IJCSI International Journal of Computer Science Issues, Volume 12, Issue 3, May 2015
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org

Some Empirical Results of Compressing High Resolution Raster Graphics Images

Chung-E Wang

Department of Computer Science, California State University, Sacramento
Sacramento, CA 95819-6021, USA

Abstract

In this paper, we introduce two simple but effective reversible pre-compression transformations for raster graphics image compressions: horizontal differential and vertical differential. By incorporating the two pre-compression transformations with general purpose compression algorithms such as arithmetic or DLZW compressions, the resulting raster image compression algorithm performs significantly better than the most commonly used lossless image compression method PNG.
Key words: *Raster graphics image, reversible pre-compression transformation, lossless image compression, DLZW compression, arithmetic compression, PNG.*

1. Introduction

As technology advances and the demands of modern society increases, the resolution of digital images becomes higher and higher. Higher resolution images capture more details of pictures and can produce higher quality printouts. However, high resolution images require more storage space and take more time to transmit thru the internet. Thus, the needs of efficient algorithms for compressing high resolution graphic images are ever increasing.

There are basically two types of methods for compressing digital images: lossy and lossless. Lossy compressions, e.g., JPEG (Joint Photographic Experts Group) (1992) [1], create smaller files by discarding (losing) some information about the original image. It discards details and color changes it believes too small for the human eye to differentiate. Lossless compressions, e.g., GIF (Graphics Interchange Format) (1987) [2], TIFF (Tagged Image File Format) (1992) [3], and PNG (Portable Network Graphics) (1996) [4], on the other hand, never remove any information of the original image. In general, lossy compressions result in smaller files than lossless compressions. However, image files produced by lossless compressions have better sharp reproductions and better quality printouts.

Most digital images are created or captured in **raster graphics** format. In computer graphics, a raster graphics image is a data structure representing a generally rectangular grid of pixels, or points of color, viewable via a computer monitor or printable on paper. Raster graphics images are palette-based images (with palettes of 24-bit RGB or 32-bit RGBA colors), grayscale images (with or without alpha channel), or full-color non-palette-based RGB images (with or without alpha channel). To simplify matters, in this paper we always use palettes-based images with palettes of 24-bit RGB in our examples. **Fig. 1** illustrates a raster graphics matrix with palettes of 24-bit RGB, where *h* and *w* are the height and width of the matrix, respectively. Also note that in most image file formats such as BMP, the size of each row is rounded up to a multiple of 4 bytes by padding. BMP (bitmap image file) is a bitmapped graphics format used internally by the Microsoft Windows graphics subsystem (GDI) and used commonly as a simple graphics file format on that platform. It is an uncompressed format.

$R_{0,h-1}$	$G_{0,h-1}$	$B_{0,h-1}$	$R_{1,h-1}$	$G_{1,h-1}$	$B_{1,h-1}$	…	$R_{w-1,h-1}$	$G_{w-1,h-1}$	$B_{w-1,h-1}$	padding
$R_{0,h-2}$	$G_{0,h-2}$	$B_{0,h-2}$	$R_{1,h-2}$	$G_{1,h-2}$	$B_{1,h-2}$	…	$R_{w-1,h-2}$	$G_{w-1,h-2}$	$B_{w-1,h-2}$	padding
…	…	…	…	…	…	…	…	…	…	…
$R_{0,1}$	$G_{0,1}$	$B_{0,1}$	$R_{1,1}$	$G_{1,1}$	$B_{1,1}$	…	$R_{w-1,1}$	$G_{w-1,1}$	$B_{w-1,1}$	padding
$R_{0,0}$	$G_{0,0}$	$B_{0,0}$	$R_{1,0}$	$G_{1,0}$	$B_{1,0}$	…	$R_{w-1,0}$	$G_{w-1,0}$	$B_{w-1,0}$	padding

Fig. 1 Raster graphics matrix with 24-bit RGB

Raster images are known to be uncompressible by general purpose compression algorithms such as LZ family compressions [5, 6, 7], Huffman compression [8] and arithmetic compression [9] alone. Thus, the general idea of compressing a raster image is to apply a pre-compression transformation on the image before using a general purpose compression algorithm. In a lossy compression algorithm, the pre-compression transformation is not reversible and in a lossless compression algorithm, the transformation is reversible.

