2.1.2 机器人运动学建模

本文选用了 7Bot 六自由度轻型机器人作为书写的执行器，该机器人共包含六个转动关节。角度精确度 0.21°，空间重复定位精度小于 0.5mm，工作范围为 434mm。在六组关节驱动舵机中，其中四组完成 60°转动最短用时 0.17s，另外两组用时 0.09s，在关节串行运动方式下，执行器末端完成最大工作空间包络循迹的理想最短用时为 0.86s，使其能够满足此次书写操作任务，该机器人的几何参数与关节轴如图 2.2 所示。

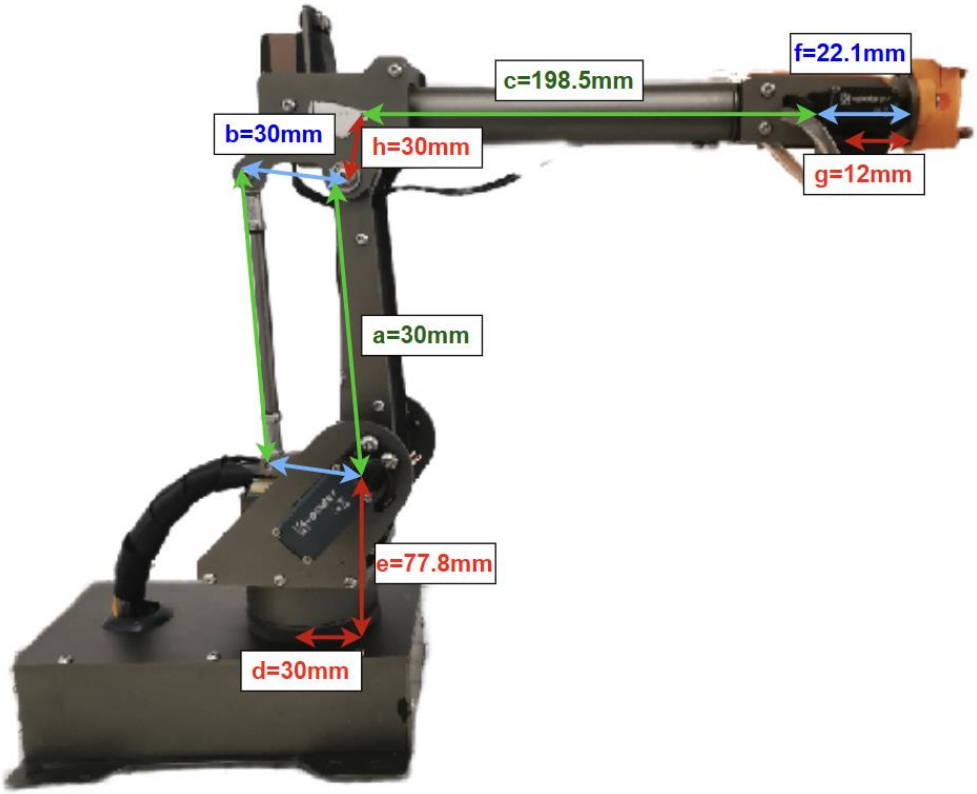


图 2.2 机器人关节与几何参数示意图

根据上文对 D-H 参数法的定义，建立 D-H 坐标系如图 2.3 所示。

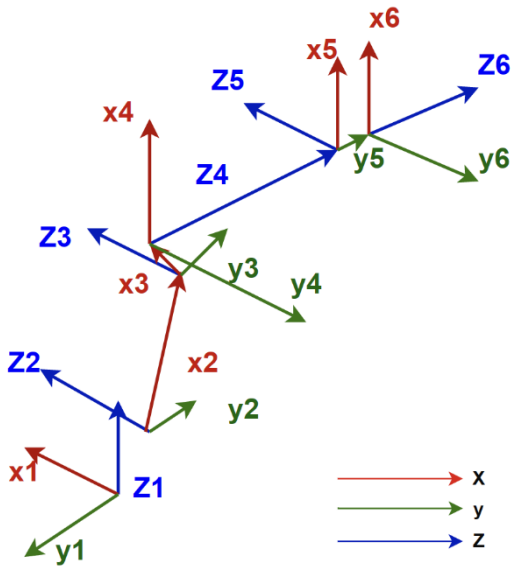


图 2.3 机器人的 D-H 坐标系

图 2.3 涉及的各个关节轴性能相关参数如表 2.1 所示，表中列出了各关节轴的运动范围，在后续的实验中应该尽量避免达到关节极限。由于机器人存在一个平行四边形结构，导致关节 2 和关节 3 存在限位，将通过公式(2.4)以及后续建模的方法加以修正。

$$Angle_{3real} = 215^\circ - \theta_2 - \theta_3 \tag{2.4}$$

表 2.1 机器人各个关节轴相关参数

关节	移动范围(°)
关节 1	0~180
关节 2	0~180（与关节 3 存在限位）
关节 3	0~180（与关节 2 存在限位）
关节 4	0~180
关节 5	0~360
关节 6	0~360

上文依据 D-H 参数法建立了如图 2.3 的机器人坐标系，再结合图 2.2 机器人几何参数，可以列写机器人的连杆参数如表 2.2。

表 2.2 书法机器人 D-H 连杆参数

连杆i	d_i(mm)	a_{i-1}(mm)	α_{i-1}	$\theta_i(t)$
1	80	30	$\pi/2$	θ_1(t)

2	0	120	0	θ_2(t)
3	0	30	$\pi/2$	θ_3(t)
4	198.5	0	$\pi/2$	θ_4(t)
5	0	0	-$\pi/2$	θ_5(t)
6	0	0	0	θ_6(t)

2.2 机器人正向运动学求解

根据式(2.3)以及表 2.2 中的参数，可以列写出每两个邻轴之间的齐次变换矩阵${}_{i}^{i-1}T$：

$$
{}_{1}^{0}T=\begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & 0.03 \\ 0 & 0 & -1 & -0.08 \\ sin\theta_1 & cos\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\quad {}_{2}^{1}T=\begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & 0.12 \\ sin\theta_2 & cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}_{3}^{2}T=\begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & 0.03 \\ 0 & 0 & -1 & 0 \\ sin\theta_3 & cos\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\quad {}_{4}^{3}T=\begin{bmatrix} cos\theta_4 & -sin\theta_4 & 0 & 0 \\ 0 & 0 & -1 & -0.1985 \\ sin\theta_4 & cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}
$$

$$
{}_{5}^{4}T=\begin{bmatrix} cos\theta_5 & -sin\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sin\theta_5 & -cos5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\quad {}_{6}^{5}T=\begin{bmatrix} cos\theta_6 & -sin\theta_6 & 0 & 0 \\ sin\theta_6 & cos\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

将式(2.5)中的齐次变换矩阵按顺序相乘，就可以得到书法机器人末端执行器坐标系{6}以机器人基坐标{0}为参考的变换矩阵${}_{6}^{0}T$，实现书法机器人从关节空间到操作空间的映射。

$$
{}_{6}^{0}T={}_{1}^{0}T{}_{2}^{1}T{}_{3}^{2}T{}_{4}^{3}T{}_{5}^{4}T{}_{6}^{5}T \tag{2.6}
$$

2.3 机器人逆向运动学求解

正向运动学通过获取六个关节角度，使用齐次变换矩阵相乘的方法，计算书法机器人末端执行器的位置和姿态。相反地，逆向运动学则根据给定的末端执行器位姿，反向计算各个关节轴角度。对于具有转动或者移动关节的任意六自由度机器人，其逆向运动学问题有解。通常可以使用反变换法获得逆向运动学问题的解析解，或者使用几何法求取数值解。

对于本文使用的 7Bot 机器人，手臂的所有关节都处于同一个平面上，因此可以使用解析法来描述求取逆向运动学的解，并计算出各关节期望角度。由于手腕关节 4 到关节 6 的变换与关节 1 到关节 3 无关，因此可以将 7Bot 机器人的运动学求解分为两部分：关节 1 到关节 3 的逆向位置运动学和关节 4 到关节 6 的逆向姿态运动学 [21]。

首先对关节 1 到关节 3 进行求解，将机器人末端执行器投影到基坐标的$\hat{x}-\hat{y}$平面，得到投影在基平面相对于关节 1 的位置（${}^0P_6^x$，${}^0P_6^y$）。由于关节 2 到关节 6 在同一平面上，所以末端执行器的位置是一个包含θ_1的函数，如图 2.4 所示，其中θ_1是投影与$\hat{x}$轴的夹角。

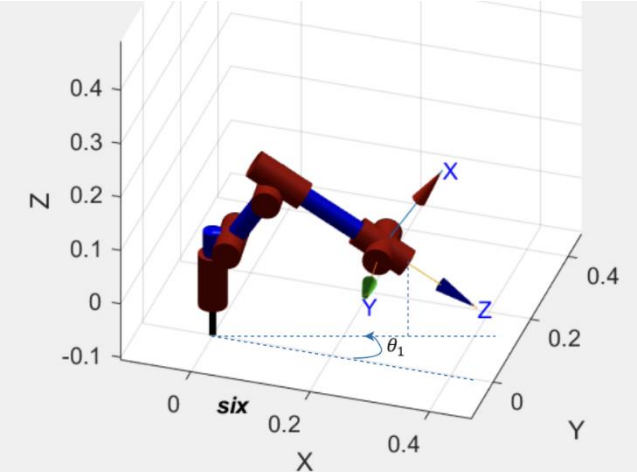


图 2.4 θ_1与基平面投影关系

$$\theta_1 = Atan2（{}^0P_6^y，{}^0P_6^x） \tag{2.7}$$

在同一个平面的关节 2、关节 3 和关节 6 可以构成一个三角形，解这个三角形就可求得θ_2和θ_3。记末端执行器相对于关节 2 的位置描述为（${}^2P_6^x$，${}^2P_6^y$，${}^2P_6^z$），利用余弦定理可得如下式子：

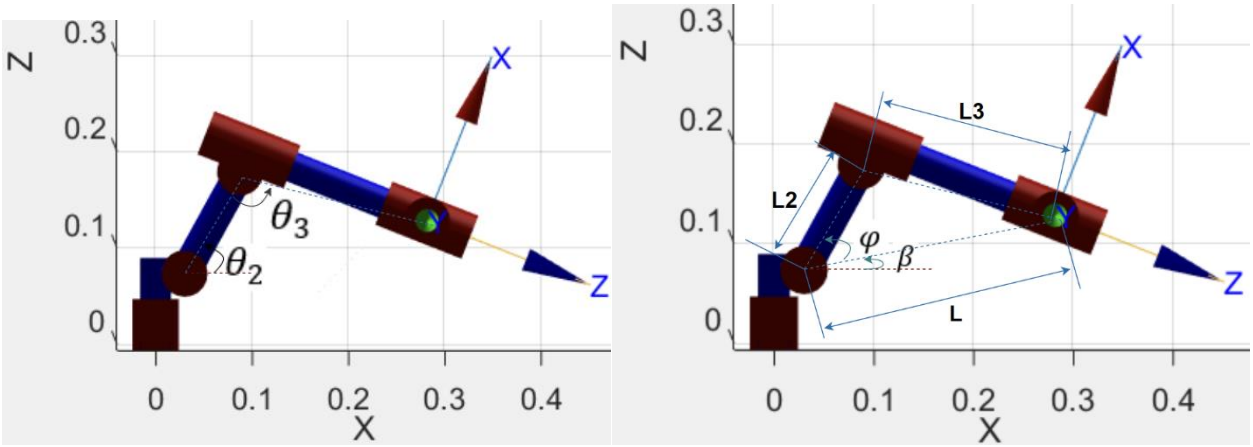


图 2.5 θ_2和θ_3几何关系

$$\theta_2 = \varphi \pm \beta \tag{2.8}$$

$$\theta_3 = arccos\frac{L^2-L_2^2-L_3^2}{2L_2L_3} \tag{2.9}$$

$$L = \sqrt{（{}^2P_6^x）^2 + （{}^2P_6^z）^2} \tag{2.10}$$

$$\beta = Atan2（{}^2P_6^z，{}^2P_6^x） \tag{2.11}$$

$$\varphi = arccos\frac{L_3^2 - L_2^2 - L^2}{2LL_2} \tag{2.12}$$

其中L_2和L_3可以在 D-H 参数表中查得。经过上述计算之后，也许会产生多个逆向解的集合，这时需要依据关节的运动范围进行合理地选取，最终可以求得前三个关节的角度，即完成了逆向位置运动学。

