分类号: _____ 单位代码: _____

学 号: _____

浙江大学

硕士学位论文

论文题目: 面向视觉惯性 SLAM 的通用增量式集束调整框架

作者姓名: _____

指导教师: _____

学科 (专业): 计算机科学与技术

研究方向: 计算机视觉

所在学院: 计算机科学与技术学院

提交日期_____

A Dissertation Submitted to Zhejiang
University for the Degree of
Master of Engineering

TITLE: A General Incremental Bundle Adjustment Framework for Visual-Inertial SLAM

Author: _____

Supervisor: _____

Subject: Computer Science and Technology

Research Domain: Computer Vision

College: College of Computer Science and Technology

Submitted Date_____

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得浙江大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解浙江大学有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权浙江大学可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

摘要

随着人工智能概念的兴起和增强现实、无人机、移动机器人、自动驾驶等行业的发展，工业界与学术界对高效率、高精度的鲁棒 SLAM 算法的需求越来越大，而 SLAM 应用中，状态估计方法的效率和精度极大地制约了 SLAM 算法的性能表现。目前主流的 SLAM 系统一般使用集束调整来进行非线性状态估计。一些系统使用了开源的通用非线性最小二乘求解器，为了适应不同类型的优化问题，这一类求解器通常采用批量式最小二乘算法，牺牲了效率；一些系统使用了增量式的集束调整算法以提升效率，但由于解法固定，只适用于特定类型的非线性目标函数和参数化方法，或与系统的耦合程度较高，通用性不佳。

本文基于以上分析，提出了面向视觉惯性 SLAM 的增量式通用集束调整框架，一方面使用了增量式集束调整算法以提升效率：1）本文使用了因子图来编码并辅助增量构建舒尔补的过程，在减少线性化和消元过程中的冗余计算的同时生成贝叶斯推断树；2）并针对基于增量式舒尔补的算法中常见的舒尔补矩阵秩亏的问题，提出了混合式 LM-DL 算法，在不破坏增量构建过程的前提下增强了线性解的数值稳定性；3）在变量回代求解部分，提出了基于贝叶斯推断树的 PBT 回代算法，以提高回代求解时的计算效率和变量一致性。

另一方面，该框架在保证一定效率和精度的基础上保持了很好的通用性，除了求解视觉惯性 SLAM 的集束调整问题，也支持求解一般的非线性最小二乘问题：1）该框架实现了视觉惯性 SLAM 中常用的 IMU 预积分目标函数和重投影误差目标函数，同时支持用户自定义任意类型的目标函数；2）并提供了 SLAM 中常用的旋转矩阵和四元数参数化方法，同时支持用户自定义任意类型的参数化方法；3）在线性部分，该框架提供了稀疏/稠密版本的 Cholesky 分解、QR 分解和增量式预处理共轭梯度法等多种线性求解算法，并支持用户定制符合实际数值需求的线性求解算法。

本文提出的集束调整框架使用了对 CPU 缓存更为友好的块状稀疏矩阵来实现矩阵计算和存储，并在模拟数据和真实的集束调整数据集上进行了测试，通过实验验证了求解的效率和精度。

关键词： SLAM，增量式舒尔补，贝叶斯推断树，集束调整，状态估计

Abstract

With the rise of artificial intelligence concepts and the development of industries such as augmented reality, drones, mobile robots, and self-driving cars, there is a growing demand on industry and academia on robust SLAM algorithms with high efficiency and high precision. In the SLAM applications, the speed and accuracy of the state estimation methods greatly limit the performance of the SLAM algorithms. Most SLAM systems employ bundle adjustment (BA) for state estimation. Some systems use the open source general nonlinear least squares (NLS) solvers. In order to adapt to different types of NLS problems, those solvers are usually based on batch NLS algorithms, which sacrifice efficiency; Existing incremental solvers are very efficient, but only able to solve BA problems with certain cost functions and local parameterizations or tightly coupled with certain SLAM systems, and thus difficult to integrate into other SLAM systems.

Upon these analyses, we proposed a general incremental BA framework for visual-inertial SLAM (VISLAM). On the one hand, incremental algorithms are used to improve efficiency: 1) A factor graph is used to encode as well as to help the incremental construction of the Schur complement, in order to reduce the redundant computations at the linearization and elimination steps. And in the meanwhile, a Bayes tree is generated simultaneously; 2) We also addressed the rank-deficient issue while solving the Schur equation, by proposing a Hybrid LM-DL method, which enhances the numerical stability of the linear solutions without destroying the incremental constructions of the Schur complement; 3) In the back-substitution part, a partial Bayes tree (PBT) algorithm is used to improve the efficiency and variable consistency.

On the other hand, the BA framework retains versatility while guaranteeing efficiency and precision. Besides BA problems, general NLS problems are supported: 1) We implemented IMU preintegration cost function and re-projection error cost function, which are commonly required by VISLAM. Users can also implement their own cost functions of arbitrary types; 2) Local parameterizations like rotation matrix and quaternion are built-in. Users can also define their own local parameterization methods; 3) At the linear solving step, a variety of linear strategies are provided, such as sparse/dense Cholesky solver, QR

solver and incremental preconditioned conjugate gradient (I-PCG) solver. It is also possible to implement customized linear algorithms that meet different numerical requirements.

The BA framework proposed is fully based on the block sparse matrix which is more cache-friendly. We tested it on both simulated datasets and real BA datasets. The efficiency and accuracy are verified by experiments.

Keywords: SLAM, incremental Schur complement, Bayes tree, bundle adjustment, state estimation

目录

插图

表格

第1章 绪论

随着人工智能概念的兴起和增强现实（Augmented Reality，简称 AR）、无人机、移动机器人、自动驾驶等行业的发展，工业界与学术界对高效率、高精度的鲁棒三维感知算法的需求越来越大。这类应用通常需要实时并持续地获取场景空间的三维结构信息以及自身的位姿（朝向和位置）和运动信息。传统的三维感知方案受限于自身的局限性，很难同时满足精度、效率和鲁棒性的需求。如普通的 GPS 定位由于频率和精度较低且无法在室内或恶劣天气中使用；高精度 GNSS 系统（Global Navigation Satellite System）和惯性导航系统 INS（Inertial Navigation System）虽然在精度、效率和鲁棒性上都能达到要求，但由于售价高达数千至数十万美元，通常只能在高端领域中使用；廉价的消费级别惯性测量单元（Inertial Measurement Unit，简称 IMU）能以非常高的频率实时获取自身的运动信息且具备非常高的稳定性，但由于其传感器噪声过大，误差累积严重，通常只能用来测量旋转信息或初步地测量平移信息。

同时定位和建图（Simultaneous Localization and Mapping，简称 SLAM）算法指在未知环境中，让搭载传感器的设备从某一位置出发对场景进行观测，在运动和观测过程中计算出自身的位姿（朝向和位置），同时逐渐构建场景地图的过程，其中传感器通常指消费级的相机设备，也可能包括但激光雷达、IMU 等其他传感器。目前在计算机视觉领域，基于多视图几何和运动恢复结构（Structure from Motion，简称 SfM）的算法[4;5] 已经能做到非常精确的场景地图重建和传感器的定位，却由于其只能离线工作而不能满足实时性要求高的应用。视觉 SLAM（Visual SLAM，简称 VSLAM）可以被认为是在线版本的 SfM，是使用相机（单目相机或多目相机）作为唯一的传感器，利用视觉信息进行定位和定位和建图的方法。相对于 SfM，VSLAM 利用了连续图像帧的局部性，逐步、增量地计算出位姿并恢复场景的三维结构，因此能比较容易地达到较高的精度和实时的效率。然而，单目 VSLAM 系统存在一定的局限性，即不能恢复场景的绝对尺度信息。另外，VSLAM 非常依赖相机的成像质量，在图像质量不佳的时候则较难保证算法的鲁棒性。以 V-LOAM[6] 为代表方法一类激光 SLAM 系统使用了昂贵的激光雷达传感器。由于激光雷达能够获取高精度的三维点云，使用激光雷达的 SLAM 系统能够获取非常高精度的建图和定位结果，同时能获得场景的绝对尺度。但是受限于成本、功耗和体积，激光雷达难以使用在移动 AR、无人机或中小型移动机器人等应用上。视觉惯性 SLAM（Visual Inertial SLAM，简称 VISLAM）则是使用了相机和 IMU 作为传感器，基于 IMU 提供的角速度测量和加速度测量，VISLAM 原

生就具备了估计绝对尺度的能力。同时，受益于 IMU 传感器的高效和稳定，通过融合视觉信息和惯性信息，VISLAM 比 VSLAM 具有更好的鲁棒性。

此外，VISLAM 所需的相机和消费级 IMU 的价格低廉，容易获取，目前大部分手机都配备可以供 SLAM 算法使用的相机和 IMU。综合成本、体积、功耗和性能，VISLAM 与传统的三维感知方案或其他类型的 SLAM 方案相比，更适合于移动终端的 AR、小型移动机器人、无人机等应用场景，是未来移动的智能设备上不可或缺的一项关键技术。

在 SLAM 应用中，状态估计方法的效率和精度极大地制约了 SLAM 算法的性能表现。目前主流的 SLAM 系统一般使用集束调整来进行非线性状态估计。一些系统使用了开源的通用非线性最小二乘求解器，如 Ceres-Solver[7]、g^{2}o[8] 等。为了适应不同类型的优化问题，这一类求解器通常采用批量式最小二乘算法，牺牲了效率，因而难以在移动设备上达到实时的性能；一些系统使用了增量式的集束调整算法以提升效率，比如 SLAM++[9]，这一类求解器由于解法固定，通常只适用于特定类型的非线性目标函数和参数化方法，或者与系统的耦合程度较高，通用性不佳。

本文基于以上分析，提出了面向 VISLAM 的增量式通用集束调整框架，一方面使用了增量式集束调整算法以提升算法的效率：

- 在线性化和消元部分，本文使用了基于因子图的增量式方法构建舒尔补。这一方法可以显著减少冗余的计算量，提升舒尔补方程的构建效率。同时，利用因子图还被用来编码舒尔补的构建过程，并在这一过程中生成对应的贝叶斯推断树；

- 现有的增量式集束调整算法认为莱文贝格-马夸特（Levenberg-Marquardt，简称 LM）方法中的阻尼因子会修改正规矩阵，从而破坏舒尔补构建的增量特性，故一般会使用高斯-牛顿（Gauss-Newton，简称 GN）法或 Dog-Leg（简称 DL）法，如 ICE-BA[10] 和 SLAM++。本文针对这一类使用增量式舒尔补的算法中常见的舒尔补矩阵秩亏的问题，提出了混合式 LM-DL（Hybrid Levenberg-Marquardt-Dog-Leg，简称 HLMDL）方法。这一方法可以在不破坏舒尔补构建的增量特性的前提下增强线性解的数值稳定性；

- 在变量回代求解部分，本文基于舒尔补构建过程中生成的贝叶斯推断树，提出了使用部分贝叶斯推断树（Partial Bayes Tree，简称 PBT）的方法编码三维点状态和相机状态之间的推断关系，以达到增量式回代求解的目的，进一步提高效率和变量一致性。

另一方面，该框架在保证一定效率和精度的基础上保持了很好的通用性：

- 该框架实现了 VISLAM 中常用的 IMU 预积分[11] 目标函数和重投影误差目标函数，同时支持用户自定义任意类型的目标函数；
- 在变量参数化部分，提供了 SLAM 中常用的旋转矩阵和四元数参数化方法，同时支持用户自定义任意类型的参数化方法；
- 在线性部分，该框架实现了块状稀疏矩阵版本和稠密版本的 Cholesky 分解、QR 分解和增量式预处理共轭梯度（Incremental Preconditioned Conjugate Gradient，简称 I-PCG）法等多种线性求解算法。并开放了线性求解器接口，支持用户定制符合实际数值需求的线性求解算法。

因此，该框架除了能够求解 VISLAM 的集束调整问题，也支持求解一般的非线性最小二乘问题。

本文提出的集束调整框架完全使用了对 CPU 缓存更为友好的块状稀疏矩阵来实现矩阵计算和存储，并在模拟数据和真实的集束调整数据集上进行了测试，通过实验验证了求解的效率和精度。

1.1 基于扩展卡尔曼滤波的状态估计

在 SLAM 领域中，主流的状态估计算法有两类：基于扩展卡尔曼滤波（Extended Kalman Filter，简称 EKF）的滤波法和基于非线性最小二乘（Nonlinear Least Squares）的优化法。两类方法没有本质上的区别，都是使用最大化后验概率（Maximum A-Posteriori estimation）的思想求解非线性系统的最优状态分布。本节将介绍 EKF 的基本原理。

EKF 将状态估计的过程分为状态预测（predict）和状态更新（update）两个步骤，分别对应状态的转移模型（propagation model）和观测模型（observation model）。在状态预测阶段，EKF 通过状态转移模型预测当前系统状态的先验分布，而在状态更新阶段，EKF 通过当前时刻的观测模型和观测值对状态的分布进行修正，得到后验状态分布：

$$\begin{aligned} &\text{转移模型} \quad \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) \\ &\text{观测模型} \quad \boldsymbol{z}_{k+1} = h(\boldsymbol{x}_{k+1}, \boldsymbol{v}_{k+1}) \end{aligned} \tag{1.1}$$

其中 $\boldsymbol{u}_k$ 为输入控制变量，$\boldsymbol{w}_k$ 和 $\boldsymbol{v}_{k+1}$ 分别为独立的系统噪声和观测噪声，均符合零均值高斯分布：

$$\begin{aligned} p(\boldsymbol{w}_k) &\sim \mathcal{N}(\mathbf{0}, \mathrm{Q}) \\ p(\boldsymbol{v}_{k+1}) &\sim \mathcal{N}(\mathbf{0}, \mathrm{R}) \end{aligned} \tag{1.2}$$

1.1.1 状态预测

对于一个离散时间的非线性系统，EKF 假设已知 t_k 时刻系统的状态 $\boldsymbol{x}_k$ 符合高斯分布：

$$p(\boldsymbol{x}_k) \sim \mathcal{N}(\hat{\boldsymbol{x}}_{k|k}, \mathrm{P}_{k|k}) \tag{1.3}$$

由于系统噪声 $\boldsymbol{w}_k$ 未知，在 t_{k+1} 时刻可以通过转移模型预测当前系统状态的先验分布：

$$\begin{aligned} p(\boldsymbol{x}_{k+1}) &\sim \mathcal{N}(\hat{\boldsymbol{x}}_{k+1|k}, \mathrm{P}_{k+1|k}) \\ \hat{\boldsymbol{x}}_{k+1|k} &= f(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{u}_k, \boldsymbol{0}) \\ \mathrm{P}_{k+1|k} &= \mathrm{A}_{k+1}\mathrm{P}_{k|k}\mathrm{A}_{k+1}^{\top} + \mathrm{W}_{k+1}\mathrm{Q}_k\mathrm{W}_{k+1}^{\top} \end{aligned} \tag{1.4}$$

其中

$$\begin{aligned} \mathrm{A}_{k+1} &\doteq \frac{\partial f}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{u}_k, \boldsymbol{0}) \\ \mathrm{W}_{k+1} &\doteq \frac{\partial f}{\partial \boldsymbol{w}}(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{u}_k, \boldsymbol{0}) \end{aligned} \tag{1.5}$$

1.1.2 状态更新

假设在 t_{k+1} 时刻获得了对系统的观测 $\boldsymbol{z}_{k+1}$ 而观测噪声 $\boldsymbol{v}_{k+1}$ 未知，根据观测模型可以得到观测值的条件概率分布：

$$\begin{aligned} p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1}) &\sim \mathcal{N}(\hat{\boldsymbol{z}}_{k+1}, \mathrm{S}_{k+1}) \\ \hat{\boldsymbol{z}}_{k+1} &= h(\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{0}) \\ \mathrm{S}_{k+1} &= \mathrm{V}_{k+1}\mathrm{R}_{k+1}\mathrm{V}_{k+1}^{\top} \end{aligned} \tag{1.6}$$

