

## 计算机视觉 life 平台介绍

计算机视觉 life 是国内较早分享「计算机视觉」技术的原创平台，公众号订阅用户 **11 万+**。同时有 **微信视频号**、**哔哩哔哩**、**知乎**、**今日头条** 同名账户，全网订阅用户 **26 万+**。在行业内积累了极高的知名度，受到不少从业者的好评和推荐。

平台主要面向智能机器人、自动驾驶汽车、无人机、增强现实、计算摄影、视觉 / 激光 / 惯性传感器、计算机视觉等方向提供前沿技术分享、系统教学课程。

主要业务如下：

### 1、AI 职业教育

面向工业实战的源码解析课程，官网 [cvlife.net](http://cvlife.net)

- 视觉 **SLAM**: ORB-SLAM2 (必学基础)
- 视觉惯性 **SLAM**: ORB-SLAM3, VINS-Mono+Fusion
- 激光 / 多传感器融合 **SLAM**: Cartographer, LeGO-LOAM、LIO-SAM、LVI-SAM
- 三维重建：视觉几何 OpenMVS，深度学习重建 MVSNet、PatchMatchNet、JDACS-MS；神经辐射场 NeRF 重建
- 机器人运动规划：Dijkstra、A-Star、DWA、TEB
- 相机标定：单目 / 鱼眼 / 双目 / 阵列
- 视觉深度估计：单目 / 双目深度估计
- C++ 编程入门
- 计算摄影（敬请期待）
- 深度学习 **SLAM**（敬请期待）
- 基于滤波的 **SLAM**（敬请期待）



扫描开始学习↑

## 2、机器人 SLAM 开发者社区

我们建立了**全国最大的机器人 SLAM 开发者交流社区**，2018 年开始运营至今，已经有 **3900+会员**、**6500+主题分享**、**9500+问答评论**、**130+教学视频**。

- **领域**：机器人 SLAM、三维视觉、自动驾驶、增强现实、无人机、图像处理
- **直播**：**每月 6/16/26 日**固定直播，涵盖大佬分享、学习经验、求职面试、实习历程、行业内幕
- **教程**：图文视频教程：涉及代码调试、OpenCV、PCL、G2O、Ceres、视觉 SLAM 十四讲、LVISAM、R3LIVE、Fastlio2、GVINS、机器人学中的状态估计、语义 SLAM 论文阅读……
- **答疑**：每日星主答疑、嘉宾答疑、星友互助答疑，搜索关键字几乎所有问题都能找到答案
- **资讯**：每日论文翻译、行业资讯汇总、每周汇总、精华汇总
- **活动**：学习小组、行业资源对接、会员激励、有偿招募助教/兼职
- **求职**：经验分享、内推职位、SLAM 面试题、笔试练习



扫描领取优惠券 ↑

招聘、产品宣传等商务合作，联系 simiter@126.com

# ORB-SLAM3 源码逐行解析

## 目录

第 1 章 ORB-SLAM3 简介与运行	1
算法概述	1
效果展示	1
算法框架图	2
ORB-SLAM3 和 ORB-SLAM2 对比	3
源码注释地址	4
变量命名规范	4
第 2 章：ORB-SLAM2 重点知识回顾	5
第 3 章：IMU 介绍及预积分推导	5
3.0 IMU 简介及噪声模型	5
陀螺仪	6
加速度计	7
3.1 推导前的公式	8
3.2 预积分	9
3.3 噪声分离	10
3.3.1 对于 $\Delta R_{ij}$ 项	10
对于 $\Delta R_{ij}$ 项	12
3.3.3 对于 $\Delta p_{ij}$ 项	13
3.4 噪声递推	13
3.4.1 对于 $\delta\phi_{ij}$ 项	14
3.4.2 对于 $\delta v_{ij}$ 项	14
3.4.3 对于 $\delta p_{ij}$ 项	14
3.5 Bias 更新时对预积分的影响	15
3.5.1 对于 $\Delta R_{ij}$ 项	15
3.5.2 对于 $\Delta v_{ij}$ 项	16
3.5.3 对于 $\Delta p_{ij}$ 项	17
3.6 求残差关于状态量的雅可比	18
3.6.1 定义残差	18
3.6.2 定义扰动	18
3.6.3 残差对于状态的雅可比	19
第 4 章 IMU 预积分、初始化及优化代码解析	23
第 5 章：相机抽象模型	23
相机模型	23
KB8 模型的投影过程	23
KB8 模型的反投影过程	24
第 6 章：跟踪线程	25

跟踪线程的目的和意义 .....	25
跟踪线程的流程图 .....	26
跟踪线程的新变化 .....	26
参考关键帧的跟踪新变化 .....	28
恒速模型跟踪新变化 .....	28
重定位的新变化 .....	28
局部地图跟踪的新变化 .....	29
插入关键帧 .....	30
跟踪中的注意事项 .....	30
第 7 章：局部建图线程 .....	33
局部建图线程的目的和意义 .....	33
局部建图线程的流程框架 .....	35
局部建图线程中 IMU 初始化的三个阶段 .....	35
ORB-SLAM2/3 局部建图线程流程对比 .....	36
第 8 章：多地图系统、地图保存和加载 .....	37
多地图基本概念 .....	38
多地图作用和效果 .....	39
创建新地图的标准 .....	41
多地图中的重定位 .....	42
ORB-SLAM3 中的地图融合 .....	42
代码解析 .....	43
如何新建地图？ .....	43
什么时候新建地图？ .....	44
地图保存和加载 .....	46
boost 序列化示例 .....	46
第 9 章 闭环及地图融合线程 .....	48
回顾 ORB-SLAM2 中的闭环过程 .....	50
第 1 步：检测闭环。DetectLoop() .....	50
第 2 步：计算 Sim3。ComputeSim3() .....	51
第 3 步：闭环矫正。CorrectLoop() .....	51
ORB-SLAM3 的闭环和地图合并 .....	52
第 1 步：检测共同区域 .....	52
第 1 步：寻找候选关键帧并求解位姿变换。 .....	52
第 2 步：地图融合 .....	56
第 3 步：闭环矫正 .....	60
第 10 章 图优化 .....	60
边缘化 .....	60
IMU 模式下单帧优化 .....	61
跟踪关键帧 .....	61
跟踪普通帧 .....	62
第 11 章 不同模式适用场景及未来研究方向 .....	62
不同传感器模式对比及应用场景 .....	63
直接法 vs. 特征点法 .....	63
直接法 .....	64

特征点法 .....	64
代码 bug 梳理 .....	65
工程化建议 .....	70
研究方向建议 .....	71
第 12 章 实战：稠密重建及课程大作业 .....	72

该内容是公众号 **计算机视觉 life** 旗下课程第③期《[VIO 天花板：ORB-SLAM3 原理剖析+逐行源码详解](#)》的课件编纂成的课程手册。持续更新。课程官网：[cvlife.net](#)

作者：1、小六，多年视觉 SLAM 从业经验、计算机视觉 life 公众号创始人。课程讲解细致，善于将复杂的原理用图表具象化，帮助学员快速理解复杂的源码并理解背后的物理意义，教学耐心仔细，广受学员好评。也是《[视觉 SLAM 必学基础：ORB-SLAM2 源码详解](#)》课程讲师。

2、老白，擅长视觉 SLAM，VIO，以及多传感器融合，熟悉框架 ORB-SLAM 系列，三年工作经验，曾在自动驾驶高精度地图公司工作，目前就职于知名机器人公司。

说明：本课程注释源码会持续更新在

[https://github.com/electech6/ORB\\_SLAM3\\_detailed\\_comments](https://github.com/electech6/ORB_SLAM3_detailed_comments)。由于笔者水平有限，编写过程难免存在错误或疏漏，读者如发现有错误请发送邮件至 [810855028@qq.com](mailto:810855028@qq.com) 告知，笔者会及时更改，提前感谢！

修订时间：2022 年 10 月 20 日

本课件后续更新电子版会同步到 ORB-SLAM3 课程群。请咨询下方客服。

扫描一下，或登录 [cvlife.net](#) 系统学习视觉/激光/多传感器融合 SLAM、三维重建等精品课程：



扫描学习SLAM、三维重建

关于课程及交流群：如果想要咨询、购买计算机视觉 life 课程，请添加下面微信，备注：“**咨询或已购课程**”。请按照格式备注，否则不予通过。



# 第1章 ORB-SLAM3简介与运行

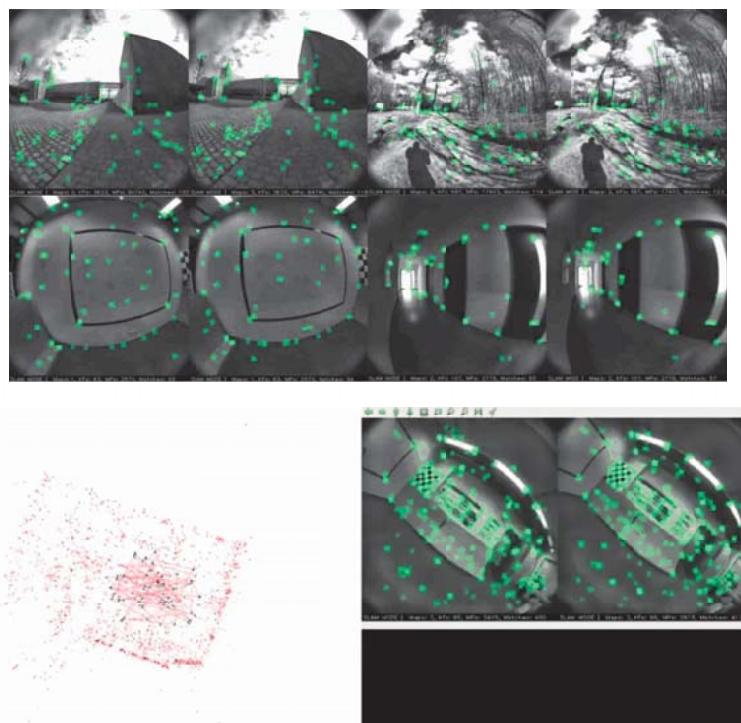
我们假设你已经系统学习过ORB-SLAM2：《视觉SLAM必学基础：ORB-SLAM2源码详解》。这里我们只关注ORB-SLAM3新增的内容，挑重点进行原理和代码解析。

## 算法概述

ORB-SLAM3是在ORB-SLAM2的基础上开发的，于2020年7月发布。它在定位精度和效果上几乎碾压了同类的开源算法，受到极大关注。从论文中我们先领略一下该算法的特点：

- 第一个可以运行视觉、视觉惯性和多地图，支持单目、双目和RGB-D相机，且支持针孔和鱼眼镜头模型的SLAM系统。
- 一个完全依赖最大后验估计，且基于特征点的单目和双目视觉惯性SLAM系统，包括在IMU初始化阶段。该算法可以在不同大小，室内和室外环境中鲁棒、实时的运行，精度上相比于以前的方法提升了2~5倍。和其他视觉惯性SLAM方法相比，即使在没有闭环的情况下，也具备良好的鲁棒性和更高的精度。
- 一个依赖高召回率的新位置识别算法的多地图系统。该多地图系统可以让系统在视觉信息缺乏的场景下长时间运行。比如当跟踪丢失的时候，它会重新建立新的地图，并在重新访问之前的地图时，无缝地与之前的地图融合。
- 与只使用最近几秒钟信息的视觉里程计相比，它是第一个能够在所有算法阶段重用之前所有信息的系统。能够在BA优化共视关键帧时，利用较大视差的观测来提高求解精度。这些大视差的观测可能间隔时间比较久，或者来自前面已经建立的不同的地图。
- 实验结果证明，在所有的传感器模式下，ORB-SLAM3和文献中可用的最优SLAM系统一样鲁棒，并且具有更高的精度。双目惯性模式下，该算法在无人机数据集EuRoC上可以达到平均3.6cm的定位精度，在手持设备快速移动的室内数据集TUM-VI上达到了9mm的定位精度。

## 效果展示



## 目前主流的视觉和视觉惯性SLAM、VO系统对比

从好到差：Excellent, Very good, Good, Fair

KLT( (Kanade-Lucas-Tomasi)：光流跟踪；Shi-Tomasi 角点检测，史建波，Good Features to Track；Thumbnail：索引图像

Table I: Summary of the most representative visual (top) and visual-inertial (bottom) systems, in chronological order.

	SLAM or VO	Pixels used	Data association	Estimation	Relocalization	Loop closing	Multi Maps	Mono	Stereo	Mono IMU	Stereo IMU	Fisheye	Accuracy	Robustness	Open source
Mono-SLAM [13], [4]	SLAM	Shi Tomasi	Correlation	EKF	-	-	-	✓	-	-	-	-	Fair	Fair	[15] <sup>1</sup>
PTAM [16]–[18]	SLAM	FAST	Pyramid SSD	BA	Thumbnail	-	-	✓	-	-	-	-	Very Good	Fair	[19]
LSD-SLAM [20], [21]	SLAM	Edgelets	Direct	PG	-	FABMAP PG	-	✓	✓	-	-	-	Good	Good	[22]
SVO [23], [24]	VO	FAST+ Hi.grad.	Direct	Local BA	-	-	-	✓	✓	-	-	✓	Very Good	Very Good	[25] <sup>2</sup>
ORB-SLAM2 [2], [3]	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	-	✓	✓	-	-	-	Exc.	Very Good	[26]
DSO [27]–[29]	VO	High grad.	Direct	Local BA	-	-	-	✓	✓	-	-	✓	Good	Very Good	[20]
DSM [31]	SLAM	High grad.	Direct	Local BA	-	-	-	✓	-	-	-	-	Very Good	Very Good	[22]
MSCKF [33]–[36]	VO	Shi Tomasi	Cross correlation	EKF	-	-	-	✓	-	✓	✓	-	Fair	Very Good	[37] <sup>3</sup>
OKVIS [38], [39]	VO	BRISK	Descriptor	Local BA	-	-	-	-	✓	✓	✓	-	Good	Very Good	[40]
ROVIO [41], [42]	VO	Shi Tomasi	Direct	EKF	-	-	-	-	✓	-	-	-	Good	Good	[43]
ORB-SLAM-VI [4]	SLAM	ORB	Descriptor	Local BA	DBoW2 PG+BA	DBoW2 PG+BA	-	✓	✓	✓	-	-	Very Good	Very Good	-
VINS-fusion [7], [44]	VO	Shi Tomasi	KLT	Local BA	DBoW2 PG	DBoW2 PG	✓	-	✓	✓	✓	✓	Very Good	Exc.	[45]
VI-DSO [46]	VO	High grad.	Direct	Local BA	-	-	-	-	✓	-	-	-	Very Good	Exc.	-
BASALT [47]	VO	FAST	(LSSD)	Local BA	-	ORB BA	-	-	-	-	✓	-	Very Good	Exc.	[48]
Kimera [8]	VO	Shi Tomasi	KLT	Local BA	-	DBoW2 PG	-	-	-	-	✓	-	Good	Exc.	[49]
ORB-SLAM3 (ours)	SLAM	ORB	Descriptor	Local BA	DBoW2 PG+BA	DBoW2 PG+BA	✓	✓	✓	✓	✓	✓	Exc.	Exc.	[5]

<sup>1</sup> Last source code provided by a different author. Original software is available at [50].

<sup>2</sup> Source code available only for the first version, SVO 2.0 is not open source.

<sup>3</sup> MSCKF is patented [51], only a re-implementation by a different author is available as open source.

## ORB-SLAM3和其他VIO算法量化效果对比

Table II: Performance comparison in the EuRoC dataset (RMSE ATE in m., scale error in %). Except where noted, we show results reported by the authors of each system, for all the frames in the trajectory, comparing with the processed GT.

		MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg <sup>4</sup>	
Monocular	ORB-SLAM [4]	ATE <sup>2,3</sup>	0.071	0.067	0.071	0.082	0.060	<b>0.015</b>	0.020	-	<b>0.021</b>	<b>0.018</b>	-	0.047*
	DSO [27]	ATE	0.046	0.046	0.072	3.810	0.110	0.089	0.107	0.903	0.044	0.132	1.152	0.601
	DSM [31]	ATE	0.039	0.036	0.055	<b>0.057</b>	0.067	0.095	0.059	0.076	0.056	0.057	<b>0.784</b>	<b>0.126</b>
	ORB-SLAM3 (ours)	ATE	<b>0.017</b>	<b>0.017</b>	<b>0.031</b>	0.066	<b>0.044</b>	0.033	<b>0.016</b>	<b>0.037</b>	<b>0.021</b>	0.022	-	0.030*
Stereo	ORB-SLAM2 [3]	ATE	0.035	<b>0.018</b>	0.028	0.119	0.060	<b>0.035</b>	<b>0.020</b>	<b>0.048</b>	0.037	0.035	-	0.044*
	VINS Fusion [44]	ATE	0.540	0.460	0.330	0.780	0.500	0.550	0.230	-	0.230	0.200	-	0.424*
	SVO [24]	ATE	0.040	0.070	0.270	0.170	0.120	0.040	0.040	0.070	0.080	0.090	0.790	0.159
	ORB-SLAM3 (ours)	ATE (m)	<b>0.025</b>	0.022	<b>0.027</b>	<b>0.089</b>	<b>0.058</b>	<b>0.035</b>	0.021	0.049	<b>0.032</b>	<b>0.027</b>	<b>0.361</b>	<b>0.068</b>
	VI-DSO [46]	ATE	0.075	0.084	0.087	0.217	0.082	<b>0.027</b>	0.028	-	<b>0.032</b>	0.041	0.074	0.075*
Monocular Inertial	ORB-SLAM [4]	ATE <sup>2,3</sup> scale error <sup>2,3</sup>	0.5	0.8	1.5	3.5	0.5	0.9	0.8	-	0.2	1.4	0.7	1.1*
	VINS [44]	ATE <sup>4</sup>	0.084	0.105	0.074	0.122	0.147	0.047	0.066	0.180	0.056	0.090	0.244	0.110
	VI-DSO [46]	ATE scale error	0.062	<b>0.044</b>	0.017	0.132	0.121	0.059	0.067	0.096	0.040	0.062	0.174	0.089
	ORB-SLAM3 (ours)	ATE scale error	<b>0.032</b>	0.053	<b>0.033</b>	<b>0.099</b>	<b>0.071</b>	0.043	<b>0.016</b>	<b>0.025</b>	0.041	<b>0.015</b>	<b>0.037</b>	<b>0.042</b>
Stereo Inertial	OKVIS [39]	ATE	0.160	0.220	0.240	0.340	0.470	0.090	0.200	0.240	0.30	0.160	0.290	0.230
	VINS Fusion [44]	ATE <sup>4</sup>	0.166	0.152	0.125	0.280	0.284	0.076	0.069	0.114	0.066	0.091	0.096	0.138
	BASALT [47]	ATE <sup>3</sup>	0.080	0.060	0.050	0.100	<b>0.080</b>	0.040	0.020	0.030	<b>0.030</b>	0.020	-	0.051*
	Kimera [8]	ATE	0.080	0.090	0.110	0.150	0.240	0.050	0.110	0.120	0.070	0.100	0.190	0.119
	ORB-SLAM3 (ours)	ATE scale error	<b>0.037</b>	<b>0.031</b>	<b>0.026</b>	<b>0.089</b>	0.086	<b>0.037</b>	<b>0.014</b>	<b>0.023</b>	0.037	<b>0.014</b>	<b>0.029</b>	<b>0.036</b>

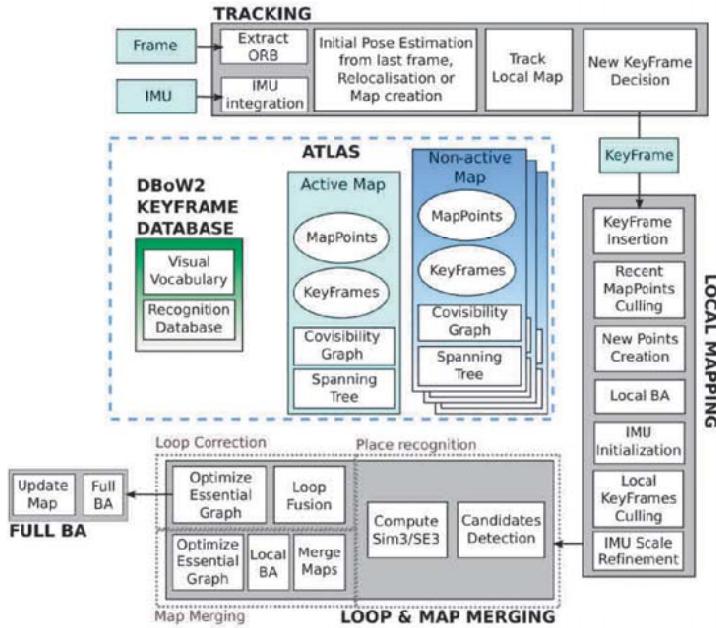
<sup>1</sup> Average error of the successful sequences. Systems that did no complete all sequences are denoted by \* and are not marked in bold.

<sup>2</sup> Errors reported with raw GT instead of processed GT.

<sup>3</sup> Errors reported with keyframe trajectory instead of full trajectory.

<sup>4</sup> Errors obtained by ourselves, running the code with its default configuration.

## 算法框架图



## 数据关联

- 短期数据关联。仅仅和最近几秒内获取的地图元素进行匹配。这是大多数视觉里程计使用的唯一数据关联类型，这种方法存在的问题是：一旦地图元素从视野中消失，就会被丢弃，即使回到原来的地方，也会造成持续的估计漂移。对应ORB-SLAM里的跟踪。
- 中期数据关联。匹配距离相机近并且累积漂移较小的地图元素。与短期观测相比，这些信息可以一并加入BA优化，当相机移动到已经建好图的区域时，可以达到零漂移。这是该系统与带闭环检测的视觉里程计相比，精度更高的关键所在。对应ORB-SLAM里的局部建图。
- 长期数据关联。使用位置识别技术将观测与之前访问过的区域中的元素匹配，不管是在闭环检测中的累积漂移，还是跟踪丢失、重定位的情况下都可以成功匹配。长期匹配允许使用位姿图优化来重置漂移和矫正闭环。这是保证中、大型闭环场景中SLAM局部较高精度的关键。对应ORB-SLAM里用词袋进行闭环和重定位。
- 多地图数据关联。可以使用之前已经建立的多块地图来实现地图中的匹配和BA优化。

ORB-SLAM3充分使用了短期、中期、长期的数据关联，可以达到在已经建图区域的零漂移。

## ORB-SLAM3和ORB-SLAM2对比

类目	ORB-SLAM2	ORB-SLAM3
传感器类型	单目、双目、RGB-D	单目、单目+IMU、双目、双目+IMU、RGB-D、RGB-D相机+IMU
相位模型	针孔模型。双目模式下假设双目完成了立体校正，匹配点位于水平的极线附近	抽象模型，支持针孔、鱼眼模型。双目模式下不依赖于立体校正，而是将双目看作两个相对位置不变的独立的、具有重叠视角的单目相机。
地图	单地图	多地图，支持地图融合
尺度	单目相机模式下地图和位姿没有绝对尺度	单目相机+IMU模式下地图和位姿具有绝对尺度
地图初始化	单目模式下地图初始化较慢，且没有尺度信息	单目+IMU模式下可以较快的初始化地图，且具有绝对尺度

跟踪线程	通过重定位来找回，如果重定位失败，则彻底跟丢。针孔相机模型下的EPnP算法	视觉惯性模式下，短期跟踪丢失时可以通过IMU预积分来推算位姿，也可通过重定位来找回位姿；如果长期跟踪丢失，则将目前地图保存，重新建立地图初始化；视觉惯性信息联合优化位姿。将相机模型和SLAM系统解耦，采用了最大似然PnP算法
局部建图线程	-	新增：视觉惯性模式下，初始化IMU，对IMU参数、重力方向、尺度信息进行优化、视觉惯性信息联合优化位姿
闭环线程	-	新增：一种新的高召回率的位置识别算法，多地图融合、视觉惯性信息联合优化位姿

## 源码注释地址

独家详细注释代码：[https://github.com/electech6/ORB\\_SLAM3\\_detailed\\_comments](https://github.com/electech6/ORB_SLAM3_detailed_comments)

代码高亮插件

## 变量命名规范

后面会有大量的源码详解，在介绍之前，我们有必要先了解一下在ORB-SLAM2中变量的命名规则，这对我们学习代码非常有用。

以小写字母m（member的首字母）开头的变量表示类的成员变量。比如：

```
1 int mSensor;
2 int mTrackingState;
3 std::mutex mMutexMode;
```

对于某些复杂的数据类型，第2个甚至第3个字母也有一定的意义，比如：

mp开头的变量表示指针（pointer）型类成员变量；

```
1 Tracking* mpTracker;
2 LocalMapping* mpLocalMapper;
3 LoopClosing* mpLoopCloser;
4 Viewer* mpViewer;
```

mb开头的变量表示布尔（bool）型类成员变量；

```
1 bool mbOnlyTracking;
```

mv开头的变量表示向量（vector）型类成员变量；

```
1 std::vector<int> mvIniLastMatches;
2 std::vector<cv::Point3f> mvIniP3D;
```

mpt开头的变量表示指针（pointer）型类成员变量，并且它是一个线程（thread）；

```
1 std::thread* mptLocalMapping;
2 std::thread* mptLoopClosing;
3 std::thread* mptViewer;
```

ml开头的变量表示列表（list）型类成员变量；

mlp开头的变量表示列表（list）型类成员变量，并且它的元素类型是指针（pointer）；

mlb开头的变量表示列表（list）型类成员变量，并且它的元素类型是布尔型（bool）；

```
1 list<double> mlFrameTimes;
2 list<bool> mlbLost;
3 list<cv::Mat> mlRelativeFramePoses;
4 list<KeyFrame*> mlpReferences;
```

ORB-SLAM3代码编译运行方法见课程视频讲解。

## 第2章：ORB-SLAM2重点知识回顾

具体内容见ORB-SLAM3第③期课程视频

## 第3章：IMU介绍及预积分推导

### 3.0 IMU简介及噪声模型

参考An introduction to inertial navigation

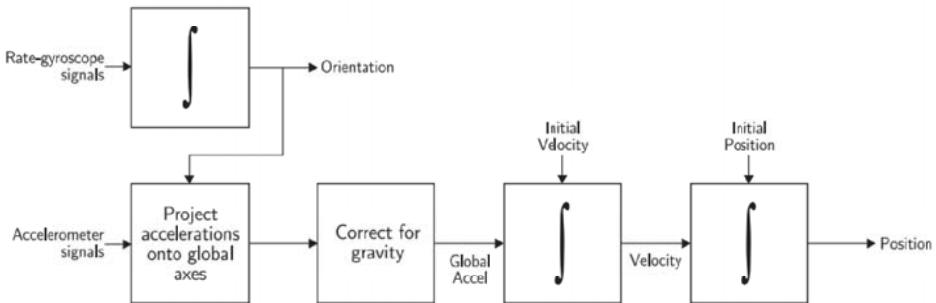
<https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf>

[http://www.acsu.buffalo.edu/~johnc/gpsins\\_gnc05.pdf](http://www.acsu.buffalo.edu/~johnc/gpsins_gnc05.pdf)

主要关注MEMS，陀螺仪+加速度计

陀螺仪输出角速度，加速度计输出加速度，绝大多数场景下认为他们是同一坐标系。

使用时认为他们是捷联的（说人话就是相对固定的，绑定在一块的）

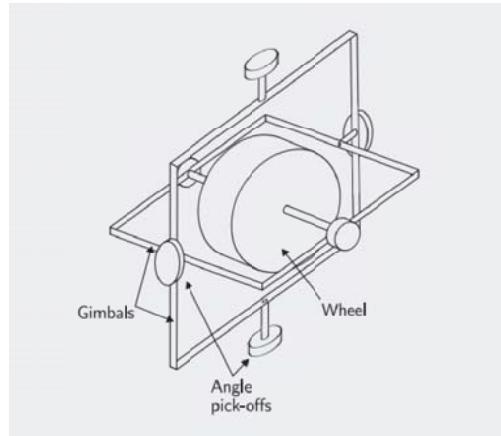


Strapdown inertial navigation algorithm.

