



# 图像信息处理实验报告

**Digital Image Processing (Experiments)**

|  |  |
| --- | --- |
| 姓 名 |  |
| 指导老师 | **宋明黎** |
| 学 号 |  |
| 专业班级 | **信安1801** |

**二〇一九年**

**秋冬学期**

## 实验一

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称：** | 作业1 | **指导老师：** | 宋明黎 | **成绩：** |  |

### 一、实验目的和要求（必填）

·读取一张彩色bmp图像，并将RGB转换成YUV；

·把bmp图像的颜色转换成灰色，灰度在YUV颜色区间内为Y；

·重映射灰度到[0, 255]，然后写一个灰度bmp；

·改变亮度值Y，再将YUV转换为RGB并输出一张bmp彩色图像。

### 二、实验内容和原理（必填）

1、bmp图像数据格式

·bmp图形文件（BitmapFile）是Windows的一种图像文件储存格式，bmp位图文件默认文件扩展名为bmp或dib。

·bmp位图文件可以分为四个部分：

1. 位图文件头（bmpfileheader）

提供文件格式、大小等信息。

1. **typedef** **struct** tagBITMAPFILEHEADER {
2. **WORD** bfType; /\* 说明文件的类型 \*/
3. **DWORD** bfSize; /\* 说明文件的大小，用字节为单位 \*/
4. **WORD** bfReserved1; /\* 保留，设置为0 \*/
5. **WORD** bfReserved2; /\* 保留，设置为0 \*/
6. **DWORD** bfOffBits; /\* 说明从BITMAPFILEHEADER结构开始到实际的图像数据之间的字节偏移量 \*/
7. } BITMAPFILEHEADER;
8. 位图信息头（bmpinforheader）

提供图像数据尺寸、位平面数、压缩方式、颜色索引等信息。

1. **typedef** **struct** tagBITMAPINFOHEADER {
2. **DWORD** biSize; /\* 说明结构体所需字节数 \*/
3. **LONG** biWidth; /\* 以像素为单位说明图像的宽度 \*/
4. **LONG** biHeight; /\* 以像素为单位说明图像的高速 \*/
5. **WORD** biPlanes; /\* 说明位面数，必须为1 \*/
6. **WORD** biBitCount; /\* 说明位数/像素，1、2、4、8、24 \*/
7. **DWORD** biCompression; /\* 说明图像是否压缩及压缩类型BI\_RGB，BI\_RLE8，BI\_RLE4，BI\_BITFIELDS \*/
8. **DWORD** biSizeImage; /\* 以字节为单位说明图像大小，必须是4的整数倍\*/
9. **LONG** biXPelsPerMeter; /\*目标设备的水平分辨率，像素/米 \*/
10. **LONG** biYPelsPerMeter; /\*目标设备的垂直分辨率，像素/米 \*/
11. **DWORD** biClrUsed; /\* 说明图像实际用到的颜色数，如果为0，则颜色数为2的biBitCount次方 \*/
12. **DWORD** biClrImportant; /\*说明对图像显示有重要影响的颜色索引的数目，如果是0，表示都重要。\*/
13. } BITMAPINFOHEADER;
14. 调色板（color palette）

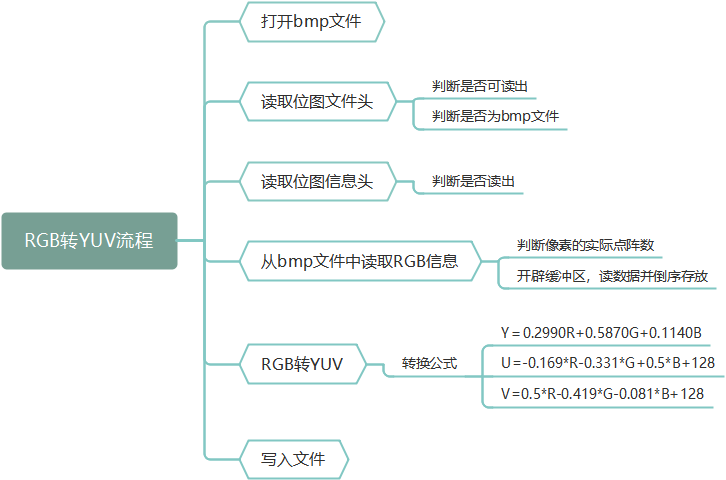
实际上是一个数组，包含元素与位图具有的颜色相同，决定于biClrUsed和biBitCount字段，数组中每个元素的类型是一个RGBQUAD结构，真彩色无调色板部分。

