The Visual Computer

A JPEG-like texture compression with adaptive quantization for 3D graphics application

C.-H. Chen, C.-Y. Lee

Department of Electronics Engineering, National Chiao Tung University, 1001, Ta Hsueh Road, Hsinchu, 300, Taiwan, R.O.C.
E-mail: chchen@royals.ee.nctu.edu.tw

Published online: 2 October 2001

DCT-based compression is widely used in video and image compression for a high compression ratio, but it suffers from random-access problem when applied to texture compression. In this paper we present a JPEG-Like DCT-based texture compression technique which is suitable for 3D graphics rendering system. We apply a simple adaptive quantization on an 8×8 block-size texture, such that the length of the encoded bit stream of one block can be approximated to the target. A pre-defined quantizer scale is encoded with the bit stream with a small overhead. Our technique achieved a high compression ratio, quality control of true color texture, and random access of texture data.

Key words: 3D graphics – Texture compression

Correspondence to: C.-H. Chen

1 Introduction

3D computer graphics have been widely used in various applications, for example, entertainment, medicine, and scientific research. There are few applications that stress modern PC computing power more than today's 3D-API. The continuous improvement of deep sub-micron semiconductor technology has brought 3D graphics processors from high-end workstations to low-end desktop PCs. These powerful processors are now capable of supporting many 3D applications. Although current 3D graphics processors have high computing power, they still cannot render lifelike animated images in real-time. The bottleneck of 3D graphics systems includes computing power, memory storage and bandwidth requirement. Although the computing power of graphics processors doubles every six months, the memory storage and bandwidth requirement become worse due to the demand for realistic and complex 3D applications. Thus the overall system performance is limited.

Among the operations in 3D graphics rendering systems, texture mapping consumes the largest bandwidth and memory storage. The bandwidth requirement for texture access may be up to giga-bytes per second, as shown in Table 1. Traditional texture mapping is used to increase the detail of the geometry visual quality without increasing the model complexity. Now multi-texture mapping techniques are widely used in lots of applications, for example, specular light-map, environment mapping, shadow and bump mapping. Texture filtering, which includes bilinear, trilinear and anisotropic filtering, is applied to reduce texture aliasing. With multi-texture mapping and filtering, the bandwidth requirement for texture access may jump up to 2-, 4- or 8-fold the single texture mapping (PowerVR^{TM} 1998).

Texture compression (TC) is the technique used to reduce the texture memory storage and bandwidth requirement. It increases system performance in several ways. First, it reduces accelerated graphics port (AGP) and frame buffer bus traffic, yielding superior texturing performance and fill rate. Second, it reduces the texture storage in frame buffer; this makes more detailed texture storage or performance possible by using the additional memory space for double or triple buffering. TC is a "must have" feature in current 3D accelerators.

Textures are ordinary bit map images. Although good and optimized image compression techniques have been described in many papers during the past few decades, they are not suitable for TC be-

The Visual Computer (2002) 18:29–40
Digital Object Identifier (DOI) 10.1007/s003710100130

Table 1. Bandwidth estimation of texture access (MB/s). Assume the following: 25 pixels/polygon, 24-bit RGB texture, single texture mapping, 10% texture cache miss rate

Texture filtering type	5 M polygons/s		10 M polygons/s		20 M polygons/s	
	Without cache	With texture cache	Without cache	With texture cache	Without cache	With texture cache
Point sampling	375	37.5	750	75	1500	150
Linear mip-map	750	75	1500	150	3000	300
Bilinear mip-map	1500	150	3000	300	6000	600
Trilinear mip-map	3000	300	6000	600	12 000	1200
Anisotropic	6000	600	12 000	1200	24 000	2400

cause of several issues with regard to texture access. In Sect. 2, we review previous work and discuss these issues considering modern technology and applications. In addition, we find a balance between texture quality and compression ratio (CR) for different applications. For example, medical and scientific 3D images must have lossless compression, 3D games can choose lossy compression and internet 3D applications can choose the highly lossy mode to shorten download time. Current results (3dfx 1999; S3^{TM} 1999) for the CR are within the range of 4 to 8. This CR may be fine for current API, but it is not sufficient for future 3D applications. Future TC should provide high CR and quality control. TREC (Torborg et al. 1996) is the first to use JPEG-like high CR and a simple quality control technique. But TREC suffers from a long texture access problem (Microsoft 1999), which may degrade performance. In this paper, we develop a JPEG-like TC technique which aims for high CR (> 8). We solve the problem of random texture access where quality control is also considered. A comparison of quality and CR with other methods is also shown.

2 Issues of texture compression and previous work

Image and video compression technologies have been extant for a long time. Still image coding schemes have been classified into predictive, block transform, and multi-resolution approaches (Egger et al. 1999). VQ, DCT and DWT are the most widely accepted techniques among these image compression schemes. Although textures are ordinary bitmap images, current image compression methods may not be applicable to TC. The nature of texture access make TC differ from traditional image compression. We discuss the issues of TC below.

2.1 Issues of texture compression

2.1.1 Random access

In a standard rasterization pipeline, the texture coordinate is generated until the polygon scan-conversion stage. The texture coordinate is evaluated or interpolated in texture space. Perspective correction and anisotropic and filtering texture mapping techniques result in more detail and correct texture mapping. The mapping procedure introduces discontinuous texture access in texture space and sometimes may cause large displacement. Thus the TC technique must provide fast random access of texture.

