Discreet Logic

10, Rue Duke
Montréal, Québec
Canada H3C 2L7
Tel: (514) 393-1616
Fax: (514) 393-0110

www.discreet.com

Merging computing with studio video: Converting between *R′G′B′* and 4:2:2

Charles Poynton
www.poynton.com

Abstract

In this paper, I explain the *R′G′B′* and $Y'C_BC_R$ 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*) tristimulus components proportional to intensity – "linear light." A nonlinear transfer function is applied to *RGB* 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 capture, recording, processing, and transmission capabilities, the subsampled $Y'C_BC_R$ 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 "matrixing" from 8-bit *R′G′B′* to 8-bit $Y'C_BC_R$, three-quarters of the available colors are lost. Upon 4:2:2 subsampling, half the color detail is discarded. However, 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 is 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 light *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 nonlinear transfer function is applied to *RGB* tristimulus signals 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, $Y'C_BC_R$, 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 $Y'C_BC_R$ 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 $Y'C_BC_R$ 4:2:2 representation of Rec. 601 is standard in studio digital video. However, $Y'C_BC_R$ has several problems in the digital studio:

ITU-R Rec. BT.601, *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios* (Geneva: ITU).

Codeword utilization in $Y'C_BC_R$ 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, $^3/_4$ or more of the "legal" $Y'C_BC_R$ code combinations do not represent colors! In 8-bit Rec. 601 standard $Y'C_BC_R$, only 17% of the codewords represent colors. $Y'C_BC_R$ has fewer colors – or equivalently, more quantization noise, or poorer signal-to-noise ratio (SNR) – 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 $Y'C_BC_R$ without incident, some people claim $Y'C_BC_R$ to have an 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, $Y'C_BC_R$ is optimized for realtime video, at the expense of more difficult interface with film, CGI, and general-purpose computer tools. *R′G′B′* does not exploit chroma subsampling, so it has somewhat higher data capacity requirements than $Y'C_BC_R$.

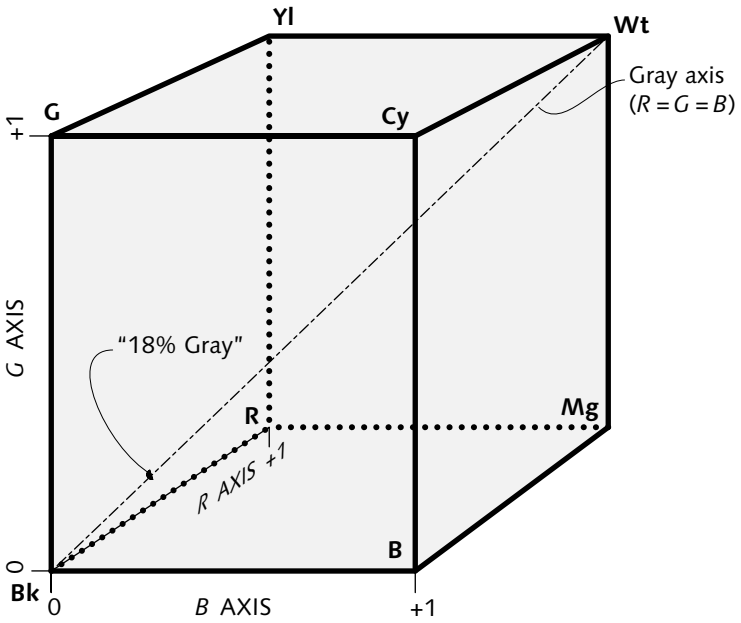
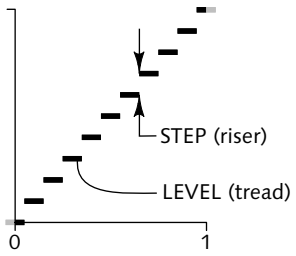


Figure 1 **RGB unit cube** encompasses linearly-coded *RGB* tristimulus values, each proportional to intensity. This scheme is poorly matched to the lightness sensitivity of vision.

Computing gamut



Linear light coding is used in CGI, where physical light is simulated. However, linear light 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 light coding is rarely used. In the display of images, linear light coding is never used. In video, computing, and many other domains, a nonlinear transfer function is applied to *RGB* tristimulus signals 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 are 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.)