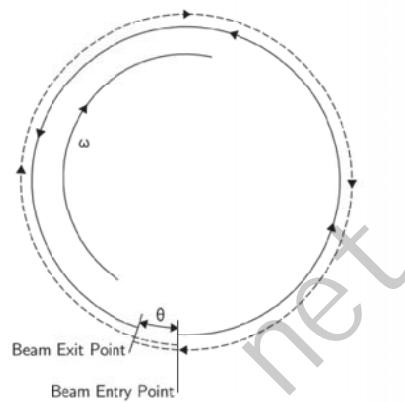
### 陀螺仪

#### 常见种类：

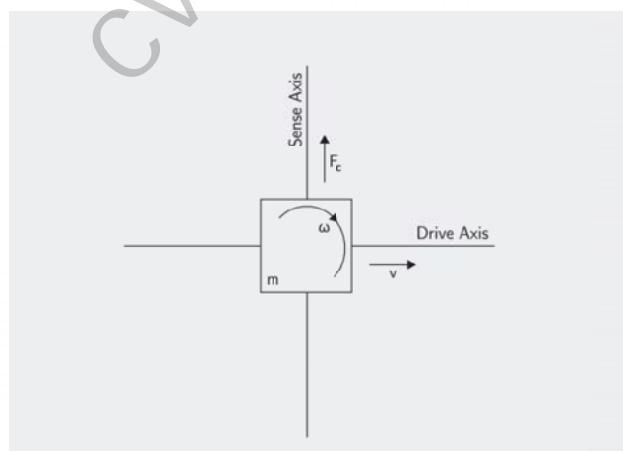
- 机械式陀螺仪，很粗糙，还要润滑



- 光学陀螺仪，与机械陀螺仪不同，光学院陀螺仪不包含运动部件，只需要几秒钟就能启动。光学院陀螺仪的精度在很大程度上取决于光传输路径的长度（越大越好），这受到器件尺寸的限制。



- MEMS陀螺仪，尽管经过多年的发展，机械和光学院陀螺仪仍然包含很多部件，并要求零件具有高精度的公差和复杂的装配技术。因此，它们仍然很昂贵。相比之下，使用硅微加工技术制造的MEMS传感器的零件计数较低（MEMS陀螺仪可以包含三个零件），而且制造成本相对便宜。主要利用科氏力。



## 误差模型

找了些资料，结合个人的理解总结。

imu的噪声模型可由下式表示（加速度计陀螺仪通用），白噪声过程在进行一次积分后就行成了随机游走过程

$$m(t) = m_t(t) + bias(t) + \epsilon(t)$$

$$bias(t) = n_b(t)$$

- 固定偏差bias，偏置，通常在系统里面作为状态量去估计，短时间内认为不变，长时间缓慢变化
- 白噪声（积分成角度随机游走）

设  $N_i$  为白噪声序列中的第  $i$  个随机变量，表示离散的噪声。每个  $N_i$  都有相同的分布，平均  $E(N_i) = E(N) = 0$  和有限方差  $Var(N_i) = Var(N) = \sigma^2$ 。当  $i = j$  时， $Cov(N_i, N_j) = 0$ 。

$$\int_0^t \epsilon(\tau) d\tau = \delta t \sum_{i=1}^n N_i$$

其中  $n$  为期间从设备接收的样本数， $\delta t$  为连续样本之间的时间。使用标准公式  $E(ax + bY) = aE(X) + bE(Y)$  和  $Var(ax+bx) = a^2 \cdot Var(X) + b^2 \cdot Var(Y) + 2abCov(X, Y)$ （其中  $a$  和  $b$  是常数， $X$  和  $Y$  是随机变量），可以得出

$$E\left(\int_0^t \epsilon(\tau) d\tau\right) = \delta t \cdot n \cdot E(N) = 0$$

$$Var\left(\int_0^t \epsilon(\tau) d\tau\right) = \delta t^2 \cdot n \cdot Var(N) = \delta t \cdot t \cdot \sigma^2$$

标准差为  $\sigma_\theta(t) = \sigma \cdot \sqrt{\delta t \cdot t}$ 。当取  $t=1$  时，为输出的标定结果，定义 angle random walk (ARW)。

$$ARW = \sigma_\theta(1)$$

所以单位为 rad/s/sqrt(hz)。这个量在 kalibr 中被称为陀螺仪的白噪声，不管叫啥吧，反正他是一个标定结果，实际用的时候要用到离散的方差，也就是  $\sigma$  的平方，所以误差的离散标准差相当于在标定结果基础上除以  $\sqrt{\delta t}$ 。

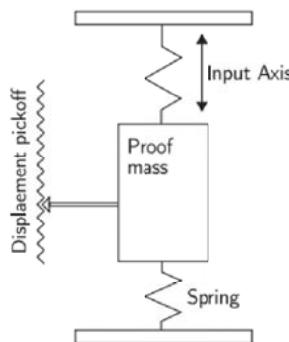
- Bias Stability (偏置随机游走)

关于这个误差来源有多种解释，这个不需要去关注。同样的，用上面的方法我们可以得出连续时间与离散时间的关系，也是要除以  $\sqrt{\delta t}$ ，只不过连续的单位为 rad/s^2/sqrt(hz)。但是这里要注意的是我们最后得到的这个标准差不管是离散的还是连续的，都是偏置的导数！！！实际算的时候我们要的是偏置随机游走，在离散情况下还要乘以一个  $\delta t$ ，所以最终看起来从标定结果到离散的偏置随机游走相当于乘以  $\sqrt{\delta t}$ 。

## 加速度计

### 常见种类：

- 机械式



- 固态

固态加速度计可以分为多种，包括表面声波、振动、硅和石英器件。固态加速度计体积小、可靠、坚固耐用。

- MEMS 加速度计

MEMS 加速度计主要有两类。第一类由使用 MEMS 技术制造的机械加速度计（即：测量支撑质量位移的设备）组成。第二类设备包括测量由张力变化引起的振动元件频率变化的设备，如波表面波加速度计。

## 误差模型

与陀螺仪计算方式一样，就是单位不太一样

### IMU内参

Parameter	YAML element	Symbol	Units
Gyroscope "white noise"	gyroscope_noise_density	$\sigma_g$	$\frac{rad}{s} \frac{1}{\sqrt{Hz}}$
Accelerometer "white noise"	accelerometer_noise_density	$\sigma_a$	$\frac{m}{s^2} \frac{1}{\sqrt{Hz}}$
Gyroscope "random walk"	gyroscope_random_walk	$\sigma_{bg}$	$\frac{rad}{s^2} \frac{1}{\sqrt{Hz}}$
Accelerometer "random walk"	accelerometer_random_walk	$\sigma_{ba}$	$\frac{m}{s^3} \frac{1}{\sqrt{Hz}}$
IMU sampling rate	update_rate	$\frac{1}{\Delta t}$	Hz

### 3.1 推导前的公式

$$\vec{\omega}^\wedge = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \quad (1.1)$$

$$\vec{a}^\wedge \cdot \vec{b} = -\vec{b}^\wedge \cdot \vec{a} \quad (1.2)$$

当  $\vec{\phi}$  是小量时

$$\text{Exp}(\vec{\phi}) = \exp(\vec{\phi}^\wedge) \approx I + \vec{\phi}^\wedge \quad (1.3)$$

当  $\delta\vec{\phi}$  是小量时

$$\text{Exp}(\vec{\phi} + \delta\vec{\phi}) \approx \text{Exp}(\vec{\phi}) \cdot \text{Exp}\left(J_r(\vec{\phi}) \cdot \delta\vec{\phi}\right) \quad (1.4)$$

$$\text{Exp}(\vec{\phi}) \cdot \text{Exp}(\delta\vec{\phi}) = \text{Exp}\left(\vec{\phi} + J_r^{-1}(\vec{\phi}) \cdot \delta\vec{\phi}\right) \quad (1.5)$$

其中：

$$J_r(\vec{\phi}) = \mathbf{I} - \frac{1 - \cos(\|\vec{\phi}\|)}{\|\vec{\phi}\|^2} \vec{\phi}^\wedge + \left( \frac{\|\vec{\phi}\| - \sin(\|\vec{\phi}\|)}{\|\vec{\phi}\|^3} \right) (\vec{\phi}^\wedge)^2 \quad (1.6)$$

$$J_r^{-1}(\vec{\phi}) = \mathbf{I} + \frac{1}{2} \vec{\phi}^\wedge + \left( \frac{1}{\|\vec{\phi}\|^2} - \frac{1 + \cos(\|\vec{\phi}\|)}{2 \cdot \|\vec{\phi}\| \cdot \sin(\|\vec{\phi}\|)} \right) (\vec{\phi}^\wedge)^2 \quad (1.7)$$

当  $\vec{\phi}$  为小量时

$$\mathbf{J}_r(\vec{\phi}) \approx \mathbf{I} \quad (1.8)$$

$$\mathbf{J}_r^{-1}(\vec{\phi}) \approx \mathbf{I} \quad (1.9)$$

$$\mathbf{R} \cdot \text{Exp}(\vec{\phi}) \cdot \mathbf{R}^T = \exp\left(\mathbf{R}\vec{\phi}^\wedge \mathbf{R}^T\right) = \text{Exp}(\mathbf{R}\vec{\phi}) \quad (1.10)$$

$$\text{Exp}(\vec{\phi}) \cdot \mathbf{R} = \mathbf{R} \cdot \text{Exp}\left(\mathbf{R}^T \vec{\phi}\right) \quad (1.11)$$

## 3.2 预积分

$$\begin{aligned} \mathbf{R}_{wj} &= \mathbf{R}_{wi} \cdot \prod_{k=i}^{j-1} \text{Exp}\left(\left(\tilde{\omega}_k - \mathbf{b}_k^g - \eta_k^{gd}\right) \cdot \Delta t\right) \\ \mathbf{v}_j &= \mathbf{v}_i + \mathbf{g} \cdot \Delta t_{ij} + \sum_{k=i}^{j-1} \mathbf{R}_{wk} \cdot \left(\tilde{\mathbf{f}}_k - \mathbf{b}_k^a - \eta_k^{ad}\right) \cdot \Delta t \\ \mathbf{p}_{wj} &= \mathbf{p}_{wi} + \sum_{k=i}^{j-1} \mathbf{v}_k \cdot \Delta t + \frac{j-i}{2} \mathbf{g} \cdot \Delta t^2 + \frac{1}{2} \sum_{k=i}^{j-1} \mathbf{R}_{wk} \cdot \left(\tilde{\mathbf{f}}_k - \mathbf{b}_k^a - \eta_k^{ad}\right) \cdot \Delta t^2 \\ &= \mathbf{p}_{wi} + \sum_{k=i}^{j-1} \left[ \mathbf{v}_k \cdot \Delta t + \frac{1}{2} \mathbf{g} \cdot \Delta t^2 + \frac{1}{2} \mathbf{R}_{wk} \cdot \left(\tilde{\mathbf{f}}_k - \mathbf{b}_k^a - \eta_k^{ad}\right) \cdot \Delta t^2 \right] \end{aligned}$$

其中：

$$\Delta t_{ij} = \sum_{k=i}^{j-1} \Delta t = (j-i)\Delta t$$

由积分引出预积分，预积分里面的每一项与起始状态无关，可以认为都是相对量，这个好处在于计算预积分时不需要考虑起始状态，值得注意的是关于速度与位置的预积分里面都包含了重力。预积分计算方式：

- 1、消除第 i 时刻对积分的影响
- 2、保留重力的影响

$$\begin{aligned} \Delta \mathbf{R}_{ij} &\triangleq \mathbf{R}_{wi}^T \mathbf{R}_{wj} = \prod_{k=i}^{j-1} \text{Exp}\left(\left(\tilde{\omega}_k - \mathbf{b}_k^g - \eta_k^{gd}\right) \cdot \Delta t\right) \\ \Delta \mathbf{v}_{ij} &\triangleq \mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) \\ &= \sum_{k=i}^{j-1} \Delta \mathbf{R}_{ik} \cdot \left(\tilde{\mathbf{f}}_k - \mathbf{b}_k^a - \eta_k^{ad}\right) \cdot \Delta t \\ \Delta \mathbf{p}_{ij} &\triangleq \mathbf{R}_{wi}^T \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) \\ &= \sum_{k=i}^{j-1} \left[ \Delta \mathbf{v}_{ik} \cdot \Delta t + \frac{1}{2} \Delta \mathbf{R}_{ik} \cdot \left(\tilde{\mathbf{f}}_k - \mathbf{b}_k^a - \eta_k^{ad}\right) \cdot \Delta t^2 \right] \end{aligned}$$

关于位置的预积分推到时要注意对于 g 的处理，其中要利用到等差数列求和的公式，公式如下：

$$\frac{j-i}{2} - \frac{(j-i)^2}{2} = -\frac{(j-i)[j-(i+1)]}{2} = -\sum_{k=i}^{j-1} (k-i)$$

## 3.3 噪声分离

目的：上面推预积分时对imu的读数会减去它的偏置与误差，其中偏置可以作为状态量去得出，但是误差是没有办法得出的，我们能做的就是拿到imu数据减去偏置后直接使用，通常的办法就是通过计算误差的方式过滤掉这部分误差，无论是优化还是滤波都跳不过一个重要的矩阵——预积分的信息矩阵（协方差矩阵的逆）由于假设了噪声是高斯白噪声，所以噪声的方差对状态方差的影响可以通过高斯分布推理过来。本节我们的目的就是推导出标定好的imu噪声对预积分的影响，也就是预积分的偏差关于噪声的式子，下一节推出协方差方差的关系。

由于假设了噪声为高斯白噪声，也就是服从了高斯分布，因此预积分噪声同样为高斯分布，整个过程以推导出预积分噪声的表达式为主，令预积分的测量噪声为：

$$\boldsymbol{\eta}_{ij}^{\Delta} \triangleq \begin{bmatrix} \delta\vec{\phi}_{ij}^T & \delta\mathbf{v}_{ij}^T & \delta\mathbf{p}_{ij}^T \end{bmatrix}^T$$

读作“伊塔”。

下面分别对3个向量噪声进行推导，推导方式：分离噪声成如下形式，可以理解成：真实值 = 测量值 - 误差

$$\Delta\mathbf{R}_{ij} \triangleq \Delta\tilde{\mathbf{R}}_{ij} \cdot \text{Exp}(-\delta\vec{\phi}_{ij}) \quad (3.1)$$

$$\Delta\mathbf{v}_{ij} \triangleq \Delta\tilde{\mathbf{v}}_{ij} - \delta\mathbf{v}_{ij} \quad (3.2)$$

$$\Delta\mathbf{p}_{ij} \triangleq \Delta\tilde{\mathbf{p}}_{ij} - \delta\mathbf{p}_{ij} \quad (3.3)$$

### 3.3.1 对于 $\Delta R_{ij}$ 项

$$\begin{aligned} \Delta\mathbf{R}_{ij} &= \prod_{k=i}^{j-1} \text{Exp}\left((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t - \eta_k^{gd} \Delta t\right) \\ &\stackrel{(1)}{\approx} \prod_{k=i}^{j-1} \left\{ \text{Exp}((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t) \cdot \text{Exp}\left(-\mathbf{J}_r ((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t) \cdot \eta_k^{gd} \Delta t\right) \right\} \\ &\stackrel{(2)}{=} \Delta\tilde{\mathbf{R}}_{ij} \cdot \prod_{k=i}^{j-1} \text{Exp}\left(-\Delta\tilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \eta_k^{gd} \Delta t\right) \end{aligned}$$

注意式中假设了这段时间内偏置不变，就是一个数。对于(1)处比较好理解，利用公式(1.4)：

$$\text{Exp}(\vec{\phi} + \delta\vec{\phi}) \approx \text{Exp}(\vec{\phi}) \cdot \text{Exp}\left(J_r(\vec{\phi}) \cdot \delta\vec{\phi}\right)$$

对于(2)处比较难理解，而且要用到公式(1.9)

$$\text{Exp}(\vec{\phi}) \cdot \mathbf{R} = \mathbf{R} \cdot \text{Exp}\left(\mathbf{R}^T \vec{\phi}\right)$$

我们先把由(1)得出的结果展开，令

$$\begin{aligned}
& \mathbf{J}_r^k = \mathbf{J}_r ((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t) \\
& \stackrel{(1)}{\approx} \exp((\tilde{\boldsymbol{\omega}}_i - \mathbf{b}_i^g) \Delta t) \cdot \exp(-\mathbf{J}_r^i \cdot \boldsymbol{\eta}_i^{gd} \Delta t) \cdot \exp((\tilde{\boldsymbol{\omega}}_{i+1} - \mathbf{b}_i^g) \Delta t) \\
& \quad \cdot \exp(-\mathbf{J}_r^{i+1} \cdot \boldsymbol{\eta}_{i+1}^{gd} \Delta t) \cdot \exp((\tilde{\boldsymbol{\omega}}_{i+2} - \mathbf{b}_i^g) \Delta t) \\
& \quad \cdot \exp(-\mathbf{J}_r^{i+2} \cdot \boldsymbol{\eta}_{i+2}^{gd} \Delta t) \cdots \exp((\tilde{\boldsymbol{\omega}}_{j-1} - \mathbf{b}_i^g) \Delta t) \\
& \quad \cdot \exp(-\mathbf{J}_r^{j-1} \cdot \boldsymbol{\eta}_{j-1}^{gd} \Delta t) \\
& = \exp((\tilde{\boldsymbol{\omega}}_i - \mathbf{b}_i^g) \Delta t) \cdot \exp((\tilde{\boldsymbol{\omega}}_{i+1} - \mathbf{b}_i^g) \Delta t) \cdots \exp\left(-(\exp((\tilde{\boldsymbol{\omega}}_{i+1} - \mathbf{b}_i^g) \Delta t))^T \mathbf{J}_r^i \cdot \boldsymbol{\eta}_i^{gd} \Delta t\right. \\
& \quad \cdot \exp(-\mathbf{J}_r^{i+1} \cdot \boldsymbol{\eta}_{i+1}^{gd} \Delta t) \cdot \exp((\tilde{\boldsymbol{\omega}}_{i+2} - \mathbf{b}_i^g) \Delta t) \\
& \quad \cdot \exp(-\mathbf{J}_r^{i+2} \cdot \boldsymbol{\eta}_{i+2}^{gd} \Delta t) \cdots \exp((\tilde{\boldsymbol{\omega}}_{j-1} - \mathbf{b}_i^g) \Delta t) \\
& \quad \cdot \exp(-\mathbf{J}_r^{j-1} \cdot \boldsymbol{\eta}_{j-1}^{gd} \Delta t) \\
& = \exp((\tilde{\boldsymbol{\omega}}_i - \mathbf{b}_i^g) \Delta t) \\
& \quad \cdot \exp((\tilde{\boldsymbol{\omega}}_{i+1} - \mathbf{b}_i^g) \Delta t) \cdot \exp((\tilde{\boldsymbol{\omega}}_{i+2} - \mathbf{b}_i^g) \Delta t) \\
& \quad \cdot \exp\left(-\left(\exp((\tilde{\boldsymbol{\omega}}_{i+2} - \mathbf{b}_i^g) \Delta t)^T \cdot \exp((\tilde{\boldsymbol{\omega}}_{i+1} - \mathbf{b}_i^g) \Delta t)^T\right) \mathbf{J}_r^i \cdot \boldsymbol{\eta}_i^{gd} \Delta t\right) \\
& \quad \cdot \exp\left(-\exp((\tilde{\boldsymbol{\omega}}_{i+2} - \mathbf{b}_i^g) \Delta t)^T \mathbf{J}_r^{i+1} \cdot \boldsymbol{\eta}_{i+1}^{gd} \Delta t\right) \exp\left(-\mathbf{J}_r^{i+2} \cdot \boldsymbol{\eta}_{i+2}^{gd} \Delta t\right) \cdots \\
& \quad \cdot \exp((\tilde{\boldsymbol{\omega}}_{j-1} - \mathbf{b}_i^g) \Delta t) \exp\left(-\mathbf{J}_r^{j-1} \cdot \boldsymbol{\eta}_{j-1}^{gd} \Delta t\right) \\
& = \prod_{k=i}^{j-1} \exp((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t) \prod_{k=i}^{j-1} \exp\left(-\prod_{m=j-1}^{k+1} \exp((\tilde{\boldsymbol{\omega}}_m - \mathbf{b}_i^g) \Delta t)^T \cdot \mathbf{J}_r^k \cdot \boldsymbol{\eta}_k^{gd} \Delta t\right) \\
& \stackrel{(2)}{=} \Delta \tilde{\mathbf{R}}_{ij} \cdot \prod_{k=i}^{j-1} \exp\left(-\Delta \tilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \boldsymbol{\eta}_k^{gd} \Delta t\right)
\end{aligned}$$

由上面可得：

$$\begin{aligned}
\exp(-\delta \vec{\phi}_{ij}) &= \prod_{k=i}^{j-1} \exp\left(-\Delta \tilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \boldsymbol{\eta}_k^{gd} \Delta t\right) \\
\delta \vec{\phi}_{ij} &= -\log\left(\prod_{k=i}^{j-1} \exp\left(-\Delta \tilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \boldsymbol{\eta}_k^{gd} \Delta t\right)\right)
\end{aligned}$$

由于结果结构比较复杂，所以还需要接着化简。

$$\text{令: } \xi_k = \Delta \tilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \boldsymbol{\eta}_k^{gd} \Delta t$$

读作“克西”或“克赛”，利用公式(1.5)：

$$\begin{aligned}
\exp(\vec{\phi}) \cdot \exp(\delta \vec{\phi}) &= \exp\left(\vec{\phi} + J_r^{-1}(\vec{\phi}) \cdot \delta \vec{\phi}\right) \\
\log(\exp(\vec{\phi}) \cdot \exp(\delta \vec{\phi})) &= \vec{\phi} + J_r^{-1}(\vec{\phi}) \cdot \delta \vec{\phi}
\end{aligned}$$

以及公式： $J_r^{-1}(\vec{\phi}) \approx \mathbf{I}$ 有：

$$\begin{aligned}
\delta \vec{\phi}_{ij} &= -\log \left( \prod_{k=i}^{j-1} \text{Exp}(-\xi_k) \right) \\
&= -\log \left( \text{Exp}(-\xi_i) \prod_{k=i+1}^{j-1} \text{Exp}(-\xi_k) \right) \\
&\approx - \left( -\xi_i + \mathbf{I} \cdot \log \left( \prod_{k=i+1}^{j-1} \text{Exp}(-\xi_k) \right) \right) \\
&= \xi_i - \log \left( \prod_{k=i+1}^{j-1} \text{Exp}(-\xi_k) \right) \\
&= \xi_i - \log \left( \text{Exp}(-\xi_{i+1}) \prod_{k=i+2}^{j-1} \text{Exp}(-\xi_k) \right) \\
&\approx \xi_i + \xi_{i+1} - \log \left( \prod_{k=i+2}^{j-1} \text{Exp}(-\xi_k) \right) \\
&\approx \dots \\
&\approx \sum_{k=i}^{j-1} \xi_k
\end{aligned}$$

最后推出：

$$\delta \vec{\phi}_{ij} = \sum_{k=i}^{j-1} \Delta \tilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \eta_k^{ad} \Delta t$$

由式可知  $\delta \vec{\phi}_{ij}$  服从零均值的高斯分布。

对于  $\Delta \mathbf{v}_{ij}$  项

首先要利用前面关于角度的式子(3.1)带入到  $\Delta \mathbf{v}_{ij}$ ，即：

$$\begin{aligned}
\Delta \mathbf{v}_{ij} &= \sum_{k=i}^{j-1} \Delta \mathbf{R}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^{ad} \right) \cdot \Delta t \\
&\approx \sum_{k=i}^{j-1} \Delta \tilde{\mathbf{R}}_{ik} \cdot \text{Exp} \left( -\delta \vec{\phi}_{ik} \right) \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^{ad} \right) \cdot \Delta t \\
&\stackrel{(1)}{\approx} \sum_{k=i}^{j-1} \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \mathbf{I} - \delta \vec{\phi}_{ik}^\wedge \right) \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^{ad} \right) \cdot \Delta t \\
&\stackrel{(2)}{\approx} \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \mathbf{I} - \delta \vec{\phi}_{ik}^\wedge \right) \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right) \cdot \Delta t - \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t \right] \\
&\stackrel{(3)}{=} \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right) \cdot \Delta t + \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right)^\wedge \cdot \delta \vec{\phi}_{ik} \cdot \Delta t - \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t \right] \\
&= \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right) \cdot \Delta t \right] + \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right)^\wedge \cdot \delta \vec{\phi}_{ik} \cdot \Delta t - \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t \right]
\end{aligned}$$

(1)利用了公式(1.3)当  $\vec{\phi}$  是小量时：  $\text{Exp}(\vec{\phi}) = \exp(\vec{\phi}^\wedge) \approx I + \vec{\phi}^\wedge$

(2)忽略了小量；

(3)利用了公式(1.2)  $\mathbf{a}^\wedge \cdot \mathbf{b} = -\mathbf{b}^\wedge \cdot \mathbf{a}$

上式令：

$$\Delta \tilde{\mathbf{v}}_{ij} \triangleq \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a) \cdot \Delta t \right]$$

$$\delta \mathbf{v}_{ij} \triangleq \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \eta_k^{ad} \Delta t - \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a)^\wedge \cdot \delta \vec{\phi}_{ik} \cdot \Delta t \right]$$

即可得出式(3.2)，且 $\delta \mathbf{v}_{ij}$ 拥有高斯分布的形式

$$\Delta \mathbf{v}_{ij} \triangleq \Delta \tilde{\mathbf{v}}_{ij} - \delta \mathbf{v}_{ij}$$

### 3.3.3 对于 $\Delta \mathbf{p}_{ij}$ 项

首先要利用前面关于角度的式子(3.1)(3.2)带入到 $\Delta \mathbf{p}_{ij}$ ，即：

$$\begin{aligned} & \Delta \mathbf{p}_{ij} \\ &= \sum_{k=i}^{j-1} \left[ \Delta \mathbf{v}_{ik} \cdot \Delta t + \frac{1}{2} \Delta \mathbf{R}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^{ad}) \cdot \Delta t^2 \right] \\ &\approx \sum_{k=i}^{j-1} \left[ (\Delta \tilde{\mathbf{v}}_{ik} - \delta \mathbf{v}_{ik}) \cdot \Delta t + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot \text{Exp}(-\delta \vec{\phi}_{ik}) \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^{ad}) \cdot \Delta t^2 \right] \\ &\stackrel{(1)}{\approx} \sum_{k=i}^{j-1} \left[ (\Delta \tilde{\mathbf{v}}_{ik} - \delta \mathbf{v}_{ik}) \cdot \Delta t + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot (\mathbf{I} - \delta \vec{\phi}_{ik}^\wedge) \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a - \boldsymbol{\eta}_k^{ad}) \cdot \Delta t^2 \right] \\ &\stackrel{(2)}{\approx} \sum_{k=i}^{j-1} \left[ (\Delta \tilde{\mathbf{v}}_{ik} - \delta \mathbf{v}_{ik}) \cdot \Delta t + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot (\mathbf{I} - \delta \vec{\phi}_{ik}^\wedge) \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a) \cdot \Delta t^2 - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t^2 \right] \\ &\stackrel{(3)}{=} \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{v}}_{ik} \Delta t + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a) \Delta t^2 + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a)^\wedge \delta \vec{\phi}_{ik} \Delta t^2 \right. \\ &\quad \left. - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t^2 - \delta \mathbf{v}_{ik} \Delta t \right] \end{aligned}$$

(1)利用了公式(1.3)当 $\vec{\phi}$ 是小量时： $\text{Exp}(\vec{\phi}) = \exp(\vec{\phi}^\wedge) \approx \mathbf{I} + \vec{\phi}^\wedge$

(2)忽略了小量；

(3)利用了公式(1.2)  $\mathbf{a}^\wedge \cdot \mathbf{b} = -\mathbf{b}^\wedge \cdot \mathbf{a}$

上式令：

$$\begin{aligned} \Delta \tilde{\mathbf{p}}_{ij} &\triangleq \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{v}}_{ik} \Delta t + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a) \Delta t^2 \right] \\ \delta \mathbf{p}_{ij} &\triangleq \sum_{k=i}^{j-1} \left[ \delta \mathbf{v}_{ik} \Delta t - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a)^\wedge \delta \vec{\phi}_{ik} \Delta t^2 + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t^2 \right] \end{aligned}$$

即可得出式(3.3)，且 $\delta \mathbf{p}_{ij}$ 拥有高斯分布的形式： $\Delta \mathbf{p}_{ij} \triangleq \Delta \tilde{\mathbf{p}}_{ij} - \delta \mathbf{p}_{ij}$

## 3.4 噪声递推

上面求出了三个预积分误差的表达式，但由于式子要么是求和，要么是多积导致每次新来一个数据都需要从头计算，这给计算平台带来资源的浪费，因此这章我们要推出误差的递推形式，从上一时刻推出当前时刻，例如通过 $\delta \mathbf{p}_{ij-1}$ 推出 $\delta \mathbf{p}_{ij}$ 。

$$\delta \vec{\phi}_{ij} \triangleq \sum_{k=i}^{j-1} \Delta \tilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \boldsymbol{\eta}_k^{gd} \Delta t \quad (4.1)$$

$$\delta \mathbf{v}_{ij} \triangleq \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t - \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a)^\wedge \cdot \delta \vec{\phi}_{ik} \cdot \Delta t \right] \quad (4.2)$$

$$\delta \mathbf{p}_{ij} \triangleq \sum_{k=i}^{j-1} \left[ \delta \mathbf{v}_{ik} \Delta t - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a)^\wedge \delta \vec{\phi}_{ik} \Delta t^2 + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t^2 \right] \quad (4.3)$$

### 3.4.1 对于 $\delta \vec{\phi}_{ij}$ 项

$$\begin{aligned} \delta \vec{\phi}_{ij} &= \sum_{k=i}^{j-1} \Delta \tilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \boldsymbol{\eta}_k^{gd} \Delta t \\ &= \sum_{k=i}^{j-2} \Delta \tilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \boldsymbol{\eta}_k^{gd} \Delta t + \Delta \tilde{\mathbf{R}}_{jj}^T \mathbf{J}_r^{j-1} \boldsymbol{\eta}_{j-1}^{gd} \Delta t \\ &= \sum_{k=i}^{j-2} \left( \Delta \tilde{\mathbf{R}}_{k+1j-1} \Delta \tilde{\mathbf{R}}_{j-1j} \right)^T \mathbf{J}_r^k \boldsymbol{\eta}_k^{gd} \Delta t + \mathbf{J}_r^{j-1} \boldsymbol{\eta}_{j-1}^{gd} \Delta t \\ &= \Delta \tilde{\mathbf{R}}_{jj-1} \sum_{k=i}^{j-2} \Delta \tilde{\mathbf{R}}_{k+1j-1}^T \mathbf{J}_r^k \boldsymbol{\eta}_k^{gd} \Delta t + \mathbf{J}_r^{j-1} \boldsymbol{\eta}_{j-1}^{gd} \Delta t \\ &= \Delta \tilde{\mathbf{R}}_{jj-1} \delta \vec{\phi}_{ij-1} + \mathbf{J}_r^{j-1} \boldsymbol{\eta}_{j-1}^{gd} \Delta t \end{aligned}$$

### 3.4.2 对于 $\delta v_{ij}$ 项

$$\begin{aligned} \delta \mathbf{V}_{ij} &= \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t - \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a)^\wedge \cdot \delta \vec{\phi}_{ik} \cdot \Delta t \right] \\ &= \sum_{k=i}^{j-2} \left[ \Delta \tilde{\mathbf{R}}_{ik} \boldsymbol{\eta}_k^{ad} \Delta t - \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a)^\wedge \cdot \delta \vec{\phi}_{ik} \cdot \Delta t \right] + \\ &\quad \Delta \tilde{\mathbf{R}}_{ij-1} \boldsymbol{\eta}_{j-1}^{ad} \Delta t - \Delta \tilde{\mathbf{R}}_{ij-1} \cdot (\tilde{\mathbf{f}}_{j-1} - \mathbf{b}_i^a)^\wedge \cdot \delta \vec{\phi}_{ij-1} \cdot \Delta t \\ &= \delta \mathbf{v}_{ij-1} + \Delta \tilde{\mathbf{R}}_{ij-1} \boldsymbol{\eta}_{j-1}^{ad} \Delta t - \Delta \tilde{\mathbf{R}}_{ij-1} \cdot (\tilde{\mathbf{f}}_{j-1} - \mathbf{b}_i^a)^\wedge \cdot \delta \vec{\phi}_{ij-1} \Delta t \end{aligned}$$

### 3.4.3 对于 $\delta p_{ij}$ 项

$$\begin{aligned} \delta \mathbf{p}_{ij} &= \sum_{k=i}^{j-1} \left[ \delta \mathbf{v}_{ik} \Delta t - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a)^\wedge \delta \vec{\phi}_{ik} \Delta t^2 + \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \mathbf{y}_k^{ad} \Delta t^2 \right] \\ &= \delta \mathbf{p}_{ij-1} + \delta \mathbf{v}_{ij-1} \Delta t - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ij-1} \cdot (\tilde{\mathbf{f}}_{j-1} - \mathbf{b}_i^a)^\wedge \delta \vec{\phi}_{ij-1} \Delta t^2 + \\ &\quad \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ij-1} \boldsymbol{\eta}_{j-1}^{ad} \Delta t^2 \end{aligned}$$

总结。综上可以写出  $\boldsymbol{\eta}_{ij}^\Delta \triangleq \begin{bmatrix} \delta \vec{\phi}_{ij}^T & \delta \mathbf{v}_{ij}^T & \delta \mathbf{p}_{ij}^T \end{bmatrix}^T$

的递推矩阵。令  $\boldsymbol{\eta}_k^d = \left[ (\boldsymbol{\eta}_k^{gd})^T \ (\boldsymbol{\eta}_k^{ad})^T \right]^T$  有

$$\boldsymbol{\eta}_{ij}^{\Delta} = \begin{bmatrix} \Delta \widetilde{\mathbf{R}}_{jj-1} & \mathbf{0} & \mathbf{0} \\ -\Delta \widetilde{\mathbf{R}}_{ij-1} \cdot (\tilde{\mathbf{f}}_{j-1} - \mathbf{b}_i^a) \wedge \Delta t & \mathbf{I} & \mathbf{0} \\ -\frac{1}{2} \Delta \widetilde{\mathbf{R}}_{ij-1} \cdot (\tilde{\mathbf{f}}_{j-1} - \mathbf{b}_i^a) \wedge \Delta t^2 & \Delta t \mathbf{I} & \mathbf{I} \\ \mathbf{J}_r^{j-1} \Delta t & \mathbf{0} \\ \mathbf{0} & \Delta \widetilde{\mathbf{R}}_{ij-1} \Delta t \\ \mathbf{0} & \frac{1}{2} \Delta \widetilde{\mathbf{R}}_{ij-1} \Delta t^2 \end{bmatrix} \boldsymbol{\eta}_{ij-1}^{\Delta} +$$

有:  $\boldsymbol{\eta}_{ij}^{\Delta} = \mathbf{A}_{j-1} \boldsymbol{\eta}_{ij-1}^{\Delta} + \mathbf{B}_{j-1} \boldsymbol{\eta}_{j-1}^d$

重点来了!  $\boldsymbol{\Sigma}_{ij} = \mathbf{A}_{j-1} \boldsymbol{\Sigma}_{ij-1} \mathbf{A}_{j-1}^T + \mathbf{B}_{j-1} \boldsymbol{\Sigma}_{\eta} \mathbf{B}_{j-1}^T$

到此为止优化时使用的信息矩阵有了!