针对关节 4 到关节 6，从图 2.3 可以看出这些关节的关节轴都相交于一点，即末端执行器的坐标原点。由此可见，末端执行相对于连杆系{4}的姿态由最后三个关节的角度决定。因此可以先令$\theta_4 = 0$，参考式(2.6)可得:

$$^0_6R = {}^0_4R_{\theta_4=0} * {}^4_6R_{\theta_4=0} \tag{2.13}$$

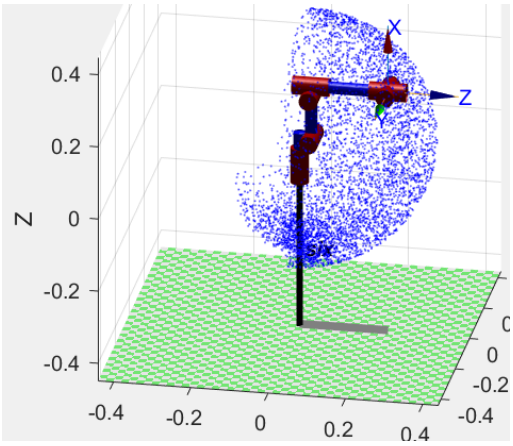
其中$^0_4R_{\theta_4=0}$表示在$\theta_4 = 0$时，从关节 1 到关节 4 的旋转矩阵，两边再同时左乘该矩阵的逆，最终可以求得关节 4 到末端执行器的旋转矩阵$^4_6R_{\theta_4=0}$：

$$^4_6R_{\theta_4=0} = {}^0_4R^{-1}_{\theta_4=0} * {}^0_6R \tag{2.14}$$

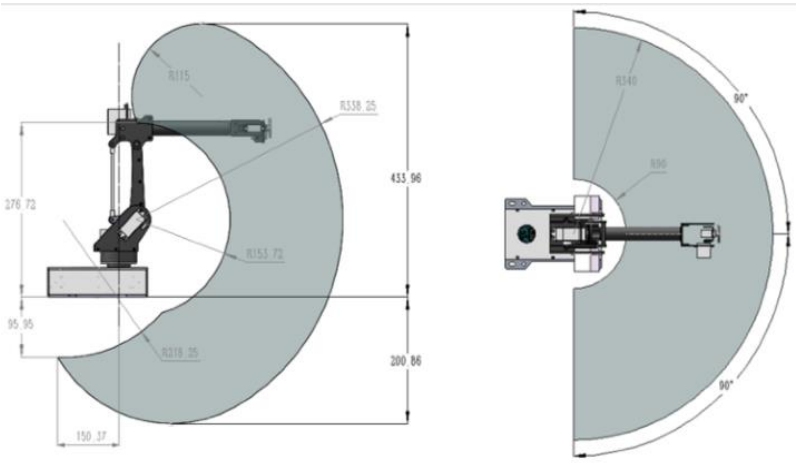
最后可以通过解欧拉角的方法得到最后三个关节角度，完成机器人逆向运动学求解。

2.4 机器人运动学仿真

为验证机器人运动学模型以及运动学公式推导的正确性，利用 MATLAB 进行仿真测试。本文采用蒙特卡洛法对机器人末端执行器的可达空间进行遍历分析[22]，该方法利用大量重复实验来估计概率或者期望。设置随机次数$N = 5000$，最终得到由 5000 个末端执行器随机位置构成的云图，将其与实际的工作空间进行比对，证明所建立模型具有一定的准确性。同时对书法机器人的运动进行仿真，通过设定起点位姿和终点位姿，让机器人进行移动，验证了运动学公式推导的正确性。



（a）仿真的工作空间范围



（b）实际的工作空间范围

图 2.6 验证工作空间的一致性

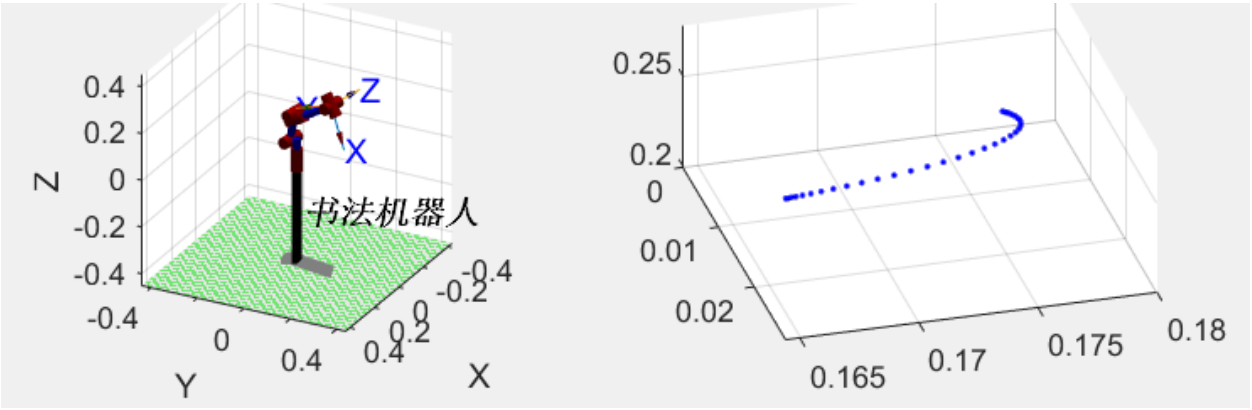


图 2.7 书法机器人在仿真环境下的运动轨迹

第三章 书法机器人系统总体方案设计

本章首先对书法机器人整体控制系统进行概述，说明系统中各个模块之间的联系。接着，介绍书法机器人的软件结构，阐述各个模块如何协调工作，同时介绍了本课题采用的轨迹规划方法、目标位姿获取方案和力反馈方法。

3.1 书法机器人控制系统概述

本文所介绍的书法机器人系统主要包括三部分：手控器与主控电脑之间的通信，在主控端中目标位姿到关节角度的映射转换，以及主控电脑与书法机器人之间的通信。控制系统主端作为系统中心，一方面，需要接收力反馈手控器传递的末端位姿作为执行器的目标，并对手控器给予适当的力反馈信号；另一方面，需要通过逆运动学和样条插补将获取的位姿转换为关节角度，并通过串口将关节信息发送给下位机进行执行，系统结构如图 3.1。

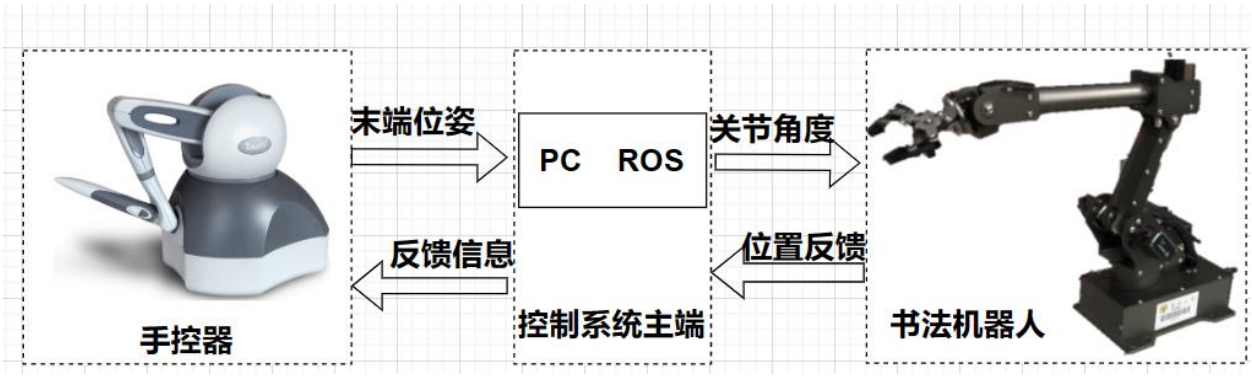


图 3.1 书法机器人系统结构

3.2 书法机器人的软件设计

3.2.1 软件开发环境

本文的书法机器人控制软件基于 Moveit 库函数进行开发，开发环境是 Noetic 版本的 ROS，使用的开发语言是 C/C++。本课题的书法机器人控制系统涉及多个进程间的并发通信，而 ROS 采用将进程封装为节点的形式，提供了优良的通信模式。对于机器人的运动规划任务，Moveit 在其中起着关键作用，其功能涵盖运动学正逆解算、运动规划以及碰撞检测所需的多项关键技术。作为 ROS 系统中的运动规划库，Moveit 还提供丰富的 API（Application Programming Interface）接口，封装了常用的机器人运动规划操作函数，使得机器人在 ROS 系统上的初始化和调试更加简单高效[23]。

ROS 的核心是将每个进程封装为节点，每个节点都具有自己的输入和输出，并可以独立运行和启动。这意味着可以将整个系统拆分为功能各异的节点，然后按顺序启动这些节

点以实现整体功能。节点之间多样灵活的通信方式是本文选择使用 ROS 的原因[24]。节点之间可以使用话题（Topic）、服务（Service）和动作（Action）三种方式进行通信，不同通信机制适用于不同的应用场景，本文将根据书法机器人系统要实现的具体功能进行选择。

3.2.2 软件结构设计

利用 ROS 提供的节点之间的通讯方式，可以将整个系统软件按照功能进行模块化分割，从而方便后续的功能设计、验证以及功能模块的组合。具体软件结构如图 3.2 所示，书法机器人软件结构主要分为字符获取与处理、书写实现两个部分。

