数字媒体技术选修实验
（一共两个必选实验+一个大项目实践）

刘绍辉
shliu@hit.edu.cn

哈尔滨工业大学计算机科学与技术学院

2023春

实验一

- 实验一
  - 熟悉编程环境
  - 熟悉BMP图像的结构，编程实现BMP图像的阅读和显示
    - 能获取图像任意一点的像素值，计算图像的直方图，计算图像的信息熵
  - 实现图像按照指定块划分，并置乱块的位置显示图像，然后尝试恢复置乱后的图像
    - 能将图像分成任意块大小，PermutationFun(inputImage, blockwidth, blockheight,seed),例如4*4,8*8,16*16,32*32,64*64，并置乱块的位置并显示（类似马赛克效果）；能指定区域内的图像分块并置乱块的顺序再显示（本条可以调用软件或库的读图接口）
    - 对置乱后的图像尝试恢复（类似拼图游戏）
  - 能阅读wav音频文件，并将**原始的PCM音频数据显示**出来，并画出其大小示意图（画出波形图）

实验一

- 调用已有库并实现人脸检测与跟踪
  - 将人脸检测、对齐和识别都集成到一个小软件里面
  - 人脸检测建议可以使用 https://github.com/ShiqiYu/libfacedetection 这个开源的库，是南科大于老师的，经过优化，参数量很少，效果也不错，但是只有检测
  - 另外一个库就是中科院山世光老师组的seetaface，里面检测、对其和识别都有，各种接口都是齐全的，c,c++,python,java都可以调用，网址在这：
  - https://github.com/SeetaFace6Open/index
  - 目标跟踪建议开始使用KCF即可，OPENCV里面应该有现成的函数

实验一

- 实验一
  - 能调用DFT，DCT对图像和音频进行变换处理
    - 对**图像进行二维DCT和DFT正变换和反变换**，并显示正变换后的图像，注意如何才能获得**更好的显示**效果
    - 对图像进行**分块8*8DCT变换**后，将其中的64个DCT系数按照**Zigzag扫描排序**，**设定保留的DCT系数作为函数参数，然后逆变换**，并显示逆变换后的图像，比较原始图像和该图像的**PSNR值和SSIM值**。
  - 能调用JBIG的编码器和解码器，对二值图像进行编码压缩和解码，并在界面上进行显示。（可选）
  - 有以上知识的同学尝试下列任务（可选）
    - 熟悉JPEG压缩的流程，对上述BMP图像按照8*8块分块后进行压缩
    - 对JPEG图像进行加密处理

实验一

- https://panchuangai.blog.csdn.net/article/details/95274845?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-95274845-blog-82385580.pc_relevant_aa&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-95274845-blog-82385580.pc_relevant_aa&utm_relevant_index=2，这个链接有十大标注工具的介绍，大家可以自己上百度上搜最新的，从中选一个作为自己的界面基准就行
- 然后可以将opencv接入到这个工具里面，直接调用opencv的库来完成上述的任务
  - 需要自己安装opencv,如果可以，可以尝试自己编译一下opencv,如果暂时不行，那就直接用opencv已经编译好的库，C++,Python，JAVA都可以调用opencv的库的

实验二

- 跨媒体的检索
  - 根据图像、视频、音频，分别提取其特征，根据某种媒体的特征来检索别的模态的内容
  - 情感属性可以换为：正能量性，积极性等，并进一步给出定量分数。
  - 可以使用CLIP模型，CLIP4CLIP模型等。

大项目实践

- 选修课的同学可以按照3-4人分组，然后每组讨论确定大项目题目，做开题PPT和结题答辩的PPT，并提交源代码和项目设计总结
  - 可以参考第一次课的推荐题目，也可以自己选题，也可以跟老师商量
  - 结课之前提交

要求

- 除BMP图像的编解码外，所有项目都可以调用库和开源代码、工具来实现
- 最后提交实验报告（包括原理介绍、论文阅读、测试数据集和测试结果等）和源代码以及可执行程序
- 具体参见模板。

