昵称：azraelly
园龄：3年2个月
粉丝：37
关注：14
+加关注

< 2013年1月 >

日	一	二	三	四	五	六
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

随笔分类

D3D(1)
DDraw(3)
ffmpeg(7)
Linux编程(4)
编程问题(6)
网络编程(10)

随笔档案

2013年4月 (1)
2013年2月 (1)
2013年1月 (14)
2012年12月 (12)
2012年8月 (5)
2012年7月 (3)
2012年6月 (9)
2012年5月 (3)
2012年4月 (1)

最新评论

1. Re:GCC 编译详解
够用，大神膜拜。
--yeayo
2. Re:GCC 编译详解
学习了
--hitwh_Gypsy
3. Re:GCC 编译详解
@fengjieji引用只是编译不汇编，生成汇编代码是不是应该改成只是汇编不编译，生成汇编代码-c 只编译不汇编，应该改成：只汇编不连接。...
--Thomas2015
4. Re:ffmpeg ./configure参数说明
总结得相当不错，支持下。已转载并附链接by www.elesos.com 站长赠免费vpn【】访问youtube不卡，亲测可用，不限速。...
--www_elesos_com站长
5. Re:GCC 编译详解
不错！
--Ydoing

阅读排行榜

1. 图文详解YUV420数据格式(32748)
2. TCP的状态 (SYN, FIN, ACK, PSH, RS

图文详解YUV420数据格式

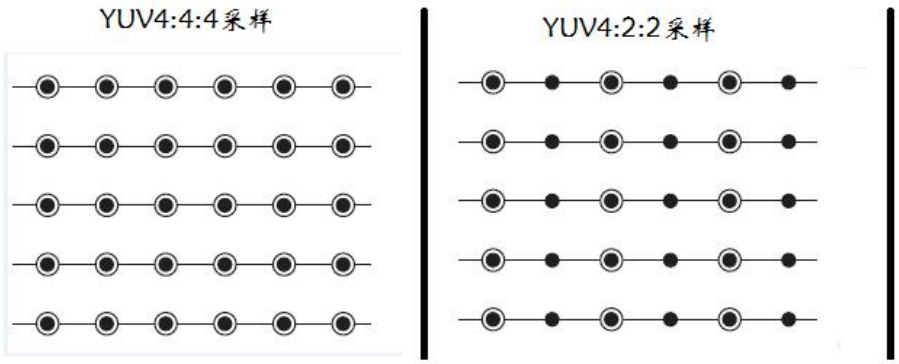
YUV格式有两大类：planar和packed。
对于planar的YUV格式，先连续存储所有像素点的Y，紧接着存储所有像素点的U，随后是所有像素点的V。
对于packed的YUV格式，每个像素点的Y,U,V是连续交*存储的。

YUV，分为三个分量，“Y”表示明亮度（Luminance或Luma），也就是灰度值；而“U”和“V” 表示的则是色度（Chrominance或Chroma），作用是描述影像色彩及饱和度，用于指定像素的颜色。

与我们熟知的RGB类似，YUV也是一种颜色编码方法，主要用于电视系统以及模拟视频领域，它将亮度信息（Y）与色彩信息（UV）分离，没有UV信息一样可以显示完整的图像，只不过是黑白的，这样的设计很好地解决了彩色电视机与黑白电视的兼容问题。并且，YUV不像RGB那样要求三个独立的视频信号同时传输，所以用YUV方式传送占用极少的频宽。

YUV码流的存储格式其实与其采样的方式密切相关，主流的采样方式有三种，YUV4:4:4，YUV4:2:2，YUV4:2:0，关于其详细原理，可以通过网上其它文章了解，这里我想强调的是如何根据其采样格式来从码流中还原每个像素点的YUV值，因为只有正确地还原了每个像素点的YUV值，才能通过YUV与RGB的转换公式提取出每个像素点的RGB值，然后显示出来。

