Dofbot大作业说明

电子信息与电气工程学院自动化系

2024年10月

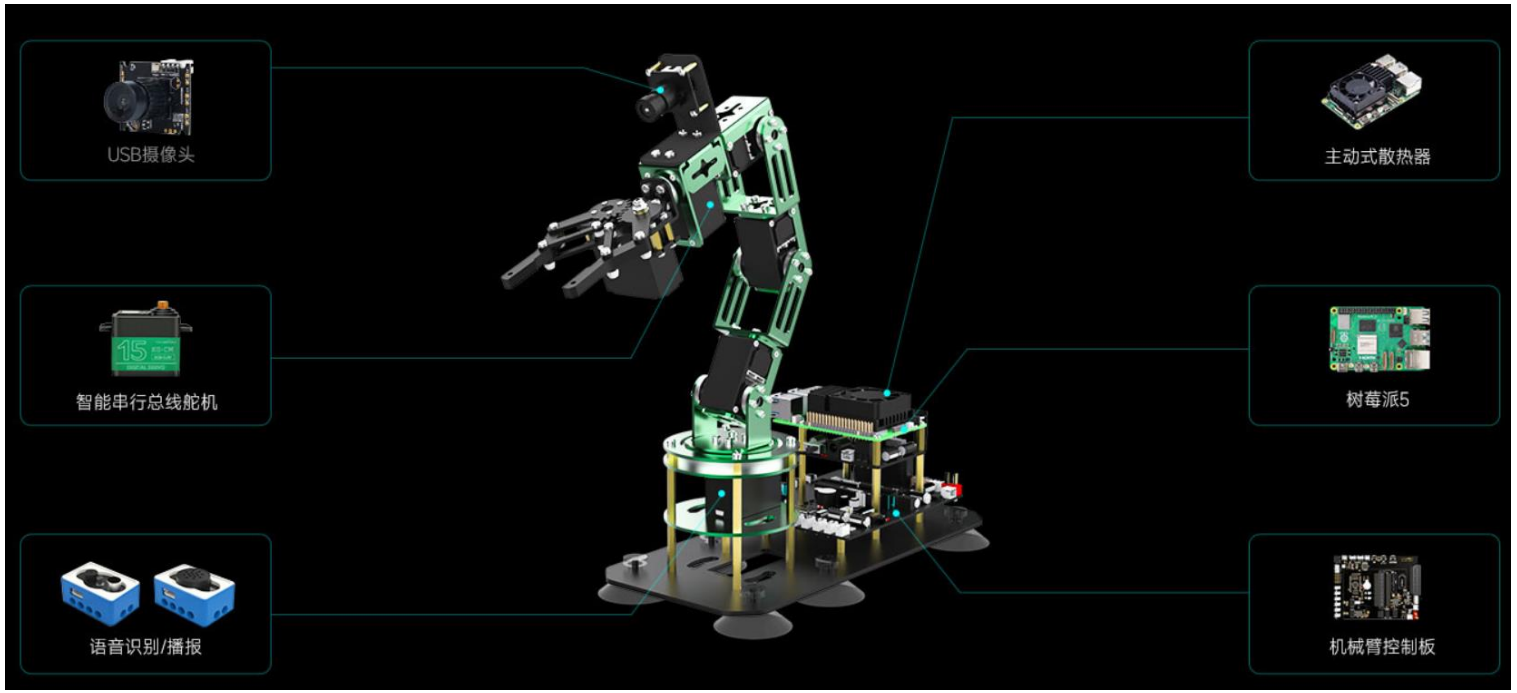
目录

- 机械臂简介
- 机械臂仿真操作
  - pybullet仿真环境
  - 机械臂仿真控制
  - 虚拟机安装
  - 大作业：仿真抓取放置
- 机械臂实机操作
  - ros简介
  - 机械臂ros控制
  - 大作业：实机抓取放置

机械臂简介

简介

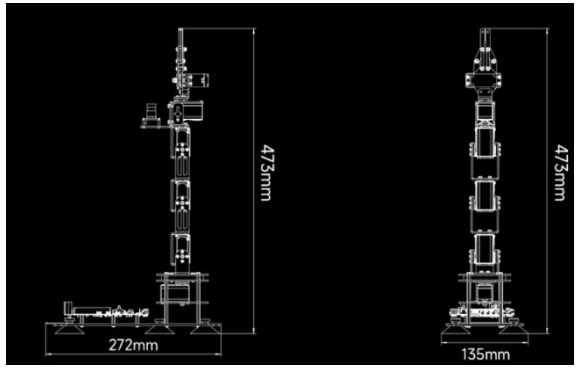
实验用机械臂： DOFBOT 树莓派AI视觉机械臂。



功能开发

- 图像处理库： Open CV。
- 开发工具： Jupyter Lab。
- 编程语言： Python。
- 操作系统： ROS。
- 图像识别检测： MediaPipe。
- 运动规划： MoveIt

规格参数

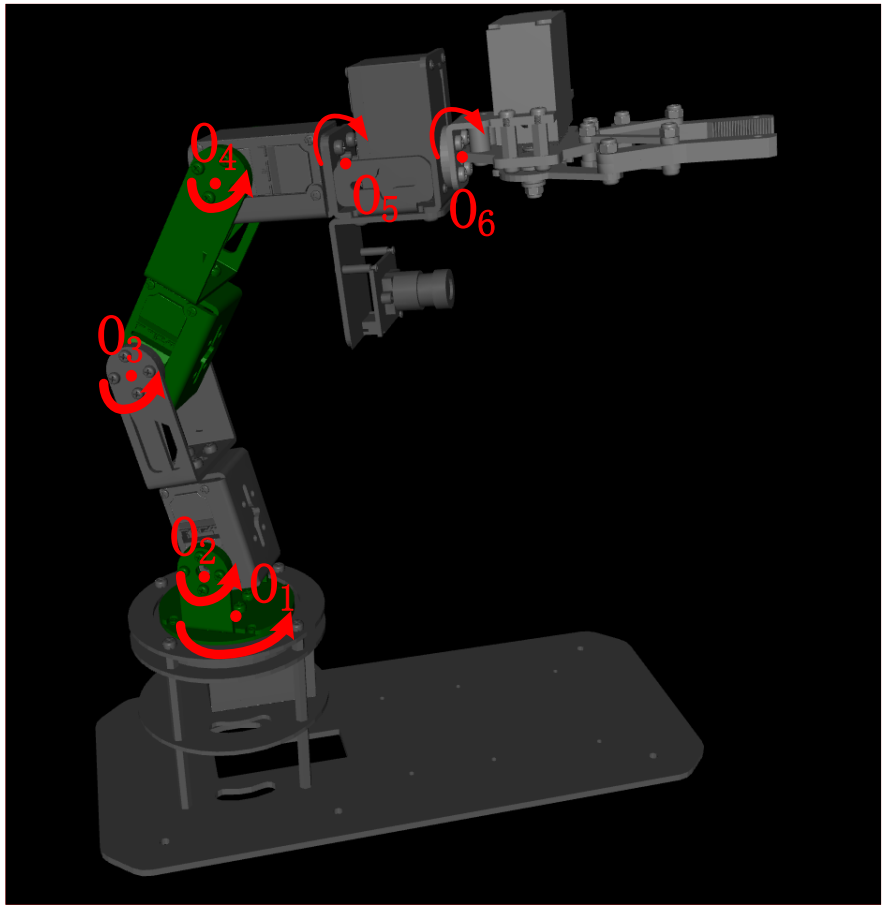


机械臂简介

机械臂实验

机械臂的机械结构

机械臂的六个自由度



机械臂仿真操作

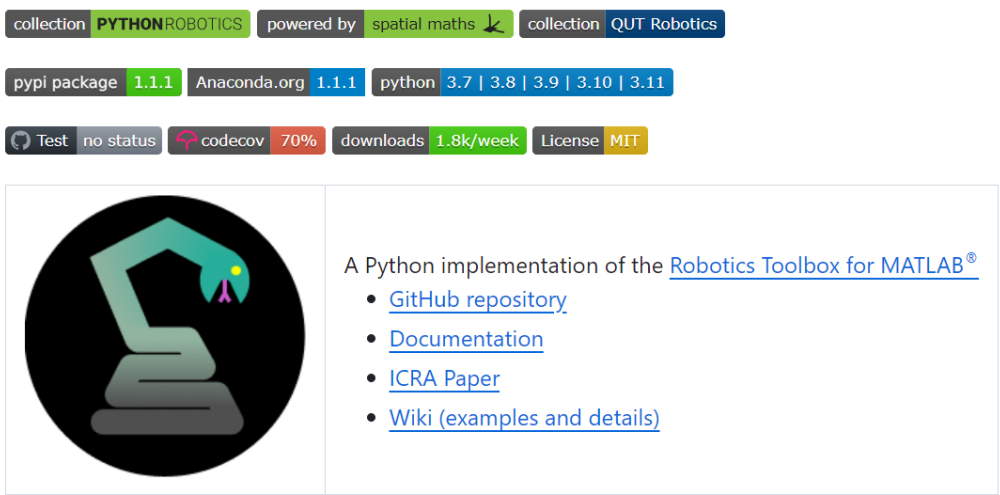
Robotics ToolBox Python

RoboticsToolboxPython提供了Matlab中的机器人学工具包的Python版本，并兼容了大量Python中的线性代数工具库(numpy等)，可视化工具包(matplotlib等)，交互工具包(jupyter等)。RoboticsToolboxPython使得用户可以方便地进行运动学和动力学的仿真。

Pybullet以及其可视化组件安装

```
pip install roboticstoolbox-python
pip install numpy<2
pip install pyqt5
```

Robotics Toolbox for Python



机械臂仿真操作

Robotics ToolBox Python使用基础

官方文档：https://petercorke.github.io/robotics-toolbox-python/

定义串联机器人

- A, alpha, d分别对应DH参数中除了θ外的另外三个，offset会加在θ项上，处理θ = θ + offset的情况。
- RevoluteMDH表示改进DH法，需要用改进DH法建立DH矩阵

