https://doi.org/10.2352/ISSN.2169-2629.2017.32.260
©2016; Society for Imaging Science and Technology

# ICC Profile Color Table Compression

*Chuohao Tang*[1], *Weibao Wang*[1], *Sean Collison*[2], *Mark Shaw*[2], *Jay Gondek*[3], *Amy Reibman*[1], *and Jan Allebach*[1]

[1] **School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana, 47907**
[2] **HP Inc., Boise, Idaho, 83714**
[3] **HP Inc., Vancouver, Washington, 98683**

## Abstract

*ICC profiles are widely used to provide transformations between different color spaces in different devices. The color look-up tables (CLUTs) in the profiles will increase the file sizes when embedded in color documents. In this paper, we discuss a compression method that decreases the storage cost of the CLUTs. A compressed color table includes quantized DCT coefficients for the color table, the additional nodes with large color difference, and the coefficients bit assignment table. This method supports lossy table compression to minimize the network traffic and delay, and also achieves relatively small maximum color difference.*

## Introduction

Color Management plays an important role in color reproduction and transformation of color information between various devices. Device profiles provide color management systems with the information necessary to convert color data between native device color spaces and device-independent color spaces. The International Color Consortium (ICC) profile framework has been used as a standard to communicate and interchange between various color spaces.

An ICC output profile mainly consists of color lookup table (LUT) pairs, so-called A2B and B2A tables, where A and B denote the device-dependent and the device-independent color spaces, respectively. For different devices, there are different LUT rendering intent pairs. For example, for CMYK output devices, there are three LUT pairs, enumerated from 0 to 2, enabling the user to choose from one of the three possible rendering intents: perceptual, colorimetric, or saturation [1].

ICC profiles are often embedded in color documents to achieve color fidelity between different devices, which increases the total size of these documents. The size of color tables will also increase with finer sampling of the spaces and larger bit depths. Each graphical element or image in a document may have its own ICC profile. In some cases, including the source CMYK profile can exceed the size of the image or graphical element itself. Thus, a method to compress the LUTs in the ICC profile while achieving small color difference and large compression ratio is desirable for the purpose of conserving memory and storage, and also reducing network traffic and delay.

Compression methods exploit two separate characteristics of the data: irrelevance and redundancy. Irrelevance refers to information that does not need to be kept for the reconstruction. Lossy methods exploit irrelevance, and lossless methods do not. Redundancy refers to the repetitive or statistical dependencies in the data, and can be removed via both lossless and lossy methods. The objective of compression is to reduce the irrelevance and redundancy of the data in order to be able to store or transmit the data effectively.

A variety of methods have been proposed to compress images and videos. But few compression methods have been used to compress color tables. Reference [2] does propose a preprocessing method for lossless compression of color look-up tables. Their method is based on hierarchical differential encoding and cellular interpolative predictive coding methods. Here, we propose to use a lossy method, followed by a lossless method to compress the color tables that are part of the ICC profile. The lossy method will be tailored to the specific characteristics of the color tables to be compressed. The lossless method is generic, and could be used to compress a variety of different types of data streams.

Accordingly, we first review two major categories of lossy methods that could be applied to this problem. Wavelet compression methods have been used for lossy 2D image and 3D video compression. [3] summarized image compression methods using coding of wavelet coefficients. [4], [5] are widely used efficient wavelet compression methods which have proved very successful in still image coding. [6], [7] proposed three-dimensional wavelet compression methods to efficiently encode 3D volumetric image data and video coding. The Discrete Cosine Transform (DCT) has also been widely used in image compression [8] and video compression [9].

For our color table compression problem, we are inspired by the various video compression methods. But the color table data has very different characteristics than video data, and the quality considerations are also very different. The color tables can be both 3D and 4D. Thus, we need to explore the characteristics of the color tables accordingly.

In this paper, we propose a color table compression method based on the multi-dimensional DCT. The compressed data consists of three types of information: the quantized DCT coefficients for the color table, the additional nodes with large color difference or visual importance, and the Coefficients Bit Assignment Table (CBAT) that we propose. If we compress the B2A tables, we evaluate the compression results using the A2B tables of the ICC profile. That is, we first compress the B2A tables, for example, the $L^*a^*b^*$ to $CMYK$. We convert the reconstructed $CMYK$ values to $L^*a^*b^*$ using the A2B tables in this profile, and calculate the color difference in CIELAB space. Our color table compression method may be generalized to other settings in which color look-up tables need to be compressed.

