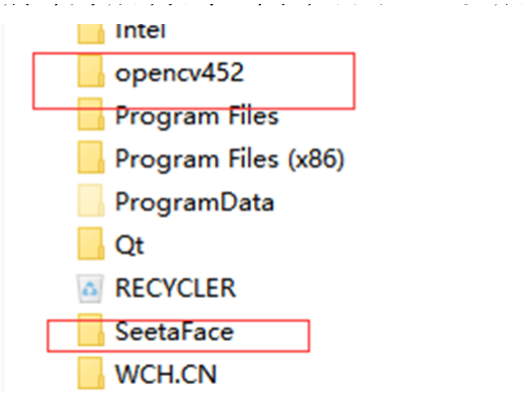
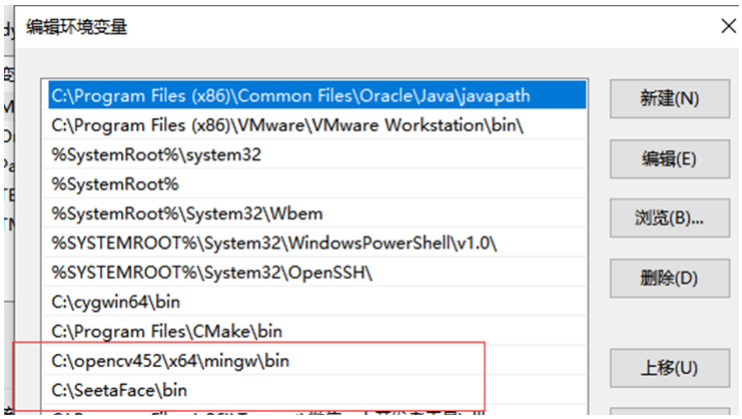
一、OpenCV和seetaface环境搭建

window已经编译好的库opencv452.zip
window已经编译好的库SeetaFace.zip

将上面的压缩包解压到C盘



把两个库的bin目录添加到环境变量中



创建一个Qt并且修改工程文件添加，opencv和seetaface的头文件和库路径

#添加opencv，seetaface头文件

INCLUDEPATH += C:\opencv452\include

INCLUDEPATH += C:\opencv452\include\opencv2

INCLUDEPATH += C:\SeetaFace\include

INCLUDEPATH += C:\SeetaFace\include\seeta

#添加opencv，seetace的库LIBS+=C:\opencv452\x64\mingw\lib\libopencv*

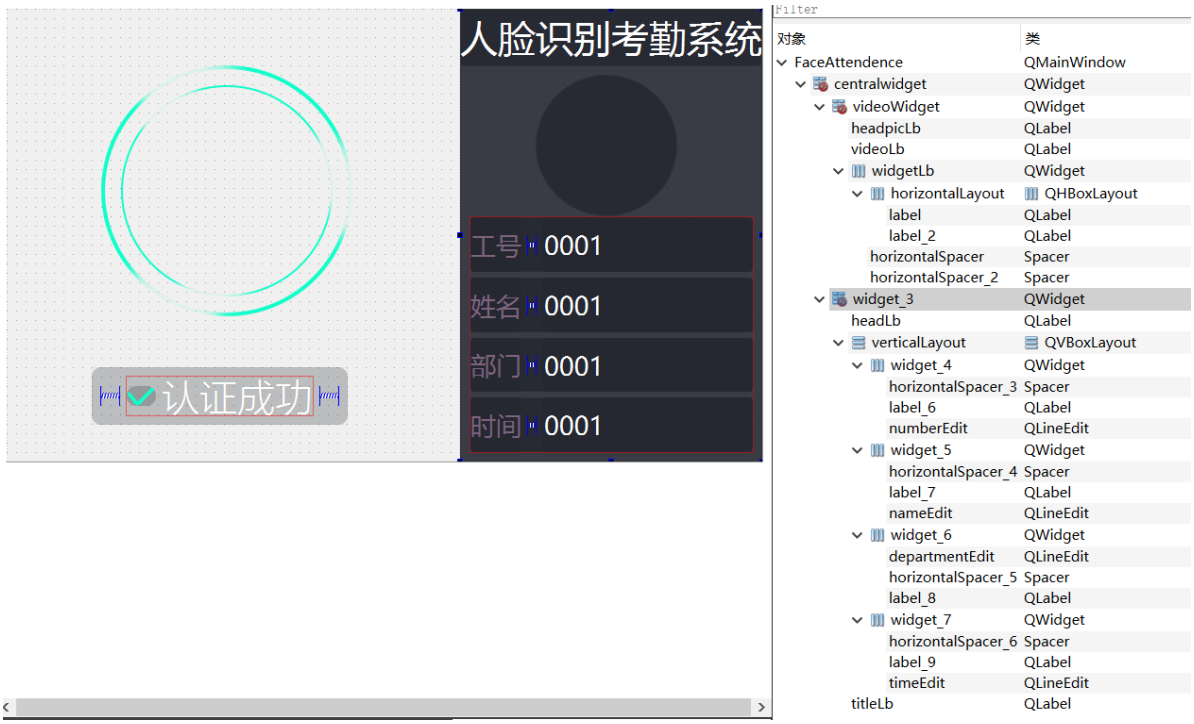
LIBS+=C:\SeetaFace\lib\libSeeta*

代码

```
#include "mainwindow.h"

#include <QApplication>
```

```
#include <opencv.hpp>
#include <FaceDetector.h>
using namespace  cv;
using namespace  seeta::v2;
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    cv::namedWindow("frame");
    Mat mt = imread("./1.jpg");
    imshow("frame",mt);


    seeta::ModelSetting::Device device = seeta::ModelSetting::CPU;
    int id = 0;
    seeta::ModelSetting FD_model( "C:/SeetaFace/bin/model/fd_2_00.dat", device,
id );
    seeta::FaceDetector FD(FD_model);

    return a.exec();
}
```

二、终端界面设计



三、摄像头数据采集显示

四、OpenCV人脸检测和显示

五、人脸框显示跟踪

```
#ifndef FACEATTENDENCE_H
#define FACEATTENDENCE_H

#include <QMainWindow>
#include <opencv.hpp>
using namespace cv;
using namespace std;

QT_BEGIN_NAMESPACE
namespace Ui { class FaceAttendence; }
QT_END_NAMESPACE

class FaceAttendence : public QMainWindow
{
    Q_OBJECT

public:
    FaceAttendence(QWidget *parent = nullptr);
    ~FaceAttendence();
    //定时器事件
    void timerEvent(QTimerEvent *e);

private:
    Ui::FaceAttendence *ui;

    //摄像头
    VideoCapture cap;
    //haar--级联分类器
    cv::CascadeClassifier cascade;

};
#endif // FACEATTENDENCE_H
```

```
#include "faceattendence.h"
#include "ui_faceattendence.h"
#include <QImage>
#include <QPainter>
#include <QDebug>
FaceAttendence::FaceAttendence(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::FaceAttendence)
{
    ui->setupUi(this);
    //打开摄像头
    cap.open(0);//dev/video
    //启动定时器事件
    startTimer(100);

    //导入级联分类器文件
```

```
 cascade.load("C:/opencv452/etc/haarcascades/haarcascade_frontalface_alt2.xml");
}

FaceAttendence::~FaceAttendence()
{
    delete ui;
}

void FaceAttendence::timerEvent(QTimerEvent *e)
{
    //采集数据
    Mat srcImage;
    if(cap.grab())
    {
        cap.read(srcImage);//读取一帧数据
    }

    Mat grayImage;
    //转灰度图
    cvtColor(srcImage, grayImage, COLOR_BGR2GRAY);
    //检测人脸数据
    std::vector<Rect> faceRects;
    cascade.detectMultiScale(grayImage,faceRects);
    if(faceRects.size()>0)
    {
        Rect rect = faceRects.at(0);//第一个人脸的矩形框
        //rectangle(srcImage,rect,Scalar(0,0,255));
        //移动人脸框（图片--QLabel）
        ui->headpicLb->move(rect.x,rect.y);
    }else
    {
        //把人脸框移动到中心位置
        ui->headpicLb->move(100,60);
    }


    if(srcImage.data == nullptr) return;
    //把opencv里面的Mat格式数据（BGR）转Qt里面的QImage(RGB)
    cvtColor(srcImage,srcImage, COLOR_BGR2RGB);
    QImage image(srcImage.data,srcImage.cols,
srcImage.rows,srcImage.step1(),QImage::Format_RGB888);
    QPixmap mmp = QPixmap::fromImage(image);
    ui->videoLb->setPixmap(mmp);
}
```

