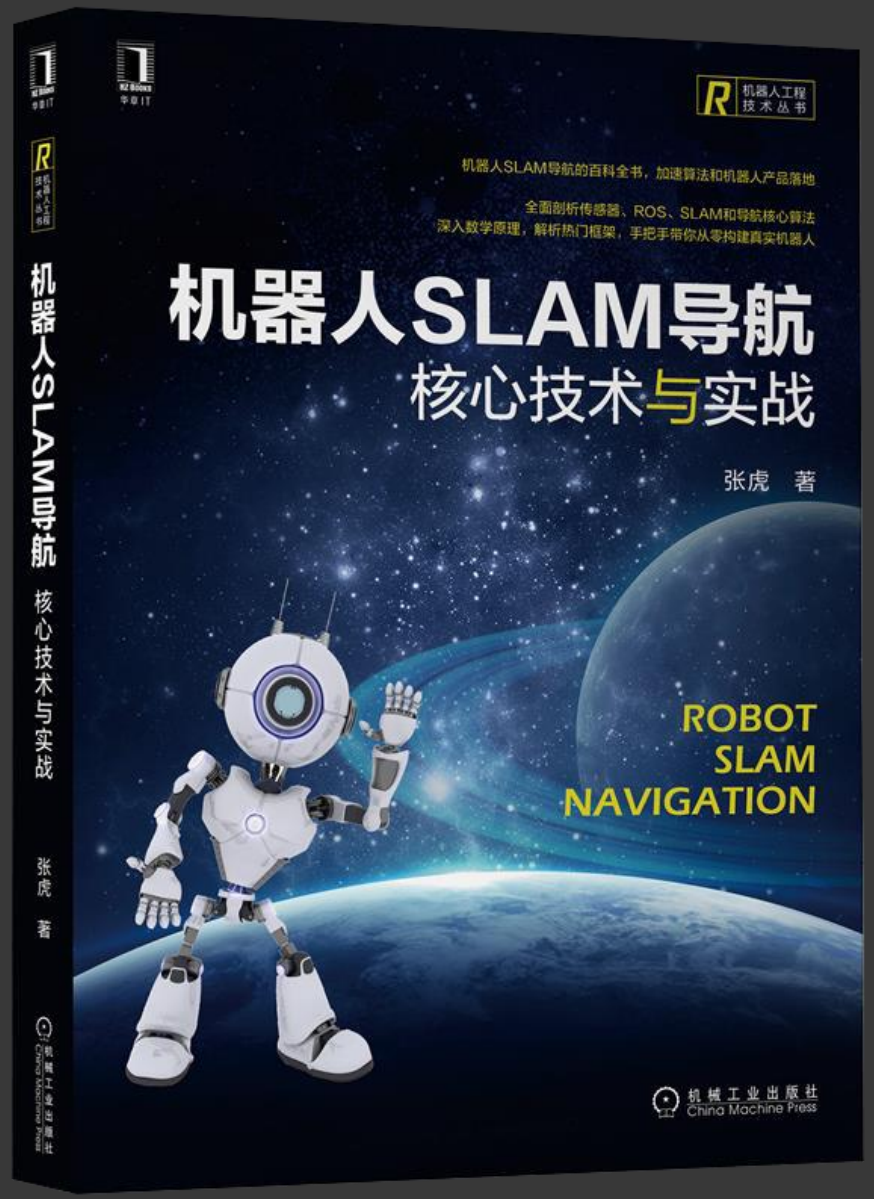


第1季

第9章：视觉SLAM系统

主讲人：张虎

（小虎哥哥爱学习）

- 先导课
- 第1季：快速梳理知识要点与学习方法 √
- 第2季：详细推导数学公式与代码解析
- 第3季：代码实操以及真实机器人调试
- 答疑课

------（永久免费●系列课程●长期更新）------

本书内容安排

一、编程基础篇

第1章：ROS入门必备知识

第2章：C++编程范式

第3章：OpenCV图像处理

二、硬件基础篇

第4章：机器人传感器

第5章：机器人主机

第6章：机器人底盘

三、SLAM篇

第7章：SLAM中的数学基础

第8章：激光SLAM系统

第9章：视觉SLAM系统

第10章：其他SLAM系统

四、自主导航篇

第11章：自主导航中的数学基础

第12章：典型自主导航系统

第13章：机器人SLAM导航综合实战

为什么需要视觉SLAM？

误区纠正：视觉SLAM并不高端

激光SLAM　?　视觉SLAM

（激光雷达）

（相机）

为什么需要视觉SLAM？

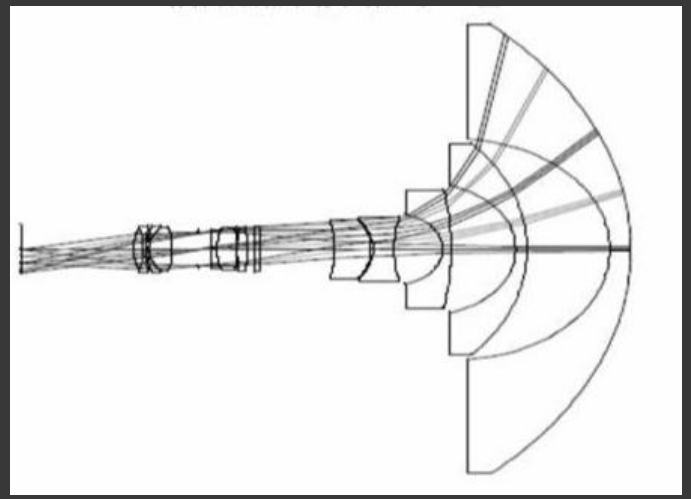
	激光SLAM	视觉SLAM
主传感器	激光雷达（单线/多线/固态）	相机（单目/双目/RGB-D）
观测数据	稳定性好（√） 测距精度高（√） 扫描范围广（√） 信息量小（×）	稳定性一般（×） 测距精度低（×） 扫描范围窄（×） 信息量大（√）
适用场景	光照变化不敏感（√） 雨雾天易失效（×）	光照变化敏感（×） 雨雾天不易失效（√）
传感器成本	高（×）	低（√）
消耗算力	小（√）	大（×）

小虎哥的观点：视觉SLAM并不高大上，激光SLAM也并不低端

激光SLAM和视觉SLAM，各有优劣，两者相结合是趋势

视觉SLAM算法有哪些？

按**主传感器**分类
- 单目（鱼眼/全景）
- 双目（三目/四目/...）
- RGB-D



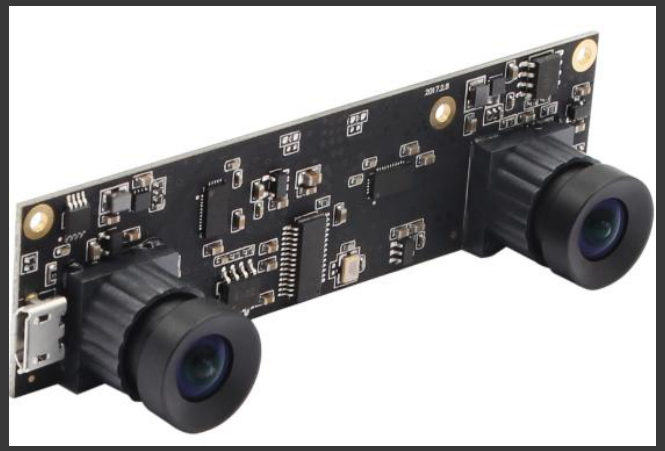
成像原理：

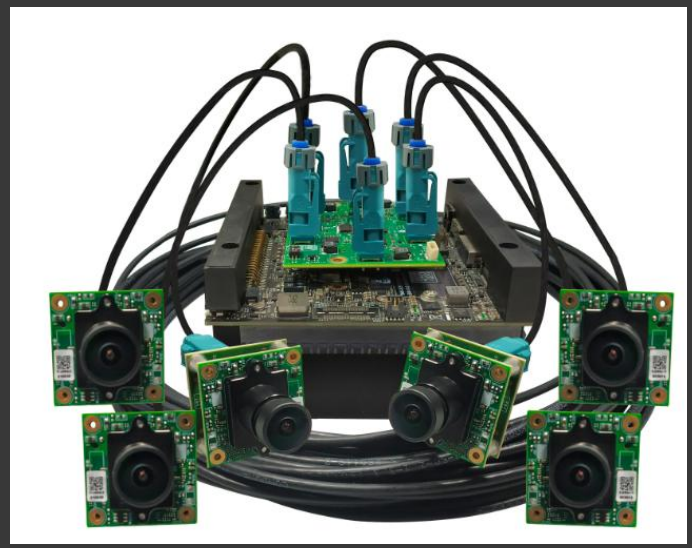
入射光线被成像芯片捕获

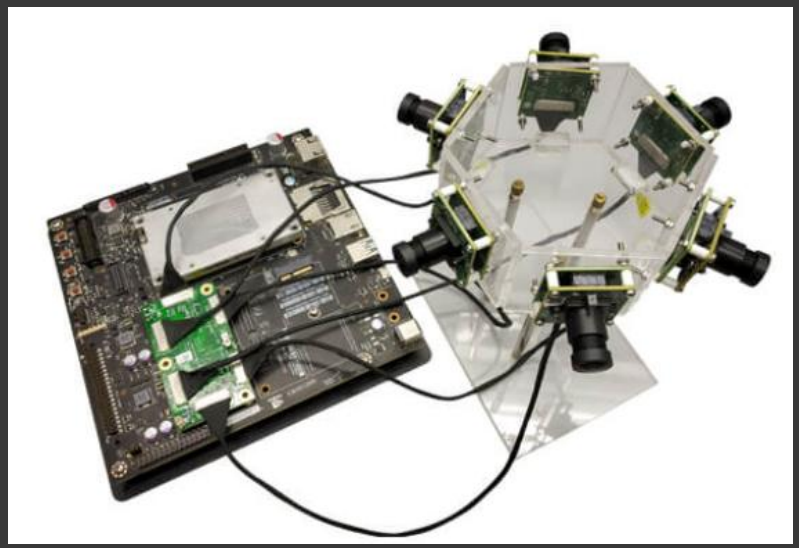
* 关于相机原理的具体讲解，请阅读书本第4章的内容

视觉SLAM算法有哪些？

按**主传感器**分类
- 单目（鱼眼/全景）
- 双目（三目/四目/...）
- RGB-D





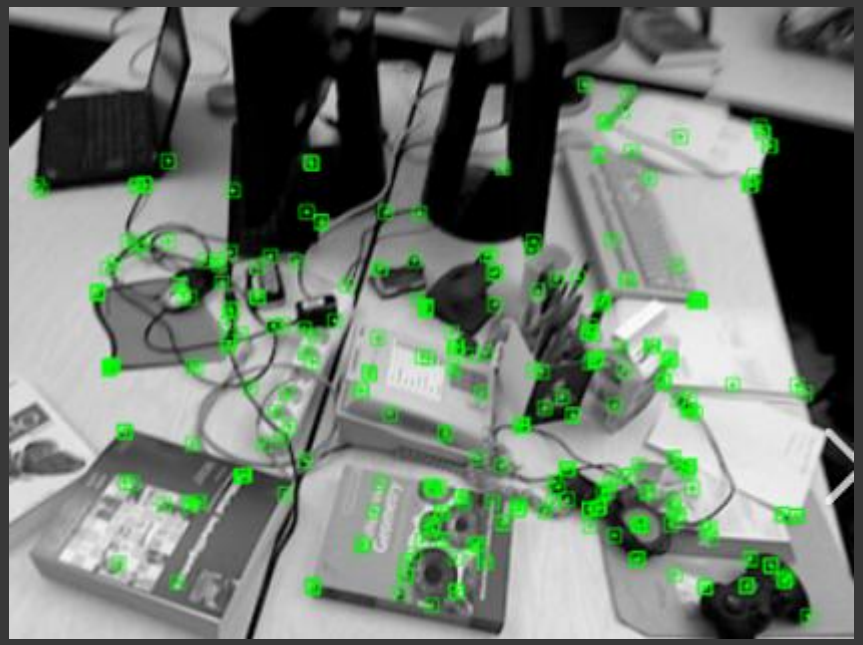


视觉SLAM算法有哪些？

按主传感器分类
- 单目（鱼眼/全景）
- 双目（三目/四目/...）
- RGB-D

视觉SLAM算法有哪些？

- 按主传感器分类
  - 单目（鱼眼/全景）
  - 双目（三目/四目/...）
  - RGB-D
- 按图像分析方法分类
  - 特征点法
  - 直接法
  - 半直接法

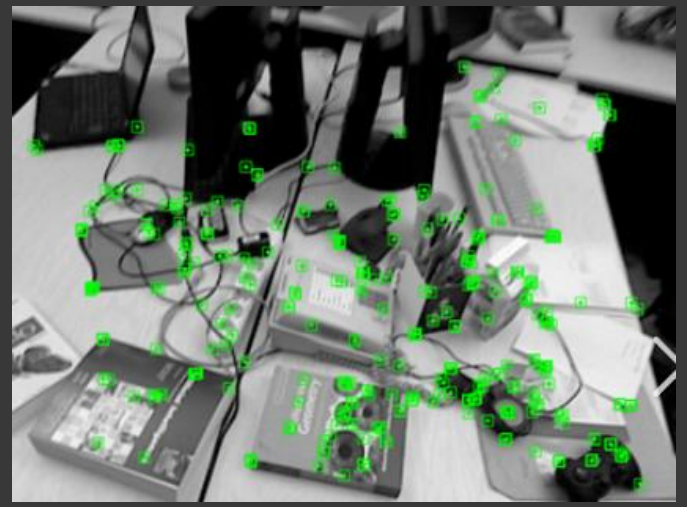


视觉SLAM算法有哪些？

- 按主传感器分类
  - 单目（鱼眼/全景）
  - 双目（三目/四目/...）
  - RGB-D
- 按图像分析方法分类
  - 特征点法
  - 直接法
  - 半直接法

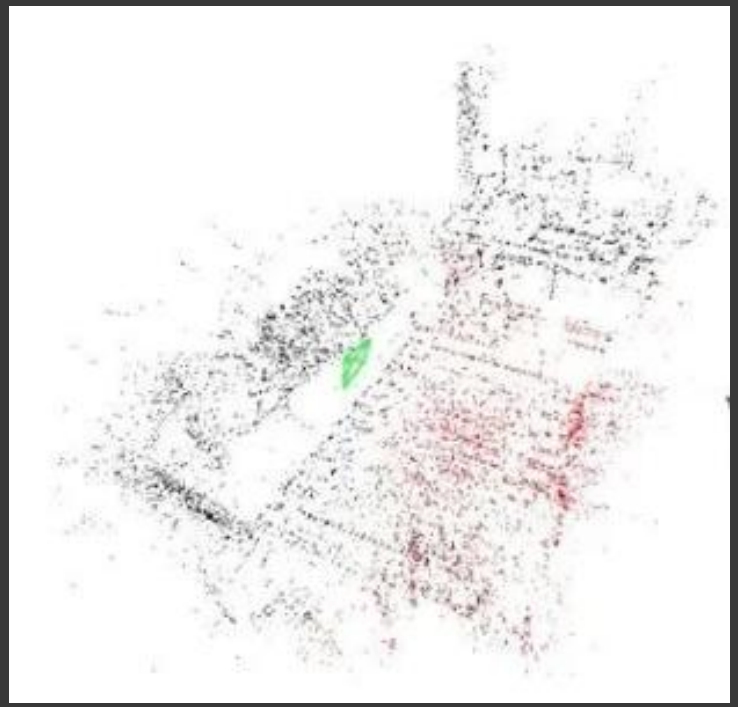
视觉SLAM算法有哪些？

- 按主传感器分类
  - 单目（鱼眼/全景）
  - 双目（三目/四目/...）
  - RGB-D
- 按图像分析方法分类
  - 特征点法
  - 直接法
  - 半直接法



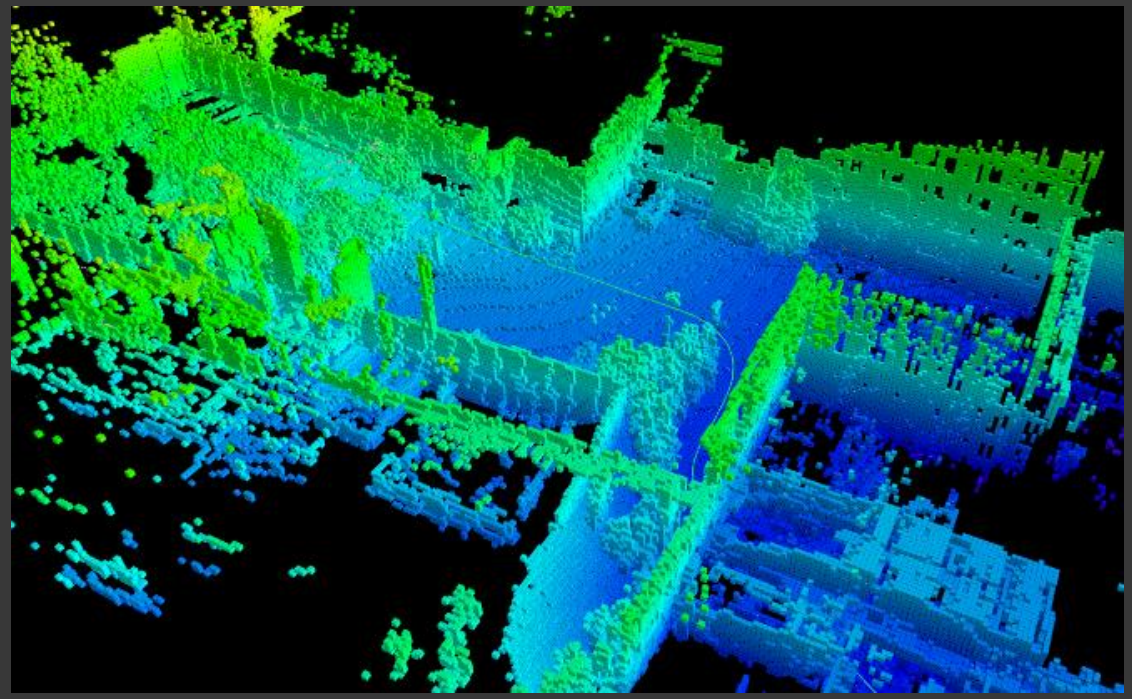
视觉SLAM算法有哪些？

- 按主传感器分类
  - 单目（鱼眼/全景）
  - 双目（三目/四目/...）
  - RGB-D
- 按图像分析方法分类
  - 特征点法
  - 直接法
  - 半直接法
- 按地图分类
  - 稀疏
  - 稠密
  - 半稠密



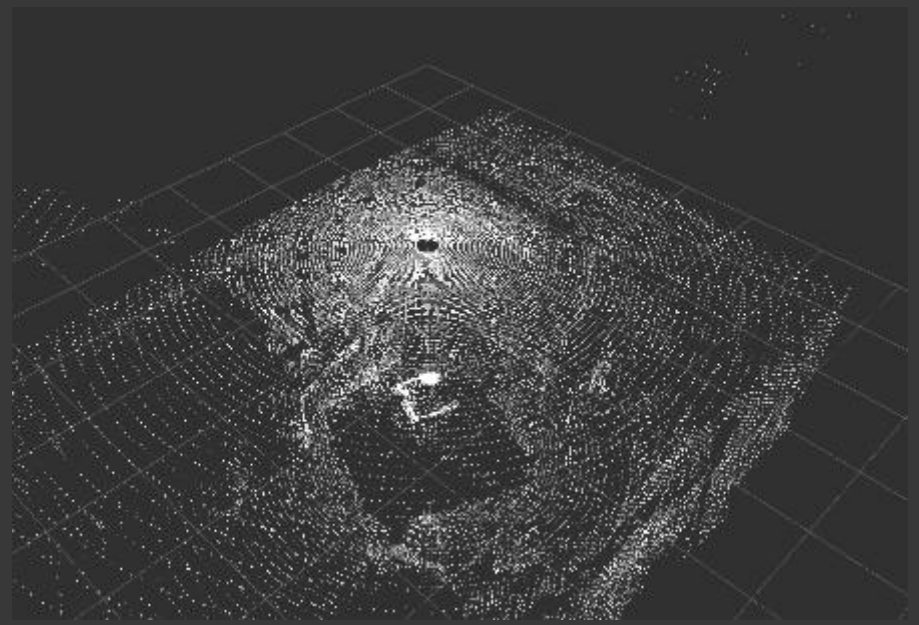
视觉SLAM算法有哪些？

- 按主传感器分类
  - 单目（鱼眼/全景）
  - 双目（三目/四目/...）
  - RGB-D
- 按图像分析方法分类
  - 特征点法
  - 直接法
  - 半直接法
- 按地图分类
  - 稀疏
  - 稠密
  - 半稠密



视觉SLAM算法有哪些？

- 按主传感器分类
  - 单目（鱼眼/全景）
  - 双目（三目/四目/...）
  - RGB-D
- 按图像分析方法分类
  - 特征点法
  - 直接法
  - 半直接法
- 按地图分类
  - 稀疏
  - 稠密
  - 半稠密



视觉SLAM算法有哪些？

- 按主传感器分类
  - 单目（鱼眼/全景）
  - 双目（三目/四目/...）
  - RGB-D
- 按图像分析方法分类
  - 特征点法
  - 直接法
  - 半直接法
- 按地图分类
  - 稀疏
  - 稠密
  - 半稠密
- 按求解方法分类
  - 滤波方法
  - 优化方法

$$
\begin{cases}
\left.\begin{aligned} &①\ \overline{\boldsymbol{\mu}_k} = g(\boldsymbol{\mu}_{k-1}, \boldsymbol{u}_k) \\ &②\ \overline{\boldsymbol{\Sigma}_k} = \boldsymbol{G}_k \boldsymbol{\Sigma}_{k-1} \boldsymbol{G}_k^{\mathrm{T}} + \boldsymbol{R}_k \end{aligned}\right\} \text{运动预测} \\
\left.\begin{aligned} &③\ \boldsymbol{K}_k = \overline{\boldsymbol{\Sigma}_k} \boldsymbol{H}_k^{\mathrm{T}} (\boldsymbol{H}_k \overline{\boldsymbol{\Sigma}_k} \boldsymbol{H}_k^{\mathrm{T}} + \boldsymbol{Q}_k)^{-1} \end{aligned}\right\} \text{卡尔曼增益} \\
\left.\begin{aligned} &④\ \boldsymbol{\mu}_k = \overline{\boldsymbol{\mu}_k} + \boldsymbol{K}_k (\boldsymbol{z}_k - h(\overline{\boldsymbol{\mu}_k})) \\ &⑤\ \boldsymbol{\Sigma}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \overline{\boldsymbol{\Sigma}_k} \end{aligned}\right\} \text{观测更新}
\end{cases}
$$

* 关于求解SLAM问题的滤波方法和优化方法的具体数学推导，请阅读书本第7章的内容

视觉SLAM算法有哪些？

- 按主传感器分类
  - 单目（鱼眼/全景）
  - 双目（三目/四目/...）
  - RGB-D
- 按图像分析方法分类
  - 特征点法
  - 直接法
  - 半直接法
- 按地图分类
  - 稀疏
  - 稠密
  - 半稠密
- 按求解方法分类
  - 滤波方法
  - 优化方法

$$
\begin{aligned}
\boldsymbol{S}_{\mathrm{MAP}} &= \arg\max_{\boldsymbol{S}} \psi(\boldsymbol{S}) \\
&\propto \arg\max_{\boldsymbol{S}} \Big(\sum_{i1} \ln \psi_{i1}(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}) + \sum_{i2} \ln \psi_{i2}(\boldsymbol{x}_k, \boldsymbol{m}_j)\Big) \\
&\propto \arg\max_{\boldsymbol{S}} \Big(-\frac{1}{2}\sum_{i1} (\boldsymbol{x}_k - g(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k))^{\mathrm{T}} \boldsymbol{\Sigma}_{\boldsymbol{R}_k}^{-1} (\boldsymbol{x}_k - g(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k)) \\
&\qquad -\frac{1}{2}\sum_{i2} (z_k - h(\boldsymbol{x}_k, \boldsymbol{m}_j))^{\mathrm{T}} \boldsymbol{\Sigma}_{\boldsymbol{Q}_k}^{-1} (z_k - h(\boldsymbol{x}_k, \boldsymbol{m}_j))\Big) \\
&\propto \arg\min_{\boldsymbol{S}} \Big(\sum_{i1} \left\| \boldsymbol{x}_k - g(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) \right\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{R}_k}^{-1}} + \sum_{i2} \left\| z_k - h(\boldsymbol{x}_k, \boldsymbol{m}_j) \right\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{Q}_k}^{-1}}\Big)
\end{aligned}
$$

