Piecewise Regular Meshes: Construction and Compression

Andrzej Szymczak, Jarek Rossignac and Davis King
GVU Center
Georgia Tech
Atlanta, GA 30332
USA

Version: February 22, 2002

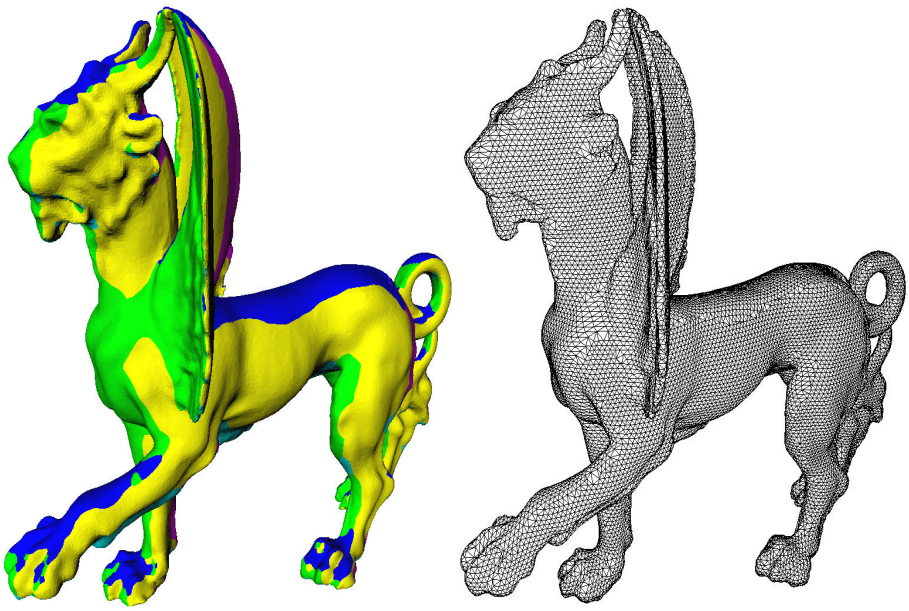

FIG. 1 Relief layout on the feline model and the resulting piecewise regular mesh

We present an algorithm which splits a 3D surface into *reliefs*, relatively flat regions that have smooth boundaries. The surface is then resampled in a regular manner within each of the reliefs. As a result, we obtain a *piecewise regular mesh* (PRM), having a regular structure on large regions. Experimental results show that we are able to approximate the input surface with the mean square error of about $0.01 - 0.02$ per cent of the diameter of the bounding box without increasing the number of vertices. We introduce a compression scheme tailored to work with our remeshed models and show that it is able to compress them losslessly (after quantizing the vertex locations) without significantly increasing the approximation error using about 4 bits per vertex of the resampled model.

1. INTRODUCTION

Surfaces, usually represented as triangle meshes, are the most common objects in graphics and computer aided design. As a result of the increase in computational power

and the advent of 3D scanning technology, the size and accuracy of surfaces used for various applications increases at a very fast rate. Today, models having a hundred thousand vertices are common, and models having billions of vertices and requiring tens of gigabytes to store are becoming available [18]. Although the increase in the network bandwidth, processor speed and availability of fast storage helps to deal with this amount of data, compact surface representation techniques are essential to make this new and exciting technology a success.

Geometric compression, aiming to develop such compact representations, has recently grown into a vast and active research area. The early compression schemes [6, 9, 31, 32, 24, 25] concentrate on compacting the triangle mesh representation of surfaces. Since the uncompressed connectivity typically dominates the storage cost [24], their crucial part is connectivity compression. The connectivity is compressed independently from the vertex data and is used to derive predictors for vertex coordinates. While this approach is particularly simple, easy to implement and fast, it suffers from severe limitations. We think the most important one is the poor performance of the vertex predictors. Technically speaking, without the knowledge of the vertex pattern or some other property of their distribution it is impossible to construct any predictor of higher order (i.e. always being exact for some class of simple surfaces, like planes). If a shape of a triangle T is not known, knowledge of all vertices of an adjacent triangle (and therefore, in particular, two of the T's vertices) cannot help much in predicting the third vertex. Of course, one can attempt to take advantage of the coherence in the 'common' meshes and use a prediction rule like the parallelogram rule [32], but making progress from there in a way applicable to all meshes proves to be a very hard task. Nevertheless, new improved techniques of compressing triangle meshes are being developed [1, 2], as they are regarded to be free from the risks associated with changing the topology, more elementary, faster, easier to implement and free from hard to control artifacts.

The compression schemes mentioned so far work much better than general purpose compression schemes for files representing 3D meshes. However, in terms of rate distortion they are less effective than the wavelet compression scheme [14, 15], which proceeds by first changing the topology of the encoded mesh to semi-regular and then applying techniques developed in the area of image compression (zerotrees and wavelet transforms [29, 26]) to compress the new triangulation. Since the wavelet schemes naturally support transmission of information in the order of importance (in this case, by transmitting the wavelet coefficients bitplane after bitplane, starting from the most significant one), the wavelet scheme is very well suited for progressive transmission. The results presented in [14, 15] in terms of rate-distortion (especially for low bit rates) are quite impressive when compared to other compression schemes [20, 21, 31]. However, the wavelet scheme also has its weaknesses: it may perform poorly for highly complex surfaces concentrating energy in isolated frequency bands (like the Michangelo project models [18]). It is complicated and hard to implement (especially the MAPS remeshing algorithm [17]).

