

廈門大學



信息学院软件工程系

《计算机网络》实验报告

题 目 实验一 庾晓萍

班 级 软件工程 2020 级卓越班

姓 名 庾晓萍

学 号 20420192201952

实验时间 2022 年 3 月 2 日

2022 年 3 月 2 日

填写说明

- 1、本文件为 Word 模板文件，建议使用 Microsoft Word 2019 打开，在可填写的区域中如实填写；
- 2、填表时勿破坏排版，勿修改字体字号，打印成 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，掌握物理层传输的原理；了解传输过程中的编解码、噪声、分辨率、波特率、调制和误码等通信概念；理解奈氏定理和香农定理。

2 实验环境

操作系统：Windows 10；

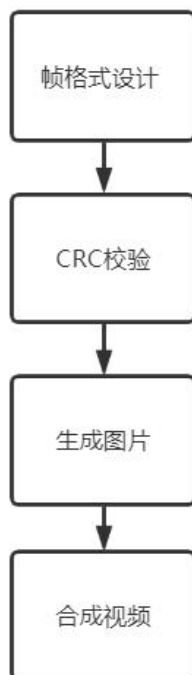
IDE：Visual Studio 2019

编程语言：C++；

3 实验结果

一、编码

0、主要流程



1、帧格式设计

(1) 图片采用彩色编码，通过红、黄、蓝、绿四种颜色表示比特信息。（黄表示 00，红色表示 01……）。

(2) 二维码包含四个定位矩形，其他区域为数据区域。每张二维码包含 5184 (72^2) 个像素格，其中 576 (4×12^2) 格被定位矩形占据。也就是说数据区的像素格为 4608 格，每格传递 2bits 信息，其中有 32bits 是最后加上的 0。故每张二维码的信息为： $2 \times 4608 - 32 = 9184$ bits。

(3) 默认帧率为 10 帧/秒，可以在 Main.cpp 中更改。



2、CRC 效验核心代码

使用模二除法，算法类似高精度除法，但是不进位，每步的运算是异或。

```
//CRC为除数，原始数据data为被除数，模二除法
string CalculateCRC(string data, string& CRC) {
    data.append(string(CRC.size(), '0')); //在原始数据data后面加上校验码位数(k)个0
    //模2除法
    for (int i = 0; i < data.size() - CRC.size(); i++) {
        //如果是0可以跳过，0与其他数异或还是其他数
        if (data[i] == '1') {
            for (int j = 0; j < CRC.size(); j++) {
                //异或运算，相同0，不同1
                (data[i+j] == CRC[j]) ? data[i+j] = '0' : data[i+j] = '1';
            }
        }
    }
    return data.substr(data.size() - CRC.size(), CRC.size()); //返回余数(后k位)
}
```

3、图片生成核心代码

使用 opencv 的 rectangle 函数涂色。每两个比特信息填成一种颜色。

```
void Help_Draw(string code, int& R, int& G, int& B, int& key) {
    if (code[key] == '0' && code[key + 1] == '0') { R = 255; G = 255; B = 0; } //黄色:00
    if (code[key] == '0' && code[key + 1] == '1') { R = 255; G = 0; B = 0; } //红色:01
    if (code[key] == '1' && code[key + 1] == '0') { R = 0; G = 255; B = 0; } //绿色:10
    if (code[key] == '1' && code[key + 1] == '1') { R = 0; G = 0; B = 255; } //蓝色:11
    key = key + 2; //跳两个比特, 继续往下读取code数组
}
```

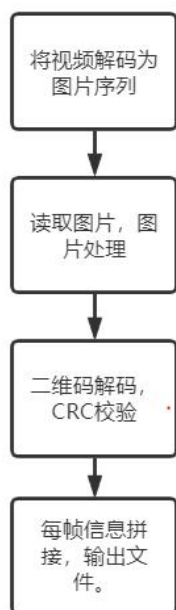
4、将生成的图片序列合成为视频

使用 opencv 的 VideoWriter 函数, 将视频保存在 videoname 路径。

```
int frame_rate = 10; //视频帧率默认为10
cout << "Video frame rate defaults to 10" << endl;
VideoWriter video(videoname, CAP_ANY, frame_rate, Size(780, 780)); //将视频保存成videoname
for (size_t i = 0; i < img.size(); i++) //i:long long unsigned int
{
    Mat image = img[i].clone();
    video << image; // 流操作符, 把图片传入视频
}
```

