Direct Reconstruction of a Displaced Subdivision Surface from Unorganized Points

Won-Ki Jeong
Max-Planck-Institut für Informatik, Saarbrücken, Germany
E-mail: jeong@mpi-sb.mpg.de

and

Chang-Hun Kim1
Department of Computer Science and Engineering, Korea University, Seoul, Korea
E-mail: chkim@korea.ac.kr

Received October 18, 2001; revised June 20, 2002; accepted June 28, 2002

In this paper we describe the generation of a displaced subdivision surface directly from a set of unorganized points. The displaced subdivision surface is an efficient mesh representation that defines a detailed mesh with a displacement map over a smooth domain surface and has many benefits including compression, rendering, and animation, which overcome limitations of an irregular mesh produced by an ordinary mesh reconstruction scheme. Unlike previous displaced subdivision surface reconstruction methods, our method does not rely on a highly detailed reconstructed mesh. Instead, we efficiently create a coarse base mesh, which is used to sample displacements directly from unorganized points, and this results in a simple process and fast calculation. We suggest a shrink-wrapping-like shape approximation and a point-based mesh simplification method that uses the distance between a set of points and a mesh as an error metric to generate a domain surface that optimally approximates the given points. We avoid time-consuming energy minimization by employing a local subdivision surface fitting scheme. Finally, we show several reconstruction results that demonstrate the usability of our algorithm.

Key Words: surface reconstruction; displaced subdivision surface; surface fitting.

1. INTRODUCTION

With the improvement of optical and mechanical technology and the need for realistic modeling, acquiring accurate surface information from a real object has become
commonplace. Such technologies, including laser scanning, mechanical probe, and structured light, give a set of unorganized points as output which should be converted into parametric surfaces or polygonal meshes. Hence, there is a large amount of literature on mesh reconstruction algorithms that give dense irregular polygonal meshes as output. Since such meshes are appropriate for expressing fine surface detail but also notorious for their huge amount of data, a number of mesh processing algorithms—simplification, multiresolution, compression, etc.—have been developed so far.

The displaced subdivision surface, proposed by Lee et al. [15], is a new mesh structure that represents a detailed model as a scalar displacement map over a smooth subdivision surface. This dramatically reduces the amount of data since it requires only a scalar value for expressing each 3D vertex in addition to the storage costs for the coarse base mesh. This can be regarded as lossy compression since a displaced subdivision surface approximates a given detailed mesh with a coarse mesh and a displacement map. The Loop subdivision scheme [16] defines parameterization and smoothness of the domain surface automatically. In addition, it might be used to create bump maps for enhanced rendering. With these benefits, the displaced subdivision surface can be a new mesh structure to overcome the limitations of irregular meshes produced by ordinary mesh reconstruction algorithms. But to date, a displaced subdivision surface has only been generated by reverse engineering; i.e., an explicit detailed surface must be given in advance. This means that we have to explicitly reconstruct a highly detailed mesh before sampling displacements, which can be an expensive process when the number of points is very large (solid arrows in Fig. 1). In addition, only geometry—positions in space—is actually sampled from the object so it would be reasonable to sample the surface detail directly from points.

Hence, we suggest a new mesh reconstruction algorithm that produces a displaced subdivision surface directly from unorganized points. Directly in our text means that we generate a coarse base mesh and sample displacements directly from unorganized input points instead of reconstructing a highly detailed mesh from which the displacements are computed. The overall shape of the given points is approximated by shrink-wrapping, which is much cheaper than reconstructing a mesh from points when we deal with a large number of input points. Thus, the expensive process described above is replaced by a much cheaper process, which is depicted using dashed arrows in Fig. 1. To generate an optimal control mesh from

![Diagram](image.png)

**FIG. 1.** Comparison between the process of the previous displaced subdivision surface reconstruction (solid arrows) and our method (dashed arrows). Substituting the reconstruction of a highly detailed mesh by shrink-wrapping and avoiding expensive mesh optimization makes our method more efficient than previous approaches.
the input points, a point-based mesh simplification is suggested. Since we would like to sample the surface detail directly from points, we have to calculate an accurate intersection between a sampling ray and a virtual surface inferred by the input points. We suggest two different solutions for this, i.e., finding ray–triangle intersections and fitting a smooth surface.

