Displaced Subdivision Surfaces

Aaron Lee
Department of Computer Science
Princeton University
http://www.aaron-lee.com/

Henry Moreton
NVIDIA Corporation
moreton@nvidia.com

Hugues Hoppe
Microsoft Research
http://research.microsoft.com/~hoppe

ABSTRACT

In this paper we introduce a new surface representation, the *displaced subdivision surface*. It represents a detailed surface model as a scalar-valued displacement over a smooth domain surface. Our representation defines both the domain surface and the displacement function using a unified subdivision framework, allowing for simple and efficient evaluation of analytic surface properties. We present a simple, automatic scheme for converting detailed geometric models into such a representation. The challenge in this conversion process is to find a simple subdivision surface that still faithfully expresses the detailed model as its offset. We demonstrate that displaced subdivision surfaces offer a number of benefits, including geometry compression, editing, animation, scalability, and adaptive rendering. In particular, the encoding of fine detail as a *scalar* function makes the representation extremely compact.

Additional Keywords: geometry compression, multiresolution geometry, displacement maps, bump maps, multiresolution editing, animation.

1. INTRODUCTION

Highly detailed surface models are becoming commonplace, in part due to 3D scanning technologies. Typically these models are represented as dense triangle meshes. However, the irregularity and huge size of such meshes present challenges in manipulation, animation, rendering, transmission, and storage. Meshes are an expensive representation because they store:

(1) the irregular connectivity of faces,

(2) the (x, y, z) coordinates of the vertices,

(3) possibly several sets of texture parameterization (u, v) coordinates at the vertices, and

(4) texture images referenced by these parameterizations, such as color images and bump maps.

An alternative is to express the detailed surface as a displacement from some simpler, smooth domain surface (see Figure 1). Compared to the above, this offers a number of advantages:

(1) the patch structure of the domain surface is defined by a control mesh whose connectivity is much simpler than that of the original detailed mesh;

(2) fine detail in the displacement field can be captured as a scalar-valued function which is more compact than traditional vector-valued geometry;

(3) the parameterization of the displaced surface is inherited from the smooth domain surface and therefore does not need to be stored explicitly;

(4) the displacement field may be used to easily generate bump maps, obviating their storage.

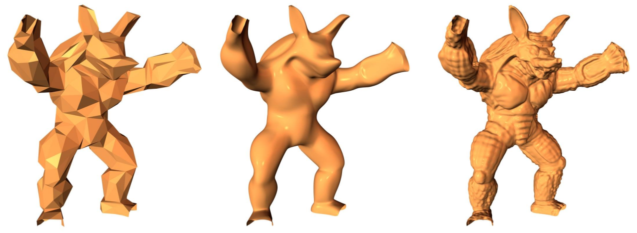

(a) control mesh (b) smooth domain surface (c) displaced subdivision surface

Figure 1: Example of a displaced subdivision surface.

A simple example of a displaced surface is terrain data expressed as a height field over a plane. The case of functions over the sphere has been considered by Schröder and Sweldens [33]. Another example is the 3D scan of a human head expressed as a radial function over a cylinder. However, even for this simple case of a head, artifacts are usually detectable at the ear lobes, where the surface is not a single-valued function over the cylindrical domain.

The challenge in generalizing this concept to arbitrary surfaces is that of finding a smooth underlying domain surface that can express the original surface as a scalar-valued offset function.

Krishnamurthy and Levoy [25] show that a detailed model can be represented as a displacement map over a network of B-spline patches. However, they resort to a vector-valued displacement map because the detailed model is not always an offset of their B-spline surface. Also, avoiding surface artifacts during animation requires that the domain surface be tangent-plane (C^1) continuous, which involves constraints on the B-spline control points.

We instead define the domain surface using subdivision surfaces, since these can represent smooth surfaces of arbitrary topological type without requiring control point constraints. Our representation, the *displaced subdivision surface*, consists of a control mesh and a scalar field that displaces the associated subdivision surface locally along its normal (see Figure 1). In this paper we use the Loop [27] subdivision surface scheme, although the representation is equally well defined using other schemes such as Catmull-Clark [5].

Both subdivision surfaces and displacement maps have been in use for about 20 years. One of our contributions is to unify these two ideas by defining the displacement function using the same subdivision machinery as the surface. The scalar displacements are stored on a piecewise regular mesh. We show that simple subdivision masks can then be used to compute analytic properties on the resulting displaced surface. Also, we make displaced subdivision surface practical by introducing a scheme for constructing them from arbitrary meshes.

We demonstrate several benefits of expressing a model as a displaced subdivision surface:

Compression: both the surface topology and parameterization are defined by the coarse control mesh, and fine geometric detail is captured using a scalar-valued function (Section 5.1).

Editing: the fine detail can be easily modified since it is a scalar field (Section 5.2).

Animation: the control mesh makes a convenient armature for animating the displaced subdivision surface, since geometric detail is carried along with the deformed smooth domain surface (Section 5.3).

Scalability: the scalar displacement function may be converted into geometry or a bump map. With proper multiresolution filtering (Section 5.4), we can also perform magnification and minification easily.

Rendering: the representation facilitates adaptive tessellation and hierarchical backface culling (Section 5.5).

2. PREVIOUS WORK

Subdivision surfaces: Subdivision schemes defining smooth surfaces have been introduced by Catmull and Clark [5], Doo and Sabin [13], and Loop [27]. More recently, these schemes have been extended to allow surfaces with sharp features [21] and fractionally sharp features [11]. In this paper we use the Loop subdivision scheme because it is designed for triangle meshes.

DeRose et al. [11] define scalar fields over subdivision surfaces using subdivision masks. Our scalar displacement field is defined similarly, but from a denser set of coefficients on a piecewise regular mesh (Figure 2).

Hoppe et al. [21] describe a method for approximating an original mesh with a much simpler subdivision surface. Unlike our conversion scheme of Section 4, their method does not consider whether the approximation residual is expressible as a scalar displacement map.

Displacement maps: The idea of displacing a surface by a function was introduced by Cook [9]. Displacement maps have become popular commercially as procedural *displacement shaders* in RenderMan [1]. The simplest displacement shaders interpolate values within an image, perhaps using standard bicubic filters. Though displacements may be in an arbitrary direction, they are almost always along the surface normal [1].

Typically, normals on the displaced surface are computed numerically using a dense tessellation. While simple, this approach requires adjacency information that may be unavailable or impractical with low-level APIs and in memory-constrained environments (e.g. game consoles). Strictly local evaluation requires that normals be computed from a continuous analytic surface representation. However, it is difficult to piece together multiple displacement maps while maintaining smoothness. One encounters the same vertex enclosure problem [32] as in the stitching of B-spline surfaces. While there are well-documented solutions to this problem, they require constructions with many more coefficients (9× in the best case), and may involve solving a global system of equations.

In contrast, our subdivision-based displacements are inherently smooth and have only quartic total degree (fewer DOF than bicubic). Since the displacement map uses the same parameterization as the domain surface, the surface representation is more compact and displaced surface normals may be computed more efficiently. Finally, unifying the representation around subdivision simplifies implementation and makes operations such as magnification more natural.