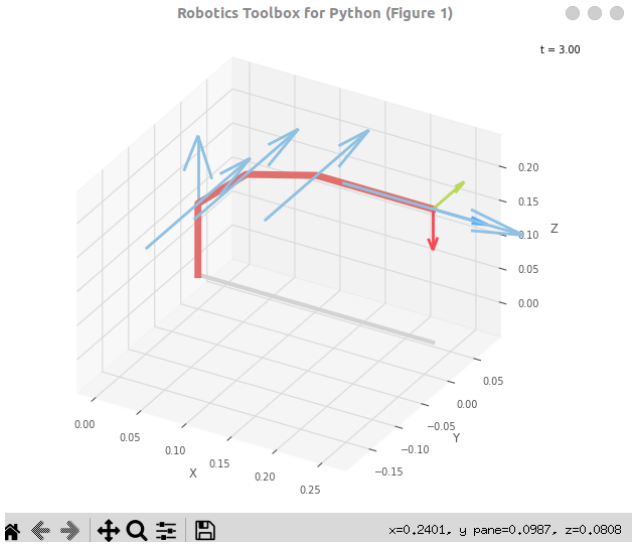
```
# student version
# 用改进DH参数发表示机器人正运动学
dofbot = rtb.DHRobot(
    [
        rtb.RevoluteMDH(a = 1, alpha=11, d=111, offset=1111),
        rtb.RevoluteMDH(),
        rtb.RevoluteMDH(),
        rtb.RevoluteMDH(),
        rtb.RevoluteMDH()
    ]
)
```

机械臂仿真操作

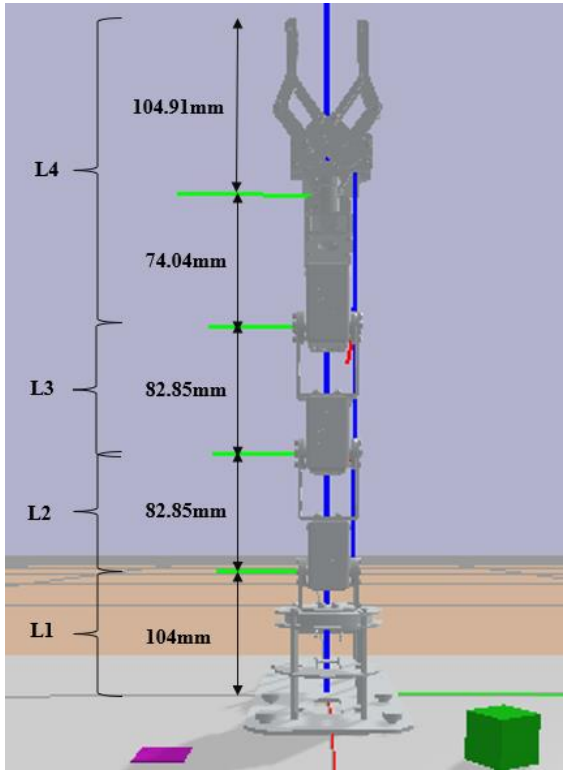
Dofbot DH参数仿真

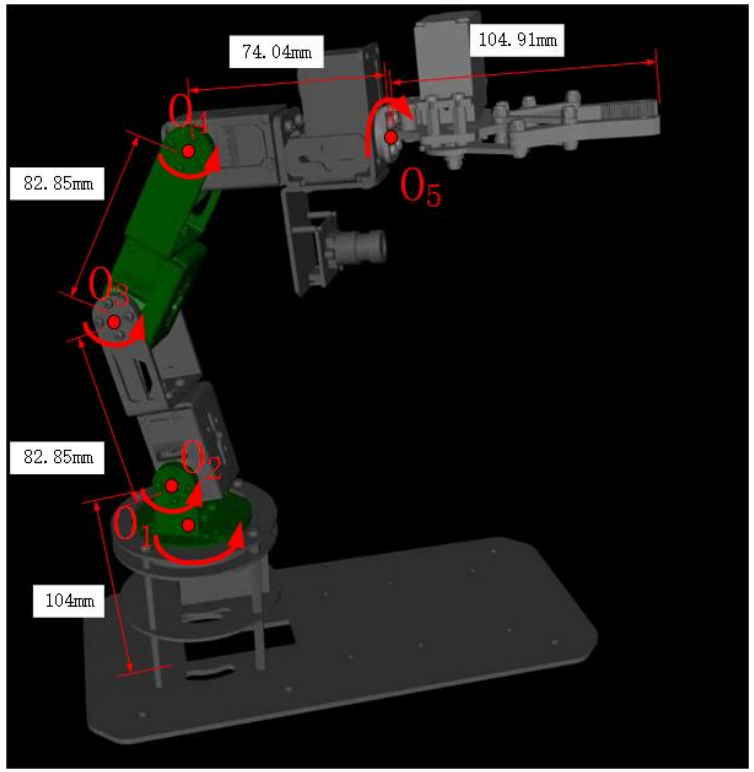
官方文档：https://petercorke.github.io/robotics-toolbox-python/

计算DH参数表，并使用RobotToolboxPython仿真正链运动学



机械臂正链运动学仿真示例





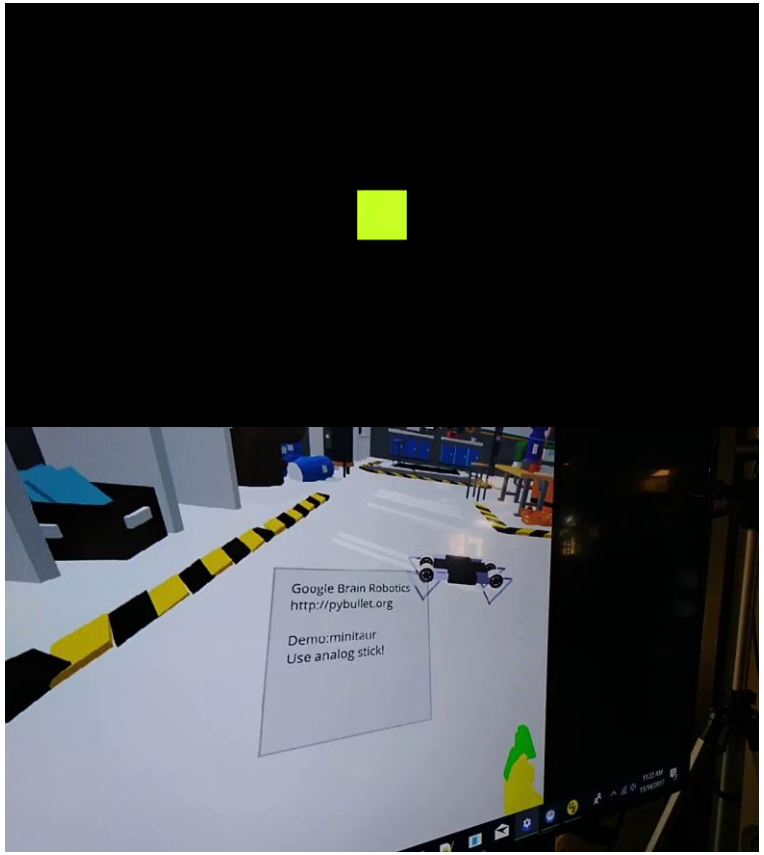
机械臂连杆参数

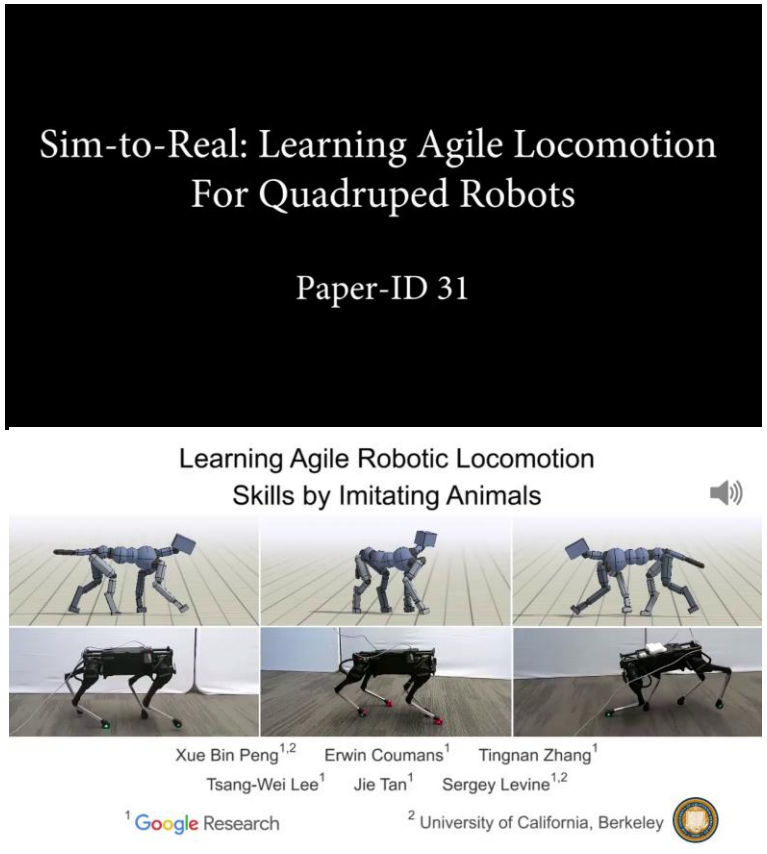
机械臂仿真操作

Pybullet

PyBullet 基于著名的开源物理引擎 bullet 开发，封装成了 Python 的一个模块，用于机器人仿真和学习。PyBullet 支持加载 URDF、SDF、MJCF 等多种机器人描述文件，并提供正/逆向运动学、正/逆向动力学、碰撞检测等功能。(https://pybullet.org/wordpress/)，Bullet 物理 SDK 包括 PyBullet 机器人示例，如模拟的 Minitaur 四足机器人、使用 TensorFlow 推理的仿人机器人跑步和 KUKA 机械臂抓取物体。