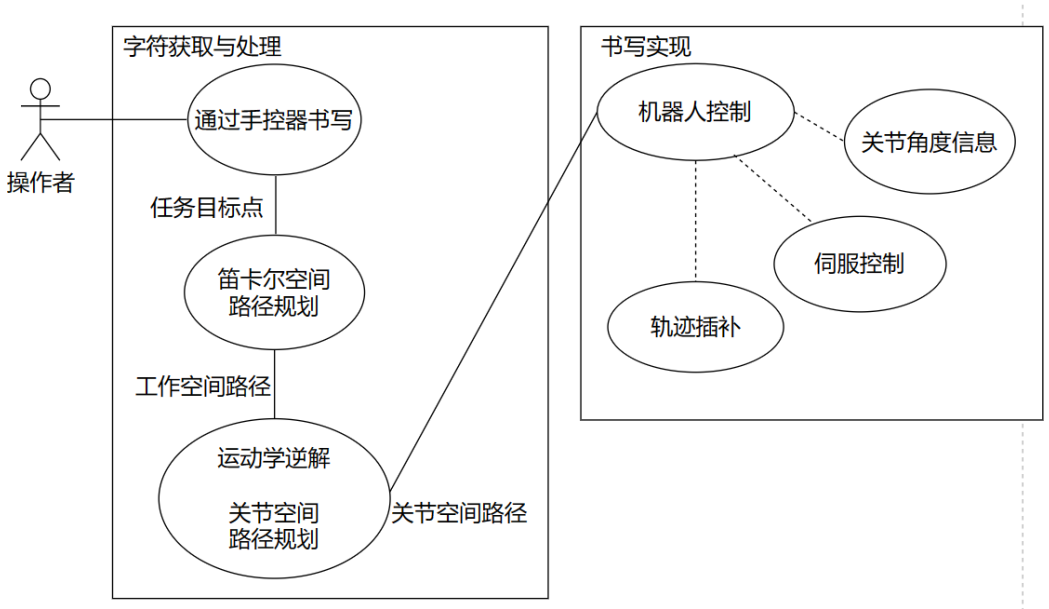


图 3.2 书法机器人软件结构用例图

从图中可以看出，对于第一个模块，首先要实现手控器末端目标位姿的获取；接着，对获取的目标位姿和当前位姿之间的路径进行笛卡尔空间规划，得到工作空间路径，即由多个点组成的运动轨迹；最后，通过运动学逆解和关节空间的路径规划，得到包含关节角和时间戳的数组，即机器人的关节空间路径，并将其发送给书法机器人控制模块。

图 3.3 展示了书法机器人系统的 ROS 节点，为实现第一个模块的功能，首先编写一个订阅节点（subscriber）来监听手控器发布的位姿信息，并且将其作为目标位姿；然后使用 Moveit 的 API 对目标位姿和当前位姿之间的轨迹进行路径规划和逆向运动学求解；最后，通过动作客户端（Action Client）的/goal 话题将计算得到的关节空间路径发送到下一个模块，完成一次字符的获取与处理。

在第二个模块中，将上一模块发布的关节空间路径信息作为输入。首先，需要对所获得的关节角度进行插补处理，以获得更加平滑的路点信息；接着，将插补后的关节角信息发送给伺服控制器，可以在伺服控制器中使用 PID 等算法对关节进行控制；最后，通过

USART（Universal Synchronous/Asynchronous Receiver/Transmitter）将关节角信息发送给 Arduino 以执行关节指令。

如图 3.3 所示，为实现第二个模块的功能，首先编写一个动作服务端（Action Server）接收客户端发出的路径信息，并且将获取到的所有路径信息点进行样条插补处理，然后通过发布（Publish）的方式将该信息发送出去，并将结果通过话题/result 反馈给客户端以实现状态更新；编写一个伺服节点订阅上述包含一系列由六个关节角度组成的路点信息，然后通过串口将其发送给下位机以驱动舵机，完成一次完整的书写过程。

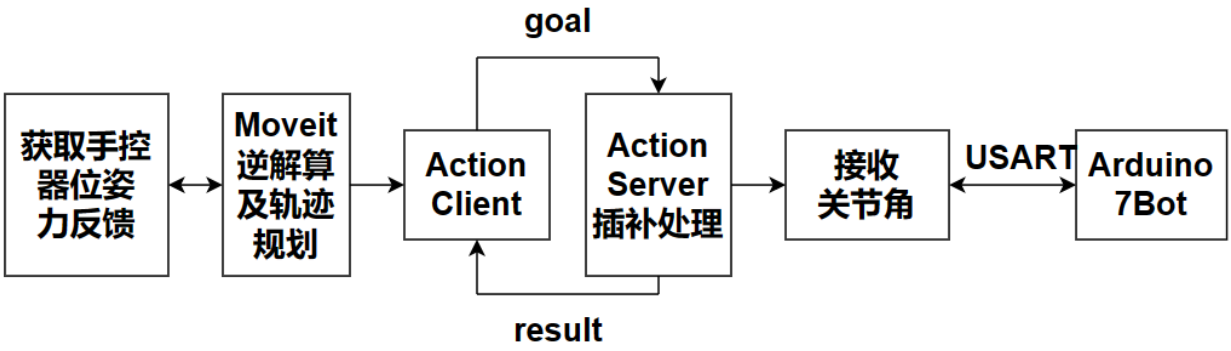


图 3.3 书法机器人 ROS 节点示意图

3.3 书写轨迹规划

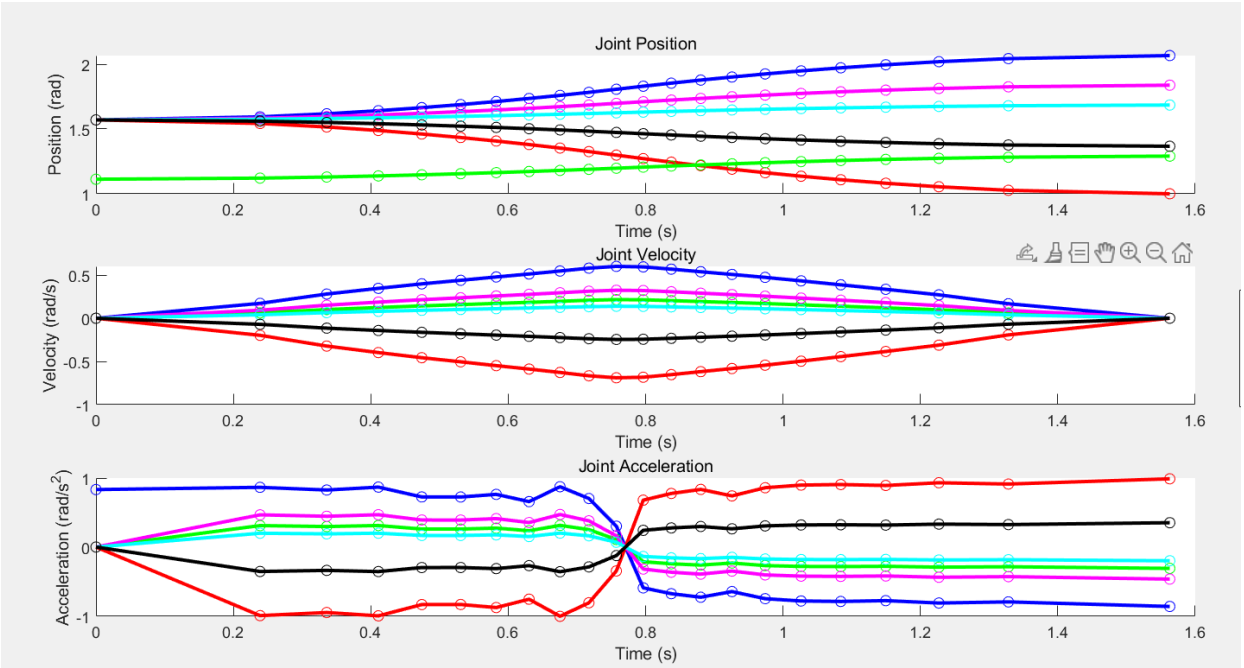
对于书法机器人来说，不仅需要确保书写过程的起点位姿和终点位姿与规划路径保持一致，还需要确保中间过程符合预定的运动轨迹，并且满足规定的位置、速度和加速度条件。然而，逆向运动学求解得到的数据可能不符合实际机器人的使用要求，因此需要对轨迹进行进一步处理。通常可以将其分为关节空间轨迹规划和笛卡尔空间轨迹规划两部分。

3.3.1 关节空间轨迹规划

对于本文使用的 7Bot 机器人，由其关节变量$q_i(i = 1,2 \dots 6)$构成的函数可以描述一条包含许多个路点的运动轨迹。关节空间规划是一种重新生成符合机器人实际运动要求轨迹的方法[25]，具体步骤是：首先，由用户给定一系列离散的原轨迹点；接着，提取轨迹中第一个路点作为初始插补点，最后一个路点作为结束插补点，以及其间一些必要的轨迹路点；最后，利用关节变量函数求导得到关节角速度和加速度的表达式，进行三次或者五次样条插补处理以得到一条连续、平滑的轨迹。其具有计算相对简单且不会发生机构奇异现象的优点。

给定起点 A 关节角θ_A和终点 B 关节角θ_B，经过 Moveit 逆向运动学求解之后得到的轨迹信息如图 3.4 所示。每个点代表一组数据，该规划轨迹共由 22 个路点组成，可以看出每个路点之间的时间差是不相等的，这就意味着书法机器人在执行时可能会发生姿态的跳变，造成冲击的危险；而且相邻路点之间的变化不够平滑，这会造成书写过程的不稳定，使得控制难度增加，所以接下来将利用三次和五次样条插补来优化轨迹。

图 3.4 机器人从起点运动到终点的原始轨迹信息



1）三次多项式

多项式差值一般是保持起点和终点不变，增加中间的规划点数，所以多为奇次多项式。其中三次多项式最为简单高效，适用场合最为广泛，其函数$\theta(t)$可表达为式(3.1)：

$$\theta(t) = a + bt + ct^2 + dt^3 \tag{3.1}$$

对于该多项式进行一次求导可得关节的角速度，进行二次求导可得关节的角加速度，具体如式(3.2)：

$$\dot{\theta}(t) = b + 2ct + 3dt^2$$

$$\ddot{\theta}(t) = 2c + 6dt \tag{3.2}$$

可以看出三次多项式共有四个未知系数，所以需要四个约束条件解未知数，现在给定路径的起点角度θ_0和终点角度θ_p，以及起点速度v_0和终点速度v_p，带入到上述式(3.1)和(3.2)可以解得：

$$s.t\begin{cases}\theta(0)=\theta_0\\ \theta(t)=\theta_p\\ \dot{\theta}(0)=v_0\\ \dot{\theta}(t)=v_p\end{cases}\Rightarrow\begin{cases}a=\theta_0\\ b=v_0\\ c=\frac{3}{t^2}(\theta_p-\theta_0)-\frac{2}{t}v_0-\frac{1}{t}v_p\\ d=-\frac{2}{t^3}(\theta_p-\theta_0)+\frac{1}{t^2}(v_0+v_{p)}\end{cases}\tag{3.3}$$

将式(3.3)代入式(3.1)，可以解得三次多项式函数$\theta(t)$。根据上述原理，编写三次样条插补算法对原始轨迹进行处理。与三次多项式插值不同，三次样条插补使用多个多项式来拟合每个小区间上的数据点，使得路点之间更加密集。处理后的轨迹路点增加到 70 个点，如图 3.5 所示。和原始轨迹进行对比可以看出，三次样条插补之后，路点之间的时间戳变为等步长增长，而且位置、速度和加速度曲线变得相对更加平滑。

然而，这样引入了加速度不连续的新问题，可以观察到每个关节的初始速度并不是从零开始，而且所有关节在轨迹末端的速度未能回归到零。与原始轨迹信息相比，加速度也发生变化，未能完全还原加速度曲线。这会使机器人由于惯性，在本该结束书写后依然移动一小段距离，这就可能引发“丢步”的情况。而在实际书写中，可以通过降低机器人的运动速度来减小这个问题带来的影响。

