哈爾濱工業大學

实验报告

题目	图片色彩空间转换
专业	计算机科学与技术
学号	1160300329
班级	1603107
学生	黄海
指导教师	郑铁然
实验地点	G709
实验日期	11.25

计算机科学与技术学院

一、BMP文件的基本解析

1.1 BMP文件头解析

BMP（全称Bitmap）是Windows操作系统中的标准图像文件格式，可以分成两类：设备有向量相关位图（DDB）和设备无向量关位图（DIB），使用非常广。它采用位映射存储格式，除了图像深度可选以外，不采用其他任何压缩，因此，BMP文件所占用的空间很大。BMP文件的图像深度可选Ibit、4bit、8bit及24bit。BMP文件存储数据时，图像的扫描方式是按从左到右、从下到上的顺序。由于BMP文件格式是Windows环境中交换与图有关的数据的一种标准，因此在Windows环境中运行的图形图像软件都支持BMP图像格式。

在BMP文件中主要有四个部分

- 位图头文件数据结构
- 位图信息数据结构
- 调色板（可选 部分色彩位图需要）
- 位图数据

其中前三个部分为主要的数据头部。其中定义了该文件的大小，头大小和图片尺寸色彩等信息。对于头部的解析是第一步。

```
self.bfType = file.read(2)          # 文件类型
self.bfSize = file.read(4)          # 文件大小
self.bfReserved1 = file.read(2)     # 文件保留字
self.bfReserved2 = file.read(2)     # 文件保留字
self.bfOffBits = file.read(4)       # 数据开始偏移位置
self.biSize = file.read(4)          # 位图数据文件头占据的字节数
self.biWidth = file.read(4)         # 图片宽度
self.biHeight = file.read(4)        # 图片长度
self.biPlanes = file.read(2)        # 目标设备级别
self.biBitCount = file.read(2)      # 像素点需要的位数
self.biCompression = file.read(4)   # 压缩类型
self.biSizeImage = file.read(4)     # 位图大小
self.biXPelsPerMeter = file.read(4) # 水平分辨率
self.biYPelsPerMeter = file.read(4) # 纵向分辨率
self.biClrUsed = file.read(4)       # 实际使用的颜色表个数
self.biClrImportant = file.read(4)  # 重要颜色个数
```

这里我们主要使用长宽和颜色位数来确定图片文件头的大小 对于24和32色图片 **文件头稳定54个字节**

- 24色图片像素三个一组 分别表示（BGR）
- 对于32色图片 四个字节一组

通过这个来确定像素数据开始的位置和大小

1.2 BMP数据的读取

数据的开始我们需要先去计算图片的尺寸 然后来进行计算 在例子中因为使用的是理想图片 所以固定的偏移了54字节 然后进行读取 将数据存入矩阵之中

```
file = open(filePath, "rb")

self.bfType = file.read(2)          # 文件类型
self.bfSize = file.read(4)          # 文件大小
self.bfReserved1 = file.read(2)     # 文件保留字
self.bfReserved2 = file.read(2)     # 文件保留字
self.bfOffBits = file.read(4)       # 数据开始偏移位置
self.biSize = file.read(4)          # 位图数据文件头占据的字节数
self.biWidth = file.read(4)         # 图片宽度
self.biHeight = file.read(4)        # 图片长度
self.biPlanes = file.read(2)        # 目标设备级别
self.biBitCount = file.read(2)      # 像素点需要的位数
self.biCompression = file.read(4)   # 压缩类型
self.biSizeImage = file.read(4)     # 位图大小
self.biXPelsPerMeter = file.read(4) # 水平分辨率
self.biYPelsPerMeter = file.read(4) # 纵向分辨率
self.biClrUsed = file.read(4)       # 实际使用的颜色表个数
self.biClrImportant = file.read(4)  # 重要颜色个数

offset = struct.unpack("<I", self.bfOffBits)[0]
imgsize = struct.unpack("<I", self.bfSize)[0]
bitCount = struct.unpack("<H", self.biBitCount)[0]
width = struct.unpack("<I", self.biWidth)[0]
heigh = struct.unpack("<I", self.biHeight)[0]
self.size = width * heigh
if bitCount == 32:
    self.count = 4
elif bitCount == 24:
    self.count = 3
self.data = np.zeros((self.size, self.count))
for i in range(self.size):
   for j in range(self.count):
       self.data[i][j] = struct.unpack("<B", file.read(1))[0]
```

二、颜色空间的转换

2.1 RGB⇒HSI

我们使用几何方法来进行计算

$$I = \frac{R+G+B}{\sqrt{3}}$$

$$S = 1 - \frac{3 \times min(R,G,B)}{R+G+B}$$

$$H = \begin{cases} \theta, & G \geq B \\ 2\pi - \theta, & G < B \end{cases}$$

$$\theta = arccos \frac{\frac{1}{2} \times ((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}}$$

所以给出算法

```
def hsi(self):
    data = self.data.copy() / 255.0
    for i in range(self.shape_x):
        B = data[i][0]
        G = data[i][1]
        R = data[i][2]
        theta = np.arccos(((R - G) + (R - B)) / (2 * np.sqrt((R - G) * (R - G) + (R - B) * (G -
        B))))
        H = theta if G >= B else 2.0 * np.pi - theta
        S = 1.0 - (3.0 * np.min((R, G, B)) / (R + B + G))
        I = (R + B + G) / 3.0
        data[i][0] = H * 255
        data[i][1] = S * 255
        data[i][2] = I * 255
    return data
```

2.2 RGB⇒YIQ

转换关系如下

$$\begin{cases} Y = 0.299R + 0.587G + 0.114B \\ I = 0.596R - 0.275G - 0.321B \\ Q = 0.212R - 0.523G + 0.311B \end{cases}$$

```
def yiq(self):
    mat = np.array([0.114, 0.587, 0.299, 0, -0.321, -0.275, 0.596, 0, 0.311, -0.523, 0.212, 0,
    0, 0, 0, 1]).reshape(4, 4).T
    data = np.dot(self.data, mat)
    return data
```

2.3 RGB⇒YCbCr

转换关系如下

$$\begin{cases} Y = 0.257R + 0.564G + 0.098B + 16 \\ Cb = -0.148R - 0.291G - 0.439B + 128 \\ Cr = 0.439R - 0.368G - 0.071B + 128 \end{cases}$$

```
def ycbcr(self):
    mat = np.mat([[0.098, 0.564, 0.257, 0], [0.439, -0.291, -0.148, 0], [-0.071, -0.368, 0.439,
    0], [0, 0, 0, 1]]).T
    print(np.shape(mat))
    data = np.dot(self.data, mat)
    for i in range(self.shape_x):
        data[i] = data[i] + np.mat([16, 128, 128, 0])
    return data
```

三、 总结

3.1 请总结本次实验的收获

本次实验较为全面的了解了不同颜色空间的定义，也理解了各种空间之间的转换关系 同时对bmp文件的理解更加的深刻

3.2 请给出对本次实验内容的建议

希望能够在仅仅对数据进行转换的基础上 加入更改文件头的内容