六、网络连接

```
#ifndef FACEATTENDENCE_H
#define FACEATTENDENCE_H

#include <QMainWindow>
#include <opencv.hpp>
#include <QTcpSocket>
#include <QTimer>
```

```
using namespace cv;
using namespace std;

QT_BEGIN_NAMESPACE
namespace Ui { class FaceAttendence; }
QT_END_NAMESPACE

class FaceAttendence : public QMainWindow
{
    Q_OBJECT
public:
    FaceAttendence(QWidget *parent = nullptr);
    ~FaceAttendence();
    //定时器事件
    void timerEvent(QTimerEvent *e);

private slots:
    void timer_connect();
    void stop_connect();
    void start_connect();


private:
    Ui::FaceAttendence *ui;

    //摄像头
    VideoCapture cap;
    //haar--级联分类器
    cv::CascadeClassifier cascade;

    //创建网络套接字，定时器
    QTcpSocket msocket;
    QTimer mtimer;

};
#endif // FACEATTENDENCE_H
```

```
#include "faceattendence.h"
#include "ui_faceattendence.h"
#include <QImage>
#include <QPainter>
#include <QDebug>
FaceAttendence::FaceAttendence(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::FaceAttendence)
{
    ui->setupUi(this);
    //打开摄像头
    cap.open(0);//dev/video
    //启动定时器事件
    startTimer(100);

    //导入级联分类器文件

 cascade.load("C:/opencv452/etc/haarcascades/haarcascade_frontalface_alt2.xml");
```

```
    //QTcpSocket当断开连接的时候disconnected信号，连接成功会发送connected
    connect(&msocket,&QTcpSocket::disconnected,this,
&FaceAttendence::start_connect);
    connect(&msocket,&QTcpSocket::connected,this,
&FaceAttendence::stop_connect);


    //定时器连接服务器
    connect(&mtimer, &QTimer::timeout,this,&FaceAttendence::timer_connect);
    //启动定时器
    mtimer.start(5000);//每5s钟连接一次，直到连接成功就不在连接
}

FaceAttendence::~FaceAttendence()
{
    delete ui;
}

void FaceAttendence::timerEvent(QTimerEvent *e)
{
    //采集数据
    Mat srcImage;
    if(cap.grab())
    {
        cap.read(srcImage);//读取一帧数据
    }

    Mat grayImage;
    //转灰度图
    cvtColor(srcImage, grayImage, COLOR_BGR2GRAY);
    //检测人脸数据
    std::vector<Rect> faceRects;
    cascade.detectMultiScale(grayImage,faceRects);
    if(faceRects.size()>0)
    {
        Rect rect = faceRects.at(0);//第一个人脸的矩形框
        //rectangle(srcImage,rect,Scalar(0,0,255));
        //移动人脸框（图片--QLabel）
        ui->headpicLb->move(rect.x,rect.y);
    }else
    {
        //把人脸框移动到中心位置
        ui->headpicLb->move(100,60);
    }

    if(srcImage.data == nullptr) return;
    //把opencv里面的Mat格式数据（BGR）转Qt里面的QImage(RGB)
    cvtColor(srcImage,srcImage, COLOR_BGR2RGB);
    QImage image(srcImage.data,srcImage.cols,
srcImage.rows,srcImage.step1(),QImage::Format_RGB888);
    QPixmap mmp = QPixmap::fromImage(image);
    ui->videoLb->setPixmap(mmp);
}
```

```
void FaceAttendence::timer_connect()
{
    //连接服务器
    msocket.connectToHost("127.0.0.1",9999);
    qDebug()<<"正在连接服务器";
}

void FaceAttendence::stop_connect()
{
    mtimer.stop();
    qDebug()<<"成功连接服务器";
}

void FaceAttendence::start_connect()
{
    mtimer.start(5000);//启动定时器
    qDebug()<<"断开连接";
}
```

七、服务器连接实现

```
#ifndef ATTENDANCEWIN_H
#define ATTENDANCEWIN_H

#include <QMainWindow>
#include <QTcpSocket>
#include <QTcpServer>

QT_BEGIN_NAMESPACE
namespace Ui { class AttendanceWin; }
QT_END_NAMESPACE

class AttendanceWin : public QMainWindow
{
    Q_OBJECT

public:
    AttendanceWin(QWidget *parent = nullptr);
    ~AttendanceWin();

protected slots:
    void accept_client();
    void read_data();
private:
    Ui::AttendanceWin *ui;
    QTcpServer mserver;
    QTcpSocket *msocket;
};
#endif // ATTENDANCEWIN_H
```

```
#include "attendancewin.h"
#include "ui_attendancewin.h"
```

```
AttendanceWin::AttendanceWin(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::AttendanceWin)
{
    ui->setupUi(this);
    //qtcpServer当有客户端连会发送newconnection
    connect(&mserver, &QTcpServer::newConnection, this,
&AttendanceWin::accept_client);
    mserver.listen(QHostAddress::Any,9999);//监听，启动服务器
}

AttendanceWin::~AttendanceWin()
{
    delete ui;
}

//接受客户端连接
void AttendanceWin::accept_client()
{
    //获取与客户端通信的套接字
    msocket = mserver.nextPendingConnection();

    //当客户端有数据到达会发送readyRead信号
    connect(msocket,&QTcpSocket::readyRead,this,&AttendanceWin::read_data);
}

//读取客户端发送的数据
void AttendanceWin::read_data()
{
    //读取所有的数据
    QString msg = msocket->readAll();
    qDebug()<<msg;
}
```

八、客户端传送图像到服务器显示

客户端：

```
#ifndef FACEATTENDENCE_H
#define FACEATTENDENCE_H

#include <QMainWindow>
#include <opencv.hpp>
#include <QTcpSocket>
#include <QTimer>
using namespace cv;
using namespace std;

QT_BEGIN_NAMESPACE
namespace Ui { class FaceAttendence; }
```

```
QT_END_NAMESPACE

class FaceAttendence : public QMainWindow
{
    Q_OBJECT
public:
    FaceAttendence(QWidget *parent = nullptr);
    ~FaceAttendence();
    //定时器事件
    void timerEvent(QTimerEvent *e);

private slots:
    void timer_connect();
    void stop_connect();
    void start_connect();


private:
    Ui::FaceAttendence *ui;

    //摄像头
    VideoCapture cap;
    //haar--级联分类器
    cv::CascadeClassifier cascade;

    //创建网络套接字，定时器
    QTcpSocket msocket;
    QTimer mtimer;

};
#endif // FACEATTENDENCE_H
```

```
#include "faceattendence.h"
#include "ui_faceattendence.h"
#include <QImage>
#include <QPainter>
#include <QDebug>
FaceAttendence::FaceAttendence(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::FaceAttendence)
{
    ui->setupUi(this);
    //打开摄像头
    cap.open(0);//dev/video
    //启动定时器事件
    startTimer(100);

    //导入级联分类器文件

 cascade.load("C:/opencv452/etc/haarcascades/haarcascade_frontalface_alt2.xml");

    //QTcpSocket当断开连接的时候disconnected信号，连接成功会发送connected
    connect(&msocket,&QTcpSocket::disconnected,this,
&FaceAttendence::start_connect);
```

```
    connect(&msocket,&QTcpSocket::connected,this,
&FaceAttendence::stop_connect);

    //定时器连接服务器
    connect(&mtimer, &QTimer::timeout,this,&FaceAttendence::timer_connect);
    //启动定时器
    mtimer.start(5000);//每5s钟连接一次，直到连接成功就不在连接
}

FaceAttendence::~FaceAttendence()
{
    delete ui;
}

void FaceAttendence::timerEvent(QTimerEvent *e)
{
    //采集数据
    Mat srcImage;
    if(cap.grab())
    {
        cap.read(srcImage);//读取一帧数据
    }

    Mat grayImage;
    //转灰度图
    cvtColor(srcImage, grayImage, COLOR_BGR2GRAY);
    //检测人脸数据
    std::vector<Rect> faceRects;
    cascade.detectMultiScale(grayImage,faceRects);//检测人脸
    if(faceRects.size()>0)
    {
        Rect rect = faceRects.at(0);//第一个人脸的矩形框
        //rectangle(srcImage,rect,Scalar(0,0,255));
        //移动人脸框（图片--QLabel）
        ui->headpicLb->move(rect.x,rect.y);

        //把Mat数据转化为QbyteArray，  --》编码成jpg格式
        std::vector<uchar> buf;
        cv::imencode(".jpg",srcImage,buf);
        QByteArray byte((const char*)buf.data(),buf.size());
        //准备发送
        quint64 backsize = byte.size();
        QByteArray sendData;
        QDataStream stream(&sendData,QIODevice::WriteOnly);
        stream.setVersion(QDataStream::Qt_5_14);
        stream<<backsize<<byte;
        //发送
        msocket.write(sendData);

    }else
    {
        //把人脸框移动到中心位置
        ui->headpicLb->move(100,60);
    }
```

```
    if(srcImage.data == nullptr) return;
    //把opencv里面的Mat格式数据（BGR）转Qt里面的QImage(RGB)
    cvtColor(srcImage,srcImage, COLOR_BGR2RGB);
    QImage image(srcImage.data,srcImage.cols,
srcImage.rows,srcImage.step1(),QImage::Format_RGB888);
    QPixmap mmp = QPixmap::fromImage(image);

    ui->videoLb->setPixmap(mmp);
}

void FaceAttendence::timer_connect()
{
    //连接服务器
    msocket.connectToHost("127.0.0.1",9999);
    qDebug()<<"正在连接服务器";
}

void FaceAttendence::stop_connect()
{
    mtimer.stop();
    qDebug()<<"成功连接服务器";
}

void FaceAttendence::start_connect()
{
    mtimer.start(5000);//启动定时器
    qDebug()<<"断开连接";
}
```

