LUX Color Transform for Mosaic Image Rendering

Franck Luthon, Brice Beaumesnil, Nicolas Dubois

LUX Color Transform for Mosaic Image Rendering

Franck Luthon, *Senior Member, IEEE*, Brice Beaumesnil, and Nicolas Dubois
University of Pau and Adour, Anglet, France
Computer Science Department, Email: Franck.Luthon@univ-pau.fr

Abstract— **In this paper, we present an image mosaic application and investigate the color rendering performances obtained with various color spaces, among which the nonlinear LUX transform, that is based on a logarithmic image processing model and on biological evidence about the retina processing within the human visual system. We focus on the color matching step for automatic forming of the mosaic image (without any user interaction nor color correction post-processing). We derive a semi-normalized version of the LUX transform, and test its benefits for the mosaic color rendering. The LUX nonlinear color space exhibits a nice behavior of adaptation to light context (logarithmic compression), which is in agreement with the biological processing made by the cones in the eye retina. We compare its behavior for color discrimination and color adaptation with other models proposed in the literature. We show qualitative and quantitative results that demonstrate the improvement achieved by the LUX transform (in terms of color contrast and peak signal to noise ratio PSNR) for building the mosaic.**

Keywords— **Mosaic, Color Space, LUX, Color Distance, Logarithmic Hue, Color Matching.**

I. Introduction

The principle of image mosaic is the following [1], [2]: given an input image (target) and an input video, an output image is built, that is a mosaic mapping looking like the original image. But instead of being made of a lot of little patches with uniform color as is usual for ordinary mosaics, the mosaic is made of a lot of little frames properly chosen from the input video in order to get a color rendering as close as possible to the target image. A typical result is shown in Fig. 1. From a certain distance, one sees the original image, but with a closer look, one sees snap-shots of the film. This technique exploits layered imagery, which is

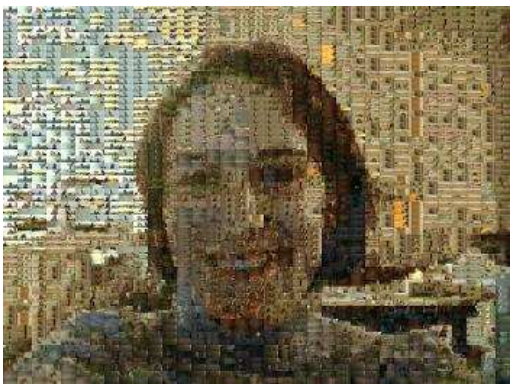


Fig. 1. Photomosaic made of frames from the French movie "Les Visiteurs"

inspired both by human portraits of 16th century painter Guiseppe Arcimboldo, and by paintings of the impressionist mouvement (small brush strokes of various colors). Image mosaics might be used for commercial or artistic purpose : posters, advertisement for new movies, cover pages, graphical art, postcards, personal pictures, etc. The keypoint in creating image mosaics lies in the matching step for choosing the best video frame to render appropriately the local pixel color of the target image. Indeed it impacts both computation cost and aesthetical effect [3]. Therefore, the choice of a proper color space is important: this is the first contribution of this paper. Moreover, the computation time of the color matching algorithm should be short, in order to have a fast enough creation procedure, which is the second contribution of this paper. Other issues, not dealt here, but addressed by researchers in the literature are the use of tiles with varying size and orientation (irregular dense tiling) in order to preserve also edge information [4], tiles and container of arbitrary shapes to make jigsaw image mosaics [5], or creation of video mosaics where not only color spatial coherence but also temporal coherence must be maintained [6].

The paper is organized as follows: in section II, we explain the principle of mosaic building, based on the optimization of a color matching algorithm. Section III describes the LUX transform used for color processing. Finally, section IV gives the computation cost and exhibits comparative results before the conclusion in section V.

