Overview of Quadtree-based Terrain Triangulation and Visualization

Renato Pajarola

UCI-ICS Technical Report No. 02-01
Department of Information & Computer Science
University of California, Irvine

January 2002

Overview of Quadtree-based Terrain Triangulation and Visualization

Renato Pajarola
Computer Graphics Lab
Information & Computer Science Department
University of California Irvine
Irvine, CA 92697 - USA
Email: pajarola@acm.org

Abstract

Real-time rendering and multiresolution modeling of triangulated surfaces has attracted growing interest over the last decade. For interactive visualization of very large scale grid digital elevation models efficiency is a particularly important aspect. The graphics rendering load must be controlled by reducing the number of rendered primitives using level-of-detail (LOD) techniques, adaptive extraction of LOD triangle meshes in main memory must be performed at interactive frame rates, and loading-on-demand of elevation data from secondary storage has to be highly efficient as well. Due to the high regularity of grid digital elevation models, hierarchical quadtree based terrain triangulation models have been shown to be among the most efficient methods available. In this paper we will survey and compare several approaches that follow this quadtree based surface decomposition and triangulation.

Keywords: level-of-detail, multiresolution modeling, adaptive triangulation, terrain visualization, interactive rendering

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.

To reduce rendering complexity without negatively impacting the visual accuracy, the terrain triangulation must be adaptive to the terrain surface characteristics: allocating fewer and larger triangles for smooth surface areas, and using more and smaller triangles for sharp terrain features. Furthermore, the triangulation model must provide means to quickly extract surface representations at variable precision, such that multiple LODs can be supported at run-time. The following properties and requirements are very important for large scale multiresolution terrain triangulation and visualization:

1. fast adaptive triangulation
2. continuous LOD display
3. efficient geometric error metric
4. high-performance geometry rendering
5. efficient spatial access
6. compact representation and storage

To represent regular grid digital input data such as terrain height-field data in a multiresolution model, hierarchical spatial data structures such as quadtrees are most efficient. Quadtree based multiresolution triangulations have been shown to be exceptionally efficient for grid digital terrain data. In this paper we will review several quadtree based triangulation methods and compare their strengths and weaknesses with respect to efficiency according to the points listed above.

The remainder of the paper is organized as follows. In Section 2 we briefly review some related work on multiresolution triangulation and terrain visualization. Section 3 chronologically presents and compares the various existing quadtree based triangulation methods, and Section 4 discusses additional system level aspects of interactive terrain visualization applications. The paper is concluded with a summary and outlook in Section 5.

2. Related work

A number of approaches for mesh simplification and multiresolution surface triangulation have been developed over the last decade, and a thorough discussion is beyond the scope of this paper. For detailed overviews on general mesh simplification and multiresolution modeling see [HG97], [CMS98] and [Lue01]. Here we want to focus more on adaptive triangulation and mesh simplification particularly aimed at terrain triangulation and visualization.

Since height-fields are so called 2.5-dimensional surfaces – no overlap in elevation is possible for a particular geographical location – related work on terrain triangulation is often based on the principle of 2D *Delaunay* triangulations [Del34] to create *triangular irregular networks* (TINs) that represent the terrain surface. In [Lee91] and [GH95] several incremental point insertion and point removal methods are discussed and compared that can be used to create TINs efficiently from height-field data. The Delaunay pyramid [DeF89, VBH94] is a multiresolution extension of Delaunay based TINs in which groups of connected triangles are hierarchically replaced by sets with fewer triangles, see also [deBD95]. Other hierarchical triangulations as discussed in [DP95], [DMP96] and [Pup96] are mainly based on recursive subdivision of triangles. In [Hop98b] a hierarchical multiresolution terrain triangulation method is presented that is based on progressive application of edge-collapse and vertex-split mesh simplification and refinement operations [Hop96].

Similar to the approaches surveyed in this paper, a quadtree hierarchy is imposed on the height-field grid by the 2D wavelet transform applied in [GGS95] to simplify the terrain. In this approach elevation points are basically removed from the grid based on the significance of their wavelet coefficients, and the remaining quadtree-like point set is locally triangulated using look-up tables. Quadtree based triangulations as discussed here for terrain visualization have also been applied to image compression, for example in conjunction with wavelet decomposition [HK95].

In fact, several of these proposed terrain simplification and triangulation methods are particularly efficient with respect to triangulation performance (i.e. [GH95] or [Hop98b]) or LOD adaptivity (i.e. [Hop98b] or [GGS95]). Nevertheless, the quadtree based triangulation methods outlined below provide a more compact representation of the terrain (including all LODs up to full resolution), better spatial access, faster LOD triangulation and rendering, and are easier to implement as well. Note that these advantages come at some expense of increased complexity (size) of the rendered triangle mesh.

3. Quadtree based triangulation methods

In this section we discuss the various approaches of quadtree based adaptive triangulation of height-fields (or parametric two-dimensional surfaces). In particular we will focus on the triangulation algorithms, approximation error metrics, and representation of the multiresolution triangulation with respect to the requirements listed above in Section 1. An example of the type of simplified triangulated surfaces that can be constructed using the discussed quadtree based multiresolution triangulation methods is given in Figure 1.

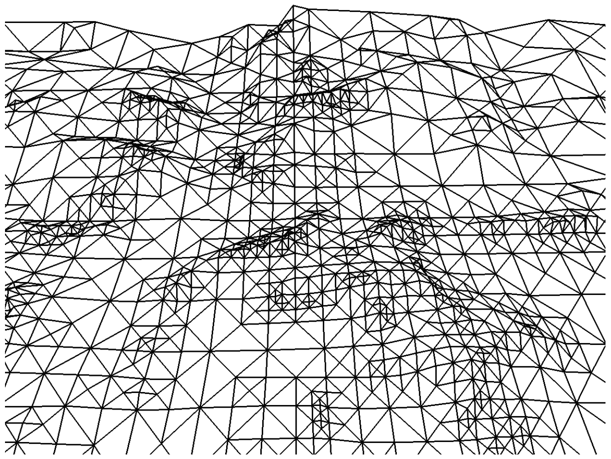


FIGURE 1. Adaptive quadtree based terrain triangulation.

3.1 Restricted quadtrees

Hierarchical, quadtree based adaptive triangulation of 2-manifold surfaces has first been presented in [VHB87] and applied to adaptively sample and triangulate curved parametric surfaces. In parameter space, the quadtree subdivision is performed recursively until for each sampled region the *Lipschitz* condition for the parametric curve is met that bounds the accuracy of the resulting polygonal approximation. Furthermore, the quadtree subdivision is restricted such that neighboring regions must be within one level of each other in the quadtree hierarchy as shown in Figure 2.

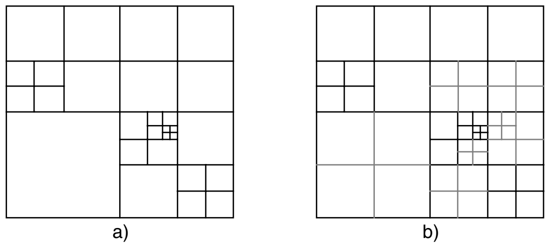


FIGURE 2. Example of an unrestricted quadtree subdivision in parameter space in a), and the restricted subdivision in b).

The basic approach for triangulation and visualization uses the following steps:

1. Initial sampling on a uniform grid.
2. Evaluation of each region with respect to some acceptance criteria (approximation error).
3. 4-split of unacceptable regions.
   Repetition of Steps 2 and 3 until adaptive sampling satisfies acceptance criteria over the entire surface.
4. Triangulation and rendering of all quadtree regions.

To prevent possible *cracks* in the polygonal representation of a restricted quadtree as shown in Figure 3, every quadtree region is triangulated with respect to the size of its adjacent regions. Due to the constraint of the restricted quadtree hierarchy that the levels of adjacent regions differ at most by one, the regions can be triangulated such that no cracks appear as outlined below. Such a crack-free triangulation is also called *matching*.

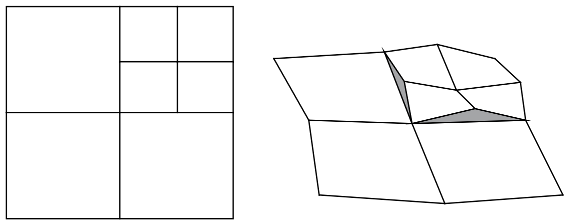


FIGURE 3. Cracks (shaded in grey) resulting from a quadrilateral polygonal representation of a restricted quadtree.

The triangulation rule as stated in [VHB87] is the following: Each square is subdivided into eight triangles, two triangles per edge, unless the edge borders a larger square in which case a single triangle is formed along that edge. Figure 4 shows a triangulation of a restricted quadtree following the above rule.

