A Surface Displaced from a Manifold

Seung-Hyun Yoon

Computer Science and Engineering, Seoul National University, Korea

Abstract. We present a new surface representation scheme based on a manifold structure and displacement functions. Given a geometric model represented as a point cloud, we construct a domain manifold which is a smooth surface blended from simple local patches. The original points are then projected on to the local patches and their displacements are adjusted so that the final displaced surface approximates the given point data with high precision. We demonstrate the effectiveness of our approach in various applications, including multi-resolution representations and skeleton-driven shape deformation.

1 Introduction

Thanks to recent advances in scanning technologies, it has become easy to obtain accurate and detailed geometric data from real-world objects. However, contemporary scanning systems produce a discrete representation, in the form of unorganized point clouds or polygonal meshes. Such models can have serious problems, including irregularities, discontinuities, huge size and missing areas, making it difficult to use this sort of data for practical applications.

Many mesh processing techniques have been developed to resolve these problems. Representative methods include simplification, smoothing, re-meshing and mesh optimization [4,9,10]. However, most of these techniques operate entirely in the domain of discrete geometry, and do not offer the analytic representations which are so useful in geometric modeling and processing.

Cook [1] introduced the displaced surface as an ingenious way of creating a detailed geometric model by applying a displacement map to a smooth surface. Krishnamurthy and Levoy [15] used tensor product B-spline patches to represent the underlying domain surface and created fine details with displacement vectors; but this leads to a discontinuity problem across patch and displacement map boundaries. More recently, Lee et al. [16] have proposed a framework to unify subdivision surfaces and scalar displacements. Thanks to the properties of subdivision surfaces, there are no discontinuity problems in this approach. However, subdivision surfaces do not offer analytic representations in general.

In this paper, we introduce a new representation scheme for displaced surfaces, based on the manifold technique proposed by Ying and Zorin [19]. We first construct a simple C^2-continuous smooth surface which interpolates the vertices of a given control mesh. Then, we define a scalar displacement function on each chart of an atlas of this domain. A detailed surface can now be constructed by applying scalar displacement functions to the local patches and blending them using the manifold structure of a control mesh.

M.-S. Kim and K. Shimada (Eds.): GMP 2006, LNCS 4077, pp. 677–686, 2006.

In order to be able to address practical applications with our technique, we also propose an algorithm to approximate a detailed geometric model given as a point cloud. This approximation is based on the analytic properties of our representation scheme and the optimization of scalar displacement functions on the local charts of the domain atlas. We demonstrate the effectiveness of our approach by presenting experimental results relating to applications in multi-resolution representations and skeleton-driven shape deformation.

The main contributions of this paper are summarized as follows:

- **Surface representation:** We propose an analytic surface representation scheme which combines a manifold technique with a displacement function.
- **Shape approximation:** We present an algorithm that approximates a detailed geometric model from an unorganized point cloud. Instead of using ray-shooting, we employ a more precise method of point projection and then minimize the approximation error.
- **Applications to significant geometric problems:** The proposed technique can be applied to a variety of applications including multi-resolution representations and skeleton-driven deformation.

The rest of this paper is organized as follows. In Section 2, we review some of the related recent work. In Section 3, the construction of our displaced surface is detailed. We go on to propose an algorithm that can approximate a detailed geometric model in Section 4. Experimental results are presented in Section 5 and we conclude this paper in Section 6.

2 Related Work

We begin with a brief review of some representative approximation schemes that use displaced surfaces and also of techniques for surface modeling that are based on the manifold structure of a model.

Approximation with displaced surfaces. Krishnamurthy and Levoy [15] put forward a technique for approximating an arbitrary mesh using B-spline patches and displacement map, using vector displacements to approximate a target mesh. This technique produces a good surface fit, but there is a discontinuity problem across patch boundaries.

Lee et al. [16] proposed the displaced subdivision surface. They constructed a subdivision surface and a map of scalar displacements along its normals. The resulting detailed surface is represented using a unified subdivision framework. However, this approach does not provide an analytic representation of the displaced surface.

Jeong and Kim [13] proposed a technique for constructing a displaced subdivision surface from an unorganized point cloud. However, their method only works for objects of genus 0. Recently, Kim and Kim [14] proposed a surface reconstruction technique that works for objects of arbitrary topology. They used a butterfly subdivision surface and a moving least-squares approach to approximate an unorganized point cloud.