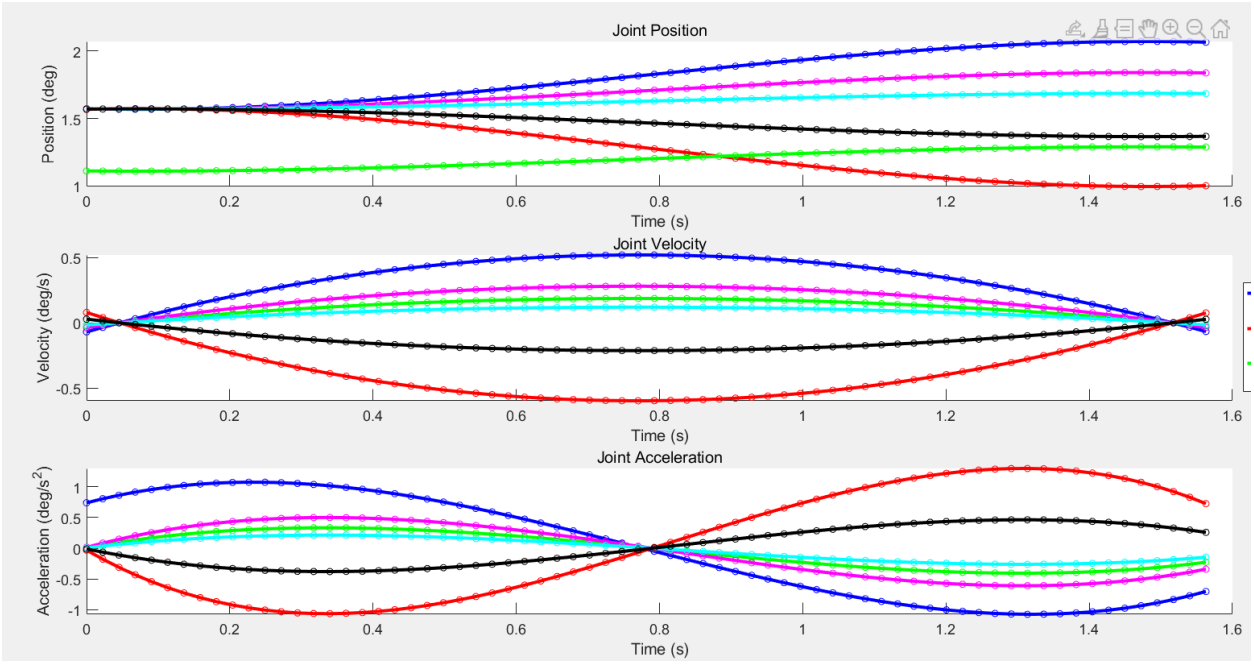


图 3.5 三次样条插补结果示意图

2）五次多项式

为使关节角的始末速度回归到零，并获得更加平滑的加速度曲线，这时需要采用更高阶的奇次多项式，接下来介绍五次多项式的插值算法[26]，函数$\theta(t)$可表达为式(3.4)：

$$\theta(t)=a+bt+ct^2+dt^3+et^4+ft^5\tag{3.4}$$

对该式进行一次求导可得关节的角速度，进行二次求导可得关节的角加速度：

$$\dot{\theta}(t) = b + 2c\mathrm{t} + 3dt^2 + 4et^3 + 5ft^4$$

$$\ddot{\theta}(t) = 2c + 6d\mathrm{t} + 12et^2 + 20ft^3 \tag{3.5}$$

相对于三次多项式的位置和速度约束，五次多项式包含六个未知系数，所以需要增加起点加速度a_0和终点加速度a_p，带入到式(3.4)和式(3.5)可以解得：

$$s.t\begin{cases}\theta(0)=\theta_0\\ \theta(t)=\theta_p\\ \dot{\theta}(0)=v_0\\ \dot{\theta}(t)=v_p\\ \ddot{\theta}(0)=a_0\\ \ddot{\theta}(t)=a_p\end{cases}\Rightarrow\begin{cases}a=\theta_0\\ b=v_0\\ c=\frac{a_0}{2}\\ d=\frac{20\theta_p-20\theta_0-(8v_p+12v_0)t-(3a_0-a_p)t^2}{2t^3}\\ e=\frac{30\theta_0-30\theta_p+(14v_p+16v_0)t+(3a_0-2a_p)t^2}{2t^4}\\ f=\frac{12\theta_p-12\theta_0-(6v_p+6v_0)t-(a_0-a_p)t^2}{2t^5}\end{cases} \tag{3.6}$$

依据上述原理，编写五次样条插补算法对原始轨迹进行处理，处理后的轨迹路点增加到 70 个点，如图 3.6 所示。与三次样条插补对比可以看出，利用五次多项式可以让关节始末速度都回归为零，加速度曲线也更加光滑。但是由于将加速度完全拟合，控制实际机器人会造成边缘的加速度抖动，也就是“龙格现象”[27]。

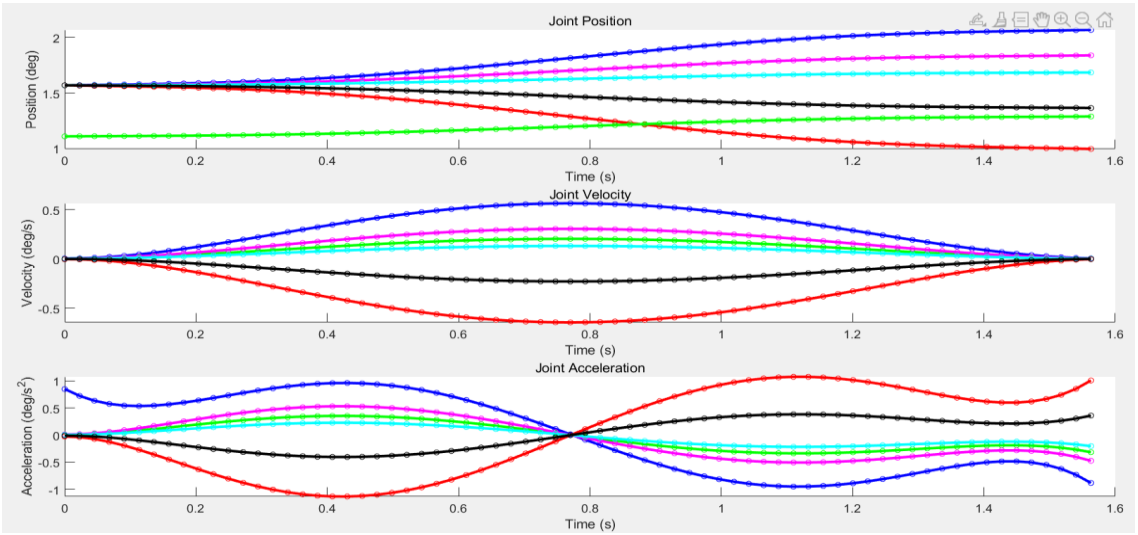


图 3.6 五次样条插补结果示意图

对比上述两种关节空间轨迹规划的方法，为避免“龙格现象”的加速度抖动超过实际机器人控制的加速度限制，综合考虑选择三次样条插补算法，并且通过降低实际舵机的运行速度来达到书写过程中各个关节的位置、速度和加速度都相对平滑的目的。

3.3.2 笛卡尔空间轨迹规划

上述在关节空间中进行的插补规划，通过确定关节角的路点序列，来确保机器人末端执行器能够到达设定的起始和结束位置。虽然计算简单，但是无法准确控制机器人连续变化时的位姿。而在书法机器人的书写过程中，往往需要保证笔画始末位置之间的轨迹连续，即要求末端执行器沿着连续的路径书写，这时就需要在笛卡尔空间中做轨迹规划[28]。

以书写汉字“十”为例，书法机器人需要完成两段连续直线的书写，所以要直接对末端位姿变化进行规划，先确定每个时刻末端需要运动到的位置，再通过逆向运动学解算获取对应的关节角变化序列，下面将介绍如何在笛卡尔空间中进行直线插补。

如图 3.7 所示，给定末端执行器的起点坐标$P_1 = (x_1, y_1, z_1)$和终点坐标$P_2 = (x_2, y_2, z_2)$，期望末端的运动轨迹沿直线从点P_1到达点P_2点。

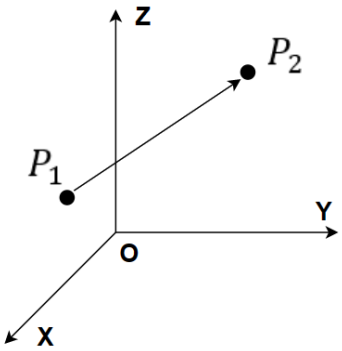


图 3.7 轨迹直线插补

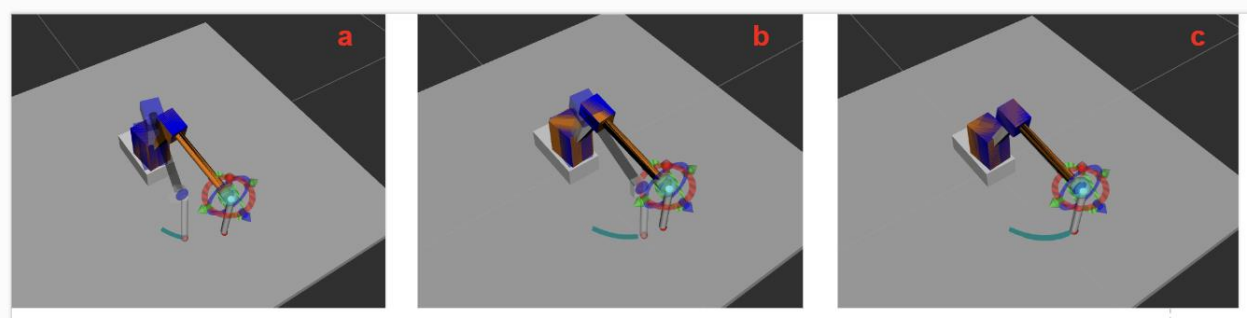
轨迹始末两点之间的距离可以表示为：

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^1 + (z_2 - z_1)^2} \tag{3.7}$$

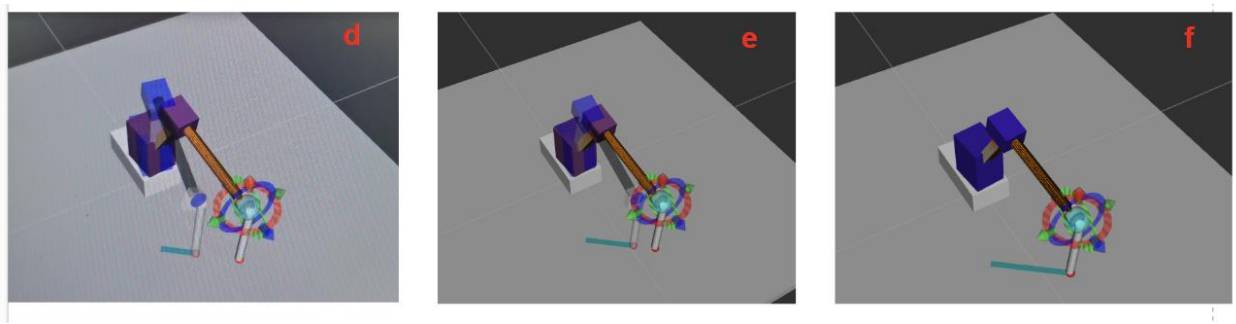
假设运动速度为v，可以通过 S 型速度曲线[29]来获取从P_1执行到P_2的最优运动时间T，那么就可以计算出末端执行器在任意时刻$t(0 < t < T)$的规划位置：

$$\begin{cases} P_{tx} = \frac{x_2 - x_1}{L} v_x t + P_{1x} \\ P_{ty} = \frac{y_2 - y_1}{L} v_y t + P_{1y} \\ P_{tz} = \frac{z_2 - z_1}{L} v_z t + P_{1z} \end{cases} \tag{3.8}$$

