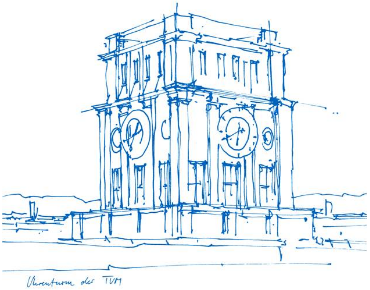
Computer Vision II: Multiple View Geometry (IN2228)

Chapter 12 Bundle Adjustment

Dr. Haoang Li

06 July 2023 11:00-11:45



Announcement Before Class

➢ Updated Lecture Schedule

For updates, slides, and additional materials:
https://cvg.cit.tum.de/teaching/ss2023/cv2

90-minute course; 45-minute course

Foundation

Wed 19.04.2023 Chapter 00: Introduction
Thu 20.04.2023 Chapter 01: Mathematical Backgrounds

Wed 26.04.2023 Chapter 02: Motion and Scene Representation (Part 1)
Thu 27.04.2023 Chapter 02: Motion and Scene Representation (Part 2)

Wed 03.05.2023 Chapter 03: Image Formation (Part 1)
Thu 04.05.2023 Chapter 03: Image Formation (Part 2)

Wed 10.05.2023 Chapter 04: Camera Calibration
Thu 11.05.2023 Chapter 05: Correspondence Estimation (Part 1)

Wed 17.05.2023 Chapter 05: Correspondence Estimation (Part 2)
Thu 18.05.2023 No lecture (Public Holiday)

Wed 24.05.2023 No lecture (Conference)
Thu 25.05.2023 No lecture (Conference)

Videos and reading materials about the combination of deep learning and multi-view geometry

Wed 31.05.2023 Chapter 05: Correspondence Estimation (Part 3)

Core part

Thu 01.06.2023 Chapter 06: 2D-2D Geometry (Part 1)

Wed 07.06.2023 Chapter 06: 2D-2D Geometry (Part 2)
Thu 08.06.2023 No lecture (Public Holiday)

Wed 14.06.2023 Chapter 06: 2D-2D Geometry (Part 3)
Thu 15.06.2023 Chapter 06: 2D-2D Geometry (Part 4)

Wed 21.06.2023 Chapter 07: 3D-2D Geometry
Thu 22.06.2023 Chapter 08: 3D-3D Geometry

Wed 28.06.2023 Chapter 09: Single-view Geometry
Thu 29.06.2023 Chapter 10: Combination of Different Configurations

Advanced topics and high-level tasks

Wed 05.07.2023 Chapter 11: Photometric Error and Direct Method
Thu 06.07.2023 Chapter 12: Bundle Adjustment

Wed 12.07.2023 Chapter 13: Robust Estimation
Thu 13.07.2023 Exam Information and Knowledge Review

Wed 19.07.2023 Chapter 14: SLAM and SFM
Thu 20.07.2023 No Onsite Lecture. Alternative: Online Meeting for Question Answering

Today’s Outline

- Error Metrics
- Definition of Bundle Adjustment
- Basic Knowledge of Non-linear Optimization

- Application of Non-linear Optimization to Bundle Adjustment Based on Lie Algebra (skipped due to limited time)

Error Metrics

➢ Overview

✓ The quality of the estimated camera pose can be measured using different error metrics:

- Algebraic error
- Epipolar Line Distance (only for 2D-2D)
- Reprojection Error

✓ By minimizing any of the above error, we can optimize the camera pose.

✓ The above metrics are not limited to 2D-2D. We can also use them to evaluate 3D-2D case. In our class, let us take 2D-2D for example.

