sustainability

MDPI

Article

Prevention of Mountain Disasters and Maintenance of Residential Area through Real-Time Terrain Rendering

Su-Kyung Sung [1], Eun-Seok Lee [2] and Byeong-Seok Shin [1,*]

[1] Department of Computer Engineering, Inha University, Incheon 22212, Korea; rebirth87@naver.com
[2] Department of VR Game and Application, Yuhan University, Sosa-gu, Bucheon 14780, Korea; elflee77@gmail.com
* Correspondence: bsshin@inha.ac.kr

Abstract: Climate change increases the frequency of localized heavy rains and typhoons. As a result, mountain disasters, such as landslides and earthworks, continue to occur, causing damage to roads and residential areas downstream. Moreover, large-scale civil engineering works, including dam construction, cause rapid changes in the terrain, which harm the stability of residential areas. Disasters, such as landslides and earthenware, occur extensively, and there are limitations in the field of investigation; thus, there are many studies being conducted to model terrain geometrically and to observe changes in terrain according to external factors. However, conventional topography methods are expressed in a way that can only be interpreted by people with specialized knowledge. Therefore, there is a lack of consideration for three-dimensional visualization that helps non-experts understand. We need a way to express changes in terrain in real time and to make it intuitive for non-experts to understand. In conventional height-based terrain modeling and simulation, there is a problem in which some of the sampled data are irregularly distorted and do not show the exact terrain shape. The proposed method utilizes a hierarchical vertex cohesion map to correct inaccurately modeled terrain caused by uniform height sampling, and to compensate for geometric errors using Hausdorff distances, while not considering only the elevation difference of the terrain. The mesh reconstruction, which triangulates the three-vertex placed at each location and makes it the smallest unit of 3D model data, can be done at high speed on graphics processing units (GPUs). Our experiments confirm that it is possible to express changes in terrain accurately and quickly compared with existing methods. These functions can improve the sustainability of residential spaces by predicting the damage caused by mountainous disasters or civil engineering works around the city and make it easy for non-experts to understand.

Keywords: mountain disaster protection; real-time terrain modeling and rendering; sustainability of residential area

Citation: Sung, S.-K.; Lee, E.-S.; Shin, B.-S. Prevention of Mountain Disasters and Maintenance of Residential Area through Real-Time Terrain Rendering. *Sustainability* **2021**, *13*, 2950. https://doi.org/10.3390/su13052950

Academic Editor: George D. Bathrellos

Received: 10 January 2021
Accepted: 4 March 2021
Published: 9 March 2021

1. Introduction

Owing to global warming and environmental pollution, phenomena related to climate change, such as landslides, are causing massive damage to residential areas [1]. Countries with mountainous terrain, such as China, suffer a lot of damage from mountain disasters. According to data collected by The Fatal Landslide Event Inventory of China (FLEIC) from 1950 to 2016, there were 1911 landslides and 28,139 deaths in China [2]. The total number of fatal landslides recorded worldwide, excluding those triggered by earthquakes, over the 12 calendar years between 2004 and 2016 (inclusive) was 4862 [3]. The whole world is working to prevent such natural disasters in advance. Identifying changes in terrain is one of the important research topics to prevent disasters, such as landslides, in advance. A number of studies have been proposed to analyze the mechanisms of landslides, such as "stage-wise" methodologies utilizing geo-hydro-mechanical (GHM) [4]. The effects of weathering or sedimentation over a long period of time cannot be ignored, but large-scale civil engineering or dam construction in a particular area can also have a significant impact

on the surrounding terrain in the short term [5]. Previously, experts in the environment and terrain used paper maps, numerical maps, and aerial photographs to estimate changes in the terrain [6]. With the recent prevalence of light direction and ranging (LiDAR), which measures the position coordinates of reflectors by shooting laser pulses and measuring the return time, high resolution terrain data can be obtained [7]. Furthermore, recently, it has been possible to obtain high-resolution data and images at a relatively low price using unmanned aerial vehicles (UAVs) [8] or photogrammetry using drones. UAVs are also utilized for data collection in hazardous areas such as volcanoes [9]. However, the source of such acquired information cannot be visualized and shown intuitively, and it also requires professional knowledge to be interpreted.

In addition, most of these studies only deal with photographs taken periodically or non-periodically to look at changes in the terrain that have occurred over a relatively short period of time, making it difficult to predict possible changes in terrain in the future. Because all terrain is interconnected, local changes in terrain affect a wide range of regions, and 3D-based real-time terrain simulations are required to efficiently represent them [10]. Converting sampled terrain data to three-dimensional geometric information not only makes it easy to calculate variations, but also allows for visualization with high resolution images, and allows non-experts to intuitively observe and interpret terrain changes. In addition to mountain disasters, real-time terrain modeling and rendering technology are preferred over professional data analysis tools because the opinions of experts in urban planning and civil engineering should be well explained to the general public, who do not have expertise in the area.

Three-dimensional real-time rendering technology is being used in various fields beyond simply copying the virtual world in movies and games [11]. One of these fields is simulation; when presenting environmental phenomena or objects using computing tools, state-of-the-art computer graphics techniques are indispensable for more realistic and accurate analysis [12].

A data elevation model (DEM) is used to represent the terrain as an alternative, as scanning terrain and rendering the same data with reality is still impossible even in modern computer graphics devices. A DEM is a set of ground surface height values obtained from uniformly sampled spaces, typically measured using 3D scanning techniques [13]. With higher quality and improved resolution of the measured data, better graphics processing units (GPUs) and memory are required to render terrain based on DEM data in real time. Many algorithms using GPUs are being studied to render large-scale geographic data using a single graphics accelerator [14,15].

Typical methods include the quadtree technique proposed by Lindstrom [16] and the real-time optimally adaptive meshes (ROAM) technique using the binary tree structure proposed by Duchaineau [17]. Lindstrom used a bottom-up approach to determine the segmentation and merging of the quadtree. ROAM uses a binary tree structure that dynamically generates terrain over time. Each patch is a simple isostatic triangle with a structure in which each triangle is recursively divided into two child nodes. This method is called screen error. In this method, based on the error between real terrain and its estimated model, the terrain that is higher than the error set by the user recursively divides. The detailed level of the terrain below the error is adjusted in such a way that no segmentation is made. In the level-of-detail (LOD) methods [18,19], a multi-resolution grid based on the distance between the camera and the object is used in the out-of-core terrain visualization algorithm.

