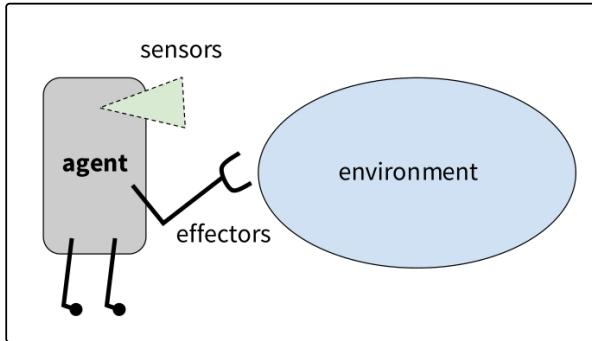


 Robotics
 什么是机器人 (Robots)?

机器人是可以执行任务的物理代理 (agent)，它们通过 导航 或 操作物理环境 来完成任务.



组成部分：

- ① **Sensors (传感器)**: 帮助机器人感知环境.
- ② **Effectors (执行器)**: 使机器人能够对环境施加物理作用.
- ③ **Agent (代理)**: 指代机器人本身.

 传感器 (Sensors)

传感器是机器人的“眼睛”和“耳朵”，让机器人能感知周围环境的信息.

 例子：

- ① **红外传感器**: 检测障碍物.
- ② **相机**: 捕捉图像.
- ③ **陀螺仪**: 感知方向和姿态.

假设机器人通过传感器获取环境信息 s ，例如距离 d ，传感器通过公式 $s = f(e)$ 转化环境信息 e 为数字信号。比如，超声波传感器通过计算声波返回的时间 t 来得出距离：

$$d = \frac{vt}{2}$$

其中 v 是声速.

 执行器 (Effectors)

执行器是机器人用来对环境进行物理作用的组件，比如腿、轮子、机械臂等.

 例子：

- ① **轮子**: 让机器人移动.
- ② **机械臂**: 抓取物体.

如果执行器控制机器人的移动速度 (v)，则其位移可以通过公式：

$$x(t) = x_0 + vt$$

机器人通过控制 v 改变自身位置.

环境 (Environment)

环境是机器人工作的场景，包括任何需要导航或操作的物理空间。机器人通过不断地感知 (sensors) 和作用 (effectors) 与环境互动。

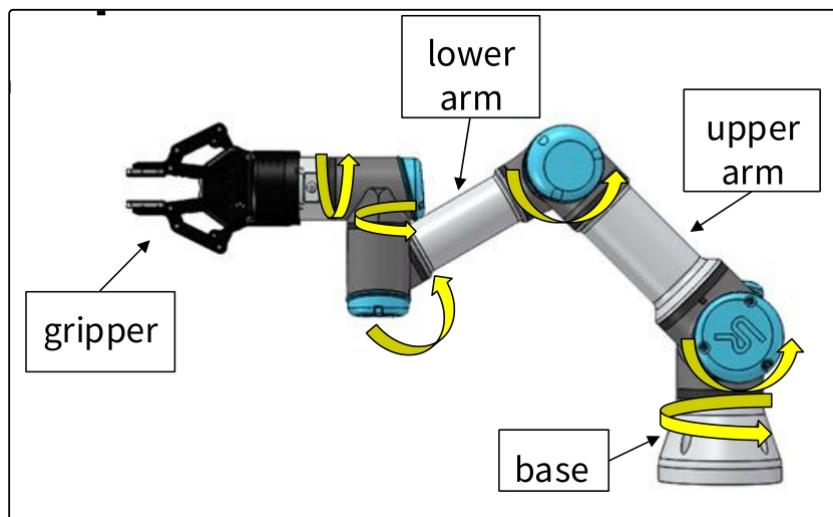
例子：

- 清扫机器人：环境是家庭地面。
- 工厂机械臂：环境是装配流水线。

机器人的种类

Manipulator Robots (机械臂机器人)

