Hierarchical Compression of Color Look Up Tables

Aravindh Balaji S.R.[*], *Gaurav Sharma*[*+], *Mark Q. Shaw*[‡], *and Randall Guay*[‡]
[*] *Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627-0126, USA*
[+] *Department of Biostatistics and Computational Biology, University of Rochester, Rochester, NY 14627-0126*
[‡] *Hewlett Packard Company, MS 227, 11311 Chinden Blvd, Boise, Idaho, 83714 USA*
rajagopa@ece.rochester.edu, gaurav.sharma@ rochester.edu, mark.q.shaw@hp.com, and randall.guay@hp.com

Abstract

Color lookup tables (CLUTs) that are embedded in printer firmware consume precious flash memory. In order to conserve memory and thereby reduce cost, it is desirable to compress CLUTs prior to storage and restore tables as required. In this paper, we investigate methods for lossless compression of CLUTs. We demonstrate that, through suitable pre-processing of data, significant improvements in compression performance can be obtained as compared to a direct application of lossless data compression methods. In particular, the gains in performance with the proposed methods are accomplished by exploiting the characteristics of CLUT data and utilizing a combination of hierarchical differential encoding and re-ordering for the pre-processing. Experimental results over a representative data set indicate that the proposed methods can result in a 26% reduction in memory requirements in comparison to direct compression. As compared to uncompressed tables this corresponds to a memory saving of 68%.

Keywords: Color look-up table, compression, ICC Profile, space filling curve, differential encoding

1. Introduction

Color lookup tables (CLUTs) that provide transformations between various color spaces are extensively used in color management, common examples being the transformations from device independent color spaces (such as CIELAB) to device dependent color spaces (e.g. CMYK) and vice versa. For color printers, these CLUTs are often embedded in the printer hardware, where they require relatively expensive flash memory for the purpose of storage. In these scenarios, the firmware memory requirements for storing these CLUTs can become a concern, particularly as the number of the look up tables in color devices increases due to the need for supporting multiple color spaces, print media, and preferences. The trend toward finer sampling of the spaces and larger bit depths also results in an increase in table sizes, further exacerbating these memory concerns. Compression of these CLUTs therefore becomes desirable for the purpose of conserving memory. In this paper, we investigate methods for CLUT compression, with the specific objective of improving compression performance over what is achievable with generic data compression methods.

Though firmware memory is the primary motivation for our research in this paper, the same concerns of memory and storage space are applicable for CLUTs that are embedded in color documents (for e.g., ICC source profiles). In applications where embedded profiles are utilized, the increased size due to the embedded profiles represents an overhead, which CLUT compression methods can help mitigate. As color workflows move from the current default sRGB assumption, to more general color spaces the use of embedded profiles is likely to increase. The reduction in overhead due to improved CLUT compression is also likely to offer a benefit in these applications.

Although there are no references for CLUT compression in prior literature, compression methods have been extensively studied for a number of applications (see for example [1]). Compression methods can be categorized as lossless or lossy. The former class of methods is used extensively for data compression, where it is required that the recovered data from the compressed representation must match the original data exactly, i.e., the decompressed data must be mathematically equal to the original data. Lossy compression methods on the other hand are commonly used in applications where requirement of mathematical equality between the compressed and decompressed forms of data can be relaxed and distortions that are perceptually negligible (or small) can be tolerated. Speech, audio, image, and video communication are common examples of such applications. In this paper, we focus our attention on lossless compression techniques for CLUTs[1].

In Section 2 we provide a brief overview of lossless compression methods which motivate the development of a number of methods for CLUT compression that are presented in Section 3. Experimental results presenting the performance of these methods are presented in Section 4. Concluding remarks form Section 5.

