Color space adaptation for video coding

Adrià Arrufat

Universitat Politècnica de Catalunya tutor: Josep Ramon Casas
Technicolor tutors: Philippe Bordes & Pierre Andrivon
Internship dates: 30-01-12 – 27-07-12

Contents

1 Introduction 4
1.1 The context 4
1.2 Objectives 4
2 State of the Art 5
2.1 Video Coding principles 5
2.1.1 HEVC: High Efficiency Video Coding 6
2.2 The role of gamma correction 7
2.2.1 Origins: human perception 7
2.2.2 Engineering trade-off: CRT screens 8
2.3 Color Spaces 9
2.3.1 RGB 9
2.3.2 Coding Color Spaces 10
2.4 Chroma downsampling 11
3 Digital video workflow 13
3.1 Pre-processing the sequence 13
3.1.1 Linearization: Gamma Correction 14
3.1.2 Down-sampling 15
3.1.3 Specific color transformations 16
3.1.4 Chroma down-sampling 20
3.2 Video coding and decoding 20
3.3 Post-processing the sequence 20
3.3.1 Chroma up-sampling 21
3.3.2 Specific Color Transforms 21
4 Adapting pictures representation to color space 22
5 Results 24
5.1 Assessing the chain without video coding 24
5.2 Lab with video coding 25
5.3 Y'CoCg and Y'FbFr 27
5.4 Content-adaptive color space with PCA 30
6 Conclusions 33
7 Further work 35

7.1 Perceptual color spaces 35
7.2 Adaptive color spaces 35
7.3 Constant Luminance 35
8 Annex 37
8.1 SGI to Planar 37
8.2 Planar to RGB 37
8.3 Down-sampling functions 37
8.4 PSNR function 38
8.5 Chroma rotation function 39
8.6 Conversion between R'G'B' and Y'CoCg or Y'FbFr color spaces 39
9 References 40

1 Introduction

The current video coding workflow has remained untouched even after the digital era advent, and there are still some parts of it that have their origin in analogue electronics constraints that no longer exist.
This project tries to improve video coding quality by analyzing the current video workflow, identifying troublesome parts, losses causes and finally proposing solutions that improve its overall objective and subjective coding quality.

1.1 The context

VCEG and MPEG are currently working on the next generation of video coding, called High Efficiency Video Coding (HEVC). The HEVC standard will be publicly available in the early of 2013 and may replace the former Advanced Video Coding (H.264/AVC) standard in many applications.
Additionally, the European Broadcasting Union (EBU) recently published a recommendation for the next generation of TV, beyond current HDTV and called Ultra High Definition TV (UHDTV) [1]. UHDTV features enhance those of HDTV with higher screen resolutions (4K, 8K), larger Field of View (FoV), better color rendering (Wide Color Gamut), higher frame rates and bit-depth.
In this context, this report proposes some ways to improve coding quality in both objective and subjective terms. The main action field is picture-by-picture processing of video sequences before encoding them (also known as pre-processing). Knowing the principles of video coding is crucial in order to find which manipulations are more appropriate, and may provide better results.

1.2 Objectives

The aim of this project is to understand the current video workflow to identify were most losses in terms of PSNR are found and try to address them.

The main approaches that will build this project guidelines are:

- Understand the origins of gamma correction and how it should be applied
- Test perceptual color spaces to avoid the use of the gamma correction and to fit human perception of light
- Test reversible color space to avoid conversion losses.

At the end of this document there is an annex with all implemented software to carry out this project.

2 State of the Art

2.1 Video Coding principles

As mentioned, even if video coding is not directly addressed, knowing its principles will allow a better approach in the carried out pre-processing techniques.

Video compression reduces the number of bits required to store or transmit the video signal by exploiting its psycho-visual, spatial, temporal and statistical redundancies. Without compression, HDTV would require around 1 Giga bits per second.

The type of video compression used in this report will be carried out by the new codec HEVC but it is extendable to prior codecs. It is a lossy compression technique, meaning not all information to be encoded will be recovered exactly (the original video signal is distorted) which allows for higher compression rates.

In this section, MPEG coding principles will be addressed, and the specific HEVC features will be discussed in the following one.

One main principle used in MPEG video coding is the fact that consecutive pictures in a video sequence, except in case of scene changes, are usually very similar (temporal redundancy).
Consequently, such hybrid (spatiotemporal) video coding algorithms adopt the following technical characteristics:

- First, pictures are split into blocks on which coding algorithm will be run.
- Some pictures of the sequence are defined as references and they can be used to predict temporally other pictures' blocks (prediction signal).
- The prediction signal is subtracted from the real picture, resulting in a difference signal called "residual", which is quantified and, therefore, there is a loss of information. It is the main source of non-reversibility of the encoding.
- Predictions for a given block can be performed within the same picture (called intra prediction) or from other pictures (called inter prediction).

A video sequence in MPEG is partitioned into groups of successive pictures (GOPs) [2]. Depending on the nature of the block predictions of each picture in the GOP, there are mainly three picture coding types, namely I, P and B. The I-picture is encoded using only spatial information (intra prediction), so once it is reconstructed it may become a key picture for predicting non-intra pictures. A P-picture contains elements predicted from one reference picture stored in the Decoded Picture Buffer (DPB), and the B-Pictures contain elements predicted from up to two reference pictures. B-Pictures may be bi-predicted by building the prediction signal as a combination of two uni-prediction processes from different frames.
Pictures received at the decoder are not necessary chronologically ordered, but eventually reordered after decoding.

The following figure illustrates an example of how pictures may be reconstructed:

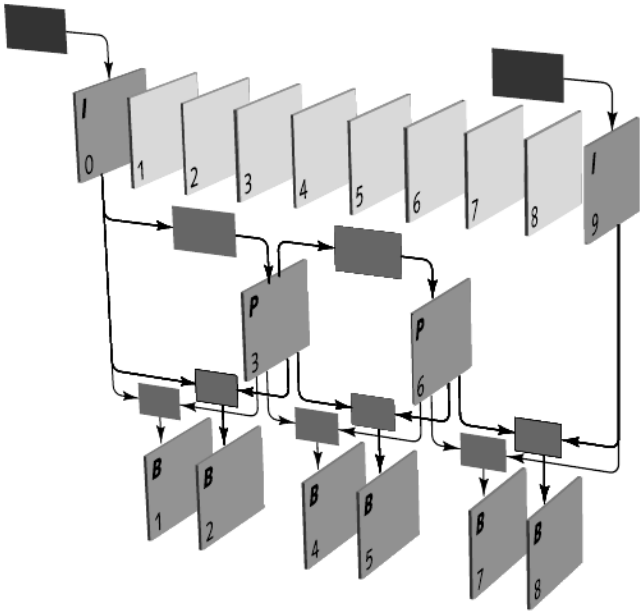

Figure 1 Example of MPEG-encoded picture sequence hierarchy (GOP)

2.1.1 HEVC: High Efficiency Video Coding

High Efficiency Video Coding (HEVC) [3] is a new video compression standard, a successor to H.264/MPEG-4 AVC (Advanced Video Coding), currently under joint development by the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG).

HEVC aims to substantially improve coding efficiency compared to AVC, i.e. reducing bitrate requirements by half with comparable subjective image quality, at the expense of increased computational complexity. Depending on the application requirements, HEVC should be able to trade off computational complexity, compression rate, robustness to errors and processing delay time.

HEVC is targeted at next-generation HDTV displays and content capture systems, which feature progressive scanned frame rates and display resolutions from SQCIF (128x96) up to 1080p (1920x1080) and Ultra High Definition TeleVision (UHDTV) namely 4K (4096x2304) or 8K (7680x4320), as well as improved picture quality in terms of noise level, color gamut and dynamic range.

This paradigm also introduces some new coding tools, as well as it improves other already used tools. Maybe, the most important change affects the picture partitioning. HEVC deals with the terms of macroblock (MB) and block for the Motion Compensation and the transform respectively and introduces three new concepts: Coding Unit (CU), Prediction Unit (PU) and Transform Unit (TU). This structure leads to a more flexible coding to suit the signal characteristics of the picture. Each picture is partitioned into squared regions of variable size

called CUs which replace the MB structure of previous VCEG/MPEG standards. To fix the size of each CU, first of all, a picture is divided into blocks of various pixels sizes, each of which is called Coding Tree Block (CTB), and it ranges from 16x16 to 64x64. For the Main Profile, each CTB can be partitioned recursively into 4 smaller sub-areas (coding block) of a quarter of the original area. Each CU may contain one or several PUs and TUs. Its size is constrained by the Main Profile and user preferences, classically from 8x8 to 64x64 pixels. This partitioning can be done with each sub-area recursively until it reaches a size of 8x8 pixels. An example of quadtree structure is shown in Figure 2 [4]:

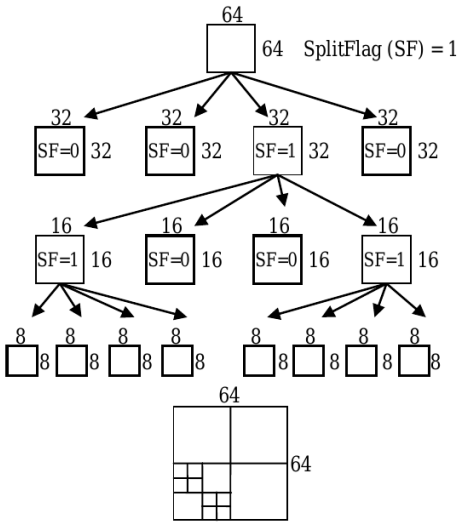

Figure 2 Example of CU splitting with its representation in a quadtree structure.

2.2 The role of gamma correction

Gamma correction concept is one of the most important parts of the whole video workflow. Understanding it is essential in order to properly apply filters and other image processing techniques.

This section introduces its origins and the reasons why encoding pictures with gamma correction provide better results.

2.2.1 Origins: human perception

The use of gamma correction in today's video coding has its origin both in human perception and display technology (CRT). Human eye perception of lightness can be modeled by the Weber-Fechner law, which states that the just-noticeable difference (JND) between two stimuli is proportional to the magnitude of the stimuli. In practical terms, if lightness is uniformly quantized using 8 bits (0 equals black and 255 equals white) differences between low lightness values are perceived as much bigger with regards to the actual lightness level than differences for high lightness levels; which means that if lightness is quantized uniformly, there is a loss of perceivable lightness information near black level, and the perception of saturation near white level (Figure 3). So, in order to adapt light coding to human perception, a quantization other than uniform has to be done. In fact, quantizing light uniformly would require 11 bits or more to

reach the same picture quality when encoding light non-uniformly with 8 bits. Thus, for a given dynamic range, signal-to-noise ratio is optimal for a perceptually uniform quantization of light.

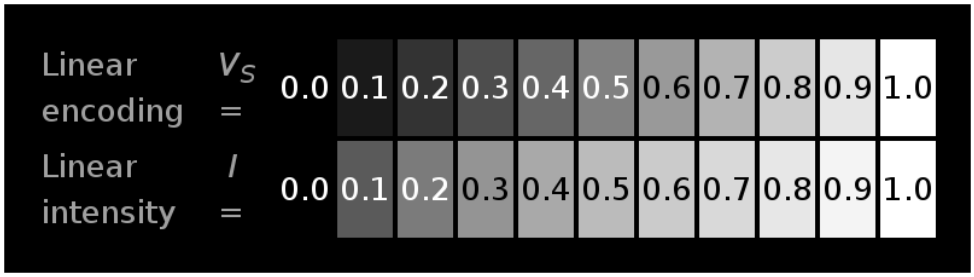

Figure 3: Top: Psycho-visually uniform (but non-linear) light quantization, Bottom: linear light quantization.

The International Commission on Illumination (CIE) adopted a standard function L^* to model human vision behavior of perceived light:

$$L^* = \begin{cases} 903.3 \dfrac{Y}{Y_n} & \dfrac{Y}{Y_n} \leq 0.008856 \\ 116 \left(\dfrac{Y}{Y_n} \right)^{\frac{1}{3}} & 0.008856 < \dfrac{Y}{Y_n} \end{cases} \quad (1)$$

Where Y is the luminance, the luminous intensity per unit projected area [5], computed from linear RGB values and Y_n is the luminance of the white reference.

L^* has a floating range of 0 to 100 and it is defined so that a unit change in L^* equals to the threshold of visibility, which is about 1% difference between two levels (or code words).
With the appropriate light coding function defined, video coding can be carried out without the need of extra bits for achieving good picture quality. Finally, an inversion of the L^* function is required before displaying the picture.