把通过上式得到的路点信息进行插补处理，再利用逆向运动学求解得到关节角序列，即将操作空间的直线运动映射到关节空间的角度序列，完成笛卡尔空间下的直线插补。利用 Rviz 的仿真如图 3.8 所示，3.8-1 为直接给定起点位置和终点位置进行规划，得到一条弯曲的轨迹，无法控制中间的书写过程；3.8-2 为经过笛卡尔空间轨迹规划后，书法机器人末端执行器能够按照所期望的直线轨迹运动，实现了对中间书法过程的控制。



(1)对给定始末位置直接规划的曲线运动轨迹



(2)经过笛卡尔空间规划之后的直线运动轨迹

图 3.8 笛卡尔空间轨迹规划效果示意图

3.4 目标位姿获取方案

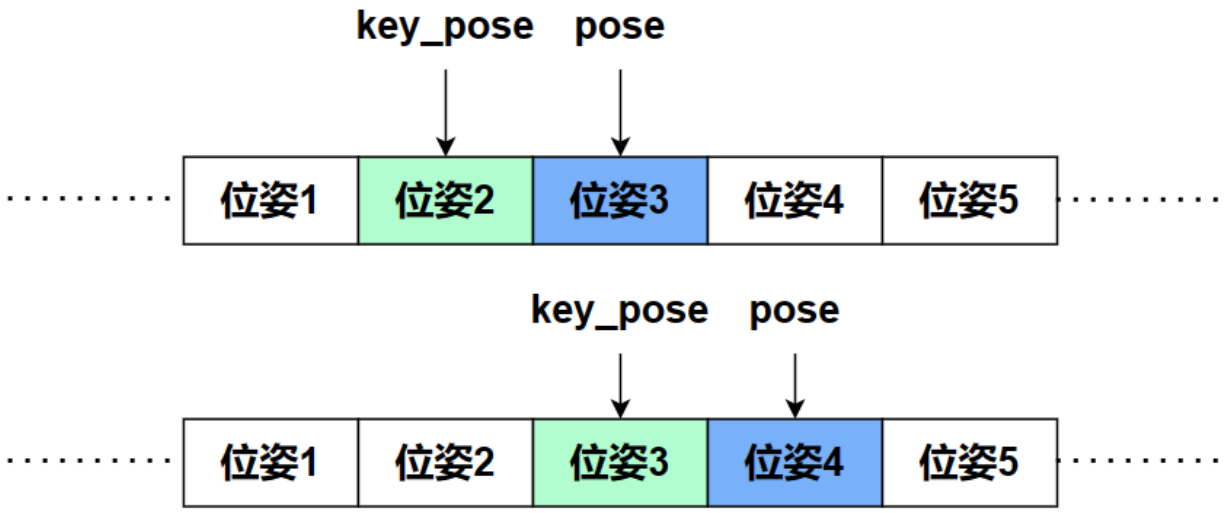
本课题将获取的手控器末端位姿当作书法机器人执行器的目标位姿，但书写者在操作过程中难免会有自然的抖动，而且由于工频干扰等原因，位姿数据本身也会有一定的波动，因此需要对从手控器所获取的数据进行滤波处理。

为降低系统对微小抖动的敏感性，本课题设计了一种滤波方法，类似于滑动窗的滤波。如下图 3.9 所示，将当前时刻从手控器获取的三维位置作为当前位姿（pose），并将上一个三维位置信息作为参考位姿（key_pose）。然后，通过式(3.7)计算这两个位置之间的欧拉距离。如果该距离较大，即判断手控器末端产生明显的位移，则将整个位姿信息发送给后续模块进行处理。如果该距离较小，那么继续接收位置信息并判断，直到距离超过设定的阈

值。

上述滤波方法中，需要设定一个合适的阈值来判断位姿数据的波动情况。在本文中，通过实验确定上述阈值为 3mm，在此阈值下可以使系统对于微小抖动不敏感，同时保持对明显位移的敏感性。

图 3.9 对获取位姿的滑动滤波判断



本章旨在实现系统整体功能的拆分，并对每个模块进行方案设计。这些方案是本文的核心内容，为进一步的书法机器人控制实验奠定基础。下一章将详细介绍实验平台的搭建过程，并展示系统在实际书写任务中的性能和表现。

第四章 书法机器人控制实验

本章综合前述的原理和系统设计，开展书法机器人的控制实验。首先介绍实验平台的搭建情况。接下来，分别介绍三个实验：第一个实验通过上位机控制书法机器人，第二个实验使用手控器控制上位机中的仿真书法机器人，以此逐步建立控制系统框架。最后，实现了书写者借助上位机通信，通过手控器控制书法机器人进行书写。

4.1 实验平台的搭建

4.1.1 基于 ROS 系统的主控制端

基于 ROS 系统的主控制端在书法机器人系统中起着关键的控制和管理作用，在第三章介绍 Moveit 库函数的基础上，为更加直观地展示轨迹规划效果，本文使用 Rviz 仿真平台对书法机器人的三维模型进行可视化处理，并以此为基础搭建主控制端平台。本课题的实验系统环境为 Ubuntu 20.04，并利用 Rviz 来进行三维可视化仿真[30]，其可以借助导入的 URDF(Unified Robot Description Format)文件来实现对模型的图形化显示。

在 ROS 系统中，URDF 是一种基于 XML 格式的文件，用于描述机器人的结构和外观，为机器人模型的定义提供了一种标准化方式。对于本文的书法机器人，它采用串联 6R 结构，即将连杆和六个关节依次连接起来形成一个运动链，每个关节承担着特定的功能。此外，7Bot 机器人采用平行四边形结构，可以在关节 3 不转动的情况下，通过转动关节 2 使机器人的前臂保持平行移动，如图 4.1 所示。除了使用式(2.4)进行修正外，还可以通过在建模时增加一个虚拟关节来表示该平行四边形结构的限位。

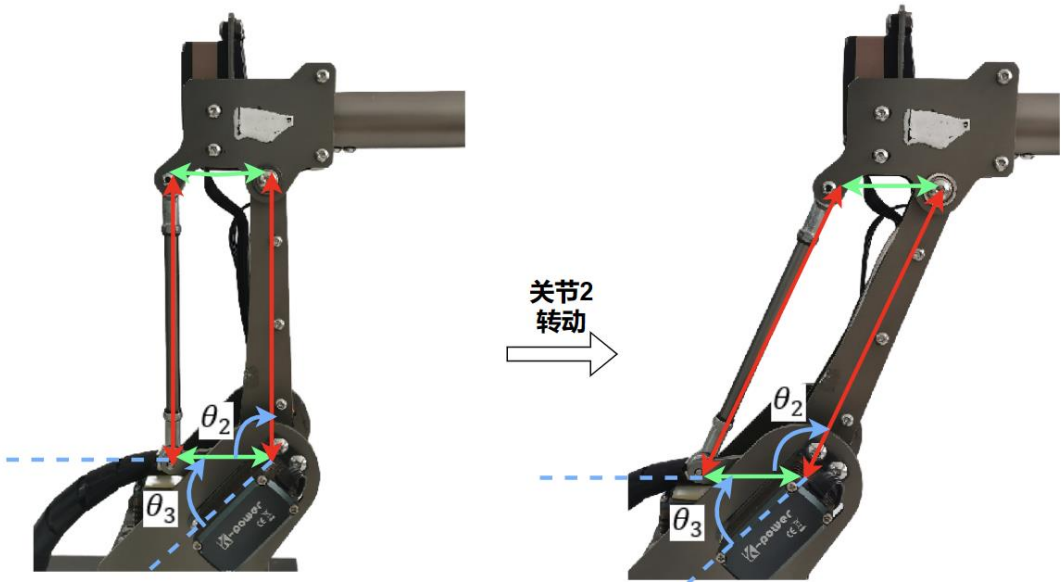


图 4.1 机器人中结构限位示意图

基于以上原理，可以将表 2.2 中的书法机器人 D-H 连杆参数转换为 URDF 文件，并编写一个 ROS 节点来订阅从舵机编码器返回的角度值作为反馈。在 Rviz 中实现书法机器人模型的位置和姿态可视化，并且在其中集成 Moveit 逆解算相关的可视化插件，这样可以方便地观察和调整机器人的轨迹规划结果。如下图 4.2 所示，是整个主控制端的展示界面。

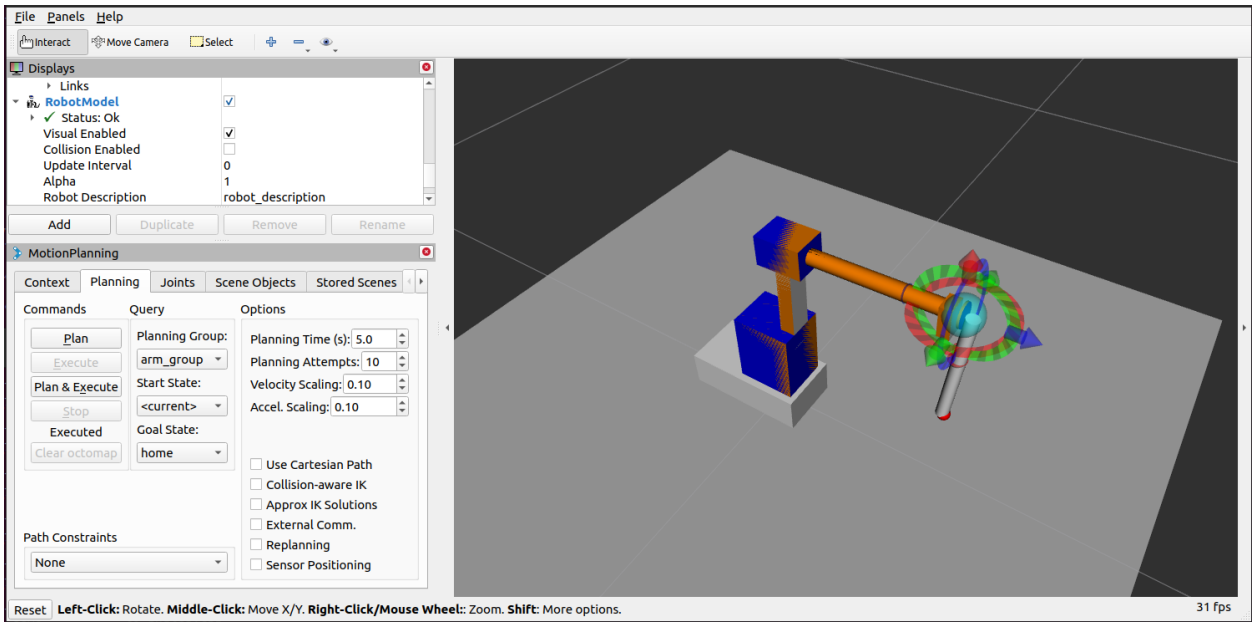


图 4.2 主控制端可视化操作界面

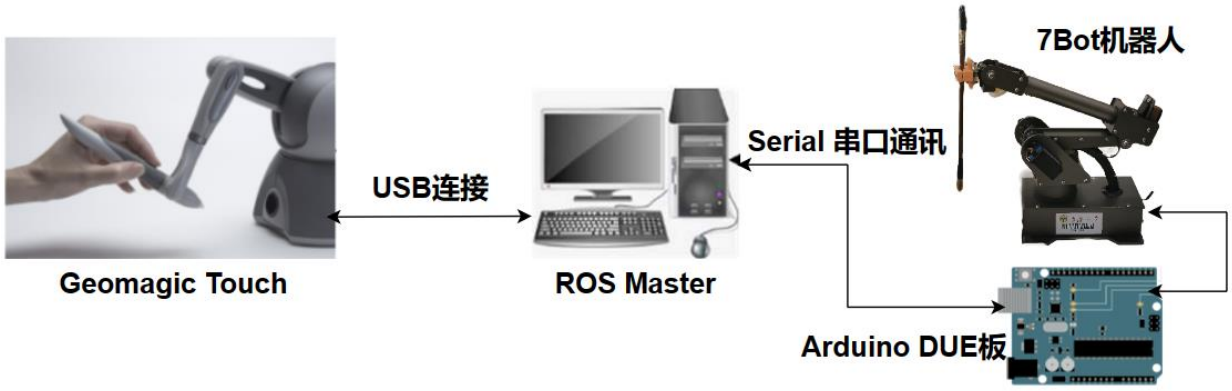
4.1.2 书法机器人硬件系统

本文使用的硬件设备包括 7Bot 六自由度轻型机器人和 Geomagic Touch 力反馈手控器两部分。其中 7Bot 的几何参数已经在第二章中介绍，该机器人的六个伺服电机可以实现在 Arduino DUE 板上的 PID 控制，其中 Arduino 作为控制器用于接收角度指令并控制机械臂运动。同时，通过 USB 接口与上位机连接以进行上下位机之间的串口通信，各个关节的角度指令可通过该串行接口发送，实现以 MATLAB 或 ROS 为上位机，Arduino 为下位机的实物操作平台搭建。

