0参考视频：B站ROS机器人操作系统入门-中国大学MOOC，古月居-古月-ROS入门21讲

古月居代码：<https://github.com/JerryRain/rosexplore_ws>

参考书籍：[1]《ROS机器人开发实践》胡春旭，[2]ros官网文件《ROS机器人编程-从基本概念到机器人应用程序编程实践》，ROS\_Robot\_Programming\_CN

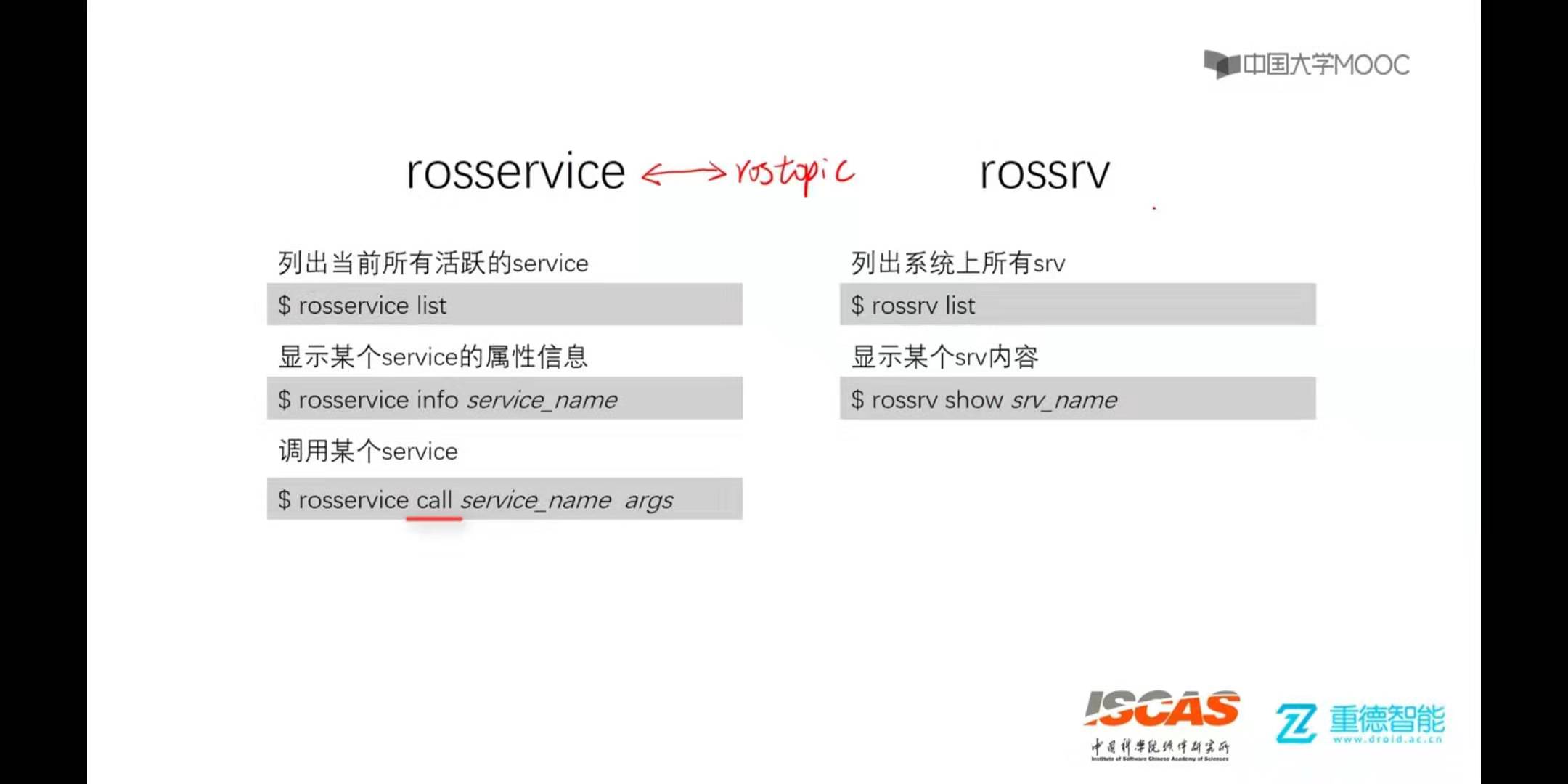
问题：

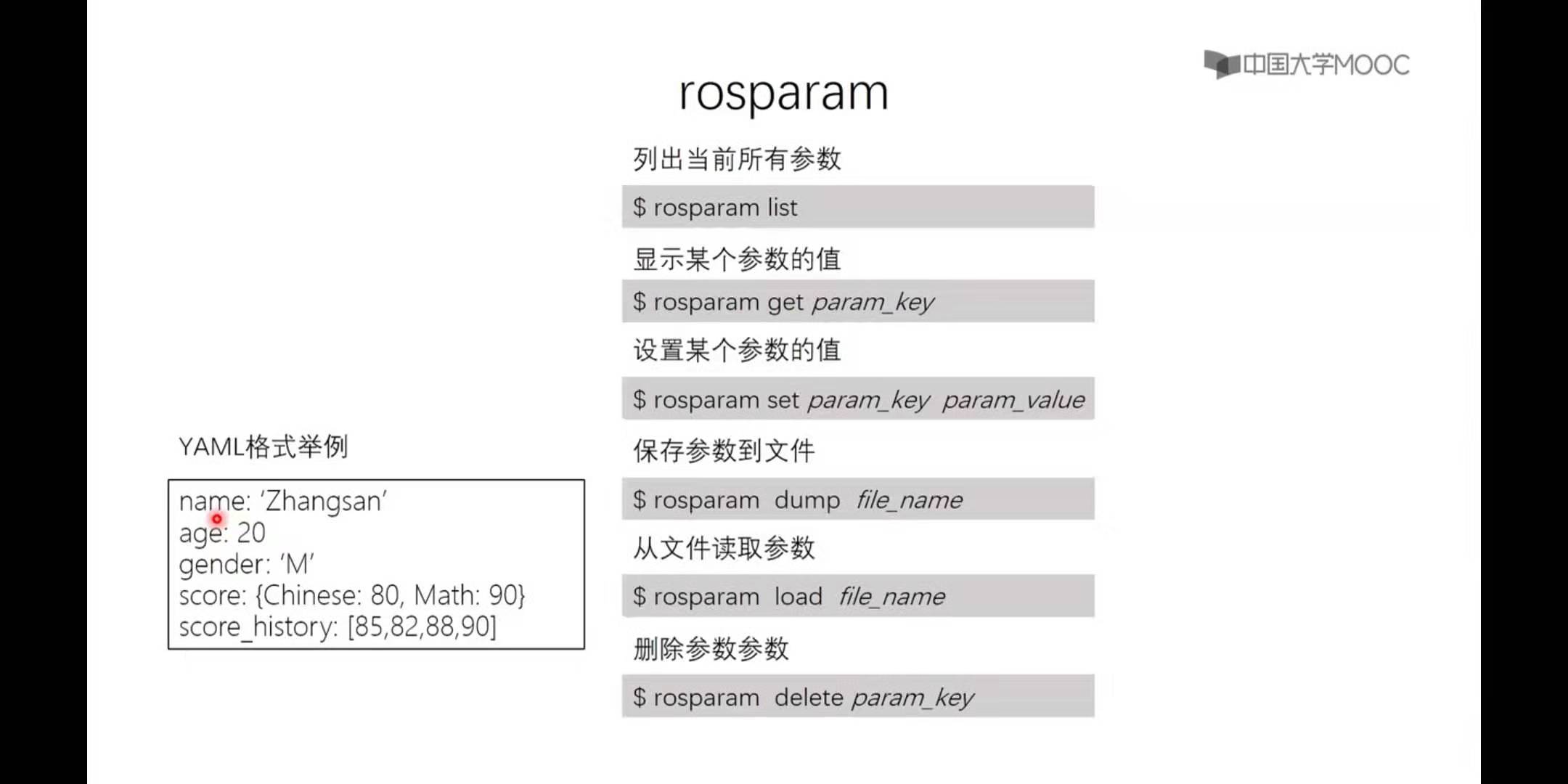
* 一台计算机上的多个终端与多台计算机中的通信的具体方法以及区别？
* 过程中更改功能包名字引发错误

ROS常用命令：先在一个终端使用roscore启动master，再在另一个终端运行节点等其他命令

实例代码：<https://github.com/lazyparser/ROS-Academy-for-Beginners>

* roscore：启动ros master，master-节点管理器，rosout-日志输出，parameter server-参数服务器
* rosrun 功能包名（cmakelists.txt中project） [node\_name]（cmakelists.txt中add\_executable）：运行某个node，程序里定义的节点名在rqt\_graph时显示，另外文件名只是在建立可执行文件add\_executable（）中使用，一个project可以包含多个可执行文件。
* roslaunch [pkg\_name] [file\_name\_launch]：启动master和多个node
* rosnode list：列出当前运行的node信息
* rosnode info[node\_name]:显示某个node的详细信息
* rosnode kill[node\_name]：结束某个node
* rqt\_graph：显示节点图，注意只显示运行的，并不实时刷新，需运行节点后再运行。
* apt-cache search ros-kinetic：显示ros功能包列表



* Rospack：获取功能包信息
* 使用rospack list命令或“rospack find 包名“命令确认是否安装了必要的功能包。
* roscp：拷贝功能包文件
* rosed：编辑功能包文件
* ros::console::set\_logger\_level(ROSCONSOLE\_DEFAULT\_NAME,ros::console::levels::Debug): ROS代码中设置消息日志级别 <https://blog.csdn.net/qq_35508344/article/details/77720249>
* stamp：ros::Time类，使用toSec()、toNSec()成员函数得到秒、毫秒，而非.sec、nsec注意大小写

<https://blog.csdn.net/vingsony/article/details/77249021>

<https://blog.csdn.net/u013834525/article/details/83863992>

* rosbag：最后

<https://www.cnblogs.com/yuanlibin/p/9420167.html>

1. 自己总结

* 可以开多个键盘控制节点，激活哪个节点控制哪个节点
* Python文件可以作为执行程序直接运行，但是在运行前，必须使用系统命令为文件添加可执行权限：sudo chmod +x XXX.py
* Kinect开源linux驱动包：openNI（openni\_camera）和freenect(freenect\_camera)，书中使用
* 可以使用plugin来拓展ROS的功能包，设置一次以后可以通用
* Qt是编写rviz的工具，也可以用来编写rviz插件
* SMACH状态机可以实现状态机
* 如何判断IP在同一网络：IP网络标识一样，网络标识=IP and 子网掩码
* 可以参考已有ROS机器人实例PR2、Turtlebot、universal robots、catvehicle（无人驾驶）、HRMRP（笔者自己设计，有详细介绍）、kungfu arm提供的功能包
* 包含头文件和第三方库：顺序不能错，得先链接库，编译cpp才有用

<http://www.pianshen.com/article/1911233795/>

* 话题的变量类型，可能在std\_msgs？中：string、uint8/16/32、int8/16/32、float8/16/32、？double8/16/32,后面加[]可变成数组
* ros1-kinetic常用消息链接,话题查找方法如，ros-wiki-package-kinetic-nav\_msgs(ros1)
* std\_msgs：[https://index.ros.org/p/std\_msgs/github-ros-std\_msgs/#kinetic-assets](https://index.ros.org/p/std_msgs/github-ros-std_msgs/" \l "kinetic-assets)
* geometry\_msgs：[https://index.ros.org/p/geometry\_msgs/github-ros-common\_msgs/#kinetic-assets](https://index.ros.org/p/geometry_msgs/github-ros-common_msgs/" \l "kinetic-assets)
* nav\_msgs: <https://index.ros.org/p/nav_msgs/github-ros-common_msgs/#kinetic-assets>
* message\_runtime: <https://index.ros.org/p/message_runtime/#kinetic>
* message\_generation: <https://index.ros.org/p/message_generation/#kinetic-overview>
* sensor\_msgs: <https://index.ros.org/p/sensor_msgs/github-ros-common_msgs/#kinetic-assets>
* tf：TransForm即两个坐标系之间转换的意思。注意树形结构，如世界坐标一般作为根节点，机器人上的器件可以相对于机器人坐标转换

