 CMES, vol.107, no.3, pp.187-199, 2015

Fast Generation of Smooth Implicit Surface Based on Piecewise Polynomial

Taku Itoh[1] **and Susumu Nakata**[2]

Abstract: To speed up generating a scalar field $g(\boldsymbol{x})$ based on a piecewise polynomial, a new method for determining field values that are indispensable to generate $g(\boldsymbol{x})$ has been proposed. In the proposed method, an intermediate for generating $g(\boldsymbol{x})$ does not required, i.e., the field values can directly be determined from given point data. Numerical experiments show that the computation time for determining the field values by the proposed method is about 10.4–12.7 times less than that of the conventional method. In addition, on the given points, the accuracy of $g(\boldsymbol{x})$ obtained by using the proposed method is almost the same as that of the conventional method. Furthermore, the computation time of the proposed method is almost not affected by the number of given points for the case where the number of all cells is large.

Keywords: Implicit Surface, Piecewise Polynomial, Meshless Methods, Computer Graphics, B-spline, Surface Reconstruction

1 Introduction

Meshless methods such as the element-free Galerkin method [Belytschko, Lu, and Gu (1994)], the meshless local Petrov-Galerkin method [Atluri and Zhu (1998)], the meshless radial point interpolation method [Wang and Liu (2002)], and the boundary node method [Mukherjee and Mukherjee (1997)] have been investigated as numerical methods to solve partial differential equations. In meshless methods, elements representing a geometrical structure are not required; however, an analysis domain must be defined. To define the analysis domain, a scalar field $g(\boldsymbol{x})$ that contains the analysis domain is sometimes employed [Nakata and Sakamoto (2014); Hasegawa, Nakata, and Tanaka (2006); Sakamoto, Nakata, and Tanaka (2012); Nakata, Hasegawa, and Tanaka (2009); Itoh, Saitoh, Kamitani, and Nakamura (2011)]. The scalar field $g(\boldsymbol{x})$ can be obtained as a result of implicit surface

[1] Nihon University, Narashino, Chiba, Japan.
[2] Ritsumeikan University, Kusatsu, Siga, Japan.

modeling, and the boundary of the analysis domain is represented as an implicit surface, $g(\boldsymbol{x}) = 0$. One of the main purposes of the implicit surface modeling is surface reconstruction from a set of surface points retrieved using three-dimensional scanning devices. In the last two decades, a number of methods for generating scalar functions of implicit surfaces have been proposed. In addition, recent studies such as reconstruction methods based on frequency analysis [Manson, Petrova, and Schaefer (2008)], Poisson equation [Kazhdan and Hoppe (2013)], splines [Wang, Yang, Jin, Deng, and Chen (2011)], radial basis functions (RBF) [Zagorchev and Goshtasby (2012)] and partition of unity [Ohtake, Belyaev, Alexa, Turk, and Seidel (2003)] have shown that accurate and detailed surfaces can be reconstructed from millions of input points.

Recently, a new method for generating $g(\boldsymbol{x})$ has been proposed [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)]. By using the method, an implicit surface can be rendered in real-time on graphical processing unit (GPU). In this method, to enable a real-time rendering of an implicit function, $g(\boldsymbol{x})$ is constructed as a set of piecewise polynomials based on the B-spline. Note that, to generate $g(\boldsymbol{x})$ in this method, field values f_{ijk} on N^3 uniform grid points $\boldsymbol{x}_{ijk}$ are indispensable. In [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)], the field values f_{ijk} are obtained by evaluating a smooth scalar field $f(\boldsymbol{x})$, that is generated by a method such as RBF or multi-level partition of unity (MPU) method [Ohtake, Belyaev, Alexa, Turk, and Seidel (2003)], on each of grid points $\boldsymbol{x}_{ijk}$. Namely, for generating $g(\boldsymbol{x})$, $f(\boldsymbol{x})$ is required as an intermediate of $g(\boldsymbol{x})$.

The purpose of the present study is to speed up generating a scalar field $g(\boldsymbol{x})$ based on a piecewise polynomial described in [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)]. To this end, a new method for determining field values f_{ijk} has been proposed. In the proposed method, any intermediates for generating $g(\boldsymbol{x})$ are not required, i.e., the field values can directly be determined from given point data.

2 Smooth Implicit Surface Based on Piecewise Polynomial

In this section, we briefly describe procedures for generating a scalar field $g(\boldsymbol{x})$ based on a piecewise polynomial. An implicit surface is represented as $g(\boldsymbol{x}) = 0$, where $\boldsymbol{x} = [x, y, z]^{\mathrm{T}}$. In addition, $g(\boldsymbol{x})$ has properties as follows:

$$\begin{cases} g(\boldsymbol{x}) > 0 & (\text{inside of the surface}), \\ g(\boldsymbol{x}) < 0 & (\text{outside of the surface}). \end{cases} \tag{1}$$

In the proposed method, a scalar field is generated, so that Eq. (1) is satisfied.

