北京化工大学

Beijing University of Chemical Technology

Computing Methods

自动化科学导论课程报告

李昊

目录

创建日期：2020 年 3 月 25 日
更新日期：2020 年 5 月 24 日

摘要

AGV 系统是集光、机、电、计算机于一体的高新技术。为了能够适应不同的使用要求和缩短新产品的开发周期，AGV 的设计研究趋向于模块化，将 AGV 的各功能模块设计成不同的系列，再根据具体的使用要求进行组合。本文从 AGV 系统的软件部分出发，主要介绍 AGV 系统的两大模块的工作原理：AGV 系统的导航（包括现有的定位与建图算法、栅格地图、现有的全局路径规划算法），AGV 系统的转向控制系统和姿态控制算法。

第 1 章　简介

1.1　AGV 的导航（Navigation）

导航技术是 AGV 技术的研究核心，没有导航功能的 AGV 系统是很难想象的。举个例子，如果 AGV 系统想要在工厂等工作场所运输货物，它必须要在不触碰到任何障碍物的情况下运动到不同的地点。随着信息化技术的提升，CPU 算力的提升和传感器的升级，AGV 的柔性路径导航吸引了越来越多的研究者进行研究，产业化应用越来越广。柔性路径导航技术应用较多的是视觉与激光导航、惯性 (IMU) 导航 [?]、基于 WIFI/UWB 等无线信号定位 [?] 的导航技术等。机器人实现其自主导航的前提是机器人自身位姿的确定，出于对定位精度的要求以及现实环境的复杂化，目前大多数导航系统都是由上述两种或者多种导航方式的结合组合导航技术 [?, ?]。

1.2　AGV 的姿态控制（Steering control）

第 2 章　同步定位与地图构建 (SLAM)

2.1　SLAM 定义

同步定位与地图构建（SLAM 或 Simultaneous localization and mapping）是一种概念：希望机器人从未知环境的未知地点出发，在运动过程中通过重复观测到的地图特征（比如，墙角，柱子等）定位自身位置和姿态，再根据自身位置增量式的构建地图，从而达到同时定位和地图构建的目的。

在误差和噪音条件下，定位和地图构建技术上的复杂度不支持两者同时获得连续的解。即时定位与地图构建（SLAM）是这样一个概念：把两方面的进程都捆绑在一个循环之中，以此支持双方在各自进程中都求得连续解；不同进程中相互迭代的反馈对双方的连续解有改进作用。

同步定位与地图构建（SLAM）被定义为以下问题：在建立新地图模型或者改进已知地图的同时，在该地图模型上定位机器人。实际上，这两个核心问题如果分开解决，将毫无意义；必须同时求解。即通过控制数据 $u_{1:t}$ 和观测数据 $z_{1:t}$ 来求解位姿和地图的联合概率分布

$$p\left(x_{1:t}, m | z_{1:t}, u_{1:t-1}\right) \tag{2.1}$$

2.2 经典 V-SLAM 方法

2.2.1 稀疏法:ORB-SLAM2

该算法 [?] 是基于特征点的实时单目 SLAM 系统，在大规模的、小规模的、室内室外的环境都可以运行。它主要有以下贡献：

1) 这是首个基于单目，双目和 RGB-D 相机的开源 SLAM 方案，这个方案包括，回环检测，地图重用和重定位。
2) 结果说明，BA 优化比 ICP 或者光度和深度误差最小方法的更加精确。
3) 通过匹配远处和近处的双目匹配的点和单目观测，双目的结果比直接使用双目系统更加精确。
4) 针对无法建图的情况，提出了一个轻量级的定位模式，能够更加有效的重用地图。

