Adaptive Multi-valued Volume Data Visualization Using Data-dependent Error Metrics

Jevan T. Gray* Lars Linsen* Bernd Hamann* Kenneth I. Joy*

Center for Image Processing and Integrated Computing (CIPIC)†
Department of Computer Science
University of California, Davis
One Shields Avenue, Davis, CA 95616-8562, U.S.A.

ABSTRACT

Adaptive, and especially view-dependent, volume visualization is used to display large volume data at interactive frame rates preserving high visual quality in specified or implied regions of importance. In typical approaches, the error metrics and refinement oracles used for view-dependent rendering are based on viewing parameters only. The approach presented in this paper considers viewing parameters and parameters for data exploration such as isovalues, velocity field magnitude, gradient magnitude, curl, or divergence. Error metrics are described for scalar fields, vector fields, and more general multi-valued combinations of scalar and vector field data. The number of data being considered in these combinations is not limited by the error metric but the ability to use them to create meaningful visualizations. Our framework supports the application of visualization methods such as isosurface extraction to adaptively refined meshes. For multi-valued data exploration purposes, we combine extracted mapping with color information and/or streamlines mapped onto an isosurface. Such a combined visualization seems advantageous, as scalar and vector field quantities can be combined visually in a highly expressive manner.

KEY WORDS

Multiresolution, error metric, view-dependent visualization, isosurfaces in scalar fields, vector fields, multi-valued data

1 Introduction

Data-intensive applications produce data sets consisting of up to several terabytes. Such large data sets can result from simulating physical phenomena, from digitization with high-resolution scanning devices, or from measuring environments with distributed sensor networks. The generated data sets are typically scalar fields, vector fields, or even multi-valued fields consisting of several scalar and/or vector values per sample point. When a simulated or measured process is changing over time, each time step can consist of terabytes of data.

Multiresolution methods provide a means to deal with large data within acceptable time delays. Data exploration and visualization becomes feasible when a data set is down-sampled to an appropriate level of resolution. Typically, certain regions in a data set are of particular interest to scientists. This fact can be exploited by applying multiresolution methods in an adaptive manner such that regions of interest are represented at higher resolutions. Higher resolution leads to higher precision in terms of approximation error.

For visualization purposes, adaptive settings are common for view-dependent visualization when navigating through data sets in a 3D fly-through-like manner. In view-dependent visualization, regions close to the viewpoint and / or line of sight are represented at relatively higher resolution, while resolution is decreasing when moving away from the viewpoint and / or line of sight. We describe a view-dependent visualization approach in Section 2.

Decisions concerning what regions are to be represented at what level of resolution can automatically be made by applying appropriate resolution oracles and error metrics. Oracles and error metrics used for view-dependent rendering of volume data are currently based on viewing parameters only and mostly restricted to scalar fields [1, 2, 3, 4, 5, 6]. We present an approach based on both viewing parameters and parameters for data exploration. Typical parameters for data exploration are isovalues for scalar fields and velocity magnitude, gradient, curl, and divergence for vector fields. For example, when exploring a scalar field with respect to a certain isovalue, only regions with values close to the isovalue are refined. This approach significantly reduces the amount of data to be processed during visualization. We discuss an error metric for scalar fields in Section 3, an error metric for vector fields in Section 4, and an error metric for multi-valued combinations of scalar and vector field data in Section 5.

The multi-valued error metric framework applies to an arbitrarily high number of combined scalar and vector values. Thus, the number of combinations is not limited by the error metric but the ability to create meaningful visualizations. For multi-valued data visualization, we combine extracted isosurfaces with color information and/or streamlines projected onto isosurfaces. Such a combined visualization is meaningful and effective, as the individual scalar and vector fields are usually correlated. Multi-valued data visualization techniques are described in more detail in Section 6.

2 View-dependent volume visualization

The motivation for view-dependent visualization is that features far from the viewpoint are mapped to few pixels only when projected onto the screen. Small details of such far-away features are often not visible. Thus, using a low-resolution representation of the feature does not impact rendering quality. In view-dependent visualization, regions

*{grayj, hamann, joy}@cs.ucdavis.edu, llinsen@ucdavis.edu
†http://graphics.cs.ucdavis.edu

close to the viewpoint and the line of sight are represented at highest resolution, while resolution is decreasing when moving away from the viewpoint or the line of sight.

For multiresolution data representation, we use a hierarchy of tetrahedra created via longest-edge bisection. The presented method works for regular and irregular tetrahedral meshes. Moreover, it can be adapted to other multiresolution data representations and is independent of the (tetrahedral) subdivision method.

