Issues on YJK colour model implemented in Yamaha V9958 VDP chip

by Ricardo Cancho Niemietz (Oct 24, 2014)

Analyst-programmer with experience
in digital image processing

Rationale

The third generation of MSX compatible personal home computers, known as MSX2+ and launched in 1988, were shipped with (then) new graphic controller chip or VDP (Video Display Processor) developed by Yamaha Corp., labelled V9958 in their catalogue. That chip features new special graphic modes (known as Screen 10, Screen 11 and Screen 12 in MSX BASIC) which store pixel colours in a proprietary colour model known as YJK.

YJK is, in principle, a luminance-chrominance colour model, storing the luminance part (Y) within 5 bits of depth, in 32 levels with values ranging from 0 (minimum) to 31 (maximum), and the chrominance part (here, J and K) with 6 bits of depth each, in 64 levels with values ranging from –32 to 31 (two's complement signed integer logic).

The V9958 also features a DAC (Digital-to-Analog Converter) to manage RGB colour signals with 5 bits per component (15 bits overall), in 32 levels with values ranging from 0 to 31 each, thus allowing up to 32^3, i.e. 32,768 different combinations, or colours.

Designers implemented V9958's internal logic to convert from, YJK stored in the video frame buffer in these new special modes, to 15-bit RGB, in order to use its DAC to provide output colour video signal, allowing up to 19,268 different colours at a time on screen. *But they made a couple of unintentional mistakes in their computations*, thus making the new video modes with thousands of colours simultaneously on screen (the first machines with V9958 in history to do so among the home computer segment of the date) almost useless for professional, or even semi-professional, digital video and photo processing.

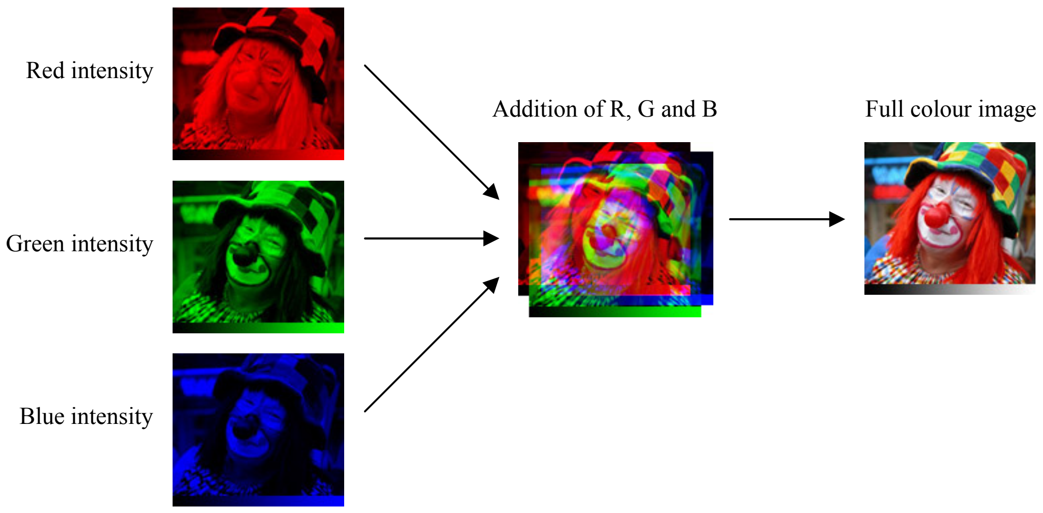
In this paper, the author explains the flaws of design of the YJK colour model, as it was finally implemented in the V9958.

1. Introduction to human colour perception and colour models

The non-colour blind human eye perceives colour in a very complex way, but it can be tricked to "see" any given colour as appropriate combination of at least *three* stimuli reaching the retina. These three stimuli are usually light in three different monochrome frequencies, given they are at some distance from each other in the electromagnetic spectrum and representing short, middle and long wavelengths. This is the tristimulus [1] model of the human perception of colour, and was developed for the first time by James Clerk Maxwell (1831 – 1879) in 1855.

Over a century of research in human colour perception has demonstrated that three very convenient stimuli are monochrome light in frequencies of red (R), green (G) and blue (B). This is the base of the so-called RGB colour model, and also, this is why there are many colour electronic display devices based on RGB model in industry, like TV sets and computer monitors.

The RGB model is an additive colour model in which absence of stimulus represents no light (black), and superimposing the maximum level of the three R, G and B components represents full light (white), being any intermediate combination of levels for every possible RGB triplet (r,g,b) a different colour.