In Section 2, we describe two pre-compression reversible transformations that can make raster images more compressible. Compared to existing pre-compression transformations used in other lossless image compression algorithms, our transformations are logically simpler and take less time to compute. In Section 3, we summarize some empirical results of effects of our two pre-compression reversible transformations. Finally, in Section 4, we conclude our experiments.

2. Two Simple and Effective Reversible Transformations

The idea of using pre-compression transformations isn't new. For example, Burrows Wheeler Transform (BWT) [10] and Prediction by Partial Matching [11] have long been used to improve the efficacy of general purpose compression algorithms. Moreover, well-known image compression algorithms GIF, TIFF, JPEG, and PNG all use pre-compression transformations. That is, they all incorporate pre-compression transformations with general purpose compression algorithms. **Fig. 2** illustrates the scheme of incorporating general compression/decompression algorithms with pre-compression transformations.

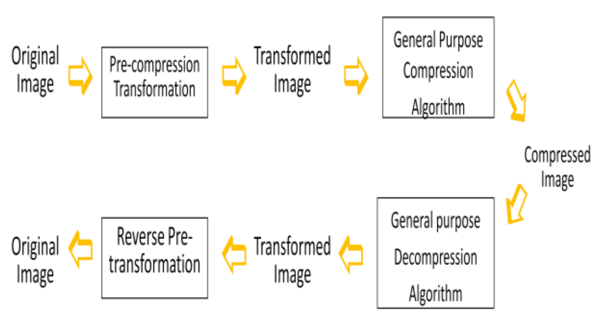

Fig. 2 Compression/Decompression Incorporating a Pre-compression Transformation

In this section we describe two reversible pre-compression transformations suitable for compressing raster graphics images. As in any other well-known image compression algorithms, the basic idea is to replace pixel values with differences of adjacent pixel values. In high resolution images, adjacent pixel values tend to change gradually. Thus the differences will generally be clustered around 0, rather than spread over all possible pixel values.

2.1 Horizontal Differential

In this transformation, every byte except the first byte of every row of the raster graphics matrix is replaced with the difference from the previous byte. **Fig. 3** shows the result of applying the horizontal differential transformation on the raster graphics image matrix of **Fig. 1**.

$R_{0,h-1}$	$G_{0,h-1}-R_{0,h-1}$	$B_{0,h-1}-G_{0,h-1}$	$R_{1,h-1}-B_{0,h-1}$	$G_{1,h-1}-R_{1,h-1}$	$B_{1,h-1}-G_{1,h-1}$	…	$R_{w-1,h-1}-B_{w-2,h-1}$	$G_{w-1,h-1}-R_{w-1,h-1}$	$B_{w-1,h-1}-G_{w-1,h-1}$	padding
$R_{0,h-2}$	$G_{0,h-2}-R_{0,h-2}$	$B_{0,h-2}-G_{0,h-2}$	$R_{1,h-2}-B_{0,h-2}$	$G_{1,h-2}-R_{1,h-2}$	$B_{1,h-2}-G_{1,h-2}$	…	$R_{w-1,h-2}-B_{w-2,h-2}$	$G_{w-1,h-2}-R_{w-1,h-2}$	$B_{w-1,h-2}-G_{w-1,h-2}$	padding
…	…	…	…	…	…	…	…	…	…	…
$R_{0,0}$	$G_{0,0}-R_{0,0}$	$B_{0,0}-G_{0,0}$	$R_{1,0}-B_{0,0}$	$G_{1,0}-R_{1,0}$	$B_{1,0}-G_{1,0}$	…	$R_{w-1,0}-B_{w-2,0}$	$G_{w-1,0}-R_{w-1,0}$	$B_{w-1,1}-G_{w-1,0}$	padding

Fig. 3 Horizontal Differential

2.2 Vertical Differential

In this transformation, every byte except the first byte of every column of the raster graphics matrix is replaced with the difference from the byte above. **Fig. 4** illustrates the result of applying the vertical differential transformation on the raster graphics image matrix of **Fig. 1**.