### 3.5 Bias更新时对预积分的影响

首先说明前面去除噪声时假设了这段时间内偏置不变,但偏置在vio算法中会作为状态量来优化,所以当通过优化后偏置会更新,这样一来如果重新计算这段时间的预积分会很浪费时间,所以本章目的是为了推出当偏置变化时直接求得新的预积分结果。

$$\Delta \widetilde{\mathbf{R}}_{ij} \triangleq \prod_{k=i}^{j-1} \text{Exp}((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t) \quad (5.1)$$

$$\Delta \tilde{\mathbf{v}}_{ij} \triangleq \sum_{k=i}^{j-1} [\Delta \widetilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a) \cdot \Delta t] \quad (5.2)$$

$$\Delta \tilde{\mathbf{p}}_{ij} \triangleq \sum_{k=i}^{j-1} [\Delta \tilde{\mathbf{v}}_{ik} \Delta t + \frac{1}{2} \Delta \widetilde{\mathbf{R}}_{ik} \cdot (\tilde{\mathbf{f}}_k - \mathbf{b}_i^a) \Delta t^2] \quad (5.3)$$

当有偏置更新时

$$\Delta \overline{\widetilde{\mathbf{R}}}_{ij} \triangleq \prod_{k=i}^{j-1} \text{Exp}((\tilde{\boldsymbol{\omega}}_k - (\mathbf{b}_i^g + \delta \mathbf{b}_i^g)) \Delta t) \quad (5.4)$$

$$\Delta \overline{\tilde{\mathbf{v}}}_{ij} \triangleq \sum_{k=i}^{j-1} [\Delta \overline{\widetilde{\mathbf{R}}}_{ik} \cdot (\tilde{\mathbf{f}}_k - (\mathbf{b}_i^a + \delta \mathbf{b}_i^a)) \cdot \Delta t] \quad (5.5)$$

$$\Delta \overline{\tilde{\mathbf{p}}}_{ij} \triangleq \sum_{k=i}^{j-1} [\Delta \overline{\tilde{\mathbf{v}}}_{ik} \Delta t + \frac{1}{2} \Delta \overline{\widetilde{\mathbf{R}}}_{ik} \cdot (\tilde{\mathbf{f}}_k - (\mathbf{b}_i^a + \delta \mathbf{b}_i^a)) \Delta t^2] \quad (5.6)$$

#### 3.5.1 对于 $\Delta \overline{\widetilde{\mathbf{R}}}_{ij}$ 项

令  $\mathbf{J}_r^k = \mathbf{J}_r ((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t)$  有:

$$\begin{aligned}
\Delta \overline{\widetilde{\mathbf{R}}}_{ij} &\triangleq \prod_{k=i}^{j-1} \text{Exp} ((\widetilde{\boldsymbol{\omega}}_k - (\mathbf{b}_i^g + \delta \mathbf{b}_i^g)) \Delta t) \\
&= \prod_{k=i}^{j-1} \text{Exp} ((\widetilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t - \delta \mathbf{b}_i^g \Delta t) \\
&\approx \prod_{k=i}^{j-1} (\text{Exp} ((\widetilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t) \cdot \text{Exp} (-\mathbf{J}_r^k \delta \mathbf{b}_i^g \Delta t)) \\
&= \Delta \widetilde{\mathbf{R}}_{ij} \cdot \prod_{k=i}^{j-1} \text{Exp} \left( -\Delta \widetilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \delta \mathbf{b}_i^g \Delta t \right) \\
&= \Delta \widetilde{\mathbf{R}}_{ij} \cdot \text{Exp} \left( \sum_{k=i}^{j-1} \left( -\Delta \widetilde{\mathbf{R}}_{k+1j}^T \cdot \mathbf{J}_r^k \cdot \delta \mathbf{b}_i^g \Delta t \right) \right)
\end{aligned}$$

对于 (1) 处比较好理解, 利用公式 (1.4) :

$$\text{Exp}(\vec{\phi} + \delta \vec{\phi}) \approx \text{Exp}(\vec{\phi}) \cdot \text{Exp} \left( J_r(\vec{\phi}) \cdot \delta \vec{\phi} \right)$$

后面是不是很眼熟, 在 3.1 节中。。

其中:  $\mathbf{J}_r^k = \mathbf{J}_r ((\widetilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g) \Delta t)$  有:

$$\Delta \overline{\widetilde{\mathbf{R}}}_{ij} \triangleq \Delta \widetilde{\mathbf{R}}_{ij} \cdot \text{Exp} \left( \frac{\partial \Delta \widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g \right)$$

因此:

$$\begin{aligned}
\frac{\partial \Delta \widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} &= \sum_{k=i}^{j-1} \left( -\Delta \widetilde{\mathbf{R}}_{k+1j}^T \mathbf{J}_r^k \Delta t \right) \\
&= \sum_{k=i}^{j-2} \left( -\Delta \widetilde{\mathbf{R}}_{k+1j}^T \mathbf{J}_r^k \Delta t \right) - \Delta \widetilde{\mathbf{R}}_{jj}^T \mathbf{j}_r^{j-1} \Delta t \\
&= \sum_{k=i}^{j-2} \left( - \left( \Delta \widetilde{\mathbf{R}}_{k+1j-1} \Delta \widetilde{\mathbf{R}}_{j-1j} \right)^T \mathbf{J}_r^k \Delta t \right) - \Delta \widetilde{\mathbf{R}}_{jj}^T \mathbf{J}_r^{j-1} \Delta t \\
&= \Delta \widetilde{\mathbf{R}}_{jj-1} \cdot \sum_{k=i}^{j-2} \left( - \left( \Delta \widetilde{\mathbf{R}}_{k+1j-1} \right)^T \mathbf{J}_r^k \Delta t \right) - \mathbf{J}_r^{j-1} \Delta t \\
&= \Delta \widetilde{\mathbf{R}}_{jj-1} \cdot \frac{\partial \Delta \widetilde{\mathbf{R}}_{ij-1}}{\partial \mathbf{b}^g} - \mathbf{J}_r^{j-1} \Delta t
\end{aligned}$$

### 3.5.2 对于 $\Delta \overline{\mathbf{v}}_{ij}$ 项

$$\begin{aligned}
\Delta \bar{\tilde{\mathbf{v}}}_{ij} &= \sum_{k=i}^{j-1} \left[ \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - (\mathbf{b}_i^a + \delta \mathbf{b}_i^a) \right) \cdot \Delta t \right] \\
&\approx \sum_{k=i}^{j-1} \left[ \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \text{Exp} \left( \frac{\partial \Delta \bar{\tilde{\mathbf{R}}}_{ik}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g \right) \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a - \delta \mathbf{b}_i^a \right) \Delta t \right] \\
&\approx \sum_{k=i}^{j-1} \left[ \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \left( \mathbf{I} + \left( \frac{\partial \Delta \bar{\tilde{\mathbf{R}}}_{ik}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g \right)^{\wedge} \right) \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a - \delta \mathbf{b}_i^a \right) \Delta t \right] \\
&= \sum_{k=i}^{j-1} \left[ \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right) \Delta t - \Delta \bar{\tilde{\mathbf{R}}}_{ik} \delta \mathbf{b}_i^a \Delta t + \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \left( \frac{\partial \Delta \bar{\tilde{\mathbf{R}}}_{ik}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g \right)^{\wedge} \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right) \Delta t \right. \\
&\quad \left. - \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \left( \frac{\partial \Delta \bar{\tilde{\mathbf{R}}}_{ik}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g \right)^{\wedge} \delta \mathbf{b}_i^a \Delta t \right] \\
&\approx \Delta \tilde{\mathbf{v}}_{ij} + \sum_{k=i}^{j-1} \left\{ - \left( \Delta \bar{\tilde{\mathbf{R}}}_{ik} \Delta t \right) \delta \mathbf{b}_i^a - \left[ \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right)^{\wedge} \frac{\partial \Delta \bar{\tilde{\mathbf{R}}}_{ik}}{\partial \mathbf{b}^g} \Delta t \right] \delta \mathbf{b}_i^g \right\}
\end{aligned}$$

所以：

$$\begin{aligned}
\frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g} &= - \sum_{k=i}^{j-1} \left( \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right)^{\wedge} \frac{\partial \Delta \bar{\tilde{\mathbf{R}}}_{ik}}{\partial \mathbf{b}^g} \Delta t \right) \\
\frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^a} &= - \sum_{k=i}^{j-1} \left( \Delta \bar{\tilde{\mathbf{R}}}_{ik} \Delta t \right) \times \\
\Delta \bar{\tilde{\mathbf{v}}}_{ij} &= \Delta \tilde{\mathbf{v}}_{ij} + \frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g + \frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^a} \delta \mathbf{b}_i^a
\end{aligned}$$

进一步推导：

$$\begin{aligned}
\frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g} &= - \sum_{k=i}^{j-1} \left( \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right)^{\wedge} \frac{\partial \Delta \bar{\tilde{\mathbf{R}}}_{ik}}{\partial \mathbf{b}^g} \Delta t \right) \\
&= \frac{\partial \Delta \tilde{\mathbf{v}}_{ij-1}}{\partial \mathbf{b}^g} - \left( \Delta \bar{\tilde{\mathbf{R}}}_{ij-1} \cdot \left( \tilde{\mathbf{f}}_{j-1} - \mathbf{b}_i^a \right)^{\wedge} \frac{\partial \Delta \bar{\tilde{\mathbf{R}}}_{ij-1}}{\partial \mathbf{b}^g} \Delta t \right) \\
\frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^a} &= - \sum_{k=i}^{j-1} \left( \Delta \bar{\tilde{\mathbf{R}}}_{ik} \Delta t \right) \\
&= \frac{\partial \Delta \tilde{\mathbf{v}}_{ij-1}}{\partial \mathbf{b}^a} - \Delta \bar{\tilde{\mathbf{R}}}_{ij-1} \Delta t
\end{aligned}$$

### 3.5.3 对于 $\Delta \bar{\tilde{\mathbf{p}}}_{ij}$ 项

$$\begin{aligned}
\Delta \bar{\tilde{\mathbf{p}}}_{ij} &\triangleq \sum_{k=i}^{j-1} \left[ \Delta \bar{\tilde{\mathbf{v}}}_{ik} \Delta t + \frac{1}{2} \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - (\mathbf{b}_i^a + \delta \mathbf{b}_i^a) \right) \Delta t^2 \right] \\
&= \sum_{k=i}^{j-1} \left[ \underbrace{\left( \Delta \tilde{\mathbf{v}}_{ik} + \frac{\partial \Delta \tilde{\mathbf{v}}_{ik}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g + \frac{\partial \Delta \tilde{\mathbf{v}}_{ik}}{\partial \mathbf{b}^a} \delta \mathbf{b}_i^a \right) \Delta t + }_{(1)} \right. \\
&\quad \left. \underbrace{\frac{1}{2} \Delta \bar{\tilde{\mathbf{R}}}_{ik} \cdot \text{Exp} \left( \frac{\partial \Delta \bar{\tilde{\mathbf{R}}}_{ik}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g \right) \cdot \left( \tilde{\mathbf{f}}_k - (\mathbf{b}_i^a + \delta \mathbf{b}_i^a) \right) \Delta t^2}_{(2)} \right]
\end{aligned}$$

对于 (1) 直接带入之前的结果；

对于 (2) :

$$\begin{aligned}
 (2) &\approx \frac{\Delta t^2}{2} \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \mathbf{I} + \left( \frac{\partial \Delta \tilde{\mathbf{R}}_{ik}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g \right)^\wedge \right) \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a - \delta \mathbf{b}_i^a \right) \right] \\
 &\approx \frac{\Delta t^2}{2} \sum_{k=i}^{j-1} \left[ \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right) - \Delta \tilde{\mathbf{R}}_{ik} \delta \mathbf{b}_i^a - \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right)^\wedge \frac{\partial \Delta \tilde{\mathbf{R}}_{ik}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g \right]
 \end{aligned}$$

最后有：

$$\begin{aligned}
 \frac{\partial \Delta \tilde{\mathbf{p}}_{ij}}{\partial \mathbf{b}^g} &= \sum_{k=i}^{j-1} \left[ \frac{\partial \Delta \tilde{\mathbf{v}}_{ik}}{\partial \mathbf{b}^g} \Delta t - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \cdot \left( \tilde{\mathbf{f}}_k - \mathbf{b}_i^a \right)^\wedge \frac{\partial \Delta \tilde{\mathbf{R}}_{ik}}{\partial \mathbf{b}^g} \Delta t^2 \right] \\
 &= \frac{\partial \Delta \tilde{\mathbf{p}}_{ij-1}}{\partial \mathbf{b}^g} + \left[ \frac{\partial \Delta \tilde{\mathbf{v}}_{ij-1}}{\partial \mathbf{b}^g} \Delta t - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ij-1} \cdot \left( \tilde{\mathbf{f}}_{j-1} - \mathbf{b}_i^a \right)^\wedge \frac{\partial \Delta \tilde{\mathbf{R}}_{ij-1}}{\partial \mathbf{b}^g} \Delta t^2 \right] \\
 \frac{\partial \Delta \tilde{\mathbf{p}}_{ij}}{\partial \mathbf{b}^a} &= \sum_{k=i}^{j-1} \left[ \frac{\partial \Delta \tilde{\mathbf{v}}_{ik}}{\partial \mathbf{b}^a} \Delta t - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ik} \Delta t^2 \right] \\
 &= \frac{\partial \Delta \tilde{\mathbf{p}}_{ij-1}}{\partial \mathbf{b}^a} + \left( \frac{\partial \Delta \tilde{\mathbf{v}}_{ij-1}}{\partial \mathbf{b}^a} \Delta t - \frac{1}{2} \Delta \tilde{\mathbf{R}}_{ij-1} \Delta t^2 \right)
 \end{aligned}$$

## 3.6 求残差关于状态量的雅可比

### 3.6.1 定义残差

$$\begin{aligned}
 \mathbf{r}_{\Delta \vec{\phi}_{ij}} &\triangleq \log \left[ \left( \Delta \tilde{\mathbf{R}}_{ij} \right)^T \Delta \mathbf{R}_{ij} \right] \\
 &= \log \left[ \left( \Delta \tilde{\mathbf{R}}_{ij} \cdot \text{Exp} \left( \frac{\partial \Delta \tilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g \right) \right)^T \mathbf{R}_{wi}^T \mathbf{R}_{wj} \right] \\
 \mathbf{r}_{\Delta v_{ij}} &\triangleq \Delta \mathbf{v}_{ij} - \Delta \tilde{\mathbf{v}}_{ij} \\
 &= \mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) - \left( \Delta \tilde{\mathbf{v}}_{ij} + \frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g + \frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^a} \delta \mathbf{b}_i^a \right) \\
 \mathbf{r}_{\Delta \mathbf{p}_{ij}} &\triangleq \Delta \mathbf{p}_{ij} - \Delta \tilde{\mathbf{p}}_{ij} \\
 &= \mathbf{R}_{wi}^T \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) - \left( \Delta \tilde{\mathbf{p}}_{ij} + \frac{\partial \Delta \tilde{\mathbf{p}}_{ij}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g + \frac{\partial \Delta \tilde{\mathbf{p}}_{ij}}{\partial \mathbf{b}^a} \delta \mathbf{b}_i^a \right)
 \end{aligned}$$

### 3.6.2 定义扰动

$$\begin{aligned}
 \mathbf{R}_{wi} &\leftarrow \mathbf{R}_{wi} \cdot \text{Exp} \left( \delta \vec{\phi}_i \right) \\
 \mathbf{p}_{wi} &\leftarrow \mathbf{p}_{wi} + \mathbf{R}_{wi} \cdot \delta \mathbf{p}_i \\
 \mathbf{v}_i &\leftarrow \mathbf{v}_i + \delta \mathbf{v}_i \\
 \delta \mathbf{b}_i^g &\leftarrow \delta \mathbf{b}_i^g + \delta \tilde{\mathbf{b}}_i^g \\
 \delta \mathbf{b}_i^a &\leftarrow \delta \mathbf{b}_i^a + \delta \tilde{\mathbf{b}}_i^a \\
 \mathbf{R}_{wj} &\leftarrow \mathbf{R}_{wj} \cdot \text{Exp} \left( \delta \vec{\phi}_j \right) \\
 \mathbf{p}_{wj} &\leftarrow \mathbf{p}_{wj} + \mathbf{R}_{wj} \cdot \delta \mathbf{p}_j \\
 \mathbf{v}_j &\leftarrow \mathbf{v}_j + \delta \mathbf{v}_j
 \end{aligned}$$

其中值得关注的有  $\mathbf{p}_{wi} \leftarrow \mathbf{p}_{wi} + \mathbf{R}_{wi} \cdot \delta \mathbf{p}_i$ ，设定位姿矩阵

$$\mathbf{T}_{wi} = \begin{bmatrix} \mathbf{R}_{wi} & \mathbf{p}_{wi} \\ 0 & 1 \end{bmatrix}$$

给一个右扰动

$$\delta\mathbf{T}_i = \begin{bmatrix} \delta\mathbf{R}_i & \delta\mathbf{p}_i \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{wi} \cdot \delta\mathbf{T}_i = \begin{bmatrix} \mathbf{R}_{wi} & \mathbf{p}_{wi} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \delta\mathbf{R}_i & \delta\mathbf{p}_i \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{wi}\delta\mathbf{R}_i & \mathbf{p}_{wi} + \mathbf{R}_{wi}\delta\mathbf{p}_i \\ 0 & 1 \end{bmatrix}$$

### 3.6.3 残差对于状态的雅可比

#### 3.6.3.1 旋转残差

对于：

$$\mathbf{r}_{\Delta\vec{\phi}_{ij}} = \log \left[ \left( \Delta\widetilde{\mathbf{R}}_{ij} \cdot \text{Exp} \left( \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\mathbf{b}_i^g \right) \right)^T \mathbf{R}_{wi}^T \mathbf{R}_{wj} \right]$$

其中不包含  $\mathbf{p}_{wi}, \mathbf{p}_{wj}, \mathbf{v}_i, \mathbf{v}_j$  以及  $\delta\mathbf{b}_i^a$ , 因此关于这些状态的雅可比矩阵都是  $\mathbf{0}$

下面分别推一下对于其他状态量的雅可比：

$$\begin{aligned} \mathbf{r}_{\Delta\vec{\phi}_{ij}} (\mathbf{R}_{wi} \text{Exp} (\delta\vec{\phi}_i)) &= \log \left[ \left( \Delta\widetilde{\mathbf{R}}_{ij} \right)^T \left( \mathbf{R}_{wi} \text{Exp} (\delta\vec{\phi}_i) \right)^T \mathbf{R}_{wj} \right] \\ &\stackrel{(1)}{=} \log \left[ \left( \Delta\widetilde{\mathbf{R}}_{ij} \right)^T \text{Exp} (-\delta\vec{\phi}_i) \mathbf{R}_{wi}^T \mathbf{R}_{wj} \right] \\ &\stackrel{(2)}{=} \log \left[ \left( \Delta\widetilde{\mathbf{R}}_{ij} \right)^T \mathbf{R}_{wi}^T \mathbf{R}_{wj} \text{Exp} (-\mathbf{R}_{wj}^T \mathbf{R}_{wi} \delta\vec{\phi}_i) \right] \\ &= \log \left\{ \text{Exp} \left[ \log \left( \left( \Delta\widetilde{\mathbf{R}}_{ij} \right)^T \mathbf{R}_{wi}^T \mathbf{R}_{wj} \right) \right] \cdot \text{Exp} \left( -\mathbf{R}_{wj}^T \mathbf{R}_{wi} \delta\vec{\phi}_i \right) \right\} \\ &= \log \left[ \text{Exp} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \right) \cdot \text{Exp} \left( -\mathbf{R}_{wj}^T \mathbf{R}_{wi} \delta\vec{\phi}_i \right) \right] \\ &\approx \mathbf{r}_{\Delta\vec{\phi}_{ij}} - \mathbf{J}_r^{-1} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \right) \mathbf{R}_{wj}^T \mathbf{R}_{wi} \delta\vec{\phi}_i \end{aligned}$$

得出：

$$\frac{\partial \mathbf{r}_{\Delta\vec{\phi}_{ij}}}{\partial \delta\vec{\phi}_i} = -\mathbf{J}_r^{-1} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \right) \mathbf{R}_{wj}^T \mathbf{R}_{wi}$$

对于  $\mathbf{R}_{wj}$

$$\begin{aligned}
\mathbf{r}_{\Delta\vec{\phi}_{ij}} \left( \mathbf{R}_{wj} \text{Exp} \left( \delta\vec{\phi}_j \right) \right) &= \log \left[ \left( \Delta\overline{\mathbf{R}}_{ij} \right)^T \mathbf{R}_{wi}^T \mathbf{R}_{wj} \text{Exp} \left( \delta\vec{\phi}_j \right) \right] \\
&= \log \left\{ \text{Exp} \left[ \log \left( \left( \Delta\overline{\mathbf{R}}_{ij} \right)^T \mathbf{R}_{wi}^T \mathbf{R}_{wj} \right) \right] \cdot \text{Exp} \left( \delta\vec{\phi}_j \right) \right\} \\
&= \log \left\{ \text{Exp} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \right) \cdot \text{Exp} \left( \delta\vec{\phi}_j \right) \right\} \\
&\approx \mathbf{r}_{\Delta\vec{\phi}_{ij}} + \mathbf{J}_r^{-1} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \right) \delta\vec{\phi}_j \\
\frac{\partial \mathbf{r}_{\Delta\vec{\phi}_{ij}}}{\partial \delta\vec{\phi}_j} &= \mathbf{J}_r^{-1} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \right)
\end{aligned}$$

对于  $\delta\mathbf{b}_i^g$

$$\begin{aligned}
&\mathbf{r}_{\Delta\vec{\phi}_{ij}} \left( \delta\mathbf{b}_i^g + \delta\tilde{\mathbf{b}}_i^g \right) \\
&= \log \left\{ \left[ \Delta\widetilde{\mathbf{R}}_{ij} \cdot \text{Exp} \left( \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \left( \delta\mathbf{b}_i^g + \delta\tilde{\mathbf{b}}_i^g \right) \right) \right]^T \mathbf{R}_{wi}^T \mathbf{R}_{wj} \right\} \\
&\stackrel{(1)}{\approx} \log \left\{ \left[ \Delta\widetilde{\mathbf{R}}_{ij} \cdot \text{Exp} \left( \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\mathbf{b}_i^g \right) \text{Exp} \left( \mathbf{J}_r \left( \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\mathbf{b}_i^g \right) \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\tilde{\mathbf{b}}_i^g \right) \right]^T \mathbf{R}_{wi}^T \mathbf{R}_{wj} \right\} \\
&\stackrel{(2)}{=} \log \left\{ \left[ \Delta\overline{\mathbf{R}}_{ij} \cdot \text{Exp} \left( \mathbf{J}_r \cdot \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\tilde{\mathbf{b}}_i^g \right) \right]^T \Delta\mathbf{R}_{ij} \right\} \\
&\stackrel{(3)}{=} \log \left[ \text{Exp} \left( -\mathbf{J}_r \cdot \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\tilde{\mathbf{b}}_i^g \right) \Delta\overline{\mathbf{R}}_{ij}^T \Delta\mathbf{R}_{ij} \right] \\
&= \log \left[ \text{Exp} \left( -\mathbf{J}_r \cdot \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\tilde{\mathbf{b}}_i^g \right) \text{Exp} \left( \log \left( \Delta\overline{\mathbf{R}}_{ij}^T \Delta\mathbf{R}_{ij} \right) \right) \right] \\
&= \log \left[ \text{Exp} \left( -\mathbf{J}_r \cdot \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\tilde{\mathbf{b}}_i^g \right) \text{Exp} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \left( \delta\mathbf{b}_i^g \right) \right) \right] \\
&\stackrel{(4)}{=} \log \left\{ \text{Exp} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \left( \delta\mathbf{b}_i^g \right) \right) \cdot \text{Exp} \left[ -\text{Exp} \left( -\mathbf{r}_{\Delta\vec{\phi}_{ij}} \left( \delta\mathbf{b}_i^g \right) \right) \cdot \mathbf{J}_r \cdot \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\tilde{\mathbf{b}}_i^g \right] \right\} \\
&\approx \mathbf{r}_{\Delta\vec{\phi}_{ij}} \left( \delta\mathbf{b}_i^g \right) - \mathbf{J}_r^{-1} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \left( \delta\mathbf{b}_i^g \right) \right) \cdot \text{Exp} \left( -\mathbf{r}_{\Delta\vec{\phi}_{ij}} \left( \delta\mathbf{b}_i^g \right) \right) \cdot \mathbf{J}_r \cdot \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\tilde{\mathbf{b}}_i^g \\
&\stackrel{(6)}{=} \mathbf{r}_{\Delta\vec{\phi}_{ij}} - \mathbf{J}_r^{-1} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \right) \cdot \text{Exp} \left( -\mathbf{r}_{\Delta\vec{\phi}_{ij}} \right) \cdot \mathbf{J}_r \left( \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\mathbf{b}_i^g \right) \cdot \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \cdot \delta\tilde{\mathbf{b}}_i^g \\
\frac{\partial \mathbf{r}_{\Delta\vec{\phi}_{ij}}}{\partial \delta\tilde{\mathbf{b}}_i^g} &= -\mathbf{J}_r^{-1} \left( \mathbf{r}_{\Delta\vec{\phi}_{ij}} \right) \cdot \text{Exp} \left( -\mathbf{r}_{\Delta\vec{\phi}_{ij}} \right) \cdot \mathbf{J}_r \left( \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g} \delta\mathbf{b}_i^g \right) \cdot \frac{\partial \Delta\widetilde{\mathbf{R}}_{ij}}{\partial \mathbf{b}^g}
\end{aligned}$$

### 3.6.3.2 速度残差

$$\begin{aligned}
\mathbf{r}_{\Delta\mathbf{v}_{ij}} &\triangleq \Delta\mathbf{v}_{ij} - \Delta\tilde{\mathbf{v}}_{ij} \\
&= \mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) - \left( \Delta\tilde{\mathbf{v}}_{ij} + \frac{\partial \Delta\tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g} \delta\mathbf{b}_i^g + \frac{\partial \Delta\tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^a} \delta\mathbf{b}_i^a \right)
\end{aligned}$$

其中不包含  $\mathbf{p}_{wi}$ ,  $\mathbf{p}_{wj}$ ,  $\mathbf{R}_{wj}$ , 因此关于这些状态的雅可比矩阵都是  $\mathbf{0}$

关于  $\delta\mathbf{b}_i^g$ ,  $\delta\mathbf{b}_i^a$  可以直接得出结论

$$\begin{aligned}
\frac{\partial \mathbf{r}_{\Delta\mathbf{v}_{ij}}}{\partial \delta\tilde{\mathbf{b}}_i^g} &= -\frac{\partial \Delta\tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g} \\
\frac{\partial \mathbf{r}_{\Delta\mathbf{v}_{ij}}}{\partial \delta\tilde{\mathbf{b}}_i^a} &= -\frac{\partial \Delta\tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^a}
\end{aligned}$$