Krishnamurthy and Levoy [25] describe a scheme for approximating an arbitrary mesh using a B-spline patch network together with a vector-valued displacement map. In their scheme, the patch network is constructed manually by drawing patch boundaries on the mesh. The recent work on surface pasting by Chan et al. [7] and Mann and Yeung [29] uses the similar idea of adding a vector-valued displacement map to a spline surface.

Gumhold and Hüttner [19] describe a hardware architecture for rendering scalar-valued displacement maps over planar triangles. To avoid cracks between adjacent triangles of a mesh, they interpolate the vertex normals across the triangle face, and use this interpolated normal to displace the surface. Their scheme permits adaptive tessellation in screen space. They discuss the importance of proper filtering when constructing mipmap levels in a displacement map. Unlike our representation, their domain surface is not smooth since it is a polyhedron. As shown in Section 5.3, animating a displaced surface using a polyhedral domain surface results in many surface artifacts.

Kobbelt et al. [23] use a similar framework to express the geometry of one mesh as a displacement from another mesh, for the purpose of multiresolution shape deformation.

Bump maps: Blinn [3] introduces the idea of perturbing the surface normal using a bump map. Peercy et al. [31] present recent work on efficient hardware implementation of bump maps. Cohen et al. [8] drastically simplify meshes by capturing detail in the related *normal maps*. Both Cabral et al. [4] and Apodaca and Gritz [1] discuss the close relationship of bump mapping and displacement mapping. They advocate combining them into a unified representation and resorting to true displacement mapping only when necessary.

Multiresolution subdivision: Lounsbery et al. [28] apply multiresolution analysis to arbitrary surfaces. Given a parameterization of the surface over a triangular domain, they compress this (vector-valued) parameterization using a wavelet basis, where the basis functions are defined using subdivision of the triangular domain. Zorin et al. [39] use a similar subdivision framework for multiresolution mesh editing. To make this multiresolution framework practical, several techniques have been developed for constructing a parameterization of an arbitrary surface over a triangular base domain. Eck et al. [14] use Voronoi/Delaunay diagrams and harmonic maps, while Lee et al. [26] track successive mappings during mesh simplification.

In contrast, displaced subdivision surfaces do not support an arbitrary parameterization of the surface, since the parameterization is given by that of a subdivision surface. The benefit is that we need only compress a *scalar*-valued function instead of *vector*-valued parameterization. In other words, we store only geometric detail, not a parameterization. The drawback is that the original surface must be expressible as an offset of a smooth domain surface. An extremely bad case would be a fractal "snowflake" surface, where the domain surface cannot be made much simpler than the original surface. Fortunately, fine detail in most practical surfaces is expressible as an offset surface.

Guskov et al. [20] represent a surface by successively applying a hierarchy of displacements to a mesh as it is subdivided. Their construction allows most of the vertices to be encoded using scalar displacements, but a small fraction of the vertices require vector displacements to prevent surface folding.

3. REPRESENTATION OVERVIEW

A displaced subdivision surface consists of a triangle control mesh and a piecewise regular mesh of scalar displacement coefficients (see Figure 2). The domain surface is generated from the control mesh using Loop subdivision. Likewise, the displacements applied to the domain surface are generated from the scalar displacement mesh using Loop subdivision.

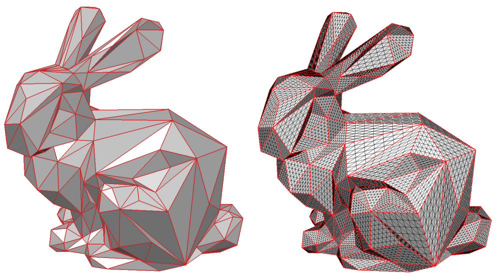


Figure 2: Control mesh (left) with its piecewise regular mesh of scalar displacement coefficients ($k = 3$).

Displacement map: The scalar displacement mesh is stored for each control mesh triangle as one half of the sample grid $(2^k+1)\times(2^k+1)$, where k depends on the sampling density required to achieve a desired level of accuracy or compression.

To define a *continuous* displacement function, these stored values are taken to be subdivision coefficients for the same (Loop) subdivision scheme that defines the domain surface. Thus, as the surface is magnified (i.e. subdivided beyond level k), both the domain surface geometry and the displacement field are subdivided using the same machinery. As a consequence, the displacement field is C^1 even at extraordinary vertices, and the displaced subdivision surface is C^1 everywhere *except* at extraordinary vertices. The handling of extraordinary vertices is discussed below.

For surface minification, we first compute the limit displacements for the subdivision coefficients at level k, and we then construct a mipmap pyramid with levels $\{0,\ldots,k\text{–}1\}$ by successive filtering of these limit values. We cover filtering possibilities in Section 4.5. As with ordinary texture maps, the content author may sometimes want more precise control of the filtered levels, so it may be useful to store the entire pyramid. (For our compression analysis in Section 5.1, we assume that the pyramid is built automatically.)

For many input meshes, it is inefficient to use the same value of k for all control mesh faces. For a given face, the choice of k may be guided by the number of original triangles associated it, which is easily estimated using MAPS [26]. Those regions with lower values of k are further subdivided *logically* to produce a mesh with uniform k.

Normal Calculation: We now derive the surface normal for a point $\vec{S}$ on the displaced subdivision surface. Let $\vec{S}$ be the displacement of the limit point $\vec{P}$ on the domain surface:

$$\vec{S} = \vec{P} + D\hat{n},$$

where D is the limit displacement and $\hat{n} = \vec{n}/\|\vec{n}\|$ is the unit normal on the domain surface. The normal $\vec{n}$ is obtained as $\vec{n} = \vec{P}_u \times \vec{P}_v$ where the tangent vectors $\vec{P}_u$ and $\vec{P}_v$ are computed using the first derivative masks in Figure 3.

The displaced subdivision surface normal at S is defined as $\vec{n}_s = \vec{S}_u \times \vec{S}_v$ where each tangent vector has the form

$$\vec{S}_u = \vec{P}_u + D_u\hat{n} + D\hat{n}_u\,.$$

If the displacements are relatively small, it is common to ignore the third term, which contains second-order derivatives [3].

However, if the surface is used as a modeling primitive, then the displacements may be quite large and the full expression must be evaluated. The difficult term $\hat{n}_u = \vec{n}_u/\|\vec{n}_u\|$ may be derived using the Weingarten equations [12]. Equivalently, it may be expressed as:

$$\hat{n}_u = \frac{\vec{n}_u - \hat{n}(\vec{n}_u\cdot\hat{n})}{\|\vec{n}\|} \quad \text{where} \quad \vec{n}_u = \vec{P}_{uu}\times\vec{P}_v + \vec{P}_u\times\vec{P}_{uv}\,.$$

At a regular (valence 6) vertex, the necessary partial derivatives are given by a simple set of masks (see Figure 3). At extraordinary vertices, the curvature of the domain surface vanishes and we omit the second-order term. In this case, the standard Loop tangent masks may be used to compute the first partial derivatives. Since there are few extraordinary vertices, this simplified normal calculation has not proven to be a problem.