对于用到调色板的位图，图像数据就是该像素颜色在调色板中的索引值（逻辑色）。对于真彩色图，图像数据就是实际的R、G、B值。图像的每一扫描行由表示图像像素的连续的字节组成，每一行的字节数取决于图像的颜色数目和用像素表示的图像宽度。规定每一扫描行的字节数必需是4的整倍数，也就是DWORD对齐的。扫描行是由底向上存储的，这就是说，阵列中的第一个字节表示位图左下角的像素，而最后一个字节表示位图右上角的像素。

1. **typedef** **struct** tagRGBQUAD {
2. **BYTE** rgbBlue; /\*指定蓝色分量\*/
3. **BYTE** rgbGreen; /\*指定绿色分量\*/
4. **BYTE** rgbRed; /\*指定红色分量\*/
5. **BYTE** rgbReserved; /\*保留，指定为0\*/
6. } RGBQUAD;
7. 实际位图数据（imagedata）

即图像数据区。位图数据记录了位图的每一个像素值，记录顺序是在扫描行内是从左到右,扫描行之间是从下到上。当biBitCount=24时，1个像素占3个字节。Windows规定一个扫描行所占的字节数必须是4的倍数（即以long为单位），不足的以0填充。

2、RGB转YUV



3、YUV文件格式

·电视信号中用的就是一种类似于YUV的颜色表示方法。在这种表示方法中，Y分量的物理含义就是亮度，Y分量包含了灰度图的所有信息，只用Y分量就能完全能够表示出一幅灰度图来。  
·从 RGB 到 YUV 空间的 Y 转换公式为：Y = 0.299R+0.587G+0.114B。将彩色转换为灰度时候，按照公式计算出对应的值，该值实际上是亮度的级别，亮度从 0 到 255。由于不同的位有不同的亮度级别，所以 Y 的具体取值为Y = Y/ (1<<(8- 转换的位数 ))，只需要把图像数据按每个点的位数都转换成相同的灰度值即可。

1. YUV转RGB

·小数形式公式，未量化

R = Y + 1.4075 \* (V-128) ;

G = Y - 0.3455 \* (U-128) - 0.7169\*(V-128) ;  
B = Y + 1.779 \* (U-128) ;

·整数形式公式（减少计算量），未量化

R = Y + ((360 \* (V - 128))>>8) ;  
G = Y - (( ( 88 \* (U - 128) + 184 \* (V - 128)) )>>8) ;   
B = Y +((455 \* (U - 128))>>8) ;

·量化后公式（Y~(16,235)  U/V ~(16,240) ）

R = 1.164\*Y + 1.596 \* V - 222.9 ;  
G = 1.164\*Y - 0.392 \* U - 0.823 \* V+ 135.6 ;  
B = 1.164\*Y + 2.017 \* U- 276.8 ;