Recent topographic rendering techniques use GPUs to perform mesh reconstruction by swapping limited geometric information such as vertex relocation [20] and persistent grid mapping (PGM) [21]. These techniques reduce errors by randomly swapping a certain number of regular grids. Because the amount of geometry data remains unchanged, these methods have the advantage of constant rendering speed, simpler operation, and being faster than conventional mesh reconstruction methods. However, because mesh cannot be optimized based on error, many geometric errors occur, showing results that are different

from the real terrain. To address these topographical rendering problems, we extract the roughness of the terrain based on a regular lattice map and determine that the part with a specific threshold or higher roughness is a curved terrain. A vertex of the 3D mesh can be gathered in a curved terrain rather than in a fixed interval to represent a more accurate terrain than a regular grid map. We propose a method for efficient LOD by managing *vertex relocated cohesion maps* in a hierarchical structure. This differs from conventional regular grid-based methods [22,23] by measuring geometric errors in Hausdorff distances for the accuracy of twisted terrain. This geometric error value is measured in chunks and the LOD is determined based on this chunk to render the terrain. To improve the efficiency of error computation, we use the Hausdorff distance instead of an existing regular grid that considers only the height differences in the terrain.

For accurate terrain rendering, it is necessary to determine the rendering criteria of the LOD considering geometric errors. The Hausdorff distance is an algorithm for calculating distance, which is a method of calculating the upper and lower bounds for the distances of two sets and any element to which they belong. We apply an approximation of the Hausdorff distance, which holds the upper bound on the distance of the surface normal as a criterion for terrain error by comparing the two vertex groups in which triangulation is generated [24]. This method can be operated faster than other circular mesh-based methods, such as batched dynamic adaptive meshes (BDAM) [25–27], and can be effectively used in map applications with frequent topographical modifications and updates.

To make an accurate comparison of the geometric error, we compare the plane of the existing height-field-based terrain rendering to our proposed method. Based on the results, we experimented with Chunk LOD and the proposed method to render and compare the valleys. By selecting a specific point of representation and measuring geometric errors, our method rendered much more accurate and sophisticated data than the GPU-based Chunk LOD. Our method solves the geometric error problem arising from the existing warping-based vertex cohesion map and expresses a more realistic terrain.

Section 2 discusses existing research techniques used to visualize terrain data, Section 3 details methods for the vertex cohesion map and the Hausdorff distance, Section 4 shows the results compared to the existing Chunk LOD method, and Section 5 concludes.

2. Related Works

A fractal is characterized by its self-similarity, so the shape of the part resembles the whole. Based on this feature, successful results in modeling natural phenomena have been obtained through a midpoint segmentation algorithm using random fractal [28] and a diamond/square algorithm [29], a branch of fractal geometry invented by Mandelbrot in the mid-1970s. Using the characteristics of random fractals, these algorithms are used to model virtual terrain and are very useful for generating various forms of realistic terrain [30], but it is very difficult to predict the final form of terrain created by randomness. Even if we want to model a terrain that is close to reality, it is difficult to predict the shape of the terrain until the result is printed on the screen. We try to solve this problem by proposing a terrain modeling algorithm that limits the Gaussian distribution function utilized in generating the terrain data so that it does not deviate from a predefined outer surface or pyramid structure [31]. However, there was a problem with complex forms of terrain modeling that required too many control points to work unless it was simple terrain modeling.

DEM data obtain high and low values of the surface by photographing the terrain through aerial measurement equipment (LiDAR) or satellites. Unlike other model data storage methods, the DEM data are in an image format. The size of the image becomes the size of the actual measured terrain, and the color value that the pixel has indicates the actual height of the actual terrain. Using image-type storage methods has the advantage of having visual convenience, providing easy management and modification, and modeling realistic forms of terrain. The higher the resolution, the higher the computation for the area within the uniform sample interval, which results in a longer processing time required.

In traditional central processing unit (CPU)-based approaches, the number of triangles was adjusted using hierarchical structures such as triangular binary trees [16] or restricted quadtrees [32], which were efficient in eliminating regions or out-of-sight triangles that did not require segmentation. Triangular binary tree methods can store more geometry data [23] than quadtree methods. The Chunk LOD [33] reduces the computation required to select LODs by using a geometric cache in the form of a specific area geometry chunk.

Since the introduction of programmable GPU pipelines, GPU-based rendering techniques such as geometry image warping [34] have been introduced. The vertex cohesion map simplifies the mesh by extracting key features mainly from the DEM data, using elasticity [35] between the vertices of the terrain mesh. As a result, a GPU can be used to reconstruct and render terrain faster than a CPU. However, owing to the limitations of available memory space, out of core algorithms are widely used [36,37]. An important problem with core extrinsic-based algorithms is to reduce the transmission time to transfer data to the GPU memory. To address this problem, GPU-based geometry decompression techniques [38] using decompressed data in GPUs are proposed, as this is how to use image layers, such as geometry clipmaps [18]. Researchers attempt to speed-up triangular rasterization and improve efficiency with fast and straightforward simplification techniques rather than simplifying the terrain using continuous development of the GPUs [14,39]. These approaches have the advantage of being able to scale out of core and can perform fast rendering by leveraging parallel processing of GPUs, but there are many drawbacks to data considering memory efficiency and terrain roughness, as they typically only perform distance-based LODs.

3. Real-Time Terrain Simulation Using a Vertex Cohesion Map

Terrain modeling and rendering using a vertex cohesion map is faster and more accurate than the existing terrain mesh reconstruction methods. However, it cannot provide LOD generation and selection based on accurate screen error. The proposed method uses the Hausdorff distance [24] to solve this problem. The Hausdorff distance refers to the maximum distance difference between the original data and the modified data. Therefore, when the terrain data of a specific area are transformed into a vertex cohesion map, the maximum geometric error with the original data can be accurately derived. Using the Hausdorff distance as a geometric error matrix, a vertex cohesion map can be applied to the conventional methods that performed the LOD selection using geometric errors.

