

A distributed environment for integrating volume reduction and implicit adaptive rendering techniques

Christoph Lürig and Thomas Ertl

University of Erlangen, IMMD IX, Computer Graphics Group
Am Weichselgarten 9, D-91058 Erlangen, Germany
Email: {cpluerig,ertl}@immd9.informatik.uni-erlangen.de

Abstract

In this paper we present a distributed environment that allows us to integrate volume reduction techniques using irregular meshes and implicit adaptive rendering methods. Different parts of the system are distributed on up to five workstations. A flexible tool for explorative and adaptive visualization of three dimensional scalar fields is provided. The system combines grid generators and visualization components. The grid generators create an adaptively refined tetrahedral mesh and the visualization components make use of the implicit adaptivity information given by the size of each tetrahedron. The parallelization has been done using asynchronous remote procedure calls. A client process, that handles the graphical interface is communicating the tasks to the four clients, which do the main computation. The client program itself just contains the Motif/Inventor user interface and some routines for storing and retrieving data.

1 Introduction

Volume visualization techniques such as iso-surface generation and volume rendering are very time consuming. This fact is especially usually unpleasant when doing explorative visualization. The adjustment of rendering parameters such as transfer functions, iso-values or view perspectives requires many experiments. To overcome this problem we present a tool, that provides the possibility to reduce computation time by the cost of reducing rendering accuracy. We follow the approach first presented by Cignoni et al. [CdFM⁺94]. The basic idea is to perform some kind of volume data compression and to make use of the implicit adaptivity information generated

during the compression process. The compression is done by describing the original data set using a tetrahedral mesh with fewer cells.

The main topic of this paper focuses on the integration of the necessary grid generation and visualization components into a single distributed system. The advantages of the adaptive volume reduction techniques can only be fully used, if grid generation and rendering components are linked into a compact parallelized system. In this way it is possible to build an explorative visualization tool. First adjustments of rendering parameters can be done on a very coarse resolution that is quickly generated and rendered. During the experimental adjustment of parameters, finer resolutions can be calculated. Finally if a well tuned parameter set is found, a more time consuming and a higher quality computation can be done. All grid generators and visualization components are parallelized. As a consequence, several rendering and grid generation tasks can be computed at the same time. The parallelization was done using an asynchronous remote procedure call as described in Bloomer [Blo92].

According to Brodlie et al. [BCE92] the presented program falls into the category of turnkey visualization systems. From the basic structure it also differs from the data stream oriented systems like Explorer, as the computation of a processing step does not automatically evoke the recomputation of the following steps. In our case this would not be desirable, as the grid generators and the rendering components should also work independently.

The volume reduction is done by generating a tetrahedral mesh, that contains fewer cells than the original data set. In regions of high spatial frequency there are more tetrahedra than in regions of lower frequency. Two approaches are in-

tegrated in our tool as described in one of the following sections.

The tetrahedral meshes generated by one of the two methods are then used to perform implicit adaptive visualization of the analyzed data set. We implemented iso-surface generation and volume-ray-casting. If we run a marching tetrahedra on the adaptively reduced grid, we generate a triangle mesh, where generally the coarser triangles are generated by the larger tetrahedra. The volume ray-caster adapts its step size to the tetrahedra size [LGE97].

2 Architecture

The processing and the visualization of a data set consists of two main steps: The generation of the tetrahedral mesh and the application of the visualization method. The tetrahedral mesh provides a flexible way to describe the data set, since the vertices of the tetrahedra may be chosen at arbitrary positions. Two grid generators and two visualization components have been implemented to accomplish these tasks. This main structure is illustrated in figure 1.

Due to the pipeline character of this problem a coarse granular parallelization is straight forward. But also the alternative branches can be parallelized, if a user wants to try two alternative methods at once. This results in four processes: two processes for the grid generation and two processes for the visualization methods. Finally a fifth process is used to perform the coordination, the display of the result and the data handling.

All partial results appearing in the computation pipeline may be saved or read from disk to avoid multiple computations if the same intermediate results are used in several sessions or if old stages have to be restored during a visualization session. The coordination process performs the data administration including the loading and the storage of the data.

