TRLE – An Efficient Data Compression Scheme for Image Composition of Parallel Volume Rendering Systems

Chin-Feng Lin, Yeh-Ching Chung, and Don-Lin Yang
Department of Information Engineering
Feng Chia University, Taichung, Taiwan 407, R.O.C.
E-mail: {cflin, ychung, dlyang}@iecs.fcu.edu.tw

Abstract

In this paper, we present an efficient data compression scheme, the template run-length encoding (TRLE) scheme, for image composition of parallel volume rendering systems. Given an image with 2n×2n pixels, in the TRLE scheme, the image is treated as n×n blocks and each block has 2×2 pixels. Since a pixel can be a blank or non-blank pixel, there are 16 templates in a block. To compress an image, the TRLE scheme uses the templates to encode blocks row by row. Blocks in the same row are encoded as a TRLE_sequence. By packing all TRLE_sequences in a packet, the packet is the compressed partial image that can be sent/received among processors. To evaluate the performance of the TRLE scheme, we compare the proposed scheme with the BR, the RLE, and the BRLC schemes. Since a data compression scheme needs to cooperate with some data communication schemes, in the implementation, the binary-swap (BS), the parallel-pipelined (PP), and the rotate-tiling (RT) data communication schemes are used. By combining the four data compression schemes with the three data communication schemes, we have twelve image composition methods. These twelve methods are implemented on a PC cluster. The data computation time and the data communication time are measured. The experimental results show that the TRLE data compression scheme with the RT data communication scheme outperforms other eleven image composition methods.

Keyword: image composition, bounding rectangle, run-length encoding, template run-length encoding, parallel volume rendering systems.

1. Introduction

Volume rendering [2] can be used to analyze the shape and volumetric property of three-dimensional objects in research areas such as medical imaging and scientific visualizing. However, most volume rendering methods that produce effective visualizations are computation intensive. It is difficult for them to achieve interactive rendering rates for large datasets. In addition, volume datasets are too large to be stored in the memory of a single processor. One way to solve the above problems is to parallelize the serial volume rendering methods [12].

A parallel volume rendering system [8], in general, consists of three stages; the data partition stage, the data render stage, and the image composition stage. In the data partition stage, the volume dataset is partitioned into sub-volumes by an efficient data partitioning method and the sub-volumes are distributed to processors. In the data render stage, each processor uses a volume rendering algorithm on the assigned sub-volume to generate a partial image. In the image composition stage, the partial images generated by processors are composited to form a final image [11]. When the number of processors is large, the image composition stage becomes a bottleneck of a parallel volume rendering system. Hence, a good image composition method is very important to the performance of a parallel volume rendering system.

In general, there are two ways to improve the performance of image composition of parallel volume rendering systems [1]. One is to use an efficient data communication scheme to minimize the data communication overheads in sending and receiving the partial images of processors. The other is to use an efficient data compression scheme to reduce the communication sizes of partial images. The reasons for using a data compression scheme are two-folds. First, a partial image may contain many blank pixels. These blank pixels are useless in image composition. If we can filter out these blank pixels in some ways, the size of a partial image can be reduced, that is, the data transmission time among processors can be reduced. Second, if we can filter out blank pixels of a partial image, the number of *over* operations spent on these blank pixels for composition can be eliminated. By reducing the data communication size of a partial image, the overall image composition time can be improved. In this paper, we focus on finding an efficient data compression scheme for image composition.

The bounding rectangle (BR) and the run-length encoding (RLE) are two well-known data compression schemes used in computer graphics [3]. They are also used in the image composition of a parallel volume

Proceedings of the First International Symposium on Cyber Worlds (CW'02)
0-7695-1862-1/02 $17.00 © 2002 IEEE

rendering system [9] [13]. Ma *et al*. [9] used the BR scheme for data compression. In [9], the BR scheme uses a bounding rectangle to embrace the non-blank pixels of the partial image of a processor. The image composition using the BR scheme consists of three steps. Assume that processor P_j needs to composite the partial images of P_i and P_j. In the first step, the bounding rectangle to embrace the non-blank pixels of the partial image of P_i is formed. P_i then sends pixels in the bounding rectangle to P_j by a data communication scheme in the second step. In the third step, P_j uses the *over* operation to composite the pixels in the sent bounding rectangle with the pixels of its partial image. An example of the image composition using the BR scheme is given in Figure 1.

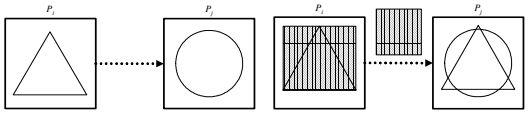


Figure 1. An example of the BR scheme

If there are many blank pixels in a bounding rectangle of a partial image, the BR scheme may not have good performance. An example of this case is given in Figure 2. In Figure 2, the bounding rectangle of P_i is the whole partial image of P_i. P_i needs to send the whole partial image to P_j. However, only the black portion of the triangle contains non-blank pixels. Most of pixels in the bounding rectangle that contains the triangle are blank pixels. In this case, the BR scheme neither reduces the data communication size for P_i nor reduces the number of *over* operations for P_j.