Manifold surfaces. Grim and Hughes [7] used a manifold concept in constructing surfaces of arbitrary topology. This technique has also been applied to the parametrization of a surface [5] and to fit a surface to point cloud data [6]. Cotrina and Pla [2] proposed a C^k surface construction algorithm which uses a regular star-shaped configuration to represent an irregular vertex and a polynomial transition function with overlapping charts. The resulting surface is of B-spline form at a boundary and may be seen as a generalized B-spline surface. A more generic approach to freeform surface generation was proposed by Cotrina et al. [3]. They can construct three different types of surfaces, but their techniques require complicated transition functions.

A simpler surface construction technique was proposed by Ying and Zorin [19]. They constructed charts in the complex plane with transition functions that have a simple analytic form. Their method provides C^∞ continuity and local support; moreover, the resulting surfaces are visually satisfactory. Recently, a new manifold spline has been proposed by Gu et al. [8], which provides a general theoretical and computational framework for manifold splines defined on a control mesh with arbitrary topology. They have also provided a practical algorithm for the construction of a triangular B-spline surface on a control mesh.

3 Displaced Surface Construction

We employ the simple manifold structure of Ying and Zorin [19]. A simple bi-quadratic local patch is constructed on each chart and a scalar displacement function is added to it. The detailed displaced surface is then constructed by blending the local patches and the scalar displacement functions in a unified way.

A local patch. Let $\mathbf{v}_i$ be the vertex of a control mesh (Figure 1(a)) and let U_i be the corresponding chart. We construct a local patch $P_i(s,t)$ on each chart U_i using the following bi-quadratic polynomial representation, which is constructed by approximating the positions of the neighboring vertices of $\mathbf{v}_i$ in the control mesh:

$$P_i(s,t) = \begin{bmatrix} s^2 & s & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{c}^i_{1,1} & \boldsymbol{c}^i_{1,2} & \boldsymbol{c}^i_{1,3} \\ \boldsymbol{c}^i_{2,1} & \boldsymbol{c}^i_{2,2} & \boldsymbol{c}^i_{2,3} \\ \boldsymbol{c}^i_{3,1} & \boldsymbol{c}^i_{3,2} & \boldsymbol{c}^i_{3,3} \end{bmatrix} \begin{bmatrix} t^2 \\ t \\ 1 \end{bmatrix}, \tag{1}$$

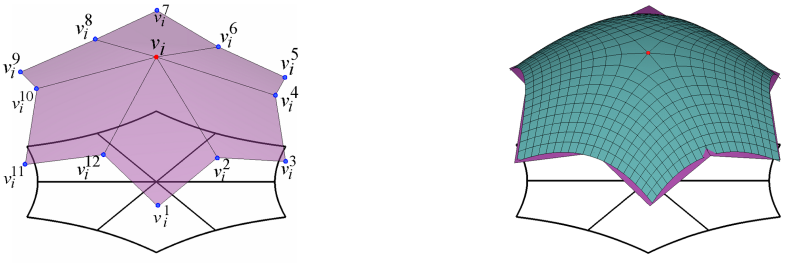
where $\boldsymbol{c}^i_{j,k} = \begin{bmatrix} x^i_{j,k} & y^i_{j,k} & z^i_{j,k} \end{bmatrix}^T$.

The coefficient vectors $\boldsymbol{c}^i_{j,k}$ of Equation (1) can be determined by minimizing the following functional:

$$L(\boldsymbol{c}^i_{1,1}, \cdots, \boldsymbol{c}^i_{3,3}) = \sum_{k=1}^{2d} \|\mathbf{v}^k_i - P_i(s_k, t_k)\|^2, \tag{2}$$

subject to $P_i(0,0) = \mathbf{v}_i$, where d is the valency of a vertex $\mathbf{v}_i$, and $\mathbf{v}^k_i$ is the k^{th} neighboring vertex of $\mathbf{v}_i$. The constraint represented by this equation forces

the coefficient $\boldsymbol{c}^i_{3,3}$ to be $\mathbf{v}_i$ in order to interpolate the position of the vertex $\mathbf{v}_i$, which is assigned to the origin of each chart. Figure 1(a) shows the vertices of a control mesh around the vertex $\mathbf{v}_i$ and Figure 1(b) shows the corresponding local patch, which interpolates the position of the center vertex $\mathbf{v}_i$.