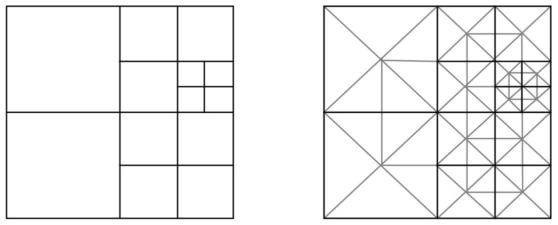


FIGURE 4. Matching triangulation of a restricted quadtree subdivision.

To render deformed and intersecting parametric surfaces, several recursive subdivision criteria are given in [VHB87] that take into account local curvature, intersection of surfaces, and silhouette boundaries. While these subdivision criteria are not directly applicable to terrain height-fields, the local curvature criterion, or flatness, is similar to other geometric approximation error metrics used for terrain triangulation.

No detailed algorithms and data structures are given in [VHB87] to construct and triangulate a restricted quadtree from a given data set, and thus analysis and comparison with subsequent approaches is very limited. Nevertheless, the presented restricted quadtree subdivision and its triangulation forms the fundamental basis on which other quadtree based triangulation approaches are built.

3.2 Quadtree surface maps

In [SS92], the restricted quadtree technique is applied to 2.5-dimensional surface data consisting of points on a regular 2D-grid and each point having a *height* value associated with it. This is the common raw representation of grid-digital terrain elevation models. In addition to the basic description of restricted quadtrees as presented in [VHB87], this paper provides two efficient construction algorithms to generate and triangulate a restricted quadtree from a set of points on a regular grid. One method is performed bottom-up and the other top-down to generate the restricted quadtree hierarchy. Furthermore, in [SS92] it is also observed that edges shared by two quadtree nodes on the same hierarchy level do not have to be split to guarantee a matching triangulation as shown in Figure 5 in comparison to Figure 4.

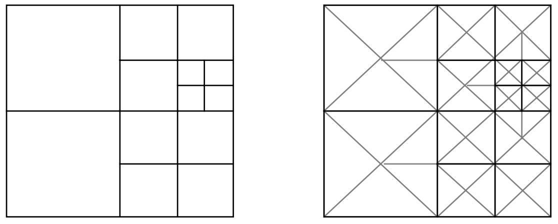


FIGURE 5. Improved matching triangulation of a restricted quadtree subdivision.

In the *bottom-up* construction method, the (square) input grid is partitioned into atomic nodes of 3x3 elevation points as shown in Figure 6. These nodes form the leaf nodes of a complete and balanced quadtree over the entire input grid.

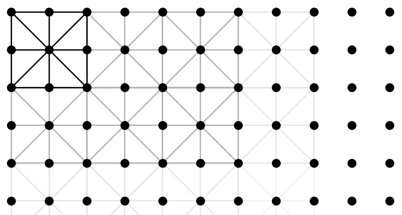


FIGURE 6. Atomic leaf-node for bottom-up construction of restricted quadtree.

The main phase of this method then consists of coalescing all mergible nodes bottom-up to create the restricted quadtree. Nodes must pass two main criteria before they can be merged:

1. *Error measure*: The approximation error introduced by removing the non-corner vertices of the nodes being merged must be within the tolerance of a given error threshold.
2. *Quadtree constraints*: The size of the node is equal to the size of its three siblings in the quadtree hierarchy, and neither the node nor its siblings have any smaller-sized neighbors.

The approximation error of Criterion 1 used in [SS92] is simply the vertical distance of the removed vertices with respect to their linear interpolation provided by the parent node as shown in Figure 7. Error of vertex B is its vertical distance to the average elevation of A and C. An example of mergible nodes is given in Figure 7, given that the approximation error of all removed vertices (outlined points in Figure 7-b) is within the given tolerance and given that no outside neighbors are smaller than the nodes in Figure 7-a. The algorithm terminates if no more merges can be performed, and it has a linear space and time cost O(n) in the size n of the input data set.

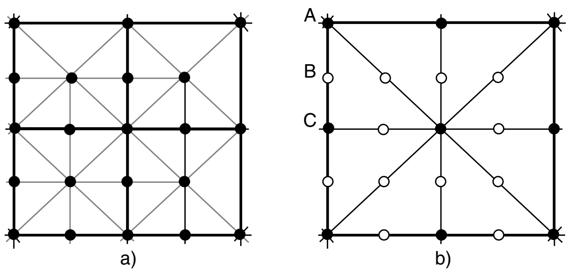


FIGURE 7. Initial four leaf nodes shown in a) that are merged in b) with the outlined points denoting the removed vertices in the merged node.

A major problem of the proposed bottom-up restricted quadtree construction method is the computation of the approximation error. While the vertical distance of a removed vertex (B in Figure 7) with respect to its linear interpolation (between A and C in Figure 7) in the immediate parent node may be below a given error threshold ε, it is not guaranteed that this removed vertex is within ε distance to a final result of the iterative bottom-up merging process of nodes as shown in a 2D example in Figure 8. In fact, the resulting simplified surface based on this method does not interpolate the removed vertices within any bounded distance.

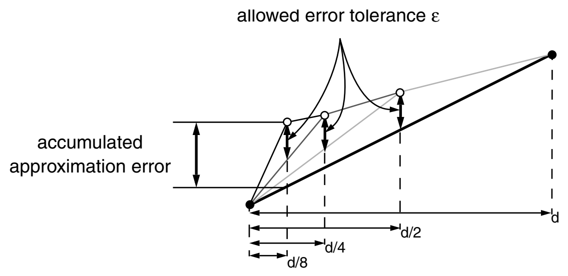


FIGURE 8. Merging of nodes satisfying the approximation error threshold locally may result in intolerable large accumulated errors with respect to the final result.

The second algorithm presented in [SS92] is a *top-down* construction of the restricted quadtree hierarchy. This method starts with representing the entire data set simplified by one root node and splits nodes recursively, never merges any, as necessary to approximate the data set. The method maintains at all time the restricted quadtree property that adjacent leaf nodes do not differ by more than one level in the hierarchy.

Vertices that can conceptually be removed by merging four sibling nodes are called *non-persistent*. Starting with the root node as shown in Figure 9-a, for each node of the partially constructed restricted quadtree the non-persistent vertices are identified in the input data set and their height value compared to the approximation of the current node. If any non-persistent vertex is not within the tolerated distance it is added to the current quadtree. However, insertion of vertices can lead to complex updates of the quadtree as outlined below.

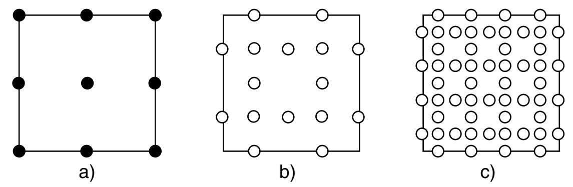


FIGURE 9. Vertices of the root node (level 0) shown in a), as well as the non-persistent vertices of level 1 in b) and level 2 in c).

To permanently maintain a restricted quadtree, the insertion of a vertex can lead to propagated splits in the parent and adjacent quadtree nodes. As shown in Figure 10, it may happen that a node on level l is not split because no vertices of level l+1 are inserted, however, a vertex v_2 on level l+2 has to be added. This insertion cannot be performed directly since no parent node covering v_2 has been created yet on level l+1. First the parent node of v_2 and its siblings on level l+1 have to be inserted by splitting the smallest node on level l enclosing v_2 into four nodes. Such propagated splits can occur over multiple levels.

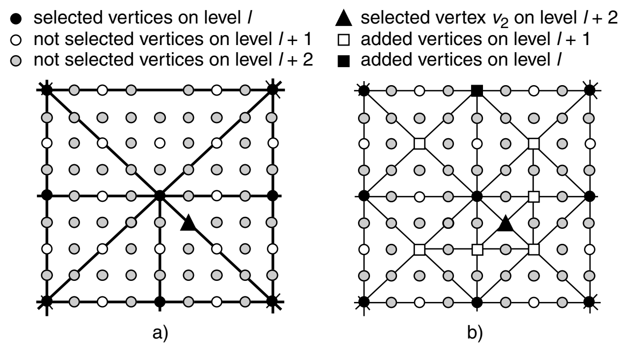

● selected vertices on level l
○ not selected vertices on level $l + 1$
○ not selected vertices on level $l + 2$
▲ selected vertex v_2 on level $l + 2$
□ added vertices on level $l + 1$
■ added vertices on level l
a)
b)

FIGURE 10. Starting triangulation of a node on level l is shown in a). No vertices are initially selected on level l+1. The selection of a vertex on level l+2 leads to forced splits and added vertices on previous levels as shown in b).

Not properly covered in the paper [SS92] is the situation where splits do not only have to be propagated vertically up in the hierarchy but also horizontally to other adjacent nodes as shown in Figure 11. To resolve such conflicts, all adjacent nodes of any node that is split have to be visited and updated accordingly. This split propagation to spatially adjacent nodes can in the worst case affect all nodes of the current quadtree, however, in the expected case will only influence quadtree nodes within a limited neighborhood.

