**YUV** is a [color space](http://en.wikipedia.org/wiki/Color_space) typically used as part of a [color image pipeline](http://en.wikipedia.org/wiki/Color_image_pipeline). It encodes a color image or video taking human perception into account, allowing reduced bandwidth for [chrominance](http://en.wikipedia.org/wiki/Chrominance) components, thereby typically enabling transmission errors or compression artifacts to be more efficiently masked by the human perception than using a "direct" RGB-representation. Other color spaces have similar properties, and the main reason to implement or investigate properties of Y'UV would be for interfacing with analog or digital television or photographic equipment that conforms to certain Y'UV standards.

The scope of the terms Y'UV, YUV, [YCbCr](http://en.wikipedia.org/wiki/YCbCr), [YPbPr](http://en.wikipedia.org/wiki/YPbPr), etc., is sometimes ambiguous and overlapping. Historically, the terms YUV and Y'UV were used for a specific *analog encoding* of color information in television systems, while YCbCr was used for *digital encoding* of color information suited for video *and* still-image compression and transmission such as [MPEG](http://en.wikipedia.org/wiki/MPEG) and [JPEG](http://en.wikipedia.org/wiki/JPEG). Today, the term YUV is commonly used in the computer industry to describe *file-formats* that are encoded using YCbCr.

The Y'UV model defines a [color space](http://en.wikipedia.org/wiki/Color_space) in terms of one [luma](http://en.wikipedia.org/wiki/Luma_(video)) (Y') and two [chrominance](http://en.wikipedia.org/wiki/Chrominance) (UV) components. The Y'UV color model is used in the [PAL](http://en.wikipedia.org/wiki/PAL) and [SECAM](http://en.wikipedia.org/wiki/SECAM) [composite color video](http://en.wikipedia.org/wiki/Composite_video) standards. Previous black-and-white systems used only luma (Y') information. Color information (U and V) was added separately via a [sub-carrier](http://en.wikipedia.org/wiki/Sub-carrier) so that a black-and-white receiver would still be able to receive and display a color picture transmission in the receiver's native [black-and-white](http://en.wikipedia.org/wiki/Black-and-white) format.

Y' stands for the [luma](http://en.wikipedia.org/wiki/Luma_(video)) component (the brightness) and U and V are the [chrominance](http://en.wikipedia.org/wiki/Chrominance) (color) components; luminance is denoted by Y and luma by Y' – the prime symbols (') denote [gamma compression](http://en.wikipedia.org/wiki/Gamma_compression),[[1]](http://en.wikipedia.org/wiki/YUV#cite_note-1) with "luminance" meaning perceptual ([color science](http://en.wikipedia.org/wiki/Color_science)) brightness, while "luma" is electronic (voltage of display) brightness.

The [YPbPr](http://en.wikipedia.org/wiki/YPbPr) color model used in analog [component video](http://en.wikipedia.org/wiki/Component_video) and its digital version [YCbCr](http://en.wikipedia.org/wiki/YCbCr) used in digital video are more or less derived from it, and are sometimes called Y'UV. (CB/PB and CR/PR are deviations from grey on blue–yellow and red–cyan axes, whereas U and V are blue–luminance and red–luminance differences.) The [Y'IQ](http://en.wikipedia.org/wiki/YIQ) color space used in the analog [NTSC](http://en.wikipedia.org/wiki/NTSC) television broadcasting system is related to it, although in a more complex way.

