**Socket通信**

利用socket和UDP协议进行电脑摄像头画面和图像识别结果的实时传送：分为send客户端和receive服务器端。

通过信号量flag的值在两个进程之间的传递来决定是否选择对视频中的障碍物进行识别，flag值为1表示识别障碍物，flag值为0表示关闭识别障碍物。两个进程之间的通信通过UDP传输连接进行，在Qt中的编程表现为发送方点击关闭或识别障碍物相关信号量利用QUdpSocket对象udpSocket函数writeDatagram传输数据，包含参数：信号量、目标ip地址以及接口，信号量以QByteArray格式封装；接收进程同样使用QUdpSocket对象receiver绑定对应的接口通过readDatagram接收信号量数据，来判断是否开启障碍物识别。

一．Send客户端：

（VideoSend()函数）

1. 获取本机的IP地址和端口号

2、使用OpenCv VideoCapture类 + Qtimer定时器实现简单的摄像头调用

3、将获取的摄像头数据利用QImage类+ QPixmap类实现在QLabel中的展示。

4、利用字节数组QByteArray类和数据缓冲区QBuffer类实现数据的存储：

5、利用QUdpSocket类发送UDP数据报，实现摄像画面和图像识别结果的实时传输。

（video\_receive\_show()函数）

和receive客户端的接收函数一样，用于在开启障碍物识别后返回具体的识别精度信息和识别后的画面，实现信息反馈功能。当关闭识别功能时，send客户端画面恢复识别前画面。

二．Receive客户端：（video\_receive\_show()函数）

1、利用QUdpSocket类接收UDP数据报，实现摄像画面的接收。

2、利用字节数组QByteArray类和数据缓冲区QBuffer类实现数据的接收和存储。QImageReader类实现设备文件中图像以及其他图像数据的读入。

3、利用QLabel进行摄像头画面的实时显示。

**模型生成训练**

主要使用了yolov5源码的train.py、test.py、detect.py、export.py等代码文件。

先要配置yolov5算法的pytorch运行环境以及导入相关依赖库，之后进行数据的收集打标签形成自己的数据集，在yolov5-master同级目录下创建datasets将数据集放入其中，根据数据集创建yaml训练数据配置文件。

在train.py中引用自己创建的obstacle.yaml、设置训练图像大小以及簇数，通过在终端引用命令行语句进行模型训练，产生best.pt权重文件，使用detect.py进行验证。最后在export.py文件中设置相应参数将best.pt文件转化为能被c++引用的onnx文件进行模式识别。

**模型识别检测**

完成模型的识别检测的四个函数

1、Yolov5::LetterBox：

分割图片，获取数据集中训练模型标签基本信息，物体的中心位置，物体的长宽（h,w），标签边界四个点坐标等信息

2、Yolov5::readModel：

读取权重文件，加载模型，获取模型的基本信息,训练权重文件

3、Yolov5::Detect：

测试模型，识别图片中的物体及类别，获取预测框含有目标的置信度和预测框预测

4、Yolov5::drawPred：

根据训练结果，画出标签框