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]

```cpp
/**
 * Decode and convert the jpeg image into @pixfmt@ image
 *解码函数
 * Note: components will be automaticaly allocated if no memory is attached.
 */
int tinyjpeg_decode(struct jdec_private *priv, int pixfmt)
{
  unsigned int x, y, xstride_by_mcu, ystride_by_mcu;
  unsigned int bytes_per_blocklines[3], bytes_per_mcu[3];
  decode_MCU_fct decode_MCU;
  const decode_MCU_fct *decode_mcu_table;
  const convert_colorspace_fct *colorspace_array_conv;
  convert_colorspace_fct convert_to_pixfmt;

  //-----------------------------------------
    FILE *fp;
    char *temp;
    int j,k;
  //-----------------------------------------

  if (setjmp(priv->jump_state))
    return -1;

  /* To keep gcc happy initialize some array */
  bytes_per_mcu[1] = 0;
  bytes_per_mcu[2] = 0;
  bytes_per_blocklines[1] = 0;
  bytes_per_blocklines[2] = 0;

  decode_mcu_table = decode_mcu_3comp_table;
  switch (pixfmt) {
     case TINYJPEG_FMT_YUV420P:
       colorspace_array_conv = convert_colorspace_yuv420p;
       if (priv->components[0] == NULL)
     priv->components[0] = (uint8_t *)malloc(priv->width * priv->height);
       if (priv->components[1] == NULL)
     priv->components[1] = (uint8_t *)malloc(priv->width * priv->height/4);
       if (priv->components[2] == NULL)
     priv->components[2] = (uint8_t *)malloc(priv->width * priv->height/4);
       bytes_per_blocklines[0] = priv->width;
       bytes_per_blocklines[1] = priv->width/4;
       bytes_per_blocklines[2] = priv->width/4;
       bytes_per_mcu[0] = 8;
       bytes_per_mcu[1] = 4;
       bytes_per_mcu[2] = 4;
       break;

     case TINYJPEG_FMT_RGB24:
       colorspace_array_conv = convert_colorspace_rgb24;
       if (priv->components[0] == NULL)
     priv->components[0] = (uint8_t *)malloc(priv->width * priv->height * 3);
       bytes_per_blocklines[0] = priv->width * 3;
       bytes_per_mcu[0] = 3*8;
       break;

     case TINYJPEG_FMT_BGR24:
       colorspace_array_conv = convert_colorspace_bgr24;
       if (priv->components[0] == NULL)
     priv->components[0] = (uint8_t *)malloc(priv->width * priv->height * 3);
       bytes_per_blocklines[0] = priv->width * 3;
       bytes_per_mcu[0] = 3*8;
       break;
```

```
        break;

      case TINYJPEG_FMT_GREY:
        decode_mcu_table = decode_mcu_1comp_table;
        colorspace_array_conv = convert_colorspace_grey;
        if (priv->components[0] == NULL)
      priv->components[0] = (uint8_t *)malloc(priv->width * priv->height);
        bytes_per_blocklines[0] = priv->width;
        bytes_per_mcu[0] = 8;
        break;

      default:
#if TRACE_PARAM
          fprintf(param_trace,"Bad pixel format\n");
          fflush(param_trace);
#endif
        return -1;
  }

  xstride_by_mcu = ystride_by_mcu = 8;
  if ((priv->component_infos[cY].Hfactor | priv->component_infos[cY].Vfactor) == 1) {
      decode_MCU = decode_mcu_table[0];
      convert_to_pixfmt = colorspace_array_conv[0];
#if TRACE_PARAM
      fprintf(param_trace,"Use decode 1x1 sampling\n");
      fflush(param_trace);
#endif
  } else if (priv->component_infos[cY].Hfactor == 1) {
      decode_MCU = decode_mcu_table[1];
      convert_to_pixfmt = colorspace_array_conv[1];
      ystride_by_mcu = 16;
#if TRACE_PARAM
      fprintf(param_trace,"Use decode 1x2 sampling (not supported)\n");
      fflush(param_trace);
#endif
  } else if (priv->component_infos[cY].Vfactor == 2) {
      decode_MCU = decode_mcu_table[3];
      convert_to_pixfmt = colorspace_array_conv[3];
      xstride_by_mcu = 16;
      ystride_by_mcu = 16;
#if TRACE_PARAM
      fprintf(param_trace,"Use decode 2x2 sampling\n");
      fflush(param_trace);
#endif
  } else {
      decode_MCU = decode_mcu_table[2];
