JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 23, 757-771 (2007)

Free-Form Deformation for Point-Sampled Surface*

PING-HSIEN LIN, TONG-YEE LEE+ AND CHENG-FON LIN+
Department of Computer Science and Information Engineering
National Changhua University of Education
Changhua, 500 Taiwan
+*Computer Graphics Group/Visual System Laboratory*
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, 701 Taiwan
E-mail: tonylee@mail.ncku.edu.tw

In this paper, we present a free-form deformation (FFD) approach to manipulate point-sampled surface. Our approach provides users the powerful free-form deformation directly on point-sampled surface. In the preprocessing, we use a robust extension of hierarchical clustering scheme to partition a point-sampled model into clusters, fit a moving least square (MLS) surface to each of them, and define a valid polygonal area on the base domain of the MLS surface. At run time, we apply the conventional free-form deformation technique on these clusters. While users adjust the grids of the FFD lattice, we propose a novel interpolation method, which responses to the curvature variation during object deformation, to curve the base domain of the MLS surfaces of those clusters concerned. Finally, we resample the new base domain to produce the deformed point set model. The proposed technique is very intuitive, effective and easy to implement. With this technique, high frequency details from the original point-set surface can be maintained well on the deformed surface and several interesting deformed results of point-sampled models are demonstrated to verify the proposed scheme.

Keywords: point-sampled models, clustering, moving least square (MLS) surface, free-form deformation (FFD), interpolation

1. INTRODUCTION

1.1 Motivation

Since the pioneering work [1] was proposed by Levoy and Whitted, the point primitives have been widely investigated in the computer graphics society. In the beginning, the point-based research had been focused on the issues about fast and high quality rendering. Recently, as the rendering techniques have become full-grown, other 3D modeling operations on point set models, such as shape modeling (smoothing, Boolean operations, editing, deformations), multi-resolution analysis, re-sampling, simplification, parameterization, morphing, and so on, have progressed rapidly.

Received April 1, 2005; revised July 4, 2005; accepted August 3, 2005.
Communicated by Pau-Choo Chung.
* This project was supported by the National Science Council of Taiwan, R.O.C., under contracts No. NSC-93-2213-E-006-026, NSC-94-2213-E-006-063, NSC-94-2213-E-006-005, and NSC-94-2213-E-218-016. The bunny, David, Lion models are courtesy of Stanford University and the Octopus and Igea models are courtesy of ETH Zurich.

Fig. 1. We swing the head of a point-sampled bunny model with our FFD method.

The point-based approach usually serves as a competitor with the mesh-based approach. An essential factor causes their operations so different is that the point primitives have no local connectivity information. This feature benefits the point-based approaches for rendering due to the straightforward hierarchical structure, as well as good for local updating, *i.e.*, shape modeling, without rebuilding the local connectivity. Additionally, the point primitives can be acquired easily and naturally, such as using the range scanners or the computer vision techniques to scan and reconstruct the real worlds without the need for triangulation. However, this feature also harms the point-based approach due to high potential risk of changing the local surface topology. For example, unpleasant holes can potentially occur as we zoom in the surface or modify the surface without careful processing. The success of many point-based methods relies on the local surface smoothness requirement. Moreover, since the point primitives have no local connectivity information, we need a huge number of samples to model the continuous surfaces, especially at the geometric features with high spatial frequency and the sharp corners. In general, the number of points of a point set model is ranging from hundred thousands to millions, or even more. This slows down many operations on point primitives. Therefore, an important challenge to point-based shape modeling is how to handle the huge number of points efficiently and interactively.

Free-form shape deformations have been studied extensively in the past [2-4]. In this paper we propose a free-form deformation technique based on the moving least square (MLS) surface [6] and our novel interpolation scheme, to manipulate point set models. While the users adjust the grids of the free-form deformation (FFD) [3] lattice, we use a novel interpolation method, which is based on a circular arc, to account for the local curvature variation during object deformation. The deformed point set model is then generated by re-sampling the curved base domain and the MLS surfaces. Our interpolation method is simple and easy to be implemented, but it can generate a good surface deformation effect. Additionally, it can generate very flexible deformation effect controlled by users. In comparison with two pioneering techniques [16, 19], our approach has some advantages. For example, in [16], this approach requires the execution of a smoothing operator several times on the original models to obtain a smooth based model. After this stage, it can easily compute displacement information between the original model and a smooth based model. Then, after applying a deformation operator to the smooth based model, this approach adds displacement, *i.e.*, details, to obtain a deformed model. Note that the number of points for the smooth based model and original model is almost equal in [16]. On the other hand, our approach does not need to execute a global smooth operation. Furthermore, we only apply deformation operation to some of original points instead of all original points in [16]. Therefore, our approach potentially performs efficiently than [16]. Muller's work [19] can efficiently simulate several physical effects like elastic, plastic and melting. However, this excellent technique is limited in physical-

based domain. In contrast, our technique is more general and can create more flexible free-form deformation (FFD) in an intuitive way. Furthermore, our approach can be easily integrated with other different deformation technique like radial basis functions or other popular FFD techniques [2, 4].