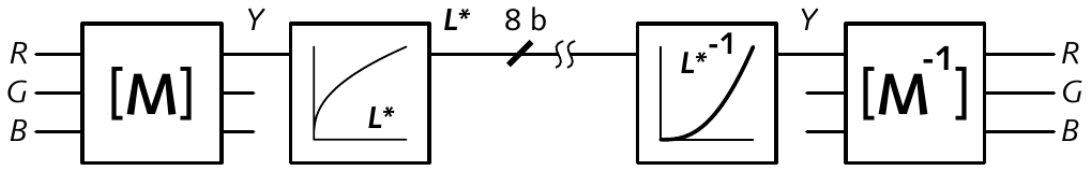

Figure 4 Initial approach for perceptual coding of light.

2.2.2 Engineering trade-off: CRT screens

In the previous subsection, the origins of gamma have been addressed, but the word "gamma" did not appear in the whole explanation.
It turns out that CRT displays have a voltage to light response, also called gamma, that is so close to the inverse of the lightness sensitivity of vision, that an engineering compromise was made: instead of encoding Y using the L^* transfer function, RGB intensities were encoded using the inverse of CRT's function, which allowed CRT displays not to have any extra circuitry.

Figure 5 CRT displays curve.

Figure 6 Adding extra circuitry to displays.

However, the use of this function instead of the "real" one implies a change in the order of operations, which causes a departure from the principle of constant luminance. Therefore, resulting pictures do not look as good as they were expected to, due to eye's nature. For this reason, instead of correcting pictures in the display, which would require extra circuitry, pictures are gamma corrected at the beginning of the chain with a slightly modified inverse CRT curve function that takes into account CRT's and eye's light response as well as the average viewing conditions of the user (dim instead of dark).

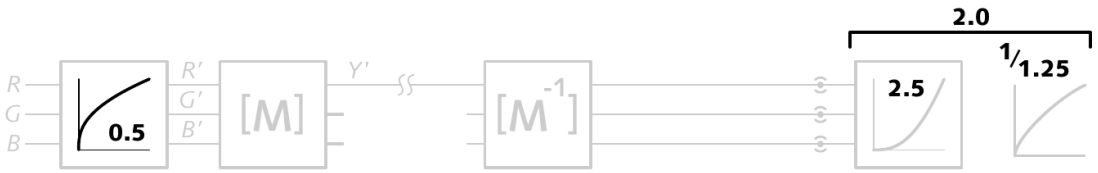

Figure 7 Final solution.

2.3 Color Spaces

A color space is an abstract mathematical model specified by color primaries that is used to describe a representation of a picture, usually as a set of tristimulus values.

To understand why there are many of them, a new concept must be introduced: the gamut. Each color space and device can be characterized by their gamut, the subset of colors which they can represent. Depending on the color space or device used, the gamut will cover more or less portion of the color space the human visual system can perceive.

2.3.1 RGB

The RGB (Red-Green-Blue) model (from Grassmann's law) uses additive color mixing: red, green and blue lights that are added together to reproduce a range of colors. It is one of the most well-known color models, due to its derived color spaces, such as sRGB, which are used in commercial monitors, printing, and digital picture processing.

The sRGB color space uses the ITU-R BT.709 (Rec. 709) primaries [6] definition for colors red, green and blue (the ones used in studio monitors and HDTV), as well as gamma correction definition.

The following color spaces use the sRGB color space as a starting point and try to separate the luminance information from the chrominance information. That is, given a picture, these color spaces separate them into a grey-scale picture (luminance) and another picture that contains all color information (chrominance) but no luminance, so that the superposition of these two pictures results in the original picture. The reason for this separation is human perception, as human eye has significantly less ability to sense details in color information than in lightness. Therefore, provided lightness detail is maintained, color detail can be reduced by sub-sampling without large perceived (subjective) difference for most contents and, consequently, reducing the whole picture data size.

2.3.2 Coding Color Spaces

The RGB family of color spaces is not suitable for video coding as the three components are not well de-correlated. They are only used for applications requiring very high quality (production, FX…) where having no losses through color space conversion is more important than reducing the video file size dramatically.
In video coding, the initial data compression rate is achieved by exploiting the nature of human perception. All target color spaces used for video coding have one thing in common: they separate the light information from the color, which allows downsampling color information by a factor of up to two vertically and horizontally without causing significant visual damage to the picture perception.

Coding color spaces work in the perceptual domain. However, luminance and chrominance concepts refer to linear magnitudes, computed from linear RGB. In video coding, luma and chroma are used to refer to light and color information in the perceptual domain, therefore they are computed from gamma corrected R'G'B' instead of linear RGB. This concept is known as non-constant luminance (NCL) that we will detail later. The following schemes describe this difference, where the starting color space is linear RGB:

Figure 8 Luminance and Chrominance.

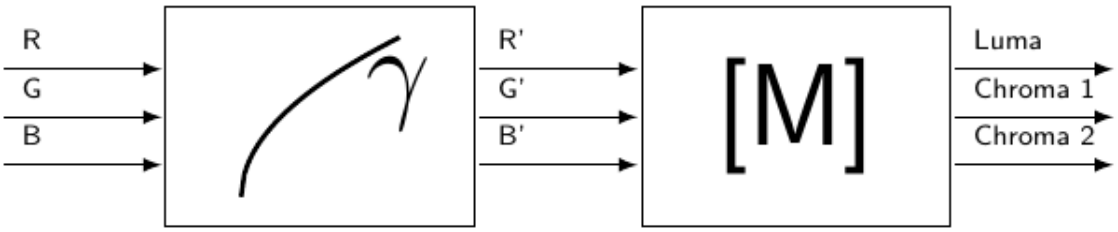

Figure 9 Luma and Chroma.

Color spaces used to represent data to be coded in this project include Y'CbCr, Y'CoCg, Y'FbFr and Lab, which are described in detail in the following sections. These color spaces have been referred to as coding color spaces, due to the fact that pictures are coded using these color spaces

2.4 Chroma downsampling

Before describing chroma downsampling, the technical difference between chrominance and chroma must be addressed. Chrominance is the color information of a linear RGB picture and can be obtained by subtracting the luminance of a picture from the RGB linear picture. The chroma concept is related to video coding and stands for the difference between the R'G'B' picture and the luma, computed from R'G'B' values.
The chroma sampling [2] format follows a specific notation, usually represented by three numbers separated by colons. The first digit indicates the luma horizontal sampling reference, the second digit specifies the horizontal sub-sampling of the chroma channels for even rows and the third digit has the same definition as the second one but applied to odd rows. The last two digits define then the chroma vertical sampling format.

The next figures illustrate how this notation is translated graphically, where circles refer to picture samples or pixels, black circles are actual chroma samples and outlined circles are blanks [7]:

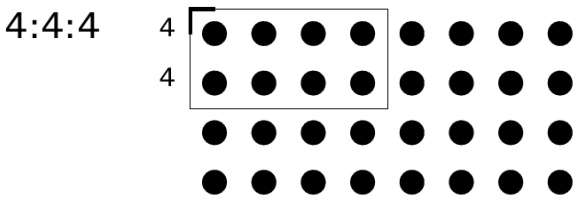

Figure 10 No chroma subsampling

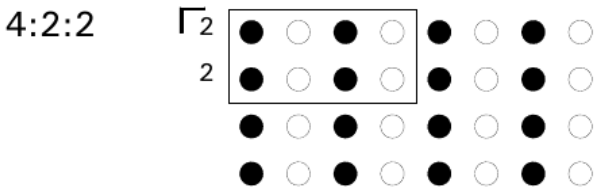

Figure 11 Horizontal chroma subsampling

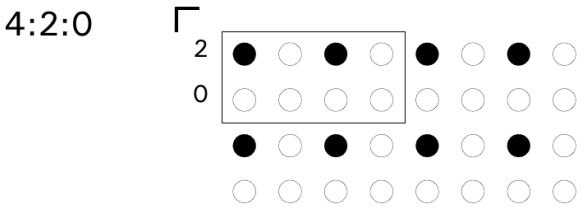

Figure 12 Horizontal and vertical chroma subsampling

These subsampling operations may be carried out by a separable half-phased Lanczos3 filter, first applied horizontally and then vertically to both chroma channels, that is explained in section 3.1.2.

3 Digital video workflow

Figure below shows the current video workflow, used in this project:

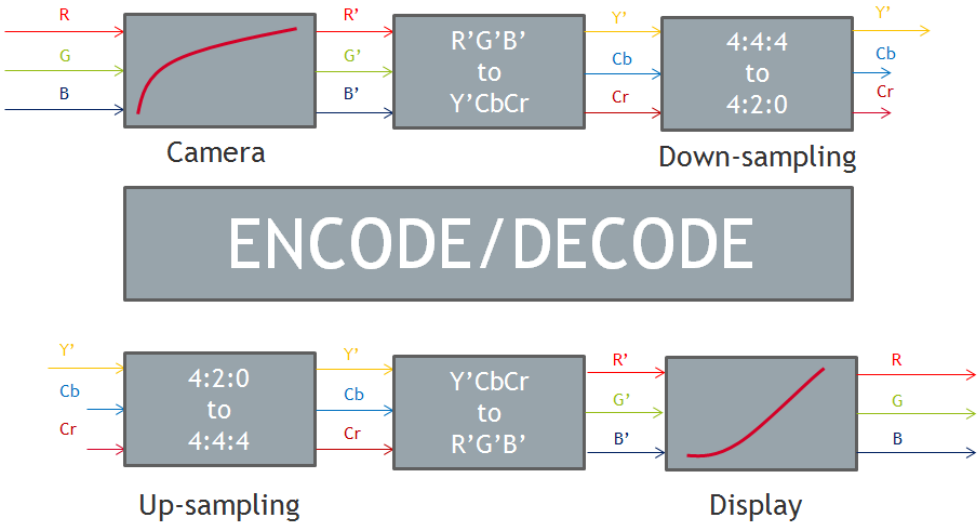

Figure 13 Current video workflow

Source images used are placed after the camera in the diagram, therefore the camera step (and it response curve) cannot be modified. However, all other steps before video encoding and decoding can be modified according to our needs, as long as the output format is suitable for the display, which is non modifiable either.

To better understand the conversion chain workflow, it is crucial to know the intrinsic properties of the source data we used:

- Format: SGI (R'G'B')
- Bit-depth: 16 bits
- Width in luma samples: 3840 pixels
- Height in luma samples: 2160 pixels

Actually, this picture format is used as we know each step of the process of how these pictures were built. These pictures are going to be pre-processed, then encoded, decoded, post-processed prior to being displayed on HDTV display (1920 x 1080) with a bit-depth of 8 and 10 bits.

3.1 Pre-processing the sequence

The figure below summarizes all steps carried out to pre-process the set of pictures that are going to be encoded with HEVC.

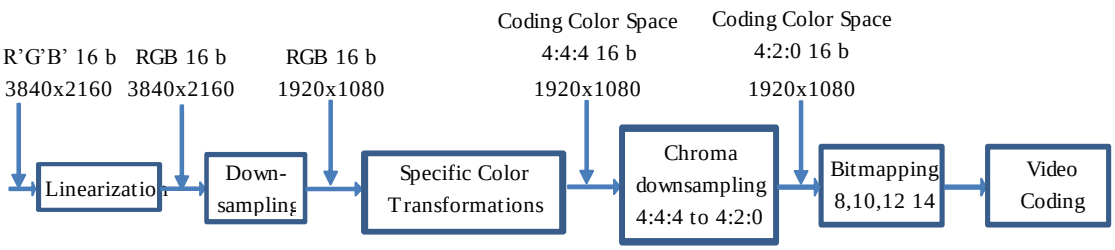

Figure 14: Pre-processing chain

Each block of the chain is explained in detail in the upcoming subsections.

3.1.1 Linearization: Gamma Correction

In video, computer graphics, and picture processing, the gamma symbol γ represents a numerical parameter that describes the non-linear relationship between pixel value and luminance. [8]

According to ITU-R BT.709 [6], the gamma function is applied for each sub-pixel C of each color plane in the sRGB color space [9] as follows:

$$C_{R'G'B'} = \begin{cases} 12.92 C_{RGB} & C_{RGB} \leq 0.031308 \\ (1 + 0.055) C_{RGB}^{1/2.4} - 0.055 & C_{RGB} > 0.031308 \end{cases} \quad (2)$$

$$C_{RGB} = \begin{cases} \dfrac{C_{R'G'B'}}{12.92} & C_{R'G'B'} \leq 0.04045 \\ \left(\dfrac{C_{R'G'B'} + 0.055}{1 + 0.055} \right)^{2.4} & C_{R'G'B'} > 0.04045 \end{cases} \quad (3)$$

Although it is normally said that sRGB has a gamma of 2.2, the above function shows an exponent of 2.4 to balance the linear part at the beginning, which is done to avoid noise issues in very dark colors due to electronic limitations.

