

## 第 5 讲 后端优化实践：逐行手写求解器

贺一家，高翔，崔华坤

2019 年 7 月 14 日

## ① 非线性最小二乘求解

solver 流程回顾

solver 代码讲解

## ② 滑动窗口算法

滑动窗口算法回顾

VINS-Mono 中的滑动窗口算法

## ③ 作业

# 非线性最小二乘问题求解：solver

## 高斯牛顿求解流程

有如下最小二乘系统，对应的图模型如有图所示：

$$\xi = \underset{\xi}{\operatorname{argmin}} \frac{1}{2} \sum_i \|\mathbf{r}_i\|_{\Sigma_i}^2 \quad (1)$$

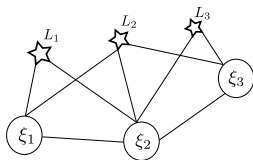
对应的高斯牛顿求解，normal equation：

$$\underbrace{\mathbf{J}^\top \Sigma^{-1} \mathbf{J}}_{\mathbf{H} \text{ or } \Lambda} \delta \xi = \underbrace{-\mathbf{J}^\top \Sigma^{-1} \mathbf{r}}_{\mathbf{b}} \quad (2)$$

连加形式：

高斯牛顿的核心：对残差本身  
进行一阶泰勒展开

$$\sum_{i=1}^n \mathbf{J}_i^\top \Sigma_i^{-1} \mathbf{J}_i \delta \xi = - \sum_{i=1}^n \mathbf{J}_i^\top \Sigma_i^{-1} \mathbf{r} \quad (3)$$



# SLAM 问题中高斯牛顿方程的求解

## 利用舒尔补加速 SLAM 问题的求解

直接求解  $\Delta \mathbf{x} = -\mathbf{H}^{-1}\mathbf{b}$ , 计算量大。**解决办法：舒尔补**, 利用 SLAM 问题的稀疏性求解。

比如, 某单目 BA 问题, 其信息矩阵如有图所示, 可以将其分为: **舒尔补：不是一起求, 而是分开来求**

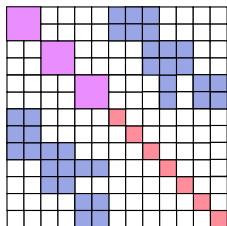
$$\begin{bmatrix} \mathbf{H}_{pp} & \mathbf{H}_{pl} \\ \mathbf{H}_{lp} & \mathbf{H}_{ll} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}_p^* \\ \Delta \mathbf{x}_l^* \end{bmatrix} = \begin{bmatrix} -\mathbf{b}_p \\ -\mathbf{b}_l \end{bmatrix} \quad (4)$$

可以利用舒尔补操作, 使上式中信息矩阵变成下三角, 从而得到:

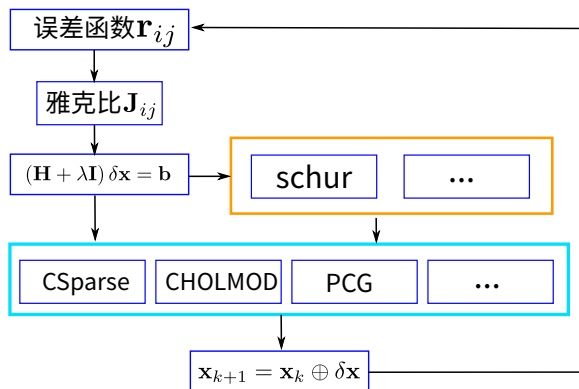
$$(\mathbf{H}_{pp} - \mathbf{H}_{pl}\mathbf{H}_{ll}^{-1}\mathbf{H}_{lp}^T) \Delta \mathbf{x}_p^* = -\mathbf{b}_p + \mathbf{H}_{pl}\mathbf{H}_{ll}^{-1}\mathbf{b}_l \quad (5)$$

求得  $\Delta \mathbf{x}_p^*$  后, 再计算  $\Delta \mathbf{x}_l^*$ :

$$\mathbf{H}_{ll}\Delta \mathbf{x}_l^* = -\mathbf{b}_l - \mathbf{H}_{lp}^T\Delta \mathbf{x}_p^* \quad (6)$$



# solver 全流程回顾



推荐阅读<sup>123</sup>.

<sup>1</sup>Giorgio Grisetti et al. "A tutorial on graph-based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.

