Vision, Modeling, and Visualization (2010)

QEM-Filtering: A New Technique for Feature-Sensitive Terrain Mesh Simplification

F. Löffler and H. Schumann

University of Rostock/Institute of Computer Science/Computer Graphics Germany

Abstract

Terrain simplification generates multi-resolution models, from which - traditionally - irregular or semi-regular triangulations are extracted to render a terrain at a suitable level of detail. Recent terrain simplification techniques, in contrast, rely on GPU-friendly regular grids and generate multiple resolutions by applying the filtering and sub-sampling paradigm. However, due to the smoothing and uniform sampling, these techniques sparsely approximate the terrain surface. Consequently, in order to guarantee a certain error threshold, considerably more triangles need to be rendered.

In this paper, we present a novel feature-sensitive simplification technique. Our approach follows the aforementioned paradigm. The key idea is to maintain the regularity while recomputing the vertex positions by taking a specific error metric into account, namely the quadric error metric (QEM). Compared to previous approaches, we apply the paradigm to the grid of vertex-associated quadrics. From these we extract vertices of the new resolution by relying on quadric error minimization. We, thus, maintain the regular grid structure while preserving terrain features. Compared to methods, which are solely based on vertex-filtering and sub-sampling, our approach reduces the approximation error. As a consequence, we require fewer triangles, which improves the rendering performance.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Terrain rendering has a long research tradition and is used in many application domains. Due to the huge amount of data, rendering systems make use of multi-resolution models to extract a specific *level of detail*. A multi-resolution model is generated a-priori: Starting from the most detailed representation, with regard to a particular error metric, the number of primitives is successively decimated. The aim is to find a well-approximated model with fewer primitives. This process is referred to as *simplification*.

Traditionally, terrain multi-resolution models use irregular or semi-regular data structures at the granularity of individual triangles. Nevertheless, such structures are complex and computational expensive both for simplification and rendering. Hence, recent multi-resolution models take advantage of triangle clusters (patches) [PG07]. Especially regular triangulations and data-layouts are gaining in attractiveness: Due to their simple layout and topology (valence 6), they are ideally suited for hardware processing [GGH02]. Regularity guarantees efficiency in terms of memory management, serialization and rendering. Furthermore, well-studied image processing and compression methods can be applied directly. Consequently, in order to generate different resolution levels, recent terrain simplification methods apply low-pass filtering - a weighted average filtering to avoid aliasing - and uniform sub-sampling. However, this *filtering & sub-sampling* paradigm adapts the terrain surface only sparsely. Compared to traditional simplification methods, this procedure leads to higher approximation errors, which are caused by smoothing and uniform sub-sampling. As a result, with regard to triangles per error rate, rendering effort increases. This work aims at finding a compromise between the following alternative approaches:

- *Regular* layouts are well-suited for hardware processing

DOI: 10.2312/PE/VMV/VMV10/001-008

but the simplification is mostly based on low-pass filtering and thus, terrain features are "smoothed away".
- Complex algorithms use *irregular* data structures and recompute the vertex positions in such a way that the approximation error is minimized. As a result, the approximation is highly accurate, while computation is expensive and unsuitable for hardware processing.

The key idea behind our novel simplification approach is to recompute the vertex positions with regard to an error metric while simultaneously maintaining the regularity without remeshing [GGH02]. This leads to the following questions: First, how to recompute vertex positions to keep the error low in an efficient manner? Second, how to preserve the regularity? To accomplish this, we apply the filtering & sub-sampling paradigm in combination with the *quadric error metric* [GH97]. We initially associate a plane-set with each vertex expressed as *vertex-quadric*, whereby a plane-set represents the faces adjacent to the vertex. By the mean of filtering we compute the weighted average of the local vertex-quadric neighborhood. For the new resolution resulting from sub-sampling, we determine the so called *representative vertices* for the averaged vertex-quadrics by minimizing the quadric error. As a result, the approximation error is lower in contrast to vertex-based filtering and sub-sampling. Consequently, fewer triangles per error need to be rendered. As shown in the results section, this leads to an increasing performance.
The paper is outlined as follows: In the following Section, we present related work. In Section 3 we introduce our simplification method and show how the QEM can be efficiently used to recompute vertex positions. Finally, we discuss the results in Section 4 and finish with a conclusion and future work.

2. Related Work

A terrain rendering algorithm is mostly divided into two major steps: First, the preprocessing step, which generates a multi-resolution model using mesh simplification methods and second, the rendering step. There exist a wide range of multi-resolution data structures as well as simplification algorithms. In the following we focus on those structures and algorithms that are applied to terrain rendering. Given the context we refer in particular to the quadric error metric.

