工控干货丨运动控制器控制SCARA机械手的方法

来源:雷赛智能 日期:2018-12-06

本文主要介绍机械手自动装配插头系统的运动控制算法和编程要点。

本文主要介绍机械手自动装配插头系统的运动控制算法和编程要点。

该系统采用雷赛SMC606 运动控制器和L5 系列交流伺服电机控制、驱动国产SCARA 机械手。如图1 所示，气动手爪安装在机械手前段，其中心点不在Z 轴的轴线上。

该系统采用嵌入式工控机为主控单元，用C#语言编程，完成机器视觉定位、机械手运动轨迹规划的工作；并通过以太网向SMC606 运动控制器实时发出运动控制指令，实现插头装配工作。

和普通工控机+运动控制卡的方案相比，该系统具有稳定可靠、性价比高等特点。

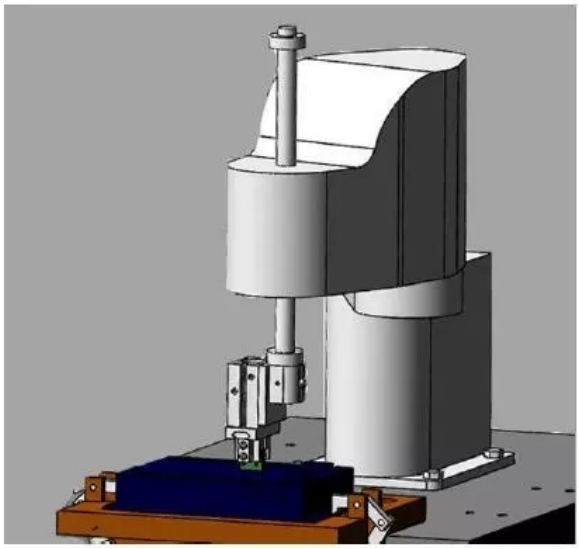

图 1. SCARA 机械手及手爪 雷赛控制技术

一. SCARA 机械手的运动学方程

1. SCARA 机械手的正向运动学的解

正向运动学是已知各关节的参数，求手爪的位置和姿态。

即：已知各杆长L1、L2、L3，各关节转角

θ 1、θ 2、θ 3，求手爪在机座坐标系中的解。

如图2 所示，坐标系{0}是机座坐标系。机座坐标系又称为全局坐标系，是一个固定坐标系，它是计算机械手运动的基础。通常其原点设在机器手底部中心。

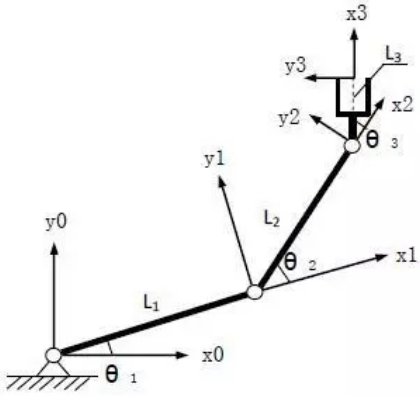

图 2. SCARA 机械手的坐标系 雷赛控制技术

坐标系{1}是连杆L1 的坐标系。坐标系{2}是连杆L2 的坐标系。

坐标系{3}是工具坐标系。工具坐标系是表示工具中心点TCP（Tool Center Point）的位置和工具姿态的直角坐标系。工具坐标系是一个活动的坐标系，使用工具坐标系便于编写手爪抓取工件的程序。

通过连杆坐标系之间的齐次变换，可得到坐标系{3}与坐标系{0}的变换关系如下：

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1+\theta_2+\theta_3) & -\sin(\theta_1+\theta_2+\theta_3) & p_x \\ \sin(\theta_1+\theta_2+\theta_3) & \cos(\theta_1+\theta_2+\theta_3) & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_3 \\ y_3 \\ 1 \end{bmatrix}$$

雷赛控制技术

其中：

$$p_x = L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) + L_3\cos(\theta_1+\theta_2+\theta_3)$$
$$p_y = L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) + L_3\sin(\theta_1+\theta_2+\theta_3)$$

雷赛控制技术

即：

$$\begin{cases} x_0 = \cos(\theta_1+\theta_2+\theta_3)x_3 - \sin(\theta_1+\theta_2+\theta_3)y_3 + p_x \\ y_0 = \sin(\theta_1+\theta_2+\theta_3)x_3 + \cos(\theta_1+\theta_2+\theta_3)y_3 + p_y \end{cases}$$

