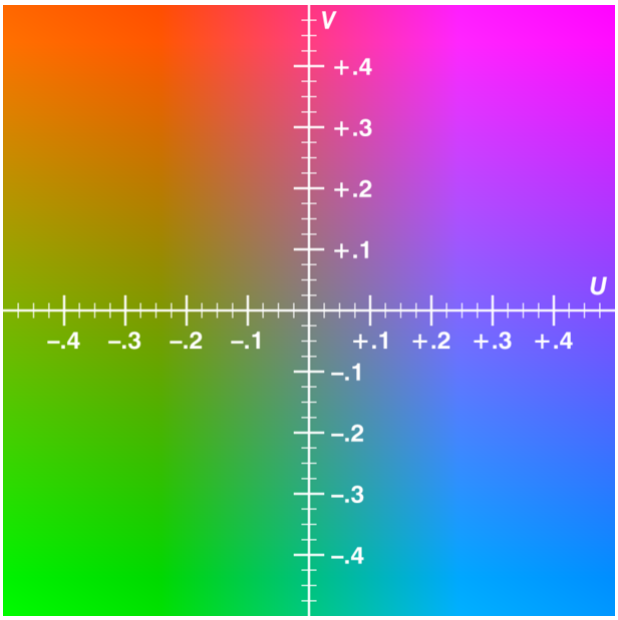
WIKIPEDIA

YUV

维基百科，自由的百科全书

YUV，是一种颜色编码方法。常使用在各个影像处理组件中。YUV在对照片或影片编码时，考虑到人类的感知能力，允许降低色度的带宽。



U–V color plane示例，Y value = 0.5，代表RGB色域（color gamut）

YUV是编译true-color颜色空间（color space）的种类，Y'UV, YUV, YCbCr，YPbPr等专有名词都可以称为YUV，彼此有重叠。“Y”表示明亮度（Luminance、Luma），“U”和“V”则是色度、浓度（Chrominance、Chroma），

Y′UV, YUV, YCbCr, YPbPr所指涉的范围，常有混淆或重叠的情况。从历史的演变来说，其中YUV和Y'UV通常用来编码电视的模拟信号，而YCbCr则是用来描述数字的影像信号，适合影片与图片压缩以及传输，例如MPEG、JPEG。但在现今，YUV通常已经在电脑系统上广泛使用。

Y'代表明亮度(luma; brightness)而U与V存储色度(色讯; chrominance; color)部分; 亮度(luminance)记作Y，而Y'的prime符号记作伽玛校正。

YUV Formats分成两个格式：

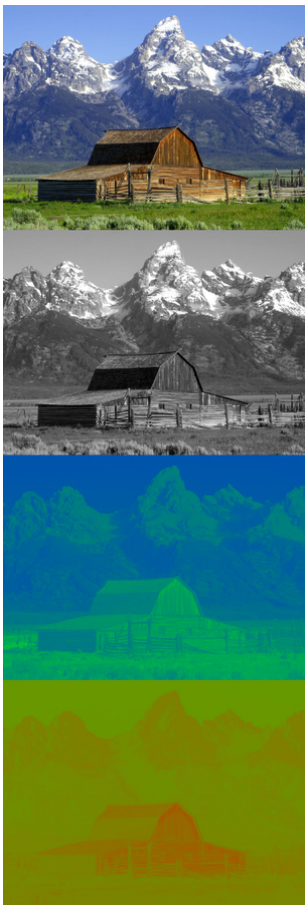
- 紧缩格式（packed formats）：将Y、U、V值存储成Macro Pixels数组，和RGB的存放方式类似。
- 平面格式（planar formats）：将Y、U、V的三个分量分别存放在不同的矩阵中。