1. ROS安装：按照如下官网及链接步骤安装，第二个链接为参考，在官网步骤作了小更改

<http://wiki.ros.org/kinetic/Installation/Ubuntu>

<https://blog.csdn.net/radiantjeral/article/details/82193370>

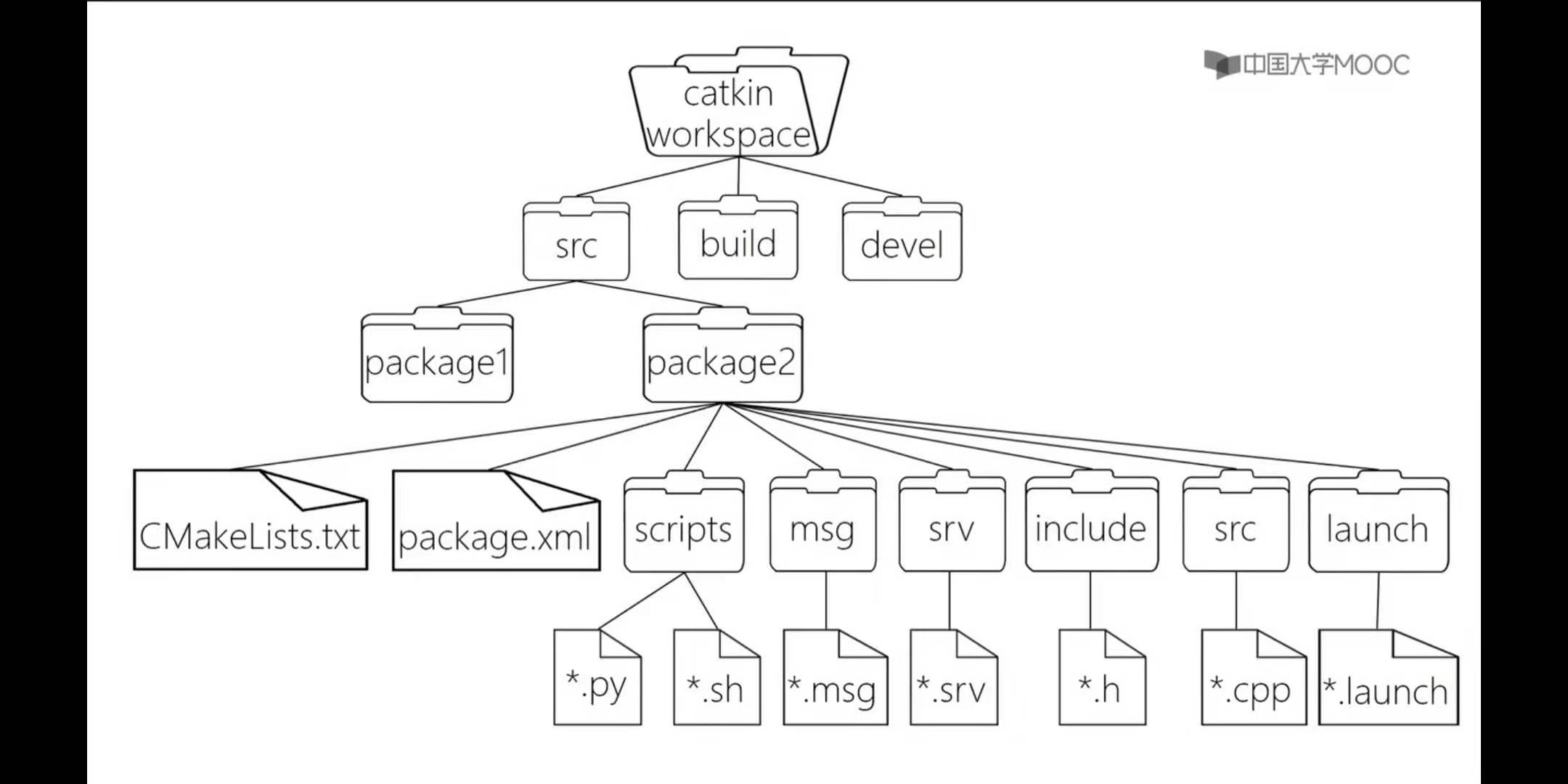
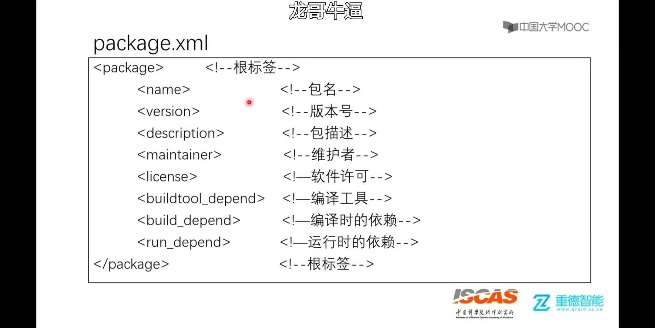
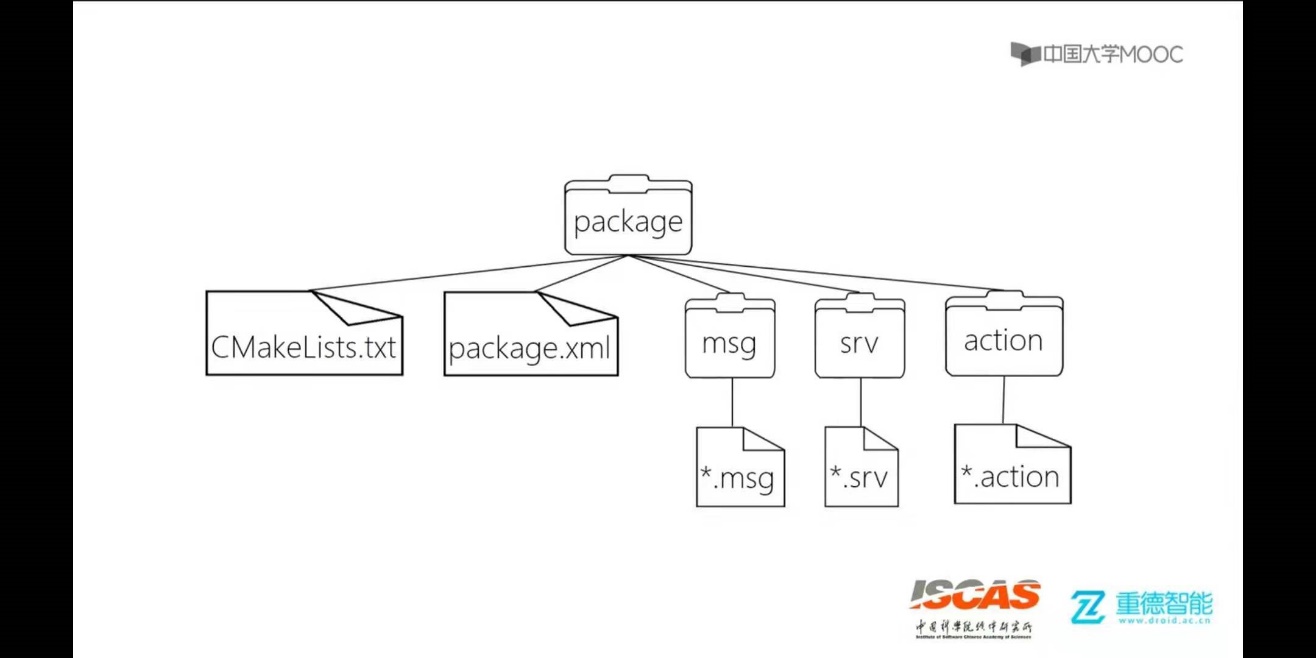
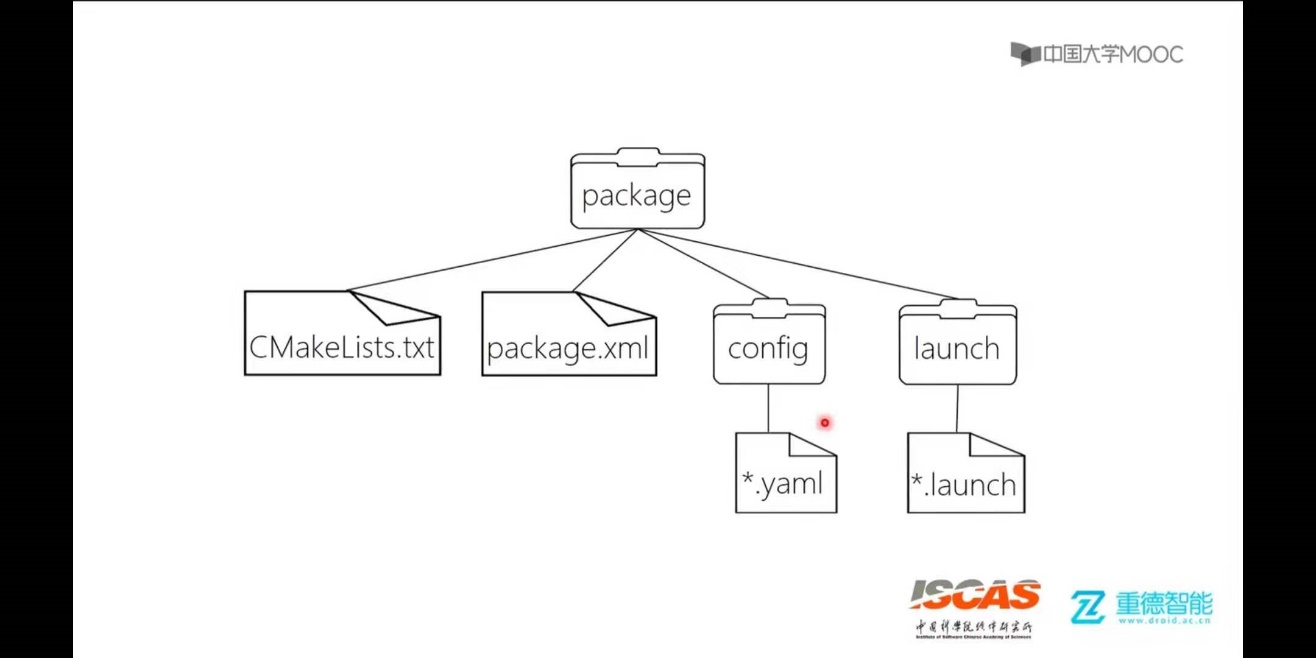
* 先将系统设置-软件和更新-下载自改为清华的镜像，这样更新比较快
* 1.2步使用国内镜像比较快：sudo sh -c '. /etc/lsb-release && echo "deb http://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu/ $DISTRIB\_CODENAME main" > /etc/apt/sources.list.d/ros-latest.list'
* 1.4增加：sudo apt-get upgrade
* 最后使用roscore命令判断ros是否安装成功

1. roboware安装（官网打不开在链接中网盘下载）：

<https://blog.csdn.net/zhuiqiuzhuoyue583/article/details/84536406>

安装pylint报错：未解决：sudo pip install pylint==1.9.3

建议参考官方网站使用手册快速上手

1. ROS工作空间结构：package（功能包pkg，自己命名）中存放多个可执行文件（node，名称必须唯一，一个节点相当于一个可执行文件，类似于C++ 一个main函数，相同节点可以通过不同名称来启动），可以多层嵌套，是catkin编译的基本单元，其中cmakelists.txt规定编译的规则，如源文件、目标文件、依赖项，pakage.xml定义package属性，包名、版本号、作者、依赖（具体如下图），src还可包含python module；build为缓存和中间文件；devel为目标文件；另外，还有install文件，功能和develop差不多，用于保存可执行文件，其并不是必须的，很多工作空间可能没有这个文件
2. package下自定义的通信格式有消息（msg）、服务（srv）、动作（action）
3. package文件下还有：launch文件出发多个可执行程序，文件如下图，配置文件（yaml）
4. 创建ROS工作空间(类似IDE工程)，catkin是ROS编译工具，node是可执行进程，在其上面设置

