Direction-Driven Shape-Based Interpolation of Volume Data

Jiří Hladůvka and Eduard Gröller

Vienna University of Technology, Institute of Computer Graphics and Algorithms
Favoritenstraße 9–11, E186, A-1040 Wien, Austria
{hladuvka | groeller}@cg.tuwien.ac.at

Abstract

We present a novel approach to shape-based interpolation of gray-level volume data. In contrast to the segmentation-based techniques our method directly processes the scalar volume requiring no user interaction. The key idea is to perform the interpolation in the directions given by analysis of the eigensystem of the *structure tensor*. Our method processes a 256×256 slice within a couple of seconds yielding satisfactory results. We give a quantitative and a visual comparison to the linear interslice interpolation. Analysis of the results lead us to the conclusion that our technique has a strong potential to compete with well-established shape-based interpolation algorithms.

1 Introduction

Interpolation is required in imaging, in general, whenever the acquired image data are not at the same level of discretization as the level that is desired [1]. In volume visualization by ray casting, for instance, it is necessary to acquire samples in regular distances on a ray. The interpolation mechanism has to provide the ray caster with values at any position between grid points. Algorithms for volume analysis are often based on filtering techniques which presume isotropy. The input data has to be resampled in most cases to an isotropic discretization. Medical imaging systems usually acquire the data in slice-by-slice order resulting to different sampling ratios in x, y, and z directions. For an appropriate medical treatment it is necessary that the data at requested locations are reconstructed as precisely as possible taking into account not only the characteristics of a 3D signal conveying the data but also the topological properties of the explored structures.

2 Related work

Interpolation algorithms can broadly be divided into two categories – scene-based (image-based) and object-based (shape-based). Although scene-based filters received a lot of attention in the visualization community in recent years [9, 7, 15, 14], there is repeated evidence in the literature of the superior performance of object-based over scene-based interpolation techniques [2].

From the beginning shape-based algorithms required a segmented volume, i.e., a description of objects to be interpolated. The first approaches are based on contour interpolation.

Raya and Udupa [12] proposed an interpolation of distance fields computed from binary slices. Higgins et al. [5] also focus on the problem of interpolating binary objects rather then high-resolution gray-scale images. The presented method extends the approach of Raya and Udupa [12] employing the original density information to avoid inconsistencies in transition of objects' cross sections and their centroids. Turk and O'Brien [16] benefit from both the contour description and distance fields encoded as implicit functions providing a powerful framework for interpolation of analytically given objects.

Moshfegi [10] proposes a technique aiming for removing staircase effects in a manner that is consistent with MIP. The direction of interpolation is aligned adaptively with the axes of the vessels. The template matching proposed here considers a pair of scan-lines iteratively looking for the best match in reference windows due to mean square error and a correlation coefficient. The interpolation method is, however, applied after the projection providing thus a solution only for 2D scenes.

The first purely gray-level-based approach yielding reasonable results seems to be the three-pass algorithm proposed by Grevera and Udupa [1].

In order to adopt the previously introduced technique [12], the n-dimensional grey scene is *lifted* to a $[n+1]$-dimensional binary scene in the first step. Then binary shape-based interpolation [12] is applied. Finally, the newly interpolated $[n+1]$-binary scene is *collapsed* back to n dimensions. A drawback of this method is a high time and space complexity. Two years later this method and two variants thereof are compared [2] to the five most referred-to interpolation techniques. The paper emphasizes the superior performance of object-based over scene-based interpolation techniques, too. To see how far this is evident in a specific application, i.e., detection of brain lesions, Grevera and Udupa [3] statistically compare 100 data sets resulting from 10 patients, 2 modalities and 5 interpolation methods.

In contrast to the frequently used techniques we propose a direction-driven interpolation. The interpolation is conducted by the eigenvectors of the *structure tensor*. Structure tensors carry information on the density distribution in *simple neighborhoods*. As they are computed directly from scalar data our method does not require any segmentation or user interaction. This makes a reasonable difference between our method and most shape-based interpolation techniques.

An introduction to simple neighborhoods and the related tensor analysis as well as the concept of our method are presented in section 3. Section 4 discusses implementation issues. To see the performance, qualitative and quantitative evaluation of our method is given in section 5.

3 Pattern-Driven interpolation

Usual strategies to study local neighborhoods are based on analyzing discontinuities in the intensity [8].