Figure 1 shows the procedure of the proposed method, which is co-processed in the CPU and the GPU [40,41]. The geometric error of the reconstructed terrain chunk is calculated using the GPU in the preprocessing step. We create a quadtree, a representative data structure used in the terrain LOD, so that the LOD technique can be applied to the area based on the geometric error of the chunk.

3.1. Processing Step of the Vertex Cohesion Map

A simplified regular terrain mesh represents the terrain by the displacement of the vertex from the height field data. The surface roughness is proportional to the frequency of geometric errors. Usually, surface roughness is estimated using a Laplacian operator. However, when using the Laplacian operator, a small bump may have a greater amount of roughness than a mountain.

Figure 2 shows an example of an incorrect estimation of the surface roughness, which is a good example in this case. When estimating the surface roughness at each position of the gray vertex, because θ_a is smaller than θ_b, the terrain in (a) may have greater surface roughness than that in (b). However, the geometric error δ_b is bigger than δ_a. To estimate the roughness more accurately, we apply Gaussian smoothing, which is a well-known method for noise reduction [42]. Gaussian smoothing efficiently solves the Laplacian operator problem for bumps with the distance-based weight. Through the convolution of

the Gaussian smoothing prior to the Laplacian, we can obtain laplacian of gaussian (LoG) as follows.

$$\mathrm{LoG}(x,y) = -1/\pi\sigma^4[1 - (x^2 + y^2/2\sigma^2)e^{-(x^2+y^2)/2\sigma^2} \quad (1)$$

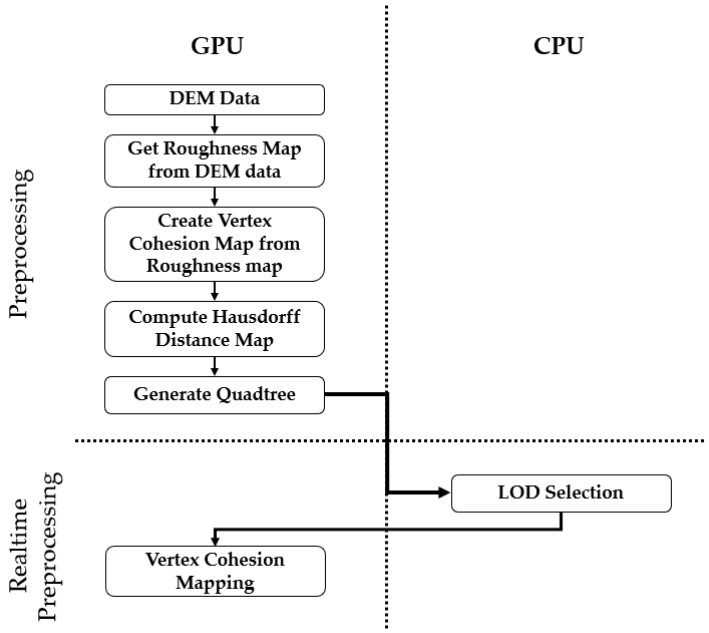


Figure 1. The procedure for vertex cohesion mapping. The level-of-detail (LOD) technique is processed on a central processing unit (CPU) asynchronously while the rendering process is performed on the graphics processing unit (GPU). DEM, data elevation model.

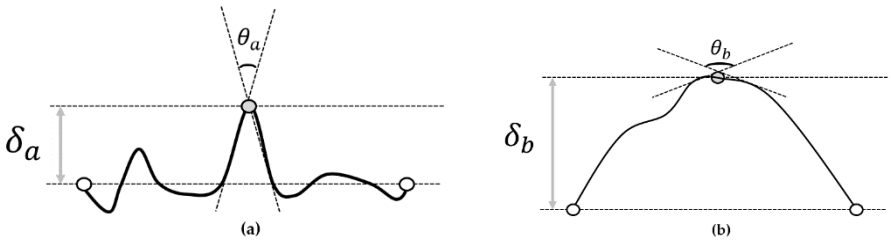


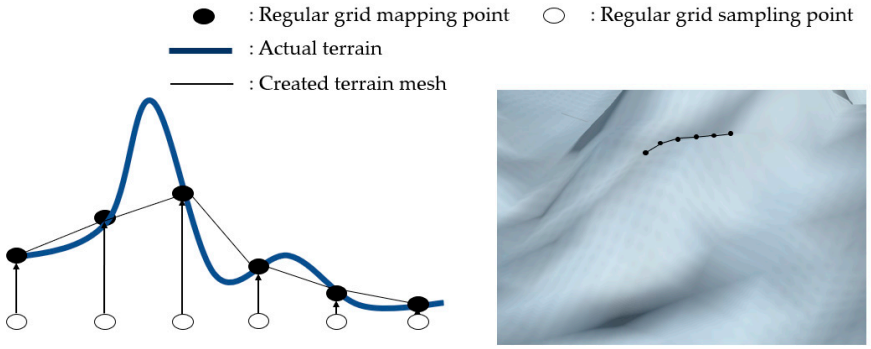
Figure 2. Comparison of flatter surface (**a**) and rougher surface (**b**).

Figure 3 shows how the terrain is reconstructed using a vertex cohesion map. The vertex coherence vector is stored in the vertex coherence map. This vector moves the vertices to locations where a large Hausdorff distance appears. Peaks and valleys will be the representative example. These points can occur through geometric popping (geo-popping).

