六轴机械臂 Matlab 动力学仿真研究

刘丰源 付宛昕 Adelbert Reinhard Riang（梁沛桦）

梁家源 王天源 殷文韬

摘要：本项目围绕一台六自由度工业机械臂，依托 URDF 建模、Matlab Robotics System Toolbox 与 Simscape 联合仿真，系统完成了运动学建模、控制验证及复杂任务演示。首先，通过零件装配、关节约束与惯量参数标定，生成准确的 URDF 模型并经 urdf_check 初检合格。随后在 Matlab 中利用 fkine / ikine 及 Simscape 多体动力学分别验证正、逆运动学，三种求解（自编代码、工具箱、Simulink 模块）齐次变换矩阵与关节角度结果一致，证明模型可靠。借助两种图像处理方案（边界提取与像素坐标提取）将二维图案转换为末端轨迹，逆解得到的关节序列在 Simulink 中成功描绘“校徽”“兔子”等图形，末端路径与目标吻合。进一步实现了多约束抓取：构建位置、姿态及安全高度等笛卡尔约束，采用广义逆运动学规划五个路径点，生成平滑抓取轨迹。最后对比多项式插值、样条曲线与梯形速度三类规划，以及任务空间与关节空间两种执行策略，结果显示任务空间法轨迹精度更高而关节空间法实时性与平滑性更优。项目解决了多解选择、奇异与不可达位姿、参数调谐和工具箱安装权限等问题，并展望了自适应重规划、增材，混合制造及医疗等应用前景。

关键词：六轴机械臂；虚拟仿真技术；URDF 模型；Matlab 动力学仿真；机械臂绘图；轨迹规划算法；任务空间控制；关节空间控制；跨领域应用

0 研究动机

（1）工业自动化迫切需要高柔性装备

六轴机械臂因具备“六自由度+末端执行器”的空间可达性，已成为汽车、电子、物流等行业装配、焊接、搬运与检测的核心装备。企业在升级产线时，亟须验证机械臂对多类工艺的适应性与精度，因此必须先在虚拟环境中评估其运动特性和控制策略。

（2）仿真可显著降低设计-调试成本与风险

直接在实物上调试不仅昂贵，也伴随工伤、碰撞和停线风险；通过仿真，可在“零停机、零损耗”的条件下暴露结构缺陷、关节极限及奇异位姿，提前优化机构与控制参数，从而缩短研发周期并提升系统可靠性。

（3）满足《机器人建模与控制》课程实践深度

课程目标要求学生将运动学、动力学与控制理论融会贯通，完成“从 URDF 建模到 Simulink / Simscape 验证”的完整闭环。本项目选取六轴机械臂作为载体，可体系化地训练学生的正逆运动学推导、轨迹规划与多体动力学仿真能力，契合教学大纲。

（4）验证新型轨迹规划与多约束抓取算法的可行性

现代产线往往要求机械臂在狭窄空间内执行“快速-平滑-高精度”任务。通过仿真平台，可对多项式插值、样条曲线、梯形速度等规划方法，以及任务空间 vs. 关节空间两类控制策略进行对比，寻找精度、实时性与能耗的最佳折中，并测试广义逆运动学在多约束抓取上的稳定性。

（5）为未来跨领域拓展奠定基础

随着成本下降与感知技术进步，六轴机械臂正向农业采摘、餐饮服务、智慧仓储和教育培训等新场景渗透。提前在仿真中验证不同应用工况下的可行性，可加速机械臂由“工业专用”向“泛在助手”转变。

综上，六轴机械臂仿真研究兼顾了产业痛点（降本增效）、教学任务（能力培养）与技术前沿（先进规划算法与跨领域应用），是连接课堂理论与工业实践、当前需求与未来机遇的理想切入点。

1 研究流程

1.1 制作原始 urdf 文件

1.1.1 导入 step 文件

在本项目中，为了重用已有的机械臂结构模型，我们首先将 STEP 格式（.step/.stp）的零件文件导入至 SolidWorks 中。对于包含多个零件的 STEP 文件，SolidWorks 会自动识别其结构并生成装配体（.SLDASM），各子零件则分别作为独立的零件文件（.SLDPRT）加载。

图 1

导入完成后，借助 SolidWorks 提供的“Import Diagnostics”工具对模型进行几何修复，以避免缝隙或边界异常影响后续约束设置与仿真。此外，为便于后续的参数化管理，我们对各零件进行了重命名和材料属性设置，并将所有模型另存为 SolidWorks 本地格式（.SLDPRT/.SLDASM），以便后续进行装配约束、参考坐标系添加以及 URDF 文件导出等操作。

1.1.2 装配体整合

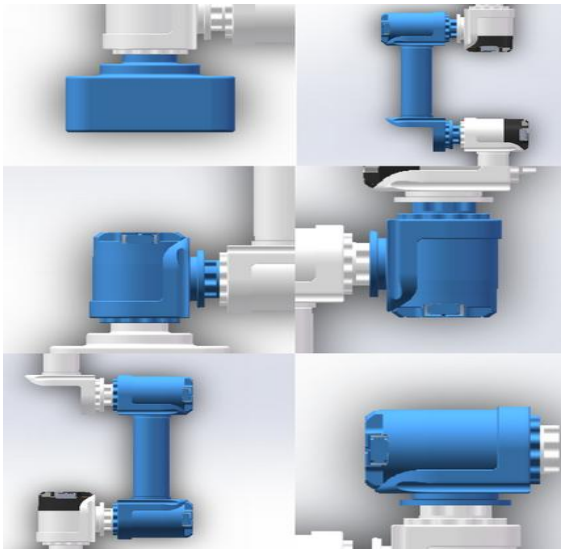


图 2

为确保装配的结构符合实际运动关系，我们依据机械臂的结构层次依次将离散模型整合为各连杆（Link）及关节组件，并从基座（Base）开始逐级向末端执行器进行组装。

1.1.3 装配体关系约束

在装配过程中，主要使用 SolidWorks 中的配合功能对零件之间的空间位置与运动关系进行约束，包括“同轴配合”、“面对齐配合”、“距离配合”等。通过合理添加配合关系，使各零部件之间形成符合实际运动自由度的相对连接。

