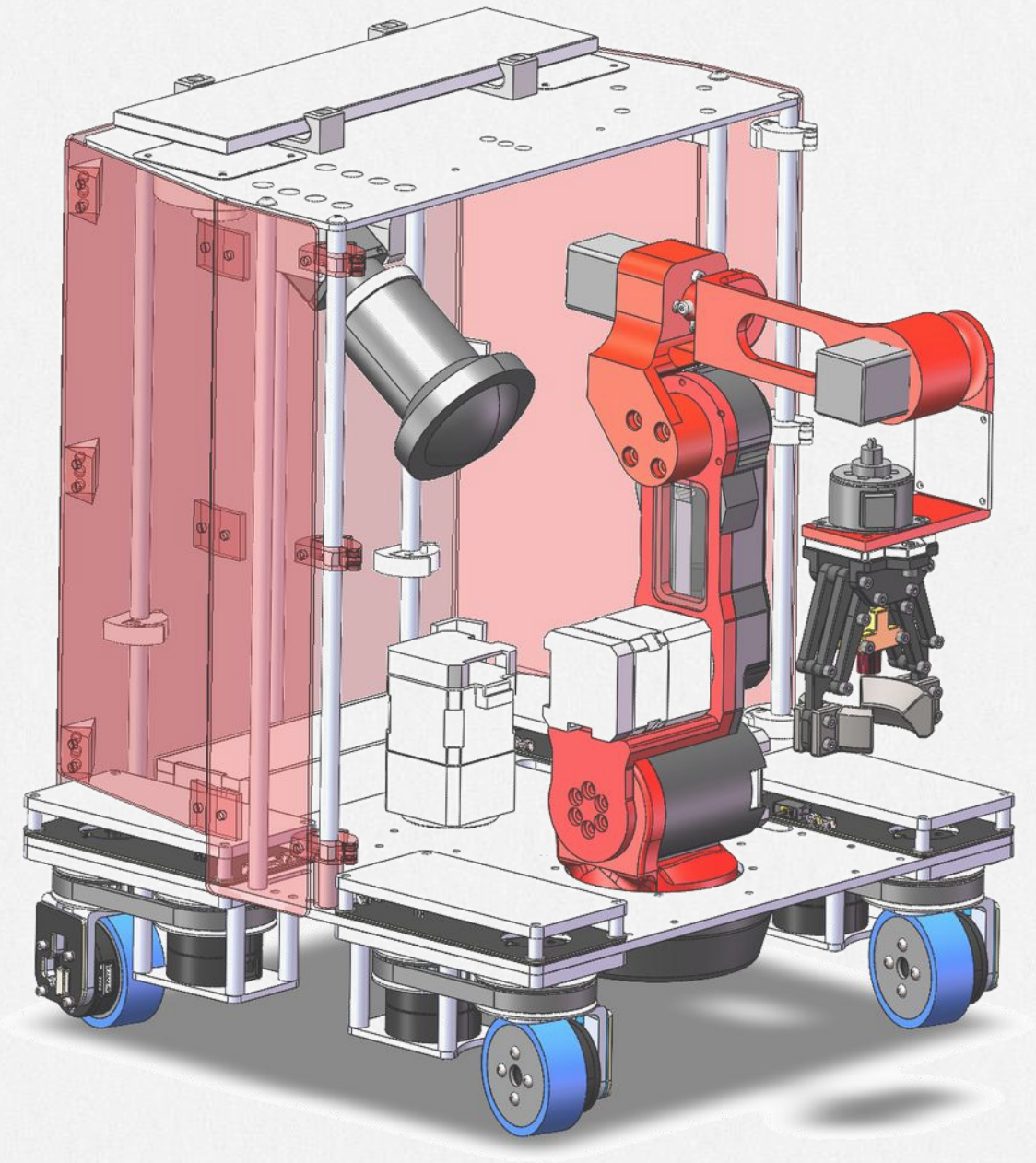


FINS Lab

未来智能网络系统实验室

Fines机器人及其配套设施、算法与软件生态介绍

Connecting Intelligence, Empowering the Future

一个问题

什么影响了机器人的效果

竞技机器人

买来的机器人

既然用的技术是相同的

两者之间的区别是什么?

执行器能力不同

不过这也不像是核心区别

看起来更灵活

各层级算法频率都要高一些

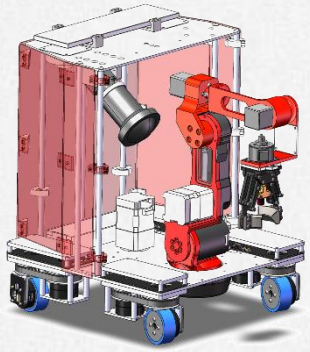
实时性也更好

同时运行的功能更多

程序逻辑并不是单一流程

机器人方案构成

通用的机器人方案



机器人本体

控制器&电路

FineMote

FineVision

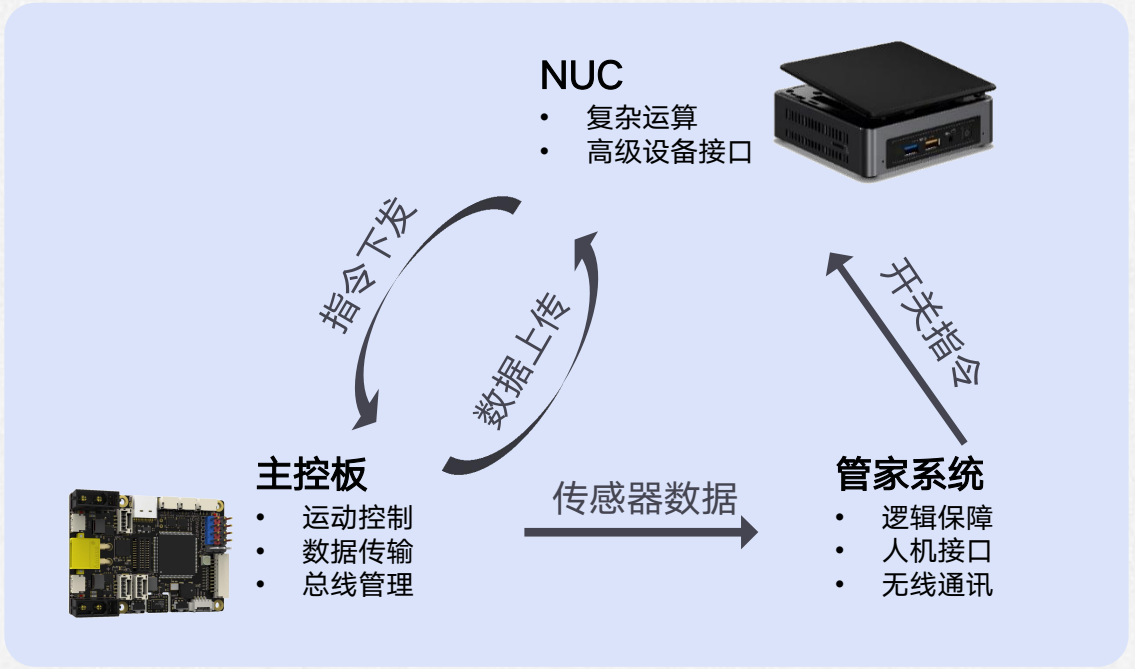
.....