The human visual system, however, can easily recognize objects that do not differ from a background by their mean gray value but only by the orientation or scale of pattern. To perform this recognition task with a digital image processing system, operators which determine the orientation of patterns are needed [6].

3.1 Simple Neighborhoods and Structure Tensor

For an appropriate representation of the emplacement of an object or a texture it is necessary to distinguish between *direction* given by an angle from the interval $\langle 0^\circ, 360^\circ)$ and *orientation* $\langle 0^\circ, 180^\circ)$. As we are locally not able to distinguish between patterns that are rotated by 180 degrees, the operators are also required to make no distinction.

A representation of the orientation by one scalar value representing the angle turns out to be not appropriate, because a measure for certainty which describes the neighborhood independently from the absolute density is not involved. This leads to a at least two-component vectorial description.

The attempt to describe a neighborhood by the gradient vector fails because it does not allow to distinguish between neighborhoods with constant values and isotropic orientation distribution where gradients cancel each other as they are integrated over the neighborhood. This is due to the opposite signs of gradients at increasing and decreasing edges.

The following optimization method has been proposed [6] to find an operator which encodes the angle of orientation, provides a certainty measure and distinguishes between constant and isotropic distributions.

The neighborhood U of the point of interest x_0 will be described by a unit vector $\vec{n}$ which exhibits the least deviation from the *orientations* of all gradients from U. As the *squared* scalar product $(\nabla g^T \vec{n})^2$ between the gradient vector ∇g and $\vec{n}$ meets criteria for measuring this deviation the following integral will be maximized:

$$\int_U h(x_0 - x)\,(\nabla g(x)^T \vec{n})^2\,dx \qquad (1)$$

where h determines a weighting function defined on U. The optimization problem given by equation (1) can be rewritten as $\vec{n}^T J \vec{n} \rightarrow \textit{maximum}$, where

$$J = \int_{-\infty}^{\infty} h(x_0 - x)\,(\nabla g(x) \nabla g(x)^T)\,dx \qquad (2)$$

is a symmetric 3×3 matrix consisting of the following elements:

$$J_{pq} = \int_{-\infty}^{\infty} h(x_0 - x) \left(\frac{\partial g(x)}{\partial x_p} \frac{\partial g(x)}{\partial x_q} \right) dx \qquad (3)$$

The matrix J is referred to as the *structure tensor* and its construction is in more details explained, e.g., in [4, 6]. The solution to $\vec{n}$ is the principal eigenvector of J.

In this work we will refer to the eigensystem of J as $\{\lambda_i, \vec{v}_i\}_{i=1}^3$ and assume the following ordering of the eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ and corresponding arrangement of the eigenvectors $\vec{e}_1, \vec{e}_2, \vec{e}_3$. We will assume, w/o l.o.g, that eigenvectors are of unit length, i.e., $|\vec{e}_i| = 1$.

3.2 Eigenvectors-Aligned Kernel

The aim of our approach to interpolation is to preserve to the most possible extent the boundaries of objects, as well as the lines, the edges, and the ridges. To achieve this we propose to conduct the interpolation in the directions given by these structures. The presumption here is that the change of the density values in these directions reaches the minimum.

Although the structure tensor has originally been proposed to describe the orientation of a neighborhood by its principal eigenvector $\vec{e}_1$, there is more useful information from its analysis. For a fixed vector $\vec{u}$, the term $\vec{u}^T J \vec{u}$ gives the square of the density change in its direction. Under the assumption that J is represented by a regular matrix it follows immediately that the minimum of the squared density change and therefore also the minimum of the density change are reached in the direction of the eigenvector $\vec{e}_3$.

According to the $rank$ of tensor J the following four cases can be distinguished in a 3D image (refer also to Table 1):

$rank(J) = 0$**:** The scalar values do not change in any direction. The represented neighborhood features constant gray values. Interpolation can simply be done with the nearest-neighborhood interpolation method.

$rank(J) = 1$**:** The scalar values significantly change in the direction of the principal eigenvector $\vec{e_1}$, while in the remaining perpendicular directions, $\vec{e_2}$ and $\vec{e_3}$, remain (nearly) constant. The corresponding neighborhood contains either a layered texture or a boundary between two objects. The interpolation will be conducted by a disc spanned by the eigenvectors $\vec{e_2}$ and $\vec{e_3}$.