2.1 Generation of Scalar Field

In this section, we consider generating a scalar field $g(\boldsymbol{x})$ in a unit cube, $[0,1) \times [0,1) \times [0,1)$. Note that the unit cube is constructed by N^3 uniform cubic cells, where N is the number of cells for each direction. Here, we assume that the scalar field $g(\boldsymbol{x})$ can be represented as the following linear combination of B-spline bases [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)]:

$$g(\boldsymbol{x}) = \sum_{i=-1}^{N} \sum_{j=-1}^{N} \sum_{k=-1}^{N} c_{ijk} b_i(x) b_j(y) b_k(z), \ \boldsymbol{x} = [x,y,z]^{\mathrm{T}} \in [0,1) \times [0,1) \times [0,1), \quad (2)$$

where $b_i(t)$ is an uniform quadratic B-spline function defined as

$$\begin{aligned} b_i(t) &= b(t-i), \\ b(t) &\equiv \begin{cases} (t+1)^2/2 & (-1 \le t < 0), \\ -t^2+t+1/2 & (0 \le t < 1), \\ (t-2)^2/2 & (1 \le t < 2), \\ 0 & (\text{otherwise}). \end{cases} \end{aligned} \quad (3)$$

In addition, c_{ijk} are the coefficients determined to satisfy the interpolation condition, $g(\boldsymbol{x}_{ijk}) = f_{ijk}$. The local polynomial $g_{ijk}(\boldsymbol{x})$ in the (i,j,k)-th cell can be retrieved from Eqs. (2) and (3) as follows [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)]:

$$g_{ijk}(\boldsymbol{x}) = \sum_{p=0}^{2} \sum_{q=0}^{2} \sum_{r=0}^{2} \alpha_{ijk}^{(p,q,r)} x^p y^q z^r, \quad (4)$$

where $\alpha_{ijk}^{(p,q,r)}$ are the coefficients of the monomials, and i, j and k are indices of the cells in the x-, y- and z-direction, respectively. It must be noted here that, to generate $g(\boldsymbol{x})$ in Eq. (2), field values f_{ijk} on N^3 uniform grid points $\boldsymbol{x}_{ijk}$ are required. Here, $\boldsymbol{x}_{ijk} = [(i+0.5)/N, (j+0.5)/N, (k+0.5)/N]^{\mathrm{T}}$ that is contained in the (i,j,k)-th cell. In next section, we describe how to obtain the field values f_{ijk}.

3 Method for Obtaining Field Values f_{ijk}

3.1 Conventional Method

In [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)], the field values f_{ijk} are obtained by evaluating a smooth scalar field $f(\boldsymbol{x})$ on each of grid points $\boldsymbol{x}_{ijk}$. To generate $f(\boldsymbol{x})$, it is assumed that the point data is given as a collection of n scattered points $\mathscr{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_{\mathrm{P}}}\}$ in the unit cube, together with outward-directed

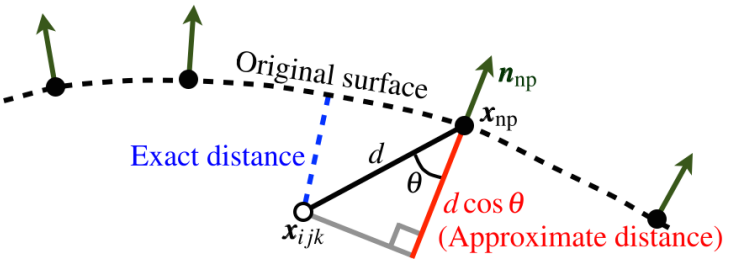


Figure 1: Schematic view for Eq. (6).

normals $\mathscr{N} = \{\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_{N_P}\}$ at each of given points, where N_P is the number of given points. Under the assumptions, the scalar field $f(\boldsymbol{x})$ is generated by a method such as RBF and MPU in the unit cube so that an original surface from which the given points $\mathscr{X}$ were obtained is reconstructed as the implicit representation, $f(\boldsymbol{x}) = 0$. Namely, before $g(\boldsymbol{x})$ in Eq. (2) is generated, $f(\boldsymbol{x})$ has to be generated first. In other words, $g(\boldsymbol{x})$ is generated by converting $f(\boldsymbol{x})$. Hence, $f(\boldsymbol{x})$ is an intermediate for generating $g(\boldsymbol{x})$.

3.2 Proposed Method

Under the same assumptions described in 3.1, we propose a new method for determining the field values f_{ijk}. In the proposed method, any intermediates are not required, i.e., f_{ijk} are directly obtained from given data, $\mathscr{X}$ and $\mathscr{N}$, without an explicit scalar field $f(\boldsymbol{x})$.

3.2.1 Determining f_{ijk} Directly

When the field values f_{ijk} are determined on N^3 uniform grid points $\boldsymbol{x}_{ijk}$, f_{ijk} have to satisfy the following conditions:

$$\begin{cases} f_{ijk} > 0 & \text{(inside of the surface)}, \\ f_{ijk} = 0 & \text{(on the surface)}, \\ f_{ijk} < 0 & \text{(outside of the surface)}. \end{cases} \tag{5}$$

Namely, f_{ijk} are determined so that a signed distance from the surface is emulated. This is because $g(\boldsymbol{x})$ is generated by using f_{ijk}, and has to satisfy Eq. (1). Although the surface is not obtained, we directly determine f_{ijk} from given data, $\mathscr{X}$ and $\mathscr{N}$, so that Eq. (5) is approximately satisfied. Note that a clue for determining f_{ijk} is $\mathscr{X}$, since $\mathscr{X}$ was obtained from an original surface by using an equipment such as a 3D laser scanner.