Let $\mathcal{E}(T)$ be an approximation error for an arbitrary tetrahedron T, based on the error metrics described in the following sections. Then, following the approach described in [7], T needs to be subdivided when its error $\mathcal{E}(T)$ is beyond a certain threshold, where the threshold increases with increasing distance from the viewpoint. Let $d(T)$ be the distance from T to the viewpoint, d_{max} the maximum distance from the viewpoint (or the range of sight), and $\mathcal{E}_{max}$ the maximum approximation error. Then, T must be subdivided when

$$\mathcal{E}(T) > \left(\frac{d(T)}{d_{max}}\right)^{\alpha} \mathcal{E}_{max} .$$

The parameter α determines how quickly the resolution decreases with increasing distance. Typically, linear or quadratic decline is used. The parameters d_{max} and $\mathcal{E}_{max}$ are application-specific and user-controlled. For fly-through exploration of a data set, one can restrict subdivision steps to regions within a view frustum, which is defined by the range of sight (d_{max}) and a maximum deviation angle from the line of sight.

3 Error metric for scalar fields

For the definition of the approximation error over a scalar field, various error metrics can be considered. A typical one is the mean-square error that compares the original scalar field to a downsampled approximation of the scalar field by summing squared difference values at all original sample points. Given the original trivariate scalar function F sampled at discrete locations, the mean-square error for a tetrahedron T is defined as

$$\mathcal{E}_{MS}(T) = \frac{1}{|T|} \sum_{\mathbf{x} \in T} \big(F(\mathbf{x}) - f(\mathbf{x})\big)^2 ,$$

where $|T|$ denotes the volume spanned by T and $f(\mathbf{x})$ the value at $\mathbf{x}$ linearly interpolated from the values at the vertices of T. The approximation error is summed over all sample values of F at vertices $\mathbf{x}$ that lie in T. If the use of a root-mean-square error is preferred, this can be accomplished by using the mean-square error and doubling the parameter α of the previous section. One can obtain a screen-space error by projecting $\mathcal{E}_{MS}(T)$ onto the screen.

Figures 2(a) and (b) show a view-dependent visualization using the mean-square error metric. It is applied to a data set representing a distance field induced by a sphere. We recognize that the adaptively refined meshes in Figure 2(a) and (b) show the same characteristics in terms of adaptivity, even though we focused on different features when exploring the data set. In Figure 2(a), we explored the data set with respect to the isovalue 63, whereas in Figure 2(b), we used the isovalue three. In Figure 2(b), the isosurface extraction algorithm had to traverse many tetrahedra close to the viewpoint, although the isosurface is not present in that region. Thus, it is possible to save a significant amount of computation time by refining the mesh only in regions with values close to the chosen isovalue.

This observation leads to a data-dependent definition for an error metric: Let v_{iso} be an isovalue, $[v_{min}, v_{max}]$ be the range of values of the scalar field, and $v_\delta = |v_{max} - v_{min}|$. Then, we define the error for a scalar field over a tetrahedron T with respect to the isovalue v_{iso} as

$$\mathcal{E}_s(T) = \frac{1}{|T|} \sum_{\mathbf{x} \in T} \Big(g\big(v_{iso} - F(\mathbf{x})\big) - g\big(v_{iso} - f(\mathbf{x})\big)\Big)^2 ,$$

where the function $g : \mathbb{R} \rightarrow \mathbb{R}$ has to satisfy the following conditions within $[-v_\delta, v_\delta]$:

- g is continuous,
- $g \geq 0$,
- $g' < 0$, and
- $g'' > 0$.

The simplest functions satisfying these conditions are functions of the form $g(x) = (v_\delta - x)^\beta$, where $\beta \in \mathbb{N}$, $\beta \geq 2$. More complicated functions could be used instead, but we observed that these simple functions suffice to produce the desired results. Figures 2(c) and (d) show how the tetrahedral subdivision steps adapt to the chosen isovalue when using our data-dependent error metric.

To make a visualization process more interactive, error values are usually precomputed for every tetrahedron and loaded during runtime. It is not practical to precompute the error values for all possibly interesting isovalues v_{iso}. However, the expression $\mathcal{E}_s(T)$ can be converted to an expression that allows one to perform most of the computation during preprocessing.

For example, when using a quadratic function $g(x) = (v_\delta - x)^2$, the error $\mathcal{E}_s(T)$ can be rewritten as

$$\mathcal{E}_s(T) = \gamma_0 + \gamma_1 v_{iso} + \gamma_2 v_{iso}^2 ,$$

where

$$\gamma_0 = \frac{1}{|T|} \sum_{\mathbf{x} \in T} \Big(\big(v_\delta + F(\mathbf{x})\big)^2 - \big(v_\delta + f(\mathbf{x})\big)^2\Big)^2 ,$$

$$\gamma_1 = -\frac{2}{|T|} \sum_{\mathbf{x} \in T} \Big(\big(v_\delta + F(\mathbf{x})\big)^2 - \big(v_\delta + f(\mathbf{x})\big)^2\Big)\big(F(\mathbf{x}) - f(\mathbf{x})\big) ,$$

and

$$\gamma_2 = \frac{4}{|T|} \sum_{\mathbf{x} \in T} \big(F(\mathbf{x}) - f(\mathbf{x})\big)^2 .$$

The values of γ_0, γ_1, and γ_2 are independent of v_{iso} and can be precomputed. From these precomputed terms, the data-dependent error $\mathcal{E}_s(T)$ can be computed efficiently.