软件生态

平台

系统架构

硬件结构



三处核心设备的功能结构与人脑的结构相似，可以类比理解

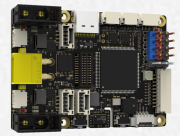
硬件结构

上位机与下位机各司其职

上位机：车载电脑

- 为复杂运算提供算力（毕竟是台电脑）
- 提供相机、激光雷达等设备的驱动（Ubuntu系统）

下位机：主控板



- 实时控制（单片机特性）
- 传感器数据汇总

管家系统

- 不对用户开放，维护机器人基础功能
- 提供系统状态检测与人机接口

特点

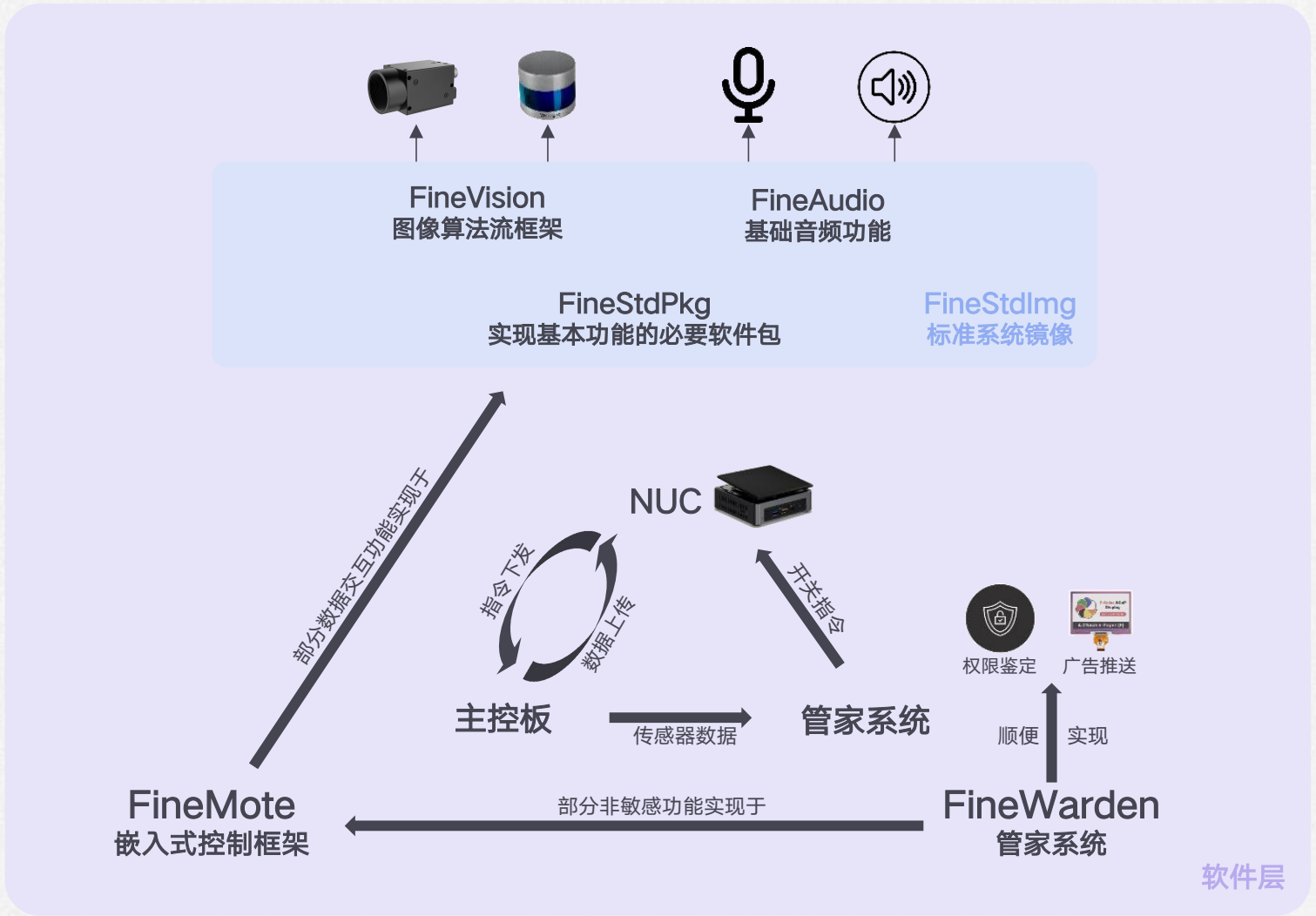
- 三个设备共同完成机器人控制

优点

- 兼顾算力需求和实时特性
- 管家系统保持了用户误操作时机器人的安全性

系统架构

软件生态



软件生态

与硬件结构对应

主控板 -> 嵌入式框架

FineMote

NUC -> 上位机软件生态

FineVision 视觉算法框架

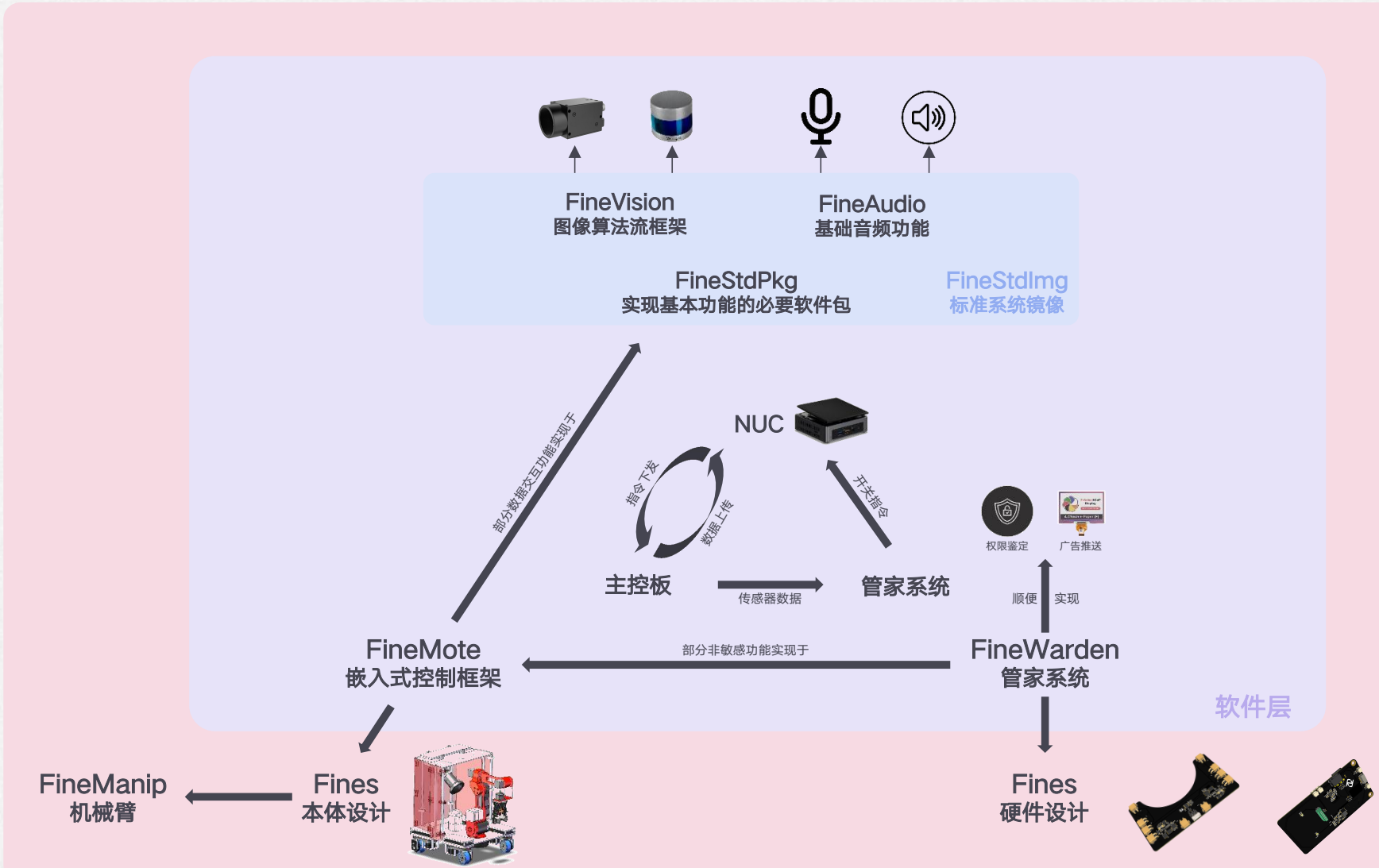
FineStdImg 标准系统镜像

管家系统 -> 管家系统固件

FineWarden

系统架构

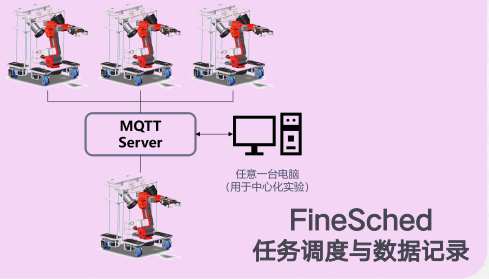
项目框架



项目架构

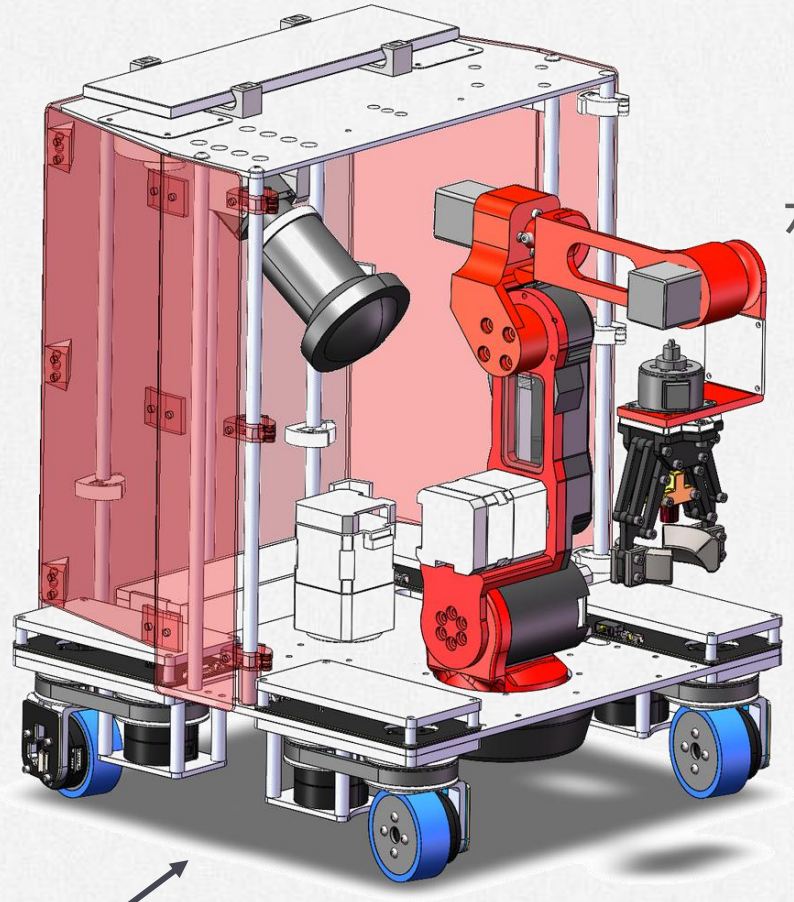
不同层级分段开发

以硬件结构为基础，Fines可被分为若干不同子模块



Fines系列机器人

全国最优秀的实验性移动机械臂（To be done）



六自由度机械臂

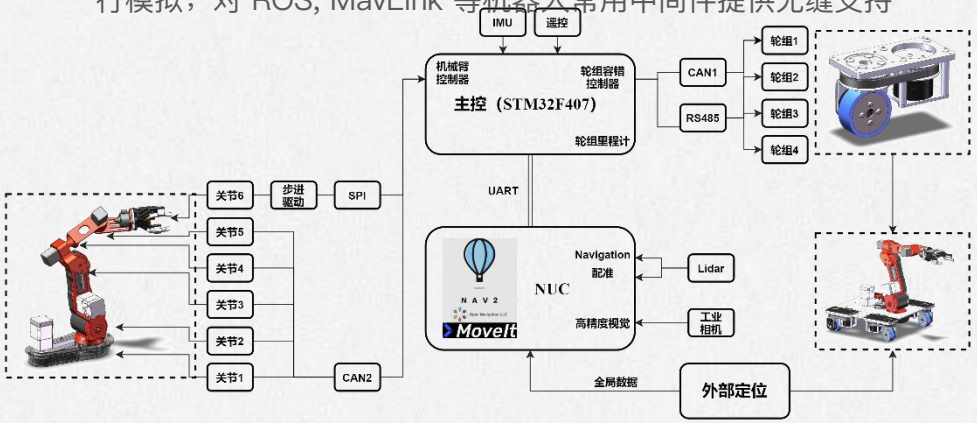
舵轮底盘（POV）

Pseudo-Omnidirectional Vehicle

Fines机器人及其配套设施、算法与软件生态

面向未来的机器人实验平台

Fines一改传统机器人实验平台的设备简陋、硬件易损、脱离生产环境的刻板印象，全制造链重构了设计思路，实现高精度丝滑控制的同时提高了稳定性；兼容工业现场总线，可以对工业生产线进行模拟；对 ROS, MavLink 等机器人常用中间件提供无缝支持



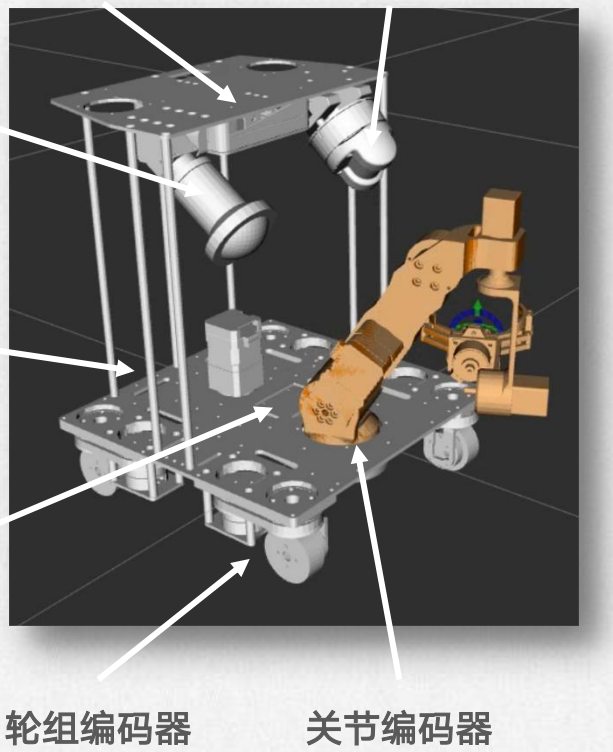
尺寸：300 * 300 * 400 mm
执行精度：1%（开环）0.3mm（全局闭环）0.05mm（机械臂）
最大动力：23.07 N
机械臂最大负载：10 N

感知系统

第一部分

感知系统

系统性传感方案



三个核心问题

相机什么特性?

工业相机，空间视觉

点云什么特性?

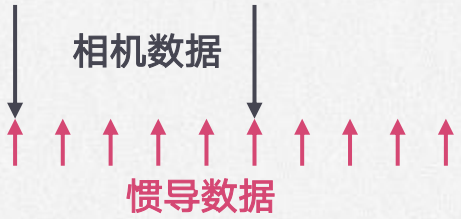
多线雷达，点云/三角面地图

标定怎么做?