II. Mosaic Processing

A. Principle of Mosaic Creation

The first step is to resize the original image, to cut it into pieces of equal size proportional to the input video format, and to store them as a matrix of subimages (top left in Fig. 2). Then, for each subimage, one looks in the video for the frame that is closest in terms of color resemblance. For that purpose, a color distance between two images must be computed in a well chosen color space (see sections II-D and II-E). Thus, the mosaic program manipulates three objects: the set of initial subimages that do not change (input subimages); the set of resulting mosaic little frames taken from the video (initialized in black); a 2D table storing the current computed distances between initial subimages and resulting mosaic little frames (this table is initialized with a value corresponding to the maximum possible distance, e.g. $d_{max} = 200$ in the example shown in Fig. 2). The algorithm is made of two processing loops for computing the distance for each image pair (input subimage and video frame). One loop scans the frames in the film and the other one scans the matrix of input subimages. For a given pair, if the distance is lower than the one currently stored in the 2D table, then the mosaic is up-

978-1-4244-6723-5/10/$26.00 ©2010 IEEE

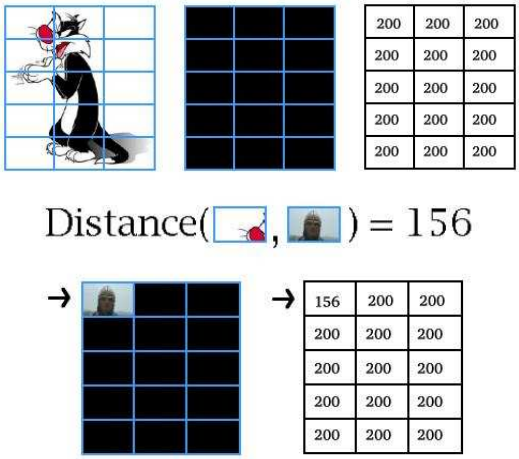


Fig. 2. Principle of image mosaicing: target image is cut into subimages (or tiles) that are replaced by selected frames from a video.

dated with the current video frame and the new distance value is stored in the 2D table.

B. Optimization: Video Frame Discarding

To get a faster rendering, two solutions may be adopted: either to simplify as much as possible the algorithm in the main loop (mainly optimization of the resizing, of color format conversion and of distance computation) and/or to decrease the number of iterations. As regards the iteration number reduction, certain frames of the video should be discarded without any further computation. Throwing away every two image would be very simple but not very clever nor efficient. The question is then how to estimate in advance the current frame potential quality. By quality, we mean a good adequacy (i.e. color likeness) with one of the subimages in the input image. To answer this question, we start from the following fact: movies are often made of a succession of sequences (e.g. a man that is speaking) separeted by sharp cuts (change of camera, change of scene, etc). Within each sequence, video frames are very similar (without speaking of the temporal redundancy from the quantitative viewpoint of information theory). We may therefore assume that if the first frame of a sequence is "very bad" in some sense, the following frames will be bad also, and this until the end of that sequence. Then two problems arise: how to detect a change of sequence or cut (i.e. how to know the length of a sequence) and what is a "very bad" frame ?

- *Length of a sequence:* since our mosaic application does not decode by itself the movie (it uses OpenCV instead), one can not use the key-frames of the codec. A solution is then to estimate the sequence length: since a sequence lasts rarely less than half a second, and in order not to discard potentially interesting frames, we adopt the following simple scheme: we reject without any computation all batches of 13 video frames that immediately follow every "very bad" frame encountered during the processing.
- *Bad frame:* What do we mean by very bad frame and why ? If a frame is simply bad (but not very bad) and the sequence gets better (as regards color likeness of course), the following frames may become "good". A good frame is a frame of the video that may be selected by the algorithm for taking place inside the final mosaic image, *i.e.* its distance to one of the original subimages is lower than the corresponding distance currently stored in the 2D lookup table (L.U.T). Conversely, a bad frame is a frame that has all its distances to all input subimages greater than the ones currently stored in the L.U.T. Hence a very bad frame has all its distances much bigger than the ones in the LUT. Stated in other words, a very bad frame is a frame for which all the differences between its distances to the input subimages and the distances currently stored in the 2D-table are greater than a certain threshold. The threshold θ is chosen proportional to the subimage dimensions: $\theta = k \times L \times C$. Here we simply take $k = 1$.

C. Unicity of Frame Placing

The algorithm described so far works, but it does not avoid trivial solutions that consist in using many times the same video frame inside the mosaic. This would lead to uniform areas with little variations, built with the same little images, which is not very interesting for a nice color rendering in the application. Therefore we introduce a unicity constraint: a frame from the input video should occur only once in the final mosaic. This constraint is not so easy to take into account: when a frame may suit at various places in the mosaic, one must choose which place will actually get it.