In this paper, we consider the problem of resampling a triangulated surface to improve the performance of compression algorithms. We use an extended farthest point Voronoi diagram [3] to split a triangle mesh into connected regions of similarly oriented triangles with simple, mostly smooth boundaries. We call them *reliefs*. Each of the reliefs is then resampled so that most of the triangles project onto triangles of a regular hexagonal grid. This greatly improves the performance of vertex predictors inside each of the regions and makes the vertex degree distribution more compact, improving the connectivity compression rates. The remeshedregions are joined together to ensure manifold topology of the remeshed surface. Figure 1 shows the layout of patches on the feline model and the triangulation (called a *piecewise regular mesh*) produced by our algorithm. In our experiments, we use resampled surfaces approximating the original model with the mean square error of about .02 per cent of the diameter of the bounding box. Applying the Touma and Gotsman algorithm to the new triangulation already leads to much better results than running it for the original one (the entropy decreases by almost 40 per cent on average

and the number of vertices decreases as well for the typical 12 bit quantization). However, by designing a compression scheme tailored to work with piecewise regular meshes, we are able to achieve about 4 bits per vertex for both connectivity and geometry (over 70 per cent less than [32]). Our compression algorithm uses the Edgebreaker [24] to encode the global connectivity and the triangles which do not belong to the regular regions. The geometry of the regular parts is compressed using an iterated two-dimensional variant of the differential coding [27]. Our procedure is lossless in the sense that we are not introducing any errors to the resampled model other than quantization of the vertex coordinates. Since it does not require global parametrization, it is equally easy to use for models with complex topologies. It is also faster than the combination of the MAPS algorithm [17] and the wavelet mesh compression algorithm of [14, 15], while offering comparable compression rates. The piecewise regular mesh framework (after a few obvious changes) is very well suited for representing and processing meshes obtained with range scans (which often produce large regions covered with regular grids).

We think that an interesting aspect of our results, which has not yet received much formal study, is that of relating the information theoretic mesh optimality measures to the geometric ones. A lot of methods of producing optimum meshes approximating a given one have been proposed (e.g. [7]). They were designed to decrease the number of vertices of the mesh while introducing little error. However, while they do that, the shape of the triangles changes and typically the structure of the mesh becomes less and less regular. This makes the simplified models harder to compress efficiently. In order for the simplified models to be represented using the same amount of space as our piecewise regular meshes, they must have considerably fewer vertices (clearly, this is true only in a certain range of approximation errors: our remeshed models cannot achieve zero error while irregular meshes clearly can). Our experiments show that, in terms of information-theoretic optimality the remeshed models are often better: the simplified models requiring about the same storage as our remeshed ones present worse approximations of the initial model.

The organization of the paper is as follows. In Section 2 we describe the details of our resampling procedure. Section 3 discusses a compression scheme designed to work with piecewise regular meshes. Finally, in Section 4 we describe experimental results.

2. DETAILS OF THE RESAMPLING PROCEDURE

In this section we describe the resampling procedure which allows to transform an irregularly sampled mesh into one covered with large regularly sampled regions. Later we show that the regular structure of the resulting mesh improves the performance of vertex predictors as well as connectivity compression rates.

The resampling procedure can be built as a composition of three operations.

1. **Relief construction.** The input surface is subdivided into reliefs (Figure 1, left). The triangles of the reliefs are required to face the same direction. More precisely, their outward normals need to form an angle less than 90 degrees with some fixed vector which will be called the *defining vector*. Locally, each relief can be thought of as a height field over a plane perpendicular to its defining vector, but projections of several portions of the same relief onto that plane may overlap.
2. **Relief resampling.** Each of the reliefs is resampled over a regular hexagonal grid. The resampled reliefs have the same boundary as the original ones, but their internal triangles project to equilateral triangles of a regular hexagonal grid. The triangles which project to the grid triangles will be called *regular*. Thus, the irregular triangles are distributed around the borders of the reliefs. Figure 4 shows one of the reliefs of the cow model and its resampled version.
3. **Relief merging.** The resampled reliefs are merged together into a single triangle mesh. The layer of the irregular triangles that surround the borders of the reliefs is

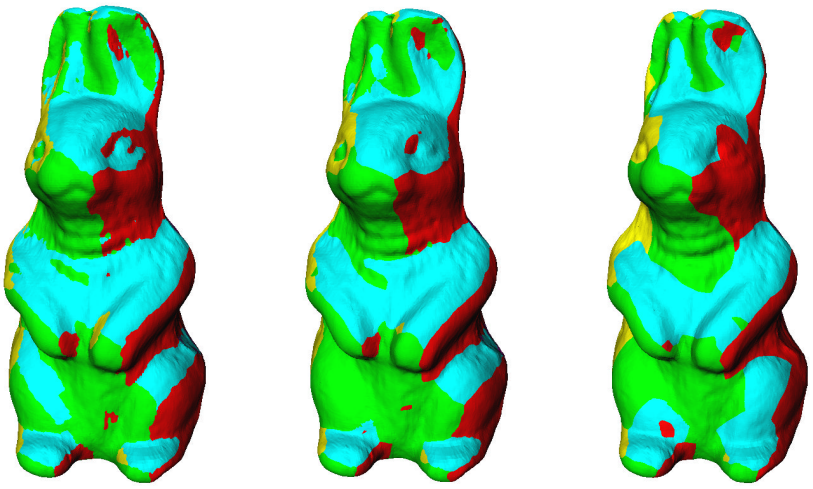

FIG. 2 The effect of the regularization parameter τ on the relief boundaries: $\tau = 0.5773$ (very close to $\sqrt{3}/3$, left), $\tau = 0.4$ and $\tau = 0.1$ (right). Notice that the relief boundaries get smoother and smoother as τ decreases.

faired and simplified. The final result for the feline model can be seen in Figure 1 (right).

In what follows we describe the three stages of our resampling procedure in more detail.

2.1. Relief Construction

The goal of this step is to split the input surface into reliefs. We do this by first selecting the set of defining vectors. After considering several strategies, we opted for the simplest one, the set of 6 vectors pointing along the coordinate axes: $\{[\pm 1, 0, 0], [0, \pm 1, 0], [0, 0, \pm 1]\}$. This choice ensures that the reliefs are rather flat while keeping their number small. Optimization of the choice of the defining vectors may be an interesting topic for future research (cf [22, 13, 8, 4]).