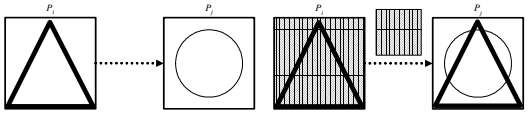


Figure 2. An example of the worst case of the BR scheme

Yang *et al*. [13] used the RLE scheme for data compression. In [13], the RLE scheme encodes each scanline of a partial image. The image composition using the RLE scheme consists of four steps. Assume that processor P_j needs to composite the partial images of P_i and P_j. In the first step, pixels of the partial image in P_i are encoded by using the RLE scheme and the corresponding RLE codes are formed. P_i then sends the RLE codes and the corresponding non-blank pixels to P_j by a data communication scheme in the second step. In the third step, P_j decodes the RLE codes to get the partial image of P_i. P_j then uses the *over* operation to composite the partial images of P_i and P_j in the forth step. An example of the image composition using the RLE scheme is given in Figure 3.

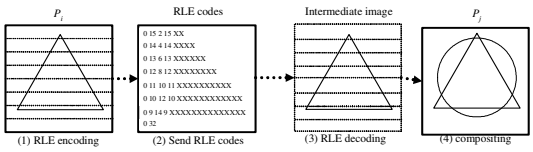


Figure 3. An example of the RLE scheme

If the non-blank and blank pixels of the partial image are interlaced, the RLE scheme is not efficient since the RLE codes are too large. An example of this case is given in Figure 4. In Figure 4, the RLE codes are larger than the values of the non-blank pixels, that is, the data size of the RLE codes is large than that of the partial image. The data transmission overhead is increased.

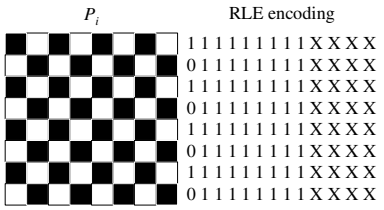


Figure 4. An example of the worst case of the RLE scheme

Yang *et al*. [13] combined the BR and the RLE schemes, denoted as the BRLC scheme, to reduce the data communication sizes. The BRLC scheme first uses the BR scheme to find a bounding rectangle with non-blank pixels of a partial image. The scheme then applies the RLE scheme to encode the pixels in the bounding rectangle. For many cases, the BRLC scheme performs better than the BR and the RLE schemes. However, the BRLC scheme cannot solve the problems presented in Figure 4 either.

To overcome the disadvantages of the BR, the RLE, and the BRLC schemes, we propose an efficient data compression scheme, the template run-length encoding (TRLE) scheme, for image composition of a parallel volume rendering system. Given an image with $2n \times 2n$ pixels, in the TRLE scheme, the image consists of $n \times n$ blocks and each block has 2×2 pixels. Since each pixel is either a blank or a non-blank pixel, there are sixteen blank/non-blank pixel combinations in a block. We call these sixteen blank/non-blank pixel combinations as *templates*. With these templates, the TRLE scheme encodes a partial image block by block similar to the RLE scheme. However, the TRLE scheme can filter out or use small space to encode blocks whose four pixels are blank pixels, that is, the TRLE scheme can encode a partial image according to the shape of non-blank pixels. In the TRLE scheme, the bit operations, *and*, *or*, and *xor*, are used to encode and decode a partial image. Hence, the TRLE scheme is easy to be implemented and the time

spent on encoding and decoding is small compared to the overall image composition time. An example of the TRLE scheme is given in Figure 5. Since the TRLE scheme can encode a partial image according to the shape of non-blank pixels, it can solve the problems of the BR, the RLE, and the BRLC schemes efficiently. For example, for the image shown in Figure 5, the data size compressed by using the BR, the RLE, the BRLC, and the TRLE schemes is 64×16 = 1024, 32×16 + 72×12 = 1376, 32×16 + 72×12 = 1376, and 32×16 + 16 = 528 bytes, respectively (assume that each pixel requires 16 bytes to store values). The data size compressed by the TRLE scheme is the smallest among these four data compressed schemes.

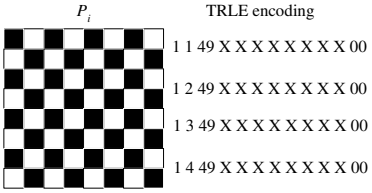


Figure 5. An example of the TRLE scheme

To evaluate the performance of the TRLE scheme, we compare the proposed scheme with the BR, the RLE, and the BRLC schemes. Both theoretical and experimental analyses are conducted. In theoretical analysis, we analyze the ranges of data compression ratio of these four schemes. By combining the four data compression schemes and three data communication schemes (the binary-swap (BS) [9], the parallel-pipelined (PP) [5], and the rotate-tiling (RT) [7] methods), we have twelve image composition methods. In the experimental, for each method, the data computation time and the data communication time are measured on a PC cluster. The experimental results show that the TRLE data compression scheme with the RT data communication scheme outperforms other image composition methods.

The rest of the paper is organized as follows. The TRLE scheme will be presented in Section 2. In Section 3, we analyze the ranges of data compression ratio of the TRLE, the BR, the RLE, and the BRLC schemes. In Section 4, we analyze these 12 image composition algorithms in terms of the communication time and the computation time. In Section 5, the experimental results of these 12 image composition methods on a PC cluster will be discussed.

