Research Article

Multiscale Feature Model for Terrain Data Based on Adaptive Spatial Neighborhood

Huijie Zhang,^{1,2} Yun Ma,^{1,2} Zhiqiang Ma,^{1,2} Xinting He,^{1,2} Yaxin Liu,^{1,2} and Zijun Feng^{1,2}

¹ School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China

² Key Laboratory of Intelligent Information Processing of Jilin Universities, Changchun 130117, China

Correspondence should be addressed to Zhiqiang Ma; mazq@nenu.edu.cn

Received 10 July 2013; Accepted 18 August 2013

Academic Editor: William Guo

Copyright © 2013 Huijie Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Multiresolution hierarchy based on features (FMRH) has been applied in the field of terrain modeling and obtained significant results in real engineering. However, it is difficult to schedule multiresolution data in FMRH from external memory. This paper proposed new multiscale feature model and related strategies to cluster spatial data blocks and solve the scheduling problems of FMRH using spatial neighborhood. In the model, the nodes with similar error in the different layers should be in one cluster. On this basis, a space index algorithm for each cluster guided by Hilbert curve is proposed. It ensures that multi-resolution terrain data can be loaded without traversing the whole FMRH; therefore, the efficiency of data scheduling is improved. Moreover, a spatial closeness theorem of cluster is put forward and is also proved. It guarantees that the union of data blocks composites a whole terrain without any data loss. Finally, experiments have been carried out on many different large scale data sets, and the results demonstrate that the schedule time is shortened and the efficiency of I/O operation is apparently improved, which is important in real engineering.

1. Introduction

Terrain model is widely applied in actual engineering, such as film, game, and simulation. One of the grand challenges is that it requires huge amounts of data set, in order to guarantee its precision. Especially in recent years, with the rapid development of spatial visualization technology, the requirement for terrain details is higher and higher; therefore, the terrain scene is more and more complex. Owing to the massive data, it is very difficult on the establishment, management, and index of terrain model in the scientific computing visualization engineering.

The capacity of RAM is limited for storing huge amounts of data set; therefore, the support of external storage is needed, when rendering a large scale of terrain. According to the requirements of scene, it is important to dynamically load data into the memory interactively for realizing to render the scene in real-time visualization engineering. However, there is a serious speed difference between RAM and external storage, so that the interactive operation between them frequently makes a big influence to the efficiency of the system directly.

It has become a bottleneck for the process of large scale data. As a result, how to establish the model with huge amounts of data set and provide effective scheduling method is one of the research hotspots.

The pyramid is a kind of data model composed by multiresolution levels [1]. It can provide all kinds of resolution of the data without real-time sampling, because the model generates the different resolution data levels of data elevation model (DEM) in advance. It has been widely applied to commercial software. However, there are two aspects of drawbacks.

- (1) The pyramid model increases the storage space of data; that is to say, a lot of data is repeated on the different layers of it. Hence, decreasing the amount of loaded data is at the cost of consuming external storage.
- (2) There is still mass of redundant data in the data set which has been loaded in memory. No matter the terrain is plain or rugged, in the classical method, the data resolution for each level is uniform, and

interlaced sampling is adopted in the various adjacent levels, which does not consider the terrain geometry feature.

Lindstrom and Silva [2] proposed the dynamic data loading method using the function of mirror file. This method is extremely simple, and it takes advantage of operation system to divide data pages automatically. Dai et al. [1] brought up a method of incremental data update, which dynamically updated local terrain data, according to the move of viewpoint and the geometry center of the data page. Li et al. [3] adopted the technique of incremental horizon to eliminate the invisible part of the model.

In order to further improve the efficiency of data schedule, grid simplified model is expressed as data stream by many scholars [4]. Space-filling curve including Hilbert curve [5], Z curve [6], and Π Curve [7] in terrain model is a common approach of linearization for data in external storage. It still takes additional time on the index of external storage even if those methods improve the efficiency of data schedule in different ways. In recent years, a new function based on cluster analysis has been proposed to simplify the data model in external storage, and improve the efficiency of data schedule [8, 9]. But the out-of-core methods based on cluster almost aimed at irregular data set. However, the primary research object in terrain modeling engineering is large scale data set of grid model; therefore, the methods mentioned above are not capable of disposing the large scale data in real engineering. In addition, the parallel algorithm using hardware to deal with out-of-core terrain data was proposed. But it is not applied widely, because of its requirement for high quality hardware [10].

This paper utilized a multiresolution out-of-core model based on geometry terrain feature. There is the following advantages (1) It consists of physical model and logic model, in which the physical model is original full resolution data, so the model is independent of the data scale. (2) The logical model is a multiresolution hierarchical structure on physical model, which establishes the index of data blocks, according to the geometry feature of terrain. (3) It decreases the amount of data loaded to memory while keeping the external storage constant.

In this paper, we proposed a strategy of scheduling data, which is based on logic model after the cluster analysis. It decreases the time cost of searching the target data in external storage, by clustering the data blocks whose static geometry errors are similar. Meanwhile, we put forward the method that sorts the data blocks in the cluster via space-filling curve and presents encoding strategy on multiresolution terrain model. It resolves two primary problems (1) One is to map terrain data blocks in two-dimension structure into the form of one dimension. (2) The other is to solve the problem of how to intercept the data blocks in cluster for the multiresolution model in a local area.

2. Related Work

As we all know, the effect of 3D modeling based on vectors is usually good; however, there are a lot of problems such as

the process of large scale data and accelerative rendering. In recent years, few scholars have achieved significant research progress in the following aspects.

Level of details (LOD) is a classical simplification method for multiresolution model. There are three categories of algorithms for LOD, that is, triangulation algorithm based on quadtree [11], simplified algorithm based on adaptive grids [12], and progressive mesh algorithm [13].

Triangulation algorithm based on quadtree is to divide one data block into four blocks according to a screen error and viewpoint. Its advantage is that it built the hierarchy using Quad-tree code, which is easy in solving the crack problem of adjacent blocks on the common edges. The representative method contains algorithm based on constrained Quad-tree [14], simplified method based on implicit restricted quadtree [15], and so on. The advantage of this kind of algorithm is that it adopts the efficient strategy to refine layers from the top to the bottom. However, a mass of information of each quadtree node needs to be stored. Thus it consumes massive storage resources.