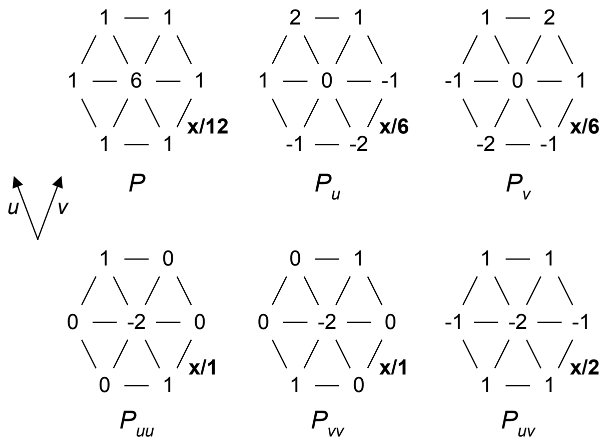


Figure 3: Loop masks for limit position P and first and second derivatives at a regular control vertex.

Bump map: The displacement map may also be used to generate a bump map during the rendering of coarser tessellations (see Figure 13). This improves rendering performance on graphics systems where geometry processing is a bottleneck. The construction of this bump map is presented in Section 5.4.

Other textures: The domain surface parameterization is used for storing the displacement map (which also serves to define a bump map). It is natural to re-use this same inherent parameterization to store additional appearance attributes for the surface, such as color. Section 4.4 describes how such attributes are re-sampled from the original surface.

Alternatively, one could define more traditional surface parameterizations by explicitly specifying (u, v) texture coordinates at the vertices of the control mesh, as in [11]. However, since the domain of a (u, v) parameterization is a planar region, this generally requires segmenting the surface into a set of charts.

4. CONVERSION PROCESS

To convert an arbitrary triangle mesh (Figure 5a) into a displaced subdivision surface (Figure 5b), our process performs the following steps:

- Obtain an initial control mesh (Figure 5c) by simplifying the original mesh. Simplification is done using a traditional sequence of edge collapse transformations, but with added heuristics to attempt to preserve a scalar offset function.
- Globally optimize the control mesh vertices (Figure 5d) such that the domain surface (Figure 5e) more accurately fits the original mesh.
- Sample the displacement map by shooting rays along the domain surface normals until they intersect the original mesh. At the ray intersection points, compute the signed displacement, and optionally sample other appearance attributes like surface color. (The black line segments visible in Figure 5f correspond to rays with positive displacements.)

4.1 Simplification to control mesh

We simplify the original mesh using a sequence of edge collapse transformations [22] prioritized according to the quadric error metric of Garland and Heckbert [16]. In order to produce a good domain surface, we restrict some of the candidate edge collapses.

The main objective is that the resulting domain surface should be able to express the original mesh using a scalar displacement map. Our approach is to ensure that the space of normals on the domain surface remains locally similar to the corresponding space of normals on the original mesh.

To maintain an efficient correspondence between the original mesh and the simplified mesh, we use the MAPS scheme [26] to track parameterizations of all original vertices on the mesh simplified so far. (When an edge is collapsed, the parametrizations of points in the neighborhood are updated using a local 1-to-1 map onto the resulting neighborhood.)

For each candidate edge collapse transformation, we examine the mesh neighborhood that would result. In Figure 4, the thickened 1-ring is the neighborhood of the unified vertex. For vertices on this ring, we compute the subdivision surface normals (using tangent masks that involve vertices in the 2-ring of the unified vertex). The highlighted points within the faces in the 1-ring represent original mesh vertices that are currently parameterized on the neighborhood using MAPS.

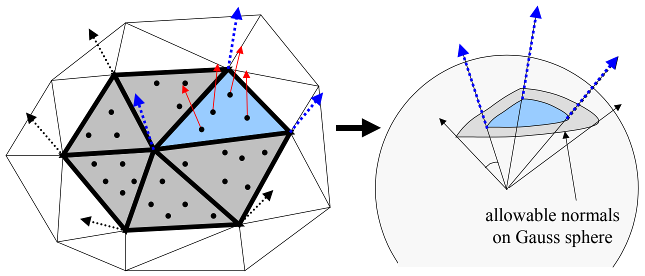


Figure 4: Neighborhood after candidate edge collapse and, for one face, the spherical triangle about its domain surface normals.

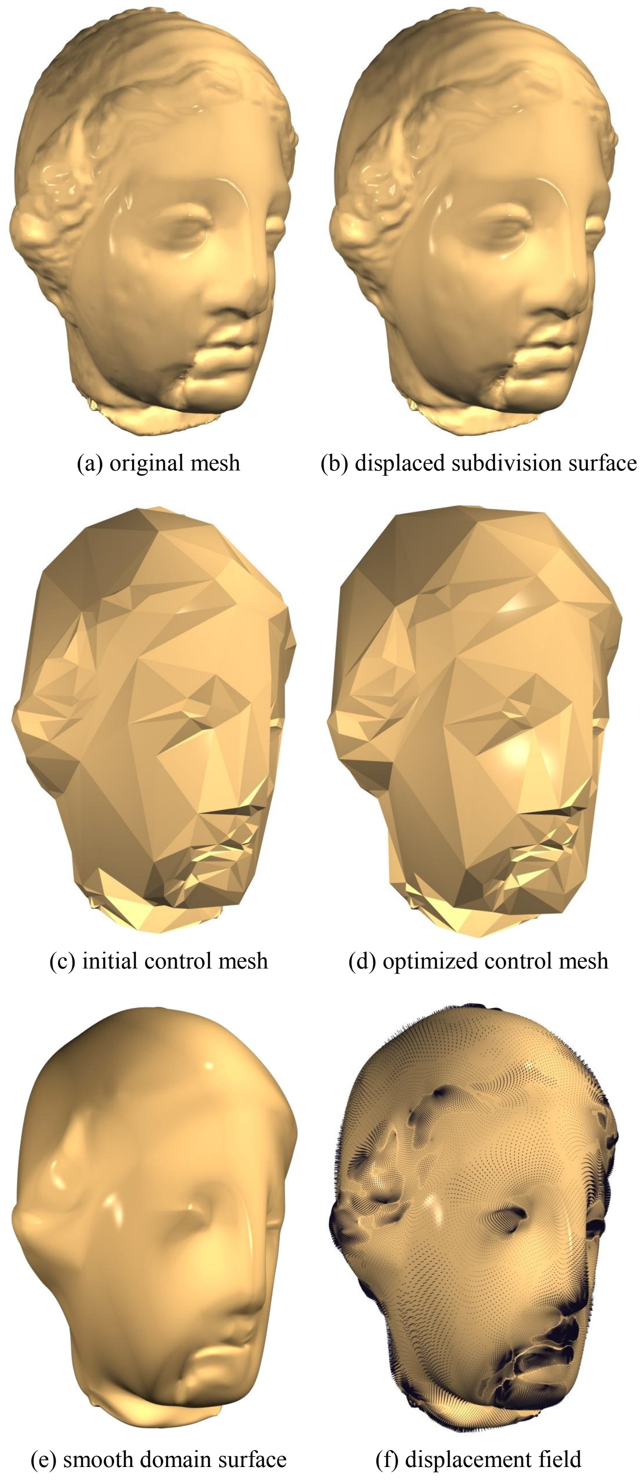

(a) original mesh
(b) displaced subdivision surface
(c) initial control mesh
(d) optimized control mesh
(e) smooth domain surface
(f) displacement field

Figure 5: Steps in the conversion process.