Error Metrics

➢ Algebraic Error

✓ We consider 8-point algorithm for illustration. It seeks to minimize the **algebraic error:**

$$err = \|QE\|^2 = \sum_{i=1}^{N} (\bar{p}^{i\,\mathrm{T}}_2 \boldsymbol{E} \bar{p}^{i}_1)^2$$

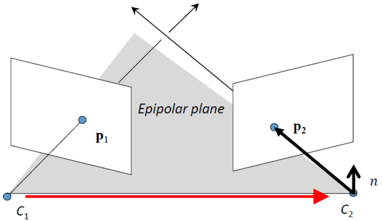
✓ From the derivation of the epipolar constraint and the property of dot product, we can observe:

$$\|\bar{\boldsymbol{p}}_2^{\mathrm{T}} \boldsymbol{E} \bar{\boldsymbol{p}}_1\| = \|\bar{\boldsymbol{p}}_2^{\mathrm{T}} \cdot (\boldsymbol{E} \bar{\boldsymbol{p}}_1)\| = \|\bar{\boldsymbol{p}}_2\| \|\boldsymbol{E} \bar{\boldsymbol{p}}_1\| \cos(\theta)$$

Associative law   Property of dot product

$$= \|\bar{\boldsymbol{p}}_2\| \| [\mathrm{T}_\times] R\, \bar{\boldsymbol{p}}_1 \| \cos(\theta)$$

Definition of essential matrix (in right camera frame)



Error Metrics

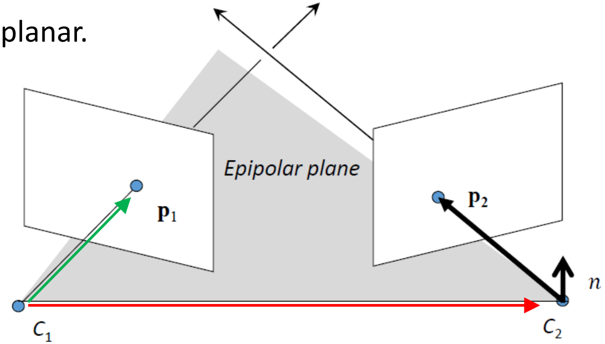
$$= \|\bar{\boldsymbol{p}}_2\| \, \| \boxed{\boxed{[\mathrm{T}_\times]}\boxed{R\, \bar{\boldsymbol{p}}_1}} \| \cos(\theta)$$

Normal in the right camera frame

- ➢ Algebraic Error

✓ We can see that this product depends on the angle θ between $\bar{\boldsymbol{p}}_2$ and the normal to the epipolar plane.

✓ It is nonzero when $\bar{\boldsymbol{p}}_1, \bar{\boldsymbol{p}}_2$, and $\boldsymbol{T}$ are not coplanar.



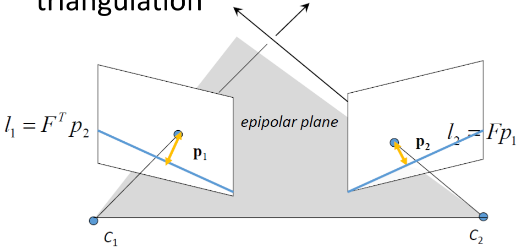
Error Metrics

- ➢ Epipolar Line Distance (only for 2D-2D configuration)

✓ Sum of squared epipolar-line-to-point distances:

$$err = \sum_{i=1}^{N} \left(d(p_1^i, l_1^i)\right)^2 + \left(d(p_2^i, l_2^i)\right)^2$$

✓ Cheaper than reprojection error (introduced later) because does not require point triangulation



Right point

$$\begin{bmatrix} u_2^i \\ v_2^i \\ 1 \end{bmatrix}^T \boxed{F} \begin{bmatrix} u_1^i \\ v_1^i \\ 1 \end{bmatrix} = 0$$

Fundamental Matrix $F = K_2^{-T} \, E \, K_1^{-1}$

Point lies on a line: dot(p, l)=0

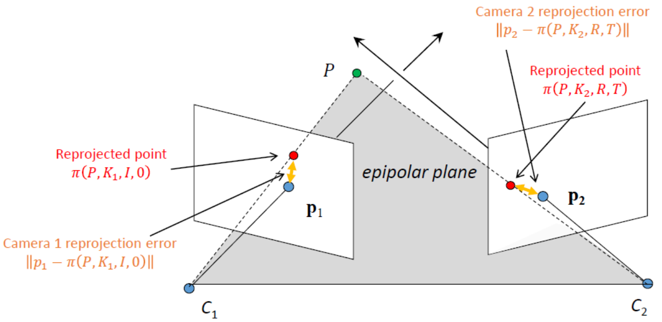
Epipolar line computed by the left point