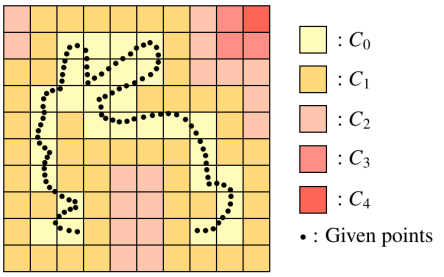


Figure 2: Levels of cells.

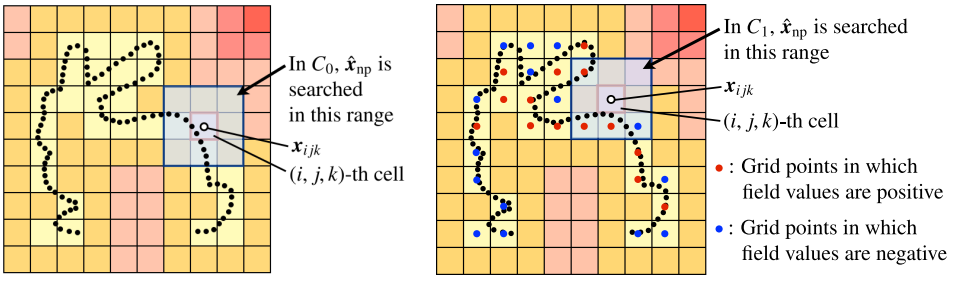


Figure 3: Range for searching an approximate nearest point $\hat{\boldsymbol{x}}_{\rm np}$ from $\boldsymbol{x}_{ijk}$ contained in (a) C_0 and (b) C_1. In C_0 and C_1, f_{ijk} on $\boldsymbol{x}_{ijk}$ are determined by Eq. (6).

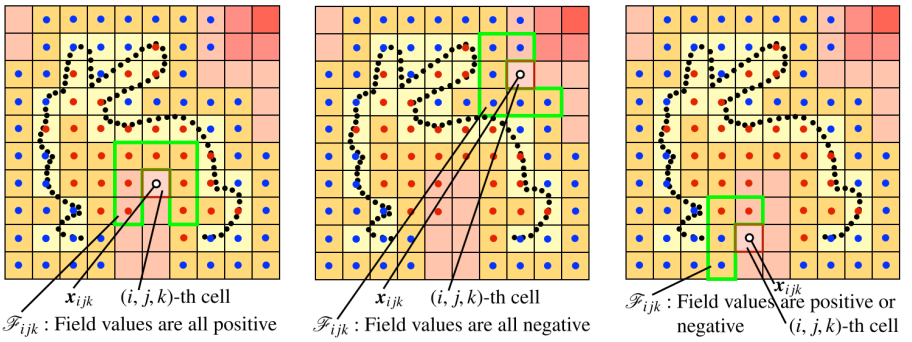
To determine f_{ijk} with satisfying Eq. (5) approximately, we adopt the following equation:

$$f_{ijk} = d\cos\theta = (\boldsymbol{x}_{\rm np} - \boldsymbol{x}_{ijk}) \cdot \boldsymbol{n}_{\rm np}, \tag{6}$$

where $d = |\boldsymbol{x}_{\rm np} - \boldsymbol{x}_{ijk}|$, and θ is defined as the angle between $\boldsymbol{x}_{\rm np} - \boldsymbol{x}_{ijk}$ and $\boldsymbol{n}_{\rm np}$. In addition, $\boldsymbol{x}_{\rm np} \in \mathscr{X}$ is the nearest point from $\boldsymbol{x}_{ijk}$, and $\boldsymbol{n}_{\rm np} \in \mathscr{N}$ is the outward-directed normal corresponding to $\boldsymbol{x}_{\rm np}$. As shown in Fig. 1, f_{ijk} determined by Eq. (6) can approximate the exact distance between $\boldsymbol{x}_{ijk}$ and an original surface.

3.2.2 Speed-up of Determining f_{ijk}

If the field values f_{ijk} are determined by Eq. (6) on all grid points $\boldsymbol{x}_{ijk}$, the computation time for determining all f_{ijk} may be large in comparison with that of [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)], even though an efficient algorithm such as an octree based method is employed for searching the nearest point. For this reason, to speed up determining f_{ijk}, we consider employing Eq. (6) only on the grid points contained in C_0 and C_1, where C_ℓ denotes a set of cells whose level is $\ell (\ell = 0, 1, \dots)$. Here, we define a level of cells as follows:



(a) (b) (c)

Figure 4: Schematic view for the case where field values on grid points contained in $\mathscr{F}_{ijk}$ are (a) all positive (Type A), (b) all negative (Type B), and (c) positive or negative (Type C). Here, red and blue dots are the same as described in Fig. 2(b).

