J Vis (2011) 14:161–170
DOI 10.1007/s12650-011-0075-0

REGULAR PAPER

Yuusuke Tsukamoto · Shuhei Kawashima · Seiji Inoue · Shin Ito · Shinji Kataoka · Kazuyuki Kojima · Kyoko Hasegawa · Susumu Nakata · Satoshi Tanaka

Data fitting independent of grid structure using a volumic version of MPU

Received: 1 June 2010 / Accepted: 13 February 2011 / Published online: 12 March 2011

Abstract Volume graphics is a key technology in fields such as fluid dynamics and medical science. The visualization of volume data requires the creation of a continuous scalar field to exactly or approximately interpolate scalar values assigned to the discrete voxels. In the present paper, we propose a method that we refer to as the volumic version of the multi-level partition of unity (volumic MPU). The method approximately interpolates the scalar values with good precision to generate a scalar field that is continuous up to second-order differentiation. The volumic MPU, being independent of grid structures of input volume data, is applicable to both irregular-grid data and regular grid data. The volumic MPU can also be used as an effective data-compression technique. The speed to evaluate the created scalar field is almost as high as that of trilinear interpolation.

Keywords Volume graphics · Data fitting · Grid independency · Data compression · MPU

Y. Tsukamoto
Hitachi Systems and Services, Ltd, Tokyo, Japan

S. Kawashima
Institute of Science and Engineering, Ritsumeikan University, Shiga, Japan

S. Inoue
Hitachi Software Engineering Co., Ltd, Tokyo, Japan

S. Ito
Sony Corporation, Tokyo, Japan

S. Kataoka
FUJITSU Kansai Systems Ltd, Osaka, Japan

K. Kojima
NEC Soft, Ltd, Tokyo, Japan

K. Hasegawa
Ritsumeikan University, Research Organization of Science and Engineering, Shiga, Japan

S. Nakata · S. Tanaka (✉)
College of Information Science and Engineering, Ritsumeikan University, Shiga, Japan
E-mail: stanaka@media.ritsumei.ac.jp

S. Nakata
E-mail: snakata@media.ritsumei.ac.jp

1 Introduction

Volume graphics is a technology that enables visualization of a target internal structure and its behavior, and provides effective means for visual insight into various scientific and engineering fields (Hansen and Johnson 2004). Targeted volume data includes regular-grid data in the form of a cubic grid structure, irregular-grid data, such as tetrahedral grid data, and non-grid data which does not have a grid structure. Hereafter, the term "irregular-grid" data are used to refer to both irregular-grid data and non-grid data.

Recently, the demand for visualizing irregular-grid data has been growing. For example, the volume data frequently handled in structural analysis is irregular-grid data with a tetrahedral grid structure. Geological data, such as underground pressure distribution data are discrete point data without a grid structure. However, most standard data fitting/interpolation methods for volume visualization assume that the target data are regular-grid data. Therefore, the application of these methods, e.g., trilinear interpolation or Spline interpolation (Marschner and Lobb 1994; Lee et al. 1997), to irregular-grid data are not straightforward.

There are methods to fit both regular and irregular-grid data using radial basis functions (Jang et al. 2004; Weiler et al. 2005). However, full application of the radial basis functions for all of the data requires significant memory and computation time. Another possible method is moving least square approximation of volume data (Lancaster 1981). However, the computational cost for evaluation of the approximated scalar field tends to be high because a least square problem must be solved for each evaluation of the field. Moreover, these methods do not consider effects of data compression.

On the other hand, some approaches use the implicit surface method to reconstruct three-dimensional surfaces from point data obtained using a range sensor (Savchenko et al. 1995; Ohtake et al. 2003). These approaches express a surface as a zero isosurface in relation to an automatically created continuous scalar field, regardless of the regularity of the input-point positions. Ohtake et al. (2003) proposed the multi-level partition of unity (MPU) method. In their method, a large point group serves as the target and the space is partitioned in response to local changes in shape, with fitting performed using a set of quadratic functions.

In the present paper, we propose a method that we refer to as the volumic version of the multi-level partition of unity (volumic MPU). The proposed method fits discrete volume data, i.e., scalar-valued voxel data, of any grid structure. (In the present paper, we use the term 'fit' to mean 'approximately interpolate'.) In the proposed method, a high-quality scalar field, which is continuous to second-order differentiation, is generated in the total space. The generated scalar field enables us to execute high-quality visualization of regular/irregular-grid data. Note that the volumic MPU is applicable to both regular and irregular-grid data without any tuning. This is because the method is grid-independent, i.e., does not use any information of grid structures of volume data. Note also that the volumic MPU can be used as an effective data-compression technique, especially, for irregular-grid data, which tend to be large because such data must include explicit information on voxel positions.