4 Error metric for vector fields

For vector fields, a single meaningful parameter for data exploration, such as isovalue for scalar fields, does not exist. Thus, when exploring a vector field data set usually several parameters, such as vector magnitude, gradient, curl, and divergence are used. We define an individual error metric for each of these parameters. The overall error metric is defined by a weighted sum of the individual error metrics.

Let $\mathbf{F} = (F_1, F_2, F_3)$ be a trivariate function defining a vector field at sample points $\mathbf{x}$, and v_l be a specific vector magnitude chosen by a user. Then, we define an error with respect to the vectors' magnitude for a tetrahedron T as

$$\mathcal{E}_l(T) = \frac{1}{|T|} \sum_{\mathbf{x} \in T} \left(g\big(v_l - \|\mathbf{F}(\mathbf{x})\|\big) - g\big(v_l - \|\mathbf{f}(\mathbf{x})\|\big) \right)^2 ,$$

where $\mathbf{f}(\mathbf{x})$ denotes the vector at $\mathbf{x}$, linearly interpolated from the vector values at the vertices of T.

From the vector field, we can also derive a gradient field, a curl field, and a divergence field. We use the following definitions from [8]:

$$\begin{aligned} \nabla \mathbf{F} &= \left(\frac{\partial \mathbf{F}}{\partial x}, \frac{\partial \mathbf{F}}{\partial y}, \frac{\partial \mathbf{F}}{\partial z} \right) , \\ \text{curl } \mathbf{F} &= \left(\frac{\partial F_3}{\partial y} - \frac{\partial F_2}{\partial z}, \frac{\partial F_1}{\partial z} - \frac{\partial F_3}{\partial x}, \frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} \right) , \\ \text{div } \mathbf{F} &= \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} + \frac{\partial F_3}{\partial z} . \end{aligned}$$

The curl measures the vorticity or "swirliness" of a vector field. The divergence measures the rate of expansion per volume unit, i. e., the difference in inflow and outflow per unit. Divergence is positive for expanding and negative for compressing vector/flow fields.

The gradient and the curl are represented by a tensor and a vector field, respectively. When using their magnitudes, we can define the error metrics $\mathcal{E}_\nabla(T)$ and $\mathcal{E}_{curl}(T)$ for a tetrahedron T analogously to the error metric $\mathcal{E}_l(T)$. The divergence is represented by a scalar field, and we can define the error metric $\mathcal{E}_{div}(T)$ analogously to $\mathcal{E}_s(T)$. The overall error for a vector field over a tetrahedron T is defined as

$$\mathcal{E}_v(T) = a_1\, \mathcal{E}_l(T) + a_2\, \mathcal{E}_\nabla(T) + a_3\, \mathcal{E}_{curl}(T) + a_4\, \mathcal{E}_{div}(T) ,$$

where $a_1, \ldots, a_4$ are user-defined weights.

5 Error metric for multi-valued volume data

In many simulated or measured data sets, several variables are of interest leading to a multi-valued volume data set, where several scalar and/or vector values are stored for each vertex of a mesh. These values are often correlated. More insight can be gained by exploring several values simultaneously. Thus, a single error metric $\mathcal{E}_m$ needs to be defined, on which the decisions for view-dependent visualization are based.

Let $F_{s,1}, \ldots, F_{s,k}$ be trivariate scalar functions and $\mathbf{F}_{v,1}, \ldots, \mathbf{F}_{v,l}$ be trivariate vector functions defining a multi-valued volume data set. We can derive errors $\mathcal{E}_{s,1}, \ldots, \mathcal{E}_{s,k}$ for the individual scalar fields according to Section 3 and errors $\mathcal{E}_{v,1}, \ldots, \mathcal{E}_{v,l}$ for the individual vector fields according to Section 4. We define the error $\mathcal{E}_m(T)$ for a multi-valued volume field over a tetrahedron T as

$$\mathcal{E}_m(T) = \sum_{i=1}^{k} b_i\, \mathcal{E}_{s,i}(T) + \sum_{i=1}^{l} b_{k+i}\, \mathcal{E}_{v,i}(T) ,$$

where the coefficients b_i, $i = 1, \ldots, k+l$, are used for normalization. To make the influence of all scalar and vector fields equal, we can set

$$b_i = \frac{1}{v_{\delta,i}^{2\beta}} , \qquad i = 1, \ldots, k+l .$$

6 Visualization techniques

Various volume visualization techniques for scalar fields exist. Two common ones are volume rendering and isosurface extraction. We focus on the latter, since it can be combined with other visualization methods.

