Optimization Methods for Scattered Data Approximation with Subdivision Surfaces

Martin Marinov and Leif Kobbelt
Computer Graphics Group, RWTH Aachen, Germany [1]

Abstract

We present a method for scattered data approximation with subdivision surfaces which actually uses the true representation of the limit surface as a linear combination of smooth basis functions associated with the control vertices. A robust and fast algorithm for exact closest point search on Loop surfaces which combines Newton iteration and non-linear minimization is used for parameterizing the samples. Based on this we perform unconditionally convergent parameter correction to optimize the approximation with respect to the L^2 metric and thus we make a well-established scattered data fitting technique which has been available before only for B-spline surfaces, applicable to subdivision surfaces. We also adapt the recently discovered local second order squared distance function approximant to the parameter correction setup. Further we exploit the fact that the control mesh of a subdivision surface can have arbitrary connectivity to reduce the L^∞ error up to a certain user-defined tolerance by adaptively restructuring the control mesh. Combining the presented algorithms we describe a complete procedure which is able to produce high-quality approximations of complex, detailed models.

Key words: Subdivision surfaces, Approximation, Fitting, Reverse engineering, Shape reconstruction

1 Introduction

Scattered data approximation methods are a key technology for shape reconstruction and reverse engineering from measured geometry data. In a typical application scenario, raw data is generated, e.g., by some 3D scanning device and fitting a globally smooth surface to the set of sample points converts this data into a geometric

[1] Email: {marinov | kobbelt}@cs.rwth-aachen.de
Phone: +49 241 8021815, Fax: +49 241 8022899

Figure 1. Original Iphigenie model (left): scanned data 351750 points, Loop surface approximation (center): maximum deviation is 0.05% of the bounding box diagonal, control mesh defining the surface (right): 15347 control points.

representation of the original object that enables sophisticated downstream applications like, e.g., free-form shape editing. Most of the work in this area has been done based on classical tensor-product spline surfaces but with the availability of more flexible subdivision surfaces many ideas have been extended to this generalized setting during the last years. Instead of being constrained to rectangular patches, subdivision surfaces can represent globally smooth surfaces of arbitrary (manifold) topology by allowing for arbitrary irregular control meshes. We are using Loop subdivision surfaces [1] in this paper, but the basic concepts could be transferred to other types of subdivision, e.g., Catmull-Clark subdivision surfaces [2].

The majority of the well-established scattered data approximation techniques focuses on the minimization of some form of the L^2 error. The main reason for this is that least squares problems are easy and efficiently handled by solving a simple linear system. However, from the application point of view, L^∞ type errors are much more relevant since the user usually prescribes some maximum tolerance δ by which the fitted surface is allowed to deviate from the given data. Because the L^2 metric is computed by some integral over the surface, one often wastes effort (and degrees of freedom) when globally improving the approximation even if the

maximum tolerance is only violated in some local region.

The motivation for the work presented in this paper is the observation that due to the flexibility of subdivision surfaces with respect to their control mesh structure, we can apply and iterate many different operations to progressively improve the approximation of the given data. If we just update the positions of the control vertices we can do least squares fitting just like for spline surfaces. However, in addition we can change the structure of the control mesh by locally inserting or removing control vertices. This allows us to efficiently reduce the L^∞ error by adaptively introducing new degrees of freedom (i.e. control vertices) in regions where the maximum tolerance is exceeded and by removing degrees of freedom in regions where the surface fitting problem is under-determined due to sparse sample data.

The specific contributions of this paper are that we present a complete scattered data fitting method for subdivision surfaces that uses the true subdivision basis functions instead of some piecewise linear approximation. This is made possible by using Stam's exact evaluation procedure [3,4] to set up the least squares system. Since we can also evaluate partial derivatives of a subdivision surface exactly, we propose a robust algorithm which finds the parameter value of the closest point on the approximating surface to a given sample. Thus we generalize the technique of parameter correction [5] from spline surfaces to subdivision surfaces and perform unconditionally convergent optimization of the approximation with respect to the L^2 error (Section 2).

To reduce the L^∞ error below a user prescribed error tolerance δ, we present an iterative procedure in Section 3 that adaptively refines the control mesh according to the local approximation error or coarsens it if the local sample density is insufficient. In combination with mesh connectivity regularization we are able to produce high quality approximations without having to add a fairness functional. Our technique is progressive and scalable in the sense that we can get a coarse fit after just a few seconds while we can further improve the approximation quality by letting the algorithm perform some more iterations.

1.1 Related work

The amount of work that has been done in the area of surface approximation is immense and a complete review is beyond the scope of this paper. We refer to [6] as a standard reference and to [7] for some more recent advances. Traditionally, tensor-product spline surfaces have been used for this task, but when it comes to the approximation of complex geometric objects, their rigid regular structure makes it necessary to fit several patches to parts of the data and then to stitch them together in a geometrically smooth fashion [8].

Another problem is that the regular structure of tensor-product patches prohibits the

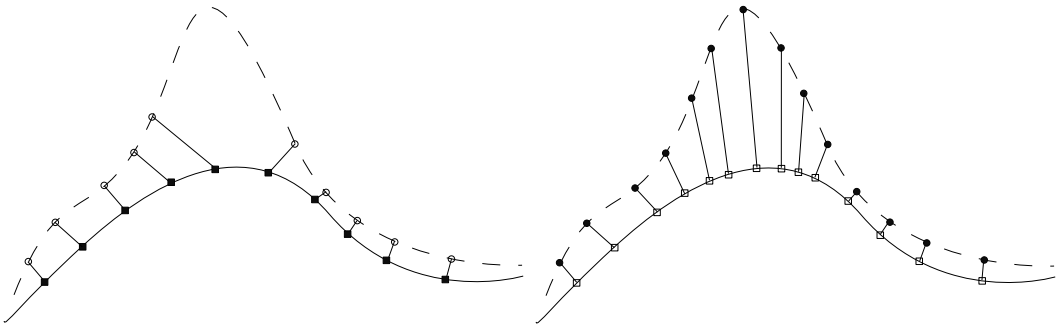

Figure 2. Uniform parametrization (left) vs. parameter correction (right). The dots represent the approximated samples and the position of the squares indicate the corresponding parameter values. In the uniform setting we have to find the closest data sample for uniformly distributed "sensor" points on the approximating surface. In the parameter correction setting we find the closest point on the approximating surface for each sample leading to a more reliable geometric distance measure.

flexible adaption of the control mesh to the local shape complexity or sample density. As a consequence penalty functionals (a.k.a. fairing functionals) usually have to be added to the approximation problem in order to stabilize it [5,8]. All these difficulties compromise the flexibility and approximation power of spline surfaces for general approximation problems.

Subdivision surfaces are globally smooth (mostly even piecewise polynomial) surfaces that do not suffer from these limitations. Complex shapes can be represented with one control mesh and local adaption of the control vertex density is straightforward [9]. This is why several papers [10,11,12,13,14,15] have addressed the scattered data fitting problem by using subdivision surface representations. However, since subdivision surfaces have no obvious explicit parametrization, modifications and simplifications of the general setting have been used for the sake of efficiency.

One issue is that, assuming the canonical parametrization, subdivision surfaces are much easier to evaluate at dyadic barycentric parameter values than at arbitrary parameter values. This is why special uniform parameterizations of the given sample data have been preferred by many authors [12,16,15,17,18,19]. Although this leads to well conditioned least squares systems and extremely simple quasi-interpolation operators [15,17], the major drawback of using uniform parameterizations is that the evaluated approximation errors differ significantly from the actual geometric deviation. In [17], Litke et al. use uniform parameterization in combination with re-sampling the target surface. While this leads to a geometrically meaningful error metric, it may affect the local sample density potentially leading to under-sampling in regions where the surface normals of the fitting surface and the target surface strongly differ or where the fitting surface has high curvature (Fig. 2).

