J Supercomput (2007) 39: 321–345
DOI 10.1007/s11227-006-0012-5

TRLE—an efficient data compression scheme for image composition of volume rendering on distributed memory multicomputers

Chin-Feng Lin · Yeh-Ching Chung · Don-Lin Yang

Published online: 17 February 2007

Abstract Data compression is a well-known method to improve the image composition time of parallel volume rendering on distributed memory multicomputers. In this paper, we propose an efficient data compression scheme, the template run-length encoding (TRLE) scheme, for image composition. 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. Since a pixel can be a blank or non-blank pixel, there 16 *templates* in a block. To compress an image, the TRLE scheme encodes an image block by block similar to the run-length encoding 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. 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, the parallel-pipelined, and the rotate-tiling 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 an IBM SP2 parallel machine. Four volume datasets are used as test samples. The data computation time and the data communication time are

A preliminary version of this work was appeared in IPDPS'01.

C.-F. Lin (✉)
Department of Information Management, Chang Jung Christian University, Tainan, Taiwan 711, R.O.C.
e-mail: cflin@cju.edu.tw

Y.-C. Chung
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 300, R.O.C.
e-mail: ychung@cs.nthu.edu.tw

D.-L. Yang
Department of Information Engineering, Feng Chia University, Taichung, Taiwan 407, R.O.C.
e-mail: dlyang@fcu.edu.tw

measured. The experimental results show that the TRLE data compression scheme with the rotate-tiling data communication scheme outperforms other eleven image composition methods for all test samples.

Keywords Image composition · Parallel volume rendering · Bounding rectangle · Run-length encoding · Template run-length encoding

1 Introduction

Volume rendering [3, 5, 6, 8] 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 [13–15]. 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 on distributed memory multicomputers [23–29].

A parallel volume rendering system on distributed memory multicomputers [18, 22], in general, consists of three stages, the data partition stage, the volume 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 volume 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 [1, 2, 21]. 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 on distributed memory multicomputers.

In general, there are two ways to improve the performance of image composition of a parallel volume rendering system on distributed memory multicomputers. 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-fold. 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 [4]. They are also used in image composition of a parallel volume rendering system on distributed memory

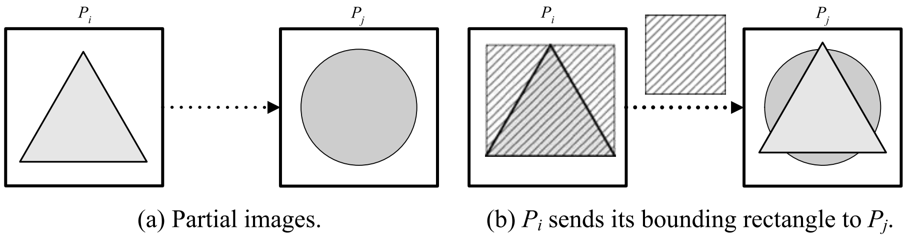


Fig. 1 An example of image composition by using the BR scheme

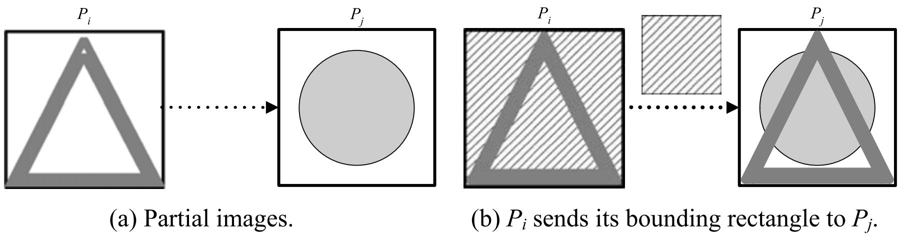


Fig. 2 An example of the worst case of the BR scheme

multicomputers [19, 30]. Ma et al. [19] used the BR scheme for data compression. In [19], the BR scheme uses a bounding rectangle to embrace the non-blank pixels of the partial image of a processor. Image composition by 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. Then, P_i 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 image composition by using the BR scheme is given in Fig. 1.

If there are many blank pixels in a bounding rectangle, the BR scheme may not have good performance. An example of this case is given in Fig. 2. In Fig. 2, for P_i, the bounding rectangle 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.

Yang et al. [30] used the RLE scheme for data compression. In [30], the RLE scheme encodes each scanline of a partial image. Image composition by using the RLE scheme consists of four steps. We 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. In the last step, P_j uses the *over* operation to composite the partial images of P_i and P_j in the forth step. An example of image composition by using the RLE scheme is given in Fig. 3.

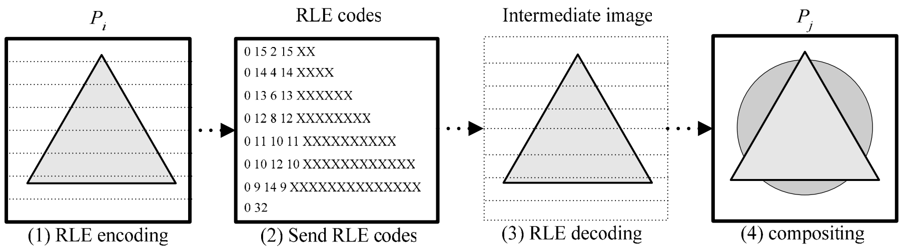


Fig. 3 An example of image composition by using the RLE scheme

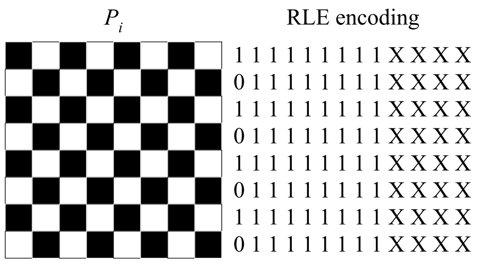

P_i
RLE encoding
1 1 1 1 1 1 1 1 1 X X X X
0 1 1 1 1 1 1 1 1 X X X X
1 1 1 1 1 1 1 1 1 X X X X
0 1 1 1 1 1 1 1 1 X X X X
1 1 1 1 1 1 1 1 1 X X X X
0 1 1 1 1 1 1 1 1 X X X X
1 1 1 1 1 1 1 1 1 X X X X
0 1 1 1 1 1 1 1 1 X X X X

Fig. 4 An example of the worst case 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 Fig. 4. In Fig. 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.

Yang et al. [30] 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 Fig. 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 on distributed memory multicomputers. 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 Fig. 5. Since the TRLE scheme can encode a partial image according to the shape of non-blank pixels, it can solve

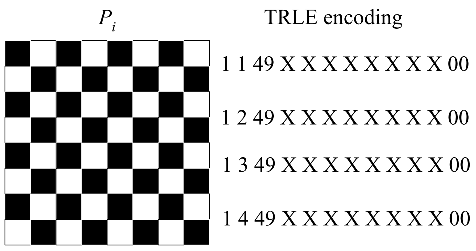

P_i
TRLE encoding
1 1 49 X X X X X X X X X 00
1 2 49 X X X X X X X X X 00
1 3 49 X X X X X X X X X 00
1 4 49 X X X X X X X X X 00

Fig. 5 The encoding codes for the TRLE scheme

the problems of the BR, the RLE, and the BRLC schemes efficiently. For example, for the image shown in Fig. 5, the data size compressed by using the BR, the RLE, the BRLC, and the TRLE schemes is $64 \times 16 = 1024$, $32 \times 16 + 72 \times 12 = 1376$, $32 \times 16 + 72 \times 12 = 1376$, and $32 \times 16 + 16 = 528$ 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.

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. We also analyze the communication time and the computation time for these four schemes combined with the binary-swap (BS) [19], the parallel-pipelined (PP) [13], and the rotate-tiling (RT) [17] data communication schemes. By combining the four data compression schemes and three data communication schemes, we have twelve image composition methods. In the experimental, four volume datasets are used as test samples. For each method, the data computation time and the data communication time are measured on an IBM SP2 parallel machine. The experimental results show that the TRLE data compression scheme with the RT data communication scheme outperforms other image composition methods for all test samples.

The rest of the paper is organized as follows. The TRLE scheme will be presented in Sect. 2. In Sect. 3, we will analyze the ranges of data compression ratio of the TRLE, the BR, the RLE, and the BRLC schemes. A generic image composition algorithm by combining the four data compression schemes with the three data communication schemes will be presented in Sect. 4. We will also analyze these twelve image composition algorithms in terms of the communication time and the computation time. In Sect. 5, the experimental results and performance analysis of these twelve image composition methods on an IBM SP2 parallel machine will be discussed.

2 The TRLE data compression scheme

The main idea of the TRLE scheme is trying to encode pixels according to the shapes of non-blank pixels. 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. The reason to choose a block with 2×2 pixels is that we want to use one-byte to encode a block. In a byte, we can use the lower 4-bit to represent 2×2 pixels and the higher 4-bit to represent the number of repetition of the block represented in the lower 4-bit. A block with

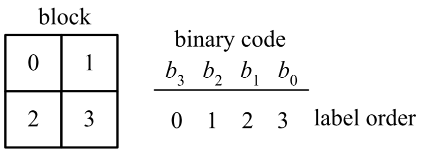


Fig. 6 Labels of pixels in a block

2×2 pixels has $2^4 = 16$ templates (the definition of template will be defined later). If a block with 3×3 pixels, 9-bit is needed to represent it. We need to use 2 bytes to encode a block. A block with 3×3 pixels has $2^9 = 512$ templates. The more the number of templates, the more time of the encoding/decoding of partial images.