The simplified algorithm based on adaptive regular grid evaluates whether model data should be eliminated according to model error and then determined to divide grid area adaptively [12].

Progressive mesh algorithm adopts embedded grids to remove crack between layers by setting "skirt" for different multiresolution layers [13]. The literature [16, 17] takes advantage of geometry clipping to optimize algorithm that it creates data buffer in order to accelerate real-time roaming of terrain scene. Meanwhile, Sun et al. [18] described the technique to compress the texture based on mipmap structure. Although this kind of algorithm improves model precision, it needs higher CPU consumption. So there are difficulties for general 3D vector model, when dealing with large scale data sets.

Cluster analysis is also a valid strategy of solving problems in construction of multiresolution model. It is widely applied in many areas. The literature [19, 20] described the method of solving problems of computing coordinates mean by using cluster analysis.

When rendering multiresolution terrain model, it is necessary to deal with huge scale data. Jeong et al. [21] proposed a new method to render the complex scene. It is hard to imagine rendering such complex scene in early years. However, with the development of graphic hardware, GPU has become an important computing resource to deal with huge scale data. The literature [22, 23] shows more details using this new method.

3. Cluster Analysis Based on Similar Error

3.1. Cluster with Similar Error. Multiresolution hierarchy based on features (FMRH) is a structure on the basis of physical model. According to the structure of FMRH proposed in the literature [24], a mass of redundant data was eliminated from the view of terrain features, so that the multiresolution model could be stored in the minimum capacity of external storage. However, the efficiency of scheduling the data in external storage would not be improved even though the amount of physical data decreased. In fact, the information

FIGURE 1: Multiresolution layer hierarchy of logical model in external storage.

of level of data blocks is not inheritance with its errors; that is to say, the errors of data blocks in the same level may not be similar. The error in current data block is only related to its source data block or its sub-data block, because the hierarchical structure proposed in our paper is based on a realization of depth-first traversal. As shown in Figure 1, B_1^l , B_{13}^{l+1} , B_{13}^{l+2} are the blocks on the same branch in FMRH, and they satisfy the following relation:

$$\text{CovSE}(B_1^l) \geq \text{CovSE}(B_{13}^{l+1}) \geq \text{CovSE}(B_{13}^{l+2}). \quad (1)$$

The error of nodes in the same layer such as B_{13}^{l+1} and B_{14}^{l+1} is uncorrelated, while the one of B_{13}^{l+2} and B_{14}^{l+2} is possibly great, so it cannot load data into memory as the unit of layer. Hence, it is necessary to traverse the whole hierarchical structure in external storage to load data. As a consequence, selecting data blocks of multiresolution on different levels increases the burden of rendering scene in real time.

A cluster algorithm, which satisfies spatial constraint and maximum static error of closure based on FMRH, is raised in this section. The main idea of this algorithm is to make cluster partition for nodes which own similar errors in external storage and propose the method of spatial index for each cluster. According to this strategy, it achieves multiresolution terrain data of the current scene rapidly for each frame, every error threshold, position of viewpoint and visible area, without traversing the entire FMRH structure of terrain.

According to the literature [24] and formula (1) in this paper, the static spatial error threshold can be computed, and a series of error thresholds will be gained by adjusting the parameter ρ . The bigger the threshold, the smaller the amount of extracted data and, the lower the resolution of model; otherwise, the larger the amount of data, the higher the data resolution for memory. Let $\delta_{\text{metric}}^{\max}$ and $\delta_{\text{metric}}^{\min}$ be the maximum and minimum static spatial error threshold. $\delta_{\text{metric}}^{\max}$ equals to $\delta_{\text{metric}}^{\min}$ when establishing the FMRH; namely, $\rho = 0.03$. $\delta_{\text{metric}}^{\max}$ is the basic condition which reserves the profile feature of terrain, so it adopts $\rho = 0.2$ in the experiment. The threshold $\delta_{\text{metric}}^{\min}$ aims at guaranteeing the required higher resolution data in memory by FMRH, while the goal of threshold $\delta_{\text{metric}}^{\max}$ is to provide the lower resolution data which maintains the profile feature of terrain with the lowest requirement of the

simplified model. Thus, the static error in all data blocks of FMRH ranged from $\delta_{\text{metric}}^{\min}$ to $\delta_{\text{metric}}^{\max}$.

Definition 1. Let $\delta_{\text{metric}}^1, \delta_{\text{metric}}^2, \dots, \delta_{\text{metric}}^i, \delta_{\text{metric}}^{i+1}, \dots, \delta_{\text{metric}}^m$ be the static error threshold, and satisfy the constraint: $\delta_{\text{metric}}^{\min} \leq \delta_{\text{metric}}^1 \leq \delta_{\text{metric}}^2 \leq \dots \leq \delta_{\text{metric}}^i \leq \delta_{\text{metric}}^{i+1} \leq \dots \leq \delta_{\text{metric}}^m \leq \delta_{\text{metric}}^{\max}$. Assume the set $G = \{\xi_1, \xi_2, \dots, \xi_{m-1}\}$ and each ξ_i is in the set of data block B_{index}^l in different layers; then define ξ_i as a cluster based on $\delta_{\text{metric}}^{i-1}$ and δ_{metric}^i (similar error cluster, SEC), where it satisfies the following constraints:

- (1) $\delta_{\text{metric}}^i \geq \text{CovSE}(B_{\text{index}}^l) \geq \delta_{\text{metric}}^{i-1}$, where l can be the index of every spatial layer in the structure of FMRH,
- (2) $\bigcup_{B_{\text{index}}^l \in \xi_i} B_{\text{index}}^l = \Omega$, where Ω denotes an entire closed terrain space.

3.2. Cluster Algorithm Based on FMRH. A cluster instance is described as Figure 2, and Figure 2(a) is a cluster, which satisfies the constraints in Definition 1 as well. The data blocks marked within a transparent rectangle build the terrain area in Figure 2(b). These data blocks distribute on four levels in FMRH and satisfy the uniform error threshold. These data blocks obviously construct a closed space area which is to be proved in Section 3.3.

