智能无人机技术设计实践
-- 系统集成与初赛环境使用

联系方式：nics-efc@tsinghua.edu.cn

时间：2021年10月23日

目录

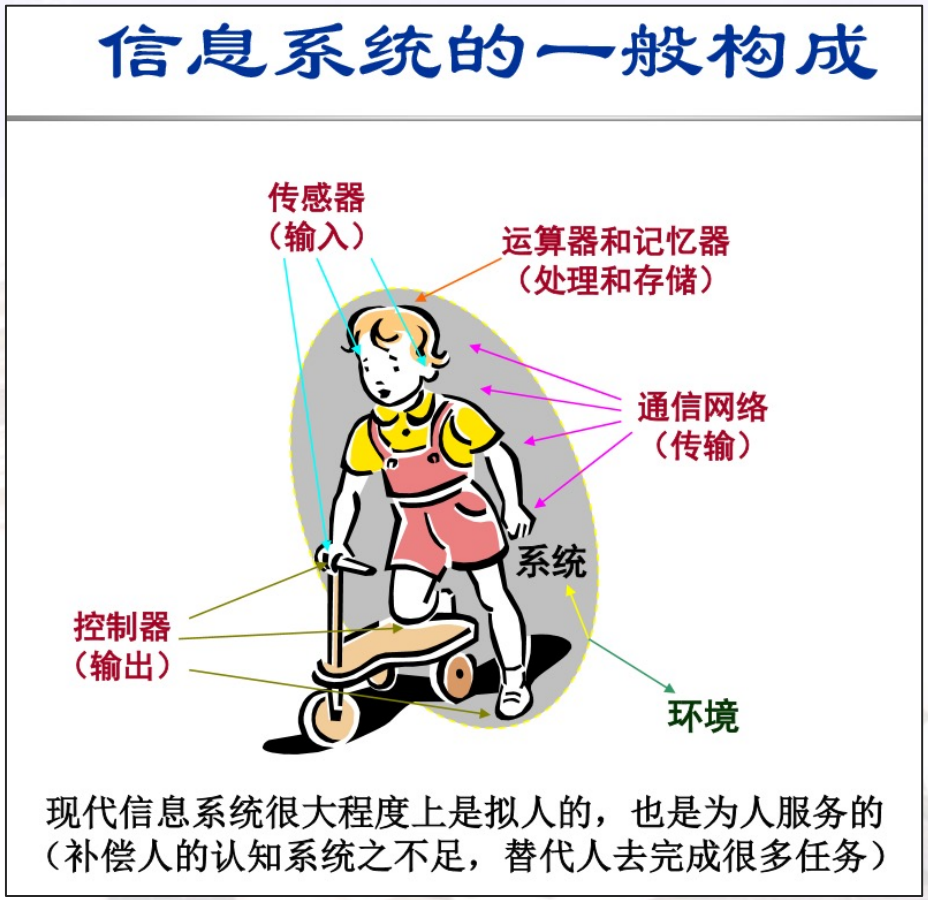
- 机器人系统及系统集成
- 无人机系统集成框架
- 有限状态机
- 初赛环境使用

目录

1 机器人系统及组成

机器人并没有严格统一的定义，但作为一种信息处理系统，一般包含以下四个部分

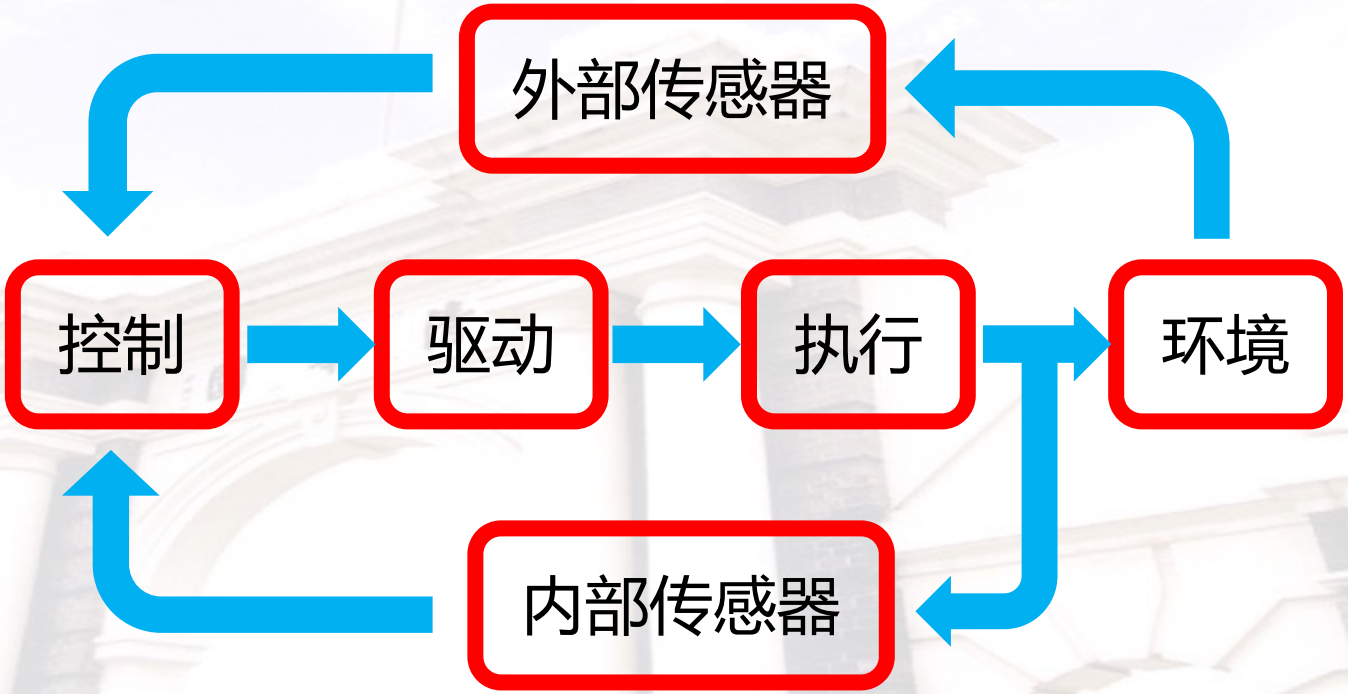
- **感知系统**：由内部传感器和外部传感器组成，感知自身状态和环境信息，是机器人与环境交互的窗口，相当于人的五觉；
- **控制系统**：具有运算、存储等功能，处理来自感知系统的信息，并协调各执行器的工作，相当于人的大脑；
- **机械系统**：执行器，是机器人作用于环境的执行体，相当于人的四肢；
- **驱动系统**：驱动机械系统的装置，相当于人的肌肉；



图源：清华大学电子工程系MAP课

1 系统集成

- **系统集成**：将各个分离的模块集成到相互关联、统一和协调的系统之中，通过各模块的协作与交互而实现具有特定功能的完整系统。
- **S-P-A结构**：机器人系统的结构组成，使其天然拥有sense-think-act的工作模式，自然而然的形成了“传感-计划-行动”(SPA)结构