$rank(J) = 2$**:** The scalar values significantly change in the directions $\vec{e_1}$ and $\vec{e_2}$ while remain (nearly) constant in $\vec{e_3}$. The corresponding neighborhood features either an edge of an object or an extruded texture. The interpolation will be driven by a stick determined by the eigenvector $\vec{e_3}$.

$rank(J) = 3$**:** The scalar values change in all directions. There is either a corner of an object or a distributed 3D texture in the neighborhood. The interpolation therefore shall consider all of the eigenvectors.

The second row of Figure 2 gives examples of color coding of the rank of the structure tensor. The areas with $rank(J) = 1$, 2, and 3 are encoded by green, red, and blue color, respectively. Homogeneous areas are excluded from the rendering.

To perform a directional interpolation at a point x_0 we will weight the density contributions $g(y)$ of voxels y from its neighborhood according to their relative position to an ellipsoid $E(x_0)$. The main axes of the ellipsoid will be given by scaled eigenvectors $\vec{a}_i$ as follows:

$rankJ$	$\vec{a}_1$	$\vec{a}_2$	$\vec{a}_3$
1	$\varepsilon\ \vec{e}_1$	$\delta\ \vec{e}_2$	$\delta\ \vec{e}_3$
2	$\varepsilon\ \vec{e}_1$	$\varepsilon\ \vec{e}_2$	$\delta\ \vec{e}_3$
3	$\delta\ \vec{e}_1$	$\delta\ \vec{e}_2$	$\delta\ \vec{e}_3$

(4)

where $\delta > 1 \gg \varepsilon > 0$. The purpose of scaling (4) is to suppress the directions of a large density change and to emphasize the directions of the small density change.

We define the weight of a point y from the interior of ellipsoid $E(x_0)$ as:

$$w(y) = 1 - \sum_{i=1}^{3} \left(\frac{\langle y - x_0, \vec{a_i} \rangle}{\langle \vec{a_i}, \vec{a_i} \rangle} \right)^2 \qquad (5)$$

and set it to zero for the exterior points and the boundary of $E(x_0)$. This sets a smooth fade-of of weights from the center of the ellipsoid to its borders. Fig. 1 demonstrates how the situation may look like in a plane given by the eigenvectors $\vec{e}_2$ and $\vec{e}_3$. The gray-level coding of the contributing grid points corresponds to their weights.

Finally, we define the interpolated density value as a weighted average:

$$g(x_0) = \frac{\sum_{y \in U(x_0)} w(y)\ g(y)}{\sum_{y \in U(x_0)} w(y)} \qquad (6)$$

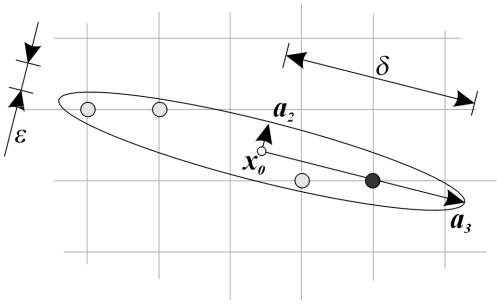


Figure 1: The ellipsoid given by the scaled eigenvectors and the weighting of the contributing grid points. The similarity in weighting of the three light-gray points is due to equation (5).

3.3 Tensor Propagation

The computation of the structure tensor due to equation (3) applies only to grid points. To calculate the directional information also at off-grid points a mechanism to compute the structure tensor there is needed.

A straightforward solution would be to generalize equation (3) involving derivative filters for computing the partial derivatives between grid points [9, 14].

An indirect approach can benefit from the already computed information at the surrounding grid points employing an interpolation of the corresponding eigensystems. Using quaternions for this task is the first choice for a high quality interpolation. On the other hand, since there is a strong coherence both in orientation and the magnitude of the structure tensor, it is not necessary to sample the tensor field densely [6]. Therefore nearest-neighbor interpolation provides a stable and a much simpler solution.

4 Implementation

As the interpolation usually is to be performed for a larger amount of voxels the implementation can be divided into two steps. In preprocessing, the eigenvalue analysis of the structure tensor is performed for a set of voxels which surround the positions to be interpolated.

