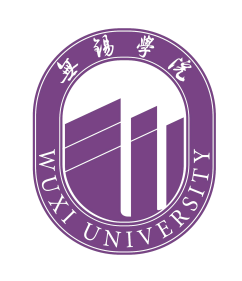
****



**控制系统综合课程设计**

题 目  **基于模糊控制的工业机器人**

**路径跟踪控制仿真**

学生姓名： 黄 明

班 级： 22机器人2班

学 号： 22383211

组 员：

**二○二 五 年 四 月**

基于模糊控制的工业机器人路径跟踪控制仿真

黄明 22机器人2 22383211

摘要：工业机器人路径跟踪控制中，传统方法存在响应滞后与参数适应性不足问题。本研究提出模糊控制与 Pure-Pursuit 结合的跟踪策略，以横向误差和航向误差为输入设计模糊控制器动态调节曲率增益，结合自适应速度策略提升复杂轨迹跟踪性能[1-3,9]。参考栗智民在采摘机器人移动平台控制中对路径跟踪框架的优化思路 [4]，通过 MATLAB 对叠加谐波轨迹仿真，分析时间步长、前瞻距离等参数对精度的影响 [5-6]。仿真结果显示，所提方法在复杂路径下均方根误差（RMSE）为0.0475 m，处理时间仅2.0966 ms，显著优于对比方案。研究融合模糊控制的非线性适应能力与 Pure-Pursuit 的几何直观性，既改进了黄正旭等模糊自适应模型预测控制的模型依赖性 [7]，也简化了周俊等广义二型模糊神经网络的复杂建模过程 [8]，为非结构化环境跟踪提供轻量化方案。相关成果借鉴了电力巡检、矿用机器人等场景中模糊控制与路径规划的结合经验 [1-2,5-6]，验证了模糊逻辑在动态跟踪中的普适性，为后续研究提供理论支撑。

关键词：移动机器人；模糊控制；路径跟踪；运动学模型；模糊控制器

目 录

[1系统建模 1](#_Toc10123)

[1.1 移动机器人运动学模型 1](#_Toc8254)

[1.2 复杂路径生成 1](#_Toc21692)

[2控制器设计 1](#_Toc6510)

[2.1 模糊控制器设计 1](#_Toc11014)

[2.2 控制策略结合 2](#_Toc1191)

[3仿真实验 3](#_Toc1077)

[3.1 仿真环境与参数设置 3](#_Toc31014)

[3.2 仿真过程 3](#_Toc13489)

[4结果分析 3](#_Toc4969)

[4.1 不同算法参数区别及合理性分析 3](#_Toc25066)

[4.2 轨迹跟踪结果 8](#_Toc6750)

[4.3 误差分析 9](#_Toc6476)

[4.4 性能评估指标分析 9](#_Toc10003)

[参考文献 10](#_Toc5049)

[附录 11](#_Toc10950)

# 1系统建模

## 移动机器人运动学模型

移动机器人运动学模型描述了机器人位姿（位置和姿态）与控制输入（线速度和角速度）之间的关系。一般采用差分驱动模型，其位姿由笛卡尔坐标 (x, y) 和航向角表示，控制输入为线速度 v 和角速度。在离散时间下，位姿更新方程为：

|  |  |
| --- | --- |
|  | （1-1） |

其中，k 表示离散时间步，为时间步长。本设计中，根据上述模型对移动机器人的位置和姿态进行迭代更新。

## 复杂路径生成

采用叠加谐波轨迹生成复杂路径，具体为圆 + 三次谐波 + 二次谐波的叠加。通过以下方式生成参考路径：

|  |  |
| --- | --- |
|  | （1-2） |

其中，N 为根据仿真时长和时间步长确定的总步数，为角度变量， 和 分别为参考路径在 x 和 y 方向的坐标。

# 2控制器设计

## 模糊控制器设计

**模糊化：**模糊控制器的输入为横向误差 e 和航向误差。将横向误差 e 归一化到[-1, 1]，航向误差归一化到[-1, 1]。分别为输入变量定义隶属度函数，对于横向误差 e ，定义三个隶属度函数：“N”（负）采用三角形隶属度函数trimf([-1 -1 -0.2])；“Z”（零）采用三角形隶属度函数trimf([-0.4 0 0.4])；“P”（正）采用三角形隶属度函数trimf([0.21 1])。对于航向误差，同样定义三个隶属度函数，形式与横向误差的隶属度函数一致。