雷赛控制技术

2. SCARA 机械手的反向运动学的解

反向运动学是已知手爪的位置和姿态，求各关节的转角。

即：已知手爪中心点P3在机座坐标系的位置P3（x0，y0）和转角 。求：各关节转角θ 1、θ 2、θ 3。如图3 所示。

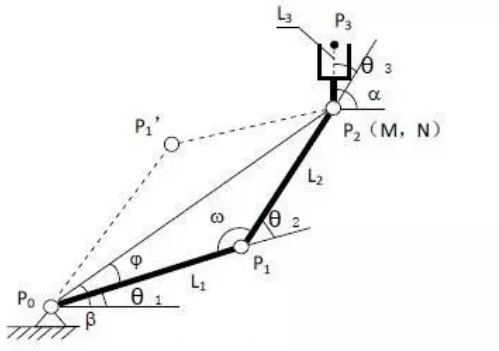

图 3. SCARA 机械手的反向运动学的几何解 雷赛控制技术

设点P2 在机座坐标系的位置为P2（M，N）。

显然，

$$M = x_0 - L_3\cos(\alpha)$$
$$N = y_0 - L_3\sin(\alpha)$$

雷赛控制技术

在三角形$\Delta P_0P_1P_2$中，有余弦定理：$M^2+N^2=L_1^2+L_2^2-2L_1L_2\cos(\omega)$

且：$\omega=180°-\theta_2$

故：$\theta_2=\arccos\frac{M^2+N^2-L_1^2-L_2^2}{2L_1L_2}$ （1）

在三角形$\Delta P_0P_1P_2$中，还有余弦定理：$L_2^2=M^2+N^2+L_1^2-2L_1\sqrt{M^2+N^2}\cos(\varphi)$

且：$\tan(\beta)=\frac{N}{M}$

$\theta_1=\beta-\varphi$

故：$\theta_1=\arctan(\frac{N}{M})-\arccos(\frac{M^2+N^2+L_1^2-L_2^2}{2L_1\sqrt{M^2+N^2}})$ （2）

则：$\theta_3=\alpha-\theta_1-\theta_2$ （3）

从图3 中可看成，P1 点的位置有2 个解，即P1 和P1’。

解一为式（2）、式（1）、式（3）。

显然，解二为：

$$\begin{cases}\theta_1=\beta+\varphi\\ \theta_2=-\theta_2\\ \theta_3=\alpha-\theta_1-\theta_2\end{cases}$$