服务器:

```
#ifndef ATTENDANCEWIN_H
#define ATTENDANCEWIN_H

#include <QMainWindow>
#include <QTcpSocket>
#include <QTcpServer>

QT_BEGIN_NAMESPACE
namespace Ui { class AttendanceWin; }
QT_END_NAMESPACE

class AttendanceWin : public QMainWindow
{
    Q_OBJECT

public:
    AttendanceWin(QWidget *parent = nullptr);
    ~AttendanceWin();

protected slots:
    void accept_client();
    void read_data();
```

```
private:
    Ui::AttendanceWin *ui;
    QTcpServer mserver;
    QTcpSocket *msocket;
    quint64 bsize;
};
#endif // ATTENDANCEWIN_H
```

```
#include "attendancewin.h"
#include "ui_attendancewin.h"

AttendanceWin::AttendanceWin(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::AttendanceWin)
{
    ui->setupUi(this);
    //qtcpServer当有客户端连会发送newconnection
    connect(&mserver, &QTcpServer::newConnection, this,
&AttendanceWin::accept_client);
    mserver.listen(QHostAddress::Any,9999);//监听，启动服务器
    bsize = 0;
}

AttendanceWin::~AttendanceWin()
{
    delete ui;
}

//接受客户端连接
void AttendanceWin::accept_client()
{
    //获取与客户端通信的套接字
    msocket = mserver.nextPendingConnection();
    //当客户端有数据到达会发送readyRead信号
  connect(msocket,&QTcpSocket::readyRead,this,&AttendanceWin::read_data);
}

//读取客户端发送的数据
void AttendanceWin::read_data()
{
    QDataStream stream(msocket); //把套接字绑定到数据流
    stream.setVersion(QDataStream::Qt_5_14);

    if(bsize == 0){
        if(msocket->bytesAvailable()<(qint64)sizeof(bsize)) return ;
        //采集数据的长度
        stream>>bsize;
    }

    if(msocket->bytesAvailable() < bsize)//说明数据还没有发送完成，返回继续等待
    {
        return ;
    }

    QByteArray data;
```

```
    stream>>data;
    bsize = 0;
    if(data.size() == 0)//没有读取到数据
    {
        return;
    }

    //显示图片
    QPixmap mmp;
    mmp.loadFromData(data,"jpg");
    mmp = mmp.scaled(ui->picLb->size());
    ui->picLb->setPixmap(mmp);
}
```

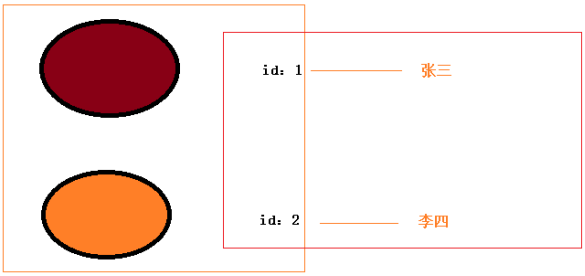
九、服务器人脸模块人脸数据注册和查询

```
#ifndef QFACEOBJECT_H
#define QFACEOBJECT_H

#include <QObject>
#include <seeta/FaceEngine.h>
#include <opencv.hpp>

//人脸数据存储，  人脸检测，人脸识别
class QFaceObject : public QObject
{
    Q_OBJECT
public:
    explicit QFaceObject(QObject *parent = nullptr);
    ~QFaceObject();
public slots:
    int64_t face_register(cv::Mat& faceImage);
    int  face_query(cv::Mat& faceImage);
signals:

private:
    seeta::FaceEngine  *fengineptr;
};

#endif // QFACEOBJECT_H
```

```
#include "qfaceobject.h"

QFaceObject::QFaceObject(QObject *parent) : QObject(parent)
{
    //初始化
    seeta::ModelSetting
 FDmode("C:/SeetaFace/bin/model/fd_2_00.dat",seeta::ModelSetting::CPU,0);
    seeta::ModelSetting
 PDmode("C:/SeetaFace/bin/model/pd_2_00_pts5.dat",seeta::ModelSetting::CPU,0);
    seeta::ModelSetting
 FRmode("C:/SeetaFace/bin/model/fr_2_10.dat",seeta::ModelSetting::CPU,0);
    this->fengineptr = new seeta::FaceEngine(FDmode,PDmode,FRmode);
}
```

```
QFaceObject::~QFaceObject()
{
    delete fengineptr;
}

int64_t QFaceObject::face_register(cv::Mat &faceImage)
{

    //把opencv的Mat数据转为seetaface的数据
    SeetaImageData simage;
    simage.data = faceImage.data;
    simage.width = faceImage.cols;
    simage.height = faceImage.rows;
    simage.channels = faceImage.channels();
    int64_t faceid = this->fengineptr->Register(simage);//注册返回一个人脸id
    if(faceid>=0){
        fengineptr->Save("./face.db");
    }
    return faceid;
}

int QFaceObject::face_query(cv::Mat &faceImage)
{
    //把opencv的Mat数据转为seetaface的数据
    SeetaImageData simage;
    simage.data = faceImage.data;
    simage.width = faceImage.cols;
    simage.height = faceImage.rows;
    simage.channels = faceImage.channels();
    float similarity=0;
    int64_t faceid = fengineptr->Query(simage,&similarity);//运行时间比较长
    return faceid;
}
```

十、数据库创建



人脸数据库

人脸id与对应的人信息数据库

员工信息表

编号	姓名	性别	生日	地址	电话	人脸ID	头像

考勤表

编号	员工编号	考勤时间	备注

qt自带的sqlite3

员工信息表 employee

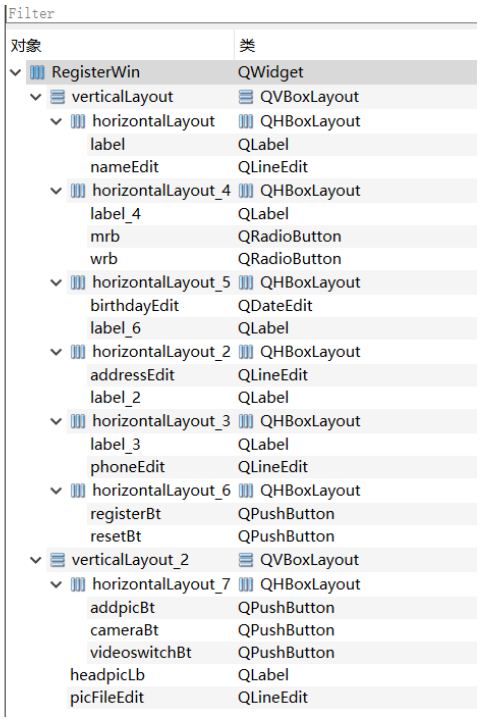
编号employeeID	姓名name	性别sex	生日birthday	地址address	电话phone	人脸IdfaceID	头像headfile

考勤表 attendance

编号attendanceID	员工编号employeeID	考勤时间attTime	备注notes

```
#include "attendancewin.h"

#include <QApplication>
#include <QSqlDatabase>
#include <QSqlError>
#include <QSqlQuery>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    //连接数据库
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    //设置数据名称
    db.setDatabaseName("server.db");
    //打开数据库
    if(!db.open())
    {
        qDebug()<<db.lastError().text();
        return -1;
    }
    //创建员工信息表格
    QString createsql = "create table if not exists employee(employeeID integer
primary key autoincrement,name varchar(256), sex varchar(32),"
                        "birthday text, address text, phone text, faceID integer
unique, headfile text)";
    QSqlQuery query;
    if(!query.exec(createsql))
    {
       qDebug()<<query.lastError().text();
       return -1;
    }

    //考勤表格
    createsql = "create table if not exists attendance(attendaceID integer
primary key autoincrement, employeeID integer,"
                "attendaceTime TimeStamp NOT NULL
DEFAULT(datetime('now','localtime')))";
    if(!query.exec(createsql))
    {
       qDebug()<<query.lastError().text();
       return -1;
    }

    AttendanceWin w;
    w.show();
    return a.exec();
}
```