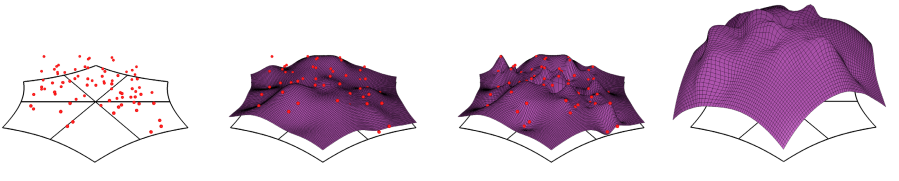
(a) (b)

Fig. 1. A local patch: (a) the vertices of a control mesh, (b) the resulting local patch

Displacement function. Once local patches have been constructed for all the charts, a scalar displacement function $d_i(s,t)$ is created on each chart U_i by interpolating the signed distances from each local patch to the point cloud data. Since these are not regular data, we used a scattered-data interpolation technique based on multi-level B-splines [17]. Figure 2(a) shows random displacements on a chart U_i, and Figures 2(b) and (c) show the corresponding displacement functions $d_i(s,t)$ at different levels of detail. Figure 2(d) shows the displaced local patch $S_i(s,t)$ generated by applying a displacement function $d_i(s,t)$ to a local patch $P_i(s,t)$. The position of a displaced point is computed as follows:

$$S_i(s,t) = P_i(s,t) + d_i(s,t) \cdot N_i(s,t), \tag{3}$$

where $N_i(s,t)$ is the unit normal vector of a local patch $P_i(s,t)$.



(a) (b) (c) (d)

Fig. 2. Displacement functions: (a) randomly generated displacements, (b) a displacement function with a low level of detail, (c) a displacement function with a high level of detail, (d) the displaced local patch $S_i(s,t)$

Displaced Surface Construction. We have now constructed a local patch and a scalar displacement function on each chart. To construct a blended surface, we

use the same transition function and blending function as Ying and Zorin [19]. Since the blending function has unit weight at its origin and zero weight at its boundary, the interpolation property of the local patches are preserved in the final blended domain surface.

The final displaced surface is achieved by applying the scalar displacement functions to the local patches. For a given parameter value z in a chart U_i, the displaced surface point $D_i(z)$ can be evaluated from Equation (4). First we need to find all the overlapping charts $\{U_j\}$ in the atlas which contains z, and then to compute the corresponding transition parameters on each chart U_j, using the transition functions $\theta_{i,j}(z)$. Then we can determine the displaced points $S_j(z)$ from Equation (3). Finally, all the displaced points are blended together smoothly using the weight functions $w_j(\theta_{i,j}(z))$ on each chart:

$$D_i(z) = \sum_{j \in J_z} w_j(\theta_{i,j}(z)) \cdot S_j((\theta_{i,j}(z)), \tag{4}$$

where z is the value of a parameter in a local chart U_i, $J_z = \{j | z \in U_j\}$ is an index set, and $w_j(\cdot)$ are the smooth blending functions that form a partition of unity.

4 Approximation to a Point Cloud

In this section, we present a new algorithm that approximates a detailed geometric model with our displaced surface. The approximation proceeds in three steps. First we generate a control mesh and the corresponding domain surface from a point cloud. Each data point is then projected on to nearby local patches and initial projection parameters and corresponding displacements are computed. Finally, they are adjusted to minimize the approximation errors and optimal displacement functions are constructed. Our contribution consists of the second and third steps, while the first step is performed using existing methods.

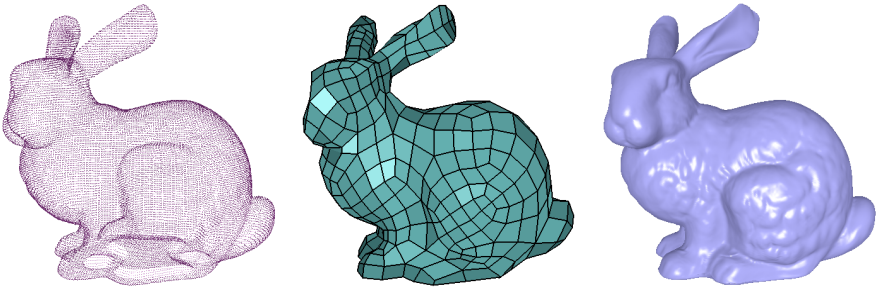
Control mesh. Many advanced techniques [9,13,14,16] exist for the construction of a good control mesh and domain surface from a detailed geometric model. We use a commercial software package [12] which is based on a combination of existing methods. Figures 3(a) and 4(a) show typical point cloud data and Figures 3(b) and 4(b) show the resulting control meshes.

