

机器人学实验三

运动控制实验室 (K319)

1 实验目的

- 初步认识 C++运动规划上位机界面及程序模块构成。
- 掌握 SCARA、Delta 和六轴机器人的正逆运动学算法, 并使用 C/C++语言实现。
- 掌握 C++运动规划上位机中点位规划程序的使用流程。
- 深刻理解机器人正逆运动学在机器人运动控制中的作用。

2 实验内容

- 使用 C/C++编程语言实现 SCARA、Delta 和六轴机器人的正逆运动学算法。
- 使用 C++运动规划上位机中的点位规划程序验证机器人正逆运动学的正确性。
- 已知给定点位, 通过 C++运动规划上位机中的避障功能模块使机器人末端绘制出圆形轨迹。

3 实验基础

本实验以 SCARA、Delta 和六轴工业机器人为实验平台, 学生使用自己编写的机器人正逆运动学来实现上述三种机器人的空间运动。为了验证正逆解程序的正确性和适用性, 本实验通过 C++运动规划上位机中的点位规划功能使机器人执行特定的轨迹运动。为了让学生更深刻地理解正逆运动学如何应用到机器人的运动控制中, 该部分介绍了 C++运动规划上位机界面及其程序模块组成。

3.1 程序介绍

3.1.1 C++运动规划上位机介绍

C++运动规划界面目前提供了 4 种运动模式, 分别是: ①轨迹规划点位运动、②结合视觉的取放料运动、③码垛运动、④避障碍运动。其中②, ③, ④的连续轨迹运动是①功能的复合运用。①功能的轨迹规划点位运动是点到点的运动。

举例说明: 若要执行功能④避障, 避障的路径是由 4 个点组成, 分别是 A,B,C,D。那么运动的流程为 $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, 可以拆分 3 组①功能点到点的运动。②和③的综合功能也是将连续的运动拆分为了几组①功能点到点的运动实现了连续的轨迹运动。