1.2 Related Work

After Levoy and Whitted's work [1] introduce the points as the rendering primitives, many researchers have been involved in the area of rendering and modeling point-sampled geometry. Rusinkiewicz *et al*. [7] and Pfister *et al*. [8] first develop the good rendering techniques for the point primitives. Then the splatting kernels are improved by Zwicker *et al*. [9] and the point rendering is accelerated by Ren *et al*. [10]. Kalaiah *et al*. [11] further improve the rendering quality using the normal mapped rectangle that mimics the local differential geometry properties.

After the point rendering techniques become matured, the shape modeling techniques for point primitives have mushroomed. Pauly *et al*. [12] present a framework of spectral methods for processing point-sampled objects. This framework can perform several operations on point models, including noise removal, enhancement, restoration, and sub-sampling. Zwicher *et al*. [13] develop an interactive system for point-based surface editing called Pointshop3d, which supports a variety of different interaction techniques to alter shape and appearance of 3D point models, including cleaning, texturing, sculpting, carving, filtering, and re-sampling. These two techniques [12, 13] only perform the shape modeling operations based on model's local normal displacements, and thus they cannot generate large scale surface modeling effects. Alexa *et al*. [6] use the moving least square (MLS) projection framework to smoothly approximate a point set surface locally and introduce techniques for re-sampling the point set surfaces. Pauly *et al*. [5, 14] propose the simplification and multi-resolution framework for point-sampled surfaces. These techniques can refine or smooth out the point set surfaces with different scale levels, but they cannot perform more flexible user controlled shape modeling operations on point objects.

Adams *et al*. [15] propose an efficient technique to test the inside-outside conditions for performing Boolean operations on point models. Pauly *et al*. [16] use a hybrid geometry representation, which possesses the advantages of implicit and parametric surface models, to perform a large constrained deformations as well as Boolean operations on point-sampled surfaces. Guo *et al*. [17] develop a system for haptics-based editing on point set surfaces. They use the dynamic implicit volumetric model [18] and the mass-spring system to perform the dynamic physics-based modeling on point-sampled objects. Recently, Muller *et al*. [19] present a physical-based method, which is derived using Finite Element Method (FEM), for modeling and animating a wide spectrum of point-based volumetric objects with material properties ranging from stiff elastic to highly plastic.

1.3 Overview

Our method contains two main stages: the preprocessing stage and the deformation stage. In the preprocessing stage, we divide the point set models into clusters and fit a MLS surface to each cluster. After the MLS surfaces of all clusters have been constructed,

we define a valid polygonal area for each cluster and discard all the points of the original point set model. In the deformation stage, we use FFD [3] to deform the cluster positions and normals. Fig. 2 shows the process of the FFD deformation in our system. By adjusting the regular grid vertices, users can control the shape deformation of an object. In Fig. 2, after several manipulations, the original bunny (Fig. 2 (a)) could be deformed into Fig. 2 (e). Other FFD-like methods such as [2, 4] or other deformation schemes such as radial-based function can be easily integrated into our approach, too. According to the deformed information, we re-sample the base domain and interpolate a curved base domain of MLS surface for each cluster, and then use the MLS surface to regenerate the deformed point set models. These processes are illustrated in Fig. 3.

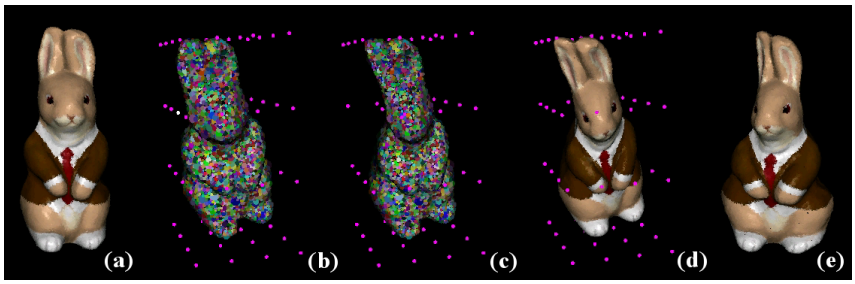


Fig. 2. The process of the free-form deformation on a point set model.

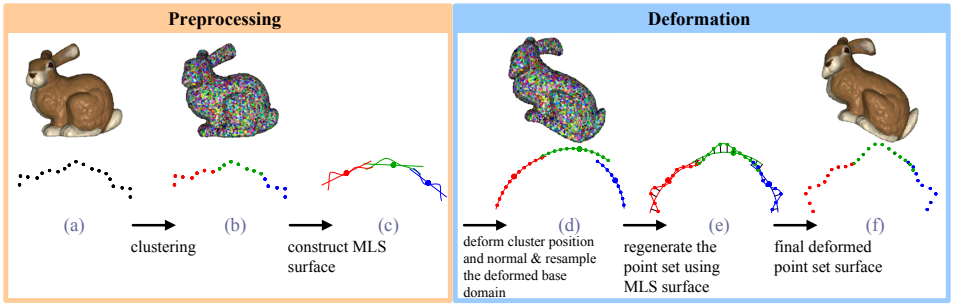


Fig. 3. The procedures of our free-form deformation technique for point-sampled surface.

The remainder of this paper is organized as follows: In sections 2 and 3, we introduce the preprocessing stage and deformation stage of the proposed techniques, respectively. In section 4, experimental results obtained from the proposed method are demonstrated. Finally, we give the conclusion and future work in section 5.

