A New Method for Quadtree Triangulation

REFIK SAMET, EMRAH OZSAVAS
Department of Computer Engineering, Ankara University, 06100, Ord.Prof.Dr.Sevket Kansu Binasi, Tandogan, Ankara, TURKEY

Abstract: The aim of the study is to increase the accuracy of a terrain triangulation while maintaining or reducing the number of triangles. To this end, a new non-trivial algorithm for quadtree triangulation is proposed. The proposed algorithm includes: i) a resolution parameters calculation technique and ii) three error metric calculation techniques. Simulation software is also devised to apply the proposed algorithm. For this purpose a data file is processed via the software. Initially, a data file is read to obtain the elevation data of a terrain. After that, 3D mesh is generated by using the original algorithm and the proposed algorithm. For each of the algorithms, two states are analyzed: i) the state with fixed resolution parameters and ii) the state with dynamically changing resolution parameters. For all of the cases, terrain accuracy value and number of triangles of 3D meshes are calculated and evaluated. Finally, it is shown that dynamically changing resolution parameters improve the algorithms' performance.

Keywords: 3D Mesh, Quadtree Triangulation, Error Metric, Terrain Accuracy, Number of Triangles

1 Introduction

Interactive visualization of very large scale terrain data imposes several efficiency problems. To best exploit the rendering performance, the scene complexity must be reduced as much as possible without leading to an inferior visual representation. Therefore, the geometric simplification must be controlled by an approximation error threshold. Additionally, different parts of the visible terrain can be rendered at different *Level-of-Detail* (LOD) to increase rendering performance [1].

Multiresolution terrain modeling is an efficient approach to improve the speed of 3D terrain modeling [2]. The concept of multiresolution refers to the possibility of using different representations of a spatial entity, having different levels of terrain accuracy and complexity [3]. The existing algorithms and models for constructing multiresolution terrain models can be divided into two categories: 1) Grid-based algorithms and 2) Triangulated Irregular Network (TIN)-based algorithms.

Grid-based algorithms include the quadtree triangulation algorithm [4], [5] and the triangle bisect algorithm [2]. The quadtree triangulation algorithm constructs multiresolution models based on a bottom-up approach which is easy to implement [4]. The triangle bisect algorithm constructs multiresolution models based on a top-down approach [2]. Many algorithms have been developed based on the triangle bisect algorithm, such as the adaptive quadtree [6], the ROAMing algorithm [7], Right TIN model [8], the longest edge bisection algorithm [9] and dynamic adaptive meshes [10]. The most common drawback of the grid-based algorithms is that the polygonalization is seldom optimal, or even near optimal. Large, flat surfaces may require the same polygon density as do small, rough areas [6].

TIN-based algorithms are inefficient because generation of even modest size TINs requires extensive computational effort. TINs are non-uniform in nature, and consequently surface following (e.g. for the animation of objects on the surface) and intersection (e.g. for collision detection, selection, and queries) are hard to handle efficiently due to the lack of a spatial organization of the mesh polygons [6]. TIN-based algorithms can produce near optimal results in the number of triangles needed to satisfy a particular error threshold, but most do not operate in real-time [11].

Quadtree based multiresolution triangulations have been shown to be exceptionally efficient for grid digital terrain data [12]. The purpose of this study is to increase the terrain accuracy while maintaining or reducing the number of triangles in order to reduce the graphics load, using quadtree triangulation. We propose a dynamic calculation technique for the resolution parameters and three new error metric calculation techniques for the subdivision criteria of the quadtree triangulation algorithm. For each of the four techniques (the original algorithm's technique and three techniques proposed in this paper), two states are analyzed: i)

the state with fixed resolution parameters and ii) the state with dynamically changing resolution parameters. For all of the cases, the terrain accuracy and number of triangles values of 3D meshes are calculated and evaluated. For this purpose, the quadtree triangulation algorithm is used on data files which include elevation data of a terrain.

