Filling Holes in Large Polygon Models Using an Implicit Surface Scheme and the Domain Decomposition Method

Dong-Jin Yoo[1,#]

1 Department of Computer Aided Mechanical Design Engineering , Daejin University, Pocheon-Si, Kyeonggi-Do, South Korea
Corresponding Author / E-mail: djyoo@daejin.ac.kr, TEL: +82-031-539-2031, FAX: +82-031-539-1970

KEYWORDS: Hole filling, Implicit surface, Radial basis function, Domain decomposition method

A new approach based on implicit surface interpolation combined with domain decomposition is proposed for filling complex-shaped holes in a large polygon model. A surface was constructed by creating a smooth implicit surface from an incomplete polygon model through which the actual surface would pass. The implicit surface was defined by a radial basis function, which is a continuous scalar-value function over the domain $\mathbf{R}^3$*. The generated surface consisted of the set of all points at which this scalar function is zero. It was created by placing zero-valued constraints at the vertices of the polygon model. The well-known domain decomposition method was used to treat the large polygon model. The global domain of interest was divided into smaller domains in which the problem could be solved locally. The LU decomposition method was used to solve the set of small local problems; the local solutions were then combined using weighting coefficients to obtain a global solution. The validity of this new approach was demonstrated by using it to fill various holes in large and complex polygon models with arbitrary topologies.*

Manuscript received: January 25, 2006 / Accepted: February 3, 2006

1. Introduction

The hole filling process for large and complex incomplete polygon models with arbitrary topologies is the most important consideration in many fields related to the treatment of polygon models. It is well-known that a complete finite element model is required to obtain stable convergence and that unexpected holes bring about numerical difficulties in computer-aided engineering. Holes must be eliminated from polygon and finite element models because they cause many geometric modeling and manufacturing problems in fields such as computer graphics, reverse engineering and rapid prototyping.

Various methods have been suggested for filling complex-shaped holes in the area of computer graphics by creating an implicit surface from an unorganized cloud of points.[1-4] A previous method devised by the author[5] based on an elastic finite element analysis and a trimmed surface can be useful for filling various types of holes. However, the method is limited by its large computation time, and the implementation of the code is difficult due to the large stiffness matrix created by large holes. A hole filling method based on an implicit surface is faster and more robust.[6] However, when implicit surfaces are used, there is a remarkable discrepancy between the shapes of the neighboring elements and newly generated elements that requires a novel advanced method.

Many studies have attempted to treat large models composed of many polygons using an implicit surface scheme. Carr *et al.*[3] attempted to apply an implicit surface to various types of large polygon models using the fast multipole method suggested by Beatson. This approach, however, is not simple to implement due to a somewhat complex mathematical algorithm and the enormous amount of computation time required to treat large matrices. To overcome this problem, Kojekine *et al.*[7] proposed a more efficient numerical method by organizing the sparse matrix into a band-diagonal sparse matrix that could be solved more efficiently. However, the method was not robust for nonuniform distributions of points. Ohtake *et al.*[8] suggested a novel MPU implicit approach that reconstructs an implicit surface from unorganized data sets containing a huge number of points using weighted sums of different types of piecewise quadratic functions.

In the present study, a complete model without holes is reconstructed from a raw incomplete model using an implicit surface scheme and the domain decomposition method. An implicit surface is defined from the incomplete polygon model through which the actual surface will pass. A mesh refinement and smoothing scheme combined with a marching cube algorithm is also suggested to visualize the reconstructed implicit surface.

2. Mathematical description of an implicit surface

In this study, a surface is reconstructed by creating a smooth implicit surface from an incomplete polygon model using an efficient interpolation method. The new surface is faithful to the input polygon model, and many of the holes are eliminated by the process.