P1. computation of the structure tensor J: Identifying the convolution in equation (3) with a smoothing of the product of partial derivatives, the elements J_{pq} of the structure tensor J can be computed in terms of two-pass convolution and a scalar product. In the first step the differential operators $\partial/\partial x_i$ are applied resulting to the partial derivatives $\partial g/\partial x_i$. Then the scalar products $\langle \partial g/\partial x_p, \partial g/\partial x_q \rangle$ are computed. In the final step the scalar products are smoothed with a filter corresponding to a weighting function h. We have used an optimized 3x3x3 Sobel filter [6] for the differentiation step and the Gauss filter defined on a 7x7x7 neighborhood for the averaging step. To reduce the computational overhead we exploited the separability of both filters.

P2. eigenvalue analysis of J: Since the structure tensor J is represented by a symmetric, real-valued matrix, the fast-converging Jacobi method can be used for eigen-analysis, as recommended by Press et al. [11]. The tuples of eigenvalues and eigenvectors are then rearranged to satisfy $\lambda_1 \geq \lambda_2 \geq \lambda_3$.

After the preprocessing step for each requested position x_0 the interpolation involves the following steps.

I1. inheriting the tensor information: In accordance to section 3.3 we have used the nearest neighbor interpolation for the tensor transfer.

I2. interpolation: For all voxels y_i from the neighborhood of x_0 determined by the scaling factor δ weights $w(y_i)$ are computed according to the equation (4) and (5) and the weighted average (6) is assigned to the result of interpolation. The values of δ and ε have been set empirically after an error analysis of several experiments to $\varepsilon = 0.1$ and $\delta = 0.2$

5 Evaluation

The purpose of this section is to provide a threefold analysis of the performance of the proposed method. Firstly, in section 5.1 we give a quantitative error analysis and a comparison to the linear filtering in z direction. Secondly, a visual evaluation of error volumes is presented in section 5.2. Thirdly, the time and space complexity of our method is discussed in section 5.3.

For the following analysis and comparison we have used 11 data sets covering a broad spectrum

in terms of modality, resolution, noise characteristics and object detail. The first group consists of the small-resolution, noise-free artificial data sets generated by the *vxt library* [13]. The second group comprises the data acquired from CT and MRI scanners. All data set were quantized to 256 levels.

5.1 Quantitative Evaluation

The approach to the comparison is to pretend there is a slice missing in the volume, to estimate this slice and compare it to the original. For the comparison, we reuse the framework for error analysis by Grevera and Udupa [2]. The authors introduced three *figures-of-merit* (FOM). In the following expression of FOMs, m denotes an interpolation method (linear in z and directional), S is the data set being interpolated, $g(v)$ represents the original density value in the voxel given by coordinates v, and $g_m(v)$ represents the density estimation due to the method m at the voxel given by coordinates v. Interpolation runs over all slices of S, where slice V^k is defined by its z-coordinate k.

1. Mean-squared difference:

$$\mathrm{FOM}_m^1(S) = \frac{1}{N} \sum_k \sum_{v \in V^k} (g(v) - g_m(v))^2 \tag{7}$$

   where N is the total number of voxels involved in the comparison.

2. Number of sites of disagreement

$$\mathrm{FOM}_m^2(S) = \sum_k \sum_{v \in V^k} \tau(|g(v) - g_m(v)|) \tag{8}$$

   where

$$\tau(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

3. Largest difference

$$\mathrm{FOM}_m^3(S) = \max_{\substack{v \in V^k \\ k}} \{|g(v) - g_m(v)|\} \tag{10}$$

The measurements due to the equations (7)–(10) for the linear interpolation and the proposed directional interpolation are summarized in Table 2.

We make the following three observations regarding the performance of our technique. 1. Our technique yields in average better estimates of the original density values than the linear interpolation. This is especially remarkable for the *vxt* data. As the Engine data set features many z-axis aligned objects the directional interpolation does not outperform the linear interpolation so dramatically. 2. An exact reconstruction of the original density values happens at more sites in the case of directional interpolation. Again, it is more obvious in the case of the *vxt* data and gets almost negligible for the scanned data sets. In our opinion, however, these measurements are of a lower importance than the mean-squared error. 3. The maximal error is generally smaller in the case of directional interpolation.

5.2 Visual Evaluation

To evaluate the performance of both interpolation methods visually, we create two error volumes consisting of voxels with densities set to $|\, g_m(v) - g(v)\,|$, where m denotes the interpolation method. The error volumes are then displayed with splatting. The 3rd and the 4th rows of Figure 2 bring examples on rendering of error introduced by the linear and the directional interpolation, respectively.