According to algorithm of construction of FMRH [24], it records the static error of closure of each data block in FMRH when constructing the FMRH model. Hence, the work is to traverse the structure of FMRH and put data blocks which own similar static error into the proper clusters, according to their static error. In order to satisfy the second constraint in Definition 1, traversing the structure of FMRH by depth-first strategy is better. More details of the cluster algorithm based on FMRH are shown as follows (Algorithm 1).

According to Algorithm 1, the data block in the same interval is aggregated into one cluster, while these different resolution data blocks distribute in different layers, and their union covers the terrain space of FMRH.

3.3. Proof of Cluster Space Closeness Proposition. The majority of data block with different resolution on the same branch distributes into different clusters by depth-first traverse, and the static error satisfies formula (1) on a branch from the top to the bottom in FMRH. The special case is that the error of data blocks, which are partly adjacent or distributed in closed layers, are similar on the same branch, so that they locate in the same error threshold interval. As you can see in Figure 3, three branches are marked by black dotted lines. Assume that there is small difference between $\text{CovSE}(B_{13}^1)$ and $\text{CovSE}(B_{134}^2)$; B_{13}^1 and B_{134}^2 are probably classified into one cluster, only through error threshold which data block belongs to. But it goes against the second constraint in Definition 1; in fact there is an overlapped partition in space between B_{13}^1 and B_{134}^2 .

In order to solve this problem, the current_block is not classified as cluster ξ_i , if there is ancestor data block in the current cluster ξ_i , in Algorithm 1. Thus, there is no overlapped area in the covered place of different data blocks for each

Input: The structure in external storage of FMRH.

Output: The set of cluster $G = \{\xi_1, \xi_2, \dots, \xi_{m-1}\}$.

Description: This algorithm generates the set of cluster by traversing the structure of FMRH with depth-first order, according to the relationship of static error and threshold.

(1) Initialization:

(1.1) According to Definition 1, set

$$\delta_{\text{metric}}^1, \delta_{\text{metric}}^2, \dots, \delta_{\text{metric}}^i, \delta_{\text{metric}}^{i+1}, \dots, \delta_{\text{metric}}^m \text{ and make } \delta_{\text{metric}}^{\min} \leq \delta_{\text{metric}}^1 \leq \delta_{\text{metric}}^2 \leq \dots \leq \delta_{\text{metric}}^i \leq \delta_{\text{metric}}^{i+1} \leq \dots \leq \delta_{\text{metric}}^m \leq \delta_{\text{metric}}^{\max}.$$

(1.2) Clear the clusters $\xi_1, \xi_2, \dots, \xi_{m-1}$ in aggregation G .

(2) Recall the function of **Traverse_FMRH** (FMRH, B_1^0 , 0).

(3) Generate the index file of clusters, and save the index of data block for each cluster.

(4) The end.

Traverse_FMRH (FMRH, current_Block, level_S)

Function Description: Load static error of data blocks in FMRH, and decide which cluster it belongs to.

Parameter Description: FMRH is a multi-resolution level of structure based on features. The parameter current_Block is the current data block. The parameter level_S is the index of current data block in the level.

(1) Load the static error of current_Block from FMRH as current_Block.CovSE.

(2) Decide which cluster current_Block belongs to:

(2.1) Construct $m - 1$ intervals according to m error

$$\text{thresholds } \delta_{\text{metric}}^1, \delta_{\text{metric}}^2, \dots, \delta_{\text{metric}}^i, \delta_{\text{metric}}^{i+1}, \dots, \delta_{\text{metric}}^m$$

(2.2) Each threshold interval corresponds a cluster from $i = 1$ to $m - 1$:

(2.2.1) Decide the size of current_Block.CovSE, if it meet two constraints:

- ① $\delta_{\text{metric}}^i \leq \text{current_Block.CovSE} \leq \delta_{\text{metric}}^{i+1}$;
- ② add data block current_Block into the cluster ξ_i , if there is no ancestor data block of current_Block including in the cluster ξ_i .

(2.2.2) When current_Block.leaf is false, then, get four data blocks sub_Block _{j} in the next level from current_Block in FMRH, and recall the function of **DeepTraverse_FMRH**(FMRH, sub_Block _{j} , level_S + 1).

(3) The end.

ALGORITHM 1: The cluster algorithm based on FMRH.

FIGURE 2: A cluster instance based on similar static error. (a) Structure of FMRH (transparent boxes denote data blocks which are gathered as the same cluster of the one corresponding to (b)). (b) The space closed terrain area of cluster corresponding to (a).

FIGURE 3: Analysis of problems about depth-first traversing FMRH.

cluster, and the union of all data blocks constructs the completed closed terrain. The following proof attempted to demonstrate the inevitability in theory.

Theorem 2. Let $G = \{\xi_1, \xi_2, \dots, \xi_{m-1}\}$ be the set of cluster generated by Algorithm 1; then for any ξ_i , $\bigcup_{B_x^{l'} \in \xi_i} B_x^{l'} = \Omega$ is held, where Ω denotes a completed terrain closed in the space.

Proof. Use induction method to prove the theorem. Let l be the maximum spatial layer in FMRH.

When $l = 0$, there is only one spatial layer in FMRH_0 .

According to Definition 1, $S^0 = \{B_1^0\}$ is consisting of single data block, so that the space is obviously closed.

Assume the theorem holds for any $\xi_i, l' \leq k$ and $\bigcup_{B_x^{l'} \in \xi_i} B_x^{l'} = \Omega$, when $l = k$. So that for FMRH_{k+1} a layer of data block is increased (denoted as Path A) on branch of each FMRH_k or remaining the branch (denoted as Path B), when $l = k + 1$. In the case of Path B, since there is no change on branch, all clusters, generated on the basis of FMRH_k , have already been closed in the space, when $l = k$. As a consequence, the closeness proposition holds. The remaining work is to prove spatial closeness proposition for the branch of Path A in cluster.