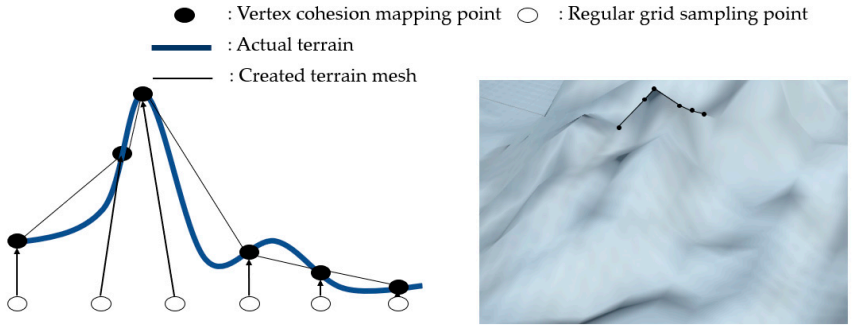
The cohesion vector in the vertex cohesion data is determined as a summation of the cohesion vector, which is related to elastic forces between vertices. The elastic force causes the vertices to push or pull each other when they are closer or farther than their initial distance. The following equation describes Hooke's law of elasticity. This equation is a commonly used approximation for elastic forces:

$$F_e = -k * (l' - l) \quad (2)$$

where l is the initial length of the spring between vertices [43], l' is the length of the spring of a deformed terrain mesh, and k is the modulus of elasticity. Our previous work [18] modified this algorithm using the elastic force between the centroid and the vertex, as shown in Figure 4. Vertices A, B, C, D, and P are relocated vertices of regular grids and T is the centroid of $\square ABCD$.



(a)



(b)

Figure 3. Terrain height sampling using regular girds (**a**) and vertex cohesion mapping (**b**).

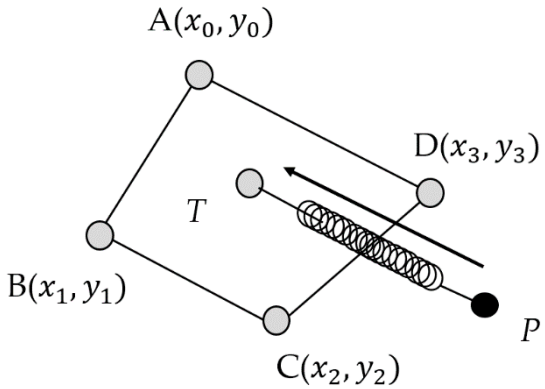


Figure 4. Elastic force using the centroid T. The greater the change in the area of the quadrilateral made with the gray vertices, the harder T will pull vertex P.

We use a centroid for the ideal position that balances the weight of the terrain patch. Finding the centroid T of an N-polygon can be expressed as follows:

$$T_x = \sum^{N-1}(x_i + y_i)\,(x_i y_{i+1} - x_{i+1} y_i)/6S \tag{3}$$

$$T_y = \sum^{N-1}(y_i + x_i)\,(y_i x_{i+1} - y_{i+1} x_i)/6S \tag{4}$$

where S is the area of the N-polygon. We used Heron's formula to compute the area S. We assume that the elastic force pulls a central vertex P toward the centroid of the quadrilateral $\square ABCD$, as shown in Figure 5. Thus, the vector in our method V_e is formulated as

$$V_e = K'P'T \tag{5}$$

where K is the adaptive modus of the elasticity in Equation (5). It is very difficult to determine a reasonable way to fix the value of K. When K is too small, the cohesion vector will be too large and geometrical overlapping or twisting may occur. On the other hand, a large value of K will intensify the elastic force. This might hold vertices and disturb the deformation. Therefore, we define K, which is proportional to the difference in area between the original regular grid and the deformed grid, as below.

$$\text{If } a < \tau_{low} \text{ or } a > \tau_{up}, \text{ then } k = 1, \tag{6}$$

$$\text{else if } \tau_{low} \leq a < 1, \text{ then } k = (1 - a)/(1 - \tau_{low}), \tag{7}$$

$$\text{else if } 1 \leq a \leq \tau_{up}, \text{ then } k = a/\tau_{up} \tag{8}$$

where a is the ratio of regular quads (input data) to $\square ABCD$. In addition, τ_{low} and τ_{up} are user-defined threshold values. When the area of $\square ABCD$ becomes either larger or smaller, the elastic force between T and P becomes stronger. Therefore, a strong elastic force will pull P toward the centroid when the neighboring vertices are too deformed. Otherwise, the elastic force will be weak, and P will be affected more by the cohesion forces.

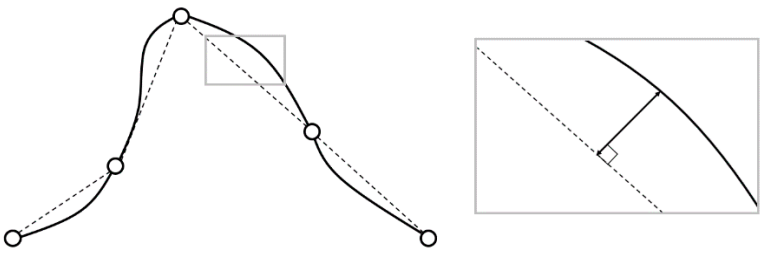


Figure 5. Evaluation of the Hausdorff distance, which measures the maximum distance from the surface normal to the original data.

In order to balance these forces, the proposed method uses a resolution ratio between the vertex cohesion map and the original terrain data. In general, the height field, which is the original data, is measured more accurately as the resolution increases. However, in the case of the vertex cohesion map, the original data are reduced with multiresolution. Therefore, this ratio varies for each detail level. For example, in the case of a *1/4*-sized vertex cohesion map, the ratio is reduced *1/2* times horizontally and vertically. Therefore, when calculating the maximum movement of the vertex, it is appropriate to set τ_{low} to *1/2* and τ_{up} to *2*. In the case of *1/64*, *64* is 2^8, so it is appropriate to set τ_{low} to *1/8* and τ_{up} to *8*. The larger the area is reduced, the larger the movement of the vertex is set. Therefore, the proposed method creates a vertex cohesion map by setting τ differently for each resolution. Therefore, as in the following equation, we finally compute the cohesion vector V through a summation of the attraction and elastic forces as follows:

$$V = V_e + \sum A_{\mathrm{n}} \tag{9}$$

As shown in Figure 5, the proposed method performs the LOD of the terrain models using the Hausdorff distance, which measures the maximum distance from the surface normal to the original data. In general, the terrain can be managed by subdividing the area into a quadtree. In this area, the depth of the quadtree is configured for each detail level of the vertex coherence map. Each node stores how much Hausdorff distance it has at the detail level so that the maximum error value can be calculated in screen space.

