《单片机系统设计》结题报告

**基于Kinect和树莓派的体感运动小车**



**学生姓名 \_\_\_赵思蒙**

**学生学号 517021910935**

**学生班级 F1702113**

**任课教师 \_\_付庄\_\_\_**

**同组同学 陆以凡**

**实验日期 \_\_2020.12.7\_\_**

# 项目背景。

Kinect传感器是由Microsoft微软公司开发的一款具有深度摄像头、红外传感器的传感设备，被广泛运用于其衍生的Xbox游戏机体感游戏的开发。利用Kinect设备和在其官网下载的Kinect相关固件，包括Microsoft的SDKs包等，可以形成在电脑上开发基于Kinect的项目的环境。

日常生活中，老人常常会遇到行动不方面的问题，尤其是当遇到需要弯腰拾取物品和取快递等情况的时候。且对于老人来说，使用比较专业的遥控器或电脑来进行控制需要长时间的学习和适应，对视力注意力等身体条件也有要求，普适性较低。针对这个问题，我们设计了一款可以通过手势来控制的带机械臂的智能机器人。让老人可以通过比较直观形象的手势来控制小车处理各种事务，如拾取掉落的钱包，到门口取报纸，查看房间角落等。兼具娱乐性和实用度。

# 项目目标。

我们希望设计一款基于Kinect传感器的体感小车，利用Kinect传感器识别出我们设计的一些手势，将其转换为指令并与下位机（树莓派）通信，进而让树莓派和相应的电机驱动板来控制小车完成相应的指令，完成我们的体感小车设想。