Despite being apparently quite simple, gamma correction is, conceptually, one of the most difficult parts to understand and to work with in image processing and video coding. This is due to its non-linear nature.
Gamma correction is needed to balance the response curve of the screen displaying the pictures and the human eye response curve, so that the colors and luminance correspond as much as possible to what we see in reality (although it depends also on display rendering fidelity).

Most common picture processing operations, such as filtering and down-sampling, are intended to operate in a linear way and do not perform well with the non-linearity introduced by the gamma correction. In order to avoid well-known problems when down-sampling non-linearized pictures, such as dithering and color information loss [10], gamma correction may be preferably applied to linearize pixel values before performing any linear-light operation, in this case, applying a linear filter to downsample the picture.

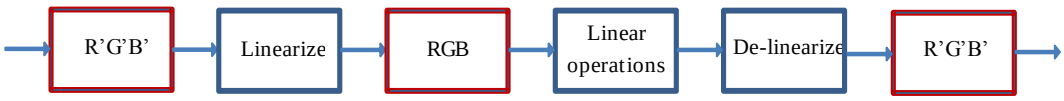

Figure 15: Correct order to operate with non-linear pictures.

The block listed as "Linear operations" may contain filtering operations, such as resampling filters, in which operations are carried out assuming linear input data.

Having linearized data, the following step is to down-sample 3840x2160 sized images to 1920x1080 in order to fit the HDTV standard.

3.1.2 Down-sampling

Once values are linearized, the filtering operations may be performed. The method chosen to down-sample the sequence is a Lanczos filter due to its quality in terms of higher sharpness and fewer artifacts when compared to other re-sampling filters such as bi-cubic. [11]

In particular, the chosen Lanczos filter is a three-lobed half-phased filter, called Lanczos3:

$$L(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x} \frac{\sin(\pi x/3)}{\pi x/3} & -3 < x < 3 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

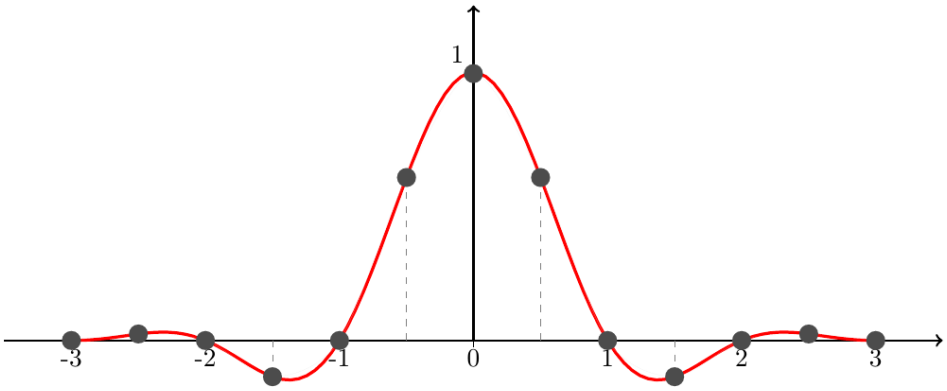

Figure 16 Zero-phased Lanczos3

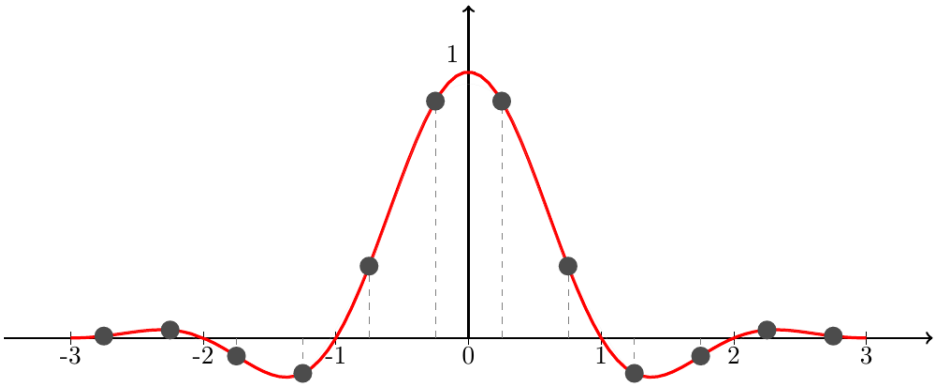

Figure 17 Half-phased Lanczos3

"Half-phased" is a subcase of linear phased filters meaning that the peak of the function is between two samples, which implies that the resulting sample after filtering will be located between those two original samples in case the down-sampling factor is two. The opposite would be a zero-phased filter, where the peak position would coincide with an input sample. If a generated sample were located with the same phase as an original sample, the Lanczos filter would not take into account all involved samples and the output sample would not be weighted fairly from the neighborhood, resulting in potential visual artifacts. These visual artifacts

include offsets between light information and color information when subsampling color channels with zero-phased filters.

Pictures are down-sampled in two steps, first horizontally and then vertically, as the filters used are separable, meaning they act in one dimension only, leaving the other one untouched. This approach is computationally more efficient, since there are fewer operations to carry out than with non-separable filter implemented as a matrix. Another aspect to take into account is the filter length. In this case, the filter has 12 taps, so when generating samples near the borders, the six pixels near the borders are padded (mirrored) to apply the filter properly. Next scheme summarizes the steps followed to down-sample pictures:

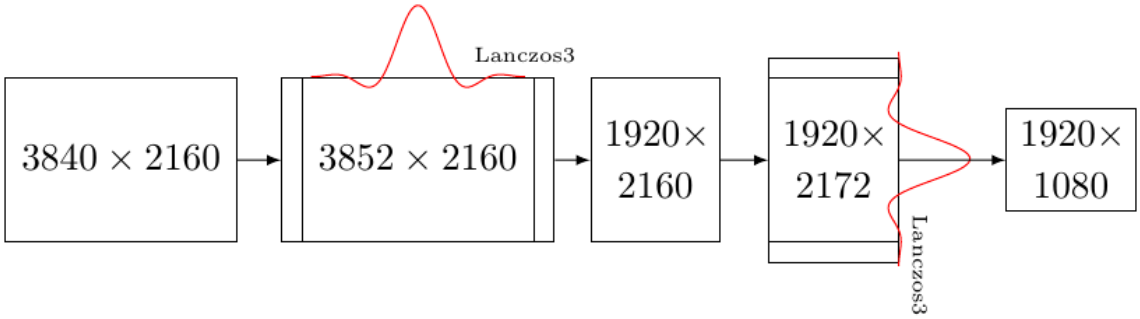

Figure 18 Applying the Lanczos3 filter in two dimensions on selected content.

Visually, the filter generates the second row of following figure using the first one as a starting point:

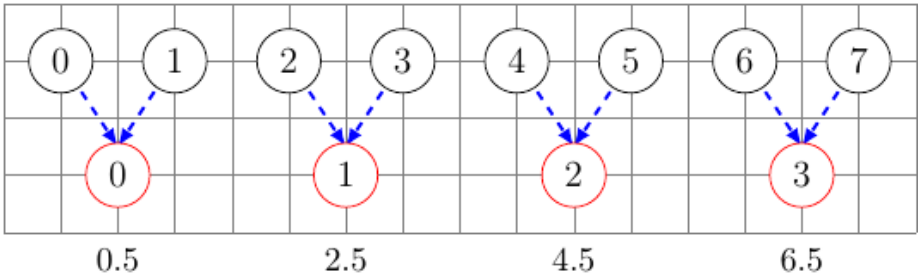

Figure 19 Lanczos generated samples with relative positions.

Numbers below samples indicate the relative position (in samples) of output samples with regards to first input sample.
After having adjusted pictures to their final size, color conversion operations can be carried out. The down-sampling has been placed before all conversion operations to avoid unnecessary calculations.

3.1.3 Specific color transformations

The source sequence has been linearized and down-sampled. The next step consists in representing the sequence in an appropriate color space for video coding.
As mentioned before, all color space conversion carried out follow the same objective: de-correlate lightness information from color information. To the extent of this report, the following color spaces will be tested: Y'CbCr, Lab, Y'CoCg and Y'FbFr. They will be detailed thoroughly in the next sections.

Moreover, all coding color spaces work on the perceptual domain, therefore, apart from Lab, which is detailed in section 3.1.3.2, all other color spaces need gamma correction. That means the starting color space is R'G'B'.

3.1.3.1 Y'CbCr

This color space is used in most MPEG coding standards variants. It is the equivalent of the Y'PbPr color space, used in analogue television, but for digital television. The main reasons for its use are historical: this color space transform was easily performed with analogue circuitry and had good perceptual properties, allowing the use of luma, the Y' component, as the monochrome signal in black and white TV (used for backward compatibility with analogue color television).
The Y' component is the luma, computed with a weighted sum of gamma-corrected RGB components, which is very similar to the L^* transfer function. Cb and Cr components are the difference between gamma-corrected red or blue and the luma.
In a more intuitive way, Y' can be seen as a grey scale picture, where no information of color is present; and the Cb and Cr components represent what needs to be added to the grey scale picture to get the colorful picture.
Its conversion equations from R'G'B' ranging from 0 to 255 (8 bits conversion) are:

$$\begin{aligned} Y' &= 16 + \left\lfloor \frac{65.738 \cdot R'}{256} + \frac{129.057 \cdot G'}{256} + \frac{25.064 \cdot B'}{256} \right\rfloor \\ C_B &= 128 + \left\lfloor \frac{-37.945 \cdot R'}{256} - \frac{74.494 \cdot G'}{256} + \frac{112.439 \cdot B'}{256} \right\rfloor \\ C_R &= 128 + \left\lfloor \frac{112.439 \cdot R'}{256} - \frac{94.154 \cdot G'}{256} - \frac{18.285 \cdot B'}{256} \right\rfloor \end{aligned} \quad (5)$$

Integer equations for integer implementation purposes are also described in ITU standards (Rec. 1361 for HDTV).
There is no need of a Cg component (the difference between the green and the luma) because of the luma equation establishing a relationship between the three RGB components.

3.1.3.2 Lab

L*a*b*, also called CIE Lab or simply Lab, is a perceptual color space in the sense that it defines all perceivable colors and, therefore, its representable colors include and exceed those of RGB.
Lab goes one step further with regards to other color spaces by making colorimetric distances between color values correspond to the actual perceived color differences or, in other words, the difference observed between two colors is reflected as a distance in their gamut position. Moreover, the Lab color space is device independent, meaning that color definitions do not

depend on the device they have been created from or displayed on; in practical terms, it means that there is no need of gamma correction when operating with device independent color spaces. The figure below shows the required steps to convert from R'G'B' to Lab:

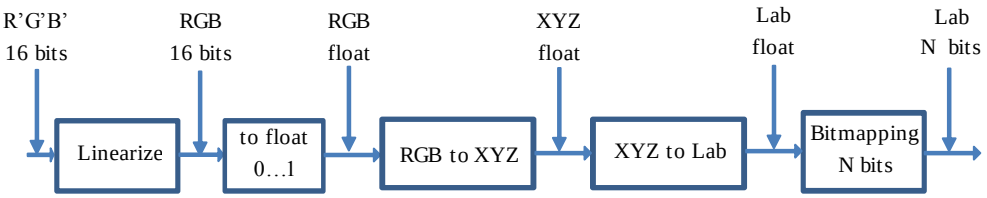

Figure 20: Conversion from R'G'B' to Lab.

In order to obtain the Lab color space, the RGB color space must be made device independent by converting it to the XYZ color space, also known as CIE 1931. Afterwards, conversion to Lab can be carried out.

The XYZ color space is based on the description of color as a luminance component Y and two additional components X and Z, whose weighting curves have been standardized by the CIE based on statistics from experiments involving human observers. XYZ tristimulus values can describe any color and are proportional to physical energy, but their spectral composition corresponds to the color matching characteristics of human vision [12].