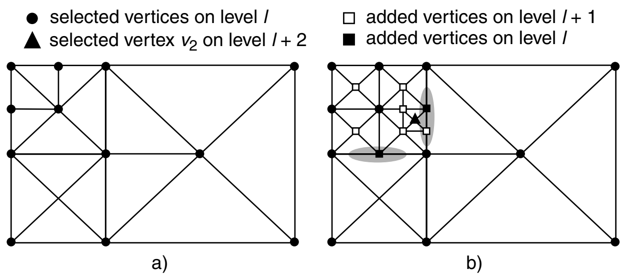

● selected vertices on level l
▲ selected vertex v_2 on level $l + 2$
□ added vertices on level $l + 1$
■ added vertices on level l
a)
b)

FIGURE 11. Initial triangulation of level l shown in a). Addition of vertex v_2 on level l+2 can lead to inconsistencies not only within the parent nodes of v_2 as shown in b).

The proposed top-down algorithm to create an adaptive surface mesh processes the entire data set and thus its cost is O(n), linear in the size n of the input. While the number of generated quadtree nodes is indeed output sensitive, the overall run-time is still directly proportional to the input data set since all vertices have to be visited.

In contrast to the bottom-up algorithm, this top-down method correctly calculates the approximation error at each vertex with respect to the current restricted quadtree surface approximation. Therefore, no accumulation of errors beyond the given threshold ε can occur, and the reconstructed surface map is a correct ε-approximation.

Both methods presented in [SS92] operate on a hierarchical quadtree data structure that must provide functionality for inserting vertices, calculating distance of a vertex to a piece-wise linear surface approximation, neighbor finding, and for merging and splitting nodes. Furthermore, the restricted quadtree nodes must be post-processed to generate the resulting matching triangulation. The presented algorithms are capable of creating adaptive and continuous LOD triangulations within the limits of the proposed error metric. However, efficiency is not suitable for real-time rendering of very large terrain data sets due to the input sensitiveness of the triangulation algorithms.

3.3 Continuous LOD rendering

A different approach to generating and triangulating a restricted quadtree is presented in [LKR+96] based on the notion of *triangle fusion*. Starting with a triangulation of the entire grid-digital terrain data set the triangle mesh is simplified bottom-up by consecutive merging of triangles. The full resolution grid triangulation as shown in Figure 12-a is equivalent to the atomic leaf nodes of the bottom-up triangulation method in [SS92]. Triangle merging is performed in two phases as shown in Figure 12-b and c. First, in an atomic node pairs of isosceles triangles (i.e. a_l and a_r in Figure 12-b) sharing a short edge are coalesced and the midpoint on the boundary edge of the quad is removed. In the second phase the center vertex of a quad region is removed by merging isosceles triangle pairs along the diagonal (i.e. e_l and e_r in Figure 12-c). To prevent cracks from occurring due to triangle merging, always two pairs of isosceles triangles that all share the same removed *base vertex* must be coalesced at the same time (i.e. the pairs e_l, e_r and f_l, f_r in Figure 12-c).

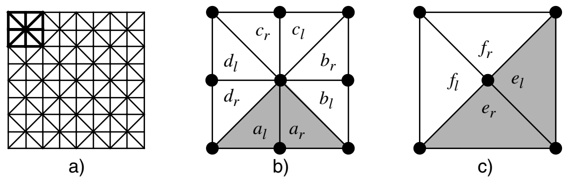


FIGURE 12. Full resolution triangulation shown in a). Merging of triangle pairs along the bottom boundary edge shown in b) and along the diagonal in c).

One of the main contributions of [LKR+96] is the introduction of vertex dependencies that can be used to prevent cracks and create matching triangulations at variable LOD. For example, considering Figures 12-b and 12-c it is clear that the midpoint of the bottom edge on level l, the base vertex of triangles a_l and a_r, cannot be part of a matching triangulation if the center vertex of the quad region, the base vertex of e_l, e_r, is missing. Or from the opposite viewpoint, the base vertex of triangles e_l, e_r and f_l, f_r cannot be removed if any of the base vertices of triangle pairs a_l,a_r, b_l,b_r, c_l,c_r or d_l,d_r persists. These constraints of a matching restricted quadtree triangulation define a binary, hierarchical dependency relation between vertices as shown in Figure 13. Each vertex to be included in a triangulation depends on two other vertices (on the same or lower resolution) to be included first. Therefore, a triangulation of a (restricted) quadtree is a matching triangulation if no such dependency relation is violated. The triangulation method proposed in [LKR+96] recursively resolves the dependency relations of a set S of selected vertices (i.e. all vertices exceeding a given error tolerance) as follows: For each vertex $v \in S$, all its dependent parents according to the dependency rules shown in Figure 13 are recursively activated and included in the triangulation as well.

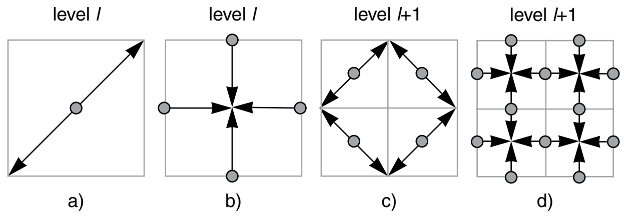


FIGURE 13. Dependency relation of a restricted quadtree triangulation. The center vertex in a) depends on the inclusion of two corners of its quad region. The boundary edge midpoints in b) depend on the center vertex of the quad region. Dependencies within and between the next resolution levels are shown in c) and d).

The second major contribution of [LKR+96] is the definition of an efficient view-dependent image-space error metric that determines removal or inclusion of vertices. As shown in Figure 14, the basic idea of this error metric is that triangle pairs can be merged if the change in slope δ at the removed base vertex v projected into screen space is smaller than a given threshold τ. The line segment $\delta = v - \bar{v}$ between the removed base vertex v and its linear interpolation $\bar{v} = (v_l + v_r)/2$ is perspectively projected into the screen space viewing plane as δ_{screen}. If δ_{screen} is smaller than the tolerance τ then the vertex v can be removed and the corresponding triangle pairs merged. Note that the projected delta segment δ_{screen} is not defined with respect to the highest resolution mesh or the current LOD mesh but rather based on the adjacent vertices v_l and v_r of the next lower resolution in the quadtree. Therefore, although hardly noticeable in practice, this metric suffers from the same limitations as the error computation of the bottom-up triangulation method presented in [SS92] and does not provide a guaranteed error bound on the final triangulation.

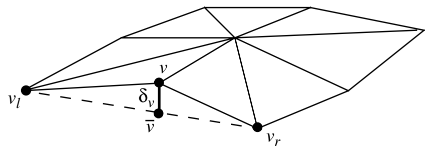


FIGURE 14. Vertical distance δbetween removed base vertex v and its linear interpolation $\bar{v}$.

The above view-dependent image-space error metric is also extended to entire quadtree blocks in [LKR+96] to allow for more efficient block-based simplification. In particular, if for a quad region R the maximum delta projection of all higher resolution vertices within R is smaller than the threshold τ then they can be replaced by a lower resolution. For an axis-aligned bounding box of a quadtree block R and given viewing parameters, one can compute the smallest elevation delta δ_l and largest δ_h of that box that when projected onto screen may exceed τ. Therefore, if the maximum vertical error δ_{max} of all vertices $v \in R$ is smaller than δ_l then R can be replaced by a lower LOD block, and if δ_{max} is larger than δ_h then R has to be split. Otherwise the screen space projected error δ_{screen} of all vertices $v \in R$ has to be computed and compared to τ.

Another performance feature presented in [LKR+96] is the construction of triangle strips for fast rendering. In fact, a triangulation of a restricted quadtree can be represented by one single *generalized triangle strip*.[1] The triangle strip generation method described in [LKR+96] is based on a recursive pre-order traversal of the triangular quadrants of quadtree blocks. Starting with a counterclockwise ordering of triangular quadrants of the root node as shown in Figure 15-a, each quadrant is recursively traversed and the traversal is stopped when a triangle is not further subdivided. In alternating order, children of triangular quadrants are visited *left-first* as for quadrant q_0 (and in Figure 15-c), or *right-first* as for the triangles in the next level shown in Figure 15-b. Based on this traversal, vertices can be ordered and outputted to form a generalized triangle strip for efficient rendering (see [LKR+96] for code details).

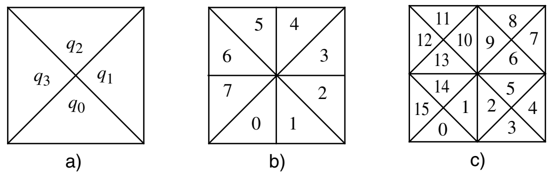


FIGURE 15. Recursive quadtree traversal for triangle strip generation. Initial order of triangular quadrants shown in a) with *left-first* traversal for odd subdivisions shown in b), and *right-first* traversal of even subdivision steps shown in c).