紧缩格式（packed format）中的YUV是混合在一起的，对于YUV4:4:4格式而言，用紧缩格式很合适的，因此就有了UYVY、YUYV等。平面格式（planar formats）是指每Y分量，U分量和V分量都是以独立的平面组织的，也就是说所有的U分量必须在Y分量后面，而V分量在所有的U分量后面，此一格式适用于采样（subsample）。平面格式（planar format）有I420（4:2:0）、YV12、IYUV等。

目录

历史

Y'UV的发明是由于彩色电视与黑白电视的过渡时期[1]。黑白视频只有Y（Luma，Luminance）视频，也就是灰阶值。到了彩色电视规格的制定，是以YUV/YIQ的格式来处理彩色电视图像，把UV视作表示彩度的C（Chrominance或Chroma），如果忽略C信号，那么剩下的Y（Luma）信号就跟之前的黑白电视频号相同，这样一来便解决彩色电视机与黑白电视机的兼容问题。Y'UV最大的优点在于只需占用极少的带宽。



图像中的Y', U，和V组成

因为UV分别代表不同颜色信号，所以直接使用R与B信号表示色度的UV。也就是说UV信号告诉了电视要偏移某象素的的颜色，而不改变其亮度。或者UV信号告诉了显示器使得某个颜色亮度依某个基准偏移。 UV的值越高，代表该像素会有更饱和的颜色。

彩色图像记录的格式，常见的有RGB、YUV、CMYK等。彩色电视最早的构想是使用RGB三原色来同时传输。这种设计方式是原来黑白带宽的3倍，在当时并不是很好的设计。RGB诉求于人眼对色彩的感应，YUV则着重于视觉对于亮度的敏感程度，Y代表的是亮度，UV代表的是彩度（因此黑白电影可省略UV，相近于RGB），分别用Cr和Cb来表示，因此YUV的记录通常以Y:UV的格式呈现。

常用的YUV格式

为节省带宽起见，大多数YUV格式平均使用的每像素位数都少于24位。主要的抽样（subsample）格式有YCbCr 4:2:0、YCbCr 4:2:2、YCbCr 4:1:1和YCbCr 4:4:4。YUV的表示法称为A:B:C表示法：

- 4:4:4表示完全取样。
- 4:2:2表示2:1的水平取样，垂直完全采样。
- 4:2:0表示2:1的水平取样，垂直2：1采样。
- 4:1:1表示4:1的水平取样，垂直完全采样。

最常用Y:UV记录的比重通常1:1或2:1，DVD-Video是以YUV 4:2:0的方式记录，也就是我们俗称的**I420**，YUV4:2:0并不是说只有U（即Cb），V（即Cr）一定为0，而是指U：V互相援引，时见时隐，也就是说对于每一个行，只有一个U或者V分量，如果一行是4:2:0的话，下一行就是4:0:2，再下一行是4:2:0...以此类推。至于其他常见的YUV格式有YUY2、YUYV、YVYU、UYVY、AYUV、Y41P、Y411、Y211、IF09、IYUV、YV12、YVU9、YUV411、YUV420等。

YUY2及常见表示方法

YUY2（和**YUYV**）格式为像素保留Y，而UV在水平空间上相隔二个像素采样一次（Y0 U0 Y1 V0），（Y2 U2 Y3 V2）...其中，（Y0 U0 Y1 V0）就是一个macro-pixel（宏像素），它表示了2个像素，（Y2 U2 Y3 V2）是另外的2个像素。 以此类推，再如：Y41P（和Y411）格式为每个像素保留Y分量，而UV分量在水平方向上每4个像素采样一次。一个宏像素为12个字节，实际表示8个像素。图像数据中YUV分量排列顺序如下：（U0 Y0 V0 Y1 U4 Y2 V4 Y3 Y4 Y5 Y6 Y7）...