3.2. Terrain Reconstruction Using Hierarchical Vertex Cohesion Map

In this section, we explain how to efficiently use the GPU and the CPU at the same time to improve rendering speed. In general, the quadtree reconstructs the terrain by a GPU-based tree traversal algorithm. However, when the rendering process is distributed

by dividing it into several renders passes, there is a task of the CPU unit that creates a GPU command to draw for each chunk and a task of the GPU that executes the command. Each task can be processed simultaneously. In general, in 3D applications, the GPU works asynchronously, so the CPU searches the quadtree. At this time, the appropriate detail level of the vertex cohesion map is selected so that the corresponding data can be rendered on the GPU. The larger the data, the more a bottleneck occurs in the quadtree traversal of the CPU. Therefore, when there are many operations to be processed, a separate GPU pass is designed so that operations being parallelized in the GPU can be performed through the compute shader [44].

The vertex cohesion map does not reduce geometric errors significantly, even at a more detailed level. Sometimes, there is also a case where the geometric error increases. In this case, in the proposed method, the geometric error may be smaller in the case of rendering with a rougher vertex cohesion map than in the case where it is not. Therefore, if you need to use a child node whose geometric error is larger than that of the parent node in the quadtree, the accuracy of the terrain may be degraded. In this case, the vertex cohesion map stored in the parent node is used. This makes it possible to show an optimal rendering speed in an image that guarantees the same error. If the detail level of the adjacent terrain is different, cracks on the terrain appear due to the *T*-vertex. Therefore, it is necessary to know the detail level of the neighbor node. This difference in detail level matches the detail level of the neighboring node in the chunk and prepares a pass for creating a patch so that cracks can be removed.

The CPU always stores information about the commands to be drawn in the quadtree parser. The quadtree parser is a list that stores the nodes of the quadtree where the index for the vertex cohesion map that generated the current command is stored. This list allows you to fill commands into the command buffer immediately when there is no camera movement. In addition, as the LOD selection can be performed directly from the node information selected in the previous frame without having to search the quadtree from the root node, the amount of computation can be greatly reduced. This optimization is very important because the CPU's pre-LOD selection task needs to be processed faster than the rendering LOD.

4. Discussion and Experimental Results

We measured the number of frames per second of the conventional method (the Chunk LOD method) and then compared it to the proposed method to prove the efficiency of the proposed method. For the cohesion vector calculation, it took an average of 0.34 s to generate a terrain with a resolution of 64^2 using the GPU (total eight detail level).

All experiments were performed on a consumer PC equipped with and Intel Core™ i5 3.3 GHz CPU, 16 GB of main memory. The GPU is nVidia™ GeForce GTX 1060 graphic card with 3 GB of local graphic memory. We used the DirectX 11 and the shader model 5.0 as the graphics API. We used the 16-bit Jeju Island, Puget Sound and the Grand Canyon dataset, which are well-known benchmarking data. As the limitation of the texture size of the graphic hardware is 8192 × 8192, we set the resolution of the vertex cohesion map as 64 × 64. The viewport size was set to 1920 × 1080 pixels.

Figure 6 shows a comparison of the rendering speed between the proposed method and the GPU-based Chunk LOD while using the same chunk size. We set the Hausdorff distance to two pixels as the threshold value for selecting the detail level. This is because it takes more than one second to render a frame using the conventional method when the current measurement condition is attached to the ground at a maximum of one pixel or less. When the error tolerance is set to two pixels, it is easy to compare fps, as 5 fps is guaranteed in the conventional method, as shown in Figure 7. Our method showed that the average rendering speed becomes 298.3% of the previous method. The average fastest speed is 358% of the classical method in the Puget Sound, which consists of a lot of flat land, and the slowest average is 224% in the Grand Canyon, which contains several complex valleys. This figure shows that it can render the data three times faster on average than it

takes in real time using a usual amount of computation. Therefore, it is possible to render high-resolution terrain data in real time using existing terrain measurement equipment, which is difficult to render in real time.

