ISPRS Journal of Photogrammetry and Remote Sensing 139 (2018) 30–45

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

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

Local curvature entropy-based 3D terrain representation using a comprehensive Quadtree

Qiyu Chen [a,b,c], Gang Liu [a,c,*], Xiaogang Ma [d], Gregoire Mariethoz [b], Zhenwen He [a,c], Yiping Tian [a,c], Zhengping Weng [a,c]

[a] School of Computer Science, China University of Geosciences, Wuhan 430074, China
[b] Institute of Earth Surface Dynamics, University of Lausanne, 1015 Lausanne, Switzerland
[c] Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China
[d] Department of Computer Science, University of Idaho, 875 Perimeter Drive MS 1010, Moscow, ID 83844-1010, USA

ARTICLE INFO

Article history:
Received 18 October 2017
Received in revised form 28 February 2018
Accepted 1 March 2018
Available online 4 April 2018

Keywords:
3D terrain visualization
Quadtree
Local curvature entropy
LOD
Terrain simplification
Crack elimination

ABSTRACT

Large scale 3D digital terrain modeling is a crucial part of many real-time applications in geoinformatics. In recent years, the improved speed and precision in spatial data collection make the original terrain data more complex and bigger, which poses challenges for data management, visualization and analysis. In this work, we presented an effective and comprehensive 3D terrain representation based on local curvature entropy and a dynamic Quadtree. The Level-of-detail (LOD) models of significant terrain features were employed to generate hierarchical terrain surfaces. In order to reduce the radical changes of grid density between adjacent LODs, local entropy of terrain curvature was regarded as a measure of subdividing terrain grid cells. Then, an efficient approach was presented to eliminate the cracks among the different LODs by directly updating the Quadtree due to an edge-based structure proposed in this work. Furthermore, we utilized a threshold of local entropy stored in each parent node of this Quadtree to flexibly control the depth of the Quadtree and dynamically schedule large-scale LOD terrain. Several experiments were implemented to test the performance of the proposed method. The results demonstrate that our method can be applied to construct LOD 3D terrain models with good performance in terms of computational cost and the maintenance of terrain features. Our method has already been deployed in a geographic information system (GIS) for practical uses, and it is able to support the real-time dynamic scheduling of large scale terrain models more easily and efficiently.

1. Introduction

3D terrain modeling and real-time visualization are crucial components of many virtual reality applications (Liu et al., 2017; Ruzinoor et al., 2012; Tucker et al., 2001). 3D digital terrain models (DTMs) have been widely used in numerous domains ranging from common virtual reality applications (e.g., computer games, films, simulators, 3D maps, and so on) to domain specific applications (e.g., geological disaster assessment, flight simulation, battlefield scene simulation, environmental modeling and visualization, etc.) (Bai et al., 2015; He et al., 2013; Kobler et al., 2007; Li et al., 2011; Stock et al., 2010; Yu et al., 2016).

In recent years, with the rapid development of the technologies in spatial data capture, such as high resolution satellite images and laser scanning (Kalbermatten et al., 2012), as well as corresponding data processing and modeling technologies (Maguya et al., 2013; Tarolli, 2014), massive terrain original datasets were generated, and the amount of data has risen to the level of gigabytes even for a small area. Such huge amounts of data can, no doubt, provide a more detailed representation of the real geomorphology (Lane et al., 2009; Wilson, 2012). However, rendering capabilities of modern computer hardware are not always high enough to address the massive terrain datasets, and are incapable to meet the requirements on real-time performance and higher resolution. Thus, the mismatch between the scale of massive terrain data and the real-time processing ability of the computer hardware is the main bottleneck of real-time visualization of large-scale terrain (Pajarola and Gobbetti, 2007). To narrow the gap between the huge terrain data and the computational capabilities of current

* Corresponding author at: School of Computer Science, China University of Geosciences, No. 388, Lumo Road, Wuhan 430074, Hubei, China.
E-mail addresses: qiyu.chen@cug.edu.cn (Q. Chen), liugang67@163.com (G. Liu).

https://doi.org/10.1016/j.isprsjprs.2018.03.001
0924-2716/

workstations and to address the increasing accuracy and performance requirements, original terrain datasets should be simplified, and the approaches of organizing and managing terrain models need to be optimized. Meanwhile, terrain features should be kept in the final representation to the greatest extent as possible, and the terrain models need to be dynamically represented at different resolutions.

Triangulated irregular network (TIN) and Grid are two common data models used to represent terrain surfaces (Dykes et al., 2005; Fan et al., 2004; Tan and Yao, 2007; Wu and Amaratunga, 2003; Zheng et al., 2017). They are suitable for the representation of multi-scale DTMs with different morphological characteristics, and they each have their respective technical advantages for simplifying huge terrain data. In general, Grid is utilized to represent the large scale and continuous terrain surfaces, while TIN is more suitable for the local and complicated geomorphological features with finer reproductions (Bertilsson, 2015; Xiao et al., 2013). Moreover, Grid-based simplification is fast and easy to organize, but the simplified effect is inferior to the TIN-based methods. The simplified results of TIN can be optimal, but it is hard to use TIN to efficiently manage and dynamically schedule massive terrain data (Bóo and Amor, 2009). Therefore, gird-based data model is relatively better for representing multi-resolution DTMs.

A regular grid, as one of the main types of digital elevation model (DEM), is the most common technique to represent and visualize terrain features (Lindstrom and Pascucci, 2002). Because of simple data structure and high efficiency, Quadtrees are most often used to partition a two-dimensional space by recursively subdividing it into four regions and to manage them effectively. Level-of-detail (LOD) techniques correspond nicely to the human's visual system which can perceive objects at different resolutions based on distance and angle of view. Thus, Quadtree-based LOD is often used to improve the efficiency of managing and scheduling a large-scale DEM. The corresponding algorithm was firstly proposed by Lindstrom et al. (1996), and some improved methods (Kang et al., 2015; Qiu et al., 2013; Zhang et al., 2013) were presented to enhance the performance from different angles and viewpoints. The main idea of this kind of algorithms is that the terrain is divided into small tiles to fit the similar original DEM data in a reasonable magnitude. However, with the ever-increasing amount of terrain data, the data source we are facing has changed to scattered point-cloud datasets. With the unordered data, DEM is usually reconstructed by using spatial interpolation techniques (Chen et al., 2016). The Grid-based simplification and Quadtree-based LOD are then embedded into the DEM to further enhance its efficiency (Pajarola, 2002). Nevertheless, those separate steps need more computational cost and storage consumption. A comprehensive 3D terrain representation is desired to perform directly from scattered point-cloud datasets to an optimal Quadtree.

In the above process, retaining geomorphological features and eliminating cracks are the two most important problems, which are also the common challenges of LOD-based terrain visualization. The measure used for building LOD meshes affects greatly how the terrain features are maintained in the results. Several possible topographical factors (e.g. elevation, slope and curvature) can be used to assess the quality of LOD meshes as a measure and they were introduced and dissuaded by Ujiie et al. (2012). The loss of terrain features will not only result unreal terrain visualization, but also cause fatal mistakes in scientific experiments and industrial applications. Therefore, a reasonable measure is crucial in terrain modeling and simplification. Cracks occur at the boundaries between different-resolution representations where different LODs meet due to non-matching points on either side of the LODs (Pajarola and Gobbetti, 2007). Different methods have also been developed to address this problem (Gerstner, 2003; Lodha et al., 2003; Schneider and Westermann, 2006). In order to address simultaneously both problems of loss of features and cracks, a reasonable measure and an efficient approach should be applied to increase the terrain features and to eliminate the needless cracks in the final representation of terrain surfaces.

To address the aforementioned challenges for real-time visualization of complex terrain models, in this paper we present an effective 3D terrain representation using a comprehensive Quadtree-based LOD model. Compared with the general flow, Fig. 1 shows that more tasks are integrated into the early data processing and the Quadtree construction, and all later operations are carried out directly on the comprehensive Quadtree in our workflow. Terrain simplification is considered in the process from scattered point-cloud datasets to an optimal LOD Quadtree. Thus, it decreases the computational cost and the memory consumption for organizing and scheduling massive terrain datasets. Meanwhile, the local entropy of terrain curvature is adopted as a reasonable measure to generate LOD meshes due to the local curvature entropy reflects terrain complexities of a local region so that more terrain features are kept in the final representation. Due to the edge-based Quadtree proposed in this work, cracks among the different LODs are removed thoroughly and easily by updating the Quadtree. As an additional parameter, the local entropy is embedded into the nodes of the Quadtree, and it makes the dynamic scheduling of large scale terrain surfaces more efficient and straightforward.

The remainder of this paper is organized as follows. Section 2 summarizes related works. Section 3 gives details about the four parts of our main method. Section 4 presents results from three real-world datasets. Section 5 discusses the novelty and advantage of our method from different aspects. The final section offers some concluding remarks and ideas for future work.

2. Related work

Multi-resolution terrain visualization has been studied extensively with different aims and angles (Li et al., 2004). In this section, we review related work, especially those with a focus on methods mentioned in this paper, such as terrain representations, terrain simplification and elimination of cracks.

2.1. Grid-based terrain representations

Grid-based representations possess more advantages of managing and rendering multi-resolution terrains (Kamat and Martinez, 2005) compared to TIN. It stores the geometry information in a uniform equispaced grid and each grid cell is composed of hierarchies of right triangles (HRT) (Evans et al., 2001). Pajarola and Gobbetti (2007) drew a detailed review of regular or semi-regular multi-resolution models for interactive terrain rendering. Lindstrom and Pascucci (2002) presented a view-dependent dynamic terrain meshing algorithm using a HRT-based grid. Wu et al. (2010) proposed a new terrain LOD algorithm using Quadtree which can meet the need of real-time rendering for massive terrain data. The grid model is a simple representation based on the sampling of terrain altitudes at equidistant positions (Yilmaz et al., 2004). The model is regular and a multi-resolution LOD can be easily applied, since the Grid-base model can be easily encoded using only the elevation values for each node in a grid cell.

Recently, a hybrid data structure combining regular grid and complicated TIN was introduced to balance the real-time performance and local details (Koca and Güdükbay, 2014; Paredes et al., 2012). The basic principle of this kind of algorithms is to represent the complete terrain with a grid and to further refine the special areas of interest (e.g., roads, rivers, buildings, ridges, etc.) with highly detailed TINs (Bóo and Amor, 2009). These progressive

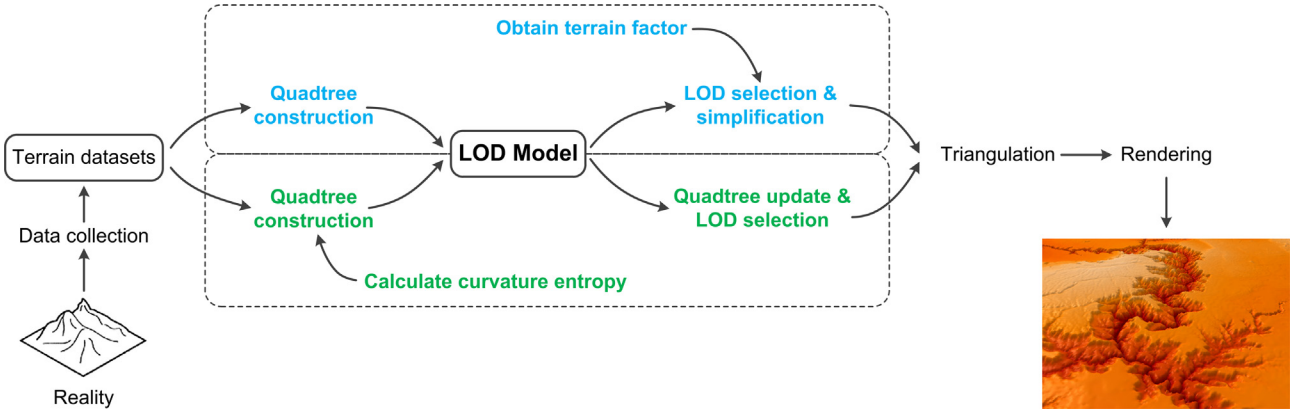

Fig. 1. Workflow of Quadtree-based terrain modeling: blue steps are the general flow according to Wilson (2012) and Yang et al. (2014); and the green ones are the workflow employed in this work.

terrain representations are better able to depict the local details of specific geographical objects, but this approach is not yet suited for managing and rendering large scale terrains in real time.