1. The level of cells that contain given points is 0.

2. As shown in Fig. 2, the level of cells that are next to C_ℓ is $\ell+1 (l=0,1,\dots)$,

The field values f_{ijk} are determined in order of increasing the level of cells. On $\boldsymbol{x}_{ijk}$ contained in C_0 and C_1, f_{ijk} is calculated by Eq. (6). Then, an approximate nearest point $\hat{\boldsymbol{x}}_{\rm np}$ from $\boldsymbol{x}_{ijk}$ can be found by searching (l,m,n)-th cells $(l=i-1,i,i+1; m=j-1,j,j+1; n=k-1,k,k+1)$ as shown in Figs. 3(a) and (b). To calculate f_{ijk} by Eq. (6), we consider $\boldsymbol{x}_{\rm np}=\hat{\boldsymbol{x}}_{\rm np}$, since the nearest point $\boldsymbol{x}_{\rm np}$ is exactly the same as $\hat{\boldsymbol{x}}_{\rm np}$ in almost all cases. In addition, even if $\hat{\boldsymbol{x}}_{\rm np}$ is not exactly the same as $\boldsymbol{x}_{\rm np}$, $\hat{\boldsymbol{x}}_{\rm np}$ is a good approximation of $\boldsymbol{x}_{\rm np}$. Note that the field values f_{ijk} calculated by Eq. (6) automatically satisfy Eq. (5).

In grid points contained in $C_\ell (\ell=2,3,\dots)$, f_{ijk} are calculated by using the field values that were already determined on neighbor grid points of $\boldsymbol{x}_{ijk}$. To satisfy Eq. (5) appropriately, we present the following three types of calculations.

Type A For the case where f_{lmn} on $\boldsymbol{x}_{lmn}$ contained in $\mathscr{F}_{ijk}$ are all positive as shown in Fig. 4(a):

$$f_{ijk}=\frac{\displaystyle\sum_{l=i-1}^{i+1}\sum_{m=j-1}^{j+1}\sum_{n=k-1}^{k+1}\beta_{lmn}\left(f_{lmn}+|\boldsymbol{x}_{lmn}-\boldsymbol{x}_{ijk}|\right)}{N_{ijk}^{\beta}}. \tag{7}$$

Type B For the case where f_{lmn} on $\boldsymbol{x}_{lmn}$ contained in $\mathscr{F}_{ijk}$ are all negative as shown

in Fig. 4(b):

$$f_{ijk} = \frac{\sum_{l=i-1}^{i+1} \sum_{m=j-1}^{j+1} \sum_{n=k-1}^{k+1} \beta_{lmn} \left(f_{lmn} - |\boldsymbol{x}_{lmn} - \boldsymbol{x}_{ijk}| \right)}{N_{ijk}^{\beta}}. \tag{8}$$

Type C Others, i.e., for the case where f_{lmn} on $\boldsymbol{x}_{lmn}$ contained in $\mathscr{F}_{ijk}$ are positive or negative as shown in Fig. 4(c):

$$f_{ijk} = \frac{\sum_{l=i-1}^{i+1} \sum_{m=j-1}^{j+1} \sum_{n=k-1}^{k+1} \beta_{lmn} f_{lmn}}{N_{ijk}^{\beta}}. \tag{9}$$

Here, $\mathscr{F}_{ijk}$ is defined as a set of cells in which f_{lmn} were already determined on $\boldsymbol{x}_{lmn} (l = i-1, i, i+1; m = j-1, j, j+1; n = k-1, k, k+1)$. In addition,

$$\beta_{lmn} \equiv \begin{cases} 0 & (\boldsymbol{x}_{lmn} \text{ is not contained in } \mathscr{F}_{ijk}) \\ 1 & (\boldsymbol{x}_{lmn} \text{ is contained in } \mathscr{F}_{ijk}) \end{cases}, \tag{10}$$

and

$$N_{ijk}^{\beta} = \sum_{l=i-1}^{i+1} \sum_{m=j-1}^{j+1} \sum_{n=k-1}^{k+1} \beta_{lmn}. \tag{11}$$

4 Numerical Experiments

In this section, numerical experiments are conducted to evaluate the proposed method by using the data of Bunny, Armadillo and Lucy models as shown in Figs. 5(a), (b) and (c), respectively. All models are first rescaled so that the given points are lie in a unit cube $[0,1) \times [0,1) \times [0,1)$. Computations were performed on a computer equipped with a 3.4 GHz Intel Core i7 4930K processor, 32 GB RAM, CentOS Linux ver. 6.5, and g++ ver. 4.4.7 with single-precision arithmetic.

Let us first investigate a reconstructed result represented as $g(\boldsymbol{x}) = 0$ for each model. For $N = 400$, the results obtained by using the proposed method are shown in Figs. 5(d), (e) and (f) for Bunny, Armadillo and Lucy models, respectively. Note that these results are obtained by an implicit surface polygonizer [Bloomenthal and Ferguson (1995)].