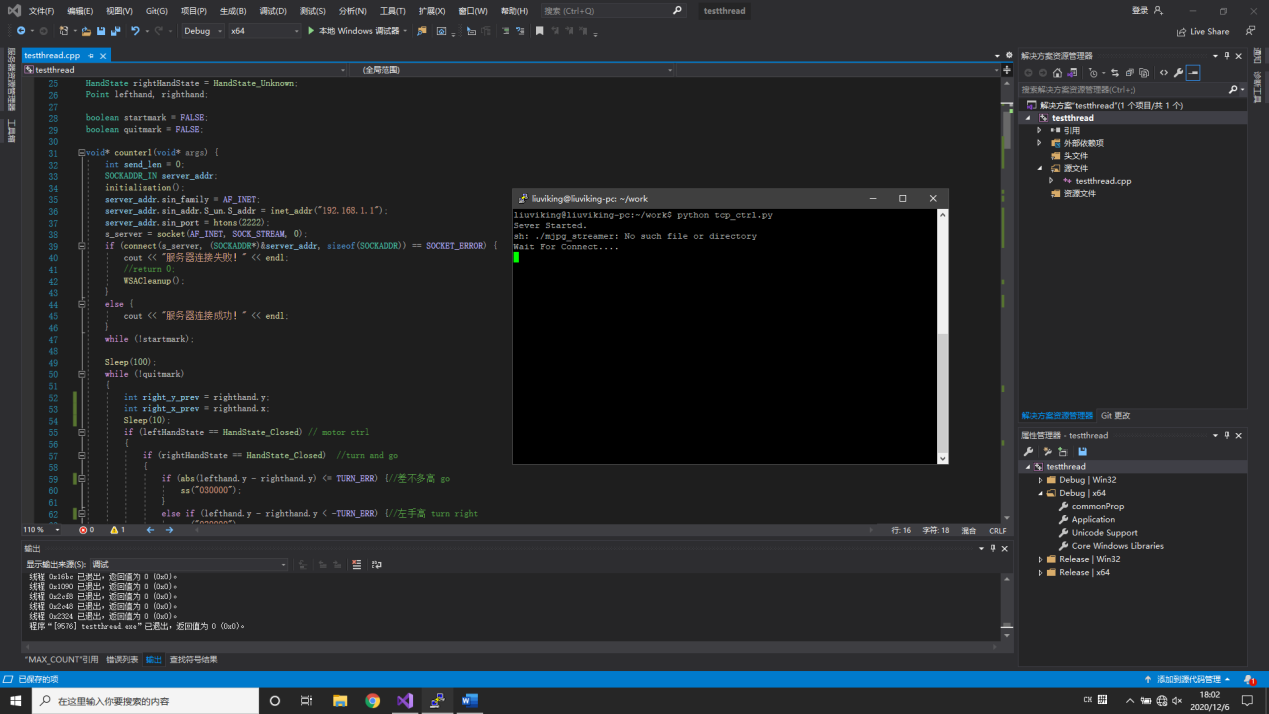
# 小组分工。

小组成员：陆以凡、赵思蒙（我）。

个人工作：Kinect部分功能的代码实现、骨骼数据提取；摄像头调试；下位机部分代码编写和调试。

# 功能实现及效果图

1.登录及初始化界面

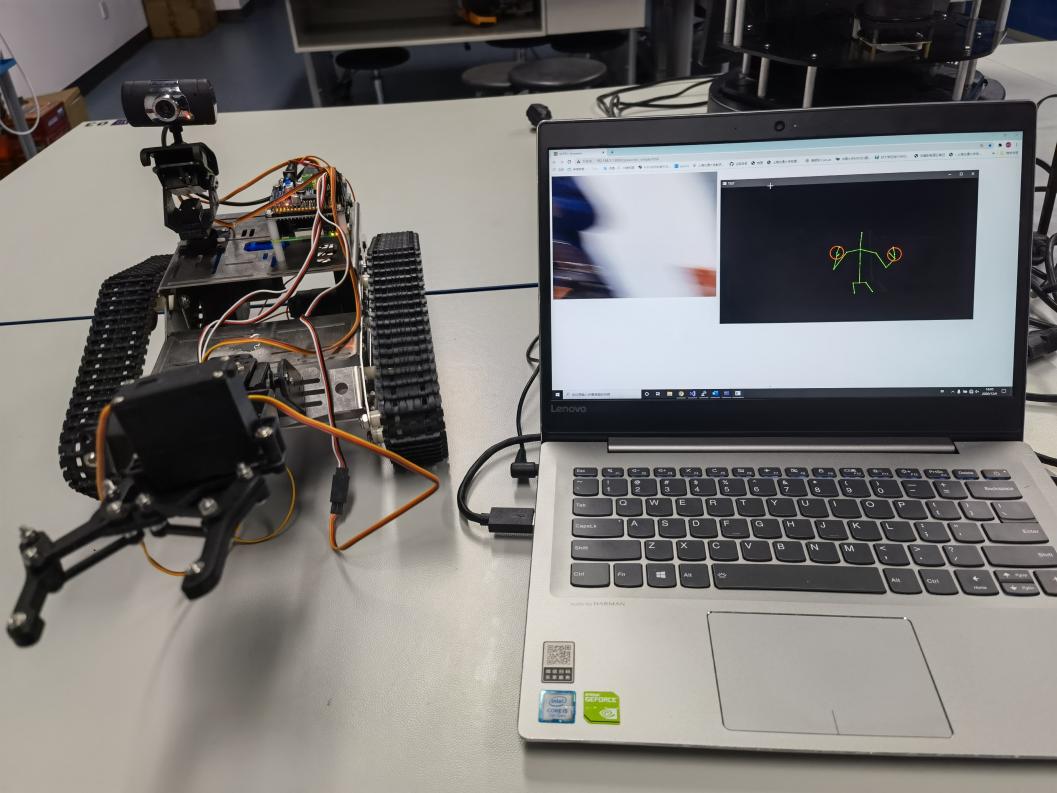


左侧为车载摄像头 右侧为kinect骨骼识别影像



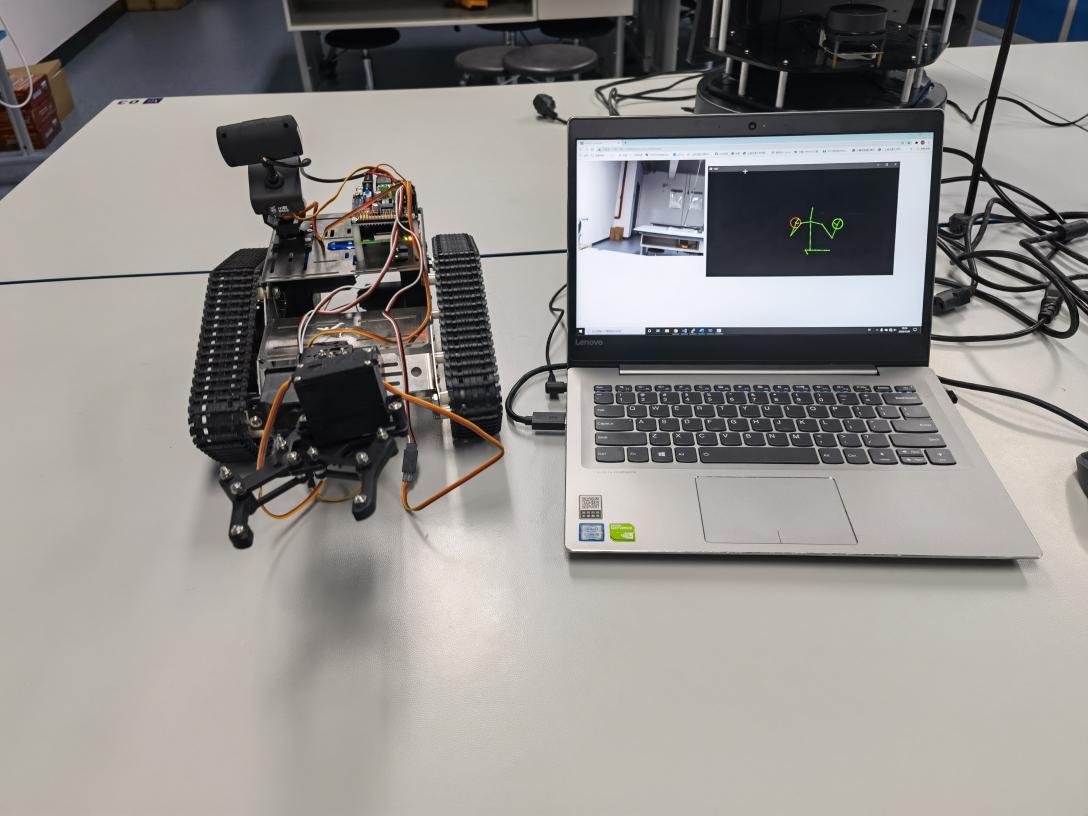
2.前进 手势：左手握拳，右手张开

（摄像头朝前）



3.后退 手势：左手握拳 右手V字

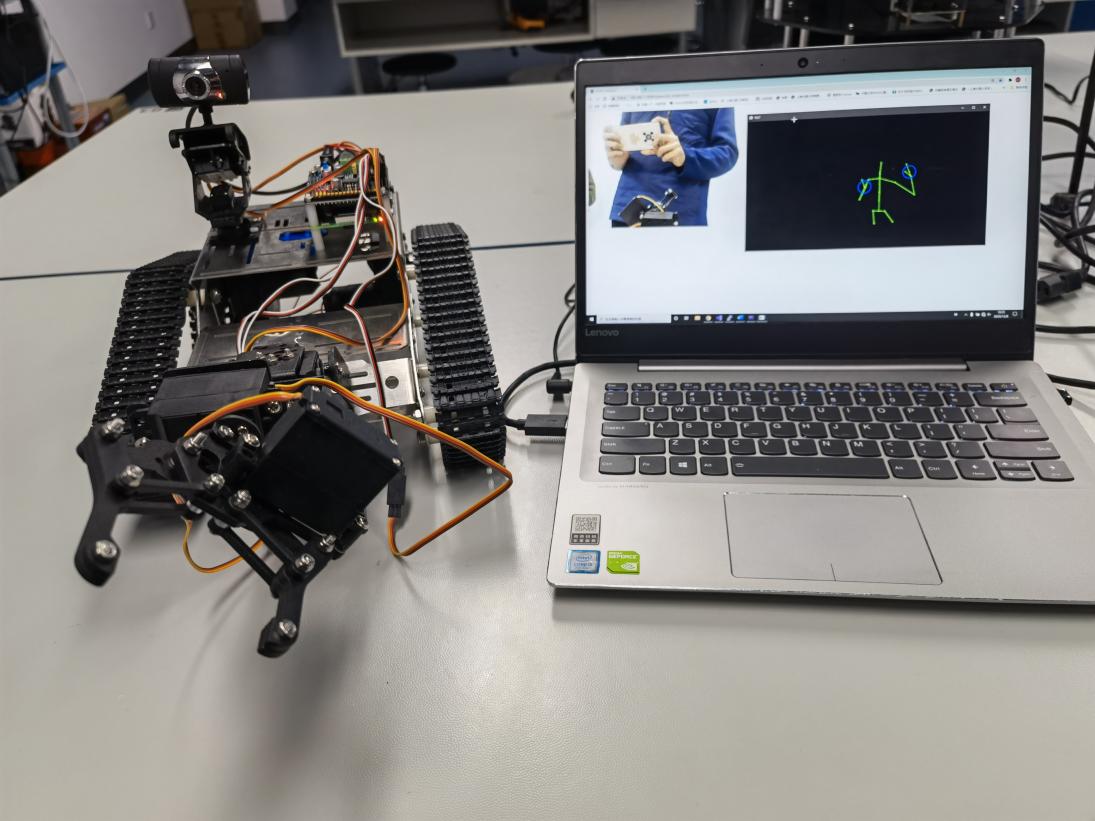
（摄像头自动转向向后，观察后方环境）



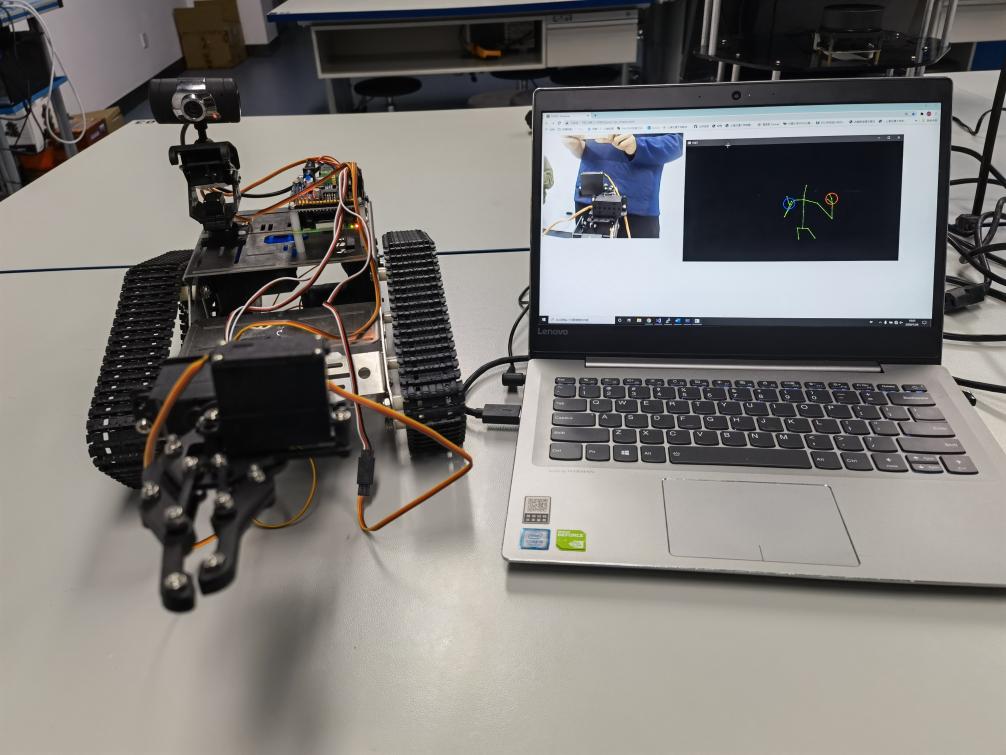
4.转向 手势：任意手势下，像方向盘那样操作，对于右手不同手势可分为前进转向和后退转向



5.末端夹具转动 手势：左手V字 右手V字 双手方向盘形式转动

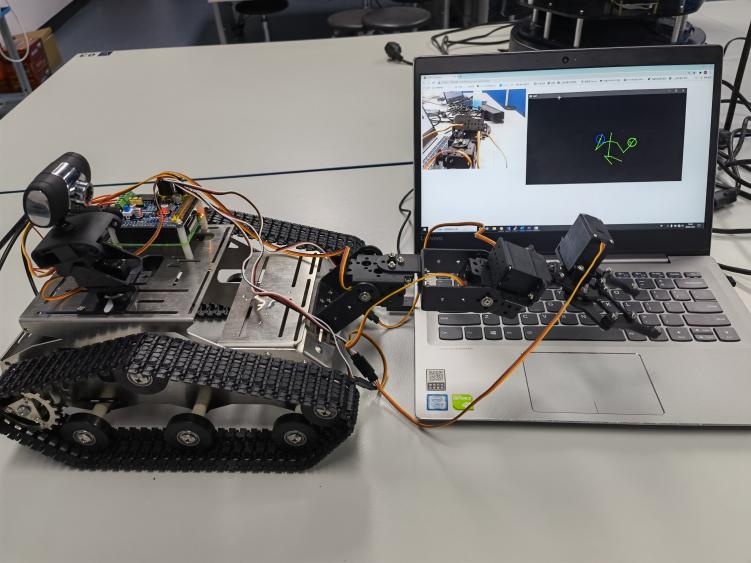
.

6.末端夹具开合 手势：左手V字 右手张开表示夹具张开 右手握拳表示夹具合拢



7.机械臂移动 手势：左手V字 右手任意手势 上下移动机械臂上下移动 左右滑动小车左右缓慢转向调整机械臂位姿。 特殊功能：摄像头跟随末端夹具移动，确保夹具始终在视野中





8.紧急制动 手势：左手张开 所有运动停止。

# 硬件、软件、开发环境介绍。

## 硬件。

1. 小R科技TH-ROBOT树莓派小车，搭载3自由度机械臂、摄像头、树莓派3B和电机驱动板。
2. Kinect2.0传感器，具有红外传感、深度感知、姿态识别等功能。
3. 笔记本电脑一台，Win10系统，作为上位机，连接Kinect，并无线连接树莓派。

## 软件。

1. Putty：用于与树莓派进行SSH通信，并进入命令行控制树莓派。
2. WinSCP：用于与树莓派方便、直观地传输文件。
3. Notepad++：方便地编写Python脚本代码供树莓派调用。
4. Microsoft Visual Studio2019：用于C++环境搭建、C++代码编写和调试。
5. OpenCV2：用于图像处理、作图、显示图像等。
6. Pthread：多线程包，管理多线程代码。
7. Kinect SDKs for Windows：和Kinect相关的一些链接库和头文件。

