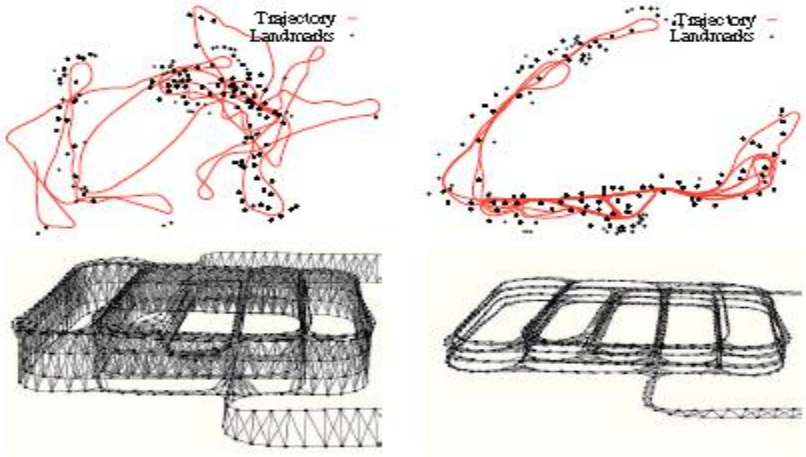
G2O：一种图优化的通用框架

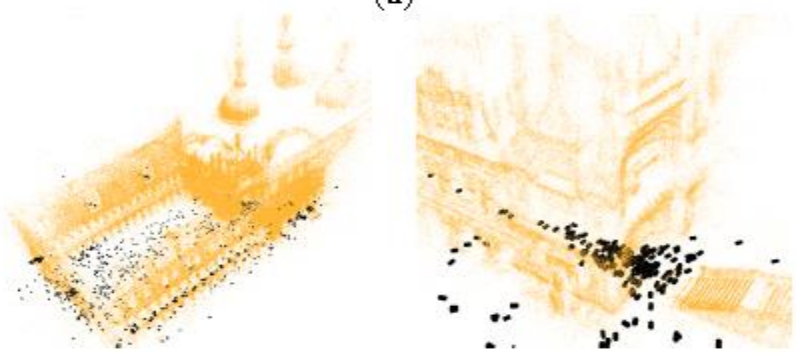
摘要：许多机器学和计算机视觉的常见问题都包括各种类型的即时定位和地图构建（SLam）和捆集调整（BA）。这些可以表达成误差函数方差最小化的优化问题也可以用图来表示。本文描述了一种解决这种问题通用的结构框架，即 G2O。G2O 是一个用于优化基于图形的非线性误差函数的开源的 C++框架。我们的系统已经设计成可以通过简单的扩展就能应用到大范围的问题上，并且，新的问题可以通过经典的几行代码进行详细描述。目前在一些不同的 SLAM 和 BA 问题上实现并提供了解决方案。我们在大范围的实际场景和模拟数据中进行了测试，结果表明 G2O 提供了一个可以和用来解决具体问题的先进的算法相抗衡的方法。

1 介绍

在机器学和计算机视觉中的大多数问题都包含了使得非线性误差函数最小化，这个问题可以用图形来表示。典型的例子就是同步定位和地图构建（SLAM），或者是捆集调整（BA）问题。这些问题的最终目标就是找到合适的参数配置或者状态变量使得他们能够最大程度的匹配一系列的由高斯噪声所影响的量测。例如，在基于图形的 SLAM 问题中，状态变量可以是机器人在环境中的位置也可以是机器人通过传感器获取的地图中的局部地标。因此，测量只取决于局部相关的两个状态变量，例如：在两个连续的位置之间的一次距离的量测只取决于连接的位置。同样，在 BA 或者基于地标的 SLAM 问题中。3D 点或者地标的测量只取决于真实环境中观测到的位置和传感器获得的位置。



(a)



(b)

所有这些问题都可以用图形来表示。图中的每一个节点表示一个需要优化的状态变量，每一条在两个变量之间的边代表了它所连接的两个节点间成对的关系（约束关系）。理论上，已经提出了很多关于这方面的理论方法。对于大多数的应用，使用标准的方法例如高斯牛顿法、Levenberg-Marquardt（LM）方法、高斯-塞德尔迭代法或者各式各样的梯度下降法等可以简单的得到可以接受的效果。但是，为了得到更好的效果，还需要本质上的努力和专业领域的知识。

本文中，我们采用图形的方式来描述一种解决非线性最小方差问题的通用框架。我们把这种框架称为 G20（通用图优化）。图 1 概要的展示了一些可以用 G2O 来解决后端优化的问题。该系统完成了和最先进的算法的对比，对比结果表明该算法能够用于通用的非线性量测。我们通过以下的方法得到了满意的效果：
利用图的稀疏连通性
利用图的特殊结构 ，这种特殊结构经常出现在之前提到的问题中
利用先进的方法解决稀疏线性系统
使用现代的处理器例如 SIMD 来指引和优化特征点在缓存区的存储

G2O 除了具有高效性外，它还具有通用性和可扩展性。一个 2D 的 SLAM 算法可以通过不到 30 行的 C++代码得以实现。使用者只要表达出算法中的具体误差函数和参数即可。

本文中，我们将 G20 算法应用到不同类型的最小二乘法优化问题中，并且与不同特殊问题算法进行比较。我们的评估是通过大量的真实场景和模拟数据得到的，在所有的实验中，G20 算法与最先进的算法得到的结果不相上下，有些结果还优于最先进的算法的结果。

本文剩下的部分主要按照以下内容展开。首先，主要讨论与 SLAM 和 BA 问题相关的工作。其次，在第三部分，我们分了图形可嵌入式的优化问题，这些问题是通过我们的系统处理并且是在高斯牛顿或者 LM 下的非线最小二乘法讨论得。在第四部分，我们分析了实验实现的特征。最后，在第五部分，我们展示了广泛的 G2O 的实验评估，并且把 G2O 的实验结果和一些先进的、具体问题具体解答的方法做了对比。