The rest of the paper is organized as follows: In Section 2, previous related works in the field of quadtree triangulation algorithm are discussed and the original algorithm is described. Section 3 describes a resolution parameter calculation technique and three new error metric calculation techniques which are proposed in this paper. In Section 4, the simulation procedure of the proposed algorithm is described. Section 5 presents implementation results and an in-depth analysis of the terrain accuracy and number of triangles values of the generated meshes. Finally, in Section 6, conclusions for this study are presented.

2 Quadtree Triangulation Algorithms

In grid-based algorithms, a terrain is also called a height field because it consists of an *NxM* field of height values. Height fields are usually stored as *NxM* gray scale images. The color of each pixel of the images represents the height value (0-255) for the corresponding location in the terrain. These height fields can be generated automatically or they can come in the form of Digital Elevation Maps (DEMs) that describe actual regions of the Earth's surface [13].

Terrain accuracy, process time, memory requirement, support for real-time processing, and number of triangles of the generated mesh are the parameters to evaluate the techniques used in 3D mesh generation step [13]. From a GIS point of view the evaluation should be based on accuracy of the approximated terrain and number of triangles required to draw the terrain.

Terrain accuracy is a measure of how close the approximated terrain resembles the original height field. This measure is calculated using the vertical distance between corresponding points in the rendered terrain and the height field. Once the vertical elevation difference has been computed for each point in the height field, the actual terrain accuracy as a percent is calculated as follows [13]:

$$accuracy = 100 * ((totalHeight - totalDelta) / totalHeight), \quad (1)$$

where the *totalHeight* is the sum of the heights of all the points in the height field and the *totalDelta* is the sum of the vertical differences of the points.

The number of triangles in the 3D model is easily counted.

The quadtree triangulation algorithms have problems with terrain accuracy and the triangulation of the 3D mesh is never optimal. Large, flat surfaces may require the same polygon density as do small, rough areas. This is due to the sensitivity to localized, high frequency data within large, uniform resolution areas of lower complexity [6]. To increase the terrain accuracy, much more triangles must be drawn in 3D mesh and this causes extra graphics load that is undesirable for real-time applications. For example, terrain data of an area of $50km^2$ with 5-meter ground elevation sampling resolution is a grid in size of *10000x10000*. After triangulation the total number of triangles is about 200 million, what will cause problems in real-time applications.

There are two construction approaches to generate 3D mesh from a set of points on a regular grid. One approach is performed *bottom-up* and the other *top-down* to generate the hierarchy [12].

2.1 Bottom-Up Triangulation Approach

In the *bottom-up* construction approach, the input grid is partitioned into atomic nodes of *3x3* elevation points. These nodes form the leaf nodes [12] of a complete and balanced quadtree over the entire input grid.

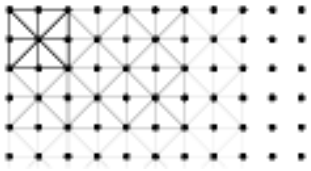

Fig.1. Bottom-Up triangulation

The main phase of this approach then consists of coalescing all mergible nodes bottom-up to create the quadtree.

2.2 Top-Down Triangulation Approach

The second approach is a *top-down* construction of the hierarchy [12]. This approach starts by representing the entire data set simplified by one root node and splits nodes recursively as necessary to approximate the data set. Nodes are never merged.

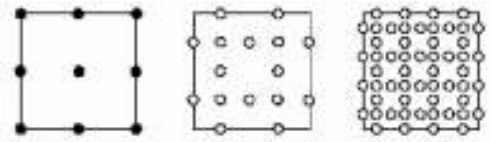

Fig.2. Top-Down triangulation

The structure used here is a tree where each node has either zero or four children; nodes with zero children are the leaf nodes. Each height level of the tree represents a greater level of detail of the terrain [14]. Creating the structure involves descending the tree and, at each node, establishing if the node is at the correct level of detail or if it should be subdivided into four children, in which case each child is then processed recursively [15].