The size and the complexity of the irregular part of the remeshed surface (defined as the union of all irregular triangles) impacts the quality of the remeshed surface and the compression rates. The simple procedure which assigns a triangle to a relief based on the smallest angle between the triangle's normal vector and a defining vector tends to produce reliefs with jaggy boundaries and an unnecessary large number of connected components. Because of this, we developed a procedure which uses the geodesic distance function [19, 11, 16] and a variant of the farthest point Voronoi diagram [3] to obtain reliefs with smooth boundaries. We start by computing sets of triangles which will define the Voronoi cells. For a smoothing parameter $\tau > 0$, each of the defining vectors v has a corresponding set A_v, the union of all triangles whose outward normal vectors form the angle with cosine larger than $-\tau$ with $-v$. The relief whose defining vector is v will be a subset of the *complement* of A_v (this will ensure that all the triangles face the direction of the defining vector). In order for our procedure to work, we need to require that the union of complements of all the A_v sets is the entire surface. Since any vector forms an angle of cosine greater than $1/\sqrt{3}$ with one of our defining vectors, this is equivalent to requiring that $\tau < 1/\sqrt{3}$. Once the A_v sets are available, we compute the function d_v, the geodesic distance function from the set A_v. Thus, $d_v(p)$ is the length of the shortest path joining p

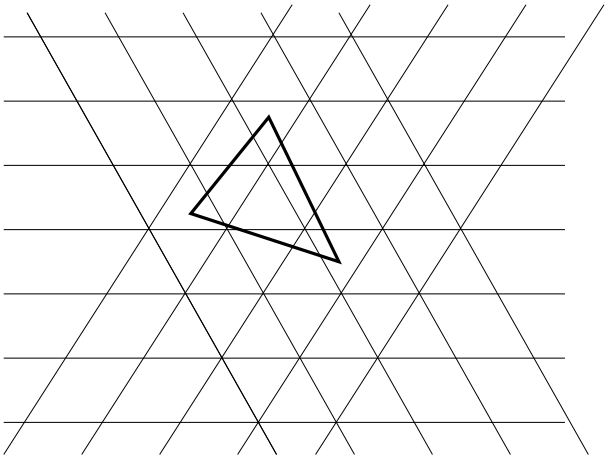

FIG. 3 The hexagonal grid is defined by three families of thin parallel lines. The thick lines bound a projection of a triangle of the original mesh. The projections of the polygons into which the triangle is subdivided are the regions inside the triangle bounded by all the lines.

and a point of A_v. The d_v functions can be approximated using the Dijkstra's algorithm, in a way similar to [11, 16]. Finally, the relief with defining vector v_0 is computed as the set of all points p for which $d_{v_0}(p)$ is the largest of all values $d_v(p)$. In our implementation, the d_v functions are piecewise linear and are represented by values at vertices. Although the A_v sets are unions of the original triangles, the reliefs are not: some triangles need to be subdivided.

In general, a smaller value of τ leads to smoother relief boundaries (which makes the resulting PRM more regular, increases its visual quality and decreases the encoding size) but, at the same time, causes the relief to have a steeper slope in some places and therefore tends to increase the distance between the resampled and original models. We found out experimentally that the values of τ in the range between 0.1 and 0.3 are a good compromise. Figure 2 illustrates the influence that the value of τ has on the smoothness of the reliefs' boundaries.

2.2. Relief Resampling

The goal of this stage is to resample the reliefs over a regular hexagonal grid. The grid size is a parameter which allows to trade the number of vertices in the resampled relief for the approximation accuracy. Our approach is similar to that of [33]. We first refine the mesh so that it contains all vertices of the target mesh. Then, the original vertices are removed by means of edge collapse operations. The rules for the collapses are designed in such a way that the final relief has the same boundary as the initial one and mostly regular triangles in the interior. Although, at the first sight, this approach seems to be more complex than using e.g. the constrained Delaunay triangulation [3], it is able to gracefully deal with the complicated cases like self-overlapping and topologically complex reliefs.

Let v be the defining vector of a relief. Consider a regular hexagonal grid in a plane perpendicular to this vector. The grid is bound by three families of parallel lines each two, intersecting at 60 degrees (see Figure 3). All points in the 3D space which are mapped into lines of one family when projected along v form a set of planes parallel to each other and v. Planes in different families intersect at 60 degrees. In the refinement stage, each

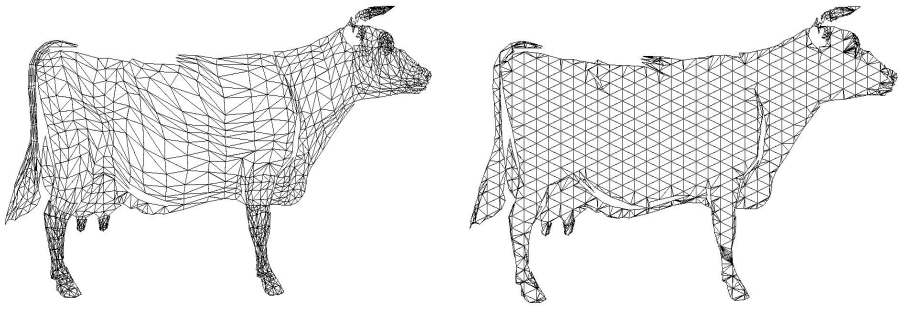

FIG. 4 An original triangulation of a relief of the cow model and its resampled version.

triangle of the original grid is sliced into smaller polygons with these planes. For the ease of representation, the polygons are further subdivided into triangles by means of inserting diagonals. All the triangles obtained this way form the refined relief. Ignoring the numerical errors, it is identical to the original one.

Clearly, the refined relief contains many more vertices than we need. The vertices which do not project to the vertices of the regular grid are removed by means of edge collapse operations. Collapsing an edge ([7, 23]) causes the two triangles of the mesh incident upon it to disappear. Our algorithm moves one of the endpoints of the edge to the location of the other (thus, no sophisticated optimization of the location of the vertex born in the edge collapse operation is needed). While edge collapses are executed, special care must be taken not to change the boundary of the relief (we want the reliefs to fit together tightly in order to be able to perform the merging operation step). We would also like to introduce as many regular (i.e. projecting to the triangles of the regular grid) triangles as possible. To achieve this, we never perform edge collapses which move boundary vertices. Also, the vertices projecting to vertices of the regular grid are not allowed move. To avoid creating irregular triangles in the interiors of the patches, we do not collapse edges whose projections intersect the edges of the regular grid unless they are contained in a single edge. We also prevent triangle flips by disallowing edge collapse operations which would produce triangles whose outward normal vectors form an angle greater than 90 degrees with the relief's defining vector. Figure 4 shows an example of a relief and its resampled version.