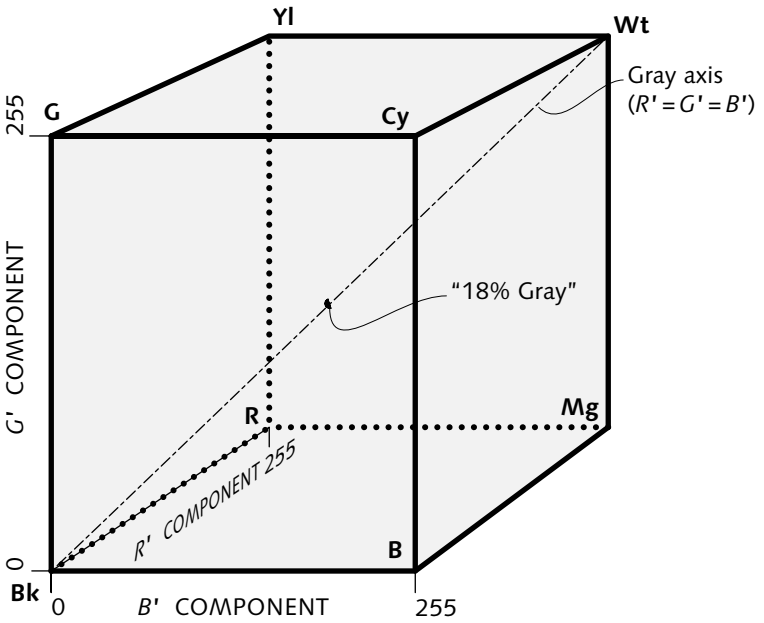
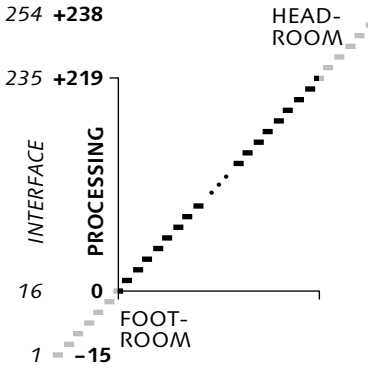


Figure 2 **R'G'B' cube** represents nonlinear (gamma corrected) *R'G'B'* typical of computer graphics. Though superficially similar to the *RGB* cube of Figure 1, it is dramatically different in practice owing to its perceptual coding.



Charles Poynton, *Concerning "legal" and "valid" video signals*, www.poynton.com

Eight-bit Rec. 601 coding has an excursion of 219 codes from black to white. For no good technical reason, footroom and headroom are assigned asymmetrically: Footroom has 15 levels, but headroom has 19. An offset of +16 is added at an 8-bit interface. (Hardware engineers say that black is at code 16, and white is at code 235.) The sketch in the margin shows abstract levels in bold, and hardware levels in italics. Interface codes 0 and 255 are reserved for synchronization purposes, and are prohibited from appearing in video or ancillary data.

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 220×220×220, or 10648000 colors: About 64% of the total volume of codewords is valid.

Linear light *RGB* is the basis for color representation in film and CGI, but linear light 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 subsampling is done correctly, it will not be noticed. Subsampling has two steps: First, a lightness component and two color components are formed. Then, detail is discarded from the two color components.