Without loss of generality, we chose any branch on Path A (shown as Figure 3) for any cluster ξ_i . According to the method of depth-first traversal in Algorithm 1, for the branch on Path A from the top to the bottom, the current data block $B_x^{l'}$, where $1' \leq k + 1$, shown in Figure 3, is divided into two categories. If the ancestor data block of $B_x^{l'}$ is included in ξ_i , it will no longer be in ξ_i , before traversing the $k + 1$ layer. On the contrary, $B_x^{l'}$ and its three brother data blocks are listed into ξ_i , conducted by recursive traversal for four sub-data blocks in step 2.2.2. For the former situation, keep ξ_i constant; thus the closeness proposition is held. $B_x^{l'}$ and its three brother data blocks are overlapped with their father data blocks, so that the latter one still keeps the closeness proposition hold. This ends the proof of Theorem 2. $\square$

4. Index of Levels of Data Blocks in Cluster

The union of entire data blocks in each cluster composites a closed space of terrain through our algorithm. When loading

data, a proper cluster is obtained accurately as long the scope of error threshold as is ascertained. Therefore, it realizes multiresolution data loading with no redundancy, which does not need to search and judge further. However, one frame roamed in a scene just requires a small part of data to load in memory. Thus, the work of encoding and sorting data blocks in each cluster by characteristics of space is necessary in order to establish the relationship of multiresolution model of level and data blocks in cluster. It realizes the index of local data blocks rapidly as well.

As a matter of fact, the reconstruction of large scale grid data is applied widely in terrain visualization, volume rendering, and matrix operation, and its advantage is that it verifies data access locally. The best strategy of linearization of multidimensional data is using the technique of space-filling curves. The reconstructed data can be intercepted as the rule of filling, according to the requirements, while the closeness proposition in space is still held. This characteristic is corresponded to the goal of indexing the feature cluster and takes advantage of the local proposition of space-filling curves. It can also realize the interception of data in any scope.

4.1. Strategy of Hilbert Space-Filling Curves. This paper focuses on the space-filling curves of two-dimension, because the data of elevation can be mapped into the a tow-dimension space. The characters of these curves are listed as follows. (1) They not only verify to traverse every data block in the space, but the data blocks are adjacent in local area of filling curves. (2) It is easy to form the triangle strips rendering real time, and the public vertices can be used more than once. It takes advantage of the modern graphic hardware resource. These papers adopted Hilbert curves to guide the process of index coding for data blocks in cluster, so that it improves the efficiency of data loading of terrain.

The basic approach of generating Hilbert curves, formed with the process of subdividing the current square to four small ones and connecting the center of these squares, is recursion. The literature [5] proposes a serial of indexes for Hilbert curve along different coordinate's axes. However, the Hilbert curve cannot be put into use directly, because data blocks in each cluster are presented with multiresolution.

It is necessary to consider the level information of data blocks, in order to apply it to Hilbert curve in the multiresolution model. The train of thought is as follows. First of all, ascertain the filling order of children data blocks where its number is less than or equal to four, which is determined by their father data blocks, by using the level information of data block in cluster. The data block traversed forms a multiresolution filling curve in a certain arrangement from the top to the bottom, by executing the process of above judgments. The details are stated as follows.

- (1) Divide the data block into four types in cluster ξ_i as type I, II, III, and IV. This division plays two roles in classifying the data blocks. On one hand, it can make sure the sequence in space-filling curve of the sub-data block is of current data block. On the other hand, it takes advantage of self-similarity of Hilbert curve to summarize rules of production. Therefore, the type of

sub data block is ascertained, and it can be used for recognizing where the position of data block of next layer in the curve is.

- (2) Sort sub-data blocks, according to the current type of data block. For type I, sort sub-data blocks as mark 0 to 3. For type II, sort sub-data blocks as mark 0 to 3. For type III, sort sub-data blocks as mark 0 to 3. For type IV, sort sub-data blocks as mark 0 to 3.
- (3) Let the type of current data block be T_B , and the type of its child's data block is probably T_0, T_1, T_2 , or T_3 . In order to make sure of the type of its child's data block, the following rules of production are proposed.

Rule 1. If $T_N = I$ then $T_0 = III, T_1 = I, T_2 = I$, and $T_3 = IV$.

Rule 2. If $T_N = II$ then $T_0 = IV, T_1 = II, T_2 = II$, and $T_3 = III$.

Rule 3. If $T_N = III$ then $T_0 = I, T_1 = III, T_2 = III$, and $T_3 = II$.

Rule 4. If $T_N = IV$ then $T_0 = II, T_1 = IV, T_2 = IV, I$, and $T_3 = I$.

4.2. Algorithm of Generating Multiresolution Model Using Hilbert Space-Filling Curve. The necessary condition that each cluster can be sorted through space-filling curve is proved as follows. Due to Definition 1 and Theorem 2, there is only one data block on the same branch in the same cluster, and the set of all data blocks in a cluster composites a closed space area. Thus, cluster is a set with no intersections and spatial closeness data blocks. An instance of cluster is shown as follows.

By traversing data blocks in cluster from layer l , top layer in cluster, to layer l' , the lowest layer in cluster, it forms space-filling curves. Data blocks are sorted in cluster, because each cluster is produced by the manner of depth-first recursion. The principle is that it backtracks to the original data blocks when a branch traverses all of the data blocks in current cluster, and then executes the depth-first recursive operation of their brother data blocks. As a result, adjacent data blocks selected in cluster are the nearest ones in FMRH space. As shown in Algorithm 2, taking an example for a cluster, more details of generating space-filling curves are described.

An instance of coding cluster is as follows. Taking the area of left bottom corner as an example, it contains 11 data blocks $B_{11}^1, B_{111}^2, B_{112}^2, B_{1131}^3, B_{1132}^3, B_{1133}^3, B_{1134}^3, B_{1141}^3, B_{1142}^3, B_{1143}^3$, and B_{1144}^3 . Checking the first and the last data blocks of the original cluster, their types can be deduced, supposing the type of the top layer l is type III. Therefore, the type of data block B_{11}^1 is type I, according to rule 3 in Algorithm 2. As sorting its sub-data block by type I, $B_{111}^2, B_{112}^2, B_{113}^2, B_{114}^2, B_{111}^2$, and B_{112}^2 are coded directly as 0 and 1, because they are data blocks in cluster. As for B_{113}^2 and B_{114}^2 , the algorithm examines their sub-data blocks furthermore because they are not in the cluster. The types of B_{113}^2 and B_{114}^2 are type I and type IV, respectively, because of the type of B_{11}^1 is type I.