2. Lossless Data Compression

Techniques for lossless data compression lie in one of two main categories: the first class of methods attempts to build an (adaptive) dictionary of "patterns" observed in the data and represents specific data values encountered as pointers to the dictionary location in which the values can be found. The LZW [3,4] compression method and its variants, including those used for zip files[1], are common examples of this first class of methods. These methods perform extremely well when the data contains frequent and long repeats since the dictionary representation becomes extremely efficient under these circumstances. A second class of methods operates by modeling the probability distribution of observed data values. In order to achieve compression, frequently occurring symbol values are assigned shorter binary representations and longer representations are used for infrequent symbol values (while maintaining distinctness that allows

[1] Fundamentally, this is not a restriction and one may consider lossy methods for CLUT compression. These may be particularly attractive for finely sampled and high bit depth tables.

reconstruction of the original values from these representations). Huffman coding and arithmetic coding [1, 6] are representative of this class of techniques.

3. Hierarchical CLUT Compression

Both classes of lossless compression methods described in the preceding section operate on a sequential "stream" of input data. In order to apply these methods to multi-dimensional constructs, such as CLUTs, it is necessary to arrange the multi-dimensional data in the form of a one-dimensional stream, a process that we will refer to as *serialization* of the CLUT. In general serialization is accomplished by defining the sequence in which the nodes of the multi-dimensional LUT are traversed, producing the one-dimensional stream. Typically, the order in which data is stored (e.g. row-major/column-major) in computer memory constitutes the natural sequence for obtaining the one dimensional data stream. However, from the characteristics of the compression methods outlined in the preceding section, it is apparent that improved compression may often be obtained simply by rearranging the data. This will be the case, if for instance, the re-ordering increases repetition within the data. Though it is feasible to determine a data adaptive re-ordering [5], it can be computationally demanding, making it unsuitable for the target application. In addition, adaptive ordering requires that the specification of the order in which the data was rearranged also be stored in the compressed representation so that this may be undone during decompression. Instead, we propose a simpler alternative for data rearrangement that utilizes the knowledge that the data represents a CLUT.

The data in a CLUT invariably represents a *smooth and continuous transform* from the space of input variables to the space of output variables. This is clearly the case for forward device tables that represent the response of a physical device (in some colorimetric space) as a function of the control values that define the LUT indices. For inverse tables, that map device independent colorimetric values to corresponding device control values, smoothness and continuity are again necessary requirements in order to ensure reasonable behavior in the presence of device drift. Thus, for instance, a smooth black generation function is necessary when computing the color correction table for a printer [12].

Space Filling Curves for Serialization

Since the data in the CLUT represents a continuous transform, nodes of the CLUT that are close to each other, contain output values that are also close to each other. This characteristic, while clearly desirable from a compression perspective, is true only in the multi-dimensional representation of the LUT and the one dimensional serialization of the data into a sequence for the purpose of compression does not necessarily preserve this trait. Specifically, consider Fig. 1 where a 2-D LUT is used for illustration. Fig. 1 (a) depicts the natural "raster-scan" order corresponding to typical ordering of the LUT data in memory. A serialization of the data in this order results in a jump from the end of one line to the start of the next, since these LUT nodes are not adjacent, this in all likelihood will cause a discontinuity in the observed values that will likely reduce data repetition in the one-dimensional stream and adversely impact compression performance. Figure 2 (a) illustrates this point where a segment of serialized CLUT data obtained through a traversal in this natural "raster" order is shown, observe that the LUT output values show several discontinuous jumps including a abrupt large jump at index 4914. In order to eliminate these discontinuities that are artificially induced by the data serialization, we propose instead to obtain the one-dimensional data through the traversal of the multi-dimensional CLUT along a space filling curve [2, 7, 8, and 9] that assures that the data is traversed in a manner that preserves continuity in the input space of the multi-dimensional CLUT. Figures 1 (b) and (c) illustrate the CScan and Hilbert [10] Space filling curves as examples. Figure 2 (b) illustrates a segment of CLUT data serialized according to the CScan order. Note that (as may be expected from our preceding discussion) the resulting data stream does not have any major discontinuities. In addition to preservation of continuity, a desirably quality of space filling curves is the preservation of contiguity, i.e., the multi-dimensional data nodes that are close to each other should appear close to each other in the serialized data. We can see that in this respect, the Hilbert curve is preferable to the CScan.

