Pros and Cons of the Nonlinear LUX Color Transform for Wireless Transmission with Motion JPEG2000

T. Totozafiny[1], F. Luthon[1], and O. Patrouix[2]

[1]Computer Science Lab LIUPPA, University of Pau, IUT Chateau Neuf 64100 Bayonne, France.

[2]Laboratory for Industrial Process and Services ESTIA, Technopole Izarbel 64210 Bidart, France.

Email: t.totozafiny@estia.fr; Franck.Luthon@univ-pau.fr; o.patrouix@estia.fr

Abstract

We present in this paper an investigation of the nonlinear LUX color transform in a system based on Motion JPEG2000 intended for road surveillance application. The system allows capturing an image, performing motion detection, encoding with the JPEG2000 coder, and transmitting the image towards the decoder through the GSM network at a rate of one image per second. The decoder reconstructs and displays the received image. The LUX color transform is used instead of standard linear color transforms like YUV. The results show that the color rendering of the reconstructed image is clearly improved.

Keywords: JPEG2000 coding, ROI, Reference image, Logarithmic color transform, Wireless transmission

1 Introduction

JPEG2000 is an ISO/ITU-T still image coding standard developed by the Joint Photographic Experts Group (JPEG). JPEG2000 is designed to provide numerous capabilities within a unified system. One interesting option of JPEG2000 is the Region Of Interest coding (ROI), when certain parts of the image are of higher importance than the background.In JPEG2000 Part 1 [1], the core coding algorithm, two linear color transforms are proposed as standard: reversible or irreversible, resulting in lossless or loosely coding respectively. The Logarithmic hUe Extension (LUX) is a non linear color transform, refering to the biological human vision system. It can improve the JPEG2000 coding results [2]. A smart encoding based on JPEG2000 coder was developed in [3]. Each frame is independently coded using the ROI option. Then, data are transmitted toward a decoder where a final image is reconstructed. The transmission is made with two layers. The first layer contains the data, the second layer contains the reference image. But the authors do not explain how to obtain the initial reference. The typical problem of this kind of system is the updating of the Remote Reference Image (RRI). Recently [4], we have developed a new system allowing image transmission through a single channel with very low bandwidth (GSM). The transmission rate reaches the performance of one image per second. The updating of the RRI is triggered when certain conditions are met (linked to the amount of motion areas such as: too many, few or no moving objects). Instead of standard linear color transforms, for example the RGB to $YCrCb$ transform, the LUX transform is used in order to improve the color rendering. Here, we investigate its advantages and drawbacks in our system [4]. This paper is organized as follows: we describe our system in Section 2 and the presentation of the non linear LUX color transform is given in Section 3. Finally, experimental results, discussion and conclusion are given in Sections 4, 5, and 6.

2 Proposed System

2.1 System Description

The system is intended for the transmission of color image series through a wireless GSM network. The image is captured by a static camera. Then, it is JPEG2000-coded and transmitted toward a decoder where the final image is reconstructed and displayed (Fig. 1). First, the system builds a reference image and transmits it towards the decoder. This initialization is necessary in order to reconstruct the final image for the next frame transmission. After the initialization is completed, the reference image is regularly updated. Motion detection is performed in order to obtain an ROI mask, which gives the region of interest for the system. Then, the current image containing only data linked with the mask is coded using the ROI

option of JPEG2000 standard and transmitted towards the decoder. At the decoder side, the image is received and the motion mask is implicitly reconstructed. The final image is built using the available reference image, the motion image and the reconstructed mask. In the spatial domain, a simple pixel substitution is used to build the displayed image. The RRI must be updated according to the scene evolution. Hence, a flag is added in the file header of the bitstream during the image encoding in order to tag the sent image, which can be a reference image (i.e. background image) or a motion image (i.e. motion data). In the first case, the received image is stored as background image on the decoder. Otherwise the image is displayed.