* 关于求解SLAM问题的滤波方法和优化方法的具体数学推导，请阅读书本第7章的内容

视觉SLAM算法有哪些？

	算法名称	传感器	前端	后端	闭环	地图	时间
特征点法	MonoSLAM	Mono	数据关联 & VO	EKF 滤波	-	稀疏	2007
	PTAM	Mono		优化	-	稀疏	2007
	ORB-SLAM2	Mono Stereo RGB-D			BOW	稀疏	2016
直接法	DTAM	Mono			-	稠密	2011
	LSD-SLAM	Mono Stereo Omni			FABMAP	半稠密	2014
	DSO	Mono			-	稀疏	2016
半直接法	SVO	Mono		-	-	稀疏	2014

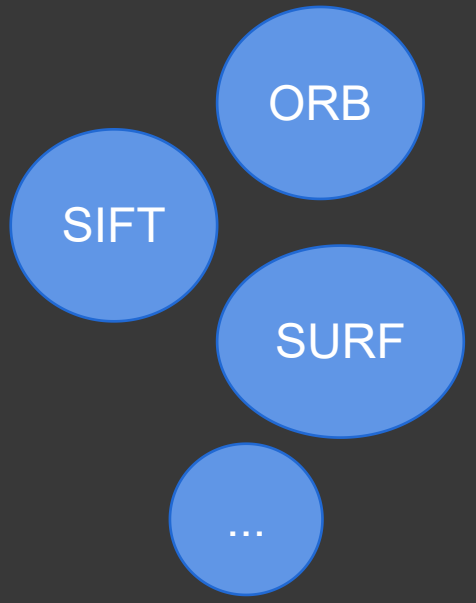
来源于：书《机器人SLAM导航：核心技术与实战》表9-1

内容概要

9.1 ORB-SLAM2算法

ORB-SLAM2究竟是什么？

一种图像特征



同时定位与建图

(Simultaneous Localization and Mapping)

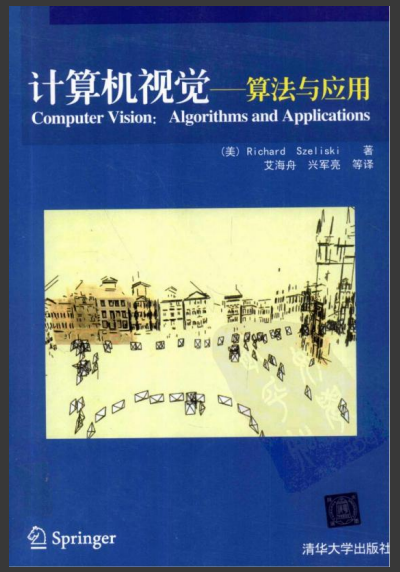
版本

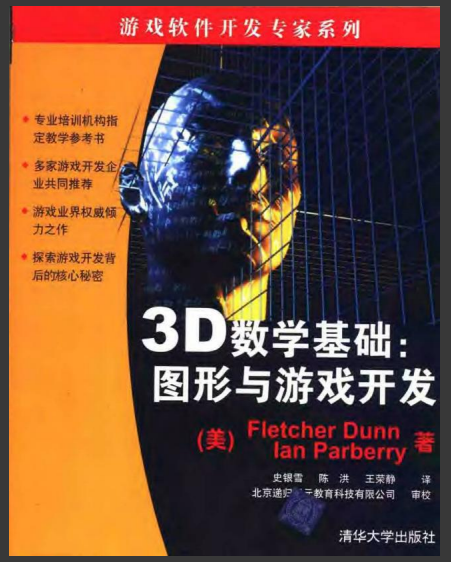
ORB-SLAM	单目
ORB-SLAM2	单目/双目/RGB-D
ORB-SLAM3	单目/双目/RGB-D + IMU +Multi-Map

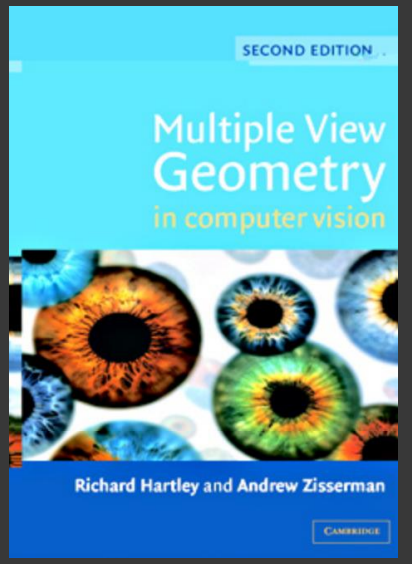
* 关于图像特征的具体讲解，请阅读书本第3章的内容

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
  - 特征点法
  - 三维空间运动
  - 多视图几何
  - ORB-SLAM2系统框架
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行







特征点法、三维空间运动、多视图几何，这些都很难很复杂，为什么不放在单独的章节分别讨论呢？

① 仅部分内容，有工程应用价值

② 结合实际算法框架讲解，更容易理解

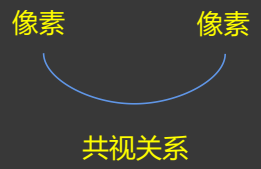
③ 篇幅限制

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析 → 特征点法、三维空间运动、多视图几何、ORB-SLAM2系统框架
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

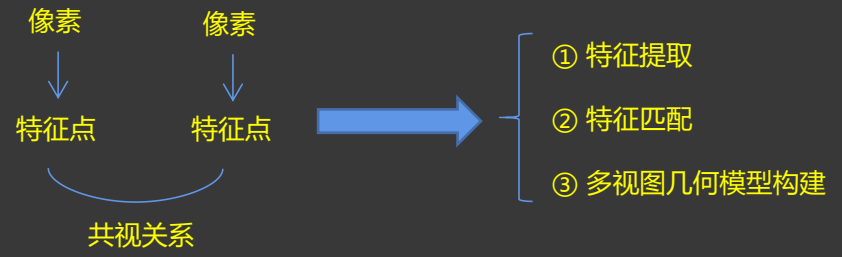
SLAM就是求解运动物体（比如机器人、无人机、相机等）的位姿和环境中路标点（也就是环境地图点）的问题。当相机从不同的角度拍摄同一个物体时可以得到不同的图像，而这些图像中具有很多相同的信息，这就构成了**共视关系**。

直接法：



特征点法：

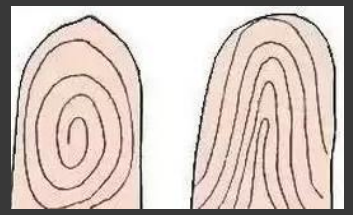
（间接法）



9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析 → 特征点法、三维空间运动、多视图几何、ORB-SLAM2系统框架
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征? → 特征描述? → 特征提取? → 特征匹配?



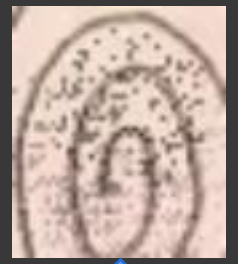
1 3 5 7 9 ...
2 4 6 8 10 ...

严重秃头，黑色，细，...
轻微秃头，黑色，粗，...

旋转纹，间距大，...
层叠纹，间距小，...

奇数，正数，...
偶数，正数，...

信息采集与编码
数据建模与运算



?

9.1 ORB-SLAM2算法

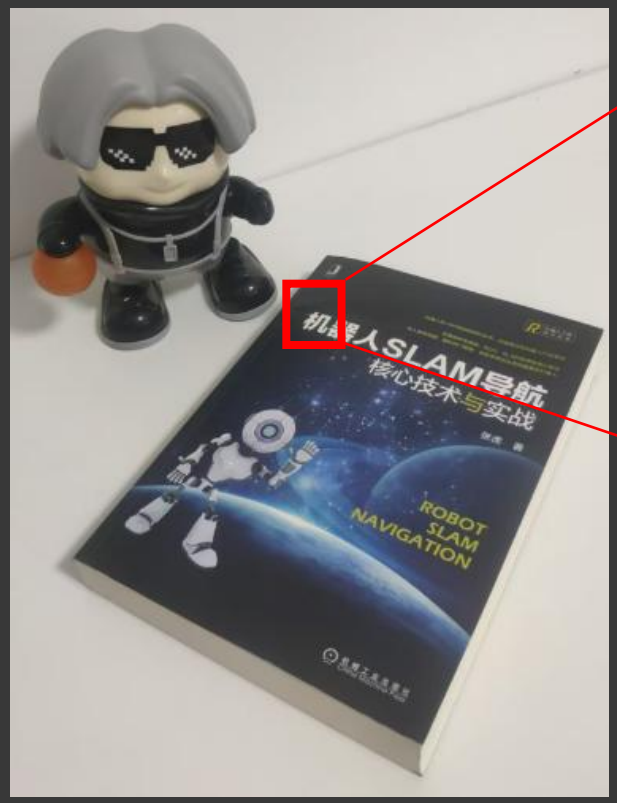
- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

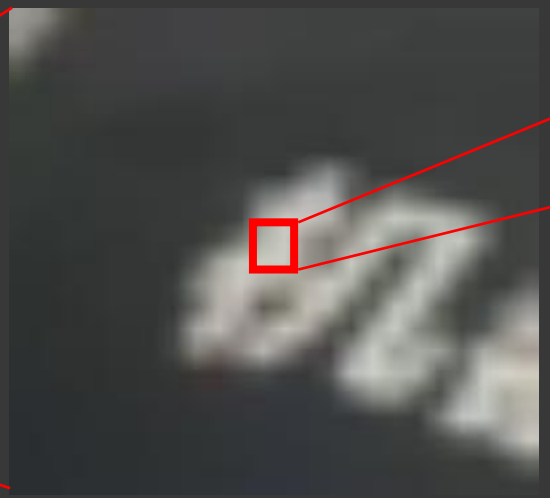
特征点是图像的区域结构信息

- 纹理
- 灰度统计
- 频谱
- 小波变换
- ...

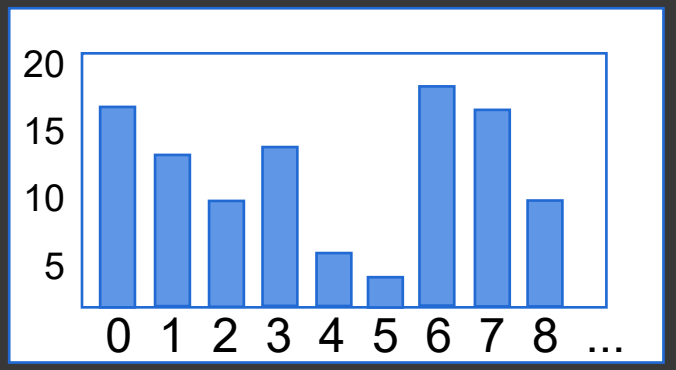
- SIFT
- SURF
- ORB
- ...



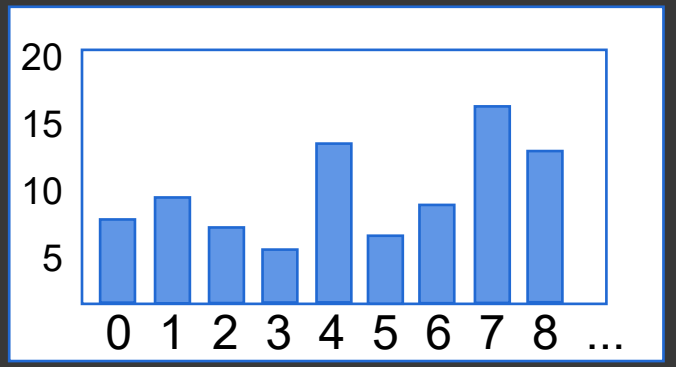
区域像素



单个像素



特征向量 [17,14,9,15,5,3,18,17,10]



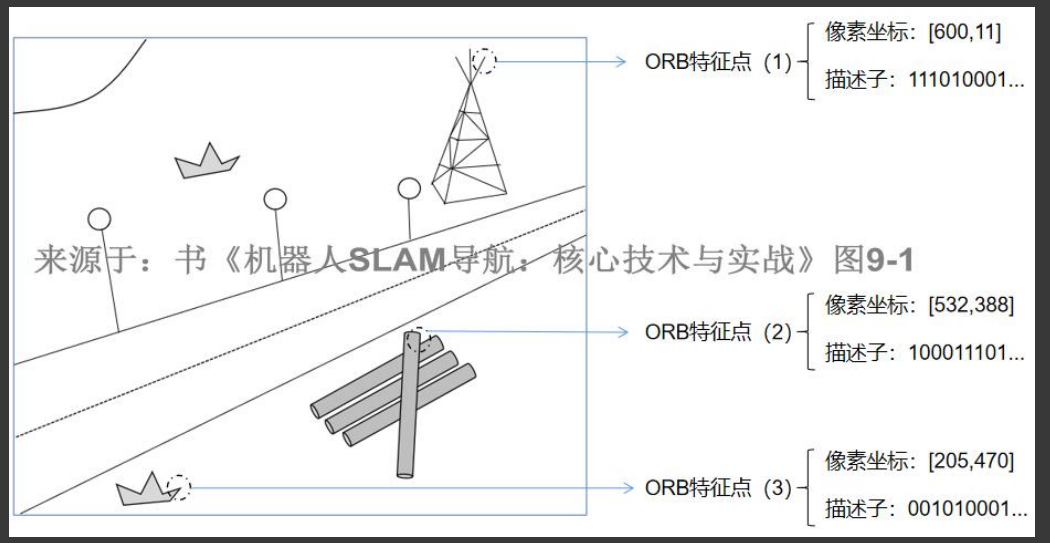
特征向量 [8,10,7,5,14,6,9,16,14]

9.1 ORB-SLAM2算法

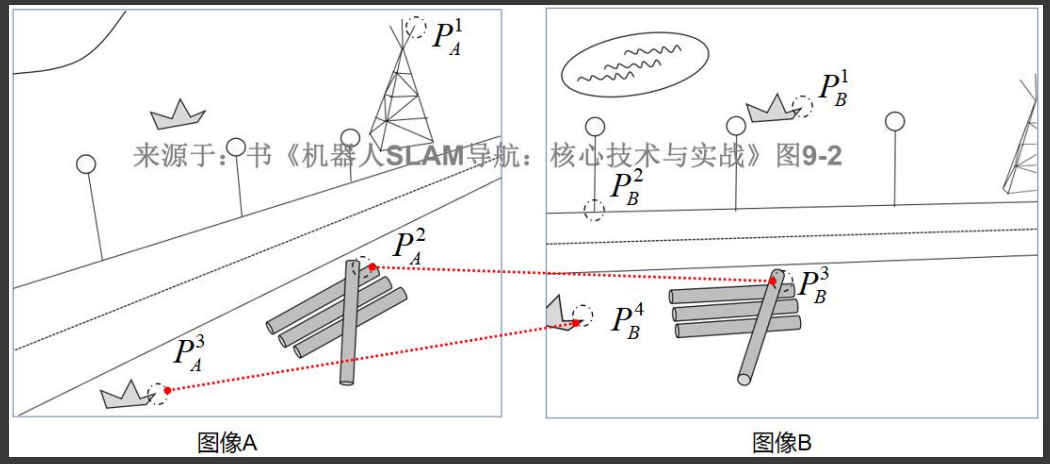
- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

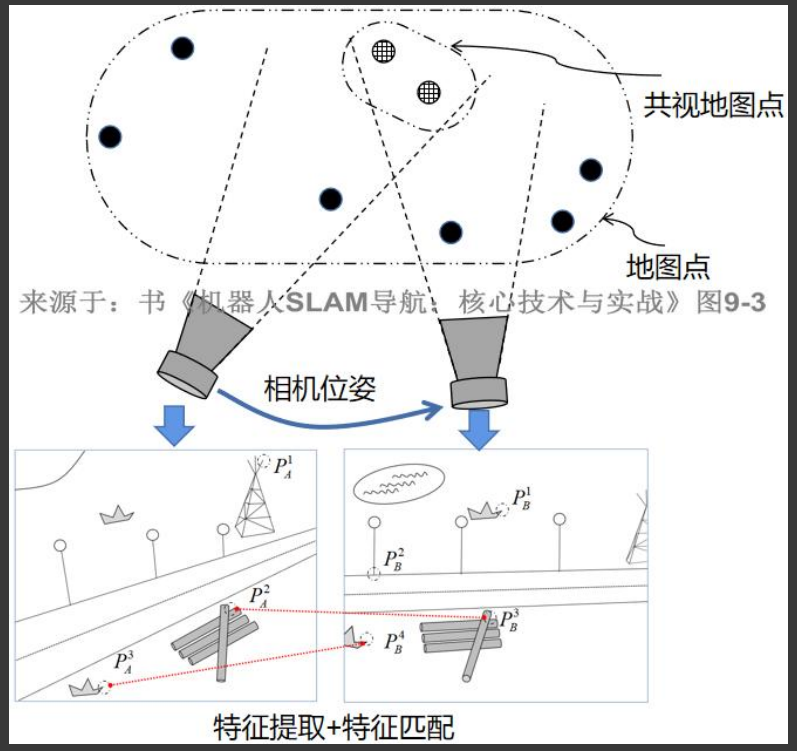
① 特征提取：特征点像素坐标，特征点描述子



② 特征匹配：暴力匹配，KNN，FLANN



③ 多视图几何模型构建：相机位姿（位置+姿态）



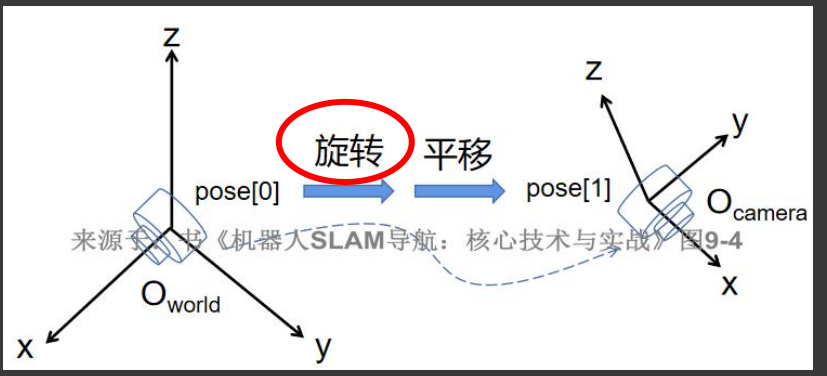
* 关于图像特征的具体讲解，请阅读书本第3章的内容

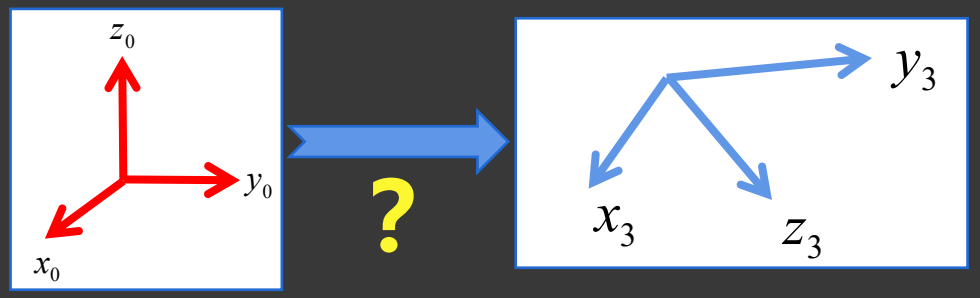
9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

欧拉角、旋转矩阵、四元数
转移矩阵
李群、李代数

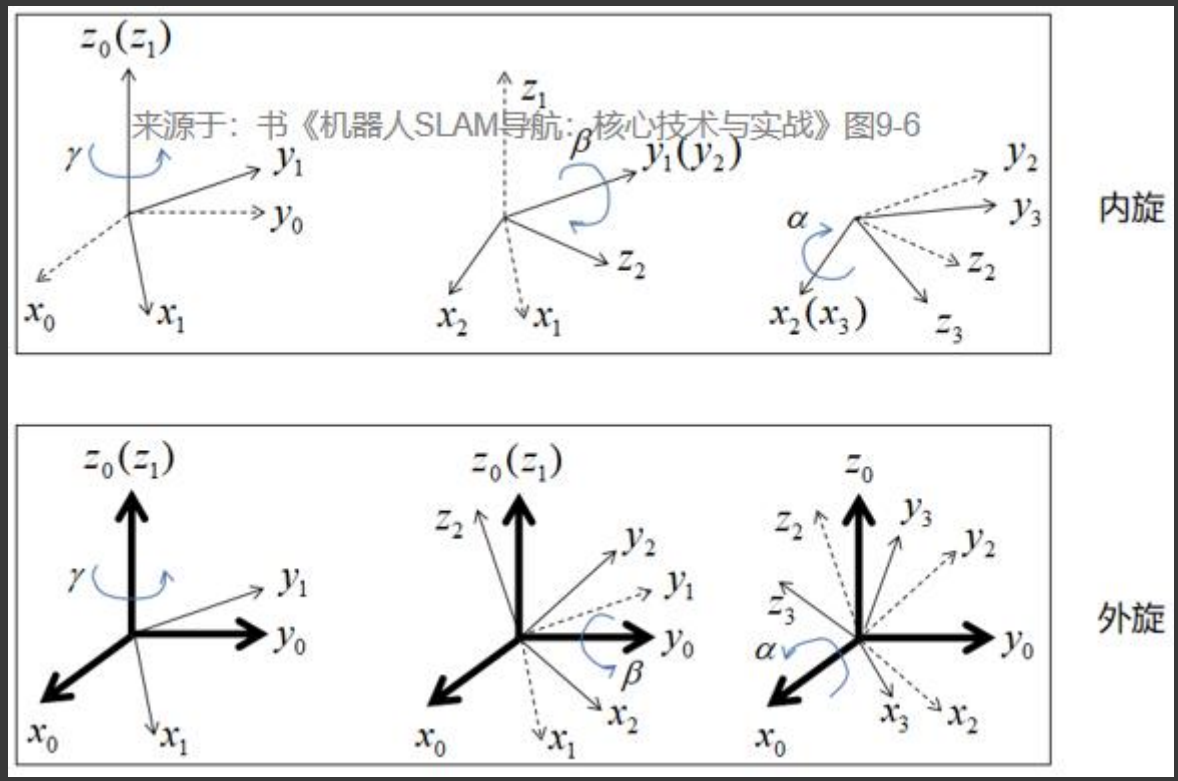




统一规则
（将绕某轴一次旋转，分解为依次绕坐标轴旋转）

欧拉角：

- 左右手坐标系（一般右手坐标系）
- 内外旋方式（动态/静态）
- 旋转顺序（2个轴6种，3个轴6种）



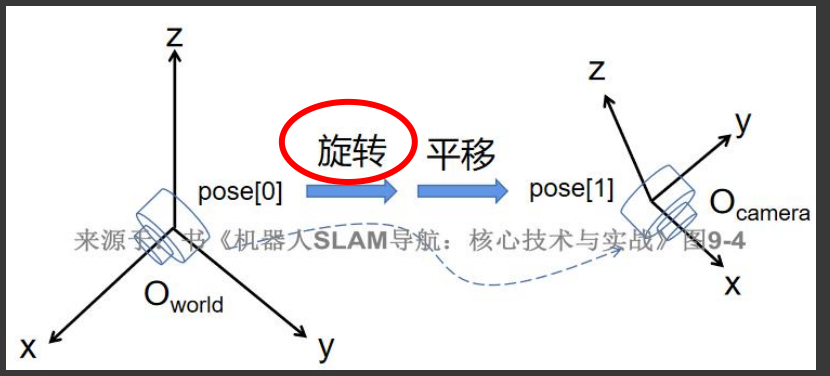
给定一组欧拉角(α，β，γ)，必须指明旋转方式和旋转顺序，才有意义

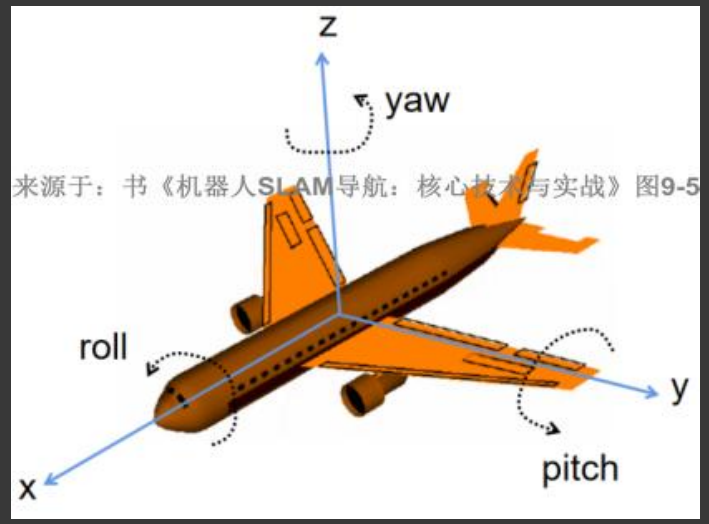
9.1 ORB-SLAM2算法

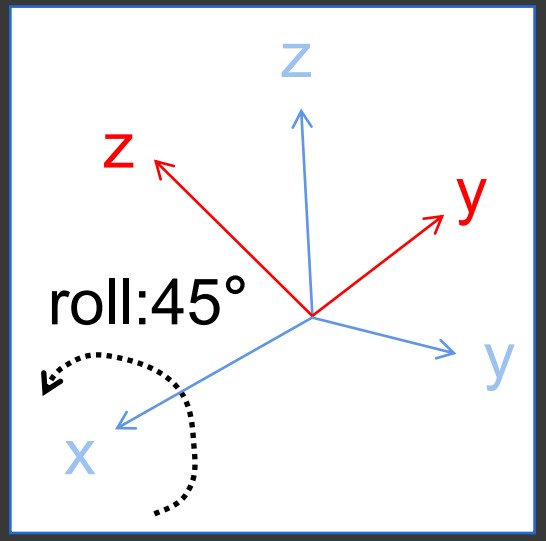
- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