The most popular algorithm for extracting isosurfaces is the marching-cubes algorithm [9], which was originally developed for structured rectilinear hexahedral grids. We use a similar algorithm, marching tetrahedra [10], which has the advantage of not producing cracks. The marching-tetrahedra algorithm is also more general, since it is applicable to regular and irregular tetrahedral meshes.

To visualize two scalar fields simultaneously, one can, for example, extract an isosurface of one scalar field and color the isosurface with respect to the other scalar field. One could use an RGB-color mapping from the range $[v_{min}, v_{max}]$ of the values of the second scalar field to RGB values. Since lighting can affect color saturation and value, we use an HSV color model instead and map scalar values to hue only. Saturation and value are kept constant for color mapping and can be used for lighting effects. If the hue is uniquely defined by a one-to-one mapping, the mapping is invertible, even when lighting is applied.

A simple linear function can be used for the color mapping from function value range $[v_{min}, v_{max}]$ to range $[H_{min}, H_{max}]$ of hue, shown in Figure 1 (*left*). If we want to emphasize a certain value $v_d \in [v_{min}, v_{max}]$, we provide a wider color spectrum for an interval close to value v_d. An example for such an "emphasizing" color mapping is shown in Figure 1 (*right*).

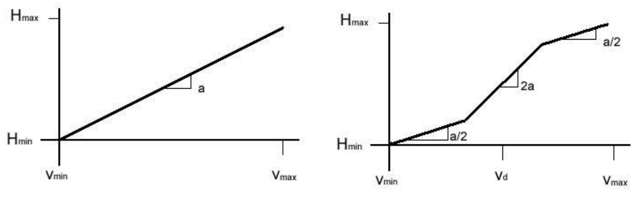

Figure 1. Mapping scalar function value to hue: Linear color mapping (*left*). "Emphasizing" color mapping to focus on a specific value v_d (*right*).

In vector field visualization, the commonly used techniques are streamlines [11] and streamribbons [12]. We make use of streamlines. Streamlines (in the case of a steady flow field) are tangential curves of the given field and correspond to the paths of massless particles. Streamlines are generated by tracing particles using numerical integration techniques, such as the Runge-Kutta or Euler methods.

To facilitate the analysis of relationships between multiple values of a data set combining scalar and vector fields, we extract and display isosurfaces with respect to a scalar field and streamlines (restricted to that isosurface) associated with a vector field. Thus, the flow of a vector field on an isosurface can be examined. When streamlines are projected onto a surface, information about the flow motion relative to the surface (displacement) can be encoded in the coloring of the streamline. Values are mapped to hues based on direction (inflow or outflow) and magnitude.

For the computation of streamlines on an isosurface, we construct a triangle mesh from the collection of triangles generated by the marching-tetrahedra algorithm. Surface streamlines are generated using a fourth-order Runge-Kutta method: For the velocity vectors used by the Runge-Kutta method, we determine the vectors at the requested locations and project them onto the isosurface. The streamline advances to a new location using the projected velocity vectors. After each iteration step of the Runge-Kutta method, we project the new location orthogonally onto the surface. All streamlines are represented by polygons whose vertices all lie on the extracted isosurface. The seed points for the streamline generation are selected interactively by clicking on points on the isosurface.

Other visualization techniques (like texturing) could be combined with the ones we are using, but, depending on the characteristics of the multi-valued data set, the generated images can become extremely complex when displaying too much information.

7 Results

Figure 3 compares a view-dependent visualization of a scalar field using our data-dependent error metric and a scalar-field visualization using the mean-square error. The data set is a CT scan of a Bonsai tree.[1] The size of the data set is 256^3, and the range of the values is $[0, 255]$. Figure 3(a) shows an isosurface (isovalue 85) extracted from an adaptively refined mesh, where the refinement decisions were made with respect to the mean-square error $\mathcal{E}_{MS}$. Figure 3(b) shows the same isosurface based on our data-dependent error metric $\mathcal{E}_s$ for scalar fields. Although both visualizations produce images of about equal quality, the adaptively refined mesh in Figure 3(b) consists of significantly less tetrahedra. Since the computation time for the isosurface extraction is linear in the number of tetrahedra to be traversed (with negligible overhead), the extraction time is reduced to about 54%.