1.1.4 基准轴和基准坐标建立

图 3

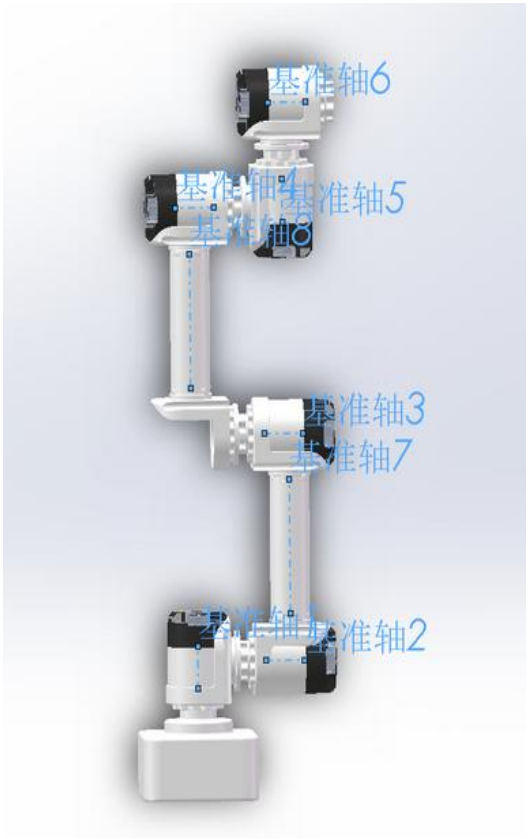


图 4

为实现机器人运动模型在仿真环境中的准确描述，本项目采用 Denavit－Hartenberg（DH）参数法建立六轴机械臂的关节空间关系。在 SolidWorks 装配体中，依据每个旋转关节的运动方向，我们分别在各连杆起始处设置基准轴（Reference Axis），用于定义该关节的 Z 轴方向，同时配合基准坐标系以构建完整的 DH 结构。

如图所示，图中已清晰标注出从底座到末端的八个基准轴，编号为 基准轴 1 至 基准轴 8，其中每个关节轴线依次对应一个旋转自由度，形成标准

的串联六轴结构。具体设定如下：

基准轴 1 与 基准轴 2 定义第 1 关节旋转轴方向；

基准轴 3 与 基准轴 7 定义第 2 关节的旋转方向；

基准轴 4 与 基准轴 8 连接至第 3 关节结构；

基准轴 5、6 则分别用于末端三轴关节的旋转约束，形成机械臂末端姿态控制系统。

在每个基准轴所在的零部件上，添加与之对齐的 基准坐标系（Reference Coordinate System），其 Z 轴对准关节轴线，X 轴定义为指向下一连杆的连接方向，Y 轴由右手规则确定，构成符合 DH 标准的坐标系定义。通过这种方式，能够确保机械臂在 URDF 文件导出和运动仿真中保持一致的空间描述，便于后续在 MATLAB Simscape 或 ROS 环境中执行正/逆运动学求解。

1.1.5 Urdf 文件参数设置

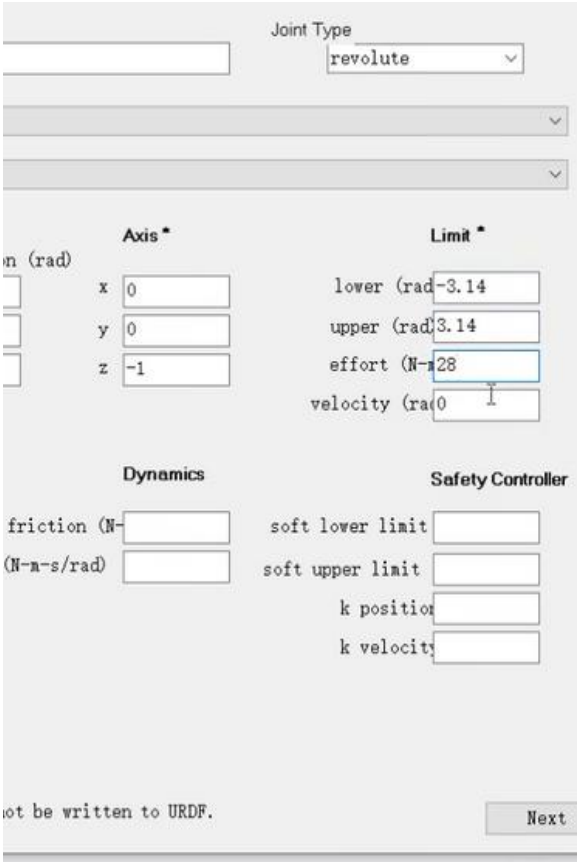


图 5

在机械臂装配体完成后，利用 SolidWorks URDF Exporter 插件导出 URDF 文件前，需要对每个关节进行精确的参数设置，以保证其在仿真环境中的运动学和动力学表现与实际设计一致。图中展示了 URDF 导出过程中某一旋转关节（Joint Type: revolute）的详细配置界面，主要包括 关节轴向（Axis）、运动范围限制（Limit）、动力学特性（Dynamics） 以及 安全控制器（Safety Controller）四类关键参数。

其中，Axis 部分设定了关节的旋转轴向，如图所示，该关节的旋转轴方向为 z=−1z = -1z=−1，代表关节绕 Z 轴负方向旋转，与先前在 SolidWorks 中定义的基准轴一致。Limit 部分设置了该关节的运动范围和性能限制，包括：

lower/upper：关节的最小与最大旋转角，图中设置为 −3.14～3.14-3.14 \sim 3.14−3.14～3.14（即 −180。 ～180。 -180^\circ \sim 180^\circ−180。 ～180。），允许关节实现完整旋转；

effort：表示该关节的最大扭矩限制，单位为 N • m，设置为 28；

velocity：关节最大转速（rad/s），暂未填写，可根据实际电机参数补充。

1. 2 检验与初步仿真

1.2.1 利用机器人工具箱

在 URDF 文件导出完成后，本项目进一步使用 MATLAB 中的 Peter Corke 机器人工具箱（Robotics Toolbox for MATLAB） 对模型进行正确性验证及轨迹仿真测试。首先，利用工具箱中的 importrobot() 函数将 URDF 文件导入为 SerialLink 或 RigidBodyTree 对象，通过 robot.teach() 或 robot.plot() 函数进行关节空间的可视化交互检查，验证连杆连接是否正确、关节轴向与设定是否一致。