机械臂机器人被固定在环境中，通常用于工业场景，比如流水线上的装配工作。它们通过多个可控关节 (joints) 将执行器 (effectors) 移动到所需位置。



组成部分

1. Base (底座)：提供支撑和稳定性。
2. Joints (关节)：每个关节提供一个旋转或移动的自由度。
3. Upper Arm 和 Lower Arm (上下臂)：用于扩展操作范围。
4. Gripper (抓手)：执行具体操作，比如抓取物体。

机械臂的运动可以通过正向运动学 (Forward Kinematics) 来计算末端位置：

$$x = f(\theta_1, \theta_2, \dots, \theta_n)$$

其中 $\theta_1, \theta_2, \dots, \theta_n$ 是关节角度， f 是将关节角度转换为末端执行器位置的函数。

应用场景

农场采摘



医疗手术



💡 Mobile Robots (移动机器人)

💡 定义

移动机器人可以在环境中移动，而不是固定在某一位置。常见的类型包括：

1. 无人地面车辆 (Unmanned Ground Vehicles)
2. 无人机 (Unmanned Aerial Vehicles)
3. 无人水下潜艇 (Autonomous Underwater Vehicles)

🌐 应用场景

①

地面机器人：在火星上探索。



② 无人机：用于快递或监控。



③ 水下机器人：用于海洋探索。



移动机器人的路径规划通常使用图算法或微积分来计算最优路径。例如，最短路径问题可以用 Dijkstra 算法解决。

💡 Mobile Manipulators (移动机械臂)

💡 定义

移动机械臂结合了移动性和操作能力，既能在环境中移动，又能通过机械臂完成任务。这种机器人多用于需要灵活操作的场景。



特点

1. 结合移动性和操作能力：既能移动到目标地点，又能使用机械臂执行任务。

2. 缺乏固定机械臂的刚性：移动平台可能会影响操作的稳定性。

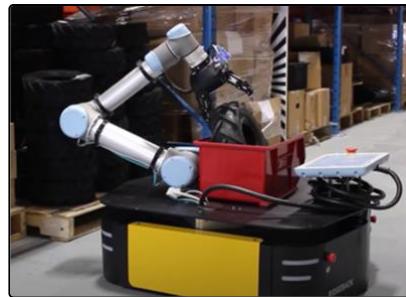
移动机械臂的控制涉及多体动力学和路径规划：

$$\mathbf{p}_{\text{end}} = f(\mathbf{x}_{\text{base}}, \theta_1, \theta_2, \dots)$$

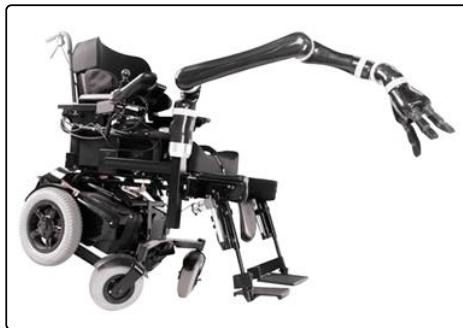
其中 \mathbf{x}_{base} 表示移动平台的位置， $\theta_1, \theta_2, \dots$ 表示机械臂的关节角度。

应用场景

仓库搬运机器人



行动辅助器



总结

这三类机器人体现了不同的特点和应用：

1. 机械臂机器人：固定在一个地方，专注于高精度操作。

2. 移动机器人：在环境中移动，用于探索或运输。

3. 移动机械臂：结合移动性和操作能力，更灵活但稳定性稍弱。

Robot Hardware: Sensors

根据感知内容分类：

1. 环境 (Environment): 感知周围环境的信息。
2. 机器人的位置 (Robot's Location): 感知机器人在环境中的位置。
3. 机器人的内部配置 (Robot's Internal Configuration): 感知机器人的内部状态，比如关节角度。

Sensing the Environment (感知环境)

Range Finders (距离传感器)

