An Improved Multiscale Normal-Mesh-Based Image Coder

Di Xu and Michael D. Adams
Dept. of Elec. and Comp. Engineering, University of Victoria, Victoria, BC, Canada
dixu@ece.uvic.ca and mdadams@ece.uvic.ca

Abstract— *Three modifications to the normal-mesh-based image coder of Jansen, Baraniuk, and Lavu are proposed, namely, the use of a data-dependent base mesh, an alternative representation for normal/vertical offsets, and a different scan-conversion scheme based on bicubic interpolation. Experimental results show that our proposed changes lead to improved coding performance in terms of both objective and subjective image quality measures.*

Keywords— *image coding, normal mesh, multiresolution.*

I. Introduction

Many of today's best image coders are based on wavelet transforms. Unfortunately, such coders cannot efficiently represent the geometric features inherent in images (i.e., edges). This has led to an interest in coders that employ geometric representations of images, such as normal (triangle) meshes. Unlike wavelets, meshes are well suited to efficiently capturing the geometric information in images. Recently, an image coder based on normal meshes was proposed by Jansen, Baraniuk, and Lavu [1], which we henceforth refer to as the JBL coder. In this paper, we propose three modifications to the JBL coder and demonstrate through experimental results that these changes lead to improved coding performance.

The remainder of this paper is structured as follows. Section II introduces the JBL coder, with some details regarding our implementation of this coder given in Section III. Section IV proposes three modifications to the JBL coder. Section V evaluates the performance gains achieved by our proposed changes. Finally, Section VI concludes the paper with a summary of our work and some closing remarks.

II. JBL Coder

A grayscale image is a function f of two variables x and y, where x and y correspond to position, and $z = f(x, y)$ corresponds to image intensity. In this way, an image can be viewed as a surface parameterized over the xy plane. Thus, mesh-based techniques for representing surfaces can be used for images. In the case of the JBL coder, a normal (triangle) mesh is employed to represent images.

For our purposes here, a normal mesh [2] is a multiresolution surface representation that consists of a nested sequence of meshes, generated by repeated refinement of a base mesh. The base mesh consists of a small number of points from the true surface. The refinement process then generates a finer mesh by adding new points from the true surface to a coarser mesh. This is done in such a way that each new vertex on the finer mesh can be expressed as a displacement from a base point on the coarser mesh in the direction of the base point's surface normal. In other words, the new vertices added during refinement are located where surface normals from base points on the coarser mesh pierce the true surface (i.e., new vertices are located at so-called **piercing points**). Since each base point and its corresponding normal direction are completely determined by the coarser mesh, only a single scalar value (i.e., a **normal offset**) is needed to identify the location of each new vertex on the finer mesh. Thus, a normal mesh can be completely characterized by its base mesh and a set of normal offsets.

As mentioned previously, an image can be represented as a surface parameterized over the xy plane. In what follows, we refer to this plane as the **parameter plane**. Since some aspects of the JBL coder are more easily explained in terms of the (2D) parameter plane rather than explicit 3D geometry, we will largely adopt a parameter-plane perspective in our description of this coder. In essence, the JBL coder creates a partitioning of the parameter plane using a triangulation, and then forms an interpolant over each of the resulting triangles in order to construct a surface in 3D (i.e., the image surface). (Unless otherwise noted, in what follows, the term "vertex" will always refer to a vertex in the parameter-plane triangulation.) Vertices are associated with height (i.e., z coordinate) values. In this way, each vertex/height-value pair corresponds to a point in 3D. With the JBL coder, the three 3D points associated with the vertices of each triangle are used to form a planar interpolant. By combining these interpolants, an image surface in 3D is formed.

To represent discontinuities, the JBL coder models edges explicitly using the so-called **horizon model**. As a matter of terminology, a contour in the parameter plane that corresponds to a discontinuity contour (i.e., image edge) is called a **horizon**. A vertex that is on a horizon is said to be a **horizon vertex**, and an edge (in the triangulation) with both of its endpoints being horizon vertices is said to be a **horizon edge**. The number of height values associated with a particular vertex, depends on whether the vertex is a horizon vertex. A nonhorizon vertex is associated with only one height value, while a horizon vertex is associated with two, in order to represent the height of the image surface on both sides of the horizon. To distinguish between these two cases, each vertex is associated with a bit, called a **horizon bit**, indicating if the vertex is a horizon vertex.