- *First Found Placing:* This technique consists in exiting the comparison loop as soon as the video frame has found a place in the mosaic. Its main advantage is to be very fast. It even speeds up the algorithm. But the results are poor since the first encountered frames are favoured: no qualitative criterion is taken into account.
- *Best Distance Placing:* This method is the more intuitive one: the comparison loop looks for the position inside the mosaic that gives the best distance. The computing complexity is the same as for the initial algorithm (it is actually faster since there is only one single copy of a frame in the mosaic, whereas in the first algorithm, the same frame could be copied many times in the mosaic). In the current implementation, it is even better since we use memory swapping (no recopy is done). Nevertheless, the drawback of this technique is that it searches for the best possible local quality at the expense of the image global quality.
- *Most Urgent Placing:* Here, one tries to maximize the improvement instead of the quality. Improvement is defined as the difference between the old and the new distance. For instance, with our sample image (Fig. 2), the improvement for the first subimage block is $\Delta = 200 - 156 = 44$. Since only one single subimage is changed at a time, the global improvement is equal to the local improvement on that image. Compared to the previous case, the algorithm is only slightly changed: instead of looking for the place yielding the best distance, one looks for the place giving the best improvement. In addition to yielding better results than the previous one, this method also guarantees that all input

subimages will be assigned one frame of the video, which was not sure with all previous techniques.

• *Most Urgent Placing with Stack:* However, all the previous techniques are not optimal in the sense that there is no guarantee that for a given video frame, the best possible place inside the mosaic will be found. Indeed, there may be alternative placings for this frame. Therefore, if this video frame turns out to be removed and replaced by a better one in the running of the selection process, it may well be that this frame, instead of being suppressed, could be placed somewhere else while increasing the improvement. To handle this situation during the placing search, one can memorize with each frame the list of all its alternative positions (candidate places) and their respective computed distances. Then, when a previous frame has just been replaced, one scans its candidate-list (or stack) to check if there were not another position for this frame, that would improve the quality. If there are more than one possible position, the most urgent placing procedure is applied. The image is then replaced and this algorithm must be run again, recursively. This technique avoids losing frames that could advantageously be placed elsewhere. But it can be very slow since the position-list (size of the stack) may be huge. In order to avoid this bottleneck, the size of the stack is limited and only the best distances are stored in the stack.

D. Color Distance

The problem to address is: how to compute the proper color distance between two subimages ? In the literature, there exist many algorithms for measuring the correlation between images but two key issues are the computation cost (long processing time to choose the optimal frame) and the choice of the best color space in which to apply this metrics. Indeed a two hours video at $25fps$ yields 180,000 frames. For an input image cut into 15 pieces only (as the example in Fig. 2), this induces 2,700,000 distances to compute. Here, we use the simplest measure that is computationaly efficient, namely the sum of absolute differences (SAD, also known as Manhattan distance) of the three color components of each pixel (either Red, Green, Blue or any other integer triplet depending on the color space chosen, like YCrCb, YUV, YIQ ...) The distance between two images A and B of size $N = L \times C$ is then given by:

$$d(A,B) = \sum_{i=1}^{3} \sum_{n=1}^{N} |A_i(n) - B_i(n)| \qquad (1)$$

where n is the pixel index and i the color component index.

E. Color Spaces: YCrCb and Others

As regards the choice of the color space, the RGB (red, green, blue) space is not best suited for our purpose. If RGB is the technological choice adopted for monitors, YUV or YCrCb color spaces are preferred for TV standards, video cameras, JPEG compression or MPEG codecs (color-opponent coding with one luminance channel Y plus two chrominance channels). Indeed the eye is sensitive to the average quantity of light: $Y \approx (R+G+B)/3$ and to color differences: $U \approx R - Y$; $V \approx B - Y$. The $YCrCb$ video format is a linear combination of RGB components. The coefficients are standard-specific. For example, we may consider the following matrices:

$$T_1 = \begin{bmatrix} 0.3 & 0.6 & 0.1 \\ 0.5 & -0.4 & -0.1 \\ -0.2 & -0.3 & 0.5 \end{bmatrix} ; \; T_2 = \begin{bmatrix} 0.3 & 0.6 & 0.1 \\ 0.7 & -0.6 & -0.1 \\ -0.3 & -0.6 & 0.9 \end{bmatrix} \qquad (2)$$