Multi-resolution models have been used in computer graphics for a long time. This is mainly caused by the desire to display large datasets. Traditionally, *vertex hierarchies* [XV96, Hop98, HSH09] or *multi-triangulation hierarchies* [Pup98, CGG*05] are a suitable choice from which highly adaptive and high quality irregular triangulations can be extracted. However, due to the complex nature and the expensive computational costs, in the field of terrain rendering, these general hierarchies have not been widely applied. Instead, recursive sub-division schemes have been developed, which produce semi-regular triangulations. Particularly, *longest-edge-bisection* sub-division is a very popular scheme (see [LP01]). This scheme recursively splits an isosceles triangle at the midpoint of its hypotenuse into two child-triangles. This approach is as simple as powerful and has been applied in various ways and various data structures (e.g. [EKT01, DWS*97, RHS98, LKR*96]). Another popular hierarchy commonly used is the *quad-tree*. A quad-tree is built by hierarchically sub-dividing the terrain in a restricted [Paj98, And07] or non-restricted manner [Ulr02].
However, to exploit the capabilities of recent GPUs, it is necessary to replace individual triangles by *patches* (triangle clusters). In more detail: Instead of associating a triangle or vertex with a node in the hierarchy, offline pre-computed and optimized triangle patches are used. Recent GPU-oriented multi-resolution hierarchies rely on both, regularly triangulated patches and regular data-layouts. This leads to new simplification methods. For a well elaborated overview we refer to [Paj02, LP02, dFKP05, PG07, DGY07].

Simplification generates a set of approximations by successively decreasing the number of primitives in a way that an approximation reflects the characteristics of the original to the greatest possible extent. For this purpose it is necessary to determine appropriate elements which can be eliminated. The extent to which this process is computationally expensive or not depends directly on the complexity of a given data structure.
Irregular data structures allow complex operations and error metrics. For instance, *vertex-clustering* [RB93], iterative *edge contraction* [Hop96], *mesh optimization* [HDD*93] or *wavelet analysis* [GGS95] techniques can be used to keep the approximation error at a low level. For an overview we refer to [CMS98]. Especially the approach proposed in [CGG*03] takes advantage of quadric-based simplification (cf. [GH97]) to generate high quality patch triangulations. This class of algorithms generates highly accurate approximations. However, the computation is very expensive and the algorithms are not designed for recent parallel hardware processing.
In contrast to that, algorithms based on semi-regular data structures merge elements by following the recursive sub-division scheme. This procedure introduces an error which can be measured. For instance, approaches as those proposed by [Pom00, Ulr02, SW06, DSW09, BGP09] generate semi-regular triangulated patches in an offline preprocess, whereas [LP01, Lev02] extract the approximation during the rendering processes. In general, due to their recursive nature, such algorithms are very fast and easy to implement.
Given the close interrelation of $n \times m$ grids of height values and images, for regular data structures, methods from the image processing community have been adapted. To generate multiple resolutions, recent terrain simplification algorithms rely on the *filtering & sub-sampling* paradigm. In more detail: An approximation is generated by uniformly sub-sampling the original regular input domain. To

avoid aliasing artifacts, a weighted average (low-pass) filtering is applied (cf. [dB00, LH04]). The algorithms proposed in [GMC*06,BGMP07] use wavelet analysis, whereas [HDJ04] apply a high quality low-pass filter for triangular patches. These algorithms are quite efficient. However, due to a sparse adaption of the terrain surface compared to irregular triangulations, they waste triangles. In conclusion, irregular triangulations lead to better results with regard to the triangles per error rate.

Quadric Error Metric - shortly QEM - is introduced by Garland et. al [GH97] and defines the distance from a vertex $\mathbf{v}$ to a set of associated planes $P(\mathbf{v})$. The metric has been developed for the *pair collapse* operator. The operator merges two vertices connected by an edge, which introduces an error. The metric is used to compute a new vertex position, which minimizes this above mentioned error: *quadric error minimization*.
Initially, for each vertex $\mathbf{v}$ the associated plane-set $P(\mathbf{v})$ is defined by the adjacent faces of $P(\mathbf{v})$ in the surface triangulation. The error $\triangle(\mathbf{v})$ is defined by the distance from $\mathbf{v}$ to the associated plane-set $P(\mathbf{v})$ whereby the explicit representation can be replaced by the *error quadric* - a symmetric 4×4 matrix $Q_{\mathbf{v}}$ - as follows:

$$\triangle(\mathbf{v}) = \sum_{\mathbf{p} \in P(\mathbf{v})} (\mathbf{p}^T \mathbf{v})^2 \quad (1)$$

$$= \mathbf{v}^T \left(\sum_{\mathbf{p} \in P(\mathbf{v})} \mathbf{p}^T \mathbf{p} \right) \mathbf{v} \quad (2)$$