A preliminary report on volumic MPU has been published in (Tsukamoto et al. 2009). In the present paper, we provide a full explanation of the proposed method and describe the following additional recent developments: (1) the rectangular support that can accelerate evaluation of the created scalar field, and (2) the data-compression effect.

2 Algorithm and characteristic features of the volumic MPU

2.1 Algorithm of the volumic MPU

The volumic MPU enables adaptive local fitting of volume data and its smooth connection over the total space. Its algorithm is as follows:

1. Divide the space, which is initially the total space where volume data is defined, into rectangular subspaces with the binary tree or octree method.
2. For each rectangular subspace, perform least square fitting of the scalar values. The use the scalar values assigned to voxels that exist inside an ellipsoidal or rectangular shape surrounding the subspace [see Fig. 1 (left)]. This fitting creates a trial local fitting function $Q(\mathbf{x})$, where $\mathbf{x}$ is a position vector.
    - We hereinafter refer to the surrounding shape as the compact support. The ellipsoidal compact support is referred to simply as the ellipsoidal support, and the rectangular compact support is referred to as the rectangular support.

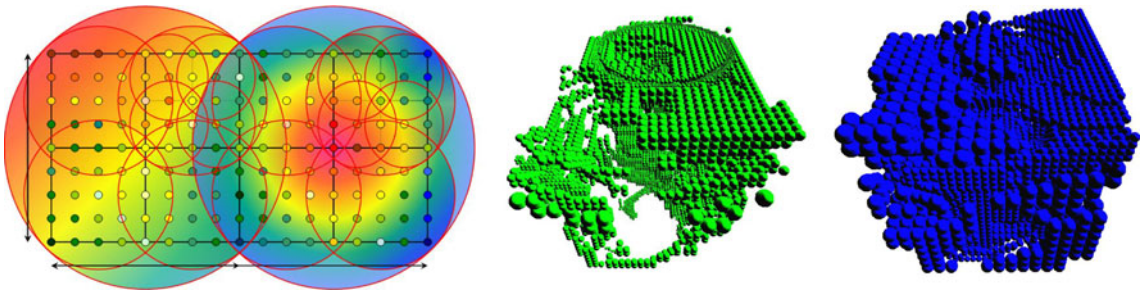

Fig. 1 Ellipsoidal supports surrounding rectangular subspaces (*left*). Distribution of the linear (*middle*) and quadratic (*right*) local fitting functions $Q_i(\mathbf{x})$ for 'tornado' data

- The compact support should be slightly larger than the rectangular subspace it contains (see step 5 below). In the present paper, we make widths of the compact support 1.1 times larger than that just circumscribing the rectangular subspace.

3. Adopt each $Q(\mathbf{x})$ that satisfies a required accuracy, and save the subspace as a leaf node of the tree. Discard each $Q(\mathbf{x})$ that does not satisfy the required accuracy and return to step 1 to divide the subspace again. We hereinafter refer to the saved leaf node as the MPU cell. The ith MPU cell stores its local definition of the local fitting function $Q_i(\mathbf{x})$.

   - Exception handling: If the fitting fails, e.g., for a very small subspace, we forcibly adopt a constant local fitting function, giving up the subdivision. Local fitting with the radial basis functions (Jang et al. 2004; Weiler et al. 2005) can also be performed to obtain higher accuracy. Exception handling is executed if one of the following conditions is satisfied: (a) the compact support contains fewer than seven voxels, and (b) the maximum width of the compact support is less than the smallest inter-voxel distance.

4. Steps 1, 2 and 3 above are repeated recursively until the total space is covered with a set of MPU cells.
5. Connect all the created $Q_i(\mathbf{x})$ smoothly to construct a continuous fitting scalar field $f(\mathbf{x})$. For this purpose, we use a proper compact support function, $\phi_i(\mathbf{x})$. The function $\phi_i(\mathbf{x})$ should have non-zero values only inside the compact support of the ith MPU cell (see Sect. 2.4 for details). The scalar field $f(\mathbf{x})$ is defined as:

$$f(\mathbf{x}) = \sum_i \phi_i(\mathbf{x}) Q_i(\mathbf{x}). \tag{1}$$

2.2 Adaptive assignment of local fitting functions

In the local data fitting described above, we properly assign one of the four types of functions to $Q_i(\mathbf{x})$. Namely, the fitting accuracy of $Q_i(\mathbf{x})$ is evaluated in the following sequence and adopt the $Q_i(\mathbf{x})$ for which the accuracy is sufficient: 0th order (constant), first order (linear), second order (quadratic), and third order (cubic). The constant $Q_i(\mathbf{x})$ is simply the average of the local scalar values. The linear $Q_i(\mathbf{x})$ has the form $Q_i(\mathbf{x}) = \lambda_{(i,1)}x + \lambda_{(i,2)}y + \lambda_{(i,3)}z + \lambda_{(i,4)}$. The quadratic $Q_i(\mathbf{x})$ has the form,

$$\begin{aligned} Q_i(\mathbf{x}) = {} & \lambda_{(i,1)}x^2 + \lambda_{(i,2)}y^2 + \lambda_{(i,3)}z^2 + 2\lambda_{(i,4)}xy + 2\lambda_{(i,5)}yz + 2\lambda_{(i,6)}zx \\ & + \lambda_{(i,7)}x + \lambda_{(i,8)}y + \lambda_{(i,9)}z + \lambda_{(i,10)}. \end{aligned}$$