2.1.2 Compression ratio and visual quality

There is a tradeoff between CR and quality. Lossless compression, such as Lempel–Ziv compression, can provide perfect reconstruction, but the CR is low. However, lossy compression introduces error. For TC, existing techniques (S3^{TM} 1999; 3dfx 1999; Ivanov and Kuzmin 2000) have average CR values of between 4 and 8. Current image compression techniques can easily achieve CR values higher than 8 and also provide some features for various applications, such as: quality control, SNR scaleable, multi-resolution and progressive coding and streaming. The major difference between images and textures is that images are viewed on their own, while texture mapping rotates, distorts,

magnifies or minifies texture to "paste" on geometry surface. The visual quality is the most important for image compression, but it is not so critical for TC.

2.1.3 Decoding speed

As shown in Table 1, the texture access rate may be up to a giga-texel per second. The texture decompression engine must provide high-speed decompression and texture access. All current designs include a small on-chip texture cache to reap the benefit of texture reuse. Hakura and Gupta (1997) showed that the cache miss rate is under 10% with proper cache design. Issue texture access or pre-fetch as early as possible can solve the problem of long texture decompression latency (Igehy et al 1998; Microsoft 1999). With the texture cache and a pre-fetch technique, the decoding speed may not be so critical, except in the worst-case scenario of texture cache miss. For example: currently, the fastest 3D GPU can achieve a rate of 3.2 G texel/s; assume a 10% texture cache miss rate; thus, the decoding speed is at least 320 M texel/s.

2.2 Previous work

Several TC techniques have been proposed and some are used in current products. These techniques can be classified into several categories:

2.2.1 Block truncation coding

Block truncation coding (BTC) (Knittel et al. 1996) is a simple TC technique. Two colors are selected to be the reconstruction level of a 4×4 block. Although BTC is very simple to encode and decode, its drawbacks are edge discontinuity, blocking effect and low CR. Kugler (1997) provides post-processing filtering to improve the quality. But this introduces more texture access of neighboring blocks and makes decompression more complex. Despite its CR being high ($= 12$), BTC is not well suited for TC due to poor quality.

2.2.2 Vector quantization

Vector quantization (VQ) was chosen for TC due to the simple decoding involved. However, VQ suffers from the problem of codebook selection. Codebook size determines the image quality and CR. Beers (1996) and PowerVR^{TM} (Butler et al. 1999) chose one codebook for the entire texture. When decompressing, the decoder first retrieves the index of the corresponding block and then consults the codebook to find the final colors. This requires two serial memory accesses of one block unless the codebook can be stored on the chip or a universal codebook can be hard-wired on the chip. A high-quality codebook will consume a large amount of chip area and require long downloading latency. It is impossible to find a universal codebook whose size is small and has good quality. Thus the traditional VQ technique is not suitable for high-performance texture systems.

2.2.3 Local palette

S3TC (S3^{TM} 1999) and 3dfx-FXT1 (3dfx 1999) are local palette techniques. They decompose texture into small 4×4 (S3^{TM} 1999; 3dfx 1999) or 8×8 (3dfx 1999) blocks. Within each block, two (S3^{TM} 1999; 3dfx 1999), three (3dfx 1999) or four (3dfx 1999) colors are encoded and linearly interpolated to four or eight colors. The drawback of this method is the lack of colors in each block. Thus Ivanov and Kuzmin (2000) proposed a "color distribution" technique to provide more colors on one block. It allows using colors from neighboring blocks instead of simply interpolating colors stored in the current block. The visual quality may become superior to that in S3TC and FXT1. However, all of the local palette techniques have a CR of 4 to 6 for RGB format. Current local palette techniques do not provide 24-bit RGB true color format, and when the alpha component is considered, the CR becomes reduced. They cannot achieve a high CR of true color texture.

2.2.4 DCT-based coding

Texture and rendering engine compression (TREC) (Microsoft 1997) is a JPEG-like compression technique, while Playstation2 (Suzuoki et al. 1999) is an MPEG2-based technique. Both of them have DCT-based coding. A CR higher than 15 can easily be achieved, and good quality can also be ensured under DCT-based coding. The challenges of DCT-based coding for TC are random access and decoding speed. Variable-length bit stream makes it unsuitable for TC. Besides, it must decode in order from the head of the bit stream to find the desired pixel. TREC solved this problem by preserving DC without DPCM. Talisman constructed an index table and link-list to address the variable-length bit

