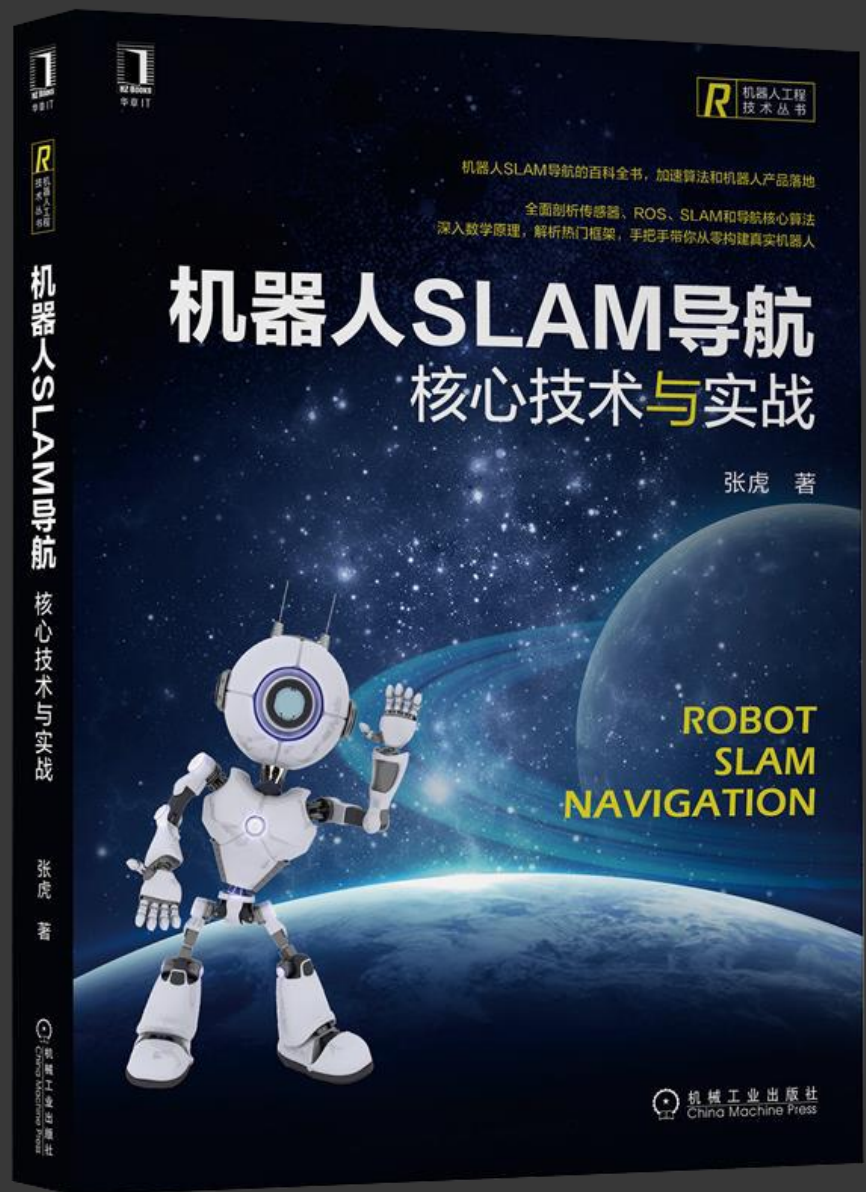




第1季

第10章：其他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

Q：第10章，是不是不重要？

A：很重要，很重要，很重要

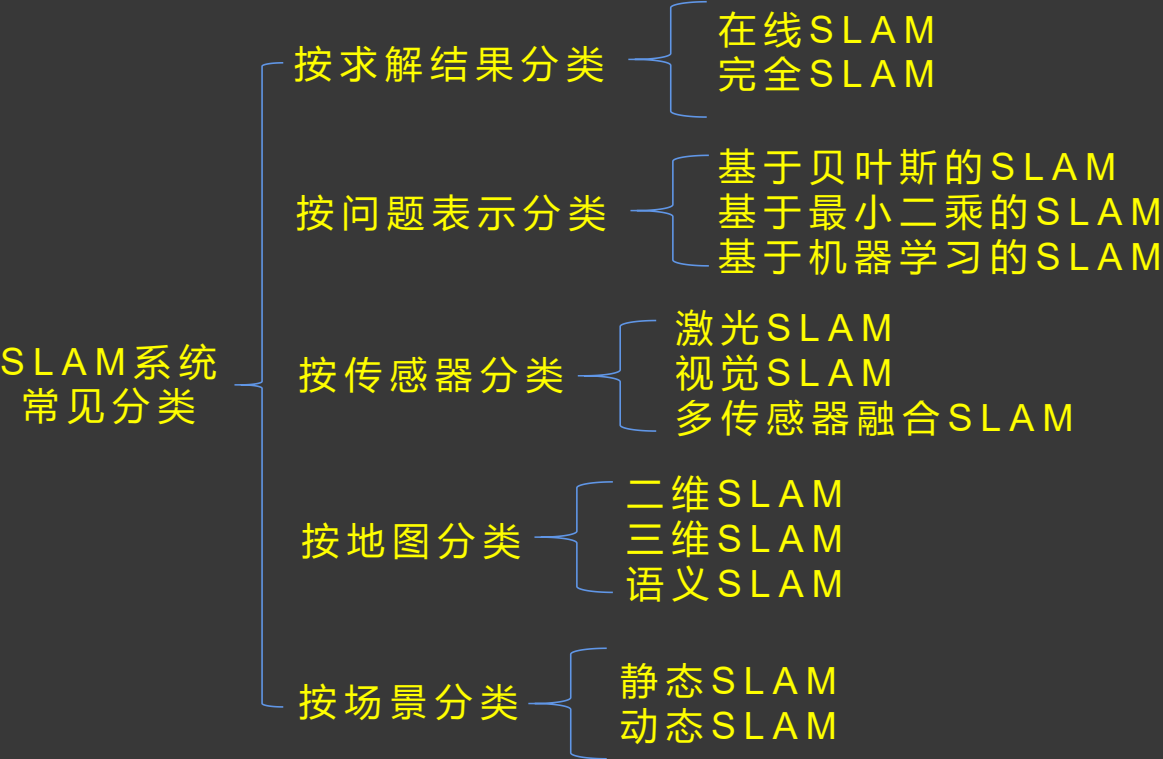
其他SLAM系统具体指什么？

问题	求解方法（也叫算法）	具体实现（也叫系统）
1+1=?	• 计算器 • 算盘 • 笔算 • 心算 • 扳手指 • ...	拨算盘的具体过程
SLAM (同时定位与建图)	• EKF • 粒子滤波 • 图优化 • 非线性优化 • 增量优化 • 端到端学习 • 生物启发 • ...	• 搭建硬件 • 编写代码

- SLAM并不仅仅指代某种特定算法（求解方法），SLAM是一种问题的总称，SLAM问题 ≠ SLAM算法
- 某种SLAM算法的软硬件具体实现，就是一种具体的SLAM系统，比如ORB-SLAM2、Cartographer等
- SLAM问题 > SLAM算法 > SLAM系统
- 一个SLAMer，还可能是程序员、数学家、硬件工程师、机械工程师、AI调包侠、...

其他SLAM系统具体指什么？

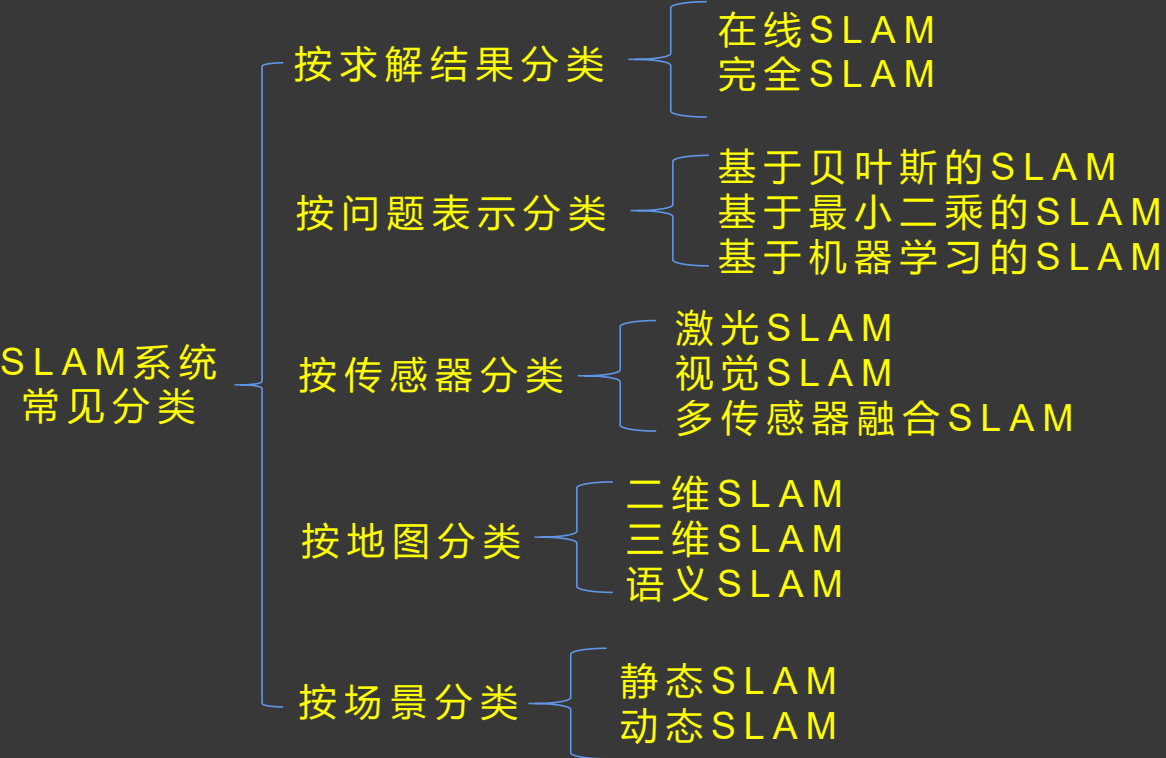
SLAM问题 > SLAM算法 > SLAM系统



		在线SLAM系统	完全SLAM系统
贝叶斯网络表示	EKF滤波	EKF-SLAM	
	粒子滤波		Fast-SLAM Gmapping
最小二乘表示	直接法		Graph-SLAM
	优化法		Cartographer LOAM ORB-SLAM
机器学习表示	端到端法	DeepVO	
	改进模块法	NeRF-SLAM NICE-SLAM	CNN-SLAM
	生物启发法	RatSLAM NeuroSLAM	

其他SLAM系统具体指什么？

SLAM问题 > SLAM算法 > SLAM系统



一种具体的SLAM系统，在不同的分类语境，有不同的称呼

内容概要

10.1 RTABMAP算法

10.2 VINS算法

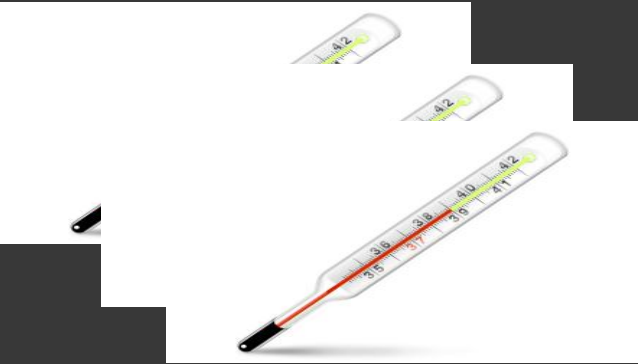
10.3 机器学习与SLAM

10.1 RTABMAP算法

为什么需要多传感器融合？
(增加信息量，取长补短)

→ { 多个同类传感器 (改进)
多个不同类传感器 (互补)

单传感器多次测量



假设独立同分布： $T_n = T + \varepsilon$,其中 $n = 1,2,3,..., N$, $\varepsilon \sim N(0, \sigma^2)$, $\sigma^2 \neq \infty$

求平均值

$$\hat{T}_{est} = \frac{T_1 + T_2 + T_3 + ... + T_N}{N}$$

$$\lim_{N \rightarrow \infty} \frac{\sigma^2}{N} = 0$$

$$\begin{aligned} D[\hat{T}_{est}] &= D[\frac{T_1 + T_2 + T_3 + ... + T_N}{N}] \\ &= \frac{D[T_1 + T_2 + T_3 + ... + T_N]}{N^2} \\ &= \frac{D[T_1] + D[T_2] + D[T_3] + ... + D[T_N]}{N^2} \\ &= \frac{\sigma^2 + \sigma^2 + \sigma^2 + ... + \sigma^2}{N^2} \\ &= \frac{N\sigma^2}{N^2} \\ &= \frac{\sigma^2}{N} \end{aligned}$$

当测量次数N趋于无穷大时，
总测量误差趋于0，也就是说
最终的测量值就是真实值。
在实际中，单传感器多次测量
很难实现。

温度测量值: T_1
温度测量值: T_2
温度测量值: T_3
⋮
温度测量值: T_N

$T_1, T_2, T_3, ..., T_N$ 都是随机变量;
而某个具体温度值 (比如10°、20°等) 是统计样本。

10.1 RTABMAP 算法

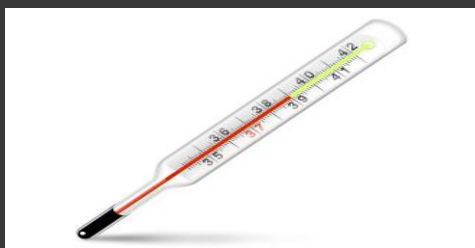
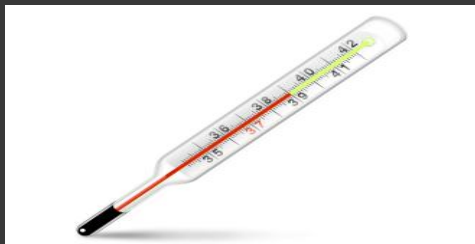
为什么需要多传感器融合？

(增加信息量，取长补短)

多个同类传感器 (改进)

多个不同类传感器 (互补)

多传感器单次测量

温度测量值: T_1 温度测量值: T_2

求平均值

假设独立同分布

$$T_1 = T + \varepsilon$$

$$T_2 = T + \varepsilon$$

其中: $\varepsilon \sim N(0, \sigma^2)$, $\sigma \neq \infty$

$$\hat{T}_{\text{est1}} = \frac{T_1 + T_2}{2}$$

$$\begin{aligned} D[\hat{T}_{\text{est1}}] &= D\left[\frac{T_1 + T_2}{2}\right] = \frac{D[T_1 + T_2]}{2^2} \\ &= \frac{D[T_1] + D[T_2]}{4} \\ &= \frac{\sigma^2 + \sigma^2}{4} \\ &= \frac{\sigma^2}{2} \end{aligned}$$

假设独立不同分布

$$T_1 = T + \varepsilon_1$$

$$T_2 = T + \varepsilon_2$$

其中:

$$\varepsilon_1 \sim N(0, \sigma_1^2), \varepsilon_2 \sim N(0, \sigma_2^2), E[\varepsilon_1 \varepsilon_2] = E[\varepsilon_1] \cdot E[\varepsilon_2] = 0, \sigma_1^2, \sigma_2^2 \neq \infty$$

$$\hat{T}_{\text{est1}} = \frac{T_1 + T_2}{2}$$

$$\begin{aligned} D[\hat{T}_{\text{est1}}] &= D\left[\frac{T_1 + T_2}{2}\right] = \frac{D[T_1 + T_2]}{2^2} \\ &= \frac{D[T_1] + D[T_2]}{4} \\ &= \frac{\sigma_1^2 + \sigma_2^2}{4} \end{aligned}$$

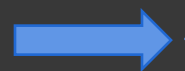
$$\frac{\sigma_1^2 + \sigma_2^2}{4} \text{ 与 } \sigma_1^2, \sigma_2^2 \text{ 比较大小?}$$

求平均值不一定是**最优**的

10.1 RTABMAP算法

为什么需要多传感器融合？

(增加信息量，取长补短)



多个同类传感器（改进）

多个不同类传感器（互补）

多传感器单次测量

二选一值

假设独立不同分布

$$T_1 = T + \varepsilon_1$$

$$T_2 = T + \varepsilon_2$$

其中：

$$\varepsilon_1 \sim N(0, \sigma_1^2), \varepsilon_2 \sim N(0, \sigma_2^2), E[\varepsilon_1 \varepsilon_2] = E[\varepsilon_1] \cdot E[\varepsilon_2] = 0, \sigma_1^2, \sigma_2^2 \neq \infty$$

if $D[T_1] < D[T_2]$

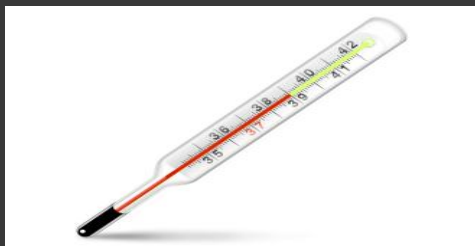
$$\hat{T}_{est2} = T_1$$

else

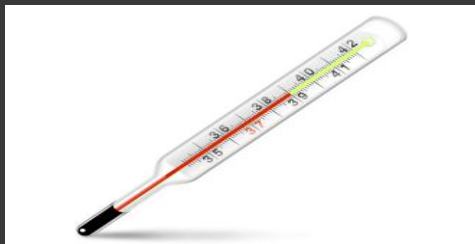
$$\hat{T}_{est2} = T_2$$

$$D[\hat{T}_{est2}] = \min(\sigma_1^2, \sigma_2^2)$$

二选一值实现了局部最优，但不一定是整体最优的



温度测量值： T_1



温度测量值： T_2

10.1 RTABMAP算法

为什么需要多传感器融合？

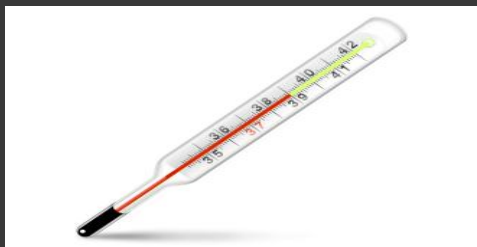
(增加信息量，取长补短)

多个同类传感器（改进）

多个不同类传感器（互补）

多传感器单次测量

加权平均值

温度测量值： T_1

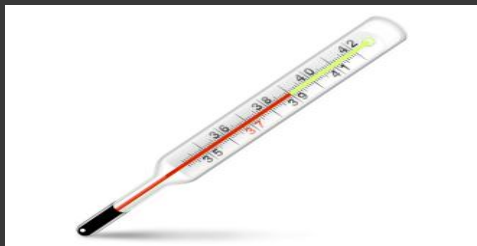
假设独立不同分布

$$T_1 = T + \varepsilon_1$$

$$T_2 = T + \varepsilon_2$$

其中：

$$\varepsilon_1 \sim N(0, \sigma_1^2), \varepsilon_2 \sim N(0, \sigma_2^2), E[\varepsilon_1 \varepsilon_2] = E[\varepsilon_1] \cdot E[\varepsilon_2] = 0, \sigma_1^2, \sigma_2^2 \neq \infty$$