为进一步检验模型的运动能力，我们设定机械臂末端沿笛卡尔空间执行圆形轨迹运动，作为动态仿真测试场景。具体方法包括：

在设定平面上生成一系列圆形轨迹点（如 X-Y 平面圆）；

利用 ikine() 函数对每个末端目标位姿进行逆解求解，获得对应的关节角解；

将得到的关节角序列输入 robot.plot(q_path)，执行多帧动画模拟末端轨迹。

仿真结果表明，末端执行器在任务空间内能够准确完成圆形路径运动，关节轨迹平稳，未出现运动突变或解耦错误。这一过程有效验证了所构建 URDF 模型的结构准确性和运动一致性，表明该模型已具备用于后续运动控制、轨迹规划与动力学分析的基础条件。

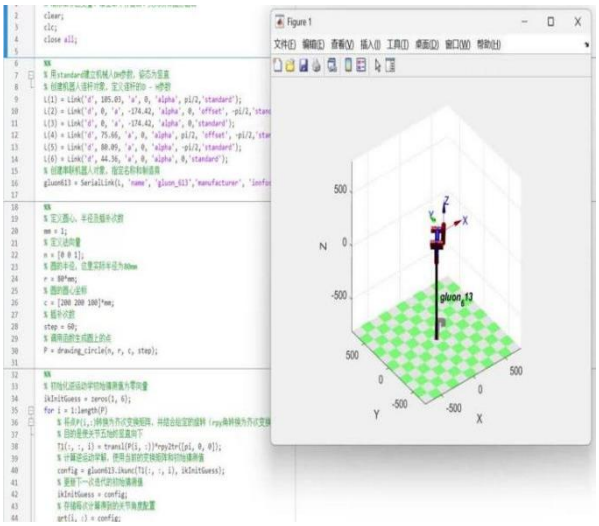


图 6

```
clear; clc; close all;

%% === 定义机器人DH参数（使用 standard DH） ===
L(1) = Link('d', 105.03, 'a', 0,       'alpha',  pi/2, 'standard');
L(2) = Link('d', 0,      'a', -174.42, 'alpha',  0,    'offset', -pi/2, 'standard');
L(3) = Link('d', 0,      'a', -174.42, 'alpha',  0,    'standard');
L(4) = Link('d', 75.66,  'a', 0,       'alpha',  pi/2, 'offset', -pi/2, 'standard');
L(5) = Link('d', 80.09,  'a', 0,       'alpha', -pi/2, 'standard');
L(6) = Link('d', 44.36,  'a', 0,       'alpha',  0,    'standard');

robot = SerialLink(L, 'name', 'gluon_613', 'manufacturer', 'innfos');

%% === 关节角输入并计算正解 ===
theta_deg = [30 30 30 30 30 30];        % 单位：度
theta = deg2rad(theta_deg);             % 转换为弧度

% 求末端正向运动学变换矩阵
T = robot.fkine(theta);

% 设定绘图工作空间范围
w = [-800, 800, -800, 800, -800, 800];

% 绘制机器人动画
robot.plot(theta, 'tilesize', 150, 'workspace', w);

%% === 验证逆解 ===
qi = rad2deg(robot.ikine(T));           % 逆解并转换为角度制

%% === 提取旋转矩阵和姿态角 ===
rot = t2r(T);                           % 提取旋转矩阵
rpy = tr2rpy(T) * 180 / pi;             % 提取 Roll-Pitch-Yaw 姿态角
trm = rpy2r(rpy, 'deg');                % 将 RPY 转回旋转矩阵
RPY = tr2rpy(trm, 'deg');               % 再次验证姿态角

%% === 四元数表示 ===
Q = UnitQuaternion(T);                  % 齐次矩阵转四元数表示

% 可视化 GUI 教学界面（启用姿态角控制）
% robot.teach('rpy');
```

图 7

1.2.2 利用 robot system toolbox

首先，使用 importrobot() 函数将导出的 URDF 文件加载至 MATLAB 工作区，生成一个 rigidBodyTree 对象。在此基础上，通过 show() 可视化机器人结构，并使用 checkCollision、showdetails 等函数进一步检查模型的连杆配置、关节属性与坐标系对齐情况，确保其与 SolidWorks 中的装配保持一致。

随后，基于 inverseKinematics 类构建逆运动学求解器，并结合设定的末端目标轨迹（如圆形路径或线性插值轨迹），计算出一系列合法的关节角度序列。为验证其动态性能，我们将逆解结果作为输入引入至 Simulink 模型中，并调用 Joint Space Motion Model、Rigid Body Tree 模块构建完整的动力学仿真结构。在 Simulink 中引入时间控制器和轨迹命令模块，可实现机器人末端按指定轨迹进行动态跟踪。

仿真结果表明，导入的 URDF 模型在 Robot System Toolbox 与 Simulink 环境中均能稳定运行，末端轨迹与设定路径高度吻合，且各关节响应平滑，符合预期的动态行为。这充分验证了前期建模过程的准确性，也展示了该建模方法在机器人正/逆运动学分析、轨迹规划和动态仿真中的有效性。

2 正运动学仿真

正运动学，是通过提供关节角度来得到末端执行器位置和姿态。齐次变换矩阵能够将一个坐标系下的点或向量转换到另一个坐标系下，通过依次乘以各关节的齐次变换矩阵，可以从机械臂的基座坐标系逐步转换到末端执行器坐标系，从而得到末端执行器在笛卡尔坐标系中的位置和姿态。