十一、注册界面设计



十二、个人信息录入到数据库，摄像头显示与拍照

```
#ifndef REGISTERWIN_H
#define REGISTERWIN_H
#include <QWidget>
#include <opencv.hpp>

namespace Ui {
class RegisterWin;
}

class RegisterWin : public QWidget
{
    Q_OBJECT
public:
    explicit RegisterWin(QWidget *parent = nullptr);
    ~RegisterWin();
    void timerEvent(QTimerEvent *e);

private slots:
    void on_resetBt_clicked();
    void on_addpicBt_clicked();
    void on_registerBt_clicked();
    void on_videoswitchBt_clicked();
    void on_cameraBt_clicked();

private:
    Ui::RegisterWin *ui;
    int timerid;
    cv::VideoCapture cap;
    cv::Mat image;
};
#endif // REGISTERWIN_H
```

```
#include "registerwin.h"
#include "ui_registerwin.h"
#include <QFileDialog>
#include <qfaceobject.h>
#include <QSqlTableModel>
#include <QSqlRecord>
#include <QMessageBox>
#include <QDebug>

RegisterWin::RegisterWin(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::RegisterWin)
{
    ui->setupUi(this);
}

RegisterWin::~RegisterWin()
{
    delete ui;
}

void RegisterWin::timerEvent(QTimerEvent *e)
{
    //获取摄像头数据并且显示在界面上
    if(cap.isOpened())
    {
        cap>>image;
        if(image.data == nullptr) return;
    }
    //Mat---》QImage
    cv::Mat rgbImage;
    cv::cvtColor(image,rgbImage,cv::COLOR_BGR2RGB);
    QImage qImg(rgbImage.data, rgbImage.cols, rgbImage.rows,rgbImage.step1(),
QImage::Format_RGB888);
    //在qt界面上显示
    QPixmap mmp=QPixmap::fromImage(qImg);
    mmp = mmp.scaledToWidth(ui->headpicLb->width());
    ui->headpicLb->setPixmap(mmp);
}

void RegisterWin::on_resetBt_clicked()
{
    //清空数据
    ui->nameEdit->clear();
    ui->birthdayEdit->setDate(QDate::currentDate());
    ui->addressEdit->clear();
    ui->phoneEdit->clear();
    ui->picFileEdit->clear();
}

void RegisterWin::on_addpicBt_clicked()
{
    //通过文件对话框 选中图片路径
    QString filepath = QFileDialog::getOpenFileName(this);
```

```
    ui->picFileEdit->setText(filepath);

    //显示图片
    QPixmap mmp(filepath);
    mmp = mmp.scaledToWidth(ui->headpicLb->width());
    ui->headpicLb->setPixmap(mmp);
}

void RegisterWin::on_registerBt_clicked()
{
    //1.通过照片，结合faceObject模块得到faceID
    QFaceObject  faceobj;
    cv::Mat image = cv::imread(ui->picFileEdit->text().toUtf8().data());
    int faceID = faceobj.face_register(image);
    qDebug()<<faceID;
    //把头像保存到一个固定路径下
    QString headfile = QString("./%1.jpg").arg(QString(ui->nameEdit-
>text().toUtf8()));
    cv::imwrite(headfile.toUtf8().data(), image);

    //2.把个人信息存储到数据库employee
    QSqlTableModel model;
    model.setTable("employee");//设置表名
    QSqlRecord record = model.record();
    //设置数据
    record.setValue("name",ui->nameEdit->text());
    record.setValue("sex",ui->mrb->isChecked()?"男":"女");
    record.setValue("birthday", ui->birthdayEdit->text());
    record.setValue("address",ui->addressEdit->text());
    record.setValue("phone",ui->phoneEdit->text());
    record.setValue("faceID", faceID);
    //头像路径
    record.setValue("headfile",headfile);
    //把记录插入到数据库表格中
    bool ret = model.insertRecord(0,record);
    //3.提示注册成功
    if(ret)
    {
        QMessageBox::information(this,"注册提示","注册成功");
        //提交
        model.submitAll();
    }else
    {
        QMessageBox::information(this,"注册提示","注册失败");
    }
}

void RegisterWin::on_videoswitchBt_clicked()
{
    if(ui->videoswitchBt->text() == "打开摄像头")
    {
        //打开摄像头
        if(cap.open(0))
        {
            ui->videoswitchBt->setText("关闭摄像头");
```

```
            //启动定时器事件
            timerid = startTimer(100);
        }
    }
    else
    {
        killTimer(timerid);//关闭定时器事件
        ui->videoswitchBt->setText("打开摄像头");
        //关闭摄像头
        cap.release();
    }
}

void RegisterWin::on_cameraBt_clicked()
{
    //保存数据
    //把头像保存到一个固定路径下
    QString headfile = QString("./%1.jpg").arg(QString(ui->nameEdit-
>text().toUtf8()));
    ui->picFileEdit->setText(headfile);
    cv::imwrite(headfile.toUtf8().data(), image);
    killTimer(timerid);//关闭定时器事件
    ui->videoswitchBt->setText("打开摄像头");
    //关闭摄像头
    cap.release();
}
```

十三、接收客户端人脸图像识别人脸ID，利用ID从数据库中查询个人信息

```
#include "attendancewin.h"
#include "ui_attendancewin.h"
#include <QDateTime>
#include <opencv.hpp>

AttendanceWin::AttendanceWin(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::AttendanceWin)
{
    ui->setupUi(this);
    //qtcpServer当有客户端连会发送newconnection
    connect(&mserver, &QTcpServer::newConnection, this,
&AttendanceWin::accept_client);
    mserver.listen(QHostAddress::Any,9999);//监听，启动服务器
    bsize = 0;

    //给sql模型绑定表格
    model.setTable("employee");
}

AttendanceWin::~AttendanceWin()
{
    delete ui;
}
```

```qt
//接受客户端连接
void AttendanceWin::accept_client()
{
    //获取与客户端通信的套接字
    msocket = mserver.nextPendingConnection();

    //当客户端有数据到达会发送readyRead信号
    connect(msocket,&QTcpSocket::readyRead,this,&AttendanceWin::read_data);
}

//读取客户端发送的数据
void AttendanceWin::read_data()
{
    QDataStream stream(msocket); //把套接字绑定到数据流
    stream.setVersion(QDataStream::Qt_5_14);

    if(bsize == 0){
        if(msocket->bytesAvailable()<(qint64)sizeof(bsize)) return ;
        //采集数据的长度
        stream>>bsize;
    }

    if(msocket->bytesAvailable() < bsize)//说明数据还没有发送完成，返回继续等待
    {
        return ;
    }
    QByteArray data;
    stream>>data;
    bsize = 0;
    if(data.size() == 0)//没有读取到数据
    {
        return;
    }

    //显示图片
    QPixmap mmp;
    mmp.loadFromData(data,"jpg");
    mmp = mmp.scaled(ui->picLb->size());
    ui->picLb->setPixmap(mmp);

    //识别人脸
    cv::Mat faceImage;
    std::vector<uchar> decode;
    decode.resize(data.size());
    memcpy(decode.data(),data.data(),data.size());
    faceImage = cv::imdecode(decode, cv::IMREAD_COLOR);
    int faceid = fobj.face_query(faceImage);
    //qDebug()<<"00000"<<faceid;

    //从数据库中查询faceid对应的个人信息
    //给模型设置过滤器
    model.setFilter(QString("faceID=%1").arg(faceid));
    //查询
    model.select();
```

```
    //判断是否查询到数据
    if(model.rowCount() == 1)
    {
        //工号，姓名，部门，时间
        //{employeeID:%1,name:%2,department:软件,time:%3}
        QSqlRecord record = model.record(0);
        QString sdmsg = QString("
{\"employeeID\":\"%1\",\"name\":\"%2\",\"department\":\"软件\",\"time\":\"%3\"}")

.arg(record.value("employeeID").toString()).arg(record.value("name").toString())
                .arg(QDateTime::currentDateTime().toString("yyyy-MM-dd
hh:mm:ss"));

        msocket->write(sdmsg.toUtf8());//把打包好的数据发送给客户端

        //把数据写入数据库--考勤表
    }
}
```

