AN EVALUATION OF SEVERAL MESH-GENERATION METHODS USING A SIMPLE MESH-BASED IMAGE CODER

Michael D. Adams

Dept. of Elec. and Comp. Engineering, University of Victoria, Victoria, BC, V8W 3P6, Canada

ABSTRACT

A simple image coder based on triangle meshes is proposed, and several mesh-generation methods are presented, some previously proposed and some new. Then, our coder is used to evaluate the performance of the various mesh-generation methods. One of our proposed mesh-generation schemes, inspired by the work of Garland and Heckbert, is shown to perform best. The performance of linear and Clough-Tocher interpolation is also compared. Linear interpolation is found to perform better than Clough-Tocher interpolation, with the latter performing more poorly largely due to overshoot/undershoot near image edges.

***Index Terms*—** image coding, triangle meshes, mesh generation, interpolation

1. INTRODUCTION

The image representations used by most conventional image coders employ regular sampling, such as lattice-based sampling. Due to the nonstationary nature of most images, however, such sampling is far from optimal. Inevitably, when regular sampling is employed, the sampling density will be too low in regions where the signal is rapidly changing, and too high in regions where the signal is varying slowly or not at all. This motivates the use of (arbitrary) irregular sampling.

In recent years, there has been a growing interest in triangle meshes for the representation of images [1, 2, 3]. Such meshes are ideally suited to accommodating arbitrary sampling patterns. One of the major challenges of the mesh-based approach is finding effective methods for constructing a mesh that accurately represents a given image. Although numerous mesh-generation methods have been proposed to date (e.g., [3, 2]), much more work needs to be done to further refine and improve upon these methods.

In this paper, we introduce a simple image coder based on triangle meshes. Then, we present a number of mesh-generation methods, and use our coder to evaluate their performance. Since some of the methods differ only in the interpolation scheme employed, our evaluation also implicitly considers the performance of different interpolation schemes.

The remainder of this paper is structured as follows. Section 2 provides some background information on mesh-based image coding. Then, in Section 3, our proposed image coder is presented. Section 4 introduces several mesh-generation methods. Then, in Section 5, our proposed image coder is used to evaluate the performance of the various mesh-generation methods. Finally, Section 6 concludes the paper with a summary of our key results.

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

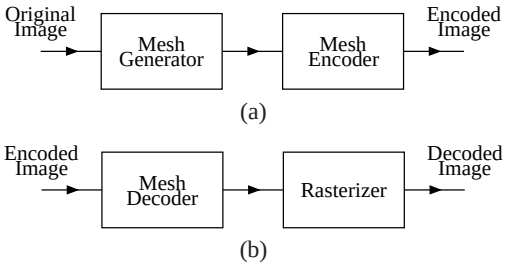

Fig. 1. General structure of a mesh-based image coder. (a) Encoder and (b) decoder.

2. BACKGROUND

A grayscale image is a function f of two variables x and y, where x and y correspond to position in the image domain D and $f(x, y)$ corresponds to image intensity. In other words, an image is simply a surface f defined on a planar domain D. Since triangle meshes can be used to represent a surface, such meshes can also be used for image representations.

Now, we briefly explain how mesh-based representations are used for image coding. The general structure of a mesh-based image coder is illustrated in Fig. 1. The encoder, shown in Fig. 1(a), first constructs a mesh that well approximates the original image. Then, the resulting mesh is coded in some efficient manner, possibly using entropy coding. The decoder, shown in Fig. 1(b), first decodes the coded version of the mesh, and then converts the resulting mesh (which is defined on a continuous domain) into a raster image. The rasterization scheme typically depends on the mesh-generation method employed in encoder. For example, if a particular type of interpolation is assumed to be used to form the surface through the mesh vertices during mesh generation, the same interpolation scheme would most likely be used for rasterization as well. Although a mesh can either interpolate or strictly approximate samples from the original image, we consider only the interpolating case herein.

3. IMAGE CODER

To provide a framework for evaluating the performance of mesh-generation methods, we have developed a simple mesh-based image coder. This coder supports both lossy and lossless compression of (grayscale) images of arbitrary dimensions (i.e., width/height). In what follows, the technical details of our coder are briefly presented.