Parameter correction is a technique that does exactly the opposite. For each given sample point, the closest point on the approximating surface is found which implies an obvious geometric interpretation of the approximation error (Fig. 2). However,

whenever parameter correction has been used for subdivision surfaces in the literature, the correction has been computed with respect to a piecewise linear approximation instead of the true limit surface [11]. In practice this often leads to numerical instabilities - mostly due to the fact that the piecewise linear approximation does not have a continuous surface normal field.

From the conceptual point of view our work is closest to [11]. In this paper Hoppe et al. describe a procedure that does least squares fitting of a subdivision surface to scattered data, however the parametrization is computed and evaluated only on a piecewise linear approximation of the approximating surface. This procedure is interleaved with a "random descent" mesh optimization scheme, which iteratively attempts to reduce the L^2 error. Minimization of the L^∞ error is not discussed.

A quite different approach to subdivision surface fitting is described in [20,18,16] and [15,17]. Here the so-called *multiresolution* subdivision surfaces [21] are used which assign an additional displacement vector to every control vertex of an adaptively refined control mesh. While this representation provides a natural hierarchy that distinguishes different levels of detail, the mathematical representations becomes extremely redundant as can be seen from the number of detail coefficients that are necessary to closely approximate complicated objects [15].

In a recent work, Cheng et al. [22] present a fitting technique for plain subdivision surfaces, based on the second order approximant of the surface squared distance function presented in [23]. We discuss the use of this new distance metric within our approximation framework in Section 2.3. A comparison between the results of [22] and our method is presented in Section 5.

1.2 Notation

With $S = \{s_1 \cdots s_M\}$ we denote the set of given data samples and with D the subdivision surface which is fitted to S. Its control mesh, the base mesh $C_0(P_0, T_0)$, is composed of two sets - vertices and triangles. We denote $N = \#(P_0)$ as the number of degrees of freedom (control vertices) and we usually assume $M \gg N$. The uniform refinement levels of C_0 are $\{C_1 \cdots C_k\}$ and the meshes $\widehat{C_j}(\widehat{P_j}, T_j)$ are obtained by applying the limit surface projection operator to the control vertices P_j of C_j.

2 Optimization of the L^2 metric

We examine the problem of finding the best L^2 approximation of a given set of samples $S = \{s_i\}$ by a Loop subdivision surface D with fixed number of control

Figure 3. From left to right: a) original model with 30696 sample points, b) initial approximation with 461 control points, L^2 error 82.9, c) after 25 parameter correction and re-fitting steps, L^2 error 42.6, d) and e) show color coded errors before and after the 25 optimization steps. Notice the concentration of the error to local "hot spots" which makes it easier to detect regions where the insertion of new control vertices effectively reduces the L^∞ error.

points and connectivity. Since S is a discrete set, the L^2 error is expressed by the following term:

$$L^2(\{s_i\}, D) = \left(\sum_{i=1}^{M} \|s_i - D(t_i)\|_2^2 \right)^{\frac{1}{2}}, \tag{1}$$

where $\{t_i\}$ are the parameter values assigned to the samples $\{s_i\}$ with respect to some parametrization of the surface D. The most common way is to use barycentric coordinates with respect to the triangles of the base mesh C_0, i.e., $t_i =< f_i, (v_i, w_i) >$, where $f_i \in T_0$ indicates the patch to which the sample s_i is mapped and $(1 - v_i - w_i, v_i, w_i)$ define the barycentric coordinates of t_i within this triangle.

Given a fixed correspondence $s_i \leftrightarrow t_i$, the problem of minimizing (1) is solved in the *least squares* sense by finding that solution P_0 which minimizes the L^2 residuum of the over-determined linear system

$$A\,P_0 = S\ . \tag{2}$$

In order to compute the matrix $A = [\phi_j(t_i)]_{M \times N}$ for arbitrary parameterizations $\{t_i\}$, we have to evaluate the basis functions $\{\phi_1 \cdots \phi_N\}$ which define D at $\{t_1 \cdots t_M\}$ (see 2.1). Solving the system (2) gives us the optimal position of the control vertices P_0 of D. The sparsity of the matrix A depends on the support of the basis functions ϕ_j. In the case of Loop subdivision surfaces, each patch (corresponding to one triangle of the base mesh) is affected by 12 control vertices on average. Hence the matrix A has about 12 non-zero coefficients per row. There are many different ways to efficiently solve (2). In our implementation we use an iterative method such

as CGLS [24].

Another way to minimize (1) is by performing parameter correction, i.e., by finding for every sample point s_i the parameter t_i of the closest point on D. Iterating least square fitting and parameter correction generates a sequence of solutions D^0, D^1, D^2, ..., each of which has a smaller L^2 error, until the approximation quality cannot be improved any more, i.e.

$$L^2(\{s_i\}, D^k) - L^2(\{s_i\}, D^{k+1}) < \varepsilon \ ,$$

or some other criterion is met, for example a maximum number of steps is performed.

2.1 Exact evaluation of the subdivision basis functions

A significant improvement of our method is due to the fact that the fitting procedure depends on the *exact* subdivision surface D instead of depending on some piecewise-linear approximation of it, as in [11]. To achieve this, we use the representation of D as a linear combination of one smooth basis function for each control vertex, i.e.

$$D(t) = \sum_{j=0}^{N} \phi_j(t) p_{0,j} \ , \tag{3}$$

where the parameter domain of D is modeled as the faces of the base mesh C_0. Unlike uniform tensor-product spline surfaces, the subdivision basis functions are not mere translates of each other. Instead, each ϕ_j depends on the valence of the corresponding control vertex $p_{0,j}$ and on the valences of its direct neighbors in C_0. Hence, the easiest way to evaluate a basis function ϕ_j is to assign an additional scalar attribute σ to each control vertex and set $\sigma = 1$ for $p_{0,j}$ and $\sigma = 0$ for all other vertices in P_0 [25], and then to apply Stam's evaluation procedure on the so-defined scalar-valued mesh.

2.2 Exact closest point search and parameter correction

The next ingredient of our L^2 error minimization procedure is an algorithm for finding the *exact* closest point $D(t_i)$ on a Loop subdivision surface D given an arbitrary sample point s_i. This is done by performing stabilized Newton iteration. The starting value for the iteration can be either a previously assigned parameter value t_i or a value obtained by searching for the closest point on a discrete piecewise linear approximation $\widehat{C_k}$. In the second case we use a variant of the MESH-framework

for evaluating Hausdorff distances between surfaces [26], which employs a spatial data structure in order to minimize "closest point on triangle" evaluations. We denote the initial solution as $t_{i,0}$.

