实验 5：机械臂抓取与搬运实验

第 4 组 金加康 王彦程 孟令祎 方辰 李有为 李俊粤

2025 年 10 月 14 日

1 实验目的和要求

1.1 实验目的

1. 了解机械臂的轨迹规划方法。
2. 掌握机械臂的逆运动学求解方法。
3. 学习对机械臂的搬运任务进行目标点的选取和轨迹规划。

1.2 任务要求

1. 编写程序控制 ZJU-I 型机械臂，实现木块抓取与搬运，具体流程为
   (1) 机械臂从零位置启动，运行至起始区域；
   (2) 启动真空吸爪，抓取起始区域内的木块，移动至 A 点（370, −90, 115）；
   (3) 移动过程中控制木块从 A 点沿直线路径运动至 B 点（288, −288, 115）；
   (4) 控制从 B 点到达目标区域，目标区域位置为机械臂 1 号关节旋转角度 90°所在位置；
   (5) 抓取第二个木块放置到目标区域，并堆叠在第一个模块上，二者姿态保持一致。
2. 程序中需要编写正、逆运动学求解代码、轨迹规划代码，要求机械臂无碰撞、所有关节速度平滑。

2 结果与分析

2.1 任务完成准确性

实验成功完成了两个木块的抓取、搬运和堆叠任务：

1. 第一个木块成功从起始位置搬运至目标区域，经过染色池时实现了直线运动。

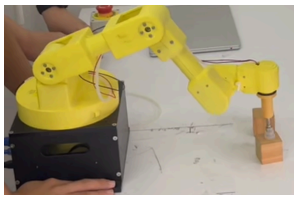


图 1 起始位置

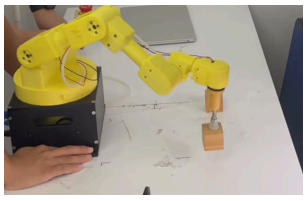


图 2 进入染色池

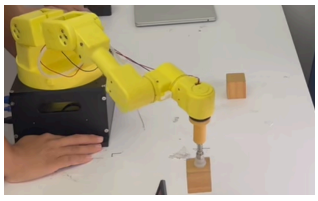


图 3 离开染色池

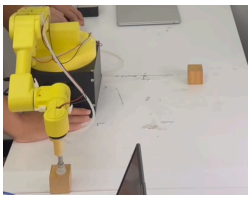
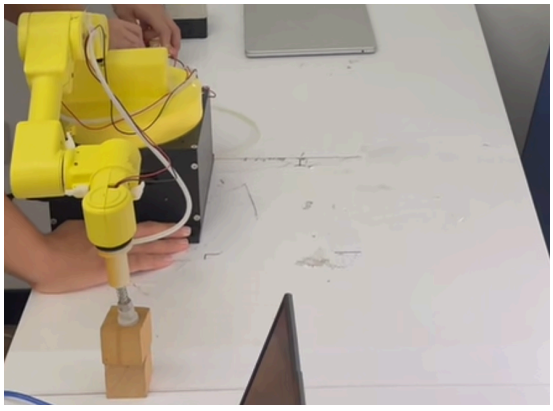


图 4 目标区域

2. 第二个木块成功堆叠在第一个木块上方，两者姿态保持一致。



3. 整个过程中机械臂未发生碰撞，各关节角度、速度、加速度均在限制范围内。

2.2 关键技术点

1. **速度边界条件优化**：通过在笛卡尔空间规划速度并利用雅可比矩阵转换到关节空间，实现了直线段前后的平滑过渡。
2. **中间点策略**：在返回起始区域时设置中间过渡点，有效避免了关节角的剧烈变化和潜在碰撞。
3. **逆运动学解选择**：对于多解情况，选择与当前关节状态最接近且无奇异性的解，保证了运动的连续性。
4. **时间分配合理**：各段运动时间根据实际距离和速度限制合理分配，既保证了平滑性又提高了效率。

3 实验内容与原理

3.1 总体规划思路

机械臂操纵物块的过程分为 7 个关键状态，整体流程为：

1. 第一个物块搬运流程：
   (1) q_0：机械臂初始零位姿态。
   (2) q_1：移动到第一个木块上方的抓取位置。
   (3) q_A：抓取后移动至染色池入口点 $A\left(370, -90, 115, \pi, 0, -\frac{\pi}{2}\right)$。
   (4) q_B：沿直线运动至染色池出口点 $B\left(288, -288, 115, \pi, 0, -\frac{\pi}{2}\right)$。
   (5) q_7：移动至目标区域上方的过渡点。
   (6) q_2：放置第一个木块于目标位置。
2. 第二个物块搬运流程：
   (1) $q_2 \to q_7$：返回目标区域上方的过渡点。
   (2) $q_7 \to q_{7'}$：将关节 1 旋转 $-90°$ 调整姿态。
   (3) $q_{7'} \to q_3^{\text{up}} \to q_3$：通过中间点避免碰撞，到达第二个木块位置。
   (4) $q_3 \to q_6 \to q_A$：抓取第二个木块并移动至染色池入口。
   (5) $q_A \to q_B$：沿直线运动通过染色池。
   (6) $q_B \to q_5 \to q_4$：移动至第一个木块上方并堆叠放置。

