智能无人机技术设计实践

-- 系统集成与有限状态机

余金城

联系方式: yujincheng@yujincheng.me

时间：2019.11.2

目录

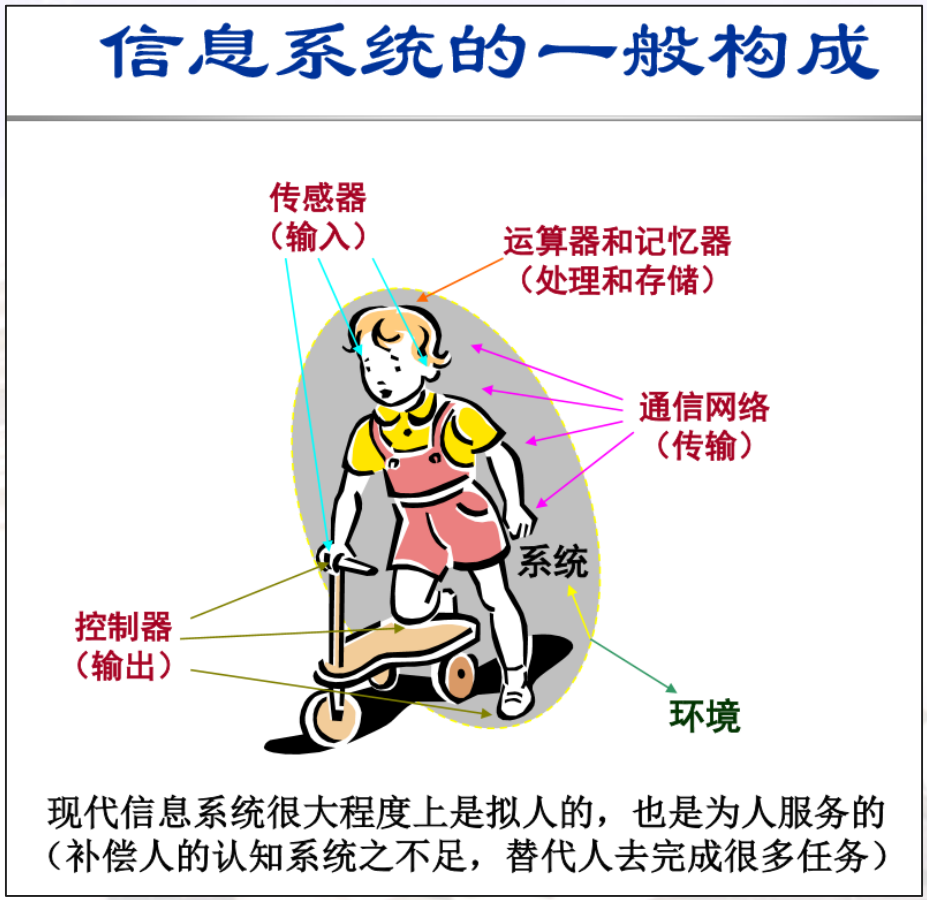
- 机器人系统及系统集成
- 无人机系统集成框架
- 有限状态机
- 初赛任务
- 决赛任务

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

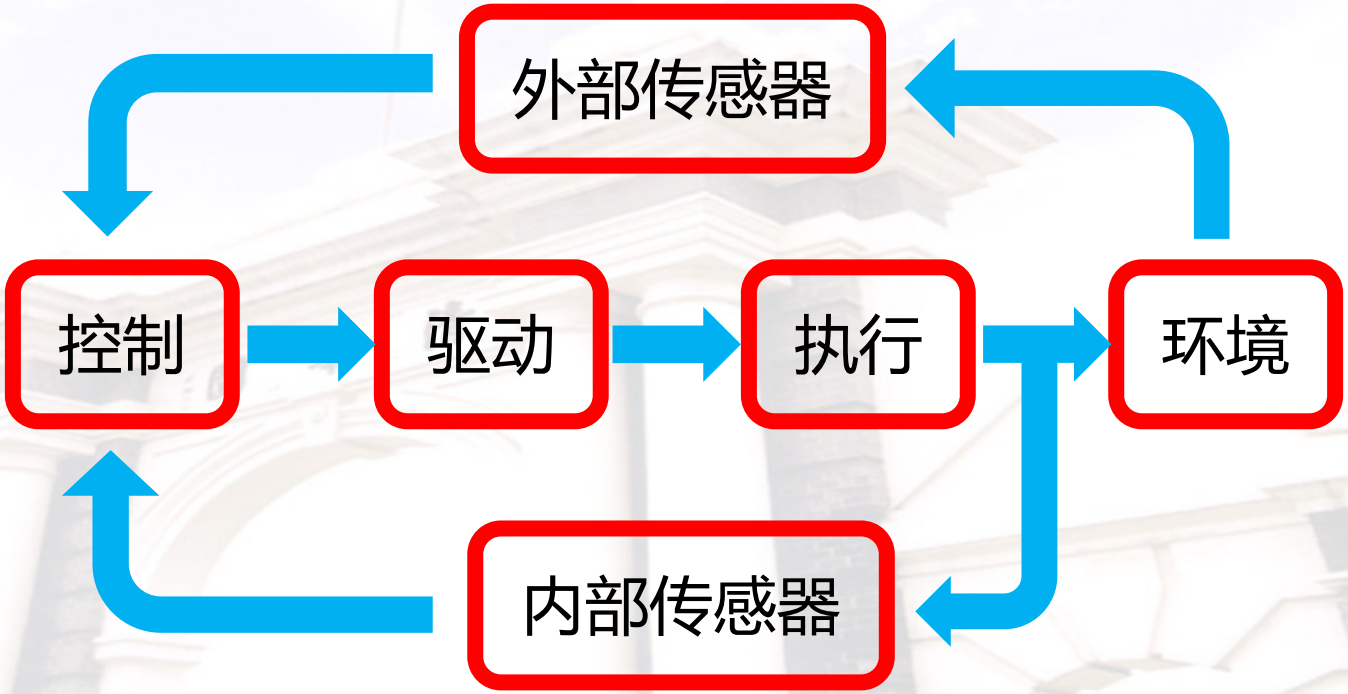
- **感知系统**：由内部传感器和外部传感器组成，感知自身状态和环境信息，是机器人与环境交互的窗口，相当于人的五觉；
- **控制系统**：具有运算、存储等功能，处理来自感知系统的信息，并协调各执行器的工作，相当于人的大脑；
- **机械系统**：执行器，是机器人作用于环境的执行体，相当于人的四肢；
- **驱动系统**：驱动机械系统的装置，相当于人的肌肉；