In the j-th step of the Newton iteration we linearize the surface D at the current approximate solution $t_{i,j}$ by computing the tangent plane $T_{i,j}$, which is given by the Jacobian $\nabla D(t_{i,j}) \in \mathbb{R}^{3\times 2}$. In order to find an update vector $q_{i,j} \in \mathbb{R}^2$ in the parameter domain towards an improved estimate for the closest point $D(t_i)$, we determine the orthogonal projection of s_i onto $T_{i,j}$ by solving the following 2×2 linear system for $q_{i,j}$:

$$\{s_i - (D(t_{i,j}) + \nabla D(t_{i,j}) \cdot q_{i,j})\} \cdot \frac{\partial D}{\partial v}(t_{i,j}) = 0$$
$$\{s_i - (D(t_{i,j}) + \nabla D(t_{i,j}) \cdot q_{i,j})\} \cdot \frac{\partial D}{\partial w}(t_{i,j}) = 0$$

Special care has to be taken when actually updating the parameter value $t_{i,j} =< f_{i,j}, (v_{i,j}, w_{i,j}) >$ since the parameter domain of the surface D is split into disjoint triangles corresponding to the faces of the base mesh C_0. In order to avoid an excessive number of special cases, we simply consider the following three cases:

(1) When $(v_{i,j}, w_{i,j}) + q_{i,j}$ still lies in the same triangle $f_{i,j}$ then $t_{i,j+1} :=< f_{i,j}, (v_{i,j}, w_{i,j}) + q_{i,j} >$.
(2) When $(v_{i,j}, w_{i,j}) + q_{i,j}$ lies outside $f_{i,j}$, i.e., the update moves into a neighboring patch, then we scale $q_{i,j}$ down by a factor $0 < \lambda < 1$ such that the updated parameter value $t_{i,j+1} :=< f_{i,j}, (v_{i,j}, w_{i,j}) + \lambda q_{i,j} >$ lies exactly on the boundary of the patch $f_{i,j}$. By this we avoid the re-parametrization that would be necessary to compute the proper barycentric coordinates of the vector $q_{i,j}$ in the next parameter triangle.
(3) If $t_{i,j}$ lies already on an edge of $f_{i,j}$ and $q_{i,j}$ is pointing outside then we switch to the neighbor face $f_{i,j+1}$ into which $q_{i,j}$ points, i.e., $t_{i,j+1} :=< f_{i,j+1}, (v_{i,j+1}, w_{i,j+1}) >$ where $(v_{i,j+1}, w_{i,j+1})$ are the barycentric coordinates of the same common boundary point with respect to the new triangle $f_{i,j+1}$. The actual parameter update will be executed in the next iteration.

The distinction between case (2) and (3) is necessary because it is difficult to predict if the update vector $q_{i,j+1}$ in the next Newton iteration will point into the same direction as $q_{i,j}$ or in the opposite direction.

If the starting value $t_{i,0}$ is not sufficiently close to the exact solution, it might happen that the Newton iteration suggests a parameter value where the L^2 distance actually increases, i.e. $\|s_i - D(t_{i,j+1})\|_2 \geq \|s_i - D(t_{i,j})\|_2$. This usually means that the length of the update step $q_{i,j}$ is incorrect, which is a common behavior of any root-finding algorithm for multivariate functions. To handle such situations in a robust manner, we switch to a reliable univariate optimization tech-

nique like Brent minimization [27] to find the minimum $\hat{h}$ of the function $g(h) = \|s_i - D((v_{i,j}, w_{i,j}) + hq_{i,j})\|_2, h \in (0,1)$. Finally we set $t_{i,j+1} =< f_{i,j}, (v_{i,j}, w_{i,j}) + \hat{h}q_{i,j} >$.

We stop the closest point search whenever $\|q_{i,j}\|_2 < \varepsilon$ or $j > n$. In our test cases we always used $\varepsilon = 10^{-6}$ and $n = 500$. Because of the robust minimization-based backtracking, we observed even for very complicated models with more than 200000 sample points less than 0.01% failures to converge with respect to the tolerance ε in less than n update steps. If such a failure occurs, we compare the newly found solution at $t_{i,n}$ with the old solution at $t_{i,0}$ and keep the better one. This guarantees the unconditionally stable and monotonic convergence of the parameter correction procedure. A case where $t_{i,0}$ is actually better than $t_{i,n}$ occurs extremely rarely. In most of our experiments the Newton iteration converges on average in less than 6 update steps.

It is important to notice the difference of our parameter correction scheme to previous approaches [11], where a piecewise linear approximation of the limit surface was used for the closest point search. An update step that reduces the distance between a sample point s_i and a piecewise linear approximation $\widehat{C_k}$ of the limit surface D does not necessarily reduce the distance between s_i and D. To illustrate this we repeated the experiment of Fig. 3 searching for the closest point on $\widehat{C_2}$, as proposed in [11]. The L^2 error after the 25 optimization steps was larger by 26.24%. Since the test control mesh was relatively coarse, we could repeat the experiment again, this time using $\widehat{C_4}$ as a piecewise linear approximation of D. The exact solution was still better by 11.4%. Note that this approach is not applicable for the optimization of large control meshes (without the use of a sophisticated adaptive subdivision algorithm), since it requires us to subdivide the control mesh at every step to a 64 times larger mesh.

2.3 Second order distance function approximants

The distance metric (1) is used in most approximation setups aiming at globally minimizing the distance between an approximating surface and sampled geometry data. However methods optimizing with respect to (1) are known to converge linearly and one would like to use a quadratic method since then the iterative approximation procedure would require less iterations to obtain a local minimum in the $3N$ dimensional space of the control vertices of the approximating mesh. As observed in [23], the squared distance between a sample point s_i and point $p = s_i + \lambda n_i$ (n_i is the sampled surface normal at s_i) is a *second* order approximant of the distance function of S at s_i only if these two points are relatively far away from another. On the other side, as proved in [28], the approximant (5) given below is a second order approximant of the squared distance function. Correspondingly an approximation procedure which interleaves fitting and correspondence correction using (5) is a

Newton method and has quadratic convergence [22].

The second order approximant ([23]) is defined as follows: Let κ_1 and κ_2 be the principal curvatures of S at s_i and $\rho_1 = 1/\kappa_1$, $\rho_2 = 1/\kappa_2$, and e_0 and e_1 are the corresponding principal curvature directions (for umbilic points any two orthogonal vectors lying in the tangent plane at s_i could be used). The vectors e_0, e_1 and $n = e_0 \times e_1$ define a local coordinate frame at s_i in which the point p has coordinates $(0, 0, d)$. If $|d| < \min(|\rho_1|, |\rho_2|)$, then the second order approximant of squared distance function of S at p is given by:

$$F_d(x_1, x_2, x_3) = \frac{d}{d - \rho_1} x_1^2 + \frac{d}{d - \rho_2} x_2^2 + x_3^2. \tag{4}$$

In practice an always positive variant of (4) is used:

$$F_d^+(x_1, x_2, x_3) = \frac{d}{d + |\rho_1|} x_1^2 + \frac{d}{d + |\rho_2|} x_2^2 + x_3^2. \tag{5}$$

The fitting methods described in [23,22] use the sample correction (Fig. 2) technique to establish the correspondence between the approximating surface D and sampled surface S. This allows them to choose the sample point s_i always so that error vector $D(t_i) - s_i$ is parallel to n_i and hence apply (5) directly. However, our parameter correction technique chooses the point $D(t_i)$ so that $D(t_i) - s_i$ is parallel to the normal vector $N(t_i)$ at $D(t_i)$. In general $n_i \neq N(t_i)$ and to apply (5) to our parameter-correction based method, we have to consider the local squared distance approximant of the surface D at the parameter point $D(t_i)$ to s_i. Hence we have to compute the principal curvatures and directions of the fitting surface at every parameter value t_i. Since D changes in every optimization step, these terms have to be recomputed, which is prohibitively expensive. Moreover Loop subdivision surfaces do not have a well defined curvature tensor at extraordinary vertices and in vicinity of such vertices the curvature behaves badly.