For each face in the 1-ring neighborhood, we gather the 3 subdivision surface normals at the vertices and form their spherical triangle on the Gauss sphere. Then, we test whether this spherical triangle encloses the normals of the original mesh vertices parameterized using MAPS. If this test fails on any face in the 1-ring, the edge collapse transformation is disallowed. To allow simplification to proceed further, we have found it useful to broaden each spherical triangle by pushing its three vertices an additional 45 degrees away from its inscribed center, as illustrated in Figure 4.

We observe that the domain surface sometimes has undesirable undulations when the control mesh has vertices of high valence. Therefore, during simplification we also disallow an edge collapse if the resulting unified vertex would have valence greater than 8.

4.2 Optimization of domain surface

Having formed the initial control mesh, we optimize the locations of its vertices such that the associated subdivision surface more accurately fits the original mesh. This step is performed using the method of Hoppe et al. [21]. We sample a dense set of points from the original mesh and minimize their squared distances to the subdivision surface. This nonlinear optimization problem is approximated by iteratively projecting the points onto the surface and solving for the most accurate surface while fixing those parameterizations. The result of this step is shown in Figure 5d-e.

Note that this geometric optimization modifies the control mesh and thus affects the space of normals over the domain surface. Although this invalidates the heuristic used to guide the simplification process, this has not been a problem in our experiments. A more robust solution would be to optimize the subdivision surface for each candidate edge collapse (as in [21]) prior to testing the neighborhood normals, but this would be much more costly.

4.3 Sampling of scalar displacement map

We apply k steps of Loop subdivision to the control mesh. At each of these subdivided vertices, we compute the limit position and normal of the domain surface. We seek to compute the signed distance from the limit point to the original surface along the normal (Figure 5f).

The directed line formed by the point and normal is intersected with the original surface, using a spatial hierarchy [17] for efficiency. We disregard any intersection point if the intersected surface is oriented in the wrong direction with respect to the directed line. If multiple intersection points remain, we pick the one closest to the domain surface. Figure 6 illustrates a possible failure case if the domain surface is too far from the original.

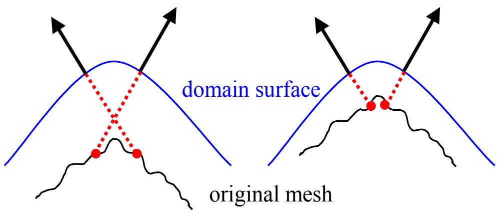


Figure 6: The displacement sampling may "fold over itself" if the domain surface is too distant from the original mesh.

Near surface boundaries, there is the problem that the domain surface may extend beyond the boundary of the original surface, in which case the ray does not intersect any useful part of the original surface. (We detect this using a maximum distance threshold based on the mesh size.) In this case, the surface should really be left undefined, i.e. trimmed to the detailed boundary of the original mesh. One approach would be to store a special illegal value into the displacement map. Instead, we find the closest original triangle to the subdivided vertex, and intersect the ray with the plane containing that triangle. Precise surface trimming can be achieved using an alpha mask in the surface color image, but we have not yet implemented this.

4.4 Resampling of appearance attributes

Besides sampling the scalar displacement function, we also sample other appearance attributes such as diffuse color. These attributes are stored, filtered, and compressed just like the scalar displacements. An example is shown in Figure 11.

4.5 Filtering of displacement map

Since our displacement field has the same structure as the domain surface, we can apply the same subdivision mask for magnification. This is particular useful when we try to zoom in a tiny region on our displaced subdivision surface. For sampling the displacements at minified levels of the displacement pyramid, we compute the samples at any level $l < k$ by filtering the limit displacements of level $l + 1$. We considered several filtering operations and opted for the non-shrinking filter of Taubin [35].

Because the displacement magnitudes are kept small, their filtering is not extremely sensitive. In many rendering situations much of the visual detail is provided by bump mapping. As has been discussed elsewhere [2], careful filtering of bump maps is both important and difficult.

4.6 Conversion results

The following table shows execution times for the various steps of the conversion process. These times are obtained on a Pentium III 550 MHz PC.

Model	armadillo	venus	bunny	dinosaur
Conversion Statistics				
Original mesh #F	210,944	100,000	69,451	342,138
Control mesh #F	1,306	748	526	1,564
Maximum level k	4	4	4	4
Execution Times (minutes)				
Simplification	61	28	19	115
Domain surface optimiz.	25	11	11	43
Displacement sampling	2	2	1	5
Total	88	41	31	163

5. BENEFITS

5.1 Compression

Mesh compression has recently been an active area of research. Several clever schemes have been developed to concisely encode the combinatorial structure of the mesh connectivity, in as few as 1-2 bits per face (e.g. [18] [35]). As a result, the major portion of a compressed mesh goes to storing the mesh geometry. Vertex positions are typically compressed using quantization, local prediction, and variable-length delta encoding. Geometry can also be compressed within a multiresolution subdivision framework as a set of wavelet coefficients [28]. To our knowledge, all previous compression schemes for arbitrary surfaces treat geometry as a vector-valued function.

In contrast, displaced subdivision surfaces allow fine geometric detail to be compressed as a scalar-valued function. Moreover, the domain surface is constructed to be close to the original surface, so the magnitude of the displacements tends to be small.

To exploit spatial coherence in the scalar displacement map, we use linear prediction at each level of the displacement pyramid, and encode the difference between the predicted and actual values. For each level, we treat the difference coefficients over all faces as a subband. For each subband, we use the embedded quantizer and embedded entropy coder described in Taubman and Zakhor [37]. The subbands are merged using the bit allocation algorithm described by Shoham and Gersho [34], which is based on integer programming.

An alternative would be to use the compression scheme of Kolarov and Lynch [24], which is a generalization of the wavelet compression method in [33].

Figure 10 and Table 1 show results of our compression experiments. We compare storage costs for simplified triangle meshes and displaced subdivision surfaces, such that both compressed representations have the same approximation accuracy with respect to the original reference model. This accuracy is measured as L^2 geometric distance between the surfaces, computed using dense point sampling [16]. The simplified meshes are obtained using the scheme of Garland and Heckbert [16]. For mesh compression, we use the *VRML compressed binary format* inspired by the work of Taubin and Rossignac [36]. We vary the quantization level for the vertex coordinates to obtain different compressed meshes, and then adjust our displacement map compression parameters to obtain a displaced surface with matching L^2 geometric error.

For simplicity, we always compress the control meshes losslessly in the experiments (i.e. with 23-bits/coordinate quantization). Our compression results would likely be improved further by adapting the quantization of the control mesh as well. However, this would modify the domain surface geometry, and would therefore require re-computing the displacement field. Also, severe quantization of the control mesh would result in larger displacement magnitudes.

Table 1 shows that displaced subdivision surfaces consistently achieve better compression rates than mesh compression, even when the mesh is carefully simplified from detailed geometry.

5.2 Editing

The fine detail in the scalar displacement mesh can be edited conveniently, as shown in the example of Figure 7.

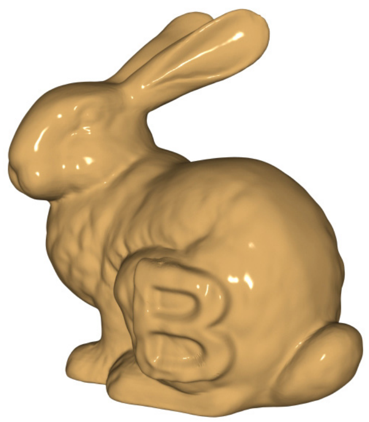


Figure 7: In this simple editing example, the embossing effect is produced by enhancing the scalar displacements according to a texture image of the character ‘B’ projected onto the displaced surface.

