Eurographics Symposium on Geometry Processing (2003), pp. 1–11
L. Kobbelt, P. Schröder, H. Hoppe (Editors)

# Top–Down View–Dependent Terrain Triangulation using the Octagon Metric

Thomas Gerstner

Department for Applied Mathematics, University of Bonn, Wegelerstr. 6, D–53115 Bonn

**Abstract**

*In this paper, we introduce the octagon metric as a useful distance metric for the interactive visualization of large–scale terrain data. Based on recursive bisection triangle meshes, this metric automatically ensures valid distance–dependent triangulations without cracks or T–junctions. We will show how the octagon metric can be used for view–dependent refinement at little computational cost and with no additional storage requirements. It can easily be combined with a suitable geometric error metric to extract and render adaptive view–dependent terrain meshes in an output–sensitive way. We show the performance of the whole system, which is straightforward to implement, and its advantages over previous approaches in several examples.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling: Curve, surface, solid, and object representations

## 1. Introduction

In the last years, many algorithms with different strengths and weaknesses have been proposed for level–of–detail rendering of large terrains. Soon after the first successful solutions, the need for output–sensitive algorithms, that is methods which scale with the number of output triangles, became apparent. Thereby, the rendered triangles should represent the visible part of the terrain as closely as possible. It turned out that the following three components are crucial in order to achieve this goal:

- **Geometric adaptivity:** The terrain should be represented by triangles of varying size. Portions of the terrain which are smooth can be represented by larger triangles, while smaller triangles should be used in rougher areas.
- **Distance–dependent refinement:** Parts of the terrain close to the viewer should be more detailed as parts far away. The detail variation typically depends on the projected screen size of the triangles which decreases with increasing distance to the viewer.
- **Visibility culling:** Only the terrain which is visible should be rendered and, even more importantly, traversed in memory. Here, we consider only the view culling part which disregards all triangles outside the current viewport. Occlusion culling which removes those triangles which are hidden by closer portions of the terrain is required only for very low flyovers.

In this paper, we address these three components by essentially the same method. Based on recursive triangle bisection, a binary tree of triangles is constructed. In a top–down traversal of the binary tree, triangles are refined locally if a suitable metric is above a user–defined (fidelity) threshold, otherwise they are drawn. If the metric fulfills a certain monotonicity property (sometimes called the saturation condition), the resulting triangle meshes are always valid and do not contain so–called hanging nodes or T–junctions which lead to cracks in the terrain surface.

The metric we use is a combination of three parts which reflect the three above components geometric adaptivity, distance–dependence and view culling. Each part is an own metric which independently fulfills the saturation condition. We will show that a proper combination of these single metrics will lead to a saturated overall metric.

The construction of monotonic metrics for adaptive geometric refinement is well–known, see e.g. [8, 7, 18, 19]. For distance–dependent refinement and view culling we will introduce a new metric, the octagon metric. The octagon metric has similar properties as the nested sphere hierarchy [2, 14, 15], which basically achieves the same goal. But, of all the distance metrics which fulfil the saturation condition, the octagon metric is the tightest possible metric. Thus, the re-

sulting triangle meshes consist of the smallest number of triangles which are necessary to define a valid triangular mesh.

In addition, we will show how the metric can be efficiently computed on–the–fly and, therefore, does not require any valuable memory or costly preprocessing. Furthermore, we will show that the different refinement criteria are easy to control and to adjust to new situations. This system provides a uniform and elegant framework for the generation of view–dependent terrain triangulations. Moreover, the whole approach is modular and further important algorithmic terrain visualization features like geomorphing or triangle strip generation are independent of the employed metric and can be applied just like in previous publications.

After the following short discussion of related work we recapitulate the top–down construction of adaptive triangle meshes based on the saturation condition in Section 3. In Section 4 we take a look at a few variants of geometric error metrics. We consider various distance metrics and the derivation of the octagon metric in Section 5. In Section 6, metrics for view–dependent refinement and view clipping are defined. How the single metrics can be combined is shown in Section 7. We illustrate the performance of the new algorithm with some examples in Section 8. Concluding remarks are made in Section 9.

## 2. Related Work

The vast literature on view–dependent visualization of large terrains can be roughly grouped into those based on regular grids and those using irregular meshes. Among the regular grid algorithms, quadtrees [9, 13, 23] and triangle bintrees [5, 12, 14, 15, 19] are most prominent. Irregular mesh methods are usually based on progressive meshes [10], hierarchical Delaunay triangulations [3, 4], or relocated regular grids [20]. Occlusion culling of hierarchical terrains has been considered e.g. in [16, 24]. An overview of the different concepts and a comparison of the various algorithms can be read in the recent book of Luebke et.al.[17].

In the now following short review we consider only regular grid approaches. Thereby, we distinguish between the different techniques for view–dependent refinement. We will not look at further issues such as triangle stripping, geomorphing, data layout and out–of–core behaviour, which are also addressed in many of the publications in detail.

One of the first algorithms which provide adaptive view–dependent terrain triangulations is that of Lindstrom et.al.[13]. Essentially, the algorithms works bottom–up, merging pairs of triangles until a screen space error tolerance is met. Cracks are avoided by an explicit storage of the vertex dependencies. In order to alleviate the inherent complexity of the bottom–up approach, the method uses a top–down coarse–grained simplification of rectangular terrain blocks beforehand. However, stitching these blocks together requires special care.

The ROAM algorithm of Duchaineau et.al.[5] avoids these problems using a top–down approach. The authors propose dual priority queue for triangle split and merge operations. The algorithm is incremental and adds or removes triangles from a given triangulation as the viewer moves on. They use a screen–based error metric using a hierarchy of bounding volumes.

Röttger et.al.[23] use a restricted quadtree approach and deal with cracks by skipping the center vertex of the higher–resolution edge. They enforce the restriction in a bottom–up traversal of the affected parts of the quadtree. They use an error metric which takes the distance from the viewer and the local roughness of the terrain into consideration.

The work of Pajarola[19] was among the first which use a monotonic geometric error indicator. However, for view–dependent refinement still forced recursive splitting of neighbouring triangles, like in the ROAM algorithm, is done.

Ohlberger and Rumpf[18] also use a monotonic geometric error, and they obtain a local top–down algorithm through an almost nested distance metric. Cracks are avoided by a distance function which compensates for this deficiency. This imposes some restrictions on the derivative of the distance function, though.

A major improvement over the ROAM algorithm was presented by Blow[2]. He precomputes a hierarchy of spherical isosurfaces which contain all vertices within a certain error bound. This way, a top–down algorithm is provided in which a triangle is split if the viewpoint intersects the sphere and triangles are merged if the viewpoint moves outside.

The SOAR algorithm of Lindstrom and Pascucci[14, 15] draws on virtually all previous works on terrain visualization. In particular, they improve on Blow's work with the help of nested bounding spheres. This allows a dynamic error threshold, more general error metrics and avoids potential cracks in the terrain surface generated by Blow's algorithm. The approach adopted in this paper is in many ways similar to this work. It significantly differs in the construction and application of error and distance metrics, though.

In the most recent work, Levenberg[12] improves the graphics I/O performance of the ROAM algorithm by aggregating triangles into blocks. During block construction, however, special care has to be taken in order to ensure that triangulations match at block boundaries.