For all these reasons, regular or semi-regular approaches generally offer higher computational performance for terrain data, and they have produced some of the most efficient methods to date (Bösch et al., 2009).

2.2. Terrain simplification

Grid-based terrain visualization can usually be divided into two steps (de Floriani et al., 2004). The first step is to achieve a regular DEM by sampling or interpolating the equidistant points with different terrain altitudes using simplified terrain data. In general, a large proportion of original points can be simplified and many terrain features are lost in this step. In the second step, Quadtree-based LODs are widely used to improve the efficiency of managing and scheduling large scale terrain surfaces (Pajarola, 2002). A valid measure is usually utilized to simplify the regular DEM to achieve an optimized LOD according to terrain features (Kang et al., 2010). However, the separate approach is not an optimized strategy on both time costs and memory consumption.

The surface simplification problem has been studied extensively, as it has important applications in computer graphics and GISs (Bjorke and Nilsen, 2003; Frey and Borouchaki, 2002). It is very important to adopt a suitable measure in surface simplification. Ben-Moshe et al. (2002) proposed a new method of simplifying terrain surfaces, designed specifically a new measure of quality based on preserving inter-point visibility relationships. The curvature-featured simplification is the most common method (Cerveri et al., 2014; Jenny et al., 2011; Kobler et al., 2007). However, curvature can only describe a point or a small area of terrain features with high sensibility. If curvature is directly taken as the criterion of terrain simplification, it will cause great leaps between adjacent LODs. In contrast, entropy can be used to measure the information content of any set of data made up of discrete values such as DEM containing elevations (Wise, 2012). Thus entropy expresses the statistics of uncertainty of terrain features in a local area. Consequently, the local entropy of curvature could reduce the sensitivity, and decrease the influence of the curvature features that cannot consider larger neighbourhood.

2.3. Elimination of cracks

The Grid-based terrain LOD will inevitably cause cracks. How to repair these cracks caused by unequal subdivision granularity is a key issue in the field of 3D terrain visualization. Lodha et al. (2003) introduced an approach to remove cracks by adding vertical skirts around each block. Losasso and Hoppe (2004) used transitional zones to connect two grid cells with different LODs, thus avoiding the T-junction. The disadvantages of this kind of methods are that they cannot ensure the geometric continuity of the terrain surface, and additional topologies are inserted to the regular structure. Pajarola (1998) proposed a restricted Quadtree triangulation to avoid complex cracks, and its principle is to ensure that the discrepancy of adjacent LODs is no more than a given threshold. However it brings unnecessary grid cells and thus increases computational cost and memory consumption. Some methods pulled the high-resolution vertex to the corresponding edge of an adjacent grid cell with a low resolution to reach the closure of a crack (Duchaineau et al., 1997; Wu et al., 2010). The most obvious advantage of this kind of methods is that no additional nodes and topologies are brought in. However, it will decrease the terrain accuracy at the positions of the cracks.

2.4. Organization of a Quadtree

For the Quadtree-based LOD representation of terrain data, the organization of the nodes for a Quadtree will affect the elimination of cracks and multi-scale scheduling of LOD meshes. In the most of existing methods, each LOD tile is stored in a node of the Quadtree as a whole unit where the points of each LOD tile are organized in a logical sequence (Lindstrom and Pascucci, 2002; Pajarola and Gobbetti, 2007; Wu et al., 2010). This kind of data organization of a Quadtree is simple and easy to implement, but it is not conducive to the updates of Quadtree-based LOD meshes, such as re-segmentation, simplification, and crack elimination of terrain surfaces. Moreover, some promising attempts have been suggested to optimize Quadtree structures in the LOD representation of terrains (Lee et al., 2014; Pajarola, 2002; Plaza et al., 2004; Suárez and Plaza, 2009). Suárez and Plaza (2009) proposed a longest-edge based refinement and coarsening algorithm for the adaptive representation of terrain LOD meshes. In this method, the nodes of LODs are organized by the edges of triangles and each triangle is bisected four sub-triangles according to the longest-edge during the refinement of LODs. Such strategy is helpful for the LOD segmentation and surface simplification, but the cracks between the different LOD tiles are hard to remove. An improved Quadtree-based representation was suggested by Lee et al. (2014) where a bimodal vertex splitting strategy is employed to perform LOD selection and crack removal. The advantage of this proposal is that every node on a LOD mesh is only stored one time. But the procedure of vertex splitting needs more CPU cost so as to reduce the effectiveness of dynamic scheduling and real-time visualization for terrain surfaces.

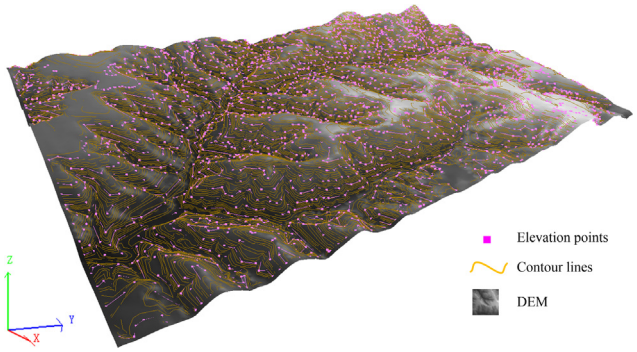

Fig. 2. Data types which can be used in our work.

3. Methodology

Our method can accommodate all types of terrain data, including the traditional survey data, telemetry data, terrain point-cloud data obtained by modern laser scanning equipment, and contour lines in GIS software (Fig. 2). However, these multi-source data have to be discretized firstly, and then the scattered points are used in the subsequent steps. The following sections draw a detailed description of the method presented in this paper from four parts.

3.1. Estimation of local curvature entropy

The concept of information entropy was first proposed by Shannon (1948) which is used to determine the uncertainty of a message or an event X with values in the set $X = \{x_1, x_2, ...x_n\}$. The probability of every event result happens is denoted as $P = \{p(x_1), p(x_2), ..., p(x_n)\}$. Entropy is the key measure for the information content, and the entropy E for event X is defined as:

$$E(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i). \quad (1)$$

Sukhov (1967) first introduced information entropy as the main method for measuring map information content. A map can be considered as a medium carrying geographic information, and the map's information content was proposed for judging the map quality (Li and Huang, 2002; Gao et al., 2017). Hence, the rendered DEM can also be regarded as the medium carrying the terrain information. Several arguments have been proposed for the use of entropy to assess DEM quality (Hu et al., 2015; Vieux, 1993; Wang et al., 2001; Wise, 2012). In these literatures, they supposed that the total number of the symbols in the map is N and that there are M types of symbols in a map. Then, for every type of symbol, its probability can be calculated as:

$$p_i = \frac{K_i}{N}, \quad (2)$$

where p_i is the probability of the i-th type of symbols, and K_i is the number of symbols of type i. The map information content is defined as the sum of entropies of every type of symbol, which can be described as:

$$E = -\sum_{i=1}^{M} p_i log_2 p_i. \quad (3)$$

In the above Eqs. (2) and (3), when the considered symbol (attribute) is terrain curvature, the local entropy of the curvature map will be obtained. Here, N is the size of the neighbourhood, namely, the total number of the elements in the local area of the curvature map; and M denotes the number of different types of

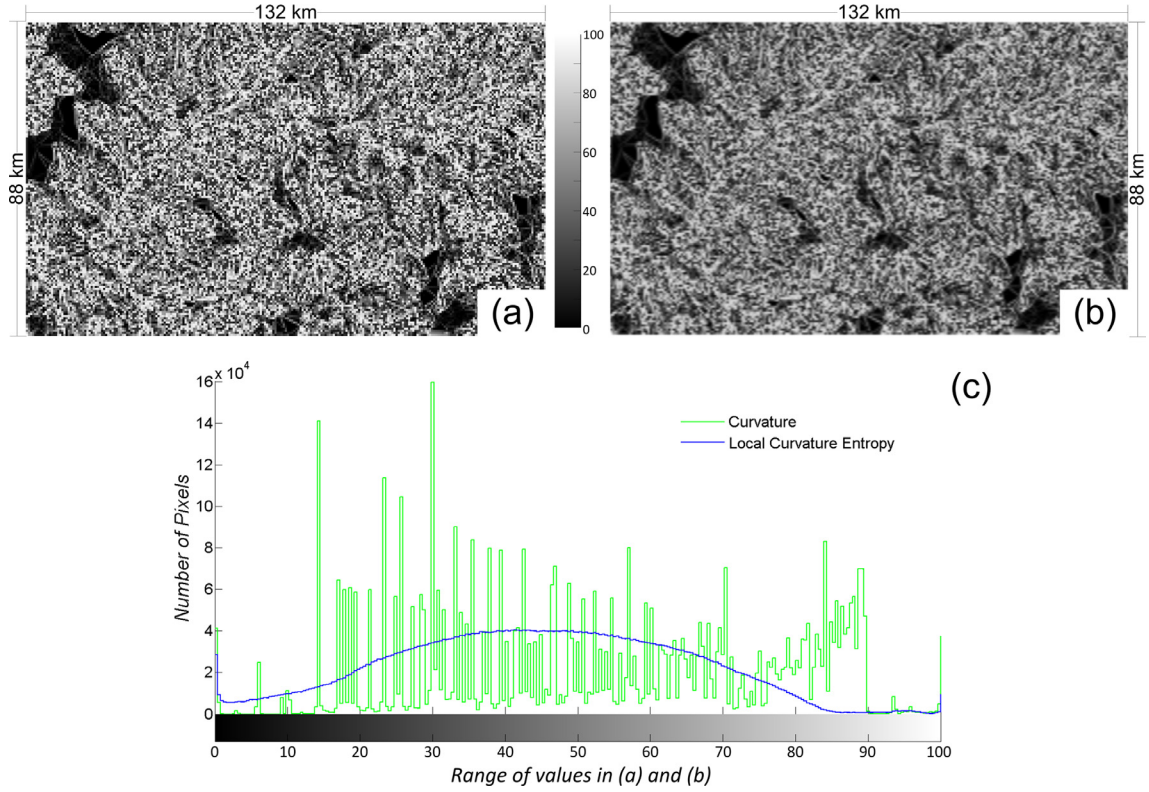

Fig. 3. Contrast between curvature and local curvature entropy: (a) curvature map of a certain terrain; (b) corresponding map of the local curvature entropy; (c) histograms of the two maps. Note that the values of curvature and curvature entropy are transformed into a same domain [0, 100].

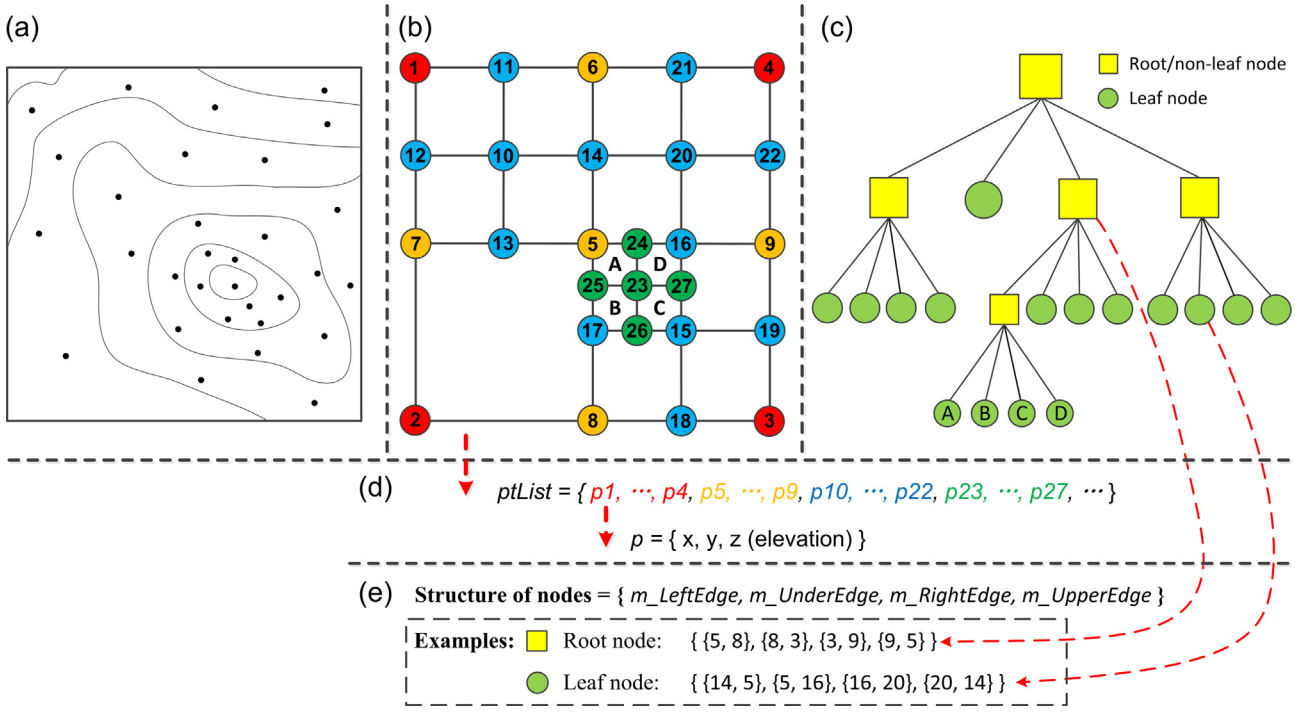

Fig. 4. Schematic diagram of the Quadtree structure and the LOD mesh presented in this paper: (a) an example of input data (elevation points and contour lines); (b) is a LOD mesh where the points are expressed in different colours according to the iterative sequences; (c) is the corresponding Quadtree; (d) presents the storage mode of the points in (b); and the structure of Quadtree nodes as well as two examples are demonstrated in (e).

curvature values in the local area. In this work, the terrain curvature (here profile curvature is utilized) map is derived from digital elevation information by using the software ArcGIS.