The sub-data blocks $B_{1131}^3, B_{1132}^3, B_{1133}^3$, and B_{1134}^3 of B_{113}^2 are not changed after sorting because the type is type I. In the

same way, the sub-data blocks $B_{1141}^3, B_{1142}^3, B_{1143}^3$, and B_{1144}^3 of B_{114}^2 have been changed after sorting as $B_{1143}^3, B_{1142}^3, B_{1141}^3$, and B_{1144}^3 ; therefore, the codes for them are $B_{1143}^3 \rightarrow 6, B_{1142}^3 \rightarrow 7, B_{1141}^3 \rightarrow 8$, and $B_{1144}^3 \rightarrow 9$. As a result, the filling curve in area of the left bottom corner consists of $B_{11}^1, B_{111}^2, B_{112}^2, B_{1131}^3, B_{1132}^3, B_{1133}^3, B_{1134}^3, B_{1143}^3, B_{1142}^3, B_{1141}^3$, and B_{1144}^3 .

4.3. Update Model with Large Scale Data Sets. The out-of-core management framework allow the update of original data sets and expansion of new data sets. The steps of updating the physical model are as follows.

- (1) Dividing new data source as square physical data block on the basis of the form of $n \times (2^m + 1) \times (2^m + 1)$. For the irregular shape data source, the manner of mending boundaries is adopted to form many physical data block files. Then arranging physical file blocks, it remains the original irregular size of terrain, which satisfies the condition of $(2^m + 1) \times (2^m + 1)$ for each file size.
- (2) Storing a physical block used head file and data body file. Replacing the old data file with new data file, if it is the one to be updated, and then modifying the head file; otherwise, adding new head file and body file into physical model. For each physical block, establishing the corresponding logical model, and adding cluster analysis. The steps of updating logical model are as follows.
 - (1) Construct the structure of FMRH.
 - (2) Get the set of cluster G , and make cluster analysis for FMRH.
 - (3) Establish the adjacent cluster set G' for each couple of neighboring physical file blocks.

5. Multiresolution Data Schedule

The majority of data schedule methods is view-dependent, which computes the error of simplified model at the same time to schedule data, so that it realizes loading multiresolution model, in order to decrease amount of loaded data in memory. The efficiency of the operation for I/O is low, because the burden of schedule program is too heavy. The multiresolution model is established directly with calculation, as long as it is achieving the error threshold of model, because the schedule program is based on multiresolution logical model and data block cluster proposed previously. Figure 4 describes the framework of data schedule based on the above thinking. The majority of works has been completed already in Sections 3 to 4. The questions still need to be solved are as follows.

- (1) Establishing the relationship between screen error of view-dependent model and the static error of closure, so that it ascertains the error threshold of loaded data in memory, according to the requirements for memory, and determines to which cluster the data source belongs.

Input: Cluster ξ .
Output: Space-filling curve (code arrangement of data block) CodeString.
Description: This algorithm generates the space-filling curve which contains all data blocks in cluster ξ , by judging the type and position of data blocks in cluster ξ .

- (1) Initialization:
 - (1.1) Define the top layer l and bottom layer l' in cluster ξ .
 - (1.2) Extract the first and the last data block from cluster ξ .
According to this, decide the type T of data block B^l in layer l .
 - (1.3) Clear CodeString.
- (2) Recall the function **DeepTraverse_Cluster** (B^l, T, l).
- (3) Modify the index file of cluster, and save the new index code of data blocks in cluster.
- (4) The end

DeepTraverse_Cluster (currentBlock, Type, l)
Function description: As Hilbert space filling curve, generate code of data block in cluster.
Parameter description: currentBlock is the current data block, Type is the type of current data block, l is the layer of current data block.

- (1) Get the four data blocks B_1, B_2, B_3 , and B_4 of currentBlock.
- (2) Sort B_1, B_2, B_3 , and B_4 , as the type of Type (shown in Figures 4(a)–4(d)).
- (3) Judge the data block is whether in cluster ξ :
 - (3.1) If B_i is a data block in cluster ξ , then encode B_i by the order after sorting, and add it into CodeString, where $i = 0, 1, 2$, and 3 .
 - (3.2) If B_i is not a data block in cluster ξ , then ascertain the type of B_i by using rules, and make the recursion furthermore, where $i = 0, 1, 2$, and 3 .
 - (3.2.1) If Type is I, then the types of B_1, B_2, B_3 , and B_4 are Type III, Type I, Type I, and Type IV.
 - (3.2.2) If Type is II, then the types of B_1, B_2, B_3 , and B_4 are Type IV, Type II, Type II, and Type III.
 - (3.2.3) If Type is III, then the types of B_1, B_2, B_3 , and B_4 are Type I, Type III, Type III, and Type II.
 - (3.2.4) If Type is IV, then the types of B_1, B_2, B_3 , and B_4 are Type II, Type IV, Type IV, and Type I.
 - (3.2.5) According to the new order, judge data blocks on next layer
- For $i = 1$ to 4:
If $l + 1 \leq l'$, then recall
DeepTraverse_Cluster (B_i , Type B_i , $l + 1$),
where Type B_i denotes the new type of data block B_i .
- (4) The end of function.

ALGORITHM 2: The algorithm of generating multi-resolution model of Hilbert space-filling curve.

- (2) Locating and gaining the real data block according to the index of cluster in physical model.

5.1. Searching Strategy of Target Cluster. The view-dependent simplification technology is that it selects the proper multi-resolution terrain model according to the position and direction of viewpoint. Finer layer is used for terrain near the viewpoint, and coarser layer for the area is abundant to the viewpoint. We adopt the static error of closure to realize the multi-resolution model in external storage, because there is no information of viewpoint, when organizing the data in

external storage. In the schedule program, determining the needed physical data block in practice, which takes advantage of static error, should be calculated according to the error of screen to select the proper index of cluster. The relation between the screen error of view-dependent model and the static error of closure is set up in this section. Let τ be the dynamic error threshold of viewpoint model, and let ε be the static error threshold. The relation is stated below:

$$\varepsilon = \frac{\tau}{\lambda} \cdot P_{\min}. \quad (2)$$

FIGURE 4: Framework of data schedule in external storage.