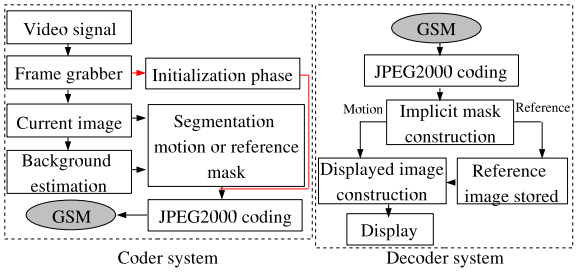

Figure 1: Our coder and decoder system.

2.2 Initialization step

The initialization phase consists in the construction of a reference image. It is a difficult task because it depends not only on the moving objects present in the scene (such as object entering, stopping at and/or leaving the scene, object moving slowly, etc) but also on the natural environment (sun, movements due to wind, illumination, shadows and so on). In a static camera framework, several methods in the computer vision literature were proposed to obtain a reference image during an initialization phase [5]. The proposed system is intended for videosurveillance. The coder system is an embedded architecture working on a PC104 format board where the memory size is restricted. In fact, we use the first order recursive filter :

$$B_{init}(p,t+1) = \alpha_p I(p,t+1) + (1-\alpha_p) B_{init}(p,t) \quad (1)$$

where $B_{init}(p,t)$ and $I(p,t)$ are the intensity values at the pixel location p at time t in the reference image and the current image respectively. $\alpha_p \in [0,1]$ is the learning rate. This model (1) gives good results only when the moving objects are small. Otherwise, the quality of the regions covered by a large object is bad in the reference image obtained. Therefore, it will be useful to improve these regions first.

2.3 Foreground extraction

The motion detection is a binary labelling problem [6]. It consists in putting, on each pixel p of image I at time t, one of the two following label values:

$$e_p = \begin{cases} 1 & \text{if } p \in \text{moving object} \\ 0 & \text{if } p \in \text{static background} \end{cases} \quad (2)$$

In order to carry out the binary labelling we use two observations. For each pixel p, we compute:

- the difference between the reference image and the current image:

$$o_{dr}(p,t) = |B(p,t) - I(p,t)| \quad (3)$$

- the difference between two successive frames:

$$o_{dt}(p,t) = |I(p,t) - I(p,t-1)| \quad (4)$$

To find the most probable label with these two observations, we can use the conditional probability (Bayes theorem). In our application, we used a logical AND in order to cope with the embedded architecture target. We put a threshold θ on both observations o_{dr} and o_{dt} and then we compute the logical AND. The moving object threshold θ is determined automatically, according to an entropy based threshold selection. A morphological gradient filter is applied to improve the binary mask obtained. Then, the current image is compressed using the ROI option with a very low bit rate (9600 bps for single GSM channel) which yields a transmission rate of 1 img/s.

2.4 Reference image updating at ground station

We use the same scheme as developed in [6] to estimate a reference image in the embedded device. For each grabbed frame the background image is updated. The method consists in the modelling of each pixel temporal evolution with K Gaussian distributions ($K = 3$).

2.5 Reference image updating at the remote station

Once the initialization phase is completed, the coder is able to send the entire background image. Thus, the decoder can reconstruct the final image.

At the coder side, the reference image is updated for each frame according to a Gaussian Mixture Model (GMM). The updating of the reference image at the decoder (RRI) should be done regularly.

The basic technique for updating the RRI is based on the transmission of the RRI towards the decoder within a specific period, for example every 4 seconds or every 10 images like in [3]. The RRI must be coded with no ROI option and with a moderate bit rate compression. In our system, the transmission of the complete RRI would slow down the overall transmission rate. For our application, image transmission should be done at a rate of 1 img/s at least, while using a single transmission channel. The complete RRI update takes more than 4 seconds. This latency is not acceptable. In order to keep the image transmission rate close to 1 img/s, we propose a new updating technique of the RRI by pieces. In this scheme, the RRI will be coded like the motion image with a ROI mask. We have chosen a square pattern in order to build the ROI mask. The information in this area will be coded and sent to the decoder for the updating of the background image (Fig. 2).