PyBullet安装：pip install pybullet





机械臂仿真操作

pybullet使用基础

官方文档：
https://docs.google.com/document/d/10sXEhzFRSnvFcl3XxNGhnD4N2SedqwdAvK3dsihxVUA/edit#heading=h.2ye70wns7io3

p.connect：连接到PyBullet物理引擎，返回一个物理引擎的客户端ID。
p.loadURDF：加载一个URDF文件表示的物体或机器人模型到仿真环境中。
p.setGravity：设置仿真环境中的重力。
p.getJointState：获取关节的状态信息，如位置、速度等。
p.getLinkState：获取连杆的状态信息。
p.setJointMotorControl2：设置关节的控制方式，如位置控制、速度控制等。
p.calculateInverseKinematics：计算逆运动学。

机械臂仿真操作

Dofbot机械臂仿真

仿真初始化：

```
class DofbotEnv:  1 usage
    def __init__(self):
        self._timeStep = 0.02
        p.connect(p.GUI)
        p.setPhysicsEngineParameter(numSolverIterations=150)
        p.setTimeStep(self._timeStep)
        p.setGravity( gravX: 0,  gravY: 0, -9.8)
        p.loadURDF( fileName: "models/floor.urdf",  basePosition: [0, 0, -0.625], useFixedBase=True)
        p.loadURDF( fileName: "models/table_collision/table.urdf",  basePosition: [0.5, 0, -0.625],p.getQuaternionFromEuler([0, 0, 0]),
                   useFixedBase=True)
        self._dofbot = dofbot("models/dfbot_urdf/dofbot.urdf")
        self._object1 = Object( urdfPath: "models/box_green.urdf", block=True,num=1)
        self._object2 = Object( urdfPath: "models/box_purple.urdf", block=True,num=2)
```

关节角度控制

逆运动学

关节角度读取

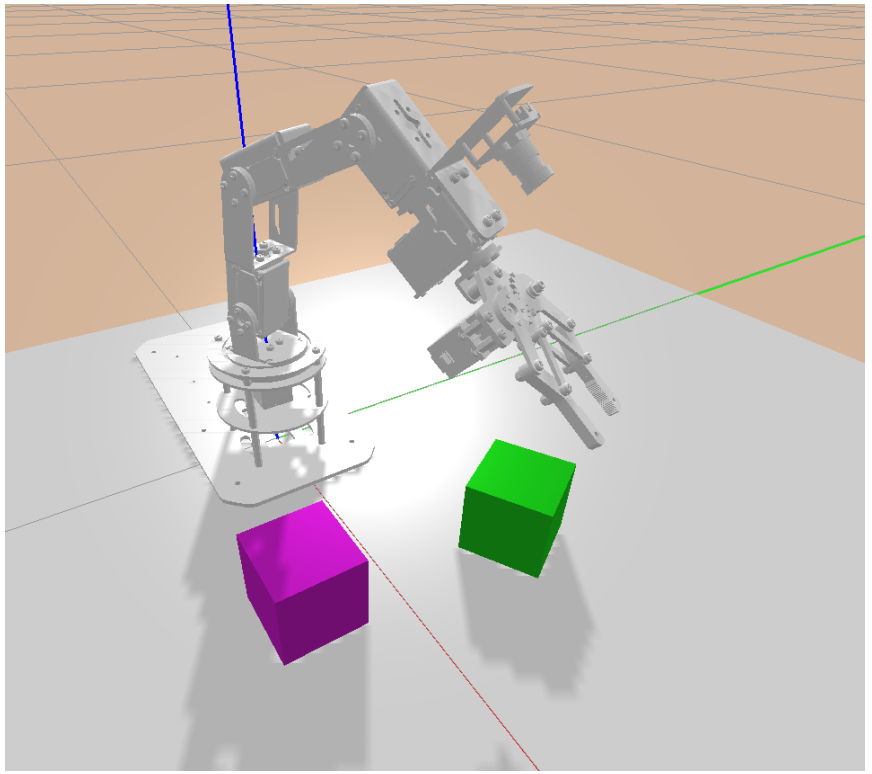
末端位姿读取

```
def dofbot_joint_control(self,jointPoses):  1 usage
    self._dofbot.jointControl(jointPoses)
    p.stepSimulation()
    time.sleep(self._timeStep)

def dofbot_setInverseKine(self,pos,orn = None):  1 usage
    jointPoses = self._dofbot.setInverseKine(pos, orn)
    return jointPoses

def get_dofbot_jointPoses(self):
    jointPoses = self._dofbot.get_jointPoses()
    return jointPoses

def get_dofbot_pose(self):
    pos,orn = self._dofbot.get_pose()
    return pos, orn
```



机械臂+两个物块

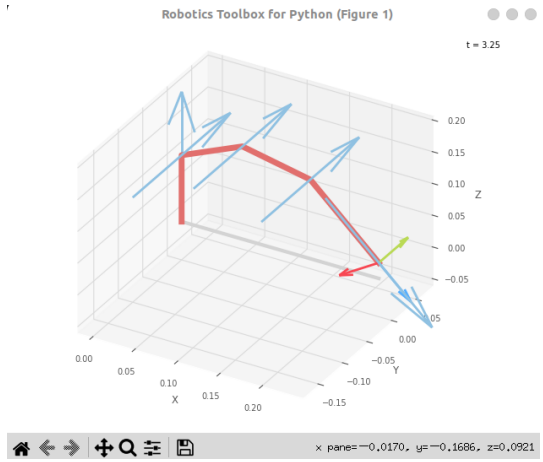
作业（机械臂仿真部分）

任务说明：使用RobotToolboxPython工具完成Dofbot构型机械臂正运动学、逆运动学、与工作空间的仿真。

仿真内容：

1. 给出Dofbot机械臂在以下关节角度下的正运动学解，并在报告中附上RobotToolboxPython给出的运动学姿态仿真。
   1. (demo) [0., pi/3, pi/4, pi/5, 0.]

```
-0.7771   -1.686e-08  0.6293    0.2326
 1.686e-08  1         4.762e-08  5.58e-09
-0.6293    4.762e-08 -0.7771    0.02468
 0         0         0         1
```



   2. [pi/2, pi/5, pi/5, pi/5, pi]
   3. [pi/3, pi/4, -pi/3, -pi/4, pi/2]
   4. [-pi/2, pi/3, -pi/3*2, pi/3, pi/3]

作业（机械臂仿真部分）

任务说明：使用RobotToolboxPython工具完成Dofbot构型机械臂正运动学、逆运动学、与工作空间的仿真。

仿真内容：

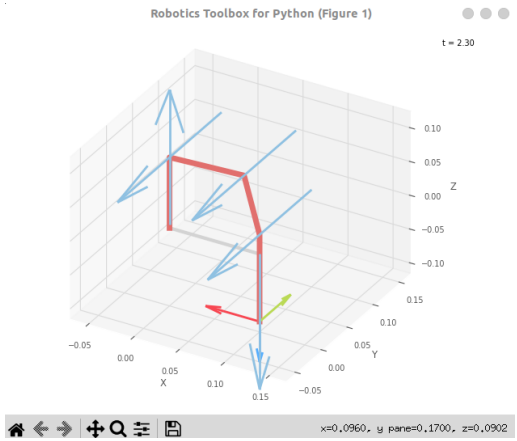
2. 给出Dofbot机械臂夹爪末端在以下笛卡尔空间姿态下的逆运动学解，并在报告中附上RobotToolboxPython给出的运动学姿态仿真。

0.(demo)

```
[
    [-1., 0., 0., 0.1,],
    [0., 1., 0., 0.],
    [0., 0., -1., -0.1],
    [0., 0., 0., 1.]
]
```

1.

```
[
    [1., 0., 0., 0.1,],
    [0., 1., 0., 0.],
    [0., 0., 1., 0.1],
    [0., 0., 0., 1.]
]
```



2.

```
[
    [cos(pi/3), 0., -sin(pi/3), 0.05,],
    [0., 1., 0., 0.03],
    [sin(pi/3), 0., cos(pi/3)., -0.1],
    [0., 0., 0., 1.]
]
```

3.

```
[
    [-0.866, -0.25, -0.433, -0.03704,],
    [0.5, -0.433, -0.75, -0.06415],
    [0., -0.866, 0.5, 0.3073],
    [0., 0., 0., 1.]
]
```

作业（机械臂仿真部分）

任务说明：使用RobotToolboxPython工具完成Dofbot构型机械臂正运动学、逆运动学、与工作空间的仿真。

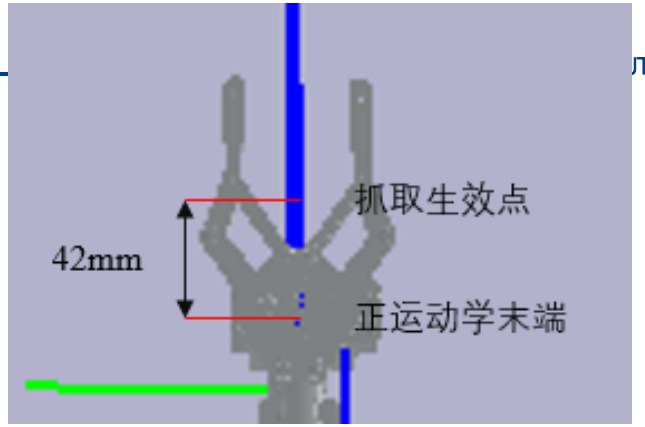
仿真内容：

3. 绘制机械臂工作空间（至少500个点）（可以按照关节空间针对每个关节进行采样）

关节角度限位：

J1：[-180°, 180°]
J2：[-90°, 90°]
J3：[-150°, 150°]
J4：[-100°, 100°]
J5：[-180°, 180°]

