汕 头 大 学 工 学 院

项 目 报 告

课 程 名 称 ： Artificial intelligence and robotics

项目题目： Probabilistic Robotics Project

指 导 教 师 ： 范衡

系　　别： 电子工程系　专业： 电子信息工程

	姓名	学号
成员	劳俊杰	2022631111
	卢凯锋	2022631121
	卢远金	2022631107

一、项目要求

（1）概率机器人三级项目要求

为了能让让机器人研发者专注于自己擅长的领域，其他模块则直接复用相关领域更专业研发团队的实现，当然自身的研究也可以被他人继续复用，ROS 应运而生。ROS 是一套机器人通用软件框架，可以提升功能模块的复用性，并且随着 ROS2 的推出，ROS 日臻完善，是机器人软件开发的不二之选。

自 ROS 诞生的十几年来，不管是机器人相关软件、硬件还是 ROS 社区都发生了天翻地覆的变化，加之 ROS1 存在一些设计上的先天性缺陷，各种内外因素叠加下，导致 ROS1 在许多应用场景下都已经显得力不从心了。此背景下，官方于 2017 正式推出了新一代机器人操作系统——ROS2，ROS2 基于全新的设计框架，保留了 ROS1 的优点并改进其缺陷，ROS2 的目标是适应新时代的新需求。

概率机器人三级项目要求使用 ROS 来完成 topic 通讯程序编写以及 TF 广播器和监听器的编程实现。题目一：控制乌龟做圆形运动熟悉 ROS 当中的 topic 通讯，并自主编写控制程序，实现乌龟的矩形运动。题目二：熟悉 ROS 中 TF 坐标变换的实现机制广播与监听的编程实现，并自主编写代码实现 TF 广播器和监听器，实现乌龟跟随移动。

（2）基于 matlab 的四轴机械臂仿真及轨迹规划

机械臂作为一个多输入多输出、高度非线性、强耦合的复杂系统，凭借其独特的操作灵活 性已在工业装配、安全防爆等领域广泛应用。在之前的学习中，我们已经了解了机械臂的空间坐 标描述、正运动学、逆运动学以及雅克比矩阵。通过掌握这些基本概念，我们现在将进一步运用 MATLAB 来实现一个三自由度机械臂控制的仿真。在本项目中，我们将探索如何使用 MATLAB 的强大功能，结合先进的算法和技术，实现对机械臂运动的精确控制。

通过仿真，我们将能够观察机械臂在不同位置和姿态下的运动轨迹，理解运动学求解的过 程，以及掌握运动规划和路径规划的基本方法。此外，我们还将考虑机械臂的运动性能、稳定性 和抗干扰能力，以提高系统的精确性和鲁棒性。在完成本项目后，我们将具备更深入的机械臂控 制理论和实践经验，为未来的机器人研究和应用奠定坚实的基础。

题目要求：

（1）项目代码能够控制一个 4 自由度的机械臂完成正向运动。

（2）项目代码要有足够的注释来解释它是怎么工作的。

（3）项目代码要包括控制过程的可视化，展示机械臂在 3D 图形环境中移动。

（4）在实现机械臂简单的移动任务的基础上，希望能够控制机械臂，以指定的线速度、角速度 或指定轨迹完成运动。（非必做）

（3）人工智能与机器人 2023 秋多视图几何三级项目

题目一：基于特征点匹配的目标检测（对比说明 SURF,ORB,BRISK 算法的性能）

题目二：基于 MATLAB 的图像校正

二、项目实现步骤

（1）概率机器人三级项目

题目一：控制乌龟做圆形运动熟悉 ROS 当中的 topic 通讯，并自主编写控制程序，实现乌龟的矩形运动。

ROS 的话题（Topics）通信机制是基于发布/订阅模型实现的异步通信方式，用于在 ROS 节点之间传输消息。它是 ROS 中最常用的通信机制之一，非常适合实时数据的传输。

在 ROS 中，节点可以发布（Publish）消息到一个话题，并且其他节点可以订阅（Subscribe）该话题来接收消息。发布者和订阅者之间是松耦合的，它们不需要知道对方的存在，只需要关注同一个话题名字即可。

对于如何控制乌龟做矩形运动，首先需要创建一个节点作为发布方，向话题'turtle1/cmd_vel'发布<geometry_msgs::msg::Twist>类型的消息，其中，'turtle1/cmd_vel'为与乌龟运动相关的话题，<geometry_msgs::msg::Twist>为话题中运动消息的类型。

然后 ROS2 可以用节点创建一个定时器用于循环发布话题消息。定时器周期进行调用回调函数，内部可利用 ROS2 节点发布方的方法进行用户自定义的消息发布。

具体而言，为了让乌龟进行矩形运动，首先需要先让乌龟直线运动一段时间(编程中设定 count=5)，然后给乌龟有一定的加速度进行专项，循环执行以上操作即可完成乌龟矩形运动。

题目二：熟悉 ROS 中 TF 坐标变换的实现机制广播与监听的编程实现，并自主编写代码实现 TF 广播器和 监听器，实现乌龟跟随移动。

ROS2 内部封装实现了 TF 坐标变换的实现机制广播与监听的相关方法函数。为了调用这些函数完成题目二，我们需要理解 ROS2 的参数服务功能、TF 广播器、TF 监听器。

首先，实现乌龟跟随移动需要两只乌龟，其中一只乌龟 1 可以调用 ros2 run turtlesim turtle_teleop_key 来控制乌龟的移动，另一只乌龟 2 通过接受 TF 监听器节点作为发布方发布的运动消息，跟随乌龟 1 的移动。由于 ros2 run turtlesim turtlesim_node 调用后只会生成一只乌龟，因此我们需要编写一个程序调用 turtlesim 相关参数服务，调用 spawn 生成第二只乌龟，编写内容主要是使用参数服务声明新的乌龟的信息，然后创建服客户端去连接服务端，最后组织发布乌龟信息，即可生成一只新乌龟。

然后我们需要两个节点分别作为 TF 广播器进行广播两只乌龟相对于 word 世界坐标系的位姿关系。其中，"tf2_ros/transform_broadcaster.h" 是创建动态广播器的类型相关的库函数，"geometry_msgs/msg/transform_stamped.hpp" 是动态广播器能广播出去的话题类型的库函数，"tf2/LinearMath/Quaternion.h"是转换欧拉角--四元数相关库函数。

最后我们需要建立一个节点作为 TF 监听器，调用监听器方法去监听广播器接受位姿关系，并调用另一个方法自动完成位姿关系计算完成 TF 坐标变换。最后该节点同样可以创建一个广播器，用于将坐标变换后换算的速度等运动消息，向'turtle2/cmd_vel'也就是第二只乌龟发布运动消息，即可让乌龟 2 完成对乌龟 1 的跟随运动。

（2）基于matlab的四轴机械臂仿真及轨迹规划

关节空间（Joint Space）

关节空间是指机械臂各个关节的状态所构成的空间。在这个空间中，机械臂的状态由关节角度或关节位置来描述。关节空间的坐标表示机械臂各个关节的状态，例如，对于一个四轴关节型机械臂，关节空间的坐标可以是四个关节的角度值。

笛卡尔空间（Cartesian Space）

笛卡尔空间是指机械臂末端执行器在三维空间中的位置和姿态构成的空间。在这个空间中，机械臂的状态由末端执行器的位置和姿态来描述。笛卡尔空间的坐标表示机械臂末端执行器在空间中的具体位置和朝向。

机械臂运动学：了解机械臂的运动学模型是非常重要的。这包括了正向运动学（从关节角度计算末端执行器位置）和逆向运动学（从末端执行器位置计算关节角度）。

逆运动学控制：熟悉逆运动学控制方法，这是实现机械臂末端执行器移动到特定位置的关键。逆运动学控制使用机械臂的逆向运动学方程来计算所需的关节角度。

末端执行器是机械臂的最后一个部件，位于机械臂的末端。它通常是用来实现机械臂与环境之间的交互和操作。末端执行器可以采用不同形式，如夹爪、吸盘、工具等，根据具体应用的需求来选择。末端执行器的功能可以包括抓取、放置、搬运、装配等，使机械臂能够完成特定任务。在控制机械臂时，我们通常会通过控制末端执行器的位置和姿态来实现所需的操作。