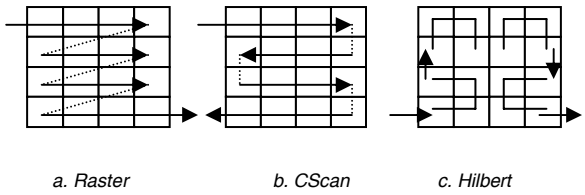


Figure 1*: 2-D Examples of Space Filling Curves*

Hierarchical Differential Encoding

Differential encoding of data is a technique commonly employed in data compression. For a one dimensional data sequence, differential encoding operates by computing differences among adjacent values, mathematically specified by the relation

$$Y_n = X_n - X_{n-1} \text{ and } Y_0 = X_0 \qquad (1)$$

where Xn and Yn denote, respectively, the input and output of the differential encoder at "time" n. When the input data is highly correlated, the process of differential encoding reduces the variance of the data. This therefore increases the propensity for repetition of data values, thereby aiding lossless compression.

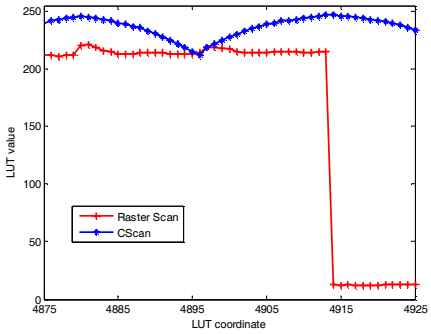


Figure 2: *Serialized Segment of CLUT Data for Raster and CScan orders. Note discontinuity in raster order is not present in CScan order.*

Noting that the CLUT is in fact multi-dimensional, we propose a hierarchical extension of the conventional differential encoding method wherein the differential encoding is performed at multiple levels of the hierarchy corresponding to the dimensions of the CLUT, viz, node, plane, cube, hyper-cube, etc. Furthermore, the differential encoding may also be recursively performed at different levels of the hierarchy. The resulting algorithm is summarized in Matlab-style vectorized pseudo-code in Fig. 3. We refer to this as recursive hierarchical differential encoding (RHD). The alternative algorithm obtained without the recursion will be referred to as non-recursive hierarchical differential (NRHD) encoding which can be expressed similarly in algorithmic form.

The process of RHD may be interpreted as encoding the LUT transformation in terms of its finite differences along the coordinates corresponding to the input axes of the CLUT. For typical CLUTs, values at adjacent nodes are highly correlated since the CLUT represents a relatively fine sampling of a smooth transformation between different color spaces. As a result, the output from the differential encoding process has a significantly lower variance than the input. This can be seen in Fig. 4, where the histograms of the input and the output of RHD are shown, respectively, in parts (a) and (b). Note the significant reduction in variance in the RHD encoded data in comparison to the original CLUT data.

```
diff =clut;
for i = q:-1:2  { diff(i, 1:q,1:q) -= diff(i-1, 1:q,1:q); }
for i=1:q {
  for j = q:-1:2
   { diff(i, j,1:q) -= diff(i, j-1,1:q); }
     for j=1:q {
       for k = q:-1:2
        { diff(i, j,k) -= diff(i, j,k-1); }
    }
}
```

a.

```
lut = construct;
for j = 1:q {
   for k = 1:q {
      for l = 2:q {  lut(j,k,l) = lut(j,k,l) + lut(j,k,l-1); }
            }
   for k = 2:q { lut(j,k,1:q) = lut(j,k,1:q) + lut(j,k-1,1:q); }
          }
for j = 2:q { lut(j,1:q,1:q) = lut(j,1:q,1:q) + lut(j-1,1:q,1:q); }
```

b.