The paper is organized as follows. We first introduce the framework for color table compression. Then we describe the decoding process. Finally we evaluate our results in terms of color difference.

## Framework for Color Table Compression

Figure 1 provides a high-level view of the color table compression-reconstruction process. The compression process consists of two stages: a lossy stage that exploits specific characteristics of the color table data to be compressed, followed by a lossless stage that is more generically designed to compress any kind of data. The reconstruction process consists of a decoding step that inverts the encoding step, except for the loss, which is not recovered.

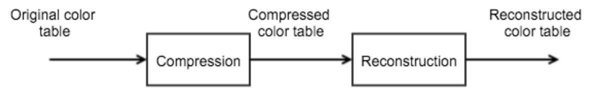


***Figure 1.*** *Compression-reconstruction process.*

As shown in Fig. 2, we employ a DCT compression method followed by a standard lossless compression technique, such as the Lempel-Ziv-Markov chain-Algorithm (LZMA) [10] or GZIP [11] to achieve high compression rates. As our ICC profile may contain a 3D or 4D color table, we will use a 3D or 4D DCT method, accordingly.

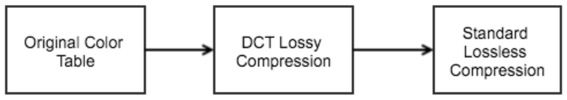


***Figure 2.*** *Compression framework.*

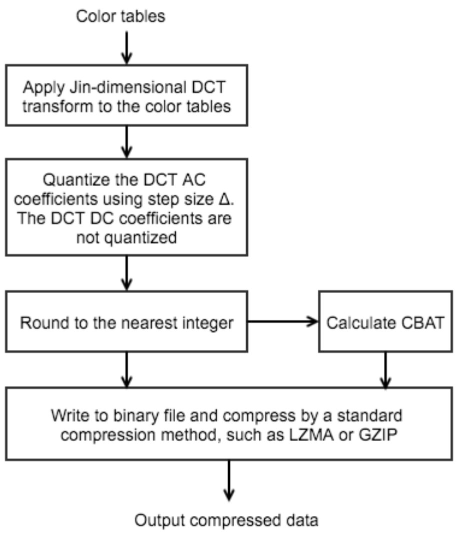

Color tables
Apply Jin-dimensional DCT transform to the color tables
Quantize the DCT AC coefficients using step size Δ. The DCT DC coefficients are not quantized
Round to the nearest integer
Calculate CBAT
Write to binary file and compress by a standard compression method, such as LZMA or GZIP
Output compressed data

***Figure 3.*** *Workflow of the compression method.*

Figure 3 contains a detailed block diagram of the compression method. Normally, color tables representing different rendering intents are included with one ICC profile. We assume there are $N$ color tables for an ICC profile: $CLUT_1, CLUT_2, ..., CLUT_N$. We assume there are $J_{in}$ channels in the input color space, and $J_{out}$ channels in the output color space. Normally, $J_{in}$ and $J_{out}$ can be 3 or 4. For each output channel, we assume that the LUT contains $M^{J_{in}}$ nodes, where $M$ is the number of nodes of each input channel.

Applying the $J_{in}$-dimensional DCT transform to the color tables $CLUT_i$, we obtain as many DCT coefficients as there are nodes in the original color table. In the next step, we quantize the AC coefficients using a fixed step size $\Delta$, and round to the nearest integer. We round the DC coefficients to the nearest integer, so they are effectively quantized to step size $\Delta = 1$.

$$ACcoef_q = \left\lfloor \frac{AC\ Coefficient}{\Delta} \right\rceil, \quad (1)$$

$$DCcoef_q = \lfloor DC\ Coefficient \rceil, \quad (2)$$

where $\lfloor \cdot \rceil$ denotes the rounding operation.

After obtaining the quantized DCT coefficients,we need to reorder the $J_{in}$-dimensional quantized coefficients to a 1D data stream in a certain order. As the energy after the DCT transform concentrates in the low frequency domain, we can use 3D zigzag ordering to reorder our data. Figure 4 shows the ordering process of 3D zigzag. The traversal is such that the planes $i+j+k=c$ are visited in increasing order of $c$ and a 2D zigzagging is performed within each plane [9]. Such traversal of the quantized coefficients from low-to-high frequency introduces a large amount of redundancy to our coefficient bit assignment table, which we will soon discuss, thus allowing efficient packing of the data.