客户端

```
#include "faceattendence.h"
#include "ui_faceattendence.h"
#include <QImage>
#include <QPainter>
#include <QDebug>
FaceAttendence::FaceAttendence(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::FaceAttendence)
{
    ui->setupUi(this);
    //打开摄像头
    cap.open(0);//dev/video
    //启动定时器事件
    startTimer(100);

    //导入级联分类器文件

 cascade.load("C:/opencv452/etc/haarcascades/haarcascade_frontalface_alt2.xml");

    //QTcpSocket当断开连接的时候disconnected信号，连接成功会发送connected
    connect(&msocket,&QTcpSocket::disconnected,this,
&FaceAttendence::start_connect);
    connect(&msocket,&QTcpSocket::connected,this,
&FaceAttendence::stop_connect);
    //关联接收数据的槽函数
    connect(&msocket, &QTcpSocket::readyRead,this, &FaceAttendence::recv_data);

    //定时器连接服务器
    connect(&mtimer, &QTimer::timeout,this,&FaceAttendence::timer_connect);
    //启动定时器
    mtimer.start(5000);//每5s钟连接一次，直到连接成功就不在连接
}

FaceAttendence::~FaceAttendence()
```

```
{
    delete ui;
}

void FaceAttendence::timerEvent(QTimerEvent *e)
{
    //采集数据
    Mat srcImage;
    if(cap.grab())
    {
        cap.read(srcImage);//读取一帧数据
    }

    Mat grayImage;
    //转灰度图
    cvtColor(srcImage, grayImage, COLOR_BGR2GRAY);
    //检测人脸数据
    std::vector<Rect> faceRects;
    cascade.detectMultiScale(grayImage,faceRects);//检测人脸
    if(faceRects.size()>0)
    {
        Rect rect = faceRects.at(0);//第一个人脸的矩形框
        //rectangle(srcImage,rect,Scalar(0,0,255));
        //移动人脸框（图片--QLabel）
        ui->headpicLb->move(rect.x,rect.y);

        //把Mat数据转化为QbyteArray，  -->编码成jpg格式
        std::vector<uchar> buf;
        cv::imencode(".jpg",srcImage,buf);
        QByteArray byte((const char*)buf.data(),buf.size());
        //准备发送
        quint64 backsize = byte.size();
        QByteArray sendData;
        QDataStream stream(&sendData,QIODevice::WriteOnly);
        stream.setVersion(QDataStream::Qt_5_14);
        stream<<backsize<<byte;
        //发送
        msocket.write(sendData);

    }else
    {
        //把人脸框移动到中心位置
        ui->headpicLb->move(100,60);
    }

    if(srcImage.data == nullptr) return;
    //把opencv里面的Mat格式数据（BGR）转Qt里面的QImage(RGB)
    cvtColor(srcImage,srcImage, COLOR_BGR2RGB);
    QImage image(srcImage.data,srcImage.cols,
srcImage.rows,srcImage.step1(),QImage::Format_RGB888);
    QPixmap mmp = QPixmap::fromImage(image);

    ui->videoLb->setPixmap(mmp);
}
```

```
void FaceAttendence::recv_data()
{
    QString msg = msocket.readAll();
    qDebug()<<msg;
    ui->lineEdit->setText(msg);
}

void FaceAttendence::timer_connect()
{
    //连接服务器
    msocket.connectToHost("127.0.0.1",9999);
    qDebug()<<"正在连接服务器";
}

void FaceAttendence::stop_connect()
{
    mtimer.stop();
    qDebug()<<"成功连接服务器";
}

void FaceAttendence::start_connect()
{
    mtimer.start(5000);//启动定时器
    qDebug()<<"断开连接";
}
```

十四、人脸采集发送次数优化

```
#include "faceattendence.h"
#include "ui_faceattendence.h"
#include <QImage>
#include <QPainter>
#include <QDebug>
FaceAttendence::FaceAttendence(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::FaceAttendence)
{
    ui->setupUi(this);
    //打开摄像头
    cap.open(0);//dev/video
    //启动定时器事件
    startTimer(100);

    //导入级联分类器文件

 cascade.load("C:/opencv452/etc/haarcascades/haarcascade_frontalface_alt2.xml");

    //QTcpSocket当断开连接的时候disconnected信号，连接成功会发送connected
    connect(&msocket,&QTcpSocket::disconnected,this,
&FaceAttendence::start_connect);
    connect(&msocket,&QTcpSocket::connected,this,
&FaceAttendence::stop_connect);
    //关联接收数据的槽函数
```

```
    connect(&msocket, &QTcpSocket::readyRead,this, &FaceAttendence::recv_data);

    //定时器连接服务器
    connect(&mtimer, &QTimer::timeout,this,&FaceAttendence::timer_connect);
    //启动定时器
    mtimer.start(5000);//每5s钟连接一次，直到连接成功就不在连接

    flag =0;
}

FaceAttendence::~FaceAttendence()
{
    delete ui;
}

void FaceAttendence::timerEvent(QTimerEvent *e)
{
    //采集数据
    Mat srcImage;
    if(cap.grab())
    {
        cap.read(srcImage);//读取一帧数据
    }

    Mat grayImage;
    //转灰度图
    cvtColor(srcImage, grayImage, COLOR_BGR2GRAY);
    //检测人脸数据
    std::vector<Rect> faceRects;
    cascade.detectMultiScale(grayImage,faceRects);//检测人脸
    if(faceRects.size()>0 && flag>=0)
    {

        Rect rect = faceRects.at(0);//第一个人脸的矩形框
        //rectangle(srcImage,rect,Scalar(0,0,255));
        //移动人脸框（图片--QLabel）
        ui->headpicLb->move(rect.x,rect.y);

        if(flag > 2){
            //把Mat数据转化为QbyteArray，  --》编码成jpg格式
            std::vector<uchar> buf;
            cv::imencode(".jpg",srcImage,buf);
            QByteArray byte((const char*)buf.data(),buf.size());
            //准备发送
            quint64 backsize = byte.size();
            QByteArray sendData;
            QDataStream stream(&sendData,QIODevice::WriteOnly);
            stream.setVersion(QDataStream::Qt_5_14);
            stream<<backsize<<byte;
            //发送
            msocket.write(sendData);
            flag = -2;
        }
        flag++;
    }
```

```
    if(faceRects.size() == 0)
    {
        //把人脸框移动到中心位置
        ui->headpicLb->move(100,60);
        flag=0;
    }

    if(srcImage.data == nullptr) return;
    //把opencv里面的Mat格式数据（BGR）转Qt里面的QImage(RGB)
    cvtColor(srcImage,srcImage, COLOR_BGR2RGB);
    QImage image(srcImage.data,srcImage.cols,
srcImage.rows,srcImage.step1(),QImage::Format_RGB888);
    QPixmap mmp = QPixmap::fromImage(image);

    ui->videoLb->setPixmap(mmp);
}

void FaceAttendence::recv_data()
{
    QString msg = msocket.readAll();
    qDebug()<<msg;
    ui->lineEdit->setText(msg);
}

void FaceAttendence::timer_connect()
{
    //连接服务器
    msocket.connectToHost("127.0.0.1",9999);
    qDebug()<<"正在连接服务器";
}

void FaceAttendence::stop_connect()
{
    mtimer.stop();
    qDebug()<<"成功连接服务器";
}

void FaceAttendence::start_connect()
{
    mtimer.start(5000);//启动定时器
    qDebug()<<"断开连接";
}
```

