The 11th International Symposium on Wireless Personal Multimedia Communications (WPMC 2008)

EXTENDED YUV COLOR TRANSFORM FOR LOSSLESS IMAGE COMPRESSION

Victor Minchenkov
State University of
Airspace Instrumentation
St. Petersburg, Russia
victor@vu.spb.ru

Anton Sergeev
State University of
Airspace Instrumentation
St. Petersburg, Russia
slaros@vu.spb.ru

Andrey Turlikov
State University of
Airspace Instrumentation
St. Petersburg, Russia
turlikov@vu.spb.ru

ABSTRACT

Colorspace transforms (e.g. well-known YCrCb (YUV) in H.264, ICT in JPEG2000 etc.) are widely used in image compression and processing. The main goal of colorspace transform in image compression algorithms is to decrease dependence between image components that improves compression efficiency of the whole image coding system. But experiments show that standard colorpace transforms quite weakly decompose and decorrelate computer graphic images with color text, icons etc.

The proposed in this paper Extended Colorspace Transform solves the described problem. Applied to lossless image compression algorithms (JPEG-LS, H.264/AVC in lossless mode) it allows to significantly (up to 52%) increase compression ratio for synthetic images and computer graphics without any losses for photorealistic images.

I. INTRODUCTION

Color space is built in the way that any color is represented by point with definite coordinates. One color corresponds to one set of coordinates and one point of color space. Number of coordinates determines the color space dimension.

One of the most widespread color spaces is RGB. It is a three-dimensional color space, in which any color is defined by three coordinates. Each coordinate corresponds to one component in color decomposition on red (R), green(G) and blue(B).

In this article widely-used in the state-of-art algorithms of image compression (JPEG, MPEG2, H264/AVC) color transforms, namely RGB↔YUV, RGB↔YCbCr, are analysed.

To explain the necessity of using these color transforms first we should explain basics about color spaces by the example of RGB and underline several important problems.

Assume *rgb* is some specific image from number of RGB images. Let us define the probability distribution $\{\mathrm{p}(rgb)\}, rgb \in RGB$ at all discrete sets of RGB color space. Number of bits, required for storage of specific image *rgb* can be calculated as a magnitude of self-information $l = \mathrm{I}(rgb) = -\log \mathrm{p}(rgb)$, where $\mathrm{p}(rgb)$ is probability of image appearance. Therefore, to get the number of bits required for coding of statistically independent image series we need to calculate the following formula:

$$L = \sum_{i=1}^{N} \mathrm{I}(rgb^{(i)}) \underset{N\to\infty}{\approx} N \cdot \mathrm{E}[\mathrm{I}(rgb)] = = N \sum_{rgb \in RGB} (\mathrm{p}(rgb) \cdot \mathrm{I}(rgb)) = N \cdot \mathrm{H}(RGB) \quad (1)$$

To calculate the minimum number of bits that is needed for coding of image series (L) it is necessary to know the entropy of all RGB multitude. Since the probability distribution $\{\mathrm{p}(rgb)\}, rgb \in RGB$ is not available, exact value of L can not be calculated. Anyway, any real system of lossless image coding can not use less than $N \cdot \mathrm{H}(RGB)$ bits for coding of image series [1].

General scheme of static image codec is presented on the Fig.1. Source image enters forward color transform block. It is intended for creating new color space components which are less dependent. The ideal color transform has two properties:

1. $\mathrm{H}(RGB) = \mathrm{H}(YUV)$.
2. The following equality holds for new color components: $\mathrm{p}(yuv) = \mathrm{p}(y)\mathrm{p}(u)\mathrm{p}(v)$.

In turn, from 1 and 2 follows $\mathrm{H}(YUV) = \mathrm{H}(Y) + \mathrm{H}(U) + \mathrm{H}(V)$, meaning that color components can be coded independently.