作业（机械臂仿真部分）



任务说明：使用Dofbot机械臂完成物块抓取放置任务

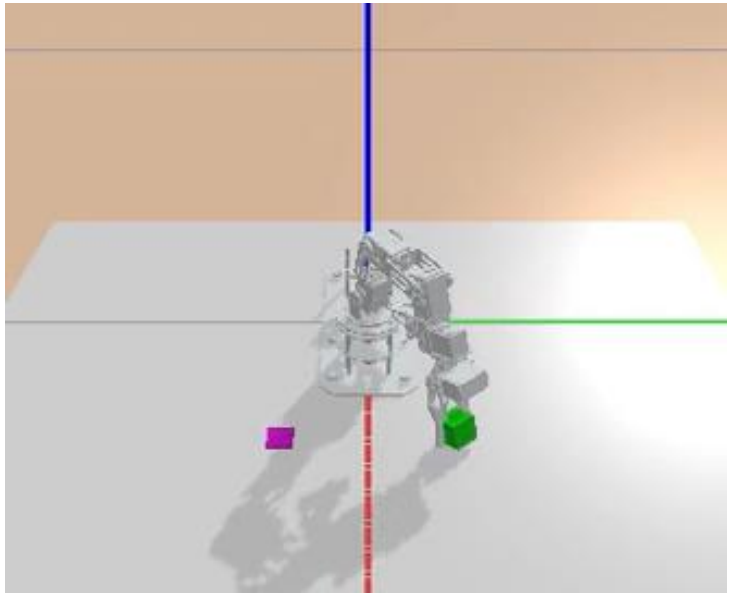
工具说明：使用PyBullet完成，使用额外工具需在报告中说明

仿真内容：

4. 夹取物块，坐标：（0.2，0.1，0）欧拉角：（0，0，pi/6）物块高度：0.03
5. 将物块放置到目标位置（0.2，-0.1，0）

* 注意事项：

1. 仿真中需要注意，机械臂所有舵机默认位置为关节角度为pi/2的位置。
2. 计算机械臂逆运动学时需要注意，由于实验使用的机械臂模型并没有全自由度，因此只有合法的工作空间姿态才能求解出合法的关节空间姿态。
3. 由于提供的urdf文件末端并不处于夹爪中心，计算逆运动学前需要适当对工作空间位置增加补偿



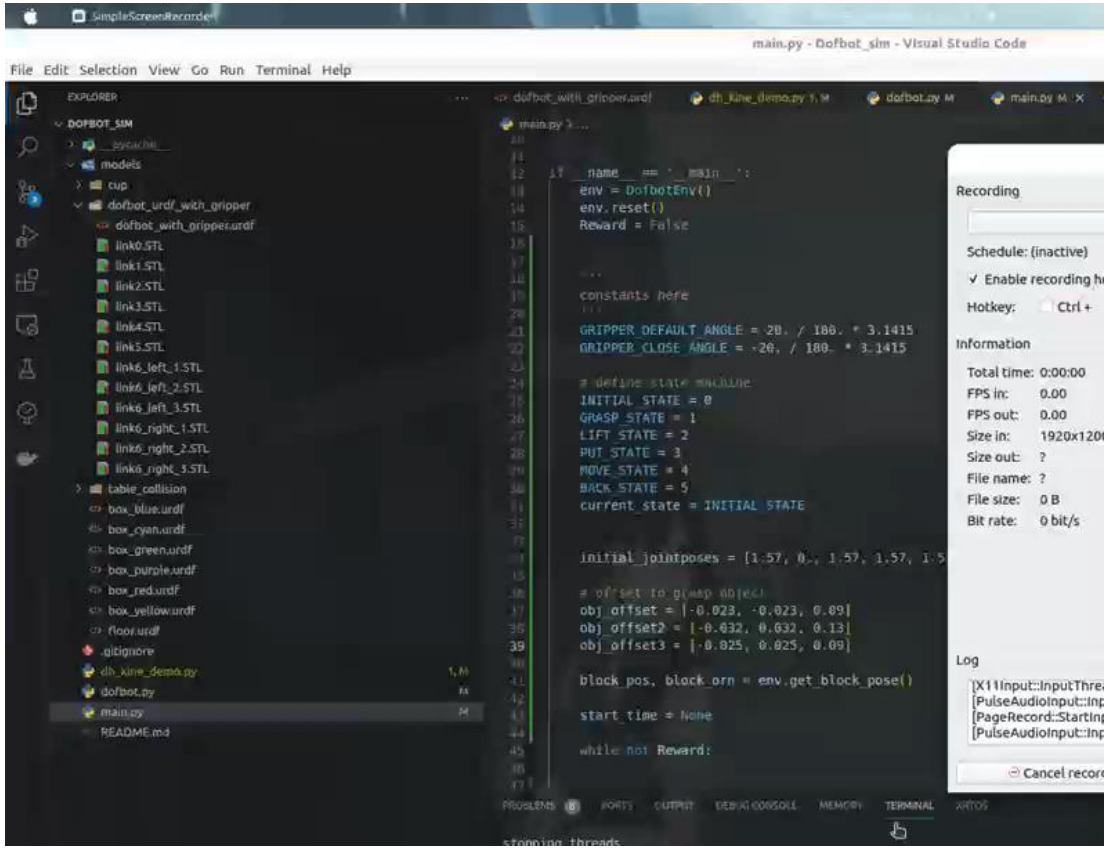
机械臂仿真操作

Dofbot机械臂仿真效果演示

实现提示：

将夹起物块并放下的过程分为5个状态：

1. 将机械臂从初始位置移动到目标位置
2. 夹起物块
3. 将物块提起
4. 夹起物块并移动到目标位置
5. 放下物块



作业（机械臂仿真部分）

报告要求:
任务一：给出Dofbot机械臂在以下关节角度下的正运动学解，并在报告中附上RobotToolboxPython给出的运动学姿态仿真。
任务二：给出Dofbot机械臂在以下笛卡尔空间姿态下的逆运动学解，并在报告中附上RobotToolboxPython给出的运动学姿态仿真。
任务三：绘制机械臂工作空间（至少500个点）
[要求]:

- 给出机械臂的DH矩阵以及其坐标系建立方法
- 针对前两个任务，给出每个数据对应解、以及姿态仿真截图
- 针对第三个任务，给出工作空间采样点阵
- 在提交附件中包含实现代码

任务四：夹取物块并将物块放置到目标位置
[要求]:

- 描述该任务实现思路以及实现方法
- 在提交附件中包含实现代码

注：如在实现任务过程中用到了第三方库，请在报告中说明。

作业（机械臂仿真部分）

大作业得分占比：

仿真部分： 实机部分 = 8： 2

机械臂仿真操作

简介

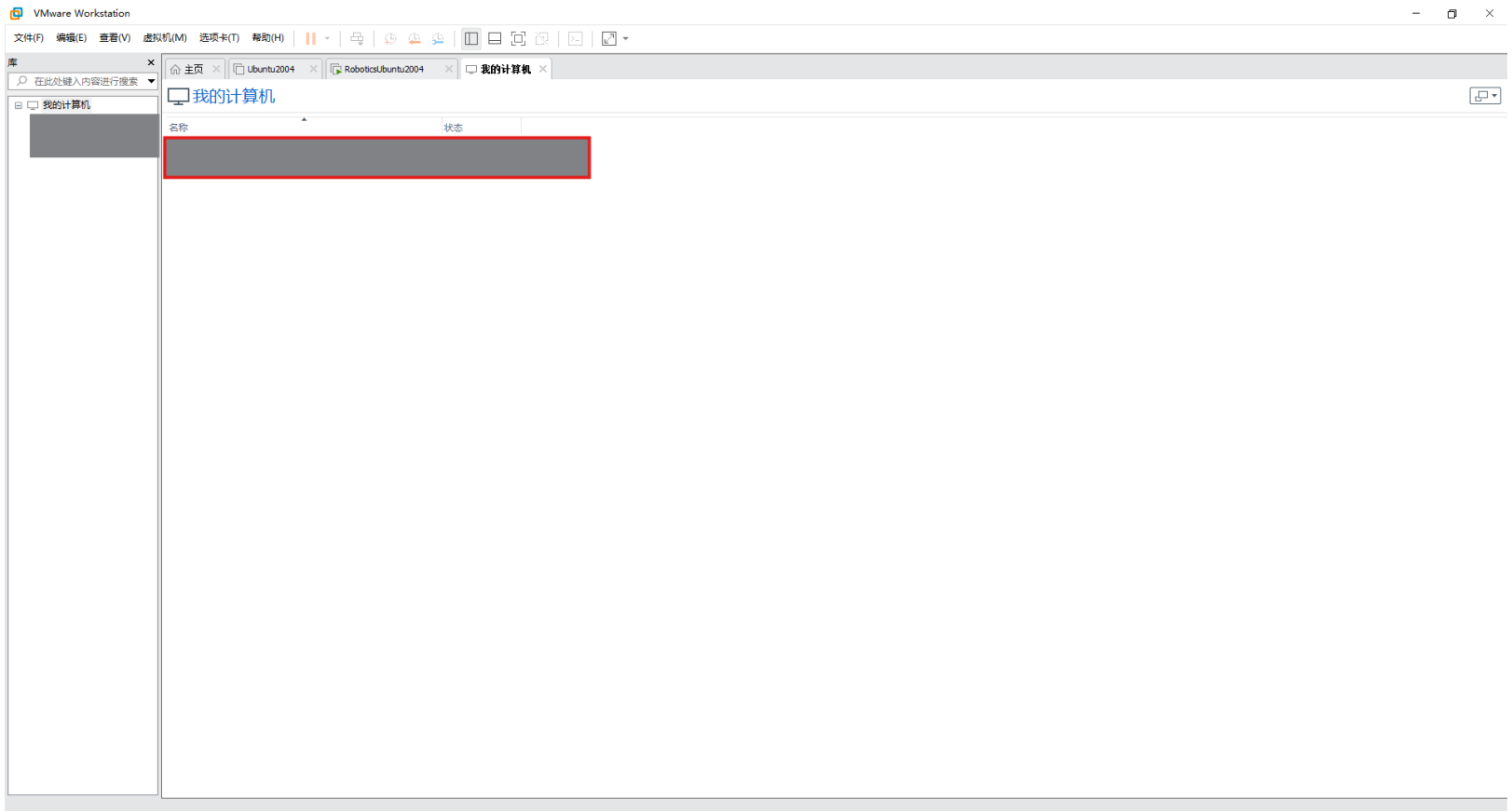
在实践课中，我们将使用**Linux系统Ubuntu20.04**进行程序编写，运行机械臂仿真与目标检测程序。可以使用虚拟机安装Ubuntu20.04操作系统。
交大软件授权中心中提供了VMware Workstation的安装授权，我们使用VMware Workstation进行虚拟机搭建。