2.3 Split Metric Calculation

The important part is deciding what the correct level of detail should be at each node. The basic idea is to render sections of the terrain at a higher level of detail when they are close to the viewpoint and at lower levels of detail when they are farther away from the viewpoint. In addition to this, the error of the terrain should be taken into account to ensure that flat areas use fewer triangles since less detail is required and bumpier areas use more triangles to show more detail. Once error values are stored, deciding on the correct level of detail for each node is left up to a *split metric* calculation. The following equation is used for the split metric variable *f* [4]:

$$f = L/(d * C * \max(c * d2,1)), \quad (2)$$

where *L* is the distance from the current node to the viewpoint, *C* is the minimum global resolution, *c* is the desired global resolution, *d* is the width of the node and *d2* is the error metric for the node. Here, *C* and *c* are user-configurable parameters and *L* and *d* are calculated in terrain vertices. If the condition *(f<1)* is satisfied, that node is subdivided into four children nodes. Calculation of the error metric *d2* is described below.

2.4 Calculation of the Error Metric *d2* of the Original Algorithm (Technique 1)

To calculate the error metric at a given node, the original algorithm [4], [12], [14] determines the elevation differences (error values) between the actual terrain height and the displayed terrain height at the centers of each of the four edges and the two diagonals of the node [4]. Let us demonstrate the calculation steps at a given node (Fig.3) by using the original algorithm. Here, *V(i)*, *i=0,1,...,8*, is a vertex of a node. *E(i)* denotes the error and *H(i)* denotes the height of the vertex *V(i)*.

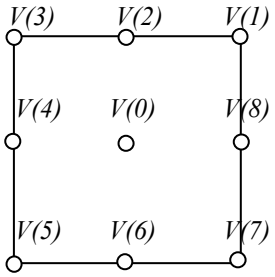

Fig.3. Vertices of a quadtree node

Step 1. Determine the maximum of error values at the two diagonals.

$$E(i) = \max(\, abs((H(i+1) + H(i+5))/2 - H(i)), \quad (3)$$
$$abs((H(i+3) + H(i+7))/2 - H(i))), i = 0,$$

Step 2. Determine the error values at the centers of each of the four edges.

$$E(i) = abs((H(i-1) + H(i+1))/2 - H(i), i = 2,4,6,8, \quad (4)$$

Step 3. The error of the node is the maximum of the five error values.

$$d2 = \max(E(i)), i = 0,2,4,6,8, \quad (5)$$

The algorithm explained above is a fast and moderately simple way to display height fields with continuous levels of detail. However, this algorithm suffers from problems of terrain accuracy and total number of triangles. The number of triangles is increased by increasing the terrain accuracy. The increase in number of triangles causes extra graphics load. It may be virtually impossible to create 3D models in real-time for increasingly large terrains. However, reduction in number of triangles causes a loss in terrain accuracy.

3 The Proposed Quadtree Triangulation Algorithm

The work presented here is based on the bottom-up quadtree triangulation approach. The proposed algorithm includes a technique to calculate the resolution parameters and 3 different techniques to calculate the *d2* value for each node in the height field.

3.1 Resolution Parameters Calculation Technique

Before the calculation of the resolution parameters for each node, the number of height intervals and the start and the end height values of each height interval should be determined, as shown in Table 3.

Step 1. Calculate the average height of 9 vertices of a node (Fig.3).

$$avg = \sum_{i=0}^{8} H(i)/9, \quad (6)$$

The calculated *avg* value is the height of that node.

Step 2. Find the height interval number that has a smaller start height than the node's height and greater end height than the node's height. The found height interval number is used as *c* value for error metric *(d2)* calculation for that node.

Step 3. Find the level of the node. The level of a node is calculated by taking the logarithm of the width of the node. The *C* parameter for that node is then equal to the level value and calculated as follows:

$$C = \log_2(d) \quad (7)$$

3.2 Calculation of Error Metric *d2* of the First Proposed Technique (Technique 2)

Step 1. Calculate the *avg* value (Eq. 6) of a node (Fig.3).