|  |
| --- |
| **Contents**   [[hide](http://en.wikipedia.org/wiki/YUV)]   * [1 History](http://en.wikipedia.org/wiki/YUV#History) * [2 Conversion to/from RGB](http://en.wikipedia.org/wiki/YUV#Conversion_to.2Ffrom_RGB)   + [2.1 Notes](http://en.wikipedia.org/wiki/YUV#Notes)   + [2.2 BT.709 and BT.601](http://en.wikipedia.org/wiki/YUV#BT.709_and_BT.601) * [3 Numerical approximations](http://en.wikipedia.org/wiki/YUV#Numerical_approximations) * [4 Luminance/chrominance systems in general](http://en.wikipedia.org/wiki/YUV#Luminance.2Fchrominance_systems_in_general) * [5 Confusion with Y'CbCr](http://en.wikipedia.org/wiki/YUV#Confusion_with_Y.27CbCr) * [6 Types of sampling](http://en.wikipedia.org/wiki/YUV#Types_of_sampling) * [7 Converting between Y'UV and RGB](http://en.wikipedia.org/wiki/YUV#Converting_between_Y.27UV_and_RGB)   + [7.1 Y'UV444 to RGB888 conversion](http://en.wikipedia.org/wiki/YUV#Y.27UV444_to_RGB888_conversion)   + [7.2 Y'UV422 to RGB888 conversion](http://en.wikipedia.org/wiki/YUV#Y.27UV422_to_RGB888_conversion)   + [7.3 Y'UV411 to RGB888 conversion](http://en.wikipedia.org/wiki/YUV#Y.27UV411_to_RGB888_conversion)   + [7.4 Y'UV420p (and Y'V12 or YV12) to RGB888 conversion](http://en.wikipedia.org/wiki/YUV#Y.27UV420p_.28and_Y.27V12_or_YV12.29_to_RGB888_conversion)   + [7.5 Y'UV420p (NV21) to ARGB8888 conversion](http://en.wikipedia.org/wiki/YUV#Y.27UV420p_.28NV21.29_to_ARGB8888_conversion) * [8 References](http://en.wikipedia.org/wiki/YUV#References) * [9 External links](http://en.wikipedia.org/wiki/YUV#External_links) |

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=1)**] History**

Y'UV was invented when engineers wanted [color television](http://en.wikipedia.org/wiki/Color_television) in a [black-and-white](http://en.wikipedia.org/wiki/Black-and-white) infrastructure.[[2]](http://en.wikipedia.org/wiki/YUV#cite_note-2) They needed a signal transmission method that was compatible with black-and-white (B&W) TV while being able to add color. The luma component already existed as the black and white signal; they added the UV signal to this as a solution.

The UV representation of chrominance was chosen over straight R and B signals because U and V are color difference signals. This meant that in a black and white scene the U and V signals would be zero and only the Y' signal would need to be transmitted. If R and B were to have been used, these would have non-zero values even in a B&W scene, requiring all three data-carrying signals. This was important in the early days of color television, because holding the U and V signals to zero while connecting the black and white signal to Y' allowed color TV sets to display B&W TV without the additional expense and complexity of special B&W circuitry. In addition, black and white receivers could take the Y' signal and ignore the color signals, making Y'UV backward-compatible with all existing black-and-white equipment, input and output. It was necessary to assign a narrower bandwidth to the chrominance channel because there was no additional bandwidth available. If some of the luminance information arrived via the chrominance channel (as it would have if RB signals were used instead of differential UV signals), B&W resolution would have been compromised.[[3]](http://en.wikipedia.org/wiki/YUV#cite_note-3)

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=2)**] Conversion to/from RGB**

Y'UV signals are typically created from [RGB](http://en.wikipedia.org/wiki/RGB) ([red](http://en.wikipedia.org/wiki/Red), [green](http://en.wikipedia.org/wiki/Green) and [blue](http://en.wikipedia.org/wiki/Blue)) source. Weighted values of R, G, and B are summed to produce Y', a measure of overall brightness or luminance. U and V are computed as scaled differences between Y' and the B and R values.

Defining the following constants:

\begin{align}
W_R &= 0.299 \\
W_B &= 0.114 \\
W_G &= 1 - W_R - W_B = 0.587 \\
U_{Max} &= 0.436 \\
V_{Max} &= 0.615
\end{align}

Y'UV is computed from RGB as follows:

\begin{align}
Y' &= W_R R + W_G G + W_B B \\
U &= U_{Max} \frac{B - Y'}{1 - W_B} \approx 0.492(B - Y') \\
V &= V_{Max} \frac{R - Y'}{1 - W_R} \approx 0.877(R - Y')
\end{align}

The resulting ranges of Y', U, and V respectively are [0, 1], [-*U*Max, *U*Max], and [-*V*Max, *V*Max].

