

Merging computing with studio video: Converting between $R'G'B'$ and 4:2:2

Charles Poynton
www.inforamp.net/~poynton

Abstract

In this paper, I explain the $R'G'B'$ and $Y'CbCr$ 4:2:2 representations, and explain the technical aspects of conversion between the two. I conclude by suggesting steps that can be taken during production and post-production to avoid difficulty with the conversion.

Film, video, and computer-generated imagery (CGI) all start with red, green, and blue (RGB) intensity components. In video and computer graphics, a nonlinear transfer function is applied to RGB intensities to give *gamma corrected* $R'G'B'$. This is the native color representation of video cameras, computer monitors, video monitors, and television.

The human visual system has poor color acuity. If $R'G'B'$ is transformed into luma and chroma, then color detail can be discarded without the viewer noticing. This enables a substantial saving in data capacity – in “bandwidth,” or in storage space. Because studio video equipment has historically operated near the limit of realtime recording, processing, and transmission capabilities, the subsampled $Y'CbCr$ 4:2:2 format has been the workhorse of studio video for more than a decade.

The disadvantage of 4:2:2 is its lossy compression. Upon conversion from 8-bit $R'G'B'$ to 8-bit $Y'CbCr$, three-quarters of the available colors are lost. Upon 4:2:2 subsampling, half the color detail is discarded. But production staff are facing increasing demands for quality, and increasing demands to integrate video production with film and CGI. The lossy compression of 4:2:2 is becoming a major disadvantage.

Owing to the enormous computing and storage capacity of general-purpose workstations, it is now practical to do production directly in $R'G'B'$ (or as it's known in studio video terminology, 4:4:4). To integrate traditional studio video equipment into the new digital studio, conversion between $R'G'B'$ and 4:2:2 is necessary.

Introduction

Linear-intensity RGB is the native color coding of CGI. In computing, the gamut of colors comprises the volume bounded by the unit RGB cube: See Figure 1 opposite. In video and computer graphics, a non-linear transfer function is applied to RGB intensities to give *gamma corrected* $R'G'B'$, often in 8 bits each. See Figure 2, on page 4.

If $R'G'B'$ is transformed into luma and color difference components, $Y'C_B C_R$, then color detail can be subsampled (lowpass filtered) without the viewer noticing. This leads to a substantial saving in data capacity – in “bandwidth,” or in storage space. Subsampling in $Y'C_B C_R$ involves a “visually lossless” lossy compression system. The 4:2:2 scheme has a compression ratio of 1.5:1, and the 4:2:0 and 4:1:1 schemes have compression ratios of 2:1. The subsampled $Y'C_B C_R$ 4:2:2 representation of Rec. 601 is standard in studio digital video. However, $Y'C_B C_R$ has several problems in the digital studio:

Recommendation ITU-R
BT.601-4, *Encoding Parameters
of Digital Television for Studios*
(Geneva: ITU, 1990)

Codeword utilization in $Y'C_B C_R$ is very poor. $R'G'B'$ coding with 8 bits per component allows every one of the 2^{24} combinations, or 16 million codewords, to represent a color. Theoretically, three quarters or more of the “legal” $Y'C_B C_R$ code combinations do not represent colors! In 8-bit Rec. 601 standard $Y'C_B C_R$, only 17% of the codewords represent colors. $Y'C_B C_R$ has fewer colors – or equivalently, more quantization noise, or poorer signal-to-noise ratio – than $R'G'B'$.

The designation *D-1* is sometimes loosely applied to 4:2:2. However, *D-1* properly refers to a particular DVTR format, not to an interface standard.

Filtering and subsampling operations that form the 4:2:2 signal remove chroma detail. If subsampling is accomplished by simply dropping or averaging alternate C_B and C_R samples, then filtering artifacts (such as aliasing) will be introduced. Artifacts can accumulate if filtering is repeated many times. Subsampling using a sophisticated filter gives much better results than simply dropping or averaging samples. However, even sophisticated filters can exhibit fringing on certain color edges, if conversion between $R'G'B'$ and 4:2:2 is repeated many times.

