-2D feature correspondences, and rotation and translation can be extracted from $\mathbf{F}$. As is shown in Figure 7, the main property of 2D-to-2D-based motion estimation is the epipolar constraint, which determines the line on which the corresponding feature point $\mathbf{p}_2$ of $\mathbf{p}_1$ lies in the other image. This constraint can be formulated by

$$\mathbf{p}_2^{\mathrm{T}}\mathbf{F}\mathbf{p}_1 = 0, \tag{71}$$

where $\mathbf{p}_1$ is a feature location in one image (e.g., I_{i-1}) and $\mathbf{p}_2$ is the location of its corresponding feature in another image (e.g., I_i).

The fundmental matrix can be computed from 2D-to-2D feature correspondences using the epipolar constraint. A simple and straightforward solution for $n \geq 8$ noncoplanar points is the eight-point algorithm, which is summarized here. Each feature match gives a constraint of the following form:

$$\begin{pmatrix} u_2u_1 & u_2v_1 & u_2 & v_2u_1 & v_2v_1 & v_2 & u_1 & v_1 & 1 \end{pmatrix} \cdot \mathbf{F} = 0, \tag{72}$$

where $\mathbf{F} = \begin{pmatrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 \end{pmatrix}^{\mathrm{T}}$. Stachking the constrains from eight points gives the linear equation system $\mathbf{AF} = 0$, and by solving the system, the parameters of F can be computed. This homogeneous equation system can easily be solved using singular value decomposition (SVD). Having more than eight points leads to an overdetermined system to solve in the least squares problem and provides a degree of robustness to noise. The SVD of $\mathbf{A}$ has the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}, \tag{73}$$

where $\mathbf{U}$ and $\mathbf{V}$ is orthogonal matrix, and $\mathbf{\Sigma}$ is singular value matrix. The least squares estimate of $\mathbf{F}$ with $||\mathbf{F}|| = 1$ can be found as the last column of $\mathbf{V}$. However, this linear estimation of $\mathbf{F}$ does not fulfill the inner constrains of an fundmental matrix, which come from the multiplication of the rotation matrix $\mathbf{R}$ and the skew-symmetric translation matrix $\mathbf{t}^{\wedge}$. A valid fundmental matrix after SVD has $\mathbf{\Sigma} = \mathrm{diag}\{\sigma, \sigma, 0\}$, which means that the first and second singular values are equal and the third one is zero. To get a valid $\mathbf{F}$ that fulfills the constraints, $\mathbf{\Sigma}$ is adjusted deliberately to the above form. Thus, the projected fundmental matrix is

$$\mathbf{F} = \mathbf{U}\mathrm{diag}\{\frac{\sigma_1 + \sigma_2}{2}, \frac{\sigma_1 + \sigma_2}{2}, 0\}\mathbf{V}^{\mathrm{T}}, \tag{74}$$

where $\sigma_1 \geq \sigma_2 \geq \sigma_3$ are the singular values.

After finding $\mathbf{F}$, the eseential matrix $\mathbf{E}$ can be computed by

$$\mathbf{E} = \mathbf{K}^{\mathrm{T}}\mathbf{F}\mathbf{K}. \tag{75}$$

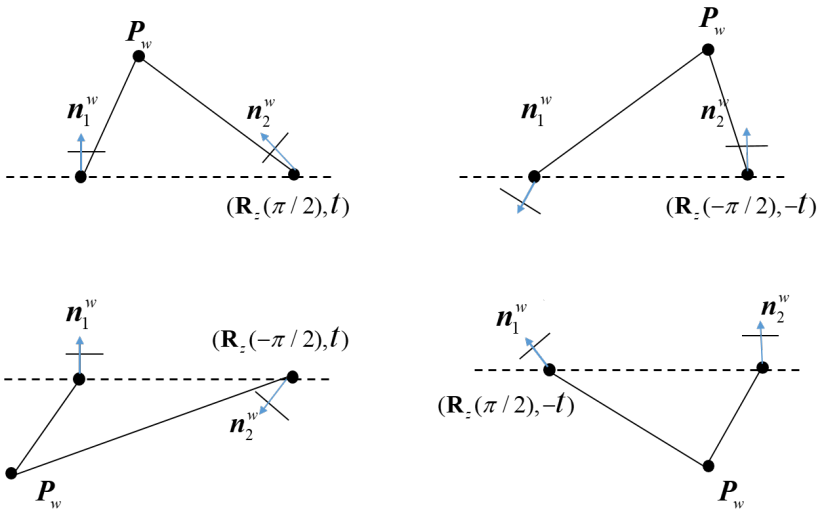


Figure 8: Four solutions obtained by decomposing the essence matrix. Between the left and right sides there is a baseline. Note, only in (a) is the reconstructed point in front of both cameras.

The rotation and translation parts can be extracted from the essential matrix $\mathbf{E}$. For an arbitrary essential matrix $\mathbf{E}$, there are four different solution for $\mathbf{R}$, $\mathbf{t}$:

$$\begin{aligned} \mathbf{R} &= \mathbf{U}(\pm\mathbf{W}^{\mathrm{T}})\mathbf{V}^{\mathrm{T}}, \\ \mathbf{t} &= \mathbf{U}(\pm\mathbf{W})\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}, \end{aligned} \tag{76}$$

where

$$\mathbf{W}^{\mathrm{T}} = \mathbf{R}_z(\pm\pi/2) = \begin{pmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{77}$$

Figure 8 graphically shows the four solutions obtained from SVD of the essential matrix. Among these four possible cases, only one solution has a positive depth in both cameras. Choose the optimal solution among the 4 solutions according to the following conditions

(1) The reconstructed 3D point is in front of the camera

$$\begin{cases} \mathbf{n}_1^{\mathrm{w}} \cdot (\mathbf{P}_{\mathrm{w}} - \mathbf{O}_1) > 0 \\ \mathbf{n}_2^{\mathrm{w}} \cdot (\mathbf{P}_{\mathrm{w}} - \mathbf{O}_2) > 0 \end{cases}, \tag{78}$$

(2) The model has the largest parallax angle(parallax angle: $\angle \mathbf{O}_2\mathbf{P}\mathbf{O}_1$).

Thus, the correct $\mathbf{R}$, $\mathbf{t}$ pair can be identified by triangulation of a single point.

3.2 Triangulation

Assuming that a 3D point $P_{\mathrm{w}} = \begin{pmatrix} X_{\mathrm{w}} & Y_{\mathrm{w}} & Z_{\mathrm{w}} & 1 \end{pmatrix}^{\mathrm{T}}$ is visible in images $i = 1, \ldots, m$, each observation $\mathbf{u}_i = \begin{pmatrix} u_i & v_i & 1 \end{pmatrix}^{\mathrm{T}}$. Thus, the projection model is

$$\forall i,\ \lambda_i \mathbf{u}_i = \mathbf{M}_i \mathbf{P}_{\mathrm{w}}, \quad \text{with } \mathbf{M}_i = \mathbf{K} \begin{pmatrix} \mathbf{R}_i & \mathbf{t}_i \end{pmatrix} \tag{79}$$

where $\begin{pmatrix}\mathbf{R}_i & \mathbf{t}_i\end{pmatrix}$ is the exrtinsic matrix of the ith camera, which is from the world frame to the camera frame. According to the third line of the above equation, λ_i is

$$\lambda_i = \mathbf{M}_{i,3}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}}, \tag{80}$$

where $\mathbf{M}_{i,3}^{\mathrm{T}}$ is the third line of $\mathbf{M}$. Thus,

$$\begin{cases} u_i \mathbf{M}_{i,3}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} = \mathbf{M}_{i,1}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} \\ v_i \mathbf{M}_{i,3}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} = \mathbf{M}_{i,2}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} \end{cases} . \tag{81}$$

Each observation will provide two such equations, and move $\mathbf{P}_{\mathrm{w}}$ to the side of the equation

$$\begin{pmatrix} u_1 \mathbf{M}_{1,3}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} - \mathbf{M}_{1,1}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} \\ v_1 \mathbf{M}_{1,3}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} - \mathbf{M}_{1,2}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} \\ \vdots \\ u_m \mathbf{M}_{m,3}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} - \mathbf{M}_{m,1}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} \\ v_m \mathbf{M}_{m,3}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} - \mathbf{M}_{m,2}^{\mathrm{T}} \mathbf{P}_{\mathrm{w}} \end{pmatrix} \mathbf{P}_{\mathrm{w}} = \mathbf{0} \ \rightarrow \ \mathbf{A}\mathbf{P}_{\mathrm{w}} = \mathbf{0} \tag{82}$$

Thus, $\mathbf{P}_{\mathrm{w}}$ is a non-zero element in the Null space of $\mathbf{A}$. Since $\mathbf{A} \in \mathbb{R}^{2m \times 4}$, when the observation is twice or more than two, it is likely that $\mathbf{D}$ is full rank and has no Null space.