Instead we observe that when $d \to 0$, then $F_d^+(x_1, x_2, x_3) = x_3^2$ which is the squared distance to tangent plane of D at t_i. Correspondingly when $d \to \infty$, $F_d^+(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$ which is the squared distance to the point $D(t_i)$. Therefore we define a distance metric at the sample s_i using a blend between these two limit cases:

$$E_i = \mu_i \left\| D(t_i) - s_i \right\|^2 + (1 - \mu_i) \left\| T_i(s_i) \right\|^2, \ \mu_i = \frac{\left\| D(t_i) - s_i \right\|}{2 \cdot \max_j \left\| D(t_j) - s_j \right\|} \tag{6}$$

where $T_i(s_i)$ is the distance to tangent plane of D at t_t, i.e, $T(s_i) =< N(t_i), s_i > - < D(t_i), N(t_i) >$. Minimizing E_i is a non-linear problem, but we replace $N(t_i)$

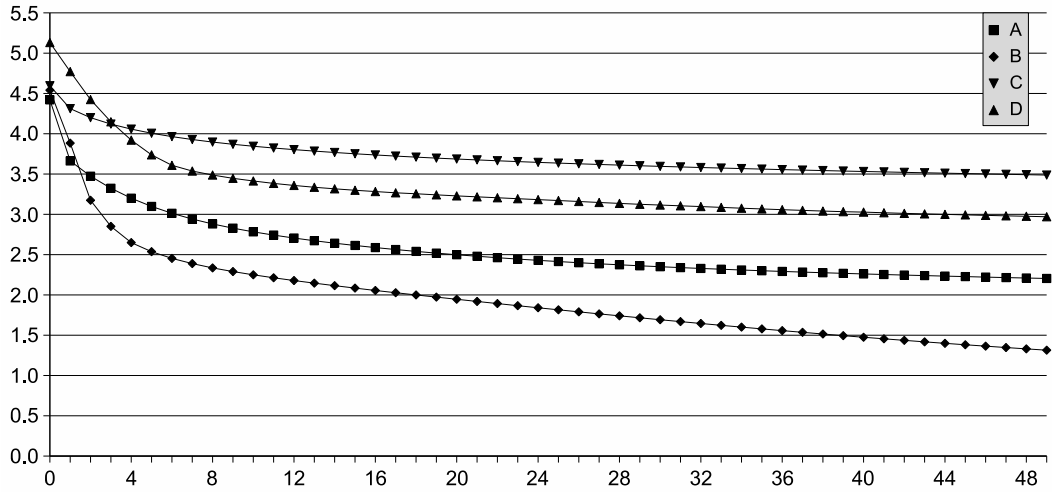

Figure 4. L^2 error optimization - convergence rate ($\log_2(L^2)$ vs. number of iterations) plot. The graphs A and B represent the convergence rate of the optimization procedure approximating a sphere (2354 samples) with a subdivision surface with 24 control points using (1) and (7) respectively. The graphs C and D represent the convergence rate of the optimization procedure approximating the Rocker Arm model (40177 samples) with a subdivision surface with 402 control points using (1) and (7) respectively.

and μ_i with their values for the current optimization step and solve only for the degrees freedom in the linear expansion of $D(t_i)$, see (3). In consequence, the energy:

$$E(\{s_i\}, D) = \sum_{i=0}^{M} E_i \tag{7}$$

is a quadratic functional and could be minimized by solving a linear system.

Note that (6) differs from (5) because it ignores the principal curvatures and directions of D since it is difficult to estimate them and only preserves the behavior of (5) in the limit cases. The blending factor μ_i is defined so that the samples relatively far from D are attracted more towards their foot-point $D(t_i)$, while points which are already very close to D are allowed to move tangentially.

To integrate (6) into our method, we simply replace the minimization of (1) with minimization of (7) when iterating the parameter correction and fitting steps in our L^2 error optimization procedure. We did not explore the theoretical properties of (7), but our practical experiments show that the so modified L^2 optimization procedure has faster convergence rate for isotropic objects and better convergence behavior for general objects (See Fig. 4). On the other hand this convergence advantage might be sometimes compromised by the fact that composing and solving the linear system required for minimizing (7) is computationally more involved. Also, due to the fact that unconstrained tangential movement of the control vertices is allowed, nothing prevents the approximating surface from folding over, and one has to use some kind of a fairing functional to prevent such artifacts.

3 Optimization of the L^∞ metric

While the L^2 error is a good measure for globally fitting a surface to sample data, the L^∞ error is much closer to the intuitive notion of approximation tolerance. Hence we present a technique to effectively reduce the L^∞ error by changing the structure of the control mesh. The L^∞ error for discrete sample data $\{s_i\}$ is defined as:

$$L^\infty(\{s_i\}, D) = \max_{1 \le i \le M} \|s_i - D(t_i)\|_2 \,. \tag{8}$$

In engineering applications this maximum tolerance $\delta \geq 0$ is usually set by the user. In [16,20,15,17] *multiresolution* subdivision surfaces are used to satisfy such user defined tolerances. Faces for which the maximum error is exceeded are subdivided adaptively and displacement vectors are added to the newly inserted vertices. This approach is robust and leads to quite good results, has the advantage of a genuine multiresolution semantics and is convenient for applications such as progressive transmission and compression. However, it also has some drawbacks:

(1) Looking at the results, e.g., in [15], it seems that the representation is highly redundant in terms of the number of coefficients that we need to faithfully represent complex objects. The representation is no longer unique, especially if multiple detail coefficients are assigned to the same control vertex on different refinement levels.
(2) The conceptual simplicity of the subdivision surface is lost since we have to deal with a combination of basis functions from different refinement levels. The evaluation can become tricky in regions where the adaptive refinement level changes.

To compensate for these difficulties, we present an iterative method for optimizing the control mesh of a plain subdivision surface D such that the approximation satisfies a given L^∞ error tolerance $\delta \geq 0$. We derive different heuristics to improve both the approximation and the surface quality. The method is based on a set of fundamental operations:

(1) Adaptive insertion of control vertices where the tolerance is not met (Section 3.1).
(2) Removal of control vertices in under-sampled regions (Section 3.2).
(3) Re-establishing the parameterization $\{t_i\}$ of the samples $\{s_i\}$ after the control mesh (and hence the surface D) has changed. This is done by re-running the parameter correction algorithm of Section 2.2.
(4) Connectivity regularization (Section 3.3).

The overall optimization procedure is described by the following pseudo-code:

```
Adaptively insert control vertices
do
  Regularize connectivity
  Re-establish the correspondence
  Remove control vertices in
  under-sampled regions
while removed_vertices_number > 0
Fit the new control mesh to the samples
```

The presented technique does not guarantee the achievement of the criterion $L^\infty(\{s_i\}, D) < \delta$ in one single iteration, it is only a heuristic which identifies regions of C_0 which should be optimized with respect to the L^∞ error and the current parametrization $\{t_i\}$. In practice, one usually needs several iterations of the above procedure to satisfy the criterion. In Section 4 we interleave this iteration with the technique described in Section 2 to bound the growth of the number of control vertices.

3.1 Adaptive insertion of control vertices

With $Sf_i = \{s_i | t_i \in f_i\}$ we denote the set of samples mapped to $f_i \in T_0$. For every $f_i \in T_0$ we define

$$L^\infty(f_i) = \max_{s_k \in Sf_i} \|s_k - D(t_k)\|_2 .$$

If $L^\infty(f_i) > \delta$, we have to split the face f_i and so locally add new degrees of freedom. We denote the set of all to-be-split faces by $G = \{f_i | L^\infty(f_i) > \delta\}$.

There are several common ways to split a face, e.g., longest edge-split, 1-to-3 split, or 1-to-4 split with crack-fixing. We empirically found that the best way to adaptively refine the mesh in terms of surface quality and approximation is to 1-to-4 split every face from G and then to fix the resulting cracks by bisecting neighboring faces. This way of adaptive refinement least affects the regularity of the mesh since all newly inserted vertices have valence 5 or 6 and only the crack-fixing changes the valence of some existing vertices by one. Other adaptive refinement operators tend to produce much more irregular, i.e., non-valence-6, vertices which has a negative effect on the quality of the resulting limit surface.