## 环境。

OpenCV、Kinect、Pthread相关的头文件、静态链接库、动态链接库文件。

详细的环境配置见个人博客：https://blog.csdn.net/qq\_15036691/article/details/109995188

# 项目具体实现介绍。

## Kinect模块。

我们将Kinect识别模块，以及数据的提取部分，放在一个线程中，Kinect模块的实现逻辑主要如下：

1. 获取Kinect传感器设备，创建设备结构体指针，并开启设备：

IKinectSensor\* mySensor = nullptr;

GetDefaultKinectSensor(&mySensor);

mySensor->Open();

1. 创建彩色图像（可选）源，打开彩色图像读取器，读取图像源，并获取摄像头图像的格式、大小等：

IColorFrameSource\* myColorSource = nullptr;

mySensor->get\_ColorFrameSource(&myColorSource);

IColorFrameReader\* myColorReader = nullptr;

myColorSource->OpenReader(&myColorReader);

int colorHeight = 0, colorWidth = 0;

IFrameDescription\* myDescription = nullptr;

myColorSource->get\_FrameDescription(&myDescription);

myDescription->get\_Height(&colorHeight);

myDescription->get\_Width(&colorWidth);

IColorFrame\* myColorFrame = nullptr;

Mat original(colorHeight, colorWidth, CV\_8UC4, Scalar(0,0,0,0));

1. 创建人体骨骼图像源，打开读取器获取图像源：

IBodyFrameSource\* myBodySource = nullptr;

mySensor->get\_BodyFrameSource(&myBodySource);

IBodyFrameReader\* myBodyReader = nullptr;

myBodySource->OpenReader(&myBodyReader);

1. 创建myMapper坐标系转换匹配结构体指针并获取值，后续进行坐标转换使用：

ICoordinateMapper\* myMapper = nullptr;

mySensor->get\_CoordinateMapper(&myMapper);

1. 获取下位机的摄像头源并打开默认浏览器显示小车摄像头视频流:

ShellExecute(NULL, L"open", L"http://192.168.1.1:8080/javascript\_simple.html",NULL, NULL, SW\_SHOWNORMAL);

1. 等待读取身体骨骼图像帧成功：

while (myBodyReader->AcquireLatestFrame(&myBodyFrame) != S\_OK);

1. 创建骨骼数据数组：

IBody\*\* myBodyArr = new IBody \* [myBodyCount];

for (int i = 0; i < myBodyCount; i++)

myBodyArr[i] = nullptr;

1. 判断骨骼数据是否成功传入数组：

if (myBodyFrame->GetAndRefreshBodyData(myBodyCount, myBodyArr) == S\_OK)

1. 判断是否侦测到骨骼：

BOOLEAN result = false;

if (myBodyArr[i]->get\_IsTracked(&result) == S\_OK && result)

1. 若侦测到则创建一个用于存储骨骼数据点的数组，数组长度为JointType\_Count，然后定义手形HandState（全局）变量，获取手形：

Joint myJointArr[JointType\_Count];

HandState leftHandState = HandState\_Unknown;

HandState rightHandState = HandState\_Unknown;

myBodyArr[i]->get\_HandLeftState(&leftHandState);

myBodyArr[i]->get\_HandRightState(&rightHandState);

1. 判断是否成功将骨骼数据传入关节数组：

if (myBodyArr[i]->GetJoints(JointType\_Count, myJointArr) == S\_OK)

1. 如果传入成功，则进行画图，并输出左右手的视频坐标到全局变量中：

draw(copy, myJointArr[JointType\_Head], myJointArr[JointType\_Neck], myMapper);

……

draw(copy, myJointArr[JointType\_HandRight], myJointArr[JointType\_HandTipRight], myMapper);

Point lefthand, righthand;

ColorSpacePoint t\_point;

myMapper->MapCameraPointToColorSpace(myJointArr[JointType\_HandLeft].Position, &t\_point);

lefthand.x = t\_point.X;

lefthand.y = t\_point.Y;

myMapper->MapCameraPointToColorSpace(myJointArr[JointType\_HandRight].Position, &t\_point);

righthand.x = t\_point.X;

righthand.y = t\_point.Y;

1. 画出手形，如果张开则是绿色圆，如果闭合则是红色圆，如果是V字手则是蓝色圆：

int linethickness = 4;

int circleradius = 50;

if (leftHandState == HandState\_Open)

circle(copy, lefthand, circleradius, Vec3b(0, 255, 0), linethickness);//green = open

else if (leftHandState == HandState\_Closed)

circle(copy, lefthand, circleradius, Vec3b(0, 0, 255), linethickness);// red = closed

else if (leftHandState == HandState\_Lasso)

circle(copy, lefthand, circleradius, Vec3b(255, 0, 0), linethickness);// blue = lasso

if (rightHandState == HandState\_Open)

circle(copy, righthand, circleradius, Vec3b(0, 255, 0), linethickness);//green = open

else if (rightHandState == HandState\_Closed)

circle(copy, righthand, circleradius, Vec3b(0, 0, 255), linethickness);// red = closed

else if (rightHandState == HandState\_Lasso)

circle(copy, righthand, circleradius, Vec3b(255, 0, 0), linethickness);// blue = lasso

1. 其中draw绘图函数如下，需要进行坐标空间的转换：

void draw(Mat& img, Joint& r\_1, Joint& r\_2, ICoordinateMapper\* myMapper)

{

//用两个关节点来做线段的两端，并且进行状态过滤

if (r\_1.TrackingState==TrackingState\_Tracked&&r\_2.TrackingState==TrackingState\_Tracked)

{

ColorSpacePoint t\_point;

Point p\_1, p\_2;

myMapper->MapCameraPointToColorSpace(r\_1.Position, &t\_point);

p\_1.x = t\_point.X;

p\_1.y = t\_point.Y;

myMapper->MapCameraPointToColorSpace(r\_2.Position, &t\_point);

p\_2.x = t\_point.X;

p\_2.y = t\_point.Y;

line(img, p\_1, p\_2, Vec3b(0, 255, 0), 2);

circle(img, p\_1, 4, Vec3b(0, 255, 0), -1);

circle(img, p\_2, 4, Vec3b(0, 255, 0), -1);

}

}

1. 一个循环结束，显示图像并释放空间，准备下一次检测：

delete[]myBodyArr;

myBodyFrame->Release();

//myColorFrame->Release();

namedWindow("TEST",0);

cv::resizeWindow("TEST",1024,576);

cv::imshow("TEST", copy);

if (waitKey(30) == VK\_ESCAPE)

{

quitmark = TRUE;

break;