在 Matlab 中，可以使用 symbol 来获得齐次变换矩阵的形式，再带入实际的机械臂 DH 参数值，可以得到最后的齐次变换矩阵，图 8 是所用代码。

```
%代码计算
n = 6;
a = sym('a%d', [1, n]);
alpha = sym('alpha%d', [1, n]);
d = sym('d%d', [1, n]);
theta = sym('theta%d', [1, n]);
T2_temp = FKM(n, a, alpha, d, theta);

a_val = [0, -174.42, -174.42, 0, 0, 0];
alpha_val = [pi/2, 0, 0, pi/2, -pi/2, 0];
d_val = [105.03, 0, 0, 75.66, 80.09, 44.36];
offsets = [0, -pi/2, 0, -pi/2, 0, 0];
theta_val = theta0 + offsets;
all_symbols = [a, alpha, d, theta];
all_values = [a_val, alpha_val, d_val, theta_val];
T2_num = subs(T2_temp, all_symbols, all_values);
T2 = double(T2_num)

%calculate Kinematics Matrix
function T_temp = FKM(n, a, alpha, d, theta)
T_temp=eye(4);
for i=1:n
    T0=[cos(theta(i)),  -cos(alpha(i)) * sin(theta(i)),  sin(alpha(i)) * sin(theta(i)),  a(i) * cos(theta(i));
        sin(theta(i)),  cos(alpha(i)) * cos(theta(i)),  -sin(alpha(i)) * cos(theta(i)),  a(i) * sin(theta(i));
        0,                  sin(alpha(i)),                  cos(alpha(i)),                  d(i);
        0,                  0,                              0,                              1          ];
    T_temp=T_temp*T0;
end
T=T_temp;
```

图 8 正运动学计算代码

使用机器人工具箱，齐次变换矩阵可以直接通过 fkine 函数求解，末端执行器的姿态可以通过 tr2rpy 函数求解。

在 simulink 中，导入 urdf 文件，可以获得如图 9 所示的初始模型：

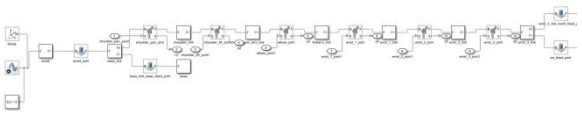


图 9 机械臂初始模型

对于六轴自由机械臂而言,需要控制六个旋转坐标系。为了实现控制，需要对其内部属性进行修改,将 Torque(力矩)设置为 Automatically Computed（自动计算），将 Motion（动力）设置为 Provide By Input（由输入提供），将传感器的信号勾选作为输出，配置好每个 Rotation 后，就可以将外界的角度信号作为输入给机械臂的每个关节。在 simulink 里，计算齐次变换矩阵使用的是 Robotics System Toolbox 里的 Get Transform，最后搭建的平台如图 10 所示。

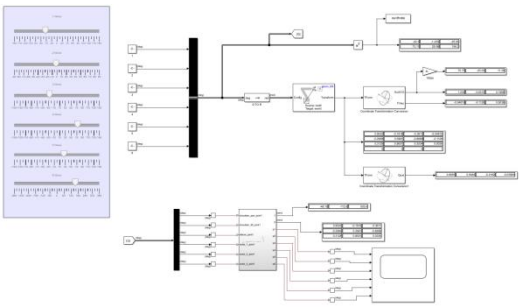


图 10 正运动学 simulink 平台

以六个关节角度均为 30° 为例，分别使用 simulink、机器人工具箱和笔者自己编写的求正运动学代码计算齐次变换矩阵，由图 11，12，13 可知，三者是一样的，在最后一列数值不同是因为机器人工作箱和代码中的单位是 mm 而 simulink 的单位 m。

0.6495	0.625	0.433	-0.2187
-0.125	0.6495	-0.75	-0.258
-0.75	0.433	0.5	0.3655
0	0	0	1

图 11 simulink 正运动学结果

```
T =
    0.6495    0.6250    0.4330    -218.7
   -0.1250    0.6495   -0.7500    -258
   -0.7500    0.4330    0.5000    365.5
         0         0         0        1
```

图 12 机器人工具箱正运动学结果

```
T2 =

    0.6495    0.6250    0.4330 -218.6626
   -0.1250    0.6495   -0.7500 -257.9696
   -0.7500    0.4330    0.5000  365.4722
         0         0         0    1.0000
```

图 13 正运动学代码结果

通过齐次变换矩阵，可以获得末端执行器的位置和姿态（其中机器人工具箱和代码的末端位置都在齐次变换矩阵的最后一列表示），如图 14，15，16 所示，其中三者计算欧拉角的顺序均为是 ZYX 顺序，但 simulink 的 x 轴和 z 轴与机器人工具箱和代码中的是相反的。

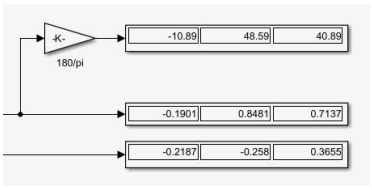


图 14 simulink 结果

```
rpy1 =

   40.8934   48.5904  -10.8934
```

图 15 机器人工具箱结果

```
rpy2 =

   40.8934   48.5904  -10.8934
```

图 16 代码结果

3 逆运动学仿真

逆运动学，是指根据给定的末端执行器位置与姿态（或直接给出齐次变换矩阵）求解机械臂各关节的角度。在实际应用中，当需要机械臂完成特定的任务，如抓取某个位置的物体时，就需要先确定

末端执行器的目标位姿，然后通过逆运动学计算出各关节所需的角度，从而控制机械臂到达目标位置。

在 Matlab 中，已知末端的齐次变换矩阵（1），

$$T_6^0 = \begin{matrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{matrix} \quad (1)$$

参考课上介绍的求逆过程，得到了各关节角度计算公式（2）。

$$\theta_1 = \operatorname{atan2}(m,n) - \operatorname{atan2}(d_4, \pm\sqrt{m^2+n^2-{d_4}^2})$$
$$\theta_5 = \pm \arccos(a_x s_1 - a_y c_1)$$
$$\theta_6 = \operatorname{atan2}(mm,nn) - \operatorname{atan2}(s_5,0)$$
$$\theta_3 = \pm \arccos\left(\frac{mmm^2 + nnn^2 - a_2^2 - a_3^2}{2a_2a_3}\right)$$
$$\theta_2 = \operatorname{atan2}(A,B) \quad (2)$$
$$\theta_4 = \operatorname{atan2}(-s_6(n_x c_1 + n_y s_1) - c_6(o_x c_1 + o_y s_1), o_z c_6 + n_z s_6) - \theta_2 - \theta_3$$

