厦門大學

信息学院软件工程系

《计算机网络》实验报告

题　　目 实验一　利用可见光传输信息的软件

班　　级 数字媒体技术 2022 级 1 班

姓　　名 魏清晨

学　　号 37220222203790

实验时间 2024 年 9 月 24 日

2024 年 9 月 24 日

填写说明

1、本文件为 Word 模板文件，建议使用 Microsoft Word 2021 打开，在可填写的区域中如实填写；

2、填表时勿改变字体字号，保持排版工整，打印为 PDF 文件提交；

3、文件总大小尽量控制在 1MB 以下，最大勿超过 5MB；

4、应将材料清单上传在代码托管平台上；

5、在实验课结束 14 天内，按原文件发送至课程 FTP 指定位置。

1 实验目的

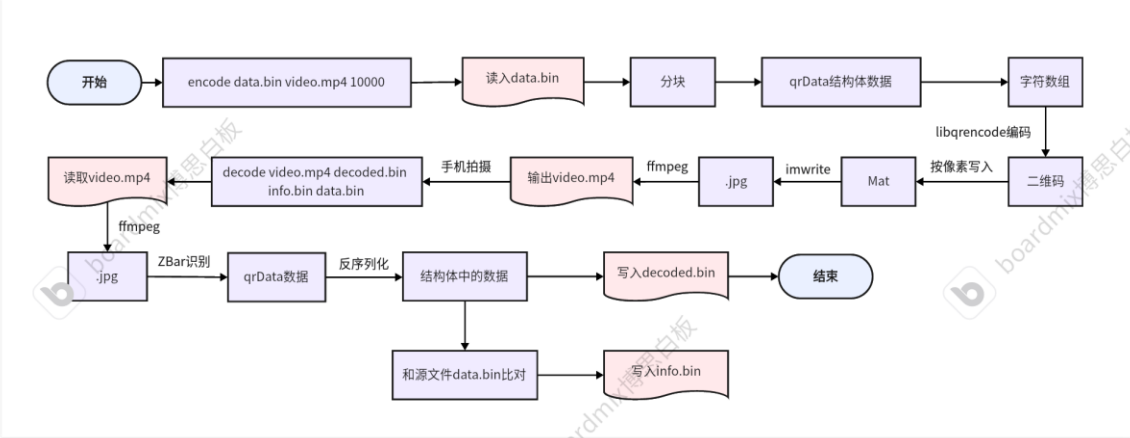
通过完成实验，理解物理层传输的基本原理。掌握传输过程中的编解码过程，熟悉传输中的噪声、分辨率、波特率、调制和误码等通信概念；了解奈氏定理和香农定理的含义。

2 实验环境

操作系统：Win11　编程语言：C++　　调试软件：CLion2023.2

3 实验结果

整体逻辑：



实验步骤：

编码接口调用

```
PS E:\code\practice\Computer_Network\Experiment\cni-exp\E1_3790\bin> .\encode.exe encode data.bin video.mp4 10000
```

