Journal of X-Ray Science and Technology 23 (2015) 33–44
DOI 10.3233/XST-140468
IOS Press

Maximum intensity projection using bidirectional compositing with block skipping

Ohjae Kwon[a], Sung-Tae Kang[a], Soo-Hong Kim[b], Yoon-Ho Kim[c] and Yeong-Gil Shin[a,*]
[a]*Department of Computer Science and Engineering, Seoul National University, Seoul, Korea*
[b]*Department of Computer Software Engineering, Sangmyung University, Seoul, Korea*
[c]*Department of Computer Science, Sangmyung University, Seoul, Korea*

Received 20 January 2014
Revised 16 September 2014
Accepted 24 October 2014

Abstract.
BACKGROUND: Maximum intensity projection (MIP) is a volume rendering technique that determines the pixel intensity as the maximum of all values sampled along the viewing direction. MIP has been successfully applied to diagnose bone fractures in computed tomography (CT) and the stenosis of vascular structures in magnetic resonance angiography (MRA). However, MIP has a major drawback in that the depth and occlusion information cannot be perceived in the output images. The most universal way to alleviate this problem is to occasionally change the viewpoint for depth perception. To support this function in real time, MIP should be performed at an interactive frame rate.
OBJECTIVE: We develop an efficient rendering algorithm for MIP so that MIP is performed at an interactive frame rate without a loss of image quality.
METHODS: The proposed method predicts the position of the maximum intensity for each ray using blockwise maximum bounds, after which it performs bidirectional compositing toward both ends of the ray from this predicted position. During the compositing process, block skipping is used as an acceleration method.
RESULTS: The proposed method outperformed the block skipping method using the sequential compositing with a speed-up factor of 2.2∼2.8 depending on the data set without any degradation of the image quality.
CONCLUSION: We proposed an efficient rendering technique for MIP. Our method was superior to the conventional block skipping method with respect to the rendering speed and degree of performance consistency.

Keywords: Maximum intensity projection, acceleration techniques, prediction of the starting position, bidirectional compositing, block skipping

1. Introduction

Maximum intensity projection (MIP) is a volume rendering technique that determines the pixel intensity as the maximum of all values sampled along the viewing direction. This rendering method is

*Corresponding author: Yeong-Gil Shin, Department of Computer Science and Engineering, Seoul National University, 1 Gwanakro, Gwanakgu, Seoul, Korea. Tel.: +82 2 880 1860; Fax: +82 2 886 7589; E-mail: yshin@snu.ac.kr.

useful especially when the region of interest has high intensity. If a high-intensity structure is located somewhere inside the volume and occluded by opaque material, direct volume rendering (DVR) based on accumulation cannot visualize it, whereas MIP exhibits it clearly regardless of its position. Therefore, MIP has been successfully applied to diagnose bone fractures in computed tomography (CT) and the stenosis of vascular structures in magnetic resonance angiography (MRA), where bones and blood vessels are represented by high intensity levels. MIP is also utilized to detect defects in mechanical structures in industrial fields.

However, MIP has a major drawback in that the depth and occlusion information cannot be perceived in the output images. This increases the risk of misinterpreting the spatial relationships between different structures [1]. The most universal way to alleviate this problem is to occasionally change the viewpoint for depth perception. To support this function in real time, MIP should be performed at an interactive frame rate. However, MIP cannot employ an acceleration technique such as early ray termination [2, 3] used widely in DVR. MIP should compare every sample value along each ray to guarantee that the value it finds is the maximum value, while DVR can terminate in the middle of rendering to skip non-contributing voxels. Therefore, in order to speed up MIP, it is necessary to devise an acceleration approach that differs from the technique developed in DVR.

Several algorithms have been proposed for accelerating MIP. Cell-based methods which optimize ray traversal and interpolation [4,5] avoid unnecessary interpolations by performing them only when the upper bound of the examined cell exceeds the maximum value calculated up to that point. To acquire an interactive frame rate, these methods use integer arithmetic for ray traversal, though this can degrade the image quality. A more advanced method [6] detects and skips a set of non-contributing cells at once by using a hierarchical octree structure and a hidden volume removal based on hierarchical occlusion maps [7]. Unlike these methods using on-the-fly visibility test, another methods with cell removal [8,9] eliminate cells occluded from a cluster of viewing directions before rendering and accomplish the noticeable speed-up by processing the remaining cells in the order sorted by cell maximum. However, these methods need the dedicated storage scheme which cannot be employed by other rendering methods. In addition, they require a substantial amount of memory increase to save possibly contributing cells for multiple clusters of viewing directions.

The splatting method in a shear-warp context [10–12] renders an MIP image by splatting in a sheared object space and by evaluating the maximum value from the splatted results. The resulting image typically loses its sharpness due to the interpolations performed during splatting and warping. The progressive method [13] performs MIP at a low resolution for fast processing and uses the low-resolution image to generate a full-resolution MIP image. Because this method uses the full-resolution voxels selected by a given threshold ratio, it has a potential disadvantage in that image quality and computation amount depend on the threshold ratio.

The block skipping method [14,15] projects the vertices of a block onto the image plane and determines whether to skip a block for each pixel by comparing its maximum value and the value of a pixel in the projected area. An additional speed-up is gained by rendering blocks in a decreasing order according to their maximum values. Another block skipping method [16] excludes irrelevant blocks using an occlusion query supported by GPUs. However, these block-based methods require optimization, such as the determination of the proper block size for cache efficiency. An enhanced method [17] increases the block culling ratio by using the initial occluder obtained from the previous view. However, this method can gain a high acceleration effect only when changing the viewing direction by small amount of angles.