交大软件授权中心（VMware）：
https://software.sjtu.edu.cn/List/View/339

Ubuntu操作系统版本：
20.04 64位

虚拟机安装

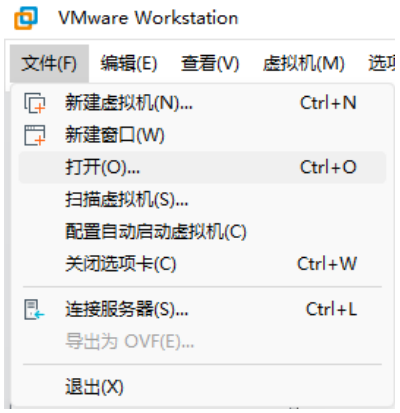
打开VMware



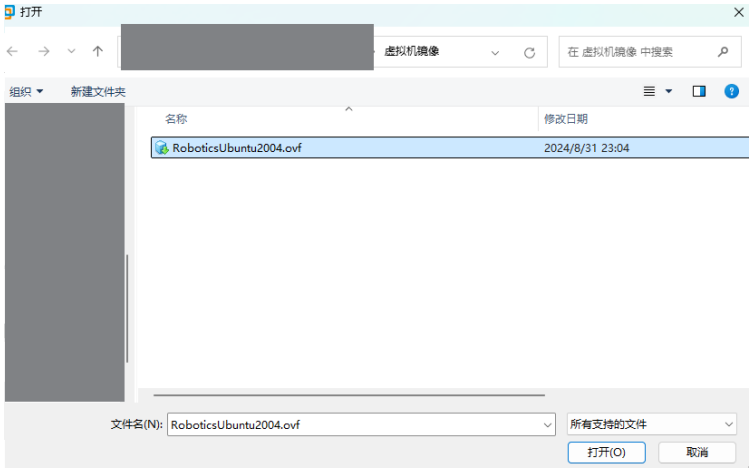
虚拟机安装

通过Jbox下载已经配置好的虚拟机环境镜像：

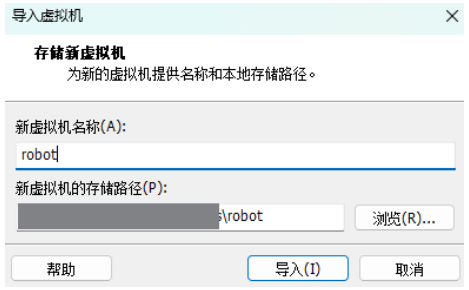
分享内容: vmware虚拟机镜像链接: https://pan.sjtu.edu.cn/web/share/6667560413f38e34cc865f77bd6f5ef4, 提取码: 1234



打开Vmware，文件->打开->选中从交大云盘中下载的ovf文件，打开虚拟机。



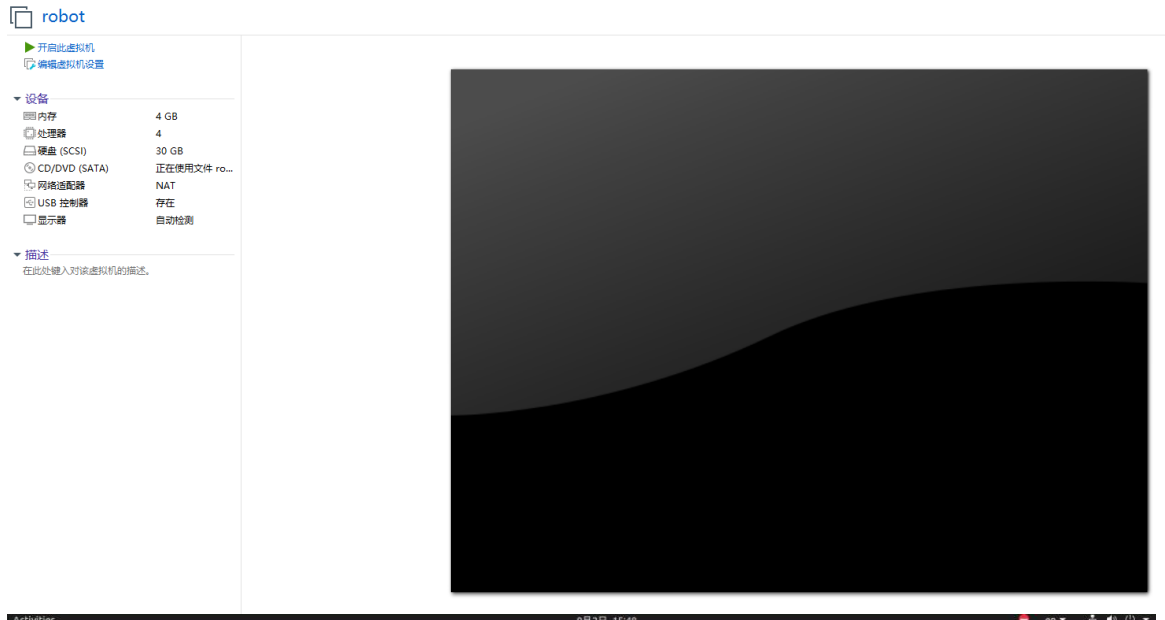
设定自己的虚拟机名称和虚拟机存储路径，导入虚拟机。

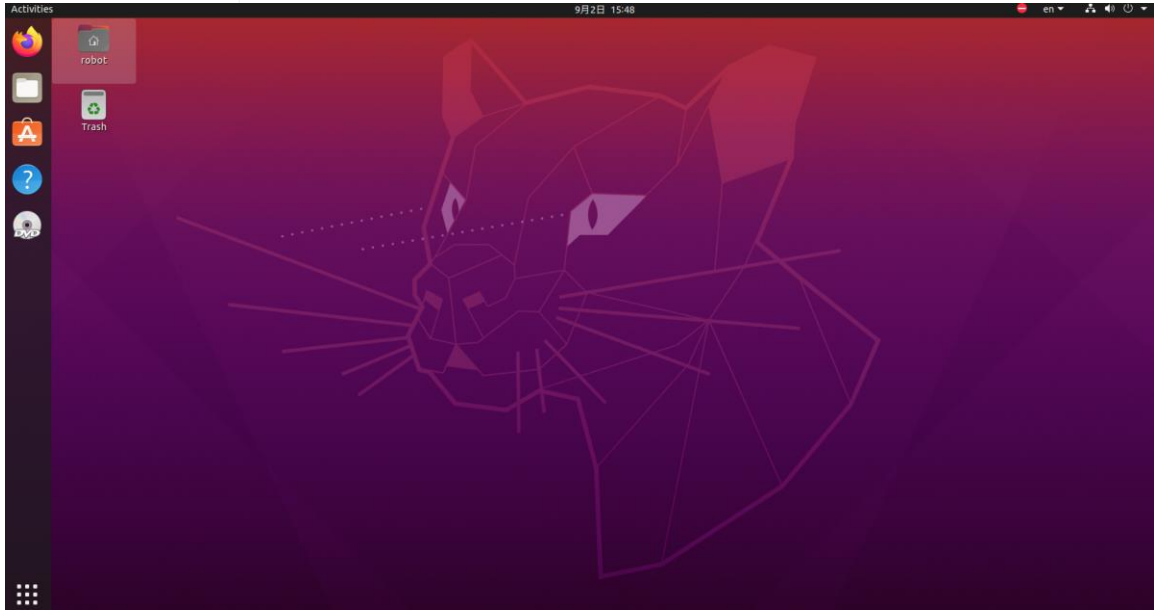


虚拟机安装

用户名：robot
密码：robot

打开虚拟机

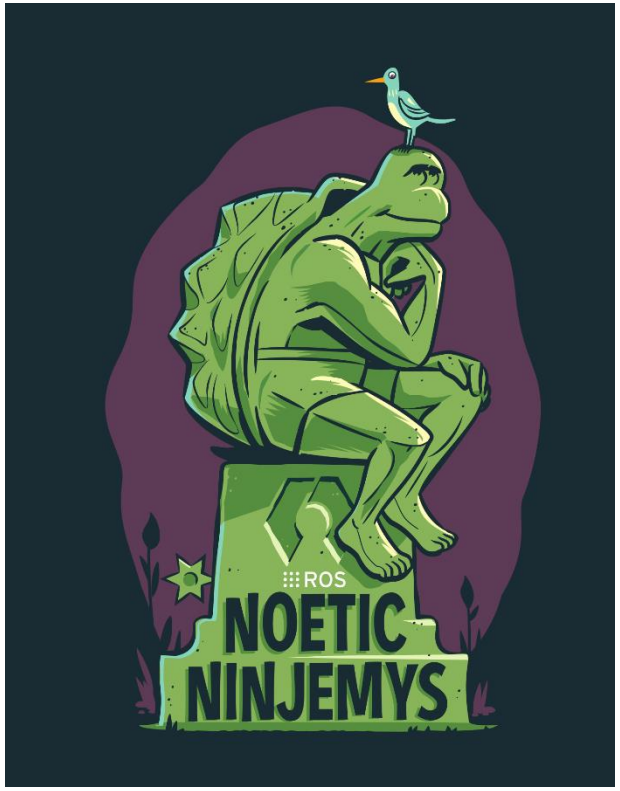


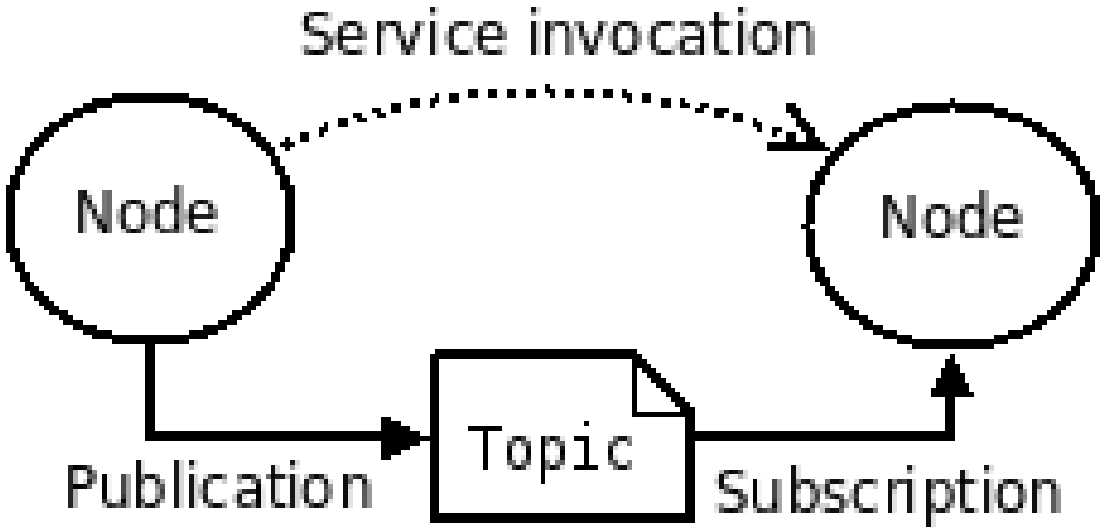


ROS 机器人操作系统

简介

在实践课中，我们将通过ROS来控制机械臂。
ROS，全称机器人操作系统，是一款分布式操作系统，作为不同机器人系统之间的通信中间件。
各个机器人系统被作为一个个节点（node），他们之间通过话题（topic）来传递信息（message）和发布服务（service）。