FIGURE 5: Five types of position relation of index of multifile data block.

Let λ be the number of pixels of screen in a unit distance, and let $P_{\min}$ be the nearest distance from the viewpoint to screen. Applying formula (2), set the value of τ , such as 0.5, 1, 1.5, or 2, which are the values of pixel. So the required static error threshold ε of current multiresolution scene can be obtained. Then the index of cluster is ascertained based on the size relation between the cluster static error threshold preestablished in advance and that in ε . Assume that the arrangement of static error threshold in cluster which is $\delta_{\text{metric}}^{\min} \leq \delta_{\text{metric}}^1 \leq \delta_{\text{metric}}^2 \leq \dots \leq \delta_{\text{metric}}^i \leq \delta_{\text{metric}}^{i+1} \leq \dots \leq \delta_{\text{metric}}^m \leq \delta_{\text{metric}}^{\max}$ is in the interval between δ_{metric}^i and $\delta_{\text{metric}}^{i+1}$, so that the multiresolution data block recorded by cluster ξ_i loads into memory.

5.2. Data Prefetching and Strategy of Incremental Data Schedule. For data prefetching operation, double-threads pattern is adopted, in order to render the terrain system in real time and guarantee roaming the scene coherently. Incremental data schedule is used to decrease the amount of data loaded in memory, by taking advantage of the correlation of different frames in space and time. The main thread is used to render foreground scene, and the other one realizes preschedule of data blocks stored in external storage.

As for single file, apply the index of cluster in Section 5.1 to ascertain the selected cluster ξ_i . Determine the code of the beginning and the end data block in cluster ξ_i , according to the required position and size of data in the current scene. For example, assuming that the codes of the beginning and the end block are 10 and 24, this part of multiresolution data clips the required data and loads into memory, through searching

the related file of physical data block from the structure of FMRH.

The aim is to locate data blocks in which files for the multifile data schedule operation. It is easy to determine the required data blocks, by traversing latitude and longitude of physical data blocks contained in it and position relation of current scene according to the index of head file of each data source. There are only five kinds of position relation as shown in Figure 5, because data blocks are classified by rules. So a set of cluster corresponds with each physical data block. The method in Section 5.1 selects the proper cluster ξ_i for each physical data block in the set of cluster and implements the same process for each ξ_i as single file. In addition, it is very important to match them using the adjacent cluster of the boundary of two files. Finally, we prepare the gotten data by the file mapping technology in data buffer.

Most operation of data schedule is for single file; the amount of data for one frame in roaming scene is not large. The maximum number of files is four, when it satisfies the condition of multifile (illustrated in Figure 5), so that it creates four file mappings at most. This question can be solved using file mapping API supported by VC. Another simple method, which can improve the efficiency of data schedule and decrease the amount of loaded data, is using incremental data schedule. A method which is similar to the literature [25] is adopted, which forms the supplantation data block of strip shape on the direction of which viewpoint changes and puts them into a data buffer, in order to support data for next frame. The difference in our method against the above method is based on multiresolution data strips of the FMRH

Input: The dynamic error of view-dependent τ ,
 data scale of current data of scene m , data
 position: longitude LONG, latitude LAT.
 Output: Pre-fetch data set D , and load file mapping buffer.
 Description of function: Locate physical data block, and
 load data buffer in memory through the threshold of screen
 error and the position of scene, as the requirement of
 the amount of data in memory.

- (1) Initialization: Clear data set D .
- (2) Traverse the head file of data source (physical model),
 and determine which data block belongs to it
 according to (LONG, LAT), meanwhile, record the number of involved files.
- (3) For each file from $n = 1$ to fileN, do the process as follows:
 //If it is a single file, then execute once,
 otherwise, execute more than once
 - (3.1) According to τ and Formula (2), compute the error
 threshold ε needed by current scene.
 - (3.2) Get cluster G , and let the arrangement of static error
 thresholds of G be: $\delta_{\text{metric}}^{\min} \leq \delta_{\text{metric}}^1 \leq \delta_{\text{metric}}^2 \leq \dots \leq \delta_{\text{metric}}^i \leq \delta_{\text{metric}}^{i+1} \leq \dots$
 $\leq \delta_{\text{metric}}^m \leq \delta_{\text{metric}}^{\max}$, determine which interval belongs to,
 and define the cluster which meets the requirements as ξ_{in} .
 - (3.3) If n is less than fileN, then go to step 3.
- (4) According to the moving direction of viewpoint, update the
 value of LONG and LAT, and establish fileN file mapping buffers.
- (5) If fileN is 1, then determine the initial position of ξ_{i1}
 (according to LONG, LAT), and intercept data block D' .
- (6) If fileN is greater than 1, then determine the position
 of cluster ξ_{in} , and let n be 1, 2, $\dots$, and fileN:
 - (6.1) Judge the relation of the file of two adjacent data block
 (as Figures 5(b), 5(c), and 5(e));
 - (6.2) Distribute data blocks in ξ_{in} , and form D' .
 According to the position relation Z-shape filling adjacent
 data block cluster after that.
- (7) If the intersection of D' and the existed data in data
 buffer is empty, then load D' , otherwise load $D' - D_{\text{old}} \cap D'$,
 namely load data incrementally.
- (8) The end.

ALGORITHM 3: The algorithm of data prefetching and incremental data schedule.

structure. But the method supposed by the literature [25] is based on the nested strip-shaped data structure. Specifically, each strip is the same resolution data block with no relation to terrain features. So the amount of loaded data is much more than the incremental data schedule. More details on data prefetching and the strategy of incremental data schedule are described in Algorithm 3.

6. Results and Discussion

We have researched on ten data sources and analyzed results in our experiment, in order to verify the model based on huge dataset in external storage and the validity of scheduling. These data sources contain GTOPO30 global elevation data, the elevation data of Jilin province, Zhujiang, and Tian-chi (crater) lake of the Changbai Mountains in China, Colorado Grand Canyon, Mount Rainier, Crater Lake, Puget Sound, Seattle, and Yakima in America. The sum amount of elevation data is about 8,111,735,217 Byte ≈ 7.554 G, and the one of

texture is 21.37 G. We used a PC with a 3.0 GHz Pentium 4 CPU, 1 GB system memory, and a Geforce 5900fx graphic card. We also used VC++ as the IDE for software developing.