Inverting the above transformation converts Y'UV to RGB:

\begin{align}
R &= Y' + V \frac{1 - W_R}{V_{Max}} = Y' +  \frac{V}{0.877}\\
G &= Y' - U \frac{W_B (1 - W_B)}{U_{Max} W_G} - V \frac{W_R (1 - W_R)}{V_{Max} W_G} = Y' - \frac{0.232 U}{0.587} -  \frac{0.341 V}{0.587} = Y' - 0.395 U - 0.581 V\\
B &= Y' + U \frac{1 - W_B}{U_{Max}} =Y' +  \frac{U}{0.492}
\end{align}

Equivalently, substituting values for the constants and expressing them as matrices gives:


\begin{bmatrix} Y' \\ U \\ V \end{bmatrix}
=
\begin{bmatrix}
  0.299   &  0.587   &  0.114 \\
 -0.14713 & -0.28886 &  0.436 \\
  0.615   & -0.51499 & -0.10001
\end{bmatrix}
\begin{bmatrix} R \\ G \\ B \end{bmatrix}



\begin{bmatrix} R \\ G \\ B \end{bmatrix}
=
\begin{bmatrix}
 1 &  0       &  1.13983 \\
 1 & -0.39465 & -0.58060 \\
 1 &  2.03211 &  0
\end{bmatrix}
\begin{bmatrix} Y' \\ U \\ V \end{bmatrix}


**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=3)**] Notes**

The weights used to compute Y' (top row of matrix) are identical to those used in the [Y'IQ](http://en.wikipedia.org/wiki/YIQ) color space.

Equal values of red, green and blue (i.e. levels of gray) yield 0 for U and V. Black, RGB=(0, 0, 0), yields YUV=(0, 0, 0). White, RGB=(1, 1, 1), yields YUV=(1, 0, 0).

These formulas are traditionally used in analog televisions and equipment; digital equipment such as [HDTV](http://en.wikipedia.org/wiki/HDTV) and digital video cameras use Y'CbCr.

* UV Planes in a range of [-1,1]
* [](http://en.wikipedia.org/wiki/File:YUV_UV_plane_Y0_100_percent.png)

Y' value of 0

* [](http://en.wikipedia.org/wiki/File:YUV_UV_plane_Y0.5_100_percent.png)

Y' value of 0.5

* [](http://en.wikipedia.org/wiki/File:YUV_UV_plane_Y1_100_percent.png)

Y' value of 1

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=4)**] BT.709 and BT.601**

When standardising [high-definition video](http://en.wikipedia.org/wiki/High-definition_video), the [ATSC](http://en.wikipedia.org/wiki/ATSC) chose a different formula for the YCbCr than that used for standard-definition video. This means that when converting between SDTV and HDTV, the color information has to be altered, in addition to [image scaling](http://en.wikipedia.org/wiki/Image_scaling) the video.

The formulae above reference [Rec. 601](http://en.wikipedia.org/wiki/Rec._601). For HDTV, a slightly different matrix is used, where WR and WB in the above formula is replaced by [Rec. 709](http://en.wikipedia.org/wiki/Rec._709):


\begin{array}{rl}
W_R &= 0.2126 \\
W_B &= 0.0722 \\
\end{array}


This gives the matrices :


\begin{bmatrix} Y' \\ U \\ V \end{bmatrix}
=
\begin{bmatrix}
  0.2126  &  0.7152  &  0.0722 \\
 -0.09991 & -0.33609 &  0.436 \\
  0.615   & -0.55861 & -0.05639
\end{bmatrix}
\begin{bmatrix} R \\ G \\ B \end{bmatrix}



\begin{bmatrix} R \\ G \\ B \end{bmatrix}
=
\begin{bmatrix}
 1 &  0       &  1.28033 \\
 1 & -0.21482 & -0.38059 \\
 1 &  2.12798 &  0
\end{bmatrix}
\begin{bmatrix} Y' \\ U \\ V \end{bmatrix}