MATLAB Robotics Toolbox：学习如何使用 MATLAB Robotics Toolbox 工具箱来定义、模拟和控制机械臂。掌握该工具箱的基本功能和各种函数的用法帮助快速进行开发和调试。

具体步骤：

1.确定机械臂类型：首先确定要使用的机械臂的类型和模型。

2.定义机械臂模型：使用 Robotics Toolbox 工具箱提供的函数来定义机械臂的运动学和动力学模型，通常涉及指定关节参数、连接方式和坐标系等信息。

3.设置目标位置：确定机械臂末端执行器需要移动到的目标位置。涉及指定一个目标点的初末位置坐标或者通过代码绘制一个轨迹。

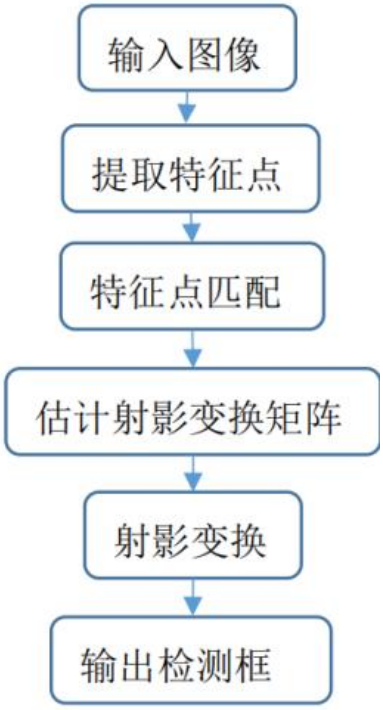
4.控制器设计：选择适当的控制方法和算法来使机械臂达到目标位置。包括逆运动学控制、PID 控制或其他高级控制方法。

5.仿真和实验平台：使用 Robotics Toolbox 提供的仿真功能和机械臂实验平台，对控制器进行测试和验证。通过对仿真结果的观察和实际实验的反馈，改进和优化控制器性能。

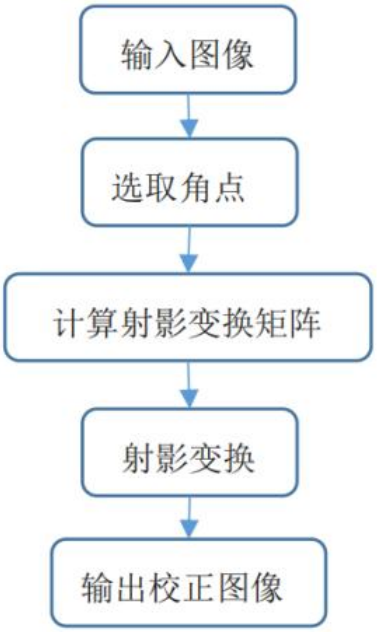
6.可视化和数据处理：学习如何可视化机械臂的运动轨迹以及关节角度变化。掌握如何处理和分析机械臂的传感器数据，如位置、速度和加速度等数据，以便于更好地监控和调整控制器。

（3）人工智能与机器人 2023秋多视图几何三级项目

题目一：



题目二：



三、项目结果与最终实现代码

1、最终实现代码

（1）概率机器人三级项目

题目一：

****使用 ROS2 编写 Rectangular.cpp****

```
#include <chrono>
#include <memory>

#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"
#include "geometry_msgs/msg/twist.hpp"

#define PI 3.14159265358979323846

//topic:turtle1/cmd_vel message_type:[geometry_msgs/msg/Twist]

using namespace std::chrono_literals;

class MinimalPublisher : public rclcpp::Node
{
  public:
    MinimalPublisher()//node
    : Node("minimal_publisher"), count_(0)
    {
      publisher_ = this->create_publisher<geometry_msgs::msg::Twist>("turtle1/cmd_vel", 10);//init
      timer_ = this->create_wall_timer(
      500ms, std::bind(&MinimalPublisher::timer_callback, this));
    }

  private:
    void timer_callback()
    {
```

```
        static int count = 0;
        geometry_msgs::msg::Twist speed;
        speed.linear.x = 1; // 设置线速度为 1m/s，正为前进，负为后退
        speed.linear.y = 0;
        speed.linear.z = 0;
        speed.angular.x = 0;
        speed.angular.y = 0;
        speed.angular.z = 0;

        count++;
        while(count == 5)
        {
            count=0;
            speed.linear.x = 1;
            speed.linear.y = 0;
            speed.linear.z = 0;
            speed.angular.x = 0;
            speed.angular.y = 0;
            speed.angular.z = PI; //转 90°
        }
        publisher_->publish(speed);
    }
    rclcpp::TimerBase::SharedPtr timer_;
    rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr publisher_;
    size_t count_;
};

int main(int argc, char * argv[])
{
    rclcpp::init(argc, argv);
    rclcpp::spin(std::make_shared<MinimalPublisher>());
    rclcpp::shutdown();
    return 0;
}
```

题目二：

****使用 ROS2 编程 spawn.cpp****

```
#include <chrono>
#include <memory>

#include "rclcpp/rclcpp.hpp"
#include "turtlesim/srv/spawn.hpp"//
#include "tf2_ros/transform_listener.h"//创建监听器的类型
#include "std_msgs/msg/string.hpp"
//1.使用参数服务声明新的乌龟的信息
//2.创建服务客户端
//3.连接服务端
//4.组织并发送数据

using namespace std::chrono_literals;

class Spawn : public rclcpp::Node
{
   public:
      Spawn(): Node("Spawn_node")
      {
      //1.使用参数服务声明新的乌龟的信息
      this->declare_parameter("x", 3.0);
      this->declare_parameter("y", 3.0);
      this->declare_parameter("theta", 0.0);
      this->declare_parameter("turtle_name", "turtle2");
      x = this->get_parameter("x").as_double();
      y = this->get_parameter("y").as_double();
      theta = this->get_parameter("theta").as_double();
      turtle_name = this->get_parameter("turtle_name").as_string();
      //2.创建服务客户端
      spawn_client_ = this->create_client<turtlesim::srv::Spawn>("/spawn");
```

```
    }
    //3.连接服务端
    bool connect_server()
    {
    while(!spawn_client_->wait_for_service(1s))
    {
        if(!rclcpp::ok())
        {
        RCLCPP_INFO(rclcpp::get_logger("rclcpp"), "强制退出!");
        return false;
        }
        RCLCPP_INFO(this->get_logger(), "服务连接中.....");
    }
    return true;
    }

    //4.组织并发送数据
    rclcpp::Client<turtlesim::srv::Spawn>::FutureAndRequestId request()
    {
    auto req = std::make_shared<turtlesim::srv::Spawn::Request>();

    req->x = x;
    req->y = y;
    req->theta = theta;
    req->name = turtle_name;

    return spawn_client_->async_send_request(req);
    }
  private:
      double_t x, y, theta;
      std::string turtle_name;
      rclcpp::Client<turtlesim::srv::Spawn>::SharedPtr spawn_client_;
};
```

```
int main(int argc, char * argv[])
{
    rclcpp::init(argc, argv);
    //由于节点任务是发布参数服务，服务完就可以关闭了，因此不需要 span 函数
    //rclcpp::spin(std::make_shared<TF_listener>());
    //创建自定义节点类对象，组织函数，处理响应结果
    auto client_ = std::make_shared<Spawn>();
    bool flag = client_->connect_server();
    if(!flag)
    {
    RCLCPP_INFO(rclcpp::get_logger("rclcpp"), "服务连接失败");
    return 1;
    }
    //发送请求
    auto response = client_->request();
    //处理响应
    if(rclcpp::spin_until_future_complete(client_, response) == rclcpp::FutureReturnCode::SUCCESS)
    {
    RCLCPP_INFO(client_->get_logger(), "响应成功！");
    std::string name = response.get()->name;
    if(name.empty())
    {
        RCLCPP_INFO(client_->get_logger(), "生成的乌龟因为重名而生成失败！");
    }
    else
    {
        RCLCPP_INFO(client_->get_logger(), "生成乌龟成功！");
    }
    }
    else
    {
    RCLCPP_INFO(client_->get_logger(), "响应失败！");
    }
    rclcpp::shutdown();
```

```
    return 0;
}
```