十五、人脸识别采样线程完成

```
#ifndef ATTENDANCEWIN_H
#define ATTENDANCEWIN_H

#include "qfaceobject.h"

#include <QMainWindow>
#include <QTcpSocket>
#include <QTcpServer>
#include <QSqlTableModel>
#include <QSqlRecord>
```

```
QT_BEGIN_NAMESPACE
namespace Ui { class AttendanceWin; }
QT_END_NAMESPACE

class AttendanceWin : public QMainWindow
{
    Q_OBJECT

public:
    AttendanceWin(QWidget *parent = nullptr);
    ~AttendanceWin();
signals:
    void query(cv::Mat& image);
protected slots:
    void accept_client();
    void read_data();
    void recv_faceid(int64_t faceid);
private:
    Ui::AttendanceWin *ui;
    QTcpServer mserver;
    QTcpSocket *msocket;
    quint64 bsize;

    QFaceObject fobj;
    QSqlTableModel model;

};
#endif // ATTENDANCEWIN_H
```

```
#include "attendancewin.h"
#include "ui_attendancewin.h"
#include <QDateTime>
#include <QThread>
#include <opencv.hpp>

AttendanceWin::AttendanceWin(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::AttendanceWin)
{
    ui->setupUi(this);
    //qtcpServer当有客户端连会发送newconnection
    connect(&mserver, &QTcpServer::newConnection, this,
&AttendanceWin::accept_client);
    mserver.listen(QHostAddress::Any,9999);//监听，启动服务器
    bsize = 0;

    //给sql模型绑定表格
    model.setTable("employee");


    //创建一个线程
    QThread *thread = new QThread();
    //把QFaceObject对象移动到thread线程中执行
    fobj.moveToThread(thread);
    //启动线程
```

```
    thread->start();
    connect(this,&AttendanceWin::query,&fobj,&QFaceObject::face_query);
    //关联QFaceObject对象里面的send_faceid信号
    connect(&fobj,&QFaceObject::send_faceid,this, &AttendanceWin::recv_faceid);

}

AttendanceWin::~AttendanceWin()
{
    delete ui;
}

//接受客户端连接
void AttendanceWin::accept_client()
{
    //获取与客户端通信的套接字
    msocket = mserver.nextPendingConnection();

    //当客户端有数据到达会发送readyRead信号
    connect(msocket,&QTcpSocket::readyRead,this,&AttendanceWin::read_data);
}

//读取客户端发送的数据
void AttendanceWin::read_data()
{
    QDataStream stream(msocket); //把套接字绑定到数据流
    stream.setVersion(QDataStream::Qt_5_14);

    if(bsize == 0){
        if(msocket->bytesAvailable()<(qint64)sizeof(bsize)) return ;
        //采集数据的长度
        stream>>bsize;
    }

    if(msocket->bytesAvailable() < bsize)//说明数据还没有发送完成，返回继续等待
    {
        return ;
    }
    QByteArray data;
    stream>>data;
    bsize = 0;
    if(data.size() == 0)//没有读取到数据
    {
        return;
    }

    //显示图片
    QPixmap mmp;
    mmp.loadFromData(data,"jpg");
    mmp = mmp.scaled(ui->picLb->size());
    ui->picLb->setPixmap(mmp);


    //识别人脸
    cv::Mat faceImage;
```

```
    std::vector<uchar> decode;
    decode.resize(data.size());
    memcpy(decode.data(),data.data(),data.size());
    faceImage = cv::imdecode(decode, cv::IMREAD_COLOR);

    //int faceid = fobj.face_query(faceImage); //消耗资源较多
    emit query(faceImage);

}

void AttendanceWin::recv_faceid(int64_t faceid)
{
    //qDebug()<<"00000"<<faceid;
    //从数据库中查询faceid对应的个人信息
    //给模型设置过滤器
    qDebug()<<"识别到的人脸id:"<<faceid;
    if(faceid < 0)
    {
        QString sdmsg = QString("
{\"employeeID\":,\"name\":,\"department\":,\"time\":}");
        msocket->write(sdmsg.toUtf8());//把打包好的数据发送给客户端
        return ;
    }
    model.setFilter(QString("faceID=%1").arg(faceid));
    //查询
    model.select();
    //判断是否查询到数据
    if(model.rowCount() == 1)
    {
        //工号，姓名，部门，时间
        //{employeeID:%1,name:%2,department:软件,time:%3}
        QSqlRecord record = model.record(0);
        QString sdmsg = QString("
{\"employeeID\":\"%1\",\"name\":\"%2\",\"department\":\"软件\",\"time\":\"%3\"}")

.arg(record.value("employeeID").toString()).arg(record.value("name").toString())
                .arg(QDateTime::currentDateTime().toString("yyyy-MM-dd
hh:mm:ss"));

        msocket->write(sdmsg.toUtf8());//把打包好的数据发送给客户端
        //把数据写入数据库--考勤表
    }
}
```

```
#include "qfaceobject.h"
#include <QDebug>
QFaceObject::QFaceObject(QObject *parent) : QObject(parent)
{
    //初始化
    seeta::ModelSetting
 FDmode("C:/SeetaFace/bin/model/fd_2_00.dat",seeta::ModelSetting::CPU,0);
    seeta::ModelSetting
 PDmode("C:/SeetaFace/bin/model/pd_2_00_pts5.dat",seeta::ModelSetting::CPU,0);
    seeta::ModelSetting
 FRmode("C:/SeetaFace/bin/model/fr_2_10.dat",seeta::ModelSetting::CPU,0);
```

```
    this->fengineptr = new seeta::FaceEngine(FDmode,PDmode,FRmode);

    //导入已有的人脸数据库
    this->fengineptr->Load("./face.db");
}

QFaceObject::~QFaceObject()
{
    delete fengineptr;
}

int64_t QFaceObject::face_register(cv::Mat &faceImage)
{

    //把opencv的Mat数据转为seetaface的数据
    SeetaImageData simage;
    simage.data = faceImage.data;
    simage.width = faceImage.cols;
    simage.height = faceImage.rows;
    simage.channels = faceImage.channels();
    int64_t faceid = this->fengineptr->Register(simage);//注册返回一个人脸id
    if(faceid>=0){
        fengineptr->Save("./face.db");
    }
    return faceid;
}

int QFaceObject::face_query(cv::Mat &faceImage)
{
    //把opencv的Mat数据转为seetaface的数据
    SeetaImageData simage;
    simage.data = faceImage.data;
    simage.width = faceImage.cols;
    simage.height = faceImage.rows;
    simage.channels = faceImage.channels();
    float similarity=0;
    int64_t faceid = fengineptr->Query(simage,&similarity);//运行时间比较长
    qDebug()<<"查询"<<faceid<<similarity;
    if(similarity > 0.7)
    {
        emit send_faceid(faceid);
    }else
    {
        emit send_faceid(-1);
    }
    return faceid;
}
```

十六、考勤机终端json数据json和显示

```
#ifndef FACEATTENDENCE_H
#define FACEATTENDENCE_H

#include <QMainWindow>
#include <opencv.hpp>
#include <QTcpSocket>
#include <QTimer>
using namespace cv;
using namespace std;

QT_BEGIN_NAMESPACE
namespace Ui { class FaceAttendence; }
QT_END_NAMESPACE

class FaceAttendence : public QMainWindow
{
    Q_OBJECT
public:
    FaceAttendence(QWidget *parent = nullptr);
    ~FaceAttendence();
    //定时器事件
    void timerEvent(QTimerEvent *e);

protected slots:
    void recv_data();
private slots:
    void timer_connect();
    void stop_connect();
    void start_connect();

private:
    Ui::FaceAttendence *ui;

    //摄像头
    VideoCapture cap;
    //haar--级联分类器
    cv::CascadeClassifier cascade;

    //创建网络套接字，定时器
    QTcpSocket msocket;
    QTimer mtimer;

    //标志是否是同一个人脸进入到识别区域
    int flag;

    //保存人类的数据
    cv::Mat faceMat;

};
#endif // FACEATTENDENCE_H
```