**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=5)**] Numerical approximations**

Prior to the development of fast [SIMD](http://en.wikipedia.org/wiki/SIMD) floating-point processors, most digital implementations of RGB->Y'UV used integer math, in particular [fixed-point](http://en.wikipedia.org/wiki/Fixed-point_arithmetic) approximations. In the following examples, the operator "a \gg b" denotes a right-shift of *a* by *b* bits.

Traditional 8 bit representation of Y'UV with unsigned integers uses the following

1. Basic transform

\begin{bmatrix}Y' \\ U \\ V \end{bmatrix} =
\begin{bmatrix}
  66 & 129 & 25  \\
 -38 & -74 & 112 \\
 112 & -94 & -18
\end{bmatrix}
\begin{bmatrix} R \\ G \\ B \end{bmatrix}


2. Scale down to 8 bits with rounding


\begin{array}{rcl}
Y' &=& (Y' + 128) \gg 8\\
U  &=& (U  + 128) \gg 8\\
V  &=& (V  + 128) \gg 8
\end{array}


3. Shift values


\begin{array}{rcl}
Y' &=& Y' + 16\\
U  &=& U + 128\\
V  &=& V + 128
\end{array}


Y' values are conventionally shifted and scaled to the range [16, 235] (referred to as studio swing) rather than using the full range of [0, 255] (referred to as full swing). This confusing practice derives from the MPEG standards and explains why 16 is added to Y' and why the Y' coefficients in the basic transform sum to 220 instead of 255. U and V values, which may be positive or negative, are summed with 128 to make them always positive.[[4]](http://en.wikipedia.org/wiki/YUV#cite_note-4)

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=6)**] Luminance/chrominance systems in general**

The primary advantage of luma/chroma systems such as Y'UV, and its relatives [Y'IQ](http://en.wikipedia.org/wiki/YIQ) and [YDbDr](http://en.wikipedia.org/wiki/YDbDr), is that they remain compatible with black and white [analog television](http://en.wikipedia.org/wiki/Analog_television) (largely due to the work of [Georges Valensi](http://en.wikipedia.org/wiki/Georges_Valensi)). The Y' channel saves all the data recorded by black and white cameras, so it produces a signal suitable for reception on old monochrome displays. In this case, the U and V are simply discarded. If displaying color, all three channels are used, and the original RGB information can be decoded.

Another advantage of Y'UV is that some of the information can be discarded in order to reduce [bandwidth](http://en.wikipedia.org/wiki/Bandwidth_(signal_processing)). The human eye has fairly little spatial sensitivity to color: the accuracy of the brightness information of the luminance channel has far more impact on the image detail discerned than that of the other two. Understanding this human shortcoming, standards such as [NTSC](http://en.wikipedia.org/wiki/NTSC) reduce the bandwidth of the chrominance channels considerably. (Bandwidth is in the temporal domain, but this translates into the spatial domain as the image is scanned out.)

Therefore, the resulting U and V signals can be substantially "compressed". In the [NTSC](http://en.wikipedia.org/wiki/NTSC) (Y'IQ) and [PAL](http://en.wikipedia.org/wiki/PAL) systems, the chrominance signals had significantly narrower bandwidth than that for the luminance. Early versions of NTSC rapidly alternated between particular colors in identical image areas to make them appear adding up to each other to the human eye, while all modern analogue and even most digital video standards use [chroma subsampling](http://en.wikipedia.org/wiki/Chroma_subsampling) by recording a picture's color information at only half the resolution compared to the brightness information. This ratio is the most common form, known as 4:2:2, which is the sample ratio of Y:U:V, or Y:I:Q. The 4:x:x standard was adopted due to the very earliest color NTSC standard which used a chroma subsampling of 4:1:1 so that the picture carried only a quarter as much resolution in color than it did in brightness. Today, only high-end equipment processing uncompressed signals uses a chroma subsampling of 4:4:4 with identical resolution for both brightness and color information.