二. 最佳路径的选择

1. 选择终点的解

如图4 所示，终点位置连杆的参数有2 个解，选用哪一个为好？需要设计一个判断准则。

因为连杆L2、Z 轴和手爪都安装在连杆L1 上，且L1 比L2 长，故其负载重、运动覆盖的面积大，所以，应该让L2 多运动、让L1 少运动。

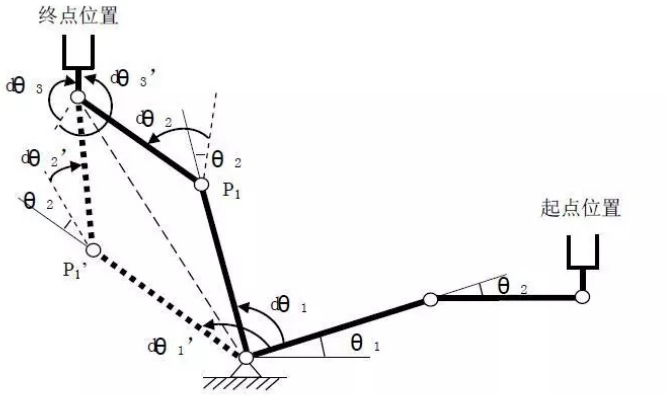

图 4. SCARA 机械手运动路径的选择

设解一的加权转角和为：

$$total1=|d\theta_1|+0.6\times|d\theta_2|$$

设解二的加权转角和为：

$$total2=|d\theta_1'|+0.6\times|d\theta_2'|$$

其中：0.6 为连杆L2 的加权系数。

比较total1 和total2，那个小就用那个解。如图5 所示，选择解1；如图6 所示，选择解2。

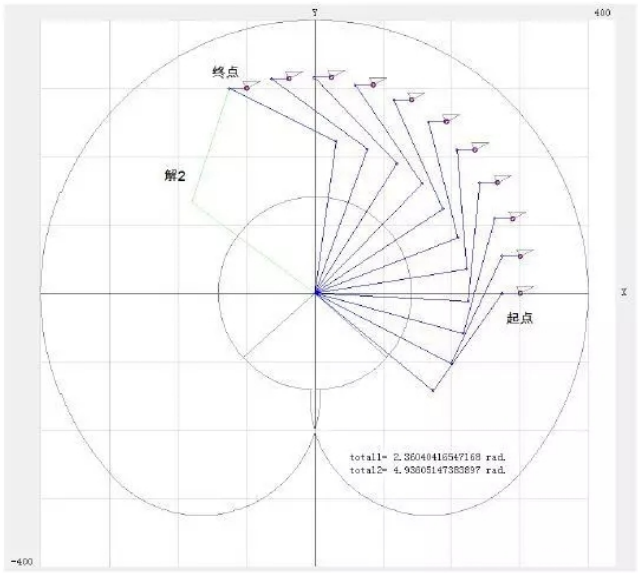

图 5. 终点选择解 1 的示例 雷赛控制技术

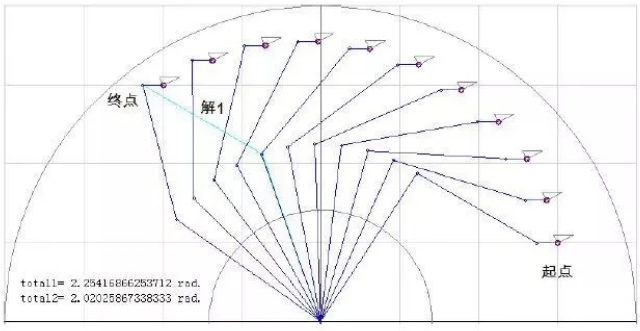

图 6. 终点选择解 2 的示例 雷赛控制技术

2. 选择手爪旋转方向

由于手爪上连接有气管，旋转角度有限制。通常手爪的旋转角θ3小于360度。

如图4所示，手爪的旋转角可是d θ3也可以是d θ3’。选择手爪旋转方向的判定准则如下：

1）在d θ3和d θ3’中选择绝对值小者α；

2）如果|α sum+α|<360o，则转动α；然后，α sum=α sum+α。式中α sum为手爪累计旋转角。

3）如果|α sum+α|≥360o，则 为d θ3和d θ3’中的绝对值大者；然后，α sum=α sum+α。

如图7所示为正常情况下，d θ3选择旋转角度小的方向。但图8所示，d θ3选择了旋转角度大的方向。因为此时如果选择角度小的方向，累计转角就会大于360o。

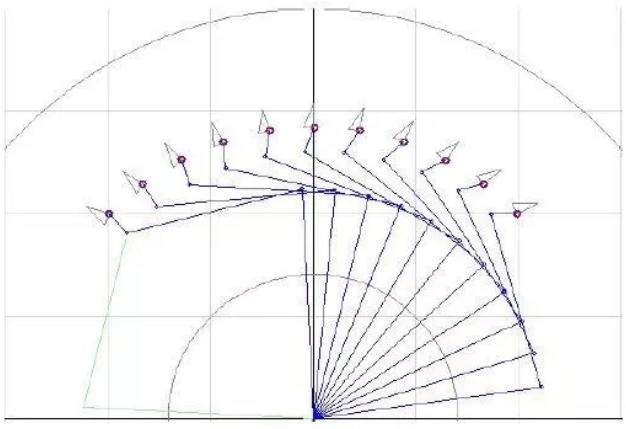

图 7. 正常情况下手爪旋转选取角度小的方向 雷赛控制技术

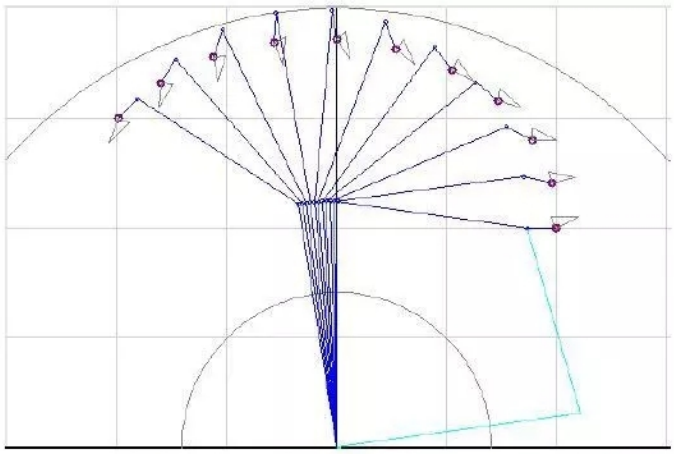

图 8. 为避免手爪旋转角累计超过 360°，手爪旋转方向和正常情况相反

三. SCARA机械手结构及参数

1. Z轴及关节J4的结构

SCARA机械手外形如图9所示。

Z轴及关节J4由一根丝杆花键轴构成， 如图10所示。

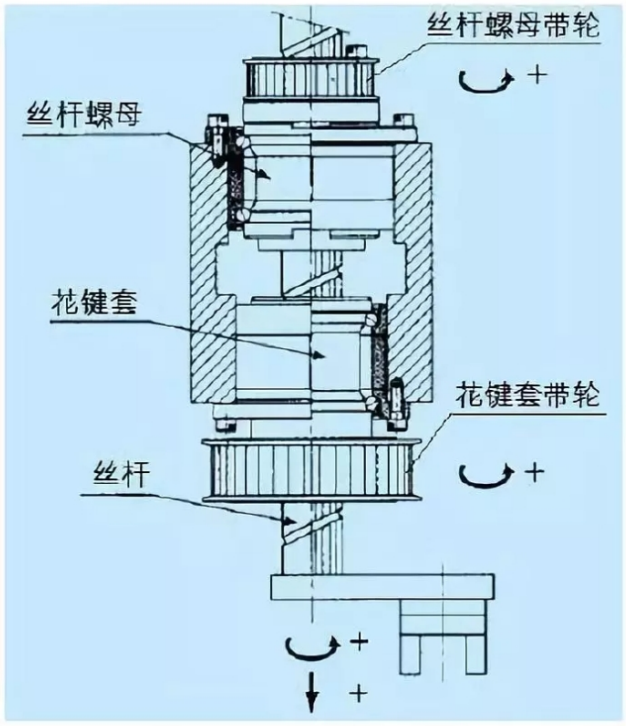

图 10. 丝杆花键轴的结构

雷赛控制技术

丝杆花键轴运动有以下特点：

1）当电机3控制丝杆螺母带轮左右旋转时，Z轴可上下运动。

2）当电机4控制花键套带轮左右旋转时，关节J4（手爪）可左右旋转，同时，Z轴上下运动。即：关节J4的旋转运动和Z轴的上下运动是耦合在一起的。

3）关节J4（手爪）要独立进行旋转运动，电机4和电机3必须联动。联动方式为：电机3的转角与电机4的转角相同，但方向相反。使Z轴向上和向下的位移量抵消。

2. 计算各轴的脉冲当量

已知该机械手的各轴参数如下：

关节J1和J2的减速器的减速比均为1:50；关节J4的减速器的减速比为1:20。Z轴减速器的减速比为1:2，丝杆导程为16mm。