$$= \mathbf{v}^T Q_{\mathbf{v}} \mathbf{v} \quad (3)$$

If a pair collapse $(\mathbf{v}_1, \mathbf{v}_2) \rightarrow \bar{\mathbf{v}}$ is applied, the position of $\bar{\mathbf{v}}$ is determined by minimizing the distance to the unified plane-set $P(\mathbf{v}_1) \cup P(\mathbf{v}_2) \rightarrow P(\bar{\mathbf{v}})$. By using the quadric representation $Q_{\mathbf{v}}$ of the plane-sets, the set-union operator reduces to quadric addition $Q_{\mathbf{v}_1} + Q_{\mathbf{v}_2} = Q_{\bar{\mathbf{v}}}$, whereby $Q_{\bar{\mathbf{v}}}$ specifies the error of the pair collapse. By the mean of error minimizing $\triangle(\mathbf{v})$ the new vertex $\bar{\mathbf{v}}$ (in homogeneous coordinates) can be directly computed from error quadric $\bar{\mathbf{v}} = Q_{\bar{\mathbf{v}}}^{-1}[0,0,0,1]^T$ as long as $Q_{\bar{\mathbf{v}}}$ is invertible.
Given its efficiency and quality, this metric is widely used in simplification processes. For instance, [CGG*03] use this metric to simplify patches and show that the contribution of a quadric can be scaled by simple matrix-scalar multiplication. Lindstrom et. al. [Lin00] rely on the metric for vertex-clustering. Vertex-clustering simplifies a given model by clustering all vertices and by computing a representative vertex for each cluster. The authors show that vertex-clustering is equivalent to performing the pair collapse operation to each vertex in a cluster simultaneously. Hence, the quadric error metric can be used to compute the representative vertex of a cluster. Based on this idea, [DT07] developed a real-time GPU-based mesh simplification method which stores the quadrics in texture memory and computes representative vertices of clusters in parallel directly on the graphics hardware.

Implications: Regular data-layouts exploit recent hardware, but the commonly applied low-pass filtering in the simplification leads to a wasting number of triangles per error rate in the rendering process. More complex techniques iteratively recompute vertex positions with regard to an error metric, leading to high quality approximations. However such algorithms are complex and time consuming. Furthermore, the data structures and layouts are unfavorable for recent hardware. Hence, it should be expected that the combination of both aspects will be beneficial. We will confirm this statement by the approach proposed here.

3. QEM-Filtering

In this section we focus on the key aspects of our novel simplification algorithm. The goal is to compute the vertex positions with regard to a certain error metric while maintaining the regularity of the data without remeshing (i.e. [GGH02]). By the mean of an iterative process, different levels of resolution are generated. Thereby the developed algorithm takes advantage of hardware processing capabilities. The benefits arising thereby are: First, the metric minimizes the approximation error. As a result, fewer primitives need to be rendered in order to guarantee a certain error threshold. Second, the regularity guarantees efficiency with respect to compression, management and hardware processing. Thus, optimized algorithms can be applied. Third, parallel hardware architectures are the key for high performance processing. Hence, it is feasible to handle a large amount of data within decent time. In order to benefit from these advantages, we need to solve the following problems:

1. We need a particular error metric, which allows for optimizing vertex positions. The metric needs to be feature-sensitive and has to be computationally efficient while avoiding complex data structures.
2. We need to maintain the regularity of the data-layout. More precisely, in order to preserve the grid layout, vertex positions can be changed only locally.
3. We need to take care of hardware processing restrictions. In other words: Complex data structures and dependencies should be avoided in order to compute vertices in a parallel manner.

Addressing the first requirement, we choose the quadric error metric (see Section 2). The error quadric is represented by a 4×4 matrix. Therefore, it is easy to implement it on current programmable hardware (cf. [DT07]). To addresses issues 2 and 3, we apply the filtering & sub-sampling paradigm. Filtering can be implemented efficiently on parallel hardware architectures and uniform sub-sampling guarantees a regular data-layout. The question arising here, is how to combine these two aspects. QEM is normally used for the pair-collapse operator or vertex-clustering, whereas

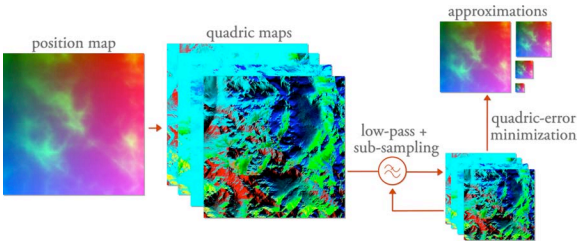

Figure 1: *The work-flow of QEM-filtering. We start with the most detailed position map and generate the quadric map. We apply a low-pass filter to the quadric map followed by a sub-sampling. The position map of the new approximation is then derived by quadric error minimization.*