The equation of an implicit surface can be defined as [2,3]

$$f(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j \phi(\mathbf{x}-\mathbf{c}_j) + P(\mathbf{x}), \quad (1)$$

where c_j represents the locations of the constraints, λ_j indicates the weights, ϕ is the basis function, and $P(\mathbf{x})$ is a degree-one polynomial. The basis function for interpolation, ϕ, is the thin-plate radial basis function defined by

$$\phi(\mathbf{x}) = |\mathbf{x}|^2 \log(|\mathbf{x}|). \quad (2)$$

After substituting the interpolation constraint conditions into equation (1), the implicit surface interpolation problem can be written in the following form:

$$h_i = \sum_{j=1}^{N} \lambda_j \phi(\mathbf{c}_i - \mathbf{c}_j) + P(\mathbf{c}_i), \quad (3)$$

where $h_i = f(c_i)$ for $(1 \le i \le N)$. Here, h_i is the scalar function value at the i^{th} point. The function value is zero at points lying on the surface and nonzero at off-surface points:

$$h_i = f(\mathbf{c}_i) = 0, \; i = 1, \ldots, n \text{ (on-surface points)} \quad (4a)$$

$$h_i = f(\mathbf{c}_i) \neq 0, \; i=n+1, \ldots, N \text{ (off-surface points).} \quad (4b)$$

The weights λ_j must satisfy the following condition[3]:

$$\sum_{j=1}^{N} \lambda j = \sum_{j=1}^{N} \lambda j C_j^x = \sum_{j=1}^{N} \lambda j C_j^y = \sum_{j=1}^{N} \lambda j C_j^z = 0. \quad (5)$$

Substituting the constraint equations (3), (4), and (5) into equation (1), we obtain a linear system of equations in matrix form as follows:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} & \cdots & \phi_{1N} & 1 & C_1^x & C_1^y & C_1^z \\ \phi_{21} & \phi_{22} & \phi_{23} & \cdots & \phi_{2N} & 1 & C_2^x & C_2^y & C_2^z \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{N1} & \phi_{N2} & \phi_{N3} & \cdots & \phi_{NN} & 1 & C_N^x & C_N^y & C_N^z \\ 1 & 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ C_1^x & C_2^x & C_3^x & \cdots & C_N^x & 0 & 0 & 0 & 0 \\ C_1^y & C_2^y & C_3^y & \cdots & C_N^y & 0 & 0 & 0 & 0 \\ C_1^z & C_2^z & C_3^z & \cdots & C_N^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \\ P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (6)$$

where $\phi_{ij} = \phi(c_i - c_j)$. The unknowns λ_j and the coefficients of $P(\boldsymbol{x})$ can be obtained by solving the equations simultaneously using the appropriate constraint conditions as function values at the points, as shown in Fig. 1.

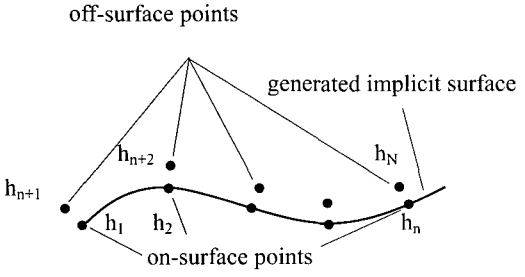


Fig. 1 Generation of the implicit surface

3. Filling holes using the domain decomposition method

3.1 Generation of a local implicit surface

The basic idea of the domain decomposition method (DDM) is to break up the 3D spatial domain occupied by the input polygon model into several sub-domains, interpolate the points data in each sub-domain separately, and then blend the local solutions together to obtain a global solution using appropriate smooth blending functions that sum to one everywhere in the domain. This is illustrated in Fig. 2. The DDM can be applied to a large polygon model consisting of millions of points to reduce the computation time required for interpolation and visualization. The number of points allocated to each sub-domain and the size of the overlapping zones between sub-domains are important factors that can be controlled by adjusting the size of each sub-domain. The adjusting scheme is an iterative process in which Ω_i is enlarged or reduced until the desired number of points is reached. In this study, the desired number of points was set between 30 and 200. For each sub-domain, a local implicit surface was built using the LU decomposition method by solving a linear system of equations defined by equation (6) of the previous section.

