北京化工大学

Beijing University of Chemical Technology

Computing Methods

AGV 系统的导航与控制：综述

刘本佳，庄梓博，贺怀宇，李昊，王昕凯
(排名不分先后)

目录

创建日期：2020 年 3 月 25 日
更新日期：2020 年 4 月 16 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　导航部分

2.1　SLAM 算法描述

2.1.1　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

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

2.1.2 激光雷达:GMAPPING

该算法 [?] 是基于滤波 SLAM 框架的常用开源 SLAM 算法。基于 RBpf 粒子滤波算法，即将定位和建图过程分离，先进行定位再进行建图，即有以下表达式：

$$p\left(x_{1:t}, m | z_{1:t},\ u_{1:t-1}\right) = p\left(m | x_{1:t}, z_{1:t}\right) \cdot p\left(x_{1:t} | z_{1:t}, u_{1:t-1}\right) \tag{2.2}$$

同时，GMAPPING 在 RBpf 算法上做了两个主要改进：**改进提议分布**和**选择性重采样**。

改进的提议分布不但考虑运动（里程计）信息还考虑最近的一次观测（激光）信息这样就可以使提议分布的更加精确从而更加接近目标分布。选择性重采样通过设定阈值，只有在粒子权重变化超过阈值时才执行重采样从而大大减少重采样的次数。

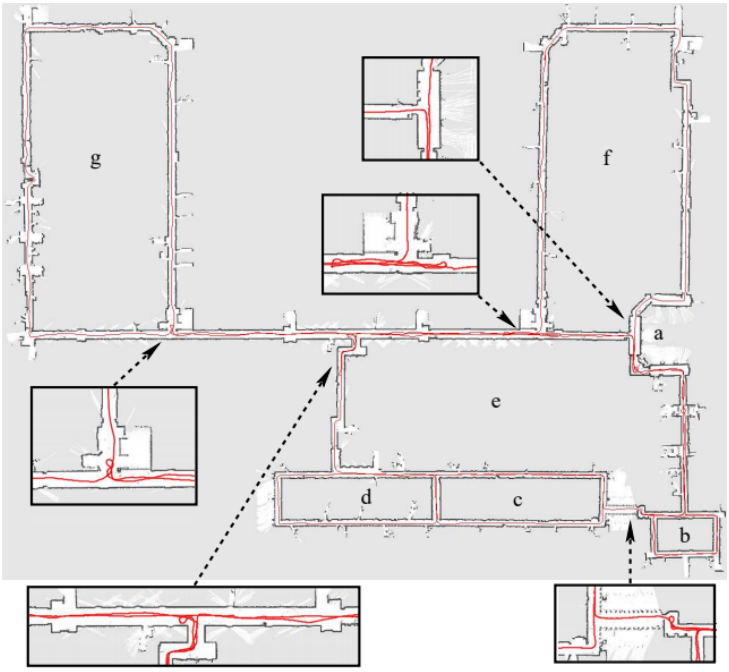

图 2.1: 机器人从标记为 a 的点开始，然后遍历标记为 b 的第一个循环。然后，它在标记为 c，d 的循环中移动，并移回到标记为 a 的位置和标记为 b 的循环中。它访问标记为 f 和 g 的两个大循环。环境的大小为 250 m x 215 m，机器人行进了 1.9 km。所描绘的地图已生成 80 个粒子。矩形显示地图的多个部分的放大图。