## 3. Top–Down Terrain Triangulation

In this section we will shortly illustrate the recursive triangle bisection hierarchy, the basic rendering algorithm and the saturation condition on the error metric. The facts in this section are well–known but they are necessary to understand the rest of the paper.

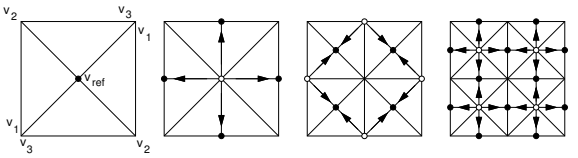


**Figure 1:** *The recursive bisection hierarchy. Refinement vertices are black circles, their respective parents are white circles. The parent–child relationship is indicated by arrows.*

### 3.1. Recursive Triangle Bisection

Let us assume that the terrain is given on a square regular grid with $n = 2^k + 1$ elevation values $h(x_i, y_j)$, $0 \le i, j < n$ in each direction. The initial triangulation consists of two isosceles triangles $\Delta(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ with a right angle at $\mathbf{v}_2$ which cover the square. From the initial triangulation finer triangulations are recursively constructed by splitting all triangles in two. To this end, the midpoint of the longest edge is chosen as a new vertex $\mathbf{v}_{ref}(T)$, called the refinement vertex, and the two new triangles are given by $C_1(T) = \Delta(\mathbf{v}_2, \mathbf{v}_{ref}, \mathbf{v}_1)$ and $C_2(T) = \Delta(\mathbf{v}_3, \mathbf{v}_{ref}, \mathbf{v}_2)$ (see Figure 1).

By this refinement procedure a binary tree hierarchy is inferred on the triangles. All refinement vertices are shared by two triangles except on the boundary. The vertices itself have no tree structure but form a directed acyclic graph. Each refinement vertex (except on the boundary) has two parent vertices which indicate the two vertices which have to be present in the mesh before this vertex can be introduced. In turn, each parent vertex except on the boundary has four children. This parent–child relationship is depicted in Figure 1.

Now, an adaptive triangulation is simply defined by the selection of an appropriate subgraph of the whole graph. This way, hanging nodes, which occur if two triangles sharing a refinement vertex are not refined conformingly, are avoided. Hanging nodes are undesirable because they will lead to cracks in the terrain since the surface defined by the triangulation is no longer continuous.

### 3.2. The Target Rendering Algorithm

Before we come to this selection in practice, we want to illustrate the final structure of the triangulation algorithm. The goal is a top–down, depth–first traversal of the binary tree of triangles which can be stated in pseudo–code as follows:

```
visit(Coord v1, v2, v3; Level l; Thresh eps) {
    Coord vref=(v1+v3)/2;
    if (mu(T) < eps)
        render_triangle(v1, v2, v3);
    else {
        visit(v2, vref, v1, l+1);
        visit(v3, vref, v2, l+1);
    }
}
```

For each triangle during the traversal, a yet–to–be specified metric $\mu(T)$ is compared against a given threshold $\varepsilon$. If this metric exceeds the threshold then the triangle is refined, else it is drawn. This way, the rendering algorithm is completely local since at no point neighbouring triangles are accessed. The remaining questions are now: how can this metric be defined and how is the threshold selected?

### 3.3. The Saturation Condition

We will now consider a sufficient condition for the metric $\mu$, the so–called saturation condition, such that for any threshold $\varepsilon$ the resulting triangulation contains no hanging nodes. To this end, the metric $\mu(T)$ for a triangle $T$ is not assigned to the triangle itself but to its refinement vertex $\mathbf{v}_{ref}(T)$, i.e.

$$\mu(T) = \mu(\mathbf{v}_{ref}(T)).$$

This way, the two triangles sharing the refinement vertex are assigned the same metric and are thus refined in unison. However, it can still happen that a descendant of one of the two triangles is refined at their common boundary while the other is not. Therefore, the saturation condition [18] states that the metric of every triangle has to be at least as large as the metric of all its children, i.e.

$$\mu(T) \ge \max\{\mu(C_1(T)), \mu(C_2(T))\}.$$

In the next chapters we will see how metrics which satisfy the saturation condition can be constructed in practice.

## 4. Geometric Refinement Metrics

The three components geometric adaptivity, distance–dependent refinement and view culling of an interactive terrain visualization system can be addressed by three different metrics. In this section we will take a look at the computation of geometric refinement metrics. There is a large variety of methods for hierarchical geometric distance measurement. For an overview and comparison, see e.g. [8]. We will shortly illustrate a few useful variants which are needed later here.

### 4.1. Vertical Distance Metric

When a refinement vertex is added to a given triangulation, the corresponding piecewise linear surface changes locally. The added (or subtracted) portion has the shape of a four–sided pyramid. The height of this pyramid is the vertical deviation between the original and the refined surfaces and serves as a well–known geometric distance metric (often called one–level lookahead error). This metric has been used in many approaches such as in [7, 14, 15, 18, 19].

The vertical distance metric $\mu_{geo}(T)$ can be computed locally on the triangle $T = \Delta(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ using the elevation values on the refinement edge of $T$ by the formula

$$\mu_{geo}(T) = \left| h(\mathbf{v}_{ref}) - \frac{1}{2}(h(\mathbf{v}_1) + h(\mathbf{v}_3)) \right|.$$

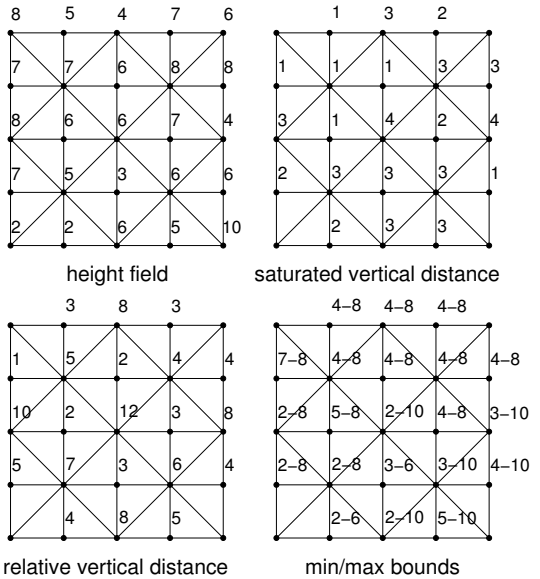


**Figure 2:** *A few examples for the construction of saturated geometric distance metrics on a* $5 \times 5$ *elevation grid.*

Note that in this definition both triangles sharing the refinement vertex $\mathbf{v}_{ref}$ are automatically assigned the same value.

The vertical distance $\mu_{geo}$ could be efficiently computed on–the–fly, but it does, in general, not fulfil the saturation condition. However, a minimally saturated metric $\bar{\mu}_{geo}$ can be constructed in a precomputation step in a level–wise bottom–up traversal of the binary tree by the recursive formula

$$\bar{\mu}_{geo}(T) = \max\{\, \mu_{geo}(T), \bar{\mu}_{geo}(C_1(T)), \bar{\mu}_{geo}(C_2(T)) \,\},$$

with $\bar{\mu}_{geo}(T) = \mu_{geo}(T)$ on the finest level. Note that a depth–first traversal would not be sufficient. The values of the metric $\bar{\mu}_{geo}(T) = \bar{\mu}_{geo}(\mathbf{v}_{ref}(T))$ can be stored at the refinement vertex positions in a square array (only the four corners of the square are not needed).