5.3 Animation

Displaced subdivision surfaces are a convenient representation for animation. Kinematic and dynamics computation are vastly more efficient when operating on the control mesh rather than the huge detailed mesh.

Because the domain surface is smooth, the surface detail deforms naturally without artifacts. Figure 8 shows that in contrast, the use of a polyhedron as a domain surface results in creases and folds even with a small deformation of a simple surface.

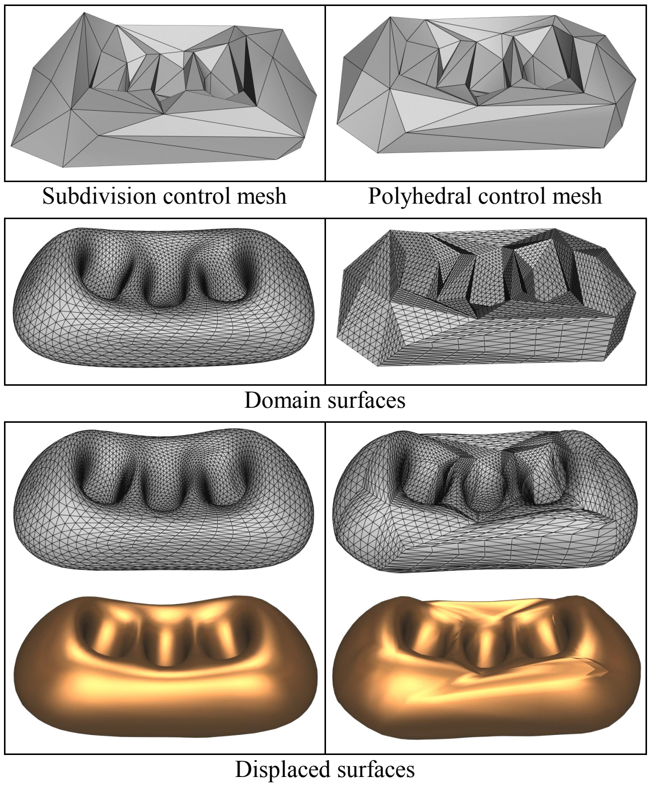


Figure 8: Comparison showing the importance of using a smooth domain surface when deforming the control mesh. The domain surface is a subdivision surface on the left, and a polyhedron on the right.

Figure 12 shows two frames from the animation of a more complicated surface. For that example, we used 3D Studio MAX to construct a skeleton of bones inside the control mesh, and manipulated the skeleton to deform this mesh. (The complete animation is on the accompanying video.)

Another application of our representation is the fitting of 3D head scans [30]. For this application, it is desirable to re-use a common control mesh structure so that deformations can be conveniently transferred from one face model to another.

5.4 Scalability

Depending on the level-of-detail requirements and hardware capabilities, the scalar displacement function can either be:

- rendered as **explicit geometry**: Since it is a continuous representation, the tessellation is not limited to the resolution of the displacement mesh. A scheme for adaptive tessellation is presented in Section 5.5.
- converted to a **bump map**: This improves rendering performance on graphics systems where geometry processing is a bottleneck. As described in [31], the calculation necessary for tangent-space bump mapping involves computing the displaced subdivision surface normal relative to a coordinate frame on the domain surface. A convenient coordinate frame is formed by the domain surface unit normal $\hat{n}$ and a tangent vector such as $\vec{P}_u$. Given these vectors, the coordinate frame is:

$$\{\hat{b},\hat{t},\hat{n}\} \text{ where } \hat{t}=\vec{P}_u/\|\vec{P}_u\| \text{ and } \hat{b}=\hat{n}\times\hat{t}\,.$$

Finally, the normal $\hat{n}_s$ to the displaced subdivision surface relative to this tangent space is computed using the transform:

$$\hat{n}_{\text{tangent space}} = \left(\hat{b}\ \hat{t}\ \hat{n}\right)^T \cdot \hat{n}_s\,.$$

The computations of $\hat{n}$, $\vec{P}_u$, and $\hat{n}_s$ are described in Section 3. Note that we use the precise analytic normal in the bump map calculation. As an example, Figure 13 shows renderings of the same model with different boundaries between explicit geometry and bump mapping. In the leftmost image, the displacements are all converted into geometry, and bump-mapping is turned off. In the rightmost image, the domain surface is sampled only at the control mesh vertices, but the entire displacement pyramid is converted into a bump map.

5.5 Rendering

Adaptive tessellation: In order to perform adaptive tessellation, we need to compute the approximation error of any intermediate tessellation level from the finely subdivided surface. This approximation error is obtained by computing the maximum distance between the dyadic points on the planar intermediate level and their corresponding surface points at the finest level (see Figure 9). Note that this error measurement corresponds to parametric error and is stricter than geometric error. Bounding parametric error is useful for preventing appearance fields (e.g. bump map, color map) from sliding over the rendered surface [8]. These precomputed error measurements are stored in a quadtree data structure. At runtime, adaptive tessellation prunes off the entire subtree beneath a node if its error measurement satisfies given level-of-detail parameters. By default, the displacements applied to the vertices of a face are taken from the corresponding level of the displacement pyramid.

Note that the pruning will make adjacent subtrees meet at different levels. To avoid cracks, if a vertex is shared among different levels, we choose the finest one from the pyramid. Also, we perform a retriangulation of the coarser face so that it conforms to the vertices along the common edges. Figure 14 shows some examples of adaptive tessellation.

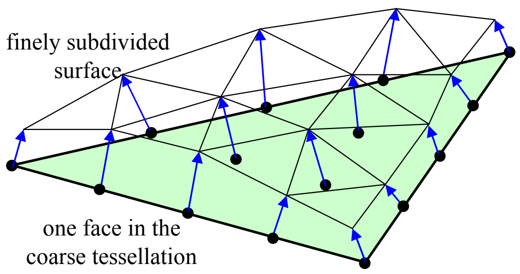

finely subdivided surface
one face in the coarse tessellation

Figure 9: Error computation for adaptive tessellation.

Backface patch culling: To improve rendering performance, we avoid rendering regions of the displaced subdivision surface that are entirely facing away from the viewpoint. We achieve this using the *normal masks* technique of Zhang and Hoff [38].

On the finely subdivided version of the domain surface, we compute the vertex normals of the displaced surface as described in Section 3. We convert these into a normal mask for each subdivided face. During a bottom-up traversal of the subdivision hierarchy, we propagate these masks to the parents using the logical *or* operation.

Given the view parameters, we then construct a viewing mask as in [38], and take its logical *and* with the stored masks in the hierarchy. Generally, we cull away 1/3 to 1/4 of the total number of triangles, thereby speeding up rendering time by 20% to 30%.

6. DISCUSSION

Remeshing creases: As in other remeshing methods [14] [26], the presence of creases in the original surface presents challenges to our conversion process. Lee et al. [26] demonstrate that the key is to associate such creases with edges in the control mesh. Our simplification process also achieves this since mesh simplification naturally preserves sharp features.