关于  $\mathbf{v}_i$

$$\begin{aligned}\mathbf{r}_{\Delta \mathbf{v}_{ij}}(\mathbf{v}_i + \delta \mathbf{v}_i) &= \mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \delta \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) - \left( \Delta \tilde{\mathbf{v}}_{ij} + \frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g + \frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^a} \delta \mathbf{b}_i^a \right) \\ &= \mathbf{r}_{\Delta \mathbf{v}_{ij}}(\mathbf{v}_i) - \mathbf{R}_{wi}^T \delta \mathbf{v}_i\end{aligned}$$

得出：

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{ij}}}{\partial \delta \mathbf{v}_i} = -\mathbf{R}_{wi}^T$$

关于  $\mathbf{v}_j$

$$\begin{aligned}\mathbf{r}_{\Delta \mathbf{v}_{ij}}(\mathbf{v}_j + \delta \mathbf{v}_j) &= \mathbf{R}_{wi}^T (\mathbf{v}_j + \delta \mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) - \left( \Delta \tilde{\mathbf{v}}_{ij} + \frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g + \frac{\partial \Delta \tilde{\mathbf{v}}_{ij}}{\partial \mathbf{b}^a} \delta \mathbf{b}_i^a \right) \\ &= \mathbf{r}_{\Delta \mathbf{v}_{ij}}(\mathbf{v}_i) + \mathbf{R}_{wi}^T \delta \mathbf{v}_j\end{aligned}$$

得出：

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{ij}}}{\partial \delta \mathbf{v}_j} = \mathbf{R}_{wi}^T$$

关于  $\mathbf{R}_{wi}$

$$\begin{aligned}\mathbf{r}_{\Delta \mathbf{v}_{ij}}\left(\mathbf{R}_{wi} \operatorname{Exp}\left(\delta \vec{\phi}_i\right)\right) &= \left(\mathbf{R}_{wi} \operatorname{Exp}\left(\delta \vec{\phi}_i\right)\right)^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) - \Delta \tilde{\mathbf{v}}_{ij} \\ &\stackrel{(1)}{=} \operatorname{Exp}\left(-\delta \vec{\phi}_i\right) \cdot \mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) - \Delta \tilde{\mathbf{v}}_{ij} \\ &\approx \left(\mathbf{I} - \left(\delta \vec{\phi}_i\right)^\wedge\right) \cdot \mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) - \Delta \tilde{\mathbf{v}}_{ij} \\ &= \mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) - \Delta \tilde{\mathbf{v}}_{ij} - \left(\delta \vec{\phi}_i\right)^\wedge \cdot \mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij}) \\ &\stackrel{(2)}{=} \mathbf{r}_{\Delta \mathbf{v}_{ij}}(\mathbf{R}_{wi}) + [\mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij})]^\wedge \cdot \delta \vec{\phi}_i\end{aligned}$$

得出：

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{ij}}}{\partial \delta \vec{\phi}_i} = [\mathbf{R}_{wi}^T (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g} \cdot \Delta t_{ij})]^\wedge$$

### 3.6.3.2 位置残差

$$\begin{aligned}\mathbf{r}_{\Delta \mathbf{p}_{ij}} &\triangleq \Delta \mathbf{p}_{ij} - \Delta \bar{\mathbf{p}}_{ij} \\ &= \mathbf{R}_{wi}^T \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) - \left( \Delta \tilde{\mathbf{p}}_{ij} + \frac{\partial \Delta \tilde{\mathbf{p}}_{ij}}{\partial \mathbf{b}^g} \delta \mathbf{b}_i^g + \frac{\partial \Delta \tilde{\mathbf{p}}_{ij}}{\partial \mathbf{b}^a} \delta \mathbf{b}_i^a \right)\end{aligned}$$

其中不包含  $\mathbf{v}_j$ ,  $\mathbf{R}_{wj}$ , 因此关于这些状态的雅可比矩阵都是  $\mathbf{0}$

关于  $\delta \mathbf{b}_i^g$ ,  $\delta \mathbf{b}_i^a$  可以直接得出结论

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \delta \tilde{\mathbf{b}}_i^g} = -\frac{\partial \Delta \tilde{\mathbf{p}}_{ij}}{\partial \mathbf{b}^g}$$

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \delta \tilde{\mathbf{b}}_i^a} = -\frac{\partial \Delta \tilde{\mathbf{p}}_{ij}}{\partial \mathbf{b}^a}$$

关于  $\mathbf{p}_{wj}$

$$\begin{aligned} & \mathbf{r}_{\Delta \mathbf{p}_{ij}} (\mathbf{p}_{wj} + \mathbf{R}_{wj} \cdot \delta \mathbf{p}_j) \\ &= \mathbf{R}_{wi}^T \left( \mathbf{p}_{wj} + \mathbf{R}_{wj} \cdot \delta \mathbf{p}_j - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) - \Delta \bar{\mathbf{p}}_{ij} \\ &= \mathbf{r}_{\Delta \mathbf{p}_{ij}} (\mathbf{p}_{wj}) + \mathbf{R}_{wi}^T \mathbf{R}_{wj} \cdot \delta \mathbf{p}_j \end{aligned}$$

得出：

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \delta \mathbf{p}_j} = \mathbf{R}_{wi}^T \mathbf{R}_{wj}$$

关于  $\mathbf{p}_{wi}$

$$\begin{aligned} & \mathbf{r}_{\Delta \mathbf{p}_{ij}} (\mathbf{p}_{wi} + \mathbf{R}_{wi} \cdot \delta \mathbf{p}_i) \\ &= \mathbf{R}_{wi}^T \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{R}_{wi} \cdot \delta \mathbf{p}_i - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) - \Delta \bar{\mathbf{p}}_{ij} \\ &= \mathbf{r}_{\Delta \mathbf{p}_{ij}} (\mathbf{p}_{wi}) - \mathbf{I} \cdot \delta \mathbf{p}_i \end{aligned}$$

得出：

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \delta \mathbf{p}_i} = -\mathbf{I}$$

关于  $\mathbf{v}_i$ ：

$$\begin{aligned} \mathbf{r}_{\Delta \mathbf{p}_{ij}} (\mathbf{v}_i + \delta \mathbf{v}_i) &= \mathbf{R}_{wi}^T \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \delta \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) - \Delta \bar{\mathbf{p}}_{ij} \\ &= \mathbf{r}_{\Delta \mathbf{p}_{ij}} (\mathbf{p}_{wj}) - \mathbf{R}_{wi}^T \Delta t_{ij} \cdot \delta \mathbf{v}_i \end{aligned}$$

得出：

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \delta \mathbf{v}_i} = -\mathbf{R}_{wi}^T \Delta t_{ij}$$

关于  $\mathbf{R}_{wi}$

$$\begin{aligned} & \mathbf{r}_{\Delta \mathbf{v}_{ij}} \left( \mathbf{R}_{wi} \text{Exp} \left( \delta \vec{\phi}_i \right) \right) \\ &= \left( \mathbf{R}_{wi} \text{Exp} \left( \delta \vec{\phi}_i \right) \right)^T \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) - \Delta \bar{\mathbf{p}}_{ij} \\ &\stackrel{(1)}{=} \text{Exp} \left( -\delta \vec{\phi}_i \right) \cdot \mathbf{R}_{wi}^T \cdot \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) - \Delta \bar{\mathbf{p}}_{ij} \\ &\stackrel{(2)}{\approx} \left( \mathbf{I} - \left( \delta \vec{\phi}_i \right)^\wedge \right) \cdot \mathbf{R}_{wi}^T \cdot \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) - \Delta \bar{\mathbf{p}}_{ij} \\ &= \mathbf{R}_{wi}^T \cdot \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) - \Delta \bar{\mathbf{p}}_{ij} - \left( \delta \vec{\phi}_i \right)^\wedge \mathbf{R}_{wi}^T \\ &\quad \cdot \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) \\ &\stackrel{(3)}{=} \mathbf{r}_{\Delta \mathbf{p}_{ij}} + \left[ \mathbf{R}_{wi}^T \cdot \left( \mathbf{p}_{wj} - \mathbf{p}_{wi} - \mathbf{v}_i \cdot \Delta t_{ij} - \frac{1}{2} \mathbf{g} \cdot \Delta t_{ij}^2 \right) \right]^\wedge \cdot \delta \vec{\phi}_i \end{aligned}$$

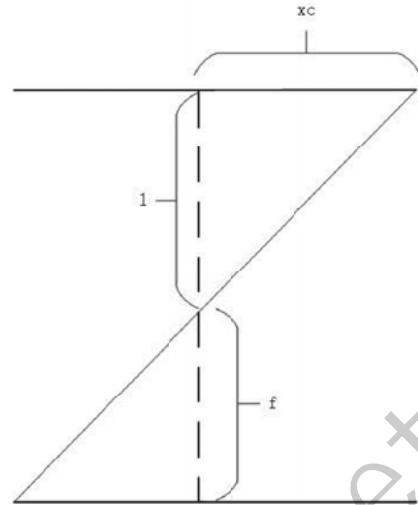
就很突然，但是确实预积分就这么完了~恭喜过关！

## 第4章 IMU预积分、初始化及优化代码解析

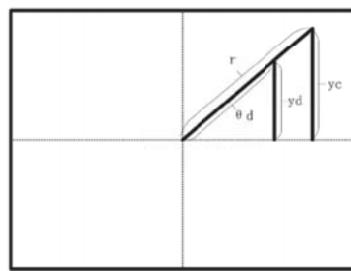
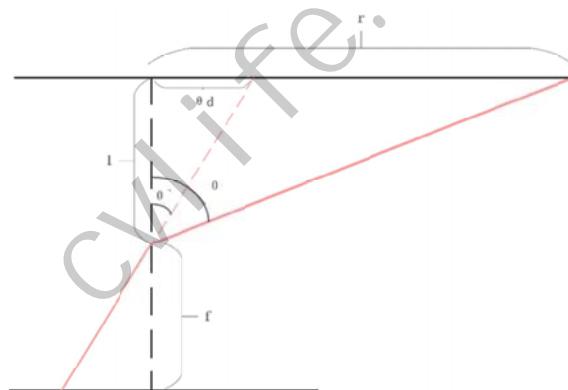
具体内容见ORB-SLAM3第③期课程视频

## 第5章：相机抽象模型

### 相机模型



无畸变投影过程



有畸变投影过程

### KB8模型的投影过程

$$\text{计算归一化坐标 } x_c = \frac{X_c}{Z_c}, \quad y_c = \frac{Y_c}{Z_c}$$

$$\text{计算} r \text{与} \theta, \quad r^2 = x_c^2 + y_c^2 \quad \theta = \arctan(r)$$

$$\text{计算 } \theta_d = k_0\theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9$$

计算去畸变后的归一化平面坐标

$$x_d = \frac{\theta_d}{r} \cdot x_c \quad y_d = \frac{\theta_d}{r} \cdot y_c$$

最后计算像素点  $u = f_x \cdot x_d + c_x \quad v = f_y \cdot y_d + c_y$

## KB8模型的反投影过程

计算去畸变后的归一化平面坐标，uv已知

$$x_d = \frac{(u - c_x)}{f_x} \quad y_d = \frac{(v - c_y)}{f_y}$$

最后的目的是计算  $x_c$  与  $y_c$

$$x_c = \frac{x_d \cdot r}{\theta_d} \quad y_c = \frac{y_d \cdot r}{\theta_d}$$

其中  $r, \theta_d$  未知，首先计算  $\theta_d$

$$x_d^2 + y_d^2 = \frac{(x_c^2 + y_c^2) \cdot \theta_d^2}{r^2}$$

继续推

$$\theta_d = \frac{\sqrt{(x_d^2 + y_d^2) \cdot r}}{\sqrt{(x_c^2 + y_c^2) \cdot r}}$$

其中

$$r = \sqrt{x_c^2 + y_c^2}$$

$$\theta_d = \sqrt{(x_d^2 + y_d^2)}$$

其次计算  $r$ ，由于  $\theta = \arctan(r)$  所以相当于求  $\theta$

$$\theta_d = k_0 \theta + k_1 \theta^3 + k_2 \theta^5 + k_3 \theta^7 + k_4 \theta^9$$

比较难求，因此选用迭代的方式计算，设

$$f(\theta) = k_0 \theta + k_1 \theta^3 + k_2 \theta^5 + k_3 \theta^7 + k_4 \theta^9$$

建立误差  $e(\theta) = f(\theta) - \theta_d$  目标是求解一个  $\theta$  使  $e(\theta) = 0$

求  $e(\theta)$  相对于  $\theta$  的导数

$$\frac{d(e(\theta))}{d(\theta)} = k_0 + 3k_1 \theta^2 + 5k_2 \theta^4 + 7k_3 \theta^6 + 9k_4 \theta^8$$

给  $\theta$  扰动，泰勒展开

$$e(\theta + \delta\theta) = e(\theta) + \frac{d(e(\theta))}{d(\theta)} \delta\theta = 0$$

$$\delta\theta = -\frac{e(\theta)}{\frac{d(e(\theta))}{d(\theta)}}$$

像素坐标对三维点求雅可比

$$u = f_x \cdot \frac{k_0\theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9}{\sqrt{x_c^2 + y_c^2}} \cdot x_c + c_x$$

$$v = f_y \cdot \frac{k_0\theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9}{\sqrt{x_c^2 + y_c^2}} \cdot y_c + c_y$$

$$\theta = \arctan\left(\sqrt{x_c^2 + y_c^2}\right)$$

$$x_c = \frac{X_c}{Z_c} \quad y_c = \frac{Y_c}{Z_c}$$

太长了。。。就以v对Zc求导为例吧

$$v = f_y \cdot \frac{k_0\theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9}{\sqrt{x_c^2 + y_c^2}} \cdot y_c + c_y$$

$$= f_y \cdot (k_0\theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9) \frac{1}{\sqrt{\frac{x_c^2}{y_c^2} + 1}} + c_y$$

$$\frac{\partial v}{\partial Z_c} = \frac{f_y \cdot y_c}{\sqrt{x_c^2 + y_c^2}} \cdot (k_0 + 3k_1\theta^2 + 5k_2\theta^4 + 7k_3\theta^6 + 9k_4\theta^8)$$

$$\cdot \frac{1}{1 + \frac{x_c^2}{y_c^2}} \cdot \frac{\partial \sqrt{x_c^2 + y_c^2}}{\partial Z_c}$$

$$\frac{\partial \sqrt{x_c^2 + y_c^2}}{\partial z_c} = \frac{1}{2\sqrt{x_c^2 + y_c^2}} \cdot \frac{\partial (x_c^2 + y_c^2)}{\partial z_c}$$

$$= -\frac{1}{\sqrt{x_c^2 + y_c^2}} \cdot \left( \frac{X_c^2 + Y_c^2}{Z_c^3} \right)$$

$$\frac{\partial v}{\partial Z_c} = -\frac{f_y \cdot y_c \cdot f_d}{x_c^2 + y_c^2} \cdot \frac{1}{1 + x_c^2 + y_c^2} \cdot \frac{X_c^2 + Y_c^2}{Z_c^3}$$

$$= -\frac{f_y \cdot \frac{Y_c}{Z_c} \cdot f_d}{1 + x_c^2 + y_c^2} \cdot \frac{1}{Z_c^1}$$

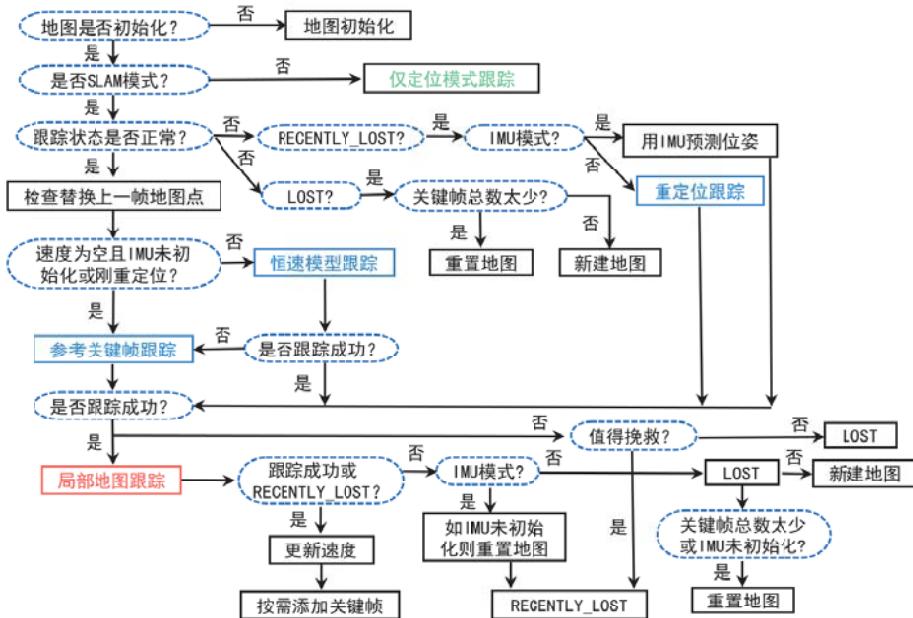
$$= -\frac{f_y \cdot Y_c \cdot f_d}{Z_c^2 + X_c^2 + Y_c^2}$$

## 第6章：跟踪线程

### 跟踪线程的目的和意义

ORB-SLAM3跟踪部分主要包括两个阶段，第一个阶段包括三种跟踪方法：用参考关键帧来跟踪、恒速模型跟踪、重定位跟踪，它们的目的是保证能够“跟的上”，但估计出来的位姿可能没那么准确。第二个阶段是局部地图跟踪，将当前帧的局部关键帧对应的局部地图点投影到该帧，得到更多的特征点匹配关系，对第一阶段的位姿再次优化得到相对准确的位姿。

### 跟踪线程的流程图



## 跟踪线程的新变化

新增加了一种跟踪状态：RECENTLY\_LOST

如下图所示是ORB-SLAM2和ORB-SLAM3中跟踪状态的对比：

```

1 // ORB-SLAM2跟踪状态类型
2 enum eTrackingState{
3     SYSTEM_NOT_READY=-1,           //系统没有准备好的状态,一般就是在启动后加载配置文件和设
4     NO_IMAGES_YET=0,               //当前无图像
5     NOT_INITIALIZED=1,             //有图像但是没有完成初始化
6     OK=2,                          //正常时候的工作状态
7     LOST=3,                         //系统已经跟丢了的状态
8 };
9
10 // ORB-SLAM3跟踪状态类型
11 enum eTrackingState{
12     SYSTEM_NOT_READY=-1,           //系统没有准备好的状态,一般就是在启动后加载配置文
13     NO_IMAGES_YET=0,               //当前无图像
14     NOT_INITIALIZED=1,             //有图像但是没有完成初始化
15     OK=2,                          //正常跟踪状态
16     RECENTLY_LOST=3,              //IMU模式: 当前地图中的KF>10,且丢失时间<5秒。纯视
17     LOST=4,                         //IMU模式: 当前帧跟丢超过5s。纯视觉模式: 重定位失
18     OK_KLT=5,                      //未使用
19 };

```

## 为什么要增加RECENTLY\_LOST状态？

当参考关键帧跟踪或恒速模型跟踪失败的时候，根据不同的传感器模式分配不同的状态：

- 如果是纯视觉模式，直接将当前状态标记为LOST。
- 如果是视觉+IMU模式，并且满足一定的条件，先将当前状态标记为RECENTLY\_LOST。然后用累积的IMU数据来预测一个粗糙的位姿，希望能够把跟丢的位姿重新找回来。

跟踪线程中对应具体代码：

```

1 // 参考关键帧、恒速模型跟踪都失败时，新增了状态RECENTLY_LOST
2 // 如果经过跟踪参考关键帧、恒速模型跟踪都失败的话，并满足一定条件就要标记为RECENTLY_LOST

```

```

3 if (!bOK)
4 {
5     // 条件1：如果当前帧距离上次重定位成功不到1s
6     // mnFramesToResetIMU 表示经过多少帧后可以重置IMU，一般设置为和帧率相同，对
7     // 条件2：单目+IMU 或者 双目+IMU模式
8     // 同时满足条件1, 2， 标记为LOST
9     if ( mCurrentFrame.mnId<=(mnLastRelocFrameId+mnFramesToResetIMU) &&
10         (mSensor==System::IMU_MONOCULAR || mSensor==System::IMU_STEREO))
11     {
12         mState = LOST;
13     }
14     else if(pCurrentMap->KeyFramesInMap(>10)
15     {
16         // 条件1：当前地图中关键帧数目较多（大于10）
17         // 条件2（隐藏条件）：当前帧距离上次重定位帧超过1s（说明还比较争气，值的救）或者
18         // 同时满足条件1, 2，则将状态标记为RECENTLY_LOST，后面会结合IMU预测的位姿看看能
19         cout << "KF in map: " << pCurrentMap->KeyFramesInMap() << endl;
20         mState = RECENTLY_LOST;
21         // 记录丢失时间
22         mTimeStampLost = mCurrentFrame.mTimeStamp;
23         //mCurrentFrame.SetPose(mLastFrame.mTcw);
24     }
25     else
26     {
27         mState = LOST;
28     }
29 }

```

下面是RECENTLY\_LOST状态下不同模式的处理方法：

```

1 // RECENTLY_LOST状态下的处理
2 if (mState == RECENTLY_LOST)
3 {
4     Verbose::PrintMess("Lost for a short time", Verbose::VERBOSITY_NORMAL);
5     // bOK先置为true
6     bOK = true;
7     // 如果是IMU模式，用IMU数据预测位姿
8     if((mSensor == System::IMU_MONOCULAR || mSensor == System::IMU_STEREO))
9     {
10        // 如果当前地图中IMU已经成功初始化，就用IMU数据预测位姿
11        if(pCurrentMap->isImuInitialized())
12            PredictStateIMU();
13        else
14            bOK = false;
15
16        // 如果IMU模式下当前帧距离跟丢帧超过5s还没有找回 (time_recently_lost默认为5s)
17        // 放弃了，将RECENTLY_LOST状态改为LOST
18        if (mCurrentFrame.mTimeStamp-mTimeStampLost>time_recently_lost)
19        {
20            mState = LOST;
21            Verbose::PrintMess("Track Lost...", Verbose::VERBOSITY_NORMAL);
22            bOK=false;
23        }
24    }
25    else
26    {
27        // 纯视觉模式则进行重定位。主要是BOW搜索，EPnP求解位姿

```

```

28         bOK = Relocalization();
29
30     {
31         // 纯视觉模式下重定位失败, 状态为LOST
32         mState = LOST;
33         Verbose::PrintMess("Track Lost...", Verbose::VERBOSITY_NORMAL);
34         bOK=false;
35     }
36 }
37 }
```

## 参考关键帧的跟踪新变化

基本一致。不同之处：

在ORB-SLAM2中，最后位姿优化后，需要成功匹配内点数目超过10才成功；

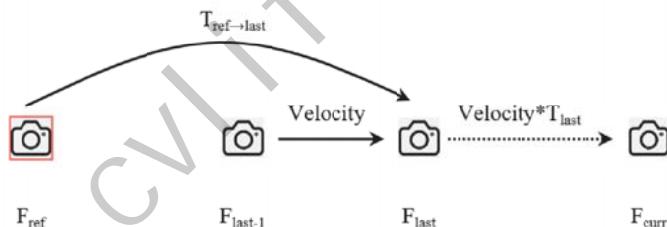
在ORB-SLAM3中，最后位姿优化后，如果是IMU模式，直接认为成功跟踪；如果是纯视觉模式，需要成功匹配内点数目超过10才成功；

总结：IMU模式下跟踪成功判断更宽松。

## 恒速模型跟踪新变化

在ORB-SLAM2中

1. 直接用位姿差来作为速度。
2. 在扩大搜索半径搜索后，如果匹配对还是小于20的情况下，认为跟踪失败。
3. 最后位姿优化后，需要成功匹配内点数目超过10才成功；



在ORB-SLAM3中，

1. 如果是IMU模式下，如果IMU完成初始化 并且 距离重定位挺久不需要重置IMU，用IMU来估计位姿；否则用位姿差来作为速度。
2. 在扩大搜索半径搜索后，如果匹配对还是小于20的情况下，如果是IMU模式则认为成功跟踪，否则认为跟踪失败。
3. 最后位姿优化后，如果是IMU模式，直接认为成功跟踪；如果是纯视觉模式，需要成功匹配内点数目超过10才成功；

## 重定位的新变化

如果相机跟丢了，利用当前帧查询多地图中DBow的数据库。这个查询能在当前的地图中（论文中说所有地图中）找到相似的关键帧。一旦有了候选关键帧，地图和匹配的地图点，就可以用 Tracking::Relocalization() 函数进行重定位。

- **Tracking thread** processes sensor information and computes the pose of the current frame with respect to the **active map** in real-time, minimizing the reprojection error of the matched map features. It also decides whether the current frame becomes a keyframe. In **visual-inertial** mode, the **body velocity** and **IMU biases** are estimated by including the **inertial residuals** in the optimization. When tracking is lost, the tracking thread tries to relocate the current frame **in all the Atlas' maps**. If relocated, tracking is resumed, switching the **active map** if needed. Otherwise, after a certain time, the active map is stored as non-active, and a new active map is initialized from scratch.

```

bool Tracking::Relocalization()
{
    Verbose::PrintMess("Starting relocalization", Verbose::VERBOSITY_NORMAL);
    // Compute Bag of Words Vector
    // Step 1: 计算当前帧特征点的Bow映射
    mCurrentFrame.ComputeBoW();

    // Relocalization is performed when tracking is lost
    // Track Lost: Query KeyFrame Database for keyframe candidates for relocalisation
    // Step 2: 找到与当前帧相似的候选关键帧组
    vector<KeyFrame*> vpCandidateKFs = mpKeyFrameDB->DetectRelocalizationCandidates(&mCurrentFrame, mpAtlas->GetCurrentMap()); // mpAtlas->GetCurrentMap()

    // 如果没有候选关键帧，则退出
    if(vpCandidateKFs.empty()) {
        Verbose::PrintMess("There are not candidates", Verbose::VERBOSITY_NORMAL);
        return false;
    }
}

```

我个人认为代码设计模式更好，重定位起一个短时追回的作用，所以适合在当前地图找。而新建地图后去所有地图找更合适，可以交给闭环线程慢慢找

在ORB-SLAM2中进行重定位。主要利用EPnP进行鲁棒的位姿估计和RANSAC阶段，然后进行引导搜索匹配、仅优化位姿的BA。

在ORB-SLAM3中进行重定位。基本流程一样，只不过将EPnP（至少需要4对点）换成了MLPnP（至少需要6对点）。主要原因是EPnP是根据标定好的针孔相机模型推导的，不具有普适性。MLPnP将相机模型解耦合了，更加通用。

## 局部地图跟踪的新变化

在ORB-SLAM2中：仅优化位姿；只要跟踪的地图点大于30个就认为成功。

在ORB-SLAM3中：

1. 如果IMU未初始化，或者虽然初始化成功但距离上次重定位时间比较近，仅优化位姿；否则，如果地图未更换（mbMapUpdated为false），使用上一普通帧+当前帧的视觉信息和IMU信息联合优化当前帧位姿（PoseInertialOptimizationLastFrame），如果地图更换（mbMapUpdated为true），使用上一关键帧+当前帧的视觉信息和IMU信息联合优化当前帧位姿（PoseInertialOptimizationLastKeyFrame）。
2. 定义跟踪成功：RECENTLY\_LOST状态下，至少成功跟踪10个才算成功。IMU模式下至少成功跟踪15个才算成功。以上情况都不满足，只要跟踪的地图点大于30个就认为成功。

总结下都在什么时候地图更新（mbMapUpdated为true）？

查询mnMapChange、IncreaseChangeIndex()、Map::ApplyScaledRotation

1. 闭环线程中矫正闭环、融合地图、全局BA等

// LoopClosing.cc 中CorrectLoop()函数中

pLoopMap->IncreaseChangeIndex();

// LoopClosing.cc 中MergeLocal()函数中

pMergeMap->IncreaseChangeIndex();

## 2. 局部建图线程中IMU三阶段的初始化

LocalMapping::InitializeIMU

LocalMapping::ScaleRefinement()

### 3. 优化过程中：

// LoopClosing.cc 中RunGlobalBundleAdjustment()函数中

Optimizer::FullInertialBA

Optimizer::OptimizeEssentialGraph

Optimizer::LocalInertialBA

Optimizer::MergeInertialBA

## 插入关键帧

ORB-SLAM3中：什么时候需要插入关键帧？

- IMU模式 && 当前地图中未完成IMU初始化 && 当前帧距离上一帧时间戳超过0.25s，直接插入
- 满足条件  $((c1a||c1b||c1c) \&\& c2) || c3 || c4$ ，如果局部建图空闲直接插入；否则中断局部建图BA，双目或双目+IMU或RGB-D模式下，如队列里没有阻塞太多关键帧，可以插入

```
1 // 判断是否需要插入关键帧
2 bool bNeedKF = NeedNewKeyFrame();
3
4 // 根据条件来判断是否插入关键帧
5 // 需要同时满足下面条件1和2
6 // 条件1: bNeedKF=true, 需要插入关键帧
7 // 条件2: bOK=true跟踪成功 或 IMU模式下的RECENTLY_LOST模式
8 if(bNeedKF && (bOK || (mState==RECENTLY_LOST && (mSensor == System::IMU_MONOCULAR
9      // 创建关键帧, 对于双目或RGB-D会产生新的地图点
10     CreateNewKeyFrame();
```

## 跟踪中的注意事项

ORB-SLAM3中如何防止不跟丢？

### D. Robustness to tracking loss

In pure visual SLAM or VO systems, temporal camera occlusion and fast motions result in losing track of visual elements, getting the system lost. ORB-SLAM pioneered the use of fast relocation techniques based on bag-of-words place recognition, but they proved insufficient to solve difficult sequences in the EuRoC dataset [3]. Our visual-inertial system enters into visually lost state when less than 15 point maps are tracked, and achieves robustness in two stages:

- **Short-term lost:** the current body state is estimated from IMU readings, and map points are projected in the estimated camera pose and searched for matches within a large image window. The resulting matches are included in visual-inertial optimization. In most cases this allows to recover visual tracking. Otherwise, after 5 seconds, we pass to the next stage.
- **Long-term lost:** A new visual-inertial map is initialized as explained above, and it becomes the active map.