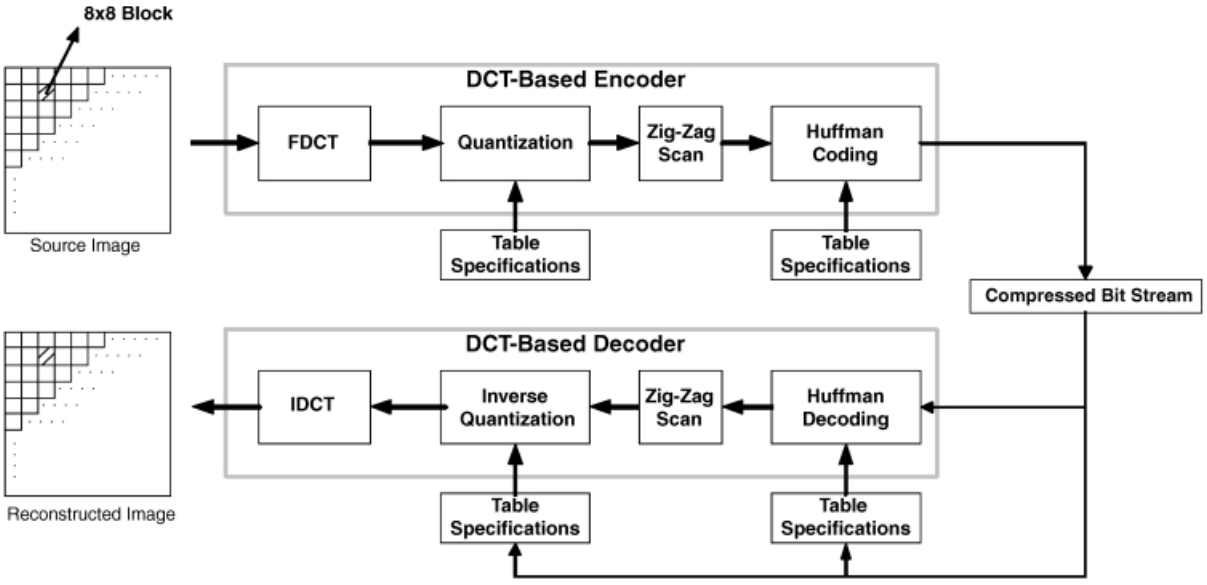


Fig. 1. JPEG encoding/decoding flow

stream of each block. However, the long latency (Microsoft 1999) of texture access is still a bottleneck and makes the hardware design more complex. In this paper, our task is to solve the problems of DCT-based TC.

3 The proposed JPEG-like texture compression technique

3.1 Brief review of JPEG

DCT-based compression techniques are based on the concept of compacting energy into fewer coefficients in the transform domain and then encoding these coefficients. JPEG (Pennebaker 1993) is a standard DCT-based still image compression technique, and it provides good visual quality. JPEG partitions the image into 8×8 blocks, applies to each block the forward discrete cosine transform (FDCT), and scalar quantizes the DCT coefficients. The DC coefficient is encoded by DPCM, while the zig-zag-ordered AC coefficients are entropy coded using Huffman coding (or variable length coding, VLC). Figure 1 shows the baseline JPEG encoding and decoding flow. In the encoding process, only the Huffman codewords and quantization tables are left for the user to control the visual quality and compression ratio. Standard JPEG has provided the example VLC codewords and quantization table based on perceptual criteria.

If we want to use JPEG for TC, we need to solve the random access problem. To achieve random access of an image block in the JPEG bit stream, we need to build an index table or truncate the bit stream. Truncating the bit stream will cause a serious block effect, since the high-frequency component cannot just be discarded. The technique used to build the index table is similar to TREC. The entries in the index table represent the memory address or offset of each block. The size of the index table is described as follows:

$$S_{\text{index}} = TotalBlocks \times \log_2 \left(\frac{TotalBlocks \times (R+G+B+A)}{CR} \right), \quad (1)$$

where $TotalBlocks = (Width/8) \cdot (Height/8)$. For 512×512 24-bit RGB format images and $CR = 24$, $S_{\text{index}} = 73\,728$ bits. Adding the index table, the overall CR decreases to 18.73. If we partition the bit stream of each block into 32, 64, 128 or 256 segment sizes for the purpose of mem-

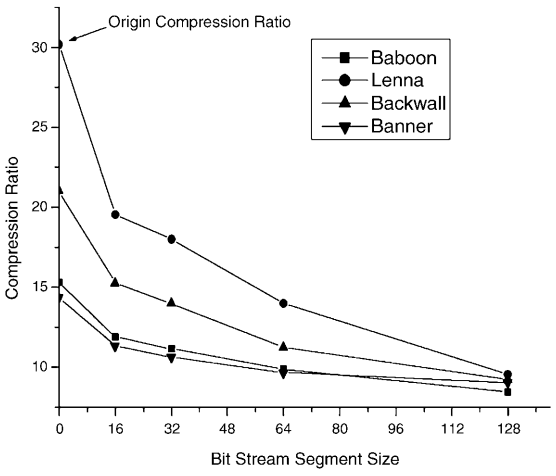


Fig. 2. Compression ratio decreases under different bit stream segment sizes in JPEG

ory alignment, the overhead of index table is as follows:

$$S_{\text{index}} = TotalBlocks \times \log_2 \left\{ \sum_{j=1}^{TotalBlocks} \left(\left\lceil \frac{\widetilde{B}_j}{Segment\,Size} \right\rceil \times Segment\,Size \right) \right\}, \quad (2)$$

where $\widetilde{B}_j$ is the encoded bit-stream length. $\lceil x \rceil$ represents the minimum integer greater than or equal to x. Note that if we down sample U and V an extra index table should be built for the U and V streams to distinguish them from the Y stream. The overall CR is reduced by adding an index table. Figure 2 shows the decrease in CR for different segment sizes. It indicates that the overhead of the table is high especially for a low-complexity image: Lenna and Backwall as shown in Table 3. Using an index table will introduce two memory accesses to obtain the desired bit-stream and degrade the overall performance.