位图格式

- 位图格式
  - 每行字节数必须是4的整数倍
  - 8比特及其以下图像都带有调色板，采用调色板的索引值来表示图像的像素值，因此可以是彩色的，例如GIF是8比特图像
  - 8比特以上的图像一般没有调色板，直接将图像的RGB值放在相应的位置上
  - 位图文件：14字节的文件头+40字节的信息头+[调色板]+像素数值

BMP图像结构

- **BITMAPFILEHEADER(14 bytes)**

```
typedef struct tagBITMAPFILEHEADER {
        WORD bfType;
        DWORD bfSize;
        WORD bfReserved1;
        WORD bfReserved2;
        DWORD bfOffBits;
} BITMAPFILEHEADER, *PBITMAPFILEHEADER;
```

BITMAP图像结构

- **BITMAPINFO**

```
typedef struct tagBITMAPINFO {
        BITMAPINFOHEADER bmiHeader;
        RGBQUAD bmiColors[1];
} BITMAPINFO, *PBITMAPINFO;
```

BITMAP图像结构

- **BITMAPINFOHEADER(40 Bytes)**

```
typedef struct tagBITMAPINFOHEADER{
        DWORD biSize;
        LONG biWidth;
        LONG biHeight;
        WORD biPlanes;(1)
        WORD biBitCount;
        DWORD biCompression;
        DWORD biSizeImage;
        LONG biXPelsPerMeter;
        LONG biYPelsPerMeter;
        DWORD biClrUsed;
        DWORD biClrImportant;
} BITMAPINFOHEADER, *PBITMAPINFOHEADER;
```

BITMAP图像结构

- **RGBQUAD**

```
typedef struct tagRGBQUAD {
        BYTE rgbBlue;
        BYTE rgbGreen;
        BYTE rgbRed;
        BYTE rgbReserved;
} RGBQUAD;
```

颜色表的起始位置

- pColor = ((LPSTR)pBitmapInfo + (WORD)(pBitmapInfo->bmiHeader.biSize));

位图字节的起始值和长度

- 起始位置：
  - PBITMAPFILEHEADER.bfOffBits;
- 长度：
  - PBITMAPFILEHEADER.bfSize - PBITMAPFILEHEADER.bfOffBits;