创建工作空间

mkdir -p ~/catkin\_ws/src

cd ~/catkin\_ws/src

catkin\_init\_workspace

编译工作空间

cd ~/catkin\_ws/

catkin\_make

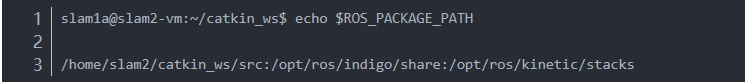
catkin\_make install //产生install

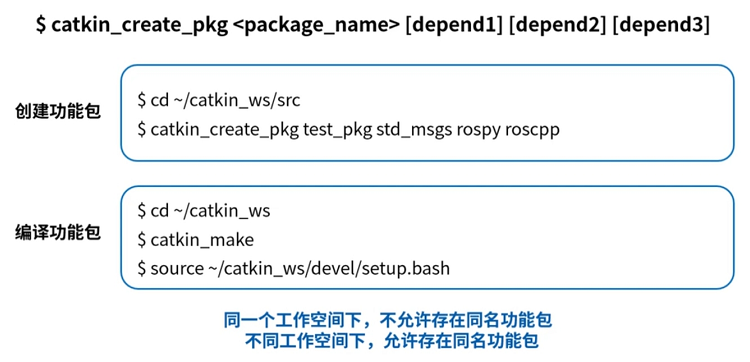
设置环境变量：设置之后才能让系统找到工作空间和环境变量

source devel/setup.bash

检查环境变量，确保已生效

echo $ROS\_PACKAGE\_PATH

显示如下结果表示已经配置好

创建功能包：不允许某个功能包中嵌套其他功能包，只能平行放置

编辑完源文件、头文件和cmakelist.txt文件后再回到catkin\_ws使用catkin\_make编译便能生成源文件，便能生成节点，catkin相当于在cmake上又加了一层编译为ros节点的功能，注意使用source命令

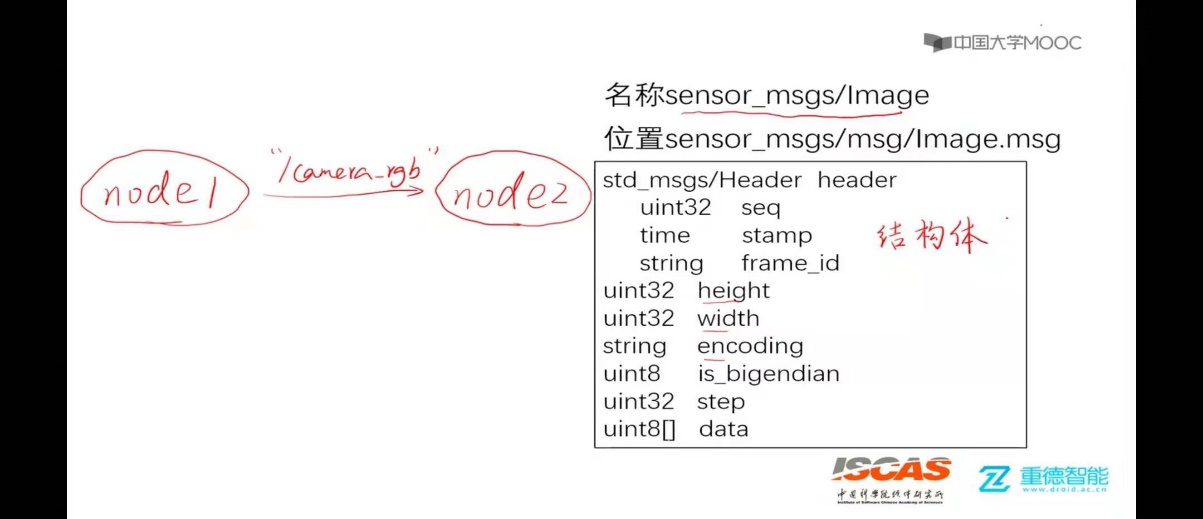
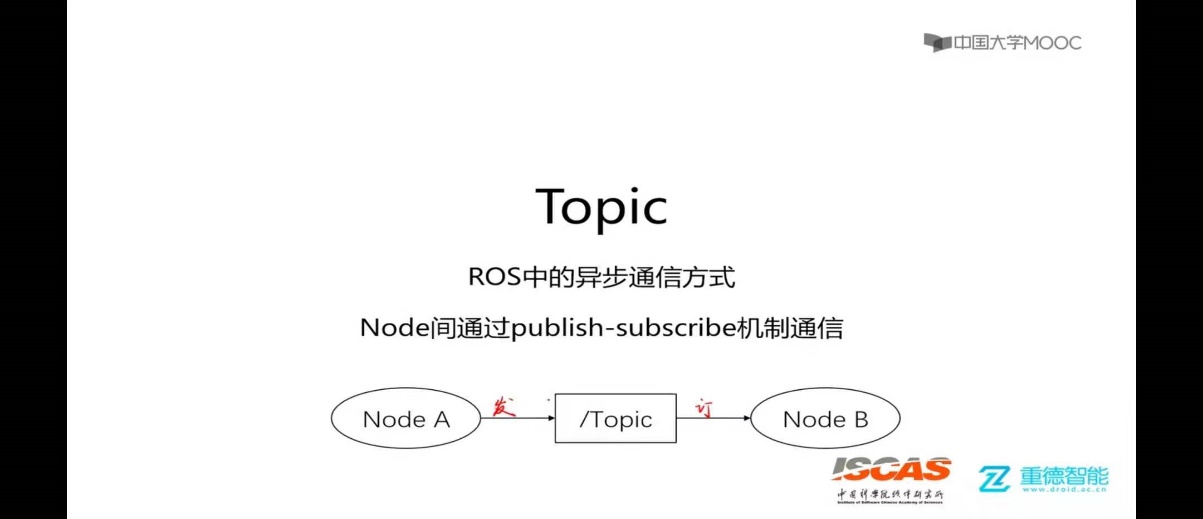
所有工作空间的路径会依次放在ROS\_PACKAGE\_PATH记录中，source后会将当前工作路径放在最前面，运行时会从前到后查找功能包

1. 官方功能包（package）：控制、规划、视觉、建图

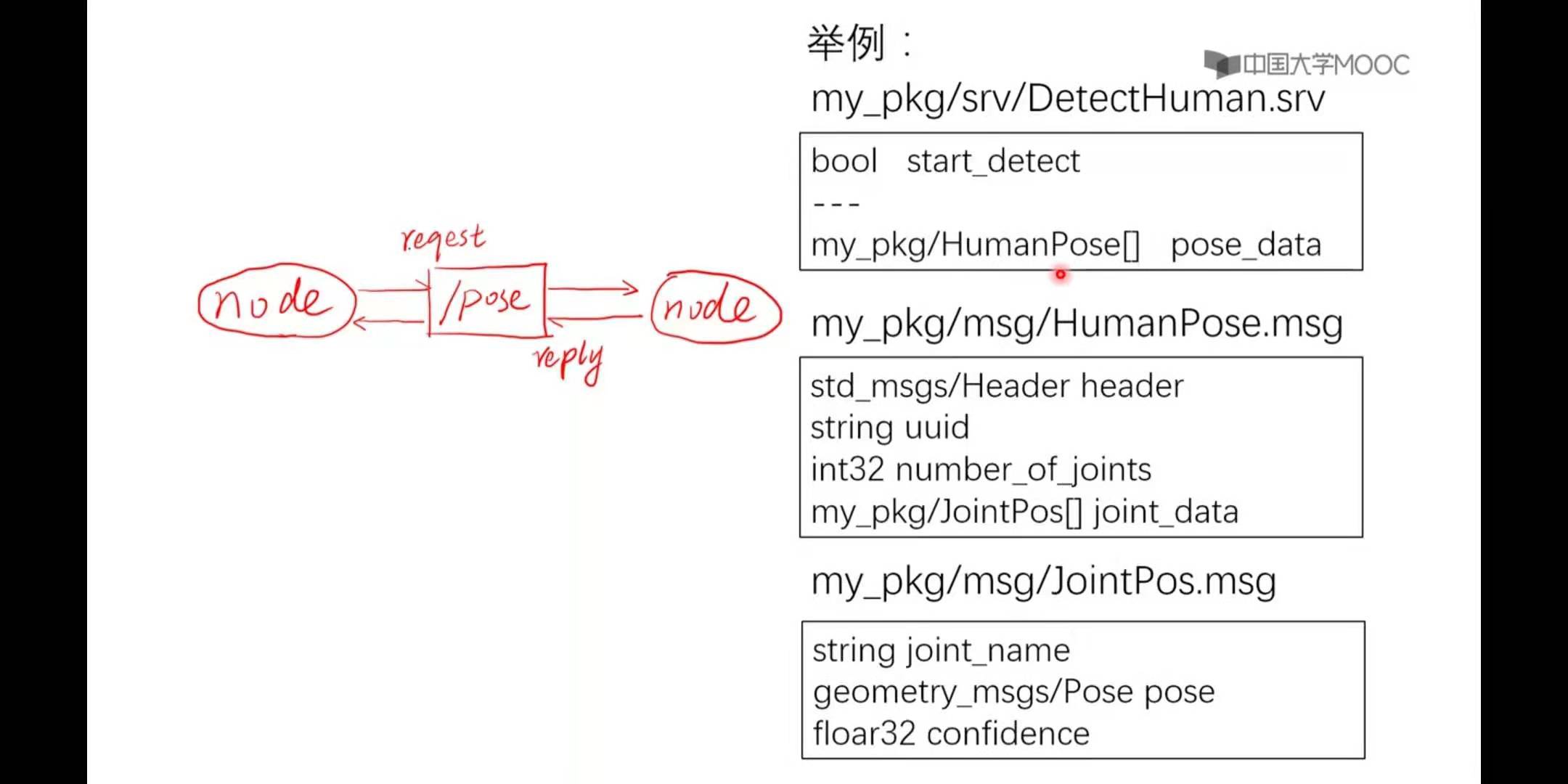
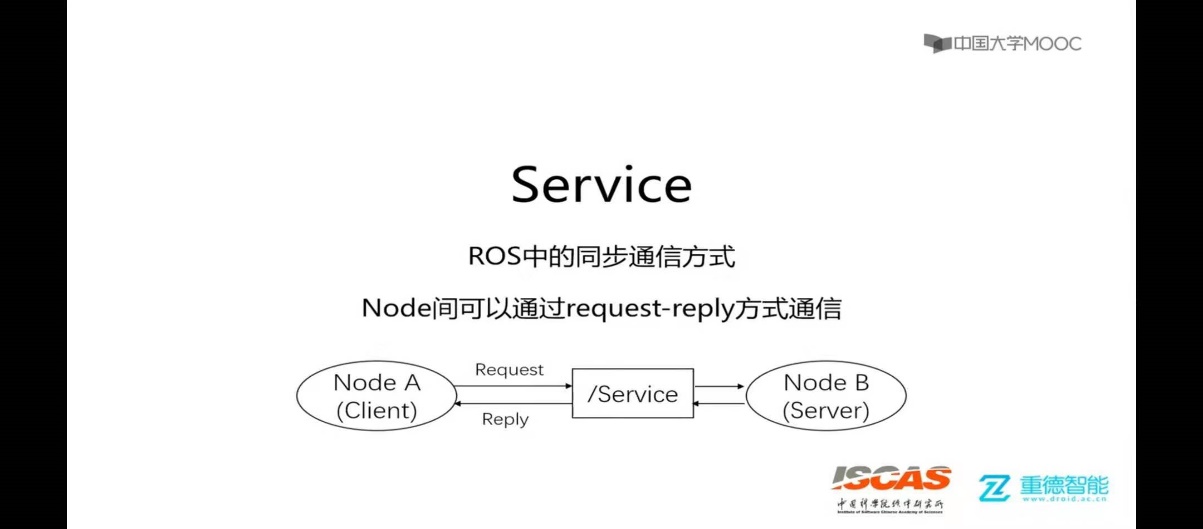
功能包安装方法; 器人功能包可以安装ROS官方功能包，或按照维基中说明的安装方法从开源存储库中下载机器人功能包，然后经过构建过程之后就可以使用