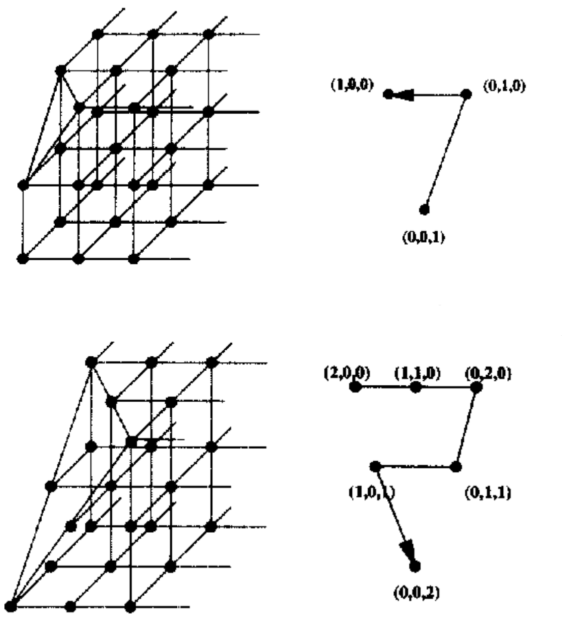
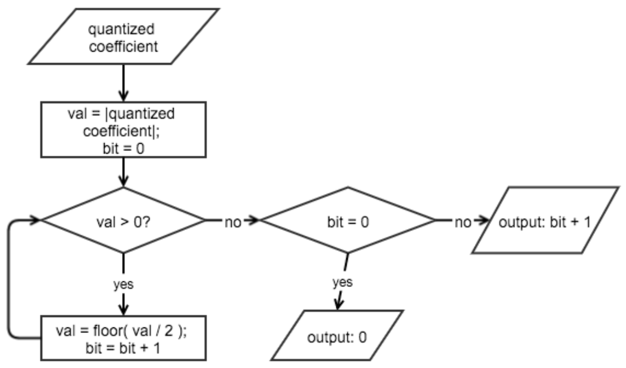


***Figure 4.*** *3D zigzag ordering: planes with $i+j+k=1$ and $i+j+k=2$ (after [9]).*

After obtaining the 1D quantized DCT coefficients, we write them to a binary file, followed by a standard lossless compression technique like LZMA. Then we use the quantized coefficients to calculate the coefficient bit assignment table (CBAT), which is used for the decoding process.

### Coefficient Bit Assignment Table

The Coefficient Bit Assignment Table stores the information of how many bits are assigned to each coefficient. To quantize a real number in the range $-0.5$ to $L-0.5$ to an integer value, we need $\lceil logL \rceil$ bits. As an alternative to implementing the logarithm function directly, Fig. 5 shows a method for determining $\lceil logL \rceil$ that can be implemented very efficiently in an embedded system. As the coefficient can be a negative number, we need one more bit for the sign.



quantized coefficient
val = |quantized coefficient|; bit = 0
val > 0?
no
yes
val = floor( val / 2 ); bit = bit + 1
bit = 0
no
yes
output: bit + 1
output: 0

*Figure 5. Algorithm to count bits needed for a given DCT coefficient.*

We use a different CBAT for each output channel. As there are $J_{out}$ output channels, we will have $J_{out}$ CBATs in one ICC profile. The nodes in each CBAT are the same as in the original color table. Now we will discuss how to calculate each CBAT. We will use the quantized DCT coefficients to calculate the CBAT. For a certain output channel, we denote the quantized DCT coefficient as $Q_{i,j}$, where $i = 1,2,...,N$ is the color table number, and $j = 1,2,...,M^{J_{in}}$ is the node number. The number of bits needed for $Q_{i,j}$ is

$$B_{i,j} = \begin{cases} \lceil log|Q_{i,j}| \rceil + 1 & \texttt{if } |Q_{i,j}| > 0 \\ 0 & \texttt{if } |Q_{i,j}| = 0 \end{cases} . \quad (3)$$