### 源代码与分析

1. 源代码
2. #include <stdio.h>
3. #include <stdlib.h>
5. #define BMPFILEHEADERLENGTH 14 //the bmp fileheader length is 14
6. #define MAXSIZE 10000
8. unsigned **char** Y[MAXSIZE][MAXSIZE], output\_Y[MAXSIZE][MAXSIZE];
9. unsigned **char** U[MAXSIZE][MAXSIZE], output\_U[MAXSIZE][MAXSIZE];
10. unsigned **char** V[MAXSIZE][MAXSIZE], output\_V[MAXSIZE][MAXSIZE];
12. **void** RGB2YUV();
13. **void** YUV2Gray();
14. **void** YUV2RGB();
15. **void** ChangeTheLuminance();
17. **void** bmpHeaderPartLength(**FILE**\* fpbmp);  //get the offset of header to data part
18. **void** bmpWidthHeight(**FILE**\* fpbmp);  //get the width and height of the bmp
19. **void** bmpDataPart(**FILE**\* fpbmp);  //get R, G, B data
20. **void** bmpoutput(**FILE**\* fpout);  //output data to corresponding txt file
22. //rewrite the file
23. **void** Initialization();
24. **FILE** \*openfile(**char** filename[]);  //open a picture
25. **FILE** \*writefile(**char** filename[]);  //create a picture
26. **FILE** \*AddHeader2File(**FILE** \*input, **FILE** \*output);  //add head to file
28. unsigned **int** Offset = 0;  //offset from Header part to Data part
29. **long** width;  //width of the Data part
30. **long** height;  //height of the Data part
31. unsigned **char** R[MAXSIZE][MAXSIZE], output\_R[MAXSIZE][MAXSIZE];
32. unsigned **char** G[MAXSIZE][MAXSIZE], output\_G[MAXSIZE][MAXSIZE];
33. unsigned **char** B[MAXSIZE][MAXSIZE], output\_B[MAXSIZE][MAXSIZE];
34. unsigned **char** records[MAXSIZE][MAXSIZE];
36. **int** main(**int** argc, **char**\* argv[])
37. {
38. /\*open bmp file\*/
39. unsigned **char** \*fp\_temp;
41. **FILE** \*fpbmp;
42. **FILE** \*fpout;
43. Initialization();
45. /\*standard operations for file\*/
46. fpbmp = openfile("1.bmp");  //open the source file
47. bmpDataPart(fpbmp);
49. fpout = writefile("2.bmp");  //output the file
50. AddHeader2File(fpbmp, fpout);
52. /\*operations for the picture\*/
53. RGB2YUV();
54. YUV2Gray();  //include rearrange
55. //ChangeTheLuminance();
56. YUV2RGB();
58. bmpoutput(fpout);
60. /\*standard operations for close\*/
61. fclose(fpbmp);
62. fclose(fpout);
64. **return** 0;
65. }
67. **void** bmpoutput(**FILE**\* fpout)
68. {
69. **long** i, j = 0;
70. **long** stride;
71. unsigned **char**\* pixout = NULL;
73. stride = (24 \* width + 31) / 8;
74. stride = stride / 4 \* 4;
75. pixout = malloc(stride);
77. fseek(fpout, Offset, SEEK\_SET);
79. **for** (j = 0; j < height; j++) {
80. **for** (i = 0; i < width; i++) {
81. pixout[i \* 3 + 2] = output\_R[height - 1 - j][i];
82. pixout[i \* 3 + 1] = output\_G[height - 1 - j][i];
83. pixout[i \* 3] = output\_B[height - 1 - j][i];
84. }
85. fwrite(pixout, 1, stride, fpout);
86. }
87. }
89. **void** bmpDataPart(**FILE**\* fpbmp)
90. {
91. **int** i, j = 0;
92. **int** stride;
93. unsigned **char**\* pix = NULL;
95. fseek(fpbmp, Offset, SEEK\_SET);
96. stride = (24 \* width + 31) / 8;
97. stride = stride / 4 \* 4;
98. pix = malloc(stride);
99. **for** (j = 0; j < height; j++) {
100. fread(pix, 1, stride, fpbmp);
102. **for** (i = 0; i < width; i++) {
103. R[height - 1 - j][i] = pix[i \* 3 + 2];
104. G[height - 1 - j][i] = pix[i \* 3 + 1];
105. B[height - 1 - j][i] = pix[i \* 3];
107. output\_R[height - 1 - j][i] = 255;
108. output\_G[height - 1 - j][i] = 255;
109. output\_B[height - 1 - j][i] = 255;
110. }
111. }
112. }