2.3. Merging the resampled reliefs

If numerical errors are neglected, all triangles of the resampled reliefs form a manifold surface. However, although the common boundaries of two adjacent reliefs are geometrically identical, their vertices typically do not match. This is because the resampled reliefs have not only the original vertices, but also new ones on their borders.

We would like to obtain a PRM with manifold topology. This requires proper merging of the reliefs' triangulations. A simple to implement approach is to add zero-area triangles stretched along the boundary edges of the reliefs. Consider an external edge pq of one of the reliefs R before the resampling stage. Since the reliefs' boundaries match, it is also an external edge of some other relief R'. In the resampling process, new vertices are introduced on that edge for each two of the reliefs. All the vertices can be arranged in a loop which starts at p, then moves through the new vertices on the boundary of R until q is reached and finally returns to p along the boundary of R' (Figure 5, left). The new vertices on pq in the two reliefs may coincide. In this case, we break the loop into two or

FIG. 5 A loop of vertices (shown as black dots) introduced in the remeshing stage (left). A situation where it has to be broken into two loops (right).

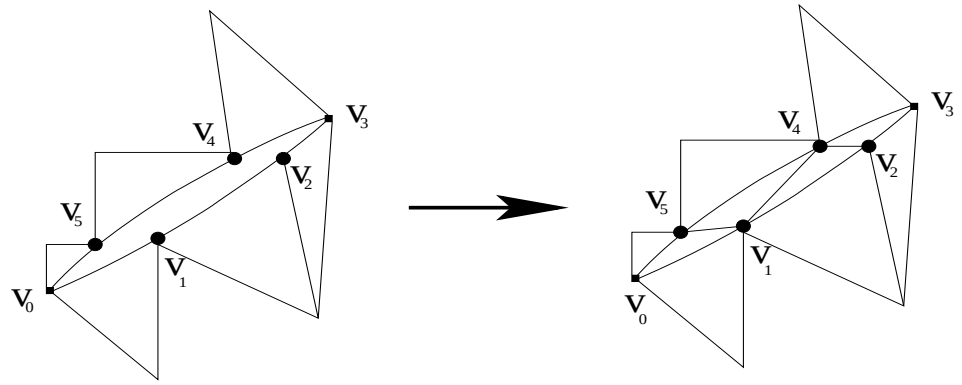

FIG. 6 A loop of boundary edges of neighboring resampled reliefs; v_0 and v_3 are vertices of the original mesh and v_1,v_2,v_4 and v_5 are vertices introduced during the overlay computation. The result of filling the loop with triangles is shown right.

more shorter loops such that each loop passes through a vertex no more than once (Figure 5, right).

Topologicaly speaking, the loops as described above are holes in the mesh. Geometrically, they are always stretched along a boundary edge of an original relief. Each boundary edge of a resampled reliefs belongs to exactly one loop. Now, it remains to fill the loops with triangles. If the loop's vertices are $v_0, v_1, v_2, \ldots, v_N = v_0$ then it can be filled with triangles, e.g. $v_0v_1v_2$, $v_0v_2v_3$, $\ldots$, $v_0v_{N-2}v_{N-1}$ or in some other way. As a result, all edges of the loop become internal (Figure 6): inserting the triangles in the way described above adds one triangle incident to each of the loop's edges and therefore causes them to have exactly two incident triangles.

At this stage, we have a manifold mesh with a large number of vertices on the boundaries of the original reliefs. We further process it to remove as many of these vertices as possible (regular vertices are cheaper to encode). Again, edge collapses turn out to be useful here: we first perform edge collapses moving vertices on relief boundaries to grid vertices. We do not perform edge collapse operations which change the mesh topology. Although this allows to get rid of most the unwanted vertices, it creates mesh of poor quality between the reliefs. Therefore, we optimize it by recursively performing edge flips operations which decrease the surface area on edges which join vertices originating from different relies. This simple procedure turns out to work very well in practice, highly improving the quality of the irregular part of the PRM.

2.4. Final remarks on the Resampling Procedure

Clearly, the resampling algorithm described above needs to use grid sizes considerably smaller than the sizes of small details that need to be preserved, since otherwise it is impossible for them to survive in the resampled mesh. Thus, it generally will not work well for sparsely sampled meshes whose connectivity and geometry are carefully laid out

to model high frequency features (like sharp edges etc). However, we found out that if the input surface is densely sampled, our procedure performs quite well, leading to small approximation errors (see Section 4). The only problem which we observed is occasional flipped triangles in the irregular part, which can easily be removed easily as a post-processing operation. Examples of piecewise regular meshes (also with grid sizes too large to capture detail) are shown in Figure 10.

It is not hard to see that by running our algorithm for sufficiently small grid size one can reach any desired approximation accuracy in the relief resampling procedure. Also, the area outside the regular triangles in each relief tends to zero as the grid size decreases to zero. Although the flipped triangle removal is hard to analyze formally, experimental results show that they are typically rare and sparse, so that their removal is a simple local operation. To sum up, one should expect that the remeshed surfaces converge to the input surface as the grid size tends to zero. More informative bounds on the approximation accuracy can be derived from differential properties of the input surface using the Taylor series approximation.

3. THE COMPRESSED FORMAT

A resampled model consists of two types of triangles: regular and irregular. Regular triangles have vertices in a single relief and are mapped into grid triangles when projected in the direction of its defining vector. All the other triangles are irregular. Our compressed format for a resampled model consists of three major parts:

1. Connectivity encoding

2. Encoding of the irregular triangles

3. Geometry encoding.

Below we discuss our strategy for each of the parts.

3.1. Connectivity encoding

The connectivity of the resampled mesh is typically very regular: the majority of vertices have degree 6. Therefore, we use a valence-based prediction scheme based on the ideas of [30]. In the case of PRMs, our algorithm leads to the same compression rates as [1], while being much faster.

Our scheme is based on the Edgebreaker encoding [24] and the Spirale Reversi decompression algorithm [12]. First, the Edgebreaker algorithm is used to convert the mesh to be encoded into a CLERS string. Then, the CLERS string is turned into a binary string using the context-based range coder [28]. Our context consists of some fixed small number of preceding symbols in the decoding order (we found out that 1 or 2 works best in practice) and the number of triangles in the already decoded part of the mesh incident upon the starting point of the gate (see [30] for details). The context constructed in the above way reduces the uncertainty about the next symbol for a highly regular mesh. For example, whenever the number of triangles incident upon the starting point of the gate is 5, C is likely to occur (since such a C finishes that vertex producing a vertex of degree 6 in the decoded mesh).

