

重庆大学本科学生毕业设计（论文）

# 多自由度机器人伺服控制软件设计



学 生：柏啸天

学 号：20124735

指导教师：凌睿

专 业：自动化

重庆大学自动化学院

二〇一六年六月

**Graduation Design(Thesis) of Chongqing University**

**Design of Servo-control Application of  
Multi-DOF Robots**



**Undergraduate: Bai Xiaotian**

**Supervisor: Prof. Ling Rui**

**Major: Automation**

**School of Automation**

**Chongqing University**

**June 2016**

## 摘 要

本文展示了一套多自由度机器人伺服控制软件的设计过程。

设计选择了典型的六自由度机械手臂作为控制对象，采用单片机作为下位机，结合其硬件结构和参数，以 PC 为平台设计了一套基于 C++ 语言的伺服控制软件。该软件具有计算、仿真和控制功能。

在计算方面，软件根据机械手臂结构，建立连杆坐标系，通过一系列的矩阵运算来得到机械手臂的运动学正解和逆解。在利用运动学算法设计得到的软件界面上，使用者可以通过输入每个关节角度来计算得到机械手臂末端执行器的位姿，也可以通过输入末端执行器的位姿来推算得到可能的关节角度。

另外，软件还采用插值法进行轨迹规划，在已知起始坐标和终止坐标的条件下，在轨迹中插值，并求解插入的每个点的坐标，同时求运动学逆解，来得到每个点上各个关节的角度。

在仿真方面，软件采用了 OpenGL 图形程序接口，绘制了与实验所采用的机械手臂形状一致的三维机构模型，同时可将各个关节角度的变化加以展示。无论是求解运动学正解、逆解，还是求解运行轨迹，软件都会提供直观的模型仿真图像。

在控制方面，采用基于 STC12C5A60S2 单片机的舵机控制器对机械手臂各个关节的数字舵机进行控制。设计中首先编写下位机程序并烧录至单片机，明确需要通过上位机传送的命令格式，然后，设计上位机软件，通过串口来发送命令至单片机，控制舵机转角以及转动时间。软件不仅可以控制单个舵机转动，也可以对所有的 6 个舵机进行联合控制。软件还可以读取在仿真模块得到的轨迹解，控制机械手臂末端执行器通过计算得到的轨迹移动到预定的坐标。

**关键词：**六自由度，C++语言，运动学，逆运动学，轨迹规划

## ABSTRACT

This paper presents the design process of a servo-control software of multi-DOF robots.

This design chooses a 6-DOF robotic arm as control object and use a single-chip microcomputer system as lower computer system. According to the hardware structure and parameters, a PC-based servo-control software is designed in C++. The main functions of the software are calculation, simulation and control.

In terms of calculation, the software establishes connecting rod coordinate systems and calculates kinematics and inverse kinematics through matrix operations. On the interface of the software, the user can get the pose of the end-effector by inputting the angles of the 6 joints, or get the solutions of the angles by inputting the pose of the end-effector.

The software also uses interpolation methods for trajectory planning. Given the coordinates of the starting point and the destination, the software interpolates several points into the trajectory and calculate inverse kinematics for every point.

As for simulation, the software draws a 3D model similar to the robotic arm using OpenGL. The 3D model is able to presents the consequent joint angles in kinematics, inverse kinematics and trajectory planning.

In the aspect of control, the design uses a servo-controller based on a STC12C5A60S2 single-chip computer to control the digital servos in the joints of the robotic arm. First, a program for the lower computer system is written and burned onto the single-chip computer, the program defines the format of controlling commands. Then, a upper-computer software is written to send commands to the controller via a serial port to control the angle and rotating time of the servos. The software is not only able to control a single servo, but also able to control all 6 servos simultaneously. In addition, the software can read the results of trajectory planning in calculation module and control the end-effector move to the scheduled pose via the specific trajectory.

**Key words:** 6-DOF, C++ Programming Language, Kinematics, Inverse Kinematics, Trajectory Planning

# 目 录

中文摘要.....	I
ABSTRACT .....	II
1 绪论 .....	1
1.1 课题目的 .....	1
1.2 国内外发展现状 .....	1
2 六自由度机械手臂的嵌入式系统设计 .....	3
2.1 嵌入式系统设计概述 .....	3
2.2 机械手臂驱动系统的设计 .....	3
2.3 舵机控制器的设计 .....	5
2.3.1 微控制器的选择 .....	5
2.3.2 各模块电路设计 .....	6
3 机械手臂的运动学分析 .....	8
3.1 运动学分析基础概念 .....	8
3.1.1 刚体的位姿描述 .....	8
3.1.2 坐标变换.....	9
3.2 机械手臂的运动学求解 .....	11
3.2.1 连杆参数和连杆坐标系.....	11
3.2.2 求运动学正解.....	14
3.2.3 求运动学逆解.....	16
4 机械手臂的运动控制 .....	20
4.1 轨迹规划 .....	20
4.1.1 转角插值法 .....	20

4.1.2 直线插补法 .....	21
4.2 轨迹执行 .....	21
5 计算与仿真模块设计 .....	23
5.1 计算与仿真模块概述 .....	23
5.1.1 模块功能设计 .....	23
5.1.2 编程语言和开发工具的选择 .....	24
5.2 计算模块的程序实现 .....	24
5.2.1 求运动学正解 .....	24
5.2.2 求运动学逆解 .....	25
5.2.3 轨迹求解 .....	26
5.3 仿真模块的程序实现 .....	27
5.3.1 OpenGL 窗口的建立 .....	28
5.3.2 机械手臂模型的绘制和仿真实现 .....	29
5.4 计算与仿真模块的整体界面 .....	31
6 控制模块设计 .....	33
6.1 控制模块概述 .....	33
6.1.1 控制流程设计 .....	33
6.1.2 开发工具的选择 .....	34
6.2 下位机的程序实现 .....	34
6.2.1 系统的初始化 .....	34
6.2.2 控制命令的处理 .....	35
6.3 PC 上位机的程序实现 .....	36
6.3.1 与下位机的通信 .....	36
6.3.2 舵机控制面板 .....	37
6.3.3 动作组操作区 .....	38
6.3.4 控制模块上位机软件整体界面 .....	39

7 总结 .....	41
参 考 文 献 .....	42
附录 A：部分主要函数代码 .....	43

## 1 绪论

### 1.1 课题目的

机器人技术作为一门交叉学科，综合了动力学、控制学等多个学科，机器人技术的水平可以体现出一个国家综合科研实力的强弱<sup>[1]</sup>。机器人技术自上世纪六十年代问世以来，逐渐成熟，机器人产业也愈发发展壮大，机器人已经在许多领域承担起重要的作用。

机器人一般是具有多个关节从而具有多自由度的机械装置，它能依靠自身的控制和动力来执行不同工作，实现各种功能。机器人具有不同的控制方式，可以直接接受人类的命令，也可以按照预设程序工作，更加先进的机器人还可以利用人工智能实现更强大而多样化的功能。

机械手臂是一种典型的机器人，它常被用于焊接、注塑等工业制造任务，也可用于拆除炸弹等特殊用途。机械手臂具有类似人的臂和手的结构，能完成与臂和手相似的功能，可以通过程序或者人工控制来完成抓取、搬运等工作。当一个机械手臂具有六个自由度时，它的末端执行器就可以利用其中的三个自由度达到任意姿态，再利用剩下的三个自由度达到任意位置。因此，六自由度的机械手臂是一个经典的且具有一般性的机器人选型。本课题的研究，就以六自由度机械手臂为对象展开。

为了精确地控制机械手臂，需要将机械手臂末端执行器的位姿转化为各个关节转动的角度，将三维的运动数字化。因此，首先我们需要一个算法来进行各个关节角度的计算，这样，我们可以得到若干组关于各个关节角度的解。其次，我们需要利用轨迹规划来对机械手臂的运动轨迹进行插值运算，对机械手臂进行连续的关节角度的控制，使得末端执行器达到预期的位姿。

本课题研究的主要目的就是利用运动学分析和轨迹规划，求解得到机械手臂末端的运动路径所对应的各个关节的角度变化。同时，设计控制软件，利用计算得到的角度值来控制各个关节转动，使其能够控制六自由度机械手臂运动完成指定的动作。

### 1.2 国内外发展现状



在一些发达国家，机器人技术研究起步较早，发展成熟，机器人在工业、农业、军事等各个产业得到了广泛的应用。

近年来，国外的机器人技术研究主要有以下几个方面：

（1）机械结构：除了主流的串联型关节机器人，还开发出了可以适应复杂工况的垂直关节机器人以及适用于大型工程的采用协作结构的超大型机器人。

（2）控制技术：采用先进的 32 位 CPU 进行机器人控制，可以操控更多的关节；一些新型的控制技术如多机器人协调控制技术等也在推广应用。

（3）驱动技术：交流伺服驱动为工业机器人主流的驱动方式，还有一些驱动方式如直接驱动(DD)则应用于一些特殊的机器人种类；新一代的基于微处理器的智能驱动已被开发应用；分布式智能驱动等更加先进的技术成为了研究的热点。

相比日本、德国、美国等机器人大国，我国的机器人产业起步较晚，研究工作始于 20 世纪 70 年代。相比发达国家，我国产业工人人均配备的机器人数量也相对较低。但是，经历几十年的发展，我国的机器人产业经历了从无到有，从少到多的转变。

目前，中国已掌握了机器人的部分生产技术，而在此之前，已经有很多企业通过引进国外的技术，先行进行了机器人及其自动化生产线的生产、设计、制造等工作<sup>[2]</sup>。

在国家有关计划项目的支持带动下，我国的机器人技术也取得了长足进步。这些进步体现在如下几个方面：

（1）机器人基础技术：我国在机器人运动学分析、控制算法等基础技术上的研究不断进步，并将基础技术的研究成果应用于实际工作中。

（2）机器人基础元部件：我国目前已开发出大量机器人的元部件产品例如同步电机、测速发电机、谐波减速器、薄壁轴承等，但一些部件仍处于样品阶段，还未能投入批量生产。

（3）机器人控制系统：我国已成功研发具有多个 CPU、采用分级控制的机器人控制装置。

（4）机器人应用工程：在汽车等工业，我国已建立了数条装配、焊接生产线，并且建立了数个产业基地。

## 2 六自由度机械手臂的嵌入式系统设计

### 2.1 嵌入式系统设计概述

在设计机械手臂的嵌入式系统之前，首先要明确系统需要完成什么样的任务。在本次设计中，系统的工作过程为：上位机得到需要控制机械手臂末端到达的位置，运行相应的算法，得到机械手臂各个关节的角度值，转化为控制指令，通过串口将指令下达给控制器，最终由控制器完成对机械手臂各个关节角度的控制，从而使得机械手臂末端到达预期位置。

机械手臂的嵌入式系统可以分为两部分：机械手臂主体和驱动控制器。机械手臂主体的设计主要是驱动系统的选择，其功能是为机械手臂关节转动提供驱动力和精确的转角控制；驱动控制器的设计则包含了基于微控制器最小系统的一系列电路设计，其功能是与上位机进行通信，接收控制指令，并对驱动系统进行控制。

### 2.2 机械手臂驱动系统的设计

机械手臂的驱动系统可以采用液压、气压或电机驱动。

液压驱动提供的推力大，可应用于极端恶劣环境，但是存在泄露、低速时不稳定等问题，而且功率单元非常笨重和昂贵，现在一般应用于巨大型机器人或者一些特殊应用场合；气压驱动在原理上和液压驱动相同，结构简单，价格低廉，但是由于空气的可压缩性，难以控制气缸的精确位置，因此通常只用于插入操作或者 1/2 自由度关；电机驱动精确度高，控制方便，在低推力需求下成本较低。考虑设计需求，选择电机驱动作为机械手臂的驱动方式。

驱动电机的选择主要有直流电机、步进电机和伺服电机等。直流电机价格便宜，控制方式也十分简单，但是转角精度非常低，因此不作考虑；步进电机因其特性，可以方便的控制转速和转角，但是由于步进电机采用开环控制，在加减速过程中容易产生丢步和过冲现象，无法对角度进行精确控制；伺服电机包含了传感器和控制器，采用闭环控制，转角控制精度高，输出力矩大。考虑到机械手臂控制需要较高的转角控制精度，因而选择伺服电机作为驱动电机。

考虑到控制的简便性，综合市场上不同伺服电机的参数，决定采用舵机作为驱动电机。舵机作为伺服电机的一种，角度控制精确，输出的力矩也比较大，具有较好的防抖动和保持姿态的能力。舵机又分为数字舵机和模拟舵机，数字舵机相比模拟舵机增加了微处理器和晶振，对信号的响应速度更快，角度控制和力矩输出也更精确。

表 2.1 对常见的数字舵机 DS3115MG 和模拟舵机 MG996R 进行了比较：

舵机参数比较表

表 2.1

项目名称	DS3115MG 数字舵机	MG996R 模拟舵机
工作电压	5-7.4V	4.8-7.2V
工作电流	40-100mA	40-100mA
工作扭矩	13kg•cm	10kg•cm
极限角度	270°	180°
无负载转速	0.13sec/60°(6V)	0.13sec/60°(6V)
齿轮材料	金属	金属
控制方式	改变 PWM 脉冲宽度	改变 PWM 脉冲宽度
角度误差	0.27°	3-5°

从上面的舵机参数表可以看出，数字舵机在工作扭矩、转角范围、和转角精度上都比模拟舵机更有优势，在其他项目上两者基本相同。因此，设计最终采用了 DS3115MG 数字舵机作为驱动电机。



图 2.1DS3115MG 数字舵机

根据 DS3115MG 数字舵机资料，当 PWM 脉冲的脉宽在  $500 \sim 2500\mu s$  可满范围变化时，舵机可控角度范围达到最大即  $270^\circ$ 。当脉宽在  $1500\mu s$  时，舵机处于中间位置。舵机的转动速度则可以通过改变占空比来控制。

## 2.3 舵机控制器的设计

选定 DS3115MG 数字舵机作为驱动之后，需要设计一个舵机控制器来对机械手臂上每一个关节的舵机进行控制，这个舵机控制器包含一个微控制器的最小系统以及一些模块如供电模块、舵机驱动模块等。

### 2.3.1 微控制器的选择

微控制器是舵机控制器的核心部分，其他模块的设计都将基于微控制器展开，因此，选择一个合适的微控制器是十分关键的。

在选择微控制器时，需要考虑以下几方面因素：

微控制器是否有完善的配套资料和开发工具；微控制器是否具有足够的集成模块、接口；微控制器是否具有足够的运算能力；微控制器是否具有高可靠性；微控制器是否具有抗静电、抗干扰的能力；微控制器的功耗和价格。