Therefore, we can find the least square solution

$$\min_{\mathbf{P}_w} ||\mathbf{A}\mathbf{P}_{\mathrm{w}} - \mathbf{0}||^2, \quad \text{s.t. } ||\mathbf{P}_{\mathrm{w}}|| = 1 \tag{83}$$

After making the Singular Value Decomposition (SVD) of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \sum_{s=1}^{4} \sigma_s \mathbf{d}_s \mathbf{d}_t, \tag{84}$$

where σ_s is the singular value in descending order, and $\mathbf{d}_s$ is orthogonal to $\mathbf{d}_t$. According to Appendix E, $\mathbf{P}_{\mathrm{w}} = \mathbf{d}_4$ is the solution to Equation (83). The solution is valid when the following conditions is met

- $\sigma_4 \ll \sigma_3$,
- the sign of projection of $\mathbf{P}_w$ is positive.

In addtion, the magnitude of D is significantly different in some cases, causing the solution to be numerically unstable. We can make a scaling operation on $\mathbf{A}$:

$$\mathbf{A}\mathbf{P}_{\mathrm{w}} = \underbrace{\mathbf{A}\mathbf{S}}_{\tilde{\mathbf{A}}} \underbrace{\mathbf{S}^{-1}\mathbf{P}_{\mathrm{w}}}_{\tilde{\mathbf{P}_{\mathrm{w}}}} = \mathbf{0}, \tag{85}$$

where $\mathbf{S}$ is a diagonal matrix, and is equal to the inverse of the largest element of D.

3.3 Jacobian

In section 2.3, $\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{a}_i}$ and $\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{b}_j}$ is used for constructing the coefficient matrix $\mathbf{H}$ of the normal equation. We now use Lie algebra to derive the specific form of $\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{a}_i}$ and $\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{b}_j}$.

Firstly, the first-order change of the Lie algebra of the reprojection error with respect to the camera pose is calculated. According to the Equations (5) and (6), the projection model can be written as

$$\lambda \mathbf{u} = \mathbf{K}\mathbf{P}_c = \mathbf{K}\mathbf{T}\mathbf{P}_{\mathrm{w}}, \quad \text{with } \mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \end{pmatrix} \tag{86}$$

where $\mathbf{P}_c = \begin{pmatrix} X_c & Y_c & W_c \end{pmatrix}^{\mathrm{T}}$ is the coordinate of the 3D point $\mathbf{P}_{\mathrm{w}}$ transformed into the camera coordinate system through $\mathbf{T}$, and $\mathbf{u}$ is the pixel coordinate in image. $\mathbf{K}$ is the intrinsic matrix of camera. Represented by matrix, Equation (86) becomes

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}. \tag{87}$$

Thus,

$$\hat{u} = f_x \frac{X_c}{Z_c} + c_x, \quad \hat{v} = f_y \frac{Y_c}{Z_c} + c_y. \tag{88}$$

When finding the residual vector, we are actually subtracting the actual measurement $\mathbf{u}$ from $\hat{\mathbf{u}}$ here:

$$\mathbf{e} = \mathbf{u} - \hat{\mathbf{u}} = \begin{pmatrix} u - \hat{u} \\ v - \hat{v} \end{pmatrix} = \begin{pmatrix} \delta u \\ \delta v \end{pmatrix}. \tag{89}$$

Left multiplication of the perturbation $\delta\mathbf{a}$ by $\mathbf{T}$, according to the chain rule, the derivative of the error $\mathbf{e}$ with respect to the perturbation can be

$$\frac{\partial \mathbf{e}}{\partial \delta \mathbf{a}} = \lim_{\delta \mathbf{a} \to \mathbf{0}} \frac{\mathbf{e}(\delta \mathbf{a} \oplus \mathbf{a}) - \mathbf{e}(\mathbf{a})}{\delta \mathbf{a}} = \frac{\partial \mathbf{e}}{\mathbf{P}_c} \frac{\partial \mathbf{P}_c}{\partial \delta \mathbf{a}}. \tag{90}$$

where $\oplus$ is the left multiplication perturbation on Lie algebra. According to Equation (88) and (89), the outcome of $\mathbf{e}$ derivative of $\mathbf{P}_c$ is

$$\frac{\partial \mathbf{e}}{\mathbf{P}_c} = -\begin{pmatrix} \frac{\partial \delta u}{\partial X_c} & \frac{\partial \delta u}{\partial Y_c} & \frac{\partial \delta u}{\partial Z_c} \\ \\ \frac{\partial \delta v}{\partial X_c} & \frac{\partial \delta v}{\partial Y_c} & \frac{\partial \delta v}{\partial Z_c} \end{pmatrix} = -\begin{pmatrix} \frac{f_x}{Z_c} & 0 & -\frac{f_x X_c}{Z_c^2} \\ 0 & \frac{f_y}{Z_C} & -\frac{f_y Y_c}{Z_c^2} \end{pmatrix}. \tag{91}$$

According Appendix F, derive $\mathbf{e}$ with respect to $\delta\mathbf{a}$:

$$\frac{\partial \mathbf{P}_{\mathrm{w}}}{\partial \delta \mathbf{a}} = \begin{pmatrix} \mathbf{I} & -\mathbf{P}_c^{\wedge} \\ \mathbf{0}^{\mathrm{T}} & 0 \end{pmatrix}. \tag{92}$$

Left multiplication of Equation (91) with the first three dimensions of Equation (92)

$$\frac{\partial \mathbf{P}_c}{\partial \delta \mathbf{a}} = \begin{pmatrix} \mathbf{I} & -\mathbf{P}_c^{\wedge} \end{pmatrix}. \tag{93}$$

results in

$$\frac{\partial \mathbf{e}}{\partial \delta \mathbf{a}} = -\begin{pmatrix} \frac{f_x}{Z_c} & 0 & -\frac{f_x X_c}{Z_c^2} & -\frac{f_x X_c Y_c}{Z_c^2} & f_x + \frac{f_x X_c^2}{Z_c^2} & -\frac{f_x Y_c}{Z_c} \\ 0 & \frac{f_y}{Z_c} & -\frac{f_y Y_c}{Z_c^2} & -f_y - \frac{f_y Y_c^2}{Z_c^2} & \frac{f_y X_c Y_c}{Z_c^2} & \frac{f_y X_c}{Z_c} \end{pmatrix}. \tag{94}$$

This Jacobian matrix describes the first-order relationship between the reprojection pixel error and the camera pose Lie algebra. The symbol $-$ in front of the matrix is defined by the observations $\mathbf{u}$ minus the predicted one $\hat{\mathbf{u}}$. Remove the sign if it is defined as the predicted minus observation.

The derivative of the reprojection error $\mathbf{e}$ with respect to P_{w} is discussed next. Also, we use the chain rule

$$\frac{\partial \mathbf{e}}{\mathbf{P}_{\mathrm{w}}} = \frac{\partial \mathbf{e}}{\partial \mathbf{P}_c} \frac{\partial \mathbf{P}_c}{\partial \mathbf{P}_{\mathrm{w}}}, \tag{95}$$

where $\mathbf{P}_c = \mathbf{R}\mathbf{P}_{\mathrm{w}} + \mathbf{t}$. Thus

$$\frac{\partial \mathbf{P}_c}{\partial \mathbf{P}_{\mathrm{w}}} = \mathbf{R}. \tag{96}$$

Combining Equation (91), the jacobian is

$$\frac{\partial \mathbf{e}}{\mathbf{P}_{\mathrm{w}}} = - \begin{pmatrix} \frac{f_x}{Z_c} & 0 & -\frac{f_x X_c}{Z_c^2} \\ 0 & \frac{f_y}{Z_c} & -\frac{f_y Y_c}{Z_c^2} \end{pmatrix} \mathbf{R}. \tag{97}$$

This Jacobian matrix describes the first-order relationship between the reprojection pixel error and the 3D point.

3.4 The Process of Bundle Adjustment

This paper implements the entire process of BA, from the restoration of 3D points and camera poses to the optimization of 3D point positions and camera poses. The implementation process of this article is as follows:

- **Step1: Data generation**. Multiple cameras and 3D points are generated randomly. According to the projection model of the camera, the corresponding image points can be generated, and Gaussian noise is added to the image points.
- **Step2: Camera pose recovery**. According to the fundmental or essential matrix(two methods) from noisy points, the rotation and translation of camera can be recovered. Usually, the first camera, regard as the world coordinate system, is fixed.
- **Step3: Triangulation**. According to the restored camera position, 3D points can be reconstructed by triangulation.
- **Step4: Bundle adjustment**. The LM algorithm is used to optimize the camera pose and 3D points simultaneously.

4 Experiment

We have done exprimental evaluation of the implemented algorithm for convergence, computational time and robustness to noise. Camera poses and 3D points are generated through simulation, and Gaussian noise of different level(i.e., 1, 2, 3, etc.., pixel) standard deviations is added to feature points. To reduce the interference of random errors during simulation, error for synthetic data, in all the experiments, is the mean error over ten trials.

This article chooses the LM algorithm for optimization, where the threshold for the number of iterations is 20, the threshold for state updates is 1e-5, and the threshold for the minimum error is set to 1e-10. We discuss the optimization results of the number of cameras in the range of 2 to 6, and the number of 3D points of different numbers.

4.1 Accuracy

This section mainly discusses the influence of the number of cameras, the number of 3D points, and the noise of image points on the optimization results. We evaluate and compare reprojection error, camera pose and 3D points location. Gaussian noise of standard deviation 1 pixel is added to the feature points.

Reprojection error comparision: Global reprojection error of the estimated 3D points in image is measured by the Root Mean Squares (RMS):

$$RMS = \sqrt{\frac{\sum_{ij} e_{ij}}{N_r}}, \tag{98}$$

where N_r is the total number of reprojecting points.