However, displaced subdivision surfaces have the further constraint that the displacements are strictly scalar. Therefore, the edges of the control mesh, when subdivided and displaced, do not generally follow original surface creases exactly. (A similar problem also arises at surface boundaries.) This problem can be resolved if displacements were instead vector-based, but then the representation would lose its simplicity and many of its benefits (compactness, ease of scalability, etc.).

Scaling of displacements: Currently, scalar displacements are simply multiplied by unit normals on the domain surface. With a "rubbery" surface, the displaced subdivision surface behaves as one would expect, since detail tends to smooth as the surface stretches. However, greater control over the magnitude of displacement is desirable in many situations. A simple extension of the current representation is to provide scale and bias factors (s,b) at control mesh vertices. These added controls enhance the basic displacement formula:

$$\vec{S}=\vec{P}+(sD+b)\hat{n}\,.$$

Exploring such scaling controls is an interesting area of future work.

7. SUMMARY AND FUTURE WORK

Nearly all geometric representations capture geometric detail as a vector-valued function. We have shown that an arbitrary surface can be approximated by a displaced subdivision surface, in which geometric detail is encoded as a scalar-valued function over a domain surface. Our representation defines both the domain surface and the displacement function using a unified subdivision framework. This synergy allows simple and efficient evaluation of analytic surface properties.

We demonstrated that the representation offers significant savings in storage compared to traditional mesh compression schemes. It is also convenient for animation, editing, and runtime level-of-detail control.

Areas for future work include: a more rigorous scheme for constructing the domain surface, improved filtering of bump maps, hardware rendering, error measures for view-dependent adaptive tessellation, and use of detail textures for displacements.

ACKNOWLEDGEMENTS

Our thanks to Gene Sexton for his help in scanning the dinosaur.

REFERENCES

[1] Apodaca, A. and Gritz, L. Advanced RenderMan – Creating CGI for Motion Pictures, Morgan Kaufmann, San Francisco, CA, 1999.

[2] Becker, B. and Max, N. Smooth transitions between bump rendering algorithms. Proceedings of SIGGRAPH 93, Computer Graphics, Annual Conference Series, pp. 183-190.

[3] Blinn, J. F. Simulation of wrinkled surfaces. Proceedings of SIGGRAPH 78, Computer Graphics, pp. 286-292.

[4] Cabral, B., Max, N. and Springmeyer, R. Bidirectional reflection functions from surface bump maps. Proceedings of SIGGRAPH 87, Computer Graphics, Annual Conference Series, pp.273-281.

[5] Catmull, E., and Clark, J. Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design 10, pp. 350-355 (1978).

[6] Certain, A., Popovic, J., DeRose, T., Duchamp, T., Salesin, D. and Stuetzle, W. Interactive multiresolution surface viewing. Proceedings of SIGGRAPH 96, Computer Graphics, Annual Conference Series, pp. 91-98.

[7] Chan, K., Mann, S., and Bartels, R. World space surface pasting. Graphics Interface '97, pp. 146-154.

[8] Cohen, J., Olano, M. and Manocha, D. Appearance preserving Simplification. Proceedings of SIGGRAPH 98, Computer Graphics, Annual Conference Series, pp. 115-122.

[9] Cook, R. Shade trees. Computer Graphics (Proceedings of SIGGRAPH 84), 18(3), pp. 223-231.

[10] Deering, M. Geometry compression. Proceedings of SIGGRAPH 95, Computer Graphics, Annual Conference Series, pp. 13-20.

[11] DeRose, T., Kass, M., and Truong, T. Subdivision surfaces in character animation. Proceedings of SIGGRAPH 98, Computer Graphics, Annual Conference Series, pp. 85-94.

[12] Do Carmo, M. P. Differential Geometry of Curves and Surfaces. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

[13] Doo, D., and Sabin, M. Behavior of recursive division surfaces near extraordinary points. Computer Aided Design 10, pp. 356-360 (1978).

[14] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. Multiresolution analysis of arbitrary meshes. Proceedings of SIGGRAPH 95, Computer Graphics, Annual Conference Series, pp. 173-182.

[15] Forsey, D., and Bartels, R. Surface fitting with hierarchical splines. ACM Transactions on Graphics, 14(2), pp. 134-161 (April 1995).

[16] Garland, M., and Heckbert, P. Surface simplification using quadric error metrics. Proceedings of SIGGRAPH 97, Computer Graphics, Annual Conference Series, pp. 209-216.

[17] Gottschalk, S., Lin, M., and Manocha, D. OBB-tree: a hierarchical structure for rapid interference detection. Proceedings of SIGGRAPH 96, Computer Graphics, Annual Conference Series, pp. 171-180.

[18] Gumhold, S., and Straßer, W. Real time compression of triangle mesh connectivity. Proceedings of SIGGRAPH 98, Computer Graphics, Annual Conference Series, pp. 133-140.

[19] Gumhold, S., and Hüttner, T. Multiresolution rendering with displacement mapping. SIGGRAPH workshop on Graphics hardware, Aug 8-9, 1999.

[20] Guskov, I., Vidimce, K., Sweldens, W., and Schröder, P. Normal meshes. Proceedings of SIGGRAPH 2000, Computer Graphics, Annual Conference Series.

[21] Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., and Stuetzle, W. Piecewise smooth surface reconstruction. Proceedings of SIGGRAPH 94, Computer Graphics, Annual Conference Series, pp. 295-302.

[22] Hoppe, H. Progressive meshes. Proceedings of SIGGRAPH 96, Computer Graphics, Annual Conference Series, pp. 99-108.

[23] Kobbelt, L., Bareuther, T., and Seidel, H. P. Multi-resolution shape deformations for meshes with dynamic vertex connectivity. Proceedings of EUROGRAPHICS 2000, to appear.

[24] Kolarov, K. and Lynch, W. Compression of functions defined on surfaces of 3D objects. In J. Storer and M. Cohn, editors, Proc. of Data Compression Conference, IEEE, pp. 281-291, 1997.

[25] Krishnamurthy, V., and Levoy, M. Fitting smooth surfaces to dense polygon meshes. Proceedings of SIGGRAPH 96, Computer Graphics, Annual Conference Series, pp. 313-324.

[26] Lee, A., Sweldens, W., Schröder, P., Cowsar, L., and Dobkin, D. MAPS: Multiresolution adaptive parameterization of surfaces. Proceedings of SIGGRAPH 98, Computer Graphics, Annual Conference Series, pp. 95-104.

[27] Loop, C. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[28] Lounsbery, M., DeRose, T., and Warren, J. Multiresolution analysis for surfaces of arbitrary topological type. ACM Transactions on Graphics, 16(1), pp. 34-73 (January 1997).

[29] Mann, S. and Yeung, T. Cylindrical surface pasting. Technical Report, Computer Science Dept., University of Waterloo (June 1999).

[30] Marschner, S., Guenter, B., and Raghupathy, S. Modeling and rendering for realistic facial animation. Submitted for publication.

[31] Peercy, M., Airey, J. and Cabral, B. Efficient bump mapping hardware. Proceedings of SIGGRAPH 97, Computer Graphics, Annual Conference Series, pp. 303-306.

[32] Peters, J. Local smooth surface interpolation: a classification. Computer Aided Geometric Design, 7(1990), pp. 191-195.