In this paper, we present a new acceleration technique based on block skipping. The basic idea for acceleration is to skip as many blocks as possible by means of a rapid approach to the maximum value.

To facilitate this idea, we introduce a new compositing strategy which exploits block skipping more efficiently. First, we determine the most probable position of the maximum value using the blockwise maximum bounds. Then, we start the compositing from this position and conduct it bidirectionally with block skipping. This compositing makes it possible to skip more blocks by initially assigning a high value to the maximum value.

The remainder of this paper is organized as follows. In Section 2, we briefly explain the ray casting method for MIP and discuss the problems of conventional acceleration techniques which use block skipping. In Section 3, a detailed description of our method is presented. We provide the experimental results with a brief discussion in Section 4, and we finally conclude the paper in Section 5.

2. Conventional ray casting method for MIP

The ray casting method for MIP is implemented through sampling and compositing. During the sampling process, we calculate the equidistant locations within the volume and compute the data values at those positions. The sampling position, $r(i)$ is calculated as follows:

$$r(i) = v + i \times d(x, y) \tag{1}$$

where v is the viewpoint and $d(x, y)$ represents the direction of the ray. By increasing the value of i, the ray progresses from the viewpoint [18]. The value at the sampled position is usually computed by the tri-linear interpolation of eight nearest neighbor voxels. The compositing is then executed by simply comparing the sample value and the current maximum value, called the ray maximum, and by taking the larger value, as

$$C_{out} = \max\left(C_{out}, C(r(i))\right), \tag{2}$$

where $C(r(i))$ is the value sampled at position $r(i)$ and C_{out} denotes the ray maximum. After processing all samples along the ray, the final ray maximum, C_{out}, is converted to the pixel intensity by means of the predefined mapping from data values to gray levels.

Equation (2) implies that MIP does not require a numerical integration at all. Instead, the maximum value only contributes to the final image. Most samples are unnecessary except for one sample at the maximum value. A block-type data structure for the volume has been widely used for efficient maximum evaluations. Every block refers to the maximum value among its constituent voxels. The ray jumps over the block if its maximum value is not larger than the maximum of the current ray. Otherwise, the conventional sampling process is utilized to search for the maximum value. These procedures continue until the ray visits all blocks that it passes through. This sequential compositing process is somewhat limited in terms of acceleration because it skips only a few blocks when most block maximums are monotonically increasing in the compositing order, as depicted in Fig. 1(a). On the other hand, if the volume is viewed in the reverse direction as shown in Fig. 1(b), the ray can hit the global maximum earlier and thereby skip more blocks. This indicates that the performance of MIP depends on the viewing direction.

3. Proposed method

In MIP, the compositing scheme may be applied front-to-back, back-to-front, or in any other order [1]. Using this fact, we present a more efficient compositing method which differs from existing methods with respect to the starting position and compositing direction. Unlike the conventional compositing

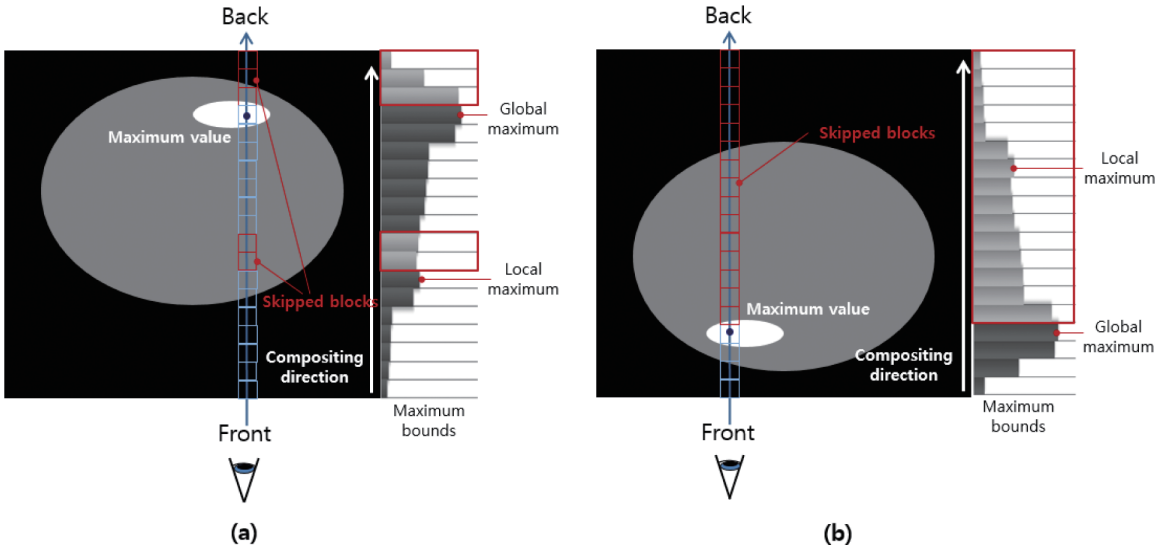

Fig. 1. Sequential compositing from front to back end position along the ray. (a) The viewpoint located at the front. (b) The viewpoint located opposite to (a). (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/XST-140468)