### 4.2. Relative Vertical Distance Metric with Bounds

The vertical distance metric $\bar{\mu}_{geo}$ works very well in practice as an indicator of terrain roughness. However, it has a significant drawback: it does not provide robust error bounds. From the values of $\mu_{geo}(T)$ or $\bar{\mu}_{geo}(T)$ and the current triangle $T$, it is not possible to derive upper and lower bounds on the variation of the terrain inside $T$. Such bounds are necessary for view–dependent refinement based on a screen–based error and are useful for a variety of other uses such as clipping, occlusion culling, line–of–sight and horizon computation, collision detection, and the extraction of isolines.

We will therefore construct a metric $\tilde{\mu}_{geo}(T)$ which allows upper and lower bounds on the variation of the terrain inside a triangle $T$ relative to the elevation value of the refinement vertex $h(\mathbf{v}_{ref}(T))$. Note that this metric corresponds to $+-$ type error indicators in [8] and to the vertical part of the nested sphere bounds in [14, 15].

On the finest level, four triangles share the vertex $\mathbf{v}_2$ which is the common refinement vertex of their two parent triangles (on the boundary, only two triangles are considered). The metric $\tilde{\mu}_{geo}$ is defined on the finest level as the maximum of

$$\max\{|h(\mathbf{v}_1) - h(\mathbf{v}_2)|, |h(\mathbf{v}_3) - h(\mathbf{v}_2)|\}$$

for all triangles incident on $\mathbf{v}_2$. Now, again in a level–wise bottom–up traversal a saturated metric can be computed by

$$\tilde{\mu}_{geo} = \max\{|h(\mathbf{v}_{ref}(T)) - h(\mathbf{v}_{ref}(C_1(T)))| + \tilde{\mu}_{geo}(C_1(T)),$$
$$|h(\mathbf{v}_{ref}(T)) - h(\mathbf{v}_{ref}(C_2(T)))| + \tilde{\mu}_{geo}(C_2(T))\}.$$

This way, we know that for any triangle $T$ in the hierarchical triangulation, the terrain inside the triangle is bounded by $h(\mathbf{v}_{ref}(T)) \pm \tilde{\mu}_{geo}(T)$.

### 4.3. Vertical Min/Max Bounds

Th relative vertical distance metric $\tilde{\mu}$ gives bounds on the terrain, but not very tight ones. A third way is to compute the explicit min/max bounds on the terrain by recursively setting

$$max(T) = \max\{h(\mathbf{v}_{ref}(T)), max(C_1(T)), max(C_2(T))\},$$
$$min(T) = \min\{h(\mathbf{v}_{ref}(T)), min(C_1(T)), min(C_2(T))\},$$

where $max(T) = max\{h(\mathbf{v}_1), h(\mathbf{v}_2), h(\mathbf{v}_3)\}$ and $min(T) = min\{h(\mathbf{v}_1), h(\mathbf{v}_2), h(\mathbf{v}_3)\}$ on the finest level. Then,

$$\hat{\mu}_{geo}(T) = max(T) - min(T)$$

can serve as a saturated geometric distance metric with tight bounds.

### 4.4. Comparison

In Figure 2 we give a small example of geometric metrics on a $5 \times 5$ terrain grid. We compare the vertical distance metric with saturation $\bar{\mu}$ to the relative vertical distance $\tilde{\mu}$ and the vertical min/max bounds. We see that the smallest values are computed for the vertical distance metric $\bar{\mu}$, but in this case we have no robust error bounds. The relative metric $\tilde{\mu}$ usually overestimates the real min/max bounds, which give the optimum result.

However, in terms of memory requirements the min/max bounds require two integer values while the other two metrics need only one integer. In case, memory is a premium and bounds are required, the relative distance metric $\tilde{\mu}$ perform best.

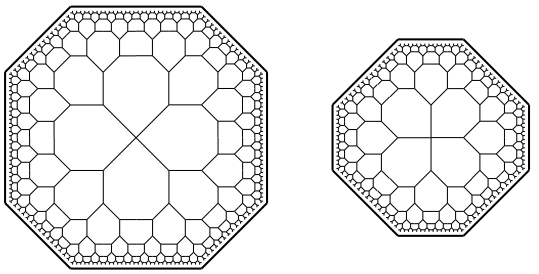

**Figure 3:** *The set of hierarchical descendants of a given point in the hierarchy has the shape of an octagon. Shown are the octagons for odd (left) and even (right) levels. Only those descendants with the largest distance from the center are displayed.*

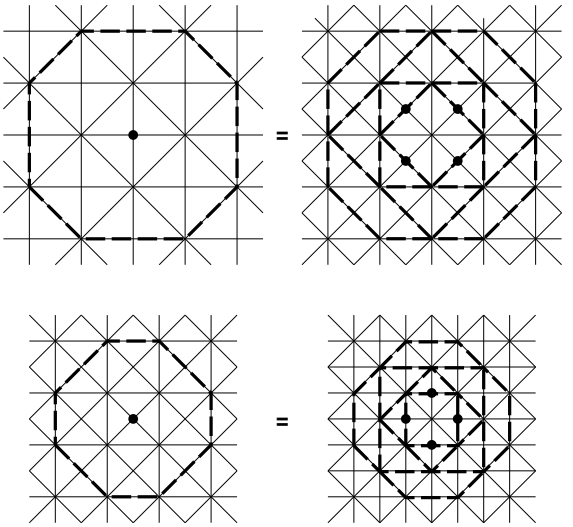

**Figure 4:** *The octagons have refinement relations as depicted above for even (top) and odd (bottom) levels. Each octagon is the union of the octagons of its four children.*

## 5. Distance Metrics

While saturated geometric metrics are known for several years now, saturated metrics for distance–dependent refinement were found only recently (see [2, 14]). The main requirements are horizontal bounds on the triangulation, as opposed to vertical bounds on the terrain which were considered in the previous section. For simplicity, we consider in this section only the two–dimensional case. The 3–D case is treated in the following section. We will describe a few distance metrics based on bounding shapes and introduce the octagon distance.

### 5.1. Bounding Shapes

Let $B(T)$ be a two–dimensional nested bounding shape for a triangle $T$, i.e.

$$B(T) \supseteq \cup\{B(C_1(T)), B(C_2(T))\}.$$

with $T \subseteq B(T)$. Popular examples for such bounding shapes are bounding rectangles and bounding circles [2]. Then, the (Euclidean) distance from a certain point $\mathbf{p}$ in the plane (consider e.g. the projection of the viewpoint) to the bounding shape $d(\mathbf{p}, B)$ is a distance metric which, per definition, satisfies an inverse saturation condition, that is,

$$d(\mathbf{p}, B(T)) \geq \max\{d(\mathbf{p}, B(C_1(T))), d(\mathbf{p}, B(C_2(T)))\}.$$

Therefore, metrics such as $\mu(T) = 1/d(\mathbf{p}, B(T))$ or $\mu(T) = d_{max} - d(\mathbf{p}, B(T))$ satisfy the (original) saturation condition and can be used as a refinement criterion.