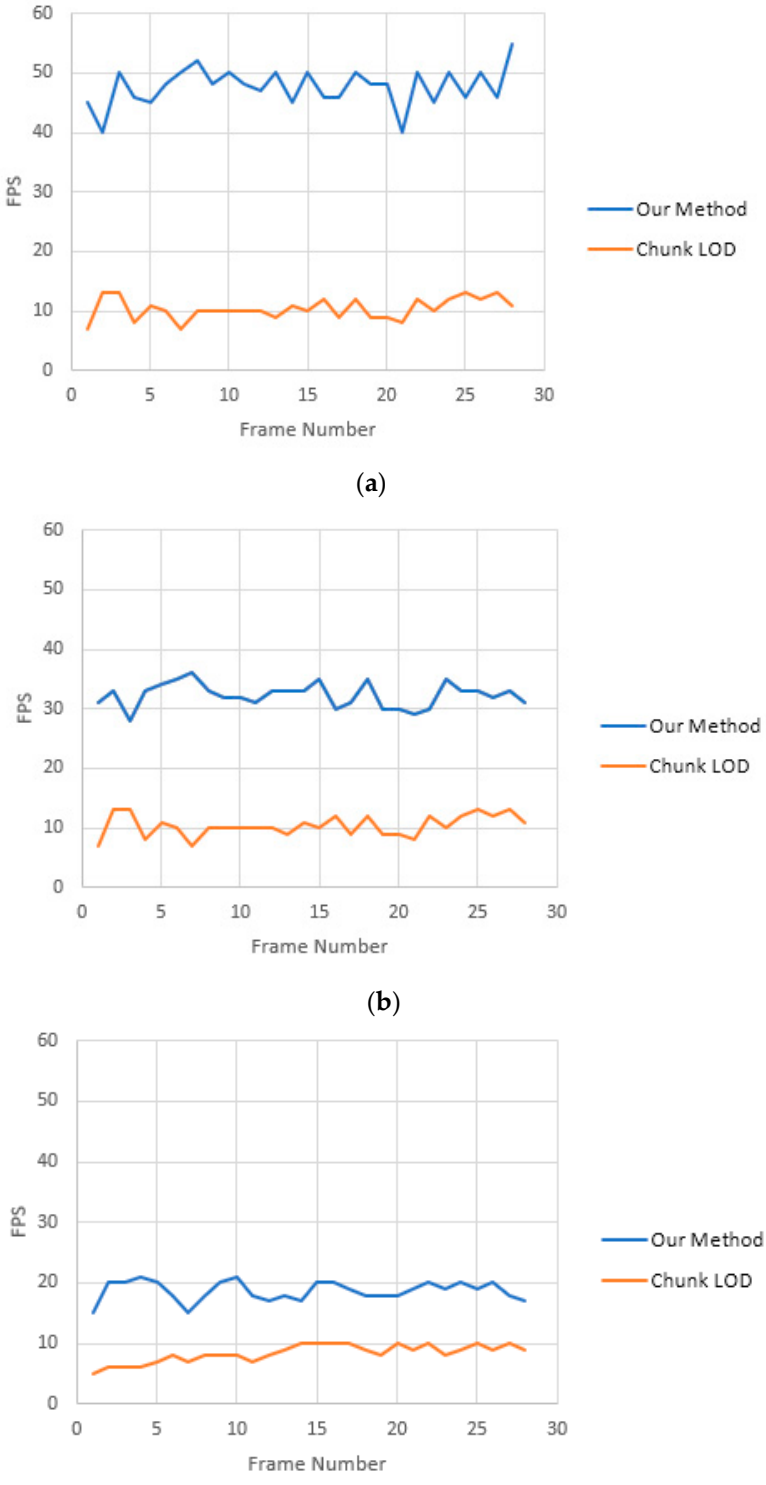


Figure 6. Comparison of the rendering speed between the proposed method and the GPU-based Chunk LOD from the Puget Sound (**a**), Jeju Island (**b**), and Grand Canyon (**c**).

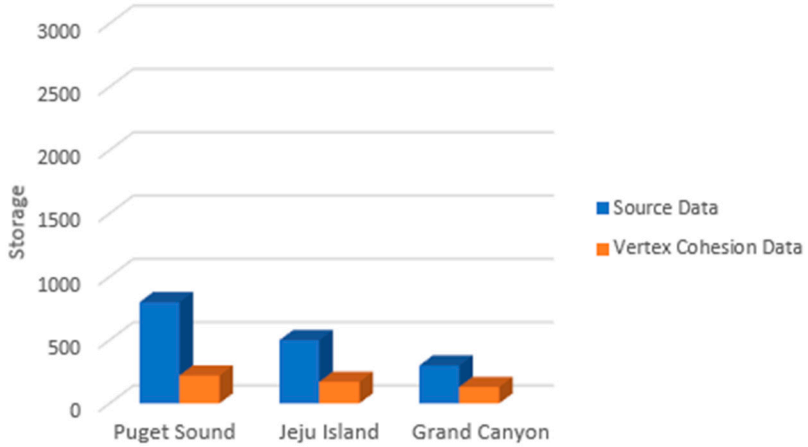


Figure 7. The real-time memory consumption comparison between the proposed method and the conventional method.

Figure 7 is the measurement result of memory consumption. The method proposed in the figure saved memory of up to 79% and an average of 71%. This can effectively reduce the cost of communication with the server. For example, in a work environment with poor network conditions, even if the network is slow, the number of downloads is small, so you can achieve the advantage of speed improvement.

In the proposed method, rendering speed is important. Therefore, by generating high-resolution data and giving distortion, terrain data of 100 m per pixel were created. Based on these data, the accuracy of the vertex cohesion map was measured. Figure 8 shows the main valley data from the Puget Sound. The red dots are the main locations of the selected valleys. Acceleration is meaningful only when there are no errors at these key points.

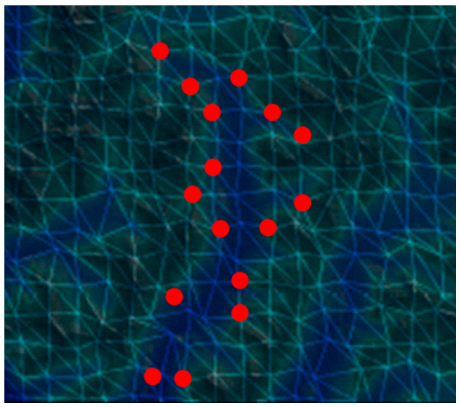


Figure 8. An example of 16 key measuring points with a rapid change in the valley.

In Tables 1 and 2, in order to measure the accuracy of the proposed method, geometric errors were measured by selecting specific points representing valleys on the map. The number of errors (red dots) measured are shown in Figure 8. The pixel error was measured in the final image rendered at a specific camera point. Compared with the experimental result comparison method (the GPU-based Chunk LOD), it shows much more accurate and sophisticated data. This reduces false operations based on simplified data for real-time rendering. Therefore, with the proposed method, the work can be performed more elaborately and quickly in the existing work environment. However, there is the disadvantage that it takes about 0.3 s to produce each chunk. Furthermore, it is efficient because it can extract at about three times the rendering speed and about two times more accurately with a pre-processing process in less than one second.

Table 1. Maximum error and number of errors of the proposed method.

	Max Geometric Error	Number of Errors (Bigger than 100 m)	Number of Errors (Bigger than 50 m)
Puget Sound	136 m (1.36 pixel)	2	4
Jeju Island	143 m (1.43 pixel)	1	3
Grand Canyon	157 m (1.57 pixel)	1	4

Table 2. Maximum error and number of errors of the conventional method.

