专业：计算机科学与技术
姓名：龙永奇
学号：3220105907
日期：2023/10/14

浙江大学实验报告

课程名称：图像信息处理 指导老师：宋明黎 成绩：

实验名称：bmp 文件读写及 rgb 和 yuv 色彩空间转化

一、实验目的和要求

1. 熟悉 BMP 文件结构，学习读取与 BMP 文件的操作。
2. 实现 RGB 与 YUV 彩色空间的相互转换。
3. 生成一张灰度图片。
4. 改变图片的亮度。

二、实验内容和原理

2.1 BMP 文件

位图文件（BMP）是一种图像文件格式，也被称为光栅图像文件。它们以像素为基础来表示图像，每个像素都包含颜色信息，通常以红、绿、蓝（RGB）通道的值来表示。每个像素的颜色和位置都被精确记录，因此位图图像通常是高分辨率的，但它们也可能会占用较大的存储空间。位图文件的主要特点包括：

1. 分辨率：位图图像具有固定的分辨率，通常以像素为单位，例如 800x600 或 1920x1080，因此 BMP 文件在不同大小的显示屏上可能会失去清晰度。
2. 像素化：由于分辨率有限，位图图像可能在放大时出现锯齿状的效果，称为像素化。
3. 像素信息：每个像素都有颜色信息，通常以 RGB 值（红、绿、蓝）或 CMYK 值（青、品红、黄、黑）来表示。
4. 文件格式：常见的位图文件格式包括 BMP、JPEG、PNG、GIF 和 TIFF 等。它们在存储和传输方面都有不同的优缺点。本次实验的文件格式为 BMP 文件。
5. 编辑复杂性：编辑位图图像需要处理每个像素的颜色和位置。
6. 照片和真实图像：因其精确的颜色和细节，位图图像通常用于照片、真实场景的图像和图形设计等领域。

BMP 文件包括四个结构：图像文件头、图像信息头、调色板、图像数据。各结构内容见“3.1

BMP 文件函数头”具体分析。

2.2 BMP 文件读取方法

本次实验所使用的为 24 位全彩 BMP 文件，BMP 文件的可使用 C 语言中<stdio.h>库中的 fopen、fread、fseek、fwite、fclose 等函数进行读取。

2.2 BMP 文件中 RGB 与 YUV 的相互转换

RGB 颜色模型是一个用于表示颜色的系统，它在三维直角坐标颜色空间中创建了一个单位正方体，其中主对角线代表从深黑到纯白的灰度。这个正方体中的各个点代表了不同颜色，其中（0，0，0）表示黑色，（1，1，1）表示白色。此外，正方体的其他六个顶点代表了基本的颜色，包括红色、黄色、绿色、青色、蓝色和品红。

RGB 颜色模型是 CIE 原色空间的一个子集，它被广泛应用于彩色显示器，如彩色 CRT 显示器和彩色液晶屏幕。这一模型的基本思想是使用红色（R）、绿色（G）和蓝色（B）三种原色光的不同强度组合来表示各种颜色。这种模型通常在计算机图形和电视显示中使用，以准确表示和呈现各种颜色对于 RGB 转 YUV 的过程，我们要首先拿到 RGB 文件的数据，再通过 YUV 计算公式对其做运算，得到 YUV 数据，从而实现转换。而对于 YUV 转 RGB 则要首先获得 YUV 数据，用第二组 RGB 公式计算得到 RGB 数据。

RGB 转化为 YUV 的公式如下：

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.435 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

而 YUV 转化为 RGB 的公式如下：

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0000 & 0 & 1.4075 \\ 1.0000 & -0.3455 & -0.7169 \\ 1.0000 & 1.7790 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

