专业：计算机科学与技术
姓名：武思羽
学号：3220105846
日期：2023/10/11

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

本实验目的在于掌握 bmp 文件的读写程序书写，以及熟练掌握 RGB 和 YUV 色彩空间的相互转化。

本实验要求实现 bmp 文件的读取和写入程序，并且在读入 bmp 文件后将 RGB 色彩格式的像素信息转换为 YUV 色彩格式。随后取每个像素的 Y（亮度）值为该像素在灰度图中的灰度值，并保证灰度范围在【0,255】之间，输出一张原图的灰度转化图。再次改变每个像素的 Y 值，再将 YUV 色彩格式转回为 RGB 色彩格式，输出一张原图的 Y 值改变彩图。

二、实验内容和原理

1.bmp 文件的 c 语言读写程序实现原理：

Image file header
Image information header
Palette
Image data

start	size	name	purpose
1	2	bfType	must always be set to 'BM' to declare that this is a .bmp-file.
3	4	bfSize	specifies the size of the file in bytes.
7	2	bfReserved1	must always be set to zero.

9	2	bfReserved2	must always be set to zero.
11	4	bfOffBits	specifies the offset from the beginning of the file to the bitmap data.

biSize	Number of bytes to define BITMAPINFOHEADER structure
biWidth	Image width (number of pixels)
biHeight	Image height (number of pixels). Note: Besides describing height, “biHeight” can be also denote whether the image is upright or not. (Positive->inverted, Negative->upright). Most of the BMP files are inverted bitmap, namely, biHeight>0.
biPlanes	Number of planes. Always be 1..
biBitCount	Bits per pixel (Bits/pixel), which is 1, 4, 8, 16, 24 or 32.
biCompression	Compression type. Only non-compression is discussed here: BI_RGB.
biSizeImage	Image size with bytes. When biCompression=BI_RGB, biSizeImage=0.
biXPelsPerMeter	Horizontal resolution, pixels/meter.
biYPelsPerMeter	Vertical resolution, pixels/meter
biClrUsed	Number of color indices used in the bitmap (0->all the palette items are used).
biClrImportant	Number of important color indices for image display. 0->all items are important.

bmp 文件结构以及 bmp 文件头、bmp 文件图像信息头所含内容如图所示。c 语言需要通过结构体构建定义这些变量，并通过二进制读取将文件信息存储到这些变量中便于后续操作。在图像数据结构体中，我们定义了图像的宽高、像素位数以及储存每个像素信息的数组。

2. RGB 和 YUV 色彩空间转化原理：

依据已知公式可知 RGB 转化为 YUV 的公式为：

Y = 0.2990 R + 0.5870 G + 0.1140 B

R-Y = 0.7010 R - 0.5870 G - 0.1140 B

B-Y = - 0.2990 R - 0.5870 G + 0.8860 B

YUV 转化为 RGB 的公式为：

R= 1.0Y + 0 +1.402(V-128)

G= 1.0Y - 0.34413 (U-128)-0.71414(V-128)

B= 1.0Y + 1.772 (U-128)+0

在 RGB 色彩空间的图像中，像素色彩存储顺序是 BGRBGRBGR...，一个像素包含三个包含颜色的数据。c 程序需要实现对每个像素中 RGB 颜色信息的读取，并依据公式进行转化。

3. 彩色图转化为灰度图原理：

24 位真彩色图有 3 通道，而 8 位的灰度图只有 1 通道，所以需要将每个像素中的 RGB 三种颜色信息转化为灰度信息。而转化的灰度即为 RGB->YUV 中的 Y 值。同时为了满足灰度范围在[0-255]之间，需要在调色板中定义 256 种灰度颜色。

三、实验步骤与分析

我将项目源代码分为 bmp.h 头文件和 mybmp.c 文件。其中 bmp.h 头文件中包含各个结构体的定义和所需函数的声明，mybmp.c 文件中包含读写函数的实现。