We use $L_j$ to represent the CBAT value in node location $j$, and it can be calculated by

$$L_j = \max_{1 \le i \le N} (B_{i,j}). \quad (4)$$

The remaining question about the CBAT is what is the total size needed to store it. For the CBAT, we assign a fixed number of bits for every node. Assume we assign $a$ bits for every node in the CBAT, then

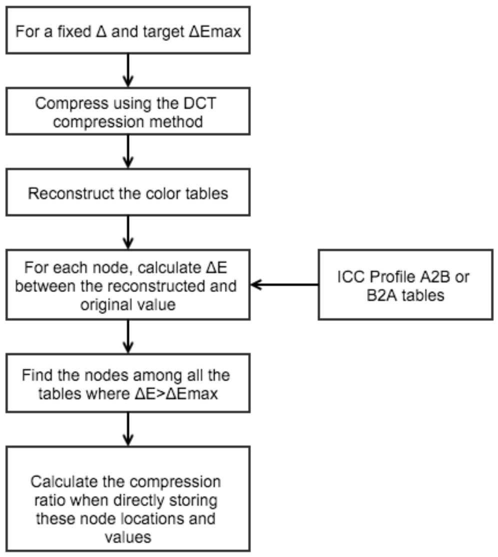
$$a = \max_{1 \le j \le M^{J_{in}}} (\lceil logL_j \rceil). \quad (5)$$

So the total size of one CBAT is $aM^{J_{in}}$ bits. Although it is a fixed number, after the 3D zigzag ordering, the neighboring nodes tend to have similar value. Thus the total CBAT size can be reduced significantly by the lossless compression stage. We discussed above how to calculate the the size of the CBAT for one output channel. We then use the same method to calculate the sizes for the rest of the CBATs. Combining them, we can obtain the total size for all $J_{out}$ CBATs.

### Determining Step Size $\Delta$

Another question is how to choose $\Delta$. If $\Delta$ is large, we can achieve a large compression ratio; but the color difference will be large at the same time. If $\Delta$ is small, the color difference is small; but the compression ratio will be limited. To achieve high compression ratio and small color difference at the same time, we store the nodes with large color difference separately.

For a fixed $\Delta$, we compress the color tables using the method discussed above. After inverting this compression process, we obtain the reconstructed color tables. Figure 6 shows how to calculate the compression ratio for a given $\Delta$ and target maximum color difference $\Delta E_{max}$ when storing additional nodes.



For a fixed Δ and target ΔEmax
Compress using the DCT compression method
Reconstruct the color tables
For each node, calculate ΔE between the reconstructed and original value
ICC Profile A2B or B2A tables
Find the nodes among all the tables where ΔE>ΔEmax
Calculate the compression ratio when directly storing these node locations and values

*Figure 6. Process to calculate compression ratio for a given $\Delta$ and $\Delta E_{max}$ when storing additional nodes.*

### *Reducing the Maximum Color Difference*

After we obtain the reconstructed color table, we can calculate the color difference between the values in the original and reconstructed color tables (details are discussed in the Evaluation Process section). Figure 7 shows clipped histograms of color differences for three different color tables. The color tables on which this data is based are described in the Results section. The histogram values are clipped to the range 0 to 10 so that we can see clearly that the number of nodes with large color difference is limited.

We then store the nodes that have a large color difference separately to improve the maximum color difference across all color tables. First, we set a target maximum color difference $\Delta E_{max}$. Second, we find the nodes among all the color tables where $\Delta E > \Delta E_{max}$, and store those node locations and node values directly.

If we have $M^{J_{in}}$ nodes in a table, then for each input channel we need $\lceil logM \rceil$ bits to store the location. So we need $\lceil logM \rceil J_{in}$ bits to store all the $J_{in}$ input channel locations. For each output channel, we need $b$ bytes (as indicated in the ICC profile *bytedepth* field) to store the value. So we need $bJ_{out}$ bytes to store all the $J_{out}$ output channel values. So in total, for each node that we wish to store directly, we need additional $\lceil logM \rceil J_{in}/8 + bJ_{out}$ bytes. This method also provides the option to store the nodes with significant visual importance directly without any loss.

### *Modified Compression Method*

Figure 8 shows the relation between the compression ratio and step size $\Delta$ when we store the additional nodes where $\Delta E > \Delta E_{max}$ separately. The color tables on which this data is based are described in the Results section.