等待编码（warning 是 ZBar 库没能识别二维码）

```
Windows PowerShell
Active code page: 65001
每张二维码携带的数据量: 2584B
[############################----------------------] 56% 二维码编码中WARNING: C:/M/B/src/zbar-0.23.93/zbar/decoder/databar.c:1210: _zbar_decode_databar: Assertion "seg->finder >= 0" failed.
        i=11 f=-1(101) part=1
WARNING: C:/M/B/src/zbar-0.23.93/zbar/decoder/databar.c:1210: _zbar_decode_databar: Assertion "seg->finder >= 0" failed.
        i=11 f=-1(101) part=1
WARNING: C:/M/B/src/zbar-0.23.93/zbar/decoder/databar.c:1210: _zbar_decode_databar: Assertion "seg->finder >= 0" failed.
        i=11 f=-1(101) part=1
WARNING: C:/M/B/src/zbar-0.23.93/zbar/decoder/databar.c:1210: _zbar_decode_databar: Assertion "seg->finder >= 0" failed.
        i=11 f=-1(101) part=1
[##################################################] 100% 二维码编码完成
[############################----------------------] 57% 二维码绘制中WARNING: C:/M/B/src/zbar-0.23.93/zbar/decoder/databar.c:1210: _zbar_decode_databar: Assertion "seg->finder >= 0" failed.
        i=11 f=-1(101) part=1
WARNING: C:/M/B/src/zbar-0.23.93/zbar/decoder/databar.c:1210: _zbar_decode_databar: Assertion "seg->finder >= 0" failed.
        i=11 f=-1(101) part=1
WARNING: C:/M/B/src/zbar-0.23.93/zbar/decoder/databar.c:1210: _zbar_decode_databar: Assertion "seg->finder >= 0" failed.
        i=11 f=-1(101) part=1
WARNING: C:/M/B/src/zbar-0.23.93/zbar/decoder/databar.c:1210: _zbar_decode_databar: Assertion "seg->finder >= 0" failed.
        i=11 f=-1(101) part=1
[##################################################] 100% 二维码绘制完成
File 'video.mp4' already exists. Overwrite? [y/N] y
PS E:\code\practice\Computer_Network\Experiment\cni-exp\E1_3790\bin>
```

解码接口调用

```
PS E:\code\practice\Computer_Network\Experiment\cni-exp\E1_3790\bin> .\encode.exe decode video.mp4 decoded.bin info.bin data.bin
```

等待解码完成

```
PS E:\code\practice\Computer_Network\Experiment\cni-exp\E1_3790\bin> .\encode.exe decode video.mp4 decoded.bin info.bin data.bin
Active code page: 65001
[##################################################] 100% 二维码解码完成
文件传输正确率: 99.25%
```

核心代码逻辑：

Main 函数逻辑：

```
int main(int argc, char** argv)
{
    string command = argv[1];
    if (input_func.find( x: command) != input_func.end())
    {
        if (!input_func[command](argc, argv))
        {
            return -1;
        }
        else
        {
            return 1;
        }
    }

    return 0;
}
```

encode 和 decode 指令分别调用对应函数

```cpp
bool encode_input(int argc, char** argv)
{
    system( Command: "chcp 65001");
    // 指令格式，encode <输入文件路径> <输出文件路径> <生成视频时长>
    if (argc < 3) return false;

    string input_file_path = argv[2];
    string output_file_path = argv[3];
    int video_length = stoi( str: argv[4]);

    QrEncoder encoder = QrEncoder();
    if (!encoder.encode( &: input_file_path, &: output_file_path, duration: video_length)) return false;

    return true;
}
```

```cpp
bool decode_input(int argc, char** argv)
{
    system( Command: "chcp 65001");
    // 指令格式，decode <输入文件路径> <输出文件路径> <解码信息输出路径> (<原文件的路径>，用于比较解码准确性)
    if (argc < 5) return false;

    string input_file_path = argv[2];
    string output_file_path = argv[3];
    string output_info_path = argv[4];

    string origin_file_path = argv[5];

    QrEncoder encoder = QrEncoder();
    if (argc == 5)
    {
        if (!encoder.decode( &: input_file_path, &: output_file_path, &: output_file_path)) return false;
    }
    else if (argc == 6)
    {
        if (!encoder.decode( &: input_file_path, &: output_file_path, &: output_info_path, origin_file_path)) return fal
    }

    return true;
}
```