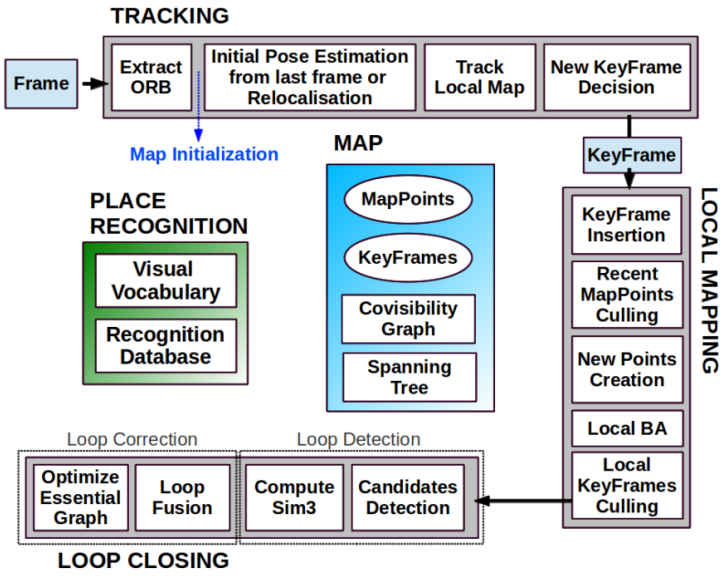


图 2.1: ORB-SLAM 主要分为三个线程进行，也就是如图所示的，分别是 Tracking、LocalMapping 和 LoopClosing

1. 跟踪（Tracking）
   这一部分主要工作是从图像中提取 ORB 特征，根据上一帧进行姿态估计，或者进行通过全局重定位初始化位姿，然后跟踪已经重建的局部地图，优化位姿，再根据一些规则确定新的关键帧。
2. 建图（LocalMapping）
   这一部分主要完成局部地图构建。包括对关键帧的插入，验证最近生成的地图点并进行筛选，然后生成新的地图点，使用局部捆集调整（Local BA），最后再对插入的关键帧进行筛选，去除多余的关键帧。

$$\{\mathbf{R},\mathbf{t}\} = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i\in\mathcal{X}} \rho\left(\left\|\mathbf{x}^i_{(\cdot)} - \pi_{(\cdot)}\left(\mathbf{R}\mathbf{X}^i + \mathbf{t}\right)\right\|^2_{\Sigma}\right) \tag{2.2}$$

3. 闭环检测（LoopClosing）
   这一部分主要分为两个过程，分别是闭环探测和闭环校正。闭环检测先使用 WOB 进行探测，然后通过 Sim3 算法计算相似变换。闭环校正，主要是闭环融合和 Essential Graph 的图优化。

开源代码链接： `https://github.com/raulmur/ORB_SLAM2`

2.2.2 多传感器融合:VINS-Mono

该算法 [**?**] 是一种鲁棒且通用的单目视觉惯性状态估计器。采用一种基于紧耦合、非线性优化的方法，通过融合预积分后的 IMU 测量值和特征观测值，获得高精度的视觉惯性里程计。结合紧耦合方法，回环检测模块能够以最小的计算代价实现重定位。处理视觉和惯性测量的最简单的方法是松耦合的传感器融合 [**?**, **?**]，其中 IMU 被视为一个独立的模块，用于辅助运动的视觉结构 (sfm) 获得的纯视觉位姿估计。融合通常由扩展卡尔曼滤波 (EKF) 完成，其中 IMU 用于状态传播，而视觉位姿用于更新。

IMU 预积分（pre-integration）

在实践中，IMU 通常以比摄像机更高的速率获取数据。不同的方法被提出来处理高速率的 IMU 测量值。最简单的方法是在基于 EKF 的方法中使用 IMU 进行状态传播 [11][13]。在图优化公式中，为了避免重复的 IMU 重复积分，提出了一种有效的方法，即 IMU 预积分 (IMU pre-integration)。通过一系列计算，得到下面的预积分估计值：

$$
\begin{aligned}
\hat{\boldsymbol{\alpha}}_{i+1}^{b_k} &= \hat{\boldsymbol{\alpha}}_{i}^{b_k} + \hat{\boldsymbol{\beta}}_{i}^{b_k}\delta t + \frac{1}{2}\mathbf{R}(\hat{\boldsymbol{\gamma}}_{i}^{b_k})(\hat{\mathbf{a}}_i - \mathbf{b}_{a_i})\delta t^2 \\
\hat{\boldsymbol{\beta}}_{i+1}^{b_k} &= \hat{\boldsymbol{\beta}}_{i}^{b_k} + \mathbf{R}(\hat{\boldsymbol{\gamma}}_{i}^{b_k})(\hat{\mathbf{a}}_i - \mathbf{b}_{a_i})\delta t \\
\hat{\boldsymbol{\gamma}}_{i+1}^{b_k} &= \hat{\boldsymbol{\gamma}}_{i}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{w_i})\delta t \end{bmatrix}
\end{aligned} \tag{2.3}
$$