YVYU UYVY

YVYU, **UYVY**格式跟YUY2类似，只是排列顺序有所不同。**Y211**格式是Y每2个像素采样一次，而UV每4个像素采样一次。**AYUV**格式则有一Alpha通道。

YV12

YV12格式与**IYUV**类似，每个像素都提取Y，在UV提取时，将图像2 x 2的矩阵，每个矩阵提取一个U和一个V。YV12格式和I420格式的不同处在V平面和U平面的位置不同。在YV12格式中，V平面紧跟在Y平面之后，然后才是U平面（即：YVU）；但I420则是相反（即：YUV）。**NV12**与YV12类似，效果一样，YV12中U和V是连续排列的，而在NV12中，U和V就交错排列的。

排列举例： **2*2**图像YYYYVU； **4 * 4**图像YYYYYYYYYYYYYYYYVVVVUUUU

转换

YUV与RGB的转换公式：

U和V组件可以被表示成原始的R，G，和B（R，G，B为γ预校正后的）：

$$
\begin{aligned}
Y &= 0.299 * R + 0.587 * G + 0.114 * B \\
U &= -0.169 * R - 0.331 * G + 0.5 * B + 128 \\
V &= 0.5 * R - 0.419 * G - 0.081 * B + 128
\end{aligned}
$$

如一般顺序，转移组件的范围可得到：

$$
\begin{aligned}
Y &\in [0, 255] \\
U &\in [0, 255] \\
V &\in [0, 255]
\end{aligned}
$$

在逆转关系上，从YUV到RGB，可得

$$
\begin{aligned}
R &= Y + 1.13983 * (V - 128) \\
G &= Y - 0.39465 * (U - 128) - 0.58060 * (V - 128) \\
B &= Y + 2.03211 * (U - 128)
\end{aligned}
$$

取而代之，以矩阵表示法（matrix representation），可得到公式：

$$
\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}
$$

$$
\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -0.00093 & 1.401687 \\ 1 & -0.3437 & -0.71417 \\ 1 & 1.77216 & 0.00099 \end{bmatrix} \begin{bmatrix} Y \\ U - 128 \\ V - 128 \end{bmatrix}
$$

YUV转RGB

function RGB* YUV444toRGB888(Y, U, V)；将YUV format移转成简单的RGB format并可以用浮点运算实现：

Y'UV444

大多数YUV格式平均使用的每像素位数都少于24位。YUV444是最逼真的格式，一格不删（24 bits），即每4个Y，配上4个U，还有4个V；YUV422则是在UV格式上减半，即每4个Y，配2个U，2个V；YUV420则是在UV上减至1/4之格式，即每4个Y，配1个U，再配1个V。

这些公式是基于NTSC standard;

$$
\begin{aligned}
Y' &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\
U &= -0.147 \times R - 0.289 \times G + 0.436 \times B \\
V &= 0.615 \times R - 0.515 \times G - 0.100 \times B
\end{aligned}
$$

在早期的非SIMD（non-SIMD）构造中，floating point arithmetic会比fixed-point arithmetic稍慢，所以有一替代公式如下：

$$
\begin{aligned}
C &= Y' - 16 \\
D &= U - 128 \\
E &= V - 128
\end{aligned}
$$

使用前面的系数并且用clip()注明切割的值域是0至255，如下的公式是从Y'UV到RGB (NTSC version):

$$
\begin{aligned}
R &= clip((298 \times C + 409 \times E + 128) >> 8) \\
G &= clip((298 \times C - 100 \times D - 208 \times E + 128) >> 8) \\
B &= clip((298 \times C + 516 \times D + 128) >> 8)
\end{aligned}
$$

注意：上述的公式多暗示为YCbCr. 虽然称为YUV，但应该严格区分YUV和YCbCr这两个专有名词有时并非完全相同。

ITU-R版本的公式差异：

$$
Y = 0.299 \times R + 0.587 \times G + 0.114 \times B + 0
$$

$$Cb = -0.169 \times R - 0.331 \times G + 0.499 \times B + 128$$
$$Cr = 0.499 \times R - 0.418 \times G - 0.0813 \times B + 128$$

$$R = clip(Y + 1.402 \times (Cr - 128))$$
$$G = clip(Y - 0.344 \times (Cb - 128) - 0.714 \times (Cr - 128))$$
$$B = clip(Y + 1.772 \times (Cb - 128))$$