2. The TRLE data compression scheme

Given an image with $2n \times 2n$ pixels, in the TRLE scheme, the image is treated as $n \times n$ blocks and each block has 2×2 pixels. Pixels in a block are labeled as shown in Figure 6. A pixel of an image is a *blank* pixel if its value is less than a threshold. Otherwise, it is a *non-blank* pixel. For a pixel in a block, it is either a blank or a non-blank pixel. There are sixteen blank and non-blank pixel combinations in a block. We define these 16 blank and non-blank pixel combinations as *templates*. To represent these templates, 4-bit binary codes are used. Given a 4-bit binary code $b_3b_2b_1b_0$, b_3, b_2, b_1, and b_0 denote the pixel with label 0, 1, 2, and 3 in a block, respectively. The value of b_i in a 4-bit binary code is 0 if the corresponding pixel is a blank pixel. Otherwise, b_i is 1. The 4-bit binary codes of the templates are given in Figure 7. In Figure 7, white squares represent blank pixels while black squares represent non-blank pixels.

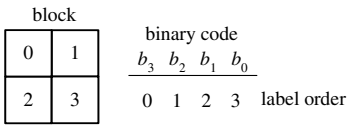


Figure 6. The label of pixels in a block

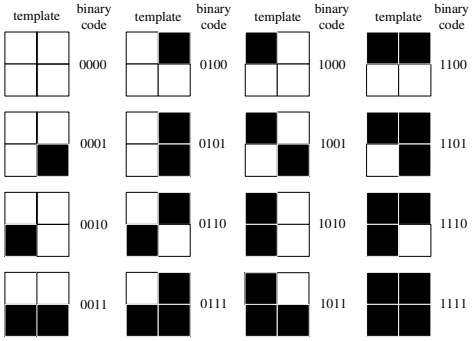


Figure 7. The 4-bit binary codes of templates

Given an image consists of $n \times n$ blocks and each block has 2×2 pixels, to compress the image, the TRLE scheme uses the templates to encode blocks row by row. Blocks in the same row are encoded as a *TRLE_sequence* (will be defined later). By packing all *TRLE_sequences* in a packet, the packet is the compressed image that can be sent/received among processors. We have the following definitions.

Definition 1: A *template_code* is an 8-bit long code. In a *template_code*, the lower four bits represent the binary code of a template. The upper four bits represent the repetition of the template specified in the lower four bits. A *template_code* can represent up to 15 replication of a template.

An example of a *template_code* is given in Figure 8. In Figure 8, the *template_code* is "2A". It means that two consecutive blocks are the same block and are encoded by template "1010".

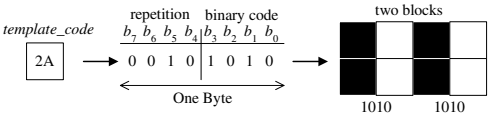


Figure 8. An example of a *template_code*

Definition 2: A *TRLE_code* consists of a *template_code* and the values of non-blank pixels in a template.

The number of bytes to store the values of a pixel depends on the volume data used. For the volume data used in this paper, each pixel is represented by 16 bytes. An example of *TRLE_code* is given in Figure 9. In Figure 9, the *template_code* of the *TRLE_code* is "2A". It means that two consecutive blocks are the same block and are encoded by template "1010". In template "1010", pixels with labels 0 and 2 are non-blank pixels. The first 16 bytes followed the *template_code* in the *TRLE_code* store the values of non-blank pixel P_1, the next 16 bytes store the values of non-blank pixel P_2 followed by the values of P_3 and P_4.

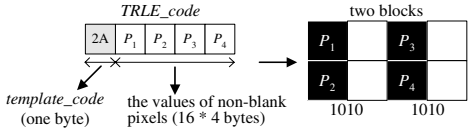


Figure 9. An example of a *TRLE_code*

Definition 3: A *TRLE_sequence* is an encoded sequence for blocks in the same row of an image. It consists of a 2-byte index to store the coordinate of the first block that contains non-blank pixels in a row, a set of *template_code*/*TRLE_code* for blocks in the same row, and an end byte "00". In the 2-byte index, the first byte and the second byte store the row index and the column index of the block, respectively.

An example of a *TRLE_sequence* is given in Figure 10. For the *TRLE_sequence* shown in Figure 10, the 2-byte index is "02 01". It means that the *TRLE_sequence* encodes the blocks in the first row of an image. The first block that contains non-blank pixels in the first row is the second block. Five *TRLE_codes* are followed the 2-byte index. They encode the blocks in the first row according to templates. At the end of the *TRLE_sequence* is an end byte with value "00". It indicates the end of row. In the example, it is possible that there are blocks followed block 8. However, they are blocks whose four pixels are blank pixels and are eliminated from the TRLE scheme. From this example, we can see that the purpose of the 2-bype index and the end byte of a *TRLE_sequence* is to find the boundary of an image. In a *TRLE_sequence*, for the 2-byte index, the TRLE scheme can handle an image with size up to 512×512 pixels. For an image size over 512×512 pixels, the TRLE scheme uses a 4-byte index (x and y occupied 2-bye each) that can handle an image with size up to 65536×65536 blocks.

