数字图像处理实验一

姚鸿勋 刘绍辉
shliu@hit.edu.cn

哈尔滨工业大学计算机科学与技术学院

2022秋

实验内容

- 实验一
  - 熟悉编程环境-大家根据自己的喜好和熟悉程序自选即可
    - Opencv,python,java等都有很多现成的项目可以作为基础，QT编写界面，或者MFC编写界面，或者直接使用opencv,python中的界面库都可以！百度的PaddlePaddle飞桨平台，华为的mindspore, fackebook的pytorch,goodle的tensorflow等都可以使用
  - 熟悉BMP图像的结构，编程实现BMP图像的阅读和显示（不能调用现有的库）
  - 统计图像直方图
  - 编程实现彩色BMP图像的三个分量R、G、B的显示，Y、I、Q，H、S、I，Y、Cb、Cr和X、Y、Z的显示
  - 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
    - 能获取图像任意一点的像素值，能将图像分成任意块大小（可以将块的长可宽设置成参数，不足的补0或255），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，可以使用任何算法进行恢复）

实验内容

- 实验一
  - 尝试实现PSNR和SSIM两种图像评价指标
    - PSNR-峰值信噪比
    - SSIM-结构相似性测量指标
  - 将BMP图像转化为GIF图像
    - 颜色进行量化，可以采用传统的聚类算法，也可以采用深度网络来进行真彩色到256种颜色，以及128,64,32,16等种颜色的自动转换
    - 然后再将颜色量化之后的图像保存为BMP图像，注意需要使用调色板来保存（GIF图像可以直接调用库，与本实验的结果进行比较）
  - 有以上知识的同学尝试下列任务
    - 熟悉JPEG压缩的流程，调试基本的JPEG压缩和显示代码，Debug某个具体图像块的压缩过程
    - 对Lena图像在不同压缩因子下的JPEG压缩的性能进行评价
    - 在高压缩比的情况下，会出现明显的块效应，请设计一种有效的能去除块效应的后处理算法

Content

- Experiment 1
  - Read BMP image and Display BMP image, master the structure of BMP file
  - Transfrom RGB color space into different color space, such as Y、I、Q，H、S、I and X、Y、Z, display different components
  - Get each pixel value in the image
  - Additional content
    - Master the JPEG image compression standard，debug the coder and decoder of JPEG，and Debug the complete compression process of one 8×8 image block
    - Evaluate and compare the quality of Lena under different compression quality factor
    - If the compression rate is very high, the image will have many block

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：ＧＩＦ图像介绍

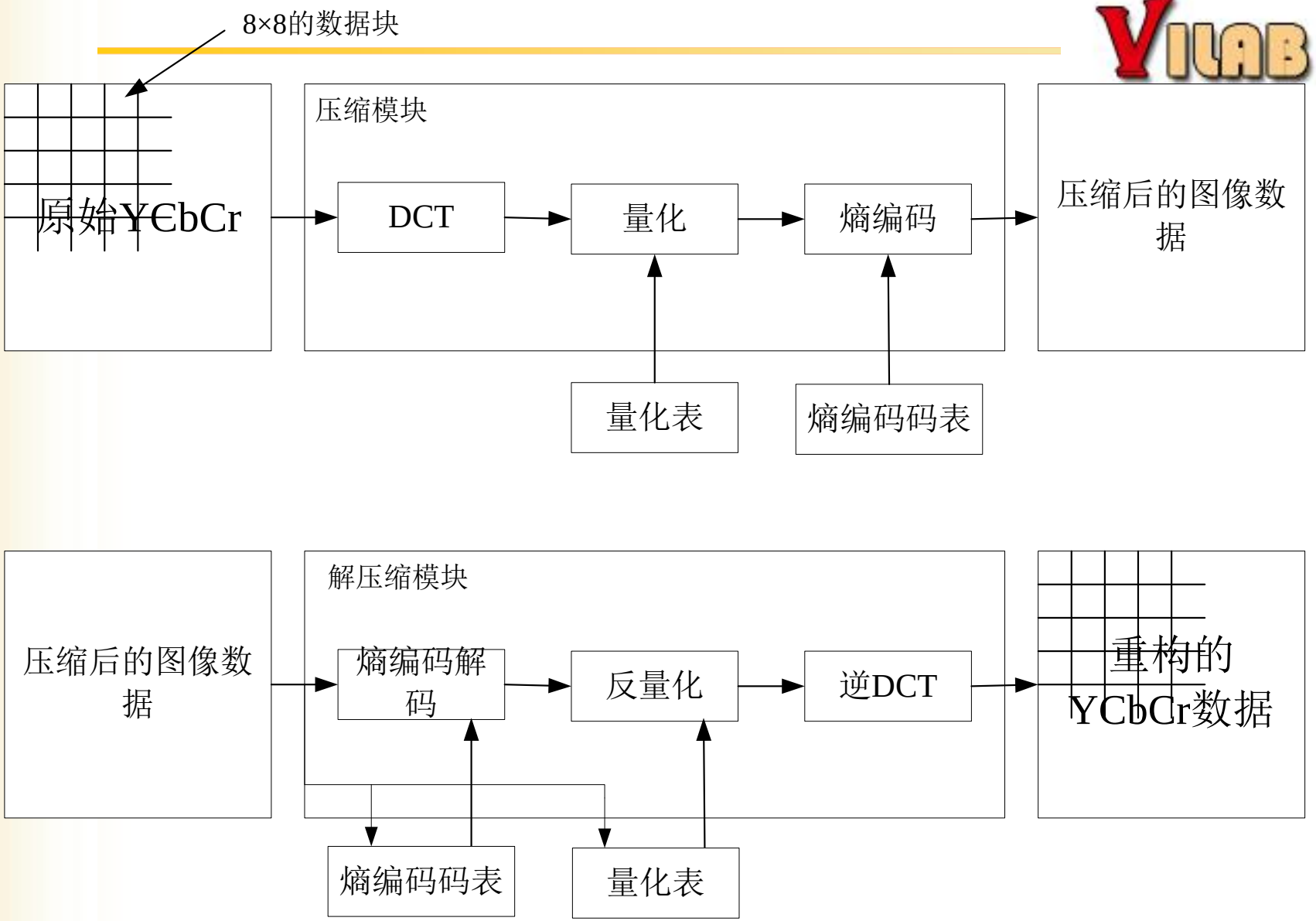
ＧＩＦ图像的特点

- 颜色数目只有２５６种颜色
- 因此，任何真彩色图像都必须将颜色数目降到256种颜色
  - 一组BITMAP中的像素值（R，G，B）就是一种颜色
  - 需要统计图像中的颜色直方图
  - 然后设计聚类或启发式算法来将颜色的数目降低到256种颜色

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

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

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)（推荐）
  - 利用JPEG压缩码流中的某些信息来进行补偿，从而减少失真