其中，

$$m = d_6 a_y - p_y, n = a_x d_6 - p_x$$
$$mm = n_x s_1 - n_y c_1, nn = o_x s_1 - o_y c_1$$
$$mmm = d_5(s_6(n_x c_1 + n_y s_1) + c_6(o_x c_1 + o_y s_1)) + p_x c_1 - d_6(a_x c_1 + a_y s_1) + p_y s_1$$
$$nnn = d_5(o_z c_6 + n_z s_6) + p_z - d_1 - a_z d_6$$
$$A = \frac{(a_3 c_3 + a_2)nnn - a_3 s_3 mmm}{a_2^2 + a_3^2 + 2a_2 a_3 c_3}, B = \frac{mmm + a_3 s_3 s_2}{a_3 c_3 + a_2}$$

根据公式（2）编写 Matlab 代码文件，即可获得逆解算法。六自由度机器人的运动学反解可能存在 8 种解，但由于结构的限制，有些解不能实现。在机器人存在多种解的情况下，应选取其中满意的一组解，以满足机器人的工作要求。

使用机器人工具箱，直接使用 ikine 函数就可以通过齐次变换矩阵求解各关节角度。

在 simulink 中，内部属性配置与正运动学相同，在平台搭建上，需要使用 Robot System Toolbox 里的 Inverse Kinematics 模块，可以直接将末端位置和姿态作为输入，输出各关节弧度，最后搭建的平台如图 17 所示。

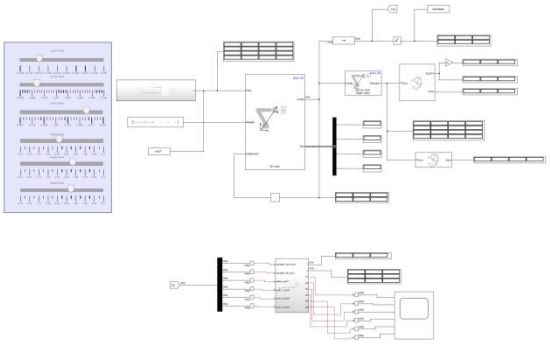
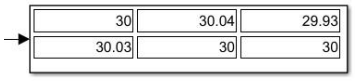


图 17 逆运动学 simulink 平台

使用在正运动学部分计算得到的末端执行器位置和姿态为例，分别使用 simulink、机器人工具箱和笔者自己编写的求正运动学代码计算各关节角度，可以得到图 18 所示各方法得到的各关节角度：



(a) simulink 结果

```
q1 =

   30.0000   30.0000   30.0000   30.0000   30.0000   30.0000
```

(b) 机器人工具箱结果

```
q2 =

   30.0000   30.0000   30.0000   30.0000   30.0000   30.0000
   30.0000   60.0000  -30.0000   60.0000   30.0000   30.0000
```

(b) 代码结果

图 18 逆运动学结果

对比可得，逆解算法求出来的八个解应该取 $30 * ones(1,6)$的那个解，也可以说明三种方法均可成为逆运动学求解方法。

因为笔者编写的 Matlab 代码程序导入 simulink 里可能导致计算时间长，但因为我们已经证明 Matlab 代码、机器人工具箱和 simulink 里的 Toolbox 的解是一样的，所以在接下来的末端姿态控制仿真中，我们直接使用 simulink 平台里自带的 Toolbox 或者机器人工具箱来节省计算时间。

4 末端轨迹控制仿真

4.1 图像处理

图像处理是末端轨迹控制仿真的关键步骤之一，其目的是将图像转化为机械臂可识别的坐标数据。在本项目中，尝试了两种图像处理方法，各自有着不同的优势和局限性。

第一种是使用 bwboundaries 函数提取边界，这个方法适用于一笔画图形或者多个一笔画图形的组合。优势是无论原图像线有多粗，因为提取了边界，所以同一个位置只会读取一个点，画出来的图比较干净，但对于更复杂的情况处理能力有限，有时会丢失特征。

第二种是使用 find 函数直接提取线条坐标，这个方法适用于更复杂的图形，但由于是直接提取坐标，所以在同一位置，可能会因为图像原线条较粗而在同一个位置读取很多个点，在画图上比第一种更凌乱，但保留了一些小的特征。