The main benefit of our algorithm is the efficient way in which the underlying mesh structure is generated. The output of our algorithm has a piecewise-regular structure that is created by successive subdivision. Since this mesh has subdivision connectivity, we can create a multiresolution mesh directly without remeshing. Moreover, large data sets can be reconstructed and manipulated fast and efficiently since the output of our algorithm has a memory-efficient structure.

1.1. Previous Work

1.1.1. Mesh Reconstruction

Many publications on 3D reconstruction from unorganized points exist in the computer vision and graphics community. Most mesh reconstruction schemes have focused on the approximation of a smooth parametric surface to given points or the derivation of a zero-set of an implicit function [4]. A radial basis function approach, which can be used for reconstructing the incomplete scan data, was suggested by Carr et al. [5]. Recently, extracting a triangular mesh from a given set of points was studied. Hoppe et al. [9–11] proposed an arbitrary 3D mesh reconstruction technique from unorganized points. He introduced a volume-based reconstruction with the optimization of energy functions, using the modified Loop subdivision scheme to optimize the result. This method has several advantages—including the ability to reconstruct meshes of arbitrary topology with optimal and robust results—but solving expensive linear systems is required. Suzuki et al. [20] proposed the subdivision surface fitting algorithm that uses the limit surface property of approximating subdivision schemes. He changes the shape of the control mesh at every level of subdivision to make the limit surface optimally fit to the points. This uses only local information and requires less computation, but the result may have a lack of fine surface detail. Amenta et al. [2] suggested a medial-axis and Voronoi-based surface reconstruction algorithm. It is an interpolating method, which means that the vertices of the resulting mesh coincide with some input points. It is robust and gives adaptive resolution, but it requires the Delaunay triangulation operation, which is somewhat expensive.

1.1.2. Displacement Maps

Recently, several algorithms have been proposed that convert an arbitrary mesh into a smooth surface and a set of displacements. The main benefits of these works are storage efficiency and efficient rendering by bump mapping. Krishnamurthi et al. [14] proposed a method of smooth surface fitting to a polygonal mesh. They manually divide the input mesh into several sections and fit B-spline surfaces to them. After the fitting process, they sample fine surface detail with displacement vectors. This method gives good smooth surface fitting, but it requires a large amount of memory, since the displacements are stored as three-dimensional vectors. Moreover, this method needs a lot of manual processing for dividing the input mesh into the patch domains and keeping continuity between each patch boundary. Recently, Lee et al. [15] proposed another displacement sampling algorithm that
uses a subdivision scheme to produce a smooth parametric surface and a scalar displacement value for sampling surface detail. Guskov et al. [8] suggested a similar mesh structure based on the butterfly subdivision scheme. This mesh stores displacement values in a multiresolution hierarchy, and heavy computation of several different parameterizations is needed. Approaches creating a head model using a displaced subdivision surface have also been introduced [12, 18].

1.2. Contributions

The main contribution of this paper is the reconstruction of a displaced subdivision surface directly from a given set of points. Thus, our algorithm does not require a highly detailed reconstructed mesh to generate a domain surface or sample fine surface detail. From another point of view, it is also reasonable to use only the positional information of the input points. Since the input points do not carry any connectivity information, the reconstruction of a detailed mesh that is in turn used for sampling surface detail is artificial. This is the summary of our contributions:

1. We suggest a new approach that produces a displaced subdivision surface directly from a given set of unorganized points without requiring a highly detailed mesh, which is expensive to generate for a large number of input points.
2. We suggest shrink-wrapping for the initial shape approximation and present a point-based mesh simplification algorithm for generating a control mesh that approximates input points well.
3. We avoid time-consuming global energy optimization by employing a local subdivision surface fitting scheme and efficient surface detail sampling from a point set so that we can generate a displaced subdivision surface in a short time.

2. OVERVIEW

The input to our algorithm is a set of points \( X = \{x_0, x_1, \ldots, x_n\} \) without connectivity information, which can be acquired, e.g., using 3D scanning hardware such as a laser range scanner or a structured light scanner. We assume that the given points are already registered in the same coordinate system and do not exhibit significant noise.