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

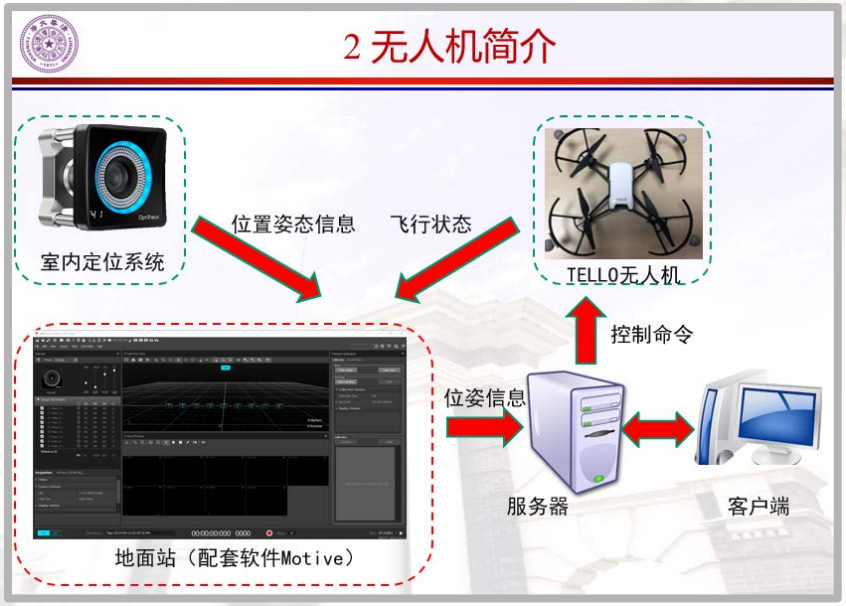


参考资料：https://en.wikipedia.org/wiki/Robotic_paradigm

目录

2 无人机系统集成

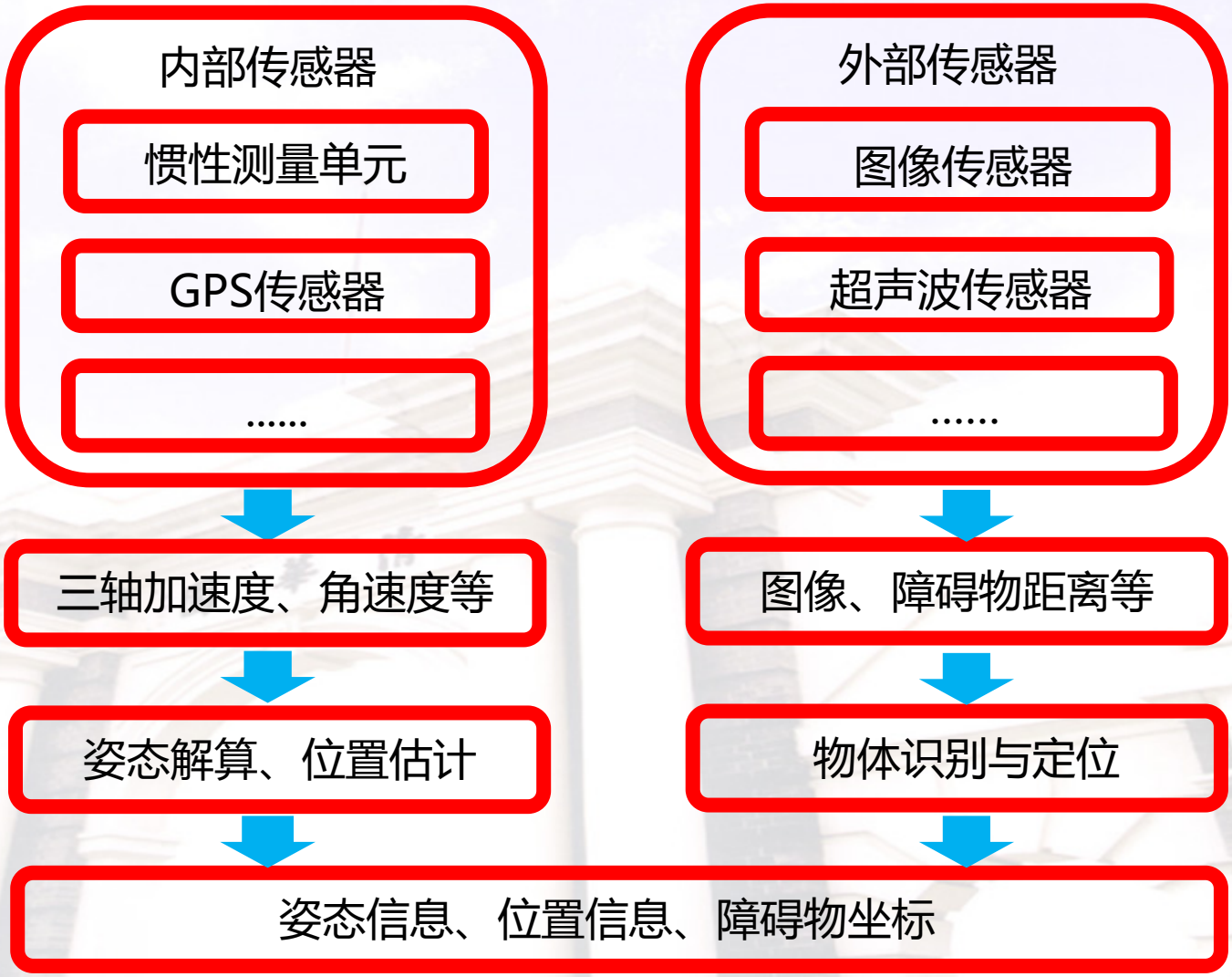
- **单个无人机**本身即是一个系统集成，需要各个模块的相互协作以完成飞行任务。无人机基本组成部分有：
  - 感知系统：惯性测量单元IMU、图传等
  - 控制系统：客户端、服务器、飞行控制器MCU
  - 驱动/机械系统：电机驱动控制、电动机、螺旋桨
  - 其他：电池



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

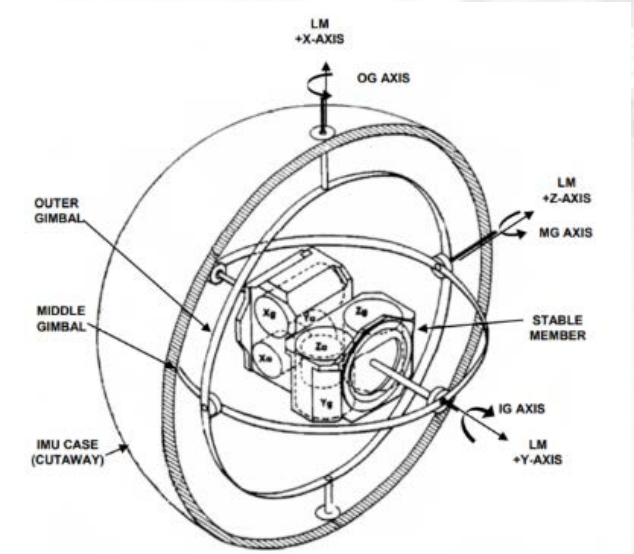
2 系统集成——感知阶段

- 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；



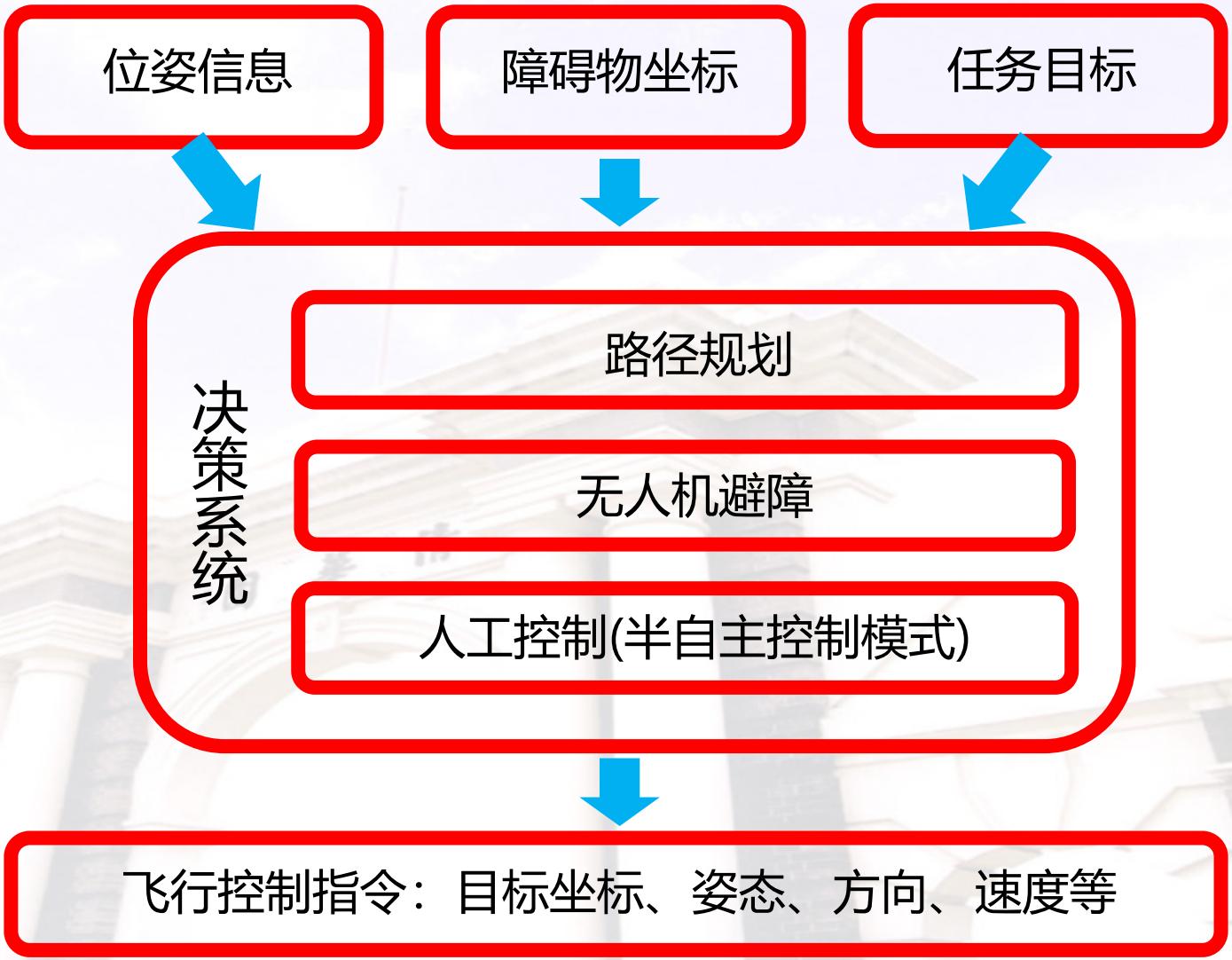
图源：https://en.wikipedia.org/wiki/Inertial_measurement_unit
http://www.cvlibs.net/datasets/kitti/index.php