Loss of color detail makes it more difficult to pull bluescreen or greenscreen mattes from 4:2:2 than from $R'G'B'$.

Test signals characterize the electrical performance of a video system. Standard video test signals include elements that are synthesized electronically as sine waves, and injected onto the signal. Many of these elements have no legitimate $R'G'B'$ representation. Since these signals can be conveyed through $Y'C_B C_R$ without incident, some people claim that $Y'C_B C_R$ has an incremental advantage. However, in my opinion, it is more important to allocate bits to picture information than to signals that cannot possibly represent picture information.

In general, $Y'C_B C_R$ is optimized for realtime video, at the expense of more difficult interface with film, with CGI, and with general-purpose computer tools. $R'G'B'$ does not exploit chroma subsampling, so it has somewhat higher data capacity requirements than $Y'C_B C_R$.

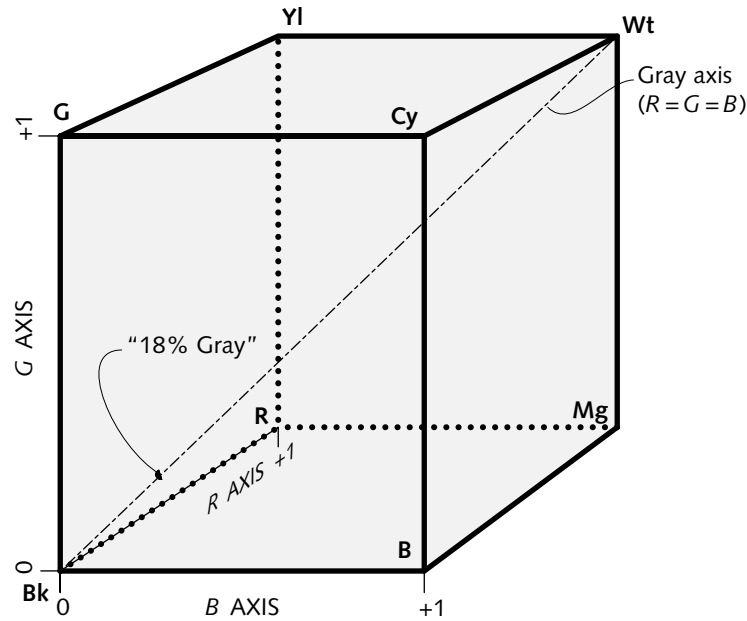
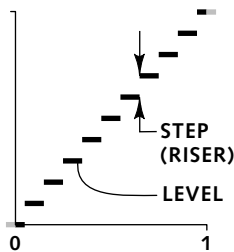


Figure 1 **RGB unit cube** encompasses linear-intensity coded RGB . This scheme is poorly matched to the lightness sensitivity of vision.

Computing gamut



Linear-intensity coding is used in CGI, where physical light is simulated. But linear intensity coding performs poorly for images to be viewed. The best perceptual use is made of the available bits by using nonlinear coding that mimics the nonlinear lightness response of human vision. In the storing and processing of images, linear-intensity coding is rarely used, and in the display of images, linear-intensity coding is never used. In video, computing, and many other domains, a nonlinear transfer function is applied to RGB intensities to give nonlinearly-coded (or *gamma corrected*) components, denoted with prime symbols: $R'G'B'$.

In an 8-bit system with nonlinear coding, each of R' , G' , and B' ranges from 0 through 255, inclusive. Each component has 255 *steps* (risers) and 256 *levels*: A total of 2^{24} colors – that is, 16777216 colors – are representable. Not all of them can be distinguished visually; not all are perceptually useful; but they're all colors. See Figure 2 overleaf.

