Design of Multi-Mode Depth Buffer Compression for 3D Graphics System

Tzung-Rung Jung, Lan-Da Van, Teng-Yao Sheu, Cheng-Wei Lin, Wai-Chi Fang*
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
*TSMC Distinguished Chair Professor, Dept. of Electronics Eng., National Chiao Tung University, Hsinchu, Taiwan
E-mail: ldvan@cs.nctu.edu.tw , wfang@mail.nctu.edu.tw

***Abstract*— An innovative multi-mode depth buffer compression algorithm has been developed for 3D graphics system. It adaptively compresses the depth buffer data according to different scene changes by employing 19 compression modes generated from three compression algorithms including DDPCM, HA, and general DDPCM. Furthermore, this novel algorithm supports one-plane and two-plane compression modes and manipulates break points more efficiently. For 8x8 tile size and 16-bit depth values, the proposed multi-mode algorithm can achieve 1.91:1 compression ratio on average and improve 76.9% and 39.4% compared with the HA and the DDPCM compression methods, respectively. The modified efficient depth buffer compression can achieve 10.56:1 and 7.76:1 compression ratio in one-plane mode and two-plane mode, respectively. The modified DDPCM can achieve 6.48:1 and 5.39:1 compression ratio in one-plane mode and two-plane mode, respectively.**

I. Introduction

Recently, the 3D computer graphics are widely used in many applications such as mobile phones, GPS (Global Positioning System), digital TV [1], games, biomedical applications [2], where these applications usually use complex GUI (graphical user interface) to generate better 3D images. It is manifest that 3D computer graphics system requires extremely high memory bandwidth to process. On the other hand, with the growth of complexity of 3D scenes, the amount of data computation increases substantially. Therefore, in the bandwidth-limited system, how to efficiently compress the depth buffer data to save bandwidth becomes a significant research issue. Fast z-clears compression algorithm [3] uses a dedicated flag to indicate whether a tile is cleared. DDPCM [4], Anchor encoding [5], efficient depth buffer compression [9] (i.e., HA method) are effective compression algorithms to exploit the coherence of interpolated depth values. The plane encoding scheme presented in [7] applies the concept of indexing to compress tiles. Depth offset compression [8] saves the differentials based on the z-max value (maximum depth value) and z-min value (minimum depth value) in a tile. Other compression methods are released in [10-12]. Since the compression performance of the above state-of-the-art algorithms are limited and cannot be adaptively compressed according to different scenes, we are motivated to proposed multi-mode depth buffer compression algorithm for 3D graphics systems.

This work was supported in part by the National Science Council (NSC) Grant NSC-96-2220-E-009-038, MOEA-96-EC-17-A-01-S1-048, and NSC-96-2221-E-009-220.

This paper is organized as follows. A brief review of depth buffer compression algorithms is described in Section II. In Section III, the proposed multi-mode algorithm has been presented. The simulation results are presented in Section IV. At last, the brief statements conclude the presentation.

II. Review of Depth Buffer Compression Algorithms

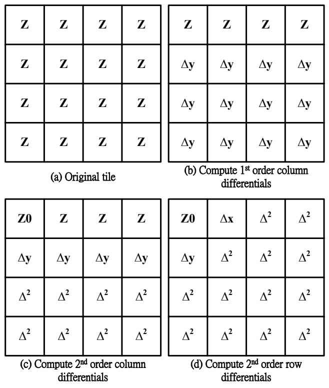
There are many state-of-the-art and existing compression algorithms [3-12], such as fast z-clears, DDPCM, Anchor encoding, efficient depth buffer compression, plane encoding, and depth offset compression. In this section, we give a review of DDPCM [4] and HA [9] compression methods that adopted in our proposed multi-mode system.