Geomagic Touch 力反馈手控器具有三个力反馈自由度和六个位置跟踪自由度，可以提供末端位姿信息，并给操作者以力反馈触觉。其工作空间角度为正负 40°，定位重复精度为 0.055 mm。本文选用的是 USB 串口通讯版本，通过与 7Bot 机器人配合使用，实现对书法机器人的控制。操作者通过手控器在虚拟空间中书写汉字，同时感受到笔尖与纸张的接触。手控器的位置和角度信息会实时通过主控制器传递给机械臂，让机械臂在真实的纸张上复现该书写动作。

如图 4.3 所示为本文书法机器人实验平台的整体结构，采用**异步串行通讯**协议，实现了主控制端和硬件系统之间的集成。

图 4.4 实验平台整体结构



4.2 上位机对书法机器人控制的建立

本节将详细介绍系统主控端对书法机器人控制的建立过程，包括 ROS 节点的建立、轨迹信息传输流程和控制策略等。其中，ROS 节点是系统的基本组件，负责实现不同功能模块之间的通信和协调，例如轨迹规划、逆向运动学解算和状态更新等。最后展示通过上位机控制 7Bot 书法机器人书写汉字“十”的过程。

下图 4.5 所示，是本阶段实验的核心节点和话题。其中，/move_group 节点是整个系统主控端的核心，负责对轨迹进行笛卡尔空间规划以及逆向运动学解算，并在 Rviz 中进行轨迹规划和运动状态的可视化处理。

另外，/action_server 节点负责接收规划后的关节空间轨迹，并进行样条插补处理，然后将路点信息发送给/Arduino_port 节点。/Arduino_port 节点负责将收到的数组信息转化为字符串，并通过串口发送给下位机。同时，/Arduino_port 节点实时获取书法机器人的当前关节状态，并通过/action_server 节点将状态发布到/real_joint_states 话题上，由/joint_state_publisher 节点进行转发，以便在 Rviz 的可视化界面中更新关节角度。

而/robot_state_publisher 节点主要用于接收当前的关节角度信息，并通过/tf 话题发布结果到核心的/move_group 节点，以实现机器人状态的更新。在这里，tf（Transform）是一种用于描述坐标系之间关系的库。例如图 2.3 所建立的六个坐标系，可以通过 tf 来很方便地描述出机器人各个关节所在的位置状态信息。

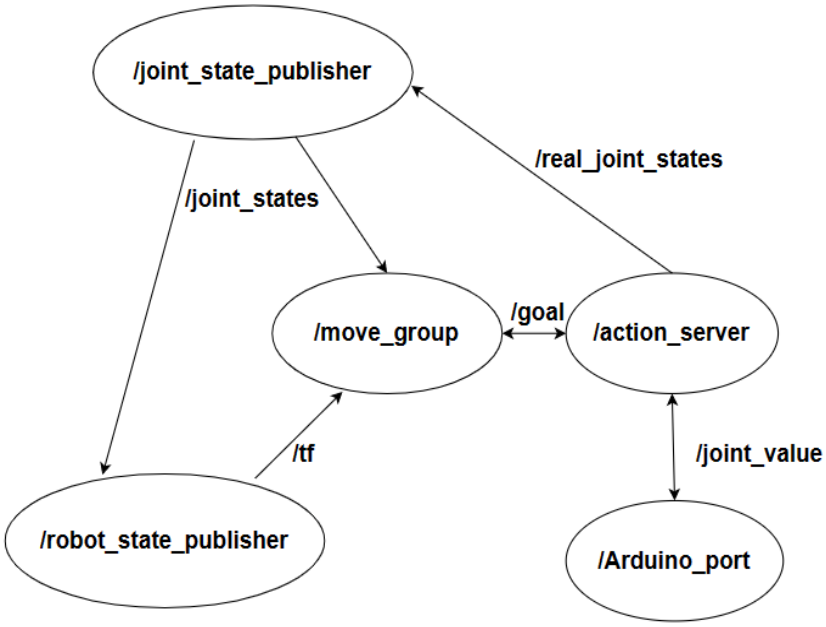


图 4.5 上位机控制书法机器人实验的核心节点

通过 Moveit 提供的 API 接口，在主控端中编写 Python 代码让书法机器人按照规定的轨迹书写汉字“十”，下图 4.6 为书写运动的流程图，其中 4.6-a 到 4.6-c 是书法机器人书写笔画“横”，4.6-d 到 4.6-f 是书法机器人进行笔画“竖”的书写。实验证明了前述运动轨迹规划算法的有效性，以及书法机器人书写过程的准确性和稳定性，为后续的进一步实验打下基础。