使用 ROS2 编程 tf_broadcaster.cpp

```
#include <chrono>
#include <memory>

#include "rclcpp/rclcpp.hpp"
#include "tf2_ros/transform_broadcaster.h"//创建动态广播器的类型
#include "turtlesim/msg/pose.hpp"//订阅乌龟 1 的 pose 话题
#include "geometry_msgs/msg/transform_stamped.hpp"//动态广播器能广播出去的话题类型
#include "tf2/LinearMath/Quaternion.h"//转换欧拉角--四元数
#include "std_msgs/msg/string.hpp"


using namespace std::chrono_literals;

class TF_broadcaster : public rclcpp::Node
{
  public:
    TF_broadcaster(): Node("TF_broadcaster")
    {
      this->declare_parameter("turtle", "turtle1");
      turtle = this->get_parameter("turtle").as_string();//乌龟名字动态获取

      broadcaster_ = std::make_shared<tf2_ros::TransformBroadcaster>(this);//创建一个广播器
      pose_sub_ = this->create_subscription<turtlesim::msg::Pose>("/" + turtle + "/pose", 10,
        std::bind(&TF_broadcaster::do_pose,this,std::placeholders::_1));//订阅乌龟 1 位姿关系
    }

  private:
    std::string turtle;
    void do_pose(const turtlesim::msg::Pose &pose)
    {
```

```
    geometry_msgs::msg::TransformStamped ts;//获取乌龟 1 位姿相对 world 的关系并发布
    ts.header.stamp = this->now();
    ts.header.frame_id = "world";

    ts.child_frame_id = turtle;

    ts.transform.translation.x = pose.x;
    ts.transform.translation.y = pose.y;
    ts.transform.translation.z = 0.0;

    tf2::Quaternion qtn;
    qtn.setRPY(0, 0, pose.theta);
        ts.transform.rotation.x = qtn.x();
        ts.transform.rotation.y = qtn.y();
        ts.transform.rotation.z = qtn.z();
        ts.transform.rotation.w = qtn.w();

    broadcaster_->sendTransform(ts);
    }
    std::shared_ptr<tf2_ros::TransformBroadcaster> broadcaster_;
    rclcpp::Subscription<turtlesim::msg::Pose>::SharedPtr pose_sub_;

};

int main(int argc, char * argv[])
{
    rclcpp::init(argc, argv);
    rclcpp::spin(std::make_shared<TF_broadcaster>());
    rclcpp::shutdown();
    return 0;
}
```

使用 ROS2 编程 tf_listener.cpp

```
#include <chrono>
```

```
#include <memory>

#include "rclcpp/rclcpp.hpp"
#include "tf2_ros/buffer.h"//缓存器,可以对多个广播器的内容进行存储
#include "tf2_ros/transform_listener.h"//创建监听器的类型
#include "geometry_msgs/msg/twist.hpp"
#include "std_msgs/msg/string.hpp"


using namespace std::chrono_literals;

class TF_listener : public rclcpp::Node
{
  public:
    TF_listener(): Node("TF_listener")
    {
        //声明参数服务
        this->declare_parameter("father_frame", "turtle2");
        this->declare_parameter("child_frame", "turtle1");
        father_frame = this->get_parameter("father_frame").as_string();
        child_frame = this->get_parameter("child_frame").as_string();

        buffer_ = std::make_unique<tf2_ros::Buffer>(this->get_clock());
    listener_ = std::make_shared<tf2_ros::TransformListener>(*buffer_, this);
    timer_ = this->create_wall_timer(1s, std::bind(&TF_listener::on_timer, this));
    cmd_pub_ = this->create_publisher<geometry_msgs::msg::Twist>("/" + father_frame + "/cmd_vel",
10);
    }

  private:
    void on_timer()
    {
    try
    {
```

```
        //实现坐标变换
        auto ts = buffer_->lookupTransform(father_frame, child_frame, tf2::TimePointZero);
        RCLCPP_INFO(this->get_logger(), "坐标转换完成");
        RCLCPP_INFO(this->get_logger(),
                    "frame:%s, child_frame:%s,偏移量(%0.2f, %0.2f, %0.2f)",
                    ts.header.frame_id.c_str(),
                    ts.child_frame_id.c_str(),
                    ts.transform.translation.x,
                    ts.transform.translation.y,
                    ts.transform.translation.z);
        //组织并发布速度指令
        geometry_msgs::msg::Twist twist;

        twist.linear.x = 0.5 * sqrt( pow(ts.transform.translation.x, 2) + pow(ts.transform.translation.y, 2));
        twist.angular.z = 1.0 * atan2(ts.transform.translation.y, ts.transform.translation.x);

        cmd_pub_->publish(twist);
    }
    catch(const tf2::LookupException& e)
    {
        RCLCPP_INFO(this->get_logger(), "异常提示：%s", e.what());
    }
    }
    std::string father_frame;
    std::string child_frame;
    std::unique_ptr<tf2_ros::Buffer> buffer_;
    std::shared_ptr<tf2_ros::TransformListener> listener_;
    rclcpp::TimerBase::SharedPtr timer_;
    rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr cmd_pub_;

};

int main(int argc, char * argv[])
{
```

```
    rclcpp::init(argc, argv);
    rclcpp::spin(std::make_shared<TF_listener>());
    rclcpp::shutdown();
    return 0;
}
```

由于题目二涉及许多节点的建立，为了方便，ROS2 允许使用 launch 文件来一次性建立多个节点，实现 ROS 高效开发。Launch 文件通常是 Python 文件

****使用 ROS2 编程 turtle_follows_launch.py****

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    #启动一个乌龟节点
    turtle =  Node(package="turtlesim", executable="turtlesim_node")
    #启动 spawn 节点召唤第二只乌龟
    spawn =  Node(package="turtle_follows", executable="spawn", parameters=[{"turtle_name":"t2"}])
    #广播两只乌龟相对 world 的坐标变换
    broadcaster1 = Node(package="turtle_follows", executable="tf_broadcaster", name="broad1")
    broadcaster2 = Node(package="turtle_follows", executable="tf_broadcaster", name="broad2",
parameters=[{"turtle":"t2"}])
    #创建监听节点
    listener = Node(package="turtle_follows", executable="tf_listener",
                    parameters=[{"father_frame":"t2", "child_frame":"turtle1"}])
    return LaunchDescription([turtle ,spawn,broadcaster1,broadcaster2,listener])
```

（2）基于 matlab 的四轴机械臂仿真及轨迹规划

机械臂的基础建模，实现能够控制机械臂完成正向移动

```
clc; % 清空命令窗口
clear; % 清空工作空间变量

%% 机械臂建模
% 定义各个连杆以及关节类型，默认为转动关节
```

```
%               theta       d          a          alpha
L1=Link([       0           0          0          pi/2], 'standard'); % 第一个连杆 DH 参数：关节角度 theta，运动方向 d，连杆长度 a，连杆扭转角 alpha
L2=Link([       0           0          0.105      0], 'standard'); % 第二个连杆 DH 参数
L3=Link([       0           0          0.09       0], 'standard'); % 第三个连杆 DH 参数
L4=Link([       0           0          0.04       0], 'standard'); % 第四个连杆 DH 参数
b=isrevolute(L1); % 检测关节是否为转动关节 1/0
robot=SerialLink([L1,L2,L3,L4],'name','Irvingao Arm'); % 将四个连杆组成机械臂
robot.name='kunkun's Robotic Arm';
view(3);
robot.teach();
```

对机械臂进行基础建模并实现直线移动

```
clc; % 清空命令窗口
clear; % 清空工作空间变量

%% 机械臂建模
% 定义各个连杆以及关节类型，默认为转动关节
%               theta       d          a          alpha
L1=Link([       0           0          0          pi/2], 'standard'); % 第一个连杆 DH 参数：关节角度 theta，运动方向 d，连杆长度 a，连杆扭转角 alpha
L2=Link([       0           0          0.105      0], 'standard'); % 第二个连杆 DH 参数
L3=Link([       0           0          0.09       0], 'standard'); % 第三个连杆 DH 参数
L4=Link([       0           0          0.04       0], 'standard'); % 第四个连杆 DH 参数
b=isrevolute(L1); % 检测关节是否为转动关节 1/0
robot=SerialLink([L1,L2,L3,L4],'name','Irvingao Arm'); % 将四个连杆组成机械臂
robot.name='kunkun's Robotic Arm';
robot.display();