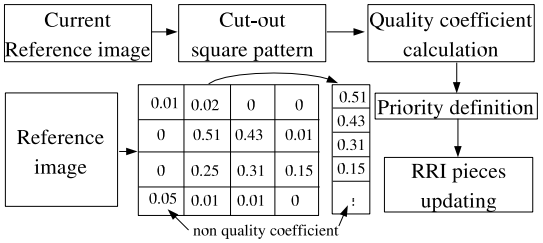

Figure 2: RRI updating strategy.

2.5.1 Regions to be updated

The background image is composed of Nb blocks. Each block will be used as an ROI mask. In order to improve the efficiency of this strategy, we define a parameter $\overline{qc}$ for each area. $\overline{qc}$ represents a non quality coefficient corresponding to the initialization phase (background image creation) or the GMM background estimation. $\overline{qc}_i$ $(0 \leq \overline{qc}_i \leq 1)$ is used to compute a refreshing priority for region i. The region with the highest priority will be updated first.

2.5.2 Local quality coefficient computation

In a given frame, the local quality coefficient for a given block is based on the intersection of the motion ROI mask and the block area. The occupation percentage of the moving object is computed as follows:

Let $Imask$, Ir be the mask of moving object and current reference image respectively, and i be the current index of the treated block.

We can write:

$$Ir = \bigcup_{i=0}^{Nb-1} Irb_i \qquad Imask = \bigcup_{i=0}^{Nb-1} Imaskb_i \quad (5)$$

where Irb, $Imaskb$ are sub-images of Ir and $Imask$ respectively. For each block i, we compute:

$$\overline{lqc}_i = \frac{NonZero(Imaskb_i)}{Nsize^2} \quad (6)$$

where $NonZero()$ is an operator that counts the number of pixels set to 1 in the current $Imaskb$ and $Nsize$ is the size of the square block.

2.5.3 Global quality coefficient computation

At frame t, the quality coefficients are updated according to the local quality coefficients and the previous values of the quality coefficient at frame $t-1$. A threshold θ_{qc} is introduced to determine a changing of given region:

$$\overline{qc}_i(t) = \begin{cases} \max(\overline{qc}_i(t-1), \overline{lqc}_i(t)) \text{ if } \overline{lqc}_i(t) \geq \theta_{qc} \\ (1-\gamma)\overline{qc}_i(t-1) \text{ else} \end{cases} \quad (7)$$

where γ is the learning rate for the background estimation. During the initialization phase for example, with model of Eq.(1), we have : $\gamma = \alpha_p$.

2.5.4 Remote reference updating decision

The region to be updated in remote reference must correspond to the block with the highest $\overline{qc}$. We introduce two thresholding values $\theta high$ and $\theta down$ in order to measure the amount of place that moving objects take in the scene. The high threshold $\theta high$ indicates that a lot of moving objects are observed in the scene while $\theta down$ indicates that few moving objects are observed in the scene. The strategy of RRI updating is engaged according to the state (amount) of moving objects in the scene (many or few). Three cases are considered:

- case 1: $\theta state < \theta down$,
- case 2: $\theta state > \theta high$,
- case 3: $\theta down < \theta state < \theta high$.

where $\theta state$ is the global state of moving objects altogether. It can be computed as:

$$\theta state = \frac{NonZero(Imask)}{dim} \quad (8)$$

where $Imask$ is a mask for the moving object and dim represents the image size.

2.5.5 Blocks pulling

The blocks to be updated are chosen according to the three cases previously stated.

- case 1: we can consider that no interesting object is present in the scene. We propose to use the block with the highest $\overline{qc}$ as the ROI mask for the JPEG2000 encoding. Then, the non-quality coefficient of this block is forced to zero ($\overline{qc} = 0$) in order to flag it as "treated".
- case 2: too many moving objects are detected in the scene, then the updating of the remote reference should not be engaged (only motion is transmitted). But if this configuration lasts too long (for example more than 2 hours), maybe the updating should be performed like in the previous case.
- case 3: there is no easy decision so we choose to force the updating every n frames (typ. value of $n = 20$) in order to improve the global quality of the displayed image. Only the blocks with a $\overline{qc}$ greater than θ_{qc} will be updated. Then, the non-quality coefficient of this block is forced to zero ($\overline{qc} = 0$) in order to flag it as "treated".