以南科大校徽，兔子和小狗的简笔画为例，分别对比两种方法，结果如图 19、20、21 所示，

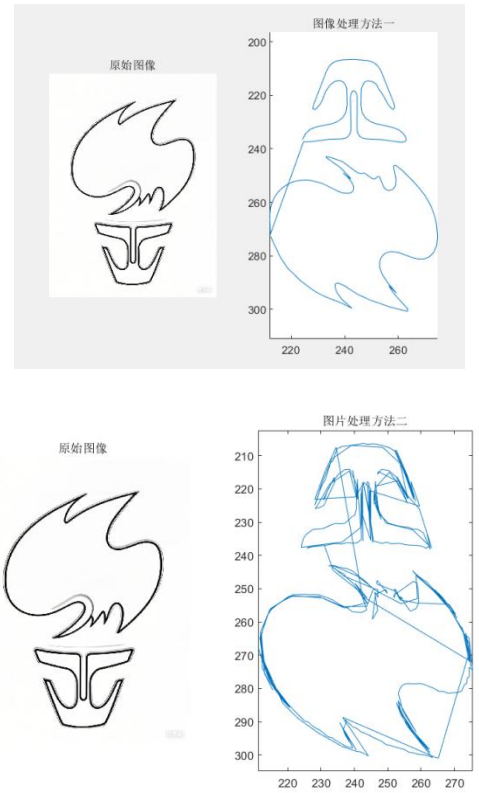


图 19 校徽的处理

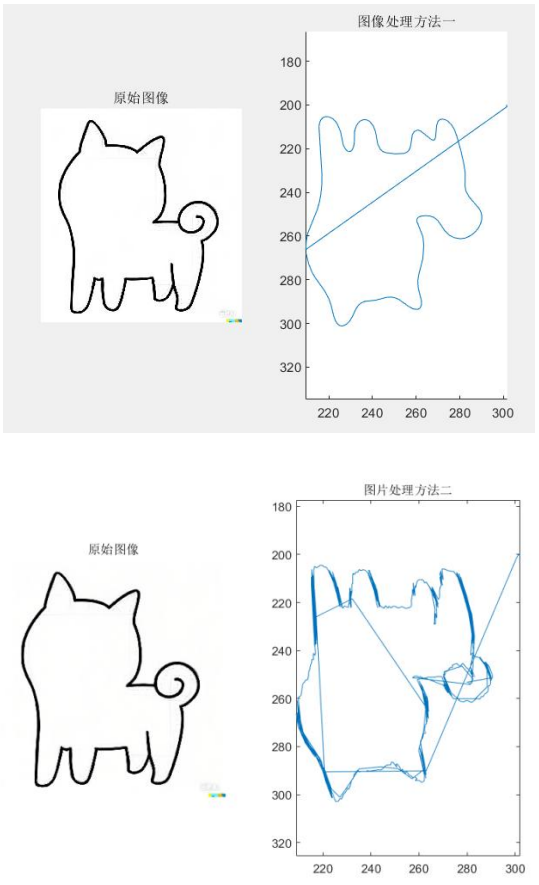
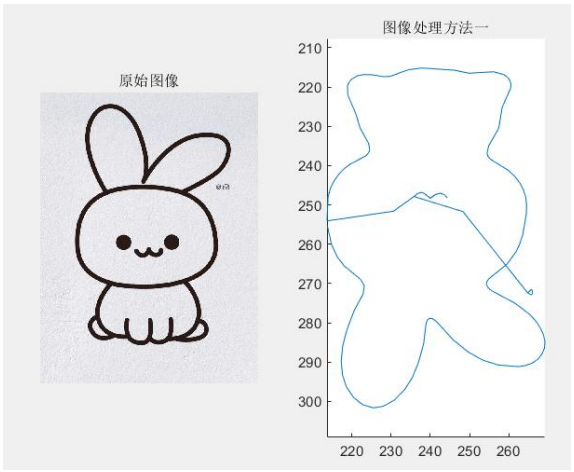


图 20 小狗的处理



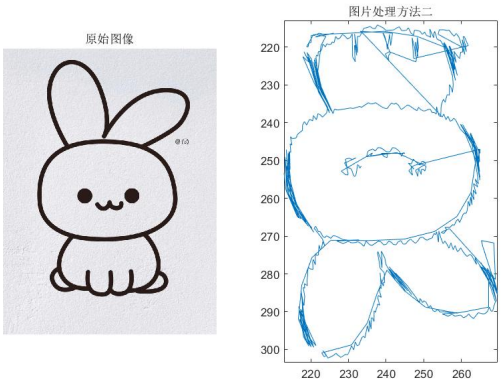


图 21 兔子的处理

可以看出，对于校徽这样由两块一笔画构成的图案，方法一的效果更好。对于兔子这样细节更多的图案，方法二的效果更好。而小狗这张图片，提取边界就会把它内卷的尾巴给忽略，所以需要根据需求选择方案。

4.2 逆运动学求解

由图片处理可以得到末端执行器的位置，假设姿态一直是竖直向下的，则可以将图像处理得到的末端执行器目标位姿作为输入，通过机器人工具箱中的逆运动学算法计算出各关节的角度值。

4.3 simulink 仿真

将逆运动学求解得到的关节角度值带入 Simulink 进行仿真，以验证末端轨迹控制的效果。机械臂内部属性配置与正运动学、逆运动学相同，在平台搭建上，使用了 Simulink 里的 Selector 模块，将从工作区（from workspace）得到的关节角度作为输入，输出末端执行器位姿，并在 XY Graph 中显示机械臂的画图过程，最后搭建得到的平台如图 22。

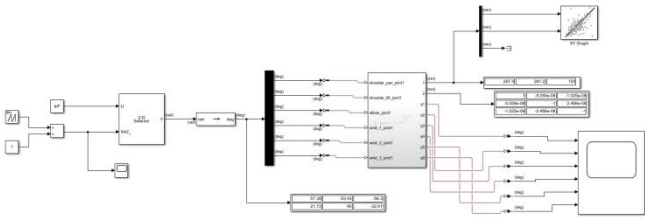


图 22 末端轨迹控制仿真 simulink 平台

以兔子为例，分别运行两个方法得到不同的输入关节角度，在 XY Graph 中可以观察到图 23 所示的仿真结果，在 Scope 中可以观察到图 24 所示的各关节角度变化。

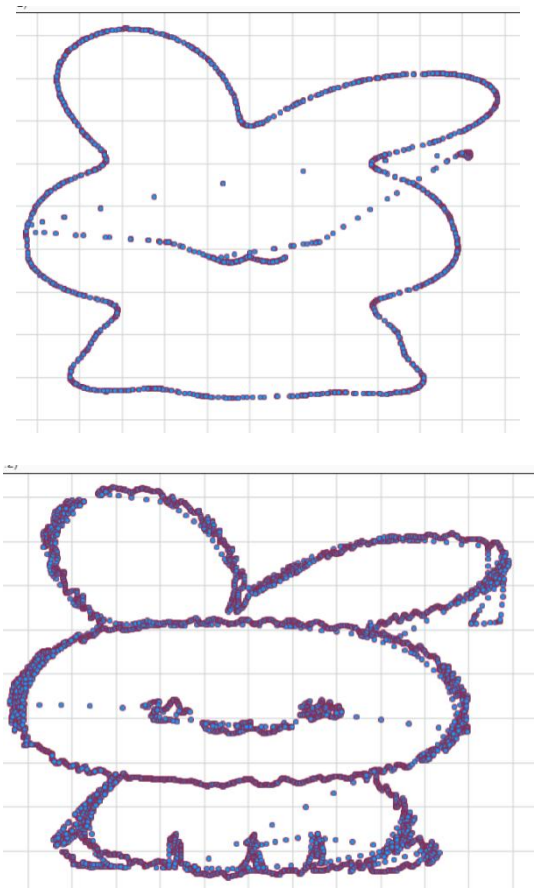
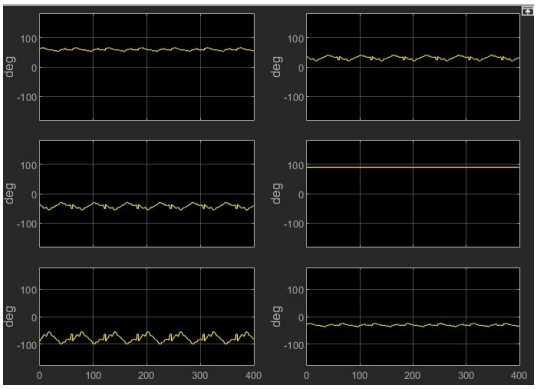