其中

$$\begin{aligned} \mathrm{H}_{k+1} &\doteq \frac{\partial h}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{0}) \\ \mathrm{V}_{k+1} &\doteq \frac{\partial h}{\partial \boldsymbol{v}}(\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{0}) \end{aligned} \tag{1.7}$$

为了简洁表示，以下省略下标。由于后验概率 $p(\boldsymbol{x}|\boldsymbol{z}) \propto p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})$，可以构建最大化后验概率问题：

$$\begin{aligned} \max_{\boldsymbol{x}} &\frac{1}{\sqrt{2\pi}\mathrm{P}^{1/2}} \cdot \exp\left(-\tfrac{1}{2}\left\|\boldsymbol{x}-\hat{\boldsymbol{x}}\right\|_{\mathrm{P}^{-1}}^2\right) \\ &\cdot \frac{1}{\sqrt{2\pi}\mathrm{S}^{1/2}} \cdot \exp\left(-\tfrac{1}{2}\left\|\boldsymbol{z}-\hat{\boldsymbol{z}}\right\|_{\mathrm{S}^{-1}}^2\right) \end{aligned} \tag{1.8}$$

EKF 假设状态先验 $\hat{\boldsymbol{x}}$ 足够接近最优解 $\boldsymbol{x}$，且 $\boldsymbol{e} \doteq \boldsymbol{x} - \hat{\boldsymbol{x}}$。估最大化上式相当于最小化：

$$\min_{\boldsymbol{x}} \tfrac{1}{2}\left\|\boldsymbol{e}\right\|_{\mathrm{P}^{-1}}^2 + \tfrac{1}{2}\left\|\boldsymbol{z}-h(\hat{\boldsymbol{x}}, \boldsymbol{0})\right\|_{\mathrm{S}^{-1}}^2 \tag{1.9}$$

$$\begin{aligned}\boldsymbol{e} &= \left(\mathrm{P} + \mathrm{H}^{\top}\mathrm{S}^{-1}\mathrm{H}\right)^{-1}\mathrm{H}^{\top}\mathrm{S}^{-1}(\boldsymbol{z} - h(\hat{\boldsymbol{x}}, \mathbf{0})) \\ &= \mathrm{P}\mathrm{H}^{\top}\left(\mathrm{H}\mathrm{P}\mathrm{H}^{\top} + \mathrm{S}\right)^{-1}(\boldsymbol{z} - h(\hat{\boldsymbol{x}}, \mathbf{0}))\end{aligned} \tag{1.10}$$

记 t_{k+1} 时刻的卡尔曼增益（Kalman gain）为

$$\mathrm{K}_{k+1} \doteq \mathrm{P}_{k+1|k}\mathrm{H}_{k+1}^{\top}\left(\mathrm{H}_{k+1}\mathrm{P}_{k+1|k}\mathrm{H}_{k+1}^{\top} + \mathrm{S}_{k+1}\right)^{-1} \tag{1.11}$$

可以对状态进行更新：

$$\hat{\boldsymbol{x}}_{k+1|k+1} = \hat{\boldsymbol{x}}_{k+1|k} + \mathrm{K}_{k+1}(\boldsymbol{z}_{k+1} - h(\hat{\boldsymbol{x}}_{k+1|k}, \mathbf{0})) \tag{1.12}$$

同时，要更新后验协方差矩阵：

$$\mathrm{P}_{k+1|k+1} = (\mathrm{I} - \mathrm{K}_{k+1}\mathrm{H}_{k+1})\,\mathrm{P}_{k+1|k}\,(\mathrm{I} - \mathrm{K}_{k+1}\mathrm{H}_{k+1})^{\top} + \mathrm{K}_{k+1}\mathrm{S}_{k+1}\mathrm{K}_{k+1}^{\top} \tag{1.13}$$

因此，在状态更新阶段，可以得到状态的后验概率分布：

$$p(\boldsymbol{x}_{k+1}) \sim \mathcal{N}(\hat{\boldsymbol{x}}_{k+1|k+1}, \mathrm{P}_{k+1|k+1}) \tag{1.14}$$

1.2 基于非线性优化方法的状态估计

继上一节介绍了基于 EKF 的状态估计方法，本节将继续介绍基于非线性最小二乘的状态估计方法，包括几种主流的求解最小二乘估计的方法以及加速求解的算法，并从理论角度分析滤波法和优化法的优劣。

目前绝大多数的 SLAM 算法通过集束调整[12]（Bundle Adjustment）方法来对状态进行估计。在传统的视觉 SfM 算法中，集束调整是指通过联合优化所有的视觉观测误差来同时求解最优的相机位姿状态和三维点状态的方法。而对于使用了更多传感器的 SLAM 系统，除了三维点和位姿状态，集束调整算法还需要实时地、持续对系统的其他状态比作出估计。对于 VISLAM，集束调整通常会通过联合优化所有的视觉观测和惯性观测来同时求解相机或 IMU 的位姿、速度以及 IMU 的 bias 状态。

1.2.1 基于图优化的状态估计

图优化方法是分析并求解 SLAM 后端状态估计问题的常用工具。最早在机器人领域，为了解决多段激光传感器数据额融合时的全局一致性问题，[13] 和 [14] 提出了基于位姿图的优化方法。[15] 在此基础上进一步提出了 GraphSLAM，使用由位姿状态和三维点状态

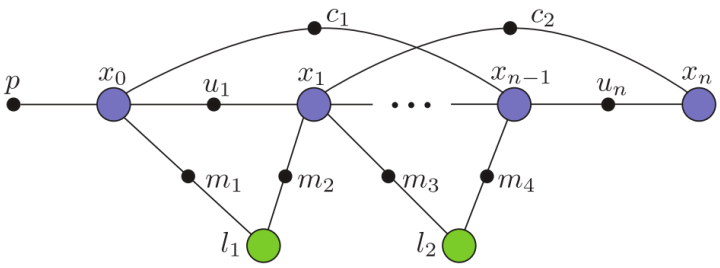

图 1.1 因子图示例 [1]：紫色节点代表位姿状态，绿色节点代表三维点状态，黑色节点代表状态之间的约束因子。

以及它们之间的约束构成的因子图来描述集束调整问题。因子图可以很直观地描述集束调整问题中状态和约束之间的关联，有助于分析集束调整问题的各种性质。经过长时间的发展，图优化方法已经成为 SLAM 领域的经典方法。

图 1.1展示了一类常见的因子图结构。按照图示的例子，一个 SLAM 问题中可能存在的约束有状态变量的先验约束 p，相邻状态之间的相对位姿约束 $u_1 \dots u_n$，非相邻状态之间的回路闭合约束 c_1, c_2 以及视觉观测约束 $m_1 \dots m_4$ 等。而求解完整集束调整的过程就相当于最大化整个因子图中的所有约束的集合 $\mathcal{Z}$ 关于所有状态集合 $\mathcal{X}$ 的联合条件概率：

$$P(\mathcal{Z}|\mathcal{X}) = \prod_{f_i \in \mathcal{Z}} P(f_i(\mathcal{X}_i)) \tag{1.15}$$

其中 $\mathcal{X}_i$ 代表所有与约束节点 f_i 相邻的状态节点。通常会假设这是一个高度非线性系统，并且所有的约束 f_i 的噪声服从高斯分布 $f_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$。这样的概率最大化问题通常被转化为非线性最小二乘问题来求解：

$$\begin{aligned} \mathcal{X}^\star &= \arg\max_{\mathcal{X}} \prod_{f_i \in \mathcal{Z}} P(f_i(\mathcal{X}_i)) \\ &= \arg\min_{\mathcal{X}} \sum_{f_i \in \mathcal{Z}} \tfrac{1}{2} \left\| f_i(\mathcal{X}_i) \right\|^2_{\sigma_i^2} \end{aligned} \tag{1.16}$$

1.2.2 非线性最小二乘

非线性最小二乘问题难以被直接求解，通常要使用迭代求解的方法来逼近局部最优解。如果变量的初值在全局最优解的附近，则一般可以使用牛顿法（Newton's method）迭代求解。如下给出了而一个典型的非线性最小二乘问题 $F(\boldsymbol{x})$ 的形式：

$$F(\boldsymbol{x}) = \tfrac{1}{2} \| \boldsymbol{f}(\boldsymbol{x}) \|^2 \tag{1.17}$$

其中变量 $\boldsymbol{x} \in \mathbb{R}^n$，$\boldsymbol{f}(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^m$ 为关于变量 $\boldsymbol{x}$ 的非线性代价函数，即目标函数。在局部最优解附近，最小化能量函数 $F(\cdot)$：

$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}) \tag{1.18}$$

等同于求解其梯度函数 $\boldsymbol{g}(\cdot)$ 的零点（对于线性的函数则为唯一的零点），即：

$$\boldsymbol{g}(\boldsymbol{x}) = \nabla F(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial F}{\partial x_1} & \frac{\partial F}{\partial x_2} & \cdots & \frac{\partial F}{\partial x_n} \end{bmatrix}^\top = \boldsymbol{0} \tag{1.19}$$

采用牛顿法求解上述零点问题，记梯度函数 $\boldsymbol{g}(\cdot)$ 的一阶导数为海森矩阵（Hessian matrix）：

$$\mathrm{H} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & & & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix} \tag{1.20}$$

求解如下线性系统：

$$\boldsymbol{\delta} = \mathrm{H} \setminus \boldsymbol{\eta} \tag{1.21}$$

其中 $\boldsymbol{\eta} = -\boldsymbol{g}(\boldsymbol{x})$。使用 $\boldsymbol{\delta}$ 更新变量：$\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\delta}$。式(1.21)通常称为正规方程（normal equation），如上求解非线性最小二乘的方法则为正规方程法。线性的问题中，矩阵 $\mathrm{H}(\boldsymbol{x})$ 至少为半正定矩阵, 给定任意初值 $\boldsymbol{x}$（这里选择了初值为 $\boldsymbol{0}$）只需一步就可解得式(1.17)的全局最优解。而对于非线性的情况，则需要多次迭代才能解得局部最优解。

高斯-牛顿法

对于非线性最小二乘问题，牛顿法中计算函数 $F(\cdot)$ 的准确海森矩阵 H 的代价往往很大，甚至没有闭解。一些算法会寻求使用近似方法计算每次迭代的海森矩阵，比如拟牛顿法[2]（Qausi-Newton method）。拟牛顿法适用于一般的最小化问题，而对于非线性最小二乘问题，通常可以高斯-牛顿法（Gauss-Newton，简称 GN）求解。GN 法是用于求解非线性最小二乘问题的经典迭代算法，也是众多其他迭代算法的基础。

依然考虑非线性最小二乘问题(1.17)，使用 GN 法迭代求解，首先要给定变量的初值 $\boldsymbol{x} = \boldsymbol{x}_0$。对其在当前点处进行泰勒展开，并忽略高阶项，可得目标函数 $\boldsymbol{f}(\cdot)$ 在 $\boldsymbol{x}$ 处的线性近似：

$$\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{\delta}) \simeq \boldsymbol{f}(\boldsymbol{x}) + \mathrm{J}\boldsymbol{\delta} \tag{1.22}$$

其中 $\mathrm{J} \in \mathbb{R}^{m \times n}$ 是目标函数 $\boldsymbol{f}(\cdot)$ 在 $\boldsymbol{x}$ 处的雅各比矩阵（Jacobian）。相应地，

$$\begin{aligned} F(\boldsymbol{x}+\boldsymbol{\delta}) \simeq L(\boldsymbol{\delta}) &= \tfrac{1}{2}\|\boldsymbol{f}(\boldsymbol{x}) + \mathrm{J}\boldsymbol{\delta}\|^2 \\ &= F(\boldsymbol{x}) + \mathrm{J}^\top \boldsymbol{f} + \tfrac{1}{2}\boldsymbol{\delta}^\top \mathrm{J}^\top \mathrm{J}\boldsymbol{\delta} \end{aligned} \tag{1.23}$$

式(1.17)被转化为局部 $\boldsymbol{x}$ 处的线性最小二乘子问题，使用牛顿法求解这个线性最小二乘子问题

$$\min_{\boldsymbol{\delta}} \tfrac{1}{2}\|\boldsymbol{f}(\boldsymbol{x}) + \mathrm{J}\boldsymbol{\delta}\|^2 \tag{1.24}$$

需要构建正规方程：

$$\mathrm{H}_{gn} = \mathrm{J}^\top \mathrm{J}, \quad \boldsymbol{\eta} = -\mathrm{J}^\top \boldsymbol{f}(\boldsymbol{x}) \tag{1.25}$$

再求解，得到 GN 法一次迭代的结果：

$$\boldsymbol{\delta}_{gn} = \mathrm{H}_{gn} \setminus \boldsymbol{\eta} \tag{1.26}$$

将此结果更新至变量：$\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\delta}$，然后不断重复以上过程，直至结果收敛。

式(1.25)和(1.21)的形式类似，也可以认为 GN 法和牛顿法的最大区别就是 GN 法使用了 $\mathrm{J}^\top \mathrm{J}$ 来近似海森矩阵。

莱文贝格-马夸特方法

除了 GN 法之外，还有一种经典的求解非线性最小二乘优化的算法：莱文贝格-马夸特（Levenberg-Marquardt，简称 LM）法。它也被认为是带有阻尼因子的 GN 法，也是信赖域方法（trust region method）的前身[16]。

前面提到，一般的纯视觉的 SLAM 算法中，尺度是不可观测的，也就是说 VSLAM 的状态估计中，对全局所有的三维点坐标和相机位置进行统一的缩放，不会引起重投影误差的改变（在不考虑数值误差的情况下）。从求解 VSLAM 集束调整问题的角度来看，这一性质造成了在直接使用 GN 法时海森矩阵的秩亏现象，即矩阵 $\mathrm{J}^\top \mathrm{J}$ 为半正定的情况。此时直接使用矩阵分解求逆的方法求解线性最小二乘子问题就会得到数值不稳定的结果，比如解得的步长在某个方向过大等。

针对这一情况，LM 法在基础的 GN 法中引入了对迭代步长的直接约束，根据海森矩阵的主元大小，对步长的不同梯度方向施加不同的阻尼：

$$\mathrm{H} = \left(\mathrm{J}^\top \mathrm{J} + \mu \mathbf{diag}(\mathrm{J}^\top \mathrm{J})\right), \quad \boldsymbol{\eta} = -\mathrm{J}^\top \boldsymbol{f}(\boldsymbol{x}) \tag{1.27}$$

$$\boldsymbol{\delta}_{lm} = \mathrm{H} \setminus \boldsymbol{\eta} \tag{1.28}$$

其中的 μ 就是所谓的阻尼因子（damping factor），其控制住了每次迭代计算的步长的模。通常阻尼因子 μ 具备下面三个性质[2]：

1. 对于任意的 $\mu > 0$，海森矩阵 $(\mathrm{J}^\top\mathrm{J} + \mu\mathbf{diag}(\mathrm{J}^\top\mathrm{J}))$ 正定，这一点保证了求得的步长 $\boldsymbol{\delta}_{lm}$ 处于能量下降的方向；

2. 当 $\mu \to \infty$ 时，求得的步长 $\boldsymbol{\delta}_{lm}$ 接近于最速下降方向，如果当前的变量 $\boldsymbol{x}$ 的值距离最优解较远时，最速下降的方向更容易符合预期；

3. 当 $\mu \to 0$ 时，则求得的步长 $\boldsymbol{\delta}_{lm}$ 更接近于 GN 法的结果，如果当前变量 $\boldsymbol{x}$ 的值较为接近最优解，则这样的步长更容易符合预期，因为在收敛点附近，真实的海森矩阵更接近标准的二次型形式。

同时，由于正则项 $\mu\mathbf{diag}(\mathrm{J}^\top\mathrm{J})$ 的加入，修改后的海森矩阵 $(\mathrm{J}^\top\mathrm{J} + \mu\mathbf{diag}(\mathrm{J}^\top\mathrm{J}))$ 一定为正定二次型矩阵，提升了求解的数值稳定性。