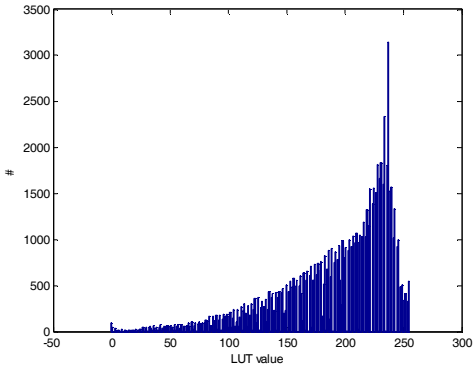
Figure 3: *Recursive Hierarchical Differential (a) Encoding and (b) Decoding algorithm for a qxqxq 3D CLUT.*

Compression

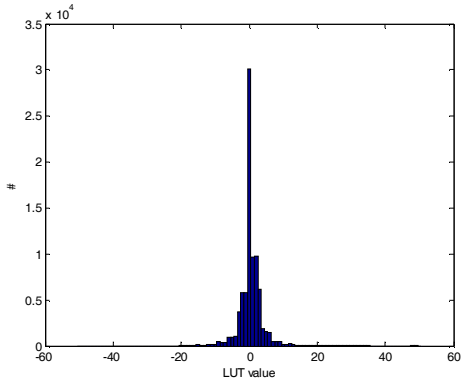
Once CLUT data is pre-processed through the hierarchical differential encoding and data reordering steps, it may be compressed by a lossless compression method. For this purpose we consider four different lossless compression methods:

(i) gzip [11]: A dictionary based compression method.

(ii) Adaptive Arithmetic Coding (AAC)[1, 6]: A probabilistic modeling based technique.



a.



b.

Figure 4*: Histogram of CLUT Data (a) before and (b) after NRHD coding*

(iii) bzip2 [13]: A compressor that combines Burrows Wheeler block sorting transform and Huffman Coding.

(iv) Lempel Ziv Markov Algorithm (LZMA)[14]: A recent dictionary based compression method with a variable compression dictionary size.

LZMA and bzip2 methods are recently developed methods that offer better compression performance than gzip and AAC. LZMA, in particular, is among the best performing lossless compression methods available today.

4. Experimental Results

To assess the performance of the proposed method, it was implemented in C++ and MATLAB. A collection of CLUTs representative of a typical printer configuration was used to perform the evaluation. All CLUTs had 8 bit values for each output channel. The tables included:

a) 4 forward CMYK device tables (mapping to device independent color space) each of size 17^4 x 3, totaling 1,002,252 bytes. These correspond to three standard CMYK tables (SWOP, EURO, and DIC) and one device response table (obtained by two different methods),

b) 4 forward RGB device tables, each of size 17^3 x 3, totaling 58,956 bytes, corresponding to different RGB input spaces, and

c) 8 pairs of inverse printer (correction) tables (mapping from device independent color space to printer CMYK), each of size 17^3 x 4, for a total of 314,432 bytes. The 8 pairs correspond to the 8 different media choices for the printer (transparency, plain paper, coated paper, etc) and two elements in each pair differed in their black generation.

The evaluation of compression was performed using the NRHD and RHD encoding methods along with one out of CScan, Hilbert and Raster Scan options for the serialization of CLUT data. Four compression methods – LZMA, bzip2, gzip, and adaptive arithmetic coding (AAC) – were evaluated. In addition to the proposed pre-processing based methods, direct compression of the tables was also performed using each of these methods for the purpose of benchmarking our performance improvement.

From the information on the sizes of the different tables provided above, we see that the CMYK tables account for a major part (72.86%) of the CLUT data. We therefore begin by comparing the performance for different methods over these tables. We first demonstrate the advantage of the proposed methods over direct compression. Table 1, compares the performance of proposed methods (NRHD + CScan + Compression) against direct compression for the four CMYK tables. From the tabulated values we can see that the proposed methods perform uniformly and significantly better than direct compression. Furthermore, we see that the performance across tables is rather similar and therefore averaged values over the tables may be used to draw useful and general conclusions.

Next we consider the performance across the different variants of the proposed methods. Table 2 lists the average compression ratios over the CMYK table data set for the different possible methods consisting of one of NRHD/RHD for differential encoding, Raster/CScan/Hilbert for data re-ordering, and LZMA/bzip2 for compression. From a comparison of the numbers in Tables 2(a) and 2(b), it is apparent that the NRHD + CScan + LZMA method performs best with NRHD + Hilbert Scan + LZMA as a fairly close second.

