

The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data

Roberto Grosso, Christoph Lürig and Thomas Ertl

Computer Graphics Group
Universität Erlangen-Nürnberg, Germany*

Abstract

Multilevel representations and mesh reduction techniques have been used for accelerating the processing and the rendering of large datasets representing scalar or vector valued functions defined on complex 2 or 3 dimensional meshes. We present a method based on finite element approximations which combines these two approaches in a new and unique way that is conceptually simple and theoretically sound. The main idea is to consider mesh reduction as an approximation problem in appropriate finite element spaces. Starting with a very coarse triangulation of the functional domain a hierarchy of highly non-uniform tetrahedral (or triangular in 2D) meshes is generated adaptively by local refinement. This process is driven by controlling the local error of the piecewise linear finite element approximation of the function on each mesh element. A reliable and efficient computation of the global approximation error combined with a multilevel preconditioned conjugate gradient solver are the key components of the implementation. In order to analyze the properties and advantages of the adaptively generated tetrahedral meshes we implemented two volume visualization algorithms: an iso-surface extractor and a ray-caster. Both algorithms, while conceptually simple, show significant speedups over conventional methods delivering comparable rendering quality from adaptively compressed datasets.

1 INTRODUCTION

Many applications in computer graphics and visualization deal with increasingly large scalar or vector valued functions defined on complex two or three dimensional meshes. Typical examples range from two dimensional data such as satellite images or laser scans to three dimensional scalar fields on regular grids like medical datasets or even time dependent flow fields on unstructured grids of a complex CFD topology. The generation of geometric primitives by visualization mappings like height fields, iso-surfaces or streamlines and the rendering for such complex meshes is very compute intensive and requires large storage capacities. As the information contained in the data is often redundant, a common strategy for hand-

ling such problems is compression with information loss as small as possible and a multiresolution representation of the dataset for a flexible level-of-detail control.

In this paper we present a method based on adaptive multilevel finite elements which generates a sequence of nested approximating spaces for a given function. The main idea is to interpret mesh reduction and optimization as an approximation problem in finite element spaces with adaptive mesh refinement and error control. The input data is considered to be a discretized representation of the smooth function to be approximated. In contrast to the techniques based on wavelets and multiresolution analysis [8, 10] our algorithm starts at the coarsest level and proceeds to the finer ones (as in [6]). This corresponds in a natural way to the sequence of steps when performing explorative visualization or progressive data transmission and display. We interpret this process as a *data driven* mesh generation, since we iteratively generate a hierarchy of tetrahedral (or triangular in 2D) meshes by adaptive refinement based on error analysis. After each iteration step of our algorithm, a complete hierarchy of nested spaces is available, with the *best approximation* of the underlying function known at the finest level of the hierarchy. Coarser representations of the function are given by a hierarchical basis [32] decomposition at the lower level of the hierarchy. The error measurement and the approximation itself are based on the standard L_2 norm. More general norms, such as the Sobolev norm which allows the integration of a priori knowledge of the smoothness of the function to be approximated can also be implemented, but this is a topic of future work.

In addition the many advantages of a multiresolution representation of the data, such as level-of-detail control, compression, etc., which are already well analyzed in the computer graphics literature [8, 14], the approach proposed here has the following remarkable features:

- *Adaptivity*: The resulting algorithms for 2D and 3D data are very simple and efficient due to the high compression potential of adaptively generated meshes.
- *Error Control*: The global accuracy of the approximation is well defined in the L_2 norm and can be controlled by the user. Additionally, control over the local contributions to the error is also given, which allows for local mesh refinement.
- *Topology Control*: Because the refinement operations are well defined, the meshes obtained exhibit very nice properties, i.e. no thin triangles or tetrahedra, and no vertices with many edges are generated.

Our approach works on irregular grids. Since the mesh is generated during the approximation, no assumptions about the mesh topology of the input data have to be made. Furthermore, the algorithm is formulated independently of the problem dimension and was implemented for 2D and 3D data. The main steps of the algorithm can be summarized as follows:

*Lehrstuhl für Graphische Datenverarbeitung (IMMD9), Universität Erlangen-Nürnberg, Am Weichselgarten 9, 91058 Erlangen, Germany, Email: grosso@informatik.uni-erlangen.de