Dog-Leg 法

Dog-Leg 法（简称 DL）是另一个经典的求解非线性最小二乘优化的算法，也是信赖域方法的代表。LM 法通过阻尼因子来间接地控制每一次迭代的步长，而 DL 法则显式地使用了信赖域来约束每一次迭代的步长。下面给出传统的使用 DL 法求解问题(1.17)的步骤[2]。首先按照常规的求解步骤进行线性化，得到线性最小二乘子问题(1.24)。DL 法首先使用式(1.26)求解 GN 迭代步 $\boldsymbol{\delta}_{gn}$，然后使用最速下降法求解一阶迭代步：

$$\boldsymbol{\delta}_{sd} = -\frac{\|\boldsymbol{g}\|^2}{\|\mathrm{J}\boldsymbol{g}\|^2}\boldsymbol{g} \tag{1.29}$$

和 LM 法类似，DL 法也是一种结合了 GN 法和最速下降法的方法。如式(1.30)所示，

$$\boldsymbol{\delta}_{dl} \coloneqq \begin{cases} \boldsymbol{\delta}_{gn}, & \|\boldsymbol{\delta}_{gn}\| \le \Delta \\ \frac{\Delta}{\|\boldsymbol{\delta}_{sd}\|}\boldsymbol{\delta}_{sd}, & \|\boldsymbol{\delta}_{sd}\| \ge \Delta \\ \boldsymbol{\delta}_{sd} + \beta(\boldsymbol{\delta}_{gn} - \boldsymbol{\delta}_{sd}), & \|\boldsymbol{\delta}_{gn}\| > \Delta, \|\boldsymbol{\delta}_{sd}\| < \Delta \end{cases} \tag{1.30}$$

DL 法根据当前的信赖域半径 Δ 和以上求解的两个步长来计算综合的迭代步 $\boldsymbol{\delta}_{dl}$：

1. 如果 GN 迭代步 $\boldsymbol{\delta}_{gn}$ 的模小于信赖域半径，则认为可以直接选择 $\boldsymbol{\delta}_{gn}$；

2. 如果 GN 迭代步 $\boldsymbol{\delta}_{gn}$ 的模和最速下降迭代步 $\boldsymbol{\delta}_{sd}$ 都大于信赖域半径，则选择 $\boldsymbol{\delta}_{sd}$ 并将其缩放到信赖域大小；

3. 如果 GN 迭代步 $\boldsymbol{\delta}_{gn}$ 的模大于信赖域半径而最速下降迭代步的模小于信赖域半径，则需要取 $\boldsymbol{\delta}_{gn}$ 和 $\boldsymbol{\delta}_{sd}$ 连线与信赖域的交点作为迭代步（如图 1.2）。

这样，当 Δ 较大时，DL 法更倾向于使用 GN 法的迭代步 $\boldsymbol{\delta}_{gn}$；当 Δ 较小时则更倾向于最速下降法的迭代步 $\boldsymbol{\delta}_{sd}$。算法 1详细描述了 DL 法的迭代过程。和 GN 法一样，DL 法在求解式(1.26)的时候也可能遇到海森矩阵 H 半正定的情况，此时应该选择数值稳定的线性求解方法来求解海森矩阵的伪逆 $\mathrm{H}^{\dagger}$。

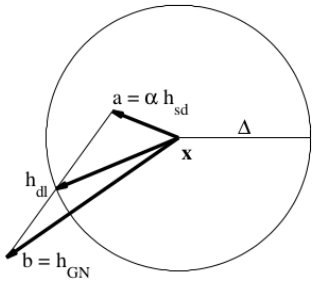

图 1.2 信赖域与 DL 法迭代步 [2]

1.2.3 舒尔补

不论是使用 GN 法，LM 法还是 DL 法，在每一轮迭代过程中，都需要进一步通过求解正规方程这个线性系统来得到每一轮迭代的步长（如式(1.26)，(1.28)，(1.30)）。根据经验，SLAM 问题具有良好的稀疏性：通常三维点状态的数量要远大于相机状态的数量，而且通常三维点之间没有直接联系；同时，只有物理上距离比较接近的三维点状态和相机状态之间有相互约束。因此，由集束调整问题构建的海森矩阵 H 也具有稀疏的性质，通过对变量适当的排序，将三维点状态和相机状态分别分组，一般可以得到如图 1.3所示的特殊的稀疏结构。

对于这样具备特殊稀疏结构的线性系统，可以用舒尔补 (Schur complement) 来加速求解。舒尔补是一种常用的加速求解稀疏线性系统的方法，特别适用于集束调整问题中的正规方程的求解。其本质上是一种基于高斯消元的分块求解线性系统的方法。首先根据图 1.3对正规方程进行分块：

$$\mathrm{H} = \begin{bmatrix} \mathrm{P} & \mathrm{W}^{\top} \\ \mathrm{W} & \mathrm{C} \end{bmatrix}, \quad \boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}_l \\ \boldsymbol{\delta}_c \end{bmatrix}, \quad \boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{l} \\ \boldsymbol{c} \end{bmatrix} \tag{1.31}$$

其中 P 是三维点状态对应的海森矩阵，C 是相机状态对应的海森矩阵，W 是三维点状态和

算法 1 Dog-Leg 法

输入: 初始信赖域半径 Δ_0，初值 $\boldsymbol{x}_0$

输出: 优化结果 $\boldsymbol{x}$

$\Delta \coloneqq \Delta_0, \boldsymbol{x} \coloneqq \boldsymbol{x}_0$

for $k = 0 \to k_{max}$ **do**

$k \coloneqq k + 1$

使用式(1.24)计算线性化子问题

使用式(1.26)计算高斯-牛顿迭代步 $\boldsymbol{\delta}_{gn}$

使用式(1.29)计算最速下降迭代步 $\boldsymbol{\delta}_{sd}$

使用式(1.30)计算 DL 法步长 $\boldsymbol{\delta}_{dl}$

if {步长 $\boldsymbol{\delta}_{dl}$ 收敛} **then**

结束优化

end if

计算迭代步质量 ϱ:

$$\varrho = \frac{F(\boldsymbol{x}) - F(\boldsymbol{x} + \boldsymbol{\delta}_{dl})}{L(\mathbf{0}) - L(\boldsymbol{\delta}_{dl})}$$

if $\{\varrho > 0.0\}$ **then**

$\boldsymbol{x} \coloneqq \boldsymbol{x} + \boldsymbol{\delta}_{dl}$

end if

if $\{\varrho > \epsilon_0\}$ **then**

$\Delta \coloneqq \mathrm{fmax}\{\Delta, 3\,\|\boldsymbol{\delta}_{dl}\|\}$

else if $\{\varrho < \epsilon_1\}$ **then**

$\Delta \coloneqq \Delta/2$

end if

if {信赖域 Δ 收敛} **then**

结束优化

end if

end for

相机状态的海森矩阵。求解三维点状态和相机状态之前，先通过行变换构建舒尔补方程:

$$\left(\mathrm{C} - \mathrm{W}\mathrm{P}^{-1}\mathrm{W}^{\top}\right)\boldsymbol{\delta}_c = \boldsymbol{c} - \mathrm{W}\mathrm{P}^{-1}\boldsymbol{l} \tag{1.32}$$

记舒尔补矩阵为 $\mathrm{S} \doteq \mathrm{C} - \mathrm{W}\mathrm{P}^{-1}\mathrm{W}^{\top}$，右侧为 $\boldsymbol{b} \doteq \boldsymbol{c} - \mathrm{W}\mathrm{P}^{-1}\boldsymbol{l}$。如前面提到的，集束调

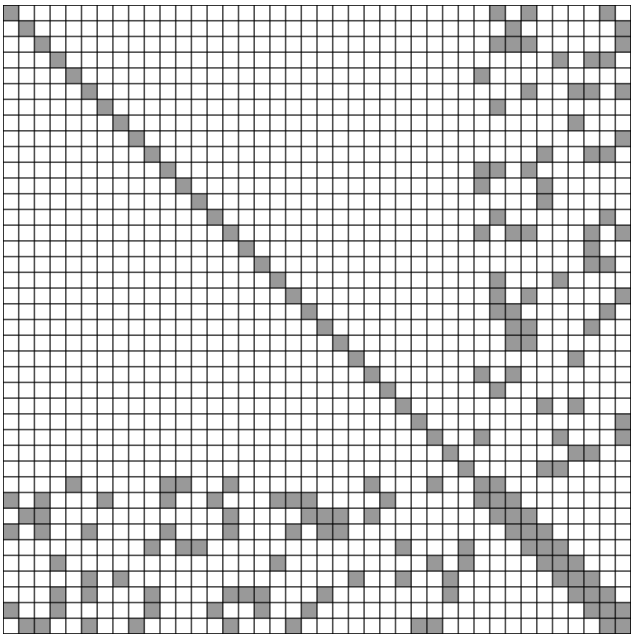

图 1.3 一般的稀疏矩阵结构：左上角部分为三维点状态对应的海森矩阵，呈对角块稀疏状；右下角部分为相机状态对应的海森矩阵，呈条带稀疏状；左下角矩阵为三维点状态和相机状态的相关海森矩阵，只有具有观测关系的三维点和相机之间的相关部分为非零。

整问题中的三维点状态数量远多于相机状态数量，故矩阵 S 的规模远小于 P 的规模，通过求解下舒尔补方程：

$$\boldsymbol{\delta}_c = \mathrm{S} \setminus \boldsymbol{b} \tag{1.33}$$

可以快速先得到相机状态的迭代步。而又因为矩阵 P 通常呈块对角状，其逆矩阵的求解 P^{-1} 也非常容易，通过变量回代又可以高效地求解剩余的三维点状态的迭代步：

$$\boldsymbol{\delta}_l = \mathrm{P} \setminus \left(\boldsymbol{l} - \mathrm{W}^{\top}\boldsymbol{\delta}_c\right) \tag{1.34}$$

舒尔补可以加速稀疏线性系统的求解，但需要注意的是，舒尔补一般会造成矩阵的填充效应 (fill in)[3]，即破坏矩阵稀疏性的情况。在使用舒尔补的时候如果不注意变量的排序，则很容易造成严重的填充。

1.2.4 边缘状态估计

对于比较大的系统，状态数量比较多，一般需要通过边缘化对状态进行精简。在图优化中对应的就是通过删减状态节点调整因子图的过程。假设状态 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 符合联合高斯分布：

$$p(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\Sigma_{xx} & \Sigma_{yx}^{\top} \\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}\right) \tag{1.35}$$

对于高斯分布，总是可以使用舒尔补将上面的联合概率分布拆分成两个独立分布的乘积 $p(\boldsymbol{x},\boldsymbol{y}) = p(\boldsymbol{x}|\boldsymbol{y})p(\boldsymbol{x})$：

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{y}) &\sim \mathcal{N}(\boldsymbol{\mu}_x + \Sigma_{yx}^{\top}\Sigma_{yy}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_y), \Sigma_{xx} - \Sigma_{yx}^{\top}\Sigma_{yy}^{-1}\Sigma_{yx}) \\ p(\boldsymbol{y}) &\sim \mathcal{N}(\boldsymbol{\mu}_y, \Sigma_{yy}) \end{aligned} \tag{1.36}$$

其中 $p(\boldsymbol{y})$ 就是变量 $\boldsymbol{y}$ 的边缘概率，而 $p(\boldsymbol{x}|\boldsymbol{y})$ 则是变量 $\boldsymbol{x}$ 关于 $\boldsymbol{y}$ 的条件概率。在最小二乘问题中，正规方程的海森矩阵 H 和右侧 $\boldsymbol{\eta}$ 对应的是信息矩阵（information matrix）和信息向量（information vector）[17]。信息矩阵也对应于状态的协方差矩阵的逆 Σ^{-1}，而正规方程的解对应与变量的期望。以状态 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 为例，可以构建正规方程：

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \mathrm{H}_{xx} & \mathrm{H}_{yx}^{\top} \\ \mathrm{H}_{yx} & \mathrm{H}_{yy} \end{bmatrix} \setminus \begin{bmatrix} \boldsymbol{\eta}_x \\ \boldsymbol{\eta}_y \end{bmatrix} \tag{1.37}$$

易得联合概率分布：

$$p(\boldsymbol{x},\boldsymbol{y}) \sim \mathcal{N}\left(\begin{bmatrix} \mathrm{H}_{xx} & \mathrm{H}_{yx}^{\top} \\ \mathrm{H}_{yx} & \mathrm{H}_{yy} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\eta}_x \\ \boldsymbol{\eta}_y \end{bmatrix}, \begin{bmatrix} \mathrm{H}_{xx} & \mathrm{H}_{yx}^{\top} \\ \mathrm{H}_{yx} & \mathrm{H}_{yy} \end{bmatrix}^{-1} \right) \tag{1.38}$$

根据式(1.36)，同样使用舒尔补可以计算得到状态 $\boldsymbol{y}$ 的边缘概率分布：

$$p(\boldsymbol{y}) \sim \mathcal{N}\Big(\underbrace{(\overbrace{\mathrm{H}_{yy} - \mathrm{H}_{yx}\mathrm{H}_{xx}^{-1}\mathrm{H}_{yx}^{\top}}^{\text{边缘信息矩阵}})^{-1}(\overbrace{\boldsymbol{\eta}_y - \mathrm{H}_{yx}\mathrm{H}_{xx}^{-1}\boldsymbol{\eta}_x}^{\text{边缘信息向量}})}_{\text{边缘期望}}, \underbrace{(\mathrm{H}_{yy} - \mathrm{H}_{yx}\mathrm{H}_{xx}^{-1}\mathrm{H}_{yx}^{\top})^{-1}}_{\text{边缘协方差}} \Big) \tag{1.39}$$

如 1.2.3 节中提到的，舒尔补一般会造成稀疏矩阵的填充效应，在使用舒尔补进行边缘化操作的时候，同样会遇到这样的情况。因此，在使用边缘化进行状态删减的时候，也需要注意使用合理的策略选择被删减的状态，尽量减少填充。

1.3 SLAM 状态估计相关工作

VSLAM 和 VISLAM 都需要根据系统的先验估计、传感器观测来对系统的状态进行估计，可能包括实时的位姿状态、传感器状态、场景三维点状态以及其他可能的信息。主流的 SLAM 方法根据状态估计的方法可以分为基于扩展卡尔曼滤波（Extended Kalman Filter，简称 EKF）的滤波法和基于非线性最小二乘（Nonlinear Least Squares）的优化法。两类方法没有本质上的区别，都是使用最大化后验概率（Maximum A-Posteriori，简称 MAP）的

理论求解非线性系统的最优状态估计，但是在求解的速度、精度以及算法的可扩展性上存在一定差异。不同的 SLAM 系统除了使用了不同的状态估计方法，其估计的状态类型以及使用的目标函数、变量参数化方法都有所不同，造成了其精度和效率的差异。本节将简要列举并分析基于滤波法和优化法的 SLAM 系统中的状态估计策略，以及分析不同 SLAM 系统中为提升计算效率所做的工作。

1.3.1 基于滤波法的 SLAM 状态估计

一般来讲滤波法的时间复杂度小于同规模的优化法，早期由于计算能力的限制，选择滤波法进行状态估计是一种合理的选择。MonoSLAM[18] 是最早的单目 VSLAM 系统之一。MonoSLAM 使用了 EKF 求解相机状态和三维点状态，也是最早的滤波法 VSLAM 系统之一。在状态估计阶段，MonoSLAM 快速地将旧的相机状态通过边缘化操作消去，将系统的状态数量限制在 $O(N)$，其中 N 为三维点的数量，同时状态估计的时间复杂度被限制在了 $O(N^3)$。因此 MonoSLAM 的运行时间得到了限制，不至于随着时间无限增长。但由于 MonoSLAM 过早地将相机状态进行边缘化，一部分尚未收敛的相机状态就将错误的信息留在了系统的状态中，导致了较大的误差累积。另一方面，频繁的边缘化操作也就导致了系统状态的协方差矩阵变得稠密，无法利用稀疏的求解方法进行加速，这一点也是基于 EKF 的算法的通病。