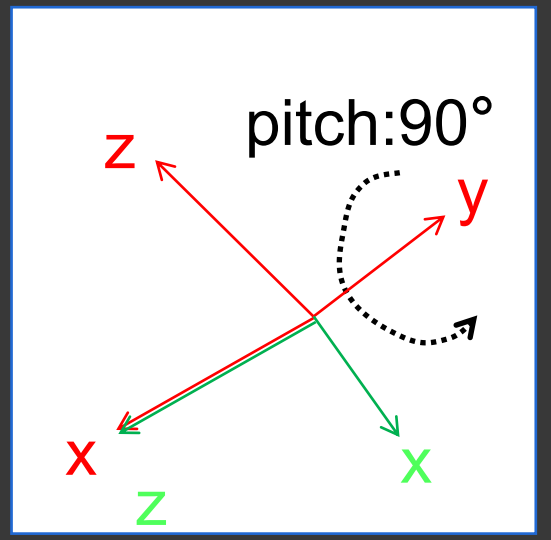
特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

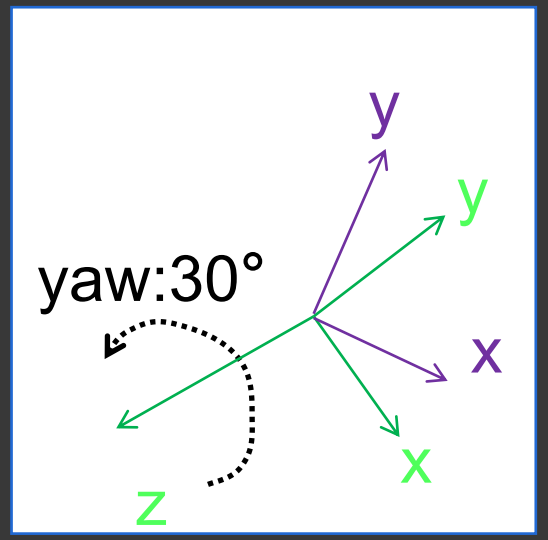
欧拉角、旋转矩阵、四元数
转移矩阵
李群、李代数











$(roll,pitch,yaw)_{in\_x\text{-}y\text{-}z} = (45°, 90°, 30°) = (75°, 90°, 0°)$

RPY-欧拉角
横滚角，俯仰角，航向角
(roll, pitch, yaw)

万向节死锁问题

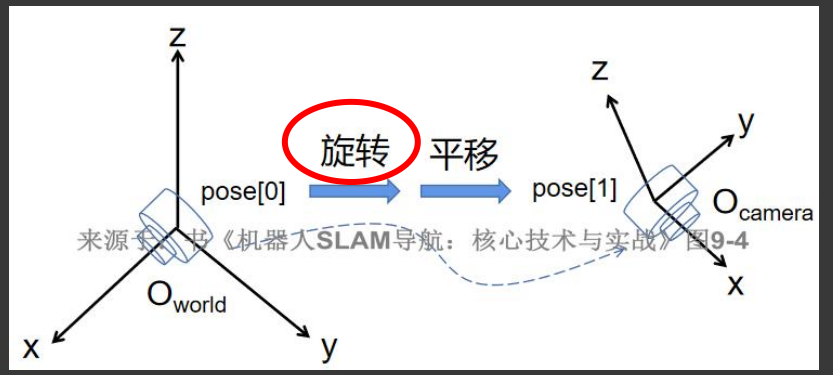
- 误区纠正：并不是说欧拉角被锁住而失去表示的完备性，万向节是欧拉角规则引入的某种虚拟限制，欧拉角表示任意的某个姿态完全没有问题
- 影响分析：只适合直观表示某个姿态，很难用于数学计算，插值或增量过程可能不平坦，在突破万向节锁的过程中雅克比矩阵趋近于无穷造成速度奇异

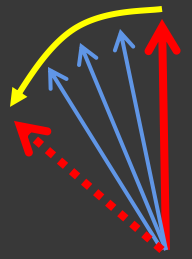
9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

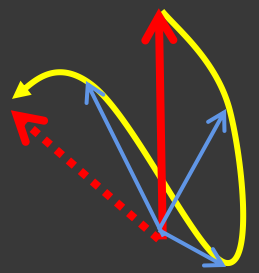
欧拉角、旋转矩阵、四元数
转移矩阵
李群、李代数



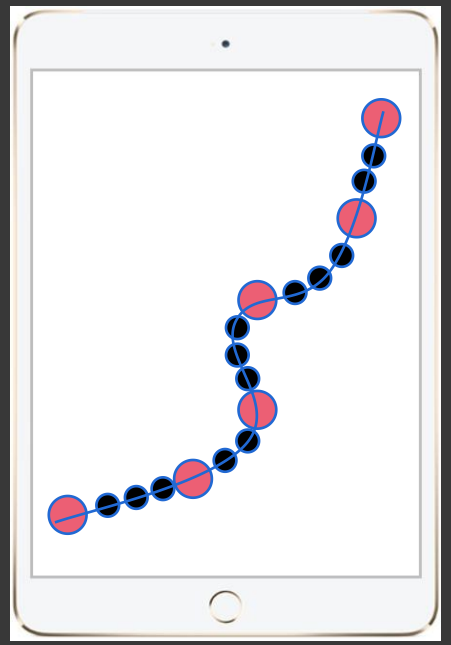


理想插值过程
平坦

(0°, 20°, 10°)
(1°, 22°, 14°)
(2°, 24°, 16°)
(3°, 26°, 20°)
...
(15°, 50°, 60°)

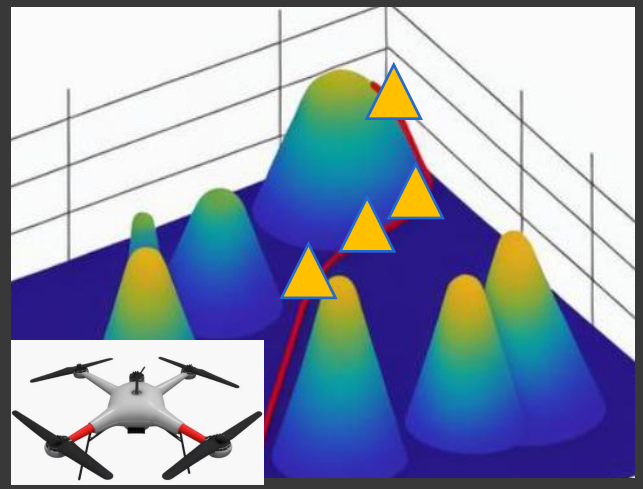


实际插值过程
歪歪扭扭



触屏手写插值
美化书写轨迹

游戏动作渲染插值，让动作更连贯

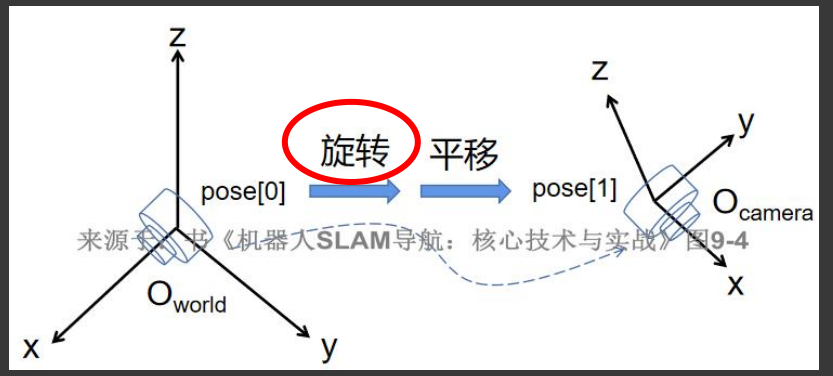


机器人轨迹跟踪插值，让运动控制更精细更稳定

9.1 ORB-SLAM2算法

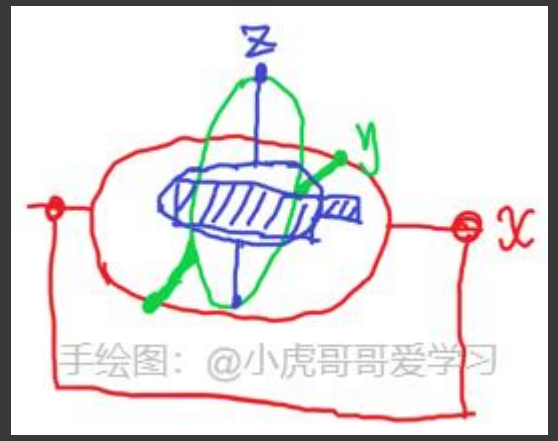
- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动 → 欧拉角、旋转矩阵、四元数
多视图几何 → 转移矩阵
ORB-SLAM2系统框架 → 李群、李代数



来源于书《机器人SLAM导航：核心技术与实战》图9-4

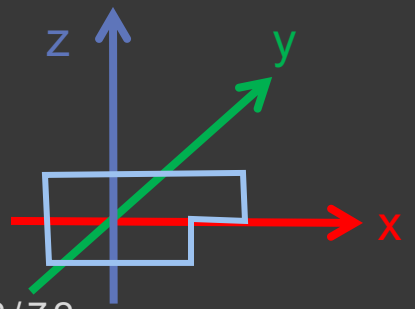
三维直角坐标系中坐标的三个分量互相独立；而欧拉角中姿态的三个分量相互影响。
就不应用三维直角坐标系的插值方式来给欧拉角做插值，或者说欧拉角根本不适合于插值计算的应用。
就像踩自行车在平衡点处会卡壳一样，欧拉角内置的某种限制在特定情况也会卡壳（万向节死锁）。



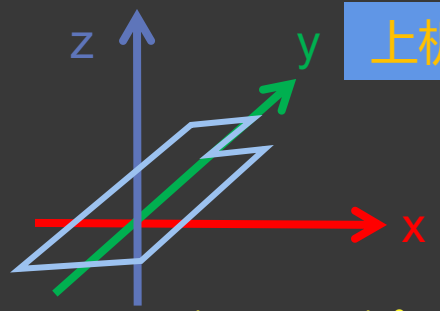
上机演示

上机演示

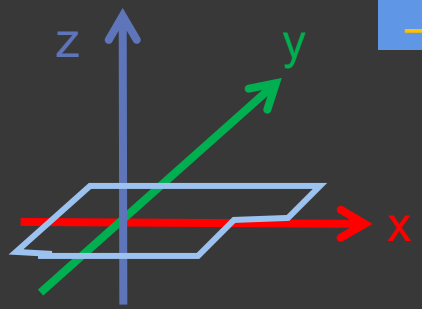
平衡点处会卡壳
使劲蹬抖动一下就可以正常转起来了



$(roll,pitch,yaw)_{in\_x\text{-}y\text{-}z} = (0°, 0°, 0°)$



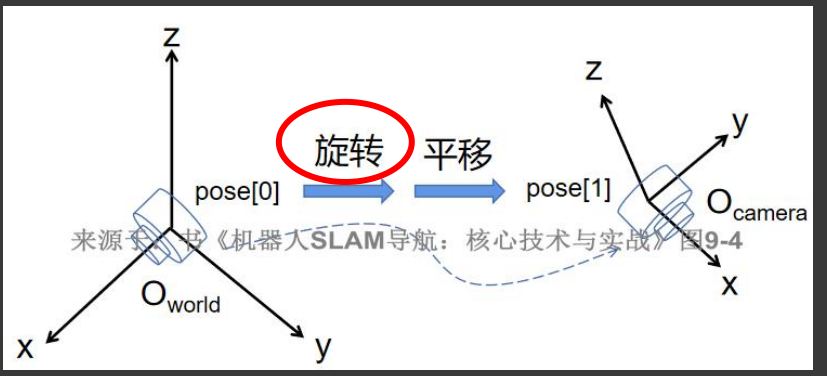
$(roll,pitch,yaw)_{in\_x\text{-}y\text{-}z} = (0°, -90°, 90°)$



$(roll,pitch,yaw)_{in\_x\text{-}y\text{-}z} = (-90°, 0°, 0°)$

9.1 ORB-SLAM2算法

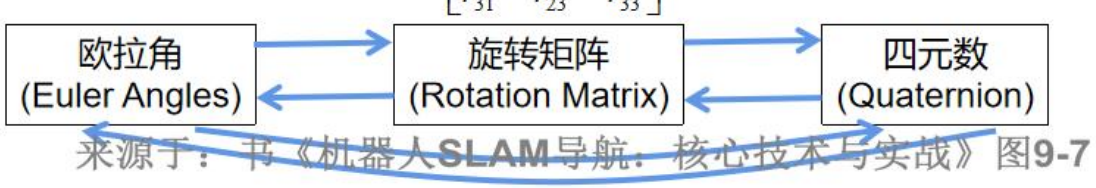
- ORB-SLAM2原理分析 → 特征点法、三维空间运动、多视图几何、ORB-SLAM2系统框架 → 欧拉角、旋转矩阵、四元数、转移矩阵、李群、李代数
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行



来源于：书《机器人SLAM导航：核心技术与实战》图9-4

- 欧拉角：左右手坐标系，内外旋，旋转顺序
- 旋转矩阵：内旋是右乘、外旋是左乘
- 四元数：JPL四元数、Hamilton四元数

$$\text{roll, pitch, yaw} \qquad \boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{23} & r_{33} \end{bmatrix} \qquad q = q_0 + q_1 \cdot i + q_2 \cdot j + q_3 \cdot k$$



来源于：书《机器人SLAM导航：核心技术与实战》图9-7

	欧拉角	旋转矩阵	四元数
在坐标系间旋转点或向量	不能	能	不能
增量迭代计算	不能	能	能
插值	能（有万向锁等问题）	基本不能	能
易用性	易	难	难
变量个数	3个	9个	4个
表达唯一性	否	是	不是（可互相为负）
内在约束	无（3个量可任意取值）	有	有

来源于：书《机器人SLAM导航：核心技术与实战》表9-2

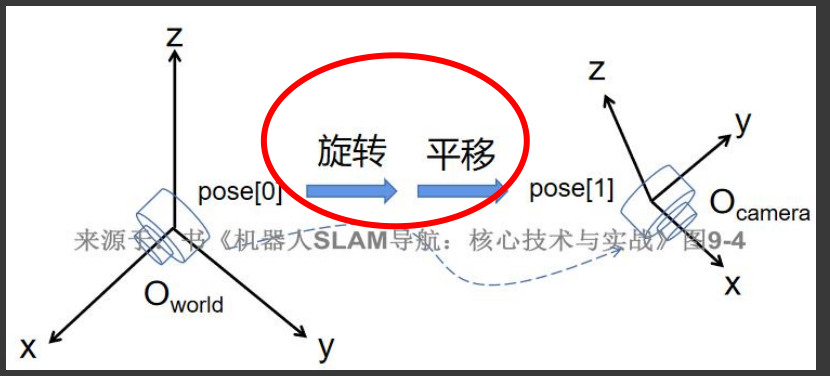
* 关于欧拉角、旋转矩阵和四元数的具体计算过程以及转换关系推导，请看书本第9章内容。公式详细推导，将在第2季课程中讨论。

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

欧拉角、旋转矩阵、四元数
转移矩阵
李群、李代数



$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \bullet \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \text{简写：} \boldsymbol{P}_w = \boldsymbol{R} \bullet \boldsymbol{P}_c + \boldsymbol{t}$$

转移矩阵T:

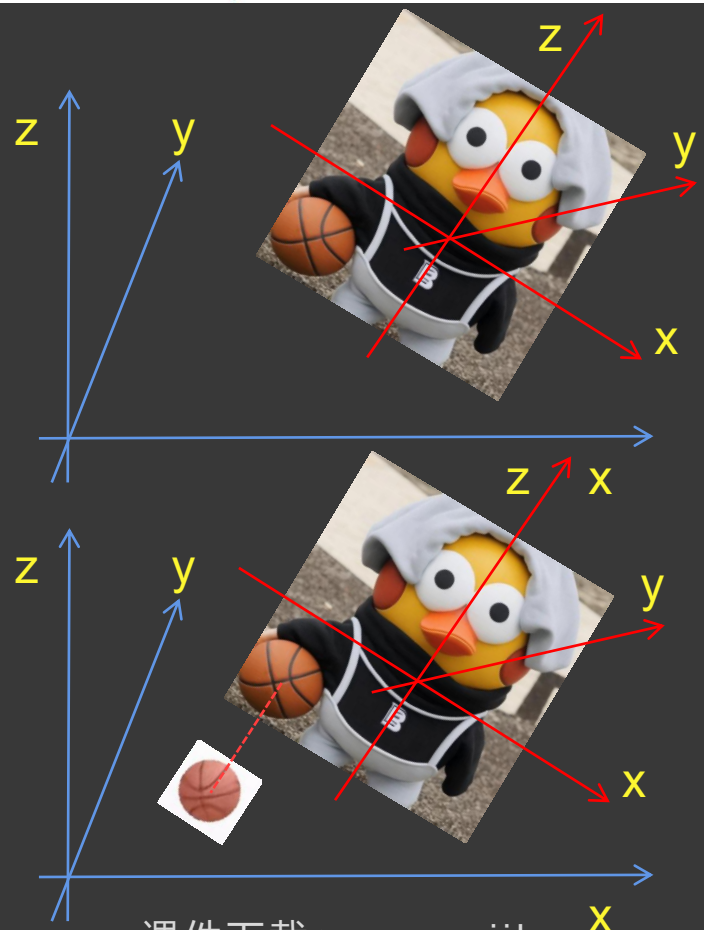
$$\text{设：} \boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}, \text{简写：} \boldsymbol{P}_w = \boldsymbol{T} \bullet \boldsymbol{P}_c$$

转移矩阵T有什么用？

① 描述物体在三维空间的位姿（位置+姿态）

② 在不同坐标系之间进行坐标换算



9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

欧拉角、旋转矩阵、四元数
转移矩阵
李群、李代数

旋转矩阵内的各元素取值受到内在约束，即旋转矩阵为正交且旋转矩阵的行列式为1。在采用优化方法进行SLAM位姿估计时，这种内在约束让优化求解变得非常困难。

- 群：一种代数结构，包含一个集合(A)和一种运算(.)
- 李群：指具有连续（光滑）性质的群
- 李代数：与李群相对应，李代数由一个集合、一个数域和一个二元运算组成

①封闭性：$\forall a_1, a_2 \in A,\ a_1 \bullet a_2 \in A$
②结合律：$\forall a_1, a_2, a_3 \in A,\ (a_1 \bullet a_2) \bullet a_3 = a_1 \bullet (a_2 \bullet a_3)$
③幺元：$\exists a_0 \in A,\ s.t.\ \forall a \in A,\ a_0 \bullet a = a \bullet a_0 = a$
④逆：$\forall a \in A,\ \exists a^{-1} \in A,\ s.t.\ a \bullet a^{-1} = a_0$

$\boldsymbol{R}_1 + \boldsymbol{R}_2 \notin SO(3)$　　$\boldsymbol{T}_1 + \boldsymbol{T}_2 \notin SE(3)$

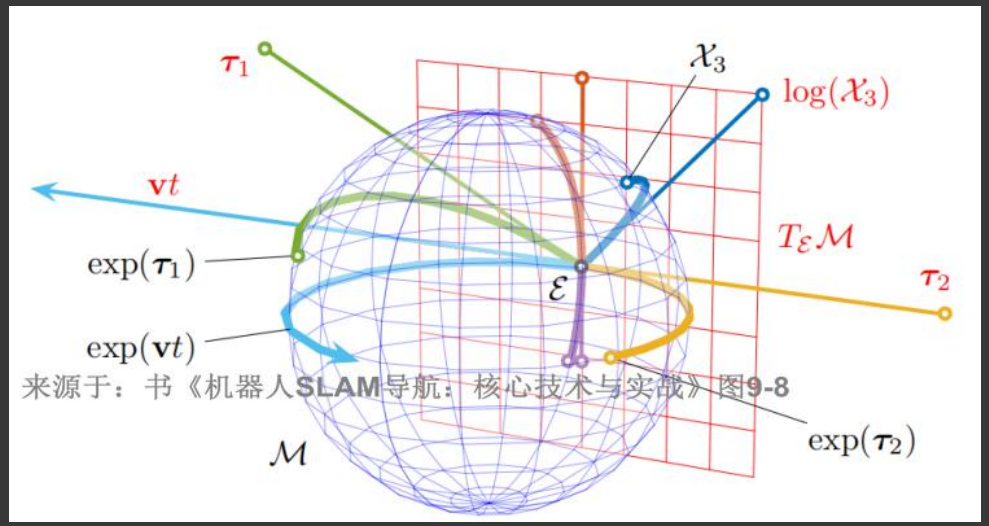
$\boldsymbol{R}_1 \boldsymbol{R}_2 \in SO(3)$　　$\boldsymbol{T}_1 \boldsymbol{T}_2 \in SE(3)$

Lie groups are named after Norwegian mathematician Sophus Lie

李代数描述了李群的局部性质，每个李群都有相对应的李代数。

李代数对应李群的切线空间，它描述了李群局部的导数。

李群是具有群结构的流形，李代数是李群上单位元处保留李括号运算的切空间。



来源于：书《机器人SLAM导航：核心技术与实战》图9-8

- 如果把球面 M 看成李群的流形
- 那么平面 $T_\varepsilon M$ 就是流形上点 ε 附近李代数的切空间
- 流形：就是高维空间的超曲面
- 切空间：与流形上某点的相切面

	李群		李代数	
旋转矩阵	$SO(2)$	$SO(3)$	$so(2)$	$so(3)$
转移矩阵	$SE(2)$	$SE(3)$	$se(2)$	$se(3)$
相似变换	$Sim(2)$	$Sim(3)$	$sim(2)$	$sim(3)$

李群李代数开源库：
- https://github.com/strasdat/Sophus
- https://github.com/artivis/manif