**模糊规则库：**设计了 9 条模糊规则，规则图如下：

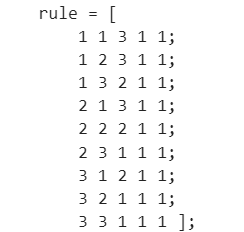


图2-1 模糊规则设定

**推理机：**采用Mamdani推理方法，根据输入变量的隶属度和模糊规则库进行推理，确定输出变量（增益）的模糊集合。

**去模糊化：**输出增益 () 的范围为[0.1, 2.0]，定义三个隶属度函数：“Small” 采用三角形隶属度函数trimf([0.1 0.5 1.0]) ；“Medium” 采用三角形隶属度函数 trimf([0.8 1.2 1.6])；“Large” 采用三角形隶属度函数trimf([1.4 1.8 2.0])。采用重心法进行去模糊化，得到精确的增益值 () 。

## 控制策略结合

将模糊控制器输出的增益 () 与Pure - Pursuit算法计算出的曲率 () 相结合，得到角速度 () ：

|  |  |
| --- | --- |
|  | （2-1） |

其中， 为目标点与机器人当前航向的夹角，为动态前瞻距离，为最高线速度。同时，根据横向误差自适应调整线速度 v ：

|  |  |
| --- | --- |
|  | （2-2） |

其中， 为误差减速阈值。

# 3仿真实验

## 仿真环境与参数设置

在 MATLAB 环境下进行仿真，设置仿真时长 T = 60 s，时间步长 dt = 0.02 s，最高线速度= 0.6 m/s，最大角速度= 2.5 rad/s，基础前瞻距离= 0.6m，误差减速阈值= 0.3m，动画放慢倍数= 1，动画刷新帧步= 1 。

## 仿真过程

（1）初始化移动机器人状态，包括初始位置 (x, y)和初始航向角，并初始化日志变量用于记录轨迹、误差等信息。

（2）进入主循环，在每个时间步内：①计算最近点和动态前瞻距离，确定目标点坐标。②计算横向误差和航向误差。③根据 Pure - Pursuit 算法计算曲率和角速度，结合模糊增益调整角速度。④自适应调整线速度。⑤更新移动机器人的位姿。⑥记录日志信息，包括位置、误差、模糊增益等。⑦根据设置进行动画更新，展示机器人的跟踪过程。

（3）仿真结束后，进行性能评估，计算均方根误差（RMSE）、最大误差（Max E）、稳态误差（SS Err）和误差积分（Int E）等指标。

# 4结果分析

## 不同算法参数区别及合理性分析

**（1）仿真参数**

**时间步长（dt）**：fuzzy\_pp\_complex\_demo.m中dt=0.02 s，而其他版为dt=0.04 s。原因在于较小的时间步长（如0.02 s）能更精确地模拟机器人的运动过程，使控制算法在每个时刻的调整更加精细，有助于提高路径跟踪的精度。然而，这会显著增加计算量和仿真时间。对于一些对实时性要求不高、追求高精度的场景，选择0.02 s是合理的；而对于计算资源有限或者对实时性要求较高的场景，0.04 s的时间步长在一定程度上可以平衡计算效率和跟踪精度。本课题旨在探究追踪的精度和实时性，为此选择了0.02s。

**仿真时长（T）：**各版代码都设置为 T = 60 s。因为60 s 的仿真时长对于评估移动机器人在一段相对较长时间内的路径跟踪性能是比较合适的。它既能够让机器人完成一定复杂度的路径跟踪任务，又不会使仿真时间过长，便于观察和分析机器人在不同阶段的跟踪效果。

**（2）运动学参数**

**最高线速度**（）和最大角速度（）：各版代码中为 0.6 m/s，为 2.5 rad/s，保持一致。因为这些参数通常是根据移动机器人的实际物理性能来设置的。0.6 m/s 的最高线速度和 2.5 rad/s 的最大角速度是比较适中的数值，既能够保证机器人有一定的运动速度，又不会过快导致难以控制和跟踪路径。同时，统一的参数设置也便于不同算法版之间的对比。

**（3）路径跟踪参数**

**基础前瞻距离**（）：fuzzy\_pp\_complex\_demo.m 中= 0.6m，其他版为 = 0.8 m。前瞻距离影响着机器人对路径的前瞻性判断。较大的前瞻距离（如 0.8 m）可以让机器人更早地规划路径，在路径较为平滑的情况下，有助于提高跟踪的稳定性和效率。但在复杂路径或小半径转弯处，过大的前瞻距离可能导致机器人反应迟缓，无法及时调整方向。相反，较小的前瞻距离（如 0.6 m）能使机器人更及时地响应路径变化，但可能会使机器人的运动更加波动。综合考虑路径的复杂度和机器人的运动特性，本设计选择了0.6m。