A displaced subdivision surface consists of two parts: a coarse control mesh \( M_c \) and a displacement map \( D = \{d_0, d_1, \ldots, d_m\} \), consisting of a set of scalar distances along each vertex normal (see Fig. 2). By repeated refinement using a subdivision operator \( S \), followed by application of the detail information \( D \), we construct a mesh \( M \) approximating the

![Fig. 2](image)

**Fig. 2.** Displacement sampling process shown in 2D. Black dots are vertices of the parametric domain mesh, and white dots are input points. \( d_i \) is the displacement value sampled from the corresponding vertex \( v_i \). Arrows are sampling vectors.
geometry described by the points $X$:

$$\mathcal{M} = M_p + D,$$

where

$$M_p = S^k M_c,$$

and $k$ is the subdivision level. Since the subdivision operator $S$ is known and does not need to be stored, we only have to store the coarse mesh $M_c$ and the scalar values in the displacement map $D$. This is the reason for the memory-efficient representation of an object using a displaced subdivision surface. In Section 3, we describe the construction of the control mesh $M_c$, the parametric domain $M_p$, and the sampling of corresponding displacements $D$ for a given level $k$ in detail.

We restrict the topology of $\mathcal{M}$ to be genus-0. We will strictly distinguish “point” and “vertex” when they are referred. A point denotes an element of the point set $X$, and a vertex denotes a vertex of a triangle mesh throughout the text. The following is an overview of our reconstruction algorithm.

1. **Control mesh generation**: The control mesh is a coarse base mesh that approximates the given points. We subdivide this control mesh to generate a parametric domain surface in the next step. To compute a control mesh from points, we first build a bounding box of the points, subdivide it, and shrink-wrap it to the points. Then, we simplify the shrink-wrapped bounding box using the point-based simplification algorithm whose error metric is the distance from points to a mesh.

2. **Parametric domain surface generation**: We subdivide the control mesh to get a smooth parametric domain surface that is a base mesh of surface detail sampling. We employ a local subdivision surface fitting scheme [20] to prevent shrinkage induced by the subdivision scheme and to avoid heavy computation of global optimization.

3. **Displacement sampling**: The last step is sampling fine surface detail from the points. We approximate local shape using smooth polynomial surfaces around each vertex of the domain surface and find a distance from the domain surface to the target surface along the normal direction.

### 3. RECONSTRUCTION ALGORITHM

In this section we describe our displaced subdivision surface reconstruction algorithm in detail.

#### 3.1. Control Mesh Generation

The control mesh generation is the critical process in our algorithm since a control mesh defines the shape of the parametric domain surface and finally affects the quality of surface detail sampling. A control mesh should be coarse while approximating the given points. In this section, we describe the initial approximation of a given point set by a triangle mesh using shrink-wrapping. Then, a coarse mesh is created by applying the point-based mesh simplification described in Section 3.1.2.
FIG. 3. Shrink-wrapping subdivided bounding boxes to the points. The initial bounding box (a) and shrink-wrapping process of the subdivided box (b)–(e). The initial bounding box is linearly subdivided up to level 5 before the shrink-wrapping is applied.

3.1.1. Initial Shape Approximation Using Shrink-Wrapping

The idea of shrink-wrapping remeshing was given in [13]. Our approach shares the idea of shrinking a bounding box so that it completely wraps up the input points (Fig. 3), but the implementation is somewhat different from that because the input of our algorithm is just a set of points that cannot be projected to a common base domain like a sphere in [13].

First, we compute a bounding box $M_b$ of the given points (Fig. 3a), which consists of 12 triangles (two triangles per each face of a bounding box). Then we linearly subdivide the box up to a user-defined level; i.e., each triangle is divided into four subtriangles and new vertices are placed at the middle of each edge. To simulate a shrink-wrapping process, we iteratively apply two basic operations: projection and smoothing.

