Fitting Smooth Surfaces to Dense Polygon Meshes

Venkat Krishnamurthy
Marc Levoy

Computer Science Department
Stanford University

Abstract

Recent progress in acquiring shape from range data permits the acquisition of seamless million-polygon meshes from physical models. In this paper, we present an algorithm and system for converting dense irregular polygon meshes of arbitrary topology into tensor product B-spline surface patches with accompanying displacement maps. This choice of representation yields a coarse but efficient model suitable for animation and a fine but more expensive model suitable for rendering.

The first step in our process consists of interactively painting patch boundaries over a rendering of the mesh. In many applications, interactive placement of patch boundaries is considered part of the creative process and is not amenable to automation. The next step is gridded resampling of each bounded section of the mesh. Our resampling algorithm lays a grid of springs across the polygon mesh, then iterates between relaxing this grid and subdividing it. This grid provides a parameterization for the mesh section, which is initially unparameterized. Finally, we fit a tensor product B-spline surface to the grid. We also output a displacement map for each mesh section, which represents the error between our fitted surface and the spring grid. These displacement maps are images; hence this representation facilitates the use of image processing operators for manipulating the geometric detail of an object. They are also compatible with modern photo-realistic rendering systems.

Our resampling and fitting steps are fast enough to surface a million polygon mesh in under 10 minutes - important for an interactive system.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling —curve, surface and object representations; I.3.7[Computer Graphics]:Three-Dimensional Graphics and Realism—texture; J.6[Computer-Aided Engineering]:Computer-Aided Design (CAD); G.1.2[Approximation]:Spline Approximation

Additional Key Words: Surface fitting, Parameterization, Dense polygon meshes, B-spline surfaces, Displacement maps

Authors' Address: Department of Computer Science, Stanford University, Stanford, CA 94305
E-mail: venkat,levoy@cs.stanford.edu
World Wide Web: http://www-graphics.stanford.edu/∼venkat, ∼levoy

1 Introduction

Advances in range image acquisition and integration allow us to compute geometrical models from complex physical models [9, 36]. The output of these technologies is a dense, seamless (i.e. manifold) irregular polygon mesh of arbitrary topology. For example, the model in figure 12, generated from 75 scans of an action figure using a Cyberware laser range scanner, contains 350,000 polygons. Models like this offer new opportunities to modelers and animators in the CAD and entertainment industries.

Dense polygon meshes are an adequate representation for some applications. Several commercial animation houses employ polygon meshes almost exclusively. However, for reasons of compactness, control, manufacturability, or appearance, many users prefer smooth surface representations. To satisfy these users, techniques are needed for fitting surfaces to dense meshes of arbitrary topology.

A notable property of these new acquisition techniques is their ability to capture fine surface detail. Whatever fitting technique we employ should strive to retain this fine detail. Surprisingly, a unified surface representation may not be the best approach. First, the heavy machinery of most smooth surface representations (for example B-splines) makes them an inefficient way to represent fine geometric detail. Second and perhaps more important, although geometric detail is useful at the rendering stage of an animation pipeline, it may not be of interest to either the modeler or the animator. Moreover, its presence may degrade the time or memory performance of the modeling system. For these reasons, we believe it is advantageous to separate the representations of coarse geometry and fine surface detail.

Within this framework, we may choose from among many representations for these two components. For representing coarse geometry, modelers in the entertainment and CAD industry have long used NURBS [14] and in particular uniform tensor product B-splines. Such models typically consist of control meshes stitched together to the level of continuity desired for an application. In order to address their needs we have chosen uniform tensor product B-splines as our surface representation.

For representing surface detail, we propose using displacement maps. Each pixel in such a map gives an offset from a point on a fitted surface to a point on a gridded resampling of the original polygon mesh. The principal advantage of this representation is that displacement maps are essentially images. As such, they can be processed, retouched, compressed, and otherwise manipulated using simple image processing tools. Some of the effects shown in figures 11 and 13 were achieved using Adobe Photoshop, a commercial photo retouching program.

1.1 System overview

Figure 1 shows the pipeline for our system. We start with a connected polygon mesh. The additional connectivity information offered by a polygonal representation is used to advantage at every stage of our pipeline. Our steps are as follows:

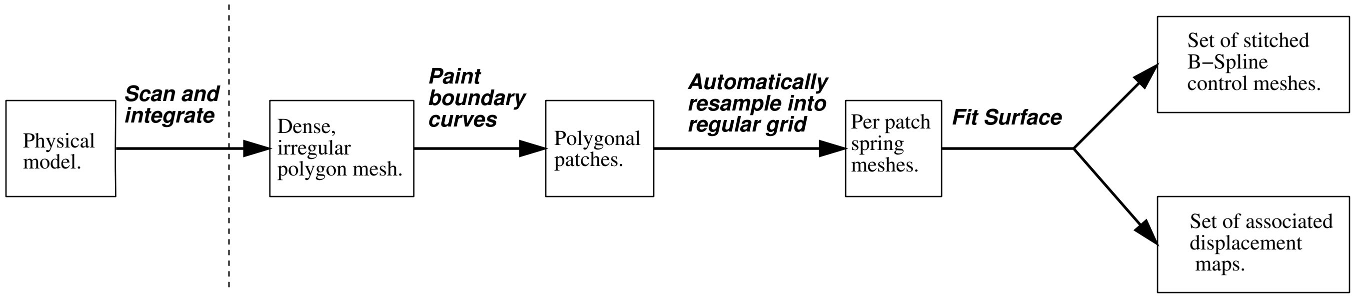

Figure 1. Our surface fitting pipeline: the input to our system is a dense irregular polygon mesh. First, boundary curves for the desired spline patches are painted on the surface of the unparameterized model. The output of this step is a set of bounded mesh regions. We call each such region a polygonal patch. We next perform an automated resampling of this polygonal patch to form a regular grid that lies on the surface of the polygonal patch. We call this regular grid a spring mesh. Finally, we fit surfaces to the spring mesh and output both a B-spline surface representation and a set of associated displacement maps to capture the fine detail.

1. Our first step is an interactive boundary curve painting phase wherein a modeler defines the boundaries of a number of patches. This is accomplished with tools that allow the painting of curves directly on the surface of the unparameterized polygonal model. Here, the connectivity of the polygon mesh allows the use of local graph search algorithms to make curve painting operations rapid. This property is useful when a modeler wishes to experiment with different boundary curve configurations for the same model. Each region of the mesh that a B-spline surface must be fit to is called a *polygonal patch*. Since patch boundaries have been placed for artistic reasons, polygonal patches are not constrained to be height fields. Our only assumptions about them are that each is a rectangularly parameterizable piece of the surface that has no holes.

2. In the next step we generate a gridded resampling for each polygonal patch. This is accomplished by an automatic coarse-to-fine resampling of the patch, producing a regular grid that is constrained to lie on the polygonal surface. We call this grid the *spring mesh*. Its purpose is to establish a parameterization for the unparameterized surface. Our resampling algorithm is a combination of relaxation and subdivision steps that iteratively refine the spring mesh at a given resolution to obtain a better sampling of the underlying polygonal patch. This refinement is explicitly directed by distortion metrics relevant to the spline fit. The output of this step is a fine gridding of each polygonal patch in the model.

3. We now use standard gridded data fitting techniques to fit a B-spline surface to the spring mesh corresponding to each polygonal patch. The output of this step is a set of B-spline patches that represent the coarse geometry of the polygonal model. To represent fine detail, we also compute a displacement map for each patch as a gridded resampling of the difference between the spring mesh and the B-spline surface. This regular sampling can conveniently be represented as a vector (rgb) image which stores a 3-valued displacement at each sample location. Each of these displacements represents a perturbation in the local coordinate frame of the spline surface. This image representation lends itself to a variety of interesting image processing operations such as compositing, painting, edge detection and compression. An issue in our technique, or in any technique for fitting multiple patches to data is ensuring continuity between the patches. We use a combination of knot line matching and a stitching post-process which together give us G^1 continuity everywhere. This solution is widely used in the entertainment industry.

The remainder of this paper is organized as follows. Section 2 reviews relevant previous work. Section 3 describes our techniques for painting boundary curves over polygonal meshes. Section 4 presents our coarse-to-fine, polygonal patch resampling algorithm and the surface fitting process. Section 5 describes our strategy for extracting displacement maps and some interesting applications thereof. Section 6, discusses techniques for dealing with continuity across patch boundaries. Finally, section 7 concludes by discussing future work.