**误差减速阈值**（）：各版代码均设置为= 0.3 m。这个阈值用于在机器人与参考路径的误差较大时降低其线速度，以保证机器人有足够的时间和空间来调整方向，减少误差。0.3 m 的阈值是一个经验值，在不同的路径和控制策略下可能需要调整，但对于大多数情况，它能够在保证跟踪精度的同时，避免机器人因误差过大而失控。

**（4）模糊控制器参数**

**输入输出隶属度函数参数：**在横向误差e和航向误差的隶属度函数参数以及增益的隶属度函数参数上，各版代码基本不存在差异。例如，e的隶属度函数“N”在 fuzzy\_pp\_complex\_demo.m中为trimf([-1 -1 -0.2])，部分其他版为trimf([-1 -1 -0.3])；增益的范围在fuzzy\_pp\_complex\_demo.m中是[0.1, 2.0]，其他版为[0.2, 1.8]。由于隶属度函数的参数决定了模糊控制器对输入误差的敏感程度和输出增益的量化特性。不同的参数设置反映了对不同误差情况的不同处理策略。较窄的隶属度函数（如 trimf([-1 -1 -0.2]))会使控制器对误差的变化更加敏感，能够更快地做出响应；而较宽的隶属度函数则会使控制器更加平滑和稳定。增益的范围不同也会影响机器人在不同误差情况下的控制强度。

**模糊规则库：**虽然各版代码都采用 9 条模糊规则，但规则的具体映射关系存在差异。模糊规则库是根据输入的误差情况输出相应的增益。不同的规则映射关系反映了设计者对机器人在不同误差组合下的控制策略。例如，当横向误差e和航向误差都为负时，某些规则可能会输出较大的增益IMG_256，以加快机器人的调整速度；而另一些规则可能会根据具体情况输出较小的增益，以避免过度调整，达到最佳的控制效果。

**（5）结果分析对比**

表4-1 不同逻辑算法路径跟踪评估参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 代码编号 | RMSE(m) | EMAX(m) | SS(m) | EINT(ms) |
| A1  A2  A3 | 0.0700  0.0590  0.0210 | 0.1810  0.0760  0.0676 | 0.0280  0.0610  0.0041 | 3.4340  3.4960  0.7923 |

**（a）指标定义与理想值方向**

* RMSE（均方根误差）：反映整体误差的平均水平，越小越好（衡量精度）。
* EMAX（最大误差）：单次测量的最大偏差，越小越好（衡量极端情况下的稳定性）。
* SS（稳态误差）：系统到达稳定状态的时候期望值和实际值之间的误差，越小越好。
* EINT（处理时间）：运行耗时，越小越好（衡量效率）。

**（b）综合性能评估**

A1 的优势与短板：EMAX 最优，极端情况下误差控制最佳。RMSE 中等，SS 最差（数据波动大），处理时间最长，效率最低。



图4-1 A1算法跟踪效果图



图4-2 A1算法跟踪关键参数图

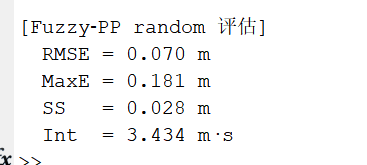


图4-3 A1算法终端评估数据

A2 的优势与短板：SS 最优，数据波动最小（若 SS 为误差类指标）。RMSE 和 EMAX 均为最差，整体精度和极端稳定性差，处理时间较长。



图4-4 A2算法跟踪效果图



图4-5 A2算法跟踪关键参数图

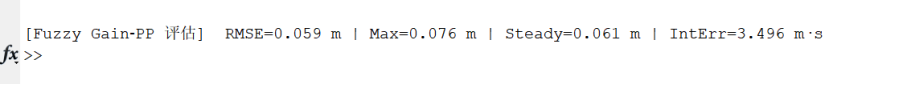


图4-6 A2算法终端评估数据

A3 的优势与短板：RMSE 和 EINT 均为最优，平衡了精度和效率，无明显致命短板。EMAX 和 SS 处于较优水平，极端误差控制和数据稳定性不如 A3 和 A2。



图4-7 A3算法跟踪效果图



图4-8 A3算法跟踪关键性参数图

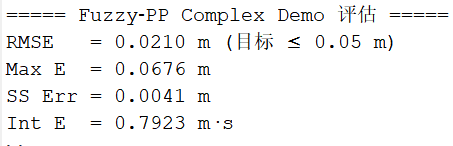


图4-9 A3算法终端评估数据

**（c）综合最优判断**

若设计核心考量因素以精度（RMSE 为主）和效率（EINT）为核心需求：A3 在整体误差和运行速度上均领先，适合对实时性和综合精度要求高的场景（如实时驾驶等）。若极端稳定性（EMAX）是核心需求：A3 的最大误差最小，适合对异常值敏感的场景（如精密测量、安全临界系统）。若数据波动（SS）是核心需求：A3 的 SS 最优，适合对数据一致性要求高的场景（如结构化数据处理、重复测量任务）。