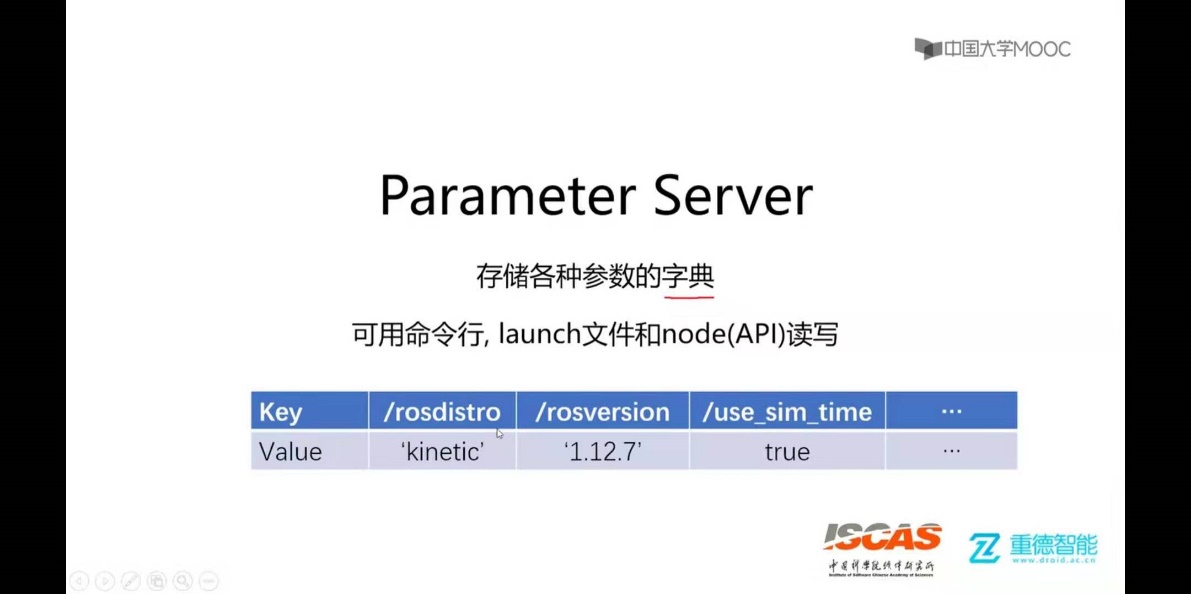
1. pr2机器人简介
2. 通信方式有：topic、service、parameter server、actionlib。中间方框表示他们不是直接交换，而是通过master交换。可以将master、node运行在不同计算机上，通过RPC通信。节点建立连接之后，可以关掉ROS master，节点之间的数据传输不会受到影响，但是其他节点无法加入这两个节点之间的网络

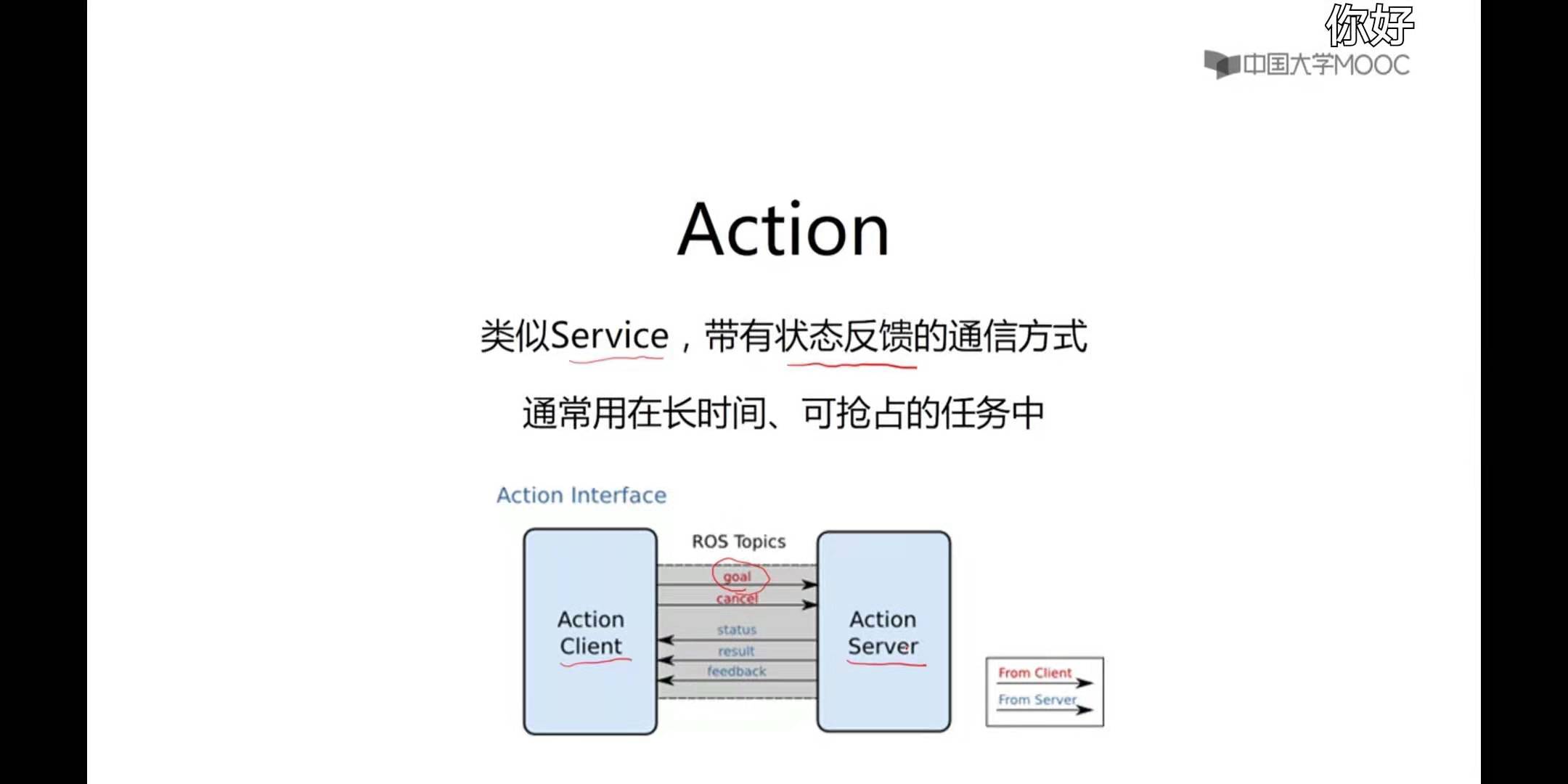
* topic（指中间部分，整个过程称为消息message）：单向通信，描述如左图， message是其数据类型，定义在\*.msg文件中，如右图。如string、uint8是语言无关的，编译时会变成各种语言对应的数据类型



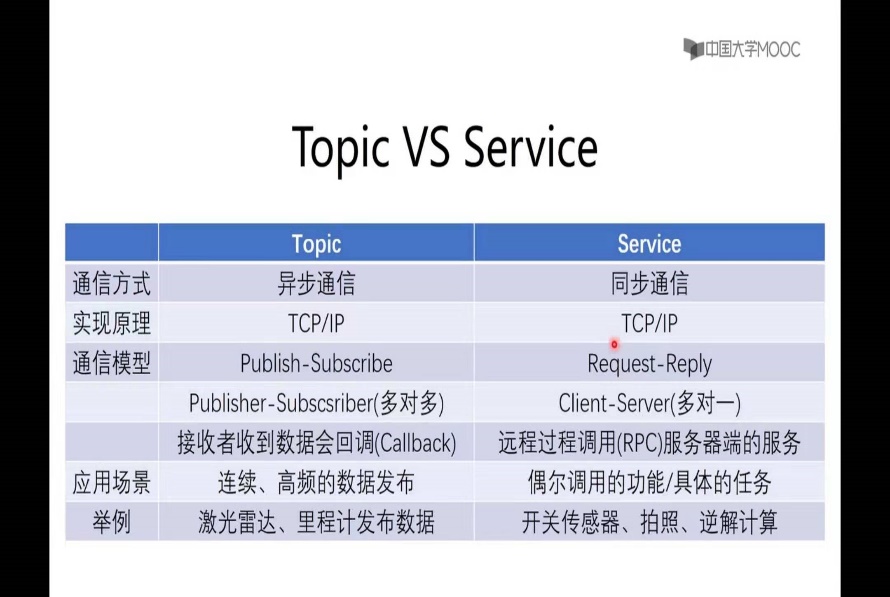
* service：双向通信，描述如左图， srv是其数据格式，定义在\*.srv文件中，如右图

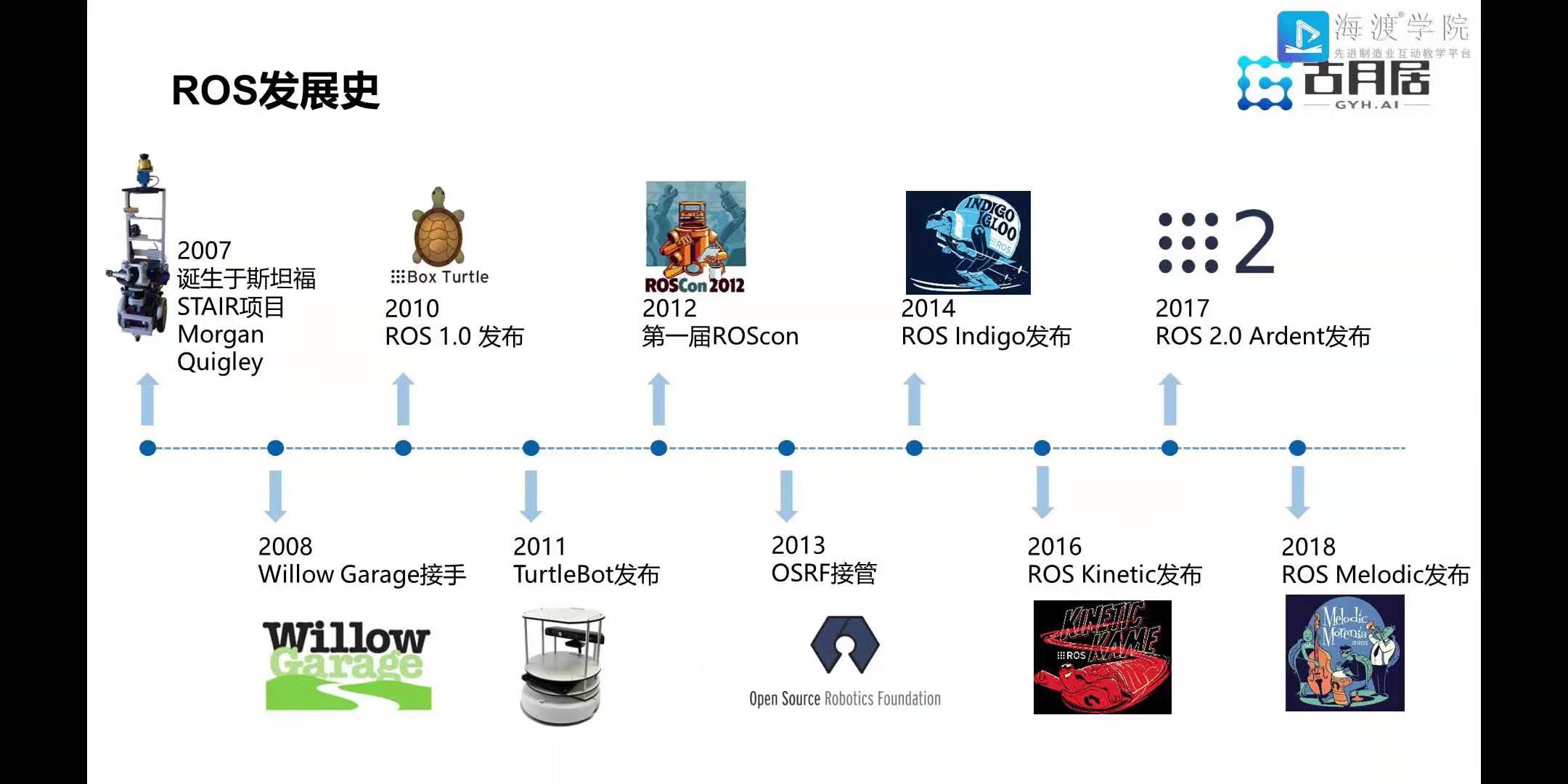


* parameter server：master的参数服务器配置，适合改动很少的信息传输，其描述如下图，一般用yaml文件设置，适合静态、非二进制配置参数，不适合动态配置参数
* action：如控制电机动作，控制信号为pwm波，状态反馈为霍尔传感器信号，定义在.action。action类似service，不同之处在于action带有连续反馈，可以不断反馈任务进度，也可以在任务过程中中止，适合控制数据较少的控制信号