From the above description, it can be exposed that the local curvature entropy represents the variability of the terrain curvature in a local area with N nodes when comparing to the central node. Hence, it can reflect the complexity of terrain features. Terrain curvature is one of the essential factors for the assessment of DEMs (Cerveri et al., 2014), but it can only represent the terrain feature at a certain point. Compared with terrain curvature, the local entropy of terrain curvature is a more comprehensive measurement for terrain features. Therefore, in this study, the local curvature entropy is used as a measure to divide terrain LOD grid.

Using terrain curvature map as the input data, the corresponding entropy map *Emap* can be obtained by using Eq. (3). Fig. 3 shows the contrast between curvature and local curvature entropy. The histogram (Fig. 3c) indicates that the distribution of curvature is extremely non-stationary, but the distribution of the local curvature entropy becomes relatively smooth. Therefore, using the local entropy as the measurement of modeling and simplifying terrain surfaces will make the generated grid more stationary. Moreover, the enormous changes of the grid granularity will be avoided.

The curvature of terrains varies greatly so that it owns higher sensitivity as a measure of constructing LOD terrain meshes. After computing the local entropy of terrain curvature, the sensitivity of curvature is reduced since a larger local neighborhood is exploited in this process. The local curvature entropy not only reflects the fluctuation of terrain surfaces, but also represents terrain complexities within a certain range. At a certain point, the larger the curvature entropy is, the greater the terrain features change.

3.2. Quadtree organization and LOD segmentation

In this section, we explicitly describe the structure of the Quadtree introduced in this paper, and the process of constructing LODs based on this comprehensive Quadtree. First, we utilize a comprehensive Quadtree to integrate LOD segmentation and surface simplification into a united framework. Second, the Quadtree organized by edges is easier used to remove the cracks existing in different adjacent LODs. Third, a reasonable measure is exploited to construct an optimized LOD mesh, and the values of the local curvature entropy are stored in the Quadtree corresponding to each tile of the LODs.

3.2.1. Edge-based Quadtree

How to organize the data structure of the Quadtree adopted in this work is a key issue for the subsequent steps. A good Quadtree structure may improve the efficiency of storing and scheduling terrain data. We present an edge-based Quadtree which is illustrated in detail in Fig. 4. An example of input data for our method is demonstrated in Fig. 4a where elevation points and contour lines are involved. A LOD mesh after four iterations is shown in Fig. 4b, and the points are expressed in different colours according to the iterative sequences. It starts from a rectangle consisted of four red points (1, 2, 3, 4), and new points are added into this mesh in the order of four iterations. The corresponding Quadtree is demonstrated in Fig. 4c. In this structure, all the mesh points are stored in a list *ptList* in sequence (Fig. 4d). Every parent node *m_pParentNode* can point to four children nodes: *m_pNode_A, m_pNode_B, m_pNode_C, m_pNode_D*. For example, four tiles *A, B, C, D* shown in Fig. 4b are embedded into the Quadtree shown in Fig. 4c according to the order $A \rightarrow D$ that is suitable for all nodes in different levels. The specific iterative process will be described in Section 3.2.2. For each node of the Quadtree, it is composed of four edges (i.e. *m_leftEdge, m_underEdge, m_rightEdge,* and *m_upperEdge*) of a tile in a LOD. To reduce storage costs, only indexes of the points on each edge are recorded in this Quadtree (see Fig. 4e). Every edge contains two points in this initial step. The further processing, especially the advantage of the edge-based Quadtree will be depicted in the following section.

3.2.2. LOD segmentation

Here, we demonstrate the constructing process of a comprehensive Quadtree and the corresponding LOD segmentation in detail on the basis of the above-mentioned Quadtree structure. Any scattered terrain data might be employed in this step. Optimized LODs are generated since the terrain features are considered in the whole process.

The recursive process of Quadtree construction is shown in Algorithm 1. Facing the original scattered points, we first get the bounding rectangle S with four vertexes $p1$, $p2$, $p3$, $p4$ (see line 1 in Algorithm 1 and Fig. 4b). The max entropy $maxE$ is obtained from a local entropy map $Emap$ which is generated using the strategy depicted in Section 3.1. We then insert S into an empty Quadtree object as the root node. $p1, p2, p3, p4$ (e.g. red points in Fig. 4b) are added into the $ptlist$. From this root node, the recursive function $QuadtreeBuilding(S, maxE)$ (lines 4–17) is qualified for the construction of the Quadtree. For each recursion, the input tile S is averagely divided into four small tiles A, B, C, D, and we can easily obtain the new points $p5, p6, p7, p8, p9$. The coordinates X and Y of these new points can be obtained directly from the four vertexes of S since S is evenly divided into four parts in each iteration of this progress. But Z need to be computed by a spatial interpolation (e.g. Inverse Distance Weighted (IDW) method) according to the original scattered points. And then, if the conditions of continuing segmentation are met (line 9), the new points will be added into the $ptlist$ in order, and the four sub-tiles A, B, C, D are inserted into the Quadtree as the children nodes. D is used to control the depth of the Quadtree, and E_0 is the measurement of LOD segmentation in the constructing process of an initial Quadtree. Four new recursions (lines 12–15) are called to continue the segmentation until the conditions of every recursion are not met.

Algorithm 1 (*Constructing process of a Quadtree*).

***Input*:** *pts*: *scattered points*; E_0: *threshold of local curvature entropy*;
Emap: *map of local curvature entropy*; *D*: *max depth of the Quadtree*.
***Output*:** *a Quadtree*
1 *Get the four vertices p1, p2, p3, p4 of the bounding rectangle of pts*
2 *Get the max entropy maxE from the Emap*
3 *Insert the tile S(p1p2p3p4) into the Quadtree as the root node*
4 ***Function*:** *QuadtreeBuilding*$(S, maxE)$
5 *Segmenting the tile S into four parts* A, B, C, D
6 *Obtain the new points* $p5, p6, p7, p8, p9$
7 *Compute* $maxE_A$, $maxE_B$, $maxE_C$, *and* $maxE_D$
8 *Get the current depth d*
9 **if** $d < D \&\& E_0 < maxE$ **then**
10 *Add new points* $p5, p6, p7, p8, p9$ *into ptlist*
11 *Insert* A, B, C, D *as the four children nodes*
12 *QuadtreeBuilding*$(A, maxE_A)$
13 *QuadtreeBuilding*$(B, maxE_B)$
14 *QuadtreeBuilding*$(C, maxE_C)$
15 *QuadtreeBuilding*$(D, maxE_D)$
16 **end if**
17 **end Function**

An optimized Quadtree is generated after all the recursive calls. As a simple example illustrated in Fig. 4, all the points of the LOD mesh are stored in the $ptlist$ in order, and the four edges of every tile are stored in each node of the Quadtree. In order to avoid the repeats of the vertexes, we only record the indexes of each edge into the Quadtree nodes. So every one of these points is only stored once. $maxE$ is used to record maximum of the curvature entropy of the current tile and it is stored in the corresponding node of the Quadtree. Furthermore, the conditions of the recursions play a crucial role in Algorithm 1. The threshold E_0 is the most important measurement of LOD segmentation, and it will be assigned as an input parameter by users. It controls the progress of the LOD segmentation so as to influence the expression of the topographic details. If $E_0 < maxE$ is met, it means that this tile need to be divided continue since there are more complex terrain features in this area and it should be depicted finer. However, for an extremely complex local area and a common threshold E_0, $E_0 < maxE$ will always be met, and we will get a very deep Quadtree. Thus, another condition, the max depth D of the Quadtree is adopted to avoid the Quadtree with an extreme depth.

3.3. Eliminating cracks

Cracks usually exist in adjacent LODs, and eliminating cracks is a common issue in terrain modeling. As illustrated in Fig. 5, a crack is formed because the levels of the right big tile and left small tiles are different, and the adjacent edges P_1P_4 and $P_1P_2P_3P_4$ are not collinear. So an effective strategy was proposed to remove the cracks by adjusting the elevations of points P_2 and P_3, so that $P_1P_2P_3P_4$ is close to P_1P_4 and the corresponding crack is removed (Fig. 5c). The advantage of this strategy is that no additional points are added in the LOD and the topologies of this mesh do not need to be changed. However, the small tiles with higher resolution are driven to match the big one, which will cause the loss of accuracy of the terrain surfaces.

To address the above-mentioned issue, we add the points P_2 and P_3 to the edge P_1P_4 to remove the crack (Fig. 5d). It is easy to be implemented since the each node in our Quadtree is organized by edges. We just need to compare the edges of adjacent tiles, and add the indexes of the extra points to the edge array of the tiles with lower resolution. All of the above will be done on the Quadtree generated by Section 3.2, and we call this process Quadtree update. It is very fast due to the advantages of the tree structure itself and the edge-based organization. But after Quadtree update, we also need to change the triangulation of the low-resolution tiles because additional points are added into their edges.

As shown in Fig. 6, an initial LOD mesh has the same structure as the one in Fig. 4. To present a clearer demonstration, here we separate the tiles according to the levels they have. The Quadtree update proposed in this paper includes the following three steps.

- First update: update the parent nodes from bottom to top. For each edge of the parent node, we add the indexes of the extra points on the edges of children nodes into the edge array of the parent node in order. Repeat it until the root node. As shown in Fig. 6c, the blue points are the extra ones on the edges of children nodes when they are compared with the corresponding parent nodes.
- Second update: update the non-leaf nodes from top to bottom. We use the adjacent edges of brother nodes to update the current non-leaf node. Non-leaf nodes are the all nodes on the Quadtree except leaf nodes, and the nodes with the same depth are called brother nodes. For one edge of the current non-leaf node, we add the indexes of the extra points on the edges of brother nodes into the edge array of the current node in order. As shown in Fig. 6d, the yellow points are the extra ones due to the different levels of adjacent brother nodes. Thus the cracks caused by different levels are eliminated by comparing and updating the adjacent brother nodes at the same level.