Bounding circles are especially useful here, since $d(\mathbf{p}, B)$ can be computed quickly by $r - ||\mathbf{p}, \mathbf{c}||$, where $\mathbf{c}$ is the center and $r$ the radius of the sphere $B(T)$. The distance to a rectangular bounding shape is more involved since a case table has to be used which differentiates to which side or which corner of the rectangle the distance has to be measured depending on the location of $\mathbf{p}$ relative to $B(T)$.

In any case, for regular triangle meshes, the diameters of bounding circles or bounding rectangles do not have to be stored for every triangle but follow easily computable rules.

### 5.2. The Set of Hierarchical Descendants

In a similar way that the relative vertical distance bounds overestimate the vertical min/max bounds of the terrain, the horizontal bounding circles or bounding rectangles overestimate the horizontal extent. We will now try to find the minimum shape which is required for horizontal bounds.

Let us therefore consider the set of hierarchical descendants of a given refinement vertex, i.e. all vertices on larger levels which have this refinement vertex as an ancestor according to the parent–child relationship of Figure 1. As already observed in [1], the set of descendants has roughly the shape of an octagon. The limit shape is indeed an isothetic octagon (i.e. an octagon with sides parallel to the horizontal, vertical and diagonal axes). In fact, two octagon types arise with two different shapes whose sizes depend on the level of the refinement vertex (actually, octagons corresponding to vertices near the boundary are clipped at the boundary, but for simplicity we ignore this fact for now).

Figure 3 shows the shape of these octagons for even and odd levels. In even levels, the set of descendants is generated by starting in the middle and moving one square in diagonal direction, then one square in horizontal or vertical direction, then half a square diagonally, then half a square horizontally or vertically and so on. For odd levels, the first diagonal step is omitted. Let the length of the first step, which is the distance from the refinement vertex (in the middle) to one of its four sons, be denoted by $s$. Then, in even levels the limit

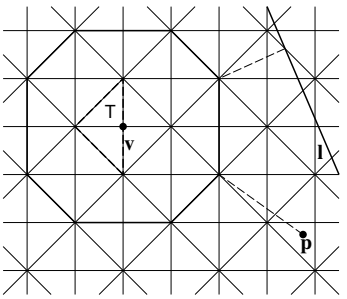


**Figure 5:** *The octagon distance from a point* **p** *to a triangle* $T$ *is the distance of* **p** *to its corresponding octagon* $o_l(T)$. *Similarly, the octagon distance from a line* **l** *to the triangle is the distance between* **l** *and its octagon* $o_l(T)$.

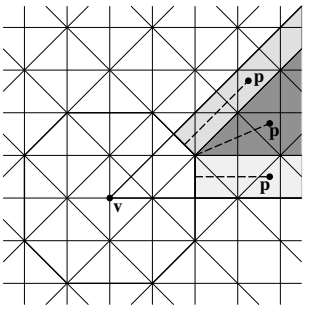


**Figure 6:** *The octagon distance has to be computed to different sides or vertices of the octagon depending on the location of the point* **p**. *Using the eightfold symmetry of octagons, three cases remain.*

octagon has side lengths $2s$ horizontally and vertically and $2\sqrt{2}s$ diagonally. In odd levels the side lengths are reversed.

Since the octagon is the limit shape of all descendants of a given vertex, which themselves have octagonal limit shapes, the octagons must have refinement relations. In Figure 4 we show these refinement relations. Each octagon corresponding to a refinement vertex is the overlapping union of the octagons of the four sons of this vertex. This is the important nesting property which is required for the saturation condition.

### 5.3. The Octagon Distance

Before we are able to define a distance metric based on these octagons, we need to define them more formally. Let us define the octagon around a refinement vertex $\mathbf{v}$ of level $l$ as

$$o_l(\mathbf{v}) = \begin{cases} \{\mathbf{x} : ||\mathbf{v}-\mathbf{x}||_\infty \leq \frac{3}{2}\cdot 2^{-l} \text{ and } ||\mathbf{v}-\mathbf{x}||_1 \leq 2\cdot 2^{-l}\} \\ \qquad \text{if } l \text{ is even,} \\ \{\mathbf{x} : ||\mathbf{v}-\mathbf{x}||_\infty \leq 2\cdot 2^{-l} \text{ and } ||\mathbf{v}-\mathbf{x}||_1 \leq 3\cdot 2^{-l}\} \\ \qquad \text{if } l \text{ is odd.} \end{cases}$$

where $\mathbf{x} = (x_1, x_2)$, $||\mathbf{x}||_\infty = \max\{x_1, x_2\}$ and $||\mathbf{x}||_1 = x_1 + x_2$. In other words, the octagon $o_l(\mathbf{v})$ consists of all the points which have a distance from $\mathbf{v}$ in horizontal, vertical and diagonal direction which is smaller than certain level–dependent constants. The extent is $3s$ in the four directions following from the refinement vertex to its four sons and $2\sqrt{2}s$ in the four rotated directions.

Now, like in the Section 5.1, the octagon distance of an arbitrary point $\mathbf{p}$ in the plane to a triangle $T$ is defined as the Euclidean distance to its corresponding octagon $o_l(T) = o_l(\mathbf{v}_{ref}(T))$, i.e.

$$d_{oct}(\mathbf{p}, T) = d(\mathbf{p}, o_l(T))$$

(see Figure 5). If $\mathbf{p}$ is in the interior of $o_l$, we will set the octagon distance to zero. Alternatively, in the interior the distance could be defined as the negative distance to the boundary of the octagon.

The octagon distance $d_{oct}$ can be efficiently computed using the eightfold mirror symmetry of the octagons. By symmetry it is e.g. possible to mirror the vertex $\mathbf{p}$ until it lies to the top right of $\mathbf{v}_{ref}(T)$ and the $x_1$–coordinate of $\mathbf{p}$ is larger than its $x_2$–coordinate. Then, only three cases remain. In the first case, the distance is measured to the left horizontal side of the octagon, in the second case, it is measured to its top left diagonal side and in the third case, it is measured to the lower of its two top–left vertices (see Figure 6). In practice, the octagon distance can be computed about as fast as e.g. the distance of a point to a bounding circle.

An example for the computation of octagon distances on a $5\times 5$ grid is shown in Figure 7. There, one can see that the octagon distance from a point $\mathbf{p}$ to a given vertex is always smaller than the octagon distances to its descendants.

The octagon distance is optimal in the sense that it is the smallest distance with bounds which satisfies the inverse saturation condition. This can be proven rigorously by showing that an octagon encompasses all triangles on the finest level whose refinement vertices are descendants of the vertex associated with the octagon. Furthermore, it is easy to see that all triangles on the finest level outside the octagon are no descendants of this vertex and thus the metric is as tight as possible.

Note that the nested circle hierarchy is much less tight. The circle diameter is about 50% larger than the diameter of the corresponding octagons (except on the finest levels the diameters compare approximately like $2+\sqrt{2}$ to $\sqrt{5}$).