Table 1 lists the numbers of tetrahedra and triangles used for the generation of the geometry shown in Figures 2 and 3. It also lists the approximation errors $\mathcal{E}_{\text{Hausdorff}}$ between the shown isosurfaces and the isosurfaces extracted at highest resolution. We computed the root-mean-square errors between the surfaces with respect to the symmetric Hausdorff distance using the MESH toolkit [13]. We generated surfaces of equal visual quality using error metrics $\mathcal{E}_{MS}$ and $\mathcal{E}_s$. The extracted surfaces are represented at about the same level of resolution (indicated by the number of generated triangles) with about the same approximation quality (indicated by $\mathcal{E}_{\text{Hausdorff}}$): In Figures 2(b) and (d), the extracted surfaces are identical (with their Hausdorff distance being zero); in Figures 2(a) and (c) and in Figure 3, the differences between the extracted surfaces are not noticeable. In general, we observed that for isosurfaces being of nearly the same quality, the error metric $\mathcal{E}_s$ leads to significantly less tetrahedra; when extracting isosurfaces from the same amount of tetrahedra, the quality of the isosurfaces was significantly higher when using $\mathcal{E}_s$. How much benefit is gained depends on the data set and, even more so, the viewpoint chosen for view-dependent refinement.

The number of tetrahedra can further be reduced, if we store for each tetrahedron T in a multiresolution mesh hierarchy the range $[v_{T,min}, v_{T,max}]$ of the function values that appear at vertices lying in the interior or on the boundary of T. Using this information, we can determine for the isovalue v_{iso} whether it falls into that range, i. e., whether $v_{iso} \in [v_{T,min}, v_{T,max}]$ holds. If it does not fall into that range, we can skip the subdivision of the tetrahedron T regardless of error. This additional check requires us to store two additional values per tetrahedron, but it typically reduces the number of tetrahedra that need to be traversed for visualization purposes by up to 50%. The rate, again, heavily depends on the data set and the viewpoint.

In Figure 4, we have applied error metric $\mathcal{E}_v$ to a vector field representing a tornado-like data set. The data set was generated by Crawfis and Max [14] to illustrate flow patterns in 3D flow fields. We set the weights a_2 and a_4 to zero, such that gradient and divergence had no impact. The weights a_1 and a_3 were used to balance the impact of vector magnitude and curl. The vector magnitudes lie between 0.02 and 0.32; and the curl magnitudes lie between zero and 0.24. For the computation of the error $\mathcal{E}_v$, we considered the value 0.15 of vector magnitude and the value 0.15 of curl magnitude. To visualize vector magnitude and curl, we extracted an isosurface of the (scalar) field representing vector magnitude and used hue mapping on the isosurface for visualizing curl magnitude. The isosurface was extracted for the isovalue 0.15; the used color map blends from red to green.

For the generation of Figure 5, we combined two scalar fields and applied the multi-valued error metric $\mathcal{E}_m$. The two scalar fields represent two brain data sets, a human and a monkey brain.[2] The original data sets were obtained from cryosections and have dimensions $1050 \times 970 \times 753$ (human) and $3008 \times 1960 \times 1501$ (monkey). We resampled them to superimpose the meshes such that two scalar values were stored at each vertex of the underlying (resampled) mesh. Moreover, the data sets contained RGB-color information, which we converted to an HSV-color representation in order to generate scalar fields using the value V. The scalar field representing the human brain was used to extract an isosurface, and the scalar field representing the monkey brain was used for color-mapping to hues from green to red. The isovalues we considered for data exploration and used for the data-dependent error metrics were 78 for both human and monkey brain. The colors indicate where and how much the brains differ. We did not perform any pre-alignment of the data sets. When the brains are aligned, this framework can be used to compare and find differences in brains of certain species or to gain insight into mental diseases (when comparing a diseased brain and a healthy brain).

Considering Figure 6, we applied the multi-valued error metric $\mathcal{E}_m$ to a combination of scalar and vector fields. The data set represents the evolution of an Argon bubble disturbed by a shock wave.[3] We used one time step with spatial resolution $640 \times 256 \times 256$. For each vertex, scalar values (between 1.34 and 3.93) for density, scalar values (between zero and 3.66) for percentage of argon inside a cell, and vector values for momentum are stored. The scalar and vector errors $\mathcal{E}_s$ and $\mathcal{E}_v$ were combined in the error $\mathcal{E}_m$ we used for view-dependent refinement. For Figure 6(a), we used density and momentum, and for Figure 6(b),

[1]Data set courtesy of S. Roettger, Abteilung Visualisierung und Interaktive Systeme, University of Stuttgart, Germany

[2]Data sets courtesy of E.G. Jones, Center for Neuroscience, University of California, Davis, and A. Toga, Ahmanson-Lovelace Brain Mapping Center, University of California, Los Angeles

[3]Data set courtesy of The Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, see http://seesar.lbl.gov/ccse

error metric	sphere (isovalue 63)			sphere (isovalue 3)			Bonsai tree		
	tetrahedra	triangles	$\mathcal{E}_{\text{Hausdorff}}$	tetrahedra	triangles	$\mathcal{E}_{\text{Hausdorff}}$	tetrahedra	triangles	$\mathcal{E}_{\text{Hausdorff}}$
$\mathcal{E}_{MS}$	137,048	17,296	0.09048	198,086	768	0.00116	4,515,148	534,298	2.06516
$\mathcal{E}_{s}$	109,970	17,034	0.09165	132,098	768	0.00116	2,452,956	535,044	2.04929

Table 1. Results for $\mathcal{E}_{MS}$ and $\mathcal{E}_s$ - numbers of generated tetrahedra and triangles.

we used density, percentage of argon inside a cell, and momentum.