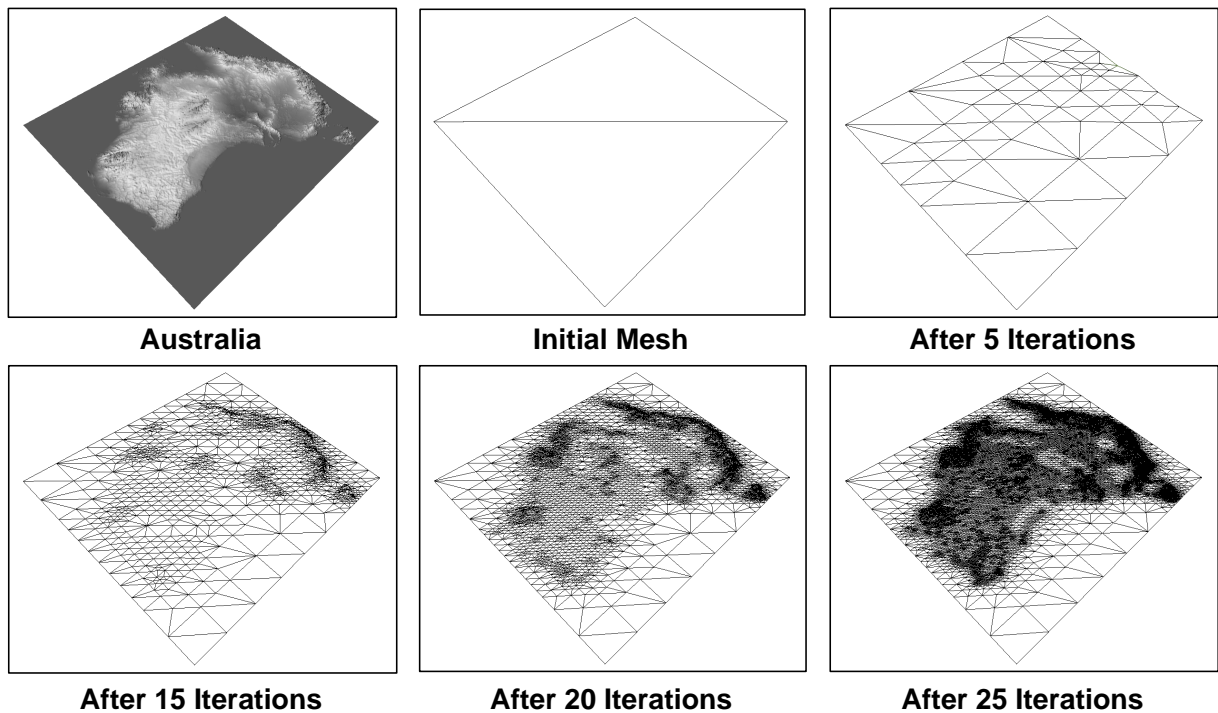


Figure 1: The topography of Australia given by a 409 x 505 height field serves as a 2D example for the mesh optimization by adaptive multilevel finite elements. Dataset contributed by M. Hutchinson of the Australian National University, Canberra

- Choose an intentionally coarse triangulation of the domain (e.g. a cube divided into 6 tetrahedra). Compute the best approximation by a direct solver to start the iteration.
- Successively refine the mesh elements with the estimated local approximation error above a given threshold. Avoid hanging nodes by adjusting the neighborhood of the refined cells. Compute an iterative solution of the improved approximation.
- Stop, if the global error is small enough.

Since one of the classical applications of mesh optimization is terrain visualization, we demonstrate the hierarchical approach by showing five characteristic meshes of a 2D dataset representing the topography of Australia in Fig. 1.

In order to investigate, how the mesh hierarchy can be exploited for 3D scalar fields, we have developed an iso-surface extraction and a volume ray-casting algorithm. The drawbacks of standard iso-surface methods for indirect volume visualization are the time spent on the extraction of the polygonal description eventually visiting each cell of a large dataset and the enormous amount of generated triangles, which makes interactive handling difficult even on high-end graphic workstations. A more general approach is based on mesh hierarchies, which are generated adaptively by local refinement operations. The idea is to work on a compact and simplified data representation rather than on the simplification of the resulting geometric primitives. With the help of an implicit adaptive integration technique we show how to use the mesh hierarchy for direct volume rendering.

The following sections describe the related work, the theoretical background of the finite element approximation, some of the implementation details of the algorithm and elaborate on the iso-surface and volume ray-casting applications. We conclude with the analysis of the experimental results and with some ideas for future work.

2 RELATED WORK

In this section we discuss related approaches for mesh optimization and multiresolution representations and analyze some important aspects and algorithms for the visualization of volume data.

2.1 Aspects of Mesh Reduction

A widely used multiresolution analysis technique is the wavelet decomposition. The use of multiresolution analysis on nested approximation spaces has successfully been extended to meshes of arbitrary topology by Lounsbery et al. [19, 8]. Besides many advantages such as compression, level-of-detail control and the generation of the multiresolution representation some aspects remain open. One problem is, that the wavelet refinement cannot be performed adaptively, resulting in a re-meshing as a post-processing step, after the wavelet decomposition has been performed. Re-meshing is necessary to avoid hanging nodes, when adding wavelet coefficients in the reconstruction step for local refinement. The orthogonal wavelet projection guarantees a best least-squares approximation with respect to the L_2 norm. Because of the successive adding of orthogonal subspaces, an error estimation of the overall accuracy of the approximation can be given [9].

The methods described above approach the approximation problem from a functional point of view. In contrast, the approaches of Schroeder et al. [22], Klein et al. [16], Turk [27] and Hoppe [14] for 2D-meshes and Cignoni et al. [6], Guo [12] and Lürig et al. [20] for 3D-meshes can be considered geometric. These approaches analyze every single node with some local criteria in order to decide whether it can be eliminated or not followed by a re-triangulation. A widely used geometric error criterion is the Hausdorff or Frechet distance, which measures the distance between manifolds. This approach also allows the construction of a hierarchy of approximations depending on the distance, but there is no simple recursion relation between objects in the hierarchy, which is one of the strong

properties of nested multiresolution spaces. An adaptive polygonalization technique for implicitly defined surfaces was proposed by Hall et al. [13]. Their main concern is to obtain polygonal representations of algebraic surfaces, which are appropriate for hardware rendering. Their approach is based on an adaptive tetrahedral mesh refinement algorithm.

2.2 Aspects of Iso-surface Generation and Volume Rendering

A wide variety of algorithms have been developed for the visualization of scalar volume data. For the main classes of surface fitting [18] and direct volume rendering [15] many acceleration techniques were suggested to deal with the huge amount of 3D data.

One approach to accelerate the extraction of iso-surfaces is to preselect the cells that may contribute triangles to the iso-surface. The octree spatial subdivision presented by Wilhelm and Van Gelder [29] annotates each octree node with the span. The sweeping simplices algorithm published by Shen and Johnson [25] uses a binary tree solely based on the relations of the spans. A more sophisticated algorithm of this approach was presented as the ISSUE algorithm by Shen and Johnson [24]. These algorithms generate the same triangles as the standard marching cubes does, but they require extra memory to store the interval information.

Another iso-surface optimization is the reduction of the amount of triangles generated for rendering. This can be done in a post-processing step as in the Schroeder algorithm [22] or interleaved with the marching cubes algorithm as in the approach presented in Shekhar et al. [23]. The reduction decisions are always restricted to a fixed iso-surface, the extraction of which is not optimized in this approach. First steps towards iso-surface generation on reduced tetrahedral grids were made by Cignoni et al. [6].

Traditional ray-casting has been accelerated by adapting the integration step size to the variance in the data [7] and by efficiently stepping through the volume [31]. Other approaches again use hierarchical [17] or compressed [28, 11] representations of the volume.

3 THE FINITE ELEMENT APPROACH

Finite element analysis is a numerical method for solving partial differential equations which is widely used in science and engineering. In this section we present a new method which extends the basic underlying ideas with respect to hierarchical approximation, local mesh refinement and error estimates for solving general approximation problems in Hilbert spaces.