<sup>2</sup>Rainer Kümmerle et al. "g 2 o: A general framework for graph optimization". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

<sup>3</sup>Manolis IA Lourakis and Antonis A Argyros. "SBA: A software package for generic sparse bundle adjustment". In: *ACM Transactions on Mathematical Software (TOMS)* 36.1 (2009), p. 2.

## solver 求解中的小疑问

- ① 上节课说到信息矩阵  $H$  不满秩，那求解的时候如何操作呢？
- 使用 LM 算法，加阻尼因子使得系统满秩，可求解，但是求得的结果可能会往零空间变化。这个零空间的变化，直观上讲就是整体的偏移
  - 添加先验约束，增加系统的可观性。比如 g2o tutorial 中对第一个 pose 的信息矩阵加上单位阵  $H_{[11]} + I$ 。

添加先验约束，白话就是固定某些项。比如固定初始位姿和一个landmark，固定两个相机位姿，等等

- ② orbslam, svo 等等求 mono BA 问题时，fix 一个相机 pose 和一个特征点，或者 fix 两个相机 pose，也是为了限定优化值不乱飘。那代码如何实现 fix 呢？

- 添加超强先验，使得对应的信息矩阵巨大（如， $10^{15}$ ），就能使得  $\Delta x = 0$ ;
- 设定对应雅克比矩阵为 0，意味着残差等于 0。求解方程为  $(0 + \lambda I) \Delta x = 0$ ，只能  $\Delta x = 0$ 。

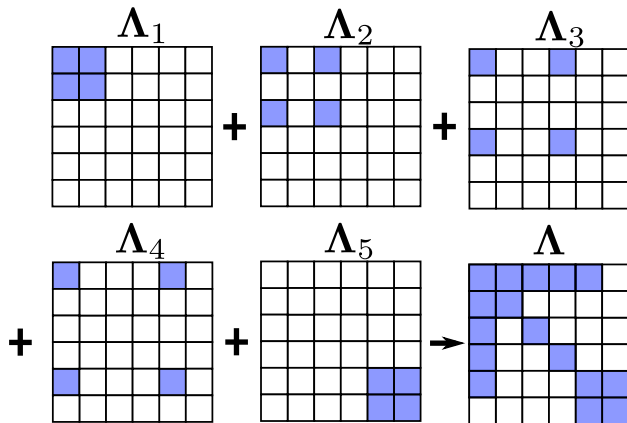
小结：不满秩就会飘，有三种解决方法：

- 1、随便你怎么飘，最后把你移回去
- 2、想办法让你行动很困难，飘不起来
- 2、想办法把你固定住，不让你飘

# 单目 Bundle Adjustment 求解代码讲解

信息矩阵的拼接很容易出bug，要小心

核心问题：矩阵块的对应关系，如何拼接信息矩阵。



# 单目 Bundle Adjustment 求解代码讲解

所谓的顶点就是我们要求的变量：相机的pose是顶点，相机系的landmark也是顶点。

我们要关注几个点：

- 一、顶点是几维的
- 二、顶点的编号
- 三、顶点在矩阵中的idx
- 四、广义加法

## 代码讲解时间

代码框架和  $g^2o$  类似<sup>4</sup>。

### 顶点：vertex

- id, 维度, 矩阵 index.
- 变量加法.

### 边：edge

- 残差计算
- 雅克比矩阵计算
- 保存对应的顶点

### 求解器：solver

- LM 算法
- make H, b
- 线性求解器：  
QR, SVD, PCG, ...

<sup>4</sup>Giorgio Grisetti et al. "g2o: A general framework for (hyper) graph optimization". In: *Arch. Rep.* (2011).



