REVERSIBLE INTEGER COLOR TRANSFORM WITH BIT-CONSTRAINT

Soo-Chang Pei, Jian-Jiun Ding
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C
Email address: pei@cc.ee.ntu.edu.tw

ABSTRACT

In color image processing, the RGB color coordinate is usually transformed into another one (e.g., YIQ or KLA) for system fitting or other purposes. Most of the color transforms are done by 3×3 matrices. However, these matrices are always not fixed-point. In this paper, we use a systematic algorithm to convert every 3×3 color transform into a reversible integer-to-integer transform. The resulted transform can be implemented with only fixed-point processor and no floating-point processor is required. Moreover, with the use of ladder-truncation technique, we can make least bit of the output the same as that of the input, and the long bit-length problem that always occurs for other integer transforms can be avoided. We derive the integer color transforms of RGB-to-KLA, IV_1V_2, YCrCb, DCT, and YIQ successfully.

1. INTRODUCTION

In color image processing, there are varieties of color systems. Most of them use three components to represent a color. Some popular ones are RGB (red, blue, and green), KLA (Karhunen-Loeve average, highest ability for color decorrelation), IHS (intensity, hue, saturation), IV_1V_2 (equivalent to IHS) [1][2], YCrCb, DCT, and YIQ. Since there are a variety of color systems, in image processing, we usually have to transform one color system into another one. The transformation is done by a 3×3 matrix. For example, the transformation of RGB-to-KLA is [2]:

$$\mathbf{A_{KLA}} = \begin{bmatrix} 0.54933 & 0.60238 & 0.57912 \\ 0.80429 & -0.19322 & -0.56194 \\ 0.22661 & -0.77447 & -0.59063 \end{bmatrix}. \quad (1)$$

Since the entries of color transforms are usually not binary fixed-point, we should use the floating-point processor to implement them. When using the fixed-point processor, we should approximate them by binary fixed-point matrices. Unfortunately, the approximated matrices are always irreversible. For example, suppose that we use the way of rounding to approximate $\mathbf{K_{KLA}}$ by a binary fixed-point matrix, i.e., $\mathbf{R_{KLA}} = \text{round}(\mathbf{A_{KLA}}2^K)\cdot 2^{-K}$ and $\mathbf{S_{KLA}} = \text{round}(\mathbf{A_{KLA}}^{-1}2^K)\cdot 2^{-K}$. From computer experiments, it can be shown that

$$\mathbf{S_{KLA}}\cdot\mathbf{R_{KLA}} \neq \mathbf{I} \quad \text{no matter how large } K \text{ is.} \quad (2)$$

Thus if we use the fixed-point processor to implement the color transform, the reversibility property is always lost. It affects the performance of many image processing applications. For example, we usually hide the watermark information in the least bit. If we can not recover the original image, even if only the least bit is wrong, the watermark information will be destroyed.

If we want to preserve the reversibility property, we should use floating-point processor, which is more time-consuming and inefficient. To overcome this problem, some integer color transforms that were used to approximate the non-integer transforms were developed [2][3][4].

In this paper, we introduce a general algorithm that can convert every color transforms into a reversible integer transform. We also use it to derive the integer RGB-to-KLA, IV_1V_2, YCrCb, DCT, and YIQ transforms successfully. They satisfy:

[**Goal 1**] The integer color transform should be reversible.
[**Goal 2**] No floating-point processor is required for both the forward and the inverse transforms.
[**Goal 3**] Accuracy: If **y** and **z** are the transform results of the original and the integer transforms, then $\mathbf{z} \approx \sigma\mathbf{y}$.
[**Goal 4**] The least bit should be constrained. For example, we may constrain that the least bit of the output must be the same as that of the input.

The advantages of the proposed algorithm are general and all the above four goals can be achieved. The existing integer transforms are hard to achieve Goals 3 and 4 at the same time. In this paper, with the ladder-truncation and the related error analysis techniques, we can make the least bit of the output the same as that of the input and at the same time the accuracy is acceptable.