温度测量值： T_2

$$\hat{T}_{\text{est3}} = \alpha \cdot T_1 + (1 - \alpha) \cdot T_2$$

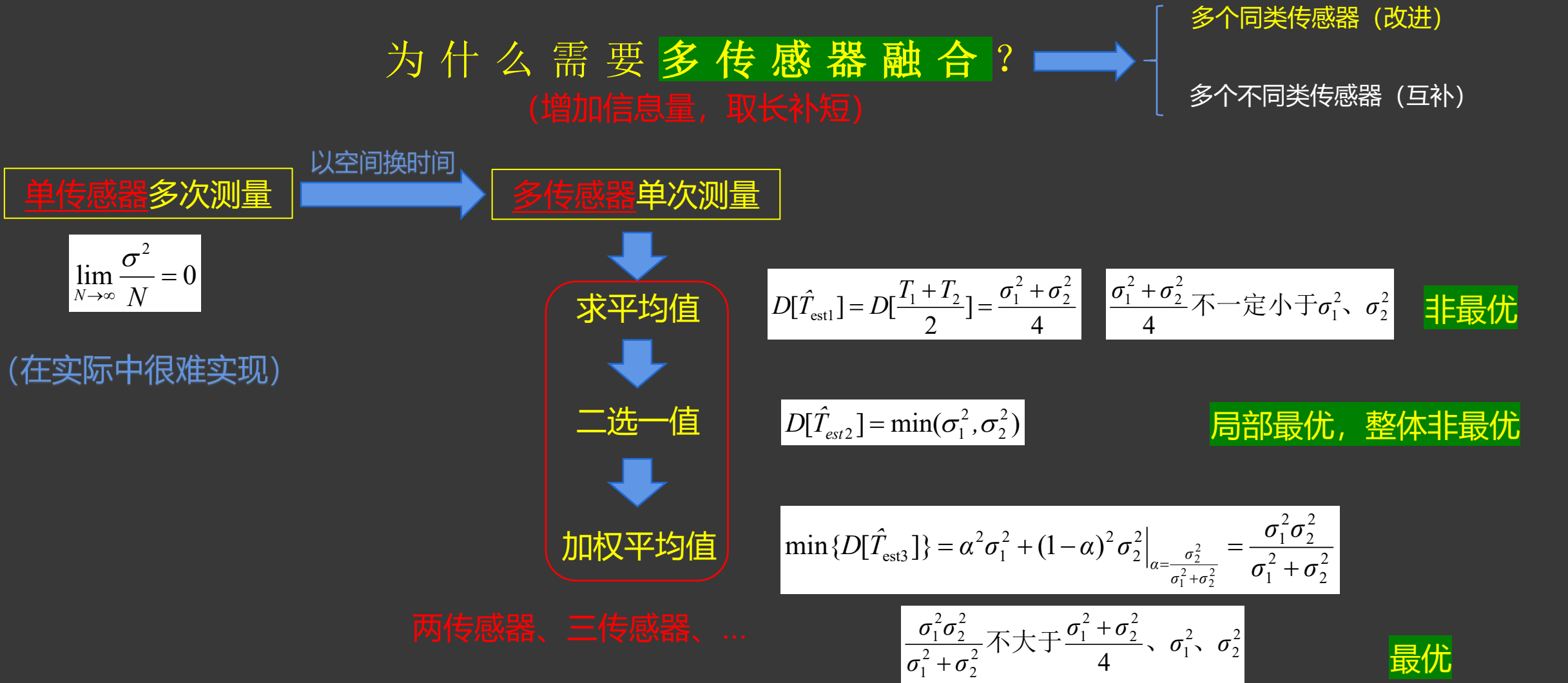
$$\begin{aligned} D[\hat{T}_{\text{est3}}] &= D[\alpha \cdot T_1 + (1 - \alpha) \cdot T_2] \\ &= \alpha^2 D[T_1] + (1 - \alpha)^2 D[T_2] \\ &= \alpha^2 \sigma_1^2 + (1 - \alpha)^2 \sigma_2^2 \end{aligned}$$

$$\begin{aligned} \min \{D[\hat{T}_{\text{est3}}]\} &= \alpha^2 \sigma_1^2 + (1 - \alpha)^2 \sigma_2^2 \Big|_{\alpha = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}} \\ &= \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \end{aligned}$$

$$\frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \text{ 与 } \frac{\sigma_1^2 + \sigma_2^2}{4}, \sigma_1^2, \sigma_2^2 \text{ 比较大小?}$$

加权平均值可以达到最优

10.1 RTABMAP 算法



10.1 RTABMAP算法

为什么需要多传感器融合？
(增加信息量，取长补短)

➡

多个同类传感器（改进）

多个不同类传感器（互补）



听诊

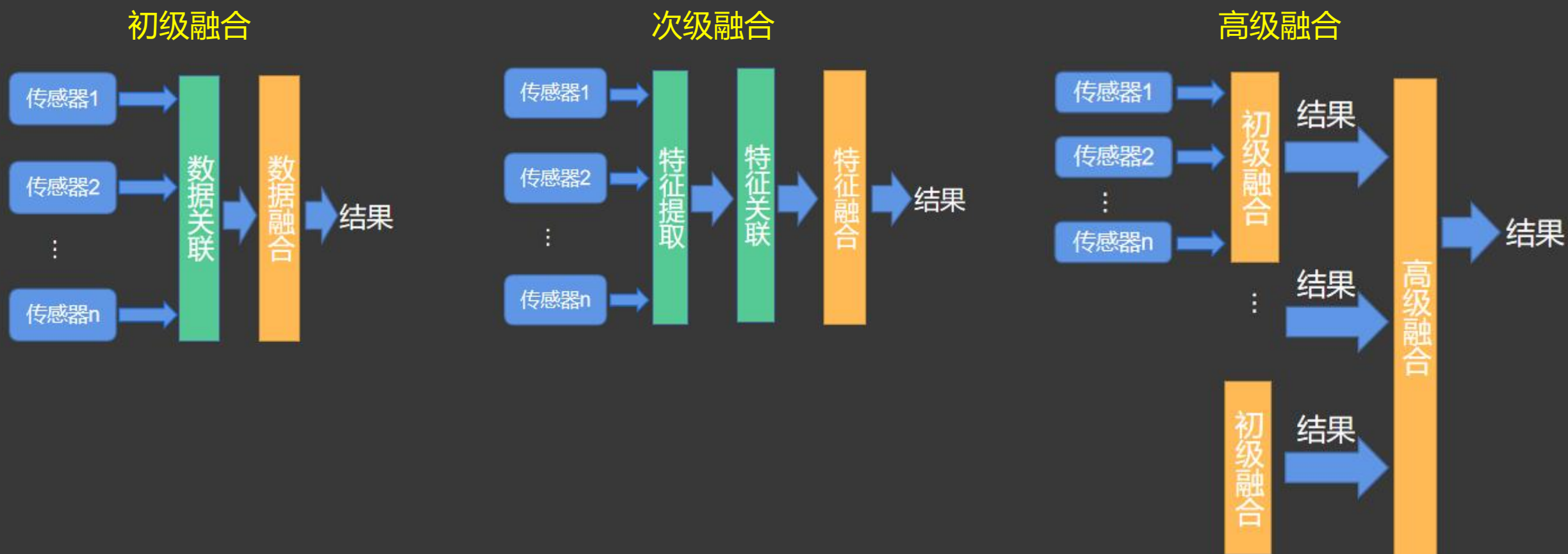


测体温

听诊	测体温	感冒可能性
正常	正常	0%
正常	异常	50%
异常	正常	30%
异常	异常	80%

10.1 RTABMAP算法

多传感器融合算法，有哪些？



10.1 RTABMAP算法

多传感器融合算法，有哪些？



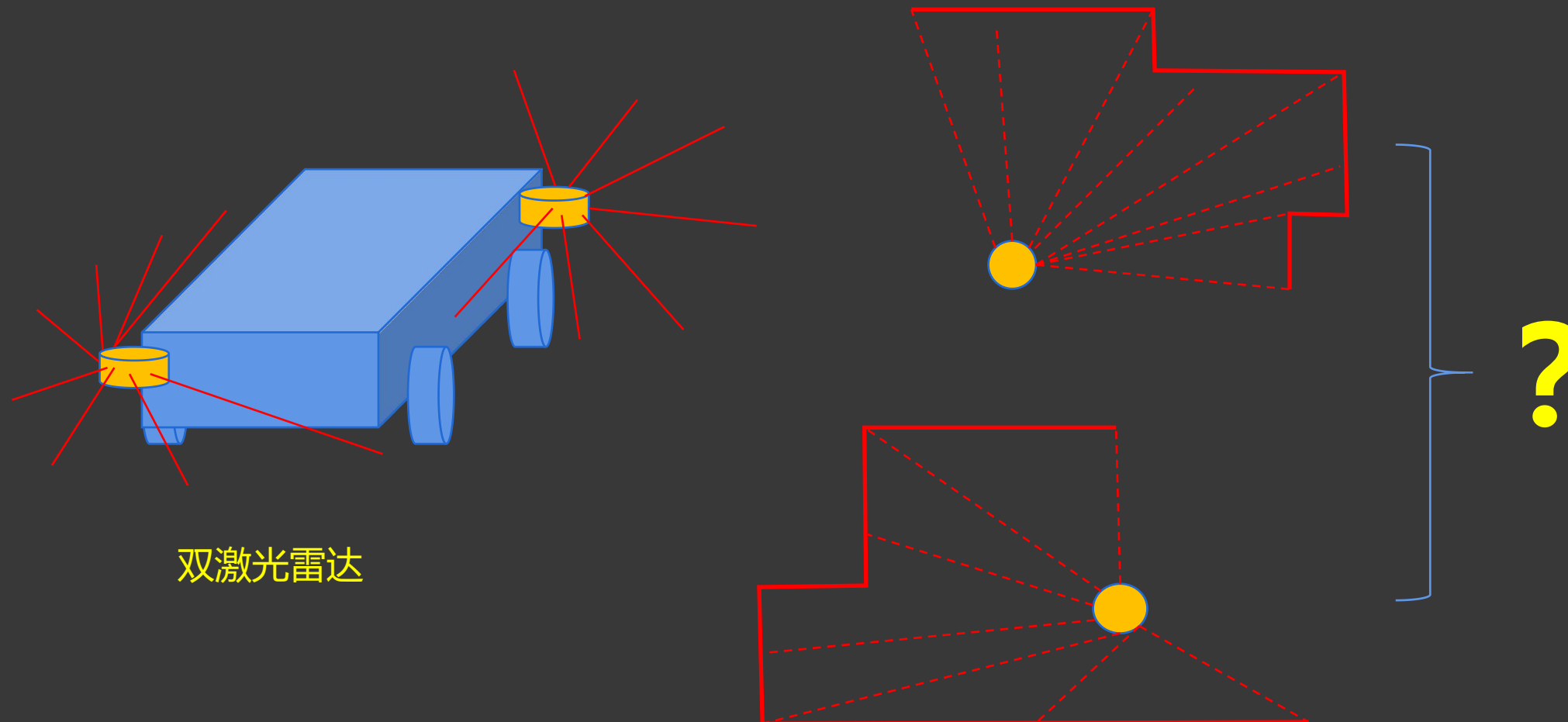
什么是数据关联？什么是数据融合？

数据关联：好比大象不同部位之间的拼接关系

数据融合：好比将大象不同部位，按拼接关系，拼接成完整的大象

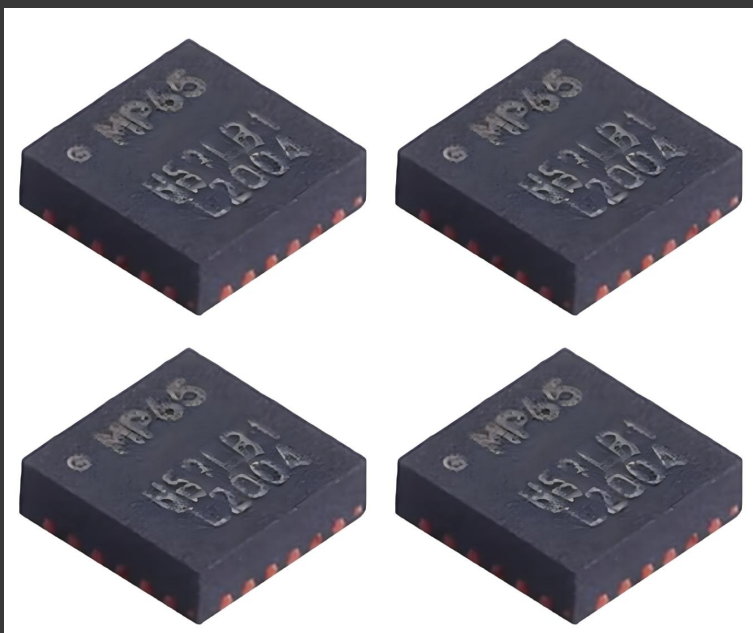
10.1 RTABMAP算法

机器人中无处不在的多传感器融合



10.1 RTABMAP算法

机器人中无处不在的多传感器融合



阵列IMU

假设独立同分布

$$ACC_n = ACC + \varepsilon, \quad n = 1, 2, 3, 4$$

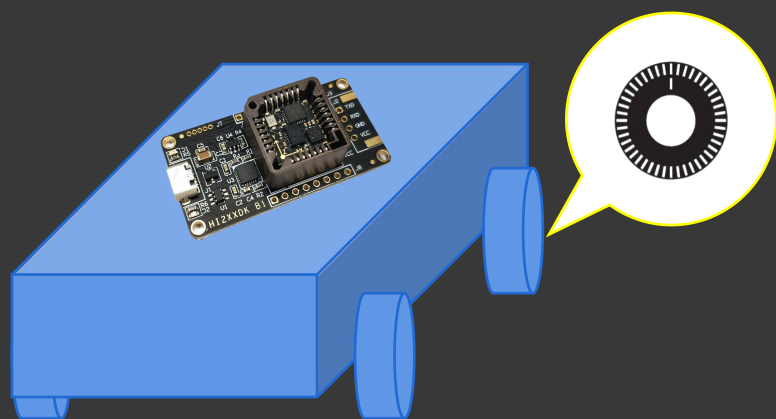
其中: $\varepsilon \sim N(0, \sigma^2)$, $\sigma \neq \infty$

$$ACC_{\text{est}} = \frac{ACC_1 + ACC_2 + ACC_3 + ACC_4}{4}$$

$$\begin{aligned} D[ACC_{\text{est}}] &= D\left[\frac{ACC_1 + ACC_2 + ACC_3 + ACC_4}{4}\right] \\ &= \frac{D[ACC_1] + D[ACC_2] + D[ACC_3] + D[ACC_4]}{4^2} \\ &= \frac{\sigma^2 + \sigma^2 + \sigma^2 + \sigma^2}{4^2} \\ &= \frac{\sigma^2}{4} \end{aligned}$$

10.1 RTABMAP算法

机器人中无处不在的多传感器融合



编码里程计惯导融合



10.1 RTABMAP算法

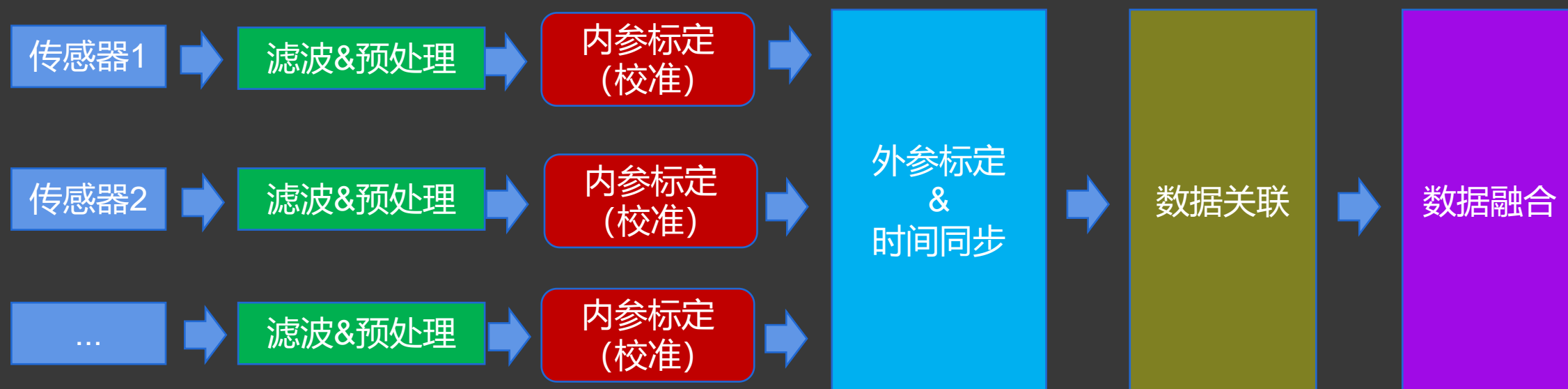
机器人中无处不在的多传感器融合



多传感器融合
似乎成了万金油

10.1 RTABMAP算法

机器人中无处不在的多传感器融合



10.1 RTABMAP算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行

	Inputs						Online Outputs				
	Camera				Lidar		Odom	Pose	Occupancy		Point Cloud
	Stereo	RGB-D	Multi	IMU	2D	3D			2D	3D	
GMapping					✓		✓	✓	✓		
TinySLAM					✓		✓	✓	✓		
Hector SLAM					✓			✓	✓		
ETHZASL-ICP					✓	✓	✓	✓	✓		Dense
Karto SLAM					✓		✓	✓	✓		
Lago SLAM					✓		✓	✓	✓		
Cartographer					✓	✓	✓	✓	✓		Dense
BLAM						✓		✓			Dense
SegMatch						✓					Dense
VINS-Mono				✓				✓			
ORB-SLAM2	✓	✓									
S-PTAM	✓							✓			Sparse
DVO-SLAM		✓						✓			
RGBiD-SLAM		✓									
MCPTAM	✓		✓					✓			Sparse
RGBDSLAMv2		✓					✓	✓		✓	Dense
RTAB-Map	✓	✓	✓		✓	✓	✓	✓	✓	✓	Dense

关于RTABMAP:

Real-Time Appearance-Based Mapping

- 采用基于优化的方法来求解SLAM问题
- 遵循前端里程计、后端优化和闭环检测的三段式范式
- 支持视觉和激光融合
- 内存管理机制

10.1 RTABMAP算法

- RTABMAP原理分析

■ RTABMAP源码解读

■ RTABMAP安装与运行
- RTABMAP系统框架

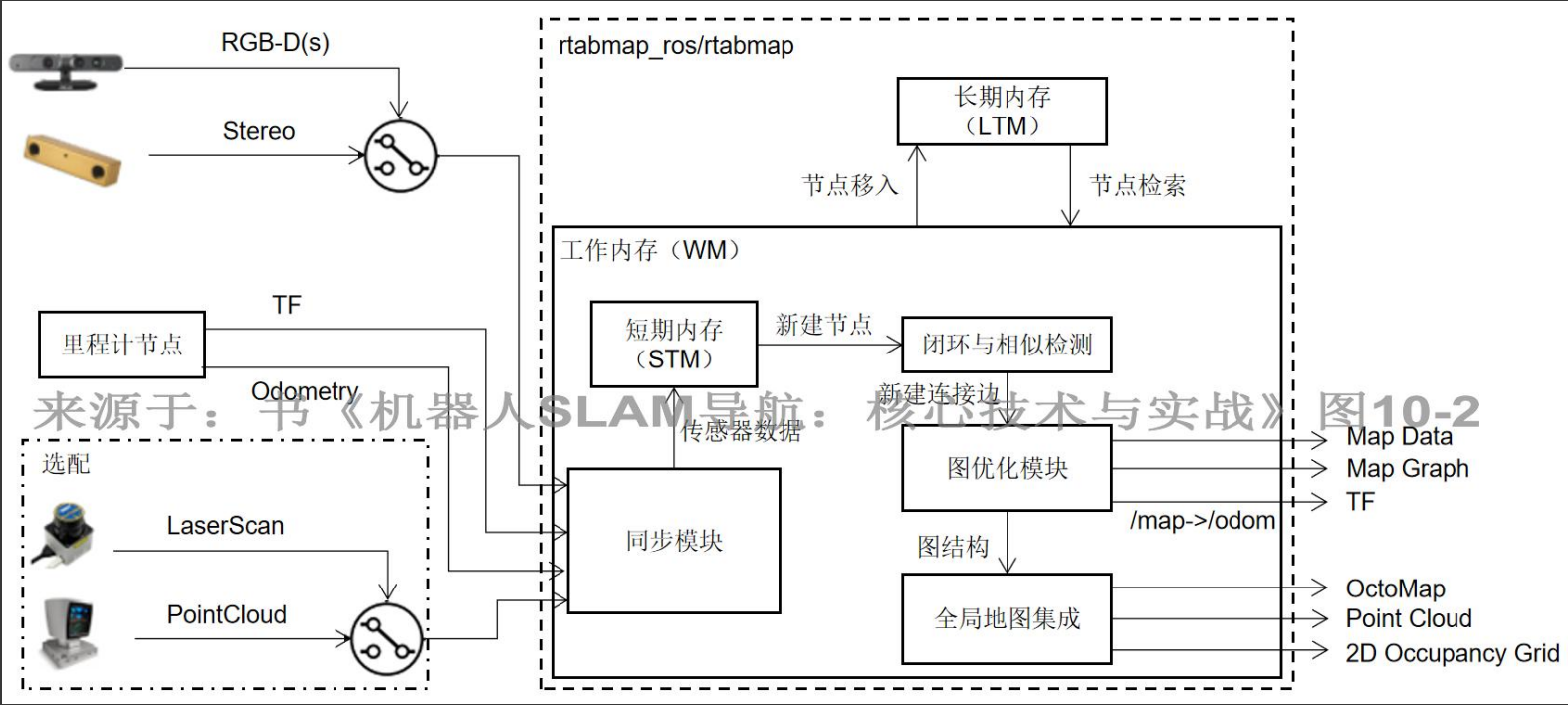
内存管理机制

里程计节点

局部地图

闭环检测与图优化

全局地图



- 输入

支持视觉、激光和里程计这3种传感器数据的输入，其中视觉是必备的传感器，激光是选配的传感器，而里程计由一个单独的ROS节点（Odometry Node）来提供
- 输出