时间标定，空间标定

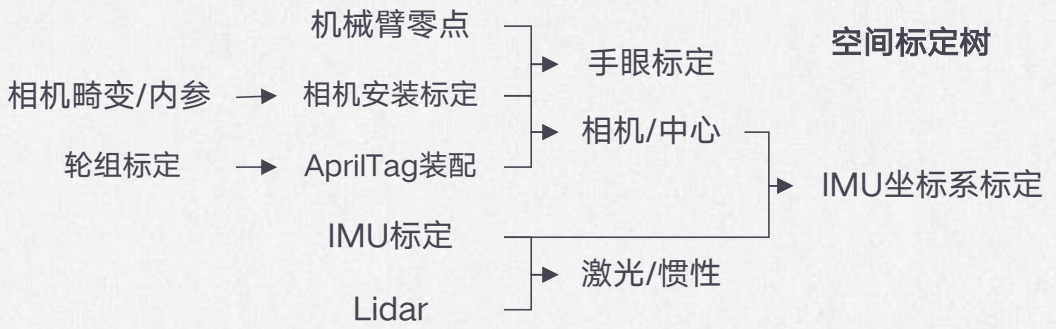
时间标定

同步传感器数据



空间标定

对齐各个传感器的坐标系



感知系统

系统性整合传感器数据

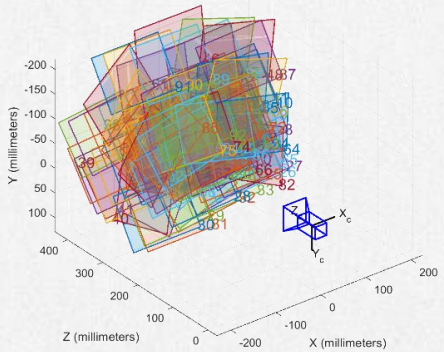
标定：相机标定

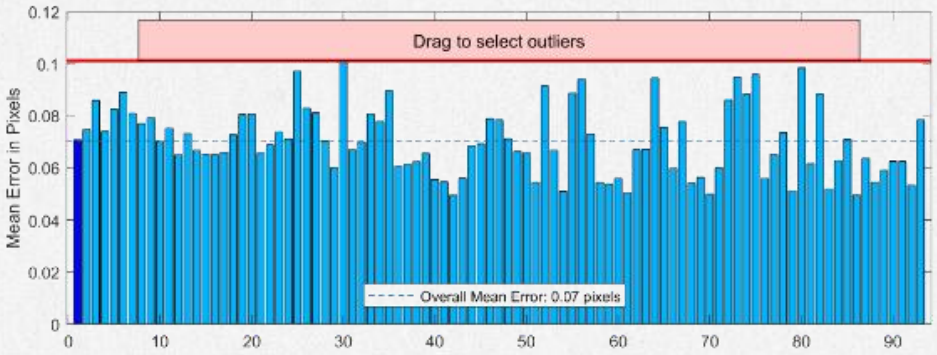
机器人精确感知环境的关键

相机内参及畸变标定

感受野内均匀采样

并对离群数据进行筛选，进一步提高标定精度

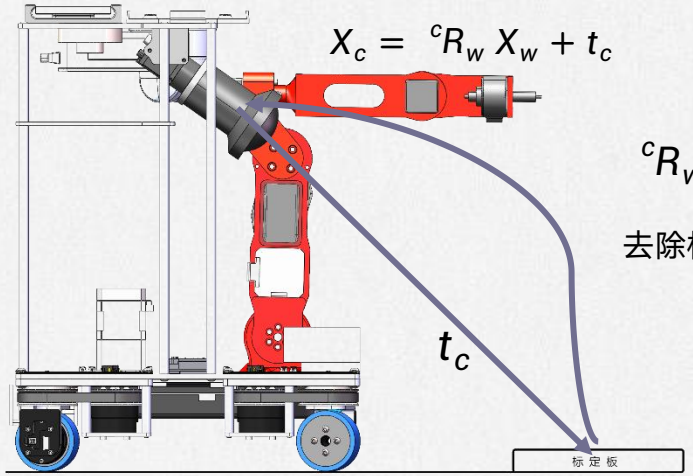




相机安装标定

Perspective–n–Point

相机成像平面位置难以通过直接测量的方式确定，需要利用标定工具获得 3*3旋转矩阵 ${}^{c}R_{w}$ 的两个自由度旋转 及 光心到标定平面的平移关系t_c的垂直分量 的精确数值



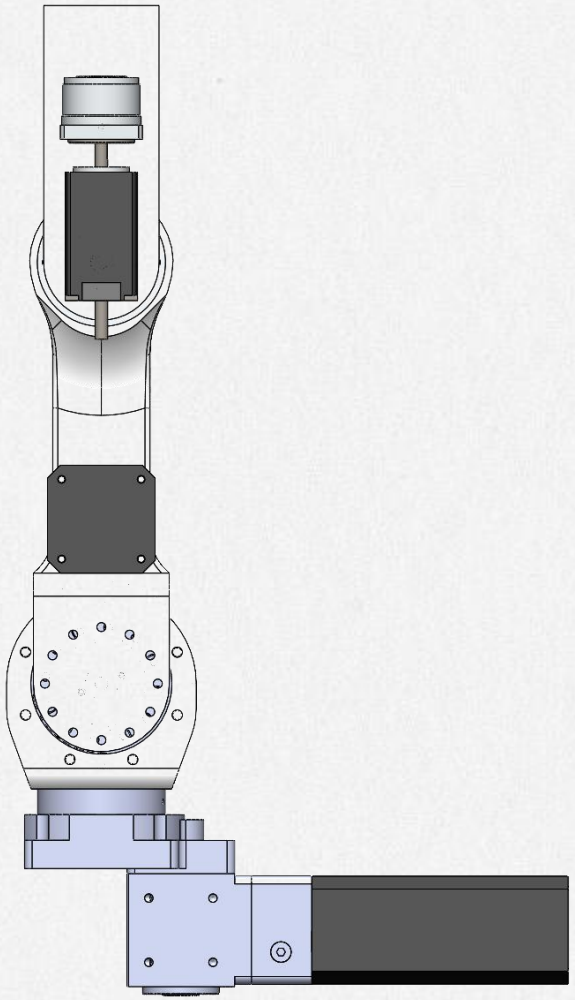
$$X_c = {}^{c}R_w X_w + t_c$$

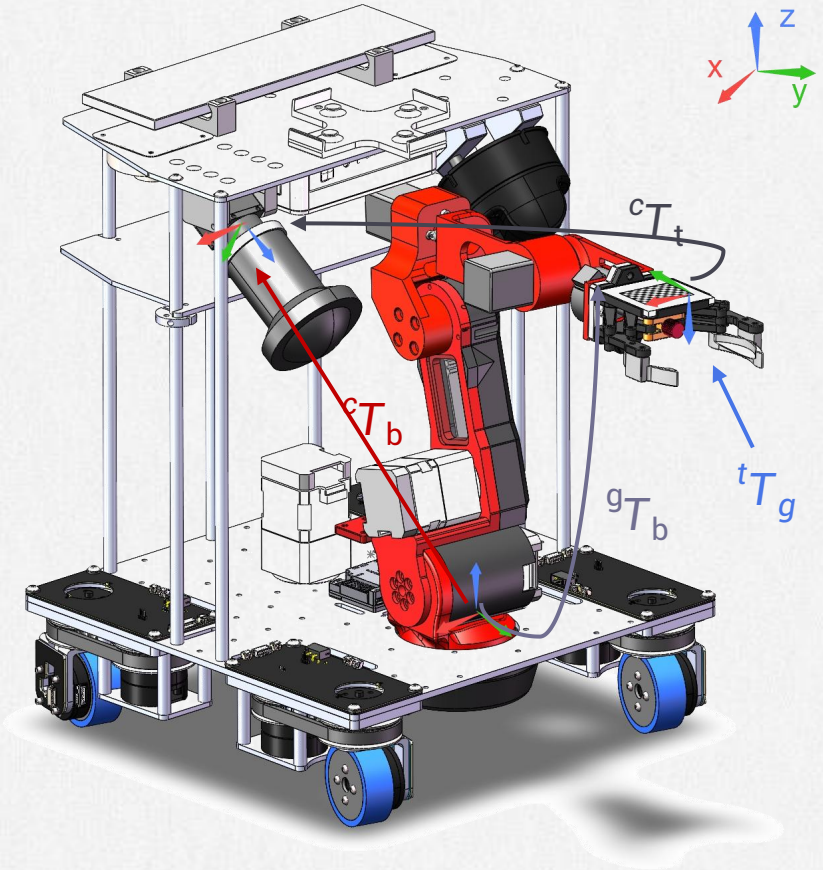
$${}^{c}R_w = {}^{c}_{1}R_w * {}^{c}_{2}R_w * {}^{c}_{3}R_w$$

去除标定板摆放不正的影响 ${}^{c}_{1}R_w$

标定：机械臂标定

FineManip





机械臂零位标定

通过治具将机械臂固定为向上伸长到最高点的位姿，并归零各个关节的编码值

机械臂标定

高质量协同的最后一步

手眼标定（眼在手外）

由于相机坐标系原点和基底无法通过外部测量确定，需要进行手眼标定以确定相机和机械臂两坐标系相对关系

- $^{c}T_{b}$ 是待标定的齐次变换
- $^{g}T_{b}$ 可以通过正运动学获得
- $^{t}T_{g}$ 未知，也不关心
- $^{c}T_{t}$ 可通过PnP获得

$$(^{g}_{2}T_{b})^{-1} \cdot {}^{g}_{1}T_{b} \cdot {}^{b}T_{c} = {}^{b}T_{c} \cdot {}^{c}_{2}T_{t} \cdot (^{c}_{1}T_{t})^{-1}$$