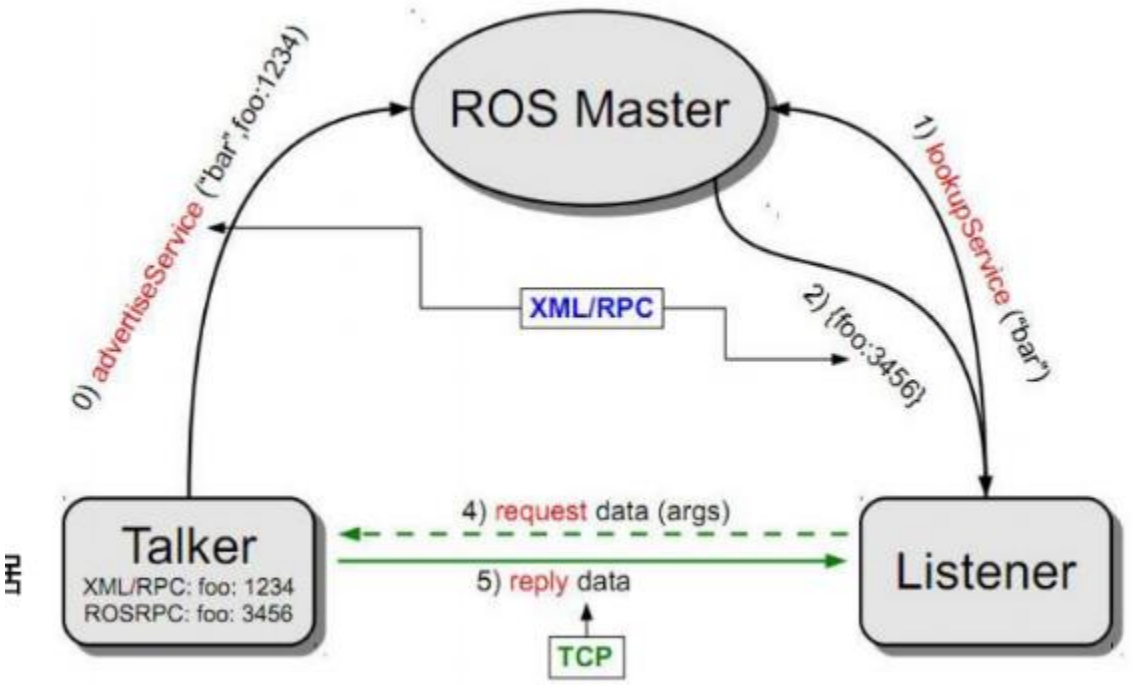




ROS基础知识

节点

节点由一个核心（roscore）进行管理。ROS的节点可以通过设置ip地址，在同一个局域网下进行通信。在我们实验中，roscore在机械臂单片机上运行。故而，在我们上位机上，只需要设置机械臂单片机上运行的ROS核心的ip地址，便可与机械臂进行通信。



本机ip地址

```
export ROS_HOSTNAME=192.168.1.189
```

```
export ROS_MASTER_URI=http://192.168.1.123:11311
```

单片机上ROS核心ip地址

ROS基础知识

话题

ROS话题是由发布者发布的，同时由接收者接收，从而实现不同节点间的通信。ROS提供了简单的命令行接口来查看当前整个ROS系统中的话题

```
comoer@comoer-G3-3500:~/Workspace/projects/src/robot_arm_control$ rostopic list
/rosout
/rosout_agg
```

rostopic list # 列出当前所有话题
rostopic info <话题名> # 查看话题信息，包括其发布者和订阅的接收者
rostopic echo <话题名> # 打印话题发布的消息内容

消息

ROS话题发布一定类型的消息。消息一般呈现为一个基本数据类型组合的结构体。
ROS提供了简单的命令行接口来查看消息的组成。

rosmsg show/info <消息名>

```
irmv@irmv:~/workspace/zy/ws_grasp$ rosmsg info sensor_msgs/JointState
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string[] name
float64[] position
float64[] velocity
float64[] effort
```

ROS机械臂控制

机械臂控制

我们可以通过自己编写的脚本来控制机械臂，脚本将通过**发布和接收话题**和单片机上的控制程序交互。

ROS提供了简单的Python接口rospy。在rospy中，一个基本的流程为建立节点，定义该节点的发布者Publisher（发布话题向其他节点传递消息），接收者Subscriber（接收其他节点的消息）。

```python
import rospy
from sensor_msgs.msg import JointState

1 usage
def callback(data:JointState):

    angles = data.position
    print("Angles are ",angles)

if __name__ == '__main__':
    rospy.init_node("Arm3")

    rospy.Subscriber( name: "/dofbot/joint_states", JointState, callback)

    rospy.spin()
```

回调函数，获取机械臂六个关节角度

定义接收者

机械臂关节角度读取

```python
import rospy
from sensor_msgs.msg import JointState


if __name__ == '__main__':
    rospy.init_node("Arm2")

    pub = rospy.Publisher("/dofbot/cmd", JointState, queue_size=10)

    rate=rospy.Rate(10)

    angle = 10


    while not rospy.is_shutdown():

        angles = [90 for _ in range(6)] +[100]

        angles[2] = angle
        angle += 0.1

        msg = JointState()
        msg.header.stamp = rospy.Time.now()
        msg.position = angles
        msg.name = [f"Joint{i}" for i in range(6)]

        pub.publish(msg)

        rate.sleep()
```

定义发布者，以发布控制指令

发布指令，前六位为目标机械臂关节角度，最后一位为运行时间

发布消息

机械臂关节控制

ROS机械臂控制

RealEnv类

基于上述基本命令封装了RealEnv类，便于同学们控制实际机械臂。

功能函数：

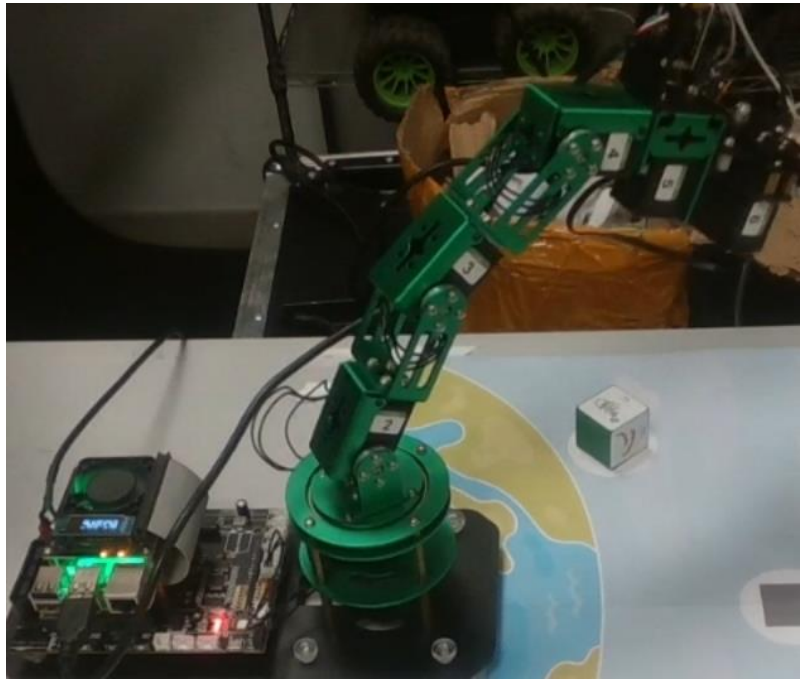
env.reset():
实现机械臂复位

env.get_state(): （内部调用机械臂关节读取命令）
得到当前机械臂关节角度

env.step(joints, gripper_angle):
（内部调用机械臂关节控制命令）
当joints和gripper_angle分别非空时，
控制前五关节角度和夹爪。

Ps：夹爪角度不应设置太大，低于140度，
太大会导致夹爪损坏

机械臂复位状态



机械臂运动控制

作业（实机部分）

任务说明：使用Dofbot机械臂完成物块抓取放置任务

已知条件：
（1）物体初始位置和放置位置确定
（2）机械臂参数已知
（3）可通过**示教**获取**初始位置**夹取和最后放置位置的机械臂关节参数
（5）助教提供机械臂端ROS代码和控制端环境接口代码，以及需要补全的控制代码