1. bmp 图像文件头信息结构体的定义：

```
typedef struct  {
        //unsigned short    bfType;  //must always be set to 'BM' to declare that this is a .bmp-file.
        unsigned long    bfSize;  //specifies the size of the file in bytes.
        unsigned short    bfReserved1;   // must always be set to zero.
        unsigned short    bfReserved2;  //must always be set to zero.
        unsigned long    bfOffBits;    //specifies the offset from the beginning of the file to the bitmap data.
        } BiFileHeader;
```

结构体 BiFileHeader 定义了图像头文件所需要读写的变量。我没有将 bfType 包含进结构体中，而是选择在读取文件时先将该 unsigned short 类型数据读出，在写入时先向目标文件中写入“BM”。

```
typedef struct  {
        unsigned long  biSize;   //Number of bytes to define BITMAPINFOHEADER structure.
        long   biWidth;   //Image width (number of pixels).
        long   biHeight;   //Image height (number of pixels),positive->inverted,negative->upright,usually be positive.
        unsigned short   biPlanes;   //Number of planes. Always be 1.
        unsigned short   biBitCount;  //Bits per pixel (Bits/pixel), which is 1, 4, 8, 16, 24 or 32.
        unsigned long  biCompression;   //Compression type. Only non-compression is discussed here: BI_RGB.(0)
```

```
        unsigned long  biSizeImage;   //Image size with bytes. When biCompression=BI_RGB,
biSizeImage=0.
        long   biXPelsPerMeter;   //Horizontal resolution, pixels/meter.
        long   biYPelsPerMeter;   //Vertical resolution, pixels/meter
        unsigned long   biClrUsed;   //Number of color indices used in the bitmap (0->all the palette
items are used).
        unsigned long   biClrImportant;   //Number of important color indices for image display.
0->all items are important.
        } BiInfoHeader;
```

结构体 BiInfoHeader 定义了图像信息文件头所需要读写的变量。其中 biBitCount 决定图像类型（彩色图/灰度图），而 biClrUsed 与 biCleImportant 则用于定义调色板。

```
typedef struct   {
        unsigned char rgbBlue; //blue part
        unsigned char rgbGreen; //green part
        unsigned char rgbRed; //red part
        unsigned char rgbReserved; //reserved,0
         } RGBQuad;
```

结构体 RGBQuad 定义了调色板需要的 R、G、B 三颜色变量以及保留变量。

2. bmp 图像文件存储结构体的定义：

```
typedef struct  {
        int width;  //specifies the width of target image.
        int height;  //specfies the height of target image.
        int channels;  //specifies biBitCount of target image.
        unsigned char* imageData;  //store image data.
        int YUVflag;  //when it is 1,means there will be a RGB->YUV and YUV->RGB.
        }Image;
```

结构体 Image 存储了该图像文件的主要信息。其中 width、height 表示图像的宽、高，一般与结构体 BiInfoHeader 中的 biWidth、biHeight 相同。Channels 表示图像通道个数，若图像为彩图，该值为 3；若图像为灰度图，该值为 1。imageData[]数组依字节存储图像信息，按 BGRBGR... 存储像素颜色信息。YUVflag 是函数检测标志，判断该图像是否需要经过 RGB<->YUV 色彩空间转化。

3. bmp 图像文件读取的函数实现：

```
Image* LoadImage(const char* path);  //load target image to a specific Image structure.
```

函数声明如图所示。

```
    Image* bmpImg;
    FILE* fpbmp;  //the bmp file pointer.
```

```
unsigned short fileType;
BiFileHeader bmpfileheader;
BiInfoHeader bmpinfoheader;
int channels = 1;
int width = 0;
int height = 0;
int step = 0;  //step means times needed to read and write image data.
int offset = 0;  //help complete the 4-byte.
unsigned char pixVal;  //pixel value,using unsigned char to represent 8-bit.
RGBQuad* quad;  //the palette.
int i, j, k;
```