- 测量与附近物体的距离。
- 应用：机器人需要避免碰撞或者导航到指定位置。
- 声波传感器 (Sonar Sensors): 声波以速度 v 传播，测量回声时间 t 来计算距离：

$$d = \frac{v \cdot t}{2}$$

其中 d 是距离， v 是声速， t 是回声返回所需时间。

- 立体视觉 (Stereo Vision): 通过两台摄像机捕获不同视角的图像，根据三角测量原理计算深度：

$$d = \frac{f \cdot B}{x_L - x_R}$$

其中 f 是摄像机的焦距， B 是两摄像头的基线距离， x_L 和 x_R 是左、右图像中的像素位置。

Sonar Sensors (声纳传感器)

- 发射声波，返回信号的时间和强度指示距离。
- 应用：水下无人机器人，例如自主水下潜艇。

Stereo Vision (立体视觉)

- 使用多个摄像头从不同视角捕获图像。
- 应用：导航、障碍物检测，或者建造环境的三维模型。

Modern Range Finders (现代距离传感器)

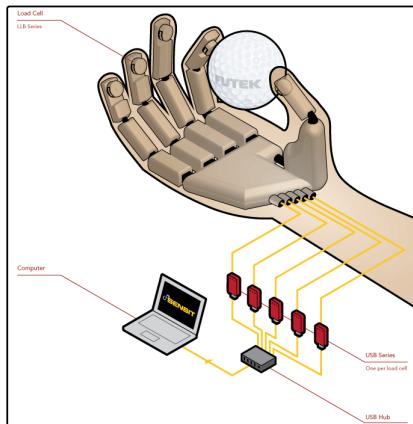
1. Optical Range Sensors (光学距离传感器):
 - 发射光线并测量光线反射回来的时间（飞行时间法，ToF）。
 - 应用：例如无人机导航、自动驾驶汽车检测障碍物。
 - 数学背景：光线传播的时间 t 与距离 d 的关系为： $d=c \cdot \frac{t}{2}$ 其中 c 是光速， t 是光信号的飞行时间。
2. Scanning LiDARs (扫描激光雷达):
 - 使用激光束扫描环境，生成高分辨率的距离数据。
 - 应用：自动驾驶车辆创建周围环境的三维地图。
 - 优势：可以覆盖更大的范围且精度极高。

📁 Other Range Sensors (其他距离传感器)

1. Radar Sensors (雷达传感器):

- 测量多公里范围内距离。
- 应用：航空、交通控制和无人驾驶汽车检测远距离障碍物。
- 数学背景：利用多普勒效应和回波时间计算目标的距离和速度。

2.



Tactile Sensors (触觉传感器):

- 通过物理接触测量范围。

- 应用：机械臂抓取物体时感知触碰力和距离。

- 特点：更适合近距离精确感知。

⚠️ Sensing the Robot's Location

💡 Global Positioning System (GPS)

- 原理：GPS通过测量机器人与多个卫星之间的距离，来确定机器人的位置。卫星会发送脉冲信号，机器人通过接收信号的时间差来计算距离。

机器人的位置可以通过三边测量法 (Trilateration) 计算：

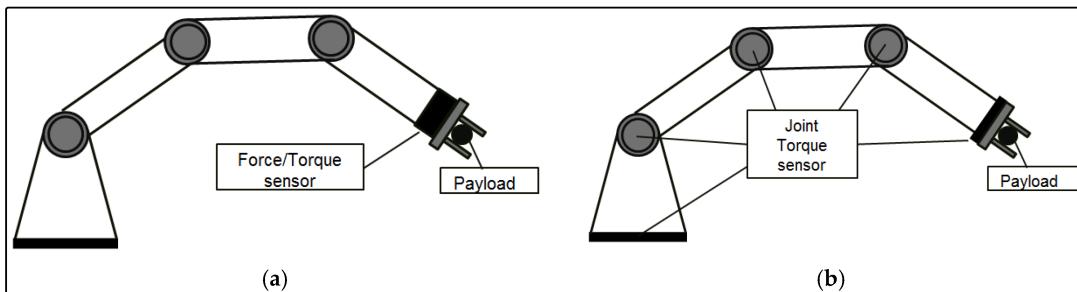
$$d = c \cdot t$$

其中 d 是距离， c 是信号传播速度（接近光速）， t 是信号传播时间。

使用三个卫星的位置 $(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)$ 和它们的距离 d_1, d_2, d_3 ，可以解三元方程组得到机器人的位置。