3 Logarithmic color transform

3.1 LUX color space introduction

In the JPEG2000 coding part one, the multicomponent color transform is carried out using a linear color transform, either $RGB \rightarrow YUV$ or $RGB \rightarrow YCrCb$, allowing a reversible coding as well as an irreversible coding respectively. The conversion matrices, forward and inverse, of the irreversible coding are:

$$T = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{bmatrix}$$

$$A = T^{-1} = \begin{bmatrix} 1.0 & 0 & 1.402 \\ 1.0 & -0.34413 & -0.71414 \\ 1.0 & 1.772 & 0 \end{bmatrix}$$

The nonlinear LUX color transform (for Logarithmic hUe eXtension) is inspired by the biological human vision system (Fig. 3). The human eye is a natural compression system and the compression is done nonlinearly: the cone transduction function may be described by a loglike function, while the action of horizontal and bipolar cells (weighted average and weighted difference resp.) may be modelled by a linear matrix like T. The logarithmic image processing (LIP) model is known to yield an impressive contrast enhancement [7].

This one LIP is basically defined in the continuous case by three equations: a transform f from the intensity space (variable x) to the space of tones, an isomorphism ϕ from the space of tones into a logarithmic space (variable y) and an inverse isomorphism ϕ^{-1} (for more details, see [8]).

The LUX color space extends the LIP model to handle colors (i.e., $YCrCb$). For that purpose, only the composition function $\Phi = \phi \circ f$ is of practical interest. The isomorphism Φ provides a logarithmic transform normalized by the maximum transmitted light:

$$\begin{array}{rcl} \Phi : x & \rightarrow & y = M \ln\left(\frac{x_0}{x}\right) \\ \Phi^{-1} : y & \rightarrow & x = x_0 \exp\left(-\frac{y}{M}\right) \end{array}$$

where $x \in]0 \cdots x_0]$ is a continuous gray level, $x_0 \in]0 \cdots M]$ is the maximum transmitted light and M is the dynamic range of gray levels (typ. $M = 256$ for 8-bit coding).

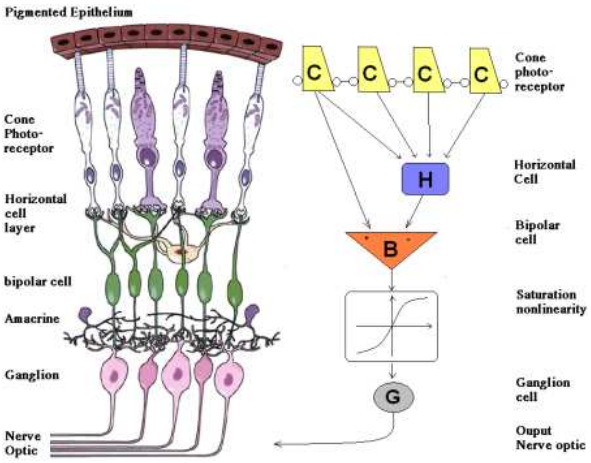

Figure 3: Biological analogy.

From a mathematical point of view, the diagram below helps understand how the LUX is built by composition of functions:

$$(R, G, B) \overset{Norm}{\longrightarrow} (r, g, b) \overset{\Psi}{\longrightarrow} (l, u, x) \overset{Denorm}{\longrightarrow} (L, U, X) \quad (9)$$

3.2 LUX forward transform

First, color components are normalized. To adapt the dynamic range, the normalization consists in two steps: translation of dynamics and rescaling of the quantities w.r.t. their maximum values. Since $(R, G, B) \in [0, M[\times [0, M[\times [0, M[$ in the discrete case, translated quantities (r, g, b) are defined to stick to the interval $]0, M]$ as required by the LIP theory, yielding normalized quantities $(\overline{R}, \overline{G}, \overline{B})$. Let (r_0, g_0, b_0) be the maximum values of (r, g, b). We have:

$$r = R+1 \quad g = G+1 \quad b = B+1$$
$$\overline{R} = \frac{r}{r_0} \qquad \overline{G} = \frac{g}{g_0} \qquad \overline{B} = \frac{b}{b_0}$$