Step 2. Calculate the differences between these 9 vertices and average of them.

$$E(i) = abs(H(i) - avg), i = 0,1,...,8, \quad (8)$$

Step 3. Find the maximum difference value, it is the error metric for that node.

$$d2 = \max(E(i)), i = 0,1,...,8, \quad (9)$$

3.3 Calculation of Error Metric *d2* of the Second Proposed Technique (Technique 3)

Step 1. Find the maximum of 9 vertices for a node (Fig.3).

$$\max = \max(H(i)), i = 0,1,...,8, \quad (10)$$

Step 2. Find the minimum of 9 vertices for a node.

$$\min = \min(H(i)), i = 0,1,...,8, \quad (11)$$

Step 3. Calculate the difference between the maximum and minimum values, it is the error metric for that node.

$$d2 = \max - \min, \quad (12)$$

3.4 Calculation of Error Metric *d2* of the Third Proposed Technique (Technique 4)

Step 1. Find the median value of 9 vertices for a node (Fig.3). Median is the middle value of the given numbers or distribution in their ascending order. If the size of the numbers or distribution is even, median is the average value of the two middle elements [16].

$$med = median(H(i)), i = 0,1,...,8, \quad (13)$$

Step 2. Calculate the differences between the median value and 9 vertices.

$$E(i) = abs(H(i) - med), i = 0,1,...,8 \quad (14)$$

Step 3. Find the maximum of the difference values, it is the error metric for that node.

$$d2 = \max(E(i)), i = 0,1,...,8 \quad (15)$$

4 Simulation Procedure

We have developed a simulator in C to generate the mesh and evaluate the terrain accuracy and number of triangles values of each of the error metric calculation techniques. The OpenGL functions [17] are used to draw the rendered terrain. The simulation procedure is realized into three steps:

4.1 Step 1

In the initial step of the simulation, the elevation data of a terrain are read from a raw file into a matrix of size *(terrain width)x(terrain height)*. This matrix is called the height field matrix.

4.2 Step 2

The quadtree data structure is created to represent terrains of size $(2^{N+1})x(2^{N+1})$. In this tree structure

every node has either four or zero children nodes. Nodes with zero children are the leaf nodes.

In order to represent the quadtree structure, a numeric matrix of size *(terrain width)x(terrain height)* is used. Each node in the tree corresponds to one value in the quadtree matrix, which is the subdivision metric. Calculating quadtree matrix values involves recursively descending the tree and, at each node, establishing if the node is at the correct level of detail or if it should be subdivided into four children, in which case each child is then processed by the recursive algorithm.

After determining the height intervals, 3D meshes are generated by using the original algorithm and the proposed algorithm with fixed and dynamically changing resolution parameters.

4.3 Step 3

Total triangle number and terrain accuracy values are calculated for all of the 3D meshes. These values are automatically saved to a text file.

While calculating the terrain accuracy, for vertices that are corners of triangles the vertical difference and thus delta value is equal to zero. To calculate delta values for other vertices in the rendered terrain, a bounding rectangle of all triangles is determined in 2D and the plane equation of all triangles is calculated [18]. The plane equation is used to check if the vertex is inside the triangle or not, for all vertices in the bounding rectangle. If a vertex is inside a triangle then its delta value is calculated.

5 Implementation Results and Evaluation

We used two raw files 'test1.raw' and 'test2.raw' as data files to the simulator.

Table 1. File properties and parameters

File Name	File Size	Viewpoint ***x*** Location	Viewpoint ***y*** Location	Viewpoint ***z*** Location
test1.raw	513*513 (N=9)	200	200	170
test2.raw	2049*2049 (N=11)	200	200	170

5.1 Implementation 1: The Situation with Fixed Resolution Parameters

Initially, the original algorithm and the three error metric calculation techniques of the proposed algorithm are applied to all of the nodes with fixed parameter values. In this implementation, the resolution parameters are taken from the user and are not changed during the triangulation process. To be able to compare the results of this implementation with the results of Implementation 2, the numbers of triangles values should be equal or nearly equal. Thus the terrain accuracy values can be compared and evaluated. In order to have much more opportunity for evaluation, we have tested different values of resolution parameters. The results for two data files are shown in Table 2.