Error Metrics

➢ Reprojection Error

✓ Sum of the Squared Reprojection Errors $err = \sum_{i=1}^{N} \|p_1^i - \pi(P^i, K_1, I, 0)\|^2 + \|p_2^i - \pi(P^i, K_2, R, T)\|^2$

✓ More expensive than the previous errors because it requires to first **triangulate** the 3D points.

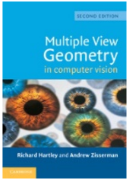


Error Metrics

➢ Reprojection Error

- ✓ It is the most popular because more accurate. The reason is that the error is computed directly with respect to the original input data, i.e., the image points. It is point-to-point distance.
- ✓ Previous algebraic error is with respect to 3D direction; Epipolar line distance is a point-to-line distance.
- ✓ Reprojection error is commonly called "golden standard" in our society. For a systematic analysis, please refer to [1].

[1] "Multiple View Geometry in Computer Vision": R. Hartley and A. Zisserman
Link: https://www.robots.ox.ac.uk/~vgg/hzbook/

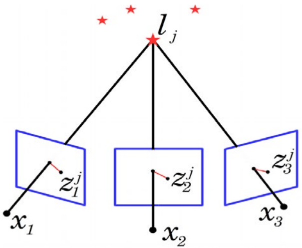


Error Metrics

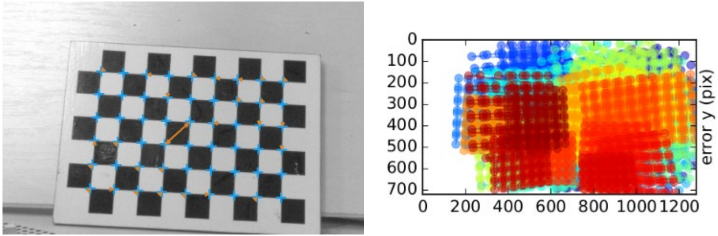
➢ Reprojection Error

✓ We often use reprojection error to perform two tasks:

- Pose and 3D point optimization
- Accuracy evaluation



Bundle adjustment for optimization



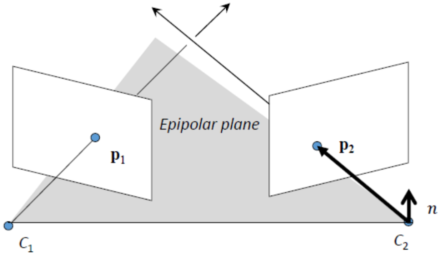
Calibration evaluation

Error Metrics

- Error Minimization

✓ Let us consider 8-point method. For **more than 8 points**, error will only be 0 if there is **no noise** in the data (if there is image noise, the linear system becomes overdetermined)

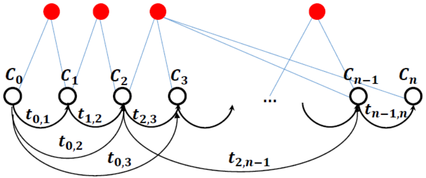
✓ We aim to find the optimal camera pose to minimize the least-squares error.

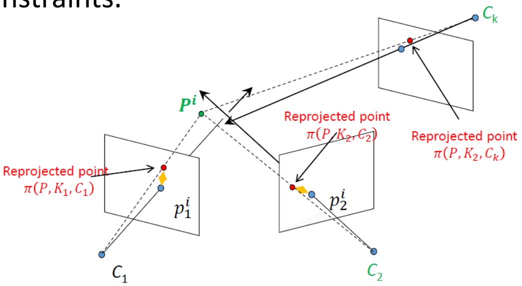


Bundle Adjustment

➢ Definition

✓ We extend two-view reprojection minimization to multi-view case, which is called "bundle adjustment".
✓ We typically treat the first camera as the world frame.
✓ We can reformulate the problem as a "graph optimization problem". Nodes are parameters to optimize, and edges are constraints.