Definition 4: A *TRLE_packet* is a one-dimensional array to store the set of *TRLE_sequence* of a partial image. An example of a *TRLE_packet* is given in Figure 11. In Figure 11, an image with 8×8 pixels that consists of 4×4 blocks is given. There are four *TRLE_seqences*. The *TRLE_packet* contains the four *TRLE_seqences*. Form the *TRLE_seqences* shown in Figure 11, we can see that the TRLE scheme can encode an image according to the shape of non-blank pixels in the image. For example, the blank pixels outside the triangle are filtered out in the *TRLE_seqences*. For blank pixels inside the triangle, they are only encoded by *template_codes*. Therefore, in general, the TRLE scheme can have better compression ratio compared with the BR, the RLE, and the BRLC schemes.

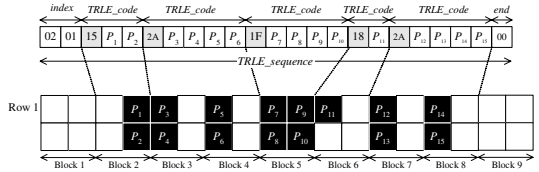


Figure 10. An example of a *TRLE_sequence*

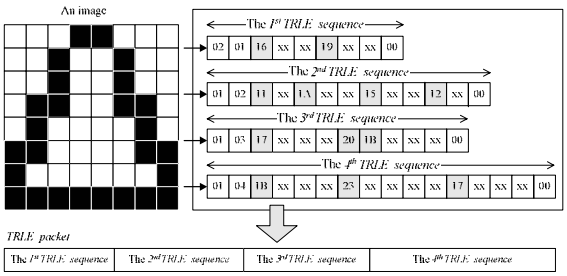


Figure 11. An example of a *TRLE_packet*

According to the above definitions, the TRLE scheme can easily encode a partial image to form a *TRLE_packet* or decode a *TRLE_packet* to get the corresponding image. In the TLRE scheme, bit operations, *and*, *or*, and *xor* are used to encode and decode an image. The computation overheads spent on encoding and decoding of an image are small.

3. Theoretical analysis of data compression schemes

One of the reasons to use a data compression scheme in the image composition stage of a parallel volume rendering system is to reduce the data transmission time of partial images. In the following, we analyze the BR, the RLE, the BRLC and the TRLE data compression schemes in terms of the data compression ratio. Based on the data compression ratio of a data compression scheme, we derive the best and the worst case bounds of a data compression scheme. A summary of the notations used in this section is given below. A pixel of a partial image is represented by 16 bytes in our analysis.

- PA – The number of pixels in a partial image.
- PA_{nb} – The number of non-blank pixels of a partial image.
- PA_{BR} – The number of pixels in a bounding rectangle of

the BR scheme.

- $CODE_{RLE}$ – The encoding code size of the RLE scheme.
- $CODE_{BRLC}$ – The encoding code size of the BRLC scheme.
- $CODE_{TRLE}$ – The encoding code size of the TRLE scheme.

The compression ratio of a partial image of method M is defined as

$$CR(M) = \frac{\text{The total data size per bytes}}{\text{The total compressed data size per bytes}}.$$

Due to paper limitation, a summary of the ranges of data compression ratio for these four data compression schemes is given in Table 1. The range comparison of the data compression ratio of the four data compression schemes are shown in Figure 12. In Figure 12, the range of the BR scheme covers those of other three schemes. It indicates that the compression ration is heavily influenced by the shape of an image. The range of the BRLC scheme also covers that of the RLE scheme. It also indicates that the BRLC scheme is more sensitive to the shape of an image than the RLE scheme. The range of the TRLE scheme overlaps those of the RLE and the BRLC schemes. However, the average compression ratio of the TRLE scheme is better than those of the RLE and the BRLC schemes.

Table 1. The ranges of data compression ratio of four data compression schemes

Method	*Ranges*
BR	$\frac{PA\times16}{PA\times16+8} \le CR(BR) \le \frac{PA\times16}{PA_{nb}\times16+8}$
RLE	$\frac{PA\times16}{PA_{nb}\times16+PA\times4+\sqrt{PA}\times2} \le CR(RLE) \le \frac{PA\times16}{PA_{nb}\times16+\sqrt{PA_{nb}}\times2+\sqrt{PA}\times2}$
BRLC	$\frac{PA\times16}{PA_{nb}\times16+PA\times4+\sqrt{PA}\times2+8} \le CR(BRLC) \le \frac{PA\times16}{PA_{nb}\times16+\sqrt{PA_{nb}}\times4+8}$
TRLE	$\frac{PA\times16}{PA_{nb}\times16+PA/2+\sqrt{PA}\times3} \le CR(TRLE) \le \frac{PA\times16}{PA_{nb}\times16+\sqrt{PA_{nb}}\times4}$

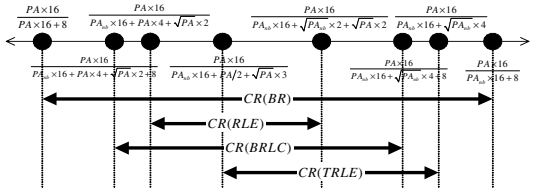


Figure 12. The comparison of the ranges of CR of the four data compression schemes

4. Analysis of image composition methods with data compression schemes

To use a data compression scheme in the image composition stage of a parallel volume rendering system, it needs to combine with some data communication schemes to send/receive partial images among processors. The following is a generic image composition algorithm with a data compression scheme.