3.1 The Approximation Problem

In order to measure the quality of an approximation of a function defined over a domain Ω we assume that the function to be approximated is an element of a linear function space equipped with a norm, which is induced by an inner product. The approximation problem in a Hilbert space \mathcal{E} with the inner product $(\cdot, \cdot)_{\mathcal{E}}$ can be formulated as follows. Suppose S is a linear subspace of \mathcal{E} . An element $u \in S$ is the *best approximation* to an element $f \in \mathcal{E}$, if the orthogonality condition

$$(f - u, v)_{\mathcal{E}} = 0 \quad \text{for all } v \in S \quad (1)$$

holds. The problem to find the best approximation can of course be reformulated into the well known form of a *least squares approximation*, where the best approximation $u \in S$ is given by

$$\|f - u\|_{\mathcal{E}} = \inf_{v \in S} \|f - v\|_{\mathcal{E}}.$$

More specifically, we work with the *Hilbert* space $L_2(\Omega)$ consisting of all square integrable functions over Ω . The inner product and the induced norm are given by

$$(f, g)_{L_2} = \int_{\Omega} fg dx \quad \text{and} \quad \|u\|_{L_2} = (u, u)_{L_2}^{1/2}$$

We discretize the problem following the Ritz-Galerkin approach. The domain of definition Ω is assumed to be a polygon for 2D meshes or a polyhedron for 3D meshes. We now consider $S \subset L_2$ to be a N -dimensional finite element subspace consisting of all the piecewise *linear* functions with respect to a triangulation \mathcal{T} of Ω . A basis $\{\phi_i\}_{i=1}^N$ of S consists of the *hat* functions which are associated with each vertex. Their support is restricted to the elements of the triangulation containing this vertex. Thus, equation (1) is reduced to the system of linear equations

$$AU = F \quad (2)$$

where

$$A_{ij} = (\phi_j, \phi_i)_{L_2}, \quad F_i = (f, \phi_i)_{L_2} \quad \text{and} \quad u = \sum U_i \phi_i.$$

The evaluation of the inner products and the *assembling* of the corresponding matrices and vectors are carried out using standard finite element techniques. The function f is usually given in discrete form over a grid, e.g. a MRI data set is given on a uniform grid. For the evaluation of the inner products (f, ϕ_i) we interpolate f bilinear or trilinear on structured grids and linear on triangular or tetrahedral grids.

Up to now we have just reformulated the problem of the best approximation for a finite element space equipped with the L_2 inner product into the solution of a linear system. The method we proposed actually generates a hierarchy of nested approximation spaces by adaptive mesh refinement. It consists of the following three steps

- *Mesh Refinement*: We need a mesh refinement algorithm which generates a hierarchy of triangulations of the domain by local refinement operations. The algorithms used will be described in Section 3.2.
- *Error Measurement*: The key ingredient of an adaptive approximation method is an efficient and robust measurement of the approximation error. This method should provide a bound for the overall accuracy of the approximation and give information about the distribution of the error among the individual mesh elements, thus forming the basis of the adaptive local mesh refinement. The error measurement step will be described in Section 3.3.
- *Multilevel Preconditioning*: An efficient iterative method for solving the potentially large linear system (2) will be addressed at the end of Section 4.

3.2 Local Mesh Refinement

A partition \mathcal{T} of a polygonal or polyhedral domain Ω into triangles or tetrahedra is called a triangulation. The mesh refinement process we use is considered a standard method in the adaptive finite element literature [4, 5]. We start with a coarse triangulation \mathcal{T}_0 , and generate a sequence $\mathcal{T}_0, \dots, \mathcal{T}_j$ of increasingly fine triangulations by successive local mesh refinement. Each triangulation in the sequence is required to be *conforming*, i.e. the intersection of two elements consists of a common face or a common edge or a common vertex or it is empty. This condition prevents *hanging nodes*, which are difficult to treat in finite element computations and

problematic for rendering purposes. For the same reasons the triangulation sequence has to be *stable* with respect to some measure of degeneracy. For example, one requires that all interior angles are bounded away from zero which is essential for the stability of the numerical computations. Finally, in order to build a hierarchy of nested spaces the triangulation sequence has to satisfy the *nestedness* condition, which means that an element in a triangulation is obtained by subdividing an element in a coarser triangulation of the sequence. We use an algorithm which combines regular (red) and

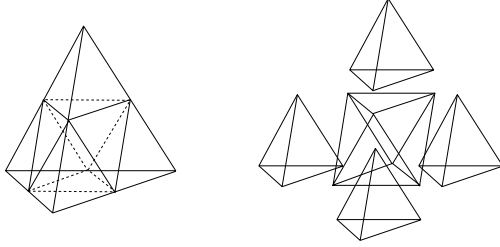


Figure 2: Regular or red refinement of a tetrahedron

irregular (green) mesh refinement as introduced for 2D meshes by Bank et al. [2] and its extension to three dimensional tetrahedral meshes of Bey [4]. These refinement algorithms are carried out in three steps. First, a refinement rule for a single element has to be defined, such that successive refinements produce stable and consistent triangulations. Such a refinement rule is called *red* or *regular*. Second, a set of *green* or *irregular* refinement rules is defined for the elements which share a common edge with regular refined elements. These refinement rules are *local*. Green refined tetrahedra are just inserted to satisfy border conditions and to avoid hanging nodes. In order to avoid stability problems green refined tetrahedra must not be refined again. If a subdivision is required, the originally green refined tetrahedron must be re-refined with the red rule. Finally, these local rules are combined and rearranged into a *global refinement algorithm* which guarantees for stability and conformity.

For 2D meshes the red (regular) refinement rule divides a triangle into four congruent ones by connecting the midpoints of its edges. The green (irregular) refinement consists of simple bisections connecting one edge midpoint with the opposite vertex. The regular refinement rule for tetrahedra first cuts off four sub-tetrahedra at the vertices as shown in Fig. 2. The subdivision of the remaining octahedron is not unique and depends on the choice of one of the three possible diagonals. The strategy proposed by Bey [4] is based on affine transformations to a reference tetrahedron and produces stable regular refinements. For tetrahedra there are $2^6 - 2 = 62$ possible green refinement patterns, which can be classified into 9 different types using symmetry considerations. In order to make the algorithm practicable, green refinement is restricted to the four different types shown in Fig. 3 performing a red refinement on all remaining patterns.