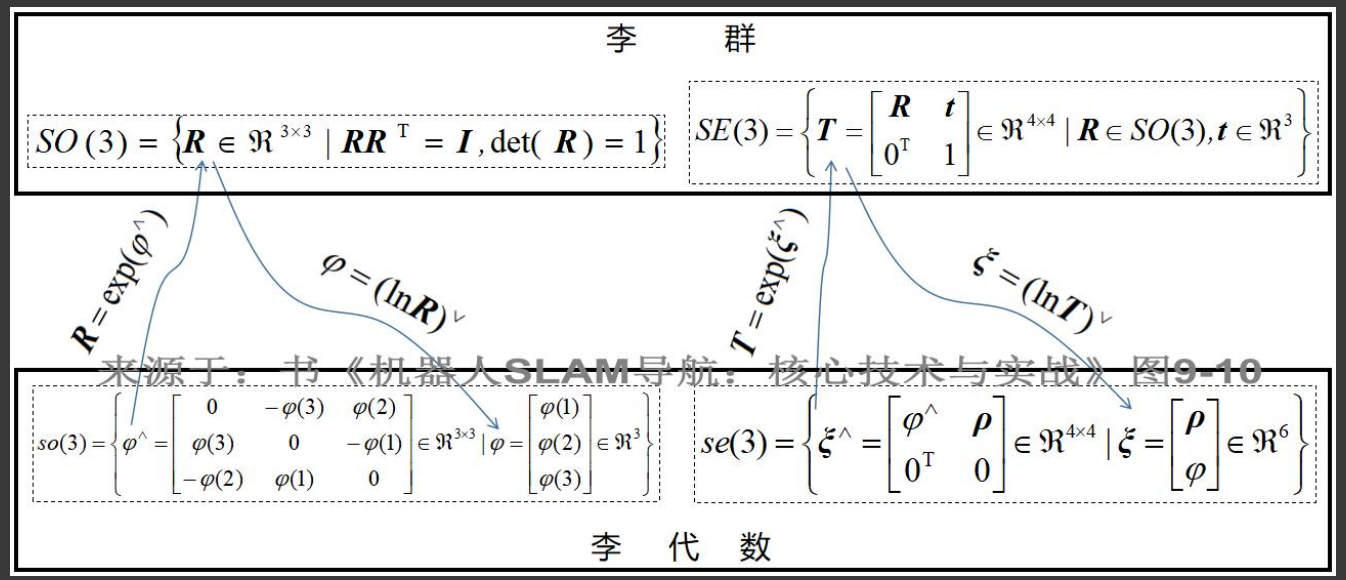
9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

欧拉角、旋转矩阵、四元数
转移矩阵
李群、李代数

旋转矩阵内的各元素取值受到内在约束，即旋转矩阵为正交且旋转矩阵的行列式为1。在采用优化方法进行SLAM位姿估计时，这种内在约束让优化求解变得非常困难。



来源于：书《机器人SLAM导航：核心技术与实战》图9-10

$$
\begin{aligned}
S_{\mathrm{MAP}} &= \arg\max_{S}\psi(S)\\
&\propto \arg\max_{S}\Big(\sum_{i1}\ln\psi_{i1}(x_k,x_{k-1})+\sum_{i2}\ln\psi_{i2}(x_k,m_j)\Big)\\
&\propto \arg\max_{S}\Big(-\frac{1}{2}\sum_{i1}(x_k-g(x_{k-1},u_k))^{\mathrm{T}}\Sigma_{R_k}^{-1}(x_k-g(x_{k-1},u_k))\\
&\qquad -\frac{1}{2}\sum_{i2}(z_k-h(x_k,m_j))^{\mathrm{T}}\Sigma_{Q_k}^{-1}(z_k-h(x_k,m_j))\Big)\\
&\propto \arg\min_{S}\Big(\sum_{i1}\left\|x_k-g(x_{k-1},u_k)\right\|^2_{\Sigma_{R_k}^{-1}}+\sum_{i2}\left\|z_k-h(x_k,m_j)\right\|^2_{\Sigma_{Q_k}^{-1}}\Big)
\end{aligned}
$$

待估量的优化迭代策略：

【方法1：求导】

①将待估量从李群变换到李代数

②计算左雅克比矩阵

③梯度下降

【方法2：扰动】

①在带估量上施加一个扰动

②求能使目标函数下降的扰动量的李代数

③利用该扰动进行优化迭代

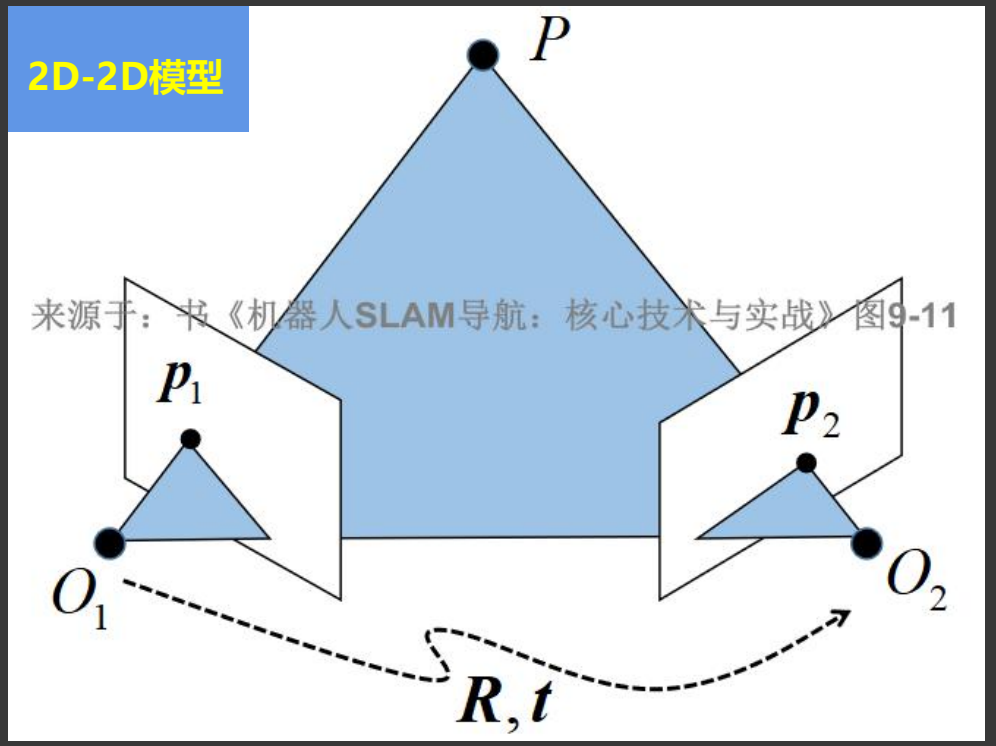
9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动 按照模型中给定已知条件的不同
多视图几何
ORB-SLAM2系统框架

2D-2D模型
3D-2D模型
3D-3D模型

2D-2D模型



来源于：书《机器人SLAM导航：核心技术与实战》图9-11

$$p_1 = \frac{1}{z_1} K \bullet P_{O_1}$$

$$p_2 = \frac{1}{z_2} K \bullet P_{O_2}$$

$$P_{O_2} = R \bullet P_{O_1} + t$$

$$z_2 K^{-1} p_2 = R \bullet (z_1 K^{-1} p_1) + t$$

$$\Updownarrow \text{左叉乘} t, t \times t = 0$$

$$t \times z_2 K^{-1} p_2 = t \times R \bullet (z_1 K^{-1} p_1)$$

$$\Updownarrow \text{左乘} (K^{-1} p_2)^{\mathrm{T}}, (K^{-1} p_2)^{\mathrm{T}} \bullet (t \times (K^{-1} p_2)) = 0$$

$$0 = z_1 (K^{-1} p_2)^{\mathrm{T}} t \times R K^{-1} p_1$$

$$= p_2^{\mathrm{T}} K^{-\mathrm{T}} \underbrace{t^{\wedge} R}_{E} K^{-1} p_1$$

$$= p_2^{\mathrm{T}} \underbrace{K^{-\mathrm{T}} E K^{-1}}_{F} p_1$$

$$= p_2^{\mathrm{T}} F p_1$$

对极约束

$$0 = p_2^{\mathrm{T}} K^{-\mathrm{T}} E K^{-1} p_1 = p_2^{\mathrm{T}} F p_1$$

尺度不确定性

- 本质矩阵E与基础矩阵F
- 单应矩阵H
- 三角化重建地图点
- BA优化

Bundle Adjustment

捆集优化
光束平差

物体光束投影 最小二乘项 重投影误差

$$T, P_{O_1} = \arg\min_{T, P_{O_1}} \sum_i \|e_i\|^2$$

$$T = \arg\min_{T} \sum_i \|e_i\|^2$$

$$P_{O_1} = \arg\min_{P_{O_1}} \sum_i \|e_i\|^2$$

$$e_i = p_2^i - \frac{1}{z_2} K \bullet T \bullet P_{O1}^i, \quad \text{其中} T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

思考：已知量是什么，待求量是什么

* 2D-2D模型的应用场景：比如，在单目SLAM初始化新地图点时。

9.1 ORB-SLAM2算法

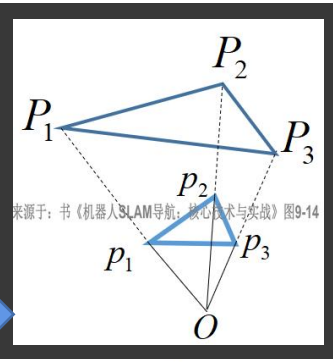
- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动 按照模型中给定已知条件的不同
多视图几何
ORB-SLAM2系统框架

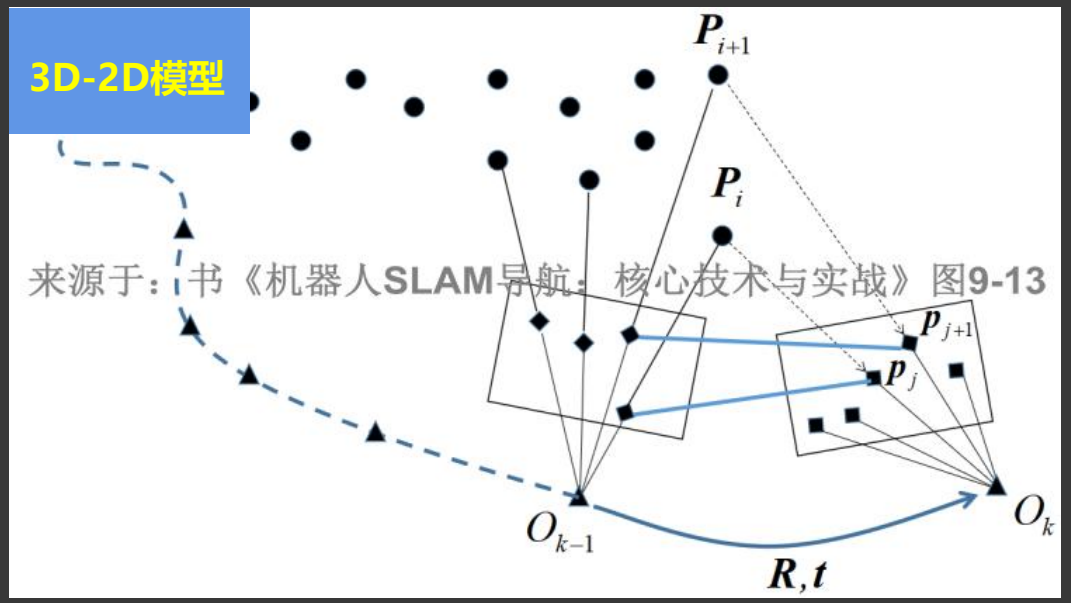
2D-2D模型
3D-2D模型
3D-3D模型

$$z_i \boldsymbol{p}_i = \boldsymbol{K}[\boldsymbol{R}\,|\,\boldsymbol{t}]\boldsymbol{P}_i \Leftrightarrow z_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \Leftrightarrow \boldsymbol{A}_{2\times 12} \bullet \boldsymbol{m}_{12\times 1} = \boldsymbol{0}$$

$$\begin{aligned} \boldsymbol{m} &= \arg\min_{\boldsymbol{m}} \|\boldsymbol{A}\bullet\boldsymbol{m}\|^2 \\ &= \arg\min_{\boldsymbol{m}} (\boldsymbol{m}^{\mathrm{T}}\bullet\boldsymbol{A}^{\mathrm{T}}\bullet\boldsymbol{A}\bullet\boldsymbol{m}) \\ &\text{令}|\boldsymbol{m}|=1, SVD(A)=\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}=\sum_{i=1}^{12}\boldsymbol{s}_i\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}} \\ &= \arg\min_{\boldsymbol{m}} (\boldsymbol{m}^{\mathrm{T}}\bullet\boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^{\mathrm{T}}\bullet\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}\bullet\boldsymbol{m}) \\ &= \arg\min_{\boldsymbol{m}} (\boldsymbol{m}^{\mathrm{T}}\bullet\boldsymbol{V}\boldsymbol{S}^2\boldsymbol{V}^{\mathrm{T}}\bullet\boldsymbol{m}) \\ &= \arg\min_{\boldsymbol{m}} (\boldsymbol{m}^{\mathrm{T}}\bullet\sum_{i=1}^{12}\boldsymbol{s}_i^2\boldsymbol{v}_i\boldsymbol{v}_i^{\mathrm{T}}\bullet\boldsymbol{m}) \end{aligned}$$



3D-2D模型



- DLT

通过直接解线性方程的方法来求解PnP问题

- P3P

通过不共面的3对3D-2D点就能求出相机运动($\boldsymbol{R},\boldsymbol{t}$)

- EPnP

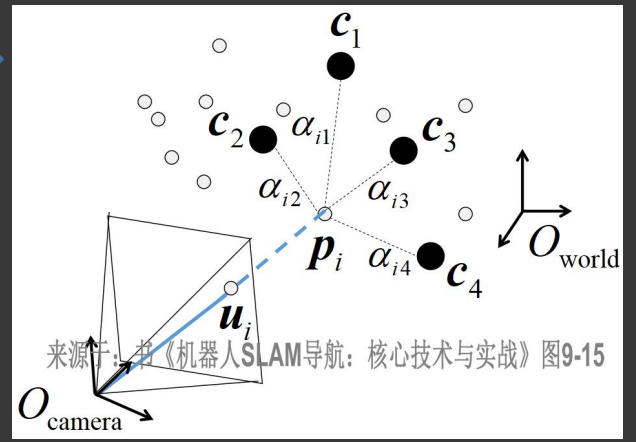
在求解PnP问题上表现更稳定更高效

- BA优化

$$\boldsymbol{T} = \arg\min_{\boldsymbol{T}} \sum_i \|\boldsymbol{e}_i\|^2$$

以解析解为初值，对相机位姿($\boldsymbol{R},\boldsymbol{t}$)进一步优化以提高精度

值得庆幸的是，在OpenCV中已经将求解PnP问题的一些常见方法封装到了函数solvePnP()：P3P、EPnP、BA等方法



思考：已知量是什么，待求量是什么

* 3D-2D模型的应用场景：比如，单目SLAM初始化完成后就能得到一些已知地图点，或者双目、RGB-D能直接给 出环境观测点。在这些情况下，当前帧图像所对应的相机位姿解算就可以用这种模型。

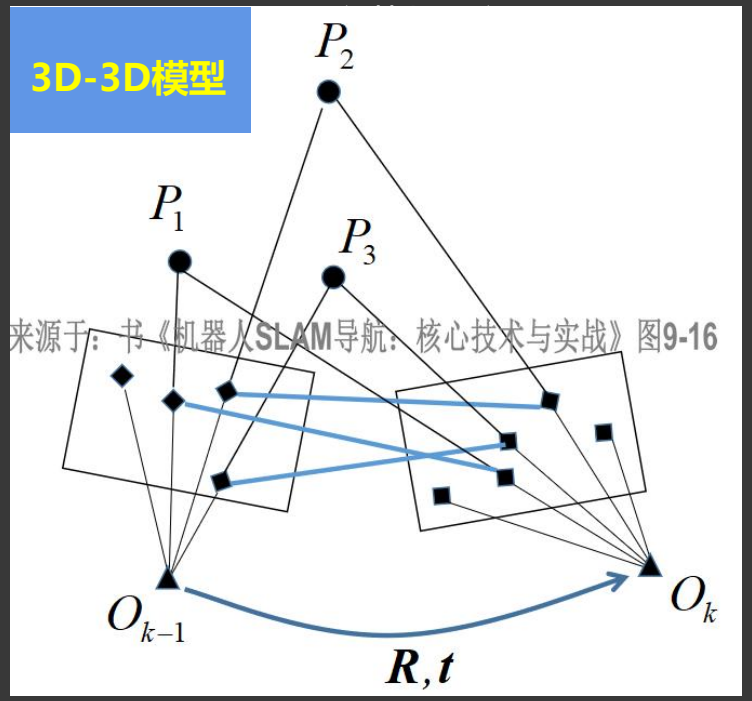
9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读

ORB-SLAM2原理分析 →
- 特征点法
- 三维空间运动
- 多视图几何
- ORB-SLAM2系统框架

三维空间运动 按照模型中给定已知条件的不同 →
- 2D-2D模型
- 3D-2D模型
- 3D-3D模型

3D-3D模型



来源于：书《机器人SLAM导航：核心技术与实战》图9-16

- ICP

求解两组点云之间的位姿变换
- 两组点云的数据关联未知
- 两组点云的数据关联已知

$$\arg\min_{R,t}\sum_{i=1}^{n}\left\|p_i'-(R\bullet p_i+t)\right\|^2$$

- BA优化

对待求量直接进行优化求解

$$T=\arg\min_{T}\sum_{i}\left\|p_i'-T\bullet p_i\right\|^2$$

$$T,p=\arg\min_{T,p}\sum_{i}\left\|p_i'-T\bullet p_i\right\|^2$$

思考：已知量是什么，待求量是什么

* 3D-3D模型的应用场景：比如，上面3D-2D模型中P3P或EPnP中不同坐标系下3D地图点的变换，或者双目、RGB-D能直接给出环境观测点。在这些情况下，当前帧图像所对应的相机位姿解算就可以用这种模型。

9.1 ORB-SLAM2算法

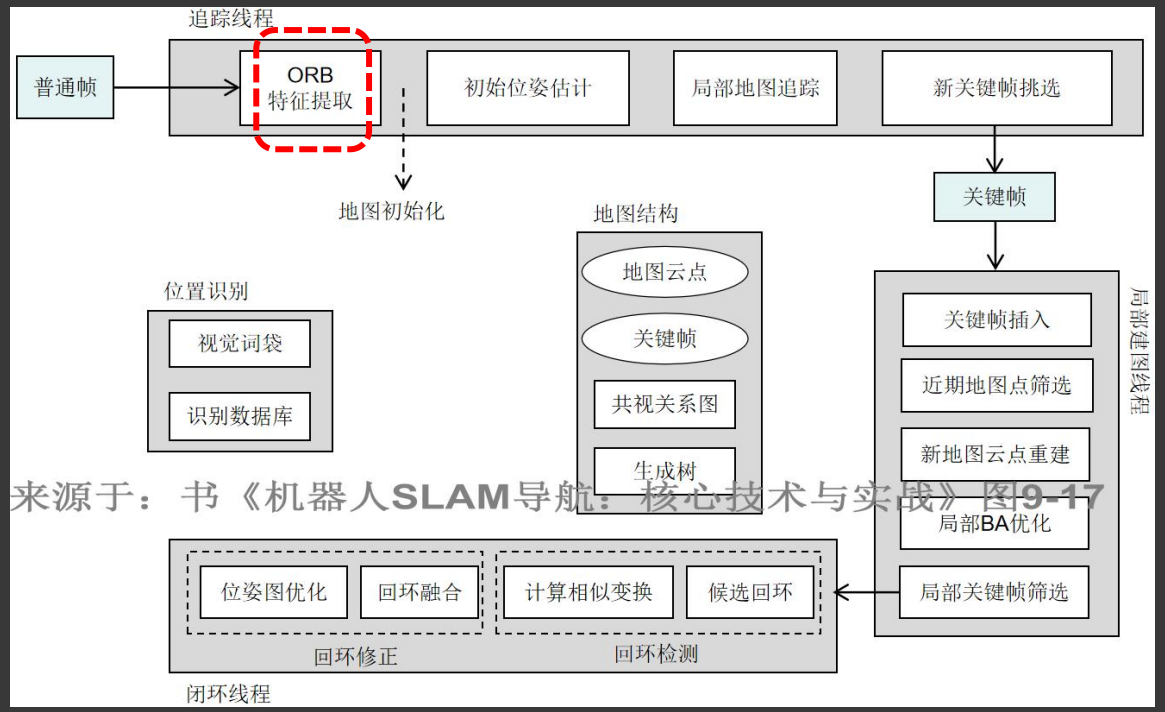
- ORB-SLAM2原理分析 → 特征点法 / 三维空间运动 / 多视图几何 / ORB-SLAM2系统框架
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

ORB-SLAM 单目 ⇨ 有尺度不确定问题

ORB-SLAM2 单目/双目/RGB-D ⇨ 无尺度不确定问题

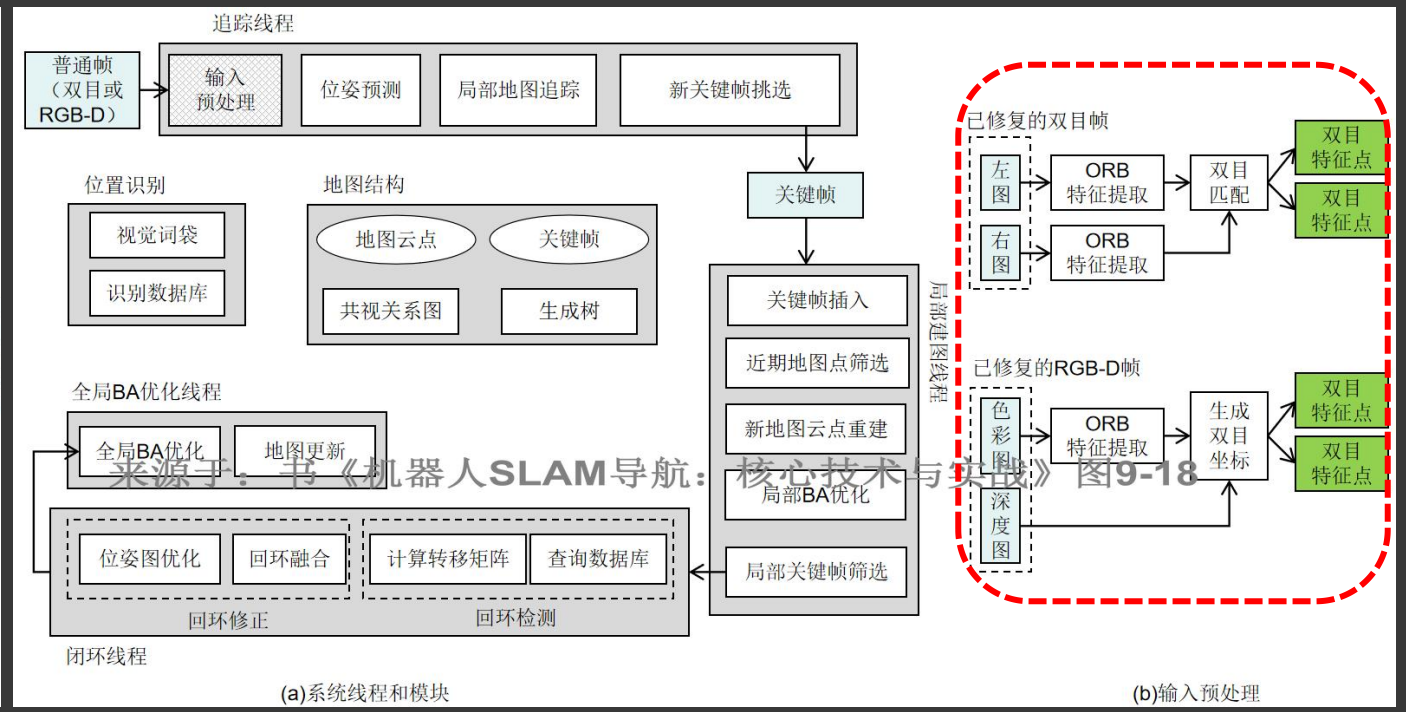
ORB-SLAM3 单目/双目/RGB-D + IMU +Multi-Map

旋转和局部动态性 + 系统持久化



来源于：书《机器人SLAM导航：核心技术与实战》图9-17

纯单目系统框架



来源于：书《机器人SLAM导航：核心技术与实战》图9-18

双目和RGB-D系统框架

9.1 ORB-SLAM2算法

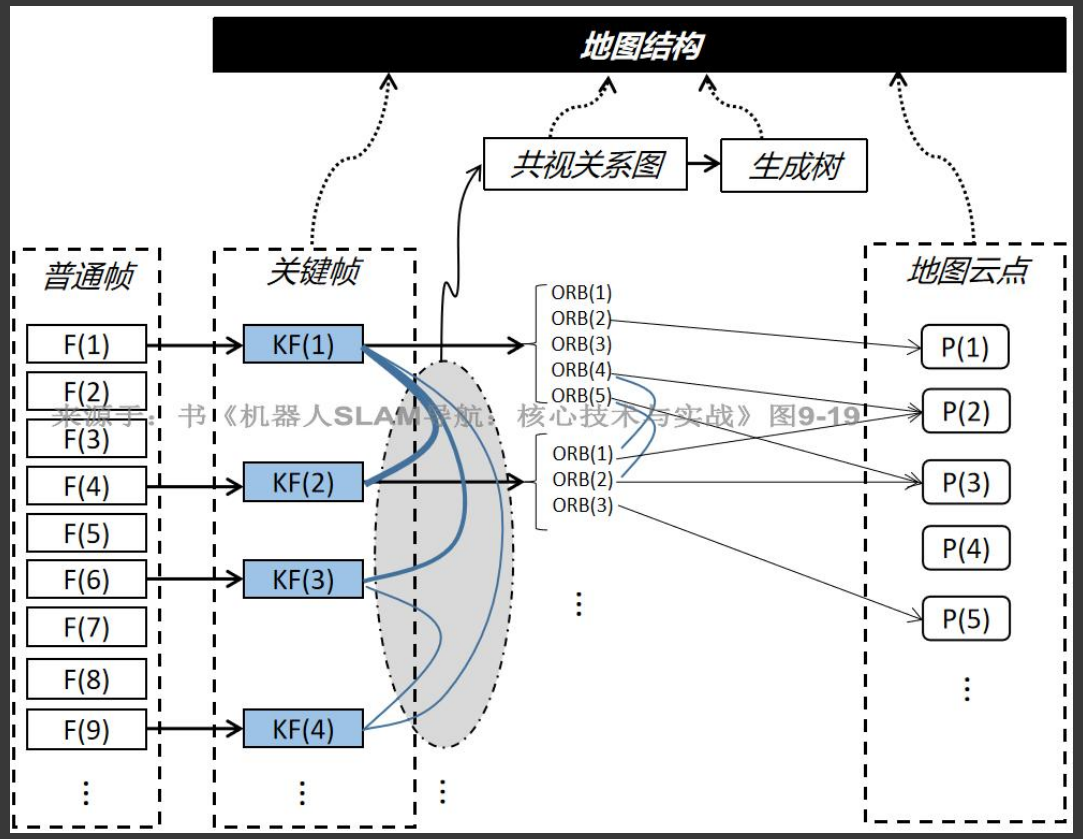
- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