$R'G'B'$ in video

Studio video $R'G'B'$ standards provide footroom below the black code, and headroom above the white code. The primary purpose of footroom and headroom is to accommodate the transients that result from filtering in either the analog or digital domains. Their secondary purpose is to provide some margin to handle level variations in signals originated in the analog domain. (Additionally, the headroom provides a marginal improvement in highlight handling and exposure latitude.)

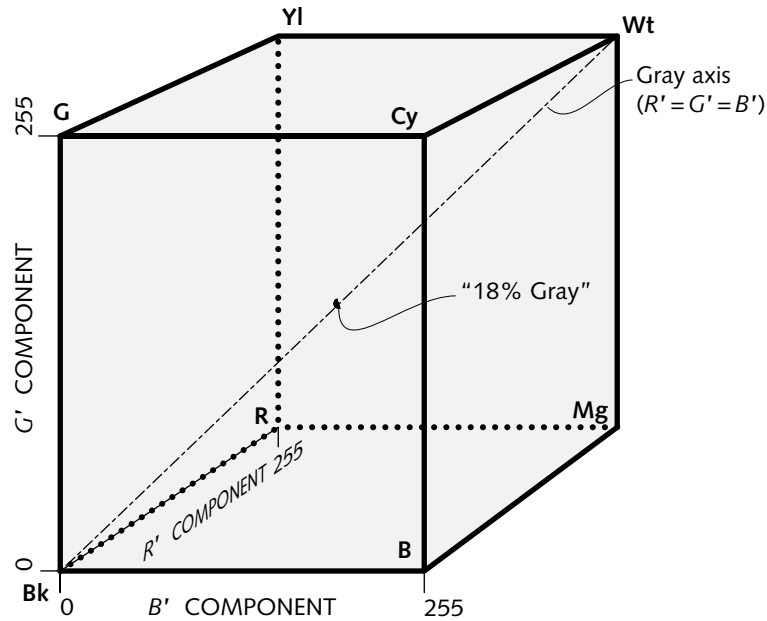
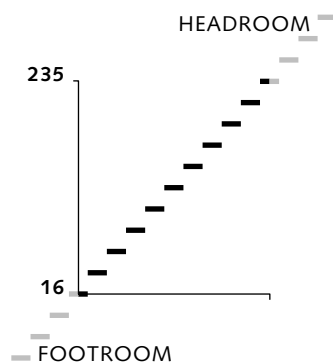


Figure 2 $R'G'B'$ cube represents nonlinear (gamma corrected) $R'G'B'$, typical of computer graphics. Though superficially similar to the RGB cube of Figure 1, it is dramatically different in practice, owing to its perceptual coding.



Charles Poynton, *Concerning "legal" and "valid" video signals*,
www.inforamp.net/~poynton

It is important to have fixed, idealized reference levels for black and white. Rec. 601 standardizes reference black at code 16 and reference white at code 235. I call this the *reference excursion*. Codes 0 and 255 are reserved for synchronization purposes, and are prohibited from appearing in video data. Codes 1 through 15 are reserved for footroom, and codes 236 through 254 are reserved for headroom. For no good technical reason, Rec. 601 assigns footroom and headroom asymmetrically: Footroom has 15 levels, but headroom has 19.

The so-called valid colors encompass the volume that is spanned when each $R'G'B'$ component ranges from reference black to reference white. In Rec. 601, each component has 219 steps (risers) – that is, 220 levels. That gives $220 \times 220 \times 220$, or 10648000 colors: About 64% of the total volume of codewords is valid.

You have seen that linear-intensity RGB is the basis for color representation in film and CGI, but that linear-intensity coding is a poor match to human perception. Greatly improved results are obtained by using nonlinear $R'G'B'$ coding that mimics the lightness sensitivity of vision. We can use another more subtle application of the properties of vision to code video signals: Vision has poor acuity to color detail, compared to its acuity for lightness. Providing that lightness detail is maintained, color detail can be discarded. Owing to the nature of the visual system, if this is done correctly, it will not be noticed. Subsampling involves two steps: First, a lightness component and two color components are formed. Then, detail is discarded from the two color components.