[33] Schröder, P., and Sweldens, W. Spherical wavelets: efficiently representing functions on the sphere. Proceedings of SIGGRAPH 95, Computer Graphics, Annual Conference Series, pp. 161-172.

[34] Shoham, Y. and Gersho, A. Efficient bit allocation for an arbitrary set of quantizers. IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 36, No. 9, pp. 1445-1453, Sept 1988.

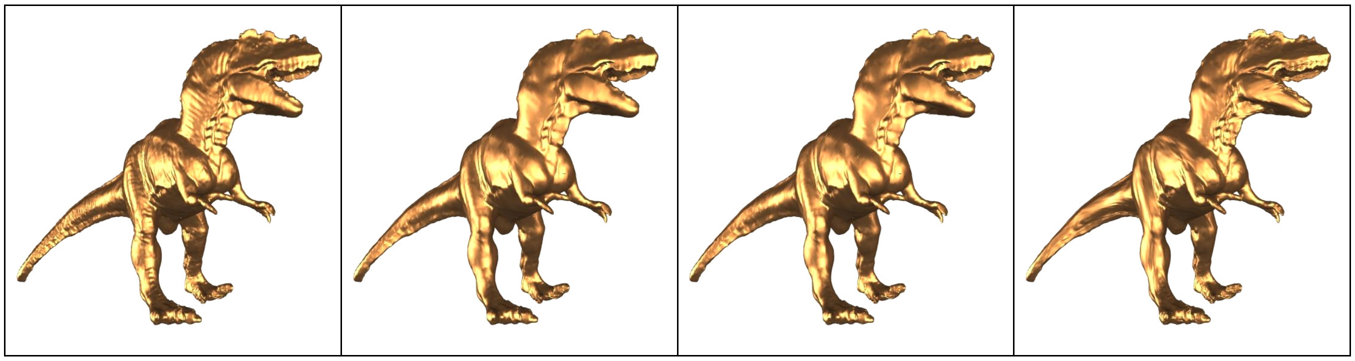
[35] Taubin, G. A signal processing approach to fair surface design. Proceedings of SIGGRAPH 95, Computer Graphics, Annual Conference Series, pp. 351-358.

[36] Taubin, G. and Rossignac, J. Geometric compression through topological surgery. ACM Transactions on Graphics, 17(2), pp. 84-115 (April 1998).

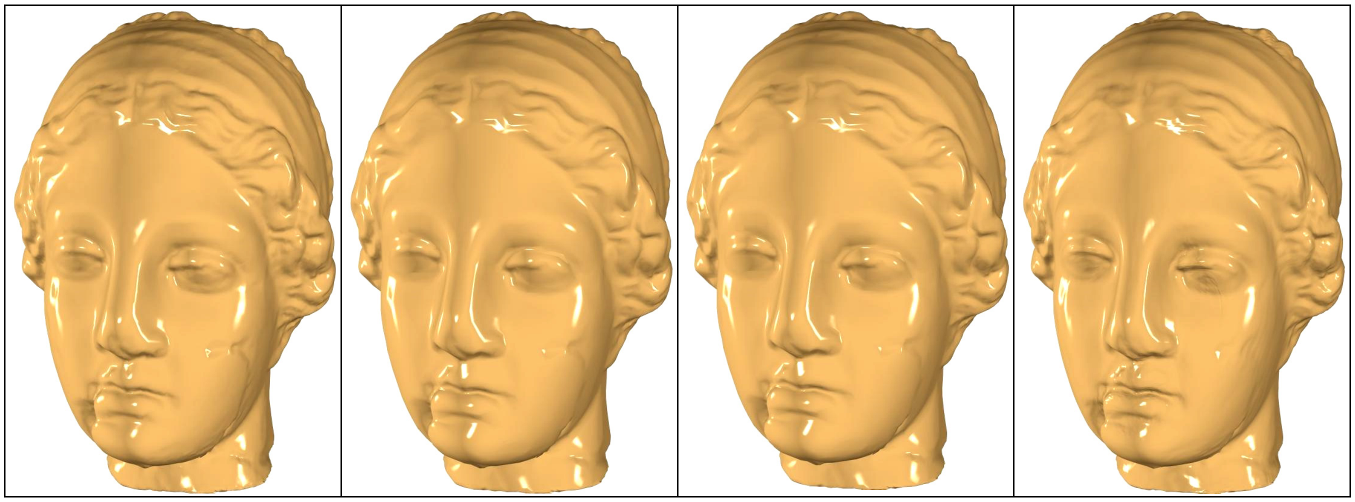
[37] Taubman, D. and Zakhor, A. Multirate 3-D subband coding of video. IEEE Transactions on Image Processing, Vol. 3, No. 5, Sept, 1994.

[38] Zhang, H., and Hoff, K. Fast backface culling using normal masks. Symposium on Interactive 3D Graphics, pp. 103-106, 1997.

[39] Zorin, D., Schröder, P., and Sweldens, W. Interactive multiresolution mesh editing. Proceedings of SIGGRAPH 97, Computer Graphics, Annual Conference Series, pp. 259-268.



Original mesh	Simplified mesh	Compressed simplified mesh	Displaced subdivision surface
342,138 faces; 1011 KB	50,000 faces; 169 KB	(12-bits/coord.); 68 KB	1564 control mesh faces; 18 KB



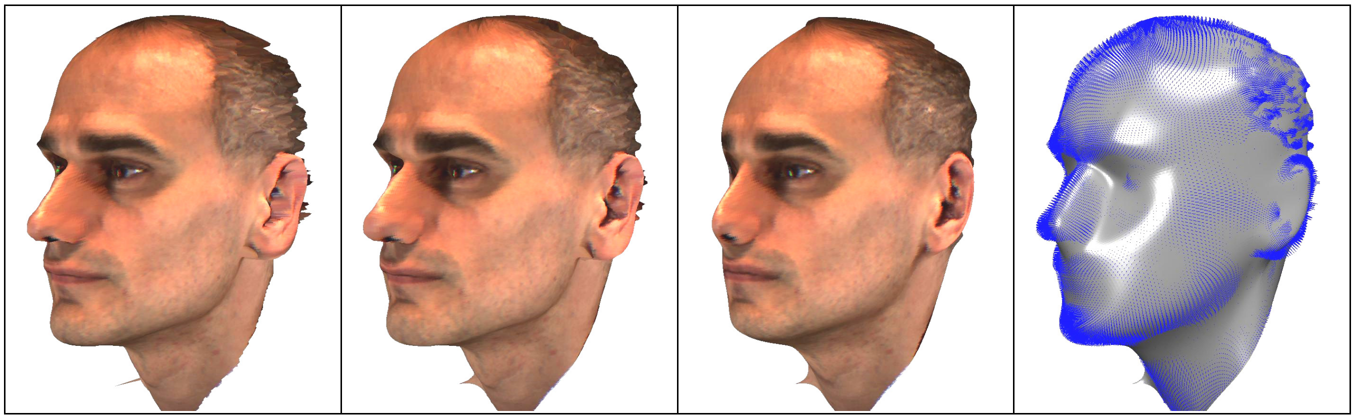
Original mesh	Simplified mesh	Compressed simplified mesh	Displaced subdivision surface
100,000 faces; 346 KB	20,000 faces; 75 KB	(12-bits/coord.); 33 KB	748 control mesh faces; 16 KB

Figure 10: Compression results. Each example shows the approximation of a dense original mesh using a simplified mesh and a displaced subdivision surface, such that both have comparable L^2 approximation error (expressed as a percentage of object bounding box).