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；

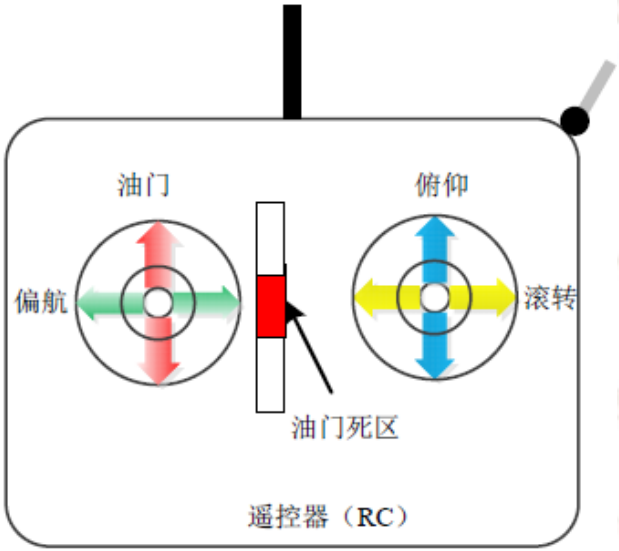
2 系统集成——计划阶段

- 无人机计划阶段



2 系统集成——计划阶段

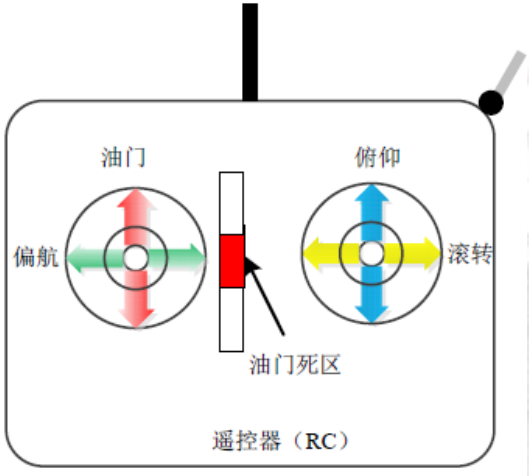
- 无人机决策机制
  - **全自主控制**：主要包括任务规划与路径规划，由操作人员离线完成，飞机起飞后无法人为在线干预；路径规划中一般还需要考虑避障功能，如人工势场法；
  - **半自主控制**：遥控模式（RC）下，飞控手可以通过遥控器人为在线干预无人机的飞行，飞控手释放摇杆时无人机自动进入自动模式（AC），如下图的油门死区；