Since the directional interpolation yields very small differences in the case of the vxt Facet Octahedron data set, almost every pixel gets rendered to the background color in the corresponding image. In case of the CT head with markers the improvement by our interpolation is mostly apparent at the positions featuring objects, e.g, the wires in the markers. As the error measurements for the Engine data set do not differ significantly a visual evaluation is almost impossible in this case.

5.3 Time and Space Complexity

Referring to the Table 3, the performance of our method is clearly dominated by the smoothing step (section 4, P1). The total timing therefore strongly depends on the size of the support of the weighting function h. Our implementation processes in average 11000 voxels per second on a 400 MHz Pentium II. This correspond approximately to 6 seconds for reconstruction of one 256×256 slice.

To store the eigenvectors of the structure tensor, our implementation requires a space of $O(3n)$ where n is the size of the data to be interpolated. This can, however, be optimized to an on-the-fly computation and caching of the structure tensors

which gives the same space complexity as the one of linear interpolation, i.e., $O(1)$.

6 Concluding Remarks

We present a new method for interpolating the grey-level volume data taking into account the topological properties of the structures. Unlike the methods which are based on an a priori information from segmentation, our method processes the density data directly requiring no user interaction.

In order to preserve the boundaries of objects, lines, and the ridges to the most possible extent we proposed to conduct the interpolation by the directions of these structures. The information on the direction of interpolation is inherited from the textural description of the objects by the structure tensors.

We compared the performance of our method to the most used linear interpolation in the direction of the z axis for 11 data sets concluding that our method outperforms the linear interpolation in the mean error, maximal error and in the number of sites with exact reconstruction.

There are two main issues not addressed in this work. Firstly, the size of the oriented filter is given by two constants which have been set empirically after many experiments. In our opinion this is not appropriate if the method is required to perform robustly for all kind of data. Instead of using fixed constants we think of using parameters which reflect the *magnitude* of the density change separately in each neighborhood. A good choice for this purpose seem to be the *eigenvalues* of the structure tensor.

Secondly, we find it necessary to compare our method to other techniques, especially to the state-of-the-art in shape-based interpolation introduced by Grevera and Udupa [1]. We would like to conclude with remarks on possible results of this comparison. Intuitively, our method requires far less space and time than the interpolation in [1]. To interpolate one slice of n voxels, the algorithm by Grevera and Udupa requires space of $O(Gn)$ where G denotes the quantization level and tens of minutes (as measured on a Sparc 2). In contrast, our method interpolates one slice in a couple of seconds requiring $O(3n)$ space. Finally, since we have used the same error measurements and a statistical comparison to the same interpolation method as in [1] we were able to indirectly compare the error performance of both methods. Even though this preliminary comparison sounds promising for our method, further work is required to be done. We are convinced that such a comparison has to be done directly for the same data sets and address it to the future work.

Acknowledgments

The work presented in this publication has been funded by the V^{is}M^{ed} project[1]. V^{is}M^{ed} is supported by *Tiani Medgraph*,[2] Vienna, and the *Forschungsförderungsfonds für die gewerbliche Wirtschaft*,[3] Austria. A part of the work has also been supported by the BandViz project[4] which is supported by FWF under project number 12811.

References

[1] G. Grevera and J. Udupa. Shape-based interpolation of multidimensional grey–level images. *IEEE Transactions on Medical Imaging*, 15(6):881–892, Dec. 1996.

[2] G. Grevera and J. Udupa. An objective comparison of 3-D image interpolation methods. *IEEE Transactions on Medical Imaging*, 17(4):642–652, Aug. 1998.

[3] G. Grevera, J. Udupa, and Y. Miki. A task-specific evaluation of three-dimensional image interpolation techniques. *IEEE Transactions on Medical Imaging*, 18(2):137–143, Feb. 1999.

[4] H. Haußecker and B. Jähne. A tensor approach for local structure analysis in multi-dimensional images. In *Proceedings 3D Image Analysis and Synthesis '96, Universität Erlangen-Nürnberg*, 1996.

[5] W. Higgins, C. Morice, and E. Ritman. Shape-based interpolation of tree-like structures in three-dimensional images. *IEEE Transactions on Medical Imaging*, 12(3):439–450, 1993.

[6] B. Jähne. *Digital Image Processing*, chapter 11.3 First-Order Tensor Representation, pages 348–364. Springer–Verlag Berlin Heidelberg, 4. edition, 1997.