Algorithm *Comm_Compress_Scheme*(P, A) {
/* P is the number of processors. */
/* A is the initial image of each processor. */
1. **for** k = 1 to communication_step **do** {
2. **for** each processor P_r **do parallel** {
3. P_r sends compress(A) to P_i;
4. P_r receives compress(A) from P_j;
5. P_r uses the over operation to composite the received compress(A) with its local image;
6. }
7. }
end_of_*Comm_Compress_Scheme*

In this section, we analyze the theoretical performance of the BS, the PP, and the RT data communication schemes with the BR, the RLE, the BRLC, and the TRLE data compression schemes. The three data communication schemes and the four data compression schemes have 12 combinations. A summary of the notations used in this section is given below.

- P – The number of processors.
- P_i – The processor with rank i.
- A – The image size in pixels.
- $S(M)$ – The number of communication steps of method M.
- N – The number of initial blocks of a partial image in the RT method.
- T_s – The startup time of a communication channel.
- T_c – The data transmission time per byte.
- T_o – The computation time of the *over* operation per pixel.
- $T_{comm}(M)$ – The total communication time of method M.
- $T_{comp}(M)$ – The total computation time of method M.
- $T^k_{comm}(M, P_i)$ – The communication time of P_i in the kth communication step of method M.
- $T^k_{comp}(M, P_i)$ – The computation time of P_i in the kth communication step of method M.
- $T^k_{encode}(M, P_i)$ – The data encoding time of P_i in the kth communication step of method M.
- $T^k_{decode}(M, P_i)$ – The data decoding time of P_i in the kth communication step of method M.
- $A^k_i(M, P_i)$ – The pixel size for sent/received by P_i in the kth communication step of method M.
- $A^{i,k}(M)$ – The pixel size of partial image of P_i in the kth communication step of method M.
- $A^{i,k}_{BR}(M)$ – The number of pixels of a bounding rectangle of $A^{i,k}(M)$.
- $A^{i,k}_{nb}(M)$ – The number of non-blank pixels of $A^{i,k}(M)$.
- $CODE^{i,k}_{RLE}(M)$ – The number of the RLE encoding codes of $A^{i,k}(M)$.

- $CODE^{i,k}_{BRLC}(M)$ – The number of the BRLC encoding codes of $A^{i,k}(M)$.
- $CODE^{i,k}_{TRLE}(M)$ – The number of the TRLE encoding codes of $A^{i,k}(M)$.
- T_{BR} – The computation time for finding a bounding rectangle.
- T_{encode} – The computation time of encoding a pixel.
- T_{decode} – The computation time of decoding a pixel.