Bundle Adjustment

- Definition

✓ We jointly optimize camera poses of all the cameras and 3D points:

$$\{P^i\}, C_1, \dots, C_n = argmin_{\{P^i\}, C_1, \dots, C_n,} \sum_{k=1}^{n} \sum_{i=1}^{N} \rho\left(p_k^i - \pi(P^i, K_k, C_k)\right)$$

where $\rho()$ is the Huber norm for robust estimation (introduced next week)

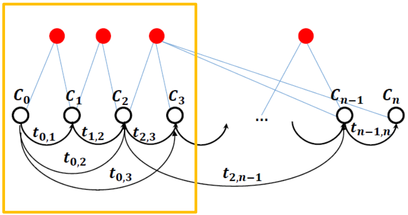
✓ We often use non-linear optimization, e.g., Gauss-Newton algorithm to minimize the error. Details will be introduced later.

Bundle Adjustment

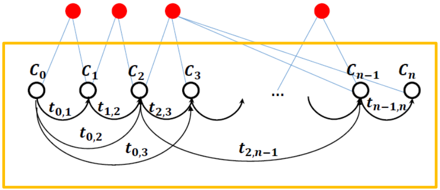
- Strategies for acceleration

- ✓ A small window size limits the number of parameters for the optimization and thus makes real time bundle adjustment possible.

- ✓ It is possible to reduce the computational complexity by just optimizing over the camera parameters and keeping the 3D landmarks fixed, e.g., motion-only BA.



A small window



Motion-only BA

Bundle Adjustment

- Photometric Bundle Adjustment

We can extend the photometric error between 1-by-1 frames to 1-by-N frames.

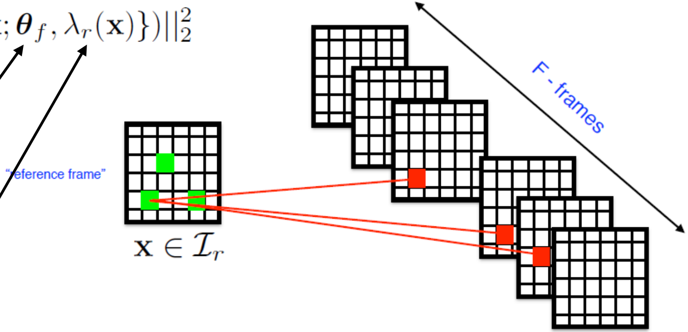
$$\arg\min_{\boldsymbol{\lambda},\boldsymbol{\theta}} \sum_{r=1}^{F} \sum_{\mathbf{x}\in\mathcal{I}_r} \sum_{f\in\mathrm{obs}(\mathbf{x})} \|\mathcal{I}_r(\mathbf{x}) - \mathcal{I}_f(\mathcal{W}\{\mathbf{x};\boldsymbol{\theta}_f,\lambda_r(\mathbf{x})\})\|_2^2$$

Perspective projection

Pixel

Pose

Depth



Non-linear Optimization

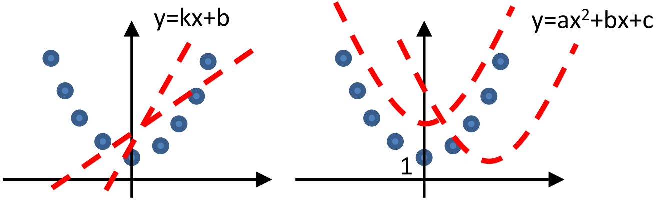
➢ Problem Formulation

✓ A teaser of curve fitting

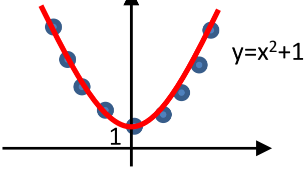
Input: A set of observed discrete points (no outliers here)

Step 1: Select a suitable model/function with unknown parameters

Step 2: Estimate the parameters by the least-squares method: We define an objective function, i.e., the sum of squared distances.