⚠️ Sensing the Robot's Internal Configuration

💡 Force Sensors (力传感器)



- 功能：检测机器人施加的力，例如机械臂抓握一个物体时施加的力。

- 应用：避免抓力过大损坏物体，或过小导致滑落。

力的计算公式为：

$$F = m \cdot a$$

其中 F 是力, m 是质量, a 是加速度.

💡 Torque Sensors (扭矩传感器)

在上图已经标注了 torque sensors

功能: 测量机器人关节处的旋转力矩.

应用: 控制机械臂关节的平滑运动.

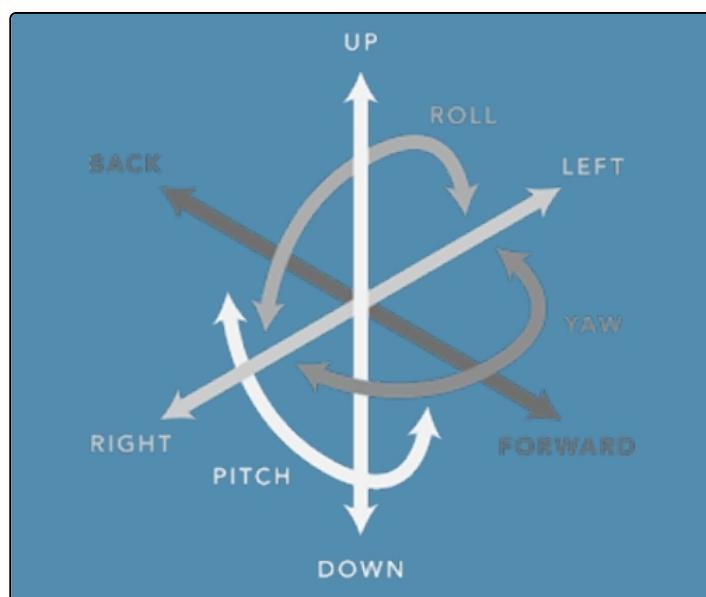
扭矩的计算公式为:

$$\tau = r \cdot F$$

其中 τ 是扭矩, r 是力臂的长度, F 是作用力.

⚠ Robot Hardware: Effectors

💡 Degree of Freedom (自由度, DoF)



自由度表示机器人或其执行器可以独立运动的方向.

位置自由度: 沿 x (左右)、 y (前后) 和 z (上下) 方向移动.

姿态自由度:

Yaw (偏航): 左右旋转.

Pitch (俯仰): 前后倾斜.

Roll (滚转): 侧向倾斜.

机器人在三维空间中的位置和方向可以用 6 个参数表示:

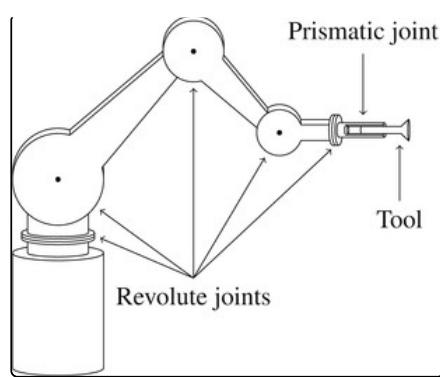
$$[x, y, z, \text{yaw}, \text{pitch}, \text{roll}]$$

⚠ Effectors Types (执行器类型)

1. 线性执行器 (Prismatic Joint, P)

产生直线运动.

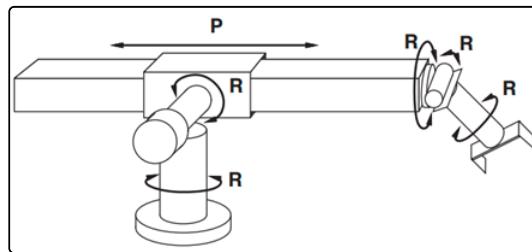
自由度: 1 (一个线性方向).