Table 2. Theoretical time of the 12 image composition methods

Method	Time
BS_BR	$T_{comm}(M)=\sum_{k=1}^{\log P}\left(T_s+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{BR}(M)\times 16+8\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{\log P}\left(T_{BR}+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{BR}(M)\right)\times T_o\right)$
BS_RLE	$T_{comm}(M)=\sum_{k=1}^{\log P}\left(T_s+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{nb}(M)\times 16+CODE^{i,k}_{RLE}(M)\times 2\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{\log P}\underset{i=0}{\overset{P-1}{MAX}}\left(T_{encode}\times A^{i,k}(M)+T_{decode}\times CODE^{i,k}_{RLE}(M)+A^{i,k}_{nb}(M)\times T_o\right)$
BS_BRLC	$T_{comm}(M)=\sum_{k=1}^{\log P}\left(T_s+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{nb}(M)\times 16+CODE^{i,k}_{BRLC}(M)\times 2+8\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{\log P}\left(T_{BR}+\underset{i=0}{\overset{P-1}{MAX}}\left(T_{encode}\times A^{i,k}_{BR}(M)+T_{decode}\times CODE^{i,k}_{BRLC}(M)+A^{i,k}_{nb}(M)\times T_c\right)\right)$
BS_TRLE	$T_{comm}(M)=\sum_{k=1}^{\log P}\left(T_s+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{nb}(M)\times 16+CODE^{i,k}_{TRLE}(M)\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{\log P}\underset{i=0}{\overset{P-1}{MAX}}\left(T_{encode}\times A^{i,k}(M)+T_{decode}\times CODE^{i,k}_{TRLE}(M)+A^{i,k}_{nb}(M)\times T_o\right)$
PP_BR	$T_{comm}(M)=\sum_{k=1}^{P-1}\left(T_s+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{BR}(M)\times 16+8\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{P-1}\left(T_{BR}+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{BR}(M)\right)\times T_o\right)$
PP_RLE	$T_{comm}(M)=\sum_{k=1}^{P-1}\left(T_s+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{nb}(M)\times 16+CODE^{i,k}_{RLE}(M)\times 2\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{P-1}\underset{i=0}{\overset{P-1}{MAX}}\left(T_{encode}\times A^{i,k}(M)+T_{decode}\times CODE^{i,k}_{RLE}(M)+A^{i,k}_{nb}(M)\times T_o\right)$
PP_BRLC	$T_{comm}(M)=\sum_{k=1}^{P-1}\left(T_s+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{nb}(M)\times 16+CODE^{i,k}_{BRLC}(M)\times 2+8\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{P-1}\left(T_{BR}+\underset{i=0}{\overset{P-1}{MAX}}\left(T_{encode}\times A^{i,k}_{BR}(M)+T_{decode}\times CODE^{i,k}_{BRLC}(M)+A^{i,k}_{nb}(M)\times T_c\right)\right)$
PP_TRLE	$T_{comm}(M)=\sum_{k=1}^{P-1}\left(T_s+\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{nb}(M)\times 16+CODE^{i,k}_{TRLE}(M)\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{P-1}\underset{i=0}{\overset{P-1}{MAX}}\left(T_{encode}\times A^{i,k}(M)+T_{decode}\times CODE^{i,k}_{TRLE}(M)+A^{i,k}_{nb}(M)\times T_o\right)$
RT_BR	$T_{comm}(M)=\sum_{k=1}^{\lceil\log P\rceil}\left(\lceil\log N\rceil T_s+\frac{\lceil\log N\rceil}{N}\times\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{BR}(M)\times 16+8\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{\lceil\log P\rceil}\left(T_{BR}+\frac{\lceil\log N\rceil}{N}\times\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{BR}(M)\right)\times T_o\right)$
RT_RLE	$T_{comm}(M)=\sum_{k=1}^{\lceil\log P\rceil}\left(\lceil\log N\rceil T_s+\frac{\lceil\log N\rceil}{N}\times\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{nb}(M)\times 16+CODE^{i,k}_{RLE}(M)\times 2\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{\lceil\log P\rceil}\frac{\lceil\log N\rceil}{N}\times\underset{i=0}{\overset{P-1}{MAX}}\left(T_{encode}\times A^{i,k}(M)+T_{decode}\times CODE^{i,k}_{RLE}(M)+A^{i,k}_{nb}(M)\times T_o\right)$
RT_BRLC	$T_{comm}(M)=\sum_{k=1}^{\lceil\log P\rceil}\left(\lceil\log N\rceil T_s+\frac{\lceil\log N\rceil}{N}\times\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{nb}(M)\times 16+CODE^{i,k}_{BRLC}(M)\times 2+8\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{\lceil\log P\rceil}\left(T_{BR}+\frac{\lceil\log N\rceil}{N}\times\underset{i=0}{\overset{P-1}{MAX}}\left(T_{encode}\times A^{i,k}_{BR}(M)+T_{decode}\times CODE^{i,k}_{BRLC}(M)+A^{i,k}_{nb}(M)\times T_c\right)\right)$
RT_TRLE	$T_{comm}(M)=\sum_{k=1}^{\lceil\log P\rceil}\left(\lceil\log N\rceil T_s+\frac{\lceil\log N\rceil}{N}\times\underset{i=0}{\overset{P-1}{MAX}}\left(A^{i,k}_{nb}(M)\times 16+CODE^{i,k}_{TRLE}(M)\right)\times T_c\right)$
	$T_{comp}(M)=\sum_{k=1}^{\lceil\log P\rceil}\frac{\lceil\log N\rceil}{N}\times\underset{i=0}{\overset{P-1}{MAX}}\left(T_{encode}\times A^{i,k}(M)+T_{decode}\times CODE^{i,k}_{TRLE}(M)+A^{i,k}_{nb}(M)\times T_o\right)$

To analyze the theoretical performance of the image composition methods, in the cost model, a synchronous communication mode is used. In this model, all processors start their computation after each processor completes its communication. In real situation, an asynchronous communication mode can be applied as well. However, it is difficult to analyze the theoretical performance if an asynchronous communication mode is used. According to above notations, the cost model of an image composition method *M* is defined as

$$T_{total}(M)=\sum_{k=1}^{S(M)}\max\{T^k_{comm}(M,P_i)+T^k_{comp}(M,P_i)\}. \quad (1)$$

In our communication model, we assume that each processor can communicate with all other processors in one communication step. $T^k_{comm}(M, P_i)$ is defined as

$$T^k_{comm}(M,\ P_i)\ =\delta^k_i\times T_s+A^k_i(M,P_i)\times T_p, \quad (2)$$

where δ^k_i is the number of processors that P_i sends data to in the *k*th communication step. In our computation model, we assume that the partial image in each processor is first encoded by method *M*. Each pixel of a compressed block then received from another processor is decoded and composited using the *over* operation. Therefore, $T^k_{comp}(M)$ is defined as

$$T^k_{comp}(M,P_i)=T^k_{encode}(M,P_i)+T^k_{dncode}(M,P_i)+A^k_i(M,P_i)T_o \quad (3)$$

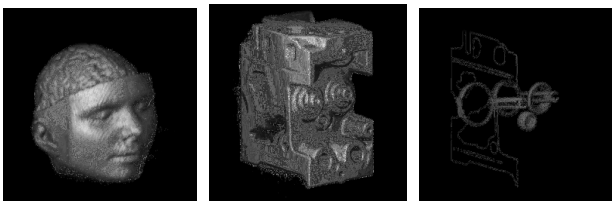
According to Equations (2) and (3), we can see that $A^k_i(M,P_i)$ affects the performance of the image composition methods. A good data compression scheme can reduce the value of $A^k_i(M,P_i)$ and is important to an image composition method. Due to paper limitation, the data communication time and the data computation time of the 12 image composition methods are shown in Table 2.