综合上述因素，设计最终采用了 STC 公司生产的 STC12C5A60S2 单片机。该单片机指令代码完全兼容传统 8051 单片机，具有运算速度快、抗干扰能力强等优点，功耗和价格也符合条件<sup>[3]</sup>。

在资源方面，该单片机含有 36 个 I/O 接口，两路 PWM 以及 4 个 16 位定时器，可以满足舵机转角和速度的控制需求。另外，该单片机还采用通用全双工异步串行口可以满足与 PC 之间通信的需求。

STC12C5A60S2 单片机相关资料丰富、全面，且常被用于教学实验，因此在设计前便积累了一些使用经验，这对设计过程有着很大的帮助。设计最终采用了 LQFP44 封装形式的 STC12C5A60S2 单片机。如图 2.2、图 2.3 所示。



图 2.2 STC12C5A60S2 单片机实物图

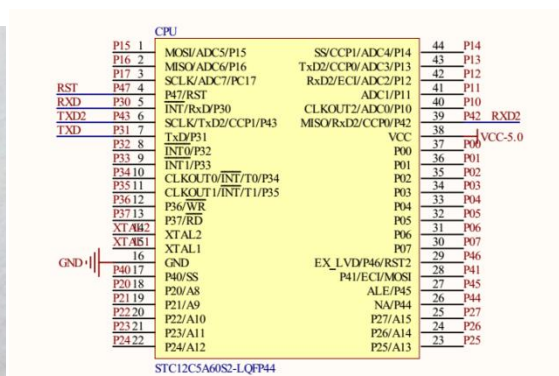


图 2.3 STC12C5A60S2 单片机原理图

### 2.3.2 各模块电路设计

为使舵机控制器能够正常工作，需要围绕单片机最小系统电路设计一些必要的功能模块电路<sup>[4]</sup>。这些电路可以分为三个部分：

#### (1) 与 PC 通信的串口电路

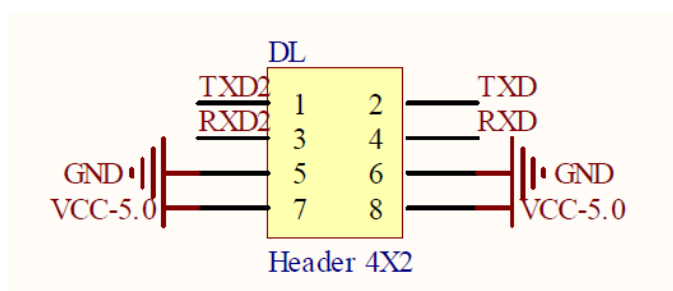


图 2.4 舵机控制器串口接口

串口电路的设计十分简单，只需将 STC12C5A60S2 单片机的串行接口引脚引出到插座即可。

#### (2) 舵机的电源接口、稳压电路和驱动电路

舵机控制器采用 7.4V 锂电池供电，可以直接供给舵机。同时，利用 AMS1117 稳压器，将电压降至 5.0V 后提供给需要 5.0V 供电的芯片，如单片机、电机驱动板等。设计中，也为 5.0V 电源预留了接口，可以采用双电源供电。

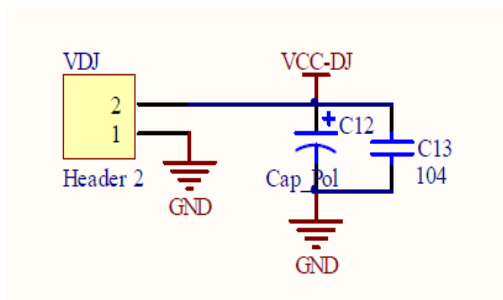


图 2.5 舵机电源接口

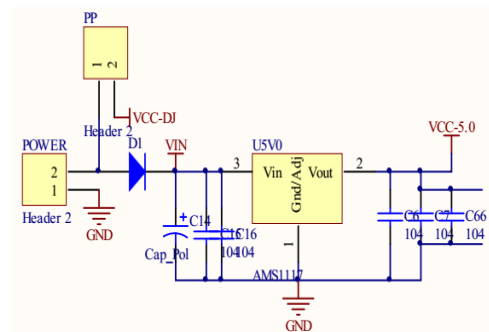


图 2.6 稳压电路

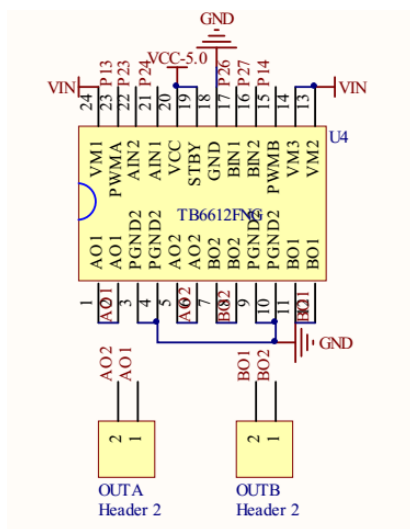


图 2.7 舵机驱动电路

### (3) 检压、报警电路

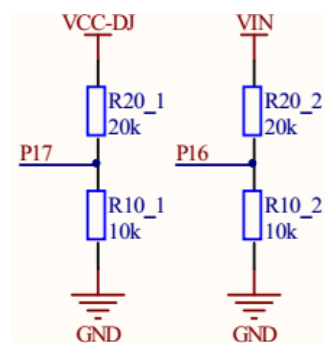
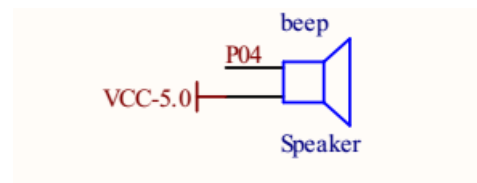


图 2.8 检压电路图



2.9 蜂鸣器电路

单片机通过 A/D 转换判断电压大小，若电压过低，则控制蜂鸣器发出警报。

### 3 机械手臂的运动学分析

#### 3.1 运动学分析基础概念

刚体参考点的位置和刚体的姿态统称为刚体的位姿，确定机器人的末端执行器在空间中的位姿所需要的独立运动参数的数目称为机器人的自由度<sup>[5]</sup>。

对于一个机器人而言，想要使得末端执行器到达任意位姿，至少需要六个独立的自由度，其中三个自由度用以控制末端的空间位置，另外三个自由度用以控制末端的姿态。机械手臂的连杆间关节具有一个自由度，本设计所采用的机械手臂，采用六个数字舵机驱动六个关节，具有六个独立自由度，满足条件。

运动学分析的目的，就是将机械手臂关节角度和末端执行器的位姿进行互相转化。运动学正解根据关节角度和连杆参数推算出末端位姿，而运动学逆解则根据末端位姿推算得到关节角度的解。在对机械手臂进行运动学分析之前，首先应该了解一些相关的基础概念。

##### 3.1.1 刚体的位姿描述

刚体的位姿描述，包括刚体参考点位置的描述和刚体姿态的描述。

（1）刚体位置的描述：对于直角坐标系{A}，其空间内任一点P的位置可以用列向量 ${}^A P$ 表示，如式（3.1）所示。

$${}^A P = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \quad (3.1)$$

其中，上角标A表示坐标系{A}， $p_x$ ， $p_y$ 和 $p_z$ 分别为点P在坐标系{A}中各个坐标轴上的分量。

（2）刚体姿态的描述：对于空间中一刚体B，为了规定其方位，在B的参考点上固接一个坐标系{B}。坐标系{B}的三个单位主矢量 $x_B$ ， $y_B$ 和 $z_B$ 相对于坐标系{A}的方向余弦可组成 $3 \times 3$ 矩阵 ${}^A R_B = \begin{pmatrix} x_B & y_B & z_B \end{pmatrix}$ ，展开得式（3.2）。

$${}^A R_B = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

(3.2)

${}^A_B R$  是一个正交的矩阵，称为旋转矩阵。接下来，可以推导出绕三个轴旋转的旋转矩阵。以绕 Z 轴旋转为例，进行坐标变换，可得：

$${}^A x_p = {}^B x_p \cos \theta - {}^B y_p \sin \theta \quad (3.3)$$

$${}^A y_p = {}^B x_p \sin \theta + {}^B y_p \cos \theta \quad (3.4)$$

$${}^A z_p = {}^B z_p \quad (3.5)$$

由式 (3.4) 至 (3.5) 可得：

$$\begin{pmatrix} {}^A x_p \\ {}^A y_p \\ {}^A z_p \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^B x_p \\ {}^B y_p \\ {}^B z_p \end{pmatrix} \quad (3.6)$$

其中式 (3.7) 即为绕 Z 轴的旋转矩阵。

$$Rot(z, \theta) = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.7)$$

同理，可得：

$$Rot(x, \theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{pmatrix} \quad (3.8)$$

$$Rot(y, \theta) = \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \quad (3.9)$$

(3.7) 至 (3.9) 这三个旋转矩阵可以表示固接在刚体上的坐标系 {B} 相对于参考坐标系的姿态，可以作为坐标变换矩阵，使得 {B} 中点的坐标转换为 {A} 中点的坐标，因此，在运动学分析中，旋转矩阵具有重要的作用。

刚体的位姿，即位置和姿态，可用刚体的旋转矩阵和方位参考坐标的原点位置矢量来表示，即

$$\{B\} = \{{}^A_B R \quad {}^A P_{B0}\} \quad (3.10)$$

这个描述形式将在运动学分析中用于表示机械手臂末端执行器的位姿。

### 3.1.2 坐标变换



坐标变换可以分解为坐标的平移和坐标的旋转。

若坐标系{A}与{B}原点不重合，但有相同的方向，则{B}相对于{A}的位置可以用位置矢量 ${}^A P_{B0}$ 表示。设{B}中一点P坐标为 ${}^B P$ ，则其相对于{A}的位置矢量可表示为坐标平移方程：

$${}^A P = {}^B P + {}^A P_{B0} \quad (3.11)$$

若坐标系{A}与{B}原点重合，但方向不同，则点P相对于{A}的位置矢量可表示为坐标旋转方程：

$${}^A P = {}^A R {}^B P \quad (3.12)$$

一般情况下，{A}和{B}原点不重合，方向也不同，可设一过渡坐标系{C}，{C}的方向和{A}相同，原点和{B}重合，根据上述两个方程，可得坐标系{B}到坐标系{C}变换方程和坐标系{C}到{A}的变换方程：

$${}^C P = {}^C R {}^B P = {}^A R {}^B P \quad (3.13)$$

$${}^A P = {}^C P + {}^A P_{C0} = {}^C P + {}^A P_{B0} \quad (3.14)$$

联立（3.13）和（3.14）两式，得到两坐标系原点和方向均不同的一般情况的坐标复合变换方程：

$${}^A P = {}^A P_{B0} + {}^A R {}^B P \quad (3.15)$$

对式（3.15）进行齐次变换，得：

$$\begin{pmatrix} {}^A P \\ 1 \end{pmatrix} = \begin{pmatrix} {}^A R & {}^A P_{B0} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} {}^B P \\ 1 \end{pmatrix} \quad (3.16)$$

称

$${}^A T_B = \begin{pmatrix} n & o & a & p \end{pmatrix} = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} {}^A R & {}^A P_{B0} \\ 0 & 1 \end{pmatrix} \quad (3.17)$$

为齐次变换矩阵，可用以描述同一点在坐标系{B}和{A}中的变换，也可描述坐标{B}相对于坐标系{A}的位置和姿态。其中，矢量 $p$ 为参考点位置矢量，用于表示{B}相对于{A}的位置。其余三个矢量分别为法向矢量 $n$ ，方向矢量 $o$ 以及接近矢量 $a$ ，用于表示{B}相对于{A}的姿态。

## 3.2 机械手臂的运动学求解

### 3.2.1 连杆参数和连杆坐标系

对于机械手臂的运动学分析，重点在于研究末端执行器的位姿，而末端位姿与各个连杆的尺寸、运动方式和相邻连杆之间的关系有着直接的联系。想要研究机械手臂末端的位姿，必须首先建立起连杆坐标系来分析连杆之间的相互关系。

连杆坐标系的建立方法一般采用 Denavit 和 Hartenberg 提出的 D-H 法<sup>[6]</sup>，这种方法严格定义了每个坐标系的坐标轴，并对连杆和关节定义了 4 个参数，如图 3.1 所示。

- (1) 连杆长度：关节  $i$  和关节  $i-1$  的轴线的公垂线段的长  $a_{i-1}$
- (2) 连杆扭角：关节  $i$  和关节  $i-1$  的轴线之间的夹角  $\alpha_{i-1}$
- (3) 连杆偏置：沿着关节  $i$  的轴线的两条公垂线之间的距离  $d_i$
- (4) 连杆转角：沿着关节  $i$  的轴线的两条公垂线之间的夹角  $\theta_i$

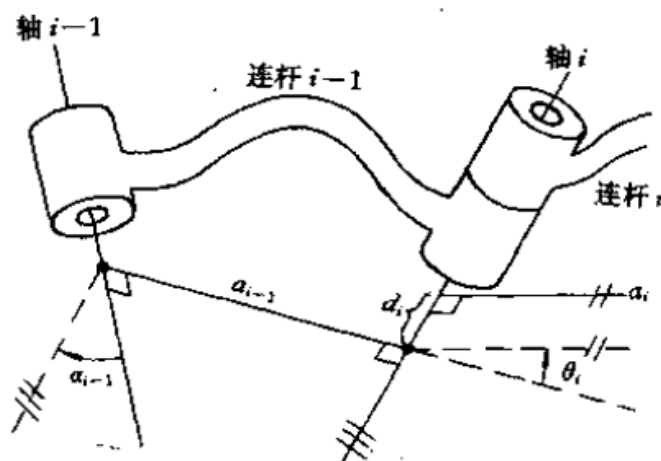


图 3.1 连杆参数示意图

对设计中所使用六自由度机械手臂各个连杆参数进行测量，得到如下连杆参数表 3.1，长度单位是 mm：

机械手臂连杆参数表

表 3.1

连杆标号 $i$	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
1	0	0	110	$\theta_1$
2	0	$-90^\circ$	115	$\theta_2$
3	100	0	0	$\theta_3$
4	0	$-90^\circ$	100	$\theta_4$
5	0	$90^\circ$	40	$\theta_5$
6	0	$-90^\circ$	45	$\theta_6$

每个关节上分别固接直角坐标系  $\{0\}, \{1\} \dots \{i\}$ ，以便于研究连杆运动的相互关系<sup>[5]</sup>。

基坐标系  $\{0\}$  被用来描述其他连杆的运动，为了简便起见，总是规定：当第一个连杆对应的关节变量为 0 时， $\{0\}$  和  $\{1\}$  的轴线重合。对于与关节  $i-1$  固接的坐标系  $\{i-1\}$ ，规定  $Z$  轴与关节轴线共线， $X$  轴与轴线的公垂线重合，指向关节  $i$ ， $Y$  轴按右手法则确定。