Despite the fact that an entire triangulated restricted quadtree can be represented by one triangle strip, triangle strips are formed for individual blocks only in [LKR+96]. For each frame the block-based view-dependent image-space error metric outlined above is used to form a (non-restricted) quadtree subdivision S of the terrain. For each block $b \in S$ of this subdivision, the vertex-based error metric is used to achieve a fine-grain selection of vertices to be included in the triangulation. Furthermore, the vertex dependencies are resolved at this stage to guarantee a matching triangulation. Finally, for each quadtree block $b \in S$ a triangle strip is generated and used for rendering.

The triangulation method presented in [LKR+96] is very efficient in terms of rendering performance. The triangulation algorithm is output sensitive since the quadtree subdivision is performed top-down and does not need to examine all vertices on the highest resolution. Furthermore, efficient rendering primitives in form of triangle strips are generated for optimized rendering. Despite the fact that the view-dependent error metric does not provide a guaranteed error bound, it is very efficient in practice and provides good terrain simplification while maintaining plausible visual results. In terms of large scale terrain visualization, several system aspects such as efficient representation, dynamic scene management or spatial access are left uncovered.

1. Generalized triangle strips allow swap operations.

3.4 Real-time optimally adapting meshes

The *Real-time Optimally Adapting Meshes* (ROAM) triangulation method presented in [DWS+97] is conceptually very similar to [LKR+96]. It is based on the notion of a *triangle bintree* hierarchy as shown in Figure 16 which is a special case of the *longest side bisection* triangle refinement method described in [Riv93, Riv95]. This method recursively refines triangles by splitting the longest edge at the *base vertex* (see also Figure 12).

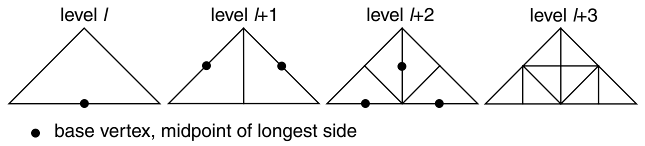


FIGURE 16. Binary longest side bisection hierarchy of isosceles triangles with indicated split vertices on the longest side.

As shown in Figure 17, for a refinement operation two triangles are split at the base vertex of their common longest edge, and a simplification operation consists of merging two triangle pairs at their common base vertex.

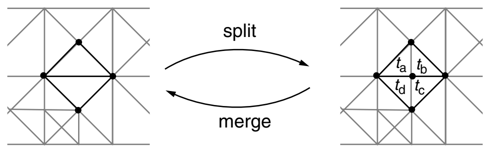


FIGURE 17. Split and merge operations on a bintree triangulation.

An important observation is that in a matching triangulation, all neighbors of a triangle t on level l in the bintree hierarchy must be either on the same level as t, or on levels l+1 or l-1 of the bintree hierarchy. Therefore, two pairs of triangles t_a,t_b and t_c,t_d sharing the same base vertex can only be merged if they are all on the same level in the bintree hierarchy as shown in Figure 17. Furthermore, a triangle t cannot be split immediately if its neighbor t_{long} across its longest edge is from a coarser level as shown in Figure 18. Triangle t can only be split if its corresponding neighbor is forced to split first. These forced splits are conceptually the same as the split propagation of [SS92] shown in Figure 10. Moreover, the dependency relation of [LKR+96] in Figure 13 denotes exactly the same forced split propagation of a bintree or restricted quadtree triangulation.

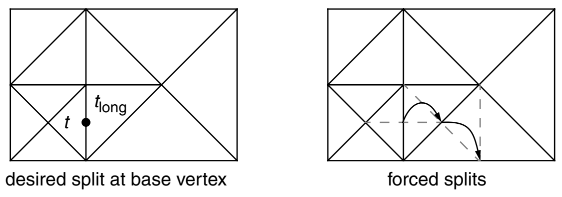


FIGURE 18. Propagation of forced triangle splits.

The run-time triangulation is based on a greedy algorithm that maintains two priority queues of the triangles $t \in T$ of the current mesh T: The split queue Q_s stores the triangles $t \in T$ according to their priority to be split next, and the merge queue Q_m maintains the mergible triangle pairs of T. For each frame the priority queues Q_m and Q_s are used to incrementally simplify and refine the current triangle mesh T to reach a triangulation that satisfies the given error threshold τ. The priorities of Q_s and Q_m are based on the error metric defined on the triangles.

To guarantee an ε-approximation with respect to a particular error metric, the proposed greedy algorithm requires the error metric, and thus the priorities of Q_m and Q_s, to be *strictly monotonic*. This means that the error or priority of any triangle in the bintree hierarchy cannot be larger than its parent's priority. This monotonicity requirement limits the direct applicability of many standard error metrics. For example, neither the view-dependent error metric in [LKR+96] nor the vertical distance measure of [SS92] (see also Figure 8) or the Hausdorff distance error metric defined hierarchically on removed vertices or triangles initially satisfy this monotonicity requirement. Special care has to be taken to enforce monotonicity of any error metric by a bottom up traversal of the triangle bintree hierarchy and calculating bounding priorities at each node.

In addition to the bintree representation of the triangulation of a restricted quadtree, several geometric object-space and image-space error metrics are proposed in [DWS+97]. An object-space geometric approximation error metric is defined by calculating for each triangle t in the bintree hierarchy the thickness δ_t of a bounding *wedgie* that encloses all children of its subtree as shown in Figure 19. This wedgie of a triangle is also used to estimate the maximal screen-space distortion, similar to the approach presented in [LKR+96]. For a given triangulation T, the distortion can be bounded by the maximum projected length δ_{screen} of δ_t for all triangles $t \in T$.

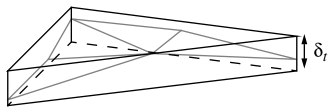


FIGURE 19. The thickness of a bounding wedgie defines an object-space geometric approximation error.

Besides the two main contributions of ROAM which are the priority-queue driven triangle-bintree based triangulation method and the screen distortion error metric, the paper [DWS+97] also contains many smaller interesting contributions. A list of twelve criteria is given that generally apply to mesh simplification and in particular to large scale terrain visualization. Additionally to the above screen-space error metric several priority metrics are proposed such as: backface detail reduction, surface normal distortion, texture-coordinate distortion, silhouette preservation, view frustum culling, atmospheric or depth attenuation, and region of interest. Furthermore, a few performance enhancements that are implemented in ROAM are described including view-frustum culling, incremental triangle strip generation, deferred priority recomputation, and progressive optimization.

The ROAM terrain triangulation method [DWS+97] is designed to provide guaranteed frame rates for an interactive terrain visualization system. Despite the maintenance of priority queues at run-time, which requires order $O(n \log n)$ cost for each update, the method is efficient since it is output sensitive (for monotonic error metrics) and because the triangulation can be updated incrementally between rendered frames. In addition to the basic algorithms, a couple of system level issues are discussed as well such as reducing the amount of CPU time spent on recalculating priorities for each frame, or limiting the number of split and merge operations to bound the triangle count and guarantee consistent frame rates.

3.5 Right-triangulated irregular networks

Right-Triangulated Irregular Networks (RTIN) as presented in [EKT97] is a multiresolution triangulation framework for the class of restricted quadtree triangulations that is based on the same triangle bintree hierarchy as ROAM [DWS+97]. The RTIN approach is particularly focused on the efficient representation of the binary triangle hierarchy, and fast mesh traversal for neighbor-finding. Starting with a square triangulated by choosing one diagonal, triangles are split recursively at the base vertex or midpoint of their longest edge, identical to the method described above in Section 3.4. To guarantee a matching triangulation without cracks the same propagation of forced splits as shown in Figure 18 is imposed on the RTIN triangulation. In [EKT97] it is observed that split propagation caused by splitting a triangle t on level l_t cannot cause triangles smaller than t to be split (on levels $l > l_t$), and that at most two triangles on each level $l \le l_t$ are split. Thus split propagation terminates in the worst case in $O(\log n)$ steps, with n being the size of the triangle bintree hierarchy (number of leaf nodes).

One of the main contributions of the paper is an efficient data structure to represent right-triangular surface approximations. Similar to Figure 12, child triangles resulting from a split are labelled as *left* and *right* with respect to the split vertex of their parent triangle. A binary labelling scheme as shown in Figure 20 is used in RTIN to identify triangular regions of the approximation. A RTIN triangulation is thus represented by a binary tree denoting the triangle splits and the elevation values (z coordinate) of the triangle vertices. The geographical (x and y) coordinates do not have to be stored for each vertex but can be computed from the triangle's label. As noted in [EKT97], a main memory implementation of such a binary tree structure with two pointers and three vertex indices[1] per node is space inefficient if used to represent one single triangulated surface approximation. However, a triangle bintree actually represents an entire hierarchy of triangulations. To reduce the storage cost of a triangle bintree hierarchy it is proposed to remove child

1. Could be reduced to only one vertex index, others are known from parent triangle nodes.

pointers by storing the nodes in an array and using an array indexing scheme based on the node labels.