We have experimented with the amount of loaded data in memory and schedule time for 10 datasets by setting error threshold τ 1 and 2.5, respectively. The resolution of screen is 1024×768 , where the clipped window size of GTOPO30 and Zhujiang is 129×129 , the one of Jilin province, Tian-chi crater, Colorado Grand Canyon, Puget Sound, and Mount Rainier is 257×257 , and the one of Seattle, Yakima, and Crater Lake is 513×513 .

Table 1 describes the result of the time and the amount of loaded data, when screen error is 0.5 pixels. There are four situations. The first one is loading data from physical block model directly. Without loss of generality, levels of detail are adopted as well, which is similar to pyramid model, but the data is the same resolution loaded in the same time. The second one is loading multiresolution data block through the FMRH structure in our logic model. The third one is

TABLE 1: Time and amount of data load when screen error is 1 pixel.

Data source	Using physical model directly		Using logic structure FMRH		Schedule time using cluster analysis (s)	Schedule time using space-filling curve (s)
	Loaded triangles	Schedule time (s)	Loaded triangles	Schedule time (s)		
GTOPO30	29,491	0.18	22,938	0.15	0.08	0.06
Jilin province	115,343	0.52	89,129	0.48	0.26	0.20
Zhujiang	29,164	0.20	21,955	0.13	0.06	0.03
Tian-chi	120,586	0.56	95,683	0.51	0.24	0.12
Colorado Grand Canyon	125,829	0.55	98,304	0.53	0.23	0.11
Seattle	477,102	2.20	367,002	1.90	0.98	0.70
Yakima	466,616	2.10	356,516	1.87	0.90	0.69
Crater Lake	498,074	2.03	398,459	1.96	0.94	0.70
Puget Sound	121,897	0.70	98,304	0.54	0.29	0.18
Mount Rainier	125,829	0.69	95,683	0.53	0.26	0.15

TABLE 2: Experiment result of the quantity of data load in different frames when screen error is 2.5 pixels.

Data source	1st frame		10th frame		100th frame		1034th frame	
	Loaded triangles	Schedule time (s)	Loaded triangles	Schedule time (s)	Loaded triangles	Schedule time (s)	Loaded triangles	Schedule time (s)
GTOPO30	19,661	0.04	1,967	0.010	1,804	0.009	1,924	0.010
Jilin province	76,022	0.16	4,842	0.050	4,947	0.052	4,859	0.050
Zhujiang	18,678	0.02	2,241	0.010	2,306	0.015	2,598	0.015
Tian-chi	85,289	0.10	5,063	0.070	49,040	0.067	5,813	0.720
Colorado Grand Canyon	83,886	0.09	5,381	0.030	5,412	0.035	5,569	0.035
Seattle	319,816	0.65	10,231	0.400	10,059	0.390	10,952	0.420
Yakima	325,059	0.60	10,565	0.360	10,884	0.370	10,237	0.340
Crater Lake	309,330	0.61	10,967	0.390	10,932	0.390	11,098	0.410
Puget Sound	85,197	0.15	4,568	0.069	4,623	0.069	4,754	0.070
Mount Rainier	83,886	0.12	4,922	0.060	4,867	0.053	4,733	0.051

loading data block by using the hierarchical cluster to index FMRH structure. The fourth one is loading data block by using cluster to index FMRH structure, with encoding and arranging nodes in cluster through space-filling curve.

Three results can be obtained from Table 1. On one hand, compared with using physical files, the data quantity in memory and the time of data schedule are reduced apparently after using the logic structure of FMRH. On the other hand, data quantity loaded in memory is invariant; however, the time of data schedule is shortened obviously, after using the data schedule of cluster analysis. In addition, it is significant to use space-filling curve on the base of cluster analysis, so that the time of data schedule is further shortened. The main reason for this result is that the physical model has been simplified according to the ups and downs of terrain by the structure FMRH. So a mass of redundant data is eliminated before loading in memory. It ensures loading small scale of multiresolution data into memory each time, so that the time of data schedule is decreased. Another reason for this result is that the cluster analysis is taken for data block of similar errors to the structure of FMRH when establishing the logic

model. Locating the suitable cluster for data block, when data schedule is preceded, shortens the time of searching data block and realizes the high efficient data schedule. Finally, it is an efficient way which is proper to file mapping, taking advantage of space-filling curve to code and arrange data blocks in cluster. Meanwhile, owing to space-filling curve is local, the method can retrieve the local data blocks rapidly.

With the threshold of screen error increased, the quantity of data loaded in memory and schedule time decreases obviously. Data in Table 1 is the result of loading in memory at the first time, but in fact the incremental data schedule technology is adopted when it roams the scene. As a result, with the rate of frame increased, the time of data schedule and I/O response is decreased furthermore. With screen error threshold 2.5, data quantity loaded in different frames is shown in Table 2. Data quantity in first frame and the time of loading data are large, and the ones in 10th, 100th, and 1034th frame tend to be stable. Compared to the first frame, data quantity in other frames in memory is decreased obviously, and the efficiency of data schedule is increased significantly, because it adopts incremental data schedule.

7. Conclusions

This paper proposed the strategy of rapid data schedule based on cluster analysis, which improves the efficiency of I/O operation. It is very significant to expand terrain modeling method for large scale data in application of real engineering. More details follow.

Firstly, the cluster analysis method based on FMRH is proposed to solve the linearization problem of multiresolution terrain model. In details, each cluster satisfies two constraints. One is that data blocks owned similar error are classified into one cluster; the other is that the multiresolution data blocks in cluster covered the whole closed terrain spatial zone. Because data needs to satisfy some screen error threshold when it loads into memory, the first constraint conforms to the data requirement of simplified model in memory. The second constraint verifies loading data in each cluster as a unit, so that the situation of data block loss does not appear.