The I and Q axes were chosen according to bandwidth needed by human vision, one axis being that requiring the most bandwidth, and the other (fortuitously at 90 degrees) the minimum. However, true I and Q demodulation was relatively more complex, requiring two analog delay lines, and NTSC receivers rarely used it.

However, this color space conversion is [lossy](http://en.wikipedia.org/wiki/Lossy_compression), particularly obvious in [crosstalk](http://en.wikipedia.org/wiki/Crosstalk) from the luma to the chroma-carrying wire, and vice versa, in analogue equipment (including [RCA connectors](http://en.wikipedia.org/wiki/RCA_connector) to transfer a digital signal, as all they carry is analogue [composite video](http://en.wikipedia.org/wiki/Composite_video), which is either YUV, YIQ, or even [CVBS](http://en.wikipedia.org/wiki/CVBS)). Furthermore, NTSC and PAL encoded color signals in a manner that causes high bandwidth chroma and luma signals to mix with each other in a bid to maintain backward compatibility with black and white television equipment, which results in [dot crawl](http://en.wikipedia.org/wiki/Dot_crawl) and [cross color](http://en.wikipedia.org/w/index.php?title=Cross_color&action=edit&redlink=1) artifacts. When the NTSC standard was created in the 1950s, this was not a real concern since the quality of the image was limited by the monitor equipment, not the limited-bandwidth signal being received. However today's modern television is capable of displaying more information than is contained in these lossy signals. To keep pace with the abilities of new display technologies, attempts were made since the late 1970s to preserve more of the Y'UV signal while transferring images, such as [SCART](http://en.wikipedia.org/wiki/SCART) (1977) and [S-Video](http://en.wikipedia.org/wiki/S-Video) (1987) connectors.

Instead of Y'UV, Y'CbCr was used as the standard format for (digital) common [video compression](http://en.wikipedia.org/wiki/Video_compression) algorithms such as [MPEG-2](http://en.wikipedia.org/wiki/MPEG-2). Digital television and DVDs preserve their [compressed video](http://en.wikipedia.org/wiki/Video_compression) streams in the [MPEG-2](http://en.wikipedia.org/wiki/MPEG-2) format, which uses a full Y'CbCr color space, although retaining the established process of chroma subsampling. The professional [CCIR 601](http://en.wikipedia.org/wiki/CCIR_601) digital video format also uses Y'CbCr at the common chroma subsampling rate of 4:2:2, primarily for compatibility with previous analog video standards. This stream can be easily mixed into any output format needed.

Y'UV is not an [absolute color space](http://en.wikipedia.org/wiki/Absolute_color_space). It is a way of encoding RGB information, and the actual color displayed depends on the actual RGB colorants used to display the signal. Therefore a value expressed as Y'UV is only predictable if standard RGB colorants are used (i.e. a fixed set of primary chromaticities, or particular set of red, green, and blue).

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=7)**] Confusion with Y'CbCr**

Y'UV is often used as the term for [YCbCr](http://en.wikipedia.org/wiki/YCbCr). However, they are different formats. Y'UV is an analog system with scale factors different from the digital Y'CbCr system.[[5]](http://en.wikipedia.org/wiki/YUV#cite_note-5)

In digital video/image systems, Y'CbCr is the most common way to express color in a way suitable for compression/transmission. The confusion stems from computer implementations and text-books erroneously using the term YUV where Y'CbCr would be correct.

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=8)**] Types of sampling**

To get a digital signal, Y'UV images can be [sampled](http://en.wikipedia.org/wiki/Sample_(signal)) in several different ways; see [chroma subsampling](http://en.wikipedia.org/wiki/Chroma_subsampling).