机械手上的4个电机转动一周的脉冲数都为5000个。

故：控制J1电机运动的脉冲当量 Equival1 = 5000×50/360 = 694.44444 脉冲/度。

控制J2电机运动的脉冲当量Equival2 = 694.44441 脉冲/度。

控制J4电机运动的脉冲当量Equival4 = 5000×20/360 = 277.77778 脉冲/度。

控制Z电机运动的脉冲当量Equival3 = 5000×2/16 = 625.0 脉冲/毫米。

四. SMC606控制SCARA机械手的程序

1. 控制界面的设计

SMC606控制SCARA机械手运动分为两类：手动控制和自动控制。手动控制又分XYZ坐标下的位置控制和3个关节旋转轴的位置控制。如图10所示。

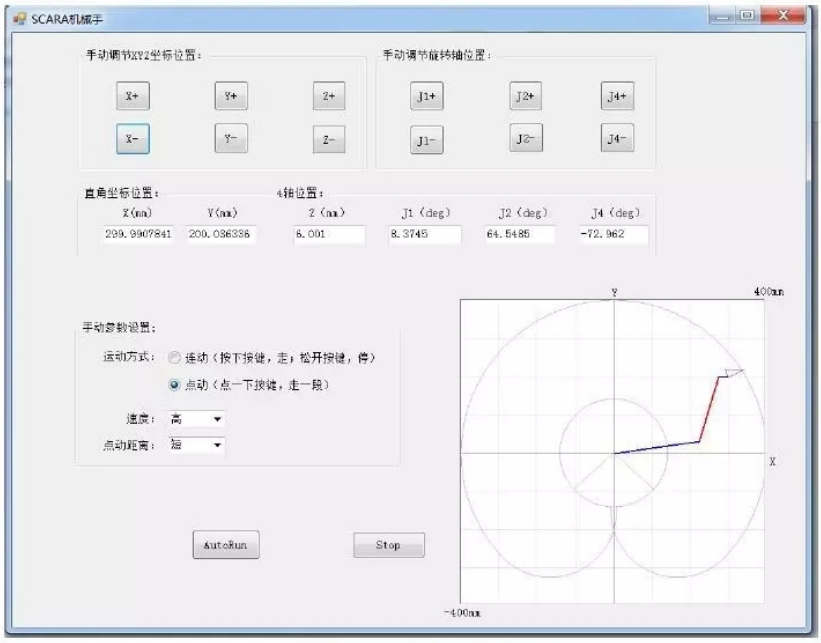

图 10. SCARA 机械手控制界面 雷赛控制技术

手动控制有2种方式：连动和点动。连动即连续运动，

按键按下后运动开始，按键抬起后运动停止。点动即点击一次按键，相应的电机运动一段距离。且每次运动的速度和距离也可选择为高、低和长、中、短。

2. 程序的初始化

1）在SMC606运动控制器资料光盘中的动态库目录下找到文件LTSMC.CS、LTSMC.dll，并拷贝到C#工程文件目录中的BIN/DEBUG内。

2）用鼠标右键点击“解决方案资源管理器”的工程名，然后点击“添加”、“现有项”，找到上述目录中的LTSMC.cs件，添加到工程中。

3）在Form1.cs头文件中添加SMC606运动控制器的命名空间 using Leadshine;

4）在Form1_Load函数中连接SMC606运动控制器。指令如下所示。

```
……..
short res = LTSMC.smc_board_init(0, 2, "192.168.5.11", 0);      // SMC606的IP地址为192.168.5.11
if (res != 0)
MessageBox.Show(string.Format("连接失败！错误码：{0}", res), "错误提示：");
else
{
MessageBox.Show("与机械手连接成功！"); LTSMC.smc_set_equiv(CardNo, 0, Equival0);      // 设置4个轴的脉冲当量
      LTSMC.smc_set_equiv(CardNo,      1,      Equival1);      LTSMC.smc_set_equiv(CardNo,      2,      Equival2);
LTSMC.smc_set_equiv(CardNo, 3, Equival3);
  }
  ……..
```

注意：与SMC606连接的PC机的IP地址前3段必须为192.168.5。

3. 手动控制的程序

1）单轴运动的点动程序

关节J1、J2和Z轴的点动过程都是单轴运动，采用点位运动指令Pmove很容易实现此功能。控制关节J1正方向点动的程序如下所示，关节J2和Z轴的点动程序类似。

```