2 相关的工作

之前，图优化问题已经在机器学和计算机视觉方面得到了集中的研究。其中，一个由 Lu 和 Milion 提出的有创造性的工作是，两个扫描之间的相关运动是通过扫描匹配测量的，产生的图是通过线性迭代来进行优化的。但是在当时，人们认为对于实时测量来说图的优化是很耗时的。但是，随着直接线性动力学的发展，基于图形的 SLAM 优化又重新得到大家的重视，使用图优化理论优化 SLAM 问题又再次被提出。例如，Howard 等人，应用稀疏化来构建一个地图。Duckett 等人，提出使用高斯赛德尔迭代的方法来减少约束网中的误差。Frese 等人，介绍了一种多层的迭代（MLR），一种变形的高斯赛德尔迭代方法应用在解决问题的不同层面上。最近，Olson 等人，建议采用一种降低梯度的方式来优化图的位姿。之后，Grisetti 等人，把这种方法进行了扩展，通过应用基于树形的参数化来加快收敛速度。这两种方法对于原始的猜想是稳定的并且容易实现的。然而，他们假设协方差分布是粗糙的球形， 因此在优化一些没有空间或者有大量的不同的数值特征的图中的位姿的时候会比较困难。

图优化可以看成是一个非线性的最小二乘法问题，通常是通过在当前的状态下构建一个线性的系统来解决这个典型的问题。一个比较可行的用于解决这样的线性系统的方法是预处理共轭梯度法

（PCG），Konolige 、Montemerlo 和 Thrun 使用 PCG 作为一种高效的用于解决有较大稀疏的约束系统。由于它针对特殊的问题会有较高的效率，G2O 包含了应用在一个雅可比块的预先调整器中实现稀疏 PCG 的解决方法。

近期，Dellaert 和他的同事提出了一种叫做$\sqrt{\mathbf{SAM}}$ 的系统，他们通过使用稀疏矩阵求解器来执行这个系统。Kaess 等人，在 Dellaert 的基础上进行了转化，形成了一种叫做 iSAM 系统，这个系统可以更新与非线性最小二乘问题相关的线性矩阵。Konolige 等人，通过利用典型的线性系统的稀疏结构来展示如何高效的构造线性矩阵。然而，后一种方法受到 2D 位姿图的约束。在 G2O 中我们会用到和这些系统相似的方法。我们的系统可以应用到 SLAM 和 BA 优化问题以及他们的所有转换当中，例如：有地标的 2DSLAM、单目相机的 BA 问题、立体视觉的 BA 问题等。然而，相比之前提到的系统，G2O 处理方法在我们用过的所有用于评估的数据中是有本质的优势的。

在计算机视觉中，稀疏捆集调整是一个非线性最小二乘方法，这种方法利用了雅可比矩阵在点和相机位置之间的稀疏性。近来，有一些改进的类似的稀疏线性求解器的概念和针对大系统（约有 100M 稀疏矩阵元素的）的加快计算的舒尔减少的概念。（详细见第三部分 D）。有些基于非线性共轭梯度的新系统，这种系统不构造明确的线性系统，这使得收敛也会变得更慢，但是可以用于更大数据量的系统（约 1000M 矩阵元素）。本文，我们对比 G2O 和 SSBA 系统，SSBA 系统是目前刊登的最好的处理系统数据的方法。

3 使用最小二乘法的非线性图优化

许多机器学和计算机视觉的问题都可通过找到如下的最小函数的形式来解决：

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j\rangle \in C} \underbrace{\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^\top \mathbf{\Omega}_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})}_{\mathbf{F}_{ij}} \quad (1)$$

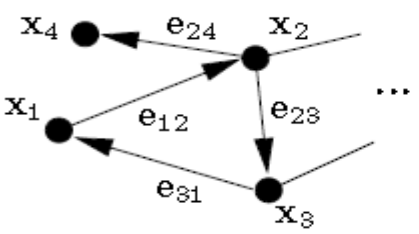
$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \mathbf{F}(\mathbf{x}). \quad (2)$$

此处的$\mathbf{x} = (\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top)^\top$ 是参数向量，其中每个 x_i 都代表一类的参数块，Z_{ij} 和$\mathbf{\Omega}_{ij}$分别代表了 X_j 和 X_i 之间约束关系的期望和信息矩阵，e (X_i, X_j, Z_{ij})是误差函数向量，这个函数时用来度量 X_i, 和 X_j, 满足 Z_{ij} 约束关系的程度大小。当 X_i, 和 X_j 完全匹配时，e=0.

为了简化符号，在后续的段落中，我们用如下的简化方式来编码误差指数函数：

$$\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) \overset{\text{def.}}{=} \mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \overset{\text{def.}}{=} \mathbf{e}_{ij}(\mathbf{x}). \quad (3)$$

注意到每个误差函数，每个参数块，每个误差函数可以跨越不同的区域。这种形式的问题可通过一个直接的图来高效的表示。图中的一个节点 i 可代表参数块 X_i，在节点 i 和 j 之间的边代表了两个参数块 X_i, 和 X_j 之间的有序的约束关系。图 2 表示的是图和目标函数构造的例子。



$$\begin{aligned}\mathbf{F}(\mathbf{x}) = &\mathbf{e}_{12}^\top\,\mathbf{\Omega}_{12}\,\mathbf{e}_{12}\\ &+\mathbf{e}_{23}^\top\,\mathbf{\Omega}_{23}\,\mathbf{e}_{23}\\ &+\mathbf{e}_{31}^\top\,\mathbf{\Omega}_{31}\,\mathbf{e}_{31}\\ &+\mathbf{e}_{24}^\top\,\mathbf{\Omega}_{24}\,\mathbf{e}_{24}\\ &+\dots\end{aligned}$$

Fig. 2. This example illustrates how to represent an objective function by a graph.

