Computers & Graphics 34 (2010) 449–459

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

A rapid 3D seed-filling algorithm based on scan slice

Wei-Wei Yu [a,*], Fei He [b], Ping Xi [a]

[a] Department of Aircraft Manufacturing Engineering, School of Mechanical Engineering and Automation, BeiHang University, Beijing 100191, PR China
[b] Department of Orthopaedic Surgery, 1st Affiliated Hospital of Kunming Medical College, Kunming 650032, PR China

ARTICLE INFO

Article history:
Received 2 February 2010
Received in revised form
11 April 2010
Accepted 19 May 2010

Keywords:
Seed-filling
Scan slice
Volume graphics
Computer graphics

ABSTRACT

In this paper, a novel and rapid 3D seed-filling algorithm is proposed to extract or fill the object-connected 3D region. An improved 2D seed-filling algorithm, which extracts connected region in slice quickly and consumes fewer stack operations and less memory compared with the existing algorithms, is presented. The improved 2D algorithm is enclosed as a basic unit within the framework of the proposed 3D seed-filling algorithm, in order to reduce the complexity of direction of seeds search, and accelerate region search on adjacent slices. Finally, a parameter of scan range is defined to leap over invalid seeds, which reduces time consumption of the proposed algorithm further. In addition, experimental results demonstrate advantages of this algorithm including eliminating the redundancy of seeds search, repetition of stack operations and running with high efficiency.

1. Introduction

The seed-filling algorithm is one of the traditional and classical algorithms in computer graphics. Its principle is that an initial seed is employed to search and then fill the object-connected 2D pixels or 3D voxels in a region with closed boundary. It can be used to fill regions with arbitrary shape through 4-connected (2D) or 6-connected (3D) approach.

The principle of seed-filling algorithm is widely applied in numerous fields. For example, seed-filling can be utilized to extract the segmented region of 2D image, obtain the characters of geometric shapes of image (such as area, circumference and center of mass), calculate regional counting and fill complicated binary images [1–4].

In addition, the principle of 3D seed-filling is used in the fields of volume graphics by many scholars. For example, Levoy [5]uses seed-filling to replace the bounding boxes to accelerate ray tracing by leaping over empty space; Oikarinen [6] uses seed-filling in a view lattice to avoid processing empty space and then accelerate their volume rendering; and Tsai et al. [7] manipulates volumetric objects by using 3D seed-filling to replace a memory-consuming voxel extension.

There are different approaches to implement the principle of seed-filling in 2D or 3D space. The simplest approach is called flood-filling algorithm. It implements seed-filling algorithm through seeds recursion, which searches valid seeds within the 4-connected region. However, in implementation of this algorithm, the connectivity of adjacent seeds is not well utilized. In addition, each seed is accessed repeatedly, and only one seed can be popped at each time. These problems lead to large memory and time consumption during high frequent stack operations (see our experiments in Section 2). Moreover, when a large area in 3D space is processed, it may probably lead to the memory resource exhaustion.

Another approach, which is called the traditional scan line seed-filling algorithm, can remarkably overcome most of the above described problems. In implementation of this approach, more seeds can be pushed or popped along the scan lines at a certain time, and only the rightmost seeds will be pushed into stack. The scan line seed-filling algorithm reduces the frequency of repetitive seeds search, the frequency of stack operations and the memory consumption. However, the connectivity of adjacent scan lines is not well utilized to reduce repetitive rollbacks (see Section 3.1 for the definition).

In addition, although the frequency of repetitive seeds search is reduced, this problem still largely exists. Therefore, when a large area is processed, the performance of this algorithm should be improved to reduce the time consumption.

Actually, only the latter approach is widely employed in 3D space. Feng and Soon [8] describe a 3D seed-filling algorithm that extends the traditional scan line seed-filling algorithm from 2D to 3D, by extending the search directions of scan lines from the axes of $-y/+y$ to the axes of $-y/+y/-z/+z$. It can be used to extract or fill the region of interest (ROI) with arbitrary shape. However, the defects of high frequency of stack operations and repetition of seeds search still exist (see our experiments in Section 2).

Ren and Liu [9] and Guo and Long [10] introduced the connectivity of adjacent scan lines into the traditional 2D scan line seed-filling algorithm, which can remarkably reduce the repetitive seeds search in 2D space. However, the independency

* Corresponding author. Tel.: +86 010 82316747; fax:+ 86 010 82317735.
E-mail addresses: mxsf.yu@gmail.com, mxsf929@yahoo.com.cn (W.-W. Yu).

0097-8493/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2010.05.005

of scan line is lost, which means that a certain scan line cannot be used to access its two neighboring scan lines. Thus it is impossible to expand their 2D algorithms into the 3D space directly.

Oikarinen [6] proposes a special 2D seed-filling algorithm that removes the use of stack to solve the above described problems. It avoids recursions by designating neighboring pixels of the current one as children and examines them in sequence. Constant and less memory is consumed in this algorithm. However, it suffers from the time-consuming demerit, caused by repetitive rollbacks for filled pixels.