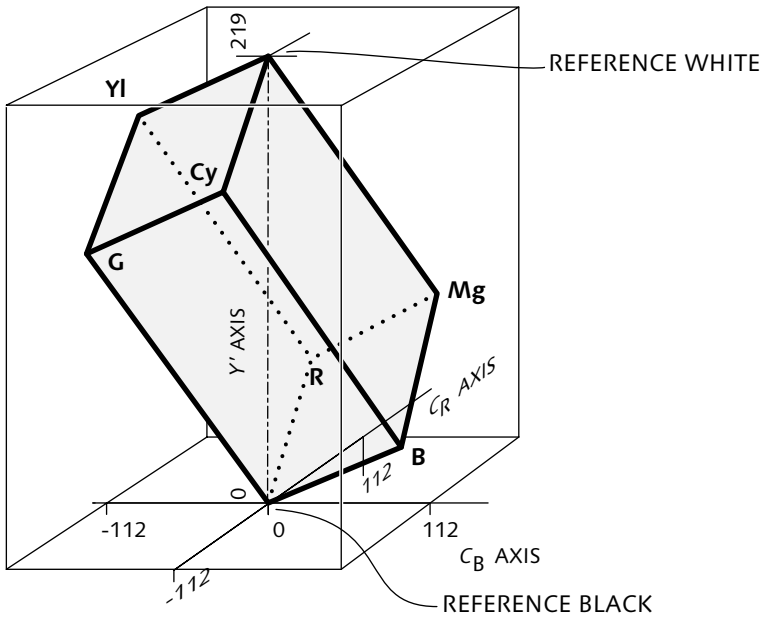


Figure 3 **$Y'C_BC_R$ cube** is formed when gamma-corrected *R′G′B′* are transformed to luma and chroma signals, which are then then scaled. Only about 1/4 of the available $Y'C_BC_R$ volume represents colors; the rest is wasted. This transform is performed before 4:2:2, 4:2:0, or 4:1:1 chroma subsampling.

$Y'C_BC_R$ video

To exploit the poor color acuity of vision, *luma* is formed as a properly-weighted sum of nonlinear *R′, G′,* and *B′*. It is standard to use the coefficients of Rec. 601. Two *color difference* – or *chroma* – components are then formed as *blue minus luma* and *red minus luma*, where blue, red, and luma incorporate gamma correction. (Luma, *B′–Y′*, and *R′–Y′* can be formed simultaneously from *R′*, *G′*, and *B′* through a 3×3 matrix multiplication.)

$$^{601}Y' = 0.299\,R' + 0.587\,G' + 0.114\,B'$$

Charles Poynton, *YUV and luminance considered harmful: A plea for precise terminology in video*, www.poynton.com

Various scale factors, and various notations, are applied to the basic *B′–Y′* and *R′-Y′* color differences. The correct scaling and nomenclature for component digital systems is $Y'C_BC_R$ (not *YUV*). The correct term for the lightness component is *luma* (not *luminance*).

If each of the *Y′*, C_B, and C_R components has 8 bits of precision, then obviously the entire $Y'C_BC_R$ cube has the same number of codewords as 8-bit *R′G′B′*. However, it is immediately obvious from the appearance of the transformed *R′G′B′* unit cube in Figure 3 above that only a small fraction of the total volume of the $Y'C_BC_R$ coordinate space is occupied by colors! The number of colors accommodated is computed as the determinant of the transform matrix. In Rec. 601 $Y'C_BC_R$, only about 1/4 of the Rec. 601 studio video *R′G′B′* codes are used.

$$\frac{\frac{1}{4}\cdot 220\cdot 225^2}{220^3} = \frac{2784375}{10648000} = 0.261$$

Of the 16.7 million colors available in studio *R′G′B′*, only about 2.75 million are available in $Y'C_BC_R$. If *R′G′B′* is transcoded to $Y'C_BC_R$,

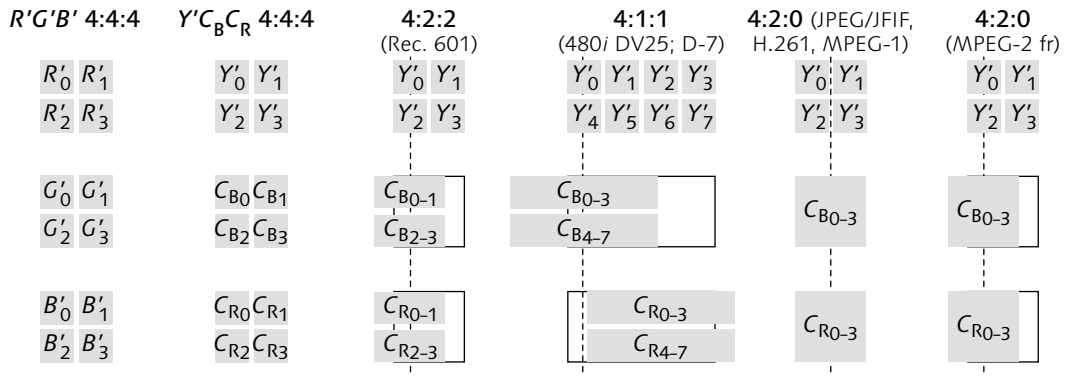


Figure 4 **Chroma subsampling.** Providing full luma detail is maintained, vision's poor color acuity enables color detail to be reduced by subsampling. A 2×2 array of *R′G′B′* pixels is *matrixed* to a luma component *Y′* and color difference (*chroma*) components C_B and C_R. C_B and C_R are then filtered (averaged). Here, 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 is due to cositing. (In 4:2:0 in JPEG/JFIF, MPEG-1, and H.261, chroma samples are sited *interstitially*, not cosited.)

then transcoded back to *R′G′B′*, the resulting *R′G′B′* cannot have any more than 2.75 million colors!