Figure 6(a) shows a density isosurface (isovalue 1.596) and streamlines on the isosurface using momentum. For the coloring of the streamlines, we mapped the flow relative to the isosurface (displacement) to hues from blue to orange. Blue indicates that the flow velocity vector is directed toward the inside of the isosurface, and orange indicates that it is directed toward the outside. For the vector field, we considered the values 0.05 of vector magnitude, 0.05 of gradient magnitude, 0.05 of curl magnitude, and 0.05 of divergence, which were used in the error $\mathcal{E}_v$. The weights $a_1, \ldots, a_4$ were all 0.25.

For Figure 6(b), we added percentage of argon inside a cell as a second scalar field. It is visualized by color mapping the isosurface with values from green to blue. For the streamlines we used colors from red to yellow. For density and momentum, we considered the same parameters as in Figure 6(a); for percentage of argon inside a cell, we considered the isovalue 0.2. All examples provided in this paper are based on quadratic decrease of resolution with increasing distance from the viewpoint ($\alpha = 2$) and a cubic function g ($\beta = 3$).

8 Conclusions and future work

We have presented a data-dependent error metric for scalar, vector, and multi-valued volume data. Besides viewing parameters, our approach takes parameters for data exploration into account. As parameters for data exploration, our method considers isovalues for scalar fields and magnitude, gradient, curl, and divergence for vector fields. We developed a multi-valued error metric applicable to any combination of scalar and vector fields. Even though our method considers the data exploration parameters, most of the error computations can be done in a preprocessing step.

We have demonstrated the benefits of data-dependent error metrics for view-dependent visualization. For implementation purposes, we used tetrahedral meshes and constructed a tetrahedral mesh hierarchy using longest-edge bisection. However, our methods are independent of grid structure and subdivision scheme. For a constant number of tetrahedra, data-dependent error metrics produce higher-quality images. For a constant image quality, data-dependent error metrics require less subdivision steps and thus less tetrahedra. We conclude that using data-dependent error metrics can lead to a significant speed-up when visualizing volume data with respect to given error bounds and/or time constraints.

For visualizing multi-valued data sets consisting of multiple scalar and vector fields, our approach combines isosurfaces, hue mapping to isosurfaces, and (colored) streamlines restricted to isosurfaces. We plan to extend our method to include also tensor fields.

Acknowledgments

This work was supported by the National Science Foundation under contracts ACI 9624034 (CAREER Award) and ACI 0222909, through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the National Institute of Mental Health and the National Science Foundation under contract NIMH 2 P20 MH60975-06A2; and the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159. We thank the members of the Visualization and Graphics Research Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis.

References

[1] Benjamin Gregorski, Mark A. Duchaineau, Peter Lindstrom, Valerio Pascucci, and Kenneth I. Joy. Interactive view-dependent rendering of large isosurfaces. In Robert Moorhead, Markus Gross, and Kenneth I. Joy, editors, *Proceedings of the IEEE Conference on Visualization 2002*, pages 475–482. IEEE, IEEE Computer Society Press, 2002.

[2] Roberto Grosso and Grzegorz Soza. Real-time exploration of scalar data on multilevel meshes. In G. Greiner, H. Niemann, T. Ertl, B. Girod, and H.-P. Seidel, editors, *Proceedings of Vision, Modeling, and Visualization 2002 (VMV 2002)*, 2002.

[3] Zhiyan Liu, Adam Finkelstein, and Kai Li. Progressive view-dependent isosurface propagation. In D. Ebert, J. M. Favre, and R. Peikert, editors, *Proceedings of the Joint Eurographics-IEEE TCVG Symposium on Visualization (VisSym-01)*, pages 223–232. Springer-Verlag, 2001.

[4] Yarden Livnat and Charles Hansen. View dependent isosurface extraction. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *Proceedings of IEEE Conference on Visualization 1998*, pages 175–180. IEEE, IEEE Computer Society Press, 1998.