三、实验步骤与分析

3.1 BMP 文件函数头（自定义，存放在 BMPHEAD.h 中）

```
#ifndef BMPHEAD
#define BMPHEAD
// 文件头
#pragma pack(1)
typedef struct FILE_HEAD
{
```

```
    unsigned short bfType;      // 19778，必须是 BM 字符串，对应的十六进制为 0x4d42,十进制为 19778，否则不是 bmp 格式文件
    unsigned int bfSize;        // 文件大小 以字节为单位(2-5 字节)
    unsigned short bfReserved1; // 保留，必须设置为 0 (6-7 字节)
    unsigned short bfReserved2; // 保留，必须设置为 0 (8-9 字节)
    unsigned int bfOffBits;     // 从文件头到像素数据的偏移  (10-13 字节)
} FILE_HEAD;
#pragma pack()
// 信息头
#pragma pack(1)
typedef struct INF_HEAD
{
    unsigned int biSize;         // 此结构体的大小 (14-17 字节)
    long biWidth;                // 图像的宽  (18-21 字节)
    long biHeight;               // 图像的高  (22-25 字节)
    unsigned short biPlanes;     // 表示 bmp 图片的平面属，显然显示器只有一个平面，所以恒等于 1 (26-27 字节)
    unsigned short biBitCount;   // 一像素所占的位数，一般为 24   (28-29 字节)
    unsigned int biCompression;  // 说明图象数据压缩的类型，0 为不压缩。 (30-33 字节)
    unsigned int biSizeImage;    // 像素数据所占大小，这个值应该等于上面文件头结构中 bfSize-bfOffBits (34-37 字节)
    long biXPelsPerMeter;        // 说明水平分辨率，用象素/米表示。一般为 0 (38-41 字节)
    long biYPelsPerMeter;        // 说明垂直分辨率，用象素/米表示。一般为 0 (42-45 字节)
    unsigned int biClrUsed;      // 说明位图实际使用的彩色表中的颜色索引数（设为 0 的话，则说明使用所有调色板项）。 (46-49 字节)
    unsigned int biClrImportant; // 说明对图象显示有重要影响的颜色索引的数目，如果是 0，表示都重要。(50-53 字节)
} INF_HEAD;
#pragma pack()
// 调色板
#pragma pack(1)
typedef struct RGB
{
    unsigned char rgbBlue;     // 该颜色的蓝色分量  (值范围为 0-255)
    unsigned char rgbGreen;    // 该颜色的绿色分量  (值范围为 0-255)
    unsigned char rgbRed;      // 该颜色的红色分量  (值范围为 0-255)
    unsigned char rgbReserved; // 保留，必须为 0
} RGB;
#pragma pack()
// 图像本体
#pragma pack(1)
typedef struct FILESTRUCT
{
    FILE_HEAD FH;
```

```
    INF_HEAD IH;
    RGB Color[256];
    unsigned char *types;
} FILESTRUCT;
#pragma pack()
#endif
```

3.2 使用到的全局变量以及 C 语言中的库

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include "BMPHEAD.h"

// 全局变量
int BMPheight;
int BMPwidth;
int BMPSize;
int Pos;
FILESTRUCT Violet;
FILE_HEAD FH;
INF_HEAD IH;
```

3.2 转灰度图

```
// 将图像转换为灰度
void GRAY()
{
    FILESTRUCT Violet1;
    int i, j;

    // 复制 Violet 的文件头和信息头到 Violet1
    memcpy(&(Violet1.FH), &(Violet.FH), sizeof(FILE_HEAD));
    memcpy(&(Violet1.IH), &(Violet.IH), sizeof(INF_HEAD));

    // 设置 Violet1 的文件头偏移和位深度
    Violet1.FH.bfOffBits = 256 * 4 + 40 + 14;
    int Pos1 = (BMPwidth + 3) / 4 * 4;
    Violet1.IH.biBitCount = 8;
    Violet1.IH.biSizeImage = BMPheight * Pos1;
```

```
    Violet1.FH.bfSize = Violet1.IH.biSizeImage + Violet1.FH.bfOffBits;
    Violet1.types = (unsigned char *)calloc(Violet1.IH.biSizeImage, sizeof(unsigned
char));

    // 初始化颜色表
    for (i = 0; i < 256; i++)
        Violet1.Color[i].rgbBlue = Violet1.Color[i].rgbGreen =
Violet1.Color[i].rgbRed = i;

    // 遍历像素并转换为灰度
    for (i = 0; i < BMPheight; i++)
        for (j = 0; j < BMPwidth; j++)
        {
            int POS = Posi(i, j);
            double B = Violet.types[POS];
            double G = Violet.types[POS + 1];
            double R = Violet.types[POS + 2];
            double Y, U, V;
            RGB_YUV(R, G, B, &Y, &U, &V);
            Y = Update(Y);
            Violet1.types[Pos1 * i + j] = Y;
        }

    FILE *fp = fopen("Violet_graywife.bmp", "wb");
    Output(&Violet1, fp);
}
```

3.3 改变亮度

```
// 改变图像亮度
void LUMIN()
{
    int i, j;
    for (i = 0; i < BMPheight; i++)
        for (j = 0; j < BMPwidth; j++){
            int Pos2 = Posi(i, j);
            double B = Violet.types[Pos2];
            double G = Violet.types[Pos2 + 1];
            double R = Violet.types[Pos2 + 2];
            double Y, U, V;
            RGB_YUV(R, G, B, &Y, &U, &V);
            // 增加亮度
            Y *= 1.8;
            Y = Update(Y);
```