}

1. 当循环break出，释放其他的结构体变量，结束线程：

myMapper->Release();

myDescription->Release();

myColorReader->Release();

myColorSource->Release();

myBodyReader->Release();

myBodySource->Release();

mySensor->Close();

mySensor->Release();

return 0;

## 树莓派端的控制方法。

树莓派端，控制的对象是6个舵机、2个电机，舵机可以直接调用小R科技的舵机角度设置函数进行控制，所有的舵机都是180度舵机，可以直接利用函数进行设置；电机方面，可以控制继电器来控制输出的电压的正反转，同样是控制几个电平即可。

## 通信和算法指令模块。

我们采用TCP/Socket协议进行通信，上位机是采用C++代码，下位机是Python脚本，写在tcp\_ctrl.py文件中。

在上位机中，创建另一个线程，用于读取全局变量值，并进行周期性的指令输出，单独创建一个线程的目的是让通信模块不影响Kinect模块的检测频率，并可以放心地控制控制周期、延时等。

1. 首先创建SOCKET相关的变量创建SOCKET通信：

SOCKET s\_server;

int send\_len = 0;

SOCKADDR\_IN server\_addr;

initialization();//初始化套接字库

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.S\_un.S\_addr = inet\_addr("192.168.1.1");

server\_addr.sin\_port = htons(2222);

s\_server = socket(AF\_INET, SOCK\_STREAM, 0);

if (connect(s\_server, (SOCKADDR\*)&server\_addr, sizeof(SOCKADDR)) == SOCKET\_ERROR) {

cout << "服务器连接失败！" << endl;

//return 0;

WSACleanup();

}

else {

cout << "服务器连接成功！" << endl;

}

1. 然后根据设计的通信指令进行通信，发送6位的数据：

while (!quitmark)

{

int right\_y\_prev = righthand.y;

int right\_x\_prev = righthand.x;

Sleep(10);

if (leftHandState == HandState\_Closed) // motor ctrl

{

if (rightHandState == HandState\_Closed) //turn and go

{

if (abs(lefthand.y - righthand.y) <= TURN\_ERR) //差不多高 go

ss("030000");

else if (lefthand.y - righthand.y < -TURN\_ERR) //左手高 turn right

ss("020000");

else if (lefthand.y - righthand.y > TURN\_ERR) //右手高 turn left

ss("010000");

}

else if (rightHandState == HandState\_Open) // turn and back

{

if (abs(lefthand.y - righthand.y) <= TURN\_ERR) //差不多高 go

ss("040000");

else if (lefthand.y - righthand.y < -TURN\_ERR) //左手高 turn right

ss("020000");

else if (lefthand.y - righthand.y > TURN\_ERR) //右手高 turn left

ss("010000");

}

else ss("000000");

}

else if (leftHandState == HandState\_Lasso) // arm

{

if (rightHandState == HandState\_Closed)

ss("140000");

else if(rightHandState == HandState\_Open)

ss("240000");

Sleep(10);

if (rightHandState == HandState\_Lasso) // turn 0x03

{

int err\_y = (lefthand.y - righthand.y)/2;

if (err\_y < 0) //左手高 turn right

ss(to\_string(230000+abs(err\_y)));

else if (err\_y > 0) //右手高 turn left

ss(to\_string(130000 + abs(err\_y)));

}

else if (rightHandState == HandState\_Open || rightHandState == HandState\_Closed)

{

int err\_ry = righthand.y - right\_y\_prev;

err\_ry = (abs(err\_ry) < 12) ? abs(err\_ry) : 12; //limit

err\_ry = (err\_ry > 1) ? err\_ry : 0;

err\_ry = int(err\_ry / 2);

if (righthand.y - right\_y\_prev > 1) // up

ss(to\_string(210000 + err\_ry));

else if (righthand.y - right\_y\_prev < -1) //down

ss(to\_string(110000 + err\_ry));

if (righthand.x - right\_x\_prev > 2) // right

ss("020000");

else if (righthand.x - right\_x\_prev < -2) //left

ss("010000");

}

}

else if (leftHandState == HandState\_Open)

ss("000000");

}

1. 这些指令中，电机控制较为简单，是发送前进、后退、转弯等设置好的指令就行；而俯仰机械臂的舵机控制，我们采用的是每一个周期与上一周期的手坐标比较，来获取改变的位姿，然后经过处理，将命令、数据按格式发给下位机。

## 下位机代码详解。

1. Motol\_motion.py

设置舵机角度函数的编写，简单调用了内置好的舵机函数：

def set\_angle(index,angle):

XRservo.XiaoRGEEK\_SetServo(index,angle)

电机控制和IO口的初始化：

ENA=13

ENB=20

IN1=19

IN2=16

IN3=21

IN4=26

def motion\_initial():

GPIO.setup(ENA,GPIO.OUT,initial=GPIO.LOW)

GPIO.setup(ENB,GPIO.OUT,initial=GPIO.LOW)

GPIO.setup(IN1,GPIO.OUT,initial=GPIO.LOW)

GPIO.setup(IN2,GPIO.OUT,initial=GPIO.LOW)

GPIO.setup(IN3,GPIO.OUT,initial=GPIO.LOW)

GPIO.setup(IN4,GPIO.OUT,initial=GPIO.LOW)

motion\_initial()

前进、后退、停止、转弯函数的定义：

def gogo():

GPIO.output(ENA,True)

GPIO.output(IN1,True)

GPIO.output(IN2,False)

GPIO.output(ENB,True)

GPIO.output(IN3,True)

GPIO.output(IN4,False)

def back():

GPIO.output(ENA,True)

GPIO.output(IN1,False)

GPIO.output(IN2,True)

GPIO.output(ENB,True)

GPIO.output(IN3,False)

GPIO.output(IN4,True)

def stop():

GPIO.output(ENA,False)

GPIO.output(IN1,False)

GPIO.output(IN2,False)

GPIO.output(ENB,False)

GPIO.output(IN3,False)

GPIO.output(IN4,False)

def left\_motol(x): ##0:stop 1:forward -1:back

GPIO.output(ENA,x)

if x==0:

GPIO.output(IN1,0)

GPIO.output(IN2,0)

elif x==1:

GPIO.output(IN1,1)

GPIO.output(IN2,0)

else:

GPIO.output(IN1,0)

GPIO.output(IN2,1)

def right\_motol(x): ##0:stop 1:forward -1:back

GPIO.output(ENB,x)

if x==0:

GPIO.output(IN3,0)

GPIO.output(IN4,0)

elif x==1:

GPIO.output(IN3,1)

GPIO.output(IN4,0)

else:

GPIO.output(IN3,0)

GPIO.output(IN4,1)

电机初始化、舵机初始化、车载摄像头启动：

motion\_initial()

angle=[0,130,175,90,100,0,0,90,0] ## init

XRservo.XiaoRGEEK\_ReSetServo() ##reset angle

os.system("cd ~/work/mjpg-streamer-experimental-xr")

try:

os.system("./mjpg\_streamer -i \"./input\_raspicam.so\" -o \"./output\_http.so -w ./www\"")

except:

print("video was opened or failed")

接收客户端连接：

print("Wait For Connect....")

client, address = mySocket.accept()

print("Connected.")

print("IP is %s" % address[0])

print("port is %d\n" % address[1])

解码客户端传输的数据，并根据指令执行：

msg = client.recv(1024)

stop()

print(msg.decode('EUC-JP'))

#把接收到的数据进行解码

##print(msg.encode("utf-8"))

msg1=msg[0:2]

##print(chardet.detect(msg))

try:

msg2=int(msg[2:6])

except:

msg2=0

#print("message received")

转弯代码（原地旋转）：

elif msg1 == b"00": ##00:stop

right\_motol(0)

left\_motol(0)

elif msg1 == b"01": ##01:left

left\_motol(-1)

right\_motol(1)

time.sleep(0.02)

elif msg1 == b"02": ##02:right

right\_motol(-1)

left\_motol(1)

time.sleep(0.02)

前进、后退代码，此处还添加了摄像头舵机控制代码，如果前进摄像头朝前，如果倒车摄像头朝后，方便远程的控制！

elif msg1 == b"03": ##03:go

right\_motol(1)

left\_motol(1)

time.sleep(0.02)

angle[7]=180

angle[8]=25

set\_angle(0x07,angle[7])

set\_angle(0x08,angle[8])

elif msg1 == b"04": ##04:back

right\_motol(-1)

left\_motol(-1)

time.sleep(0.02)

angle[7]=0

angle[8]=25

set\_angle(0x07,angle[7])

set\_angle(0x08,angle[8])

机械手前两个关节的俯仰，为了简便我们进行了耦合控制，即转完一个电机再转另一个：

elif msg1 == b"11": ##up(inc) 1&2 1:up(130) low(45) 2: up(175) low(135)

if (angle[2]==175):## move 1

if (angle[1]+msg2<130):

angle[1]=angle[1]+msg2

else:

angle[1]=130

angle[8]=10+40\*(angle[1]-45)/85

else:

if (angle[2]+msg2<175):

angle[2]=angle[2]+msg2

else:

angle[2]=175

angle[8]=(angle[2]-135)/4

angle[7]=180

set\_angle(0x08,angle[8])

set\_angle(0x07,angle[7])

set\_angle(0x01,angle[1])

set\_angle(0x02,angle[2])

elif msg1 == b"21": ##down(dec) 1&2

if (angle[1]==45):## move 2

if (angle[2]-msg2>135):

angle[2]=angle[2]-msg2

else:

angle[2]=135

angle[8]=(angle[2]-135)/4

else:

if (angle[1]-msg2>45):

angle[1]=angle[1]-msg2

else:

angle[1]=45

angle[8]=10+40\*(angle[1]-45)/85

angle[7]=180

set\_angle(0x08,angle[8])

set\_angle(0x07,angle[7])

set\_angle(0x01,angle[1])

set\_angle(0x02,angle[2])

舵机第三自由度的旋转、机械手的开合代码：

elif msg1 == b"13": ##left(dec) 3

if (90-msg2)>=0:

angle[3]=90-msg2

else:

angle[3]=0

set\_angle(0x03,angle[3])

elif msg1 == b"23": ##right(inc) 3

if (90+msg2)<=180:

angle[3]=90+msg2

else:

angle[3]=180

set\_angle(0x03,angle[3])

elif msg1 == b"14": ##he(inc 150) 4 100~150

angle[4] = 150

set\_angle(0x04,angle[4])

elif msg1 == b"24": ##zhang(dec 100) 4

angle[4] = 100

set\_angle(0x04,angle[4])

1. 项目总结

## 项目的亮点：

1. **可以模拟体感开车状态，旋转手臂来控制小车转弯；**
2. **可以用手臂俯仰来控制机械手俯仰；**
3. **可以用手掌开合来控制机械手的开合状态；**
4. **不同状态下，摄像头的位姿会随动：**

**前进、后退时摄像头分别朝前和朝后；机械臂俯仰时，摄像头会追随机械手的位置俯仰。**

感想：

本次项目有大量需要自学的内容，第一次接触kinect模块，从环境配置到函数理解都花费了不少时间但也收获颇丰。本次项目涉及知识面较广，需要PC机和下位机分别编写程序，分别用到了两种编程语言（C++ ，PYTHON)。通信方面则是第一次接触到Socket，也耗费了很多时间去学习相关内容。用到了7款不同的软件和各类硬件，工作量较大。CSDN和github以及官方指导文件是非常好的学习资源，非常适合有大量自学内容的项目设计。感谢老师的指导以及同组同学的配合。

展望：

在未来我们希望可以进一步提高小车的控制精度，并且可能的话增加可用于控制的关节而不仅限于手势。目前Kincet的骨骼识别功能主要局限于二维平面，在未来可以进一步完善算法，增加深度属性，这样可以大幅度降低手势的复杂度，并且各种控制姿势的设计也可以更加符合人体力学，增加控制的舒适度。

## 附：代码及演示视频

## 全部C++代码。

#include <iostream>

#include <opencv2\imgproc.hpp>

#include <opencv2\calib3d.hpp>

#include <opencv2\highgui.hpp>

#include <Kinect.h>

#include<codecvt>

#include<winsock.h>

#include<string>

#include <math.h>

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

#include <pthread.h>

#include <windows.h>//使用Sleep的头

using namespace std;

using namespace cv;

int g\_number = 0;

#define TURN\_ERR 100

int ss(string x);

void initialization();

void draw(Mat& img, Joint& r\_1, Joint& r\_2, ICoordinateMapper\* myMapper);

SOCKET s\_server;

HandState leftHandState = HandState\_Unknown; // 手形

HandState rightHandState = HandState\_Unknown;