Table 2. Results of fixed resolution parameters for test1.raw and test2.raw

File	Technique	***C*** Value	***c*** Value	Number of Triangles	Terrain Accuracy
test1.raw	1	1	20	152840	99,233%
	1	1	13	63777	98,757%
	1	2	4	25055	98,023%
	2	3	9	71540	98,978%
	2	3	5	24010	98,131%
	2	2	5	10305	97,288%
	3	4	5	163588	99,329%
	3	3	5	96799	99,095%
	3	2	4	27388	98,239%
	4	5	5	106307	99,132%
	4	3	5	37765	98,486%
	4	3	3	13464	97,540%
test2.raw	1	2	6	922893	98,716%
	1	2	4	492739	97,600%
	1	2	2	140934	93,785%
	2	2	8	490543	97,598%
	2	2	5	211478	95,245%
	2	3	3	175479	94,568%
	3	2	7	1208153	99,021%
	3	2	4	496183	97,605%
	3	2	3	294356	96,304%
	4	2	8	517045	97,640%
	4	2	5	222984	95,338%
	4	2	3	89489	91,628%

5.2 Implementation 2: The Situation with Height Intervals

In this implementation, we used height intervals in order to determine the desired global resolution parameter *c* as described in Section 3.1. We tried different numbers of height intervals, but in this paper we present the results of the trials with 2, 4 and 8 height intervals separately in order to shorten the presentation. Table 3 shows the height intervals and corresponding *c* values.

Table 3. Height intervals and corresponding *c* values

Number of Height Intervals	Interval Start Height-End Height	Corresponding ***c*** Value
2	0-127	2
	128-255	1

4	0-63	4
	64-127	3
	128-191	2
	192-255	1
8	0-31	8
	32-63	7
	64-95	6
	96-127	5
	128-159	4
	160-191	3
	192-223	2
	224-255	1

In the triangulation process, for each node of the height field, the interval number that has a smaller start height than the node's height and greater end height than the node's height is found. The found height interval number is used as *c* value for error metric *(d2)* calculation for that node. As shown in Table 3, the corresponding *c* values and the start-end height values of height intervals are in reverse order. The reason for using such a reverse order is to have greater *c* values, smaller *f* values and many more triangles in the low parts of the terrain. The *C* parameter is calculated for every node separately during the triangulation process as described in Section 3.1.

The results of the original algorithm and the proposed algorithm for two data files are shown in Table 4.

Table 4. Results of dynamically changing resolution parameters for test1.raw and test2.raw

File		test1.raw		test2.raw	
Number of Height Intervals	Technique	Number of Triangles	Terrain Accuracy	Number of Triangles	Terrain Accuracy
2	1	24521	98,198%	80500	93,082%
	2	9515	97,314%	192731	96,277%
	3	27977	98,366%	192044	96,271%
	4	12983	97,672%	83182	93,180%
4	1	62025	98,954%	464957	98,200%
	2	27993	98,366%	191167	96,269%
	3	70938	99,060%	468127	98,206%
	4	38285	98,620%	203338	96,330%
8	1	168003	99,388%	1005405	99,064%
	2	67189	99,058%	464555	98,199%
	3	183876	99,440%	1005405	99,064%
	4	102060	99,209%	483785	98,237%

5.3 Evaluations

The results show that dynamically changing parameters make the four techniques give greater terrain accuracy values with decreasing or nearly same number of triangles. For example, Technique 1 gives 98.023% terrain accuracy with 25055 triangles by using fixed resolution parameters (*C*=2, *c*=4) as shown in Table 2 and the same technique gives greater (98.198%) terrain accuracy with decreasing number of triangles (24521) by using two height intervals (Table 4) for test1.raw. This result is achieved also for the different values of viewpoint locations, *C* and *c* parameters. We propose three different techniques in order to show the effect of using dynamically changing resolution parameters much more powerful. According to need and technical supports, a user may choose one of the techniques to apply. The figures below show the comparisons of the results of the four techniques with dynamically changing resolution parameters (the situation with height intervals) against fixed resolution parameters (the situation without height intervals) for two data files. Each figure has two curved lines: the dashed ones show the results of the situations with 2, 4 and 8 height intervals and so have three points. The other lines show the results of Implementation 1 and have three points, each point shows the results of a fixed resolution parameters group. Each figure shows the results of one of the four techniques for one of the files and so there are eight figures.