Pixels in a block are labeled as shown in Fig. 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 sixteen 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 Fig. 7. In Fig. 7, white squares represent blank pixels while black squares represent non-blank pixels.

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.

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 Fig. 8. In Fig. 8, the *template_code* is "2A." It means that two consecutive blocks are the same block and are encoded by template "1010."

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. Each pixels consists of intensity and opacity. An example of *TRLE_code* is given in Fig. 9. In Fig. 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.

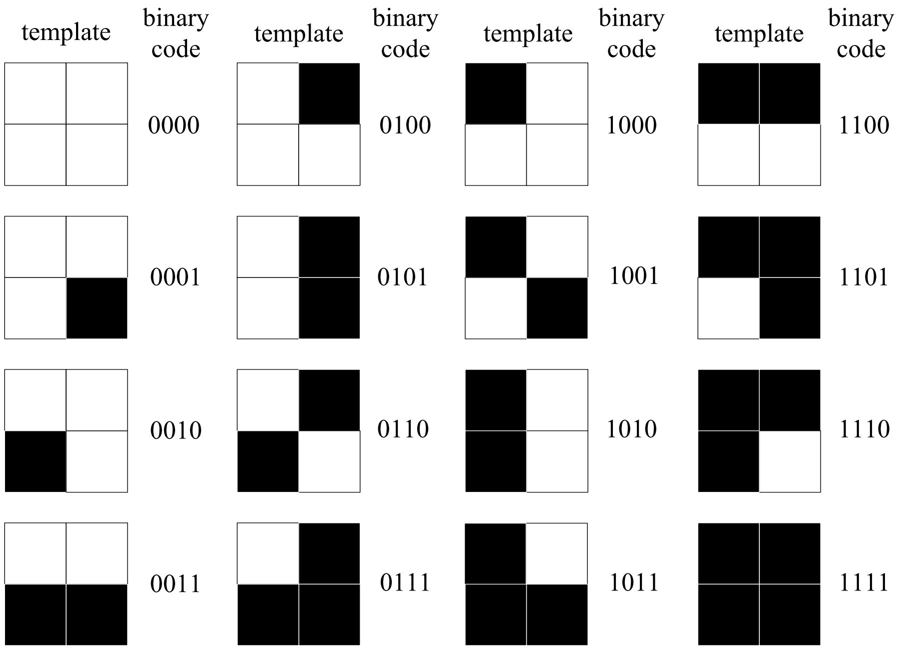


Fig. 7 4-bit binary codes of templates

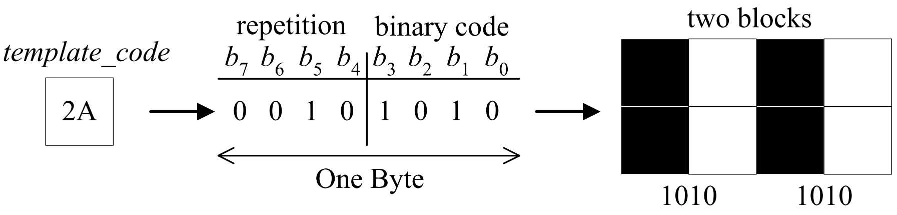


Fig. 8 An example of a *template_code*

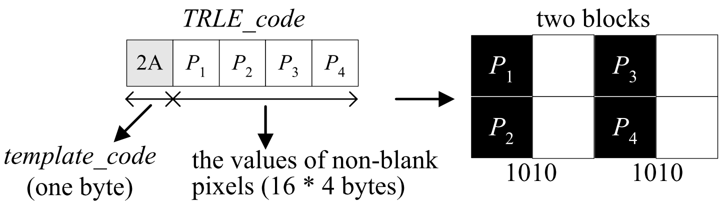


Fig. 9 An example of a *TRLE_code*

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

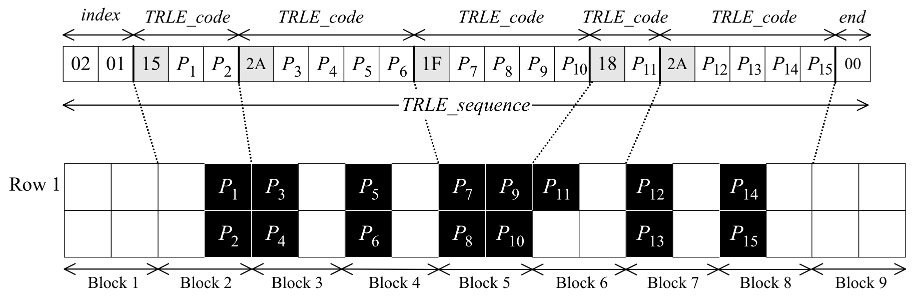


Fig. 10 An example of a *TRLE_sequence*

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 Fig. 10. For the *TRLE_sequence* shown in Fig. 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 *TELE_sequence* of a partial image.

An example of a *TRLE_packet* is given in Fig. 11. In Fig. 11, an image with 8×8 pixels that consists of 4×4 blocks is given. There are four *TRLE_sequences*. The *TRLE_packet* contains the four *TRLE_sequences*. Form the *TRLE_sequences* shown in Fig. 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_sequences*. For blank pixels inside the triangle, they are only encoded by *template_codes*. Their attributes are also filtered out from *TRLE_packet*. Therefore, in general, the TRLE scheme can have better compression ratio compared with the BR, the RLE, and the BRLC schemes.

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. The encoding and decoding algorithms are given as follows.

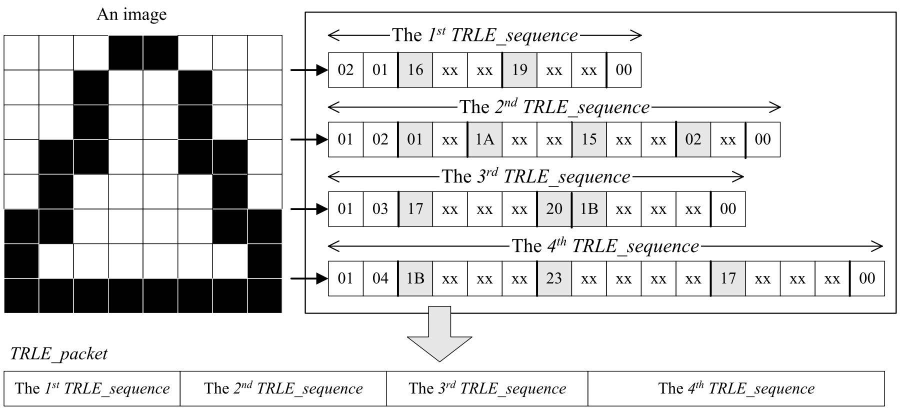


Fig. 11 An example of a *TRLE_packet*

```
Algorithm TRLE_Encode(A){
/* A is the partial image that has to encoding. */
/* packet is the TRLE_Packet for the non-blank pixels of a partial image. */
 1.    $x := 1, y := 1, packet := \varnothing$;
 2.    do {
 3.       To find the first non-blank pixel $P(x, y)$, and save the pixel's block
          value of $x$ and $y$ into packet;
 4.       do {
 5.          To find template of block of $P(x, y)$, $P(x+1, y)$, $P(x, y+1)$,
             and $P(x+1, y+1)$ and save the TRLE_code into packet;
 6.          $y := y + 1$;
 7.    } while $y < width(A)$;
 8.    To add a byte '00';
 9.    $x := x + 1$;
10. } while $x < height(A)$;
11. return packet;
12. }
end_of_TRLE_Encode
```

```
Algorithm TRLE_Decode(packet) {
/* packet is the TRLE_Packet for the non-blank pixels of a partial image*/
/* A is the partial image */
 1. $x := 0$;
 2. if (packet != $\varnothing$){
 3.    do {
 4.       read the two values of packet;
 5.       do {
 6.          read TRLE_code;
 7.          composite the non-blank pixels with the same position pixels of A;
 8.          $x := x + 1$;
9.        } while $packet(x)! = $ '00';
10.    $x := x + 1$;
11.    } while $packet(x)! = $ 'eof';
12.    return $A$;
13. }
end_of_TRLE_Decode
```

In algorithms *TRLE_Encode* and *TRLE_Decode*, 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 on distributed memory multicomputers 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.

- 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.
- C_{RLE}—The encoding code size of the RLE scheme.
- C_{BRLC}—The encoding code size of the BRLC scheme.
- C_{TRLE}—The encoding code size of the TRLE scheme.

