中華民國八十六年全國計算機會議

A New Rendering Method: Scan Line Based Semi-Boundary Algorithm
以掃瞄線爲基礎的 Semi-Boundary 演算法

翁子倫　　李同益　　孫永年

Tzu-Lun Weng　　Tong-Yee Lee　　Yung-Nien Sun

成功大學資訊工程研究所

Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, ROC

摘要

虛擬手術模擬常用於演練手術進行的過程，以便讓醫師設計適當的手術步驟，這其中所牽涉的技術包括即時顯示人體各部位的器官,而且模擬的手術過程必須讓使用者直覺上很容易操作，本論文的主要目的就是描述一種新的顯示物體的方法，我們稱此方法爲 SSB 演算法

Abstract

The computer graphics simulation system has been used as a powerful tool in surgical and therapeutic applications. In such system, it must provide a real-time rendering performance as well as flexible manipulation on 3D objects. In this paper, we propose a new volume rendering scheme, namely SSB (Scan-line Based Semi-Boundary).

中文關鍵字 ： 掃瞄線、手術模擬、虛擬實境。

Keywords : SSB 、 SB 、 surgical simulation 、 volume rendering 、 pelvis.

1. Introduction

3D visualization technique has been exploited to explore new discovery of science in many areas such as engineering simulation, fluid dynamics and medical applications [1,2,3,4]. In medical applications, the clinics use computer-generated images to assist in surgical diagnosis and therapy planning. Three-dimensional arrays of digital data representing spatial volumes are commonly used in medical applications like sequences of two-dimensional images derived from CT or MRI. These data are always very large. Several storage mechanisms have been propose to reduce the size of storage such as Semi-boundary [3,4] and Oct-tree with run-length coding [3]. These schemes are also exploited to speedup rendering computation. In the past, many efforts have been proposed for volume rendering such as ray casting [1,5,6], back-to-front and front-to-back projection [9]. These techniques are only capable of real-time rendering large volume data on massively parallel computers. With these schemes, it is hard to develop VR applications for medical purposes in which require high interactions. Recently, P. G. Lacroute proposed a shear-warp scheme that exploits coherence in both volume data and the image [1]. Their solution achieves about 1 second rendering performance for 256x256x167 Brain data set but requires an expensive preprocessing step and three copies of volume data is represented as SB data structure. Our new scheme is termed SSB (Scan-line Based Semi-Boundary) and is currently being used in our virtual surgical system. With this new algorithm, we achieve a very fast rendering performance comparable to those obtained on the state-of-art graphics workstations.

In Section 2, we briefly overview SB concept and its data representation. Our SSB scheme will be introduced in Section 3. Our implementation details and experimental results will be presented in Section 4. In final, some concluding remarks and future work are given in Section 5.

2. Overview on Semi-boundary (SB)

Our SSB (Scan-line Based Semi-Boundary) derives from SB scheme [3].There is some preprocessing work required to create SB data structure as well as other useful data structures such as a normal lookup table. The SB data structure is represented by a two-dimensional link-list (shown in Figure 1) and is used to

Supported by:(1) NSC-87-2213-E-006-062
(2) NSC-87-2213-E-006-012

represent the surface information of volume data [3]. In Figure 1, the $P[i][j]$ is the starting position of each linked-list, d is depth, m is neighboring configuration, code n is normal vector index, and *link* is a pointer to the next node in the same linked-list. Each $P[i][j]$ denotes a scan-line list starting from location $[i][j]$ on YZ plane along the X axis.

For the SB, the image data captured by an imaging device such as CT scanners can be considered as a pair $C=(g, c)$, where C is called the scene domain and g is a mapping of C into a set of numbers.

$$C=\{c=(c1,c2,c3)|for1\le i\le 3,1\le c_i\le h_i\}$$

Each c is termed voxel and g (c) is the density of c. Segmentation is a mapping of a scene to a binary scene C as denoted as $S(C)=C_s=(g_s,C)$,where it represents the binary scene resulting from C using a segmentation function s. A commonly used s segmentation function is defined below, where T is a fixed threshold.

$$g_s(C)=\begin{cases}1 & if g(C)\ge T\\ 0 & otherwise\end{cases}$$

And
$U_s(C)=\{c\in C|g_s(c)=1\},N_s(C)=\{c\in C|g_s(c)=0\}$
, where U_s represents the SB structure we want to visualize, and N_s represents the background. For the SB, we define neighboring nodes of a voxel c as:

$$n(c)=\{d|j,1\le j\le 3,|c_j-d_j|=1 and c_i=d_i if i\ne j\}$$

We can encode these neighboring nodes in a single neighboring configuration code as $\Delta(c)=\sum_{d\in n(c)} g_s(d)\cdot m(d)$,
where $m(d)=2^0,2^1,2^2,2^3,2^4,2^5$. This code is very useful to determine a voxel's potential visibility [3].

