

## 最简单的视频编码器：基于libx265（编码YUV为H.265）

2014年12月23日 17:36:43 阅读数：13739

最简单的视频编码器系列文章列表：

[最简单的视频编码器：编译](#)

[最简单的视频编码器：基于libx264（编码YUV为H.264）](#)

[最简单的视频编码器：基于libx265（编码YUV为H.265）](#)

[最简单的视频编码器：libvpx（编码YUV为VP8）](#)

本文记录一个最简单的基于libx265的H.265(HEVC)视频编码器。此前记录的编码器是通过FFmpeg调用libx265完成编码的，例如：

[《最简单的基于FFmpeg的视频编码器-更新版（YUV编码为HEVC\(H.265\)）》](#)

相比与上文中的编码器，本文记录的编码器属于“轻量级”的编码器。因为它不再包含FFmpeg的代码，直接调用libx265完成编码。因此项目的体积非常小巧。该编码器可以将输入的YUV数据编码为H.265码流文件。

## 流程图

调用libx265进行视频编码的流程图如下所示。

从流程图中可以看出x265的API和x264的API十分相似。它们在用法上只有微小的不同。

流程图中主要的函数如下所示。

x265\_param\_alloc()：为参数集结构体x265\_param分配内存。  
x265\_param\_default()：设置参数集结构体x265\_param的缺省值。  
x265\_picture\_alloc()：为图像结构体x265\_picture分配内存。  
x265\_picture\_init()：设置图像结构体x265\_picture的缺省值。  
x265\_encoder\_open()：打开编码器。  
x265\_encoder\_encode()：编码一帧图像。  
x265\_encoder\_close()：关闭编码器。  
x265\_picture\_free()：释放x265\_picture\_alloc()申请的资源。  
x265\_param\_free()：释放x265\_param\_alloc()申请的资源。

存储数据的结构体如下所示。

x265\_picture：存储压缩编码前的像素数据。  
x265\_nal：存储压缩编码后的码流数据。

此外流程图中还包括一个“flush\_encoder”模块，该模块使用的函数和编码模块是一样的。唯一的不同在于不再输入视频像素数据。它的作用是输出编码器中剩余的码流数据。

## 源代码

```
[cpp]
1.  /**
2.   * 最简单的基于X265的视频编码器
3.   * Simplest X265 Encoder
4.   *
5.   * 雷霄骅 Lei Xiaohua
6.   * leixiaohua1020@126.com
7.   * 中国传媒大学/数字电视技术
8.   * Communication University of China / Digital TV Technology
9.   * http://blog.csdn.net/leixiaohua1020
10.  *
11.  * 本程序可以YUV格式的像素数据编码为H.265码流，是最简单的
12.  * 基于libx265的视频编码器
13.  *
14.  * This software encode YUV data to H.265 bitstream.
15.  * It's the simplest encoder example based on libx265.
16.  */
17. #include <stdio.h>
18. #include <stdlib.h>
19.
20. #if defined ( __cplusplus )
21. extern "C"
```

```

22. {
23. #include "x265.h"
24. };
25. #else
26. #include "x265.h"
27. #endif
28.
29. int main(int argc, char** argv){
30.     int i,j;
31.     FILE *fp_src=NULL;
32.     FILE *fp_dst=NULL;
33.     int y_size;
34.     int buff_size;
35.     char *buff=NULL;
36.     int ret;
37.     x265_nal *pNals=NULL;
38.     uint32_t iNal=0;
39.
40.     x265_param* pParam=NULL;
41.     x265_encoder* pHandle=NULL;
42.     x265_picture *pPic_in=NULL;
43.
44.     //Encode 50 frame
45.     //if set 0, encode all frame
46.     int frame_num=50;
47.     int csp=X265_CSP_I420;
48.     int width=640,height=360;
49.
50.     fp_src=fopen("../cuc_ieschool_640x360_yuv420p.yuv","rb");
51.     //fp_src=fopen("../cuc_ieschool_640x360_yuv444p.yuv","rb");
52.
53.     fp_dst=fopen("cuc_ieschool.h265","wb");
54.     //Check
55.     if(fp_src==NULL||fp_dst==NULL){
56.         return -1;
57.     }
58.
59.     pParam=x265_param_alloc();
60.     x265_param_default(pParam);
61.     pParam->bRepeatHeaders=1;//write sps,pps before keyframe
62.     pParam->internalCsp=csp;
63.     pParam->sourceWidth=width;
64.     pParam->sourceHeight=height;
65.     pParam->fpsNum=25;
66.     pParam->fpsDenom=1;
67.     //Init
68.     pHandle=x265_encoder_open(pParam);
69.     if(pHandle==NULL){
70.         printf("x265_encoder_open err\n");
71.         return 0;
72.     }
73.     y_size = pParam->sourceWidth * pParam->sourceHeight;
74.
75.     pPic_in = x265_picture_alloc();
76.     x265_picture_init(pParam,pPic_in);
77.     switch(csp){
78.     case X265_CSP_I444:{
79.         buff=(char *)malloc(y_size*3);
80.         pPic_in->planes[0]=buff;
81.         pPic_in->planes[1]=buff+y_size;
82.         pPic_in->planes[2]=buff+y_size*2;
83.         pPic_in->stride[0]=width;
84.         pPic_in->stride[1]=width;
85.         pPic_in->stride[2]=width;
86.         break;
87.     }
88.     case X265_CSP_I420:{
89.         buff=(char *)malloc(y_size*3/2);
90.         pPic_in->planes[0]=buff;
91.         pPic_in->planes[1]=buff+y_size;
92.         pPic_in->planes[2]=buff+y_size*5/4;
93.         pPic_in->stride[0]=width;
94.         pPic_in->stride[1]=width/2;
95.         pPic_in->stride[2]=width/2;
96.         break;
97.     }
98.     default:{
99.         printf("Colorspace Not Support.\n");
100.        return -1;
101.    }
102. }
103.
104. //detect frame number
105. if(frame_num==0){
106.     fseek(fp_src,0,SEEK_END);
107.     switch(csp){
108.     case X265_CSP_I444:frame_num=ftell(fp_src)/(y_size*3);break;
109.     case X265_CSP_I420:frame_num=ftell(fp_src)/(y_size*3/2);break;
110.     default:printf("Colorspace Not Support.\n");return -1;
111.     }
112.     fseek(fp_src,0,SEEK_SET);
113. }