A 方差最小化

如果初始的参数估计 $\breve{\mathbf{x}}$ 是已知的，方程（2）的数值解就可以通过使用主流的高斯牛顿法或者 LM 算法获得。这种方法是通过在当前估计的初始位置 $\breve{\mathbf{x}}$ 处使用泰勒一阶展开式来得到近似的误差函数，

$$\begin{aligned}\mathbf{e}_{ij}(\breve{\mathbf{x}}_i+\mathbf{\Delta x}_i,\breve{\mathbf{x}}_j+\mathbf{\Delta x}_j) &= \mathbf{e}_{ij}(\breve{\mathbf{x}}+\mathbf{\Delta x}) &(4)\\ &\simeq \mathbf{e}_{ij}+\mathbf{J}_{ij}\mathbf{\Delta x}. &(5)\end{aligned}$$

其中，J_{ij} 是通过计算 $\breve{\mathbf{x}}$ 和 $\mathbf{e}_{ij}\overset{\text{det.}}{=}\mathbf{e}_{ij}(\breve{\mathbf{x}})$'得到的 $\mathbf{e}_{ij}(\mathbf{x})$ 的雅可比矩阵。将方程（5）带入方程（1）F_{ij} 的误差项，可以得到：

$$\begin{aligned}&\mathbf{F}_{ij}(\breve{\mathbf{x}}+\mathbf{\Delta x}) &(6)\\ &= \mathbf{e}_{ij}(\breve{\mathbf{x}}+\mathbf{\Delta x})^\top\mathbf{\Omega}_{ij}\mathbf{e}_{ij}(\breve{\mathbf{x}}+\mathbf{\Delta x}) &(7)\\ &\simeq (\mathbf{e}_{ij}+\mathbf{J}_{ij}\mathbf{\Delta x})^\top\mathbf{\Omega}_{ij}(\mathbf{e}_{ij}+\mathbf{J}_{ij}\mathbf{\Delta x}) &(8)\\ &= \underbrace{\mathbf{e}_{ij}^\top\mathbf{\Omega}_{ij}\mathbf{e}_{ij}}_{c_{ij}}+2\underbrace{\mathbf{e}_{ij}^\top\mathbf{\Omega}_{ij}\mathbf{J}_{ij}}_{\mathbf{b}_{ij}}\mathbf{\Delta x}+\mathbf{\Delta x}^\top\underbrace{\mathbf{J}_{ij}^\top\mathbf{\Omega}_{ij}\mathbf{J}_{ij}}_{\mathbf{H}_{ij}}\mathbf{\Delta x} &(9)\\ &= c_{ij}+2\mathbf{b}_{ij}\mathbf{\Delta x}+\mathbf{\Delta x}^\top\mathbf{H}_{ij}\mathbf{\Delta x} &(10)\end{aligned}$$

通过这种局部逼近，我们可以重写一下在方程（1）中的方程，如下所示：

$$\begin{aligned}\mathbf{F}(\breve{\mathbf{x}}+\mathbf{\Delta x}) &= \sum_{\langle i,j\rangle\in\mathcal{C}}\mathbf{F}_{ij}(\breve{\mathbf{x}}+\mathbf{\Delta x}) &(11)\\ &\simeq \sum_{\langle i,j\rangle\in\mathcal{C}}c_{ij}+2\mathbf{b}_{ij}\mathbf{\Delta x}+\mathbf{\Delta x}^\top\mathbf{H}_{ij}\mathbf{\Delta x} &(12)\\ &= c+2\mathbf{b}^\top\mathbf{\Delta x}+\mathbf{\Delta x}^\top\mathbf{H}\mathbf{\Delta x}. &(13)\end{aligned}$$

方程（13）是通过设置方程（12）中的 $c=\sum c_{ij}$，$\mathbf{b}=\sum\mathbf{b}_{ij}$ 和 $\mathbf{H}=\sum\mathbf{H}_{ij}$ 获得的。这个方程式可以进一步简化成一个线性的系统：

$$\mathbf{H}\mathbf{\Delta x}^* = -\mathbf{b}. \qquad (14)$$

其中 H 是系统的信息矩阵。这个解决方案是通过向初始的估计中加入一个增量来获得的。

$$\mathbf{x}^* = \breve{\mathbf{x}} + \Delta\mathbf{x}^*. \quad (15)$$

主流的高斯牛顿算法在方程（13）中进行迭代，求解方程（14）,并且在方程（15）中更新步长。在每一次迭代中，之前的结果被作为线性点和初始估计使用，直到，遇到一个给定的终止准则。

LM 算法在高斯牛顿法中引入了一个阻尼系数和备份方案，用来控制收敛。LM 用如下的阻尼版本来替代方程（14）

$$(\mathrm{H} + \lambda\mathrm{I})\,\Delta\mathbf{x}^* = -\mathrm{b}. \quad (16)$$

其中，λ 是一个阻尼系统，阻尼系数越高，$\Delta\mathbf{x}$ 就越小。这个可以控制步长防止出现非线性表面。LM 的算法其实是用来控制动态的阻尼系数。在每次迭代的时候，新配置所带来的误差都会被监控。如果新的误差比之前的误差小，在下次迭代过程中 λ 就将会相应的减小。否则，解决方案就相反 λ 会变大。对于更加详细的关于 LM 算法的实现，我们参考了[18]。

B 可选参数化

上面的方法描述的是通用的用来最小化多元函数的方法。他们都假设参数的是在欧几里得空间，但是对于一些像 SLAM 和 BA 问题来说，这些假设是无效的。为了处理分布在非欧几里得空间的状态变量，一个常用的方法就是在 X_i 中增加一个$\Delta\mathbf{x}_i$。

例如，在 SLAM 问题中，每一个参数块 $\mathbf{x}_i$ 都包含一个平移矢量 $\mathbf{t}_i$ 和一个旋转矢量 α_i。这个平移矢量构成欧几里得空间。相对的，旋转矩阵部分构成了非欧几里得 2D 或者 3D 旋转群 SO（2）或者 SO（3）。为了避免奇异性，这些空间通常是用多参数来描述的，例如，通过旋转矩阵和四元数。直接把方程（15）应用到多参数模型上会打破由多参数引导的约束关系。为了解决这个难题，可以在旋转中使用一个极小点代表（例如 3D 中的欧拉角）。这样才能满足奇异点。另一个想法是计算一个新的误差函数，在这个函数中$\Delta\mathbf{x}_i$表示当前变量点 $\breve{\mathbf{x}}_i$ 附近的扰动，其中 $\mathbf{x}_i$ 是多元函数中的一个点。由于$\Delta\mathbf{x}_i$通常都比较小，它们都远离奇异点。在优化之后得到的变量 $\mathbf{x}_i^*$ 的值可以通过使用如下的非线性增量操作 $\boxplus : \mathrm{Dom}(\mathbf{x}_i) \times \mathrm{Dom}(\Delta\mathbf{x}_i) \rightarrow \mathrm{Dom}(\mathbf{x}_i)$ 来获得：

$$\mathbf{x}_i^* = \breve{\mathbf{x}}_i \boxplus \Delta\mathbf{x}_i^*. \quad (17)$$