	Max Geometric Error	Number of Errors (Bigger than 100 m)	Number of Errors (Bigger than 50 m)
Puget Sound	175 m (1.75 pixel)	5	7
Jeju Island	192 m (1.92 pixel)	2	5
Grand Canyon	183 m (1.83 pixel)	3	9

5. Conclusions

This paper presents an efficient GPU-based terrain modeling and rendering technique that can increase the efficiency of the LOD selection and eliminate geometric errors. Existing terrain simulation approaches based on the height-field focus primarily on how large amounts of terrain are being reconstructed at a high speed. By warping a regular grid sample, it can improve performance by reducing errors in rendering compared with conventional mesh reconstruction. However, the resulting warping results in geometric error, resulting in the reconstruction of the mesh with different forms from the real terrain surface. To solve this problem, our method stores how much Hausdorff distance is at the corresponding detail level to allow calculation of the maximum error values in screen space. The Hausdorff distance can solve the geometric error problem arising from the existing warping-based vertex cohesion map. Furthermore, the processing of vertex cohesion maps and the geometry error was pre-processed at the GPU level, which allowed us to identify that the sample topography is faster than the existing height-field based papers. Our method renders a more accurate terrain by solving the geometric error held by the existing regular grid-based terrain rendering. It can be useful in terms of residential maintenance because it is possible to predict the damage caused by mountainous disasters around a city by checking changes in the terrain in real time.

Author Contributions: Conceptualization, S.-K.S. and E.-S.L.; Methodology, S.-K.S.; Supervision, B.-S.S.; Writing—original draft, S.-K.S.; Writing—review & editing, E.-S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (No.NRF-2019R1A2C1090713). This work was also supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No.2020-0-00594, Morphable Haptic Controller for Manipulating VR·AR Contents).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Godt, J.; Baum, R.; Chleborad, A. Rainfall Characteristics for Shallow Landsliding in Seattle, Washington, USA. *Earth Surface Process. Landf.* **2006**, *31*, 97–110. [CrossRef]
2. Lin, Q.; Wang, Y. Spatial and temporal analysis of a fatal landslide inventory in China from 1950 to 2016. *Landslides* **2018**, *15*, 2357–2372. [CrossRef]

3. Froude, M.J.; Petley, D.N. Global fatal landslide occurrence from 2004 to 2016. *Nat. Hazards Earth Syst. Sci.* **2018**, *18*, 2161–2181. [CrossRef]
4. Cotecchia, F.; Santaloia, F.; Tagarelli, V. Towards A Geo-Hydro-Mechanical Characterization of Landslide Classes: Preliminary Results. *Appl. Sci.* **2020**, *10*, 7960. [CrossRef]
5. Kotiukov, P.V.; Lange, I.Y. Engineering Geological Analysis of the Landslide Causes During the Construction of Industrial Building. *Int. J. Civ. Eng. Technol.* **2019**, *10*, 1471–1478.
6. Hapke, C.; Richmond, B. Monitoring Beach Morphology Changes Using Small-Format Aerial Photography and Digital Softcopy Photogrammetry. *Environ. Geosci.* **2000**, *7*, 32–37. [CrossRef]
7. Kang, D.-S.; Lee, E.-S.; Shin, B.-S. Real-Time Terrain Rendering Framework for GIS Applications. *J. Korea Spat. Inf. Syst. Soc.* **2009**, *11*, 73–78.
8. Niethammer, U.; James, M.R.; Rothmund, S.; Travelletti, J.; Joswig, M. UAV-Based Remote Sensing of the Super-Sauze Landslide: Evaluation and Results. *Eng. Geol.* **2012**, *128*, 2–11. [CrossRef]
9. Granados-Bolaños, S.; Quesada-Román, A.; Alvarado, G.E. Low-Cost UAV Applications in Dynamic Tropical Volcanic Landforms. *J. Volcanol. Geotherm. Res.* **2021**, *410*, 107143. [CrossRef]
10. Koller, D.; Lindstrom, P.; Ribarsky, W.; Hodges, L.F.; Faust, N.; Turner, G. Virtual GIS: A Real-Time 3D Geographic Information System. In Proceedings of the Proceedings Visualization'95, Atlanta, GA, USA, 29 October 1995; pp. 94–100.
11. Akeley, K.; Jermoluk, T. High-Performance Polygon Rendering. In Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, Atlanta, GA, USA, 1–5 August 1988; Association for Computing Machinery: New York, NY, USA, 1 June 1988; pp. 239–246.
12. Jepson, W.; Liggett, R.; Friedman, S. An Environment for Real-Time Urban Simulation. In Proceedings of the 1995 Symposium on Interactive 3D Graphics, Monterey, CA, USA, 9–12 April 1995; Association for Computing Machinery: New York, NY, USA, 15 April 1995.
13. Khatamian, A.; Arabnia, H.R. Survey on 3D Surface Reconstruction. *J. Inf. Process. Syst.* **2016**, *12*, 338–357.
14. Hong, M.P.; Oh, K. Motion-Blurred Shadows Utilizing a Depth-Time Ranges Shadow Map. *J. Inf. Process. Syst.* **2018**, *14*, 877–891.
15. Cozzi, P.; Stoner, F. GPU Ray Casting of Virtual Globes. In Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, Los Angeles, CA, USA, 26–30 July 2010; ACM: New York, NY, USA, 2010; p. 128.
16. Lindstrom, P.; Koller, D.; Ribarsky, W.; Hodges, L.F.; Faust, N.; Turner, G.A. Real-time, Continuous Level of Detail Rendering of Height Fields. In Proceedings of the Art and Interdisciplinary Programs of SIGGRAPH'96, New Orleans, LA, USA, 4–9 August 1996; pp. 109–118.
17. Duchaineau, M.; Wolinsky, M.; Sigeti, D.E.; Miller, M.C.; Aldrich, C.; Mineev-Weinstein, M.B. ROAMing terrain: Real-time optimally adapting meshes. In Proceedings of the 8th Conference on Visualization'97, Phoenix, AZ, USA, 24 October 1997; pp. 81–88.
18. Asirvatham, A.; Hoppe, H. Terrain rendering using GPU-based geometry clipmaps. In *GPU Gems 2*; Addison-Wesley: Boston, MA, USA, 2005; Volume 2, pp. 27–46.
19. Bhattacharjee, S.; Narayanan, P.J. Hexagonal geometry clipmaps for spherical terrain rendering. In Proceedings of the 1st ACM SIGGRAPH Conference and Exhibition in Asia, Siggraph, Singapore, 10–13 December 2008; pp. 10–13.
20. Lee, E.-S.; Lee, J.-H.; Shin, B.-S. Vertex relocation: A feature-preserved terrain rendering method for pervasive computing environments. *Multimed. Tools Appl.* **2016**, *75*, 14057–14073. [CrossRef]
21. Livny, Y.; Sokolovsky, N.; Grinshpoun, T.; El-Sana, J. A GPU Persistent Grid Mapping for Terrain Rendering. *Vis. Comput.* **2008**, *24*, 139–153. [CrossRef]
22. Lee, E.-S.; Lee, J.-H.; Shin, B.-S. Bimodal Vertex Splitting: Acceleration of Quadtree Triangulation for Terrain Rendering. *IEICE Trans. Inf. Syst.* **2014**, *E97-D*, 1624–1633. [CrossRef]
23. Pomeranz, A.A. ROAM Using Surface Triangle Clusters (RUSTiC). Master's Thesis, University of California, Davis, CA, USA, January 2000.
24. Aspert, N.; Santa-Cruz, D.; Ebrahimi, T. MESH: Measuring Errors between Surfaces Using the Hausdorff Distance. In Proceedings of the IEEE International Conference on Multimedia and Expo, Lausanne, Switzerland, 26–29 August 2002; Volume 1, pp. 705–708.
25. Cignoni, P.; Ganovelli, F.; Gobbetti, E.; Marton, F.; Ponchio, F.; Scopigno, R. BDAM—Batched Dynamic Adaptive Meshes for High Performance Terrain Visualization. *Comput. Graph. Forum* **2003**, *22*, 505–514. [CrossRef]
26. Gobbetti, E.; Marton, F.; Cignoni, P.; Di Benedetto, M.; Ganovelli, F. C-BDAM–compressed batched dynamic adaptive meshes for terrain rendering. *Comput. Graph. Forum* **2006**, *25*, 333–342. [CrossRef]
27. Cignoni, P.; Ganovelli, F.; Gobbetti, E.; Marton, F.; Ponchio, F.; Scopigno, R. Planet-Sized Batched Dynamic Adaptive Meshes (P-BDAM). In Proceedings of the IEEE Visualization, Washington, DC, USA, 22–24 October 2003; p. 20.
28. Mandelbrot, B. *The Fractal Geometry of Nature*; WH Freeman and Co.: New York, NY, USA, 1982.
29. Miller, G.S.P. The Definition and Rendering of Terrain Maps. In Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, Dallas, Texas, USA, 18–22 August 1986; Association for Computing Machinery: New York, NY, USA, 31 August 1986; pp. 39–48.
30. Stachniak, S.; Stuerzlinger, W. An algorithm for automated fractal terrain deformation. *Comput. Graph. Artif. Intell.* **2005**, *1*, 64–76.