In Table 3 we compare the performance of the aforementioned two methods against direct compression over the complete data set of printer tables. Once again from the compression ratios enumerated in Table 3, we see that the proposed methods offer significant gains over direct compression. The best performing method (NRHD + CScan + LZMA) offers a compression ratio of approximately 3.2, which represents a 68.32% reduction in size over the uncompressed binary representation and a 25.69% reduction in size in comparison with direct compression.

Finally, in order to determine the impact of finer sampling on compression performance, we also conducted the experiments again using tables of size 33^4 and 33^3 respectively, instead of 17^4 and 17^3. The performance of the different methods relative to each other follows the same trends as observed for the coarsely sampled CLUTs. We therefore present only the summary results over the ensemble of printer CLUTs for the better performing methods that were included in Table 3. Table 4 lists the average compression ratios over the complete ensemble of printer CLUTs for direct compression with LZMA/bzip2 and for NRHD with Hilbert/CScan reordering followed by LZMA. Comparing the tabulated compression ratios against the corresponding values in Table 3, we see that much higher compression ratios (better compression performance) are obtained for the finer sampled versions of the CLUTs. The average compression ratio for the best method (NRHD + CScan + LZMA) over the complete data set in Table 4 was found to be 4.8457 as opposed to 3.1567 listed in Table 3. This illustrates that with finer sampling, compression using the proposed method offers greater gains, partly offsetting the increase in table size.

The finer sampling used here is more representative of ICC profiles than the coarse CLUTs considered for our results in Tables 1-3 (which were more representative of hardware CLUTs). Thus the numbers in Table 4 illustrate that CLUT compression can significantly reduce the memory and storage overhead caused by profile embedding in documents. For our data set, the compressed profile CLUTs takes only 20.64% of the space required by their uncompressed versions.

5. Conclusion

Significant gains in Color Lookup Table (CLUT) compression can be obtained through pre-processing of CLUT data prior to compression. Utilizing the knowledge of CLUT structure for hierarchical differential encoding and re-ordering the data using a space filling curve can reduce storage requirements significantly. For the best performing LZMA method the savings are approximately 29% when compared with direct compression (without preprocessing) and approximately 68% when compared to uncompressed binary tables. The methods proposed here are computationally simple and readily implemented in printer firmware without an undue overhead in computation. With finer sampled CLUTs the gains are even more significant.

Future work will investigate the extension of this algorithm to higher bit depths, and the use of improved methods of prediction.

Acknowledgment

The authors would like to thank Dr. Marcelo J. Weinberger and Dr. Erik Ordentlich of HP Labs, Palo Alto, California for useful suggestions and discussions.

References

[1] Khalid Sayood, Introduction to Data Compression, Third Edition, Morgan Kaufmann, San Francisco, CA, 2005, pp. 1-65

[2] Hans Sagan, Space Filling Curves, Springer-Verlag, 1994, pp. 1-29

[3] Welch T.A., "A technique for high performance data compression," *Computer* vol. 17, June 1984, pp. 8-19

[4] Jacob Ziv and Abraham Lempel, "Compression of Individual Sequences via variable rate coding", *IEEE Transactions on Information Theory*, vol. 24, no. 5, September 1978, pp. 530-536

[5] Slobodan Vucetic, "A fast algorithm for Lossless Compression of Data Tables by Reordering", *Proceedings of Data Compression Conference*, March 2006

[6] Guy E.Blelloch, "Introduction to Data Compression", pp. 15-21 URL: **www.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/compression.pdf**

[7] Sei-ichiro Kamata, Michiharu Nimmi, and Eigi Kawaguchi, "A Gray Image Compression using Hilbert Scan", *Proceedings of ICPR*, 1996, pp. 905-909