开源代码链接： `http://wiki.ros.org/gmapping/`

2.1.3 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。

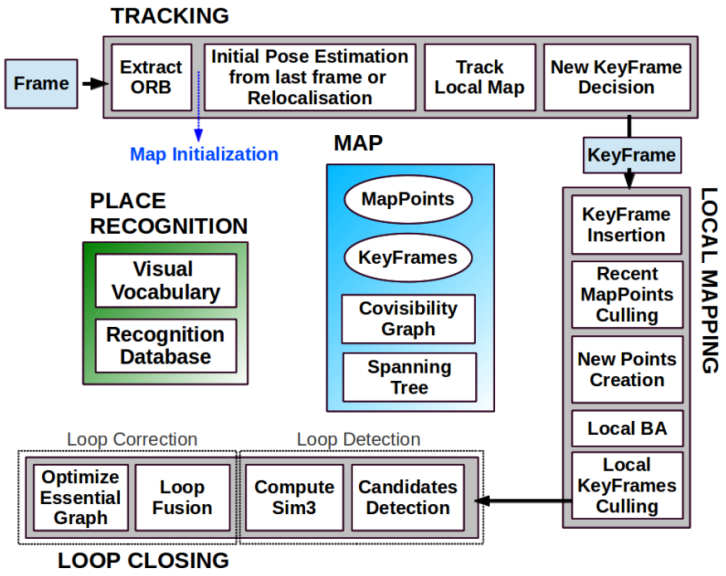

图 2.2: ORB-SLAM 主要分为三个线程进行，也就是如图所示的，分别是 Tracking、LocalMapping 和 LoopClosing

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。

$$\{\mathbf{R},\mathbf{t}\} = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i\in\mathcal{X}} \rho\left(\left\|\mathbf{x}^i_{(\cdot)} - \pi_{(\cdot)}\left(\mathbf{R}\mathbf{X}^i + \mathbf{t}\right)\right\|^2_{\Sigma}\right) \tag{2.3}$$

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`

2.1.4 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：

$$\begin{aligned}
\hat{\boldsymbol{\alpha}}_{i+1}^{b_k} &= \hat{\boldsymbol{\alpha}}_{i}^{b_k} + \hat{\boldsymbol{\beta}}_{i}^{b_k}\delta t + \frac{1}{2}\mathbf{R}(\hat{\boldsymbol{\gamma}}_{i}^{b_k})(\hat{\mathbf{a}}_i - \mathbf{b}_{a_i})\delta t^2 \\
\hat{\boldsymbol{\beta}}_{i+1}^{b_k} &= \hat{\boldsymbol{\beta}}_{i}^{b_k} + \mathbf{R}(\hat{\boldsymbol{\gamma}}_{i}^{b_k})(\hat{\mathbf{a}}_i - \mathbf{b}_{a_i})\delta t \\
\hat{\boldsymbol{\gamma}}_{i+1}^{b_k} &= \hat{\boldsymbol{\gamma}}_{i}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{w_i})\delta t \end{bmatrix}
\end{aligned} \tag{2.4}$$

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

$$\begin{bmatrix} \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_w^{b_k}(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w + \frac{1}{2}\mathbf{g}^w\Delta t_k^2 - \mathbf{v}_{b_k}^w\Delta t_k) \\ \mathbf{R}_w^{b_k}(\mathbf{v}_{b_{k+1}}^w + \mathbf{g}^w\Delta t_k - \mathbf{v}_{b_k}^w) \\ \mathbf{q}_{b_k}^{w^{-1}} \otimes \mathbf{q}_{b_{k+1}}^w \\ \mathbf{b}_{a b_{k+1}} - \mathbf{b}_{a b_k} \\ \mathbf{b}_{w b_{k+1}} - \mathbf{b}_{w b_k} \end{bmatrix}. \tag{2.5}$$