This may be written in a generic vector form as: $[Y,U,V]' = T.[R,G,B]'$, where $'$ denotes transposition and T is a 3×3 matrix of chromatic coefficients. In fact, any matrix $T = [t_{ij}]$ corresponding to one of the various TV standards like YIQ, NTSC, PAL, YUV might be used as well, provided that the coefficients verify:

$$\sum_{j=1}^{3} t_{1j} = 1 \qquad \sum_{j=1}^{3} t_{2j} = 0 \qquad \sum_{j=1}^{3} t_{3j} = 0 \qquad (3)$$

But from a perception point of view, which is our concern, biological and psychovisual studies have proven that the eye works nonlinearly with color components (cf. logarithmic compression realized by the transfer function of the cones in the retina). It implies for example that the eye is more sensitive to variations for low intensity (dark) context than for high intensity (bright) context. The HSI color space (hue, saturation, intensity) is a typical nonlinear (angular) transform of the RGB space. It has proven to be suited for color processing since it is more related to psychovisual perception. However, the computation of the hue is very sensitive to noise, since it is based on a nonlinear function applied to a ratio of color differences. Another nonlinear color space called LUX was proposed in [7], that proved to be relevant for image segmentation and compression. In what follows, we use this color space to compute color distances as given by Eq. (1).

III. Nonlinear LUX color transform

A. Presentation of LUX Color Space

Here, we investigate the use of the original nonlinear transform, called the LUX transform, that already proved to be efficient for color segmentation [7] and color compression [8]. This color transform originates both from biology [9] and mathematics [10]. The nonlinear color space is based on a logarithmic transform. The idea of introducing a logarithmic non linearity is in accord with the human visual system (Fig. 3): 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. LUX space is inspired not only by biological considerations, *i.e.,* cone distribution in the fovea and nonlinear transduction of cones followed by bipolar cell differencing in the retina, but also by a mathematical framework, namely the logarithmic image processing (LIP) model, known to yield impressive contrast enhancement [11]. The LIP theory was developed for gray level images. The LIP model is basically defined in the continuous case by three equations:

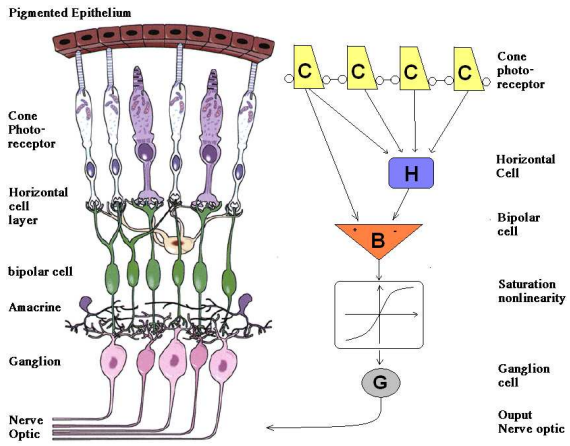


Fig. 3. Biological analogy (*cf.* Fig. 6 in [9]).

a transform f from the intensity space (variable x) to the space of tones, an isomorphism ϕ from the space of tones onto a logarithmic space (variable y) and an inverse isomorphism ϕ^{-1} (expressions are not given here, the reader is referred to [10] for details). For our purpose, only the composition function $\Phi = \phi \circ f$ is of pratical interest. The isomorphism Φ provides a logarithmic transform normalized by the maximum transmitted light:

$$\begin{aligned} \Phi : x &\rightarrow y = M \ln\left(\frac{x_0}{x}\right) \\ \Phi^{-1} : y &\rightarrow x = x_0 \exp\left(-\frac{y}{M}\right) \end{aligned} \tag{4}$$

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

The LUX color space (for **L**ogarithmic h**U**e e**X**tension) extends the LIP model to handle colors (*i.e.,* $YCrCb$) as well. From a mathematical viewpoint, the diagram below helps understand how the LUX color space is built by composition of three functions:

$$(R,G,B) \overset{norma}{\longrightarrow} (\overline{R},\overline{G},\overline{B}) \overset{\Phi^{-1}\circ T\circ\Phi}{\longrightarrow} (l,u,x) \overset{denorma}{\longrightarrow} (L,U,X)$$

