

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

2013年10月11日 15:48:08 阅读数：10057

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。

真正的解码开始于 `convert_one_image()` 函数：

```

1.  /**
2.   * Load one jpeg image, and decompress it, and save the result.
3.   */
4.   int convert_one_image(LPVOID lparam, const char *infilename, const char *outfilename, int output_format)
5.   {
6.       FILE *fp;
7.       unsigned int length_of_file;
8.       unsigned int width, height;
9.       unsigned char *buf;
10.      struct jpeg_private *jdec;
11.      unsigned char *components[3];
12.
13.      /* Load the Jpeg into memory */
14.      fp = fopen(infilename, "rb");
15.      if (fp == NULL)
16.          exitmessage("Cannot open filename\n");
17.      length_of_file = filesize(fp);
18.      buf = (unsigned char *)malloc(length_of_file + 4);
19.      if (buf == NULL)
20.          exitmessage("Not enough memory for loading file\n");
21.      fread(buf, length_of_file, 1, fp);
22.      fclose(fp);
23.
24.      /* Decompress it */
25.      //分配内存
26.      jdec = tinyjpeg_init();
27.      //传入句柄-----
28.      jdec->dlg=(CSpecialVIJPGDlg *)lparam;
29.
30.      if (jdec == NULL)
31.          exitmessage("Not enough memory to alloc the structure need for decompressing\n");
32.      //解头部
33.      if (tinyjpeg_parse_header(jdec, buf, length_of_file)<0)
34.          exitmessage(tinyjpeg_get_errorstring(jdec));
35.      /* Get the size of the image */
36.      //获得图像长宽
37.      tinyjpeg_get_size(jdec, &width, &height);
38.
39.      snprintf(error_string, sizeof(error_string), "Decoding JPEG image...\n");
40.      //解码实际数据
41.      if (tinyjpeg_decode(jdec, output_format) < 0)
42.          exitmessage(tinyjpeg_get_errorstring(jdec));
43.      /*
44.       * Get address for each plane (not only max 3 planes is supported), and
45.       * depending of the output mode, only some components will be filled
46.       * RGB: 1 plane, YUV420P: 3 planes, GREY: 1 plane
47.       */
48.      tinyjpeg_get_components(jdec, components);
49.
50.      /* Save it */
51.      switch (output_format)
52.      {
53.          case TINYJPEG_FMT_RGB24:
54.          case TINYJPEG_FMT_BGR24:
55.              write_tga(outfilename, output_format, width, height, components);
56.              break;
57.          case TINYJPEG_FMT_YUV420P:
58.              //开始写入成YUV420P文件
59.              write_yuv(outfilename, width, height, components);
60.              break;
61.          case TINYJPEG_FMT_GREY:
62.              //开始写入成灰度文件
63.              write_pgm(outfilename, width, height, components);
64.              break;
65.      }
66.
67.      /* Only called this if the buffers were allocated by tinyjpeg_decode() */
68.      //modify by lei! tinyjpeg_free(jdec);
69.      /* else called just free(jdec); */
70.
71.      free(buf);
72.      return 0;
73.  }

```

tinyjpeg_init() 用于初始化：


```

35. //Define quantization table
36. case DQT:
37. //开始解析DQT
38. if (parse_DQT(priv, stream) < 0)
39. return -1;
40. break;
41. case SOS:
42. //开始解析SOS
43. if (parse_SOS(priv, stream) < 0)
44. return -1;
45. sos_marker_found = 1;
46. break;
47. //Define Huffman table
48. case DHT:
49. //开始解析DHT
50. if (parse_DHT(priv, stream) < 0)
51. return -1;
52. dht_marker_found = 1;
53. break;
54. case DRI:
55. //开始解析DRI
56. if (parse_DRI(priv, stream) < 0)
57. return -1;
58. break;
59. default:
60. #if TRACE_PARAM
61. fprintf(param_trace, "> Unknown marker %2.2x\n", marker);
62. fflush(param_trace);
63. #endif
64. break;
65. }
66. //解下一个segment
67. stream = next_chunck;
68. }
69.
70. if (!dht_marker_found) {
71. #if TRACE_PARAM
72. fprintf(param_trace, "No Huffman table loaded, using the default one\n");
73. fflush(param_trace);
74. #endif
75. build_default_huffman_tables(priv);
76. }
77.
78. #ifdef SANITY_CHECK
79. if ( (priv->component_infos[cY].Hfactor < priv->component_infos[cCb].Hfactor)
80. || (priv->component_infos[cY].Hfactor < priv->component_infos[cCr].Hfactor))
81. snprintf(error_string, sizeof(error_string), "Horizontal sampling factor for Y should be greater than horizontal sampling factor for Cb or Cr\n");
82. if ( (priv->component_infos[cY].Vfactor < priv->component_infos[cCb].Vfactor)
83. || (priv->component_infos[cY].Vfactor < priv->component_infos[cCr].Vfactor))
84. snprintf(error_string, sizeof(error_string), "Vertical sampling factor for Y should be greater than vertical sampling factor for Cb or Cr\n");
85. if ( (priv->component_infos[cCb].Hfactor!=1)
86. || (priv->component_infos[cCr].Hfactor!=1)
87. || (priv->component_infos[cCb].Vfactor!=1)
88. || (priv->component_infos[cCr].Vfactor!=1))
89. snprintf(error_string, sizeof(error_string), "Sampling other than 1x1 for Cr and Cb is not supported");
90. #endif
91.
92. return 0;
93. bogus_jpeg_format:
94. #if TRACE_PARAM
95. fprintf(param_trace, "Bogus jpeg format\n");
96. fflush(param_trace);
97. #endif
98. return -1;
99. }