### 5.4. Octagon Distance with Maximum Level

Since the octagons are the limit shapes for the set of hierarchical descendants, they are independent of the maximum level of the hierarchical triangulation. Thus, the octagons can

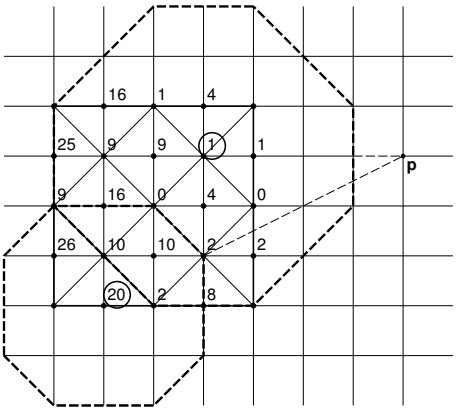


**Figure 7:** *In this example we show the squared octagon distances (in order to avoid square roots) from the point* **p** *to the refinement vertices of a* $5 \times 5$ *grid. The distance between the grid points is* 1*. The corresponding octagons of the two encircled vertices are shown dashed.*

even be used if data sets at higher resolution are inserted dynamically in an out–of–core system (see [7]) or if fractal interpolation or subdivision is used to refine the terrain beyond the resolution of the original data.

If the maximum level of the terrain is fixed, the size of the octagons can be reduced further somewhat (although the change is significant only on higher levels). The reduced octagon sizes in horizontal, vertical, and diagonal direction are obtained by a multiplication with a factor of $1-2^{-(l_{max}-l)/2}$.

### 5.5. Octagon Distance for Lines

The octagon distance with respect to points can be used for distance–dependent refinement, as we will see in the next section. For view clipping, the octagon distance with respect to clipping planes is required. This corresponds in two dimensions to a distance metric with respect to lines.

The octagon distance of a line $\mathbf{l}: ax_1+bx_2=c$ to a triangle $T$ is simply defined as

$$d_{oct}(\mathbf{l}, o_l(T)) = \min_{\mathbf{x}\in\mathbf{l}} d_{oct}(\mathbf{x}, o_l(T)),$$

i.e. it is the distance of the line to the corresponding octagon of $T$ (see Figure 5). Obviously, this distance metric also satisfies an inverse saturation condition and can thus be used as a refinement criterion.

The octagon distance with respect to lines can be computed as efficiently as the octagon distance with respect to points. The only difference is that the different cases depend on the normal of $\mathbf{l}$ and that only the vertices of the octagon have to be considered.

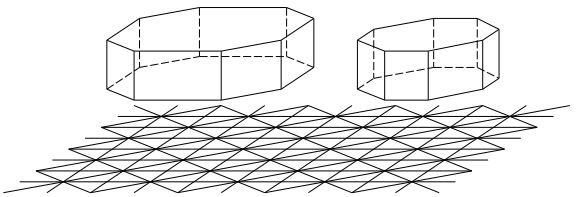

**Figure 8:** *The two types of octohedra have as base an octagon and as height the bounds of the corresponding geometric distance.*

## 6. View–Dependent Refinement

In the previous section we have only considered two–dimensional distance metrics. For view–dependent refinement we will now consider their natural extension to three dimensions. We will define the octohedron and derive corresponding distance and view culling metrics.

### 6.1. The Octohedron

We will now extend the octagons in $x_3$ direction using upper and lower bounds for the elevation values inside the triangles described in Section 4. This leads to optimally tight bounding shapes for terrain. We define the 3–D octohedron $O_l(T)$ (not to be mistaken with oct*a*hedron) by

$$O_l(T) = \{\mathbf{x} : (x_1, x_2) \in o_l(T), |x_3 - h(\mathbf{v}_{ref}(T))| \le \mu_{geo}(T)\},$$

where $\mu_{geo}$ is one of the geometric distance metrics with bounds. The resulting octohedra are depicted in Figure 8.

The octagon distance $d_{oct}$ of a point or a plane in three dimensions to a triangle $T$ are defined analogously to the 2–D case as the distance to its corresponding octohedron. For the computation of the octagon distance just one more case has to be added which discriminates if the point is above or below the octohedron. In this case, the octagon distance is measured to the corresponding (top or bottom) face of the octohedron. Let us remark here, that the octohedra give much tighter bounds as bounding spheres [14, 15]. Especially if the terrain is rough, the diameter of bounding spheres will quickly become very large.

### 6.2. The Distance Metric

The main idea behind view–dependent visualization is to try to achieve an equal distribution of the approximation error after projection to the screen. Here, the octohedron can serve as a footprint of the projection. Since projected triangle area decreases quadratically with the distance to the viewer we define a saturated distance metric by

$$\mu_{dist}(T) = \frac{1}{d_{oct}(\mathbf{p}, O_l(T))^2}.$$

Let us remark here, that this metric can easily be replaced by different distance metrics for other applications.

### 6.3. The View Culling Metric

Hierarchical view culling is surprisingly the most time–consuming algorithmic part of terrain visualization. Nevertheless, especially for large terrains it is the most important part since it reduces the number of visited and drawn triangles most immensely.

For view culling we have to consider the six clipping planes, (respectively their normals). Now, a saturated culling metric is simply given by the characteristic function of the octagon distances of the clipping planes $\mathbf{P}_i : ax_1 + bx_2 + cx_3 = c$, $1 \leq i \leq 6$,

$$\mu_{cull}(T) = \begin{cases} 1 & \text{if} \quad d_{oct}(\mathbf{P}_i, O_l(T)) \leq 0 \text{ for all } \mathbf{P}_i, \\ 0 & \text{else.} \end{cases}$$

In this way, refinement takes place as long as the corresponding octagon intersects the viewing area. If the octagon is completely outside the area, refinement is not necessary and the current triangle can be drawn or entirely skipped. Note that the number of triangles outside the viewing area is usually small compared to the number of finally drawn triangles (because they get large quickly with increasing distance to the viewing area). Drawing triangles outside the area can be advantageous if triangle strips are used for rendering (see [7, 14, 15, 19]). Outside triangles are then discarded by the graphics engine after clipping.

Note that not every octohedron has to be checked against all clipping planes. If a parent octohedron is completely on the inside of a clipping plane, clipping against this plane is not necessary for its children any more (see [15]).

## 7. Metric Combination

Now, we will show how the three metrics for geometric adaptivity, distance–dependent refinement and view culling can be combined into a single metric which also fulfills the saturation condition. In the following, we will illustrate how saturated metrics can be in principle combined, we select a particular useful combination and address the question of threshold selection.

### 7.1. A Little Metric Algebra

In principle, metrics could be combined at will, but often the saturation condition will not be fulfilled any more for the combined metric. We will now show a few possible valid combinations. Let $\mu_1$ and $\mu_2$ be two saturated metrics, then the following combined metrics $\mu$ are also saturated:

- Addition: $\mu = \mu_1 + \mu_2$
- Multiplication: $\mu = \mu_1 \cdot \mu_2$
- Maximum: $\mu = \max\{\mu_1, \mu_2\}$
- Minimum: $\mu = \min\{\mu_1, \mu_2\}$

Note, that the difference or the quotient of two metrics is in general not saturated. However, if $f(x)$ is a (not necessarily strictly) monotonically increasing function in $x$, and $\mu$ a saturated metric then $f(\mu)$ is also saturated. Since the constant metric $\mu = c$ is trivially saturated, multiplication or addition of a constant to a metric also does not change the saturation condition.