2. THE GENERAL ALGORITHM FOR DERIVING THE INTEGER TRANSFORMS

First, we normalize the original 3×3 color transform $\mathbf{A_0}$ as **A** such that $\det(\mathbf{A}) = \pm 1$.

$$\mathbf{A} = \sigma\cdot\mathbf{A_0}. \quad \sigma = |\det(\mathbf{A_0})|^{-1/3}. \quad (3)$$

Note that, if $\mathbf{y} = \mathbf{A}\mathbf{x}$ and $\mathbf{y_1} = \mathbf{A_0}\mathbf{x}$, then $\mathbf{y} = \sigma\mathbf{y_1}$. Thus, if the difference of scaling is ignored, the performances of **A** and $\mathbf{A_0}$ are in essence the same.

Then we do row and column permutations for **A**:

$$\mathbf{C} = \mathbf{P_1}\mathbf{A}\mathbf{P_2} \quad (4)$$

0-7803-9134-9/05/$20.00 ©2005 IEEE

where $\mathbf{P_1}$ and $\mathbf{P_2}$ are permuting matrices, i.e., for each row and column, only one entry is 1 and others are 0. Then, applying the lifting scheme [5] and the triangular matrix scheme [6] with several modifications, we decompose $\mathbf{C}$ into three triangular matrices and one diagonal matrix:

$$\mathbf{C} = \mathbf{D}\mathbf{T_3}\mathbf{T_2}\mathbf{T_1} \quad \text{where} \tag{5}$$

$$\mathbf{D} = \begin{bmatrix} s & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad s = \text{sign}[\det(\mathbf{C})], \tag{6}$$

$$\mathbf{T_1} = \begin{bmatrix} 1 & t_1 & t_2 \\ 0 & 1 & t_3 \\ 0 & 0 & 1 \end{bmatrix}, \quad t_1 = (c_{22}-1)/c_{21}, \quad t_2 = -(t_1 z_2 + z_1),$$

$$t_3 = -z_2, \tag{7}$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}^{-1} \begin{bmatrix} -c_{23} \\ 2^{k_3} - c_{33} \end{bmatrix}, \tag{8}$$

c_{mn}'s (m, n = 1~3) are the entries of $\mathbf{C}$,

$$\mathbf{T_2} = \begin{bmatrix} 1 & 0 & 0 \\ t_4 & 1 & 0 \\ t_5 & t_6 & 1 \end{bmatrix}, \quad t_4 = c_{21}, \quad t_5 = c_{31},$$

$$t_6 = (c_{32} - t_1 c_{31}), \tag{9}$$

$$\mathbf{T_3} = \begin{bmatrix} 1 & t_7 & t_8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad t_7 = s(c_{12} - t_1 c_{11}) - t_6 t_8,$$

$$t_8 = s(c_{13} + z_1 c_{11} + z_2 c_{12}). \tag{10}$$

We then use fixed-point values g_n's to approximate t_n's:

$$g_n = sign(t_n) \sum_{r=-\infty}^{b} h_{n,r} 2^{-r}, \quad h_{n,r} = 0 \text{ or } 1, \tag{11}$$

$$g_n \approx t_n, \qquad n = 1 \sim 8, \qquad \text{Max}(r) = b.$$

Then we derive the reversible integer transform $\mathbf{B}$ that approximates $\mathbf{A}$ from:

$$\mathbf{B} = \mathbf{P_1^T} \mathbf{D} \mathbf{G_3} \mathbf{G_2} \mathbf{G_1} \mathbf{P_2^T} \tag{12}$$

where $\mathbf{G_1} = \begin{bmatrix} 1 & g_1 & g_2 \\ 0 & 1 & g_3 \\ 0 & 0 & 1 \end{bmatrix}$, $\mathbf{G_2} = \begin{bmatrix} 1 & 0 & 0 \\ g_4 & 1 & 0 \\ g_5 & g_6 & 1 \end{bmatrix}$, $\mathbf{G_3} = \begin{bmatrix} 1 & g_7 & g_8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. (13)

[Corollary 1] $\mathbf{B}$ is a binary-valued matrix that approximates $\mathbf{A}$. Moreover, if we define $\mathbf{B'} = \mathbf{P_2 G_1^{-1} G_2^{-1} G_2^{-1} D P_1}$, then it is no hard to prove that $\mathbf{B'B} = \mathbf{I}$. Moreover, since $\mathbf{G_1^{-1}}$, $\mathbf{G_2^{-1}}$, and $\mathbf{G_3^{-1}}$ are all binary-valued matrices, thus the inverse transform matrix $\mathbf{B^{-1}}$ is also a binary-valued matrix and the first two goals listed in Sec. 1 are satisfied.