[1] `http://www.vismed.at/`
[2] `http://www.tiani.com/`
[3] `http://www.fff.co.at/`
[4] `http://bandviz.cg.tuwien.ac.at/`

[7] T. Lehmann, C. Gonner, and K. Spitzer. Survey: interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18(11):1049–1075, Nov. 1999.

[8] E. Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 465–476, 1996.

[9] T. Möller, R. Machiraju, K. Müller, and R. Yagel. Evaluation and Design of Filters Using a Taylor Series Expansion. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):184–199, 1997.

[10] M. Moshfeghi. Directional interpolation for magnetic resonance angiography data. *IEEE Transactions on Medical Imaging*, 12(2):366–379, June 1993.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*, chapter 11, pages 456–469. Cambridge University Press, 2. edition, 1992.

[12] S. Raya and J. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Transactions on Medical Imaging*, 9(1):32–42, Mar. 1990.

[13] M. Šrámek and A. Kaufman. vxt: A class library for voxelization of geometric objects. `http://www.cs.sunysb.edu/~vislab/vxtools/vxt/`, 2000.

[14] T. Theußl, H. Hauser, and E. Gröller. Mastering windows: Improving reconstruction. In *Proceedings of IEEE Volume Visualization*, pages 101–108, 2000.

[15] P. Thevenaz, T. Blu, and M. Unser. Interpolation revisited. *IEEE Transactions on Medical Imaging*, 19(7):739–758, July 2000.

[16] G. Turk and J. F. O'Brien. Shape transformation using variational implicit functions. *Computer Graphics*, 33(Annual Conference Series):335–342, 1999.

$rank(J)$	conditions	neighborhood description
0	$\lambda_1 \simeq \lambda_2 \simeq \lambda_3 \simeq 0$	constant neighborhood
1	$\lambda_1 > \lambda_2 \simeq \lambda_3 \simeq 0$	boundary or layered texture
2	$\lambda_1 \simeq \lambda_2 > \lambda_3 \simeq 0$	edge or extruded texture
3	$\lambda_1 \simeq \lambda_2 \simeq \lambda_3 > 0$	corner or isotropy

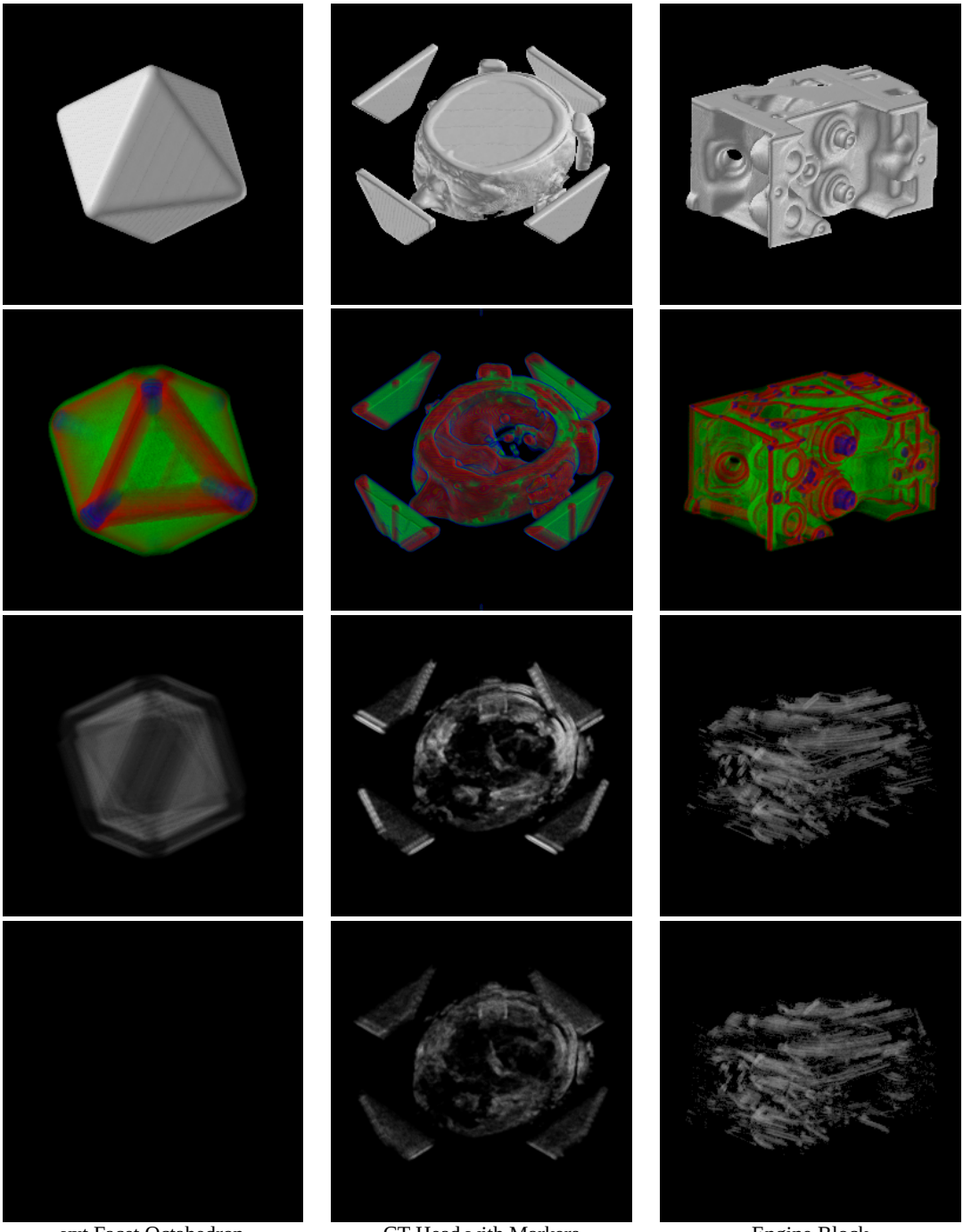
Table 1: The cases of density distribution in a 3D scene