用三个图来直观地表示采集的方式吧，以黑点表示采样该像素点的Y分量，以空心圆圈表示采用该像素点的UV分量。



先记住下面这段话，以后提取每个像素的YUV分量会用到。

1. YUV 4:4:4采样，每一个Y对应一组UV分量。
2. YUV 4:2:2采样，每两个Y共用一组UV分量。
3. YUV 4:2:0采样，每四个Y共用一组UV分量。

2. 存储方式

下面我用图的形式给出常见的YUV码流的存储方式，并在存储方式后面附有取样每个像素点的YUV数据的方法，其中，Cb、Cr的含义等同于U、V。

（1） YUVY 格式 （属于YUV422）

start + 0:	Y'_{00}	Cb_{00}	Y'_{01}	Cr_{00}	Y'_{02}	Cb_{01}	Y'_{03}	Cr_{01}
start + 8:	Y'_{10}	Cb_{10}	Y'_{11}	Cr_{10}	Y'_{12}	Cb_{11}	Y'_{13}	Cr_{11}
start + 16:	Y'_{20}	Cb_{20}	Y'_{21}	Cr_{20}	Y'_{22}	Cb_{21}	Y'_{23}	Cr_{21}
start + 24:	Y'_{30}	Cb_{30}	Y'_{31}	Cr_{30}	Y'_{32}	Cb_{31}	Y'_{33}	Cr_{31}

YUYV为YUV422采样的存储格式中的一种，相邻的两个Y共用其相邻的两个Cb、Cr，分析，对于像素点Y'00、Y'01 而言，其Cb、Cr的值均为 Cb00、Cr00，其他的像素点的YUV取值依次类推。 **（2） UYVY 格式 （属于YUV422）**

start + 0:	Cb_{00}	Y'_{00}	Cr_{00}	Y'_{01}	Cb_{01}	Y'_{02}	Cr_{01}	Y'_{03}
start + 8:	Cb_{10}	Y'_{10}	Cr_{10}	Y'_{11}	Cb_{11}	Y'_{12}	Cr_{11}	Y'_{13}
start + 16:	Cb_{20}	Y'_{20}	Cr_{20}	Y'_{21}	Cb_{21}	Y'_{22}	Cr_{21}	Y'_{23}
start + 24:	Cb_{30}	Y'_{30}	Cr_{30}	Y'_{31}	Cb_{31}	Y'_{32}	Cr_{31}	Y'_{33}

UYVY格式也是YUV422采样的存储格式中的一种，只不过与YUYV不同的是UV的排列顺序不一样而已，还原其每个像素点的YUV值的方法与上面一样。

（3） YUV422P（属于YUV422）

T, URG)(24479)
3. GCC 编译详解(22001)
4. C++文件操作详解（ifstream、ofstream、fstream）(16013)
5. 字符编码之间的相互转换 UTF8与GBK(8329)

评论排行榜

1. GCC 编译详解(5)
2. 图文详解YUV420数据格式(3)
3. <<ffmpeg/ffplay源码剖析>> 笔记(1)
4. ffmpeg ./configure参数说明(1)
5. 字符编码之间的相互转换 UTF8与GBK(1)

推荐排行榜

1. 图文详解YUV420数据格式(6)
2. GCC 编译详解(5)
3. C++文件操作详解（ifstream、ofstream、fstream）(4)
4. TCP的状态 (SYN, FIN, ACK, PSH, RST, URG)(3)
5. 字符编码之间的相互转换 UTF8与GBK(2)

start + 0:	Y'_{00}	Y'_{01}	Y'_{02}	Y'_{03}
start + 4:	Y'_{10}	Y'_{11}	Y'_{12}	Y'_{13}
start + 8:	Y'_{20}	Y'_{21}	Y'_{22}	Y'_{23}
start + 12:	Y'_{30}	Y'_{31}	Y'_{32}	Y'_{33}
start + 16:	Cb_{00}	Cb_{01}		
start + 18:	Cb_{10}	Cb_{11}		
start + 20:	Cb_{20}	Cb_{21}		
start + 22:	Cb_{30}	Cb_{31}		
start + 24:	Cr_{00}	Cr_{01}		
start + 26:	Cr_{10}	Cr_{11}		
start + 28:	Cr_{20}	Cr_{21}		
start + 30:	Cr_{30}	Cr_{31}		

YUV422P也属于YUV422的一种，它是一种Plane模式，即平面模式，并不是将YUV数据交错存储，而是先存放所有的Y分量，然后存储所有的U（Cb）分量，最后存储所有的V（Cr）分量，如上图所示。其每一个像素点的YUV值提取方法也是遵循YUV422格式的最基本提取方法，即两个Y共用一个UV。比如，对于像素点Y'00、Y'01 而言，其Cb、Cr的值均为 Cb00、Cr00。