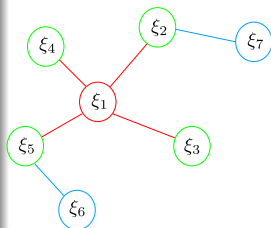
## Section 2

# 滑动窗口算法

## 第四讲滑动窗口算法回顾

### toy example 3

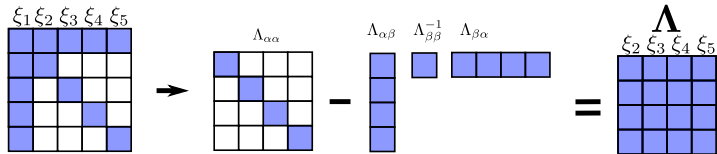
- ① 如右图所示，在  $t \in [0, k]s$  时刻，系统中状态量为  $\xi_i, i \in [1, 6]$ 。第  $k'$  时刻，加入新的观测和状态量  $\xi_7$ 。
- ② 在第  $k$  时刻，最小二乘优化完以后，marg 掉变量  $\xi_1$ 。被 marg 的状态量记为  $x_m$ ，剩余的变量  $\xi_i, i \in [2, 5]$  记为  $x_r$ 。
- ③ marg 发生以后， $x_m$  所有的变量以及对应的测量将被丢弃。同时，这部分信息通过 marg 操作传递给了保留变量  $x_r$ 。marg 变量的信息跟  $\xi_6$  不相关。
- ④ 第  $k'$  时刻，加入新的状态量  $\xi_7$  (记作  $x_n$ ) 以及对应的观测，开始新一轮最小二乘优化。



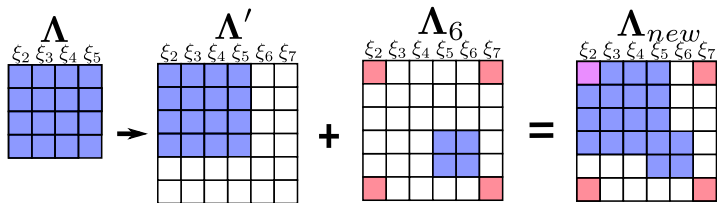
- 红色为被 marg 变量以及测量约束。
- 绿色为跟 marg 变量有关的保留变量。
- 蓝色为和 marg 变量无关联的变量。

# 滑动窗口算法关键步骤可视化

## 步骤 1: 构建先验



## 步骤 2: 先验 + 新测量信息 → 新的信息矩阵



和直接 Bundle Adjustment 相比，多了一个先验矩阵的维护。

# 滑动窗口算法中关键问题

## 如何更新先验残差？

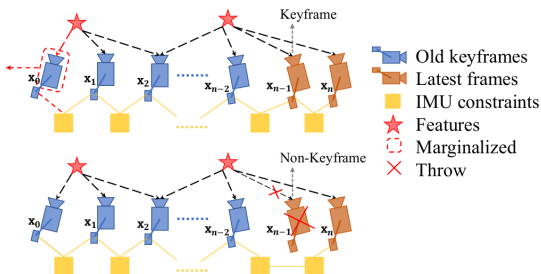
- 目的：虽然先验信息矩阵固定不变，但随着迭代的推进，变量被不断优化，先验残差需要跟随变化。否则，求解系统可能崩溃。
- 方法：先验残差的变化可以使用一阶泰勒近似。

$$\begin{aligned} \mathbf{b}'_p &= \mathbf{b}_p + \frac{\partial \mathbf{b}_p}{\partial \mathbf{x}_p} \delta \mathbf{x}_p \\ &= \mathbf{b}_p + \frac{\partial \left( -\mathbf{J}^\top \Sigma^{-1} \mathbf{r} \right)}{\partial \mathbf{x}_p} \delta \mathbf{x}_p \\ &= \mathbf{b}_p - \Lambda_p \delta \mathbf{x}_p \end{aligned} \tag{7}$$

# VINS-Mono 中的滑动窗口算法

## two way marginalization

- 当滑动窗口中第二新的图像帧为关键帧，则 marg 最老的帧，以及上面的路标点。
- 当滑动窗口中第二新的图像帧不是关键帧，则丢弃这一帧上的视觉测量信息，IMU 预积分传给下一帧。



引自<sup>5</sup>

<sup>5</sup>Tong Qin, Peiliang Li, and Shaojie Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

# VINS-Mono 中的滑动窗口算法

## 代码讲解时间

## 基础题

- 1 完成单目 Bundle Adjustment 求解器 problem.cc 中的部分代码。
  - 完成 Problem::MakeHessian() 中信息矩阵  $H$  的计算。
  - 完成 Problem::SolveLinearSystem() 中 SLAM 问题的求解。
- 2 完成滑动窗口算法测试函数。
  - 完成 Problem::TestMarginalize() 中的代码，并通过测试。

说明：为了便于查找作业位置，代码中留有 TODO:: home work 字样。

## 提升题

paper reading<sup>a</sup>，请总结论文：优化过程中处理  $H$  自由度的不同操作方式。总结内容包括：具体处理方式，实验效果，结论。

<sup>a</sup>Zichao Zhang, Guillermo Gallego, and Davide Scaramuzza. "On the comparison of gauge freedom handling in optimization-based visual-inertial state estimation". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2710–2717.