3. SSB Algorithm (Scan-line Based Semi-Boundary)

Our SSB method derives from the fact that the parallel lines are still parallel after a parallel projection transform. We can exploit this property to speed up rendering computation. Note that our SSB can do not work correctly for the perspective projection. However, in general medical applications, the image is always generated in a parallel projection manner. So, our scheme is still very promising in medical applications. The SSB scheme is described as follows.

As shown in Figure 3, let X, Y, Z be axes for reference frame O, where its origin is located at the center of volume data, and the world reference frame W is defined by X', Y', Z' axes. The volume data is parallel projected into an image plane that is on O's XY plane. To view an object in any orientation can be achieved through a sequence of rotations. We find these three rotation angles regarding the reference frame O's X,Y,Z axes, say the angles α about the X axis, β about the Y axis, and γ about the Z axis, respectively. In our scheme, the image projection plane is on the X-Y plane and the linked-list orientation of SB data structure is created along the X-axis. All linked-lists can be thought as parallel lines along the X-axis. In such arrangement, after a sequence of rotations plus a parallel projection, all parallel scan-lines are still parallel on the image plane. In case that only X-axis rotation is required, it is obvious to know that the projected lines are still parallel and they also preserve their length. However, in case that the Y-axis rotation is required, each parallel line has to be scaled by the value of $\cos(\beta)$ as shown in Figure. 2

When the angle γ is not zero, the parallel lines in the SB are still a sole parallel projection onto the image plane. However, the image plane must rotate with the same angle γ about the Z-axis in a reverse direction to obtain correct result (illustrated in Figure 3). With above concept, we can only multiply each starting node of each list on the YZ plane by a parallel projection matrix. For the remaining nodes in each list, we multiply each node's *depth* by a constant value, $\cos(\beta)$, to obtain its relative offset to the starting node, and thus to obtain its exact location on the image plane without need of a parallel projection transform. In contrast, the original SB algorithm must perform a parallel projection matrix multiplication for all nodes in the SB data structure. Therefore, the complexity of SSB is $O(MN)$ plus a 2-dimensional rotation, but SB is ordered by $O(LMN)$, where L,M,N are the SB nodes in X, Y, Z axes, respectively.

To fill in the detail, a pseudo-code version of the SSB algorithm is listed below:

Procedure scan-line_semi_boundary;

```
begin
//-------------------------------------------------------------
//Part I: initial parameter and get lookup table
//-------------------------------------------------------------
//get image coordinate transform configuration
  spin=trans->spin; tilt=trans->tilt;
  cosx=trans->cosx; cosy=trans->cosy;
  sinx=trans->sinx; siny=trans->siny;
  // get lookup table
  SetConfigLookup();
  SetNormalLookup();
  SetPhoneLookup();
  for i :=0 to number_of_height do begin
    for j := 0 to number_of_width do begin
      cur = &sb[i][j];
      //-------------------------------------------------------
      // part III-A : compute the starting position of each
      //              linked-list
      //-------------------------------------------------------
      do transformation of the start position of linked-list
      while ( cur!=NULL )
      {
      //-------------------------------------------------------
      // Part II: lookup configuration table
      //-------------------------------------------------------
      if configuration of SB node is visible do begin
      //-------------------------------------------------------
      // Part III-B: computation other nodes along scan-line
      //-------------------------------------------------------
        x = x1 + cur->depth*cosy ;
        z = z1 + cur->depth*siny ;
      //-------------------------------------------------------
      // Part IV: Z-buffer, lookup normal and shading table
      //-------------------------------------------------------
        if depth less than the depth in Z buffer do begin
            Z-buffering;
            do lookup normal vector table;
            do lookup shading table;
        end;
      end ;
      }
      cur := next link-list node;
    end; // end of if
  end; // end of j
end; // end of i
if γ ≠ 0 do image rotation;
end ; // end of procedure
```

Fig. 4 Pseudo-code of scan-line semi-boundary algorithm

In this procedure, Part I setups all initial parameters and all related lookup tables, Part II checks the visibility of each SB node, Part III-A computes parallel projection of the starting node for each linked-list, Part III-B calculates offsets for the remaining nodes, and Part IV performs Z-buffering for the removal of the hidden-surface and illumination shading.