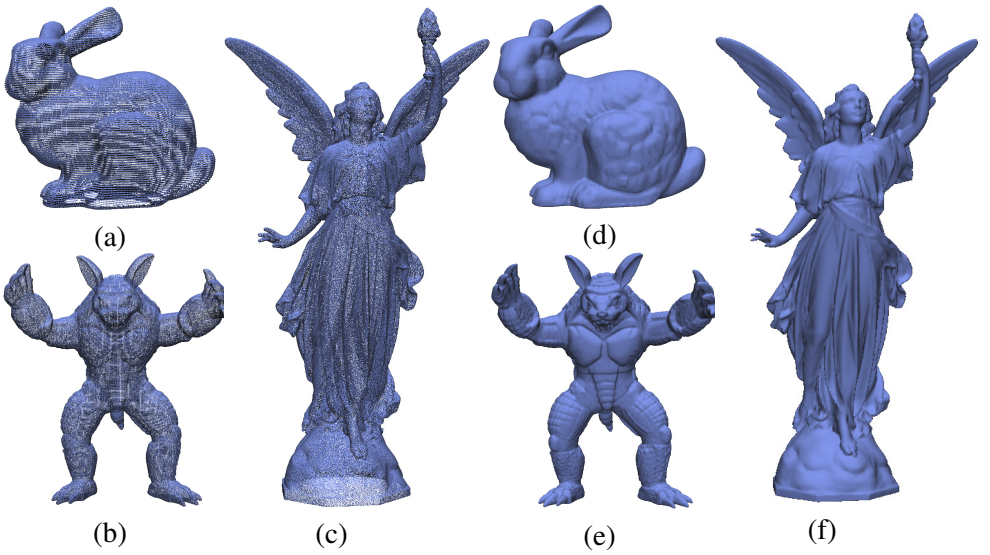


Figure 5: Left side: Given point data used in the numerical experiments: (a) Bunny ($N_P = 34834$), (b) Armadillo ($N_P = 172974$) and (c) Lucy ($N_P = 1001991$). Right side: Reconstructed Results obtained by using the proposed method for (d) Bunny, (e) Armadillo and (f) Lucy ($N = 400$).

To investigate the accuracy of these results quantitatively, an average error ε_{avg} of $g(\boldsymbol{x})$ obtained by using the proposed method is compared with that of the conventional method. Here, the average error is defined by $\varepsilon_{avg} \equiv \sum_{n=1}^{N_P} |g(\boldsymbol{x}_n)| / N_P$. If $g(\boldsymbol{x})$ is accurate on the given points, $\varepsilon_{avg} \to 0$, since $g(\boldsymbol{x})$ is generally generated so that $g(\boldsymbol{x}_n) = 0$ $(n = 1, 2, \ldots, N_P)$ are satisfied. The average errors of $g(\boldsymbol{x})$ obtained by both methods for Bunny, Armadillo and Lucy models are plotted as a function of the number N of cells for each direction in Figs. 6(a), (b) and (c), respectively. We see from these figures that, for the case where N is relatively large, the average error of the proposed method is smaller than that of the conventional method. However, we consider that the difference between the average errors obtained by both methods is not large. Hence, we conclude that, on the given points, the accuracy of $g(\boldsymbol{x})$ obtained by using the proposed method is almost the same as that of the conventional method.

Note that, as shown in Fig. 6(d), some big holes can be found in the bottom part of Bunny model. Regardless of the accuracy of $g(\boldsymbol{x})$ on given points, the surface obtained by the conventional method becomes unnatural around big holes of given data as shown in Fig. 6(e) for Bunny model. The proposed method also has the same property as shown in Fig. 6(f). It must be noted here that not only above property but also other properties of the conventional method are inherited to the