```

parse_SOF() 用于解析SOF标签：

注意：其中包含了部分自己写的代码，形如：

```

1. itoa(width, temp_str1, 10);
2. priv->dlg->AppendBInfo("SOF0", "宽", temp_str1, "图像的宽度");

```

这些代码主要用于在解码过程中提取一些信息，比如图像宽，高，颜色分量数等等

```

1. static int parse_SOF(struct jdec_private *priv, const unsigned char *stream)
2. {
3.     int i, width, height, nr_components, cid, sampling_factor;
4.     int Q_table;
5.     struct component *c;
6.     #if TRACE_PARAM
7.         fprintf(param_trace, "> SOF marker\n");
8.         fflush(param_trace);
9.     #endif
10.    print_SOF(stream);
11.
12.    height = be16_to_cpu(stream+3);
13.    width = be16_to_cpu(stream+5);
14.    nr_components = stream[7];
15.    #if SANITY_CHECK
16.        if (stream[2] != 8)
17.            snprintf(error_string, sizeof(error_string), "Precision other than 8 is not supported\n");
18.        if (width>JPEG_MAX_WIDTH || height>JPEG_MAX_HEIGHT)
19.            snprintf(error_string, sizeof(error_string), "Width and Height (%dx%d) seems suspicious\n", width, height);
20.        if (nr_components != 3)
21.            snprintf(error_string, sizeof(error_string), "We only support YUV images\n");
22.        if (height%16)
23.            snprintf(error_string, sizeof(error_string), "Height need to be a multiple of 16 (current height is %d)\n", height);
24.        if (width%16)
25.            snprintf(error_string, sizeof(error_string), "Width need to be a multiple of 16 (current Width is %d)\n", width);
26.    #endif
27.    char temp_str1[MAX_URL_LENGTH]={0};
28.    itoa(width, temp_str1, 10);
29.    priv->dlg->AppendBInfo("SOF0", "宽", temp_str1, "图像的宽度");
30.    itoa(height, temp_str1, 10);
31.    priv->dlg->AppendBInfo("SOF0", "高", temp_str1, "图像的高度");
32.    itoa(nr_components, temp_str1, 10);
33.    priv->dlg->AppendBInfo("SOF0", "颜色分量数", temp_str1, "图像的颜色分量数。一个字节，例如03，代表有三个分量，YCrCb");
34.    itoa(stream[2], temp_str1, 10);
35.    priv->dlg->AppendBInfo("SOF0", "精度", temp_str1, "图像的精度。一个字节，例如08，即精度为一个字节。");
36.    stream += 8;
37.    for (i=0; i<nr_components; i++) {
38.        cid = *stream++;
39.        sampling_factor = *stream++;
40.        Q_table = *stream++;
41.        c = &priv->component_infos[i];
42.        #if SANITY_CHECK
43.            c->cid = cid;
44.            if (Q_table >= COMPONENTS)
45.                snprintf(error_string, sizeof(error_string), "Bad Quantization table index (got %d, max allowed %d)\n", Q_table, COMPONENTS-1);
46.        #endif
47.        c->Vfactor = sampling_factor&0xf;
48.        c->Hfactor = sampling_factor>>4;
49.        c->Q_table = priv->Q_tables[Q_table];
50.        //-----
51.        char temp_str2[MAX_URL_LENGTH]={0};
52.        sprintf(temp_str2, "垂直采样因子 [%d] ", i);
53.        itoa(c->Hfactor, temp_str1, 10);
54.        priv->dlg->AppendBInfo("SOF0", temp_str2, temp_str1, "颜色分量信息：每个分量有三个字节，第一个为分量的ID，01：Y 02：U 03：V；第二个字节高位为水平采样因子，低位为垂直采样因子。");
55.        sprintf(temp_str2, "水平采样因子 [%d] ", i);
56.        itoa(c->Hfactor, temp_str1, 10);
57.        priv->dlg->AppendBInfo("SOF0", temp_str2, temp_str1, "颜色分量信息：每个分量有三个字节，第一个为分量的ID，01：Y 02：U 03：V；第二个字节高位为水平采样因子，低位为垂直采样因子。");
58.        sprintf(temp_str2, "对应量化表ID [%d] ", i);
59.        itoa((int)Q_table, temp_str1, 10);
60.        priv->dlg->AppendBInfo("SOF0", temp_str2, temp_str1, "颜色分量信息：第三个字节代表这个分量对应的量化表ID，例如，Y对应的量化表ID索引值为00，而UV对应的量化表ID都为01，即它们共用一张量化表。");
61.        //-----
62.        #if TRACE_PARAM
63.            fprintf(param_trace, "Component:%d factor:%dx%d Quantization table:%d\n",
64.                cid, c->Hfactor, c->Hfactor, Q_table );
65.            fflush(param_trace);
66.        #endif
67.    }
68.
69.
70.
71.    priv->width = width;
72.    priv->height = height;
73.    #if TRACE_PARAM
74.        fprintf(param_trace, "< SOF marker\n");
75.        fflush(param_trace);
76.    #endif
77.    return 0;
78.
79. }