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）

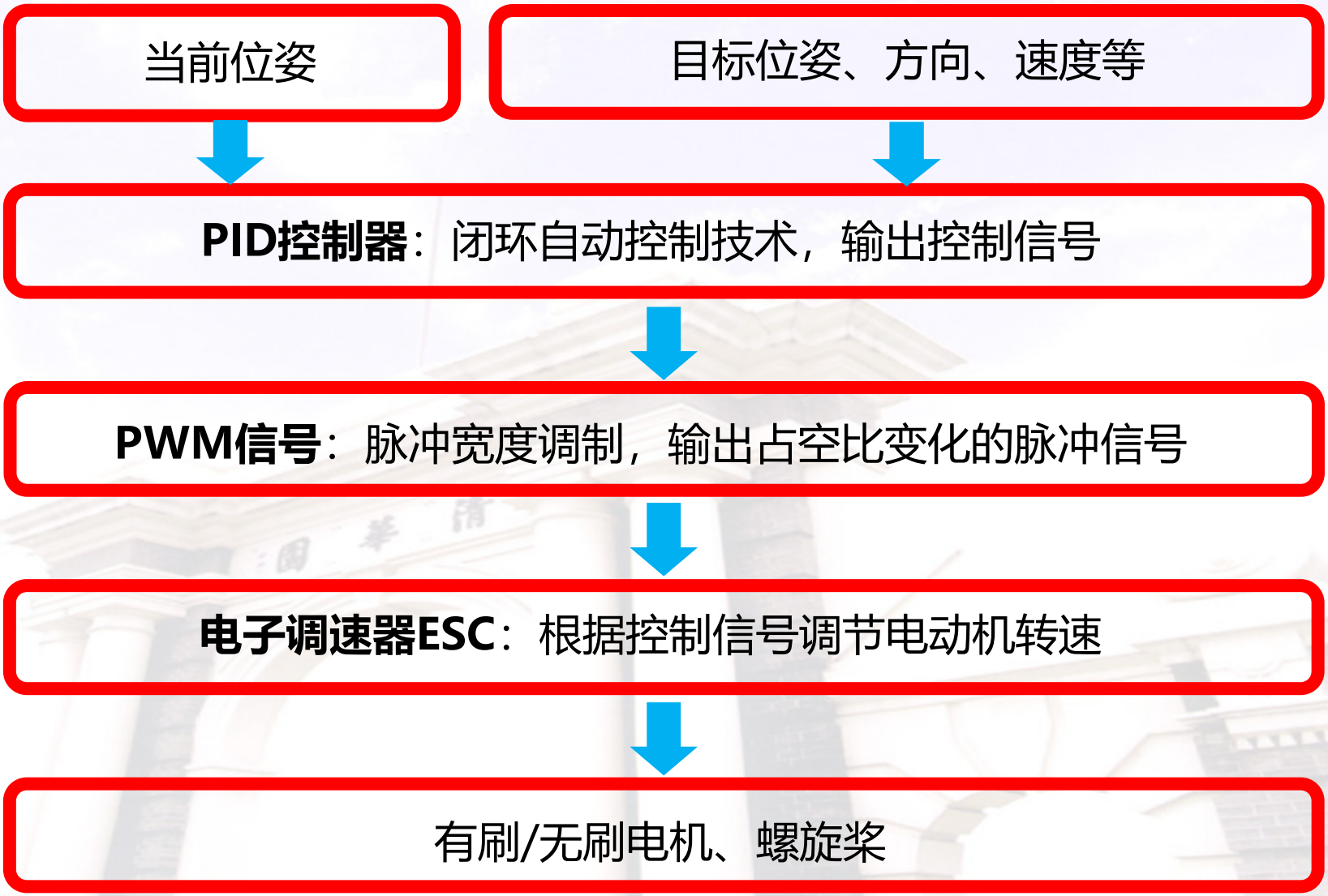
2 系统集成——计划阶段

- **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）

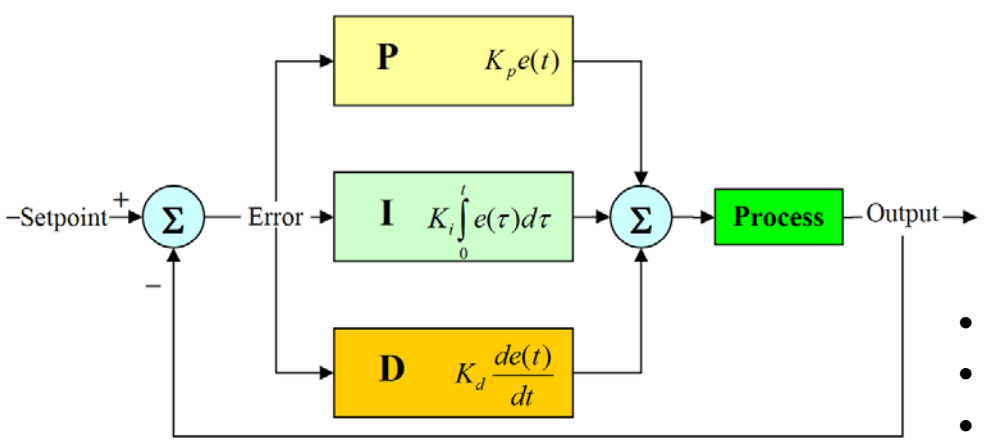
2 系统集成——执行阶段

- 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；

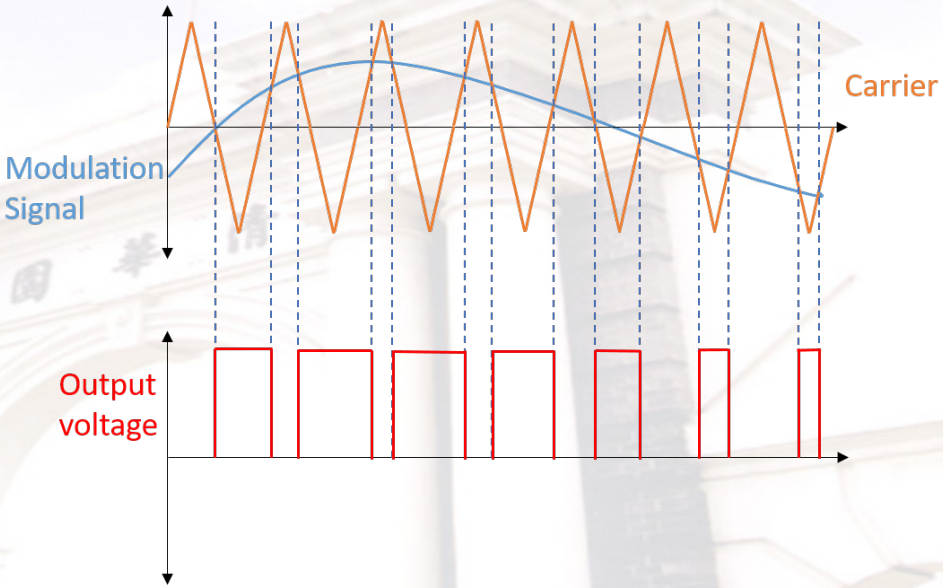


- P：当前误差
- I：过去误差的积累
- D：误差的微分

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

2 系统集成——执行阶段

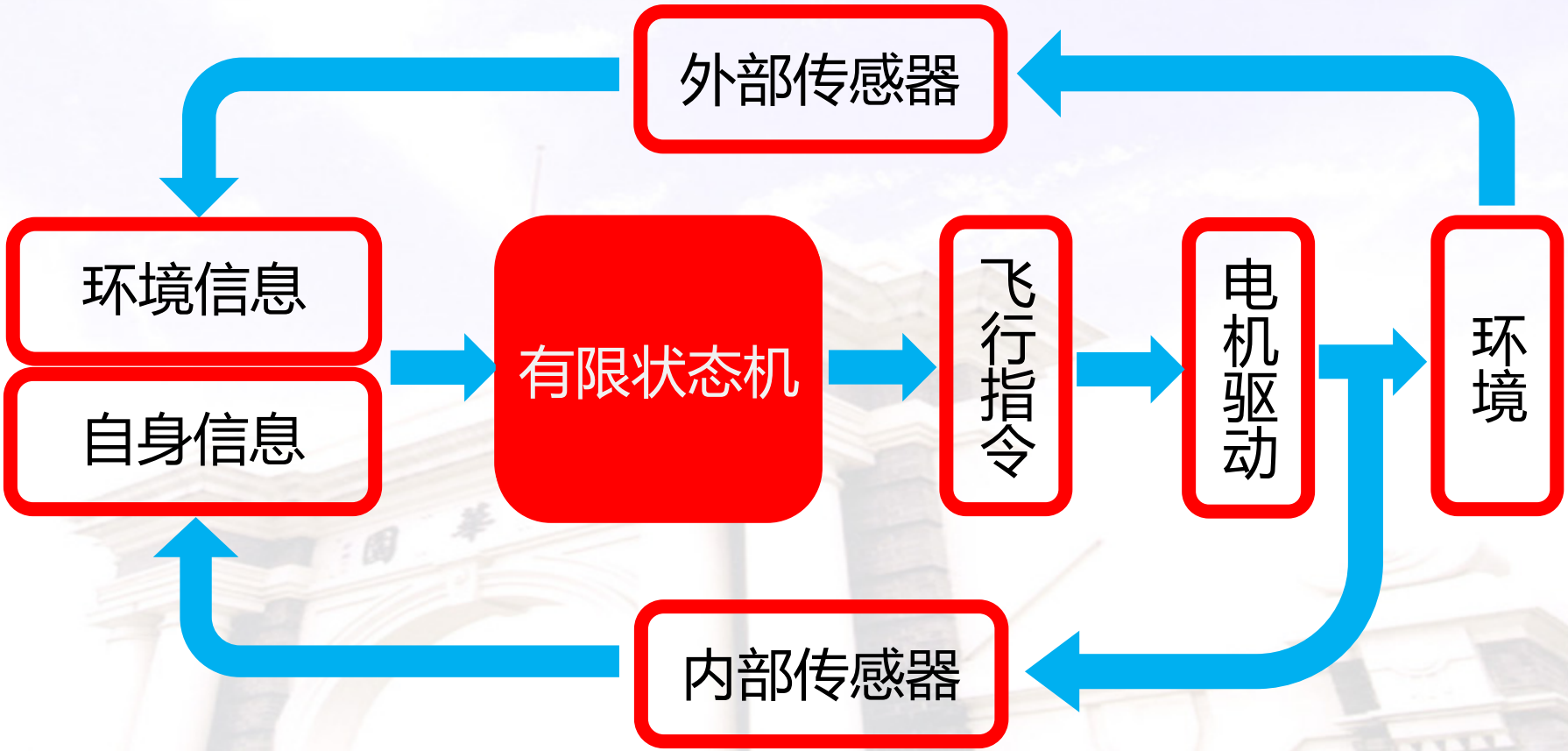
- PWM信号
  - 脉冲宽度调制，对电路开关器件进行控制，从而改变脉冲的宽度或占空比，进而实现通过数字输出控制模拟电路；
  - 其优点是从处理器到被控系统信号都是数字形式的，无需进行数模转换，相比于模拟控制电路，其抗噪声能力强；



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
- 初赛任务
- 决赛任务

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。 ——《数字电路与处理器基础》。
- 【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