The proofs of these properties of saturated metrics are simple adaptions from the theory of monotonic functions (e.g. the sum of two monotonic functions is again a monotonic function). The large variety of possible combinations shows the great flexibility in the construction of saturated metrics and allows easy adjustment to new situations.

### 7.2. The Combined Metric

Now we are finally able to define the overall view–dependent metric. Let us first observe that the single metrics already have the correct asymptotic behaviour. The geometric metric increases linearly with the roughness of the terrain and the distance–dependent metric decreases quadratically with the distance. The culling metric is just an indicator function. Therefore, we only have to multiply the three metrics in order to get a combined metric with also has the correct asymptotic behaviour:

$$\mu(T) = \mu_{geo}(T) \cdot \mu_{dist}(T) \cdot \mu_{cull}(T).$$

At this point, it is not necessary to weigh the three metrics differently, since any constants can be thrown over to the right–hand side (the threshold). If only multiplication is used, threshold selection can be performed totally independent of way the metrics are combined.

Let us note here that this combined metric has in fact a similar structure (although not at first sight) as the one used by Lindstrom and Pascucci in [14, 15]. It is, however, much easier to modify. For example, parts of the terrain (e.g. airports) can be highlighted through inclusion of a new saturated metric which measures the distance to the highlighted part. Terrain tiles of different resolution can be seamlessly merged by a saturated metric which measures the distance to the higher resolution part. Metrics could be turned on and off dynamically.

### 7.3. The Threshold

Let us recall that the target rendering algorithm refines a triangle if the overall metric is larger than a given threshold $\varepsilon$, otherwise it is drawn. There are several ways how to select this threshold, depending on the goal of the visualization system.

If a stable frame rate is paramount, the threshold has to be adapted from frame to frame. If the number of drawn triangles increases too strongly in between frames, the threshold is increased as well. If the number decreases, the threshold can be also decreased (yielding a higher image quality). Since typically the number of drawn triangles decreases

quadratically with the threshold increment, a good adjustment guess for the threshold is simply the square root of the triangle count difference.

In other applications, image quality is most important. For example, it might be desirable that the output image of a view–dependent terrain renderer should be identical to the full terrain. In this case, a screen–space error tolerance is computed against which the error threshold is compared, see e.g. [13, 14, 15]. Due to the multitude of approximations involved in error measurement and projection, the screen–space error is usually either not met exactly or has to be overestimated. The octagon metric, however, allows tight screen–space error bounds.

## 8. Examples

We will now illustrate the different metrics and their interplay with a few examples. We use a $4096 \times 4096$ terrain grid which is part of the gtopo30 data set (courtesy by the US Geological Survey) covering the northern part of central Europe. In this data set, ocean areas are marked (by -9999). Triangles are not drawn, if at least one of the height values of its three vertices is in the ocean (this shows the coastlines).

Let us first show the behaviour of three single metrics in two dimensions. In Figure 9 we show triangulations using only the (normalized) geometric distance metric $\overline{\mu}_{geo}$ for thresholds of 1, 0.1, 0.01 and 0.001. For this example, we modified the metric slightly by increasing the threshold at coastlines artificially prior to saturation. This enhances the coastlines significantly since height differences in coastal areas is usually quite small yielding small geometric errors. We see that this way at coastlines and in mountainous areas (in central Germany) triangles are smaller than in flat areas (Denmark or the Netherlands).

In Figure 10 we illustrate the behaviour of the distance metric $\mu_{dist}$. Here, again, the metric is normalized and the thresholds are 1, 0.1, 0.01 and 0.001 from left to right. It is visible that the triangle size increases continuously with increasing distance from the viewpoint. Note that for the leftmost image, the number of triangles is the minimal number of triangles required to refine at the viewpoint up to the finest level.

The culling metric is shown in Figure 11 for three clip lines. Here, the threshold is 0.5 in all images, but the maximum level increases by two from left to right from 8 to 14. We see that the triangles outside the triangular clipping area does not change with the maximum level but stays exactly the same. This tells us the octagon metric used for culling is tight and thus optimal.

In Figure 12 we show the combined view–dependent metric again for thresholds of 1, 0.1, 0.01 and 0.001. The result is an overlay of the three metrics and shows exactly the desired behaviour. Triangles are large in smooth areas, increase slowly with the distance to the viewpoint and increase sharply outside the viewing area.

In three dimensions (Figure 14 and 13) we compare the vertical distance metric together with the nested sphere hierarchy (like as they are used in the SOAR algorithm, left) with the octagon distance together with relative vertical distance bounds (middle) and vertical min/max bounds (right). Shown is a view from the south with the Upper Rhine Valley (a great wine–growing area) in the foreground. The terrain is textured with a simple elevation–dependent colormap.

Figure 13 illustrates the view–dependent triangle meshes generated by the three algorithms. In this example, the triangle counts are 81.934, 77.029 and 74.323 from left to right. For the same error threshold the octagon metric together with the min/max bounds yield the best approximations.

In Figure 14 we show the view clipping performance. Here the clip area is indicated by the square in the middle of the images. From left to right, the number of rendered triangles is 34.969, 22.707 and 21.035. We see that the nested sphere hierarchy overestimates the vertical extent of the terrain for clipping and many unnecessary triangles are generated especially above the clip area, but also to the other areas outside the clip area. The min/max bounds yield the smallest triangle count with the relative vertical distance bounds performing only slightly worse.

## 9. Concluding Remarks

In this paper, we have illustrated the construction, combination and use of metrics which satisfy the saturation condition for view–dependent refinement of large terrain meshes. We introduced the octagon metric as a universally applicable and optimal distance metric and showed its application for distance–dependent refinement and view culling. Finally, we have shown how metrics can be combined in order to define new metrics which have the same properties as the originals.

Let us note again, that the whole system is very general and can easily modified and adapted to incorporate different behavior. In addition, advanced features such as triangle stripping, geomorphing, data compression and out–of–core management can be implemented identically to previous publications (see of the following references, most of them deal with some these features).

Finally, the octagon metric can be directly extended to enable view–dependent visualization of volume data based on recursive bisection tetrahedral meshes. We will show the corresponding construction (e.g. for multiresolution isosurface extraction) in a companion paper.

**Acknowledgment**

The author likes to thank Peter Lindstrom and Valerio Pascucci for the perusal of their SOAR code, with which the comparison runs have been performed.