The system is implemented as a client-server application with one client and four servers. The coordination process is the client, which schedules the commands to perform the necessary computations to the servers. The resulting architecture is shown in figure 2.

The communication between the processes is done via asynchronous remote procedure calls. There are two types of calls needed to accom-

plish the data exchange. The first call delivers the data to one of the servers and returns immediately to the calling client without waiting for a result. The second one is a polling call, that collects the data from the server. Basically there are three possibilities how this collection could be performed.

1. The collection could be done via a follow up remote procedure call. In this case the server would start a remote procedure call when finished to deliver the data to a process residing at the client side. This process would deliver the result via interprocess communication to the process, which originally made the request.
2. The client starts a polling try from time to time, to check whether the server has finished its computation job. If this is the case the client collects the computed data from the server.
3. When the server has finished its computation, it displays a message to the user. The user may then initiate the polling request via the user interface.

We have decided to make use of the last option. Event though it seems to be the less comfortable approach for the user at the first glance, it has a big advantage with respect to the first two approaches: The user can decide when to collect the data from which server. This gives him the complete control over all computational steps. This is necessary for instance, if a coarse grid has been generated for a certain dataset and the user is running visualization experiments on this grid. While he is running the experiments, a finer resolution is computed. If the transfer of data to the client would be done automatically, the user would suddenly start the next visualization experiment with the finer data set, even if he did not intend to.

The user interface, that resides at the client side, has been implemented using a mixture of X/Motif and the Inventor library, which provides functionality to display three dimensional objects. The interface without the Inventor Examiner viewer is displayed in figure 5.

The user interface provides controls for data storage and retrieval, for interactions with the clients and for parameter adjustments necessary for the grid generators and visualization components.