$R_{0,h-1}$	$G_{0,h-1}$	$B_{0,h-1}$	$R_{1,h-1}$	$G_{1,h-1}$	$B_{1,h-1}$	…	$R_{w-1,h-1}$	$G_{w-1,h-1}$	$B_{w-1,h-1}$	padding
$R_{0,h-2}-R_{0,h-1}$	$G_{0,h-2}-G_{0,h-1}$	$B_{0,h-2}-B_{0,h-1}$	$R_{1,h-2}-R_{1,h-1}$	$G_{1,h-2}-G_{1,h-1}$	$B_{1,h-2}-B_{1,h-1}$	…	$R_{w-1,h-2}-R_{w-1,h-1}$	$G_{w-1,h-2}-G_{w-1,h-1}$	$B_{w-1,h-2}-B_{w-1,h-1}$	padding
…	…	…	…	…	…	…	…	…	…	…
$R_{0,0}-R_{0,1}$	$G_{0,0}-G_{0,1}$	$B_{0,0}-B_{0,1}$	$R_{1,0}-R_{1,1}$	$G_{1,0}-G_{1,1}$	$B_{1,0}-B_{1,1}$	…	$R_{w-1,0}-R_{w-1,1}$	$G_{w-1,0}-G_{w-1,1}$	$B_{w-1,0}-B_{w-1,1}$	padding

Fig. 4 Vertical Differential

3. Some Empirical Results

In order to demonstrate the effectiveness of our pre-compression transformations we need graphics images which have never been thru compressing and decompressing of any lossy image compression algorithm. To do so, we used a Canon camera to take 10 pictures, saving them in the uncompressed TIFF format, and then converted them to the BMP format using the Microsoft Paint program. Our sample pictures are captured in 3648×2736 resolution with palettes of 24-bit RGB and thus have a raster graphics matrix size of 3648×2736×3, i.e. 29,942,784 bytes. Since BMP files created by Paint has a file header of 54 bytes, all resulting BMP files has a size of 29,942,838 bytes. In this section, we ignore the file headers of images files and only compare the resulting compressed sizes of raster image matrices.

Besides the effectiveness of individual pre-compression transformation, we also show the effectiveness of the combination of the two pre-compression transformations.

3.1 No Reversible Transformation

First we compress the 10 sample raster images without any pre-compression transformations. Four general compression algorithms, arithmetic, Huffman, LZW [7], and DLZW [12] have been used. LZW compression is Terry Welch's implementation of LZ compression. DLZW (Dynamic LZW) compression is Wang's modification of LZW. Without any pre-compression transformations, LZW inflates image sizes. On average, arithmetic and Huffman compressions reduce image sizes by 10% and DLZW reduces image sizes by 35%. The results are summarized in **Table 1**. In the table, we use color to high light the best case of each sample file. Clearly, without any pre-compression transformation, DLZW works better than other compression methods.

3.2 With Horizontal Differential

In this section, we test sample images with combinations of the horizontal differential transformation and general compression algorithms. With only horizontal differential transformation, LZW still inflates image sizes. On average, arithmetic and Huffman compressions reduce image sizes by 18% and DLZW reduces image sizes by 41%. The results are summarized in **Table 2.**

3.3 With Vertical Differential

In this section, we test the effectiveness of the vertical differential transformation. With vertical transformation, on average, LZW reduces image sizes by 55%, arithmetic and Huffman compressions reduce images sizes by 45% and DLZW reduces image sizes by 57%. The results are recapitulated in **Table 3.** Clearly, vertical differential transformation can significantly enhance the efficacy of the compression.