坐标系  $\{i\}$  相对于坐标系  $\{i-1\}$  的变换可分解为 4 个子变换：

- (1) 坐标系  $\{i\}$  绕  $X_{i-1}$  轴旋转角  $\alpha_{i-1}$ ，相应算子为  $Rot(x, \alpha_{i-1})$
- (2) 坐标系  $\{i\}$  沿  $X_{i-1}$  平移  $a_{i-1}$  距离，相应算子为  $Trans(x, a_{i-1})$
- (3) 坐标系  $\{i\}$  绕  $Z_i$  轴旋转角  $\theta_i$ ，相应算子为  $Rot(z, \theta_i)$
- (4) 坐标系  $\{i\}$  沿  $Z_i$  轴平移  $d_i$  距离  $Trans(z, d_i)$

上述子变换均相对动坐标系描述，因此，按照“从左向右”的原则，得到连杆变换通式：

$${}^{i-1}_i T = Rot(x, \alpha_{i-1}) Trans(x, a_{i-1}) Rot(z, \theta_i) Trans(z, d_i) \quad (3.18)$$

$$= \begin{pmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

其中， $c$  表示  $\cos$  函数， $s$  表示  $\sin$  函数，为简化表达式，下文中均采用这种表示方式，并用  $i$  表示  $\theta_i$ 。结合表 3.1 中的连杆参数，得到机械手臂各个连杆变换矩阵：

$${}^0_1T = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.19)$$

$${}^1_2T = \begin{pmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.20)$$

$${}^2_3T = \begin{pmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.21)$$

$${}^3_4T = \begin{pmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.22)$$

$${}^4_5T = \begin{pmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.23)$$

$${}^5_6T = \begin{pmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.24)$$

利用变换矩阵（3.19）至（3.24），就可以对机械手臂求运动学正解，利用它们的逆矩阵，还可以对机械手臂求运动学逆解。

### 3.2.2 求运动学正解

机械手臂的运动学正解的目的，是根据关节转角  $\theta_1, \theta_2 \dots \theta_6$  来推导得到末端执行器相对于基坐标系的位姿  ${}^0_7T$ 。求  ${}^0_7T$  需要先求出关节 6 相对于基坐标系的位姿  ${}^0_6T$ 。

坐标系{6}至基坐标系{0}的变换可以分解为{6}至{5}，{5}至{4}……{1}至{0}的变换，每一次变换都相当于左乘一个变换矩阵，得到：

$${}^0_6T = {}^0_1T(\theta_1){}_1^2T(\theta_2){}_2^3T(\theta_3){}_3^4T(\theta_4){}_4^5T(\theta_5){}_5^6T(\theta_6) \quad (3.25)$$

对式（3.25）进行分步计算，分别求  ${}^1_3T$  和  ${}^3_6T$ ，相乘得到  ${}^1_6T$ ，再将  ${}^1_6T$  左乘  ${}^0_1T$  得到  ${}^0_6T$ 。

$${}^1_3T = {}^1_2T(\theta_2){}_2^3T(\theta_3) = \begin{pmatrix} c_{23} & -s_{23} & 0 & a_2c_2 \\ 0 & 0 & 1 & 0 \\ -s_{23} & -c_{23} & 0 & -a_2s_2+d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.26)$$

其中，  $s_{23} = \sin(\theta_2 + \theta_3) = s_2c_3 + c_2s_3$ ，  $c_{23} = \cos(\theta_2 + \theta_3) = c_2c_3 - s_2s_3$ 。

$${}^3_6T = {}^3_4T(\theta_4){}_4^5T(\theta_5){}_5^6T(\theta_6) = \begin{pmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & -c_4s_5 & -d_6c_4s_5 \\ s_5c_6 & -s_5s_6 & c_5 & d_6c_5 + d_5 + d_4 \\ -s_4c_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 & d_6s_4s_5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.27)$$

将式（3.26）与（3.27）相乘，得到：

$${}^1_6T = {}^1_3T{}^3_6T = \begin{pmatrix} c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6 - c_{23}(c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6 & -c_{23}c_4s_5 - s_{23}c_5 & -d_6c_{23}c_4s_5 - s_{23}(d_6c_5 + d_4 + d_5) + a_2c_2 \\ -s_4s_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 & d_6s_4s_5 \\ -s_4c_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 & d_6s_{23}c_4s_5 - c_{23}(d_6c_5 + d_4 + d_5) - a_2s_2 + d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(3.28)

将坐标系{6}至基坐标系{0}的变换矩阵用式（3.17）中的形式表示，即：

$${}^0T_6 = {}^0T_1 T(\theta_1) {}^1T_6 = \begin{pmatrix} n & o & a & p \end{pmatrix} = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.29)$$

将（3.19）与（3.28）相乘后计算得到矩阵中各项：

$$\begin{aligned} n_x &= c_1 c_{23} (c_4 c_5 c_6 - s_4 s_6) - c_1 s_{23} s_5 s_6 + s_1 (s_4 c_5 c_6 + c_4 s_6), \\ n_y &= s_1 c_{23} (c_4 c_5 c_6 - s_4 s_6) - s_1 s_{23} s_5 s_6 - c_1 (s_4 c_5 c_6 + c_4 s_6), \\ n_z &= -s_{23} (c_4 c_5 c_6 - s_4 s_6) - c_{23} s_5 c_6, \\ o_x &= -c_1 c_{23} (c_4 c_5 s_6 + s_4 c_6) + c_1 s_{23} s_5 s_6 - s_1 (s_4 c_5 c_6 - c_4 c_6), \\ o_y &= -s_1 c_{23} (c_4 c_5 s_6 + s_4 c_6) + s_1 s_{23} s_5 s_6 + c_1 (s_4 c_5 s_6 - c_4 c_6), \\ o_z &= s_{23} (c_4 c_5 s_6 + s_4 c_6) + c_{23} s_5 s_6, \\ a_x &= -c_1 c_{23} c_4 s_5 - c_1 s_{23} c_5 - s_1 s_4 s_5, \\ a_y &= -s_1 c_{23} c_4 s_5 - s_1 s_{23} c_5 + c_1 s_4 s_5, \\ a_z &= s_{23} c_4 s_5 - c_{23} c_5, \\ p_x &= -d_6 c_1 c_{23} c_4 s_5 - d_6 s_1 s_4 s_5 - c_1 s_{23} (d_6 c_5 + d_5 + d_4) + a_2 c_1 c_2, \\ p_y &= -d_6 s_1 c_{23} c_4 s_5 + d_6 c_1 s_4 s_5 - s_1 s_{23} (d_6 c_5 + d_5 + d_4) + a_2 s_1 c_2, \\ p_z &= d_6 c_{23} c_4 s_5 - c_{23} (d_6 c_5 + d_5 + d_4) - a_2 s_2 + d_1 + d_2. \end{aligned}$$

本次设计所采用的机械手臂，并未搭载实体的末端执行器，在运动学分析中，将末端执行器视作关节 6 上的一点，从而可以得到末端执行器坐标系{T}相对于{0}的变换矩阵：

$${}^0T_T = {}^0T_6 {}^6T_T = {}^0T \quad (3.30)$$

将各个关节的转角  $\theta_1, \theta_2 \dots \theta_6$  代入（3.29）中各项，即可得到末端执行器的位姿  ${}^0T_T$ ，也就是机械手臂的运动学正解。

### 3.2.3 求运动学逆解

机械手臂运动学逆解的目标，是根据末端执行器的位姿，求解得到各个关节的转角。求运动学逆解的方法有很多种，如 Paul 等人提出的反变化法，Manocha 等人提出的符号化法等<sup>[12-13]</sup>。本设计采用反变换法，利用代数方法求解。

(1) 求关节 1 转角  $\theta_1$ ：

将  ${}^0_1T$  的逆矩阵与  ${}^0_7T$  相乘，得：

$${}^0_1T^{-1} {}^0_7T = \begin{pmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = {}^1_6T \quad (3.31)$$

矩阵  ${}^1_6T$  中，元素  $r_{24} = d_6 s_4 s_5 = d_6 r_{23}$ ，代入到 (3.31)，可得：

$$c_1(d_6 a_y - p_y) - s_1(d_6 a_x - p_x) = 0 \quad (3.32)$$

由于期望得到的角度值范围在  $-180^\circ$  与  $180^\circ$  之间，当  $\tan(\theta) = \infty$  时（如  $\pm 90^\circ$ ）， $\arctan$  函数无法求解，所以使用  $\arctan 2$  函数求解：

$$\theta_1 = \arctan 2(d_6 a_y - p_y, d_6 a_x - p_x)$$

或 (3.33)

$$\theta_1 = \arctan 2(-d_6 a_y + p_y, -d_6 a_x + p_x)$$

(2) 求关节 2 转角  $\theta_2$ ：

令式 (3.31) 等号两边矩阵元素  $r_{14} - d_6 r_{13}$ ， $r_{34} - d_6 r_{33}$  对应相等，

得到等式：

$$-s_{23}(d_4 + d_5) + a_2 c_2 = p_x c_1 + p_y s_1 - d_6(a_x c_1 + a_y s_1) \quad (3.34)$$

$$-c_{23}(d_4 + d_5) + d_2 - a_2 s_2 = -d_6 a_z + p_z - d_1 \quad (3.35)$$

整理式 (3.34) 和 (3.35)，可得：

$$\begin{cases} (d_4 + d_5)s_{23} = d_6(a_x c_1 + a_y s_1) + a_2 c_2 - p_x c_1 - p_y s_1 \\ (d_4 + d_5)c_{23} = d_6 a_z - p_z + d_1 \end{cases} \quad (3.36)$$

对 (3.36) 中上下两等式两边平方和相加，消去  $\theta_{23}$ ，得到：

$$\begin{aligned}
& c_2[2a_2(-d_6a_xc_1-d_6a_ys_1+p_xc_1+p_ys_1)]-s_2[2a_2(-d_6a_z+p_z-d_1-d_2)] \\
& =(-d_6a_xc_1-d_6a_ys_1+p_xc_1+p_ys_1)^2+(-d_6a_z+p_z-d_1-d_2)^2-(d_4+d_5)^2+a_2^2
\end{aligned} \tag{3.37}$$

将上式写作：

$$c_2t_{21}-s_2t_{22}=k_2 \tag{3.38}$$

从而求得

$$\theta_2 = \arctan 2(t_{21}, t_{22}) - \arctan 2(k_2, \pm\sqrt{t_{21}^2 + t_{22}^2 - k_2^2}) \tag{3.39}$$

$\theta_1$  有 2 个解，故  $\theta_1 \sim \theta_2$  有 4 组解。

(3) 求关节 3 转角  $\theta_3$ ：

式 (3.34) 减去式 (3.35) 可得：

$$\begin{aligned}
& c_{23}(d_4+d_5)-s_{23}(d_4+d_5)=k_{23}, \\
& k_{23}=-d_6(a_xc_1+a_ys_1-a_z)+p_xc_1+p_ys_1-a_2c_2-a_2s_2-p_z+d_1+d_2
\end{aligned} \tag{3.40}$$

求得

$$\theta_{23} = \arctan 2(d_4+d_5, d_4+d_5) - \arctan 2(k_{23}, \pm\sqrt{2(d_4+d_5)^2 - k_{23}^2}) \tag{3.41}$$

需要注意的是，由于上式中根据正切值求得的  $\theta_{23}$ ，实际上是  $\theta_{23} \pm 180^\circ$ ，若直接使用，会导致  $\theta_3$  出现错误的解。因此，需要将  $\theta_{23}$  值代入 (3.34) 和 (3.35) 进行验证。对于每个经过验证的  $\theta_{23}$ ，可以求得：

$$\theta_3 = \theta_{23} - \theta_2 \tag{3.42}$$

对于  $\theta_1 \sim \theta_2$  的每组解，可以求得 2 个  $\theta_{23}$  的解，其中一个为验证后符合条件的解，每个  $\theta_{23}$  的解又对应一个  $\theta_3$  的解，因此  $\theta_1 \sim \theta_3$  有 4 组解。

(4) 求关节 4 转角  $\theta_4$ ：

将  ${}^0_3T$  的逆矩阵与  ${}^0_6T$  相乘，得：

$$\begin{aligned}
& {}^0_3T^{-1} {}^0_6T = \begin{pmatrix} c_1c_{23} & s_1c_{23} & -s_{23} & d_1s_{23}+d_2s_{23}-a_2c_3 \\ -c_1s_{23} & -s_1s_{23} & -c_{23} & d_1c_{23}+d_2c_{23}+a_2s_3 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& = {}^3_6T
\end{aligned} \tag{3.43}$$



令等式两边矩阵元素  $r_{13}$  和  $r_{33}$  对应相等，得到方程组：

$$\begin{cases} -c_4s_5 = a_xc_1c_{23} + a_ys_1c_{23} - a_zs_{23} \\ s_4s_5 = -a_xs_1 + a_yc_1 \end{cases} \quad (3.44)$$

当  $s_5 = 0$  时，机械手臂处于奇异形位，此时，只能得到  $\theta_4$  和  $\theta_6$  的和或差，因此具有无穷多解，在计算程序中，需要对这种情况作出警告<sup>[6]</sup>。

当  $s_5 \neq 0$  时，上下两式相除，可得：

$$\theta_4 = \arctan 2(-a_xs_1 + a_ys_1, -a_xc_1c_{23} - a_ys_1c_{23} - a_zs_{23})$$

或 (3.45)

$$\theta_4 = \arctan 2(a_xs_1 - a_ys_1, a_xc_1c_{23} + a_ys_1c_{23} + a_zs_{23})$$

对于每个  $\theta_{23}$ ， $\theta_4$  有 2 个解，因此  $\theta_1 \sim \theta_4$  一共有 8 组解。

(5) 求关节 5 转角  $\theta_5$ ：

将  ${}^0_4T$  的逆矩阵与  ${}^0_6T$  相乘，令等式两边矩阵元素  $r_{11}$  和  $r_{33}$  分别相等，得：

$$\begin{cases} -s_5 = a_x(c_1c_{23}c_4 + s_1s_4) + a_y(s_1c_{23}c_4 - c_1s_4) - a_zs_{23}c_4 \\ c_5 = -a_xc_1s_{23} - a_zc_{23} - a_ys_1s_{23} \end{cases} \quad (3.46)$$

上下两式相除，得：

$$\theta_5 = \arctan 2(-a_x(c_1c_{23}c_4 + s_1s_4) - a_y(s_1c_{23}c_4 - c_1s_4) + a_zs_{23}c_4, -a_xc_1s_{23} - a_zc_{23} - a_ys_1s_{23}) \quad (3.47)$$