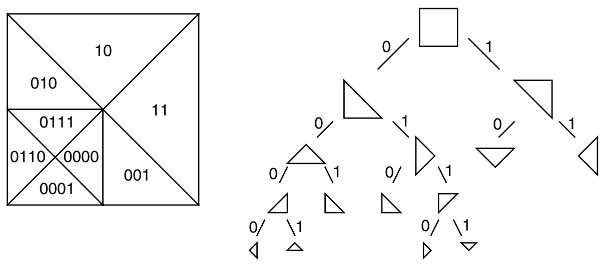


FIGURE 20. RTIN triangle bintree labelling using 0 for *left* and 1 for *right*.

Based on the binary tree representation of the RTIN hierarchy as shown in Figure 20, an efficient neighbor finding scheme is the second main contribution of [EKT97]. Given a counterclockwise numbering from v_1 to v_3 of the vertices of triangle t with vertex v_3 being the right-angled vertex, the *i-neighbor* of triangle t is defined as the adjacent triangle t_i that does not share vertex i. Furthermore, the *same-size i-neighbors* of any triangle are the edge adjacent triangles at the same level in the bintree hierarchy. For example, triangle 10 in Figure 20 is the same-size 1-neighbor of triangle 11, and triangle 001 is the 3-neighbor of triangle 0000 but not a same-size neighbor. The neighbor-finding function $N_I(t)$ presented in [EKT97] first finds the same-size i-neighbor of a triangle and then determines the actual i-neighbor for a particular triangulation. The recursive neighbor-finding function $N_I(t)$, that returns the label of the same-size i-neighbor of a given triangle t, is conceptually identical to a recursive tree traversal for finding adjacent regions in any binary space partition (BSP-tree), see also [Sam89a, Sam89b]. An efficient non-recursive implementation of $N_I(t)$ based on arithmetic and logical operations is also given in [EKT97].

For terrain visualization purpose, each triangle is assigned an approximation error during the preprocess phase of constructing the RTIN hierarchy. This approximation error is the maximal vertical distance to the triangle over all points covered by that triangle. At run-time, starting with the two triangles at the root of the RTIN hierarchy a depth-first traversal recursively splits triangles whose approximation errors exceed a given tolerance threshold. Forced splits are propagated to the corresponding i-neighbors to avoid cracks in the triangulated surface approximation.

The main focus of the RTIN approach is efficient representation of the triangle bintree hierarchy and neighbor finding techniques on the adaptively triangulated surface. Similar to the previous approaches [LKR$^+$96] and [DWS$^+$97] the RTIN method is efficient in creating an adaptive surface triangulation since its top-down algorithm is output sensitive. Furthermore, because the approximation error is calculated per triangle in the bintree hierarchy the RTIN method provides a guaranteed geometric error bound on the extracted triangulation and thus can be used to generate ε-approximations. The RTIN approach is almost identical to the ROAM method and only differs in notation and representation of the triangle bintree hierarchy. No detailed algorithms are given in [EKT97] on how to incorporate propagation of forced splits to generate a matching triangulation.

3.6 Restricted quadtree triangulation

The *Restricted Quadtree Triangulation* (RQT) approach presented in [Paj98a, Paj98b] is focused on large scale real-time terrain visualization. The triangulation method is based on a quadtree hierarchy as in [SS92] and uses the dependency relation presented in [LKR$^+$96] to generate minimally matching quadtree triangulations. Both, top-down and bottom-up triangulation algorithms are given in pseudo code for a terrain height-field maintained in a region quadtree, and where each vertex has a an approximation error associated with it. It is noted that the quadtree hierarchy can be defined implicitly on the regular grid input data set by appropriate point indexing and recursive functions, and no hierarchical data structure actually needs to be stored. This reduces the storage cost effectively down to only the elevation and approximation error values per vertex for such an *implicit quadtree*.

As shown in [Paj98b], for each point $P_{i,j}$ of the $2^k+1 \times 2^k+1$ height-field grid its level l in the implicit quadtree hierarchy can efficiently be determined by arithmetic and logical operations on the integer index values i and j, see also Figure 21. Furthermore, it is also observed that the dependency relation of Figure 13 can be expressed by arithmetic expressions as functions of the point's index i,j. The implicit definition of quadtree levels and dependency relations between points by arithmetic functions allows the top-down and bottom-up algorithms presented in [Paj98a] to run very fast and directly on the height-field grid data instead of relying on a hierarchical data structure (i.e. as in [SS92]).

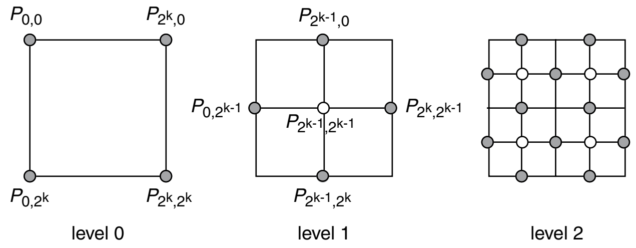


FIGURE 21. Implicit quadtree hierarchy and point indexing defined on the height-field grid.

Similar to the RTIN approach in [EKT97], the RQT method in [Paj98a] also proposes an approximation error metric based on vertical displacement that satisfies the monotonicity requirement outlined in [DWS$^+$97]. However, in contrast to [EKT97] and [DWS$^+$97] which define the error metric on triangles, the RQT approach defines the error metric on the vertices. Thus reducing the memory cost for storing error values significantly.

It has also originally been observed in [Paj98a, Paj98b] that for static precomputed error metrics the dependency graph shown in Figure 13 can be encoded into the error metric itself by a technique known as *error saturation*. As shown in Figure 22-a, the selection of a particular point *P* (black square) due to its error value *e*=9 exceeding the allowed tolerance τ=5 causes several forced splits (dashed grey lines). To avoid forced splits, error values are propagated and maximized along the dependency graph, as shown in Figure 22-b. Error saturation is performed in a preprocess: Each vertex stores the maximum value of all propagated errors and its own computed error, and propagates this maximum further in the dependency graph. Therefore, a simple and fast top-down selection of vertices according to their saturated error values directly yields a matching triangulation of a restricted quadtree without the need of enforcing any quadtree constraints, forced splits or resolving dependency relations. This error saturation techniques has also been observed in [GRW00] and can be applied in various ways to enforce constraints on multiresolution hierarchies such as topology preservation in isosurface extraction [GP00].

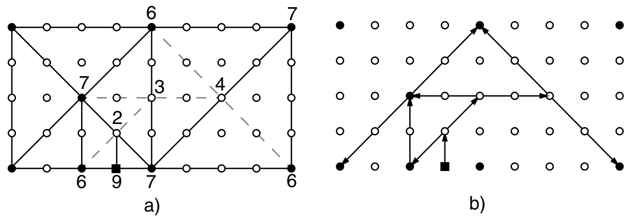


FIGURE 22. Initial error metric shown in a) for selected vertices, white vertices are below and black vertices above the error threshold τ=5. Forced splits are indicated with dashed grey lines. Propagation of error saturation shown in b) for the vertex causing the forced splits.

To improve rendering performance, a triangle strip construction algorithm is presented in [Paj98a] that traverses the quadtree hierarchy instead of the triangle bintree hierarchy as proposed in [LKR+96]. As shown in Figure 23, the RQT triangle strip that recursively circles counterclockwise around each quadtree block's center vertex is a *space filling curve* that visits all triangles exactly once. It also represents a *Hamiltonian* path in the dual graph of the triangulation. This triangle strip can be generated by a depth-first traversal of the quadtree in linear time, proportional to the size of the generated triangle strip. Moreover, the above error saturation technique and the quadtree based triangle strip generation support a highly efficient unified vertex selection, triangle strip generation and rendering algorithm based on a depth-first traversal of the implicit height-field quadtree.

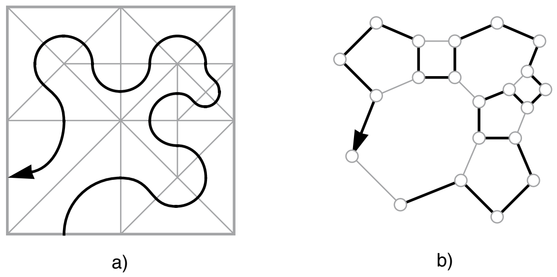
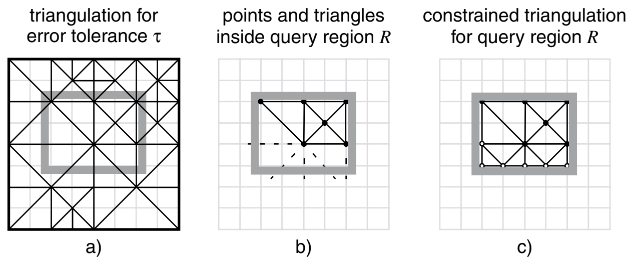


FIGURE 23. Generalized RQT triangle strip shown in a) and its Hamiltonian path on the dual graph in b).