	FOM[1]		FOM[2]		FOM[3]	
data set	z-lin.	dir.	z-lin.	dir.	z-lin.	dir.
vxt Wire Tetrahedron	13.1	4.5	1 380	1 156	30	16
vxt Wire Octahedron	41.0	0.1	9 672	1 428	24	3
vxt Wire Dodecahedron	14.3	1.8	1 300	963	37	17
vxt Facet Tetrahedron	14.9	4.7	7 877	6 846	24	23
vxt Facet Octahedron	77.7	0.1	22 895	5 644	35	5
vxt Facet Dodecahedron	35.9	2.5	19 612	17 891	31	16
Lobster	51.0	19.8	6 815	6 567	103	61
Engine Block	7.1	6.3	705 850	609 911	36	34
CT Head with Markers	105.3	25.1	133 312	131 901	124	114
MRI Head	84.7	41.3	570 784	556 033	116	90
Vertebra	7.0	4.8	189 584	114 351	32	26

Table 2: Error measurements due to the Equations (7)–(10) for the linear interpolation in z-direction and the newly proposed directional interpolation.

Input Volume		eigensystem of J			interpo-	total
Data set	Dimensions	3x3 Sobel	7x7 Gauss	Jacobi	lation	
vxt Wire Tetrahedron	50 × 44 × 42	1.2	3.4	1.0	1.6	7.2
vxt Wire Octahedron	59 × 59 × 59	2.3	6.2	1.8	3.3	13.6
vxt Wire Dodecahedron	58 × 56 × 48	1.8	4.9	1.8	3.2	11.7
vxt Facet Tetrahedron	50 × 45 × 42	1.1	3.0	1.0	2.7	7.8
vxt Facet Octahedron	59 × 59 × 59	2.4	6.7	2.0	5.2	16.3
vxt Facet Dodecahedron	59 × 56 × 49	2.0	5.3	2.2	6.8	16.3
Lobster	120 × 120 × 34	5.7	15.6	6.2	5.8	33.3
Engine Block	256 × 256 × 110	88.8	383.2	94.8	140.0	706.8
CT Head with Markers	256 × 256 × 44	33.4	110.6	30.0	39.4	213.4
MRI Head	256 × 256 × 109	74.7	249.1	80.3	127.3	531.4
Vertebra	128 × 128 × 74	12.7	50.4	15.4	20.4	98.9

Table 3: Time in seconds for the directional interpolation as measured on a 400 MHz Pentium II.



vxt Facet Octahedron | CT Head with Markers | Engine Block

Figure 2: Volume rendering of the original data sets (1st row), color coding of the rank of the structure tensor (2nd row), error volume of the linear interpolation (3rd row), and error volume of the directional interpolation (4th row).