Throughout this paper we will draw on examples from the entertainment industry. However, our techniques are generally applicable.

2 Previous work

There is a large literature on surface fitting techniques in the CAD, computer vision and approximation theory fields. We focus here on only those techniques from these areas that use dense (scanned) data of arbitrary topology to produce smooth surfaces. We can classify such surface fitting techniques as manual, semi-automated and automated.

2.1 Manual techniques

Manual approaches can be divided into two categories. The first category includes all methods for digitizing a physical model directly. For example, using a touch probe, one can acquire only data that is relevant to the final surface model. Catalogues of computer models published by ViewPoint Data Labs and the work of Pixar's animation group [24, 28] exemplify these methods. These methods involve human intervention throughout the data acquisition process and are hence time-consuming, especially if the model is complex or the data set required is large. In contrast, our pipeline employs automatic data acquisition methods [9].

The second category uses scanned data as a template to assist in the model construction process. Point cloud or triangulated data is typically imported into a conventional modeling system. A user then manually projects isolated points to this data as a means of determining the locations of control points (or edit points [15]) for smooth parametric surfaces. These methods require less human intervention than those in the first category but complex models may still require a lot of labour.

2.2 Semi-automated techniques

The approaches in this category take point cloud data sets as input. Examples include commercial systems such as Imageware's Surfacer [33], Delcam's CopyCAD, and some research systems [20, 22]. These approaches begin by identifying a subset of points that are to be approximated. Parameterization of data points is usually accomplished by a user-guided process such as projection of the points to a manually constructed base plane or surface. A constrained, non-linear least squares problem is then solved on this subset of the point cloud to obtain a B-spline surface for the specified region. While point cloud techniques are widely applicable, they fail to exploit topological information already present in the input data. As demonstrated by Curless et al [9] and Turk et al [36], using this additional information can significantly improve quality of reconstruction. In the context of our surface fitting algorithm, working with connected polygonal representations has also facilitated the development of an automatic parameterization scheme.

2.3 Automated surface fitting techniques

Eck et al [12] describe a method for fitting irregular meshes with a number of automatically placed bicubic Bezier patches. For the parameterization step, a piecewise linear approximation to harmonic maps [11] is used, and the number of patches is adjusted to achieve fitting tolerances. While this method produces high quality surfaces, it includes a number of expensive optimization steps, making it too slow for an interactive system. Further, their technique does not separate fine geometric detail from coarse geometry. Particularly for very dense meshes, we find this separation both useful and preferable, as already explained. We compare some other aspects of the parameterization scheme of Eck et al [11] with ours in section 4.10.

We briefly mention some techniques [29, 31] that use hierarchical algorithms to fit parametric surfaces to scanned data sets. While these approaches work well for regular data, they do not address the problem of unparameterized, irregular polygon meshes. Finally, Sclaroff et al [32] demonstrate the use of displacement maps in the context of interpolating data with generalized implicit surfaces. However, this method also works only on regular data sets.

2.4 Relevant work in texture mapping

A key aspect of our method is an automatic parameterization scheme for irregular polygon meshes. As such, there are techniques in the texture mapping literature that address similar problems, notably the work of Bennis et al [6] and that of Maillot et al [21]. Both of these papers present schemes to re-parameterize surfaces for texture mapping. These algorithms work well with regular data sets, such as discretized splines. However, they can exhibit objectionable parametric distortions in general [11]. Pedersen [25] describes a method for texture mapping (and hence parameterizing) implicit surfaces. While the methods work well with implicit surfaces, they rely on smoothness properties of the surface and require the evaluation of global surface derivatives. Irregular polygon meshes in general, are neither smooth nor conducive to the evaluation of global surface derivatives, as discussed by Welch et al [38].

3 Boundary curve specification

Our surface fitting pipeline starts with the user interactively segmenting the polygonal model into a number of regions that are to be approximated by spline patches. A patch is specified by interactively painting its boundary curves. This operation should be fast and provide intuitive feedback to the user. We have found that curves that lie on the surface of the model are easier to specify and manipulate than unconstrained space curves. A polygonal (discrete) geodesic [23] is one possible representation. Unfortunately, this is expensive both to compute and to maintain. We instead represent patch boundaries as sampled geodesics. We call these face-point curves. The steps for painting a boundary curve are shown and described in figure 3.

This painting process yields a piecewise linear face-point curve on the surface through a sequence of picked vertices. We now smooth this face-point curve on the surface using a fitted B-spline curve [30]. The resolution of the B-spline for the fit is set interactively by the user. The smoothing operation consists of attracting a face-point curve, which is constrained to lie on the surface of the polygon mesh, to a curve in space. This is a fast technique for obtaining the projection of a space curve to a sampled curve on the polygon mesh. Figure 4 sums up this procedure.

Graph corresponding to polygon mesh	The graph consisting of vertices and edges of the polygon mesh, with edge lengths given by Euclidean distance.
Graph path	A path between two vertices of the above graph.
Polygonal patch	Each bounded region of the polygon mesh, as determined by the painted boundary curves.
Face point	A point that lies on some face of the polygon mesh.
Face−point curve	A list of face points that form a piecewise linear curve.
Spring mesh	A grid of face points that lie on the faces of a polygonal patch.
Force (P2, P1) where P1 and P2 represent points in space	A 3−vector given by P2 − P1.

Figure 2. A glossary of common terms we use through this paper.

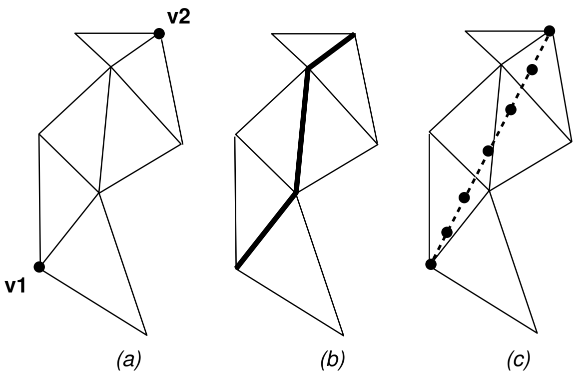

Figure 3. Boundary curve painting. The user picks a sequence of points with a mouse on a 2-D projection (rendering) of the polygon mesh. The program automatically associates the nearest vertex of the mesh to each of these points. Two such vertices (v1 and v2) are shown in (a). Between each successive pair of such vertices from (a), we compute the projection on the surface, of a straight line connecting the two. This is performed in two steps. First, the greedy graph path between these two vertices is computed as shown in (b) (as a thick polyline). This path is then sampled into a face-point curve and smoothed into a straight line as illustrated in (c). Filled circles represent individual face points. The face-point curve now represents a sampling of the projection on the polygonal surface, of a line from v1 to v2.

This smoothing process yields a face-point curve that is a sampled representation of the projection of a spline curve on the polygon mesh; the spline determines how much the constructed face-point curve will be smoothed. Since both the painting and smoothing processes use local graph search techniques, they are efficient.

Figures 12b and 12c show two examples of complex sets of curves painted on the object of figure 12a. Each configuration has about 220 curves and took about 2 hours to specify; most of that time was spent actively painting the curves. The next section describes how tensor product B-spline surfaces are fit to each of the delineated patches.

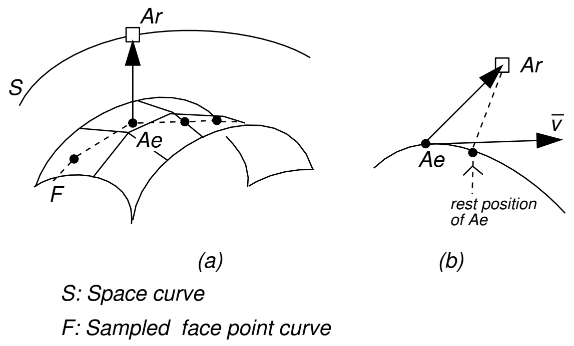

Figure 4. Sliding a face-point curve along a polygon mesh to smooth it. (a) shows an attractor Ar on the space curve and an attractee Ae on the polygon mesh. (b) shows a 1-D version and the rest position of Ae.

4 Fitting B-spline patches

4.1 B-spline fitting theory: overview

In general, parametric curve and surface fitting to irregular data can be formulated as a non-linear least squares problem [10, 30]. The following discussion assumes uniform cubic (order 4) tensor product B-spline surfaces but holds for other kinds of parametric surfaces as well. The equation for a B-spline surface $\vec{P}(u, v)$ can be written as:

$$\vec{P}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} X_{i,j} B_i(u) B_j(v) \tag{1}$$

where $\vec{P}$ is a point in 3-space, u and v are parameter values in the two parametric directions of the surface, $X_{i,j}$'s are control points of the B-spline surface and the B_i are fourth order B-spline basis functions [4] .