Schematic representation of the additive RGB model. Notice that addition of R, G and B scales does give grayscale.

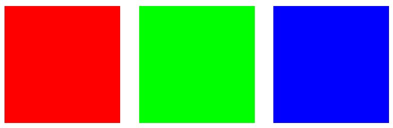
In the RGB colour model, numerically each component ranges from 0 (no intensity) to a given convenient practical maximum (here, *max*). Colours described with lower RGB values are darker than those described with higher ones. When the values for the three RGB components are the same, i.e. $r = g = b$, the described colour is a shade of grey, a *neutral* colour that bears neither hue nor tint.

The Yamaha V9958's DAC employs 5 bits per RGB component, so it has quantized, 32 levels of intensity per component ($max = 31$). Then, it gives 32^3, or 32,768 different combinations to values of RGB triplets.

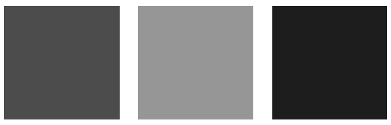
But the human eye is more sensitive to the stimulus of *bright* than of the *colour*, due to physiological differences in receptors (rod cells and cone cells of the retina, respectively). Brightness is the way we perceive certain amount of light, and it makes the difference between "dark" and "light". By convention, we call the absence of light being "black", and given maximum of light being "white". Brightness is the amount of bright that a given colour has as our vision perceives it. So, to us, some colours are darker or lighter than others.

To the human eye, each of the R, G and B component of the RGB model has its own intrinsic brightness, being green the lighter and blue the darker, and red the (approx.) midpoint. In rough numbers, pure red (in RGB, $r = max$, g = 0, b = 0) is about 30% of the brightness of the white, pure green ($r = 0$, g = *max*, b = 0) is about 59% of the brightness of the white ($r = 0$, g = 0, b = *max*) is about 11% of the brightness of the white.

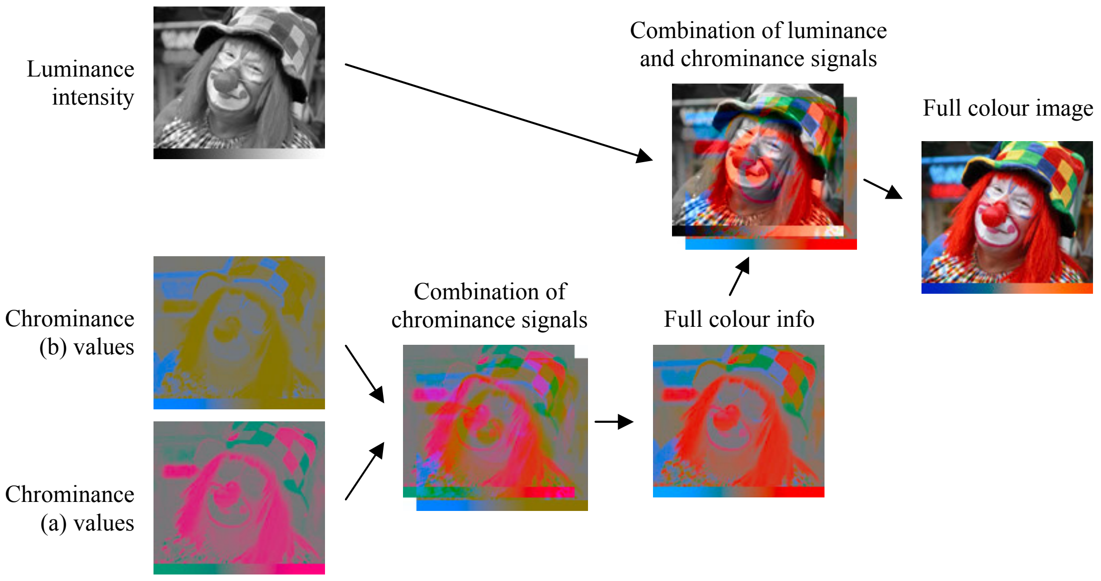
In the following three colour patches, it is clearly seen that the green patch at centre is lighter than the blue patch at right:



Luminance, in a broad sense, is the measure of the relative brightness or luminous intensity a given colour has, and is usually designated with the symbol Y. Luminance carries no information about hue or tint (*frequency* of light), but only that of brightness (*intensity* of light); it is indeed *achromatic*. Here are the shades of grey corresponding to luminance of the R, G and B colour patches shown above:



For a given colour, information about its hue can be carried in separate signals, called collectively as chrominance, then can be combined with its luminance to represent the complete colour. And following the tristimulus principle, luminance being one stimulus (intensity), chrominance must be carried with two signals, that is, two stimuli combined to represent frequency and saturation together. They conform to luminance-chrominance colour models to describe colours.