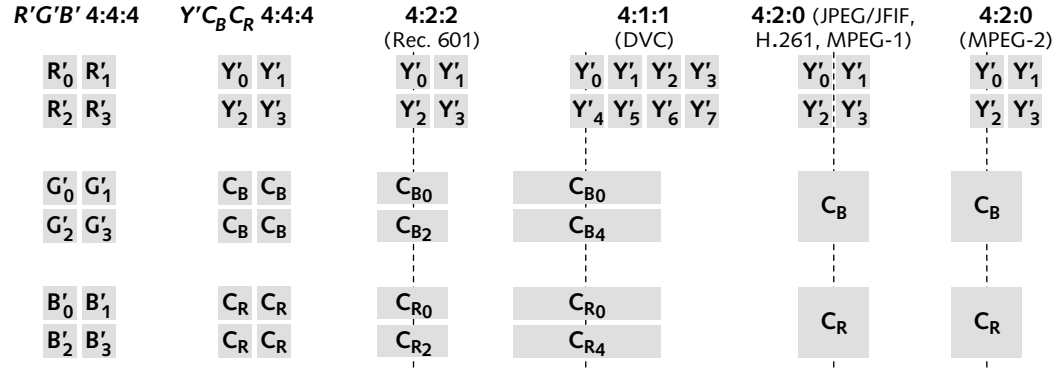


Figure 4 **Chroma subsampling**. A 2×2 array of $R'G'B'$ pixels is transformed into a luma component Y' , and two color difference components C_B and C_R scaled from $B'-Y'$ and $R'-Y'$ respectively. Color detail can be reduced by subsampling, provided that full luma detail is maintained. Here, the C_B and C_R samples are drawn wider or taller than the luma samples, to indicate their spatial extent. The horizontal offset of C_B and C_R from the center of the luma samples is due to cositing. (In 4:2:0 in JPEG/JFIF, MPEG-1, and H.261, the chroma samples are not cosited, but are sited *interstitially*.)

David Izraelevitz and Joshua L. Koslov, "Code Utilization for Component-coded Digital Video," in *Tomorrow's Television, Proceedings of 16th Annual SMPTE Television Conference* (Scarsdale, New York: SMPTE, 1982), 22–30

The color difference components are bipolar. Unscaled, they range from roughly -1 to $+1$. If you are an analog engineer, the doubled excursion represents a 6 dB signal-to-noise ratio (SNR) penalty for the chroma components. If you are a digital engineer, consider the sign to consume an extra bit in each of C_B and C_R . This codeword utilization issue represents a serious limitation of 8-bit $Y'CbCr$ performance. It necessitates techniques such as Quantel's patented *dynamic rounding*®.

In addition to this obvious problem of codeword utilization, transforms between $Y'CbCr$ and $R'G'B'$ must employ carefully-chosen matrix coefficients. If the product of the encoding matrix and the decoding matrix is not very nearly an identity matrix, then roundoff errors will accumulate every time an image is transcoded. High end manufacturers take great care in choosing these matrix coefficients. But the entire problem is circumvented by operating in $R'G'B'$.

Chroma subsampling

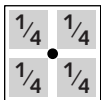
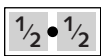


Figure 5 **Interstitial filters** for subsampling may be implemented using simple averaging. The rectangular outline indicates the subsampled $Y'CbCr$ block; the black dot suggests the effective siting of the computed chroma sample.

Once color difference components have been formed, they can be subsampled (filtered). In fact, the data compression that results from subsampling is the justification for using $Y'CbCr$ in the first place! To subsample by simply dropping samples leads to aliasing, and consequent poor image quality. It is necessary to perform some sort of averaging operation. The various subsampling schemes in use are sketched in Figure 4 above.