private void buttonJ1p_Click(object sender, EventArgs e)      // 按键“J1+”的Click事件被触发
{
Axis = 0;      // 0号轴运动
if (LTSMC.smc_check_done(CardNo, Axis) == 1)      // 如果电机停止
{
```

```
LTSMC.smc_set_profile_unit(CardNo, Axis, Min_Vel, Max_Vel0, Tacc, Tdec, Stop_Vel);      // 设置速度曲线参数
LTSMC.smc_set_s_profile(CardNo, Axis, 0, S_para);      // 设置S段参数
LTSMC.smc_pmove_unit(CardNo, Axis, Dist0, Posi_mode);      // 启动S形定长运动
}
```

2）关节J4（手爪）旋转的点动程序

如本文三.2节所述，关节J4做旋转运动时，Z轴也会运动。为了实现J4独立运动，必须同时让Z轴反向运动。采用2轴插补运动可实现此功能。详见如下程序。

```
private void buttonJ4p_Click(object sender, EventArgs e)      // 按键“J4+”的Click事件被触发
{
double dis;
dis = -16.0 * Dist3 / 360.0;     // 计算Z轴的运动距离，16mm为丝杆螺距
TwoAxesMotion(2, 3, dis, Dist3);     // 调用2轴插补函数
}
private void TwoAxesMotion(ushort Axis0, ushort Axis1, double D0, double D1)      // 2轴插补函数
{
ushort[] AxisArray = { 0, 1 };      // 定义数组
double[] Dist = { 0, 1 };
if (LTSMC.smc_check_done(CardNo, Axis0) == 1 && LTSMC.smc_check_done(CardNo, Axis1) == 1)
{      // 2轴无运动
AxisArray[0] = Axis0;      // 定义插补轴
AxisArray[1] = Axis1;
Dist[0] = D0;       // 设置运动距离
Dist[1] = D1;
LTSMC.smc_set_vector_profile_unit(CardNo, 0, Min_Vel, Max_Vel3, Tacc, Tdec, Stop_Vel);
LTSMC.smc_set_vector_s_profile(CardNo, 0, 0, S_para);
LTSMC.smc_line_unit(CardNo, 0, 2, AxisArray, Dist, 0);       // 起动2轴插补运动
}
```

3）X和Y方向的点动程序

如本文一.2节所述，当手爪仅向X方向运动时，关节J1、J2、J4都会运动；由本文三.2节所述，关节J4运动时，Z轴也会运动。故X方向的点动，实际上涉及到4个轴的运动，采用4轴插补运动可实现该功能。