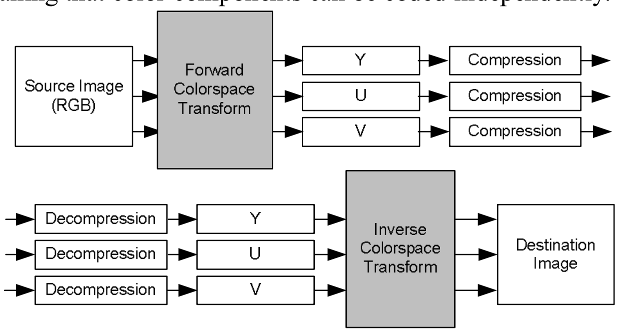

Figure 1: Structure of static image codec.

The inequality $\mathrm{H}(YUV) \le \mathrm{H}(Y) + \mathrm{H}(U) + \mathrm{H}(V)$ is satisfied for majority of color transforms, as the dependence among color components remains. It is impossible to verify given inequality because neither distributions RGB nor YUV are known. The presence of dependence among components can be observed only on the qualitative level. The paper [2] suggested using correlation criterion instead of dependence characteristics among images. In conformity with this criterion color transform decreasing the components cross-correlation is suggested. We can give the examples of images for which transform from paper [2] really decreases cross-correlation coefficient among components (as compared with YCbCr), but in phase of compressing this transform gives worse results then usage of YCbCr, so correlation criterion does not always work.

II. YUV Colorspace Tranform

The problem of RGB color space is that components are very dependent. The same information is represented in all three components, meaning that the same image details could be observed in every component. As a result, compression efficiency decreases and another non-RGB color space is needed. That is why linear transformations like YUV (YCbCr) [3] are used in the majority of codecs for reducing of components dependence. YCbCr is the digital version of well-known YUV transform and very often these two notations are used as synonyms. So in this paper YCbCr is selected for all explanations and calculation.

Forward transform from RGB in YCbCr:

$$\begin{cases} Y = 0.299R + 0.587G + 0.114B \\ Cb = -0.168736R - 0.331264G + 0.5B + 128 \\ Cr = 0.5R - 0.418688G - 0.081312B + 128 \end{cases} \quad (2)$$

Inverse transform from YCbCr in RGB:

$$\begin{cases} R = Y + 1.402(Cr - 128) \\ G = Y - 0.7141(Cr - 128) - 0.34414(Cb - 128) \\ B = Y + 1.772(Cb - 128) \end{cases} \quad (3)$$

One could note that all values of components Y, Cr and Cb are aligned in range [0,255]. YCbCr transform on average shows good results in decomposition of photorealistic images. And therefore YCbCr image representation could be compressed more efficiently than RGB one.

But experiments show that for computer graphics and synthetic images YCbCr demonstrates much worse results. Due to specifics of that class of images (many edges, color transitions etc.) YCbCr decomposition works weakly and an information redundancy may be found in all the components after transform (see Fig. 2 a, b and c). That leads to degrading of compression ratio of computer graphics (with color text, icons etc.) in comparison to the photorealistic images.

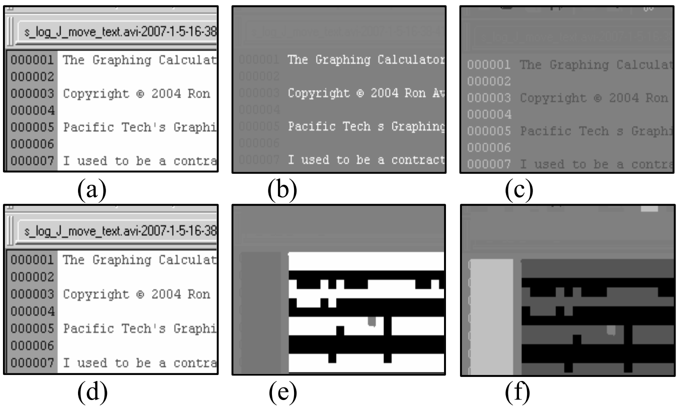

Figure 2: Y (a), Cr (b) and Cb (c) components of color text image after standard YCbCr transform;
Y (d), Cr (e) and Cb (f) components of color text image after Extended YCbCr transform.

In the Fig. 2 (a, b, c) Y, Cr and Cb components of color text image are represented. One could see that the same text exists in all the 3 components due to high dependence between them.