另一个经典的基于 EKF 的 SLAM 系统是 MSCKF[19]。MSCKF 全称是多状态约束卡尔曼滤波（Multi-State Constraints Kalman Filter）。与 MonoSLAM 不同的是，MSCKF 使用了相机状态窗口的概念，计算大小为 M 的系统的 EKF 更新，其中 M 为状态窗口的大小。因此 MSCKF 将状态估计的实现复杂度限制在了 $O(M^3)$。由于在 SLAM 问题中，三维点的数量往往远远大于相机状态的数量：$N \gg M$，MSCKF 算法可以达到实时性的要求，效率要高于 MonoSLAM。又由于 MSCKF 保留了一定数量的历史相机状态，而不是尽早地将旧的状态消去，因此误差累积更小，更适合长时间、大尺度的应用。

如前面描述的，MSCKF 算法是无三维结构信息的，也就是只保留相机状态，而不估计三维点的状态。在状态估计前先根据一定的策略选取一部分视觉特征，快速地通过三角化计算出对应的三维点状态，然后快速地通过边缘化操作消去，最后再对相机状态进行更新。

MSCKF 求解的状态估计仍然不是全局最优的。前面提到，被消去的状态的信息要先以边缘化的方式将保留下来，即得到剩余状态的边缘概率分布。一部分剩余状态的雅各比矩阵，其线性化点的被永远固定在了发生边缘化的时刻。后续即使状态值发生了改变，旧的雅各比矩阵由于已经被编码进了先验状态分布中而不能再改变，而随后得到的新的观测信息

总是会根据最新的状态值计算雅各比矩阵，这就造成了所谓的信息不一致（inconsistency）。客观上，信息不一致造成了 MSCKF 的状态估计在某些不可观测的自由度上引入了一些不必要的误差。为了解决信息一致性的问题，后续版本的 MSCKF 2.0[20] 引入了 FEJ 技术（First Estimate Jacobian）技术[21]，在计算新的观测关于旧的变量的雅各比矩阵时，总是使用状态边缘化发生时刻的线性化点，一定程度上缓解了这个问题，提高了精度。另一些侧重于解决信息不一致性的 SLAM 相关工作有通过选用特殊的线性化点的方式保持了系统的不可观测的自由度的 [22] 和 [23] ，以及使用了一个变种 EKF 算法弥补错误不可观测自由度的 [24] 。

1.3.2 基于集束调整优化的 SLAM 状态估计

随着硬件能力的发展，状态优化的计算效率已经越来越不再是 SLAM 系统的瓶颈所在。一方面，由于优化法和滤波法并没有时间复杂度上的差异，也有许多算法能够利用 SLAM 问题的稀疏性和局部性减少优化的计算量，比如使用舒尔补、稀疏矩阵分解甚至增量式优化的方法。一些高效的基于优化法的 SLAM 系统已经能在效率上逼近甚至超越基于滤波法的 SLAM 系统。另一方面，优化法在精度上相对于滤波法有着不可比拟的优势。尽管有很多的手段提升滤波法的精度，但是由于难以求解全局最优解，滤波法仍然会有较大的误差累积。前面提到，滤波法和优化法在概率上都是 MAP 估计，由于每次求解状态估计，EKF 都只做一次线性化并进行一次更新，而优化法则通过迭代不断更新线性化点求解直至收敛，因此滤波法可以认为是使用了一轮迭代的优化法。显而易见，相比于优化法，滤波法虽然速度更快，但不保证状态收敛。而且基于优化法的 SLAM 可以在基础的视觉观测约束、IMU 约束上其他约束，进一步消除误差累积，基于滤波法的 SLAM 则在这方面的可扩展性上稍差。因此，越来越多的主流 SLAM 系统已经在使用基于优化法的状态估计。

PTAM[25] 是最经典的基于优化法的 VSLAM 系统的代表。在系统架构上，PTAM 创新性地将局部的相机跟踪定位和全局优化分发到两个线程中运行。在前端相机跟踪线程中，PTAM 对系统的移动速度进行了建模，使用估计的速度预测相机的位姿。借助于预测的位姿，可以得到旧的视觉特征点在新的相机图像中的投影，从而缩小特征匹配时的搜索范围。给定特征匹配，通过最小化重投影误差可以得到相机位姿的粗略估计。由于前端线程只求解位姿，这一步可以实时运行。当前端线程还会筛选出质量较高的图像帧作为关键帧（keyframe）加入到后端全局优化线程中。而在后端全局优化线程中，PTAM 使用了集束调整来求解全局的地图。全局地图的求解时间会随关键帧的增长而呈 $O(M^3)$ 级别的增长，受限与此，PTAM 将关键帧的数量限制在了 100 帧。但即便如此，将相机跟踪和全局优化分

发到不同的线程中执行的策略被证明可以大幅提高基于优化法的 SLAM 的性能，在随后的 SLAM 研究工作中，这种策略被大量借鉴。目前的主流 SLAM 系统大多数都使用了多线程的方法提高系统的效率。

ORB-SLAM[26;27] 是另一个经典的 VLSAM 工作，也是目前最先进的 SLAM 系统之一。ORB-SLAM 使用了 ORB 特征来提升跟踪质量。在 PTAM 的基础上，ORB-SLAM 还额外利用了全局地图的信息，加入了重定位和和回路闭合模块。同样的，ORB 也使用了多线程架构：局部相机跟踪、地图优化和回路闭合被分发到三个不同的线程中执行。局部相机跟踪线程维护了一个由关键帧组成的滑动窗口，会适时地优化最新关键帧和与其具有共视关系的一系列关键帧。由于局部窗口大小固定，局部跟踪线程的计算时间被限制在一个可接受的水平。为了尽量避免全局优化，在回路闭合线程中，ORB-SLAM 使用了最小生成树建立的 Essential Graph 的方法来构建关键帧之间的回路约束。回路闭合的过程中也没有包括三维点状态的求解，Essential Graph 以较低的计算开销获取了较好的回路闭合性能。

OKVIS[28] 是另一个基于优化法的 SLAM 系统。同时还是 VISLAM 的经典框架之一。与 ORB-SLAM 不同的是，OKVIS 仅维护了一个局部的滑动窗口优化。由于不包含全局地图优化，OKVIS 并不具备回路闭合和重定位能力。OKVIS 在相机跟踪部分使用了和 MSCKF 类似的迭代式 IMU 积分技术，随后通过滑动窗口优化，同时最小化 IMU 运动误差和重投影误差，得到包含比较准确的尺度信息的状态估计结果。同时，OKVIS 在保持优化的稀疏性的前提下使用了包含边缘化的滑动窗口优化，以提升优化的精度。滑动窗口会随着时间增长逐渐滑动，滑出窗口的状态一般需要进行边缘化，为了尽可能保持优化的稀疏性，OKVIS 会根据滑出窗口的是否为关键帧来判断是否要进行完整的边缘化操作。

VINS-Mono[29] 则是最新的一个较为完整的 VISLAM 系统实现。相对于以上的 SLAM 系统，VINS-Mono 的框架更为完整，包含了鲁棒的初始化模块、包含边缘化的滑动窗口优化模块、重定位模块和全局优化模块。同样，这些模块被分发到了不同的线程中执行，以提升效率。

VINS-Mono 使用了一个松耦合的方式对系统的状态进行初始化。在初始化阶段，VINS-Mono 使用视觉 SfM 方法由一系列初始帧构建一个小规模的地图，然后通过与独立的 IMU 积分算法构建的初始位姿进行对齐，获取初始的地图状态和位姿状态，以及一系列初始的 IMU 状态参数。在滑动窗口模块，VINS-Mono 首先通过 KLT 跟踪获取每一帧的初始位姿估计。在优化部分，VINS-Mono 借鉴了 OKVIS 的做法，使用稀疏求解器求解状态估计，同时使用了类似 OKVIS 的选择性边缘化策略来保持优化的稀疏性。不同的是，VINS-Mono 使用了的基于 IMU 预积分[18] 技术，而不是传统的迭代式 IMU 积分，以提升 IMU 状态的

优化效果。对于滑出窗口的状态，如果判断为关键帧，则会被加入到后端的全局优化中。全局优化运行于独立的线程，VINS-Mono 使用了位姿图优化的方式对这些历史关键帧进行进一步更新，提供给回路闭合模块使用。VINS-Mono 在系统的功能、速度和精度上达到了较高的水准。而且其移动端的版本 VINS-Mobile[30] 经过精简，可以在移动设备上达到实时的性能。

以上介绍的 SLAM 系统均基于特征法。另一类直接法 SLAM 系统近年来也获得了较大的关注。特征法和直接法也没有绝对的优劣之分，特征法利用了图像中的关键信息，因此通常对于几何误差（如相机内参的误差）和图像噪声（如光照变化、卷帘快门相机的图像撕裂）更为鲁棒，但是关键点的提取和匹配通常比较耗时；直接法不需要提取和匹配关键点，因此速度较快，但对图像噪声较为敏感；另一方面，由于直接法通常利用了更多图像信息，因此在弱纹理情况下表现更好，而且其构建的地图往往比较稠密，可以进一步提供给三维重建算法使用。

DSO（Direct Sparse Odometry）[31] 是目前最先进的直接法 VSLAM 系统之一。DSO 使用了类似[32] 提出的稀疏直接法进行跟踪，这一点与以往的大部分直接法 VSLAM 都不同。DSO 也使用了优化法进行状态估计。除了传统直接法中使用的光度误差，DSO 还对场景中的光照进行建模，提出了使用曝光时间、镜头晕影（lens vignetting）和非线性响应函数。这一改进使得 DSO 在光照变化的场景下具有更好的精度和鲁棒性。类似 OKVIS，DSO 也使用了包含边缘化的滑动窗口优化方法。为了保证滑动窗口中的相机状态在三维空间中有较好的分布，DSO 设计了一个特殊设计的评分函数对关键帧进行评分。当状态数量达到上限时，DSO 会根据评分挑选需要消去的相机状态。为了保持稀疏性，DSO 也会有选择地对消去的状态进行边缘化。

在实现 SLAM 算法时，需要在算法的精度和性能两者中做出权衡。完整的集束调整包括了对所有历史状态的所有观测，虽然可以获得最优的状态估计，但其计算代价往往是难以接受的。在一些对性能要求高的应用场景，例如 AR 应用和自动驾驶应用中，算法的性能往往决定了它的可用性甚至安全性。随着这类应用对实时的、高效 SLAM 算法需求的日益增加，一些致力于在保证一定精度的前提下降低计算代价的集束调整算法应运而生，其主要通过两种策略来减少算法的计算量。一类是针对应用场景的特点减小集束调整问题的规模，基于以上 SLAM 系统的介绍，可以总结出以下几种方式：

- **减小全局优化的规模：**只保留状态变量，而不保留三维点变量，如 ORB-SLAM 的 Essential Graph 和 VINS-Mono 的位姿图优化；
- **基于历史状态窗口：**只估计最近的数个历史状态或一系列选定的数个历史状态，OKVIS、

VINS-Mono 等的滑动窗口优化；

- **基于关键帧：**只估计一部分选定的携带了足够信息的历史状态，而放弃一些冗余历史状态，如 OKVIS、VINS-Mono 等。

以上的方法通常也可以结合，或搭配多线程技术使用。

1.4 增量式集束调整方法

除了减小集束调整问题的规模，另一中加速的策略是通过深入分析集束调整问题的特点，做针对性优化，减少冗余的计算。本节将介绍基于增量式集束调整算法的 SLAM 系统相关工作。

集束调整问题通常具有非常特殊的性质，合理利用这些性质，可以帮助更高效地求解 SLAM 系统的状态估计。比如，在 SLAM 系统中，通常需要求解的三维点变量的数量要远大于状态变量的数量，而且通常状态约束集中在状态与状态之间、状态与三维点之间，而一般不会直接出现在三维点与三维点之间。这就导致集束调整构建的正规方程具有特别的稀疏结构，如图 1.3。应该利用这种稀疏结构，并且在优化的过程中尽可能保持这种稀疏性。另外，在线的集束调整中，通常只有较新加入的变量会发生比较大的变动，旧的变量由于经过持续的优化而变动很小。可以利用这种局部的性质，只对少部分变量进行更新，从而减少集束调整的计算量。

以 iSAM 系列（主要包括 iSAM[3] 和 iSAM2[1]）为例，增量式集束调整算法很好地利用了 SLAM 集束调整问题的稀疏性和局部性，实现了高效的求解。iSAM 系列算法提出，集束调整算法中的矩阵分解过程等同于使用消元法将因子图转化成贝叶斯网络的过程；分解时产生的稀疏矩阵填充现象对应于因子图转化过程中新增的边；并且在该过程中选择的变量顺序会极大地影响矩阵填充的程度。iSAM 系列算法通过贝叶斯网络进一步生成了一种特殊的贝叶斯推断树结构，使用贝叶斯推断树编码并维护了矩阵分解得到的平方根信息矩阵，使得每次更新平方根信息矩阵时只要做很少的改动即可。iSAM 系列算法的主要策略如下：

1. **减少填充现象：**通过一些启发式算法如 COLAMD[33]、CHOLMOD[34] 算法找到次优的变量消去顺序，使得因子图转化为贝叶斯置信网络的过程中尽可能地不产生新的边（矩阵填充）；

2. **使用贝叶斯推断树编码平方根信息矩阵：**对于因子图转化而来的贝叶斯置信网络，使用特殊的贝叶斯推断树来编码其中的稀疏结构和变量因果关系；

3. **即时重新线性化：**利用贝叶斯推断树中的编码的变量因果关系，每当因子图中加入新的变量或约束，则即时地对影响到的变量进行线性化，避免每次都重新构造整个平方根信息矩阵；

4. **局部变量更新策略：**通过设置一个阈值 ϵ，在求解得到变量的增量 Δ 后，根据其是否满足 $|\Delta| > \epsilon$ 来判断是否需要更新这个变量。

这样，维护一个增量求解的集束调整算法的时间复杂度就会远低于一般的完整的集束调整，从而在不损失精度或损失很少的精度情况下将 SLAM 算法的速度提升一个甚至数个数量级。

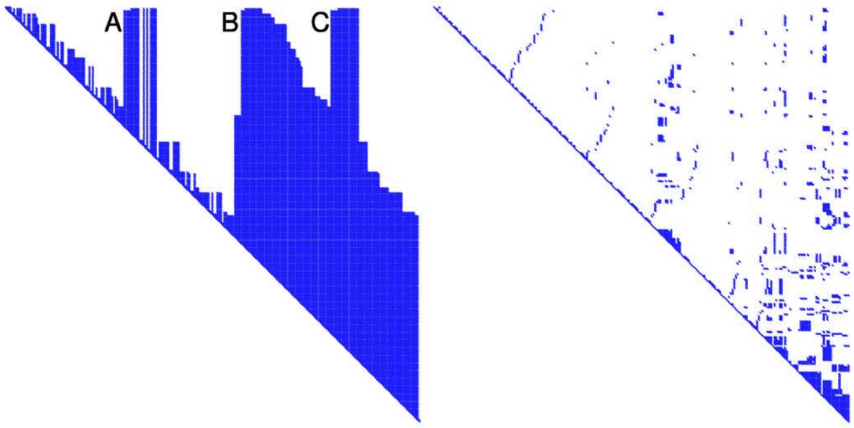

图 1.4 信息矩阵稀疏结构 [3]：左图代表按照一般的消元顺序得到的平方根信息矩阵，右图代表先使用 COLAMD 算法选择一个较优的消元顺序，然后得到的平方根信息矩阵。可见在矩阵分解时，不同的主元顺序（pivoting）对应于因子图转化过程中变量的消去顺序，其所产生的填充现象的程度也有显著的差异。

另一类增量式集束调整算法是以 ICE-BA 和 SLAM++ 为代表的增量式舒尔补方法。如 1.2.3 节中提到的，舒尔补显式地先消去三维点变量，而不是类似 iSAM 系列算法使用程序自动生成消元的顺序。另一方面，增量式舒尔补方法也没有使用贝叶斯推断树或类似的结构编码消元的过程。在线性求解部分，ICE-BA 提出了 I-PCG 算法加速舒尔补方程的求解，然后通过完整的回代算法求解三维点部分。而 SLAM++ 则通过批量式 Cholesky 分解求解舒尔补方程，然后通过完整回代算法求解三维点部分。此外，ICE-BA 还提出了 sub-track 方法来进一步稀疏化舒尔补矩阵，并使用了相对边缘化 (relative marginalization) 方法来使滑动窗口集束调整和全局集束调整保持一致。