最终输出的全局地图有3种格式，即OctoMap（3D Occupancy Grid）、Point Cloud和2D Occupancy Grid可选

10.1 RTABMAP算法

■ RTABMAP原理分析

■ RTABMAP源码解读

■ RTABMAP安装与运行

RTABMAP系统框架

内存管理机制

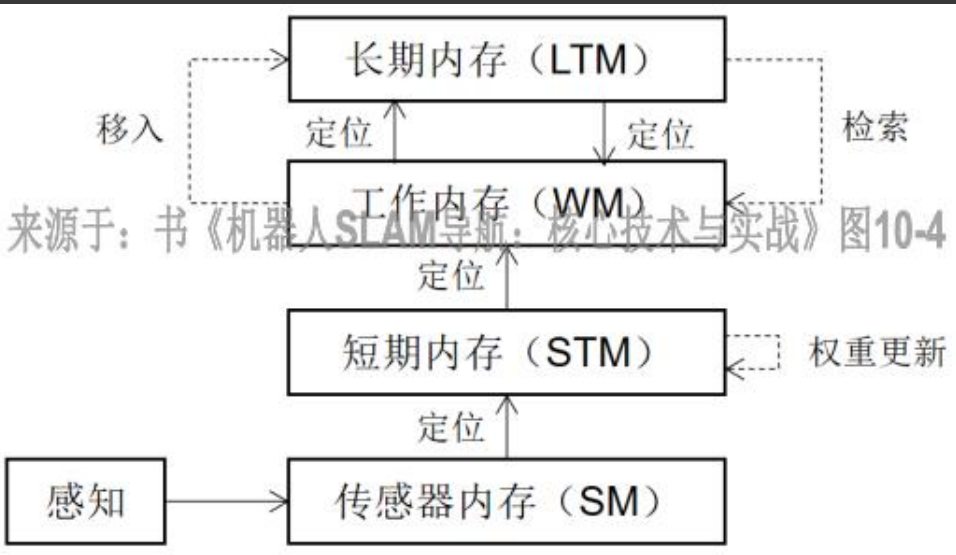
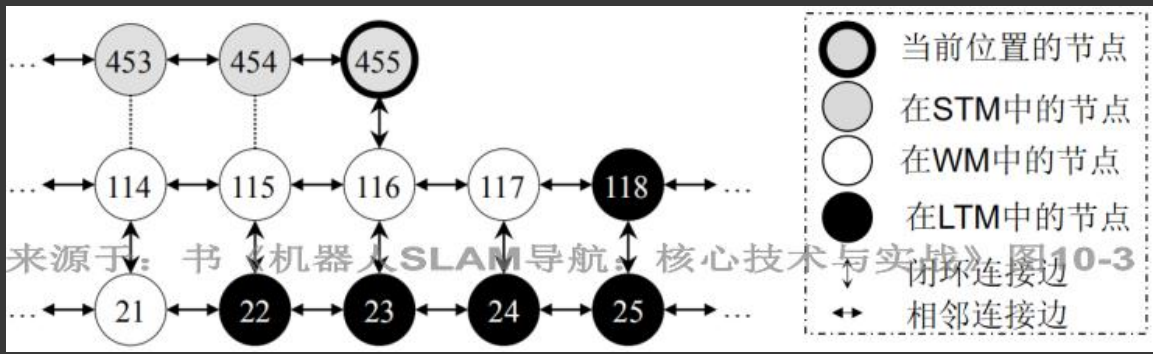
里程计节点

局部地图

闭环检测与图优化

全局地图

- RTABMAP采用图结构（Graph）来组织地图，也就是每进来一帧传感器数据都会创建一个对应节点用于存储相关数据，当建图规模很大时，所创建的节点数量将非常巨大，如果在全部节点上搜索闭环和做全局优化计算实时性会很糟糕。
- RTABMAP的内存管理机制其实就是分级管理这些节点，以保证大规模地图上闭环检测和全局优化的实时性。



10.1 RTABMAP算法

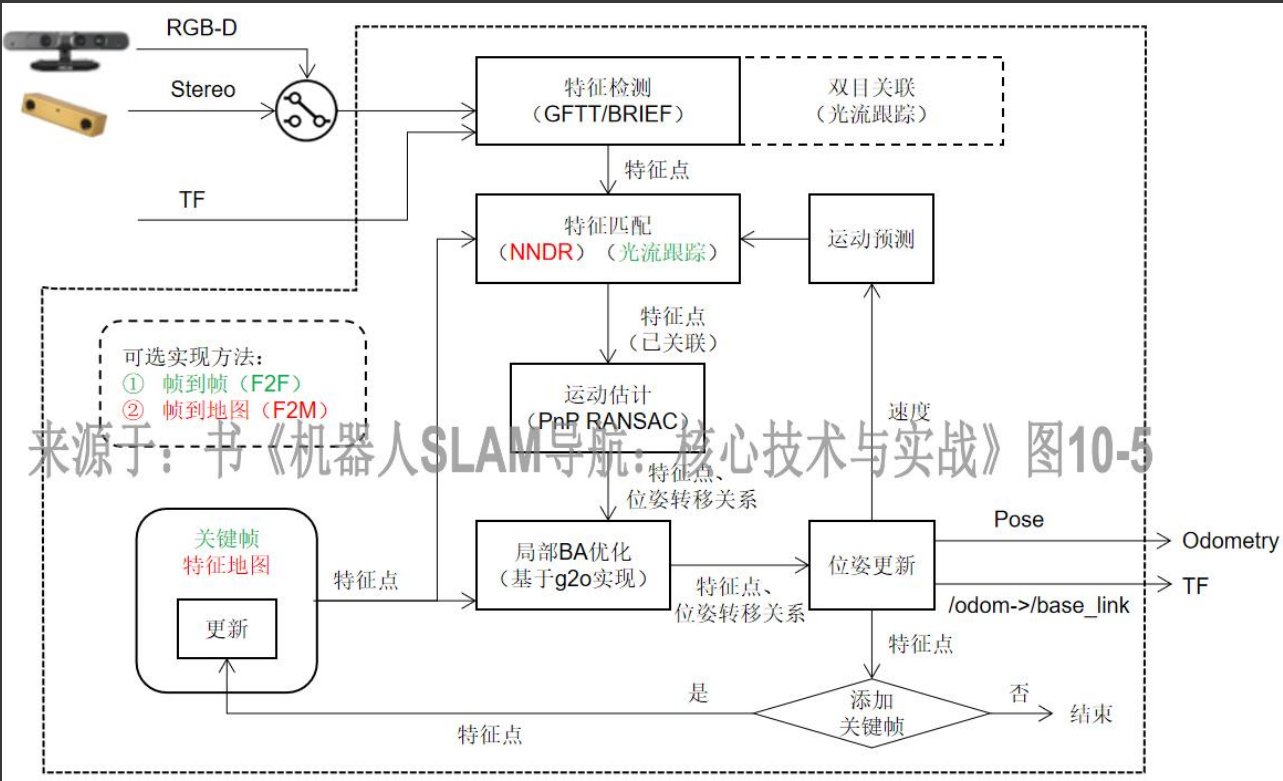
RTABMAP原理分析

RTABMAP源码解读

RTABMAP安装与运行

RTABMAP系统框架
内存管理机制
里程计节点
局部地图
闭环检测与图优化
全局地图

- 通过一个单独的ROS节点（Odometry Node）来为RTABMAP提供里程计信息，里程计信息最直接的来源就是机器人轮子上安装的编码器，也叫轮式里程计。
- 当没有轮式里程计或者其精度不够时，就需要启用视觉里程计（Visual Odometry）或者激光里程计（Lidar Odometry）。

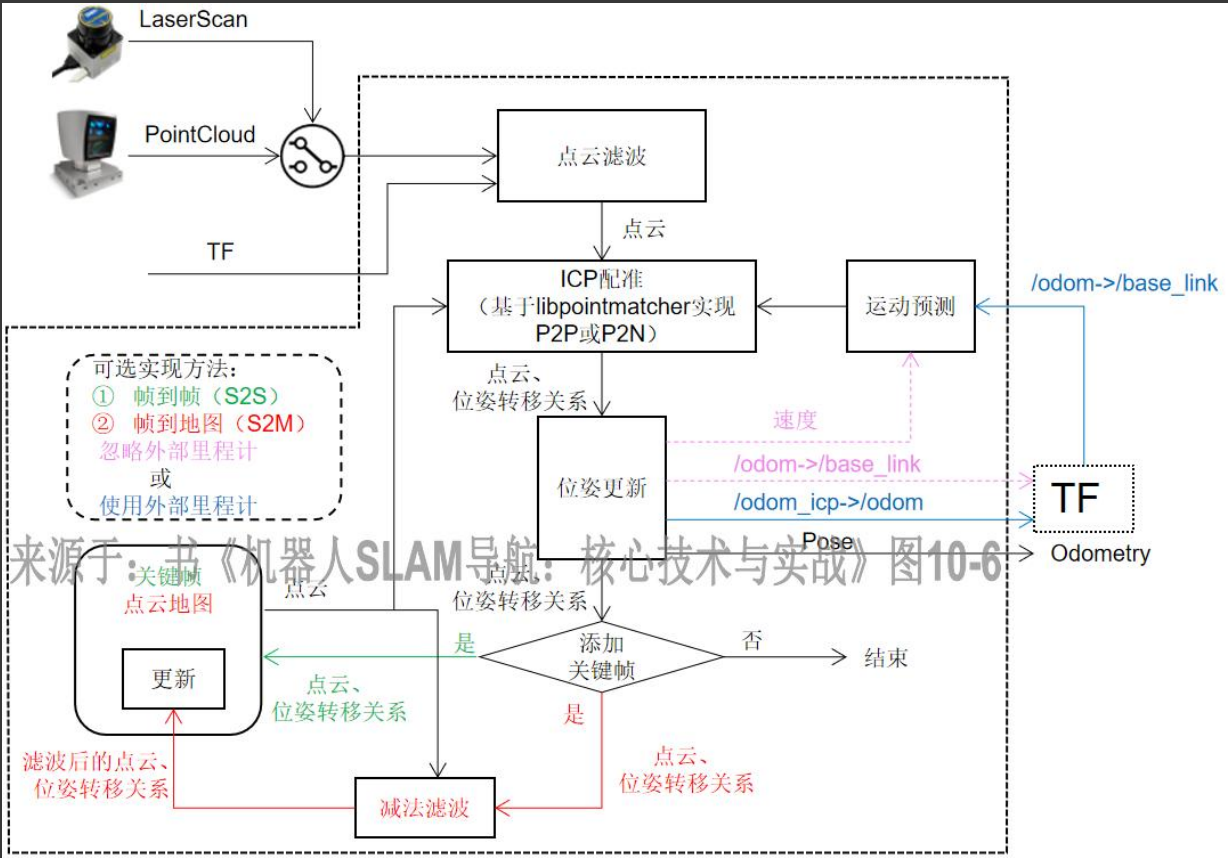


10.1 RTABMAP算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行

- RTABMAP系统框架
- 内存管理机制
- 里程计节点
- 局部地图
- 闭环检测与图优化
- 全局地图

- 通过一个单独的ROS节点（Odometry Node）来为RTABMAP提供里程计信息，里程计信息最直接的来源就是机器人轮子上安装的编码器，也叫轮式里程计。
- 当没有轮式里程计或者其精度不够时，就需要启用视觉里程计（Visual Odometry）或者激光里程计（Lidar Odometry）。



来源于: 书《机器人SLAM导航: 核心技术与实战》图10-6

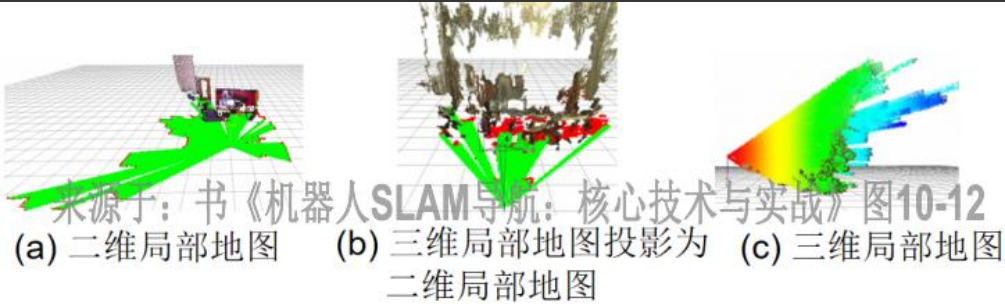
10.1 RTABMAP算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行

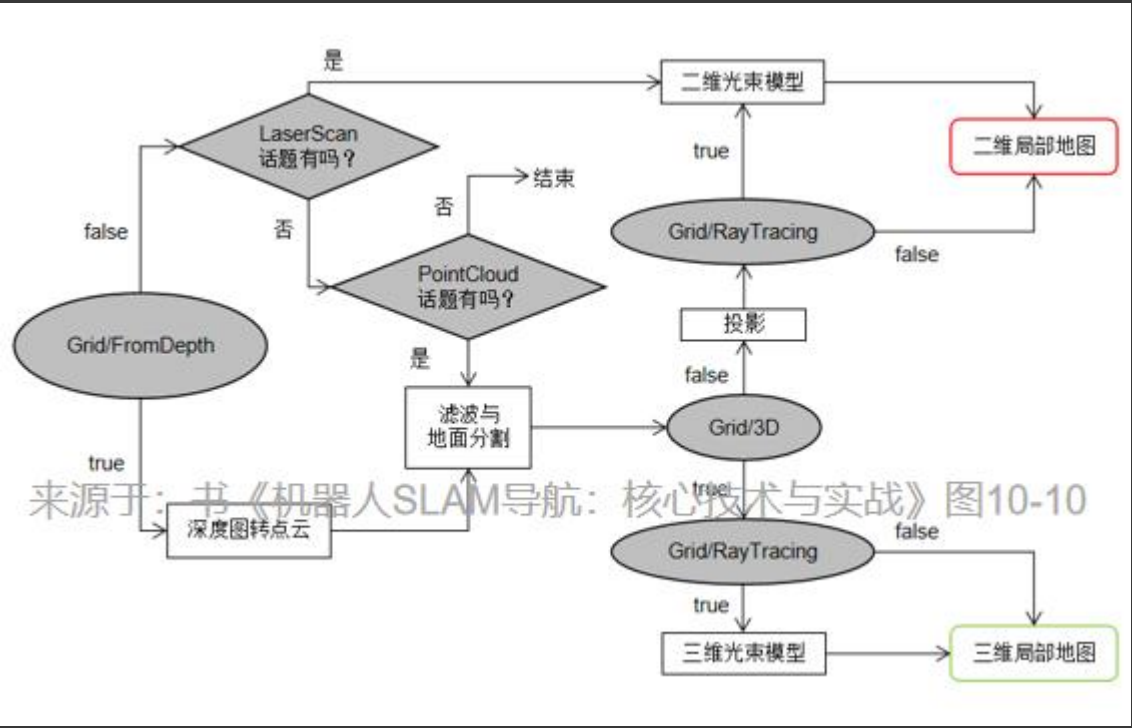
RTABMAP系统框架
内存管理机制
里程计节点
局部地图
闭环检测与图优化
全局地图

RTABMAP
工作模式

- RGB-D 工作模式
- 双目 工作模式
- RGB-D+单线激光雷达+轮式里程计 工作模式
- ...



- 当STM中一个新的节点被创建时，会利用深度图像（depth image）、激光扫描数据（laser scan）、点云数据（point cloud）等生成一个对应的局部地图（local occupancy grid）。



10.1 RTABMAP算法

- RTABMAP原理分析

■ RTABMAP源码解读

■ RTABMAP安装与运行
- RTABMAP系统框架

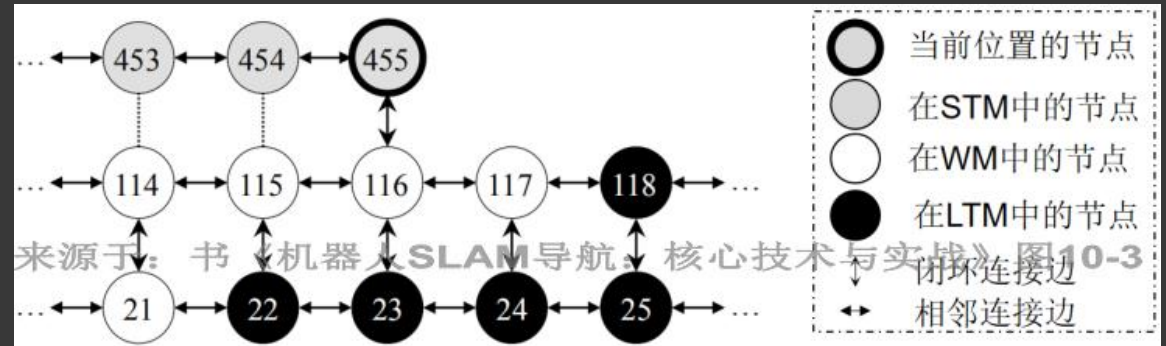
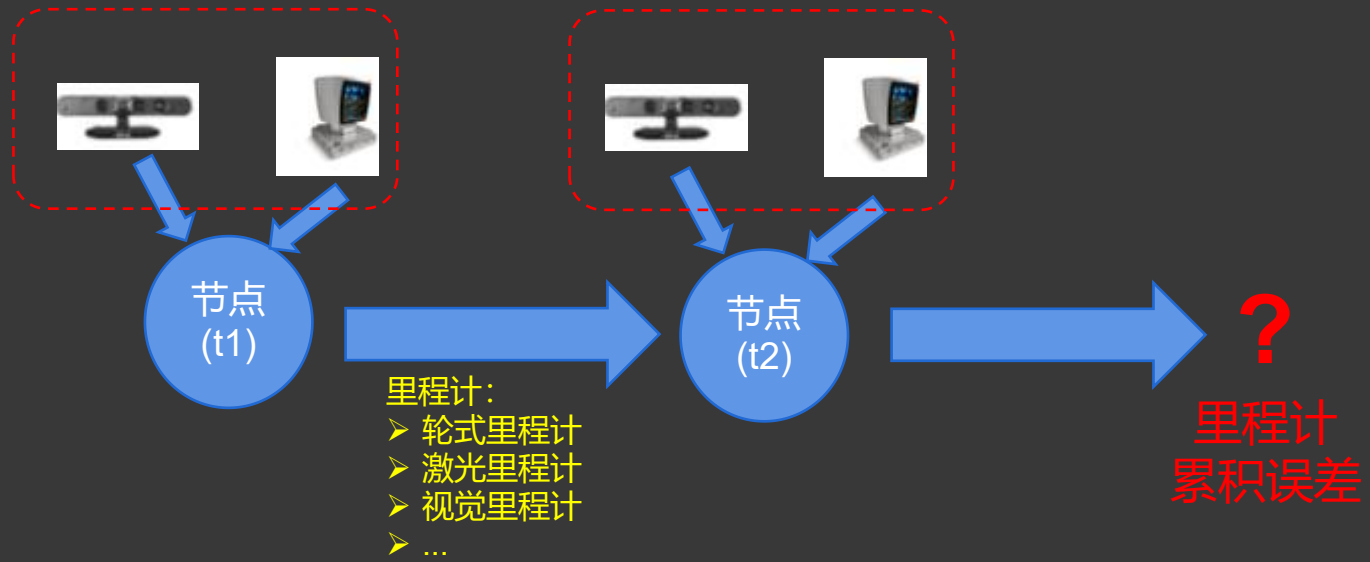
内存管理机制