提炼重点：

短时间跟丢 (RECENTLY\_LOST状态) : 用IMU数据来预测位姿, 对应函数 Tracking::PredictStateIMU()。然后在跟踪局部地图中用估计的位姿在更大的图像窗口中来进行投影匹配。

在ORB-SLAM2中

```
1 // Step 3: 如果需要进行投影匹配的点的数目大于0, 就进行投影匹配, 增加更多的匹配关系
2 if(nToMatch>0)
3 {
4     ORBmatcher matcher(0.8);
5     int th = 1;
6     if(mSensor==System::RGBD)    //RGBD相机输入的时候, 搜索的阈值会变得稍微大一些
7         th=3;
8
9     // If the camera has been relocalised recently, perform a coarser search
10    // 如果不久前进行过重定位, 那么进行一个更加宽泛的搜索, 阈值需要增大
11    if(mCurrentFrame.mnId<mnLastRelocFrameId+2)
12        th=5;
13
14    // 投影匹配得到更多的匹配关系
15    matcher.SearchByProjection(mCurrentFrame, mvpLocalMapPoints, th);
16 }
```

在ORB-SLAM3中, RECENTLY\_LOST状态下将搜索窗口阈值提高到了15

```
1 // Step 3: 如果需要进行投影匹配的点的数目大于0, 就进行投影匹配, 增加更多的匹配关系
2 if(nToMatch>0)
3 {
4     ORBmatcher matcher(0.8);
5     int th = 1;
6     if(mSensor==System::RGBD)    //RGBD相机输入的时候, 搜索的阈值会变得稍微大一些
7         th=3;
8     if(mpAtlas->isImuInitialized())
9     {
10         if(mpAtlas->GetCurrentMap()->GetInertialBA2())
11             th=2;
12         else
13             th=3;
14     }
15     else if(!mpAtlas->isImuInitialized() && (mSensor==System::IMU_MONOCULAR || m
16     {
17         th=10;
18     }
19
20     // If the camera has been relocalised recently, perform a coarser search
21     // 如果不久前进行过重定位, 那么进行一个更加宽泛的搜索, 阈值需要增大
22     if(mCurrentFrame.mnId<mnLastRelocFrameId+2)
23         th=5;
24
25     if(mState==LOST || mState==RECENTLY_LOST) // Lost for less than 1 second
26         th=15;
27     // 投影匹配得到更多的匹配关系
28     int matches = matcher.SearchByProjection(mCurrentFrame, mvpLocalMapPoints, t
29 }
```

## 标定问题

检查内参、外参是否正确，如果是RGB-D模式检查彩色图和深度图是否对齐。内参是否设置的是彩色相机的内参，深度缩放系数是否正确。

```
Examples > RGB-D > ! TUM1.yaml
 4  # Camera Parameters. Adjust them!
 5  #-----
 6  Camera.type: "PinHole"
 7
 8  # Camera calibration and distortion parameters (OpenCV
 9  Camera.fx: 517.306408
10  Camera.fy: 516.469215
11  Camera.cx: 318.643040
12  Camera.cy: 255.313989
13
14  Camera.k1: 0.262383
15  Camera.k2: -0.953104
16  Camera.p1: -0.005358
17  Camera.p2: 0.002628
18  Camera.k3: 1.163314
19
20  Camera.width: 640
21  Camera.height: 480
22
23  # Camera frames per second
24  Camera.fps: 30.0
25
26  # IR projector baseline times fx (approx.)
27  Camera.bf: 40.0
28
29  # Color order of the images (0: BGR, 1: RGB. It is
30  Camera.RGB: 1
31
32  # Close/Far threshold. Baseline times.
33  ThDepth: 40.0
34
35  # Depthmap values factor
36  DepthMapFactor: 5000.0 # 1.0 for ROS_bag
37
```

## 分析跟丢原因

显示当前帧提取的原始特征点、成功跟踪的特征点，输出中间变量查看

经验阈值设置，比如分辨率最好调整到640X480附近，记得内参也要对应改。因为ORB-SLAM2/3中很多都是经验参数，比如期望提取的特征点数目、图像金字塔层数、跟踪成功判断阈值条件、关键帧设置条件等。有时候根据自己的应用场景做适当修改也可以获得不错效果。

## 每次运行结果不同原因

RANSAC的随机性。

```

// Select a minimum set
// 选择最小的数据样本集，使用八点法求，所以这里就循环了八次
for(size_t j=0; j<8; j++)
{
    // 随机产生一对点的id，范围从0到N-1
    int randi = DUtills::Random::RandomInt(0, vAvailableIndices.size()-1);
    // idx表示哪一个索引对应的特征点对被选中
    int idx = vAvailableIndices[randi];

    // 将本次迭代这个选中的第j个特征点对的索引添加到mvSets中
    mvSets[it][j] = idx;

    // 由于这对点在本次迭代中已经被使用了，所以我们为了避免再次抽到这个点，就在“点的可选
    // 将这个点原来所在的位置用vector最后一个元素的信息覆盖，并且删除尾部的元素
    // 这样就相当于将这个点的信息从“点的可用列表”中直接删除了
    vAvailableIndices[randi] = vAvailableIndices.back();
    vAvailableIndices.pop_back();
}

```

多线程造成的随机性。比如插入关键帧、局部BA的中断等

```

// 相比ORB-SLAM2多了c3,c4
if(((c1a||c1b||c1c) && c2)||c3 ||c4)
{
    // If the mapping accepts keyframes, insert keyframe.
    // Otherwise send a signal to interrupt BA
    // Step 7.6: local mapping空闲时可以直接插入，不空闲的时候要根据情况插入
    if(bLocalMappingIdle)
    {
        // 可以插入关键帧
        return true;
    }
    else
    {
        mpLocalMapper->InterruptBA();
        if(mSensor!=System::MONOCULAR && mSensor!=System::IMU_MONOCULAR)
        {
            // 双目或双目+IMU或RGB-D模式下，如队列里没有阻塞太多关键帧，可以插入
            // tracking插入关键帧不是直接插入，而且先插入到mlNewKeyFrames中，
            // 然后localmapper再逐个pop出来插入到mspKeyFrames
            if(mpLocalMapper->KeyframesInQueue()<3)
                // 队列中的关键帧数目不是很多，可以插入
                return true;
            else
                // 队列中缓冲的关键帧数目太多，暂时不能插入
                return false;
        }
    }
}

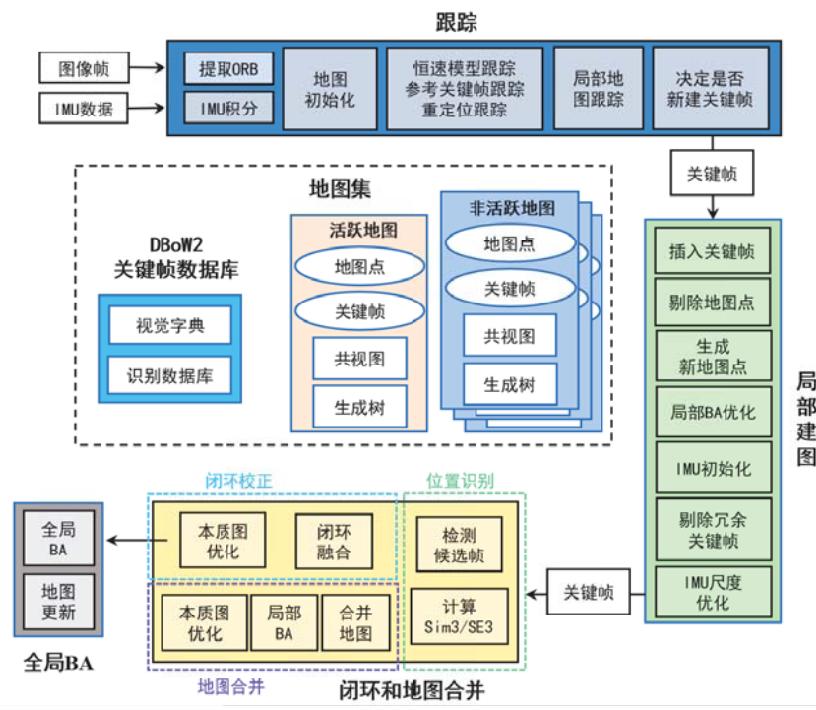
```

## 第7章：局部建图线程

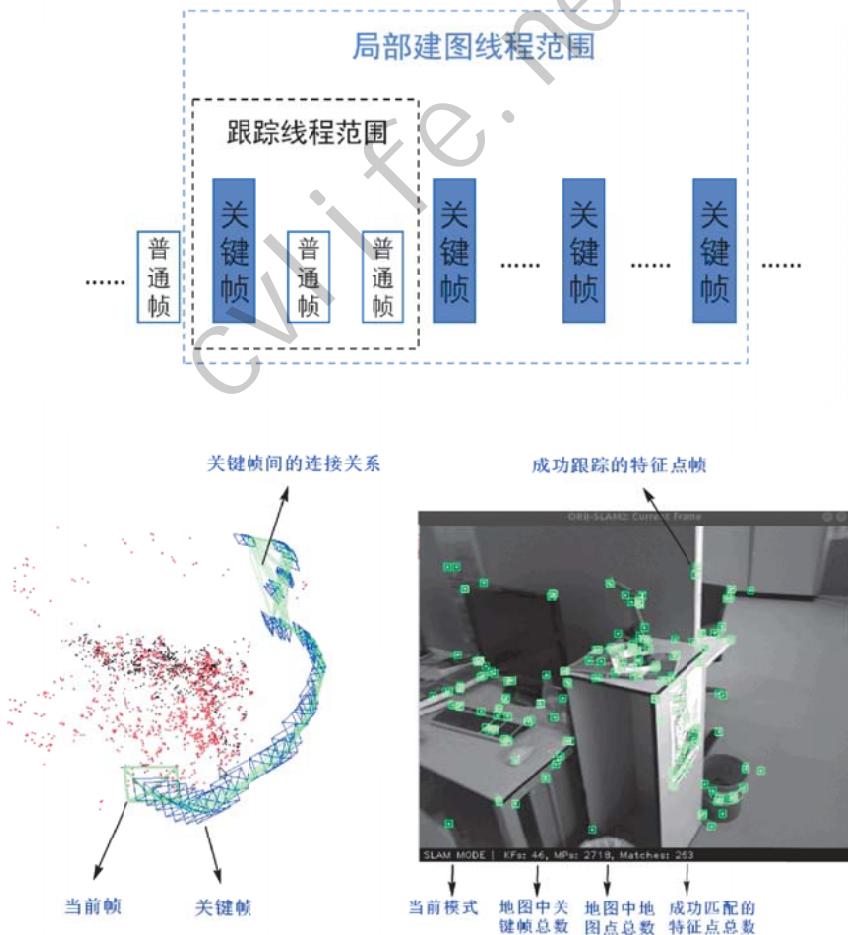
### 局部建图线程的目的和意义

目的和意义：

- 承上启下。接收跟踪线程输入的关键帧并进行局部地图优化、删除冗余关键帧等；将优化后的关键帧发送给闭环线程。



- 实现中期数据关联。跟踪线程中仅使用了相邻普通帧或关键帧的信息，而且只优化当前帧的位姿（没有联合优化多个位姿，没有优化地图点）；局部建图线程里优化满足一定共视关系的多个关键帧及其对应的地图点。使得关键帧的位姿和地图点更加准确。



- 利用共视关键帧之间重新匹配得到更多新的地图点，增加地图里地图点的数目，可以提高跟踪的稳定性。
- 删除冗余关键帧可以降低局部BA的规模和次数。提高实时性。
- 完成IMU的初始化（==非常重要==）。得到比较准确的IMU参数、重力方向、尺度（单目模式）

# 局部建图线程的流程框架

## 局部建图线程中IMU初始化的三个阶段

```
1 //第1阶段初始化: isImuInitialized()
2 //第2阶段初始化: GetInertialBA1()
3 //第3阶段初始化: GetInertialBA2()
4
5 // 局部建图线程中和IMU有关的优化
6 while(1)
7 {
8     if /* IMU成功完成第一阶段初始化 */
9     {
10        // 局部地图+IMU一起优化，优化关键帧位姿、地图点、IMU参数
11        LocalInertialBA();
12    }
13 else
14 {
15    //局部地图BA，不包括IMU信息。优化关键帧位姿、地图点
16    LocalBundleAdjustment();
17 }
18 if /* IMU未完成第一阶段初始化 */
19 {
20     // 执行IMU第一阶段初始化
21     // 目的：快速初始化IMU，尽快用IMU来跟踪
22     InitializeIMU();
23 }
24 else if /* IMU已完成第一阶段初始化 并且 累计时间>5s */
25 {
26     // 执行IMU第二阶段初始化
27     // 目的：快速修正IMU，短时间内使得IMU参数相对靠谱
28     InitializeIMU();
29 }
30 else if /* IMU已完成第二阶段初始化 并且 累计时间>15s */
31 {
32     // 执行IMU第三阶段初始化
33     // 目的：再次优化IMU，保证IMU参数高精度
34     InitializeIMU();
35 }
36 if /* 单目惯性模式 并且 关键帧数目<100 并且 满足一定时间间隔 */
37 {
38     // 优化重力方向和尺度
39     InertialOptimization();
40 }
41 }
```

标记说明：

```
1 bool mbMonocular; //mSensor==MONOCULAR || mSensor==IMU_MONOCULAR
2
3 bool mbInertial; //mSensor==IMU_MONOCULAR || mSensor==IMU_STEREO
4
5 bool IsInitializing; //跟踪线程使用，如果为true，暂不添加关键帧
6
7 Tracking::NeedNewKeyFrame()
8 {
```

```

9     if (mpLocalMapper->IsInitializing())
10    return false;
11 }
12 void Tracking::CreateNewKeyFrame()
13 {
14     if(mpLocalMapper->IsInitializing())
15         return;
16 }
```

## ORB-SLAM2/3局部建图线程流程对比

### ORB-SLAM2中局部建图线程流程

```

1 // ORB-SLAM2中local mapping
2 while(1)
3 {
4     SetAcceptKeyFrames(false);
5     // 等待处理的关键帧列表不为空
6     if(CheckNewKeyFrames())
7     {
8         // 处理列表中的关键帧，包括计算Bow、更新观测、描述子、共视图，插入到地图等
9         ProcessNewKeyFrame();
10        // 根据地图点的观测情况剔除质量不好的地图点
11        MapPointCulling();
12        // 当前关键帧与相邻关键帧通过三角化产生新的地图点，使得跟踪更稳
13        CreateNewMapPoints();
14        // 已经处理完队列中的最后的一个关键帧
15        if(!CheckNewKeyFrames())
16        {
17            // 检查并融合当前关键帧与相邻关键帧（两级相邻）中重复的地图点
18            SearchInNeighbors();
19        }
20        // 已经处理完队列中的最后的一个关键帧，并且闭环检测没有请求停止LocalMapping
21        if(!CheckNewKeyFrames() && !stopRequested())
22        {
23            if(mpMap->KeyFramesInMap(>2)
24                Optimizer::LocalBundleAdjustment(mpCurrentKeyFrame,&mbAbortBA, m
25                // 检测并剔除当前帧相邻的关键帧中冗余的关键帧
26                KeyFrameCulling();
27            }
28            // 将当前帧加入到闭环检测队列中
29            mpLoopCloser->InsertKeyFrame(mpCurrentKeyFrame);
30        }
31        SetAcceptKeyFrames(true);
32 }
```

### ORB-SLAM3中局部建图线程流程

```

1 // ORB-SLAM3中local mapping
2 while(1)
3 {
4     SetAcceptKeyFrames(false);
5     // 等待处理的关键帧列表不为空，并且imu正常
6     if(CheckNewKeyFrames()&& !mbBadImu)
7     {
8         // 处理列表中的关键帧，包括计算Bow、更新观测、描述子、共视图，插入到地图等
```

```

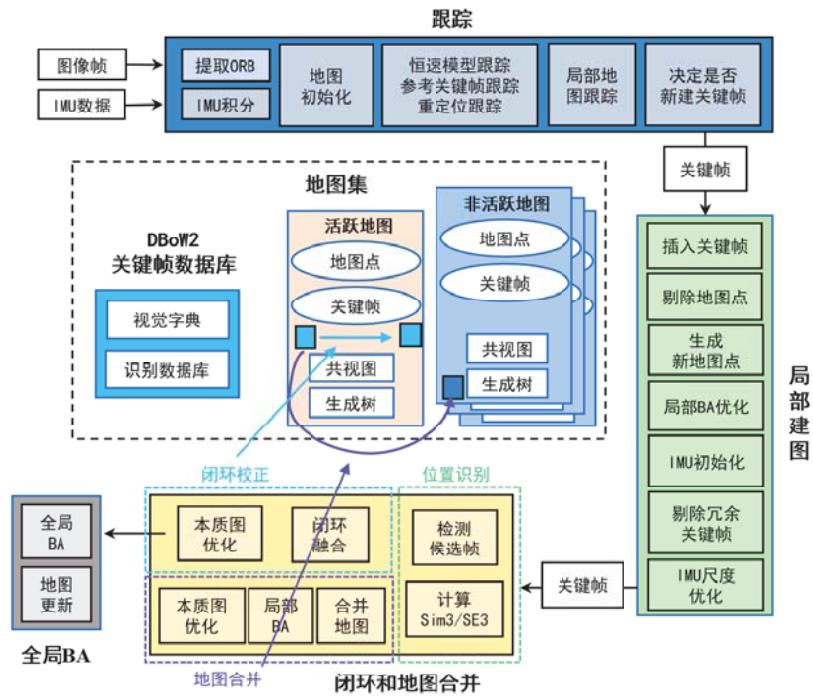
9     ProcessNewKeyFrame();
10    // 根据地图点的观测情况剔除质量不好的地图点
11    MapPointCulling();
12    // 当前关键帧与相邻关键帧通过三角化产生新的地图点，使得跟踪更稳
13    CreateNewMapPoints();
14    // 已经处理完队列中的最后的一个关键帧
15    if(!CheckNewKeyFrames())
16    {
17        // 检查并融合当前关键帧与相邻关键帧帧（两级相邻）中重复的地图点
18        SearchInNeighbors();
19    }
20    // 已经处理完队列中的最后的一个关键帧，并且闭环检测没有请求停止LocalMapping
21    if(!CheckNewKeyFrames() && !stopRequested())
22    {
23        // 当前地图中关键帧数目大于2个
24        if(mpAtlas->KeyFramesInMap()>2)
25            if /* IMU成功完成第一阶段初始化 */
26                Optimizer::LocalInertialBA(); //局部地图+惯性BA
27            else
28                Optimizer::LocalBundleAdjustment(); //局部地图BA
29        if /* IMU未完成第一阶段初始化 */
30            InitializeIMU(); // 执行IMU第一阶段初始化。目的：快速初始化IMU，尽快用
31            // 检测并剔除当前帧相邻的关键帧中冗余的关键帧
32            KeyFrameCulling();
33            // 如果距离IMU第一阶段初始化成功累计时间差小于100s，进行VIBA
34            if ((mTinit<100.0f) && mbInertial)
35                if /* IMU已完成第一阶段初始化 并且 正常跟踪 */
36                    if /* IMU未完成第二阶段初始化 并且 累计时间>5s */
37                        InitializeIMU(); // 执行IMU第二阶段初始化。目的：快速修正IMU
38                    else if /* IMU未完成第三阶段初始化 并且 累计时间>15s */
39                        InitializeIMU(); // 执行IMU第三阶段初始化。目的：再次优化IMU
40                    if /* 单固惯性模式 并且 关键帧数目<100 并且 满足一定时间
41                        ScaleRefinement(); //优化重力方向和尺度
42                }
43                // 将当前帧加入到闭环检测队列中
44                mpLoopCloser->InsertKeyFrame(mpCurrentKeyFrame);
45            }
46            SetAcceptKeyFrames(true);
47    }

```

## 第8章：多地图系统、地图保存和加载

### 多地图基本概念

多地图（Atlas）系统由一系列不连续的子地图（map）构成，并建立了一个唯一的基于DBOW2的关键帧数据库，子地图之间能够无缝连接，实现重定位、回环检测、位置识别等功能。这些子地图分为两种：活跃的地图和不活跃的地图。



### 什么是活跃的地图？

在活跃地图中，当采集到一个新的图像帧时，跟踪线程立即追踪并定位该帧的位姿，如果被选为关键帧，它会在局部建图线程里其他共视关键帧一起不断优化。同一时间只有一个活跃地图。

### 什么是不活跃的地图？

在多地图系统中，除当前活跃地图外的其他子地图都会被标记为不活跃地图。

### 活跃地图和不活跃地图如何转化？

当相机跟踪失败（跟踪中的LOST）就开始进行重定位，如果过了一段时间重定位仍然失败，则将活跃地图标记为不活跃地图，储存在地图集中。然后，重新初始化并启动一个新地图。

### 如何区别闭环和地图融合？

如果检测到公共地图区域的关键帧都来自当前的活跃地图，那么执行闭环操作；如果来自不同的子地图，则执行地图融合。如果同时检测到闭环和融合，执行融合操作，忽略闭环。

以下是共同区域检测的部分代码。

```

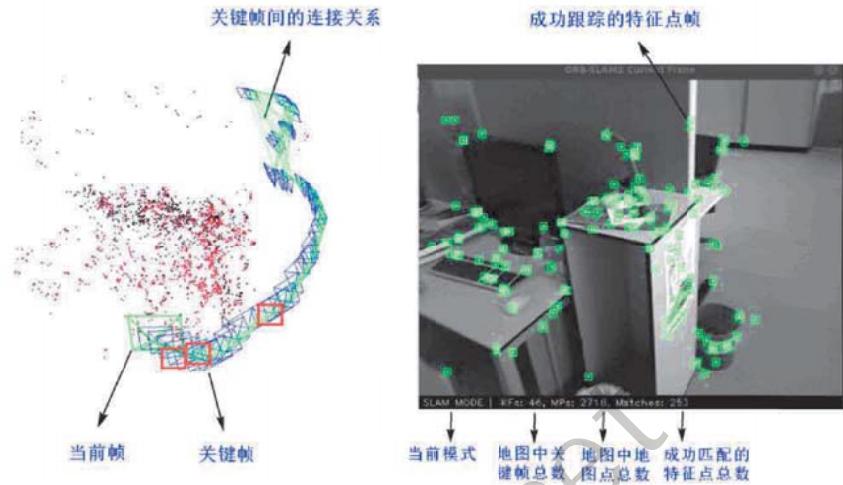
1 // KeyFrameDatabase.cc
2 // KeyFrameDatabase::DetectNBestCandidates
3
4 // 如果候选帧pKFi与当前关键帧pKF在同一个地图里,且候选者数量还不够 (nNumCandidates=3)
5 if(pKF->GetMap() == pKFi->GetMap() && vpLoopCand.size() < nNumCandidates)
6 {
7     // 添加到回环候选帧里
8     vpLoopCand.push_back(pKFi);
9 }
10 // 如果候选者与当前关键帧不在同一个地图里, 且候选者数量还不够, 且候选者所在地图不是bad
11 else if(pKF->GetMap() != pKFi->GetMap() && vpMergeCand.size() < nNumCandidates &
12 {
13     // 添加到融合候选帧里
14     vpMergeCand.push_back(pKFi);
15 }
```

### 宽基线

基线的本意是指立体视觉系统中两摄像机光心之间的距离。



**宽基线**一词用于匹配时，泛指两幅图像有明显不同的情况下的匹配。依据拍摄两幅图像的视点位置关系可将对应点匹配问题分为宽基线(Wide Baseline)和窄基线匹配(Short Baseline)。产生这种情况的原因有可能是摄像机之间的位置相差很大，也有可能由于摄像机旋转或焦距的变化等因素产生的。



宽基线匹配和窄基线匹配的分界不是很严格，但是在窄基线匹配中存在如下假设：摄像机焦距及其它内参数变化不大：摄像机位置不会相差很远，不会有大的转动，对应点的邻域是相似的。

宽基线匹配中则存在如下假设：对图像上的任意点，在另一图像上的对应点可以为任意位置；摄像机可以任意移动，且摄像机的焦距及其它内参数可以有较大的变化；一幅图像上的景物在另一幅图像上可能被遮挡；对应点的邻域有相似的地方，但由于摄像机位置的变化及光照的变化，单依靠邻域的相似不能得到正确的对应。

窄基线匹配中典型方法是利用邻域的互相关(Neighborhood Cross-Correlation)方法。但在宽基线的情况下，图像之间拍摄距离较远，成像条件存在较大差异，即使是空间同一特征，在图像中所表示出来的光学特性(灰度值，颜色值等)、几何特性(外形，大小等)及空间位置(图像中的位置，方向等)都有很大的不同，再加上噪声、遮挡等因素的存在，此时基于邻域互相关的匹配方法就失效了。在宽基线匹配中，仅仅使用特征本身的信息(比如边缘、角点的位置信息)是难以正确匹配的，研究学者将多个特征尤其是结构性特征予以组合，以形成稳定的特征向量(称为特征描述符)。这种对于图像的几何变形、光照变化等因素保持一定稳定性的特征向量称为不变量Invariant)。不变量技术是宽基线匹配应用中的重要技术。

## 多地图作用和效果

主要作用：

- 能够处理无限数量的非连接的子地图，能够在大场景下进行S。每个子地图有自己的关键帧、地图点、共视图、生成树。每个子地图的参考帧固定为它的第一帧。新采集的图像帧只更新所有地图中的一个子地图，也就是活跃地图。所有子地图共用一个唯一的关键帧词袋数据库，保证了地图场景重识别的高效率。
- ORBSLAM-Atlas将宽基线匹配引入到多地图领域，结果更加通用和鲁棒。
- 如果在探索过程中跟踪丢失了，那么将暂存当前地图为非活跃地图，并启动一个新的子地图。当后续地图之间检测到公共区域的时候可以实现无缝地图融合。

- 在ORB-SLAM2中，相机跟踪丢失的判断标准是简单计算跟踪点数量。在ORB-SLAM3中做了改进，制定了新的跟踪丢失的评判标准。当几何约束不好的时候，我们建议放弃不准确的相机位姿估计。这可以避免在闭环的过程中由于高度不确定的位姿导致的位姿图优化误差过大。最终地图被分割为多个更精确的子地图，有了多地图功能，这些子地图最终融合为更精确的全局地图。
- 在处理动态场景时更加鲁棒。

效果：

ORB-SLAM3多地图系统生成的多会话全局地图是VINS-Mono全局地图精度的2倍。Atlas：指代整个地图；map：指代其中的子地图。

We provide extensive experimental validation in the EuRoC datasets, where ORBSLAM-Atlas obtains accurate monocular and stereo results in the difficult sequences where ORBSLAM failed. We also build global maps after multiple sessions in the same room, obtaining the best results to date, between 2 and 3 times more accurate than competing multi-map approaches. We also show the robustness and capability of our system to deal with dynamic scenes, quantitatively in the EuRoC datasets and qualitatively in a densely populated corridor where camera occlusions and tracking losses are frequent.

We also compare with VINS-Mono [4] in the multi-session processing of the Machine Hall EuRoC datasets. VINS-Mono is a visual odometry system, in which loop correction is estimated by pose graph optimization. As ORBSLAM-Atlas is able to detect and process with BA numerous high parallax observations, their individual maps are 2 times more accurate than those of VINS-Mono. ORBSLAM-Atlas multi-session global map retains the 2 times higher accuracy over the VINS-mono global map, because thanks to the map merging, it is able to detect and take profit from high parallax matches also in the multi-map and multi-session case.

#### A. Multiple map performance

We focus our quantitative evaluation on the EuRoC V1\_03\_difficult and V2\_03\_difficult datasets because ORB-SLAM2 stereo [2] or ORBSLAM monocular [3] reported them as failure due to a coverage below 90 %. Coverage is defined as the fraction of localized frames with respect to the total number of ground truth frames in the dataset. The differences in performance in the rest of the datasets are negligible because ORBSLAM-Atlas never lost track, and hence never used more than a single map.

Table I reports the quantitative comparison, see also Figure 3. We have made new experiments with ORBSLAM to report both the RMS ATE and the coverage. Thanks to the multi-maps, ORBSLAM-Atlas is able to significantly boost the coverage from 10-15 % to 70-90 %, with an RMS ATE lower than ORBSLAM.

In the stereo case, in V1\_3 the differences between ORB-SLAM2 and ORBSLAM-Atlas are negligible. In contrast, in V2\_3 ORBSLAM-Atlas produces 5 intermediate maps that eventually are merged in a global map able to achieve around 95 % coverage and an RMS ATE lower than ORBSLAM2.

	ORBSLAM-Atlas Monocular			ORBSLAM Monocular			ORBSLAM-Atlas Stereo			ORBSLAM2 Stereo		
	ATE (m)	Cover (%)	# Maps	ATE (m)	Cover (%)	# Maps	ATE (m)	Cover (%)	# Maps	ATE (m)	Cover (%)	# Maps
V1_03	<b>0.106</b>	<b>90.74</b>	2	0.132	10.32	1	0.051	100	1	<b>0.046</b>	100	1
V2_03	<b>0.093</b>	<b>70.74</b>	2	0.146	15.71	1	<b>0.218</b>	<b>94.55</b>	5	0.316	89.21	1

TABLE I: Performance on the difficult Vicon Room EuRoC datasets. RMS ATE in meters. Median values after 5 runs.