1.5 本文内容及结构

本章总结了 VSLAM、VISLAM 工作中关于状态估计的研究现状。对基于滤波方法和基于非线性优化方法的状态估计进行了解释。阐述了当前 SLAM 领域的常用的基于滤波法和基于集束调整优化的状态估计方法，列举并分析了这些 SLAM 系统中状态估计方法的优缺点，并简要介绍了目前最高效的增量式集束调整方法的思想。接下来本文将着重介绍一个面向 VISLAM 的通用集束调整框架的实现。第 2 章将详细介绍基于增量式舒尔补的集束调整框架，包括该框架的设计，以及内置提供的常用目标函数实现。然后介绍增量式舒尔补的基本算法以及本文提出的 HLMDL 算法和基于贝叶斯推断树的 PBT 算法，并分析与其他增量式算法之间的区别；第 3 章通过实验结果评估求解效率，进一步验证本文提出的增量式集束调整框架的效率和精度；第 4 章给出本文的结论以及后续工作的展望。

第 2 章 基于增量式舒尔补的集束调整框架

本章将详细介绍基于增量式舒尔补的集束调整框架。首先第 2.1 节中简要介绍该框架的设计和内置提供的目标函数；在第 2.2 节中介绍基于因子图的增量式舒尔补构建方法；在第 2.3 节中，针对 ICE-BA、SLAM++ 算法中常见的舒尔补矩阵秩亏的问题，介绍本文提出的 HLMDL 算法，在不破坏舒尔补构建的增量特性的前提下增强线性解的数值稳定性。然后将介绍本文提出的基于贝叶斯推断树的 PBT 回代算法，以及如何使用 PBT 回代算法进行增量回代求解，并保证变量的一致性；第 2.4 节将总结本文提出的增量式框架与其他增量式算法之间的区别。

2.1 增量式集束调整框架

为了保证通用性，本文将集束调整的框架划分为因子图、非线性策略、线性求解器三个模块：

- **因子图：**此模块对应整个集束调整问题，包括所有目标函数因子和状态变量，以及求解所需的块状稀疏矩阵数据结构。每一个因子数据结构存储了目标函数对应的协方差矩阵和残差、雅各比矩阵计算函数。用户可以使用内置的 IMU 预积分目标函数和重投影误差目标函数，也可以通过重写相关的虚函数来加入自定义的目标函数。每一个状态变量数据结构包含了状态变量的数值和对应的参数化方法，用户可以使用内置的旋转矩阵或四元数参数化方法，也可以通过重写相关的虚函数来加入自定义的参数化方法。另外，因子图还保存了目标函数和状态变量之间的关系，即因子图的边。

- **非线性策略：**此模块对应集束调整求解过程中的线性化过程和舒尔补过程，以及变量回代求解的过程。此模块内置了本文提出的 HLMDL 算法和 PBT 回代算法，用户可以通过配置相关的选项，改变判断变量是否发生变化的阈值，也可以选择关闭对应的选项，只使用常规的批量式求解策略，如 LM 方法、DL 方法等。

- **线性求解器：**此模块对应舒尔补方程的线性求解策略，内置了块状稀疏矩阵版本和稠密版本的 Cholesky 分解、QR 分解和 I-PCG 求解器。用户可以通过配置相关的选项来选择符合需求的线性求解器。此外，线性求解的接口也是开放的，用户也可以自行实现符合具体数值需求的线性求解器。

2.1.1 目标函数

本框架提供了基于 [11] 提出的 IMU 预积分技术的目标函数。与基于传统迭代式 IMU 积分的 VISLAM 系统不同，IMU 预积分使用了更精确的相对运动模型。将 IMU 观测模型包含三个部分：相对旋转 $\Delta\mathrm{R}$、相对速度 $\Delta\boldsymbol{v}$、相对平移 $\Delta\boldsymbol{p}$，可以认为它们是仅关于 bias $\boldsymbol{b}^g$ 和 $\boldsymbol{b}^a$ 的函数。

记 i 时刻相机-IMU 状态为：

$$\boldsymbol{X}_i \doteq \begin{bmatrix} \mathrm{R}_i & \boldsymbol{v}_i & \boldsymbol{p}_i & \boldsymbol{b}_i^g & \boldsymbol{b}_i^a \end{bmatrix} \tag{2.1}$$

分别代表该时刻系统在全局坐标系下的朝向、速度、位置以及系统自身坐标系下的角速度 bias 和加速度 bias。那么状态 $\boldsymbol{X}_i$ 时刻和 $\boldsymbol{X}_j$ 时刻之间的 IMU 预积分目标函数为：

$$\begin{aligned}
\boldsymbol{r}_{\Delta\mathrm{R}_{ij}} &\doteq \log\left(\left(\Delta\bar{\mathrm{R}}_{ij}\exp\left(\tfrac{\partial\Delta\bar{\mathrm{R}}_{ij}}{\partial\boldsymbol{b}_i^g}\delta\boldsymbol{b}_i^g\right)\right)\mathrm{R}_i^\top\mathrm{R}_j\right) \\
\boldsymbol{r}_{\Delta\boldsymbol{v}_{ij}} &\doteq \mathrm{R}_i^\top(\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}\Delta t_{ij}) - \left[\Delta\bar{\boldsymbol{v}}_{ij} + \tfrac{\partial\Delta\bar{\boldsymbol{v}}_{ij}}{\partial\boldsymbol{b}_i^g}\delta\boldsymbol{b}_i^g + \tfrac{\partial\Delta\bar{\boldsymbol{v}}_{ij}}{\partial\boldsymbol{b}_i^a}\delta\boldsymbol{b}_i^a\right] \\
\boldsymbol{r}_{\Delta\boldsymbol{p}_{ij}} &\doteq \mathrm{R}_i^\top(\boldsymbol{p}_j - \boldsymbol{p}_i - \boldsymbol{v}_i\Delta t_{ij} - \tfrac{1}{2}\boldsymbol{g}\Delta t_{ij}^2) - \left[\Delta\bar{\boldsymbol{p}}_{ij} + \tfrac{\partial\Delta\bar{\boldsymbol{p}}_{ij}}{\partial\boldsymbol{b}_i^g}\delta\boldsymbol{b}_i^g + \tfrac{\partial\Delta\bar{\boldsymbol{p}}_{ij}}{\partial\boldsymbol{b}_i^a}\delta\boldsymbol{b}_i^a\right] \\
\boldsymbol{r}_{\boldsymbol{b}_{ij}^g} &\doteq \boldsymbol{b}_j^g - \boldsymbol{b}_i^g \\
\boldsymbol{r}_{\boldsymbol{b}_{ij}^a} &\doteq \boldsymbol{b}_j^a - \boldsymbol{b}_i^a
\end{aligned} \tag{2.2}$$

其中 $\boldsymbol{g}$ 为全局坐标系下的重力加速度，$\delta\boldsymbol{b}_i^g$ 和 $\delta\boldsymbol{b}_i^a$ 分别为角速度及加速度 bias 的增量，$\Delta\bar{\boldsymbol{X}}_{ij}$ 为 IMU 预积分的结果：

$$\Delta\bar{\boldsymbol{X}}_{ij} \doteq \begin{bmatrix} \Delta\bar{\mathrm{R}}_{ij} & \Delta\bar{\boldsymbol{v}}_{ij} & \Delta\bar{\boldsymbol{p}}_{ij} \end{bmatrix} \tag{2.3}$$

log 和 exp 是李群 $\mathbb{SO}^3$ 和对应李代数 $\mathfrak{so}^3$ 之间的对数映射和指数映射。以上均采用了 IMU 预积分原文的标记方法，具体细节可参考 [11] 。

本框架还实现了常用的重投影误差目标函数，某时刻的全局坐标系下的三维点 $\boldsymbol{l}_k$ 在相机-IMU 状态 $\boldsymbol{X}_i$ 中的投影误差如下所示：

$$\boldsymbol{r}_{\pi_{ik}} \doteq \pi\left(\mathrm{K}\mathrm{R}_i^\top(\boldsymbol{l}_k - \boldsymbol{p}_i)\right) \tag{2.4}$$

其中 $\pi(\cdot)$ 是投影函数，K 是相机内参矩阵。

某时刻 VISLAM 中的集束调整问题的总能量函数可以记为：

$$\begin{aligned}
E \doteq \sum_{i=1,j=i+1}^{I-1} \Big(&\|\boldsymbol{r}_{\Delta\mathrm{R}_{ij}}\|_{\Sigma_R}^2 + \|\boldsymbol{r}_{\Delta\boldsymbol{v}_{ij}}\|_{\Sigma_v}^2 + \|\boldsymbol{r}_{\Delta\boldsymbol{p}_{ij}}\|_{\Sigma_p}^2 \\
&+ \|\boldsymbol{r}_{\boldsymbol{b}_{ij}^g}\|_{\Sigma_{bg}}^2 + \|\boldsymbol{r}_{\boldsymbol{b}_{ij}^a}\|_{\Sigma_{ba}}^2 \Big) + \sum_{i=1}^{I}\sum_{k=1}^{K_i}\|\boldsymbol{r}_{\pi_{ik}}\|_{\Sigma_\pi}^2
\end{aligned} \tag{2.5}$$

其中 I 为当前所有相机-IMU 状态的集合，K_i 为相机-IMU 状态 $\boldsymbol{X}_i$ 观测到的所有三维点的集合，Σ_R，Σ_v，Σ_p，Σ_{bg}，Σ_{ba}，Σ_π 为对应目标函数的协方差。

2.2 基于因子图的增量舒尔补算法

iSAM2 算法[1;3] 创新性地在 SLAM 问题中使用了贝叶斯推断树来编码集束调整过程中的信息矩阵分解过程，达到高效地更新平方根信息矩阵的目的，显著地提升了全局优化的性能。其算法不仅适用于 SLAM 中的集束调整问题，也适用于一般的非线性最小二乘问题。也正是为了保证通用性，iSAM2 难以在状态之间的关联特点高度统一的 SLAM 问题中发挥更高的性能。

例如在图 1.3中，正规方程的三维点部分（左上角部分）高度稀疏，呈对角块状，在求解时应该优先考虑这一部分的分解。iSAM2 算法需要依赖 COLAMD[33] 算法来被动地检测矩阵分解的顺序，一方面需要额外的计算时间，另一方面也不一定能得到比经验更好的结果。

舒尔补虽然可以大幅加速集束调整的求解，但是在标准的批量式集束调整算法中，每一轮迭代重新构建舒尔补方程的过程仍然需要耗费大量的时间。如前面提到的，SLAM 问题具有很好的局部性，每一轮求解一般都只有一小部分较新的变量有所更新，如果采用增量式的方法，每一轮迭代时只重新计算这些变量在舒尔补方程中对应的部分，则可以进一步减少计算量。

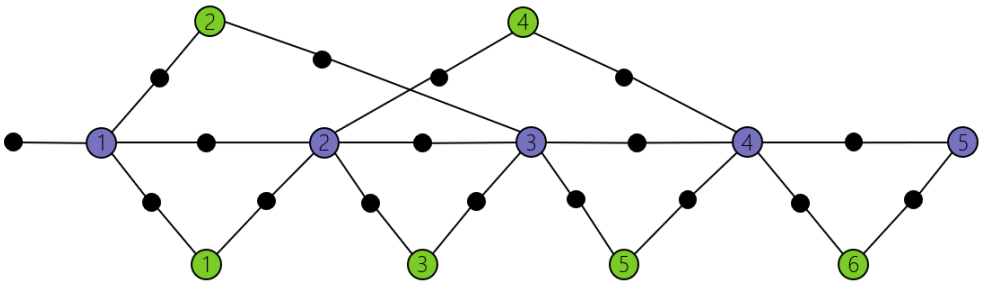

图 2.1 因子图

如图 2.1是一个常见集束调整问题的因子图示例，其对应的正规方程具有如图 2.2所示的稀疏结构。按照式(1.32)构建相机部分方程的舒尔补，依次迭代计算每一个三维点状态对应的舒尔补部分，并累加到图 2.3中的高亮部分所示的舒尔补中。图 2.4展示了计算舒尔补的第一次迭代。

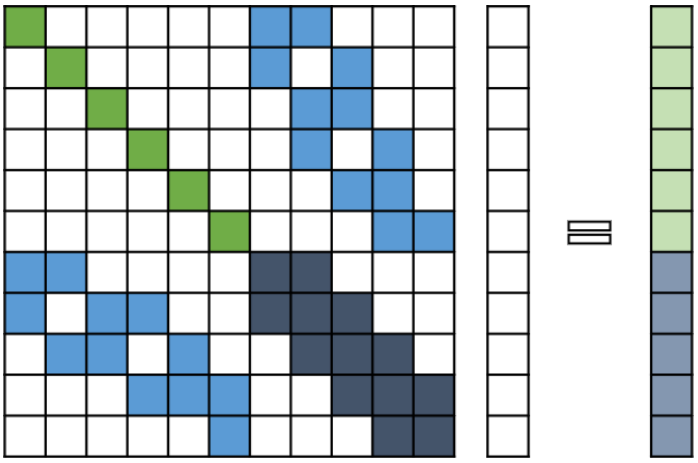

图 2.2 正规方程：左上角块对角矩阵部分对应三维点状态 1 至 6 的信息矩阵 P，右下角部分为相机状态 1 至 5 对应的信息矩阵 C，左下角和右上角为矩阵 W 和 $W^{\top}$；中间列向量从上至下依次对应三维点状态 1 至 6 和相机状态 1 至 5 的迭代步；右侧部分从上至下依次对应三维点状态 1 至 6 和相机状态 1 至 5。

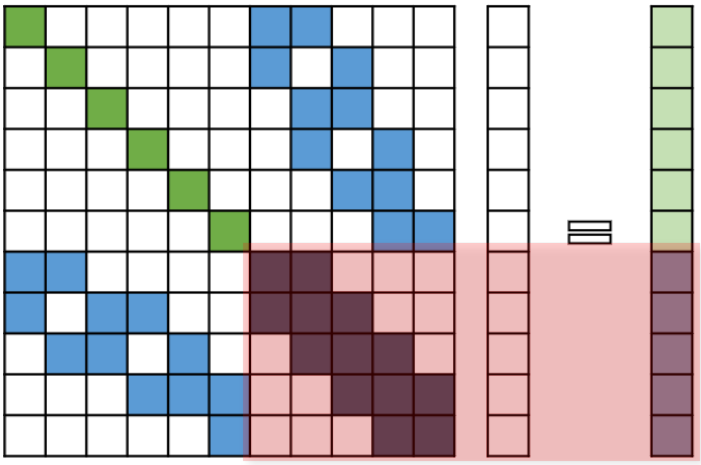

图 2.3 舒尔补：高亮部分为相机状态对应舒尔补方程，其余部分为原方程。

2.2.1 标记脏子图

增量更新舒尔补的方法关键在于尽可能地复用上一次迭代的结果，只针对性地计算舒尔补方程中变化较大的变量对应的部分。以因子图 2.1为例，假设只有相机状态 5 有较大的更新，则先将其标记为脏变量。在因子图中，只有与脏变量直接相连的因子需要标记为脏因子，如图 2.5所示。需要注意的是，尽管相机状态 5 和三维点状态 6 的值并未发生足够大的改变，但是由于与之相连的部分因子被标记为脏因子，因此也需要更新这些因子关于它们的雅各比矩阵。这些脏变量和脏因子共同组成了一个脏子图，对应地，图 2.6高亮标示了舒尔补方程中需要更新的脏块。算法 2详细说明了如何通过一个简单的宽度优先搜索