4. Implementation Details and Experimental Results

In our implementation, the normal vector $n(x,y,z)$ is computed by the moment-based edge operator [8]. The normal vectors of objects can be pre-computed and stored in a lookup table to reduce the rendering time. Instead of storing all three components of the normal as floating point numbers we use an encoded representation that requires less storage and provides a convenient integer index for lookup table. The general method for encoding normal vectors is described below:

1. Tessellate a sphere uniformly to 1280 triangular faces in which their unit normal vectors, denoted as N, are approximately uniformly distributed over all directions. Note that from our experience, the normal vectors generated from these 1280 triangular faces are sufficient enough to represent the entire surface normal vectors for our tested medical images.

2. Assign a unique index to each unit normal vector.

3. For each $n(x,y,z)$ of c, in N, find the closest unit vector with a unique index n, and substitute $n(x,y,z)$ by n.

In our previous work [7], we proposed a MSB scheme (Modified Semi-Boundary) which performs better than the original SB algorithm. In this section, we will study performance comparisons between MSB and SSB. For the completeness, we outline the MSB algorithm in Figure 5.

```
Procedure modified semi_boundary();
begin
//-------------------------------------------------------------
// Part I: initial
// ------------------------------------------------------------
//get image coordinate transform configuration
spin=trans->spin; tilt=trans->tilt;
cosx=trans->cosx; cosy=trans->cosy;
sinx=trans->sinx; siny=trans->siny;
// get lookup table
SetConfigLookup();
SetNormalLookup();
for i :=0 to number_of_height do begin
  for j := 0 to number_of_width do begin
      cur = &sb[i][j];
      while ( cur!=NULL )
      {
      //-----------------------------------------
      // Part II: lookup configuration table
      //-----------------------------------------
      if configuration of SB node is visible do
      begin
      //-----------------------------------------
      // Part III: parallel projection
      //-----------------------------------------
      do transformation of each semi-boundary
```

```
      /-------------------------------------------------
      // Part IV: Z-buffer, lookup normal vector
      //table , and phone shading
      //-------------------------------------------------
      if current depth less than the depth in
      buffer do begin
            do Z_buffering;
            do lookup normal vector table;
            do lookup shading table;
       end;
      end ;
      }
      cur := next link-list node;
      end; // end of if
   end; // end of j
  end; // end of i
end ; // end of procedure
```

Fig. 5 Pseudo-code of modified semi-boundary algorithm

We implemented both MSB and SSB algorithms on SUN Sparc station-20 with 64 M memory workstation, and evaluate their performance difference in rendering of a 512x512x245 pelvis volume data with a 512x512 image resolution. Our experimental measurements both use wall clock execution time of each algorithm on the pelvis data set. The primary sources of overheads in both algorithms can be classified into four categories listed below:

1. Looping: It includes overheads spent on control overheads such as updating loop counters, advancing pointers and traveling the linked-list structure.

2. Neighboring configuration tests: Time spent checking visibility of nodes using neighboring configuration code. This test is very essential to the overall performance and, in general, cuts off the half number of nodes in the linked-list and thus reduces rendering time. Our tested pelvis volume data consists of approximately 450,000 SB nodes. The half number of nodes is culled after the neighboring configuration tests.

3. Parallel projection transformation: Time spent parallel projecting nodes in the linked-list structure. This work is the most time-consuming part compared with the other overheads.

4. Z-buffering and shading: Time spent removing the hidden-surface and shading the volume data.

For other miscellaneous fixed overheads such as lookup table initializations (about 2~3 milliseconds) is less significant than the above listed overheads. The total rendering time for MSB and SSB are 1742 (1.0 frame/sec.) and 957 (0.57 frame/sec.) milliseconds, respectively. The SSB is faster than the MSB by a factor of 1.82 times. To further analyze the performance difference, a breakdown of the total execution time for MSB and SSB are shown in Tables 1 and 2. Both show the exact timings and the percentages of the total execution time for each category of overheads.