Camera pose: We can calculate the root mean square error of each pose Lie algebra to measure the error fo rotation and translation for each camera.

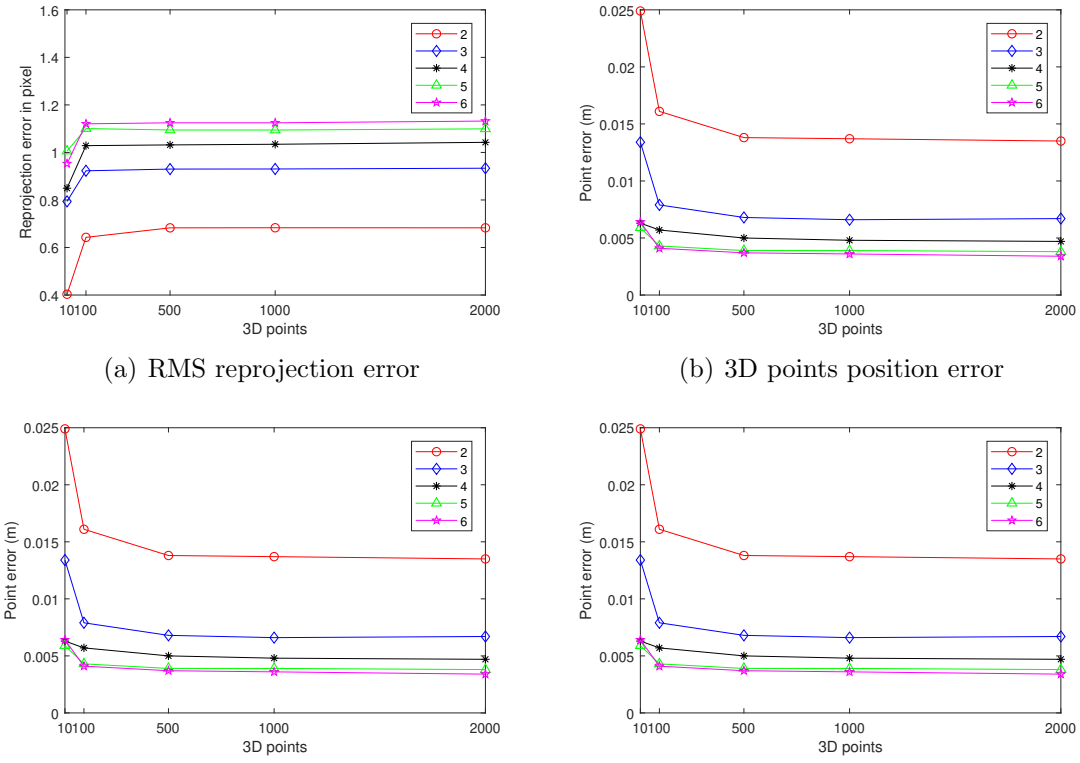
$$RMSE = \sqrt{\frac{1}{m}\sum_{i}^{m} ||\log \mathbf{T}_{\text{gt},i}^{-1}\mathbf{T}_{\text{est},i}||_2^2}\,, \tag{99}$$

where $\mathbf{T}_{\text{gt},i}$ and $\mathbf{T}_{\text{est},i}$ are ground truth and estimated pose respectively.

3D points location error: We comapre the mean error for all of 3D points position(in meters):

$$RMSE = \frac{1}{N_p}\sum_{j}^{n} \sqrt{e_{xj} + e_{yj} + e_{zj}}, \tag{100}$$

where N_p is the total number of 3D points. Besides, e_{xj}, e_{yj}, e_{zj} are position errors in directions x, y and z of the first camera (the world coordinate) compared to the groundtruth value.



(a) RMS reprojection error

(b) 3D points position error

(c) The rotation error

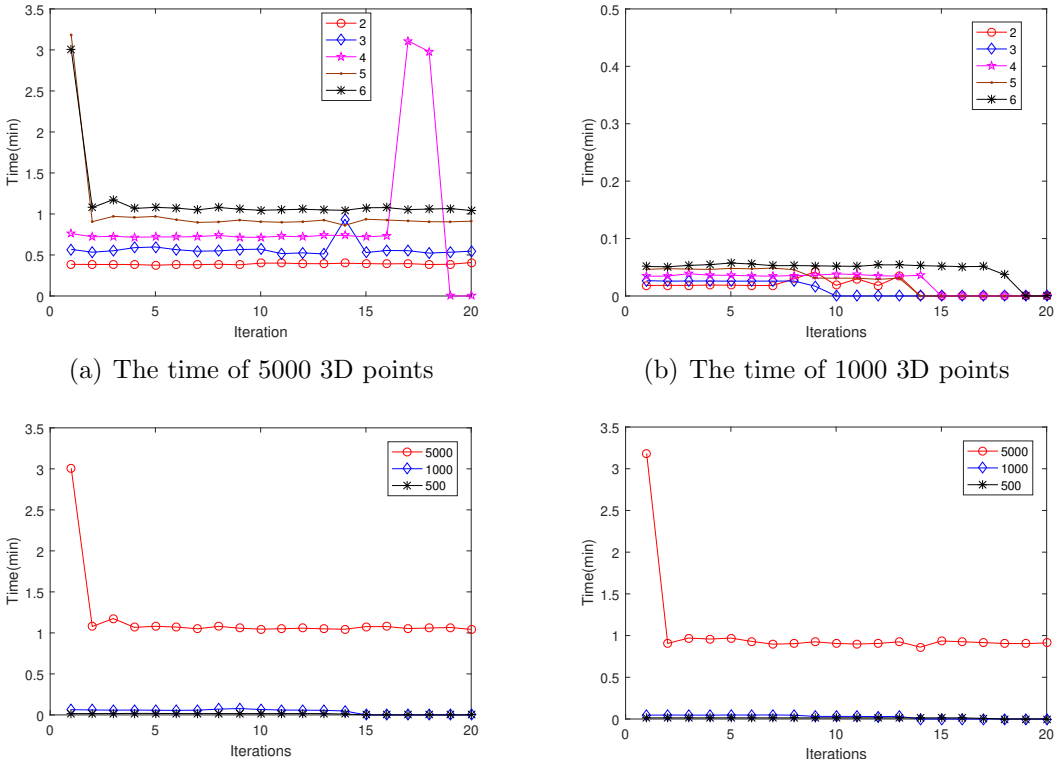
(d) The translation error

Figure 9: Optimization results versus 3D points.

The reprojection error, camera pose error and 3D points position error are shown in Figure 9, which we can see that when the observation of each camera is relatively small(e.g., 6 camera and 10 points), the error is very large. However, with the increase of observation in each camera, the error is getting smaller, and the trend is becoming flatten. Moreover, as is shown in Figure 9 (a) and (b), the behavior of the reprojection error is contrary to the point error as more and more image pairs are included in the simulation.

4.2 Time Complexity and Convergence

This section mainly discusses the effect of the number of cameras and the number of three-dimensional points on the time complexity of the LM algorithm.



(a) The time of 5000 3D points

(b) The time of 1000 3D points

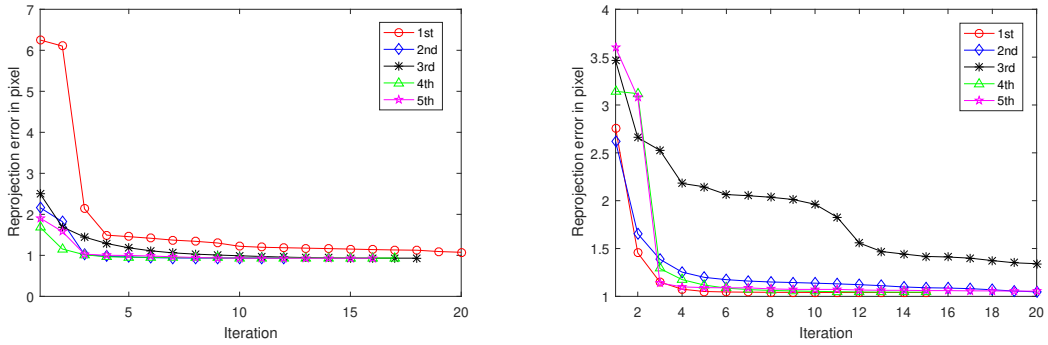
(c) The time of 6 cameras

(d) The time of 5 cameras

Figure 10: The comparison of time complexity.

As can be seen from the comparison in Figure 10, the larger the number of cameras, the greater the time complexity increase. This is because the parameterized camera pose has a dimension of 6(the rotation and the translation is 3 respectively). If there is an extra three-dimensional point, there will be 6 more observations and time complexity Will increase quickly. In addition, if the number of cameras is the same, the increase in the number of three-dimensional points has a relatively small impact on time complexity. Besides, it can be seen from Figure 10 that the fewer cameras, the faster the algorithm converges (time in the Figure 10 is 0). Therefore, the number of cameras also has a relatively large impact on the convergence speed of the algorithm.

This article assumes that all 3D points are visible in the image. As a result, if the number of cameras is 6 and the number of 3D points is 5000, the number of obseravtions is 30000. The LM algorithms are difficult to converge and are very time consuming.