The cubic $Q_i(\mathbf{x})$ has a similar form with 20 coefficients. Each MPU cell contains the coefficients $\lambda = (\lambda_{(i,1)}, \lambda_{(i,2)}, \ldots)$ which define $Q_i(\mathbf{x})$. Figure 1 (middle) shows an example of distribution of the linear $Q_i(\mathbf{x})$ for the well-known 'tornado' sample data, and Fig. 1 (right) is a similar figure of the quadratic function.

The coefficients λ of $Q_i(\mathbf{x})$ are determined with least square method. Namely, these coefficients are determined such that the following values are minimized: $\sum_{j=1}^{N_i} \left(Q_i(\mathbf{x}_j) - \rho(\mathbf{x}_j)\right)^2$ where $\mathbf{x}_j$ is a position vector of the jth voxel inside the compact support surrounding the ith MPU cell, $\rho(\mathbf{x}_j)$ is a scalar value assigned to the jth voxel, and N_i is the number of voxels in the compact support. The accuracy of the least square fitting is evaluated by calculating the error defined by

$$\varepsilon_i \equiv \max_j \left| Q_i(\mathbf{x}_j) - \rho(\mathbf{x}_j) \right|^2. \tag{2}$$

If the error ε_i is smaller than a required accuracy, $\varepsilon^{(0)}$, we adopt the $Q_i(\mathbf{x})$. In the case that volume data of scalar values are normalized to the range between 0.0 and 255.0, $\varepsilon^{(0)} \simeq 0.5$ or 1.0 appears to be a proper choice. In the present paper, we adopt 1.0.

Note that no grid structural information is used in the above-mentioned algorithm/prescription of the volumic MPU. As such, calculation can be performed using exactly the same source code, regardless of the type of grid structure, or even, whether a grid structure exists or not.

Note also that the volumic MPU can be regarded as a type of smoothing method. The difference from conventional smoothing techniques is that errors caused by smoothing are minimized based on the prescription of the multi-level partition of unity. Moreover, sizes of the errors are controllable with the input accuracy parameter $\varepsilon^{(0)}$.

2.3 Mixed use of binary tree and octree

In creating the tree of MPU cells, we use a mixture of binary tree type and octree type space division. This is a new technique, and it is not included in the original MPU to create implicit surfaces (Ohtake et al. 2003). The goal of this technique is to treat, e.g., volume data that are thin in one direction. For such data, subspaces generated only by octree type division become thin rectangular boxes. Areas of such shapes are sometimes not appropriate for least square data fitting, because the number of fitted voxels tends to be too small in the thin direction. Therefore, at the early stage of creating the tree, we adopt the binary tree type division, and always divide the longest edges of the rectangular subspaces. This can make the rectangular subspaces more square. The space division is switched to the octree type, when the binary tree type division can no longer make the rectangular subspaces more square. In Fig. 2, we show schematically an example of the created subspaces in two dimensions.

2.4 Multi-level partition of unity and the compact support function

The scalar field $f(\mathbf{x})$, which is created by connecting the local fitting functions $Q_i(\mathbf{x})$ according to Eq. (1), has good mathematical features in smoothness and accuracy of fitting. These good features originate from the property of the multi-level partition of unity. It is to execute adaptive normalization in each local region, when we evaluate $f(\mathbf{x})$ of given $\mathbf{x}$, regardless of the tree level of related MPU cells. For example, if $\mathbf{x}$ is inside two compact supports, 1 and 2, then the normalized compact support function $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ are defined as follows:

$$\phi_1(\mathbf{x}) = \frac{w_1(\mathbf{x})}{w_1(\mathbf{x}) + w_2(\mathbf{x})}, \quad \phi_2(\mathbf{x}) = \frac{w_2(\mathbf{x})}{w_1(\mathbf{x}) + w_2(\mathbf{x})}, \tag{3}$$

where $w_1(\mathbf{x})$ and $w_2(\mathbf{x})$ are proper unnormalized compact support functions defined in compact supports 1 and 2, respectively.