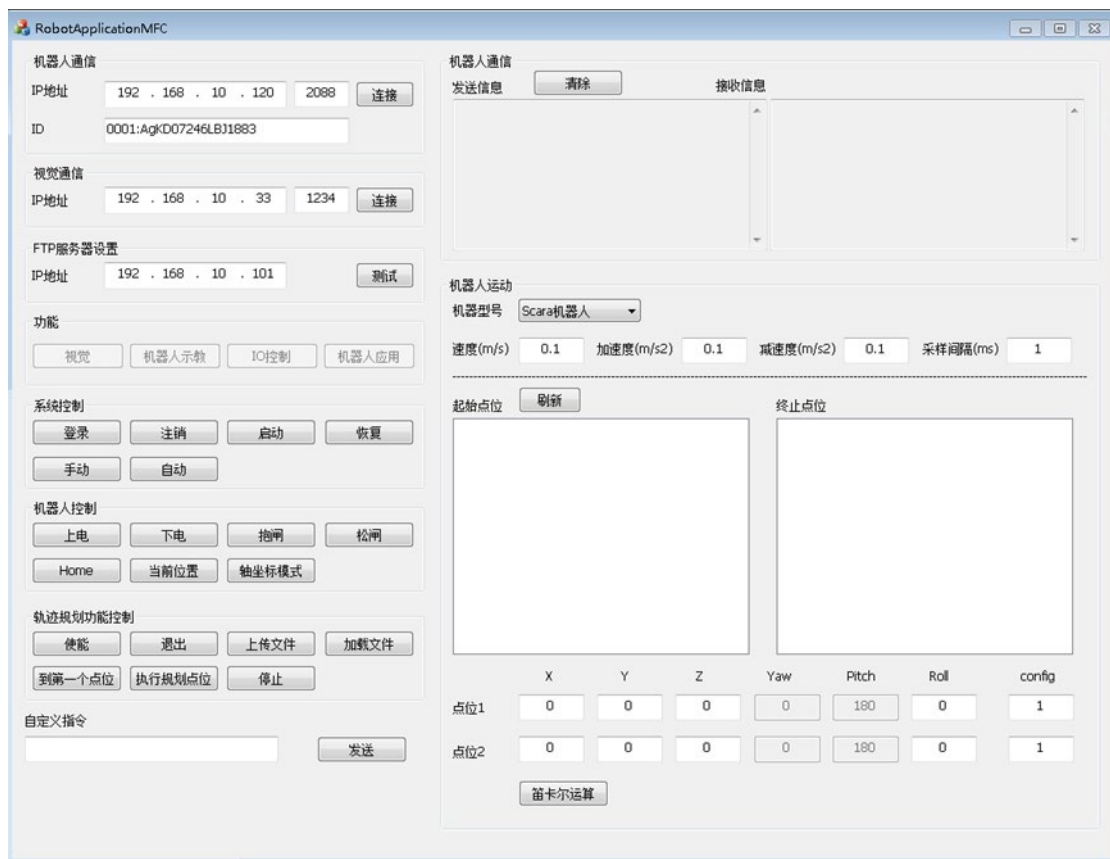


图 1 C++运动规划上位机界面

3.1.2 程序模块概述

C++运动规划上位机由图所示中的源码组成。

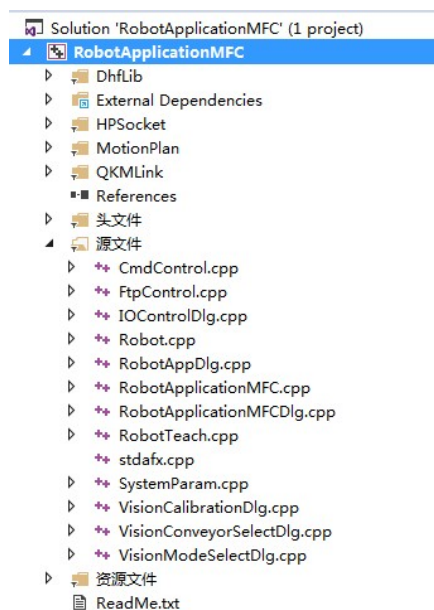


图 2 C++运动规划上位机源码组成

其中源码的各模块的名称和意义如下表所示。

模块名	意义
DhfLib	单例模式模板
HPSocket	Socket 网络通讯库
MotionPlan	轨迹规划算法
CmdControl	启动视觉软件
FtpControl	Ftp 通讯将 Data 文件放入、机器人的本体
IOControlDlg	IO 控制界面底层函数
Robot	机器人的宏指令接口
RobotApplicationMFC	上位机程序的入口
RobotApplicationMFCDlg	上位机主界面底层函数
RobotTeach	上位机示教界面底层函数
SystemParam	系统参数保存模块，用来保存点位等参数
VisionCalibrationDlg	视觉标定界面底层函数
VisionConveyorSelectDlg	传送带选择界面底层函数
VisionModeSelectDlg	视觉模式选择界面底层函数
RobotApplicationMFC	资源文件，包含了界面的 DATA 文件

RobotApplicationMFC 资源模块包含了窗口界面设计，一共有以下几个界面。

IDD_DIALOG_CALIBRATION	视觉标定界面
IDD_DIALOG_CONVEYOR	选择传送带颜色界面
IDD_DIALOG_IO_CONTROL	IO 控制界面
IDD_DIALOG_ROBOT_APP	机器人应用界面
IDD_DIALOG_TEACH	示教界面
IDD_DIALOG_VISIONMODEL	视觉模式选择界面
IDD_ROBOTAPPLICATIONMFC_DIALOG	主界面