## References


1. L. Balmelli, S. Ayer, and M. Vetterli. Efficient algorithms for embedded rendering of terrain models. In *Proceeding of the IEEE International Conference on Image Processing (ICIP)*, pages 914–918, 1998. 5
2. J. Blow. Terrain rendering at high levels of detail. In *Proceedings 2000 Game Developers Conference*, San Jose, 2000. 1, 2, 5
3. P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer*, 13(5), 1997. 2
4. L. DeFloriani, P. Magillo, and E. Puppo. Building and traversing a surface at variable resolution. In *Proceedings IEEE Visualization 97*, pages 103–110. IEEE Computer Society Press, 1997. 2
5. M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, and M. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In *Proceedings IEEE Visualization 97*, pages 81–88. IEEE Computer Society Press, 1997. 2
6. W. Evans, D. Kirkpatrick, and G. Townsend. Right-triangulated irregular networks. *Algorithmica*, 30(2):264–286, 2001.
7. T. Gerstner. Multiresolution visualization and compression of global topographic data. *GeoInformatica*, 7(1):7–32, 2003. 1, 3, 7, 8
8. T. Gerstner, M. Rumpf, and U. Weikard. Error indicators for multilevel visualization and computing on nested grids. *Computers & Graphics*, 24(3):363–373, 2000. 1, 3, 4
9. M. Gross, R. Gatti, and O. Staadt. Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):130–143, 1996. 2
10. H. 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, 1998. 2
11. R. Klein, D. Cohen-Or, and T. Hüttner. Incremental view–dependent multiresolution triangulation of terrain. *Journal of Visualization and Computer Animation*, 9:129–143, 1998.
12. J. Levenberg. Fast view–dependent level–of–detail rendering using cached geometry. In *Proceedings Visualization 02*, pages 259–266. IEEE Computer Society Press, 2002. 2
13. P. Lindstrom, D. Koller, W. Ribarsky, H. L.F., N. Faust, and G. Turner. Real–time, continuous level of detail rendering of height fields. *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 109–118, 1996. 2, 9
14. P. Lindstrom and V. Pascucci. Visualization of large terrains made easy. In *Proceedings Visualization 01*, pages 363–370. IEEE Computer Society Press, 2001. 1, 2, 3, 4, 5, 7, 8, 9
15. P. Lindstrom and V. Pascucci. Terrain simplification simplified: A general framework for view–dependent out–of–core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239–254, 2002. 1, 2, 3, 4, 7, 8, 9
16. B. Lloyd and P. Egbert. Horizon occlusion culling for real–time rendering of hierarchical terrains. In *Proceedings Visualization 02*, pages 403–409. IEEE Computer Society Press, 2002. 2
17. D. Luebke, J. Reddy, M. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufman, 2002. 2
18. M. Ohlberger and M. Rumpf. Adaptive projection methods in multiresolutional scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 4(4):74–94, 1998. 1, 2, 3
19. R. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *Proceedings Visualization 98*, pages 19–26. IEEE Computer Society Press, 1998. 1, 2, 3, 8
20. R. Pajarola, M. Antonijuan, and R. Lario. QuadTIN: Quadtree based triangulated irregular networks. In *Proceedings Visualization '02*, pages 395–401. IEEE Computer Society Press, 2002. 2
21. A. Paul and K. Dobler. Adaptive realtime terrain triangulation. In *Proceedings of the WSCG '97*, 1997.
22. M. Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *International Journal on Numerical Methods in Engineering*, 20:745–756, 1984.
23. S. Röttger, W. Heidrich, P. Slussalek, and H.-P. Seidel. Real–time generation of continuous levels of detail for height fields. In *Proceedings WCSG '98*, pages 315–3222. IEEE Computer Society Press, 1998. 2
24. J. Stewart. Hierarchical visibility in terrains. In *Eurographics Workshop on Rendering '97*, pages 217–228, 1997. 2
25. B. Von Herzen and A. Barr. Accurate triangulations of deformed, intersecting surfaces. In *SIGGRAPH '87 Conference Proceedings*, pages 103–110, 1987.

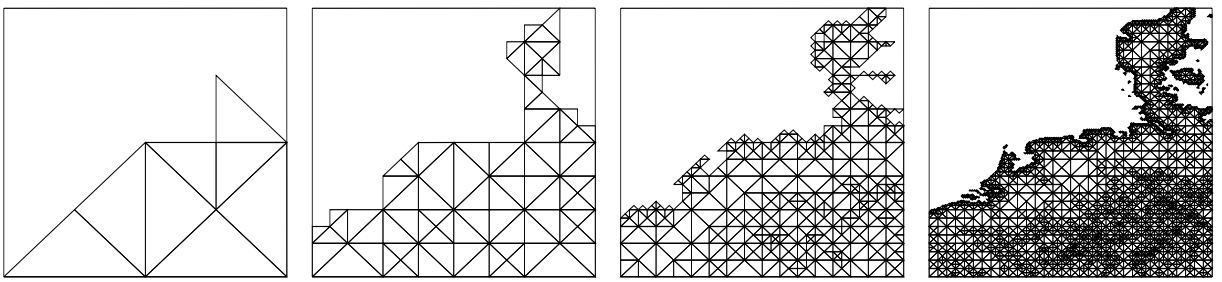

**Figure 9:** *The geometric distance metric* $\mu_{geo}$ *for thresholds of* $1, 0.1, 0.01$, *and* $0.001$.

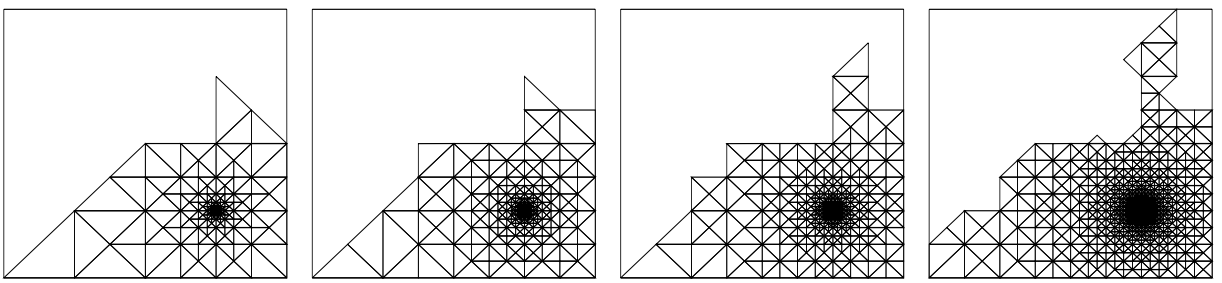

**Figure 10:** *The distance metric* $\mu_{dist}$ *based on the octagon distance for thresholds of* $1, 0.1, 0.01$, *and* $0.001$.

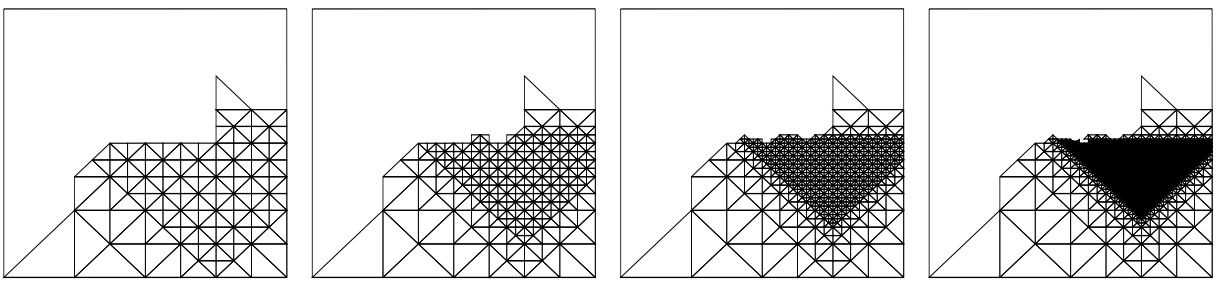

**Figure 11:** *The culling metric* $\mu_{cull}$ *based on the octagon distance for maximum levels of* $8, 10, 12$, *and* $14$.