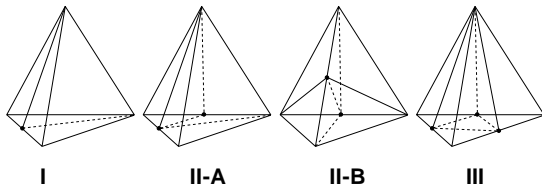


Figure 3: Irregular or green refinement of a tetrahedron

3.3 Error Analysis

In order to obtain information about the accuracy of the approximation, we are interested in an efficient method to measure the error $\|f - u\|$. Because we have *a priori* knowledge of the function to be approximated, we evaluate the error by a direct integration. In order to obtain information on the local contributions to the global error we proceed as follows. Let \mathcal{T} be a triangulation of the domain of definition Ω and Δ an element in \mathcal{T} . The error can be written in the form

$$\begin{aligned} \|f - u\|_{L_2} &= \int_{\Omega} (f - u)^2 d\Omega \\ &= \sum_{\Delta \in \mathcal{T}} E_{\Delta} \end{aligned} \quad (3)$$

where the E_{Δ} are given by

$$E_{\Delta} = \int_{\Delta} (f - u)^2 d\Omega. \quad (4)$$

For the computation of the E_{Δ} coefficients we use a four points third order integration formula for triangles and a five points third order integration formula for tetrahedra [26]. The function values at the integration points are obtained by interpolation.

4 ALGORITHM

If we think of mesh reduction and optimization as an abstract approximation problem in Hilbert spaces we can use powerful methods to adaptively generate a sequence of approximations based on local mesh refinement and error computation. In this section we present an iterative algorithm to generate sequences of nested approximating spaces. After each iteration a new sequence of nested spaces which are obtained by local mesh refinement is derived from the previous hierarchy of spaces. The first step is to construct the initial, intentionally coarse triangulation of the domain. Depending on the geometry of the domain this may be a very complex task. Based on the initial triangulation the first approximation is computed by using a direct solver for the linear system (2). Then, we enter a loop where the mesh is iteratively refined depending on the global accuracy of the approximation. At each step of this process, which we call *data driven* mesh generation, mesh elements are marked for refinement, if their local error coefficients E_{Δ} in (4) exceed a certain threshold. After the mesh refinement step, the new approximation is computed by an iterative multilevel solver. The algorithm is summarized by the following pseudo-code segment:

```

procedure MeshOptimization()
  choose coarse triangulation  $\mathcal{T}_0$ 
  compute first approximation  $u_S$  by direct solver
  while (global error too high)
    for each element
      evaluate local error  $E_{\Delta}$ 
    end for
    for each element
      if  $|E_{\Delta}| > threshold$ 
        mark for refinement
      end if
    end for
    mesh refinement
    compute new approximation  $u_S$ 
  end while
end procedure

```

The result of an iteration step k is a sequence of triangulations $\mathcal{T}_0, \dots, \mathcal{T}_{j_k}$ satisfying the nestedness condition and being used to

construct the hierarchy of nested piecewise linear finite element spaces:

$$S_0 \subset S_1 \subset \dots \subset S_{j_k} \quad (5)$$

The best approximation u_S for the level S_{j_k} is efficiently computed by a multilevel linear solver. At each iteration step a few of the upper level meshes might be modified due to the substitution of irregular refinements by regular ones. Once the solution is known at the finest level of resolution, an approximation to f at the other levels of the hierarchy can be computed by a hierarchical basis decomposition as shown by Yserentant [32].

During the successive refinement steps the number of vertices in the meshes will increase. For a large number of vertices, especially for 3D applications, the linear system (2) will become very large, with the matrix A being sparse but non-diagonal. Standard iterative methods such as Gauss-Seidel or even conjugate gradient iteration, are not appropriate. We solve this linear system (2) with a conjugate gradient method combined with a multilevel BPX preconditioner. The very efficient method makes use of the hierarchy (5) for the solution of the linear system and is successfully applied in many modern finite element computations [1, 5]. All our implementation work for the multilevel iterative solver is based on the C++ object oriented finite element package KASKADE from the Konrad-Zuse-Zentrum in Berlin [3].

Iter.	vertices	triangles	time (s)	ratio
5	63	104	0.01	0.68
15	1,207	2,353	0.12	0.87
20	5,368	10,651	0.94	0.87
25	22,275	44,450	4.98	0.82

Table 1: Processing results for the Australia height fields data set

We illustrate the algorithm by a closer investigation of the mesh optimization process which generated the hierarchy shown in Fig. 1. The initial triangulation \mathcal{T}_0 consists of just two triangles, the inner products are evaluated using bilinear interpolation of the input height field. In Table 1 we give the results corresponding to the best approximation obtained after a given number of iterations of the algorithm. We used a *threshold* of 0.3 and the run times were measured on a SGI Onyx with a 194 MHz MIPS R10000 processor. Important are the number of nodes and triangles, the time required for the solution of the linear system and the ratio between the new number of triangles and the previous one for the finest mesh, which gives an idea of the number of new triangles.

5 ISO-SURFACE EXTRACTION

While we have seen in section 2 that there are many other approaches to optimize 2D meshes, one important aspect of our method is that it easily extends to 3D datasets. In order to demonstrate this we implemented and analyzed an iso-surface generation algorithm which directly operates on the optimized meshes described above. This algorithm inherits all the advantages of the competing approaches by accelerating the iso-surface extraction compared to the marching cubes algorithm as well as by reducing the number of generated triangles. A standard polygon reduction algorithm has to run a marching cubes type algorithm first, and in a second step it has to spend extra time for the reduction process. In our approach the reduction is made on the volume itself, which saves time and memory during the iso-surface generation. On the other hand, any iso-surface extraction accelerator will need extra memory for its internal data structures and it does not reduce the amount of generated triangles. For huge data sets this may become a problem due to the large storage requirements.

In our case, the generation of the iso-surface has to be done by a marching tetrahedra algorithm. There are three main cases of tetrahedral triangle generation. Interpolation in each tetrahedron is done using a linear function of the form

$$f(x, y, z) = a + bx + cy + dz \quad (6)$$

where the coefficients are precomputed from the vertex values by solving a linear system. In order to compute the shading normals of the triangle vertices, the computation of the gradient at the tetrahedral vertices is required. First, we determine the gradient in every tetrahedron, which is constant due to the affine interpolating function (6) within the tetrahedron. The gradient at the vertices of the iso-surface is then computed by the volume-weighted average of the gradients of the adjacent tetrahedra.