Although the JBL coder employs a normal mesh, the base mesh and its subsequent refinement are more easily described in terms of the parameter-plane triangulation (introduced above) rather than directly in terms of the 3D mesh itself. First, the refinement process requires the notion of a true surface. Since images are essentially assumed to be piecewise constant in [1], the true surface is constructed using piecewise constant interpolation of the original image sample data. The base mesh is associated with a particular base (i.e., initial) triangulation of the parameter plane. In the JBL scheme, the base triangulation is chosen to have four vertices, corresponding to the four corner points of the image bounding box. The refinement of the mesh then corresponds to a refinement of the parameter-plane triangulation through the addition of new vertices. In particular, refinement of the triangulation is performed by quaternary **subdivision**, whereby a new vertex is added for each edge in the triangulation (resulting in each triangle being split into four new triangles). The location of the new vertex for each edge is determined in one of two ways, depending on whether the edge is a horizon edge. To simplify the explanation that follows, we neglect a few exceptional cases. In the case of a horizon

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

edge, we define the base point as the midpoint of the edge. Then, the new vertex is added where the normal to the edge through its base point pierces the horizon. In the case of a nonhorizon edge, the base point is defined as the midpoint of the 3D line segment joining the 3D points associated with the vertices at the endpoints of the edge. The new point is added where the normal to the 3D line segment, through its base point and in the vertical plane containing the line segment, pierces the true image surface. Due to the manner in which refinement of the parameter-plane triangulation is performed (using numerous normal directions), this entire process can be viewed as the refinement of a normal mesh.

As suggested earlier, some exceptional cases can occur in the above refinement process. This is because it is not always possible to find a valid piercing point in a normal direction in the manner described above. For example, during the refinement involving a horizon edge, it is possible that the new piercing point does not lead to a valid triangulation (due to non-disjoint triangles). To avoid this and other problems, we must, in some exceptional circumstances, include a distance in the vertical direction, in lieu of or in addition to the normal direction. Therefore, an additional value, called the **direction value**, is required for each offset to remember which combination of normal/vertical directions it employs.

The mesh-based representation of the JBL coder described above is completely characterized by the base mesh, normal/vertical offsets, horizon bits, and direction values. Using this information, the corresponding surface can be reconstructed.

Having introduced the JBL coder, we now take a moment to briefly introduce a greatly simplified version of this coder that is implicitly suggested in [1] (in the context of natural images). We refer to this simplified version of the coder as JBL-S herein. In the JBL-S coder, the true image surface is constructed using piecewise planar interpolation rather than piecewise constant interpolation. Due to the use of a piecewise planar interpolant, no discontinuities exist in the true image surface, and hence no horizons exist either. Thus, the JBL-S coder is essentially the JBL coder without the horizon model. In the case of the JBL-S coder, the mesh-based image representation is completely characterized by only the base mesh and normal/vertical offsets. Unlike in the JBL case, horizon bits and direction values are not used.

III. Our JBL Coder Implementation

Before proceeding further, we briefly describe our implementation of the JBL coder, which was used as the basis for our work herein. Generally, our implementation is written in MATLAB. The encoder uses a Canny edge detector [3] to determine horizons. Then, a normal-mesh representation is constructed, and the corresponding normal/vertical offsets are then quantized and coded.

The normal/vertical offsets are scalar quantized with a separate quantizer being used for the offsets of each subdivision level. Rather than specifying all of the quantizer step sizes individually, all step sizes are computed from a single encoder parameter q. In particular, the step size Δ_k for the offsets of the kth subdivision level (where $k = 0$ corresponds to the base mesh) is chosen as $\Delta_k = q2^k$. This choice results in offsets from coarser levels being weighted more heavily than those from finer levels. Such is desirable, since in a normal mesh, errors in the coarser-level offsets introduce considerably more distortion than errors in the finer-level offsets.