The projection operation applies an attracting force to each vertex. The attracting force $f$ of a vertex is a vector between the subdivided bounding box $M_b$ and the given points $X$. For each vertex $v_i$ of the bounding box, we simply find the nearest point $x_j$, calculate an attracting force vector $f_{v_i}$, and apply it back to the vertex $v_i$ with a projection speed parameter $\mu \in [0, 1]$. Finding a nearest point is a well-known problem, and we used the kd-tree searching algorithm in our implementation [3, 6].

$$f_{v_i} = x_j - v_i$$
$$M_b = \sum_i (v_i + \mu f_{v_i})$$

For a subdivided bounding box, it is possible that two or more vertices of the box are attracted by the same input point. In such a case, if we apply full attraction force ($\mu = 1.0$) to those vertices, some of them can meet at the same position by this projection operation causing nonmanifold regions. To avoid such artifacts, we assign $\mu$ a value less than 1.0.

The smoothing operation is a relaxation process of the subdivided bounding box to achieve uniform sampling. We employ an approximation of Laplacian $\mathcal{L}$ as in [21]. This is an average vector of 1-neighbor edge vectors of a given vertex, and its application usually results in shrinkage. Thus we take the tangential component $\mathcal{L}_t$ (Fig. 4a) of this Laplacian $\mathcal{L}$ perpendicular to the vertex normal $n$. Hence, the final tangential Laplacian $\mathcal{L}_t$ of a given
vertex $v_i$ and an iterative smoothing equation are as follows:

$$L(v_i) = \frac{1}{\text{valence}(v_i)} \sum_{v_k \in N_b(v_i)} (v_k - v_i)$$

$$L_t(v_i) = L(v_i) - (L(v_i) \cdot \mathbf{n})\mathbf{n}$$

$$M'_b = M_b + \lambda L_t.$$ 

$\lambda$ in Eq. (1) is the speed parameter that controls the convergence speed of the smoothing operation. If we use a large value for $\lambda$ (i.e., close to 1.0), we obtain a uniformly sampled mesh, but also our shrink-wrapping procedure might fail to capture detailed regions. If we apply a small value for $\lambda$, we do not get a uniformly sampled mesh. Hence, finding proper parameters is important in practice. An example of our shrink-wrapping process is given in Figs. 3a–3e. Figures 4b and 4c compare the result of the shrink-wrapping process for different parameters. Using a small $\lambda$ (Fig. 4b) gives a much more detailed result, e.g., top of ears, but there are wrinkles since several vertices may share the same position induced by the strong projection force. Figure 4c shows an opposite result because of the strong relaxation force. In our experiments, we found a value of $\lambda = 0.8$ to be a reasonable starting choice.

### 3.1.2. Point-Based Mesh Simplification

Even though the result of our shrink-wrapping process $M_b$ is much coarser than the input points, a control mesh that can be used as a base mesh for subdivision to generate a smooth surface should be coarser than this, and thus we have to simplify the mesh $M_b$ given above. In [15], the simplification of the original mesh is sufficient to generate a control mesh since the original mesh is the target of reconstruction. But in our algorithm, a set of points is the target of reconstruction. Hence, we have to simplify a mesh so that it approximates the given points as well.

Our point-based mesh simplification is an extension of the original QEM algorithm [7], which employs a different error metric. In the original QEM, the quadric distance between an original and a simplified mesh is measured. In our point-based simplification, we measure the quadric distance between points and a simplified mesh and perform selective edge-collapses. In order to measure the distance from points to a mesh, we find a set of tangent planes that approximate the given points and calculate error quadrics based on these planes.
First, we find a tangent plane for each point in a given set of points. A tangent plane \( T_x(v) = n \cdot v + d \) associated with a point \( x \) is defined by the least squares fitting plane to the neighborhood \( N_b(x) \) of \( x \), which consists of all points \( x_i \in X \) such that \( |x - x_i| < \epsilon \). The nearest points are found in logarithmic time by a spatial searching algorithm [3, 6]. The user defined \( \epsilon \) should be large enough so that the neighborhood represents the local surface shape correctly. Usually 3-10 points are used to calculate a tangent plane in our implementation. After that, we project every point to the mesh. Then we assign corresponding points to each vertex \( v \) of the mesh by collecting all projected points from the 1-neighbor triangles around \( v \).