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

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

In the BR scheme, the compression ratio is

$$CR(BR) = \frac{PA \times 16}{PA_{\text{BR}} \times 16 + 8}.$$

The worst case of the BR scheme is $PA_{\text{BR}} = PA$. The best case of the BR scheme is $PA_{\text{BR}} = PA_{\text{nb}}$. We have

$$\frac{PA \times 16}{PA \times 16 + 8} \le CR(BR) \le \frac{PA \times 16}{PA_{\text{nb}} \times 16 + 8}.$$

In the RLE scheme, the compression ratio is

$$CR(RLE) = \frac{PA \times 16}{PA_{\text{nb}} \times 16 + C_{\text{RLE}} \times 2 + \sqrt{PA} \times 2}.$$

The worst case of the RLE scheme is that the blank and non-blank pixels are interlaced in each row. C_{RLE} is $2 \times PA$. The best case of the RLE scheme is that all non-blank pixels form a square. $C_{\mathrm{RLE}} = \sqrt{PA_{\mathrm{nb}}} \times 2$. We have

$$\frac{PA \times 16}{PA_{\mathrm{nb}} \times 16 + PA \times 4 + \sqrt{PA} \times 2} \le CR(RLE) \le \frac{PA \times 16}{PA_{\mathrm{nb}} \times 16 + \sqrt{PA_{\mathrm{nb}}} \times 2 + \sqrt{PA} \times 2}.$$

In the BRLC scheme, the compression ratio is

$$CR(BRLC) = \frac{PA \times 16}{PA_{\mathrm{nb}} \times 16 + C_{\mathrm{BRLC}} \times 2 + \sqrt{PA_{\mathrm{BR}}} \times 2 + 8}.$$

The worst case of the BRLC scheme is that the blank and non-blank pixels are interlaced in each row. C_{BRLC} is $2 \times PA$. The best case of the BRLC scheme is that all non-blank pixels form a square. $C_{\mathrm{BRLC}} = \sqrt{PA_{\mathrm{nb}}} \times 2$. We have

$$\frac{PA \times 16}{PA_{\mathrm{nb}} \times 16 + PA \times 4 + \sqrt{PA} \times 2 + 8} \le CR(BRLC) \le \frac{PA \times 16}{PA_{\mathrm{nb}} \times 16 + \sqrt{PA_{\mathrm{nb}}} \times 4 + 8}.$$

In the TRLE scheme, the compression ratio is

$$CR(TRLE) = \frac{PA \times 16}{PA_{\mathrm{nb}} \times 16 + C_{\mathrm{TRLE}}}.$$

The worst case of the TRLE scheme is that any two consecutive blocks are encoded by different templates. C_{TRLE} is $PA/2 + \sqrt{PA} \times 3$. The best case of the TRLE scheme is that all non-blank pixel form a square. $C_{\mathrm{TRLE}} = \sqrt{PA_{\mathrm{nb}}} \times 4$. We have

$$\frac{PA \times 16}{PA_{\mathrm{nb}} \times 16 + PA/2 + \sqrt{PA} \times 3} \le CR(TRLE) \le \frac{PA \times 16}{PA_{\mathrm{nb}} \times 16 + \sqrt{PA_{\mathrm{nb}}} \times 4}.$$

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 Fig. 12. In Fig. 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.

4 Analysis of image composition with data compression schemes

To use a data compression scheme in the image composition stage of a parallel volume rendering system on distributed memory multicomputers, it needs to be combined with some data communication schemes. The following is a generic image

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

Method	Ranges
BR	$\frac{PA\times 16}{PA\times 16+8} \le CR(\text{BR}) \le \frac{PA\times 16}{PA_{\text{nb}}\times 16+8}$
RLE	$\frac{PA\times 16}{PA_{\text{nb}}\times 16+PA\times 4+\sqrt{PA}\times 2} \le CR(RLE) \le \frac{PA\times 16}{PA_{\text{nb}}\times 16+\sqrt{PA_{\text{nb}}}\times 2+\sqrt{PA}\times 2}$
BRLC	$\frac{PA\times 16}{PA_{\text{nb}}\times 16+PA\times 4+\sqrt{PA}\times 2+8} \le CR(BRLC) \le \frac{PA\times 16}{PA_{\text{nb}}\times 16+\sqrt{PA_{\text{nb}}}\times 4+8}$
TRLE	$\frac{PA\times 16}{PA_{\text{nb}}\times 16+PA/2+\sqrt{PA}\times 3} \le CR(TRLE) \le \frac{PA\times 16}{PA_{\text{nb}}\times 16+\sqrt{PA_{\text{nb}}}\times 4}$

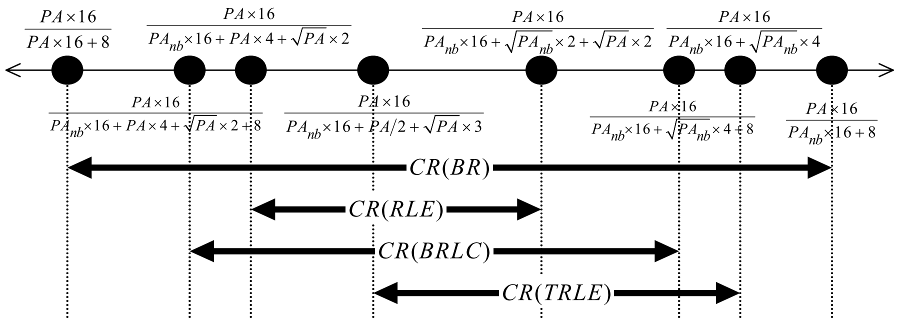

$\frac{PA\times 16}{PA\times 16+8}$
$\frac{PA\times 16}{PA_{nb}\times 16+PA\times 4+\sqrt{PA}\times 2}$
$\frac{PA\times 16}{PA_{nb}\times 16+\sqrt{PA_{nb}}\times 2+\sqrt{PA}\times 2}$
$\frac{PA\times 16}{PA_{nb}\times 16+\sqrt{PA_{nb}}\times 4}$
$\frac{PA\times 16}{PA_{nb}\times 16+PA\times 4+\sqrt{PA}\times 2+8}$
$\frac{PA\times 16}{PA_{nb}\times 16+PA/2+\sqrt{PA}\times 3}$
$\frac{PA\times 16}{PA_{nb}\times 16+\sqrt{PA_{nb}}\times 4+8}$
$\frac{PA\times 16}{PA_{nb}\times 16+8}$
$CR(BR)$
$CR(RLE)$
$CR(BRLC)$
$CR(TRLE)$

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

```
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
```

composition algorithm with a data compression scheme, where *compress*(A) is a function call to a data compression scheme for image A.

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_p—The data transmission time per byte.
- T_o—The computation time of the *over* operation per pixel.
- $T_{\text{comm}}(M)$—The total communication time of method M.
- $T_{\text{comp}}(M)$—The total computation time of method M.
- $T^k_{\text{comm}}(M, P_i)$—The communication time of P_i in the kth communication step of method M.
- $T^k_{\text{comp}}(M, P_i)$—The computation time of P_i in the kth communication step of method M.
- $T^k_{e\_c}(M, P_i)$—The data encoding time of P_i in the kth communication step of method M.
- $T^k_{d\_c}(M, P_i)$—The data decoding time of P_i in the kth communication step of method M.
- $A^k_i(M, P_i)$—The number of pixels sent/received by P_i in the kth communication step of method M.
- $A^{i,k}(M)$—The number of pixels of partial image of P_i in the kth communication step of method M.
- $A^{i,k}_{\text{BR}}(M)$—The number of pixels in a bounding rectangle of $A^{i,k}(M)$.
- $A^{i,k}_{\text{TRLE}}(M)$—The number of pixels encoded by the TRLE method of $A^{i,k}(M)$.
- $A^{i,k}_{\text{nb}}(M)$—The number of non-blank pixels of $A^{i,k}(M)$.
- $C^{i,k}_{\text{RLE}}(M)$—The number of the RLE encoding codes of $A^{i,k}(M)$.
- $C^{i,k}_{\text{BRLC}}(M)$—The number of the BRLC encoding codes of $A^{i,k}(M)$.
- $C^{i,k}_{\text{TRLE}}(M)$—The number of the TRLE encoding codes of $A^{i,k}(M)$.
- T_{BR}—The computation time for finding a bounding rectangle.
- T_{e_c}—The computation time of encoding a pixel.
- T_{d_c}—The computation time of decoding a pixel.

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

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

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

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

where δ^k_i is the number of processors that P_i sends data to in the kth 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, $T^k_{\text{comp}}(M)$ is defined as

$$T^k_{\text{comp}}(M, P_i) = T^k_{e\_c}(M, P_i) + T^k_{d\_c}(M, P_i) + A^k_i(M, P_i) \times T_o. \tag{3}$$

According to Eqs. (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 size of $A^k_i(M, P_i)$ and is important to an image composition method. In the following, we analyze the performance of these twelve image composition methods.

In the *BS* scheme, there are $\log P$ communication steps. In the kth communication step, the partial image size is $A/2^k$, where $k = 1, \ldots, \log P$. When the BR scheme is applied ($M = BS\_BR$), the size of $A^k_i(M, P_i)$ is $(A^{i,k}_{\text{BR}}(M))$. $T^k_{\text{comm}}(M, P_i)$ and $T^k_{\text{comp}}(M, P_i)$ are $T_s + (A^{i,k}_{\text{BR}}(M) \times 16 + 8) \times T_p$ and $T_{\text{BR}} + (A^{i,k}_{\text{BR}}(M)) \times T_o$, respectively. We have $T_{\text{comm}}(M) = \sum_{k=1}^{\log P}(T_s + MAX_{i=0}^{P-1}(A^{i,k}_{\text{BR}}(M) \times 16 + 8) \times T_p))$ and $T_{\text{comp}}(M) = \sum_{k=1}^{\log P}(T_{\text{BR}} + MAX_{i=0}^{P-1}(A^{i,k}_{\text{BR}}(M)) \times T_o)$.