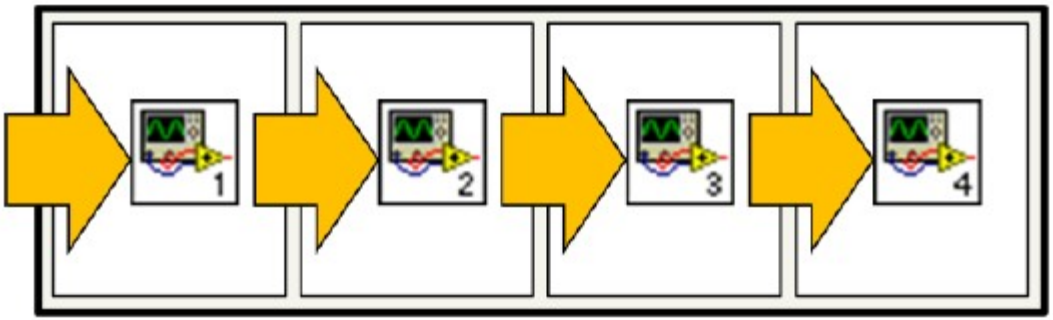


图 1　顺序结构模式

- 动态结构

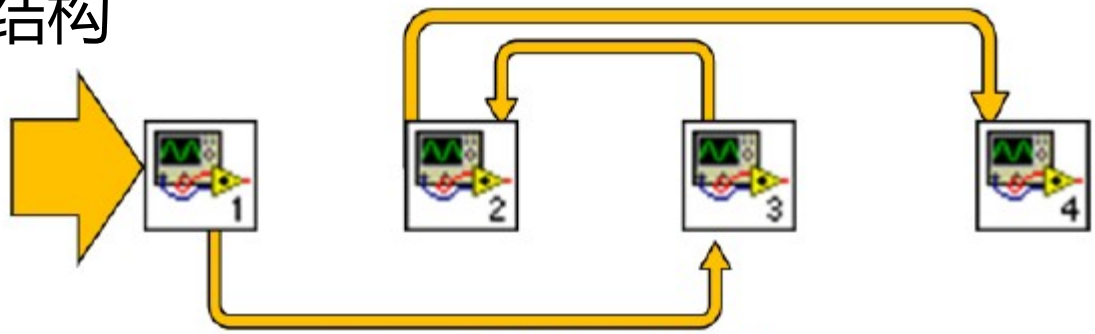


图 2　动态的程序结构

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。

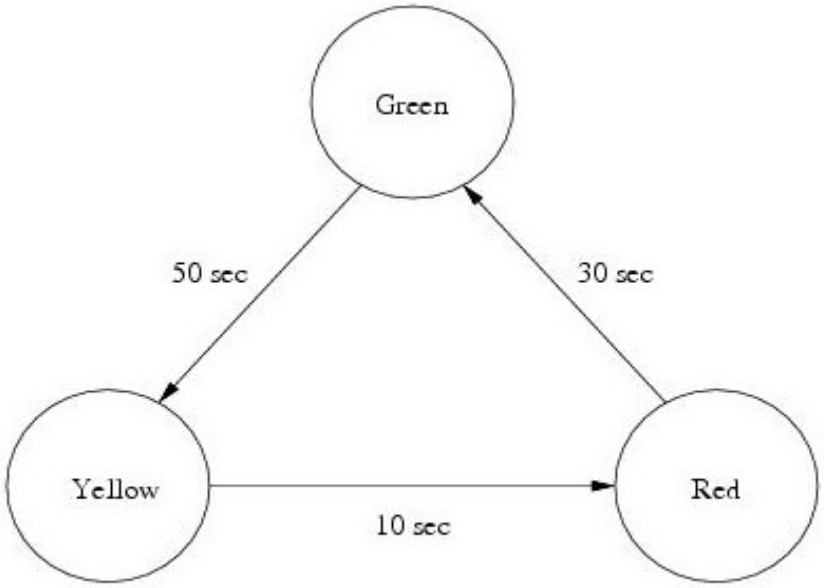


Figure 1: traffic light

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。 (五分钟思考下状态转移)

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

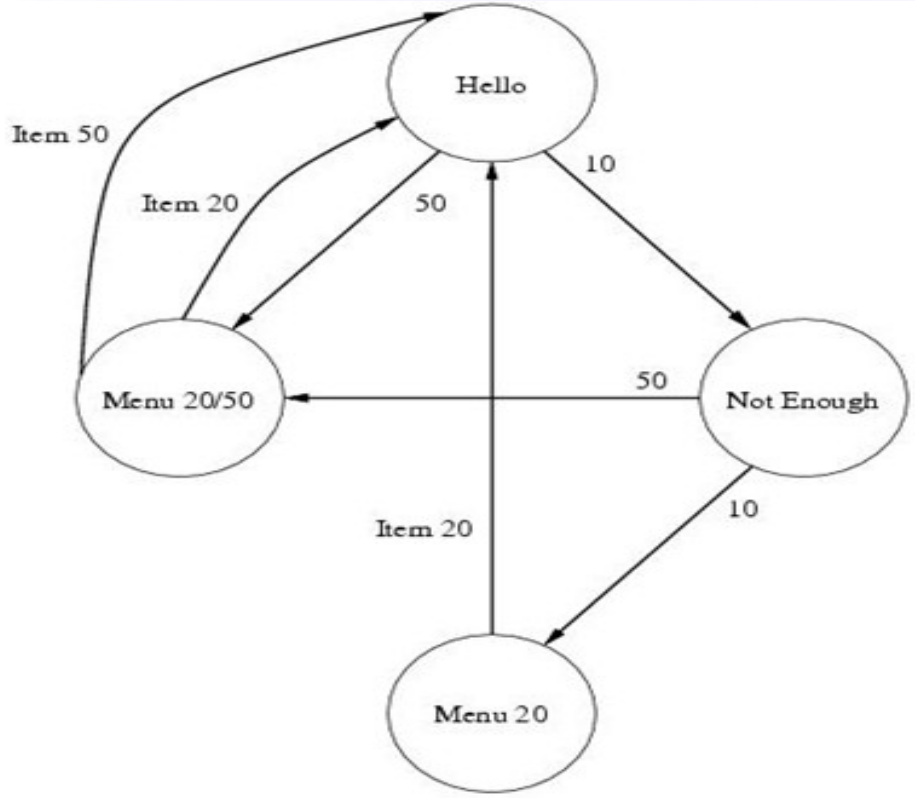


Figure 2: sales machine

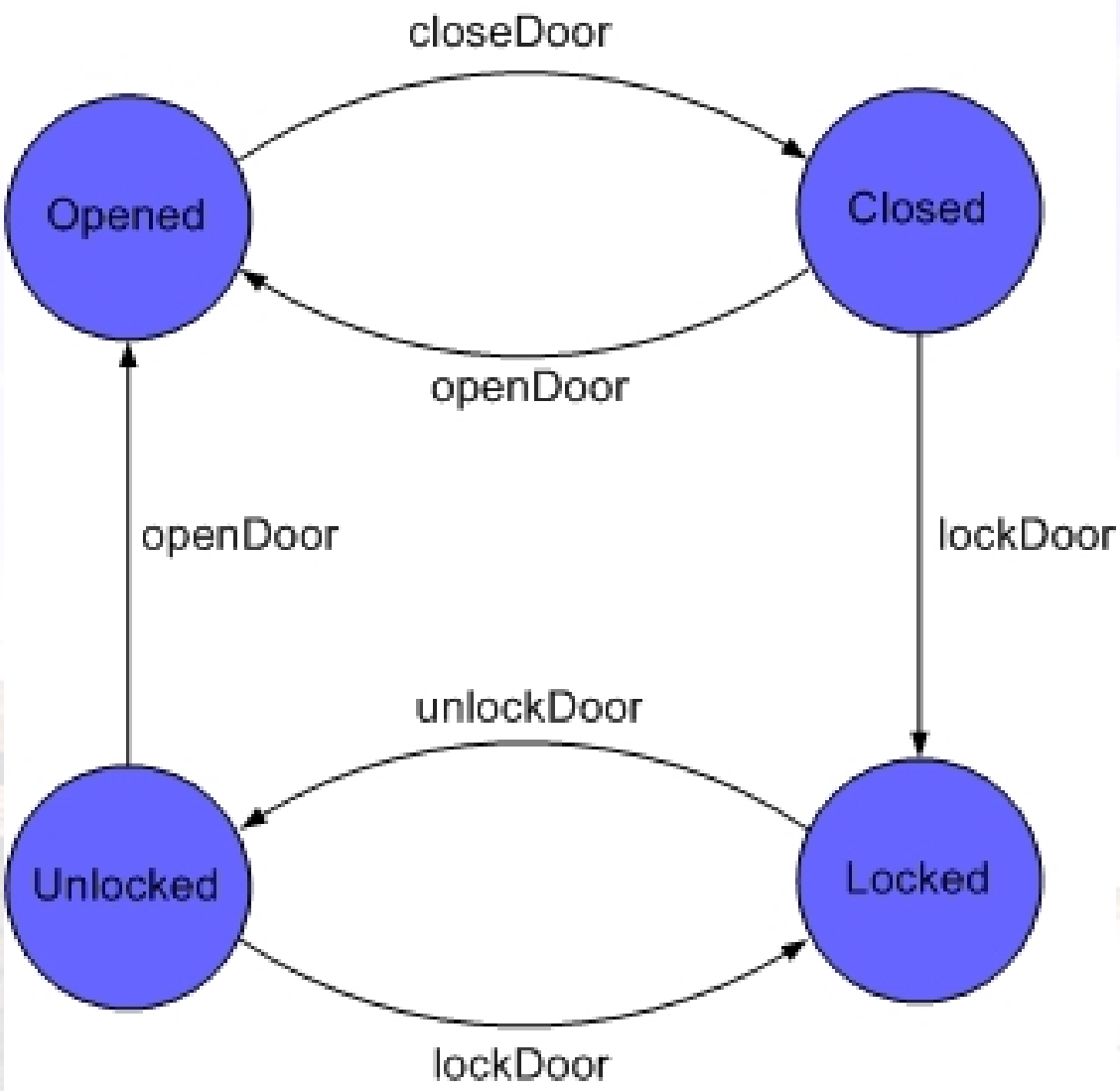
3 有限状态机模型在系统中应用

- 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 为什么使用有限状态机

- 游戏引擎是有限状态机最为成功的应用领域之一，由于设计良好的状态机能够被用来取代部分的人工智能算法，因此游戏中的每个角色或者器件都有可能内嵌一个状态机。
- 考虑RPG（Role-playing Game）游戏中城门这样一个简单的对象，它具有打开（Opened）、关闭（Closed）、上锁（Locked）、解锁（Unlocked）四种状态。当玩家到达一个处于状态Locked的门时，如果此时他已经找到了用来开门的钥匙，那么他就可以利用它将门的当前状态转变为Unlocked，进一步还可以通过旋转门上的把手将其状态转变为Opened，从而成功进入城内。