low-pass filtering and sub-sampling generates new resolutions by computing the weighted average of the local neighborhood.
Although filtering & sub-sampling is similar to vertex-clustering, the major difference is that vertex-clustering requires a unique mapping from the vertices to clusters, whereas filtering weights their local neighborhoods. However, both approaches compute the representative vertex based on local information. With quadrics in mind we can derive two conclusions: First, quadrics represent the local surface information of a vertex. Second, the contribution of a quadric with respect to a quadric addition can be scaled (cf. [CGG*03]). This means that instead of applying the filtering to the vertices, we are able to apply it to the associated quadrics. This is equitable to a weighted combination of the local surface information. As a consequence, the sub-sampled quadrics represent all surface information of the filtered local neighborhoods. Thus, the representative vertices computed by quadric error minimization represent the optimal positions for the approximated surface. In contrast to vertex-clustering, we need no mapping from vertices to clusters. Hence, no dependency rules need to be considered. Furthermore, neighborhood is implicit on a regular grid. In this way, both the filtering & sub-sampling as well as quadric-error minimization can be implemented resource-friendly on programmable hardware. The basic principle of our algorithm can be summarized as follows (see Figure 1):

1. We generate a *quadric map* by computing the quadric for each vertex.
2. We apply a low-pass filtering and sub-sampling to the quadric map, which results in a new resolution.
3. For each quadric in the sub-sampled quadric map, we compute the representative vertex by quadric-error minimization, which generates the new approximation.

However, some problems need to be solved. First, the initial quadric-set requires surface information, which is usually not defined when working with height-fields as a source. In this particular case, the generation of these initial quadrics is a sensitive process, which has direct impact on the approximation result. Second, we need to evaluate the low-pass filtering of quadrics. And third, we need to take care of restrictions related to multi-resolution structures (e.g. handling patch edges).

3.1. Quadric-Map Generation

The quadric map contains the associated quadric $Q_{\mathbf{v}}$ for each vertex $\mathbf{v}$. Initially, the quadric $Q_{\mathbf{v}}$ represents the plane-set $P(\mathbf{v})$ which is defined by all adjacent triangles of $\mathbf{v}$. Commonly, for each plane $\mathbf{p} \in P(\mathbf{v})$ the *fundamental quadric* $K_{\mathbf{p}} = \mathbf{p}\mathbf{p}^T$ is computed. The whole set of these fundamental quadric defines the initial vertex-quadric $Q_{\mathbf{v}}$ for the vertex $\mathbf{v}$ (cf. [GH97]). That means that this procedure expects an existing triangulation of the surface which is not given if working with height-fields.
A height-field is a discretization of a continuous terrain surface and it is the preferred representation of terrains. Due to the discretization, no information about the surface between the sample points is available. However, most terrain rendering systems use a regular triangulation (valence 6) or a semi-regular (valence 4 or 8) triangulation (cf. [PG07]). In this case, it is not ensured that the surface is well-described by the triangulation, due to the non coplanarity of sample points representing a grid-cell.
A significantly better result can be achieved by determining additional sample points through interpolation (bilinear or bicubic). A triangulation with regard to these additional sample points results in a piecewise more precise description of the surface. Thus, the simplification process generates more accurate approximations than without interpolated sample points. Nevertheless, the question that arises is, how the additional sample points can be taken into account without increasing memory costs.
Our idea is to encode the surface described by the interpolated sample points in the vertex-quadrics of the original sample points. For this purpose, we determine the first ring neighborhood of a vertex by interpolating between adjacent vertices (see Figure 2). In other words: We up-sample the discretized surface by using an appropriate interpolation method. In this way we add new vertices. The initial plane-set $P(\mathbf{v})$ for each vertex $\mathbf{v}$ is then defined by constructing the faces for the newly interpolated vertices. Hence, each $Q_{\mathbf{v}}$ in

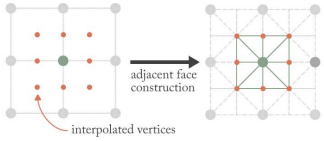

Figure 2: *For a vertex (green) we determine the first ring neighborhood (vertices in red) by interpolation and construct the 8 adjacent faces (triangles in green). These faces are used to compute the initial vertex-quadric.*

the original resolution piece wisely specifies the more precise surface triangulation (see Figure 2). The algorithm can be outlined as follows:

- We transform the height-field to a position map (cf. Figure 1), which is an explicit representation of the grid vertices. In detail: For each sample point in the height-field we compute the 3D position using a particular projection.
- For each vertex $\mathbf{v}$ we determine the first ring neighborhood by interpolation.
- We connect $\mathbf{v}$ with the interpolated first ring neighborhood which results in 8 adjacent faces. For each plane, we compute the fundamental-quadric as described above. The whole set of these defines the vertex-quadric $Q_\mathbf{v}$.