Let Q be the set of control vertices affected by the adaptive refinement of G (including the crack-fixing). These vertices have a natural one-to-one correspondence to certain control vertices from the mesh C_1 obtained by applying the uniform Loop subdivision operator to the given mesh C_0. In the adaptively refined mesh, we assign to all control vertices from Q the vertex positions of the corresponding vertex

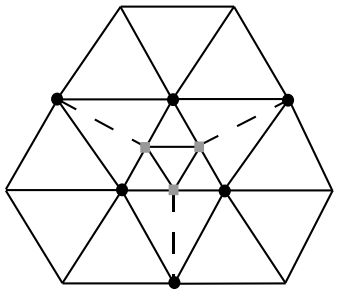

Figure 5. Adaptive insertion of control vertices. Squares correspond to the newly inserted control vertices when splitting the central face. Dots correspond to the original control vertices shifted to their position in C_1. Dashed lines denote crack-fixing edges.

in C_1 while the other control vertices keep their position from C_0 (Fig. 5). By this we minimize the instant modification of the surface D since the locally refined resolution of the control mesh is compensated by shifting the control vertices to their corresponding position on the next refinement level. The resulting change of the limit surfaces is significantly smaller compared to the adaptive refinement operator that simply inserts the new vertices at the midpoints of the edges. Notice, however, that we still treat the resulting control mesh as a single resolution plain subdivision surface.

3.2 Removal of control vertices in under-sampled regions

During the iteration of parameter correction, least squares fitting, and the adaptive insertion of new control vertices it can happen that some control vertices in C_0 become under-determined if too few samples (or more precisely, their associated parameter values) lie close enough to the center of the corresponding basis function's support. This leads to very unpleasant artifacts like ripples and bumps. The standard answer to these kind of instabilities, especially in the spline world, is to add a penalizing term (a fairing functional) to the minimization problem which, however, might affect the approximation quality in the properly sampled regions. Again by exploiting the flexibility of subdivision surfaces with respect to the connectivity of their control meshes, we propose an alternative approach that is based on detecting under-sampled vertices and removing them from the mesh.

From the mathematical point of view, under-determined control vertices degrade the condition number of the matrix A in the least squares equation (2) which makes the solution less robust. An appropriate value to rate the degree of being under-determined for a given vertex is the sum of the absolute values of the coefficients in the corresponding column of the matrix A. In the case of Loop subdivision, the basis functions and hence the matrix coefficients are all positive anyway.

The major drawback of this stability measure is that we need to evaluate it during the L^∞ optimization phase when no valid matrix A is available and hence we have

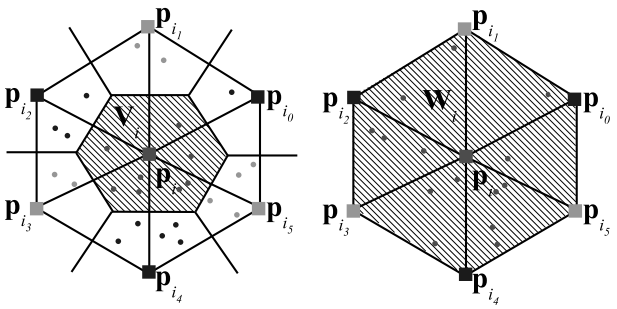

Figure 6. Left: the Voronoi cell V_i of a control vertex p_i is defined as the union of all Voronoi sectors of p_i. Right: the one-ring region W_i is defined as the union of the control faces adjacent to p_i. The dots represent the parametrization of the samples in the domains defined by the control faces. Samples are associated to the set SV_i (resp. SW_i) if their parameter value is inside V_i (resp. W_i). Depending on the L^∞ error, a vertex p_i is classified as under-sampled if SV_i or SW_i is empty.

to compute each of these coefficients by Stam's evaluation procedure. For efficiency reasons we therefore check if a control vertex is under-determined with a simplified criterion. The idea is to simply check if there are samples present at all in some region around the center of a basis function's support.

Since we observed that the stability of the least squares system is less critical if the approximating surface already fits very well to the given sample data, we actually define two criteria. One to be used when the L^∞ error is above some threshold τ and one to be used when it is below.

For every control vertex p_i we define the *Voronoi region* V_i as the union of all Voronoi sectors corresponding to the adjacent (parameter) triangles [29] and the *one-ring region* W_i as the union of all adjacent (parameter) triangles. Both regions cover some inner part of the basis function ϕ_i's support. Further we denote the set of samples that are associated with some parameter value in V_i or W_i by SV_i or SW_i respectively. Based on these definitions, we decide that a vertex p_i is under-determined iff the current L^∞ error is above the threshold τ and the set SV_i is empty or the current L^∞ error is below the threshold τ and the set SW_i is empty (Fig. 6).

The threshold τ could be set by the user. However, in our experiments we found that the choice of τ is not very critical so we simply set it to 0.1% of the bounding box diagonal of S for all the models that we tested.

Once a vertex is classified as under-determined we remove it by collapsing that half-edge connected to it which minimizes the connectivity metric (9) of the control mesh after the collapse.

3.3 Connectivity regularization

We use the following common metric [30] for rating the regularity of the connectivity of a mesh M:

$$R(M) = \sum_{v \in M} (d(v) - d_{opt}(v))^2, \tag{9}$$

where $d(v)$ is the valence of the vertex v and $d_{opt}(v) = 4$ if v is a boundary vertex or $d_{opt}(v) = 6$ if v is a non-boundary vertex. An edge flip is called *good* if it decreases $R(M)$. We build a candidate set H of all good edge flips and perform a greedy optimization by choosing the best flip $h_i \in H$, i.e., the one that maximizes the improvement $R(C_0) - R(C_0')$, in every step. After flipping we remove h_i and all edge flips affected by it from H and continue the greedy selection. Once H is empty we build a new set of candidates and check if there exist more good flips. The procedure stops once there are no more good edge flips in C_0. Although this approach is not as sophisticated as the one in [30] it usually converges quickly to a local minimum of $R(C_0)$ and successfully prevents the occurrence of high or low valence vertices in the control mesh.

4 Overall approximation procedure

Finding a good balance between the optimization of the L^2 error (Section 2) and the optimization of the L^∞ error (Section 3) is one of the key issues for achieving a high-quality approximation and in our implementation this balance is determined by the user who has to select two parameters - K and J. Here, K is the maximum number of optimization steps, and every J-th step we perform optimization with respect to the L^∞ error, i.e., change the structure of C_0. The user also prescribes the L^∞ error tolerance δ. The following pseudo-code implements the main approximation loop:

```
while k < K and L_k^∞ > δ
  if ((k+1 mod J) == 0)
    Optimize the L^∞ error
  else
    Optimize the L^2 error
  end
  k=k+1
end
```

As a rule of thumb, using relative large $J = 5, 6, .., 10$ is a good idea and often leads to control meshes with smaller complexity (Fig. 7) since it is generally worth investing effort in finding the best approximating surface with the current number of

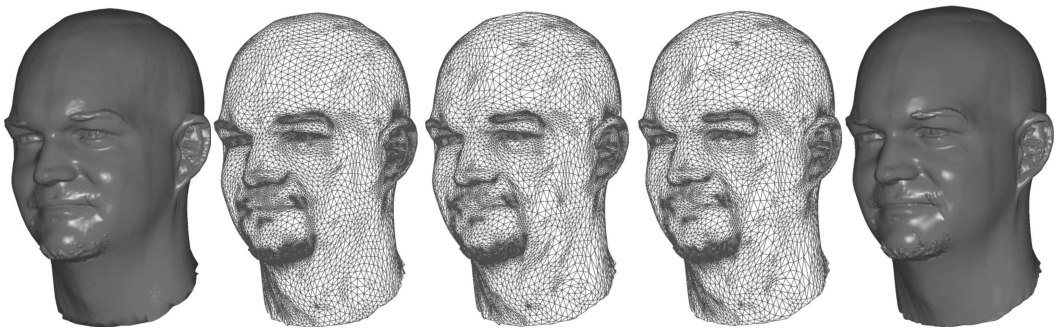

Figure 7. a) Original Cyberware scan of a male head with $320k$ triangles, $M = 160k$. From left to right: approximations produced by our method with relative tolerance 0.03% (as in [15]) for different values of the parameter J, $\widehat{C_0}$ is shown: b) $N = 16792$, $J = 1$, $8min$, c) $N = 14365$, $J = 3$, $15min$, d) $N = 13642$, $J = 5$, $22min$, e) shows (d) shaded. The initial approximating surface has 1600 vertices.

degrees of freedom, before trying to optimize the fit by adaptively inserting control vertices in the high error regions. One can also look at J as a parameter controlling the trade-off between mesh complexity and running time for a given tolerance.