[5] Vijaya Ramachandran, Xiaoyu Zhang, and Chandrajit Bajaj. Paralel and out-of-core view-dependent isocontour visualization. In David Ebert, Pere Brunet, and Isabel Navaz, editors, *Proceedings of the Joint Eurographics-IEEE TCVG Symposium on Visualization (VisSym-02)*. Springer-Verlag, 2002.

[6] Rüdiger Westermann, Leif Kobbelt, and Thomas Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, pages 100–111, 1999.

[7] Lars Linsen and Hartmut Prautzsch. Fan clouds – an alternative to meshes. In T. Asano, R. Klette, and Ch. Ronse, editors, *Geometry, Morphology, and Computational Imaging, (Proceedings of Dagstuhl Seminar 02151 on Theoretical Foundations of Computer Vision)*, LNCS 2616 Theoretical Foundations of Computer Vision. IEEE, Springer-Verlag, 2003.

[8] Jerrold E. Marsden and Anthony Tromba. *Vector Calculus*. W.H. Freeman, New York, 4th edition, 1996.

[9] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM Press, 1987.

[10] S. Chan and E. Purisima. A new tetrahedral tessellation scheme for isosurface generation. *Computer and Graphics*, 22(1):83–90, 1998.

[11] Jarke J. van Wijk. Flow visualization with surface particles. *IEEE Computer Graphics and Applications*, 13(4):18–24, 1993.

[12] Shyh-Kuang Ueng, Christopher Sikorski, and Kwan-Liu Ma. Efficient streamline, streamribbon, and streamtube constructions on unstructured grids. *IEEE Transaction on Visualization and Computer Graphics*, 2(2):100–110, 1996.

[13] Nicolas Aspert, Diego Santa-Cruz, and Touradj Ebrahimi. Mesh: Measuring errors between surfaces using the hausdorff distance. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 1, pages 705–708. IEEE, IEEE Computer Society Press, 2002.

[14] Roger A. Crawfis and Nelson Max. Texture splats for 3d vector and scalar field visualization. In Gregory M. Nielson and Dan Bergeron, editors, *Proceedings of IEEE Conference on Visualization 1993*, pages 261–266. IEEE, IEEE Computer Society Press, 1993.

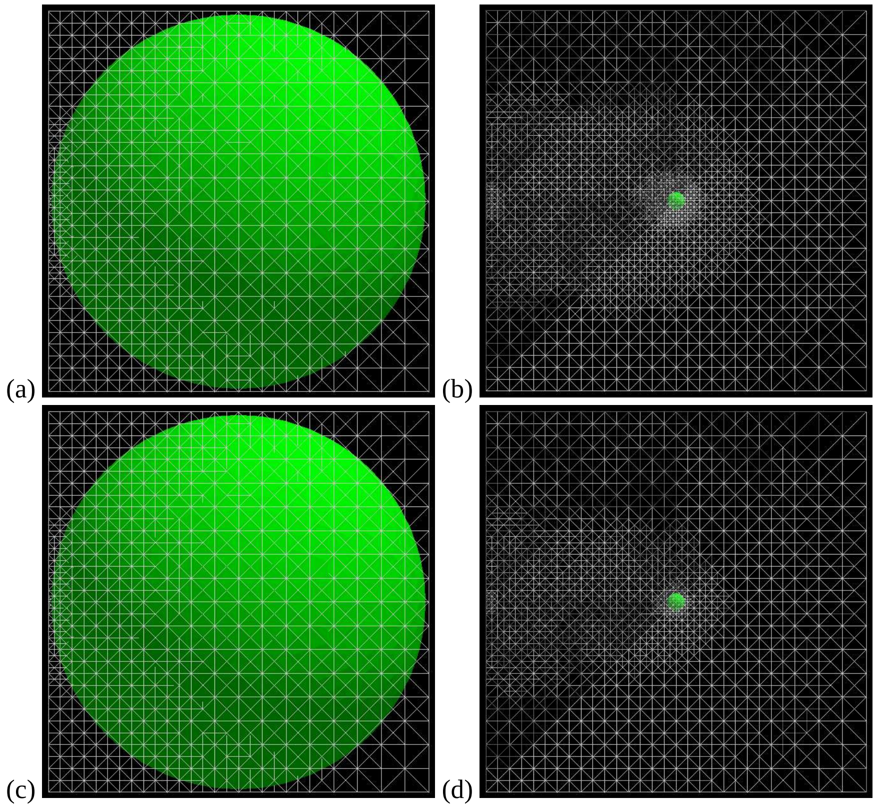

Figure 2. View-dependent visualizations using mean-square (top) and our data-dependent error metric (bottom), using isovalue 63 (left) and isovalue three (right).