3.2 点位规划程序原理

3.1.2 程序原理介绍

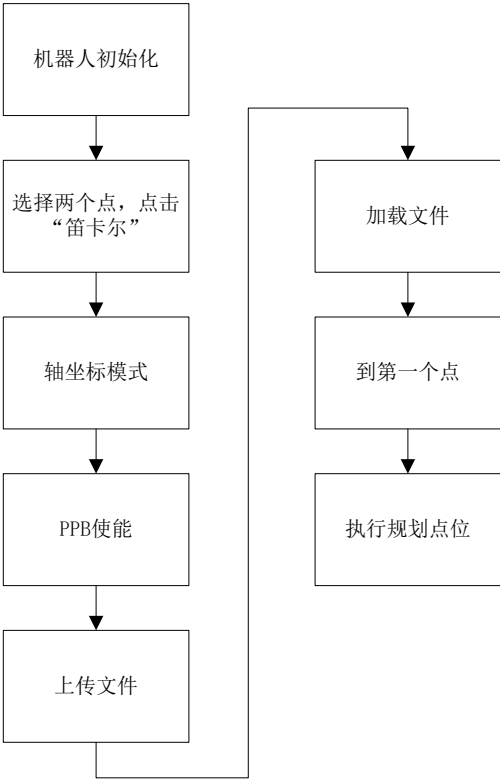


图 3 点位规划流程

图 3 介绍了 C++上位机程序中点位规划功能的实现流程，其中在第二个步骤“选择两个点，点击笛卡尔运算”中，将会应用机器人的正逆运动学来得到运动轨迹中每个关节的旋转角度。点位规划功能主要应用到下图中红色方框中的内容。

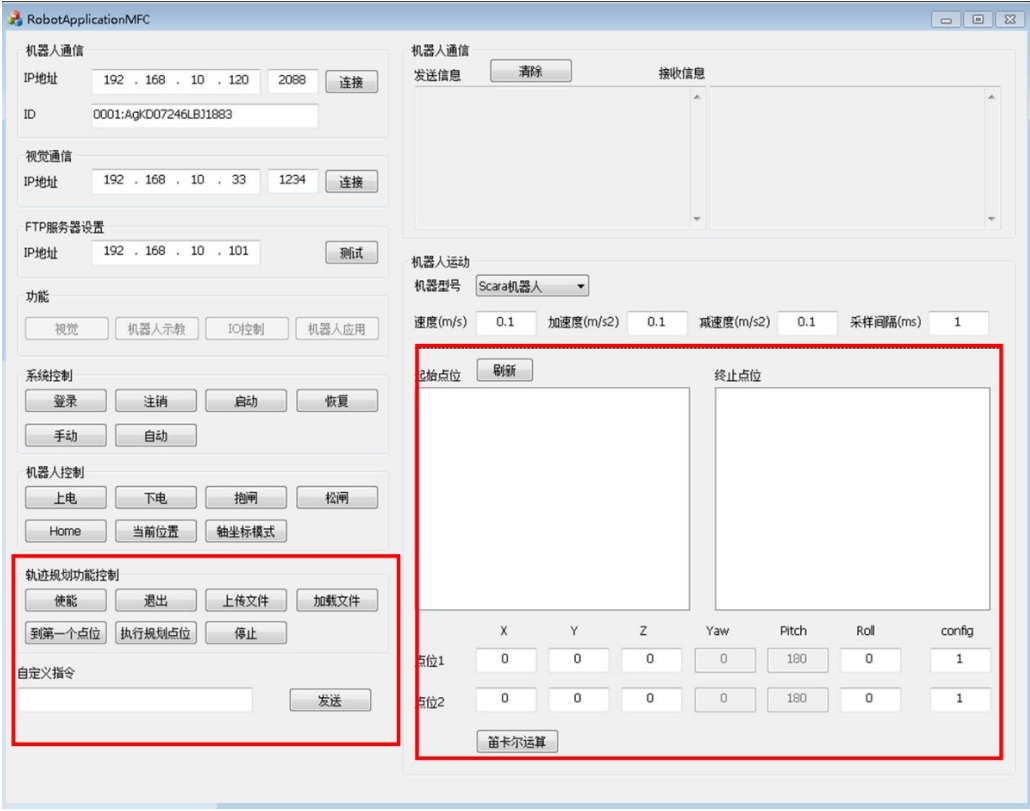


图 4 点位规划界面图

如图 5 所示的 Button 按钮，在点击时会触发相应的事件，并执行对应的操作。比如点击“登陆”按钮，底层会触发 OnBnClickedButtonSystemLogin 函数，调用里面的 Login（）函数发送“System.Login 0”的宏指令到机器人端。同理其它按钮也是这样的效果。

```
//登陆
void CRobotApplicationMFCDlg::OnBnClickedButtonSystemLogin()
{
    // TODO: 在此添加控件通知处理程序代码
    Login();
}
```