3.2 轨迹规划方法

3.2.1 五次多项式轨迹规划

五次多项式共有六个参数 $a_0, a_1, a_2, a_3, a_4, a_5$，通过给定的初始位置、初始速度、初始加速度、目标位置、目标速度、目标加速度，可以求解出这六个参数，从而得到一个五次多项式函数。该函数可以保证关节位置、速度、加速度的平滑性。

1. 五次多项式的形式：

$$\begin{cases} \theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \\ \dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \\ \ddot{\theta}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \end{cases} \tag{1}$$

2. 约束条件：

$$\begin{cases} \theta(0) = \theta_0, \dot{\theta}(0) = \dot{\theta}_0, \ddot{\theta}(0) = \ddot{\theta}_0 \\ \theta(T) = \theta_T, \dot{\theta}(T) = \dot{\theta}_T, \ddot{\theta}(T) = \ddot{\theta}_T \end{cases} \tag{2}$$

3. 矩阵形式：

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} \theta_0 \\ \theta_T \\ \dot{\theta}_0 \\ \dot{\theta}_T \\ \ddot{\theta}_0 \\ \ddot{\theta}_T \end{pmatrix} \tag{3}$$

本实验中，在 A、B 两点设置了非零的速度边界条件，以保证机械臂在直线段前后的运动更加连续平滑。具体做法是在笛卡尔空间中规划速度，然后用雅可比矩阵将其转化为关节空间的速度。

3.2.2 直线轨迹规划（三次样条插值）

为保证物块在染色池中的直线运动，采用在笛卡尔空间进行三次样条插值的方法。给定起点位姿 $\boldsymbol{p}_A$、终点位姿 $\boldsymbol{p}_B$ 和运动时间 T，在笛卡尔空间生成 1000 个等间距插值点：

$$\boldsymbol{p}(t_i) = \boldsymbol{p}_A + (\boldsymbol{p}_B - \boldsymbol{p}_A) \cdot \frac{i}{N}, \quad i = 0, 1, ..., N-1 \tag{4}$$

对每个笛卡尔位姿 $\boldsymbol{p}(t_i) = \left[x, y, z, r_x, r_y, r_z\right]^T$，利用逆运动学求解器求解对应的关节角：

$$\boldsymbol{\theta}(t_i) = \mathrm{IK}(\boldsymbol{p}(t_i)) \tag{5}$$

然后利用 `move()` 函数直接驱动机械臂到达各关节角位置，实现笛卡尔空间的直线运动。

3.2.3 带中间点的三次多项式规划

当机械臂从目标区域返回起始区域时，为避免关节角的剧烈变化和碰撞，设置了中间过渡点。采用两段三次多项式规划，确保在中间点处速度和加速度连续。

1. 两段三次多项式的形式：

$$\begin{cases} \theta_1(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ \theta_2(t) = a_4 + a_5 t + a_6 t^2 + a_7 t^3 \end{cases} \tag{6}$$

2. 约束条件：

$$\begin{cases} \theta_1(0) = \theta_0, \dot{\theta}_1(0) = \dot{\theta}_0 \\ \theta_1(T_1) = \theta_{\text{mid}}, \theta_2(0) = \theta_{\text{mid}} \\ \theta_2(T_2) = \theta_{\text{end}}, \dot{\theta}_2(T_2) = \dot{\theta}_{\text{end}} \\ \dot{\theta}_1(T_1) = \dot{\theta}_2(0), \ddot{\theta}_1(T_1) = \ddot{\theta}_2(0) \end{cases} \tag{7}$$

3.3 逆运动学求解

本实验采用自主编写的逆运动学求解器，基于 DH 参数和几何方法求解。求解器输入为末端执行器的位姿 $[x, y, z, \alpha, \beta, \gamma]$，输出为所有可能的关节角解。