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。

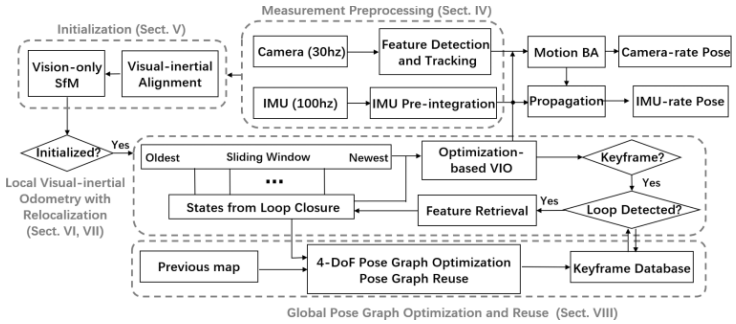

图 2.3: 总体架构图

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

2.2 路径规划算法描述

2.2.1 总体概述

路径规划是导航系统的一部分，其总结来看可以分为全局规划层（global planner)和局部规划层（local planner）。

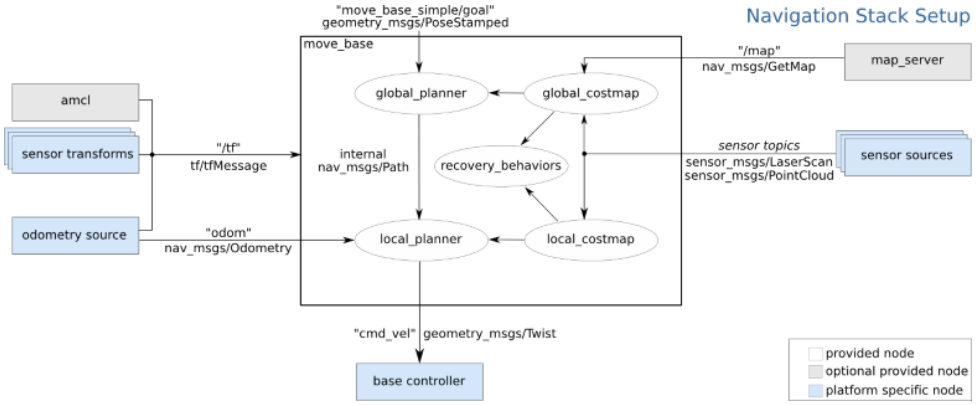

图 2.4: 导航总体架构图

2.2.2 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。

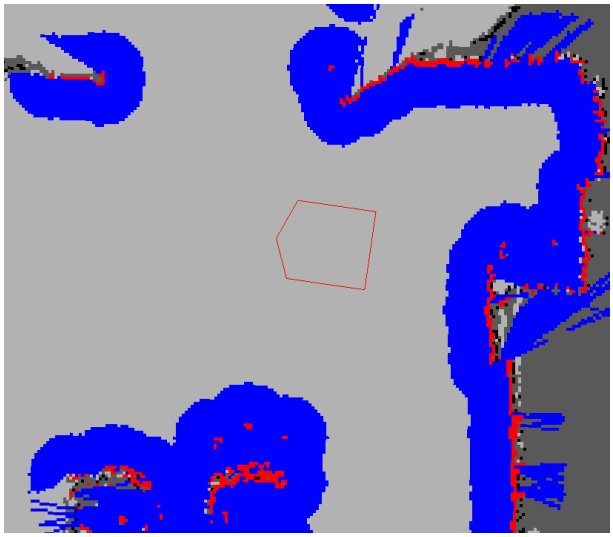

图 2.5: 代价地图表示形式

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。

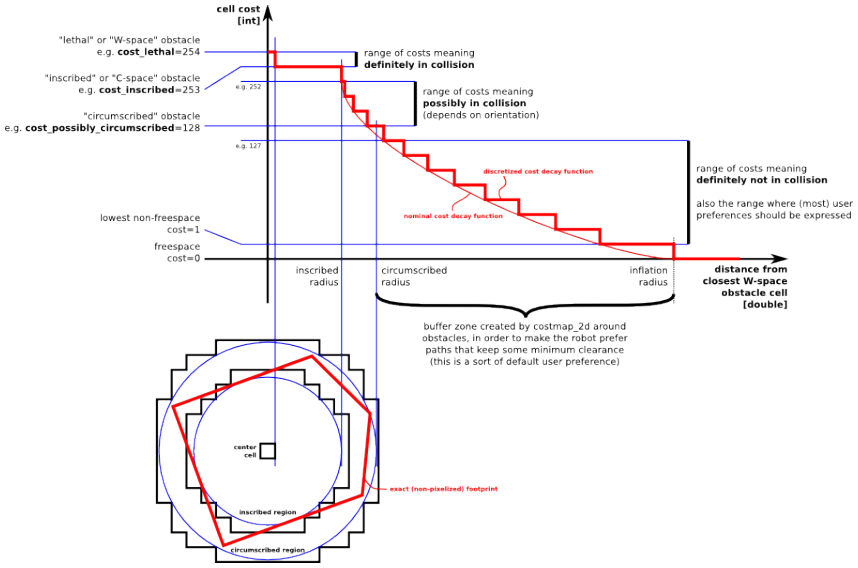

图 2.6: 代价地图原理

2.2.3 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。

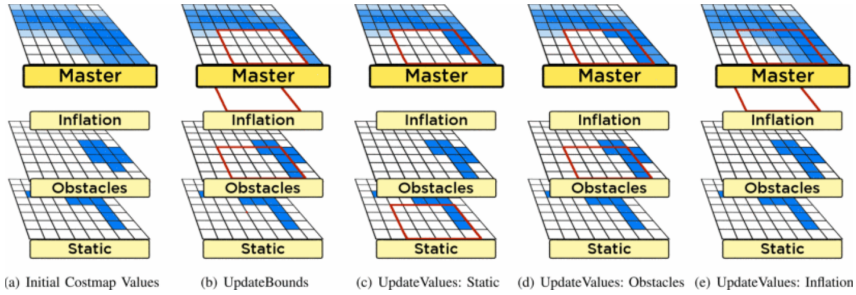

图 2.7: 代价地图生成过程

2.2.4 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：

$$f(n) = g(n) + h(n) \tag{2.6}$$

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。

2.2.5 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。

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。

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) 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：