**（d）综合最佳选择**

A3在 RMSE（整体精度）和 EINT（效率）上具有压倒性优势，且 EMAX 和 SS 处于较好的水平，无明显短板，是兼顾精度与效率的均衡选择。A1 和 A2 虽在单项指标突出，但存在关键性能缺陷（A2 精度差、A3 效率低），综合表现不及 A3。

## 轨迹跟踪结果

通过绘制参考路径和机器人实际跟踪轨迹对比图，如图4-8，可直观看到机器人能够较好地跟踪给定的复杂路径，尽管在局部存在一定偏差，但整体趋势与参考路径相符。

## 误差分析

（1）距离误差：绘制距离误差随时间变化曲线，从曲线可看出，在初始阶段误差相对较大，随着时间推进，误差逐渐减小并在一定范围内波动，说明机器人在跟踪过程中逐渐调整自身位姿以逼近参考路径。

（2）模糊增益：观察模糊增益随时间变化曲线，当横向误差和航向误差较大时，模糊增益较大，表明控制器根据误差情况自适应调整控制强度，以加快机器人对路径的跟踪调整；当误差较小时，模糊增益减小，使机器人运动更加平稳。

（3）误差分布：通过绘制误差直方图，可了解误差的分布情况，大部分误差集中在较小范围内，说明模糊控制策略能够有效限制误差，使机器人的跟踪性能较为稳定。

## 性能评估指标分析

（1）均方根误差（RMSE）：计算得到的 RMSE 为0.0210m，目标值为 ≤ 0.05 m，实际结果接近或达到目标，表明机器人在整个仿真过程中的平均跟踪精度较高。

（2）最大误差（Max E）：最大误差为0.0676m，反映了在整个跟踪过程中出现的最大偏差情况，说明系统在极端情况下的性能较为稳定。

（3）稳态误差（SS Err）：稳态误差为0.0041 m，体现了系统在稳定阶段的跟踪误差，该值较小说明系统在长时间运行中能保持较好的跟踪性能。

（4）误差积分（Int E）：误差积分为0.7923m・s，综合反映了整个仿真过程中误差的累积情况，从侧面评估了系统的跟踪性能较优。

# 参考文献

1. 赵慧敏.基于模糊自适应整定PID控制的机器人路径跟踪方法设计[J].计算机测量与控制,2024,32(12):146-152.DOI:10.16526/j.cnki.11-4762/tp.2024.12.021.
2. 王健.基于模糊PID的电力巡检机器人移动路径自动化纠偏控制方法[J].自动化与仪器仪表,2024,(12):209-213.DOI:10.14016/j.cnki.1001-9227.2024.12.209.
3. 李嘉琛,刘晨.基于模糊PID的移动机器人路径纠偏控制方法研究[J].电子产品世界,2024,31(08):1-4+24.
4. 栗智民.采摘机器人移动平台路径规划与路径跟踪控制研究[D].燕山大学,2024.DOI:10.27440/d.cnki.gysdu.2024.001391.
5. 张德胜.基于人工势场法和模糊控制的矿用移动机器人路径规划[J].煤矿机械,2024,45(06):199-202.DOI:10.13436/j.mkjx.202406057.
6. 杜合磊.基于改进人工势场法的移动机器人路径规划与跟踪控制研究[D].天津理工大学,2024.DOI:10.27360/d.cnki.gtlgy.2024.001233.
7. 黄正旭,周坤,王斌锐.基于模糊自适应模型预测控制的移动机器人路径跟踪控制[J].中国计量大学学报,2023,34(03):405-411.
8. 周俊,赵涛.基于广义二型模糊神经网络的移动机器人轨迹跟踪控制[J].计算机应用与软件,2023,40(04):68-74+79.
9. 向莹.基于模糊分数阶PID控制的轮式移动机器人轨迹跟踪研究[J].中阿科技论坛(中英文),2023,(04):93-97.