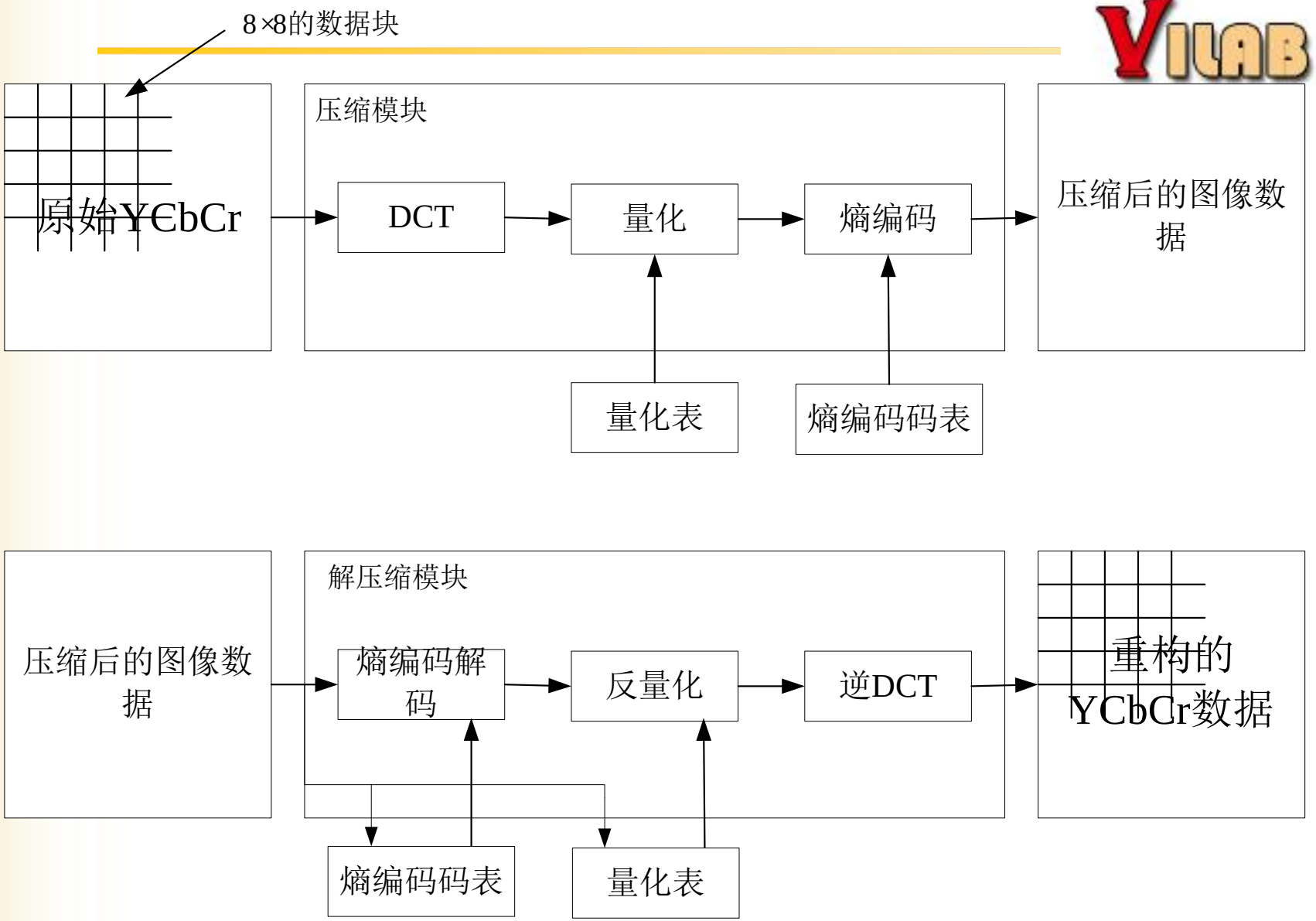
位图显示

- **BOOL StretchBlt(HDC** *hdcDest*, // handle to destination DC **int** *nXOriginDest*, // x-coord of destination upper-left corner **int** *nYOriginDest*, // y-coord of destination upper-left corner **int** *nWidthDest*, // width of destination rectangle **int** *nHeightDest*, // height of destination rectangle **HDC** *hdcSrc*, // handle to source DC **int** *nXOriginSrc*, // x-coord of source upper-left corner **int** *nYOriginSrc*, // y-coord of source upper-left corner **int** *nWidthSrc*, // width of source rectangle **int** *nHeightSrc*, // height of source rectangle **DWORD** *dwRop* // raster operation code **);**

附录：JPEG压缩流程

JPEG图像压缩的基本流程

- RGB->YCbCr
  - 注意Cb+128,Cr+128
- Y-128
- 亮度和色度分量分别分块做DCT变换
- 亮度和色度分量分别用不同的量化矩阵进行量化
  - 注意量化矩阵的计算公式
- DC系数进行处理，ZigZag扫描
- 熵编码



标准量化表和量化因子

$$QuanTable = \begin{cases} round(StdQuanTable \cdot (2 - 0.02 \cdot QualityFactor)) & QualityFactor \geq 50 \\ round(StdQuanTable \cdot (50 / QualityFactor)) & QualityFactor < 50 \end{cases}$$

JPEG推荐的标准亮度量化表

$$StdQuanTable = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

附加

- 如何设计去除块效应的后处理算法
  - JPEG压缩图像随着压缩因子的增大，其块效应愈发明显，主要是由于量化造成边缘区域的失真明显不连续造成的
- 去除块效应算法可以从以下几个方面考虑：
  - 对块的边界处进行平滑处理，可以参考视频编码标准中的环路滤波(de-blocking) 利用JPEG压缩码流中的某些信息来进行补偿，从而减少失真