图 23 画图仿真结果



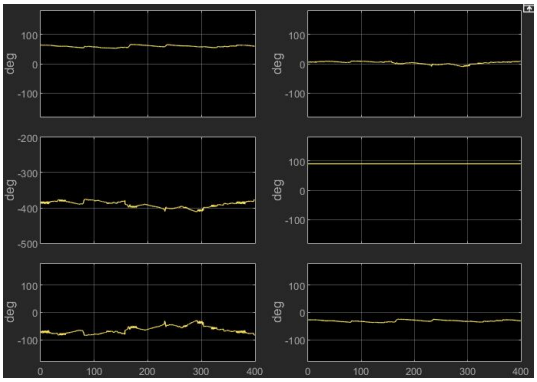


图 24 各关节角度

通过观察仿真结果，发现末端执行器的运动轨迹与图像中的目标轨迹一致，各关节角度变换平稳，说明仿真结果理想。

5 规划具多重运动学约束的到达轨迹

5.1 摘要

本部分演示了如何规划一个复杂的机器人轨迹，以引导我们的 Gluon 6L3 机械臂从桌子上拾取一个杯子。为执行此过程，我们采用了广义逆运动学，并结合了多个运动学约束。这使得机器人能够在其环境中沿着合理的轨迹和路径参考移动，而无需预先计算精确的夹爪位置。

5.2 假设

机器人在保持安全约束和平滑运动的同时，从其起始位置移动到距离杯子 2 厘米的范围内。

5.3 方法论

首先，加载 Gluon 6L3 机器人模型，并创建一个半径为 3 厘米、高 10 厘米的杯子，放置在坐标 [0.2, 0, 0.4] 处。一个名为 ‘cupFrame’ 的坐标系被放置在机器人基座上，作为机器人到达和跟随的追踪器。

在规划策略上，本实验将采用五路径点法。这些路径点包括起始位置、接近准备位置、中途接近位置、临近位置和最终位置。此外，一个广义逆运动学求解器将处理五种约束类型：笛卡尔边界、位置目标、瞄准约束、姿态目标和关节位置边界。

我们设置了两个约束系统：安全性和定位要求。安全约束确保夹爪悬停在桌面以上至少 5 厘米处，并且路径点之间的关节移动限制在 10° 以内。这可以防止任何被认为不自然的突兀动作。定位约束控制末端执行器与杯子之间的距离，容差为 5 毫米，使夹爪的 Z 轴对准杯子，并使用四元数 [0.7071, 0.7071, 0, 0] 锁定其姿态。

Matlab 代码：

```
heightAboveTable = constraintCartesianBounds(gripper);
distanceFromCup = constraintPositionTarget('cupFrame');
alignWithCup = constraintAiming('ee_link');
```

规划过程采用两阶段法。第一阶段将机器人定位到距离杯子 10 厘米的位置。此阶段的约束也更为宽松，允许从初始位置进行较大范围的移动。求解器将计算关节角度，并找到一个使用 ‘ee_link’ 坐标系使夹爪直接指向杯子的配置。

第二阶段实现动态姿态锁定。一旦达到前一个路径点的姿态，该姿态就会被锁定，并在剩余的轨迹中保持不变。此外，通过路径点 3、4 和 5，采用渐进式距离缩减，将夹爪从最初的 10 厘米距离移动到中间的 2 厘米距离，从而缓慢接近杯子。

为了绘制和动画化机器人轨迹，将使用 PCHIP（分段三次 Hermite 插值多项式）插值法连接所有路径点，以平滑曲线避免过冲或违反关节限制。该系统还以每秒 15 帧的速度生成 150 帧，以获得平滑的动画效果。

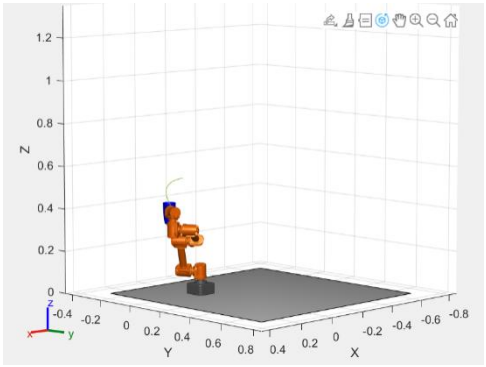


图 25

5.4 实际应用

需要对齐约束的装配操作

在杂乱环境中的拾取和放置操作

具有安全约束的人机协作

5.5 总结

本实验展示了在执行复杂操作任务时，为定制机器人系统进行基于约束的轨迹规划的真实模拟。我们为机器人设置了多重约束，使其能够以一次流畅的动作从起始位置移动到目标位置。通过定义机器人应实现的目标（即让末端执行器接近杯子），为机器人创造了一个安全的环境和流畅的运动路径。这对于拾取物体同时避开障碍物的任务非常有用。

6 规划并执行任务空间和关节空间的轨迹

6.1 摘要

本实验比较了两种基本的机器人轨迹规划方法：

任务空间控制和关节空间控制。本实验将使用 Gluon 6L3 机器人，通过不同的规划方法，模拟两种独立且截然不同的末端执行器运动。这两种轨迹各有优势，适用于特定的应用场景。

6.2 假设

任务空间和关节空间的轨迹规划都将成功地将机器人末端执行器从起始位置移动到目标位置，但其运动路径和限制会有所不同。

6.3 方法论

首先，加载机器人模型，并为模拟真实世界环境添加向下的重力加速度。通过结合平移 [0.2, 0, 0.1] 和绕 Y 轴旋转 180° 来定义目标位置。此外，轨迹时间是根据 0.1 米/秒的工具速度计算的。

任务空间 轨迹使末端执行器沿直线移动，它使用一个名为 transformtraj() 的函数在初始和最终位置之间创建路径点。任务空间运动模型禁用了位置控制增益（Kp 和 Kf 等于 0），从而实现了自然的运动动态。此外，通过常微分方程（ODE）积分，使用基于时间的任务输入来求解轨迹，这产生了基于笛卡尔坐标系的运动，非常适用于需要特定末端执行器路径的应用。