Our proposed coder has the same general structure as that introduced earlier in Fig. 1. In principle, any mesh-generation method could be used in our coder, provided that the resulting mesh can be completely characterized by only its geometry information (i.e., the mesh connectivity must be implicit) and the vertex coordinates are

integer. As it turns out, these restrictions, which are imposed by our choice of mesh-coding scheme, are not very limiting. In our work, we explicitly consider several mesh-generation methods, which are introduced later in Section 4. Given a particular mesh-generation method, the rasterization process (in the decoder) is straightforward to deduce. A raster image is obtained simply by sampling the mesh surface on an appropriate lattice. Thus, it only remains to discuss the mesh-coding method used in our image coder.

The mesh-coding scheme employed by our coder is based on the scattered data coding (SDC) method [1, Section 3] proposed by Demaret and Iske. The SDC method codes only geometry information, assuming the connectivity information to be implicitly determined. Also, vertex coordinates are assumed to be integer. To begin, the scheme finds the 3D bounding box of the mesh vertices. Then, a 3D binary array is constructed, where each element in the array corresponds to a possible vertex position on the 3D integer lattice inside the bounding box. An element in the array is set to one if a vertex is present at the corresponding position and zero otherwise. The contents of the 3D array are then represented and coded using an octree. Octree generation involves the recursive splitting of cells. The root cell is set to the bounding box of the mesh vertices, and each cell is recursively split into eight new subcells, with the recursion terminating when the current cell is empty, full, or is a so called atomic cell (i.e., a small cell that is not split further). The SDC coding process is effectively lossless, although lossy coding can be achieved by quantizing the image-intensity portion of the vertex-coordinate data prior to coding as is done in [1].

Unfortunately, the SDC method, as presented in [1], can only directly handle images that are square with integer-power-of-two dimensions. In our work, we have extended this method to handle images with arbitrary dimensions. In order to accomplish this, a few changes to the original method were required, which we summarize in what follows. First, when splitting cells, we must use rounding in the determination of subcell boundaries in order to force such boundaries to fall on the integer lattice. Second, in the original method, atomic cells (subject to some reasonable assumptions) always contained four elements in a $2\times2\times1$ geometric configuration, but in the modified scheme, atomic cells can contain two, three, or four elements in a variety of geometric configurations. For each atomic cell, we must indicate which elements are zero/one. Bearing in mind that an atomic cell is neither empty nor full, this is accomplished as follows. In the two-element-cell case, the single nonzero element is indicated with a single bit. In the three-element-cell case, a fixed Huffman code is used to indicate the single zero/nonzero element (which has one of three possibilities). The four-element-cell case is then handled using a scheme similar to that in the original SDC method.

In our image coder, the rate is controlled by adjusting the mesh vertex count (rather than through quantization of the mesh data). Since the mesh coder only employs very trivial variable-length codes, the encoding and decoding of mesh data are very fast. Given the simplicity of our image coder, its performance is quite reasonable, as demonstrated by later experimental results.

4. MESH GENERATION METHODS

In our work, we consider three (nontrivial) mesh-generation methods as well as some variants thereof. Before discussing the details of each individual method, however, we first describe the general approach shared by all of the methods.

All of the mesh-generation methods considered herein have the same general structure. For a given image, a set of sample points is selected. The various methods essentially differ in how this set is chosen. To avoid unnecessary complications (such as the need for extrapolation), the four corners of the image bounding box are always included in the set of sample points. The chosen sample points are then triangulated. Each triangulation vertex and its corresponding intensity value, together comprise a vertex in the mesh, and the triangulation connectivity determines the connectivity of the mesh. An interpolant is defined over each triangular domain in the triangulation to yield a mesh surface defined over the entire image domain.

In general, a mesh is characterized by its geometry (i.e., vertices), and connectivity, as well as the interpolant used to form a surface through the mesh vertices. With all of the methods considered herein, the connectivity of the mesh is assumed to be implicit in its geometry. This has the benefit of not requiring any connectivity information in order to completely characterize the mesh.

For the purposes of triangulation, all of the methods employ the Delaunay triangulation, due to its good approximation properties. Unfortunately, the Delaunay triangulation of a given point set is not necessarily unique. This poses a problem, since we require the connectivity of the mesh to be completely determined from its geometry. To resolve this nonuniqueness problem, we employ the preferred-direction method [4]. This method, in effect, provides a simple means for uniquely choosing one particular triangulation out of all possible Delaunay triangulations of a point set.