%% 轨迹规划
% 初始值及目标值
init_ang=[0 0 0 0]; % 初始关节角度
targ_ang=[0, -pi/6, -pi/5, pi/6]; % 目标（结束）关节角度
step=200; % 轨迹离散点数
```

```
[q,qd,qdd]=jtraj(init_ang,targ_ang,step); % 关节空间规划轨迹（根据初始和结束关节角度），得到机器人末端运动的[位置，速度，加速度]
T0=robot.fkine(init_ang); % 正运动学解算，得到初始末端变换矩阵
Tf=robot.fkine(targ_ang); % 正运动学解算，得到目标末端变换矩阵

subplot(2,4,3); i=1:4; plot(q(:,i)); title("位置"); grid on; % 绘制关节角度随时间的变化
subplot(2,4,4); i=1:4; plot(qd(:,i)); title("速度"); grid on; % 绘制关节角速度随时间的变化
subplot(2,4,7); i=1:4; plot(qdd(:,i)); title("加速度"); grid on; % 绘制关节角加速度随时间的变化
 %提取轨迹
Tc=ctraj(T0,Tf,step); % 笛卡尔空间规划轨迹，得到机器人末端运动位置和姿态的变换矩阵（轨迹信息）
Tjtraj=transl(Tc); % 提取变换矩阵中的平移部分
subplot(2,4,8); %大窗口内同时显示多个图
plot2(Tjtraj, 'r'); % 在末端空间上绘制机器人的笛卡尔轨迹,r 红色
title('p1 到 p2 直线轨迹');
grid on;%打开子图的网格显示，以提高图形的可读性

subplot(2,4,[1,2,5,6]);
plot3(Tjtraj(:,1),Tjtraj(:,2),Tjtraj(:,3),"b"); grid on; % 在 3D 空间上绘制机器人的笛卡尔轨迹
hold on;
view(3); % 设置视图方向
qq=robot.ikine(Tc, 'q0',[0 0 0 0], 'mask',[1 1 1 1 0 0]); % 逆解算，得到关节角度
robot.plot(qq); % 绘制机器人在轨迹上的位置（机械臂移动可视化）

绘制圆形
%计算圆弧各点坐标
points = (center + radius*[cos(theta) sin(theta) zeros(size(theta))])' ;
plot3(points(1,:),points(2,:),points(3,:)-0.04,'r');
grid on;
hold on;
view(3);
%计算圆弧各点坐标对应变换算子
T = transl(points');
for i=1:201
```

```
    T(:,:,i) = T(:,:,i)*rpy2tr(-180,0,0);
End


%%画圆
L1 = Link('offset',0,'d', 0.16124+0.19026, 'a', 0.02970, 'alpha', pi/2,'qlim',[-150,150]/180*pi);
L2 = Link('offset',pi/2,'d', 0, 'a', 0.26043, 'alpha', 0,'qlim',[-120,120]/180*pi);
L3 = Link('d', 0, 'a', 0.04, 'alpha', pi/2,'offset',0,'qlim',[-100,100]/180*pi);
L4 = Link('d', 0.19451+0.07276, 'a', 0, 'alpha', -pi/2,'offset',0,'qlim',[-150,150]/180*pi);
L5 = Link('offset',-pi/2,'d', 0, 'a', 0, 'alpha', pi/2,'qlim',[-100,100]/180*pi);
L6 = Link('d', 0.14, 'a', 0, 'alpha', 0,'offset',0,'qlim',[-150,150]/180*pi);
n=3000;
bot = SerialLink([L1 L2 L5 L6], 'name', 'kunkun's Robotic Arm');
view(3);
bot.teach();
hold on;
%定义圆
N = (0:0.5:100)';
center = [0.28 0 0.4];
radius = 0.12;
theta = ( N/N(end) )*2*pi;
%计算圆弧各点坐标
points = (center + radius*[cos(theta) sin(theta) zeros(size(theta))])' ;
plot3(points(1,:),points(2,:),points(3,:)-0.04,'r');
grid on;
hold on;
view(3);
%计算圆弧各点坐标对应变换算子
T = transl(points');
for i=1:201
    T(:,:,i) = T(:,:,i)*rpy2tr(-180,0,0);
end
%进行逆运动学求解
q1 = bot.ikine(T,'mask',[1 1 1 1 1 1]);
bot.plot(q1,'movie','trail.gif');%保存
```

robot.plot(qq);

(3) 人工智能与机器人 2023 秋多视图几何三级项目

题目一：

SURF:

```
%% 清楚工作空间
clc;

%% 读取参考图片
WZImage = imread('2.jpg');
WZImage = rgb2gray(WZImage);
figure;
imshow(WZImage);
title('Image of a Wangzai');

%% 读取要处理的图片
WZCImageColor = imread('3.jpg');
WZCImage = rgb2gray(WZCImageColor);
figure;
imshow(WZCImage);
title('Image of a WangzaiCan');

%% 提取特征点
tic;
WZPoints = detectSURFFeatures(WZImage);
WZCPoints = detectSURFFeatures(WZCImage);
t_sys = toc;

%% Visualize the strongest feature points found in the reference image.
figure;
imshow(WZImage);
title('100 Strongest Feature Points from WZ Image');
hold on;
plot(selectStrongest(WZPoints, 100));

%% Visualize the strongest feature points found in the target image.
figure;
imshow(WZCImage);
title('300 Strongest Feature Points from WZC Image');
```

```
hold on;
plot(selectStrongest(WZCPoints, 300));

%% 提取特征描述
[WZFeatures, WZPoints] = extractFeatures(WZImage, WZPoints);
[WZCFeatures, WZCPoints] = extractFeatures(WZCImage, WZCPoints);

%% 找到匹配点
Pairs = matchFeatures(WZFeatures, WZCFeatures);

%% 显示匹配效果
matchedWZPoints = WZPoints(Pairs(:, 1), :);
matchedWZCPoints = WZCPoints(Pairs(:, 2), :);
figure;
showMatchedFeatures(WZImage, WZCImage, matchedWZPoints, ...
    matchedWZCPoints, 'montage');
title('Putatively Matched Points (Including Outliers)');

%% 通过匹配找到特定的物体
[tform, inlierIdx] = ...
    estgeotform2d(matchedWZPoints, matchedWZCPoints,
'affine');%https://www.bilibili.com/video/BV1n441147kH/?vd_source=e2d95ae1a4a936f9c286d
0e588576c09
inlierWZPoints = matchedWZPoints(inlierIdx,:);
inlierWZCPoints = matchedWZCPoints(inlierIdx,:);
%tform 是图像 1 内点转化为图像 2 内点的 affine 变换矩阵
%% 显示匹配效果
figure;
showMatchedFeatures(WZImage, WZCImage, inlierWZPoints, ...
    inlierWZCPoints, 'montage');
title('Matched Points (Inliers Only)');

%Get the bounding polygon of the reference image.
WZPolygon = [1, 1;...                           % top-left
       size(WZImage, 2), 1;...                  % top-right
       size(WZImage, 2), size(WZImage, 1);... % bottom-right
       1, size(WZImage, 1);...                  % bottom-left
       1, 1];                 % top-left again to close the polygon
newWZPolygon = transformPointsForward(tform, WZPolygon);
%将参考图 4 个点的位置通过 tform 矩阵进行变换，找出再检测图中参考图的位置。
```

```
%% 显示被检测到的物体
figure;
imshow(WZCImageColor);
hold on
line(newWZPolygon(:, 1), newWZPolygon(:, 2), Color = "blue");
title(['SURFDetected WZ 耗时(s)： ',num2str(t_sys)])
```

ORB:

```
%% 清楚工作空间
clc;

%% 读取参考图片
WZImage = imread('2.jpg');
WZImage = rgb2gray(WZImage);
figure;
imshow(WZImage);
title('Image of a Wangzai');

%% 读取要处理的图片
WZCImageColor = imread('3.jpg');
WZCImage = rgb2gray(WZCImageColor);
figure;
imshow(WZCImage);
title('Image of a WangzaiCan');