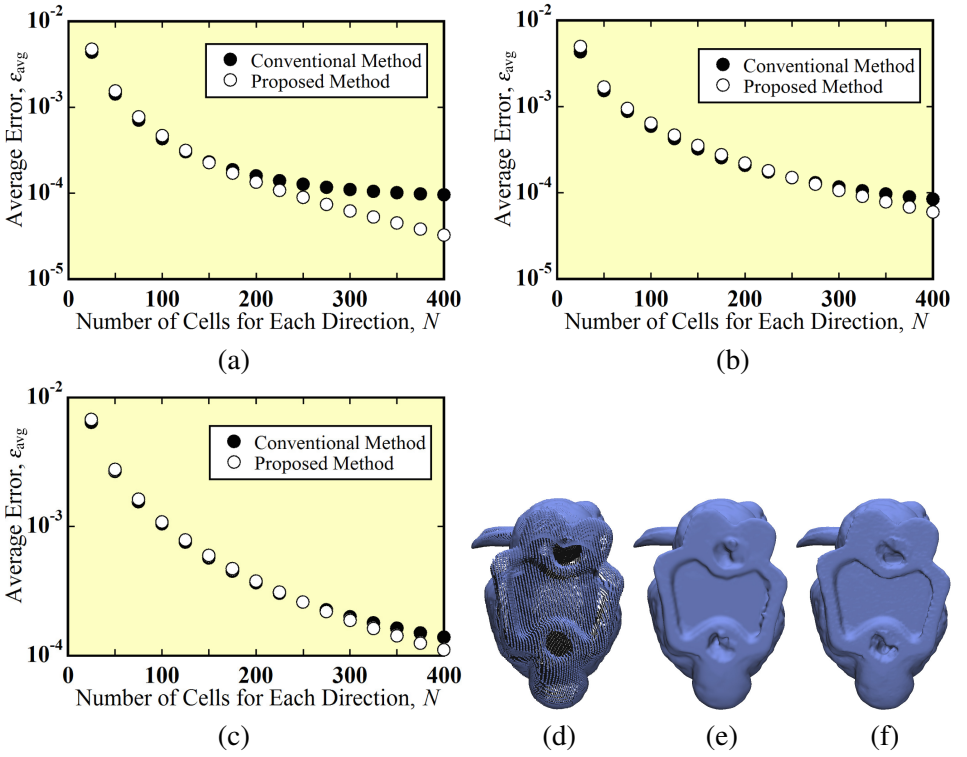


Figure 6: Dependence of average error ε_{avg} of $g(\boldsymbol{x})$ on the given points for (a) Bunny, (b) Armadillo and (c) Lucy models on the number N of cells for each direction. (d) Some big holes can be found in the bottom part of Bunny model. Regardless of accuracy of $g(\boldsymbol{x})$ on the given points, the surfaces obtained by (e) conventional method and (f) proposed method become unnatural around big holes of given points.

proposed method, since the difference between conventional and proposed methods is only how to obtain the field values. Other process to generate $g(\boldsymbol{x})$ in the proposed method is exactly the same as that of the conventional method. Hence, $g(\boldsymbol{x})$ obtained by using the proposed method can be rendered in real-time on GPU by the same algorithm described in [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)], though the results shown in Figs. 5(d), (e) and (f) are rendered by the implicit surface polygonizer as mentioned above.

Finally, we investigate computation time of obtaining all field values by using both methods. For Bunny, Armadillo and Lucy models, the computation time by both methods is plotted as a function of the number N of cells for each direction in Figs. 7(a), (b) and (c), respectively. Note that, for generating $f(\boldsymbol{x})$ in the con-

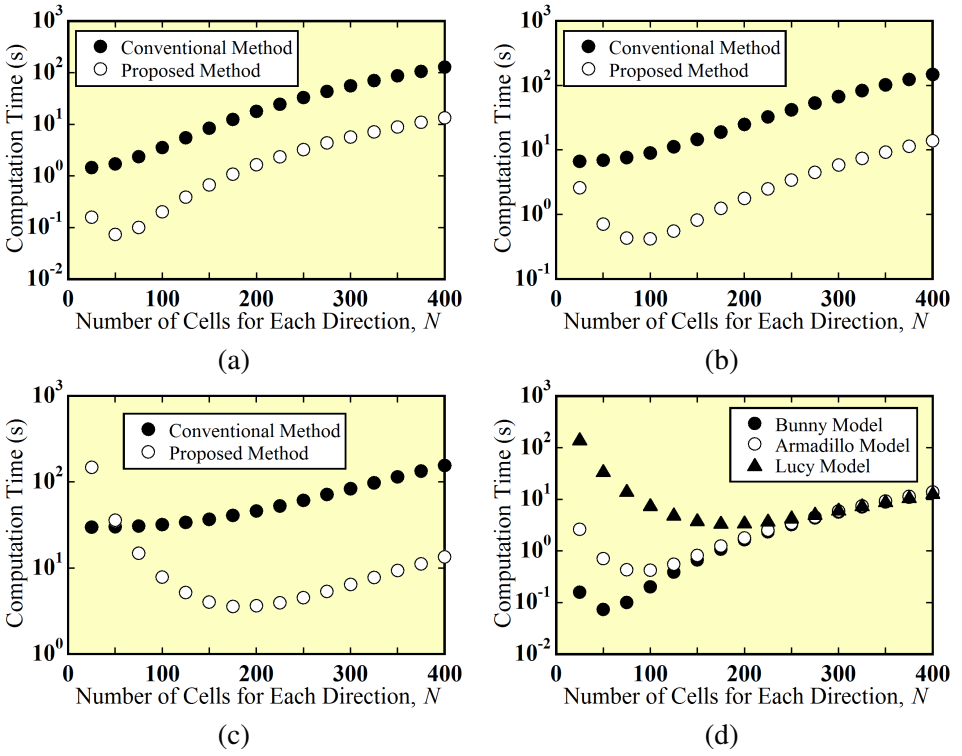
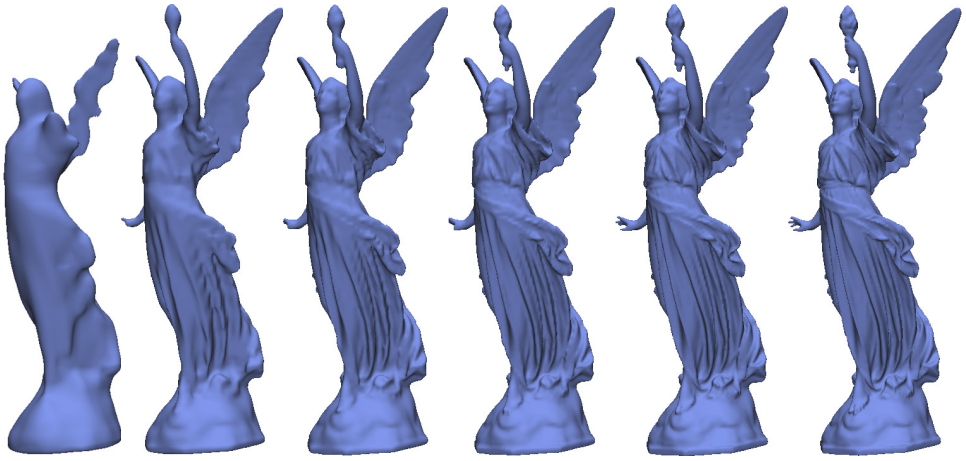