图 5 点击 Button 触发此函数

```

bool Robot::Login()
{
    if (IsConnected)
    {
        //复位接收标志
        RecvLock.lock();
        RecvStatus = 0;
        Message = L"";
        RecvLock.unlock();

        CString sendData;
        sendData.Format(_T("System.Login 0"));

        //发送数据
        bool result = Send(sendData);
        if (!result)
        {
            return false;
        }

        //等待数据返回
        for (size_t i = 0; i < ROBOT_RECV_TIME_OUT; i++)
        {
            RecvLock.lock();

```

图 6 Robot 模块内部的宏指令函数 Login

3.1.2 程序应用流程

1) 机器人初始化

机器人初始化是指在机器人执行轨迹规划运动前的准备。常见的操作：
登陆（加载权限。宏指令“System.Login 0”）

↓

上电（机器人上使能。宏指令“Robot.PowerEnable 1,1”）

↓

回零（更新编码器值，一般断点重启后执行一次即可。宏指令“Robot.Home”）

2) 选择两个点，点击笛卡尔运算

界面点击“刷新”后会显示目前已记录的点位，选择起始点和终点后，点击“笛卡尔运算”按钮，便会生产“/Data.txt”文档，里面包含了规划的点位。

```

-11.0458 43.2569 10.024 154.71
-11.0458 43.2569 10.024 154.71
-11.0458 43.2569 10.024 154.71
-11.0458 43.2568 10.024 154.71
-11.0458 43.2568 10.024 154.71
-11.0458 43.2568 10.024 154.71
-11.0458 43.2567 10.024 154.71
-11.0458 43.2567 10.024 154.71
-11.0458 43.2566 10.024 154.71
-11.0458 43.2566 10.024 154.71
-11.0458 43.2565 10.024 154.71
-11.0458 43.2565 10.024 154.71
-11.0458 43.2564 10.024 154.71
-11.0458 43.2563 10.024 154.71

```

图 7 轨迹规划生产的关节坐标

3) 使能

使能是 PPB 宏指令的一种，使能是指是开启此模式。Ping-pong buffer (PPB) 功能是，使用者 bypass Pallas 内部默认的 motion planner 和 trajectory generator，以实时网络通信周期 (e.g 1ms) 把指令讯息直接给到 servo，然后执行 robot 动作。或者，用户给出一些关键点位，在底层进行样条曲线拟合，轨迹规划，最后将实时指令讯息给到伺服。

4) 上传文件

将点位文件通过 FTP 拷入 192.168.10.101/data。

5) 加载文件

调用宏指令 PPB.ReadFile 1,/data/XXX.txt 的方式，将规划的点位压入底层的 Buffer 中。

6) 运动到第一个点

运动到起始位置，为执行后面的运动做准备。

7) 执行轨迹规划点位

PPB.Run 1，执行 Data 文件内的点位。

4 实验操作指南

本实验过程中将会应用同学自己完成的机器人正逆运动学程序来实现机器人的运动，因此在实验前学生务必完成机器人正逆运动学的编程任务。在完成实验内容前，需要先将原程序中的正逆运动学部分替换为学生的正逆运动学程序。为了保证机器人的正常运行和实验的安全性，切记在替换之前要验证所写程序的正确性。下面内容介绍了正逆运动学程序替换的流程以及后续实验操作步骤。