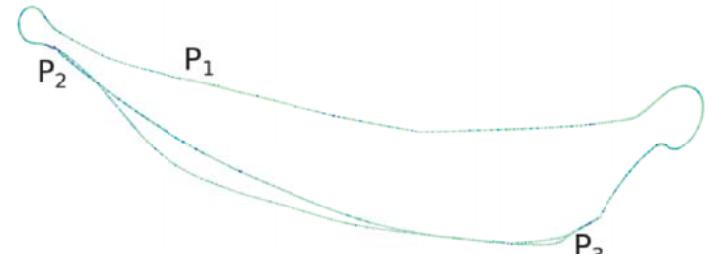
## 创建新地图的标准

当相机跟踪失败就开始进行重定位，如果过了一段时间重定位仍然失败，则将活跃地图标记为不活跃地图，储存在地图集合里。然后初始化一个新地图，评判跟踪丢失的标准：

- 成功匹配的特征点的数量：当前帧和局部地图中成功匹配数量高于定义的阈值。
  - 相机位姿的可观测性：如果检测到的点几何条件不好，那么估计的相机位姿也不准，相机位姿是不可观的。
  - 下图 (a) 中显示的是室外汽车上相机拍摄的图像，图像对应在 (b)、(c) 中的  $P_1$  位置。在  $P_1$  满足成功匹配的特征点数量的要求，但是匹配的绝大多数都是远点（超过40倍基线），因此估计的位姿中平移分量非常不准确（原因见下式），不满足相机位姿可观测性的要求。
  - 假设  $x_1, x_2$  是两个匹配特征点的归一化坐标， $s_1, s_2$  是两个特征点对应的深度（理论上是相等，但因为噪声，不一定相等）。根据三角化原理有
- $$> s_2 x_2 = s_1 R x_1 + t >$$
- 当  $s_1, s_2$  数值比较大时， $t$  的估计很困难，可能淹没在噪声里
- 如果没有可观测性的约束，不精确的位姿会导致闭环矫正误差很大，如下图 (b) 中  $P_2, P_3$  两处分别发生了闭环，但最终位姿图优化后的轨迹误差很大。本来应该是在笔直道路上的一个来回，经过错误的闭环矫正变成了奇怪的轨迹。
  - 当使用了可观测性约束后，下图 (c) 中  $P_1$  位置处矩形内这些位姿不准确的关键帧会被移除，当运行到矩形左边的时候，系统新建了地图，之后到达  $P_2$  处开始位置识别闭环，两个地图融合为一个地图。在  $P_3$  处，检测到闭环并进行矫正，最终得到了一个缺少了部分定位帧但更精确的全局地图。



(a)



(b)



(c)

## 多地图中的重定位

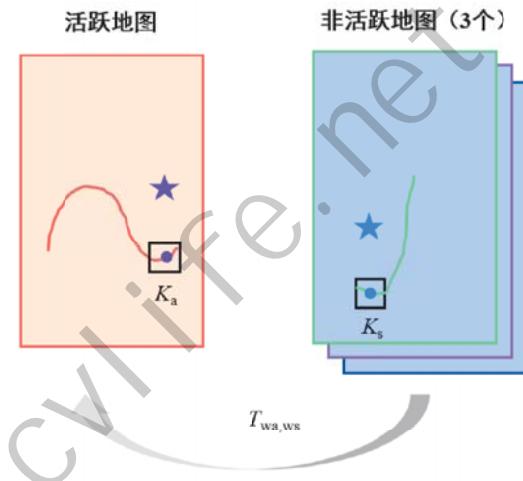
如果相机跟丢了，利用当前帧查询多地图中DBoW的数据库。这个查询能在所有的地图中找到相似的关键帧。一旦有了候选关键帧，地图和匹配的地图点，就可以按照ORB-SLAM2中方式进行重定位。主要包括利用MLPnP进行鲁棒的位姿估计和RANSAC阶段，然后进行引导搜索匹配、仅优化位姿的BA。

## ORB-SLAM3中的地图融合

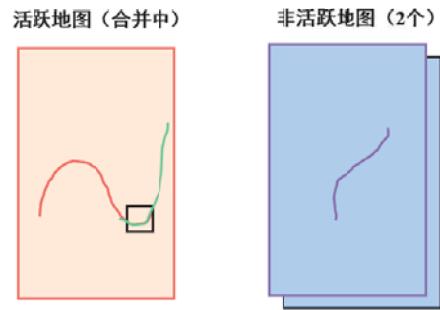
注意：论文《ORBSLAM-Atlas: a robust and accurate multi-map system》中描述和ORB-SLAM3代码中差别较大。建议看代码。

地图融合方面：活跃地图吞并和它有共同区域的子地图，然后用融合完的地图作为新的活跃地图。为方便描述，下面用下标  $a, s, m$  分别代表活跃地图、被吞并的地图、融合后的地图。

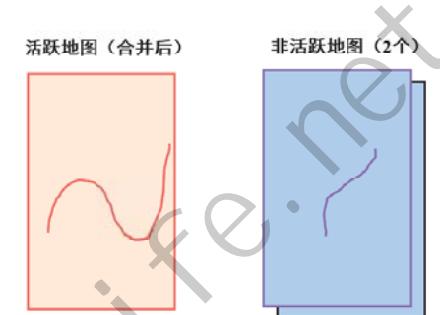
- 1) 在两个地图中检测共同区域。场景重识别模块提供了两个匹配的关键帧  $K_a$  和  $K_s$ ，以及地图  $M_a$  和  $M_s$  之间一系列匹配的地图点。
- 2) 估计用于地图对齐的变换矩阵  $T_{Wa, Ws}$ 。对应双目的  $SE(3)$  或者单目的  $Sim(3)$  变换，这样就可以在世界参考帧中对齐两个待融合的地图。基于  $M_a$  和  $M_s$  两个地图的匹配关系，我们结合了求解  $Sim(3)$  变换和RANSAC的方法来进行初始的估计。然后将估计出来的位姿变换在关键帧  $K_s$  中进行引导匹配，通过非线性优化重投影误差最终得到两个地图之间的变换矩阵  $T_{Wa, Ws}$ 。



- 3) 融合地图。首先利用位姿  $T_{Wa, Ws}$  把  $M_s$  窗口的所有关键帧和地图点都投影到  $M_a$  的窗口内；然后检测重复的地图点并进行融合，用旧的地图点替换新的地图点。
- 4) 融合生成树。由于两个地图的生成树无法粗暴的直接相连，所以需要特殊的方法来将生成树进行融合。
- 5) 在地图焊接区域的welding BA。



- 纯视觉地图合并：优化当前关键帧共视窗口里的所有关键帧和地图点，固定所有融合帧共视窗口里的关键帧。
- 视觉-惯性地图融合。
- 6) 位姿图优化。视觉地图融合中用本质图优化来优化所有的关键帧，根据需要进行全局BA。视觉-惯性地图融合中没有做本质图优化和全局BA。



地图融合线程和跟踪线程、局部建图线程、全局BA线程（会根据需要启动）并行运行。

#### 在地图融合开始之前：

局部建图线程会停止，避免在地图集合中加入新的关键帧。如果全局的BA线程在运行，局部建图也会停止，因为生成树在BA后会发生改变。

#### 地图融合过程中：

为了保证实时性，跟踪线程会在旧的活跃地图中继续运行。

#### 完成地图融合：

重启局部建图线程。如果全局BA停止了，也会重启来处理新的数据。

## 代码解析

### 如何新建地图？

如果当前活跃地图有效，先存储当前地图为不活跃地图，然后新建地图；否则，可以直接新建地图。

```

1 // 在Atlas.cc文件中
2 // 地图集中新建地图
3 void Atlas::CreateNewMap()

```

```

4 {
5     // 锁住地图集
6     unique_lock<mutex> lock(mMutexAtlas);
7     cout << "Creation of new map with id: " << Map::nNextId << endl;
8
9     // 如果当前活跃地图有效，先存储当前地图为不活跃地图后退出
10    if(mpCurrentMap){
11        cout << "Exits current map " << endl;
12        // mnLastInitKFidMap为当前地图创建时第1个关键帧的id，它是在上一个地图最大关键帧
13        if(!mspMaps.empty() && mnLastInitKFidMap < mpCurrentMap->GetMaxKFid())
14            mnLastInitKFidMap = mpCurrentMap->GetMaxKFid()+1; //The init KF is t
15        // 将当前地图储存起来，其实就是把mIsInUse标记为false
16        mpCurrentMap->SetStoredMap();
17        cout << "Saved map with ID: " << mpCurrentMap->GetId() << endl;
18
19        //if(mHasViewer)
20        //    mpViewer->AddMapToCreateThumbnail(mpCurrentMap);
21    }
22    cout << "Creation of new map with last KF id: " << mnLastInitKFidMap << endl
23
24    mpCurrentMap = new Map(mnLastInitKFidMap); //新建地图
25    mpCurrentMap->SetCurrentMap();           //设置为活跃地图
26    mspMaps.insert(mpCurrentMap);           //插入地图集
27 }

```

## 什么时候新建地图？

1、SLAM系统刚启动，新建地图集Atlas类的时候。

```

1 // 在System.cc文件中
2 // 创建地图集Atlas类，参数0表示初始化关键帧id为0
3 mpAtlas = new Atlas(0);
4
5 // 在Atlas.cc文件中
6 // Atlas类的构造函数
7 Atlas::Atlas(int initKFid): mnLastInitKFidMap(initKFid), mHasViewer(false)
8 {
9     mpCurrentMap = static_cast<Map*>(NULL);
10    // 创建新地图
11    CreateNewMap();
12 }

```

2、跟踪线程中时间戳异常的情况下

```

1 // 跟踪线程中时间戳异常情况下的处理
2 if(mState!=NO_IMAGES_YET)
3 {
4     // 进入以下两个if语句都是不正常的情况，不进行跟踪直接返回
5     if(mLastFrame.mTimeStamp>mCurrentFrame.mTimeStamp)
6     {
7         // 如果当前图像时间戳比前一帧图像时间戳小，说明出错了，清除imu数据，创建新的子地图
8         cerr << "ERROR: Frame with a timestamp older than previous frame detected"
9         unique_lock<mutex> lock(mMutexImuQueue);
10        mlQueueImuData.clear();
11        // 创建新地图
12        CreateMapInAtlas();

```

```

13         return;
14     }
15     else if(mCurrentFrame.mTimeStamp>mLastFrame.mTimeStamp+1.0)
16     {
17         // 如果当前图像时间戳和前一帧图像时间戳大于1s, 说明时间戳明显跳变了, 重置地图后直接
18         cout << "id last: " << mLastFrame.mnId << "      id curr: " << mCurrentFrame.mnId
19         //根据是否是imu模式,进行imu的补偿
20         if(mpAtlas->isInertial())
21         {
22             // 如果当前地图imu成功初始化
23             if(mpAtlas->isImuInitialized())
24             {
25                 cout << "Timestamp jump detected. State set to LOST. Resetting IMU"
26                 if(!pCurrentMap->GetInertialBA2())
27                 {
28                     // 如果当前子图中imu没有经过BA2, 重置活跃地图
29                     mpSystem->ResetActiveMap();
30                 }
31             else
32             {
33                 // 如果当前子图中imu进行了BA2, 重新创建新的子地图
34                 CreateMapInAtlas();
35             }
36         }
37     else
38     {
39         // 如果当前子图中imu还没有初始化, 重置活跃地图
40         cout << "Timestamp jump detected, before IMU initialization. Resetting active map"
41         mpSystem->ResetActiveMap();
42     }
43 }
44 // 不跟踪直接返回
45 return;
46 }
47 }
```

### 3、跟踪线程中跟踪丢失后。

如果是第一阶段跟踪丢失，当前活跃地图的处理方法：如果当前活跃地图中关键帧数量小于10个，认为该地图中有效信息太少，直接重置，丢弃当前地图；否则，该地图仍有一定价值，储存起来并新建一个地图。

如果到第二阶段跟踪丢失，当前活跃地图的处理方法：如果当前是纯视觉模式且地图中关键帧超过5个或者IMU模式下已经完成IMU初始化，认为该地图仍有一定价值，储存起来并新建一个地图；否则重置，丢弃当前地图。

```

1 // 第一阶段确定跟踪丢失后
2 if (mState == LOST)
3 {
4     // 开启一个新地图
5     Verbose::PrintMessage("A new map is started...", Verbose::VERBOSITY_NORMAL);
6
7     if (pCurrentMap->KeyFramesInMap()<10)
8     {
9         // 当前地图中关键帧数目小于10, 重置当前地图
10        mpSystem->ResetActiveMap();
11        cout << "Resetting current map..." << endl;
12    }else
```

```

13     // 当前地图中关键帧数目超过10, 先储存当前活跃地图为不活跃地图, 然后创建新地图
14     CreateMapInAtlas();
15     // 清空上一个关键帧
16     if(mpLastKeyFrame)
17         mpLastKeyFrame = static_cast<KeyFrame*>(NULL);
18     Verbose::PrintMess("done", Verbose::VERBOSITY_NORMAL);
19
20     return;
21 }
22
23 // ...
24
25 // 第二阶段确定跟踪失败
26 if(mState==LOST)
27 {
28     // 如果地图中关键帧小于5, 重置当前地图, 退出当前跟踪
29     if(pCurrentMap->KeyFramesInMap()<=5)
30     {
31         mpSystem->ResetActiveMap();
32         return;
33     }
34     if ((mSensor == System::IMU_MONOCULAR) || (mSensor == System::IMU_STEREO))
35         if (!pCurrentMap->isImuInitialized())
36         {
37             // 如果是IMU模式并且还未进行IMU初始化, 重置当前地图, 退出当前跟踪
38             Verbose::PrintMess("Track lost before IMU initialisation, resetting..");
39             mpSystem->ResetActiveMap();
40             return;
41         }
42     // 如果地图中关键帧超过5 并且 纯视觉模式 或 虽然是IMU模式但是已经完成IMU初始化了, 创建新地图
43     CreateMapInAtlas();
44 }

```

## 地图保存和加载

### boost序列化示例

#### 1. CMakeLists.txt

```

1 cmake_minimum_required(VERSION 2.8)
2 project(boost_serialization_test)
3
4 # debug or release
5 SET(CMAKE_BUILD_TYPE Debug)
6 # IF(NOT CMAKE_BUILD_TYPE)
7 #     SET(CMAKE_BUILD_TYPE Release)
8 # ENDIF()
9
10 # 引用c11
11 include(CheckCXXCompilerFlag)
12 CHECK_CXX_COMPILER_FLAG("-std=c++11" COMPILER_SUPPORTS_CXX11)
13 CHECK_CXX_COMPILER_FLAG("-std=c++0x" COMPILER_SUPPORTS_CXX0X)
14 if(COMPILER_SUPPORTS_CXX11)
15     set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
16     add_definitions(-DCOMPILEDWITHC11)
17     message(STATUS "Using flag -std=c++11.")
18 elseif(COMPILER_SUPPORTS_CXX0X)

```

```

19     set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++0x")
20     add_definitions(-DCOMPILEDWITHC0X)
21     message(STATUS "Using flag -std=c++0x.")
22 else()
23     message(FATAL_ERROR "The compiler ${CMAKE_CXX_COMPILER} has no C++11 support.
24 endif()
25
26 # 编译成可执行文件
27 add_executable(boost_serialization_example
28     boost_serialization_example.cpp)
29 target_link_libraries(boost_serialization_example -lboost_serialization)

```

## 2. 数据保存与读取的代码

```

1 #include <boost/archive/text_iarchive.hpp>
2 #include <boost/archive/text_oarchive.hpp>
3 #include <boost/archive/binary_iarchive.hpp>
4 #include <boost/archive/binary_oarchive.hpp>
5
6 // #include <boost/serialization/base_object.hpp>
7 #include <boost/serialization/vector.hpp>
8 #include <boost/serialization/map.hpp>
9 #include <boost/serialization/set.hpp>
10
11 #include <iostream>
12 #include <string>
13 #include <fstream>
14
15 class A
16 {
17 public:
18     friend class boost::serialization::access;
19     template<class Archive>
20     void serialize(Archive& ar, const unsigned int version)
21     {
22         ar & num;
23         ar & text;
24         ar & data;
25     }
26
27
28     int num;
29     std::string text;
30     std::vector<double> data;
31 };
32
33 class B
34 {
35 public:
36     friend class boost::serialization::access;
37     template<class Archive>
38     void serialize(Archive& ar, const unsigned int version)
39     {
40         ar & a;
41         ar & map_data;
42         ar & set_data;
43     }

```

```

44
45     A a;
46     std::map<std::string, int> map_data;
47     std::set<float> set_data;
48 };
49
50 // 保存
51 void Save(const std::string & save_file_name, const B & b)
52 {
53     std::ofstream ofs(save_file_name, std::ios::binary);
54     boost::archive::binary_oarchive oa(ofs);
55     // boost::archive::text_oarchive oa(ofs);
56     oa << save_file_name;
57     oa << b;
58 }
59
60 // 读取
61 void Read(const std::string & save_file_name, B & b)
62 {
63     std::string text_data;
64     std::ifstream ifs(save_file_name, std::ios::binary);
65     boost::archive::binary_iarchive oa(ifs);
66     // boost::archive::text_iarchive oa(ifs);
67     oa >> b;
68     oa >> text_data;
69     std::cout << "读取的文本: " << text_data << std::endl;
70 }
71
72 int main()
73 {
74     B b;
75     b.map_data["key1"] = 1;
76     b.set_data.insert(0.1);
77
78     A a;
79     a.data = {1, 2, 3, 4, 5};
80     a.num = 110;
81     a.text = "hello world";
82
83     b.a = a;
84
85     // 保存
86     std::string save_file_name = "save_file.map";
87     Save(save_file_name, b);
88
89     // 读取
90     B b_read;
91     Read(save_file_name, b_read);
92     std::cout << b_read.a.num << std::endl;
93     std::cout << b_read.a.text << std::endl;
94
95     return 0;
96 }

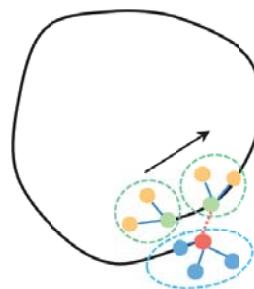
```

## 第9章 闭环及地图融合线程

### 闭环和地图融合的作用

- 建立更多的中长期数据关联。寻找闭环/融合候选关键帧、窗口内welding BA、本质图BA、全局BA
- 可以将多个子地图连接成一个精确的全局地图
- 极大的降低整体的位姿和地图点误差，从而获得全局一致的地图和准确的位姿估计

### 闭环和地图合并示意图



- 当前关键帧 pKF
- 当前关键帧的连接关键帧 spConnectedKF
- 闭环候选关键帧（不和当前关键帧连接，但和它有公共单词） lkFsSharingWords
- 闭环候选关键帧的连接关键帧 vpNeighs

### ORB-SLAM2/3 闭环流程对比

```

1 // ORB-SLAM2、ORB-SLAM3的闭环流程对比(为了保证结构清晰，省略了部分代码)
2
3 // ORB-SLAM2的闭环流程
4 while(1)
5 {
6     // 检查是否有新关键帧
7     if(CheckNewKeyFrames())
8     {
9         // 检测闭环候选关键帧和共视连续性
10        if(DetectLoop())
11        {
12            // 计算Sim3变换[sR/t]，在双目/RGBD模式下s=1
13            if(ComputeSim3())
14            {
15                // 闭环矫正融合及位姿图优化
16                CorrectLoop();
17            }
18        }
19    }
20 }
21
22
23 // ORB-SLAM3的闭环流程
24 while(1)
25 {
26     // 检查是否有新关键帧
27     if(CheckNewKeyFrames())
28     {
29         // 如果检测到公共区域
30         if(NewDetectCommonRegions())
31         {
32             // 如果公共区域发生在当前帧和非活跃地图，执行地图融合
33             if(mbMergeDetected)

```

```

34     {
35         if /* 视觉+IMU模式 */
36     {
37             // 视觉+IMU地图融合及优化
38             MergeLocal2();
39         }
40     else /* 纯视觉模式 */
41     {
42         // 视觉地图融合及优化
43         MergeLocal();
44     }
45 }
46 }
47 // 如果公共区域发生在当前帧和活跃地图, 执行闭环
48 if(mbLoopDetected)
49 {
50     // 闭环矫正及位姿图优化
51     CorrectLoop();
52 }
53 }
54 }
55 }

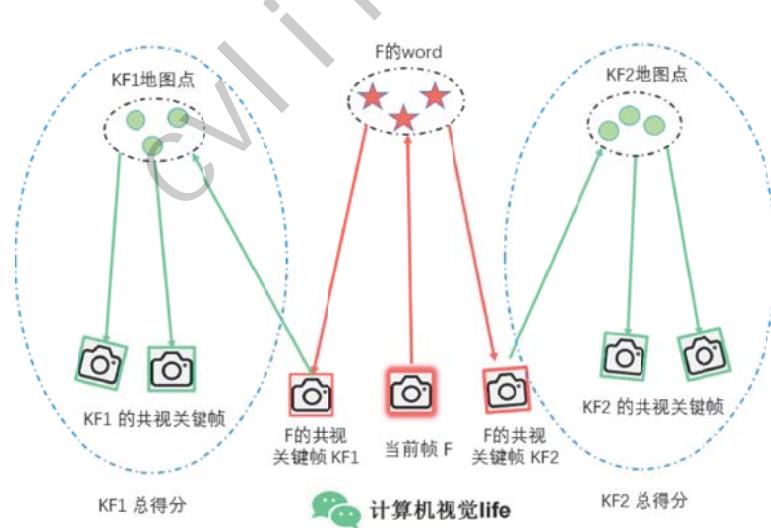
```

## 回顾ORB-SLAM2中的闭环过程

在ORB-SLAM2中只有闭环，没有地图融合。流程如下：

### 第1步：检测闭环DetectLoop()

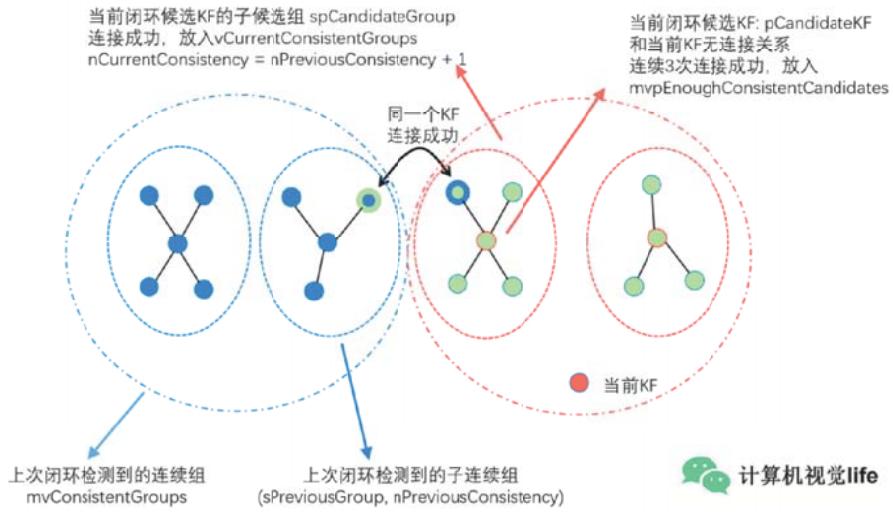
- 寻找初始的（多个）闭环候选帧（注意不和当前帧连接）。



- 在候选帧中检测具有连续性的候选帧。

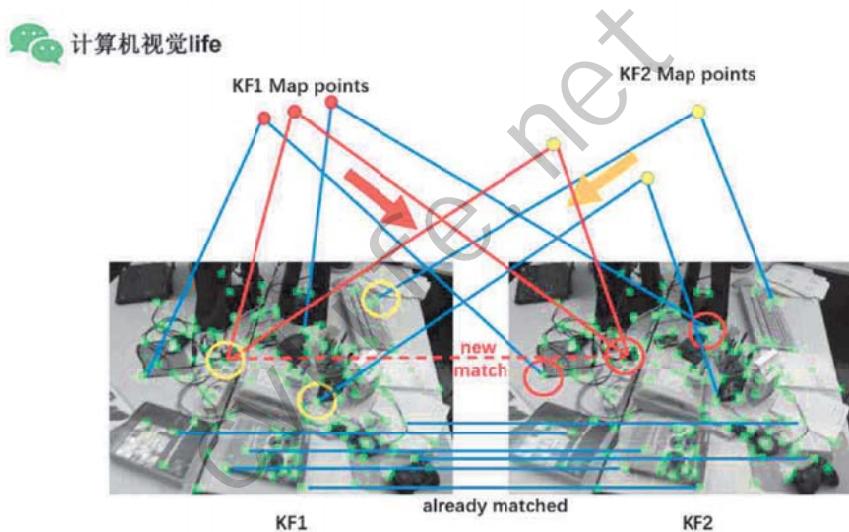
依据：时序上连续进来的三个关键帧的回环候选帧的共视窗口应该会有重叠

缺点：延迟+低Recall



## 第2步：计算Sim3 ComputeSim3()

1. 为每一个闭环候选帧构造一个Sim3Solver，利用词袋将当前关键帧CurrKF和闭环候选关键帧LoopKF进行迭代匹配，得到初始的Sim3变换。
2. 基于初始的Sim3变换，双向搜索匹配两个关键帧中更多的匹配点对，只有在两次互相匹配中都存在才能够认为是可靠的匹配。



3. 用以上可靠的匹配关系进行Sim3优化，得到比较准确的Sim3变换。
4. 将闭环关键帧及其连接关键帧的所有地图点用上述比较准确的Sim3位姿投影到当前关键帧进行投影匹配，希望得到更多的匹配关系。

## 第3步：闭环矫正CorrectLoop()

1. 通过求解的Sim3位姿进行传播，调整与当前帧相连的关键帧位姿以及它们观测到的地图点坐标。
2. 将闭环相连关键帧组依次投影到调整过的当前关键帧组中中的每个关键帧，进行匹配，融合，新增或替换当前关键帧中的地图点。
3. 进行本质图优化，优化本质图中所有关键帧的位姿和地图点。
4. 新建一个线程用于全局BA优化。

## ORB-SLAM3的闭环和地图合并

### 第1步：检测共同区域

目的

找出和当前关键帧有闭环关系或融合关系的关键帧

## 检测共同区域和ORB-SLAM2里的闭环检测有何区别？

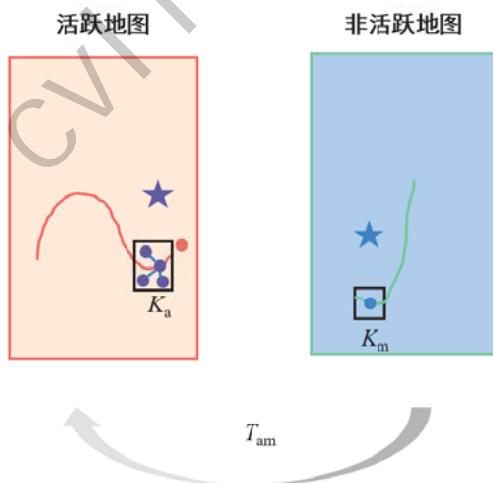
1. 同时检测闭环和融合。
2. 改进的校验方式，可以减少延时。

在ORB-SLAM2中：DBoW2在检测几何连续性之前，需要先检测时间连续性，也就是需要在相同的位置连续成功匹配上三个关键帧。这种方法牺牲了召回率来提升准确率。结果就是，该方法在闭环过程及重用之前地图方面，响应速度过于缓慢。

在ORB-SLAM3里：

- **High-recall place recognition.** Many recent visual SLAM and VO systems [2], [7], [8] solve place recognition using the DBoW2 bag of words library [9]. DBoW2 requires *temporal consistency*, matching three consecutive keyframes to the same area, before checking *geometric consistency*, boosting precision at the expense of recall. As a result, the system is too slow at closing loops and reusing previous maps. We propose a novel place recognition algorithm, in which candidate keyframes are first checked for *geometrical consistency*, and then for *local consistency* with three *covisible keyframes*, that in most occasions are already in the map. This strategy increases recall and densifies data association improving map accuracy, at the expense of a slightly higher computational cost.