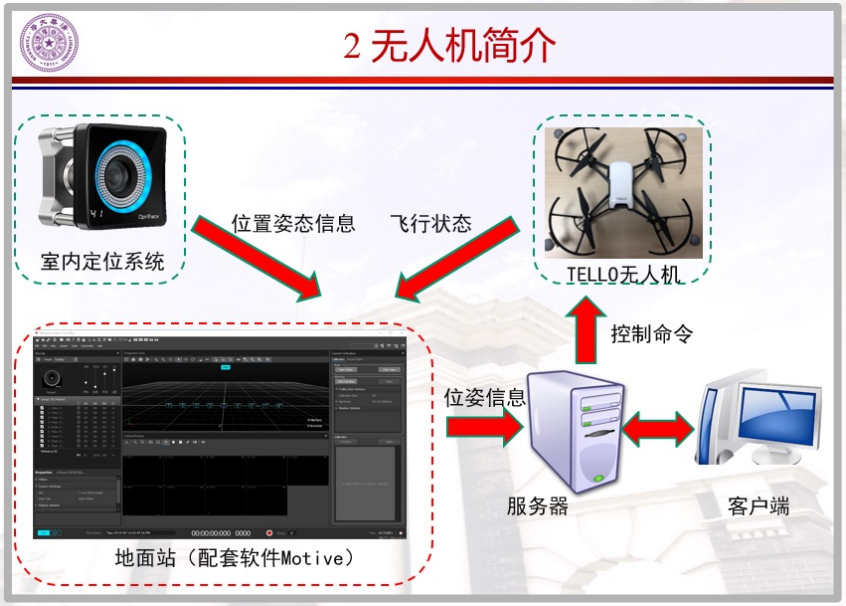
参考资料：https://en.wikipedia.org/wiki/Robotic_paradigm

目录

- 机器人系统及系统集成
- 无人机系统集成框架
- 有限状态机
- 初赛环境使用

2 无人机系统集成

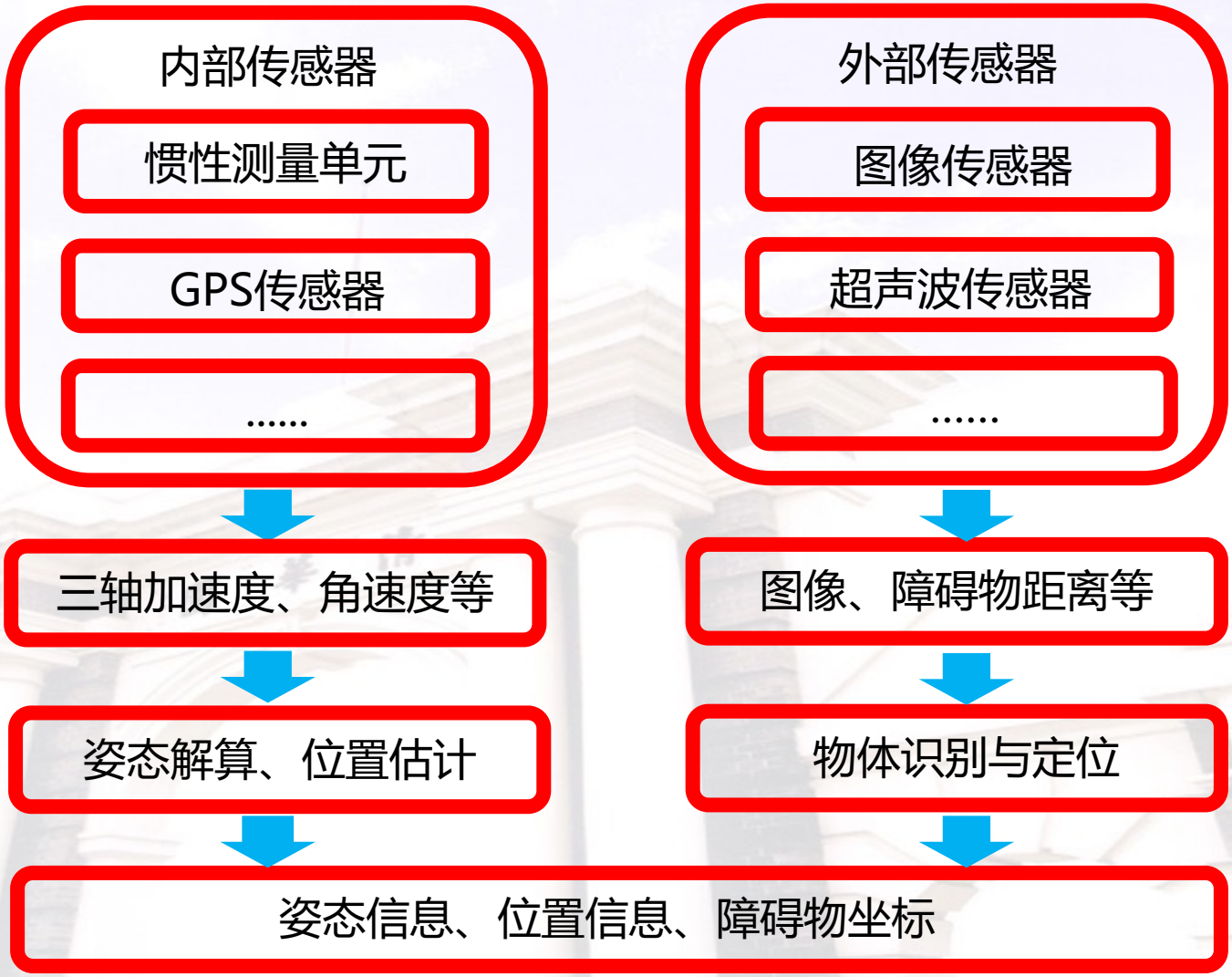
- **单个无人机**本身即是一个系统集成，需要各个模块的相互协作以完成飞行任务。无人机基本组成部分有：
  - 感知系统：惯性测量单元IMU、图传等
  - 控制系统：客户端、服务器、飞行控制器MCU
  - 驱动/机械系统：电机驱动控制、电动机、螺旋桨
  - 其他：电池



图源：http://www.crazepony.com/wiki/heli-quad-fix-copter.html

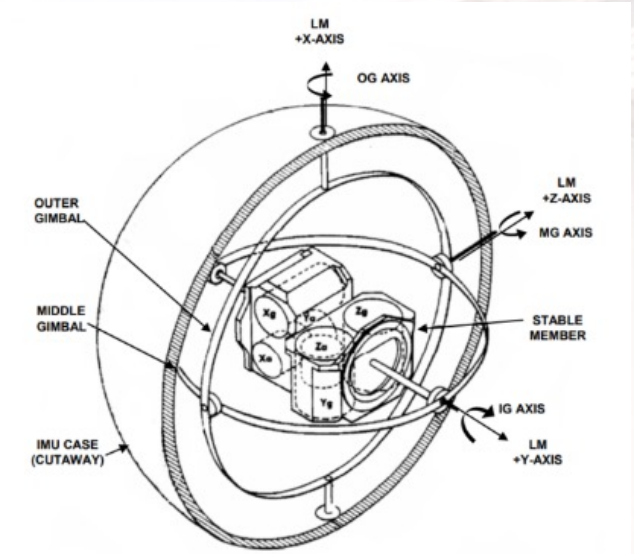
2 系统集成——感知阶段

- 无人机感知阶段



2 系统集成——感知阶段

- 数据采集
  - 各种各样的传感器，如工业机器人需要力觉传感器、环境监测系统中需要温湿度、气压等多种传感器等；
  - 超声波雷达、毫米波雷达、激光雷达等多种感知手段，LiDAR由于出色的精度和速度，一直是无人驾驶感知系统中的主角；
  - 新型无线通信技术如超宽带(UWB)技术具有更高的时间分辨率，在协同定位网络中就可以提高定位精度；