4.1 Initial fitting surface

As in [13,19], given a polygonal mesh S we find the initial approximating subdivision surface D by decimating S using QEM-based mesh simplification [31] until the number of degrees of freedom, i.e., the number of control vertices of C_0, reaches some predefined number. A necessary condition for C_0 is to have the same topology as S. The final quality of the approximation might vary depending on the number of C_0 control vertices, but values from 1% to 5% of M provide always very good results for dense S. One might expect that using relatively large initial N over 5% of M will produce better results, but this often leads only to unjustified waste of degrees of freedom which do not contribute to the quality of the approximation and the overall minimization of the error.

The connectivity information of S is not used at any other place throughout the approximation procedure, therefore one could use any other method for determining the initial surface. In the future we intend to examine alternative approaches to construct the initial approximating surface with the same topology as the sample set, which will allow us to perform approximation also of non-triangulated point sets.

Figure 8. From left to right: a) the original Stanford bunny model. Different approximations (N, relative $L^\infty(S, D)$, time in *min:sec*): b) 612, 0.63%, 0:32, c) 913, 0.30%, 1:54 d) 4680, 0.12%, 2:36, e) 8440, 0.049%, 4:39.

5 Results

We tested our approximation method on several models (Table 1). The goal was to achieve high-quality approximation with L^∞ error not larger than 0.05% of the bounding box diagonal of the corresponding model. We also illustrate that we are able to quickly produce a relatively coarse fit with tolerance less than 1% and progressively improve the approximation by investing more time into the fitting procedure (Fig. 8).

We first compare our algorithm with the B-spline approximation method in [8]. The best approximation of the bunny model presented in that paper has a relative maximum deviation of 1.44%. The approximating surface consists of 153 patches. Taking the inter-patch G^1 smoothness conditions into account, we count on average 4 dofs (degrees of freedom) per bi-cubic patch. Note that the actual patches are defined by more control vertices, however most of them are used up to satisfy the C^0 continuity and the G^1 smoothness constraints across the patch boundaries. The estimated complexity corresponds to $153 \times 4 = 612$ dofs in the approximating Loop surface, where each dof corresponds to one control vertex. Using our procedure with an initial surface obtained by decimating the bunny model down to 612 vertices and performing 5 parameter correction steps, gives a relative maximum deviation of 0.63% and takes $32s$ to compute on 2.8GHz Pentium IV including the decimation (Fig. 8).

Next we compare our results to the multiresolution subdivision surface fitting technique proposed in [15] since this is, to our knowledge, the only work where subdivision surfaces have been used to produce high-quality approximations of complex objects comparable to ours. The algorithm presented in that paper is very efficient due to the quasi-interpolation fitting and the multiresolution hierarchy. However, as we show in Fig. 7, the number of degrees of freedom required in [15] for obtaining the same precision is significantly larger (8 times for this example) than in our method.

Finally we compare our results with the results of the most recent work in the field: [22]. We have to mention that both methods differ in several aspects. In [22] an ini-

Model	M	initial N	final N	$L^\infty(\%)$	hours:min
Fig. 1	352K	3518	15347	0.049	0:30
Fig. 7	160K	1600	13642	0.029	0:22
Fig. 8	37K	612	8440	0.048	0:05
Fig. 9	40K	804	4494	0.036	0:03
Fig. 10	546K	4093	17995	0.049	1:04
Fig. 11	51K	2028	4733	0.049	0:06
Fig. 12	31K	307	4698	0.048	0:05

Table 1
Results obtained by the procedure described in Section 4. The L^∞ error is given as a percentage of the bounding box diagonal of S. $J = 5$ in all of the experiments and the algorithm converged in less than the maximum allowed ($K = 100$) optimization steps. Timings are taken on 2.8GHz Pentium IV with 2GB RAM.

Model	M	initial N	final N	$L^\infty(\%)$	avg. error(%)	min:sec
Fig. 13	134K	336	1553	0.247	0.0575	08:29
Fig. 14	173K	433	2820	0.248	0.0598	12:26

Table 2
Approximation results of our algorithm for the models on Fig. 13. The L^∞ and the average error ($\frac{L^2(S,D)}{\sqrt{M}}$) are given as a percentage of the bounding box diagonal of S.

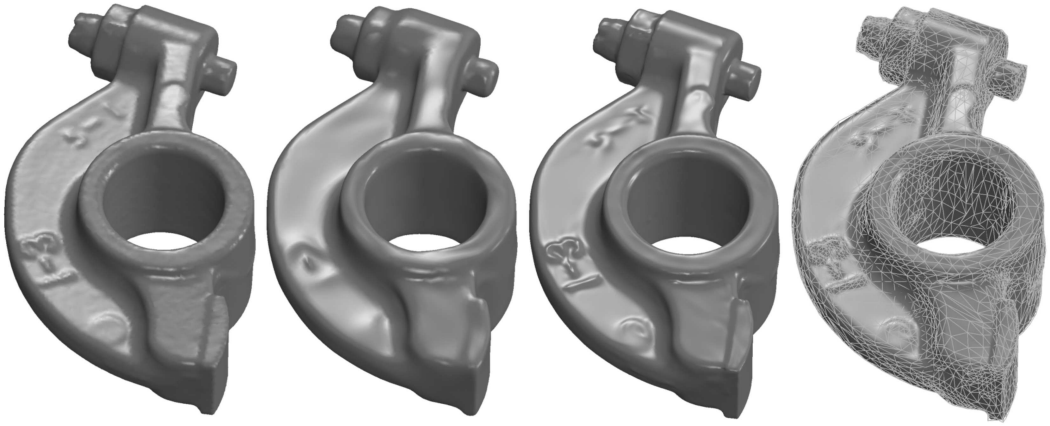

Figure 9. From left to right: Cyberware scan of a machine detail, initial D, final D and base mesh C_0 overlaid on the final D.

tial approximation surface is generated directly from the point cloud, while we use the triangulation of the original data samples in order to generate the initial control mesh through decimation. Also while [22] minimizes the approximation error only with respect to the L^2 metric, our work addresses the minimization of both L^∞ and L^2 metrics, so it is not surprising that we obtain significantly better results for the maximum deviation. Therefore in order to compare the both methods on a common