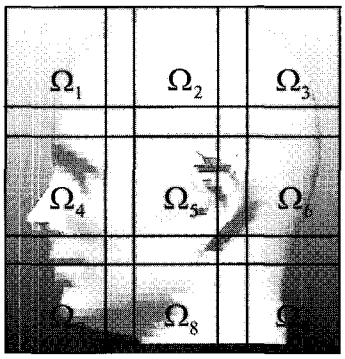


Fig. 2 Schematic diagram illustrating the concept of the domain decomposition method

3.2 Generation of a global implicit surface using a blending function

For each Ω_i, a local implicit surface equation is computed by solving the equations simultaneously. The global solution is then defined as a combination of the local solutions weighted by the partition of unity functions w_i as follows:

$$F(\mathbf{P}) = \sum_{i=1}^{ND} f_i(\mathbf{P}) w_i(\mathbf{P}) \quad (7a)$$

$$w_i(\mathbf{P}) = \frac{W_i(\mathbf{P})}{\sum_{j=1}^{ND} W_j(\mathbf{P})} \quad (7b)$$

The choice of weighting function determines the continuity between the local solutions f_i and the global solution F. In this study, a smooth blending function that ensures C^2 continuity over the domain Ω_i was introduced as follows:

$$d_i(\mathbf{P}) = 1 - \prod_{r \in x,y,z} \frac{4(\mathbf{P}_r - \mathbf{S}_r)(\mathbf{T}_r - \mathbf{P}_r)}{(\mathbf{T}_r - \mathbf{S}_r)^2} \quad (8a)$$

$$Wi(\mathbf{P}) = -6d_i^5 + 15d_i^4 - 10d_i^3 + 1 \quad (8b)$$

where $\mathbf{S}$ and $\mathbf{T}$ are the position vectors of two opposite corners of each sub-domain and Wi is the weighting coefficient of the i^{th} sub-domain.

3.3 Generation of a polygon model from the global implicit surface

Various visualization methods can be used to obtain a polygon model from the global implicit surface defined by equation (7).

Conventional techniques for visualizing implicit surface models include polygonization, ray tracing, and volume rendering. From these visualization methods, a mixed method including mesh refinement and smoothing was selected to visualize the resulting implicit surfaces. This is robust and fast because of its logically stable algorithm based on the well-known marching cube algorithm, which is the most common method used to create polygons for an implicit surface. The 3D space occupied by an implicit surface is divided into regular cells such as cubes. If the value of the implicit function takes on a mixture of positive and negative values at the corners of a given cube, then the implicit surface must pass through the cube. At such cubes, a small set of polygons can be created that approximate the shape of the surface within the cube, as shown in Fig. 3. If higher quality rendering is required, the size of the cubes must be reduced in order to describe the original implicit surface with sufficient accuracy. This implies an increase in computation time and more memory usage.

In this work, a coarse polygon model was generated using the marching cube algorithm. Then each triangle was refined into four triangles. Finally, a quality polygon model was obtained by smoothing the existing model and projecting it onto the implicit surface. The refinement and smoothing were repeated until the desired level of accuracy was attained. For the smoothing, a node was relocated to the averaged center of gravity of the neighboring polygons to improve the quality of the polygon model as follows:

$$\mathbf{P} = \frac{\sum_{i=1}^{n} A_i \mathbf{C}_i}{\sum_{i=1}^{n} A_i} \tag{9}$$

where $\mathbf{P}$ is the position vector of the new location, A_i is the area of the i^{th} triangle, $\boldsymbol{C}_i$ is the center of gravity of the i^{th} triangle, and n is the number of triangles connected with the concerned point.

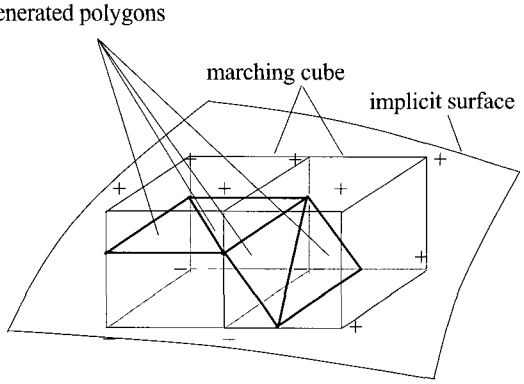


Fig. 3 Schematic diagram illustrating the concept of the marching cube algorithm

4. Results and Discussion

Various hole filling operations were performed for large and complex polygon models with arbitrary topologies to verify the effectiveness and validity of the proposed implicit surface scheme and domain decomposition method. The main part of the developed code, including the implicit surface interpolation and polygonization programs, were constructed using the C language on an IBM RS/6000 workstation, as shown in Table 1. The program for visualizing the polygon model was constructed using Visual C^{++} and OpenGL on a PC. Fig. 4 shows a flowchart of the details for generating an implicit surface, and Fig. 5 shows the whole process of generating the polygon model for visualization, including mesh