For our current implementation we use bilinear interpolation, which is supported by programmable hardware. For both regular and semi-regular triangulations this interpolation works well. Empirical tests provide evidence that compared to non-interpolation, this procedure leads to better approximation results. However, more accurate methods such as bicubic interpolation might be applied, which are likely to improve the approximation quality.

3.2. Filtering & Sub-Sampling

Multiple resolutions of a regular input are derived by sub-sampling. In its simplest form, sub-sampling is implemented by nearest neighborhood interpolation. For a regular parameterized grid $I(s,t)$ with the dimension $2^n+1 \times 2^n+1$, which is usually used in terrain rendering, the next resolution level can be efficiently generated by sub-sampling as follows (cf. [PG07]):

- For semi-regular triangulations, all vertices that have an even linear index $i = s+t(2^n+1)$ are sampled.
- For regular triangulations, the resolution is halved by sampling all vertices, which have an even parameterization in each dimension: $s \bmod 2 == 0$ and $t \bmod 2 == 0$.

However, by sub-sampling important information is lost, which leads to aliasing artifacts. These artifacts are avoided by applying a low-pass filter. The low-pass filter eliminates high-frequencies, which are not reconstructible by the sub-sampled approximation. In other words: The low-pass filter smoothes the original surface by computing the weighted sum of the local neighborhood for each vertex. Accordingly, vertices that have been sampled for the new resolution include local neighborhood information.
For the input $I(s,t)$ the smoothing can be realized by weighting the $k \times k$ local neighborhood by an appropriate *filter-kernel*, which specifies the weights w_{ij} with $i,j = 0,\ldots,k-1$ and the sum of the weights sum_w. In this case, the weighted average $avg(s,t)$ is computed as follows:

$$avg(s,t) = \frac{1}{sum_w} \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} I(s+i-\frac{k}{2}, t+j-\frac{k}{2}) w_{ij} \quad (4)$$

After the low-pass filtering of the quadric map, we sub-sample the map to the desired resolution. For each vertex-quadric $\bar{Q}_\mathbf{v}$ in the quadric map, we compute the vertex $\bar{\mathbf{v}}$ by quadric error minimization (see Section 2). In case that $\bar{Q}_\mathbf{v}$ is not invertible, we fall back to the weighted average of the local neighborhood vertices. However, due to the following issues, filtering & sub-sampling has to be done carefully:

Kernel properties: Quadrics represent local surface information (plane-sets). Thus, large kernels $k > 3$ take the mean of these information from a wide local neighborhood. Consequently, the quadric-error minimization determines the representative vertex by finding the minimum distance to the unified plane-sets. However, the representative vertex does not approximate the local neighborhood well, especially if the resolution has been halved. As a rough guideline, we propose to use kernels with the size $k = 2^{scale-1}+1$ where $scale$ is the down-scale factor.

Approximation quality: By low-pass filtering the local neighborhood, degenerated or small surface elements - so called slivers (cf. [Hop98]) - can be generated. This problem can be solved by applying a final smoothing step which reduces the quality, or by adding so called *dihedral planes* to the initial plane-sets (cf. [CGG*03]). For this purpose, we determine the dihedral plane-set $P_D(\mathbf{v})$ of a vertex $\mathbf{v}$ by its adjacent faces as follows: For each pair of two adjacent faces $f_1, f_2 \in P(\mathbf{v})$ both the edge vector $\mathbf{e}$ of the shared edge and the average normal of the two faces $\mathbf{n}_{f_1 f_2}$ are computed. These two vectors and the vertex $\mathbf{v}$ span the dihedral plane. Accordingly, the initial quadric $Q_\mathbf{v}$ is iteratively computed as follows: $Q_\mathbf{v} = (1-a)Q_\mathbf{v} + aD_\mathbf{v}$, whereby $D_\mathbf{v}$ is the quadric representing the dihedral plane-set $P_D(\mathbf{v})$ and a is a user-defined scaling factor. In our implementation, we use a scaling factor of $\frac{1}{64}$ which has to be found as a good compromise between mesh-smoothness and approximation quality (see Figure 3).

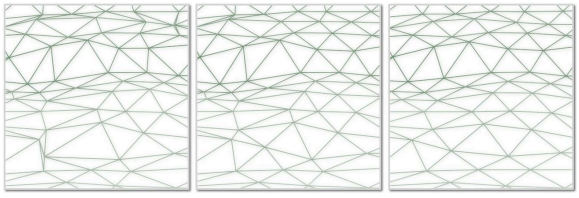

Figure 3: *Comparison of different smoothing variants. No smoothing (left) leads to degenerated surface elements, whereas Laplacian smoothing (centre) and weighted dihedral planes (right) improve the mesh quality.*

Boundaries: With multi-resolution hierarchies in mind, we need to take care of patch boundaries. Neighbored patches are processed independently. Hence, the boundaries have been modified in such a way that they are coincident. For this purpose, the quadric error metric allows to define boundary constraints. However, we choose a straightforward approach and determine the boundary vertices by nearest-neighbor interpolation (cf. [dB00]).