二、解码

0、主要流程



1、将视频解码为图片序列

使用 opencv 的 VideoCapture 函数，将得到的图片序列储存在 srcImages 中。

```
void Read_Video(string videopath, vector<Mat>& srcImages) {
    VideoCapture capture(videopath); //使用cv2库函数读取视频, 储存在srcImages中
    Mat frame;

    while (1) { //读取视频帧
        capture >> frame;
        if (frame.empty()) break; //停止读取
        else {
            resize(frame, frame, Size(720, 720), 0, 0, INTER_NEAREST); //调整大小 (使用最近邻插值)
            srcImages.push_back(frame.clone()); //将该帧克隆到srcImages中
            frame.release(); //释放视频流
        }
    }
}
```

2、读取图片，图片处理

先进行简单处理，去除部分噪声（彩色转灰度、高斯滤波、二值化。）

```
string Code_Translate(Mat& srcImage, ofstream& verify) {
    Mat midImage, dstImage; //中间生成图像, 最终图像
    string code = "";

    //处理原始图片
    midImage = Handle_Img(srcImage); //简单初步处理, 去噪
    Get_ROI(midImage, srcImage, dstImage); //提取图片中的二维码存入dstImage中
    code = Decode(dstImage, verify); //将图片解码
    return code; //返回图片编码
}
```

3、二维码解码，CRC 校验

将识别到的颜色转化为对应的比特编码。

```
void Help_Decode(Scalar& color, string& code) {
    if (color[1] > 100 && color[2] > 100 && color[0] < 1.5 * color[1] && color[0] < 1.5 * color[2]) {
        code += "00"; //黄色
    }
    else if (color[0] < color[2] / 1.5 && color[1] < color[2] / 1.5) code += "01"; //红色
    else if (color[0] < color[1] / 1.5 && color[2] < color[1] / 1.5) code += "10"; //绿色
    else if (color[1] < color[0] / 1.5 && color[2] < color[0] / 1.5) code += "11"; //蓝色
}
```

4、将每帧信息拼接，输出二进制文件与校验文件。

```

string temp;
for (i = 0; i < srcImages.size(); i++) {
    temp = Code_Translate(srcImages[i], verify); //将图片转换为代码
    if (temp.empty()) continue; //第一次读时，如果当该张图片读空，读取下一张
    else {
        code_array.push_back(temp); //将读到的编码存入编码数组中
        break; //跳出循环
    }
}

```

三、实验结果

(1) 传输速度

实验中，彩色二维码的分辨率是 5184 (72^2) 个像素方块。帧率 10 帧/秒，每张二维码包含 9184 bits 信息。故波特率以及传输速率为 $10 \times 9184 \times 10^{-3} = 91.94 \text{Kb/s}$ 。