（4）YV12，YU12格式（属于YUV420）

start + 0:	Y'_{00}	Y'_{01}	Y'_{02}	Y'_{03}
start + 4:	Y'_{10}	Y'_{11}	Y'_{12}	Y'_{13}
start + 8:	Y'_{20}	Y'_{21}	Y'_{22}	Y'_{23}
start + 12:	Y'_{30}	Y'_{31}	Y'_{32}	Y'_{33}
start + 16:	Cr_{00}	Cr_{01}		
start + 18:	Cr_{10}	Cr_{11}		
start + 20:	Cb_{00}	Cb_{01}		
start + 22:	Cb_{10}	Cb_{11}		

YU12和YV12属于YUV420格式，也是一种Plane模式，将Y、U、V分量分别打包，依次存储。其每一个像素点的YUV数据提取遵循YUV420格式的提取方式，即4个Y分量共用一组UV。注意，上图中，Y'00、Y'01、Y'10、Y'11共用Cr00、Cb00，其他依次类推。

（5）NV12、NV21（属于YUV420）

start + 0:	Y'_{00}	Y'_{01}	Y'_{02}	Y'_{03}
start + 4:	Y'_{10}	Y'_{11}	Y'_{12}	Y'_{13}
start + 8:	Y'_{20}	Y'_{21}	Y'_{22}	Y'_{23}
start + 12:	Y'_{30}	Y'_{31}	Y'_{32}	Y'_{33}
start + 16:	Cb_{00}	Cr_{00}	Cb_{01}	Cr_{01}
start + 20:	Cb_{10}	Cr_{10}	Cb_{11}	Cr_{11}

NV12和NV21属于YUV420格式，是一种two-plane模式，即Y和UV分为两个Plane，但是UV（CbCr）为交错存储，而不是分为三个plane。其提取方式与上一种类似，即Y'00、Y'01、Y'10、Y'11共用Cr00、Cb00

YUV420 planar数据， 以720×488大小图象YUV420 planar为例，

其存储格式是： 共大小为(720×480×3>>1)字节，

分为三个部分:Y,U和V

Y分量： (720×480)个字节

U(Cb)分量：(720×480>>2)个字节

V(Cr)分量：(720×480>>2)个字节

三个部分内部均是行优先存储，三个部分之间是Y,U,V 顺序存储。

即YUV数据的0－－720×480字节是Y分量值，

720×480－－720×480×5/4字节是U分量

720×480×5/4 －－720×480×3/2字节是V分量。

4：2： 2 和4：2：0 转换：

最简单的方式：

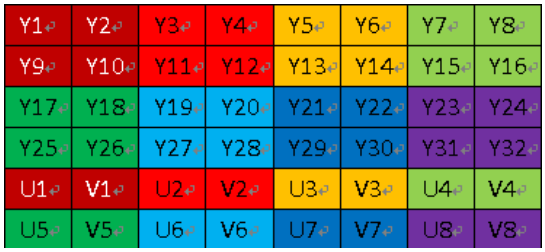
YUV4:2:2 ---> YUV4:2:0 Y不变，将U和V信号值在行(垂直方向)在进行一次隔行抽样。 YUV4:2:0 ---> YUV4:2:2 Y不变，将U和V信号值的每一行分别拷贝一份形成连续两行数据。

在YUV420中，一个像素点对应一个Y，一个4X4的小方块对应一个U和V。对于所有YUV420图像，它们的Y值排列是完全相同的，因为只有Y的图像就是灰度图像。YUV420sp与YUV420p的数据格式它们的UV排列在原理上是完全不同的。420p它是先把U存放完后，再存放V，也就是说UV它们是连续的。而420sp它是UV、UV这样交替存放的。(见下图) 有了上面的理论，我就可以准确的计算出一个YUV420在内存中存放的大小。
width * hight =Y（总和） U = Y / 4 V = Y / 4