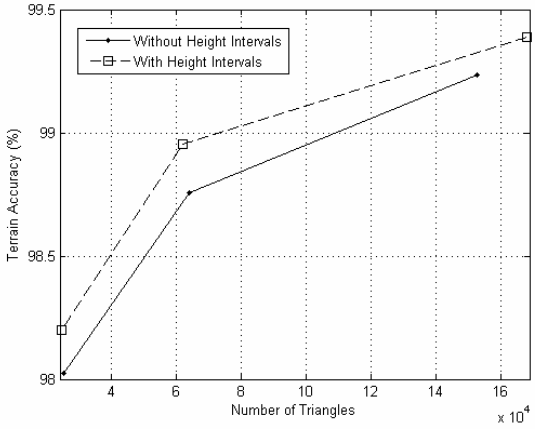

Fig.4. Results of the Technique 1 for test1.raw

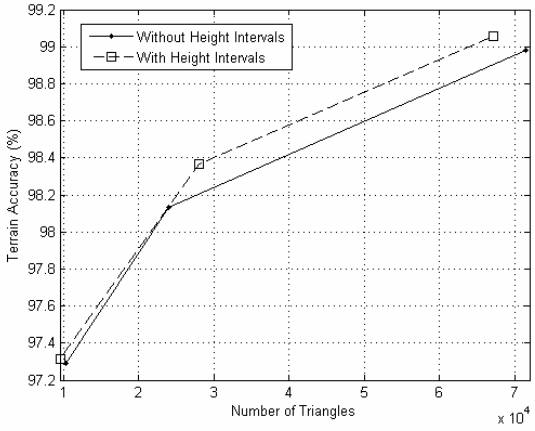

Fig.5. Results of the Technique 2 for test1.raw

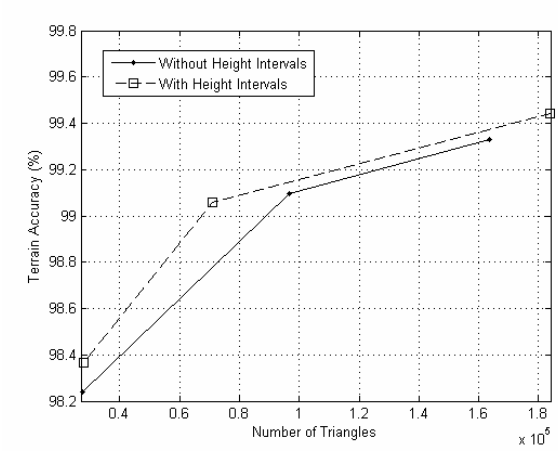

Fig.6. Results of the Technique 3 for test1.raw

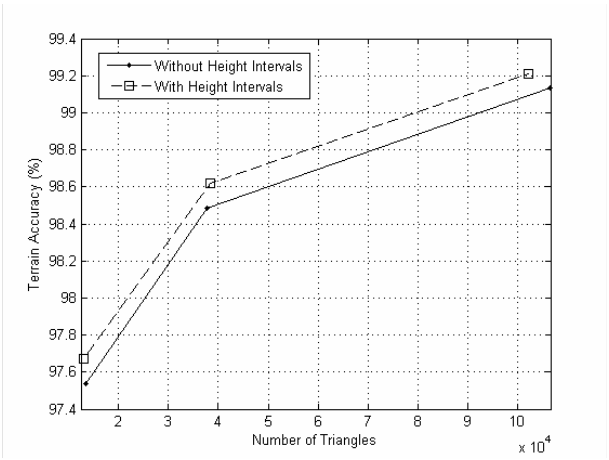

Fig.7. Results of the Technique 4 for test1.raw

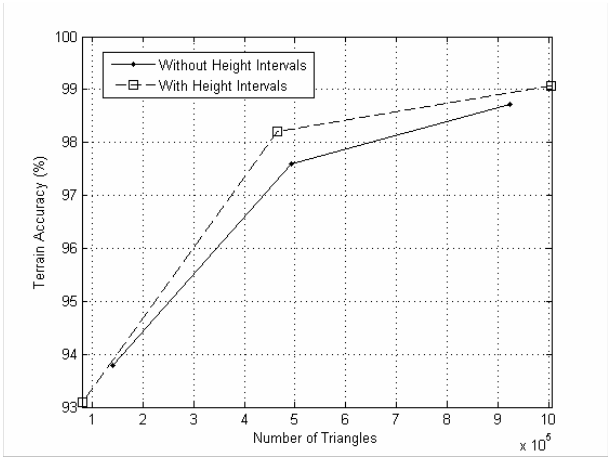

Fig.8. Results of the Technique 1 for test2.raw

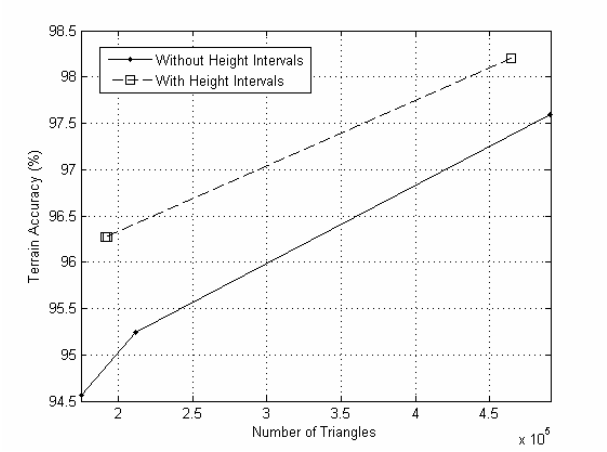

Fig.9. Results of the Technique 2 for test2.raw

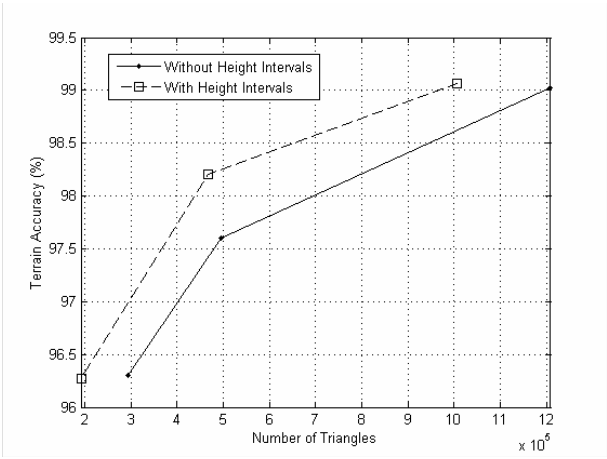

Fig.10. Results of the Technique 3 for test2.raw

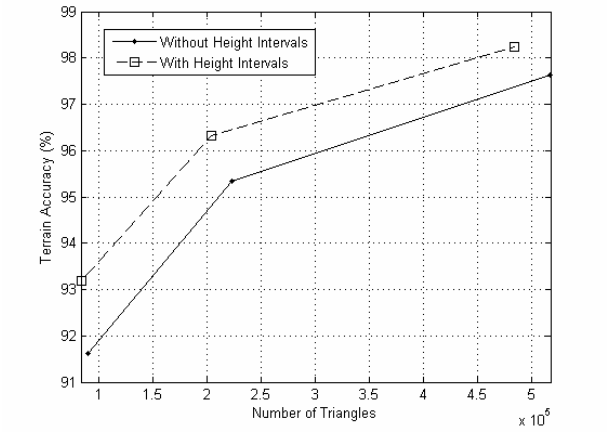

Fig.11. Results of the Technique 4 for test2.raw

6 Conclusion

We have presented a new algorithm that includes a technique to calculate the resolution parameters and three new techniques to calculate the error metric for the split operation of the quadtree triangulation. This algorithm results in greater terrain accuracy, with less triangles than the original quadtree triangulation algorithm. In particular, the proposed algorithm gives better results with increasing file sizes (see Fig.4 – Fig.11). The most powerful graphic card available for workstations at this time can handle around 100 million triangles per second with 256MB of graphic card memory. The importance of the contribution of the proposed algorithm becomes much more clear with increasing file sizes. This is very important for real-time applications because of the memory and capacity possibilities.