5. Experimental results

To evaluate the performance of the TRLE scheme, we compare the TRLE scheme with the BR, the RLE, and the BRLC schemes on a PC cluster. The PC cluster is located at Parallel and Distributed Processing Laboratory of Feng Chia University in Taiwan. Each node in the PC cluster uses an INTEL Pentium III CPU with a clock rate of 800 MHz. There are 32 CPUs in this PC cluster, and each node has 512KB first-level data cache and 256MB of memory space. Each node is fully connected by Myrinet, and the network bandwidth of the PC cluster is 650 MB/sec.

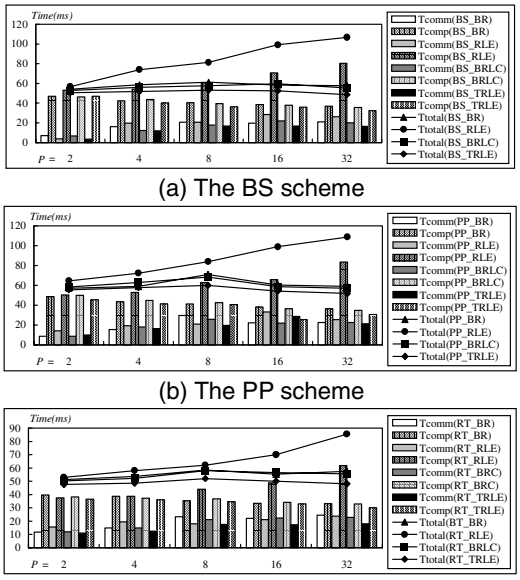
A parallel volume rendering system consists of three main stages: the data partition stage, the volume render stage, and the image composition stage. To implement the data compression schemes, in the data partition stage, we use the efficient 1-D and 2-D partitioning schemes [6] to distribute a volume dataset to processors. In the data render stage, each processor uses the shear-warp factorization [4] volume rendering method to generate a partial image. In the image composition stage, the twelve image composition methods are used to composite partial images. In the PC cluster, we use C and MPICH_GM [10] message passing libraries to implement the data compression schemes.

Three volume dataset are used to evaluate the performance of these data compression schemes. The first test sample is a "*brain*" dataset, which generated from the MR scan of a human brain, and the dimensions of the dataset is 256× 256 × 225. The second test sample is an "*Engine_low*" dataset, which is the CT scan of an engine block and the dimensions of the dataset is 256× 256 × 110. The third is an "*Engine_high*" dataset, which is the CT scan of an engine block. The density of each voxels is larger than 180, and the dimensions of the dataset is 256× 256 × 110. Figure 13 shows the final images of the test samples. Each image is grayscale color and contains 512 × 512 pixels.



(a)*Brain* (b) *Engine_low* (c) *Engine_high*

Figure 13. Test samples for the data compression schemes



(a) The BS scheme

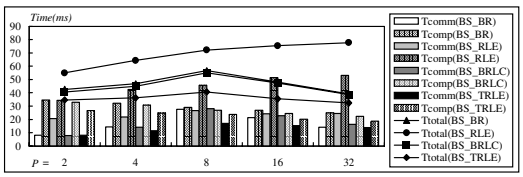
(b) The PP scheme

(c) The RT scheme

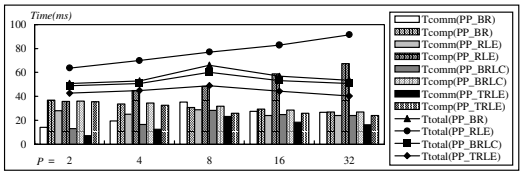
Figure 14. The image composition time for "*Brain*"

Figure 14 shows the data communication time and the data computation time of the 12 image composition methods for test sample "*Engine_low*" dataset on a PC cluster, respectively. From Figure 14, we have the following observations.

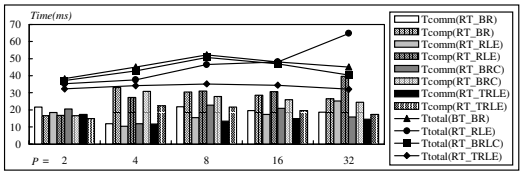
1. The RT scheme with the TRLE scheme has the best performance of the 12 image composition methods for different numbers of processors.