Model/function selection



Parameter estimation

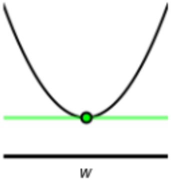
Non-linear Optimization

➢ Motivation of Gradient Descent Algorithm

To minimize the function, we can employ first-order optimality condition

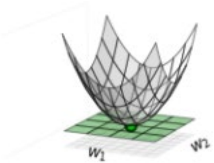
$$\min_{\mathbf{x}} F(\mathbf{x}) = \frac{1}{2}\|f(\mathbf{x})\|_2^2 \qquad \frac{\mathrm{d}F}{\mathrm{d}\mathbf{x}} = \mathbf{0}$$

If the derivative is simple, we can directly obtain the global minimum of objective function. However, what if the objective function is more complex?



$$\frac{\mathrm{d}}{\mathrm{d}w} g(w) = 0$$

1D function



$$\frac{\partial}{\partial w_1} g(\mathbf{v}) = 0$$
$$\frac{\partial}{\partial w_2} g(\mathbf{v}) = 0$$
$$\vdots$$
$$\frac{\partial}{\partial w_N} g(\mathbf{v}) = 0$$

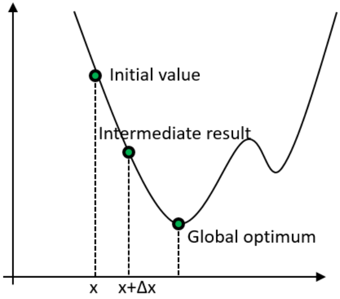
2D function

Non-linear Optimization

➢ Motivation of Gradient Descent Algorithm

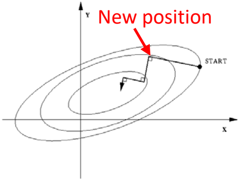
Instead of directly obtaining the global minimum, we iteratively minimize the function.
x_k is a temporary value. It is known.
Δx_k is the adjustment of the above temporary value. It is unknown.

1. Give an initial value $\mathbf{x}_0$.
2. For k-th iteration, we find an incremental value of $\Delta \mathbf{x}_k$, such that the object function $\boxed{\|f(\mathbf{x}_k + \Delta \mathbf{x}_k)\|_2^2}$ reaches a smaller value.
3. If $\Delta \mathbf{x}_k$ is small enough, stop the algorithm.
4. Otherwise, let $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}_k$ and return to step 2.



Non-linear Optimization

➢ Steepest method



Now consider the *k*-th iteration. Suppose the current solution is at x_k and we want to find the increment Δx_k. For problem simplification, we use the first-order Taylor expansion to re-write the objective function:

$$F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx \underbrace{F(\mathbf{x}_k)}_{\text{Known}} + \underbrace{\mathbf{J}(\mathbf{x}_k)^T}_{\text{Known}} \underbrace{\Delta\mathbf{x}_k}_{\text{Unknown}}$$

Gradient (Jacobi Matrix)

Along the negative gradient **direction**, we can ensure that the function decreases:

$$\Delta\mathbf{x}^* = -\mathbf{J}(\mathbf{x}_k)$$

We do not explicitly compute Δx

Δx is only a direction. We also manually select another step length parameter (learning rate), say, λ. The smaller function value is $F(\mathbf{x}_k) - \mathbf{J}(\mathbf{x}_k)\,\lambda$

Non-linear Optimization

➢ Newton's method

$$\boxed{F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{J}(\mathbf{x}_k)^T \Delta\mathbf{x}_k + \frac{1}{2}\Delta\mathbf{x}_k^T \mathbf{H}(\mathbf{x}_k)\Delta\mathbf{x}_k}$$

We can also use the second-order Taylor expansion to re-write the objective function:

$$\Delta\mathbf{x}^* = \arg\min \left(F(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T \mathbf{H} \Delta\mathbf{x} \right)$$