When the RLE scheme is applied ($M = BS\_RLE$), the size of $A^k_i(M, P_i)$ is $(A^{i,k}(M))$. $T^k_{\text{comm}}(M, P_i)$ and $T^k_{\text{comp}}(M, P_i)$ are $T_s + (A^{i,k}_{\text{nb}}(M) \times 16 + C^{i,k}_{\text{BRLC}}(M) \times 2) \times T_p$ and $T_s + (A^{i,k}_{\text{nb}}(M) \times 16 + C^{i,k}_{\text{RLE}}(M) \times 2) \times T_o$, respectively. We have $T_{\text{comm}}(M) = \sum_{k=1}^{\log P}(T_s + MAX_{i=0}^{P-1}(A^{i,k}_{\text{nb}}(M) \times 16 + C^{i,k}_{\text{RLE}}(M) \times 2) \times T_p)$ and $T_{\text{comp}}(M) = \sum_{k-1}^{\log P} MAX_{i=0}^{P-1}(T_{e\_c} \times A^{i,k}(M) + T_{d\_c} \times C^{i,k}_{\text{RLE}}(M) + A^{i,k}(M) \times T_o)$.

When the BRLC scheme is applied ($M = BS\_BRLC$), the size of $A^k_i(M, P_i)$ is $MAX_{i=0}^{P-1}(A^{i,k}_{\text{BR}}(M))$. $T^k_{\text{comm}}(M, P_i)$ and $T^k_{\text{comp}}(M, P_i)$ are $MAX_{i=0}^{P-1}(A^{i,k}_{\text{nb}}(M) \times 16 + C^{i,k}_{\text{BRLC}}(M) \times 2 + 8)$ and $T_{\text{BR}} + MAX_{i=0}^{P-1}(T_{e\_c} \times A^{i,k}_{\text{BR}}(M) + T_{d\_c} \times C^{i,k}_{\text{BRLC}}(M) + A^{i,k}_{\text{BR}}(M) \times T_o)$, respectively. We have $T_{\text{comm}}(M) = \sum_{k=1}^{\log P}(T_s + MAX_{i=0}^{P-1}(A^{i,k}_{\text{nb}}(M) \times 16 + C^{i,k}_{\text{BRLC}}(M) \times 2 + 8) \times T_p)$ and $T_{\text{comp}}(M) = \sum_{k=1}^{\log P}(T_{\text{BR}} + MAX_{i=0}^{P-1}(T_{e\_c} \times A^{i,k}_{\text{BR}}(M) + T_{d\_c} \times C^{i,k}_{\text{BRLC}}(M) + A^{i,k}_{\text{BR}}(M) \times T_o))$.

When the TRLE scheme is applied ($M = BS\_TRLE$), the size of $A^k_i(M, P_i)$ is $MAX_{i=0}^{P-1}(A^{i,k}_{\text{TRLE}}(M))$. $T^k_{\text{comm}}(M, P_i)$ and $T^k_{\text{comp}}(M, P_i)$ are $T_s + MAX_{i=0}^{P-1}(A^{i,k}_{\text{nb}}(M) \times 16 + C^{i,k}_{\text{TRLE}}(M) \times T_p)$ and $MAX_{i=0}^{P-1}(T_{e\_c} \times A^{i,k}(M) + T_{d\_c} \times C^{i,k}_{\text{TRLE}}(M) + A^{i,k}_{\text{TRLE}}(M) \times T_o)$, respectively. $A^{i,k}_{\text{TRLE}}(M)$ may contain both blank and non-blank pixels. However, the attributes (intensity and opacity) of blank pixels will be filtered out when blank pixels are encoded as *TRLE_code*, i.e., only attributes of non-blank pixels are encoded in the TRLE scheme. We have $T_{\text{comm}}(M) = \sum_{k=1}^{\log P}(T_s + MAX_{i=0}^{P-1}(A^{i,k}_{\text{nb}}(M) \times 16 + C^{i,k}_{\text{TRLE}}(M)) \times T_p)$ and $T_{\text{comp}}(M) = \sum_{k=1}^{\log P} MAX_{i=0}^{P-1}(T_{e\_c} \times A^{i,k}(M) + T_{d\_c} \times C^{i,k}_{\text{TRLE}}(M) + A^{i,k}_{\text{TRLE}}(M) \times T_o)$. The data communication time and the data computation time of the four image composition methods are summarized in Table 2.

In the PP scheme, there are $(P - 1)$ communication steps. In the kth communication step, the partial image size is A/P. When the BR, the RLE, the BRLC, and the TRLE schemes are applied, we have similar analysis as those for the BS scheme. The data communication time and the data computation time of the four image composition methods are summarized in Table 3.

Table 2 Theoretical time of the BS scheme with four data compression schemes

Method	Time
BS_BR	$T_{\mathrm{comm}}(M)=\sum_{k=1}^{\log P}\big(T_s+MAX_{i=0}^{P-1}\big(A_{\mathrm{BR}}^{i,k}(M)\times 16+8\big)\times T_p\big)$
	$T_{\mathrm{comp}}(M)=\sum_{k=1}^{\log P}\big(T_{\mathrm{BR}}+MAX_{i=0}^{P-1}\big(A_{\mathrm{BR}}^{i,k}(M)\big)\times T_o\big)$
BS_RLE	$T_{\mathrm{comm}}(M)=\sum_{k=1}^{\log P}\big(T_s+MAX_{i=0}^{P-1}\big(A_{\mathrm{nb}}^{i,k}(M)\times 16+C_{\mathrm{RLE}}^{i,k}(M)\times 2\big)\times T_p\big)$
	$T_{\mathrm{comp}}(M)=\sum_{k-1}^{\log P}MAX_{i=0}^{P-1}\big(A^{i,k}(M)T_{e\_c}+C_{\mathrm{RLE}}^{i,k}(M)T_{d\_c}+A^{i,k}(M)T_o\big)$
BS_BRLC	$T_{\mathrm{comm}}(M)=\sum_{k=1}^{\log P}\big(T_s+MAX_{i=0}^{P-1}\big(A_{\mathrm{nb}}^{i,k}(M)\times 16+C_{\mathrm{BRLC}}^{i,k}(M)\times 2+8\big)\times T_p\big)$
	$T_{\mathrm{comp}}(M)=\sum_{k=1}^{\log P}\big(T_{\mathrm{BR}}+MAX_{i=0}^{P-1}\big(A_{\mathrm{BR}}^{i,k}(M)T_{e\_c}+C_{\mathrm{BRLC}}^{i,k}(M)T_{d\_c}+A_{\mathrm{BR}}^{i,k}(M)T_o\big)\big)$
BS_TRLE	$T_{\mathrm{comm}}(M)=\sum_{k=1}^{\log P}\big(T_s+MAX_{i=0}^{P-1}\big(A_{\mathrm{nb}}^{i,k}(M)\times 16+C_{\mathrm{TRLE}}^{i,k}(M)\big)\times T_p\big)$
	$T_{\mathrm{comp}}(M)=\sum_{k=1}^{\log P}MAX_{i=0}^{P-1}\big(A^{i,k}(M)T_{e\_c}+C_{\mathrm{TRLE}}^{i,k}(M)T_{d\_c}+A_{\mathrm{TRLE}}^{i,k}(M)T_o\big)$

Table 3 Theoretical time of the PP scheme with four data compression schemes

Method	Time
PP_BR	$T_{\mathrm{comm}}(M)=\sum_{k=1}^{P-1}\big(T_s+MAX_{i=0}^{P-1}\big(A_{\mathrm{BR}}^{i,k}(M)\times 16+8\big)\times T_p\big)$
	$T_{\mathrm{comp}}(M)=\sum_{k=1}^{P-1}\big(T_{\mathrm{BR}}+MAX_{i=0}^{P-1}\big(A_{\mathrm{BR}}^{i,k}(M)\big)\times T_o\big)$
PP_RLE	$T_{\mathrm{comm}}(M)=\sum_{k=1}^{P-1}\big(T_s+MAX_{i=0}^{P-1}\big(A_{\mathrm{nb}}^{i,k}(M)\times 16+C_{\mathrm{RLE}}^{i,k}(M)\times 2\big)\times T_p\big)$
	$T_{\mathrm{comp}}(M)=\sum_{k=1}^{P-1}MAX_{i=0}^{P-1}\big(A^{i,k}(M)T_{e\_c}+C_{\mathrm{RLE}}^{i,k}(M)T_{d\_c}+A^{i,k}(M)T_o\big)$
PP_BRLC	$T_{\mathrm{comm}}(M)=\sum_{k=1}^{P-1}\big(T_s+MAX_{i=0}^{P-1}\big(A_{\mathrm{nb}}^{i,k}(M)\times 16+C_{\mathrm{BRLC}}^{i,k}(M)\times 2+8\big)\times T_p\big)$
	$T_{\mathrm{comp}}(M)=\sum_{k=1}^{P-1}\big(T_{\mathrm{BR}}+MAX_{i=0}^{P-1}\big(A_{\mathrm{BR}}^{i,k}(M)T_{e\_c}+C_{\mathrm{BRLC}}^{i,k}(M)T_{d\_c}+A_{\mathrm{BR}}^{i,k}(M)T_o\big)\big)$
PP_TRLE	$T_{\mathrm{comm}}(M)=\sum_{k=1}^{P-1}\big(T_s+MAX_{i=0}^{P-1}\big(A_{\mathrm{nb}}^{i,k}(M)\times 16+C_{\mathrm{TRLE}}^{i,k}(M)\big)\times T_p\big)$
	$T_{\mathrm{comp}}(M)=\sum_{k=1}^{P-1}MAX_{i=0}^{P-1}\big(A^{i,k}(M)T_{e\_c}+C_{\mathrm{TRLE}}^{i,k}(M)T_{d\_c}+A_{\mathrm{TRLE}}^{i,k}(M)T_o\big)$