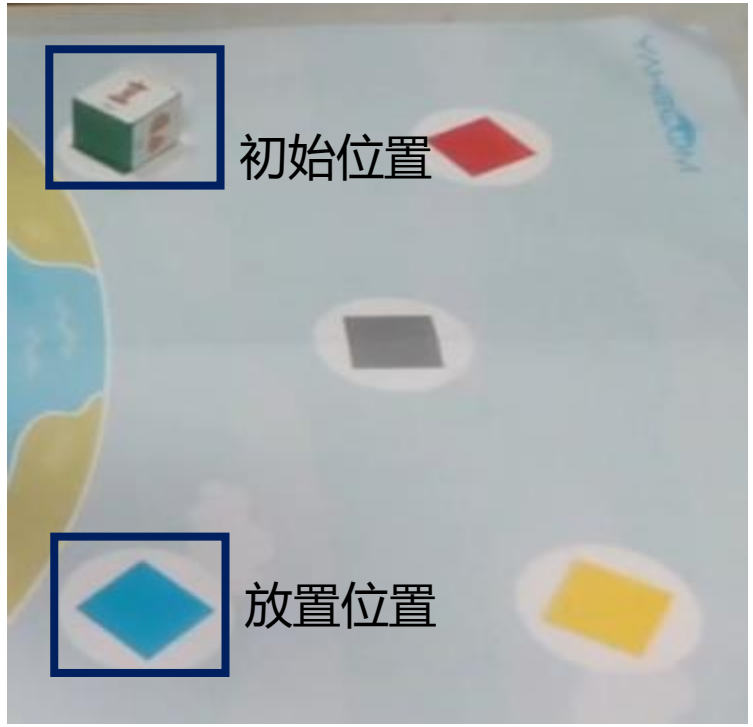
代码环境：
Python， ROS

任务要求：

基于仿真实验和相应的控制代码，结合ROS与Dofbot机械臂进行通信，控制机械臂完成实机抓取放置任务，即机械臂抓取**初始位置**处物块放置到**放置位置**（**不要求精准放置，放置到粗略位置即可**）。

Ps: 作业最终效果要求可参照实现思路参考的完整demo部分。

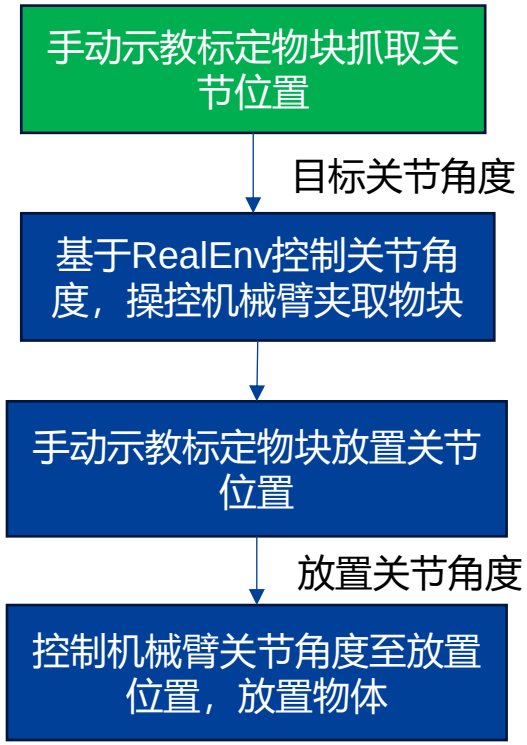
报告要求:
姓名，学号，班级
实现思路
代码
图片



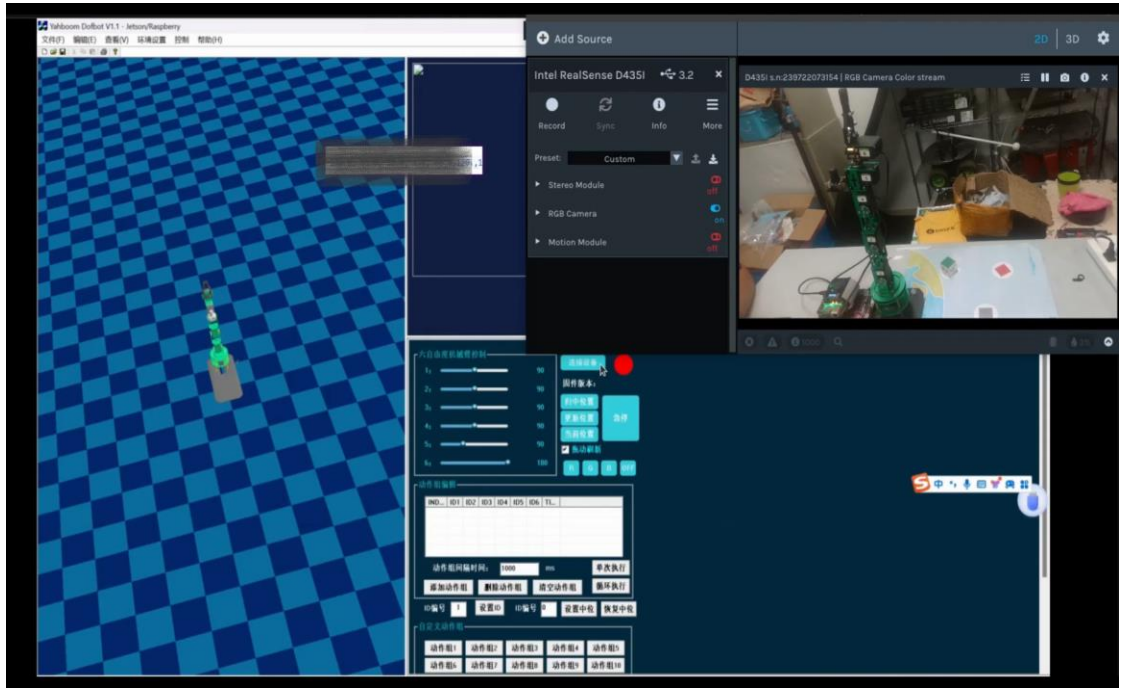
作业-实现思路参考（实机部分）

步骤1：通过示教软件手动示教获得物体抓取位置，记录抓取放置在**绿色区域的方块**时的关节位置

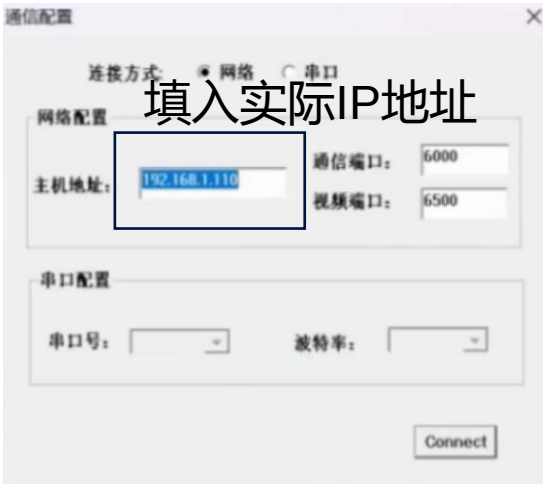
大致角度为（137,51,52,2,90,120）便于同学们调整（最后一位是夹爪，120是夹取角度）



任务pipeline

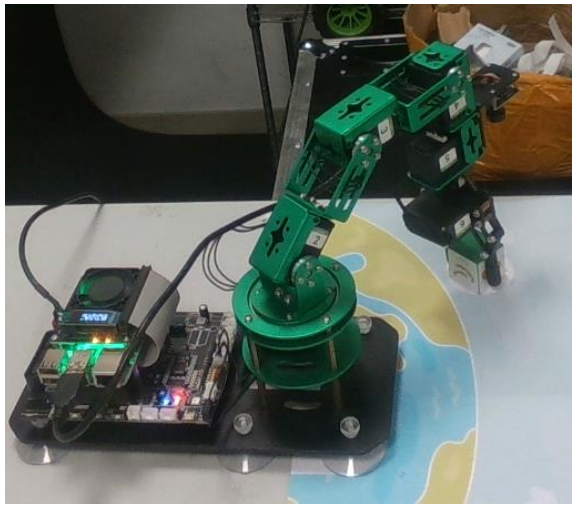


手动示教演示视频



填入实际IP地址

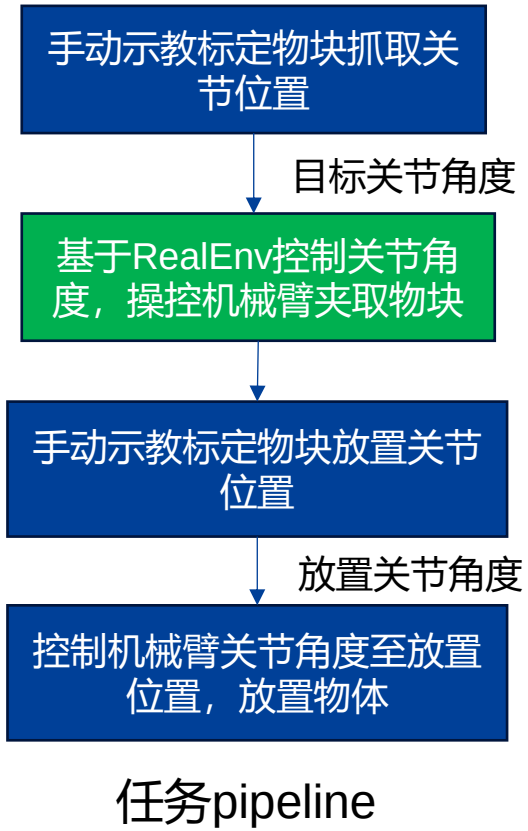
手动示教标定结果



手动示教标定结果

作业-实现思路参考（实机部分）

步骤2：基于RealEnv控制关节角度，操控机械臂夹取物块



任务pipeline

机械臂的IP

```
(dexcap) irmv@irmv:~/dofbot/src/dofbot_ctrl/script$ export ROS_MASTER_URI=http://192.168.1.110:11311
(dexcap) irmv@irmv:~/dofbot/src/dofbot_ctrl/script$ export ROS_HOSTNAME=192.168.1.191
(dexcap) irmv@irmv:~/dofbot/src/dofbot_ctrl/script$ rostopic list
/dofbot/cmd
/dofbot/joint_state
/rosout
/rosout_agg
```

同学们电脑的IP

ROS IP设置，通过rostopic list检查是否连上机械臂

环境初始化

```python
# 调用realenv
env = RealEnv()
env.reset()
```

中间路点+状态机示例

```python
# 可以实现简单状态机来实现分段控制
# 状态机的中间路点
points = [
    np.asarray([90., 90., 90., 90., 90.]), # 初始位置
    ...
]

for i in range(len(points) - 1):
    # 取出路点并做路径规划得到路径
    path = ...
    for p in path:
        # 执行路径上各点
        # env.step(joint=...)可以控制关节
        # env.step(gripper=...)可以控制夹爪
        # 建议分开控制
        ...
```