通过计算可以写下 IMU 测量模型所其对应的协方差 $\mathbf{P}_{b_{k+1}}^{b_k}$：

$$
\begin{bmatrix} \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_w^{b_k}(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w + \frac{1}{2}\mathbf{g}^w\Delta t_k^2 - \mathbf{v}_{b_k}^w\Delta t_k) \\ \mathbf{R}_w^{b_k}(\mathbf{v}_{b_{k+1}}^w + \mathbf{g}^w\Delta t_k - \mathbf{v}_{b_k}^w) \\ \mathbf{q}_{b_k}^{w^{-1}} \otimes \mathbf{q}_{b_{k+1}}^w \\ \mathbf{b}_{a b_{k+1}} - \mathbf{b}_{a b_k} \\ \mathbf{b}_{w b_{k+1}} - \mathbf{b}_{w b_k} \end{bmatrix}. \tag{2.4}
$$

估计器初始化

单目紧耦合 VIO 是一个高度非线性的系统。由于单目相机无法直接观测到尺度，因此，如果没有良好的初始值，很难直接将这两种测量结果融合在一起。当 IMU 测量结果被大偏置破坏时，情况就变得更加复杂了。事实上，初始化通常是单目 VINS 最脆弱的步骤。需要一个鲁棒的初始化过程以确保系统的适用性。通过对齐 IMU 预积分与纯视觉 SfM 结果，我们可以粗略地恢复尺度、重力、速度，甚至偏置。这足以引导非线性单目 VINS 估计器。

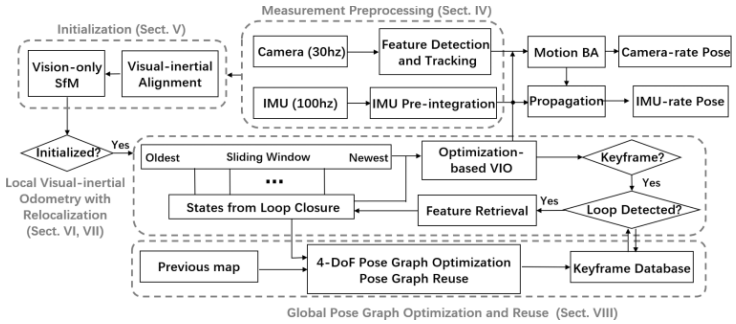


图 2.2: 总体架构图

开源代码链接： `https://github.com/HKUST-Aerial-Robotics/VINS-Mono`

第 3 章　路径规划

3.1　路径规划所使用的地图的建立

3.1.1　代价地图的建立

代价地图（COSTMAP）是一种机器人收集传感器信息建立和更新的二维或三维地图。在 ROS 中，红色部分代表 costmap 中的障碍物，蓝色部分表示通过机器人内切圆半径膨胀出的障碍，红色多边形是 footprint(机器人轮廓的垂直投影)。红色部分代表 costmap 中的障碍物，蓝色部分表示通过机器人内切圆半径膨胀出的障碍，红色多边形是 footprint(机器人轮廓的垂直投影)。可以从下图简要了解。

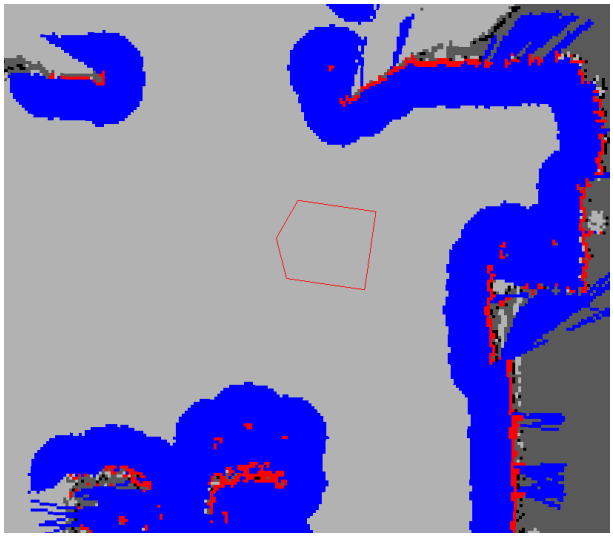


图 3.1: 代价地图表示形式

ROS 的代价地图（costmap）采用网格（grid）形式，每个网格的值（cell cost）从 0 255。分成三种状态：被占用（有障碍）、自由区域（无障碍）、未知区域。其原理也可以由下图得出。