3.4 With Combination of Horizontal Differential and Vertical Differential Transformations

In this section, we demonstrate that the horizontal and the vertical differential transformations can be used together to further improve the compression efficacy achieved by either transformation alone. With the two transformations combined, on average, LZW reduces image sizes by 46%, arithmetic and Huffman compressions reduce images sizes by 47% and DLZW reduces image sizes by 58%. The results are exhibited in **Table 4.** Surprisingly, in more than 50% of the cases, arithmetic compression works better than DLZW compression when combined with both transformations even though DLZW works better on average and when incorporated with only one individual transformation.

3.5 Combination of Horizontal and Vertical Differential Transformations VS. PNG

In this section, we compare our raster graphic image compression algorithms with PNG image compression algorithm. To do so, first, we use Microsoft Paint to convert our sample uncompressed TIFF files to PNG files and then compare PNG file sizes with sizes of those images produced by our algorithms. Note that since our pre-compression transformations are very simple, if we want to save the compressed images in files we only need simple file headers as BMP file headers. Since the header size of BMP files created by Paint is 54 bytes long, in **Table 5**, we add 54 bytes to the sizes of the resulting images produce by our resulting image compression algorithms.

As shown in **Table 5**, on average, by incorporating any one of the general-purpose compression algorithms, the combination of our two pre-compression transformations is 9% better than the PNG algorithm.

4. Conclusion

In this paper we have described two simple pre-compression transformations, horizontal differential and vertical differential, suitable for raster image compressions. We have tested our transformations with general purpose compression algorithms on sample raster graphic images. The test results show that by incorporating both transformations with general purpose compression algorithms, the compression efficacy can be significantly improved. Moreover, as shown in **Table 5**, the combination of our two per-compression transformations can do notably better than the most commonly used lossless image compression method PNG.

References

[1] JPEG Requirements and guidelines (ISO/IEC IS 10918-1 | ITU-T T.81)

[2] "Graphics Interchange Format, Version 89a" online available from the Internet at http://www.w3.org/Graphics/GIF/spec-gif89a.txt.

[3] "TIFF 6.0 Specification" online available from the Internet at http://partners.adobe.com/public/developer/tiff/index.html.

[4] T. Boutell, ed. 1996, "PNG Portable Network Graphics Specification, Version 1.0" online, available from the Internet at http://www.w3.org/TR/REC-png.

[5] J. Ziv, and A. Lempel "A universal algorithm for sequential data compression". IEEE Trans. Inf. Theory 23, 3 (May), 1977, 337-343.

[6] J. Ziv, and A. Lempel, "Compression of individual sequences via variable-rate coding", IEEE Trans. Inform. Theory, 24(5), 1978, 530-536.

[7] T. A. Welch, "A technique for high-performance data compression". Computer 17, 6 (June), 1984, 8-19.

[8] D. A. Huffman, "A Method for the construction of Minimum-redundancy Codes", Proc. IRE, vol.40, no.10, pp., 1952, 1098-1101.

[9] J. Rissanen, and G. G. Landon, "Arithmetic coding", IBM J. Res. Dev. 23, 2 (Mar.), 1979, 149-162.

[10] M. Burrows, and. J. Wheeler, "A Block – sorting Lossless Data compression Algorithm", SRC Research report 124, Digital Research Systems Research Centre.

[11] A. Moffat, "Implementing the PPM Data compression scheme", IEEE Transaction on Communications, 38(11): 1990, 1917-1921.

[12] C. E. Wang, "Dynamic LZW for Compressing Large Files", Proceedings of FCS'11, 57- 60, July 2011.