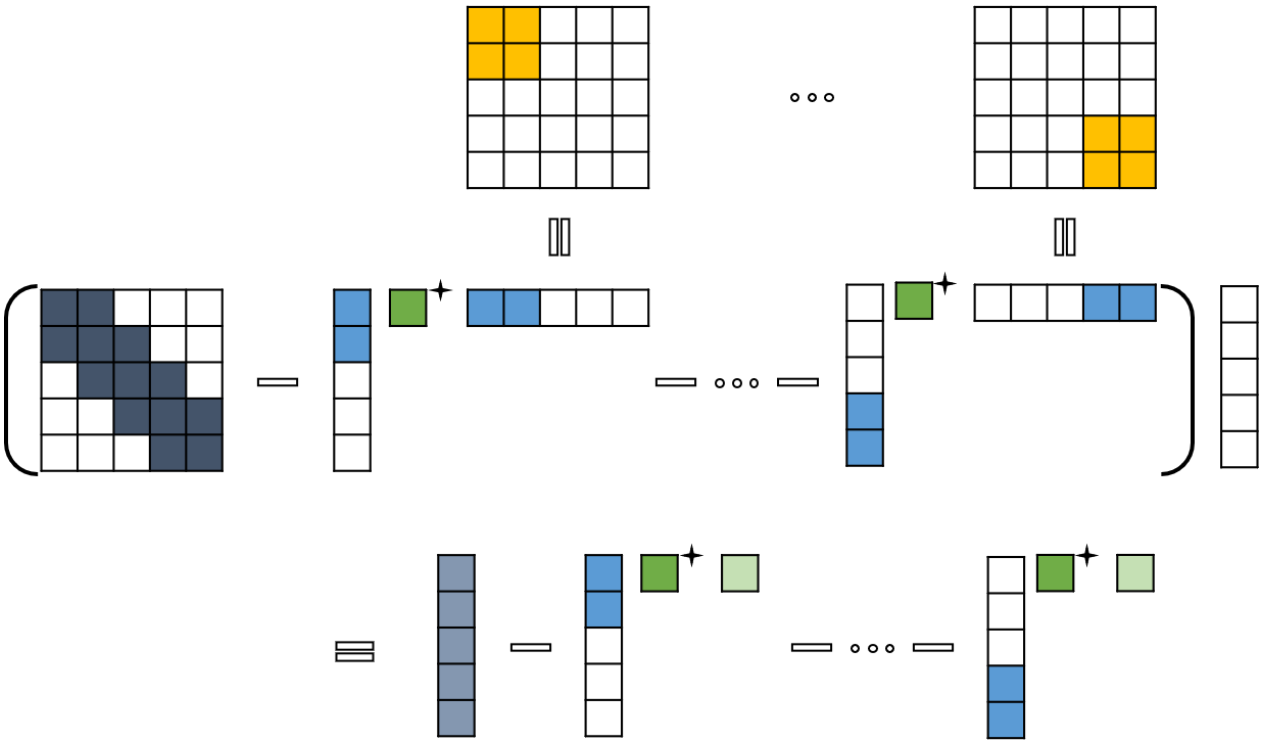

图 2.4 计算舒尔补的一次迭代：从左上角开始依次迭代计算每一个三维点对应的舒尔补部分，并累加到右下角相机部分中。

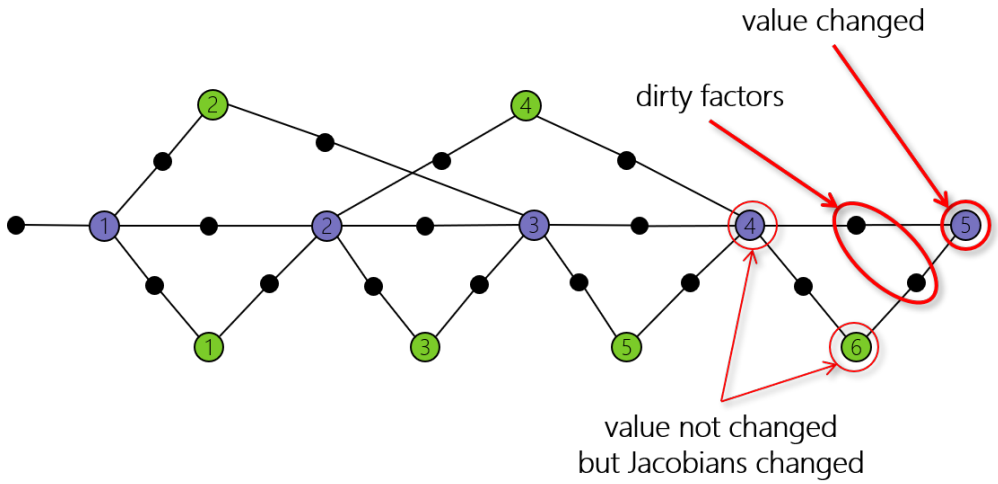

图 2.5 因子图待更新部分：相机状态 5 为脏变量；与其直接相连的因子为脏因子；相机状态 4 和三维点状态 6 为只有梯度发生了变化的脏变量。这些节点构成了一个脏子图。

标记出因子图中的脏子图。

算法 2 标记脏子图

输入: 因子图 $\check{\mathcal{G}} = \{\check{\mathcal{F}}, \check{\Theta}, \check{\mathcal{E}}\}$，迭代步长 $\boldsymbol{\delta}$

输出: 脏子图 $\mathcal{G} = \{\mathcal{F}, \Theta, \mathcal{E}\}$

```
F := {}, Θ := {}, E := {}
for all ‖δ_i‖_∞ ≥ ε do
    Θ := Θ ∪ {θ_i}                          {将变量 θ_i 标记为脏变量}
    for all {f_j ∈ F̌ | θ_i ∈ Θ_j} do        {Θ_j 为与 f_j 相连的变量集合}
        Θ := Θ ∪ Θ_j
        E := E ∪ {e_ij}                     {记录变量 θ_i 与因子 f_j 相连的边 e_ij}
        F := F ∪ {f_j}                      {将因子 f_j 标记为脏因子}
    end for
end for
```

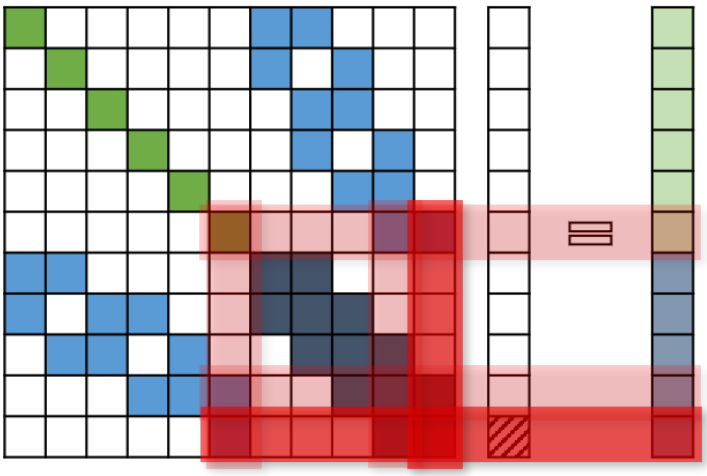

图 2.6　图 2.5对应的舒尔补方程中的脏块：相机状态 4、5 和三维点状态 6 构成的线性子系统。

2.2.2　增量更新舒尔补

舒尔补方程中被标记为脏块的矩阵构成了一个线性子系统，对应因子图中的脏子图。如图 2.7所展示的，在增量更新舒尔补的过程中，需要先从舒尔补方程中减去脏子图的贡献，随后重新线性化所有的脏因子，最后重新计算舒尔补。算法 3详细描述了这个过程。需要注意的是，在计算舒尔补的过程中会重复用到 $\mathrm{W}_i\mathrm{P}_{ii}^{-1}\mathrm{W}_i^{\top}$，$\mathrm{W}_i\mathrm{P}_{ii}^{-1}\boldsymbol{l}_i$，$\mathrm{J}_j^{\top}\mathrm{J}_j$，$\mathrm{J}_j^{\top}\boldsymbol{f}_j$，需要将这些中间项缓存下来，避免重复计算。

算法 3 增量更新舒尔补

输入: 脏子图 $\mathcal{G} = \{\mathcal{F}, \Theta, \mathcal{E}\}$，上一轮迭代的 $\boldsymbol{\delta}$，H，$\boldsymbol{\eta}$，S，$\boldsymbol{b}$

输出: 更新后的 H，$\boldsymbol{\eta}$，S，$\boldsymbol{b}$

for all 三维点状态 $\boldsymbol{\theta}_i \in \Theta$ **do** {减去三维点对舒尔补的贡献}

$$\mathrm{S} \coloneqq \mathrm{S} + \mathrm{W}_i {\mathrm{P}_{ii}}^{-1} \mathrm{W}_i^{\top}$$

$$\boldsymbol{b} \coloneqq \boldsymbol{b} + \mathrm{W}_i {\mathrm{P}_{ii}}^{-1} \boldsymbol{l}_i$$

end for

for all 脏因子 $\boldsymbol{f}_j \in \mathcal{F}$ **do** {更新脏因子对正规方程的贡献}

$$\mathrm{H} \coloneqq \mathrm{H} - \mathrm{J}_j^{\top} \mathrm{J}_j$$

$$\boldsymbol{\eta} \coloneqq \boldsymbol{\eta} + \mathrm{J}_j^{\top} \boldsymbol{f}_j$$

$$\mathrm{J}_j \coloneqq \frac{\partial \boldsymbol{f}_j(\Theta_j)}{\partial \Theta_j}$$

$$\boldsymbol{f}_j \coloneqq \boldsymbol{f}_j(\Theta_j)$$

$$\mathrm{H} \coloneqq \mathrm{H} + \mathrm{J}_j^{\top} \mathrm{J}_j$$

$$\boldsymbol{\eta} \coloneqq \boldsymbol{\eta} - \mathrm{J}_j^{\top} \boldsymbol{f}_j$$

end for

for all 三维点状态 $\boldsymbol{\theta}_i \in \Theta$ **do** {更新三维点对舒尔补的贡献}

$$\mathrm{S} \coloneqq \mathrm{S} - \mathrm{W}_i {\mathrm{P}_{ii}}^{-1} {\mathrm{W}_i}^{\top}$$

$$\boldsymbol{b} \coloneqq \boldsymbol{b} - \mathrm{W}_i {\mathrm{P}_{ii}}^{-1} \boldsymbol{l}_i$$

end for

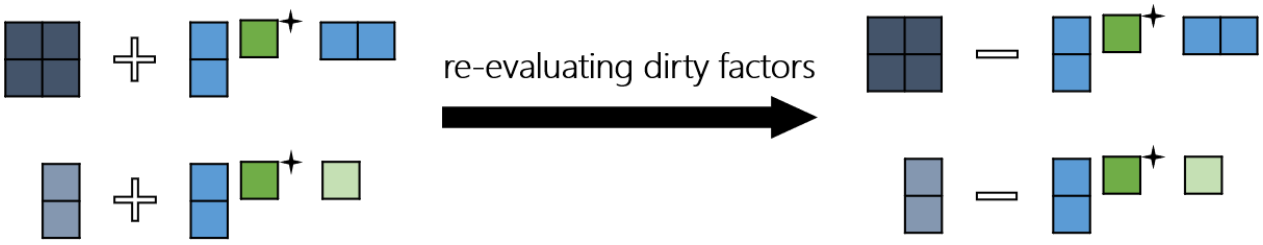

图 2.7 更新舒尔补

2.2.3 状态增广

除了状态变量发生较大变化的时候需要更新因子图和舒尔补方程，集束调整过程中，如果有新的状态或因子加入，也需要对因子图进行增广，并对和舒尔补方程进行相应更新。实际上，这个过程也可以认为是算法 3的一个特例。如图 2.8，新加入的状态通过新的因子与原因子图中相关联的部分状态共同构成了一个子图，将这个子图标记为脏子图，即可使用算法 3进行更新。算法 4给出了详细的过程，为了提高效率，因子图的增广和舒尔补方程的更新会被同时执行。

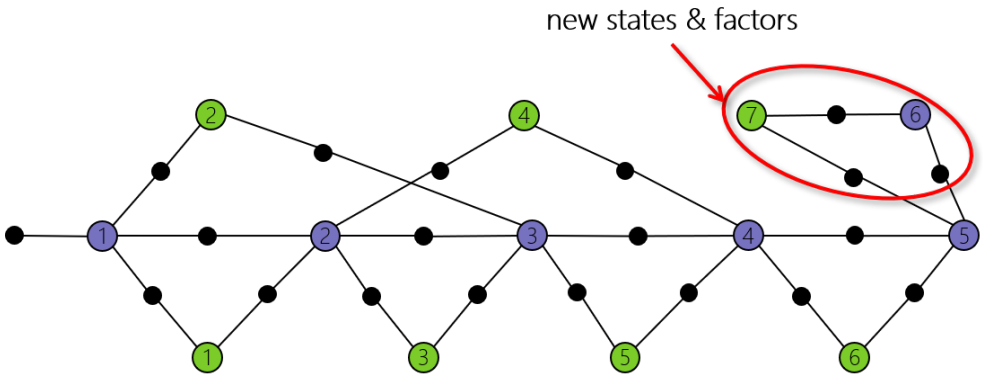

图 2.8 增广因子图

2.3 线性求解

通过算法 3更新舒尔补方程后，可以通过求解舒尔补方程快速得到相机状态的增量 $\boldsymbol{\delta}^c$，再进一步通过回代求解三维点状态的增量 $\boldsymbol{\delta}^l$。为了提升这部分求解数值稳定性，本文提出了使用混合式 LM-DL 方法（Hybrid Levenberg-Marquardt-Dog-Leg，简称 HLMDL）求解舒尔补方程。在回代求解部分，受 iSAM2[1] 算法的启发，本文提出了使用部分贝叶斯推断树（Partial Bayes Tree，简称 PBT）的方法编码三维点状态和相机状态之间的推断关系，以达到增量式回代的目的，进一步提高效率。

算法 4 状态增广

输入: 因子图 $\check{\mathcal{G}} = \{\check{\mathcal{F}}, \check{\Theta}, \check{\mathcal{E}}\}$，新加入的因子和状态 $\{\mathcal{F}_0, \Theta_0\}$，上一次优化的 H，$\boldsymbol{\eta}$，S，$\boldsymbol{b}$

输出: 更新后的 H，$\boldsymbol{\eta}$，S，$\boldsymbol{b}$

for all 三维点状态 $\boldsymbol{\theta}_i \in \Theta_0$ **do** {增广正规方程，如图 2.9}

增广矩阵：P，W，$\boldsymbol{l}$

end for

for all 相机状态 $\boldsymbol{\theta}_i \in \Theta_0$ **do**

增广矩阵：C，S，W，$\boldsymbol{c}$，$\boldsymbol{b}$

end for

$\Theta \coloneqq \{\}$ {更新正规方程和舒尔补，如图 2.10}

for all $\{\boldsymbol{f}_j \in \mathcal{F}_0 | \Theta_j \cap \check{\Theta} \neq \emptyset\}$ **do**