3 控制城门的状态机


closeDoor
Opened
Closed
openDoor
openDoor
lockDoor
unlockDoor
Unlocked
Locked
lockDoor

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
- 初赛任务
- 决赛任务

4 初赛任务回顾

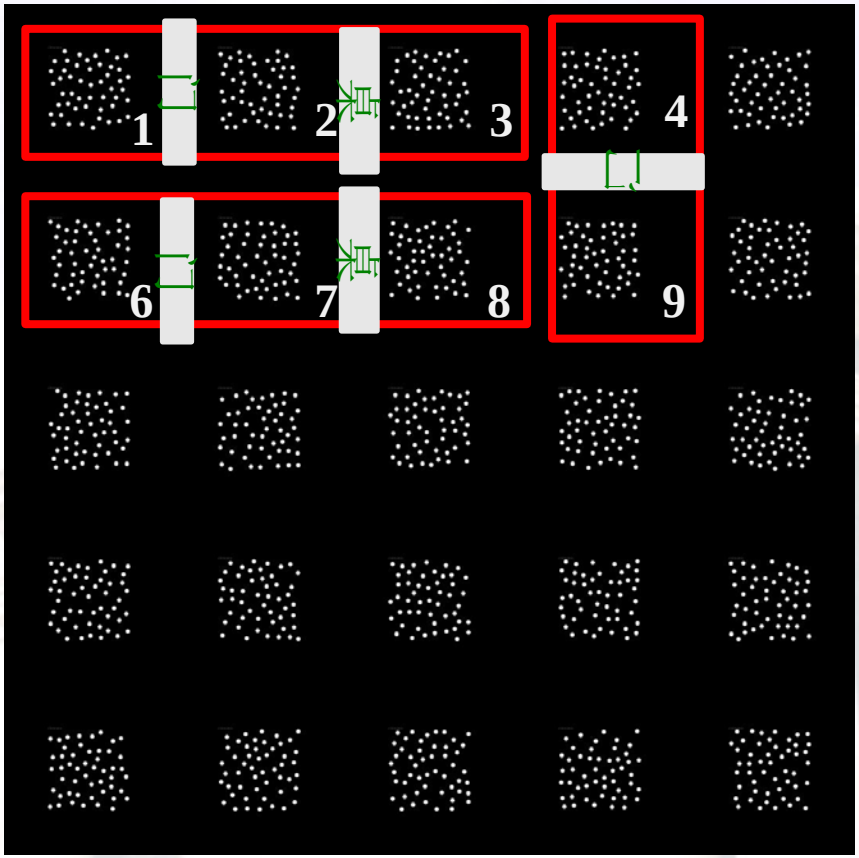
- 流程回顾
  - 起飞：ROS 的理解，话题的发布
  - 找到着火点： 计算机视觉基本方法，OpenCV库调用
  - 穿过着火点：ROS话题的订阅，闭环控制
  - 找到目标物体：CNN 目标检测算法
  - 穿过（停靠）目标物体：控制

4 初赛任务回顾

- 已经提供了基础任务的框架 tello_control：
  - https://github.com/zoeyuchao/tello_control
  - 主要有两个文件：
  - tello_state.py 是课程上的程序，主要完成三件事情：publish一个图像信息，publish一个状态信息，subscribe一个command的控制信息
  - tello_control.py 是助教开发的控制程序，订阅图像和状态信息，通过publish到command话题来完成反馈控制，实现识别定位毯并且飞到定位毯的中心位置。
  - 需要在 Python2.7 环境中运行

4 初赛任务回顾

- 初赛门和桌的摆放位置如下：



1,2,3； 6,7,8作为初赛场地

4,9作为调试场地

4 初赛任务回顾

- 在定位毯上返回坐标如下：
  - 每个定位毯尺寸是 1m x 1m
  - 两个定位毯之间间隔是 1m

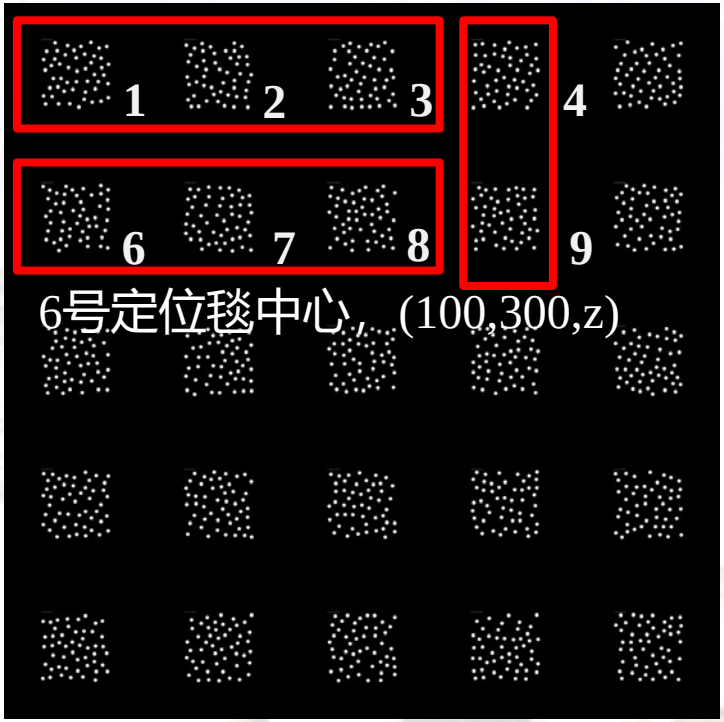
1号定位毯中心，(100,100,z)

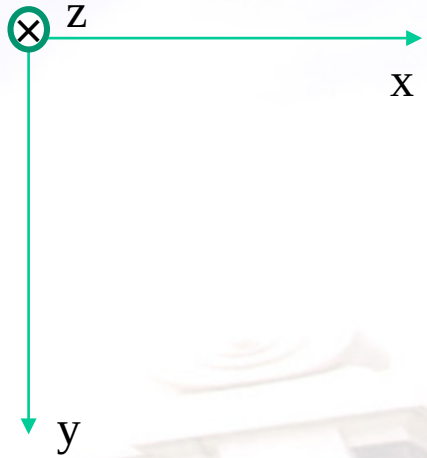
2号定位毯中心，(300,100,z)

3号定位毯中心，(500,100,z)

7号定位毯中心，(300,300,z)

8号定位毯中心，(500,300,z)





4 初赛任务回顾

- 框架 tello_control.py 的基础功能：
  - 通过ROS与 tello_state, tello_img 进行交互。将更新之后的状态保存在全局变量 tello_state 和 img 中
  - 让无人机移动到(100,100,-170)
  - 将摄像头数据显示在屏幕上

```
    def main(self): # 循环检测无人机是否完成任务，无人机移动到100
        while not (self.now_stage == self.taskstages.finished):
            if(self.now_stage == self.taskstages.finding_location):
                self.finding_location()
            elif(self.now_stage == self.taskstages.order_location):
                self.order_location()
        print("finished")
        exit(0)
```

```
def showimg():
    global img,img_lock
    img_lock.acquire()
    cv2.imshow("img",img)
    cv2.waitKey(30)
    img_lock.release()
```

4 初赛任务回顾

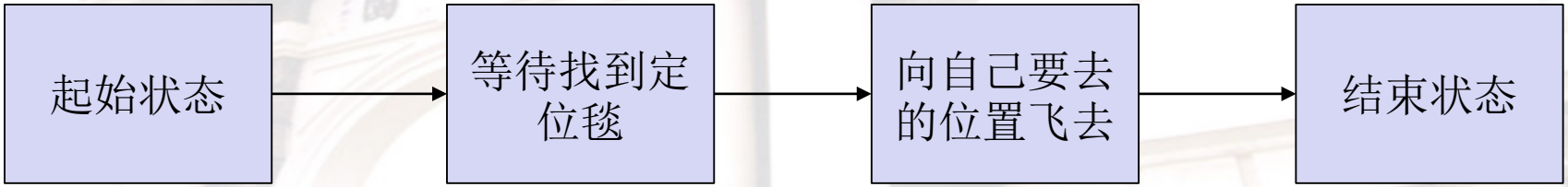
- 任务导航函数解析
  - 规定当前无人机状态： self.now_stage
  - 整个项目无人机课程出现的状态
    - taskstages.finding_location （刚刚起飞还没有找到定位毯）
    - taskstages.order_location （找到了定位毯，还没有飞到位置）
    - taskstages.finished （任务整体完成已经飞到目标位置）

```
def main(self): # 循环检测无人机是否完成任务，无人机移动到100
    while not (self.now_stage == self.taskstages.finished):
        if(self.now_stage == self.taskstages.finding_location):
            self.finding_location()
        elif(self.now_stage == self.taskstages.order_location):
            self.order_location()
    print("finished")
    exit(0)
```