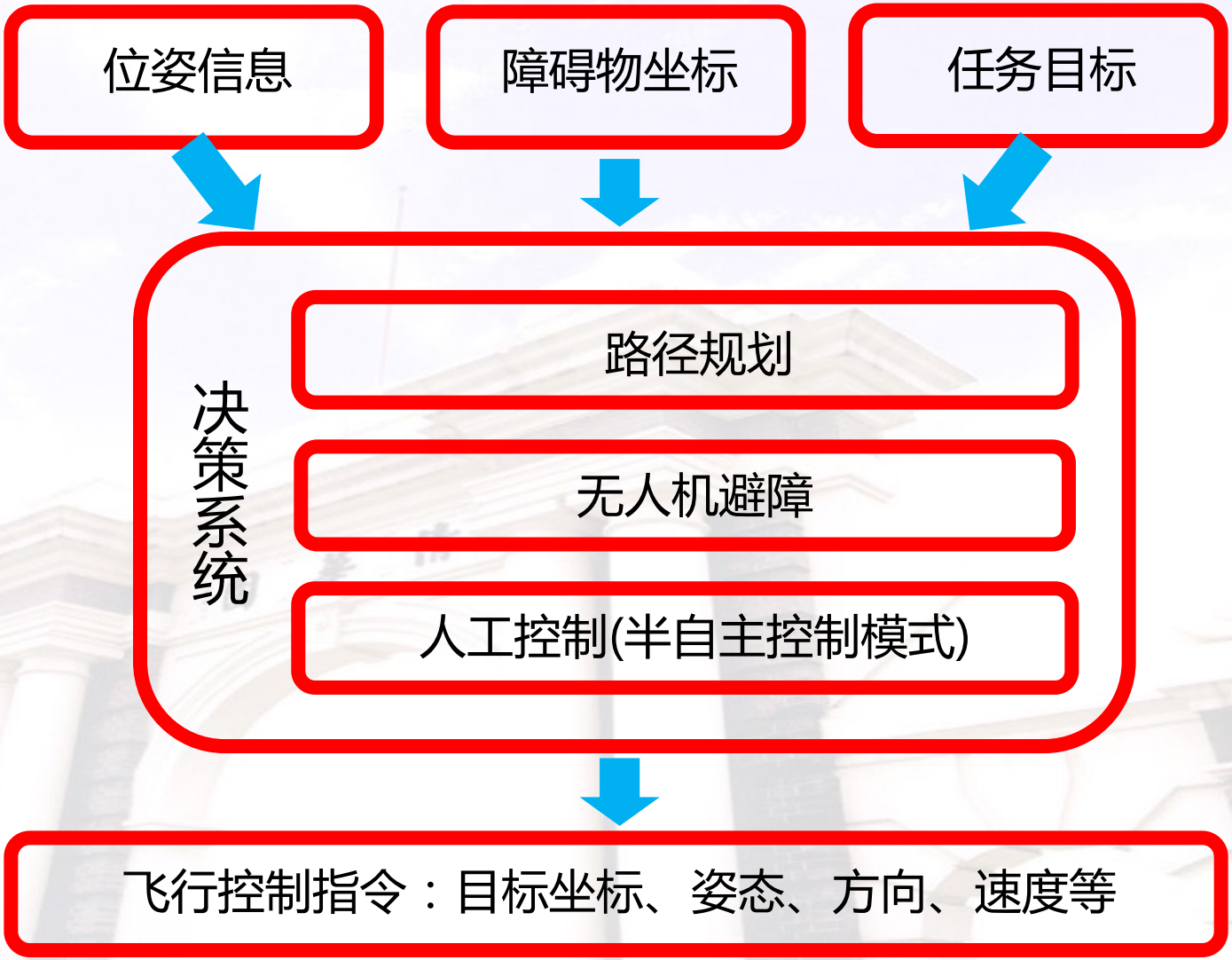
图源：https://en.wikipedia.org/wiki/Inertial_measurement_unit
http://www.cvlibs.net/datasets/kitti/index.php

2 系统集成——感知阶段

- 数据处理
  - **数据滤波**：对带噪观测数据进行处理从而降低噪声干扰，更好的提取数据特征，如卡尔曼滤波及其改进EKF、UKF等；
  - **参数估计**：根据抽取的随机样本数据估计总体分布的未知参数，如最大似然估计、最小二乘等；
  - **多源融合**：对通过不同手段获得的观测数据进行综合处理和分析，按照融合的层次还可分为
    - 数据级融合：直接对原始数据处理，信息损失量少，但有较大局限性，且计算量较大；
    - 模型级融合：处于中间层次，可以降低数据量和计算量，但也会导致信息损失；
    - 决策级融合：是最高层面的智能化融合，容错与抗干扰性强；

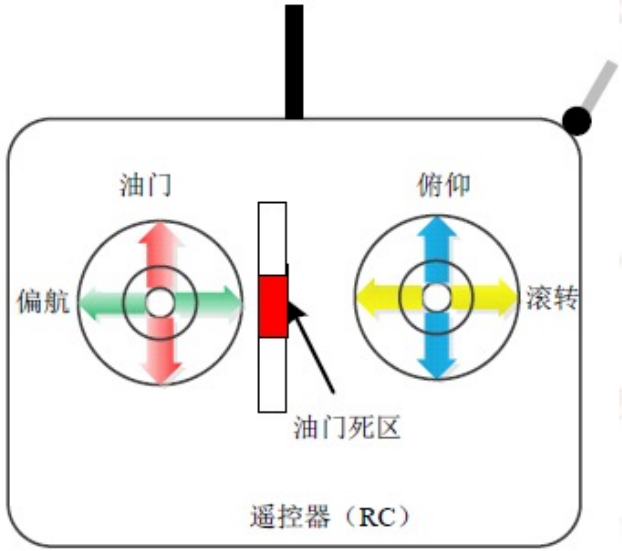
2 系统集成——计划阶段

- 无人机计划阶段



2 系统集成——计划阶段

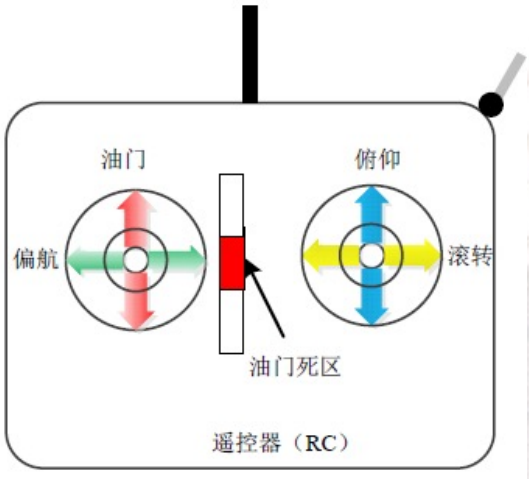
- 无人机决策机制
  - **全自主控制**：主要包括任务规划与路径规划，由操作人员离线完成，飞机起飞后无法人为在线干预；路径规划中一般还需要考虑避障功能，如人工势场法；
  - **半自主控制**：遥控模式（RC）下，飞控手可以通过遥控器人为在线干预无人机的飞行，飞控手释放摇杆时无人机自动进入自动模式（AC），如下图的油门死区；