2. 旋转执行器 (Revolute Joint, R)

- 产生旋转运动。
- 自由度: 1 (一个旋转轴)。

斯坦福 Manipulator (Stanford Manipulator Example)



- 组合了 P (线性运动) 和 R (旋转运动):

- P : 负责执行器的前后伸缩。
- R : 负责执行器的旋转和定位。

应用场景

1. 工业机器人:

- 自由度较少, 例如 4-6 个自由度, 适合流水线作业。

2. 手术机器人:

- 高自由度 (7+), 用于精细的医疗操作

Example: ROBOTIS OpenMANIPULATOR-P

特点



- 关节数目: 具有 6 个可控关节 (自由度为 6)。

- 功能:

通过每个关节的旋转实现复杂的空间定位和操作。

可以灵活调整末端执行器（如抓手）的姿态，用于精准的物体抓取或操作。

💡 Robotic Perception

📝 定义

机器人感知是将传感器数据映射为环境的内部表示的过程。这包括从噪声数据中提取有意义的信息，用于导航、决策和控制。

💡 挑战

① **噪声数据**: 传感器测量可能受到环境干扰。

② **动态环境**: 环境是部分可观察的、不确定的，并可能不断变化。

💡 好的内部表示的特点

1. **信息足够 (Enough Information)**: 包含足够的信息以帮助机器人做出决策。

2. **高效更新 (Efficient Update)**: 结构化，能快速更新和调整。

3. **自然映射 (Natural Mapping)**: 内部变量与物理世界的变量对应。

💡 Bayesian Updating

📝 目标

在给定观测值和机器人动作的基础上，估计新的系统状态的概率分布，即：

$$P(X_{t+1} | Z_{1:t+1}, A_{1:t})$$

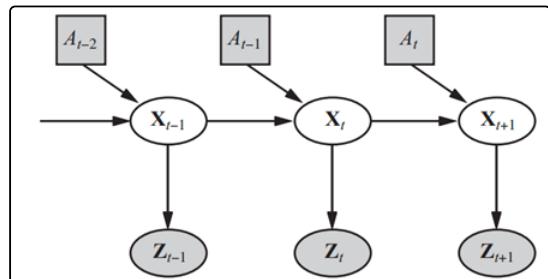
① X_t : 时间 t 的环境状态（包括机器人状态）。

② Z_t : 时间 t 的观测数据（传感器测量值）。

③ A_t : 时间 t 的动作。

💡 动态贝叶斯网络 (Dynamic Bayes Network)

贝叶斯更新的推理基于动态贝叶斯网络，其中依赖关系为：



1. X_t 由之前的状态 X_{t-1} 和动作 A_{t-1} 共同决定。

2. 当前观测 Z_t 与当前状态 X_t 相关。

3. 当前动作 A_t 基于当前观测 Z_t 决定。

这形成了一个时间序列模型，其中每一步状态可以递归计算。

递归更新公式

通过贝叶斯定理，可以递归地更新状态概率分布：

$$P(X_{t+1} | Z_{1:t+1}, A_{1:t}) = \eta \cdot P(Z_{t+1} | X_{t+1}) \int P(X_{t+1} | X_t, A_t) P(X_t | Z_{1:t}, A_{1:t-1}) dX_t$$

1. 先验预测 (Prediction):

根据当前状态 X_t 和动作 A_t 预测下一状态的分布：

$$P(X_{t+1} | X_t, A_t)$$

2. 后验校正 (Correction):

根据当前观测 Z_{t+1} ，更新预测分布：

$$P(Z_{t+1} | X_{t+1})$$

3. 归一化常数 (Normalization):

η 确保概率分布总和为 1.

