

## 原 Tiny Jpeg Decoder （JPEG解码程序） 源代码分析 2：解码数据

2013年10月11日 16:00:12 阅读数：4989

注：分析Tiny Jpeg Decoder源代码的文章：

[Tiny Jpeg Decoder （JPEG解码程序） 源代码分析 1：解码文件头](#)

[Tiny Jpeg Decoder （JPEG解码程序） 源代码分析 2：解码数据](#)



=====

Tiny Jpeg Decoder是一个可以用于嵌入式系统的JPEG解码器。也可以在Windows上编译通过。在此分析一下它部分的源代码，辅助学习JPEG解码知识。

通过TinyJpeg可以将JPEG (\*.jpg) 文件解码为YUV (\*.yuv) 或者RGB (\*.tga) 文件。

真正的解码数据开始于tinyjpeg\_decode()函数：

注意：本代码中包含部分自己写的代码，用于提取DCT系数表，解码后亮度数据表等数据。

```
[cpp]  
1.  /**
2.   * Decode and convert the jpeg image into @pixfmt@ image
3.   *解码函数
4.   * Note: components will be automatically allocated if no memory is attached.
5.   */
6.  int tinyjpeg_decode(struct jdec_private *priv, int pixfmt)
7.  {
8.      unsigned int x, y, xstride_by_mcu, ystride_by_mcu;
9.      unsigned int bytes_per_blocklines[3], bytes_per_mcu[3];
10.     decode_MCU_fct decode_MCU;
11.     const decode_MCU_fct *decode_mcu_table;
12.     const convert_colorspace_fct *colorspace_array_conv;
13.     convert_colorspace_fct convert_to_pixfmt;
14.
15.     //-----
16.     FILE *fp;
17.     char *temp;
18.     int j,k;
19.     //-----
20.
21.     if (setjmp(priv->jump_state))
22.         return -1;
23.
24.     /* To keep gcc happy initialize some array */
25.     bytes_per_mcu[1] = 0;
26.     bytes_per_mcu[2] = 0;
27.     bytes_per_blocklines[1] = 0;
28.     bytes_per_blocklines[2] = 0;
29.
30.     decode_mcu_table = decode_mcu_3comp_table;
31.     switch (pixfmt) {
32.         case TINYJPEG_FMT_YUV420P:
33.             colorspace_array_conv = convert_colorspace_yuv420p;
34.             if (priv->components[0] == NULL)
35.                 priv->components[0] = (uint8_t *)malloc(priv->width * priv->height);
36.             if (priv->components[1] == NULL)
37.                 priv->components[1] = (uint8_t *)malloc(priv->width * priv->height/4);
38.             if (priv->components[2] == NULL)
39.                 priv->components[2] = (uint8_t *)malloc(priv->width * priv->height/4);
40.             bytes_per_blocklines[0] = priv->width;
41.             bytes_per_blocklines[1] = priv->width/4;
42.             bytes_per_blocklines[2] = priv->width/4;
43.             bytes_per_mcu[0] = 8;
44.             bytes_per_mcu[1] = 4;
45.             bytes_per_mcu[2] = 4;
46.             break;
47.
48.         case TINYJPEG_FMT_RGB24:
49.             colorspace_array_conv = convert_colorspace_rgb24;
50.             if (priv->components[0] == NULL)
51.                 priv->components[0] = (uint8_t *)malloc(priv->width * priv->height * 3);
52.             bytes_per_blocklines[0] = priv->width * 3;
53.             bytes_per_mcu[0] = 3*8;
54.             break;
55.
56.         case TINYJPEG_FMT_BGR24:
57.             colorspace_array_conv = convert_colorspace_bgr24;
58.             if (priv->components[0] == NULL)
59.                 priv->components[0] = (uint8_t *)malloc(priv->width * priv->height * 3);
60.             bytes_per_blocklines[0] = priv->width * 3;
61.             bytes_per_mcu[0] = 3*8;
62.             break;
```

```

62.         break;
63.
64.         case TINYJPEG_FMT_GREY:
65.             decode_mcu_table = decode_mcu_lcomp_table;
66.             colorspace_array_conv = convert_colorspace_grey;
67.             if (priv->components[0] == NULL)
68.                 priv->components[0] = (uint8_t *)malloc(priv->width * priv->height);
69.             bytes_per_blocklines[0] = priv->width;
70.             bytes_per_mcu[0] = 8;
71.             break;
72.
73.         default:
74. #if TRACE_PARAM
75.             fprintf(param_trace, "Bad pixel format\n");
76.             fflush(param_trace);
77. #endif
78.         return -1;
79.     }
80.
81.     xstride_by_mcu = ystride_by_mcu = 8;
82.     if ((priv->component_infos[cY].Hfactor | priv->component_infos[cY].Vfactor) == 1) {
83.         decode_MCU = decode_mcu_table[0];
84.         convert_to_pixfmt = colorspace_array_conv[0];
85. #if TRACE_PARAM
86.         fprintf(param_trace, "Use decode 1x1 sampling\n");
87.         fflush(param_trace);
88. #endif
89.     } else if (priv->component_infos[cY].Hfactor == 1) {
90.         decode_MCU = decode_mcu_table[1];
91.         convert_to_pixfmt = colorspace_array_conv[1];
92.         ystride_by_mcu = 16;
93. #if TRACE_PARAM
94.         fprintf(param_trace, "Use decode 1x2 sampling (not supported)\n");
95.         fflush(param_trace);
96. #endif
97.     } else if (priv->component_infos[cY].Vfactor == 2) {
98.         decode_MCU = decode_mcu_table[3];
99.         convert_to_pixfmt = colorspace_array_conv[3];
100.         xstride_by_mcu = 16;
101.         ystride_by_mcu = 16;
102. #if TRACE_PARAM
103.         fprintf(param_trace, "Use decode 2x2 sampling\n");
104.         fflush(param_trace);
105. #endif
106.     } else {
107.         decode_MCU = decode_mcu_table[2];
108.         convert_to_pixfmt = colorspace_array_conv[2];
109.         xstride_by_mcu = 16;
110. #if TRACE_PARAM
111.         fprintf(param_trace, "Use decode 2x1 sampling\n");
112.         fflush(param_trace);
113. #endif
114.     }
115.
116.     resync(priv);
117.
118.     /* Don't forget to that block can be either 8 or 16 lines */
119.     bytes_per_blocklines[0] *= ystride_by_mcu;
120.     bytes_per_blocklines[1] *= ystride_by_mcu;
121.     bytes_per_blocklines[2] *= ystride_by_mcu;
122.
123.     bytes_per_mcu[0] *= xstride_by_mcu/8;
124.     bytes_per_mcu[1] *= xstride_by_mcu/8;
125.     bytes_per_mcu[2] *= xstride_by_mcu/8;
126.
127.     /* Just the decode the image by macroblock (size is 8x8, 8x16, or 16x16) */
128.     //纵向
129.     for (y=0; y < priv->height/ystride_by_mcu; y++)
130.     {
131.         //trace("Decoding row %d\n", y);
132.         priv->plane[0] = priv->components[0] + (y * bytes_per_blocklines[0]);
133.         priv->plane[1] = priv->components[1] + (y * bytes_per_blocklines[1]);
134.         priv->plane[2] = priv->components[2] + (y * bytes_per_blocklines[2]);
135.         //横向 (循环的写法还不一样?)
136.         for (x=0; x < priv->width; x+=xstride_by_mcu)
137.         {
138.             decode_MCU(priv);
139.             convert_to_pixfmt(priv);
140.
141.             //DCT系数-----
142.             //temp=(char *)priv->component_infos->DCT;
143.             //if(y==4&&x==xstride_by_mcu*3){
144.             if(priv->dlog->m_vijpgoutputdct.GetCheck()==1){
145.                 fp = fopen("DCT系数表.txt", "a+");
146.                 //fwrite(temp,64,1,fp);
147.                 fprintf(fp, "第%d行, 第%d列\n", y, x/xstride_by_mcu);
148.                 for(j=0; j<64; j++){
149.                     fprintf(fp, "%d ", priv->component_infos[cY].DCT[j]);
150.                 }
151.                 fprintf(fp, "\n");
152.                 fclose(fp);
153.             }