Encode 逻辑：

先根据二维码数据量分割数据，保存到结构体中，再将序列化后的字符串编码为二维码对象

```cpp
    for (int i = ch_per_qr - 1; i < input_file.size(); i += ch_per_qr)
    {
#ifndef DEBUG
        print_progress_bar( progress: (i - (ch_per_qr - 1)) / ch_per_qr, total: input_file.size() / ch_per_qr, info: "二维码编码中");
#endif

        QrData tmp = QrData();
        tmp.index = (i - (ch_per_qr - 1)) / ch_per_qr + 1;
        if (i + ch_per_qr >= input_file.size())
        {
            tmp.data = vector<uchar>( first: input_file.begin() + (i - ch_per_qr), last: input_file.end());
            tmp.end = 1;
        }
        else
        {
            tmp.data = vector<uchar>( first: input_file.begin() + (i - ch_per_qr + 1), last: input_file.begin() + i + 1);
            if (i == ch_per_qr - 1) tmp.start = 1;
        }

        tmp.len = tmp.data.size();

        vector<uchar> tmp_string = serialize( qrData: tmp);

        QrData deserialized_qr = deserialize( serializedData: tmp_string);
        debug_print_qrData( qr_data: deserialized_qr);

#ifdef QRCODEGEN
        ...
#else
    #ifdef QRENCODE
        QRcode qrCode = *QRcode_encodeData( size: tmp_string.size(),
                                            data: reinterpret_cast<const unsigned char *>(tmp_string.data()), version: 0, level: QR_ECLEVEL_H);
    #endif
#endif

        Mat input_image = qrCode_to_mat( qr: qrCode, scale: 10);
        vector<uchar> data;
        decode(input_image, &: data);

        QrData qr_data = deserialize( serializedData: data);
        debug_print_qrData(qr_data);

        qr_arr.push_back(qrCode);
    }
```

随后将二维码转换为 Mat 后写入文件夹，之后再通过调用 ffmpeg（非核心代码）合成视频

```
        // 由QrCode转换为Mat后，由imwrite写入文件夹
        for (int i = 0; i < qr_arr.size(); i++)
        {
#ifndef DEBUG
            print_progress_bar( progress: i,  total: qr_arr.size() - 1,  info: "二维码绘制中");
#endif

#ifdef QRCODEGEN
            ...
#else
    #ifdef QRENCODE
            QRcode qrCode = qr_arr[i];
    #endif
#endif
//            Mat input_image = qrCode_to_mat(qrCode);
            Mat input_image = qrCode_to_mat( qr: qrCode,  scale: 10);

            vector<uchar> data;
            decode(input_image,  &: data);

            QrData qr_data = deserialize( serializedData: data);
            debug_print_qrData(qr_data);

            string img_path = qr_path + std::format( fmt: "\\qrCode_{}.{}",  &: i + 1, image_extension);

            if (!imwrite( filename: img_path,  img: input_image)) return false;
        }
```

Decode 逻辑：

首先对于每一张读入的二维码，先转换为结构体

```
            string img_path = tmp_frame_folder_path + img;
            if (!filesystem::exists( p: img_path)) break;

            Mat mat = imread( filename: img_path);
            if (previous_img && are_images_identical( img1: *previous_img,  img2: mat)) continue;

            mat = convert_to_gray( &: mat);

            vector<uchar> current_qr_data_string;
            decode( input_image: mat,  &: current_qr_data_string,  length: previous_data.data.size());

            if (current_qr_data_string.empty()) continue;
            QrData current_qr_data = deserialize( serializedData: current_qr_data_string);
```

接受到开始信号后，接收数据，并对一些特殊情况进行处理

```
            // 数据接收开始
            if (!data_start)
            {
                if (current_qr_data.start)
                {
                    data_start = true;
                } else
                {
                    continue;
                }
            }

            // 二维码数据序号一样，重复
            if (!previous_data.data.empty() && previous_data.index == current_qr_data.index) continue;

            // 有中间二维码没识别出来
            if (!previous_data.data.empty() && previous_data.index + 1 < current_qr_data.index)
            {
                forup (k, 1, current_qr_data.index - previous_data.index - 1)
                {
                    QrData recovery_qrcode = QrData();
                    recovery_qrcode.index = previous_data.index + k;
                    recovery_qrcode.data = vector<uchar>( n: previous_data.len,  value: 'a');
                    encoded_data.push_back(recovery_qrcode);
                }
            }

            encoded_data.push_back(current_qr_data);
#ifdef DEBUG
    ...
#endif
            previous_img = &mat;
            previous_data = current_qr_data;

            // 数据接收结束
            if (current_qr_data.end) break;
```