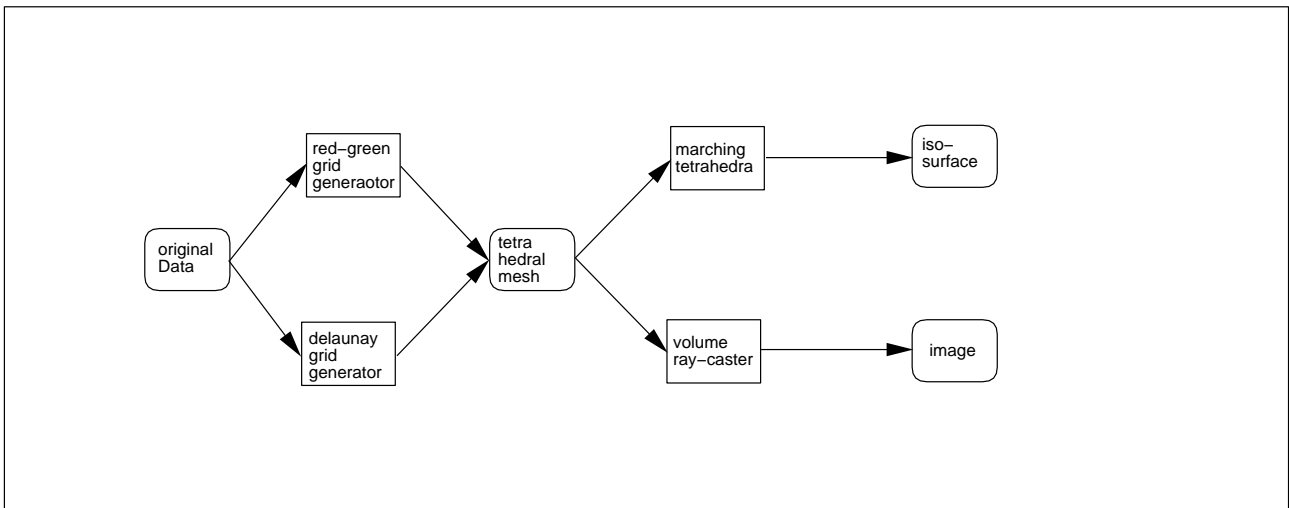


Figure 1: The data flow structure of the system

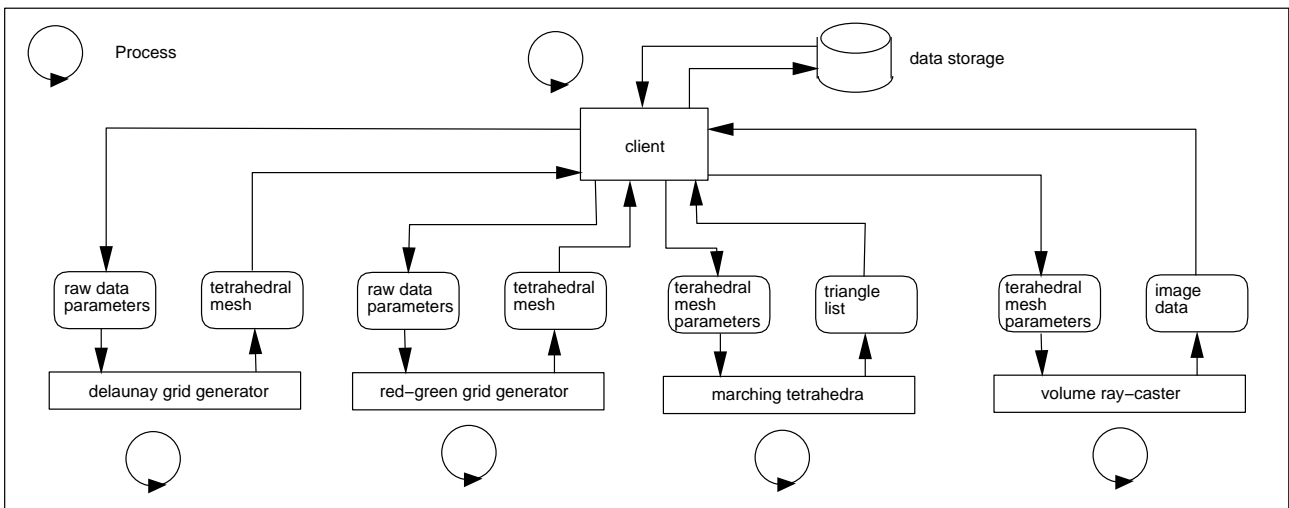


Figure 2: Distribution of the system

In the following sections we will give a more detailed overview of the grid generation and visualization components.

3 Grid Generators

One crucial component for the implicit adaptive visualization system is the grid generator. The basic idea is to generate a tetrahedral mesh with linear interpolation within the tetrahedra, that describes the original volume with fewer cells. The original volume is defined on a uniform grid with trilinear interpolation. Many data sets especially in the field of medical visualization are defined that way. The grid generator itself consists of an error estimator and a component, that generates the cells of the tetrahedral mesh. Be-

sides the possibility for volume data compression, the size of each cell within the mesh may be used for adaptivity decisions during visualization later on.

One implemented approach that has already been presented [LE96] is to generate a list of vertices using an octree error estimator and to perform a Delaunay tetrahedrization afterwards. The error estimator is based on an interpolative criterion using the maximum norm. A progressive refinement is not possible in this case.

The other approach is based on a successive refinement technique, that has been developed in the FEM community (see Bank [Ban96] and Babuska et al. [BZGdAO86]). The implemented approach, that has been presented [GLE97], is based on a red-green refinement triangulator in

combination with a simple least squares error estimator and a BPX-Multilevel preconditioner to solve the least squares approximation equation. In consequence the error estimator is based on an approximation and not on an interpolation criterion. The multilevel preconditioner makes an efficient refinement process possible, as the already computed approximation of the coarser grid serves as a starting point for the computation of the next level.

4 Visualization Components

Once the adaptively reduced grid has been computed, it can be used for implicit adaptive visualization methods. The size of a tetrahedron provides information of the accuracy that has to be applied at this location during the rendering process. We have implemented two tetrahedra orientated algorithms, that make use of this adaptivity information.

The first algorithm generates iso-surfaces. It is an implementation of the marching tetrahedra algorithm. Every tetrahedron may contribute none, one or two triangles to the iso-surface. As in the reduced data set are fewer tetrahedra than in the original one, also the amount of generated triangles is decimated. This affects the iso-surface generation and the rendering time. In subregions with large tetrahedra fewer triangles are generated than in regions with smaller ones.