里程计节点

局部地图

闭环检测与图优化

全局地图



位姿图优化

10.1 RTABMAP算法

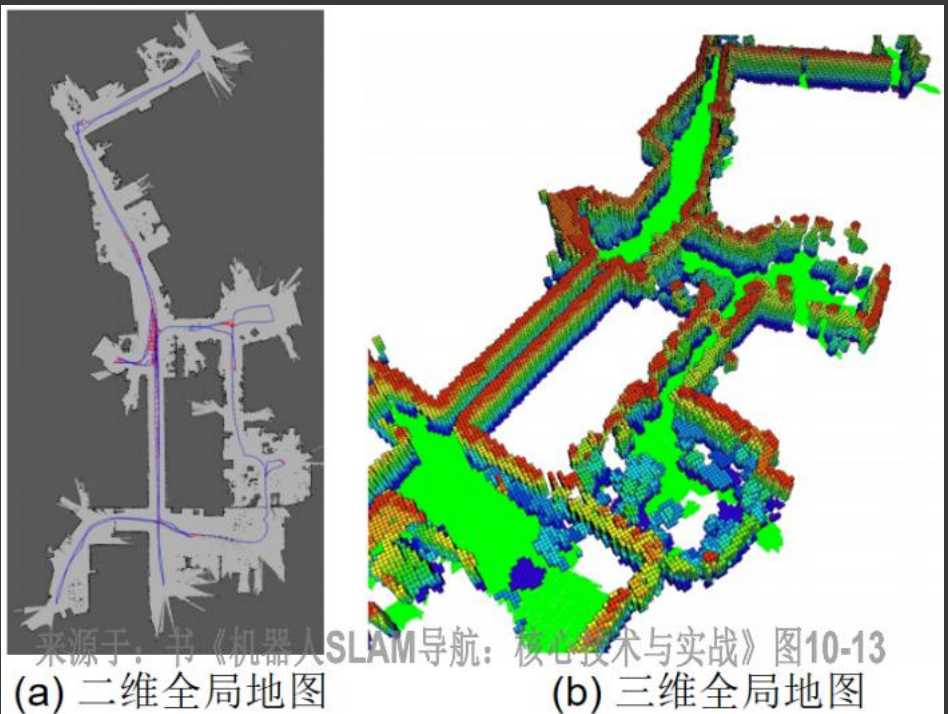
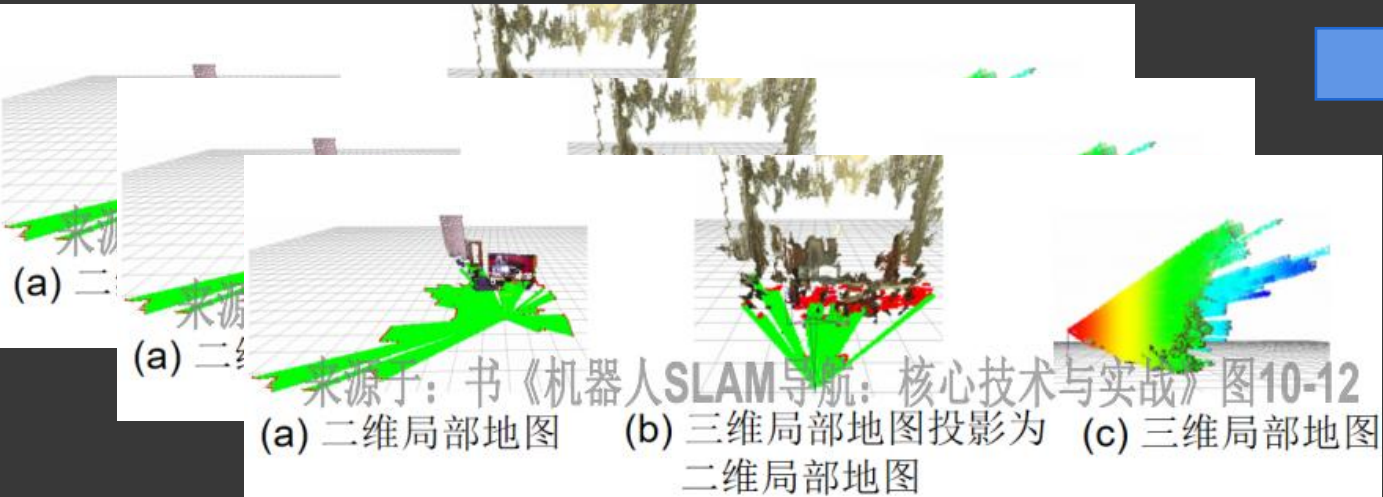
■ RTABMAP原理分析

■ RTABMAP源码解读

■ RTABMAP安装与运行

RTABMAP系统框架
内存管理机制
里程计节点
局部地图
闭环检测与图优化
全局地图

- 当STM中一个新的节点被创建时，会利用深度图像（depth image）、激光扫描数据（laser scan）、点云数据（point cloud）等生成一个对应的局部地图（local occupancy grid）。
- 局部建图所基于的里程计，不管是轮式里程计还是视觉里程计或激光里程计都存在累积误差，因此需要进行回环和全局优化。
- 当全局优化完成后，就可以基于各个节点的里程计位姿，将各个局部地图拼接起来得到一张全局地图。

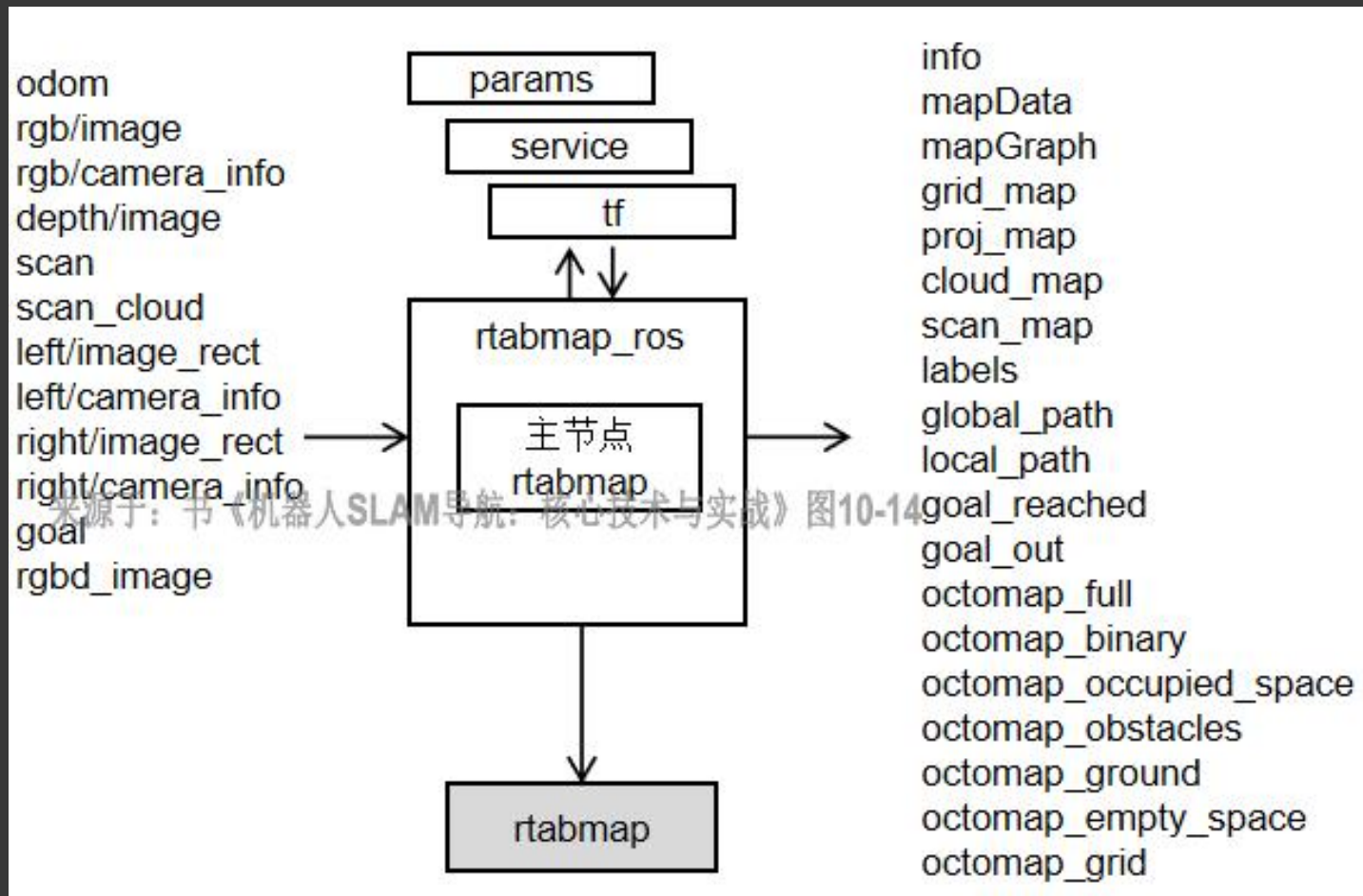


根据程序中不同的配置参数，可以选择RTABMAP是构建二维局部地图（2D local occupancy grid）还是三维局部地图（3D local occupancy grid）

10.1 RTABMAP算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行

RTABMAP代码框架



- 与前面已经分析过的算法类似，RTABMAP的代码由接口部分和核心库部分组成。
- rtabmap_ros功能包实现了算法的ROS接口，其中的主节点rtabmap负责启动算法主逻辑，而算法的底层具体实现被封装在rtabmap核心库之中。

rtabmap_ros功能包中包含的节点

10.1 RTABMAP算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行

节点	源码	功能
rtabmap	src/CoreNode.cpp	建图主节点
rtabmapviz	src/GuiNode.cpp	建图结果可视化
	src/GuiWrapper.cpp	
	src/PreferencesDialogROS.cpp	
rgbd_odometry	src/RGBDOdometryNode.cpp	提供视觉或激光里程计
stereo_odometry	src/StereoOdometryNode.cpp	
rgbdicp_odometry	src/RGBDICPOdometryNode.cpp	
icp_odometry	src/ICPOdometryNode.cpp	
rgbd_sync	src/RGBDSyncNode.cpp	相机图像数据预处理
stereo_sync	src/StereoSyncNode.cpp	
rgb_sync	src/RGBSyncNode.cpp	
rgbd_relay	src/RGBDRelayNode.cpp	
camera	src/CameraNode.cpp	从相机或文件获取图像数据
stereo_camera	src/StereoCameraNode.cpp	
data_player	src/DbPlayerNode.cpp	播放数据集
map_optimizer	src/MapOptimizerNode.cpp	地图优化
map_assembler	src/MapAssemblerNode.cpp	地图拼接
point_cloud_assembler	src/PointCloudAssemblerNode.cpp	点云拼接
imu_to_tf	src/ImuToTFNode.cpp	数据转换
odom_msg_to_tf	src/OdomMsgToTFNode.cpp	
pointcloud_to_depthimage	src/PointCloudToDepthImageNode.cpp	
wifi_signal_pub	src/WifiSignalPubNode.cpp	在建图时添加用户数据
wifi_signal_sub	src/WifiSignalSubNode.cpp	
save_objects_example	src/SaveObjectsExample.cpp	物体识别示例
external_loop_detection_example	src/ExternalLoopDetectionExample.cpp	外部闭环示例

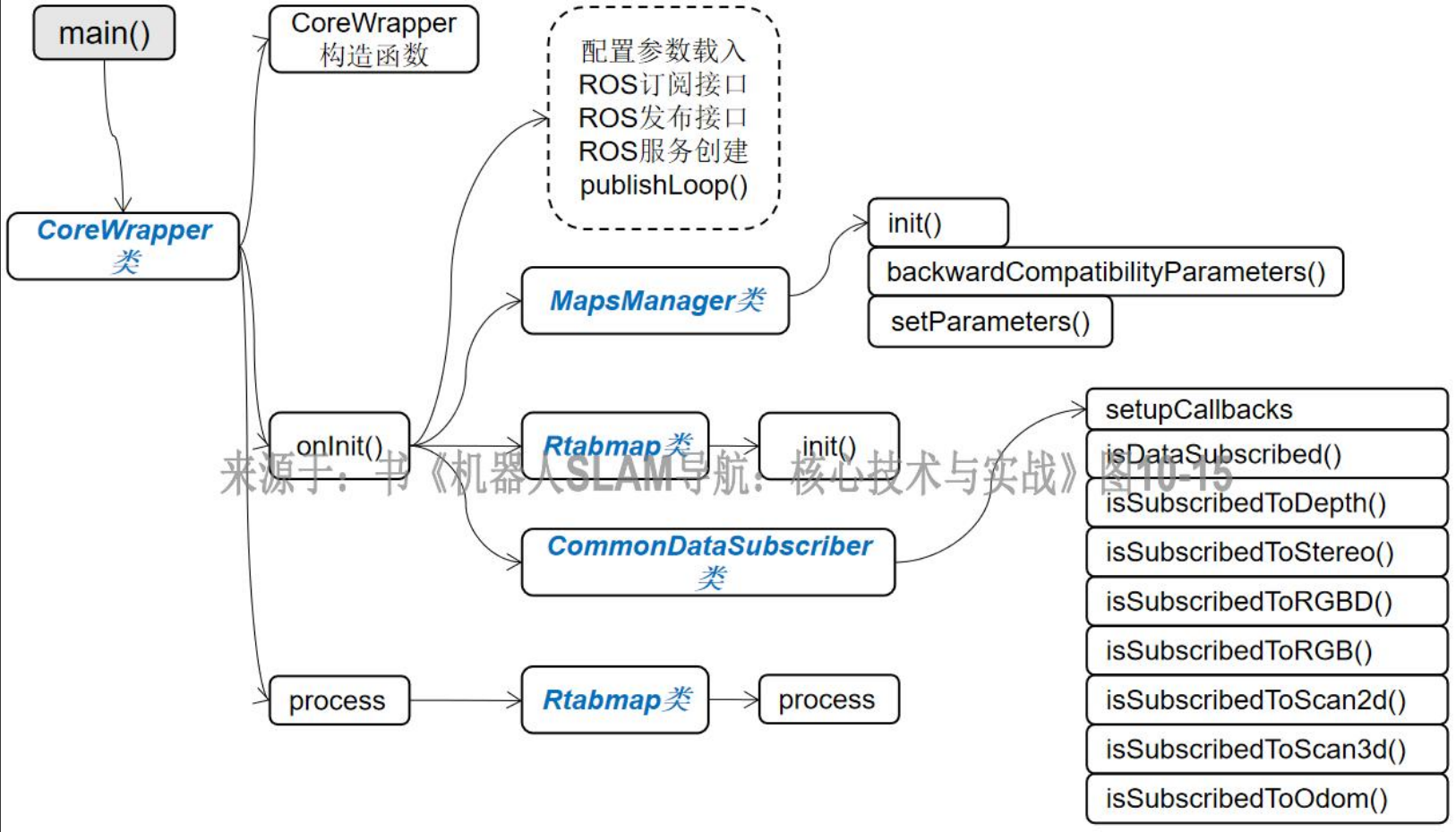
rtabmap_ros功能包中包含的plugin:

- nodelet
- src/nodelets/*
- src/CoreWrapper.cpp
- costmap_2d
- src/costmap_2d/*
- rviz
- src/rviz/*

ROS支持功能包动态加载和卸载，这个功能由一个C++的pluginlib库来实现，不用改动源码和重编译就能动态加载和卸载功能，使用plugin扩展和升级程序很便捷。

10.1 RTABMAP算法

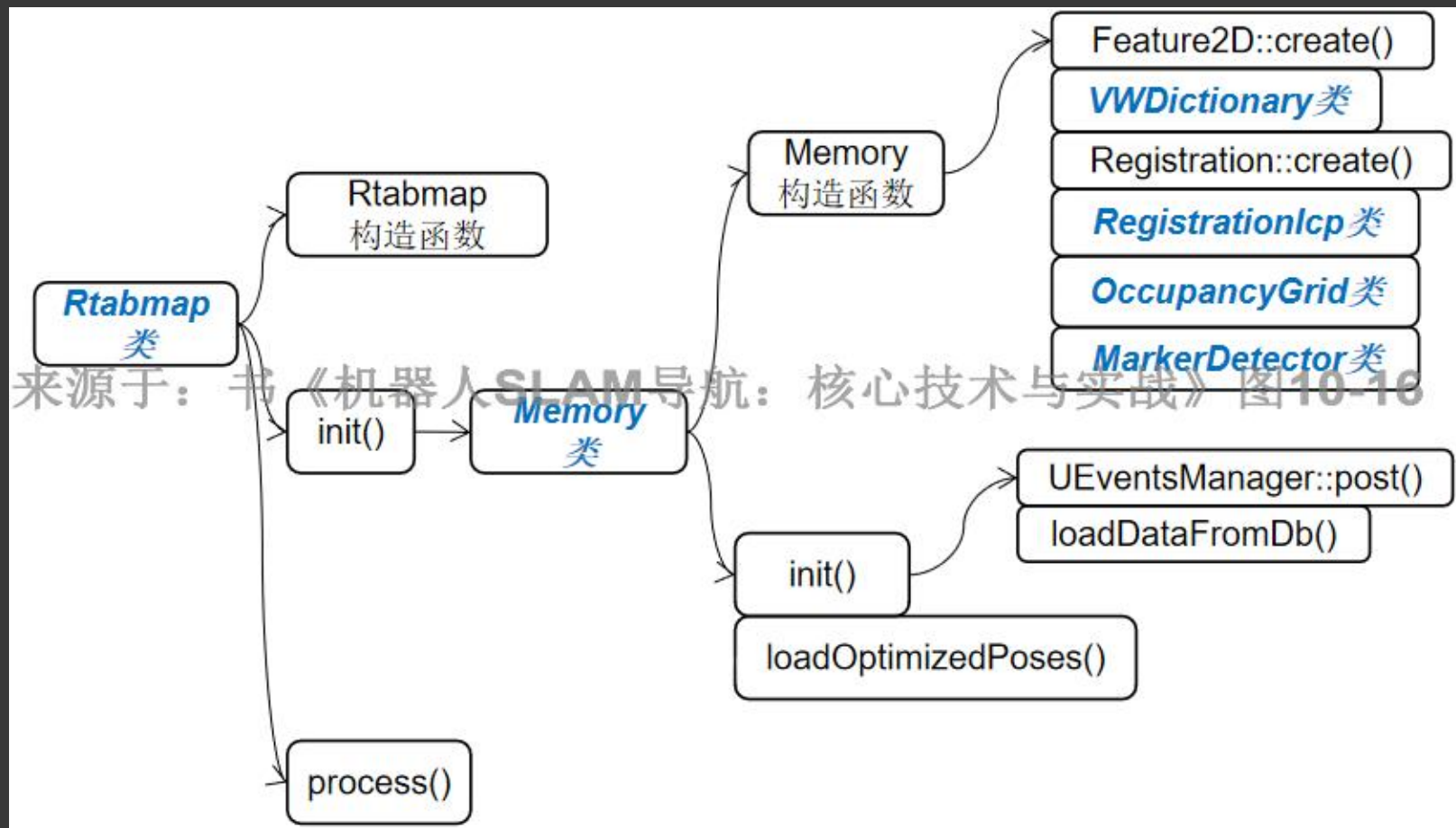
- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行



rtabmap主节点调用流程

10.1 RTABMAP算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行



rtabmap核心库调用流程

10.1 RTABMAP算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行

环境：Ubuntu18.04+ROS melodic

①安装依赖：

```
sudo apt install ros-melodic-rtabmap ros-melodic-rtabmap-ros
sudo apt remove ros-melodic-rtabmap ros-melodic-rtabmap-ros
```

```
#安装g2o
sudo apt install ros-melodic-libg2o
```

```
#取别名来移除Eigen3.2.10, 并还原系统自带的Eigen3.4.4
#如果没有改动系统自带Eigen的话可以忽略这一步
sudo mv -f /usr/include/eigen3 /usr/include/eigen3.bak-3.2.10
sudo mv -f /usr/include/Eigen /usr/include/Eigen.bak-3.2.10
sudo mv -f /usr/include/eigen3.default.bak /usr/include/eigen3
```

```
#路径拷贝
sudo cp -rf /usr/include/eigen3/Eigen /usr/include/
sudo cp -rf /usr/include/eigen3 /usr/local/include/
sudo cp -rf /usr/include/Eigen /usr/local/include/
#下载GTSAM源码
git clone https://github.com/borglab/gtsam.git
#编译安装
cd gtsam
mkdir build
cd build
#设置GTSAM中使用系统的Eigen, 而不是自带的Eigen, 其实就是将
GTSAM_USE_SYSTEM_EIGEN参数设置为ON
cmake -DGTSAM_USE_SYSTEM_EIGEN=ON ..
sudo make install
```

```
#安装依赖Boost
sudo apt-get install libboost-all-dev
#安装依赖eigen3
sudo apt-get install libeigen3-dev
```

```
#安装依赖libnabo
cd ~/
git clone git://github.com/ethz-asl/libnabo.git
cd libnabo
SRC_DIR=$PWD
BUILD_DIR=${SRC_DIR}/build
mkdir -p ${BUILD_DIR} && cd ${BUILD_DIR}
cmake -DCMAKE_BUILD_TYPE=RelWithDebInfo ${SRC_DIR}
make
make test
sudo make install
```

```
#安装libpointmatcher
cd ~/
git clone git://github.com/ethz-asl/libpointmatcher.git
cd libpointmatcher
SRC_DIR=$PWD
BUILD_DIR=${SRC_DIR}/build
mkdir -p ${BUILD_DIR} && cd ${BUILD_DIR}
cmake -D CMAKE_BUILD_TYPE=RelWithDebInfo ${SRC_DIR}
make
sudo make install
```