对于  $\theta_1 \sim \theta_4$  的每组解， $\theta_5$  都有一个对应的解，因此  $\theta_1 \sim \theta_5$  共 8 组解。

(6) 求关节 6 转角  $\theta_6$ ：

将  ${}^0_5T$  的逆矩阵与  ${}^0_6T$  相乘，令等式两边矩阵元素  $r_{11}$  和  $r_{33}$  分别相等，得：

$$\begin{cases} -s_6 = a_x[c_5(c_1c_{23}c_4 + s_1s_4) - c_1s_{23}s_5] + a_y[c_5(s_1c_{23}c_4 - c_1s_4) - s_1s_{23}s_5] - a_z(s_{23}c_4c_5 + s_5c_{23}) \\ c_6 = n_x[c_5(c_1c_{23}c_4 + s_1s_4) - c_1s_{23}s_5] + n_y[c_5(s_1c_{23}c_4 - c_1s_4) - s_1s_{23}s_5] - n_z(s_{23}c_4c_5 + s_5c_{23}) \end{cases} \quad (3.48)$$

从而，

$$\theta_6 = \arctan 2(-a_x[c_5(c_1c_{23}c_4 + s_1s_4) - c_1s_{23}s_5] - a_y[c_5(s_1c_{23}c_4 - c_1s_4) - s_1s_{23}s_5] + a_z(s_{23}c_4c_5 + s_5c_{23}), n_x[c_5(c_1c_{23}c_4 + s_1s_4) - c_1s_{23}s_5] + n_y[c_5(s_1c_{23}c_4 - c_1s_4) - s_1s_{23}s_5] - n_z(s_{23}c_4c_5 + s_5c_{23})) \quad (3.49)$$

对于  $\theta_1 \sim \theta_5$  的每组解， $\theta_6$  都有一个对应的解，因此  $\theta_1 \sim \theta_6$  共 8 组解。这 8 组角度也就是机械手臂的运动学逆解的结果，每一组解都能使得末端执行器达到给定的位姿。

## 4 机械手臂的运动控制

### 4.1 轨迹规划

对于机械手臂的运动控制，分为轨迹规划和轨迹执行两个部分。

轨迹规划是将机械手臂的工作轨迹进行分解，通过插值算法生成大量的中间点，再通过求运动学反解等方法来得到每个中间点的对应关节转角。

插值的算法有很多种，在关节空间内可使用三次样条插值法、多项式插值法等，在直角坐标空间内可以有直线插补法、圆弧插补法等<sup>[8][14][15]</sup>。

由于本设计所采用的驱动系统为数字舵机，无法控制其加速度，只能对机械手臂各个关节的角度变化进行控制。因此，从实际应用角度出发，可采用关节空间内的转角插值法，或是直角坐标空间内的直线插补法。

#### 4.1.1 转角插值法

转角插值法在关节空间内进行，完全针对关节转角变量进行，不需要获取直角坐标系中的位姿信息。转角插值法的具体算法如下：

首先，确定轨迹的起点和终点的末端执行器位姿信息，根据这两个点的位姿信息求运动学逆解，可以得到起点和终点的关节转角数据  $P_0(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$  和  $P_1(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ 。

然后，求出这两组转角数据中每个关节转角的变化量  $\Delta\theta_1 \sim \Delta\theta_6$ ，取变化量最大值  $\Delta\theta_{\max}$  除以插值速度  $\theta_d$ ，对结果向上取整后即得到插补线段数，再加上 1，即为包括起点和终点在内的所有点的个数  $N$ ，即：

$$N = \text{ceil}(\Delta\theta_{\max} / \theta_d) + 1 \quad (4.1)$$

接着，对  $\theta_1 \sim \theta_6$  进行角度插补，关节  $i$  在第  $n$  个插入点的转角  $\theta_i(n)$  可表示为：

$$\theta_i(n) = \theta_i(1) + [n/(N-1)] * [\theta_i(N) - \theta_i(1)] \quad (4.2)$$

其中， $\theta_i(1)$  和  $\theta_i(N)$  分别为起点转角值和终点转角值。

将每组角度值发送至舵机控制器对舵机进行控制，就可以使得机械手臂按照轨迹运动。

### 4.1.2 直线插补法

直线插补法在直角坐标空间内进行，需要进行大量的运动学逆解求解。直线插补法的具体算法如下：

首先，确定轨迹的起点和终点的末端执行器位姿信息，得到两个点在直角坐标空间中的位置  $(p_{x0}, p_{y0}, p_{z0})$  和  $(p_{x1}, p_{y1}, p_{z1})$

然后，求出起点与终点的直线距离  $L$ ，

$$L = \sqrt{(p_{x1} - p_{x0})^2 + (p_{y1} - p_{y0})^2 + (p_{z1} - p_{z0})^2} \quad (4.3)$$

将  $L$  除以插补速度  $d$ ，对结果向上取整后即得到插补线段数，再加上 1，即为包括起点和终点在内的所有点的个数  $N$ ，即：

$$N = \text{ceil}(L/d) + 1 \quad (4.4)$$

接着，对起点和终点之间的线段插入中间点，第  $n$  个插入点的坐标可表示为：

$$\begin{cases} p_x(n) = p_x(1) + [n/(N-1)] * [p_x(N) - p_x(1)] \\ p_y(n) = p_y(1) + [n/(N-1)] * [p_y(N) - p_y(1)] \\ p_z(n) = p_z(1) + [n/(N-1)] * [p_z(N) - p_z(1)] \end{cases} \quad (4.5)$$

其中， $p_{x,y,z}(1)$  和  $p_{x,y,z}(N)$  分别为起点坐标和终点坐标。

对于  $N$  个点的坐标，分别求运动学逆解，考虑约束条件，对每个点取一组最优解，则可得到  $N$  组角度，将每组角度值发送至舵机控制器对舵机进行控制，就可以使得机械手臂按照轨迹运动。

需要注意的是，直线插补法中，插入点仅仅改变了末端执行器的位置，而末端执行器的姿态并未改变，所以这种方法适用于仅需要控制末端位置的情况，不适用于末端姿态的控制。

## 4.2 轨迹执行

4.1 节中介绍的两种插值算法有各自的优缺点：转角插值法适用于机械手臂末端执行器的位置和姿态均需要控制的情况，但在直角坐标空间中的运动不够平滑；直线插补法使得机械手臂末端执行器能够平滑地运动，不会出现速度过快或过慢的情况，但是不适用于末端姿态的控制。

两种插值算法的轨迹规划和轨迹执行过程如下：

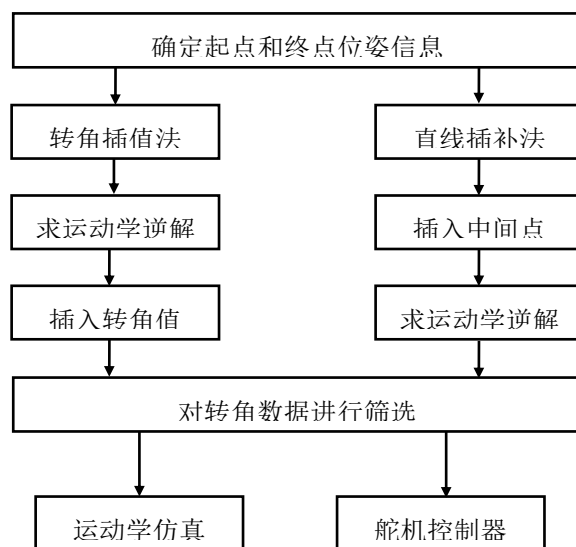


图 4.1 机械手臂轨迹规划、轨迹执行过程

在用插值法求得轨迹上每个点的转角数据之后，用软件中的仿真模块进行仿真，同时将转角数据转化为控制命令，发送至舵机控制器，完成对机械手臂的控制。

在实际的轨迹执行过程中，需要考虑到机械手臂的运动空间限制，若对关节转角不加以控制，则机械手臂有可能会与底座、云台等发生碰撞。因此，在软件的设计中，需要对轨迹规划中取角度步骤添加了严格的范围限制，以防止发生碰撞。

## 5 计算与仿真模块设计

### 5.1 计算与仿真模块概述

#### 5.1.1 模块功能设计

为了对机械手臂进行轨迹控制，控制软件必须具有计算模块，来进行运动学求解和轨迹规划。同时，还需要一个仿真模块来对轨迹进行监视，来确认计算得到的轨迹是否满足要求<sup>[9]</sup>。

这两个模块关系密切，而它们的设计并不依赖于机械手臂的嵌入式系统设计，因此独立于控制模块。所以，在实际设计中，对计算与仿真模块和控制模块的设计是分开进行的。

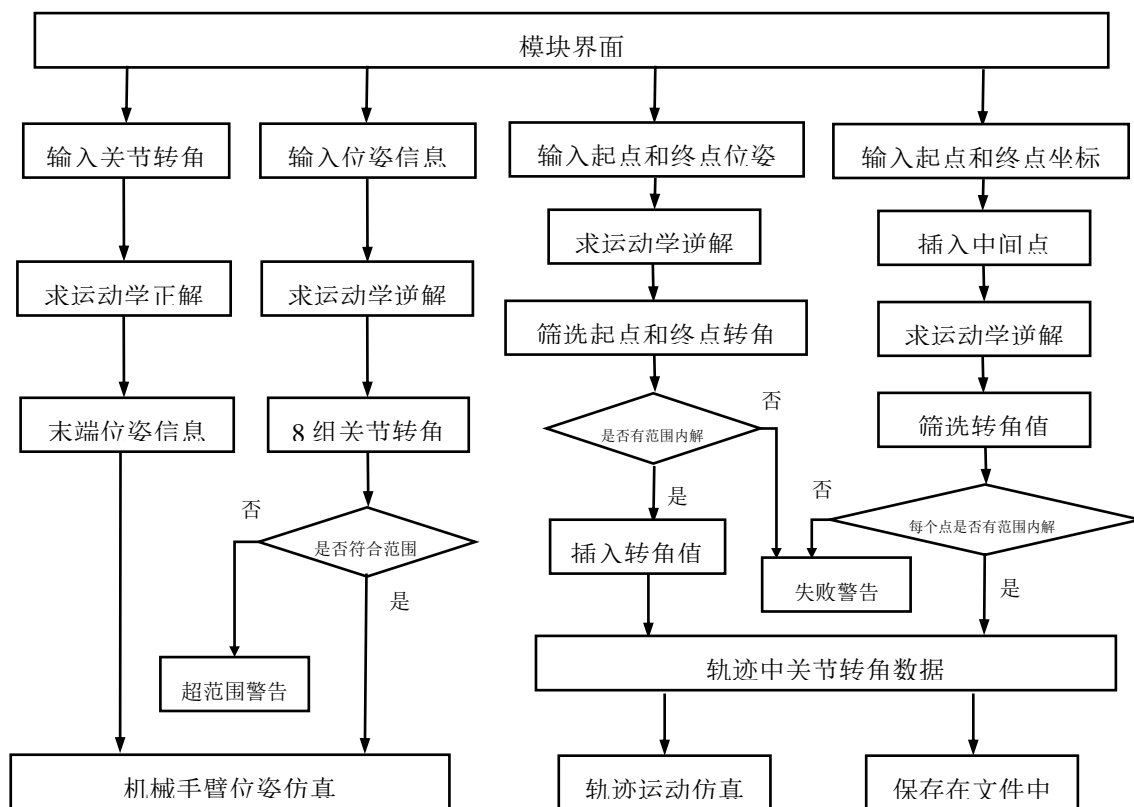


图 5.1 计算与仿真模块各项功能流程图

如图 5.1 所示，计算与仿真模块需要完成的功能有：

(1) 从界面输入各个关节的转角，求运动学正解，得到末端执行器的位姿，

同时用三维模型对机械手臂整体位姿进行仿真

（2）从界面输入末端执行器的位姿，求运动学逆解，得到 8 组不同的关节转角的解，每一组解都可以用三维模型进行仿真

（3）从界面输入起点和终点的空间坐标和插补速度，利用直线插补法进行轨迹规划，得到插补后的各个点对应的关节转角，同时用三维模型对机械手臂的轨迹运动进行仿真

（4）从界面输入起点和终点的位姿信息和插补速度，利用关节插值法进行轨迹规划，得到插值后的每一组关节转角，同时用三维模型对机械手臂的轨迹运动进行仿真

（5）在轨迹规划后，保存角度信息，同时可被控制模块取用。

### 5.1.2 编程语言和开发工具的选择

设计采用的编程语言和开发工具取决于软件的目标平台。目前 PC 的主流操作系统为 Windows 7，因此，计算与仿真的模块将针对 Windows 7 平台来进行。编程语言有很多种，如 Java，C，Python 等。在 Windows 平台上，C++语言是应用最广泛的语言之一，C++作为一种以 C 语言为基础的面向对象编程语言，对 C 语言进行了扩充<sup>[16]</sup>。C++丰富的标准库完全可以满足本设计的软件开发需要。同时，C 语言常在本科教学中使用，使用经验丰富，且 C++有着充足的相关资料，学习成本相对其他语言要低。综合考虑，设计采用 C++作为编程语言。

本设计使用 Windows 7 x64 操作系统作为开发平台。在此平台上，设计中选择微软公司的 Visual Studio 2015 作为开发工具，Visual Studio 2015 不仅对 Windows 7 x64 平台具有良好的支持性，其搭载的 MSVC2015 编译器，在 Windows 平台上的 C++开发上也具有编译速度快、软件体积小等优点。

## 5.2 计算模块的程序实现

### 5.2.1 求运动学正解

运动学正解的求解，其输入是 6 个关节的转角变量，输出一个  $4 \times 4$  的末端执行器位姿矩阵。矩阵除最后一行外均为变量。

输入关节转角(°)

$\theta_1$	$\theta_2$	$\theta_3$	$\theta_4$	$\theta_5$	$\theta_6$
11	22	33	44	55	66

正解计算

末端位姿

nx=	ox=	ax=	px=
-0.374414625	0.216803861	-0.90155963	-6.21295638
ny=	oy=	ay=	py=
-0.90732135	0.114889875	0.404435760	1.400889019
nz=	oz=	az=	pz=
0.191263308	0.969430968	0.153694322	11.41548840
0	0	0	1

图 5.2 求运动学正解模块界面

针对运动学正解计算的输入输出变量，首先设计出程序界面，如图 5.2 所示。对于通过程序界面输入的 6 个关节转角值，程序根据 3.2.2 节中的位姿矩阵各元素计算公式，计算得到机械手臂的末端位姿，并在仿真模块中加以展示。