1. topic和service不同



1. ROS发展过程：ROS是中间件/类操作系统，包括硬件抽象、底层设备控制、常用函数实现、进程间消息传递、包管理，由通信机制或框架（分布式、进程管理、进程间通信）、开发工具（仿真、数据可视化、图形界面、数据记录）、应用功能（控制、规划、视觉、建图）、生态系统（ROS官网（邮件列表、answer、wiki）、社区等，软件包管理、文档、教程）四部分组成。

* 优点：节点容错性强；不同语言模块隔离；模块开发低耦合
* 缺点：过度依赖master节点（增加master镜像）；node节点异常问题；计算资源浪费；消息数据未加密

1. 开发环境、常用组件

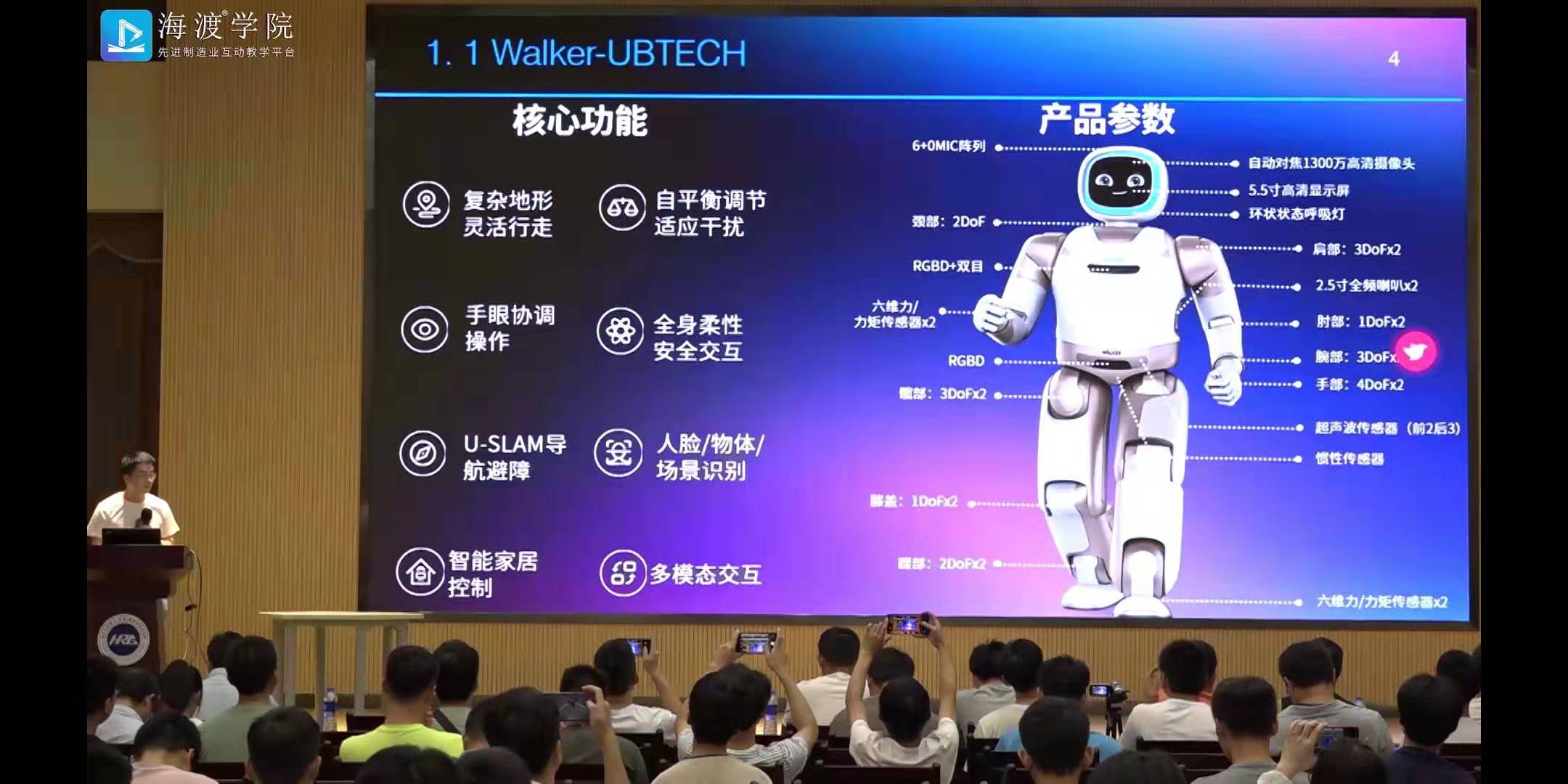
* 适用于ROS的IDE：roboware、eclipse
* 建模与仿真：可以看到物体及环境的3D环境，还有能控制机器人运动及使用传感器感知周围环境。URDF文件创建一个机器人模型，然后使用xacro文件优化该模型，xacro需要手动或自动转化成URDF使用，并且放置在rviz+arboti或gazebo环境使用
* URDF（Unified Robot Description Format，统一机器人描述格式）：编程式建模，是描述机器人模型的XML格式文件，ROS提供URDF的C++解析器。
* xacro：一种特殊的URDF，提供了更高级的方式来组织编辑机器人描述，优化模型代码（通过宏定义复用代码）、提供编程接口，是URDF精简化、可复用、模块化的描述形式。
* sw2urdf：solidworks模型通过插件自动生成URDF模型。
* rviz：可视化显示URDF模型、显示所有检测信息、通过滑动条、数值等方法控制机器人。
* rqt：可视化调试和显示。包括日志输出工具、计算图可视化工具数据绘图工具、参数动态配置工具。
* gazebo：动力学建模，模型完善，各种传感器模型。模型（几何模型）与rviz模型相同，但是需要在模型中加入机器人和周围环境的物理属性，例如质量、摩擦系数、弹性系数等。模型下载网站（<https://bitbucket.org/osrf/gazebo_models/downloads/>）
* airsim：适合无人机、高度还原真实场景，方便视觉的强化学习训练。
* arbotix：是一款控制电机、舵机的控制板，rviz+arbotix的传感器无法获取环境任何数据。
* ros\_contorl：控制学，提供了机器人控制中间件，封装了多种类型的控制器接口，传动装置接口、硬件接口、控制工具箱等，统一数据通信接口，多机器人硬件资源进行了抽象。
* ros+matlab+V-REP：MATLAB（2013之后）的robotics system toolbox提供了ROS的大部分功能，可以与ros通信，matlab可以结合V-REP进行仿真。
* rosbag：数据记录与回放，如记录摄像头或激光走完一条轨迹的数据之后，可以通过回放来仿真，而无需重复走同样的轨迹。
* 命令行工具：rostopic（可以模拟node发送信息）、rosbag
* 专用工具：moveit
* 学习平台：AI大学（机器人专栏）、ROS小课堂（网站+公众号）、古月居（公众号+网站）、阿木实验室、openAI(openai\_ros package: <https://wiki.ros.org/openai_ros>)
* 开发板：树莓派、jetson nano、jetson TX2、miniPC、工控机（如精锋微控SOC的机器人驱动系统）、arduino
* 具有代表性机器人：NAO机器人、pepper、marmorf、yanshee人形机器人、PX4无人机开源飞控（阿木实验室）

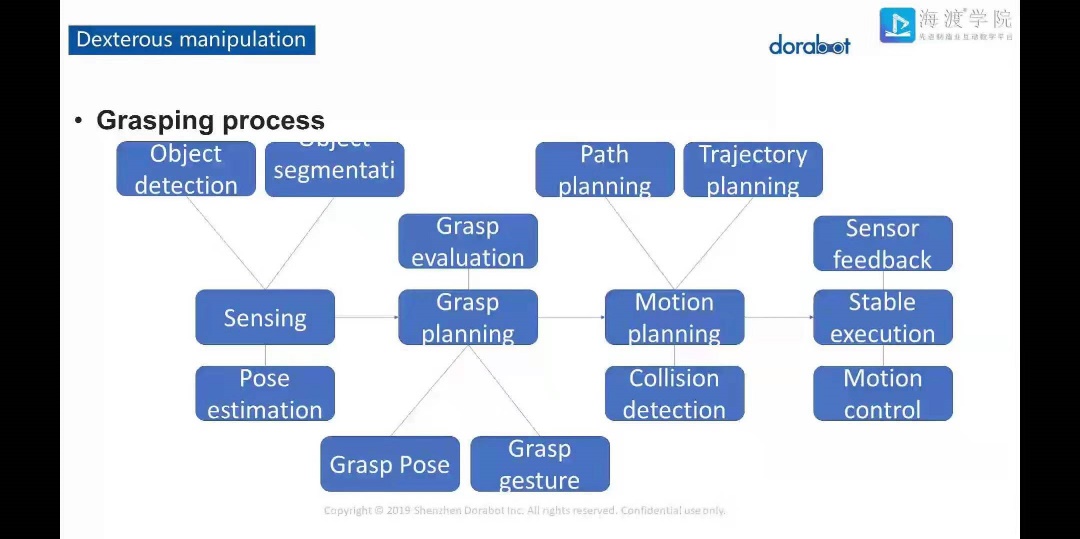
1. 机器人结构分析（以人形机器人为例，其相对复杂）

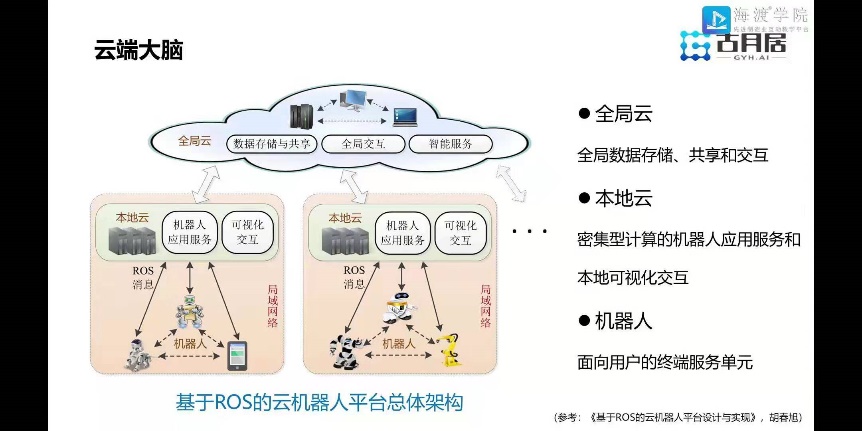
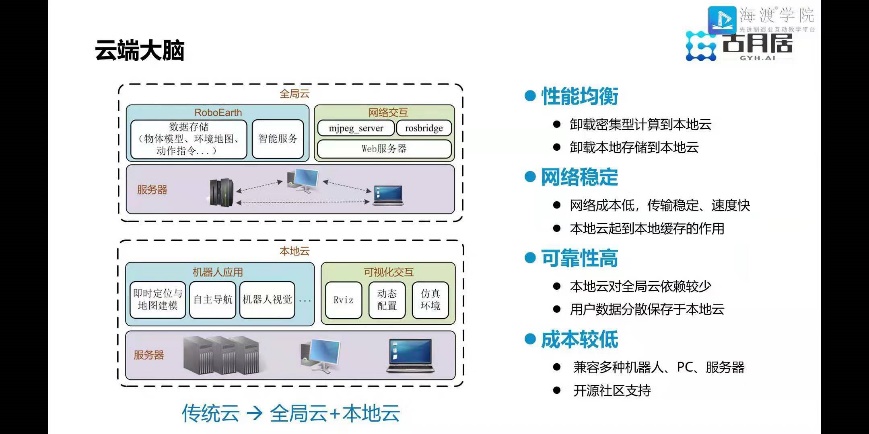
应用层：SLAM、路径规划、计算机视觉、自然语言处理、运动学、动力学等