例如，在 3DSLAM 中，可以通过平移矢量和标准的坐标系的四元数来表示增量$\Delta\mathbf{x}_i$ 。位姿是通过一个平移矢量和全部四元数表示的。田运算应用在 $\mathbf{x}_i$ 的增量$\Delta\mathbf{x}_i$ 上是通过使用标准运

动合成算子 $\oplus$ （见[25]），在将增量转换成相同的状态变量：

$$\breve{\mathbf{x}}_i \boxplus \boldsymbol{\Delta}\mathbf{x}_i^* \overset{\text{def.}}{=} \breve{\mathbf{x}}_i \oplus \boldsymbol{\Delta}\mathbf{x}_i^*. \qquad (18)$$

通过上述的操作，一个新的误差方程可以定义如下：

$$\mathbf{e}_{ij}(\boldsymbol{\Delta}\mathbf{x}_i, \boldsymbol{\Delta}\mathbf{x}_j) \overset{\text{def.}}{=} \mathbf{e}_{ij}(\breve{\mathbf{x}}_i \boxplus \boldsymbol{\Delta}\mathbf{x}_i, \breve{\mathbf{x}}_j \boxplus \boldsymbol{\Delta}\mathbf{x}_j) \qquad (19)$$

$$= \mathbf{e}_{ij}(\breve{\mathbf{x}} \boxplus \boldsymbol{\Delta}\mathbf{x}) \simeq \mathbf{e}_{ij} + \mathbf{J}_{ij}\boldsymbol{\Delta}\mathbf{x}, \qquad (20)$$

其中 $\breve{\mathbf{x}}$ 跨越了原始的多参数空间。雅可比矩阵则表示为

$$\mathbf{J}_{ij} = \left.\frac{\partial \mathbf{e}_{ij}(\breve{\mathbf{x}} \boxplus \boldsymbol{\Delta}\mathbf{x})}{\partial \boldsymbol{\Delta}\mathbf{x}}\right|_{\boldsymbol{\Delta}\mathbf{x}=\mathbf{0}}. \qquad (21)$$

由于增量 $\boldsymbol{\Delta}\bar{\mathbf{x}}^*$ 是在局部欧几里德环境中的初始位置计算得来的，它们需要通过田运算重新回到原来的那个冗余空间。

我们的框架允许一些简单的不同空间增量和状态变量的定义，因此，也支持同一个问题间的任意参数化。无论如何选择参数化，黑塞矩阵的结构一直保留着。

C 线性化系统的结构

通过方程（13），矩阵 H 和向量 b 是通过求一系列的矩阵和向量的和得到的，一个用于每一种约束。如果我们定义 $\mathbf{b}_{ij} = \mathbf{J}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}$ 和 $\mathbf{H}_{ij} = \mathbf{J}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}$，那么 b 和 H 就可以表示成：

$$\mathbf{b} = \sum_{\langle i,j\rangle \in \mathcal{C}} \mathbf{b}_{ij} \qquad \mathbf{H} = \sum_{\langle i,j\rangle \in \mathcal{C}} \mathbf{H}_{ij}. \qquad (22)$$

每个约束都是通过一个附加项作用在这个系统上。附加相的结构取决于误差函数的雅可比矩阵。由于误差函数的约束至取决于图中两个节点值，所以方程（5）中的雅可比矩阵的形式如下：

$$\mathbf{J}_{ij} = \begin{pmatrix} 0 \cdots 0 & \underbrace{\mathbf{A}_{ij}}_{i} & 0 \cdots 0 & \underbrace{\mathbf{B}_{ij}}_{j} & 0 \cdots 0 \end{pmatrix}. \qquad (23)$$

其中 $\mathbf{A}_{ij}$ 和 $\mathbf{B}_{ij}$ 是关于 $\boldsymbol{\Delta}\breve{\mathbf{x}}_i$ 和 $\boldsymbol{\Delta}\mathbf{x}_j$ 的误差函数。从方程（9）中我们获得了 $\mathbf{H}_{ij}$ 的分块矩阵如下所示：

$$\mathbf{H}_{ij} = \begin{pmatrix} \ddots & & \\ & \mathbf{A}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} \cdots \mathbf{A}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} & \\ & \vdots \qquad\qquad \vdots & \\ & \mathbf{B}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} \cdots \mathbf{B}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} & \\ & & \ddots \end{pmatrix} \quad \mathbf{b}_{ij} = \begin{pmatrix} \vdots \\ \mathbf{A}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij} \\ \vdots \\ \mathbf{B}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij} \\ \vdots \end{pmatrix}$$

为了简化我们省略了 0 矩阵块。正如读者所见，分块矩阵 H 是图的邻接矩阵。因此，H 矩阵块中非零的矩阵块数目与图的边成正比，这就导致了 H 矩阵的稀疏性。在 G2O 中，我们利用先进的方法和 H 矩阵的稀疏性来解决方程（14）的线性系统的问题。

D 有特殊结构的系统

一些例如 BA 的问题是的 H 矩阵有很多结构上的特性。我们的系统能够利用这些特殊的性能提高效果。在 BA 问题中通常有两种变量，一个是相机的位置 P，另一个是通过相机获取的地标的位置 L。通过重组方程（14）中的变量使得计算相机的位置的指数降低，我们得到了如下方程：

$$\begin{pmatrix} \mathbf{H}_{\mathbf{PP}} & \mathbf{H}_{\mathbf{Pl}} \\ \mathbf{H}_{\mathbf{Pl}}^\top & \mathbf{H}_{\mathbf{ll}} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{x}_{\mathbf{P}}^* \\ \Delta \mathbf{x}_{\mathbf{l}}^* \end{pmatrix} = \begin{pmatrix} -\mathbf{b}_{\mathbf{P}} \\ -\mathbf{b}_{\mathbf{l}} \end{pmatrix}. \qquad (24)$$