3.2 The proposed algorithm

Figure 3 shows the compression and decompression flow of our JPEG-like TC algorithm for an RGBA, 32-bit texture. The major distinctions of our algorithm from JPEG are:

- It operates on one 8×8 block individually. Each 8×8 Y or U or V or Alpha block is encoded, decoded and accessed independently.
- It compresses every block toward a fixed length bit stream, thus random access can be achieved and memory space can be utilized efficiently. The bit stream length of every encoded block can be any size, as controlled by user definition or quality control.
- Every 8×8 block has its corresponding quantizer scale, which is pre-defined to control quality and CR.
- The "Alpha" component, which represents the transparency of every texel, is compressed in addition to the R, G and B color components.

Both DCT and VLC are the same as those for JPEG; however, we change the quantization step in our algorithm. In the JPEG quantization stage, Hung and Meng (1991) found the optimized quantizer of JPEG using a complex search. But if the same quantizer is applied to every image block as in JPEG, it will lack local adaptation and also result in a variable-length bit stream. In our algorithm, we introduce a localized quantizer, an adaptive quantization for every block. One of the pre-defined quantizer scale or quantization tables is chosen to quantize each 8×8 block to the desired bit stream length.

In JPEG or MPEG, the quantizer scale can be specified to control the rate of the encoded bit stream. Rate control is introduced to provide consistent visual quality in video transmission. The problem found is, given a target bit rate, how to encode the video sequence for minimized distortion and consistent visual quality. In JPEG and MPEG, the quantization table or quantizer scale is applied to all blocks. We can describe the quantization for each 8×8 block as follows:

$$\widehat{C}_n = Round\left(\frac{C_n}{Q_n \cdot Q_{\text{p}}}\right) \quad n = 1, 2, \ldots, 64, \quad (3)$$

where C_n represents the DCT coefficients in one block, Q_n is the corresponding quantization step and Q_{p} is the quantizer scale, whose default is 1. $Round(x)$ represents the closest integer to x.

Although an optimized quantizer can be found to achieve minimum distortion of one block, the decoder has to know the quantizer of each block. As a consequence the CR will strongly decrease. Thus

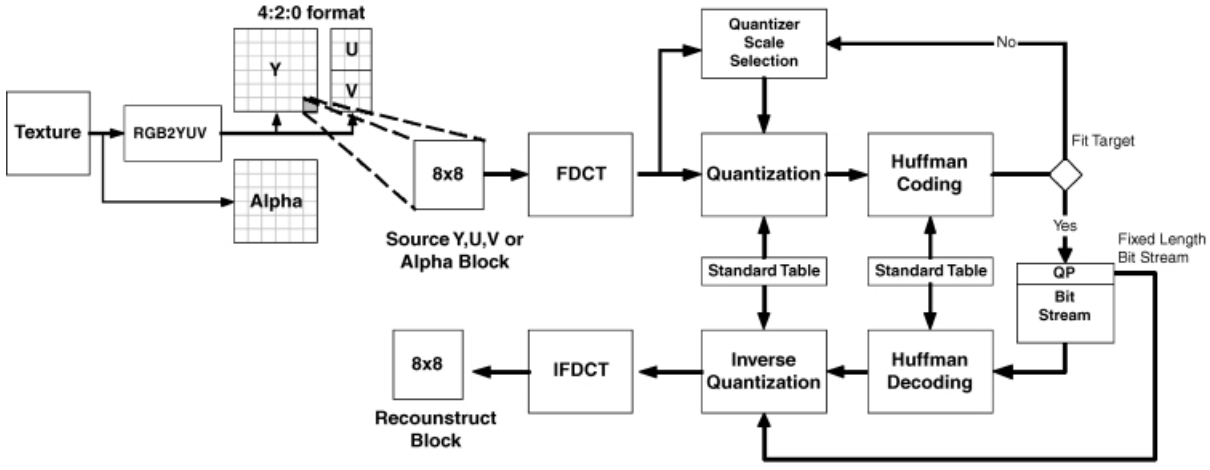


Fig. 3. Our JPEG-Like texture compression/decompression flow

we have to define some quantizer scale Q_p for TC and encode 4 bit data into each corresponding block to indicate which quantizer scale or table is selected.

3.3 Adaptive quantization for 8×8 blocks

Considering hardware implementation, we define sixteen Q_p for the quantizer scale. Thus a 4-bit index is encoded in each 8×8 block to indicate which Q_p has been selected. Therefore (3) can be written as follows:

$$\widehat{C}_n = Round\left(\frac{16 \cdot C_n}{Q_n \cdot Q_\mathrm{p}}\right)$$
$$Q_\mathrm{p} = 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 40, 48. \quad (4)$$

The range and resolution of Q_p can be increased or decreased to control the rate more precisely or coarsely. Assume the desired compression ratio is CR. The allowed bit count for one 8×8 block j is $B = 64 \cdot 24/CR$ for RGB true color texture. The bit count after Huffman coding is $\widetilde{B}_j$. The encoding target is to let $\widetilde{B}_j$ come as close as possible to (but remain less than) B. The selection of Q_p could be made by searching all of the Q_p. However, we introduce the activity-based bit estimation model (Cheng and Hang 1995) used in the rate control of video sequences to increase the encoding speed.