```

```

113.     }
114.
115.     //Loop to Encode
116.     for( i=0;i<frame_num;i++){
117.         switch(csp){
118.             case X265_CSP_I444:{
119.                 fread(pPic_in->planes[0],1,y_size,fp_src);    //Y
120.                 fread(pPic_in->planes[1],1,y_size,fp_src);    //U
121.                 fread(pPic_in->planes[2],1,y_size,fp_src);    //V
122.                 break;}
123.             case X265_CSP_I420:{
124.                 fread(pPic_in->planes[0],1,y_size,fp_src);    //Y
125.                 fread(pPic_in->planes[1],1,y_size/4,fp_src); //U
126.                 fread(pPic_in->planes[2],1,y_size/4,fp_src); //V
127.                 break;}
128.             default:{
129.                 printf("Colorspace Not Support.\n");
130.                 return -1;}
131.         }
132.
133.         ret=x265_encoder_encode(pHandle,&pNals,&iNal,pPic_in,NULL);
134.         printf("Succeed encode %5d frames\n",i);
135.
136.         for(j=0;j<iNal;j++){
137.             fwrite(pNals[j].payload,1,pNals[j].sizeBytes,fp_dst);
138.         }
139.     }
140.     //Flush Decoder
141.     while(1){
142.         ret=x265_encoder_encode(pHandle,&pNals,&iNal,NULL,NULL);
143.         if(ret==0){
144.             break;
145.         }
146.         printf("Flush 1 frame.\n");
147.
148.         for(j=0;j<iNal;j++){
149.             fwrite(pNals[j].payload,1,pNals[j].sizeBytes,fp_dst);
150.         }
151.     }
152.
153.     x265_encoder_close(pHandle);
154.     x265_picture_free(pPic_in);
155.     x265_param_free(pParam);
156.     free(buff);
157.     fclose(fp_src);
158.     fclose(fp_dst);
159.
160.     return 0;
161. }

```

## 运行结果

程序的输入为一个YUV文件（已经测试过YUV444P和YUV420P两种格式）。

□

输出为H.265码流文件。

□

H.265码流文件的信息如下所示。

□

## 下载

**Simplest Encoder**

**项目主页**

SourceForge：<https://sourceforge.net/projects/simplestencoder/>

Github：[https://github.com/leixiaohua1020/simplest\\_encoder](https://github.com/leixiaohua1020/simplest_encoder)

开源中国：[http://git.oschina.net/leixiaohua1020/simplest\\_encoder](http://git.oschina.net/leixiaohua1020/simplest_encoder)

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8284105>

该解决方案包含了几个常见的编码器的使用示例：

simplest\_vpx\_encoder：最简单的基于libvpx的视频编码器  
simplest\_x264\_encoder：最简单的基于libx264的视频编码器  
simplest\_x265\_encoder：最简单的基于libx265的视频编码器

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/42079101>

文章标签：[x265](#) [h265](#) [视频](#) [编码](#) [YUV](#)

个人分类：[我的开源项目](#) [x265](#)

此PDF由spygg生成, 请尊重原作者版权!!!  
我的邮箱:liushidc@163.com