可以看出，一个等价减少的系统可以通过 H 矩阵的舒尔补获取。

$$\left(\mathbf{H}_{\mathbf{PP}} - \mathbf{H}_{\mathbf{Pl}} \mathbf{H}_{\mathbf{ll}}^{-1} \mathbf{H}_{\mathbf{Pl}}^\top\right) \Delta \mathbf{x}_{\mathbf{P}}^* = -\mathbf{b}_{\mathbf{P}} + \mathbf{H}_{\mathbf{Pl}} \mathbf{H}_{\mathbf{ll}}^{-1} \mathbf{b}_{\mathbf{l}}. \quad (25)$$

我们发现，计算 $\mathbf{H}_{\mathbf{ll}}^{-1}$ 是很简单的，因为 $\mathbf{H}_{\mathbf{ll}}$ 是一个对角矩阵块。求解方程（25）我们可以得到相机的 $\Delta \mathbf{x}_{\mathbf{P}}^*$ 增量，使用这个增量，我们可以得到：

$$\mathbf{H}_{\mathbf{ll}} \Delta \mathbf{x}_{\mathbf{l}}^* = -\mathbf{b}_{\mathbf{l}} - \mathbf{H}_{\mathbf{Pl}}^\top \Delta \mathbf{x}_{\mathbf{P}}^*, \qquad (26)$$

这使得 $\Delta \mathbf{x}_{\mathbf{l}}^*$ 成为调整观测量特征的一个值。实际上真实的观测场景的特征比相机获得的特征值要多很多，因此方程（25）会比方程（14）处理的速度更快一些，尽管在方程（25）中会花费额外的时间用来计算方程左边的矩阵。

4 执行（实现）

使用 C++的目的是为了比通常所用的方法获得更快的执行速度。我们实现这一目标的方法是在我们的图中抽象出一些点和边的基类。虽然大多数内部操作都使用模版参数来提高效率，但这两个基类都提供了虚函数来方便使用者进行子类化。我们使用线性代数包（eigen），这个包在其他优化技术上应用了 SSE 算法，例如惰性评估和循环展开来提高系统的性能。

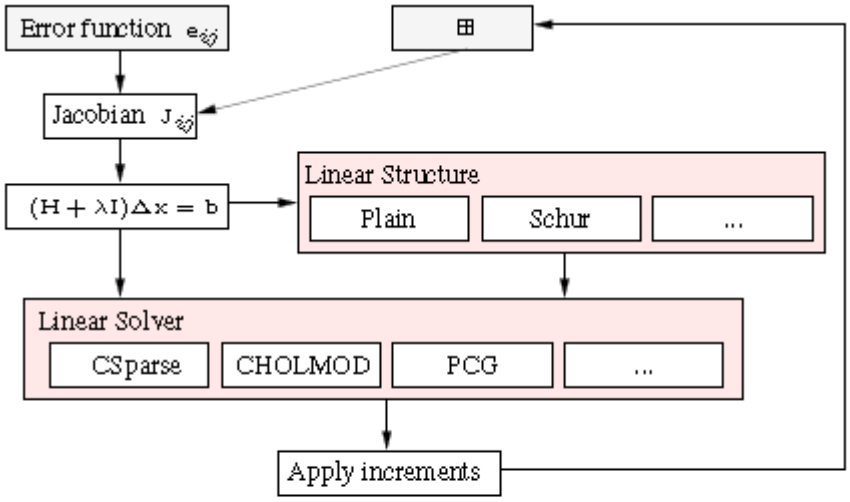


Fig. 3. Overview of our framework. For addressing a new optimization problem, only the boxes in gray need to specified. Furthermore, the framework allows to add different linear solvers.

图 3 展示了我们系统的设计。图中，只有灰色的部分需要被定义以解决新的优化问题。使用基类的到一个新的节点的类型只需要对增量定义一个田操作。一条边连接这 X_i 和 X_j 两个节点，同时也需要定义误差函数 $\mathbf{e}_{ij}(\cdot)$。雅可比矩阵 $\mathbf{J}_{ij}$ 是用来计算数值或者为了获得更高的计算效率，用户可以通过重新编写虚拟的基类函数来明确的指定雅可比矩阵 $\mathbf{J}_{ij}$。因此，创建一个新的类型用于新的优化问题或者比较不同参数化问题只需要写几行代码即可。

通常矩阵 H 的计算要使用一个变化的矩阵。如果变量的维度是已知的，我们的框架是允许大小固定的矩阵的运算的。利用先前已知的维度会优化像循环矩阵乘法的编译时间。

注意在需要用舒尔减少来实现矩阵乘法的方程（25）。用基本图的稀疏结构来乘以非零的部分从而得到 $\mathbf{H}_{\mathbf{PP}}$ 矩阵。此外，我们在底层矩阵中处理了结构块，通过对比一个标量的矩阵乘积，这种方法可以提高矩阵乘积的存储效率。

我们的框架对于嵌入式线性规划求解器是未知的，所以针对不同的问题我们选择不同的方法。在实验中我们使用两种求解器。由于 H 是半正定和堆成的，稀疏的 Cholesky 分解会得到一个有效的分解结果[4],[3].注意到最小方差的迭代在非零模式下是一个常量。因此我们就可以重新使用第一次迭代计算的结果，这样可以较少不必要的后续迭代从而减少整个过程的时间消耗。值得注意的是 Cholesky 分解并不利用参数矩阵块结构。第二种方法是共轭条件梯度（PCG）和一个预制的雅可比矩阵调节器，雅可比矩阵块可以利用矩阵块间的运算。由于 PCG 算法本身就是一个迭代的算法，所以在计算 N*N 的矩阵的时候需要 N 次迭代。由于执行 N 次 PCG 迭代比 Cholesky 分解明显慢很多，所以，我们依照 PCG 中平方剩余最小原则来减少迭代的次数。通过这种方法我们能够及早的量化引入 PCG 终止所带来的精度损失。在实验中，我们会比较不同的求解器。