To subsample to 4:2:2 with minimum computation, some systems simply average two adjacent C_B and C_R samples. This described in filter theory as having weights $[\frac{1}{2}, \frac{1}{2}]$; this filter has the same normalized response as a $[1, 1]$ -filter. A comparable approach to generate 4:2:0 is to average C_B over a 2×2 block, and average C_R over the 2×2 block. That technique is ubiquitous in JPEG/JFIF stillframes in computing. Low-end systems simply replicate the 4:2:2 (or 4:2:0) C_B and C_R to obtain the missing chroma samples, prior to conversion back to $R'G'B'$.



Figure 6 **Cosited filters** for subsampling use weights that cause the computed chroma sample to be located exactly on the site of a luma sample.

Simple averaging causes subsampled chroma to take an effective position halfway between two luma samples, what I call *interstitial* siting, not the cosited position standardized for studio video. The expedient arithmetic is the reason that 4:2:0 subsampling in JPEG/JFIF, H.263, and MPEG-1 differs from 4:2:0 subsampling in MPEG-2.

Weights of [1, 2, 1] can be used to achieve cositing as required by Rec. 601, while still using simple computation. That filter can be combined with [1, 1] vertical averaging, so as to be extended to 4:2:0.

Simple averaging filters exhibit poor image quality. Providing the weights are carefully chosen, a filter combining a large number of samples – that is, a filter with a larger number of *taps* – will always perform better than a filter with a smaller number of taps. (This fact is not intuitive, because high frequency information is only apparent across a small scale.) High-end digital video and film equipment uses sophisticated subsampling filters, where the subsampled C_B and C_R of a 2×1 pair or 2×2 quad take contributions from many surrounding samples.

Sample aspect ratio, “square pixels”

In computing, it is a *de facto* standard to have samples equally-spaced horizontally and vertically (“square pixels”). In conventional video, various sample aspect ratios are in use: Sample aspect ratios differ between 525/59.94 and 625/50, and neither has equally-spaced samples. In HDTV, thankfully, square pixels have been adopted.

In certain adaptations of $Y'C_B C_R$ for film, the nonsquare sample aspect ratio of conventional 625/50 video has been maintained. This forces a resampling operation when that imagery is imported into the CGI environment, and another resampling operation when it is exported. If resampling is done well, it is intrinsically expensive. If resampling is done poorly, or done often (in tandem), it introduces artifacts.

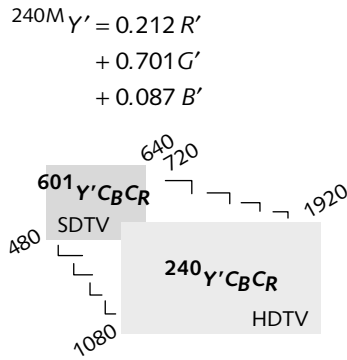
$R'G'B'$ and $Y'C_B C_R$ characterization

Charles Poynton, “The rehabilitation of *gamma*,” in *Human Vision and Electronic Imaging III*, Proc. SPIE/IS&T Conf. 3299, ed. B.E. Rogowitz and T.N. Pappas (Bellingham, Wash.: SPIE, 1998)

$R'G'B'$ is completely characterized by four technical parameters: white point, primary chromaticities, transfer function, and coding range. (A fifth *rendering intent* parameter is implicit; see my SPIE/IS&T paper.)

White point, primary chromaticities, and transfer function are all standardized by Rec. 709. The parameters of Rec. 709 closely represent current practice in video and in computing. We have, in effect, reached worldwide consensus on $R'G'B'$ coding. This is highly significant.

Coding range in computing has a *de facto* standard excursion, 0 to 255. Studio video accommodates footroom and headroom; its range is standardized from 16 to 235. (Modifications to Rec. 709 have been proposed to achieve wide-gamut operation in HDTV.)