Then the nonlinear transform Ψ which is the composition of three functions $\Phi^{-1} \circ T \circ \Phi$ may be computed as:

$$\begin{aligned} l &= \overline{R}^{t_{11}} \overline{G}^{t_{12}} \overline{B}^{t_{13}} \\ u &= \overline{R}^{t_{21}} \overline{G}^{t_{22}} \overline{B}^{t_{23}} \\ x &= \overline{R}^{t_{31}} \overline{G}^{t_{32}} \overline{G}^{t_{33}} \end{aligned}$$

where t_{ij} are coefficients of matrix T.

So far, this logarithmic model works only for positive values of chrominances. To take account of the possibly negative values of the chromatic components, we can use the following restriction:

$$\overline{u} = \begin{cases} \frac{u}{2} & \text{if} \quad u \leq 1 \\ 1 - \frac{1}{2u} & \text{if} \quad u > 1 \end{cases}$$

$$\overline{x} = \begin{cases} \frac{x}{2} & \text{if} \quad x \leq 1 \\ 1 - \frac{1}{2x} & \text{if} \quad x > 1 \end{cases}$$

Finally, a proper denormalization step yields the three nonlinear color components $\in [0, 255]$:

$$\begin{aligned} L &= Ml - 1 \\ U &= M\overline{u} - 1 \\ X &= M\overline{x} - 1 \end{aligned} \tag{10}$$

To simply the Eq.(10), we use the following expression to compute the LUX forward transform:

$$\begin{aligned} L &= (R+1)^{t_{11}} (G+1)^{t_{12}} (B+1)^{t_{13}} - 1 \\ U &= (R+1)^{t_{21}} (G+1)^{t_{22}} (B+1)^{t_{23}} - 1 \\ X &= (R+1)^{t_{31}} (G+1)^{t_{32}} (B+1)^{t_{33}} - 1 \end{aligned} \tag{11}$$

3.3 LUX inverse transform

The inverse of Eq.(11) is :

$$\begin{aligned} R &= (L+1)^{a_{11}} (U+1)^{a_{12}} (X)^{a_{13}} - 1 \\ G &= (L+1)^{a_{21}} (U+1)^{a_{22}} (X)^{a_{23}} - 1 \\ B &= (L+1)^{a_{31}} (U+1)^{a_{32}} (X)^{a_{33}} - 1 \end{aligned} \tag{12}$$

where a_{ij} are coefficients of inverse matrix $A = T^{-1}$. In our experiment, we use (11) and (12) to perfom the forward and inverse LUX color transform respectively.

4 Experimental results

The system has been implemented into the embedded device (working with a PC104 format board). This one is mainly based on a camera, a CPU and a wireless transmission device. The system can currently send an image at a rate of one image per second through the RS232 port restricted to 9600 bps. For experimental tests we used 2 PCs. For JPEG2000 encoding we used the Kakadu SDK (`http://www.kakadusoftware.com`). Then we changed the standard linear color transform by LUX. We can see on Fig. 4 that LUX gives a better color rendering (rear light on left) than the standard color transform (here we have used the irreversible one). Fig. 5 shows a result of the reconstructed image at the decoder side. The size of the image is 320×240 with 8 bit-coding. It is coded on a low bit rate compression $0.1bpp$. The PSNR is used to evaluate the result of reconstructed image using the standard color and LUX. The PSNR computation for LAPS[1] video is given on Fig. 6. We can see that the LUX color transform result is better than the standard color tranform. We gain between 1 and $5dB$, but the average computation time is five times more, as shown Tab. 1.