Schematic representation of generic luminance-chrominance model. Notice that combination of three scales does not give grayscale. Grayscale is luminance scale only.

There are a number of standard luminance-chrominance colour models, for example:

- YUV used in PAL colour TV;
- YIQ used in NTSC colour TV;
- YDbDr used in SECAM colour TV;
- YPbPr used in colour component video. When used in digital form, it is known as YCbCr:
- CIE L*a*b, used in Adobe Photoshop as internal colour format.

As far as human eye is less sensitive to chrominance that is to luminance, chrominance signals can be carried using much less bandwidth than that of luminance. For example, typical image data lossy compression formats [2], as JPEG and MPEG, convert colour from RGB to YCbCr as first step in order to manage luminance separately from chrominance, applying more severe loss to the latter than to former.

Numerically, luminance ranges from 0 (black) to a convenient maximum value (which represents white). It can be computed from the RGB description of the colour assigning a different weight to each component, the higher to the green, the lower to the blue, and in that case the maximum value for luminance is the same that of the RGB components. Incidentally, the shades of grey have a luminance value Y very close to each of their R, G and B independent components, so for practical purposes in this cases $y = r = g = b$.

In the aforementioned standard luminance-chrominance colour models, typical weights to compute luminance Y from RGB values of a given colour are about 0.299 for red, 0.587 for green and 0.114 for blue, and the formula is:

$$y = 0.299r + 0.587g + 0.114b$$

Chrominance is represented by a pair of signed integer values that range from a minimum, negative value (here *min*) to a maximum, positive value (here *max*). Usually, this range is the same for both values and symmetric (that is $min = -max$). Shades of grey have both values set at 0, and tinted colours have at least one of the chrominance values different than 0.

2. Description of the YJK colour model

The YJK colour model of the Yamaha V9958 [3] is conceived as a proprietary luminance-chrominance colour model. The Y component is defined in YJK, using 5 bits per RGB component, as being:

$$y = (r / 4) + (g / 8) + (b / 2) = 0.25r + 0.125g + 0.5b$$

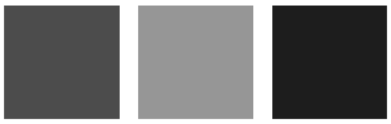
But this Y value *cannot be* luminance due to the weight assigned to the green component G, 0.125, which is lower than that assigned to the blue component B, 0.5. That is, as if green were intrinsically *darker* than blue, which is *not* true: in real world, green is *lighter* that blue, as we see above.

Comparing the magnitudes of the weights assigned to G and B components in both true Y luminance and Y in YJK, we see:

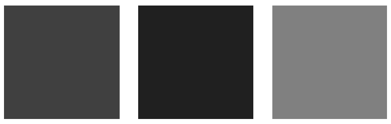
	Y luminance	YJK model
G	0.587	0.125
B	0.114	0.5

It seems as if weights for green and blue were *swapped*: being the correct about 0.5... for green and 0.1... for blue, in YJK the values are reversed: 0.1... for green and 0.5... for blue.

Maybe designers of V9958 unintentionally implemented the computation of Y with weight values for G and B exchanged. Thus, that Y in YJK colour model is *not* proper luminance. Compare the true luminance greys corresponding with R, G and B already shown in the previous section:



with that of Y resulting of the YJK formula:



Clearly, the intensity of central and rightmost patches does not match in both cases, they appear to be exchanged: that of green is the darker when it should be the lighter.

The computation of the chrominance part of YJK, J and K, from a 15 bit RGB colour is quite simple once Y has been calculated:

$$j = r - y$$

$$k = g - y$$

Value of Y not being true luminance in YJK coding does not make the YJK useless to encode colours. Simply, the Y part must be interpreted as one “component” of the encoded triplet, not “luminance”.

The algorithm is fully reversible, so we can get RGB values again from YJK this way:

$$r = j + y$$

$$g = k + y$$

$$b = (5y / 4) - (j / 2) - (k / 4) = 1.25y - 0.5j - 0.25k$$

And that way it is implemented in V9958, so that Y in YJK was not true luminance is not a problem *per se*. The problem is how it is used when reading pixel values from the video fame buffer's memory.