10.1 RTABMAP 算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行

环境：Ubuntu18.04+ROS melodic

②安装rtabmap核心库：

```
#下载rtabmap核心库源码
cd ~
git clone https://github.com/introlab/rtabmap.git rtabmap
#切换到分支melodic
cd rtabmap
git checkout melodic-devel
#编译安装
cd build
cmake ..
make
sudo make install
```

③安装rtabmap_ros功能包：

```
#将上面编译好的rtabmap核心库安装到ROS工作空间来
cd ~/rtabmap/build
cmake -DCMAKE_INSTALL_PREFIX=~/.catkin_ws/devel ..
make
sudo make install
#下载rtabmap_ros功能包源码
cd ~/.catkin_ws/src
git clone https://github.com/introlab/rtabmap_ros.git
rtabmap_ros
#切换到分支melodic
cd rtabmap_ros
git checkout melodic-devel
#编译安装
cd ~/.catkin_ws
catkin_make -
DCATKIN_WHITELIST_PACKAGES="rtabmap_ros"
```

10.1 RTABMAP 算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行

环境：Ubuntu18.04+ROS melodic

RTABMAP离线运行：

```
#启动建图
roslaunch rtabmap_ros demo_robot_mapping.launch
#播放数据集
rosbag play --clock demo_mapping.bag
```

10.1 RTABMAP算法

- RTABMAP原理分析
- RTABMAP源码解读
- RTABMAP安装与运行

环境：Ubuntu18.04+ROS melodic

RTABMAP在线运行：

结合实际机器人上搭配的传感器，编写launch配置文件，
然后启动传感器以及由该launch配置的rtabmap开始建图。

```
<launch>
<group ns="rtabmap">
  <!-- Use RGBD synchronization -->
  <!-- Here is a general example using a standalone nodelet,
    but it is recommended to attach this nodelet to nodelet manager of the camera to avoid topic serialization -->
  <node pkg="nodelet" type="nodelet" name="rgbd_sync" args="standalone rtabmap_ros/rgbd_sync" output="screen">
    <remap from="rgb/image" to="/camera/rgb/image_rect_color"/>
    <remap from="depth/image" to="/camera/depth_registered/image_raw"/>
    <remap from="rgb/camera_info" to="/camera/rgb/camera_info"/>
    <remap from="rgbd_image" to="rgbd_image"/> <!-- output -->
    <!-- Should be true for not synchronized camera topics (e.g., false for kinectv2, zed, realsense, true for xtion, kinect360)-->
    <param name="approx_sync" value="true"/>
  </node>
  <node name="rtabmap" pkg="rtabmap_ros" type="rtabmap" output="screen" args="--delete_db_on_start">
    <param name="frame_id" type="string" value="base_link"/>
    <param name="subscribe_depth" type="bool" value="false"/>
    <param name="subscribe_rgb" type="bool" value="true"/>
    <param name="subscribe_scan" type="bool" value="true"/>

    <remap from="odom" to="/base_controller/odom"/>
    <remap from="scan" to="/base_scan"/>
    <remap from="rgbd_image" to="rgbd_image"/>

    <param name="queue_size" type="int" value="10"/>

    <!-- RTAB-Map's parameters -->
    <param name="RGBD/NeighborLinkRefining" type="string" value="true"/>
    <param name="RGBD/ProximityBySpace" type="string" value="true"/>
    <param name="RGBD/AngularUpdate" type="string" value="0.01"/>
    <param name="RGBD/LinearUpdate" type="string" value="0.01"/>
    <param name="RGBD/OptimizeFromGraphEnd" type="string" value="false"/>
    <param name="Grid/FromDepth" type="string" value="false"/> <!-- occupancy grid from lidar -->
    <param name="Reg/Force3DoF" type="string" value="true"/>
    <param name="Reg/Strategy" type="string" value="1"/> <!-- 1=ICP -->

    <!-- ICP parameters -->
    <param name="Icp/VoxelSize" type="string" value="0.05"/>
    <param name="Icp/MaxCorrespondenceDistance" type="string" value="0.1"/>
  </node>
</group>
</launch>
```

内容概要

10.1 RTABMAP算法

10.2 VINS算法

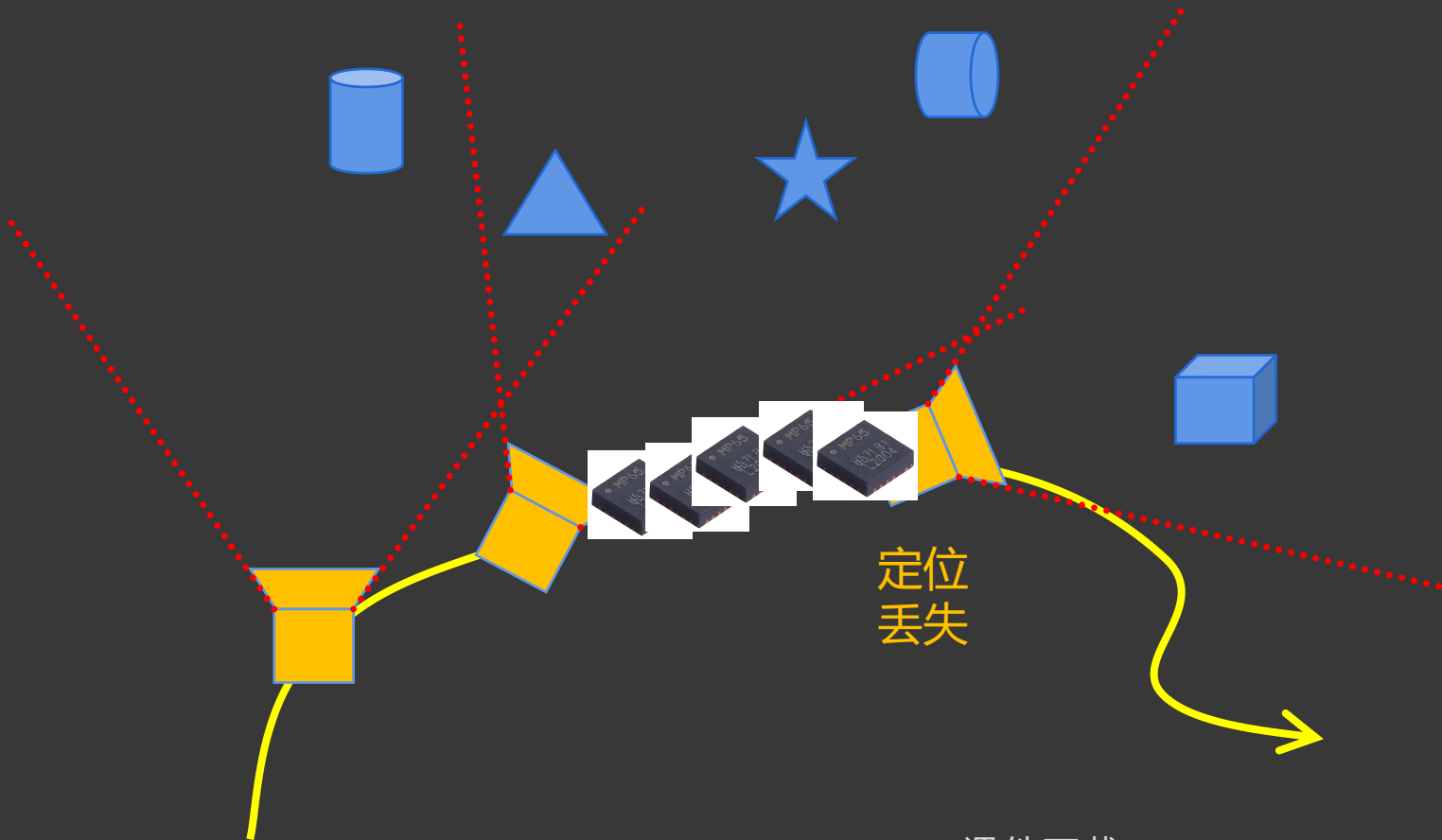
10.3 机器学习与SLAM

10.2 VINS算法

SLAM 为什么需要IMU？

关于VI-SLAM：

- 不管是激光SLAM还是视觉SLAM，由于传感器采样率、传感器测量精度、主机计算力等因素的限制，在高速运动状态下定位跟踪极易丢失。
- 虽然轮式里程计能为激光SLAM系统提供短期运动预测以避免高速运动时丢失的风险，轮式里程计还能在长走廊这样的低特征环境下提供定位辅助。但是轮式里程计也不是万能的，当地面起伏较大或轮子打滑时，轮式里程计将不再可靠；对于在三维空间工作的视觉SLAM来说，提供二维空间定位信息的轮式里程计很难应用其中。而在高速运动（尤其是高速旋转）或特征严重缺失（比如白墙、天空、地面等场景）时视觉SLAM基本上无法工作。
- 采用IMU进行融合，无疑成了解决这些棘手问题的香饽饽。

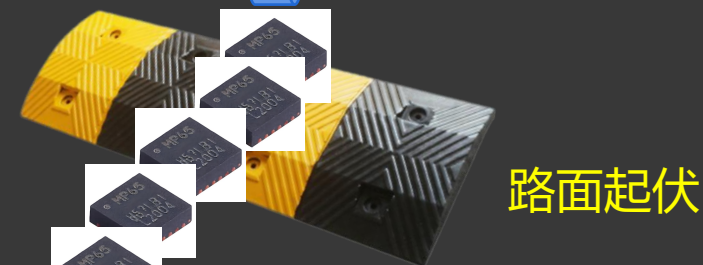
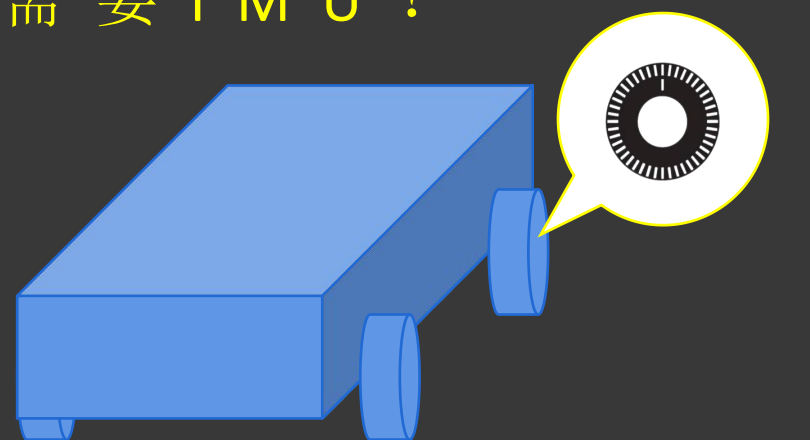


10.2 VINS算法

SLAM 为什么需要IMU？

关于VI-SLAM：

- 不管是激光SLAM还是视觉SLAM，由于传感器采样率、传感器测量精度、主机计算力等因素的限制，在高速运动状态下定位追踪极易丢失。
- 虽然轮式里程计能为激光SLAM系统提供短期运动预测以避免高速运动时丢失的风险，轮式里程计还能在长走廊这样的低特征环境下提供定位辅助。但是轮式里程计也不是万能的，当地面起伏较大或轮子打滑时，轮式里程计将不再可靠；对于在二维空间工作的视觉SLAM来说，提供二维空间定位信息的轮式里程计很难应用其中。而在高速运动（尤其是高速旋转）或特征严重缺失（比如白墙、天空、地面等场景）时视觉SLAM基本上无法工作。
- 采用IMU进行融合，无疑成了解决这些棘手问题的香饽饽。



路面起伏



路面打滑

10.2 VINS算法

SLAM 为什么需要IMU？

- 关于VI-SLAM：
- 不管是激光SLAM还是视觉SLAM，由于传感器采样率、传感器测量精度、主机计算力等因素的限制，在高速运动状态下定位追踪极易丢失。
 - 虽然轮式里程计能为激光SLAM系统提供短期运动预测以避免高速运动时丢失的风险，轮式里程计还能在长走廊这样的低特征环境下提供定位辅助。但是轮式里程计也不是万能的，当地面起伏较大或轮子打滑时，轮式里程计将不再可靠；对于在三维空间工作的视觉SLAM来说，提供二维空间定位信息的轮式里程计很难应用其中。而在高速运动（尤其是高速旋转）或特征严重缺失（比如白墙、天空、地面等场景）时视觉SLAM基本上无法工作。
 - 采用IMU进行融合，无疑成了解决这些棘手问题的香饽饽。

内部融合	$\begin{Bmatrix} roll, pith, yaw \\ acc, gyro, mag \end{Bmatrix} = AHRS(acc, gyro, mag)$	内参标定 数据滤波 姿态融合
外部融合	$\begin{Bmatrix} orientation(roll, pitch, yaw) \\ position(x, y, z) \end{Bmatrix} = fusion(imu + odom)$	外参标定 松/紧耦合 位姿估计
	$\begin{Bmatrix} orientation(roll, pitch, yaw) \\ position(x, y, z) \end{Bmatrix} = fusion(imu + camera)$	
	$\begin{Bmatrix} orientation(roll, pitch, yaw) \\ position(x, y, z) \end{Bmatrix} = fusion(imu + lidar)$	
	$\begin{Bmatrix} orientation(roll, pitch, yaw) \\ position(x, y, z) \end{Bmatrix} = fusion(imu + GPS)$	
	$\begin{Bmatrix} orientation(roll, pitch, yaw) \\ position(x, y, z) \end{Bmatrix} = fusion \begin{pmatrix} imu + \\ odom + \\ camera + \\ \dots \end{pmatrix}$	
其他	机器人上下坡判断	
	激光雷达运动畸变校正	
	机器人实时运动操控	
	...	

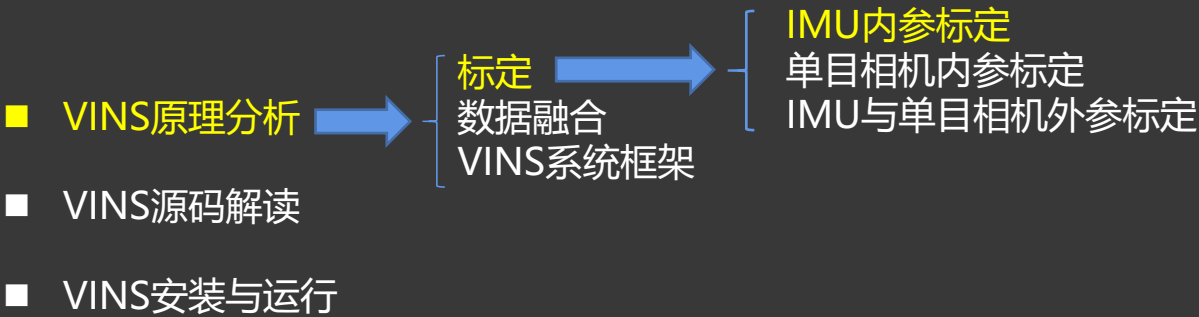
来源于：书《机器人SLAM导航：核心技术与实战》表10-3

10.2 VINS算法

有哪些SLAM系统支持IMU融合？

- Cartographer
- ORB-SLAM3
- RTABMAP
- MSCKF
- OKVIS
- ROVIO
- VINS
- ...

10.2 VINS算法



$A^{cal} = T^{acc} \bullet S^{acc} \bullet (A - B^{acc})$, 写成矩阵的形式:

$$\begin{bmatrix} A_x^{cal} \\ A_y^{cal} \\ A_z^{cal} \end{bmatrix} = \begin{bmatrix} 1 & T_{xy}^{acc} & T_{xz}^{acc} \\ T_{yx}^{acc} & 1 & T_{yz}^{acc} \\ T_{zx}^{acc} & T_{zy}^{acc} & 1 \end{bmatrix} \bullet \begin{bmatrix} S_x^{acc} & 0 & 0 \\ 0 & S_y^{acc} & 0 \\ 0 & 0 & S_z^{acc} \end{bmatrix} \bullet \left(\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} - \begin{bmatrix} B_x^{acc} \\ B_y^{acc} \\ B_z^{acc} \end{bmatrix} \right)$$

$$\sqrt{[\text{var}(A_x)]^2 + [\text{var}(A_y)]^2 + [\text{var}(A_z)]^2} < P$$

$$\min \sum_{n=1}^N \left(\|g\|^2 - \|T^{acc} \bullet S^{acc} \bullet (A - B^{acc})\|^2 \right)^2$$

加速度计标定

$W^{cal} = T^{gyro} \bullet S^{gyro} \bullet (W - B^{gyro})$, 写成矩阵的形式:

$$\begin{bmatrix} W_x^{cal} \\ W_y^{cal} \\ W_z^{cal} \end{bmatrix} = \begin{bmatrix} 1 & T_{xy}^{gyro} & T_{xz}^{gyro} \\ T_{yx}^{gyro} & 1 & T_{yz}^{gyro} \\ T_{zx}^{gyro} & T_{zy}^{gyro} & 1 \end{bmatrix} \bullet \begin{bmatrix} S_x^{gyro} & 0 & 0 \\ 0 & S_y^{gyro} & 0 \\ 0 & 0 & S_z^{gyro} \end{bmatrix} \bullet \left(\begin{bmatrix} W_x \\ W_y \\ W_z \end{bmatrix} - \begin{bmatrix} B_x^{gyro} \\ B_y^{gyro} \\ B_z^{gyro} \end{bmatrix} \right)$$

$u_{g,k} = \text{RK4n}(T^{gyro} \bullet S^{gyro} \bullet (W_i - B^{gyro}), u_{a,k-1})$, 其中 $i = 1, 2, 3, \dots$

$$\min \sum_{n=2}^N \|u_{a,k} - u_{g,k}\|^2$$

陀螺仪标定

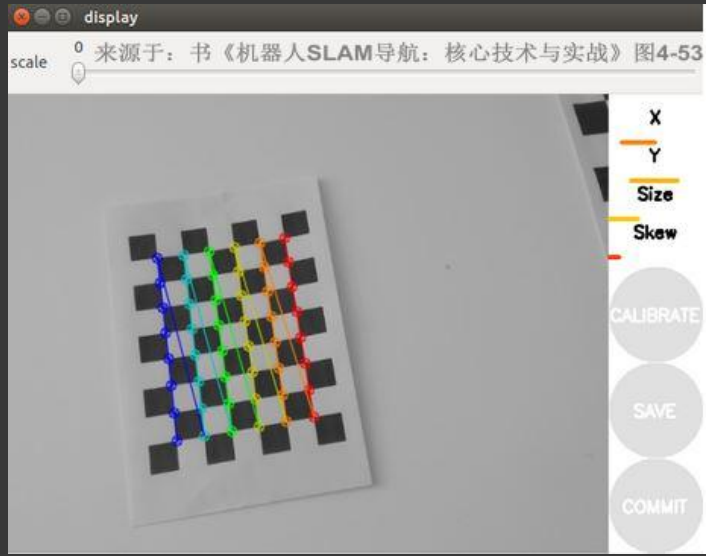
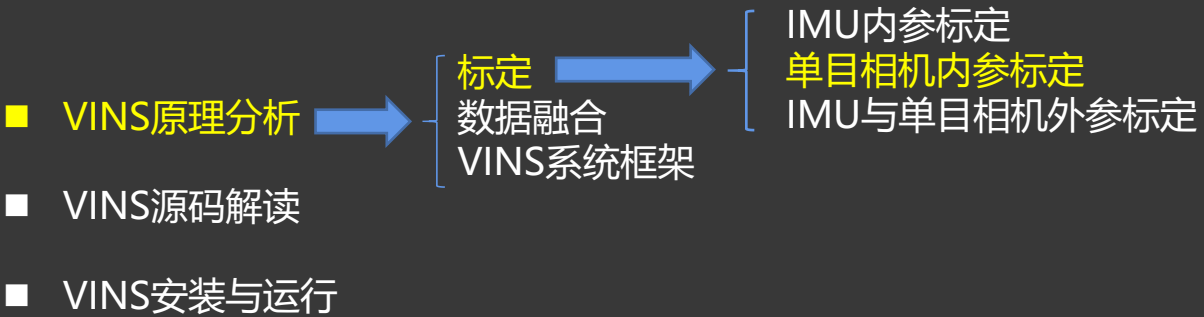
$M^{cal} = T^{m2a} \bullet (M - B^{mag})$, 写成矩阵的形式:

$$\begin{bmatrix} M_x^{cal} \\ M_y^{cal} \\ M_z^{cal} \end{bmatrix} = \begin{bmatrix} T_{11}^{m2a} & T_{12}^{m2a} & T_{13}^{m2a} \\ T_{21}^{m2a} & T_{22}^{m2a} & T_{23}^{m2a} \\ T_{31}^{m2a} & T_{32}^{m2a} & T_{33}^{m2a} \end{bmatrix} \bullet \left(\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} B_x^{mag} \\ B_y^{mag} \\ B_z^{mag} \end{bmatrix} \right)$$

$$\min \sum_{n=1}^N \left(A^{cal} \bullet (T^{m2a} \bullet (M - B^{mag})) - Q \right)^2$$