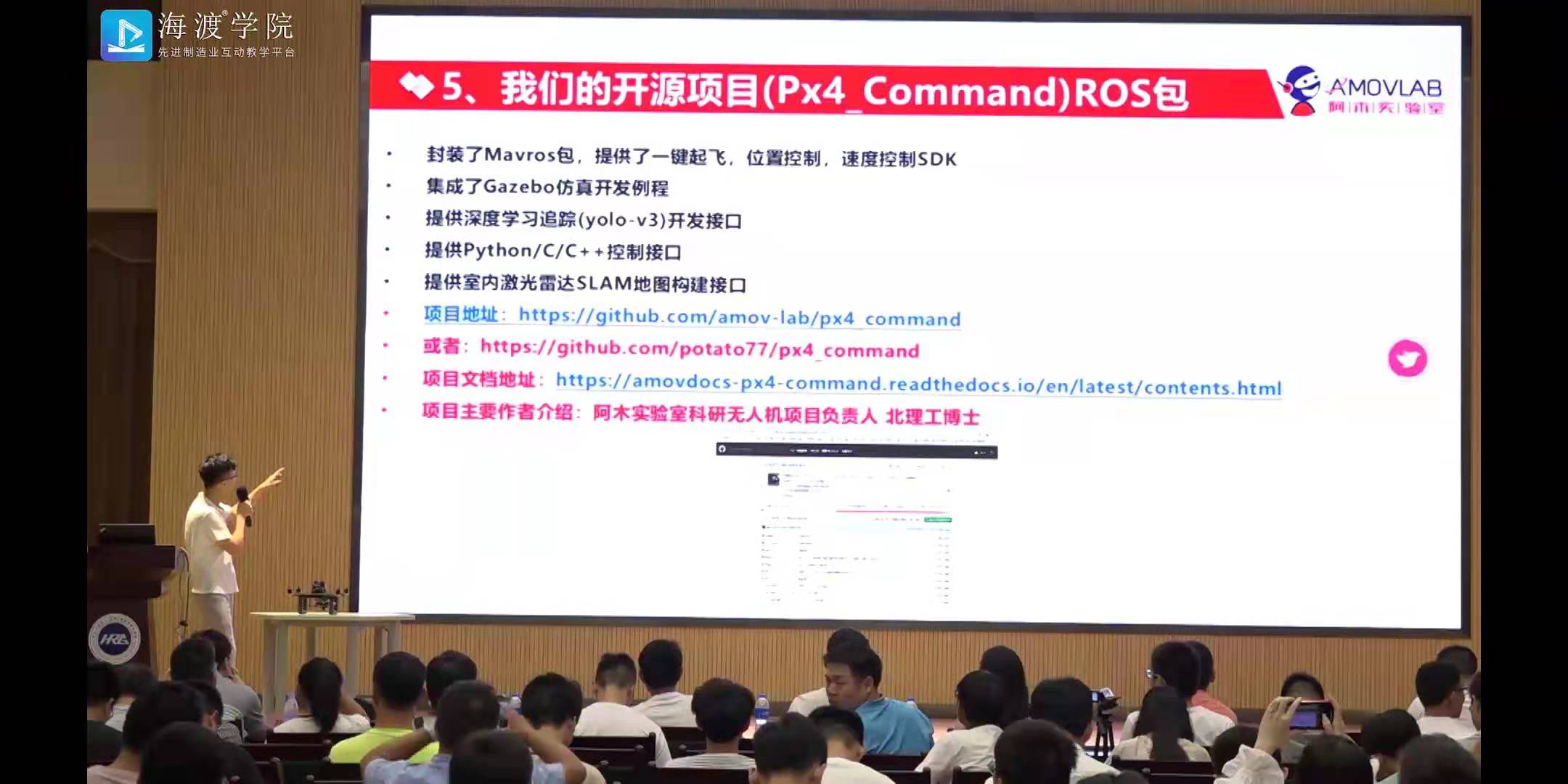
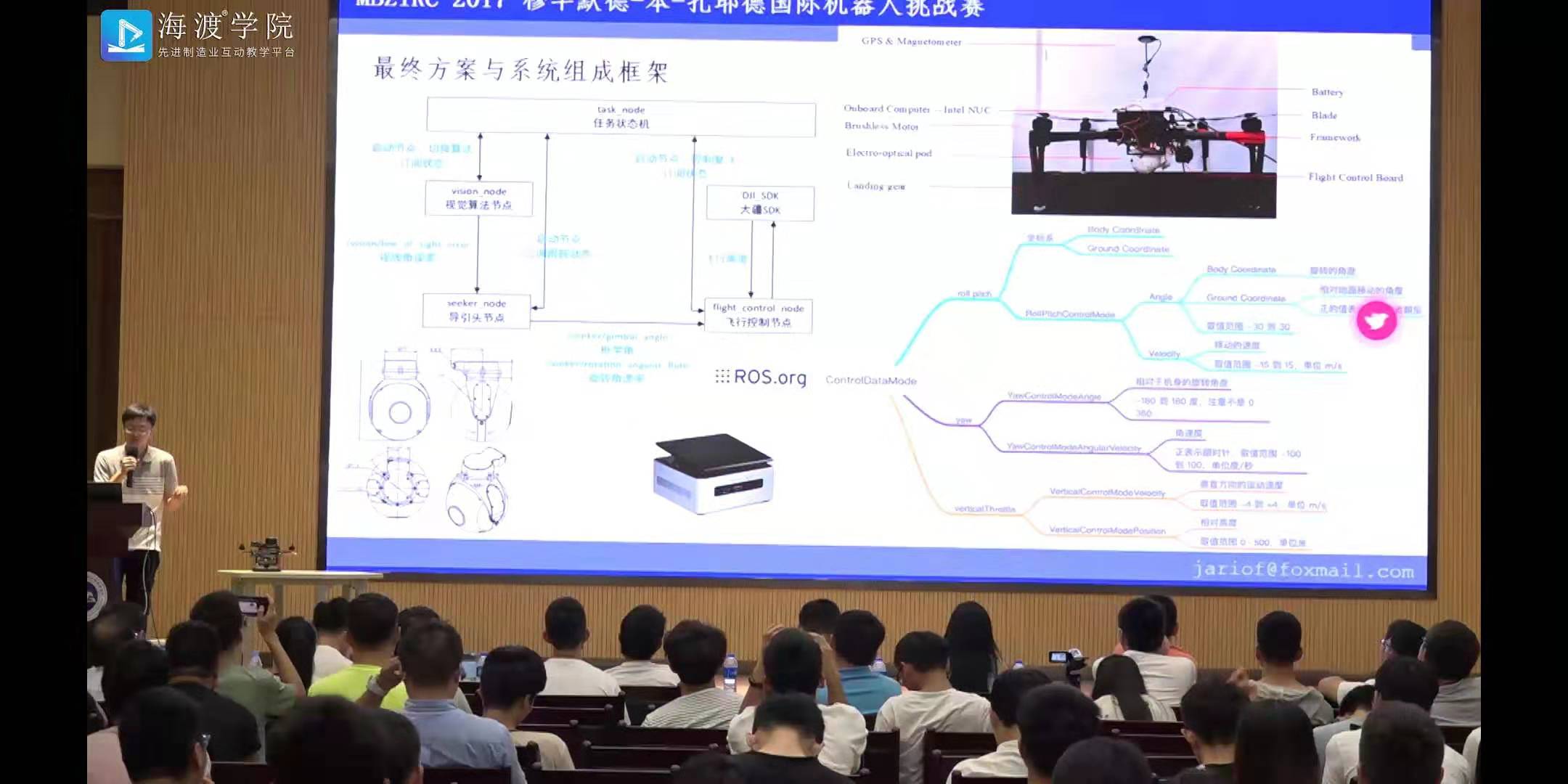
控制层：工控机、miniPC、树莓派、Jetson Nano、Jetson TX2等

驱动层：STM32单片机、FPGA、DSP、arduino等

执行层：相机、激光雷达、里程计、姿态传感器、激光、超声波、舵机、伺服电机、步进电机等

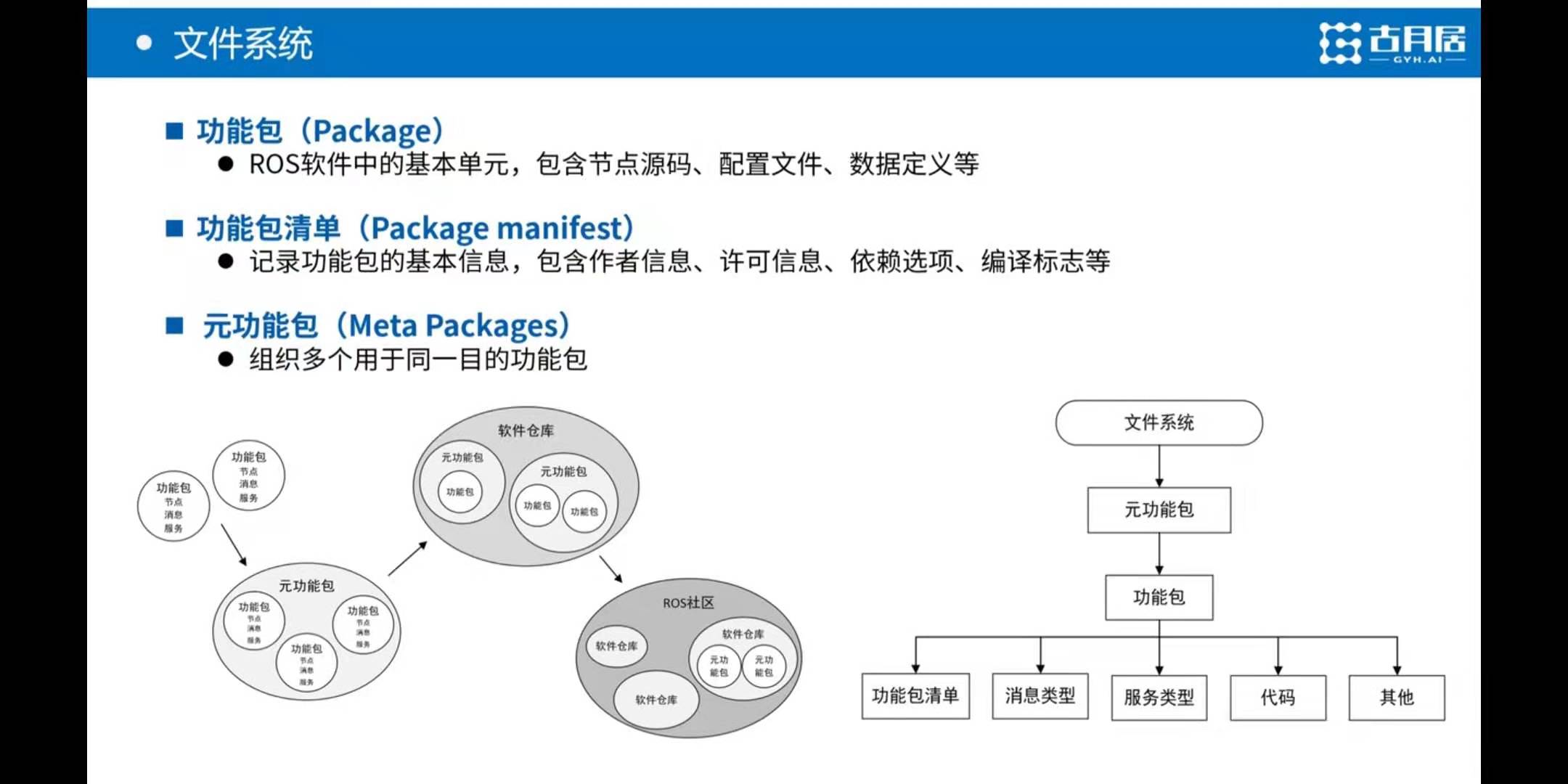
1. 抓取过程
2. 云端大脑（全局云+本地云）：也就是将一些常识性知识存储在云端，供所有机器人访问，如香蕉是什么样的