3. Use of YJK colour model with the Screen 12 video mode of the MSX2+

The Yamaha V9958 encodes the J and K parts of the YJK colour as 6 bits signed integer (two's complement) values, so 12 bits are needed for a single colour. Thus, a single pixel would need 17 bits, counting 5 bits for the Y component part, 6 bits for the J part and another 6 bits for the K part.

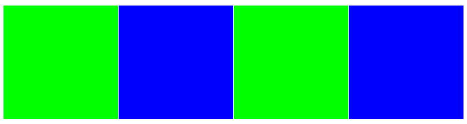
In order to save memory, Screen 12 in MSX2+ devotes one single byte to a single pixel (as Screen 8 with *only* 256 colours has). This byte has 5 bits of the Y part of the pixel's colour in YJK format, and 3 bits left available. Every four consecutive pixels sum 12 bits aside of Y parts, so they can store the J and K part of *one* colour. But what colour? There are four pixels, that is, four Y values but only a JK pair, so the four pixels *must share* the same chrominance values. But chrominance represents hue, tint, so in Screen 12, YJK mode, every four consecutive pixels must have *the same* hue or tint, as if they were packed as $\{Y_1,Y_2,Y_3,Y_4,J,K\}$ instead of $\{Y_1,J_1,K_1\}$, $\{Y_2,J_2,K_2\}$, $\{Y_3,J_3,K_3\}$ and $\{Y_4,J_4,K_4\}$.

This technique presents the problem. While having a shared chrominance (hue, tint) for every four consecutive pixels is severe limitation, it is tolerable due the secondary role of chrominance for the human eye, but then it is mandatory that the Y parts of the four pixels *must represent luminance* to achieve the desired effect. In typical lossy compression formats as JPEG the chrominance part can be, and usually is, shared among several adjacent pixels, being the most prevalent two (in a block of 2×1) to four (in a block of 2×2); this technique is called "chrominance subsampling" and also helps to reduce the amount of data of the compressed file. But with the YCbCr colour model used in JPEG the Y part is true luminance, so it is not a problem at all.

But it is not the case with YJK. As we saw before, Y part in YJK definition is *not* true luminance; it would be if weights for G and B components in computing Y were reversed, but it is not the case.

As J and K must be shared between every four adjacent pixels, one can compute the common J and K values from the *average* of the original RGB of those four pixels to obtain the subsampled chrominance. Then one must compute the individual *luminance* to achieve the effect to be as closest as possible to the original four pixels' colours. But in calculating the Y part of the YJK colours, we get no luminance *but another thing*, thus ruining the effect when greenish and bluish and related colours are used.

Let's see it with an example, with the following four adjacent original RGB pixels green, blue, green and blue:



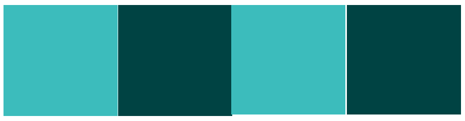
Their average 15-bit RGB values are:

$$r' = (r_1 + r_2 + r_3 + r_4) / 4 = (0 + 0 + 0 + 0) / 4 = 0$$

$$g' = (g_1 + g_2 + g_3 + g_4) / 4 = (31 + 0 + 31 + 0) / 4 = 15.5 \ (\rightarrow 15)$$

$$b' = (b_1 + b_2 + b_3 + b_4) / 4 = (0 + 31 + 0 + 31) / 4 = 15.5 \ (\rightarrow 15)$$

These r′, g′ and b′ values represent dark cyan. If combined with the *luminance* (intrinsic brightness) of green and blue, it would give:



which, given the limitations of the Screen 12 format, could be regarded as a "good" compromise: both intensity and tint are not the original ones, but they do not severely deviate from the original appearance.