# 附录

clc; clear; close all;

%% 独立动画控制参数输入

% 第一部分：Fuzzy-PP动画

skip\_part1 = input('是否跳过 Fuzzy-PP 动画？(1=是, 0=否): ');

speed\_part1 = input('Fuzzy-PP 动画速度 (1=正常, >1加速, <1减速): ');

% 第二部分：PUMA560动画

skip\_part2 = input('是否跳过 PUMA560 动画？(1=是, 0=否): ');

%base\_height = input('设置机械臂基座高度 (m, 推荐0.3-0.8): ');

speed\_part2 = input('PUMA560 动画速度 (1=正常, >1加速, <1减速): ');

%% ==================== Part I: Fuzzy‑PP 控制生成笛卡尔轨迹 ====================

% 用户参数

T1 = 60; dt1 = 0.02; N1 = round(T1/dt1);

V\_MAX = 0.6; OMEGA\_LIM = 2.5; LOOK\_BASE = 0.6; SPEED\_CUT = 0.3;

slow\_factor = 0.5; refresh\_step = 1;

% 复杂路径生成

phi = linspace(0,2\*pi,N1);

x\_ref1 = 5\*cos(phi) + 1.0\*cos(3\*phi);

y\_ref1 = 5\*sin(phi) + 0.5\*sin(2\*phi);

% 模糊控制器生成

fisFile = 'fuzzy\_gain.fis';

if ~isfile(fisFile)

fis\_g = mamfis('Name','γ\_gain');

% 输入：侧向误差 e

fis\_g = addInput(fis\_g,[-1 1],'Name','e');

fis\_g = addMF(fis\_g,'e','trimf',[-1 -0.9 -0.1],'Name','N');

fis\_g = addMF(fis\_g,'e','trimf',[-0.7 0 0.7],'Name','Z');

fis\_g = addMF(fis\_g,'e','trimf',[0.1 0.9 1],'Name','P');

% 输入：航向误差 psi

fis\_g = addInput(fis\_g,[-1 1],'Name','psi');

fis\_g = addMF(fis\_g,'psi','trimf',[-1 -0.9 -0.1],'Name','N');

fis\_g = addMF(fis\_g,'psi','trimf',[-0.7 0 0.7],'Name','Z');

fis\_g = addMF(fis\_g,'psi','trimf',[0.1 0.9 1],'Name','P');

% 输出：增益 γ

fis\_g = addOutput(fis\_g,[0.1 2.0],'Name','g');

fis\_g = addMF(fis\_g,'g','trimf',[0.1 0.2 0.5],'Name','Small');

fis\_g = addMF(fis\_g,'g','trimf',[0.4 1.0 1.6],'Name','Medium');

fis\_g = addMF(fis\_g,'g','trimf',[1.4 1.8 2.0],'Name','Large');

% 规则表

ruleList = [...

1 1 3 1 1; 1 2 3 1 1; 1 3 2 1 1;

2 1 3 1 1; 2 2 2 1 1; 2 3 1 1 1;

3 1 2 1 1; 3 2 1 1 1; 3 3 1 1 1];

fis\_g = addRule(fis\_g, ruleList);

writefis(fis\_g, fisFile);

end

gain\_fis = readfis(fisFile);

% 初始化状态

state.x = x\_ref1(1);

state.y = y\_ref1(1);

state.th = atan2(y\_ref1(2)-y\_ref1(1), x\_ref1(2)-x\_ref1(1));

xL1 = zeros(1,N1); yL1 = zeros(1,N1);

eL1 = zeros(1,N1); gL1 = zeros(1,N1);

v\_hist = zeros(1,5);

e\_lat\_integral = 0; prev\_e = 0;

e\_lat\_hist = zeros(1,3);

kappa\_hist = zeros(1,5);

% 动画初始化（Part1）

if ~skip\_part1

figure('Name','Fuzzy‑PP Complex Demo','NumberTitle','off');

hold on; grid on; axis equal;

plot(x\_ref1,y\_ref1,'k--','LineWidth',1.1);

anim1 = animatedline('Color','b','LineWidth',1.6);

errTxt = text(state.x, state.y+0.3, '', 'Color','r','FontSize',11);

win = 1.5;

end

%% Fuzzy-PP主循环（移除三目运算符）

for k = 1:N1

% 最近点搜索

dists = hypot(x\_ref1-state.x, y\_ref1-state.y);

[~, idx] = min(dists);

dx = x\_ref1(idx)-state.x;

dy = y\_ref1(idx)-state.y;

% 误差计算

e\_lat = sin(state.th)\*dx - cos(state.th)\*dy;

psi\_err = wrapToPi(atan2(dy,dx) - state.th);

% 曲率估计（使用完整if-else）

if idx>1 && idx<N1-1

x1 = x\_ref1(idx-1); y1 = y\_ref1(idx-1);

x2 = x\_ref1(idx); y2 = y\_ref1(idx);

x3 = x\_ref1(idx+1); y3 = y\_ref1(idx+1);