Recall that the algorithmic details in both MSB and SSB are all the same except Part III. The MSB executes parallel projection transformation (i.e., 9 multiplication and 6 addition operations) for all SB nodes in Part III, however, the SSB executes parallel projection for the starting node only and calculate less computations (i.e., 2 multiplication and 2 addition operations) for the remaining SB nodes. Difference between two kinds of transformations is approximately 5 times. For Part III, the ratio of improvement for the SSB over MSB can be approximately formulated as $(5L)/(5+L)$. The average number (L) of the SB nodes in each list (i.e., X dimension) for the pelvis volume data is approximately 4.9. Thus, the improved ratio is about 2.43 in theoretical respect. Tables 1 and 2 show that in Part III the improved ratio is 2.73 in experimental respect. Both ratios match well, and thus our formula can serve as an approximate predictor of the improved performance. The percentage results show the most time consuming overheads are incurred in Part III. Some future work can be done to improve performance of other time killers such as looping, Z-buffering and shading.

Rendering time depends on viewing angles, the number of SB culled in Part II may vary with view-points. Figures 6 and 7 show rendering time versus viewing angle for the MSB and SSB algorithms. Recall that un-culled SB nodes will execute the Part III and thus make difference in rendering time. In both figures, we see some variations in rendering time as the rotation angle increases. To verify our claim (i.e., variations are mainly due to the number of un-culled SB nodes), we plot an extra Figure 8 for the un-culled SB nodes executing the SSB algorithm. Refer to both Figures 7 and 8, we see clearly that variations in both figures behaves in the same manner. Finally, we present one slice of our rendered results for the pelvis data in Figure 9.

5. Conclusion and Future Work

In this paper, we present an efficient volume rendering technique termed SSB. The SSB can achieve an interactive performance for large volume data such as a 512x512x245 pelvis data. We carefully analyze our SSB algorithm and compared with the original SB and MSB algorithms. The SSB performs better than SB and MSB both in theoretical and experimental respects. Many future works will be done as follows. We plan to enhance our SSB with the morphing technology to perform surgical simulation for the pelvis system. Some user-friendly manipulation tools with the SSB will be developed soon.

Acknowledgement
This research is supported in part by NSC-87-2213-E-006-062 and NSC-87-2213-E-006-012.

References:

[1] Philippe G.Lacroute "Fast Volume Rendering Using A Shear-Warp Factorization of the Viewing Transformation" PhD. Thesis. September 1995.

[2] R. Anthony Reynolds Dan Gordon and Lih-Shyang Chen "A Dynamic Screen Technique for Shaded Graphics Display of Slice-Represented Objects. Computer Vision, Graphics and Image Processing 38,pp.275-298, 1987

[3] Jayaram K. Udupa and Dewey Odhner "Fast Visualization, Manipulation, and Analysis of Binary Volumetric Objects" IEEE CG&A. pp.53-62 Nov. 1991.

[4] Jayaram k.Udupa and Dewey Odhner "Shell Rendering" IEEE CG&A. pp58-67 Nov. 1993.

[5] Andrew S. Glassner. "Space Subdivision for Fast Ray Tracing" IEEE CG&A pp15-22 Oct. 1984

[6] Akira Fujimoto, Takayuki Tanaka, and Kansei Iwata "ARTS:Accelerated Ray-Tracing System" IEEE CG&A pp.16-26 1986

[7] Yuh-Hwan Liu , Chi-Wu Mao and Yung-Nien Sun "Three-Dimension Reconstruction, Registration and Fusion For Multimodality Medical Images" , PhD. Thesis. Of Department of Electrical Engineering National Cheng Kung University, Tainan, Taiwan. May 1997.

[8] L.M. Luo, C.Hamitouche, J.L. Dillenseger, and J.L. Coatrieux, "A moment-based three-dimensional edge operator", IEEE Trans. Biomedical Engineering, Vol. 40. No.7, pp.693-703, July 1993.

[9] Giedeon Frieder , Dan Gordon and R.Anthony Reynolds " Back-to-Front Display of Voxel-Based Objects" IEEE CG&A pp.52-60 January 1985.