In [1], Jansen et al. do not make any attempt to estimate the bit rate required to code the data of the mesh-based representation. In our work, however, this rate is estimated using the data's entropy. To be brief, a Laplacian distribution is used to model the normal/vertical offsets, while a simple first-order entropy is used for the horizon-bit and direction-value data. The parameters of the associated probability distributions are encoded as side information. The base mesh data is not assumed to be entropy coded, however.

IV. Proposed Modifications to JBL Coder

Having introduced the JBL coder, we now propose three modifications to it. As we will later show, each of these changes leads to improved coding performance.

The data-independent base mesh used in the JBL coder takes many levels of subdivision to satisfy the fast convergence condition given in [1]. This motivates our first proposed modification to the JBL coder, which is to employ a data-dependent base mesh. In so doing, we can satisfy the above fast convergence condition with the base mesh. In our scheme, we locate the horizons using a Canny edge detector, and then estimate their curvature by the method of [4]. In order to help satisfy the fast convergence condition, vertices are selected from horizon points based on the preceding curvature estimates. The base triangulation in the parameter plane is achieved through a constrained conforming Delaunay triangulation with the Triangle software [5] and some additional code to ensure the uniqueness of the triangulation. Through the appropriate choice of constraints for the triangulation, we are able to ensure the correct connectivity of the horizon vertices, which also contributes to satisfying the fast convergence condition.

Our second modification to the JBL coder involves the representation of normal/vertical offsets. Since integers can be more efficiently coded than real numbers, we propose to use an integer offset instead of a real offset for locating each piercing point. Recall that an offset is a real value measuring the distance between a 3D point on an 3D line segment and its corresponding 3D piercing point. Due to the piecewise constant interpolation process used to generate the true image surface, we observe that at least one of the x, y, or z coordinates of each piercing point must be integer. Therefore, along the normal/vertical search line through the 3D base point, all potential piercing points can be identified, by finding all intersections of the search line and the planes that are parallel to xy, yz, and xz planes with integer z, x, and y coordinates, respectively. These potential piercing points can then be enumerated using an integer index, and this index can be used to specify which of the potential piercing points is the actual piercing point.

Our third proposed modification to the JBL coder involves the scheme employed for scan-conversion. Ideally, we desire a scan-conversion scheme that both preserves smooth regions in an image as well as the sharp intensity changes present along horizons. Bicubic interpolation offers a higher degree of smoothness than planar interpolation, which can often be desirable with respect to subjective image quality. Since we would like to preserve sharp image edges, however, we must apply bicubic interpolation carefully so as to avoid blurring of image edges. In our scheme, when determining the bicubic interpolant for a particular triangle, we only use vertices/height-values from the same side of the horizon as the triangle under consideration. This allows us to avoid unnecessarily smoothing the sharp intensity changes across horizons.

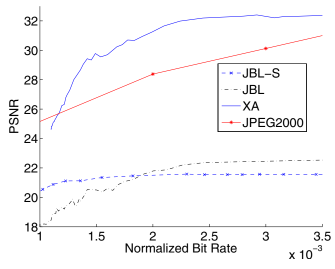
V. Experimental Results

Earlier, we suggested that our proposed modifications to the JBL coder improve its performance. In this section, we support our claim through experimental evidence. Although numerous test images were employed in our work, we focus our attention on the results for two representative images herein, namely, the `paw` and `peppers` images. The `paw` and `peppers` images are both 8-bit grayscale and have dimensions of 1024×1024 and 512×512, respectively. Since we are primarily interested in low bit rates in our work, when we subsequently use qualifiers like "low" or "high" for the bit rate, these qualifiers should be understood in relative terms (i.e., relative to the range of bit rates under consideration in our study). In what follows, we now consider each of the three proposed coder modifications in turn.