Now, we turn our attention to the specifics of the individual mesh-generation methods. The first method considered herein was proposed by Yang, Wernick, and Brankov (YWB) [2]. In the YWB method, a feature map is computed that approximates the largest magnitude second directional derivative at each sample point in the image. Then, the well-known Floyd-Steinberg error diffusion algorithm is used to distribute sample points so that their local spatial density is approximately proportional to the corresponding value in the feature map. The sample points are then triangulated in order to establish the mesh connectivity. The number of vertices in the mesh is indirectly controlled through an error-diffusion threshold parameter. The YWB method results in more sample points being placed in regions containing significant high-frequency information, and fewer sample points being placed in smoother regions. Although several variations on the YWB method are presented in [2], we consider only the interpolating scheme with the exact feature map, serpentine scan order, nonleaky error diffusion, and the contrast parameter $\gamma = 1$.

The second mesh-generation method considered in our work is the data-independent greedy insertion method proposed by Garland and Heckbert (GH) [3]. In the GH method, a triangulation of the image bounding box is chosen as the basis for an initial approximation. Then, the (unused) sample point with the largest absolute error is inserted into the triangulation. This process is repeated until a vertex-count budget is exhausted or distortion criterion (e.g., peak-absolute error or mean-squared error) is satisfied.

As it turns out, the point-selection process in the GH method can be equivalently viewed as consisting of two steps: 1) in the current triangulation, select a triangle in which to insert a new point; and then 2) in this triangle, select an unused point to insert. In the GH method, in both of these two steps, the decision to be made is driven by absolute error. That is, triangle selection is performed by choosing the triangle with the (unused) sample point having the largest absolute error, and point selection is done by choosing the sample point in the selected triangle with the largest absolute error. This alternative way of viewing the GH method leads us to propose a new mesh-generation method, which we henceforth refer to as the modified GH (MGH) method. This is the third method to be consid-

ered. The MGH method is identical to the GH method except that in step 1 in the above point-selection process, we change how the triangle is selected. Instead of choosing the triangle containing the sample point with the largest absolute error, we choose the triangle with the largest squared error.

For benchmarking purposes later, we also define a random mesh-generation method. This mesh generator is very trivial and simply chooses sample points randomly and then triangulates them to form a mesh.

In all of the mesh-generation methods introduced so far, a linear interpolant is used to form a surface through the mesh vertices. It is interesting, however, to consider the effects of using other interpolants. To this end, we propose variants of the GH and MGH methods that employ the Clough-Tocher (CT) interpolant [5], yielding what we refer to as the GH-CT and MGH-CT methods, respectively. The CT interpolation scheme produces surfaces with continuous first-order partial derivatives (i.e., C^1 surfaces). Since CT interpolation requires the first-order partial derivatives at the mesh nodes, a scheme for estimating such derivatives is required. For partial-derivative estimation, we use the scheme described in [6].

5. EXPERIMENTAL RESULTS

Using the proposed image coder, we now evaluate the performance of the various mesh-generation methods introduced earlier. Although numerous test images were employed in our work, we focus our attention herein on the results obtained for the well-known `lena` and `peppers` images (from the USC image database). In what follows, the term "normalized bit rate" will frequently appear. As a matter of terminology, the normalized bit rate (NBR) is simply defined as the reciprocal of the compression ratio.

The proposed image coder was used to compress several images at various bit rates using the YWB, GH, and MGH mesh-generation methods. A representative subset of the results are presented here. In particular, the peak-signal-to-noise ratio (PSNR) results for the `lena` and `peppers` images are given in Figs. 2(a) and (b), respectively. For reference purposes, we also include results for the random mesh-generation scheme (described earlier). Clearly, any practically useful mesh-generation method must yield significantly better results than the random scheme. From the graphs, we can see that the proposed MGH scheme performs best, especially at low bit rates, with the GH method taking second place, and the YWB approach assuming a distant third place. It is worth noting that, for NBRs below 0.04 (i.e., compression ratios above 25:1), the MGH method often beats the GH scheme by more than 1 dB. Interestingly, the random scheme often outperforms the YWB method for NBRs below 0.02 (i.e., compression ratios above 50:1). Obviously, the YWB method is quite ineffective at very low bit rates. Furthermore, it consistently performs more poorly than the GH and MGH methods at all bit rates under consideration.

To illustrate the subjective image quality obtained with the various methods, a set of lossy reconstructed images is shown in Fig. 3. The image-domain triangulation obtained with each method is also shown. Again, for reference purposes, the results for the random method are included. From the reconstructed images, we can see that the subjective image quality is consistent with the PSNR results from above. That is, our proposed MGH scheme performs best, followed by the GH and YWB methods (in that order). As for whether the YWB method yields a better quality image reconstruction than the random scheme is debatable, as both image reconstructions look very poor.