Dinosaur	Original mesh		Compressed simplified mesh		Displaced subdivision surface (k=4)		
	#V=171,074 #F=342,138		#V=25,005 #F=50,000		#V^0=787 #F^0=1564 ≡ 6.5KB		
Quantization (bits/coord.)	L^2 error	Size (KB)	L^2 error	Size (KB)	L^2 error	Size (KB)	Size ratio
23	0.002%	1011	0.024%	169	0.025%	22	7.7
12	0.014%	322	0.028%	68	0.028%	18	3.8
10	0.053%	217	0.059%	50	0.058%	10	5.0
8	0.197%	169	0.21%	35	0.153%	7	5.0

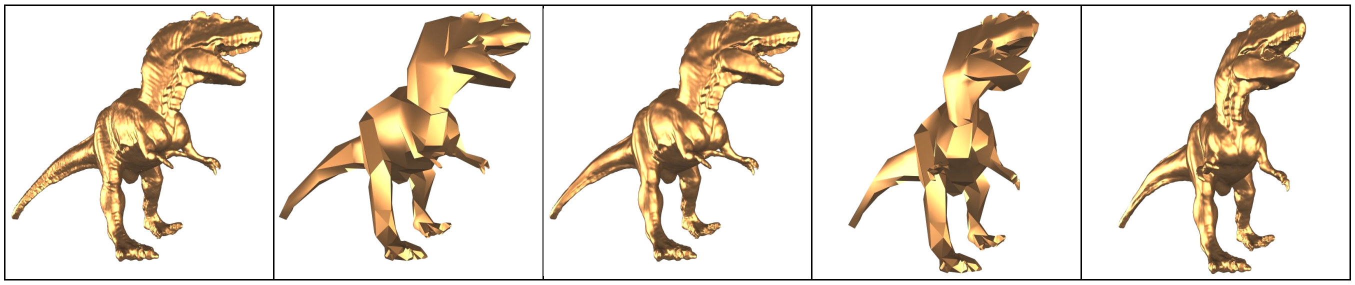
Venus	Original mesh		Compressed simplified mesh		Displaced subdivision surface (k=4)		
	#V=50,002 #F=100,000		#V=10,002 #F=20,000		#V^0=376 #F^0=748 ≡ 3.4KB		
Quantization (bits/coord.)	L^2 error	Size (KB)	L^2 error	Size (KB)	L^2 error	Size (KB)	Size ratio
23	0.001%	346	0.027%	75	0.027%	17	4.4
12	0.014%	140	0.030%	33	0.031%	16	2.0
10	0.054%	102	0.059%	26	0.053%	8	3.2
8	0.207%	69	0.210%	18	0.149%	4	4.5

Table 1: Quantitative compression results for the two examples in Figure 10. Numbers in red refer to figures above.



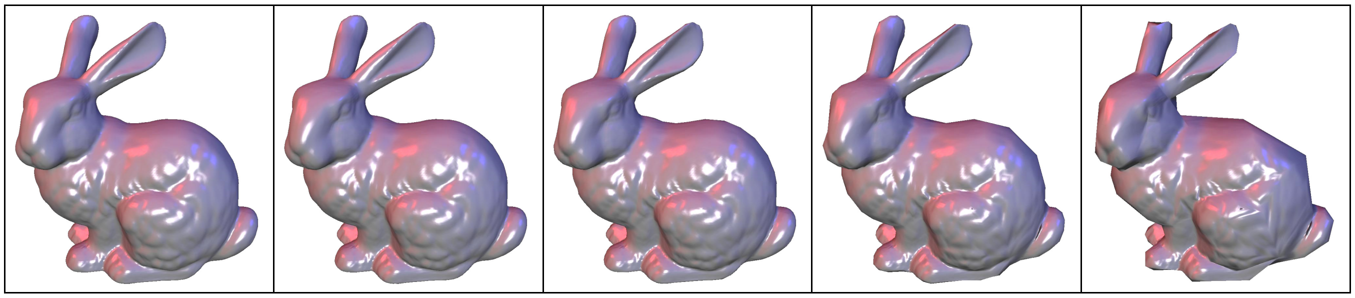
Original colored mesh | Displaced subdivision surface | Domain surface | Displacement samples ($k = 4$)

Figure 11: Example of a displaced subdivision surface with resampled color.



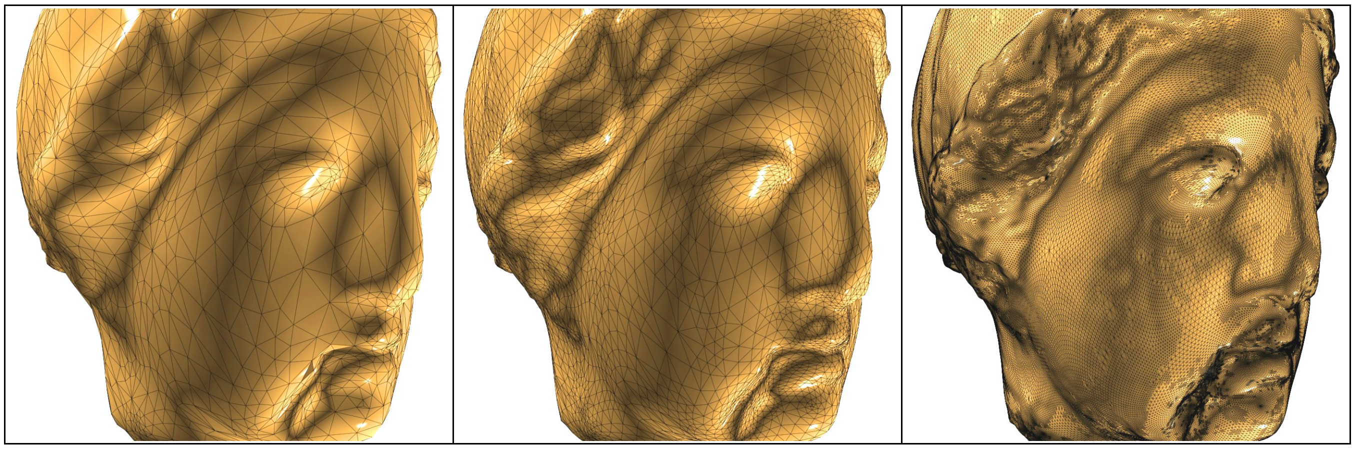
Original mesh | Control mesh | Displaced subdiv. surface | Modified control mesh | Resulting deformed surface

Figure 12: The control mesh makes a convenient armature for animating the displaced subdivision surface.



Level 4 (134,656 faces) | Level 3 (33,664 faces) | Level 2 (8,416 faces) | Level 1 (2,104 faces) | Level 0 (526 faces)

Figure 13: Replacement of scalar displacements by bump-mapping at different levels.



Threshold = 1.87% diameter
12,950 triangles; L^2 error = 0.104%

Threshold = 0.76% diameter
88,352 triangles; L^2 error = 0.035%

Threshold = 0.39% diameter
258,720 triangles; L^2 error = 0.016%

Figure 14: Example of adaptive tessellation, using the view-independent criterion of comparing residual error with a global threshold.