A. DDPCM Compression Method

DDPCM (Differential Differential Pulse Code Modulation) [4] is an off-the-shelf data compression algorithm. Since the depth values are linearly interpolated in screen space, DDPCM algorithm is very suitable for this kind of condition. DeRoo *et al.* [4] proposed a depth buffer compression algorithm as illustrated in Fig. 1 DDPCM can achieve 8:1 compression ratio on 8x8 tile size, using a 32-bit depth values and reading 256 bits from memory. DeRoo *et al.* also proposed an extended depth buffer compression algorithm, called two-plane mode, in order to handle specific cases that tiles can be separated into two planes. In 24-bit case, we have to save two 24-bit reference points, the upper left and lower right pixels in the tile, two 24-bit x differentials, two 24-bit y differentials, 2-bit predicted terms and 8-bit break point used to combine two planes based on different reference points.

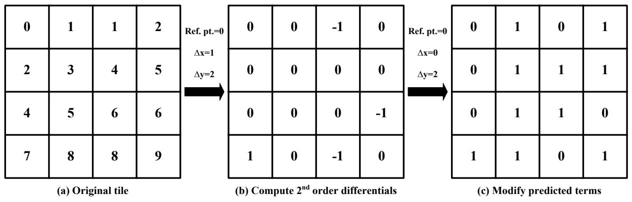
B. HA Compression Method

Hasselgren and Akenine-Möller proposed a new depth buffer compression algorithm, which can achieve high compression ratio by exploiting the coherence of interpolated depth values in screen space [9]. The operations of the one-plane mode are illustrated in Fig. 2. From the one-plane mode example, the predicted terms are saved in only 1 bit that is the reason why this algorithm can achieve better compression ratio than other compression algorithms. Additionally, this algorithm provides an adaptive scheme to handle two-plane mode cases rather a fixed-position-reference-point scheme of the extended DDPCM algorithm.



(a) Original tile
(b) Compute 1st order column differentials
(c) Compute 2nd order column differentials
(d) Compute 2nd order row differentials

Fig. 1. Illustration of DDPCM [4, 9].



Ref. pt.=0
Δx=1
Δy=2
Ref. pt.=0
Δx=0
Δy=2
(a) Original tile
(b) Compute 2nd order differentials
(c) Modify predicted terms

Fig. 2. Illustration of one-plane mode of HA compression [9].

III. Proposed Multi-mode Compression Algorithm

In this section, we propose a multi-mode algorithm for depth buffer compression. According to different scene changes, the proposed algorithm is capable of adaptively employing three compression algorithms including DDPCM, HA, and general DDPCM to generate 19 compression modes in Table 3. General DDPCM similar to DDPCM makes use of 7 bits to save the predicted terms. The architecture of our proposed algorithm as shown in Fig. 3 also supports one-plane mode and two-plane mode compression scheme. The proposed architecture depicted in Fig. 3 is demonstrated as follows. Notice that the uncompression mode has to be considered for the proposed compression scheme. In the first step, we compute the 2nd order differentials. In the second step, we calculate and check the range of these differentials. If any differential is larger than the maximum number or less than the minimum number that general DDPCM can serve, we will label this tile as an uncompressed tile. If the tile passes the check point, we will determine whether the tile is one-plane or two-plane. In case the tile belongs to the two-plane mode, we have to compute another set of differentials according to another reference point. After computing another set of differentials, we combine these two sets of differentials separated by break points. Besides, if the tile is a falling case, we have to compute another two sets of differentials based on lower-left pixel and upper-right pixel, denoted as R, as shown in Fig. 5. To combine two sets of differentials, the break-point map, composed of 0 and 1, must be prepared. In Fig. 5, break points of four supported cases are also illustrated. When we are combining, we will do the following operations for each row. In each row, we scan the first column to the eighth column. If the break point is 0 and the combination mode is not a falling case, the corresponding position will be placed the differentials based on upper-left pixel. If the combination mode is a falling case, the corresponding position will be placed the differentials based on lower-left pixel. On the other hand, if the break point is 1 and the combination mode is not a falling case, the corresponding position will be placed the differentials based on lower-right pixel. If the combination mode is a falling case, the corresponding position will be placed the differentials based on upper-right pixel. There is an exception that in each row scanning when we have scanned a break point with 1; the all break points of the remainder columns will be view as 1, no matter what the original values of these break points are. Fig. 7 shows a two-plane mode example, how to compute two sets of differentials according to different reference points and how to combine the two planes. Notice that for hardware-oriented design, we do not completely follow the steps presented by HA method [9]. The format of the op-code and break points are depicted in Fig. 6. The first bit of the op-code represents whether the tile is compressed. The second bit of the op-code indicates whether the tile is one-plane mode or two-plane mode. The third and the fourth bits represent what kind of compression algorithm is applied in horizontal direction. The fifth and the sixth bits represent what kind of compression algorithms is applied in vertical direction. The horizontal means the positions, Z2, Z3, Z5, Z6, Z7, Z9, Z10, Z11, Z13, Z14, and Z15, in Fig. 4. The vertical direction means the positions, Z8 and Z12, in Fig. 4.