(a) The BS scheme

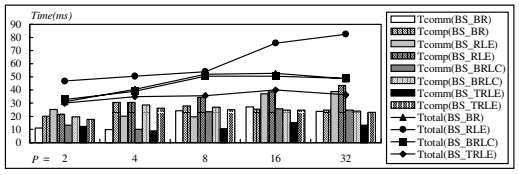


(b) The PP scheme

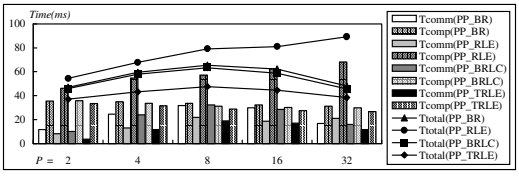


(c) The RT scheme

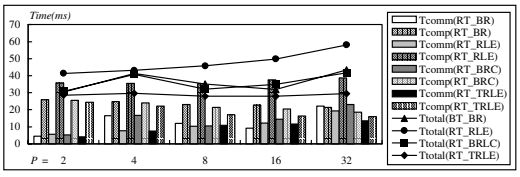
Figure 15. The image composition time for "*Engine_low*"



(a) The BS scheme



(b) The PP scheme



(c) The RT scheme

Figure 16. The image composition time for "*Engine_high*"

2. The BS, the PP, and the RT schemes with the TRLE scheme have better performance than those with the BR, the RLE, and the BRLC schemes.
3. When the number of processor increases, the image composition time of any data communication scheme with the TRLE scheme is slightly increased or decreased compared to other data compression schemes, that is, the time is in a horizontal line.

Figures 15 and 16 show the data communication time and the data computation time of the 12 image composition schemes for test samples "*Engine_low*" and

"*Engine_high*" dataset on a PC cluster, respectively. From Figures 15 and 16, we have similar observations as those of Figure 14.

6. Conclusions

In this paper, we have proposed an efficient data compression scheme, the template run-length encoding (TRLE) scheme, for image composition of parallel volume rendering systems. To evaluate the performance of the TRLE scheme, we compare the proposed scheme with the BR, the RLE, and the BRLC schemes. Both theoretical and experimental analyses are conducted. For the theoretical analysis, we compare the ranges of the compression ratio of these four data compression schemes. For the experimental analysis, the BR, the RLE, the BRLC, and the TRLE data compression schemes have implemented with the binary-swap (BS), the parallel-pipelined (PP), and the rotate-tiling (RT) data communication schemes. The data computation time and the data communication time are measured on a PC cluster. From the experimental results, we have the following remarks.

Remark 1: The RT scheme with the TRLE scheme has the best performance of the 12 image composition methods for different numbers of processors.

Remark 2: The BS, the PP, and the RT schemes with the TRLE scheme have better performance than those with the BR, the RLE, and the BRLC schemes.

Remark 3: When the number of processor increases, the image composition time of any data communication scheme with the TRLE scheme is almost the same.

7. Acknowledgement

This work was partially supported by the National Science Council of Republic of China under contract NSC90-2213-E-035-021.

References

[1] J. Ahrens and J. Painter, "Efficient Soft-Last Rendering Using Compression-Based Image Compositing," *Proceedings of the second Eurographics Workshop on Parallel Graphics and Visualization*, 1998.

[2] R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *Computer Graphics* (*In Proceedings of SIGGRAPH'88*), vol. 22, no. 4, Atlanta, 1988, pp. 65-74.

[3] J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes, *Computer Graphics: Principles and Practice Second Edition in C*, Mass.: Addison-Wesley, 1990.

[4] P. Lacroute, "Analysis of a Parallel Volume Rendering System Based on the Shear-Warp Factorization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 3, 1996, pp. 218-231.

[5] T.Y. Lee, "Image Composition Schemes for Soft-Last Polygon Rendering on 2D Mesh Multicomputers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 3, Sep. 1996, pp. 202-217.

[6] C.F. Lin, Y.C. Chung, and D.L. Yang, "Parallel Shear-Warp Factorization Volume Rendering Using Efficient 1-D and 2-D Partitioning Schemes on Distributed Memory Multicomputers," *The Journal of Supercomputing*, vol. 22, no. 3, Jul. 2002, pp. 277-302.

[7] C.F. Lin, D.L. Yang, and Y.C. Chung, "A Rotate-Tiling Image Composition Scheme for Parallel Volume Rendering on Distributed Memory Multicomputers," *Proceedings of IEEE International Parallel and Distributed Processing Symposiums (CD-ROM)*, San Francisco, USA, Apr. 2001.

[8] K.L. Ma, J.S. Painter, C.D. Hansen, and M.F. Krogh, "A Data Distributed, Parallel Algorithm for Ray-Traced Volume Rendering," *Proceedings of 1993 Parallel Rendering Symposium* (*PRS'93*), San Jose, Oct. 1993, pp. 15-22.

[9] K.L. Ma, J.S. Painter, C.D. Hansen, and M.F. Krogh, "Parallel Volume Rendering Using Binary-Swap Composition," *IEEE Computer Graphics and Applications*, vol. 14, no. 4, Jul. 1994, pp. 59-68.

[10] MPI Forum, MPI: A Message-Passing Interface Standard, May 1994.

[11] T. Porter and T. Duff, "Composition Digital Images," *Computer Graphics* (*In Proceedings of SIGGRAPH'84*), vol. 18, Jul. 1984, pp. 253-259.

[12] J.P. Singh, A. Gupta, and M. Levoy, "Parallel Visualization Algorithms: Performance and Architectural Implications," *Computer*, vol. 27, no. 7, Jul. 1994, pp. 45-55.

[13] D.L. Yang, J.C. Yu, and Y.C. Chung, "Efficient Compositing Schemes for the Sort-Last-Sparse Parallel Volume Rendering System on Distributed Memory Multicomputers," *The Journal of Supercomputing*, vol. 18, no. 2, Feb. 2001, pp. 201-220.