The ellipsoidal support can be defined by adopting the B-Spline function as $w_i(\mathbf{x})$, which is continuous to second-order differentiation:

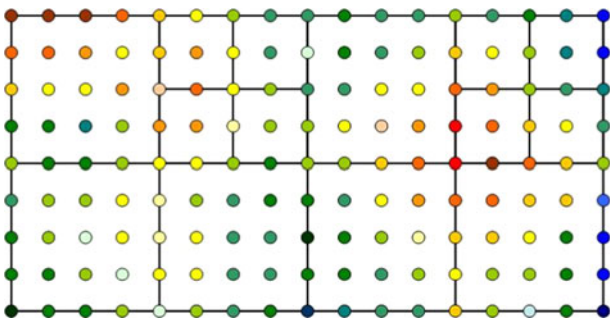

Fig. 2 Subspaces created by the mixed use of the binary tree type and octree type space division. The *filled circles* indicate voxels and the *colors* express scalar values

$$w_i(\mathbf{x}) = b(t), \quad t \equiv 2\sqrt{\left(\frac{x-c_x}{a_x}\right)^2+\left(\frac{y-c_y}{a_y}\right)^2+\left(\frac{z-c_z}{a_z}\right)^2}, \tag{4}$$

$$b(t) = \begin{cases} \frac{1}{2}(t+2)^3 - 4(t+2)^2 + 10(t+2) - \frac{22}{3} & (0 \le t < 1) \\ -\frac{1}{6}(t-2)^3 & (1 \le t < 2), \\ 0 & (2 \le t) \end{cases} \tag{5}$$

where (c_x, c_y, c_z) is the center of the ith MPU cell, and a_x, a_y, a_z are half-axis lengths of the ellipsoidal support surrounding the MPU cell. Note that $w_i(\mathbf{x})$ vanishes at the boundary of the ellipsoid. In the original MPU to create implicit surfaces (Ohtake et al. 2003), a B-Spline function of quadratic order is used. In the proposed method we adopt a cubic-order B-Spline function to make the created scalar field continuous to second-order differentiation.

The rectangular support can be defined by adopting the following $w_i(\mathbf{x})$:

$$w_i(\mathbf{x}) = b(|s_x|)b(|s_y|)b(|s_z|), \tag{6}$$

where $(s_x, s_y, s_z) \equiv 2\big((x-c_x)/\ell_x, (y-c_y)/\ell_y, (z-c_z)/\ell_z\big)$, (c_x, c_y, c_z) is the center of the ith MPU cell, and ℓ_x, ℓ_y, ℓ_z are half widths of the rectangular support of the MPU cell. Note that $w_i(\mathbf{x})$ vanishes at the rectangular boundary of the support.

The ellipsoidal support can generate smoother images than the rectangular support, although the rectangular support is beneficial in quicker evaluation of the created scalar field $f(\mathbf{x})$. This is because the ellipsoidal support tends to become larger than the rectangular support and so includes more voxels. For example, a spherical support must be at least approximately 2.7 times larger than the MPU cell, whereas a cubic support can be almost as small as the MPU cell.

3 Application to regular-grid data

In this section, we show examples of applying the volumic MPU to regular-grid data. The ellipsoidal support of Eq. (4) is used in creating and evaluating the scalar field $f(\mathbf{x})$ of Eq. (1). Although we have also investigated the rectangular support, we could not find any significant differences in the created images. We used simulation data created by Japan Agency for Marine-Earth Science and Technology after slight modification. Visualization of the differential fields are also demonstrated.

Figure 3 shows volume visualization of the ocean–temperature distribution defined on a 3,600 (east-west) × 1,500 (north-south) × 54 (depth) regular grid. The scalar values are normalized to real values between 0.0 and 255.0 beforehand. Simple ray-casting rendering without opacity accumulation is applied to a scalar field $f(\mathbf{x})$ generated by fitting the input volume data with the volumic MPU. Because the volume data are thin in the depth direction, the binary tree type space division is executed 11 times at the first stage of the space division, until the rectangular subspaces sufficiently approach a square shape (see Sect. 2.3). In generating Fig. 3, we evaluated fitting errors of $f(\mathbf{x})$ at input voxel points. The average error was evaluated to be 0.25 for $\varepsilon^{(0)} = 1.0$. Figure 4 compares visualization based on the volumic MPU (left) to the Spline interpolation (right). Each generated image is digitally zoomed to obtain a magnified image to investigate the differences in detail. The volumic MPU produces as good an image as the Spline interpolation that has been widely used as a high-quality interpolation method.