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=9)**] Converting between Y'UV and RGB**

RGB files are typically encoded in 8, 12, 16 or 24 bits per pixel. In these examples, we will assume 24 bits per pixel, which is written as RGB888. The standard byte format is

r0 = rgb[0];

g0 = rgb[1];

b0 = rgb[2];

r1 = rgb[3];

g1 = rgb[4];

b1 = rgb[5];

...

Y'UV files can be encoded in 12, 16 or 24 bits per pixel. The common formats are Y'UV444 (or YUV444), YUV411, Y'UV422 (or YUV422) and Y'UV420p (or YUV420). The apostrophe after the Y is often omitted, as is the "p" after YUV420p. In terms of actual file formats, YUV420 is the most common, as the data is more easily compressed, and the file extension is usually ".YUV".

To convert from RGB to YUV or back, it is simplest to use RGB888 and YUV444. For YUV411, YUV422 and YUV420, the bytes need to be converted to YUV444 first.

YUV444 3 bytes per pixel

YUV422 4 bytes per 2 pixels

YUV411 6 bytes per 4 pixels

YUV420p 6 bytes per 4 pixels, reordered

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=10)**] Y'UV444 to RGB888 conversion**

The function [R, G, B] = Y'UV444toRGB888(Y', U, V) converts Y'UV format to simple RGB format

The RGB conversion formulae used for Y'UV444 format are also applicable to the standard NTSC TV transmission format of YUV420 (or YUV422 for that matter). For YUV420, since each U or V sample is used to represent 4 Y samples that form a square, a proper sampling method can allow the utilization of the exact conversion formulae shown below. For more details, please see the 420 format demonstration in the bottom section of this article.

These formulae are based on the NTSC standard;

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

On older, non-[SIMD](http://en.wikipedia.org/wiki/SIMD) architectures, floating point arithmetic is much slower than using fixed-point arithmetic, so an alternative formulation is:

C = Y' - 16

D = U - 128

E = V - 128

Using the previous coefficients and noting that clamp() denotes clamping a value to the range of 0 to 255, the following formulae provide the conversion from Y'UV to RGB (NTSC version):

R = \mathrm{clamp}(( 298 \times C                + 409 \times E + 128) >> 8)

G = \mathrm{clamp}(( 298 \times C - 100 \times D - 208 \times E + 128) >> 8)

B = \mathrm{clamp}(( 298 \times C + 516 \times D                + 128) >> 8)

Note: The above formulae are actually implied for YCbCr. Though the term YUV is used here, it should be noted that YUV and YCbCr are not exactly the same in a strict manner.

The ITU-R version of the formulae is different:

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 = \mathrm{clamp}(Y + 1.402 \times (Cr - 128))

G = \mathrm{clamp}(Y - 0.344 \times (Cb - 128) - 0.714 \times (Cr - 128))

B = \mathrm{clamp}(Y + 1.772 \times (Cb - 128))

Integer operation of ITU-R standard for YCbCr(8 bits per channel) to 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)

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=11)**] Y'UV422 to RGB888 conversion**

Input: Read 4 bytes of Y'UV (u, y1, v, y2 )

Output: Writes 6 bytes of RGB (R, G, B, R, G, B)

y1 = yuv[0];

u = yuv[1];

y2 = yuv[2];

v = yuv[3];

Using this information it could be parsed as regular Y'UV444 format to get 2 RGB pixels info:

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

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

Y'UV422 can also be expressed in YUY2 [FourCC](http://en.wikipedia.org/wiki/FourCC) format code. That means 2 pixels will be defined in each macropixel (four bytes) treated in the image. [Yuv422 yuy2.svg](http://en.wikipedia.org/wiki/File:Yuv422_yuy2.svg).

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=12)**] Y'UV411 to RGB888 conversion**

Input: Read 6 bytes of Y'UV

Output: Writes 12 bytes of RGB

// 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);

So the result is we are getting 4 RGB pixels values (4\*3 bytes) from 6 bytes. This means reducing the size of transferred data to half, with a loss of quality.