Next equation shows the conversion from RGB linear in floating point between 0 and 1 to XYZ:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.95905 \end{bmatrix} \begin{bmatrix} R_{linear} \\ B_{linear} \\ G_{linear} \end{bmatrix} \tag{6}$$

Once RGB are made device independent by converting them to XYZ, the following equations are applied to reach the Lab color space.

$$\begin{aligned} L^* &= 116\, f(Y/Y_n) - 16 \\ a^* &= 500\, \left[f(X/X_n) - f(Y/Y_n) \right] \\ b^* &= 200\, \left[f(Y/Y_n) - f(Z/Z_n) \right] \end{aligned} \tag{7}$$

where

$$f(t) = \begin{cases} t^{1/3} & t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{otherwise} \end{cases}$$

X_n, Y_n and Z_n are the values for the normalized white point.

Despite the fact that these conversion equations do not work with integers and they are not fully reversible, Lab has been designed to approximate human vision, which may lead to better subjective results.
In Lab, the *L* channel, sometimes referred as *L**, specifies the Luminance of the picture; the *a* and *b* channels specify the hues from green to magenta and from blue to yellow, respectively. *L* ranges from 0 to 100 in floating point and *a* and *b* range from -128 to 127 in floating point, too. This is due to the fact that Lab's gamut is much larger than RGB's, so for precision's sake, dynamic range must be increased.

3.1.3.3 Y'CoCg

This color space was conceived to achieve even a better de-correlation than the one provided by Y'CbCr and to keep very high quality in top-notch applications due to its reversibility [13]. Coefficients of this color transform are calculated by simplifying the resulting Karhunen-Loève transform (KLT) of a wide range of sequences. Y'CoCg stands for luma, orange offset and green offset and it has a variant called Y'CoCg-R [14], which is completely reversible, hence the R, and which has a very low computational cost, yielding a very fast conversion from and to R'G'B'. As a result, this variant has been the one chosen in this project to encode the sequences, due to its processor-friendly properties: only 4 additions and two shifts per pixel are required, as shown in the schema below, where '>>' is the arithmetic right shift operator, which represents a division by two:

$$\begin{aligned} Co &= R - B \\ t &= B + (Co >> 1) \\ Cg &= G - t \\ Y &= t + (Cg >> 1) \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} t &= Y - (Cg >> 1) \\ G &= Cg + t \\ B &= t - (Co >> 1) \\ R &= B + Co \end{aligned} \tag{8}$$

It can be spotted from the equations above that chroma channels may have negative values. Therefore, an extra bit is added to both of them to increase the dynamic of the positive range enough. In order to preserve coherence in bit depth amongst luma and chroma channels, an extra bit is added to luma.

3.1.3.4 Y'FbFr

Y'FbFr [15] is another color space built upon on the same ideas as Y'CoCg: making use of the KLT to design an integer reversible color transform with a high coding gain and low bit expansion.
Not only does it share its design ideas with Y'CoCg, but also the way it has been implemented through a lifting scheme [13] in which each pixel is computable with five additions and three bit shifts.

$$\begin{aligned} Fr &= R - B \\ t &= (R + B) >> 1 \\ Fb &= G - t \\ Y &= t + 3Fb >> 3 \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} t &= Y - 3Fb >> 3 \\ G &= t + Fb \\ B &= t - Fr >> 1 \\ R &= t + (Fr >> 1) \end{aligned} \tag{9}$$

Similarly to the previous case, an extra bit needs to be added to chroma channels to avoid negative values, and to the luma channel to preserve bit depth coherence.

3.1.4 Chroma down-sampling

Chroma down-sampling is performed with the same half-phased Lanczos3 filter described in section 3.1.2, but instead of applying the filter to all picture channels, it is only applied to chroma channels, first horizontally and then vertically. The principles of chroma down-sampling are explained in section 2.4.

3.2 Video coding and decoding

The original sequences, which were 3840x2160 R'G'B' 16 bits, have been down-sampled and converted to a coding color space in format 4:2:0 10 bits, suitable for video coding. The version of the coder used in this project is HEVC test model HM 6.0, High-Efficiency profile. A 10 bits bit-depth is chosen to improve coding precision. However, Y'CoCg and Y'FbFr are encoded with a bit-depth of 12 for the reasons mentioned in the above section. The used encoder is able to work with a bit-depth of up to 14 bits.

3.3 Post-processing the sequence

After the coded pictures are decoded with the HEVC decoder, the inverse process needs to be done to reverse all changes and obtain displayable pictures, that is, DPX [16] RGB pictures with a bit-depth of 8 or 10 bits, since these are the ones supported by the display. The 8 bits bit-depth is chosen since it is the standard for current video coding of broadcast and mainstream applications, however, it has a low dynamic range. The 10 bits bit-depth is chosen to overcome the limitations due to a low bit-depth and achieve higher precision in calculations done by the encoder.

Except for the Y'CoCg and Y'FbFr, whose transformation depends on the bit-depth, the first processing step applied to the received stream is to shift all channels up to 16 bits in order to carry out all operations with the maximum possible precision.

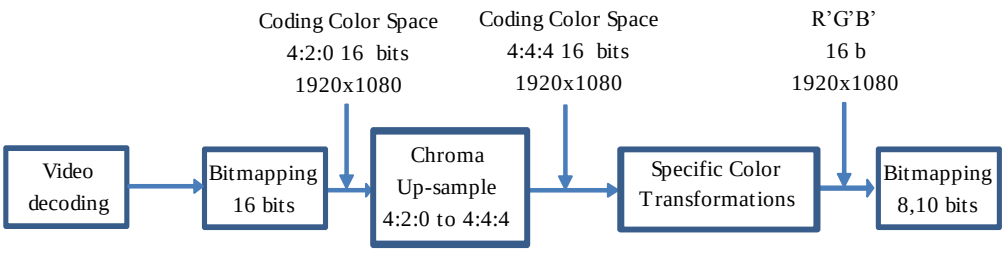

Figure 21 Post-processing chain

3.3.1 Chroma up-sampling

For chroma up-sampling, an interpolation filter based on cubic interpolating splines is used [17]. In the same way as Lanczos filter, up-sampling is done in two steps: vertical and horizontal. However, this up-sampling filter implementation is more complex than the down-sampling Lanczos half phased filter described before, as it changes depending on the position of the picture: its definition changes on the borders, near the borders, for even pixels and for odd pixels.

Although this up-sampling method has different filters for both steps, being the vertical filtering half-phased and the horizontal filtering zero-phased, objective and subjective tests show that using a half-phased filter in both cases provides better results: PSNR using half-phased up-sampling filters are about 1 dB higher and color presents a phase offset when using the non-half-phased filter.

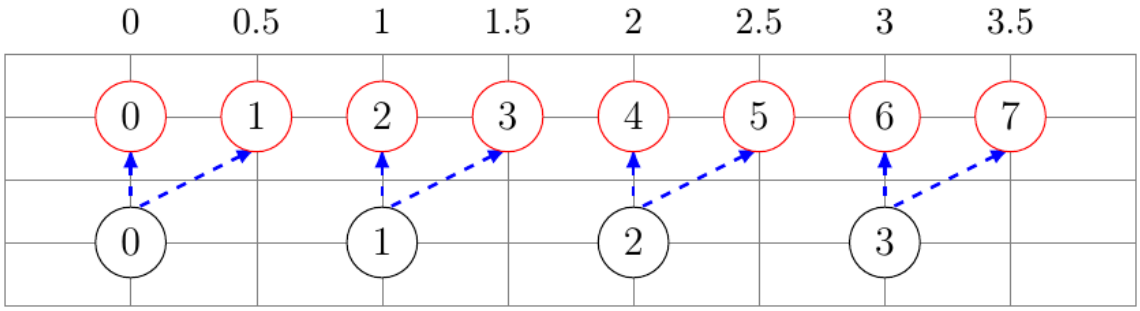

Figure 22 Using a zero phased up-sampling filter causes en error in phase

The reason is that this filter operates with data already down-sampled by a half-phased filter. An approach to use a zero-phased up-sampling filter is to modify the Lanczos filter in order to be zero-phased. However, as explained in section 3.1.4, this method causes visual artifacts.

3.3.2 Specific Color Transforms

This block is in charge to convert sequences from coding color spaces to displayable R'G'B'. It does the inverse operations carried out when pre-processing the sequence.

4 Adapting pictures representation to color space

Coding quality varies largely depending on the processed content. This is due to the nature of the predictions that are carried out inside the coding process. This fact was taken into account in the design of Y'CoCg and Y'FbFr, which were designed by encoding a representative set of video sequences and tuning filters to maintain a threshold between reversibility and results. However, if sequences under test differ significantly from the design sequences, coding quality with those color spaces might decrease with regards to Y'CbCr.

In this section, a new adaptive color transform is being addressed. Instead of designing a color space for all kind of sequences, an adaptive color space per pictures sequence (content adapted) will be used. The way of adapting to the content of the pictures is intuitive, as the figure below shows:

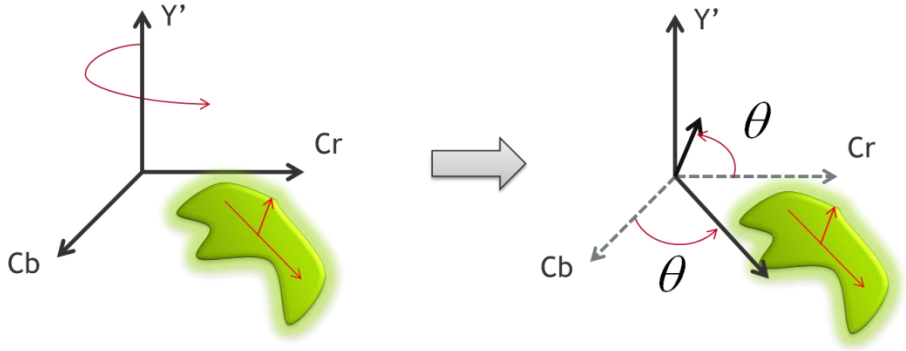

Figure 23 Rotation graphical representation

The goal of this transformation is to reduce the inter-component correlation by rotating the chroma axes around the luma axis. Therefore, in this color adaptation, no luma transformations are carried out, only chroma is adapted by representing its histogram on the plane formed by the chroma axes and rotating them to adapt to the color space.

The first chosen naïve method to find the rotation angle has been the Principal Component Analysis (PCA), a technique of multivariate analysis, introduced by Pearson in 1901 [18].

The main idea of PCA is to reduce the dimensionality of a data set consisting of large number of interrelated variables. The approach followed in this report is slightly different, since the goal is to represent the dataset formed by all chroma values of a picture with new adapted axes. Therefore, no need of dimension reduction is needed, since the starting space has the same dimensions as the output space.

The idea of designing a color transform using the Principal Component Analysis (PCA) in the chroma channels implies adapting the transform to content, which is close to what the optimal Karhunen-Loève Transform (KLT) does. The way KLT is employed with other state-of-the-art transforms is that it uses the RGB color space as the starting space and tries to find the three orthogonal axes that de-correlate most the picture data, whereas the approach in this project

starts from the perceptual color space Y'CbCr and applies the PCA only to the chroma axes, Cb and Cr.

The only extra information needed for a decoder is the rotation angle (or its inverse) of the new chroma axes with respect to the original ones in order to display properly decoded pictures. This angle is computed with the principal component of the dataset formed by both chroma channels with the standard procedure:

1. Calculate the correlation matrix of the zero-mean input data: the chroma channels (10)
2. Find its Eigen vectors and Eigen values (11)
3. Project the data set to the space defined by the Eigen vectors (12)

$$\mathbf{R} = \begin{bmatrix} Cb - mb & Cr - mr \end{bmatrix} \cdot \begin{bmatrix} Cb - mb \\ Cr - mr \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} = \begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix} = \mathrm{eig}(\mathbf{R}) \tag{11}$$

$$\begin{bmatrix} Cb^* & Cr^* \end{bmatrix} = \begin{bmatrix} Cb - mb & Cr - mr \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix}^+ + \begin{bmatrix} mb & mr \end{bmatrix} \tag{12}$$

This transformation can also be seen as a rotation since the normalized Eigen vectors of a 2x2 correlation matrix can always be expressed as:

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{13}$$

This is because the PCA forces the first Eigen vector to match the direction of the dataset maximum variance and places the second one orthogonally to the first one. So, with this transformation orthogonality between color space axes is maintained.