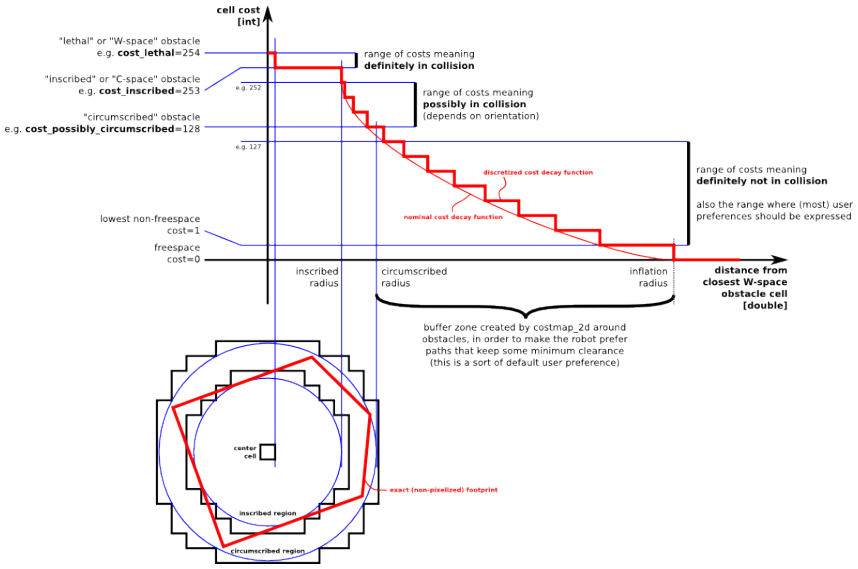


图 3.2: 代价地图原理

3.1.2 代价地图的更新

通过不断地接受激光雷达扫描的数据（laserScan），不断更新代价地图（costmap）。由下图 [?] 可以看到代价地图的生成流程。

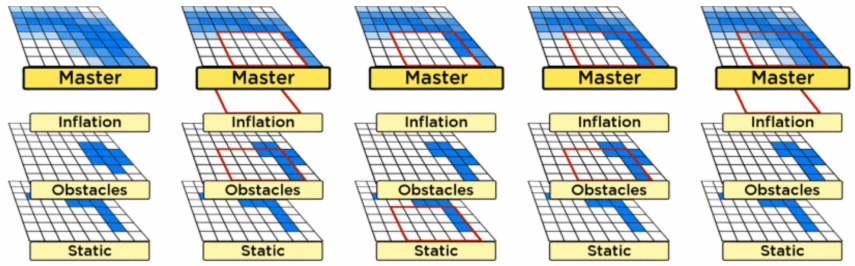


图 3.3: 代价地图生成过程

3.2 全局路径规划：Dijskra

该算法 [?] 由经典最短路径算法 Dijskra 在 2D 占据地图上的应用得到。是一种静态路网中求解最短路径最有效的直接搜索方法，也是解决许多搜索问题的有效算法。算法中的距离估算值与实际值越接近，最终搜索速度越快。公式表示为：

$$f(n) = g(n) + h(n) \tag{3.1}$$

其中 $f(n)$ 是从初始状态经由状态 n 到目标状态的代价估计，$g(n)$ 是在状态空间中从初始状态到状态 n 的实际代价，$h(n)$ 是从状态 n 到目标状态的最佳路径的估计代价。对于路径搜索问题，状态就是图中的节点，代价就是距离。

3.3 全局路径规划：混合 A* 算法 (Hybrid A*)

3.3.1 提出背景

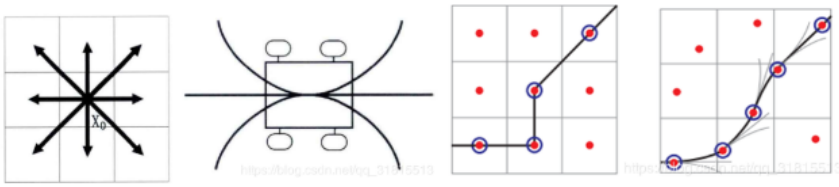
对于传统的 Dijiskra [?]/A* [?] 等路径规划算法都是以最短路径为唯一目标，这导致路径的生成取决于地图网格的分辨率而不考虑车辆的实际模型。所以，这种路径生成方式对于机器人的机械结构以及操控系统都是极大的挑战和负担。于是，在 2007 年斯坦福大学首次提出了 Hybrid A* 算法 [?]，通过考虑机器人、自动驾驶汽车的非整体约束，Hybrid A* 产生的路径更加平滑、实践性更强。