[8] Zhexuan Song and Nick Roussopoulous, "Using Hilbert Curve in Image Storing and Retrieving", *Informations Systems*, vol. 27, no. 8, 2002, pp. 523-536
[9] Theodore Bially, "Space Filling Curves:Their generation and their application to bandwidth reduction", *IEEE Transactions on Information Theory*, vol. 15, no. 6, Nov. 1969, pp. 658-664
[10] David Hilbert, "Ueber stetige Abbildung einer Linie auf ein Flachenstuck", *Mathematische Annalen*, vol. 38, 1891, pp. 459-460
[11] Jean-loup Gially, Gzip Users Manual, **www.gnu.org/software/gzip/manual/gzip.html**
[12] Raja Bala, "Device Characterization", Chapter 5 in *Digital Color Imaging Handbook*, G. Sharma Ed., CRC Press, 2003, pp. 269-379
[13] URL: **www.bzip.org/**
[14] 7-zip Software Development Kit, URL**: www.7-zip.org/sdk.html**

Biography

Aravindh Balaji S.R. received his B.E. degree in Electrical and Electronics Engineering from Anna University, Chennai, India in May 2006. Since September 2006, he is pursuing his Master's degree in Electrical and Computer Engineering from University of Rochester, Rochester. His interests include image processing, color imaging, and data compression. He is a student member of IEEE and IS&T.

Table 1: Compression ratios for direct compression and hierarchical differential (NRHD + CScan) compression across 4 forward CMYK device tables. Each uncompressed table size is 17^4 x 3 = 250563 bytes

CLUT	Compression method							
	(Direct)				NRHD + CScan +			
	gzip	AAC	bzip2	LZMA	gzip	AAC	bzip2	LZMA
Device 1	1.7637	1.1203	2.1412	2.4655	2.6302	2.3968	2.8312	3.0657
DIC	1.8214	1.2121	2.1848	2.2511	2.5026	2.3895	2.6648	2.8933
EURO	1.9284	1.1740	2.2979	2.3021	2.7354	2.5254	2.9591	3.2286
SWOP	1.9155	1.2640	2.2814	2.4533	2.8639	2.2990	3.2487	3.4485

Table 2: Average compression ratios for Recursive Hierarchical Differential (RHD) and Hierarchical Differential (NRHD) Compression methods. Averages are computed over 4 forward CMYK device tables, where the size of each table is 250563 bytes in uncompressed format.

Compressor	Preprocessing					
	RHD +			NRHD +		
	CScan	Hilbert	Raster	CScan	Hilbert	Raster
gzip	2.3125	2.3012	2.3038	2.6830	2.6361	2.2903
AAC	2.6257	2.6330	2.6288	2.4027	2.4076	2.3456
bzip2	2.3835	2.3125	2.3669	2.9260	2.8485	2.3669
LZMA	2.5784	2.5751	2.5612	3.1458	3.0650	2.8272

Table 3: Compression ratios (total uncompressed bytes / total compressed bytes) observed over a representative data set of printer CLUTs with 17 nodes along each input dimension of the CLUT. The total size of the uncompressed data for the full set of CLUTs is 1375640 bytes. The best method (NRHD + CScan + LZMA) results in a saving of 1.57 x10^5 bytes over direct compression (a 25.97% reduction) for the full CLUT data set.

Compression Methods		Data set			
		Input tables		Output tables	Full set of tables
		CMYK	RGB		
Direct Compression	LZMA	2.3680	1.6543	2.4126	2.3370
	bzip2	2.3620	1.6866	1.9537	2.2095
NRHD + CScan + LZMA		3.1458	2.2398	3.4237	3.1567
NRHD + Hilbert + LZMA		3.0650	2.2154	3.0409	3.0109

Table 4: Compression ratios (total uncompressed bytes / total compressed bytes) observed over a representative data set of printer CLUTs with 33 nodes along each input dimension of the CLUT. The total size of the uncompressed data for the full set of CLUTs is 16862264 bytes. The best method (NRHD + CScan + LZMA) results in a saving of 8.65 x10^5 bytes over direct compression (a 19.62% reduction) for the full CLUT data set.

Compression Methods		Data set			
		Input tables		Output tables	Full set of tables
		CMYK	RGB		
Direct Compression	LZMA	3.9886	2.6464	3.2635	3.9233
	bzip2	3.8235	2.7399	3.2265	3.7703
NRHD + CScan + LZMA		4.9065	3.7745	4.1444	4.8457
NRHD + Hilbert + LZMA		4.7932	3.5082	3.9471	4.7249