$$A \cdot X = X \cdot B$$

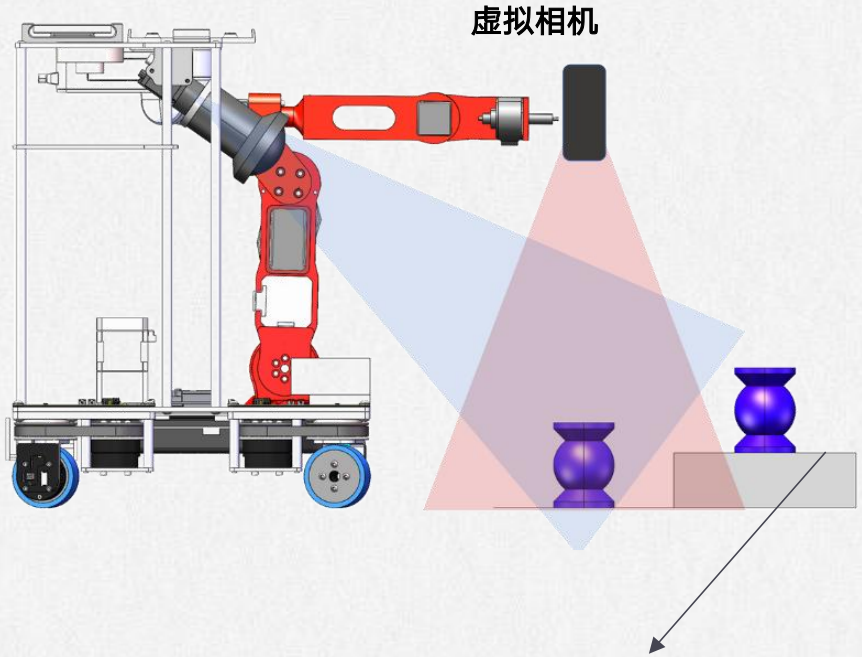
可以通过大量数据解出X，从而得到相机与机械臂的位置关系

相机：空间视觉定位

机器人精确感知环境的关键

内参标定 -> 安装标定 -> 单应变换

空间视觉三部曲



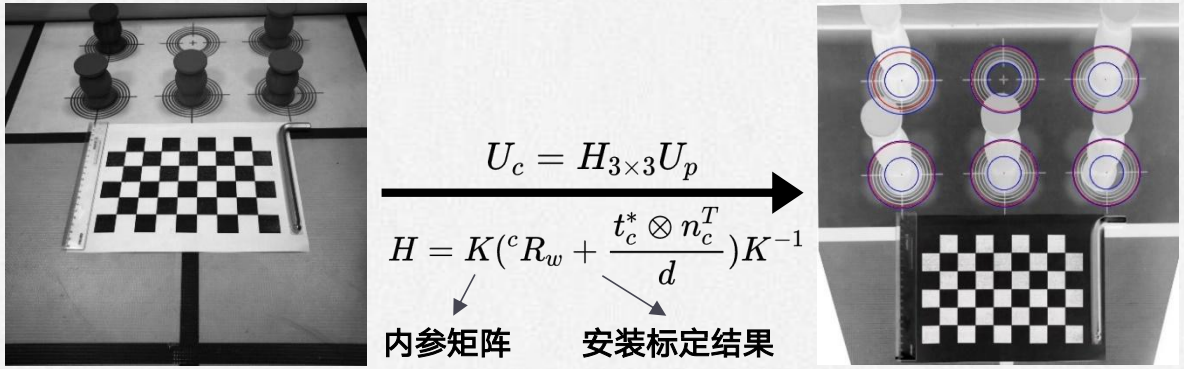
虚拟相机的垂直视角更有助于特征的提取与识别

在Z轴高度已知的情况下，可以计算得到对应特征所处的空间位置

鸟瞰图获取

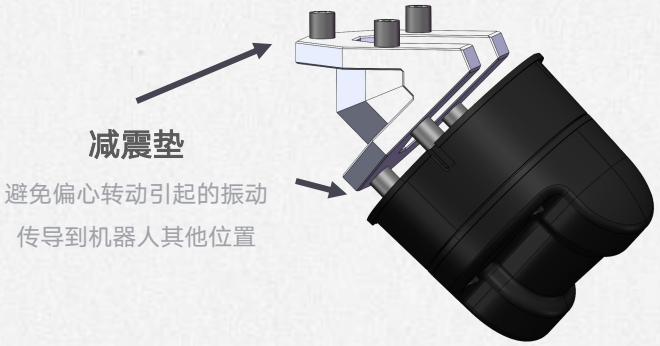
单应变换（Homography）

针对特定Z轴高度的平面，通过单应变换可以得到指定位置垂直虚拟相机的成像结果，从而获得更好的特征识别结果

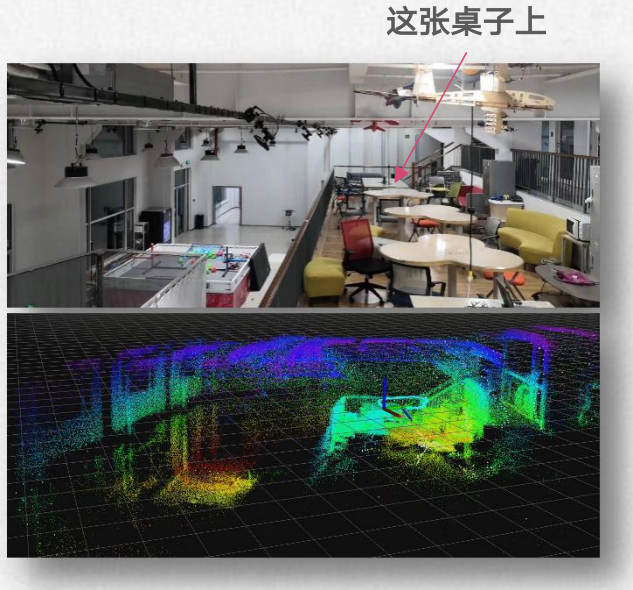


点云：视觉惯性里程计

点云地图获取



激光雷达安装方式

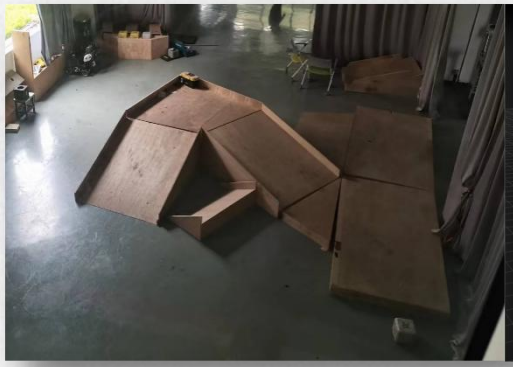


示例：静态点位扫描大范围场景

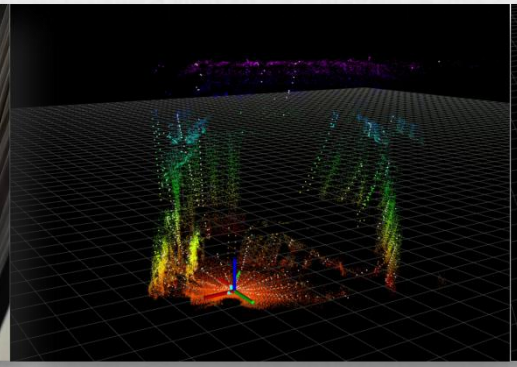
视觉惯性里程计

增量式获取高可靠点云地图

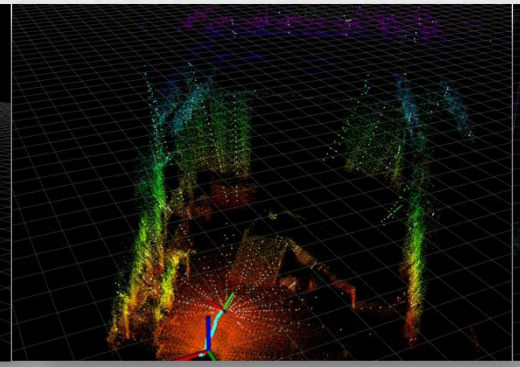
- 激光雷达型号：Unitree-4DLiDAR，21600点/s
- FOV：360° × 90°
- 里程计框架：Point-LIO
- 地图发布形式：PointCloud2？