In order not to lose information with the rotation transform, original 8 bit sequences have been shifted to 10 bits per channel and then have been rotated, so that the actual content has a bit-depth of 8 but pictures are stored in a 10 bits bit-depth format. This approach allows measuring the improvements made by the PCA without the risk of losing precision due to rotation operations. Therefore, PSNR's between those sequences are calculated using 10 bits, even if for the original sequences the two least significant bits (LSB) are always set to zero. Although it may seem it decreases the PSNR values, since there are two artificial bits, the effect is balanced by the increase of the dynamic range.

5 Results

Results in the sections of this chapter have been obtained with up to four sequences of 300 pictures sized 1920x1080 and encoded in the 4:2:0 chroma format with 10 bits (12 bits for Y'CoCg and Y'FbFr). The version of the encoder tested is HEVC test model HM 6.0.

Figure below shows the experiments workflow and the points where the PSNR has been computed:

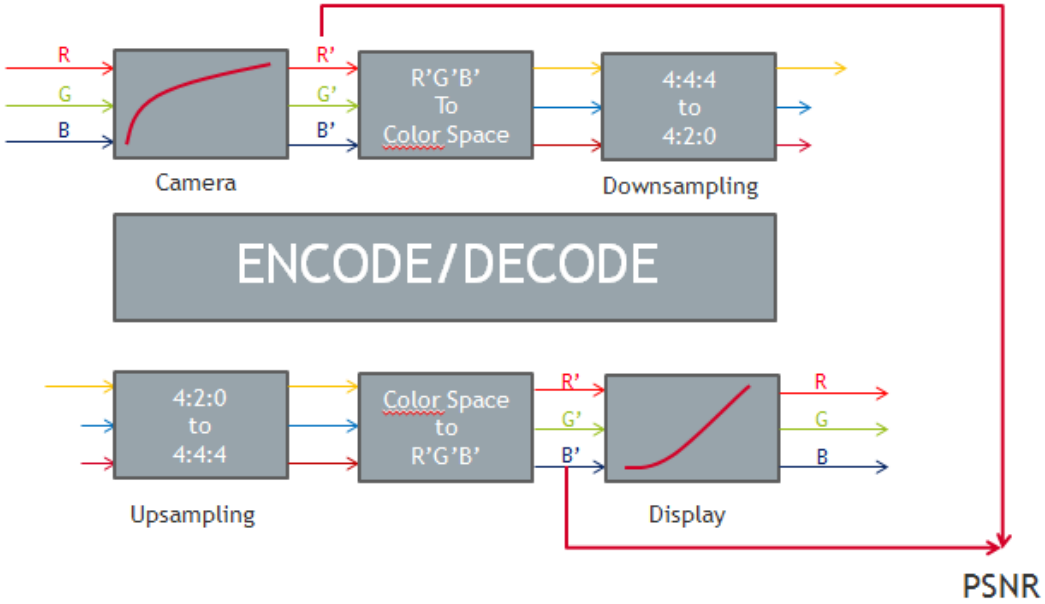

Figure 24 PSNR computation points.

5.1 Assessing the chain without video coding

In order to understand the contribution of each workflow module, the actual conversion chain is tested without the coder module in order to identify losses due to color space conversions and bit mapping. Tests have been carried out by generating a 14 bit-depth sequence and making it go through the whole chain for different color spaces and chroma sampling formats without performing any video compression/decompression. Hence, PSNR results between the source and the decoded sequences are expected to be high (ideally infinite if the whole process is reversible). The PSNR has been calculated between the source pictures, which are in 16 bits (remapped to 14 bits with two right shifts and an offset), and the decoded pictures, which are also in 14 bits.

Figure 25 Original Crowd Run and Ducks Takeoff sequences

Coding Color Space	PSNR Red (dB)	PSNR Green (dB)	PSNR Blue (dB)	Average PSNR (dB)
Lab 4:4:4	78.46	86.78	84.34	83.19
Y'CbCr 4:4:4	85.62	80.74	84.94	83.77
Y'CoCg 4:4:4	99.99	99.99	99.99	99.99
Lab 4:2:0	35.67	44.42	32.13	37.41
Y'CbCr 4:2:0	35.28	46.37	32.47	38.04
Y'CoCg 4:2:0	47.19	53.79	45.91	48.96

Table 1 PSNR in RGB 14 bits without encoding for "Crowd Run" to test the chain.

In case of Y'CoCg, since it is reversible, the losses are caused by sub-sampling and up-sampling operations only. The software used to calculate the PSNR displays 99.99 instead of infinity when the mean square error is 0 to avoid a division by 0, as it is in the denominator of the PSNR formula.

From Table 1 it can be seen that the conversions are almost lossless when there is no sub-sampling involved (4:4:4), given the very high PSNR values.

Such experiments allow setting a reference point, since once sequences are encoded and residues are quantified, losses due to quantization may appear, consequently, PSNRs may be lower.

An important parameter to take into account is the quantization parameter (QP): an index representing the quantization strength of residues. The lower the QP, the finer quantization (quality improvement, distortion decrease). The quantization step is divided by two for every QP increased by six (logarithmic-like behaviour), For H.264/AVC and HEVC, QP is in the range of 0 to 51 for 8-bit content.

5.2 Lab with video coding

For the most perceptual color space, Lab, both objective and subjective tests are performed because it is very hard to find objective metrics that translate human perception of quality that depends on many factors like culture, environment, experience…
Objective results (PSNR) have been slightly better in almost every color channel for the following sequences, contrary to the results without video coding, as the table below shows:

Sequence	Red PSNR (dB)	Green PSNR (dB)	Blue PSNR (dB)
Crowd Run Y'CbCr	31.85	36.00	29.73
Crowd Run Lab	32.21	36.50	30.07
Ducks Take Off Y'CbCr	29.34	33.16	26.43
Ducks Take Off Lab	29.13	34.15	27.36

Table 2 Objective comparison between Y'CbCr and Lab for the Crowd Run and Ducks Takeoff sequences

However, looking at the bitrate distortion curves, the Lab values are obtained with higher bitrates, which means that to reach the same quality as Y'CbCr, Lab needs a higher bitrate.

The following figure shows the bitrate distortion curves for the "Crowd Run" sequence:

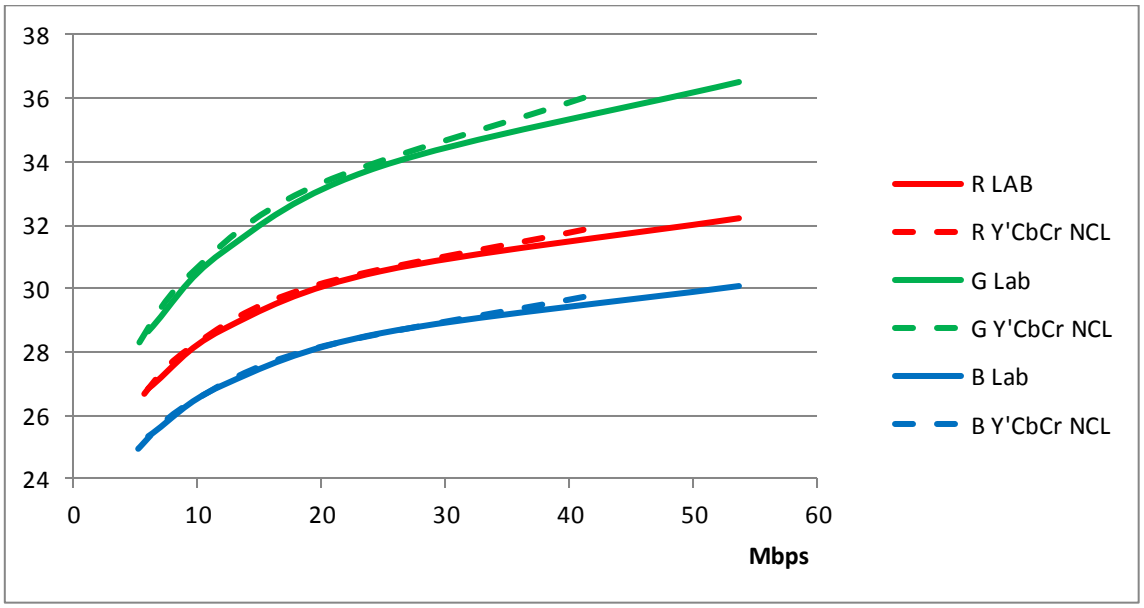

Figure 26 Bitrate distortion curves in dB for "Crowd Run"

It can be seen that, in terms of PSNR, Y'CbCr achieves the same quality as Lab with a lower bit rate, but from Table 1, Lab values of PSNR for a given QP are slightly higher.

The next figure displays the bitrate distortion curves for the "Ducks Takeoff" sequence. There are some remarkable points: green channel behaves slightly better when coding in Lab at higher bitrates. Red channel from Lab is notably below red channel from Y'CbCr, but this sequence has nearly no red in it. Moreover, there is a substantial improvement in the blue channel, the dominant color in this sequence; however, the improvement is only visually perceivable when showing the sequences in a 10 bits display.

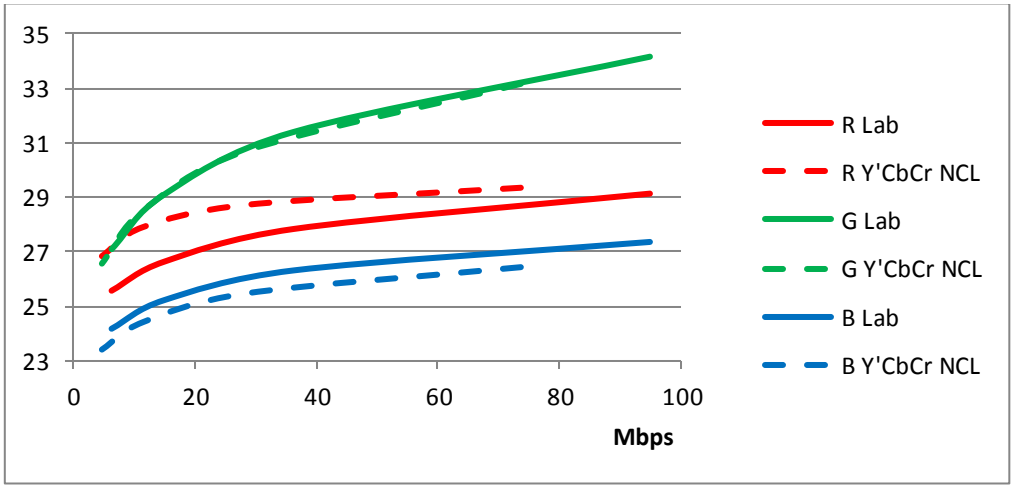

Figure 27 Bitrate distortion curves in dB for "Ducks Takeoff"

Due to non decisive results in objective tests, subjective tests were carried out on a 10 bits bit-depth display, and there were almost non-perceivable differences between resulting sequences at low QPs (22, 27). However, at higher QPs, Lab data representation seems to be slightly better than Y'CbCr one in cluttered zones, filled up with details. So even if RGB PSNRs in Lab is about 1 dB lower than Y'CbCr's for the same bitrate. The fact that Lab was designed to fit human perception is observable when sequences are encoded with relatively high QPs (low bitrates):

Figure 28 Left: Lab QP 37. Right: Y'CbCr QP 37

5.3 Y'CoCg and Y'FbFr

Due to the principle behind Y'CoCg and Y'FbFr and the similar results they have provided, their results have been merged into one section.

As explained in sections 3.1.3.3 and 3.1.3.4, these color spaces require one more bit per channel. In order to make the comparison fair, instead of converting from R'G'B' 16 bits to Y'CoCg or Y'FbFr, as it was done in Y'CbCr, and then shift the resulting sequences to 10 bits, the original R'G'B' content has been shifted to 10 bits. This allows having the same format, R'G'B' 10 bits, after decoding the sequence and converting back to R'G'B' from Y'CbCr, Y'CoCg and Y'FbFr.

Another important point to take into account in this comparison is the location of PSNR calculation. In [14] and [15] PSNR has been calculated before and after the video coding. However, comparing these results with the ones obtained from Y'CbCr is not fair, since source sequences in different color space cannot be compared as encoded content is also different. For this reason, comparisons between different color spaces are all performed in the R'G'B' color space.