115. //get the offset of header to data part
116. **void** bmpHeaderPartLength(**FILE**\* fpbmp)
117. {
118. fseek(fpbmp, 10L, SEEK\_SET);
119. fread(&Offset, **sizeof**(**char**), 4, fpbmp);
120. printf("Header Part is of length %d.\n", Offset);
121. }
123. //get the width and height of the bmp file
124. **void** bmpWidthHeight(**FILE**\* fpbmp)
125. {
126. **int** size;
127. fseek(fpbmp, 18L, SEEK\_SET);
128. fread(&width, **sizeof**(**char**), 4, fpbmp);
129. fseek(fpbmp, 2L, SEEK\_SET);
130. fread(&size, **sizeof**(**char**), 4, fpbmp);
131. printf("The size of the bmp file is %ld.\n", size);
132. fseek(fpbmp, 22L, SEEK\_SET);
133. fread(&height, **sizeof**(**char**), 4, fpbmp);
134. printf("The width of the bmp file is %ld.\n", width);
135. printf("The height of the bmp file is %ld.\n", height);
136. }
138. **void** RGB2YUV()
139. {
140. //convert
141. **int** i=0,j=0;
143. **for** (j = 0; j < height; j++) {
144. **for** (i = 0; i < width; i++) {
145. Y[j][i] = 0.299\*R[j][i] + 0.587\*G[j][i] + 0.114\*B[j][i];
146. U[j][i] = -0.147\*R[j][i] - 0.289\*G[j][i] + 0.436\*B[j][i];
147. V[j][i] = 0.615\*R[j][i] - 0.515\*G[j][i] + 0.100\*B[j][i];
148. }
149. }
150. }
152. **void** YUV2Gray()
153. {
154. **int** i=0,j=0;
155. **for** (j = 0; j < height; j++) {
156. **for** (i = 0; i < width; i++) {
157. output\_Y[j][i] = Y[j][i];
158. output\_U[j][i] = 0;
159. output\_V[j][i] = 0;
160. }
161. }
162. }
163. **void** YUV2RGB()
164. {
165. **int** i=0,j=0;
166. **for** (j = 0; j < height; j++) {
167. **for** (i = 0; i < width; i++) {
168. output\_R[j][i] = output\_Y[j][i] + 1.140\*output\_V[j][i];
169. output\_G[j][i] = output\_Y[j][i] - 0.394\*output\_U[j][i] - 0.581\*output\_V[j][i];
170. output\_B[j][i] = output\_Y[j][i] + 2.032\*output\_U[j][i];
171. }
172. }
173. }
175. **void** ChangeTheLuminance()
176. {
177. **int** i=0,j=0;
178. **for** (j = 0; j < height; j++) {
179. **for** (i = 0; i < width; i++){
180. output\_Y[j][i] = 255 - Y[j][i];
181. output\_U[j][i] = U[j][i];
182. output\_V[j][i] = V[j][i];
183. }
184. }
185. }
187. //functions for files
188. **void** Initialization()
189. {
190. **int** i=0,j=0;
191. **for** (j = 0; j < height; j++) {
192. **for** (i = 0; i < width; i++) {
193. output\_Y[i][j] = 255;
194. output\_U[j][i] = 0;
195. output\_V[j][i] = 0;
196. records[j][i] = 0;
197. }
198. }
199. }
201. **FILE** \*openfile(**char** filename[])
202. {
203. **FILE** \*fpbmp;
204. fpbmp = fopen("1.bmp", "rb");
205. **if** (fpbmp == NULL) {
206. printf("Open bmp failed!\n");
207. exit(-1);
208. }

211. bmpHeaderPartLength(fpbmp);
212. bmpWidthHeight(fpbmp);
214. fseek(fpbmp, 0L, SEEK\_SET);
215. **return** fpbmp;
216. }
217. **FILE** \*writefile(**char** filename[])
218. {
219. **FILE** \*fpout;
220. fpout = fopen("2.bmp", "wb+");  //write the grayscale bmp
221. //fpout = fopen("3.bmp", "wb+");  //write the color bmp
222. **if** (fpout == NULL) {
223. printf("Open out bmp failed.\n");
224. exit(-1);
225. }
226. fseek(fpout, 0L, SEEK\_SET);
227. **return** fpout;
228. }
230. **FILE** \*AddHeader2File(**FILE** \*input, **FILE** \*output)
231. {
232. unsigned **char** \*fp\_temp;
234. fseek(input, 0L, SEEK\_SET);
235. fseek(output, 0L, SEEK\_SET);
237. fp\_temp = malloc(Offset);
238. fread(fp\_temp, 1, Offset, input);  //read the head file
239. fp\_temp[18] = (**int**)width;
240. fp\_temp[22] = (**int**)height;
241. fp\_temp[2] = (**int**)(Offset + height \* width \* 3);
242. fp\_temp[34] = (**int**)(height\*((24 \* width / 8 + 3) / 4 \* 4));
244. fwrite(fp\_temp, 1, Offset, output);
245. }

2、源代码功能解释

函数RGB2YUV()，YUV2Gray()，YUV2RGB()，ChangeTheLuminance()分别实现了将RGB图像转换成YUV格式，将YUV转换为灰度图，再将YUV转换为RGB，以及改变亮度值Y。

若要得到灰度图，则注释掉main()函数中的ChangeTheLuminance();以及FILE \*writefile(char filename[])中的fpout = fopen("3.bmp", "wb+"); 反之若要改变Y的亮度值，则注释掉main()函数中的YUV2Gray();以及FILE \*writefile(char filename[])中的fpout = fopen("2.bmp", "wb+");

### 心得体会

·这次实验中由于对bmp等的格式认识存在问题，将原本简单的问题复杂化了，浪费了大量时间，其实可以用Windows自带的画图工具调色板直接获得24位bmp。

·这次实验暴露了对文件的一些操作不熟悉，经过这个过程对基本的开闭写入等操作有了进一步的认识和体会，也为后面对图像进行二值化等处理做了准备。

·写完之后发现其实只要将实验原理和流程弄清，数据结构并不复杂，在实验之前应该充分弄清实验原理和实验流程，这样有助于我们的编程也能提高工作效率。

·一直拖延一直爽，希望在接下来的实验中能改掉拖延的毛病，又快又好地完成实验。