Initial displacements. Once the control mesh has been constructed, a smooth domain surface is generated by blending the local patches. Scalar displacement functions are then constructed on each chart. To construct optimal scalar displacement functions, we compute initial displacements on each chart and adjust them to minimize the approximation errors. The initial displacements are computed in a different way from previous methods [13,14,16], which shoot rays from the domain surface to the detailed geometry. We project each point $\mathbf{p}_i$ on to a nearby local patch, which is considerably easier to implement and produces a more precise result than shooting methods. Assuming that we are using a quadrilateral control mesh, $\mathbf{p}_i$ will be projected on to four local patches

P_j, $j = 0, 1, 2, 3$. The use of a bi-quadratic surface representation for the local patches greatly simplifies projection.

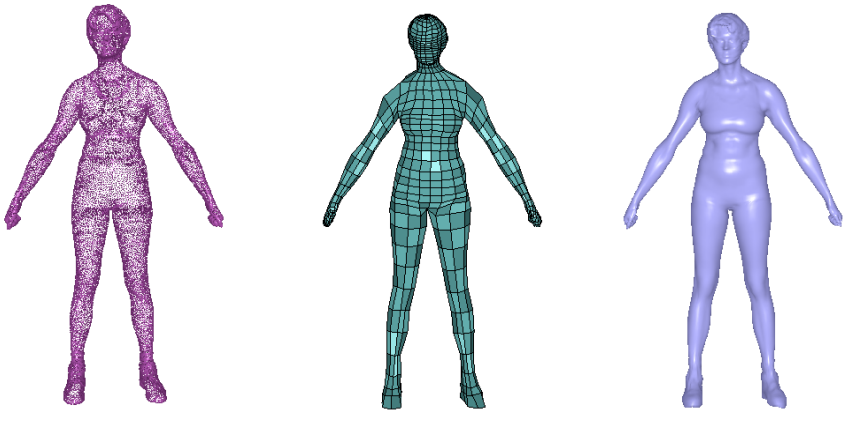
Let $\hat{\mathbf{p}}_i$ denote the projection of $\mathbf{p}_i$ on to the closest face. Then we find the vertex $\mathbf{v}_0$ of that face which is nearest to the projection point $\hat{\mathbf{p}}_i$. The local patch corresponding to $\mathbf{v}_0$ will be denoted as $P_0(s, t)$. An initial guess of the parameters of the projected point, (s_0^0, t_0^0), is made from the coordinates of $\hat{\mathbf{p}}_i$ projected on to the corresponding face of the control mesh. From the initial solution (s_0^0, t_0^0), we compute more precise parameters $(\hat{s}_0, \hat{t}_0)$ using a Newton iteration.

Once the projection parameter $(\hat{s}_0, \hat{t}_0)$ on the first patch has been obtained, the other three projection parameters $(\hat{s}_j, \hat{t}_j)$, $j = 1, 2, 3$ are guessed using the transition functions $\theta_{i,j}$. More precise solutions can be computed in a few additional steps of a Newton iteration. For robustness, we ignore any projection parameters that end up outside their chart.



(a) (b) (c)

Fig. 3. Approximation process: (a) point cloud, (b) control mesh, (c) result



(a) (b) (c)

Fig. 4. Approximation process: (a) point cloud, (b) control mesh, (c) result

The displacements $\hat{d}_j, j = 0, 1, 2, 3$ are then computed as signed distances from $P_j(\hat{s}_j, \hat{t}_j)$ to $\mathbf{p}_i$. These become the initial scalar displacements that correspond to the projection parameters $(\hat{s}_j, \hat{t}_j)$, $j = 0, 1, 2, 3$ on each chart. Using existing techniques [13,14,16] it is difficult to find the projected points since the surface does not have an analytic form; which is why previous authors have used ray-shooting. Our analytic representation greatly simplifies the projection procedure and produces considerably more precise results, by the use of an optimization technique to be discussed below.