为什么要这么写？？ 如何进行拓展？？

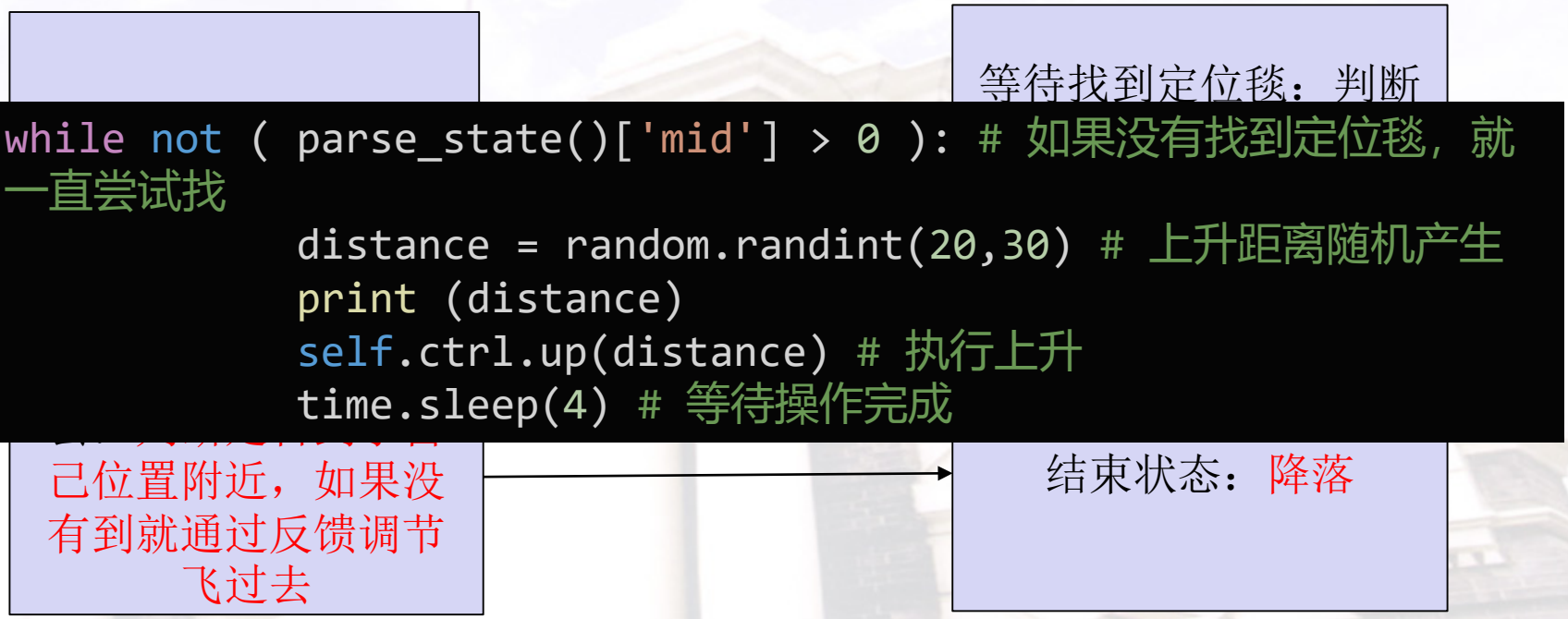
4 有限状态机编程方法

- 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
- 以tello_control为例
  - 梳理状态转移



4 以tello_control为例

- 以tello_control为例
  - 梳理状态转移
  - 实现各状态处理方法



4 以tello_control为例

- 以tello_control为例
  - 梳理状态转移
  - 实现各状态处理方法
  - 判断状态实现状态转移

```
    def finding_location(self): # 找到定位毯位置，一般来说，只要够高
就可以识别到
        print ( "1234123" )
        assert (self.now_stage == self.taskstages.finding_location)
        while not ( parse_state()['mid'] > 0 ): # 如果没有找到定位
毯，就一直尝试找
            distance = random.randint(20,30) # 上升距离随机产生
            print (distance)
            self.ctrl.up(distance) # 执行上升
            time.sleep(4) # 等待操作完成
            showimg()
        self.now_stage = self.taskstages.order_location
```

4 整体可能的框架

State1：起始状态：起飞

State2：等待找到定位毯：判断是否找到定位毯，如果没找到就一直往上飞

State3：飞到一个能看到所有可能出现气球的位置：判断是否到位，没到位就接着飞

State4：找到着火点位置：进行图像处理，找到着火点位置，计算着火点大致距离

State5：移动对准着火点可通过窗户：可以根据实现已知的着火点大致位置靠近窗户（对准）

State6：飞越着火点：往前走

State7：飞到一个能看到待检测物体的位置：操作同 State3。只是到达坐标不同

State8/9/10:找到目标并:操作同 State4/5/6。只是寻找目标、靠近位置不同。

State11：结束：降落

4 评分标准

- 助教会摆放着火点和待检测物体的位置。人工裁判，会给选手发指令开始指令，选手启动程序，助教开始计时，在完成所有任务后停止计时。
  - 完成起飞： 20 分
  - 完成穿过着火点：60分
  - 穿过目标物体：80分
  - 安稳降落：100分。并登记时间。
- 初赛会选择十支左右队伍进入决赛。依据完成时间、得分多寡进行选拔。

4 评分标准

- 集中验收：
  - 第十周周六(11月16日)。从早上八点到晚上九点。稍后会给各组安排具体验收时间，地点是李兆基地下。
- 提前验收：
  - 可以在 11月2日、11月3日、11月9日、11月10日找助教进行提前验收，地点是李兆基地下。

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
- 初赛任务
- 决赛任务

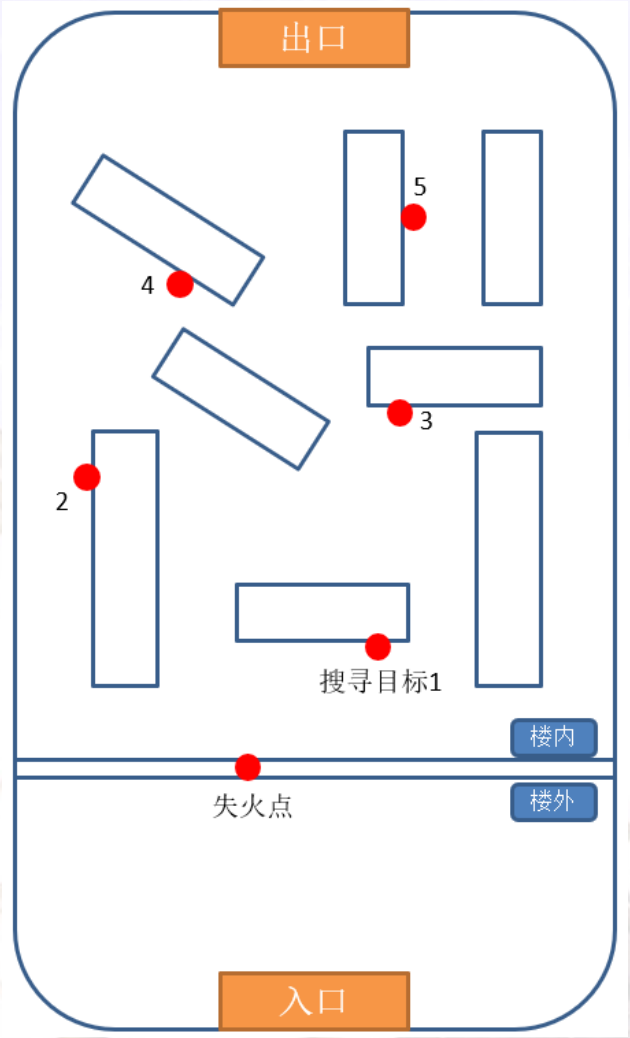
5 评分标准

- 决赛：
  - 13周左右 （12月7日，时间没有完全确定）
  - 第一名奖金 1W
  - 第二名奖金 8K
  - 第三名奖金 5K
  - 决赛其他队伍 >2K