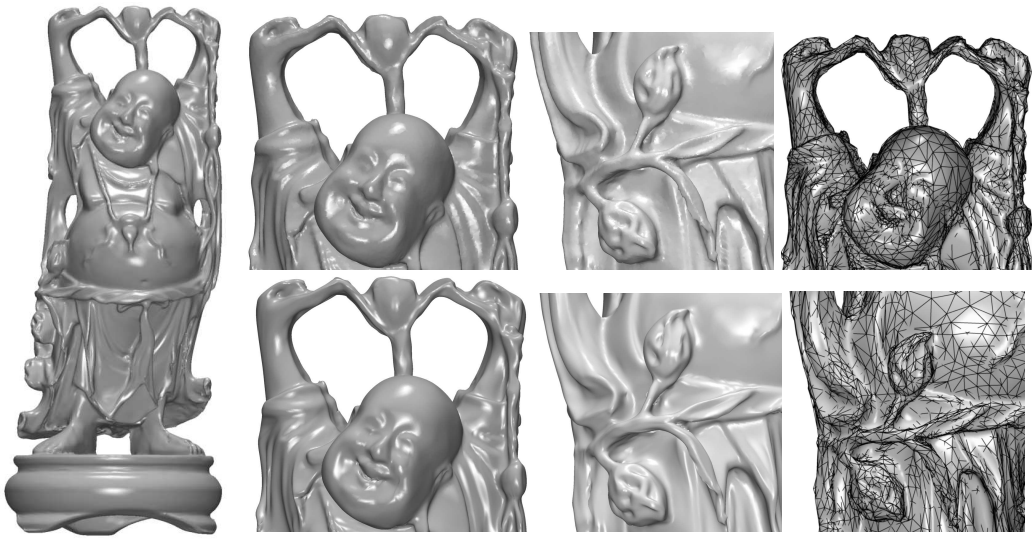

Figure 10. Left: an approximation D of the Buddha model. Third and second columns: top - close-up views of the original data, bottom - close-up views of D. Fourth column:close-up views of C_0 overlaid on D.

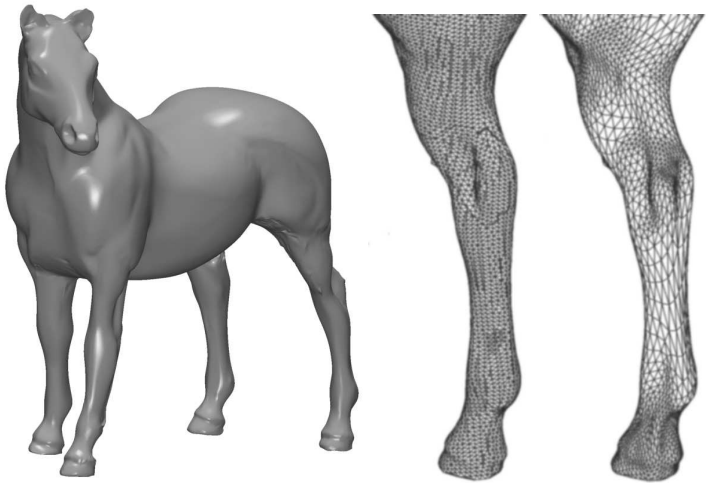

Figure 11. Left: an approximation of the Horse model. Right: close-up wire-frame views from behind on the leg: left - the original mesh, right - $\widehat{C_1}$. Note the non-uniform density of the approximating mesh corresponding to the varying geometrical complexity of the model.

ground, we increased the relative target L^∞ error tolerance for these experiments to 0.25%. In this setting our method gives more than 2 times smaller *average* error and produces control meshes with significantly less complexity (Table 2). The most important reason for that is that we use a parameter correction technique and therefore all of the original samples are taken into account during the fitting process. However, because of the same reason, our method requires about 30% more computational time. Instead in [22] a sample correction technique is used. The sample correction procedure might leave important regions of the original surface undersampled (Fig. 2) and in consequence this might affect negatively the approximation

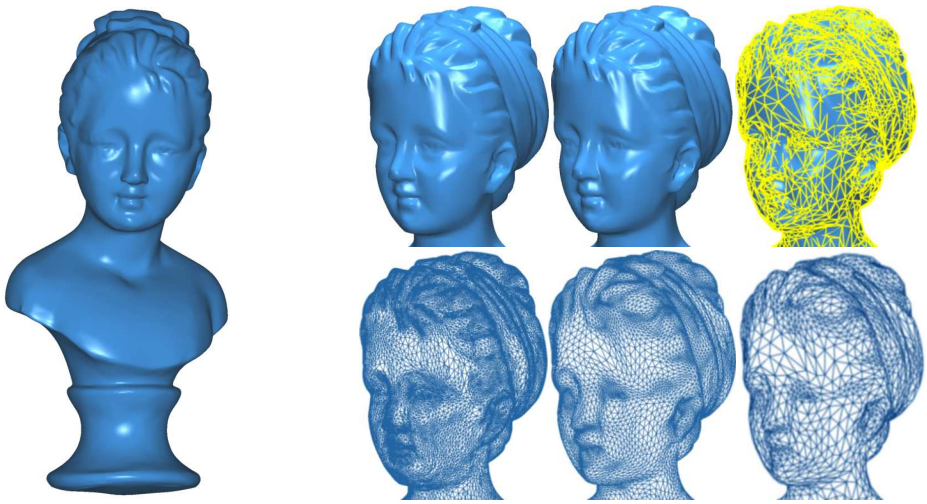

Figure 12. Left: an approximation D of the Bust model. Right: solid and wire-frame close-up views: left - original data, middle - final D and $\widehat{C_2}$, right - C_0 overlaid on D and $\widehat{C_0}$.

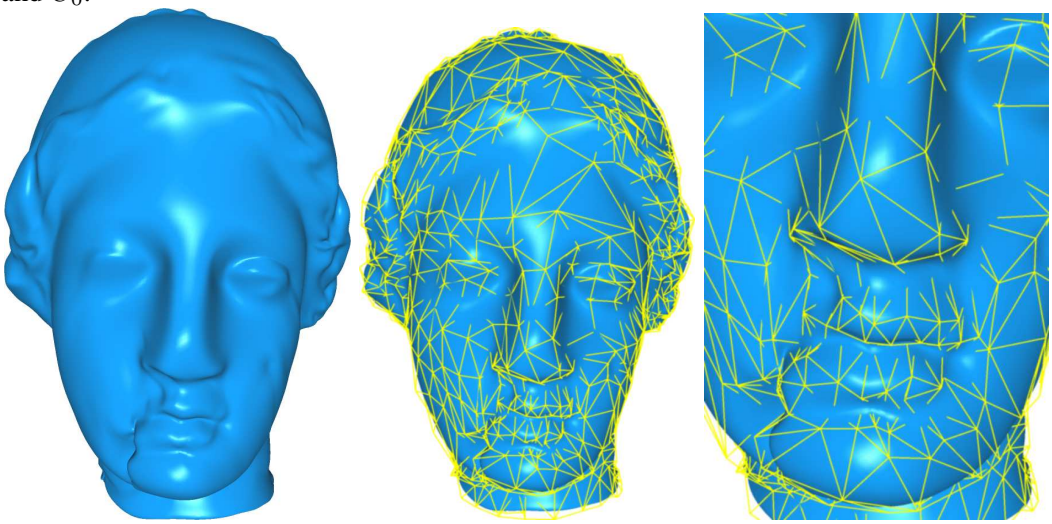

Figure 13. Approximation of the Igea dataset produced by our method. C_0 is shown overlaid on D.