We are currently investigating the effect of using different numbers of height intervals. Finally, as a future work, new techniques to calculate the error metric may need to be improved and evaluated.

References

[1] F. Mello, E. Strauss, A. Oliviera, A. Gesualdi, Non-Uniform Mesh Simplification Using Adaptative Merge Procedures, 2000.

[2] B. Yang, W. Shi, O. Li, An integrated TIN and Grid method for constructing multi-resolution digital terrain models, *International Journal of Geographical Information Science*, Vol. 19, No. 10, 2005, pp. 1019–1038.

[3] P. Magillo, V. Bertocci, Managing Large Terrain Data Sets with a Multiresolution Structure, *11th International Workshop on Database and Expert Systems Applications Proceedings,* 2000, pp. 894-898.

[4] S.Röttger, W. Heidrich, P. Slusallek and H. Seidel, Real-Time Generation of Continuous Levels of Detail for Height Fields, 1998.

[5] M. Lee, H. Samet, Navigating through triangle meshes implemented as linear quadtrees, *ACM Transactions on Graphics*, 19, 2000, pp. 79–121.

[6] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, G. Turner, Real-time continuous level of detail rendering of height fields*, Computer Graphics*, 20, 1996, pp. 109–118.

[7] M. Duchaineau, M. Wolinsky, D. Sigeti, ROAMING terrain: real-time optimally adapting meshes*, Proceedings of IEEE Visualization'97*, Phoenix, Arizona (IEEE Computer society), 1997, pp. 81–88.