The second algorithm implemented is an implicit adaptive volume ray-caster. The basic idea is to ray-cast the volume using a step size, that always samples the functions at the faces of the tetrahedra. This sampling technique has been first described by Garrity [Gar90]. As we use an adaptively reduced grid, we get a higher density of sampling points in regions of small tetrahedra than we get in regions of large ones. The sampling technique and its consequences for adaptively reduced grids is shown in figure 3.

5 Application

In order to make use of the explorative visualization, the presented tool is best used in the following way:

1. Compute a coarse tetrahedral mesh for the data set to analyze

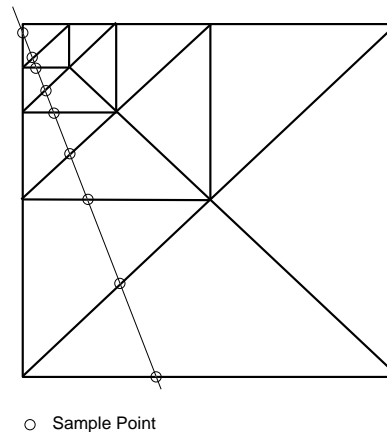


Figure 3: Sampling technique for adaptively reduced grids

2. Start experimenting with the visualization components, meanwhile compute a mesh with finer resolution.
3. When an interesting parameter set and visualization method is found switch over to finer resolutions with higher accuracy and longer rendering times. Keep adjusting your parameters.
4. Save the generated meshes for visualization experiments later on.

This way one can efficiently calibrate rendering parameters for the visualization of the data set.

We have included two examples of implicit adaptive visualization. The first one shown in image 5 shows a volume ray-casting of a chromosome data set at two different resolutions. In the coarser resolution the main features of the data set, like orientation of the chromosome and the nucleus, are still visible.

In the same figure we also show an example of an iso-surface at two different resolutions. Here again the main features of the surface are still visible at the coarser resolution.

6 Conclusions

In this paper we have presented a distributed environment, that integrates grid generation and implicit adaptive visualization components for explorative visualization.

Besides the client, which is responsible for user interaction and data administration, the system contains four servers. Two of them are grid generators and the other two of them are visualization components.

The system itself may be distributed over a workstation cluster with five workstations. The communication between the components is done using two types of remote procedure calls. An asynchronous remote procedure call to initiate the computation and a synchronous one to collect the data. This gives the user the control over when to proceed the visualization process and which step to choose next.

As future work we are currently thinking about compression and progressive transmission methods for irregular grids. If we use the red-green triangulator, we could encode the data set for progressive transmission if we just mark the tetrahedra, that are refined in the next generation with the refinement rule, that has been applied. To avoid the transmission of all of the vertex values for each refinement level, either only the value vertices used in the finest level should be transmitted, or the approximation of the original function should be replaced by an interpolation.

7 Acknowledgment

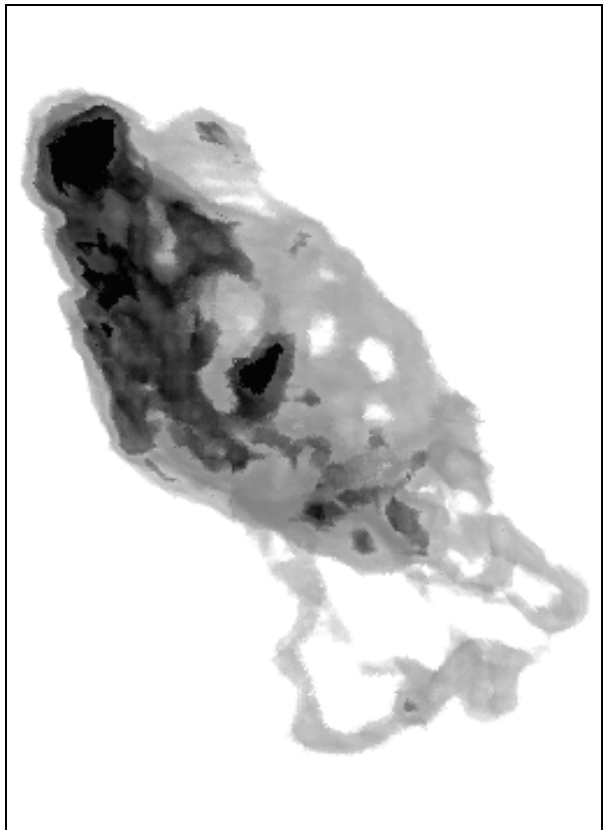
The chromosome data set is a courtesy of the institute of anatomy, University of Giessen. We would like to thank Roberto Grosso for his help in integrating the the red-green triangulator into the system.