参考资料：http://rfly.buaa.edu.cn/course.html（北航可靠飞行控制研究组）

2 系统集成——计划阶段

- 半自主控制
  - **自稳定模式**：自动模式下，多旋翼会自动保持自身水平，但是水平位置和高度均会漂移。
  - **定高模式**：自动模式下，无人机自动调整油门来保持当前的高度，但飞控手需要不断地调整遥控器的滚转/俯仰摇杆保持悬停。定高模式需要高度传感器的支持才能实现，例如气压计、超声波测距仪等。
  - **定点模式**：自动模式下，多旋翼能自动保持当前的水平位置、航向和高度。定点模态需要测高仪器和位置传感器的支持才能实现，例如摄像机和GPS等。



参考资料：http://rfly.buaa.edu.cn/course.html（北航可靠飞行控制研究组）

2 系统集成——计划阶段

- **系统鲁棒性**：无人机决策层除了完成正常规划任务，还要考虑各种可能的异常情况，如通信故障、动力系统异常、传感器失效等；可考虑增加健康评估和失效保护等功能；
- **健康评估**：根据监测数据检验传感器、动力系统、机械系统等是否有故障；
  - 气压计所获高度值出现较大范围地波动，导致多旋翼无法定高，则考虑气压计不健康的可能性。
  - 多旋翼出现自转现象，则需要考虑电子罗盘不健康的可能性。
  - 若多旋翼出现较大抖动，则需要考虑惯导系统不健康的可能性。

参考资料：http://rfly.buaa.edu.cn/course.html（北航可靠飞行控制研究组）

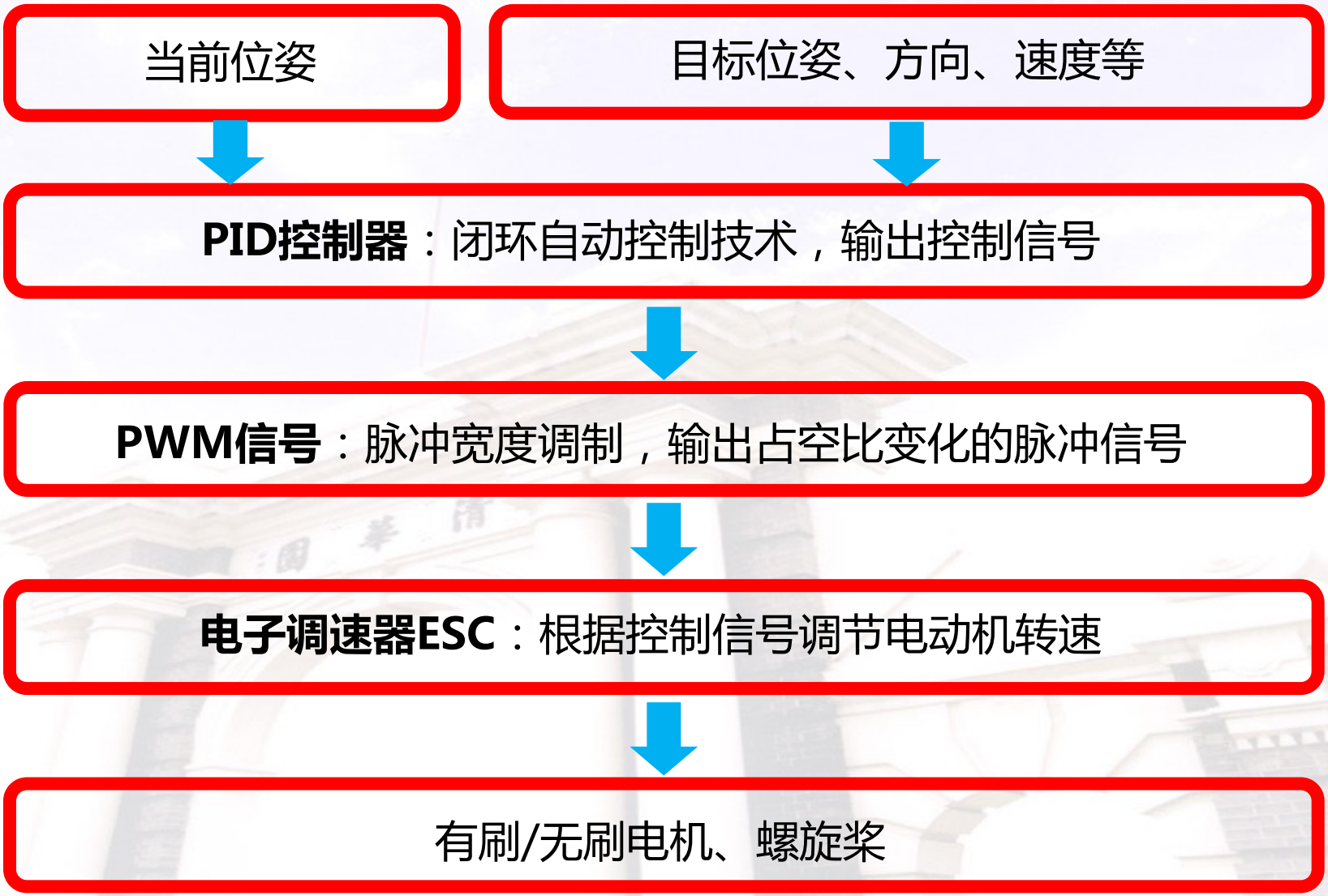
2 系统集成——计划阶段

- **系统鲁棒性**：无人机决策层除了完成正常规划任务，还要考虑各种可能的异常情况，如通信故障、动力系统异常、传感器失效等；可考虑增加健康评估和失效保护等功能；
- **失效保护**：系统故障时对关键模块进行保护；
  - 气压计失效保护：若多旋翼监测到气压计故障，则建议多旋翼保持油门不变，从定点模式降级为自稳定模式。
  - 电子罗盘失效保护：若多旋翼监测到电子罗盘故障，则建议多旋翼根据用户配置，从定点模式降级为定高模式。
  - GPS失效保护：若多旋翼监测到GPS存在问题，则建议多旋翼根据用户配置，从定点模式降级为定高模式。
  - 惯导系统失效保护：若多旋翼监测到惯导系统失效，则建议多旋翼以逐渐减少拉力的方式实现紧急着陆。