Figure 30 and Figure 30 show the bitrate distortion curves for Y'CbCr and Y'CoCg:

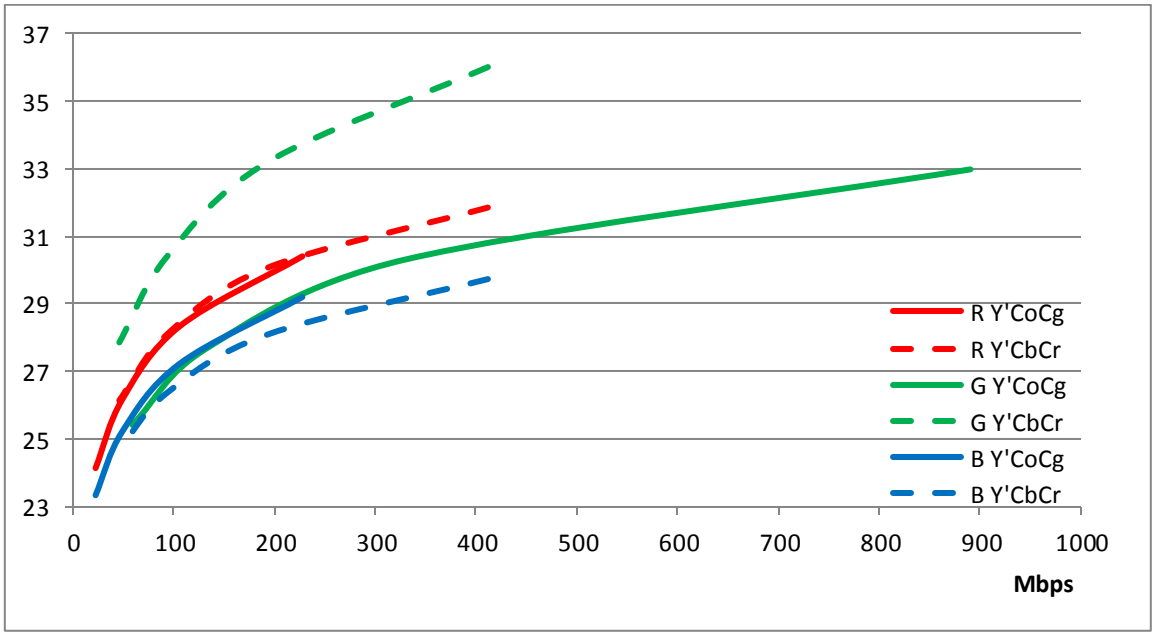

Figure 29 Bitrate distortion curves in dB for "Crowd Run".

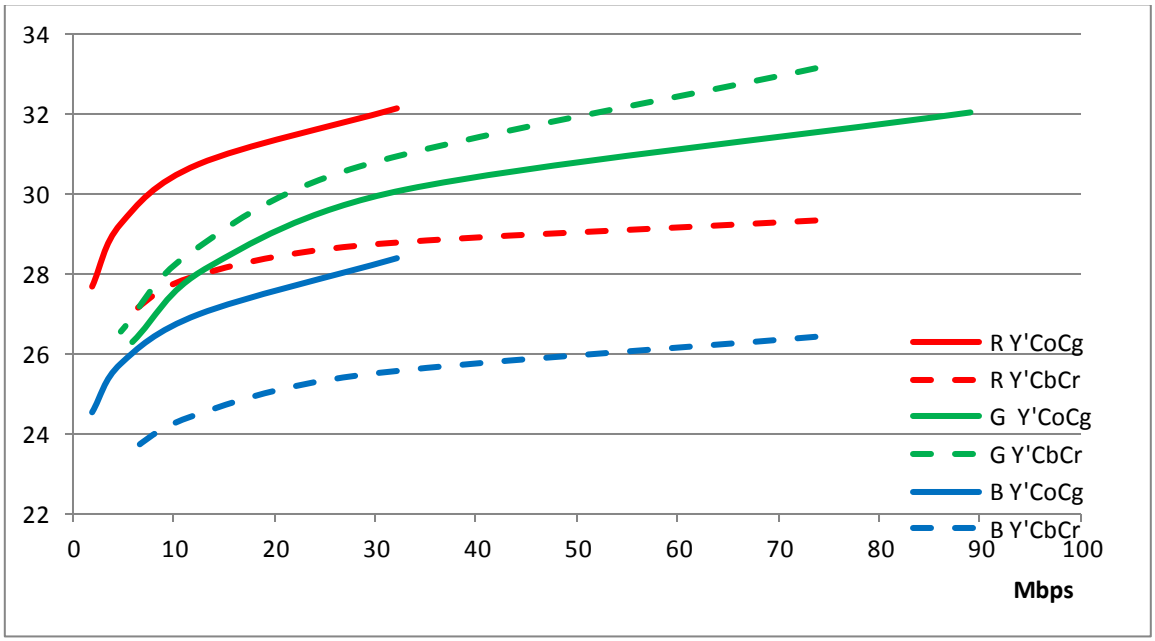

Figure 30 Bitrate distortion in dB curves for "Ducks Takeoff".

Figure 32 and Figure 32 show the bitrate distortion curves for Y'CbCr and Y'FbFr:

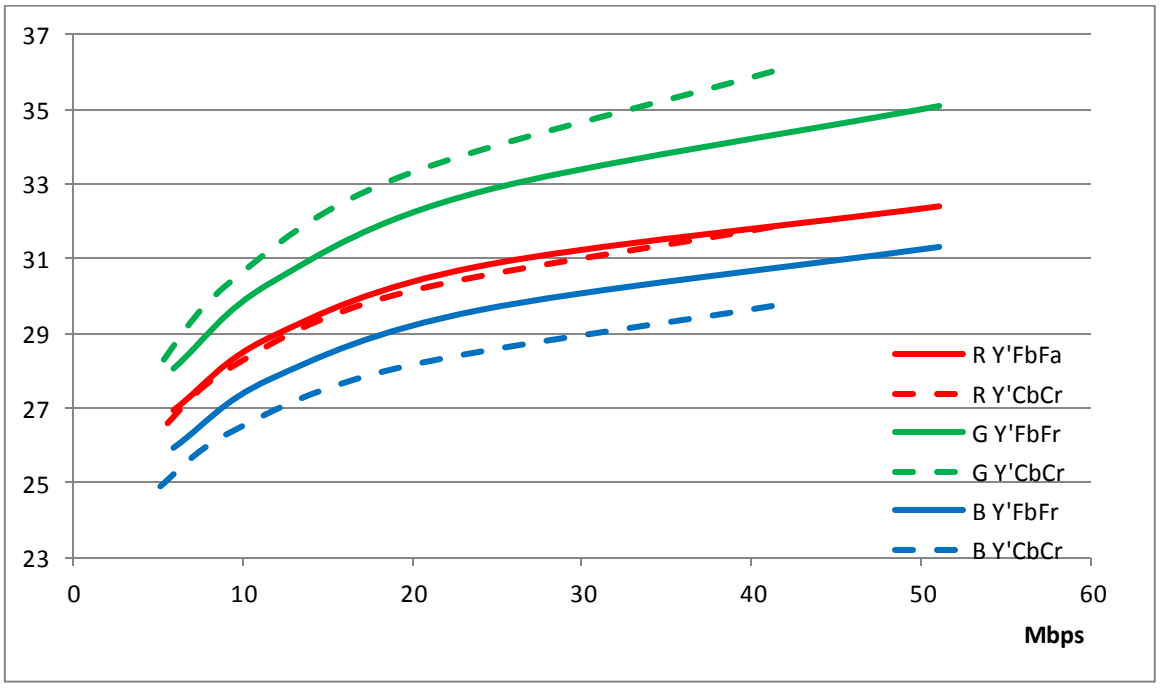

Figure 31 Bitrate distortion curves in dB for "Crowd Run".

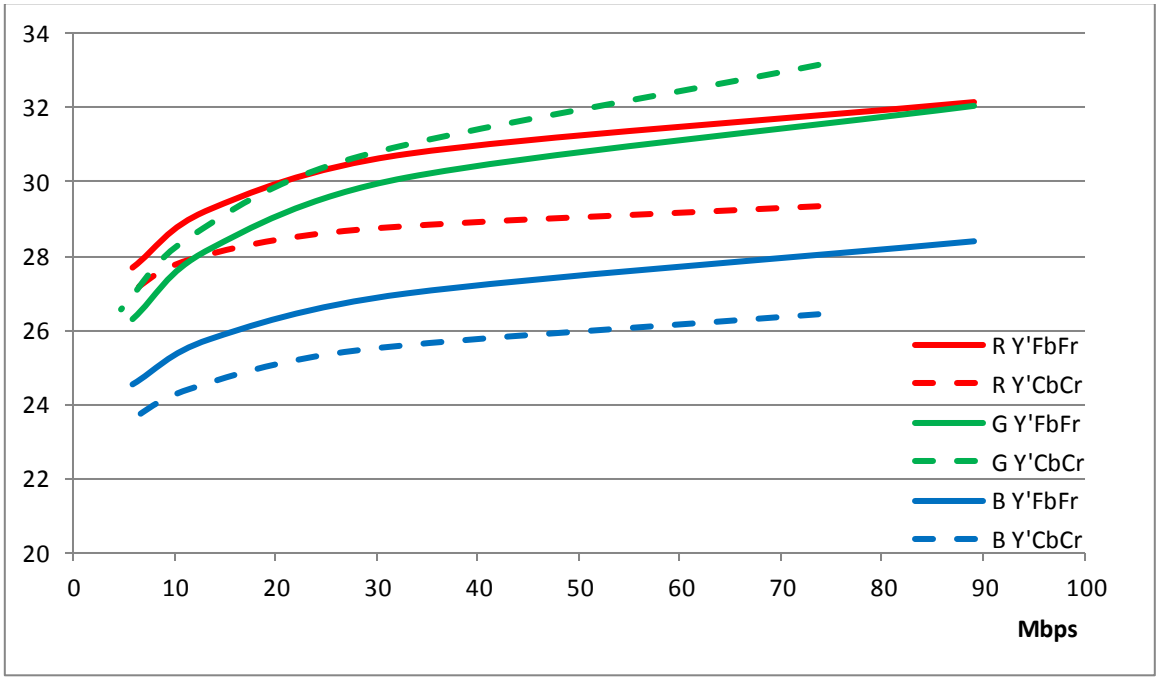

Figure 32 Bitrate distortion curves for "Ducks Takeoff".

In all cases there is a significant loss in the green channel, which means a loss of the luma of the picture, since luma has a high percentage of green (5). Even if other channels improve with regards to Y'CbCr, losses in green channel are perceptually unnoticeable.

An explanation to this effect may lay in the way these color spaces were created: by adapting the color transform to a wide range of selected sequences. Therefore, if the new tested sequences are not representative of the design ones, results may be hardly predictable.

5.4 Content-adaptive color space with PCA

Unlike previous experiments, with this content-adapted color space, PSNR is not computed in R'G'B'. This is because sequences compared are already in the same color space before applying the adaptation, prior to encoding, and after removing the adaptation to the decoded sequence. Furthermore, once adaptation is removed, both sequences will follow the same operations to obtain the R'G'B' color space:

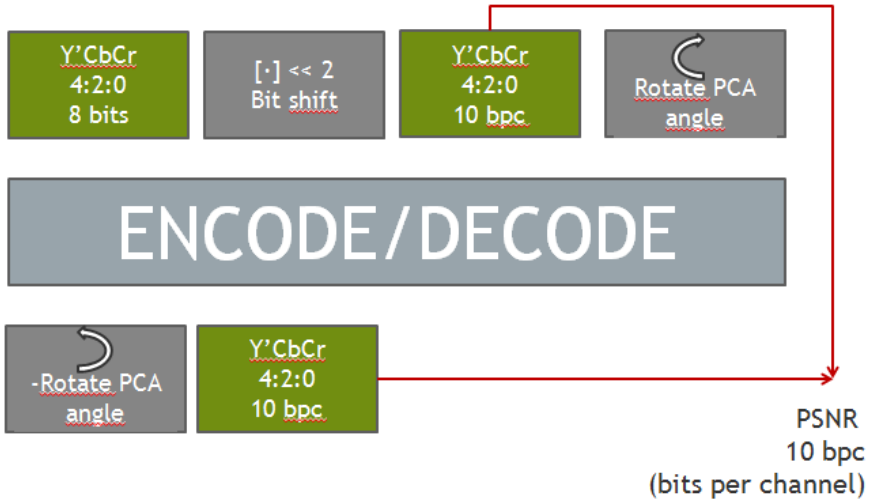

Figure 33 PSNR computation points

After comparing the PSNR between sequences modified with PCA and original sequences, the results are always better or at least equal with sequences PCA processed, as shown in the curves below, for four different sequences:

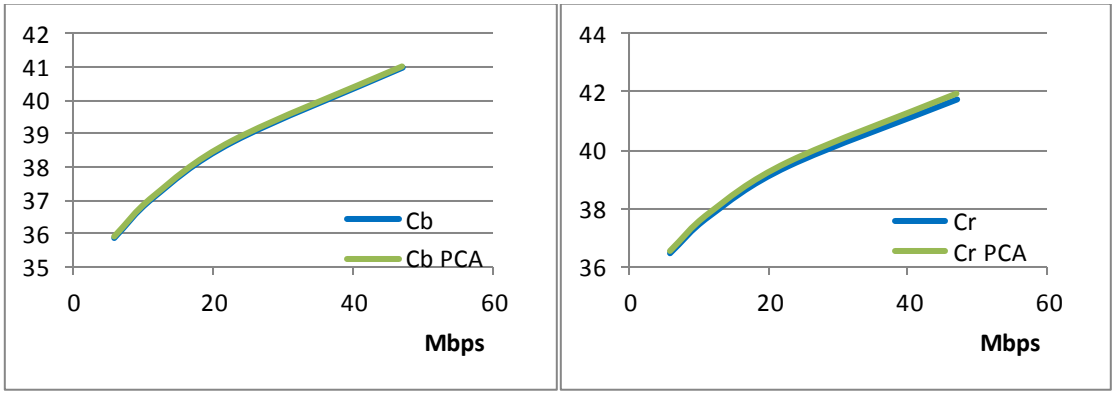

Figure 34 Rate distortion curves in dB for Crowd Run.