3.3.2 混合 A* 算法与传统 A* 算法的对比

以下表格展示了传统 A* 算法 [?] 和 Hybrid A* 算法 [?] 的区别：

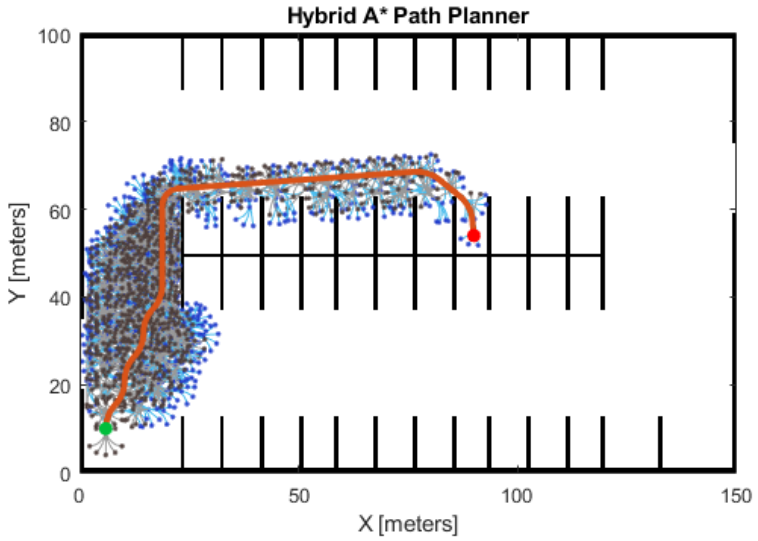
表 3.1: 对比

性质	Hybrid A*	A*
维数	(x, y, θ, R)	(x, y)
H(n)	Max(Rends_Shepp, A*)	Manhattan,Euclidean
节点	车辆的运动学模型为节点（2）	二维平面坐标点（1）
节点与节点连接处	交点可以不是栅格顶点（4）	交点是栅格顶点（3）
缺点	不具有完备性	不满足车辆运动学特性



(1) (2) (3) (4)

3.3.3 最终展示结果



3.4 全局路径规划：RRT* 的改进方法

该算法 [?] 由 RRT 改进而来，同时对轨迹进行了平滑处理。该算法方法通常能用更小的树在更短的时间内产生更平滑的路径。对于 RRT*，该方法还生成最短路径，并在给定更多规划时间时实现成本最低的解决方案。

3.4.1 RRT / RRT* 的特点

通过抽样来在已知的地图上建立无向图，进而通过搜索方法寻找相对最优的路径。不同点在于，PRM 算法在一开始就通过抽样在地图上构建出完整的无向图，再进行图搜索；而 RRT 算法则是从某个点出发一边搜索，一边抽样并建图。

3.4.2 RRT / RRT* 的缺点

RRT 算法是概率完备的：只要路径存在，且规划的时间足够长，就一定能确保找到一条路径解。注意“且规划的时间足够长”这一前提条件，说明了如果规划器的参数设置不合理（如搜索次数限制太少、采样点过少等），就可能找不到解。

3.4.3 改进方法：POSQ

由于 RRT 产生了很多点，所以根据原来的反馈方程，目标会在每个点处停下来。本方法通过修改反馈方程，使得在多次扩展中的恒定正向速度有所需的最大值。即以下

闭环模型：

$$\begin{aligned}\dot{\rho} &= -K_\rho \cos\alpha \tanh(K_v\rho),\\ \dot{\alpha} &= K_\rho \frac{\sin\alpha}{\rho}\tanh(K_v\rho) - K_\alpha\alpha - K_\phi\phi,\\ \dot{\phi} &= -K_\alpha\alpha - K_\phi\phi.\end{aligned} \tag{3.2}$$

由此得到如图所示效果：

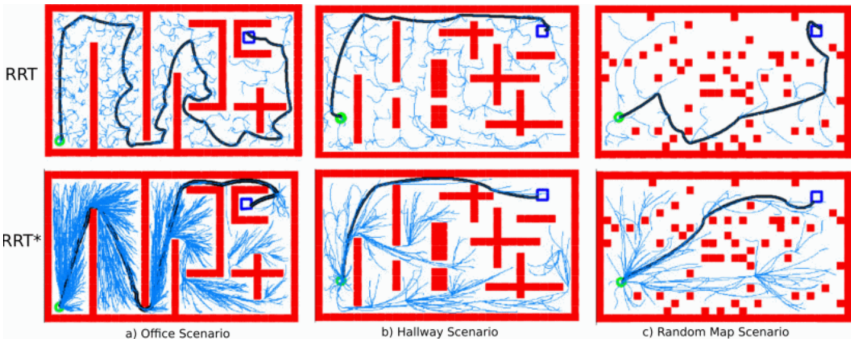


图 3.4: RRT-EXTEND 的效果图