Optimal displacement function. In general, the four projection parameters $(\hat{s}_j, \hat{t}_j)$, $j = 0, 1, 2, 3$ on each chart do not satisfy the transition relations. Each initial scalar displacement only represents the exact displacement of the point data from one local patch. When the four displacements are blended together, the resulting surface may not interpolate the original point data exactly. This approximation error is due to a mismatch in the transition relations among the projection parameters $(\hat{s}_j, \hat{t}_j)$, $j = 0, 1, 2, 3$. To minimize the approximation error, we need to compute optimal parameters (s_j, t_j) which satisfy the transition relations and their corresponding displacements d_j, $j = 0, 1, 2, 3$. These optimal parameters and displacements are obtained by minimizing the following error functional:

$$\Psi(s_0, t_0, d_0, d_1, d_2, d_3) = \|\mathbf{p}_i - \sum_{j=0}^{3} w_j(s_j, t_j) \cdot (S_j(s_j, t_j) + N_j(s_j, t_j) \cdot d_j)\|^2, \quad (5)$$

where the parameters (s_0, t_0) are initialized to the projection parameter $(\hat{s}_0, \hat{t}_0)$, and $(s_j, t_j) = \theta_{i,j}(s_0, t_0)$ are its transition parameters. The displacements d_j are initialized to $\hat{d}_j$, $j = 0, 1, 2, 3$. From the initial projection parameters and displacements, we can then find the optimal ones which minimize Equation (5). This is a general non-linear optimization problem in 6-dimensional space, which is solved by a direction-set method [18]. Although the optimal parameters may not be the exact projection parameters on to local patches, they represent the least-square minimization of the approximation error.

The optimal parameters and displacements are assigned to each chart, and optimal scalar displacement functions are then constructed by interpolating the assigned displacements.

5 Experimental Results

We implemented our displaced surface algorithm in C++ on a Pentium-IV (3.2GHz) desktop PC with a 2GB main memory. Figures 3 and 4 illustrate the approximation process and its results on two different models. The processing time can be divided into two parts: at the vertex projection stage all the vertices are projected on to nearby local patches, which involves the calculation of optimal projection parameters and displacements; and at the second stage optimized displacement functions are constructed on each chart. Table 1 itemizes processing times for each stage, together with the number of points in the point cloud data and the control meshes.

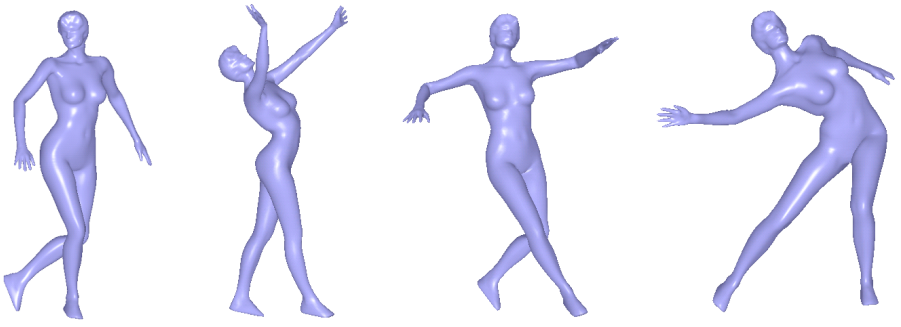


Fig. 5. Skeleton-driven deformation results

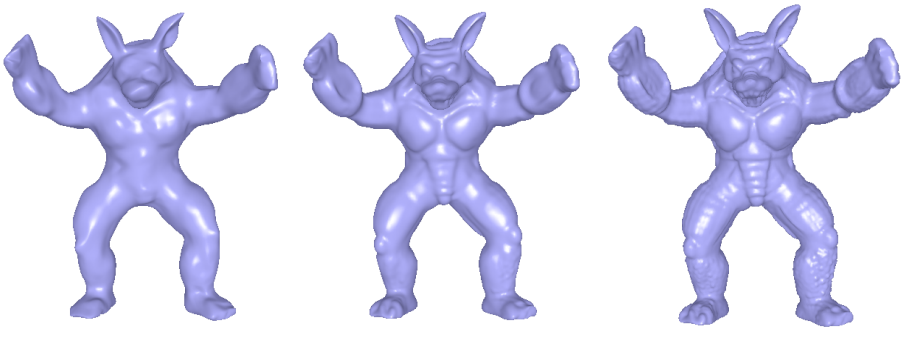


Fig. 6. Multi-resolution representations

Table 1. Approximation results

Models	Bunny	Human	Armadillo
Number of data points	34,834	43,817	159,103
Vertices on control mesh	540	1,134	720
Time for projection and optimization (sec)	221.07	318.72	813.72
Time for displacement functions (sec)	1.43	1.95	4.21
Approximation error without optimization	$1.78 * 10^{-3}$	$6.86 * 10^{-4}$	$7.84 * 10^{-3}$
Approximation error with optimization	$1.52 * 10^{-4}$	$1.23 * 10^{-4}$	$2.78 * 10^{-4}$

To demonstrate the effectiveness of our approach in practical applications, we applied the sweep-based deformation technique of Hyun et al. [11] to the approximation of a human model. Figure 5 shows clips from an animation made using this technique. It demonstrates that our displaced surface representation can produce continuous and natural deformations of a human model.