• First, the color components are normalized. The normalization consists in two steps to adapt the dynamics: translation of dynamic range (Eq. 5) and rescaling of the quantities w.r.t. their maximum values (Eq. 6). Indeed, since $(R,G,B) \in [0,M[\times[0,M[\times[0,M[$ in the discrete case, one has to define translated quantities (r,g,b) to stick to the interval $]0,M]$ as required by the LIP theory, and also normalized quantities $\overline{R},\overline{G},\overline{B}$. Let (r_0,g_0,b_0) be the maximal values of (r,g,b). We have :

$$r = R+1 \quad g = G+1 \quad b = B+1 \tag{5}$$

$$\overline{R} = r/r_0 \quad \overline{G} = g/g_0 \quad \overline{B} = b/b_0 \tag{6}$$

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

$$\begin{aligned} l &= \overline{R}^{t_{11}}\overline{G}^{t_{12}}\overline{B}^{t_{13}} \\ u &= \overline{R}^{t_{21}}\overline{G}^{t_{22}}\overline{B}^{t_{23}} \\ x &= \overline{R}^{t_{31}}\overline{G}^{t_{32}}\overline{B}^{t_{33}} \end{aligned} \tag{7}$$

where t_{ij} are coefficients of matrix T.

• *Chromaticity range testing :* So far, this logarithmic model works only for positive values of chrominances. To take account of the possibly negative values of the chromatic components, one has to consider also the opposite formulae and implement a test:

$$\begin{aligned} \overline{u} &= \begin{cases} \frac{u}{2} & \text{if } u \le 1 \\ 1 - \frac{1}{2u} & \text{if } u > 1 \end{cases} \\ \overline{x} &= \begin{cases} \frac{x}{2} & \text{if } x \le 1 \\ 1 - \frac{1}{2x} & \text{if } x > 1 \end{cases} \end{aligned} \tag{8}$$

• Finally, a proper denormalization step yields the three nonlinear color components in the range $[0,M[$ i.e. $[0,255]$:

$$\begin{aligned} L &= Ml - 1 \\ U &= M\overline{u} - 1 \\ X &= M\overline{x} - 1 \end{aligned} \tag{9}$$

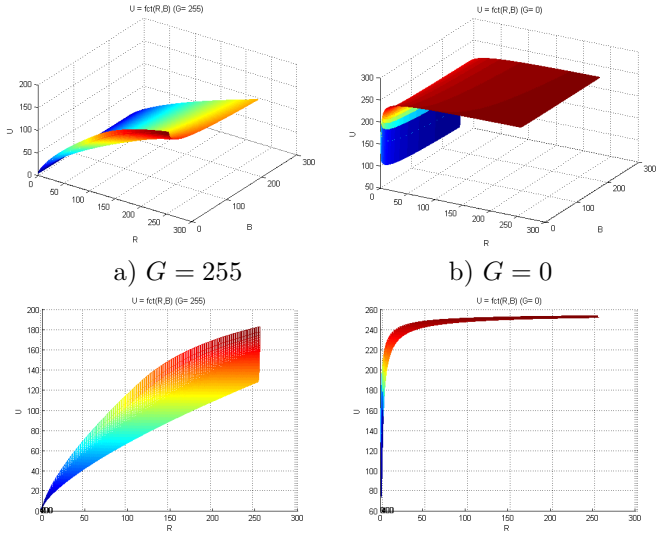
• *Simplified formulae :* Since the computation of the maxima is expensive, we may assume that each value r_0, g_0 and b_0 is close to a maximal intensity I_0. This hypothesis is valid when the camera is calibrated for full range on the white values. A second approximation is that I_0 is close to the dynamic range M. This assumption corresponds to an automatic contrast correction. Moreover we impose maximal values: $l_0 = u_0 = x_0 = M$ in order to keep the same maximum dynamic range. Incorporating those hypotheses inside the previous equations yields the expression of LUX as given in [7], valid for matrix T_2 of Eq. 2:

$$\begin{aligned} L &= (R+1)^{0.3}(G+1)^{0.6}(B+1)^{0.1} - 1 \\ U &= \begin{cases} \frac{M}{2}\left(\frac{R+1}{L+1}\right) & \text{if } R < L, \\ M - \frac{M}{2}\left(\frac{L+1}{R+1}\right) & \text{otherwise.} \end{cases} \\ X &= \begin{cases} \frac{M}{2}\left(\frac{B+1}{L+1}\right) & \text{if } B < L, \\ M - \frac{M}{2}\left(\frac{L+1}{B+1}\right) & \text{otherwise.} \end{cases} \end{aligned} \tag{10}$$