(2) 准确度

下面第一幅图为输入的二进制文件，第二幅图为输出的二进制文件，信息大致可以完整传输，但是仍然有传输错误的部分。

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F	ASCII or	...
000000 0A 0D 01 0A 3F 6D 74 74 70	5A 2F 2F 74 68 69 72	... 0h11 p//thlr	
000001 04 71 71 71 62 69 67	6F 22 63 68 2F 67 3F 65	000 0100 0000000	
000002 3D 0F 69 64 62 28 68 3D	3D 72 62 6D 7A 31 65 53	=1d0000=00010000	
000003 07 35 28 62 43 64 65 2D	70 57 3D 6A 47 51 23 73	00010000 00010000	
000004 3D 31 31 3D 3D 12 54 43 5A	5C 55 73 65 72 73 50 31	1000.TC:\Users\1	
000005 38 32 32 32 50 41 70 70	44 61 74 61 5C 52 69 61	0223\app data\ro	
000006 05 65 65 67 5C 54 55 68	63 65 68 74 5C 4F 44 5C	01000000 00010000	
000007 54 65 6D 70 50 51 51 41	70 61 74 61 72 43 61 63	T000\QQ0000000000	
000008 05 65 65 62 28 28 28 28	29 32 33 30 33 33 33 34	00010000 00010000	
000009 38 3D 34 61 38 65 62 66	66 37 1B F7 A5 D3 F1 05	00000000 00010000	
00000A 0A 9C 01 0A 40 6D 74 74	70 5A 2F 2F 74 68 69 72	... 0h11 p//thlr	
00000B 04 71 71 71 62 69 67	6F 22 63 68 2F 67 3F 65	000 0100 0000000	
00000C 3D 0F 69 64 62 28 68 3D	3D 72 62 6D 7A 31 65 53	=1d0000=00010000	
00000D 05 47 21 69 62 72 6D 6D	41 41 72 43 38 66 41 26	00010000 00010000	
00000E 72 3D 31 3D 3D 12 54 43	5A 5C 55 73 65 72 73 50	1000.TC:\Users\1	
00000F 01 38 32 32 32 50 41 70	70 44 61 74 61 5C 52 69	0223\app data\ro	
000010 01 65 69 68 67 5C 54 68	68 62 68 74 5C 4F 44	01000000 00010000	
000011 5C 54 65 6D 70 50 51 51	41 70 61 74 61 72 43 61	T000\QQ0000000000	
000012 03 68 69 5C 31 38 32 38	33 33 34 3F 5C 65 65	00010000 00010000	
000013 51 54 64 3D 61 38 33 68	38 1B FA BA D1 B5 05 0A	00000000 00010000	
000014 9C 01 0A 3F 6D 74 74 70	5A 2F 2F 74 68 69 72 64	... 0h11 p//thlr	
000015 71 71 71 62 69 67 69	6F 22 63 68 2F 67 3F 65	000 0100 0000000	
000016 0F 69 64 62 28 68 3D 32	40 63 65 46 46 74 69 4D	01d0000=00010000	
000017 0A 54 65 6D 43 54 5C 4F	66 70 48 71 77 26 73 3D	00010000 00010000	
000018 31 3D 3D 12 54 43 5A 5C	55 73 65 72 73 50 31 38	1000.TC:\Users\1	
000019 32 32 32 32 50 41 70 70	41 74 61 5C 52 69 61 6D	0223\app data\ro	
00001A 05 64 67 5C 54 65 68 6D	68 62 74 5C 4F 44 5C 54	01000000 00010000	
00001B 65 6D 70 50 51 51 41 70	61 74 61 72 43 61 63 68	00010000 00010000	
00001C 65 5C 38 3C 34 5C 32 3D	55 55 5C 5C 65 65 65 6D	00000000 00010000	
00001D 54 5C 38 38 34 5C 3D 3D	1B 85 C2 7C 85 05 0A 9D	0223\app data\ro	
00001E 01 0A 3F 6D 74 74 70 5A	2F 2F 74 68 69 72 64 71	... 0h11 p//thlr	
00001F 71 71 71 62 69 67 67 2D	6F 22 63 68 2F 67 3F 65	000 0100 0000000	
000020 65 64 62 28 68 3D 34 6A	72 45 4C 69 69 43 54 44	01d0000=00010000	
000021 0F 69 64 44 40 40 61 66	54 6D 47 51 2D 7D 3D 31	00010000 00010000	
000022 31 3D 12 54 43 5A 5C 55	73 65 72 73 50 31 38 35	1000.TC:\Users\1	
000023 32 33 5C 41 70 70 44 61	74 61 5C 52 69 61 6D 69	0223\app data\ro	
000024 05 67 5C 54 65 68 62 68	68 74 5C 4F 44 5C 54 65	01000000 00010000	
000025 0F 70 50 51 51 41 70 61	74 61 72 43 61 63 68 65	00010000 00010000	