$$\begin{aligned}
\dot{\rho} &= -K_\rho \cos\alpha \tanh(K_v \rho), \\
\dot{\alpha} &= K_\rho \frac{\sin\alpha}{\rho} \tanh(K_v \rho) - K_\alpha \alpha - K_\phi \phi, \\
\dot{\phi} &= -K_\alpha \alpha - K_\phi \phi.
\end{aligned} \tag{2.7}$$

由此得到如图所示效果：

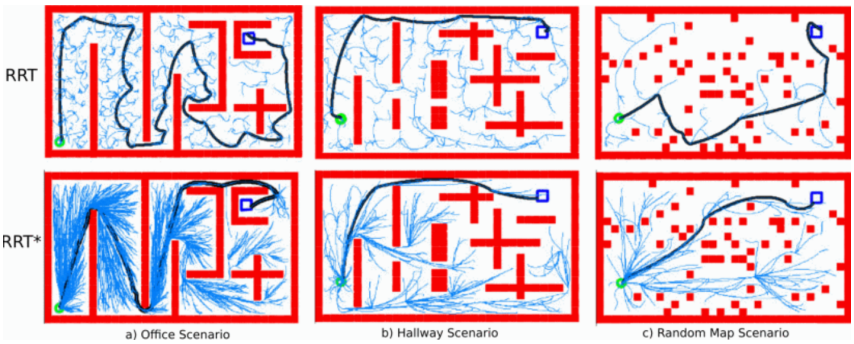

图 2.8: RRT-EXTEND 的效果图

第 3 章　控制部分

3.0.1　AGV 姿态控制

随着工业物流技术的进步，AGV 已越来越广泛地应用于工业制造和仓储场景。为了使 AGV 适应不同的工作场景，提高运行稳定度，AGV 车身需及时响应计划的路径，完成物理位置及车身姿态的变换。控制方法为，在 AGV 车身的每一侧安装一个直流电动机，两个直流电动机安装有编码器，控制器通过编码器的反馈，控制电动轮的旋转，实现物理位置及车身姿态的变换。通过使用 PID 控制，AGV 系统可以根据计划的路径快速改变位姿并修正错误。

3.0.2　AGV 的运动学分析

2 轮差速驱动 AGV，即 AGV 的 2 驱动轮分别由 1 台直流伺服电机驱动。2 驱动轮安装在车体的中间部位，车体前后各有 1 个万向轮支撑。AGV 在绝对坐标系中可简化为图 3.9 所示模型 [?]。图中 L 为 AGV 左右轮之间的距离，D 为驱动轮直径，ω_L、ω_L、ω 分别为左轮、右轮和车体转动角速度，v_L、v_R、v 分别为左轮、右轮和车体几何中心线速度。左、右轮的运动学方程分别为

$$\dot{x}\cos\theta + \dot{y}\sin\theta - \frac{L}{2}\dot{\theta} = \frac{D}{2}\omega_{\mathrm{L}} = v_{\mathrm{L}} \tag{3.1}$$

$$\dot{x}\cos\theta + \dot{y}\sin\theta - \frac{L}{2}\dot{\theta} = \frac{D}{2}\omega_{\mathrm{R}} = v_{\mathrm{R}} \tag{3.2}$$