```

parse_DHT() 用于解析DHT标签：

```

[cpp]

```

```

1. //解析DHT表
2. static int parse_DHT(struct jdec_private *priv, const unsigned char *stream)
3. {
4.     unsigned int count, i,j;
5.     unsigned char huff_bits[17];
6.     int length, index;
7.     //-----
8.     char *temp;
9.     FILE *fp;
10.    //-----
11.    length = be16_to_cpu(stream) - 2;
12.    //跳过length字段
13.    stream += 2; /* Skip length */
14.    #if TRACE_PARAM
15.        fprintf(param_trace, "> DHT marker (length=%d)\n", length);
16.        fflush(param_trace);
17.    #endif
18.
19.    while (length>0) {
20.        //跳过第1字节:
21.        //Huffman 表ID号和类型, 高 4 位为表的类型, 0:DC 直流;1:AC交流
22.        //低四位为 Huffman 表 ID。
23.        index = *stream++;
24.
25.        /* We need to calculate the number of bytes 'vals' will takes */
26.        huff_bits[0] = 0;
27.        count = 0;
28.
29.        //不同长度 Huffman 的码字数量: 固定为 16 个字节, 每个字节代表从长度为 1到长度为 16 的码字的个数
30.        for (i=1; i<17; i++) {
31.            huff_bits[i] = *stream++;
32.            //count记录码字的个数
33.            count += huff_bits[i];
34.        }
35.        #if SANITY_CHECK
36.            if (count >= HUFFMAN_BITS_SIZE)
37.                snprintf(error_string, sizeof(error_string), "No more than %d bytes is allowed to describe a huffman table", HUFFMAN_BITS_SIZE);
38.        #endif
39.        if ( (index & 0xf) >= HUFFMAN_TABLES)
40.            snprintf(error_string, sizeof(error_string), "No more than %d Huffman tables is supported (got %d)\n", HUFFMAN_TABLES, index&0xf);
41.        #if TRACE_PARAM
42.            fprintf(param_trace, "Huffman table %s[%d] length=%d\n", (index&0xf0?"AC":"DC", index&0xf, count);
43.            fflush(param_trace);
44.        #endif
45.
46.        if (index & 0xf0 ){
47.            //-----
48.            char temp_str1[MAX_URL_LENGTH]={0};
49.            char temp_str2[MAX_URL_LENGTH]={0};
50.            temp=(char *)stream;
51.            //fp = fopen("DHT.txt", "a+");
52.            //fwrite(temp, 16, 1, fp);
53.            for(j=0;j<16;j++){
54.                //fprintf(fp, "%d ", temp[j]);
55.                sprintf(temp_str2, "%d ", temp[j]);
56.                strcat(temp_str1, temp_str2);
57.            }
58.            //fprintf(fp, "\n-----\n");
59.            //fclose(fp);
60.            //-----
61.            priv->dlg->AppendBInfo("DHT", "定义霍夫曼表【交流系数表】", temp_str1, "Huffman表ID号和类型: 1字节, 高4位为表的类型, 0:DC直流; 1:AC交流  
可以看出这里是直流表; 低四位为Huffman表ID");
62.            //-----
63.            //交流霍夫曼表
64.            build_huffman_table(huff_bits, stream, &priv->HTAC[index&0xf]);
65.        }
66.        else{
67.            //-----
68.            char temp_str1[MAX_URL_LENGTH]={0};
69.            char temp_str2[MAX_URL_LENGTH]={0};
70.            temp=(char *)stream;
71.            //fp = fopen("DHT.txt", "a+");
72.            //fwrite(temp, 16, 1, fp);
73.            for(j=0;j<16;j++){
74.                //fprintf(fp, "%d ", temp[j]);
75.                sprintf(temp_str2, "%d ", temp[j]);
76.                strcat(temp_str1, temp_str2);
77.            }
78.            //fprintf(fp, "\n-----\n");
79.            //fclose(fp);
80.            //-----
81.            priv->dlg->AppendBInfo("DHT", "定义霍夫曼表【直流系数表】", temp_str1, "Huffman表ID号和类型: 1字节, 高4位为表的类型, 0:DC直流; 1:AC交流  
可以看出这里是直流表; 低四位为Huffman表ID");
82.            //-----
83.            //直流霍夫曼表
84.            build_huffman_table(huff_bits, stream, &priv->HTDC[index&0xf]);
85.        }
86.
87.        length -= 1;
88.        length -= 16;