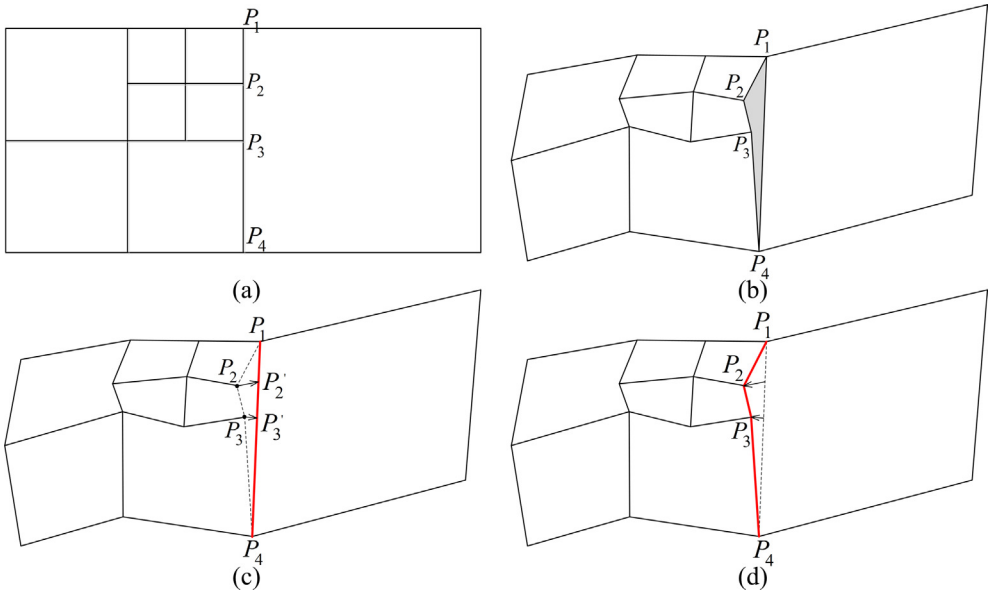

Fig. 5. Schematic diagram of the formation and elimination of a crack: (a) a simple LOD mesh; (b) forming cause of a crack; (c) eliminating strategy proposed by Wu et al. (2010); (d) our strategy for eliminating a crack.

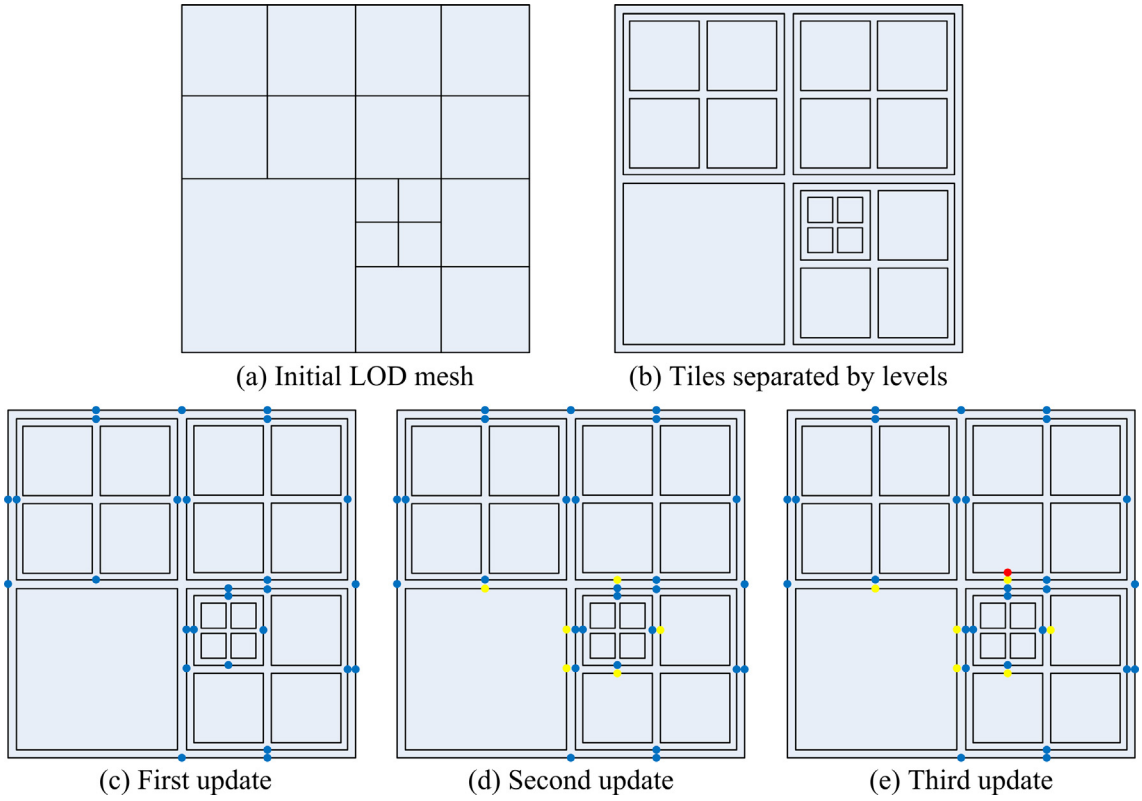

Fig. 6. Quadtree update: (a) is the initial LOD mesh; (b) is the separated tiles according to the different levels; (c) is the first update, and the blue points are the added ones on the edges of parent nodes; (d) is the second update, and the yellow points are the added ones on the edges of non-leaf nodes; (e) is the third update, and the red point is the added one on the edges of children nodes.

- Third update: update the children nodes from top to bottom. In the second step, new points might be added into the parent nodes due to the influence from the brother nodes. It causes the new unconformities between parent node and their children nodes. Therefore, we compare each edge of children nodes with the edges of the corresponding parent node, and add the indexes of the extra points on the edges of the parent node into the edge array of the children nodes. As shown in Fig. 6e, the red point is the extra one on the edge of the parent node comparing with their children nodes.

For the examples described in Fig. 4e, the edge arrays of those two nodes will be changed to the below form through the above steps.

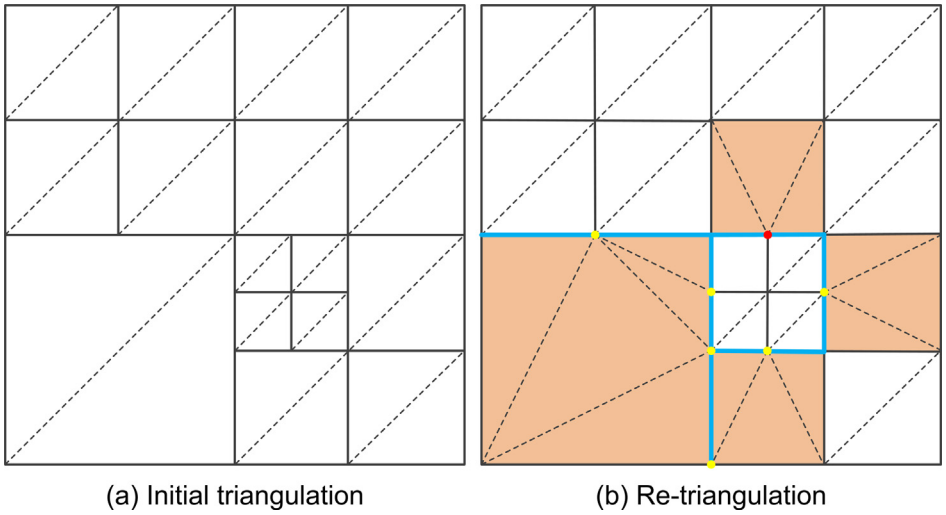

Fig. 7. Triangulation of a LOD mesh: (a) is the initial status; and (b) is the re-triangulation after Quadtree update.

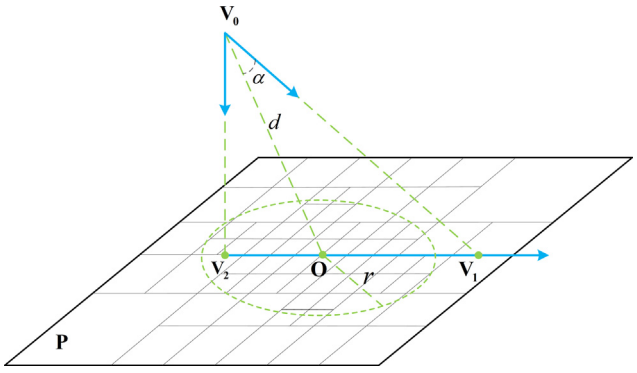

Fig. 8. Determination of visual sensitive center (adapted from Debevec et al., 1998).

Root node: {{5, 25, 17, 8}, {8, 18, 3}, {3, 19, 9}, {9, 16, 24, 5}}
Leaf node: {{14, 5}, {5, 24, 16}, {16, 20}, {20, 14}}

A crack-free Quadtree-based terrain LOD is generated after the above steps. A 3D terrain model is obtained easily by extracting all the leaf nodes of the Quadtree. Delaunay triangulation is implemented in every independent tile. Due to the new points added into the edges of leaf nodes, the topologies of the mesh should be updated. As illustrated in Fig. 7, the coloured tiles should be re-triangulated since the new points are added into their edges in blue.

3.4. Dynamic scheduling

View-dependent dynamic scheduling is the most common means for visualizing and managing lager scale terrain data. On this Quadtree generated from the above sections, the dynamic scheduling of terrain surfaces is easier to implement since the factors of the curvature entropy are embedded into the nodes of the Quadtree.

In order to get the center of visual sensitivity, the following model (Debevec et al., 1998) is introduced (Fig. 8). $\mathbf{V_0}$ is the observing point; $\mathbf{V_0V_1}$ is the direction of sight. $\mathbf{P}$ is the horizontal plane, $\mathbf{V_2}$ is the projective point of $\mathbf{V_0}$ on $\mathbf{P}$, α is the angle between $\mathbf{V_0V_1}$ and $\mathbf{V_0V_2}$. $\mathbf{O}$ is defined as the visual center:

$$\mathbf{O} = \mathbf{V_2}(1 - cos\alpha) + \mathbf{V_1}cos\alpha. \quad (4)$$

Based on the above model, we present a new strategy to get the threshold E of curvature entropy for each tile on the LOD grid obtained by the above-mentioned method. It is assumed that the

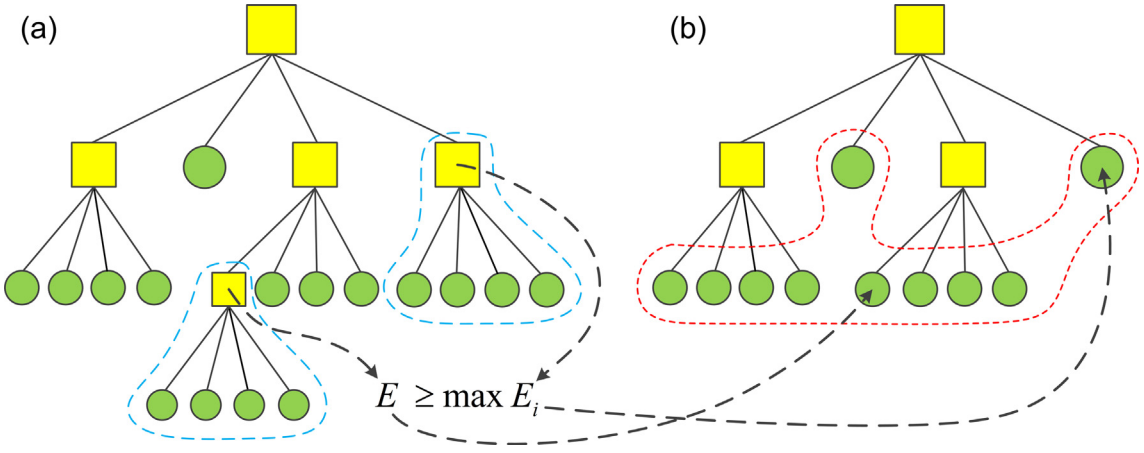

Fig. 9. Dynamic scheduling of the Quadtree considering terrain features. (a) initial status; (b) new status after shifting the viewpoint.

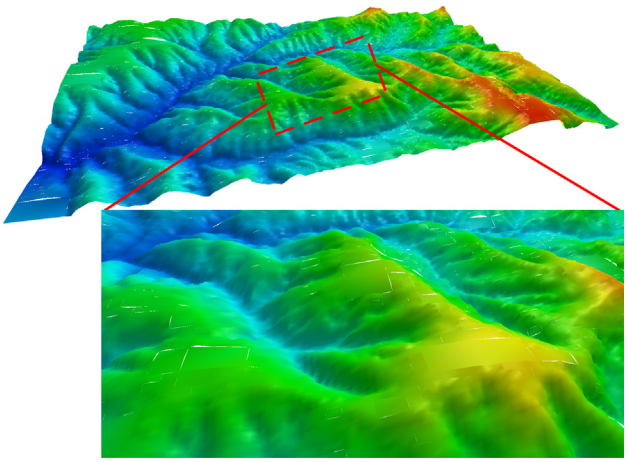

Fig. 10. A Quadtree-based terrain surface (before eliminating cracks).

threshold of curvature entropy in the above constructing of the original Quadtree is E_0. Then the threshold E of curvature entropy for each tile can be expressed as:

$$E = E_0 + c_1 * r + c_2 * d, \tag{5}$$

where c_1 and c_2 are the adjustable parameters which are assigned by users, r is the distance between one tile on the LOD grid and the visual center **O**, and d is the distance between the viewpoint and the visual center **O**. We can see that E is related to both of distances r and d. It is obvious that the larger the two distances are, the larger E is, and the higher degree of simplification at this position. On the contrary, if the viewpoint is close to the terrain surface, the tiles near the visual center **O** need to be depicted more finely. c_1 and c_2 are used to condition the effect degrees of the distances r and d.