We define the fundamental error quadric \( K_x \) according to a tangent plane \( T_x \). Since we know the plane equation of \( T_x \), we can define the vector \( p = [abcd]^T \) where \( [abc] = n \) and \( d = T_x(O) \), \( O \) is the origin. Then \( K_x \) is defined as in [7]:

\[
K_x = pp^T = \begin{bmatrix}
a^2 & ab & ac & ad \\
ab & b^2 & bc & bd \\
ac & bc & c^2 & dc \\
ad & bd & cd & d^2
\end{bmatrix}
\]

Then the initial error quadric matrix \( Q_v \) for a vertex \( v \) in the mesh \( M_b \) is just the sum of fundamental quadrics of points projected into one of the 1-neighbor triangles of \( v \),

\[
Q_v = \sum_{x_i} (K_{x_i})
\]

where \( x_i \) is the point projected into a 1-neighbor triangle of \( v \).

Finally, we calculate the cost of all edges in \( M_b \) as described in the original QEM paper [7] and build a heap according to the cost. Next, we collapse the edge on top of the heap and update the new error quadric \( Q \) after edge-collapse by just adding two error quadrics of the end vertices of the removed edge. We repeat the edge-collapse operation until the mesh reaches the user-defined level. In our experiments, we simplify the mesh until the number of vertices of the mesh obtained by subdividing the control mesh five times is approximately equal to the number of input points.

As shown in Figs. 5b and 5d, our point-based simplification produces control meshes that approximate the input points better for a low number of vertices. Our result was also numerically much closer to the point set compared with that of original QEM: the ratio of the average distance from points to the simplified mesh divided by the length of the diagonal of the bounding box is 2.923 \( \cdot 10^{-3} \) for our approach and 3.261 \( \cdot 10^{-3} \) for the original QEM [7]. Since we only use a point set for the whole reconstruction process, we have to generate a control mesh that approximates the given points, and our point-based simplification performs this task very well.

3.1.3. Discussion

To make a coarse base domain mesh from a given set of unorganized points, we employed the shape approximation method that detects topological information; i.e., shrink-wrapping of a bounding box and point-based mesh simplification. Since the target of our reconstruction is a set of points, the above-mentioned shrink-wrapping and simplification techniques are modified to be worked with points only. One limitation of our method is that the shrink-wrapping approach can be successfully applied when the given points are sampled from a
FIG. 5. Comparison of the results for different mesh simplification algorithms. Original QEM (a), (b) and our point-based QEM (c), (d), simplified up to a number of 100 vertices. The original QEM has been applied to a highly detailed mesh obtained by triangulation of the dense point cloud depicted in Fig. 9a. Our point-based QEM has been applied to the mesh shown in Fig. 9b obtained by shrink-wrapping.

well-shaped model, i.e., the topology of the model should be spheric, and the surface should not have very deep concave or convex shapes. Shrink-wrapping a complexity-increased bounding box can help the sampling of deep convex or concave regions since more vertices can be moved onto the surface and reach such regions, but it is expensive to move many vertices altogether. As in Fig. 4, a five times subdivided bounding box has been used in our experiments. We expect that using a volumetric approach can overcome such a topological limitation.

3.2. Parametric Domain Surface Generation

In this step we generate a smooth parametric domain surface $M_p$ from a coarse control mesh $M_c$. The parametric domain surface is a base mesh for sampling surface detail. Since the original displaced subdivision surface algorithm employed the Loop subdivision scheme [16] to generate a smooth surface, we follow that approach. To capture surface detail successfully, the parametric domain surface should approximate the input points well. Since the Loop subdivision induces shrinkage, we import a local subdivision surface fitting scheme [20] to modify the control mesh (Fig. 6b) in order to correctly fit the domain surface to the points after subdivision (Fig. 6c).