In the RT scheme, there are $\lceil \log P\rceil$ communication steps. In the first communication step ($k=1$), the maximum number of send/receive operations performed by processors is $\lceil \frac{N}{P}\rceil$. The block size in each sent or received by a processor is $\frac{A}{N}$. The maximum data communication and computation time among processors are $\lceil \frac{N}{P}\rceil \times T_s + \lceil \frac{N}{P}\rceil \frac{A}{N} \times T_p$ and $\lceil \frac{N}{P}\rceil \frac{A}{N} \times T_o$, respectively. In the kth communication step, where $k>1$, the maximum number of send/receive operations performed by processors is $\lceil \frac{2B_k}{P}\rceil$, where

$$B_k=\begin{cases} N & \text{for } k=1\\ B_{k-1}-\lfloor B_{k-1}/P\rfloor & \text{for } k>1. \end{cases} \tag{4}$$

The block size in each sent or received by a processor is $\frac{A}{2^{k-1}N}$, where $k=2,\ldots,\lceil \log P\rceil$. The maximum data communication and computation time among

Table 4 Theoretical time of the RT scheme with four data compression schemes

Method	Time
RT_BR	$T_{\text{comm}}(M)=\sum_{k=1}^{\lceil \log P\rceil}\left(\left(\left\lceil\frac{2B_k}{P}\right\rceil+\left\lceil\frac{N}{P}\right\rceil-\left\lceil\frac{2N}{P}\right\rceil\right)\left(T_s+\frac{1}{N}\times MAX_{i=0}^{P-1}(A_{\text{BR}}^{i,k}(M)\times 16+8)\times T_p\right)\right)$
	$T_{\text{comp}}(M)=\sum_{k=1}^{\lceil \log P\rceil}\left(T_{\text{BR}}+\frac{1}{N}\left(\left\lceil\frac{2B_k}{P}\right\rceil+\left\lceil\frac{N}{P}\right\rceil-\left\lceil\frac{2N}{P}\right\rceil\right)\times MAX_{i=0}^{P-1}(A_{\text{BR}}^{i,k}(M))\times T_o\right)$
RT_RLE	$T_{\text{comm}}(M)=\sum_{k=1}^{\lceil \log P\rceil}\left(\left(\left\lceil\frac{2B_k}{P}\right\rceil+\left\lceil\frac{N}{P}\right\rceil-\left\lceil\frac{2N}{P}\right\rceil\right)\left(T_s+\frac{1}{N}MAX_{i=0}^{P-1}(A_{\text{nb}}^{i,k}(M)\times 16+C_{\text{RLE}}^{i,k}(M)\times 2)\times T_p\right)\right)$
	$T_{\text{comp}}(M)=\sum_{k=1}^{\lceil \log P\rceil}\left(\left\lceil\frac{2B_k}{P}\right\rceil+\left\lceil\frac{N}{P}\right\rceil-\left\lceil\frac{2N}{P}\right\rceil\right)\frac{1}{N}MAX_{i=0}^{P-1}(A^{i,k}(M)T_{e\_c}+C_{\text{RLE}}^{i,k}(M)T_{d\_c}+A_{\text{nb}}^{i,k}(M)T_o)$
RT_BRLC	$T_{\text{comm}}(M)=\sum_{k=1}^{\lceil \log P\rceil}\left(\left(\left\lceil\frac{2B_k}{P}\right\rceil+\left\lceil\frac{N}{P}\right\rceil-\left\lceil\frac{2N}{P}\right\rceil\right)\left(T_s+\frac{1}{N}MAX_{i=0}^{P-1}(A_{\text{nb}}^{i,k}(M)\times 16\right.\right.$ $\left.\left.+C_{\text{BRLC}}^{i,k}(M)\times 2+8)\times T_p\right)\right)$
	$T_{\text{comp}}(M)=\sum_{k=1}^{\lceil \log P\rceil}\left(T_{\text{BR}}+\frac{1}{N}\left(\left\lceil\frac{2B_k}{P}\right\rceil+\left\lceil\frac{N}{P}\right\rceil-\left\lceil\frac{2N}{P}\right\rceil\right)MAX_{i=0}^{P-1}(A_{\text{BR}}^{i,k}(M)T_{e\_c}\right.$ $\left.+C_{\text{BRLC}}^{i,k}(M)T_{d\_c}+A_{\text{nb}}^{i,k}(M)T_c)\right)$
RT_TRLE	$T_{\text{comm}}(M)$ $=\sum_{k=1}^{\lceil \log P\rceil}\left(\left\lceil \log N\right\rceil T_s+\frac{\lceil \log N\rceil}{N}\times MAX_{i=0}^{P-1}(A_{\text{nb}}^{i,k}(M)\times 16+CODE_{\text{TRLE}}^{i,k}(M))\right)\times T_c$
	$T_{\text{comp}}(M)=\sum_{k=1}^{\lceil \log P\rceil}\frac{1}{N}\left(\left\lceil\frac{2B_k}{P}\right\rceil+\left\lceil\frac{N}{P}\right\rceil-\left\lceil\frac{2N}{P}\right\rceil\right)MAX_{i=0}^{P-1}(A^{i,k}(M)T_{e\_c}+C_{\text{TRLE}}^{i,k}(M)T_{d\_c}$ $+A_{\text{TRLE}}^{i,k}(M)T_o)$

processors are $\lceil\frac{2B_k}{P}\rceil T_s + \lceil\frac{2B_k}{P}\rceil\frac{A}{N2^{k-1}}T_p$ and $\lceil\frac{2B_k}{P}\rceil\frac{A}{N2^{k-1}}T_o$, respectively. When the BR, the RLE, the BRLC, and the TRLE schemes are applied, we have similar analysis as those for the BS scheme. The data communication time and the data computation time of the four image composition methods are summarized in Table 4.

5 Experimental results and performance analysis

To evaluate the performance of the TRLE scheme, we compare the TRLE scheme with the BR, the RLE, and the BRLC schemes on an IBM SP2 parallel machine [7]. The IBM SP2 parallel machine is located at National Center of High Performance Computing (NCHC) in Taiwan. The IBM SP2 parallel machine is a super-scalar architecture that it uses IBM RISC System/6000 POWER2 SuperChips (P2SCs) with clock rate of 120 and 140 MHz. There are 110 IBM POWER2 CPUs in this machine, and each CPU has a 128 KB first-level data cache, a 32 KB first-level instruction cache, and 256 MB or 512 MB of memory space. Each node is connected to a low-latency, high-bandwidth interconnection network called High Performance Switch (HPS).

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 2-D partitioning scheme [16] to distribute a volume dataset to processors. In the data render stage, each processor uses the shear-warp factorization [9–12] volume rendering method to generate a partial image. In the image composition stage, the twelve image composition methods are used to composite partial images. We use C and MPICH [20] message passing libraries to implement the data compression schemes.

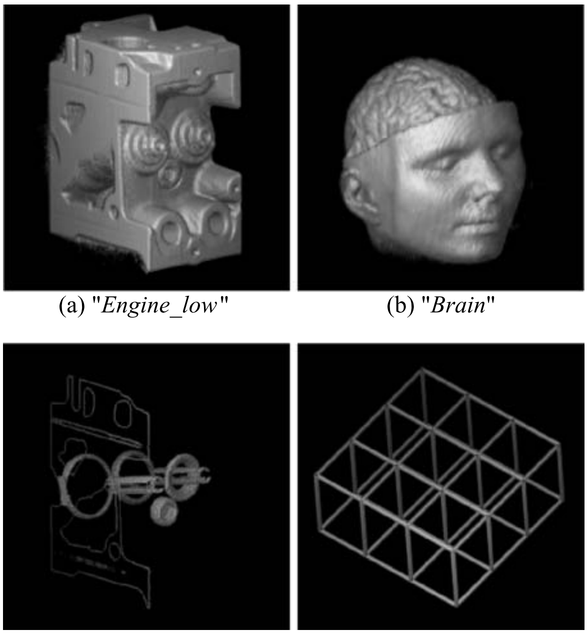


Fig. 13 The final images of the four test samples

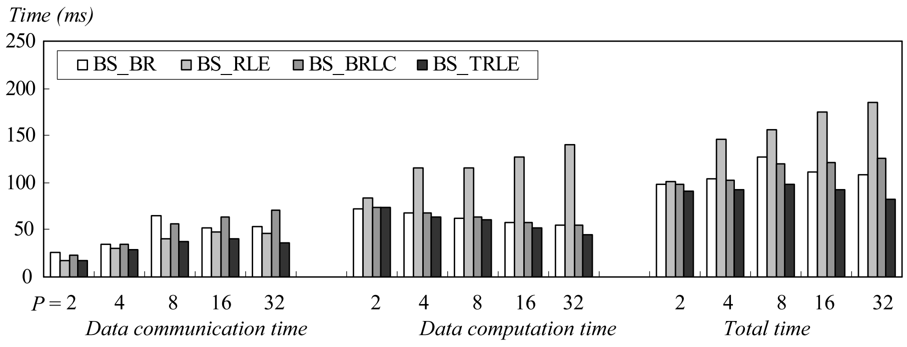
(a) "*Engine_low*" (b) "*Brain*"