程序中使用了 `UpdateData()` 函数来进行控件和程序变量的数据交换。当参数为 `True` 时，`UpdateData(True)` 可将界面上控件所示的数据传送给相关联的变量；当参数为 `False` 时，`UpdateData(False)` 根据程序中变量的数据来更新界面上控件所示内容。

用户在程序界面输入了所有转角信息后，点击“正解计算”按钮，程序随即调用 `UpdateData(True)` 函数，将输入的转角数据传送给关联变量，由关联变量参与相关运算，得到各元素的值。然后，程序调用 `UpdateData(False)` 函数，将界面上矩阵的控件内容更新为各元素的值。

同时，程序将输入的角度数据发送到仿真模块进行仿真，用户可以看到输入的角度对应的机械手臂位姿。

### 5.2.2 求运动学逆解

运动学逆解的求解，其输入为机械手臂末端位姿的 12 个变量，其输出为 8 组关节转角的解，一共有 48 个输出变量。



清空数据		逆解计算		<input checked="" type="checkbox"/> 角度限制				
	$\theta_1$	$\theta_2$	$\theta_3$	$\theta_4$	$\theta_5$	$\theta_6$		
角度范围	-135 ~ 135	-135 ~ 135	-135 ~ 135	-135 ~ 135	-135 ~ 135	-135 ~ 135		
第一组解	11.0000	179.128	147	44.2035	125.298	124.317	仿真展示	$\theta_2 \theta_3$ 超出其角度范围
第二组解	11.0000	179.128	147	-135.79	-125.29	-55.682	仿真展示	$\theta_2 \theta_3 \theta_4$ 超出其角度范围
第三组解	11.0000	22.0000	32.9999	44	55.0000	66	仿真展示	
第四组解	11.0000	22.0000	32.9999	-136	-55.000	-114	仿真展示	$\theta_4$ 超出其角度范围
第五组解	-169	158	147	-136	55.0000	66	仿真展示	$\theta_1 \theta_2 \theta_3 \theta_4$ 超出其角度范围
第六组解	-169	158	147	44	-55.000	-114	仿真展示	$\theta_1 \theta_2 \theta_3$ 超出其角度范围
第七组解	-169	0.87102	32.9999	-135.79	125.298	124.317	仿真展示	$\theta_1 \theta_4$ 超出其角度范围
第八组解	-169	0.87102	32.9999	44.2035	-125.29	-55.682	仿真展示	$\theta_1$ 超出其角度范围

图 5.3 求运动学逆解模块界面

针对运动学逆解计算的输入输出变量，设计出程序界面如图 5.3 所示。对于通过程序界面输入的 12 个位姿变量，程序根据 3.23 节中的转角计算公式，计算得到机械手臂各个关节可能的转角值，并可以在仿真模块中分别展示。

用户在图 5.2 中的转换矩阵中输入位姿信息后，点击“逆解计算”按钮，程序运行运动学逆解算法。与运动学正解的求解类似，程序通过调用 `UpdateData()` 函数完成控件和程序变量之间的数据传递，并将结果显示在界面上。

另外，程序还加入了判断各个关节转角是否超出角度范围的功能选项，在输入角度范围并且勾选“角度限制”项，程序会判断各组解各个角度是否处于范围内，并给出超范围警告。这一功能同样适用于轨迹的计算。

在图 5.3 中，设定的角度范围是舵机的最大转角范围，在实际控制中，还需要对各个舵机添加更加严格的角速度限制以防止机械手臂发生碰撞。

### 5.2.3 轨迹求解

轨迹求解与仿真的程序中包含了 4.1 节中介绍的转角插值法和直线插补法两种算法。转角插值法适用于控制末端的位置和姿态，但平滑性无法得到保障；直线插补法仅适用于控制末端的位置，运动相对平滑。两种方法各有特点，可以根据实际情况进行选择。

插值方法	
<input checked="" type="radio"/> 直线插补法	$\Delta d =$ <input type="text" value="3"/>
<input type="radio"/> 转角插值法	$\Delta \theta =$ <input type="text" value="10"/>
<input type="button" value="轨迹计算"/>	<input type="button" value="设置插值速度"/>
单步时间(s)= <input type="text" value="0.5"/>	<input type="checkbox"/> 连续轨迹

图 5.4 轨迹求解与仿真模块界面

如图 5.4 所示，在界面上，用户可以在两种插值方法中进行选择，对于每一种插值方法，可以分别设定插值速度，即插入相邻两点之间变量的差值。插值速度影响了轨迹中插入点的数量。另外，用户还可以设置单步时间，即机械手臂末端从一个插入点移动到下一个插入点的时间，插值速度和单步时间影响了机械手臂完成轨迹运动的总时间。根据机械手臂轨迹的长度和期望的完成动作时间，可以适当调节插值速度和单步时间。

若选择直线插补法，用户需要首先将起点位姿信息填入位姿矩阵，可以手动填入，也可以通过求运动学正解来让程序填入。然后，用户需要在终点坐标栏填入终点坐标，如图 5.5 所示。

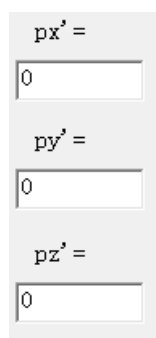


图 5.5 轨迹终点坐标栏

在填入终点坐标之后，点击“轨迹计算”按钮，程序就会根据起点和终点坐标以及插值速度进行轨迹规划，得到插入中间点后每个点对应的关节转角，并且在仿真模块进行动态的仿真。界面中输入的单步时间即为相邻两个点的仿真间隔。

若选择转角插值法，用户首先需要在位姿矩阵中填入起点位姿信息，与直线插补法一样，可以手动填入，也可以通过程序计算后自动填入。随后，点击“逆解计算”按钮，程序会在计算后自动选择一组转角作为起点转角。接着，用户需要填入终点位姿信息，再点击“轨迹计算”按钮，程序会选出一组转角作为终点转角，并根据插值速度，在起点和终点这两组转角之间进行插值。

对转角进行插值后，仿真模块对于每一个点的转角值进行仿真，间隔时间同样为界面中输入的单步时间。

### 5.3 仿真模块的程序实现

计算模块中，无论是求运动学正解、逆解，还是轨迹的计算，都需要仿真模

块进行、展示和验证。

仿真模块所需要完成的功能，是在一个三维的机械手臂模型上展示出转角的变化。因此，程序需要一个图形程序接口，即 API（Application Program Interface）来进行三维图形的处理。

现在主流的图形 API 有 Direct3D、OpenGL、Quick Draw 3D 等。由于设计需求并不复杂，不需要网络、声效等功能，因此可以选择图形处理能力强大的 OpenGL 来进行机械手臂三维模型的制作。并且，OpenGL 与 C 语言紧密结合，其命令一开始便是以 C 语言进行描述的，因此，在 Visual Studio 2015 和 C++ 环境下使用 OpenGL 制作仿真模块会比使用其他图形 API 更加简单易用。

### 5.3.1 OpenGL 窗口的建立

由于 C++ 没有为 OpenGL 提供一个可以直接使用的视图类，因此，需要创建一个 COpenGLView 类，并将 OpenGL 绘图所需的内容封装在这个类中<sup>[10]</sup>。

COpenGLView 类为 Windows 窗口和 OpenGL 提供了关联，使得 OpenGL 可以直接在 Windows 窗口中进行绘制，而不需要一个独立的窗口。该类的结构如下：

```
class COpenGLView
{
public:
    COpenGLView();    //构造函数，用于构建窗口
    CPoint pointcurrent;    //当前点
    BOOL lbuttondown;    //判断鼠标左键按下
    CDrawArm drawarm;    //CDrawArm 类提供绘制机械手臂模型的函数
    BOOL SetupLighting();    //设置光照
    BOOL SetupViewingTransform();    //设置视角
    BOOL SetupViewingFrustum(GLdouble aspect_ratio);    //设置视锥，其参数
    aspect_ratio=cx/cy，即窗口长宽比
    BOOL SetupViewPort(int cx,int cy);    //设置窗口大小
    BOOL SetupPixelFormat(HDC hdc);    //定义像素格式
    HGLRC hglrc;    //绘制描述表
```

```

HDC hdc;          //设备描述表

~COpenGLView();    //析构函数，用于清理

protected:

int OnCreate(LPCREATESTRUCT lpCreateStruct);    //设置窗口属性

void OnPaint();    //绘制机械手臂

void OnSize(UINT nType, int cx, int cy);    //根据窗口大小设置视角、光源等

//鼠标操作函数：左键按下，左键释放，鼠标移动，左键双击，滑轮滚动

void OnLButtonDown(UINT nFlags, CPoint point);

void OnLButtonUp(UINT nFlags, CPoint point);

void OnMouseMove(UINT nFlags, CPoint point);

void OnLButtonDblClk(UINT nFlags, CPoint point);

BOOL OnMouseWheel(UINT nFlags, short zDelta, CPoint pt);

};

```

其中，public 类型的成员被 protected 类型成员所引用来实现窗口初始化、机械手臂绘制、鼠标控制等功能。

在设计中，程序为 COpenGLView 类构建了一个对象 CWnd，并对其进行了初始化，在 Windows 窗口内建立起了一个 OpenGL 窗口，用户可以在这个窗口中进行鼠标操作。

### 5.3.2 机械手臂模型的绘制和仿真实现

为了绘制机械手臂的模型，设计时在程序中创建了一个名为 CDrawArm 的类，该类中的成员函数利用 OpenGL 库提供的函数，完成了绘制机械手臂的任务。CDrawArm 类结构如下：

```

class CDrawArm
{
public:

void SolidCube(GLfloat Length,GLfloat Width,GLfloat Height);//绘制长方体

void CalcNormal(GLfloat *p0,GLfloat *p1,GLfloat *p2,GLfloat *normal);//计算法向量，用于绘制斜坡和圆柱体

```

```
void SolidXieCube(GLfloat Lenght,GLfloat Width,GLfloat Height); //绘制斜坡  
void SolidCylinder(GLdouble radius,GLdouble height,int n_div);//绘制圆柱体  
void DrawArmBody(float OS,float OL,float OU,float OR,float OB,float OT);//绘制  
机械手臂  
void DrawDynamicGoods();      //在坐标处绘制机械手臂  
CDrawArm();    //构造函数  
~CDrawArm();    //析构函数  
};
```

在 DrawArmBody( )函数中，使用了 SolidCube( )等函数来绘制三维物体，使用 glTranslate( )和 glRotate( )函数完成绘制点的移动和绘制方向的旋转。在仿真时，将 6 个关节的转角值作为其参数。根据传入的参数值，DrawArmBody( )利用 glRotate( )函数来完成各个关节的旋转操作。

绘制完成后的机械手臂模型如图 5.6 所示：

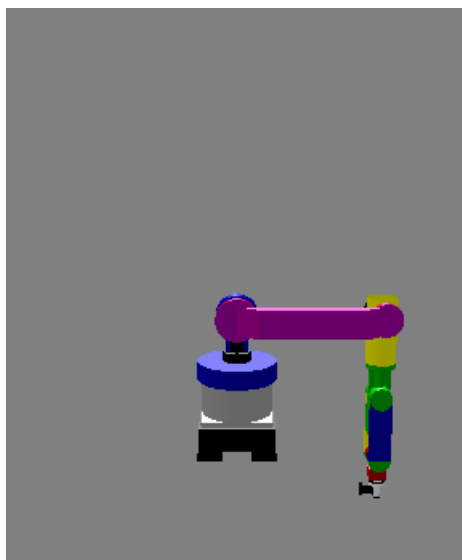


图 5.6 机械手臂三维模型

按照连杆坐标系的坐标方向，各关节的转动正方向为：关节 2,3,5 为顺时针方向，1,4,6 为逆时针方向。以传入转角参数（30,-30,-30,30,30,30）为例，得到的机械手臂位姿仿真如图 5.7 所示：

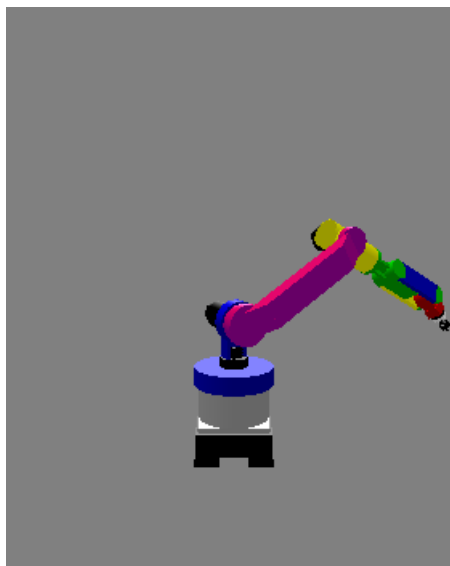


图 5.7 机械手臂仿真测试

可以看到，机械手臂模型进行了正确的仿真。

COpenGLView 类中的 Onpaint() 函数调用了 DrawArmBody() 函数。程序将要仿真的关节角度值存在数组 CS\_Left[] 中，Onpaint() 函数将数组中的元素作为参数传入 DrawArmBody() 函数，完成仿真。