Performance: In the filtering processes we have to deal with 4×4 matrices. Hence, it is desirable to reduce the quadratic $k * k$ look-up overhead introduced by a two-dimensional filter kernel. Our solution is to use separable filter-kernels only. In this case we apply a one-dimensional filter-kernel successively to each dimension. This reduces the number of look-ups to $2k$. Given the fact that the most low-pass filters are separable, this can be applied easily.

4. Results

To evaluate our novel algorithm, we measure both, the approximation quality as well as the impact on the rendering performance. For this purpose, we test three representative height-fields, which differ in the feature distribution and scaling, respectively. The results are compared to vertex-based filtering with the following filter kernels:

- A Gaussian $\sqrt{2}$ filter (size $k = 3$)
- A 5/3 Daubechies wavelet filter (size $k = 5$)

The presented approach performs the QEM-filtering with a Gaussian $\sqrt{2}$ kernel (size $k = 3$). For our tested height-fields this works best. Basis for these tests is a multi-resolution hierarchy, which relies on a restricted quad-tree (cf. [And07]). The respective patches are generated by filtering & sub-sampling. For the vertex-based filtering we construct a 1D-displacement map for patches which stores the height values at a grid point only. In contrast, the QEM-filtering generates a 3D-displacement map for patches storing a 3D-displacement for each grid point. Hence a hierarchy of 1D or 3D displacement map patches is generated. During the rendering, appropriate patches are view-dependently selected from the hierarchy by using the screen space metric proposed by [LP02]. This metric requires the *object space error* (approximation error) of the patches. Since the widely applied vertical distance metric is inexact [Hop98], we use the approximated *Hausdorff distance* for triangle-meshes to measure the object space error. The proof-of-concept is implemented in C++ using OpenGL, whereby the filtering & sub-sampling is implemented in GLSL. Patch rendering is performed on programmable hardware by *instanced tessellation* [Tat08] of the 1D- and 3D-displacement map, respectively. All tests are carried out on a Core 2 Quad 2,83 GHz with a GeForce 280 GTX. The frame-statistics is captured at a screen resolution of 1920×1080 during the playback of a flight path over the height-fields using a screen space error tolerance of $0.9px$.

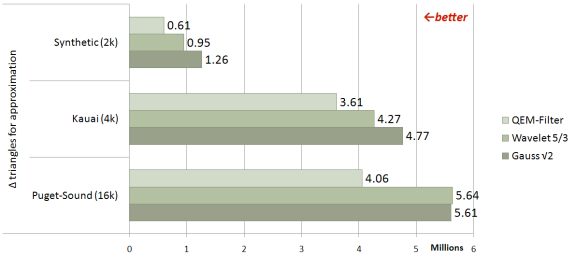

Figure 4: *Comparison of the average numbers of triangles for a view-dependent approximation regarding to a screen space error tolerance of 0.9px.*

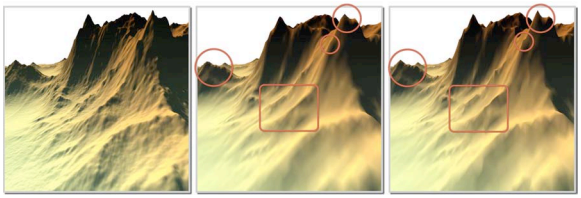

Figure 5: *Visual comparison of a high resolution height-field (left) with a approximation level generated by a Gaussian vertex-filtering (centre) and the same approximation level generated with the proposed method (right). Please note the features (marked red) preserved by our method.*

Our results provide evidence that the presented approach reduces significantly the average number of triangles per view-dependent approximation (see Figure 4). The algorithm nearly halves, for instance, the number of triangles compared to the Gaussian vertex-filtering for the *Synthetic (2k)* height-field. The QEM-filtering approach allows for adapting significant features. Thus, sharp features are maintained and heights are preserved as illustrated in Figure 5. As a result, the approximation error is decreased leading to high quality approximations as compared to vertex-based filtering. Given the fact that both, the 1D-displacement map as well as the 3D-displacement map use the same rendering algorithm, the reduction of the number of triangles directly affects the rendering performance. This is illustrated in Figure 6. Consequently, we increase the rendering performance by more than 15% in contrast to the wavelet filter and in some cases by more than 30% compared to a Gaussian filter.
However, we need to mention that due to the filtering of quadrics and the matrix operations involved, time costs for the pre-processing step slightly increase. Nevertheless, time costs for rendering are reduced. Moreover, our algorithm is designed for recent programmable hardware, which provides massive computational power. The overhead, therefore, is negligible.
Although our algorithm generates high quality approximations, we need to store the full 3D vertex position instead of