(c) "*Engine_high*" (d) "*Cube*"

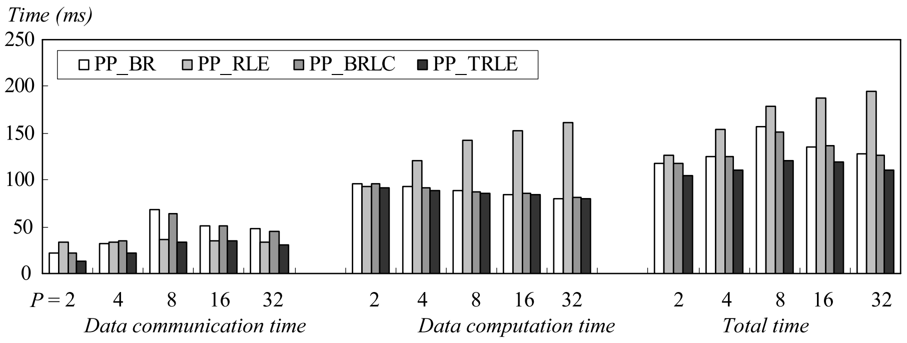
Since the compression ratio of a data compress scheme is affected by the shape of an image, two kinds of image shapes, *concentrated* and *scatter*, are used to evaluate the performance of these data compression schemes. For the concentrated shape, all the non-blank pixels fill out one block of area of an image. Shapes not in the concentrated category are scatter shapes. Four volume datasets are used as test samples. The first test sample is an "*Engine_low*" dataset, which is the CT scan of an engine block and the dimensions of the dataset is $256 \times 256 \times 110$. Each voxel in "*Engine_low*" consists of grayscale intensity. The second test sample is a "*Brain*" dataset generated from the MR scan of a human brain, and the dimensions of the dataset is $256 \times 256 \times 225$. Each voxel in "*Brain*" consists of grayscale intensity. The images of "*Engine_low*" and "*Brain*" datasets belong to the concentrate shape category. The third test sample is an "*Engine_high*" dataset, which is obtained by extracting those voxels whose intensity is greater than 180 from "*Engine_low*". The fourth test sample is a "*Cube*" dataset generated from the CT scan of nine combined cubes, and the dimensions of the dataset is $256 \times 256 \times 110$. Each voxel of "*Cube*" consists of grayscale intensity, and the value is 180. The images of "*Engine_high*" and "*Cube*" datasets belong to the scatter shape category. Figure 13 shows the final images of the four test samples. Each image is grayscale color and contains 512×512 pixels.

5.1 Performance results for the concentrate shape

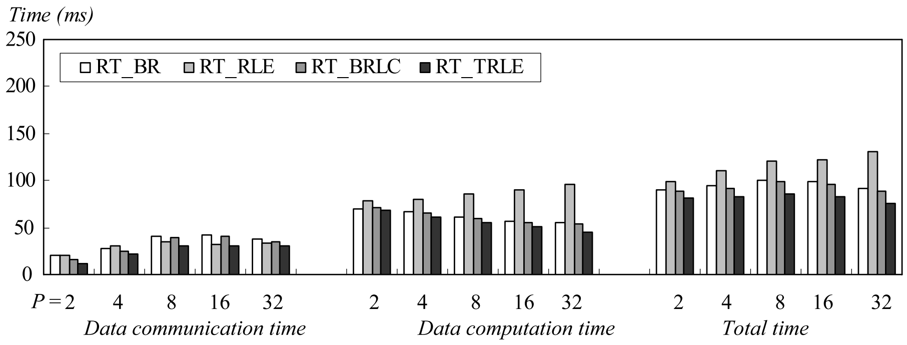
Figure 14 shows the data communication time, the data computation time, and the total image composition time of the twelve image composition methods for the "*Engine_low*" dataset on an IBM SP2 parallel machine. Figure 14a shows the results for the BS scheme. For a concentrated shape, in general, the RLE scheme or the BRLC



(a) The BS scheme



(b) The PP scheme



(c) The RT scheme

Fig. 14 The image composition time for "*Engine_low*"

should be sufficient. In Fig. 14a, the order of the data communication time, in general, is $T_{\text{comm}}(BS\_TRLE) < T_{\text{comm}}(BS\_RLE) < T_{\text{comm}}(BS\_BRLC) < T_{\text{comm}}(BS\_BR)$. We can see that $T_{\text{comm}}(BS\_TRLE)$ is the smallest of the four image composition methods. This implies that the TRLE scheme can achieve a better compression ratio than other three compression schemes for the test sample "*Engine_low*". The reason is that the TRLE scheme only encodes non-blank pixels while other schemes may encode both blank and non-blank pixels.

In Fig. 14a, the order of the data computation time, in general, is $T_{\mathrm{comp}}(BS\_TRLE) < T_{\mathrm{comp}}(BS\_BRLC) < T_{\mathrm{comp}}(BS\_BR) < T_{\mathrm{comp}}(BS\_RLE)$. We can see that $T_{\mathrm{comp}}(BS\_TRLE)$ is the smallest of the four image composition methods. The reason is that, in the TRLE scheme, only non-blank pixels are encoded. For other schemes, both blank and non-blank pixels are encoded. The time spent on the *over* operations in the TRLE scheme is the smallest. Since the encoding/decoding time is small compare to the time of *over* operations, $T_{\mathrm{comp}}(BS\_TRLE)$ is the smallest in this case. Since $T_{\mathrm{comm}}(BS\_TRLE)$ and $T_{\mathrm{comp}}(BS\_TRLE)$ are the smallest, $T_{\mathrm{total}}(BS\_TRLE)$ is the smallest as well.

In Fig. 14a, the order of the total composition time is $T_{\mathrm{total}}(BS\_TRLE) < T_{\mathrm{total}}(BS\_BRLC) < T_{\mathrm{total}}(BS\_BR) < T_{\mathrm{total}}(BS\_RLE)$. Figure 14b and Fig. 14c show the results for the PP and the RT schemes, respectively. From Fig. 14b and Fig. 14c, we have similar observations as those in Fig. 14a.

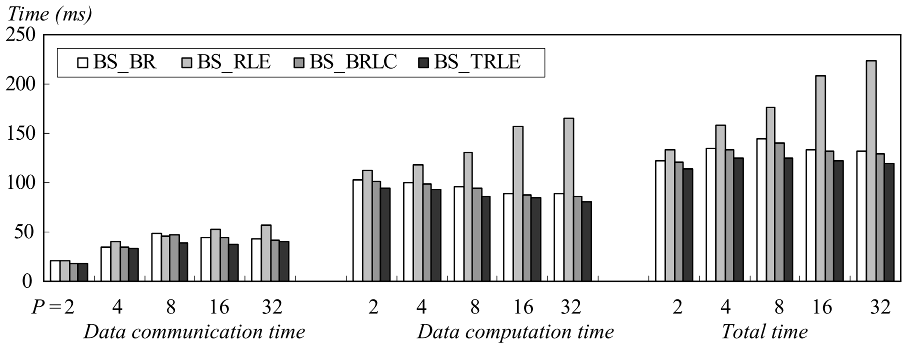
Figure 15 shows the data communication time, the data computation time, and the total image composition time of the twelve image composition methods for the "*Brain*" datasets on an IBM SP2 parallel machine. From Fig. 15, we have similar observations as those in Fig. 14.

From Figs. 14 and 15, we can see that, in general, the RLE scheme has smaller data communication time but the highest data computation time. The BRLC scheme has smaller data computation time but higher data communication time. Only the proposed scheme produces the smallest data communication and computation time. This implies that the encoding/decoding method used in the TRLE scheme can produce good data compression ratio and take less data computation time for the concentrated shape.

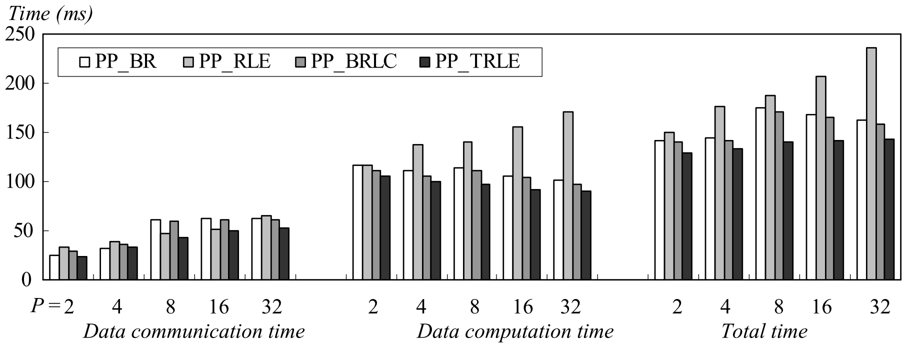
5.2 Performance results for the scatter shape

Figure 16 shows the data communication time, the data computation time, and the total image composition time of the twelve image composition methods for the "*Engine_high*" datasets on an IBM SP2 parallel machine. Figure 16a shows the results for the BS scheme. In Fig. 16a, the order of the data communication time, in general, is $T_{\mathrm{comm}}(BS\_TRLE) < T_{\mathrm{comm}}(BS\_RLE) < T_{\mathrm{comm}}(BS\_BR) < T_{\mathrm{comm}}(BS\_BRLC)$. We can see that $T_{\mathrm{comm}}(BS\_TRLE)$ is the smallest of the four image composition methods. This implies that the TRLE scheme can achieve a better compression ratio than other three compression schemes for the test sample "*Engine_high*". There are two reasons. First, for the scatter shape, the TRLE scheme may encode both blank and non-blank pixels as others. However, the TRLE scheme encodes less blank pixels than other schemes since the TRLE method can encode an image according to the outer shapes of objects in an image. Second, the attributes (intensity and opacity) of blank pixels will be filtered out when blank pixels are encoded as *TRLE_code*, i.e., only attributes of non-blank pixels are encoded in the TRLE scheme. This will reduce the encoded code size.