$\Theta \coloneqq \Theta \cup (\Theta_j \cap \check{\Theta})$

end for

for all 三维点状态 $\boldsymbol{\theta}_i \in \Theta$ **do** {减去三维点对舒尔补的贡献}

$$\mathrm{S} \coloneqq \mathrm{S} + \mathrm{W}_i \mathrm{P}_{ii}{}^{-1} \mathrm{W}_i^{\top}$$

$$\boldsymbol{b} \coloneqq \boldsymbol{b} + \mathrm{W}_i \mathrm{P}_{ii}{}^{-1} \boldsymbol{l}_i$$

end for

for all 脏因子 $\boldsymbol{f}_j \in \mathcal{F}_0$ **do** {更新新因子对正规方程的贡献}

$$\mathrm{J}_j \coloneqq \frac{\partial \boldsymbol{f}_j(\Theta_j)}{\partial \Theta_j}$$

$$\boldsymbol{f}_j \coloneqq \boldsymbol{f}_j(\Theta_j)$$

$$\mathrm{H} \coloneqq \mathrm{H} + \mathrm{J}_j^{\top} \mathrm{J}_j$$

$$\boldsymbol{\eta} \coloneqq \boldsymbol{\eta} - \mathrm{J}_j^{\top} \boldsymbol{f}_j$$

end for

for all 三维点状态 $\boldsymbol{\theta}_i \in \Theta$ **do** {更新三维点对舒尔补的贡献}

$$\mathrm{S} \coloneqq \mathrm{S} - \mathrm{W}_i \mathrm{P}_{ii}{}^{-1} \mathrm{W}_i{}^{\top}$$

$$\boldsymbol{b} \coloneqq \boldsymbol{b} - \mathrm{W}_i \mathrm{P}_{ii}{}^{-1} \boldsymbol{l}_i$$

end for

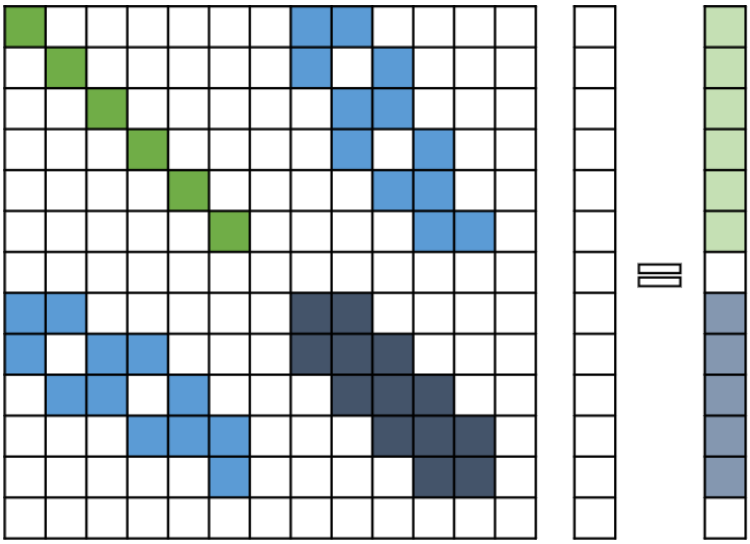

图 2.9 增广正规方程

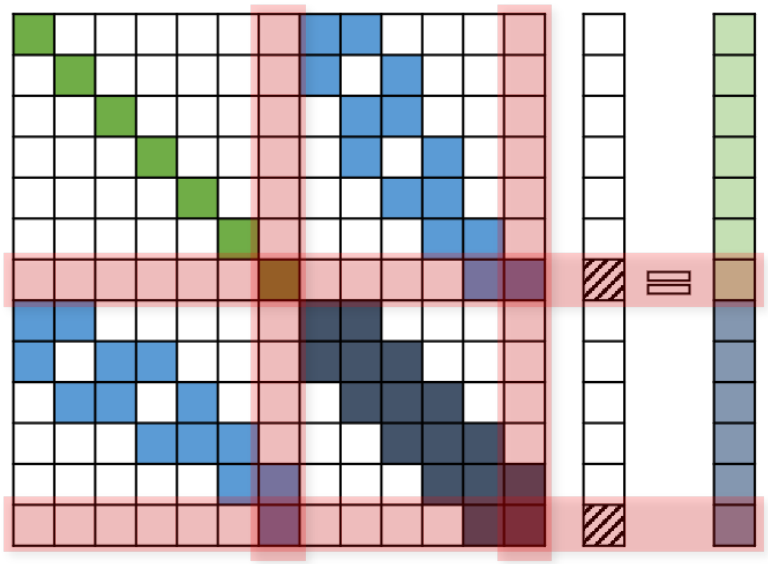

图 2.10 更新正规方程

2.3.1 HLMDL 方法求解舒尔补方程

LM 法的阻尼因子会影响正规矩阵的对角线元素，经过舒尔补操作之后，阻尼因子的影响被不可逆地累积到了舒尔补矩阵中，使得每次调整阻尼因子的时候必须要重新完整计算舒尔补。这会导致舒尔补无法进行增量更新，因此 ICE-BA[10] 和 SLAM++ 中均使用 DL 法进行非线性求解。然而如第 1 章中提到的，在实际求解线性问题时，往往会遇到矩阵秩亏或不正定的情况，而且这种情况在舒尔补矩阵中更为严重，这就造成了直接求解时数值不稳定的问题。为了避免这个问题，本文提出了改进，使用 HLMDL 的方法，在舒尔补矩

阵的对角线上加上阻尼因子：

$$\mathrm{S}' \doteq \mathrm{S} + \mu \mathbf{diag}(\mathrm{S}) \tag{2.6}$$

来改善舒尔补矩阵的秩亏现象。然后通过求解新的舒尔补方程得到相机状态增量 $\boldsymbol{\delta}^c$：

$$\boldsymbol{\delta}^c = \mathrm{S}' \setminus \boldsymbol{b} \tag{2.7}$$

最后从 S' 中减去阻尼因子的影响，恢复为原先的舒尔补矩阵：

$$\mathrm{S} = \mathrm{S}' - \mu \mathbf{diag}(\mathrm{S}) \tag{2.8}$$

由于阻尼因子直接施加在构建好的舒尔补矩阵上而非原始的正规矩阵 H，其影响是可逆的，经过简单的操作即可恢复，在下一轮迭代中仍可以使用增量式方法更新舒尔补方程。通过 HLMDL 方法求得的 $\boldsymbol{\delta}^c$ 是原始舒尔补方程的一个近似解，但是由于使用了迭代的方式求解，单次迭代的近似解对最终的结果影响很小。这里的阻尼因子的调整与 DL 的信赖信赖的调整是独立的，本文使用了 [2] 中的方法进行阻尼调节，逐步缩小阻尼因子，使得最后的解收敛于准确的解。

方程(2.7)的求解可以使用任何常用的矩阵分解法，如 QR 分解、Cholesky 分解等。也可以采用迭代式的线性求解方法，比如预处理共轭梯度法（preconditioned conjugate gradient，简称 PCG）。本文提出的框架提供了自定义线性求解方法的接口，用户可以自己实现符合实际数值稳定性需求和求解效率要求的线性求解方法，也可以使用内置的 Cholesky 分解求解器和 I-PCG 求解器。

由于 HLMDL 方法在舒尔补方程上施加了阻尼因子，本文没有使用增量式 Cholesky 分解[35] 来求解式(2.7)，而是提供了常规的 Cholesky 分解法和 ICE-BA 中提出的 I-PCG 方法，并配合 Jacobi 预处理方法[36] 来提升线性求解的效率。

2.3.2 使用 PBT 方法增量回代

ICE-BA 中使用了直接回代的方法求解三维点状态的增量，即求解完相机状态增量 $\boldsymbol{\delta}^c$ 后，进一步使用式(1.34)直接批量回代求解三维点状态的增量 $\boldsymbol{\delta}^l$。最后得到的总的状态增量 $\boldsymbol{\delta}$，通过阈值判断其每个分量 $\boldsymbol{\delta}_i$ 是否发生了足够大的更新并将发生较大更新的分量更新到状态中：

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{\delta}_i, \quad \forall \left\| \boldsymbol{\delta}_i \right\|_\infty \geq \varepsilon \tag{2.9}$$

这一步求解并不是增量进行的，除了计算速度受到一定的影响，还存在变量更新不一致的问题。一种典型的情况是：某个相机状态 i 的增量 $\boldsymbol{\delta}_i^c$ 较小，而与之相关的某个三维点

状态 j 的增量 $\boldsymbol{\delta}_j^l$ 发生了较大的更新，此时对应的因子 $f(\boldsymbol{x}_i^c, \boldsymbol{x}_j^l)$ 被标记为脏因子，需要重新线性化 $f(\boldsymbol{x}_i^c + \boldsymbol{\delta}_i^c, \boldsymbol{x}_j^l + \boldsymbol{\delta}_j^l)$，但由于 $\boldsymbol{\delta}_i^c$ 较小而被忽略，实际计算的是 $f(\boldsymbol{x}_i^c, \boldsymbol{x}_j^l + \boldsymbol{\delta}_j^l)$。这就造成变量更新的不一致，在实际运行过程中，会导致增量求解过程中能量收敛的变慢，影响求解效率。

针对这一情况，本文基于 iSAM2 算法的贝叶斯推断树方法提出了 PBT 方法，编码了三维点状态和相机状态之间的推断关系。iSAM2 算法认为通过高斯消元法进行 Cholesky 分解或 QR 分解过程对应了将因子图转化为贝叶斯推断树的过程，贝叶斯推断树编码了所有变量之间的推断关系。在求解完根节点对应的变量之后，需要从根节点开始递归地回代求解叶节点。如果某个节点的变量的更新较小，则停止回代，其所有后代节点均不再继续回代求解（即使其中仍有可能有潜在的更新较大的节点）。这一操作使得整个回代求解过程与变量推断关系严格一致。

构建舒尔补的过程实际也是使用高斯消元将一部分变量（三维点状态）进行消去的过程，同样这个过程也可以生成贝叶斯推断树。以因子图 2.1为例，其舒尔补过程可以生成如图 2.11所示的贝叶斯推断树。由于舒尔补只消去了三维点状态，而相机状态部分则保留原始状态，因此构建的贝叶斯推断树为部分贝叶斯推断树 $\mathcal{T}_p$ (Partial Bayes Tree，简称 PBT)，高度固定为 2，所有相机变量共同构成了根节点 $Rt(\mathcal{X})$，每一个叶节点 $L(l_i|\mathcal{S}_i)$ 代表一个三维点状态 l_i 及其前置条件状态集合 $\mathcal{S}_i$。

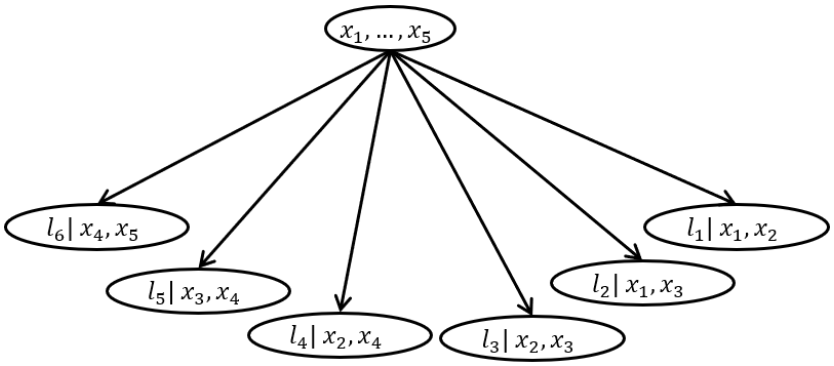

图 2.11 舒尔补构建的贝叶斯推断树

利用 PBT，可以方便地推断出任意一个三维点状态的前置条件变量。如果某个相机状态变量没有更新或更新较小，则以其为前置条件的三维点状态也不会更新（即使该三维点可能也发生了较大更新）。这就保证了三维点状态的更新和相机状态的更新的一致性。

算法 5介绍了 PBT 回代的基本步骤。通过 PBT 回代求解得到 $\boldsymbol{\delta}^l$ 后，进一步地，如果还需要进行下一轮迭代，则将综合更新量 $\boldsymbol{\delta}$ 传递给算法 2，计算新一轮的迭代。整个增量舒尔补和线性求解的过程均可以基于块状稀疏矩阵实现。相比与稠密矩阵或一般的稀疏矩

算法 5 PBT 回代算法

输入: PBT 树 $\mathcal{T}_p$，相机状态增量 $\boldsymbol{\delta}^c$

输出: 三维点状态增量 $\boldsymbol{\delta}^l$

$\boldsymbol{\delta}^l \coloneqq \mathbf{0}$

for all 叶子节点 $L(l_i|\mathcal{S}_i) \in \mathcal{T}_p$ **do**

记 $\mathcal{S}_i$ 对应的相机状态增量为 $\boldsymbol{\delta}^c_{S_i}$

if $\{\left\|\boldsymbol{\delta}^c_{S_i}\right\|_\infty < \varepsilon\}$ **then**

$\boldsymbol{\delta}^l_i \coloneqq \mathbf{0}$

else

$\boldsymbol{\delta}^l_i \coloneqq \boldsymbol{l}_i$

for all $\{\boldsymbol{\delta}^c_j \in \mathcal{S}_i\}$ **do**

$\boldsymbol{\delta}^l_i \coloneqq \boldsymbol{\delta}^l_i - \mathrm{W}^\top_{ji}\boldsymbol{\delta}^c_j$

end for

$\boldsymbol{\delta}^l_i \coloneqq \mathrm{P}^{-1}_{ii}\boldsymbol{\delta}^l_i$

end if

end for

阵算法，块状稀疏矩阵的计算过程对 cache 更友好，也更适合使用 CPU 浮点指令集加速。得益于块状稀疏矩阵的高效，整个增量式集束调整的效率相比批量式集束调整可以有非常大的提升。

2.4 增量式集束调整方法对比

一般来说，使用一个一般的求解器求解集束调整问题的流程可以用图 2.12来表示：1）用户通过调用相应的接口构建集束调整问题（因子图）；2）首先求解器对问题进行线性化或同时计算鲁棒函数，随后不同求解器根据不同的策略先对变量进行分组或排序，或进行预消元（计算舒尔补）；3）然后求解器依次进行线性求解和线性化点更新，并判断是否继续迭代，循环这一过程直到收敛。更进一步地，当问题（因子图）中增加了新的目标函数或变量时，对相关的数据结构进行增广。

本文实现的集束调整框架同样符合上述流程，区别在于该框架几乎任意一个部分均可自定义实现，也可以采用内置的算法，具有很好的通用性和扩展性。表 2.1展示了本文提出的增量式集束调整框架和其他增量式优化算法的对比，并用粗体表示非增量求解的部分。

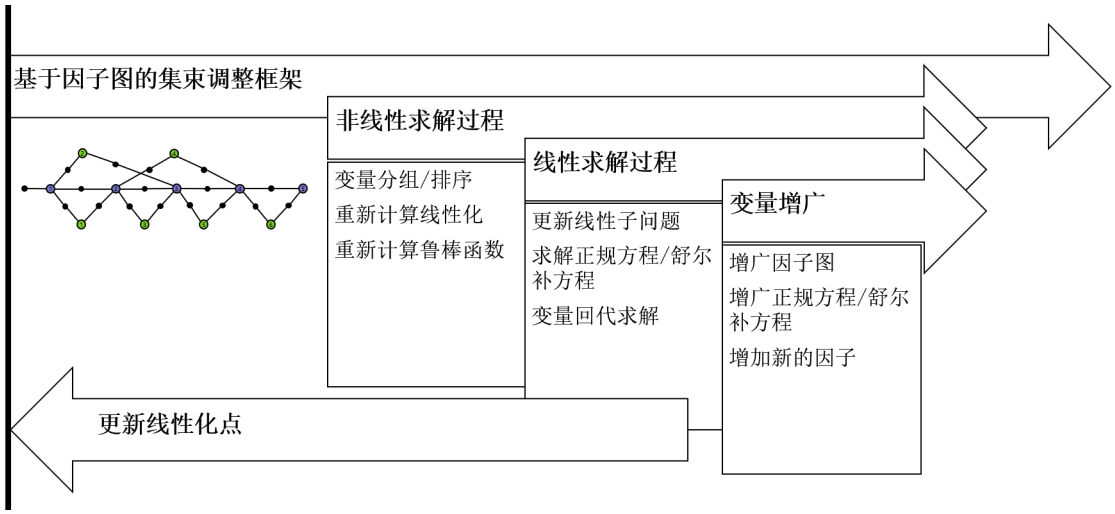

图 2.12 通用集束调整框架求解流程

需要注意的是，只有所有模块均符合增量操作的求解器才是完全增量的算法。而 I-PCG 仅仅是提高了收敛效率而没有降低时间复杂度，因此和 Cholesky 分解法一样，不属于增量操作。