We can see from the figure that for each fixed target $\Delta E_{max}$, as $\Delta$ increases, the compression ratio first increases, and then decreases after a peak compression ratio value is reached. That is because when $\Delta$ increases, the compression is more aggressive, and the error $\Delta E$ gets larger. Thus, the number of the reconstructed nodes which have $\Delta E > \Delta E_{max}$ will increase. So we will store more nodes directly, thereby reducing the efficiency of the lossy phase of our compression method. Therefore, the peak value is the optimal balance between the compression ratio and the color difference.

Our final output bit stream consists of three parts: CBATs, quantized DCT coefficients, and additional nodes. When $\Delta$ increases, the size of quantized DCT coefficients will decrease but the number of nodes with large $\Delta E$ will increase. So we want to find a balance between the two parts. For a fixed target $\Delta E_{max}$, we find the optimal $\Delta_{opt}$ when it achieves maximum compression ratio in the compression ratio delta curve. Figure 9 shows the modified workflow of the compression method that includes storing these additional nodes.

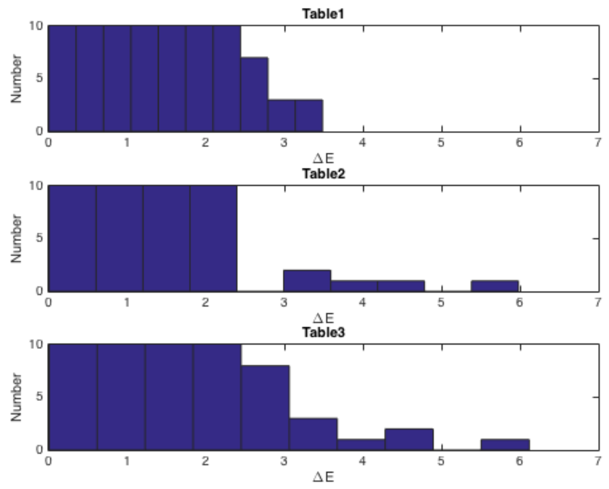


*Figure 7. Clipped histograms of color differences.*

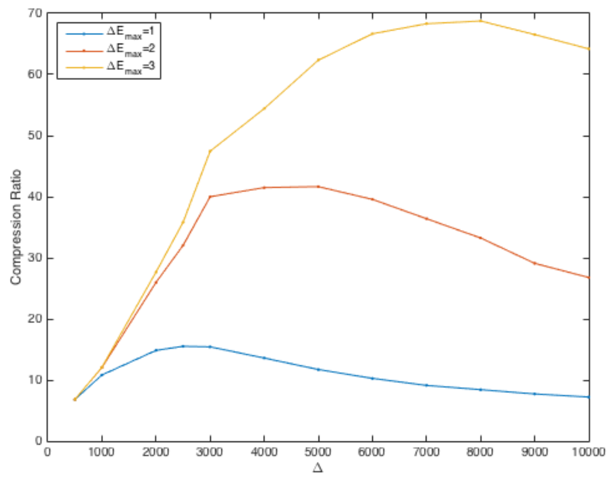


*Figure 8. Compression ratio delta curve.*

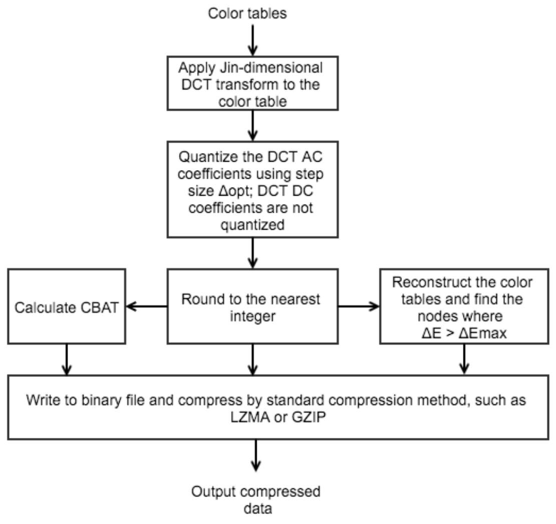

Color tables
Apply Jin-dimensional DCT transform to the color table
Quantize the DCT AC coefficients using step size Δopt; DCT DC coefficients are not quantized
Calculate CBAT
Round to the nearest integer
Reconstruct the color tables and find the nodes where ΔE > ΔEmax
Write to binary file and compress by standard compression method, such as LZMA or GZIP
Output compressed data

*Figure 9. Modified workflow of the compression method in which additional nodes are stored.*

## Decoding Process

The decoding process extracts CBATs, quantized DCT coefficients, and the nodes locations and values from the compressed data.