6 VOLUME RAY-CASTING

In order to take advantage of the adaptively reduced grid for volume ray-casting, a special algorithm has been developed. The basic idea is to implicitly adapt the integration step size to the size of the tetrahedral elements of the optimized mesh. Therefore, the ray integration is performed from cell face to cell face using the trapezoidal rule. For the two dimensional case this is explained in Fig. 4. One can clearly see, that sample points become denser in regions of smaller tetrahedra, which corresponds to regions of high variance in the data. This sampling technique was also used by Max et al. [21]. Danskin [7] introduced the idea of importance sampling based on octrees. The variance of an octree cell is used to determine the next step size. In our case the step size information is extracted from the data structure itself

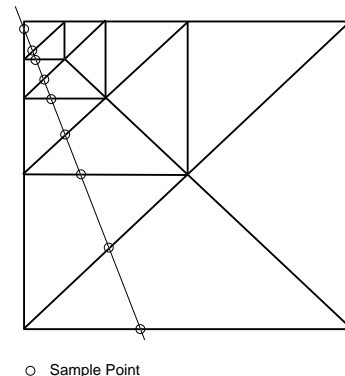


Figure 4: Sampling points for ray integration in a 2D mesh

Since the connectivity of the tetrahedral mesh is generated by the decomposition algorithm, volume traversal can be done very efficiently. Sorting as in scan-line based algorithms [30] is not necessary. The first tetrahedron that is traversed by the ray is computed using an iterative search procedure. Exploiting coherence by means of a bidirectional scanning algorithm in image space, the entrance point is always close to the last one, leaving the search process negligible. For interpolation we use the same linear scheme as in the iso-surface case. Additionally, other acceleration techniques can be integrated. As an example we implemented early ray-termination.

Instabilities during volume-traversal may occur if the ray is exiting a tetrahedron nearby an edge. In this case the wrong neighbor tetrahedron may be chosen. To overcome this problem an extra adaptation procedure has been implemented. First, we check if the stability problem is relevant. In this case, the exit point is moved a certain amount towards the geometric center of the next tetrahedron.

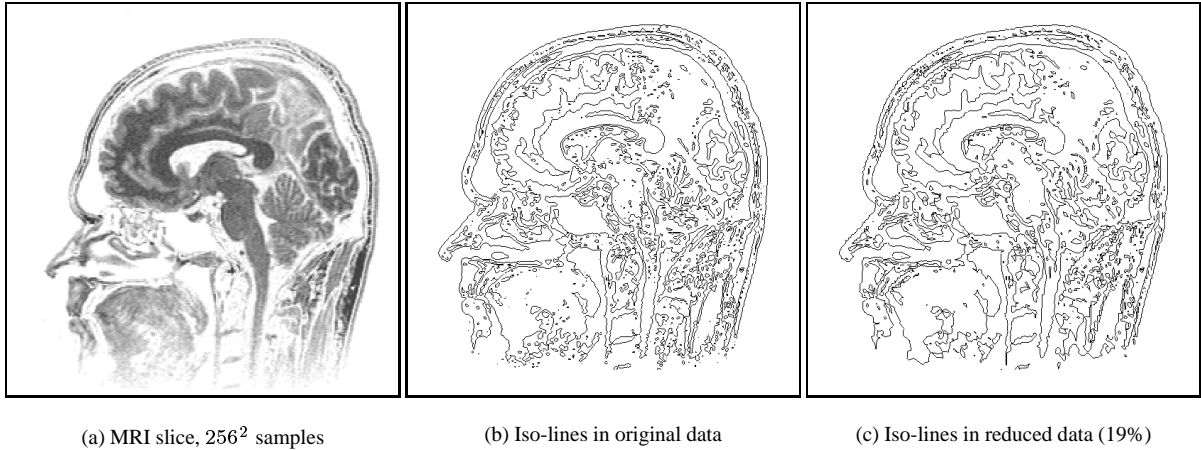


Figure 5: Information contents of the reduced 2D grid

7 RESULTS

The quality of the generated meshes is demonstrated in Fig. 1 where the effect of the local refinement operations can clearly be seen. The quality of the error analysis can be appreciated from the fact that in the outer parts triangles are not refined between the 15th and the 25th iteration. Furthermore, triangles are very *nice* in the sense that no thin triangles or vertices with large number of edges are present in the meshes.

Since it is difficult to depict details from large 3D meshes we demonstrate the quality of the mesh optimization by using a 256^2 slice of a MRI data set shown in Fig. 5(a). In order to check whether the relevant features are still contained in the compressed data, we compute iso-lines. Comparing the structures found in the original image (see Fig. 5(b)) with those computed from the optimized dataset ($threshold = 0.2$) containing only 19% of the vertices (see Fig. 5(c)), we find only very small details missing.

In order to show the strengths of the presented iso-surface algorithm several experiments have been carried out with a MRI head-scan with 128^3 voxels of 16 bit precision, and — as a large data set — a CAT abdomen scan with $(512)^3 * 181$ voxels of 8 bit precision. The generated surfaces show a sufficiently complex structure (see Figs. 7(a) and 7(b)) to judge their quality with respect to surfaces resulting from a standard marching cubes algorithm (Fig. 7(c)). In Figs. 7(d), 7(e) and 7(f) an iso-surface extracted from reduced data sets corresponding to the abdomen is shown. Extraction time, generated triangles and image quality will be of interest.

data set	vertices	tetrahedra	triangles	time (s)
Orig. Head	2,097,152	–	182,280	6.18
Head 1	49,853	286,452	98,113	1.28
Head 2	81,174	470,388	128,786	2.08
Head 3	120,656	700,848	165,834	2.57
Orig. Abd.	47,448,064	–	1,471,254	126.5
Abd. 1	36,892	204,229	103,592	1.2
Abd. 2	61,486	344,217	158,659	1.9
Abd. 3	148,563	845,086	346,573	4.4

Table 2: Iso-surface extraction for the MRI-heads and CAT-abdomens in Fig.7, $threshold = 0.2$

Considering the triangle generation speed with respect to the number of tetrahedra shown in Table 2, we see that the extraction time is nearly constant. An optimal algorithm would just inspect those tetrahedra, that contribute to the iso-surface. Such an algo-

rithm would result in a constant extraction speed for triangles independently of the number of tetrahedra used. Our algorithm does not exhibit constant behavior, but the slope is quite low which means, that the triangulation of the volume implicitly results in an effective processing of the generated tetrahedra.