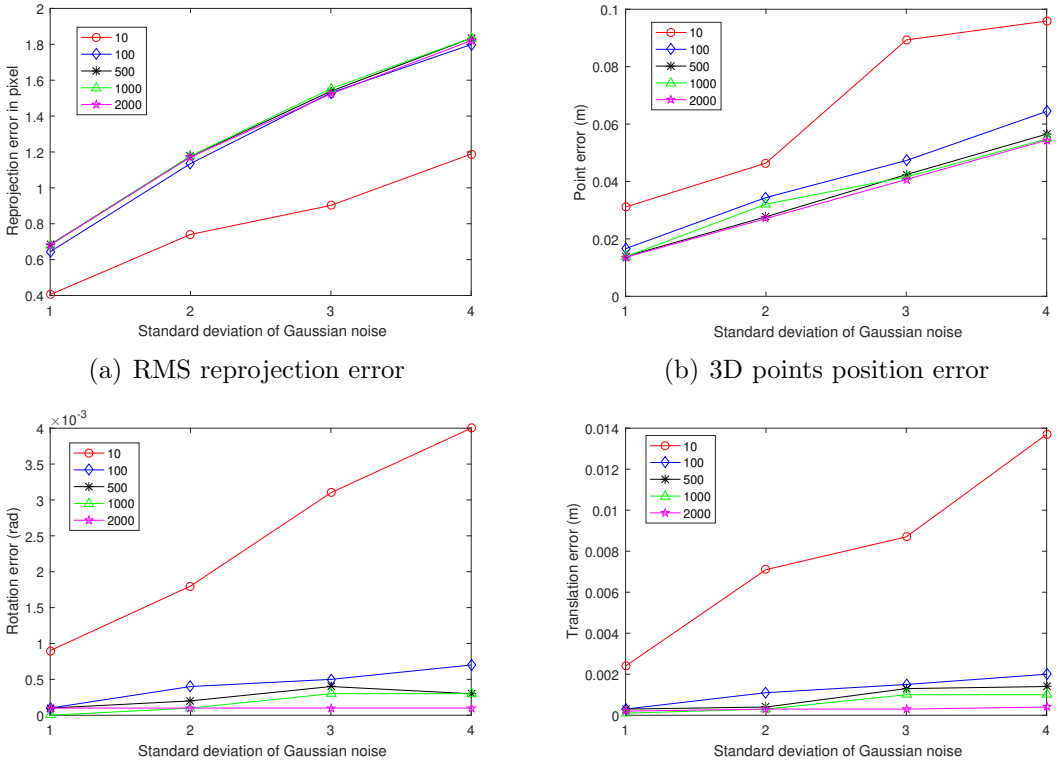
(a) 3 cameras and 2000 points (b) 4 cameras and 2000 points

Figure 11: RMS reprojection error.

Five experiment of the convergence of 3 cameras and 4 cameras(e.g., 2000 points) is shown in Figure 11. As is illustrated in Figure 11, it is almost difficult for 4 cameras to converge within 20 iterations, and the convergence of 3 cameras is better.

4.3 Behvaior with Noise

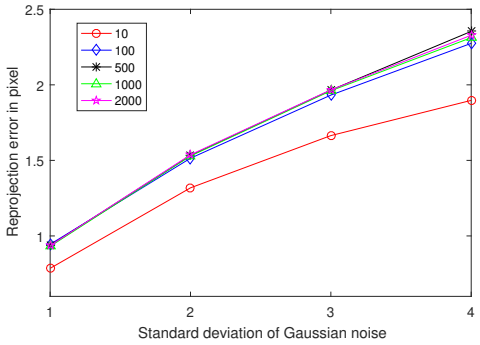
Gaussian noise of different standard deviations(i.e., 1, 2, 3, 4) are added to the feature points. Figure 12-16 shows the RMS reprojection error in pixels, RMSE camera pose error and RMSE points error with noise for different number of cameras. Generally the error is more sensitive to noise, as more and more cameras are considered into the system. Besides, as the noise level increases, the convergence of algorithm is slowing down.



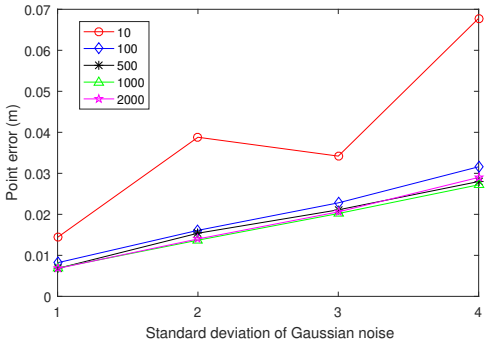
(a) RMS reprojection error (b) 3D points position error

(c) The rotation error (d) The translation error

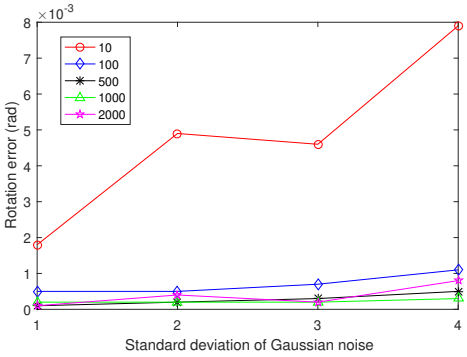
Figure 12: Error for 2 cameras.



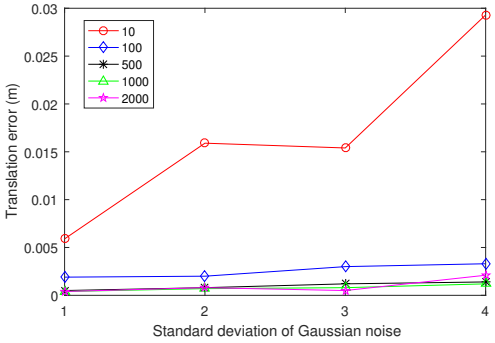
(a) RMS reprojection error



(b) 3D points position error

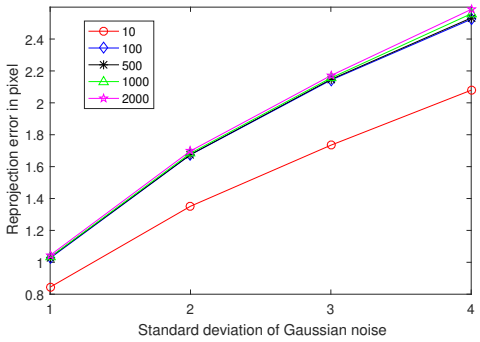


(c) The rotation error

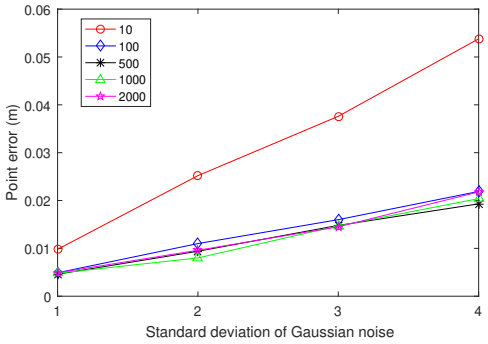


(d) The translation error

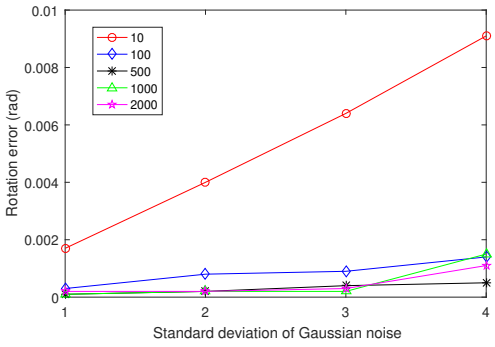
Figure 13: Error for 3 cameras.



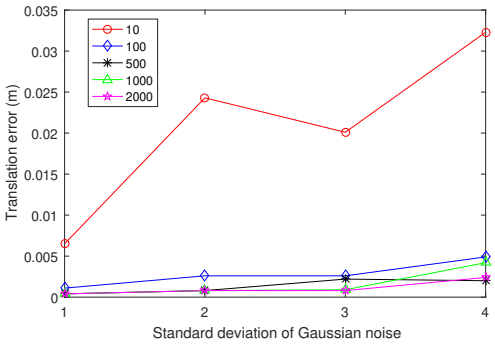
(a) RMS reprojection error



(b) 3D points position error

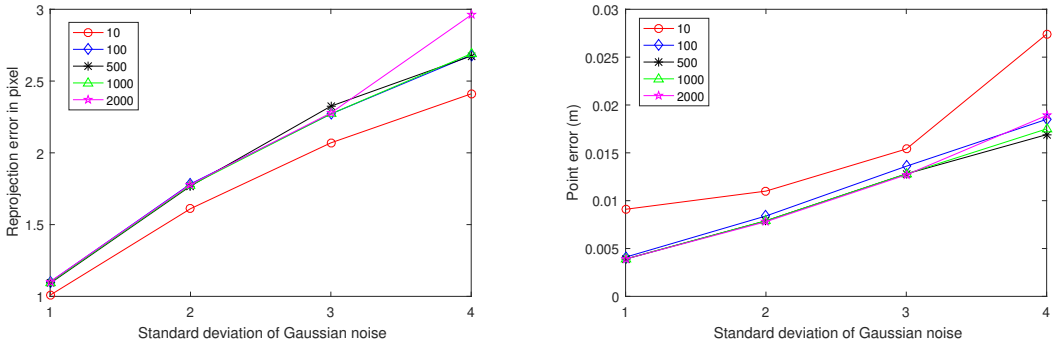


(c) The rotation error



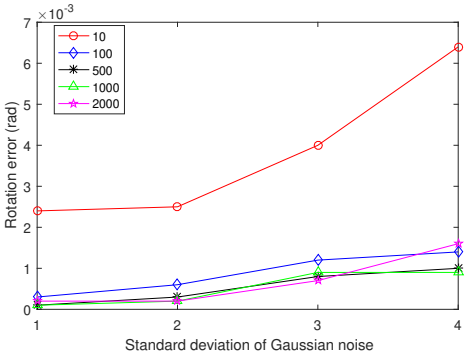
(d) The translation error

Figure 14: Error for 4 cameras.