The bit count $\widetilde{B}_j$ depends on the complexity in the block. For example, a texture block with higher activity indicates that the content of the block is more complex than one with low activity and that more bits will be needed to encode it. Activity here means an absolute-value summation of the 63 DCT AC coefficients in one 8×8 block. Therefore we can construct a linear equation of block activity and bit counts for different Q_p. The following is an empirical first-order bits model:

$$\widetilde{B}_j = K_1 \frac{act_j}{Q_{\mathrm{p}j}} + K_2, \quad (5)$$

where K_1 and K_2 are constants derived from training data to minimize the error and act_j is the activity of block j. By using this model, we can train lots of various blocks under different Q_p and then obtain 16 approximation lines. When encoding block j, we calculate act_j first and then solve (6):

$$Q_{\mathrm{p}j} = K_1 \frac{act_j}{B - K_2}. \quad (6)$$

Thus we obtain the approximate $Q_{\mathrm{p}j}$. Figure 4 illustrates the concept of $Q_{\mathrm{p}j}$ selection. By controlling $Q_{\mathrm{p}j}$, $\widetilde{B}_j$ can be specified to be as close as possible to B. Since the content of neighboring blocks usually has little variation, the previous blocks $Q_{\mathrm{p}j}$ can be used as a reference for $Q_{\mathrm{p}j+1}$.

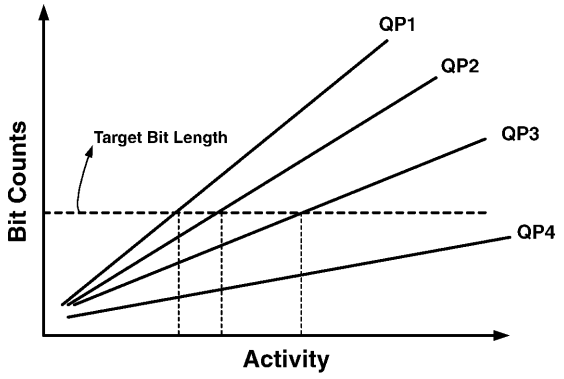


Fig. 4. Q_{p} selection under different activities

The pseudo-code to encode one 8×8 block is as follows:

1. Calculate activity act_j of block j.
2. Solve (6) to obtain the value of Q_{p} closest to or equal to $Q_{\mathrm{p}_{j-1}}$.
3. Quantize j with Q_{p} and Huffman coding to obtain the bit count $\widetilde{B}_j$; if $\widetilde{B}_j > B$ go to step 4, else if $\widetilde{B}_j < B$ go to step 5, else go to last step 6.
4. If $\widetilde{B}_j > B$, decrease the bit count:
   while $(\widetilde{B}_j > B)$ {
   increase Q_{p} one level to Q'_{p};
   quantize with Q'_{p};
   Huffman encoding to obtain new $\widetilde{B}_j$; };
   $Q_{\mathrm{p}_j} = Q'_{\mathrm{p}}$.
5. If $\widetilde{B}_j < B$, increase the bit count but do not allow it to exceed the budget B:
   while $(\widetilde{B}_j < B)$ {
   decrease Q_{p} one level to Q'_{p};
   quantize with Q'_{p};
   Huffman encoding to obtain new $\widetilde{B}_j$; };
   $Q_{\mathrm{p}_j} = Q_{\mathrm{p}}$.
6. Encode corresponding index of Q_{p_j} into block.

As shown above, complex blocks are assigned large Q_{p}, and low-activity blocks smaller ones. Although higher Q_{p} means higher loss and lower Q_{p} means smaller loss, all of the blocks are encoded towards the target length and stored in a fixed-size memory block with efficient memory utilization. Thus the texture in any block in the memory can be accessed. This fixed-length technique also provides efficient memory allocation and consistent decoding speed.

3.4 Bit budgets

The texture formats in 3D graphics include gray-scale, 16-bit RGB, 24-bit RGB, 32-bit RGBA and others. How to choose the bit target for each component in each block is another problem in our adaptive quantization technique. Here, we profile the bit allocation of each component in JPEG to see the bit counts of each component. When arranging the bit budget, the memory alignment problem should also be considered. Bit counts of 32, 64, 128 or 256 are suitable for memory access and hardware design. Other size may cause two or three memory accesses in one texture request and decrease the performance of texture mapping.

We down-sample U and V by two in our algorithm. This refers to the YUV 4 : 2 : 0 format used in JPEG and MPEG. Figure 5 is the bit allocation profile of three 24-bit true color JPEG images, where the quantizer and Huffman coding are set to default. Obviously, we can find that almost all of the U and V allocated bit counts are within 32 bits in one block. But Y may be distributed widely from 16 to 160 bits. Thus we can conclude that a 32-bit budget is sufficient for U and V for most cases in TC. For Y, it depends on the complexity of the texture. A 64-, 96-, 128- or 256-bit budget can be chosen for Y.

In addition to the RGB format, the texture format may include the "Alpha" component, which is used to represent the transparency of the texture. The Alpha component can be treated as a gray-level image; thus we encode it individually. The Alpha resolution may be 4, 5, 6 or 8 bit, depending on the application. When the resolution is 5 or 6 bit, we can combine the bit stream of Alpha with Y for a total bit length of 128 or 256, which is a good memory alignment size. If an 8-bit Alpha is used, the bit stream is separated from Y. Table 2 shows the performance of different resolution Alpha under various CR in our algorithm. Here, we treat the gray-level image as the Alpha component, since the detail of Alpha can be any shape.