该算法首先检查：几何一致性，当前关键帧的5个共视关键帧（已经在地图中）中只要有3个满足条件（和候选关键帧组匹配成功）即可认为检测到共同区域；如果不够3个，则再检查后续新进来关键帧（不在地图中）的时间一致性。这种策略以略高的计算成本为代价，提高了召回率和地图的精度。



### 第1步：寻找候选关键帧并求解位姿变换。

对应函数：LoopClosing::NewDetectCommonRegions()

### 共同区域检测流程

```
1 // 奇葩的代码顺序
2 LoopClosing::NewDetectCommonRegions(){
3     mnLoopNumCoincidences=0;           //成功验证的总次数
4     mnMergeNumCoincidences=0;          //成功验证的总次数
5     bMergeDetectedInKF = false;        //某次时序验证是否成功
6     bLoopDetectedInKF = false;         //某次时序验证是否成功
```

```

7
8
9 // 实际执行顺序 3, 如果2没完成才执行, 2完成任务则不执行3
10 if(mnLoopNumCoincidences > 0){
11     // ...
12     bLoopDetectedInKF = true;           //成功进行一次时序验证
13     mbLoopDetected = mnLoopNumCoincidences >= 3;          // 最终成功验证
14     // ...
15 }
16 if(mnMergeNumCoincidences > 0){
17     // ...
18     bMergeDetectedInKF = true;           //成功进行一次时序验证
19     mnMergeNumCoincidences++;           //总验证成功次数+1
20     mbMergeDetected = mnMergeNumCoincidences >= 3;          // 最终成功验证
21     // ...
22 }
23
24
25 // 实际执行顺序 1
26 vector<KeyFrame*> vpMergeBowCand, vpLoopBowCand;
27 if(!bMergeDetectedInKF || !bLoopDetectedInKF){
28     DetectNBestCandidates(vpLoopBowCand, vpMergeBowCand);
29 }
30
31 // 实际执行顺序 2
32 if(!bLoopDetectedInKF && !vpLoopBowCand.empty()){
33     // 超过3次几何验证(mnLoopNumCoincidences>=3), 就认为最终验证成功 (mbLoopDetected)
34     mbLoopDetected = DetectCommonRegionsFromBoW(vpLoopBowCand, mnLoopNumCoincidences);
35 }
36 if(!bMergeDetectedInKF && !vpMergeBowCand.empty()){
37     // 超过3次几何验证(mnMergeNumCoincidences>=3), 就认为最终验证成功 (mbMergeDetected)
38     mbMergeDetected = DetectCommonRegionsFromBoW(vpMergeBowCand, mnMergeNumCoincidences);
39 }
40
41 // 实际执行顺序 4, 只要一种验证成功就返回true
42 if(mbMergeDetected || mbLoopDetected){
43     return true;
44 }
45 }

```

1.1 找到和当前关键帧  $K_a$  对应的最佳的3个回环候选帧和融合候选帧，统一称为  $K_m$ 。

### ORB-SLAM2中初始候选关键帧的三阈值筛选

对应函数 KeyFrameDatabase::DetectLoopCandidates()

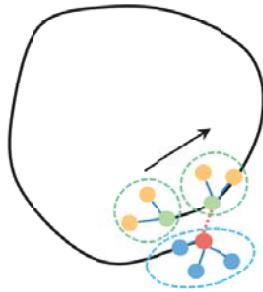
阈值1：minCommonWords，最大共同单词的0.8倍

阈值2：minScore，当前关键帧与它的共视关键帧的最低相似度

阈值3：minScoreToRetain，统计符合上述条件的回环候选帧的共视关系最好的10帧中总相似度最高的组，它的总分的0.75倍

步骤：

1. 找出和当前帧具有公共单词的所有关键帧，不包括与当前帧连接的关键帧
2. 只保留和其中共同单词超过minCommonWords并且相似度超过minScore的关键帧
3. 计算上述候选帧对应的共视关键帧组的总得分，只取最高组得分超过minScoreToRetain的组中分数最高的关键帧作为闭环候选关键帧



- 当前关键帧 pKF
- 当前关键帧的连接关键帧 spConnectedKF
- 闭环候选关键帧（不和当前关键帧连接，但和它有公共单词） IKFsSharingWords
- 闭环候选关键帧的连接关键帧 vpNeighs

### ORB-SLAM3中初始候选关键帧的筛选

对应函数：KeyFrameDatabase::DetectNBestCandidates()

阈值1：minCommonWords，最大共同单词的0.8倍

步骤：

1. 找出和当前帧具有公共单词的所有关键帧，不包括与当前帧连接的关键帧
2. 只保留和其中共同单词超过minCommonWords的关键帧
3. 计算上述候选帧对应的共视关键帧组的总得分，闭环候选关键帧和融合候选帧分别从中取得分最高的前N（代码中N=3）个组中单个分数最高的关键帧（小组MVP）

以下都是在函数LoopClosing::DetectCommonRegionsFromBoW()里

### 1.2 定义局部窗口

对于每个候选帧  $K_m$  我们都定义一个局部窗口  $W_m$ ，窗口内包含了：

- vpCovKFi：候选帧  $K_m$  及其前5个共视关系最好的关键帧
- 把候选关键帧及其共视组的所有地图点记为  $X_m$
- vvpMatchedMPs：通过DBow2找到  $X_m$  和当前关键帧匹配的地图点

### 1.3 计算初始相对位姿变换

不同时期的数据关联	ORB-SLAM2	ORB-SLAM3
Sim3初始值计算	1-1	1-N
基于初始值的Sim3优化	1-1	1-N
Sim3验证	1-N	1-N
welding BA	无	N-N welding BA

构造Sim3Solver，利用RANSAC求解候选关键帧窗口与当前关键帧的初始相对位姿  $T_{am}$ 。

在单目或单目+IMU模式下  $T_{am}$  就是Sim(3)；其他模式下， $T_{am}$  就是SE(3)，后面统一用  $T_{am}$  来表示

### 1.4 Guided Matching refinement

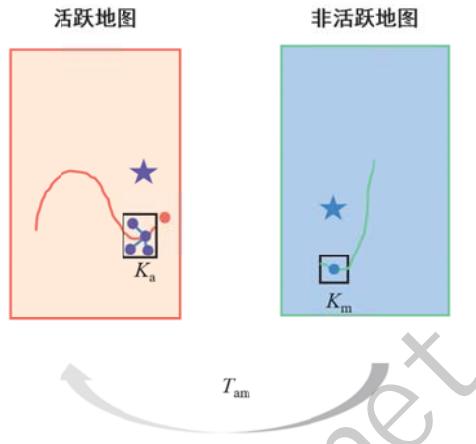
目的：利用上面的位姿初始值，通过投影的方式搜索更多的匹配，并优化位姿  $T_{am}$ 。具体流程：

- 把  $X_m$  通过初始相对位姿  $T_{am}$  转换并投影到当前关键帧  $K_a$  中, 寻找更多的匹配 (searchByProjection)
- 反向搜索把  $K_a$  的地图点也通过  $T_{am}^{-1}$  转换并投影到局部窗口  $W_m$  里的所有关键帧上
- 利用双向搜索到的更多的匹配点非线性优化重投影误差得到更精确的相对位姿  $T_{am}$
- 如果内点超过一定的阈值, 用更严格的匹配点搜索半径和汉明匹配距离, 重新进行上述引导匹配操作。期望得到最高精度的相对位姿  $T_{am}$

## 1.5 关键帧校验

ORB-SLAM3采用的验证模式是集卡式（一堆关键帧里面有3个可以验证的候选关键帧即可）

对应的函数 LoopClosing::DetectCommonRegionsFromLastKF()



### step1：共视几何校验（先进行）

用当前关键帧共视帧来对候选帧进行几何校验的方式：已在地图里，无需等待

- 5个共视关键帧里有3个成功验证，则校验成功
- 大于0个小于3个-->继续进行时序几何校验
- 等于0-->失败, 结束验证

### step2：时序几何校验（step1未成功才进入，用后面新进来的关键帧校验）

用时间上连续进来的新的关键帧对候选帧进行几何校验的方式：还没有在地图里，需要等等

- step1+step2 集齐3个就校验成功
- 连续两个新进来的关键帧时序校验失败，则失败，结束验证

## 1.6 惯性模式下用尺度变化大小验证融合结果，通过约束来保证闭环的准确性

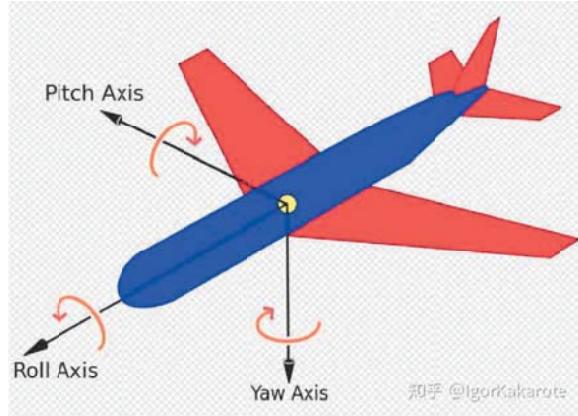
- 验证之前的位姿  $T_{am}$ ，避免假阳
- 如果地图已经成熟, 检查通过位姿  $T_{am}$  转换后 pitch 和 roll 的夹角是否足够小

问题：LoopClosing::DetectCommonRegionsFromBoW()中的参数g2oScw的物理意义？

倒退：g2oScw → g2oBestScw → gScw → gScm\*gSmw → pMostBoWMatchesKF → 候选关键帧

这里的gSmw 里的w 是指 候选关键帧所在的世界坐标系，如果候选关键帧是融合候选关键帧，则位于不同的地图，此处的世界坐标系就是旧（非活跃）地图所在的世界坐标系

滚转角 (roll)、俯仰角 (pitch)、偏航角 (yaw)



## 第2步：地图融合

纯视觉模式下地图融合

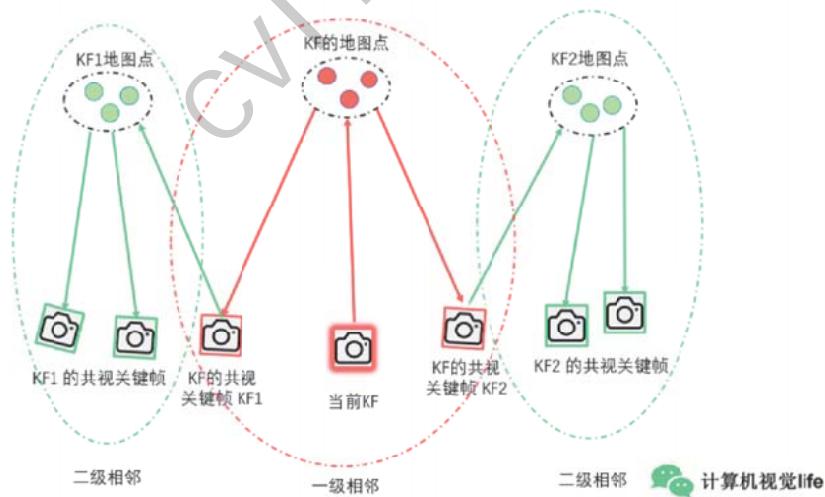
对应函数 LoopClosing::MergeLocal()

特点：

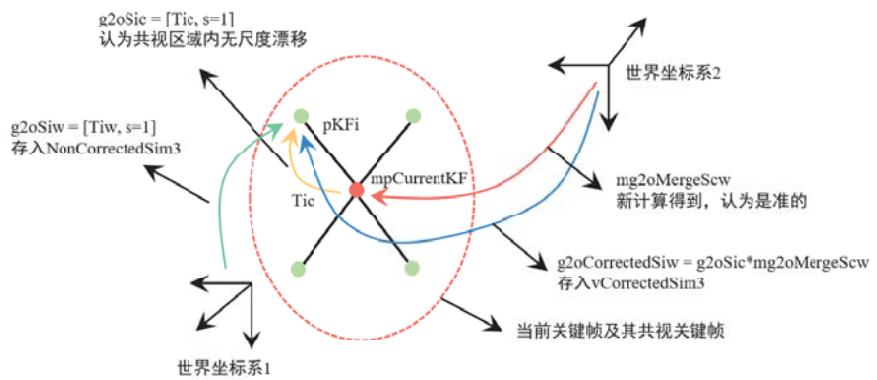
- 融合是在不同的地图中进行操作，而ORB-SLAM2中闭环只有一个地图
- 因为融合的地图可能比较大，为了提高实时性。在进行完局部窗口的welding BA后，就开启了 local mapping线程。之后再进行剩下的地图中（局部窗口外，认为没那么紧急处理）对位姿和地图点进行矫正传播。然后进行本地图优化。
- 多了一个融合生成树的操作。

步骤

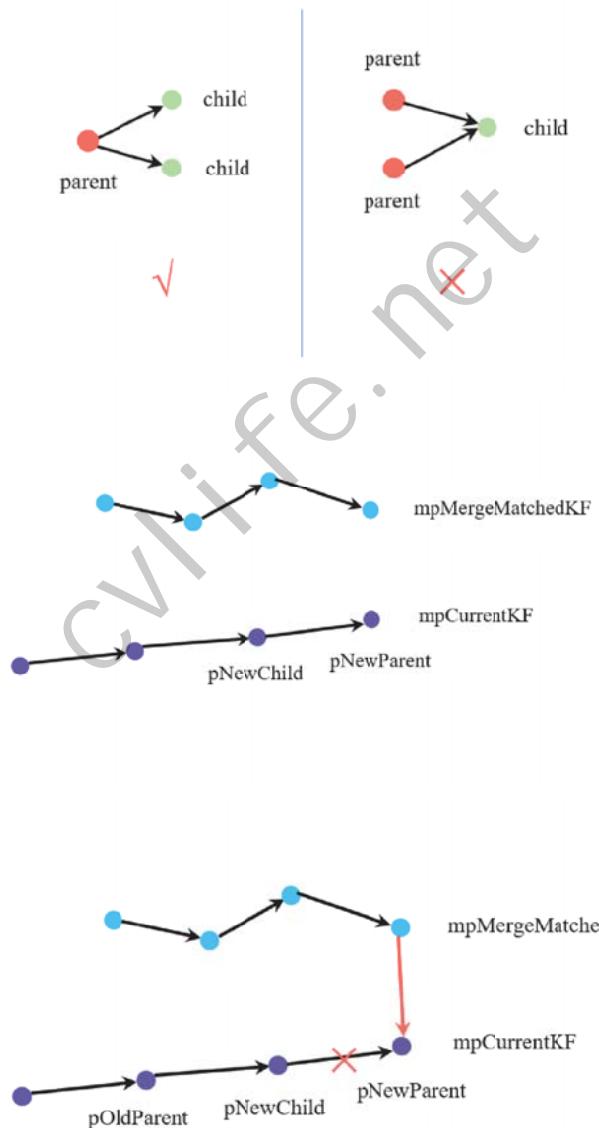
1. 停止全局BA和局部建图线程
2. 构建当前关键帧和融合关键帧的局部窗口，局部窗口包括一级相邻和二级相邻共视关键帧（总共15个），以及它们的地图点。

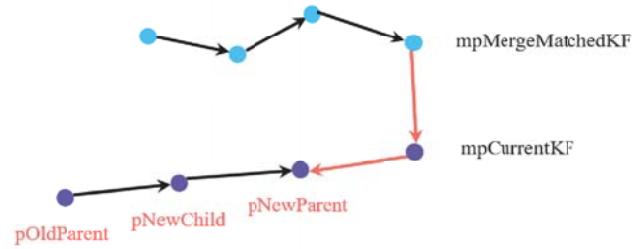


3. 根据之前的Sim3初始值, 记录当前帧窗口内关键帧、地图点的矫正前的值 (vNonCorrectedSim3、eigP3Dw)，和矫正后的初始值 (vCorrectedSim3、cvCorrectedP3Dw)

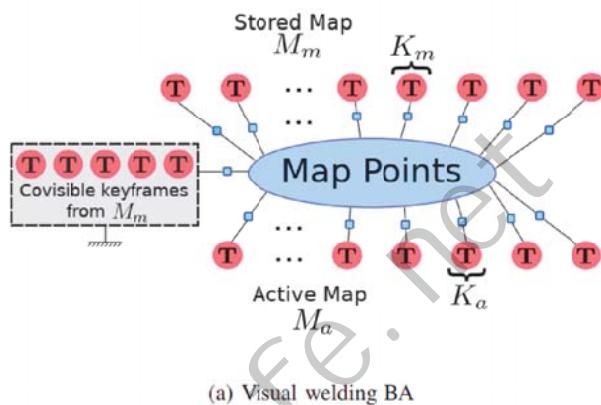


4. 两个地图以新（当前帧所在地图）换旧（融合帧所在地图），包括关键帧及地图点关联地图的以新换旧、地图集的以新换旧。
5. 融合新旧地图的生成树。

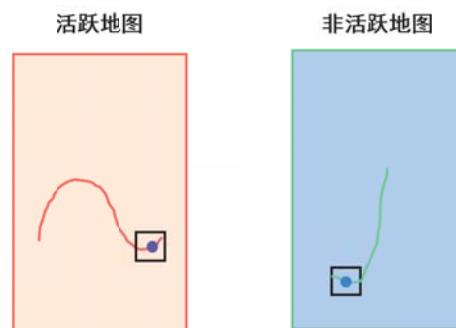




6. 把融合关键帧的共视窗口里的地图点投到当前关键帧的共视窗口里，把重复的点融合掉（以旧换新）
7. 因为融合导致地图点变化。需要更新关键帧中图的连接关系
8. 在缝合(Welding)区域进行local BA（论文中称为Welding BA）。优化当前关键帧共视窗口里的所有关键帧和地图点，固定所有融合帧共视窗口里的关键帧。



9. 在当前帧整个剩下的地图中对位姿和地图点进行矫正传播并进行本质图优化（没那么紧急，可以晚点做）。优化的对象是当前关键帧所在地图里的所有关键帧(除了当前关键帧共视窗口内的关键帧) + 当前地图里的所有地图点，固定不优化的是所有融合帧共视窗口内的关键帧 + 所有当前关键帧共视窗口内的关键帧。



10. 如果需要的话，进行全局BA

地图以新换旧代码

```

1 // LoopClosing.cc
2 // 在Altas中把当前地图休眠，老图重新激活
3 mpAtlas->ChangeMap(pMergeMap);
4 // 当前地图的信息都到融合帧所在地图里去了，可以设置为bad
5 mpAtlas->SetMapBad(pCurrentMap);

```

```

6
7 // Atlas.cc
8 void Atlas::ChangeMap(Map* pMap)
9 {
10     unique_lock<mutex> lock(mMutexAtlas);
11     cout << "Chage to map with id: " << pMap->GetId() << endl;
12     if(mpCurrentMap){
13         mpCurrentMap->SetStoredMap();
14     }
15
16     mpCurrentMap = pMap;
17     mpCurrentMap->SetCurrentMap();
18 }

```

## 惯性模式下地图融合

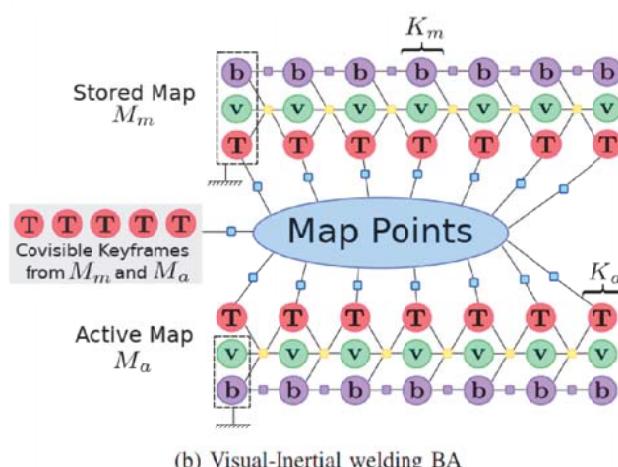
对应函数 LoopClosing::MergeLocal2()

特点：

- 只对缝合区域进行了welding BA。其他地图区域是直接位姿变换过来了，没有类似纯视觉地图融合里的本质图优化或全局优化。
- 如果当前地图IMU没有完全初始化，那么再进行一次IMU快速优化后，强制认为已经完成IMU初始化。
- 地图以旧换新。在纯视觉地图里是以新换旧。

步骤：

1. 停掉正在进行的全局BA、局部建图线程。
2. 利用前面计算的坐标系变换位姿 $g_{Sw2w1}$ ，把整个当前地图（关键帧及地图点）变换到融合帧所在地图。
3. 如果当前地图IMU没有完全初始化，帮助IMU快速优化
4. 地图以旧换新。把融合帧所在地图里的关键帧和地图点从原地图里删掉，变更为当前关键帧所在地图。
5. 融合新旧地图的生成树
6. 把融合关键帧的共视窗口里的地图点投到当前关键帧的共视窗口里，把重复的点融合掉（以旧换新）
7. 针对缝合区域的窗口内进行welding BA。共视关键帧只优化位姿，不优化IMU参数。



## 第3步：闭环矫正

对应函数：CorrectLoop()

和ORB-SLAM2过程一致。

## 第10章 图优化

### 边缘化

SLAM属于渐进式的算法，当前帧的位姿是由上一帧（或之前帧）三角化三维点计算而得，而当前帧的三维点三角化过程又与当前帧的位姿决定，因此会有累计误差的存在。想要消除这种累计误差最直接的方式就是优化时将所有帧一起优化，但是随着帧数增加，计算量也会增加，所以这么做行不通。但是优化如果只优化最近的几帧这样势必会丢失重要数据，这就引出了边缘化的作用。边缘化通过处理信息矩阵的方式，保证只优化一定数量的数据，但又不丢失太多没有参与优化的数据（实现边缘化的经典框架有：VINS与DSO）。ORB-SLAM系列为什么不用边缘化但精度也很高，那是因为有共视图的存在，以当前关键帧为起点，优化与其共视的所有关键帧，基本也不丢失数据。

ORB-SLAM3式下加入了边缘化，与其他VIO框架不同的是，本作的边缘化十分简单，不像其他框架一样考虑三维点，而且滑窗中只保留了一帧。

状态有15维（旋转3，平移3，速度3，陀螺仪偏置3，加速度偏置3）

单帧的状态信息矩阵 $15 \times 15$ ，两帧就是 $30 \times 30$



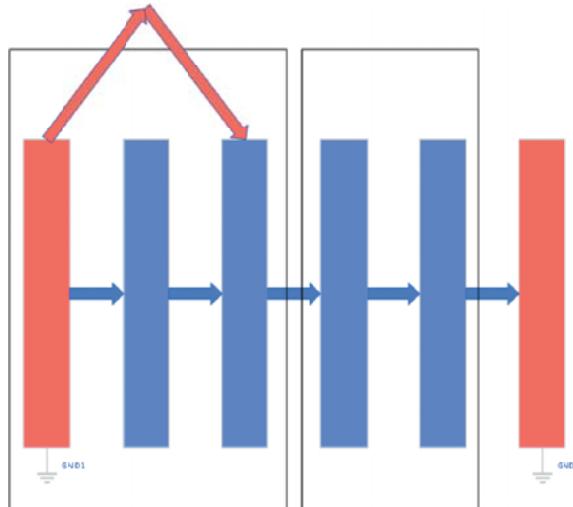
假设两帧状态的信息矩阵如上图，可分为4部分，待边缘化部分表示上一帧的状态信息矩阵，保留部分表示当前帧的状态信息矩阵，优化时由于要考虑速度问题，对于下一帧的优化，我们就不需要用上一帧的状态了，因此在当前帧需要把30维的信息矩阵压缩到15维，用比较通俗的说法就是下一帧的优化不会考虑上一帧的状态，但是出于保证信息不丢失的考虑，我们会把上一帧的信息融入到当前帧的信息中，这个过程就叫做边缘化。

边缘化的核心算法——舒尔补，对于上图来说我们要做的就是只保留右下角部分，但又不能直接切掉。

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C - B^T A^{-1} B \end{bmatrix}$$

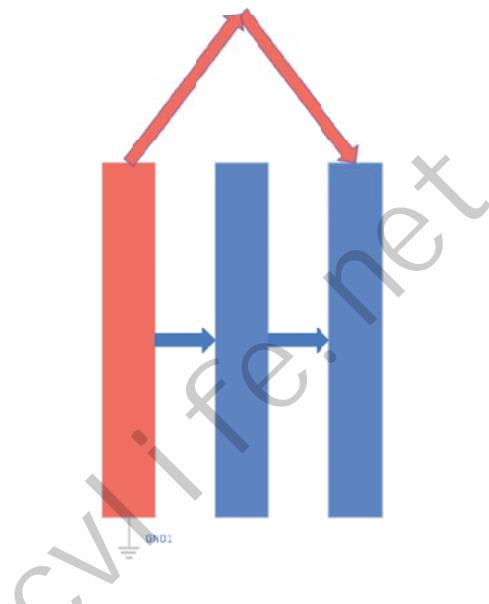
这样一来就相当于把矩阵的其他位置融入了右下角。最后用右下角表示当前帧的信息矩阵，也就是这个信息矩阵包含了前面帧与当前帧的信息

### IMU模式下单帧优化



每个框表示一种跟踪方式

### 跟踪关键帧



#### 条件：

当所在地图发生变化优化时（二三阶段初始化，尺度更新，局部地图优化，回环或地图融合）

#### 优化内容：

当前帧位姿，速度，偏置

#### 优化误差：

视觉重投影的边，与上一关键帧残差的边，与上一关键帧相比偏置不变的边（上一关键帧相关数据固定）

#### 边缘化：

不做边缘化，但保留优化后的当前帧的数据。

#### 数据包括：

当前帧位姿，速度，偏置，以及信息矩阵

#### 雅可比：

$r_n + t_n \ v_n \ g_n \ a_n$   
 $e_r \ J_{p1} \ J_{v1} \ J_{g1} \ J_{a1}$

#### 信息矩阵：

列数 6 3 3 3

// ----- 行数

```

//| Jp1.t * Jp1 Jp1.t * Jv1 Jp1.t * Jg1 Jp1.t * Ja1|6
//| Jv1.t * Jp1 Jv1.t * Jv1 Jv1.t * Jg1 Jv1.t * Ja1|3
//| Jg1.t * Jp1 Jg1.t * Jv1 Jg1.t * Jg1 Jg1.t * Ja1|3
//| Ja1.t * Jp1 Ja1.t * Jv1 Ja1.t * Jg1 Ja1.t * Ja1|3
//|-----

```

这里只是简单表示这个矩阵，实际上的内容要比这个复杂一点点，另外每一块实际应该是  $J.t^* H^* J$  这里省略了

## 跟踪普通帧



### 条件：

当所在地图未发生变化优化时（二三阶段初始化，尺度更新，局部地图优化，回环或地图融合）

### 优化内容：

当前帧位姿，速度，偏置

### 优化误差：

视觉重投影的边，与上一帧残差的边，与上一关键帧相比偏置不变的边，**上一帧数据维持不变的边  
(上一帧相关数据不固定)**

### 边缘化：

优化后做边缘化（边缘化的目标是上一帧的信息矩阵与当前帧的信息矩阵的融合），保留优化后的当前帧的数据。

### 数据包括：

当前帧位姿，速度，偏置，以及信息矩阵

### 信息矩阵：



## 第11章 不同模式适用场景及未来研究方向

### 不同传感器模式对比及应用场景

	单目	单目+IMU	双目	双目+IMU	RGB-D
地图初始化	SFM方式, 较慢	SFM方式, 较慢	第一帧即可初始化	第一帧即可初始化	第一帧即可初始化
鲁棒性	最低	较好	一般	最高	一般
定位精度	最低	较好	一般	最高	一般
运动较慢、匀速场景	√	✗	✓	不适合	✓

一般量化对比结果都是多次测量求平均值，并没有一种模式在所有数据集下都是最佳

		MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg <sup>1</sup>	
Monocular	ORB-SLAM <sup>2</sup> [4]	ATE <sup>2,3</sup>	0.071	0.067	0.071	0.082	0.060	<b>0.015</b>	0.020	-	<b>0.021</b>	<b>0.018</b>	-	0.047*
	DSO <sup>2</sup> [27]	ATE	0.046	0.046	0.172	3.810	0.110	0.089	0.107	0.903	0.044	0.132	1.152	0.501
	DSM <sup>2</sup> [31]	ATE	0.039	0.036	0.055	<b>0.057</b>	0.067	0.095	0.059	0.076	0.056	0.057	<b>0.784</b>	<b>0.126</b>
	ORB-SLAM3 (ours)	ATE	<b>0.017</b>	<b>0.017</b>	<b>0.031</b>	<b>0.066</b>	<b>0.044</b>	<b>0.033</b>	<b>0.016</b>	<b>0.037</b>	<b>0.021</b>	0.022	-	0.030*
Stereo	ORB-SLAM2 <sup>2</sup> [3]	ATE	0.035	<b>0.018</b>	0.028	0.119	0.060	<b>0.035</b>	<b>0.020</b>	<b>0.048</b>	0.037	0.035	-	0.044*
	VINS Fusion <sup>2</sup> [44]	ATE	0.540	0.460	0.330	0.780	0.500	0.550	0.230	-	0.230	0.200	-	0.424*
	SVO <sup>2</sup> [24]	ATE	0.040	0.070	0.270	0.170	0.120	0.040	0.040	0.070	0.050	0.090	0.790	0.159
	ORB-SLAM3 (ours)	ATE (m)	<b>0.025</b>	<b>0.022</b>	<b>0.027</b>	<b>0.089</b>	<b>0.058</b>	<b>0.035</b>	0.021	0.049	<b>0.032</b>	<b>0.027</b>	<b>0.361</b>	<b>0.068</b>
Monocular Inertial	ORB-SLAM VI <sup>2,3</sup> [4]	ATE <sup>2,3</sup> scale error <sup>2,3</sup>	0.075	0.084	0.087	0.217	0.082	<b>0.027</b>	0.028	-	<b>0.032</b>	0.041	0.074	0.075*
	VINS Mono <sup>2</sup> [7]	ATE <sup>4</sup>	0.084	0.105	0.074	0.122	0.147	0.047	0.066	0.180	0.056	0.090	0.244	0.110
	VI-DSO <sup>2</sup> [46]	ATE	0.062	<b>0.044</b>	0.117	0.132	0.121	<b>0.059</b>	0.067	0.096	0.040	0.062	0.174	0.089
	ORB-SLAM3 (ours) scale error	ATE	<b>0.032</b>	0.053	<b>0.033</b>	<b>0.099</b>	<b>0.071</b>	0.043	<b>0.016</b>	<b>0.025</b>	0.041	<b>0.015</b>	<b>0.037</b>	<b>0.342</b>
Stereo Inertial	OKVIS <sup>2,3</sup> [39]	ATE	0.160	0.220	0.240	0.340	0.470	0.090	0.200	0.240	0.130	0.160	0.290	0.230
	VINS Fusion <sup>2</sup> [44]	ATE <sup>4</sup>	0.166	0.152	0.125	0.280	0.284	0.076	0.069	0.114	0.066	0.091	0.096	0.138
	BASALT <sup>2</sup> [47]	ATE <sup>3</sup>	0.080	0.060	0.050	0.100	<b>0.080</b>	0.040	0.020	0.030	<b>0.030</b>	0.020	-	0.051*
	Kimera <sup>2</sup> [8]	ATE	0.080	0.090	0.110	0.150	0.240	0.050	0.110	0.120	0.070	0.100	0.190	0.119
ORB-SLAM3 (ours)	ATE	<b>0.037</b>	<b>0.031</b>	<b>0.026</b>	<b>0.059</b>	0.086	<b>0.037</b>	<b>0.014</b>	<b>0.023</b>	0.037	<b>0.014</b>	<b>0.029</b>	<b>0.336</b>	
	scale error	0.7	0.2	0.2	0.4	1.0	0.6	0.6	0.6	1.4	0.2	0.8	0.6	

<sup>1</sup> Average error of the successful sequences. Systems that did not complete all sequences are denoted by \* and are not marked in bold.

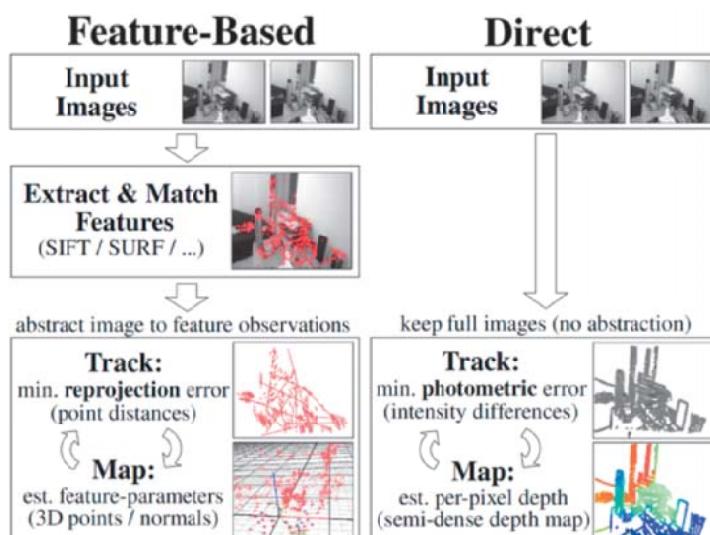
<sup>2</sup> Errors reported with raw GT instead of processed GT.

<sup>3</sup> Errors reported with keyframe trajectory instead of full trajectory.

<sup>4</sup> Errors obtained by ourselves, running the code with its default configuration.