2. PREPROCESSING

Point-sampled models are usually composed of huge number of points. To make an operation on such models efficiently and interactively, we attempt to downscale the number of primitives to be handled. In our preprocessing stage, we partition the point set

surface into clusters. The number of clusters is usually less than a tenth of the number of points. Then we fit a MLS surface to each cluster and define its boundary on the base domain of the MLS surface.

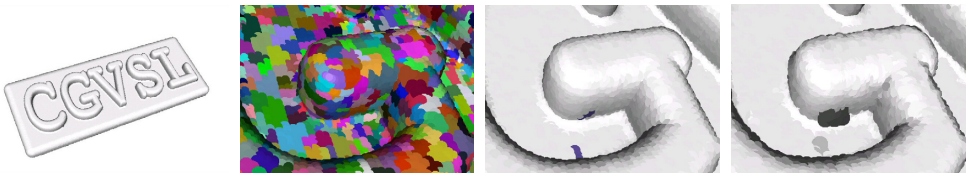
2.1 Clustering

We use a hierarchical clustering scheme [5] to partition a point set surface into clusters. In Eq. (1), the surface variation σ of a point set surface P is defined by the ratio of the minimum eigenvalue with the sum of the three eigenvalues of the P's covariance matrix.

$$\sigma(P) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{1}$$

A point set is split if the following condition holds:

- The size of a point set surface P is larger than the user specified maximum cluster size $n_{\max}$ or
- The variation $\sigma(P)$ is above a maximum threshold called $\sigma_{\max}$.



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

Fig. 4. Wrong clustering example. (a) Original point-based model; (b) A close view to a letter 'G' after the hierarchical clustering and each cluster is colored using a distinct color; (c) The same view as (b), but a cluster consisting of unconnected parts is colored, *i.e.*, blue; (d) Wrong reconstructed MLS surface caused by the unconnected parts.

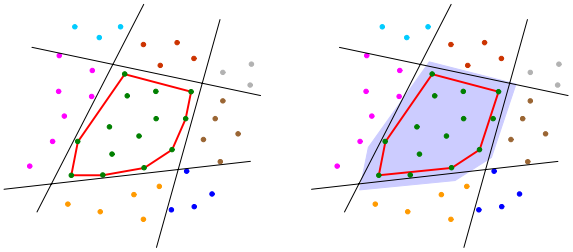
The splitting plane is determined by the centroid of P and the largest eigenvector of the covariance matrix. However, in some cases, if $n_{\max}$ and $\sigma_{\max}$ are not selected properly, this method could produce wrong clusters consisting of unconnected parts as shown in Fig. 4. This phenomenon causes the constructed MLS surface not to fit the original surface properly. Therefore, to make the hierarchical clustering more robust, we remedy it by the following two procedures: (1) cluster checking and (2) cluster merging. The cluster checking is simply done by a region growing process. If the region growing process can not travel all the points of a cluster, it means that these points are not connected inside this cluster. Then we randomly select a point that was not visited by the previous region growing in the cluster and thereafter we restart the region growing process again to form a new sub-cluster. After repeating the above processes several times, we can determine how many unconnected parts, *i.e.*, new sub-clusters, inside the original cluster and the size of each sub-cluster. If the size of a sub-cluster is less than a user specified minimum cluster size $n_{\min}$, this sub-cluster will be merged to its nearest cluster. Other-

wise, this sub-cluster will be classified to be a new cluster. Finally, for each cluster C_i, we record its centroid c_i, referred as the *cluster position* in the remaining of this paper, and an enclosing sphere with a radius r_i.

2.2 Cluster Boundary in the Base Domain of a MLS Surface

Now, let us assume the model is composed of $|C_i|$ clusters. Each cluster C_i is represented by a local MLS surface [6]. For more details to implement a MLS surface, please see [6]. Each C_i has the following information:

- cluster position: c_i
- a local reference domain $H_i = \{x \mid \langle n_i, x\rangle - D_i = 0, x \in R^3\}$, $n_i \in R^3$, $\|n_i\| = 1$, n_i is called cluster normal, and D_i is a constant.
- a bivariate polynomial g_i for the MLS surface of each cluster C_i. The base domain H_i of this MLS surface passes through c_i.
- cluster boundary (a series of lines on base domain of MLS surface).
- an enclosing sphere with a radius cr_i.



(a) (b)

Fig. 5. The boundary of a cluster. The black lines are the cutting planes during the hierarchical clustering. In (b), the region is inflated by an offset as shown in a blue color.

Because we use the MLS surface to represent the surface of a cluster, we need to define the boundary on H_i, that is valid for this local MLS surface. We use a convex hull to define the cluster boundary. After clustering, we first project the points of a cluster C_i onto its H_i, and find the convex hull of these projected points on H_i by a quick-hull finding algorithm. The region enclosed by the convex hull is the valid region for the MLS surface. The convex hull usually has too many edges (see the red lines in Fig. 5 (a)). This slows down the speed to judge whether a point is within the region or not. Therefore, we like to reduce the number of edges of this convex hull as follows. First, after finding its convex hull, we randomly select a point on the convex hull and examine the included angle between the two edges connecting to it. If this angle is beyond a threshold, the point is removed from the convex hull. This procedure is repeated until all the points on the convex hull have been visited. The red lines in Fig. 5 (b) show the modified convex hull. Finally, we inflate this region by an offset to prevent the gaps between adjacent clusters. This approach helps us to quickly decide whether a point is inside a cluster or not.