Given some set of points $\{ \vec{p_l}$(x, y, z), l=1 ...M$\}$ to which a B-spline surface must be fit, we must first make an association of parameter values u and v to each of these data points. Given these associations, an over-constrained system of linear equations can be generated from (1), where the $X_{i,j}$ are unknowns. Each linear equation in this system corresponds to a point $\vec{p_l}$ satisfying (1). Therefore, a least squares solution [19] can be performed to obtain the $X_{i,j}$.

In our application, we are not given parameter associations for our data points. Because the B-spline basis functions are non-linear in the parameter values, the problem of parametric surface fitting requires a non-linear optimization process. This is usually solved by starting with an initial guess for each p_l's u and v values and subsequently iterating between refining these values and re-fitting the surface with the improved parameter associations until some tolerance of fit is achieved [30]. This process is expensive since it requires calculation of spline surface partial derivatives at each of the original data points at every step of the iteration. Furthermore, the convergence of this iteration (and hence the quality of the fitted spline surface) is strongly dependant on the initial parameter values. If these are not good, convergence can be slow and in general is not guaranteed [18, 20].

4.2 Our surface fitting strategy

To avoid the complexity and cost of the non-linear optimization process described above, we first resample each irregular polygonal patch into a regular grid of points (the spring mesh). We can then apply gridded data fitting techniques [29] to this spring mesh to obtain a spline approximation. The advantage of these techniques is that they avoid the parameter re-estimation step described earlier and are hence significantly faster. It is worth pointing out that in our application there is nothing sacrosanct about the original mesh vertices. In particular, the vertices produced by our range image integration method [9] are at a scale that approaches the noise-limited resolution of our sensor. As long as the grid is a reasonably careful sampling of the polygon mesh, surface quality is not compromised. We use a piecewise linear reconstruction for this sampling, which we find to be satisfactory. If in other applications it is required to fit the original mesh vertices, this can be accomplished by first parameterizing the mesh vertices using our regular spring grid and then running the standard non-linear optimization process described above.

The following subsections describe how to perform a gridded resampling of each polygonal patch and discuss some of its advantages and drawbacks. For the discussion, patches are assumed to be four sided; cylindrical, toroidal and triangular patches are all modeled as special cases of four-sided patches.

4.3 Gridded resampling of each polygonal patch

Each polygonal patch can be an arbitrary four-sided region of the polygon mesh. The only constraints are that it must be rectangularly parameterizable and must not have holes. These are reasonable assumptions since the models input to our system are seamless or can easily be made so by acquiring and integrating more scans and by recent hole-filling techniques [9]. Our goal is to generate a uniform grid of points over the polygonal surface that samples the surface well. Finite element literature [34] describes a number of techniques for generating grids over smooth surfaces. Unfortunately, these techniques rely on the existence of higher order global derivatives (i.e. a smooth surface definition already exists). While it is possible to make local approximations to surface curvature for irregular polygonal surfaces [37], there is no scheme to evaluate global derivatives at arbitrary surface positions.

4.4 What is a good gridding of a surface?

Although we cannot utilize the finite element literature directly, it offers useful insight on objective functions one might minimize to produce different surface parameterizations.

Since each polygonal patch is resampled into a regular grid in order to fit a smooth surface, it is important that the grid not lose any geometric detail present in the original data. We have chosen three criteria for our surface grids: (In the following, a grid line along either direction is referred to as an iso-curve; the two directions are called u and v.)

1) *Arc length uniformity*: the grid spacing along a particular iso-curve should be uniform.
2) *Aspect ratio uniformity*: the grid spacing along a u iso-curve should be the same as the grid spacing along a v iso-curve.
3) *Parametric fairness*: Every u and v iso-curve should be of the minimum possible length given the first two criteria.

An obvious criterion we have omitted above is that iso-curves should always lie within the polygonal patch they are supposed to sample.

Our intuitions for the above criteria are based on sampling theory. Since our triangulations come from real models that have been sampled uniformly over their surfaces, our triangle meshes tend to be uniformly dense across different parts of the polygonal model. For a resampling of such a mesh to be faithful, it should give equal importance to equal areas of the surface.

With this intuition in mind, the arc-length criterion accounts for the fact that geometrically interesting detail is equally likely along any given section of an iso-curve on the surface. The aspect-ratio criterion captures the fact that detail is equally likely in either direction of a gridding. Finally, the parametric fairness term, minimizes "wiggles" along the spring iso-curves. This is important since

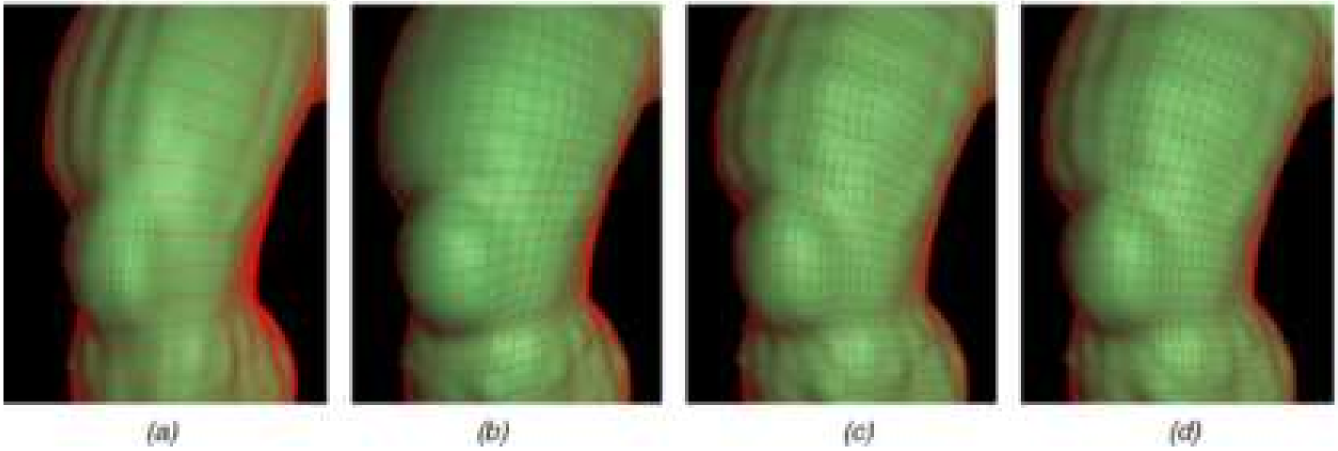

Figure 5. This figure explores the three sampling criteria on part of the right leg of the model in figure 12a. Each of the above images represents a triangulated and smooth shaded spring mesh at a very low resolution. In each case, the number of spring points sampling the polygon mesh was kept the same. The differences arise from their redistribution over the surface. The spring edges are shown in red. (a) shows what happens when the aspect ratio criterion is left out. Notice how a lot of detail is captured in the vertical direction, but not in the horizontal. (b) shows the effect of leaving out the arc length criterion. Notice how the kneecap looks slightly bloated and that detail above and around the knee region is missed. This is because few samples were distributed over the knee resulting in a bad sampling of this region. (c) shows a missing fairness criterion. The iso-curves exhibit many "wiggles". Finally (d) shows the result when all three criteria are met. See figure 8a for the original model and 8e for a full resampling of the leg.

the spline surface follows the spring mesh closely. The fairness criterion thus indirectly minimizes unnecessary wiggles in the spline iso-curves. Note that this term bears some similarities to the idea of fairness in the parametric interpolation literature [16].

4.5 A fast gridding algorithm

Our algorithm is a coarse-to-fine procedure that for each polygonal patch, incrementally builds a successively refined sampling of the patch into a spring mesh. At each subdivision level, the spring mesh points included in the procedure are a subset of face points of the polygonal patch. Here are the steps of the gridding algorithm:

1) Perform a seed fill of the patch. This restricts graph searches to vertices of this patch only.
2) Compute an initial guess for the iso-curves using Dijkstra's shortest-path algorithm, with an appropriate aspect ratio for this patch.
3) Refine the spring mesh using the arc length and fairness criteria.
4) Subdivide the spring mesh.
5) Iterate between steps 3 and 4 until the number of spring mesh points reaches a density close to that of the number of vertices of the polygonal patch.