Figure 7: Dependence of computation time to obtain all field values for (a) Bunny, (b) Armadillo and (c) Lucy models on the number N of cells for each direction. The computation time of the proposed method for each model is summarized in (d).

ventional method, we employ the MPU method whose parameters as described in [Ohtake, Belyaev, Alexa, Turk, and Seidel (2003)] are fixed as $\alpha = 1.35$, $\lambda = 0.2$ and $\varepsilon_0 = 10^{-3}$. In addition, the computation time of the conventional method in Fig. 7 contains the computation time of generating $f(\boldsymbol{x})$, since this is indispensable in the conventional method to obtain the field values. We see from Figs. 7(a) and (b) that the computation time of the proposed method is always less than that of the conventional method. In addition, we see from Fig. 7(c) that, for $N \geq 75$, the computation time of the proposed method is less than that of the conventional method. Here, for $N = 50, 100, 150, 200, 250$ and 300, the results obtained by using the proposed method for Lucy model are shown in Figs. 8(a), (b), (c), (d), (e) and (f), respectively. From these figures, in the following, we consider comparing the computation time of both methods for $N \geq 150$, since the reconstructed results are insufficient for the case where N is relatively small. On average for $N \geq 150$, the computation time of the proposed method for Bunny, Armadillo and Lucy models



(a) (b) (c) (d) (e) (f)

Figure 8: Reconstructed results of Lucy model for (a) $N = 50$, (b) $N = 100$, (c) $N = 150$, (d) $N = 200$, (e) $N = 250$ and (f) $N = 300$.

is about 10.4, 12.7 and 12.2 times less than that of the conventional method, respectively. In addition, as shown in Fig. 7(d), the computation time of three models is almost the same for $N \geq 300$. This is because almost all field values are calculated by Eqs. (7), (8) and (9) for the case where the number N^3 of all cells is large. Namely, the total number of C_0 and C_1 is very small in comparison with that of $C_\ell (\ell = 2, 3, \dots)$. Hence, the process for searching the nearest point $\boldsymbol{x}_{\mathrm{np}}$ in Eq. (6) does not required in almost all cells. Since Eqs. (7), (8) and (9) do not use the given points, the computation time of the proposed method is almost not affected by the number N_{P} of given points for the case where the number N^3 of all cells is large.

5 Discussion

In this section, we discuss the applicability of the scalar field $g(\boldsymbol{x})$ generated by the proposed method to the meshless based analysis.

A scalar field $g(\boldsymbol{x})$ generated by conventional method has sometimes been employed in the meshless methods to define an analysis domain as an implicit surface, $g(\boldsymbol{x}) = 0$. For example, with the scalar field $g(\boldsymbol{x})$ generated by conventional method, the structural analysis was done by the element-free Galerkin method (EFG) in [Hasegawa, Nakata, and Tanaka (2006)], and a fluid simulation was done by the smoothed particle hydrodynamics method (SPH) in [Nakata and Sakamoto (2014)].

In meshless methods, $g(\boldsymbol{x})$ fulfills roles both defining the analysis domain and rec-

ognizing inside/outside of the analysis domain by using the properties as described in Eq. (1). Concretely, to recognize inside/outside of the analysis domain, evaluation of $g(\boldsymbol{x})$ is required. This is because, in the meshless based analysis, there are indispensable procedures such as evaluation of integrals inside the domain and confining particles to the domain. By using the properties as described in Eq. (1), these procedures can be executed easily. Note that, in the proposed method, $g(\boldsymbol{x})$ is generated so that Eq. (1) is satisfied.

The accuracy of $g(\boldsymbol{x})$ generated by the proposed method is almost the same as that generated by the conventional method as shown in Section 4. In addition, the computation time for evaluating $g(\boldsymbol{x})$ of the proposed method is almost equal to that of the method in [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)]. One of the main properties of the method in [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)] is parallel evaluation of $g(\boldsymbol{x})$. Note that $g(\boldsymbol{x})$ generated by the proposed method has the same properties, since the proposed method is based on the method in [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)]. Hence, we consider that $g(\boldsymbol{x})$ generated by the proposed method can be applied to the meshless based analysis as well as that generated by the conventional methods. In addition, by using the proposed method, the meshless based analysis may be more efficient because of the parallel evaluation of $g(\boldsymbol{x})$.