%% 提取特征点
tic;
WZPoints = detectORBFeatures(WZImage);
WZCPoints = detectORBFeatures(WZCImage);
t_sys = toc;

%% Visualize the strongest feature points found in the reference image.
figure;
imshow(WZImage);
title('100 Strongest Feature Points from WZ Image');
hold on;
plot(selectStrongest(WZPoints, 100));

%% Visualize the strongest feature points found in the target image.
```

```
figure;
imshow(WZCImage);
title('300 Strongest Feature Points from WZC Image');
hold on;
plot(selectStrongest(WZCPoints, 300));

%% 提取特征描述
[WZFeatures, WZPoints] = extractFeatures(WZImage, WZPoints);
[WZCFeatures, WZCPoints] = extractFeatures(WZCImage, WZCPoints);

%% 找到匹配点
Pairs = matchFeatures(WZFeatures, WZCFeatures);

%% 显示匹配效果
matchedWZPoints = WZPoints(Pairs(:, 1), :);
matchedWZCPoints = WZCPoints(Pairs(:, 2), :);
figure;
showMatchedFeatures(WZImage, WZCImage, matchedWZPoints, ...
    matchedWZCPoints, 'montage');
title('Putatively Matched Points (Including Outliers)');

%% 通过匹配找到特定的物体
[tform, inlierIdx] = ...
    estgeotform2d(matchedWZPoints, matchedWZCPoints, 'affine');
inlierWZPoints = matchedWZPoints(inlierIdx,:);
inlierWZCPoints = matchedWZCPoints(inlierIdx,:);
%% 显示匹配效果
figure;
showMatchedFeatures(WZImage, WZCImage, inlierWZPoints, ...
    inlierWZCPoints, 'montage');
title('Matched Points (Inliers Only)');

WZPolygon = [1, 1;...                           % top-left
        size(WZImage, 2), 1;...                 % top-right
        size(WZImage, 2), size(WZImage, 1);... % bottom-right
        1, size(WZImage, 1);...                 % bottom-left
        1, 1];                  % top-left again to close the polygon
newWZPolygon = transformPointsForward(tform, WZPolygon);
%t_sys = toc;

%% 显示被检测到的物体
```

```
figure;
imshow(WZCImageColor);
hold on
line(newWZPolygon(:, 1), newWZPolygon(:, 2), Color = 'b');
title(['OBRDetected WZ 耗时(s): ',num2str(t_sys)])
```

BRISK:

```
%% 清楚工作空间
clc;

%% 读取参考图片
WZImage = imread('2.jpg');
WZImage = rgb2gray(WZImage);
figure;
imshow(WZImage);
title('Image of a Wangzai');

%% 读取要处理的图片
WZCImageColor = imread('3.jpg');
WZCImage = rgb2gray(WZCImageColor);
figure;
imshow(WZCImage);
title('Image of a WangzaiCan');

%% 提取特征点
tic;
WZPoints = detectBRISKFeatures(WZImage);
WZCPoints = detectBRISKFeatures(WZCImage);
t_sys = toc;

%% Visualize the strongest feature points found in the reference image.
figure;
imshow(WZImage);
title('100 Strongest Feature Points from WZ Image');
hold on;
plot(selectStrongest(WZPoints, 100));

%% Visualize the strongest feature points found in the target image.
figure;
imshow(WZCImage);
```

```
title('300 Strongest Feature Points from WZC Image');
hold on;
plot(selectStrongest(WZCPoints, 300));

%% 提取特征描述
[WZFeatures, WZPoints] = extractFeatures(WZImage, WZPoints);
[WZCFeatures, WZCPoints] = extractFeatures(WZCImage, WZCPoints);

%% 找到匹配点
Pairs = matchFeatures(WZFeatures, WZCFeatures);

%% 显示匹配效果
matchedWZPoints = WZPoints(Pairs(:, 1), :);
matchedWZCPoints = WZCPoints(Pairs(:, 2), :);
figure;
showMatchedFeatures(WZImage, WZCImage, matchedWZPoints, ...
    matchedWZCPoints, 'montage');
title('Putatively Matched Points (Including Outliers)');
disp(WZCImage)

%% 通过匹配找到特定的物体
[tform, inlierIdx] = ...
    estgeotform2d(matchedWZPoints, matchedWZCPoints, 'affine');
inlierWZPoints = matchedWZPoints(inlierIdx,:);
inlierWZCPoints = matchedWZCPoints(inlierIdx,:);
%% 显示匹配效果
figure;
showMatchedFeatures(WZImage, WZCImage, inlierWZPoints, ...
    inlierWZCPoints, 'montage');
title('Matched Points (Inliers Only)');

WZPolygon = [1, 1;...                           % top-left
       size(WZImage, 2), 1;...                  % top-right
       size(WZImage, 2), size(WZImage, 1);... % bottom-right
       1, size(WZImage, 1);...                  % bottom-left
       1, 1];                  % top-left again to close the polygon
newWZPolygon = transformPointsForward(tform, WZPolygon);
%t_sys = toc;

%% 显示被检测到的物体
figure;
```

```
imshow(WZCImageColor);
hold on
line(newWZPolygon(:, 1), newWZPolygon(:, 2), Color = 'b');

title(['BRISKDetected WZ 耗时(s): ',num2str(t_sys)])
```

题目二：

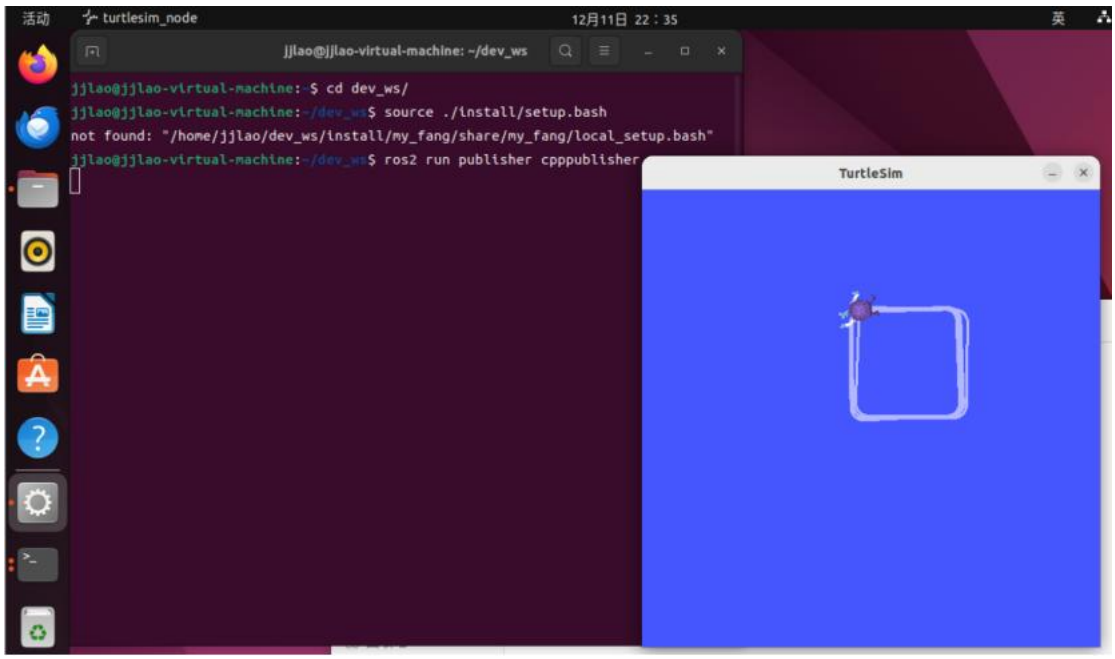
```
img = rgb2gray(imread('cpuls.jpg'));
width = size(img,2);
height = size(img,1);
figure;imshow(img)
moving_points = ginput(4);
hold on ; plot(moving_points(:,1),moving_points(:,2),'ro');
fixed_points = [0,0;
    100,0;
    0,200;
    100,200];