III. Extended Colorspace Tranform

A. Main Idea

In this section the Extension for YUV transform is proposed to improve compression of computer graphics.

From the equations (2) and (3) one could note that value Y=255 holds only for RGB = 0,0,0 (black color). This is the only point in RGB colorspace for which Y takes on the value of 255. So for inverse transform YCbCr → RGB of black color Cb and Cr components are optional. And black color in RGB representation could be reconstructed from YCbCr using Y component only. Forward transform:

$$\text{IF } RGB = (0,0,0) \text{ THEN } YCbCr = (0,0,0) \quad (4)$$

Inverse transform:

$$\text{IF } Y = 0 \text{ THEN } RGB = (0,0,0) \quad (5)$$

Therefore for black pixels any data may be written to Cb and Cr components at coder side because Y=0 is the unique condition for successful YCbCr → RGB reconstruction in this case.

$$\begin{aligned} &\text{IF } RGB = (0,0,0) \\ &\text{THEN } Y = 0 \text{ AND } [\text{data1}] \to Cb, [\text{data2}] \to Cr \end{aligned} \quad (5)$$

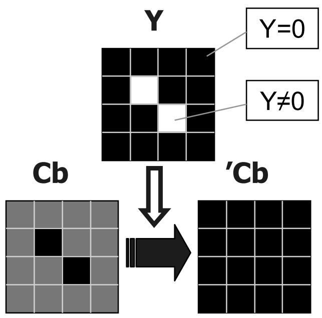

Figure 3: Example of redefining Cb component.

Assuming that background is black the redefinition mechanism from equation (5) could be applied. Assume alsothat image consists of equal domains. Obviously from the position of compression CrCb components of the second frequent color in the current domain should be used for redefinition of the corresponding components of background pixels (see Fig. 3) to improve coding rate. Forward transform:

$$\begin{aligned} &\text{IF } R_b G_b B_b = (0,0,0) \\ &\text{THEN } Y_b = 0 \text{ AND } Cb_s \to Cb_b, Cr_s \to Cr_b \end{aligned} \quad (6)$$

Inverse transform:

$$\text{IF } Y_b \text{ THEN } R_b G_b B_b = (0,0,0) \quad (7)$$

where $R_b G_b B_b$ and $Y_b Cb_b Cr_b$ are correspondently RGB and YCrCb components of the background pixels in the current domain; $Y_s Cb_s Cr_s$ stands for YCrCb components of the second most frequent color in the domain that is used for forward redefinition.

The CrCb components of the domain after (6) become much smoother that allows significantly improve compression performance of the successive blocks of an image coding system. The described algorithm could be used for domains with color background also. In that case let us define background as the most frequent color in the current domain. The algorithm is presented below.

B. Forward E-YUV Tranform

In the forward transform at the coder side for every image domain the following steps are processed:

Step 0: REPEAT for every pixel *j* in the domain

$$Y_j = 0.299R_j + 0.587G_j + 0.114B_j$$

$$Cb_j = -0.168736R_j - 0.331264G_j + 0.5B_j + 128$$

$$Cr_j = 0.5R_j - 0.418688G_j - 0.081312B_j + 128$$

Step 1:
SELECT MainColorY, SecondColorCr, SecondColorCb

Step 2:
IF *MainColorY* is NOT UNIQUE in the domain
THEN
MainColorY= MainColorCr= MainColorCb=0
GOTO Step 5

Step 3:

$$\text{IF}\left(\max_i(C_i) \le (\sum_{i=1}^{n} C_i - \frac{\sum_{i=1}^{n} C_i}{n})\right)$$

THEN
MainColorY= MainColorCr= MainColorCb=0
GOTO Step 5

Step 4: REPEAT for every pixel *j* in the domain
IF $Y_j = MainColorY$
THEN
$Cb_j = SecondColorCb$
$Cr_j = SecondColorCr$

Step 5:
GET next domain AND GOTO Step 1

C. Inverse E-YUV Tranform

Step 0:
Receive MainColorY, MainColorCr, MainColorCb

Step 1:
IF MainColorY= MainColorCr= MainColorCb=0
THEN GOTO Step 3
ELSE GOTO Step 2