Moreover, the proposed strategy sorts each cluster through space-filling curve and encodes it. Because the nodes within the cluster are closed in space and obtained through the depth-first traversal. Small adjustments and local updating on the clustering can be mapped to form a one-dimensional sequence of contiguous spatial data blocks. The advantage of those strategies is that there is no need traversing the whole data set.

Finally, our strategy solves the problem of file-block splicing and adopts incremental data scheduling strategy to further reduce the amount of data loaded into memory when roaming the scene. As a result, our method improved the scheduling efficiency, which is important in real engineering. In our future work, the theory and idea about scheduling [26], planning [27], and phase transitions [28] will be introduced to improve the efficiency of terrain modeling further.

Acknowledgments

The authors are grateful for the helpful comments and suggestions of the reviewers, which have improved the presentation. This work is supported by the Research Fund for the Doctoral Program of Higher Education of China (no. 20100043120012) and supported by the national Natural Science Foundation of China for Young Scholars (no. 41101434).

References

- [1] C.-G. Dai, Y.-S. Zhang, and X.-Q. Deng, "An organization and management approach of data for real-time visualization of massive terrain dataset," *Journal of System Simulation*, vol. 17, no. 2, pp. 406–413, 2005.
- [2] P. Lindstrom and C. T. Silva, "A memory insensitive technique for large model simplification," in *Proceedings of the IEEE Conference on Visualization*, pp. 121–126, October 2001.
- [3] S. Li, J.-F. Ji, X.-H. Liu, and E.-H. Wu, "High performance navigation of very large-scale terrain environment," *Journal of Software*, vol. 17, no. 3, pp. 535–545, 2006.
- [4] H. T. Vo, S. P. Callahan, P. Lindstrom, V. Pascucci, and C. T. Silva, "Streaming simplification of tetrahedral meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 145–155, 2007.
- [5] Y. Bandoh and S. Kamata, "An address generator for an N-dimensional pseudo-Hilbert scan in a hyper-rectangular parallelepiped region," in *Proceedings of the International Conference on Image Processing (ICIP '00)*, pp. 737–740, September 2000.
- [6] P. Lindstrom and V. Pascucci, "Terrain simplification simplified: a general framework for view-dependent out-of-core visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 239–254, 2002.
- [7] J. Wu, X. Liu, and E. Wu, "Real-time rendering for out-of-core terrain model," *Journal of Computer-Aided Design and Computer Graphics*, vol. 17, no. 10, pp. 2196–2202, 2005.
- [8] E. Shatter and M. Garland, "A multiresolution representation for massive meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 2, pp. 139–148, 2005.
- [9] S.-E. Yoon, B. Salomon, R. Gayle, and D. Manocha, "Quick-VDR: out-of-core view-dependent rendering of gigantic models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 4, pp. 369–382, 2005.
- [10] D. Dalecki, J. Lebiedz, K. Mieloszyk, and B. Wiszniewski, "Efficiency of interactive terrain visualization with a PC-cluster," in *Parallel Processing and Applied Mathematics*, vol. 4967 of *Lecture Notes in Computer Science*, pp. 391–399, 2008.
- [11] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner, "Real-time, continuous level of detail rendering of height fields," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 109–118, August 1996.
- [12] H. Zhang, Y. Lü, and S. Liu, "A terrain model simplification method based on adaptive areas division," *Journal of Computer Research and Development*, vol. 47, no. 1, pp. 53–61, 2010.
- [13] H. Hoppe, "Progressive meshes," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 99–108, August 1996.
- [14] R. Pajarola, "Large scale terrain visualization using the restricted quadtree triangulation," in *Proceedings of the IEEE Visualization Conference*, pp. 19–26, October 1998.
- [15] H.-J. Zhang, J.-G. Sun, Y.-H. Lu, N. Lu, and Y.-Z. Wang, "A new simplification method for terrain model based on divergence function," *Chinese Journal of Computers*, vol. 32, no. 5, pp. 962–973, 2009.
- [16] F. Losasso and H. Hoppe, "Geometry clipmaps: terrain rendering using nested regular grids," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 769–776, 2004.
- [17] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein, "ROAMing terrain: real-time optimally adapting meshes," in *Proceedings of the IEEE Visualization Conference*, pp. 81–88, October 1997.
- [18] C.-H. Sun, Y.-M. Tsao, and S.-Y. Chien, "High-quality mipmapping texture compression with alpha maps for graphics processing units," *IEEE Transactions on Multimedia*, vol. 11, no. 4, pp. 589–599, 2009.
- [19] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions: a texture classification example," in *Proceedings of the 9th IEEE International Conference on Computer Vision*, pp. 456–463, October 2003.
- [20] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 561–566, 2005.

- [21] Y. Jeong, K. Kim, and S. Lee, "Multi-resolution depth-of-field rendering," in *Proceedings of the International Conference and Exhibition on Computer Graphics and Interactive Techniques (SIGGRAPH '12)*, p. 107, 2012.
- [22] J. D. Owens, D. Luebke, N. Govindaraju et al., "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.
- [23] J. Vanek, B. Beneš, A. Herout, and O. Stava, "Large-scale physics-based terrain editing using adaptive tiles on the GPU," *IEEE Computer Graphics and Applications*, vol. 31, no. 6, pp. 35–44, 2011.
- [24] H. Zhang, Z. Ma, M. Zhang, H. Sun, and Y. Li, "An out-of-core model for large terrain based on right-triangle irregular network," *ICIC Express Letters B*, vol. 2, no. 5, pp. 1021–1026, 2011.
- [25] F. Losasso and H. Hoppe, "Geometry clipmaps: terrain rendering using nested regular grids," in *Proceedings of the International Conference and Exhibition on Computer Graphics and Interactive Techniques (SIGGRAPH '04)*, pp. 769–776, Los Angeles, Calif, USA, August 2004.
- [26] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software*, vol. 55, pp. 10–31, 2013.
- [27] D. Cai and M. Yin, "On the utility of landmarks in SAT based planning," *Knowledge-Based Systems*, vol. 36, pp. 146–154, 2012.
- [28] J. Zhou, P. Huang, M. Yin, and C. Zhou, "Phase transitions of expspace-complete problems," *International Journal of Foundations of Computer Science*, vol. 21, no. 6, pp. 1073–1088, 2010.