所以YUV420 数据在内存中的长度是 width * hight * 3 / 2，

假设一个分辨率为8X4的YUV图像，它们的格式如下图：

YUV420sp格式如下图



YUV420p数据格式如下图

旋转90度的算法:

```
public static void rotateYUV240SP(byte[] src,byte[] des,int width,int height)
{

 int wh = width * height;
 //旋转Y
 int k = 0;
 for(int i=0;i<width;i++) {
  for(int j=0;j<height;j++)
  {
            des[k] = src[width*j + i];
       k++;
  }
 }

 for(int i=0;i<width;i+=2) {
  for(int j=0;j<height/2;j++)
  {
            des[k] = src[wh+ width*j + i];
            des[k+1]=src[wh + width*j + i+1];
       k+=2;
  }
```

azraelly 博客园 首页 新随笔 联系 订阅 管理

随笔- 49 文章- 2 评论- 14

YV12和I420的区别　　一般来说，直接采集到的视频数据是RGB24的格式，RGB24一帧的大小size＝width×heigth×3 Bit，RGB32的size＝width×heigth×4，如果是I420（即YUV标准格式4：2：0）的数据量是size＝width×heigth×1.5 Bit。　　在采集到RGB24数据后，需要对这个格式的数据进行第一次压缩。即将图像的颜色空间由RGB2YUV。因为，X264在进行编码的时候需要标准的YUV（4：2：0）。但是这里需要注意的是，虽然YV12也是（4：2：0），但是YV12和I420的却是不同的，在存储空间上面有些区别。如下： YV12： 亮度（行×列） ＋ U（行×列/4）＋V（行×列/4）

I420： 亮度（行×列） ＋ V（行×列/4）＋U（行×列/4）

可以看出，YV12和I420基本上是一样的，就是UV的顺序不同。

继续我们的话题，经过第一次数据压缩后RGB24－>YUV（I420）。这样，数据量将减少一半，为什么呢？呵呵，这个就太基础了，我就不多写了。同样，如果是RGB24－>YUV（YV12），也是减少一半。但是，虽然都是一半，如果是YV12的话效果就有很大损失。然后，经过X264编码后，数据量将大大减少。将编码后的数据打包，通过RTP实时传送。到达目的地后，将数据取出，进行解码。完成解码后，数据仍然是YUV格式的，所以，还需要一次转换，这样windows的驱动才可以处理，就是YUV2RGB24。

YUY2 是 4:2:2 [Y0 U0 Y1 V0]

yuv420p 和 YUV420的区别 在存储格式上有区别

yuv420p：yyyyyyyy uuuuuuuu vvvvv yuv420： yuv yuv yuv

YUV420P，Y，U，V三个分量都是平面格式，分为I420和YV12。I420格式和YV12格式的不同处在U平面和V平面的位置不同。在I420格式中，U平面紧跟在Y平面之后，然后才是V平面（即：YUV）；但YV12则是相反（即：YVU）。
YUV420SP, Y分量平面格式，UV打包格式, 即NV12。 NV12与NV21类似，U 和 V 交错排列,不同在于UV顺序。
I420: YYYYYYYY UU VV =>YUV420P
YV12: YYYYYYYY VV UU =>YUV420P
NV12: YYYYYYYY UVUV =>YUV420SP
NV21: YYYYYYYY VUVU =>YUV420SP

绿色通道： 好文要顶 关注我 收藏该文 与我联系

azraelly
关注 - 14
粉丝 - 37
+加关注

6 0

(请您对文章做出评价)

« 上一篇：ffmpeg ./configure参数说明
» 下一篇：directdraw overlay 和flip及blt的区别

posted @ 2013-01-01 01:14 azraelly 阅读(32748) 评论(3) 编辑 收藏

评论列表

#1楼 2013-07-16 15:34 凡人修行

写的很好，尤其是最后的
I420: YYYYYYYY UU VV =>YUV420P
YV12: YYYYYYYY VV UU =>YUV420P
NV12: YYYYYYYY UVUV =>YUV420SP
NV21: YYYYYYYY VUVU =>YUV420SP

支持(0) 反对(0)

#2楼 2013-10-02 17:17 多米诺

很详细很直观的《图文详解YUV420数据格式》

支持(0) 反对(0)

#3楼 2013-12-11 14:12 蜗##牛

脉络很清晰，赞一个！
另外
引用
RGB24一帧的大小size = width×heigth×3 Bit，RGB32的size = width×heigth×4，如果是I420（即YUV标准格式4：2：0）的数据量是 size = width×heigth×1.5 Bit
中的Bit是否笔误而应当是byte？谢谢！

支持(1) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
【推荐】融云即时通讯云 – 专注为 App 开发者提供IM云服务
【推荐】如何让你的程序拥有象Excel一样强大的数据编辑功能
【活动】RDS邀您6.5折体验PostgreSQL