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

The color difference components are bipolar. Unscaled, they range from roughly –1 to +1. For analog engineers, the doubled excursion represents a 6 dB SNR penalty for the chroma components. Digital engineers should 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 $Y'C_BC_R$ performance. It necessitates techniques such as Quantel's patented *dynamic rounding*®.

In addition to this obvious problem of codeword utilization, transforms between $Y'C_BC_R$ and *R′G′B′* must have 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; however, the entire problem is circumvented by operating in *R′G′B′*.

Chroma subsampling

Once color difference components have been formed, they can be subsampled (filtered). The data compression that results from subsampling is the justification for using $Y'C_BC_R$ 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.

Some systems implement 4:2:0 subsampling with minimum computation by simply averaging C_B over a 2×2 block, and averaging C_R over the same 2×2 block. Simple averaging causes subsampled chroma to take an effective position centered among a 2×2 block of luma

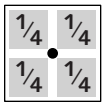


Figure 6 **Interstitial 4:2:0 filter** for subsampling may be implemented using simple averaging. The rectangular outline indicates the subsampled $Y'C_BC_R$ block; the black dot suggests the effective siting of the computed chroma sample.

samples, what I call *interstitial* siting. Low-end decoders simply replicate the subsampled 4:2:0 C_B and C_R to obtain the missing chroma samples, prior to conversion back to *R′G′B′*. This technique, sketched in Figure 6 in the margin, is used in JPEG/JFIF stillframes in computing, MPEG-1, and ITU-R Rec. H.261 videoconferencing.

Simple averaging causes subsampled chroma to take an effective position halfway between two luma samples, what I call *interstitial* siting. This approach is inconsistent with standards for studio video and MPEG-2, where C_B and C_R are *cosited* horizontally.

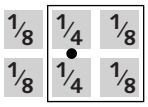


Figure 7 **Cosited filters** for subsampling use weights that cause each computed chroma sample to be horizontally aligned with a luma sample.

Weights of [1/4, 1/2, 1/4] can be used to achieve horizontal cositing as required by Rec. 601, while still using simple computation, as sketched at the top of Figure 7 in the margin. A [1/4, 1/2, 1/4] filter can be combined with [1/2, 1/2] vertical averaging, so as to be extended to 4:2:0 used in MPEG-2, as sketched at the bottom of Figure 7.

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 in 4:2:2, or 2×2 quad of 4:2:0, take contributions from many surrounding samples.

Sample aspect ratio, "square sampling"

In computing, it is a *de facto* standard to have samples equally-spaced horizontally and vertically ("square sampling"). 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 high-definition television (HDTV), thankfully, square sampling has been adopted.

In certain adaptations of $Y'C_BC_R$ 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_BC_R characterization

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.)

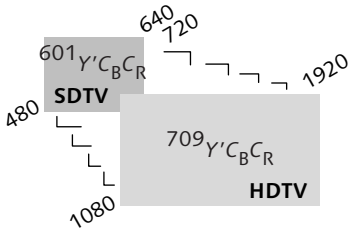
Charles Poynton, "The rehabilitation of *gamma*," in *Human Vision and Electronic Imaging III*, Proc. SPIE/IS&T Conf. 3299, ed. B.E. Rogowitz and T.N. Pappas (Bellingham, Wash.: SPIE, 1998).

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. (In ITU-R Rec. BT.1361, the coding range of Rec. 709 is extended to achieve a wider gamut.)

$$^{709}Y' = 0.2126R' + 0.7122G' + 0.0722B'$$

$Y'C_BC_R$ 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 Advanced Television Systems Committee (ATSC) standards, HDTV will use a new, different set: the luma coefficients of Rec. 709. This introduces a huge problem: There will be one flavor of $Y'C_BC_R$ for small, standard-definition television (SDTV) pictures, and another for big (HDTV) pictures. $Y'C_BC_R$ 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.)



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 already subjected to the $Y'C_BC_R$ 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 $Y'C_BC_R$ 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 after 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 sampling 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.)