ROS2 Odometry 介紹與原理

1. Odometry 的概念

Odometry（里程計是移動機器人定位與導航的基礎技術之一，用來估計機器人從初始位置出發後的 **相對位姿（pose）**，包括位置 (x, y) 與朝向 θ。

在 ROS2 中，Odometry 的資訊通常透過 `/odom` topic 發布，類型為 `nav_msgs/msg/Odometry`，包含：

- **位置(Position)**：`pose.pose.position`（x, y, z）
- **朝向(Orientation)**：`pose.pose.orientation`（四元數 quaternion）
- **速度(Twist)**：`twist.twist`（線速度與角速度）

Odometry 主要依賴 **編碼器 (encoder)**、IMU 或其他感測器數據，並結合運動模型估算位置。

---

2. Odometry 的原理

Odometry 的核心原理是 **根據機器人的運動模型，將速度或輪子旋轉量積分成位移**。
以常見的 **差速驅動機器人 (Differential Drive Robot)** 為例，其運動原理如下：

- 機器人有兩個輪子，分別為左輪 v_L 和右輪 v_R 線速度（或編碼器脈衝轉換後的速度）。
- 車輪間距為 b（兩輪軸距）。

2.1 差速驅動 kinematic 模型

假設機器人的位姿為 (x, y, θ)，其瞬時運動可以表示為：

$$v = \frac{v_R + v_L}{2} \quad \text{(線速度)}$$

$$\omega = \frac{v_R - v_L}{b} \quad \text{(角速度)}$$

其中：

- v 為機器人沿前進方向的速度
- ω 為機器人旋轉速度
- b 為左右輪軸距

2.2 Odometry 更新公式

對於小時間步長 Δt，機器人的位姿更新公式為：

$$x_{t+1} = x_t + v \cdot \cos(\theta_t) \cdot \Delta t$$

$$y_{t+1} = y_t + v \cdot \sin(\theta_t) \cdot \Delta t$$

$$\theta_{t+1} = \theta_t + \omega \cdot \Delta t$$

這是 **Euler 積分法** 的基本形式，也可以使用更精確的 **Runge-Kutta 方法** 處理高速運動或大時間步長情況。

2.3 車輪編碼器轉速度公式

如果使用編碼器，輪子線速度可以透過以下公式計算：

$$v_L = \frac{2\pi R}{N} \cdot \frac{\Delta \text{ticks}_L}{\Delta t}$$

$$v_R = \frac{2\pi R}{N} \cdot \frac{\Delta \text{ticks}_R}{\Delta t}$$

- R：車輪半徑
- N：每圈脈衝數 (encoder resolution)
- $\Delta \text{ticks}_{L/R}$：左/右輪在 Δt 時間內的編碼器脈衝增量
  代入 kinematic 模型，即可計算機器人瞬時速度 v, ω，進而更新位姿。

2.4 ROS2 中的實現

在 ROS2 中，Odometry 通常透過以下步驟實現：

1. **讀取編碼器數據**（透過自訂的驅動節點或硬體介面）
2. **轉換為車輪線速度** v_L, v_R
3. **計算機器人速度** v, ω
4. **更新位姿** (x, y, θ)
5. **發布 `/odom` Topic，可供導航套件或 TF 使用

典型 ROS2 節點使用 `rclcpp` 或 Python API，發布的消息包含：

```
from nav_msgs.msg import Odometry
odom_msg.pose.pose.position.x = x
odom_msg.pose.pose.position.y = y
```

```
odom_msg.pose.pose.orientation = quaternion_from_euler(0, 0, theta)
odom_msg.twist.twist.linear.x = v
odom_msg.twist.twist.angular.z = omega
```

---

3. 數學摘要

名稱	符號	計算公式
左輪速度	v_L	$\frac{2\pi R}{N} \cdot \frac{\Delta \text{ticks}_L}{\Delta t}$
右輪速度	v_R	$\frac{2\pi R}{N} \cdot \frac{\Delta \text{ticks}_R}{\Delta t}$
線速度	v	$\frac{v_R + v_L}{2}$
角速度	ω	$\frac{v_R - v_L}{b}$
位姿更新 x	x_{t+1}	$x_t + v\cos\theta\Delta t$
位姿更新 y	y_{t+1}	$y_t + v\sin\theta\Delta t$
位姿更新 θ	θ_{t+1}	$\theta_t + \omega\Delta t$

---

4. 注意事項

1. **累積誤差 (drift)**：Odometry 是相對定位，長時間會累積誤差，需要與 IMU 或 SLAM 做融合。
2. **小角度假設**：Euler 積分適用於小步長或小角度運動，若步長過大需使用更精確的積分方法。
3. **坐標系**：
   - `/odom`：相對起點的位姿
   - `/base_link`：機器人自身座標
   - `/map`：全局地圖坐標（通常透過 SLAM 或 AMCL 對 `/odom` 做校正）