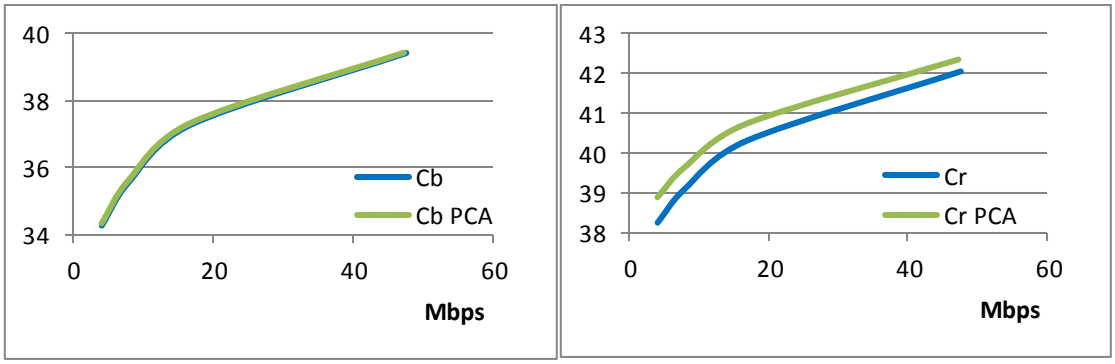

Figure 35 Rate distortion curves in dB for Ducks Takeoff.

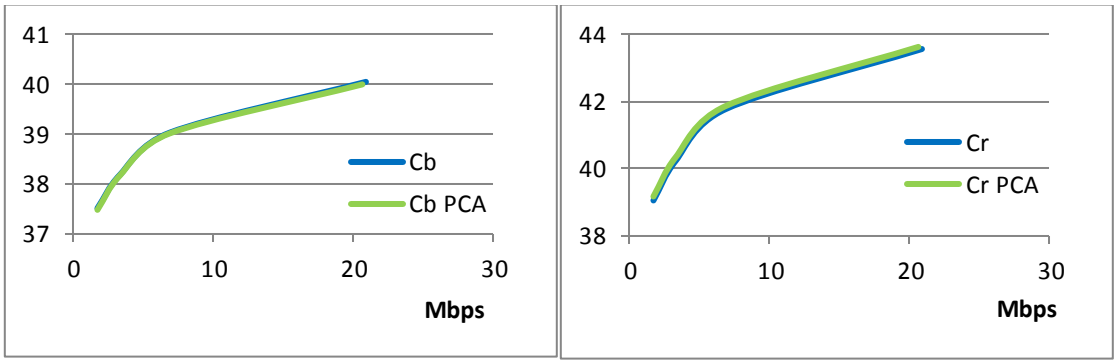

Figure 36 Rate distortion curves in dB for Cactus.

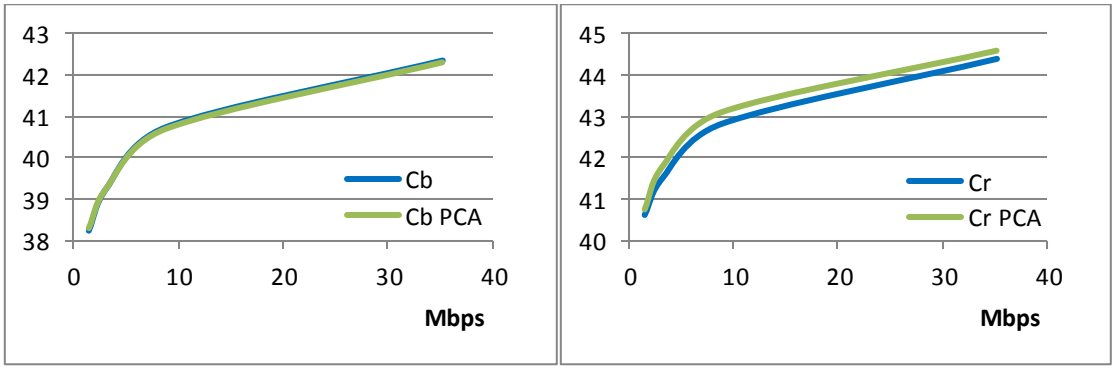

Figure 37 Rate Distortion curves in dB for Terrace.

The results above can be expressed in a table, showing the decrease in bitrate for a given quality when using the adapted color space:

Sequence	Y'	Cb	Cr
Crowd Run	-0.1%	-1,9%	-5.2%
Ducks Takeoff	-0.8%	-2.7%	-25.9%
Cactus	-1.3%	1.9%	-5.2
Terrace	-1.3%	1.4%	-19.2%

Table 3 Decrease in bitrate for a given quality

From Table 3, it is reportedly shown that there are huge improvements in some chroma channels, especially when sequences have a predominant color: in Ducks Takeoff it is blue, and in Terrace it is white. This fact proves this transform to be very efficient on this kind of content. There are also some improvements in luma channels, due to the fact that luma and chroma are not completely de-correlated in this color space transform. There are also some losses in chroma, but the improvement in chroma is always higher than its losses.

The improvement on luma is not very high, but perceptually, to achieve the same improvement on luma than on chroma, chroma PSNR improvement must be about 6 times the improvement on luma, this fact makes even small improvements on luma very important.

6 Conclusions

This project has presented several ways to attempt to improve the video coding quality without changing the way data is encoded. It has been claimed that pre-processing sequences to encode may lead to significant quality improvements.

All the modifications done to sequences prior to encoding, need a fairly good knowledge about video coding principles, even if no coding algorithm is modified.

The first change was the correct application of gamma correction by removing it before down-sampling pictures and re-applying it afterwards. This procedure improves the color fidelity of down-sampled pictures.

Tests started with the aim to improve subjective quality, hence the Lab color space, which resulted on varying qualities depending on the coded content. However, Lab color space has provided higher subjective quality among the tested sequences.

The non reversibility of the R'G'B' to Lab process indicated that reversible color spaces might be a viable alternative, therefore, Y'CoCg and Y'FbFr were tested. Result were not as good as claimed on their respective JVT documents. The facts that caused such results are namely the way these transformations were designed (KLT-based) and the unfairness of the PSNR calculations.

These experiments with reversible color spaces have also an important conclusion: the reversibility feature is not crucial, since once chroma is sub-sampled and resulting data is encoded and quantified, it is already not reversible, which makes losses due to non-reversibility negligible.

Those reversible color spaces lead the final part of this project to adapt the color transformation to each sequence, instead of designing a color space that tries to cover most common cases.

Tests with color space adaptation have lead to impressive results in chroma channels for sequences with a clearly dominant color. Results are also good for general sequence, but not as stunning.

In order to settle down obtained results, more test should be carried out to confirm the improving trend, especially for the adaptation part. Nevertheless, it seems natural that taking into account the special features for a sequence when encoding, may not harm the quality.

Finally, Technicolor has started the process of deposing a patent on the results of this project. The patent will be focused in the way the adaptation is implemented and signalized inside the coder.

From my personal experience, this internship has allowed me to put into practice the knowledge of signal, image and video processing acquired during my degree, and to get familiar with other more domain specific concepts.

Even though I had already worked as an intern at the university, this stay abroad has given me the opportunity to work in a professional research and development environment for the first

time. Moreover, I had the chance to go to France, a well-known country in telecommunications area, and to work with exceptional professionals who let me feel confident.

Last but not least, the change this internship has meant for me culminates in the fact that I have empowered my interest for research. Therefore, next year I will be doing a PhD in adaptive video coding.

7 Further work

7.1 Perceptual color spaces

In this report, the number of tested sequences in Lab color space should be increased to confirm the trend in subjective quality. Nevertheless, results are already consistent with the perceptual properties Lab has over Y'CbCr.

Results of Y'CoCg and Y'FbFr have not been conclusive, since the tested sequences seem to differ too much from the design sequences. A more exhaustive test should be run with more sequences. However, as stated in previous sections, tests have given much lower results than the ones provided in [14] and [15], as the PSNR has been computed in the R'G'B' color space. That fact makes Y'CoCg and Y'FbFr color spaces not appropriate for general use without taking into account the pictures content nature.

7.2 Adaptive color spaces

Results in this field have been encouraging, since there are significant improvements on all tested sequences.

An important point with regards to the gains is that they are always higher for the channel that has the lowest dynamic range. That is, the component that has not been chosen as the main axis of inertia. This is because in the channel that is orthogonal to the axis of inertia, the same bit-depth is used to quantify a less changing variable, therefore as dynamic range is lower, precision is better..

Small losses on some chroma channels have lead to think about another way to improve the adaptation.
Currently, the orthogonality constraint between chroma axes is maintained by PCA construction, which doesn't let the color space to perfectly adapt to the content. Therefore, if this constraint is released, an Independent Component Analysis (ICA) may be carried out.

7.3 Constant Luminance

At the beginning of this report, much emphasis has been put into operating processing in linear space, as the filters assume linear-light data when performing linear operations on such values. However, after down-sampling the picture, all following operations assumed values from linear-light color space, which was not the case for used coding color spaces because as video coding aims at optimizing perceptual quality quantization operates in a perceptual color space (not linear-light). Despite having "non linear data", their respective chroma channels have been down-sampled as if they were linear; it is the way it is mainly done currently in video coding.

This section addresses chroma down-sampling in a linear-light way. To achieve this linearity, the starting point will be RGB with no gamma correction, so the output color spaces will also be linear-light. Only then, chroma downsampling will be carried out, and afterwards gamma-corrected. Pictures must be gamma-corrected before encoding because video coders are designed to work with perceptual color spaces.

The following scheme shows an approach to work with constant luminance:

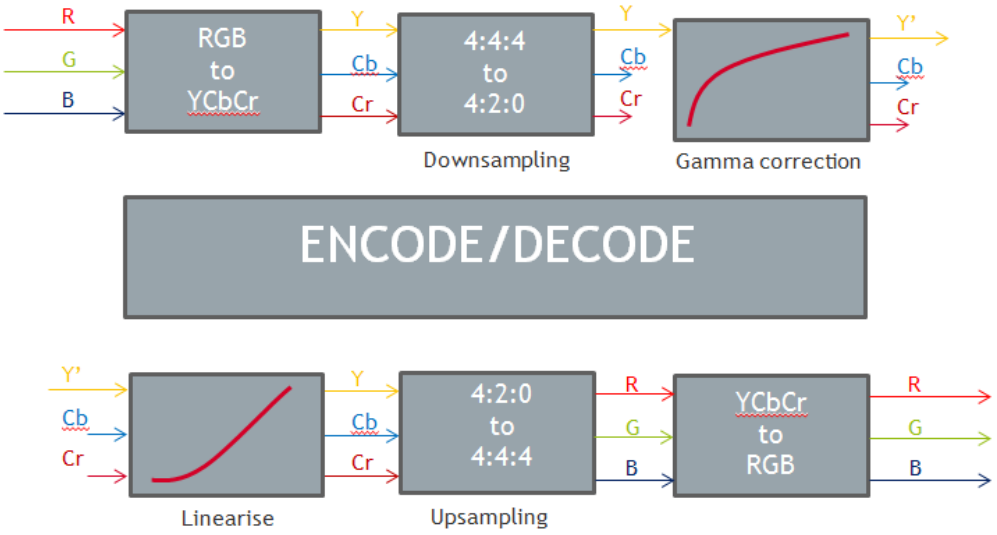

Figure 38 Operating in the linear-light domain

Rec. 2020 for UHDTV let both appear non-constant luminance in order to process data the legacy way and constant luminance to allow proper high quality processing. Next work may be based on the results of this recommendation and the presented work.