Shyan-Bin and Ming-Dar [11] extend Oikarinen's 2D seed-filling into 3D space, and presents an improved scan line seed-filling algorithm. It uses a 2D pointer array of linked lists to avoid the redundant seeds search, and classifies the relationship of neighboring scan ranges into five cases: no overlap, complete overlap, right partial overlap, left partial overlap and middle partial overlap. However, the main limitation of this algorithm is that a large and constant 2D pointer array is required, but the dimensions of different volume datasets are diverse. The algorithm will fail if a dimension value of a certain volume dataset is greater than the constant value of 2D pointer array. Therefore, a huge value that can exceed the maximum dimension value must be set for the 2D pointer array. In addition, according to our experiments (see Section 2), the frequency of stack operations is still very high.

Although the above described scan line seed-filling algorithms have remarkably reduced the time consumption and the size of stack, two demerits also exist. First, the operations of seeds stack and repetitive seeds search are still too frequent. It is because the filling sequence of adjacent scan lines is not utilized to eliminate the unnecessary repetitive seeds search. Second, for each scan line, if it is assumed to be a "seed" in 3D space, the adjacent scan lines will be accessed by using the recursion within the 4-connected region essentially. Therefore, the process of the seeds search can be considered as the flood-filling in 3D space that only one "seed" will be pushed or popped in each time. Thus the time consumption is still large.

In this paper, the existing scan line seed-filling algorithms are analyzed and their main defects are pointed out first. Then a new 3D seed-filling algorithm is proposed to search and fill consecutive 3D object based on scan slice. The main difference between the proposed algorithm and the existing ones is that the process of seeds search is based on slice in the proposed algorithm, which means that the process of seeds search is executed slice by slice to eliminate the above described defects of the existing algorithms.

2. Performance of the existing algorithms

To intuitively analyze and describe the performance and problems of existing algorithms, a region of interest (ROI) is extracted from a brain volume dataset ($512 \times 512 \times 30$) to test the performance of the existing algorithms. The hardware environment for this implementation is: Genuine Intel (R) CPU-1.66 GHz (2 CPUs), 2046 MB RAM, NVIDIA Quadro NVS-256 MB. The software environment is Visual C++6.0. The following experiments (see Sections 4 and 5) are executed in the same environment. Fig. 1 shows the extraction process of the ROI. Fig. 1(a) and (b) shows the reconstructed models of the original dataset and ROI by using marching cubes (MC) method, respectively.

The original dataset contains 7,864,320 ($512 \times 512 \times 30$) voxels, while the dataset of ROI only contains 72,072 voxels and constitutes 0.92% of the original one. The performance of algorithm is evaluated from the following aspects: the number of extracted voxels (marked with Voxels); the frequency of stack operation (marked with Frequency) including push and pop; the maximum depth of stack (marked with Depth) and the time consumption (marked with Time). Table 1 lists the performance data of three existing 3D seed-filling algorithms during the implementation of the ROI.

The first algorithm is seed flood-filling algorithm, which does not use scan lines but the simplest approach of searching seeds by recursion in 3D space. The second one is Feng's algorithm [5], while the third one is Jou's algorithm [6], both of which search seeds by scan lines in 3D space.

According to Table 1, the flood-filling algorithm consumes 11.94 min for the extraction of this small region, which is obviously not feasible for 3D space. The second and the third algorithms reduce Frequency, Depth and Time remarkably. Although Jou's algorithm is improved based on Feng's, the performance data of both are similar, which means the improvement is not as effective as expected. In addition, Frequency and Depth are still high, and the memory resource and time consumption will linearly increase with the increase in size of extracted ROI (see our experiments in Section 5). The reason is that during the implementations of the above mentioned scan line seed-filling algorithms, the scan line can be essentially assumed as a "seed point" in 3D space. Thus the seeds search is executed by recursion as well as the flood-filling algorithm.

Fig. 2 illustrates two modes of seeds search for the existing 3D seed-filling algorithms. The black and hatched blocks represent filled and unfilled seeds, respectively. Fig. 2(a) is the model of

Table 1
The performance of existing algorithms.

	Voxels	Frequency	Depth	Time
Flood-filling	72,000	72,000	16,929	11.94 min
Feng's	72,072	12,642	1251	0.142 s
Jou's	72,072	12,564	1730	0.141 s

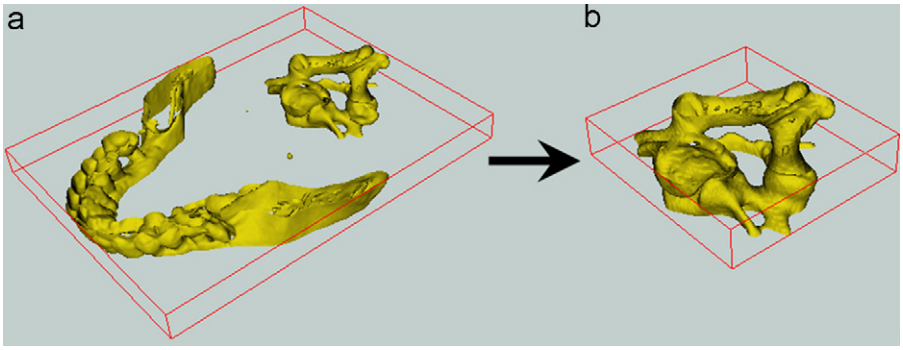

Fig. 1. The process of extracting the region of interest.