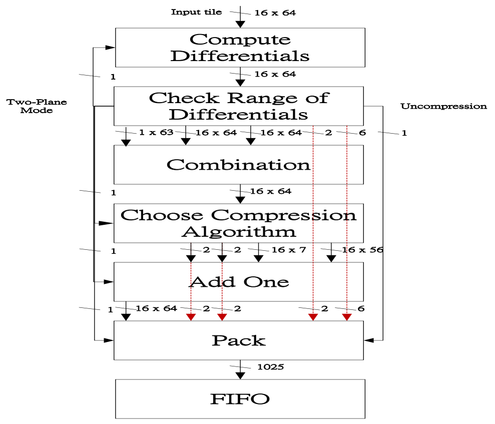
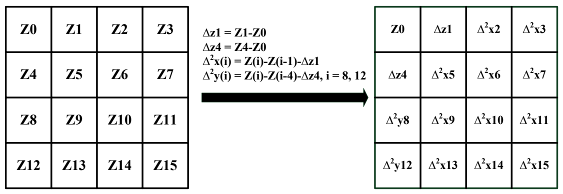


Fig. 3. Architecture of the proposed multi-mode depth buffer compression.

In break point, the first and the second bits indicates what kind of combination modes, such as rising case, is applied to the break points. The third, the fourth and the fifth bits mean the row number of the top break point. The sixth, the seventh and the eighth bits mean the column number of the top break point. Notice that the break points will be saved only when the tile is two-plane mode. Additionally, if the tile is uncompressed, only the first bit of the op-code will be packed with the tile.



Z0 Z1 Z2 Z3
Z4 Z5 Z6 Z7
Z8 Z9 Z10 Z11
Z12 Z13 Z14 Z15
Δz1 = Z1-Z0
Δz4 = Z4-Z0
$\Delta^2x(i)$ = Z(i)-Z(i-1)-Δz1
$\Delta^2y(i)$ = Z(i)-Z(i-4)-Δz4, i = 8, 12
Z0 Δz1 Δ^2x2 Δ^2x3
Δz4 Δ^2x5 Δ^2x6 Δ^2x7
Δ^2y8 Δ^2x9 Δ^2x10 Δ^2x11
Δ^2y12 Δ^2x13 Δ^2x14 Δ^2x15

Fig. 4. Example of one-plane mode.

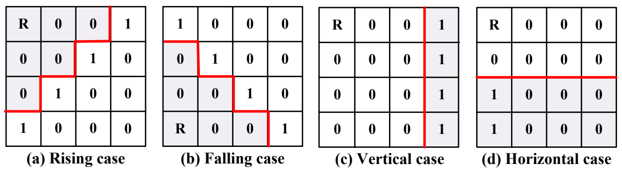


Fig. 5. Four cases supported by the proposed algorithm and corresponding break-point maps