As we can see in Table 2 the time needed to generate the triangles is decreasing significantly compared to the standard marching cubes algorithm. Looking especially at Head 1 we see, that the number of triangles is about one half and the processing time is about 6 times faster. In the case of the Abdomen 3 (Fig. 7(f)), the number of triangles is reduced to 25% compared with the marching cubes algorithm. Iso-surface extraction has also be done in combination with the sweeping simplices algorithm [25]. However, a maximal extra acceleration of up to 10% in the best case has been achieved. This is because of the adaptive mesh refinement and a cell search technique are not orthogonal acceleration methods.

The generation of the grid hierarchy, on which all the algorithms are based, is done in a preprocessing step. For example, the total computation time including mesh refinement, linear solver and error analysis was 166 sec for the Abdomen 1 (level 11) and 857 sec. for the Abdomen 3 (level 14).

In order to analyze the characteristics of the volume ray-caster several experiments have been performed. Our goal was to analyze the relation of computation time and image quality. For this purpose a chromosome data set consisting of 256^3 voxels has been used. All images are rendered at a resolution of $500 * 500$ pixels. Fig. 7(i) shows the chromosome ray-traced on the original uniform grid using a standard algorithm with two samples per voxel. Figs. 7(g) and 7(h) show the same data set with the same transfer functions for different reduction stages. The number of vertices and tetrahedra left for each data set is given besides the rendering time in Table 3.

data set	vertices	tetrahedra	time (s)
Chrom. 1	2,320	12,570	17
Chrom. 2	4,613	25,571	21
Chrom. 3	9,076	50,817	27
Chrom. 4	19,769	111,891	35
Chrom. 5	34,269	194,885	45
Original	16,777,216	–	197

Table 3: Volume ray-casting for the chromosome data set in Fig. 7, $threshold = 0.2$

The differences between the image obtained from the original data set (Fig. 7(i)) and the image obtained from the reduced data

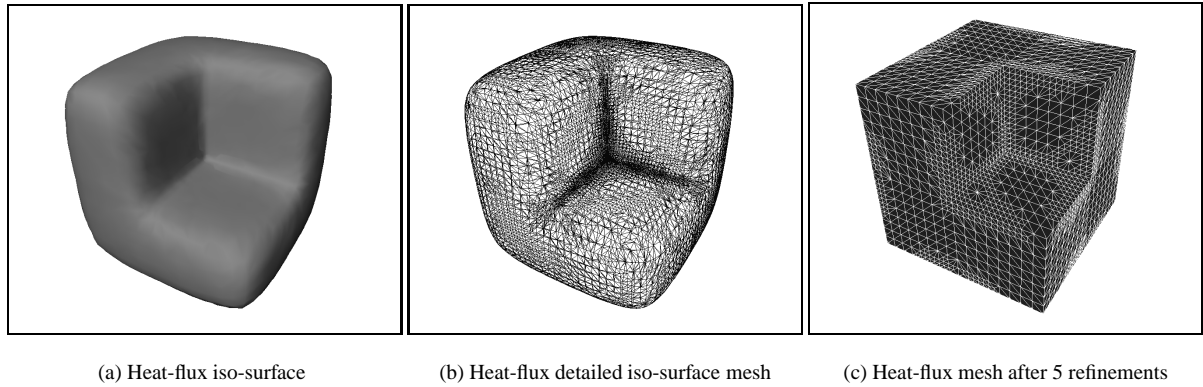


Figure 6: Poisson equation on a cube with a cut corner

with the most tetrahedra left (Fig. 7(h)) is marginal. The number of vertices has been reduced to 0.2%. The ray-tracing time has been reduced down to 23%. These numbers show the strength of the combination of the finite element analysis algorithm with the inherently adaptive ray-casting method. The analysis algorithm tends to finer tetrahedrization of regions with high gradient magnitude. This can be clearly seen at the edge of the chromosome in Figs. 7(g),7(h). This slope degenerates at the lower levels of resolution very quickly. On the other hand the orange kernel, which is the nucleus in the upper left corner of the chromosome, remains quite stable even in levels of lower resolution. In 7(g) the amount of vertices has been reduced down to 0.01% of the original data set, and the required computational time has been reduced down to 8.6% of the original data set. The main characteristics as the orientation of the chromosome and the position of the nucleus are still visible.

At the first glance, the reduction of computation time seems to be quite low, compared with the reduction of cells. But traversing a simplicial mesh is much more complicated due to clipping operations and traversal instabilities. Additional acceleration techniques such as early ray termination have been implemented. The performance gain remains constant as these techniques are orthogonal to our approach.

In order to show the versatility of the method a further data set from an engineering application was visualized, which is defined on a non tensor product grid. It represents the solution of the Poisson equation on a cube with a cut corner and was produced with the finite element code KASKADE [3]. Fig. 6(a) shows the iso-surface corresponding to the iso-value of 0.03 which was computed from an optimized mesh with only 8,076 vertices. The mesh of the same iso-surface computed from a refined grid containing 48,140 vertices is shown in Fig. 6(b). The outside of the underlying 3D mesh can be seen in Fig. 6(c).

8 CONCLUSIONS

We have presented a new method which uses techniques and ideas from the finite element community to solve the problem of mesh optimization for arbitrary unstructured two and three dimensional grids. We have shown, that the algorithm produces adaptively refined grids which are well behaved and suitable for accelerating a broad class of visualization and rendering methods. For the example of iso-surface extraction we have given timing results which demonstrate that the underlying grids allow for a considerable speedup of existing algorithms while maintaining the desired accuracy.