+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII of	
000000	02	9D	01	0A	3F	68	74	74	70	0A	29	29	74	68	69	72	...thrt //thlr
000010	24	71	71	25	71	6C	68	67	6F	6C	68	67	67	67	67	62	486...c/c/c/c/c
000020	3D	6F	61	64	42	06	08	39	30	70	62	6D	7A	31	05	53	+addR.k= 0bbbs1e8
000030	47	85	38	6C	46	44	45	25	70	47	28	4A	47	01	29	72	6861C088 8861C088
000040	3D	31	30	30	12	54	43	34	5C	55	73	65	72	73	5C	31	+106.TC\ \Terev\l
000050	38	32	32	33	5C	41	70	70	44	61	74	61	5C	55	69	61	222\app Data\Ro
000060	4D	93	68	67	3C	54	65	68	4C	65	68	74	5C	4F	44	5C	FileT= c/c/D
000070	54	65	6D	70	5C	51	51	41	76	61	74	61	72	43	61	63	Temp\QQA\atarc
000080	68	68	3C	39	30	32	39	39	69	63	68	68	68	68	68	68	88888888 88888888
000090	38	30	34	61	38	45	62	66	66	37	18	F7	85	D3	F1	05	88888888 88888888
0000A0	0A	1C	01	0A	3F	68	74	74	70	0A	29	29	74	68	69	72	...thrt //thlr
0000B0	24	71	71	25	71	6C	68	67	6F	6C	68	67	67	67	67	62	486...c/c/c/c/c
0000C0	3D	6F	61	64	42	06	08	39	30	77	57	4A	37	39	37	5A	+id`sp= 0wJ7B72
0000D0	47	85	38	6C	46	44	45	25	41	41	72	43	38	44	41	20	c61C088 88888888
0000E0	73	3D	31	30	30	12	52	43	38	5C	55	73	65	72	73	5C	+100.RC \Uper
0000F0	31	38	32	32	33	5C	41	70	70	44	61	74	61	5C	55	69	1223\app Data\Ro
000100	41	6D	68	67	3C	54	65	68	4C	65	68	74	5C	4F	44	5C	+18888888 88888888
000110	5C	54	65	6D	70	5C	51	51	41	76	61	74	61	72	43	61	Temp\QQQ\atarc
000120	43	68	68	3C	39	30	32	39	39	69	63	68	68	68	68	68	88888888 88888888
000130	33	33	34	64	35	61	38	33	66	38	18	FA	3A	D1	E5	0A	88888888 88888888
000140	9C	01	0A	3F	68	74	74	70	7A	29	29	74	68	69	72	44	...thrt //thlr
000150	71	71	25	71	6C	68	67	68	6F	6C	68	67	67	67	67	62	486...c/c/c/c/c
000160	6F	69	64	42	06	08	39	30	32	40	43	50	49	48	74	69	+idbk=2 MCFFNt8
000170	3A	54	66	43	54	50	49	49	69	70	48	31	77	28	78	30	21TactP2 88888888
000180	31	30	30	12	53	43	34	3C	55	73	65	72	73	5C	31	38	106.SCT\ \Terev\l
000190	32	32	33	5C	41	70	70	44	41	74	61	5C	55	69	61	60	223\app Data\Ro
0001A0	59	68	67	3C	54	65	68	68	4C	6C	74	5C	4F	44	5C	74	18888888 88888888
0001B0	45	6D	70	5C	51	51	41	76	61	74	61	72	43	61	63	68	Temp\QQQ\atarc
0001C0	65	68	3C	39	30	32	39	39	69	63	68	68	68	68	68	68	88888888 88888888
0001D0	34	32	35	35	38	34	65	39	18	85	CE	C7	F5	05	0A	9D	88888888 88888888
0001E0	01	0A	3F	68	74	74	70	7A	29	29	74	68	69	72	44	71	...thrt //thlr
0001F0	71	71	25	71	6C	68	67	68	6F	6C	68	67	67	67	67	62	486...c/c/c/c/c
000200	69	64	42	06	08	39	30	34	4A	72	45	4C	6F	63	43	54	idb.k=4J 88888888
000210	46	6F	3A	44	45	45	45	45	54	40	47	51	66	72	68	31	88888888 88888888
000220	30	30	12	54	43	32	50	55	73	65	72	73	5C	31	30	32	06.TC\ \Terev\l
000230	32	33	5C	41	70	70	44	61	74	61	5C	55	69	61	60	69	23\app Data\Ro
000240	48	67	3C	54	65	68	68	68	4C	74	5C	4F	44	5C	74	65	FileT= c/c/D
000250	6D	70	5C	51	51	41	76	61	74	61	72	43	61	63	68	65	Temp\QQA\atarc

(3) 输出文件说明

在附加文件包“EA1-1952-庾晓萍”中，包含了附加的文件。其中 input.bin 是输入的二进制文件，output.mp4 是输出的彩色二维码视频文件，byphone.mp4 是手机拍摄下的视频，output.bin 是由手机拍摄的视频解码得到的二进制文件，verify.bin 是作为校验的二进制文件。

4 实验代码

本次实验的代码已上传于以下代码仓库：

https://github.com/ryanregal/ExpOne_ComputerNetwork

5 实验总结

一、实验课后问题

1、在实验中的，编解码算法是什么？

编码算法：核心部分时将字符转化为比特，具体如下图。先将文件内容读入字符数组 `input_string` 中，将字符转换为二进制储存在字符串 `data` 中。

```
for (int i = 0; i < length_char; i++) {
    Binary_Code((unsigned char)input_string[i]); //将字符转换为二进制
    for (int j = 0; j < LEN; j++) { //储存在data中
        if (binary_digit[j] == 1) data.append(1, '1');
        else if (binary_digit[j] == 0) data.append(1, '0');
    }
}
data += "\0"; //结束符
cout << "Finish encode" << endl;
```