5 实验

在这个部分，我们将 G2O 与先进的优化方法在真实数据和人工数据集中做了对比。

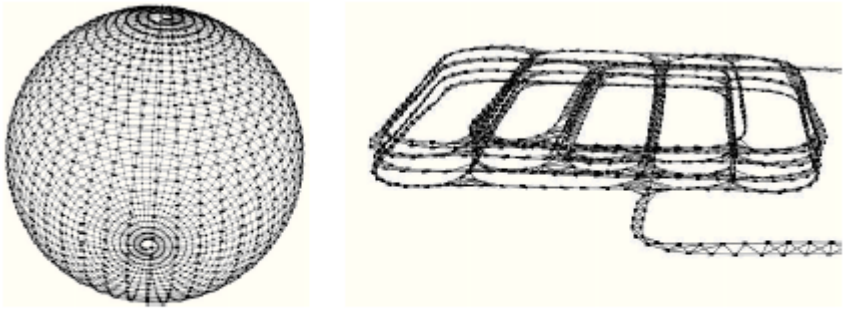


Fig. 4. 3D pose-graph datasets used for evaluating $\mathbf{g^2o}$: the left image shows a simulated sphere, the right image depicts a partial view of a real-world dataset of a multi-level parking garage.

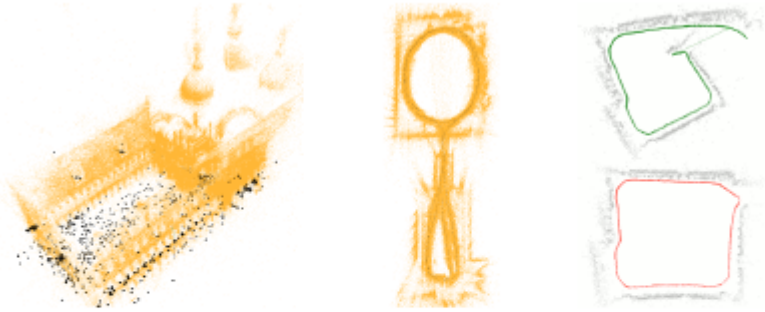


Fig. 5. The BA real world datasets and the scale-drift dataset used for evaluating $\mathbf{g^2o}$: the left image shows the Venice dataset, whereas the middle image depicts the New College dataset [24]. The pair of images on the right shows the Keble college dataset which was processed by monocular SLAM. Here, scale drift occurs (top) which can be corrected when closing the loop using 7 DoF similarity transformations (bottom).

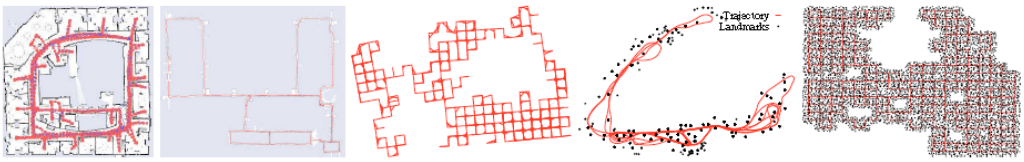


Fig. 6. 2D Datasets used for evaluating $\mathbf{g^2o}$. From left to right: 2D pose-graph of the Intel Research Lab; 2D pose-graph of the Killian Court; Manhattan3500, a simulated pose-graph; 2D dataset with landmarks of the Victoria Park; and Grid5000, a simulated landmark dataset.

图 4 展示的是 3D 位姿图，图 5 展示的是克伯学院使用 7 自由度相关约束的大规模漂移 SLAM 的位姿图。图 6 展示的是一个 2D 的数据集。其中每一个数据集变量的数目和约束条件数目都在表 1 中可也看到。所以的实验都是在同一台电脑上进行的（Intel Core i7-930，2.8 Ghz）。

TABLE I

OVERVIEW OF THE TEST DATASETS.

Dataset	# poses	# landmarks	# constraints
Intel	943	-	1837
MIT	5489	-	7629
Manhattan3500	3500	-	5598
Victoria	6969	151	10608
Grid5000	5000	6412	82486
Sphere	2500	-	4949
Garage	1661	-	6275
Venice	871	530304	2838740
New College	3500	488141	2124449
Scale Drift	740	-	740

我们将 G2O 与其他先进的算法比较了一下：使用开源的$\sqrt{\text{SAM}}$ 、SAP、SSBA、和机器视觉。

实验结果表明这些方法只能解决一些特定的优化问题，而 G2O 可以解决所有这些问题并且可以很容易的扩展到一些新问题的解决。

A 运行时间比较

下面我们展示的是每一种方法进行一次迭代所需要的时间，我们给每一种方法提出一个全局优化的问题，并且进行 10 次迭代，并且量测了每一次迭代的平均时间。在这种实验条件下，G2O 使用 CHOLMOD 包和 Cholesky 分解来处理这个线性系统。这个 CHOMOD 包也用于实验中其他的方法。因此，求解线性系统索要的时间对于所有的方法来说没有很大差别，但是区别在于构造线性系统的快慢上。图 7 展示了实验的结果。

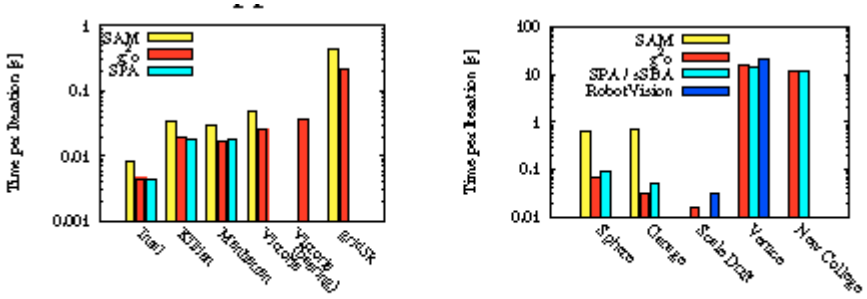


Fig. 7. Time per iteration for each approach on each dataset.