A major advantage of this approach is that the mesh hierarchy is constructed from coarse to fine with the computational complexity of the approximation of the function on the coarse grids being nearly independent of the actual data size. Intermediate levels can be easily reconstructed from the initial mesh based on the well defined refinement rules and a few integers indicating the simplex-ids to be regularly subdivided. This will make the proposed method very attractive for progressive transmission and display across a network. In future work we will exploit this fact by adapting visualization and rendering algorithms to interactive exploration based on local switching between refinement levels according to a user-requested level-of-detail or a view dependent error threshold. Further on, we will extend this work to other 3D visualization algorithms which we envision to profit from the optimized meshes, e.g. particle tracing and vector field topology.

The approximation error used in the mesh reduction process can be defined in any linear Hilbert spaces. The function space L_2 is not always well suited for approximating smooth functions. Functions rapidly oscillating with small amplitudes are in the L_2 sense a good approximation to a given smooth function and reciprocally. However, in order to suppress such oscillating contributions, other norms which include differentiability information can be used. In particular, we are interested in the Sobolev space H^1 equipped with the inner product

$$(u, v)_{H^1} = (u, v)_{L_2} + (\nabla u, \nabla v)_{L_2}.$$

Comparison and analysis (theoretical and experimental) of the results produced by the different norms will be the topic of research of future work.

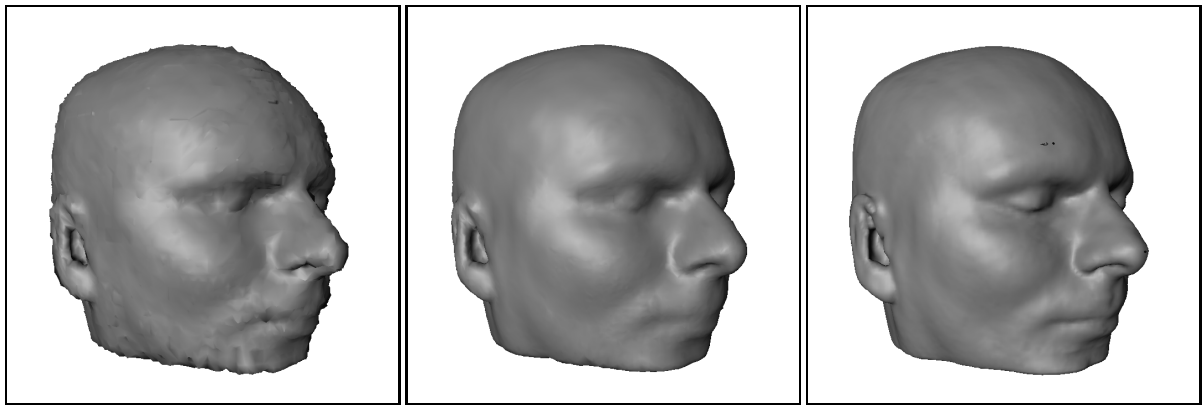
ACKNOWLEDGMENTS

The CAT scan of the male abdomen is a courtesy of the Visualization Laboratory of the State University of New York at Stony Brook. The chromosome data set is a courtesy of the institute of anatomy, University of Giessen. We would like to thank Dr. Günther Greiner for many fruitful discussions

References

- [1] R. E. Bank. Hierarchical Bases and the Finite Element Method. *Acta Numerica*, 1996.
- [2] R. E. Bank, A. H. Sherman, and A. Weiser. Refinement Algorithms and Data Structures for Regular Local Mesh Refine-