参考资料：http://rfly.buaa.edu.cn/course.html（北航可靠飞行控制研究组）

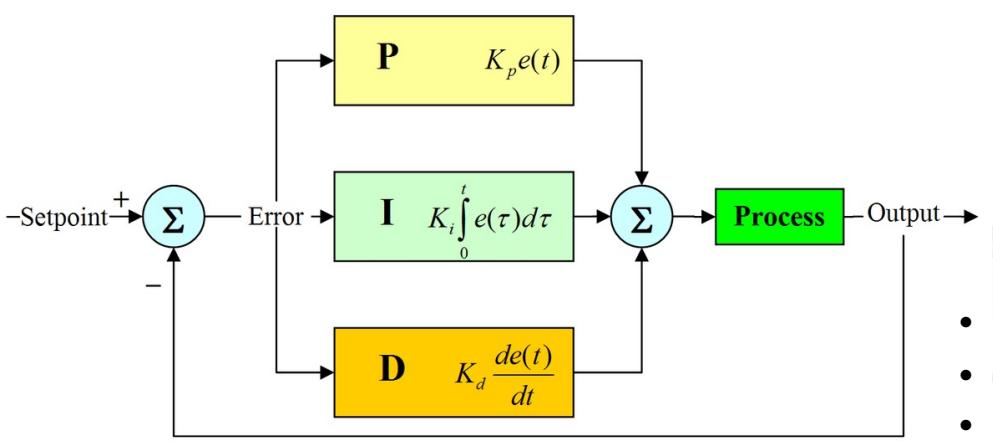
2 系统集成——执行阶段

- 无人机执行阶段



2 系统集成——执行阶段

- PID控制器
  - 比例-积分-微分控制器，是工业控制中常用的反馈回路部件；
  - **比例控制**：即其控制器的输出与输入误差信号成比例关系，是PID的控制基础；
  - **积分控制**：控制器的输出与输入误差信号的积分成正比关系，可消除稳态误差，但可能增加超调；
  - **微分控制**：控制器的输出与输入误差信号的微分（即误差的变化率）成正比关系，可加快大惯性系统响应速度以及减弱超调趋势；

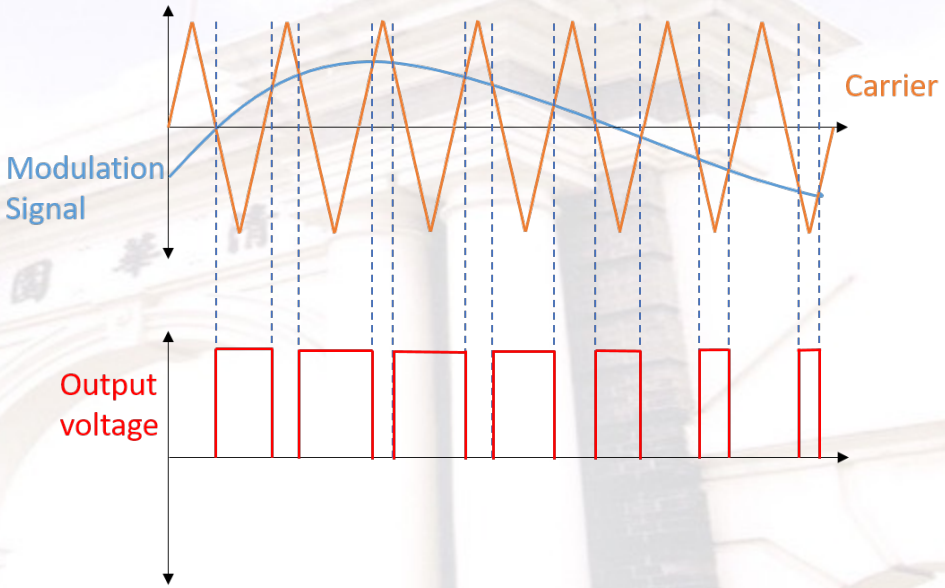


- P：当前误差
- I：过去误差的积累
- D：误差的微分

图源： http://www.360doc.com/content/19/0531/17/44130189_839466938.shtml

2 系统集成——执行阶段

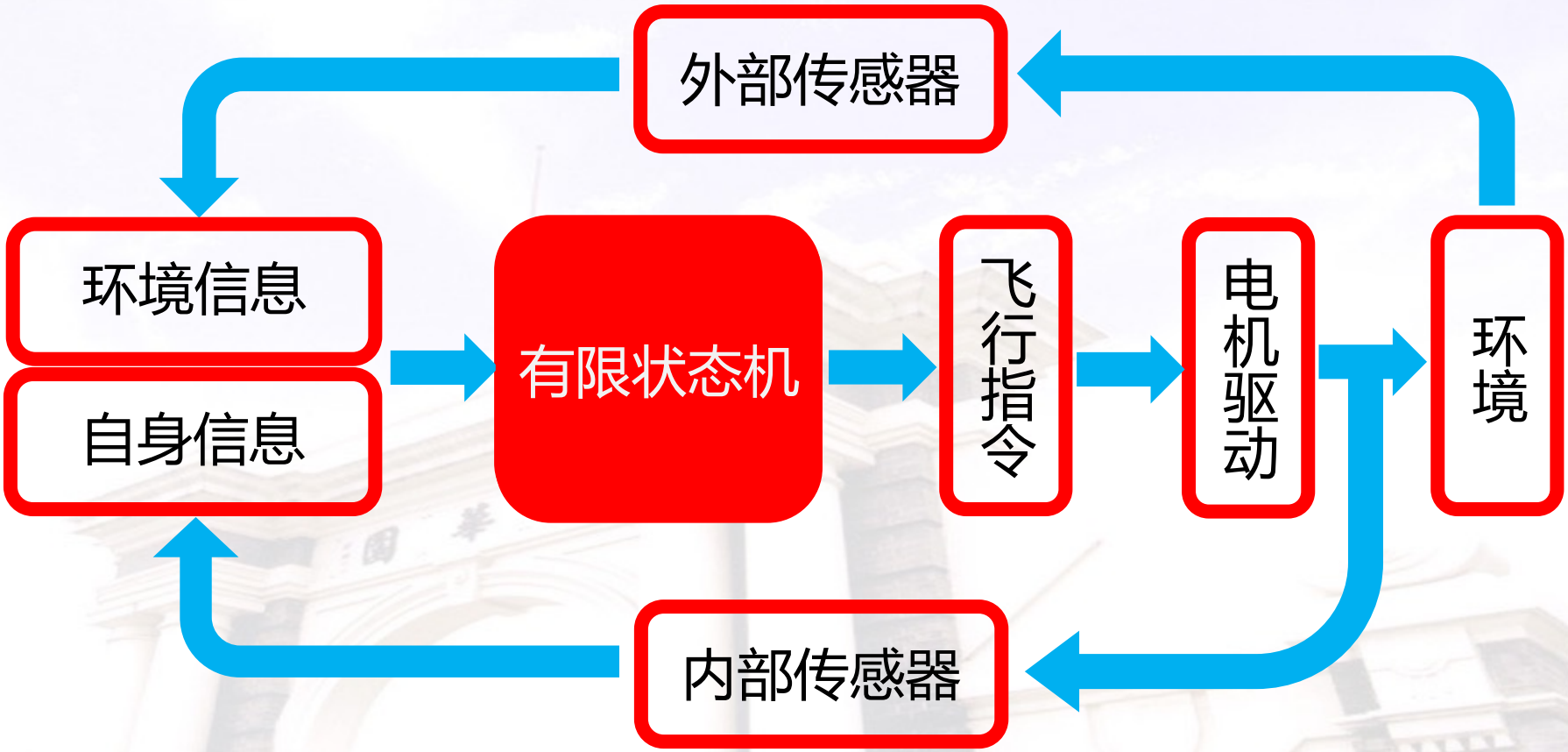
- PWM信号
  - 脉冲宽度调制，对电路开关器件进行控制，从而改变脉冲的宽度或占空比，进而实现通过数字输出控制模拟电路；
  - 其优点是从处理器到被控系统信号都是数字形式的，无需进行数模转换，相比于模拟控制电路，其抗噪声能力强；