4.1 替换 C++ 正逆运动学程序

4.1.1 SCARA 机器人

- 1) 打开桌面上的 RobotApplicationMFC 文件夹，双击 RobotApplicationMFC.sln 程序打开，如图 8 所示。

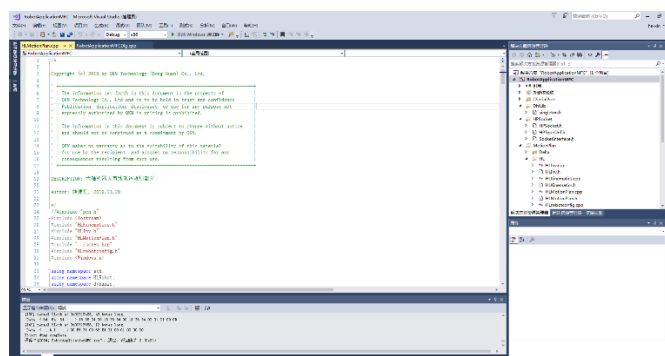


图 8 RobotApplicationMFC 程序界面

- 2) 在解决方案资源管理器当中找到 MotionPlan 部分，打开该文件列表，找到 SCARA 文件夹，打开后如图 9 所示。

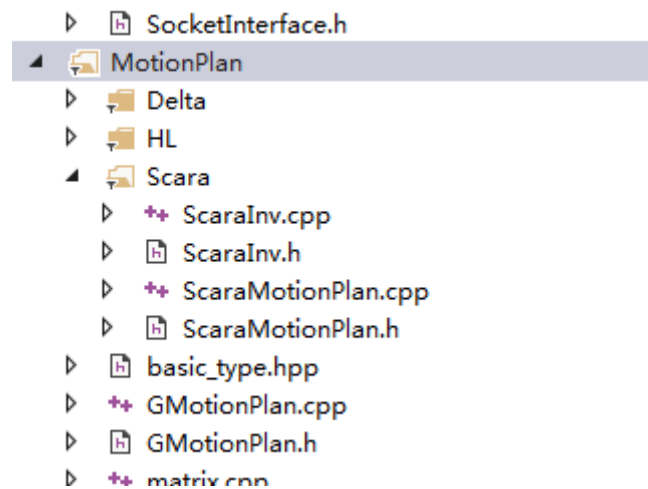


图 9 SCARA 文件夹列表

- 3) 先将列表中的 ScaraInv.cpp 和 ScaraInv.h 删除，和在 SCARA 文件夹上点击鼠标右键找到添加选项，点击添加现有项，如图 10 所示。将自己的 ScaraInv.cpp 和 ScaraInv.h 文件添加到该文件夹内。若有其他的头文件、执行文件和库文件，一并拷贝到当前文件夹下。

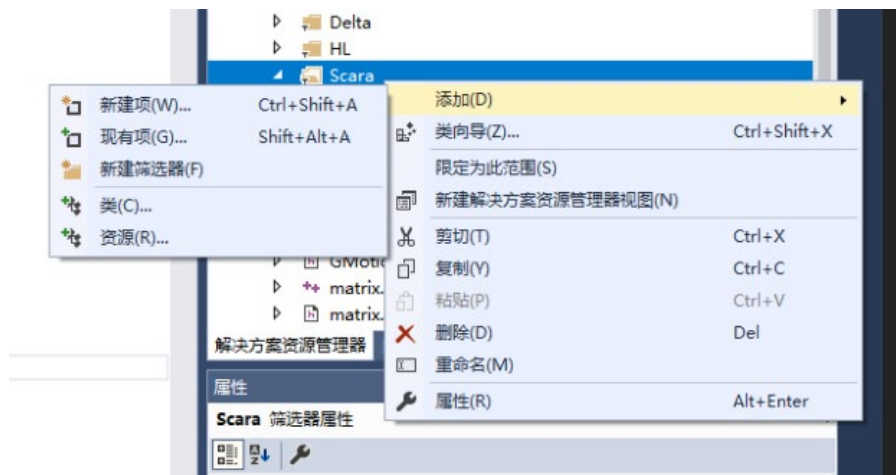


图 10 添加现有项

4.1.2 Delta 机器人

先前步骤可参考 SCARA 机器人的正逆运动学程序替换步骤，在 MotionPlan 部分中的 Delta 文件列表中双击打开 AProbotconfig.cpp 文件，在该文件中找到 void robotBackward(const double* TransVector, double* theta) 和 void robotForward(const double* q, const double* Tool, double* TransVector)，将自己所写的两个函数拷贝进去即可。

4.1.3 六轴机器人

先前步骤可参考 SCARA 机器人的正逆运动学程序替换步骤，在 MotionPlan 部分中的 HL 文件列表中双击打开 HLrobotconfig.cpp 文件，在该文件中找到 void robotBackwardHJQ(const double* T, bool* config, double* Tool, bool* Turns, double* theta) 和 void robotForwardHJQ(const double* q, const double* Tool, double* TransVector, bool* config, bool* turns)，将自己所写的两个函数拷贝进去即可。