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=13)**] Y'UV420p (and Y'V12 or YV12) to RGB888 conversion**

Y'UV420p is a planar format, meaning that the Y', U, and V values are grouped together instead of interspersed. The reason for this is that by grouping the U and V values together, the image becomes much more compressible. When given an array of an image in the Y'UV420p format, all the Y' values come first, followed by all the U values, followed finally by all the V values.

The Y'V12 format is essentially the same as Y'UV420p, but it has the U and V data switched: the Y' values are followed by the V values, with the U values last. As long as care is taken to extract U and V values from the proper locations, both Y'UV420p and Y'V12 can be processed using the same algorithm.

As with most Y'UV formats, there are as many Y' values as there are pixels. Where X equals the height multiplied by the width, the first X indices in the array are Y' values that correspond to each individual pixel. However, there are only one fourth as many U and V values. The U and V values correspond to each 2 by 2 block of the image, meaning each U and V entry applies to four pixels. After the Y' values, the next X/4 indices are the U values for each 2 by 2 block, and the next X/4 indices after that are the V values that also apply to each 2 by 2 block.

Translating Y'UV420p to RGB is a more involved process compared to the previous formats. Lookup of the Y', U and V values can be done using the following method:

size.total = size.width \* size.height;

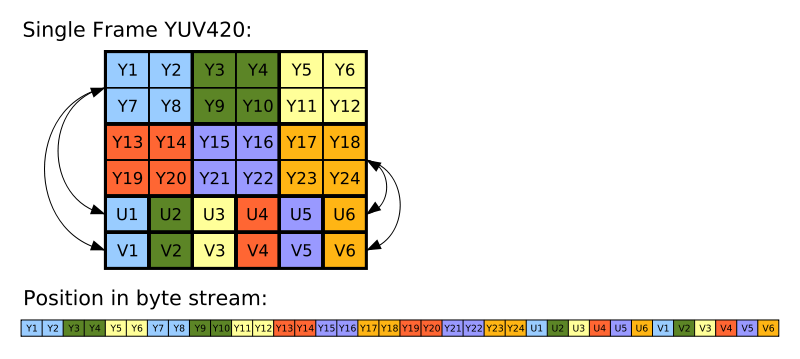
y = yuv[position.y \* size.width + position.x];

u = yuv[(position.y / 2) \* (size.width / 2) + (position.x / 2) + size.total];

v = yuv[(position.y / 2) \* (size.width / 2) + (position.x / 2) + size.total + (size.total / 4)];

rgb = Y'UV444toRGB888(y, u, v);

Here "/" is Div not division.

[](http://en.wikipedia.org/wiki/File:Yuv420.svg)

As shown in the above image, the Y', U and V components in Y'UV420 are encoded separately in sequential blocks. A Y' value is stored for every pixel, followed by a U value for each 2×2 square block of pixels, and finally a V value for each 2×2 block. Corresponding Y', U and V values are shown using the same color in the diagram above. Read line-by-line as a byte stream from a device, the Y' block would be found at position 0, the U block at position x×y (6×4 = 24 in this example) and the V block at position x×y + (x×y)/4 (here, 6×4 + (6×4)/4 = 30).

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=14)**] Y'UV420p (NV21) to ARGB8888 conversion**

This format (NV21) is the standard picture format on android camera preview. YUV 4:2:0 planar image, with 8 bit Y samples, followed by interleaved V/U plane with 8bit 2x2 subsampled chroma samples.[[6]](http://en.wikipedia.org/wiki/YUV#cite_note-6)

Java source code used on Android:

/\*\*

\* Converts YUV420 NV21 to ARGB8888

\*

\* @param data byte array on YUV420 NV21 format.

\* @param width pixels width

\* @param height pixels height

\* @return a ARGB8888 pixels int array. Where each int is a pixels ARGB.