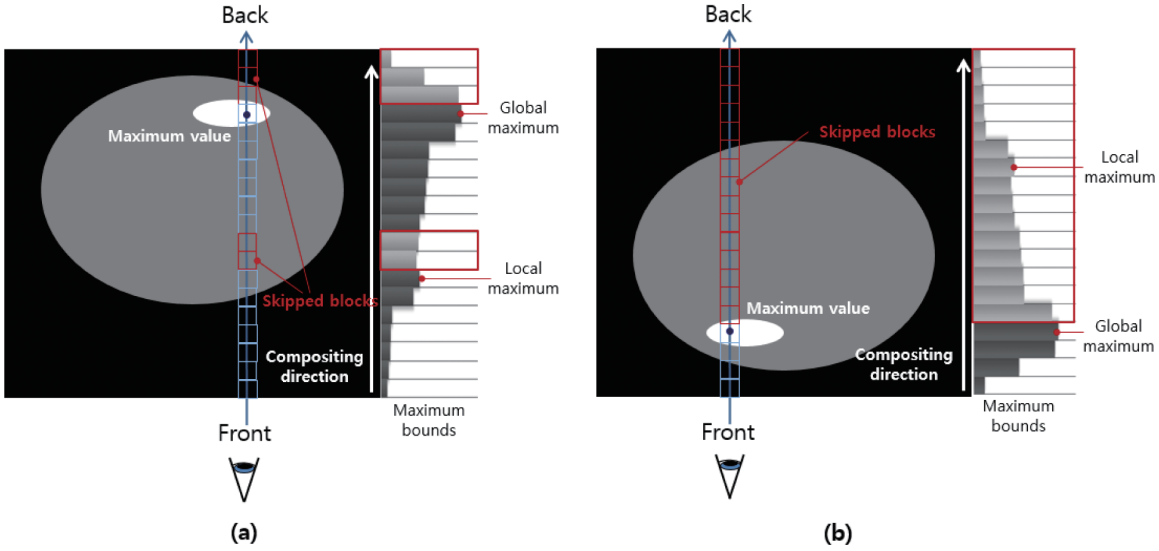

Fig. 2. Bidirectional compositing from an intermediate to both ends. (a) The viewpoint located at the front. (b) The viewpoint located opposite to (a). (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/XST-140468)

scheme starting from either the front or the back end position, the proposed compositing begins from a certain intermediate position. Starting from this position, the sampling process is performed bidirectionally toward both ends of the ray. Therefore, if the position of the global maximum is predictable, the proposed compositing process can maximize the number of skipped blocks, as shown in Fig. 2.

Bidirectional compositing can also provide consistent performance regardless of the viewing direction. If the chosen starting position is close to the maximum position, the global maximum can be found by traversing only a few blocks. This makes the rendering time consistent in all viewing directions.

Table 1
Comparison in the way of compositing

Items	Sequential compositing	Bidirectional compositing	Improved effects
Prediction	Without prediction	With prediction	Approach to the maximum value earlier
Starting position	The front or end position	An intermediate position determined by prediction	
Direction	Unidirectional from the one to the other and position along the ray	Bidirectional from the intermediate to both end positions of the ray	Maximize the number of skippd blocks
Performance consistency	Varying depending on the viewing direction	Consistent regardless of the viewing direction	Consistent rendering time

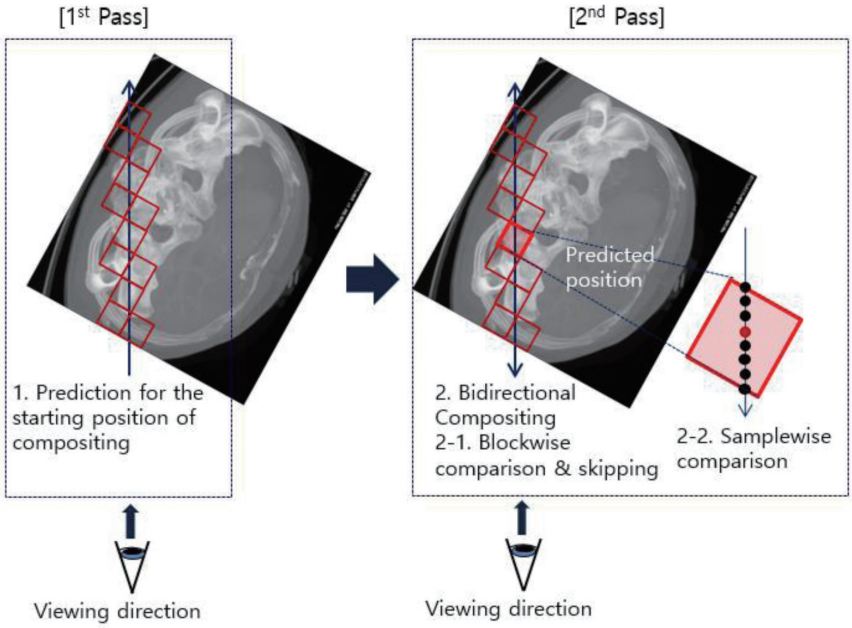

Fig. 3. Schematic overview of the proposed method, bidirectional compositing with block skipping. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/XST-140468)

Especially when the viewing directions are opposite to each other, as shown in Figs 2(a) and (b), the rendering time is identical because the bidirectional compositing is identically performed in both cases. The summary in Table 1 compares the conventional and the proposed compositing methods.