FIG. 6. Generating a smooth parametric domain surface: (a) control mesh with input points, (b) deformed control mesh, (c) parametric domain surface fitting to input points.
Using the Loop scheme [16], we find a limit position $v_i^\infty$ of each vertex $v_i$ of the current control mesh $\mathcal{M}_c$. Then we find the closest input point $x_j \in \mathcal{X}$ for each limit position $v_i^\infty$. The resultant force $r_i$ [20] is defined as follows,

$$r_i = (x_j - v_i) - \kappa_i \sum_{v_k \in N_b(v_i)} (v_k - v_i),$$

where

$$\kappa_i = \left( \frac{3}{8\beta} + k \right)^{-1}, \quad k = \text{val}(v_i)$$

$$\beta = \begin{cases} \frac{3}{32} & \text{for } k = 3 \\ \frac{1}{k} \left( \frac{5}{8} - \left( \frac{3}{8} + \frac{1}{4} \cos \frac{2\pi}{k} \right)^2 \right) & \text{for } k > 3. \end{cases}$$

We modify each vertex $v_i$ of the control mesh $\mathcal{M}_c$ by the following iterative approximation method:

$$v'_i = v_i + \rho r_i.$$

The optimal speed parameter $\rho$ has been suggested as 0.8 in [20], but this parameter might vary heavily. After the fitting process, we subdivide the control mesh $\mathcal{M}_c$ until we get a smooth domain surface consisting of roughly as many vertices as the number of input points.

### 3.3. Surface Detail Sampling

In a previous work [15], Lee et al. compute a signed distance from the limit position of each vertex on the parametric domain surface to the detailed mesh along the vertex normal. Since the sampling target is a polygonal mesh, it is easy to find an intersection point between a normal vector and a surface. In our approach, however, we do not have any mesh or point connectivity information. Hence, we find a proper local surface approximation for each sampling vector to compute a displacement value. In this section, we suggest two different sampling methods classified by way of approximating local shape: finding ray–triangle intersections and fitting smooth surfaces.

#### 3.3.1. Finding Ray–Triangle Intersections

In this heuristic sampling method, we find a triangle defined by three of the input points that intersects with a sampling vector and has minimum area. We call it valid triangle $T_{\text{valid}}$ in this section. Hence, for a vertex $v_i$, the displacement value $d_i$ is the distance from $v_i$ to the valid triangle $T_{\text{valid}}$ along the vertex normal direction $s$. Figure 7 shows the valid triangle and proper displacement value $d_i$ for a given parametric vertex $v_i$.

To compute a displacement $d_i$, we find $T_{\text{valid}}$ for the vertex $v_i$. The main idea is that we find a local parametric plane orthogonal to the sampling vector $v_i$, project neighbor points to the plane, and find three projected points that the triangle made with them to enclose the vector $v_i$. To make this procedure simpler, we find three points closest to a sampling vector vertically (Fig. 7b) instead of finding a parametric plane and projecting points to it explicitly. Then we test whether the vector intersects the triangle defined by these three
points. If it does, this triangle is a valid triangle and we calculate the signed distance from the domain surface to this triangle along the normal direction. If there is no intersection, we include the next closest point and try different combinations of the points to generate other triangles until we find an intersected triangle. If we get multiple intersected triangles, we choose the one that has the smallest area among all valid triangles since we assume that the domain surface is close enough to the input points and locally flat, thus making the smallest triangle representing the local geometry best.

Figure 8 shows an example of finding a valid triangle $T_{\text{valid}}$. The black dot in this figure represents the sampling vector $s$, which extends toward the reader. White dots represents five neighbor points $\{x_1, x_2, \ldots, x_5\}$ projected orthogonally to the base plane. There are 10 prospective valid triangles that can be generated out of five input points $\binom{5}{3} = 10$. 

![Diagram](image)

**FIG. 7.** Finding a valid triangle. (a) 3D view of valid triangle $T_{\text{valid}}$ and displacement value $d_i$. (b) 2D view of finding vertically close points. Dashed arrows are vertical distances to the sampling vector.

**FIG. 8.** Example of selecting a valid triangle. (a) input points $x_1$–$x_5$ and sampling vector $s$, (b)–(d) candidate triangles, (d) valid triangle.
Among these triangles, the triangles \((x_1, x_2, x_4), (x_1, x_3, x_4),\) and \((x_1, x_4, x_5)\) intersect with the normal vector \(s\) (Figs. 8b–8d), and \((x_1, x_4, x_5)\) is the smallest triangle among them. Hence, triangle \((x_1, x_4, x_5)\) is the valid triangle for the sampling vector \(s\) (Fig. 8d).