3.5 Compression ratio and visual quality comparison

As discussed in Sect. 3.4, we can use the adaptive quantization in Sect. 3.3 for each block to efficiently use the bit budget. Table 3 shows the CR and a visual-quality comparison with other techniques. The CR,

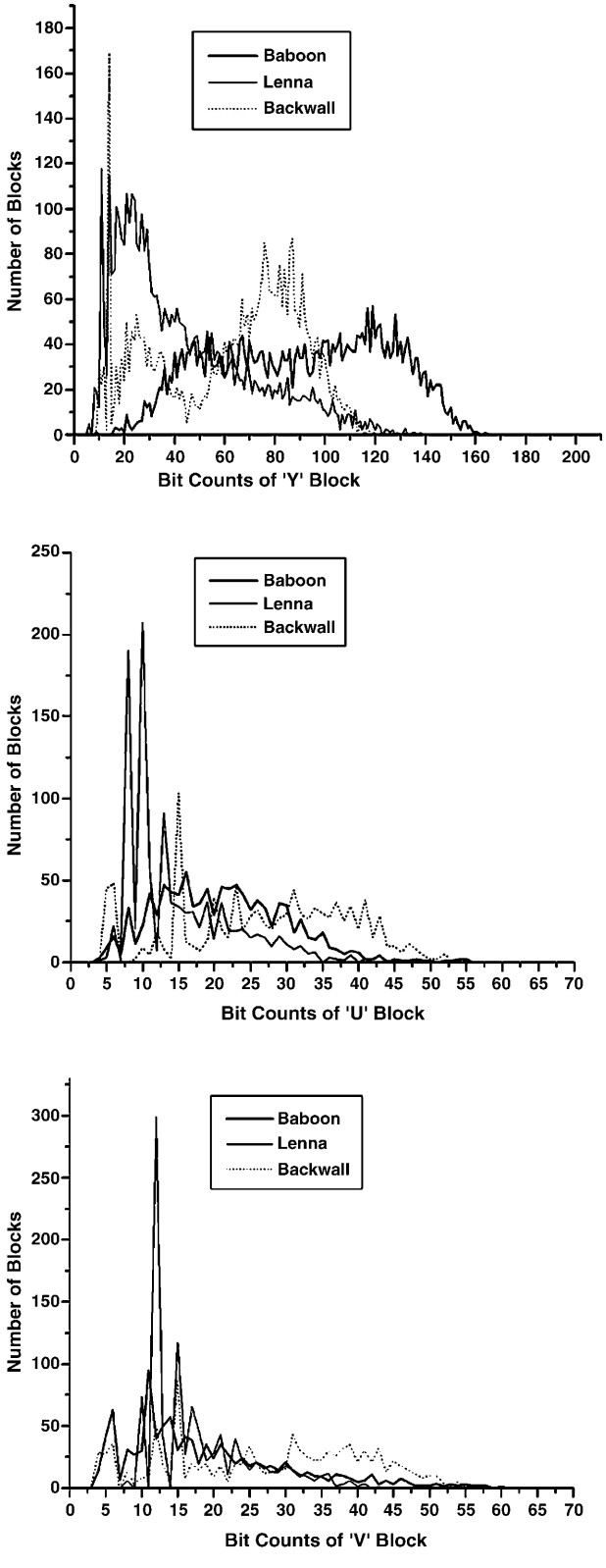


Fig. 5. Bit allocation profile of *YUV* components in JPEG

Table 2. PSNR performance of different resolution Alpha; CR is 8

	Baboon	Lenna	Backwall	Banner	Texture
4 bits/pixel	45.84	51.05	49.68	46.92	49.57
5 bits/pixel	41.12	47.80	46.93	42.78	45.71
6 bits/pixel	36.01	44.01	43.08	38.11	41.07
8 bits/pixel	24.76	34.04	33.58	27.57	31.61

when U and V are down-sampled by 2, is defined as follows:

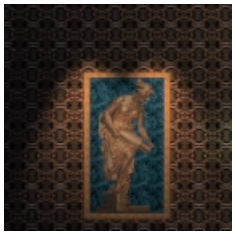
$$CR = \frac{(R+G+B+A)\times 64}{Bit_Y + \left(\frac{Bit_U + Bit_V}{4}\right) + Bit_A}, \tag{7}$$

Where R, G, B, A represent the format of the texture and $Bit_Y, Bit_U, Bit_V, Bit_A$ represent the bit counts of one block. For a 24-bit RGB format, $R = G = B = 8$ when $Y = 64$, $U = V = 32$ and $CR = 19.2$. For $U = V = 32$ and $Y = 128$ or 64, sizes suitable for memory alignment, $CR = 10.67$ and 19.2. As shown in Table 3, our TC technique can achieve a CR about 2-fold greater than the S3TC local palette technique and still hold the visual quality of JPEG. The random access problem of TREC is also solved. When the Alpha precision is low, for example 4 or 5 bit, combining the Alpha bit stream with Y to 128 bits results in good quality, and CR becomes 14.2, which is more than three times that in S3TC. For a low-complexity image (Lenna and Backwall), the quality is still good at $CR = 19.2$. The rendering results in Figs. 6 and 7 show the overall visual quality is good even under high CR. It is hard to find the difference between compressed and uncompressed texture-mapped 3D scenes. A feature comparison with other TC techniques is given in Table 4. DCT-based coding is good for high CR and good quality, while others cannot preserve good quality under high CR. With a fixed length bit stream technique, our technique is better than TREC, where both a high CR and good quality can be achieved.