As shown in Fig. 9, if $E \geqslant maxE_i$ is met for the i-th parent node of the Quadtree, its four children nodes will be removed and the node itself is set to a leaf node. We then extract all the leaf nodes of the Quadtree at the new status to generate the corresponding terrain surface. Therefore, not only the distance between the viewpoint and visual center is considered in the browsing process of terrain surfaces, but also the terrain features. Thus, our method can present a more continuous representation for a large-scale terrain surface.

4. Experimental results

In this section, we present experimental results to demonstrate the quality improvement and efficiency of the proposed method. All the experiments were achieved on a desktop computer with Intel Xeon X5650 2.66 GHz and 8 GB RAM. The workflows and algorithms were realized in C++ language and implemented in QuantyView, a software platform for geo-information processing.

To validate the basic features of the method presented in this paper, we first loaded a set of terrain scattered points into the program module we designed. The number of the input terrain points was 2,509,683 and they came from the Loess Plateau area, northwest of China, which covers a rectangle area with length 132 km and width 88 km. We loaded the scattered point clouds to the program module we designed, and used the method described in Section 3.2 to construct the Quadtree-based terrain LODs. Fig. 10 shows the results of the terrain visualization where we set the threshold of local curvature entropy $E_0 = 10$, and set the max depth of this Quadtree $D = 8$. In the result, cracks exist in the adjacent tiles of the initial terrain surface.

On the basis of the initial Quadtree structure, we were able to rapidly eliminate the cracks between the tiles with different levels by updating the Quadtree directly, according to the method presented in Section 3.3. As shown in Fig. 11, a crack-free terrain surface was represented as different rendering formats, in which the altitude of terrain surface was depicted using the rainbow ribbon. Our method output a simplified terrain surface since the terrain features are considered in the whole process of constructing the Quadtree. In the result, there were sparse tiles at the areas with slight terrain changes, while some areas with complex terrain features had relatively intensive tiles. The initial Quadtree-based LOD model had the highest resolution. The follow-up dynamic scheduling and real-time browsing were conducted based on this initial Quadtree.

The threshold E_0 was the most important parameter of constructing and scheduling the Quadtree. As shown in Fig. 12, we increased E_0 gradually to test its performance while the max depth D was fixed. The simplifying degree of the initial terrain surface changed correspondingly as the changes of E_0. Table 1 shows the detailed contrast of the performance our method as the E_0 increasing. From this table, the number of grid cells decreased as the E_0 rose, as well as the algorithm efficiency, the simplifying degree and the rendering frame rate constantly increased. The algorithm efficiency consisted of Quadtree construction and crack elimination. The construction of the Quadtree needed relatively more

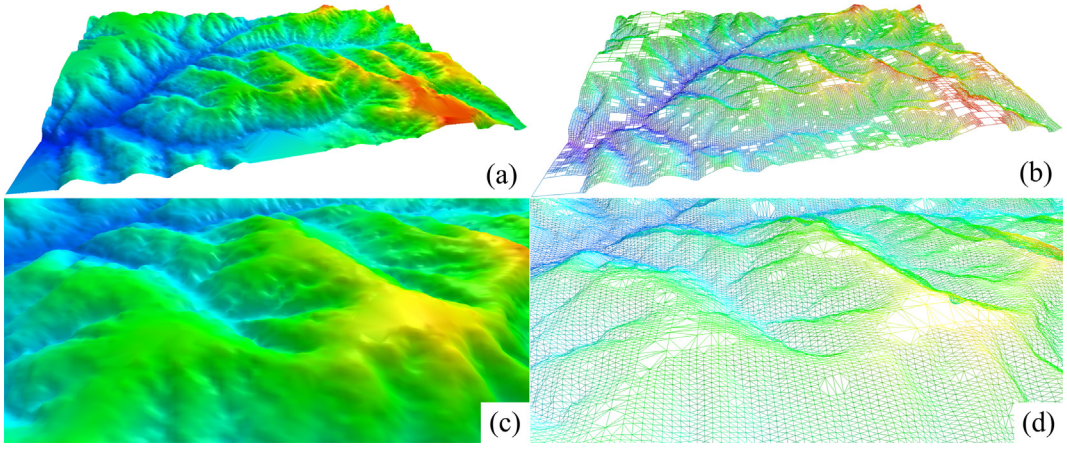

Fig. 11. Terrain surface after eliminating cracks: (a) is the global terrain surface; (b) is the corresponding grid representation; (c) is the terrain surface of a local area; and (d) is the corresponding triangulated representation.

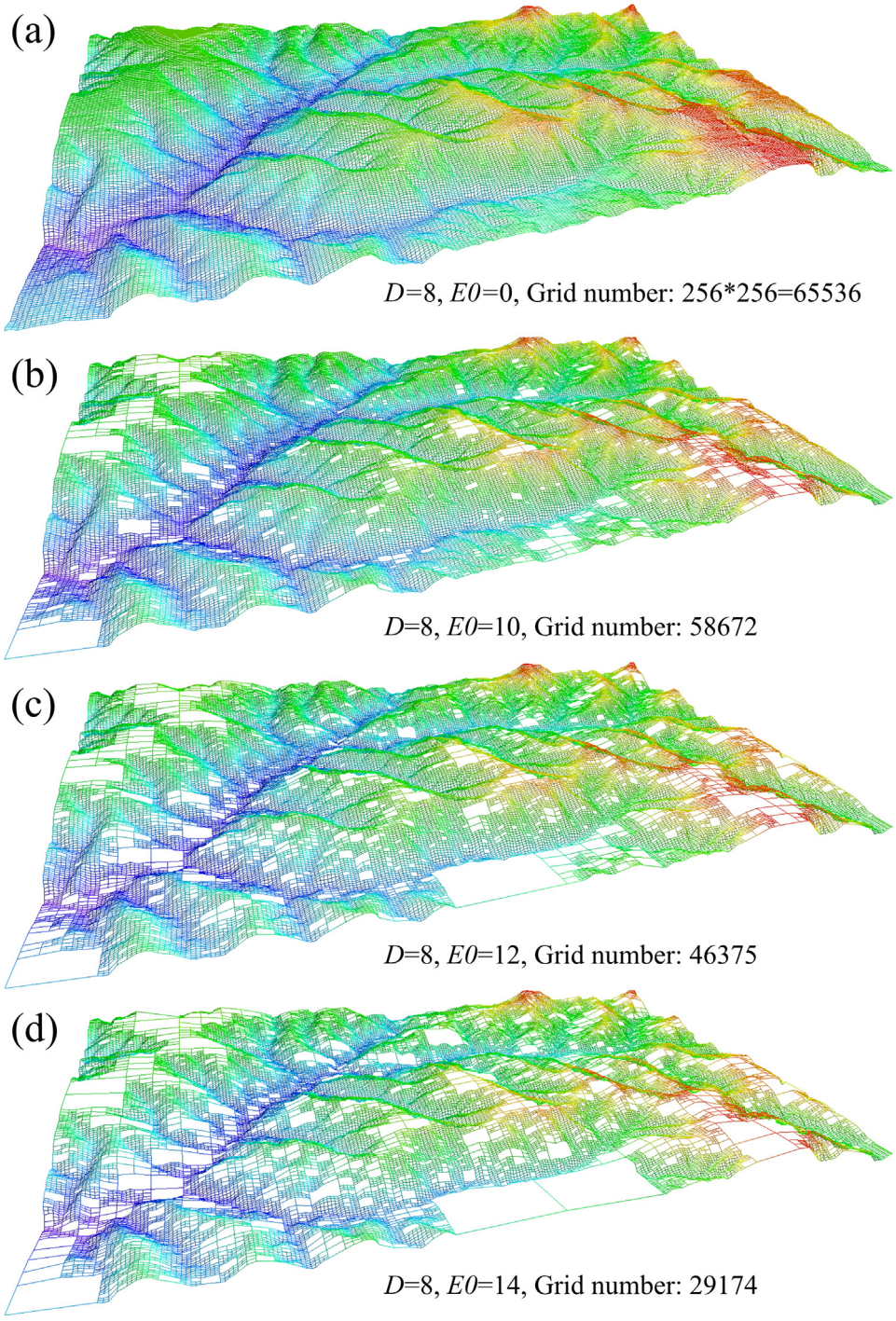

Fig. 12. Terrain surfaces with different simplification degrees.

Table 1
Comparison of the performance with different entropy threshold E_0 (Number of input terrain points n = 2,509,683, and the maximum depth D = 8).

E_0	Tiles	Triangles	Algorithm efficiency		Simplification (%)	FPS
			Quadtree construction (s)	Crack elimination (ms)		
0	65,536	131,072	19.7	86	0	224
10	58,672	122,895	17.1	77	6.24	241
12	46,375	104, 309	13.3	64	20.42	268
14	29,174	71,645	8.6	48	45.34	295
16	11,683	33,009	2.9	29	74.81	319
18	4,596	14,696	0.8	21	88.79	332

Note: FPS, frames per second.

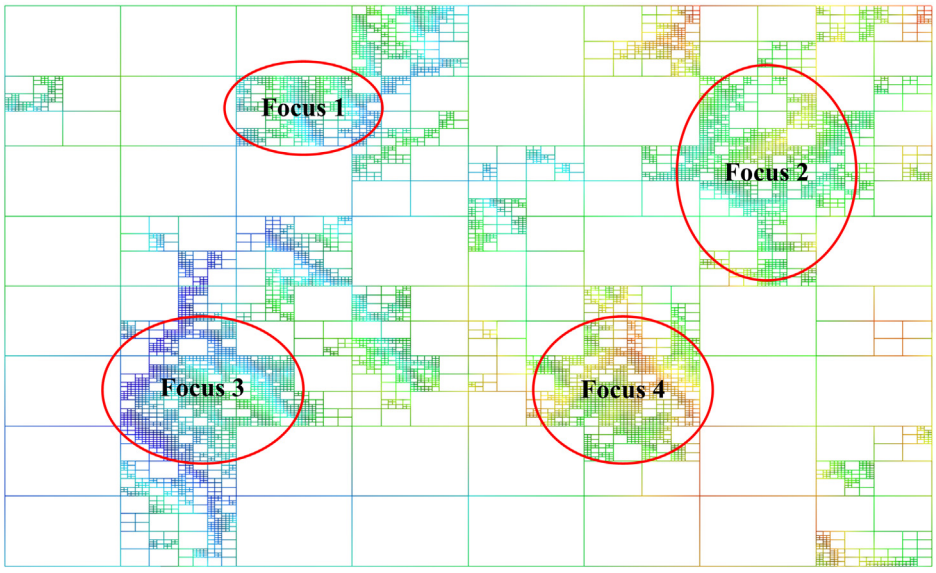

Fig. 13. Static scheduling of the multilevel terrain surface with multiple focuses.