该部分定义所需变量。默认图像为灰度图。Offset 将图像信息补全为 4 字节。

```
bmpImg = (Image*)malloc(sizeof(Image));
  fpbmp = fopen(path, "rb");
  if (!fpbmp)
  {
      free(bmpImg);
      return NULL;
  }  //if the read failed,return.

  fread(&fileType, sizeof(unsigned short), 1, fpbmp);  //check whether the bfType is "BM".
```

该部分判断文件读入是否正常，如果不为 bmp 文件，则返回错误信息并退出程序。

```
      fread(&bmpfileheader, sizeof(BiFileHeader), 1, fpbmp);
      fread(&bmpinfoheader, sizeof(BiInfoHeader), 1, fpbmp);

      if (bmpinfoheader.biBitCount == 24)  //this is a real color image.
      {
          channels = 3;
          width = bmpinfoheader.biWidth;
          height = bmpinfoheader.biHeight;

          bmpImg->width = width;
          bmpImg->height = height;
          bmpImg->channels = 3;  //load contents.
          bmpImg->imageData = (unsigned char*)malloc(sizeof(unsigned char)*width*3*height);
          step = channels*width;

          offset = (channels*width)%4;
          if (offset != 0)
          {
              offset = 4 - offset;
          }  //if the step is not the multiple of 4,complete it.
```

将文件的重要信息读入结构体指针中。如果图像信息字节不为 4 的倍数，则用 0 补全。

```
for (i=0; i<height; i++)
         {
              for (j=0; j<width; j++)
              {
                  for (k=0; k<3; k++)
                  {
                      fread(&pixVal, sizeof(unsigned char), 1, fpbmp);
                      bmpImg->imageData[(height-1-i)*step+j*3+k] = pixVal;  //read the image data to
the array imageData[].
                  }
              }
              if (offset != 0)
              {
                  for (j=0; j<offset; j++)
                  {
                      fread(&pixVal, sizeof(unsigned char), 1, fpbmp);
                  }
              }
          }
```

该部分依照从上至下、从左至右的顺序依次读取每个像素的颜色信息，存储在数组中。注意缺省部分同样需要用 0 补全。

4. bmp 图像文件写入函数的实现，以及 RGB<->YUV、灰度转化的实现：

```
int SaveImage(const char* path, Image* bmpImg);  //output the image stored in the Image structure.
```

函数声明如图所示。

```
   FILE *fpbmp;
    unsigned short fileType;
    BiFileHeader bmpfileheader;
    BiInfoHeader bmpinfoheader;
    int step;
    int offset;
    unsigned char pixVal = '\0',pb = '\0', pg = '\0', pr = '\0', py = '\0', pu = '\0', pv = '\0';  //RGB
and YUV temp variables.
    int i, j;
```