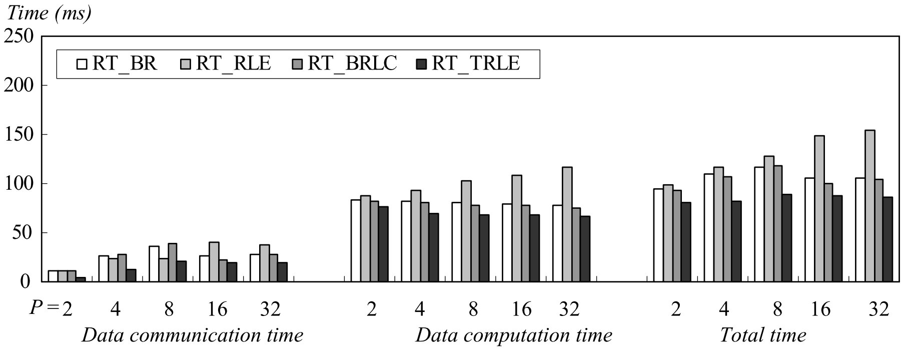
In Fig. 16a, the order of the data computation time, in general, is $T_{\mathrm{comp}}(BS\_TRLE) < T_{\mathrm{comp}}(BS\_BRLC) < T_{\mathrm{comp}}(BS\_BR) < T_{\mathrm{comp}}(BS\_RLE)$. We can see that $T_{\mathrm{comp}}(BS\_TRLE)$ is the smallest of the four image composition methods. The reason is that the TRLE scheme encodes less blank pixels than others. The time



(a) The BS scheme



(b) The PP scheme

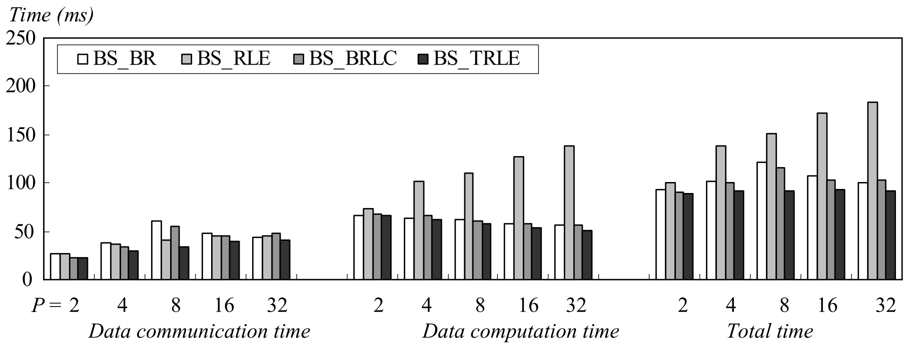


(c) The RT scheme

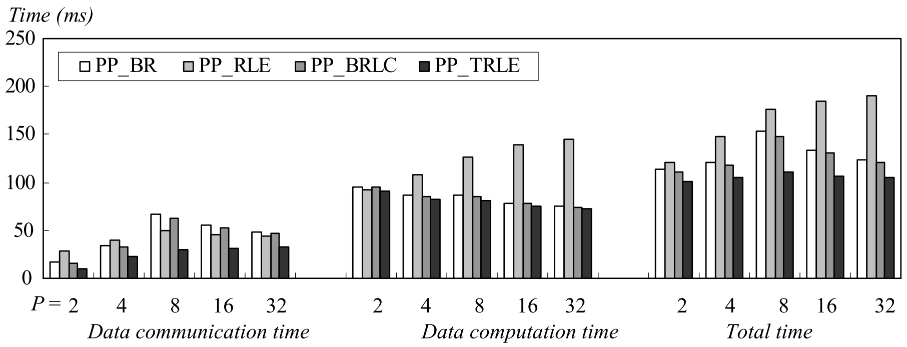
Fig. 15 The image composition time for "*Brain*"

spent on the *over* operations in the TRLE scheme is the smallest. Since the encoding/decoding time is small compare to the time of *over* operations, $T_{\text{comp}}(BS\_TRLE)$ is the smallest in this case. Since $T_{\text{comm}}(BS\_TRLE)$ and $T_{\text{comp}}(BS\_TRLE)$ are the smallest, $T_{\text{total}}(BS\_TRLE)$ is the smallest as well.

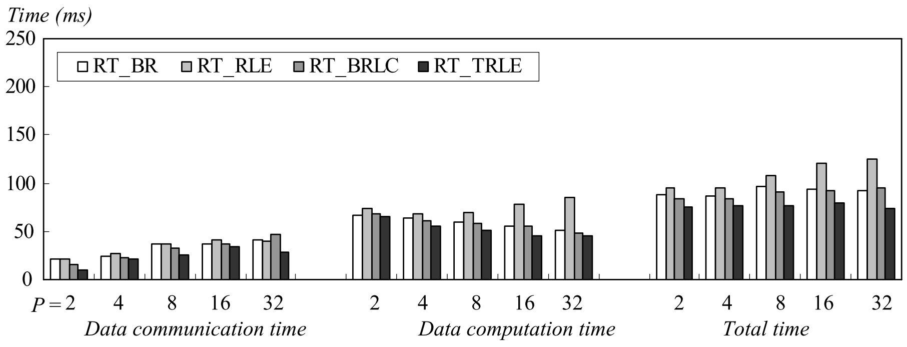
In Fig. 16a, the order of the total composition time is $T_{\text{total}}(BS\_TRLE) < T_{\text{total}}(BS\_BRLC) < T_{\text{total}}(BS\_BR) < T_{\text{total}}(BS\_RLE)$. Figure 16b and Fig. 16c show



(a) The BS scheme



(b) The PP scheme

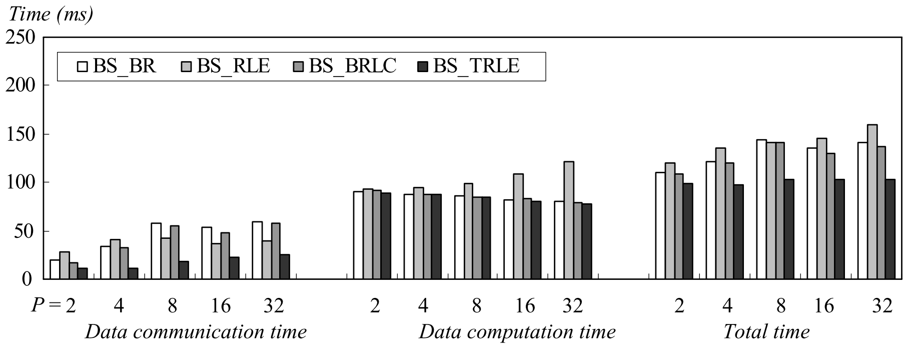


(c) The RT scheme

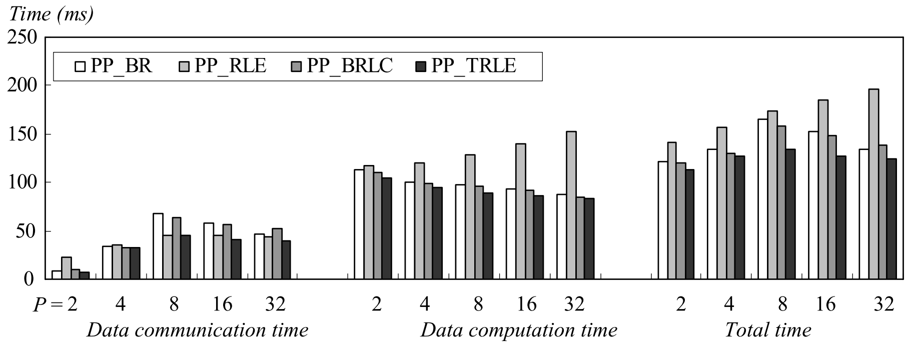
Fig. 16 The image composition time for "*Engine_high*"

the results for the PP and the RT schemes, respectively. From Fig. 16b and Fig. 16c, we have similar observations as those in Fig. 16a.