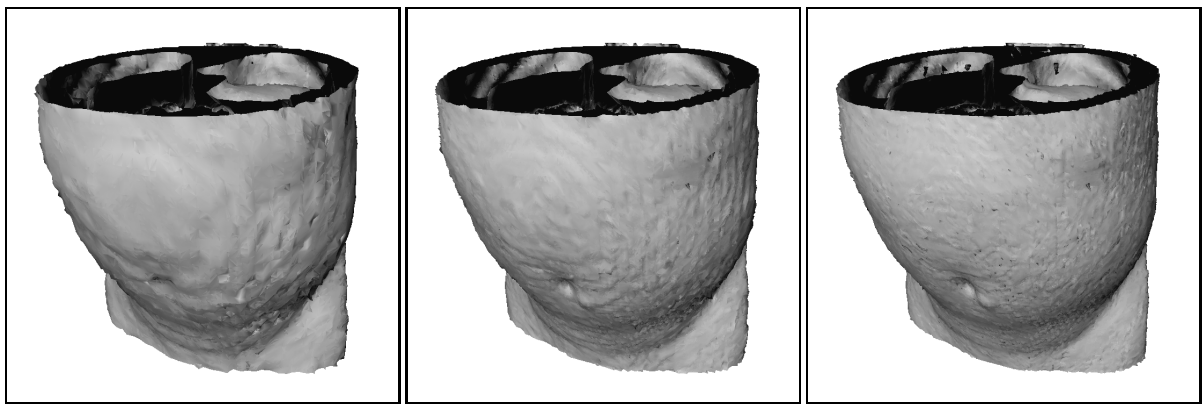
- ment. In R. Stepleman, editor, *Scientific Computing*, pages 3–17, Amsterdam, 1983. IMACS/North Holland.
- [3] R. Beck, B. Erdmann, and Roitzsch R. KASKADE User's Guide. Technical report, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1995.
- [4] J. Bey. Tetrahedral Mesh Refinement. *Computing*, 55:355–378, 1995.
- [5] J. H. Bramble. *Multigrid Methods*. Number 294 in Pitman Research Notes in Mathematical Sciences. Longman Sci. & Techn., Harlow, UK, 1993.
- [6] P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution Modeling and Visualization of Volume Data based on Simplicial Complexes. In *Proceedings 1994 Symposium on Volume Visualization*, pages 19–26, 1994.
- [7] J. Danskin and P. Hanrahan. Fast Algorithms for Volume Ray Tracing. In *Transactions 1992 workshop on Volume Visualization*, pages 91–98, 1992.
- [8] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes. In *Proceedings SIGGRAPH '95*, pages 173–182, 1995.
- [9] M. H. Gross, R. Gatti, and O. Stadt. Fast Multiresolution for Surface Meshing. In *Proceedings IEEE Visualization '95*, pages 135–142, October 1995.
- [10] M. H. Gross, L. Lippert, A. Dreger, and R. Koch. A new Method to Approximate the Volume Rendering Equation Using Wavelet Bases and Piecewise Polynomials. *Comput. & Graphics*, 19(1):47–62, 1995.
- [11] R. Grosso, Th. Ertl, and J. Aschoff. Efficient data structures for volume rendering of wavelet-compressed data. In N. M. Tahlmann and V. Skala, editors, *WSCG '96 - The Fourth International Conference in Central Europe on Computer Graphics and Visualization*, Plzen, Czech Republic, Feb. 1996.
- [12] B. Guo. A Multiscale Model for Structure-Based Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):291–301, 1995.
- [13] M. Hall and J. Warren. Adaptive Polygonalization of Implicit Defined Surfaces. *IEEE Computer Graphics & Applications*, pages 33–42, November 1990.
- [14] H. Hoppe. Progressive Meshes. In *Proceedings SIGGRAPH '96*, pages 99–108, 1996.
- [15] A. Kaufmann. *Introduction to Volume Visualization*. IEEE Computer Society Press, 1991.
- [16] R. Klein, G. Liebich, and W. Strasser. Mesh Reduction with Error Control. In *Proceedings IEEE Visualization '96*, pages 311–318, 1996.
- [17] D. Laur and P. Hanrahan. Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering. *Computer Graphics*, 25(4):285–288, July 1991.
- [18] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, 1987.
- [19] M. Lounsbery, T. DeRose, and J. Warren. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. Technical Report TR 93-10-05b, Department of Computer Science and Engineering, University of Washington, January 1994.
- [20] Ch. Lürig and Th. Ertl. Adaptive Iso-Surface Generation. In B. Girod, H. Niemann, and H.-P. Seidel, editors, *3D Image Analysis and Synthesis '96*, pages 183–190, 1996.
- [21] N. Max, P. Hanrahn, and R. Crawfis. Area and Volume Coherence for Efficient Visualization of 3D Scalar functions. *Computer Graphics*, 24(5):27–33, November 1990.
- [22] W. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of Triangle Meshes. In *Proceedings SIGGRAPH '92*, pages 65–70, 1992.
- [23] R. Shekhar, W. Fayyad, R. Yagel, and J. Fredrick. Octree-Based Decimation of Marching Cubes Surface. In *Proceedings IEEE Visualization '96*, pages 335–342, 1996.
- [24] H.-W. Shen, C. Hansen, Y. Livnat, and C. R. Johnson. Iso-surfacing in Span Space with Utmost Efficiency (ISSUE). In *Proceedings IEEE Visualization '96*, pages 287–294, 1996.
- [25] H.-W. Shen and C. Johnson. Sweeping Simplices: A Fast Iso-surface Extraction Algorithm for Unstructured Grids. In *Proceeding IEEE Visualization '95*, pages 143–150, 1995.
- [26] A. H. Stroud. *Approximate Calculation of Multiple Integrals*. Prentice Hall, 1971.
- [27] G. Turk. Re-tiling Polygonal Surfaces. In *Computer Graphics*, volume 26, pages 55–64, 1992.
- [28] R. Westermann. A Multiresolution Framework for Volume Rendering. In A. Kaufman and W. Krüger, editors, *1994 Symposium on Volume Visualization*, pages 51–58. ACM SIGGRAPH, 1994.
- [29] J. Wilhelms and A. Van Gelder. Octrees for Faster Iso-surface Generation. In *ACM Transactions on Graphics*, pages 201–227, 1992.
- [30] J. Wilhelms, A. Van Gelder, P. Tarantino, and J. Gibbs. Hierarchical and Parallelizable Direct Volume Rendering for Irregular and Multiple Grids. In *Proceedings IEEE Visualization '96*, pages 57–64, 1996.
- [31] R. Yagel and Z. Shi. Accelerating Volume Animation by Space Leaping. In G.M. Nielson and Bergeron D., editors, *Visualization 93*, pages 62–69, Los Alamitos, CA, 1993. IEEE, IEEE Computer Society Press.
- [32] H. Yserentant. Hierarchical Bases. In R. E. O'Malley, editor, *ICIAM 91*. SIAM, Philadelphia, 1992.



(a) Head 1, 2% of the vertices

(b) Head 2, 4% of the vertices

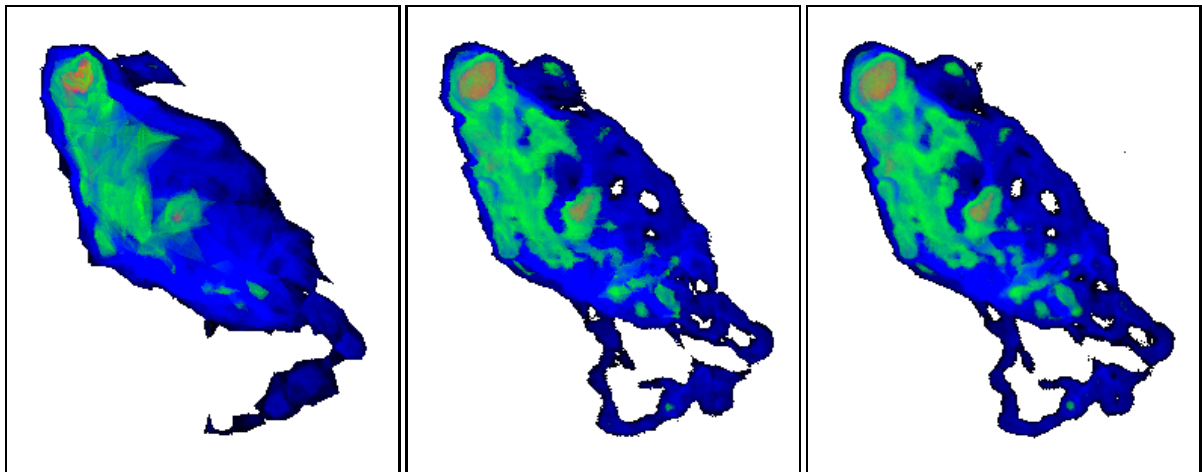
(c) Head generated with marching cubes



(d) Abdomen 1, 0.05% of the vertices

(e) Abdomen 2, 0.16% of the vertices

(f) Abdomen 3, 0.33% of the vertices



(g) Chromosome 1, 0.07% of the cells

(h) Chromosome 5, 1.2% of the cells

(i) Chromosome at full resolution

Figure 7: Image plate showing the results of iso-surface extraction and volume ray-casting