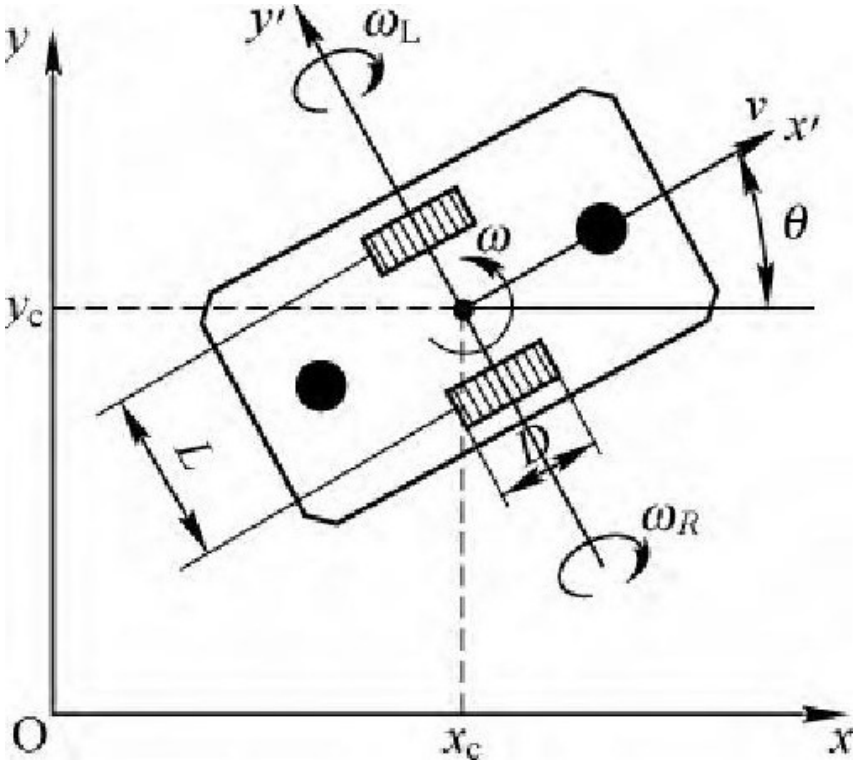

图 3.1: AGV 运动学模型

AGV 车体转动角速度

$$\omega = \dot{\theta} = \frac{\omega_R r - \omega_L r}{L} = \frac{v_R - v_L}{L} \tag{3.3}$$

AGV 车体几何中心的线速度

$$v = \frac{\omega_n r + \omega_L r}{2} = \frac{v_R + v_L}{2} \tag{3.4}$$

式中: 驱动轮的半径 $r = \frac{D}{2}$。
定义速度向量

$$U = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ -1/L & 1/L \end{bmatrix} \begin{bmatrix} v_L \\ v_R \end{bmatrix} \tag{3.5}$$

假设车轮与地面之间只有滚动，没有滑动，即满足理想约束情况，有

$$\dot{x}\cos\theta - \dot{y}\sin\theta = 0 \tag{3.6}$$

则 AGV 的运动学方程可以表示为

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.7}$$

由 AGV 的运动学方程可看出，通过控制 AGV 车体线速度 v 和角速度 ，即可实现 AGV 在绝对坐标系中不同位姿的运动。

3.0.3 AGV 姿态控制算法

导航控制算法控制机器人沿着规划路径行驶。为了快速准确地跟踪目标点，PID 作为主要的调节器 [?]。计算由摄像头与 IMU 及轮速计经卡尔曼滤波融合检测到的 AGV 行程轨迹的航向偏差和航向偏差作为 PID 控制器的输入，并将姿态调整量用作输出，控制器实现对预定路径的跟踪。PID 控制原理图如图 3.10 所示。

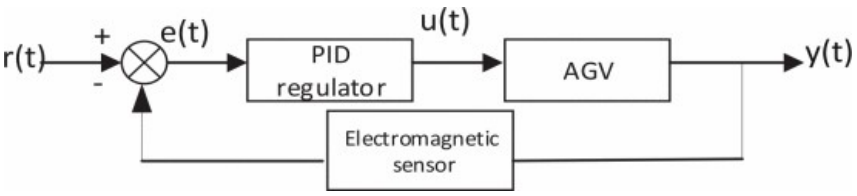

图 3.2: PID 控制原理图

离散点的 PID 表达式为

$$u(k) = K_p \left\{ e(k) + \frac{T}{T_I} \sum_{i=0}^{k} e(i) + \frac{T_d}{T}[e(k) - e(k-1)] \right\} \tag{3.8}$$

其中 K_p 是比例增益；T_I 是积分系数，T_d 是微分时间常数；$u(k)$ 是控制器采样为 k 时的输出；T 是采样周期。

简化后

$$\Delta u(k) = K_p[e(k) - e(k-1) + K_l e(k) + K_D[e(k) - 2e(k-1) + e(k-2)]$$

在公式中，比例增益决定系统的响应速度和控制精度，积分环节的作用是消除残差，提高控制性能，微分环节在大偏移发生前加快位姿修正的速率，从而快速完成小车轨迹的修正。因此，PID 控制器可以有效地结合比例环节和微分环节的功能，既提高了系统的稳定性，又加快了系统的响应速度。

第 4 章　实验部分

4.1　GUI 控制程序演示

4.2　仿真模型构建

4.3　流程图

4.4　运行情况