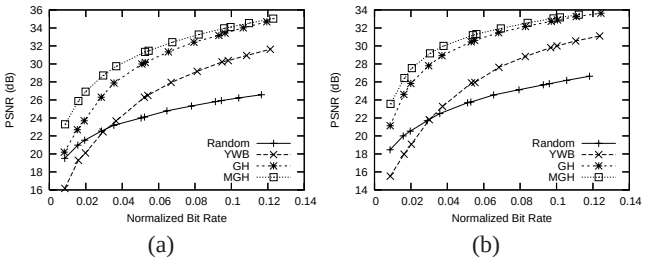

Fig. 2. Lossy coding results for the (a) `lena` and (b) `peppers` images using the random, YWB, GH, and MGH methods.

Now, let us consider the image-domain triangulations shown in Figs. 3(f), (g), and (h). By comparing their respective image-domain triangulations, we can see why the GH method performs more poorly than the MGH scheme at lower rates. With the GH method, since points are added solely by considering absolute error, there is a tendency for too many points to cluster around edges and other high-frequency features, leaving too few points for elsewhere. This results in more long thin triangles, which are known to be undesirable for approximation. Since the MGH method considers the squared error over each triangle in the triangulation, this method is much less likely to cluster too many points around high-frequency features. The preceding behavior is clearly evident in the figures, as the GH method has a triangulation with more long thin triangles than the one obtained with the MGH method. By examining the image-domain triangulation for the YWB method, we can see the reason for its poor performance. At low rates, the YWB method does not place enough vertices along edges, leading to severe distortion in the vicinity of edges which manifests itself as triangle-teeth artifacts.

There is yet another problem with the YWB method. At low rates, the error-diffusion threshold employed in the method must be chosen so large that the likelihood of choosing sample points from the image rows first processed by error diffusion becomes quite small (or even zero). The number of rows affected increases with the threshold. In our implementation of the YWB method, error diffusion begins at the bottom of the image and proceeds upwards. This explains why the triangulation in Fig. 3(f) has very few sample points selected from the bottommost rows of the image.

Lastly, we consider the effects of changing the interpolant from linear to CT in the GH and MGH methods. Recall, that the GH-CT and MGH-CT mesh-generation methods are simply the GH and MGH methods, respectively, with the interpolant changed to CT. The proposed image coder was used to compress several images at various bit rates using the GH, MGH, GH-CT, and MGH-CT methods, with the results for the `lena` and `peppers` images given in Fig. 4. From these graphs, we can see that the GH and MGH methods clearly outperform the GH-CT and MGH-CT methods, respectively. In other words, linear interpolation outperforms CT interpolation. In terms of subjective image quality, a similar conclusion is also drawn. An example of two reconstructed images is shown for the MGH/MGH-CT case in Fig. 5. The schemes employing the CT interpolant have annoying artifacts caused by excessive undershoot/overshoot in the vicinity of edges.

6. CONCLUSIONS

In this paper, we have proposed a simple mesh-based image coder and presented several mesh-generation methods (some previously proposed and some new). Our coder has been used to evaluate the