磁力计标定

10.2 VINS算法



$$\begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{Z} \mathbf{K} \mathbf{P}$$

无畸变内参模型

$$\begin{cases} r^2 = X^2 + Y^2 \\ X_{\text{distort}} = X \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 XY + p_2 \cdot (r^2 + 2X^2) \\ Y_{\text{distort}} = Y \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 XY + p_1 \cdot (r^2 + 2Y^2) \\ U = \frac{f_x}{Z} \cdot X_{\text{distort}} + c_x \\ V = \frac{f_y}{Z} \cdot Y_{\text{distort}} + c_y \end{cases}$$

畸变内参模型

10.2 VINS算法

- VINS原理分析

■ VINS源码解读

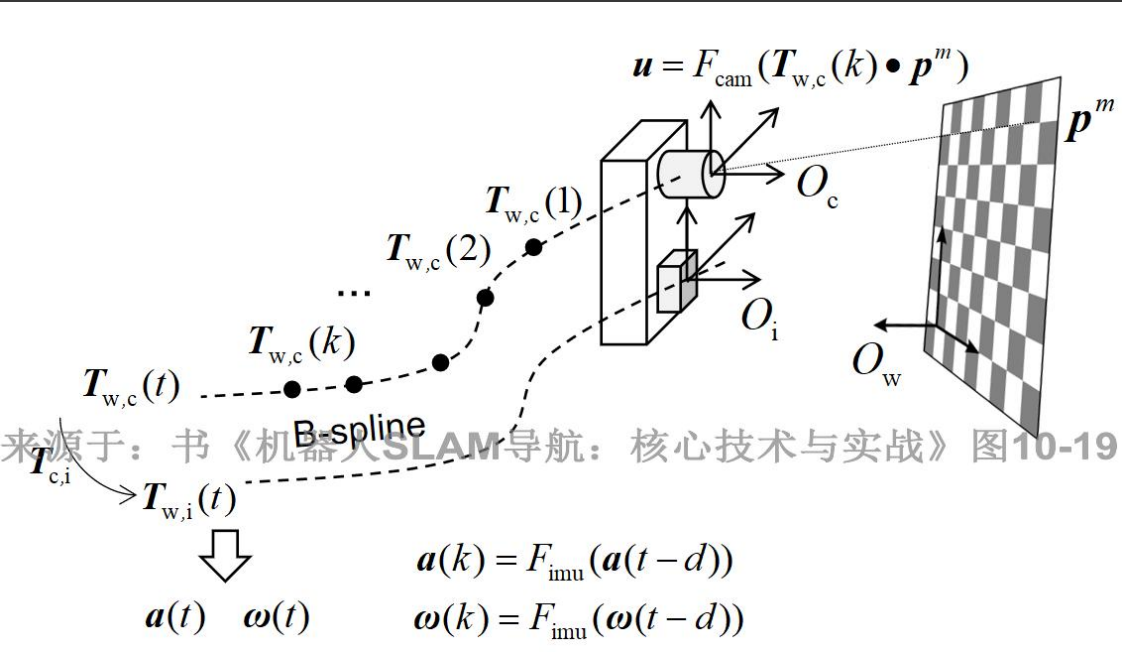
■ VINS安装与运行
- 标定

数据融合

VINS系统框架
- IMU内参标定

单目相机内参标定

IMU与单目相机外参标定



离线外参标定

实测值重投影值

$$u = F_{cam}(T_{w,c}(k) \bullet p^m), \quad e_u(k, m) = u - F_{cam}(T_{w,c}(k) \bullet p^m)$$

$$a(k) = F_{imu}(a(t-d)), \quad e_a(k) = a(k) - F_{imu}(a(t-d))$$

$$\omega(k) = F_{imu}(\omega(t-d)), \quad e_\omega(k) = \omega(k) - F_{imu}(\omega(t-d))$$

$$\arg \min_{F_{cam}, F_{imu}, T_{c,i}, d} \sum (e_u + e_a + e_\omega)$$

离线外参标定 VS 在线外参标定

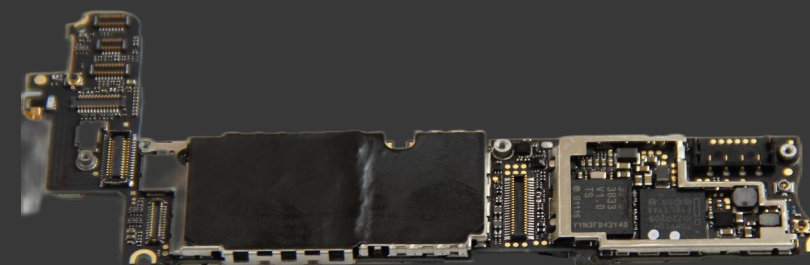
10.2 VINS算法

- VINS原理分析 → 标定
数据融合
VINS系统框架
- VINS源码解读
- VINS安装与运行

松耦合 vs 紧耦合？



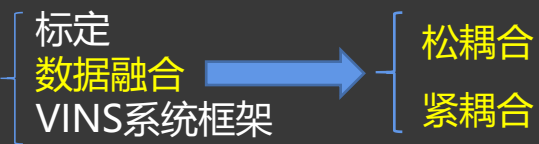
电脑主板（拆卸改装容易）



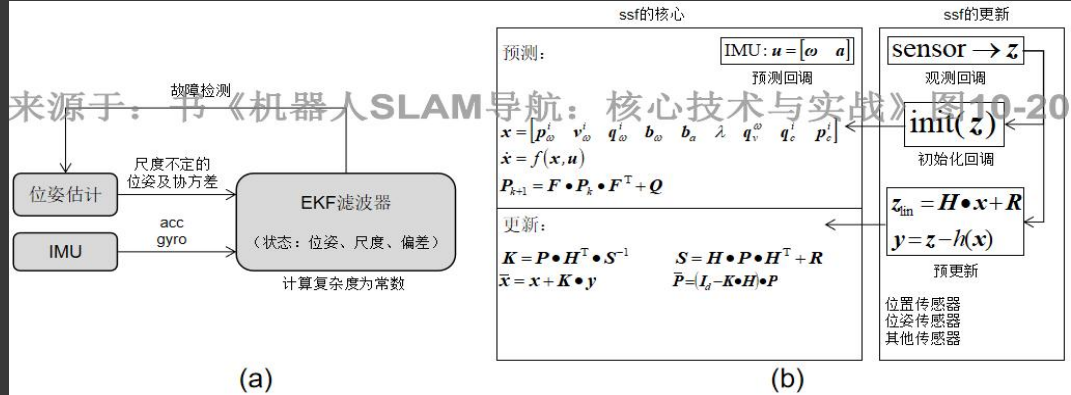
手机主板（拆卸改装困难）

10.2 VINS算法

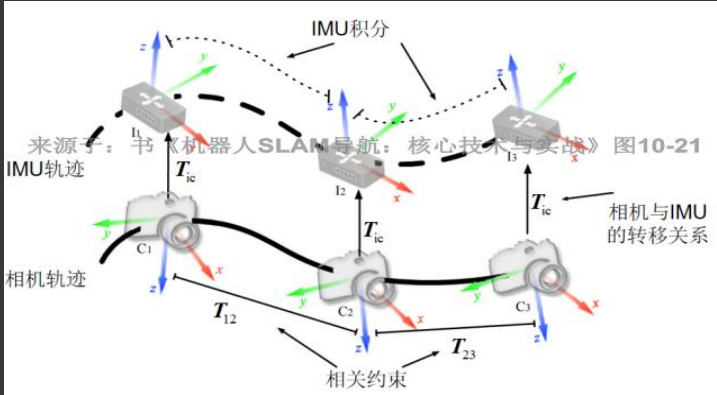
- VINS原理分析
- VINS源码解读
- VINS安装与运行



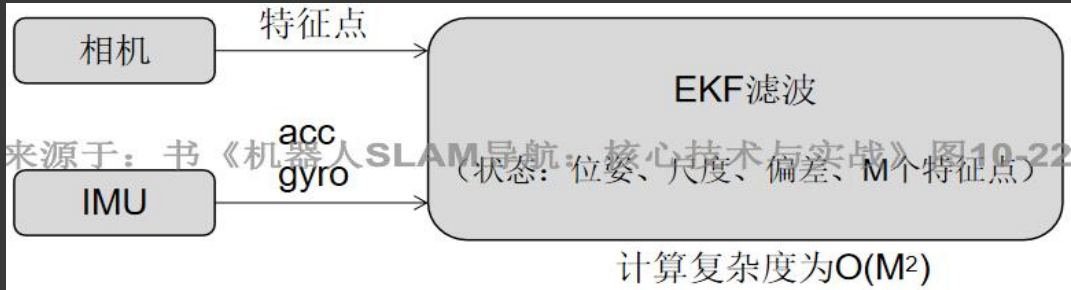
Visual-IMU SLAM	滤波方法	优化方法
松耦合	ssf、msf	
紧耦合	MSCKF、ROVIO	OKVIS、ORB-SLAM3、VINS



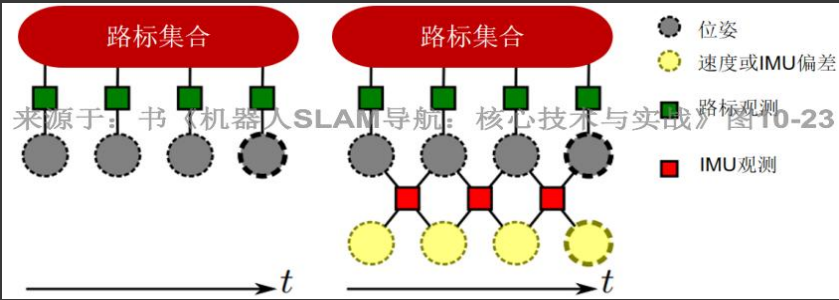
滤波方法的松耦合方



优化方法的松耦合方



滤波方法的紧耦合方式



优化方法的紧耦合方式

10.2 VINS算法

- VINS原理分析

■ VINS源码解读

■ VINS安装与运行
- 标定数据融合

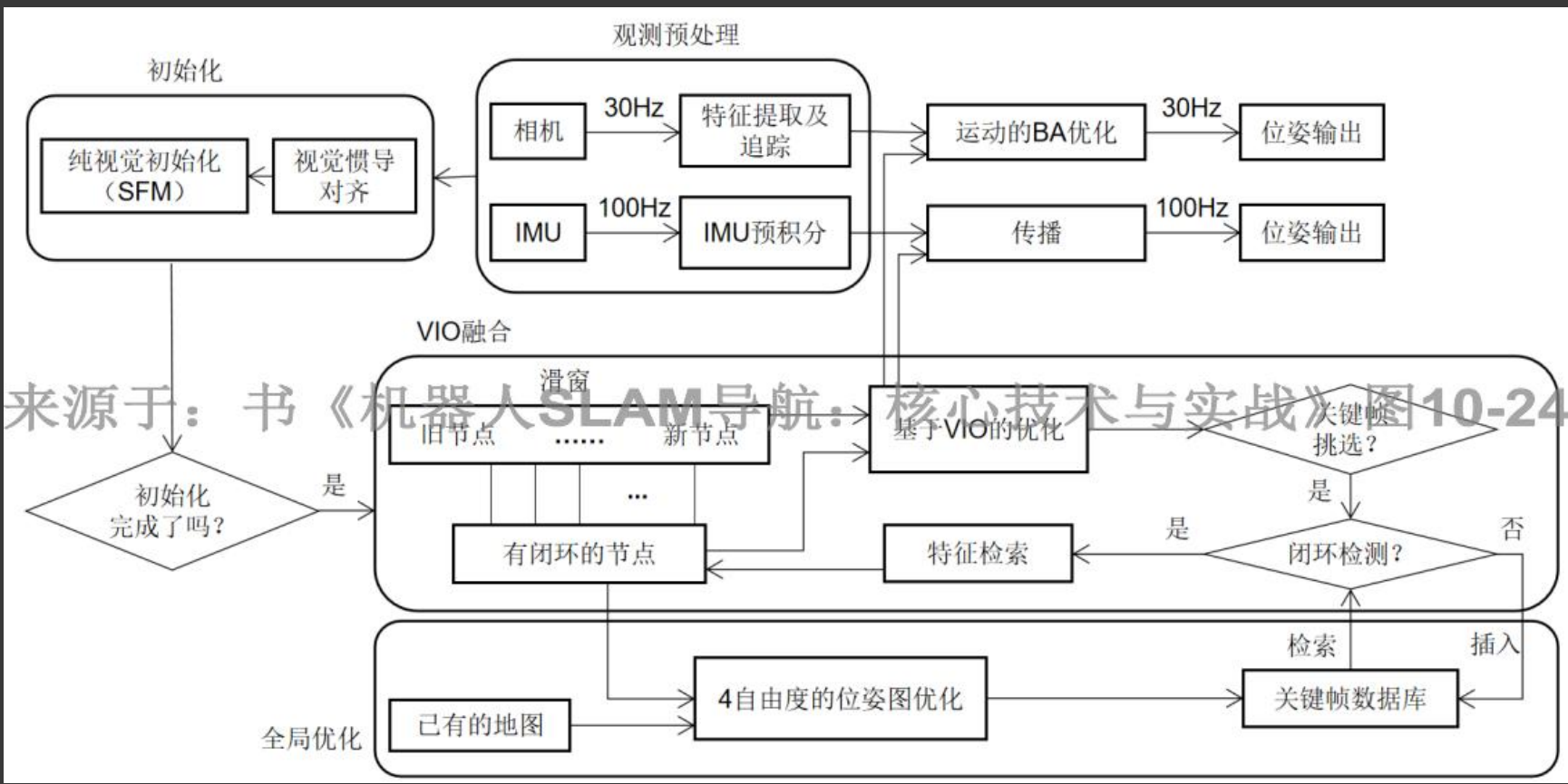
VINS系统框架
- VINS-Mono

VINS-Mobile

VINS-Fusion

- 基于优化的紧耦合方案的融合精度是最高的
- VINS正是基于优化的紧耦合方案的典型代表

- 观测预处理模块
- 初始化模块
- VIO融合模块
- 全局优化模块



10.2 VINS算法

- VINS原理分析
- VINS源码解读
- VINS安装与运行

节点	订阅话题	发布话题
vins_estimator	/vins_estimator/imu_propagate /vins_estimator/path /vins_estimator/relocalization_path /vins_estimator/odometry /vins_estimator/point_cloud /vins_estimator/history_cloud /vins_estimator/key_poses /vins_estimator/camera_pose /vins_estimator/camera_pose_visual /vins_estimator/keyframe_pose /vins_estimator/keyframe_point /vins_estimator/extrinsic /vins_estimator/relo_relative_pose	/vins_estimator/imu_propagate /vins_estimator/path /vins_estimator/relocalization_path /vins_estimator/odometry /vins_estimator/point_cloud /vins_estimator/history_cloud /vins_estimator/key_poses /vins_estimator/camera_pose /vins_estimator/camera_pose_visual /vins_estimator/keyframe_pose /vins_estimator/keyframe_point /vins_estimator/extrinsic /vins_estimator/relo_relative_pose
feature_tracker	/feature_tracker/feature /feature_tracker/restart /pose_graph/match_points	/feature_tracker/feature /feature_tracker/feature_img /feature_tracker/restart
pose_graph	/pose_graph/match_image /pose_graph/camera_pose_visual /pose_graph/key_odometrys /pose_graph/no_loop_path /pose_graph/match_points /pose_graph/pose_graph_path /pose_graph/base_path /pose_graph/pose_graph /pose_graph/path_1 /pose_graph/path_2 /pose_graph/path_3 ...	/pose_graph/match_image /pose_graph/camera_pose_visual /pose_graph/key_odometrys /pose_graph/no_loop_path /pose_graph/match_points /pose_graph/pose_graph_path /pose_graph/base_path /pose_graph/pose_graph /pose_graph/path_1 /pose_graph/path_2 /pose_graph/path_3 ...

来源于：书《机器人SLAM导航：核心技术与实战》表10-5

VINS的代码非常简洁，整个算法由3个独立的ROS节点来实现，各个ROS节点之间通过话题传递数据。

■ feature_tracker

对输入的图像进行特征提取和光流追踪，然后将处理得到的特征点发布出去

■ vins_estimator

利用输入的IMU及图像特征数据计算VIO里程计，然后将VIO里程计发布出去用于定位

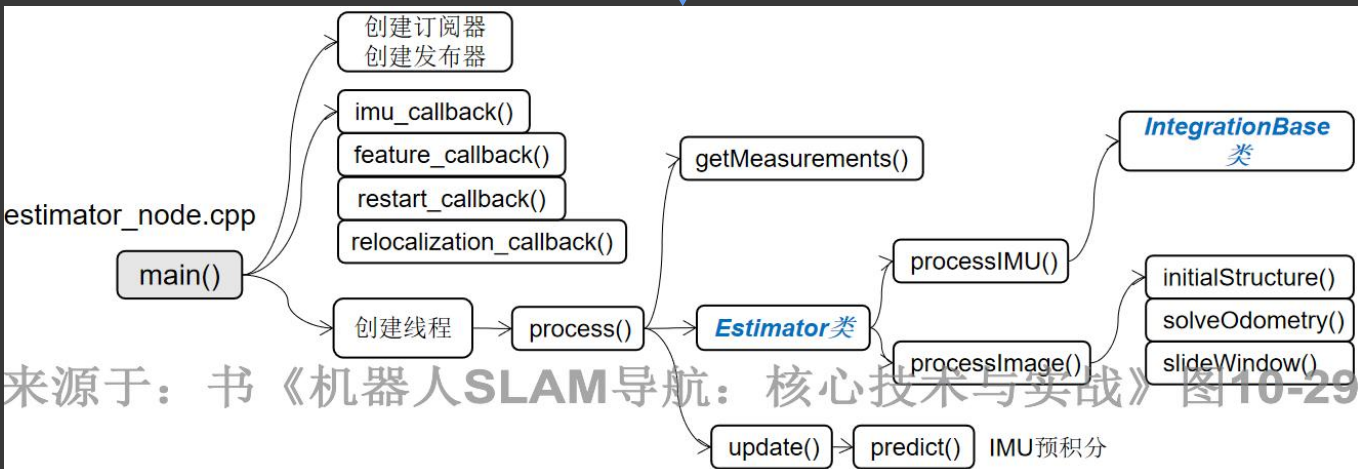
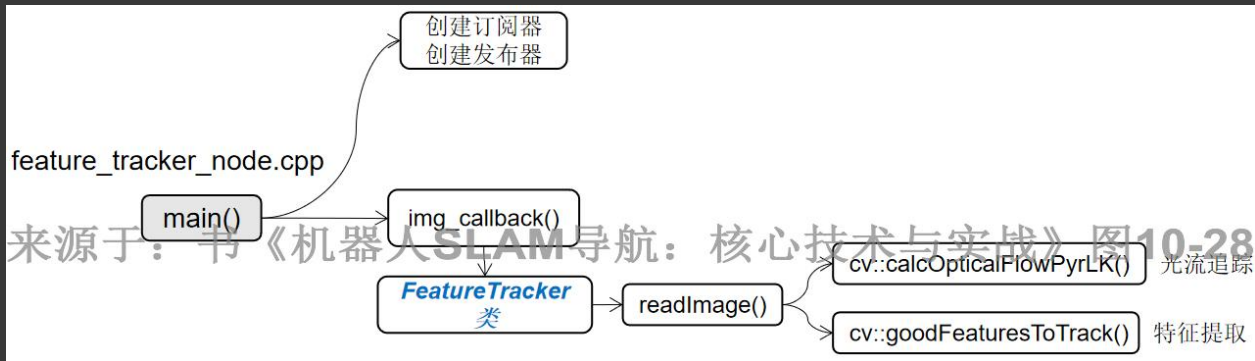
■ pose_graph