In our system, the user can stop the resampling at any stage to view incremental results and fit a spline surface to the spring mesh points at the current resolution. We consider some of these steps in detail in the following subsections.

4.6 Initialization of iso-curves

To obtain an initial guess for each u and v iso-curve, we use Dijkstra's single-source, single-destination, shortest-path algorithm [1] to compute a path between corresponding pairs of points along opposing boundary curves. The initial number of iso-curves in each direction are chosen to be proportional to the aspect ratio of the patch. This is computed as the ratio of the longer of the two boundary curves in either direction.

The starting spring mesh points are computed as intersections of these initial iso-curves; the curves must intersect if the patch is rectangularly parameterizable. Dijkstra's algorithm is $O(n \log(n))$ in the number of vertices to be searched. However, since the vertex set is restricted to that of a single patch and we search for only a small set of initial iso-curves, this procedure is rapid. Starting with a large number of iso-curves is both slower and not guaranteed to produce as good a final spring mesh as starting with a small number of iso-curves and using a coarse-to-fine refinement. We return to this point in section 4.10.

4.7 Refining the spring mesh: relaxation

The initial guess for the spring mesh, as obtained above, can be quite poor. The next step in the algorithm is to refine the position of the spring mesh points using a relaxation procedure. In our choice of the number of spring mesh points to place along each boundary curve, we have implemented criteria 2 (section 4.4): aspect ratio uniformity. Subsequent subdivisions are all uniform. During our relaxation procedure, we apply the remaining two criteria of arc length and fairness.

The relaxation procedure works as follows. Let P_{up}, P_{down}, P_{left} and P_{right} represent the positions of the 4 neighboring spring points in the u and v directions of the spring point P. The algorithm computes a resultant force on each of these points and slides it along the surface to a new position.

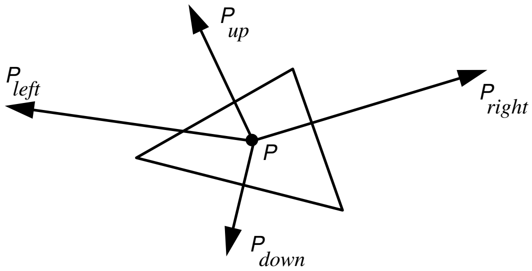

Figure 6. shows the neighbors of a face point P of the spring mesh. The resultant force in a relaxation step is some linear combination of these forces. See the text for details.

Minimizing arc length distortion along one of P's iso-curves is achieved by moving P towards whichever neighbor (on the same iso-curve) is farther away from it. Consider the two forces *Force*(P_{up}, P) and *Force*(P_{down}, P). We make the direction of the larger of these two the direction of a new force F_{ud}. The magnitude of F_{ud} is set to be the difference of the two magnitudes. We perform a similar computation in the other direction (left-right) as well to get a force F_{lr}. Let us denote the resultant of F_{lr} and F_{ud} by F_{arc}. This resultant becomes one of the two terms in equation (2) below.

Fairness distortion is minimized by moving the point P to a position that minimizes the energy corresponding to the set of springs consisting of P and P's immediate neighbors along both iso-curves. This corresponds to computing a force F_{fair} equal to the resultant of the forces acting on P by its four neighbors: *Force*(P_{up}, P), *Force*(P_{down}, P), *Force*(P_{left}, P) and *Force*(P_{right}, P).

The point P is moved according to a force given by a weighted sum of F_{fair} and F_{arc}:

$$F_{result} = \alpha * F_{fair} + \beta * F_{arc} \tag{2}$$

The relaxation iteration starts with $\alpha = 0$ and $\beta = 1$ and ends with $\alpha = 1, \beta = 0$. This strategy has proved to produce satisfactory results.

Note that we have used Euclidean forces in the previous step, i.e. forces that represent the vector joining two spring points. A relaxation step based on Euclidean forces alone is fast but not guaranteed to generate good results in all cases. Figure 7a shows an example where Euclidean forces alone fail to produce the desired effect.

In contrast to Euclidean forces, geodesic forces are forces along the surface of the mesh. These would produce the correct motion for the spring points in the above case. One approach to solving the problem exemplified by Figure 7a, would be to use geodesic forces, or approximations thereof, as substitutes for Euclidean forces in the relaxation step. However this is an expensive proposition since the fastest algorithm for point to point geodesics is $O(n^2)$ in the size of the patch [7]. Even approximations to geodesics such as local graph searches are $O(n)$ and would be too expensive to perform at every relaxation step.

A solution to the problem is motivated by figure 7b; create a new spring point $P_{mid-point}$ that lies on the surface halfway between $P1$ and $P2$. This point generates new Euclidean forces acting on the original points, moving them towards each other on the surface. We call this process spring mesh subdivision.

4.8 Subdividing the spring mesh

Spring mesh subdivision is based on a graph search and refinement algorithm. Given two spring points $P1$ and $P2$ our algorithm computes a new face point P that is the mid-point of the two spring points and that lies on the graph represented by the patch. The procedure is:

1.) Find the two closest vertices v1 and v2 on $P1$ and $P2$'s faces.
2.) Compute a breadth first graph path from v1 to v2. The mid-point of this path serves as a first approximation to P's location.
3.) Refine this location by letting the forces given by *Force*($P1$, P) and *Force*($P2$, P) act on P, moving it to a new position on the surface. This process is distinct from the relaxation process. It is used only to obtain a better approximation for P.

Subdivision along boundary curves is based on a static resampling of the face point curve representation; these points are never moved during the relaxation and subdivision steps. We terminate subdivision when the number of spring points increases to within a certain range of the polygonal patch's vertices.

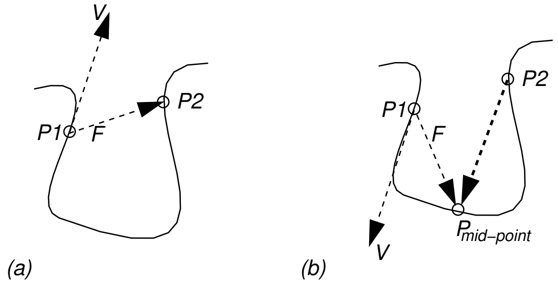

Figure 7. shows a case where relaxation alone fails to move a spring mesh point in the desired direction. In each case F represents the force on $P1$ from its right neighbor and V represents the resulting direction of motion. The desired motion of the point $P1$ is into the cavity. In (a) just the opposite occurs; the points move apart. (b) shows how this case is handled by subdividing the spring mesh along the surface. See the text for details.

4.9 B-spline fitting to gridded data

The techniques described above minimize parametric distortion in the spring mesh. In particular, they enforce minimal distortion with respect to aspect ratio and edge lengths while ensuring parametric fairness. The resulting spring meshes have low area distortion as well, as evidenced by the example shown in figure 5.

The final step in our algorithm is to perform an unconstrained, gridded data fit of a B-spline surface to each spring mesh. As pointed out earlier, fitting to a good resampling of the data does not compromise surface quality. We refer the reader to [29] for an excellent tutorial on the subject of gridded data fitting. Figure 8 summarizes the sampling and fitting processes on a cylindrical patch of the model from figure 12.

4.10 Discussion

Our two-step approach of gridding and then fitting has several desirable characteristics. First, it is fast. This can be understood as follows. At each level of subdivision, each spring mesh point must traverse some fraction of the polygons as it relaxes. The cost of this relaxation thus depends linearly on size of the polygon mesh. It obviously also depends on the size of the spring mesh. If these two were equal, as would occur if we immediately subdivided the spring mesh to the finest level, then the cost of running the relaxation would be $O(n^2)$. If, however, we employ the coarse-to-fine strategy described in the foregoing sections, then at each subdivision level, four times as many spring mesh points move as on the previous (coarser) level, but they move on average half as far. Thus, the cost of relaxation at each subdivision level is linear in the number of spring mesh points, and the total cost due to relaxation is $O(n \log n)$. This argument breaks down if we start with a large initial set of iso-curves. Similar arguments apply to the cost of subdivision.

A second advantage of our overall strategy is that it allows a user to pause the iteration at an intermediate stage and still obtain good quality previews of the model. This is a useful property for an interactive system, specially when dealing with large meshes. In particular, subdivision to higher levels can be postponed until the model designer is satisfied with a patch configuration.

A third advantage of our approach is that once the resampling is done, the spline resolution can be changed interactively, since no further parameter re-estimation is necessary. We have found this to be a useful interactive tool for a modeler, specially when making the tradeoff between explicitly represented geometry and displacement mapped detail as explained in section 5.