31. Koh, E.; Hearn, D. *Generating and Structuring Terrain Features on Parallel Vector Architectures*; Department of Computer Science, University of Illinois at Urbana-Champaign: Urbana, IL, USA, 1988.
32. Pajarola, R. Large Scale Terrain Visualization Using the Restricted Quadtree Triangulation. In Proceedings of the Visualization '98, Research Triangle Park, NC, USA, 18–23 October 1998; IEEE Computer Society Press: Los Alamitos, CA, USA, 1998; pp. 19–26.
33. Ulrich, T. Rendering Massive Terrains Using Chunked Level of Detail Control. In Proceedings of the ACM SIGGRAPH, San Antonio, TX, USA, 21–26 July 2002.
34. Shi, J.Y.; Taifi, M.; Khreishah, A.; Wu, J. Sustainable GPU Computing at Scale. In Proceedings of the 2011 14th IEEE International Conference on Computational Science and Engineering, Dalian, China, 24–26 August 2011; pp. 263–272.
35. Akbarov, S.D.; Mehdiyev, M.A. The interface stress field in the elastic system consisting of the hollow cylinder and surrounding elastic medium under 3D non-axisymmetric forced vibration. *Comput. Mater. Contin.* **2018**, *54*, 61–81.
36. Isenburg, M.; Gumhold, S. Out-of-core Compression for Gigantic Polygon Meshes. In *ACM SIGGRAPH 2003 Papers*; ACM: New York, NY, USA, July 2003; pp. 935–942.
37. Ripolles, O.; Chover, M.; Gumbau, J.; Ramos, F.; Puig-Centelles, A. Rendering continuous level-of-detail meshes by Masking Strips. Graphical Models. *Graph. Model.* **2009**, *71*, 184–195. [CrossRef]
38. Lindstrom, P.; Cohen, J.D. On-the-fly Decompression and Rendering of Multiresolution Terrain. In Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Washington, DC, USA, 19–21 February 2010; pp. 65–73.
39. Hong, M.; Oh, K. Efficient motion blurred shadows using a temporal shadow map. *Hum. Cent. Comput. Inf. Sci.* **2017**, *7*, 1–13. [CrossRef]
40. Saravanan, V.; Pralhaddas, K.D.; Kothari, D.P. An optimizing pipeline stall reduction algorithm for power and performance on multi-core CPUs. *Hum. Cent. Comput. Inf. Sci.* **2015**, *5*, 1–13. [CrossRef]
41. Song, W.; Zou, S.; Tian, Y.; Sun, S.; Fong, S.; Cho, K.; Qiu, L. A CPU-GPU Hybrid System of Environment Perception and 3D Terrain Reconstruction for Unmanned Ground Vehicle. *J. Inform. Process. Syst.* **2018**, *14*, 1445–1456.
42. Gunn, S.R. Edge detection error in the discrete laplacian of gaussian. *ICIP* **1998**, *2*, 515–519.
43. Hong, M.; Jeon, J.-H.; Yum, H.-S. Plausible mass-spring system using parallel computing on mobile devices. *Hum. Cent. Comput. Inf. Sci.* **2016**, *6*, 1–11. [CrossRef]
44. Nie, D.-H.; Han, K.-P.; Lee, H.-S. GPU-based Stereo Matching Algorithm with the Strategy of Population-based Incremental Learning. *J. Inform. Process. Syst.* **2009**, *5*, 105–116. [CrossRef]