图源： https://openlabpro.com/guide/pulse-width-modulation/

2 无人机系统集成框架

- 无人机系统集成



目录

- 机器人系统及系统集成
- 无人机系统集成框架
- 有限状态机
- 初赛环境使用

3 系统组织工具——状态机

- 状态机理论最初的发展在数字电路设计领域。
- 【在电路系统中定义】状态机由状态寄存器和组合逻辑电路构成，能够根据控制信号按照预先设定的状态进行状态转移，是协调相关信号动作,完成特定操作的控制中心。在数字电路方面，根据输出是否与输入信号有关，状态机可以划分为Mealy型和Moore型状态机。 Moore型状态机的输出只和当前状态有关，和输入无关。 ——《数字逻辑与处理器基础》。
- 【将状态机概念应用到软件设计】用来描述一些复杂的算法，表明一些算法的内部的结构和流程，更多的关注于程序对象的执行顺序。
  - 状态寄存器 -> 系统运行状态
  - 组合逻辑电路 -> 在对应状态下的处理程序
  - 状态转移 -> 根据观察和反馈修改状态

3 电路中有限状态机

- 静态顺序结构

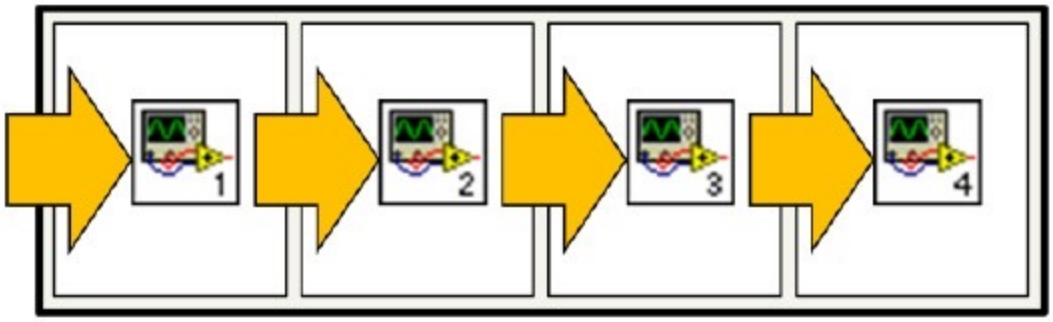


图 1　顺序结构模式

- 动态结构

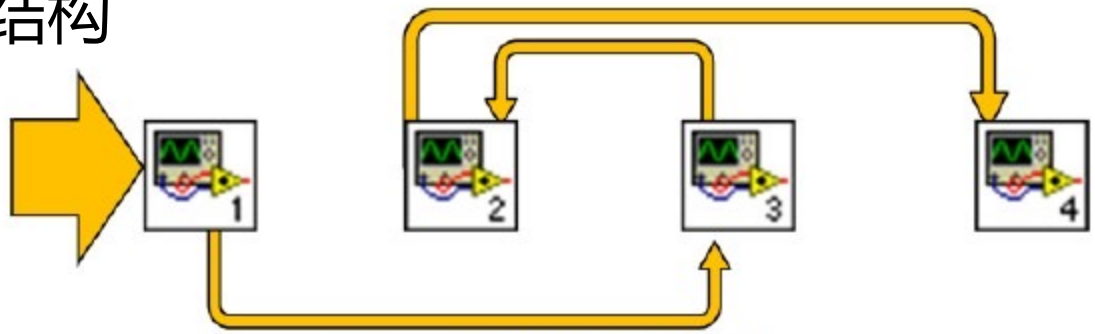


图 2　动态的程序结构

3 有限状态机

- 有限自动机（Finite Automata Machine）是计算机科学的重要基石，它在软件开发领域内通常被称作有限状态机（Finite State Machine），是一种应用非常广泛的软件设计模式。
- 有限状态机的作用主要是描述对象在它的生命周期内所经历的状态序列，以及如何响应来自外界的各种事件。
- 在现实中，有许多事情可以用有限个状态来表达，如:红绿灯、电话机等等。 其实，在资讯领域中，很多事情都是由有限的状态所组成，再由于不同的输入而衍生出各个状态。

3 有限状态机

- 有限状态机FSM思想广泛应用于硬件控制电路设计，也是软件上常用的一种处理方法。它把复杂的控制逻辑分解成有限个稳定状态，在每个状态上判断事件，变连续处理为离散数字处理，符合计算机的工作特点。
- 同时，因为有限状态机具有有限个状态，所以可以在实际的工程上实现。但这并不意味着其只能进行有限次的处理，相反，有限状态机是闭环系统，有限无穷，可以用有限的状态，处理无穷的事务。

3 基本概念

- 在描述有限状态机时，常会碰到的几个基本概念：
  - 状态（State） 指的是对象在其生命周期中的一种状况，处于某个特定状态中的对象必然会满足某些条件、执行某些动作或者是等待某些事件。
  - 事件（Event） 指的是在时间和空间上占有一定位置，并且对状态机来讲是有意义的那些事情。事件通常会引起状态的变迁，促使状态机从一种状态切换到另一种状态。
  - 转换（Transition） 指的是两个状态之间的一种关系，表明对象将在第一个状态中执行一定的动作，并将在某个事件发生同时某个特定条件满足时进入第二个状态。
  - 动作（Action） 指的是状态机中可以执行的那些原子操作，所谓原子操作指的是它们在运行的过程中不能被其他消息所中断，必须一直执行下去。