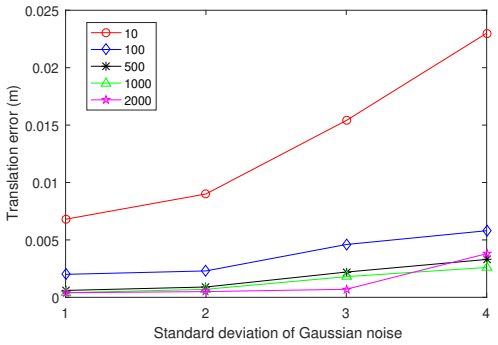


(a) RMS reprojection error

(b) 3D points position error

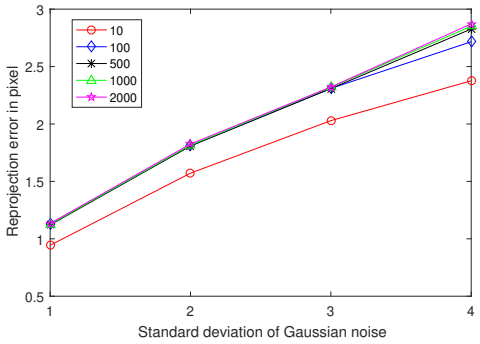


(c) The rotation error

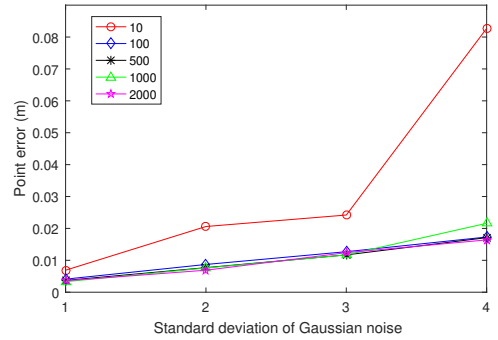


(d) The translation error

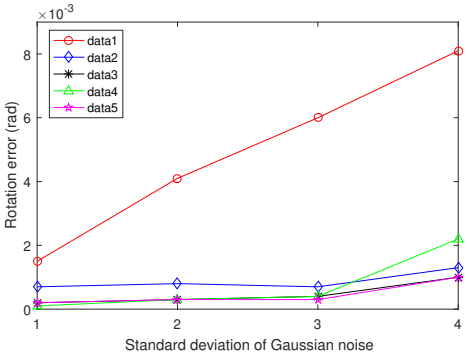
Figure 15: Error for 5 camera.



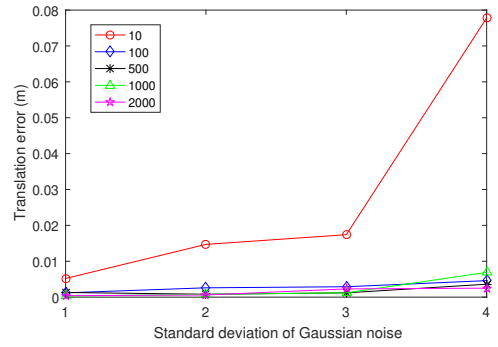
(a) RMS reprojection error



(b) 3D points position error



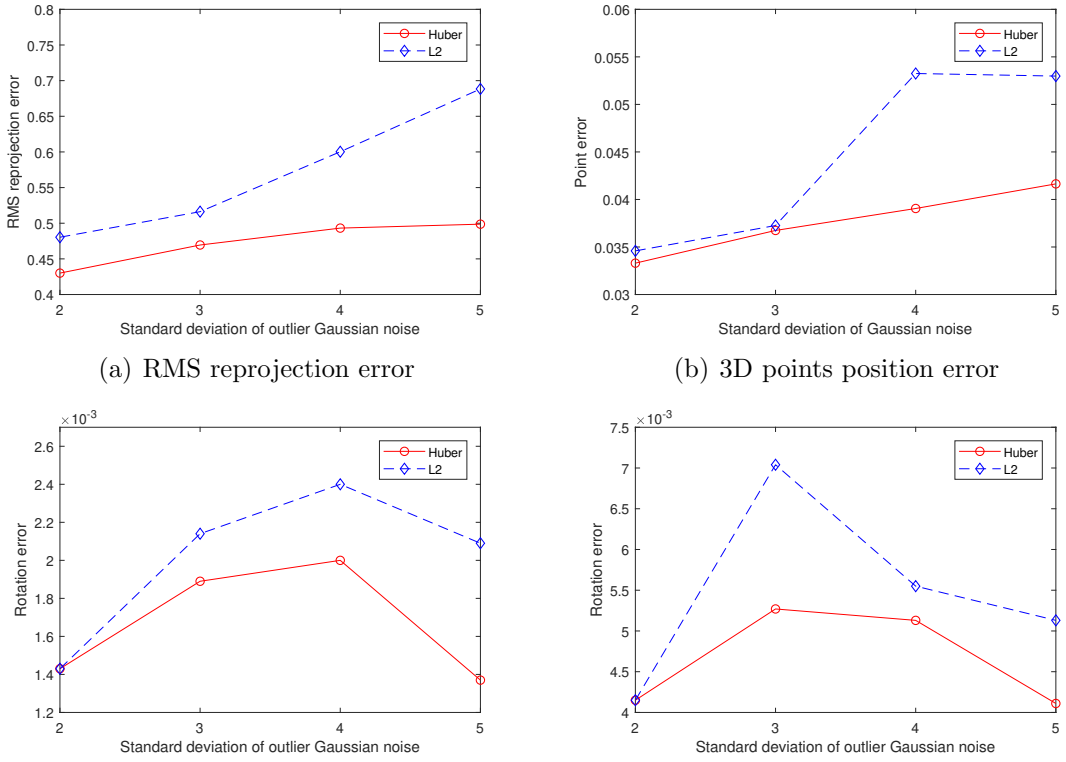
(c) The rotation error



(d) The translation error

Figure 16: Error for 6 cameras.

Moreover, take into outlier image point to the algorithm. Assuming 2 cameras and 10 3D points, Gaussian noise of 1 pixel standard deviation is added to all image pairs except one point pair, which is added to different standard deviation of Gaussian noise(i.e., 2, 3, 4 pixel) and is considered as oulier image pairs. As is illustrated in Figure 17, the sensitivity of L2 and huber residual functions to outlier image points is compared. It is obvious that the huber kernel function is robust to the outliers of the image pairs.



(a) RMS reprojection error

(b) 3D points position error

(c) The rotation error

(d) The translation error

Figure 17: Different outlier Gaussian noise.

4.4 Limits of the Implemented Algorithm

In order to discuss the limits of the algorithm, the threshold for state updates are set to 1e-5 and 1e-16, respectively. The standard deviation of the image points here is 1 pixel. The average error of 10 experiments is shown in Table 1. It is obvious that the smaller update state will improve the accuracy of the algorithm.

Table 1: Mean error for different threshold.

Threshold \ error	reprojection error	point error	rotation error	translation
1e-16	1.0415	0.0047	0.0001	0.0002
1e-5	1.0465	0.005	0.0002	0.0006

5 Personal Summary

This article mainly explains a simple implementation process of bundle adjustment, from recovering 3D points and camera poses to optimizating 3D point positions and camera poses. Simulated numerical experiments used to verify the entire process.

During the process of learning and realizing bundle adjustment, my personal summary is as follows:

(1) BA is essentially a non-linear least squares optimization problem. The more data in theory, the better the model fits. However, if the data used to fit the model has a large error, the more data there is, the more error.

(2) In order to solve the large-scale optimization problem, we do not directly construct the entire Jacobian matrix, but construct a large $\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J}$ matrix. Nonetheless, the replication construction of the matrix $\mathbf{H}$ is also very time-consuming, especially when the Shure complement element is used to solve a linear equation. the matrix $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$ need to be duplicated. The replication of large matrices in Eigen is very time-consuming.