利用 COpenGLView 类提供的鼠标操作函数，还可以对模型进行旋转、放大缩小等操作，便于观察。

## 5.4 计算与仿真模块的整体界面

在完成了计算模块与仿真模块的主要设计之后，还加入了一些附加功能：

（1）清空数据。点击“清空数据”按钮后，转角值、位姿信息和运动学逆解将被清空。

（2）设置连杆参数。设计中可能对机械手臂进行改装，因此程序中加入了设置参数功能，来对机械手臂的连杆偏置和连杆长度进行修改。

（3）将转角值转化为舵机控制命令，并存于文本文件中，可以打开浏览，或供控制模块调取，如图 5.8 所示，其中的控制命令格式将在下一节中介绍。

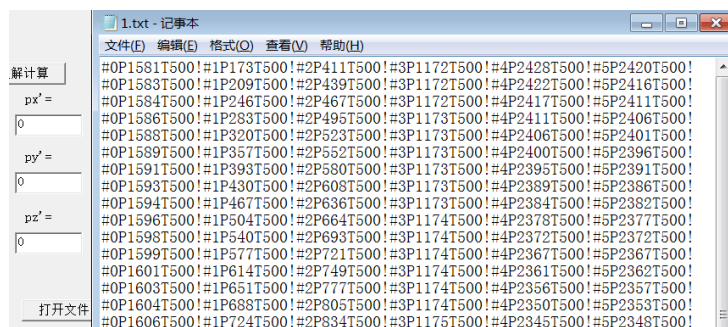


图 5.8 “打开文件”功能展示

在加入了附加功能之后，计算与仿真模块的设计就全部完成了。

在对模块进行了界面上的控件排布和初始变量设置之后，得到的整体界面如图 5.9 所示：



图 5.9 计算与仿真模块整体界面

## 6 控制模块设计

### 6.1 控制模块概述

#### 6.1.1 控制流程设计

控制模块所需要完成的功能，是在 PC 上通过串口通信，将转角值转化为控制命令，发送给下位机，由下位机完成对舵机的控制。

因此，控制模块的设计包含 2 个部分：PC 上位机和 51 单片机下位机的程序设计。控制模块工作流程如图 6.1 所示：



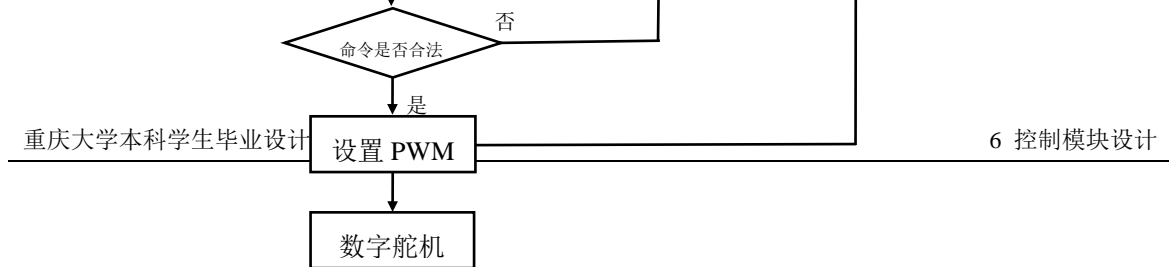


图 6.1 控制模块工作流程图

### 6.1.2 开发工具的选择

对于下位机程序的开发，设计采用了在教学实验中的 Keil uVision4 开发软件，该软件支持对 C 语言程序的编译，并可以通过串口将编译得到的\*.hex 文件下载到 STC12C5A60S2 单片机中。

对于 PC 上位机程序的开发，在 C++的基础上，采用了 Qt 框架进行开发。Qt 是由奇趣公司开发，后被诺基亚公司收购的 C++图形用户界面（GUI）应用程序开发框架。Qt 自带的串口通信模块，以及方便易用的组件，可以为程序设计提供帮助。另外，对于许多机械手臂，采用的平台并不是 Windows 操作系统，而 Qt 具有跨平台的优势，利用 Qt 框架编写的程序可以在不同的平台进行编译，如 Linux，Android 等，这为控制软件的移植提供了便利<sup>[11][17]</sup>。

## 6.2 下位机的程序实现

基于 STC12C5A60S2 单片机的下位机程序需要完成的主要功能有两个：

- （1）对上位机命令进行接收和分析，对舵机进行控制
- （2）向上位机发送反馈信息

### 6.2.1 系统的初始化

在接收和分析命令之前，下位机需要首先进行初始化，其中包含对 PCA（可编程计数器阵列）、串行口和定时器等初始化。

系统初始化程序代码如下：

```
void system_init(void)
{
    io_init();          //IO 口初始化
    global_var_init();  //全局变量初始化
    uart_init();        //串口初始化
    uart_open();        //打开串口
}
```

```
uart_send_str("uart1 init ok!");    //向上位机发送初始化成功反馈

timer0_init();        //定时器 0 初始化

timer0_open();        //打开定时器 0

timer1_init();        //定时器 1 初始化

timer1_open();        //打开定时器 1

pwm_init();           //PCA 初始化

pwm_open();           //打开 PCA 的 8 位 PWM 无中断模式

interrupt_open();

// 蜂鸣器响三声表示初始化完毕

beep_on();mdelay(100);

beep_off();mdelay(100);

beep_on();mdelay(100);

beep_off();mdelay(100);

beep_on();mdelay(100);

beep_off();mdelay(100);

return;

}
```

这些初始化函数通过对定时器、寄存器等单片机模块写控制字来完成，具体的实现代码可以查询 STC12C5A60S2 的资料手册，本文中就不再赘述。

### 6.2.2 控制命令的处理

上位机向下位机发送的控制命令应包括三个参数：舵机编号、PWM 脉宽、完成动作时间，在程序中分别以 `index`，`pwm`，`time` 表示，其中，`index` 参数影响 IO 口的选择；`pwm` 参数影响 PCA 发出的 `pwm` 脉宽；`time` 参数影响定时器的定时长度。这三项功能对应的函数，分别写在 `io.c`，`pwm.c`，`timer.c` 这三个源文件中。

为了将上述的三个参数放在控制命令中供下位机识别分析，需要规定一个命令的格式。程序中设计的针对单个舵机命令格式为“`#+index+P+pwm+T+time+!`”，将六个舵机的命令以“`{ }`”符号包括在一起，如：`{#0P1603T500!#1P651T500!`

```
#2P777T500!#3P1174T500!#4P2356T500!#5P2357T500!}
```

基于相关函数和命令格式，编写控制命令处理程序，从上位机发送的控制命令中获得 `index`，`pwm`，`time` 这三个参数的值，并返回反馈信息。对于有效的命令，输出 PWM 脉冲控制舵机转动。

## 6.3 PC 上位机的程序实现

PC 上位机需要实现对机械手臂的两种控制方式：编程输入型控制和示教输入型控制。编程输入型控制是通过计算与仿真模块计算得到轨迹运动的控制命令；示教输入型控制是通过用户手动控制机械手臂完成每个动作后，将命令保存，得到轨迹运动的控制命令。

在 Qt 提供的 QtDesigner 工具中，创建一个 MainWindow 界面文件，将所需控件摆放完毕后，针对每个控件编写相应函数。

整个上位机界面分为三个板块：PC 与下位机的通信界面、舵机控制面板、动作组操作区。下面分别对这三个板块进行介绍。

### 6.3.1 与下位机的通信

利用 Qt 框架提供的串口通信模块<QThread>，可以获得每一个串口所连接设备的名称、对应的设备描述、生产商等信息。在设计中舵机控制器采用的串口模块是 Prolific 公司生产的 PL2303 模块，其设备描述为“Prolific USB-to-Serial Comm Port”，因此，遍历所有 PC 串口，找寻描述包含上述文本的设备，即可找到与机械手臂控制器相连接的串口号。

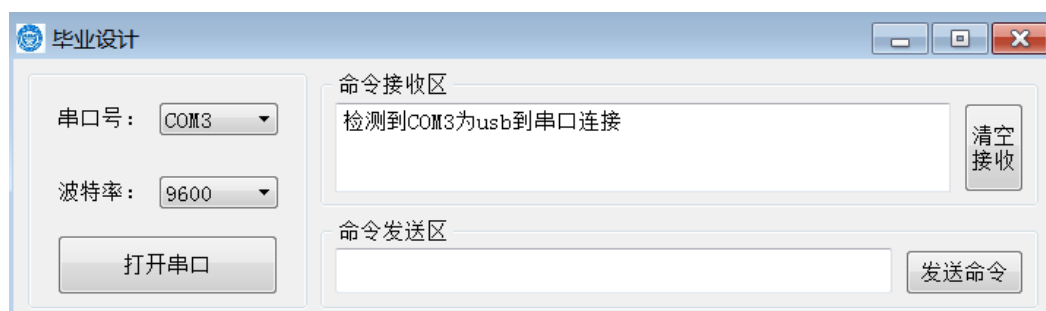


图 6.2 PC 与下位机通信界面

如图 6.2 所示，程序在波特率列表中添加了多种选择，默认的波特率为 9600，在检测到串口后，点击“打开串口”，即可与下位机进行通信。在命令发送区，可以直接手动输入舵机控制命令，在命令接收区，将下位机发送的反馈信息进行

转化后加以显示。

### 6.3.2 舵机控制面板

在图 6.3 所示的舵机控制面板中，可以对单个舵机或者多个舵机进行控制，通过选定舵机来确定舵机的 index 参数，通过拖动进度条控件（Bar）或者改变进度条右侧可调数框控件（SpinBox）的值，可以改变舵机的参数 pwm 或 time。

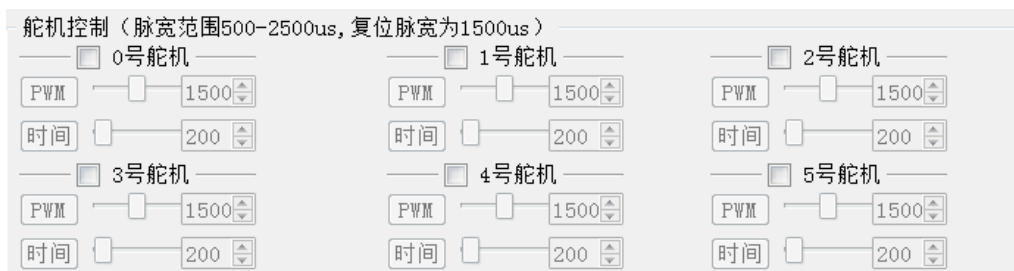


图 6.3 舵机控制面板

对于每个舵机的 pwm 和 time 参数，各有 1 个 Bar 控件和 1 个 SpinBox 控件，因此，每个舵机对应 4 个控件，需要分别编写相应的函数。

以 0 号舵机为例，对于四个控件的操作对应的函数如下：

```
void MainWindow::on_pwmBar0_valueChanged(int value)
```

```
{
    ui->pwmSpinBox0->setValue(value);
    this->pwmBarHandle(0, ui->pwmBar0->value(), ui->timeBar0->value());
}
```

```
void MainWindow::on_pwmSpinBox0_valueChanged(int value)
```

```
{
    ui->pwmBar0->setValue(arg1);
    this->pwmBarHandle(0, ui->pwmBar0->value(), ui->timeBar0->value());
}
```

```
void MainWindow::on_timeBar0_valueChanged(int value)
```

```
{  
    ui->timeSpinBox0->setValue(value);  
    this->pwmBarHandle(0, ui->pwmBar0->value(), ui->timeBar0->value());  
}  
  
void MainWindow::on_timeSpinBox0_valueChanged(int value)  
{  
    ui->timeBar0->setValue(arg1);  
    this->pwmBarHandle(0, ui->pwmBar0->value(), ui->timeBar0->value());  
}
```

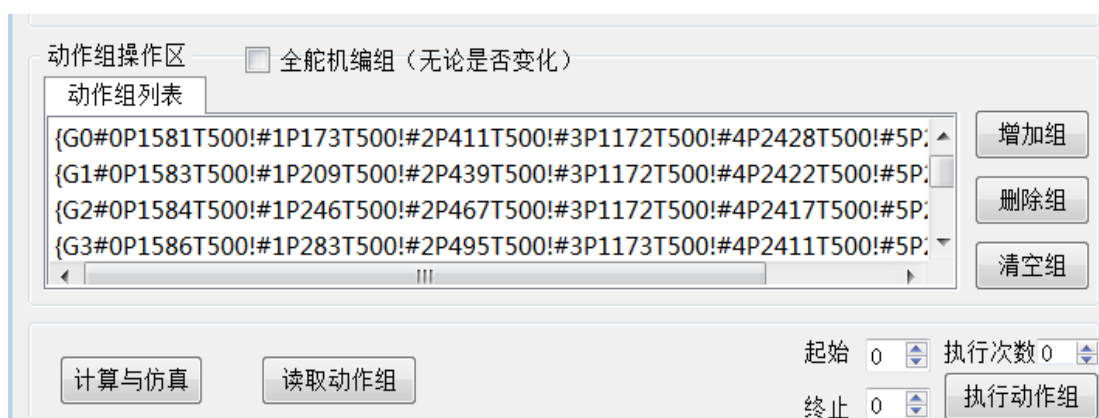
其中，pwmBarHandle(index,pwm,time)函数可以将三个参数合并为控制命令“#+index+P+pwm+T+time+!”格式。

另外，程序中还加入了键盘操控功能，通过“W”、“S”、“A”、“D”、“↑”、“↓”、“←”、“→”、“8”、“2”、“4”、“6”这12个按键，可以对6个舵机的正反方向转动进行控制。

舵机控制面板的重要性在于，它对舵机的控制提供了另外一种方式：示教输入型控制。在对舵机参数进行控制时，程序会将参数对应的控制命令实时地发送到舵机控制器。因此，舵机控制面板可以实现对各个舵机的实时控制，用户可以通过手动控制，完成机械手臂运动轨迹的命令录制。

### 6.3.3 动作组操作区

无论是利用计算模块实现编程输入型控制，还是利用舵机控制面板实现示教输入型控制，最终得到的都是关于轨迹上各个点上舵机转角值和动作时间的控制命令，如5.3节中的图5.8所示，不妨称一条关于6个舵机转角值和动作时间的控



制命令为“动作组”，表示 6 个舵机到达某一个点所需完成的动作的集合。

图 6.4 动作组操作区

在图 6.4 所示的动作组操作区中，记录了所有动作组，并标记了序号。

动作组列表中的命令，可以通过示教输入的方式获得，也可以点击“读取动作组”按钮，直接读取计算与仿真模块保存的命令。动作组操作区中也添加了“计算与仿真”按钮，可用来打开计算与仿真模块。

如要通过示教输入的方式获得，需要通过舵机控制面板，控制舵机完成一系列动作组。每完成一个动作组，点击“增加组”按钮，即可在列表中保存对应命令，并使程序开始录制下一个动作组的命令。如果录制了错误的动作组，或者需要重新录制，可以通过“删除组”按钮来删除错误动作组，也可以利用“清空组”按钮直接将列表清空。

在界面中，程序还加入了“全舵机编组”的功能，表示强制录制所有选中舵机的命令，无论舵机参数是否发生变化。这一功能的主要作用是在于，如果删除了某一动作组之前的动作组，即便这一动作组仅仅改变了若干个舵机的参数，也能保证所有舵机的动作得到正确执行。但是，这一功能导致动作组列表过于臃肿，不便于阅读，因此，该功能被设计为一个选项。

在添加了所有的动作组之后，可以通过动作组操作区右下角的执行模块来输出命令。设置了起始动作组、终止动作组和执行次数之后，点击“执行动作组”按钮，程序就会向下位机发送起点到终点的所有动作组命令，并可根据执行次数重复发送。

#### 6.3.4 控制模块上位机软件整体界面

在完成主要功能的设计之后，程序中还针对舵机的整体控制加入了一些附加功能：

- （1）舵机控制面板时间总控：对所有舵机的时间参数 **time** 进行统一设置。
- （2）舵机控制面板 **PWM** 重置：将所有舵机的 **pwm** 参数重置为 1500。
- （3）舵机复位：对所有舵机进行复位操作。