Matlab 代码：

```
[taskWaypoints,taskVelocities] = transformtraj(taskInit, taskFinal, timeInterval, trajTimes);
tsMotionModel.Kp(1:3,1:3) = 0;
```

另一方面，关节空间轨迹规划使用逆运动学来精确计算末端执行器接近最终位置时的最终配置所需的关节角度。该系统使用加权逆运动学（权重为 [0.25 0.25 0.25 1 1 1]），这将优先保证姿态而非位置。之后，将使用三次多项式插值生成平滑的关节轨迹，并辅以 B 样条平滑处理，使运动更加连续。关节空间中的 PD 控制也为关节提供了平滑的运动。

两种轨迹都被绘制和动画化，任务空间轨迹为蓝色，关节空间轨迹为红色。将两条路径重叠显示，可以实现更好的可视化效果，并有助于进行比较分析。

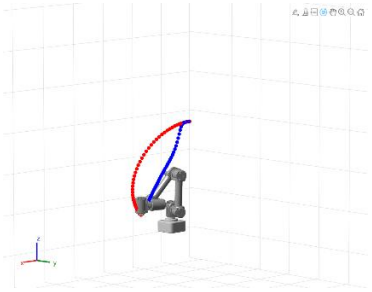


图 26

6.4 任务空间与关节空间的应用场景对比

6.4.1 任务空间控制应用

精确的线性操作：焊接、切割或喷涂等，要求末端执行器必须遵循精确的笛卡尔路径并保持一致的姿态。路径为直线。

表面跟随：要求工具与曲面保持接触或遵循复杂 3D 轮廓的装配任务。

6.4.2 关节空间控制应用

高速点对点运动：拾取和放置等只关心起点和终点位置的操作，以优化速度和能源效率为目标。

避免奇异点：在复杂的机械臂配置中，关节空间规划能比笛卡尔方法更好地避免运动学奇异点。

6.5 总结

本实验模拟了两种不同的路径轨迹：任务空间和关节空间。任务空间主要是一种基于笛卡尔坐标系的运动，适用于需要特定且直接的末端执行器路径的应用。另一方面，关节空间控制在计算上更高效，并且关节运动平滑，适用于高速定位任务。图示揭示了这两种轨迹规划方法如何实现相同的目标，并允许机器人工程师根据应用需求选择最优的方法。

7. 合作分工与组员贡献情况

项目围绕六自由度 UR 机器人展开，组内六人分工明确，紧密合作。刘丰源负责项目的整体统筹与进度协调，主导了机器 urdf 模型的构建与物理参数优化，并完成了初步仿真验证。付宛昕负责运动学算法的开发，完成了正运动学与逆运动学的解析解计算、编程与验证，并在此基础上对末端轨迹进行控制仿真。Adelbert Reinhard Riang（梁沛桦）负责进行轨迹规划与执行，完成了具多重运动学约束的到达轨道的规划和任务空间和关节空间轨迹的规划与执行。梁家源负责前期的资料检索，并协助刘丰源制作 urdf 模型和初始仿真验证，验证模型物理

可靠性。王天源负责项目后期的结果处理，对项目进行分析、概括和总结，并尝试进行了操纵形态跟踪。殷文韬负责整合技术文档与代码注释，主导 PPT 设计与报告撰写，并与梁沛桦进行高效沟通与合作，协助进行规划轨迹仿真。

8.总结

项目旨在借助虚拟仿真系统性评估六轴机械臂的建模与控制性能，在零风险条件下降低实际调试成本，并满足《机器人建模与控制》课程的实践深度。团队先以零件装配与惯量标定构建 URDF 模型，使用 urdf_check 完成语法与拓扑校验及初步运动验证；再在 Matlab 环境中结合 Robotics System Toolbox、Simulink 和 Simscape 实现正、逆运动学与多体动力学仿真，各途径结果一致，模型可靠。功能演示涵盖两大典型任务：①通过边界提取与像素坐标法将校徽、兔子等图案转换为末端轨迹，实现“机械臂绘图”；②构建位置、姿态与安全高度等多约束，借广义逆运动学规划五个路径点生成安全抓取轨迹，并对任务空间与关节空间控制进行对比。仿真显示，多项式插值平滑、样条曲线灵活、梯形速度效率高；任务空间轨迹精度更佳，而关节空间轨迹实时性和运动平顺性占优。

面对关节多解、奇异与不可达位姿、参数配置和工具箱安装等难题，团队通过加权逆解、限幅与可达性检测、轨迹预处理及远程安装服务等手段化解。最终，仿真验证了机械臂在复杂环境下的作业能力，为工业自动化柔性升级及农业、服务、教育等跨领域应用奠定基础，并培养了成员的系统集成与团队协作能力 。

参 考 文 献

[1] https://www.mathworks.com/help//releases/R2021a/robotics/index.html?s_tid=CRUX_lftnav

[2] https://www.mathworks.com/help//releases/R2021a/simulink/index.html

[3] https://www.mathworks.com/help//releases/R2021a/simulink/simulation.html

[4] https://ww2.mathworks.cn/help/robotics/ug/plan-a-reaching-trajectory-with-kinematic-constraints.html

[5] https://www.mathworks.com/help//releases/R2021a/robotics/ug/plan-and-execute-trajectory-kinova-gen3.html

[6] 利用 Robotics Toolbox 进行机械臂运动学分析（五）：逆运动学求解_机械臂逆运动学求解-CSDN 博客

[7] 六自由度机械臂运动学分析及其轨迹规划_rb08 的 6 自由度机械臂运动学方程-CSDN 博客

[8] 基于 MATLAB 的焊接机器人轨迹规划与仿真_基于 matlab 的焊接机器人轨迹规划与仿真方法-CSDN 博客

[9] 利用 Robotics Toolbox 进行机械臂运动学分析（五）：逆运动学求解_机械臂逆运动学求解-CSDN 博客

[10] P.I. Corke, Robotics, Vision \& Control: Fundamental Algorithms in MATLAB. Second edition. Springer, 2017. ISBN 978-3-319-54413-7.