X和Y方向的点动程序类似。仅给出X正方向点动程序如下。

```
private void buttonXp_Click(object sender, EventArgs e)      // 按键“X+”的Click事件被触发
{
x1 = x0 + DistX;      // 终点坐标为x1
InverseSolution(x1, y0);      // 求终点参数。算法参见本文一.2节
FourAxesMotion(d0, d1, d2, d3, Min_Vel, Max_Vel0, Stop_Vel);       // 调用四轴插补函数
}
private void FourAxesMotion(double D0, double D1, double D2, double D3, double Vstart, double Vmax, double Vend)
{      // 四轴插补运动
ushort[] AxisArray = { 0, 1, 2, 3 };      // 定义数组
double[] Dist = { 0, 1, 2, 3 };
if (LTSMC.smc_check_done(CardNo, 0) == 1 && LTSMC.smc_check_done(CardNo, 1) == 1 &&
LTSMC.smc_check_done(CardNo, 2) == 1 && LTSMC.smc_check_done(CardNo, 3) == 1)      // 四轴无运动
{
AxisArray[0] = 0;      // 定义插补0轴为J0轴
AxisArray[1] = 1;      // 定义插补1轴为J1轴
AxisArray[2] = 2;      // 定义插补2轴为Z轴
AxisArray[3] = 3;      // 定义插补3轴为J4轴
  Dist[0] = D0;      // J1轴运动距离
Dist[1] = D1;      // J2轴运动距离
Dist[2] = D2;      // Z轴运动距离
```

```
Dist[3] = D3;      // J4轴运动距离
LTSMC.smc_set_vector_profile_unit(CardNo, 0, Vstart, Vmax, Tacc, Tdec, Vend);      // 设置速度曲线参数
LTSMC.smc_set_vector_s_profile(CardNo, 0, 0, S_para);      // 设置S段参数
LTSMC.smc_line_unit(CardNo, 0, 4, AxisArray, Dist, 0);      // 起动四轴插补运动
}
```

4）关节J1、J2、Z轴的连动程序

关节J1、J2和Z轴的连动过程都是单轴运动，采用速度控制指令Vmove很容易实现此功能。控制关节J1正方向点动的程序如下所示，关节J2和Z轴的连动程序类似。

```
private void buttonJ1p_MouseDown(object sender, MouseEventArgs e)      // 按键按下，开始连续运动
{
Axis = 0;      // 选定0号轴
if (radioButton1.Checked == true && LTSMC.smc_check_done(CardNo, Axis) == 1)
{     // 单选钮“连动”选中，且电机未动
LTSMC.smc_set_profile_unit(CardNo,      Axis,      Min_Vel,      Max_Vel0,      Tacc,      Tdec,      Stop_Vel);
LTSMC.smc_set_s_profile(CardNo, Axis, 0, S_para);  // 设置速度曲线和S段参数  LTSMC.smc_stop(CardNo, Axis, 0);      // 电机减速停止  }
```

5）关节J4的连动程序

当按键按下后，用连续运动指令vmove控制关节J4按一定的速度持续运动；同时，用点位运动指令pmove控制Z轴跟随关节J4运动；并且打开一个定时器，每0.2秒自动检查一次位置误差，然后用在线变位指令调整Z轴的目标位置。