4.2 使用正逆解完成点位规划

1) 点击本地 Windows 调试器运行该程序，其界面如图 11 所示。



图 11 机器人初始化及刷新点位

首先进行机器人初始化，依次点击 1（连接）、2（连接）、3（登录）、4、（上电）、5（自动）、6（Home）、7（轴坐标模式）、8（刷新）。

2) 点击刷新后出现下图所示界面，分别选取起始点位和终止点位，然后点击笛卡尔运算，运算成功后将会在 ...\\RobotApplicationMFC-VS2017\\RobotApplicationMFC 文件夹下生成具有一系列点位的 data.txt 文件，打开数据文件可以查看各关节坐标，可以用来验证机器人正逆解的正确性。

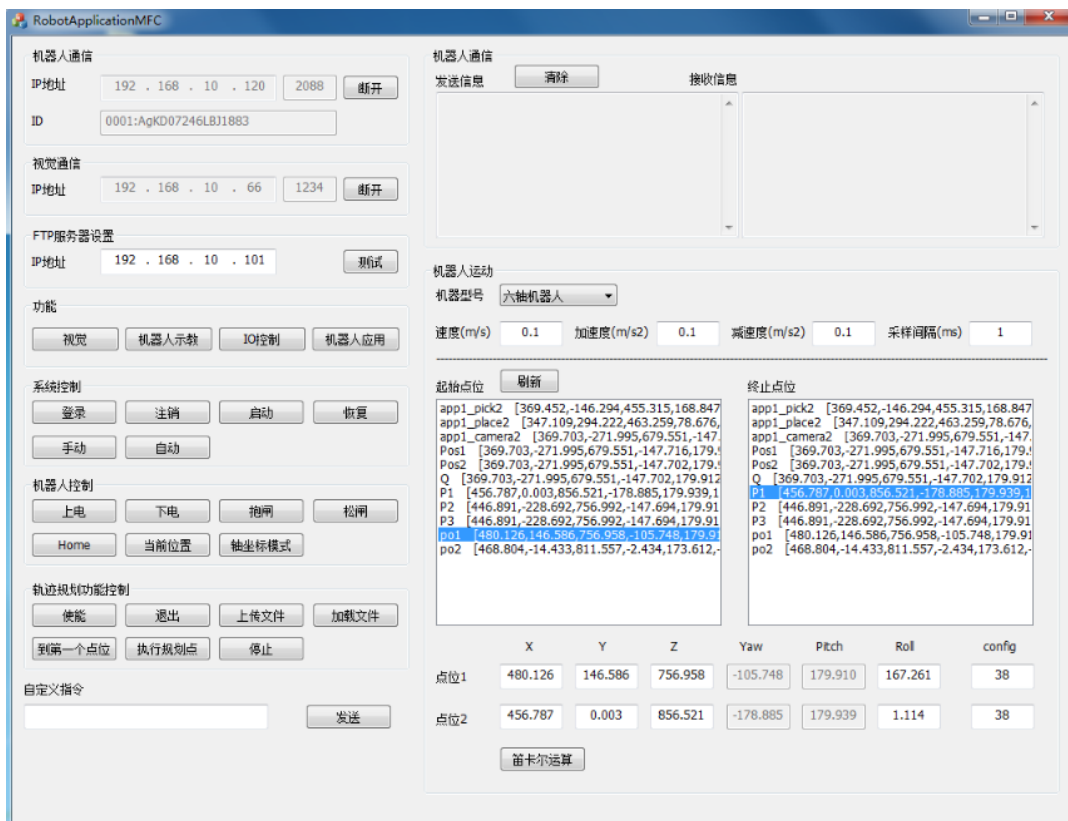


图 12 机器人初始化及刷新点位

接下来依次点击**使能**、**上传文件**、**加载文件**、**到第一个点位**，点击之后机器人会运动到起始点位，点击**执行规划点**，机器人将会按照 data.txt 中已生成点位信息移动到终止点位。本次实验共使用三组点位信息来验证机器人正逆运动学。