\*/

public static int[] convertYUV420\_NV21toARGB8888(byte [] data, int width, int height) {

int size = width\*height;

int offset = size;

int[] pixels = new int[size];

int u, v, y1, y2, y3, y4;

// i along Y and the final pixels

// k along pixels U and V

for(int i=0, k=0; i < size; i+=2, k+=2) {

y1 = data[i ]&0xff;

y2 = data[i+1]&0xff;

y3 = data[width+i ]&0xff;

y4 = data[width+i+1]&0xff;

u = data[offset+k ]&0xff;

v = data[offset+k+1]&0xff;

u = u-128;

v = v-128;

pixels[i ] = convertYUVtoARGB(y1, u, v);

pixels[i+1] = convertYUVtoARGB(y2, u, v);

pixels[width+i ] = convertYUVtoARGB(y3, u, v);

pixels[width+i+1] = convertYUVtoARGB(y4, u, v);

if (i!=0 && (i+2)%width==0)

i+=width;

}

return pixels;

}

private static int convertYUVtoARGB(int y, int u, int v) {

int r,g,b;

r = y + (int)1.402f\*v;

g = y - (int)(0.344f\*u +0.714f\*v);

b = y + (int)1.772f\*u;

r = r>255? 255 : r<0 ? 0 : r;

g = g>255? 255 : g<0 ? 0 : g;

b = b>255? 255 : b<0 ? 0 : b;

return 0xff000000 | (r<<16) | (g<<8) | b;

}

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=15)**] References**

* 1. [**^**](http://en.wikipedia.org/wiki/YUV#cite_ref-1) Engineering Guideline EG 28, "Annotated Glossary of Essential Terms for Electronic Production," SMPTE, 1993.
  2. [**^**](http://en.wikipedia.org/wiki/YUV#cite_ref-2) Maller, Joe. [RGB and YUV Color](http://joemaller.com/fcp/fxscript_yuv_color.shtml), *FXScript Reference*
  3. [**^**](http://en.wikipedia.org/wiki/YUV#cite_ref-3) W. Wharton & D. Howorth, *Principles of Television Reception*, Pitman Publishing, 1971, pp 161-163
  4. [**^**](http://en.wikipedia.org/wiki/YUV#cite_ref-4) Keith Jack. *Video Demystified*. [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [1-878707-09-4](http://en.wikipedia.org/wiki/Special:BookSources/1-878707-09-4).
  5. [**^**](http://en.wikipedia.org/wiki/YUV#cite_ref-5) Poynton, Charles (1999-06-19). [*YUV and luminance considered harmful*](http://www.poynton.com/papers/YUV_and_luminance_harmful.html). <http://www.poynton.com/papers/YUV_and_luminance_harmful.html>. Retrieved 2008-08-22.
  6. [**^**](http://en.wikipedia.org/wiki/YUV#cite_ref-6) fourcc.com YUV pixel formas [[1]](http://www.fourcc.org/yuv.php#NV21)

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=YUV&action=edit&section=16)**] External links**

* [RGB/Y'UV Pixel Conversion](http://www.fourcc.org/fccyvrgb.php)
* [Explanation of many different formats in the Y'UV family](http://www.fourcc.org/yuv.php)
* Poynton, Charles. [Video engineering](http://www.poynton.com/Poynton-video-eng.html)
* Kohn, Mike. [Y'UV422 to RGB using SSE/Assembly](http://www.mikekohn.net/stuff/image_processing.php)
* [YUV, YCbCr, YPbPr color spaces.](http://discoverybiz.net/enu0/faq/faq_YUV_YCbCr_YPbPr.html)
* [Color formats](http://www.equasys.de/colorformat.html) for image and video processing - [Color conversion](http://www.equasys.de/colorconversion.html) between RGB, YUV, YCbCr and YPbPr.
* [How to convert RGB to YUV420P](http://www.erazer.org/how-to-convert-rgb-to-yuv420p)
* [libyuv](http://code.google.com/p/libyuv/)
* [pixfc-sse](http://code.google.com/p/pixfc-sse/) - C library of SSE-optimized color format conversions.