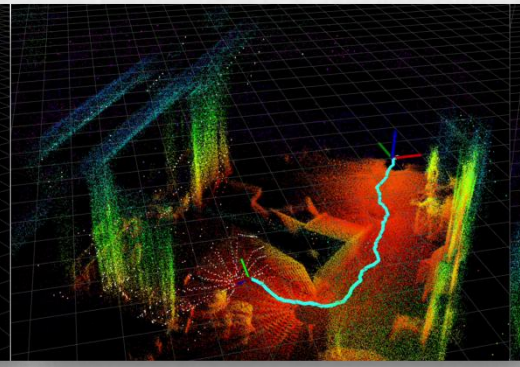
待建模场景



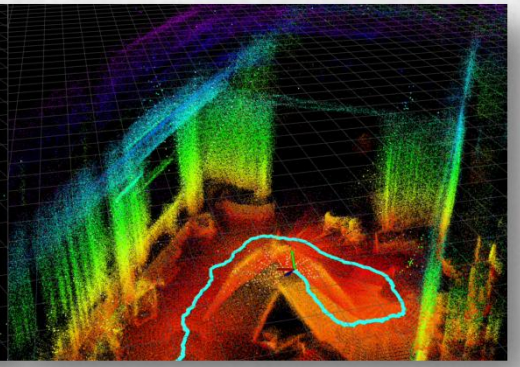
获取第一帧，开始建图



扫描到部分场地特征



绕道场地后面，逐渐补全特征



爬上顶端，补全地图

示例：动态扫描复杂场景

软件生态

第二部分

一个应用案例

Fines软件生态的引入

问题0：开发模式

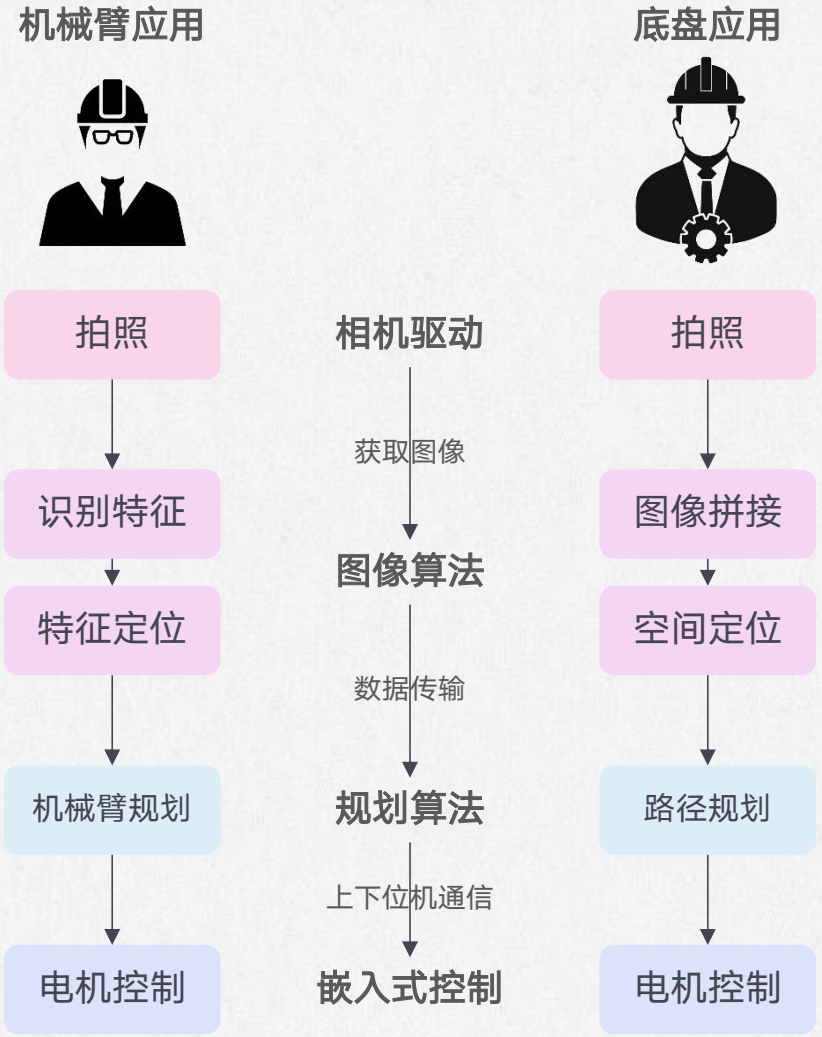
多个开发者之间不会知道对方的实现细节

问题2：线程调度

图像处理流程涉及多个环节，帧率较高时需要并行处理

问题4：嵌入式相关

嵌入式入门难，程序考虑点多且杂，需要一种有效的嵌入式代码组织方式



问题1：设备资源冲突

相机的每一帧图像可能被多个用户程序同时使用

问题3：依赖关系

后级算法依赖前一级算法的处理结果

问题5：调试需求

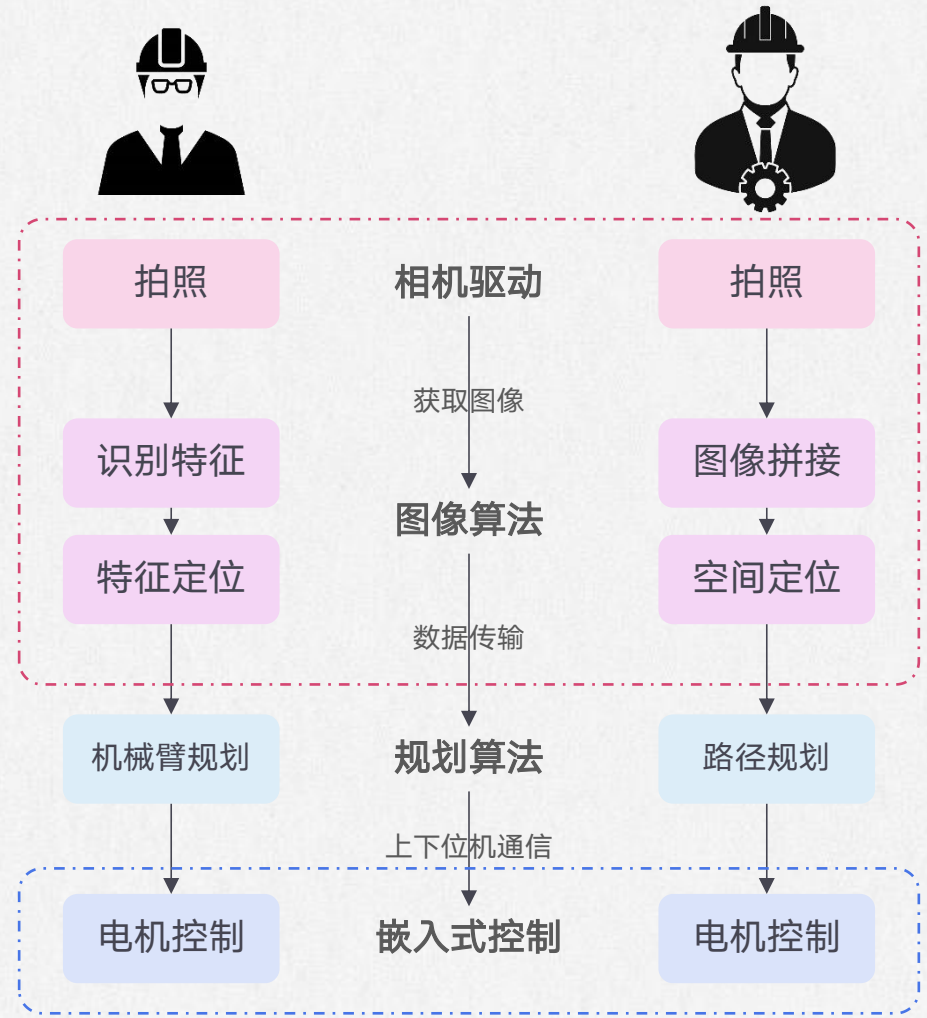
调试阶段可能会频繁调整处理流程的组合方式

处理流程频率是影响机器人效果的关键因素

$e^{0.1} - 1 = 10.5\%$ $e^{0.01} - 1 = 1.01\%$ $e^{0.001} - 1 = 0.1\%$

一个应用案例

Fines软件生态的引入



FineVision

- 相机驱动
- 流式图像处理
- 可视化与系统监控

FineMote

- 电机&传感器驱动
- 实时控制
- 上下位机通信

FineVision

功能详解

功能1：多核处理

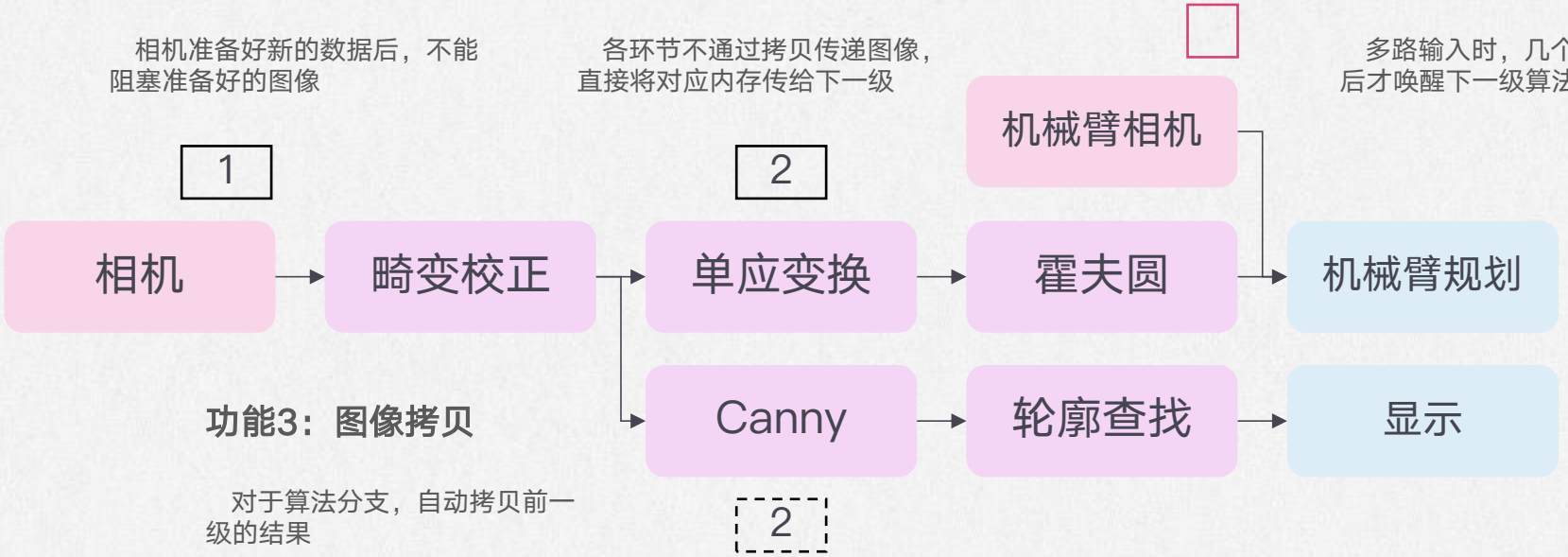
相机准备好新的数据后，不能阻塞准备好的图像

功能2：图像传递

各环节不通过拷贝传递图像，直接将对应内存传给下一级

功能4：多路同步

多路输入时，几个输入均就绪后才唤醒下一级算法

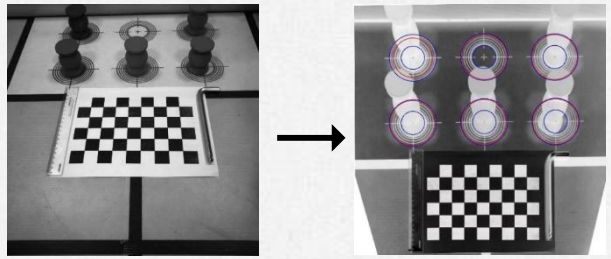


功能3：图像拷贝

对于算法分支，自动拷贝前一级的结果

面向用户的特性

- 仅需要考虑算法流，不需要考虑多线程
- 支持 C++ / Python 混编，数据共享



图像源　算法节点

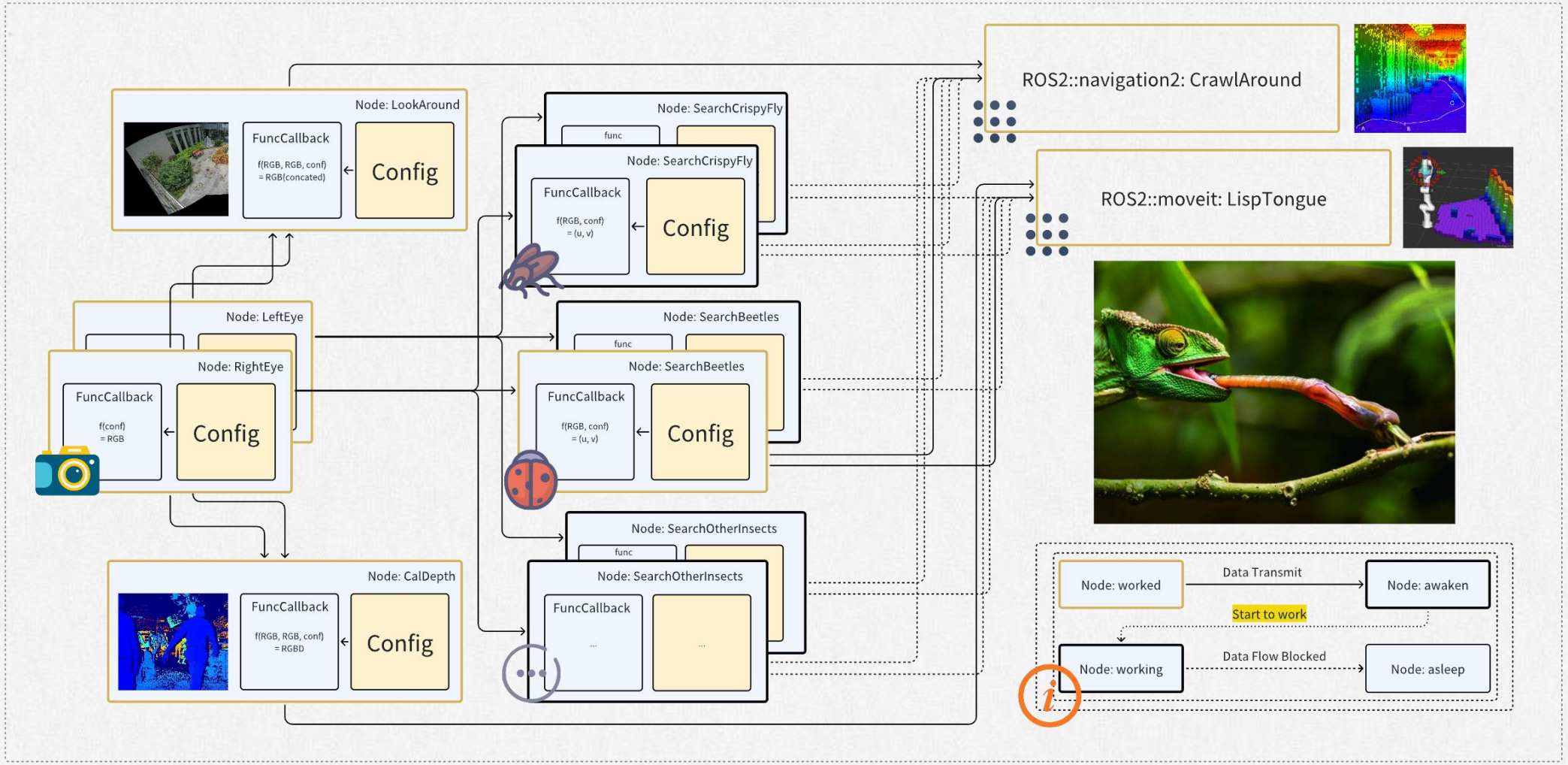
流末级　图像

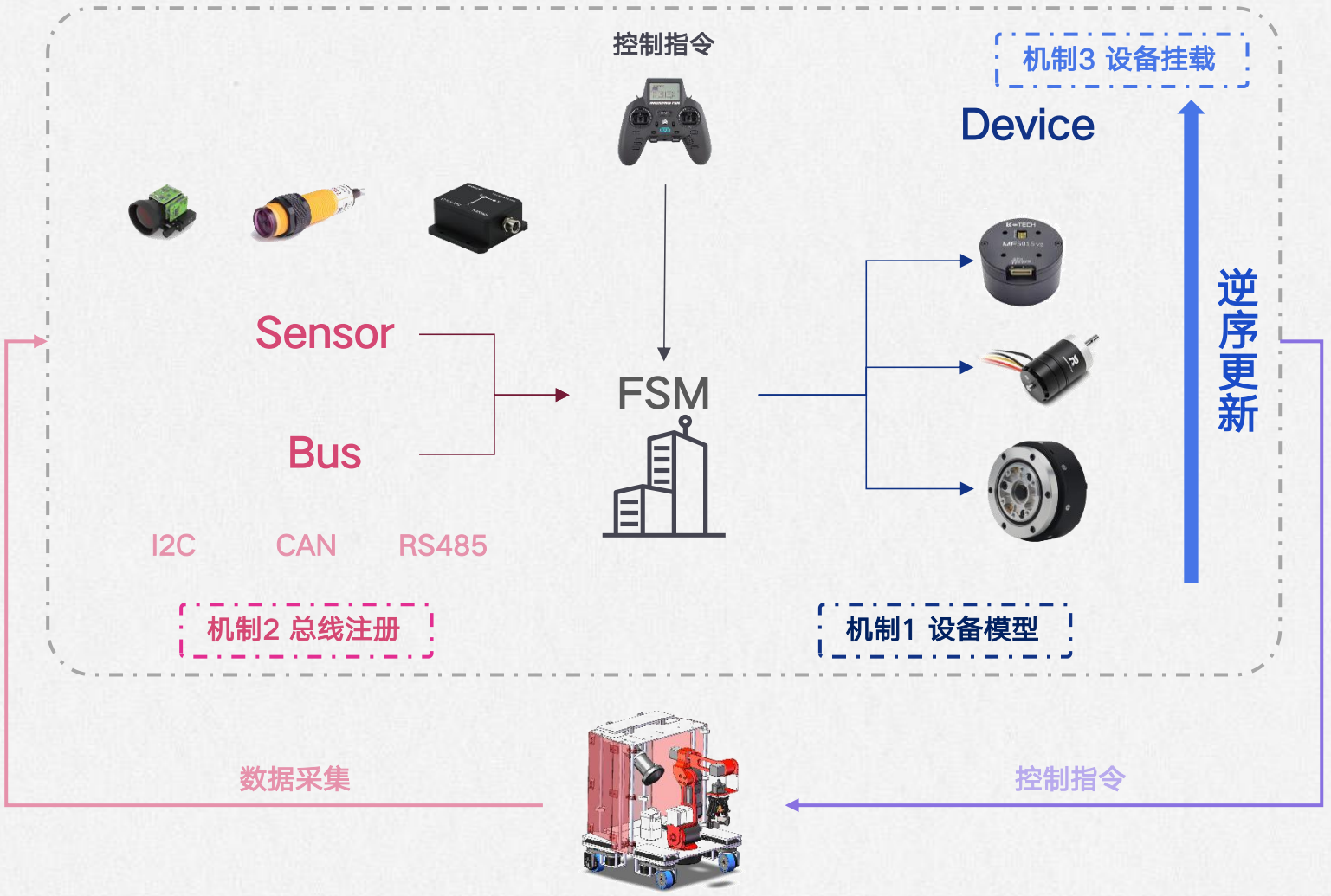
FineVision

复杂应用示例

Flow: Predators of a Chameleon

when a beetle comes by





FineMote
高可靠且灵活的机器人嵌入式框架

Fines采用上下位机协同控制，下位机的任务是处理传感器数据与实时控制需求

功能
适用场景广泛
- 适用于多种不同开发板
- 支持工业总线（CAN RS485 etc.）
- 支持绝大多数能买到的执行器和传感器

应用
框架应用模式
- 代码框架对用户代码实现反调用
- 用户改动仅涉及极少几个文件
- 不强制要求嵌入式开发经验

开发
现代开发模式
- 工程由CMake组织，一站式集成IDE
- Valgrind分析和单元测试（基于QEMU）

FineMote

高可靠且灵活的机器人嵌入式框架

Pixhawk

STM32F427

Cortex-M4@168MHz

2M Flash 256K RAM

协处理器

STM32F100

Cortex-M3@24MHz

CUAV V6X

STM32H753

Cortex-M7@480MHz

2M Flash 1M RAM

协处理器

STM32F10X

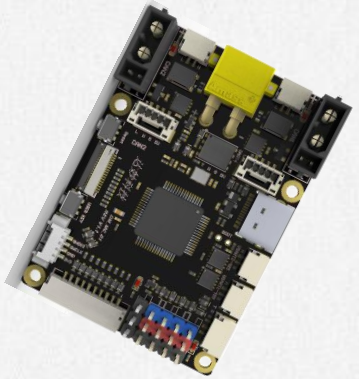
Cortex-M3@72MHz

Mcontroller

STM32H743

Cortex-M7@480MHz

2M Flash 1M RAM



FineMote(DM-MC01)

STM32F446

Cortex-M4@168MHz

512K Flash 128K RAM

普遍受限的硬件资源

主频低、内存小、C++支持程度差

并且需要其他设备交叉编译

实时性与高算力不可兼得，RAM空间较小导致 new 成为了奢侈行为，代码静态性要求较高

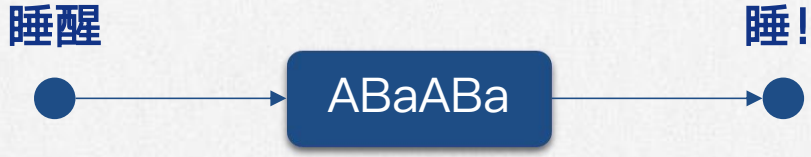
```
class DeviceBase {
private:
    static std::list<DeviceBase*> deviceList;      //存储设备基类指针的设备列表
 virtual void Handle() = 0;                        //纯虚函数，由派生类重写，将被定时调用
public:
    static void DevicesHandle() {                  //统一管理所有设备的定时执行行为
   for (auto rit = deviceList.rbegin(); rit != deviceList.rend(); ++rit) {
            (*rit)->Handle();                      //反向遍历设备列表，并执行其Handle函数
    };
     }
    DeviceBase(){                                  //派生类构造函数调用基类构造函数，将自身的基类指针存入设备列表，完成设备注册
      deviceList.push_back(this);