```

```
88.         length -= 10;  
89.         length -= count;  
90.         stream += count;  
91.     }  
92.     #if TRACE_PARAM  
93.         fprintf(param_trace, "< DHT marker\n");  
94.         fflush(param_trace);  
95.     #endif  
96.     return 0;  
97. }
```

parse_DQT() 用于解析DQT标签：

```

1. //解析Define quantization table
2. static int parse_DQT(struct jdec_private *priv, const unsigned char *stream)
3. {
4.     int qi;
5.     float *table;
6.     const unsigned char *dqt_block_end;
7.
8.     //-----
9.     int j,k;
10.    char *temp;
11.    FILE *fp;
12.    //-----
13.    #if TRACE_PARAM
14.        fprintf(param_trace, "> DQT marker\n");
15.        fflush(param_trace);
16.    #endif
17.    //该Segment末尾
18.    dqt_block_end = stream + be16_to_cpu(stream);
19.    //跳过标签length (2字节)
20.    stream += 2; /* Skip length */
21.    //没到末尾
22.    while (stream < dqt_block_end)
23.    {
24.        //跳过该Segment的第1个字节,QT信息
25.        //precision: 00 (Higher 4 bit)
26.        //index: 00 (Lower 4 bit)
27.        //qi取1, 第1张量化表 (例如, 亮度表), 取2, 第2张量化表 (例如, 色度表)
28.        qi = *stream++;
29.        #if SANITY_CHECK
30.            if (qi>4)
31.                snprintf(error_string, sizeof(error_string), "16 bits quantization table is not supported\n");
32.            if (qi>4)
33.                snprintf(error_string, sizeof(error_string), "No more 4 quantization table is supported (got %d)\n", qi);
34.        #endif
35.        //table指向jdec_private的Q_tables数组, 为了在其中写入数据
36.        table = priv->Q_tables[qi];
37.        //注意: 一次搞定整张! 写入
38.        //需要对数值进行变换!  $\cos(k*PI/16) * \sqrt{2}$ 
39.        //这样才能得到离散余弦变换的系数
40.        build_quantization_table(table, stream);
41.        //-----
42.
43.        temp=(char *)stream;
44.        //fp = fopen("DQT.txt", "a+");
45.        //fwrite(temp, 64, 1, fp);
46.        char temp_str1[MAX_URL_LENGTH]={0};
47.        char temp_str2[MAX_URL_LENGTH]={0};
48.        for(j=0;j<64;j++){
49.            sprintf(temp_str2,"%d ",temp[j]);
50.            strcat(temp_str1,temp_str2);
51.            //fprintf(fp,"%d ",temp[j]);
52.        }
53.        //计数
54.        char temp_str3[MAX_URL_LENGTH]={0};
55.        sprintf(temp_str3,"量化表 [%d] ",priv->DQT_table_num);
56.        priv->dlg->AppendBInfo("DQT",temp_str3,temp_str1,"JPEG格式文件的量化表, 一般来说第一张是亮度的, 后面是色度的");
57.        priv->DQT_table_num++;
58.        //fprintf(fp,"\n-----\n");
59.        //fclose(fp);
60.        #if TRACE_PARAM
61.            for(j=0;j<8;j++){
62.                for(k=0;k<8;k++){
63.                    fprintf(param_trace,"%d ",temp[j*8+k]);
64.                }
65.                fprintf(param_trace,"\n");
66.            }
67.            fprintf(fp,"\n-----\n");
68.            fflush(param_trace);
69.        #endif
70.        //-----
71.        //完事了!
72.        stream += 64;
73.    }
74.    #if TRACE_PARAM
75.        fprintf(param_trace, "< DQT marker\n");
76.        fflush(param_trace);
77.    #endif
78.    return 0;
79. }

```

其他标签的解析不一一列举。

待续未完。。。

主页：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/12617079>

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[spygg](#)生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com