在添加完所有功能之后，对界面上的控件进行排布，得到的整体界面如图 6.5

所示：

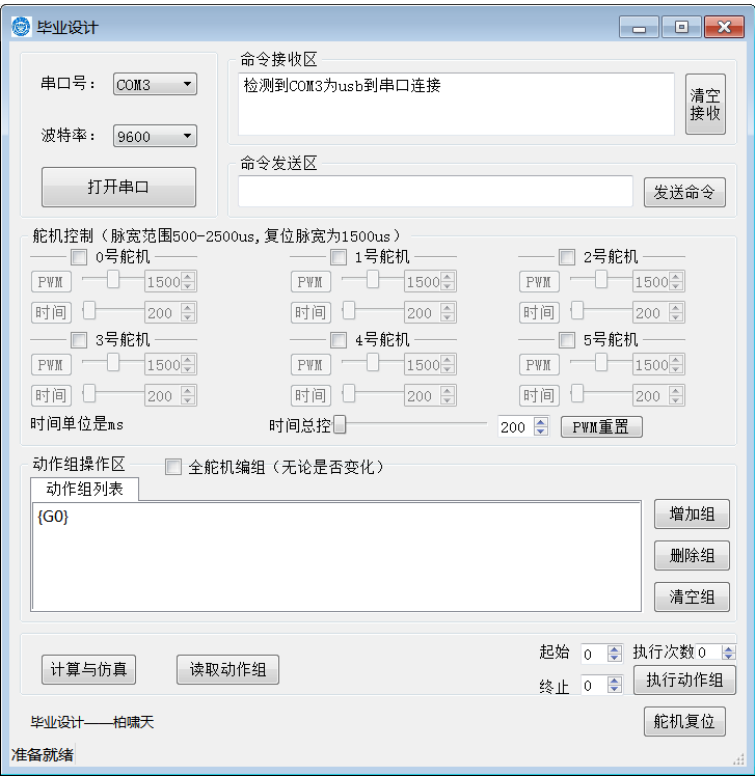


图 6.5 控制模块上位机软件整体界面

## 7 总结

本次设计针对六自由度机械手臂设计了一套伺服控制软件。

首先，我对机械手臂进行了嵌入式系统的设计，确定了机械手臂的驱动系统和微控制器，并且设计了相关的电路。这是设计伺服控制软件的硬件基础。

然后，我对机械手臂进行了运动学分析，根据机械手臂的连杆参数推导了运动学正解和逆解的公式。利用运动学逆解公式，我选择了两种各有优缺点的轨迹插值算法来对机械手臂的运动进行轨迹规划。这是伺服控制软件的理论基础。

在设计软件时，我将伺服控制软件分成了两个模块：计算与仿真模块和控制模块。这两个模块均使用 C++ 语言编写，可独立运行，也可由计算与仿真模块向控制模块传递数据。计算与仿真模块根据运动学正解和逆解公式进行机械手臂关节转角和末端位姿之间的计算转化，同时在利用 OpenGL 绘制的模型上进行仿真演示；控制模块中编写了上位机和下位机程序，实现了两者之间的通信，由上位机将舵机转角和运动时间信息转化为控制命令，传输到舵机控制器中的 STC12C5A60S2 单片机，由单片机对机械手臂各个关节上的舵机进行控制。

本着由特殊向一般推广的理念，在计算与仿真模块的设计中，我加入了自定义参数的功能，使计算和仿真的对象不局限于设计中使用的机械手臂；在控制模块的设计中，我使用了 Qt 框架，不仅简化了设计，而且具有跨平台的特性，可以在不同平台上编译，因此可以在 Linux 等机器人常用的控制平台上使用。

由于对 C++ 语言不熟悉，在设计初期遇到了不小的困难。通过学习相关教材、研究大量例程，从掌握基本语法开始一步一步完成了全部设计。在设计过程中，遇到了大量问题，例如公式推导错误，编译不通过，程序崩溃等。由于代码量比较大，各种小问题层出不穷，所幸这些问题都被一一解决，最终整个软件实现了预期的功能，还加入了一些附加的功能。

整体来看，本次设计涉及了硬件和软件设计，不仅有理论的推导，也有具体的实现，包含了嵌入式系统、51C 语言、C++ 语言、运动学、线性代数等学科知识，对我是一次十分全面的知识巩固和技能提升。



## 参 考 文 献

- [1] 张涛. 机器人引论[M]. 机械工业出版社, 2010.
- [2] 张红霞. 国内外工业机器人发展现状与趋势研究[J]. 电子世界, 2013(12):5-5.
- [3] 王宜怀. 嵌入式技术基础与实践[M]. 清华大学出版社, 2007.
- [4] 孙学智. 基于 STC12C5A60S2 智能小车控制系统设计[J]. 电子世界, 2013(22):127-128.
- [5] 蔡自兴. 机器人学基础[M]. 机械工业出版社, 2009.
- [6] 蒋新松. 机器人学导论[M]. 辽宁科学技术出版社, 1994.
- [7] 刘成良, 张凯, 曹其新,等. 机器人奇异形位分析及协调控制方法[J]. 上海交通大学学报, 2002, 36(8):1138-1142.
- [8] 陶恒铭. 六自由度工业机器人运动分析与控制技术的研究[D]. 合肥工业大学, 2014.
- [9] 胡孔林. 六自由度关节式教学机器人上位机控制软件的研究[D]. 哈尔滨工业大学, 2005.
- [10] 胡世峰, 高翔. 封装 OpenGL 功能的 C++类的设计[J]. 舰船电子工程, 2005, 25(3):76-79.
- [11] 霍亚飞. Qt 及 Qt Quick 开发实战精解[M]. 北京航空航天大学出版社, 2012.
- [12] Zhang H, Paul R P. A parallel solution to robot inverse kinematics[C]// IEEE International Conference on Robotics and Automation, 1988. Proceedings. 1988:1140-1145 vol.2.
- [13] Manocha D, Canny J F. Efficient inverse kinematics for general 6R manipulators[J]. Robotics & Automation IEEE Transactions on, 1999, 10(5):648-657.
- [14] Gasparetto A, Boscariol P, Lanzutti A, et al. Path Planning and Trajectory Planning Algorithms: A General Overview[M]// Motion and Operation Planning of Robotic Systems. Springer International Publishing, 2015:3-27.
- [15] Gasparetto A, Boscariol P, Lanzutti A, et al. Trajectory Planning in Robotics[J]. Mathematics in Computer Science, 2012, 6(3):269-279.
- [16] Lippman S B, Lajoie J, Moo B E. C++ Primer, 5th Edition[J]. 2013.
- [17] Ezust A. An Introduction to Design Patterns in C++ with Qt[M]. Prentice Hall, 2011.

## 附录 A：部分主要函数代码

下位机分析控制命令函数代码：

```
void handle_uart(void)
{
    static unsigned int index=0, pwm=0, time=0, i;

    unsigned int len

    len = strlen(uart_receive_buf);    //数组 uart_receive_buf 用来储存接收到的命令

    for(i=0;i<len;i++)
    {
        if(uart_receive_buf[i] == '#')
        {
            i++;

            while(uart_receive_buf[i] != 'P')
            {
                index = index*10 + uart_receive_buf[i] - '0'; //得到舵机编号

                i++;
            }

            i--;

        }

        else if(uart_receive_buf[i] == 'P')
        {
            i++;

            while(uart_receive_buf[i] != 'T')
            {
                pwm = pwm*10 + uart_receive_buf[i] - '0';    //得到 pwm 脉宽
```

```

        i++;
    }
    i--;
}
else if(uart_receive_buf[i] == 'T')
{
    i++;
    while(uart_receive_buf[i] != '!')
    {
        time = time*10 + uart_receive_buf[i] - '0';    //得到动作时间
        i++;
    }
    sprintf(cmd_return, "%dP%dT%d!", index, pwm, time);
    uart_send_str(cmd_return);    //发送成功反馈
    index = pwm = time = 0;
}
else uart_send_str("err!");    //发送错误反馈
return;
}

uart_open();    //打开串口
}

```

机械手臂仿真模型绘制函数代码:

```

void CDrawArm::DrawArmBody(float OS, float OL, float OU, float OR, float OB, float OT)
{
    float a;    //比例系数

```

```

a=scaleview;

//立体材质的定义

//红色

GLfloat red_ambient[] = { 0.2f, 0.0f, 0.0f };    //环境光
GLfloat red_diffuse[] = { 0.5f, 0.0f, 0.0f };    //扩散光
GLfloat red_specular[] = { 0.7f, 0.6f, 0.6f };    //镜面光
GLfloat red_shininess[] = { 32.0f };            //反射强度

//.....其他颜色.....//

//黑色固定基台部

//立体材质的定义：黑

glMaterialfv( GL_FRONT, GL_AMBIENT, black_ambient);
glMaterialfv( GL_FRONT, GL_DIFFUSE, black_diffuse );
glMaterialfv( GL_FRONT, GL_SPECULAR,black_specular);
glMaterialfv( GL_FRONT, GL_SHININESS,black_shininess );

glPushMatrix();

    SolidCube(32*a,32*a,10*a);        //方形底座

glPopMatrix();

glPushMatrix();

glTranslated(11*a,21*a,0);

    SolidXieCube(10*a,10*a,10*a);        //底座旁四个斜面

glPopMatrix();

glPushMatrix();

glTranslated(11*a,-21*a,0);

glRotatef(180,0.0,0.0,1.0);

    SolidXieCube(10*a,10*a,10*a);

glPopMatrix();

glPushMatrix();

glTranslated(-11*a,21*a,0);

    SolidXieCube(10*a,10*a,10*a);

```

```
        glPopMatrix();

//.....其他形状.....//

//关节 1 旋转部分

glRotated( OS, 0, 0, 1 );    //S 为关节 1 转角参数

//立体材质的定义：蓝色

glMaterialfv( GL_FRONT, GL_AMBIENT, blue_ambient);

glMaterialfv( GL_FRONT, GL_DIFFUSE, blue_diffuse );

glMaterialfv( GL_FRONT, GL_SPECULAR,blue_specular);

glMaterialfv( GL_FRONT, GL_SHININESS,blue_shininess );

glPushMatrix();

        SolidCylinder(17 * a, 8 * a, 30);    //关节 1 下方圆盘

glPopMatrix();

glTranslated(0.0, 0.0, 8 * a);    //把原点向上挪

glPushMatrix();

glTranslated(0, 0, 0);

        SolidCube(9.2*a, 5 * a, 20 * a);    //画竖着的大长方体托

glPopMatrix();

glPushMatrix();

glTranslated(-12 * a, 3 * a, -3 * a);

glRotated(-90, 0, 1, 0);

glRotated(90, 0, 0, 1);

        SolidXieCube(5 * a, 6 * a, 3.2*a);    //链接斜三角

glPopMatrix();

glTranslated(0, 0.0, 20 * a);    //把原点往上挪

glPushMatrix();

glRotated(-90, 1, 0, 0);

        SolidCylinder(9 * a, 5 * a, 30);    //横放的圆片

glPopMatrix();

//立体材质的定义：黑色
```

```

glMaterialfv(GL_FRONT, GL_AMBIENT, black_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, black_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, black_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, black_shininess);

glPushMatrix();

glTranslated(0, -14 * a, 0.0);

glRotated(-90, 1, 0, 0);

    SolidCylinder(6 * a, 14.1*a, 8);    //两个黑色圆柱表示舵机

glPopMatrix();

glPushMatrix();

glTranslated(0, -18 * a, 0.0);

glRotated(-90, 1, 0, 0);

    SolidCylinder(4.5*a, 4.1*a, 8);

glPopMatrix();

//.....//

glRotated( OL, 0, 0, 1 );    //L 为关节 2 转角参数

//.....//

glRotated( OU, 0, 0, 1 );    //U 为关节 3 转角参数

//.....//

glRotated( OR, 0, 0, 1 );    //R 为关节 4 转角参数

//.....//

glRotated( OB, 0, 0, 1 );    //B 为关节 5 转角参数

//.....//

glRotated( OT, 0, 0, 1 );    //T 为关节 6 转角参数

//.....//

}

```

求运动学正解函数代码：

```

void CBai_SimulationDlg::RunZheng()
{
    double s1,s2,s3,s4,s5,s6,c1,c2,c3,c4,c5,c6,s23,c23;

    s1=sin(ino[1]);           //ino[7]数组存放界面中输入的转角值
    s2=sin(ino[2]);
    s3=sin(ino[3]);
    s4=sin(ino[4]);
    s5=sin(ino[5]);
    s6=sin(ino[6]);
    c1=cos(ino[1]);
    c2=cos(ino[2]);
    c3=cos(ino[3]);
    c4=cos(ino[4]);
    c5=cos(ino[5]);
    c6=cos(ino[6]);

    s23=s2*c3+s3*c2;
    c23=c2*c3-s2*s3;

    //求转换矩阵各个元素的值

    nx = c1*c23*(c4*c5*c6-s4*s6) - c1*s23*s5*c6 + s1*(s4*c5*c6 + c4*s6);
    ny = -1*c1*(s4*c5*c6 + c4*s6) + s1*c23*c4*c5*c6 - s1*s23*s5*c6 - s1*c23*s4*s6;
    nz = -1*s23*c4*c5*c6 - c23*s5*c6 + s23*s4*s6;

    ox = -1*s1*(s4*c5*s6 - c4*c6) - c1*c23*c4*c5*s6 + c1*s23*s5*s6 - c1*c23*s4*c6;
    oy = c1*(s4*c5*s6 - c4*c6) - s1*c23*c4*c5*s6 + s1*s23*s5*s6 - s1*c23*s4*c6;
    oz = s23*c4*c5*s6 + c23*s5*s6 + s23*s4*c6;

    ax = -c1*c23*c4*s5 - c1*s23*c5 - s1*s4*s5;
    ay = -s1*c23*c4*s5 - s1*s23*c5 + c1*s4*s5;
    az = s23*c4*s5 - c23*c5;

    px = -1 * d6*c1*c23*c4*s5 - d6*s1*s4*s5 - d6*c1*s23*c5 - d5*c1*s23 - d4*c1*s23 +
    a2*c1*c2;

```

```

py = -1 * d6*s1*c23*c4*s5 + d6*c1*s4*s5 - d6*s1*s23*c5 - d5*s1*s23 - d4*s1*s23 +
a2*s1*c2;

pz = d6*s23*c4*s5 - d6*c23*c5 - d5*c23 - d4*c23 - a2*s2 + d2+d1;

}