5 决赛任务

智能无人机挑战赛——火场救援

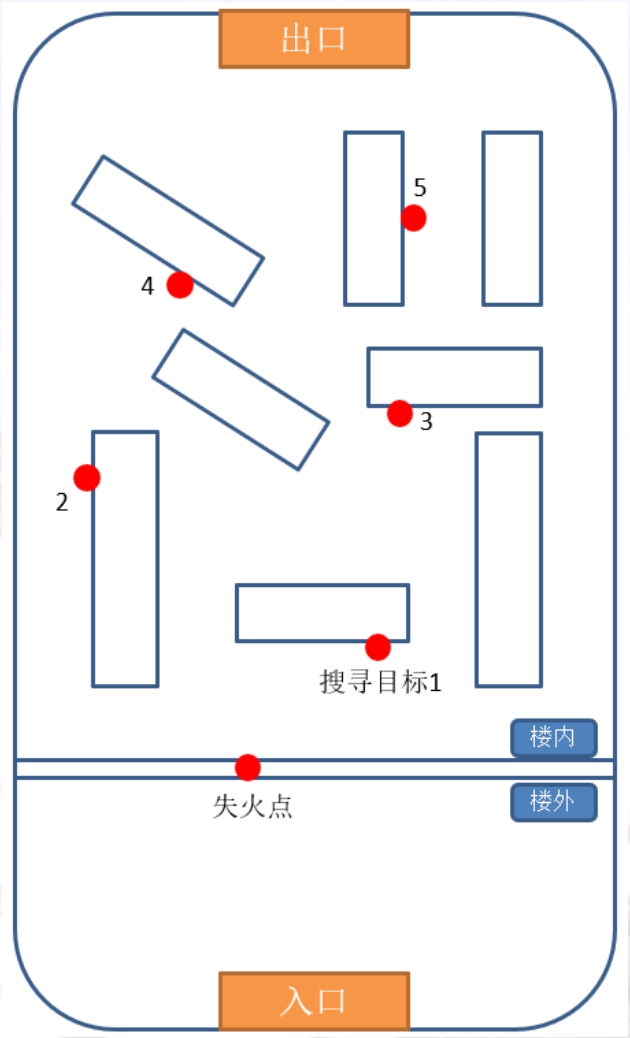
- **任务规划与分解**
  - 巡视并识别模拟大楼着火点
  - 向比赛裁判机器发布着火点坐标
  - 通过着火点所在的模拟窗户进入楼内
  - 识别出要求的搜寻任务目标
  - 离开场地，在指定区域安全落地



5 决赛任务

智能无人机挑战赛——火场救援

- **物体检测**
  - 识别着火点
  - 识别障碍物
- **路径规划**
  - 搜寻目标、巡视路径
  - 寻找出口离开大楼
  - 实时考虑避障
- **底层控制**(比赛中不需要考虑)
  - 飞行控制器
  - 电机驱动控制
  - 无人机定位(待考虑)

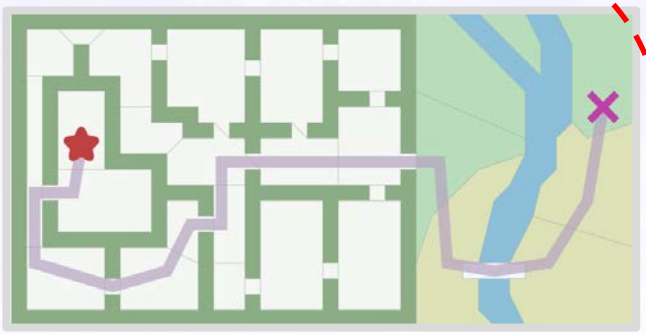


5 决赛任务

目标位姿

方向、速度

路径规划

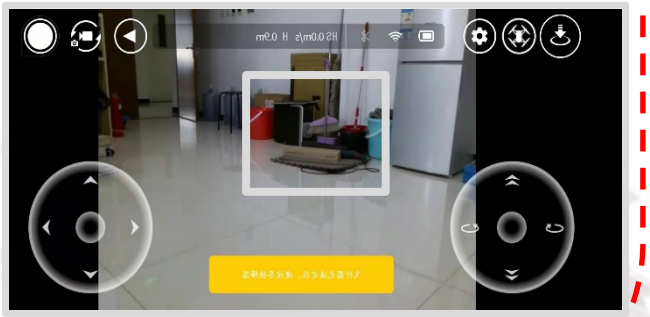


位姿信息

无人机飞控

障碍物坐标

拍摄图像

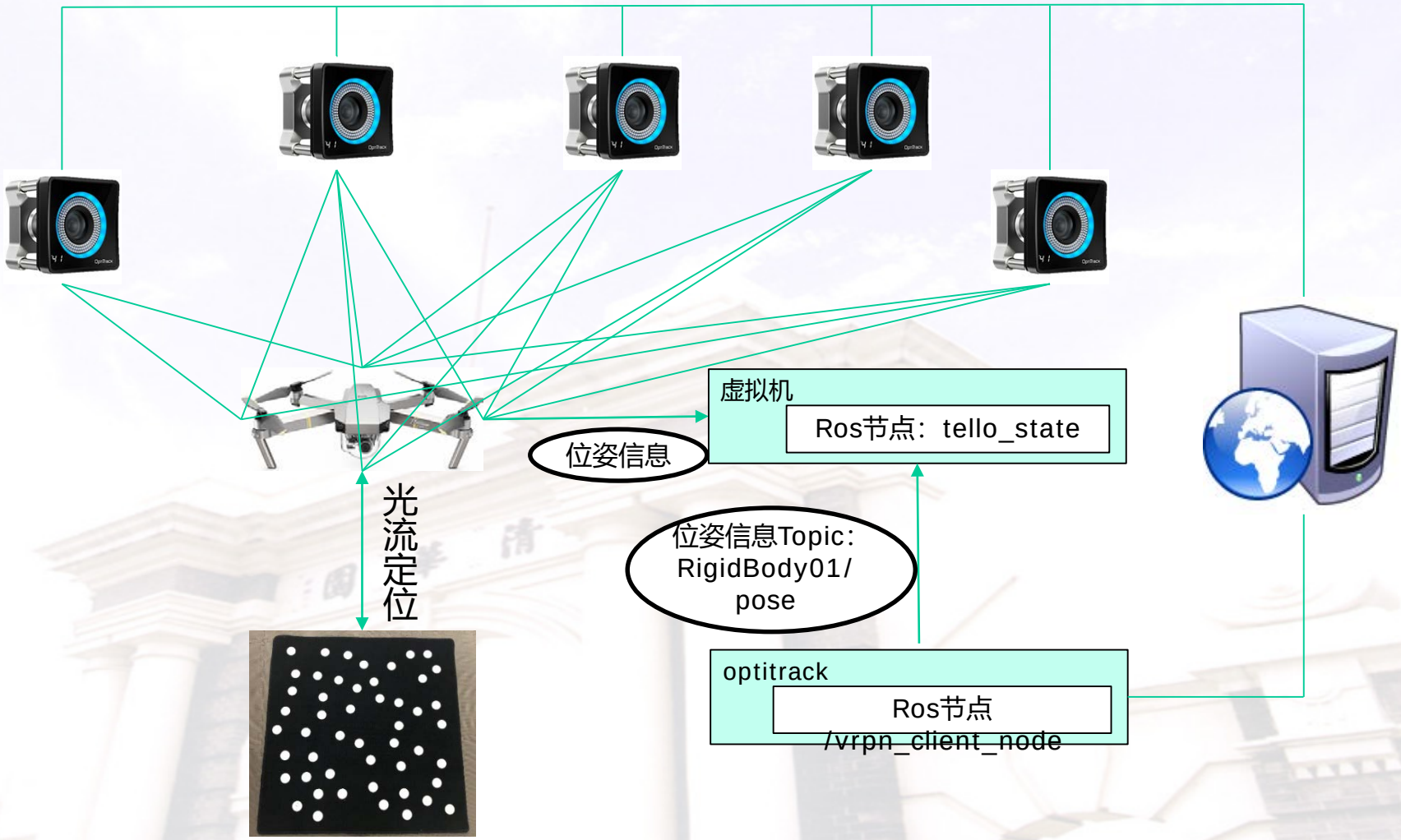


OptiTrack/Marker定位系统

物体识别与定位

图源：http://www.crazepony.com/wiki/opensource-quadcopter.html
https://www.redblobgames.com/pathfinding/a-star/introduction.html

5 决赛任务


虚拟机
Ros节点：tello_state
位姿信息
光流定位
位姿信息Topic:
RigidBody01/
pose
optitrack
Ros节点
/vrpn_client_node

谢谢！