3. MODEL DEFORMATION

3.1 Cluster Deformation

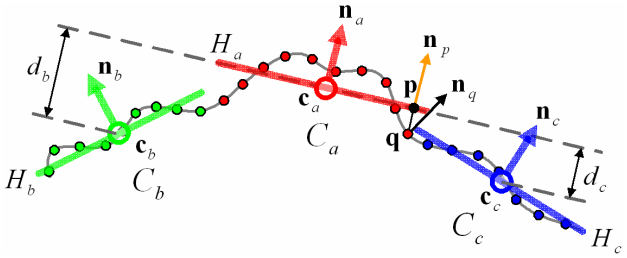
In our current implementation, we employ a conventional free-form deformation (FFD) technique [3] to deform models. Other free-form deformation techniques such as [2, 4] can be easily integrated into our approach. Our approach does not directly apply FFD to deform each sampled point. Instead, we only use FFD to re-calculate or deform the following information for each cluster C_i: (1) its cluster position c_i, and (2) its local coordinate frame. The local coordinate frame is determined by the cluster normal n_i and the other two orthogonal vectors s and t spanning the cluster base domain H_i. After deformation by FFD, the local coordinate (s, t, n) is mapped to $(\hat{s}, \hat{t}, \hat{n})$. We use the first fundamental form [16] at c_i defined in Eq. (2) to measure the local distortion of a surface under deformation.

$$\begin{bmatrix} \hat{s}^2 & \hat{s}\cdot\hat{t} \\ \hat{s}\cdot\hat{t} & \hat{t}^2 \end{bmatrix} \tag{2}$$

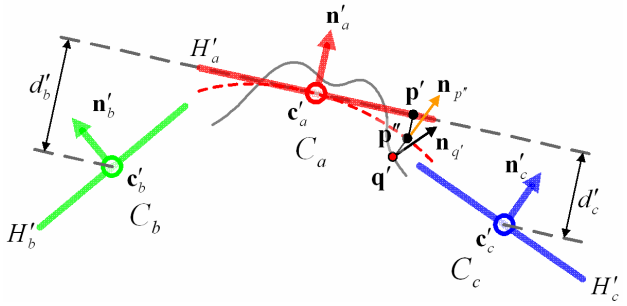
The amount of distortion can be measured by taking the ratio of the two eigenvalues of Eq. (2). This distortion measurement will be used as the criterion to split the cluster in section 3.3. We also adjust cr_i to cr_i' by multiplying the largest eigenvalue.

3.2 Local Surface Deformation Using Interpolation

After FFD, each cluster C_i has new position c_i', new normal $n_i' = \frac{\hat{n}_i}{|\hat{n}_i|}$, and new base domain H_i' (H_i' is spanned by $\hat{s}_i$ and $\hat{t}_i$). Our strategy for local surface deformation is described as follows. First, we resample the new base domain H_i' of a cluster uniformly (see section 3.4). These samples can be mapped between H_i and H_i' directly using the coordinate transformation between (s, t) and $(\hat{s}, \hat{t})$. For each sample on H_i', we deform the new base domain H_i' using a novel interpolation scheme, then add the corresponding displacement from g_i to get the deformed information for the sample, including position and normal. Figs. 6 and 7 shows the idea behind our interpolation scheme in two-dimension and three-dimension respectively. In this figure, C_a is the cluster to be reconstructed. We first construct a local angular parameterization (*i.e.*, polar coordinates) on H_a'. For a sample p' on H_a', we use its angle value to find the two neighboring clusters C_b and C_c. The signed distances before and after deformation of cluster C_b and C_c to local base domain (H_a or H_a') are d_b, d_b', d_c and d_c', respectively (in Fig. 6, these four distances are negative). Let $\Delta_b = d_b' - d_b$ and $\Delta_c = d_c' - d_c$. We first interpolate this value between cluster C_b and C_c using $\Delta_{w'} = \frac{\theta_c\Delta_b + \theta_b\Delta_c}{\theta_b + \theta_c}$, where θ_b and θ_c are the included angles between cluster projection on H_a' and p' as illustrated in Fig. 7 (a). This $\Delta_{w'}$ is related to the location, $w' = \frac{\theta_c p_b' + \theta_b p_c'}{\theta_b + \theta_c}$ on H_a', where p_b' and p_c' are the projection of c_b' and c_c' on H_a'. Note that, w' may not locate on line $\overline{c_a' p'}$ exactly, but it is always very close to this line. Then we assume H_a' is curved to become a circular arc along line $\overline{c_a' p'}$, thus we can find