3.2. Encoding of the irregular triangles

The purpose of encoding the irregular triangles separately is to increase the efficiency of geometry predictors. Firstly, one can predict better if it is known that all vertices involved in a predictor are on a regular grid. Secondly, in connected regions consisting of only regular triangles, only one coordinate (height) over a base plane, perpendicular to the determining vector, needs to be encoded. This, in a way similar to [10], allows to encode vertex locations as one number per vertex rather than three.

status (discovered or regular?)		action	
Right(h)	Left(h)	symbol output	half-edges pushed (in order)
no	no	S	left(h), right(h)
no	yes	R	left(h)
yes	no	L	right(h)
yes	yes	E	none

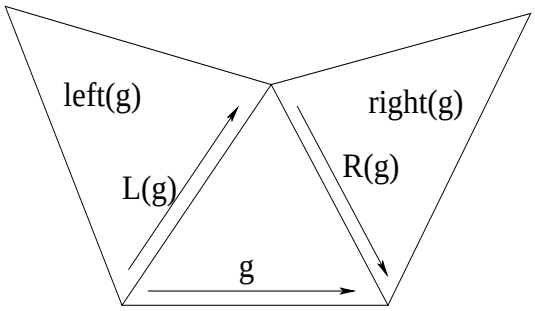

FIG. 7 Encoding the irregular triangles: actions for a given status and notation for a neighborhood of a half-edge of h

The irregular triangles are encoded as mesh traversals. The traversals are represented using four symbols: L,E,R and S, whose meaning is similar as in Edgebreaker. A stack of half-edges is used to control the traversal. First, the initial half-edge (gate) is selected and placed on the stack as the only item. We require that the initial gate bounds an irregular triangle and has a regular one on the other side. Now, until the stack is empty, we pop a half-edge h and, depending on its status (i.e. whether the Left(h) and Right(h) are irregular and not traversed yet, see Figure 7), we generate a symbol in $\{\mathrm{L}, \mathrm{E}, \mathrm{R}, \mathrm{S}\}$ and push one, two or none of the half-edges L(h), R(h) on the stack. The actions taken and symbols generated are listed in Figure 7.

The encoding of the traversal consists of the identifier of the initial gate and the entropy encoding of the LERS sequence (our implementation uses the second order entropy coder). Since the set of all irregular triangles may not be connected, it is possible that several traversals are necessary.

3.3. Geometry encoding

Having the connectivity and the encoding of all irregular triangles, the decoder is able to compute all *regular patches*, i.e. connected components of the set of all regular triangles. All such patches have to be contained in a single relief. The coordinates of vertices within a regular patch are decoupled into components parallel (*height*) and perpendicular (*base*) to the defining vector of that relief and the two are encoded separately. Since all triangles of the patch are known to project to grid triangles, in order to recover the base component it is enough to know base component of one vertex and one of the six ways of aligning one of its incident triangles with the grid. Thus, two (typically small) integers representing the base component in the grid coordinates, the grid size and a number in $\{0, 1, 2, 3, 4, 5\}$ per patch suffice. In order to be able to convert the base and height coordinates to xyz, we also need to encode the relief type (the identifier of its defining vector), as a number

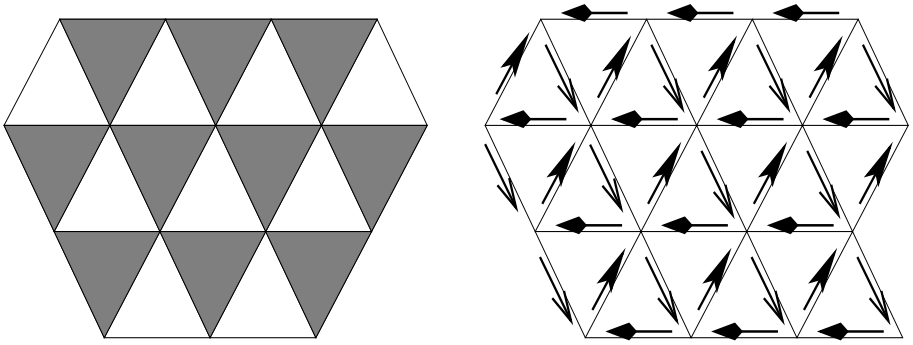

FIG. 8 Left: Odd and even triangles; one are shaded, the other are not. Right: Three groups of oriented edges (shown as arrows of different types).

between 0 and 5.

The heights within a single regular patch will be encoded using a variant of differential coding. As a setup pass, we assign orientations to all edges of the patch and split the directed edges in three groups, each consisting of edges running in the same direction when projected to the base plane (Figure 8, right). Since our patches may be self-overlapping, we discuss the general form of our procedure below.

At this point we know the base component of the coordinates of each vertex. The triangles of the base grid can be split into two groups, which we call *odd* and *even* in such a way that no two in one group are adjacent across an edge (Figure 8,left). This splitting can be carried over to triangles of the regular patch: triangles which project to odd triangles are marked as odd and all the other as even. Now, orient the edges of the even triangles so that they project to clockwise order in the base plane and the even triangles so that they project to counter-clockwise ordering. In this way, we can orient all the edges of the regular patch. The projections of the oriented edges can only be one of three vectors in the base plane. This allows to split them into three groups according to what the projected vector is. Let G_1, G_2 and G_3 be the sets of edges of the same type this splitting. Consider a function f assigning a number to each vertex of the regular patch. Let us assume that differences of the values of f at the endpoint and at the starting point of every edge in one group G_k are available. Then, if the value at a vertex v is known, one can also reconstruct all values at vertices that can be reached from v by following a path following edges in G_k (either backward of forward). The set of such vertices will be called the *region of influence* of v. Clearly, the set of all vertices in the regular patch is a disjoint union of all regions of influence. Our encoding of f proceeds in the following way. First, we send the value of an arbitrary vertex of the patch. Assuming that the differences are available, all vertices in its region of influence can be determined (decoded). Now, we select an edge of the patch joining a decoded vertex to one which is not (such an edge will be called a *bridge*) and the difference of the values at its endpoints are sent. Based on this information, all vertices in the region of influence of the second vertex of the bridge can be decoded. Recursively, we select the next bridge, in the same way as before and keep doing so until enough information is sent to decode all vertices. Thus, the encoding of the function f consists of three parts:

1. The value at the base vertex

2. The encoding of differences of values across all the bridges

3. The encoding of differences across edges of the same type.

The differences in the third part need not to be encoded explicitly: they can be treated as a function defined on another regular (but not necessarily connected) patch, which we

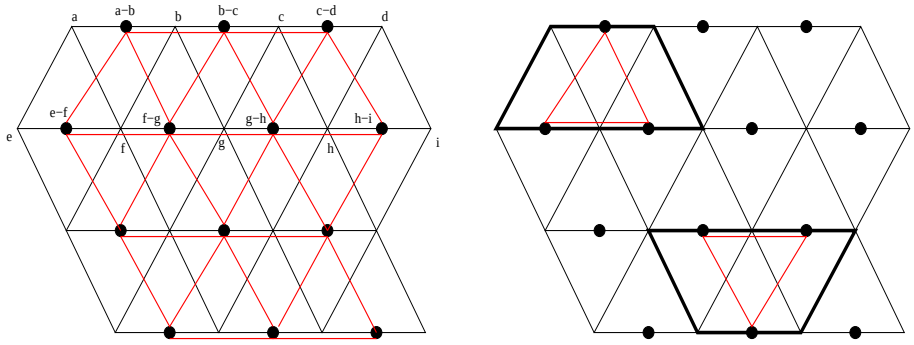

FIG. 9 Left: The original patch (black lines) and the difference patch with respect to the horizontal edges (red edges). A few example function values are shown. Right: Triples of triangles aligned with the edges in G_k (in this case, horizontal) and their corresponding triangles in the difference patch.

call a *difference patch* and encoded using differential encoding again. Figure 9 (left) shows an example of a difference patch. Its vertices are in one-to-one correspondence with the edges of the original patch belonging to G_k. Its triangles can be obtained in the following way. Consider triples of consecutive triangles aligned with the edges in G_k, being the horizontal edges in Figure 9 (right). Every such triple defines a triangle of the difference patch. The vertices of that triangle are the vertices of the difference patch corresponding to the horizontal edges of the triangles in the triple. This procedure works well even for patches with complex self-overlaps.

Note that the difference patch need not be connected any more. However, the same scheme as described before can be used for the encoding of the differences treated as a function on the difference grid: edges can be oriented so that their projections are the same vectors in the base plane as before. Again, this defines their splitting into three groups. Thus, instead of encoding the differences explicitly, we may encode them as base vertex values (since there may be several connected components, we generally will have more than one), differences across bridges and second order differences. This procedure can be applied recursively some number of times and the higher order differences that result in the end can be encoded explicitly. The decision as to how many iterations to perform before resorting into explicit encoding is a complex tradeoff. First of all, each iterations requires some computation time: the initial ones can be estimated to take time roughly proportional to the size of the resampled mesh. If quantization is neglected, applying the differencing procedure more times leads to more vanishing moments of the filter producing the residuals. However, it also increases the support of the filter, which makes it more sensitive to discontinuities. The filters with larger support also tends to behave poorly after quantization step is performed, since it naturally decreases the smoothness of the function. After experimenting with several different ways of applying the ideas described before, we found out that three iterations of the differencing operator with respect to edges running in all three different directions, leads to best results, and this is what will be discussed in the results section.

4. EXPERIMENTAL RESULTS

To measure the performance of our algorithms, we ran them for five well-known densely sampled manifold models: the horse, the Venus head, the rabit, the Stanford bunny and the feline. Since our implementation does not handle boundaries too well at this time, we

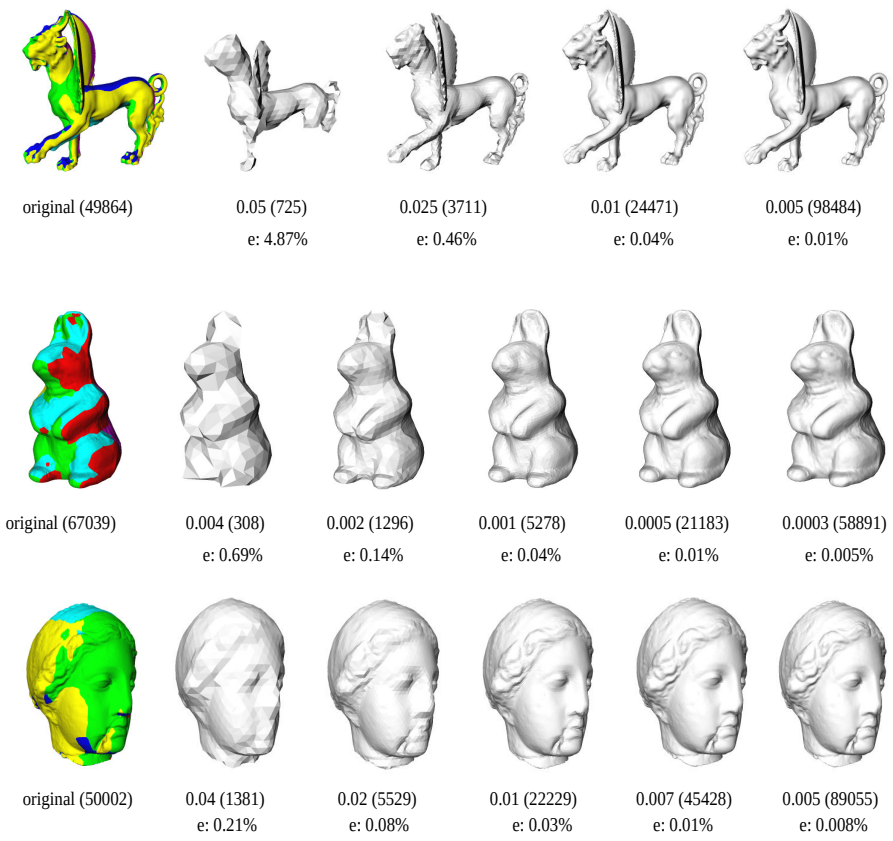

FIG. 10 The output of the resampling procedure for the feline, the rabbit and the Venus head models. The grid sizes, numbers of vertices and the approximation errors (per cent of the diameter of the bounding box) are shown next to each mesh. The original models, with color-coded relief layout, are shown on the left.

preprocessed the bunny model by patching each hole with a closed triangle fan centered at the average of all vertices bounding the hole. The results of our PRM resampling procedure for three of these models are shown in Figure 10.