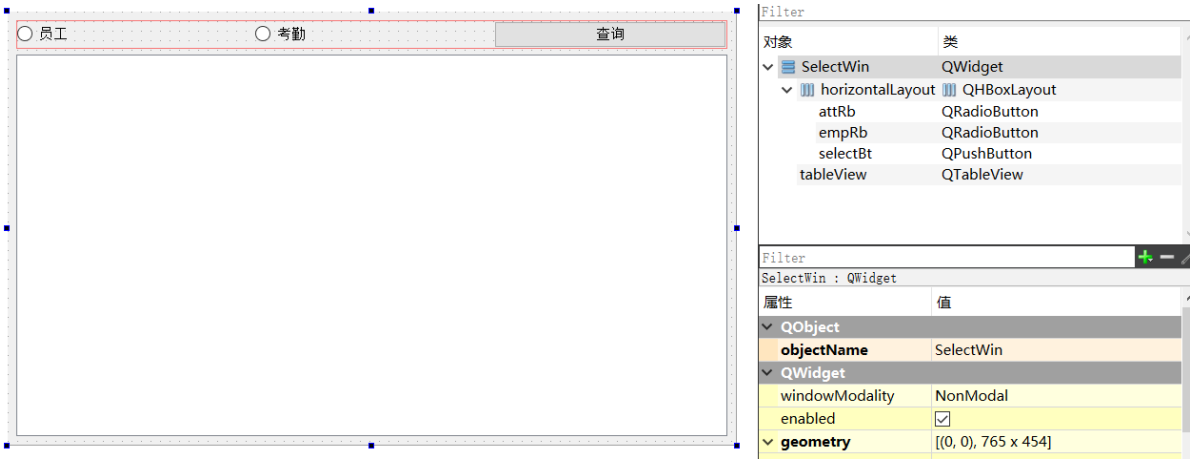
```
#include "faceattendence.h"
```

```
#include "ui_faceattendence.h"
#include <QImage>
#include <QPainter>
#include <QDebug>
#include <QJsonDocument>
#include <QJsonParseError>
#include <QJsonObject>
FaceAttendence::FaceAttendence(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::FaceAttendence)
{
    ui->setupUi(this);
    //打开摄像头
    cap.open(0);//dev/video
    //启动定时器事件
    startTimer(100);

    //导入级联分类器文件

 cascade.load("C:/opencv452/etc/haarcascades/haarcascade_frontalface_alt2.xml");

    //QTcpSocket当断开连接的时候disconnected信号，连接成功会发送connected
    connect(&msocket,&QTcpSocket::disconnected,this,
&FaceAttendence::start_connect);
    connect(&msocket,&QTcpSocket::connected,this,
&FaceAttendence::stop_connect);
    //关联接收数据的槽函数
    connect(&msocket, &QTcpSocket::readyRead,this, &FaceAttendence::recv_data);

    //定时器连接服务器
    connect(&mtimer, &QTimer::timeout,this,&FaceAttendence::timer_connect);
    //启动定时器
    mtimer.start(5000);//每5s钟连接一次，直到连接成功就不在连接

    flag =0;

    ui->widgetLb->hide();
}

FaceAttendence::~FaceAttendence()
{
    delete ui;
}

void FaceAttendence::timerEvent(QTimerEvent *e)
{
    //采集数据
    Mat srcImage;
    if(cap.grab())
    {
        cap.read(srcImage);//读取一帧数据
    }

    //把图片大小设与显示窗口一样大
    cv::resize(srcImage,srcImage,Size(480,480));
```

```
    Mat grayImage;
    //转灰度图
    cvtColor(srcImage, grayImage, COLOR_BGR2GRAY);
    //检测人脸数据
    std::vector<Rect> faceRects;
    cascade.detectMultiScale(grayImage,faceRects);//检测人脸
    if(faceRects.size()>0 && flag>=0)
    {

        Rect rect = faceRects.at(0);//第一个人脸的矩形框
        //rectangle(srcImage,rect,Scalar(0,0,255));
        //移动人脸框（图片--QLabel）
        ui->headpicLb->move(rect.x,rect.y);

        if(flag > 2){
            //把Mat数据转化为QbyteArray，  --》编码成jpg格式
            std::vector<uchar> buf;
            cv::imencode(".jpg",srcImage,buf);
            QByteArray byte((const char*)buf.data(),buf.size());
            //准备发送
            quint64 backsize = byte.size();
            QByteArray sendData;
            QDataStream stream(&sendData,QIODevice::WriteOnly);
            stream.setVersion(QDataStream::Qt_5_14);
            stream<<backsize<<byte;
            //发送
            msocket.write(sendData);
            flag = -2;

            faceMat = srcImage(rect);
            //保存
            imwrite("./face.jpg",faceMat);
        }
        flag++;
    }
    if(faceRects.size() == 0)
    {
        //把人脸框移动到中心位置
        ui->headpicLb->move(100,60);
        ui->widgetLb->hide();
        flag=0;
    }

    if(srcImage.data == nullptr) return;
    //把opencv里面的Mat格式数据（BGR）转Qt里面的QImage(RGB)
    cvtColor(srcImage,srcImage, COLOR_BGR2RGB);
    QImage image(srcImage.data,srcImage.cols,
srcImage.rows,srcImage.step1(),QImage::Format_RGB888);
    QPixmap mmp = QPixmap::fromImage(image);
    ui->videoLb->setPixmap(mmp);
}

void FaceAttendence::recv_data()
{
```

```
    //{employeeID:%1,name:%2,department:软件,time:%3}
    QByteArray array = msocket.readAll();
    qDebug()<<array;
    //Json解析
    QJsonParseError err;
    QJsonDocument doc = QJsonDocument::fromJson(array,&err);
    if(err.error != QJsonParseError::NoError)
    {
        qDebug()<<"json数据错误";
        return;
    }

    QJsonObject obj = doc.object();
    QString employeeID = obj.value("employeeID").toString();
    QString name = obj.value("name").toString();
    QString department = obj.value("department").toString();
    QString timestr = obj.value("time").toString();

    ui->numberEdit->setText(employeeID);
    ui->nameEdit->setText(name);
    ui->departmentEdit->setText(department);
    ui->timeEdit->setText(timestr);

    //通过样式来显示图片
    ui->headLb->setStyleSheet("border-radius:75px;border-image:
url(./face.jpg);");
    ui->widgetLb->show();
}

void FaceAttendence::timer_connect()
{
    //连接服务器
    msocket.connectToHost("127.0.0.1",9999);
    qDebug()<<"正在连接服务器";
}

void FaceAttendence::stop_connect()
{
    mtimer.stop();
    qDebug()<<"成功连接服务器";
}

void FaceAttendence::start_connect()
{
    mtimer.start(5000);//启动定时器
    qDebug()<<"断开连接";
}
```

十七、服务器查询模块设计与实现

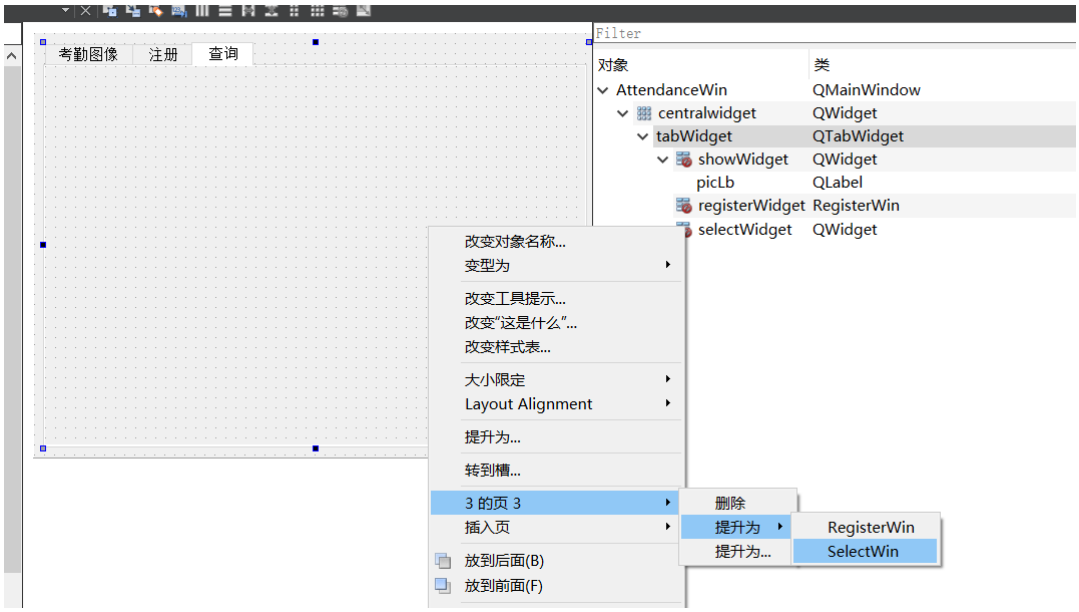


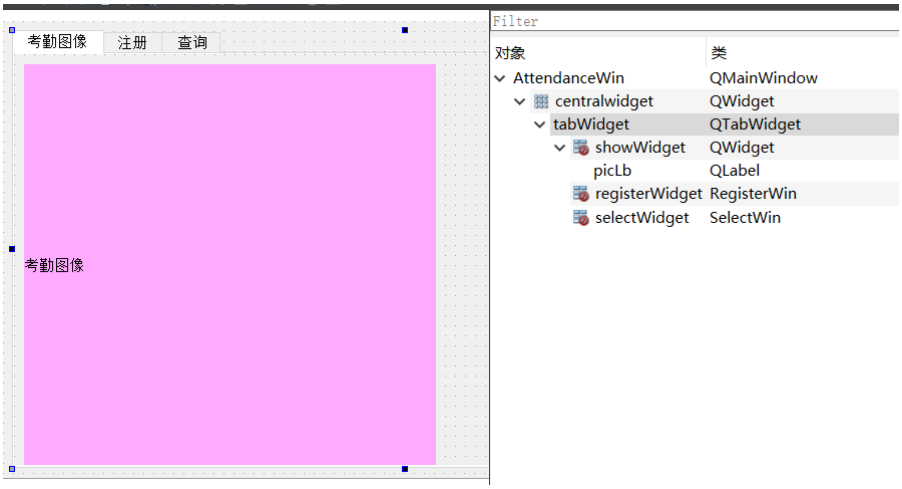

```
#ifndef SELECTWIN_H
#define SELECTWIN_H

#include <QWidget>
#include <QSqlTableModel>
namespace Ui {
class SelectWin;
}

class SelectWin : public QWidget
{
    Q_OBJECT

public:
    explicit SelectWin(QWidget *parent = nullptr);
    ~SelectWin();

private slots:
    void on_selectBt_clicked();

private:
    Ui::SelectWin *ui;
    QSqlTableModel *model;

};

#endif // SELECTWIN_H
```

```
#include "selectwin.h"
#include "ui_selectwin.h"

SelectWin::SelectWin(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::SelectWin)
{
    ui->setupUi(this);
    model = new QSqlTableModel();
}
```

```
SelectWin::~SelectWin()
{
    delete ui;
}

void SelectWin::on_selectBt_clicked()
{
    if(ui->empRb->isChecked())
    {
        model->setTable("employee");//设置员工表格
    }
    if(ui->attRb->isChecked())
    {
        model->setTable("attendance");
    }

    //设置过滤器
    //model->setFilter("name='张三'");
    //查询
    model->select();

    ui->tableView->setModel(model);
}
```