8 Annex

8.1 SGI to Planar

This software uses a self-explanatory configuration file which have options for input and output sequences, color space of the output format, input and output bit-depth, down-sampling format (4:4:4 or 4:2:0). It is a front-end to already existing functions in TCH libraries. The software reads a group of sequences in SGI format (a way of storing raw R'G'B' pictures) in 16 bits and converts them into a 4:2:0 sequence in the specified color space with the given bit-depth. A planar image, in opposite to interleaved images, have the three components of the color space stored one next to the other, meaning that, when reading the image, the first third of the data corresponds to the first component in the color space, and so on.

8.2 Planar to RGB

This software was designed with the idea of reverting all operations done by SGI to Planar conversion software. It takes an input sequence, which may have been coded, and the original SGI pictures the sequence was converted from. After that, it processes the sequence using the provided parameters from the configuration file to output an R'G'B' picture. Finally, it computes the PSNR between the decoded picture and the original one in the specified bit-depth.

8.3 Down-sampling functions

In Technicolor's library (TCH) provided for the internship, down-sampling functions were basic since they were carried out by just removing one pixel every two, without any low-pass filtering so that results are subject to spatial frequencies aliasing.. Therefore, my first contribution was to improve those functions by the already mentioned linear half-phased Lanczos3 filter.
The implementation has two variants:

1. A version that operates with pictures whose pixels are interleaved $(R_0 G_0 B_0 R_1 G_1 B_1 R_2 G_2 B_2 \ldots)$, which allows down-sampling RGB pictures:

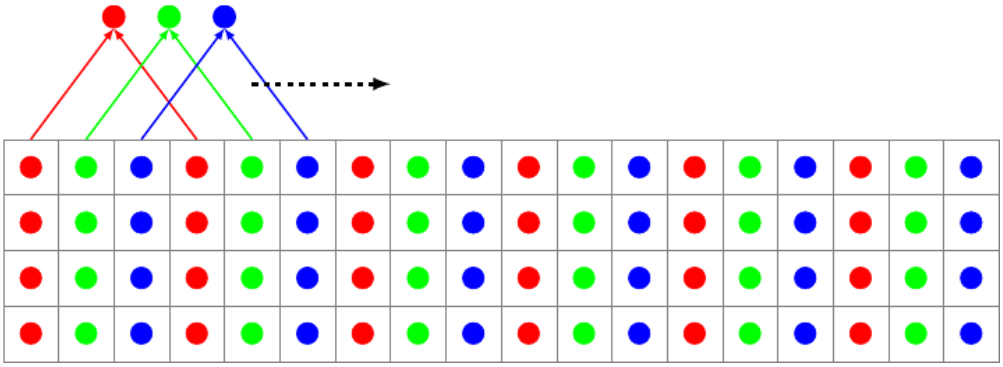

Figure 39 Down-sampling an interleaved picture.

2. A version that operates with pictures whose pixels are in a planar format $(Y_0 Y_1 Y_2 \ldots Cb_0 Cb_1 Cb_2 \ldots Cr_0 Cr_1 Cr_2 \ldots)$, which allows carrying out chroma down-sampling operations without changing the luma:

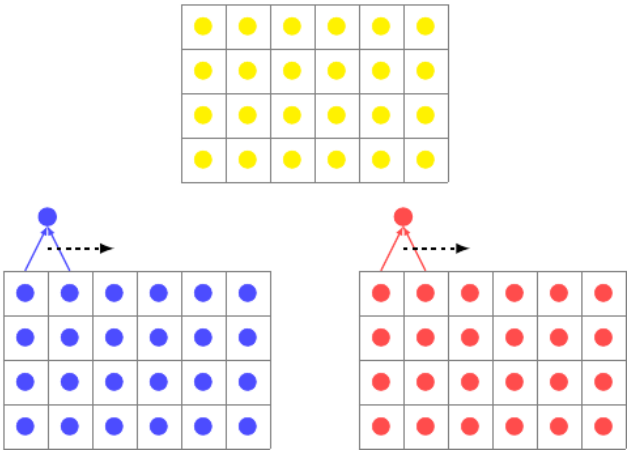

Figure 40 Downsampling a planar picture.

The current implementation mirrors pixels on the borders to apply the filter properly on the pictures edges. The mirroring is done by allocating extra memory and filling the gaps with the appropriate values. However, this is not optimal, as lots of memory allocations are done when processing a sequence, so a solution would be to re-use the actual pixel values instead of copying them (in-place processing).

8.4 PSNR function

The peak signal-to-noise ratio (PSNR) is an objective measure computing the ratio between the maximum possible power of a signal and the power of noise that affects the fidelity of its representation. Even if this metric is subject to many criticisms [19], it is currently the most widely used metric in image and video processing because of its concept and implementation simplicities.

$$PSNR = 10 \cdot \log_{10}\left(\frac{2^{2 \cdot bitdepth} - 1}{MSE}\right) = 10 \cdot \log_{10}\left(\frac{m \cdot n \cdot 2^{2 \cdot bitdepth} - 1}{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left[I(i,j) - K(i,j)\right]^2}\right) \quad (14)$$

The formula above performs the calculation of the PSNR between two pictures, I and K, by computing the mean square error (MSE) of all pixels and normalizing by the size of the picture and the picture dynamic range bit-depth.

TCH libraries lacked a function to calculate this PSNR, then such a function was implemented to enable calculation of the PSNR for both planar and interleaved pictures. Required parameters are the involved pictures in the TCH format and, bit-depth and a float pointer with three fields in which the PSNR will be stored. Those three fields may contain the PSNR values for each

color channel (R'G'B') or the PSNR of the luma and the chroma channels, depending on the type of picture. The function prototype is:

```
void calculatePSNR(tch_picture *img1, tch_picture *img2, float *psnr)
```

Due to the usefulness of the PSNR, it was also created a command line utility, which acts as a front-end for this function.
The arguments to this function are: the two sequences to calculate the PSNR, the initial picture to start calculation from, the number of pictures to process, the picture size, the picture bit-depth and the presented PSNR bit-depth, which must be lesser or equal to the picture bit-depth.
Apart from the sequences, all other parameters are optional and common default values are assumed if not provided. These common values can be seen by launching the program without any arguments.

8.5 Chroma rotation function

In the PCA adaptation of color spaces, a rotation in the chroma axis was needed, and was also implemented in TCH libraries.
The coded function only works to rotate the chroma axis with regards to the luma, although it has been thought to be easily expandable to rotate around other axis, but has not been tested.
This function works entirely with integers, since once the angle is provided, and the sine and cosine of the angle are computed, they are mapped to a 9 bits integer (512 possibilities, since they are non-signed), used to quantify the angle, which ranges between 0 and 90 degrees, allowing a resolution of about 0.2 degrees.
This 9 bit-depth is a safe value, since the pixels have a bit-depth of 16 bits, and all calculations are carried out in 32 bits, which means there is a 7 bits barrier until overflow.

A front-end for this function was also coded, very similar to the PSNR one. The mandatory parameters are the input sequences and the output file where the rotated sequence will be populated. If no angle is provided, zero is assumed, which can be used to change the bit-depth of a sequence by specifying two different bit-depths. Optional parameters include the input and output bit-depths, the start and end picture number to rotate and the picture size.

As with the PSNR function, if these parameters are not provided, launching the program without any parameters will display all options and the default values.

8.6 Conversion between R'G'B' and Y'CoCg or Y'FbFr color spaces

These conversion libraries were done following the same structure as the TCH existing ones to convert between R'G'B' and Y'CbCr.
Due to the reversibility, these new functions need an extra parameter: the bit-depth of the original R'G'B' picture, since the conversion is bit-depth dependent.

9 References

[1] ITU-R, "Recommendation BT.2020 (8/12)," 2012.

[2] C. Poynton, Digital Video and HD Algorithms and Interfaces, San Francisco: Elsevier Science, 2003.

[3] "JCTVC-J1003_d7, High efficiency video coding (HEVC) text specification draft 8".

[4] A. J. Díaz-Honrubia, J. L. Martínez and P. Cuenca, "HEVC:A Review, Trends and Challenges," 2012.

[5] C. Poynton, 16 December 2002. [Online]. Available: http://www.poynton.com/notes/colour_and_gamma/GammaFAQ.html.

[6] ITU-R, "Recommendation BT.709-5 (04/02)," 2002.

[7] D. A. Kerr, "Chrominance Subsampling in Digital Images," 19 January 2012. [Online]. Available: http://dougkerr.net/pumpkin/articles/Subsampling.pdf.

[8] C. Poynton, 28 November 2006. [Online]. Available: http://www.poynton.com/notes/colour_and_gamma/ColorFAQ.html.

[9] "sRGB," Hewlett-Packard, 24 January 2003. [Online]. Available: http://web.archive.org/web/20030124233043/http://www.srgb.com/.

[10] E. Brasseur, 1 April 2012. [Online]. Available: http://www.4p8.com/eric.brasseur/gamma.html.

[11] Avisynth, "Resize," 27 July 2011. [Online]. Available: http://avisynth.org/mediawiki/Resize.

[12] CIE, "Colorimetry, Second Edition," in *Central Bureau of the Commission Internationale de L'Éclairage*, Vienna, Austria, 1986.

[13] H. S. Malvar, G. J. Sullivan and S. Srinivasan, "Lifting-based reversible color transforms for image compression," SPIE, Redmond, 2008.

[14] H. Malvar and G. Sullivan, "YCoCg-R: a Color Space with RGB Reversibility and Low Dyanmic Range," Joint Video Team, 2003.

[15] P. Topiwala and T. Chengjie, "Reversible Integer Color Transform For H.264/AVC," FastVDO LLC, Waikoloa, Hawaii, USA, 2003.

[16] "Cineon Image File Format Draft," [Online]. Available: http://www.cineon.com/ff_draft.php.

[17] G. Sullivan, "Color format upconversion for video display," Joint Video Team (JVT), San Diego, 2003.

[18] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine*, p. 559–572, 1901.

[19] C. Poynton, "A Guided Tour of Color Space," *New Foundations for Video Technology*, 1997.

[20] I. E. G. Richardson, H.264 and Mpeg-4 Video Compression: Video Coding for Next-Generation Multimedia, Chichester, England: John Wiley & Sons, 2004.

[21] H. Y. L. Y. T. K. J. Y. H. D. S. P. a. C. Y. K. Seo Young Choi, "New Color Encoding Method and RGB Primaries for Ultrahigh-Definition Television (UHDTV)," 18th Color Imaging Conference Final Program and Proceedings, Multimedia Lab, SAIT, Samsung Electronics, Nongseo-dong, Giheung-gu, Yongin-si, Gyeonggi-do, South-Korea, 2010.

[22] Y. Tikhomirov, "Intel® AVX Realization of Lanczos Interpolation in Intel® IPP 2D Resize Transform," 2008.

[23] C. Poynton, "YUV and luminance considered harmful," 19 June 1996. [Online]. Available: http://www.poynton.com/papers/YUV_and_luminance_harmful.html.

[24] D. V. Rijsselbergen, "YCoCg(-R) Color Space Conversion on the GPU," 2005.

[25] CIE, "Colorimetry," Comission Internationale de l'Eclairage, Vienna, 2004.

[26] C. Poynton, "The rehabilitation of gamma," 2002.

[27] "Parameter values for the HDTV standards for production and international programma exchange," ITU-R Recommendation BT.709, 1990.

[28] I. Jolliffe, Principal Component Analysis, Second Edition, Springer, 2002.

[29] B. Lindbloom, "Bruce Lindbloom's Web Site," 20 April 2003. [Online]. Available: http://brucelindbloom.com/.

[30] "Lanczos lobs/taps," Avisynth, [Online]. Available: http://avisynth.org/mediawiki/Lanczos_lobs/taps. [Accessed 21 March 2007].