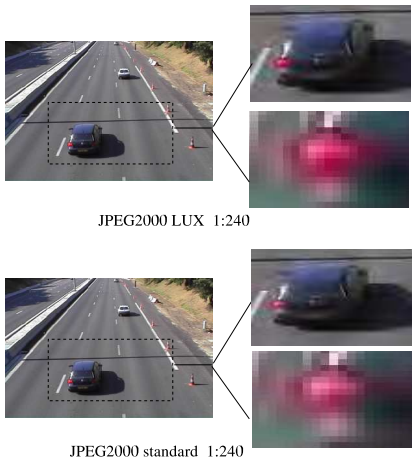

Figure 4: A particular region coded with JPEG2000 ROI option at bit rate $0.1bpp(1:240)$: nonlinear color transform $RGB \rightarrow LUX$ (top) ; standard color transform $RGB \rightarrow YCrCb$ (bottom).

(a) (b)

Figure 5: Reconstructed images: a) standard color transform ($RGB \rightarrow YCrCb$); b) LUX color transform.

[1] Thanks to LAPS laboratory `http://www.u-bordeaux1.fr` for providing a sequences images; We have also tested our algorithm with the sequences available at `http://i21www.ira.uka.de/image_sequences/`

Table 1: Average computation time (in ms) under Linux system.

Color Transform methods	PC AMD 1.4GHz	Embedded PC104 Celeron 800GHz
Standard	180	250
LUX	600	900

Figure 6: PSNR computation result of LAPS video, compressed and transmitting image at $1:240$ ratio compression and $1 img/s$ respectively

5 Discussion

We note that LUX is not a standard transform of the JPEG2000 still image coding scheme. The user that hopes to use an available hardware JPEG2000 coder on the market is wedged because the LUX color transform is not implemented. To accomplish the inverse color transform, in general the maximal values of the colors dynamic range will be sent to the decoder separately from the codestream coded by JPEG2000. Then the computation time of LUX is very expensive (see Tab. 1). Nevertheless, the logarithmic hue extension LUX improves the rendering quality of the image in high compression applications, as shown with PSNR computation (Fig. 6). In road surveillance applications, obtaining the best color restitution is always an advantage. For example when one car is overtaking another one, capturing the flashlights is essential. Then, finding a good compromise between the LUX computation time and the color rendering may be interesting. As the power of processors increases (Moore law), the computation time of the nonlinear color transform LUX could be considerably decreased in the future. In the perspective, the fast implementation of LUX computation by means of eather software or hardware will be interesting. We would use JPEG2000 non compliant in this case.

6 Conclusion

The LUX computation time is expensive and it is not implemented in the JPEG2000 coding standard. Yet, in a road surveillance context we have developed a system allowing an image transmission at a rate of one image par second through the GSM network. The image is coded on a low bit rate compression $0.1bpp(1:240)$. The rendering of color can be improved using nonlinear color transforms like the logarithmic hue extension color transform.

References

[1] ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG8), "JPEG2000 Part I Final Committee Draft Version 1.0," tech. rep., Mars 2000.

[2] F. Luthon and B. Beaumesnil, "Color and R.O.I. with JPEG2000 for wireless video-surveillance," in *IEEE Int. Conf. on Image Processing, ICIP'04*, Singapore, October 2004.

[3] J. Meessen, C. Parisot, C. Lebarz, D. Nicholson, and J. F. Delaigle, "Smart Encoding for Wireless Video Surveillance," in *SPIE Proc. Image and Video Communications and Processing*, San Jose, CA, January 2005.

[4] T. Totozafiny, O. Patrouix, F. Luthon, and J. M. Coutellier, "Dynamic Background Segmentation for Remote Reference Image Updating within Motion Detection JPEG2000," in *International Symposium on Industrial Electronics (ISIE2006)*, 9-13 July, ETS-Downtown Montréal, Canada 2006.

[5] H. Wang and D. Suter, "A Novel Robust Statistical Method for Background Initialization and Visual Surveillance," in *ACCV*, pp. 328–337, 2006.

[6] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *CVPR*, pp. 246–252, 1999.

[7] S. Shah and M. D. Levin, "Visual Information Processing in Primate Cone Pathvays," *Part I A Model. IEEE Trans. on System, Man and Cybernetics*, no. 26, pp. 259–274, 1996.

[8] M. Jourlin and J. Pinoli, "Image dynamic range enhancement and stabilization in the context of the logarithmic image processing model," *Signal Processing*, no. 41, pp. 225–237, 1995.