    };
    ~DeviceBase();
};
```

层级0：框架开发

层级2：用户开发

```
Motor_4010<1>  motor1(MOTOR_INIT_t{0x141, &PID1, POSITION});

void BalabalaFunc(){
    motor1.setTargetAngle(90);
}
```



设备模型

由定时器周期性触发

所有设备以恒定周期（1ms）执行自己的操作，需要的数据在周期间隔内自动送到对应的位置

```
template<int busID>
class Motor_4010 : public DeviceBase{
    CAN_Agent<busID> canAgent;                     //CAN总线代理对象
 void Handle() override {
        uint8_t data = canAgent.rxbuf[0];
  };
public:
    explicit Motor_4010(uint32_t addr):canAgent(addr){};
    ~Motor_4010(){};
};
```

层级1：设备开发

FineMote

机制详解

```
class DeviceBase {
private:
    static std::list<DeviceBase*> deviceList;      //存储设备基类指针的设备列表
 virtual void Handle() = 0;                        //纯虚函数，由派生类重写，将被定时调用
public:
    static void DevicesHandle() {                  //统一管理所有设备的定时执行行为
   for (auto rit = deviceList.rbegin(); rit != deviceList.rend(); ++rit) {
            (*rit)->Handle();                      //反向遍历设备列表，并执行其Handle函数
   };
    }
    DeviceBase(){                                  //派生类构造函数调用基类构造函数，将自身的基类指针存入设备列表，完成设备注册
      deviceList.push_back(this);
    };
    ~DeviceBase();
};
```

层级0：框架开发

层级2：用户开发

```
Motor_4010<1>  motor1(MOTOR_INIT_t{0x141, &PID1, POSITION});

void BalabalaFunc(){
    motor1.setTargetAngle(90);
}
```

总线注册

解决数据来源的异步问题

为解决不同设备共用总线外设的问题，每个设备向总线对象注册后即可收到期望收到的总线数据

```
template<int busID>
class Motor_4010 : public DeviceBase{
    CAN_Agent<busID> canAgent;                     //CAN总线代理对象
 void Handle() override {
        uint8_t data = canAgent.rxbuf[0];
    };
public:
    explicit Motor_4010(uint32_t addr):canAgent(addr){};
    ~Motor_4010(){};
};
```

层级1：设备开发

FineMote

机制详解

```
class DeviceBase {
private:
    static std::list<DeviceBase*> deviceList;      //存储设备基类指针的设备列表
 virtual void Handle() = 0;                        //纯虚函数，由派生类重写，将被定时调用
public:
    static void DevicesHandle() {                  //统一管理所有设备的定时执行行为
     for (auto rit = deviceList.rbegin(); rit != deviceList.rend(); ++rit) {
            (*rit)->Handle();                      //反向遍历设备列表，并执行其Handle函数
    };
     }
    DeviceBase(){                                  //派生类构造函数调用基类构造函数，将自身的基类指针存入设备列表，完成设备注册
      deviceList.push_back(this);
    };
    ~DeviceBase();
};
```

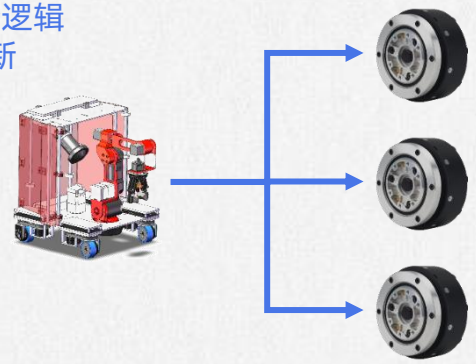
层级0：框架开发

层级2：用户开发

```
Motor_4010<1>  motor1(MOTOR_INIT_t{0x141, &PID1, POSITION});

void BalabalaFunc(){
    motor1.setTargetAngle(90);
}
```