3 有限状态机—例1

- 红绿灯
  - 红绿灯运作的原理相当简单，从一开始绿灯，经过一段时间后，将变为黄灯， 再隔一会儿，就会变成红灯，如此不断反覆。 其FSM如下。

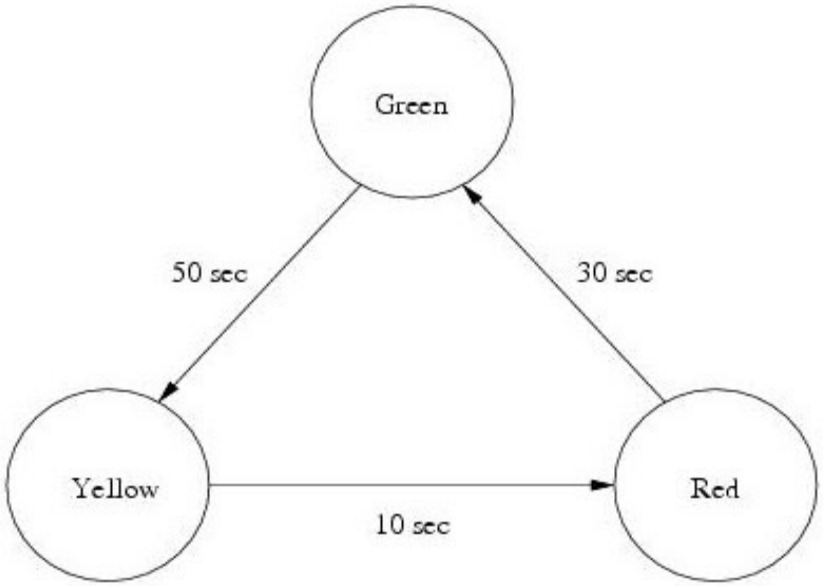


Figure 1: traffic light

3 有限状态机—例2

- 自动贩售机
  - 假设有简单的一自动贩卖机贩售两类商品，一类售价20元，另一类售价50元。 如果该贩卖机只能辨识10元及50元硬币。 一开始机器处于Hello的状态，当投入10元时，机器会进入余额不足的状态，直到投入的金额大于20元为止。 如果一次投入50元，则可以选择所有的产品，否则就只能选择20元的产品。 完成选择后，将会卖出商品并且找回剩余的零钱，随后，机器又将返回初始的状态。

3 有限状态机—例2

- 自动贩售机
  - 状态转移从初始状态Hello开始：
  - 状态允许买50元东西的状态可以合并，也可以分立

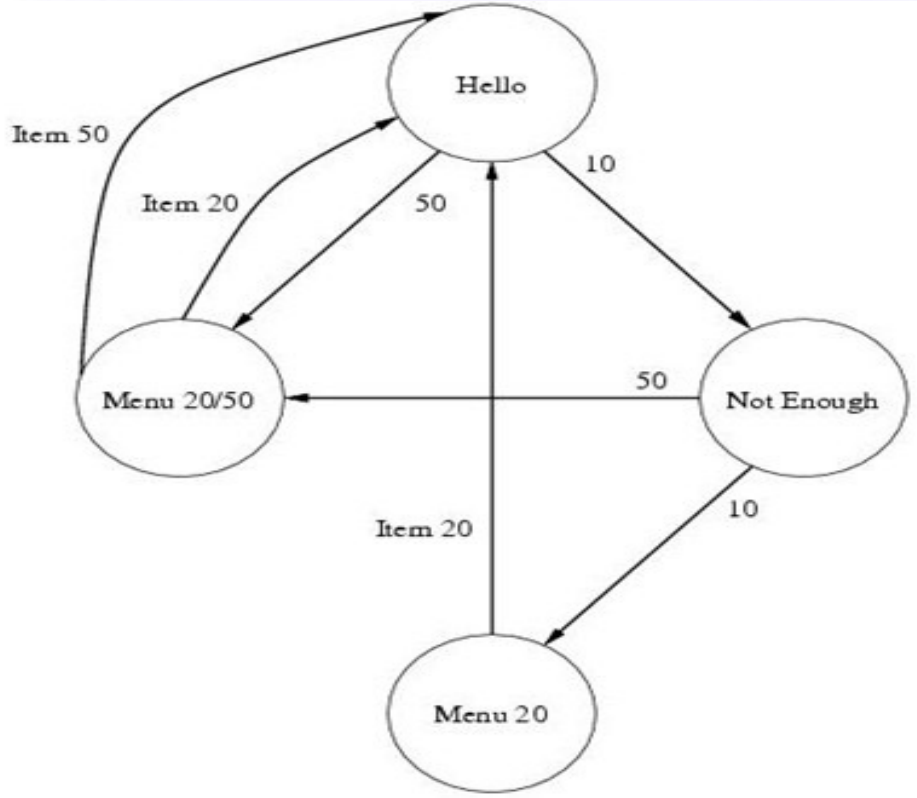


Figure 2: sales machine

3 有限状态机模型在系统中应用

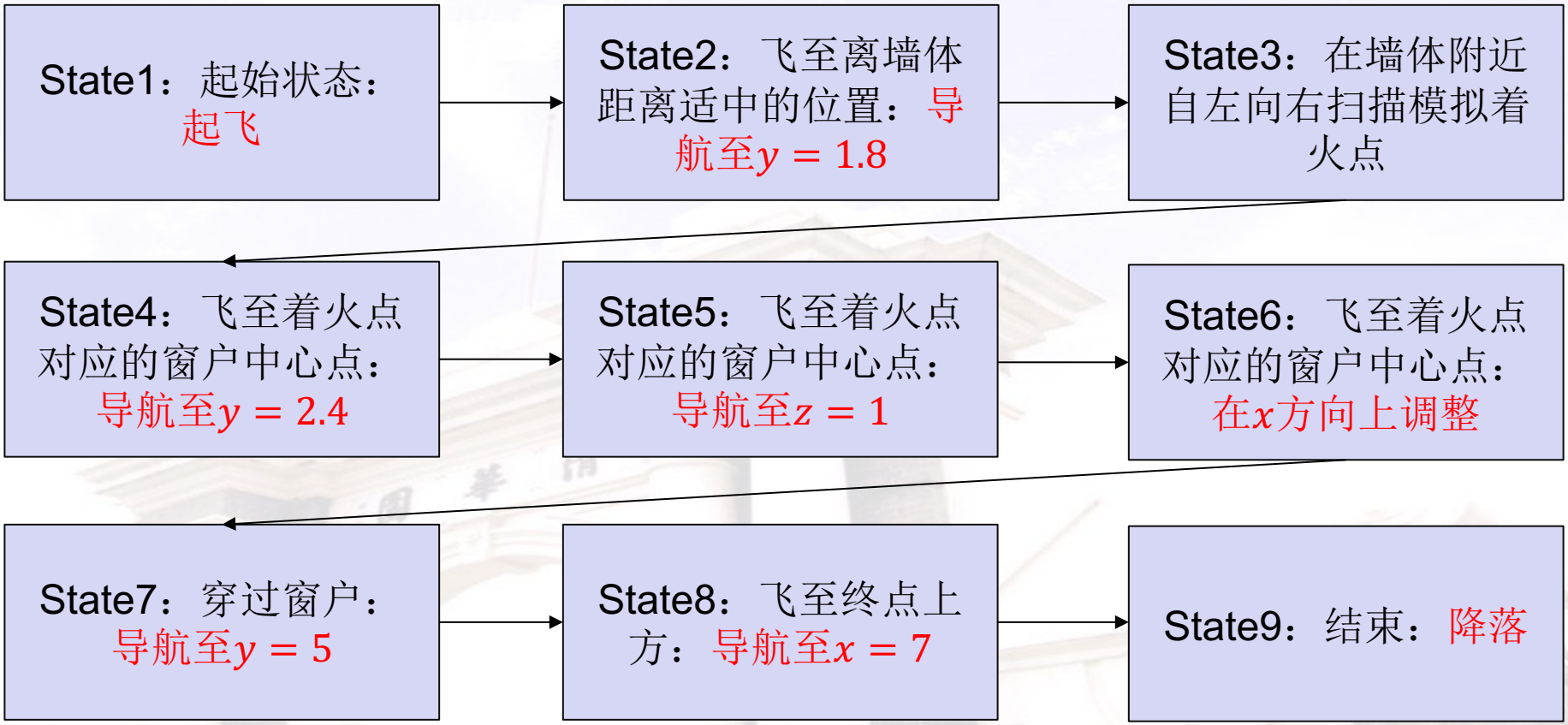
- 基本出发点：认为整体系统主要是由响应多个“事件”的相对简单的处理过程组成。
- 开发步骤：
  - 梳理状态转移图
  - 实现各状态处理方法
  - 判断条件完成状态转移
- 优点：简单明了，比较精确。对许多复杂的协议，事件数和状态数虽然增加，但是只要搞清楚了状态转移，对于每个状态下的处理都很简单。