通过回调函数获取输入数据，然后在各个回调函数中检测闭环

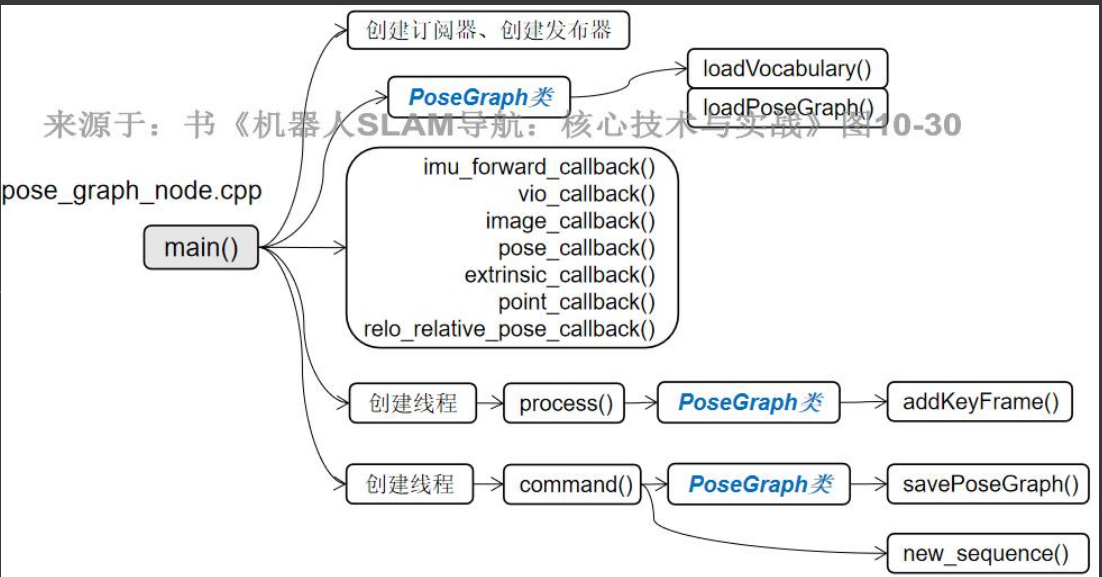
10.2 VINS算法

- VINS原理分析
- VINS源码解读
- VINS安装与运行

feature_tracker节点调用流程



vins_estimator节点调用流程

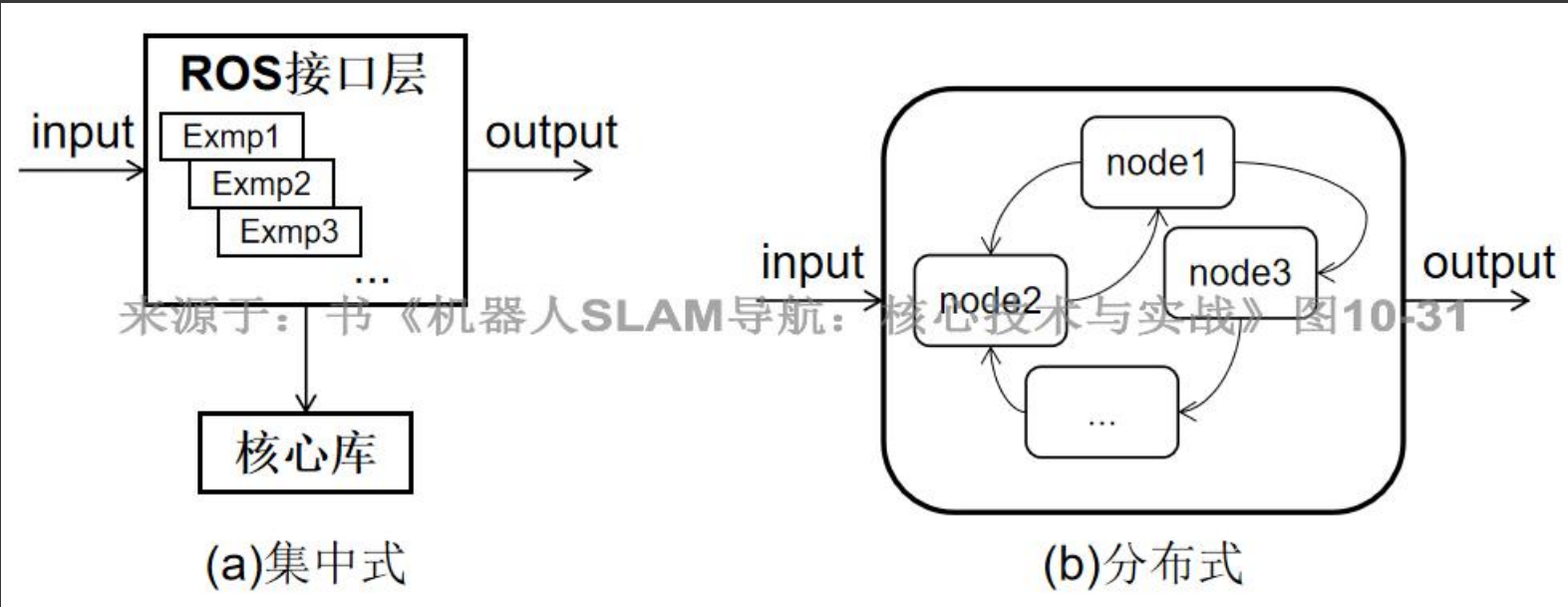


pose_graph节点调用流程

10.2 VINS算法

- VINS原理分析
- VINS源码解读
- VINS安装与运行

- 到这里为止，本书已经介绍了Gmapping、Cartographer、LOAM、ORB-SLAM2、LSD-SLAM、SVO、RTABMAP和VINS这8种SLAM算法的代码框架。可以发现这些SLAM代码框架可以分为两类，一类是集中式，另一种是分布式。
- 集中式，就是SLAM的核心逻辑被封装在一个核心库之中，然后由ROS接口层的各个实例程序调用，比如Gmapping、Cartographer、ORB-SLAM2、LSD-SLAM、RTABMAP。
- 分布式，就是SLAM算法按照功能模块直接被划分成几个独立的ROS节点来实现，节点之间通过话题进行数据交换，比如LOAM、VINS。



拿到一个开源SLAM代码，学习重点当然不是代码中的各种细枝末节的语法，因为代码细枝末节的语法是千变万化无穷无尽的。

学习重点一方面应该是了解代码框架设计思路，这不仅能帮助大家快速上手并且也为后续修改代码和独立设计自己的算法打好了基础；另一方面是通过代码更深入地解读相应论文中的数学公式。

10.2 VINS算法

- VINS原理分析
- VINS源码解读
- VINS安装与运行

环境：Ubuntu18.04+ROS melodic

①安装依赖：

```
#安装ROS系统包
sudo apt-get install ros-melodic-cv-bridge ros-melodic-tf ros-melodic-
message-filters ros-melodic-image-transport

#安装Ceres-solver的依赖项
sudo apt-get install cmake
sudo apt-get install libgoogle-glog-dev libgflags-dev
sudo apt-get install libatlas-base-dev
sudo apt-get install libeigen3-dev
sudo apt-get install libsuitesparse-dev
#下载Ceres-solver源码
git clone https://github.com/ceres-solver/ceres-solver.git
#编译安装Ceres-solver
cd ceres-solver
mkdir build && cd build
cmake ..
make
make test
sudo make install
```

②编译源码：

```
#下载VINS-Mono源码
cd ~/catkin_ws/src
git clone https://github.com/HKUST-Aerial-Robotics/VINS-Mono.git
#编译VINS-Mono
cd ..
catkin_make
```

10.2 VINS算法

- VINS原理分析
- VINS源码解读
- VINS安装与运行

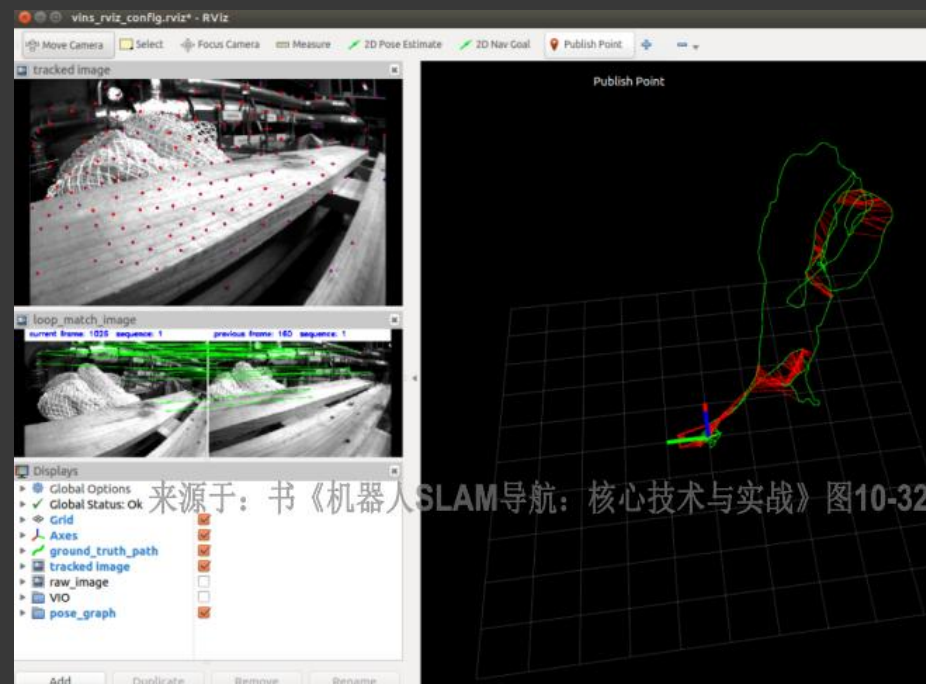
VINS离线运行：

```
#启动VINS
roslaunch vins_estimator euroc.launch
#启动rviz
roslaunch vins_estimator vins_rviz.launch
#播放数据集
rosbag play MH_01_easy.bag
```

VINS在线运行：

在离线运行中，图像和IMU数据由*.bag文件提供。在实际的机器人上运行时，图像和IMU数据由实际的传感器发布。并且还要在配置文件中，对必要的参数进行设置。

环境：Ubuntu18.04+ROS melodic



- 发布传感器数据
- 标定
- 启动VINS

内容概要

10.1 RTABMAP算法

10.2 VINS算法

10.3 机器学习与SLAM

10.3 机器学习与SLAM

传统方法 vs 机器学习方法

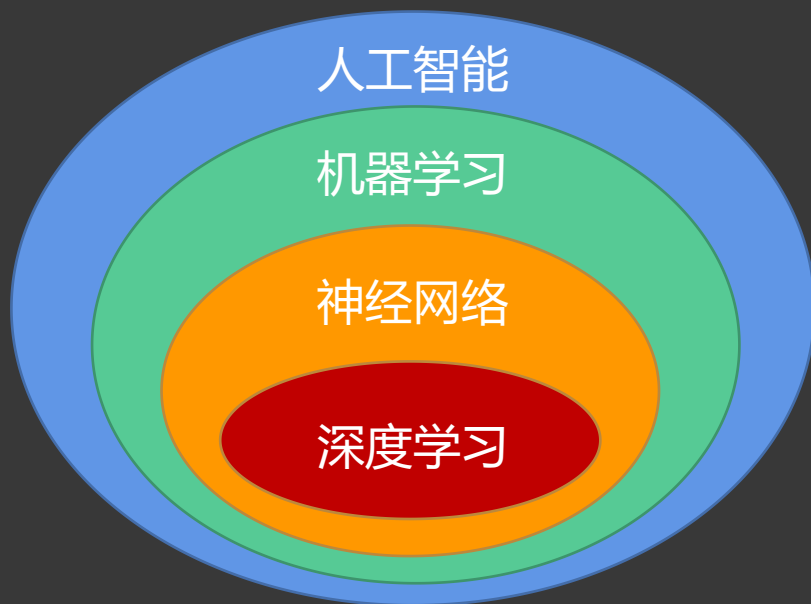
■ 基于传统方法的SLAM：

- 比如Gmapping、Cartographer、LOAM、ORB-SLAM2、LSD-SLAM、SVO、RTABMAP、VINS等。
- 因为这些算法都是在人为精心设计的特定规则下工作，一个算法取得的好效果往往归因于大量小技巧的运用，这与人们心目中的智能相去甚远。比如三段式（前端里程计、后端优化和闭环检测）已经成了SLAM算法的普遍范式，这种特定规则是经过长期发展总结出来的宝贵经验，三段式设计能保证算法运行的实时性，而闭环检测成了解决累积误差的首选方法。再比如从图像中提取特征点（SIFT、SURF、ORB等）然后利用多视图几何和三角化重建求解运动位姿及地图路标，这个思路在工程应用中被证明很有效。

■ 基于机器学习方法的SLAM：

- 一种思路是利用机器学习方法对SLAM传统方法中的局部模块进行改进：比如CNN-SLAM。
- 一种思路是完全抛弃SLAM传统方法全部改用端到端机器学习方法实现：比如DeepVO。
- 机器人所处在一个复杂多变的环境之中，传统方法中设计的各种精妙的规则都仅仅适用于某些特定的环境，如果按照传统方法的思路需要设计一个无限复杂的规则才能适用于所有可能的环境，这显然是不现实的。这就需要一种更通用的方式，让机器人在感知环境的交互过程中，通过自身学习不断提高适应环境的能力。

10.3 机器学习与SLAM



人工智能应用案例：

- 人脸识别
- 语音助手
- 无人驾驶
- 智能推荐
- ...

- 所谓机器学习，就是让机器通过学习获取知识，并利用知识解决特定的问题。
- 这里的机器指计算机，而学习指运行在计算机上的程序算法。

10.3 机器学习与SLAM

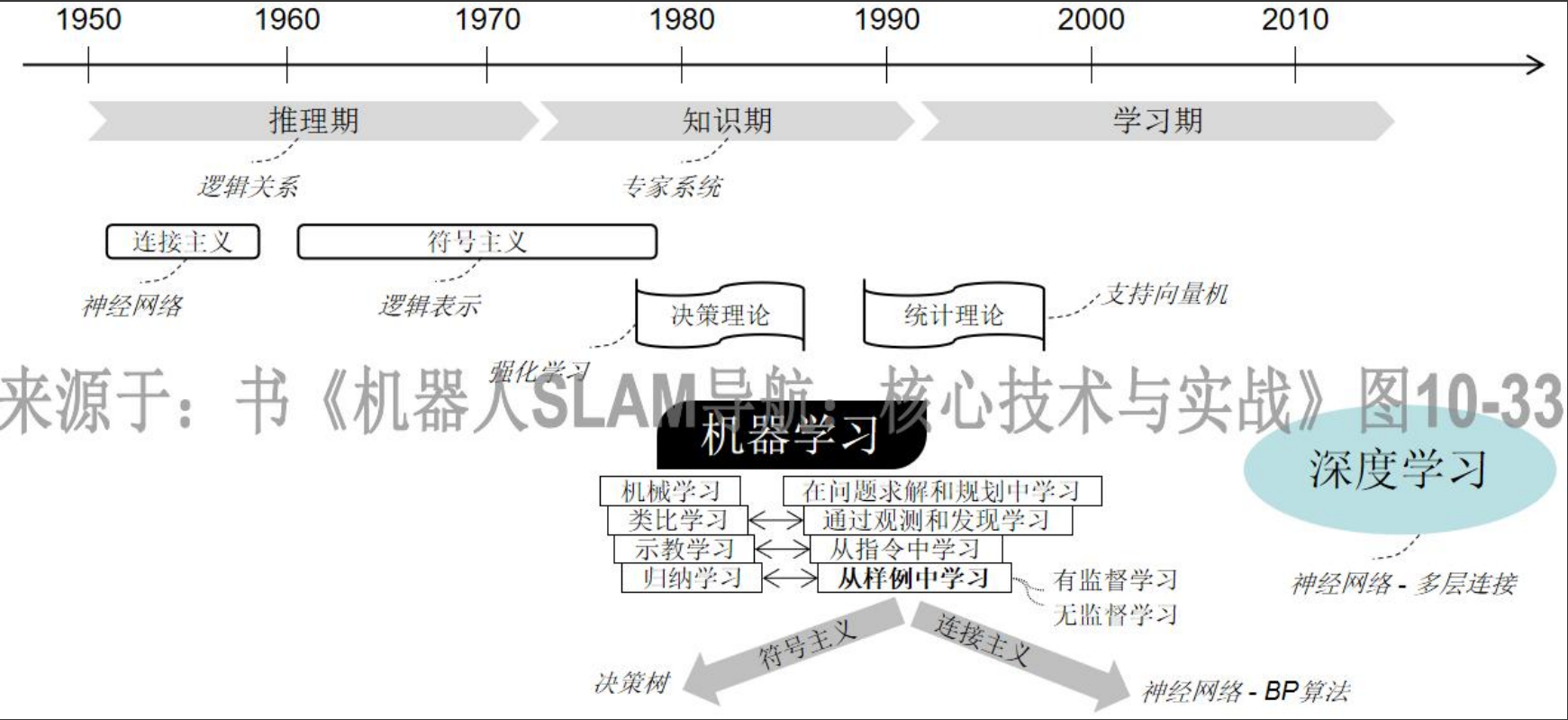
- 机器学习

■ CNN-SLAM算法

■ DeepVO算法
- 发展历程

神经网络

深度学习



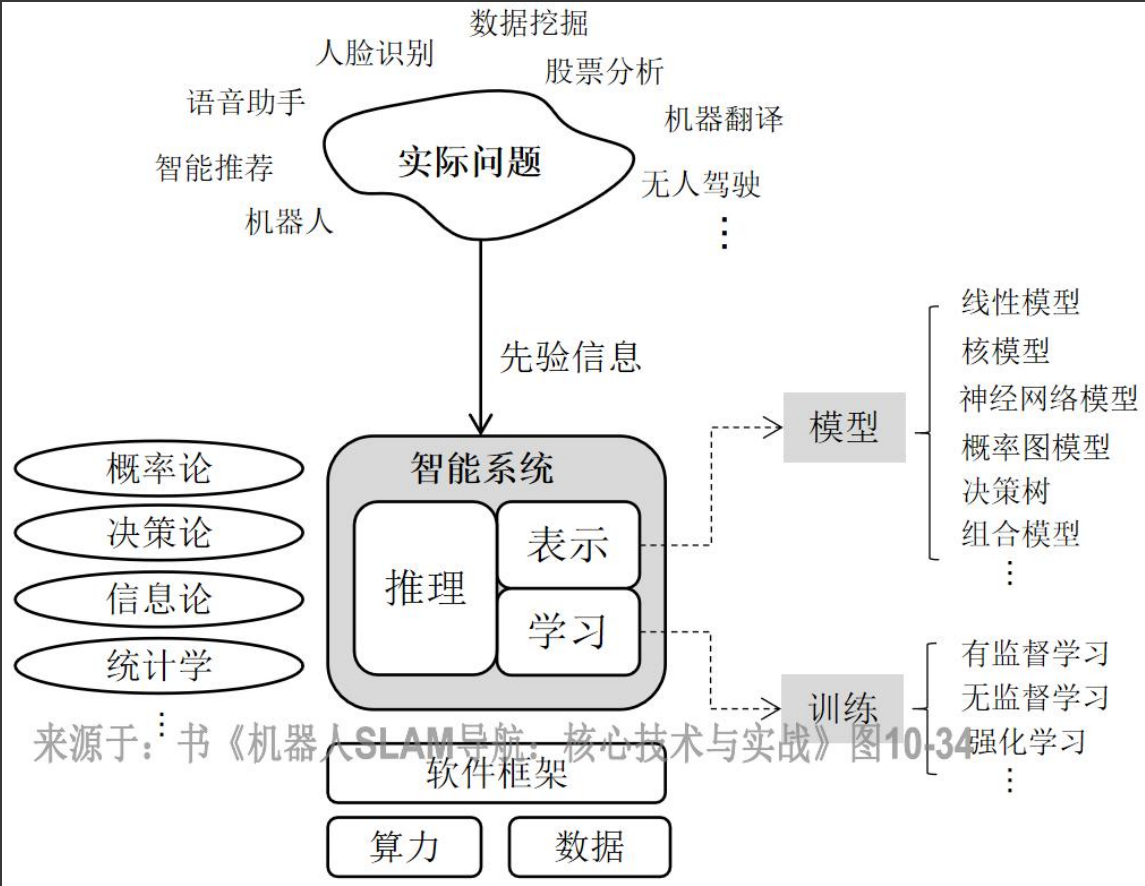
来源于：书《机器人SLAM导航：核心技术与实战》图10-33

人工智能发展历程

10.3 机器学习与SLAM

- 机器学习
- CNN-SLAM算法
- DeepVO算法
- 发展历程
神经网络
深度学习

- 因为关于智能，目前并没有科学的定义，也没有统一的研究方法。推理、知识、神经网络、机器学习、决策理论等诸多的概念，本身就是在不同的层面不同的时期来讨论的，这些概念之间有些是互相重叠的、有些是发展递进的、有些是特例、有些是从不同角度出发的同一观点等等。
- 既然从理论上暂时无法得知智能的真正面貌，那么广大读者也就不必拘泥于理论层面的各种概念，仅从实践的角度讨论和学习主流的一些技术应用就已经够了。