在我们测试的数据中，G2O 系统在 2D 和 3D 的数据集合上效果比$\sqrt{\text{SAM}}$ 要更快、运行时间更短。理论上，这两种方法完成的是相同的算法，但是 G2O 利用一个高效的前端来构造线性化问题。在 2D 的位姿图数据中，G2O 框架完成的效率可以和最有的 SPA 算法相媲美。在 BA 数据集下，G2O 的效果和 sSBA 的完成效果相类似，sSBA 算法比我们的框架要快一点点。对比机器人视觉，G2O 平均运行时间快了 2 倍多。

注意到 G2O 专注于批量的优化，但是它也可以用来逐步优化已经加入到图中的节点。G2O 的效率和用于增量的方法例如 ISAM[14]或者 HOG-MAN[9]是相似的。如图 8 所示，通过每 10 个节点优化一次或者是在加入一个节点后通过释放 PCG 的终止准则来进行优化，G2O 的运行时间都

是可以接受的。还有，G2O 可以用来更有效的构造一个更复杂的在线系统块[21]，[9]。

正如提及的，G2O 可以计算雅可比矩阵，它可以构造新的优化问题的原型或者一个新的误差函数。然而，通过确定雅可比矩阵可以很大程度上加快运行速度。例如，G2O 在 GARAGE 数据库里的一次迭代所需要的时间如果在解析或者指定的雅可比矩阵下就会从原来的 80 毫秒降低到 40 毫秒。当使用数字雅可比矩阵的时候，我们的运行时间减少了，但是精度并没有降低。

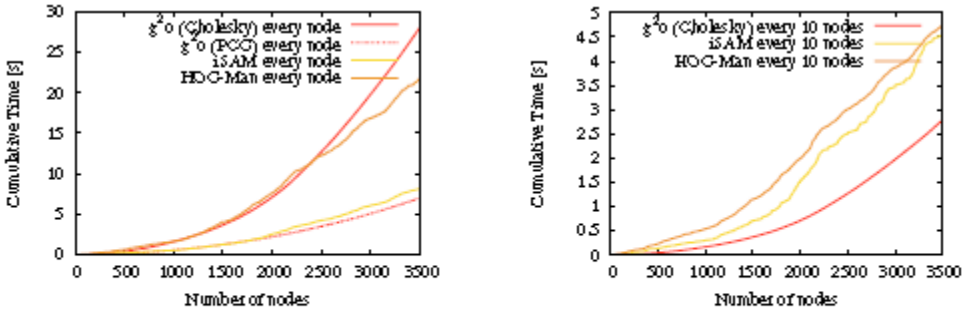


Fig. 8. Online processing of the Manhattan3500 dataset. The left image shows the runtime for optimizing after inserting a single node whereas the right image show the runtime for optimizing after inserting 10 nodes.

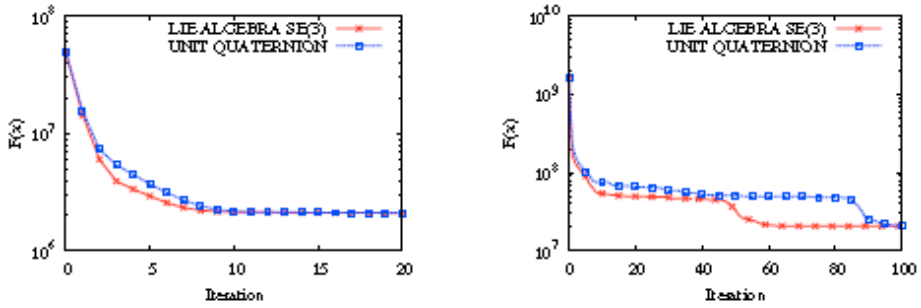


Fig. 9. Evolution of $\mathbf{F}(\mathbf{x})$ using unit quaternions versus the Lie algebra $se(3)$ on the New College dataset (left) and Venice (right) dataset.

TABLE II

COMPARISON OF DIFFERENT LINEAR SOLVERS (TIME IN SECONDS).

Dataset	CHOLMOD	CSparse	PCG
Intel	0.0028	**0.0025**	0.0064 ± 0.0026
MIT	0.0086	**0.0077**	0.381 ± 0.364
Manhattan3500	0.018	0.018	**0.011 ± 0.0009**
Victoria	0.026	**0.023**	1.559 ± 0.683
Grid5000	**0.178**	0.484	1.996 ± 1.185
Sphere	0.055	0.398	**0.022 ± 0.019**
Garage	0.019	0.032	**0.017 ± 0.016**
New College	6.19	200.6	**0.778 ± 0.201**
Venice	1.86	39.1	**0.287 ± 0.135**
Scale Drift	0.0034	**0.0032**	0.005 ± 0.01

B 测量不同的参数化

由于我们的框架之需要输入误差函数和更新步数，所以我们可以很容易的比较不同的参数化的结果。为了这个目的，我们在 BA 中用两种不同的参数化来代表位姿。在第一种参数化中，增量Δx_i是通过一个平移矢量和一个轴四元数表示。在第二种参数化中，我们使用 Lie algebra se(3)来表示增量Δx_i。我们在牛津大学新学院和威尼斯的数据集上测试了这两种参数化，实验结果如图 9 所示：两种参数化收敛得到了相同的结果，但是 se(3)参数集收敛的速度更快一些。

C 线性方法的比较

我们的系统适用于不同线性求解器来求解方程(14)或者方程(25)。目前我们实现了基于 Cholesky 分解的两种解决器，也就是 CHOLMOD 和 CSparse[4]。此外，我们实现的 PCG 作为一种使用预制条件的雅可比矩阵块的迭代方法。表 2 总结了在一些数据集上解决线性系统问题所需要的时间。PCG 在牛津新学院和威尼斯数据集上有很好的效果，它的执行时间比 CHOLMOD 要快 7 倍多。PCG 的收敛取决于初始估计的点和优化点之间的接近程度。当相对剩余低于一个约束值时（在实验中取值为 10^{-8}）我们就终止 PCG 过程。因此，PCG 需要更多的时间进行收敛，例如，在麻省理工学院或者维多利亚的数据集中，CHOLMOD 在大数据上的处理速度要比 CSparse 快 30 倍以上。但是令人吃惊的是，CSparse 在较小的数据集上，例如麻省理工学院数据集，它要比 CHOLMOD 和 PCG 的效果都好。

TABLE III

COMPARISON OF DIFFERENT LINEAR SOLVERS. WE MEASURED THE AVERAGE TIME PER ITERATION OF $\mathbf{g^2o}$ (IN SECONDS).