### 3.3.2. Fitting Smooth Surfaces

Using valid triangles introduced in the previous section is simple and fast, but it has a drawback—it only approximates local shape with a flat triangle. Let us assume that there are two adjacent triangles and we have many sampling vectors near the common edge of these two triangles. Then there must be a discontinuity in the smoothness across this edge except when the two triangles are coplanar. To avoid such discontinuities, we try to approximate local shape with a smooth surface instead of a flat triangle.

When we have a set of vertically close neighbor points of a sampling vector \(s\), we have to find their parametric domain plane. Notice that this parametric domain is different from the one in Section 3.2, since the parametric domain plane in this section is used only for fitting a smooth parametric surface to the neighbor points locally. We use a least squares fitting plane as the parametric domain plane. Since we have to find the intersection point of \(s\) and the local surface approximation, we restrict the plane to be orthogonal to the sampling vector \(s\). After we find a local parametric domain plane, we project all neighbor points to this plane and set up a local frame \((u, v, w)\), where \(u\) and \(v\) span the parametric plane and \(w\) is orthogonal to the plane. Then all neighbor points can be parameterized by a 2D coordinate \((u, v)\) and \(w\) is the height of the point from the parametric domain plane. The parametric coordinates \((u_i, v_i)\) of \(v_i\) are also the parametric coordinates of the intersection point.

Since we have made a parametric reference domain, we find a smooth surface best fitting to the neighbor points. We use a biquadratic polynomial \(F(u, v)\) for the smooth approximation as follows:

\[
F(u, v) = a_1 + a_2 u + a_3 v + a_4 u^2 + a_5 uv + a_6 v^2.
\]

This polynomial has six coefficients. Assume that we have \(N\) neighbor points. Then we can set up a linear system as follows:

\[
Va = w, \quad (2)
\]

where

\[
V = \begin{bmatrix}
1 & u_0 & v_0 & u_0^2 & u_0v_0 & v_0^2 \\
1 & u_1 & v_1 & u_1^2 & u_1v_1 & v_1^2 \\
& \vdots & & \vdots & & \vdots \\
1 & u_{N-1} & v_{N-1} & u_{N-1}^2 & u_{N-1}v_{N-1} & v_{N-1}^2
\end{bmatrix},
\]

\[
a = \begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
a_4 \\
a_5 \\
a_6
\end{bmatrix}, \quad w = \begin{bmatrix}
w_0 \\
w_1 \\
\vdots \\
w_{N-1}
\end{bmatrix}.
\]
In practice, the number of neighbor points is usually larger than the number of unknown coefficients, and in this case the linear system (2) can be solved for $a$ in the least-square sense by solving the normal equation

$$V^T V a = V^T w.$$  

(3)

If the number of neighbor points is either less than six or equal to six with singular matrix $V$, the linear system (2) can be solved by singular value decomposition [19].

Now, $(u_{v_i}, v_{v_i}, F(u_{v_i}, v_{v_i}))$ is the intersection point between the sampling vector $s$ and the local approximation of the smooth surface inferred by the neighborhood of $v_i$. Then we compute the displacement value of the sampling vector $s$ as the distance from $v_i$ to $(u_{v_i}, v_{v_i}, F(u_{v_i}, v_{v_i}))$, which is the sum of $F(u_{v_i}, v_{v_i})$ and the distance between $v_i$ and the parametric domain plane.

3.3.3. Discussion

We have suggested two different sampling methods in the previous sections. Finding ray–triangle intersections is simple and fast, but it may result in discontinuities. Smooth surface fitting usually results in smooth sampling, but some details may be lost and the approach is about three times slower than the other method. For example, finding ray–triangle intersections took 8 s, while the smooth surface fitting took 26 s for the displacement sampling of the Igea model (see Section 4). It has to be considered, though, that even the fitting of smooth surfaces may result in discontinuous sampling since the fitted surfaces might not be connected smoothly. We expect that this can be overcome by employing a method that guarantees global continuity as in [1]. In practice, the finding ray–triangle intersections usually results in good visual appearance and fast sampling.