Then we try to satisfy Goal 3 and Goal 4 in Sec. 1. Note that, if in (11) the least bit we use for approximating t_n's is 2^{-b}, and we use 2^{-h1} and 2^{-h2} to denote the least bit of $\mathbf{x}$ and $\mathbf{z}$ ($\mathbf{z} = \mathbf{Bx}$), then

$$h_2 = h_1 + 3b. \tag{14}$$

Thus the bit-length of $\mathbf{z}$ is obviously longer than that of $\mathbf{x}$. For example, if the least bit of g_n's is 1/32 and k_n's = 0, then $h_2 = h_1 + 15$. It may not be economical. To solve the problem, we can convert the triangular matrices $\mathbf{G_1}$, $\mathbf{G_2}$, and $\mathbf{G_3}$ in (13) into ladder-truncation operations.

From (12), the multiplication of $\mathbf{z} = \mathbf{Bx}$ can be divided into the following process

(1) $\mathbf{x_1} = \mathbf{P_2^T x}$, (2) $\mathbf{x_2} = \mathbf{G_1 x_1}$, (3) $\mathbf{x_3} = \mathbf{G_2 x_2}$,
(4) $\mathbf{x_4} = \mathbf{G_3 x_3}$, (5) $\mathbf{z} = \mathbf{P_1^T D x_4}$. (15)

If the ladder-truncation operations are applied, the 2nd, 3rd and 4th Steps will be modified, and **the process of the forward integer color transform** is:

(Step 1): $\mathbf{x_1} = \mathbf{P_2^T x}$. (16)

(Step 2): $x_2[1] = x_1[1] + Q_r\{g_1 x_1[2] + g_2 x_1[3]\}$,
$x_2[2] = x_1[2] + Q_r\{g_3 x_1[3]\}$, $x_2[3] = x_1[3]$, (17)

where Q_r is the truncation operation, which throws the bits that are less than 2^{-r}:

$$Q_r\left[\sum_{n=-\infty}^{\infty} d_n 2^{-n}\right] = \sum_{n=-\infty}^{r} d_n 2^{-n} \qquad (d_n\text{'s} = 0 \text{ or } 1). \tag{18}$$

(Step 3): $x_3[1] = x_2[1]$, $x_3[2] = x_2[2] + Q_r\{g_4 x_2[1]\}$,
$x_3[3] = x_2[3] + Q_r\{g_5 x_2[1] + g_6 x_2[2]\}$. (19)

(Step 4): $x_4[1] = x_3[1] + Q_r\{g_7 x_3[2] + g_8 x_3[3]\}$,
$x_4[2] = x_3[2]$, $x_4[3] = x_3[3]$. (20)

(Step 5): $\mathbf{z} = \mathbf{P_1^T D x_4}$. (21)

The process of the inverse integer color transform is:

(Step 1): $\mathbf{x_4} = \mathbf{D P_1 z}$. (22)

(Step 2): $x_3[3] = x_4[3]$, $x_3[2] = x_4[2]$,
$x_3[1] = x_4[1] - Q_r\{g_7 x_3[2] + g_8 x_3[3]\}$. (23)

(Step 3): $x_2[1] = x_3[1]$, $x_2[2] = x_3[2] - Q_r\{g_4 x_2[1]\}$,
$x_2[3] = x_3[3] - Q_r\{g_5 x_2[1] + g_6 x_2[2]\}$. (24)

(Step 4): $x_1[3] = x_2[3]$, $x_1[2] = x_2[2] - Q_r\{g_3 x_1[3]\}$,
$x_1[1] = x_2[1] - Q_r\{g_1 x_1[2] + g_2 x_1[3]\}$. (25)

(Step 5): $\mathbf{x} = \mathbf{P_2 x_1}$. (26)