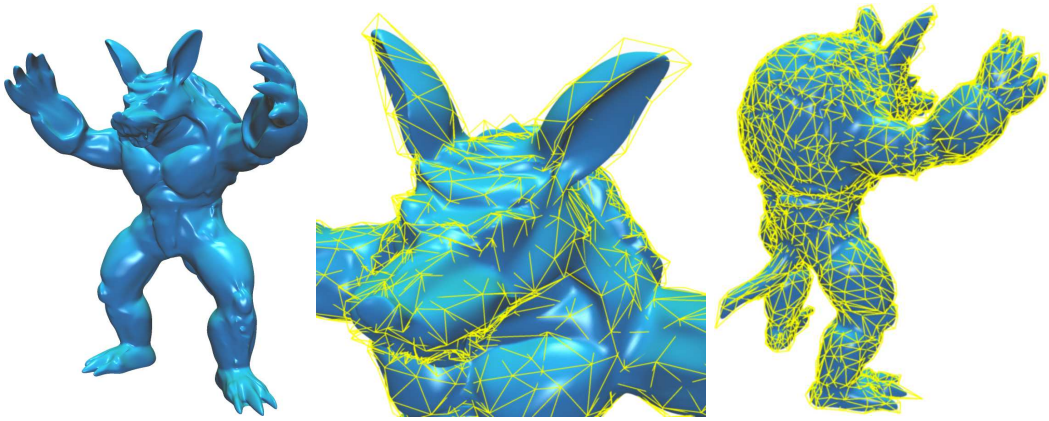

Figure 14. Approximation of the Armadillo dataset produced by our method. C_0 is shown overlaid on D.

error.

6 Future work

The parameter correction procedure, which we use to establish the correspondence between the samples and the approximating surface, does *not* guarantee one-to-one mapping in all cases. Nevertheless, in practice we observed flipping only when the initial fitting control mesh was extremely coarse and could always be avoided by allowing enough degrees of freedom from the beginning. The connectivity regularization procedure could be improved using [30]. Full support of piecewise-smooth subdivision surfaces [32] and a lot of performance optimizations are still pending in our implementation. We plan also to make more research into possible applications of the method described in Section 2.3 for reducing even more the complexity of the control meshes required for high quality approximations.

7 Acknowledgment

This work was partially supported by the European Union Research Training Network "Multiresolution in Geometric modeling (MINGLE)" under grant HPRN-CT-1999-00117. The Buddha model is courtesy of Igor Guskov and Zoë Wood and is produced with the procedure described in [33]. The Rocker Arm, the male head, the Igea and the Armadillo models are courtesy of Cyberware. The Bunny model is courtesy of the Stanford University. We would like to thank Sergey Alekseev and Mario Botsch for their help to convert the original models on Fig. 1 and Fig. 7 and the anonymous reviewers for their constructive comments.

References

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

[2] E. Catmull, J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, Computer Aided Geometry Design 10 (6) (1978) 350–355.

[3] J. Stam, Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values, in: SIGGRAPH'98, Vol. 32, 1998, pp. 395–404.

[4] J. Stam, Evaluation of Loop subdivision surfaces, in: SIGGRAPH'99 Course notes, 1999.

[5] U. Dietz, Erzeugung glatter flähen aus messpunkten, Technikal Report 1717, Department of Mathematics, University of Darmstadt, Germany (1995).

[6] N. S. Sapidis (Ed.), Designing fair curves and surfaces: shape quality in geometric modeling and computer-aided design, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994.

[7] A. Cohen, J.-L. Merrien, L. L. Schumaker (Eds.), Curve and Surface Fitting: Saint-Malo 2002, Nashboro Press, Brentwood, TN, 2003.

[8] M. Eck, H. Hoppe, Automatic reconstruction of B-spline surfaces of arbitrary topological type, Computer Graphics 30 (Annual Conference Series) (1996) 325–334.

[9] D. Zorin, P. Schröder, W. Sweldens, T. DeRose, L. Kobbelt, A. Levin, Subdivision for modeling and animation, in: SIGGRAPH'2000 Course notes, 2000.

[10] M. Halstead, M. Kass, T. DeRose, Efficient, fair interpolation using Catmull-Clark surfaces, in: Computer Graphics, ACM Press, 1993, pp. 35–44.

[11] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, W. Stuetzle, Piecewise smooth surface reconstruction, in: SIGGRAPH'94, 1994, pp. 295–302.

[12] H. Suzuki, S. Takeuchi, T. Kanai, Subdivision surface fitting to a range of points, in: Pacific Graphics, 1999, pp. 158–167.

[13] S. Takeuchi, H. Suzuki, F. Kimura, T. Kanai, K. Shimada, Subdivision surface fitting using QEM-based mesh simplification and reconstruction of approximated B-spline surfaces, in: Pacific Graphics, 2000.

[14] W. Ma, N. Zhao, Catmull-Clark surface fitting for reverse engineering, in: Geometric Modeling and Processing, 2000, pp. 274–284.

[15] N. Litke, A. Levin, P. Schröder, Fitting subdivision surfaces, in: IEEE Visualization 2001, 2001, pp. 319–324.

[16] A. Lee, H. Moreton, H. Hoppe, Displaced subdivision surfaces, in: K. Akeley (Ed.), SIGGRAPH'2000, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000, pp. 85–94.

[17] N. Litke, A. Levin, P. Schröder, Trimming for subdivision surfaces, Computer Aided Geometry Design 18 (5) (2001) 463–481.

[18] W.-K. Jeong, C.-H. Kim, Direct reconstruction of a displaced subdivision surface from unorganized points, Graph. Models 64 (2) (2002) 78–93.

[19] W. Ma, X. Ma, S.-K. Tso, Z. Pan, Subdivision surface fitting from a dense triangle mesh, in: Geometric Modeling and Processing, 2002, pp. 94–103.

[20] H. Biermann, D. Kristjansson, D. Zorin, Approximate Boolean operations on free-form solids, in: SIGGRAPH'2001, 2001, pp. 185–194.

[21] D. Zorin, Subdivision and multiresolution surface representations, Ph.D. thesis, Caltech, Pasadena, California (1997).

[22] D. Cheng, W. Wang, H. Qin, K. K. Wong, H. Yang, Y. Liu, Fitting subdivision surfaces to unorganized point data using SDM, in: Proceedings of the 12th Pacific Conference on Computer Graphics and Applications, IEEE Computer Society, 2004, pp. 16–24.

[23] H. Pottmann, S. Leopoldseder, A concept for parametric surface fitting which avoids the parametrization problem, Computer Aided Geometry Design (20) (2003) 350–355.

[24] T. Elfving, On the conjugate gradient method for solving linear least squares problems, Tech. Report LiTH-MAT-R-78-3, Linköping University, Sweden (1978).

[25] J. Bolz, P. Schröder, Rapid evaluation of Catmull-Clark subdivision surfaces, in: Proceeding of the seventh international conference on 3D Web technology, ACM Press, 2002, pp. 11–17.

[26] N. Aspert, D. Santa-Cruz, T. Ebrahimi, MESH: Measuring errors between surfaces using the Hausdorff distance, in: Proceedings of the IEEE International Conference on Multimedia and Expo, Vol. I, 2002, pp. 705 – 708, `http://mesh.epfl.ch`.

[27] R. P. Brent, Algorithms for Minimization without Derivatives, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[28] H. Pottmann, M. Hofer, Geometry of the squared distance function to curves and surfaces, Springer, 2003.

[29] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, in: H.-C. Hege, K. Polthier (Eds.), Visualization and Mathematics III, Springer-Verlag, Heidelberg, 2003, pp. 35–57.

[30] V. Surazhsky, C. Gotsman, Explicit surface remeshing, in: Proceedings of Eurographics Symposium on Geometry Processing, 2003, pp. 17–28.

[31] M. Garland, P. S. Heckbert, Surface simplification using quadric error metrics, Computer Graphics 31 (Annual Conference Series) (1997) 209–216.

[32] D. Zorin, D. Kristjansson, Evaluation of piecewise smooth subdivision surfaces, The Visual Computer 18 (5-6) (2002) 299–315.

[33] I. Guskov, Z. Wood, Topological noise removal, in: B. Watson, J. W. Buchanan (Eds.), Proceedings of Graphics Interface, 2001, pp. 19–26.