3.6 Decoding speed

The decoding speed is an important issue in TC. Previous works (S3^{TM} 1999; 3dfx 1999; Ivanov and Kuzmin 2000) challenge the decoding speed and hardware cost of JPEG. But from the a system view, it is not critical in current 3D graphics systems. Current 3D graphics accelerators all include

Table 3. Compression ratio and visual quality (average RGB PSNR) comparison. The number in the parenthesis is CR

Baboon	Lenna	Backwall[a]
		
Banner[a]	**Texture**	**Venus[a]**
		

	FXT1 $CR=6$	S3TC $CR=6$	JPEG[b]	TREC[c]	Proposition 1, $CR=8$	Proposition 2, $CR=10.67$	Proposition 3, $CR=12$	Proposition 4, $CR=19.2$
Baboon	28.30	27.82	24.94 (15.34)	26.20 (12.16)	25.92	25.03	24.64	23.08
Lenna	34.91	35.03	31.71 (30.27)	31.32 (36.03)	33.16	32.89	32.72	31.51
Backwall	33.14	32.6	29.44 (21.10)	28.87 (24.46)	29.71	29.56	29.45	28.47
Banner	30.26	30.01	25.88 (14.39)	26.89 (12.08)	26.80	26.10	25.81	24.31
Texture	37.97	36.51	44.78 (21.94)	34.93 (24.5)	43.65	42.49	41.44	31.85
Venus	33.44	33.05	29.90 (19.17)	30.36 (17.49)	32.10	31.35	30.90	28.83

[a] Textures from 3D Winbench™ 2000, 3D Winbench™ is a trademark of ZD Inc.
[b] JPEG [13]: UV down-sample, and use default quantizer scale and default Huffman table.
[c] TREC [8]: UV down-sample, and use uniform quantization mode.

the MPEG video decoder. The transistor count of current 3D accelerators may be up to twenty million, and the MPEG video decoder is under 5% of the total chip area. Our JPEG-like TC algorithm can share hardware resources with the MPEG decoder, for example, the IDCT, inverse quantizer and VLD. High throughput IDCT and VLD techniques are available to support high-speed decompression, for example, 600 Mpixels/s ISDCT (Lin and Lee 2000) and group-based VLD (Shieh et al. 2001). Even under sequential decoding, this decoding speed is sufficient to support DCT-based texture decompression.

4 Conclusion

In this paper, we review the texture compression techniques of previous works and address the requirements of texture compression. The compression ratio of current standard S3TC and FXT1 is under $8:1$. A high-compression-ratio technique such as TREC suffers from the texture random access problem. By analyzing the properties of JPEG, we have proposed a JPEG-like DCT-based texture compression technique to obtain higher compression ratios (>12). We have solved the random access problem in TREC. In the proposed technique,

6

7

Fig. 6. 3D rendering result 1 (CR = 16 RGB format)
Fig. 7. 3D rendering result 2 (CR = 16 RGB format)

Table 4. Features comparison of texture compression techniques

	Kugler [11]	Beers [1]	PowerVR [5]	S3TC [2]	FXT1 [3]	TREC [8]	Proposed
TC type	BTC	VQ	VQ	Local palette	Local palette	DCT	DCT
Encoding speed	Fast	Slow	Slow	Medium	Medium	Fast	Fast
Decoding complexity	Low	Low	Low	Medium	Medium	Highest	High
CR	12	> 8	$2 - 16$	$4 \sim 6$	$4 \sim 8$	> 10	> 10
Quality	Bad	Medium	Medium	Very good	Very good	Good	Good

we apply localized block adaptive quantization to each 8×8 block and encode it to target bit stream length. A 4-bit quantizer-scale is encoded with the bit stream in each block. And a bit-count estimation model is proposed to increase the encoding speed for real-time texture encoding. The quality of our technique is at an acceptable level and the compression ratio can be defined by the user. Thus quality control, which is used in video applications, can also be achieved. Applications can choose the compression ratio versus quality trade-off depending on the hardware computing power or network bandwidth. This is important for internet and wireless 3D applications.

Bandwidth and storage are always the issues in various applications. Texture compression is also an issue. High compression ratio and quality control are the trends for future complex 3D applications. Our technique provides both features and brings a video technique into graphics applications. It provides an alternative approach to meet the ever-increasing memory storage and bandwidth requirements.

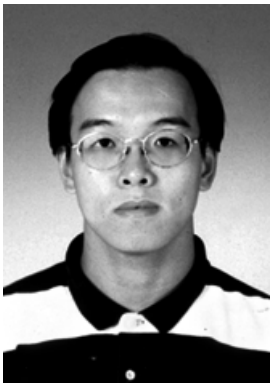
Acknowledgements. The authors would like to thank their colleagues within the SI2 group of NCTU and the Multimedia Division of SiS (Silicon Integrated Systems Corp.) for many fruitful discussions. The support from SiS and the NSC is also acknowledged. This work was supported by the National Science Council of Taiwan, ROC, under Grant NSC89-2218-E-009-080.