状态沿设备组合逻辑
自上向下更新



设备挂载

初始化与句柄均被自动调用

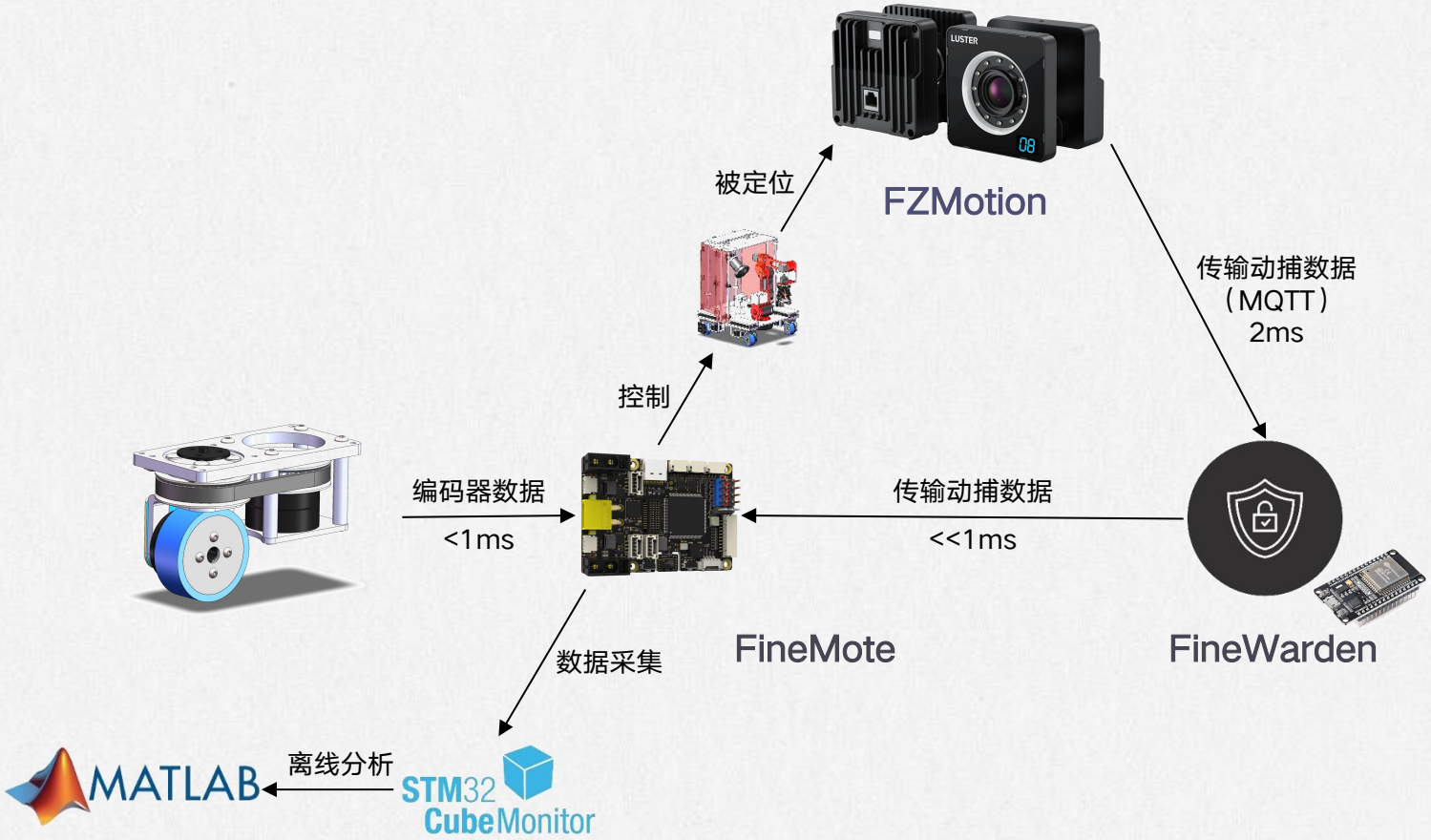
保证了沿设备层级关系自上至下更新状态，并且将用户的修改限制在了单个文件当中

```
template<int busID>
class Motor_4010 : public DeviceBase{
    CAN_Agent<busID> canAgent;                     //CAN总线代理对象
 void Handle() override {
        uint8_t data = canAgent.rxbuf[0];
    };
public:
    explicit Motor_4010(uint32_t addr):canAgent(addr){};
    ~Motor_4010(){};
};
```

层级1：设备开发

动力学优化

第三部分

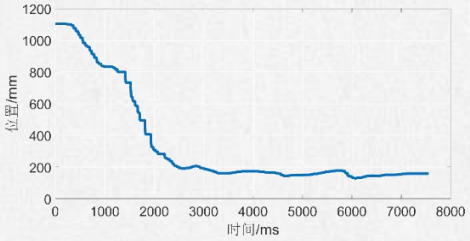


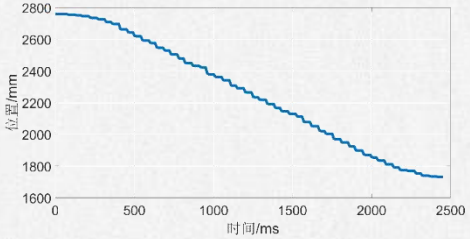
实验方案

检验模型的正确程度

通过离线分析动捕系统获取的轨迹数据，对模型的效果进行评估

最理想的有效实验结果，是通过自学习的方式对各个轮组的支持力影响的权值进行回归。如果能够收敛，则说明模型的准确程度极高





系统集成

第四部分

FineManip

高性能高可靠性机械臂设计

机械臂设计
及参数校核

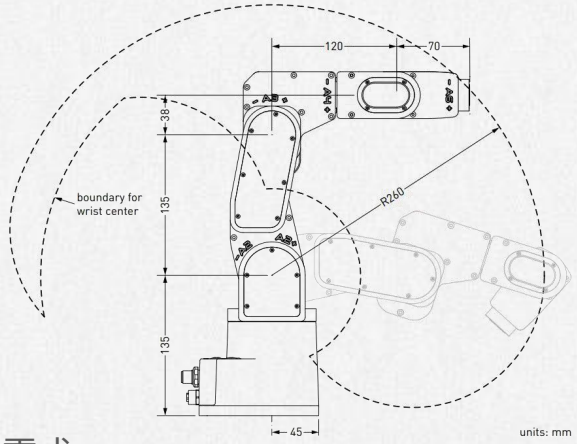
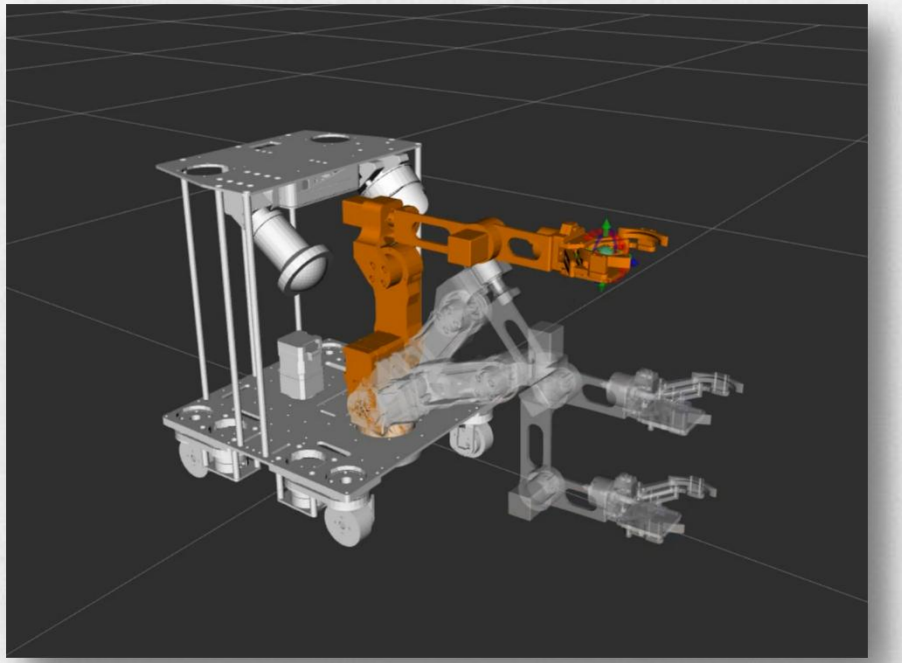
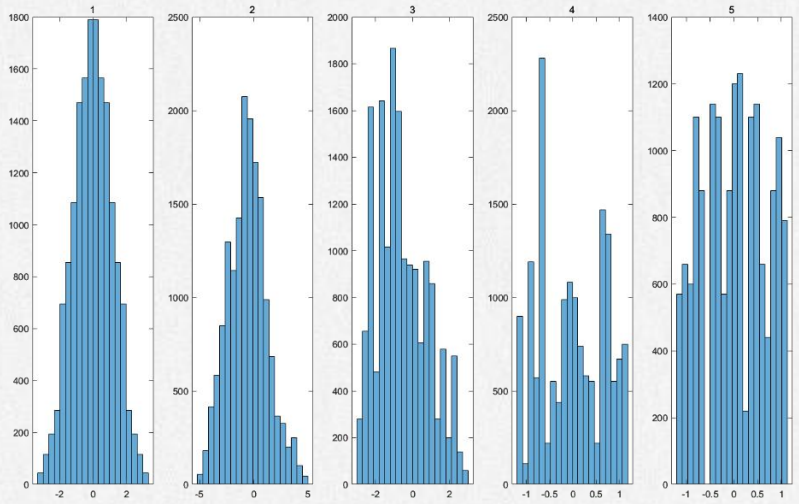


Figure 6: The dimensions of the Meca500 (R3 and R4)

- 满足Pieper条件，具有封闭逆解
- 全工作空间采样，统计关节力矩需求



MoveIt!



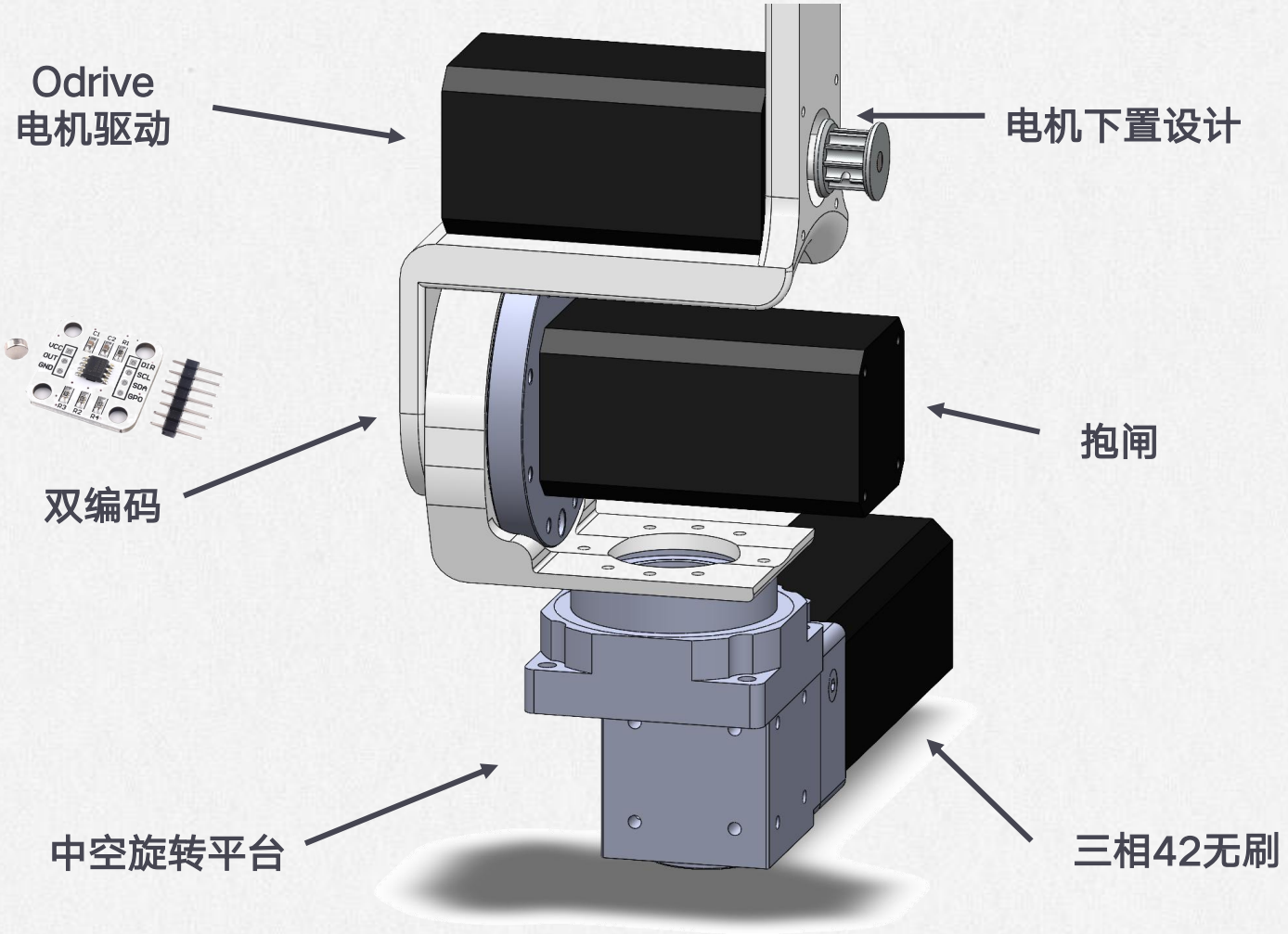
θ_i	d_i/mm	α_i	a_i/mm
θ_1	0	0	0
θ_2	0	$\frac{\pi}{2}$	0
θ_3	0	π	164
θ_4	120	$-\frac{\pi}{2}$	50
θ_5	0	$-\frac{\pi}{2}$	0