ITU-R标准YCbCr（每一通道8位）至RGB888:

Cr = Cr - 128; Cb = Cb - 128;

$$R = Y + Cr + Cr >> 2 + Cr >> 3 + Cr >> 5$$
$$G = Y - (Cb >> 2 + Cb >> 4 + Cb >> 5) - (Cr >> 1 + Cr >> 3 + Cr >> 4 + Cr >> 5)$$
$$B = Y + Cb + Cb >> 1 + Cb >> 2 + Cb >> 6$$

Y'UV422

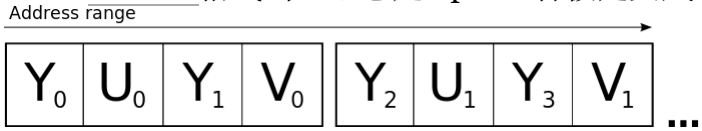
Input：读取Y'UV的4bytes（u, y1, v, y2）
Output：写入RGB的6bytes（R, G, B, R, G, B）

```
u = yuv[0];
y1 = yuv[1];
v = yuv[2];
y2 = yuv[3];
```

以此一信息可以剖析出regular Y'UV444格式而成为2 RGB pixels info:

```
rgb1 = Y'UV444toRGB888(y1, u, v);
rgb2 = Y'UV444toRGB888(y2, u, v);
```

Y'UV422可被表达成Y'UY'2 FourCC格式码。意思是2 pixels将被定义成each macropixel (four bytes) treated in the image.



Y'UV411

```
// Extract YUV components
u = yuv[0];
y1 = yuv[1];
y2 = yuv[2];
v = yuv[3];
y3 = yuv[4];
y4 = yuv[5];
```

```
rgb1 = Y'UV444toRGB888(y1, u, v);
rgb2 = Y'UV444toRGB888(y2, u, v);
rgb3 = Y'UV444toRGB888(y3, u, v);
rgb4 = Y'UV444toRGB888(y4, u, v);
```

所以结果会得到4 RGB像素的值 (4*3 bytes) from 6 bytes. This means reducing size of transferred data to half and with quite good loss of quality.

YV12

The Y'V12的格式相当类似Y'UV420p，但U与V数据反转：Y'跟随着V, U殿后。Y'UV420p与Y'V12使用相同算法。许多重要的编码器都采用YV12空间存储视频：MPEG-4（x264，XviD，DivX），DVD-Video存储格式MPEG-2，MPEG-1以及MJPEG。

将Y'UV420p转换成RGB

```
Height = 16;
Width = 16;
Y'ArraySize = Height × Width;    // (256)
Y' = Array[7 × Width + 5];
U = Array[(7/2) × (Width/2) + 5/2 + Y'ArraySize];
V = Array[(7/2) × (Width/2) + 5/2 + Y'ArraySize + Y'ArraySize/4];
```

```
RGB = Y'UV444toRGB888(Y', U, V);
```

注释

1. Maller, Joe. RGB and YUV Color (http://joemaller.com/fcp/fxscript_yuv_color.shtml) 互联网档案馆的存档 (https://web.archive.org/web/20080224143835/http://www.joemaller.com/fcp/fxscript_yuv_color.shtml)，存档日期2008-02-24., *FXScript Reference*

参见

- 色度抽样

外部链接

- RGB/YUV Pixel Conversion (http://www.fourcc.org/fccyvrgb.php)
- Explanation of many different formats in the YUV family (http://www.fourcc.org/yuv.php)
- Charles Poynton – Video engineering (http://www.poynton.com/Poynton-video-eng.html)
- YUV422 to RGB using SSE/Assembly (http://www.mikekohn.net/stuff/image_processing.php)