[10] B.K. P. Horn, "Extended Gaussian Images," Proceedings of the IEEE, Vol.72, No. 12, pp 1671-1686, Dec.1984

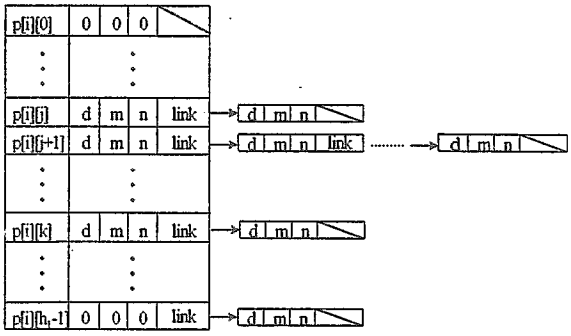

Fig 1. A SB (Semi-Boundary) data structure

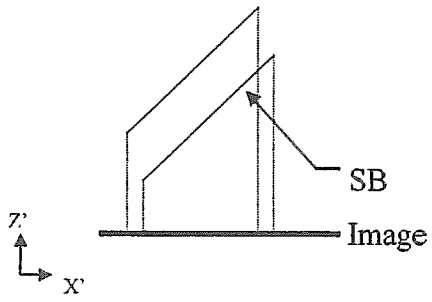

Fig . 2 if $\beta \neq 0$ link-list is scaled by $\cos\beta$

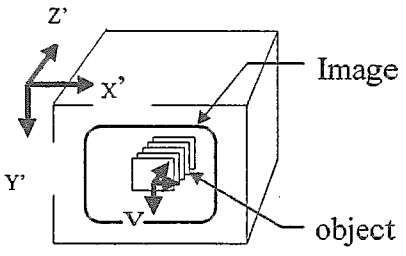

Fig. 3 if $\gamma = 0$, image plane not rotation.

MSB algorithm	Timing	Percentage
Looping	324	21%
Neighboring configuration test	68	4%
Parallel projection transformation	947	61%
Z-buffering and shading	218	14%

Table 1. Detailed timing breakdowns for the MSB algorithm

SSB algorithm	Timing	Percentage
Looping	324	34%
Neighboring configuration test	68	7%
Parallel projection transformation	347	36%
Z-buffering and shading	218	23%

Table 2. Detailed timing breakdowns for the SSB algorithm

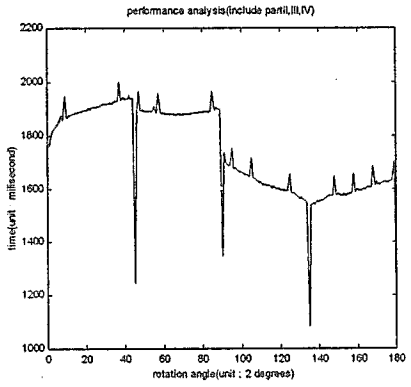

Figure 6. Rendering time versus viewing point for the MSB

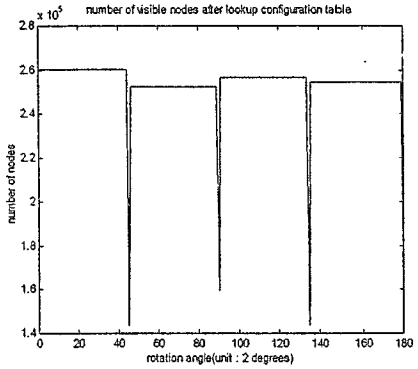

Figure 8. The number of un-culled SB nodes versus different view angles

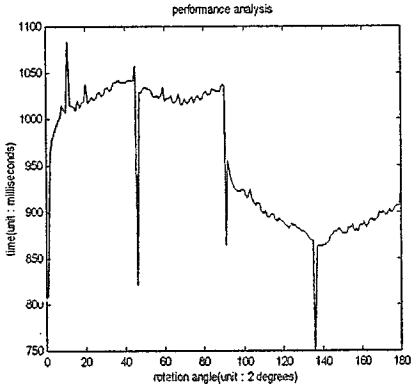

Figure 7. Rendering time versus viewing point for the SSB

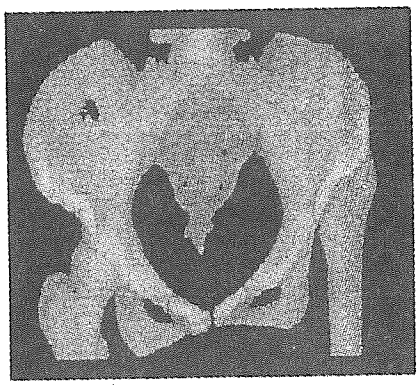

Figure 9. The rendering image of pelvis (data set:512x512x245)