7.人脸数据的压缩、发送、接收和显示

人脸数据的压缩和发送

发送的条件：在定时器中检测到人脸数据后 `if(faceRects.size()>0)//检测到人脸`
Qt 不能直接发送Mat格式的图片数据，而是要转化成QByteArray数据后再发送。

```
// Mat 转换为能够发送的数据QByteArray
        // 编码成jpg格式
        std::vector<uchar> buf;
        cv::imencode(".jpg", srcImage, buf);
        // 新建传输的数据格式
        QByteArray byte((const char*)buf.data(),buf.size());

        // 获取数据大小
        quint64 backsize = byte.size();
        // 创建发送对象
        QByteArray sendData;
        // 将用户定义的一些变量保存到文件的模块
        QDataStream stream(&sendData, QIODevice::WriteOnly);
        // 设置QDataStream版本
        stream.setVersion(QDataStream::Qt_5_14);
        // 将数据大小和字节写入sendData
        stream << backsize << byte;

        // 发送
        msocket.write(sendData);
```

服务器端接受人脸并显示

```
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){// 等待接收状态
        if(msocket->bytesAvailable() < (qint64)sizeof(bsize)) return ;
        // 采集数据的长度
        stream >> bsize;
    }

    if(msocket->bytesAvailable() < bsize){// 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
    QPixmap mmp;
    mmp.loadFromData(data, "jpg");
    mmp = mmp.scaled(ui->picLb->size());
    ui->picLb->setPixmap(mmp);
}
```

编解码函数

函数说明

1. imencode()

```
bool imencode(const String& ext,
                                InputArray img,
                                vector<uchar>& buf,
                                const vector<int>& params=vector<int>())
```

参数:
ext-定义输出文件格式的扩展名
img-需要被编码的图像
buf-输出的缓存区，类型是vector
parms-被编码的格式和压缩率,类型是vector

prams目前支持以下参数：
JPEG：它的压缩率范围（cv_imwrite_jpeg_quality）从0到100（越大越好）。默认值是95。100为没有压缩。
WEBP：它的压缩范围（cv_imwrite_webp_quality）从1到100（越大越好）。默认情况下（不含任何参数）和质量在100以上，则使用无损压缩。
PNG：可以压缩级别（cv_imwrite_png_compression）从0到9。更高的值意味着更小的尺寸和更长的压缩时间。默认值是3。
PPM、PGM、或PBM，它可以是一个二进制格式的标志（cv_imwrite_pxm_binary），0或1。默认值是1。
例：

```
std::vector<int> pram = std::vector<int>(2);
parm[0] = IMWRITE_JPEG_OPTIMIZE;
parm[1] = 95;
cv::imencode("1.jpg", mat, buf, parm);
```

2. 解码函数在11节接收识别人脸部分使用。
   `imdecode()`

```
Mat imdecode(InputArray buf, int flags)
Mat imdecode(InputArray buf, int flags, Mat* dst)
```

参数：
buf-输入解压的buf
flags-和imread()的flags是一样的
CV_LOAD_IMOSE_COLOR-如果设置，始终将图像转换为彩色图像
CV_LOAD_IMAGE_GRAYSCALE如果设置，始终将图像转换为灰度图像
dst -解码矩阵的可选输出占位符。不填则是NULL。
例：

```
sed::vector<uchar> decode;
Mat image = imdecode(decode, CV_LOAD_IMAGE_COLOR);
```