Figure 3 shows a schematic overview of the proposed method. The overall scheme consists of the prediction of the starting position and the bidirectional compositing with block skipping. These processes can be implemented by two processing passes. The first pass searches for the optimal starting position. The second pass searches for the maximum intensity, which is divided into a blockwise comparison for block skipping and a sample-wise comparison for the maximum compositing.

3.1. Generation of blockwise maximum bounds

The blockwise maximum bound is defined as the maximum value among voxels that belong to a block. We compute the maximum bounds for equal sized blocks as the preprocessing step when loading

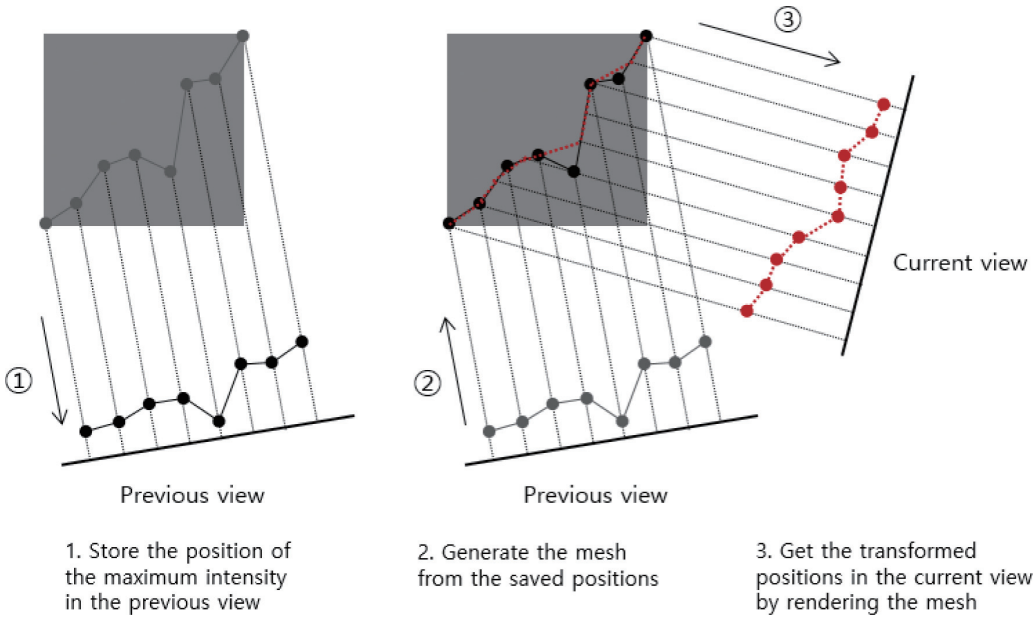

Fig. 4. The prediction of the starting position using the temporal coherency. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/XST-140468)

the volume data. A constant block size is usually preferred for easy and fast block identification by the ray-block intersection. To ensure that no sample value of a given block exceeds its maximum bound, it is necessary to overlap voxels at the block border depending on the interpolation kernel size. Otherwise, a false maximum bound may cause the ray to skip a block inappropriately. The maximum bounds are stored in the memory buffer and utilized during the rendering process.

3.2. Prediction of the starting position

Before the compositing process, we predict the starting position for each ray. In order to skip more unnecessary blocks, we need to choose as the starting position the global maximum position, which is known only after examining all sample values along the ray. Instead, we determine the starting position as the block having the largest maximum bound among blocks encountered by a ray. These blocks can be found through a ray intersection test with axis-aligned boxes [19]. This prediction process is very fast because it performs low cost calculations such as simple block identification and blockwise comparison. Although this prediction method may find an incorrect maximum position due to the block-by-block comparison, a high value is more likely to be initially assigned as the ray maximum by high spatial correlation among voxels.

Another prediction method based on temporal coherency [20] can be used in the user environment mainly using a small variation of viewing direction. This method stores the maximum intensity position for each ray in the previous view and calculates its corresponding position in the current view by re-projecting the saved position to the current view as shown in Fig. 4. In order to remove possibly introduced holes, we connect the saved positions as a mesh and perform the mesh rendering in the same manner as in [17] to obtain the starting position for the current view. When more than two positions in volume are projected on the same pixel in the current view, we select the position having the largest intensity as the starting position.

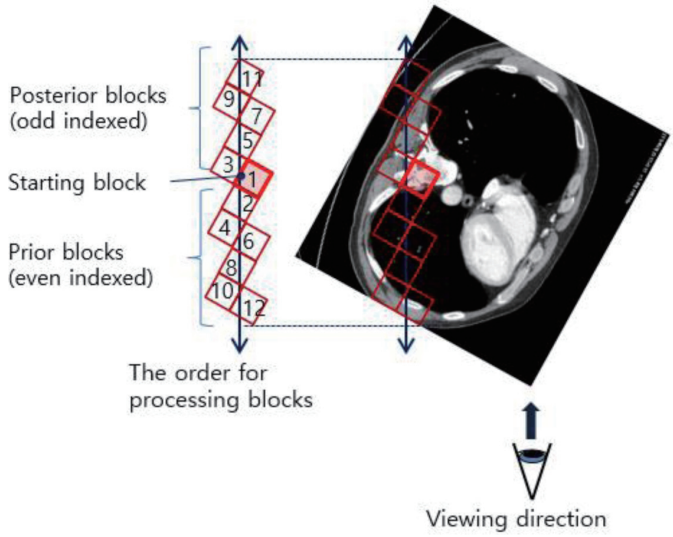

Fig. 5. The order for processing blocks. The processing order is defined in the increasing order according to the distance from the starting block. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/XST-140468)