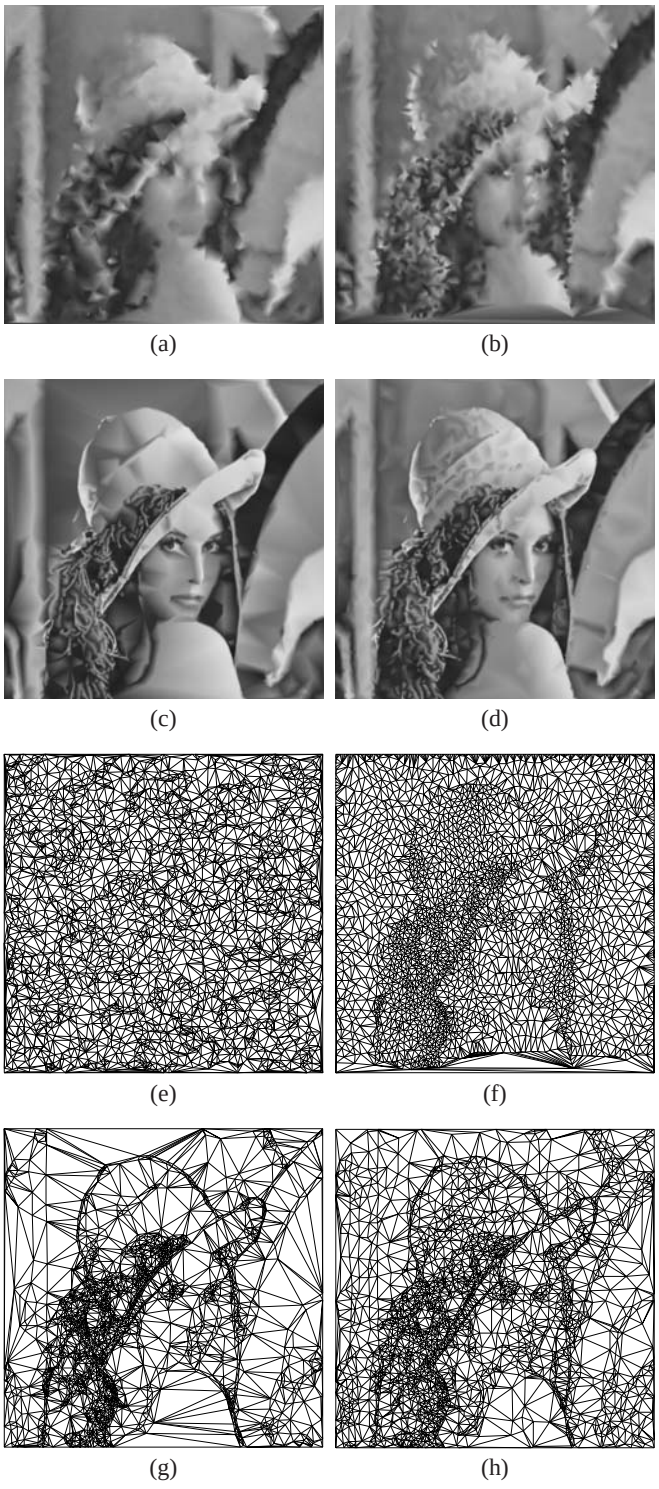

Fig. 3. Lossy reconstructed images obtained at about 50:1 compression with the (a) random (21.52 dB), (b) YWB (20.10 dB), (c) GH (23.67 dB), and (d) MGH (26.91 dB) methods; and the image-domain triangulation associated with each of the (e) random, (f) YWB, (g) GH, and (h) MGH methods.

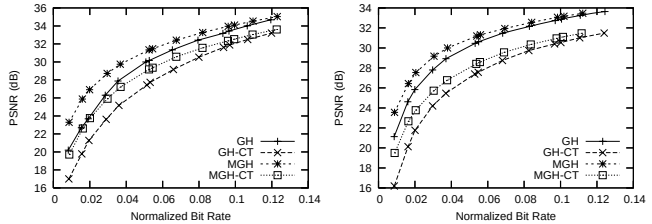

Fig. 4. Lossy coding results for the (a) `lena` and (b) `peppers` images using the GH, GH-CT, MGH, and MGH-CT methods.

Fig. 5. Lossy reconstructed images obtained at about 50:1 compression with the (a) MGH (26.91 dB) and (b) MGH-CT (23.76 dB) methods.

performance of the various mesh-generation methods. Of the methods considered, our proposed MGH mesh-generation scheme was shown to yield the best coding performance (both objectively and subjectively). Through our evaluation we have also shown that the use of a CT interpolant leads to much poorer results than a linear interpolant, due to severe overshoot/undershoot in the vicinity of image edges. Through the insights provided by our work, one can hope to develop improved mesh-based image coders in the future.

7. REFERENCES

[1] L. Demaret and A. Iske, "Scattered data coding in digital image compression," in *Curve and Surface Fitting: Saint-Malo 2002*, Brentwood, TN, USA, 2003, pp. 107–117, Nashboro Press.

[2] Y. Yang, M. N. Wernick, and J. G. Brankov, "A fast approach for accurate content-adaptive mesh generation," *IEEE Trans. on Image Processing*, vol. 12, no. 8, pp. 866–881, Aug. 2003.

[3] M. Garland and P. S. Heckbert, "Fast polygonal approximation of terrains and height fields," Tech. Rep. CMU-CS-95-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, Sept. 1995.

[4] C. Dyken and M. S. Floater, "Preferred directions for resolving the non-uniqueness of Delaunay triangulations," *Computational Geometry—Theory and Applications*, vol. 34, pp. 96–101, 2006.

[5] R. Clough and J. Tocher, "Finite element stiffness matrices for analysis of plates in bending," in *Proc. of Conference on Matrix Methods in Structural Mechanics*, Wright-Patterson AFB, OH, USA, 1965.

[6] H. Akima, "On estimating partial derivatives for bivariate interpolation of scattered data," *Rocky Mountain Journal of Mathematics*, vol. 14, no. 1, pp. 41–52, 1984.