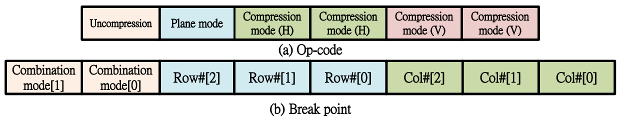
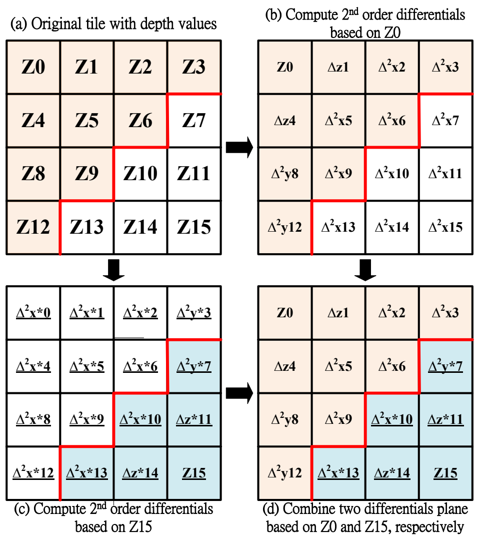


Figure 6. Op-code and break point.



(a) Original tile with depth values
(b) Compute 2nd order differentials based on Z0
(c) Compute 2nd order differentials based on Z15
(d) Combine two differentials plane based on Z0 and Z15, respectively

Fig. 7. Two-plane mode of the prposed multi-mode algorihtm.

IV. Comparision and Simulation Results

In this section, the comprehensive comparisons as listed in Table 1, 2, and 3 with HA and DDPCM in terms of average compression ratio, compression ratio in one-plane mode and two-plane mode, and number of bits of the proposed algorithm, respectively. The teapot benchmark is used as a reference simulation as shown in Fig. 8. The average compression ratio as listed in Table 1 shows that the proposed sachem outperforms others by 76.9% and 39.4% compared with DDPCM and HA methods. In this simulation, we can find that the average compression ratio of HA method is not better than either DDPCM or our proposed scheme. The reason is that HA compression scheme is suitable for the high precision interpolation. In other words, if the simulation environment is high precision interpolation, it turns out a much higher average compression ratio obtained by HA compression method. Table 2 shows the compression ratio distribution for one-plane mode and two-plane mode. Instead of 26 bits or 32 bits for saving break points in 8x8 tile size applied by efficient depth buffer compression, our proposed scheme only needs 8 bits. Therefore, the compression ratio of the proposed scheme is higher than other existing algorithms. The 19 compression modes are illustrated and the total compressed bits of a tile are listed in Table 3. In HA-HA-one-plane mode as listed in Table 3, for example, when vertical direction and horizontal direction both are compressed by HA compression method, the total compressed tile size is 16+7+7+61+6=97 bits, including one reference point, one Δx, one Δy, 61-bit predicted terms, and 6-bit op-code. In HA-HA-two-plane mode, in the same conditions, the total compressed tile size is 16+16+7+7+7+7+58+6+8 =132 bits, including two reference points, two Δx's, two Δy's, 61-bit predicted terms, 6-bit op-code, and break point. Other title sizes using different mode scheme can be calculated in similar way. Notice that vertical and horizontal directions can be compressed by different compression methods. Concerning the general DDPCM method, we expect that the size of the compressed tile can be smaller than that of half size of original tile. The sample distribution of the average compression ratio as shown in Figs. 9 and 10 illustrates the usefulness of our proposed scheme compared with HA and DDCPM compression schemes, respectively. A point in Fig. 9 and Fig. 10 indicates an average compression ratio of five tiles. It is obvious that our proposed scheme can achieve more stable average compression ratio than HA and DDPCM methods.

Fig. 8. Benchmark scene.

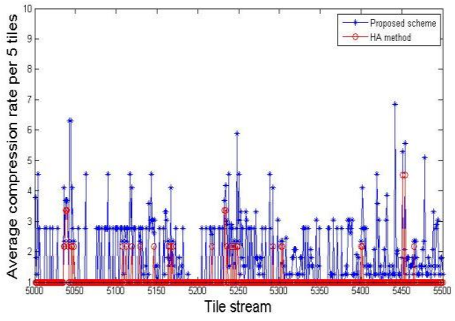


Fig. 9. Proposed scheme vs HA compression method.