tfom = fitgeotrans(moving_points,fixed_points,'projective');
%X = moving_points(:,1);
%Y = moving_points(:,2);
%[x,y] = transformPointsForward(tfom,X(:),Y(:));
%figure;plot(x,y,'ro');title('验证坐标点对齐')
%grid on
tic;
dst_img = imwarp(img,tfom);
t_sys = toc;
figure;imshow(dst_img);title(['图像仿射变换后（系统函数），耗时(s)：',num2str(t_sys)])
```

2、项目最终实现效果：

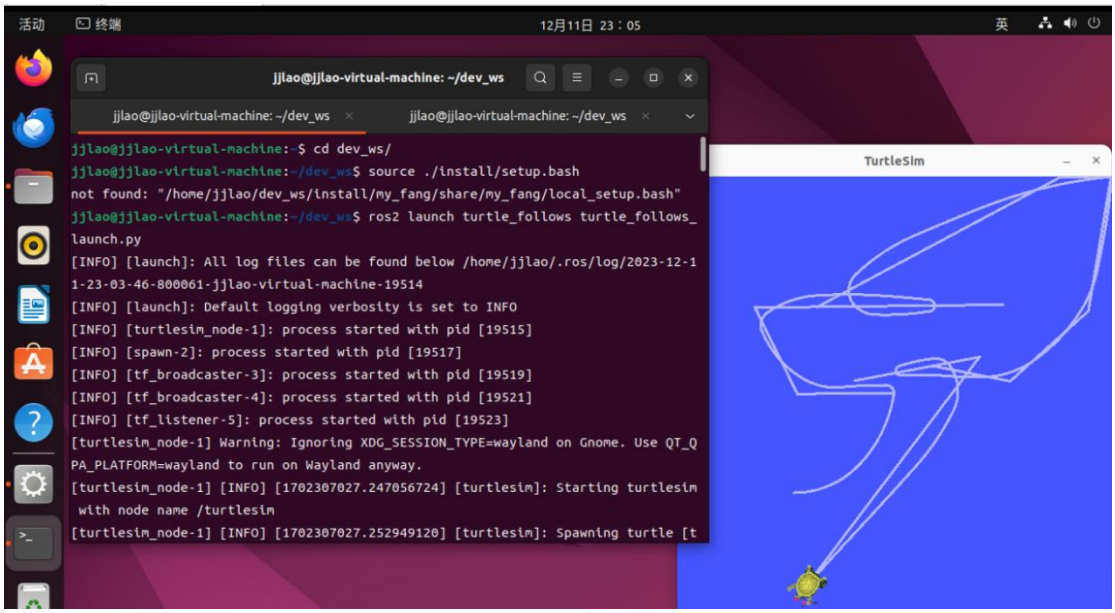
（1）概率机器人三级项目

题目一代码功能效果如图所示:



图中乌龟成功实现矩形运动。说明代码实现了乌龟运动消息的发布。

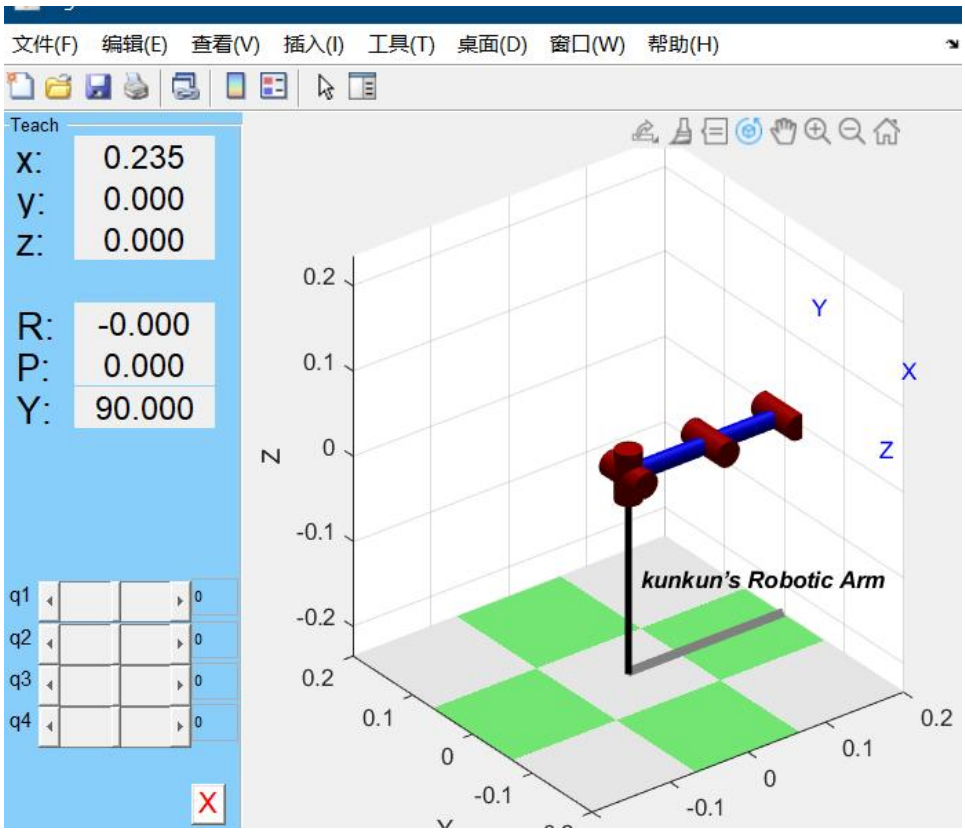
题目二代码功能效果如图所示：



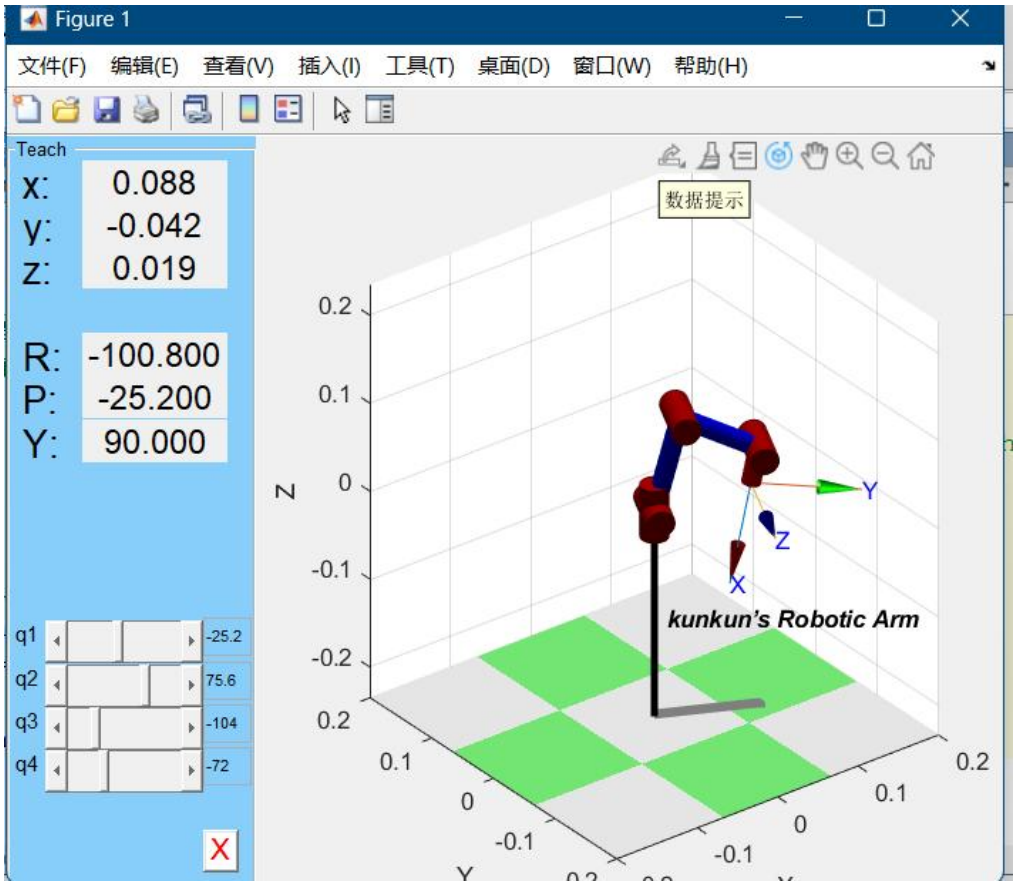
如图所示，乌龟 2 成功实现对乌龟 1 的跟随运动。

（2）基于 matlab 的四轴机械臂仿真及轨迹规划

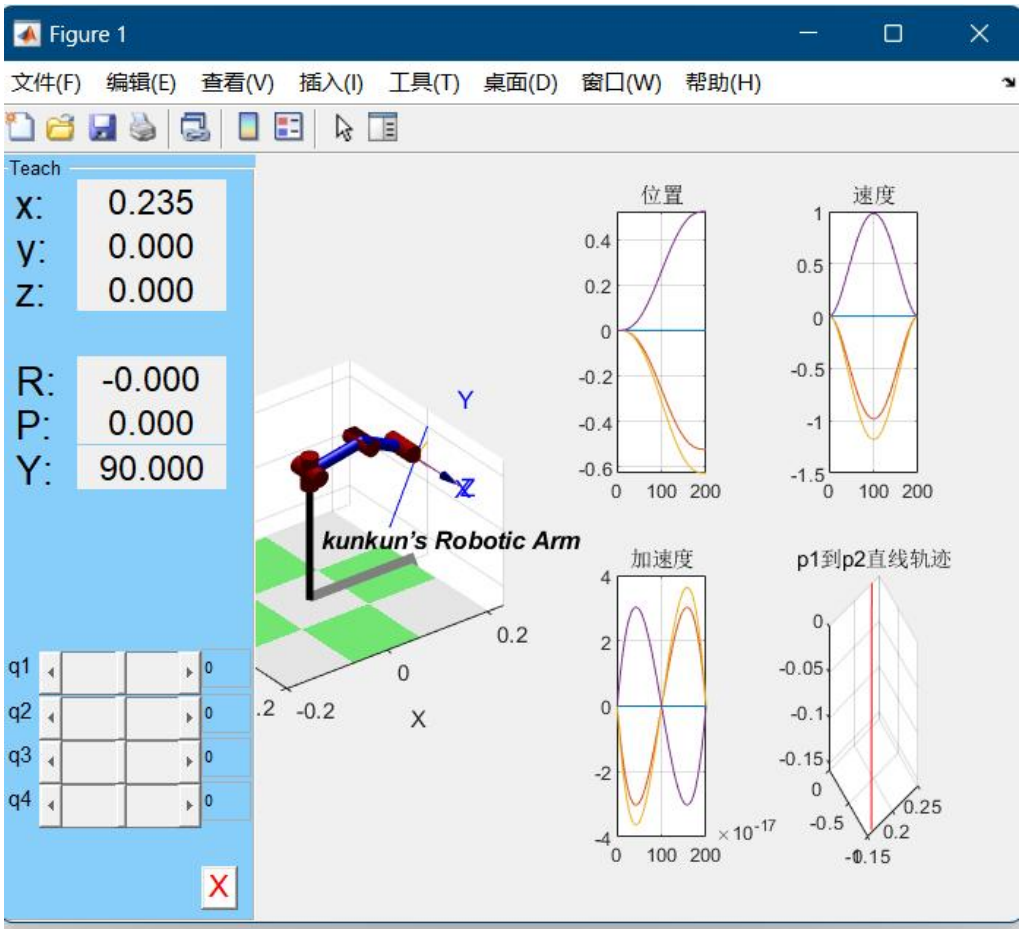
机械臂基础建模



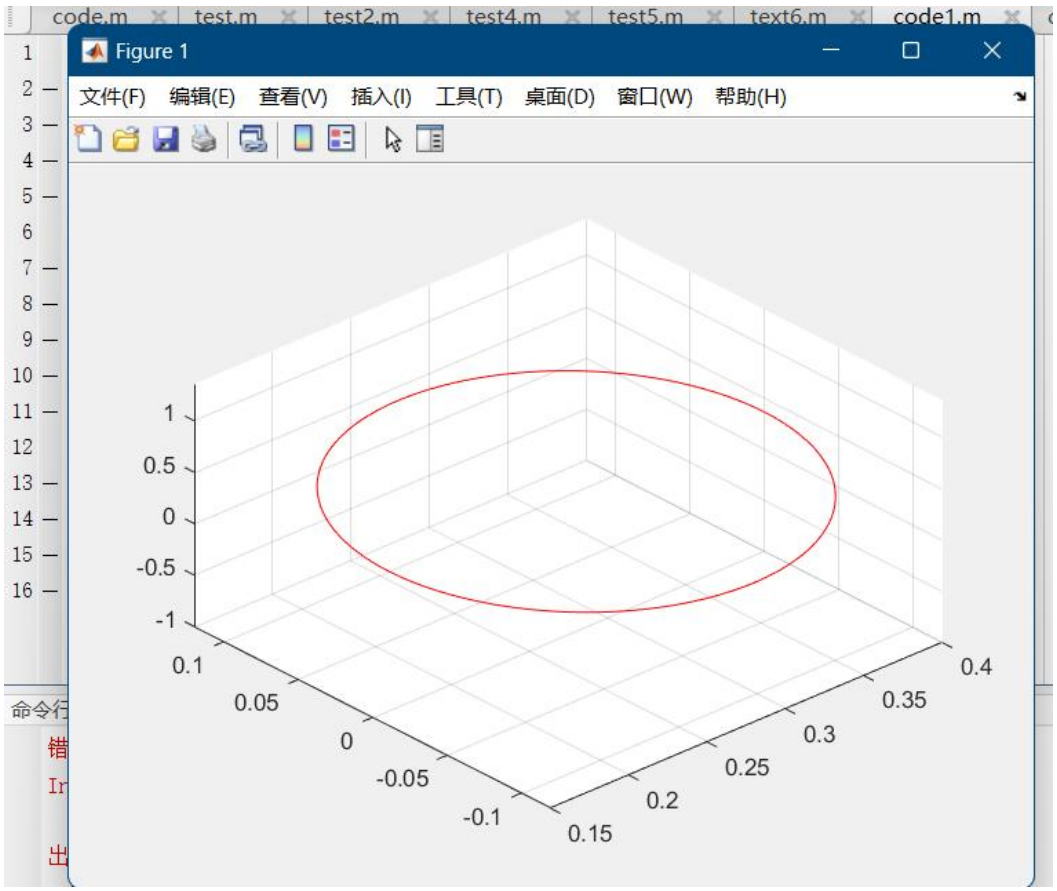
可完成对机械臂的基础调控及正向移动



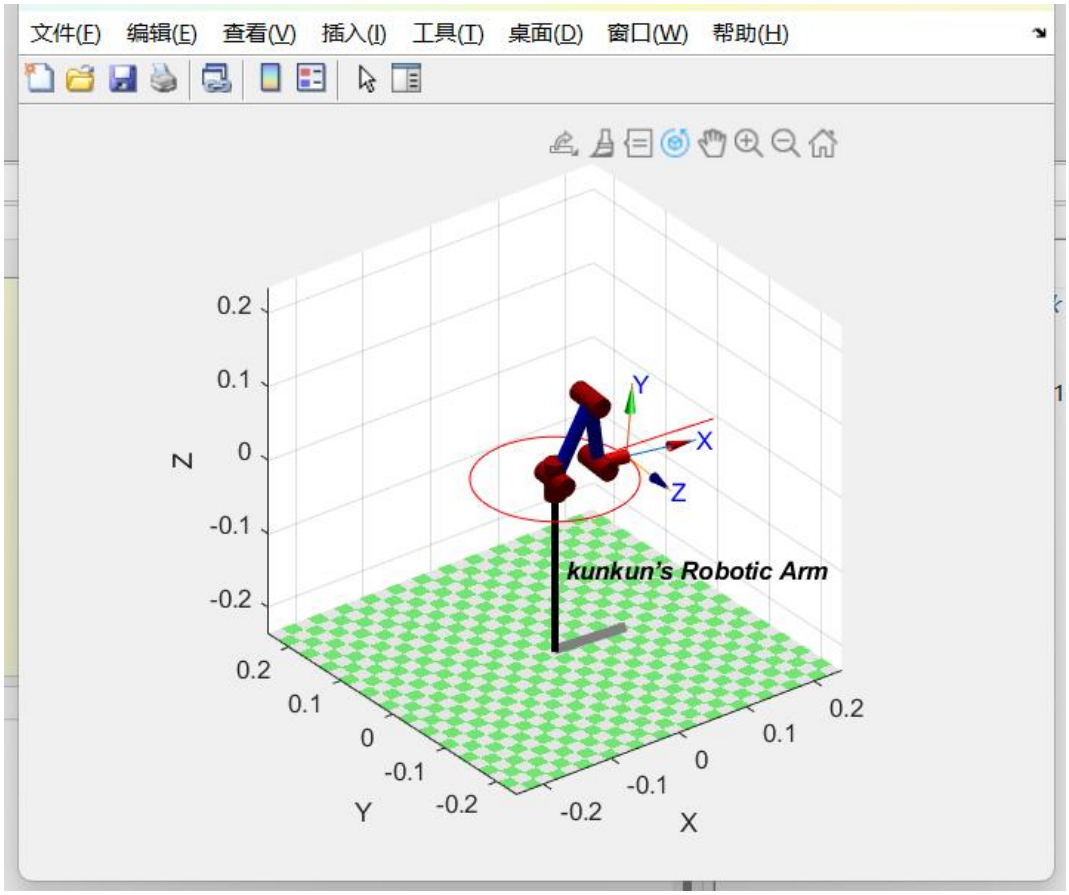
对机械臂进行直线轨迹规划，并可视化速度，加速度等数据



圆形绘制



机械臂圆形轨迹规划

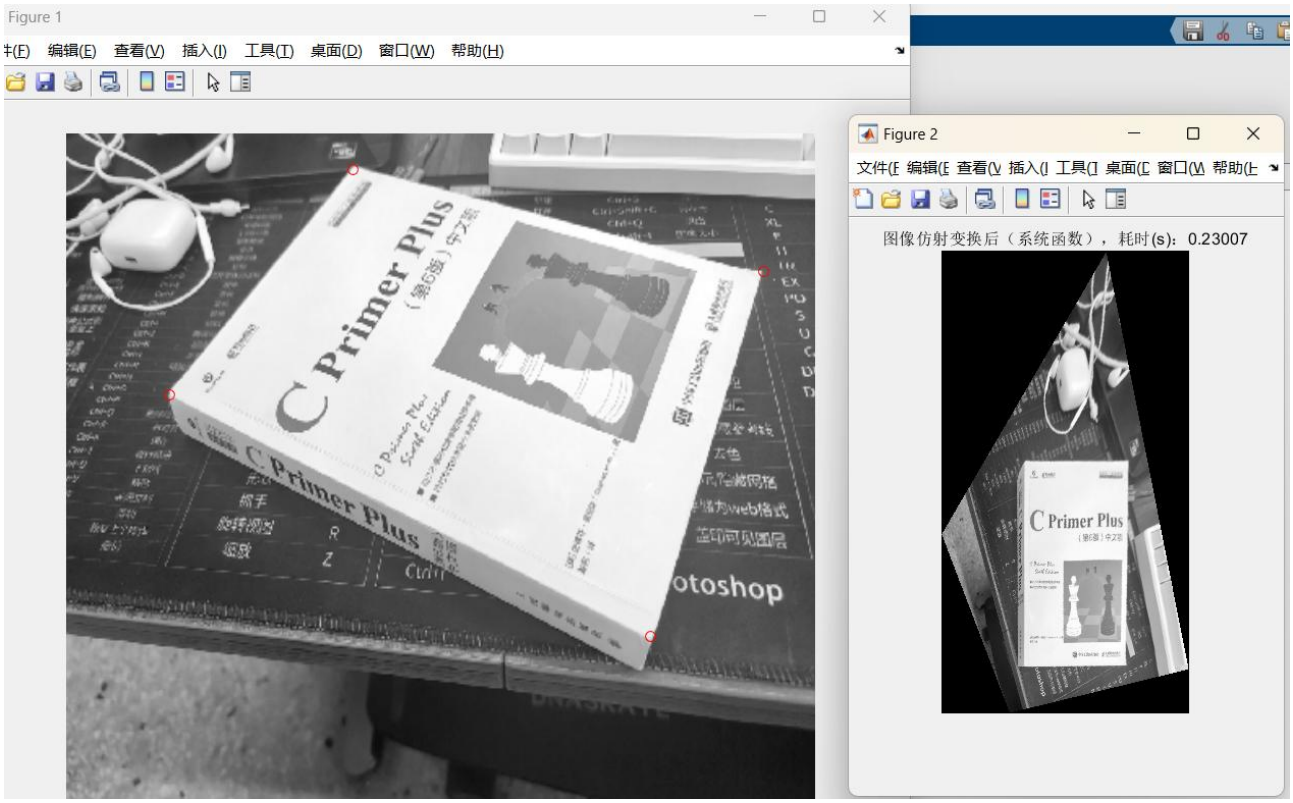


(3) 人工智能与机器人 2023 秋多视图几何三级项目

题目一：