```

```

154. #if TRACE_PARAM
155.     fprintf(param_trace, "\n第3行, 第4列\n");
156.     for(j=0; j<8; j++){
157.         for(k=0; k<8; k++){
158.             fprintf(param_trace, "%d ", priv->component_infos[cY].DCT[j*8+k]);
159.         }
160.         fprintf(param_trace, "\n");
161.     }
162.     fprintf(fp, "\n-----\n");
163.     fflush(param_trace);
164. #endif
165.     //}
166.
167.     //解码后系数 (Y) -----
168.     //temp=(char *)priv->Y;
169.     //if(y==4&&x==xstride_by_mcu*3){
170.     if(priv->dlg->m_vijpgoutputy.GetCheck()==1){
171.         fp = fopen("解码后Y系数表.txt", "a+");
172.         //fwrite(temp, 64*4, 1, fp);
173.         fprintf(fp, "第%d行, 第%d列\n", y, x/xstride_by_mcu);
174.         for(j=0; j<64*4; j++){
175.             fprintf(fp, "%d ", priv->Y[j]);
176.         }
177.         fprintf(fp, "\n");
178.         fclose(fp);
179.     }
180. #if TRACE_PARAM
181.     fprintf(param_trace, "第3行, 第4列\n");
182.     for(j=0; j<8; j++){
183.         for(k=0; k<8; k++){
184.             fprintf(param_trace, "%d ", priv->Y[j*8+k]);
185.         }
186.         fprintf(param_trace, "\n");
187.     }
188.     fprintf(fp, "\n-----\n");
189.     fflush(param_trace);
190. #endif
191.     //}
192.
193.     //-----
194.     priv->plane[0] += bytes_per_mcu[0];
195.     priv->plane[1] += bytes_per_mcu[1];
196.     priv->plane[2] += bytes_per_mcu[2];
197.
198.     if (priv->restarts_to_go>0)
199.     {
200.         priv->restarts_to_go--;
201.         if (priv->restarts_to_go == 0)
202.         {
203.             priv->stream -= (priv->nbits_in_reservoir/8);
204.             resync(priv);
205.             if (find_next_rst_marker(priv) < 0)
206.                 return -1;
207.         }
208.     }
209. }
210. }
211. #if TRACE_PARAM
212.     fprintf(param_trace, "Input file size: %d\n", priv->stream_length+2);
213.     fprintf(param_trace, "Input bytes actually read: %d\n", priv->stream - priv->stream_begin + 2);
214.     fflush(param_trace);
215. #endif
216.
217.     return 0;
218. }

```

主页：[http://www.saillard.org/programs\\_and\\_patches/tinyjpegdecoder/](http://www.saillard.org/programs_and_patches/tinyjpegdecoder/)

源代码下载：<http://download.csdn.net/detail/leixiaohua1020/6383115>

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/12618335>

文章标签：[jpeg](#) [解码](#) [源代码](#) [tinyjpeg](#)

个人分类：[TinyJPEG](#)

所属专栏：[开源多媒体项目源代码分析](#)

此PDF由spyyg生成, 请尊重原作者版权!!!

我的邮箱: liushidc@163.com