As mentioned earlier, there are other schemes that may be used to parameterize irregular polygon meshes. In particular, the harmonic maps of Eck et al[11] produce parameterizations with low edge and aspect ratio distortions. However, the scheme has two main draw-

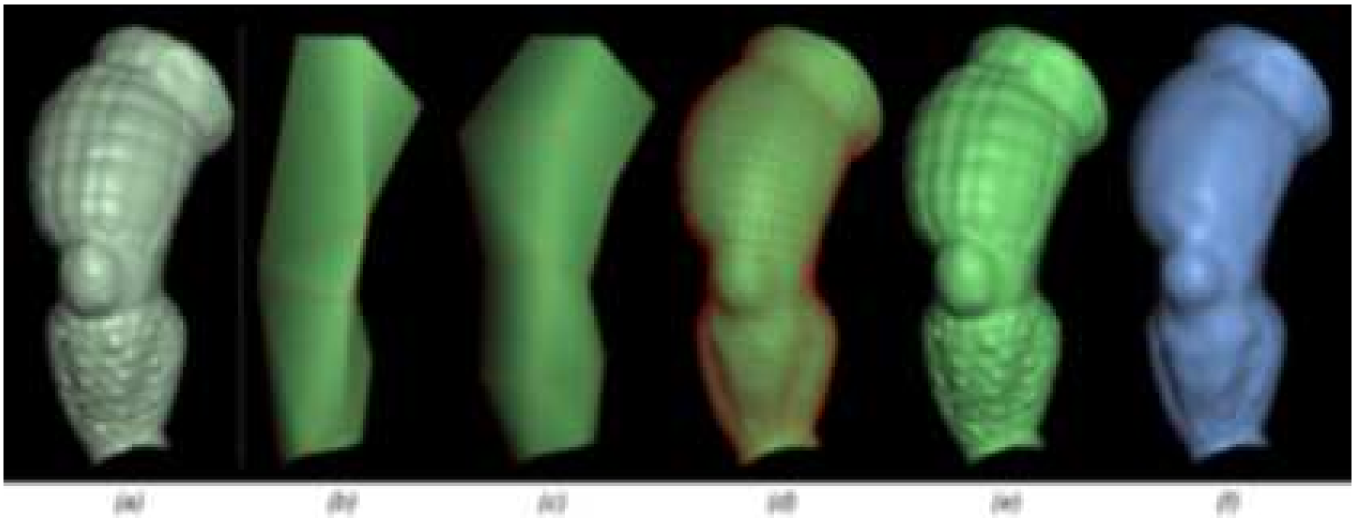

Figure 8. The above represents a summary of our strategy for resampling a polygonal patch into a regular grid. (a) shows the original polygonal patch (the right leg from the model in figure 12a. This particular patch is cylindrical and has over 25000 vertices. (b), (c), (d) and (e)show a triangulated and smooth shaded reconstruction of the spring mesh at various stages of our re-sampling algorithm. We omit the lines from (e) to prevent clutter. (b) shows the initial guess for u and v iso-curves (under 4 seconds). Notice that the guess is of a poor quality. (c) shows the mesh after the first relaxation step (under 1 second). (d) shows the spring mesh at an intermediate stage, after a few relaxation and subdivision steps (under 3 seconds). (e) shows the final spring mesh without the spring iso-curves. Notice how the fine detail on the leg was accurately captured by the resampled grid. This resampling took 23 seconds. All times are on a 250 Mhz Mips R4400 processor. (f) shows a spline fit that captures the coarse geometry of the patch. This surface has 27x36 control points. It took under 1 second to perform a gridded data fit to the spring mesh of (e).

backs for our purposes. First, it can cause excessive area distortions in parameter space. Second, the algorithm employs an $O(n^2)$ iteration to generate final parameter values for vertices of the mesh and no usable intermediate parameterizations are produced. As pointed out in the discussion above, we have found intermediate parameterizations useful in an interactive system.

Our fitting technique is capable of capturing both fine and coarse geometry. However, we typically use it only to capture the coarse geometry. Consider for example the polygonal patch from figure 8a. We find that most of its coarse geometry has been captured by the spline surface of figure 8e. Although the remaining surface detail might be of little use to an animator, it is desirable to retain and use this information as well, if only for rendering.

While there are a variety of multi-resolution techniques that can be applied to capture these details in a unified surface representation [13, 15], for reasons discussed earlier, we represent the fine detail in our models as displacement maps. In the next section we first describe how to extract displacement maps from a polygonal patch and then demonstrate some of the operations that are enabled by this representation.

5 Capturing fine detail with displacement maps

A *displacement map* perturbs the position of a surface based on a displacement function defined over the surface [8]. Displacement maps are usually applied during rendering and are available in a number of commercial renderers. A typical formulation for a bivariate parametric surface, such as a B-spline is: given a point $P(u, v)$ on the surface, a displacement map is a function $\mathrm{d}(u, v)$ giving a perturbation of the point P in space. In general d can be a vector or a scalar. In the first case, the new position of the point is $P + \vec{\mathrm{d}}$. In the second case, the new position of the point is usually interpreted as $P + \hat{N}\mathrm{d}$, where $\hat{N}$ represents the surface normal at (u, v).

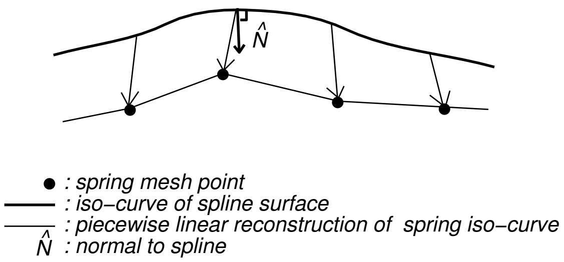

Figure 9. A vector displacement map over a curve. Displacement vectors are shown from an iso-curve S of the spline surface to an iso-curve P of the spring mesh. This map recreates the spring mesh (with a bilinear reconstruction).

5.1 Vector displacement maps

In the context of our fitting system, the obvious displacement function relates points on the spline surface to points on triangles of the original polygon mesh. However, computing such a function requires projecting perpendicularly from the spline surface to the original unparameterized mesh - an expensive operation. Furthermore, our fitting procedure is premised on the assumption that the spring mesh is a faithful representation of the original mesh. Therefore, we define a displacement function that relates points on the spline surface to points on the parameterized spring mesh surface.

Even given this simplification, computing a displacement function using perpendicular projection is difficult. In particular, the method may fail if the spring mesh curves sharply away from the spline surface. We can avoid this difficulty by defining displacements as offsets to vertices of the spring mesh from corresponding points on the spline surface. Recall that there is a natural association of the spring mesh points to parameter values: these are the same parameters that were used for the surface fitting step. We thus obtain a regular grid of displacement vectors at the resolution of the spring mesh. These are represented in the local coordinate frame of the spline surface. For applications that modify the underlying surface (such as animation), this choice of coordinate frame allows the displacements to move and deform relative to the surface. The displacement function $\mathbf{d}(u, v)$ is then given by a reconstruction from this grid of samples. We have used a bilinear filter for the images shown in this paper.

Note that the displacement map is essentially a resampled error function since it represents the difference of the positions of the spring points from the spline surface.

Since the displacement map, as computed, is a regular grid of 3-vectors, it can conveniently be represented as an rgb image. This representation permits the use of a variety of image-processing operations such as painting, compression, scaling and compositing to manipulate fine surface detail. Figure 11 and Figure 13 explore these and other games one can play with displacement maps.

5.2 Scalar displacement maps

While vector displacement maps are useful for a variety of effects, some operations such as (displacement image) painting are more intuitive on grayscale images. There are several methods of arriving at a scalar displacement image. One method is to compute a normal offset from the spline surface to the spring mesh. However, as discussed earlier, this method is both expensive and prone to non-robustness in the presence of high curvature in the spring mesh.

Instead we have used two alternative formulations. The first computes and stores at each sample location (or pixel) the magnitude of the corresponding vector displacement. In this case, modifying the scalar image scales the associated vector displacements along their existing directions. A second alternative, stores at each sample location the component of the displacement vector normal to the spline surface. Modifying the scalar image therfore changes only the normal component of the vector displacement.

Both of these last two options offer different interactions with the displacement map. Figure 11 employs the third option - normal component editing.