定义所需变量。额外定义了 RGB<->YUV 所需的中间变量。

（1）如果要输出的图像为彩图，则补充所有其他头信息，并写入文件。

```
if (bmpImg->channels == 3)  //24bytes, which means the color image.
    {
        step = bmpImg->channels*bmpImg->width;
```

```
        offset = step%4;
        if (offset != 4)
        {
            step += 4-offset;
        }  //complement handler.

        bmpfileheader.bfSize = bmpImg->height*step + 54;
        bmpfileheader.bfReserved1 = 0;
        bmpfileheader.bfReserved2 = 0;
        bmpfileheader.bfOffBits = 54;  //load contents.
        fwrite(&bmpfileheader, sizeof(BiFileHeader), 1, fpbmp);

        bmpinfoheader.biSize = 40;  //total bytes.
        bmpinfoheader.biWidth = bmpImg->width;
        bmpinfoheader.biHeight = bmpImg->height;
        bmpinfoheader.biPlanes = 1;
        bmpinfoheader.biBitCount = 24;
        bmpinfoheader.biCompression = 0;
        bmpinfoheader.biSizeImage = bmpImg->height*step;
        bmpinfoheader.biXPelsPerMeter = 0;
        bmpinfoheader.biYPelsPerMeter = 0;
        bmpinfoheader.biClrUsed = 0;
        bmpinfoheader.biClrImportant = 0;
        fwrite(&bmpinfoheader, sizeof(BiInfoHeader), 1, fpbmp);

 if(bmpImg->YUVflag = 0)  //the common output.
        {
          for (i=bmpImg->height-1; i>-1; i--)
            {
                 for (j=0; j<bmpImg->width; j++)
                 {
                      pixVal = bmpImg->imageData[i*bmpImg->width*3+j*3];
                      fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);
                      pixVal = bmpImg->imageData[i*bmpImg->width*3+j*3+1];
                      fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);
                      pixVal = bmpImg->imageData[i*bmpImg->width*3+j*3+2];
                      fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);  //sequentially output the
pixel values to the image data.
                 }
                 if (offset!=0)
                 {
                     for (j=0; j<4-offset; j++)
                     {
                         pixVal = 0;
```

```
                    fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);
                }
            }
        }
    }
```

如果先前定义的标志为 0，代表无需经过 RGB<->YUV 的转换，正常写入即可。由于 bmp 图像像素一般为逆向排列，故写入时从下至上、从右至左写入像素颜色信息。同样注意缺省补全。

```
else if(bmpImg->YUVflag = 1)  //the RGB->YUV and YUV->RGB changer.
    {
      for (i=bmpImg->height-1; i>-1; i--)
        {
          for (j=0; j<bmpImg->width; j++)
          {
              pb = bmpImg->imageData[i*bmpImg->width*3+j*3];
              pg = bmpImg->imageData[i*bmpImg->width*3+j*3+1];
              pr = bmpImg->imageData[i*bmpImg->width*3+j*3+2];
              py = 0.299*pr + 0.587*pg + 0.114*pb;
              pu = 0.493*(pb-py) + 128;
              pv = 0.877*(pr-py) + 128;

              py /= 2;  //my unique luminance value Y changing formula, you can change whatever
you want.
             pixVal = py + 1.779*(pu-128);
             fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);
             pixVal = py - 0.3455*(pu-128) - 0.7169*(pv-128);
             fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);
             pixVal = py + 1.4075*(pv-128);
             fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);
          }
          if (offset!=0)
          {
             for (j=0; j<4-offset; j++)
             {
                 pixVal = 0;
                 fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);
             }
          }
        }
    }
```

如果标志为 1，代表需要经过 RGB<->YUV 的转换。先分别按 BGR 的顺序将颜色信息存储到中间变量中，再使用实验原理中的公式进行计算。由于本实验要求在输出图片中更改 YUV

格式中的 Y 值，所以我在原图片的 Y 值基础上作 Y /= 2 的变换后再输出。

（2）如果要输出的图像为灰度图，则先补充头信息。

```
step = bmpImg->width;
     offset = step%4;
     if (offset != 4)
     {
         step += 4-offset;
     }

     bmpfileheader.bfSize = 54 + 256*4 + step;  //the total bytes.
     bmpfileheader.bfReserved1 = 0;
     bmpfileheader.bfReserved2 = 0;
     bmpfileheader.bfOffBits = 54 + 256*4;
     fwrite(&bmpfileheader, sizeof(BiFileHeader), 1, fpbmp);

     bmpinfoheader.biSize = 40;
     bmpinfoheader.biWidth = bmpImg->width;
     bmpinfoheader.biHeight = bmpImg->height;
     bmpinfoheader.biPlanes = 1;
     bmpinfoheader.biBitCount = 8;
     bmpinfoheader.biCompression = 0;
     bmpinfoheader.biSizeImage = bmpImg->height*step;
     bmpinfoheader.biXPelsPerMeter = 0;
     bmpinfoheader.biYPelsPerMeter = 0;
     bmpinfoheader.biClrUsed = 256;  //the gray intensity [0,255].
     bmpinfoheader.biClrImportant = 256;
     fwrite(&bmpinfoheader, sizeof(BiInfoHeader), 1, fpbmp);
```