3.3. Bidirectional compositing with block skipping

When the starting position is found, the process of searching for the maximum intensity begins. In the starting block, the initial ray maximum is computed from samples between the block entry and exit coordinates of the ray. If the initial ray maximum is equal to the current block bound, it becomes the final maximum intensity. In such a case, we stop the rendering process. Otherwise, the ray progresses to the next block. The next block is chosen in an increasing order according to its distance from the starting position. Figure 5 illustrates blocks indexed by the processing order. The prior and posterior blocks are alternately processed toward both ends of the ray. This distance-based processing order increases the possibility of finding the global maximum earlier because the global maximum is more likely to be found in the proximity of the starting position. After all blocks are processed, the ray maximum is finally converted to the corresponding pixel intensity.

3.4. Additional acceleration

Because a small block size lowers the variation between the voxel values inside a block, the starting position has more opportunities to reach the maximum value, as shown in Fig. 6. As a result, this makes our prediction more accurate.

For block skipping, we use two block sizes, as depicted in Fig. 7. The maximum bound of a large sized block is initially compared. When it exceeds the current ray maximum, the maximum bounds of small sized blocks which are included in the current large sized block are subsequently examined. In the worst case in which the maximum bounds of all small blocks continuously exceed the ray maximum, more blockwise comparisons are required without any performance gain. However, this case is not common because the ray maximum is successively updated to a larger value by the maximum bounds of small sized blocks. The two-level block hierarchy reduces more samplings of time-consuming trilinear interpolation by exploiting maximum bounds which are precalculated and saved in the memory buffer.

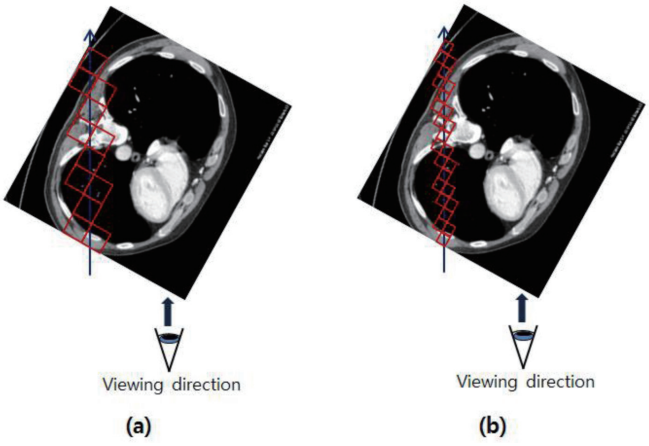

Fig. 6. Comparison in block size of the maximum bounds used to predict the starting position of the compositing. (a) Large sized block. (b) Small sized block. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/XST-140468)

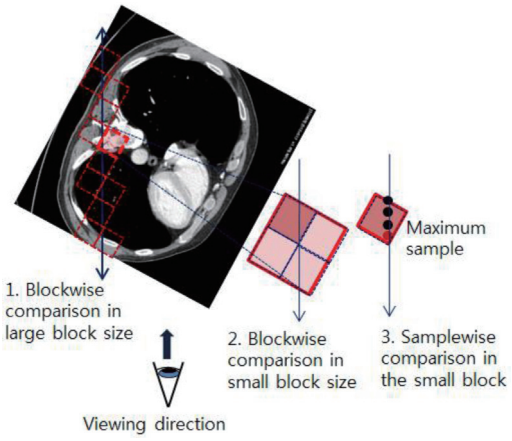

Fig. 7. Two-level block hierarchy used for block skipping. (Colours are visible in the online version of the article; http://dx.doi. org/10.3233/XST-140468)

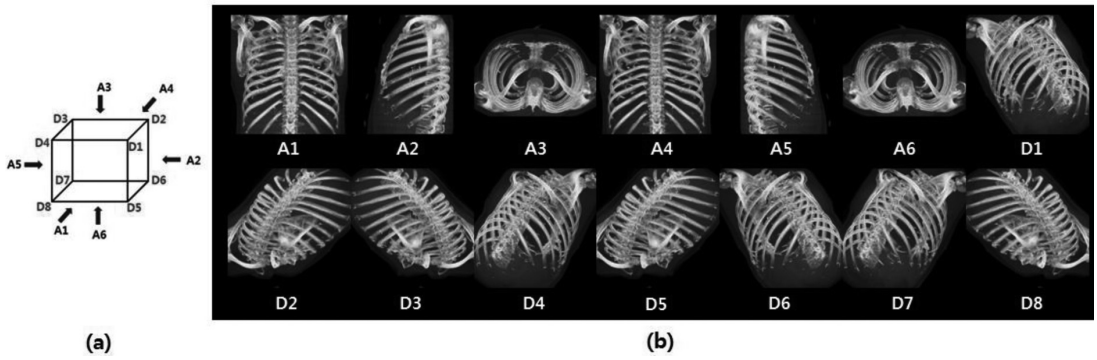

Fig. 8. viewing directions used for performance evaluation. (a) The definition of viewing directions. A1∼6 and D1∼8 denote 6 axis-aligned directions and 8 non axis-aligned directions respectively. (b) The MIP images of Ribs viewed from 14 directions.