In the context of large scale terrain visualization, level-of-detail (LOD) based spatial access to the restricted quadtree triangulation is also discussed in [Paj98a, Paj98b]. The RQT approach allows efficient spatial access by supporting rectangular range queries for a particular LOD. The query range can be any height-field grid aligned rectangular region *R*, and the LOD can be specified either by an error threshold or tolerance interval. As shown in Figure 24, special attention has to be paid to the border ∂R of the query region *R*. The geometrical test wether a vertex is inside the query region or not, is extended by constraints of the restricted quadtree triangulation. The border ∂R must include vertices to form a matching restricted quadtree triangulation of the query region *R*.


triangulation for error tolerance τ
points and triangles inside query region *R*
constrained triangulation for query region *R*
a)
b)
c)
FIGURE 24. Rectangular range query shown in a) and initial vertex selection given in b). RQT constraints are enforced on the range query as shown in c).

The restricted quadtree triangulation (RQT) approach [Paj98a] provides an efficient top-down triangulation algorithm similar to the other methods that is output sensitive and theoretically optimal in the sense of [Pup96]. Moreover, the introduced error saturation technique allows fast extraction of a matching triangulation at arbitrary LOD without the need for handling any forced split propagation. The bottom-up RQT vertex selection algorithm as presented in [Paj98a] can efficiently be used to perform the proposed error saturation in a preprocess phase. High-performance rendering is achieved by the fast triangulation method using error saturation as well as the efficient triangle strip generation technique that can actually be integrated directly into the point selection process. The RQT approach requires minimal representation cost due to the implicit definition of the quadtree hierarchy on the input height-field grid, and provides efficient spatial access at arbitrary LOD to rectangular regions of the large scale terrain data set. Furthermore, several system level aspects such as dynamic scene management and progressive mesh updates of a client-server based implementation are also discussed.

3.7 Multiresolution compression

In [Ger99], the class of restricted quadtree or right-triangular bintree triangulations is studied with respect to efficient data storage and processing, search and access methods, and data compression. It is proposed to manage the data always in compressed form, even interactive processing is performed on the compressed data. The multiresolution triangulation framework in [Ger99] follows the binary triangle hierarchy approach as used in [DWS+97] and [EKT97]. To prevent cracks in the triangulation resulting from recursive triangle bisection, error saturation as presented in [Paj98a] is applied.

The main contribution of the paper is a compressed representation of the triangle bintree hierarchy based on an efficient mesh traversal and triangle numbering scheme. The traversal order of triangles in the bintree hierarchy is equivalent to the triangle strip ordering as shown in Figure 23. Furthermore, each triangle is numbered such that the left child of a triangle with number n receives the number $2n$ and the right child is numbered $2n+1$ if the level l of the parent triangle is odd and vice versa if it is even as shown in Figure 25. For a given triangle, bitwise logical operations can be used to compute the adjacent triangle that shares the common refinement vertex. Each vertex is associated with the two numbers of the triangles that it refines.

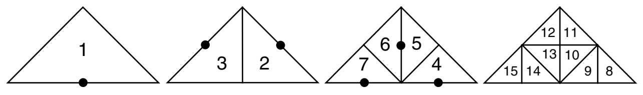


FIGURE 25. Triangle numbering.

This ordering and triangle numbering imposes a binary classification of triangles in a matching bintree triangulation into up- or down-triangles. In a depth-first traversal of the bintree hierarchy, an *up-triangle* can only be followed by a triangle on the same or higher level (coarser triangle) in the hierarchy. Similarly, a *down-triangle* can only have a neighbor on the same or lower level of the hierarchy. Therefore, the starting triangle and one bit per triangle is sufficient to encode an adaptive bintree triangulation. Furthermore, vertices only need to be specified on their first occurrence in the bintree traversal. Based on this traversal and numbering technique an efficient compressed representation of a triangle bintree hierarchy is proposed. Moreover, it is shown how an arbitrary adaptive triangulation can efficiently be extracted from the code stream that represents the entire bintree hierarchy, and that can be read and processed efficiently from disk.

The triangulation algorithm and data structure presented in [Ger99] are particularly tailored towards efficient representation and traversal of the binary triangle hierarchy. The proposed encoding of the triangle bintree is very interesting from the point of view that it can be used to access an adaptive triangulation efficiently even if the bintree is stored sequentially on disk. The proposed multiresolution triangulation framework provides most of the important features such as continuous LOD, fast rendering, and compact representation.

3.8 Summary

The multiresolution terrain triangulation approaches reviewed in this section all contribute unique features and improvements to the class of restricted quadtree and bintree triangulations. The basic adaptive multiresolution triangulation framework is outlined in [SS92]. The approaches of [LKR+96] and [Paj98a] follow this concept of an adaptive quadtree hierarchy, while the methods presented in [DWS+97], [EKT97] and [Ger99] describe the same class of triangulations based on a binary triangle subdivision hierarchy.

The most efficient triangulation algorithms are [LKR+96] and [Paj98a] which are based on a simple vertex selection strategy and both using the dependency relation introduced in [LKR+96] to guarantee matching triangulations. The RQT method in [Paj98a] takes this approach even a step further by incorporating the dependency relation into the error metric similar to [Ger99]. While very efficient, the other triangle bintree based approaches are slightly more complex due to recursively splitting triangles and resolving propagated forced splits, and thus have some disadvantages compared to the simple quadtree based vertex selection algorithms. All methods are capable of generating smooth adaptive LODs for efficient terrain surface approximation, and while not explicitly described RTIN [EKT97] can generate triangle strips for fast rendering.

The error metric for the triangle bintree approaches [DWS+97, EKT97, Ger99] is defined on the triangles in the binary hierarchy. Due to the property of a binary tree having roughly $2n$ nodes for n leaf nodes and a triangle mesh having $2n$ triangles for n vertices, storage of a triangle based error metric requires maintaining about $4n$ error values. In contrast, the quadtree based approaches [LKR+96, Paj98a] define the error metric on vertices and only require n error values to be stored. Simple geometric approximation error metrics based on vertical displacement can be found in [SS92], [EKT97], [Paj98a] and [Ger99]. More sophisticated view-dependent error metrics such as screen space distortion are discussed in [LKR+96] and [DWS+97]. A combination of global geometric approximation errors and screen-space error metrics will be most efficient for large scale terrain visualization in practice. In [DWS+97] it was observed that an error metric must be hierarchically monotonic to guarantee ε-bounded approximations, and RTIN [EKT97] as well as RQT [Paj98a] provide such monotonic geometric error metrics.

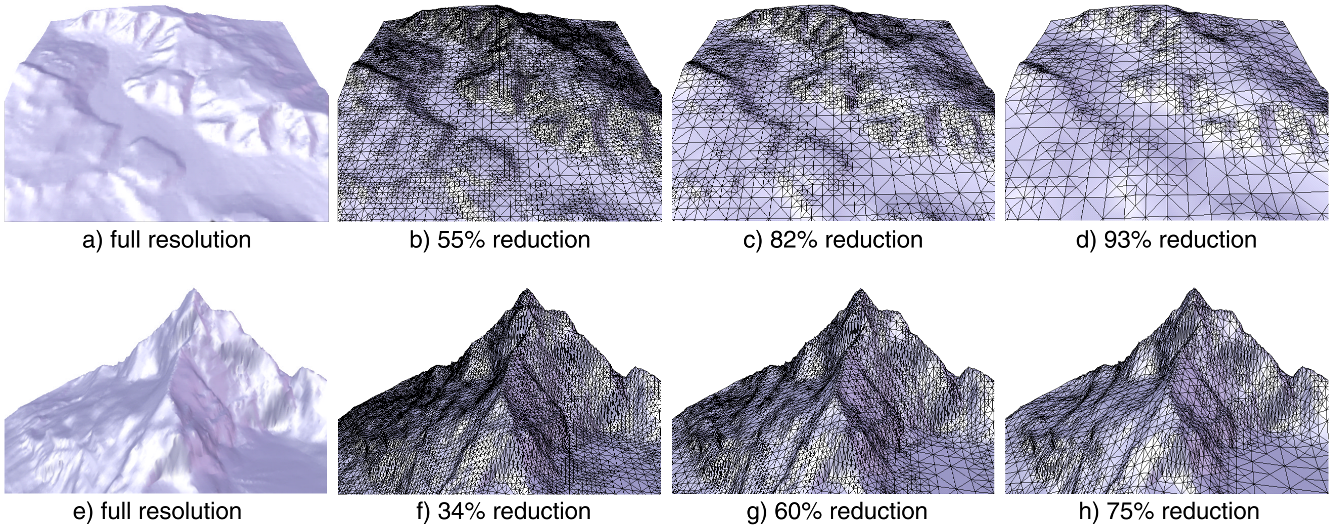


FIGURE 26. Top row of images shows the *Albis* terrain patch with a) 25921, b) 11636, c) 4664 and d) 1789 points. The bottom row shows the *Matterhorn* terrain patch with e) 25921, f) 17156, g) 10492 and h) 6367 points.