But this is *not* what really happens when using YJK colour model. First, we convert the RGB average values to YJK to get common J and K values:

$$y' = (r' / 4) + (g' / 8) + (b' / 2) = 0 + 1.9375 + 7.75 = 9.6875$$

$$j = r' - y' = 0 - 9.6875 = -9.6875 \ (\rightarrow -9)$$

$$k = g' - y' = 15.5 - 9.6875 = 5.8125 \ (\rightarrow 5)$$

Then we compute Y for the green and blue original pixels:

$$y_1 = y_3 = (0 / 4) + (31 / 8) + (0 / 2) = 0 + 3.875 + 0 = 3.875 \ (\rightarrow 3)$$

$$y_2 = y_4 = (0 / 4) + (0 / 8) + (31 / 2) = 0 + 0 + 15.5 = 15.5 \ (\rightarrow 15)$$

Converting again from YJK to RGB:

$$r_1 = r_3 = j + y_1 = -9.6875 + 3.875 = -5.8125 \ (\rightarrow 0\text{, clipped})$$

$$g_1 = g_3 = k + y_1 = 5.8125 + 3.875 = 9.6875 \ (\rightarrow 9)$$

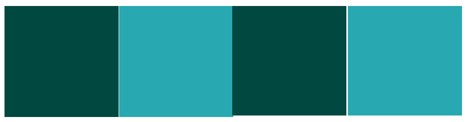
$$b_1 = b_3 = (5y_1 / 4) - (j / 2) - (k / 4) = 4.84375 - (-4.84375) - 1.453125 = 8.234375 \ (\rightarrow 8)$$

$$r_2 = r_4 = j + y_2 = -9.6875 + 15.5 = 5.8125 \ (\rightarrow 5)$$

$$g_2 = g_4 = k + y_2 = 5.8125 + 15.5 = 21.3125 \ (\rightarrow 21)$$

$$b_2 = b_4 = (5y_2 / 4) - (j / 2) - (k / 4) = 19.375 - (-4.84375) - 1.453125 = 22.765625 \ (\rightarrow 22)$$

And as they are shown on screen:



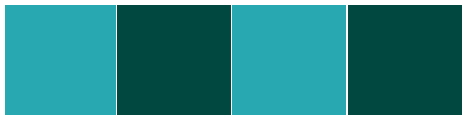
The hue, or tint, is almost the same as predicted, a dark cyan, but intensity *has been reversed*: lighter original pixels become darker and vice-versa.

This happens not only in that extreme case of pure green and blue, but also with every colour with RGB combinations involving values of G and B greater than zero: yellow- and magenta-like colours, along with those greenish and bluish are affected. The final appearance of the resulting image depends a lot on the colours of the original one, and pixels with higher values of G and B will look more distorted, so there will be some cases worse than others. But anyway, accurate colour presentations cannot be fully achieved in Screen 12 (and related Screen 10 and 11) video mode of the MSX2+.

4. Possible workaround

There are ways to circumvent, to a point, the fact that the Y part in the YJK colour is not true luminance information of that colour. Once computed the average RGB of each group of four adjacent pixels, and computed the common J and K components they share, software can choose their four individual Y components not by direct computation with the Y formula of the YJK regular algorithm, as we did in the previous section, but searching for the more appropriate values to match as much as possible original and final pixels. In other words, by *actively processing* every pixel before writing YJK values in video frame buffer's memory.

Thus, in the example given in the previous section, the program displaying the pixels would decide that the better matching is the actual reverse of Y for the green and blue pixels, giving a visual result more akin to the intended RGB original:



Of course it is not an ideal solution:

- It requires a great overhead of CPU usage, making many repetitive computations for every pixel of the screen every time a new and colourful image is displayed.
- YJK values stored in video memory can not be reverted again to RGB (to save a new image file, for example).

But it can be used to *pre-process* image files prior displaying them as static images in Screen 12 video mode, using temporary or *ad hoc* intermediate files (e.g. for a slide show).

In any case, the potential applications of this workaround may be limited to a narrow scope. While YJK was a breakthrough in terms of technology, described issues complicate its application for the real-time and professional use (e.g. by photographers). To date there is scarce software available which uses YJK, and no professional imaging software.

5. Other issues of YJK colour model

There is another caveat with the definition of Y in YJK: the V9958 stores Y values in 5 bits, 32 levels with values ranging from 0 to 31. But the Y formula gives no values above 27 at most, managing up to 28 levels only out of 32. Using the maximum value of 31 for each RGB component (white), we obtain:

$$y = (31 / 4) + (31 / 8) + (31 / 2) = 7.75 + 3.875 + 15.5 = 27.125 \ (\rightarrow 27)$$

from which the fractional part is discarded. So up to four levels of the full range are formally *wasted*. However V9958 can display values of 28 to 31 for the Y part that are stored in the video frame buffer. But resulting colours are *out of gamut* colours, that is, colours "out of" any of the RGB valid combinations, being *unsupported* by the Yamaha's YJK model. Out of gamut colours are not *impossible* colours, but colours impossible to be represented inside the valid range of the given colour YJK model, the space for valid combinations for the components within the formal limits for its values.