Flipping or self intersection after displacement sampling might be found on the region where the domain surface and the points are too far from each other, and this happens when the shrink-wrapping fails to capture the surface shape correctly due to deep convexity or concavity. One way to prevent this is to use a bounding box that is subdivided several times for the shrink-wrapping in the initial shape approximation step as discussed in Section 3.1.3.

4. RESULTS

We tested our algorithm on an sgi Octane with a 300 MHz R12k processor. The results are shown in Figs. 9, 10, and 11. Table 1 lists the data sizes and execution times. Parametric domain surfaces are generated by subdividing each control mesh up to level 5. Displacement sampling is done by finding intersecting triangles as described in Section 3.3.1.

The shrink-wrapping process takes a lot of time, making it the primary goal for optimization. In [13], a multilevel approach for fast convergence is suggested. However, starting the shrink-wrapping on the high level of the mesh hierarchy is preferred in our approach because if shrink-wrapping is applied to the coarse level of the mesh then some concave or convex regions are likely to be missed and might be hard to recover in later processes.
### TABLE 1

<table>
<thead>
<tr>
<th>Model</th>
<th>Number Rabbit</th>
<th>Igea</th>
<th>Max Planck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input # of points</td>
<td>67038</td>
<td>134345</td>
<td>100086</td>
</tr>
<tr>
<td>Control #V/#F</td>
<td>270/536</td>
<td>377/750</td>
<td>500/996</td>
</tr>
<tr>
<td>Parametric domain #V/#F</td>
<td>68610/137216</td>
<td>96002/192000</td>
<td>127490/254976</td>
</tr>
<tr>
<td>Shrink-wrapping</td>
<td>17 s</td>
<td>1 min 23 s</td>
<td>1 min 41 s</td>
</tr>
<tr>
<td>Simplification and fitting</td>
<td>21 s</td>
<td>24 s</td>
<td>29 s</td>
</tr>
<tr>
<td>Displacement sampling</td>
<td>5 s</td>
<td>8 s</td>
<td>11 s</td>
</tr>
<tr>
<td>Total</td>
<td>43 s</td>
<td>1 min 55 s</td>
<td>2 min 21 s</td>
</tr>
</tbody>
</table>

**FIG. 9.** Result of displaced subdivision surface reconstruction (rabbit model). (a) point set, (b) shrink-wrapped bounding cube, (c) control mesh, (d) displacement sampling, (e) final result.

**FIG. 10.** Result of displaced subdivision surface reconstruction (Igea model). (a) point set, (b) shrink-wrapped bounding cube, (c) control mesh, (d) displacement sampling, (e) final result.

**FIG. 11.** Result of displaced subdivision surface reconstruction (Max Planck model). (a) point set, (b) shrink-wrapped bounding cube, (c) control mesh, (d) displacement sampling, (e) final result.
5. CONCLUSIONS

In this paper, we introduced a new approach for reconstruction of a displaced subdivision surface directly from unorganized points. Directly means we generated a coarse base mesh and sample displacements directly from unorganized input points, which is faster than previous approaches, since we avoid the highly detailed 3D mesh reconstruction and global energy minimization process. It is also reasonable since the geometry of points is the only information available from the object modeled. We generate a simple control mesh whose limit surface after subdivision is well fitted to the given points by shrink-wrapping of a bounding box, point-based mesh simplification, and the local subdivision surface fitting scheme. Scalar displacements from a domain surface to points are calculated along every vertex normal of the parametric domain surface, generated by subdividing of the control mesh. To sample displacement values from the input points, we either find ray–triangle intersections for fast computation or fit a biquadratic polynomial surfaces to neighbor points for sampling smoothly varying displacement values. The resulting surfaces can be stored compactly and used for a number of applications, e.g., LOD control, compression, and efficient rendering.

Even though we restrict the topology of the data to have genus-0, our shrink-wrapping approach might fail to wrap up the points completely when they are sampled from an object with a complex shape. So, we would like to extend our algorithm to the reconstruction of models with arbitrary topology. This can be achieved by incorporating a topology recognition scheme into our algorithm, e.g., a volumetric reconstruction method [17]. An extension of our algorithm for a noisy or imperfect point set is also required. A theoretical study of the efficiency of displaced subdivision surface is also expected as a future work.

REFERENCES