$Y'CbCr$ is characterized by all of the parameters of $R'G'B'$, plus a set of luma coefficients. The coefficients of Rec. 601 are ubiquitous in conventional 525/59.94 video, 625/50 video, and computing. But according to recently-adopted SMPTE and ATSC standards, ATV and HDTV will use a new, different set: the luma coefficients of SMPTE 240M. This introduces a huge problem: There will be one flavor of $Y'CbCr$ for small (SDTV) pictures, and another for big (HDTV) pictures. $Y'CbCr$ data cannot be accurately exchanged between these flavors of coding without undergoing a mathematical transform of comparable complexity – and comparable susceptibility to artifacts – as resampling for the correction of pixel aspect ratio. (If the mathematical transform is not performed, then dramatic color errors result.)

Black level

When scene elements are combined, a discrepancy between black levels will be immediately objectionable to the viewer. (A difference in white levels of similar magnitude is unlikely to be noticed.) It is good operational practice to establish accurate black level when an element – whether $R'G'B'$ or $Y'CbCr$ – is brought into the digital environment.

Production and post-production houses occasionally exploit the foot-room of studio standards to convey “out of band” information sometimes known as *superblack*. Superblack can be an effective solution to certain production problems. But generally, using superblack compromises your freedom to work with material in the digital domain, and especially in the CGI domain.

$R'G'B'$ equipment that operates in native CGI coding generally cannot accommodate superblack at all. $Y'CbCr$ equipment is tolerant of incorrect black levels, but many image processing operations are liable to fail on data containing superblack.

Practical suggestions

To maximize performance at the interface of computing and video, I recommend that you take these steps:

Acquire $R'G'B'$ 4:4:4 images wherever possible, instead of acquiring images that have already been subjected to the $Y'CbCr$ transform and 4:2:2 subsampling. For realtime transfer, use the dual SDI link.

Stay in $R'G'B'$ if your production situation permits. The first conversion to $Y'CbCr$ will cause an unrecoverable loss of 75% of the available $R'G'B'$ codewords, and the first subsampling to 4:2:2 will cause an unrecoverable loss of half the color detail.

Avoid repeated conversions back and forth between $R'G'B'$ and 4:2:2. Conversions subsequent to the first are liable to accumulate rounding errors, and are liable to accumulate filtering artifacts such as aliasing.

Retain intermediates in R'G'B' 4:4:4 format where possible. Use DLT or Exabyte computer media, instead of videotape. Where intermediate or archival work must be recorded on video equipment, use 10-bit D-5 recording, instead of 8-bit D-1.

Minimize resampling. To the extent possible, avoid changing from one sample structure to another – for example, from square pixels to nonsquare, or from nonsquare to square.

Establish and maintain accurate black levels. Establish the correct black level for a scene or an element upon entry to the digital domain. When possible, perform this adjustment using video playback equipment. (Establishing and maintaining white level is not quite so important.)

Charles Poynton

Charles Poynton graduated with a degree in Mathematics from Queen's University in Kingston, Ontario. Subsequently, he studied at the Ontario College of Art in Toronto. He joined the faculty at OCA, where he taught electronics to art students and learned about video.

From 1981 to 1988, Mr. Poynton was principal of Poynton Vector Corporation. He and his colleagues designed and built the digital video equipment used by NASA's Johnson Space Center to convert video from the space shuttle into NTSC for recording and distribution.

While at Sun Microsystems in California, from 1988 to 1995, Mr. Poynton initiated Sun's HDTV research project, and introduced color management technology to Sun.

Mr. Poynton is a Fellow of the Society of Motion Picture and Television Engineers (SMPTE). He has contributed to many SMPTE and ITU-R standards, including Rec. 601, SMPTE RP 145, SMPTE 170M, and Rec. 709. He was the document editor for SMPTE 274M, the foundation for all of SMPTE's HDTV studio standards. In 1994, he was awarded the Society's David Sarnoff Gold Medal for his work to integrate video technology with computing and communications, including his work to establish square pixels as the basis for HDTV standards.

Mr. Poynton's book, *A Technical Introduction to Digital Video*, was published in 1996 by John Wiley, and is now in its second printing.

He now works as an independent contractor, to integrate video technology – particularly digital video, high definition television (HDTV) and accurate color reproduction – into computer workstations.