The most efficient representation of a multiresolution triangulation of a height-field is the implicit quadtree structure with a saturated error metric defined on the grid of elevation values as proposed in [Paj98a]. This representation does not require any information to be stored that describes the multiresolution hierarchy, and only needs the elevation and error values for each grid point. Furthermore, such an elevation grid can also efficiently be partitioned and stored on disk as shown in [Paj98a] and [Paj98b]. Efficient representation and compression of the triangle bintree hierarchy is discussed in [EKT97] and [Ger99], however, both approaches use triangle based error metrics which are space inefficient due to the large number of error values that have to be stored. Efficient LOD-based spatial access and triangulation is discussed in [Paj98a], and extraction of an adaptive triangulation in a sequentially stored and compressed triangle bintree representation is considered in [Ger99].

In addition to the technical comparisons between individual methods given above we also want to provide some visual examples of triangulation results that can be achieved with restricted quadtree or right-triangular bintree multiresolution triangulation frameworks. However, for detailed numerical simplification results consult the referenced original papers.

Figure 26 shows different simplification results with error tolerances (from left to right) of 0, 2, 5 and 10 meters on two topologically very different terrain patches. Both sample regions have an original grid resolution of 25 meters and an elevation accuracy of one meter. Already a very small error tolerance results in a drastic reduction of the number of points used to sufficiently approximate and triangulate the terrain for real-time visualization. Although the mesh complexity itself is reduced drastically, the detail quality of a moderate error tolerance of 5 meters is still excellent, see also Figure 26-c and Figure 26-g. Even with a 10 meter error threshold and high simplification rates, significant visual details are retained as shown in Figure 26-d and Figure 26-h. Furthermore, smooth shading and appropriate texturing of the adaptively triangulated surfaces as shown in Figure 27 smoothens out any remaining differences between different LODs.

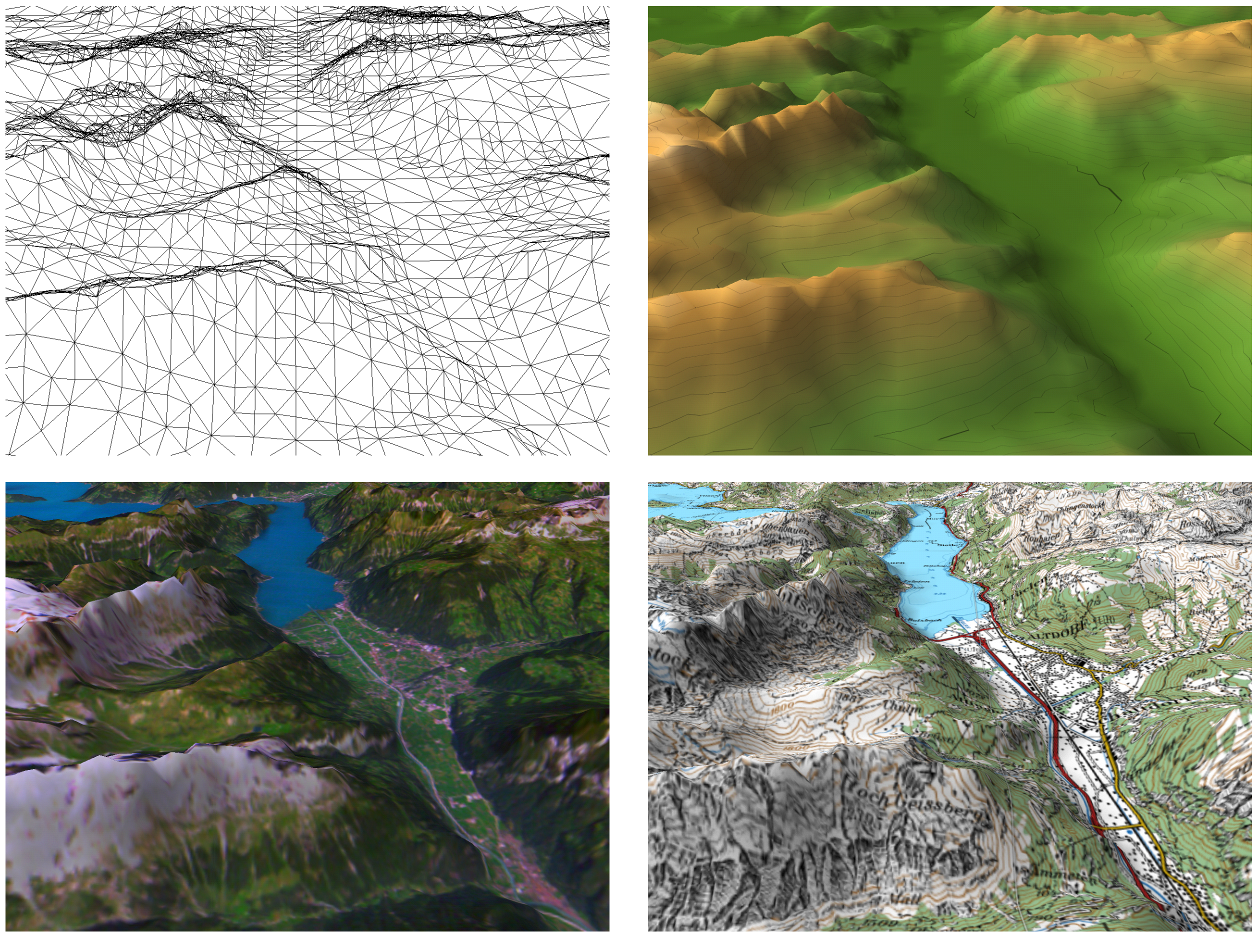


FIGURE 27. Smooth shaded and textured examples of adaptively triangulated terrain using the restricted quadtree triangulation.

4. Terrain visualization systems

In this section we want to briefly study a few additional system and database level aspects such as dynamic scene management, progressive or incremental meshing, data storage and retrieval, and client-server architecture that are important for a large scale real-time terrain visualization system. In such systems the terrain data can easily exceed the main (or even virtual) memory capacity of the workstation used for rendering, and thus terrain data has to be dynamically loaded on-demand from disk. Furthermore, due to the large size of the data set it may also be the case that the data is maintained on a separate data-server and not locally stored on the workstation used for rendering. Most approaches on real-time terrain triangulation and visualization reported in the literature assume the entire terrain data set to be accessed directly in virtual memory and do not actually consider dynamically loading terrain from disk or from a database server. In fact, of the reviewed approaches in this paper only the *VGIS* [KLR+95, LKR+96, LKR+97] and *ViRGIS* [Paj98, Paj98a, Paj98b, POS+98, PW01] systems deal with large terrain data sets stored on disk. A notable exception of not-quadtree based approaches is [Hop98b] where block-based access to large terrain on disk is also considered as outlined below.

VGIS maintains the terrain data on disk partitioned into files of 129x129 vertices each. All levels of the quadtree hierarchy are stored on disk, each as a set of blocks of 129^2 vertices. Due to this multi-level sub-sampling of the height-field, elevation values are duplicated on multiple levels. At run-time, retrieval of the terrain data from disk is strictly constraint to accessing fixed grid resolutions of 129x129 vertex blocks at a time and no further LOD based access is supported. In main memory, an approximation of the entire data set as a global quadtree of the terrain is maintained that can be dynamically updated by loading *macro nodes* of 129x129 vertex blocks from disk. On this macro-level, the decision on which parts of the terrain are loaded at which fixed grid resolution from disk into main memory is based on the distance of the viewpoint from the surface. The height-field data in main memory is then simplified for each frame based on the view-dependent block- and vertex-level error metrics discussed in Section 3.3 to render a continuous adaptive LOD triangulation of the visible terrain.