Our technique can also be used to construct multi-resolution representations. Our scalar displacement functions are in the form of a multi-level B-spline representation, which allows us to change their level of detail at runtime. Figure 6 shows an Armadillo model represented by varying levels of the displacement

functions. We analyzed the approximation errors (average distances) for each model (normalized to unit cube size), and they are listed in the last two rows of Table 1.

6 Conclusion

We have introduced a new displaced surface representation based on a manifold structure. Our representation consists of simple local patches and scalar displacement functions on each chart of a control mesh. These two geometries are smoothly blended to represent a detailed surface in a unified way. Moreover, our domain surface has a convenient interpolation property for the vertices of the control mesh and our scalar displacement function supports multi-level representations.

We have also presented an algorithm for approximating a detailed geometric model given as a point cloud. This algorithm is different from existing techniques in that our approach is based on vertex projection and the optimization of displacement functions. The vertex projection technique eliminates the need for connectivity information in the model, and an optimization procedure produces precise results. We present experimental results which demonstrate the effectiveness of our approach in applications such as multi-resolution modeling and skeleton-driven deformation.

Acknowledgements. The authors would like to thank the anonymous reviewers for their invaluable comments. This work was supported by the Korean Ministry of Information and Communication (MIC) under the Program of IT Research Center on CGVR.

References

1. R. Cook. Shade trees. *Proceedings of ACM SIGGRAPH*, 223–231, 1984.
2. J. Cotrina and N. Pla. Modeling surfaces from planar irregular meshes. *Computer Aided Geometric Design 17*, 1, 1–15, 2000.
3. J. Cotrina, N. Pla and M. Vingo. A generic approach to free form surface generation. *Proceedings of the ACM Symposium on Solid Modeling and Applications*, 35–44, 2002.
4. M. Eck, T. DeRose, H. Duchamp, M. Lounsbery and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Proceedings of ACM SIGGRAPH*, 303–312, 1995.
5. C. Grim. Simple manifolds for surface modeling and parameterization. *Proceedings of Shape Modeling International*, 237, 2002.
6. C. Grim, J. Crisco and D. Laidlaw. Fitting manifold surfaces to 3D point clouds. *Journal of Biomechanical Engineering 124*, 1, 136–140, 2002.
7. C. Grim and J. Hughes. Modeling surfaces of arbitrary topology using manifolds. *Proceedings of ACM SIGGRAPH*, 359–368, 1995.
8. X. Gu, Y. He and H. Qin. Manifold spline. *Proceedings of the ACM Symposium and Solid and Physical Modeling*, 27–38, 2005.

9. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle. Surface reconstruction from unorganized points. *Proceedings of ACM SIGGRAPH*, 71–78, 1992.
10. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle. Mesh optimization. *Proceedings of ACM SIGGRAPH*, 19–26, 1993.
11. D.-E. Hyun, S.-H. Yoon, J.-W. Chang, J.-K. Seong, M.-S. Kim and B. Jüttler. Sweep-based human deformation. *The Visual Computer 21*, 8, 542–550, 2005.
12. INUS TECHNOLOGY. *RapidForm Users Manual*, 2006.
13. W.-K. Jeong and C.-H. Kim. Direct reconstruction of displaced subdivision surface from unorganized points. *Graphical Models 64*, 2, 78–93, 2002.
14. S.-J. Kim and C.-H. Kim. Point cloud approximation by displaced butterfly subdivision surfaces. *Proceedings of Israel-Korea Bi-National Conference*, 5–10, 2005.
15. V. Krishnamurthy and M. Levoy. Fitting smooth surface to dense polygon meshes. *Proceedings of ACM SIGGRAPH*, 313–324, 1996.
16. A. Lee, H. Moreton and H. Hoppe. Displaced subdivision surfaces. *Proceedings of ACM SIGGRAPH*, 85–94, 2000.
17. S.-Y. Lee, G. Wolberg and S.-Y. Shin. Scattered data interpolation with multilevel B-splines. *IEEE Transactions on Visualization and Computer Graphics 3*, 3, 228–244, 1997.
18. W.-H. Press, S.-A. Teukolsky, W.-T. Vetterling and B.-P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.
19. L. Ying and D. Zorin A simple manifold-based construction of surfaces of arbitrary smoothness. *Proceedings of ACM SIGGRAPH*, 271–275, 2004.