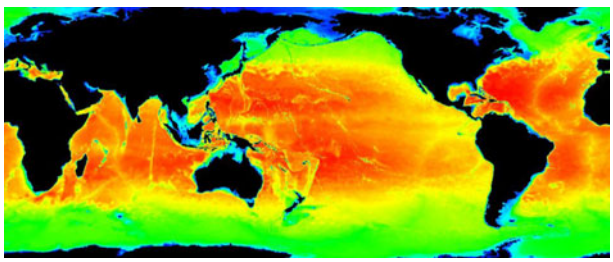

Fig. 3 Ocean–temperature distribution visualized using the volumic MPU

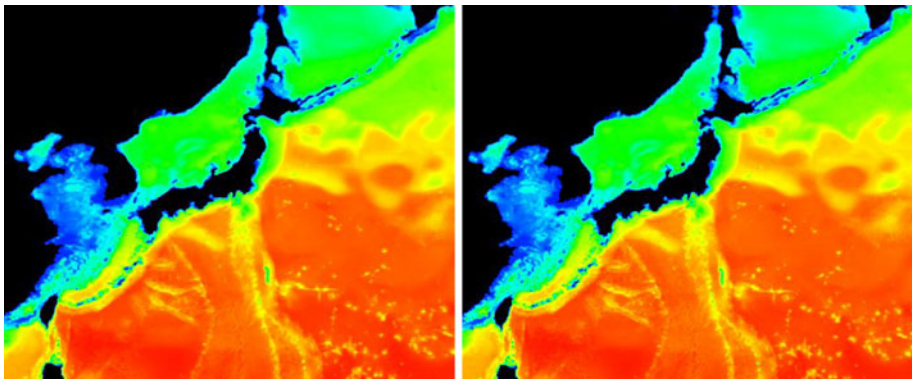

Fig. 4 Comparison of magnified images by the volumic MPU (*left*) and the Spline interpolation (*right*)

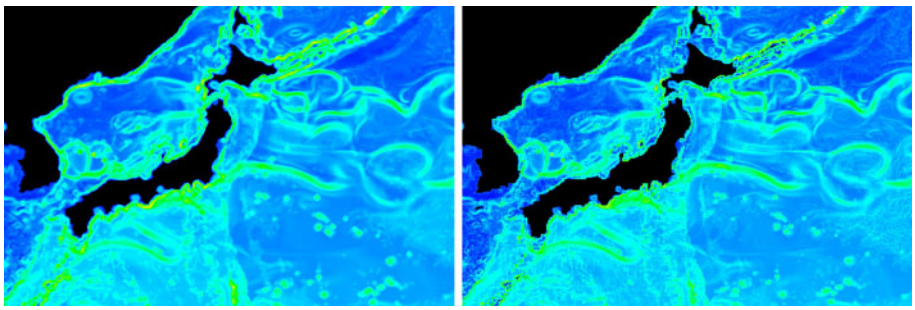

Fig. 5 Comparison of magnified images of the gradient field by the volumic MPU (*left*) and the trilinear interpolation (*right*)

Figure 5 compares magnified images of the gradient field based on the volumic MPU (left) and the trilinear interpolation (right), which has been widely used as a fast interpolation method. The image on the left (volumic MPU) looks smoother than the image on the right (trilinear interpolation). Therefore, Fig. 5 confirms the smoothing effect of the volumic MPU. Moreover, the errors occurring in the smoothing are under control. In generating Fig. 5 (left), the average error measured at the voxel points is 0.20 for $\varepsilon^{(0)} = 1.0$. The exception handling using the constant function (see Sect. 2.1) is applied to about 5% of the created MPU cells.

Because the scalar field $f(\mathbf{x})$ generated by the volumic MPU is continuous to second-order differentiation, the volumic MPU is suitable for visualizing differential fields, such as a gradient field $\|\nabla f(\mathbf{x})\|$ or a Laplacian field $|\nabla^2 f(\mathbf{x})|$. Visualizing differential fields is useful, e.g., for understanding boundaries/edges inside 3D structures.

Figure 6 shows a magnified image of the Laplacian field of $f(\mathbf{x})$ of Fig. 3. We have also created a similar image with the Spline interpolation and confirmed that it is as good as Fig. 6.

4 Application to irregular-grid data

Because the volumic MPU does not use any grid-structural information, the volumic MPU is applicable to irregular-grid data as well as regular-grid data.

Figure 7 (left) is visualization of tetrahedral-grid data of 'plasma', while Fig. 7 (right) shows the grid structure of this data. Figure 8 (left) is visualization of volume data of the human 'foot' consisting of seven million randomly scattered voxels, which are created by randomly resampling the original regular-grid data. The scattered voxels are also shown in Fig. 8 (right). The voxels of the plasma and the foot have scalar values between "1.010 and 1.897" and "10 and 159", respectively. Before executing the volumic MPU, the scalar values are normalized to real values of between 0.0 and 255.0. We then set the accuracy parameter to $\varepsilon^{(0)} = 1.0$. In generating Figs. 7 and 8, we evaluated the fitting errors of $f(\mathbf{x})$ at input voxel points. The average errors were 8.09×10^{-2} for the 'plasma' and 1.57×10^{-1} for the 'foot' data. These errors are acceptable for practical use of the method.