解码算法：解码算法先调用 `Read_Video` 函数读取视频，将图片储存在 `srcImages` 数组中。再调用 `Code_Translate` 函数将图片转换为代码。

```
string videopath(videoname); //字符串videopath取值videoname
Read_Video(videopath, srcImages); //读取视频，将图片储存在srcImages数组中

string outfile(outname); //字符串outfile取值outname
ofstream out(outfile, ios::binary); //二进制打开outfile写入out
ofstream verify(verifysname, ios::binary); //二进制打开verifysname写入verify

int i = 0;
string temp;
for (i = 0; i < srcImages.size(); i++) {
    temp = Code_Translate(srcImages[i], verify); //将图片转换为代码
    if (temp.empty()) continue; //第一次读时，如果当该张图片读空，读取下一张
    else {
        code_array.push_back(temp); //将读到的编码存入编码数组中
        break; //跳出循环
    }
}
```

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？

答：调制是将信息搭载到载体上，本次实验中，就是把二进制文件写到一张图片上，调制算法对应绘制二维码数据区的部分。在本次实验中，解调就是从二维码中获取数据，得到二进制文件。载体信号是二维码，调制信号是从文件得到的比特流。在实验中，是通过可见光的不同频率（像素块的不同颜色）来表达不同信息的，所以算法属于调频算法。

调制算法：调制算法将比特流转化为二维码，主要是在比特流末尾加上 CRC 校验码，然后调用绘制二维码。调制算法具体如下图：

```
//读取待测文件字符串，加入crc校验码，用来校验数据传输
void Encode_Crc(string code, int flag) {
    string CRC = "10000010011000001000111011011011";
    code.append(CalculateCRC(code, CRC)); //向code中加入计算得到的余数
    QR_Draw(code, flag); //绘制二维码
}
```

解调算法：解调算法先获取图片，获取图片后进行灰度处理、二值化，然后根据定位码定位，校正图片，最后得到一张二维码的数据区数据。解调算法在 Code_Translate(Mat& srcImage, ofstream& verify)，具体如下图：

```
//将srcImage图片转换为编码
string Code_Translate(Mat& srcImage, ofstream& verify) {
    Mat midImage, dstImage; //中间生成图像，最终图像
    string code = "";

    //处理原始图片
    midImage = Handle_Img(srcImage); //简单初步处理，去噪
    Get_ROI(midImage, srcImage, dstImage); //提取图片中的二维码存入dstImage中
    code = Decode(dstImage, verify); //将图片解码
    return code; //返回图片编码
}
```

3、在实验中的，主要的噪声强度有多大，噪声来自哪些因素？

主要的噪声来自拍摄视频时手机的色差，背景的光线，以及拍摄图片变形。

4、你的编码算法分辨率是多少？（作为实验，不要求分辨率太大）

实验中，彩色二维码的分辨率是每张图 5184（722*722）个像素。

5、你的编码波特率是多少？传输率是多少？

默认帧率 10 帧/秒，每张二维码包含 9184 bits 信息。故波特率以及传输速率为 $10 \times 9184 \times 10^{-3} = 91.94 \text{Kb/s}$ 。

6、按奈氏定理和香农定理，通信率上限是多少？

319.2Kb /s

二、实验思考与反思

通过这一次的计网实验，学习到了很多关于通信的知识，在实验过程中遇到了一些问题，首先是 `opencv` 库配置问题，出现了 `dll` 报错，最后发现是链接器配置问题。然后是像素块的颜色判断，识别手机拍摄视频时出现了错误，没有成功编码，最后修改了 `RPG` 的判断条件。还有在对图片处理时，没有采用掩模的，可能导致背景图片对解码的干扰，可以再对代码进行改进。

另外也和老师讨论了二维码的定位矩形问题。实验中采用了四个定位矩形。四个定位矩形的优点在于可以对变形的拍摄图形更好地处理，代码中用 `opencv` 的 `getPerspectiveTransform` 函数对图片进行了透视变换。但是四个定位矩形也存在不能识别二维码方向的问题，实际中会在右下角的中间用一个小矩形替代。