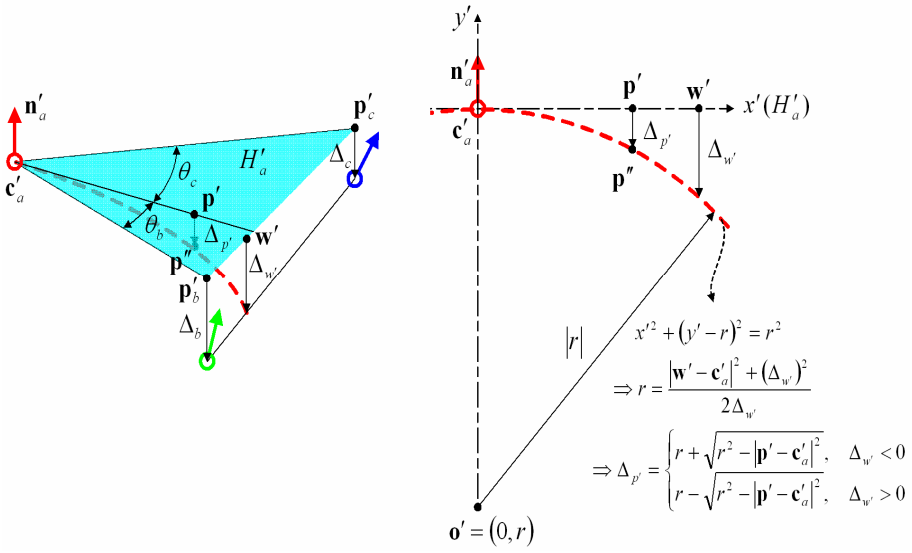


(a) Samples on clusters C_a, C_b and C_c before deformation.



(b) New sample q' corresponding to q of cluster C_a.

Fig. 6. Local surface interpolation.



(a) (b)

Fig. 7. (a) The 3D view of our interpolation scheme for curving the base domain H_a'; (b) The profile of (a) along line $\overline{c'_a p'}$. Note c_a' is the origin of the local coordinate.

$\Delta_{p'}$ using a circle equation as shown in Fig. 7 (b). Finally, we map the point p' to p'' (on the curved base domain) by Eq. (3).

$$p'' = p' + \Delta_{p'} n_a' \tag{3}$$

The normal of p'' is $n_{p''} = nor(p'' - o')$, where $nor(\ )$ is the vector normalization function.

Next, the point q' on the deformed surface corresponding to p' is computed by adding the scaled local displacement along $n_{p''}$ to p''.

$$q' = p'' + \frac{|\hat{n}_a|}{|n_a|} g_a(p) n_{p''} \tag{4}$$

In our method, the original points are discarded after the MLS surface of each cluster has been constructed. All new points are generated by resampling the MLS surface (section 3.4). Therefore, the normal of a new point should be obtained from the related MLS surface. Under the local coordinate (s, t, n) of C_a, we have $q = (x_q, y_q, z_q)$ and $z = g_a(x, y)$. This equality can be reorganized as $F(x, y, z) = z - g_a(x, y) = 0$. The normal of this equation at q is

$$n_q = \nabla F(x_q, y_q, z_q) = \frac{\partial F(x_q, y_q, z_q)}{\partial x} s + \frac{\partial F(x_q, y_q, z_q)}{\partial y} t + \frac{\partial F(x_q, y_q, z_q)}{\partial z} n. \tag{5}$$

After deformation, the base domain is curved implicitly, thus the Eq. (5) cannot be applied directly on C_a. Therefore, for each vertex on curved base domain, we project the $\hat{s}$ and $\hat{t}$ onto the plane with normal $n_{p''}$ to form a $(s_{p''}, t_{p''}, n_{p''})$ coordinate. Then, using Eq. (5) by replacing (s, t, n) with $(s_{p''}, t_{p''}, n_{p''})$, we can get the normal $n_{q'}$. In practice, we won't do the above step for each sample, because it takes a lot of computational time. Instead, we use $n_a' \cdot n_{p''}$ as a threshold to decide whether the new local coordinate should be formed or not. If $n_a' \cdot n_{p''}$ is larger than a user specified value, we just use $(\hat{s}, \hat{t}, \hat{n})$ coordinate to compute n_q.

3.3 Cluster Splitting

As mentioned above, we use the ratio of the two eigenvalues of Eq. (2) to measure the stretch of each cluster. If this ratio of a cluster is larger than a user specified value, it means the local deformation of this cluster is extreme and its local coordinate and enclosing sphere radius cr_i will become inaccurate. Therefore, this cluster will be split into two clusters at the c_i along the eigenvector corresponding to the largest eigenvalue. The boundaries of these two new clusters are defined by using the boundary of the original cluster and this splitting plane. The MLS surface of the new clusters can be obtained easily by translating the local coordinates on the base domain (s', t').

3.4 Re-sampling and Rendering

In our approach, the original points of each cluster are discarded after the MLS surface has been constructed. The new point set surface is generated by re-sampling the deformed base domain and the interpolation scheme mentioned in section 3.2. Based on our interpolation scheme, we can resample the deformed base domain dynamically while

rendering. However this strategy slows down the rendering speed dramatically. Instead, we first resample the deformed base plane densely by a user specified density. Then, similar to Wand *et al*.'s approach [20], we randomly pick the sufficient points to be rendered according to the sampling density of the current camera on the point set surface. This sampling density can be calculated using the camera-sampling field (CSF) proposed in our previous work [21]. Let A_i be the area of cluster C_i, and c be the camera-sampling field of the current camera. The number of point needed to be drawn is calculated by the ceiling function $\lceil (c \cdot n_i) A_i \rceil$. This number approximates the number of samples that the current camera samples the surface of a cluster C_i.