References

1. 3dfx Interactive (1999) FXT1 texture compression technology white paper. Available at http://www-dev.3dfx.com/fxt1/fxt1whitepaper.pdf
2. Hung AC, Meng TH-Y (1991) Optimal quantizer step sizes for transform coders. In: Chan YT, Venetsanopoulos AN (eds) IEEE Int. Conf. Acoustics Speech Signal Process., Toronto, Canada, IEEE Press, New York, pp 2621–2624
3. Beers AC, Agrawala M, Chaddha N (1996) Rendering from compressed textures. Proc. SIGGRAPH '96, pp 373–378
4. Kugler A (1997) High performance texture decompression hardware. Visual Comput 13(2):51–63
5. Shieh B-J, Lee Y-S, Lee C-Y (2001) A new approach of group-based VLC codec system with fully table programmability. IEEE Trans Circuits Syst. Video Technol. 11(2):210–221
6. Ivanov D, Kuzmin Ye (2000) Color distribution – a new approach to texture compression. Comput Graph Forum 19(3):283–289
7. Knittel G, Schilling A, Kugler A, Straber W (1996) Hardware for superior texture performance. Comput Graph 20(4):475–481
8. Igehy H, Eldridge M, Proudfoot K (1998) Prefetching in a texture cache architecture. Proc. 1998 EUROGRAPHICS/ SIGGRAPH Workshop Gr. Hardware, pp 133–142
9. Cheng J-B, Hang H-M (1995) Adaptive piecewise linear bits estimation model for MPEG based video coding. In: Liu B, Chellappa R (eds) Int Conf Image Process, Washington DC, IEEE Computer Society Press, Los Alamitos, 2:551–554
10. Torborg J, Kajiya JT (1996) Talisman: commodity realtime 3D graphics for the PC. Proc. SIGGRAPH '96, pp 353–363
11. Ribas-Corbera J, Lei S (1999) Rate control in DCT video coding for low-delay communications. IEEE Trans Circuits Syst Video Technol 9(1):172–185
12. Ramchandran K, Vetterli M (1994) Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility. IEEE Trans Image Process 3(5):700–704
13. Crouse M, Ramchandran K (1997) Joint thresholding and quantizer selection for transform image coding: entropy-constrained analysis and applications to baseline JPEG. IEEE Trans Image Process 6(2):285–297
14. Butler M, Pinter-Krainer M, VideoLogic Ltd (1999) PowerVR^{TM} second generation white paper of vector quantization texture compression, hardware bump mapping and generalized modifier volumes. Available at http://www.powervr.com
15. Suzuoki M, Kutaragi K, Hiroi T, Magoshi H, Okamoto S, Oka M, Ohba A, Yamamoto Y, Furuhashi M, Tanak M, Yatak T, Okada T, Nagamatsu M, Urakawa Y, Funyu M, Kunimatsu A, Goto H, Hashimoto K, Ide N, Murakami H, Ohtagu (1999) A microprocessor with a 128-Bit CPU, ten floating-point MAC's, four floating-point dividers and an MPEG-2 decoder. IEEE J Solid State Circuit 34(11):1608–1618
16. Microsoft (1997) Escalante hardware overview. Talisman Graph Multimedia Syst, pp 89–106
17. Microsoft (1999) Method and system for accessing texture data in environments with high latency in a graphics rendering system. United States Patent, Patent Number: 5880737

18. Egger O, Fleury P, Ebrahimi T, Kunt M (1999) High-performance compression of visual information – a tutorial review – part I: still pictures. Proc. IEEE 87(6):974–1011
19. PowerVR^{TM} (1998) White paper: the future of 3D graphics technology. Available at http://www.powervr.com
20. S3^{TM} Inc (1999) White paper of S3TC. Available at http://www.s3.com/savage3d/s3tc.pdf
21. Lin S-T, Lee C-Y (2000) Analysis and design of a high-throughput two dimension inverse scan discrete cosine transform processor. Master's thesis, NCTU EE, Taiwan
22. Pennebaker WB (1993) JPEG still image data compression standard. Van Nostrand Reinhold, New York
23. Hakura ZS, Gupta A (1997) The design and analysis of a cache architecture for texture mapping. In: Pleszkum A, Mudge T (eds) Proc. 24th Int Symp Comput Archit, Denver, Colorado, ACM, New York, pp 108–120

CHEN-YI LEE received his B.S. degree from the National Chiao Tung University, Hsinchu, Taiwan, in 1982 and his M.S. and Ph.D. degrees from Katholieke University Leuven (KUL), Belgium, in 1986 and 1990, respectively, all in electrical engineering. From 1986 to 1990 he was with IMEC/VSDM, working in the area of architecture synthesis for DSP. In February 1991, he joined the faculty of the Electronics Engineering Department, National Chiao Tung University, where he is currently a Professor. His research interests mainly include VLSI algorithms and architectures for high-throughput DSP applications. He is also active in various aspects of high-speed networking, system-on-chip design technology, very low bit rate coding, and multimedia signal processing.



CHENG-HSIEN CHEN received his B.S. degree from the Department of Electronics Engineering, National Chiao Tung University, in 1997. Since September 1997, he has been working toward a Ph.D. degree in electronics engineering at the same university. His research interests include VLSI algorithms and architectures (include 3D graphics and video systems) and memory optimization for system-on-chip design.