表 2.1 增量式集束调整求解器对比：加粗显示了非增量操作的部分

求解器	本文提出的框架	ICE-BA	SLAM++	iSAM2
非线性部分	a）三维点和相机状态分别成组 b）根据构建舒尔补构建 PBT	a）三维点和相机状态分别成组	a）三维点和相机状态分别成组 b）相机部分使用 COLAMD 排序	a）使用 CCOLAMD 对变量进行排序 b）通过消元构建贝叶斯推断树
线性化	a）部分线性化 b）根据脏子图增量更新舒尔补	a）部分线性化 b）根据脏变量更新舒尔补	a）部分线性化 b）根据脏变量更新舒尔补	a）根据贝叶斯推断树进行即时线性化 b）更新平方根信息矩阵
线性求解	a）**Cholesy 分解或增量 I-PCG 求解舒尔补方程** b）根据 PBT 增量回代求解三维点	a）**I-PCG 求解舒尔补方程** b）**完整回代求解三维点**	a）**Cholesky 分解求解舒尔补方程** b）**完整回代求解三维点**	a）求解贝叶斯推断树根节点变量 b）根据贝叶斯推断树增量回代求解剩余变量
变量更新	标记脏变量	标记脏变量	标记脏变量	标记脏变量

2.5 本章小结

本章详细介绍了基于增量式舒尔补的集束调整框架，包括框架的设计和内置提供的目标函数、基于因子图的增量式舒尔补构建方法、HLMDL 算法和 PBT 回代算法，并对比了本文提出的增量式框架与其他增量式算法之间的区别。一组增量式集束调整优化的流程总结如下：

1. 根据因子图，采用传统方法构建舒尔补并使用 Cholesky 分解或 PCG 方法求解首轮

迭代，并构建 PBT；

2. 使用算法 2标记因子图和舒尔补方程需要更新的部分；

3. 使用算法 3更新舒尔补；

4. 使用 Cholesky 分解或 I-PCG 方法求解舒尔补方程；

5. 基于 PBT 推断进行回代求解三维点状态；

6. 重复步骤 2至步骤 5直到收敛；

7. 当新增了因子或状态时，使用算法 4进行增广，并重新从步骤 2开始新一组优化。

第 3 章 实验和结论

本章将通过模拟数据和真实数据，测试本文提出的增量式集束调整算法和传统批量式集束调整算法（基于 Ceres-Solver）的速度和精度。Ceres-Solver 是 SLAM 系统中常用的非线性最小二乘求解器，提供了稠密版本和稀疏版本的多种批量式数值优化算法实现（包括 LM 方法、DL 法等），且具有较好的数值稳定性和效率。

3.1 测试设置

测试数据包含本文自行模拟生成的相机-IMU-三维点的数据集 Cubic-50 和包含相机-三维点的 VSLAM 真实数据集 Cathedral[37] 和 Venice[8]。其中 Cubic-50 数据集为模拟生成的带高斯噪声的 VISLAM 数据集，包含 IMU 预积分的结果和协方差、相机状态的初值、视觉观测。Cathedral 和 Venice 为真实数据集，包含相机状态的初值、三维点状态的初值和视觉观测。这样的设置省去了 SLAM 前端视觉匹配、IMU 积分的过程，保证所有算法求解的问题相同。几组数据具体包含的内容如表 3.1所示，由于 Ceres-Solver 的 DL 法策略求解完整的 Cathedral 数据集和 Venice 数据集，顾只取 Cathedral 数据集的前 45 个相机状态，和 Venice 数据集的前 50 个相机状态，后文将用 Cathedral-46 和 Venice-51 代表这两个数据集。

表 3.1 测试数据集明细

数据集	相机数量	三维点数量	IMU 目标函数数量	重投影误差目标函数数量
Cubic-50	50	163	49	789
Venice-51	51	35110	0	117988
Cathedral-46	46	33717	0	208079

所有测试均基于最新的 Arch Linux 操作系统，搭载主频为 3.6GHz 的 Intel Core i7-7700 处理器，内存为 8GB。运行测试期间关闭其他所有应用程序，每一组测试的结果均取 10 次运行后的平均值。

3.2 结果对比

为了保证公平，所有求解器尽可能地设置成相同的算法，并将算法运行时间 T 定义为从问题构建开始到完成求解的总时间。所有优化器均使用稀疏求解策略，本文的优化策略

非线性部分为 HLMDL 法，线性部分为增量舒尔补-Cholesky 分解和增量舒尔补 I-PCG 迭代法。Ceres-Solver 的非线性优化策略则选取 LM 法和 DL 法，线性部分设置为舒尔补-QR 分解法。将所有优化算法的迭代次数设置为相同，通过对比运行时间 T 和迭代次数 t 来比较不同的求解策略的效率。通过对比求解结束时的总能量，即重投影误差目标函数的值 E_{proj} 和 IMU 预积分目标函数的值 E_{IMU} 来对比精度。

表 3.2展示了时间测试对比结果，包含运行时间、总的迭代次数以及每条实例耗时相对于测试基准耗时的百分比，并将第一行设置为其他测试实例的对比基准。例如第一行代表的是 Ceres-Solver 的 DL 法运行结果，使用的非线性策略为 DL 法，线性策略为舒尔补和 QR 分解，在 Cubic-50 数据集上求解共进行 5 次迭代，平均耗时 69 毫秒，相对于基准耗时的百分比为 100%。

表 3.2 测试结果：运行时间和迭代次数

优化策略耗时（T/t）	Cubic-50	Venice-51	Cathedral-46
Ceres-LM：Schur-QR	48ms/4	23018ms/230	42706ms/204
耗时相对于基准的百分比	100.0%	100.0%	100.0%
Ceres-DL：Schur-QR	44ms/4	失败	失败
耗时相对于基准的百分比	91.7%	N/A	N/A
HLMDL：增量 Schur-Cholesky	17ms/4	4545ms/230	7924ms/204
耗时相对于基准的百分比	35.4%	19.7%	18.6%
HLMDL：增量 Schur-I-PCG	18ms/4	失败	6710ms/205
耗时相对于基准的百分比	37.5%	N/A	15.7%

表 3.3展示了精度测试对比结果，每个数据集的初始总能量和求解结束收敛时的总能量。

表 3.3 测试结果：收敛时的误差

优化策略及收敛时总能量 E	Cubic-50	Venice-51	Cathedral-46
总能量初始值 E_0	2.514×10^3	9.171×10^0	3.600×10^0
Ceres-LM：Schur-QR	4.956×10^2	1.599×10^{-1}	7.135×10^{-1}
Ceres-DL：Schur-QR	5.153×10^2	N/A	N/A
HLMDL：增量 Schur-Cholesky	4.982×10^2	2.045×10^{-1}	6.111×10^{-1}
HLMDL：增量 Schur-I-PCG	5.004×10^2	N/A	2.064×10^0

可以看到，基于本文提出的增量式集束调整框架无论是在效率还是在精度上都达到了可用的水准。不同的数据集测试结果表示，在精度接近的情况下（收敛时总能量接近），本文提出的 HLMDL-增量舒尔补算法相对于 Ceres-Solver 的测试基准，求解时间减少了 84.3% 至 62.5% 不等，显著提升了效率。

3.3 本章小结

本章通过在模拟 VISLAM 数据集 Cubic-50 和开源 VSLAM 数据集上测试对比了本文提出的增量式舒尔补集束调整优化算法和流行的开源 Ceres-Solver 的求解效率和精度。得益于高效的增量式舒尔补算法和基于贝叶斯树的 PBT 回代算法，以及基于块状稀疏矩阵的矩阵计算和存储方式，本文提出的框架达到了较高的求解效率。可以看到在各个数据集上，相比开源的 Ceres-Solver，本文提出的基于增量舒尔补的集束调整方法均在保证一定精度的前提下将求解时间减少了 60% 以上，显著提升了效率。

值得注意的是，本文提出的集束调整框架仍然具有较高的通用性和可扩展性：除了集束调整问题，也可以求解一般的非线性最小二乘问题；除了提供的 Cholesky 分解法和 I-PCG 方法，用户也可以自定义新的线性求解方法，以供不同场景使用。

第 4 章 总结与展望

4.1 总结

在 SLAM 算法中，状态估计算法的速度和精度一定程度上决定了 SLAM 应用的可用性。传统的批量式集束调整求解器为了保证通用性，牺牲了一定的效率，而现有的基于增量式集束调整的求解器虽然效率很高，但由于求解方法固定，通常只适用于特定类型的问题，且由于耦合程度过高，难以在其他 SLAM 系统中使用。本文基于以上分析，提出了面向 VISLAM 的基于增量式集束调整的通用求解框架，并针对现有的基于增量式舒尔补的集束调整算法中的问题，提出了因子图编码的增量式舒尔补、HLMDL、PBT 回代等算法，在保证一定精度和变量一致性的前提下提升了求解效率。

另一方面，本文提出的框架在非线性优化部分，提供了 VISLAM 中常用的基于 IMU 预积分的目标函数接口和重投影误差目标函数，同时保留了可扩展性，用户可以直接使用内置的目标函数，也可以自定义目标函数；在线性求解部分，基于块状稀疏矩阵，提供了 Cholesky 分解法、I-PCG 等常用的线性求解接口，同时用户也可以定制专用的线性求解器。最后，在模拟数据和 SLAM++ 提供的集束调整数据集上进行了测试，通过实验验证了求解器的效率和精度。

4.2 展望

增量式集束调整利用了 SLAM 问题的稀疏性和局部性，极大地提升了效率。但是仍有一些问题可以改进：

- 在发生回路闭合的时候，通常大量的状态变量都需要进行更新，增量式集束调整会退化到普通的批量式集束调整方法，效率严重下降；
- 状态边缘化通常会影响问题的稀疏性，频繁的边缘化操作则会导致系统变得十分稠密，不利于基于块状稀疏矩阵的求解算法。可以考虑一些稀疏化方法，使用一些近似的较为稀疏的先验约束来代替原本由边缘化得到的较为稠密的先验约束，以加快计算。

- 本文实现的增量式集束调整框架仍然采用了非增量的 Cholesky 分解或 I-PCG 迭代来求解舒尔补方程。如何在保证数值稳定性的情况下进一步利用增量特点，加速线性舒尔补方程的求解（如使用增量式 Cholesky 分解），是本文的下一步工作。
- 集束调整算法的线性化部分、舒尔补部分仍有可能采用更高效的并行算法提升效率。

如何在集束调整求解器中解决以上问题，或是在遇到以上问题的时候保持计算效率和精度，仍是一个很大的挑战。

参考文献

[1] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree[J]. The International Journal of Robotics Research, 2012, 31(2):216–235.

[2] O Tingleff, K Madsen, HB Nielsen. Methods for non-linear least squares problems[J]. Lecture Note in Computer Science 02611 Optimization and Data Fitting, 2004.

[3] Michael Kaess, Ananth Ranganathan, Frank Dellaert. isam: Incremental smoothing and mapping[J]. IEEE Transactions on Robotics, 2008, 24(6):1365–1378.

[4] Richard Hartley, Andrew Zisserman. Multiple view geometry in computer vision[M]. Cambridge university press, 2003.

[5] Yi Ma, Stefano Soatto, Jana Kosecka, S Shankar Sastry. An invitation to 3-d vision: from images to geometric models[M], volume 26. Springer Science & Business Media, 2012.

[6] Ji Zhang, Sanjiv Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast[C]. In Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015, 2174–2181.

[7] Sameer Agarwal, Keir Mierle, Others. Ceres solver[EB/OL]. `http://ceres-solver.org`, 2012.

[8] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, Wolfram Burgard. g 2 o: A general framework for graph optimization[C]. In Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011, 3607–3613.

[9] Viorela Ila, Lukas Polok, Marek Solony, Klemen Istenic. Fast incremental bundle adjustment with covariance recovery[C]. In International Conference on 3D Vision. 2017, 4321–4330.

[10] Haomin Liu, Mingyu Chen, Guofeng Zhang, Hujun Bao, Yingze Bao. Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam[C]. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, 1974–1982.

[11] Christian Forster, Luca Carlone, Frank Dellaert, Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry[J]. IEEE Transactions on Robotics, 2017, 33(1):1–21.

[12] Bill Triggs, Philip F McLauchlan, Richard I Hartley, Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis[C]. In International workshop on vision algorithms. Springer, 1999, 298–372.

[13] Feng Lu, Evangelos Milios. Globally consistent range scan alignment for environment mapping[J]. Autonomous robots, 1997, 4(4):333–349.

[14] Feng Lu, Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans[J]. Journal of Intelligent and Robotic systems, 1997, 18(3):249–275.

[15] Sebastian Thrun, Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures[J]. The International Journal of Robotics Research, 2006, 25(5-6):403–429.

[16] Jorge Nocedal, Stephen J. Wright. Numerical Optimization[M]. New York, NY, USA: Springer, 2006, 2nd edition.

[17] Timothy D Barfoot. State Estimation for Robotics[M]. Cambridge University Press, 2017.

[18] Andrew J Davison, Ian D Reid, Nicholas D Molton, Olivier Stasse. Monoslam: Real-time single camera slam[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2007, (6):1052–1067.

[19] Anastasios I Mourikis, Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation[C]. In Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007, 3565–3572.

[20] Mingyang Li, Anastasios I Mourikis. Improving the accuracy of ekf-based visual-inertial odometry[C]. In Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012, 828–835.

[21] Guoquan P Huang, Anastasios I Mourikis, Stergios I Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based slam[C]. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on. IEEE, 2008, 473–479.

[22] Guoquan P Huang, Anastasios I Mourikis, Stergios I Roumeliotis. An observability-constrained sliding window filter for slam[C]. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011, 65–72.

[23] Guoquan P Huang, Anastasios I Mourikis, Stergios I Roumeliotis. A quadratic-complexity observability-constrained unscented kalman filter for slam[J]. IEEE Transactions on Robotics, 2013, 29(5):1226–1243.

[24] Axel Barrau, Silvere Bonnabel. An ekf-slam algorithm with consistency properties[J]. arXiv preprint arXiv:1510.06263, 2015.

[25] Georg Klein, David Murray. Parallel tracking and mapping for small ar workspaces[C]. In Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on. IEEE, 2007, 225–234.

[26] Raul Mur-Artal, Jose Maria Martinez Montiel, Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system[J]. IEEE Transactions on Robotics, 2015, 31(5):1147–1163.

[27] Raúl Mur-Artal, Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular,

stereo and RGB-D cameras[J]. IEEE Transactions on Robotics, 2017, 33(5):1255–1262.

[28] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization[J]. The International Journal of Robotics Research, 2015, 34(3):314–334.

[29] Tong Qin, Peiliang Li, Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator[J]. IEEE Transactions on Robotics, 2018, 34(4):1004–1020.

[30] Peiliang Li, Tong Qin, Botao Hu, Fengyuan Zhu, Shaojie Shen. Monocular visual-inertial state estimation for mobile augmented reality[C]. In 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). IEEE, 2017, 11–21.

[31] Jakob Engel, Vladlen Koltun, Daniel Cremers. Direct sparse odometry[J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 40(3):611–625.

[32] Hailin Jin, Paolo Favaro, Stefano Soatto. A semi-direct approach to structure from motion[J]. The Visual Computer, 2003, 19(6):377–394.

[33] Timothy A Davis, John R Gilbert, Stefan I Larimore, Esmond G Ng. Algorithm 836: Colamd, a column approximate minimum degree ordering algorithm[J]. ACM Transactions on Mathematical Software (TOMS), 2004, 30(3):377–380.

[34] Yanqing Chen, Timothy A Davis, William W Hager, Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate[J]. ACM Transactions on Mathematical Software (TOMS), 2008, 35(3):22.

[35] Lukas Polok, Viorela Ila, Marek Solony, Pavel Smrz, Pavel Zemcik. Incremental block cholesky factorization for nonlinear least squares in robotics.[C]. In Robotics: Science and Systems. 2013.

[36] Yekeun Jeong, David Nister, Drew Steedly, Richard Szeliski, In-So Kweon. Pushing the envelope of modern methods for bundle adjustment[J]. IEEE transactions on pattern analysis and machine intelligence, 2012, 34(8):1605–1617.

[37] Hansung Kim, Adrian Hilton. Influence of colour and feature geometry on multi-modal 3d point clouds data registration[C]. In 3D Vision (3DV), 2014 2nd International Conference on. IEEE, 2014, volume 1, 202–209.