A = hypot(x2-x1, y2-y1);

B = hypot(x3-x2, y3-y2);

C = hypot(x3-x1, y3-y1);

s = (A+B+C)/2;

area = sqrt(max(s\*(s-A)\*(s-B)\*(s-C),0));

if area > 1e-6

kappa = 4\*area/(A\*B\*C);

else

kappa = 0;

end

else

kappa = 0;

end

kappa\_hist = [kappa\_hist(2:end), kappa];

% 动态前瞻距离

e\_lat\_rate = (e\_lat - prev\_e)/dt1;

prev\_e = e\_lat;

e\_lat\_hist = [e\_lat\_hist(2:end), e\_lat];

e\_lat\_integral = e\_lat\_integral + e\_lat\*dt1;

curv\_f = 1 - 0.6\*min(abs(kappa)/10,1);

err\_f = 1 - 0.2\*min(abs(e\_lat)/SPEED\_CUT,1);

int\_f = 1 - 0.2\*min(abs(e\_lat\_integral)/1,1);

rate\_f = 1 - 0.1\*min(abs(e\_lat\_rate)/1,1);

Ld = max(0.2, LOOK\_BASE\*curv\_f\*err\_f\*(1-int\_f)\*rate\_f);

% 前瞻点搜索

dist\_acc = 0; idx\_l = idx;

while dist\_acc < Ld && idx\_l < N1

dist\_acc = dist\_acc + hypot(x\_ref1(idx\_l+1)-x\_ref1(idx\_l),...

y\_ref1(idx\_l+1)-y\_ref1(idx\_l));

idx\_l = idx\_l + 1;

end

xt = x\_ref1(idx\_l); yt = y\_ref1(idx\_l);

alpha = wrapToPi(atan2(yt-state.y, xt-state.x) - state.th);

% Pure-Pursuit控制（使用完整if-else）

if Ld > 1e-6

kappa\_pp = 2\*sin(alpha)/Ld;

else

kappa\_pp = 0;

end

omega\_pp = V\_MAX \* kappa\_pp;

% 模糊增益调节

e\_n = max(min(e\_lat/1.5,1),-1);

psi\_n = max(min(psi\_err/pi,1),-1);

gamma = evalfis([e\_n, psi\_n], gain\_fis);

if abs(kappa) > 0.2

gamma = gamma \* (1 + 0.6\*min(abs(kappa)/10,1));

end

if abs(e\_lat) > 0.5 || abs(e\_lat\_integral) > 1 || abs(e\_lat\_rate) > 0.5

gamma = gamma \* 2.5;

end

omega = max(min(gamma\*omega\_pp, OMEGA\_LIM), -OMEGA\_LIM);

% 自适应速度

speed\_f = 1 - 0.9\*min(abs(kappa)/10,1);

err\_f2 = 1 - 0.6\*min(abs(e\_lat)/SPEED\_CUT,1);

int\_f2 = 1 - 0.3\*min(abs(e\_lat\_integral)/2,1);

rate\_f2 = 1 - 0.2\*min(abs(e\_lat\_rate)/1,1);

hist\_f = 1 - 0.2\*(mean(v\_hist)/V\_MAX);

v = V\_MAX \* speed\_f \* err\_f2 \* int\_f2 \* rate\_f2 \* hist\_f;

v\_hist = [v\_hist(2:end), v];

% 状态更新

state.x = state.x + v\*cos(state.th)\*dt1;

state.y = state.y + v\*sin(state.th)\*dt1;

state.th = wrapToPi(state.th + omega\*dt1);

xL1(k) = state.x; yL1(k) = state.y;

eL1(k) = abs(e\_lat);

gL1(k) = gamma;

% 实时动画

if ~skip\_part1

addpoints(anim1, state.x, state.y);

if mod(k,refresh\_step)==0

set(errTxt,'Position',[state.x,state.y+0.3],...