关键步骤：

1. 根据末端位姿计算姿态矩阵的方向向量 $\boldsymbol{n}, \boldsymbol{o}, \boldsymbol{a}$。
2. 求解关节 1 角度 θ_1：利用几何关系 $\theta_1 = \arctan2(A, B) - \arctan2\left(d_4, \pm\sqrt{A^2 + B^2 - d_4^2}\right)$。
3. 求解关节 5 角度 θ_5：$\theta_5 = \arcsin\left(a_y \cos\theta_1 - a_x \sin\theta_1\right)$。
4. 求解关节 6 角度 θ_6：利用姿态约束。
5. 求解关节 3 角度 θ_3：利用位置约束 $\theta_3 = \arccos\left(\frac{E^2 + F^2 - a_2^2 - a_3^2}{2a_2 a_3}\right)$。
6. 求解关节 2 角度 θ_2 和关节 4 角度 θ_4：利用几何关系。

逆运动学求解器会返回多组解（最多 8 组），在实际应用中选择最接近当前关节状态且无奇异性的解。

4 代码实现

本实验的核心代码包括三个主要模块：轨迹规划、逆运动学求解和主控制循环。

4.1 五次多项式轨迹规划

通过构建时间矩阵和边界条件矩阵，求解五次多项式系数：

```python
def quinticCurvePlanning(qStart, qEnd, vStart, vEnd, duration):
    # Time matrix for quintic polynomial
    timeMatrix = np.matrix([
        [0, 0, 0, 0, 0, 1],
        [duration**5, duration**4, duration**3, duration**2, duration, 1],
        [0, 0, 0, 0, 1, 0],
        [5*duration**4, 4*duration**3, 3*duration**2, 2*duration, 1, 0],
        [0, 0, 0, 2, 0, 0],
        [20*duration**3, 12*duration**2, 6*duration, 2, 0, 0]
    ])
    # Boundary conditions for each joint
    qArray = []
    for i in range(len(qStart)):
        qArray.append([qStart[i], qEnd[i], vStart[i], vEnd[i], 0, 0])
    qMatrix = np.matrix(qArray).T
    # Solve for coefficients
    aMatrix = timeMatrix.I * qMatrix
    return aMatrix.T

def quinticCurveExcute(CoefMatrix, t):
    # Evaluate polynomial at time t
    timeVector = np.matrix([t**5, t**4, t**3, t**2, t, 1]).T
    q = (CoefMatrix * timeVector).T.A[0]
    return q
```

4.2 直线轨迹规划

在笛卡尔空间生成 1000 个插值点，通过逆运动学求解对应的关节角序列：

```python
def splinePlanning(startPosition, endPosition):
    iks = IKSolver()
    num_line_points = 1000
    spline_points = []
    spline_kArray = []
    # Generate interpolation points in Cartesian space
    spline_points = [(startPosition + (endPosition - startPosition) * i /
                      num_line_points) for i in range(num_line_points)]
    # Solve IK for each point
    for i in range(num_line_points):
        spline_kArray.append(
            iks.solve(np.append(spline_points[i], [np.pi, 0, -np.pi/2]))[:, 2])
        # Check for NaN solutions
```

```
        contains_nan = np.isnan(spline_kArray[i]).any()
        if contains_nan:
            print("IK solution contains NaN!")
            return False
    return spline_kArray

def splineExcute(spline_kArray, t, duration):
    # Calculate index based on time
    index = int(t / duration * len(spline_kArray))
    if index >= len(spline_kArray):
        index = len(spline_kArray) - 1
    return spline_kArray[index]
```

4.3 主控制循环

根据时间窗口判断当前应执行的轨迹段，计算关节角并发送给机械臂：

```python
# main control loop
r.go_home()
while True:
    start = time.time()
    if t >= t_tot:
        print('Control Finished')
        r.go_home()
        break
    # execute current segment based on time
    q = None
    print_dots = False
    if t_points['O1'][0] <= t < t_points['O1'][1]:
        q = vo1 * (t - t_points['O1'][0])
    elif t_points['1A'][0] <= t < t_points['1A'][1]:
        q = quinticCurveExcute(planners['1A'], t - t_points['1A'][0])
    # ... Other trajectory segments
    if q is not None:
            if print_dots:
                print("... ")
            r.syncMove(np.reshape(q, (6, 1)))
    t += 0.02  # 20ms control period
    q_array.append(r.syncFeedback())
```

5 实验总结

本实验成功实现了机械臂的抓取与搬运任务，主要成果包括：

1. 掌握了五次多项式和三次样条插值两种轨迹规划方法，理解了它们在不同场景下的应用优势。
2. 实现了自主编写的逆运动学求解器，能够准确求解给定末端位姿对应的关节角，为轨迹规划提供了基础。
3. 学会了合理设置速度边界条件和中间过渡点，实现了平滑且高效的运动轨迹。
4. 完成了从理论到实践的完整流程，包括任务分析、轨迹规划、代码实现和实验验证。

通过本次实验，深入理解了机器人轨迹规划的原理和方法，掌握了实际编程实现的技巧，为后续更复杂的机器人控制任务打下了坚实基础。