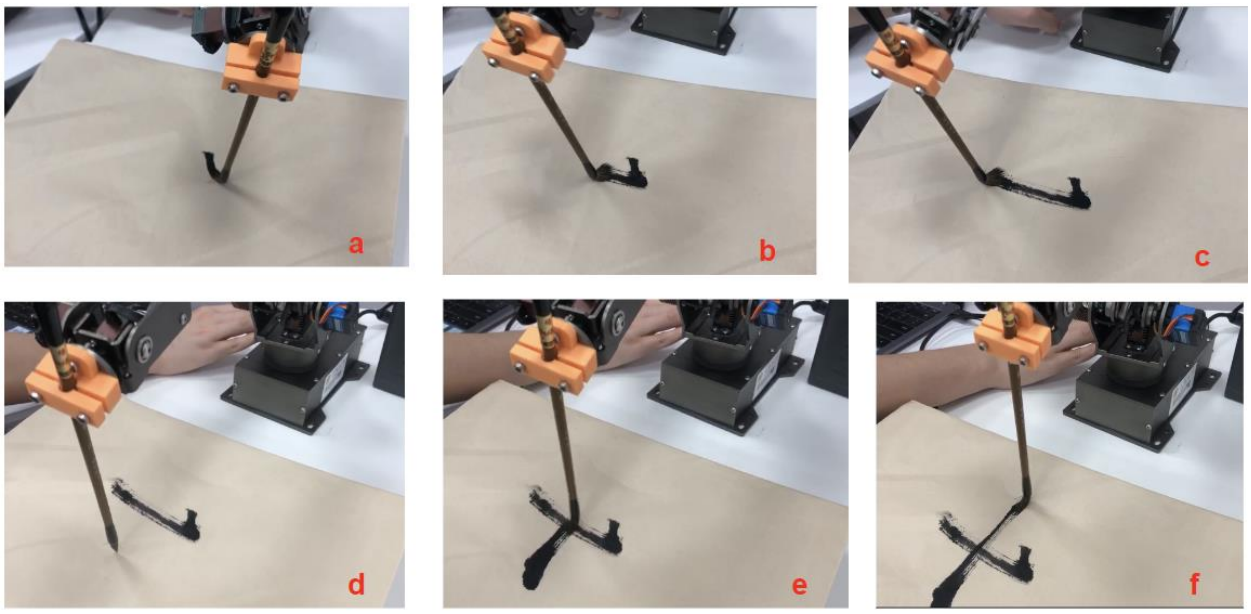


图 4.6 机器人书写汉字“十”测试

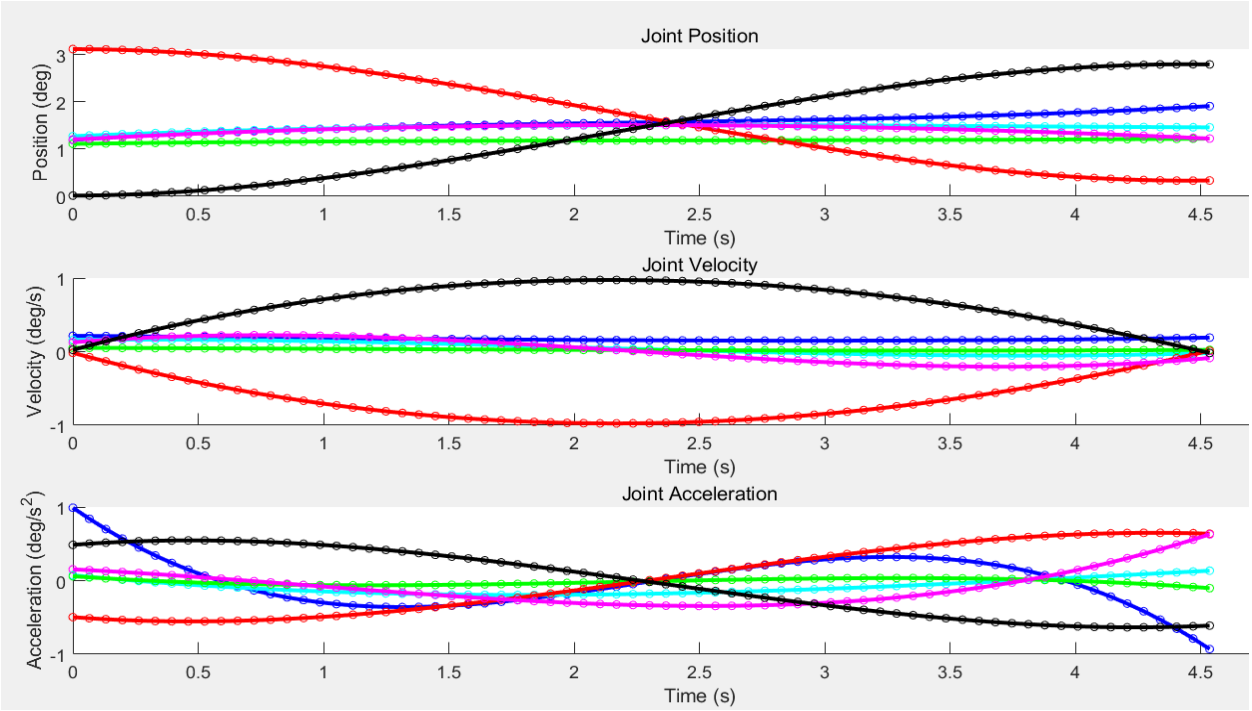


图 4.7 书写笔画“横”的关节运动曲线图

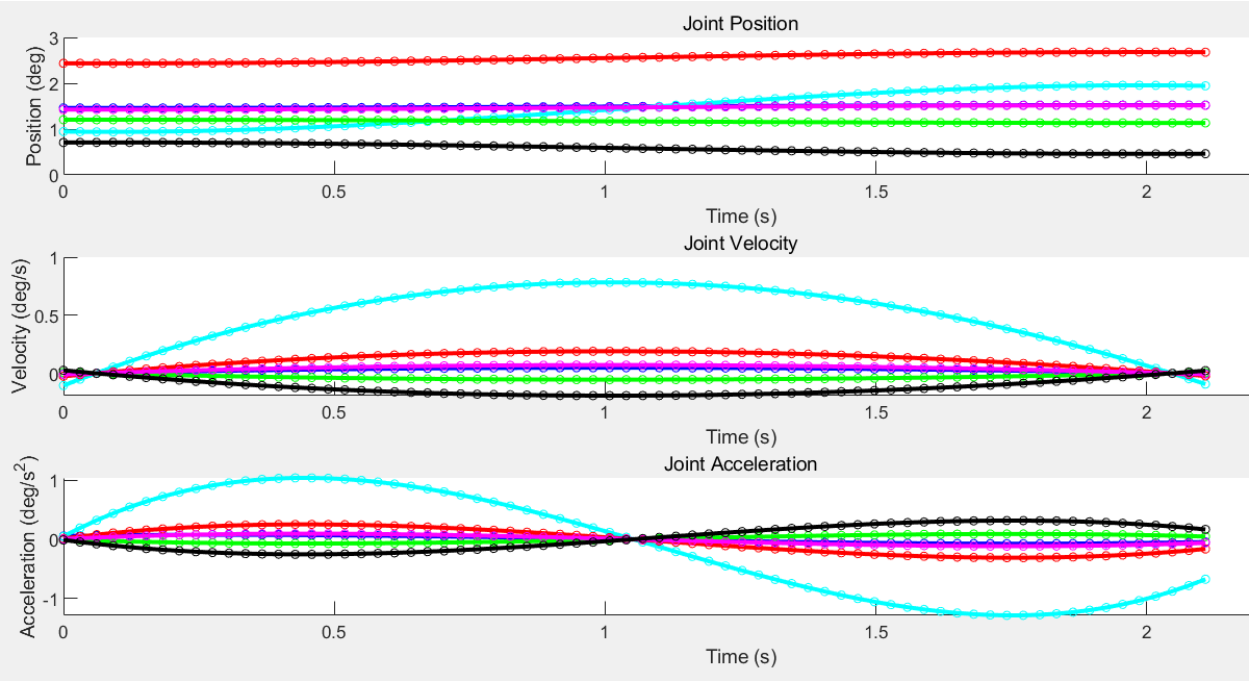


图 4.8 书写笔画“竖”的关节运动曲线图

4.3 基于力反馈手控器的控制实验

通过上一节的实验，成功建立了主控端和书法机器人之间的控制。本节将详细介绍力反馈手控器和主控端之间控制的建立过程，包括关键 ROS 节点的建立、感知操作者手部运动信息并转化为控制指令和力反馈机制的实现等[31]。最后展示通过手控器操作仿真书法机器人运动的实验过程。

下图 4.9 所示，是本阶段实验的核心节点与话题。仍然需要使用核心节点/move_group 来进行逆运动学解算和轨迹规划，同时负责在 Rviz 中可视化规划后的轨迹，并将模拟力信息反馈给手控器。

另外，新增一个名为/touch_state 的节点，它负责通过话题/pose 发送手控器的末端姿态信息，并通过话题/button 发送手控器的按钮信息。对于获取的位置信息，需要进行一步滤波以去除微小抖动，以判断是否发生明显位移。除去位置信息，获取按钮信息的目的是为处理可能出现的逆运动学解算超时导致的"丢步"问题。当书法机器人的姿态与手控器操作时期望的姿态不一致，可以通过按钮触发使机器人回到初始点，从而实现书法机器人位姿的复位。

话题/force_feedback 是实现力反馈机制的关键，可以通过力传感器获取执行器末端的力信息，或者将毛笔在书写方向上的位移乘以一定比例系数当力信息进行反馈，并结合阻抗或导纳控制[32]，使书写过程中的力反馈更加平滑，增强书法机器人的稳定性。

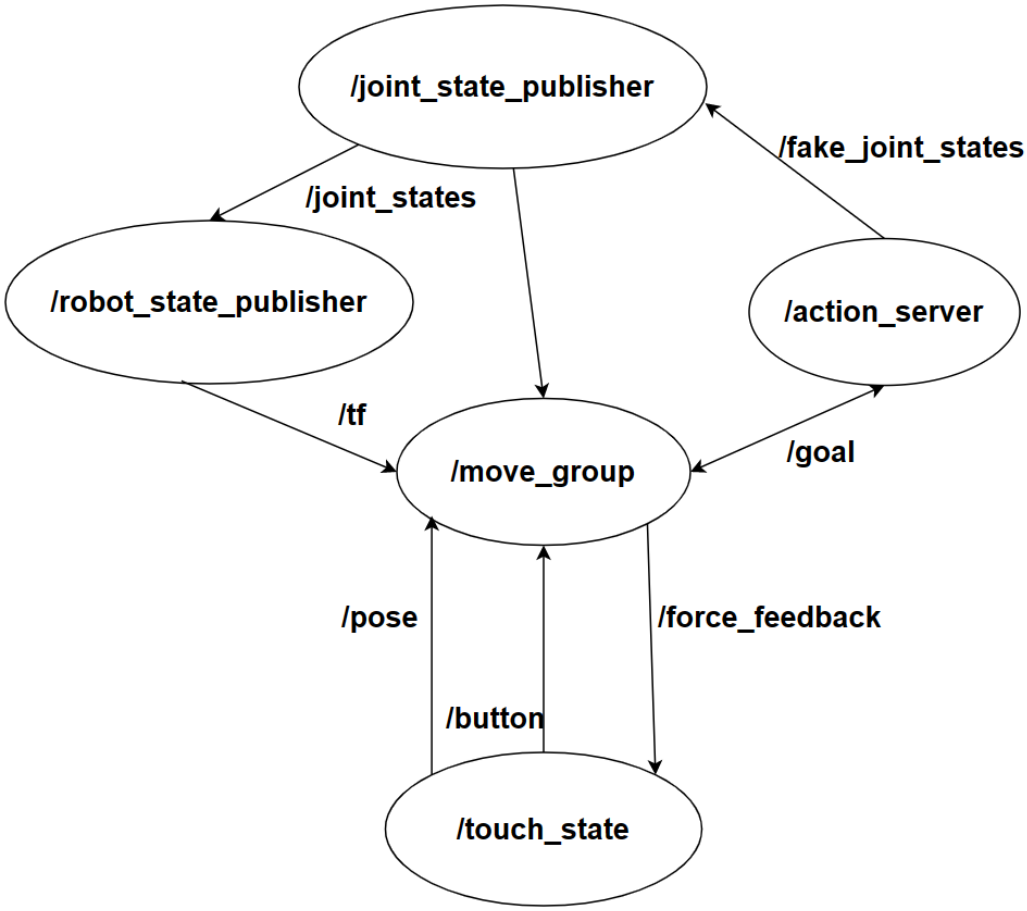


图 4.9 手控器控制仿真机器人实验的核心节点

按照上述原理进行手控器与仿真机器人的测试，如下图 4.10 所示。其中图 4.10-a 是手控器与仿真书法机器人的初始位姿，图 4.10-b 是手控器沿 x 轴正方向移动，图 4.10-c 是手控器沿着 x 轴负方向移动，图 4.10-d 是手控器沿着 z 轴负方向移动，可以看到 Rviz 中仿真机器人都实现了相对应的移动跟随。证明手控器与主控端之间控制的成功建立。

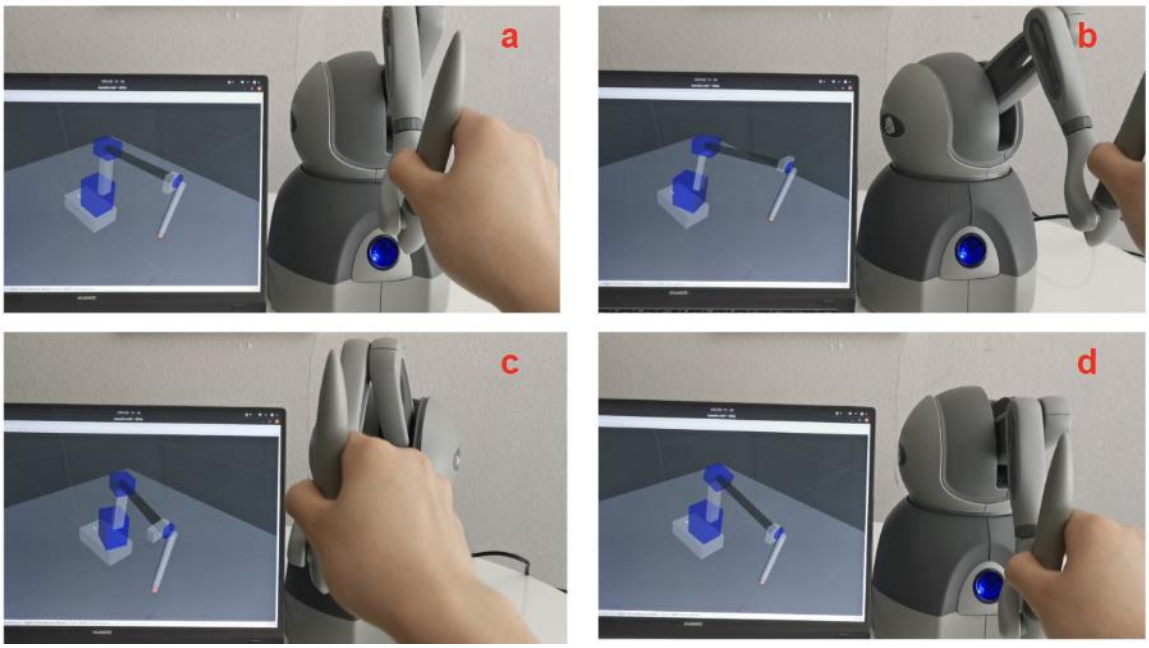


图 4.10 手控器操作仿真机器人流程图

4.4 书法机器人综合控制实验

通过以上两个实验，分别成功建立主控端与书法机器人、力反馈手控器之间的控制联系。本节将在此基础上实现书写者借助主控制端通信，通过手控器直接控制书法机器人进行书写，并且实现力信息的反馈。

下图 4.11 为书写者通过手控器操作书法机器人的流程图。首先，通过监听位姿信息的话题获取当前手控器末端的位姿。接着，对获取到的位姿进行滤波处理，通过比较当前位置与上一个关键位置之间的欧拉距离判断是否发生一次明显的位移。然后，如果达到了阈值标准，则进行笛卡尔空间规划、逆向运动学解算和轨迹规划操作。在这一过程中，还需要判断在设定的时长内是否成功完成逆向运动学求解，因为有时可能会出现在规定时间内无解的情况。最后，如果求解及规划成功，再将得到的关节角度路点信息通过串口发送给 Arduino，从而驱动舵机执行一步笔画的书写。通过重复以上流程，可以实现完整汉字的书写。