Step 2: REPEAT for every pixel *j* in the domain
IF $Y_j = MainColorY$
THEN
$Cb_j = MainColorCb$
$Cr_j = MainColorCr$

Step 3: REPEAT for every pixel *j* in the domain

$$R_j = Y_j + 1.402(Cr_j - 128)$$

$$G_j = Y_j - 0.7141(Cr_j - 128) - 0.34414(Cb_j - 128)$$

$$B_j = Y_j + 1.772(Cb_j - 128)$$

D. Description Details

For correct inverse transform the RGB values of the background should be transmitted to the decoder side. But the overhead is very small: for 8x8 domain (192 bytes for 8 bit/pixel color depth) the amount of additional information is 3 bytes only (1.6%). Backgrounds of the neighbouring domains usually have close values and therefore could be efficiently compressed losslessly. In our experiments the additional data was compressed by 3-4 times on average using run length encoding with monotonous code of Gallager-van Voorhis [4].

It should be noticed that the suggested algorithms with redefinition of background CrCb components is efficient only for domains with several prevalent colors: texts, computer graphics etc. Therefore multicoloured domains of photorealistic images with textures and gradients should be compressed using standard YCrCb. To detect domains with computer graphics for which the proposed Extended Colorspace transform should be applied the following. empirical rule is used here (see Step 3 of the Forward algorithm):

$$\text{IF}\,(\max_i(C_i) > (\sum_{i=1}^{n} C_i - \frac{\sum_{i=1}^{n} C_i}{n}))\ \text{THEN redefine} \quad (8)$$

where n – number of different color sets in the domain, C_i – number of pixels with color i in the domain

One could see in (2) that different colorsets $Y_1Cr_1Cb_1$ and $Y_2Cr_2Cb_2$ may have equal Y components ($Y_1 = Y_2$) but differ in other two. Fortunately the probability that two different colors with the same Y meet in one domain is very small (<0.0006). So such domains are detected and simply skipped at the coder side and 3 zeros are put into the stream with background values. Receiving (000) at the decoder side means that E-YUV is not applied to the current domain. So step X is used to ensure that MainColorY is unique in the domain because Y component is used to detect redefined pixels at the decoder side (see Step 1 of the inverse transform)

In Fig. 2 (d, e, f) one could see an example of Extended YCrCb transform applied to computer image with color text and icons. Y component was not changed but components Cr and Cb were redefined in each domain separately and become much smother in comparison to the original ones in the Fig. 2 (a, b, c).

IV. Practical Results

To estimate the performance of the proposed approach the image components YCrCb after Extended YCrCb transform were compressed in lossless mode using the H264/AVC [5] (Table 1) and JPEG-LS [6] (Table 2) codecs. The summary of results is shown in Fig. 4. Typical compression pipeline is shown in Fig. 5.

Test images were divided into several groups: photorealistic ones, screen shots with small amount of text and icons, computer graphic images.

For photos like Lena and Peppers E-YUV transform have shown no gain because the text domains were not found by the detector (see Step X in Algorithm Description).

Additional data is compressed very efficiently because consists of all zero values.

At the same time E-YCrCb allows increasing compression ratio for computer graphic images (Calendar, VisualStudioIDE) up to 2 times!

V. Acknowledgements

This research work is supported by special grant of Intel CTG Research Council. The article is prepared within the scope of Finnish-Russian University Cooperation Program in Telecommunications (FRUCT) [7] and the authors would like to thank all experts and organizers of the FRUCT program for their help and contribution. The authors would also like to thank Nokia university collaboration program for providing publication and travelling grants.

References

[1] Krasilnikov N. "Digital image processing", Moscow, 2001. 319 p.

[2] Dae-Sung Cho, D. Birinov, Hyun Mun Kim, Shi-Hwa Lee and Yang-Seock Seo, "RGB Video Coding Using Residual Color Transform", Samsung Journal of Innovative Technology, 2005.

[3] Ford A., Roberts A. "Colour Space Conversions". August 11, 1998.