Table 1 Configuration of the developed system

Classification	Hardware	Software	
		O/S	Language
Reconstruction & polygonization	IBM RS / 6000 200 MHz Power CPU 128 MB RAM	AIX	C
Visualization	Pentium 1.7 GHz 512 MB RAM	WINDOWS XP	Visual C^{++} & OpenGL

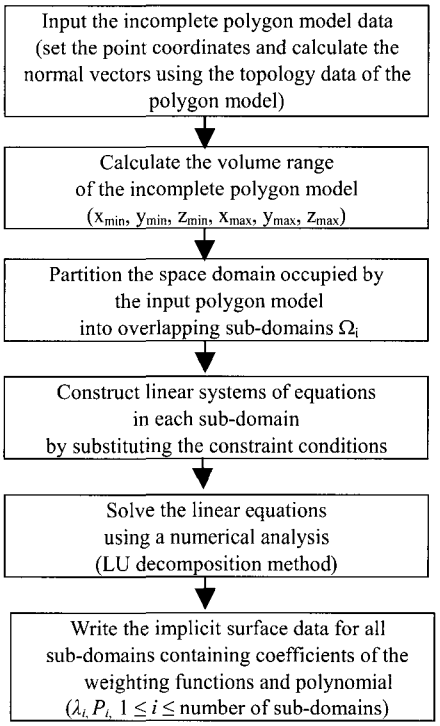


Fig. 4 Flowchart showing the generation of an implicit surface

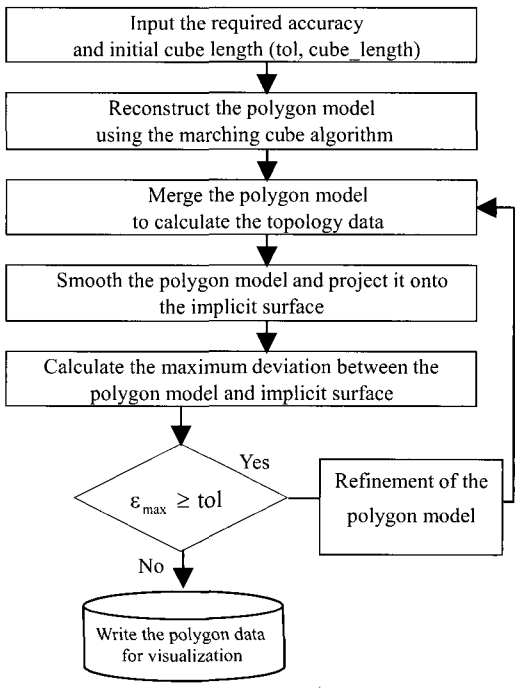
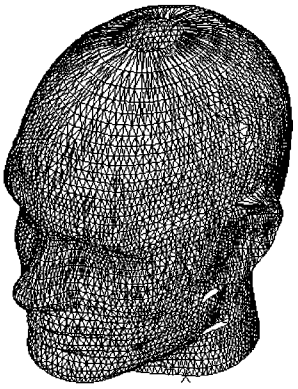
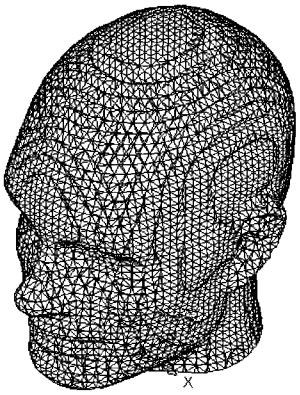


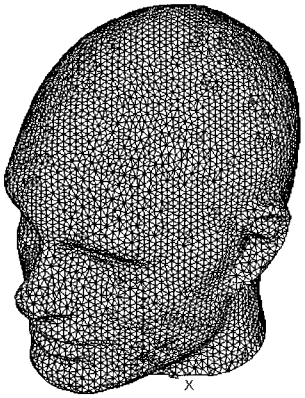
Fig. 5 Flowchart of the overall reconstruction of the polygon model