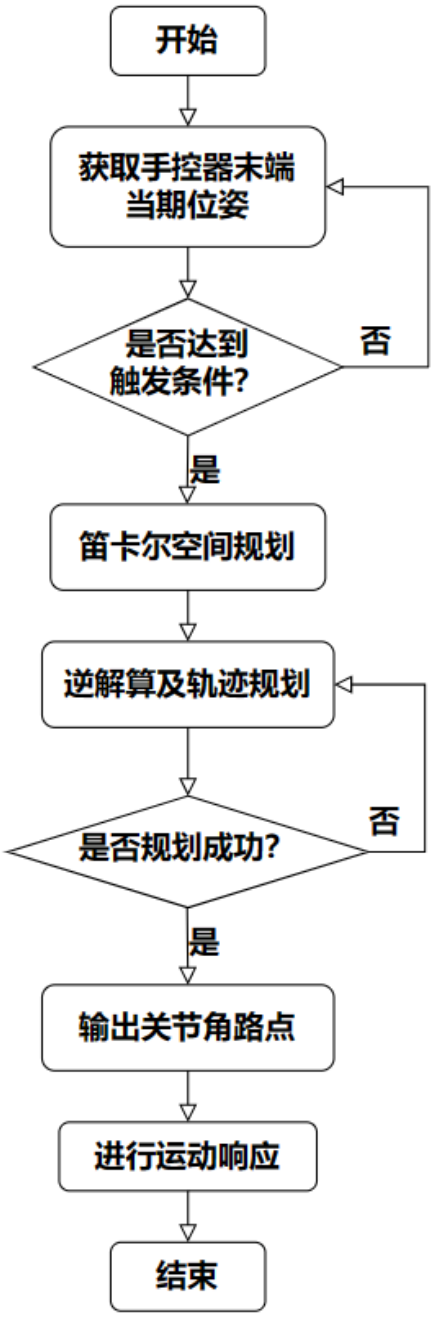


图 4.11 书法机器人书写流程图

在进行逆向运动学解算和轨迹规划时，需要特别注意处理可能出现的无解情况。这种情况可能导致书法机器人出现失“丢步”现象，进而导致手控器和书法机器人之间的状态连接断开。为应对这种情况，可以采取以下措施：当发现书法机器人的实际位姿与理想位姿存在较大误差时，可以通过前文提及的按键功能，使书法机器人的位姿回到初始设定点，重新建立两者之间的控制联系。这样可以确保书法机器人在书写过程中始终保持准确的位姿和轨迹规划，提高书写的稳定性和精确度。

在前述实验的基础上，可以构建出书法机器人综合控制实验的核心节点，如下图 4.12 所示。其中，节点/touch_state 代表力反馈手控器，节点/move_group 和/action_server 代表系统主控端，节点/Arduino_port 代表执行端的书法机器人。这些节点实现从手控器位姿(/pose)到规划工作空间路径(/goal)，再到插补之后的关节角路点(/joint_value)的数据流动过程。同时，通过话题/force_feedback 实现对书写过程中力觉信息的反馈。

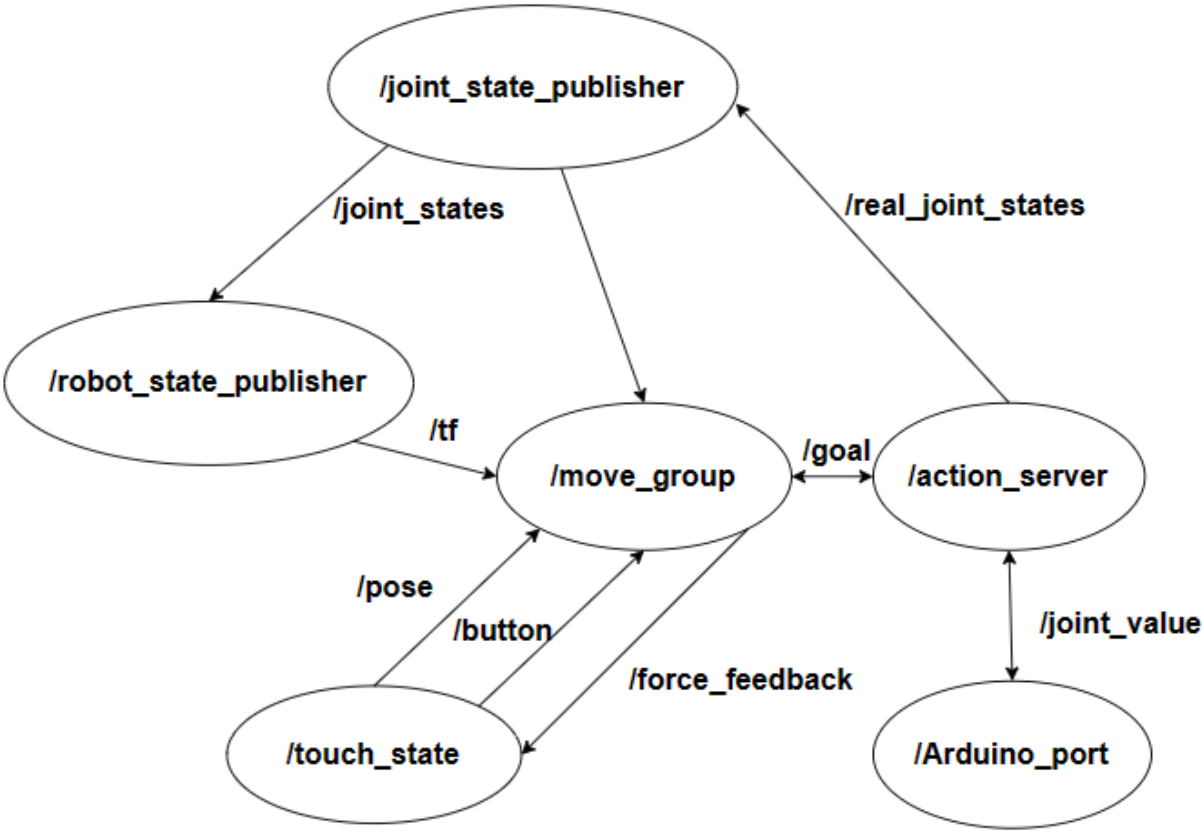


图 4.12 系统整体核心节点

书法机器人汉字“二”的书写如下图 4.13 所示。其中，图 4.13-a 是第一个笔画的起始状态，通过操作手控器末端朝右下方移动，使书法机器人将毛笔与纸张接触，并移动到相应的落笔位置；图 4.13-b 展示第一个笔画的结束，通过操作手控器末端向左移动一段较长的距离，使毛笔在纸面上留下较长的横笔画；图 4.13-c 是第二个笔画的开始，首先，操作手控器向上运动并向右下方移动，使毛笔完成抬笔，并移动到相应的落笔位置；图 4.13-d 展示第二个笔画的结束，与第一笔类似，操作手控器向左移动一段较短的距离，在纸面上留下较短的横笔画。通过以上步骤，完成了书法机器人汉字"二"的书写。

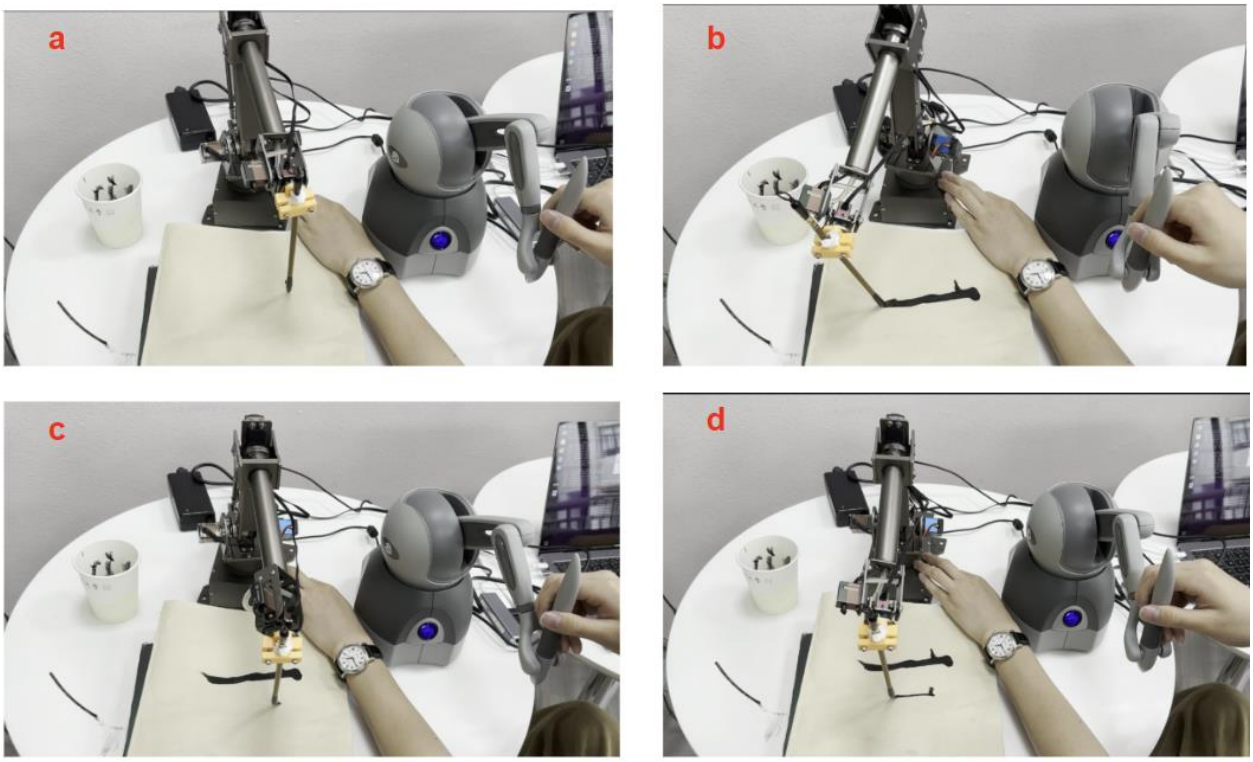


图 4.13 书法机器人汉字“二”的书写

书法机器人数字“1”的书写如下图 4.14 所示。图 4.14-a 是书写到指定位置的落笔，图 4.14-b 展示通过操纵手控器沿着 y 轴移动，实现书法机器人在纸面上的跟随书写。

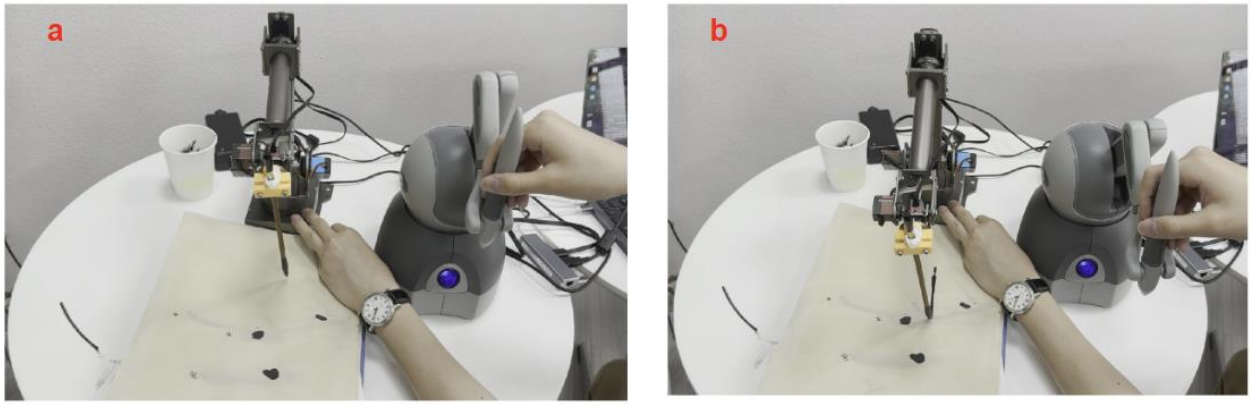


图 4.14 书法机器人数字“1”的书写