4.3 使用正逆解完成避障运动

- 1) 在 C++轨迹规划主程序界面点击**机器人应用**，点击**刷新**，出现下图所示画面。

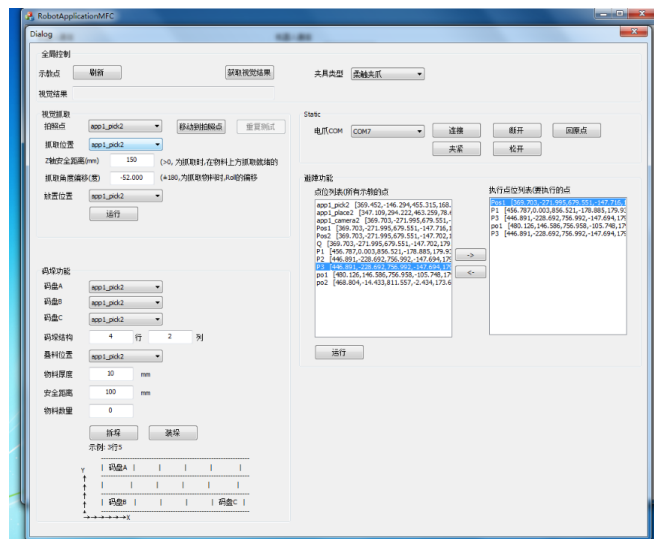




图 13 机器人应用界面

- 2) 在右边的避障功能区域，选择左侧点位列表中的一个点，点击则可以将点位保存到右边的执行点位列表中，点击则机器人可以按照列表中的点位依次完成点到点的运动。
- 3) 本实验已经给定了 8 个点位信息，8 个点在同一个圆上，所以机器人所绘制出的轨迹近似为一个圆形，如图 14 所示。

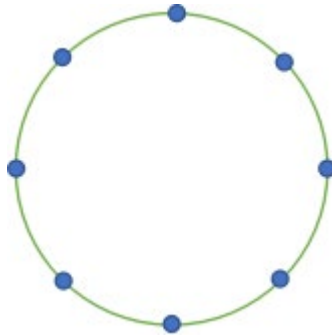


图 14 圆形轨迹上的 8 个点位

具体的点位信息如下所示，yaw，pitch，roll 的值可以自己确定。

SCARA:

中心点 320, 85.198, -3.784 R=30

1 点: 350, 85.198, -3.784

2 点: 341.213, 63.985, -3.784

3 点: 320, 55.198, -3.784

4 点: 298.787, 63.985, -3.784

5 点: 290, 85.198, -3.784

6 点: 298.787, 106.411, -3.784

7 点: 320, 115.198, -3.784

8 点: 341.213, 106.411, -3.784

Delta:

中心点-12.81, -26.53; -574.6 R=30

1 点: 17.19, -26.53 ; -574.6

2 点: 8.403, -47.743, -574.6

3 点: -12.81, -56.53, -574.6

4 点: -34.023, -47.743, -574.6

5 点: -42.81, -26.53 ; -574.6

6 点: -34.023, 5.317, -574.6

7 点: -12.81, 3.47, -574.6

8 点: 8.403, 5.317, -574.6

六轴:

中心点 421.357, 85.418, 705.568 R=30

1 点: 451.357, 85.418, 705.568

2 点: 442.57, 64.205, 705.568

3 点: 421.357, 55.418, 705.568

4 点: 400.144, 64.205, 705.568

5 点: 391.357, 85.418, 705.568

6 点: 400.144, 106.631, 705.568

7 点: 421.357, 115.418, 705.568

8 点: 442.57, 106.631, 705.568

- 4) 使用机器人示教不可以准确的将上述点位保存到点位列表中，在 ...\\RobotApplicationMFC-VS2017 \\RobotApplicationMFC 文件夹中找到 SystemParams.json 文件，打开后，自己添加图 15 中所示的部分。添加后，在机器人应用界面中点击刷新，即可看到自己添加的点位信息。

```
{
    "Config" : 38,
    "Joint" :
    [
        -16.166,
        22.248000000000001,
        107.233,
        -1.113,
        51.537999999999997,
        -57.444000000000003
    ],
    "Name" : "A",
    "Point" :
    [
        495.44999999999999,
        -144.95699999999999,
        530.0,
        -155.84200000000001,
        178.64599999999999,
        -17.538
    ]
},
```

图 15 点位信息的添加

6 实验报告

该实验的实验报告所写内容均根据自己所实际操作的机器人来完成。

- 1) 写出机器人正逆解的推导过程，并给出 C/C++ 代码。
- 2) 在点位规划实验内容中，记录所保存的三组点位信息（即三组起始点和终止点）和三个 data.txt 文件，使用 matlab 分别将三条规划的轨迹绘制在笛卡尔坐标系中。