This is the formulae we actually use in our experiments. Note that there is another reason for using these formulae: indeed, as we want to compare two color images, the normalization w.r.t. the maxima in *each* image is not appropriate since the dynamic ranges would not be comparable anymore. Hence the color distance computed in the max-normalized LUX space would not be meaningful.

B. Behavior of LUX Nonlinearity

Fig. 4 shows the variation of nonlinearity for two opposite contexts: we plot the redish chrominance U as a function of R and B for two limiting cases : a) $G = 255$ simulating a case of bright image, and b) $G = 0$ simulating a case of dark image. This shows the adaptation of the curves to the color context, in accord with the Human Visual System behavior. We may compare these curves with Fig. 5 given in [12] where a generic model with nonlinear and adaptive processing is presented that explains the variablity in color discrimination data from various observers [13]. In this fig-



a) $G = 255$ b) $G = 0$

c) side views of a) and b)

Fig. 4. Logarithmic nonlinearity of U as a function of (R, B): a) for $G = 255$; b) for $G = 0$; c) side views of a) and b) resp.

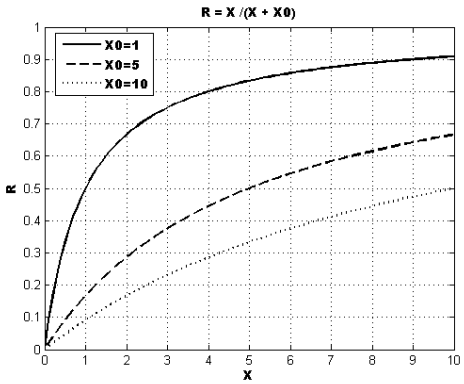


Fig. 5. Photoreceptor nonlinear transduction function depending on the adaptation state X_0 (cf. [12])

ure, X is the photoreceptor excitation level and X_0 is the background level to which the photoreceptor is adapted. $R = X/(X + X_0)$ is the photoreceptor response modeled in [12] by a Naka-Rushton law. We see that the LUX transform exhibits also this nice adaptation behavior.

IV. Experimental Results

A. Prototype Implementation and Computation Cost

A software prototype was implemented with GTK and OpenCV. Two versions are available: Linux version is based on MPlayer and wxWindows, whereas Windows version is based on DirectX, MFC and VisualStudio. The prototype gives the choice between various color spaces for computing the distance: RGB, YCrCb, XYZ, HSV, HLS, CIE Lab, CIE Luv and LUX. For a target image cut into 4,800 subimages (60×80 tiles), the video can be read at 40 frames per second on a Pentium Centrino 1.7GHz, with subimages of size 20×15. It takes about 8ms to compare a video frame with a sub-image: the processing includes proportional resizing, color transform, comparison (color distance computation), and swap (stack management). So it is fast compared to other work [2]: indeed it takes less than one minute to create an image mosaic from a video.

B. Quantitative Comparison of Color Spaces

The Peak SNR between input image A and its output mosaic B is computed as the mean value of the PSNR for each of three color planes in the RGB space:

$$PSNR(A,B) = \frac{1}{3}\sum_{i=1}^{3} \frac{255^2}{\frac{1}{N}\sum_{n=1}^{N}[A_i(n) - B_i(n)]^2} \qquad (11)$$

LUX yields always the best (highest) PSNR compared to all other color spaces that we have tested (Fig. 6).