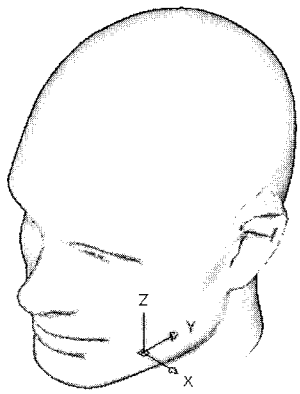
(a) before hole filling



(b) generation of a polygon model using the marching cube algorithm from the generated implicit surface



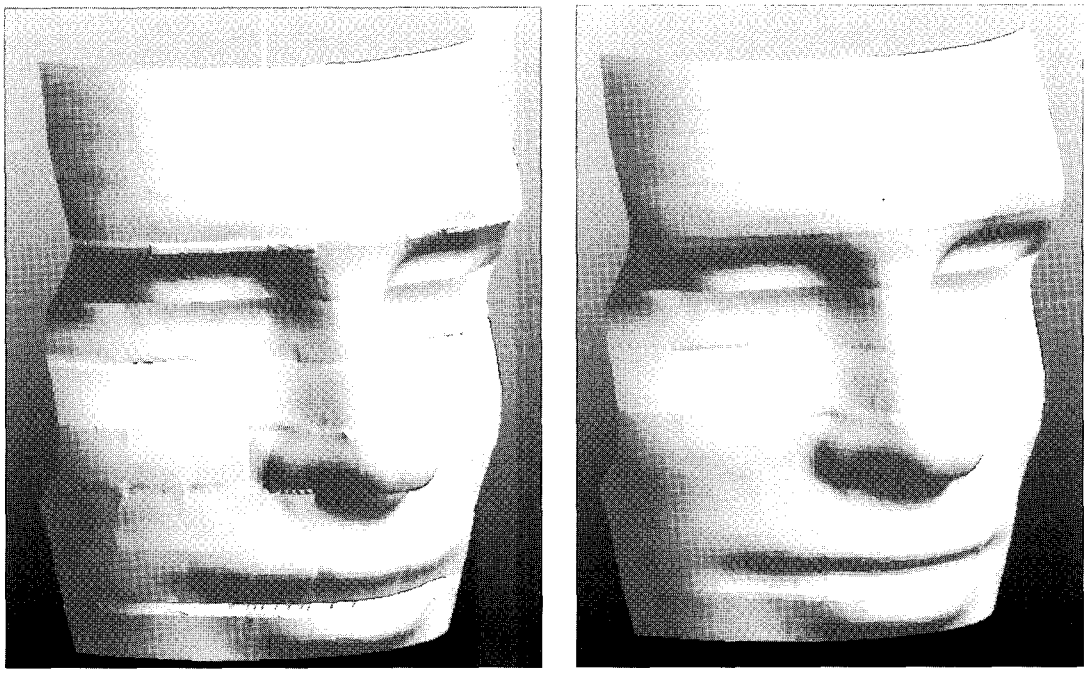
(c) smoothed polygon model from relocating the vertices



(d) view of the polygon model after refining and projecting onto the implicit surface