(3) In the optimization process of BA, we take the second norm squared sum of the minimized error terms as the cost function. However, when the data given by an error term is wrong, an impossible error will be added to the optimization process. Since the optimization algorithm cannot distinguish this erroneous data, it will treat all data as error. This will cause a large gradient error in the optimization, which means that adjusting the variables related to it will make the target drop more. Therefore, the algorithm tries to prioritize the variables related to this error term, which will smooth out the impact of other correct error terms. The reason for this problem is that when the error is large, the second norm increases too quickly. Therefore, a kernel function can be used here. The specific way is to replace the second norm metric with a slower growing function, while ensuring its own smoothness. Commonly used robust kernel functions include huber kernel, Cauchy kernel, etc. Huber kernel functions are as follows

$$H(e) = \begin{cases} \frac{1}{2}e^2 & \text{if } |e| < \delta \\ \delta(|e| - \frac{1}{2}\delta) & \text{else} \end{cases} \tag{101}$$

where δ represent a threshold.

(4) For BA problems, whether the initial value is accurate will obviously affect the results of BA. During the experiment, I found that if the triangulated 3D points have a large error, the optimization effect of bundle adjustment on 3D points will have a side effect (the error increases anyway).

(5) The time complexity of bundle adjustment is $O(m^3 + mn)$. Since the parameterized camera pose dimension is 6 and the three-dimensional point parameterized dimension is 3, the increase of the camera has a great impact on the calculation time consumption. In addition, as the number of cameras increases, the algorithm converges more slowly. Therefore, in order to ensure the practicality of the algorithm, we need to control the number of cameras and find a method that can help the LM algorithm to quickly converge.

6 Appendix

A. The Proof of Equation (38)

The eigenvalues of $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ are $\{\lambda_j\}$, and the eigenvectors are $\{v_j\}$. $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ is decomposed to $\mathbf{J}^{\mathrm{T}}\mathbf{J} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathrm{T}}$ through Eigen Decomposition, where $\mathbf{V}$ is orthogonal matrix and $\mathbf{\Lambda}$ diagonal matrix respectively.

$$
\begin{aligned}
& (\mathbf{J}^{\mathrm{T}}\mathbf{J} + \mu\mathbf{I})\Delta\mathbf{x}_{\mathrm{lm}} = -\mathbf{J}^{\mathrm{T}}\mathbf{f} \\
\Rightarrow \quad & (\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathrm{T}} + \mu\mathbf{I})\Delta\mathbf{x}_{\mathrm{lm}} = -\mathbf{F}'^{\mathrm{T}} \\
\Rightarrow \quad & \mathbf{V}(\mathbf{\Lambda} + \mu\mathbf{I})\mathbf{V}^{\mathrm{T}}\Delta\mathbf{x}_{\mathrm{lm}} = -\mathbf{F}'^{\mathrm{T}} \\
\Rightarrow \quad & (\mathbf{\Lambda} + \mu\mathbf{I})\mathbf{V}^{\mathrm{T}}\Delta\mathbf{x}_{\mathrm{lm}} = -\mathbf{V}^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}} \\
\Rightarrow \quad & \mathbf{V}^{\mathrm{T}}\Delta\mathbf{x}_{\mathrm{lm}} = -(\mathbf{\Lambda} + \mu\mathbf{I})^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}} \\
\Rightarrow \quad & \mathbf{V}^{\mathrm{T}}\Delta\mathbf{x}_{\mathrm{lm}} = -\begin{bmatrix} \frac{1}{\lambda_1+\mu} & & & \\ & \frac{1}{\lambda_1+\mu} & & \\ & & \ddots & \\ & & & \frac{1}{\lambda_n+\mu} \end{bmatrix} \mathbf{V}^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}} \\
\Rightarrow \quad & \mathbf{V}^{\mathrm{T}}\Delta\mathbf{x}_{\mathrm{lm}} = -\begin{bmatrix} \frac{\mathbf{v}_1^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}}}{\lambda_1+\mu} & & & \\ & \frac{\mathbf{v}_2^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}}}{\lambda_2+\mu} & & \\ & & \ddots & \\ & & & \frac{\mathbf{v}_n^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}}}{\lambda_n+\mu} \end{bmatrix} \\
\Rightarrow \quad & \Delta\mathbf{x}_{\mathrm{lm}} = -\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} \frac{\mathbf{v}_1^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}}}{\lambda_1+\mu} & & & \\ & \frac{\mathbf{v}_2^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}}}{\lambda_2+\mu} & & \\ & & \ddots & \\ & & & \frac{\mathbf{v}_n^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}}}{\lambda_n+\mu} \end{bmatrix} \\
\Rightarrow \quad & \Delta\mathbf{x}_{\mathrm{lm}} = -\sum_{j=1}^{n} \frac{\mathbf{v}_j^{\mathrm{T}}\mathbf{F}'^{\mathrm{T}}}{\lambda_j + \mu}\mathbf{v}_j
\end{aligned}
$$

B. The Structure of Hessian

- $\mathbf{H}_1$

$$
\mathbf{J}_1 = \frac{\partial \mathbf{e}_{11}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}
$$

$$
\mathbf{H}_1 = \mathbf{J}_1\Sigma_1^{-1}\mathbf{J}_1 = \begin{pmatrix} (\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1})^{\mathrm{T}}\Sigma_1^{-1}\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1})^{\mathrm{T}}\Sigma_1^{-1}\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1})^{\mathrm{T}}\Sigma_1^{-1}\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1})^{\mathrm{T}}\Sigma_1^{-1}\frac{\partial \mathbf{e}_{11}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}
$$

- $\mathbf{H}_2$

$$\mathbf{J}_2 = \frac{\partial \mathbf{e}_{21}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \mathbf{0} & \frac{\partial \mathbf{e}_{21}}{\partial \mathbf{a}_2} & \mathbf{0} & \frac{\partial \mathbf{e}_{21}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

$$\mathbf{H}_2 = \mathbf{J}_2 \Sigma_2^{-1} \mathbf{J}_2 = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\partial \mathbf{e}_{21}}{\partial \mathbf{a}_2})^{\mathrm{T}} \Sigma_2^{-1} \frac{\partial \mathbf{e}_{21}}{\partial \mathbf{a}_1} & \mathbf{0} & (\frac{\partial \mathbf{e}_{21}}{\partial \mathbf{a}_2})^{\mathrm{T}} \Sigma_2^{-1} \frac{\partial \mathbf{e}_{21}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\partial \mathbf{e}_{21}}{\partial \mathbf{b}_1})^{\mathrm{T}} \Sigma_2^{-1} \frac{\partial \mathbf{e}_{21}}{\partial \mathbf{a}_2} & \mathbf{0} & (\frac{\partial \mathbf{e}_{21}}{\partial \mathbf{b}_1})^{\mathrm{T}} \Sigma_2^{-1} \frac{\partial \mathbf{e}_{21}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

- $\mathbf{H}_3$

$$\mathbf{J}_3 = \frac{\partial \mathbf{e}_{31}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{31}}{\partial \mathbf{a}_3} & \frac{\partial \mathbf{e}_{31}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

$$\mathbf{H}_3 = \mathbf{J}_3 \Sigma_3^{-1} \mathbf{J}_3 = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{31}}{\partial \mathbf{a}_3})^{\mathrm{T}} \Sigma_3^{-1} \frac{\partial \mathbf{e}_{31}}{\partial \mathbf{a}_3} & (\frac{\partial \mathbf{e}_{31}}{\partial \mathbf{a}_3})^{\mathrm{T}} \Sigma_3^{-1} \frac{\partial \mathbf{e}_{31}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{31}}{\partial \mathbf{b}_1})^{\mathrm{T}} \Sigma_3^{-1} \frac{\partial \mathbf{e}_{31}}{\partial \mathbf{a}_3} & (\frac{\partial \mathbf{e}_{31}}{\partial \mathbf{b}_1})^{\mathrm{T}} \Sigma_3^{-1} \frac{\partial \mathbf{e}_{31}}{\partial \mathbf{b}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

- $\mathbf{H}_4$

$$\mathbf{J}_4 = \frac{\partial \mathbf{e}_{12}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \frac{\partial \mathbf{e}_{12}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{12}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

$$\mathbf{H}_4 = \mathbf{J}_4 \Sigma_4^{-1} \mathbf{J}_4 = \begin{pmatrix} (\frac{\partial \mathbf{e}_{12}}{\partial \mathbf{a}_1})^{\mathrm{T}} \Sigma_4^{-1} \frac{\partial \mathbf{e}_{12}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{12}}{\partial \mathbf{a}_1})^{\mathrm{T}} \Sigma_4^{-1} \frac{\partial \mathbf{e}_{12}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\frac{\partial \mathbf{e}_{12}}{\partial \mathbf{b}_2})^{\mathrm{T}} \Sigma_4^{-1} \frac{\partial \mathbf{e}_{12}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{12}}{\partial \mathbf{b}_2})^{\mathrm{T}} \Sigma_4^{-1} \frac{\partial \mathbf{e}_{12}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

- $\mathbf{H}_5$

$$\mathbf{J}_5 = \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \mathbf{0} & \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

$$\mathbf{H}_5 = \mathbf{J}_5 \Sigma_5^{-1} \mathbf{J}_5 = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\partial \mathbf{e}_{22}}{\partial \mathbf{a}_2})^{\mathrm{T}} \Sigma_5^{-1} \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{22}}{\partial \mathbf{a}_2})^{\mathrm{T}} \Sigma_5^{-1} \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\partial \mathbf{e}_{22}}{\partial \mathbf{b}_2})^{\mathrm{T}} \Sigma_5^{-1} \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{22}}{\partial \mathbf{b}_2})^{\mathrm{T}} \Sigma_5^{-1} \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