Tips:中间路点对于任务成功非常重要，通过线性插值得到中间路点

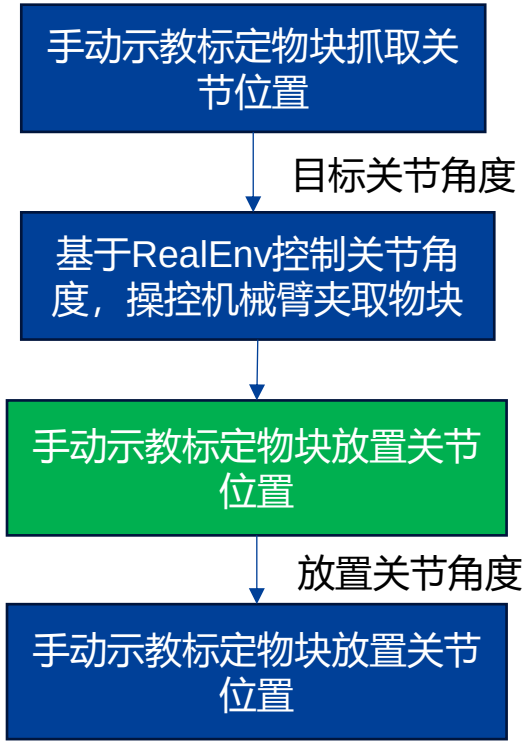
机械臂控制

助教提供不完整示例代码

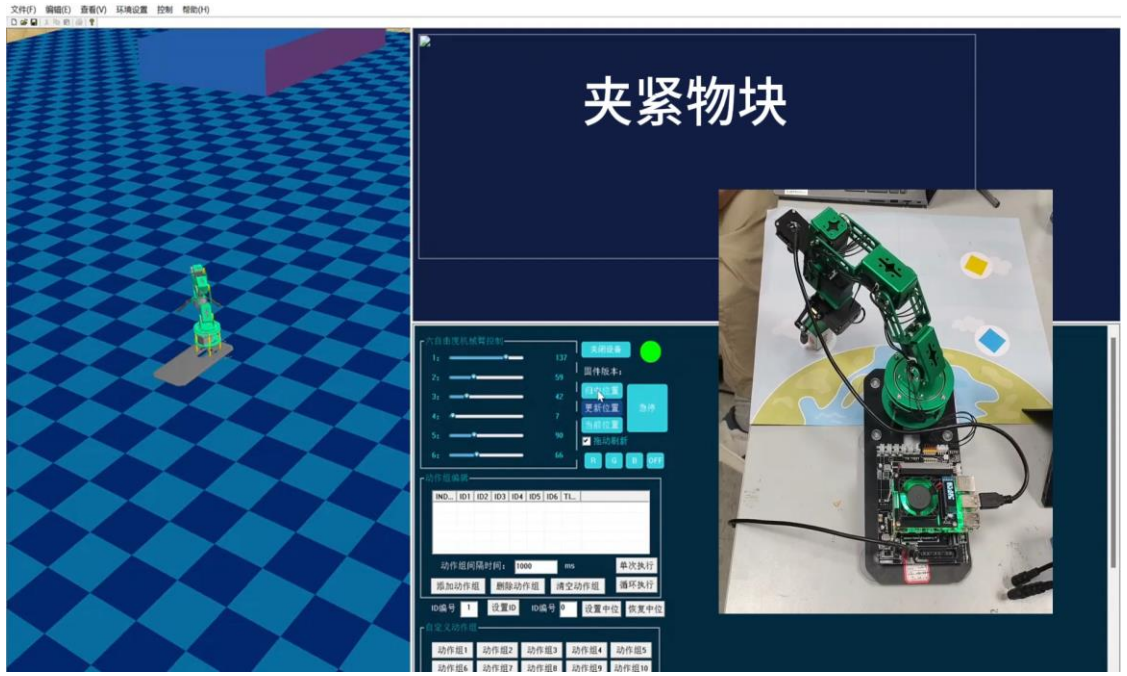
作业-实现思路参考（实机部分）

步骤3：通过示教软件手动示教获得物体放置位置，记录机械臂将物块放置在**蓝色区域**时的关节位置

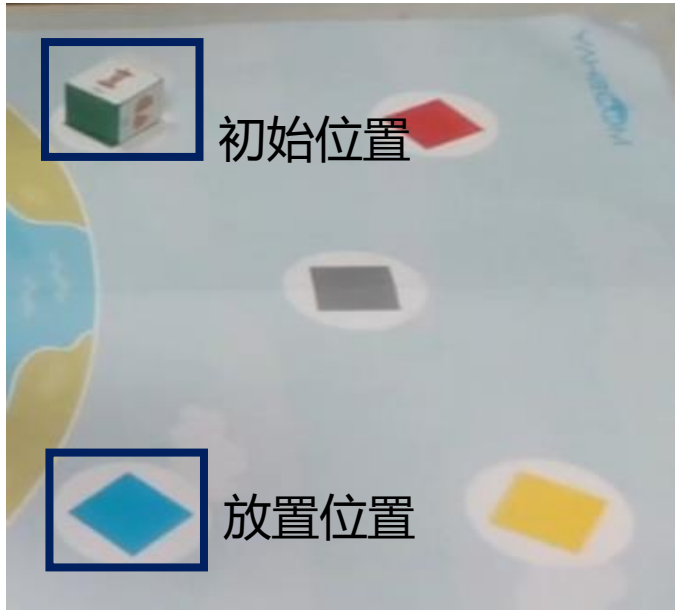
大致角度为（40,61,42,7,90,94）便于同学们调整（最后一位是夹爪，94是放开角度）



任务pipeline



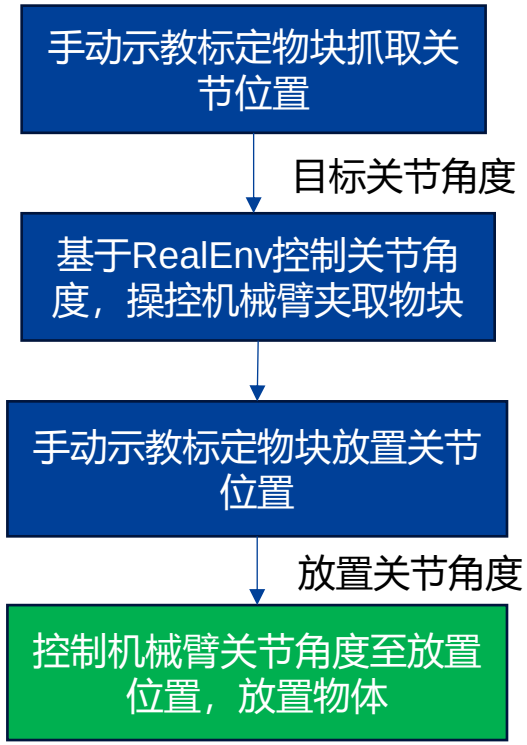
手动示教放置位置演示视频



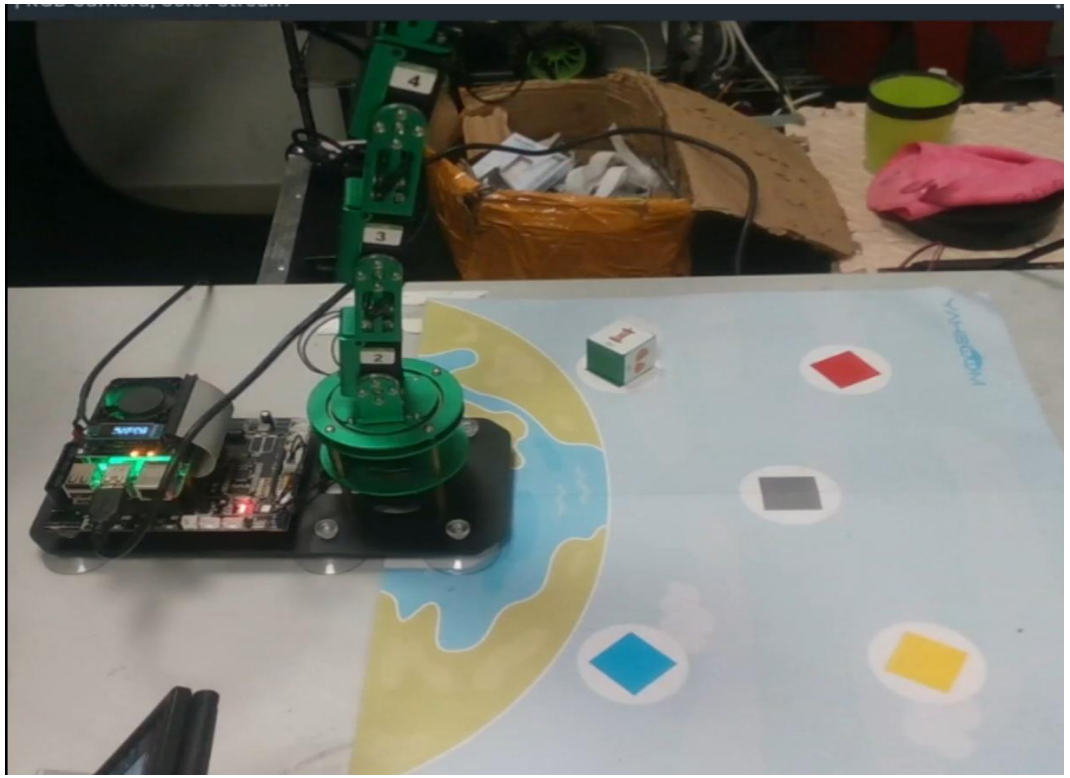
放置位置示意

作业-实现思路参考（实机部分）

步骤4：控制机械臂关节角度至步骤3得到的放置位置，放置物体



任务pipeline



完整抓取放置demo效果

谢谢！