4. IMPLEMENTATION AND RESULTS

Our method has been implemented using the OpenGL API on a system with Pentium IV 2.4GHz CPU and 1GB RAM running Windows 2000 OS. As the user manipulates a lattice of control grids surrounding a given point-sampled model, the model can be free-form deformed well. We have tested our approach on a variety of point-sampled models. We implemented three of the most deformations of FFD: the ability to twist, bend, and stretch/squash. Additionally, some interesting effects such as swing, inflation/shrinking and melting were demonstrated, too. In our examples, the geometric details are well maintained after deformation. In contrast to other approaches such as [14, 16], we do not require a decomposition operator on models to separate low and high frequency information before deformation. Therefore, we can save computational cost significantly. Our approach provides users the powerful free-form deformation directly on point-sampled surface. A digital video clip of examples presented in this paper can be found on this Web site: http://graphics.csie.ncku.edu.tw/FFD/FFD_demo.mp4. In Fig. 1, we demonstrate a local deformation, *i.e*., swinging the head of the model of the Stanford bunny (originally 168,900 points) whose head is rotating about a fixed axis. We can see that bunny's skin and ears follow the movements dynamically. In this example, the surface of the deformed bunny model still retains the details, *i.e*., the fur, the nose and the details of the bunny ears and feet, which are similar to those on the original model surface. Fig. 8 (a) shows a global deformation example, *i.e*., shrinking and inflating a rabbit model (originally 60,002 points). Fig. 8 (b) is an interesting example of animating eight feet of a detailed Octopus model (originally 791,992 points) and this example includes a close view to the shapes of suckers on the feet that is still maintained well after a local deformation on them. In Fig. 8 (c), the bunny model acts like a soft elastic model that is gradually melting to the ground. In Fig. 8 (d), we show our CGVSL logo model (originally 104,458 points) that is bended and deformed using a wave function. In Fig. 8 (e), Michelangelo's David model (originally 273,097 points) is bended at large scale backwards and forwards, and is finally squashed into the ground. Finally, in Fig. 8 (f), we twist, stretch and bend the Igea model (originally 297,388 points) in an animation sequence. For the examples in this paper, the preprocessing stages are done in few seconds and the deformation stages can achieve interactive frame rates of between 0.8 second to 5.7 seconds on average. Fig. 9 shows a zoomed example before and after deformation to demonstrate the quality of the proposed technique. We can see that the quality of the deformed model is similar to the original one. Fig. 10 shows some statistical data for

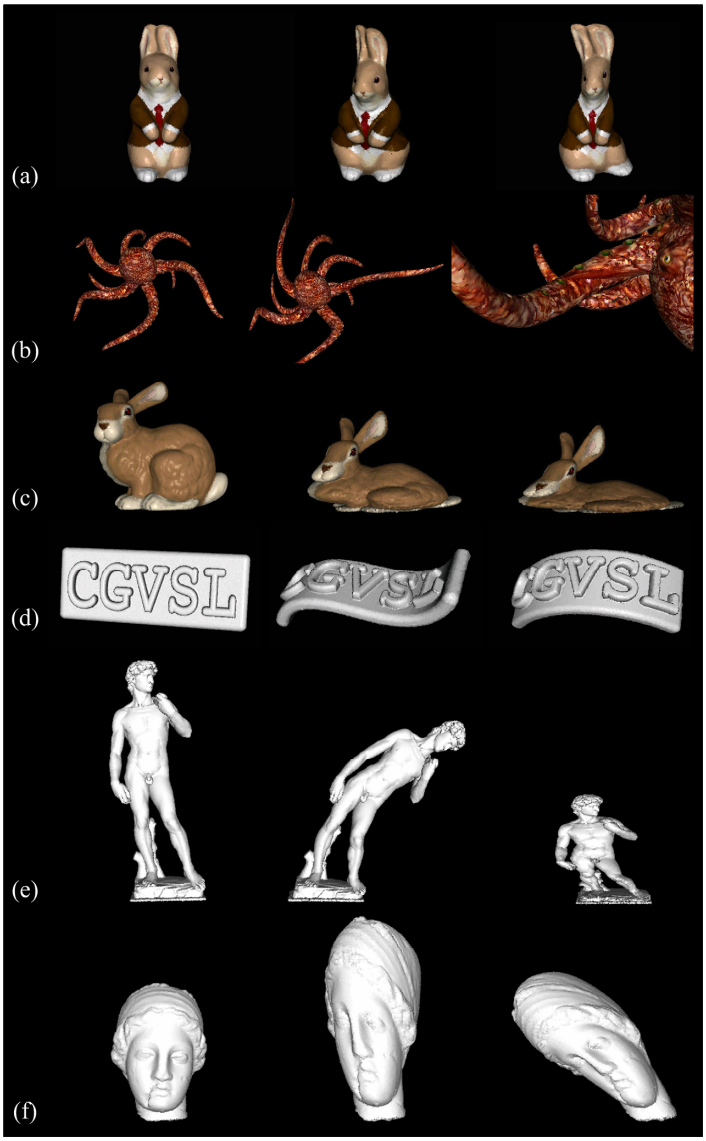


Fig. 8. The free-form deformation results using the proposed method.

(a) (b)

Fig. 9. A close view of (a) original point model and (b) deformed model.