[4] Gallager R., van Voorhis D. "Optimal source codes for geometrically distributed alphabets", IEEE Trans. Inform. Theory 21 (2). 1975. P. 228-230.

[5] Wiegand T., Sullivan G. J., Bjøntegaard G, and Luthra A., "Overview of the H.264/AVC Video Coding Standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, July, 2003. P. 560 – 576.

[6] FCD 14495, Lossless and near-lossless coding of continuous tone still images (JPEG-LS).

[7] Finnish-Russian University Cooperation Program in Telecommunications (FRUCT), http://www.fruct.org

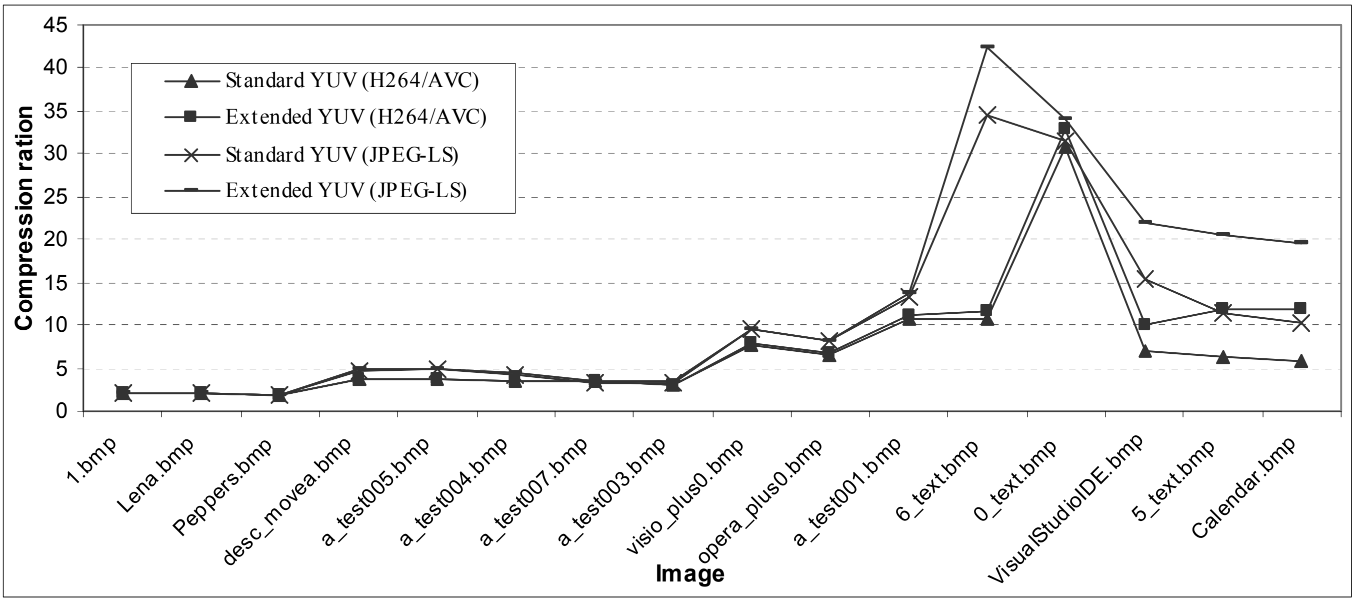

Figure 4: Image compression results using standard YUV and E-YUV transforms in H.264/AVC and JPEG-LS codecs.