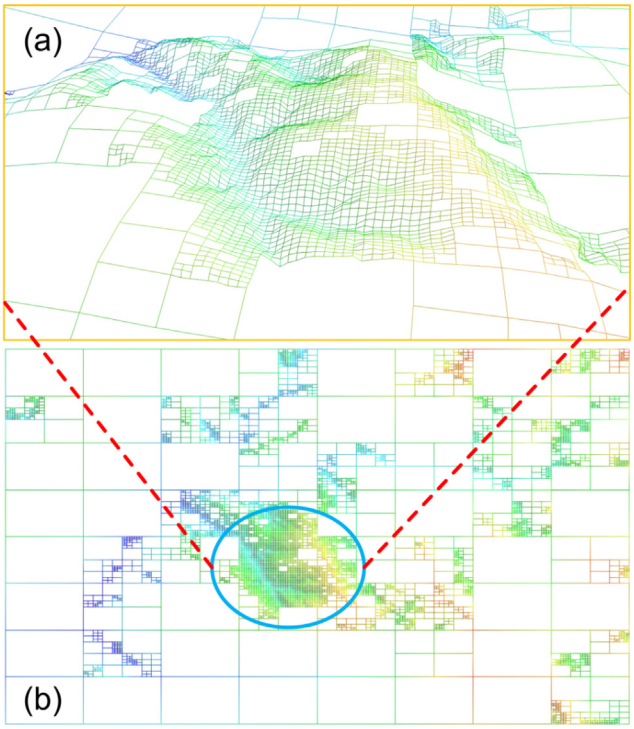

Fig. 14. View-dependent dynamic scheduling: (a) is the terrain grid of a local area; and (b) is the corresponding global status.

time-consuming because our strategy was a comprehensive process where both spatial interpolation and terrain simplification considered geomorphic features were integrated. Moreover, crack elimination was very fast in our method due to the edge-based Quadtree organization and the whole process of eliminating cracks was done by updating the Quadtree straightforwardly. The initial terrain LODs was organized by a simplified Quadtree according to E_0. Following the increase of E_0, the simplification degree and the rendering frame rate increased continuously. The output was same to the result described in Wu et al. (2010) when $E_0 = 0$. In this kind of methods, simplification was not considered in the constructing of Quadtree-based LOD. Therefore, our method integrated LOD segmentation and simplification into the constructing of the Quadtree so that an optimized and less storage-consuming LOD model was obtained. The simplification degree can be controlled flexibly by changing the entropy threshold E_0 according to the needs of users.

On the initial LOD model, static scheduling was very easy to be implemented by setting the focus areas. Then, the usable tiles were selected from the initial Quadtree LOD to reconstruct the terrain surface. However, as the distance to the focuses increased, the simplification degree of the terrain surface became higher, while the basic form of the terrain surface still remained. Fig. 13 shows the reconstruction with four focuses on the basis of the initial surface from Fig. 12b. In the new one, the grid cells in the four focus areas maintained the highest resolution, while other places were simplified according to the distance to the centers of focuses. The total number of grid cells decreased from 58,672 of the initial status to 7369.

View-dependent dynamic scheduling was also easy to be implemented, and it was very fast because the geomorphic features of each tile had been saved in the nodes of the Quadtree. During the roaming process, the program module described in Section 3.4 was triggered. Fig. 14a shows the selected tiles of a local area, and the corresponding global surface is shown in Fig. 14b. The scheduling shown in Fig. 14 was also based on the surface of Fig. 12b. The majority of the grid cells were simplified and the number dwindled to 3543, but the detailed terrain features in the field of view were maintained well.

To verify the performance of the edge-based Quadtree proposed in this paper for eliminating cracks between different LOD tiles, we compared it with two existing organizations of the nodes for a Quadtree: a widely used traditional method (e.g. Lindstrom and Pascucci, 2002; Wu et al., 2010) and an improved strategy presented by Lee et al. (2014). In this test, 10 m contours of the terrains from a south-east coastal area of China (3174 km^2) were used as the input dataset. The threshold E_0 was fixed to 6 and the max depth D was varied from 4 to 11. With the increase of D, the triangles of terrain surfaces were increased from hundreds to hundreds of millions. Fig. 15a and 15b show the terrain surface of this area in different ways when $D = 10$. Fig. 15c and 15d are the

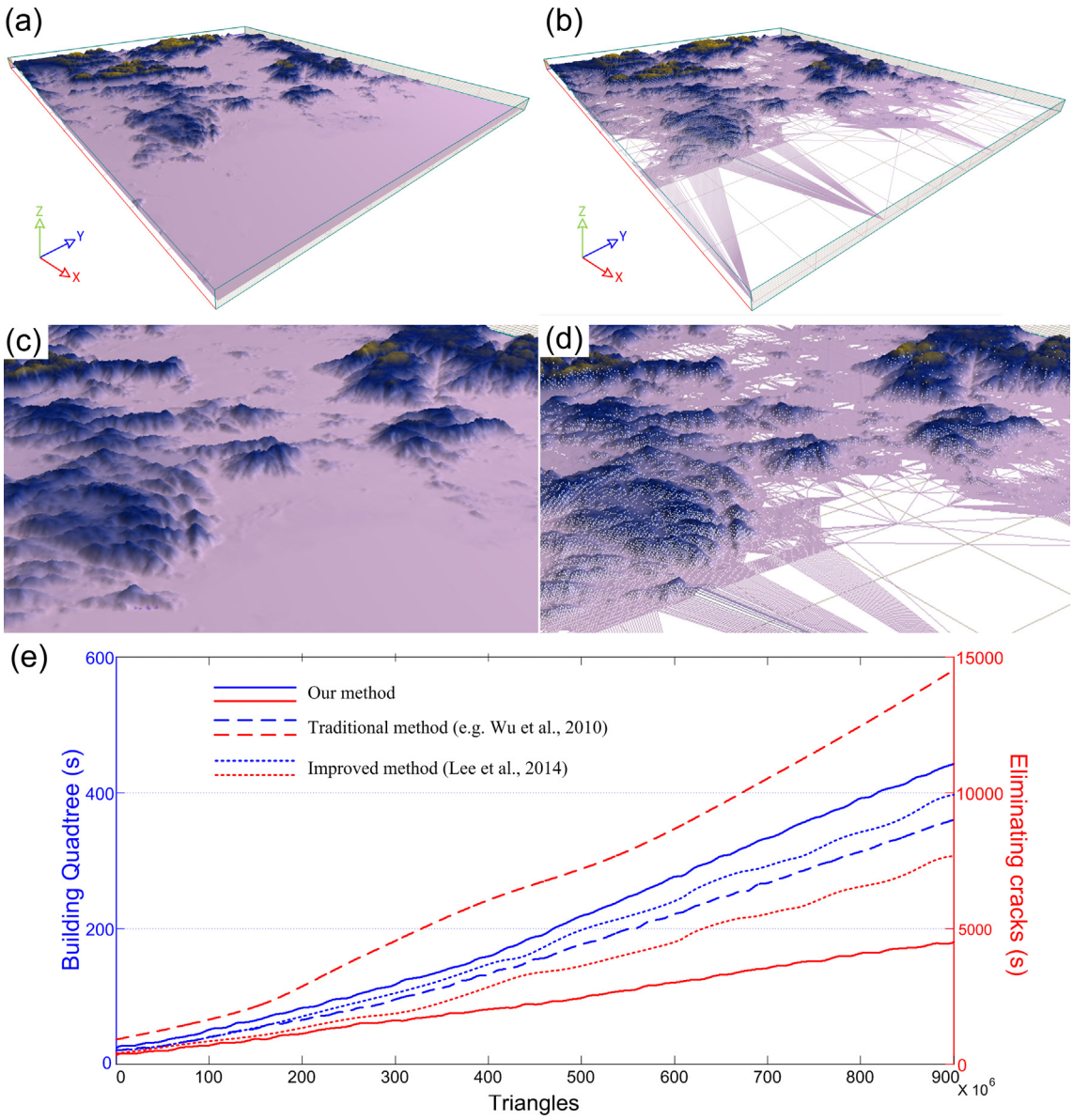

Fig. 15. Comparison of the efficiency for building Quadtree and eliminating cracks. (a) is the terrain surface generated by our method when $D = 10$; (b) is the triangulation representation of this terrain surface; (c) and (d) show a local area of the terrain surface in different ways; (e) shows the curves of the efficiency with the increase of D and triangles.

Table 2
Data description and the corresponding algorithm performance.

Data sources			D	E_0	Tiles	Triangles	Quadtree (s)	Cracks (ms)	Simplification (%)
Location	Size (M)	L × W (km × km)							
Puget Sound	111.6	192 × 180	10	5	887, 095	1,843,397	295.5	1128	12.10
Grand Canyon	228.0	208 × 164	11	2	3,699,376	7,674,913	989.6	4752	8.51

corresponding expression of a local area of the terrain surface. In order to further illustrate the computational efficiency in each stage of the Quadtree construction, the comparisons of the efficiency for building Quadtree and eliminating cracks using three different methods are shown in Fig. 15e. It can be observed that our method costs a little bit more time than the other two methods in the Quadtree construction. However, in the crack elimination our edge-based organization of a Quadtree is much faster than the existing methods (Lee et al., 2014; Wu et al., 2010) with the increase of D and triangles.

In order to further verify the performance of managing and scheduling large-scale terrain data, two bigger terrain datasets, Puget Sound (Washington State, USA) and Grand Canyon (Arizona State, USA), were tested as the data sources. The detailed information of data sources and the performance is shown in Table 2. And Fig. 16 is the visualizing results with different rendering formats at

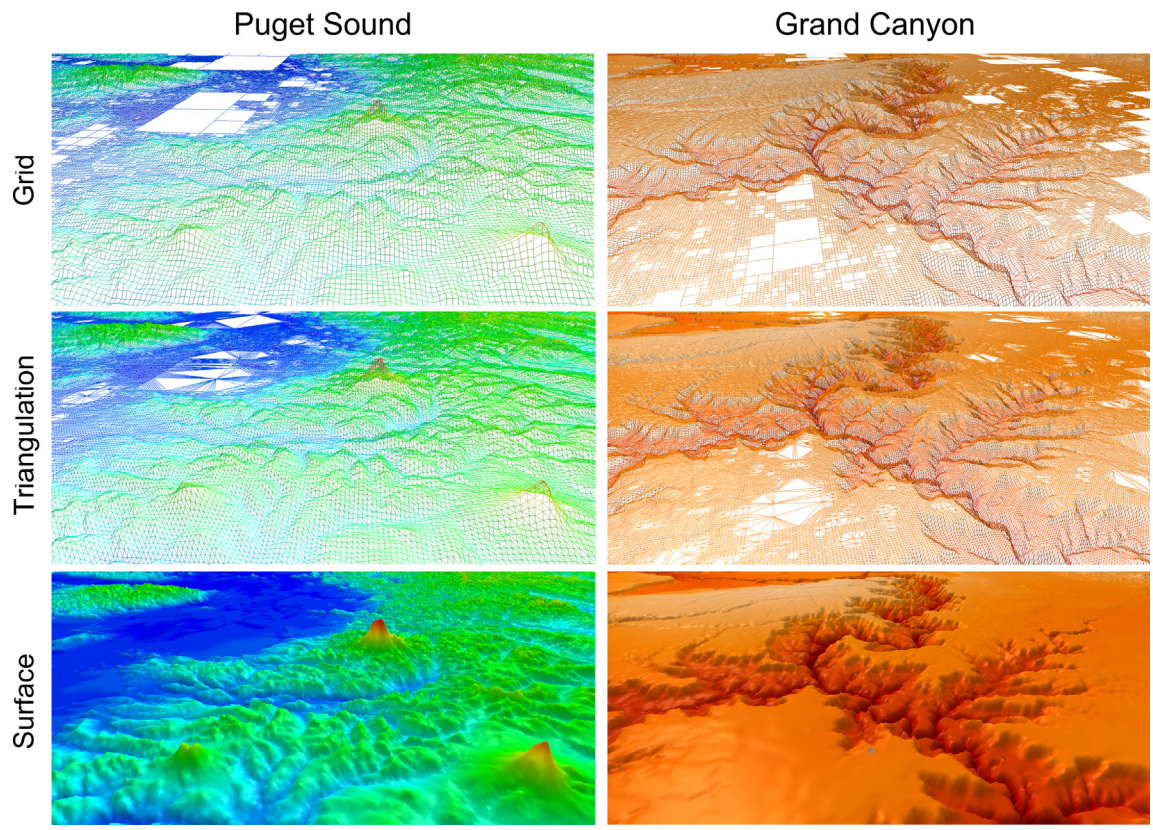

Fig. 16. Screenshots of the two terrain models with different rendering formats in the terrain browsing. In order to insure that there is the same amount of terrain triangles in each browsing process using different methods, roaming paths should be fixed in the tests of the two terrain models. The flight paths for the two roamings are designed in advance with the same flying height, 500 m.

a certain time in the terrain browsing. For the millions triangles, a common workstation is not able to support the real-time rendering and interactive visualization. Therefore, the view-dependent dynamic scheduling was launched to select the LOD tiles distributed in the current visual field. As demonstrated in Fig. 17, the triangles for rendering are reduced greatly, and the frame rates maintain at a high level for the two test data by using our method. Specifically, the two existing methods maintain the similar triangles in the terrain browsing but much more than our method (Fig. 17a). As shown in Fig. 17b, the method proposed by Lee et al. (2014) reduces the time costs due to a bimodal vertex splitting strategy. However, taken together our method still has a better performance for the frames per second in the terrain browsing. The fluctuations of the curves are caused by the different terrain features which are distributed in different areas of the two models.