- The HSV and HLS color spaces are the least efficient (colors do not match well). For example (Fig. 6a,b), with Lena as input image and a film (Simpson cartoon) whose color content is far from this target image, HLS and HSV render a bluish face whereas other color spaces give a rendering closer to the input image.
- The XYZ color space yields a very pale rendering.
- CIE Lab and CIE Luv color spaces give better results: good agreement in color, but little contrast.
- The RGB color space works quite well. But its performance is variable: it may give sometimes a PSNR better than YCrCb. But colors tend to smudge and the mosaic has less contrast than with YCrCb or LUX.
- The best visual results were systematically obtained with YCrCb and LUX that perform well in all our tests. LUX gives better contrast and renders better the scene lighting. Fig. 7c,d shows mosaics obtained for Lena with the Matrix movie. The PSNR with LUX is 3dB higher than with YCrCb. Finally, Fig. 7f shows the particular case when the input image chosen is one of the frames from the video itself. It that case, the PSNR is of course higher (here 31dB) since the color matching is better than when the input image and the video have nothing in common.

V. Conclusion

From our experience and observers visual evaluation, the best color spaces for mosaic creation are respectively LUX space in first position, and YCrCb in second position. Choosing a proper color space like LUX (instead of YIQ) is interesting to avoid the color correction post-processing step often used by other authors [2], [6]. We are currently investigating further the proper max-normalization that should be used with LUX for applications like image mosaics, involving color comparison between two images.

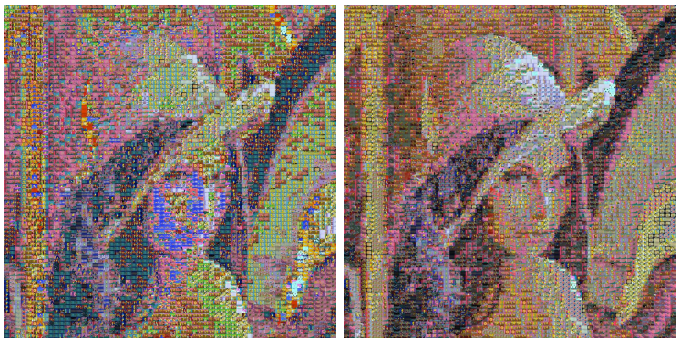
References

[1] R. Silvers and M. Hawley, *Photomosaics*. New York: Henry Holt & Company Inc., 1997.

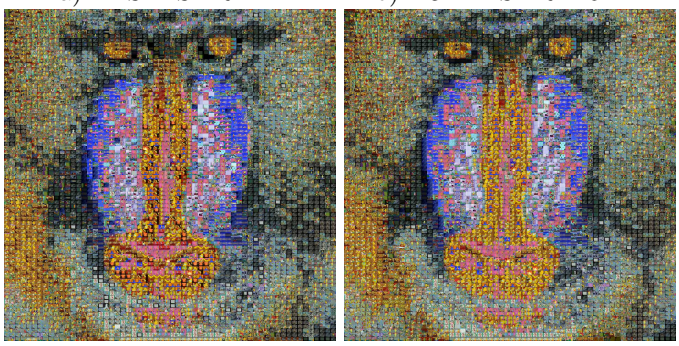
[2] A. Finkelstein and M. Range, "Image mosaics," in *Proc. RIDT*, 1998, pp. 11–22.

[3] W. Yunli, Z. Maojun, and G. Hui, "Selecting contributive frames for image fusion in video mosaic," in *Int.Conf. Information Engineering and Computer Science (ICIECS 2009)*, 2009, pp. 1–4.

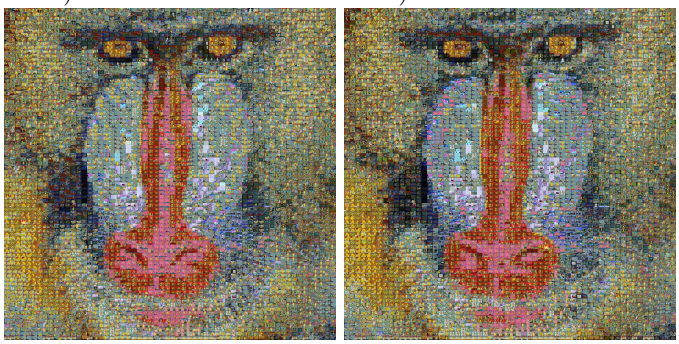
[4] A. Hausner, "Simulating decorative mosaics," in *Proc. SIGGRAPH2001*, Los Angeles, CA, 2001, pp. 573–580.