10.3 机器学习与SLAM

- 机器学习

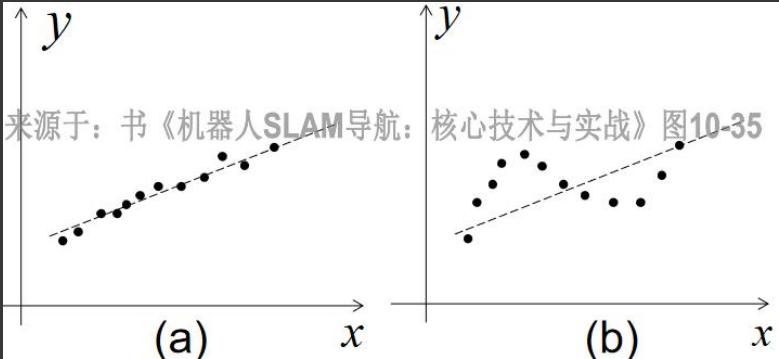
■ CNN-SLAM算法

■ DeepVO算法
- 发展历程

神经网络

深度学习

- 从统计学习的角度看，学习其实就是不断调整模型参数以实现给定数据的拟合过程。
- 整个学习过程涉及四个要素：数据、模型、学习策略和学习算法。

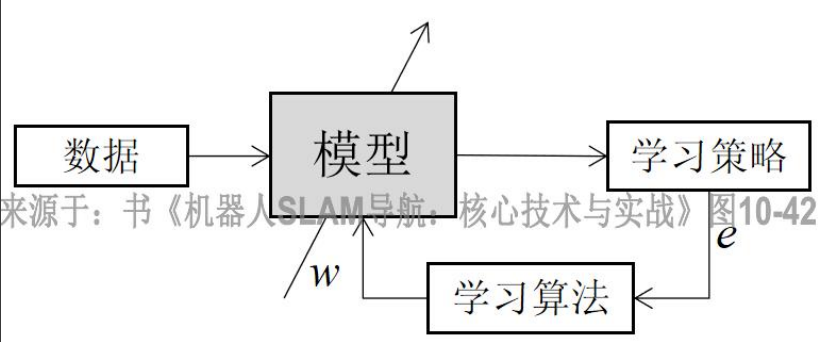


数据的线性关系拟合： $y = ax + b$

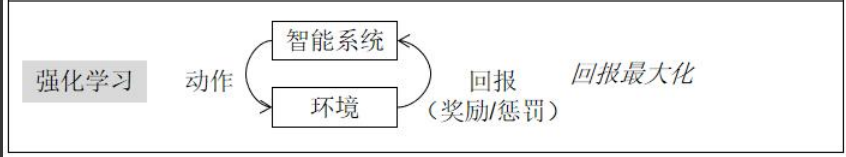
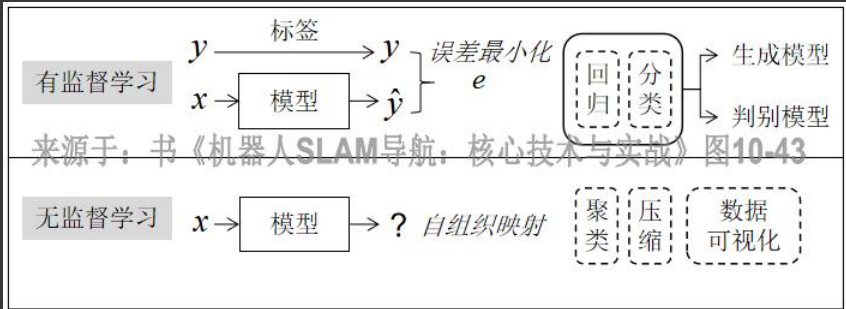
数据的非线性关系拟合：

$$y = w^T \cdot \phi(x)$$
$$= [w_1 \quad w_2 \quad \dots \quad w_n] \cdot \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_n(x) \end{bmatrix}$$
$$= w_1 \phi_1(x) + w_2 \phi_2(x) + \dots + w_n \phi_n(x)$$

- 高维数据的非线性关系拟合：
- 核模型
 - 神经网络模型
 - 概率图模型
 - 决策树
 - ...



基于训练样本的模型学习过程

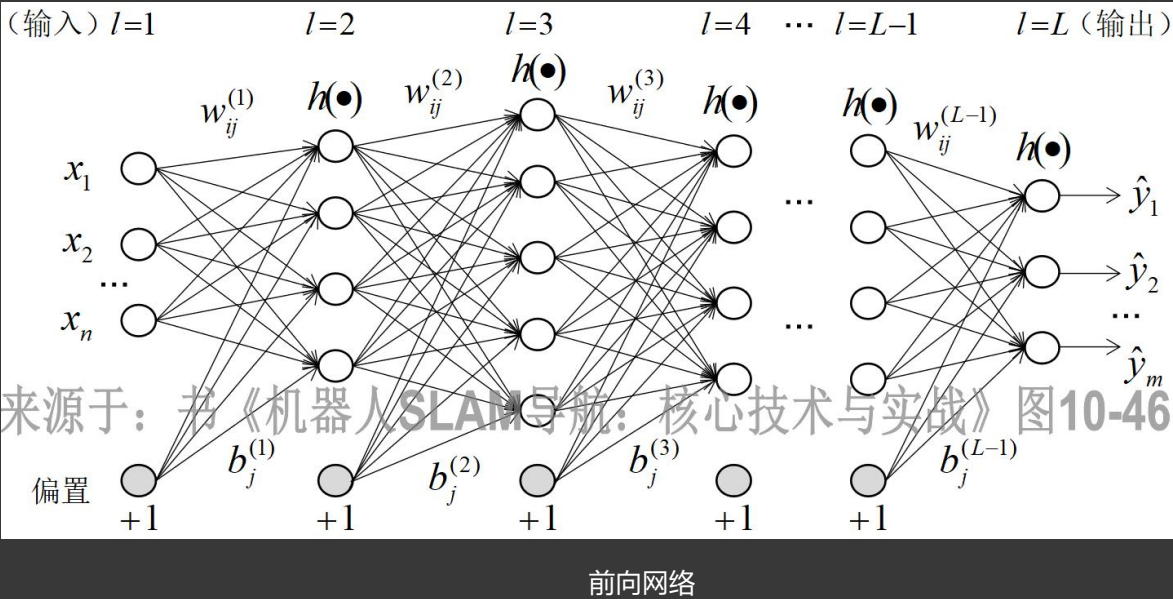
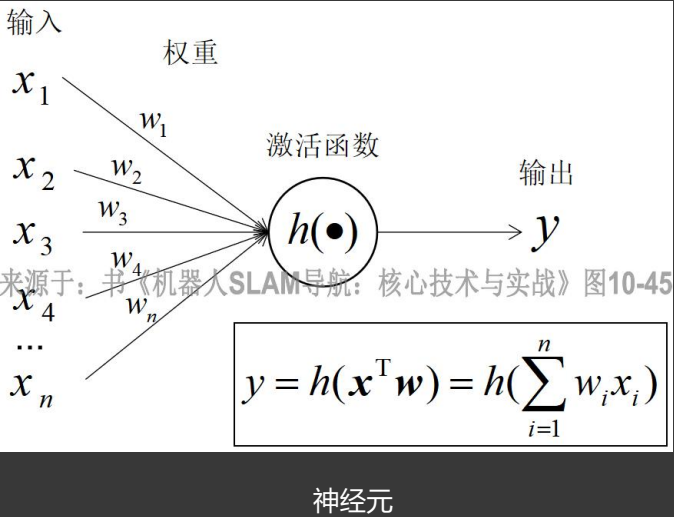


常用学习策略：有监督、无监督和强化学习

10.3 机器学习与SLAM

- 机器学习
- CNN-SLAM算法
- DeepVO算法
- 发展历程
神经网络
深度学习

- 相较于其他机器学习模型，神经网络似乎更加通用，因为它仅通过非线性复合函数方式就能构造出具有足够表达能力的基函数，并通过简单增加神经元数量的方式就能轻松扩展模型的规模。
- 不过由于模型参数w以非线性的方式包含于复合函数之中，这给学习这些参数的过程增加了很大的难度，因此神经网络在很长一段时间里并不被人重视。
- 得益于当今大数据时代数据的极大丰富和硬件计算能力的极大提升，以及神经网络训练方法的改进，使得神经网络几乎成了机器学习领域中最好用的模型。
- 而基于神经网络模型的学习发展起来的深度学习，也成了当今最受关注的研究方向。



- 由于神经元之间可以有很多种连接方式，因此神经网络有很多种类：
- 前向网络
 - 卷积网络
 - 循环网络
 - ...
- 一般采用误差反向传播算法来训练神经网络的参数

10.3 机器学习与SLAM

- 机器学习

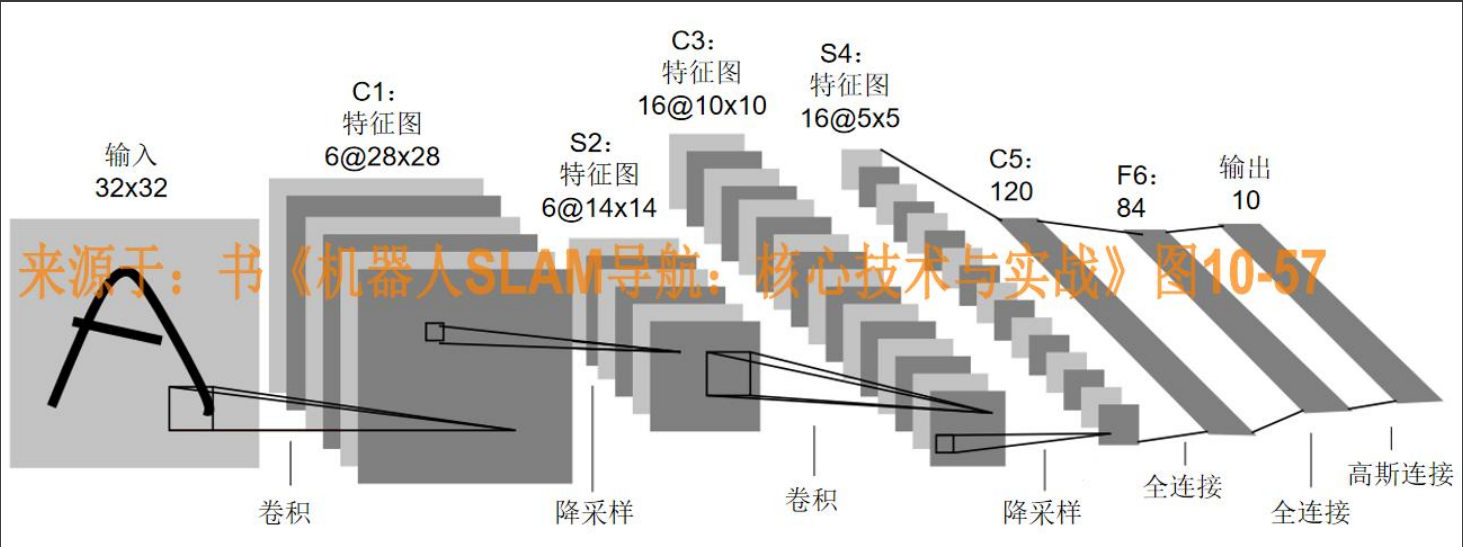
■ CNN-SLAM算法

■ DeepVO算法
- 发展历程

神经网络

深度学习

- 这里**深度**的含义是指模型的规模很大，也就是各种神经网络结构大规模堆叠出来的深度神经网络，当然广义上也包含那些非神经网络模型堆叠出来的深度模型；而**学习**是指这些深度模型配套的学习策略。
- 讨论深度学习技术，需要从深度模型和学习策略这两个方面切入。
- 深度前向网络、深度卷积网络、深度循环网络。
- 深度自编码网络、深度置信网络。
- 深度稀疏网络、深度二值网络、深度脉冲网络、深度支持向量机、深度森林。
- 生成对抗网络、深度强化学习。

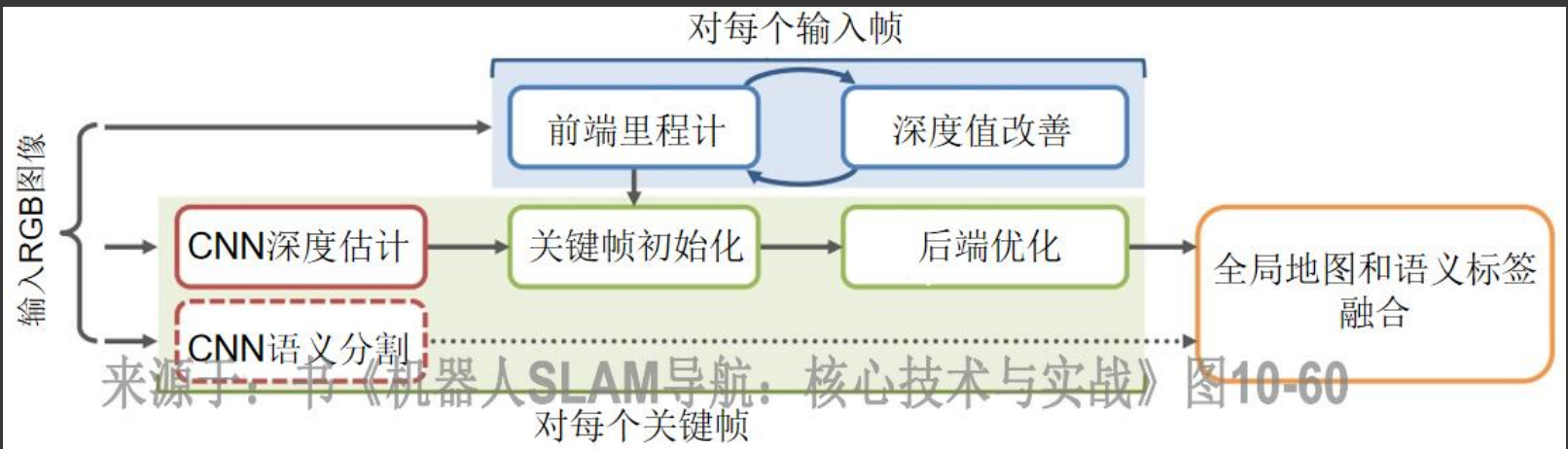


一个经典的深度卷积网络：LeNet-5

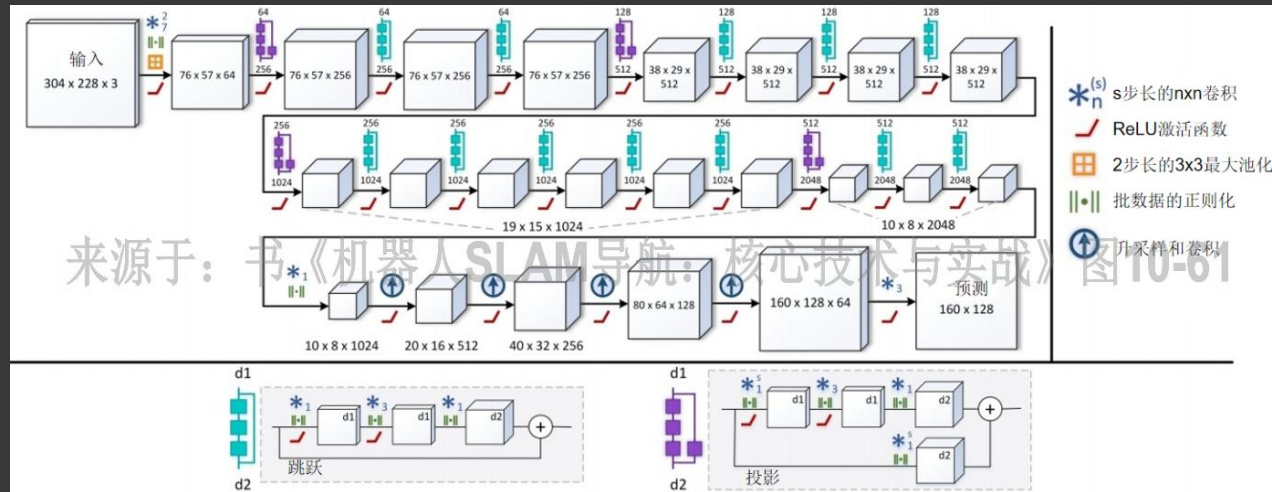
10.3 机器学习与SLAM

- 机器学习
- CNN-SLAM算法
- DeepVO算法

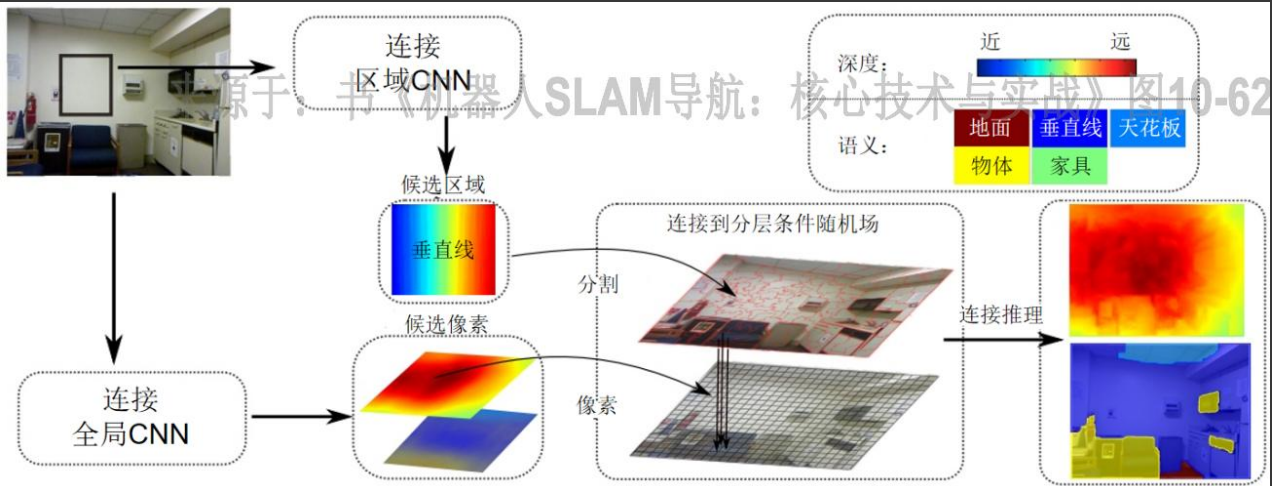
CNN-SLAM依然保留了传统SLAM的前端-后端设计范式，其创新是引入了CNN对输入图像直接进行深度估计和语义分割。



CNN-SLAM系统框架



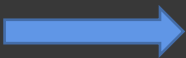
CNN深度估计模块



CNN语义分割模块

10.3 机器学习与SLAM

- 机器学习
- CNN-SLAM算法
- DeepVO算法



完全抛弃SLAM传统方法，即端到端方式。
感知数据直接通入机器学习算法后直接输出结果。



- 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