3 有限状态机模型在系统中应用

- 以仿真例程中windows.launch中的任务为例
- 任务：无人机识别墙体上的模拟着火点，穿过其下方对应的窗户，最终在指定位置降落
- 思路：
  - 起飞后飞至与墙体距离适中的位置，以便后续识别
  - 从左向右扫描模拟着火点可能出现的位置
  - 发现模拟着火点后，导航至窗户中心点位置，然后穿过窗户并到达终点附近，最终降落

3 有限状态机模型在系统中应用

- 梳理状态转移图



- 缺点：状态数过多，FSM复杂，每导航一步都要引入新状态

3 有限状态机模型在系统中应用

- FSM状态的简化：可以将多段导航的导航信息用队列存储
- 进一步地，可以将所有的导航操作合并为一个状态


```
class FlightState(Enum):
    WAITING = 1
    NAVIGATING = 2
    DETECTING_TARGET = 3
    LANDING = 4
```

3 有限状态机模型在系统中应用

- 每次进入NAVIGATING状态前，初始化导航信息队列，说明导航状态之后的状态，然后调用switchNavigatingState函数切换状态。


```
self.navigating_queue_ = deque([['y', 1.8]])
self.next_state_ = self.FlightState.DETECTING_TARGET
self.switchNavigatingState()
```

```
# 在向目标点导航过程中，更新导航状态和信息
def switchNavigatingState(self):
    if len(self.navigating_queue_) == 0:
        self.flight_state_ = self.next_state_
    else: # 从队列头部取出无人机下一次导航的状态信息
        next_nav = self.navigating_queue_.popleft()
        # TODO 3: 更新导航信息和飞行状态
        self.navigating_dimension_ = next_nav[0]
        self.navigating_destination_ = next_nav[1]
        self.flight_state_ = self.FlightState.NAVIGATING
        # end of TODO 3
```

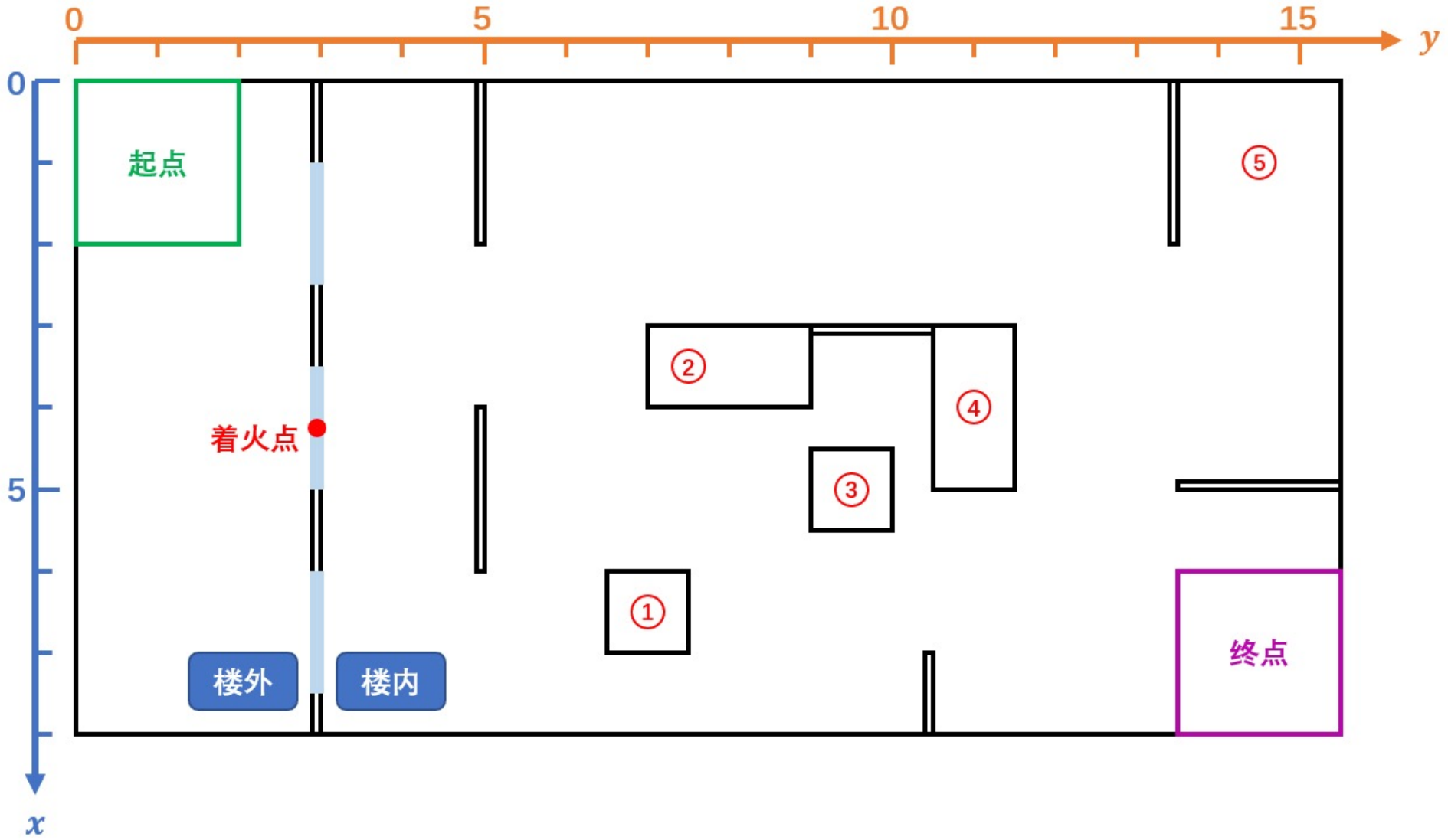
目录

4 初赛任务回顾

- 流程回顾
  - 起飞→检测着火点→穿过窗户→楼内区域导航→检测识别目标→指定位置降落
- 考察技能
  - ROS框架基础
  - 基于OpenCV的计算机视觉基本方法
  - 无人机闭环控制
- 已经提供了基础程序框架controller.py

4 赛场环境介绍

- 赛场：8×15.5矩形



4 初赛仿真环境开发

- 裁判机：judge.py
- 修改launch文件
- ROS调试方法
  - rostopic echo
  - rqt_graph & rviz
  - 直接在程序中打印要调试的信息

谢谢！