旋转矢量UR官方

<http://folk.ntnu.no/tomgra/Diplomer/Kufieta.pdf?id=ansatte/Gravdahl_Jan.Tommy/Diplomer/Kufieta.pdf#page74>

旋转矢量-矩阵-欧拉角-四元数验证

<https://www.andre-gaschler.com/rotationconverter/>

详细步骤：

1.UR开机并写入程序（ps:UR使用socket通讯时，只能作为客户端）

2.网线连接PC并设置IP地址（ps:这里IP设置为192.168.1.102）

3.PC编辑程序（ps:注意端口号，需大于5000）

4.先运行PC程序（服务端），次运行UR程序（ps：调试时减小UR速度）

5.成功

案例1：

**UR程序<标定>**

 程序

   初始化变量

   BeforeStart

     var\_2≔p[-0.58,-0.05,-0.05,3.1415927,0,0]            //初始位置，xyz单位为m

     nGripper90:  nGripper\_0 22                           //夹爪张开

     open=socket\_open("192.168.1.102",8000)           //连接PC

     循环 open≟ False                                     //连接不成功，则再次请求连接

       open=socket\_open("192.168.1.102",8000)

     rv≔[0,0,0]

   机器人程序

     MoveJ                                         //移动到初始位姿<后续的移动转弯半径设置为0，便于观察和获取位姿>

       var\_2

     循环 5 次数

       pos3≔get\_actual\_tcp\_pose()             //获取点，前三个为位置（xyz），后三个为姿态（rxryrz）

       socket\_send\_string(pos3)               //发送位姿给PC（类型为char，格式为p[x,y,z,rx,ry,rz]，rx,ry,rz是旋转矢量）

       receivepos\_grip=socket\_read\_ascii\_float(6)       //获取PC传给UR的值（float类型）

       循环 receivepos\_grip[0]≠6                          //未取得值，循环读取

         等待: 0.1

         receivepos\_grip=socket\_read\_ascii\_float(6)

       Xgrip≔receivepos\_grip[1]                      //赋值

       Ygrip≔receivepos\_grip[2]

       Zgrip≔receivepos\_grip[3]

       RXgrip≔receivepos\_grip[4]

       RYgrip≔receivepos\_grip[5]

       RZgrip≔receivepos\_grip[6]

       rv≔[RXgrip,RYgrip,RZgrip]                      //如果输入的是欧拉角需转为旋转矢量

       pos2≔get\_actual\_tcp\_pose()                   //ur提供欧拉角zyx转为旋转矢量的函数：rpy2rotvec

       MoveJ                  //也可以在PC上把欧拉角转为旋转矢量http://download.csdn.net/download/yuleshare/10208339

         pos2

         pos1\_2≔p[Xgrip,Ygrip,Zgrip,RXgrip,RYgrip,RZgrip]//赋值

         MoveJ

           pos1\_2                                                              //移动

           pos3≔get\_actual\_tcp\_pose()

   线程\_1                                                                      //夹爪维持状态

     nGripper90:  nGripper\_0 21

     等待: 1.0

**PC程序<标定>**

#include "winsock2.h"

#pragma comment(lib, "ws2\_32.lib")

#include <iostream>

using namespace std;

int main(int argc, char\* argv[])

{

const int BUF\_SIZE = 64;

WSADATA         wsd;            //WSADATA变量

SOCKET          sServer;        //服务器套接字

SOCKET          sClient;        //客户端套接字

SOCKADDR\_IN     addrServ;;      //服务器地址

char            buf[BUF\_SIZE];  //接收数据缓冲区

//char            sendBuf[60];//返回给客户端得数据

int             retVal;         //返回值

//初始化套结字动态库

if (WSAStartup(MAKEWORD(2, 2), &wsd) != 0)

{

cout << "WSAStartup failed!" << endl;

return 1;

}

//创建套接字

sServer = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (INVALID\_SOCKET == sServer)

{

cout << "socket failed!" << endl;

WSACleanup();//释放套接字资源;

return  -1;

}

//服务器套接字地址

addrServ.sin\_family = AF\_INET;

addrServ.sin\_port = htons(8000);

addrServ.sin\_addr.s\_addr = INADDR\_ANY;

//绑定套接字

retVal = bind(sServer, (LPSOCKADDR)&addrServ, sizeof(SOCKADDR\_IN));

if (SOCKET\_ERROR == retVal)

{

cout << "bind failed!" << endl;

closesocket(sServer);   //关闭套接字

WSACleanup();           //释放套接字资源;

return -1;

}

//开始监听

retVal = listen(sServer, 1);

if (SOCKET\_ERROR == retVal)

{

cout << "listen failed!" << endl;

closesocket(sServer);   //关闭套接字

WSACleanup();           //释放套接字资源;

return -1;

}

//接受客户端请求

sockaddr\_in addrClient;

int addrClientlen = sizeof(addrClient);

sClient = accept(sServer, (sockaddr FAR\*)&addrClient, &addrClientlen);

if (INVALID\_SOCKET == sClient)

{

cout << "accept failed!" << endl;

closesocket(sServer);   //关闭套接字

WSACleanup();           //释放套接字资源;

return -1;

}

ZeroMemory(buf, BUF\_SIZE);

retVal = recv(sClient, buf, BUF\_SIZE, 0);//接受数据，char类型，格式为p[x,y,z,rx,ry,rz]

if (SOCKET\_ERROR == retVal)

{

cout << "recv failed!" << endl;

closesocket(sServer);   //关闭套接字

closesocket(sClient);   //关闭套接字

WSACleanup();           //释放套接字资源;

return -1;

}

char sendBuf[80];

char sendBuf[80] = "(0.46,0.1,0.4,3.14,0,0)";//后三个为旋转矢量，可以写欧拉角转旋转矢量函数，然后赋值

send(sClient, sendBuf, strlen(sendBuf), 0);

cout << "客户端发送的数据: " << sendBuf << endl;

cout << "向客户端发送数据: " << buf << endl;;

}

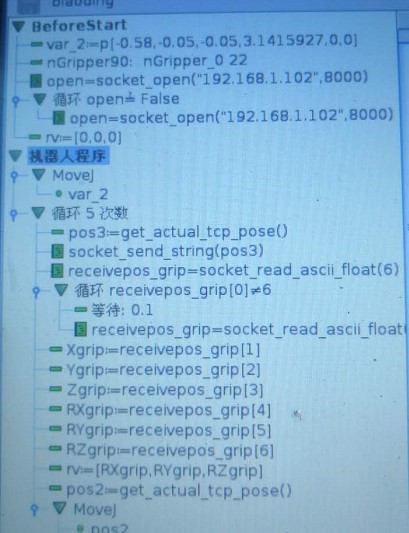
closesocket(sServer);   //关闭套接字

closesocket(sClient);   //关闭套接字

WSACleanup();           //释放套接字资源;

return 0;

}//程序有删减，逻辑测试过



**参考：**

1.旋转矢量UR官方

<http://folk.ntnu.no/tomgra/Diplomer/Kufieta.pdf?id=ansatte/Gravdahl_Jan.Tommy/Diplomer/Kufieta.pdf#page74>

2.官方脚本手册