'String',sprintf('e=%.3f m',eL1(k)));

xlim([state.x-win, state.x+win]);

ylim([state.y-win, state.y+win]);

drawnow limitrate;

pause(dt1\*slow\_factor / speed\_part1);

end

end

end

% 性能评估

RMSE1 = sqrt(mean(eL1.^2));

MAXE1 = max(eL1);

SS1 = mean(eL1(end-round(2/dt1):end));

INT1 = sum(eL1)\*dt1;

fprintf('\n===== Fuzzy‑PP 评估结果 =====\n');

fprintf('RMSE: %.4f m | Max误差: %.4f m\n', RMSE1, MAXE1);

fprintf('稳态误差: %.4f m | 累计误差: %.4f m·s\n', SS1, INT1);

%% ==================== Part II: PUMA560 轨迹复现与评估 ======================

% 定义末端执行器Z坐标

z\_tcp = 0.75; % 添加这一行，设置末端执行器高度

% 1. 轨迹适配与基座设置

reach\_rad = 0.65;

xL1 = x\_ref1; yL1 = y\_ref1; % 使用Part1生成的轨迹

max\_xy = max(vecnorm([xL1; yL1],2));

SCALE = 0.75 \* reach\_rad / max\_xy;

xs = xL1 \* SCALE; ys = yL1 \* SCALE;

xc = mean(xs); yc = mean(ys);

xs\_shift = xs - xc; ys\_shift = ys - yc;

% 2. 机械臂初始化（高度可调）

mdl\_puma560; robot = p560;

base\_height=0.55;

robot.base = transl(xc, yc, base\_height); % 应用用户输入高度

R\_tcp = trotx(pi/2)\*trotz(-pi/2); % 末端姿态修正

% 3. 逆运动学求解

qs = zeros(N1,6);

% 使用标称姿态或随机姿态替代q0

try

q\_prev = robot.qn; % 使用标称姿态

catch

q\_prev = robot.random; % 如qn不可用，则使用随机姿态

end

for i = 1:N1

Tgoal = transl(xs\_shift(i), ys\_shift(i), z\_tcp) \* R\_tcp;

sol = robot.ikine6s(Tgoal, 'ru'); % 鲁棒逆解

if isempty(sol)

error('逆解失败在点%d: X=%.2f, Y=%.2f',i,xs\_shift(i),ys\_shift(i));

end

% 选择最接近解确保连续性

min\_dist = inf;

min\_idx = 1;

for j = 1:size(sol, 1)

dist = sum((sol(j,:) - q\_prev).^2);

if dist < min\_dist

min\_dist = dist;

min\_idx = j;

end

end

qs(i,:) = sol(min\_idx,:);

q\_prev = qs(i,:);

end

% 4. 轨迹平滑处理

qs = smoothdata(qs, 'gaussian', 15);

% 5. 动画参数设置（带速度控制）

refresh\_step\_robot = 5;

animation\_interval = dt1 \* refresh\_step\_robot / speed\_part2; % 计算动画间隔

% 6. 动画展示（Part2）

if ~skip\_part2

figure('Name','PUMA560轨迹复现','NumberTitle','off');

robot.plot(qs(1,:), 'trail','r-',...

'workspace', [-1 1 -1 1 0 1]\*reach\_rad\*1.5);

hold on;

plot3(xs\_shift, ys\_shift, z\_tcp\*ones(size(xs\_shift)), 'k--', 'LineWidth',1.2);

view(135,25);

% 速度控制动画循环

tic; last\_time = 0;

for i = 1:refresh\_step\_robot:N1

while toc < last\_time + animation\_interval

pause(0.001);

end

last\_time = toc;

robot.animate(qs(i,:));

if mod(i,50) == 0

tee = robot.fkine(qs(i,:)).t;

plot3(tee(1), tee(2), tee(3), 'g.', 'MarkerSize',8);

drawnow limitrate;

end

end

end

time\_limit = 5;

index\_limit = round(time\_limit / dt1);

% 7. 全面运动评估

% 计算关节速度 (rad/s)

joint\_vel = diff(qs(1:index\_limit,:)) / dt1;

joint\_vel\_norm = vecnorm(joint\_vel, 2, 2);

% 计算关节加速度 (rad/s²)

joint\_accel = diff(joint\_vel) / dt1;

joint\_accel\_norm = vecnorm(joint\_accel, 2, 2);

% 计算关节抖动 (急动度 jerk) (rad/s³)

joint\_jerk = diff(joint\_accel) / dt1;

joint\_jerk\_norm = vecnorm(joint\_jerk, 2, 2);

% 计算平滑度 (基于功率谱密度的逆)

fft\_data = abs(fft(qs(1:index\_limit,:), [], 1));

smoothness = sum(fft\_data(1:floor(end/2), :), 1) ./ sum(fft\_data, 1);