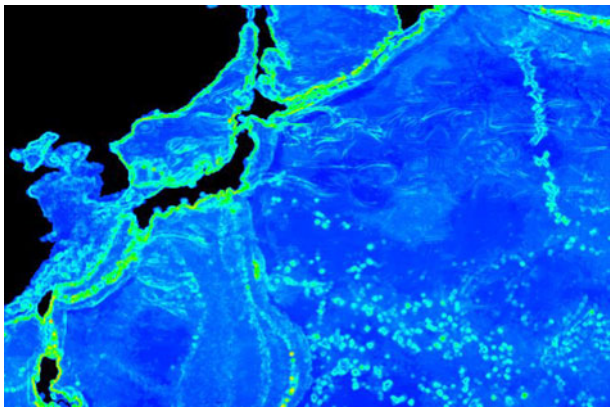

Fig. 6 Magnified image of the Laplacian field visualized by the volumic MPU

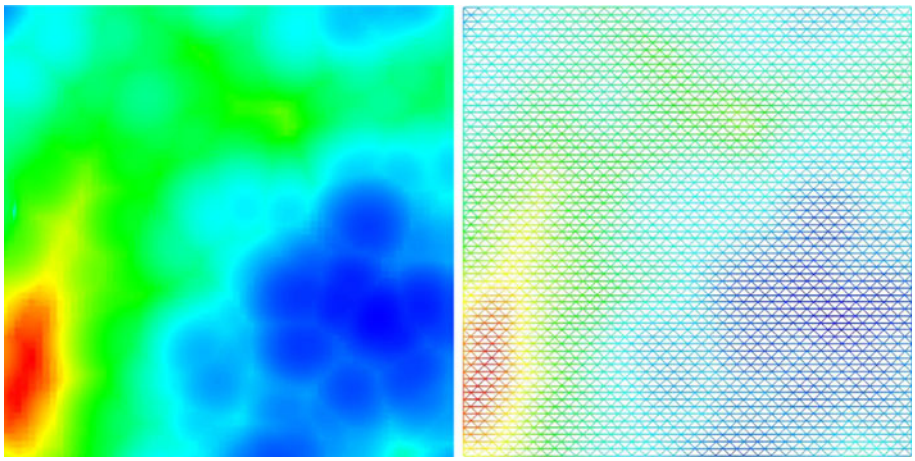

Fig. 7 Application of volumic MPU to tetrahedral-grid data of plasma: ray-casting visualization (*left*) and grid structure (*right*)

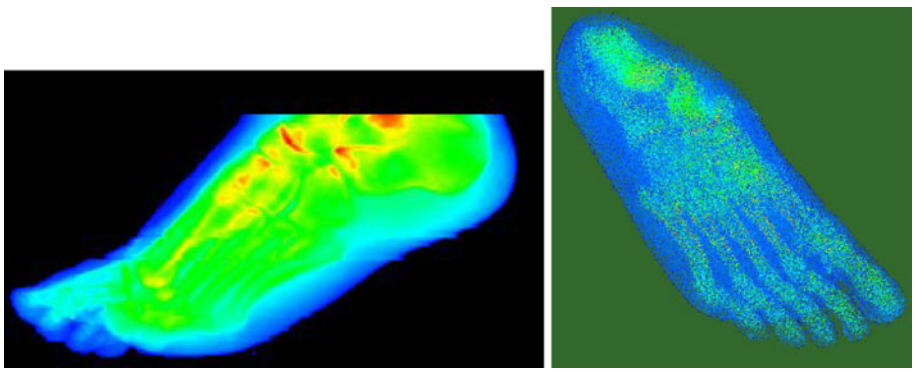

Fig. 8 Application of volumic MPU to non-grid (scattered voxel) data: ray-casting visualization (*left*) and input voxels (*right*)

In Fig. 9 (human head), we compare the visualization of the Laplacian field $|\nabla^2 f(\mathbf{x})|$ for regular-grid and irregular-grid data (scattered-voxel data similar to Fig. 8), describing the same medical data. The visualization for the irregular-grid data (right) is similar to and consistent with that for the regular-grid data (left). The fitting accuracy appears to be slightly better for the regular-grid data.

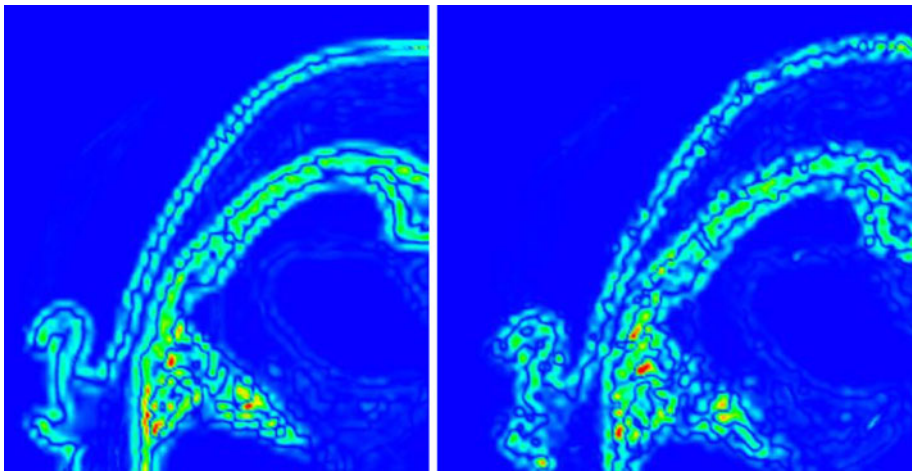

Fig. 9 Comparison of visualizing slices of Laplacian field $|\nabla^2 f(\mathbf{x})|$ of regular-grid data (*left*) and irregular-grid data (scattered voxel data) (*right*)