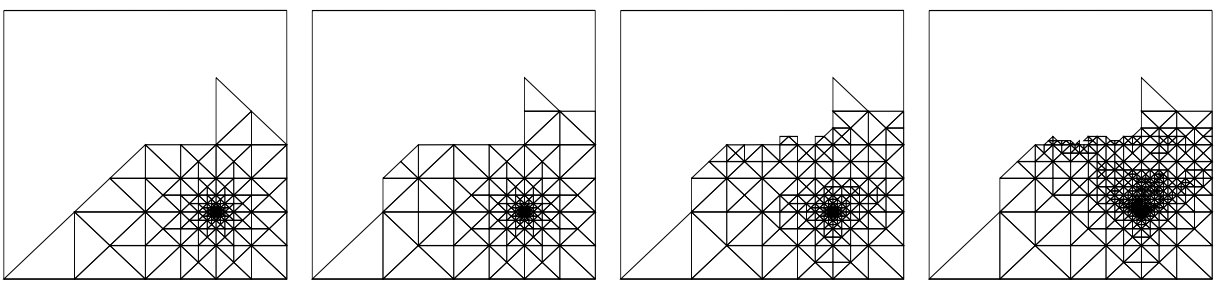

**Figure 12:** *The combined metric* $\mu = \mu_{geo} \cdot \mu_{dist} \cdot \mu_{cull}$ *for thresholds of* $1, 0.1, 0.01$, *and* $0.001$.

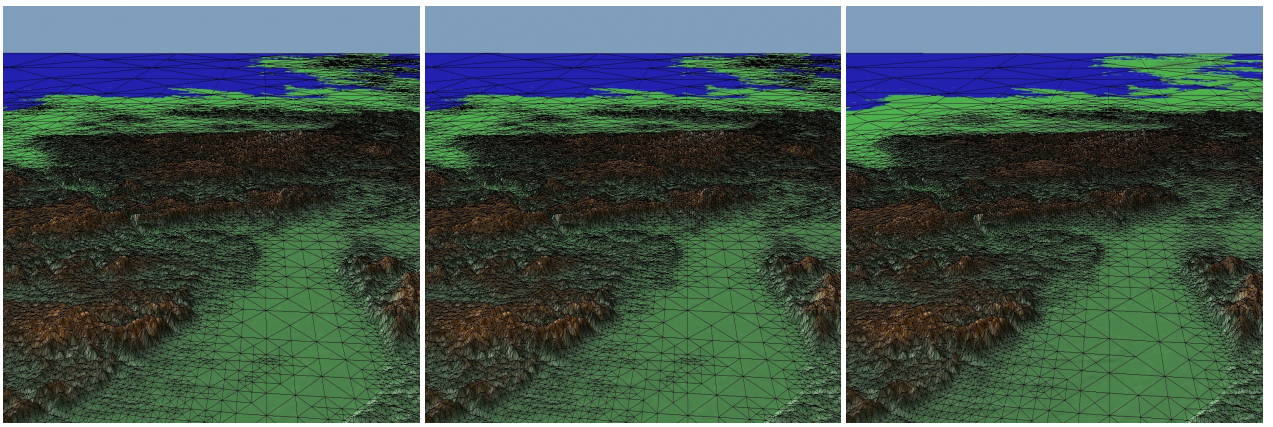

**Figure 13:** *Shown is a view from the south over the Upper Rhine Valley in Germany. We compare view–dependent triangulations for the vertical distance metric together with the nested sphere hierarchy (left), the octagon distance together with relative vertical distance bounds (middle), and the octagon distance with vertical min/max bounds (right).*

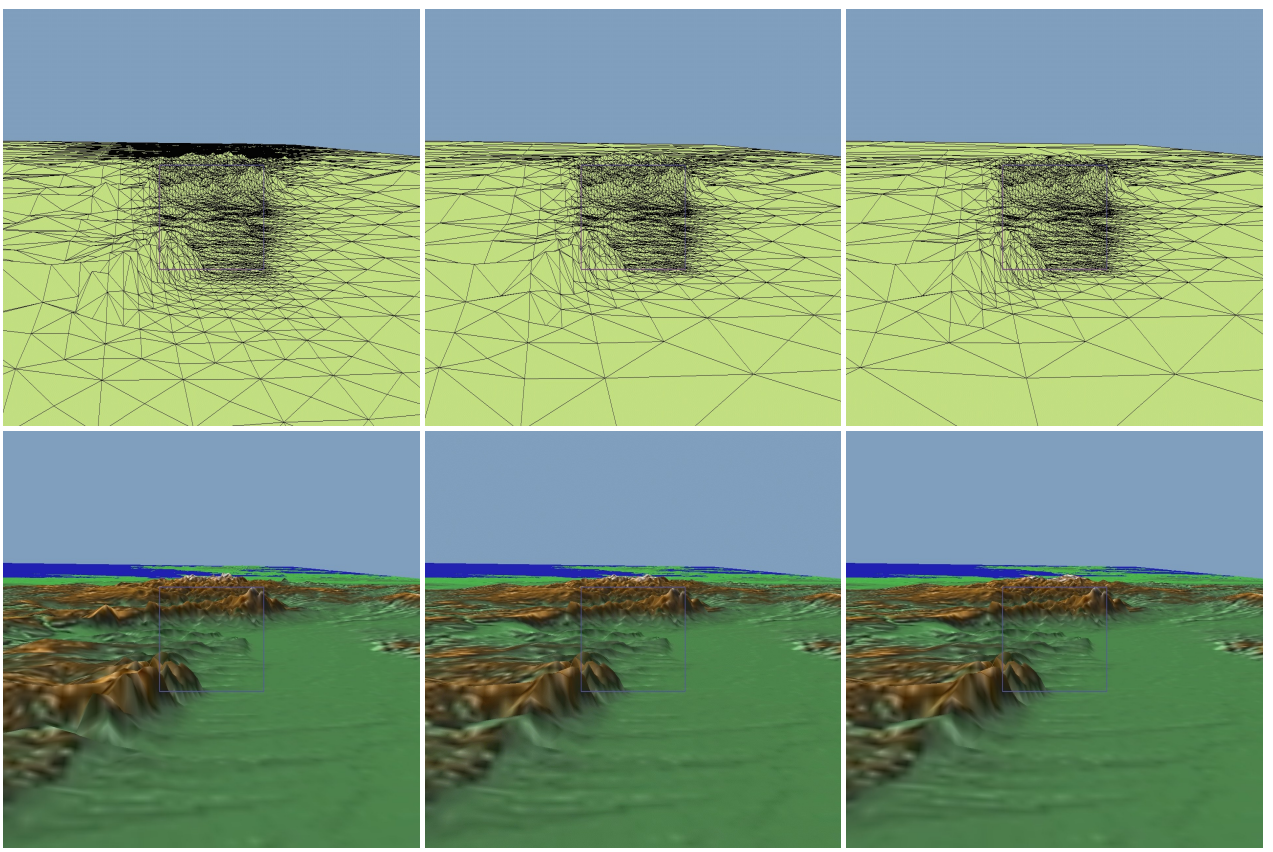

**Figure 14:** *For the same metrics as above we show here the clipping behaviour of the three algorithms. The viewport has been artificially reduced and is indicated by the square region in the middle of the images.*