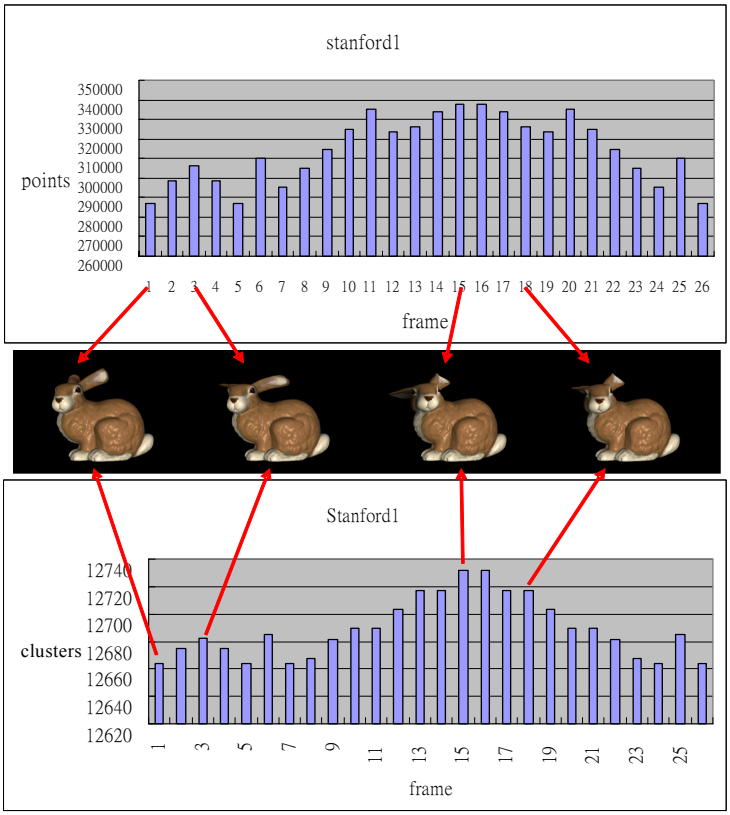


Fig. 10. Stanford Bunny model local deformation.

deforming the Stanford Bunny model. Note that, the label 'points' in this figure indicates the number of points of the re-sampled point model using our technique. Therefore, in frame 1, the statistical data is already different with the number of points of the original model. In this example, it is composed of several local deformations. As you can observe, the number of clusters increases when the deformation becomes larger. This is due to our region splitting method to maintain the accuracy of each region's MLS surface. Fig. 11 shows another experimental data for the deformation of the octopus model. In this example, we can observe the similar phenomenon. In above three figures, we demonstrate the quality, the adaptive cluster splitting and re-sampling of the proposed method. Currently, to render our model, we use a purely software-based splatting to draw points. In future, other hardware supported renderers such as [7] can be used to further speed up rendering cost.

5. CONCLUSION AND FUTURE WORK

We present a free-from deformation approach to manipulate the point-set surfaces. This technique provides users the powerful free-form deformation directly on point-sampled surface at interactive rates. Experimental results show that this new approach

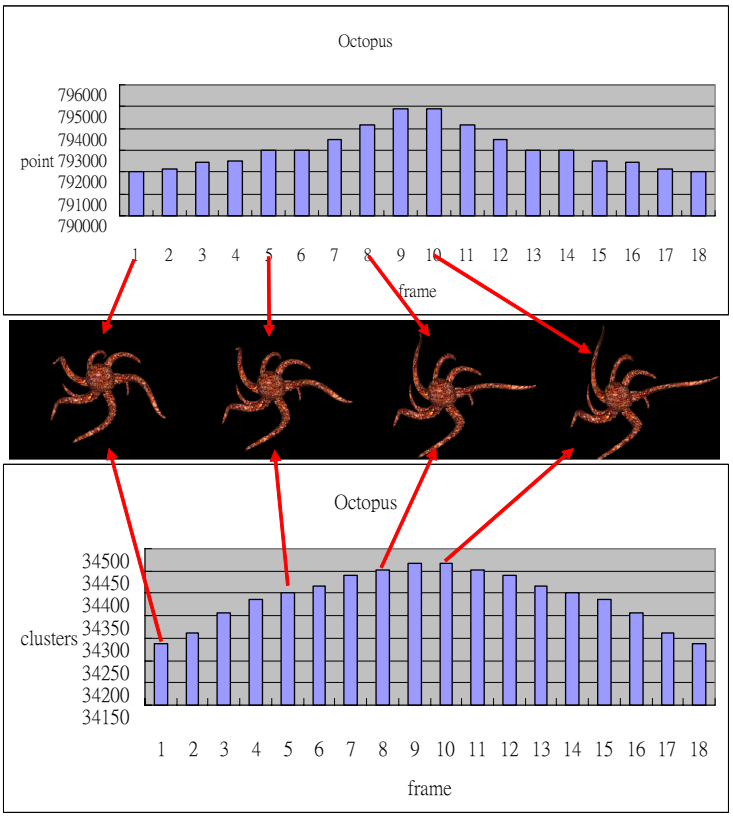


Fig. 11. Octopus model local deformation.

successfully free-form deforms point-set models and still retains the details of the original models. Although we currently only implement the conventional FFD technique [3], we believe that other free-form deformation techniques can be easily integrated into our approach. In future, we plan to design a better user interface to alleviate the cumbersome, time-consuming manipulation of individual points in a control lattice of FFD. We will like to implement some Boolean operations or local editing, and sculpting to enhance the current system. In addition, we also plan to explore the possibility of applying merging [22] or progressive approach [23] from mesh morphing to generate the metamorphosis of point-based surface. Finally, to render NPR style of point-based surface [24] with deformation will be another challenging topic to be investigated.