[Corollary 2] If 2^{-h1} is the least bit of $\mathbf{x}$, $b > 0$ (b is the least bit used for approximating t_n's, see (11)), and r satisfies

$$h_1 \le r \le h_1 + b, \tag{27}$$

then the least bit of the output $\mathbf{z}$ is

$$2^{-r} \qquad \text{(independent of } b\text{)}. \tag{28}$$

Therefore, for our algorithms, the least bit of the output is determined by how many bits are preserved by the truncation operation Q. It is independent of b. That is, in (11), *no matter how many bits we use for approximating t_n's, the least bit of the output is remained to be r.*

[Corollary 3] The proposed integer color transform requires 8 multiplications and the original transform requires 9 multiplications. Thus the integer color transform does not increase the computing complexity.

Then we discuss the problem of accuracy. Note that, in Steps 2, 3, and 4, the truncation operation Q_r is equivalent to adding a small number:

$$Q_r\{a\} = a + \tau \qquad \text{where } -2^{-r-1} < \tau \le 2^{-r-1}. \tag{29}$$

If the input of Q_r is not known, τ can be treated as a random variable uniformly distributed in $(-2^{-r-1}, 2^{-r-1})$ and

$$E[\tau] = 0, \qquad E[\tau^2] = 4^{-r}/12, \tag{30}$$

where E means the expected value. Thus the process in (16)~(21) can be rewritten as:

$$\mathbf{z} = \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\{\mathbf{G}_3[\mathbf{G}_2(\mathbf{G}_1\mathbf{P}_2^{\mathrm{T}}\mathbf{x} + \boldsymbol{\Delta}_1) + \boldsymbol{\Delta}_2] + \boldsymbol{\Delta}_3\} \tag{31}$$

where $\boldsymbol{\Delta}_1$, $\boldsymbol{\Delta}_2$, and $\boldsymbol{\Delta}_3$ are 3×1 random vector:

$$\Delta_1[3] = \Delta_2[1] = \Delta_3[2] = \Delta_3[3] = 0, \tag{32}$$

and $\Delta_1[1]$, $\Delta_1[2]$, $\Delta_2[2]$, $\Delta_2[3]$, and $\Delta_3[1]$, are random variables whose statistical characters are the same in (30). We can compare (31) with the original non-integer color transform. If $\mathbf{y} = \mathbf{A}\mathbf{x}$, then

$$\mathbf{y} = \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\mathbf{T}_2\mathbf{T}_1\mathbf{P}_2^{\mathrm{T}}\mathbf{x}, \tag{33}$$

$$\begin{aligned}\mathbf{z} - \mathbf{y} &= \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{G}_3\mathbf{G}_2\boldsymbol{\Delta}_1 + \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{G}_3\boldsymbol{\Delta}_2 + \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\boldsymbol{\Delta}_3 \\ &\quad + \mathbf{P}_1^{\mathrm{T}}\mathbf{D}(\mathbf{G}_3\mathbf{G}_2\mathbf{G}_1 - \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1)\mathbf{P}_2^{\mathrm{T}}\mathbf{x}.\end{aligned} \tag{34}$$

Suppose that b is large enough such that

$$\nabla_1 = \mathbf{G}_1 - \mathbf{T}_1 \approx \mathbf{0}, \ \nabla_2 = \mathbf{G}_2 - \mathbf{T}_2 \approx \mathbf{0}, \ \nabla_3 = \mathbf{G}_3 - \mathbf{T}_3 \approx \mathbf{0}. \tag{35}$$

$$\begin{aligned}&\mathbf{G}_3\mathbf{G}_2\mathbf{G}_1 - \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1 \\ &= (\mathbf{T}_3 + \nabla_3)(\mathbf{T}_2 + \nabla_2)(\mathbf{T}_1 + \nabla_1) - \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1 \\ &\approx \mathbf{T}_3\mathbf{T}_2\nabla_1 + \mathbf{T}_3\nabla_2\mathbf{T}_1 + \nabla_3\mathbf{T}_2\mathbf{T}_1.\end{aligned} \tag{36}$$