6 Conclusion

To speed up generating a scalar field $g(\boldsymbol{x})$ based on a piecewise polynomial described in [Nakata, Aoyama, Makino, Hasegawa, and Tanaka (2012)], a new method for determining field values that are indispensable to generate $g(\boldsymbol{x})$ has been proposed. In the proposed method, any intermediates for generating $g(\boldsymbol{x})$ are not required, i.e., the field values can directly be determined from given point data that consists of point locations together with outward-directed normals. In numerical experiments, by using the data of Bunny, Armadillo and Lucy models, the performance of the proposed method has been investigated by comparing with that of the conventional method. Conclusions obtained in the present study are summarized as follows:

1. On the given points, the accuracy of $g(\boldsymbol{x})$ obtained by using the proposed method is almost the same as that of the conventional method.

2. The process for determining the field values by the proposed method is faster than that of the conventional method. In the numerical experiments, the computation time for determining the field values by the proposed method is about 10.4–12.7 times less than that of the conventional method.

3. For the case where the number of all cells is large, the computation time of the proposed method is almost not affected by the number of given points.

In future study, the meshless methods with the implicit functions generated by using the proposed method will be applied for solving partial differential equations in complex shaped domains.

Acknowledgement: We would like to thank the Stanford 3D Scanning Repository for Bunny, Armadillo and Lucy models. This work was partially supported by JSPS KAKENHI Grant Number 24700053.

References

Atluri, S. N.; Zhu, T. (1998): A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics. *Comput. Mech.*, vol. 22, pp. 117–127.

Belytschko, T.; Lu, Y. Y.; Gu, L. (1994): Element-free Galerkin methods. *Int. J. Numer. Methods Eng.*, vol. 37, pp. 229–256.

Bloomenthal, J.; Ferguson, K. (1995): Polygonization of non-manifold implicit surfaces. In *Proceedings of ACM SIGGRAPH 95, Computer Graphics Proceedings, Annual Conference Series*, pp. 309–316.

Hasegawa, K.; Nakata, S.; Tanaka, S. (2006): Meshless structural analyses of complex shape models using implicit surface representations. *Journal of Plasma Physics*, vol. 72, no. 6, pp. 1081–1086.

Itoh, T.; Saitoh, A.; Kamitani, A.; Nakamura, H. (2011): Efficient evaluation of influence coefficients in three-dimensional extended boundary-node method for potential problems. *Plasma and Fusion Research*, vol. 6, 2401106.

Kazhdan, M.; Hoppe, H. (2013): Screened poisson surface reconstruction. *ACM Transactions on Graphics*, vol. 32, no. 3, article 29.

Manson, J.; Petrova, G.; Schaefer, S. (2008): Streaming surface reconstruction using wavelets. *Computer Graphics Forum*, vol. 27, no. 5, pp. 1411–1420.

Mukherjee, Y. X.; Mukherjee, S. (1997): The boundary node method for potential problems. *Int. J. Numer. Methods Eng.*, vol. 44, pp. 797–815.

Nakata, S.; Aoyama, S.; Makino, R.; Hasegawa, K.; Tanaka, S. (2012): Real-time isosurface rendering of smooth fields. *Journal of Visualization*, vol. 15, no. 2, pp. 179–187.

Nakata, S.; Hasegawa, K.; Tanaka, S. (2009): Meshfree structural analysis using the modified radial point interpolation method. In *Proceedings of Civil-Comp*, Paper 102, Funchal, Portugal.

Nakata, S.; Sakamoto, Y. (2014): Particle-based parallel fluid simulation in three-dimensional scene with implicit surfaces. In *Proceedings of CMMSE 2014*, pp. 1367–1376, Cadiz, Spain.

Ohtake, Y.; Belyaev, A.; Alexa, M.; Turk, G.; Seidel, H.-P. (2003): Multi-level partition of unity implicits. *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 463–470.

Sakamoto, Y.; Nakata, S.; Tanaka, S. (2012): Particle based fluid simulation for 3D scene with implict surfaces. In *Proceedings of JSST International Conference on Simulation Technology*, pp. 65–71, Kobe, Japan.

Wang, J.; Yang, Z.; Jin, L.; Deng, J.; Chen, F. (2011): Parallel and adaptive surface reconstruction based on implicit PHT-splines. *Computer Aided Geometric Design*, vol. 28, no. 8, pp. 463–474.

Wang, J. G.; Liu, G. R. (2002): A point interpolation meshless method based on radial basis functions. *Int. J. Numer. Meth. Engng*, vol. 54, no. 11, pp. 1623–1648.

Zagorchev, L.; Goshtasby, A. (2012): A curvature-adaptive implicit surface reconstruction for irregularly spaced points. *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 9, pp. 1460–1473.