REFERENCES

1. M. Levoy and T. Whitted, "The use of points as a display primitive," Technical Report No. 85-022, UNC-Chapel Hill Computer Science, 1985.
2. A. H. Barr, "Global and local deformations of solid primitives," *Computer Graphics*, Vol. 18, 1984, pp. 21-30.
3. T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric mod-

els," *Computer Graphics*, Vol. 20, 1986, pp. 151-160.
4. C. Sabine, "Extended free-form deformation: a sculpting tool for 3D geometric modeling," *Computer Graphics*, Vol. 2, 1990, pp. 187-196.
5. M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *Proceedings of the Conference on Visualization*, 2002, pp. 163-170.
6. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, 2003, pp. 3-15.
7. S. Rusinkiewicz and M. Levoy, "Qsplat: a multiresolution point rendering system for large meshes," in *Proceedings of SIGGRAPH*, 2000, pp. 343-352.
8. H. Pfister, M. Zwicker, J. van Barn, and M. Gross, "Surfels: surface elements as rendering primitives," in *Proceedings of SIGGRAPH*, 2002, pp. 335-342.
9. M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "Surface splatting," in *Proceedings of SIGGRAPH*, 2001, pp. 371-378.
10. L. Ren, H. Pfister, and M. Zwicker, "Object space EWA surface splatting: a hardware accelerated approach to high quality point rendering," in *Proceedings of EUROGRAPHICS*, *Computer Graphics Forum*, Vol. 21, 2002, pp. 461-470.
11. A. Kalaiah and A. Varshney, "Modeling and rendering points with local geometry," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, 2003, pp. 30-42.
12. M. Pauly and M. Gross, "Spectral processing of point-sampled geometry," in *Proceedings of SIGGRAPH*, 2001, pp. 379-386.
13. M. Zwicker, M. Pauly, O. Knoll, and M. Gross, "Pointshop 3D: an interactive system for point-based surface editing," in *Proceedings of SIGGRAPH*, 2002, pp. 322-329.
14. M. Pauly, L. P. Kobbelt, and M. Gross, "Multiresolution modeling of point-sampled geometry," Technical Report No. 378, Computer Science Department, ETH Zurich, 2002.
15. B. Adams and P. Dutre, "Interactive boolean operations on surfel-bounded solids," in *Proceedings of SIGGRAPH*, 2003, pp. 651-656.
16. M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross, "Shape modeling with point-sampled geometry," in *Proceedings of SIGGRAPH*, Vol. 22, 2003, pp. 641-650.
17. X. Guo, J. Hua, and H. Qin, "Enhancing interactive editing on point set surfaces through touch-based haptics," *IEEE Computer Graphics and Applications*, Vol. 24, 2004, pp. 31-39.
18. J. Hua and H. Qin, "Haptics-based volumetric modeling using dynamic spline-based implicit functions," in *Proceedings of IEEE Symposium on Volume Visualization and Graphics*, 2002, pp. 55-64.
19. M. Muller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa, "Point based animation of elastic, plastic and melting objects," in *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004, pp. 141-151.
20. M. Wand, M. Fischer, P. Ingmar, F. M. auf der Heide, and W. Straer, "The randomized z-buffer algorithm: Interactive rendering of highly complex scenes," in *Proceedings of SIGGRAPH*, 2001, pp. 361-370.
21. P. H. Lin and T. Y. Lee, "Camera-sampling field and its applications," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 10, 2004, pp. 241-251.
22. T. Y. Lee and P. H. Huang, "Fast and intuitive metamorphosis of 3D polyhedral

models using SMCC mesh merging scheme," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, 2003, pp. 85-98.
23. C. H. Lin and T. Y. Lee, "Metamorphosis of 3D polyhedral models using progressive connectivity transformations," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 11, 2005, pp. 2-12.
24. M. T. Chi and T. Y. Lee, "Stylized and abstract painterly rendering system using a multi-scale segmented sphere hierarchy," to appear in *IEEE Transactions on Visualization and Computer Graphics*, 2006.

Ping-Hsien Lin (林炳賢) received the B.S. degree in Mechanical Engineering and the Ph.D. degree in Computer Engineering from National Cheng Kung University, Taiwan, in 1993 and 2004, respectively. He is currently an assistant professor in the Department of Computer Science and Information Engineering at National Changhua University of Education in Taiwan, R.O.C. His research interests include computer graphics, computer vision, and image-based rendering.

Tong-Yee Lee (李同益) was born in Tainan county, Taiwan, Republic of China, in 1966. He received his B.S. in Computer Engineering from Tatung Institute of Technology in Taipei, Taiwan, in 1988, his M.S. in Computer Engineering from National Taiwan University in 1990, and his Ph.D. in Computer Engineering from Washington State University, Pullman, in May 1995. Now, he is a Professor in the Department of Computer Science and Information Engineering at National Cheng Kung University in Tainan, Taiwan, R.O.C. He serves as a guest associate editor for IEEE Transactions on Information Technology in Biomedicine from 2000 to 2005. His current research interests include computer graphics, non-photorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, distributed & collaborative virtual environment. He leads a Computer Graphics Group/Visual System Lab at National Cheng Kung University (http://graphics.csie.ncku.edu.tw). He is a member of the IEEE.

Cheng-Fon Lin (林震凡) received the B.S. degree from the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, in 2002, and the M.S. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, in 2004. His research interests include computer graphics, computer vision, and image processing.