To implement this additional acceleration scheme, a block is divided into eight small blocks of equal size. These small sized blocks are used for predicting the starting position and block skipping, whereas large ones are employed for block skipping only. For a massive sized data set, blocks may be constructed in a hierarchical fashion by dividing each block recursively. Experimental results show that this acceleration scheme brings a substantial speed-up with a slight increase in memory usage.

4. Experimental results and discussion

All tests were performed on a 2.93 GHz Intel Core i7 PC with 8 GB memory. We implemented the proposed and reference algorithms using DirectX 11 and its programmable shaders on graphics hardware equipped with Nvidia's GeForce GTX 680 GPU and 2 GB memory. The proposed method using bidirectional compositing (abbreviated to BC) is compared with a basic ray casting method with no acceleration (NA) and an accelerated method using sequential compositing (SC). Another proposed

Table 2
Test data sets

Name	Application field	Volume size	Size (MB)
Ribs	Medical	512 × 512 × 345	178
abdomen		512 × 512 × 469	243
legs		512 × 512 × 1170	598
Engine	Industrial	512 × 512 × 512	256
wheel		776 × 776 × 324	381
disc		976 × 976 × 323	601

Each data set is a CT scan with two bytes allocated for each voxel.

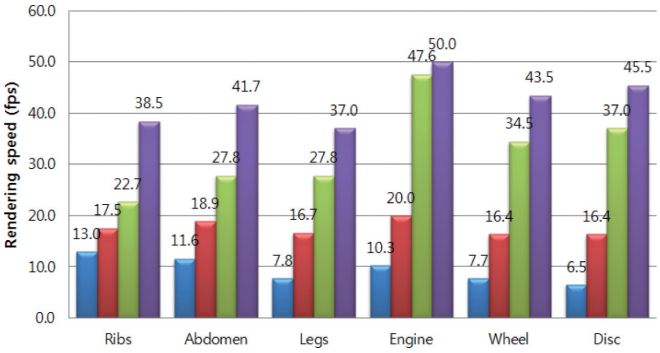

Fig. 9. The average rendering speed measured on 6 test data sets. The rendering speed is expressed in frames per second, fps. Four color bars for each data set indicate four methods, NA (blue), SC (red), BC (green), BC + 2b (purple) respectively. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/XST-140468)

method using two-level block hierarchy (BC + 2b) also was tested to verify its additional acceleration effect. The proposed methods employed the prediction method that determines the starting position using blockwise maximum bounds. We utilized six CT data sets used in medical and industrial fields, as listed in Table 2. In our implementation, each data set is loaded into a single three-dimensional texture to employ a fast texture sampling supported by GPUs. This data structure has the advantage in that it can be reused by other rendering methods. Blockwise maximum bounds are calculated in a preprocessing step and then loaded additionally into another three-dimensional texture for our method. However, it is not a big burden since its size is significantly small corresponding to one of the block size of the original data size. The block size was set to 16^3 which showed a satisfactory rendering speed for all test data sets. For two-level block hierarchy, a size of 8^3 was used for small sized blocks. The output image was rendered at a resolution of 768 × 768 pixels.

Figure 9 shows a comparison of the average rendering speeds from 10 repetitive executions for 14 predefined viewing directions, as shown in Fig. 8. Experimental results showed that our methods always outperformed conventional methods.

A rendering speed that exceeds 30 frames per second (fps) is usually required for a delayless response to user interactions. Our first method provided this frame rate for industrial data sets but failed to meet this specification for medical data sets. This is due to the different characteristics of the spatial correlation between two classes of data sets. Industrial data sets typically show higher spatial correlation because target objects are mechanical parts made by a single metal material, whereas medical data sets consist of many objects with various intensities. This performance difference mainly stems from the prediction accuracy of the starting position. With two-level block hierarchy accompanying additional speed-up of 9.8 fps on average, our second method attained a rendering speed of more than 30 fps for all data sets.

The improved rendering speed of our algorithm comes from the high block skipping rate. Table 3 shows the average block skipping rate per ray. Our first method yielded a higher block skipping rate (72.8%) than the conventional method (50.7%). Furthermore, our second method showed a better result (74.2%) due to a more accurate prediction of the starting position. This block skipping rate does not clearly represent the additional speed-up of our second method because large sized blocks only are counted. Table 4 shows a more accurate performance comparison by including the skipping of small sized blocks. The last row presents the reduced samplings of our second method against the sequential compositing method.

Table 3
The average block skipping rate per ray (%)

	Ribs	Abdomen	Legs	Engine	Wheel	Disc	Average
$SC^{(1)}$	46.6	47.5	42.8	59.4	52.3	55.7	50.7
BC	64.3	72.5	59.1	83.7	80.4	76.8	72.8
BC + $2b^{(2)}$	66.4	74.5	60.1	84.9	81.4	77.8	74.2
(2) − (1)	19.8	27.0	17.3	25.4	29.1	22.1	23.5

The average block skipping rate per ray is calculated on 6 test data sets. Rays passing through the central image area only are considered to gain the more exact skipping rate. The block skipping rate is calculated by dividing the number of skipped blocks by that of ray-intersected blocks. The skipping rate of large sized block only is specified in BC + 2b for a fair comparison.

Table 4
The average number of samples per ray used for maximum evaluation

	Ribs	Abdomen	Legs	Engine	Wheel	Disc	Average
$SC^{(1)}$	588.4	588.4	380.0	483.3	476.5	378.3	482.5
BC	419.9	299.8	209.1	191.9	160.1	147.1	238.0
BC + $2b^{(2)}$	268.1	178.1	144.2	109.7	96.8	90.9	148.0
((2) − (1))/(1)	0.54	0.70	0.62	0.77	0.80	0.76	0.69

The average number of samples per ray used for maximum evaluation is calculated on 6 test data sets. Rays passing through the central image area only are considered to gain the more exact sampling number. Our second method using two block sizes (BC + 2b) uses the smallest number of samples.