5. Discussion

Compared with the existing methods (Lee et al., 2014; Wu et al., 2010), our method present a comprehensive Quadtree to organize the LOD tiles where both LOD segmentation and simplification are integrated in the early data processing and all the subsequent operations (e.g. LOD selection, crack elimination, dynamic scheduling, etc.) are implemented on the comprehensive Quadtree (Fig. 1). In general, a full Quadtree was constructed in the initial state of existing methods, and then the subsequent operations were carried out on the full Quadtree. However, our approach will mitigate the subsequent computational costs of dynamic scheduling which need a rigorous real-time requirement. On the other hand, a Quadtree is constructed on the basis of scattered terrain points, and a regular and optimized LOD model is built by using the strategies introduced in this paper. More computational time is consumed in the Quadtree construction since more computational tasks are distributed into this process. The abovementioned tests indicate that the efficiency of dynamic scheduling for large scale terrain models sustains a high level in the method proposed in this paper.

A related advantage of our method is that local curvature entropy is adopted as a measure to divide terrain LOD grid. It will reduce the sensitivity of the existing methods using curvature as a measure (Cerveri et al., 2014; Kobler et al., 2007). This approach may not only ensure the simplification degree, but also can keep the detailed features of terrains as many as possible. Compared with the method by using curvature, our method allows good continuity and stability of the terrain surfaces during the transitions among different LODs. As shown in Fig. 18, curvature and its local entropy are considered as the measurement of the LOD subdivision respectively. It can be seen that the density of the terrain grid cells generated by using the existing methods is extremely uneven, and there are great leaps in the different LODs (Fig. 18c). However, the approach using the local entropy of terrain curvature overcomes this challenge and the generated initial LODs are smoother and more continuous, and no leap-type changes between different LODs (Fig. 18d).

An improved Quadtree structure is exploited in our method where the Quadtree is organized by the edges of LODs. It enhances the efficiency of storing and scheduling terrain data; especially it has a good performance for eliminating cracks in the different adjacent LODs. Compared to two existing Quadtree-based terrain representations (Lee et al., 2014; Wu et al., 2010), cracks are removed quickly and absolutely by straightforward updating the Quadtree, and the accuracy of terrains are maintained as much as possible due to we adjust the tiles with lower resolution to

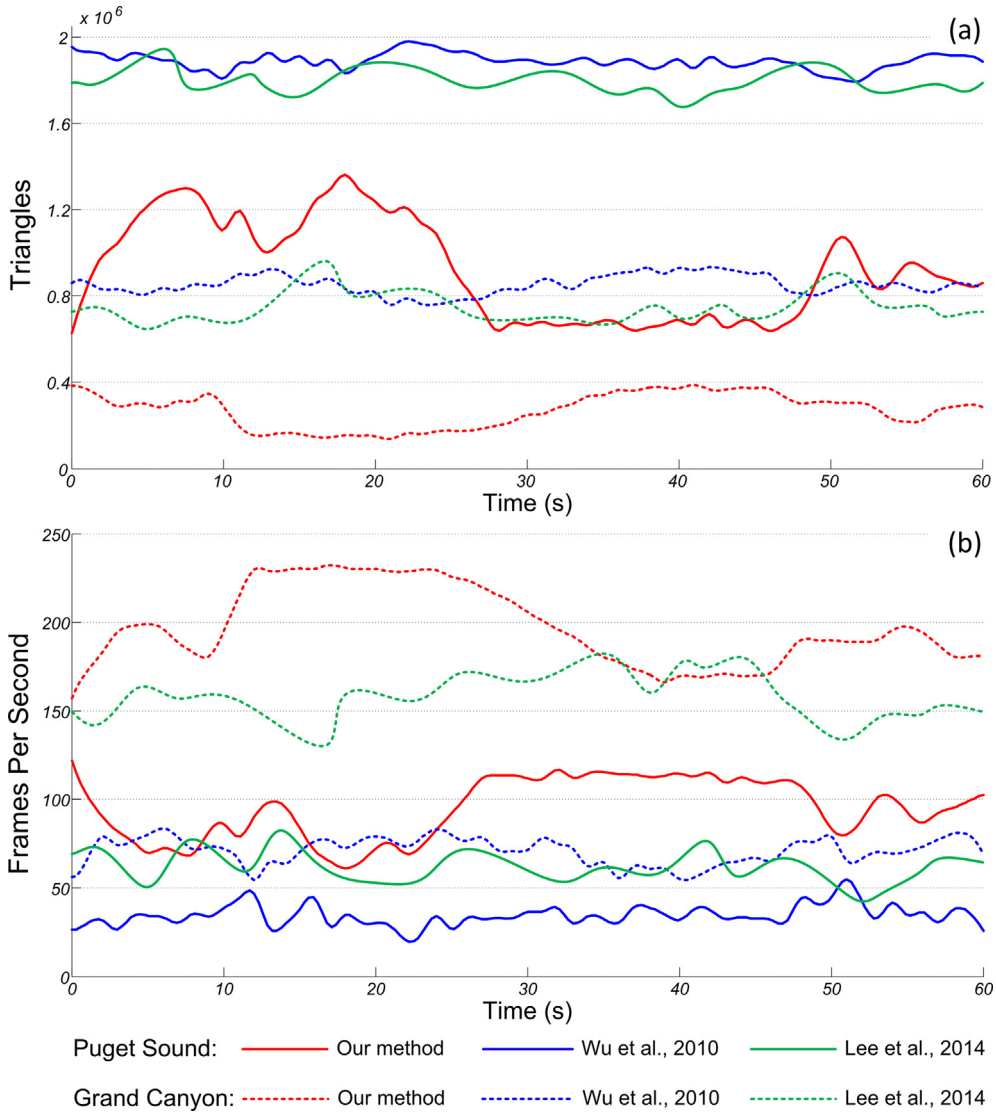

Fig. 17. Comparisons of the performance in the two roamings: (a) triangles; (b) frames per second.

match the higher ones in our work. An improved method proposed by Lee et al. (2014) can reduce the time costs due to a bimodal vertex splitting strategy is employed in the stages of LOD selection and crack removal. However, our method still has a better performance since the processes of vertex splitting are related to the shape of adjacent nodes in the both stages. In the edge-based Quadtree, the four edges of each tile of LODs are stored in the nodes of Quadtree respectively. Thus, we just need to compare the nodes of the Quadtree in order and add the extra points into the corresponding edge arrays. The experimental results indicate that the edge-based Quadtree is able to improve computational performance significantly for the crack elimination since all the steps are done on the Quadtree and no more additional calculations are placed in this process except Quadtree updating. Although the neighbouring edges are respectively stored in different nodes of our Quadtree, every point is only stored once since only indexes of the points are recorded in the nodes of Quadtree to reduce the internal storage costs.

The method presented in this paper is able to meet the requirements of real-time browsing and analysis for large-scale DTMs with millions triangles. In our method, we consider the influence of terrain features in the earlier stage so that an optimized Quadtree is obtained, and we propose an improved strategy of scheduling terrain LODs by embedding variable threshold of curvature entropy in the view-dependent scheduling processes. Therefore, fewer triangles are selected to represent the terrain surface in our method, but the ability of representing terrain features is not reduced. When shifting the viewpoint, we just need to change the entropy threshold E for each node according to the distance between the viewpoint and visual center to update the Quadtree. The new terrain surface can be reconstructed efficiently by extracting the leaf nodes of the Quadtree. Therefore, not only the distance is considered in the browsing process, but also the terrain features. It is implemented straightforwardly due to no additional calculations are utilized to select the usable tiles in each moment. Consequently, a more continuous representation for terrain surfaces is presented in a browsing process by the proposed method and it is competent to support the dynamic scheduling and real-time visualization for a large scale terrain surface.

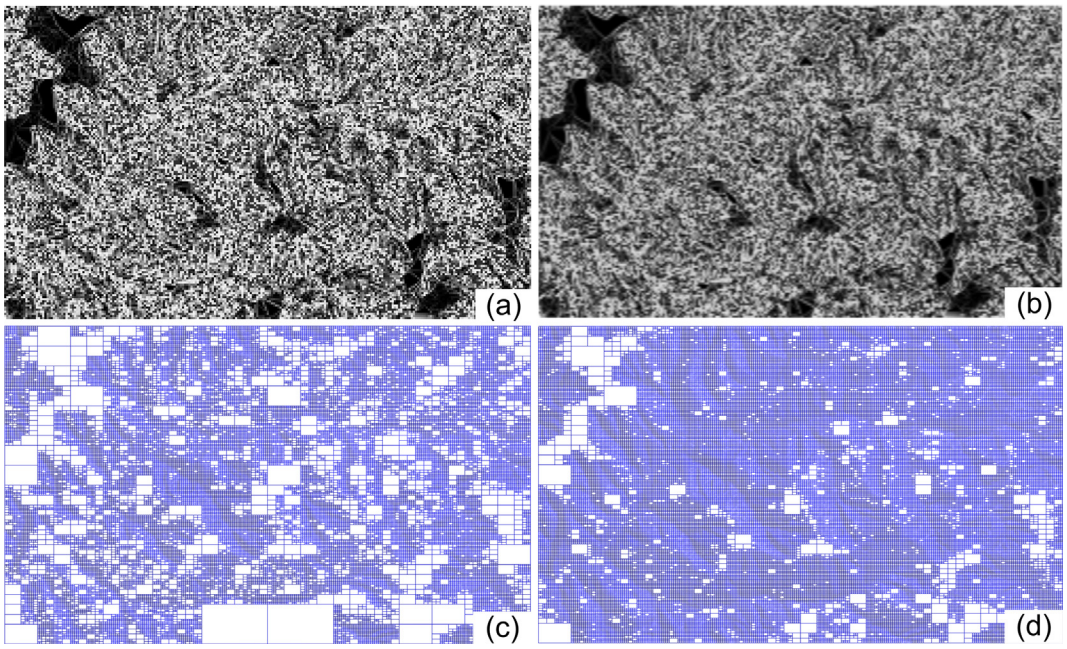

Fig. 18. Contrast of the simplification degree between different measures: (a) and (b) are the intensity maps of the curvature and the corresponding local entropy; (c) and (d) respectively are the density of terrain grid cells for these two measures maintaining the same values of other parameters.

6. Conclusions

This paper presents an improved 3D terrain representation for the dynamic managing and scheduling of massive terrain data using a comprehensive Quadtree, which aims to address an ongoing challenge in the field of terrain modeling and visualization. In this work the reintegration of constructing terrain LODs was implemented on a comprehensive Quadtree, which integrated the more computational tasks into the early data processing and the real-time performance for visualizing large scale terrain surfaces will be guaranteed. Local entropy of terrain curvature was regarded as a measure to generate an optimized LOD model. The elimination of the cracks among in different LODs was a key issue in this work, and was realized rapidly by exploiting a novel Quadtree organization. This edge-based Quadtree can not only produce a more continuous and crack-free hierarchical terrain surface, but also is conducive to implement the view-dependent dynamic scheduling. We have tested our proposal using several terrain datasets with different orders of magnitude. The experimental results validated that our method can be applied to construct LOD 3D terrain models with good performance in terms of computational cost and the maintenance of terrain features. The implementation of the method in a GIS software for practical uses allowed flexible control of parameters, and it could support the real-time dynamic scheduling of large scale terrain models. These experimental results and the actual implementation prove the effectiveness of the method, and also show that the presented method is ready to be reused or adapted in GIS applications for terrain modeling and visualization.

A possible direction for further research could be to construct the global scale terrain models relying on the super calculation platform, and further improve the rendering efficiency by using GPUs (cf. Livny et al., 2009).

Acknowledgments

We are grateful to Professor Lichti and two anonymous reviewers for their insightful comments and suggestions which led to the improvements in the manuscript. This work was supported by the National Natural Science Foundation of China (U1711267, 41172300, 41572314), the National High-tech R&D Program of China (863 Program) (2012AA121401) and the Open Fund of Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences (KLIGIP201504).