References

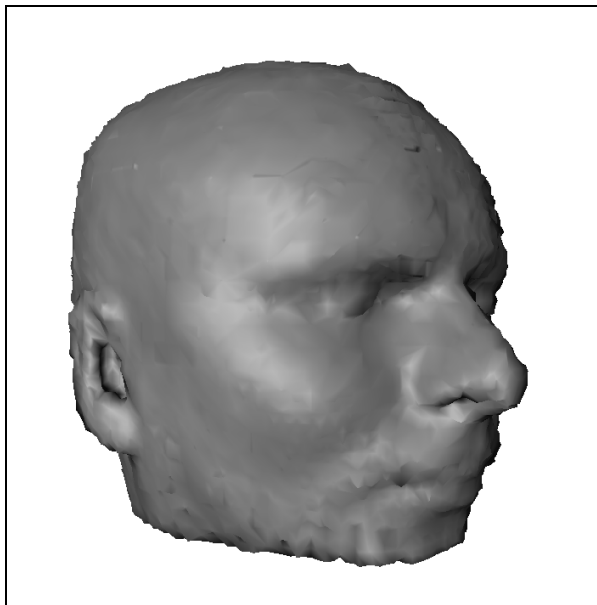
- [Ban96] R. E. Bank. Hierarchical Bases and the Finite Element Method. *Acta Numerica*, 1996.
- [BCE92] K.W. Brodlie, L. Carpenter, and R. A. Earnshaw. *Scientific Visualization*. Springer, 1992.
- [Blo92] J. Bloomer. *Power Programming with RPC*. O'Reilly Associates, Inc., 1992.
- [BZGdAO86] I. Babuska, O. C. Zienkiewicz, J.P. de S.R. Gago, and E.R. de Arantes Oliviera. *Accuray Estimates and Adaptive Refinements in Finite Element Computations*. Wiley, New York, 1986.
- [CdFM⁺94] 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.
- [Gar90] M.P. Garrity. Raytracing irregular volume data. *ACM Computer Graphics*, 24(5), 1990.
- [GLE97] R. Grosso, Ch. Lürig, and Th. Ertl. The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data. In *Proceedings of the Visualization '97 Conference*, Phoenix AZ, U.S.A., October 1997.
- [LE96] C. Luerig and T. Ertl. Adaptive iso-surface generation. In H.-P. Seidel B. Girod, H. Niemann, editor, *3D Image Analysis and Synthesis '96*, pages 183–190. Graduiertenkolleg 3D Bildanalyse und Synthese, infix, 1996.
- [LGE97] C. Luerig, R. Grosso, and T. Ertl. Implicite adaptive volume ray-casting. In *Proceedings Graphicon 97*, pages 114–120, 1997.