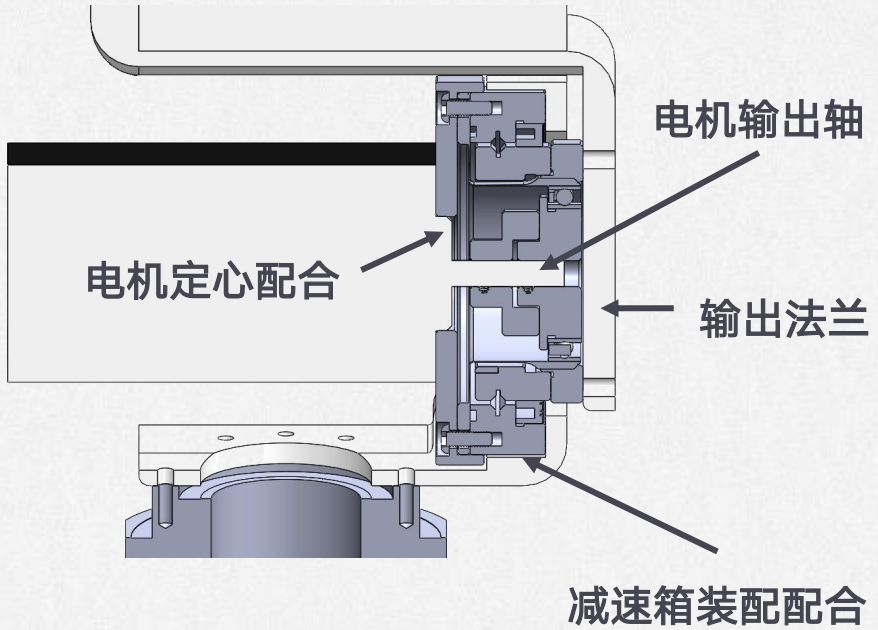
[1] Pieper D L. The kinematics of manipulators under computer control[M]. Stanford University, 1969.

机械臂设计

高性能高可靠性桌面级6DoF机械臂

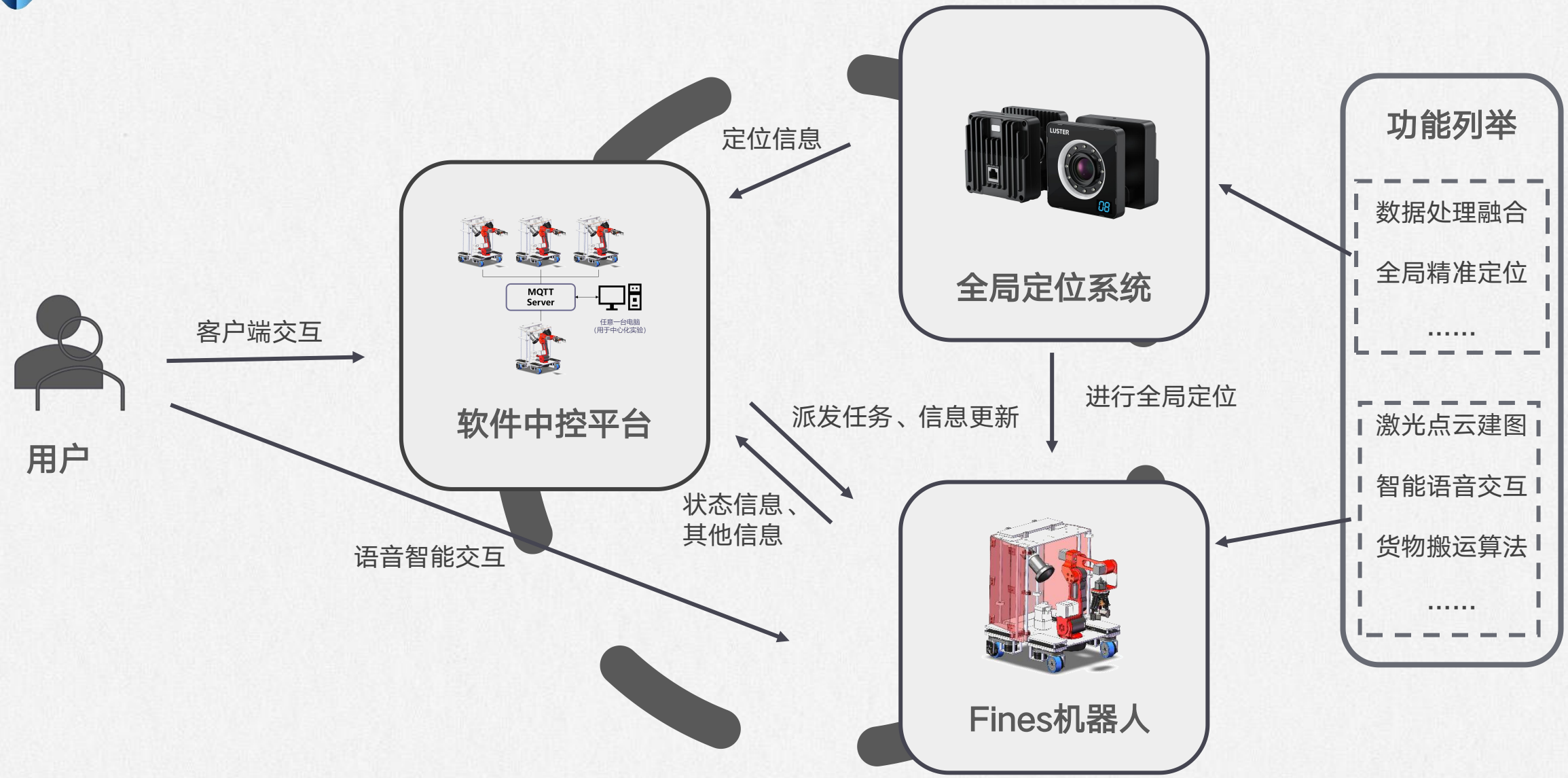
- 力学特性
- 控制与传感
- 性能与可靠性





FineSched

机器人调度系统

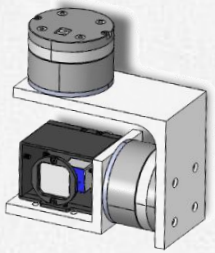


机器人实验系统

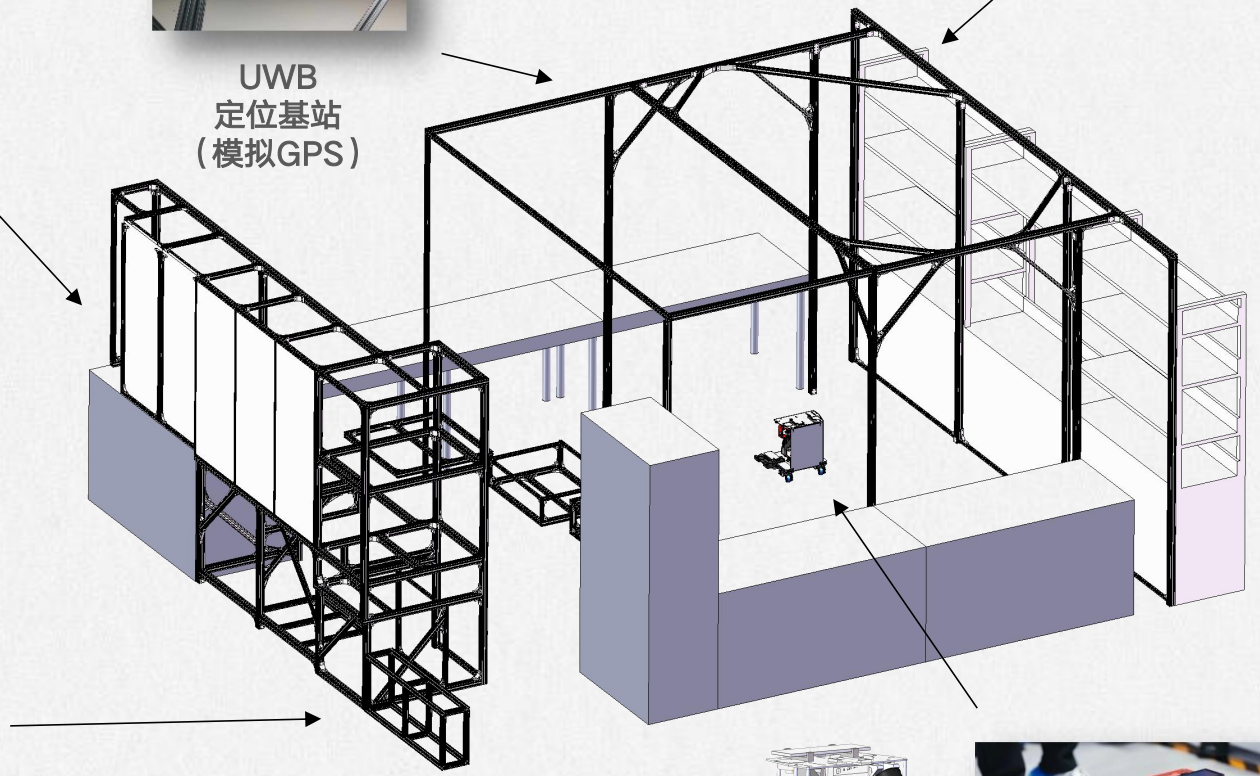
各项关键技术的有机结合体

存储区域

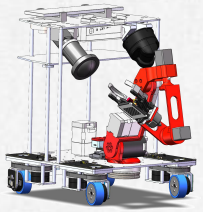
UWB
定位基站
（模拟GPS）



光学跟踪
定位基站



展示区域

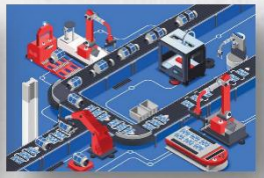


机器人个体

多机器人沙盒平台

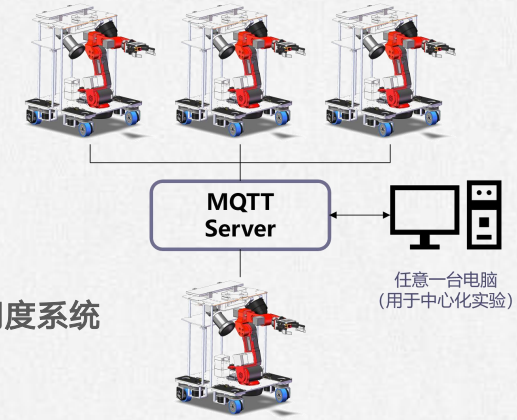
在实验室条件模拟多种应用场景

多智能体攻防



工业4.0

恶劣环境探索



THANK YOU

连接智能，赋能未来