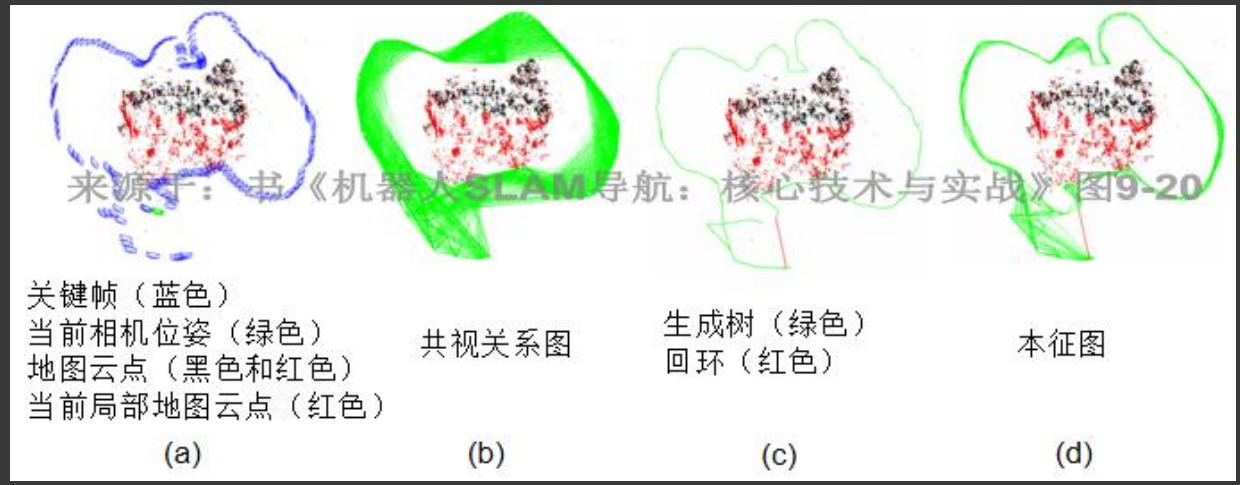
讨论纯单目情况为例

地图结构
地图初始化
位置识别
追踪线程
局部建图线程
闭环线程

- 关键帧
- 地图云点
- 共视关系图
- 生成树



来源于：书《机器人SLAM导航：核心技术与实战》图9-19



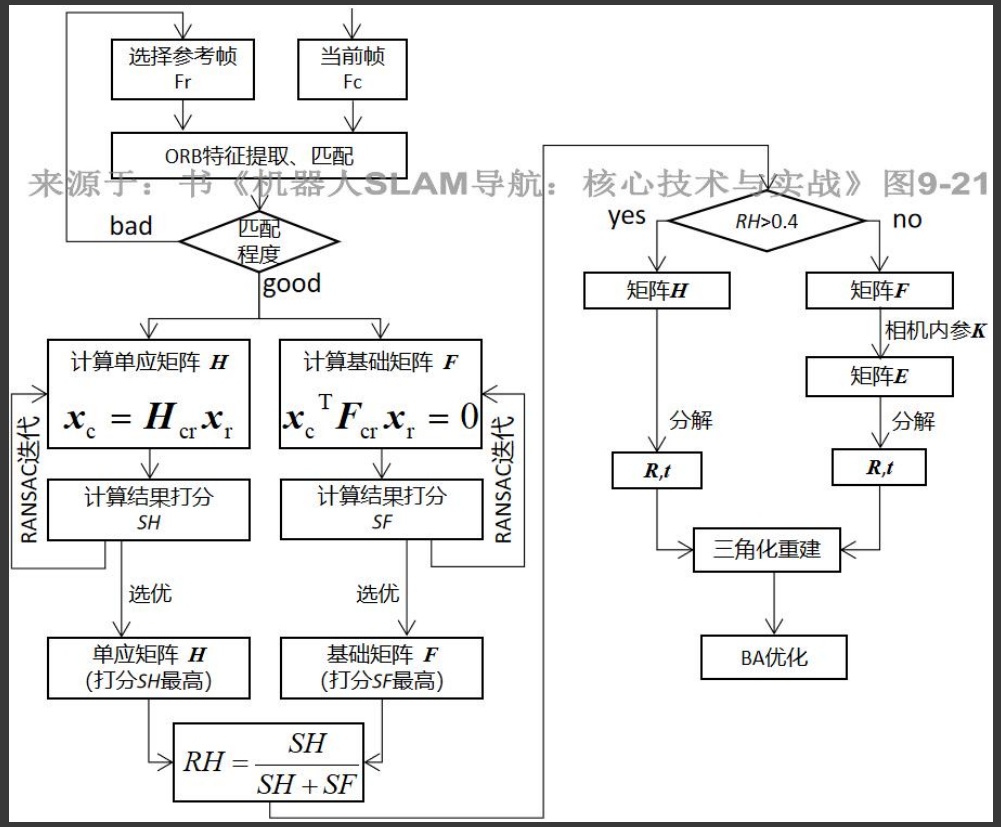
来源于：书《机器人SLAM导航：核心技术与实战》图9-20

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

ORB-SLAM2原理分析 →
- 特征点法
- 三维空间运动
- 多视图几何
- ORB-SLAM2系统框架

讨论纯单目情况为例 →
- 地图结构
- 地图初始化
- 位置识别
- 追踪线程
- 局部建图线程
- 闭环线程



单目SLAM系统的地图初始化过程中，通过计算选取的两帧图像之间相机的位姿变换关系，三角化重建地图初始云点，方法有很多：

- 方法1：追踪环境中的某个已知物体（比如贴在墙上的棋盘标定板）地图云点
- 方法2：追踪环境中某个平面上的一些点
- 方法3：追踪环境中非共面的一些点

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

讨论纯单目情况为例

地图结构
地图初始化
位置识别
追踪线程
局部建图线程
闭环线程

如果SLAM建图过程中追踪线程突然跟丢了，此时就要启动**重定位**尽快找回跟丢的位置信息；或者SLAM在构建了一个很大的地图后，要判断当前路过的位置是否先前已经来到过，也就是**闭环检测**：

- Image-to-image matching
- Image-to-map matching
- Map-to-map matching

ORB-SLAM中的位置识别选用了**词袋模型**，为什么？ 搜索效率

* 什么是词袋模型（Bag-of-Words，BoW）？

词袋模型最早用于自然语言处理（Natural Language Processing，NLP）和信息检索领域，比如一个博客网站收录了成千上万篇博客文章，现在你输入一些关键词查询出你想看的文章，或者网站管理员要审核提交的新文章是否与已有文章雷同。

```
Doc1:I like programming,you like too.
Doc2:I also like robots.

#字典
{
1:" I",
2:" like",
3:" programming",
4:" you"
5:" too",
6:" also",
7:" robots"
}

v1 = [1,2,1,1,1,0,0]
v2 = [1,1,0,0,0,1,1]
```

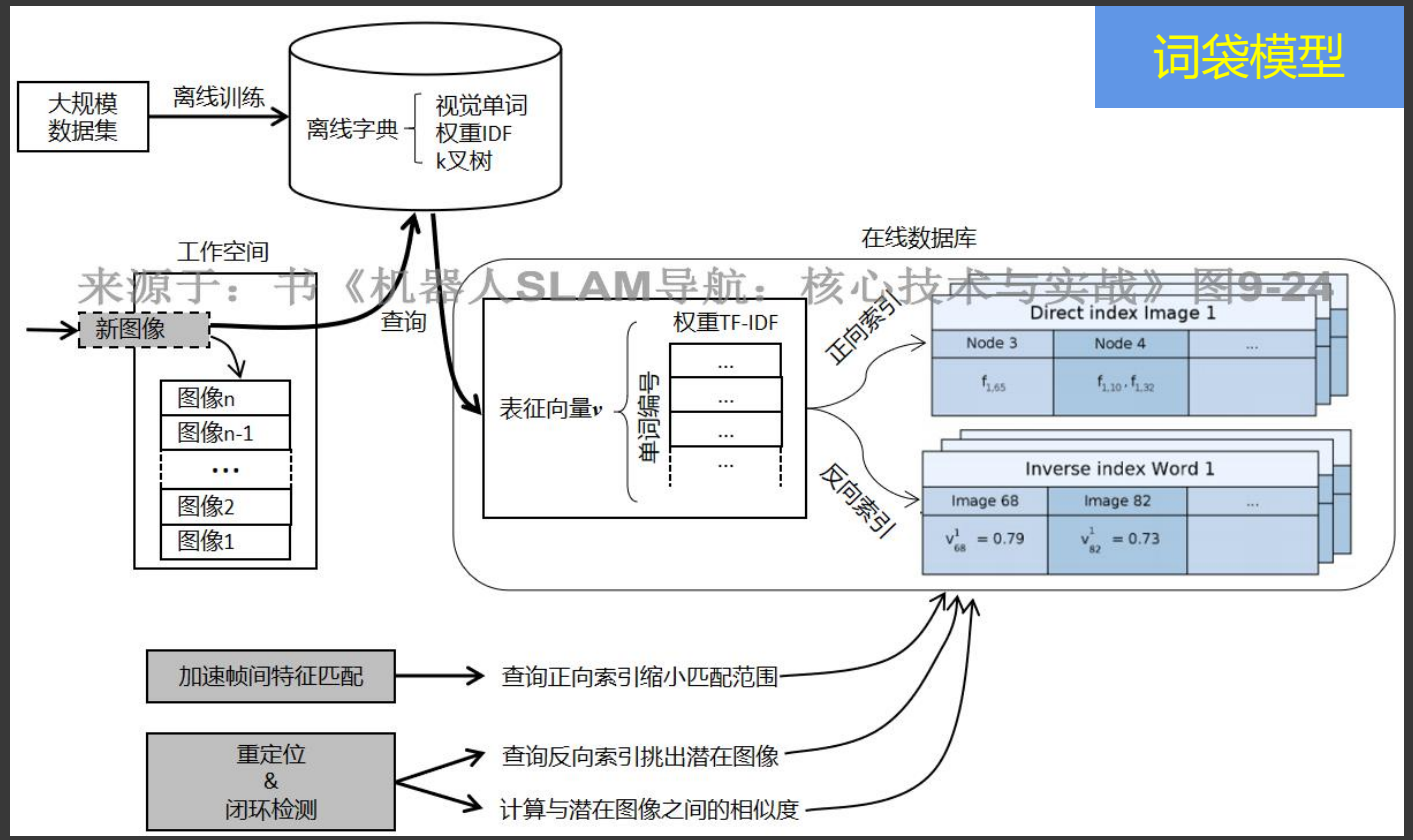
9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

特征点法
三维空间运动
多视图几何
ORB-SLAM2系统框架

讨论纯单目情况为例

地图结构
地图初始化
位置识别
追踪线程
局部建图线程
闭环线程



① 构建视觉单词的离线字典

② 构建图像序列的在线数据库

③ 应用词袋模型

* 关于视觉词袋模型的具体构建过程，请看书本第9章对应内容。

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

ORB-SLAM2原理分析 →
- 特征点法
- 三维空间运动
- 多视图几何
- ORB-SLAM2系统框架 → 讨论纯单目情况为例
  - 地图结构
  - 地图初始化
  - 位置识别
  - 追踪线程
  - 局部建图线程
  - 闭环线程