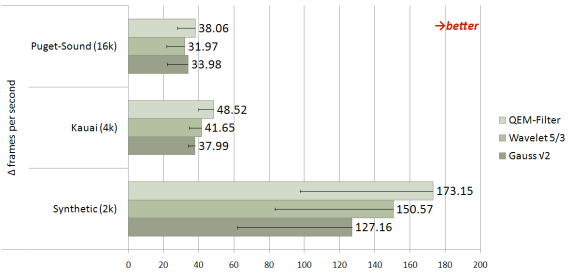

Figure 6: *Comparison of the average frames per second (fps) for a view-dependent approximation regarding an error tolerance of* 0.9*px. The black bars show the minimum number of fps. Due to the numbers of reduced triangles (see Figure 4), the rendering performance increases.*

a single scalar value for the height. Hence, the storage costs increase by factor 3. For instance, the multi-resolution hierarchy for the 4k height-field requires 86MB of memory for the 1D-displacement map, whereas 258MB are allocated for the 3D-displacement map. Even though memory costs are increased, rendering costs are decreased. Moreover, since we maintain a regular data-layout, optimized compression schemes can be applied.
To conclude, the presented approach generates high quality approximations by minimizing the approximation error. On the one hand, this requires more memory resources in contrast to other techniques. However, on the other hand, rendering performance is increased. Hence, a compromise between time and resources has been found.

5. Conclusion

We present a novel feature sensitive simplification method for terrain rendering. In detail: We apply the filtering & subsampling paradigm to so-called vertex-quadrics, which represent the terrain surface. By the mean of quadric error minimization we generate new well-approximated resolutions. Due to the use of the quadric error metric in combination with sub-sampling, we are able to reduce the approximation error in contrast to vertex-based filtering & sub-sampling. At the same time, we maintain a regular data-layout. Our testing scenarios provide evidence that the approximation error is significantly reduced and that the rendering performance is improved. Given the fact that the algorithm is well-suited for parallel hardware processing, the computational overhead is negligible. Finally, the presented algorithm requires more memory resources caused by the 3D-displacement for an approximation. However, the data-layout is well-suited for hardware rendering. Hence, a good compromise between both rendering time and memory resources has been found.
Future work should focus on evaluating different filter-kernels and filter-sizes. Moreover, during the simplification process other aspects, such as lighting and surface attributes, should be taken into account. In this case, an extended quadric error metric has to be used. Finally, with the new tessellation capabilities in mind, bi-cubic interpolation for both rendering and simplification has to be considered.

References

[And07] ANDERSSON J.: Terrain rendering in frostbite using procedural shader splatting. In *ACM SIGGRAPH courses* (2007), ACM, pp. 38–58. 2, 6

[BGMP07] BETTIO F., GOBBETTI E., MARTON F., PINTORE G.: High-quality networked terrain rendering from compressed bitstreams. In *Proceedings of the twelfth international conference on 3D web technology* (2007), ACM, p. 44. 3

[BGP09] BÖSCH J., GOSWAMI P., PAJAROLA R.: RASTeR: Simple and Efficient Terrain Rendering on the GPU. In *Eurographics - Areas Papers* (2009), Eurographics Association, pp. 35–42. 2

[CGG*03] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: BDAM - batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum 22* (2003), 505–514. 2, 3, 4, 5

[CGG*05] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Batched multi triangulation. *IEEE Visualization* (2005), 207–214. 2

[CMS98] CIGNONI P., MONTANI C., SCOPIGNO R.: A comparison of mesh simplification algorithms. *Computers & Graphics 22*, 1 (1998), 37–54. 2

[dB00] DE BOER W. H.: Fast terrain rendering using geometrical mipmapping, 2000. 3, 6

[dFKP05] DE FLORIANI L., KOBBELT L., PUPPO E.: *Mathematics and Visualization*. Springer Berlin Heidelberg, 2005, ch. A Survey on Data Structures for Level–of–Detail Models, pp. 49–74. 2

[DGY07] DIETRICH A., GOBBETTI E., YOON S.: Massive-Model Rendering Techniques. *IEEE Computer Graphics and Applications* (2007), 20–34. 2

[DSW09] DICK C., SCHNEIDER J., WESTERMANN R.: Efficient geometry compression for GPU-based decoding in realtime terrain rendering. *Computer Graphics Forum 28*, 1 (2009), 67–83. 2

[DT07] DECORO C., TATARCHUK N.: Real-time mesh simplification using the gpu. In *I3D: Proceedings of the symposium on Interactive 3D graphics and games* (2007), pp. 161–166. 3