Figure 10 shows the workflow of the decoding process. First, we apply the standard lossless decompression technique, such as inverse LZMA or inverse GZIP, to the compressed data to get the binary stream. Next we use the CBATs to tell how many bits of the binary stream belong to each node location. Then we can get the reconstructed DCT coefficients. We apply the Inverse DCT transform to the coefficients, multiply by the quantizer step size $\Delta$, and round to the nearest integer to obtain the initial reconstructed color tables. We add back the additionally stored nodes to the initial reconstructed color tables, and obtain the final reconstructed color tables. Then the reconstructed $J_{in}$-dimensional to $J_{out}$-dimensional color tables can be used.

## Evaluation Process

The method we discussed above can be used for A2B and B2A tables. When compressing B2A color tables, we use A2B color tables to evaluate. When compressing A2B color tables, we evaluate the color difference directly. For different input and output spaces, we will use different evaluation metrics, accordingly.

Now we will describe the evaluation process of the CIELAB space to give a more concrete idea. After compressing the B2A tables, our evaluation process will measure the color difference between the original color table and the reconstructed color table in CIELAB space.

For a given input triple $L^*a^*b^*$ as in the B2A table, we use the tetrahedral interpolation [12] method for the original color table and the reconstructed color tables separately, and obtain the output triple or quadruple values. The choice of this interpolation method is based on the premise that it is what would be used in the implementation of the color tables in an actual color imaging system workflow. If this is not the case, a different interpolation method can be used, as appropriate. If the B2A table is from $L^*a^*b^*$ to $CMYK$, then the output quadruples are $CMYK$ and $CMYK_r$ for the original and reconstructed tables, respectively. Using A2B tables which are the output triple or quadruple to $L^*a^*b^*$, we transform tables to interpolate the output triple or quadruple, and obtain $L^*a^*b^*$ and $L^*a^*b^*_r$. We calculate the color difference $\Delta E$ between the original table and the reconstructed table. The color difference between two samples in CIELAB can be calculated using [13]:

$$\Delta E = \sqrt{(L^* - L^{*\prime})^2 + (a^* - a^{*\prime})^2 + (b^* - b^{*\prime})^2}, \qquad (6)$$

where $L^*, a^*, b^*$ are for the original color tables, and $L^{*\prime}, a^{*\prime}, b^{*\prime}$ are for the reconstructed color tables. The error is computed at all points in the input color space. We can also use other models to calculate the $\Delta E$, such as CIECAM02 [14] and CIEDE2000 [15]. While using other measures for color difference may be expected to have some effect on the compression ratios that can be realized, our overall method and the benefits that can be achieved with it should still be valid. Figure 11 describes the evaluation process.

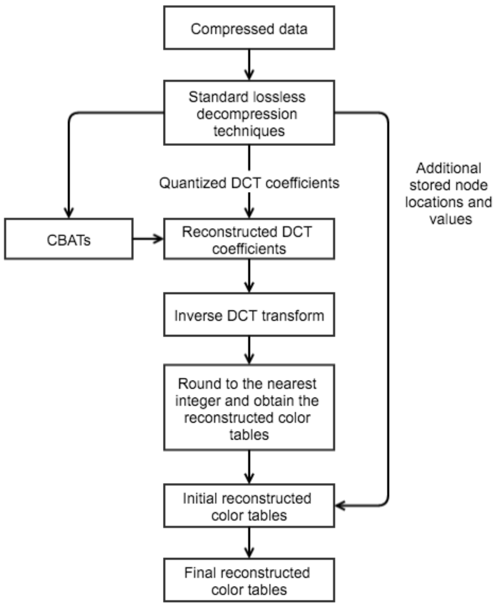

Compressed data
Standard lossless decompression techniques
Quantized DCT coefficients
Additional stored node locations and values
CBATs
Reconstructed DCT coefficients
Inverse DCT transform
Round to the nearest integer and obtain the reconstructed color tables
Initial reconstructed color tables
Final reconstructed color tables

***Figure 10.*** *Decoding process.*

After compressing the B2A color tables, we transform to A2B color tables in order to calculate the color difference. After compressing the A2B color tables, we calculate the color difference directly from the compression output triples instead of transforming to B2A color tables and calculate.