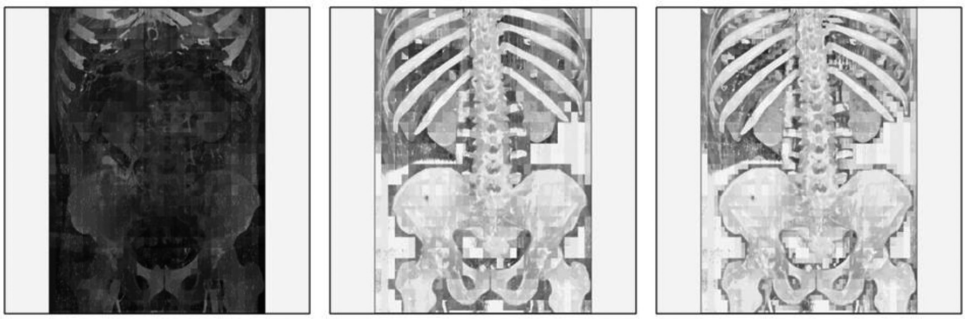

Fig. 10. Images expressed by the number of processed blocks prior to the global maximum for Abdomen data set. Brighter pixels imply that a small number of blocks are processed. (a) SC. (b) BC. (c) BC + 2b.

Each pixel in Fig. 10 represents the number of processed blocks prior to the global maximum. Brighter pixels show fewer processed blocks. Compared with the conventional method, our first method expressed brighter results in most areas. In our second method, more areas were bright specifically around the bone boundaries because small sized blocks yielded fewer prediction errors compared with large sized blocks in regions showing a considerable amount of variation.

Figure 11 shows the rendering time for different viewing directions on data set of Legs. We measured the rendering time of 14 different viewpoint directions, specifically six axis-aligned and eight non-axis-aligned directions. The axis-aligned directions are defined as the forward and backward directions along x, y, and z-axis respectively. The non-axis-aligned directions correspond to the directions viewed from each vertex to its opposite vertex in terms of the volume (see Fig. 8). The rendering time was generally longer in non-axis-aligned views because these views had to process more blocks or samples along a longer ray with lower cache efficiency. The existing methods showed the dependency of the rendering

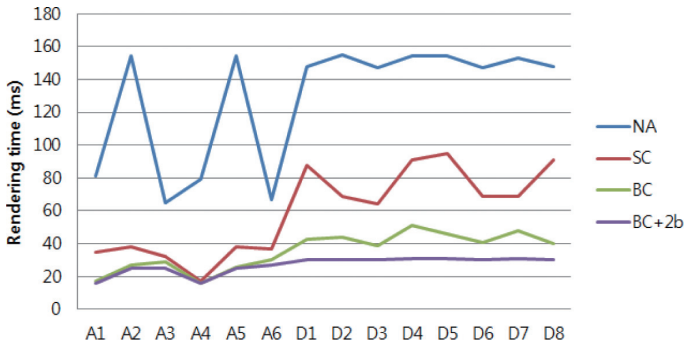

Fig. 11. Rendering time measured for 14 viewing directions on the data set of Legs. Four methods, NA, SC, BC, BC + 2b are compared. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/XST-140468)

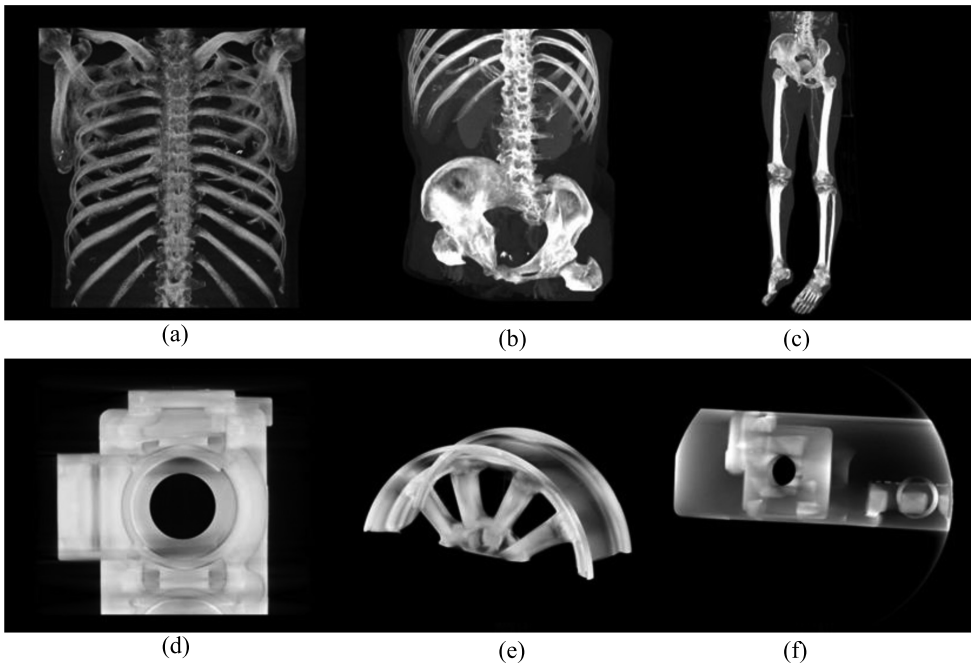

Fig. 12. MIP images of test data sets rendered by our second method (BC + 2b). (a) Ribs. (b) Abdomen. (c) Legs. (e) Engine. (f) Wheel. (g) Disc. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/XST-140468)

performance on the viewing direction, whereas our methods exhibited a relatively consistent rendering time. This result comes from the bidirectional compositing process, which reduces the influence of the spatial distribution of high values by starting from the predicted maximum position.