```

“正解计算”按钮函数代码：

```

void CBai_SimulationDlg::OnRunZhengJie()
{
    UpdateData(true);    //将界面输入值传入关联变量
    if ((int)m_ino5 % 180 == 0)
    {
        MessageBox("sin(  $\theta_5$  ) 为 0，无法确定  $\theta_4$  与  $\theta_6$  的值", "警告", 0);
    }
    ino[1]=m_ino1*PI/180;
    ino[2]=m_ino2*PI/180;
    ino[3]=m_ino3*PI/180;
    ino[4]=m_ino4*PI/180;
    ino[5]=m_ino5*PI/180;
    ino[6]=m_ino6*PI/180;
    CS_Left[0]=m_ino1;    //CS_Left[6]将转角参数传入机械手臂绘制函数
    CS_Left[1]=m_ino2;
    CS_Left[2]=m_ino3;
    CS_Left[3]=m_ino4;
    CS_Left[4]=m_ino5;
    CS_Left[5]=m_ino6;
    m_pDisplay->OnPaint();    //对输入的转角值进行仿真
    RunZheng();    //求运动学正解
    m_nx=nx;
}

```



```

    m_ny=ny;

    m_nz=nz;

    m_ox=ox;

    m_oy=oy;

    m_oz=oz;

    m_ax=ax;

    m_ay=ay;

    m_az=az;

    m_px=px;

    m_py=py;

    m_pz=pz;

    UpdateData(false);    //将位姿矩阵显示在界面上
}

```

求运动学逆解函数代码：

```

void CBai_SimulationDlg::RunNijie()
{
    double s1[3],s2[5],s3[9],s4[9],s5[9],c1[3],c2[5],c3[9],c4[9],c5[9],s23[9],c23[9],c6[9],s6[9];

    double t11,t12,k2[3], k23[9],t21, t22[3],o23[9];

    //计算关节 1 转角

    outo[1][1] = atan2(ay*d6 - py, ax*d6 - px);

    c1[1] = cos(outo[1][1]);

    s1[1] = sin(outo[1][1]);

    outo[2][1] = outo[1][1];

    outo[3][1] = outo[1][1];

    outo[4][1] = outo[1][1];
}

```

```

outo[5][1] = atan2(-ay*d6 + py, -ax*d6 + px);

c1[2] = cos(outo[5][1]);

s1[2] = sin(outo[5][1]);

outo[6][1] = outo[5][1];

outo[7][1] = outo[5][1];

outo[8][1] = outo[5][1];

//计算关节 2 转角

t21 = 2 * a2*(pz - d1 - d6*az - d2);

t22[1] = 2 * a2*(px*c1[1] + py*s1[1] - d6*ax*c1[1] - d6*ay*s1[1]);

t22[2] = 2 * a2*(px*c1[2] + py*s1[2] - d6*ax*c1[2] - d6*ay*s1[2]);

k2[1] = (t21*t21) / (4 * a2*a2) + (t22[1] * t22[1]) / (4 * a2*a2) + a2*a2 - (d4 + d5)*(d4 + d5);

k2[2] = (t21*t21) / (4 * a2*a2) + (t22[2] * t22[2]) / (4 * a2*a2) + a2*a2 - (d4 + d5)*(d4 + d5);

outo[1][2] = atan2(t22[1], t21) - atan2(k2[1], sqrt(t21*t21 + t22[1] * t22[1] - k2[1] * k2[1]));

outo[2][2] = outo[1][2];

c2[1] = cos(outo[1][2]);

s2[1] = sin(outo[1][2]);

outo[3][2] = atan2(t22[1], t21) - atan2(k2[1], -sqrt(t21*t21 + t22[1] * t22[1] - k2[1] * k2[1]));

outo[4][2] = outo[3][2];

c2[2] = cos(outo[3][2]);

s2[2] = sin(outo[3][2]);

outo[5][2] = atan2(t22[2], t21) - atan2(k2[2], sqrt(t21*t21 + t22[2] * t22[2] - k2[2] * k2[2]));

outo[6][2] = outo[5][2];

c2[3] = cos(outo[5][2]);

s2[3] = sin(outo[5][2]);

outo[7][2] = atan2(t22[2], t21) - atan2(k2[2], -sqrt(t21*t21 + t22[2] * t22[2] - k2[2] * k2[2]));

outo[8][2] = outo[7][2];

c2[4] = cos(outo[7][2]);

s2[4] = sin(outo[7][2]);

```

```

//计算关节 3 转角

k23[1] = px*c1[1] + py*s1[1] - d6*(ax*c1[1] + ay*s1[1]) - a2*c2[1] + d1 + d2 - a2*s2[1] +
d6*az - pz;

k23[2] = px*c1[1] + py*s1[1] - d6*(ax*c1[1] + ay*s1[1]) - a2*c2[2] + d1 + d2 - a2*s2[2] +
d6*az - pz;

k23[3] = px*c1[2] + py*s1[2] - d6*(ax*c1[2] + ay*s1[2]) - a2*c2[3] + d1 + d2 - a2*s2[3] +
d6*az - pz;

k23[4] = px*c1[2] + py*s1[2] - d6*(ax*c1[2] + ay*s1[2]) - a2*c2[4] + d1 + d2 - a2*s2[4] +
d6*az - pz;

o23[1] = atan2(d4 + d5, d4 + d5) - atan2(k23[1], sqrt(2 * (d4 + d5)*(d4 + d5) - k23[1] *
k23[1]));

o23[2] = atan2(d4 + d5, d4 + d5) - atan2(k23[1], -sqrt(2 * (d4 + d5)*(d4 + d5) - k23[1] *
k23[1]));

o23[3] = atan2(d4 + d5, d4 + d5) - atan2(k23[2], sqrt(2 * (d4 + d5)*(d4 + d5) - k23[2] *
k23[2]));

o23[4] = atan2(d4 + d5, d4 + d5) - atan2(k23[2], -sqrt(2 * (d4 + d5)*(d4 + d5) - k23[2] *
k23[2]));

o23[5] = atan2(d4 + d5, d4 + d5) - atan2(k23[3], sqrt(2 * (d4 + d5)*(d4 + d5) - k23[3] *
k23[3]));

o23[6] = atan2(d4 + d5, d4 + d5) - atan2(k23[3], -sqrt(2 * (d4 + d5)*(d4 + d5) - k23[3] *
k23[3]));

o23[7] = atan2(d4 + d5, d4 + d5) - atan2(k23[4], sqrt(2 * (d4 + d5)*(d4 + d5) - k23[4] *
k23[4]));

o23[8] = atan2(d4 + d5, d4 + d5) - atan2(k23[4], -sqrt(2 * (d4 + d5)*(d4 + d5) - k23[4] *
k23[4]));

//判断  $\theta_{23}$  是否正确

if((sin(o23[1])-(c1[1]*px+s1[1]*py-d6*(c1[1]*ax+s1[1]*ay)-a2*c2[1])/(-d4-d5))<0.01)
{
    o23[2] = o23[1];
}
else o23[1] = o23[2];

```

```

    if ((sin(o23[3]) - (c1[1] * px + s1[1] * py - d6*(c1[1] * ax + s1[1] * ay) - a2*c2[2]) / (-d4 -
d5))<0.01)
    {
        o23[4] = o23[3];
    }
    else o23[3] = o23[4];

    if ((sin(o23[5]) - (c1[2] * px + s1[2] * py - d6*(c1[2] * ax + s1[2] * ay) - a2*c2[3]) / (-d4 -
d5))<0.01)
    {
        o23[6] = o23[5];
    }
    else o23[5] = o23[6];

    if ((sin(o23[7]) - (c1[2] * px + s1[2] * py - d6*(c1[2] * ax + s1[2] * ay) - a2*c2[4]) / (-d4 -
d5))<0.01)
    {
        o23[8] = o23[7];
    }
    else o23[7] = o23[8];

    s23[1] = sin(o23[1]);
    c23[1] = cos(o23[1]);
    s23[2] = sin(o23[2]);
    c23[2] = cos(o23[2]);
    s23[3] = sin(o23[3]);
    c23[3] = cos(o23[3]);
    s23[4] = sin(o23[4]);
    c23[4] = cos(o23[4]);

    //.....//

    s23[7] = sin(o23[7]);
    c23[7] = cos(o23[7]);
    s23[8] = sin(o23[8]);

```

```

c23[8] = cos(o23[8]);

outo[1][3] = o23[1]-outo[1][2];

outo[2][3] = o23[2] -outo[1][2];

//.....//

outo[7][3] = o23[7] -outo[7][2];

outo[8][3] = o23[8] -outo[7][2];


//计算关节 4 转角

outo[1][4] = atan2( -1 * ax*s1[1] + ay*c1[1],-ax*c1[1] * c23[1] - ay*s1[1] * c23[1] +
az*s23[1]);

outo[2][4] = atan2(1 * ax*s1[1] - ay*c1[1], ax*c1[1] * c23[2] + ay*s1[1] * c23[2] - az*s23[2]);

//.....//

outo[7][4] = atan2(-1 * ax*s1[2] + ay*c1[2], -ax*c1[2] * c23[7] - ay*s1[2] * c23[7] +
az*s23[7]);

outo[8][4] = atan2(1 * ax*s1[2] - ay*c1[2], ax*c1[2] * c23[8] + ay*s1[2] * c23[8] - az*s23[8]);

c4[1] = cos(outo[1][4]);

s4[1] = sin(outo[1][4]);

//.....//

c4[8] = cos(outo[8][4]);

s4[8] = sin(outo[8][4]);


//计算关节 5 转角

s5[1] = -ax*(c1[1] * c4[1] * c23[1] + s1[1] * s4[1]) - ay*(s1[1] * c23[1] * c4[1] - c1[1] * s4[1])
+ az*s23[1] * c4[1];

c5[1] = -ax*c1[1] * s23[1] - ay*s1[1] * s23[1] - az*c23[1];

//.....//

s5[8] = -ax*(c1[2] * c4[8] * c23[8] + s1[2] * s4[8]) - ay*(s1[2] * c23[8] * c4[8] - c1[2] * s4[8])
+ az*s23[8] * c4[8];

c5[8] = -ax*c1[2] * s23[8] - ay*s1[2] * s23[8] - az*c23[8];

outo[1][5] = atan2(s5[1], c5[1]);

```

```

//.....//

outo[8][5] = atan2(s5[8], c5[8]);

//计算关节 6 转角

c6[1] = nx*(c5[1] * (c1[1] * c23[1] * c4[1] + s1[1] * s4[1]) - c1[1] * s23[1] * s5[1]) + ny*(c5[1]
* (s1[1] * c23[1] * c4[1] - c1[1] * s4[1]) - s1[1] * s23[1] * s5[1]) - nz*(s23[1] * c4[1] * c5[1] + s5[1]
* c23[1]);

s6[1] = -ox*(c5[1] * (c1[1] * c23[1] * c4[1] + s1[1] * s4[1]) - c1[1] * s23[1] * s5[1]) - oy*(c5[1]
* (s1[1] * c23[1] * c4[1] - c1[1] * s4[1]) - s1[1] * s23[1] * s5[1]) + oz*(s23[1] * c4[1] * c5[1] + s5[1]
* c23[1]);

//.....//

c6[8] = nx*(c5[8] * (c1[2] * c23[8] * c4[8] + s1[2] * s4[8]) - c1[2] * s23[8] * s5[8]) + ny*(c5[8]
* (s1[2] * c23[8] * c4[8] - c1[2] * s4[8]) - s1[2] * s23[8] * s5[8]) - nz*(s23[8] * c4[8] * c5[8] + s5[8]
* c23[8]);

s6[8] = -ox*(c5[8] * (c1[2] * c23[8] * c4[8] + s1[2] * s4[8]) - c1[2] * s23[8] * s5[8]) - oy*(c5[8]
* (s1[2] * c23[8] * c4[8] - c1[2] * s4[8]) - s1[2] * s23[8] * s5[8]) + oz*(s23[8] * c4[8] * c5[8] + s5[8]
* c23[8]);

outo[1][6] = atan2(s6[1], c6[1]);

//.....//

outo[8][6] = atan2(s6[8], c6[8]);

//调整转角值为-180-180

for (int i = 1; i < 9; i++)

{

    for (int j = 1; j < 7; j++)

    {

        if (outo[i][j] * 180 / PI < -180)

        {

            outo[i][j] = outo[i][j] + 2 * PI;

        }

        else if (outo[i][j] * 180 / PI > 180)

        {

```

```
        outo[i][j] = outo[i][j] - 2 * PI;
    }
}
}
```

角度范围判断函数代码：

CString CBai\_SimulationDlg::DetAngScal(double aa, double bb, double cc, double dd, double ee, double ff)

```
{
    int i=0;
    CString str=" ";
    CString str0 = " ";
    if (aa<m_lmt1||aa>m_lmt2)
    {
        i=1;
        str+=" 1 ";
    }
    if (bb<m_lmt3||bb>m_lmt4)
    {
        i=1;
        str+=" 2 ";
    }

    //.....//

    if (ff<m_lmt11 || ff>m_lmt12)
    {
```

```
        i = 1;

        str += " 0 6";

    }

    if (i)

    {

        str+="超出其角度范围";

    }

    if(IsDlgButtonChecked(IDC_ANG))        //判断是否选择“角度限制”选项

        return str;

    else return str0;

}
```

“逆解计算”按钮函数代码：

```
void CBai_SimulationDlg::OnRunNi()

{

    UpdateData(true);        //将界面上数据传入关联变量

    nx=m_nx;

    ny=m_ny;

    nz=m_nz;

    ox=m_ox;

    oy=m_oy;

    oz=m_oz;

    ax=m_ax;

    ay=m_ay;

    az=m_az;

    px=m_px;

    py=m_py;
```



```

pz=m_pz;

m_explain1="";

m_explain2="";

m_explain3="";

m_explain4="";

m_explain5="";

m_explain6="";

m_explain7="";

m_explain8="";

RunNijie();          //求运动学逆解

//将弧度制角度值转化为角度制角度值

m_OutO11=outo[1][1]*180/PI;

m_OutO12=outo[1][2]*180/PI;

m_OutO13=outo[1][3]*180/PI;

m_OutO14=outo[1][4]*180/PI;

m_OutO15=outo[1][5]*180/PI;

m_OutO16=outo[1][6]*180/PI;

//.....//

m_OutO81=outo[8][1]*180/PI;

m_OutO82=outo[8][2]*180/PI;

m_OutO83=outo[8][3]*180/PI;

m_OutO84=outo[8][4]*180/PI;

m_OutO85=outo[8][5]*180/PI;

m_OutO86=outo[8][6]*180/PI;

//判断转角是否处于范围内

m_explain1+=DetAngScal(m_OutO11,m_OutO12,m_OutO13,m_OutO14,m_OutO15,m_OutO
16);

m_explain2+=DetAngScal(m_OutO21,m_OutO22,m_OutO23,m_OutO24,m_OutO25,m_OutO
26);

//.....//

```

```
m_explain8+=DetAngScal(m_OutO81,m_OutO82,m_OutO83,m_OutO84,m_OutO85,m_OutO86);  
  
UpdateData(false);    //在界面上显示逆解  
  
}
```