% 计算末端执行器位置（带错误处理）

tcp\_pos = zeros(index\_limit, 3);

valid\_points = true(index\_limit, 1);

for i = 1:index\_limit

try

tcp\_pos(i, :) = robot.fkine(qs(i, :)).t';

catch e

valid\_points(i) = false;

warning('正向运动学计算失败在点 %d: %s', i, e.message);

if i > 1 && any(valid\_points(1:i-1))

prev\_valid = find(valid\_points(1:i-1), 1, 'last');

tcp\_pos(i, :) = tcp\_pos(prev\_valid, :);

else

tcp\_pos(i, :) = robot.fkine(robot.qn).t';

end

end

end

% 计算末端执行器速度 (m/s)

tcp\_vel = diff(tcp\_pos) / dt1;

tcp\_vel\_norm = vecnorm(tcp\_vel, 2, 2);

% 计算末端执行器加速度 (m/s²)

tcp\_accel = diff(tcp\_vel) / dt1;

tcp\_accel\_norm = vecnorm(tcp\_accel, 2, 2);

% 关节空间性能评估

stepnorm = vecnorm(diff(qs(1:index\_limit,:)),2,2);

% 打印评估结果

fprintf('\n===== 机械臂运动评估=====\n');

% 关节运动指标

if ~isempty(stepnorm)

fprintf('最大关节变化率: %.4f rad/step\n', max(stepnorm));

fprintf('平均关节变化率: %.4f rad/step\n', mean(stepnorm));

else

fprintf('关节变化率: 数据不足无法计算\n');

end

if ~isempty(joint\_vel\_norm)

fprintf('最大关节速度: %.4f rad/s\n', max(joint\_vel\_norm));

fprintf('平均关节速度: %.4f rad/s\n', mean(joint\_vel\_norm));

else

fprintf('关节速度: 数据不足无法计算\n');

end

if ~isempty(joint\_accel\_norm)

fprintf('最大关节加速度: %.4f rad/s²\n', max(joint\_accel\_norm));

fprintf('平均关节加速度: %.4f rad/s²\n', mean(joint\_accel\_norm));

else

fprintf('关节加速度: 数据不足无法计算\n');

end

if ~isempty(joint\_jerk\_norm)

fprintf('最大关节抖动: %.4f rad/s³\n', max(joint\_jerk\_norm));

fprintf('平均关节抖动: %.4f rad/s³\n', mean(joint\_jerk\_norm));

else

fprintf('关节抖动: 数据不足无法计算\n');

end

% 平滑度指标

if ~isempty(smoothness)

fprintf('\n关节平滑度指标 (越高越平滑):\n');

for j = 1:min(6, length(smoothness))

fprintf('关节%d: %.4f\n', j, smoothness(j));

end

else

fprintf('\n关节平滑度: 数据不足无法计算\n');

end

% 末端执行器指标

if ~isempty(tcp\_vel\_norm)

fprintf('\n末端执行器最大速度: %.4f m/s\n', max(tcp\_vel\_norm));

fprintf('末端执行器平均速度: %.4f m/s\n', mean(tcp\_vel\_norm));

else

fprintf('\n末端执行器速度: 数据不足无法计算\n');

end

if ~isempty(tcp\_accel\_norm)

fprintf('末端执行器最大加速度: %.4f m/s²\n', max(tcp\_accel\_norm));

fprintf('末端执行器平均加速度: %.4f m/s²\n', mean(tcp\_accel\_norm));

else

fprintf('末端执行器加速度: 数据不足无法计算\n');

end

% 8. 绘制分析图表

if ~skip\_part2 && ~isempty(joint\_vel\_norm) && ~isempty(joint\_accel\_norm)

% 绘制关节速度和加速度曲线

figure('Name','关节速度和加速度分析','NumberTitle','off');

subplot(2,1,1);

plot((1:length(joint\_vel\_norm))\*dt1, joint\_vel\_norm);

title('关节速度分析');

xlabel('时间 (s)');

ylabel('关节速度 (rad/s)');

grid on;

subplot(2,1,2);

plot((1:length(joint\_accel\_norm))\*dt1, joint\_accel\_norm);

title('关节加速度分析');

xlabel('时间 (s)');

ylabel('关节加速度 (rad/s²)');

grid on;

end

if ~skip\_part2 && ~isempty(tcp\_vel\_norm) && ~isempty(tcp\_accel\_norm)

% 绘制末端执行器速度和加速度曲线

figure('Name','末端执行器速度和加速度分析','NumberTitle','off');

subplot(2,1,1);

plot((1:length(tcp\_vel\_norm))\*dt1, tcp\_vel\_norm);

title('末端执行器速度分析');

xlabel('时间 (s)');

ylabel('速度 (m/s)');

grid on;

subplot(2,1,2);

plot((1:length(tcp\_accel\_norm))\*dt1, tcp\_accel\_norm);

title('末端执行器加速度分析');

xlabel('时间 (s)');

ylabel('加速度 (m/s²)');

grid on;

end