5.3 Bump maps

A *bump map* is defined as a function that performs perturbations on the direction of the surface normal before using it in lighting calculations [5]. In general, a bump map is less expensive to render than a displacement map since it does not change the geometry (and occlusion properties) within a scene but instead changes only the shading. Bump maps can achieve visual effects similar to displacement maps except in situations where the cues provided by displaced geometry become evident such as along silhouette edges. We compute and store bump maps using techniques very similar to those used for displacement maps; at each sample location instead of storing the displacement we store the normal of the corresponding spring mesh point. Figures 13d and e show a comparison of a displacement mapped spline and a bump mapped spline, both of which are based on the same underlying spring mesh. Notice how, differences are visible at the silhouette edges.

6 Continuity across patch boundaries

Thus far we have addressed the sampling and fitting issues connected with a single polygonal patch. In the presence of multiple patches, we are faced with the problem of keeping patches continuous across shared boundaries and corners. If displacement maps are used, it is essential to keep the displacement mapped spline surface continuous.

The extent of inter-patch continuity desired in a multi-patch B-spline (or more generally, NURBS) model depends on the domain of application. For example, in the construction of the exterior of a car body, curvature plots and reflection lines [14] are frequently used to verify the quality of surfaces. In this context, even curvature continuous (C^2) surfaces might be inadequate. Furthermore, workers in the automotive industry often use trimmed NURBS and do not necessarily match knot lines at shared patch boundaries during model construction. Therefore it is not always possible to enforce mathematical continuity across patch boundaries. Instead, statistical or visual continuity is enforced based on user specified tolerances for position and normal deviation. These are either enforced as linear constraints to the fitting process [2, 33], or they are achieved through an iterative optimization process [22].

In the animation industry, by contrast, curvature continuity is seldom required and tangent continuity (G^1) usually suffices. To achieve this the number of knots are usually forced to be the same for patches sharing a boundary. This has several advantages. In the first place, control point deformations are easily propagated across patch boundaries. Secondly, there is minimal distortion at boundaries in the process of texture mapping. Finally, the process of maintaining patch continuity during an animation becomes easier; continuity is either made part of the model definition [24] or is re-established on a frame by frame basis using a stitching post-process.

Our system can be adapted to either of the above paradigms (i.e. either statistical or geometric continuity). Since our focus in this paper is on the entertainment industry, we enforce geometrical continuity, and for this purpose we use a stitching post-process. Specifically, we allow an animator to specify the level of continuity required for each boundary curve (C^0 or G^1). Unconstrained, gridded data fitting to each patch leaves us with C^{-1} spline boundaries. We use end-point interpolating, uniform, cubic B-splines. To maintain mathematical continuity we constrain adjacent patches to have the same number of control points along a shared boundary. Following boundary conditions for these surfaces as defined by Barsky [3], C^0 continuity across a shared boundary curve is obtained by averaging end control points between adjacent patches. Alternatively, G^1 continuity can be obtained by modifying the end control points such that the tangent control points (last two rows) “line up” in a fixed ratio over the length of the boundary.

Patch corners pose a harder problem. We refer the reader to [27] for a detailed account of this problem. For the kinds of basis functions we use, projecting the 4 corner control points of each of the patches meeting at that corner to an average plane guarantees G^1 continuity.

For the case of displacement mapped splines, continuity may be defined on the basis of the reconstruction filter used for the displacement maps. Recall that we generate these from the spring meshes and that we use bilinear reconstruction. Displacement mapped splines will therefore exactly recreate the spring mesh. Also adjacent patches can at most be position continuous with a bilinear reconstruction filter. Therefore, if the spring resolutions are the same at a shared boundary of two patches, they will be continuous by virtue of the reconstruction. However, spring mesh resolutions can differ across shared boundaries. This can result in occasional T-joint discontinuities. The problem is solved by averaging bordering rows of displacement maps in adjacent patches. This ensures that there is no cracking at patch boundaries.

Figure 12 shows a case study of the use of our system to fit spline surfaces, with associated displacement maps, to a large and detailed polygonal mesh of an action figure.

7 Conclusions and future work

In conclusion, we have presented fast techniques for fitting smooth parametric surfaces, in particular tensor product B-splines, to dense, irregular polygon meshes. A useful feature of our system is that it allows incremental previewing of large patch configurations. This feature is enabled by our coarse-to-fine gridded resampling scheme and proves invaluable when modelers wish to experiment with different patch configurations of the same model. We also provide a useful method for storing and manipulating fine surface detail in the form of displacement map images. We have found that this representation empowers users to manipulate geometry using tools outside our modeling system.

Our system has several limitations. First, because it relies on surface walking strategies and mesh connectivity to resample polygonal patches, it breaks down in the presence of holes in the polygon mesh. However, new range image integration techniques include methods for filling holes.

Another limitation is that B-spline surface patches, our choice to represent coarse geometry, perform poorly for very complex surfaces such as draped cloth. B-splines have other disadvantages as well, such as the inability to model triangular patches without excessive parametric distortion. Despite these limitations, B-splines (and NURBS in general) are widely used in the modeling industry. This has been our motivation for choosing this over other representations.

There are a number of fruitful directions for future research. Straightforward extensions include developing tools to assist in boundary curve painting and editing, improving robustness in the presence of holes, adding further constraints to the the parameterization process, allowing variable knot density at the fitting stage, implementing other continuity solutions, and using adaptive spring grids for sampling decimated meshes. An example of a boundary painting tool is a "geometry-snapping" brush that attaches curves to features as the user draws on the object. Examples of constraints to the parameterization process include interactively placed curve and point "attractors" within a patch.

An interesting application of the parameterization portion of our system is the interactive texture mapping and texture placement [26] for complex polygonal models. Related to this is the possibility of applying procedural texture analysis/synthesis techniques [17] to create synthetic displacement maps from real ones. Using our techniques such maps can be applied to objects of arbitrary topology.

8 Acknowledgements

We thank Brian Curless for his range image integration software, David Addleman and George Dabrowski of Cyberware for the use of a scanner, Lincoln Hu and Christian Rouet of Industrial Light and Magic for advice on the needs of animators, and Pat Hanrahan, Hans Pederson, Bill Lorensen for numerous useful discussions. We also thank the anonymous reviewers for several useful comments. This work was supported by the National Science Foundation under contract CCR-9157767 and Interval Research Corporation.

References

[1] A.V.Aho and J.D.Ullman. *Data structures and algorithms*. Addison-Wesley, 1979.

[2] L Bardis and M Vafiadou. Ship-hull geometry representation with b-spline surface patches. *Computer Aided Design*, 24(4):217–222, 1992.

[3] Brian A. Barsky. End conditions and boundary conditions for uniform b-spline curve and surface representations. *Computers In Industry*, 3, 1982.

[4] Richard Bartels, John Beatty, and Brian Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers, Palo Alto, CA, 1987.

[5] James F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12, pages 286–292, 1978.

[6] J. Vezien Chakib Bennis and G. Iglesias. Piecewise surface flattening for non-distorted texture mapping. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 237–246, July 1991.

[7] Jindong Chen and Yijie Han. Shortest paths on a polyhedron. In *Proc. 6th Annual ACM Symposium on Computational Geometry*, pages 360–369, June 1990.

[8] Robert L. Cook. Shade trees. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 223–231, July 1984.

[9] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, August 1996.

[10] Paul Dierckx. *Curve and Surface Fitting with Splines*. Oxford Science Publications, New York, 1993.

[11] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *Computer Graphics (Proceedings of SIGGRAPH '95)*, pages 173–182, August 1995.

[12] Matthias Eck and Hugues Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *Computer Graphics (Proceedings of SIGGRAPH '96)*, August 1996.

[13] Hugues Hoppe et al. Piecewise smooth surface reconstruction. In *Computer Graphics (Proceedings of SIGGRAPH '94)*, pages 295–302, July 1994.

[14] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1990.

[15] David R. Forsey and Richard H. Bartels. Hierarchical B-spline refinement. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 205–212, August 1988.

[16] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 35–44, August 1993.

[17] David Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 229–237, July 1995.

[18] J Hoschek and D Lasser. *Fundamentals of Computer Aided Geometric Design*. AK Peters, Wellesley, 1993.

[19] Charles L. Lawson and Richard J. Hanson. *Solving Least Square Problems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

[20] W. Ma and J P Kruth. Parameterization of randomly measured points for least squares fitting of b-spline curves and surfaces. *Computer Aided Design*, 27(9):663–675, 1995.

[21] Jerome Maillot. Interactive texture mapping. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 27–34, July 1993.