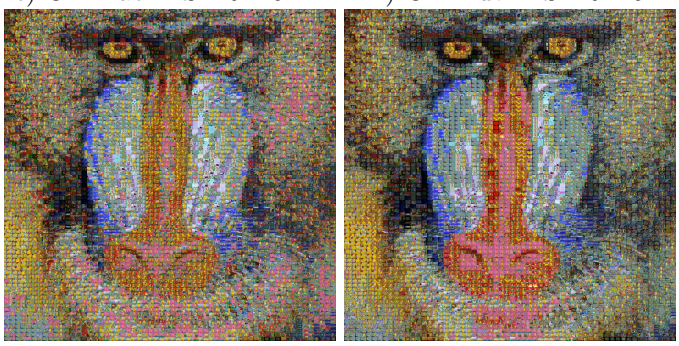
a) HLS: PSNR=22.4 b) LUX: PSNR=26.4



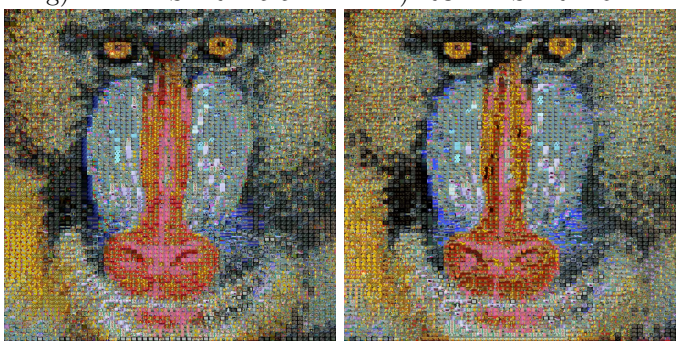
c) HLS: PSNR=24.3 d) HSV: PSNR=24.7



e) CIE Lab: PSNR=25.2 f) CIE Luv: PSNR=25.4



g) XYZ: PSNR=25.5 h) RGB: PSNR=26.2



i) YCrCb: PSNR=25.6 j) LUX: PSNR=26.9

Fig. 6. Results obtained with various color spaces for mosaics made of video frames from the Simpson Family cartoon: a) and b) Lena mosaics ; c) to j) Mandril mosaics.

a) Mandril b) Lena

c) YCrCb: PSNR=22.2 d) LUX: PSNR=25.3

e) Shrek f) LUX: PSNR=31.2

Fig. 7. a) b) and e) Input images ; c) and d) Lena mosaics with frames from the film Matrix ; f) Mosaic made with video frames from the Shrek animation movie.

[5] J. Kim and F. Pellacini, "Jigsaw image mosaics," in *Proc. SIGGRAPH2002*, 2002, pp. 657–664.

[6] A. Klein, T. Grant, A. Finkelstein, and M. Cohen, "Video mosaics," in *Proc. 2nd Int. Symposium on Nonphotorealistic animation and rendering (NPAR'02)*. Annecy, France: ACM Press, 2002.

[7] M. Liévin and F. Luthon, "Nonlinear color space and spatiotemporal MRF for hierarchical segmentation of face features in video," *IEEE Trans. on Image Processing*, vol. 13, no. 1, pp. 63–71, Jan. 2004.

[8] F. Luthon and B. Beaumesnil, "Color and R.O.I. with JPEG2000 for wireless videosurveillance," in *IEEE Int. Conf. on Image Processing (ICIP'04)*, Singapore, October 24-27 2004.

[9] S. Shah and M. Levine, "Visual information processing in primate cone pathways-Part I: A model," *IEEE Trans. on Systems, Man and Cybernetics-B*, vol. 26, no. 2, pp. 259–274, Apr. 1996.

[10] M. Jourlin and J. Pinoli, "Image dynamic range enhancement and stabilization in the context of the logarithmic image processing model," *Signal Processing*, vol. 41, pp. 225–237, 1995.

[11] G. Deng and J.-C. Pinoli, "Differentiation-based edge detection using the logarithmic image processing model," *Journal of Mathematical Imaging and Vision*, vol. 8, pp. 161–180, 1998.

[12] D. Alleysson and J. Hérault, "Variability in color discrimination data explained by a generic model with nonlinear and adaptive processing," *Color Research and Application*, vol. 26, 2001, supplement.

[13] D. MacAdam, "Visual sensitivities to color differences in daylight," *J. Opt. Soc. Am.*, vol. 32, pp. 247–273, 1942.