In Table 1, we show the resampling and compression results for our test models. For each of the models, the grid size was chosen so that the resampled model approximates the original with accuracy of the order of 0.02 per cent of the diameter of the bounding box. The first part of the table compares the number of vertices of the original mesh, n_o to the number of vertices of the resampled model, n_r for the grid size of g. The approximation error (the mean square error output by the Metro tool [5]) is denoted by e and measured as the fraction of the diameter of the bounding box. The measurements show that our procedure is able to achieve very good approximation rates while decreasing the number of vertices.

The second part of the table compares the sizes of the encodings of the original (S_o) and resampled (S_r) models (both measured in kilobytes) using the algorithm of [32] (i.e. the parallelogram rule prediction) for compressing the vertex locations and the Edgebreaker [24] with valence-based prediction as described in Section 3.1 to compress the connectivity. Notice that the savings are, on average, about 60 per cent. The main contributing factor is that our resampling procedure causes the entropy of the parallelogram rule residuals

model	n_o	n_r	g	$e * 10^4$	S_o	S_r	conn	irreg	geom	total	T [s]
Venus	50002	45428	0.007	1.3	116.3	43.1	1.7	0.7	19.0	21.4	0.73
horse	48485	34401	0.001	1.3	73.3	35.8	2.1	1.0	12.6	15.8	0.54
rabbit	67038	21183	0.0005	1.1	100.0	23.0	1.2	0.6	9.1	10.8	0.32
feline	49864	38431	0.008	2.6	88.4	40.4	3.0	1.7	17.9	22.6	0.65
bunny	34839	24499	0.0015	2.0	49.1	24.2	1.3	0.6	12.4	14.4	0.38

TABLE 1
Experimental results

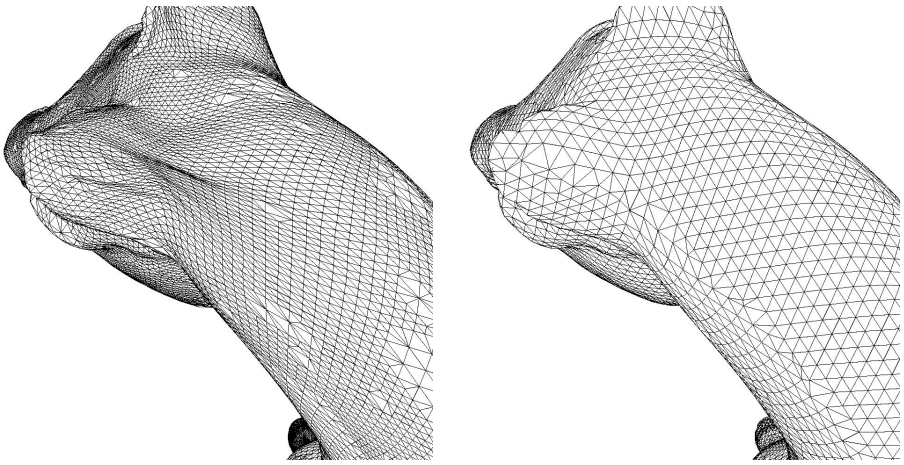

FIG. 11 A closeup of MAPS (left) and PRM (right) remeshes of the horse model.

to decrease by about 40 per cent when compared to the entropy for the original models, which is due to the increased order of the parallelogram rule predictor on the regular pieces and the fact that, on the regular pieces, two out of three coordinates of the residual vectors become zero (up to the quantization error).

Finally, the last part of the table shows the results of our encoding procedure, and the breakdown of the compressed size (measured in kilobytes) between the three components: the connectivity, the encoding of the irregular triangles and the vertex locations (again, quantized to 12 bits). Notice that our scheme is about 50 per cent better than the scheme of [32] on the piecewise regular meshes. This means about 80 per cent storage savings when compared to the algorithm of [32] executed on the original models. The additional error induced by the quantization of the vertex coordinates is, for all of the models, only about 0.002 per cent. Moreover, the decoding can be implemented very efficiently, with decoding times much below one second for each of the models (on a 650MHz PIII processor). The compression and resampling take about 4 second and 5 minutes (respectively) in each of the cases discussed above. However, their implementation has been optimized for programmer's time rather than the running time and we are quite sure that they can be made run even a few times faster.

Let us also mention that we did a comparison with the following simple alternative to our scheme: simplify the original model until the resulting model can be compressed (e.g. using the scheme of [32]) to fit into space required by our compressed piecewise regular mesh. However, it turns out that this actually leads to more approximation error. Our estimates show that, for example, for the horse model, one needs to simplify it until the number of vertices reaches around 6300 which leads to an approximation error of over 0.018 per cent. The corresponding figures for the Venus model are 8000, 0.029 and for the

rabbit model – 4000, 0.028.

In terms of rate-distortion, the algorithm presented here is still not as good as the wavelet geometry compression of [15, 14] (although our difference scheme for the regular parts can certainly be improved upon). However, it is faster and has other desirable properties. We do not require global reparametrization, which makes our algorithms deal well with situation where high energy is concentrated in frequency bands isolated from each other; thus, the problems which the MAPS algorithm has where the shape of a base triangle does not fit with the shape of some intermediate-scale detail (typically leading to ill-shaped triangles in the remeshed model, see Figure 11) does not appear here. Being more local in nature and more efficient, we hope that our procedure will work well for very large and detailed models, like those discussed in [18]. Possible future developments include an adaptive PRM representation and a progressive coder (e.g. one based on hexagonal wavelets), using graphics hardware to speed up the remeshing process (the height component can be computed as the depth component for a carefully specified parallel projection) and optimization of the relief layout and base grids.