注意此先前时所有的 3 通道信息都要对应转换为 1 通道信息。同时此处定义了调色板的 256 种颜色，对应灰度范围的 256 个取值。

```
quad = (RGBQuad*)malloc(sizeof(RGBQuad)*256);
     for (i=0; i<256; i++)
     {
         quad[i].rgbBlue = i;
         quad[i].rgbGreen = i;
         quad[i].rgbRed = i;
         quad[i].rgbReserved = 0;
     }  //define all gray colors.
     fwrite(quad, sizeof(RGBQuad), 256, fpbmp);
     free(quad);
```

该部分将 256 个灰度取值写入文件。

```
for (i=bmpImg->height-1; i>-1; i--)
      {
          for (j=0; j<bmpImg->width; j++)
          {

                                    pb = bmpImg->imageData[i*bmpImg->width*3+j*3];
                                    pg = bmpImg->imageData[i*bmpImg->width*3+j*3+1];
                                    pr = bmpImg->imageData[i*bmpImg->width*3+j*3+2];
                                    pixVal = 0.299*pr + 0.587*pg + 0.114*pb;  //compute the Y value
to be the gray value of the new image.
              fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);  //3 in 1, 24bytes to 8 bytes.
          }
          if (offset!=0)
          {
              for (j=0; j<4-offset; j++)
              {
                  pixVal = 0;
                  fwrite(&pixVal, sizeof(unsigned char), 1, fpbmp);
              }
          }
      }
```

再利用 Y 值计算公式，每 3 个 char 类型数据转换为 1 个 char 类型数据，对应将 3 通道的彩色像素转换为 1 通道的灰度像素，写入目标文件。

四、实验环境及运行方法

该程序使用 Dev-C++软件编写。在“mybmp”文件夹中我创建了 compiled、src、project、test 四个子文件夹。其中 compiled 文件夹存储可执行文件以及测试图片；src 文件夹存储源代码的.c、.h 文件；project 文件夹存储项目文件。同时，在 test 文件夹中我放入了待测试的图片，您也可以选择任意 bmp 格式的图片。

您可使用 Dev-C++直接打开 project 文件夹中的项目文件“mybmp”查看源代码。如需运行检验，需要按以下步骤执行：

1. 将 test 文件夹中的图片复制 3 份到 compiled 文件夹中，并分别命名为“testimage”、“outimage1”、“outimage2”，代表输入图片、输出的灰度图片、输出的 Y 值改变的彩色图片。

2. 使用 Dev-C++打开“mybmp”项目文件，编译并运行后等待片刻（约 3-5s）。

3. 查看 compiled 文件夹的 outimage1、outimage2 图片，观察是否分别变为 testimage 图片的对应灰度图和有明显颜色改变的图片。如果产生上述变化，说明实验成功。

五、实验结果展示

选择一张颜色较鲜艳的图片作为测试图片。

输出的灰度图如上所示。

输出的 Y 值改变图如上所示。

六、心得体会

在实验开始阶段，由于理论知识的严重不足，我几乎无法将对图像与程序的理解结合在一起，更无法实现对 bmp 图像的读写。在我依据 PPT 和参考资料学习后，我意识到可以利用结构体实现对头部信息的读取。在灰度转换部分，我通过思考意识到要将 3 通道的彩图像素每 3 字节的颜色元素转换为 1 字节的灰度元素，同时还要修改头部信息，才能成功实现转换。通过这次实验，我熟练掌握了使用 C 语言进行 bmp 文件读写操作的程序书写，同时也掌握了 RGB<->YUV 色彩空间的转化，这对我之后的学习将会很有帮助！