In contrast, ViRGIS applies a windowing concept that dynamically maintains a fraction of the entire data set in main memory, and uses a tiling of the visible scene to interactively navigate over large terrain. This tiling approach supports efficient dynamic scene management, and the updates from the terrain database can be composed from fixed sized rectangular range-queries, one range-query for every newly visible patch. The dynamic scene manager maintains each terrain patch as a restricted quadtree triangulation that generates adaptive LOD triangle strips for rendering if necessary for each frame. Since it is not efficient to perform updates of the terrain triangulation for each frame, a *deferred cumulative updates* strategy is proposed to reduce the data management costs significantly without significant loss of display quality. The terrain-server, that can be accessed by multiple concurrent ViRGIS rendering-clients, maintains the terrain height-field in a quadtree database that efficiently supports LOD-based rectangular range queries as outlined in Section 3.6 (see also [POS+98, PW01]). Furthermore, the terrain-server also supports LOD-interval queries which allow efficient updates of the client's current ε-approximation to a smaller approximation error δ by incrementally loading more detail for a particular LOD update interval Δ=[ε,δ]. In that case the terrain-server extracts an *incremental error-range* quadtree Q_Δ for a particular scene patch that is added to the client's current quadtree Q_ε to form the updated quadtree $Q_\delta = Q_\varepsilon \oplus Q_\Delta$. Progressive triangulation on the rendering-client is also supported.

In [Hop98b] the entire terrain data set is block-partitioned into quadratic patches on disk. Each patch may be pre-simplified to a minimum tolerance and stored on disk, however, block boundaries are preserved at the finest resolution to guarantee matching triangulations. In main memory, the interior of each quadratic block is adaptively simplified for each frame. Simplification across the highly tessellated block boundaries is performed in a second stage, after block-internal simplification, to reduce artifacts between block regions.

5. Conclusion

The quadtree based triangulation methods described in this paper provide the highest performance in terms of triangulation speed, representation cost, and storage and retrieval efficiency for grid-digital terrain height-field data. This comes at some limited expense of triangle mesh complexity compared to *triangular irregular network* (TIN) approaches. TIN-based multiresolution terrain triangulation methods such as [GH95], [DMP96], [Pup96] and [DP95] are capable of generating same-quality surface approximations at lower triangle counts. However, extraction and incremental refinement of these triangulations are more costly due to the complex mesh connectivity and difficult to achieve in real-time on a per frame basis. Furthermore, the triangulation topology and multiresolution hierarchy is complicated and requires much more storage space than the quadtree based triangulations, making dynamic scene management and disk based retrieval less efficient.

Despite the slightly increased size of the produced triangle meshes compared to TIN approaches, we believe that the quadtree or triangle bintree based multiresolution triangulation methods described in this paper are among the best choices for real-time visualization of very large scale height-field data sets. The various reviewed approaches provide different alternatives in data structures, triangulation algorithms, error metrics, dynamic scene management and rendering methods that can be exploited for an optimized implementation.

The main future work we anticipate in this area is better integration of the multiresolution triangulation algorithms and data structures with the disk resident terrain representation.

Acknowledgements

The digital elevation models (DHM25/RIMINI © 1996) shown in this paper were provided by the Swiss Federal Office of Topography as well as the Landsat™ satellite images (© 1994/1995 ESA/EURIMAGE).

References

[CMS98] P. Cignoni, C. Montani and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[deBD95] Mark de Berg and Katrin Dobrindt. On levels of detail in terrains. In *11th Symposium on Computational Geometry*, pages C26–C27. ACM, 1995.

[DeF89] Leila De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics & Applications*, 9(2):67–78, March 1989.

[DP95] Leila De Floriani and Enrico Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14(4):363–411, 1995.

[DMP96] Leila De Floriani, Paola Marzano, and Enrico Puppo. Multiresolution models for topographic surface description. *The Visual Computer*, 12(7):317–345, August 1996.

[Del34] B. Delaunay. Sur la sphère vide. A la memoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, 7:793–800, 1934. Translated as Bull. Acad. Sci. USSR: Class. Sci. Math. Nat.

[DWS$^+$97] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. In *Proceedings Visualization 97*, pages 81–88. IEEE, Computer Society Press, Los Alamitos, California, 1997.

[EKT97] Williams Evans, David Kirkpatrick, and Greg Townsend. Right-triangulated irregular networks. Technical Report 97-09, Department of Computer Science, University of Arizona, 1997. to appear in Algorithmica.

[GH95] Michael Garland and Paul S. Heckbert. Fast polygonal approximation of terrains and heigt fields. Technical Report cmu-cs-95-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1995.

[Ger99] Thomas Gerstner. Multiresolution visualization and compression of global topographic data. Technical Report 29, Institut für Angewandte Mathematik, Universität Bonn, 1999. to appear in Geoinformatica.

[GRW00] Thomas Gerstner, Martin Rumpf, and Ulrich Weikard. Error indicators for multilevel visualization and computing on nested grids. *Computers & Graphics*, 24(3):363–373, 2000.

[GP00] Thomas Gerstner and Renato Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proceedings Visualization 2000*, pages 259–266. IEEE Computer Society Press, 2000.

[GGS95] Markus H. Gross, Roger Gatti, and Oliver G. Staadt. Fast multiresolution surface meshing. In *Proceedings Visualization 95*, pages 135–142. IEEE Computer Society Press, 1995.

[HK95] D.J. Hebert and HyungJun Kim. Image encoding with triangulation wavelets. In *Proceedings of SPIE*, volume 2569, pages 381–392. SPIE, 1995.

[HG97] Paul S. Heckbert, and Michael Garland. Survey of polygonal surface simplification algorithms. SIGGRAPH 97 Course Notes 25, 1997.

[Hop96] H. Hoppe. Progressive meshes. In *Proceedings SIGGRAPH 96*, pages 99–108. ACM SIGGRAPH, 1996.

[Hop98b] Hugues Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings Visualization 98*, pages 35–42. IEEE, Computer Society Press, Los Alamitos, California, 1998.

[KLR$^+$95] D. Koller, P. Lindstrom, W. Ribarsky, L. F. Hodges, N. Faust, and G. Turner. Virtual GIS: A real-time 3D geographic information system. In *Proceedings Visualization 95*, pages 94–100. IEEE, Computer Society Press, Los Alamitos, California, 1995.

[Lee91] Jay Lee. Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models. *International Journal of Geographic Information Systems*, 5(3):267–285, 1991.

[LKR$^+$96] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner. Real-time, continuous level of detail rendering of height fields. In *Proceedings SIGGRAPH 96*, pages 109–118. ACM SIGGRAPH, 1996.

[LKR$^+$97] Peter Lindstrom, David Koller, William Ribarsky, Larry Hodges, and Nickolas Faust. An Integrated Global GIS and Visual Simulation System. GVU Technical Report 97-0, Georgia Tech Research Institute, 1997.

[Lue01] David Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics & Applications*, 21(3):24–35, May/June 2001.

[Paj98] Renato Pajarola. *Access to large scale Terrain and Image Databases in Geoinformation Systems*. Dissertation No. 12729, Deptartment of Computer Science, ETH Zürich, 1998.

[Paj98a] Renato Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *Proceedings Visualization 98*, pages 19–26 and 515. IEEE Computer Society Press, 1998.

[Paj98b] R. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. Technical Report 292, Dept. of Computer Science, ETH Zürich, 1998. ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/2xx/292.ps.

[POS$^+$98] Renato Pajarola, Thomas Ohler, Peter Stucki, Kornel Szabo, and Peter Widmayer. The alps at your fingertips: Virtual reality and geoinformation systems. In

Proceedings 14th International Conference on Data Engineering, ICDE '98, pages 550–557. IEEE, 1998.

[PW01] Renato Pajarola and Peter Widmayer. Virtual geoexploration: Concepts and design choices. *International Journal of Computational Geometry and Applications*, 11(1):1–14, 2001.

[Pup96] E. Puppo. Variable resolution terrain surfaces. In *Proceedings of the 8th Canadian Conference on Computational Geometry*, pages 202–210, 1996.

[Riv93] M. C. Rivara. A discussion on the triangulation refinement problem. In *Proceedings of the 5th Canadian Conference on Computational Geometry*, pages 42–47, 1993.

[Riv95] M. C. Rivara. A discussion on mixed (longest-side midpoint insertion) delaunay techniques for the triangulation refinement problem. In *Proceedings of the 4th International Meshing Roundtable*, pages 335–346, 1995.

[Sam89a] Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, Reading, Massachusetts, 1989.

[Sam89b] Hanan Samet. *Applications of Spatial Data Structures: computer graphics, image processing, and GIS*. Addison Wesley, Reading, Massachusetts, 1989.

[SS92] R. Sivan and H. Samet. Algorithms for constructing quadtree surface maps. In *Proc. 5th Int. Symposium on Spatial Data Handling*, pages 361–370, August 1992.

[VBH94] Andreas Voigtmann, Ludger Becker, and Klaus Hinrichs. Hierarchical surface representations using constrained delaunay triangulations. In Thomas C. Waugh and Richard G. Healey, editors, *Proc. 6th Int. Symposium on Spatial Data Handling*, volume 2 of *Advances in GIS Research*, pages 848–867. Taylor & Francis, London, 1994.

[VHB87] B. Von Herzen and A. H. Barr. Accurate triangulations of deformed, intersecting surfaces. In *Proceedings SIGGRAPH 87*, number 4 in ACM Journal Computer Graphics, pages 103–110. ACM SIGGRAPH, 1987.