[22] M. J. Milroy, C. Bradley, G. W. Vickers, and D. J. Weir. G1 continuity of b-spline surface patches in reverse engineering. *Computer-Aided Design*, 27:471–478, 1995.

[23] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647–668, 1987.

[24] Eben Ostby. Describing free-form 3d surfaces for animation. In *Workshop on Interactive 3D Graphics*, pages 251–258, 1986.

[25] Hans K. Pedersen. Decorating implicit surfaces. In *Computer Graphics (Proceedings of SIGGRAPH '95)*, pages 291–300, August 1995.

[26] Hans K. Pedersen. A framework for interactive texturing on curved surfaces. In *Computer Graphics (Proceedings of SIGGRAPH '96)*, August 1996.

[27] Jorg Peters. *Fitting smooth parametric surfaces to 3D data*. Ph.d. thesis, Univ. of Wisconsin-Madison, 1990.

[28] William T. Reeves. Simple and complex facial animation: Case studies. In *State Of The Art In Facial Animation, SIGGRAPH course 26*, pages 90–106. 1990.

[29] David R.Forsey and Richard H. Bartels. Surface fitting with hierarchical splines. In *Topics in the Construction, Manipulation, and Assessment of Spline Surfaces, SIGGRAPH course 25*, pages 7–0–7–14. 1991.

[30] D. F. Rogers and N. G. Fog. Constrained b-spline curve and surface fitting. *Computer Aided Geometric Design*, 21:641–648, December 1989.

[31] Francis J. M. Schmitt, Brian A. Barsky, and Wen hui Du. An adaptive subdivision method for surface-fitting from sampled data. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 179–188, August 1986.

[32] Stan Sclaroff and Alex Pentland. Generalized implicit functions for computer graphics. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 247–250, July 1991.

[33] Sarvajit S. Sinha and Pradeep Seneviratne. Single valuedness, parameterization and approximating 3d surfaces using b-splines. *Geometric Methods in Computer Vision 2*, pages 193–204, 1993.

[34] J. F. Thompson. *The Eagle Papers*. Mississippi State University, P.O. Drawer 6176, Mississippi State, MS 39762.

[35] Greg Turk. Re-tiling polygonal surfaces. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.

[36] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 311–318, July 1994.

[37] William Welch and Andrew Witkin. Free–Form shape design using triangulated surfaces. In *Computer Graphics (Proceedings of SIGGRAPH '94)*, pages 247–256, July 1994.

[38] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, volume 28, pages 237–246, July 1994.

Appendix A

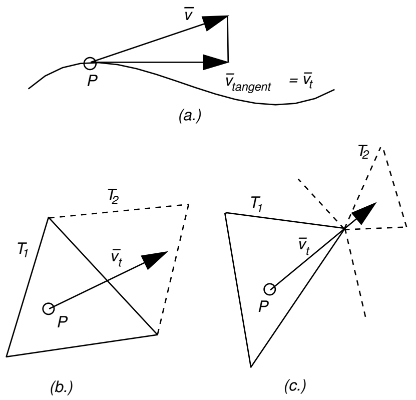

Figure 10. Fig a.) shows a side view of a face point being pulled over the surface. (b) and (c) show a top view of the two cases that arise when P moves: it either intersects an edge or it intersects a vertex.

An operation we use often on face points is sliding them on the surface. We call this procedure *MovePointOnSurface*. There are a number of ways of implementing this on polygonal surfaces. Turk [35] describes a scheme where points first leave the surface and then are re-projected back. We use instead a scheme where points never leave the surface but instead just slide along it. Our algorithm projects the force on a face-point P to P's plane. The point P is moved along the surface, till it either meets an edge or a vertex. In either case we determine the appropriate next triangle to move in to using our adjacency structure (eg: a winged-edge representation).

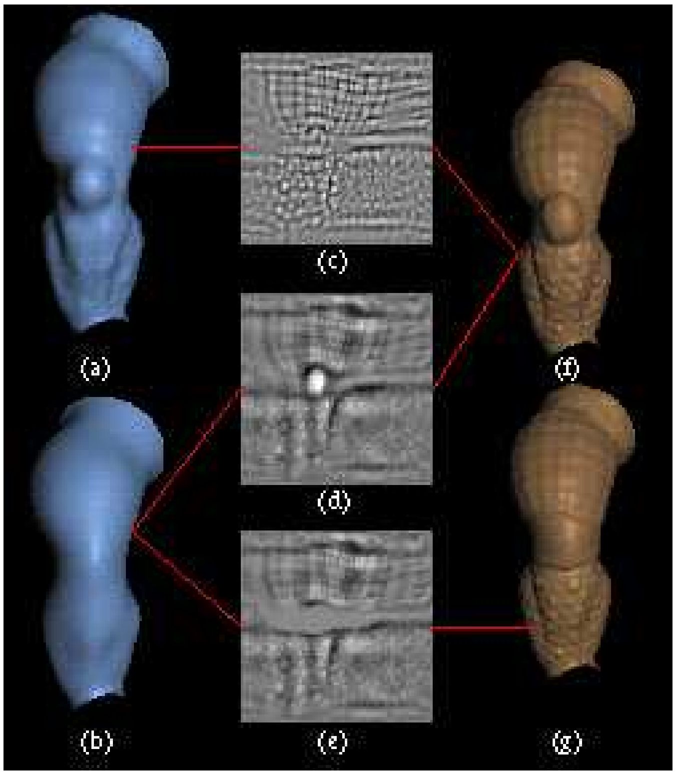

Figure 11. This figure explores the possibility of multi-resolution editing of geometry using multiple displacement map images. All grayscale displacement images in this figure represent the normal component of their corresponding displacement maps. Displacement values are scaled such that a white pixel represents the maximum displacement and black, the minimum displacement. (a) shows a B-spline surface with 24x30 control points that has been fit to the patch from figure 8a. (c) is its corresponding displacement image. (b) shows a B-spline surface with 12x14 control points that was also fit to the same patch. Its displacement image is shown in (d). The combination of spline and displacement map in both cases reconstructs the same surface (i.e. the original spring mesh of figure 8e). This surface is shown in (f). We observe that (c) and (d) encode different frequencies in the original mesh. For example (d) encodes a lot of the coarse geometry of the leg as part of the displacement image (for example the knee), while (c) encodes only the fine geometric detail, such as bumps and creases. As such, the two images allow editing of geometry at different scales. For example, one can edit the geometry of the knee using a simple paint program on (d). In this case, the resulting edited displacement map is shown in (e) and the result of applying this image to the spline of (b) gives us an armour plated knee that is shown in (g). Operations such as these lead us to the issue of whether multiple levels of displacement map can essentially provide a image filter bank for geometry i.e. an alternative multi-resolution surface representation based on images. Note however that the images from (c) and (d) are offsets from different surfaces and the displacements are in different directions, so they cannot be combined using simple arithmetic operations.