      convert_to_pixfmt = colorspace_array_conv[2];
      xstride_by_mcu = 16;
#if TRACE_PARAM
      fprintf(param_trace,"Use decode 2x1 sampling\n");
      fflush(param_trace);
#endif
  }

  resync(priv);

  /* Don't forget to that block can be either 8 or 16 lines */
  bytes_per_blocklines[0] *= ystride_by_mcu;
  bytes_per_blocklines[1] *= ystride_by_mcu;
  bytes_per_blocklines[2] *= ystride_by_mcu;

  bytes_per_mcu[0] *= xstride_by_mcu/8;
  bytes_per_mcu[1] *= xstride_by_mcu/8;
  bytes_per_mcu[2] *= xstride_by_mcu/8;

  /* Just the decode the image by macroblock (size is 8x8, 8x16, or 16x16) */
  //纵向
  for (y=0; y < priv->height/ystride_by_mcu; y++)
   {
     //trace("Decoding row %d\n", y);
     priv->plane[0] = priv->components[0] + (y * bytes_per_blocklines[0]);
     priv->plane[1] = priv->components[1] + (y * bytes_per_blocklines[1]);
     priv->plane[2] = priv->components[2] + (y * bytes_per_blocklines[2]);
     //横向（循环的写法还不一样？）
     for (x=0; x < priv->width; x+=xstride_by_mcu)
      {
        decode_MCU(priv);
        convert_to_pixfmt(priv);

//DCT系数-------------------------------------------------------
        //temp=(char *)priv->component_infos->DCT;
        //if(y==4&&x==xstride_by_mcu*3){
        if(priv->dlg->m_vijpgoutputdct.GetCheck()==1){
            fp = fopen("DCT系数表.txt", "a+");
            //fwrite(temp,64,1,fp);
            fprintf(fp,"第%d行，第%d列\n",y,x/xstride_by_mcu);
            for(j=0;j<64;j++){
            fprintf(fp,"%d ",priv->component_infos[cY].DCT[j]);
            }
            fprintf(fp,"\n");
            fclose(fp);
        }
```

```
#if TRACE_PARAM
        fprintf(param_trace,"\n第3行，第4列\n");
        for(j=0;j<8;j++){
            for(k=0;k<8;k++){
                fprintf(param_trace,"%d ",priv->component_infos[cY].DCT[j*8+k]);
            }
            fprintf(param_trace,"\n");
        }
        fprintf(fp,"\n----------------------\n");
        fflush(param_trace);
#endif
        //}

//解码后系数（Y）-----------------------------------------------
        //temp=(char *)priv->Y;
        //if(y==4&&x==xstride_by_mcu*3){
        if(priv->dlg->m_vijpgoutputy.GetCheck()==1){
            fp = fopen("解码后Y系数表.txt", "a+");
            //fwrite(temp,64*4,1,fp);
            fprintf(fp,"第%d行，第%d列\n",y,x/xstride_by_mcu);
            for(j=0;j<64*4;j++){
            fprintf(fp,"%d ",priv->Y[j]);
            }
            fprintf(fp,"\n");
            fclose(fp);
        }
#if TRACE_PARAM
        fprintf(param_trace,"第3行，第4列\n");
        for(j=0;j<8;j++){
            for(k=0;k<8;k++){
                fprintf(param_trace,"%d ",priv->Y[j*8+k]);
            }
            fprintf(param_trace,"\n");
        }
        fprintf(fp,"\n----------------------\n");
        fflush(param_trace);
#endif
        //}

//-------------------------------------------------------------
        priv->plane[0] += bytes_per_mcu[0];
        priv->plane[1] += bytes_per_mcu[1];
        priv->plane[2] += bytes_per_mcu[2];

        if (priv->restarts_to_go>0)
         {
           priv->restarts_to_go--;
           if (priv->restarts_to_go == 0)
            {
              priv->stream -= (priv->nbits_in_reservoir/8);
              resync(priv);
              if (find_next_rst_marker(priv) < 0)
            return -1;
            }
         }
      }
    }
#if TRACE_PARAM
  fprintf(param_trace,"Input file size: %d\n", priv->stream_length+2);
  fprintf(param_trace,"Input bytes actually read: %d\n", priv->stream - priv->stream_begin + 2);
  fflush(param_trace);
#endif

  return 0;
}
```

主页： http://www.saillard.org/programs_and_patches/tinyjpegdecoder/

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

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