Therefore,

[Corollary 4] The error of the integer color transform can be approximated by:

$$\begin{aligned}\mathbf{z} - \mathbf{y} \approx{}& \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\mathbf{T}_2\boldsymbol{\Delta}_1 + \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\boldsymbol{\Delta}_2 + \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\boldsymbol{\Delta}_3 \\ &+ \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\mathbf{T}_2\nabla_1\mathbf{P}_2^{\mathrm{T}}\mathbf{x} + \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\nabla_2\mathbf{T}_1\mathbf{P}_2^{\mathrm{T}}\mathbf{x} \\ &+ \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\nabla_3\mathbf{T}_2\mathbf{T}_1\mathbf{P}_2^{\mathrm{T}}\mathbf{x}.\end{aligned} \tag{37}$$

Thus the error comes from the six terms:

(1) the truncation in Step 2, which causes $\mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\mathbf{T}_2\boldsymbol{\Delta}_1$.
(2) the truncation in Step 3, which causes $\mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\boldsymbol{\Delta}_2$.
(3) the truncation in Step 4, which causes $\mathbf{P}_1^{\mathrm{T}}\mathbf{D}\boldsymbol{\Delta}_3$.
(4)quantization $\mathbf{T}_1$ into $\mathbf{G}_1$, which causes $\mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\mathbf{T}_2\nabla_1\mathbf{P}_2^{\mathrm{T}}\mathbf{x}$.
(5)quantization $\mathbf{T}_2$ into $\mathbf{G}_2$, which causes $\mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\nabla_2\mathbf{T}_1\mathbf{P}_2^{\mathrm{T}}\mathbf{x}$.
(6)quantization $\mathbf{T}_3$ into $\mathbf{G}_3$, which causes $\mathbf{P}_1^{\mathrm{T}}\mathbf{D}\nabla_3\mathbf{T}_2\mathbf{T}_1\mathbf{P}_2^{\mathrm{T}}\mathbf{x}$.

There are some things to be noticed.

(a) Since if (27) is satisfied the least bit of $\mathbf{z}$ is independent of b, thus in (11) we can choose a large value of b. If b is very large, ∇_1, ∇_2, and ∇_3 will be very small and the last three terms in (37) can be ignored, i.e.,

$$\mathbf{z} - \mathbf{y} \approx \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\mathbf{T}_2\boldsymbol{\Delta}_1 + \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\mathbf{T}_3\boldsymbol{\Delta}_2 + \mathbf{P}_1^{\mathrm{T}}\mathbf{D}\boldsymbol{\Delta}_3. \tag{38}$$

Also notice that in this case the error is independent of the input $\mathbf{x}$.

(b) From (37), it can be seen that the entries of $\mathbf{T}_1$, $\mathbf{T}_2$, and $\mathbf{T}_3$ affect the error of approximation. Especially, from (38), the entries of $\mathbf{T}_2$ and $\mathbf{T}_3$ have larger effects than those of $\mathbf{T}_1$. Thus, to make the error small, the values of t_4, t_5, t_6, t_7, and t_8 should be as small as possible.

(c) After the integer transform is designed, we can use (37) together with the following equation to estimate the normalized root mean square error (NRMSE):

$$NRMSE = \sqrt{\frac{E[(\mathbf{z}-\mathbf{y})^{\mathrm{T}}(\mathbf{z}-\mathbf{y})]}{E[\mathbf{y}^{\mathrm{T}}\mathbf{y}]}}. \tag{39}$$

where E means the expected value. In (4), we can vary the permuting matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ iteratively and calculate the NRMSE of the integer transform we obtain. Since there are 3! = 6 choices for each of $\mathbf{P}_1$ and $\mathbf{P}_2$, there are at least 36 possible integer color transforms we can obtain. We can choose the optimal one that can minimize the NRMSE.

In addition to $\mathbf{P}_1$ and $\mathbf{P}_2$, in (4), we can use $\mathbf{A}^{\mathrm{T}}$, $\mathbf{A}^{-1}$, or $(\mathbf{A}^{\mathrm{T}})^{-1}$ instead of $\mathbf{A}$ to search the optimal color transforms.