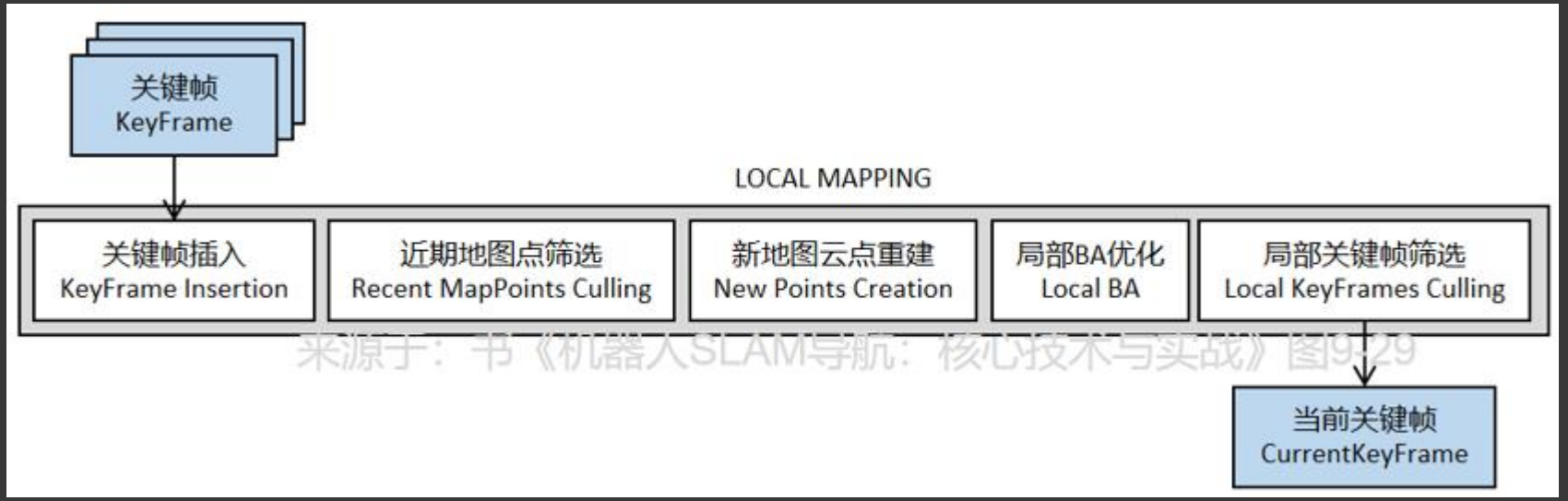
追踪线程

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

ORB-SLAM2原理分析 →
- 特征点法
- 三维空间运动
- 多视图几何
- ORB-SLAM2系统框架

讨论纯单目情况为例 →
- 地图结构
- 地图初始化
- 位置识别
- 追踪线程
- 局部建图线程
- 闭环线程



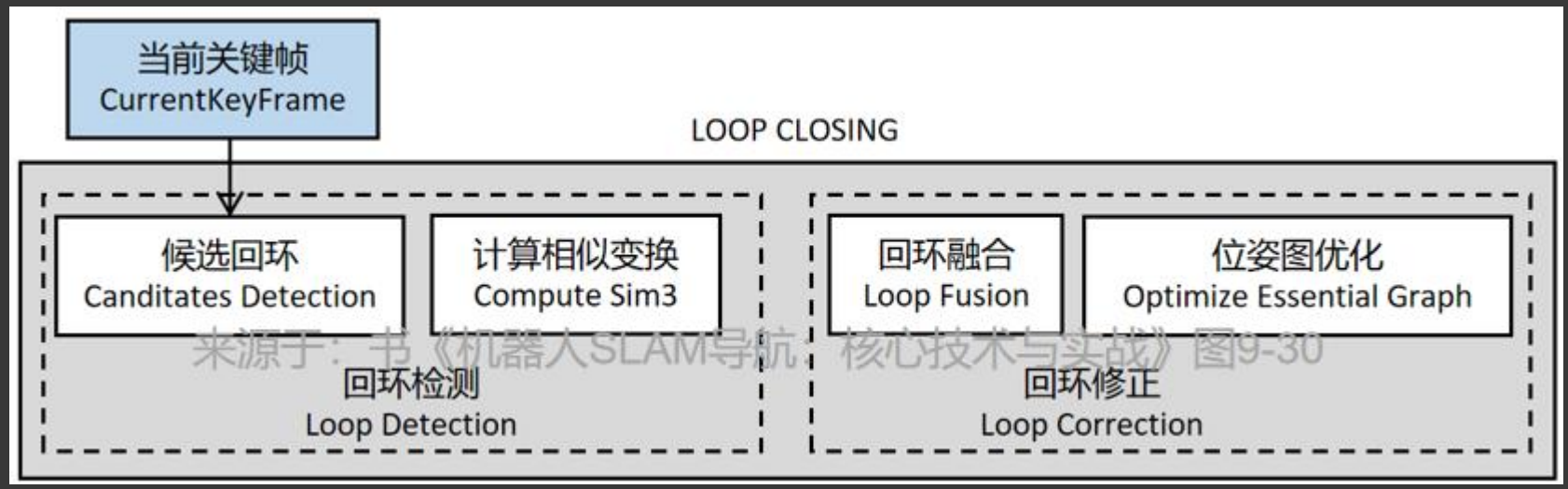
局部建图线程

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

ORB-SLAM2原理分析 → 特征点法、三维空间运动、多视图几何、ORB-SLAM2系统框架

讨论纯单目情况为例 → 地图结构、地图初始化、位置识别、追踪线程、局部建图线程、闭环线程

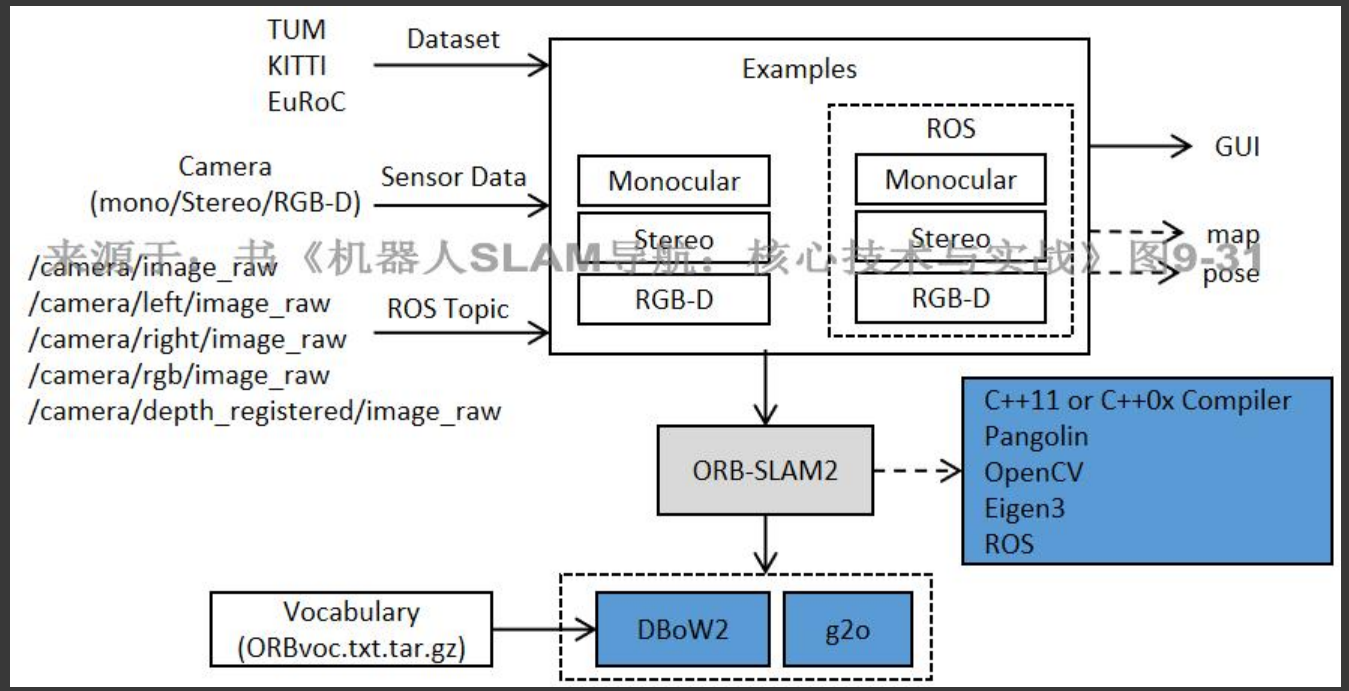


来源于：书《机器人SLAM导航：核心技术与实战》图9-30

闭环线程

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行



ORB-SLAM2代码框架

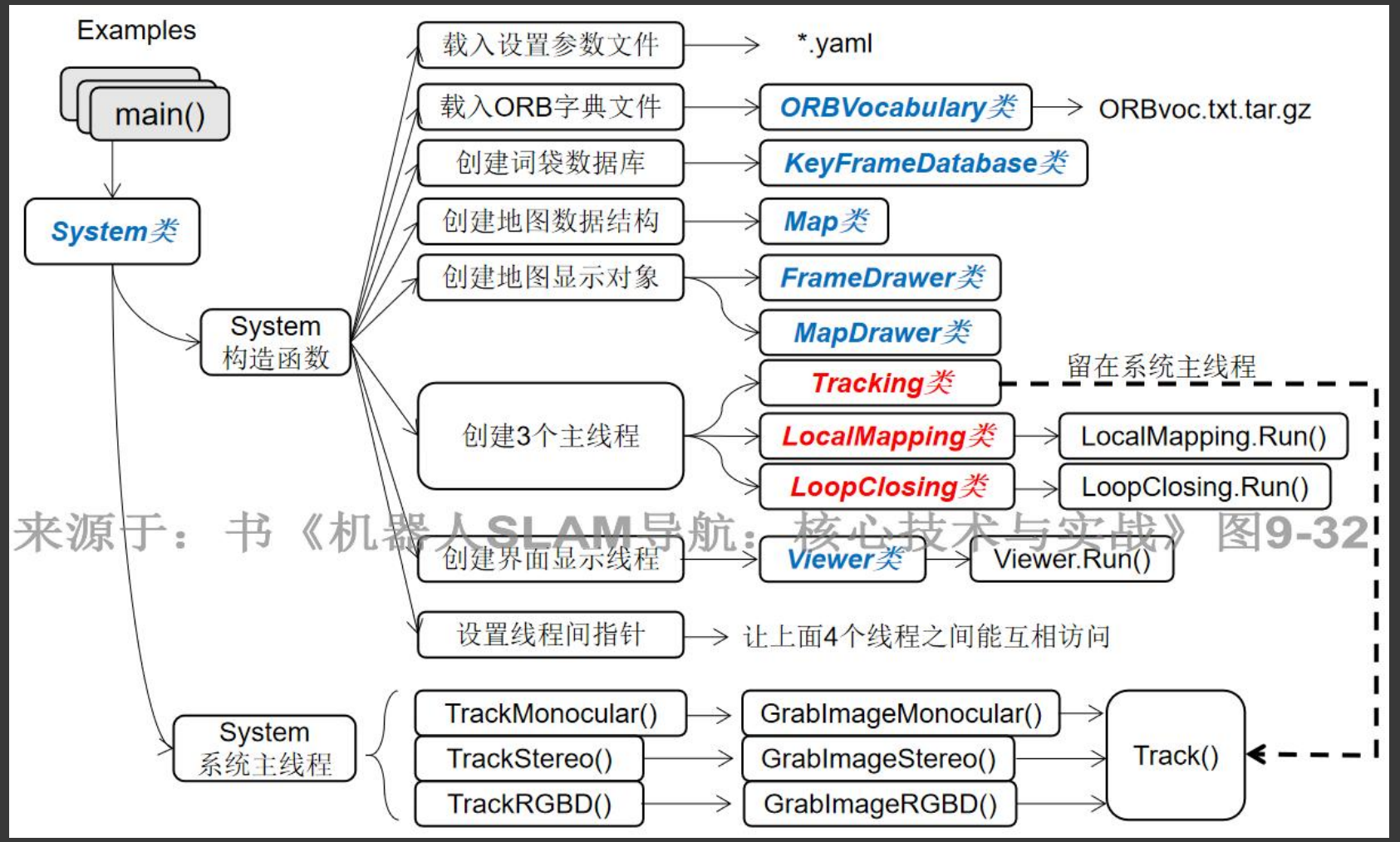
ORB-SLAM2核心库文件

用途		源文件	
算法顶层接口	System 类	System.cc	System.h
地图数据结构	Map 类	Map.cc	Map.h
	MapPoint 类	MapPoint.cc	MapPoint.h
	Frame 类	Frame.cc	Frame.h
	KeyFrame 类	KeyFrame.cc	KeyFrame.h
图形界面显示	MapDrawer 类	MapDrawer.cc	MapDrawer.h
	FrameDrawer 类	FrameDrawer.cc	FrameDrawer.h
	Viewer 类	Viewer.cc	Viewer.h
地图初始化	Initializer 类	Initializer.cc	Initializer.h
词袋模型接口	KeyFrameDatabase 类	KeyFrameDatabase.cc	KeyFrameDatabase.h
			ORBVocabulary.h
ORB 特征处理	ORBextractor 类	ORBextractor.cc	ORBextractor.h
	ORBmatcher 类	ORBmatcher.cc	ORBmatcher.h
PnP 求解器	PnPsolver 类	PnPsolver.cc	PnPsolver.h
Sim3 求解器	Sim3Solver 类	Sim3Solver.cc	Sim3Solver.h
优化	Optimizer 类	Optimizer.cc	Optimizer.h
数据格式转换	Converter 类	Converter.cc	Converter.h
3 个主线程	Tracking 类	Tracking.cc	Tracking.h
	LocalMapping 类	LocalMapping.cc	LocalMapping.h
	LoopClosing 类	LoopClosing.cc	LoopClosing.h

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

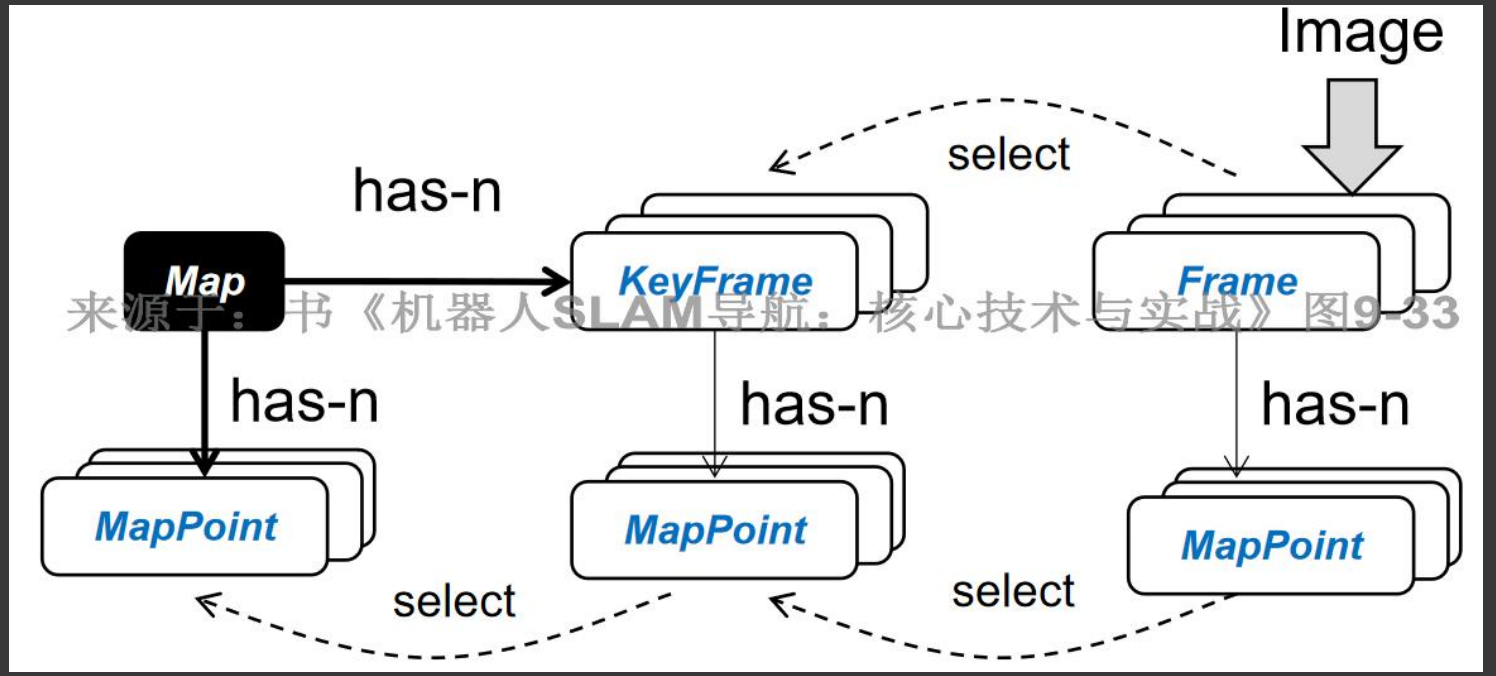
System类是整个ORB-SLAM2核心库的顶层接口，Examples文件夹中所有例程的main()函数都会创建一个System类的对象。



System类

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行



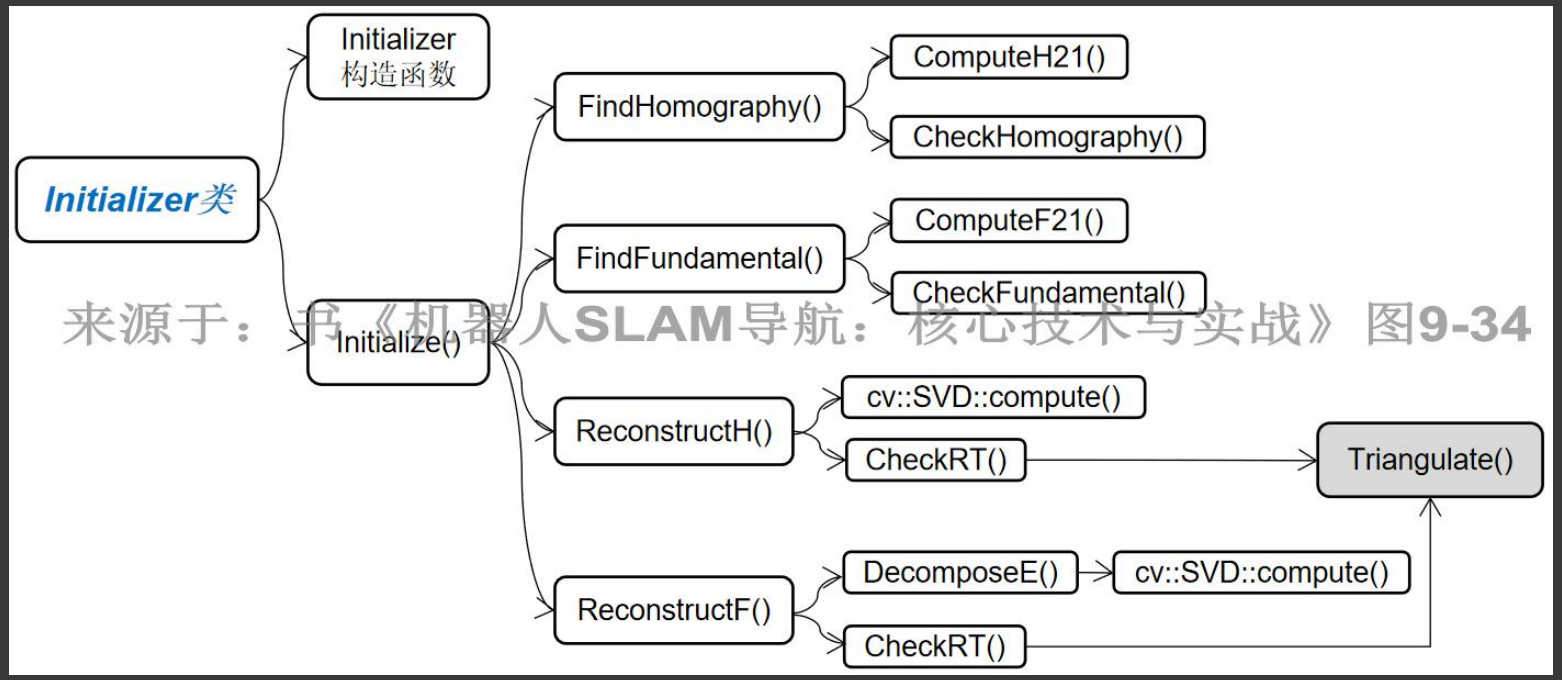
Map、MapPoint、Frame和KeyFrame类

地图数据结构涉及到的4个类Map、MapPoint、Frame和KeyFrame之间的关系是错综复杂的，各个类之间既互相调用又互相包含。

每个从传感器输入的图像（Image）都会建立一个对应的Frame，经过追踪线程中的特征提取、特征匹配、三角化重建等处理，每个Frame都包含许多个MapPoint。通过关键帧筛选机制，将符合要求的一些Frame筛选出来成为KeyFrame，同时原Frame包含的MapPoint也要经过严苛地筛选后成为KeyFrame中的MapPoint。而KeyFrame中的MapPoint经过进一步的筛选后成为Map中的MapPoint，这些MapPoint和KeyFrame共同构成了整个Map。

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行



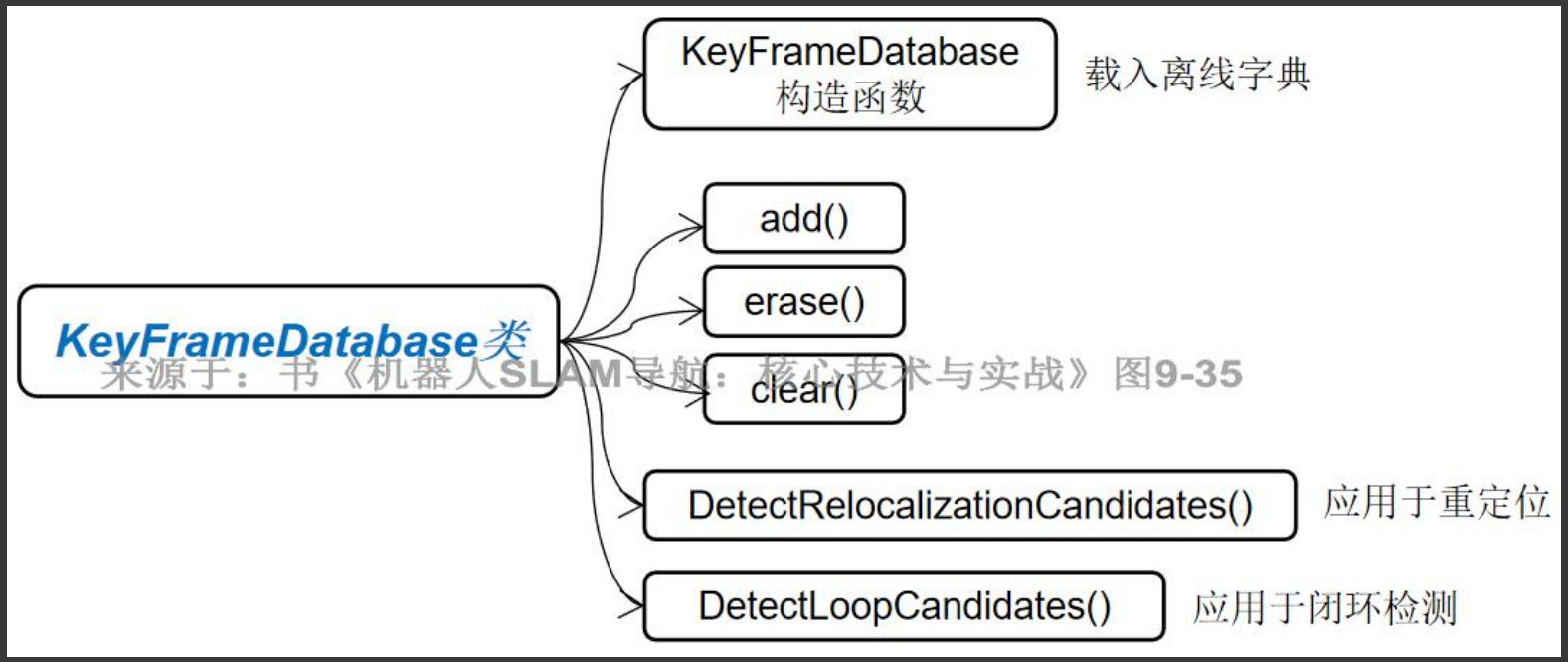
来源于：书《机器人SLAM导航：核心技术与实战》图9-34

Initializer类

由于双目或RGB-D的地图初始化就简单多了，就没有封装成专门的类。而单目SLAM的地图初始化比较复杂，所以被封装成了一个专门的Initializer类。

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行



KeyFrameDatabase类

词袋模型包括离线字典、在线数据库和应用3部分，这里的KeyFrameDatabase类就是来实现在线数据库的。

9.1 ORB-SLAM2算法

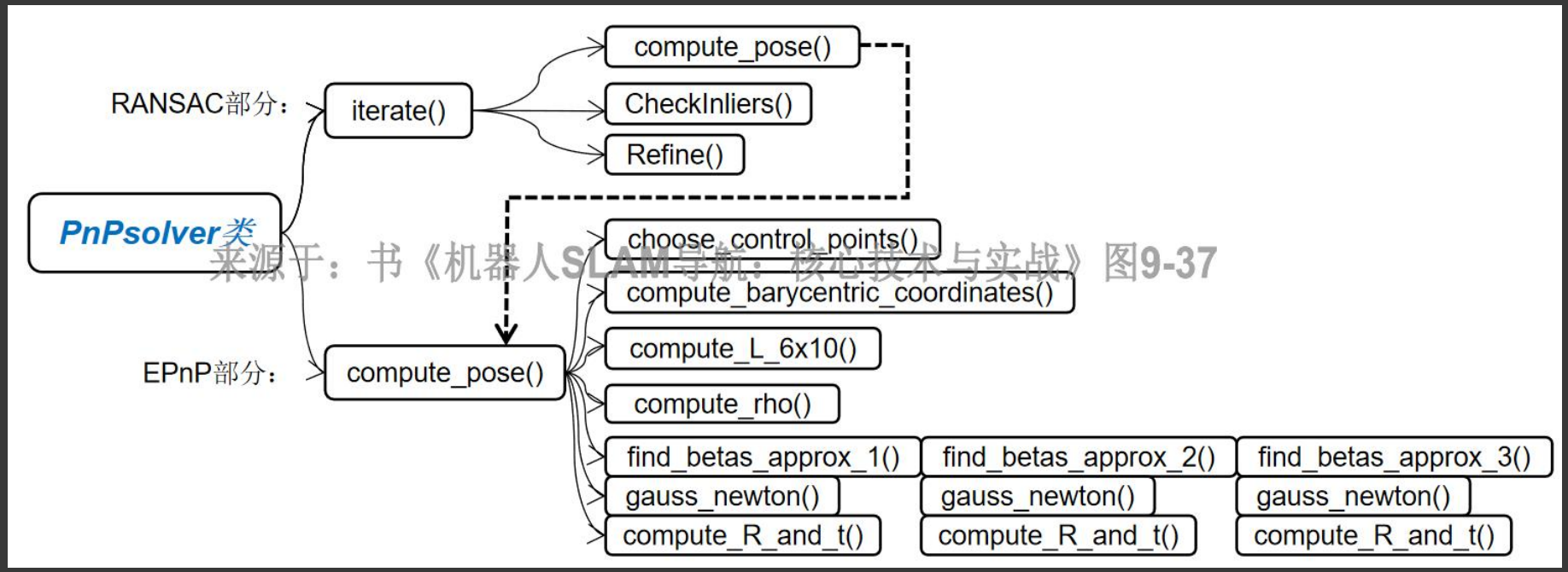
- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

ORBextractor和ORBmatcher类

ORBextractor和ORBmatcher类分别对应于ORB特征提取和ORB特征匹配。其实在OpenCV中有关于ORB特征提取和匹配的实现，只不过这里ORB-SLAM的作者为了满足SLAM应用中的实际需求，重写了ORB特征提取和ORB特征匹配的实现。

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

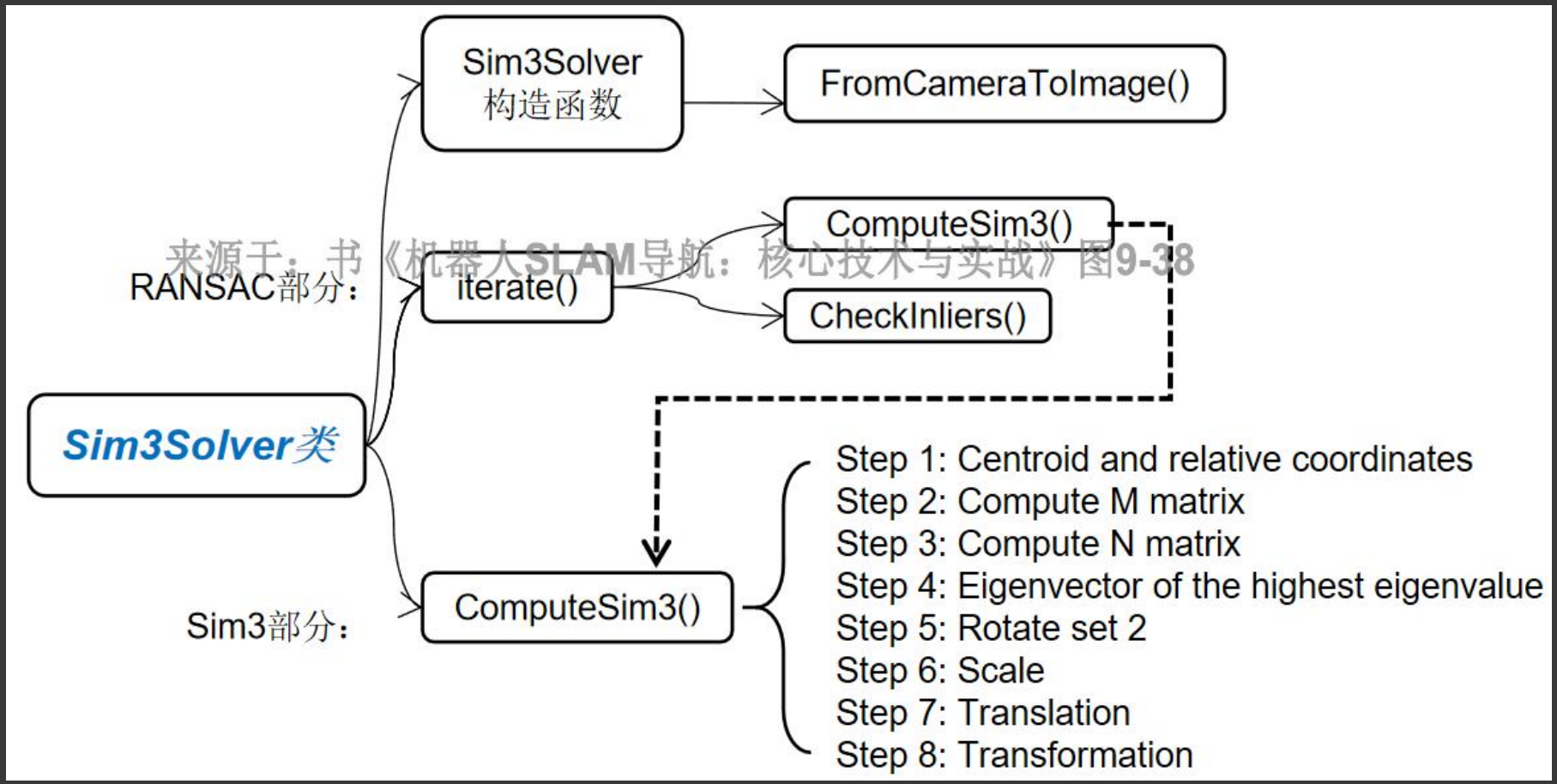


PnPsolver类

在ORB-SLAM中采用RANSAC+EPnP的方式求解PnP问题，具体过程被封装在PnPsolver类。

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

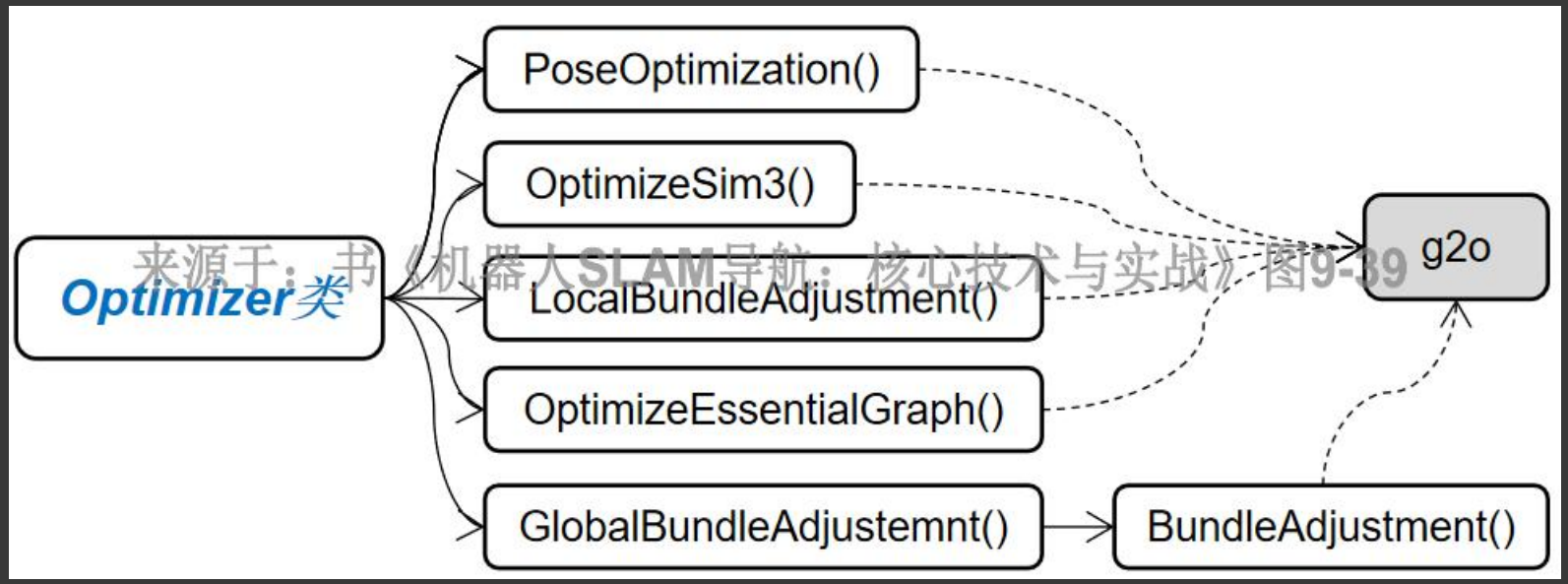


Sim3Solver类

在闭环线程中，当检测到回环候选帧后，要计算回环候选帧与当前关键帧之间的相似变换Sim3便于后续的回环融合，求解Sim3的过程被封装在Sim3Solver类。

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行



来源于：书《机器人SLAM导航：核心技术与实战》图9-39

Optimizer类

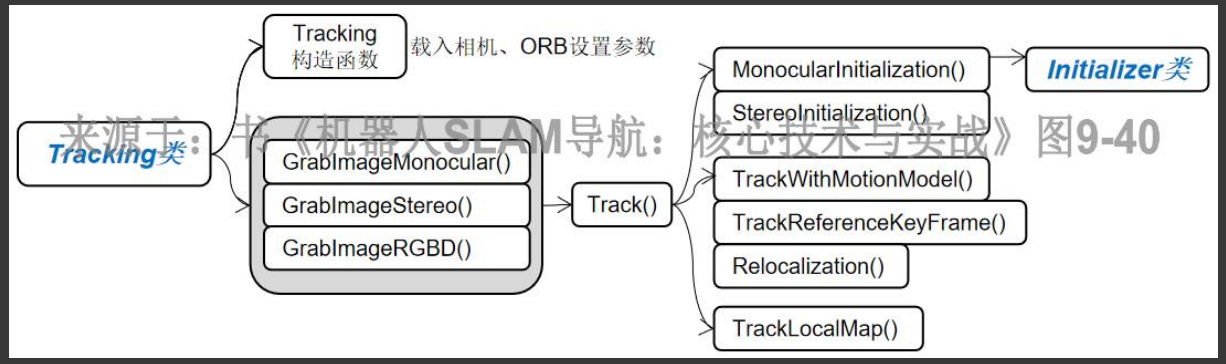
- 当前帧位姿优化：PoseOptimization()
- 闭环帧间位姿优化：OptimizeSim3()
- 局部BA优化：LocalBundleAdjustment()
- 全局BA优化（简化版）：OptimizeEssentialGraph()
- 全局BA优化（完整版）：GlobalBundleAdjustemnt()