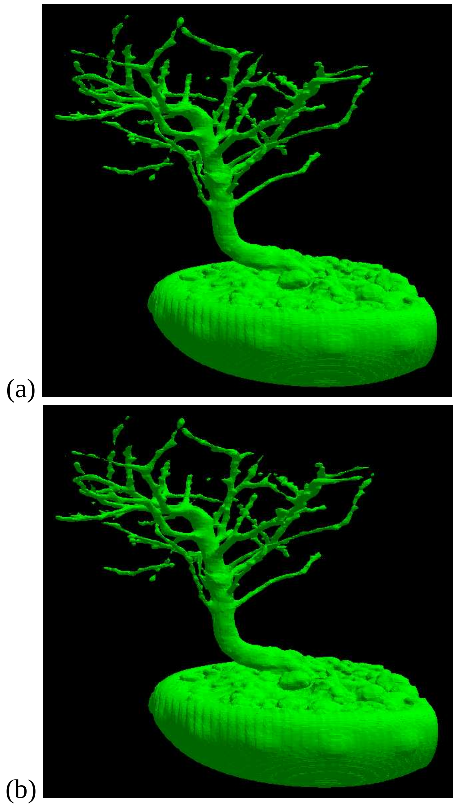

Figure 3. View-dependent visualization of a scalar field: (a) Mean-square error metric $\mathcal{E}_{MS}$ leading to 4,515,148 tetrahedra. (b) Data-dependent error metric $\mathcal{E}_s$ leading to 2,452,956 tetrahedra.

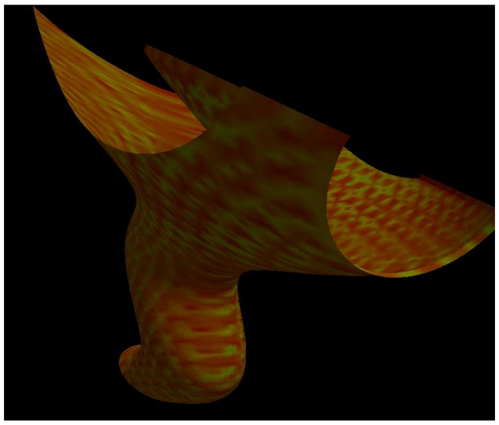

Figure 4. View-dependent refinement for a vector field using error metric $\mathcal{E}_v$. Visualization of vector magnitude and curl magnitude with isosurfacing and color mapping.

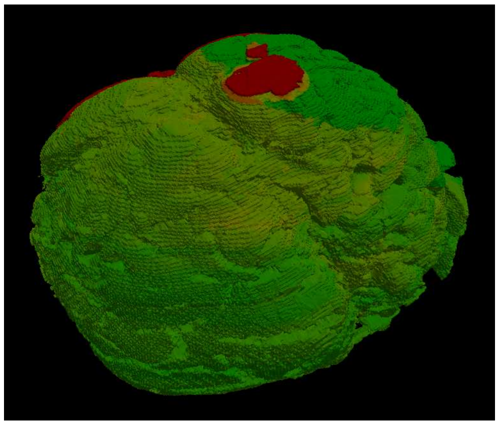

Figure 5. View-dependent refinement of two scalar fields using error metric $\mathcal{E}_m$. Visualization of two scalar quantities with isosurfacing and color mapping.

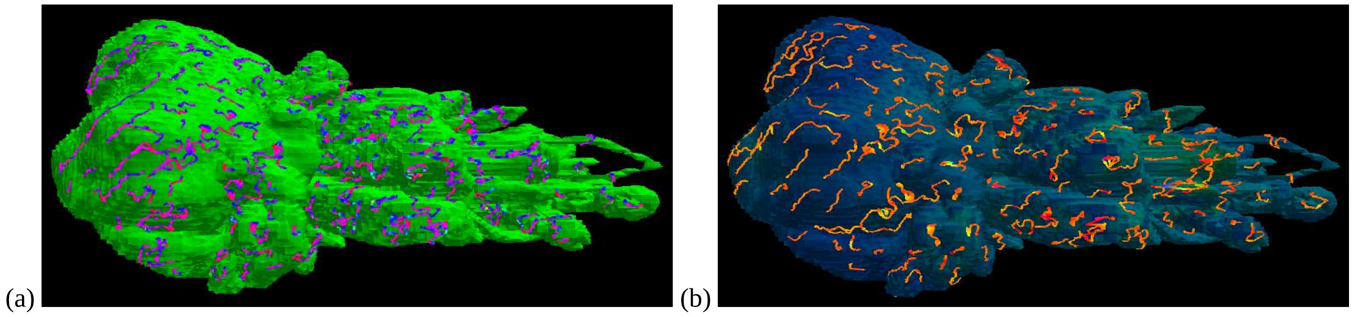

Figure 6. View-dependent refinement of one scalar field and one vector field (a) and two scalar and one vector field (b) using multi-valued error $\mathcal{E}_m$. The visualization combines isosurfacing, color mapping, and colored surface streamlines.