Point lefthand, righthand;

boolean startmark = FALSE;

boolean quitmark = FALSE;

void\* counter1(void\* args) //socket

{

int send\_len = 0;

SOCKADDR\_IN server\_addr;

initialization();

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.S\_un.S\_addr = inet\_addr("192.168.1.1");

server\_addr.sin\_port = htons(2222);

s\_server = socket(AF\_INET, SOCK\_STREAM, 0);

if (connect(s\_server, (SOCKADDR\*)&server\_addr, sizeof(SOCKADDR)) == SOCKET\_ERROR) {

cout << "服务器连接失败！" << endl;

//return 0;

WSACleanup();

}

else {

cout << "服务器连接成功！" << endl;

}

while (!startmark);

Sleep(100);

while (!quitmark)

{

int right\_y\_prev = righthand.y;

int right\_x\_prev = righthand.x;

Sleep(10);

if (leftHandState == HandState\_Closed) // motor ctrl

{

if (rightHandState == HandState\_Closed) //turn and go

{

if (abs(lefthand.y - righthand.y) <= TURN\_ERR) {//差不多高 go

ss("030000");

}

else if (lefthand.y - righthand.y < -TURN\_ERR) {//左手高 turn right

ss("020000");

}

else if (lefthand.y - righthand.y > TURN\_ERR) {//右手高 turn left

ss("010000");

}

}

else if (rightHandState == HandState\_Open) // turn and back

{

if (abs(lefthand.y - righthand.y) <= TURN\_ERR) {//差不多高 go

ss("040000");

}

else if (lefthand.y - righthand.y < -TURN\_ERR) {//左手高 turn right

ss("020000");

}

else if (lefthand.y - righthand.y > TURN\_ERR) {//右手高 turn left

ss("010000");

}

}

else ss("000000");

}

else if (leftHandState == HandState\_Lasso) // arm

{

//ss("000000");

if (rightHandState == HandState\_Closed)

ss("140000");

else if(rightHandState == HandState\_Open)

ss("240000");

/\*4\*/

Sleep(10);

if (rightHandState == HandState\_Lasso) // turn 0x03

{

int err\_y = (lefthand.y - righthand.y)/2;

if (err\_y < 0) {//左手高 turn right

ss(to\_string(230000+abs(err\_y)));

}

else if (err\_y > 0) {//右手高 turn left

ss(to\_string(130000 + abs(err\_y)));

}

}

else if (rightHandState == HandState\_Open || rightHandState == HandState\_Closed) // up down 1 2

{

int err\_ry = righthand.y - right\_y\_prev;

int err\_rx = righthand.x - right\_x\_prev;

err\_ry = (abs(err\_ry) < 12) ? abs(err\_ry) : 12; //limit

err\_ry = (err\_ry > 1) ? err\_ry : 0;

err\_ry = int(err\_ry / 2);

if (righthand.y - right\_y\_prev > 1) // up

{

ss(to\_string(210000 + err\_ry));

printf("21:%d\n",err\_ry);

}

else if (righthand.y - right\_y\_prev < -1) //down

{

ss(to\_string(110000 + err\_ry));

printf("11:%d\n", err\_ry);

}

if (righthand.x - right\_x\_prev > 2) // right

{

ss("020000");

}

else if (righthand.x - right\_x\_prev < -2) //left

{

ss("010000");

}

}

}

else if (leftHandState == HandState\_Open)

{

ss("000000");

}

else

{

}

}

ss("000000");

Sleep(10);

ss("qq");

Sleep(100);

closesocket(s\_server);

WSACleanup();

return 0;

}

void\* counter2(void\* args) {

startmark = TRUE;

IKinectSensor\* mySensor = nullptr;

GetDefaultKinectSensor(&mySensor);

mySensor->Open();

IColorFrameSource\* myColorSource = nullptr;

mySensor->get\_ColorFrameSource(&myColorSource);

IColorFrameReader\* myColorReader = nullptr;

myColorSource->OpenReader(&myColorReader);

int colorHeight = 0, colorWidth = 0;

IFrameDescription\* myDescription = nullptr;

myColorSource->get\_FrameDescription(&myDescription);

myDescription->get\_Height(&colorHeight);

myDescription->get\_Width(&colorWidth);

IColorFrame\* myColorFrame = nullptr;

Mat original(colorHeight, colorWidth, CV\_8UC4, Scalar(0,0,0,0));

cout << colorHeight << ";" << colorWidth << endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*以上为ColorFrame的读取前准备\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IBodyFrameSource\* myBodySource = nullptr;

mySensor->get\_BodyFrameSource(&myBodySource);

IBodyFrameReader\* myBodyReader = nullptr;

myBodySource->OpenReader(&myBodyReader);

int myBodyCount = 0;

myBodySource->get\_BodyCount(&myBodyCount);

IBodyFrame\* myBodyFrame = nullptr;

ICoordinateMapper\* myMapper = nullptr;

mySensor->get\_CoordinateMapper(&myMapper);

ShellExecute(NULL, L"open", L"http://192.168.1.1:8080/javascript\_simple.html",NULL, NULL, SW\_SHOWNORMAL);

while (1)

{

//while (myColorReader->AcquireLatestFrame(&myColorFrame) != S\_OK);

//myColorFrame->CopyConvertedFrameDataToArray(colorHeight \* colorWidth \* 4, original.data, ColorImageFormat\_Bgra);

Mat copy = original.clone();

while (myBodyReader->AcquireLatestFrame(&myBodyFrame) != S\_OK);

IBody\*\* myBodyArr = new IBody \* [myBodyCount];

for (int i = 0; i < myBodyCount; i++)

myBodyArr[i] = nullptr;

if (myBodyFrame->GetAndRefreshBodyData(myBodyCount, myBodyArr) == S\_OK)

for (int i = 0; i < myBodyCount; i++)

{

BOOLEAN result = false;

if (myBodyArr[i]->get\_IsTracked(&result) == S\_OK && result)

{

Joint myJointArr[JointType\_Count];

myBodyArr[i]->get\_HandLeftState(&leftHandState);

myBodyArr[i]->get\_HandRightState(&rightHandState);

if (myBodyArr[i]->GetJoints(JointType\_Count, myJointArr) == S\_OK)