三者之间，SURF 算法的旋转不变性、尺度不变性较好；但速度也较慢，而 ORB 速度最快，可用作实时检测；BRISIK 有较好的旋转不变性、尺度不变性，在对有较大模糊的图像配准时，BRISK 算法表现较为出色。

题目二：



四、项目总结

通过本次概率机器人三级项目的学习，我组使用 ROS2 完成题目一和题目二的程序编写我认为我们已经初步理解 ROS2 的相关开发框架，能进行 ROS2 相关知识技能的搜索和学习，能尝试进行相关简单程序的编写与学习，具备开发机器人系统最基本的能力。

在对机械臂的运动仿真中，我学习到了如何使用 MATLAB 的 Robotics Toolbox 工具箱进行了机械臂的运动仿真和轨迹规划。这个项目不仅为我提供了一个实际的应用场景，而且让我深入了解了机械臂运动控制的关键技术和挑战。机械臂运动学：通过对机械臂的各个关节和链接进行建模，我学习了如何描述和计算机械臂的运动学特性。轨迹规划算法：在项目中，我探索了不同的轨迹规划算法，如笛卡尔空间和关节空间的规划方法，以实现平滑和高效的机械臂运动。MATLAB Robotics Toolbox 的应用：这个工具箱为我提供了丰富的函数和工具，帮助我更高效地进行机械臂的仿真和控制。实时控制与仿真验证：通过模拟不同的工作场景和任务，我学习了如何实时控制机械臂并验证轨迹规划的准确性和可行性。

这个项目让我深刻理解了理论知识与实际应用之间的联系，实践中的挑战和解决方案使我受益匪浅。在项目中，与团队成员紧密合作和沟通是成功的关键。我学会了如何有效地与他人合作，共同解决问题。机械臂技术日新月异，我深知持续学习和创新的重要性，未来我将继续探索更先进的控制算法和技术应用。总之，这个项目不仅增强了我的技术能力和实践经验，而且为我未来的研究和职业发展打下了坚实的基础。我期待在未来能够将所学应用到

更多的实际问题中，为机械臂技术的发展做出贡献。

对计算机视觉的一些概念以及相关算法有了一定的了解，激发了学习兴趣，今后打算继续学习计算机视觉领域里面的目标检测，并将其应用到项目以及比赛中去。

五、参考文献及链接

[1].https://blog.csdn.net/qq_43616471/article/details/107855268?ops_request_misc=%257B%2522request%255Fid%2522%253A%252217025633451680018583289 3%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=170256334516800185832893&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-107855268-null-null.142^v96^pc_search_result_base9&utm_term=ORB%E7%AE%97%E6%B3%95&spm=1018.2226.3001.4187

[2]. https://blog.csdn.net/shenziheng1/article/details/72579635

[3].https://blog.csdn.net/weixin_41063476/article/details/90407916?ops_request_misc=&request_id=&biz_id=102&utm_term=brisk%E7%AE%97%E6%B3%95%E5%8E%9F%E7%90%86&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-1-90407916.nonecase&spm=1018.2226.3001.4187

[4]https://blog.csdn.net/ooorczgc/article/details/125110656

[5]夏伟,吴玉文.基于 MATLAB Robtics Toolbox 的机械臂轨迹仿真研究[J].河南科技,2020(04):54-56.

[6]周霏,陈富林,沈金龙等.基于 MATLAB 的四自由度机械臂运动学仿真研究[J].机械制造与自动化,2016,45(01):115-119.DOI:10.19344/j.cnki.issn1671-5276.2016.01.034.