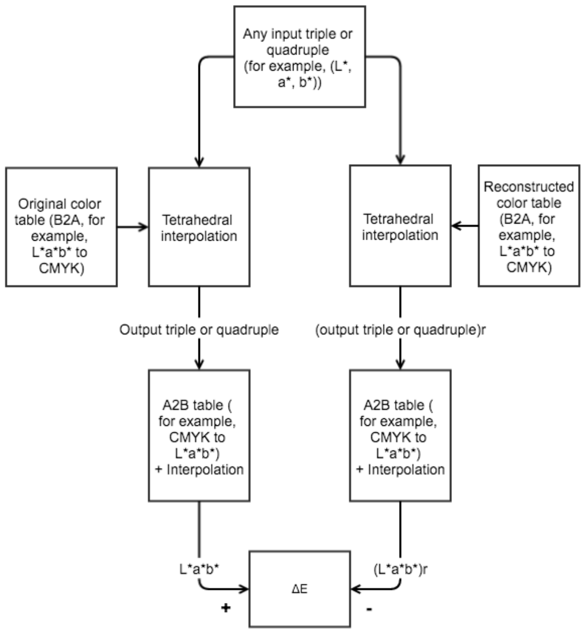

Any input triple or quadruple (for example, (L*, a*, b*))
Original color table (B2A, for example, L*a*b* to CMYK)
Tetrahedral interpolation
Tetrahedral interpolation
Reconstructed color table (B2A, for example, L*a*b* to CMYK)
Output triple or quadruple
(output triple or quadruple)r
A2B table ( for example, CMYK to L*a*b*) + Interpolation
A2B table ( for example, CMYK to L*a*b*) + Interpolation
L*a*b*
(L*a*b*)r
ΔE
+
-

***Figure 11.*** *Evaluation workflow.*

## Results

Some sample ICC profiles can be downloaded from [16]. We present experimental results for the CGATS21_CRPC7.icc profile. In this profile, the B2A tables are from $L^*a*b^*$ to $CMYK$, so $J_{in} = 3$ and $J_{out} = 4$. And $M = 33$, so there are $M^{J_{in}} = 33^3$ nodes for each output channel. The byte depth of the table is $b = 2$. In total, there are $bJ_{out}M^{J_{in}} = 2 \cdot 4 \cdot 33^3 = 287,496$ bytes in one B2A table. There are $N = 3$ rendering intents, so we have $287,496 \cdot 3 = 862,488$ bytes for all the B2A tables. These B2A tables are the tables we are going the compress. The A2B tables are from $CMYK$ to $L^*a^*b^*$, so $J_{in} = 4$ and $J_{out} = 3$. And there are $17^4$ nodes for each output channel. Table 1 shows the parameter values for each type of color table. We are going to compress the B2A tables first, and use the A2B tables to evaluate the compression performance.

From Fig. 8 we can see that for a maximum target $\Delta E_{max} = 3$, a compression ratio of 69 can be achieved. Table 2 shows for $\Delta E_{max} = 1, 2$, or 3, the highest compression ratio that can be achieved, the average $\Delta E$, and the standard deviation of the $\Delta E$ among all the color tables in the ICC profile. Figure 12 shows the compression performance curves of the average, standard deviation and maximum $\Delta E$ among all color tables in the ICC profile mentioned above. For a specific $\Delta E_{max}$, we can look up from the figure what is the highest compression ratio that can be achieved. From the results, we can also see that the average $\Delta E$ and standard

**Table 1: Parameters for the Color Tables**

| Type | B2A | A2B |
|---|---|---|
| Byte Depth | 2 | 2 |
| $J_{in}$ | 3 | 4 |
| $J_{out}$ | 4 | 3 |
| $M$ | 33 | 17 |
| $N$ | 3 | 3 |
| Total Size (Bytes) | 862,488 | 1,503,378 |

deviation are quite small, which indicates good performance.