- $\mathbf{H}_6$

$$\mathbf{J}_6 = \frac{\partial \mathbf{e}_{32}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{32}}{\partial \mathbf{a}_3} & \mathbf{0} & \frac{\partial \mathbf{e}_{22}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

$$\mathbf{H}_6 = \mathbf{J}_6 \Sigma_6^{-1} \mathbf{J}_6 = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{32}}{\partial \mathbf{a}_3})^{\mathrm{T}} \Sigma_6^{-1} \frac{\partial \mathbf{e}_{32}}{\partial \mathbf{a}_3} & \mathbf{0} & (\frac{\partial \mathbf{e}_{32}}{\partial \mathbf{a}_3})^{\mathrm{T}} \Sigma_6^{-1} \frac{\partial \mathbf{e}_{32}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{32}}{\partial \mathbf{b}_2})^{\mathrm{T}} \Sigma_6^{-1} \frac{\partial \mathbf{e}_{32}}{\partial \mathbf{a}_3} & \mathbf{0} & (\frac{\partial \mathbf{e}_{32}}{\partial \mathbf{b}_2})^{\mathrm{T}} \Sigma_6^{-1} \frac{\partial \mathbf{e}_{32}}{\partial \mathbf{b}_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

- $\mathbf{H}_7$

$$\mathbf{J}_7 = \frac{\partial \mathbf{e}_{13}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \frac{\partial \mathbf{e}_{13}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{13}}{\partial \mathbf{b}_3} & \mathbf{0} \end{pmatrix}$$

$$\mathbf{H}_7 = \mathbf{J}_7 \Sigma_7^{-1} \mathbf{J}_7 = \begin{pmatrix} (\frac{\partial \mathbf{e}_{13}}{\partial \mathbf{a}_1})^{\mathrm{T}} \Sigma_7^{-1} \frac{\partial \mathbf{e}_{13}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{13}}{\partial \mathbf{a}_1})^{\mathrm{T}} \Sigma_7^{-1} \frac{\partial \mathbf{e}_{13}}{\partial \mathbf{b}_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\frac{\partial \mathbf{e}_{13}}{\partial \mathbf{b}_3})^{\mathrm{T}} \Sigma_7^{-1} \frac{\partial \mathbf{e}_{13}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{13}}{\partial \mathbf{b}_3})^{\mathrm{T}} \Sigma_7^{-1} \frac{\partial \mathbf{e}_{13}}{\partial \mathbf{b}_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

- $\mathbf{H}_8$

$$\mathbf{J}_8 = \frac{\partial \mathbf{e}_{23}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \mathbf{0} & \frac{\partial \mathbf{e}_{23}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{23}}{\partial \mathbf{b}_3} & \mathbf{0} \end{pmatrix}$$

$$\mathbf{H}_8 = \mathbf{J}_8 \Sigma_8^{-1} \mathbf{J}_8 = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\partial \mathbf{e}_{23}}{\partial \mathbf{a}_2})^{\mathrm{T}} \Sigma_8^{-1} \frac{\partial \mathbf{e}_{23}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{23}}{\partial \mathbf{a}_2})^{\mathrm{T}} \Sigma_8^{-1} \frac{\partial \mathbf{e}_{23}}{\partial \mathbf{b}_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\partial \mathbf{e}_{23}}{\partial \mathbf{b}_3})^{\mathrm{T}} \Sigma_8^{-1} \frac{\partial \mathbf{e}_{23}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{23}}{\partial \mathbf{b}_3})^{\mathrm{T}} \Sigma_8^{-1} \frac{\partial \mathbf{e}_{23}}{\partial \mathbf{b}_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

- $\mathbf{H}_9$

$$\mathbf{J}_9 = \frac{\partial \mathbf{e}_{33}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{33}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{33}}{\partial \mathbf{b}_3} & \mathbf{0} \end{pmatrix}$$

$$\mathbf{H}_9 = \mathbf{J}_9 \Sigma_9^{-1} \mathbf{J}_9 = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{33}}{\partial \mathbf{a}_3})^{\mathrm{T}} \Sigma_9^{-1} \frac{\partial \mathbf{e}_{33}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{33}}{\partial \mathbf{a}_3})^{\mathrm{T}} \Sigma_9^{-1} \frac{\partial \mathbf{e}_{33}}{\partial \mathbf{b}_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{33}}{\partial \mathbf{b}_3})^{\mathrm{T}} \Sigma_9^{-1} \frac{\partial \mathbf{e}_{33}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{33}}{\partial \mathbf{b}_3})^{\mathrm{T}} \Sigma_9^{-1} \frac{\partial \mathbf{e}_{33}}{\partial \mathbf{b}_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

- $\mathbf{H}_{10}$

$$\mathbf{J}_{10} = \frac{\partial \mathbf{e}_{14}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \frac{\partial \mathbf{e}_{14}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{14}}{\partial \mathbf{b}_2} \end{pmatrix}$$

$$\mathbf{H}_{10} = \mathbf{J}_{10} \Sigma_{10}^{-1} \mathbf{J}_{10} = \begin{pmatrix} (\frac{\partial \mathbf{e}_{14}}{\partial \mathbf{a}_1})^{\mathrm{T}} \Sigma_{10}^{-1} \frac{\partial \mathbf{e}_{14}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{14}}{\partial \mathbf{a}_1})^{\mathrm{T}} \Sigma_{10}^{-1} \frac{\partial \mathbf{e}_{14}}{\partial \mathbf{b}_4} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\frac{\partial \mathbf{e}_{14}}{\partial \mathbf{b}_4})^{\mathrm{T}} \Sigma_{10}^{-1} \frac{\partial \mathbf{e}_{14}}{\partial \mathbf{a}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{14}}{\partial \mathbf{b}_4})^{\mathrm{T}} \Sigma_{10}^{-1} \frac{\partial \mathbf{e}_{14}}{\partial \mathbf{b}_4} \end{pmatrix}$$

- $\mathbf{H}_{11}$

$$\mathbf{J}_{11} = \frac{\partial \mathbf{e}_{24}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \mathbf{0} & \frac{\partial \mathbf{e}_{24}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{24}}{\partial \mathbf{b}_3} \end{pmatrix}$$

$$\mathbf{H}_{11} = \mathbf{J}_{11} \Sigma_{11}^{-1} \mathbf{J}_{11} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\partial \mathbf{e}_{24}}{\partial \mathbf{a}_2})^{\mathrm{T}} \Sigma_{11}^{-1} \frac{\partial \mathbf{e}_{24}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{24}}{\partial \mathbf{a}_2})^{\mathrm{T}} \Sigma_{11}^{-1} \frac{\partial \mathbf{e}_{24}}{\partial \mathbf{b}_4} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\partial \mathbf{e}_{24}}{\partial \mathbf{b}_4})^{\mathrm{T}} \Sigma_{11}^{-1} \frac{\partial \mathbf{e}_{24}}{\partial \mathbf{a}_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{24}}{\partial \mathbf{b}_4})^{\mathrm{T}} \Sigma_{11}^{-1} \frac{\partial \mathbf{e}_{24}}{\partial \mathbf{b}_4} \end{pmatrix}$$

- $\mathbf{H}_{12}$

$$\mathbf{J}_{12} = \frac{\partial \mathbf{e}_{34}}{\partial \mathbf{\Delta x}} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{34}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_{34}}{\partial \mathbf{b}_4} \end{pmatrix}$$

$$\mathbf{H}_{12} = \mathbf{J}_{12} \Sigma_{12}^{-1} \mathbf{J}_{12} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{34}}{\partial \mathbf{a}_3})^{\mathrm{T}} \Sigma_{12}^{-1} \frac{\partial \mathbf{e}_{34}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{34}}{\partial \mathbf{a}_3})^{\mathrm{T}} \Sigma_{12}^{-1} \frac{\partial \mathbf{e}_{34}}{\partial \mathbf{b}_4} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{34}}{\partial \mathbf{b}_4})^{\mathrm{T}} \Sigma_{12}^{-1} \frac{\partial \mathbf{e}_{34}}{\partial \mathbf{a}_3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & (\frac{\partial \mathbf{e}_{34}}{\partial \mathbf{b}_4})^{\mathrm{T}} \Sigma_{12}^{-1} \frac{\partial \mathbf{e}_{34}}{\partial \mathbf{b}_4} \end{pmatrix}$$

C. The Schur Complement

Given an arbitrary matrix block M, as shown below

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}.$$

There are some conclusions:

- if $\mathbf{D}$ is invertible matrix, $\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}$ is the Schur complement of $\mathbf{D}$;

- if $\mathbf{A}$ is invertible matrix, $\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ is the Schur complement of $\mathbf{A}$.