更新步骤分解

1. 初始状态：

$$P(X_0)$$

是初始的状态分布。

2. 状态传播：

$$P(X_t | Z_{1:t}, A_{1:t-1}) = \int P(X_t | X_{t-1}, A_{t-1}) P(X_{t-1} | Z_{1:t-1}, A_{1:t-2}) dX_{t-1}$$

3. 结合观测：

$$P(X_t | Z_{1:t}, A_{1:t-1}) \propto P(Z_t | X_t) \cdot P(X_t | Z_{1:t-1}, A_{1:t-1})$$

Localisation and Mapping

1. Localisation (定位)

● 定义：确定环境中物体（包括机器人本身）的位置。

○ 机器人位姿 (Robot Pose):

○ 用笛卡尔坐标 (x_t, y_t) 和朝向 θ_t 表示。

○ 位置向量表示为：

$$X_t = (x_t, y_t, \theta_t)^T$$

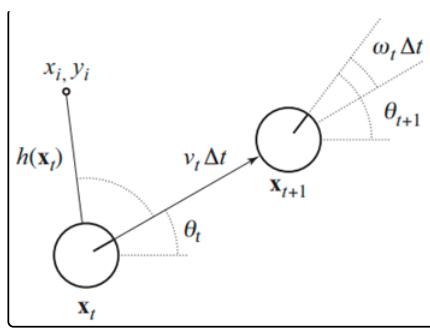
2. Robot Motion (机器人运动)

○ 机器人动作用两种速度 表示：

○ 平移速度 v_t ：描述机器人在 t 时刻的线速度。

○ 旋转速度 ω_t ：描述机器人在 t 时刻的角速度。

机器人在时间步长 Δt 后的新状态为：



位置更新：

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

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

朝向更新：

$$\theta_{t+1} = \theta_t + \omega_t \Delta t$$

3. Landmark (地标)

定义：环境中稳定、可识别的特征，作为观测的参考点。

测量：

距离 (Range): 机器人与地标之间的相对距离。

方位 (Bearing): 机器人与地标之间的相对角度。

观测模型：

假设地标在位置 (x_i, y_i) ，机器人位姿为 (x_t, y_t, θ_t) ，地标观测值 \hat{z}_t 可表示为：

$$\hat{z}_t = h(X_t) = \begin{bmatrix} \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2} \\ \arctan\left(\frac{y_i - y_t}{x_i - x_t}\right) - \theta_t \end{bmatrix}$$