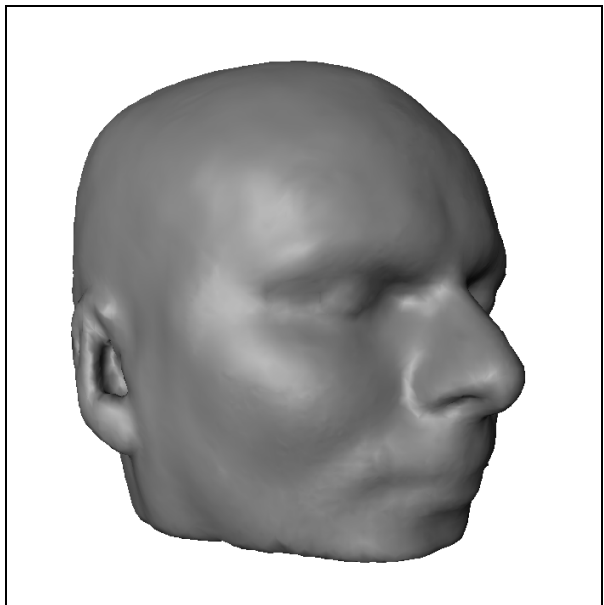
(a) Chromosome at coarse resolution



(b) Chromosome at finer resolution



(c) Iso-surface at coarse resolution



(d) Iso-surface at finer resolution

Figure 4: Example of volume ray-casting

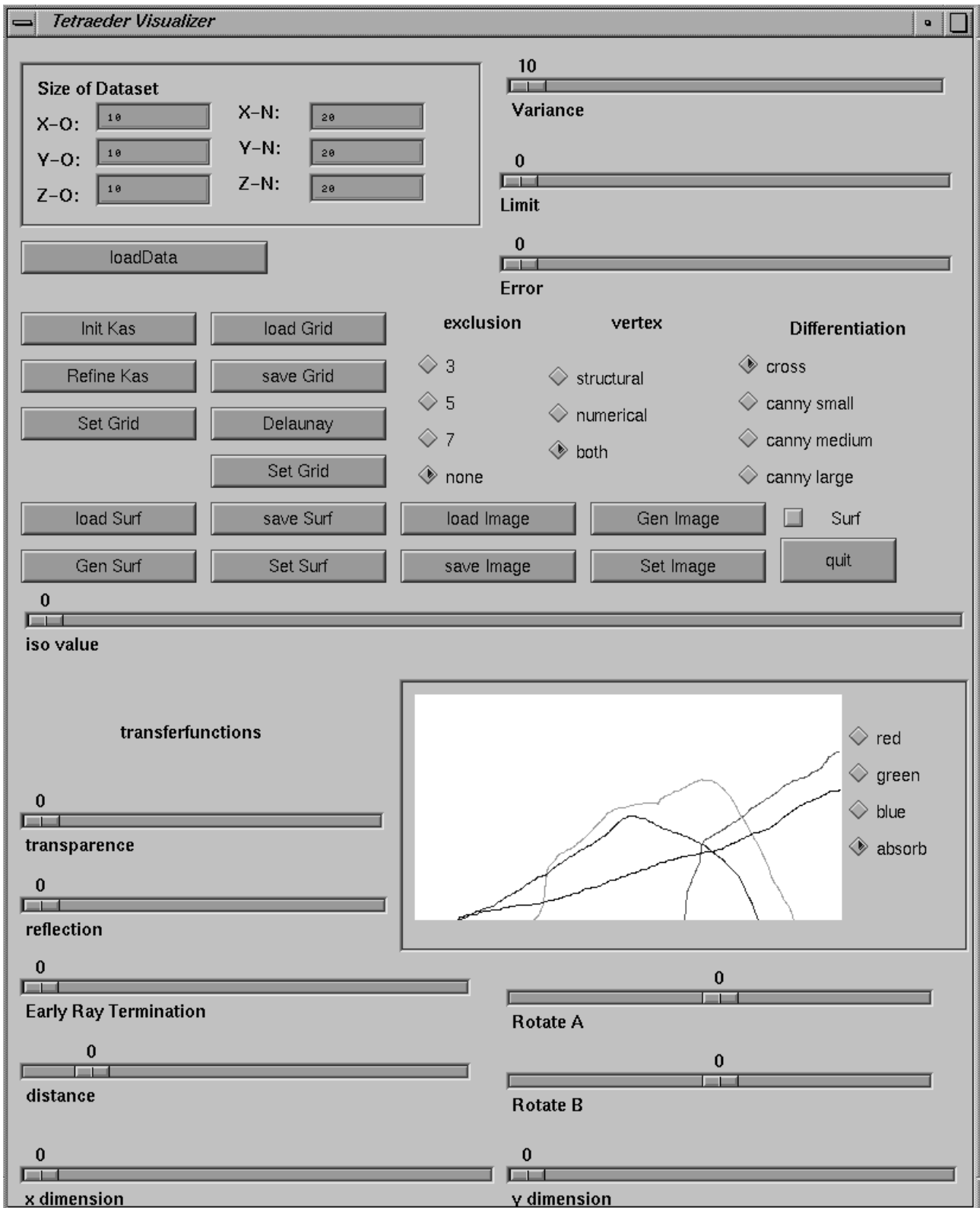


Figure 5: The user interface