In the process of turning the matrix $\mathbf{M}$ into an upper triangle or a lower triangle, we can use Shure complement

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}$$
$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{C} & \mathbf{S} \end{pmatrix}$$

where $\mathbf{S}=\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}$. Also, we can transform $\mathbf{M}$ into a diagonal matrix:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix},$$

and recover $\mathbf{M}$ from the diagonal matirx:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

D. The Algebraic Derivation of F and E

Assuming that a 3D point $P_w = \begin{pmatrix} X_\mathrm{w} & Y_\mathrm{w} & Z_\mathrm{w} \end{pmatrix}^\mathrm{T}$, the location of the corresponding pixel $\mathbf{p}_1$, $\mathbf{p}_2$ is

$$\lambda_1 \mathbf{p}_1 = \mathbf{K}\mathbf{P}_\mathrm{w}, \quad \lambda_2 \mathbf{p}_2 = \mathbf{K}(\mathbf{R}\mathbf{P}_\mathrm{w} + \mathbf{t}).$$

The equation is equal up to a scale, and can be rewritten as

$$\mathbf{p}_1 \simeq \mathbf{K}\mathbf{P}_\mathrm{w}, \quad \mathbf{p}_2 \simeq \mathbf{K}(\mathbf{R}\mathbf{P}_\mathrm{w} + \mathbf{t}).$$

Next, let $\mathbf{x} = \mathbf{K}^{-1}\mathbf{p}_1$ and $\mathbf{x}_2 = \mathbf{K}^{-1}\mathbf{p}_2$ be the normalized point. Thus

$$\mathbf{x}_2 \simeq \mathbf{R}\mathbf{x}_1 + \mathbf{t}.$$

Left multiplication the equation by $\mathbf{t}^\wedge$ results in

$$\mathbf{t}^\wedge \mathbf{x}_2 \simeq \mathbf{t}^\wedge \mathbf{R}\mathbf{x}_1.$$

Then, Left multiplication both sides by $\mathbf{x}_2^\mathrm{T}$ results in

$$\mathbf{x}_2^\mathrm{T}\mathbf{t}^\wedge \mathbf{x}_2 \simeq \mathbf{x}_2^\mathrm{T}\mathbf{t}^\wedge \mathbf{R}\mathbf{x}_1.$$

where $\mathbf{t}^\wedge \mathbf{x}_2$ is a vector perpendicular to both $\mathbf{t}$ and $\mathbf{x}_2$. Therefore, its inner product with $\mathbf{x}_2$ is equal to 0, and $\simeq$ can written as $=$. Then

$$\mathbf{x}_2^\mathrm{T}\mathbf{t}^\wedge \mathbf{R}\mathbf{x}_1 = 0$$

Replacing $\mathbf{x}_1$, $\mathbf{x}_2$ with $\mathbf{p}_1$, $\mathbf{p}_2$, the equation is now becoming

$$\mathbf{p}_2^\mathrm{T}\mathbf{K}^{-\mathrm{T}}\mathbf{t}^\wedge \mathbf{R}\mathbf{K}^{-1}\mathbf{p}_1 = 0.$$

This equation is so-called epipolar constraint. Let $\mathbf{E} = \mathbf{t}^\wedge \mathbf{R}$ and $\mathbf{F} = \mathbf{K}^{-\mathrm{T}}\mathbf{t}^\wedge \mathbf{R}\mathbf{K}^{-1}$, thus

$$\mathbf{x}_2^\mathrm{T}\mathbf{E}\mathbf{x}_1 = \mathbf{p}_2^\mathrm{T}\mathbf{K}^{-\mathrm{T}}\mathbf{t}^\wedge \mathbf{R}\mathbf{K}^{-1}\mathbf{p}_1 = 0$$

E. The Proof of the $\mathbf{P}_{\mathrm{w}} = \mathbf{d}_4$

For solving the triangulation problem using the least square method

$$\min_{\mathbf{P}_{\mathrm{w}}} ||\mathbf{A}\mathbf{P}_{\mathrm{w}}||^2, \quad \text{s.t. } ||\mathbf{P}_{\mathrm{w}}|| = 1$$

where $A \in \mathbb{R}^{2m\times 4}$, and m is the number of observations.

Step1: Make SVD of $\mathbf{A}$: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$, and thus $\min||\mathbf{A}\mathbf{P}_{\mathrm{w}}||^2 = \min||\mathbf{P}_{\mathrm{w}}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{P}_{\mathrm{w}}||$.

Step2: Compute the matrix

$$\begin{aligned}\mathbf{A}^{\mathrm{T}}\mathbf{A} &= \mathbf{V}\mathbf{\Sigma}^{\mathrm{T}}\mathbf{U}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} \\ &= \mathbf{V}\begin{pmatrix}\sigma_1^2 & & \\ & \ddots & \\ & & \sigma_4^2\end{pmatrix}\mathbf{V}^{\mathrm{T}} \\ &= \sum_{i=1}^{4}\sigma_i\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}\end{aligned}$$

where $\mathbf{v}_i$ is ith column of $\mathbf{V}$, and $\sigma_1 \geq \ldots \geq \sigma_4$. If $\sigma_3 = \sigma_4$, the equations have no solution.

Step3: Since $\mathbf{V}$ is an orthogonal matrix, that is, $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$, $\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j$ is equal to zero.

Step4: Since this is a homogeneous linear equations, if $\mathbf{P}_w$ is the solution of the equations, $k\mathbf{y}$ is also the solution. Let $\mathbf{y} = \sum_1^4 k_i\mathbf{v}_i$, and $\mathbf{v}_i$ is the base vector. Thus,

$$\begin{aligned}\mathbf{P}_{\mathrm{w}}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{P}_{\mathrm{w}} &= \sigma_1^2k_1^2\mathbf{v}_1^{\mathrm{T}}\mathbf{v}_1\mathbf{v}_1^{\mathrm{T}}\mathbf{v}_1 + \ldots + \sigma_4^2k_4^2\mathbf{v}_4^{\mathrm{T}}\mathbf{v}_4\mathbf{v}_4^{\mathrm{T}}\mathbf{v}_4 \\ &= \sigma_1^2k_1^2 + \sigma_2^2k_2^2 + \sigma_3^2k_3^2 + \sigma_4^2k_4^2.\end{aligned}$$

Since $||\mathbf{P}_{rmw}|| = 1$, $\sum k_i = 1$.

Step5: σ_4 is the smallest singular Value. Thus, when $k_1 = k_2 = k_3 = 0$ and $k_4 = 1$, the above formula gets the minimum value. At this time, $\mathbf{P}_{\mathrm{w}} = \mathbf{v}_4$ is the column vector corresponding to the minimum eigenvalue of the matrix $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

F. Derivative of Lie Algebra of SE(3)

$$\begin{aligned}\frac{\partial\mathbf{P}_c}{\partial\delta\mathbf{a}} = \frac{\partial(\mathbf{T}\mathbf{P}_{\mathrm{w}})}{\partial\delta\mathbf{a}} &= \lim_{\delta\mathbf{a}\to\mathbf{0}}\frac{\exp(\delta\mathbf{a}^\wedge)\exp(\mathbf{a}^\wedge)\mathbf{P}_{\mathrm{w}} - \exp(\mathbf{a}^\wedge)\mathbf{P}_{\mathrm{w}}}{\delta\mathbf{a}} \\ &= \lim_{\delta\mathbf{a}\to\mathbf{0}}\frac{(\mathbf{I}+\delta\mathbf{a}^\wedge)\exp(\mathbf{a}^\wedge)\mathbf{P}_{\mathrm{w}} - \exp(\mathbf{a}^\wedge)\mathbf{P}_{\mathrm{w}}}{\delta\mathbf{a}} \\ &= \lim_{\delta\mathbf{a}\to\mathbf{0}}\frac{\delta\mathbf{a}^\wedge\exp(\mathbf{a}^\wedge)\mathbf{P}_{\mathrm{w}}}{\delta\mathbf{a}} \\ &= \lim_{\delta\mathbf{a}\to\mathbf{0}}\frac{\begin{pmatrix}\delta\phi^\wedge & \delta\rho \\ \mathbf{0}^{\mathrm{T}} & 0\end{pmatrix}\begin{pmatrix}\mathbf{R}\mathbf{P}_{\mathrm{w}}+\mathbf{t} \\ 1\end{pmatrix}}{\delta\mathbf{a}} \\ &= \lim_{\delta\mathbf{a}\to\mathbf{0}}\frac{\begin{pmatrix}\delta\phi^\wedge(\mathbf{R}\mathbf{P}_{\mathrm{w}}+\mathbf{t})+\delta\rho \\ 1\end{pmatrix}}{\begin{pmatrix}\delta\rho & \delta\phi^\wedge\end{pmatrix}^{\mathrm{T}}} \\ &= \lim_{\delta\mathbf{a}\to\mathbf{0}}\begin{pmatrix}\frac{\partial(\delta\phi^\wedge(\mathbf{R}\mathbf{P}_{\mathrm{w}}+\mathbf{t})+\delta\rho)}{\partial\delta\rho} & \frac{\partial(\delta\phi^\wedge(\mathbf{R}\mathbf{P}_{\mathrm{w}}+\mathbf{t})+\delta\rho)}{\partial\delta\phi^\wedge} \\ \frac{\partial(1)}{\partial\rho} & \frac{\partial(1)}{\partial\delta\phi^\wedge}\end{pmatrix} \\ &= \begin{pmatrix}\mathbf{I} & -(\mathbf{R}\mathbf{P}_{\mathrm{w}}+\mathbf{t})^\wedge \\ \mathbf{0}^{\mathrm{T}} & 0\end{pmatrix} = \begin{pmatrix}\mathbf{I} & -\mathbf{P}_c^\wedge \\ \mathbf{0}^{\mathrm{T}} & 0\end{pmatrix}\end{aligned}$$