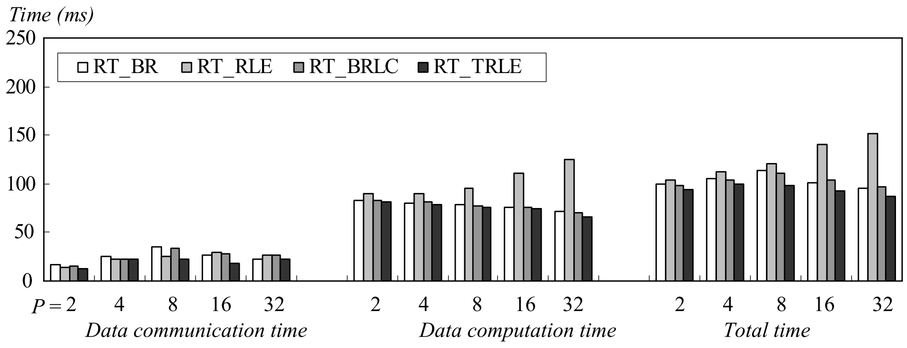
Figure 17 shows the data communication time, the data computation time, and the total image composition time of the twelve image composition methods for "*Cube*" datasets on an IBM SP2 parallel machine. From Fig. 17, we have similar observations as those in Fig. 16.



(a) The BS scheme



(b) The PP scheme



(c) The RT scheme

Fig. 17 The image composition time for "*Cube*" and the final image size is 512×512

From Figs. 16 and 17, we can see that, in general, the RLE scheme has smaller data communication time but the highest data computation time. The BRLC scheme has smaller data computation time but higher data communication time. Only the proposed scheme produces the smallest data communication and computation time. This implies that the encoding/decoding method used in the TRLE scheme can produce good data compression ratio and take less data computation time for the scatter shape as well.

6 Conclusions

In this paper, we have proposed an efficient data compression scheme, the template run-length encoding scheme, for image composition of parallel volume rendering on distributed memory multicomputers. The main idea of the TRLE scheme is to encode an image according to the outer shapes of objects in the image. To evaluate the performance of the TRLE scheme, we compared the proposed scheme with the BR, the RLE, and the BRLC schemes. Both theoretical and experimental analyses were conducted. For the theoretical analysis, we compared 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), parallel-pipelined (PP), and rotate-tiling (RT) data communication schemes. The data computation time and the data communication time were measured on an IBM SP2 parallel machine. Four volume datasets were used as test samples. From the experimental results, we had the following remarks.

Remark 1 The *RT_TRLE* method has the best performance among the twelve image composition methods in terms of the data communication time, the data computation time, and the image composition time.

Remark 2 The TRLE scheme, in general, can achieve a better compression ratio than other three compression schemes for images that are either concentrated shape or scatter shape.

References

1. Ahrens J, Painter J (1998) Efficient sort-last rendering using compression-based image compositing. In: Proceedings of the second eurographics workshop on parallel graphics and visualization, 1998, pp 145–151
2. Cox M, Hanrahan P (1993) Pixel merging for object-parallel rendering: a distributed snooping algorithm. In: Proceedings of 1993 parallel rendering symposium (PRS'93), San Jose, Oct 1993, pp 49–56
3. Drebin RA, Carpenter L, Hanrahan P (1988) Volume rendering. In: Proceedings of SIGGRAPH'88 22(4):65–74, Atlanta
4. Foley JD, Van Dam A, Feiner SK, Hughes JF (1990) Computer graphics: principles and practice, 2nd edn. Addison-Wesley, Reading
5. Groeller E, Purgathofer W (1995) Coherence in computer graphics. Technical Reports TR-186-2-95-04, Institute of Computer Graphics 186-2, Technical University of Vienna, March 1995
6. Hsu WM (1993) Segmented ray casting for data parallel volume rendering. In: Proceedings of 1993 parallel rendering symposium (PRS'93), San Jose, USA, Oct 1993, pp 7–14
7. IBM, IBM AIX parallel environment, Parallel Programming Subroutine Reference
8. Kaufman A (1991) Volume visualization. IEEE Computer Society Press
9. Lacroute P (1995) Fast volume rendering using a shear-warp factorization of the viewing transformation. PhD dissertation, Stanford University
10. Lacroute P (1995) Real-time volume rendering on shared memory multiprocessors using the shear-warp factorization. In: Proceedings of 1995 parallel rendering symposium (PRS'95), Atlanta, Oct 1995, pp 15–22

11. Lacroute P (1996) Analysis of a parallel volume rendering system based on the shear-warp factorization. IEEE Trans Vis Comput Graph 2(3):218–231
12. Lacroute P, Levoy M (1994) Fast volume rendering using a shear-warp factorization of the viewing transformation. In: Proceedings of SIGGRAPH'94, Orlando, July 1994, pp 451–458
13. Lee TY, Raghavendra CS, Nicholas JB (1996) Image composition schemes for sort-last polygon rendering on 2D mesh multicomputers. IEEE Trans Vis Comput Graph 2(3):202–217
14. Laur D, Hanrahan P (1991) Hierarchical splatting: a progressive refinement algorithm for volume rendering. In: Proceedings of SIGGRAPH'91, Las Vegas, July 1991, vol 25, pp 285–288
15. Levoy M (1990) Efficient ray tracing of volume data. ACM Trans Graph 9(3):245–261
16. Lin CF, Chung YC, Yang DL (2002) Parallel shear-warp factorization volume rendering using efficient 1-D and 2-D partitioning schemes on distributed memory multicomputers. J Supercomput 22(3):277–302
17. Lin CF, Liao SK, Chung YC, Yang DL (2004) A rotate-tiling image composition method for parallel volume rendering on distributed memory multicomputers. J Inf Sci Eng 20(4):643–664
18. Ma KL, Painter JS, Hansen CD, Krogh MF (1993) A data distributed, parallel algorithm for ray-traced volume rendering. In: Proceedings of 1993 parallel rendering symposium (PRS'93), San Jose, Oct 1993, pp 15–22
19. Ma KL, Painter JS, Hansen CD, Krogh MF (1994) Parallel volume rendering using binary-swap composition. IEEE Comput Graph Appl 14(4):59–68
20. MPI Forum. MPI: A Message-Passing Interface Standard, May 1994
21. Porter T, Duff T (1984) Composition digital images. In: Proceedings of SIGGRAPH'84, Jul 1984, vol 18, pp 253–259
22. Sano K, Kitajima H, Kobayasi H, Nakamura T (1997) Parallel processing of the shear-warp factorization with the binary-swap scheme on a distributed-memory multiprocessor system. In: Proceedings of 1997 parallel rendering symposium (PRS'97), Oct 1997, pp 87–94
23. Singh JP, Gupta A, Levoy M (1994) Parallel visualization algorithms: performance and architectural implications. Comput 27(7):45–55
24. Upson C, Keeler M (1988) V-BUFFER: visible volume rendering. In: Proceedings of SIGGRAPH'88, Atlanta, 1988, vol 22, issue 4, pp 59–64
25. Westover L (1990) Footprint evaluation for volume rendering. In: Proceedings of SIGGRAPH'90, Dallas, 1990, vol 24, pp 367–376
26. Wilhelms J, Van Gelder A (1991) A coherent projection approach for direct volume rendering. In: Proceedings of SIGGRAPH'91, Jul 1991, vol 25, issue 4, pp 275–283
27. Wittenbrink CM, Somani AK (1993) Permutation warping for data parallel volume rendering. In: Proceedings of 1993 parallel rendering symposium (PRS'93), San Jose, USA, Oct 1993, pp 57–60
28. Wittenbrink CM (1998) Extensions to permutation warping for parallel volume rendering. Parallel Comput 24(9–10):1385–1406
29. Yoo TS, Neumann U, Fuchs H, Pizer SM, Cullip T, Rhoades J, Whitaker R (1992) Direct visualization of volume data. IEEE Comput Graph Appl 12(4):63–71
30. Yang DL, Yu JC, Chung YC (2001) Efficient compositing schemes for the sort-last-sparse parallel volume rendering system on distributed memory multicomputers. J Supercomput 18(2):201–220

Chin-Feng Lin received his B.S. and Ph.D. degrees in computer science from Feng Chia University in 1996 and 2004, respectively. He joined the department of Information Management at Chang Jung Christian University as an assistant professor in 2004. His research interests include data visualization, parallel and distributed processing, high performance computing, parallel rendering, grid computing, virtual reality, and high level architecture. He is a member of IEEE computer society and ACM.

Yeh-Ching Chung received a B.S. degree in Information Engineering from Chung Yuan Christian University in 1983, and the M.S. and Ph.D. degrees in Computer and Information Science from Syracuse University in 1988 and 1992, respectively. He joined the Department of Information Engineering at Feng Chia University as an associate professor in 1992 and became a full professor in 1999. From 1998 to 2001, he was the chairman of the department. In 2002, he joined the Department of Computer Science at National Tsing Hua University as a full professor. His research interests include parallel and distributed processing, pervasive computing, cluster computing, grid computing, embedded software, and system software for SOC design. He is a member of the IEEE computer society and ACM.

Don-Lin Yang received the B.E. degree in Computer Science from Feng Chia University in 1973, the M.S. degree in Applied Science from the College of William and Mary in 1979, and the Ph.D. degree in Computer Science from the University of Virginia in 1985. He was a staff programmer at IBM Santa Teresa Laboratory from 1985 to 1987 and a member of technical staff at AT&T Bell Laboratories from 1987 to 1991. Since 1991, he has been with Feng Chia University, where he was in charge of the University Computer Center from 1993 to 1997 and the head of the Department of Information Engineering and Computer Science from 2001 to 2003. Dr. Yang is currently a professor at Feng Chia University. His research interests include distributed and parallel computing, image processing, and data mining. He is a member of the IEEE computer society and the ACM.