Table 1: No Pre-compression Transformation

	IMG0001	IMG0002	IMG0003	IMG0004	IMG0005	IMG0006	IMG0007	IMG0008	IMG0009	IMG0010	Average
Arithmetic	28023647 (93.59%)	27224952 (90.92%)	26785347 (89.46%)	26159746 (87.37%)	29486922 (98.48%)	23284100 (77.76%)	29300244 (97.85%)	27107285 (90.53%)	27979304 (93.44%)	22599036 (75.47%)	26795058 (89.49%)
Huffman	28109229 (93.88%)	27305049 (91.19%)	26905675 (89.86%)	26292924 (87.81%)	29589366 (98.82%)	23394981 (78.13%)	29428553 (98.28%)	27268085 (91.07%)	28055759 (93.70%)	22768637 (76.04%)	26911825 (89.88%)
L Z W	36031034 (120.33%)	41402126 (138.27%)	40828744 (136.36%)	38039926 (127.04%)	39591696 (132.22%)	50321798 (168.06%)	45615302 (152.34%)	34701696 (115.89%)	45805418 (152.98%)	27400900 (91.51%)	39973864 (133.50%)
D L Z W	22757020 (76.00%)	21041122 (70.27%)	20359444 (67.79%)	19300520 (64.46%)	23741476 (79.29%)	13446436 (44.91%)	19382456 (64.73%)	19263742 (64.34%)	18044920 (60.26%)	16485742 (55.06%)	19382287 (64.73%)

Table 2: With Horizontal Differential Transformation

	IMG0001	IMG0002	IMG0003	IMG0004	IMG0005	IMG0006	IMG0007	IMG0008	IMG0009	IMG0010	Average
Arithmetic	25632022 (85.60%)	24831641 (82.93%)	24145158 (80.64%)	23489272 (78.45%)	25456683 (85.02%)	20269675 (67.69%)	25398309 (84.82%)	23594805 (78.80%)	24090285 (80.45%)	27915453 (93.23%)	24482330 (81.76%)
Huffman	25747787 (85.99%)	24924712 (83.24%)	24267965 (81.05%)	23598319 (78.81%)	25535120 (85.28%)	20343697 (67.94%)	25536134 (85.28%)	23706421 (79.17%)	24192439 (80.80%)	28051038 (93.68%)	24590363 (82.12%)
L Z W	31436578 (104.99%)	33282152 (111.15%)	34615030 (115.60%)	34047440 (113.71%)	34094392 (113.86%)	40434716 (135.04%)	47263028 (157.84%)	26664050 (89.05%)	32341832 (108.01%)	34633944 (115.67%)	348811316 (116.49%)
D L Z W	20049516 (66.96%)	19303378 (64.47%)	18194160 (60.76%)	17066020 (57.00%)	21263478 (71.01%)	12072766 (40.32%)	17231770 (57.55%)	16125840 (53.86%)	14602002 (48.77%)	20445960 (68.28%)	17635489 (58.90%)

Table 3: With Vertical Differential Transformation

	IMG0001	IMG0002	IMG0003	IMG0004	IMG0005	IMG0006	IMG0007	IMG0008	IMG0009	IMG0010	Average
Arithmetic	18958810 (63.32%)	18321384 (61.19%)	19003710 (63.47%)	18884982 (63.07%)	19186957 (64.08%)	12138779 (40.54%)	15687402 (52.39%)	14108989 (47.12%)	13965046 (46.64%)	11848188 (39.57%)	16210424 (54.14%)
Huffman	19079338 (63.72%)	18415165 (61.50%)	19135595 (63.91%)	19014598 (63.50%)	19312081 (64.50%)	12258491 (40.94%)	15830797 (52.87%)	14193175 (47.40%)	14124947 (47.17%)	12001280 (40.08%)	16336546 (54.56%)
L Z W	15268194 (51.00%)	15627168 (52.20%)	15215130 (50.81%)	15159202 (50.63%)	16718784 (55.83%)	10398768 (34.73%)	14443728 (48.24%)	11252352 (37.58%)	10906508 (36.42%)	10623566 (35.48%)	13561340 (45.29%)
D L Z W	15011766 (50.13%)	14964702 (49.98%)	14712316 (49.13%)	14428962 (48.19%)	15948254 (53.26%)	8501354 (28.39%)	12290522 (41.05%)	10975864 (36.66%)	10348660 (34.56%)	10039498 (33.53%)	12722189 (42.49%)