当按键抬起后，关节J4运动停止；然后，检查Z轴位置，最后一次在线调整Z轴的目标位置，使其误差为零。程序如下所示。

误差为零。程序如下所示。

```
private void buttonJ4p_MouseDown(object sender, MouseEventArgs e)      // 按键按下，开始连续运动
{
Axis = 3;
if (radioButton1.Checked == true && LTSMC.smc_check_done(CardNo, Axis) == 1)
{     // 单选钮“连动”选中，且电机未动
LTSMC.smc_get_position_unit(CardNo, 3, ref J3pos);      // 读取轴3指令脉冲位置
J3pos0 = J3pos;
LTSMC.smc_get_position_unit(CardNo, 2, ref J2pos);      // 读取轴2指令脉冲位置
J2posAbs = J2pos;
LTSMC.smc_set_profile_unit(CardNo,      Axis,      Min_Vel,      Max_Vel3,      Tacc,      Tdec,      Stop_Vel);
LTSMC.smc_set_s_profile(CardNo, Axis, 0, S_para); LTSMC.smc_vmove(CardNo, Axis, 1);      // 关节J4连续运动，方向为正
Axis = 2;
Max_Vel2 = 16.0 * Max_Vel3 / 360.0;
Dist2 = Max_Vel2 * 0.21;      // 计算Z轴的运动距离
LTSMC.smc_set_profile_unit(CardNo,      Axis,      Min_Vel,      Max_Vel2,      Tacc,      Tdec,      Stop_Vel);
LTSMC.smc_set_s_profile(CardNo, Axis, 0, S_para); LTSMC.smc_pmove_unit(CardNo, Axis, J2posAbs - Dist2, 1);      // Z轴跟随关节J4运动
timer3.Start();      // timer3的定时时间为0.2秒
}
private void timer3_Tick(object sender, EventArgs e)  // 0.2秒调整一次Z轴在线变位的坐标
{
LTSMC.smc_get_position_unit(CardNo, 3, ref J3pos);
Dist2 = (J3pos - J3pos0) * 16.0 / 360.0;      // 计算Z轴误差  LTSMC.smc_update_target_position_unit(CardNo, 2, J2posAbs - Dist2);      // 在线变位，消除Z轴误差
private void buttonJ4p_MouseUp(object sender, MouseEventArgs e)      // 按键抬起，停止运动
{
if (radioButton1.Checked == true)      // 单选钮“连动”选中
{
MotorStop(3);
timer3.Stop();
```

```
while (LTSMC.smc_check_done(CardNo, 3) == 0)
// 等待轴3停止
Application.DoEvents();      // 处理当前消息队列中的消息
LTSMC.smc_get_position_unit(CardNo, 3, ref J3pos);
Dist2 = (J3pos - J3pos0) * 16.0 / 360.0;      // 计算Z轴误差
LTSMC.smc_update_target_position_unit(CardNo, 2, J2posAbs - Dist2);      // 在线变位，消除Z轴误差
}
```

6）X、Y轴的连动程序

以X轴正方向连动为例：当按键按下后，计算位移量dx，然后计算终点处4轴的参数，调用4轴插补运动函数，且有加速过程；指令脉冲发送完后，继续向前运动dx，但4轴插补运动为匀速过程。

当按键抬起后，再向前运动最后一个dx，但4轴插补运动有减速过程。这样连动过程就比较连续、平稳。程序如下所示。

```
private void buttonXp_MouseDown(object sender, MouseEventArgs e)      // “X+”按键按下，开始连续运
{
double Xabs, Xabs0, deltaX;
if (radioButton1.Checked == true && LTSMC.smc_check_done(CardNo, 0) == 1
&& LTSMC.smc_check_done(CardNo, 1) == 1)      // 单选钮“连动”选中，且电机未动
{
deltaX = Max_Vel0 * 0.4;      // 根据速度计算运动距离
Xabs0 = x0;      //  （x0，y0）为工具中心点
Xabs = Xabs0 + deltaX;
Xabs0 = Xabs;
InverseSolution(Xabs, y0);      // 求目标位置各轴的参数
FourAxesMotion(d0, d1, d2, d3, Min_Vel, Max_Vel0, Max_Vel0);      // 有加速的4轴运动
while (LTSMC.smc_check_done(CardNo, 0) == 0)      // 等待0号轴停止
Application.DoEvents();      // 处理当前消息队列中的消息。
while (XpUp == 0)      // 标志位=按键未抬起
{
Xabs = Xabs0 + deltaX;      // 继续向前运动
Xabs0 = Xabs;
InverseSolution(Xabs, y0);
FourAxesMotion(d0, d1, d2, d3, Max_Vel0, Max_Vel0, Max_Vel0);      // 匀速的4轴运动
while (LTSMC.smc_check_done(CardNo, 0) == 0)      // 等待0号轴停止
Application.DoEvents();      // 处理当前消息队列中的消息。
}
Xabs = Xabs0 + deltaX;      // 再向前走最后一步
Xabs0 = Xabs;
InverseSolution(Xabs, y0);
```