Known Unknown

We leverage the first-order optimality condition, i.e., computing the derivative with respect to Δx and setting the result to zero. We thus can obtain

$$\mathbf{J} + \mathbf{H}\Delta\mathbf{x} = \mathbf{0} \Rightarrow \mathbf{H}\Delta\mathbf{x} = -\mathbf{J}$$

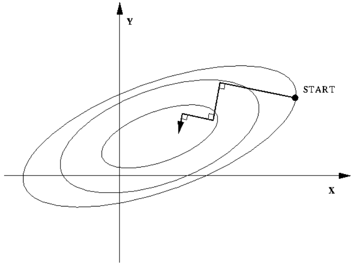
Hessian matrix

Non-linear Optimization

➢ Gauss-Newton Method

✓ Motivation

Steepest method results in the zig-zag descending trajectory



Newton's method is time consuming due to the computation of Hessian matrix

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

We need a more effective method: We will introduce a representative method "**Gauss-Newton algorithm**".

Non-linear Optimization

$$\boxed{\min_{\mathbf{x}} F(\mathbf{x}) = \frac{1}{2}\left\|\boxed{f(\mathbf{x})}\right\|_2^2}$$

- Gauss-Newton Method

Similar to the steepest method, we begin with first-order Taylor expansion

$$\boxed{f(\mathbf{x}+\Delta\mathbf{x})} \approx \boxed{f(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T \Delta\mathbf{x}}$$

We aim to find the optimal Δx to minimize this function

$$\Delta\mathbf{x}^* = \arg\min_{\Delta\mathbf{x}} \frac{1}{2}\left\|\boxed{f(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T \Delta\mathbf{x}}\right\|^2$$

Let us first expand this function:

$$\begin{aligned}\frac{1}{2}\left\|f(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T \Delta\mathbf{x}\right\|^2 &= \frac{1}{2}\left(f(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T \Delta\mathbf{x}\right)^T\left(f(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T \Delta\mathbf{x}\right) \\ &= \frac{1}{2}\left(\|f(\mathbf{x})\|_2^2 + 2f(\mathbf{x})\mathbf{J}(\mathbf{x})^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{J}(\mathbf{x})\mathbf{J}(\mathbf{x})^T\Delta\mathbf{x}\right)\end{aligned}$$

Perfect square formula

Non-linear Optimization

$$\boxed{\frac{1}{2}\left(\|f(\mathbf{x})\|_2^2 + 2f(\mathbf{x})\mathbf{J}(\mathbf{x})^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{J}(\mathbf{x})\mathbf{J}(\mathbf{x})^T\Delta\mathbf{x}\right)}$$

- Gauss-Newton Method

We compute the derivative of the above function with respect to Δx, and then set the derivate to zero:

$$\mathbf{J}(\mathbf{x})f(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{J}^T(\mathbf{x})\Delta\mathbf{x} = \mathbf{0}$$

We transform the above into

$$\underbrace{\mathbf{J}(\mathbf{x})\mathbf{J}^T}_{\mathbf{H}(\mathbf{x})}(\mathbf{x})\Delta\mathbf{x} = \underbrace{-\mathbf{J}(\mathbf{x})f(\mathbf{x})}_{\mathbf{g}(\mathbf{x})}$$

We obtain a linear system to compute Δx

An approximation to Hessian matrix

$$\mathbf{H}\Delta\mathbf{x} = \mathbf{g}$$

Non-linear Optimization

- Application to Bundle Adjustment (A Teaser)

✓ General objective function simplification by Gauss-Newton

Jacobian matrix w.r.t. pose and point

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

Adjustment of camera pose and 3D point

✓ We have to compute derivative w.r.t. SO3/SE3. It evolves addition and subtraction operation.

✓ Intuitively, R1 is in SO3 and R2 is in SO3, but we cannot guarantee that R1 + R2 is in SO3.

✓ To solve this problem, we first map Lie Group to Lie Algebra, and compute the derivative by Lie Algebra. **Due to limited time, we will skip this content this year**.

Summary

- Error Metrics
- Bundle Adjustment
- Non-linear Optimization

Thank you for your listening!
If you have any questions, please come to me :-)