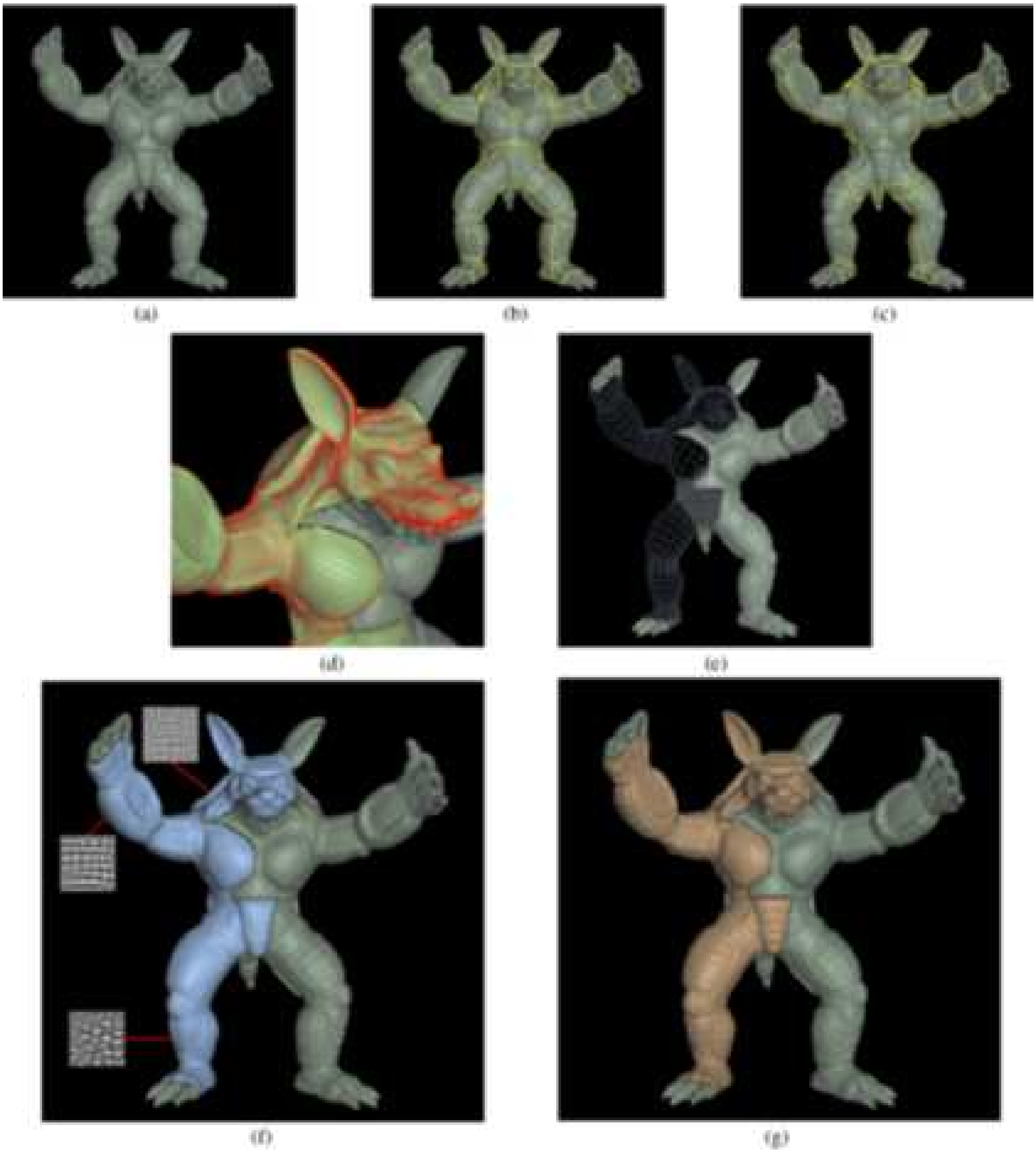

Figure 12. Data fitting to a scanned model. (a) is the polygonal model (over 350,000 polygons, 75 scans). (b) and (c) show two different sets of boundary curves painted on the model. Each was specified interactively in under 2 hours. The patch boundaries for (d), (e), (f) and (g) are taken from (b). (d) is a close up of the results of our gridded resampling algorithm at an intermediate stage. The spring mesh is reconstructed and rendered as triangles and the spring edges are shown as red lines. The right half of the figure is the original polygon mesh. (e) shows u and v iso-curves for all the fitted and stitched spline patches. (The control mesh resolution was chosen to be 8x8 for all the patches.) (f.) shows a split view of the B-spline surfaces smooth shaded on the left with the polygon mesh on the right. A few interesting displacement maps are shown alongside their corresponding patches. (g) shows a split view of the displacement mapped spline patches on the left with the polygon mesh on the right. Note that the fingers and toes of the model were not patched. This is because insufficient data was acquired in the crevices of those regions. This can be easily remedied by using extra scans or hole filling techniques [9]. The total number of patches for (b and d through g) were 104 (only the left half have been shown here). The gridding stage took 8 minutes and the gridded fitting with 8x8 control meshes per patch, took under 10 seconds for the entire set of 104 patches. All timings are on a 250 Mhz MIPS R4400 processor.

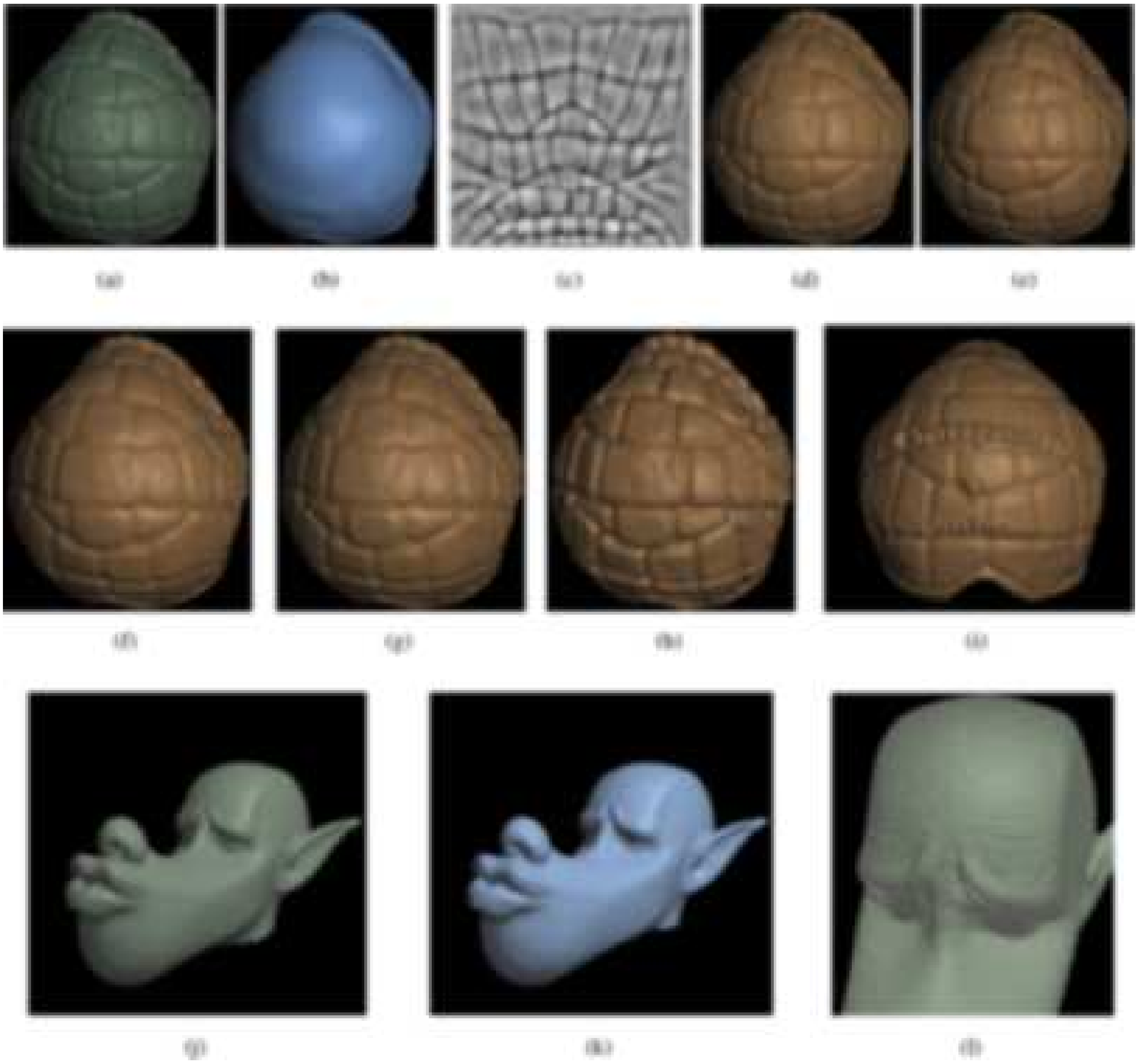

Figure 13. Games one can play with displacement maps: (a) shows a patch from the back of the model in 12a. The patch has over 25,000 vertices. We obtained a spline fit (in 30 seconds) with a 15x20 control mesh, shown in (b) and a corresponding vector displacement map. The normal component of the vector displacement map, is displayed as a grayscale image in (c). (d) and (e) show the corresponding displacement and bump mapped spline surface. The differences between (d) and (e) are evident at the silhouette edges. The second row of images show a selection of image processing games on the displacement map. (f) shows jpeg compression of the displacement image to a factor of 10 and (g) shows compression to a factor of 20. (h) represents a scaling of the displacement image, to enhance bumps. (i) demonstrates a compositing operation, where an image with some words was alpha composited with the displacement map. The result is an embossed effect for the lettering. Finally, the third row of images (j - l) show transferring of displacement maps between different objects. (j) is a relatively small polygonal model of a wolf's head (under 60,000 polygons). It was fit with 54 spline patches in under 4 minutes. The splined model is shown in (k). (l) shows a close up view of a partially splined result, where we have mapped the displacement map from (c) onto each of 4 spline patches around the eyes of the model.