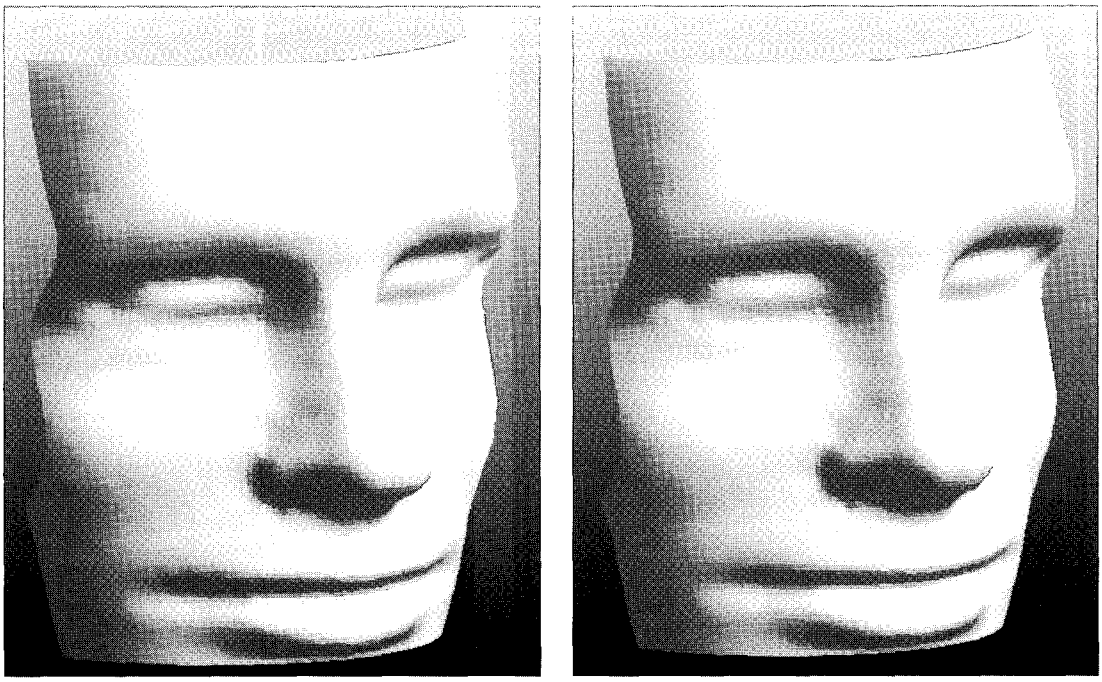
(e) detailed view of the ear

Fig. 6 Filling holes in the human head model



(a) overlap factor : 0%

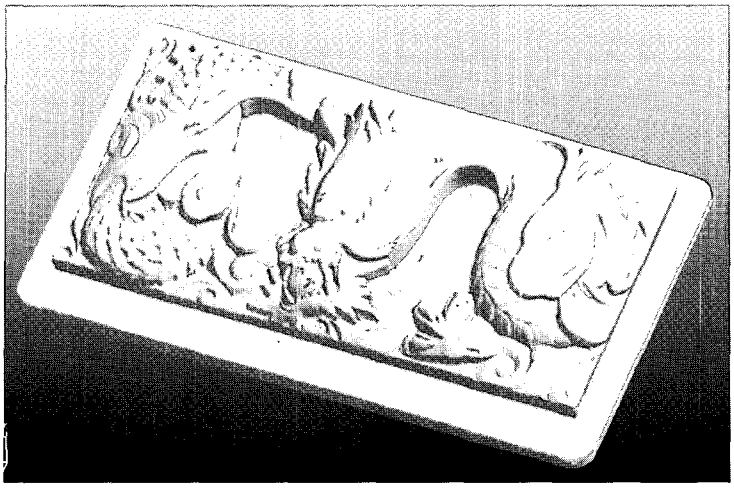
(b) overlap factor : 5%



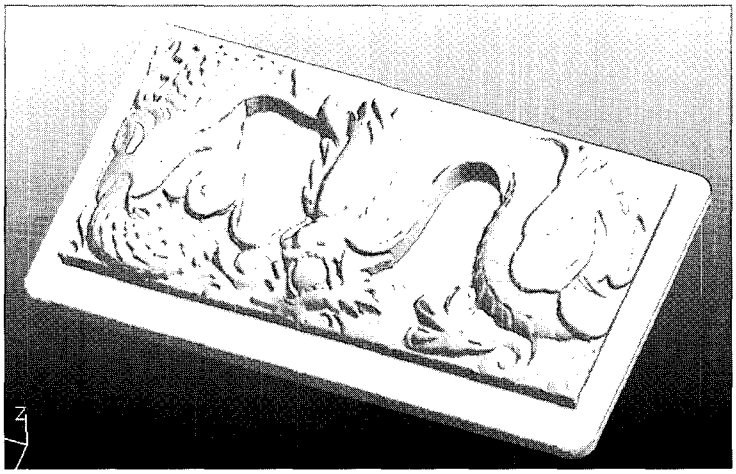
(c) overlap factor : 15%

(d) overlap factor : 25%

Fig. 7 Effect of the size of the overlapping zones

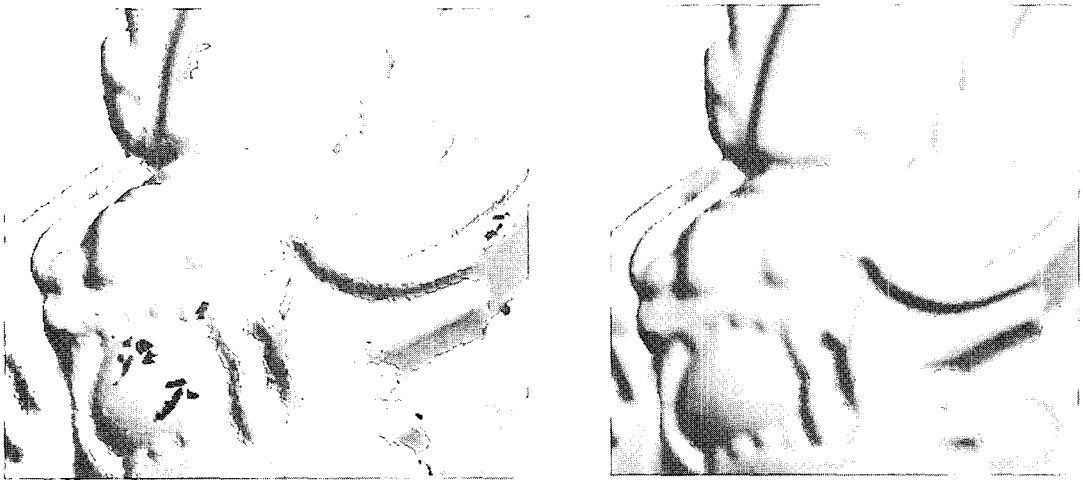


(a) view of the polygon model before hole filling



(b) view of the polygon model after hole filling

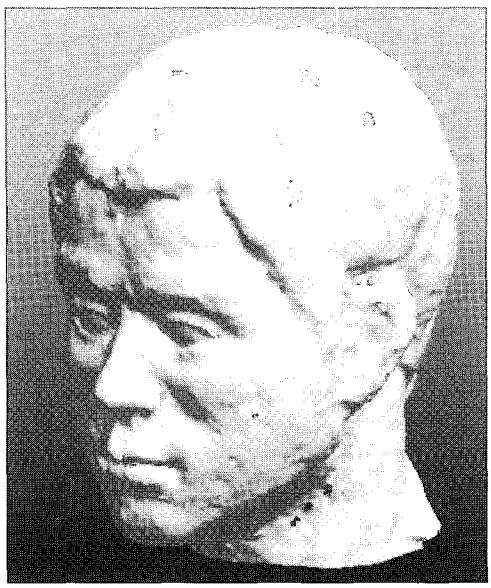
Fig. 8 Filling holes in a dragon model



(a) incomplete polygon model before hole filling

(b) polygon model after hole filling

Fig. 9 Detailed view of the dragon model



(a) before hole filling

(b) after hole filling

Fig. 10 Filling holes in a sculpture model

smoothing, projection, and refinement. Fig. 6 illustrates the application of this technique to a human head model that has many holes. Fig. 6(a) shows a view of the original polygon model before the hole filling. Fig. 6(b) shows a view of the coarse polygon model generated from the reconstructed implicit surface using the marching cube algorithm. Although the many holes that existed in the original polygon model were eliminated, several poorly shaped triangles from a mesh quality viewpoint still remained. As shown in Fig. 6(c), the mesh quality improved remarkably after smoothing the polygon model. Fig. 6(d) illustrates a view of the final polygon model after refining and projecting onto the implicit surface. The proposed smoothing and mesh refinement scheme yielded a satisfactory mesh quality and geometric accuracy. Fig. 6(e) shows a detailed view of one ear and illustrates that the proposed method effectively repaired the mesh by filling the holes and improving the quality. Since the original polygon model was composed of 14,562 triangles and 7,363 nodes, it would be impossible to generate an implicit surface using a conventional implicit surface interpolation method based on radial basis function (RBF). However, the computation time consumed for the interpolation and polygonization was reduced dramatically by introducing the domain decomposition method into the traditional RBF. In this example, the spatial domain occupied by the original polygon model was divided into 633 sub-domains for the calculations; 55 seconds were required to interpolate the implicit surface and 113 seconds were required to generate the polygon model.

From experience, we recommend using between 30 and 200 points in each sub-domain to ensure stable reconstruction of the implicit surface. If the number of allocated points in some sub-domains is less than 30, the local interpolation f_i can lead to unexpected results. If the number of allocated points exceeds 200, the computation time will increase remarkably. Therefore, we implemented an automatic adjusting scheme for the sub-domain size to allocate an optimum number of points to each sub-domain.