```
            YUV_RGB(Y, U, V, &R, &G, &B);
            R = Update(R);
            G = Update(G);
            B = Update(B);
            Violet.types[Pos2] = B;
            Violet.types[Pos2 + 1] = G;
            Violet.types[Pos2 + 2] = R;
        }
    FILE *fp = fopen("Violet_lightwife.bmp", "wb");
    Output(&Violet, fp);
}
```

3.4 RGB 与 YUV 的相互转化

```
// 将 RGB 颜色值转换为 YUV 颜色空间
void RGB_YUV(double R, double G, double B, double *Y, double *U, double *V)
{
    *Y = 0.299 * R + 0.587 * G + 0.114 * B;
    *U = -0.147 * R - 0.289 * G + 0.435 * B;
    *V = 0.615 * R - 0.515 * G - 0.100 * B;
}
// 将 YUV 颜色值转换为 RGB 颜色空间
void YUV_RGB(double Y, double U, double V, double *R, double *G, double *B)
{
    *R = Y + 1.4075 * V;
    *G = Y - 0.3455 * U - 0.7169 * V;
    *B = Y + 1.779 * U;
}
```

四、实验环境及运行方法

4.1 程序编译环境

系统 Windows10

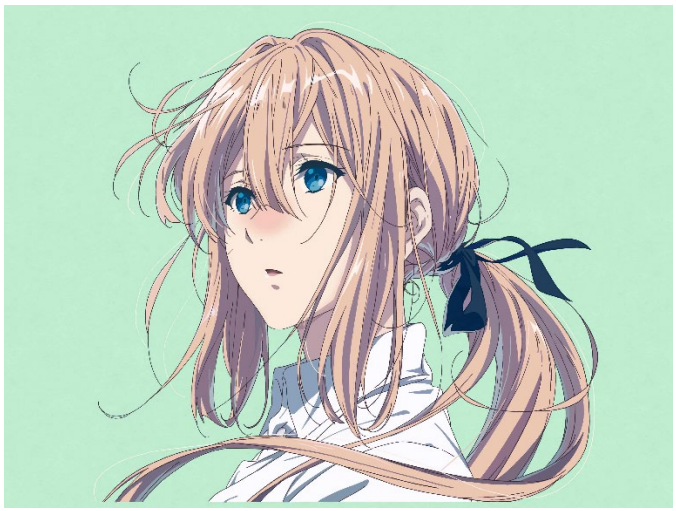
编译器 gcc 10.3.0x86_mingw32

4.2 运行方法

使用编译器打开 Lab1.c 文件编译运行，第一次输出“Violet_lightwife.bmp”提高亮度的图片以及灰度图片，第二次运行前将“viod LUMIN()”函数中“Y=Y*1.8”改为“Y=Y*0.3”，再编译运行即可得到降低亮度图片。

五、实验结果展示

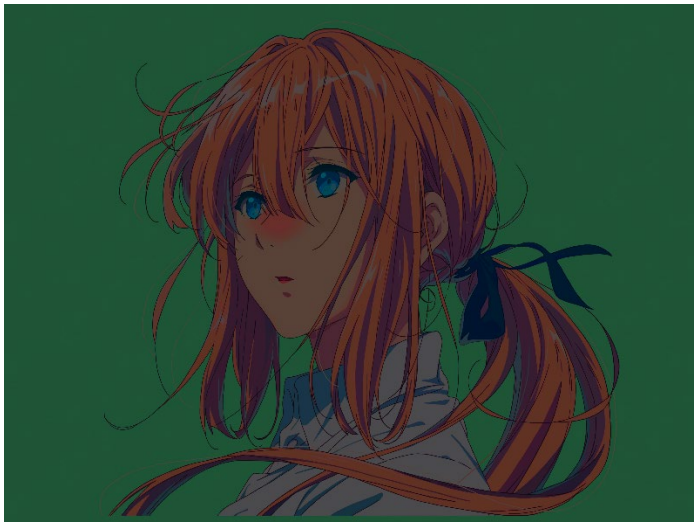
原图像：



灰度：

变亮：

变暗：



六、心得体会

本次实验通过 BMP 文件的读取、灰度转化、RGB 与 YUV 的相互转化的编程实现，学习了解了 BMP 文件的内容格式以及特点，对图像信息处理学有了初步的认知实践。