[DWS*97] DUCHAINEAU M. A., WOLINSKY M., SIGETI D. E., MILLER M. C., ALDRICH C., MINEEV-WEINSTEIN M. B.: Roaming terrain: real-time optimally adapting meshes. In *IEEE Visualization* (1997), pp. 81–88. 2

[EKT01] EVANS W., KIRKPATRICK D., TOWNSEND G.: Right-triangulated irregular networks. *Algorithmica 30*, 2 (2001), 264–286. 2

[GGH02] GU X., GORTLER S., HOPPE H.: Geometry images. *ACM Transactions on Graphics 21*, 3 (2002), 355–361. 1, 2, 3

[GGS95] GROSS M., GATTI R., STAADT O.: Fast multiresolution surface meshing. In *IEEE Conference on Visualization Proceedings* (1995), pp. 135–142. 2

[GH97] GARLAND M., HECKBERT P.: Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., p. 216. 2, 3, 4

[GMC*06] GOBBETTI E., MARTON F., CIGNONI P., DI BENEDETTO M., GANOVELLI F.: C-bdam - compressed batched dynamic adaptive meshes for terrain rendering. *Computer Graphics Forum 25*, 3 (sep 2006). 3

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), ACM, p. 26. 2

[HDJ04] HWA L. M., DUCHAINEAU M. A., JOY K. I.: Adaptive 4-8 texture hierarchies. In *Proceedings of the conference on Visualization* (2004), IEEE Computer Society, pp. 219–226. 3

[Hop96] HOPPE H.: Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 99–108. 2

[Hop98] HOPPE H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings of the conference on Visualization* (1998), IEEE Computer Society Press, pp. 35–42. 2, 5, 6

[HSH09] HU L., SANDER P., HOPPE H.: Parallel view-dependent refinement of progressive meshes. In *Proceedings of the symposium on Interactive 3D graphics and games* (2009), ACM New York, NY, USA, pp. 169–176. 2

[Lev02] LEVENBERG J.: Fast view-dependent level-of-detail rendering using cached geometry. In *Proceedings of the conference on Visualization* (2002), IEEE Computer Society, pp. 259–266. 2

[LH04] LOSASSO F., HOPPE H.: Geometry clipmaps: terrain rendering using nested regular grids. In *ACM Transactions on Graphics* (2004), ACM, pp. 769–776. 3

[Lin00] LINDSTROM P.: Out-of-core simplification of large polygonal models. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 259–262. 3

[LKR*96] LINDSTROM P., KOLLER D., RIBARSKY W., HODGES L. F., FAUST N., TURNER G. A.: Real-time, continuous level of detail rendering of height fields. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 109–118. 2

[LP01] LINDSTROM P., PASCUCCI V.: Visualization of large terrains made easy. In *Proceedings of the conference on Visualization* (2001), IEEE Computer Society, p. 371. 2

[LP02] LINDSTROM P., PASCUCCI V.: Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 239–254. 2, 6

[Paj98] PAJAROLA R.: Large scale terrain visualization using the restricted quadtree triangulation. *IEEE Visualization Conference 0* (1998), 19. 2

[Paj02] PAJAROLA R.: Overview of quadtree-based terrain triangulation and visualization. *UCI-ICS Technical Report* (2002). 2

[PG07] PAJAROLA R., GOBBETTI E.: Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer 23*, 8 (2007), 583–605. 1, 2, 4, 5

[Pom00] POMERANZ A.: *ROAM Using Surface Triangle Clusters (RUSTiC)*. PhD thesis, UNIVERSITY OF CALIFORNIA, 2000. 2

[Pup98] PUPPO E.: Variable resolution triangulations. *Computational Geometry: Theory and Applications 11*, 3-4 (1998), 219–238. 2

[RB93] ROSSIGNAC J., BORREL P.: Multi-resolution 3D approximations for rendering complex scenes. *Modeling in Computer Graphics 455* (1993), 466. 2

[RHS98] RÖTTGER S., HEIDRICH W., SEIDEL H.-P.: Real-time generation of continuous levels of detail for height fields. pp. 315–322. 2

[SW06] SCHNEIDER J., WESTERMANN R.: GPU-friendly high-quality terrain rendering. *Journal of WSCG 14*, 1-3 (2006), 49–56. 2

[Tat08] TATARINOV A.: Instanced tessellation in directx10. *Presentation, Game Developers Conference* (2008). 6

[Ulr02] ULRICH T.: Chunked lod, 2002. http://www.tulrich.com/geekstuff/chunklod.html. 2

[XV96] XIA J. C., VARSHNEY A.: Dynamic view-dependent simplification for polygonal models. In *Proceedings of the 7th conference on Visualization* (1996), IEEE Computer Society Press, pp. 327–ff. 2