Fig. 7 illustrates the effect of the size of the overlapping zone. Fig. 7(a) shows interpolated results for a 0% overlap factor, *i.e.*, with no overlapping zones between sub-domains. As expected, it was not difficult to distinguish the boundary zones from neighboring sub-domains due to discontinuities between the sub-domains. A larger value of the overlap factor yielded a more stable and continuous reconstructed implicit surface, as shown in Fig. 7(b)–(d). Although the optimum overlap factor varies according to the characteristics of the input polygon model, a factor of about 10% was sufficient to obtain stable interpolation results.

Finally, the robustness and effectiveness of the proposed methods were verified using larger polygon models. Fig. 8(a) shows a view of an original model that was composed of 154,246 nodes and 306,835 triangles before the hole filling process. The implicit surface was reconstructed by dividing the 3D spatial domain into 3,398 sub-domains; 8 minutes were required for the interpolation and 16 minutes were required to reconstruct the final polygon model, shown in Fig. 8(b), composed of 340,945 nodes and 680,480 triangles. As shown in Fig. 9, the many holes scattered in the original polygon model were successfully eliminated and the reconstructed polygon model was much smoother compared to the original polygon model. Fig. 10 presents the results of the hole filling procedure for a sculpture model composed of 368,746 nodes and 623,434 triangles. The implicit surface was reconstructed by dividing the space into 4,734 sub-domains; 16 minutes were required for the interpolation and 35 minutes were required for the polygonization of the final model, shown in Fig. 10(b), composed of 437,605 nodes and 985,600 triangles. Once again, the various holes scattered in the original scanned polygon model were successfully eliminated.

5. Conclusions

A novel shape reconstruction method was proposed to fill many types of holes that are often found scattered throughout a polygon model. The proposed method used an implicit surface scheme and the domain decomposition method. A new surface was reconstructed by creating a smooth implicit surface from the incomplete polygon model through which the actual surface would pass. Then a mixed method of mesh refinement and smoothing was used to efficiently visualize the resulting implicit surface. The 3D spatial domain occupied by the input polygon model was divided into several sub-domains. Local solutions were then obtained by interpolating points allocated in each sub-domain separately. Finally, the local solutions were blended together using a smooth blending function, forming a partition of unity to obtain a global solution. The numerical results demonstrated that the proposed method could be used to fill holes in a large polygon model using a reasonable expenditure of computing time and with sufficient accuracy, irrespective of the hole size and complexity of the hole shape.

REFERENCES

1. Floater, M. S. and Iske, A. A., "Multi-step scattered data interpolation using compactly supported radial basis functions," Journal of Computational and Applied Mathematics, Vol. 73, pp. 65-78, 1996.
2. Turk, G. and O'Brien, J. F., "Variational implicit surfaces," Tech. Rep. GIT-GVU-99-15, Georgia Institute of Technology, 1999.
3. Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C. and Evans, T. R., "Reconstruction and representation of 3D objects with radial basis functions," in Proceedings of SIGGRAPH 2001, pp. 67-76, 2001.
4. Lazzaro, D. and Montefusco, L. B., "Radial basis functions for multivariate interpolation of large scattered datasets," Journal of Computational and Applied Mathematics, Vol. 140, pp. 521-536, 2002.
5. Yoo, D. J., "A study on the automatic elimination of free edge for sheet metal forming analysis," Journal of the Korean Society for Technology of Plasticity, Vol. 13, No. 7, pp. 614-622, 2004.
6. Yoo, D. J., "A study on filling holes of the polygon model using implicit surface scheme," Journal of the Korean Society of Precision Engineering, Vol. 22, No. 3, pp. 107-114, 2005.
7. Kojekine, N., Hagiwara, I. and Savchenko, V., "Software tools using CSRBFs for processing scattered data," Computers and Graphics, Vol. 27, pp. 311-319, 2003.
8. Ohtake, Y., Belyaev, A., Alexa, M., Turk, G. and Seidel, H. P., "Multi-level partition of unity implicits," ACM Transactions on Graphics (TOG), Vol. 22, pp. 463-470, 2003.