十八、服务器考勤数据记录到数据库、服务器界面整合






```
#include "attendancewin.h"
#include "ui_attendancewin.h"
#include <QDateTime>
#include <QSqlRecord>
#include <QThread>
#include <opencv.hpp>
#include <QSqlQuery>
#include <QSqlError>
AttendanceWin::AttendanceWin(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::AttendanceWin)
{
    ui->setupUi(this);
    //qtcpServer当有客户端连会发送newconnection
    connect(&mserver, &QTcpServer::newConnection, this,
&AttendanceWin::accept_client);
    mserver.listen(QHostAddress::Any,9999);//监听，启动服务器
    bsize = 0;

    //给sql模型绑定表格
    model.setTable("employee");


    //创建一个线程
    QThread *thread = new QThread();
    //把QFaceObject对象移动到thread线程中执行
    fobj.moveToThread(thread);
    //启动线程
    thread->start();
    connect(this,&AttendanceWin::query,&fobj,&QFaceObject::face_query);
    //关联QFaceObject对象里面的send_faceid信号
    connect(&fobj,&QFaceObject::send_faceid,this, &AttendanceWin::recv_faceid);

}

AttendanceWin::~AttendanceWin()
{
    delete ui;
}

//接受客户端连接
```

```
void AttendanceWin::accept_client()
{
    //获取与客户端通信的套接字
    msocket = mserver.nextPendingConnection();

    //当客户端有数据到达会发送readyRead信号
    connect(msocket,&QTcpSocket::readyRead,this,&AttendanceWin::read_data);
}

//读取客户端发送的数据
void AttendanceWin::read_data()
{
    QDataStream stream(msocket); //把套接字绑定到数据流
    stream.setVersion(QDataStream::Qt_5_14);

    if(bsize == 0){
        if(msocket->bytesAvailable()<(qint64)sizeof(bsize)) return ;
        //采集数据的长度
        stream>>bsize;
    }

    if(msocket->bytesAvailable() < bsize)//说明数据还没有发送完成，返回继续等待
    {
        return ;
    }
    QByteArray data;
    stream>>data;
    bsize = 0;
    if(data.size() == 0)//没有读取到数据
    {
        return;
    }

    //显示图片
    QPixmap mmp;
    mmp.loadFromData(data,"jpg");
    mmp = mmp.scaled(ui->picLb->size());
    ui->picLb->setPixmap(mmp);


    //识别人脸
    cv::Mat faceImage;
    std::vector<uchar> decode;
    decode.resize(data.size());
    memcpy(decode.data(),data.data(),data.size());
    faceImage = cv::imdecode(decode, cv::IMREAD_COLOR);

    //int faceid = fobj.face_query(faceImage); //消耗资源较多
    emit query(faceImage);

}

void AttendanceWin::recv_faceid(int64_t faceid)
{
    //qDebug()<<"00000"<<faceid;
```

```
    //从数据库中查询faceid对应的个人信息
    //给模型设置过滤器
    qDebug()<<"识别到的人脸id:"<<faceid;
    if(faceid < 0)
    {
        QString sdmsg = QString("{\"employeeID\":\"
\",\"name\":\"\",\"department\":\"\",\"time\":\"\"}");
        msocket->write(sdmsg.toUtf8());//把打包好的数据发送给客户端
        return ;
    }
    model.setFilter(QString("faceID=%1").arg(faceid));
    //查询
    model.select();
    //判断是否查询到数据
    if(model.rowCount() == 1)
    {
        //工号，姓名，部门，时间
        //{employeeID:%1,name:%2,department:软件,time:%3}
        QSqlRecord record = model.record(0);
        QString sdmsg = QString("
{\"employeeID\":\"%1\",\"name\":\"%2\",\"department\":\"软件\",\"time\":\"%3\"}")

.arg(record.value("employeeID").toString()).arg(record.value("name").toString())
                .arg(QDateTime::currentDateTime().toString("yyyy-MM-dd
hh:mm:ss"));

        //把数据写入数据库--考勤表
        QString insertSql = QString("insert into attendance(employeeID)
values('%1')").arg(record.value("employeeID").toString());
        QSqlQuery query;
        if(!query.exec(insertSql))
        {
            QString sdmsg = QString("{\"employeeID\":\"
\",\"name\":\"\",\"department\":\"\",\"time\":\"\"}");
            msocket->write(sdmsg.toUtf8());//把打包好的数据发送给客户端
            qDebug()<<query.lastError().text();
            return ;
        }else
        {
            msocket->write(sdmsg.toUtf8());//把打包好的数据发送给客户端
        }
    }
}
```

十九、调试

```
#include "faceattendence.h"
#include "ui_faceattendence.h"
#include <QImage>
#include <QPainter>
#include <QDebug>
#include <QJsonDocument>
```

```
#include <QJsonParseError>
#include <QJsonObject>
FaceAttendence::FaceAttendence(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::FaceAttendence)
{
    ui->setupUi(this);
    //打开摄像头
    cap.open(0);//dev/video
    //启动定时器事件
    startTimer(100);

    //导入级联分类器文件

 cascade.load("C:/opencv452/etc/haarcascades/haarcascade_frontalface_alt2.xml");

    //QTcpSocket当断开连接的时候disconnected信号，连接成功会发送connected
    connect(&msocket,&QTcpSocket::disconnected,this,
&FaceAttendence::start_connect);
    connect(&msocket,&QTcpSocket::connected,this,
&FaceAttendence::stop_connect);
    //关联接收数据的槽函数
    connect(&msocket, &QTcpSocket::readyRead,this, &FaceAttendence::recv_data);

    //定时器连接服务器
    connect(&mtimer, &QTimer::timeout,this,&FaceAttendence::timer_connect);
    //启动定时器
    mtimer.start(5000);//每5s钟连接一次，直到连接成功就不在连接

    flag =0;

    ui->widgetLb->hide();
}

FaceAttendence::~FaceAttendence()
{
    delete ui;
}

void FaceAttendence::timerEvent(QTimerEvent *e)
{
    //采集数据
    Mat srcImage;
    if(cap.grab())
    {
        cap.read(srcImage);//读取一帧数据
    }

    //把图片大小设与显示窗口一样大
    cv::resize(srcImage,srcImage,Size(480,480));

    Mat grayImage;
    //转灰度图
    cvtColor(srcImage, grayImage, COLOR_BGR2GRAY);
    //检测人脸数据
```

```
    std::vector<Rect> faceRects;
    cascade.detectMultiScale(grayImage,faceRects);//检测人脸
    if(faceRects.size()>0 && flag>=0)
    {

        Rect rect = faceRects.at(0);//第一个人脸的矩形框
        //rectangle(srcImage,rect,Scalar(0,0,255));
        //移动人脸框（图片--QLabel）
        ui->headpicLb->move(rect.x,rect.y);

        if(flag > 2){
            //把Mat数据转化为QbyteArray，  -->编码成jpg格式
            std::vector<uchar> buf;
            cv::imencode(".jpg",srcImage,buf);
            QByteArray byte((const char*)buf.data(),buf.size());
            //准备发送
            quint64 backsize = byte.size();
            QByteArray sendData;
            QDataStream stream(&sendData,QIODevice::WriteOnly);
            stream.setVersion(QDataStream::Qt_5_14);
            stream<<backsize<<byte;
            //发送
            msocket.write(sendData);
            flag = -2;

            faceMat = srcImage(rect);
            //保存
            imwrite("./face.jpg",faceMat);
        }
        flag++;
    }
    if(faceRects.size() == 0)
    {
        //把人脸框移动到中心位置
        ui->headpicLb->move(100,60);
        ui->widgetLb->hide();
        flag=0;
    }

    if(srcImage.data == nullptr) return;
    //把opencv里面的Mat格式数据（BGR）转Qt里面的QImage(RGB)
    cvtColor(srcImage,srcImage, COLOR_BGR2RGB);
    QImage image(srcImage.data,srcImage.cols,
srcImage.rows,srcImage.step1(),QImage::Format_RGB888);
    QPixmap mmp = QPixmap::fromImage(image);
    ui->videoLb->setPixmap(mmp);
}

void FaceAttendence::recv_data()
{
    //{employeeID:%1,name:%2,department:软件,time:%3}
    QByteArray array = msocket.readAll();
    qDebug()<<array;
    //Json解析
    QJsonParseError err;
```

```
    QJsonDocument doc = QJsonDocument::fromJson(array,&err);
    if(err.error != QJsonParseError::NoError)
    {
        qDebug()<<"json数据错误";
        return;
    }

    QJsonObject obj = doc.object();
    QString employeeID = obj.value("employeeID").toString();
    QString name = obj.value("name").toString();
    QString department = obj.value("department").toString();
    QString timestr = obj.value("time").toString();

    ui->numberEdit->setText(employeeID);
    ui->nameEdit->setText(name);
    ui->departmentEdit->setText(department);
    ui->timeEdit->setText(timestr);

    //通过样式来显示图片
    ui->headLb->setStyleSheet("border-radius:75px;border-image:
url(./face.jpg);");
    ui->widgetLb->show();
}

void FaceAttendence::timer_connect()
{
    //连接服务器
    msocket.connectToHost("127.0.0.1",9999);
    qDebug()<<"正在连接服务器";
}

void FaceAttendence::stop_connect()
{
    mtimer.stop();
    qDebug()<<"成功连接服务器";
}

void FaceAttendence::start_connect()
{
    mtimer.start(5000);//启动定时器
    qDebug()<<"断开连接";
}
```