Probably Yamaha designers had chosen divisor values of 4, 8 and 2 in order to perform easier implementation of the YJK algorithm in the V9958's hardware, but in doing so they unintentionally downgraded the ability to manage wider range of colours than finally they implemented, plus adding unmanaged (and undesired) out-of-gamut colours.

Using 5 bits per RGB component, a better yet simple formula to compute in YJK the Y part from given RGB for a colour would be:

$$y = (2r / 7) + (4g / 7) + (b / 7)$$

which yields both a) a true luminance level, and b) values in the full range available, avoiding the problems explained in this paper.

Another limitation, by design, is that chrominance subsampling is done every four consecutive pixels. This produces undesirable and very noticeable colour artefacts in slanted borders of the images with sharp colour contrast, in natural images taken from the real world.

Instead of a 4×1 subsampling block it would be better a 2×2 block. This way, still using four pixels to share a single JK common pair, colour artefacts would be much less noticeable. On the other hand, arguably it would complicate the hardware design beyond reasonable limits, so the final implementation is understandable.

Final conclusions

It was a good idea to implement some extra video modes capable to display thousands of colours simultaneously to show realistic, real life images with a home computer, a very advanced and brave step at that time, by using techniques alike as those now known as “lossy compression”, by exploiting the nature of human colour perception and the unique characteristics of the luminance-chrominance models, however a couple of issues in design led to non-optimal, erroneous results, and ineffective graphical performance and colour accuracy.

The engineering around YJK was very advanced at its time, being a precursor of the concepts like luminance-chrominance separated management and chrominance subsampling, later widely used in lossy compression formats as JPEG, today’s mainstream standard for photographic imaging. It is a pity that bugs in the implementation broke down the good intentions. A machine that could be regarded as the first personal home computer capable being used in semi-professional video and photography fields, paving the way to the (then) future of digital imaging, but instead it was relegated to be *only* an obscure upgrade of the previous MSX2 generation in terms of graphics. The Screen 10, 11 and 12 video modes, new to MSX2+, with the flaws of design of YJK colour mode, were not as useful as the previous Screen 8 video mode of the MSX2, with the same resolution of 256×212 pixels and its 256 fixed out of 512 RGB colours.

Yamaha did their best to offer quasi-photographic video output for home computers at affordable price, and they devoted a great effort to develop YJK colour format and Screen 12 video modes, with great foundations. Only a couple of unintentional bugs prevented them to become a milestone in digital graphic processing's history and an acclaimed success in that field.

About the author

Ricardo Cancho Niemietz was born in 1969 in Madrid, Spain. He joined the biggest, in those years, professional videogame company of Spain, Erbe Software, in 1988, along with some other mates when he was only 18 years old. He collaborates in the production of titles for ZX Spectrum, Amstrad CPC, MSX, Commodore 64, PC-CGA, Atari ST and Commodore Amiga under the brand of Topo Soft, and later within the company Animagic, where he worked on some videogame portings to MSX. Later in 1990 he became recognized in digital image processing software area, using then state-of-the-art equipment running on PCs. Within next years, he created many applications to transmit, receive, capture, process, print, export, and document the professional press digital image files for the main Spanish press agency, Agencia EFE. Also, he designed and programmed a video codec for Windows 3.x known as SuperAVI. His activities in these fields lasted up to 2002.

Today, he works as analyst-programmer in fields like telecom, engineering and business management. He was an author and contributing editor for several English Wikipedia articles related to digital image technologies, especially focused on “classic” machines and techniques of the ’80s and the ’90s of the 20th century.

References

1. Wikipedia (2014) *CIE 1931 color space: tristimulus values*, available online at http://en.wikipedia.org/wiki/CIE_1931_color_space#Tristimulus_values (accessed 24-Oct-2014)
2. Wikipedia (2014) *Lossy compression*, available online at http://en.wikipedia.org/wiki/Lossy_compression (accessed on 24-Oct-2014)
3. Yamaha Corporation (1988) *Yamaha® LSI V9958 MSX-Video Technical Data Book*, available online at http://primrosebank.net/computers/mtx/techlib/design/V9958-Technical-manual_v1.0.pdf (accessed on 24-Oct-2014)

This article was edited by Eugeny Brychkov (2014)