[8] W. Evans, D. Kirkpatrick, G. Townsend, Right triangular irregular networks, Technical Report 97-09, Department of Computer Science (Tucson, Arizona; University of Arizona), 1997.

[9] P. Lindstrom, V. Pascucci, Terrain simplification simplified: a general framework for view-dependent out-of-core visualization, *IEEE Transactions on Visualization and Computer Graphics*, 8, 2002 pp. 239–254.

[10] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, R. Scopigno, BDAM: batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22, 2003, pp. 505–514.

[11] D. Cine, P. Egbert, Terrain Decimation Through Quadtree Morphing, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 7, 2001, pp. 62-69.

[12] R. Pajarola, Overview of Quadtree-based Terrain Triangulation and Visualization, UCI ICS Technical Report No. 02-01, 2002.

[13] M. Lanthier, D. Bradley, Evaluation of Real-Time Continuous Terrain Level of Detail Algorithms, Ottawa Carleton University, Honours Project, 2003.

[14] R. Pajarola**,** Large Scale Terrain Visualization Using The Restricted Quadtree Triangulation, 1998.

[15] R. Pajarola, M. Antonijuan and R. Lario, QuadTIN: Quadtree based Triangulated Irregular Networks, *Proceedings IEEE Visualization,* 2002, pp. 395–402. IEEE Computer Society Press.

[16] Math Median Tutorial, Web Page Source, http://www.easycalculation.com/statistics/learn-median.php

[17] R. Wright, M.Sweet, B. Lipchak, *OpenGL SuperBible*, Third Edition, 2004.

[18] Plane Equation, MathWorld, Web Page Source, http://mathworld.wolfram.com/Plane.htm