**Table 2: Compression Performance**

| $\Delta E_{max}$ | Compression Ratio | Average $\Delta E$ | Std.Dev. $\Delta E$ |
|---|---|---|---|
| 1 | 15.52:1 | 0.39 | 0.25 |
| 2 | 41.65:1 | 0.60 | 0.40 |
| 3 | 68.76:1 | 0.78 | 0.54 |

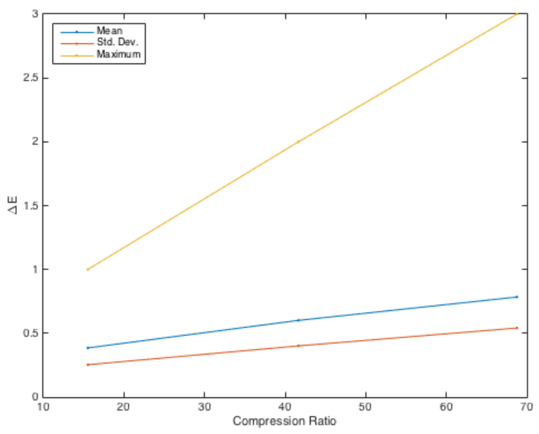


***Figure 12.*** *Compression performance curve.*

## Conclusion

We have proposed a compression framework for color look-up tables, for which there is no constraint on the dimensions of the input or output color spaces. Our compression method enables the encoding of color tables in the ICC profile by storing the information in three parts: Coefficient Bit Assignment Tables (CBATs), quantized DCT coefficients, and additional nodes. It also provides the flexibility to store customized color nodes. Our multi-dimensional DCT compression method provides significant compression and small color difference at the same time. The current A2BX and B2AX tags do not support a compression scheme, thus providing such a mechanism within the framework of the ICC would be beneficial. As ICC profiles are widely used in various devices and platforms, our compression method has a wide application prospect. This method can also be extended to other color workflows.

## Future Work

We can explore if the smoothness of the compressed look-up tables can be preserved in the future.

## Acknowledgments

This research is supported by HP Inc., Boise, Idaho.

## References

[1] Specification ICC.1:2010. Image technology colour management architecture, profile format, and data structure. *International Color Consortium*, 2010.

[2] Aravindh Balaji, Gaurav Sharma, Mark Shaw, and Randall Guay. Preprocessing methods for improved lossless compression of color look-up tables. *Journal of Imaging Science and Technology*, 52(4):40901–1, 2008.

[3] R Sudhakar, R Karthiga, and S Jayaraman. Image compression using coding of wavelet coefficients–a survey. *ICGST-GVIP Journal*, 5(6):25–38, 2005.

[4] Jerome M Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *Signal Processing, IEEE Transactions on*, 41(12):3445–3462, 1993.

[5] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, Jun 1996.

[6] Beong-Jo Kim, Zixiang Xiong, and W. A. Pearlman. Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT). *IEEE Transactions on Circuits and Systems for Video Technology*, 10(8):1374–1387, Dec 2000.

[7] Xiaoli Tang and William A Pearlman. Three-dimensional wavelet-based compression of hyperspectral images. In *Hyperspectral Data Compression*, pages 273–308. Springer, 2006.

[8] Andrew B Watson. Image compression using the discrete cosine transform. *Mathematica journal*, 4(1):81, 1994.

[9] Boon-Lock Yeo and Bede Liu. Volume rendering of DCT-based compressed 3D scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):29–43, Mar 1995.

[10] 7 zip software development kit. http://www.7zip.org/sdk.html. accessed Sept. 2007.

[11] IETF RFC1952. GZIP file format specification version 4.3. May 1996.

[12] Henry R Kang. *Color technology for electronic imaging devices*. SPIE press, 1997.

[13] Alan R Robertson. The CIE 1976 color-difference formulae. *Color Research & Application*, 2(1):7–11, 1977.

[14] M Ronnier Luo, Guihua Cui, and Changjun Li. Uniform colour spaces based on CIECAM02 colour appearance model. *Color Research & Application*, 31(4):320–330, 2006.

[15] M Ronnier Luo, Guihua Cui, and B Rigg. The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application*, 26(5):340–350, 2001.

[16] ICC profiles. http://www.color.org/profiles2.xalter.

## Author Biography

*Chuohao Tang received her BS in communication engineering from Beijing University of Posts and Telecommunications (2011). Currently, she is a Ph.D candidate at Purdue University. Her work has focused on image quality, color science, data compression, and halftoning.*