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

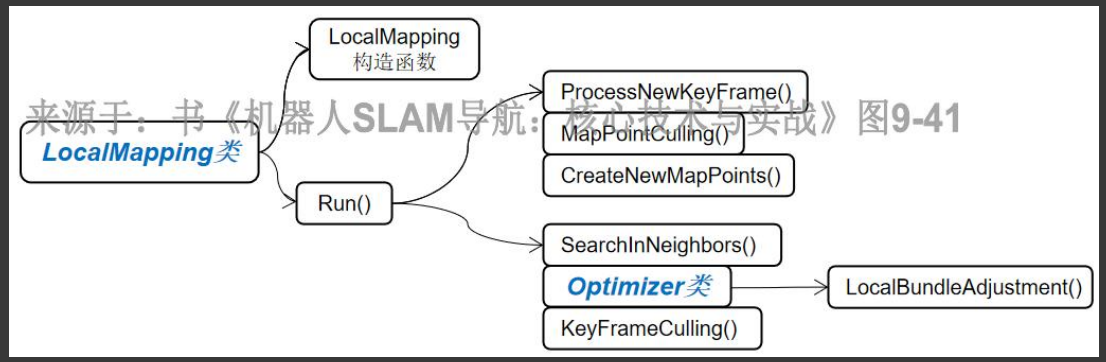
系统的3个主线程追踪线程、局部建图线程和闭环线程的具体实现分别封装在Tracking类、LocalMapping类和LoopClosing类之中。

Tracking类



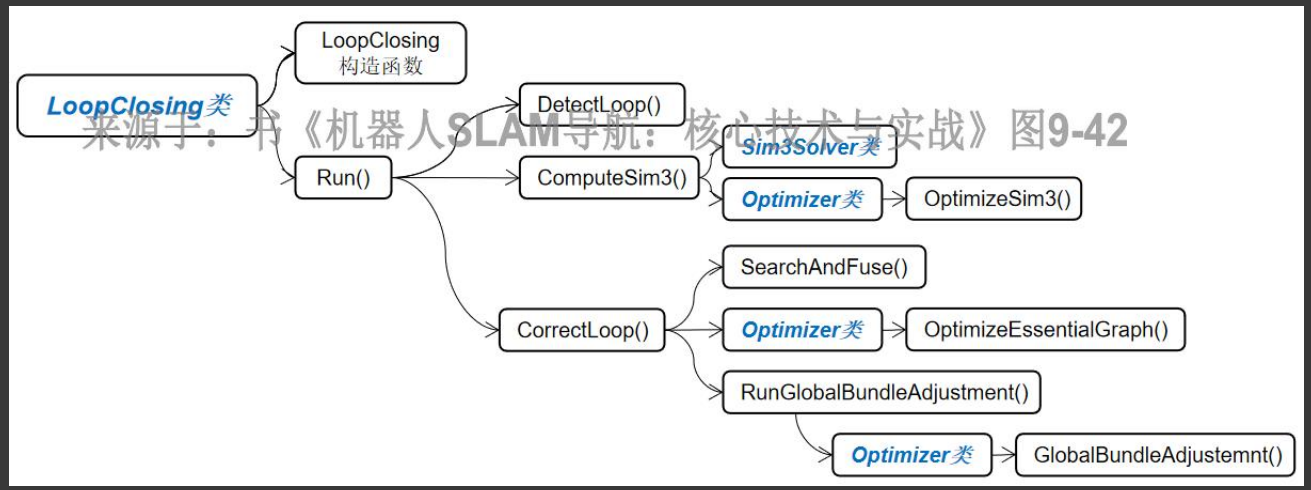
来源于：书《机器人SLAM导航：核心技术与实战》图9-40

LocalMapping类



来源于：书《机器人SLAM导航：核心技术与实战》图9-41

LoopClosing类



来源于：书《机器人SLAM导航：核心技术与实战》图9-42

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

① ORB-SLAM2安装
② ORB-SLAM2离线运行
③ ORB-SLAM2在线运行

① 安装依赖项（C++11、Pangolin、OpenCV、Eigen）
② 编译源码

环境：Ubuntu18.04+ROS melodic

```
g++ -v

#安装Pangolin必须依赖
sudo apt install libgl1-mesa-dev
sudo apt install libglew-dev
sudo apt install cmake
#下载Pangolin源码
git clone https://github.com/stevenlovegrove/Pangolin.git
#编译安装
cd Pangolin
mkdir build
cd build
cmake ..
cmake --build .

#ROS melodic已经自带了OpenCV3.2.0，故无须再安装，版本查看方法如下
pkg-config opencv --modversion
```

```
#安装Eigen3
#取别名来移除系统自带的Eigen3.3.4
sudo mv -f /usr/include/eigen3 /usr/include/eigen3.default.bak
#下载Eigen源码
git clone https://gitlab.com/libeigen/eigen.git
#切换代码分支
cd eigen
git checkout 3.2.10
#编译安装
mkdir build
cd build
cmake ..
sudo make install
#将eigen3/Eigen拷贝到上一级目录
sudo cp -rf /usr/local/include/eigen3/Eigen /usr/local/include/
#将eigen3和Eigen拷贝到/usr/include/之中
sudo cp -rf /usr/local/include/eigen3 /usr/include/
sudo cp -rf /usr/local/include/Eigen /usr/include/
```

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

① ORB-SLAM2安装
② ORB-SLAM2离线运行
③ ORB-SLAM2在线运行

① 安装依赖项（C++11、Pangolin、OpenCV、Eigen）
② 编译源码

环境：Ubuntu18.04+ROS melodic

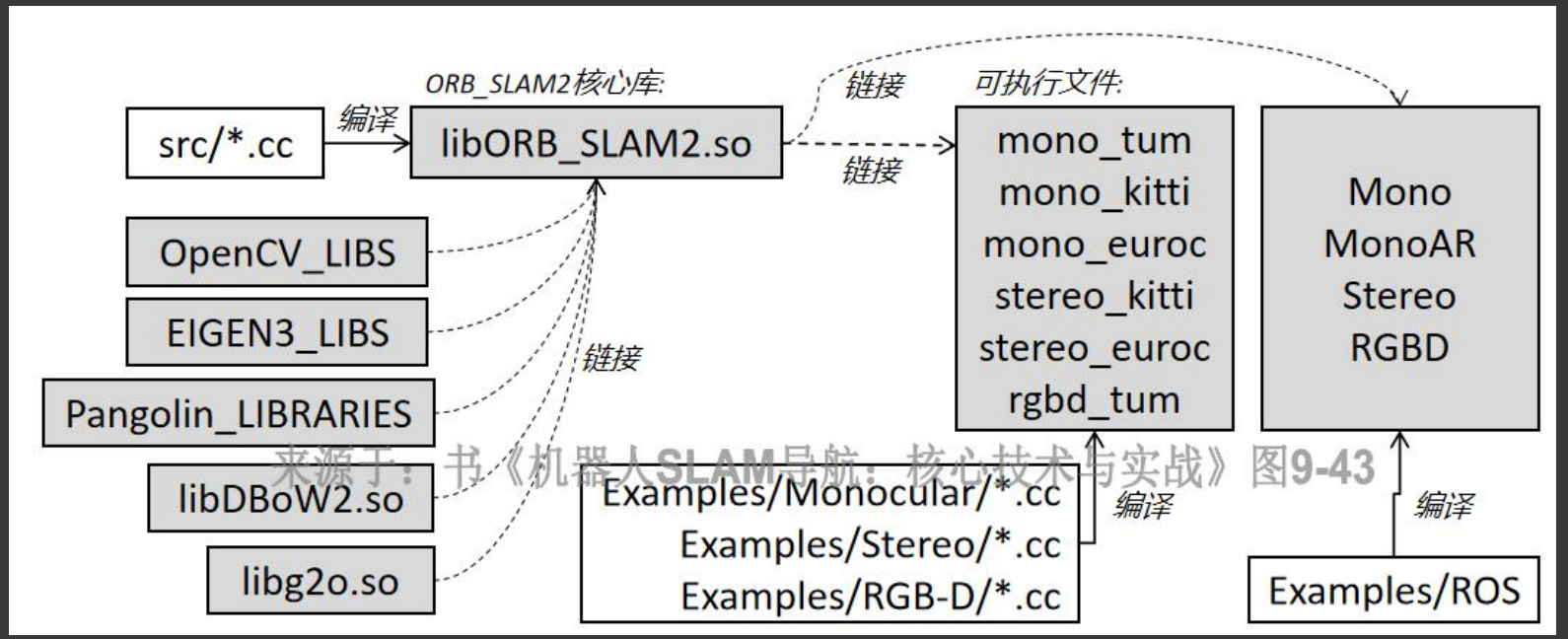
```
#下载ORB_SLAM2源码
git clone https://github.com/raulmur/ORB_SLAM2.git ORB_SLAM2

#编译ORB_SLAM2源码
cd ORB_SLAM2
chmod +x build.sh
./build.sh
```

思考：

通过解读build.sh和CMakeLists.txt文件的内容来理解

ORB_SLAM2编译具体流程ORB_SLAM2编译具体流程?



来源于：书《机器人SLAM导航：核心技术与实战》图9-43

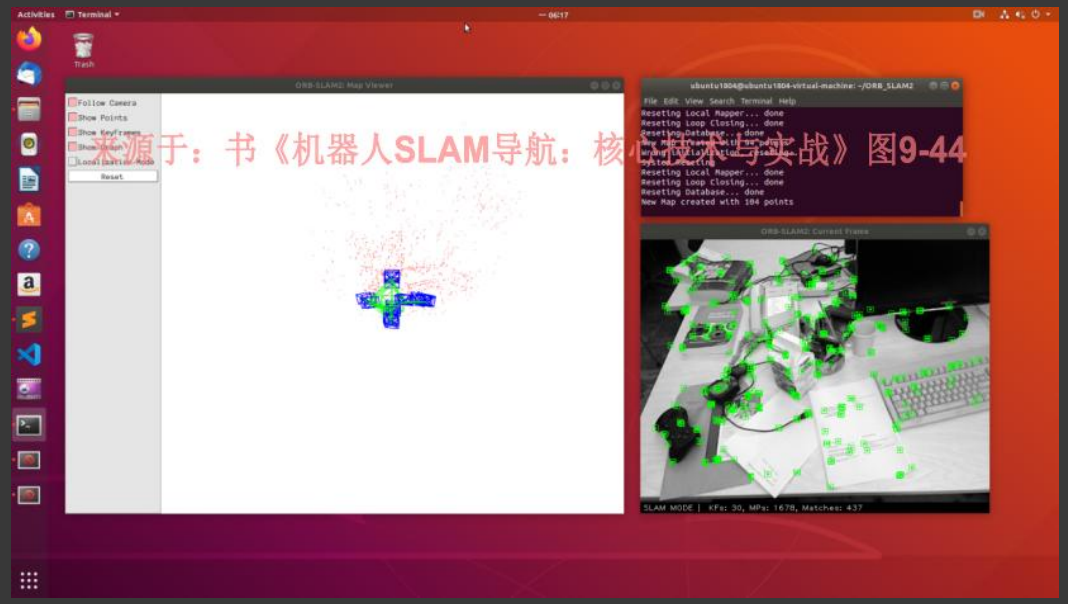
9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

① ORB-SLAM2安装
② ORB-SLAM2离线运行
③ ORB-SLAM2在线运行

环境：Ubuntu18.04+ROS melodic

```
cd ORB_SLAM2
#启动可执行文件mono_tum，注意下面是一句长的完整命令
./Examples/Monocular/mono_tum Vocabulary/ORBvoc.txt Examples/Monocular/TUM1.yaml ../rgbd_dataset_freiburg1_xyz
```



ORB_SLAM2离线运行效果

9.1 ORB-SLAM2算法

- ORB-SLAM2原理分析
- ORB-SLAM2源码解读
- ORB-SLAM2安装与运行

环境：Ubuntu18.04+ROS melodic

① ORB-SLAM2安装
② ORB-SLAM2离线运行
③ ORB-SLAM2在线运行

?

① 编写自己的实例节点程序

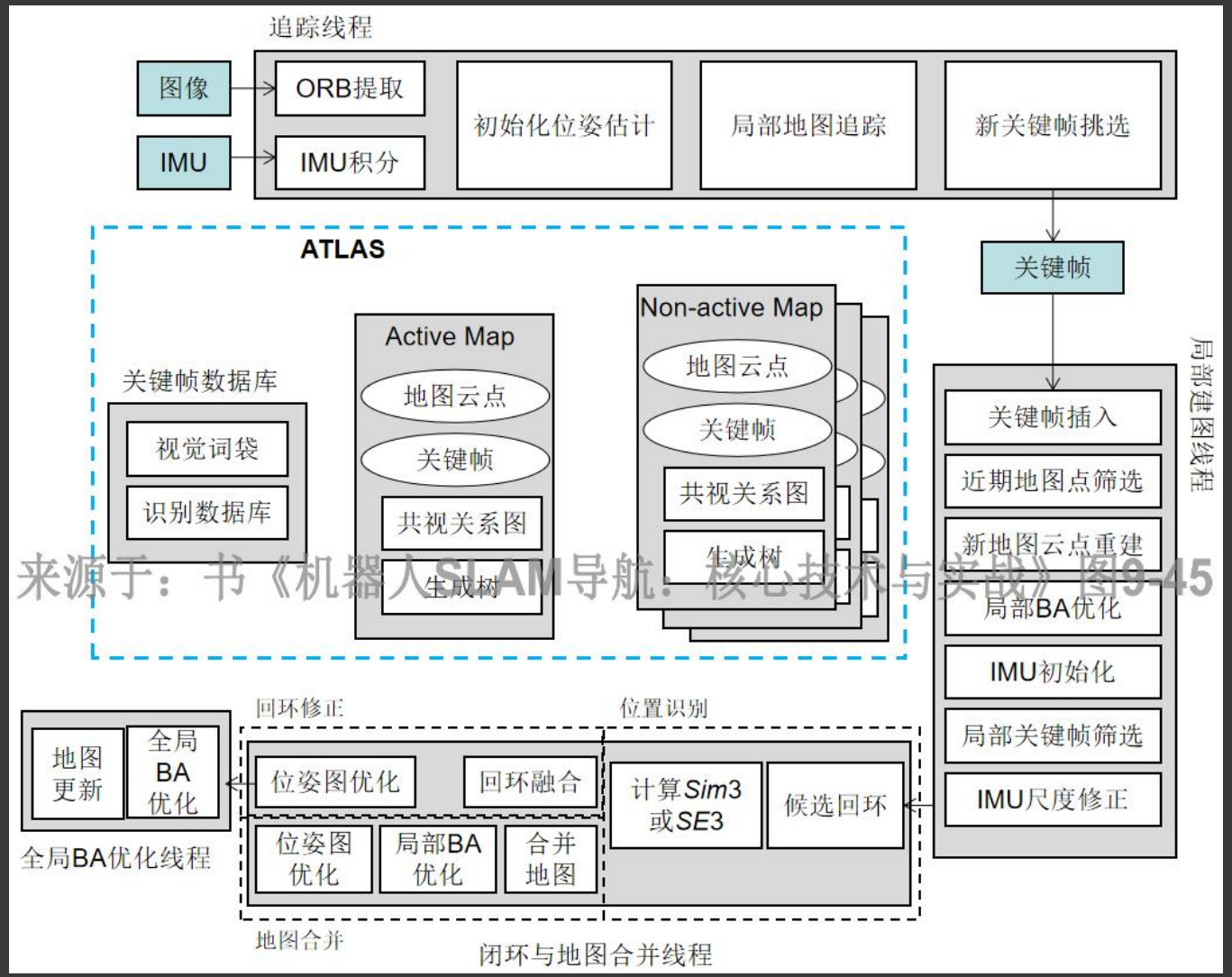
② 获取实际图像数据

③ 标定相关的配置文件

9.1 ORB-SLAM2算法

拓展

ORB-SLAM
ORB-SLAM2
ORB-SLAM3



来源于：书《机器人SLAM导航：核心技术与实战》图9-45

ORB-SLAM3的特色

- **多地图机制（multi-map）**

保证持久建图过程的鲁棒性

重定位更容易

ATLAS机制与Cartographer的submap机制很类似

- **视觉惯导融合（visual-inertial）**

初始化更容易

加入IMU约束的BA优化过程效果更好

ORB-SLAM3系统架构

9.1 ORB-SLAM2算法

拓展 ORB-SLAM
ORB-SLAM2
ORB-SLAM3

来源于：书《机器人SLAM导航：核心技术与实战》图9-46

	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][1]
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][2]
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][3]
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 [7], [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]

内容概要

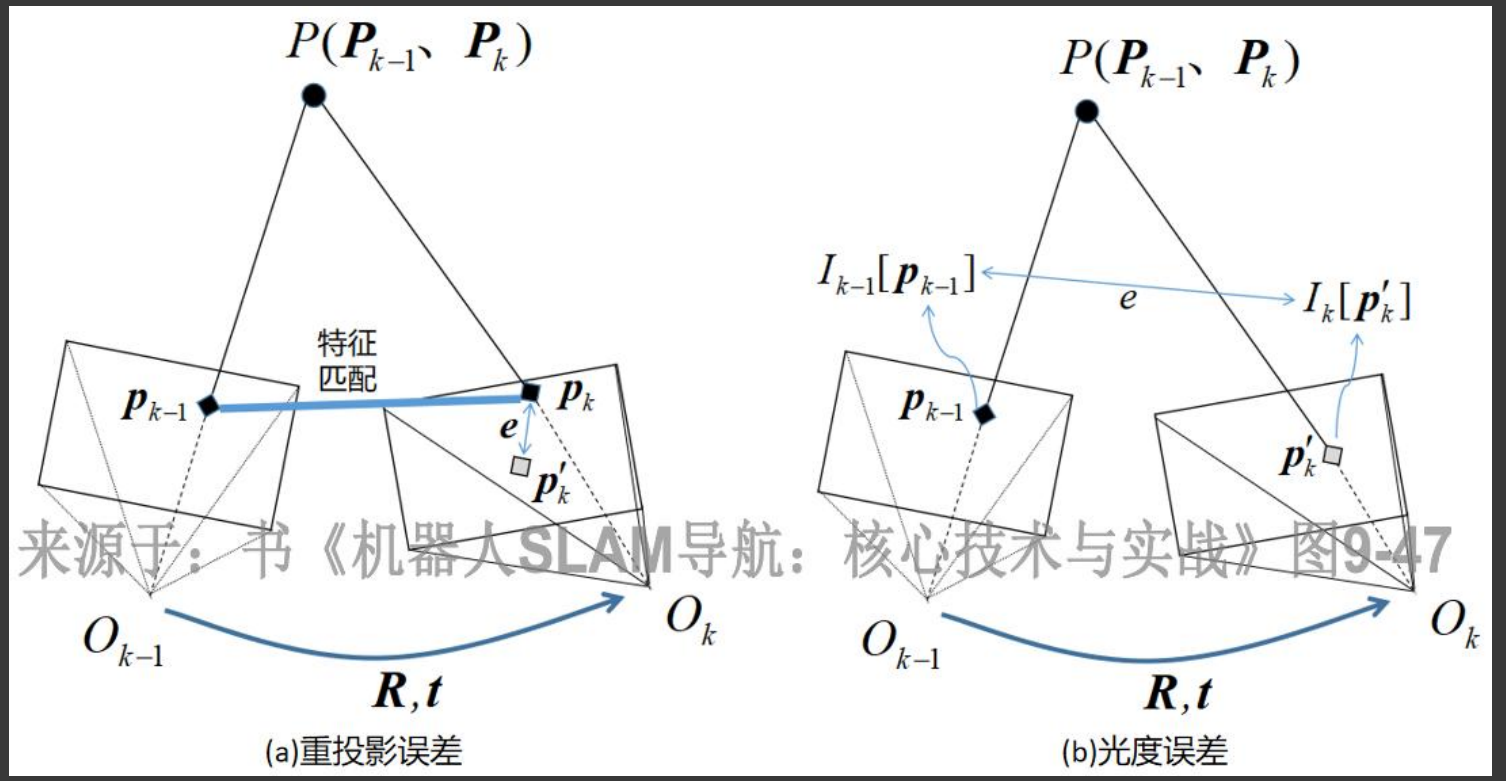
9.2 LSD-SLAM算法

Engel J , Schps T , Cremers D . LSD-SLAM: Large-scale direct monocular SLAM[J]. 2014.