鲁棒性和定位精度最好的是双目+IMU模式；但是如果对体积、成本、算力等有一定要求，推荐单目+IMU模式，其综合效果仅次于双目+IMU。

## 直接法 vs. 特征点法



## 直接法

优势：

1. 相比特征点法（只使用了特征点周围的信息）使用了图像中大部分的信息（半直接法只用了梯度）；
2. 节省特征提取与匹配的大量时间，易于移植到嵌入式系统中，以及与IMU进行融合；
3. 使用的是像素梯度而不必是角点，可以在特征缺失的场合使用，如环境中存在许多重复纹理或是缺乏角点，但出现许多边缘或光线变化不明显区域；
4. 可以进行稠密或半稠密的地图重建；

劣势：

1. 一般对相机要求较高，需要全局快门相机，进行光度标定等
2. 灰度不变是一个强假设，难以满足（易受曝光和模糊影像）；
3. 直接法成功的前提，是目标函数从初始值到最优值之间一直是下降的，然而图像非凸。因此需要有一个相当不错的初始估计，还需要一个质量较好的图像；
4. 只有短期、中期数据关联，缺乏长期数据关联。所以比较难实现地图复用、闭环和重定位、多地图系统。除非存储所有的关键帧图像，否则很难利用先前建好的地图；即使有办法存储所有关键帧的图像，那么在重用地图时，我们还需要对位姿有一个比较准确的初始估计——这通常是困难的。

## 特征点法

优势：

1. 相对直接法，特征点对光照变化、较大运动、动态物体不敏感，比较稳定。
2. 通过特征匹配，可以实现长期数据关联、多地图数据关联，方便实现地图复用、闭环和重定位、多地图系统。
3. 是视觉SLAM里的主流方法，也是比较成熟的方案。

劣势：

1. 一幅图像有几十上百万的像素，但特征点只有几百个。丢掉了大部分的图像信息。
2. 提取特征点比较耗时，计算资源消耗较大
3. 建立的地图一般是稀疏的特征点地图，主要用来定位
4. 在缺乏纹理（比如空旷的走廊）或者重复纹理（瓷砖）场景下，特征点数目会明显减少，可能找不到足够的匹配点来估计相机位姿，导致跟踪丢失。在该场景下特征点法跟踪没有直接法（比如Lucas-Kanade光流）稳定。

Table I: Summary of the most representative visual (top) and visual-inertial (bottom) systems, in chronological order.

	SLAM or VO	Pixels used	Data association	Estimation	Relocation	Loop closing	Multi Maps	Mono	Stereo	Mono IMU	Stereo IMU	Fisheye	Accuracy	Robustness	Open source
Mono-SLAM [13], [14]	SLAM	Shi Tomasi	Correlation	EKF	-	-	-	✓	-	-	-	-	Fair	Fair	[15] <sup>1</sup>
PTAM [16]–[18]	SLAM	FAST	Pyramid SSD	BA	Thumbnail	-	-	✓	-	-	-	-	Very Good	Fair	[19]
LSD-SLAM [20], [21]	SLAM	Edgelets	Direct	PG	-	FABMAP PG	-	✓	✓	-	-	-	Good	Good	[22]
SVO [23], [24]	VO	FAST+ Hi.grad.	Direct	Local BA	-	-	-	✓	✓	-	-	✓	Very Good	Very Good	[25] <sup>2</sup>
ORB-SLAM2 [2], [3]	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	-	✓	✓	-	-	-	Exc.	Very Good	[26]
DSO [27]–[29]	VO	High grad.	Direct	Local BA	-	-	-	✓	✓	-	-	✓	Good	Very Good	[30]
DSM [31]	SLAM	High grad.	Direct	Local BA	-	-	-	✓	-	-	-	-	Very Good	Very Good	[32]
MSCKF [33]–[36]	VO	Shi Tomasi	Cross correlation	EKF	-	-	-	✓	-	✓	✓	-	Fair	Very Good	[37] <sup>3</sup>
OKVIS [38], [39]	VO	BRISK	Descriptor	Local BA	-	-	-	-	-	✓	✓	-	Good	Very Good	[40]
ROVIO [41], [42]	VO	Shi Tomasi	Direct	EKF	-	-	-	-	-	✓	-	-	Good	Good	[43]
ORBSLAM-VI [4]	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	-	✓	✓	✓	-	-	Very Good	Very Good	-
VINS-Fusion [44]	VO	Shi Tomasi	KLT	Local BA	DBoW2	DBoW2 PG	✓	-	✓	✓	✓	✓	Very Good	Exc.	[45]
VI-DSO [46]	VO	High grad.	Direct	Local BA	-	-	-	-	-	✓	-	-	Very Good	Exc.	-
BASALT [47]	VO	FAST	KLT (LSSD)	Local BA	-	ORB BA	-	-	-	-	✓	-	Very Good	Exc.	[48]
Kimera [8]	VO	Shi Tomasi	KLT	Local BA	-	DBoW2 PG	-	-	-	-	✓	-	Good	Exc.	[49]
ORB-SLAM3 (ours)	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	✓	✓	✓	✓	✓	✓	Exc.	Exc.	[5]

<sup>1</sup> Last source code provided by a different author. Original software is available at [50].<sup>2</sup> Source code available only for the first version, SVO 2.0 is not open source.<sup>3</sup> MSCKF is patented [51], only a re-implementation by a different author is available as open source.

## 代码bug梳理

注意：和ORB-SLAM2相同的代码bug这里不再重复，请看ORB-SLAM2课程里最后一章的讲解。这里只考虑ORB-SLAM3里新内容的bug。

Tracking.cc

RGB-D模式下忘记读取mbf, mbf = baseline \* fx

```
// if(mSensor==System::STEREO || mSensor==System::IMU_STEREO) !bug 忘记读取rgbd的mbf
if(mSensor==System::STEREO || mSensor==System::IMU_STEREO || mSensor==System::RGBD)
{
    cv::FileNode node = fSettings["Camera.bf"];
    if(!node.empty() && node.isReal())
    {
        mbf = node.real();
    }
    else
    {
        std::cerr << "*Camera.bf parameter doesn't exist or is not a real number*" << std::endl;
        b_miss_params = true;
    }
}
```

影响：

1、会导致深度值阈值为0，近远点判断失效

```

if(mSensor==System::STEREO || mSensor==System::RGBD || mSensor==System::IMU_STEREO)
{
    float fx = mpCamera->getParameter(0);
    cv::FileNode node = fSettings["ThDepth"];
    if(!node.empty() && node.isReal())
    {
        mThDepth = node.real();
        mThDepth = mbf*mThDepth/fx;
        cout << endl << "Depth Threshold (Close/Far Points): " << mThDepth << endl;
    }
    else
    {
        std::cerr << "*ThDepth parameter doesn't exist or is not a real number*" << std::endl;
        b_miss_params = true;
    }
}

```

Tracking.cc > {} ORB\_SLAM3 > NeedNewKeyFrame()

```

// Check how many "close" points are being tracked and how many could be tracked
// Step 6: 对于双目或RGBD摄像头，统计成功跟踪的近点的数量，如果跟踪到的近点太少，没有足够的信息进行匹配
int nNonTrackedClose = 0; // 双目或RGB-D中没有跟踪到的近点
int nTrackedClose= 0; // 双目或RGB-D中成功跟踪的近点（三维点）

if(mSensor!=System::MONOCULAR && mSensor!=System::IMU_MONOCULAR)
{
    int N = (mCurrentFrame.Nleft == -1) ? mCurrentFrame.N : mCurrentFrame.Nleft;
    for(int i =0; i<N; i++)
    {
        // 深度值在有效范围内
        if(mCurrentFrame.mvDepth[i]>0 && mCurrentFrame.mvDepth[i]<mThDepth)
        {
            if(mCurrentFrame.mvpMapPoints[i] && !mCurrentFrame.mvbOutlier[i])
                nTrackedClose++;
            else
                nNonTrackedClose++;
        }
    }
}

```

## 2、新生成地图点时匹配出错

Tracking.cc > {} ORB\_SLAM3 > CreateNewMapPoints()

```

if(!bStereo2)
{
    float u2 = fx2*x2*invz2+cx2;
    float v2 = fy2*y2*invz2+cy2;
    float errX2 = u2 - kp2.pt.x;
    float errY2 = v2 - kp2.pt.y;
    if((errX2*errX2+errY2*errY2)>5.991*sigmaSquare2)
        continue;
}
else
{
    float u2 = fx2*x2*invz2+cx2;
    float u2_r = u2 - mpLastKeyFrame->mbf*invz2;
    float v2 = fy2*y2*invz2+cy2;
    float errX2 = u2 - kp2.pt.x;
    float errY2 = v2 - kp2.pt.y;
    float errX2_r = u2_r - kp2_ur;
    if((errX2*errX2+errY2*errY2+errX2_r*errX2_r)>7.8*sigmaSquare2)
        continue;
}

```

3、优化等其他需要用到mbf的地方失效，不再一一列举

重定位跟踪中忘记累加候选数目



```
C Tracking.cc > {} ORB_SLAM3 > Relocalization()
int nCandidates=0;

// Step 3: 遍历所有的候选关键帧，通过BoW进行快速匹配，用匹配结果初始化PnP
for(int i=0; i<nKFs; i++)
{
    KeyFrame* pKF = vpCandidateKFs[i];
    if(pKF->isBad())
        vbDiscarded[i] = true;
    else
    {
        // 当前帧和候选关键帧用BoW进行快速匹配，匹配结果记录在vvvMapPoint
        int nmatches = matcher.SearchByBoW(pKF,mCurrentFrame,vvvMapPoint);
        // 如果和当前帧的匹配数小于15，那么只能放弃这个关键帧
        if(nmatches<15)
        {
            vbDiscarded[i] = true;
            continue;
        }
        else
        {
            // 如果匹配数目够用，用匹配结果初始化MLPnP solver
            // ? 为什么用MLPnP? 因为考虑了鱼眼相机模型，解耦某些关系？
            // 参考论文《MLPnP-A REAL-TIME MAXIMUM LIKELIHOOD SOLUTION TO
            MLPnPSolver* pSolver = new MLPnPsolver(mCurrentFrame,vvvMapPoint);
            // 构造函数调用了一遍，这里重新设置参数
            pSolver->SetRansacParameters(
                0.99,                                     // 模型最大概率值，默认0.9
                10,                                       // 内点的最小阈值，默认8
                300,                                      // 最大迭代次数，默认300
                6,                                         // 最小集，每次采样六个点，即最小
                0.5,                                       // 理论最少内点个数，这里是按照总
                5.991);                                    // 卡方检验阈值 //This solver ne
            vpMLPnPsolvers[i] = pSolver;
            // !bug 忘记更新nCandidates
            ++nCandidates;
        }
    }
}
```

影响：会导致后续重定位失败

```

4234
4235
4236
4237
4238
4239
4240
4241 // Alternatively perform some iterations of P4P RANSAC
4242 // Until we found a camera pose supported by enough inlier
4243 // 足够的内点才能匹配使用PNP算法，MLPnP需要至少6个点
4244 // 是否已经找到相匹配的关键帧的标志
4245 bool bMatch = false;
4246 ORBmatcher matcher2(0.9,true);
4247
4248 // Step 4: 通过一系列操作,直到找到能够匹配上的关键帧
4249 // 为什么搞这么复杂? 答: 是担心误闭环
4250 while(nCandidates>0 && !bMatch)
4251 {
4252     // 遍历当前所有的候选关键帧
4253     for(int i=0; i<nKFs; i++)
4254     {

```

## LocalMapping.h/cc

LocalMapping初始化列表里const float bMonocular 应该改为bool bMonocular

影响：影响严谨性

```

LocalMapping::LocalMapping(System* pSys, Atlas *pAtlas, const float bMonocular, bool bInertial, const string & strSequence)
{
    mpSystem(pSys), mbMonocular(bMonocular), mbInertial(bInertial), mbResetRequested(false), mbResetRequestedActive(false),
    mbAbortBA(false), mbStopped(false), mbStopRequested(false), mbNotStop(false), mbAcceptKeyFrames(true),
    mbNewInit(false), mIdxInit(0), mScale(1.0), mInitSect(0), mbNotBA1(true), mbNotBA2(true), infoInertial(Eigen::Vector3d());
}

//Initialize the Local Mapping thread and launch
//创建并开启local_mapping线程
mpLocalMapper = new LocalMapper(this, mpAtlas, mSensor==MONOCULAR || mSensor==IMU_MONOCULAR, mSensor==IMU_MONOCULAR || mSensor==IMU_STEREO, strSequence);
mpLocalMapper = new thread(&ORB_SLAM3::LocalMapper::run, mpLocalMapper);

bool mbMonocular; //mSensor==MONOCULAR || mSensor==IMU_MONOCULAR
bool mbInertial; //mSensor==IMU_MONOCULAR || mSensor==IMU_STEREO

// Input sensor
enum eSensor{
    MONOCULAR=0,
    STEREO=1,
    RGBD=2,
    IMU_MONOCULAR=3,
    IMU_STEREO=4
};

```

## LoopClosing.cc

忘记用优化后的Sim3位姿来更新。

影响：会影响最后用优化后的sim3位姿投影匹配成功的点数目

```

1 // LoopClosing.cc > {} ORB_SLAM3 > DetectAndRefineSim3FromLastKF(KeyFrame * pKF, KeyFrame *& pNewKF, g2o::Sim3 &, int &, std::vector<MapPoint*> &, std::vector<MapPoint*> &)
2
3 // 为 OptimizeSim3接口准备数据
4 cv::Mat mScw = Converter::toCvMat(gScw);
5 cv::Mat mTwm = pMatchedKF->GetPoseInverse();
6 g2o::Sim3 gSwm(Converter::toMatrix3d(mTwm.rowRange(0, 3).colRange(0, 3)), Converter::toVector3d(mTwm.rowRange(0, 3).col(3)), 1.0);
7 g2o::Sim3 gScm = gScw * gSwm;
8 Eigen::Matrix<double, 7, 7> mHessian7x7;
9
10
11 // 单目情况下不锁定尺度
12 bool bFixedScale = mbFixScale; // TODO CHECK; Solo para el monocular inertial
13 // 如果是imu模式且未完成初始化,不锁定尺度
14 if(mpTracker->mSensor==System::IMU_MONOCULAR && !pCurrentKF->GetMap()->GetInertialBA2())
15 | bFixedScale=false;
16 // 继续优化 Sim3
17 int numOptMatches = Optimizer::OptimizeSim3(mpCurrentKF, pMatchedKF, vpMatchedMPs, gScm, 10, bFixedScale, mHessian7x7, true);
18
19 // 若匹配的数量大于一定的数目
20 if(numOptMatches > nProjOptMatches)
21 {
22     //! bug, 以下gScw_estimation应该通过上述sim3优化后的位姿来更新。以下mScw应该改为 gScm * gSwm^-1
23     g2o::Sim3 gScw_estimation(Converter::toMatrix3d(mScw.rowRange(0, 3).colRange(0, 3)),
24         Converter::toVector3d(mScw.rowRange(0, 3).col(3)), 1.0);
25
26     vector<MapPoint*> vpMatchedMP;
27     vpMatchedMP.resize(mpCurrentKF->GetMapPointMatches().size(), static_cast<MapPoint*>(NULL));
28     // 再次通过优化后的Sim3搜索匹配点。
29     nNumProjMatches = FindMatchesByProjection(pCurrentKF, pMatchedKF, gScw_estimation, spAlreadyMatchedMPs, vpMPs, vpMatchedMPs);
30     //cout << "REFINE-SIM3: Projection with optimize Sim3 from last KF with " << nNumProjMatches << " matches" << endl;

```

检测共同区域时，寻找最佳候选关键帧时，忘记了更新。

影响：导致寻找最佳候选关键帧代码失效，一直都是用的第一个候选帧。

```

1 // LoopClosing.cc > {} ORB_SLAM3 > DetectCommonRegionsFromBoW(std::vector<KeyFrame*> &, KeyFrame *&, KeyFrame *&, g2o::Sim3 &, std::vector<MapPoint*> &, std::vector<MapPoint*> &)
2
3 std::set<MapPoint*> spMatchedMPi;
4 int numBoWMatches = 0;
5 // 记录窗口中能通过bow在当前关键帧ka中找到最多匹配点的关键帧
6 KeyFrame* pMostBoWMatchesKF = pKFi;
7 // 记录窗口中能通过bow在当前关键帧ka中找到最多匹配点的数量
8 int nMostBoWNumMatches = 0;
9
10
11 // 下面两个变量是为了sim3 solver准备的
12 //记录窗口中的地图点能在当前关键帧中找到的匹配的点(数量的上限是当前关键帧地图点的数量)
13 std::vector<MapPoint*> vpMatchedPoints = std::vector<MapPoint*>(mpCurrentKF->GetMapPointMatches().size(), static_cast<MapPoint*>(NULL));
14 // 记录上面的地图点分别对应窗口中的关键帧(数量的上限是当前关键帧地图点的数量)
15 std::vector<KeyFrame*> vpKeyFrameMatchedMP = std::vector<KeyFrame*>(mpCurrentKF->GetMapPointMatches().size(), static_cast<KeyFrame*>(NULL));
16
17 // 记录在W_km中有最多匹配点的帧的局部index, 这个后面没有用到
18 int nIndexMostBoWMatchesKF=0;
19 //! bug: 以下循环中并没有重新赋值pMostBoWMatchesKF, 一直是初始值: 候选关键帧
20 // 遍历窗口内Wn的每个关键帧
21 // Step 1.1 通过Bow寻找候选帧窗口内的关键帧地图点与当前关键帧的匹配点
22 for(int j=0; j<vpCovKFi.size(); ++j)
23 {
24     if(!vpCovKFi[j] || vpCovKFi[j]->isBad())
25         continue;
26
27     int num = matcherBoW.SearchByBoW(mpCurrentKF, vpCovKFi[j], vvpMatchedMPs[j]);
28     //cout << "BoW: " << num << " putative matches with KF " << vpCovKFi[j]->mnId << endl;
29     if (num > nMostBoWNumMatches)
30     {
31         nMostBoWNumMatches = num;
32         nIndexMostBoWMatchesKF = j; //!
33     }
34 }
35 // 标记是否因为窗口内有当前关键帧的共视关键帧

```

惯性模式下的地图融合，地图以旧换新。换完后忘记把旧地图设置为BAD

影响：没办法删除旧地图了。mpAtlas->RemoveBadMaps();

```

LoopClosing.cc > {} ORB_SLAM3 > MergeLocal2()

    // Make sure connections are updated
    // 把该关键帧从融合帧所在地图删掉,加入到当前的地图中
    pKFi->UpdateMap(pCurrentMap);
    pCurrentMap->AddKeyFrame(pKFi);
    pMergeMap->EraseKeyFrame(pKFi);
}

// 遍历每个融合帧所在地图的地图点
for(MapPoint* pMPi : vpMergeMapMPs)
{
    if(!pMPi || pMPi->isBad() || pMPi->GetMap() != pMergeMap)
        continue;

    // 把地图点添加到当前帧所在地图,从融合帧所在地图删掉
    pMPi->UpdateMap(pCurrentMap);
    pCurrentMap->AddMapPoint(pMPi);
    pMergeMap->EraseMapPoint(pMPi);
}

// ? BUG! pMergeMap没有设置为BAD
// ? 应该加入如下代码吧?
// ? mpAtlas->SetMapBad(pMergeMap);

```

## 工程化建议

工程上：无非三个要素，速度，精度，鲁棒性

所有开源框架都是学术成果，（不考虑版权问题的情况下）无法做到拿来就直接能用到项目或产品，必须要在不同应用场景下做移植、魔改、加速、封装等。

不盲从、敢于质疑作者。开源框架中一般都有不少出错的地方。

ORB-SLAM3不适用场景：

- 运动速度较慢、匀速运动的场景下不适合用视觉惯性模式。因为此时IMU没有有效输出，很难成功初始化。
- 自动驾驶。因为自动驾驶一般不需要闭环，比较适合轻量级的VIO。不过特定范围的园区内仍然可以考虑。

ORB-SLAM3工程化改进建议：

- 代码bug方面：代码中存在不少bug（见前面bug说明，不全）。需要重新梳理，修复bug（包括ORB-SLAM2遗留的代码）。
- 代码规范化方面：单个文件代码量过大、部分功能封装的不好。不利于阅读、不利于多人协同开发，推荐再细化，拆分。例如跟踪线程里那几个跟踪状态，理起来很费劲，容易出错；回环线程一个函数不要包含那么多代码；匹配函数里大量重复的代码；Initializer.cc和TwoViewReconstruction.cc代码重复，可删除。
- 代码逻辑和参数方面：个别地方的逻辑有待商榷，个别地方的阈值有待商榷，个别地方的处理有待商榷。这个也是要结合具体的使用场景，特别是阈值，很多都是写死的，那如果数据集的图像分辨率大，有些阈值是不是可以随着分辨率变化而变化做一些超参。优化部分也是有一些处理不当的地方。
- 地图复用问题。ORB-SLAM3没有实现，可以参考[https://github.com/UZ-SLAMLab/ORB\\_SLAM3/pull/310](https://github.com/UZ-SLAMLab/ORB_SLAM3/pull/310)

- 代码加速。读取词袋部分可以改成二进制词袋，占用空间小，读取速度快。此外，好多地方代码冗余，做了很多无意义的操作，例如优化中好多for循环都没用，简直是时间杀手。比如更新共视关系的地方也有bug，导致每次都会更新小于15个共视地图点的关键帧关系（实际上用不到）；另外，特别注意和地图点相关的遍历、更新操作，因为量大，处理起来都是时间杀手。
- 代码中调试工具：源码里包含了很多调试代码，建议弄成宏定义判断是否执行，利用好这些调试的代码可以事半功倍，例如回环时显示一下两张回环的匹配图像等等。

## 研究方向建议

1. 用直接法实现短期、中期、长期、多地图数据关联。作者在探索这个方向，用于人体内窥镜建图。
2. 基于CNN的单目深度估计可以得到可靠的深度图，可以和ORB-SLAM3结合。
3. 特征提取加速方法。

比如这篇论文对ORB-SLAM2进行了加速。《Fast ORB-SLAM with Descriptor Independent Keypoint Matching》

### DEMO

思路：普通帧提取特征点只是为了跟踪，但如果后续没有被确定为关键帧，这些特征点再也用不到了。而提取特征点很费时间，能否用直接法（比如稀疏光流来替代特征点来跟踪）。只有关键帧才计算描述子。具体分2步，第1步：结合UAM (uniform acceleration motion)运动模型，用稀疏光流+金字塔来粗糙跟踪；第2步：用GMS和基于RANSAC的极线约束来滤除误匹配。

## Fast ORB-SLAM with Descriptor Independent Keypoint Matching

Qiang Fu, Hongshan Yu, Xiaolong Wang, Zhengeng Yang, Yong He, Hong Zhang, Fellow, IEEE, Ajmal Mian

**Abstract**—Indirect methods for visual SLAM are gaining popularity due to their robustness to environmental variations. ORB-SLAM2 [1] is a benchmark method in this domain, however, it consumes significant time for computing descriptors that never get reused unless a frame is selected as a keyframe. To overcome these problems, we present FastORB-SLAM which is light-weight and efficient as it tracks keypoints between adjacent frames without computing descriptors. To achieve this, a two stage coarse-to-fine descriptor independent keypoint matching method is proposed based on sparse optical flow. In the first stage, we predict initial keypoint correspondences via a simple but effective motion model and then robustly establish the correspondences via pyramid-based sparse optical flow tracking. In the second stage, we leverage the constraints of the motion smoothness and epipolar geometry to refine the correspondences. In particular, our method computes descriptors only for keyframes. We test FastORB-SLAM on TUM and ICL-NUIM RGB-D datasets and compare its accuracy and efficiency to nine existing RGB-D SLAM methods. Qualitative and quantitative results show that our method achieves state-of-the-art accuracy and is about twice as fast as the ORB-SLAM2.

**Index Terms**—Visual SLAM, ORB SLAM, Keypoint Matching, Optical Flow, Motion Model.

v2 [cs.RO] 24 Jun 2021

## 结果

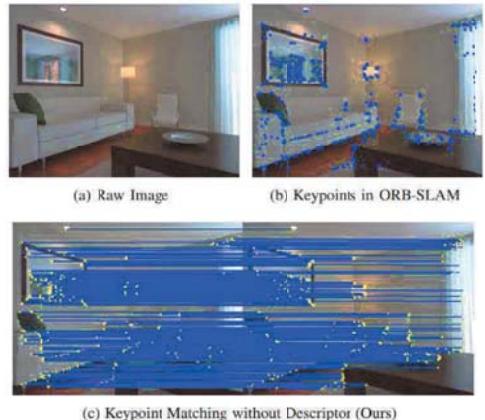


Fig. 1. Illustration of our keypoint matching method between two adjacent

**TABLE II**  
CAMERA LOCALIZATION RMSE [M] ERROR AND AVERAGE TIME [S] COMPARISON ON THE *ICL-NUIM* DATASET

	Office 0		Office 1		Office 2		Office 3		Living 0		Living 1		Living 2		Living 3	
	RMSE	Time														
ORB-SLAM2	0.038	0.021	<b>0.070</b>	0.024	<b>0.011</b>	0.020	0.066	0.021	<b>0.006</b>	0.021	0.101	0.024	<b>0.015</b>	0.021	0.013	0.022
Ours	<b>0.034</b>	<b>0.013</b>	0.080	<b>0.013</b>	0.015	<b>0.014</b>	<b>0.037</b>	<b>0.012</b>	0.010	<b>0.014</b>	<b>0.026</b>	<b>0.015</b>	0.016	<b>0.013</b>	<b>0.009</b>	<b>0.012</b>

All statistics are collected via real reproduction test i.e. median over 5 executions for each sequence. RMSE represents translation in meters. Time represents average time per frame in seconds. On the **Office 3** and **Living 1** sequences, our method achieves much higher localization accuracy than ORB-SLAM2 with less computation time. Our method is also highly competitive on the other sequences.

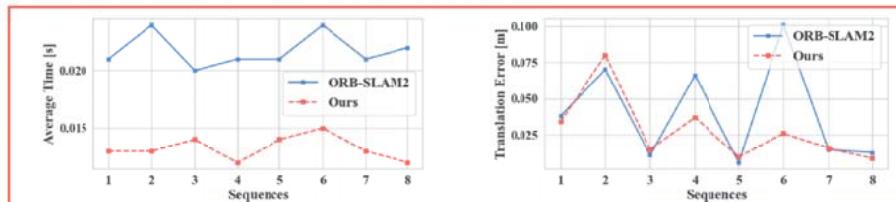


Fig. 8. Average Time and Translation Error comparison in all 8 sequences (Office 0-3 and Living 0-3) of *ICL-NUIM* dataset. More details are in Table II. Our method produces highly competitive localization accuracy with significantly lower processing time. In Sequence 4 (Office 3) and Sequence 6 (Living 1), ORB-SLAM2 has a large drift error whereas our method maintains lower errors. Fig. 9 provides a more detailed comparison of these two sequences.

#### 4. 深度学习特征点

2020 IROS 《DXSLAM: A Robust and Efficient Visual SLAM System with Deep Features》

论文地址: <https://arxiv.org/pdf/2008.05416.pdf>

论文代码: <https://github.com/ivipsourcecode/dxslam>

思路:

使用CNN (Convolutional Neural Network, 卷积神经网络) 提取特征, 然后将所提取的特征整合到ORB SLAM2框架中。其中, 作者选择了性能优异的HF-Net, 提取每帧图像的局部特征以及整幅图像的全局特征, 使得所整合的SLAM系统, 相比使用手工特征点的SLAM系统, 在环境变化、视角变化情况下拥有更好的鲁棒性。ORB SLAM2中使用BoW (Bag of Words, 词袋模型) 来加速特征点的匹配, 本文中作者使用了FBoW (Fast Bag of Words) 来进一步加速词袋的训练以及通过词袋模型进行的特征点匹配过程。使用Intel神经网络优化工具OpenVINO之后, 该系统可以不需要GPU, 只靠CPU达到了实时性。

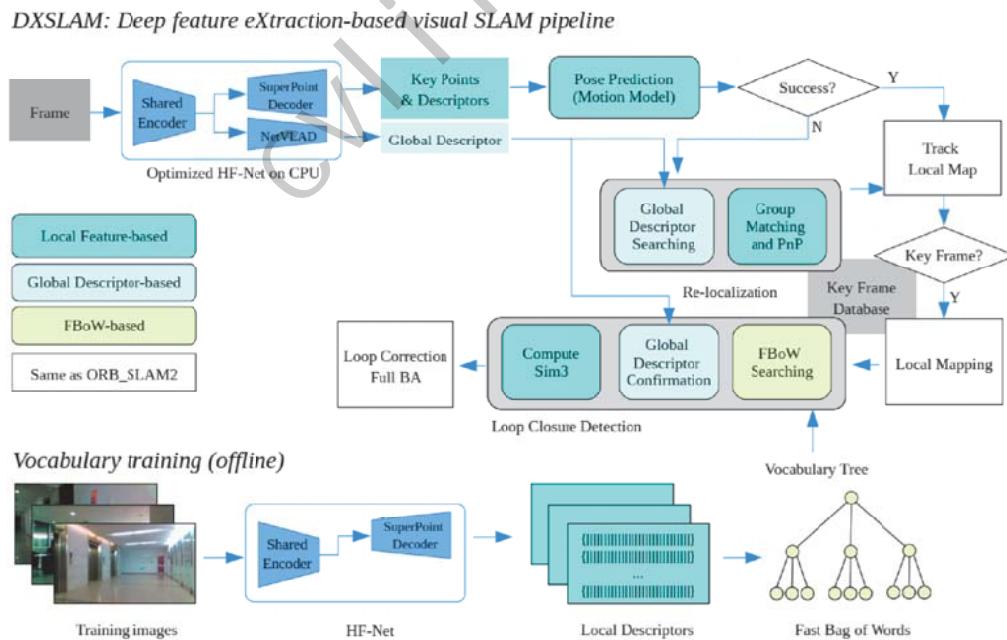


Fig. 1. The framework of the proposed visual SLAM system. The pipeline is largely the same with ORB-SLAM2 [5], with deep features incorporated into various modules of the system.

如何有思路? 多看论文、多写代码、多尝试

## 第12章 实战：稠密重建及课程大作业

具体内容见ORB-SLAM3第③期课程视频

本课件是计算机视觉life平台第③期《VIO天花板：ORB-SLAM3原理剖析+逐行源码详解》课程手册，配套课程视频学习。更多内容详见课程视频。课程官网：[cvlife.net](http://cvlife.net)

本课件后续更新电子版会同步到ORB-SLAM3课程群，请购买课程后联系客服领取。

cvlife.net