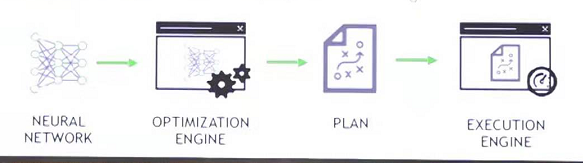
1. 空间定位：GPS-32(卫星数)、北斗2-18、北斗3-35、galileo-24、glonass-24。可以通过增加卫星数和通过基站增加测量精度，也可通过算法增强（菲曼公司）
2. 阿木实验室：深度学习追踪yolo-v3，比赛方案如下图
3. PX4：无人机领域的安卓，网站如下

<https://px4.io/>

<https://github.com/PX4/Firmware>

1. 功能包（package）简介：元功能包如导航元功能包包括建模、定位、导航
2. deepmind：openAI
3. 利用tensorRT加速推理实践：<https://github.com//NVIDIA/TensorRT>

* 优化方法：weight&activation precision calibration、layer&tensor fusion、kernel auto-turning(根据不同层和算法选择合适的GPU)、dynamic tensor memory（动态重复利用tensor内存，使内存更高效）、muti-stream execution
* 优化步骤：tensorRT会根据网络结构生成PLAN，然后GPU会根据这个plan来训练及推理，PLAN可以在相同实体中使用，不需要重复生成

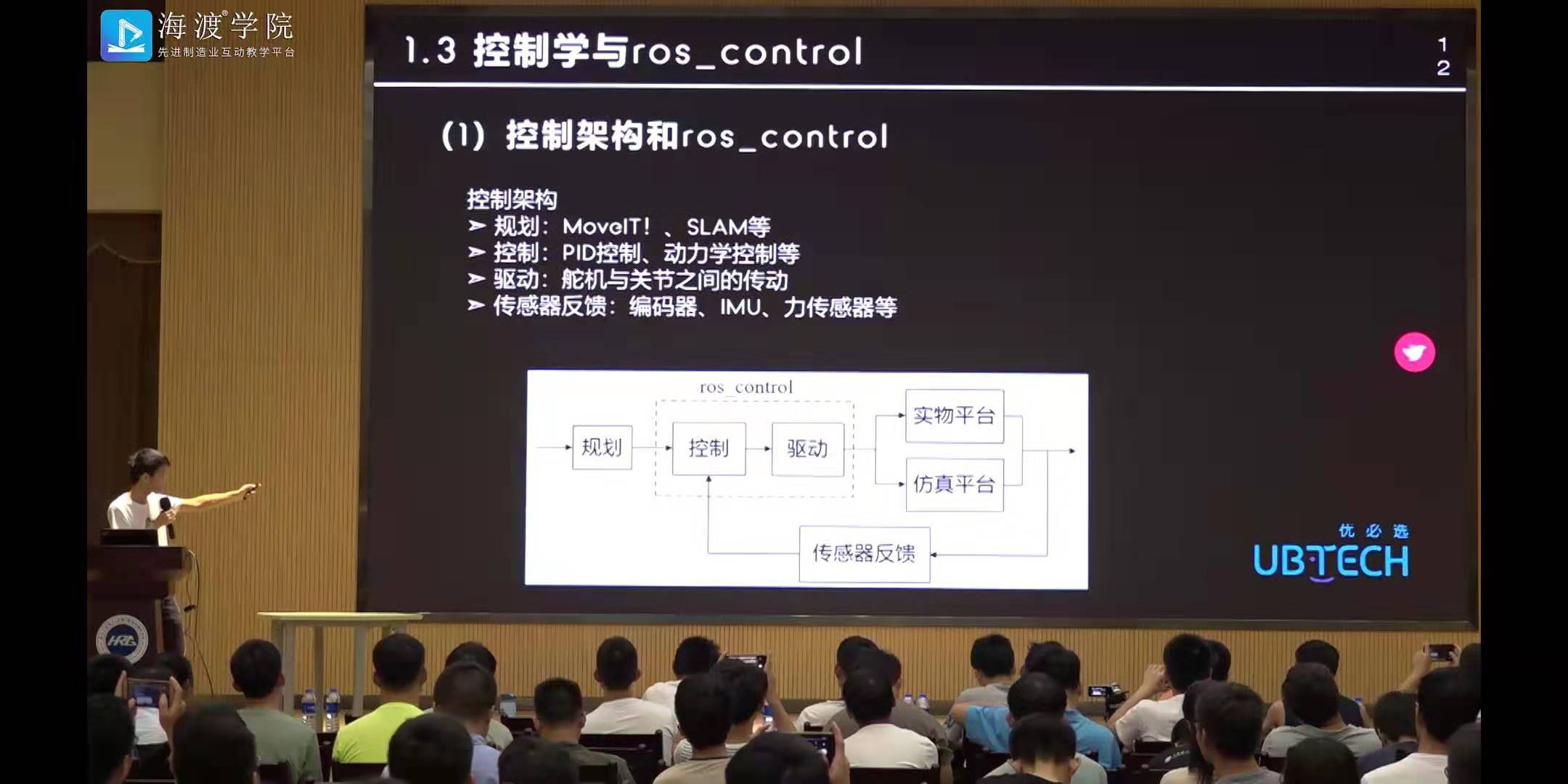


* 实现步骤

1. 平衡控制：引入平衡控制的原因是建模不准确及外部扰动

* 常用方法是“三段式”控制原理，姿态控制、高度控制、前进速度控制。
* 基本思想：把ZMP点控制在支撑区域内

1. 控制框架如下图

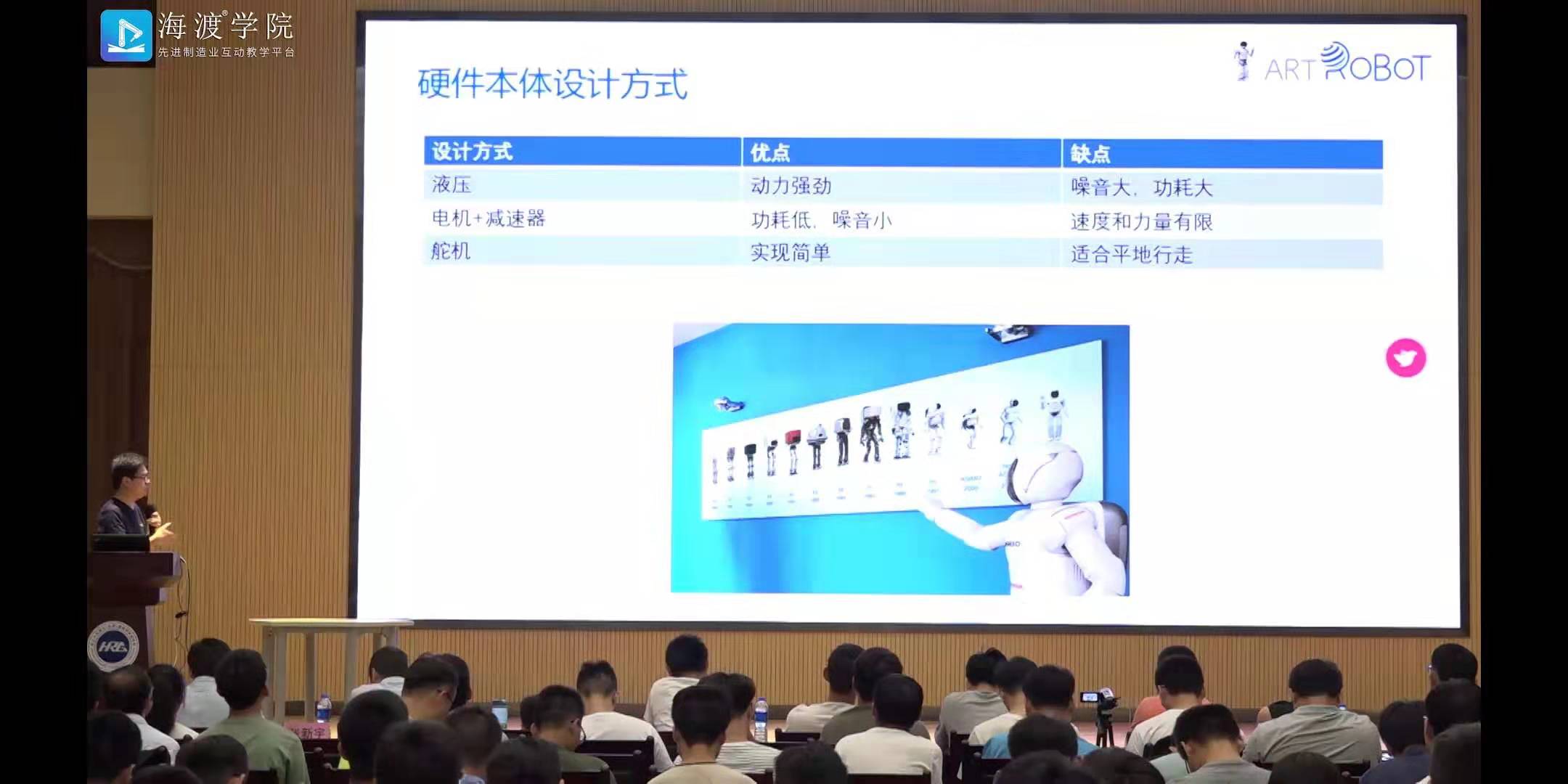
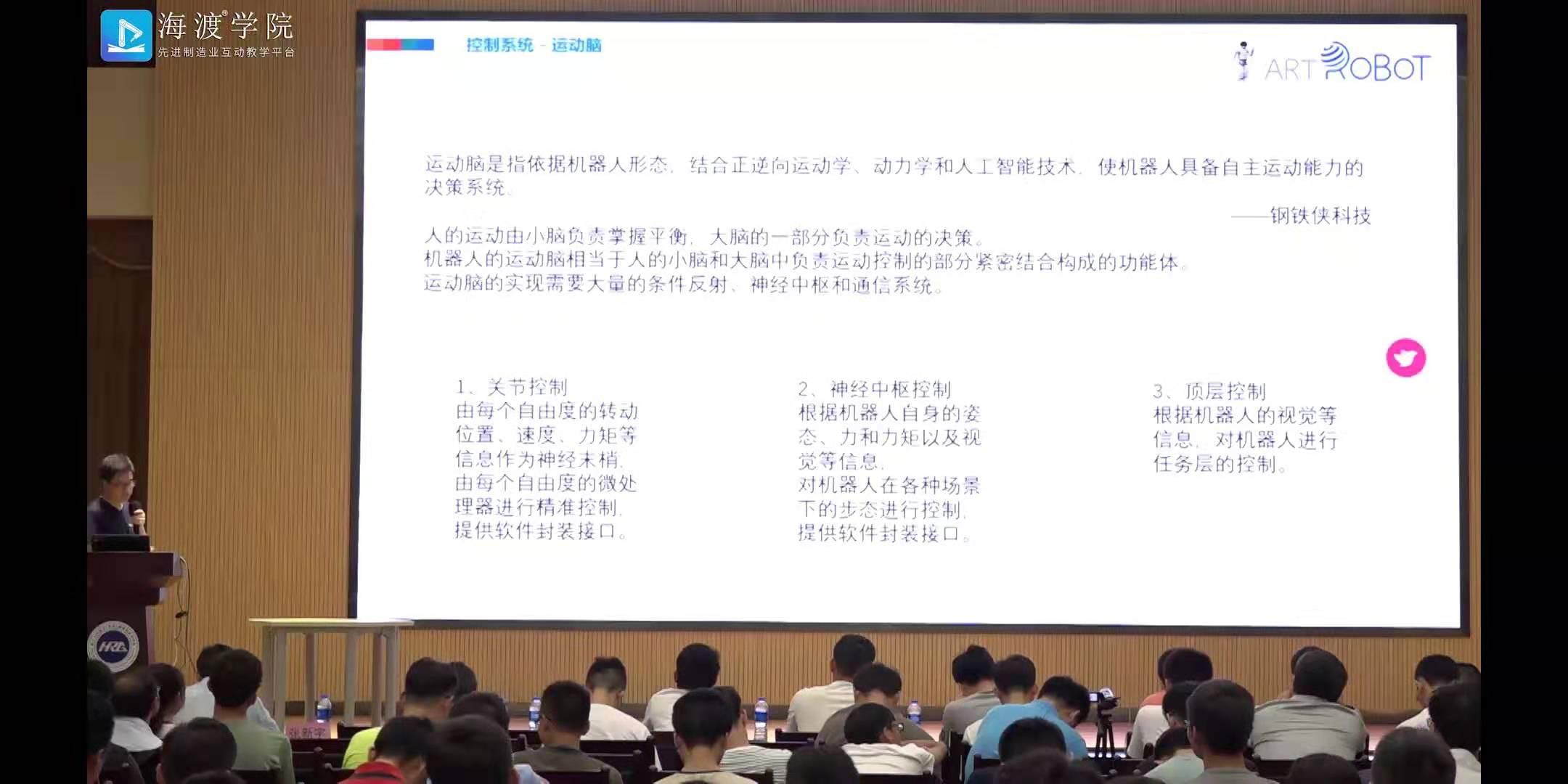