LSD-SLAM是什么？

9.2 LSD-SLAM算法

- LSD-SLAM原理分析 → 直接法 LSD-SLAM系统框架
- LSD-SLAM源码解读
- LSD-SLAM安装与运行



最小化重投影误差：

$$P_k = T \bullet P_{k-1}, \text{其中} T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

$$p'_k = \pi(P_k) = \frac{1}{z_k} K \bullet P_k$$

$$e = p_k - p'_k$$

$$\min_{T,P} \sum_i \|e_i\|^2 = \min_{T,P} \sum_i \left\| p_k^i - \pi(T \bullet P_{k-1}^i) \right\|^2$$

最小化光度误差：

$$e = I_{k-1}[p_{k-1}] - I_k[p'_k]$$ （光度不变假设？）

$$\min_{T,P} \sum_i (e_i)^2 = \min_{T,P} \sum_i \left(I_{k-1}[p_{k-1}^i] - I_k[\pi(T \bullet P_{k-1}^i)] \right)^2$$

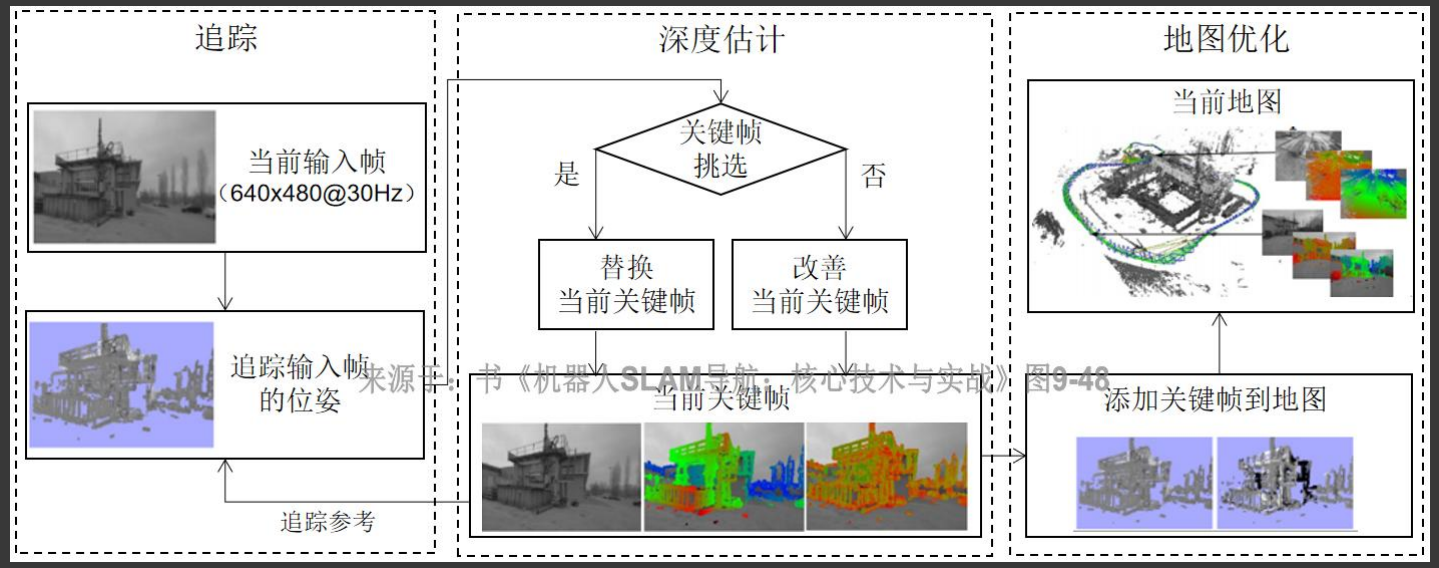
特征点法：像素点->特征点->**重投影误差**->建立数据关联

直接法：　像素点---------->**光度误差**----->建立数据关联

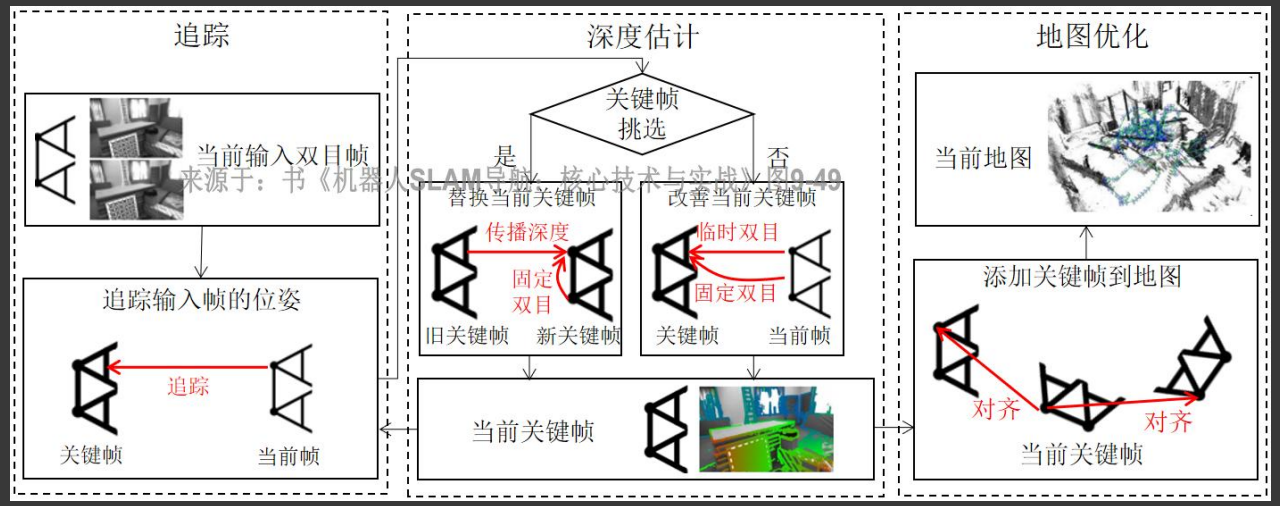
思考：特征点法和直接法各自的优缺点？

9.2 LSD-SLAM算法

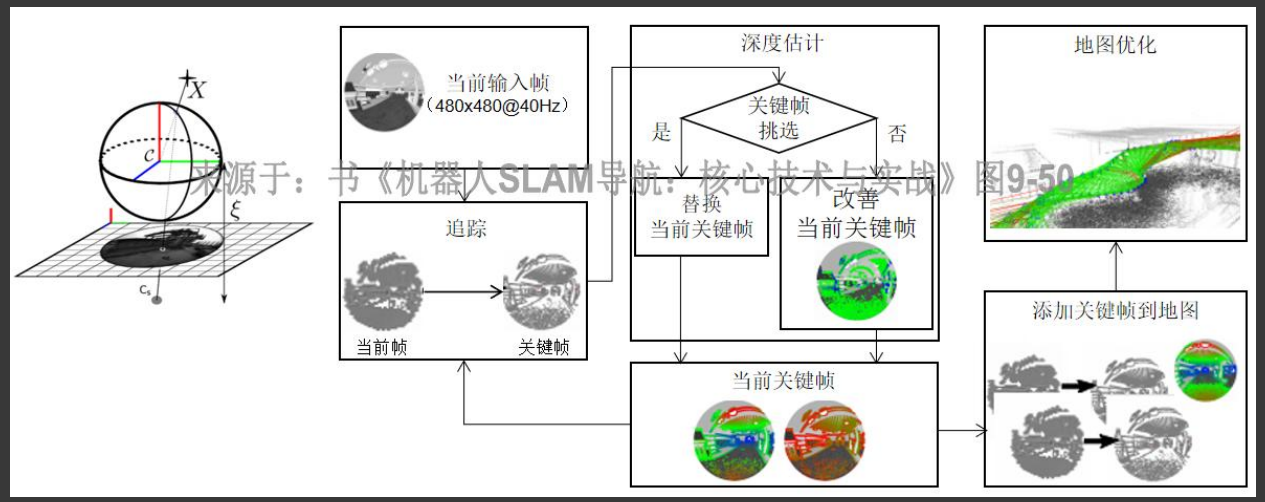
- LSD-SLAM原理分析 → 直接法 LSD-SLAM系统框架
  - 追踪
  - 深度估计
  - 地图优化
- LSD-SLAM源码解读
- LSD-SLAM安装与运行



单目系统框架



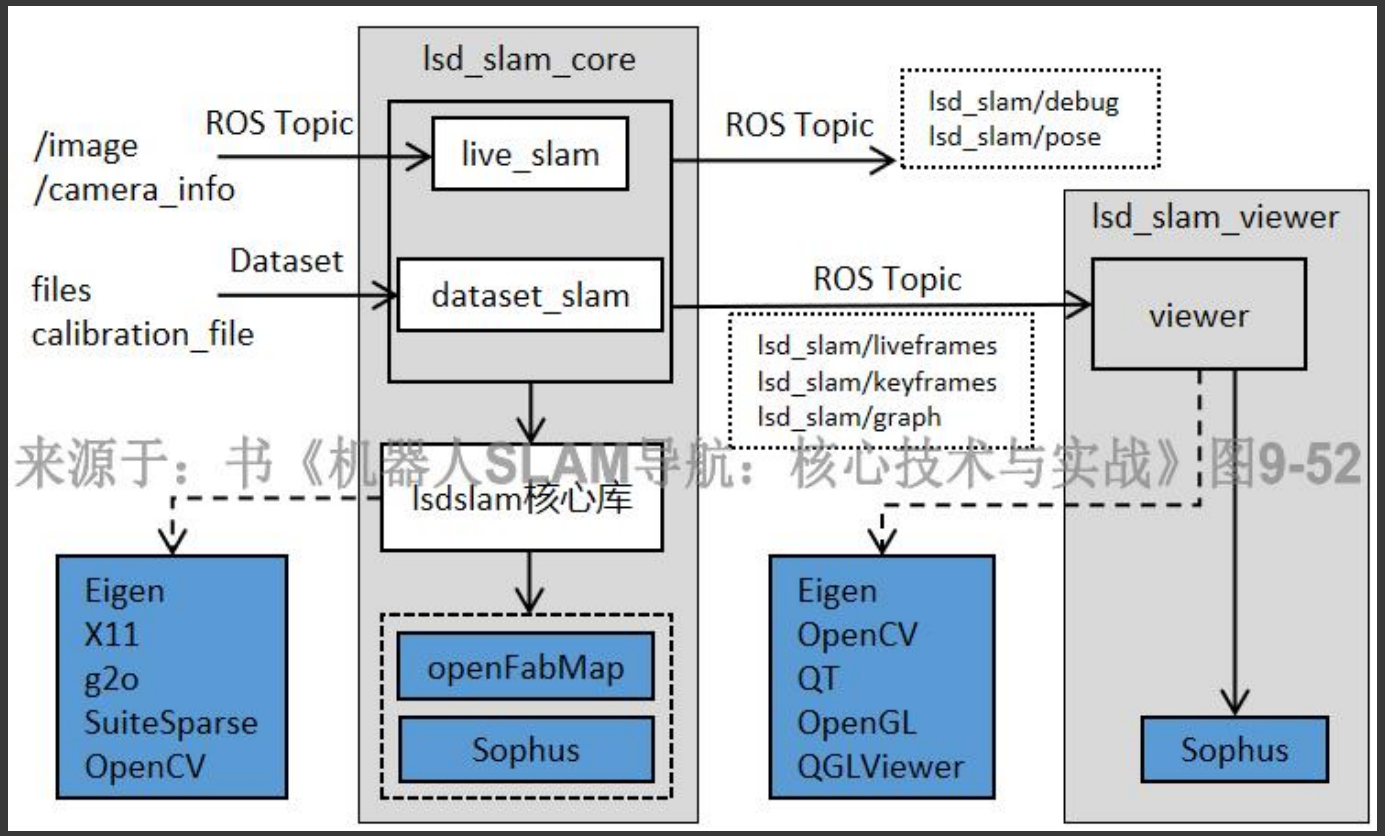
双目系统框架



全景系统框架

9.2 LSD-SLAM算法

- LSD-SLAM原理分析
- LSD-SLAM源码解读
- LSD-SLAM安装与运行

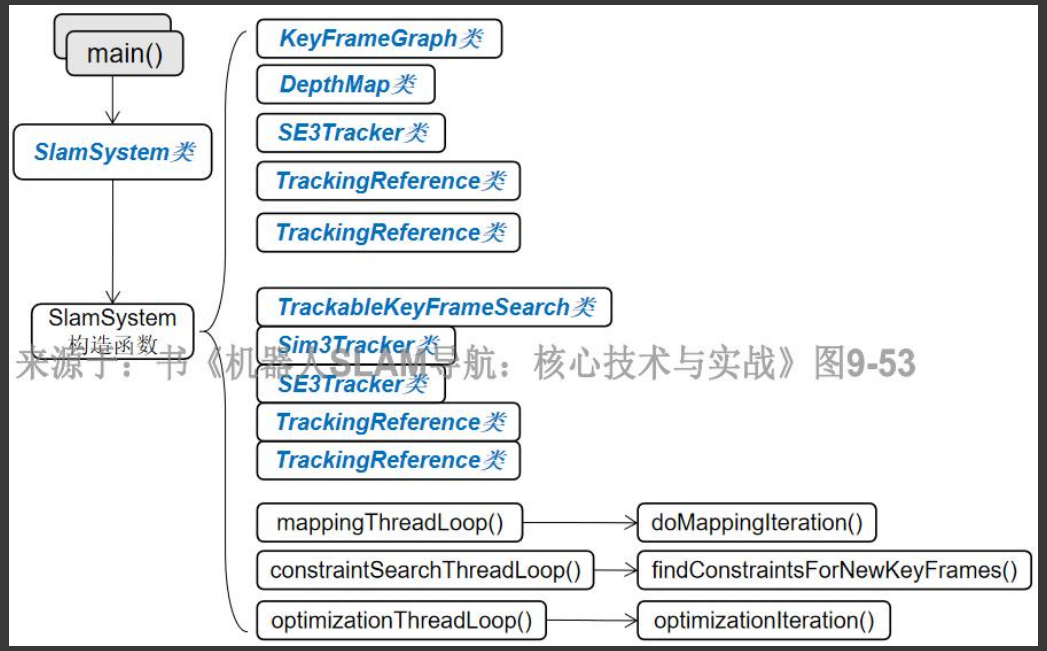


LSD-SLAM代码框架

9.2 LSD-SLAM算法

- LSD-SLAM原理分析
- LSD-SLAM源码解读
- LSD-SLAM安装与运行

SlamSystem类



LSD-SLAM核心库文件

用途		源文件	
算法顶层接口	SlamSystem 类	SlamSystem.cpp	SlamSystem.h
地图数据结构（DataStructures）	Frame 类	Frame.cpp	Frame.h
	FrameMemory 类	FrameMemory.cpp	FrameMemory.h
	FramePoseStruct 类	FramePoseStruct.cpp	FramePoseStruct.h
追踪（Tracking）	TrackingReference 类	TrackingReference.cpp	TrackingReference.h
	SE3Tracker 类	SE3Tracker.cpp	SE3Tracker.h
	Sim3Tracker 类	Sim3Tracker.cpp	Sim3Tracker.h
	Relocalizer 类	Relocalizer.cpp	Relocalizer.h
	OptimizedSelfAdjointMatrix6x6f 类 NormalEquationsLeastSquares 类 NormalEquationsLeastSquares4 类 NormalEquationsLeastSquares7 类	least_squares.cpp	least_squares.h
深度估计（DepthEstimation）	DepthMap 类	DepthMap.cpp	DepthMap.h
	DepthMapPixelHypothesis 类	DepthMapPixelHypothesis.cpp	DepthMapPixelHypothesis.h
地图优化（GlobalMapping）	KeyFrameGraph 类	KeyFrameGraph.cpp	KeyFrameGraph.h
	VertexSim3 类 EdgeSim3 类	g2oTypeSim3Sophus.cpp	g2oTypeSim3Sophus.h
	TrackableKeyFrameSearch 类	TrackableKeyFrameSearch.cpp	TrackableKeyFrameSearch.h
	FabMap 类	FabMap.cpp	FabMap.h
其他	IOWrapper 组件 util 组件	IOWrapper/* util/*	IOWrapper/* util/*

9.2 LSD-SLAM算法

- LSD-SLAM原理分析
- LSD-SLAM源码解读
- LSD-SLAM安装与运行

① LSD-SLAM安装
② LSD-SLAM离线运行
③ LSD-SLAM在线运行

https://github.com/tum-vision/lsd_slam/blob/master/README.md

9.2 LSD-SLAM算法

- LSD-SLAM原理分析
- LSD-SLAM源码解读
- LSD-SLAM安装与运行

① LSD-SLAM安装
② LSD-SLAM离线运行
③ LSD-SLAM在线运行

```
#启动dataset_slam建图
rosrun lsd_slam_core dataset_slam _files:=<files> _hz:=<hz> _calib:=<calibration_file>
#启动viewer显示
rosrun lsd_slam_viewer viewer
```

9.2 LSD-SLAM算法

- LSD-SLAM原理分析
- LSD-SLAM源码解读
- LSD-SLAM安装与运行

① LSD-SLAM安装
② LSD-SLAM离线运行
③ LSD-SLAM在线运行

```
#启动live_slam建图
rosrun lsd_slam_core live_slam /image:=<yourstreamtopic>  /camera_info:=<yourcamera_infotopic>
#启动viewer显示
rosrun lsd_slam_viewer viewer
```

* 不管是运行离线节点还是在线节点，LSD-SLAM中都提供的大量的ROS动态参数可以进行配置来调节算法的效果。

内容概要

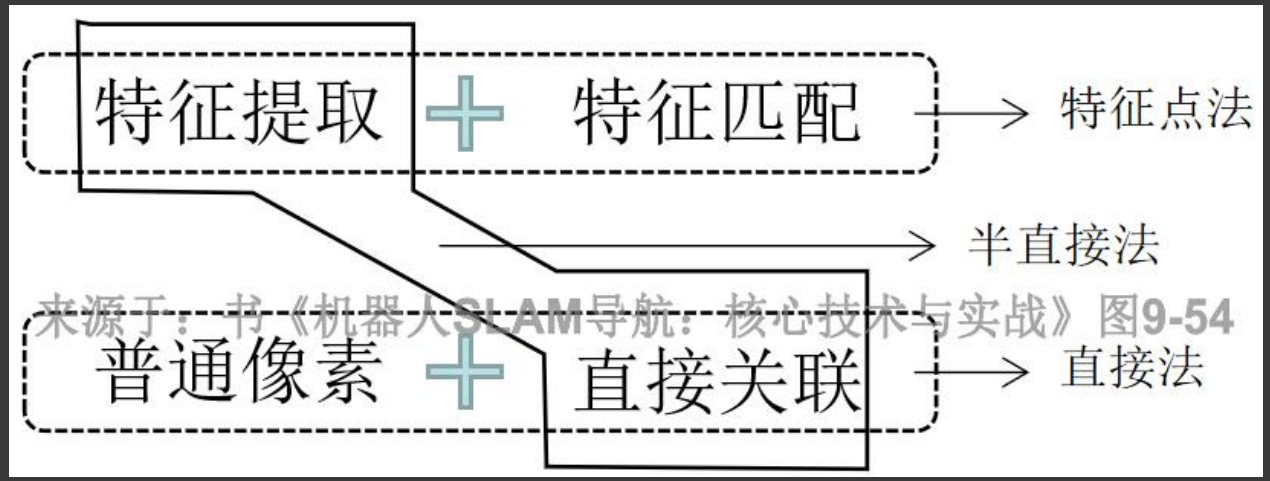
9.3 SVO算法

Forster C , Pizzoli M , Davide Scaramuzza* . SVO: Fast semi-direct monocular visual odometry[C]// IEEE International Conference on Robotics & Automation. IEEE, 2014.

SVO是什么？

9.3 SVO算法

- SVO原理分析 → 半直接法 → SVO系统框架
- SVO源码解读



来源于：书《机器人SLAM导航：核心技术与实战》图9-54

- 特征点法：

先通过特征提取和特征匹配建立两帧图像之间的数据关联，然后最小化重投影误差求解相机位姿和地图云点

- 直接法：

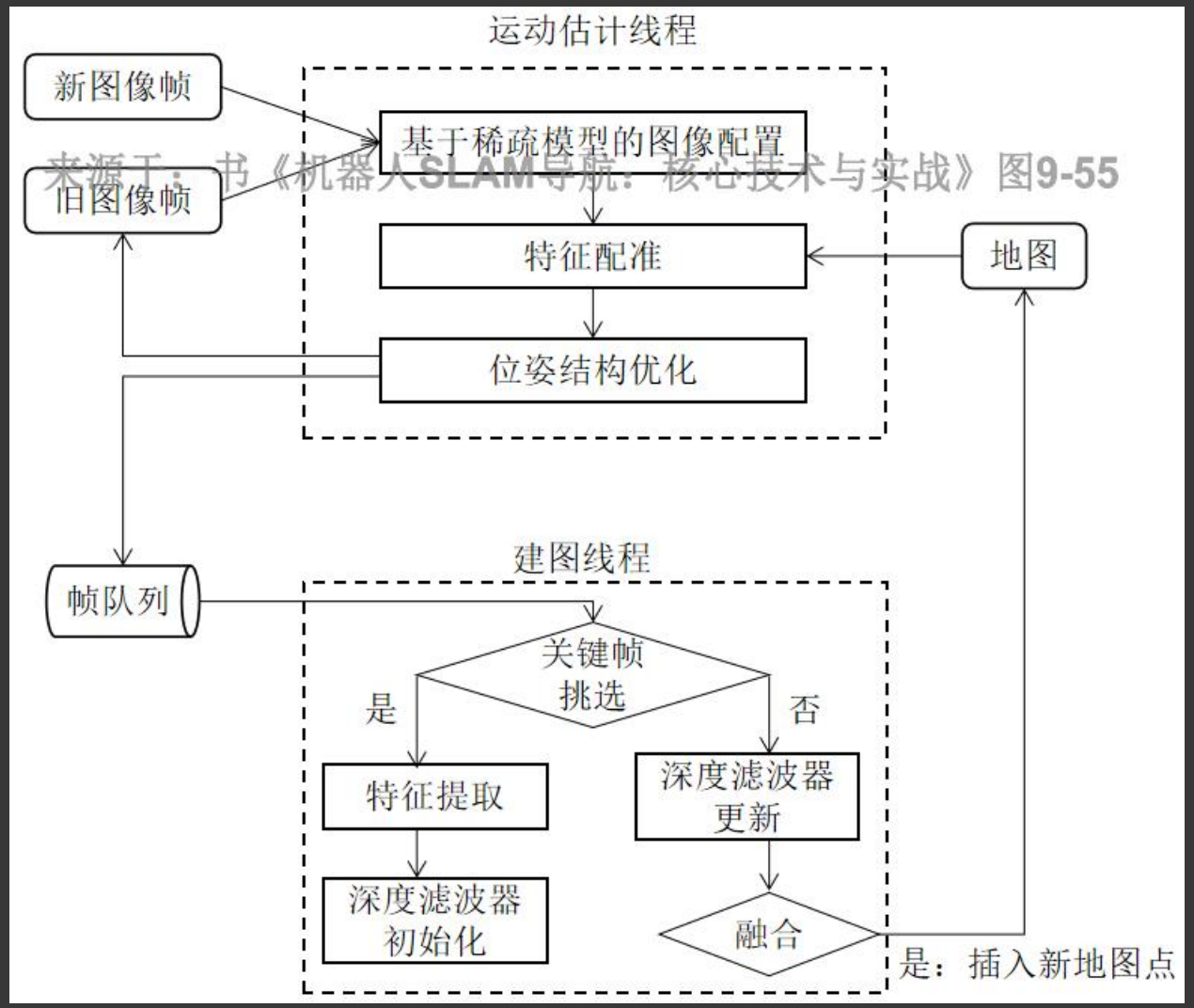
直接在两帧图像的像素上建立数据关联，然后最小光度误差求解相机位姿和地图云点

- 半直接法：

将特征点法中鲁棒性较好的特征提取部分与直接法中高效的直接关联部分结合起来，就是所谓的半直接法

9.3 SVO算法

- SVO原理分析 → 半直接法 SVO系统框架 →
- SVO源码解读



运动估计线程：

- 基于稀疏模型的图像配准
- 特征配准
- 位姿结构优化

建图线程：

- 深度估计

* 关于SVO系统框架的实现细节，将在第2季课程中详细讨论。

9.3 SVO算法

- SVO原理分析
- SVO源码解读

* 由于开源出来的SVO是经过阉割后的版本，并没有后端优化和闭环检测环节，严格意义上并不能称之为SLAM系统，确切点说只是一个VO而已。也就是说分析其源码[https://github.com/uzh-rpg/rpg_svo]的意义不大，感兴趣的读者可以按照上面讲过的思路自行分析即可。

- 例程源码下载： https://github.com/xiihoo/Books_Robot_SLAM_Navigation
- 课件PPT下载： www.xiihoo.com

敬请关注,长期更新...

下集预告