Table 4: With Horizontal and Vertical Differential Transformations

	IMG0001	IMG0002	IMG0003	IMG0004	IMG0005	IMG0006	IMG0007	IMG0008	IMG0009	IMG0010	Average
Arithmetic	14191762 (47.40%)	13889931 (46.39%)	13997716 (46.75%)	13965976 (46.64%)	14048434 (46.92%)	9128625 (30.49%)	11406570 (38.09%)	10484779 (35.02%)	10049772 (33.56%)	14495673 (48.41%)	12565923 (41.97%)
Huffman	14349101 (47.92%)	14047106 (46.91%)	14207567 (47.45%)	14158080 (47.28%)	14190389 (47.39%)	9240406 (30.86%)	11616196 (38.79%)	10630444 (35.50%)	10182448 (34.01%)	14617725 (48.82%)	12723946 (42.49%)
L Z W	14654036 (48.94%)	14983962 (50.04%)	14441200 (48.23%)	14399094 (48.09%)	15342124 (51.24%)	9834374 (32.84%)	12865856 (42.97%)	10731148 (35.84%)	10406232 (34.75%)	12346298 (41.23%)	13000432 (43.42%)
D L Z W	14498002 (48.42%)	14582276 (48.70%)	14227890 (47.52%)	14019370 (46.82%)	14878412 (49.69%)	8232290 (27.50%)	11718242 (39.14%)	10609036 (35.43%)	9991328 (33.37%)	12056888 (40.27%)	12481373 (41.68%)

Table 5: Combination of Horizontal and Vertical Differential Transformations VS. PNG

	IMG0001	IMG0002	IMG0003	IMG0004	IMG0005	IMG0006	IMG0007	IMG0008	IMG0009	IMG0010	Average
Arithmetic	14191816 (47.40%)	13889985 (46.39%)	13997770 (46.75%)	13966030 (46.64%)	14048488 (46.92%)	9128679 (30.49%)	11406624 (38.09%)	10484833 (35.02%)	10049826 (33.56%)	14495727 (48.41%)	12565977 (41.97%)
Huffman	14349155 (47.92%)	14047160 (46.91%)	14207621 (47.45%)	14158134 (47.28%)	14190443 (47.39%)	9240460 (30.86%)	11616250 (38.79%)	10630498 (35.50%)	10182502 (34.01%)	14617779 (48.82%)	12724000 (42.49%)
L Z W	14654090 (48.94%)	14984016 (50.04%)	14441254 (48.23%)	14399148 (48.09%)	15342178 (51.24%)	9834428 (32.84%)	12865910 (42.97%)	10731202 (35.84%)	10406286 (34.75%)	12346352 (41.23%)	13000486 (43.42%)
D L Z W	14498056 (48.42%)	14582330 (48.70%)	14227944 (47.52%)	14019424 (46.82%)	14878466 (49.69%)	8232344 (27.50%)	11718296 (39.14%)	10609090 (35.43%)	9991382 (33.37%)	12056942 (40.27%)	12481427 (41.68%)
P N G	17146217 (57.26%)	16990564 (56.74%)	16603057 (55.45%)	16296394 (54.43%)	18484673 (61.73%)	11219007 (37.47%)	15541703 (51.90%)	14512070 (48.47%)	13520412 (45.15%)	11840880 (39.54%)	15215497 (50.82%)