{

draw(copy, myJointArr[JointType\_Head], myJointArr[JointType\_Neck], myMapper);

draw(copy, myJointArr[JointType\_Neck], myJointArr[JointType\_SpineShoulder], myMapper);

draw(copy, myJointArr[JointType\_SpineShoulder], myJointArr[JointType\_ShoulderLeft], myMapper);

draw(copy, myJointArr[JointType\_SpineShoulder], myJointArr[JointType\_SpineMid], myMapper);

draw(copy, myJointArr[JointType\_SpineShoulder], myJointArr[JointType\_ShoulderRight], myMapper);

draw(copy, myJointArr[JointType\_ShoulderLeft], myJointArr[JointType\_ElbowLeft], myMapper);

draw(copy, myJointArr[JointType\_SpineMid], myJointArr[JointType\_SpineBase], myMapper);

draw(copy, myJointArr[JointType\_ShoulderRight], myJointArr[JointType\_ElbowRight], myMapper);

draw(copy, myJointArr[JointType\_ElbowLeft], myJointArr[JointType\_WristLeft], myMapper);

draw(copy, myJointArr[JointType\_SpineBase], myJointArr[JointType\_HipLeft], myMapper);

draw(copy, myJointArr[JointType\_SpineBase], myJointArr[JointType\_HipRight], myMapper);

draw(copy, myJointArr[JointType\_ElbowRight], myJointArr[JointType\_WristRight], myMapper);

draw(copy, myJointArr[JointType\_WristLeft], myJointArr[JointType\_ThumbLeft], myMapper);

draw(copy, myJointArr[JointType\_WristLeft], myJointArr[JointType\_HandLeft], myMapper);

draw(copy, myJointArr[JointType\_HipLeft], myJointArr[JointType\_KneeLeft], myMapper);

draw(copy, myJointArr[JointType\_HipRight], myJointArr[JointType\_KneeRight], myMapper);

draw(copy, myJointArr[JointType\_WristRight], myJointArr[JointType\_ThumbRight], myMapper);

draw(copy, myJointArr[JointType\_WristRight], myJointArr[JointType\_HandRight], myMapper);

draw(copy, myJointArr[JointType\_HandLeft], myJointArr[JointType\_HandTipLeft], myMapper);

draw(copy, myJointArr[JointType\_KneeLeft], myJointArr[JointType\_FootLeft], myMapper);

draw(copy, myJointArr[JointType\_KneeRight], myJointArr[JointType\_FootRight], myMapper);

draw(copy, myJointArr[JointType\_HandRight], myJointArr[JointType\_HandTipRight], myMapper);

ColorSpacePoint t\_point;

myMapper->MapCameraPointToColorSpace(myJointArr[JointType\_HandLeft].Position, &t\_point);

lefthand.x = t\_point.X;

lefthand.y = t\_point.Y;

myMapper->MapCameraPointToColorSpace(myJointArr[JointType\_HandRight].Position, &t\_point);

righthand.x = t\_point.X;

righthand.y = t\_point.Y;

// cout << righthand.x << "," << righthand.y << endl; // 坐标系测试：坐标原点是视频左上角，x是到竖轴的距离，y是到横轴的距离

int linethickness = 4;

int circleradius = 50;

if (leftHandState == HandState\_Open) {

circle(copy, lefthand, circleradius, Vec3b(0, 255, 0), linethickness);//green = open

}

else if (leftHandState == HandState\_Closed) {

circle(copy, lefthand, circleradius, Vec3b(0, 0, 255), linethickness);// red = closed

}

else if (leftHandState == HandState\_Lasso) {

circle(copy, lefthand, circleradius, Vec3b(255, 0, 0), linethickness);// blue = lasso

}

if (rightHandState == HandState\_Open)

circle(copy, righthand, circleradius, Vec3b(0, 255, 0), linethickness);//green = open

else if (rightHandState == HandState\_Closed)

circle(copy, righthand, circleradius, Vec3b(0, 0, 255), linethickness);// red = closed

else if (rightHandState == HandState\_Lasso)

circle(copy, righthand, circleradius, Vec3b(255, 0, 0), linethickness);// blue = lasso

}

}

}

delete[]myBodyArr;

myBodyFrame->Release();

//myColorFrame->Release();

namedWindow("TEST",0);

cv::resizeWindow("TEST",1024,576);

cv::imshow("TEST", copy);

if (waitKey(30) == VK\_ESCAPE)

{

quitmark = TRUE;

break;

}

}

myMapper->Release();

myDescription->Release();

myColorReader->Release();

myColorSource->Release();

myBodyReader->Release();

myBodySource->Release();

mySensor->Close();

mySensor->Release();

return 0;

}

int main() {

pthread\_t t1;

pthread\_t t2;

pthread\_create(&t1, NULL, counter1, NULL);

pthread\_create(&t2, NULL, counter2, NULL);

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

return 0;

}

void draw(Mat& img, Joint& r\_1, Joint& r\_2, ICoordinateMapper\* myMapper)

{

//用两个关节点来做线段的两端，并且进行状态过滤

if (r\_1.TrackingState == TrackingState\_Tracked && r\_2.TrackingState == TrackingState\_Tracked)

{

ColorSpacePoint t\_point; //要把关节点用的摄像机坐标下的点转换成彩色空间的点

Point p\_1, p\_2;

myMapper->MapCameraPointToColorSpace(r\_1.Position, &t\_point);

p\_1.x = t\_point.X;

p\_1.y = t\_point.Y;

myMapper->MapCameraPointToColorSpace(r\_2.Position, &t\_point);

p\_2.x = t\_point.X;

p\_2.y = t\_point.Y;

line(img, p\_1, p\_2, Vec3b(0, 255, 0), 2);

circle(img, p\_1, 4, Vec3b(0, 255, 0), -1);

circle(img, p\_2, 4, Vec3b(0, 255, 0), -1);

}

}

void initialization() {

//初始化套接字库

WORD w\_req = MAKEWORD(2, 2);//版本号

WSADATA wsadata;

int err;

err = WSAStartup(w\_req, &wsadata);

if (err != 0) {

cout << "初始化套接字库失败！" << endl;

}

else {

cout << "初始化套接字库成功！" << endl;

}

//检测版本号

if (LOBYTE(wsadata.wVersion) != 2 || HIBYTE(wsadata.wHighVersion) != 2) {

cout << "套接字库版本号不符！" << endl;

WSACleanup();

}

else {

cout << "套接字库版本正确！" << endl;

}

//填充服务端地址信息

}

int ss(string x) {

int y;

y = send(s\_server, x.c\_str(), strlen(x.c\_str()), 0);

return y;

}