REFERENCES

[1] P.Alliez and M.Desburn, *Valence-Driven Connectivity Encoding for 3D Meshes*, Computer Graphics Forum. 20 (3), pp. 480-489, 2001.

[2] P.Alliez and M.Desburn, *Progressive Compression for Lossless Transmission of Triangle Meshes*, Computer Graphics (Proc. SIGGRAPH 2001), pp.195–202.

[3] M.de Berg, M.van Kreveld, M.Overamrs and O.Schwartzkopf, Computational Geometry, Algoriths and Applications, Springer-Verlag Berlin Heidelberg 1997.

[4] B.Chazelle, D.P.Dobkin, N.Shouraboura and A.Tal *Strategies for Polyhedral Surface Decomposition: An Experimental Study*, Symposium on Computational Geometry 1995, 297–305.

[5] P.Cignoni, C.Rocchini and R.Scopigno, *Metro: Measuring error on simplified surfaces*, Computer Graphics Forum 17, 2(1998), 167–174.

[6] M.Deering, *Geometry Compression*, Computer Graphics, Proc. SIGGRAPH 1995, pp. 13–20.

[7] M.Garland, P.Heckbert, *Surface Simplification Using Quadratic Error Metric*, Computer Graphics (Proc. SIGGRAPH 1997), pp. 209–216, August 1997.

[8] M.Garland, A.Willmott and P.S.Heckbert, *Hierarchical Face Clustering on Polygonal Surfaces*, 2001 ACM Symposium on Interactive 3D Graphics. pp. 49-58, 2001.

[9] S.Gumhold and W.Strasser, *Real Time Compression of Triangle Mesh Connectivity*, Computer Graphics (Proc. SIGGRAPH), pp.133-140, July 1998.

[10] I.Guskov, K.Vidimce, W.Sweldens, P.Schroder, *Normal Meshes*, Proc SIGGRAPH 2000, pp. 95–102.

[11] M.Hilaga, Y.Schinagawa, T.Kohmura and T.L.Kuni, *Topology Matching for Fully Automatic Similarity Estimation of 3D shapes*, Computer Graphics (Proc. SIGGRAPH 2001), pp. 203–212.

[12] M.Isenburg and J.Snoeyink, *Spirale Reversi: Reverse decoding of Edgebreaker encoding*, Technical Report TR-99-08, Department of Computer Science, University of British Columbia, September 1999.

[13] Z.Karni and C.Gotsman, *Spectral Compression of Mesh Geometry*, Computer Graphics (Proc. SIGGRAPH 2000), pp. 279-286, 2000.

[14] A.Khodakovsky, I.Guskov, *Normal Mesh Compression*, preprint 2000.

[15] A.Khodakovsky, P.Schroder, W.Sweldens, *Progressive Geometry Compression*, Proc. SIGGRAPH 2000, pp. 271-278, 2000.

[16] F.Lazarus and A.Verroust, *Level Set Diagrams for Polyhedral Objects*, Proc. Fifth ACM Symposium on Solid Modeling and Applications, June 9-11, 1999, pp.130–140.

[17] A.Lee, W.Sweldens, P.Schrder, L.Cowsar and D.Dobkin *MAPS: Multiresolution Adaptive Parameterization of Surfaces*, Proceedings of SIGGRAPH 98, pp.95–104.

[18] M.Levoy, K.Pulli, B.Curless, S.Rusinkiewicz, D.Koller, L.Pereira, M.Ginzton, S.Anderson, J.Davis, J.Ginsberg, J.Shade, and D.Fulk, *The Digital Michelangelo Project: 3D scanning of large statues*, Computer Graphics (SIGGRAPH 2000 Proceedings)

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

[20] R. Pajarola and J. Rossignac. *Compressed Progressive Meshes* , IEEE Transactions on Visualization and Computer Graphics, vol. 6, no. 1, pp, 79–93, 2000.

[21] R. Pajarola and J. Rossignac. *Squeeze: Fast and Progressive Decompression of Triangle Meshes*, Computer Graphics International conference, Switzerland, pp. 173–182, June 2000.

[22] M.Pauly and M.Gross, *Spectral Processing of Point-Sampled Geometry*, Computer Graphics (Proc. SIGGRAPH 2001), pp. 379–386.

[23] Jovan Popovic, Hugues Hoppe, *Progressive Simplicial Complexes*, Proceedings of SIGGRAPH 97. pp. 217-224, 1997.

[24] J.Rossignac, *Edgebreaker: Compressing the connectivity of triangle meshes*, GVU Technical Report GIT-GVU-98-17, Georgia Institute of Technology, IEEE Transactions on Visualization and Computer Graphics vol.5(1), 1999.

[25] J.Rossignac and A.Szymczak, *Edgebreaker compression and Wrap&Zip decoding of the connectivity of triangle meshes*, Computational Geometry: Theory and Applications, 1999.

[26] A.Said, W.A.Pearlman, *A new, fast and efficient image codec based on set partitioning in hierarhical trees*, IEEE Trans.Circuits Syst.Video Technol. 6, 243-250, (1996).

[27] D.Salomon, *Data Compression: The Complete Reference*, Springer-Verlag Berlin Heidelberg 2000.

[28] M.Schindler, *A Fast Renormalization for Arithmetic Coding*, Proc. IEEE Data Compression Conference, Snowbird UT, p. 572, 1998; `www.compressionconsult.com/rangecoder`.

[29] J.Shapiro, *Embedded image coding using zero-trees of wavelet coefficients*, IEEE Trans.Signal Process.41, 3445-3462 (1993).

[30] A.Szymczak, D.King and J.Rossignac, *An Edgebreaker-based efficient compression scheme for regular meshes*, Computational Geometry: Theory and Applications 2000.

[31] G.Taubin and J.Rossignac, *Geometric Compression Through Topological Surgery*, ACM Transactions on Graphics 17 (1998), pp. 84-115.

[32] C.Touma and C.Gotsman, *Triangle Mesh Compression*, Proceedings Graphics Interface 1998, pp. 26-34.

[33] G.Turk, *Re-Tiling Polygonal Surfaces*, Computer Graphics, Vol. 26, No. 2 (July 1992) (SIGGRAPH 92 Conference Proceedings) pp. 55-64.