3. EXAMPLES

In this section, we show some integer color transforms we derived. Before deriving the integer color transform, we normalize the original color transform such that $\det(\mathbf{A}) = \pm 1$. Suppose that the least bit of the input data is 1 (i.e., in Corollary 2, $h_1 = 0$). We choose the values of b and r as

$$b = 10, \qquad r = 0. \tag{40}$$

It is easy to see that (27) is satisfied. Thus the least bit of the output for the following integer transforms is

$$2^{-0} = 1. \tag{41}$$

Therefore, *the least bits of both the input and the output data are 1*. We also suppose that the input signal is uniform distributed in [0, 255] such that

$$E(x[n]) = 127.5, \ E(x^2[n]) = 255^2/3, \ E^2(x[n]) = 127.5^2. \tag{42}$$

Use (42) together with (30), (37), (39), and the fact that $\mathbf{y}^{\mathrm{T}}\mathbf{y} = \mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x}$, we can calculate the NRMSE.

(1) RGB to KLA

$$\text{original: } \mathbf{A} = \begin{bmatrix} 0.8185 & 0.8975 & 0.8629 \\ 1.1984 & -0.2879 & -0.8373 \\ 0.3376 & -1.1539 & -0.8800 \end{bmatrix}. \tag{43}$$

From (4)~(11), the parameters of g_n's we obtain are:

$$\begin{aligned} &g_1 = -215/1024, \quad g_2 = 1313/1024, \quad g_3 = -107/512, \\ &g_4 = 221/256, \quad g_5 = -857/1024, \quad g_6 = 1047/1024, \\ &g_7 = -149/1024, \quad g_8 = -7/1024, \end{aligned} \tag{44}$$

$$\mathbf{D} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{P}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \ \mathbf{P}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \tag{45}$$

$$NRMSE = 0.187\%, \tag{46}$$

where $\mathbf{P}_1$ and $\mathbf{P}_2$ are found iteratively to minimize the NRMSE. Thus, from (16)~(21), the process of the integer RGB to KLA transform is:

(Step 1): $x_1[1] = x[3], \quad x_1[2] = x[1], \quad x_1[3] = x_1[2]$.

(Step 2) :$x_2[1] = x_1[1] + Q_0\{-215x_1[2]/1024 + 1313x_1[3]/1024\}$

$x_2[2] = x_1[2] + Q_0\{-107x_1[3]/512\}, \quad x_2[3] = x_1[3]$,

(Step 3): $x_3[1] = x_2[1]$, $x_3[2] = x_2[2]+Q_0\{221x_2[1]/256\}$,
$x_3[3]=x_2[3]+Q_0\{-857x_2[1]/1024 + 1047x_2[2]/1024\}$
(Step 4) $x_4[1]=x_3[1]+Q_0\{-149x_3[2]/1024 - 7x_3[3]/1024\}$,
$x_4[2] = x_3[2]$, $x_4[3] = x_3[3]$.
(Step 5): $z[1] = x_4[2]$, $z[2] = x_4[3]$, $z[3] = -x_4[1]$. (47)

(2) RGB to IV_1V_2

$$\text{original: } \mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ -1/\sqrt{6} & -1/\sqrt{6} & 2/\sqrt{6} \\ 1/\sqrt{6} & -1/\sqrt{6} & 0 \end{bmatrix}. \quad (48)$$

• parameters of the integer color transform in (16)~(26):
$g_1 = 115/256$, $g_2 = 47/256$, $g_3 = -341/1024$,
$g_4 = -209/512$, $g_5 = 1$, $g_6 = 141/256$,
$g_7 = 119/1024$, $g_8 = -557/1024$, (49)

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{P_1} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \ \mathbf{P_2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad (50)$$

NRMSE = 0.175%. (51)

(3) RGB to YCrCb

$$\text{original: } \mathbf{A} = \begin{bmatrix} 0.4836 & 0.9495 & 0.1844 \\ 0.8087 & -0.6777 & -0.1310 \\ -0.2734 & -0.5354 & 0.8087 \end{bmatrix}. \quad (52)$$