4. Mapping (建图)

定义：构建环境的地图，用于导航和定位。

挑战：鸡与蛋问题 (Chicken-and-egg problem)：

定位依赖于地图：机器人需要知道地图信息来定位。

建图依赖于定位：机器人需要准确的位置来更新地图。

SLAM

SLAM，全称 Simultaneous Localization and Mapping，即同时定位与建图。

想象一下，你是一位探险家，迷失在未知的森林中，你需要：

1. 定位自己在哪里（本地化）。

2. 绘制一张地图，帮助自己探索并最终找到回家的路（建图）。

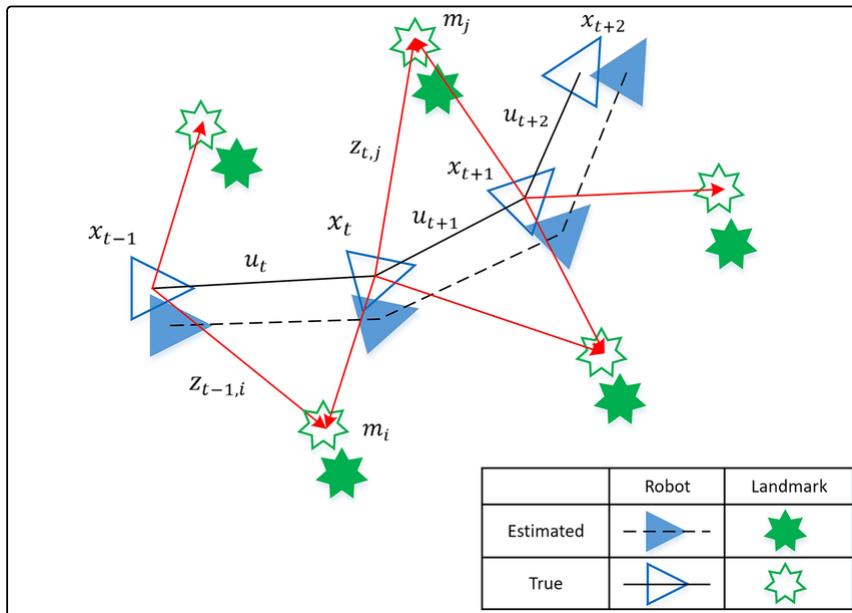
SLAM的难点在于：

① 你不知道自己的确切位置（没有现成的地图）。

② 地图还未绘制（你不知道周围环境）。

这就像一个“先有鸡还是先有蛋”的问题。

💡 SLAM的核心问题



SLAM解决的核心问题是：

1. 机器人如何根据传感器观察值，估计自身位置？

2. 机器人如何在运动中，逐步绘制周围环境的地图？

数学上，这涉及到概率推断：

我们希望通过已知的数据（传感器观测、控制输入等），求解以下后验概率：

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0)$$

这里：

① x_k : 机器人在第 k 时刻的位置和姿态（位置和方向）。

② m : 地图（例如地标的位置）。

③ $Z_{0:k}$: 从时间 0 到 k 时刻的所有传感器观测值。

④ $U_{0:k}$: 从时间 0 到 k 时刻的所有控制输入（例如机器人移动的方向和速度）。

⑤ x_0 : 初始位置。

换句话说，SLAM尝试根据传感器测量值和运动信息，估计机器人的位置和地图。

💡 SLAM的主要组成

1. 机器人状态 (State)

表示机器人的位置和方向，通常用向量 $x_k = (x, y, \theta)^T$ 表示，其中：

① x, y 是平面上的位置。

② θ 是机器人朝向的角度 (Heading)。

2. 传感器数据 (Observations)

机器人通过传感器测量地标的位置，相对机器人位置的距离和角度：

$$z_k = h(x_k)$$

通常，地标信息以“距离”和“方位角”描述，比如：

- 距离： $\sqrt{(x - x_i)^2 + (y - y_i)^2}$

- 方位角： $\arctan\left(\frac{y_i-y}{x_i-x}\right) - \theta$

3. 控制输入 (Control Inputs)

控制指令描述了机器人如何移动，包括：

- 平移速度 v_k

- 旋转速度 ω_k

新状态通过以下公式计算：

$$x_{k+1} = f(x_k, u_k)$$

其中 $u_k = (v_k, \omega_k)$ 是控制输入.

💡 SLAM的难点

1. 不确定性

传感器数据可能存在噪声（例如 GPS 精度误差），而控制指令可能存在执行误差。

2. 闭环问题 (Loop Closure)