4. 机械手位置动画显示程序

如图10所示，程序由一个定时器控制，每隔0.2秒在控制界面上显示一次机械手连杆L1、L2和手爪的位置。为提高动画显示质量，避免画面闪烁，坐标轴及网格、机械手运动范围不是每次重画一遍，而是直接调用事先画好的一个图片。动画程序如下所示。

```
……..
Bitmap pic = new Bitmap("d:\\robot\\Robot606\\back.png");      // 定义图片的位置和文件名
……..
private void timer1_Tick(object sender, EventArgs e)  // 定时器触发的函数
{
double a10, a20, a30;
LTSMC.smc_get_position_unit(CardNo, 0, ref J0pos);      // 读取指令脉冲，单位同电机脉冲当量
LTSMC.smc_get_position_unit(CardNo, 1, ref J1pos);
LTSMC.smc_get_position_unit(CardNo, 2, ref J2pos);
LTSMC.smc_get_position_unit(CardNo, 3, ref J3pos);
a10 = J0pos;
```

```
a20 = J1pos;
a30 = J3pos;
textBoxJ1Pos.Text = Convert.ToString(a10);      // 在文本框中显示4个轴的位置
textBoxJ2Pos.Text = Convert.ToString(a20);
textBoxJ3Pos.Text = Convert.ToString(J2pos);
textBoxJ4Pos.Text = Convert.ToString(a30);
a1 = a10 * pi / 180;      // 转换为弧度。
a2 = a20 * pi / 180;
a3 = a30 * pi / 180;
DrawArms(a1, a2, a3);      // 调用画图函数
x0 = L1 * Math.Cos(a1) + L2 * Math.Cos(a1 + a2) + L3 * Math.Cos(a1 + a2 + a3);
y0 = L1 * Math.Sin(a1) + L2 * Math.Sin(a1 + a2) + L3 * Math.Sin(a1 + a2 + a3);
textBoxXpos.Text = Convert.ToString(x0);      // 在文本框中显示x、y的位置
textBoxYpos.Text = Convert.ToString(y0);
}
private void DrawArms(double a1, double a2, double a3)      // 画2个连杆和手爪
{
double L1h, L2h, L3h;
double x, y, M, N;
int Ax0, Ay0, Bx0, By0;
int P1x, P1y, P2x, P2y, P3x, P3y;
L1h = L1 / 2;
L2h = L2 / 2;
L3h = L3 / 2;
Graphics g = pictureBox1.CreateGraphics();      // 创建画板
g.TranslateTransform(200, 200);      // 坐标平移，原点在中间，但y轴方向还是朝下
Pen BluePen = new Pen(Color.Blue, 2);      // 定义画笔
Pen RedPen = new Pen(Color.Red, 2);
Pen WhitePen = new Pen(Color.White, 80);
Pen GrayPen = new Pen(Color.Gray, 1);
Pen LightGrayPen = new Pen(Color.LightGray, 1); Pen BlackPen = new Pen(Color.Black, 1);
g.DrawImage(pic, -200, -200, pictureBox1.Width, pictureBox1.Height);      // 显示背景图片
aSum = a1 + a2 + a3;
J2x = L1h * Math.Cos(a1);      // 计算2个连杆的坐标
J2y = L1h * Math.Sin(a1);
x = J2x + L2h * Math.Cos(a1 + a2);
y = J2y + L2h * Math.Sin(a1 + a2);
M = x + L3h * Math.Cos(aSum);
N = y + L3h * Math.Sin(aSum);
Ax0 = (int)(Ax * Math.Cos(aSum) - Ay * Math.Sin(aSum) + M);      // 计算手爪的2个坐标点
Ay0 = (int)(Ax * Math.Sin(aSum) + Ay * Math.Cos(aSum) + N);
Bx0 = (int)(Bx * Math.Cos(aSum) - By * Math.Sin(aSum) + M);
By0 = (int)(Bx * Math.Sin(aSum) + By * Math.Cos(aSum) + N);
P1x = (int)J2x;
P1y = (int)-J2y;
P2x = (int)x;
P2y = (int)-y;
P3x = (int)M;
P3y = (int)-N;
g.DrawLine(BluePen, 0, 0, P1x, P1y);      // 画L1   g.DrawLine(RedPen, P1x, P1y, P2x, P2y);      // 画
L2   g.DrawLine(BluePen, P2x, P2y, P3x, P3y);      // 画L3   g.DrawLine(GrayPen, P3x, P3y, Ax0, -Ay0);      //
画手爪的3条边
g.DrawLine(GrayPen, P3x, P3y, Bx0, -By0); g.DrawLine(GrayPen, Ax0, -Ay0, Bx0, -By0);
```

```
}
```

5. 机械手自动控制程序

根据机械手动作要求，编写自动控制程序并不复杂。可调用点位运动指令、多轴插补指令、样条差值（PVT函数）指令等。重点是机器视觉与运动控制的混合编程。因篇幅受限，相关内容将在后续文章中介绍。

五． 小结

SMC606运动控制器的运动控制指令的种类多、功能强。只要充分理解SCARA机械手的运动学正解和反解的算法、灵活应用SMC606的指令，控制SCARA机械手达到快、稳、准的要求并不难。

六． 参考文献

[1] 郭洪红. 工业机器人技术（第三版）. 西安：西安电子科技大学出版社，2016

[2] 张爱红. 工业机器人应用与编程技术. 北京：电子工业出版社，2015

上一条：工控干货丨3C产品典型外形加工方案及示例
下一条：测评丨雷赛SMC304与SMC64X0系列运动控制器性能对比