Our method was superior to the block skipping method using the sequential compositing with respect to the rendering speed and degree of performance consistency. These advantages are very important for practical uses, which require frequent changes of viewpoints. The proposed method outperformed the sequential compositing method with a speed-up factor of 2.2∼2.8 depending on the data set without any degradation of the image quality. Figure 12 shows the final results rendered by our method, which yielded the same image quality as the conventional method. Hence, our method is suitable for applications requiring both high interactivity and high-quality images.

5. Conclusion

In this study, we proposed an efficient rendering technique for MIP. The proposed method accelerates the rendering process by block skipping through the bidirectional compositing process. In order to maximize the number of skipped blocks, we presented the following three techniques: prediction of the starting position for compositing, bidirectional compositing with block skipping, and two-level block hierarchy for prediction and block skipping. These techniques make it possible to find the maximum value earlier while increasing the number of skipped blocks, which brings a significant enhancement of the rendering time. Experimental results showed that our method exhibited superior performance for all test data sets and achieved a 2.5x rendering speed-up compared with the block skipping method using the sequential compositing while preserving the image quality. Combined with a more exact prediction method, the proposed algorithm will be able to accelerate MIP further.

References

[1] K. Engel, M. Hadwiger, J.M. Kniss, C. Rez-Sajama and D. Welskopr, *Real-Time Volume Graphic*, A.K. Peters Ltd., 2006.
[2] M. Levoy, Efficient ray tracing of volume data, *ACM Transactions on Graphics* **9**(3) (1990), 245–261.
[3] J. Danskin and P. Hanrahan, Fast algorithms for volume ray tracing, *Workshop on Volume Visualization* (1992), 91–98.
[4] G. Sakas, M. Grimm and A. Savopoulos, Optimized maximum intensity projection, *Proceedings of 5th EUROGRAPHICS Workshop on Rendering Technique* (1995), 55–63.
[5] K.J. Zuiderveld, A.H.J. Koning and M.A. Viergever, Techniques for speeding up high-quality perspective maximum intensity projection, *Pattern Recognition Letters* **15** (1994), 507–517.
[6] B. Mora and D.S. Ebert, Low-complexity maximum intensity projection, *ACM Transactions on Graphics* **24**(4) (2005), 1392–1416.
[7] H. Zhang, D. Manocha, T. Hudson and K.E. HOFF, Visibility culling using hierarchical occlusion maps, *Proceedings of ACM SIGGRAPH'97* (1997), 77–88.
[8] L. Mroz, A. König and E. Gröller, Maximum intensity projection at warp speed, *Computer and Graphics* **24** (2000), 343–352.
[9] L. Mroz, H. Hauser and E. Gröller, Interactive high-quality maximum intensity projection, *Proceedings of EUROGRAPHICS* (2000), 341–350.
[10] W. Cai and G. Sakas, Maximum intensity projection using splatting in sheared object space, *Computer Graphics Forum'98* **17**(3) (1998), 113–124.
[11] P. Lacroute and M. Levoy, Fast volume rendering using a shear-warp factorization of the viewing transformation, *Proceedings of SIGGRAPH '94* (1994), 451–458.
[12] B. Csébfalvi, A. König and E. Gröller, Fast maximum intensity projection using binary shear-warp factorization, *Proceedings of WSCG'99* (1999), 47–54.
[13] K.H. Kim and H.W. Park, A fast progressive method of maximum intensity projection, *Computerized Medical Imaging and Graphics* **25** (2001), 433–441.
[14] V. Pekar, D. Hempel, G. Kiefer, M. Busch and J. Weese, Efficient visualization of large medical image datasets on standard PC hardware, *Proceedings of the Symposium on Data Visualization '03* (2003), 135–140.
[15] G. Kiefer, H. Lehmann and J. Weese, Fast maximum intensity projections of large medical data sets by exploiting hierarchical memory architectures, *IEEE Transactions on Information Technology in Biomedicine* **10**(2) (2006), 385–94.
[16] H. Kye and D. Jeong, Accelerated MIP based on GPU using blocks clipping and occlusion query, *Computer amd Graphics* **32** (2008), 283–292.
[17] H. Kye, B.S. Sohn and J. Lee, Interactive GPU-based maximum intensity projection of large medical data sets using visibility culling based on the initial occluder and the visible block classification, *Computerized Medical Imaging and Graphics* **36** (2012), 366–374.
[18] J. Krüger and R. Westermann, Acceleration techniques for GPU-based volume rendering, *Proceedings of IEEE Visualization'03* (2003), 287–292.
[19] A. Williams, S. Barrus, R.K. Morley and P. Shirly, An efficient and robust ray-box intersection algorithm, 2004.
[20] R. Yagel and Z. Shi, Accelerating volume animation by space-leaping, *Proceedings of IEEE Visualization* (1993), 62–69.