ZBar 库识别、解码二维码逻辑：

```
ImageScanner scanner;
scanner.set_config( symbology: ZBAR_QRCODE,  config: ZBAR_CFG_ENABLE,  value: 1);
Image zbar_image( width: input_image.cols,  height: input_image.rows,  format: "Y800", input_image.data,  length: input_image.cols * input_image.rows);
int res = scanner.scan( &: zbar_image);
if (res == 0) return false;

string data;

for (Image::SymbolIterator symbol = zbar_image.symbol_begin();
     symbol != zbar_image.symbol_end();
     ++symbol)
{
    data += symbol->get_data();
}
```

4 实验代码

本次实验的代码已上传于以下代码仓库：[CNI-Exp: 厦门大学计算机网络课程实验项目集 (gitee.com)](gitee.com)

5 课后思考题

1. **在实验中，编解码算法是什么？**

编码算法是依赖于QR Code标准编码，通过选择不同的编码模式，使用Reed-Solomon 纠错码纠错，编码生成二维码。此处使用 libqrencode 库进行编码。

解码算法也是依赖于QR Code标准来进行解码。此处使用zbar库进行识别、解码。

2. 在实验中的，调制和解调算法是什么？其中，载体信号、调制信号是什么？使用的算法属于调频、调幅还是调相？

这里的调试是计算机通过显示器将视频文件调制为光信号；而解调是手机摄像头接收光信号后将其转化为电子信号储存在视频中。

载体信号是光，调制信号是视频文件的电子信号。

使用的算法属于调幅，因为它是通过控制每个像素的红绿蓝信号的高低传输光信号的。

3. 在实验中的，主要的噪音强度有多大，噪音来自哪些因素？

选取生成的二维码和其对应的视频的一帧，计算得到信噪比约为 2，ssim 约为 0.6，可见噪音强度相当大。

一个是手机拍摄时光信号传输过程中，可能有其他光线的干扰，物理介质如屏幕、空气、摄像头在传输过程中不能完整地传输信号，以及可能有其他电气信号的干扰。

4. 你的编码算法分辨率是多少？

libqrencode 编码模式为 0 自动选择，按项目中的示例来讲，是 93x93。

5. 你的编码波特率是多少？传输率是多少？

项目设计的是按照视频长度控制每张二维码的信息量，按项目中的 data.bin 和 10 秒的视频，每张是 2584B，即 20,672bit。

视频默认传输为 10 帧，那么传输率是 20672 * 10 帧/秒 = 206720 bit/s = 206.72 kbps

6. **按奈氏定理和香农定理，通信率上限是多少？**

奈氏定理：$C = 2Blog_2M$，B = 206.72kbps，M = 2，C = 2 * 206.72 * 1.414 = 584.60416。

香农定理：$C = 2Blog_2\left(1+\frac{S}{N}\right)$，B = 206.72kbps，$\frac{S}{N} = 2.2682$，C = 212.63576。

6 实验总结

本次实验对我而言难度很大，但收获颇丰。

首先是对于这个项目的目的本身，通过进行二维码的编码、视频传输，我对噪声、误码有了更直观的感受，通过计算也对波特率有了一定的认识，另外通过 ffmpeg 和视频传输，以及后面的思考题，也对调制在不同信号间的作用有了更明确的了解。

然后是对于这个项目的实现的感受。

首先，在实现项目功能时遇到很多没能预想到的情况，像是 opencv 不能识别某些二维码、视频分出的帧的图片无法被识别、由于手机录像和视频的帧率不一，会有重复二维码或跳过二维码的情况。

对于这些问题，我基本都设计了一到三个综合的解决方案，尽管有些效果不佳，但整体上令这个项目能够按预期地运行了。

其次，我通过使用 C++的宏，实现了测试和发行时不同的控制台输出，也是第一次除#define 外使用其他的宏。

然后，我也对 CLion 的调试功能、Cmakelist 的项目配置有了更深的理解，学会了一些更方便的应用。