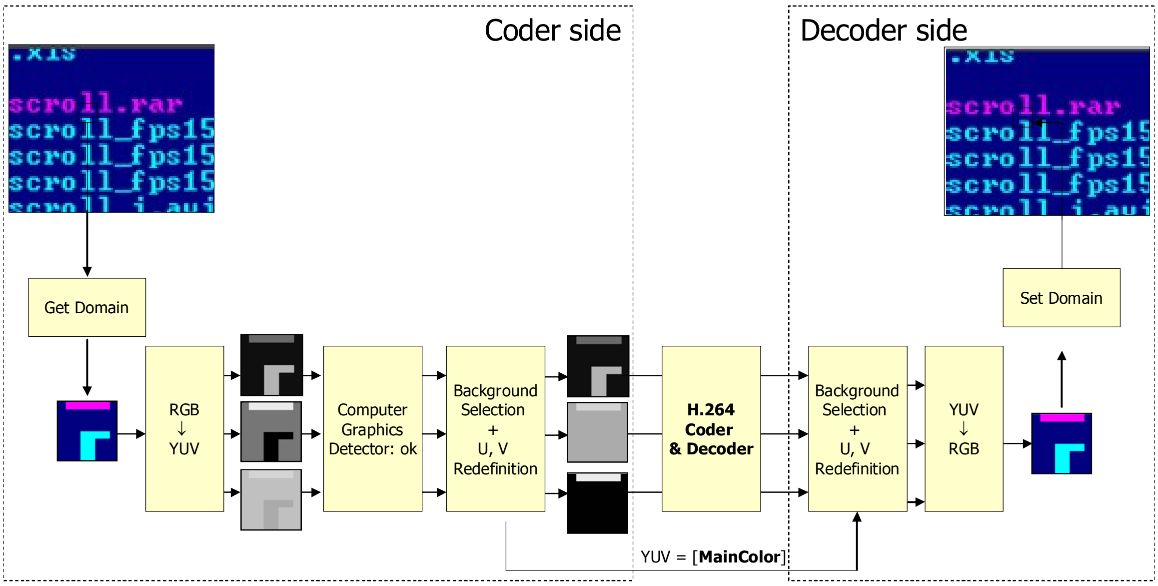

Figure 5: E-YUV transform for lossless image compression using H.264/AVC

Table 1: E-YUV: Compression results using H264/AVC codec in lossless mode.

Image	YUV: compressed size, bytes	E-YUV:compressed size, bytes	Gain, bytes	Gain, %	Type of image
1.bmp	3030602	3030602	0	0.00%	**Photos**
Lena.bmp	395712	395712	0	0.00%	
Peppers.bmp	434445	434445	0	0.00%	
desc_movea.bmp	622167	617344	4823	0.78%	**Screen Shots** + **Photo**. MSWindows desktop (photo) with small amount of text and icons
a_test005.bmp	1050993	1041751	9242	0.89%	
a_test007.bmp	1122772	1110026	12746	1.15%	
a_test003.bmp	1288251	1269895	18356	1.45%	
visio_plus0.bmp	310398	301517	8881	2.86%	
opera_plus0.bmp	366183	353466	12717	3.47%	
a_test001.bmp	365889	353031	12858	3.64%	
6_text.bmp	574360	538562	35798	6.23%	
0_text.bmp	201917	188771	13146	6.51%	
VisualStudioIDE.bmp	899765	619081	280684	31.22%	**Screen Shots**. A lot of icons & color text
5_text.bmp	986067	528283	457784	46.43%	
Calendar.bmp	1087062	518678	568384	52.31%	

Table 2: E-YUV: Compression results using JPEG-LS codec in lossless mode.

Image	YUV: compressed size, bytes	E-YUV:compressed size, bytes	Gain, bytes	Gain, %	Type of image
1.bmp	2909749	2909749	0	0.00%	**Photos**
lena.bmp	375456	375456	0	0.00%	
peppers.bmp	411528	411528	0	0.00%	
desc_movea.bmp	501858	493248	8610	1.72%	**Screen Shots** +**Photo**. MSWindows desktop (photo) with small amount of text and icons
a_test005.bmp	796403	787184	9219	1.16%	
a_test007.bmp	1183964	1103286	80678	6.81%	
a_test003.bmp	1176360	1149495	26865	2.28%	
visio_plus0.bmp	248261	244845	3416	1.38%	
opera_plus0.bmp	288959	287754	1205	0.42%	
a_test001.bmp	295837	287983	7854	2.65%	
6_text.bmp	180070	146962	33108	18.39%	
0_text.bmp	197270	183318	13952	7.60%	
VisualStudioIDE.bmp	405785	285006	120779	29.76%	**Screen Shots**. A lot of icons & color text
5_text,bmp	545527	302538	242989	44,54%	
Calendar.bmp	608070	318678	289392	47,62%	