5 Effects of data compression

A characteristic feature of the volumic MPU is that it is not necessary to store input voxels after creation of the scalar field $f(\mathbf{x})$. The information that should be stored in order to describe the created $f(\mathbf{x})$ is as follows: (1) coefficients of $Q_i(\mathbf{x})$, and (2) position and widths of the compact support, for each of the created MPU cell. If the data size of the stored information is smaller than that of the input voxels, the volumic MPU can be used as a data-compression technique for lossy compression. In the following, we investigate the volumic MPU from this point of view.

We have investigated the data-compression effect for the two regular-grid data of the well-known 'hydrogen' and 'tornado" data. The results are summarized in Fig. 10, which shows that the original data are successfully compressed in the created $f(\mathbf{x})$. The figure also shows that the data-compression effect becomes stronger as the input grid size increases. For example, for the $1{,}024^3$-grid data, the data are compressed to approximately 1% of the original data, because the data size of $f(\mathbf{x})$ is not determined by the input grid size, but rather by the complexity of the input volume data itself. Therefore, data size of $f(\mathbf{x})$ does not increase in proportion to the input grid size. This advantageous feature comes from the grid independency of the volumic MPU.

We also investigated the data-compression effect for irregular-grid data. We created random-voxel data similar to that shown in Fig. 8, resampling the regular-grid data of 'hydrogen' and 'tornado', and used these data for the study. The results are summarized in Fig. 11. The figure shows that the data are successfully compressed, and that the compression effect is stronger than the regular-grid data (compare Fig. 11 with Fig. 10). The reason for the improvement is that irregular-grid data must include explicit information on voxel positions, which is not necessary for regular-grid data. The grid independency of the volumic MPU keeps data size of the created scalar field $f(\mathbf{x})$ small, even if the input is replaced with irregular-grid data.

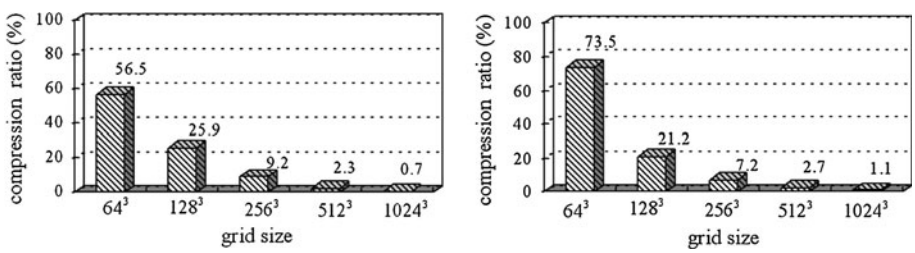

Fig. 10 Data compression for regular-grid data of 'hydrogen' (*left*) and 'tornado' (*right*)

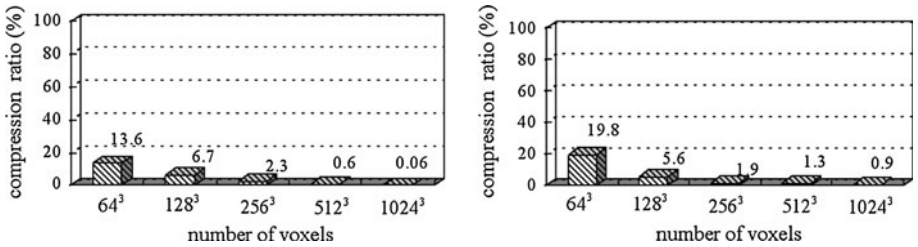

Fig. 11 Data compression for irregular-grid data of 'hydrogen' (*left*) and 'tornado' (*right*)

6 Time to evaluate created scalar field $f(\mathbf{x})$

One of the advantages of the volumic MPU is quick evaluation of the created scalar field $f(\mathbf{x})$. This is possible because only a few MPU cells, whose compact supports include the given position $\mathbf{x}$, contribute to the evaluation of $f(\mathbf{x})$. To use this locality, we divide the total space into a hypothetical grid consisting of small cubic areas. Note that this hypothetical grid is different from the input volume-data grid. Each ellipsoidal/rectangular support of an MPU cell is 'cached' to cubic unit areas that intersect the support. This pre-processing enables the traversing of the entire MPU cell tree to be skipped and realize quick evaluation of $f(\mathbf{x})$.