如果机器人再次回到之前的位置，如何识别这个地方已经绘制在地图上？

例如，当机器人走了一圈又回到起点时，SLAM需要“闭环检测”来校正累积误差。

3. 实时性

机器人需要在运动中即时更新位置和地图，计算速度要求很高。

💡 SLAM的工作流程

1. 运动模型

根据控制输入预测机器人的位置：

$$x_{k+1} = f(x_k, u_k) + \text{噪声}$$

2. 观测模型

根据传感器数据测量周围地标：

$$z_k = h(x_k) + \text{噪声}$$

3. 贝叶斯更新

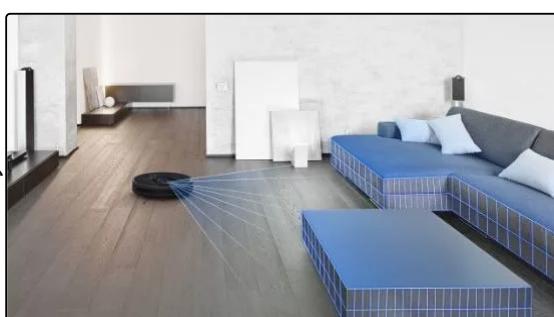
根据传感器观测值和控制输入，更新机器人位置和地图：

$$P(x_k, m | Z_{0:k}, U_{0:k})$$

💡 SLAM的应用

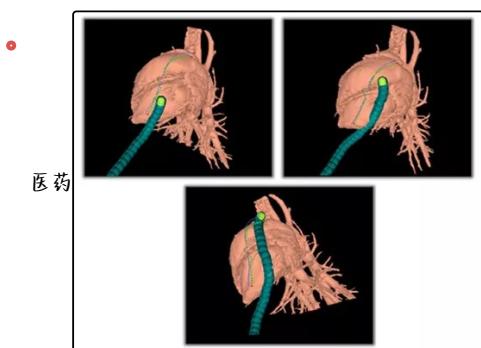
-

清洁机器人



清洁机器人实际上是同步定位和地图构建的最佳教程之一。如果没有 SLAM，清洁机器人只会在地上随机移动。它无法检测障碍物，这意味着它会不断撞到椅子或脚。它也无法“记住”它已经清洁过的区域，这首先违背了自动吸尘器的整个目的。

然而，借助 SLAM，机器人能够越过它已经覆盖的区域（测绘），并能够避开任何障碍物或地标（定位）。它还能够同时（同时）完成这两件事，这使其成为 SLAM 技术如何在家庭和其他地方发挥作用的完美示例。

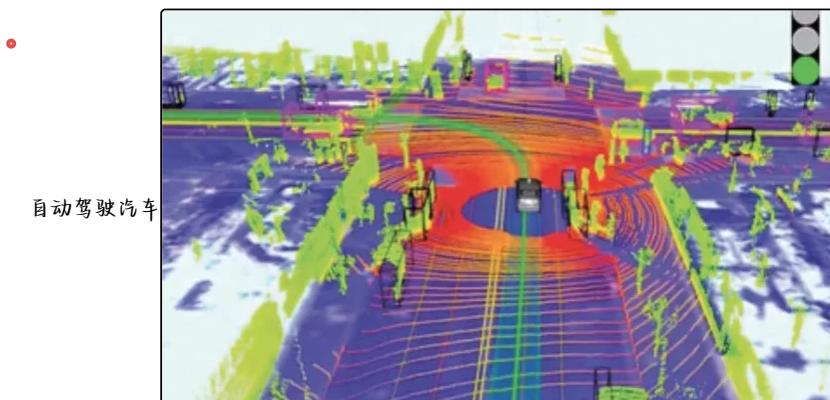


SLAM 正在医疗领域得到应用，帮助手术室里的医生进行更轻松、更微创的手术。通过在人体内外使用 SLAM 技术和自主技术，医生现在能够更快、更准确地识别问题并使用 SLAM 寻找解决方案。医疗 SLAM 可以让外科医生“鸟瞰”患者体内的物体，而无需进行深切。通过快速准确地显示患者体内动态物体的 3D 模型，SLAM 技术将在未来许多年继续用于辅助手术和其他医疗工作。

迪士尼公司获得了基于 SLAM 技术运行的“虚拟世界模拟器”的专利



通过持续跟踪游客不断变化的视点，虚拟世界模拟器可以让多个用户体验现实世界主题公园景点内的动态 3D 环境 - 无需使用眼镜或耳机。根据专利，这款虚拟世界模拟器未来可以使用 SLAM 技术将道具、艺术品甚至动画人物等一切东西直接投射到现实世界中。这项基于 SLAM 的技术比虚拟现实或增强现实更先进，能够彻底颠覆主题公园世界和整个娱乐行业。



自动驾驶汽车可以使用 SLAM 软件来识别从车道线到交通信号灯再到道路上的其他车辆等一切事物。SLAM 比 GPS 技术更准确、响应更快，很可能成为释放自动驾驶汽车真正潜力的关键。随着未来几年越来越多精确的 SLAM 解决方案的出现，自动驾驶汽车几乎肯定会成为大众市场首先应用这些解决方案的领域之一。