• parameters of the integer color transform in (16)~(26):
$g_1 = 289/1024$, $g_2 = 343/1024$, $g_3 = 99/1024$,
$g_4 = -347/512$, $g_5 = -137/256$, $g_6 = -125/1024$,
$g_7 = 201/1024$, $g_8 = -79/512$, (53)

$$\mathbf{D} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{P_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{P_2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (54)$$

NRMSE = 0.288%. (55)

(4) RGB to DCT

$$\text{original: } \mathbf{A} = \begin{bmatrix} 0.5774 & 0.5774 & 0.5774 \\ 0.7071 & 0 & -0.7071 \\ 0.4082 & -0.8165 & 0.4082 \end{bmatrix} \quad (56)$$

• parameters of the integer color transform in (16)~(26):
$g_1 = 53/128$, $g_2 = 325/1024$, $g_3 = 115/512$,
$g_4 = -181/256$, $g_5 = -209/512$, $g_6 = -245/1024$,
$g_7 = 53/128$, $g_8 = 325/1024$, (57)

$$\mathbf{D} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{P_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{P_2} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad (58)$$

NRMSE = 0.267%. (59)

(5) RGB to YIQ

$$\text{original: } \mathbf{A} = \begin{bmatrix} 0.4722 & 0.9270 & 0.1800 \\ 0.9412 & -0.4327 & -0.5085 \\ 0.3332 & -0.8259 & 0.4927 \end{bmatrix}. \quad (60)$$

• parameters of the integer color transform in (16)~(26):
$g_1 = 139/1024$, $g_2 = 227/512$, $g_3 = -81/256$,
$g_4 = -443/1024$, $g_5 = -423/512$, $g_6 = 57/128$,
$g_7 = 205/512$, $g_8 = -31/256$, (61)

$$\mathbf{D} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{P_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{P_2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (62)$$

NRMSE = 0.297%. (63)

Notice that the NRMSEs of all the integer color transform is no more than 0.3%. (In contrast, the NRMSEs of the integer RGB-to-KLA and YCrCb, transforms shown in [2][3] are more than 10%). We can achieve higher accuracy because the least bit 2^{-b} in (11) used for approximating t_n's can be chosen very small and at the same time the least bit of the output is not affected.

4. CONCLUSIONS

In this paper, we introduced a systematic way that approximates a 3×3 color transform matrix by a reversible integer transform. When doing fixed-point approximation, the goals of (1) binary matrix entry, (2) reversibility, (3) good approximation accuracy, and (4) constraint for the least bit are hard to satisfy at the same time. However, with the algorithm introduced in Sec. 2, we are easy to achieve all the above four goals. In Sec. 3 we successfully convert several well-known color transforms into the reversible integer transforms. With them, we can use the fixed-point processor instead of the floating-point one to do color transformation. It is helpful for improving the efficiency of digital image processing.

5. REFERENCES

[1] W. K. Pratt, *Digital Image Processing*, 2nd ed., Wiley, New York, 1991.

[2] B. Deknuydt, J. Smolders, L. V. Eychen, and A. Oosterlinck, "Color space choice for nearly reversible image compression", *SPIE* vol. 1818, *Visual Communications and Image Processing*, p. 1300-1311, 1992.

[3] M. J. Gormish, E. L. Schwartz, A. Keith, et al, "Lossless and nearly lossless compression for high quality images", *Proceedings of the SPIE*, vol. 3025, pp. 62-70, Feb. 1997.

[4] P. Hao and Q. Shi., "Comparative study of color transforms for image coding and derivation of integer reversible color transform", *Int. Conf. Pattern Recognition*, vol. 3, pp. 224-227, Sept. 2000.

[5] S. Oraintara, Y. J. Chen, and T. Q. Nguyen, ' Integer Fast Fourier Transform', *IEEE Trans. Signal Processing*, vol. 50, p. 607-618, 2002.

[6] P. Hao and Q. Shi., "Matrix Factorizations for Reversible Integer Mapping", *IEEE Trans. Signal Processing*, vol. 49, no. 10, pp. 2314-2324, Oct. 2001.

[7] M. D. Adams, F. Kossentini, and R. K. Ward, 'Generalized S transform', *IEEE Trans. Signal Processing*, vol. 50, no. 11, pp. 2831-2842, Nov. 2002.