We compared the time to evaluate $f(\mathbf{x})$ created by the volumic MPU with the time to evaluate the trilinear and Spline interpolations. The trilinear interpolation is fast, whereas the Spline interpolation is of high quality. For each method, we measured the total time required to evaluate the fitted/interpolated scalar field at all voxels. We used a 64-bit Linux work station with an Intel Xeon X5482 CPU with 3.20 GHz clock. The results are summarized in Table 1. The volumic MPU is approximately twice as fast as the Spline interpolation. Moreover, the volumic MPU is as fast as the trilinear interpolation, especially for the case of using the rectangular support.

One disadvantage of the volumic MPU is that we need to create the MPU cell tree beforehand. For the case of the 512^3 'hydrogen' data, this pre-processing required 1,194.9 seconds with the rectangular support and 1,347.6 s with the ellipsoidal support. For the case of the 512^3 'tornado' data, this pre-processing required 2,239.4 s with the rectangular support and 2,399.9 s with the ellipsoidal support. For the case of the ocean-flow data, this pre-processing required 6,596.7 s with the rectangular support. Although the pre-processing does take time, once the MPU-cell tree has been created, $f(\mathbf{x})$ can be evaluated quickly. Moreover, we can reuse the created MPU cell tree for, e.g., visualization using different camera angles. This is because the MPU cell tree fully describes the created 3D scalar filed $f(\mathbf{x})$, being independent of any rendering parameters.

7 Conclusions

In the present paper, we have proposed the volumic version of the multi-level partition of unity (volumic MPU). The volumic MPU fits the scalar values with good precision to generate a scalar field that is continuous up to second-order differentiation. The volumic MPU, being independent of the grid structures of the input volume data, is applicable to irregular-grid data as well as regular grid data. The volumic MPU can also be used as an effective data-compression technique, especially, for irregular-grid data. The created scalar field can be evaluated rather quickly. Roughly speaking, the volumic MPU is a few times faster than the Spline interpolation and slightly slower than the trilinear interpolation. We applied the volumic MPU to large-scale regular-grid data of the ocean flow, medical data, tetrahedral-grid data, and random-voxel data, and we successfully demonstrated the effectiveness of the volumic MPU.

Table 1 Time to evaluate fitted/interpolated field of 512^3 'hydrogen' and 'tornado" data sets

Method	Time (s) for hydrogen	Time (s) for tornado
Trilinear interpretation	74.7	76.0
Volumic MPU (rect. support)	83.2	87.6
Volumic MPU (ellip. support)	105.6	111.4
Spline interpretation	188.8	190.6

We are working to apply the volumic MPU to large scale irregular-grid data, such as tetrahedral-grid data of FEM simulation. In addition, we intend to develop a parallel evaluation method of the created scalar field $f(\mathbf{x})$. As the parallel-processing units, the cubic areas of the hypothetical grid to cache the compact supports should be useful.

Acknowledgments The authors thank Prof. K. Koyamada for valuable comments. They also thank T. Kaneko for his cooperation in experiments.

References

Hansen CD, Johnson CR (2004) The visualization handbook. Elsevier, New York

Jang Y, Weiler M, Hopf M, Huang J, Ebert DS, Gaither KP, Ertl T (2004) Interactively visualizing procedually encoded scalar fields. In: Proceedings of Eurographics/IEEE TCVG symposium on visualization VisSym (Eurographics Association), pp 35–44

Lancaster P, Salkauskas K (1981) Surfaces generated by moving least squares methods. Math Comput 37(155):141–158

Lee S, Wolberg G, Shin SY (1997) Scattered data interpolation with multilevel B-splines. IEEE Trans Vis Comput Graphics 3(3):228–244

Marschner SR, Lobb RJ (1994) An evaluation of reconstruction filters for volume rendering. Proc IEEE Vis 100–107

Ohtake Y, Belyaev A, Alexa M, Turk G, Seidel H-P (2003) Multi-level partition of unity implicits. In: Proceedings of SIGGRAPH 2003, ACM Transactions on Graphics, vol 22(3), pp 463–470

Savchenko VV, Pasko AA, Okunev OG, Kunii TL (1995) Function representation of solids reconstructed from scattered surface points and contours. Comput Graphics Forum 14(4):181–188

Tsukamoto Y, Kataoka S, Hasegawa K, Nakata S, Tanaka S (2009) Data interpolation independent of grid structure using a volumic version of MPU. In: Proceedings of Asia simulation conference 2009 (CD-ROM), Shiga, Japan

Weiler M, Botchen RP, Stegmaier S, Huang J, Jang Y, Ebert D, Gaither K, Ertl T (2005) Hardware-assisted feature analysis and visualization of procedurally encoded multifiled volumetric data. Comput Graphics Appl 25(5):72–81