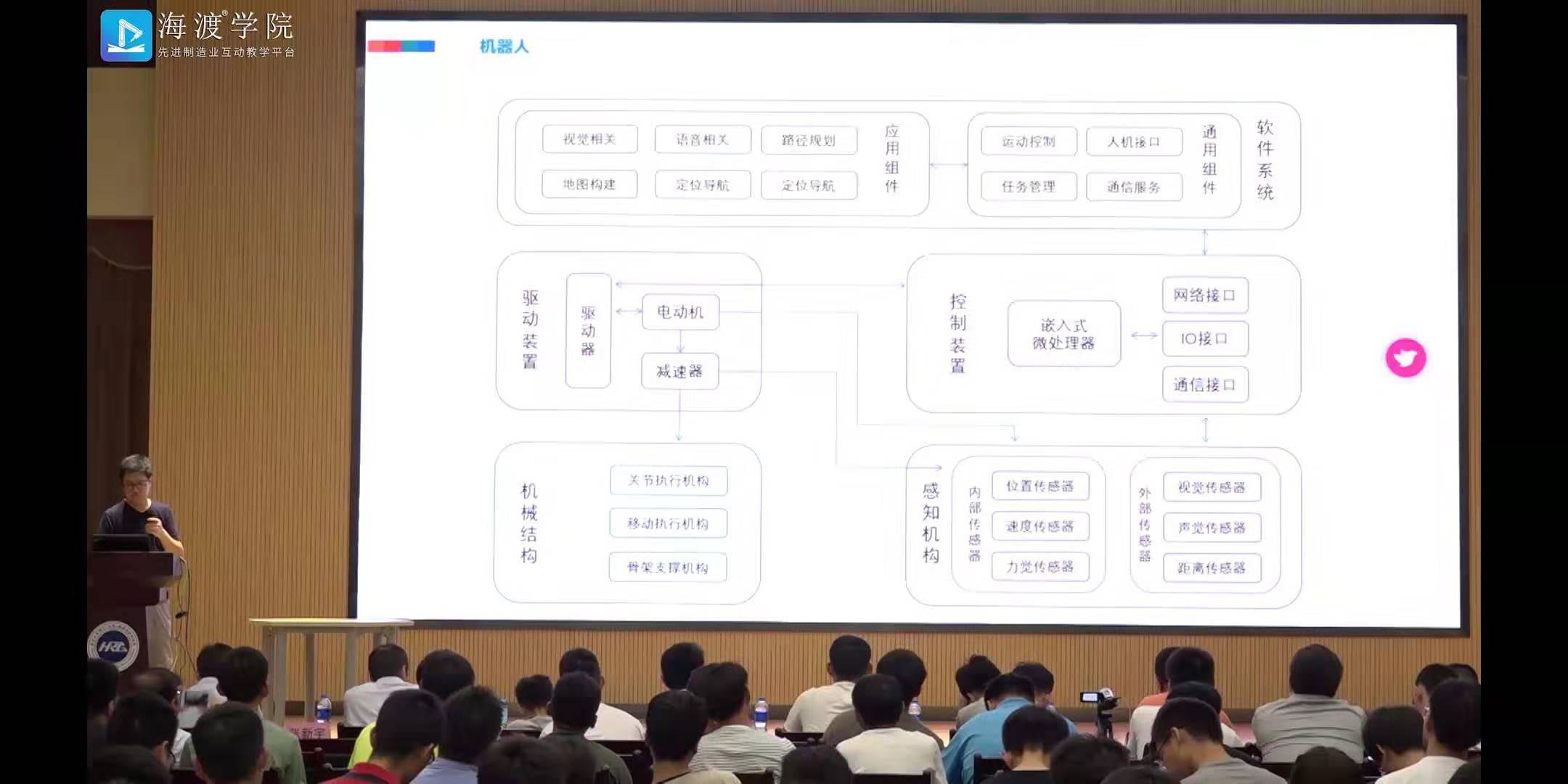
1. sw2urdf使用方法

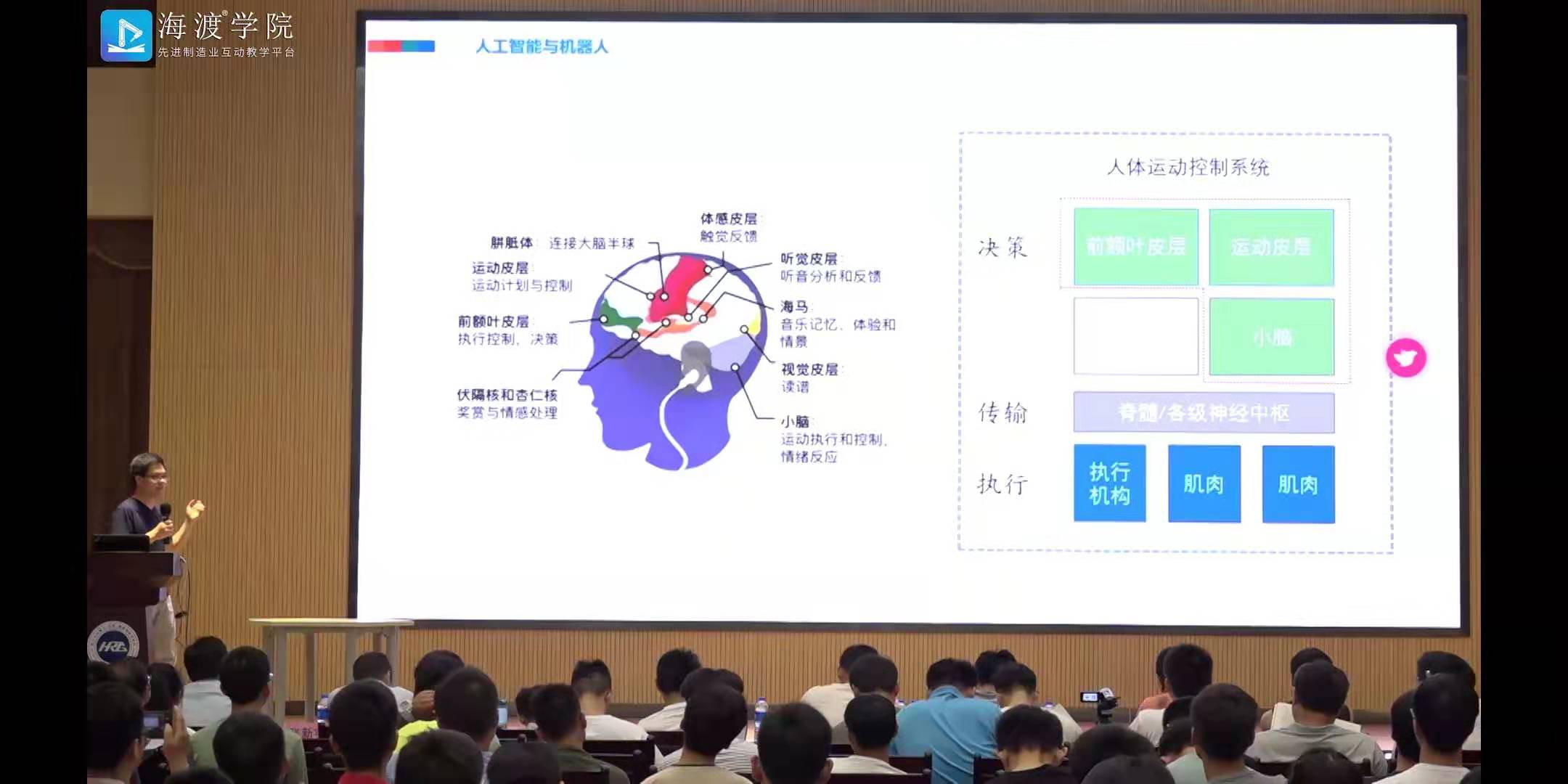


1. 优必选及钢铁侠推荐书籍及学习资料



1. 各种动力优缺点
2. 钢铁侠仿人机器人总结



1. ROS1主要构建于Linux系统之上，ROS2带来了改变，支持构建的系统包括Linux、Windows、Mac、RTOS，甚至没有操作系统的裸机。也可以使用macOS、arch、debian
2. RTOS(是统称，其核心是实时内核，不像前后台系统，规定的循环方式): FreeRTOS（免费，用得较多）、 µC/OS-II（收费）、 RThread、 WindowsCE、 VxWorks
3. ROS的三层结构包括：OS层、中间层（TCP/UDP（分布式主机）、nodelet API（进程内）、clientlibrary）、应用层（mater、node（自己编写））
4. PC+嵌入式系统方案：PC-SLAM、导航、图像识别、语音识别；嵌入式系统：外部传感器数据采集、与主控板通信、其他外设连接
5. 1

《机器人未来简史》-李鹰译