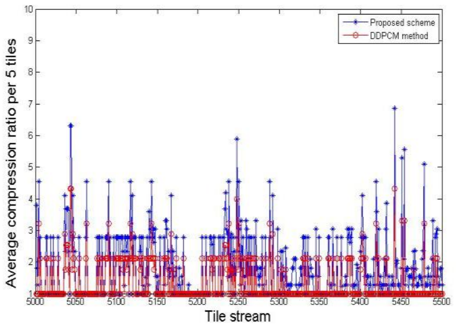


Fig. 10. Proposed scheme vs. DDPCM compression method.

TABLE 1. AVERAGE COMPRESSION RATIO WITH 8X8 TILE SIZE

	Average compression ratio
HA Method[9]	1.08
DDPCM [4]	1.37
Proposed scheme	1.91

TABLE 2. COMPRESSION RATIO WITH 8X8 TILE SIZE

	One-plane mode	**Two-plane mode**
HA Method [9]	10.89 : 1	6.87 : 1
DDPCM [4]	6.52 : 1	4.85 : 1
Proposed scheme – modified HA	10.56 : 1	7.76 : 1
Proposed scheme – modified DDPCM	6.48 : 1	5.39 : 1
Proposed scheme – general DDPCM	2.21 : 1	2.13 : 1

V. CONCLUSION

In this work, the multi-mode algorithm for depth buffer compression is presented. This proposed algorithm not only supports HA, DDPCM as well as general DDPCM algorithms, but also handles one-plane mode and two-plane mode compression. In addition, the algorithm saves the break points more efficiently for 8x8 tile size. From the experimental results, the proposed multi-mode scheme can provide more stable average compression ratio to achieve the quality guarantee performance.

TABLE 3. SIZE OF COMPRESSED TILE IN PORPOASED SCHEME

Vertical	**Horizontal**	**One-plane mode (bits)**	**Two-plane mode (bits)**
HA	HA	97	132
HA	DDPCM	152	184
HA	General DDPCM	427	444
DDPCM	HA	103	138
DDPCM	DDPCM	158	190
DDPCM	General DDPCM	433	450
General DDPCM	HA	133	168
General DDPCM	DDPCM	188	220
General DDPCM	General DDPCM	463	480
Uncompression		1025	1025

REFERENCES

[1] TS 102 812, "DVB Multimedia Home Platform (MHP) Specification 1.1", Nov. 2001.

[2] T. Heinonen, A. Lahtinen and V. Hakkinen, "Implementation of three-dimensional EEG brain mapping," Computers and Biomedical Research 32, pp. 123–131, 1999.

[3] S. Morein., "Method and apparatus for efficient clearing of memory," in US Patent 6,421,764, 2002.

[4] J. DeRoo, S. Morein, B. Favela, M. Wright, "Method and apparatus for compressing parameter values for pixels in a display frame," in US Patent 6,476,811, 2002.

[5] J. Van Dyke, J. Margeson, "Method and apparatus for managing and accessing depth data in a computer graphics system," in US Patent 6,961,057, 2005.

[6] T. Van Hook, "Method and Apparatus for Compression and Decompression of Z Data," in US Patent 6,630,933, 2003.

[7] B.-S. Liang, Y.-C. Lee, W.-C. Yeh, and C.-W. Jen, "Index rendering: hardware-efficient architecture for 3-D graphics in multimedia system," in *IEEE Transactions on Multimedia*, vol. 4, no. 2, pp. 343-360, June 2002

[8] S. Morein, M. Natale, "System, method, and apparatus for compression of video data using offset values," in US Patent 6,762,758, 2004.

[9] J. Hasselgren, T. Akenine-Möller, "Efficient depth buffer compression," *Graphics Hardware*, pp. 102-110, 2006.

[10] S. Morein, "ATI Radeon HyperZ technology," *in Hot3D Proc. ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, Aug. 2000.

[11] C.-H. Chen and C.-Y. Lee, "Two-level hierarchical Z-buffer with compression technique for 3D graphics hardware," The Visual Computer, Springer, vol. 19, no. 7-8, pp. 467-479, Dec. 2003.

[12] C.-H. Yu and L.-S. Kim, "A hierarchical depth buffer for minimizing memory bandwidth in 3D rendering engine: depth filter," ISCAS '03, vol.2, pp.II-724- II-727, May 2003