Dataset	direct solution solve	Schur decomposition build / solve / total
Victoria	0.026	0.029 / 0.121 / 0.150
Grid5000	0.18	0.12 / 0.16 / 0.28
New College	15.18	3.37 / 7.07 / 10.44
Venice	33.87	11.25 / 1.78 / 13.03

D 利用的知识结构

正如在第三部分 D 中讨论的，确定的问题有特有的结构。使用这种结构可以本质上的提高解决线性系统的速度。基于地标的 SLAM 和 BA 都有相同的线性结构：地标或者点只和机器人的位置/相机有关系，从而形成了地标的对角矩阵块，也是了海塞矩阵的一部分$\mathbf{H}_{ll}$。

实验中我们评估了针对基于地标的 SLAM 和 BA 的具体分解的优势。表三展示了在有何没有分解的条件下不同数据集的运行时间。从表三中可以明显的看出，当地标数目超出位置时，使用分解可以本质上提高运行速度，这个结果在 BA 中尤其明显。然而，当位置的数量很大时，使用舒尔边缘化会导致一个高度连接系统，这个系统会稍微缩小，但是不容易解决。

6 结论

本文我们描述了一个 G2O 框架，它是一个可扩展的、高效的用于函数批量优化的开源框架。这个框架可以嵌入图中。 这个方面相关的问题是基于图优化的 SLAM 和 BA 问题，这是两种基础的并且和机器学以及计算机视觉高度相关的。使用 G2O 的时候，用户只需要定义一个误差函数和一个应用程序来干扰当前的解决方案。此外，使用者还可以针对不同的问题很方便的在图中嵌入一个线性求解器。我们展示了 G2O 用于不同变量的 SLAM 问题（2D，3D ，只有位姿和有地标的情况）和 BA 问题。实验的结果是建立在大量的数据之上的结果表明，G2O 能够和解决专门问题的算法达到一样的效果，有的效果甚至更优。整个开源的系统作为 ROS 的一部分是可以在 OpenSLAM.org 中免费获取。

鸣谢

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

参考文献

[1] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, “Bundle adjustment in the large,” in Proc. of the European Conf. on Computer Vision (ECCV), 2010.

[2] M. Byrod and K. Astroem, “Conjgate gradient bundle adjustment,” in Proc. of the European Conf. on Computer Vision (ECCV), 2010.

[3] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, “Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate,” ACM Trans. Math. Softw., vol. 35, no. 3, pp. 1–14, 2008.

[4] T. A. Davis, Direct Methods for Sparse Linear Systems. SIAM, 2006, part of the SIAM Book Series on the Fundamentals of Algorithms.

[5] F. Dellaert and M. Kaess, “Square Root SAM: Simultaneous localization and mapping via square root information smoothing,” Int. Journal of Robotics Research, vol. 25, no. 12, pp. 1181–1204, Dec 2006.

[6] T. Duckett, S. Marsland, and J. Shapiro, “Fast, on-line learning of globally consistent maps,” Autonomous Robots, vol. 12, no. 3, pp. 287 – 300, 2002.

[7] U. Frese, “A proof for the approximate sparsity of SLAM information matrices,” in Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2005.

[8] U. Frese, P. Larsson, and T. Duckett, “A multilevel relaxation algorithm for simultaneous localisation and mapping,” IEEE Transactions on Robotics, vol. 21, no. 2, pp. 1–12, 2005.

[9] G. Grisetti, R. K ̈ummerle, C. Stachniss, U. Frese, and C. Hertzberg, “Hierarchical optimization on manifolds for online 2d and 3d mapping,” in Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2010.

[10] G. Grisetti, C. Stachniss, and W. Burgard, “Non-linear constraint network optimization for efficient map learning,” IEEE Trans. on Intelligent Transportation Systems, 2009.

[11] G. Guennebaud, B. Jacob, et al., “Eigen v3,” http://eigen.tuxfamily.org, 2010.

[12] A. Howard, M. Matari ́c, and G. Sukhatme, “Relaxation on a mesh:

a formalism for generalized localization," in Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2001.
[13] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I. Kweon, "Pushing the envelope of modern methods for bundle adjustment," in Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR), 2010.
[14] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," IEEE Trans. on Robotics, vol. 24, no. 6, pp. 1365–1378, Dec 2008.
[15] K. Konolige, "Sparse sparse bundle adjustment," in Proc. of the British Machine Vision Conference (BMVC), 2010.
[16] K. Konolige, G. Grisetti, R. K ¨ummerle, W. Burgard, B. Limketkai, and R. Vincent, "Sparse pose adjustment for 2d mapping," in Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2010.
[17] K. Konolige, "Large-scale map-making," in Proc. of the National Conference on Artificial Intelligence (AAAI), 2004.
[18] M. A. Lourakis and A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment," ACM Trans. Math. Software, vol. 36, no. 1, pp. 1–30, 2009.
[19] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," Autonomous Robots, vol. 4, pp. 333–349, 1997.
[20] M. Montemerlo and S. Thrun, "Large-scale robotic 3-d mapping of urban structures," in Proc. of the Int. Symposium on Experimental Robotics (ISER), 2004.
[21] K. Ni and F. Dellaert, "Multi-level submap based SLAM using nested dissection," in Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2010.
[22] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2006, pp. 2262–2269.
[23] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, Numerical Recipes, 2nd Edition. Cambridge Univ. Press, 1992.
[24] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," Int. Journal of Robotics Research, vol. 28, no. 5, pp. 595–599, May 2009.
[25] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial realtionships in robotics," in Autonomous Robot Vehicles, I. Cox and G. Wilfong, Eds. Springer Verlag, 1990, pp. 167–193.
[26] H. Strasdat, J. M. M. Montiel, and A. Davison, "Scale drift-aware large scale monocular SLAM," in Proc. of Robotics: Science and Systems (RSS), 2010.
[27] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzibbon,

“Bundle adjustment - a modern synthesis,” in Vision Algorithms:
Theory and Practice, ser. LNCS. Springer Verlag, 2000, pp. 298–375.

朱小英 译
2016 年 5 月 11 日