References

Bai, R., Li, T., Huang, Y., Li, J., Wang, G., 2015. An efficient and comprehensive method for drainage network extraction from DEM with billions of pixels using a size-balanced binary search tree. Geomorphology 238, 56–67.
Ben-Moshe, B., Mitchell, J.S., Katz, M.J., Nir, Y., 2002. Visibility preserving terrain simplification: an experimental study. In: Proceedings of the Eighteenth Annual Symposium on Computational Geometry. ACM, pp. 303–311.
Bertilsson, E., 2015. Dynamic Creation of Multi-resolution Triangulated Irregular Network. Blekinge Institute of Technology, Karlskrona, Sweden. Thesis (PhD).
Bjorke, J.T., Nilsen, S., 2003. Wavelets applied to simplification of digital terrain models. Int. J. Geograph. Inform. Sci. 17, 601–621.
Bóo, M., Amor, M., 2009. Dynamic hybrid terrain representation based on convexity limits identification. Int. J. Geograph. Inform. Sci. 23 (4), 417–439.
Bösch, J., Goswami, P., Pajarola, R., 2009. RASTeR: simple and efficient terrain rendering on the GPU. In: Eurographics 2009. Munich, Germany, pp. 35–42.
Cerveri, P., Manzotti, A., Confalonieri, N., Baroni, G., 2014. Automating the design of resection guides specific to patient anatomy in knee replacement surgery by enhanced 3D curvature and surface modeling of distal femur shape models. Comput. Med. Imaging Graph. 38 (8), 664–674.
Chen, C., Liu, F., Li, Y., Yan, C., Liu, G., 2016. A robust interpolation method for constructing digital elevation models from remote sensing data. Geomorphology 268, 275–287.
de Floriani, L., Kobbelt, L., Puppo, E., 2004. A survey on data structures for level-of-detail models. In: Dodgson, N., Floater, M., Sabin, M. (Eds.), Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization. Springer, Berlin Heidelberg, pp. 49–74.
Debevec, P., Yu, Y., Borshukov, G., 1998. Efficient view-dependent image-based rendering with projective texture-mapping. In Proceedings of the 9th Eurographics Workshop on Rendering, pp. 105–116.
Duchaineau, M., Wolinsky, M., Sigeti, D.E., Miller, M.C., Aldrich, C., Mineev-Weinstein, M.B., 1997. ROAMing terrain: real-time optimally adapting meshes. In: Proceedings of the 8th Conference on Visualization'97. IEEE Computer Society Press, pp. 81–88.
Dykes, J., MacEachren, A.M., Kraak, M.J., 2005. Exploring Geovisualization. Elsevier.
Evans, W., Kirkpatrick, D., Townsend, G., 2001. Right-triangulated irregular networks. Algorithmica 30 (2), 264–286.
Fan, M., Tang, M., Dong, J., 2004. A review of real-time terrain rendering techniques. In: Proceedings of the 8th international conference on Computer Supported Cooperative Work in Design. IEEE, Piscataway, NJ, pp. 685–691.

Frey, P.J., Borouchaki, H., 2002. Terrain simplification by minimization of the local deformation. Comptes Rendus Mathematique 334, 227–232.
Gao, P., Zhang, H., Li, Z., 2017. A hierarchy-based solution to calculate the configurational entropy of landscape gradients. Landscape Ecol. 32 (6), 1133–1146.
Gerstner, T., 2003. Multiresolution compression and visualization of global topographic data. Geoinformatica 7 (1), 7–32.
He, Z., Kraak, M.-J., Huisman, O., Ma, X., Xiao, J., 2013. Parallel indexing technique for spatio-temporal data. ISPRS J. Photogramm. Remote Sens. 78, 116–128.
Hu, L., He, Z., Liu, J., Zheng, C., 2015. Method for measuring the information content of terrain from Digital Elevation Models. Entropy 17, 7021–7051.
Jenny, B., Jenny, H., Hurni, L., 2011. Terrain generalization with multi-scale pyramids constrained by curvature. Cartogr. Geograph. Inform. Sci. 38 (2), 110–116.
Kalbermatten, M., Van De Ville, D., Turberg, P., Tuia, D., Joost, S., 2012. Multiscale analysis of geomorphological and geological features in high resolution digital elevation models using the wavelet transform. Geomorphology 138 (1), 352–363.
Kamat, V.R., Martinez, J.C., 2005. Large-scale dynamic terrain in three-dimensional construction process visualizations. J. Comput. Civ. Eng. 19 (2), 160–171.
Kang, H., Jang, H., Cho, C.S., Han, J., 2015. Multi-resolution terrain rendering with GPU tessellation. Visual Comput. 31 (4), 455–469.
Kang, L., Xu, J., Yang, C., Yang, B., Wu, L., 2010. An efficient simplification and real-time rendering algorithm for large-scale terrain. Int. J. Comput. Appl. Technol. 38 (1–3), 106–112.
Kobler, A., Pfeifer, N., Ogrinc, P., Todorovski, L., Oštir, K., Džeroski, S., 2007. Repetitive interpolation: a robust algorithm for DTM generation from Aerial Laser Scanner Data in forested terrain. Remote Sens. Environ. 108 (1), 9–23.
Koca, Ç., Güdükbay, U., 2014. A hybrid representation for modeling, interactive editing, and real-time visualization of terrains with volumetric features. Int. J. Geogr. Inform. Sci. 28 (9), 1821–1847.
Lane, S.N., Reaney, S.M., Heathwaite, A.L., 2009. Representation of landscape hydrological connectivity using a topographically driven surface flow index. Water Resour. Res. 45 (8).
Lee, E.S., Lee, J.H., Shin, B.S., 2014. Bimodal vertex splitting: acceleration of quadtree triangulation for terrain rendering. IEICE Trans. Inform. Syst. 97 (6), 1624–1633.
Li, J., Wu, H., Yang, C., Wong, D.W., Xie, J., 2011. Visualizing dynamic geosciences phenomena using an octree-based view-dependent LOD strategy within virtual globes. Comput. Geosci. 37 (9), 1295–1302.
Li, Z., Huang, P., 2002. Quantitative measures for spatial information of maps. Int. J. Geogr. Inform. Sci. 16 (7), 699–709.
Li, Z., Zhu, C., Gold, C., 2004. Digital Terrain Modeling: Principles and Methodology. CRC Press.
Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L.F., Faust, N., Turner, G.A., 1996. Real-time, continuous level of detail rendering of height fields. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. ACM, pp. 109–118.
Lindstrom, P., Pascucci, V., 2002. Terrain simplification simplified: a general framework for view-dependent out-of-core visualization. IEEE Trans. Visual Comput. Graphics 8 (3), 239–254.
Liu, M., Zhu, J., Zhu, Q., Qi, H., Yin, L., Zhang, X., Feng, B., He, H., Yang, W., Chen, L., 2017. Optimization of simulation and visualization analysis of dam-failure flood disaster for diverse computing systems. Int. J. Geogr. Inform. Sci. 31 (9), 1891–1906.
Livny, Y., Kogan, Z., El-Sana, J., 2009. Seamless patches for GPU-based terrain rendering. Visual Comput. 25 (3), 197–208.
Lodha, S.K., Roskin, K.M., Renteria, J.C., 2003. Hierarchical topology-preserving simplification of terrains. Visual Comput. 19 (7), 493–504.
Losasso, F., Hoppe, H., 2004. Geometry clipmaps: terrain rendering using nested regular grids. ACM Trans. Graphics 23 (3), 769–776.
Maguya, A.S., Junttila, V., Kauranne, T., 2013. Adaptive algorithm for large scale dtm interpolation from lidar data for forestry applications in steep forested terrain. ISPRS J. Photogram. Remote Sens. 85, 74–83.
Pajarola, R., 1998. Large scale terrain visualization using the restricted quadtree triangulation. In: Proceedings of the Conference on Visualization'98. IEEE Computer Society Press, pp. 19–26.
Pajarola, R., 2002. Overview of Quadtree-based Terrain Triangulation and Visualization. Department of Information and Computer Science, University of California, Irvine.
Pajarola, R., Gobbetti, E., 2007. Survey of semi-regular multiresolution models for interactive terrain rendering. Visual Comput. 23 (8), 583–605.
Paredes, E.G., Bóo, M., Amor, M., Bruguera, J.D., Döllner, J., 2012. Extended hybrid meshing algorithm for multiresolution terrain models. Int. J. Geogr. Inform. Sci. 26 (5), 771–793.
Plaza, Á., Suárez, J.P., Padrón, M.A., Falcón, S., Amieiro, D., 2004. Mesh quality improvement and other properties in the four-triangles longest-edge partition. Comput. Aided Geom. Des. 21 (4), 353–369.
Qiu, H., Chen, L., Qiu, G., Yang, H., 2013. An effective visualization method for large-scale terrain dataset. WSEAS Trans. Inform. Sci. Appl. 10 (5), 149–158.
Ruzinoor, C.M., Shariff, A.R.M., Pradhan, B., Rodzi Ahmad, M., Rahim, M.S.M., 2012. A review on 3D terrain visualization of GIS data: techniques and software. Geo-spatial Inform. Sci. 15 (2), 105–115.
Schneider, J., Westermann, R., 2006. GPU-friendly high-quality terrain rendering. J. WSCG 14, 49–56.
Shannon, C.E., 1948. A mathematical thoery of communication. Bell Syst. Tech. J. 27 (3), 379–423.
Stock, A., Tolvanen, H., Kalliola, R., 2010. Crossing natural and data set boundaries: coastal terrain modelling in the South-West Finnish Archipelago. Int. J. Geogr. Inform. Sci. 24 (9), 1435–1452.
Suárez, J.P., Plaza, A., 2009. Four-triangles adaptive algorithms for RTIN terrain meshes. Math. Comput. Modell. 49 (5), 1012–1020.
Sukhov, V.I., 1967. Information capacity of a map entropy. Geodesy Aerophotogr. 10 (4), 212–215.
Tan, R., Yao, L., 2007. Optimized triangulation algorithm in terrain modeling. Geo-spatial Inform. Sci. 10 (1), 71–74.
Tarolli, P., 2014. High-resolution topography for understanding Earth surface processes: opportunities and challenges. Geomorphology 216, 295–312.
Tucker, G.E., Lancaster, S.T., Gasparini, N.M., Bras, R.L., Rybarczyk, S.M., 2001. An object-oriented framework for distributed hydrologic and geomorphic modeling using triangulated irregular networks. Comput. Geosci. 27 (8), 959–973.
Ujiie, Y., Kato, T., Sato, K., Matsuoka, Y., 2012. Curvature entropy for curved profile generation. Entropy 14 (3), 533–558.
Vieux, B.E., 1993. DEM aggregation and smoothing effects on surface runoff modeling. J. Comput. Civ. Eng. 7 (3), 310–338.
Wang, G., Gertner, G., Parysow, P., Anderson, A., 2001. Spatial prediction and uncertainty assessment of topographic factor for revised universal soil loss equation using digital elevation models. ISPRS J. Photogramm. Remote Sens. 56 (1), 65–80.
Wilson, J.P., 2012. Digital terrain modeling. Geomorphology 137 (1), 107–121.
Wise, S., 2012. Information entropy as a measure of DEM quality. Comput. Geosci. 48, 102–110.
Wu, J., Amaratunga, K., 2003. Wavelet triangulated irregular networks. Int. J. Geogr. Inform. Sci. 17 (3), 273–289.
Wu, J., Yang, Y., Gong, S.R., Cui, Z., 2010. A new Quadtree-based terrain LOD algorithm. J. Softw. 5 (7), 769–776.
Xiao, X., Weiping, X., Qing, Z., Yeting, Z., Zhiqiang, D., 2013. Integration method of TINs and Grids for multi-resolution surface modeling. Geo-spatial Inform. Sci. 16 (1), 61–68.
Yilmaz, T., Gudukbay, U., Akman, V., 2004. Modeling and visualization of complex geometric environments. In: Norwell, N.A. (Ed.), Geometric Modeling: Techniques, Applications, Systems and Tools. Kluwer, pp. 3–30.
Yu, M., Huang, Y., Xu, Q., Guo, P., Dai, Z., 2016. Application of virtual earth in 3D terrain modeling to visual analysis of large-scale geological disasters in mountainous areas. Environ. Earth Sci. 75 (7), 563.
Zhang, H., Ma, Y., Ma, Z., He, X., Liu, Y., Feng, Z., 2013. Multiscale feature model for terrain data based on adaptive spatial neighborhood. Math. Probl. Eng. 2013, 1–12.
Zheng, X., Xiong, H., Gong, J., Yue, L., 2017. A morphologically preserved multi-resolution TIN surface modeling and visualization method for virtual globes. ISPRS J. Photogramm. Remote Sens. 129, 41–54.