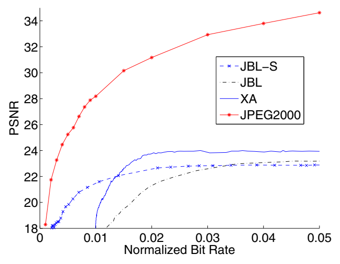
To begin, we consider our proposed modification to the JBL coder of employing a data-dependent base mesh. In what follows, we refer to the JBL coder with this change by the name "XA". To assess the value of our proposed change, numerous test images were compressed at various bit rates with both the JBL and XA methods and the results examined. In what follows, we provide a representative subset of these results for the `paw` and `peppers` images. For reference purposes, we also include results obtained from the JBL-S and (in some cases) JPEG2000 [6] coders. To maintain a fair comparison for the mesh-based methods, the number of subdivision levels for each method was chosen so that the final-mesh vertex counts would be as close to one another as possible (without giving an unfair advantage to our XA method). Since these counts can only be controlled very coarsely, it is only possible to have them match to within a factor of about three. More specifically, for the `paw` image, the JBL and JBL-S coders use six levels of subdivision, resulting in 4225 vertices, while the XA coder uses two levels of subdivision, resulting in 3308 vertices. For the `peppers` image, the JBL and JBL-S coders use seven levels of subdivision, resulting in 16641 vertices, and the XA coder uses two levels of subdivision, resulting in 6543 vertices. Now, we examine the results obtained in detail.

The rate-distortion plots obtained for the `paw` and `peppers` images using the various methods are shown in Fig. 1. From these results, we can see that, at high rates, the XA coder outperforms the JBL and JBL-S coders, and the XA coder even outperforms the JPEG2000 coder in the case of the (synthetic) `paw` image. At low bit rates, however, the XA coder can sometimes perform more poorly than the other three methods due to the rate overhead associated with the base mesh (which is not entropy coded in our scheme). Clever schemes for coding the base mesh, however, could reduce this overhead, and significantly improve the coding efficiency of the XA method at low bit rates. From the coding results, we can also see that, at low bit rates, the JBL coder performs worse than the JBL-S coder, while at high bit rates, the JBL coder performs better than or comparable to the JBL-S coder. The superior performance of the JBL coder at high rates can be attributed to the higher efficiency of the horizon model, while the inferior performance at low rates can be attributed to the overhead of encoding horizon bits and direction values.

Now, we consider the subjective performance of the various methods. For the case of the `paw` and `peppers` images, examples of the obtained reconstructed images are shown in Figs. 2 and 3, respectively. In the first case, the final mesh employed by each method is shown superimposed on the original image with the horizon

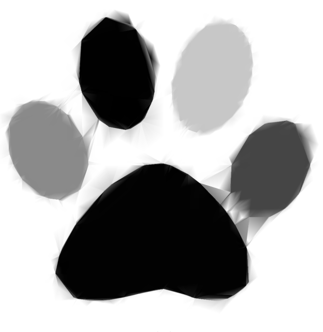


(a)

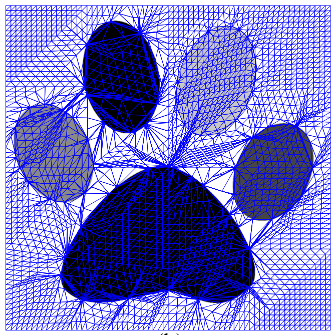


(b)

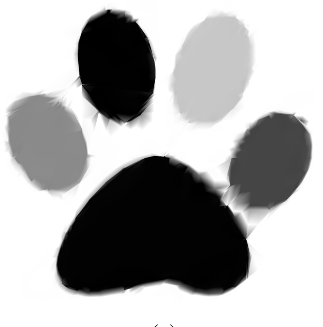
Fig. 1. Coding performance for the (a) `paw` and (b) `peppers` images using the JBL-S, JBL, XA, and JPEG2000 methods.



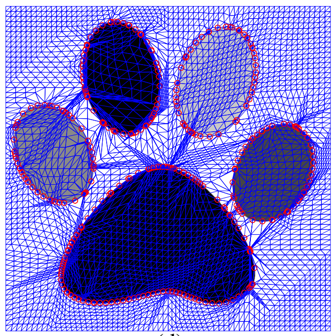
(a)



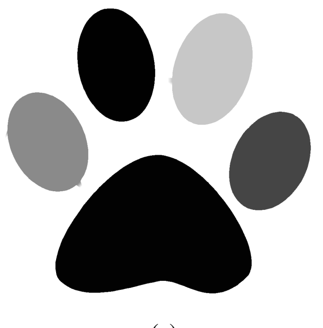
(b)



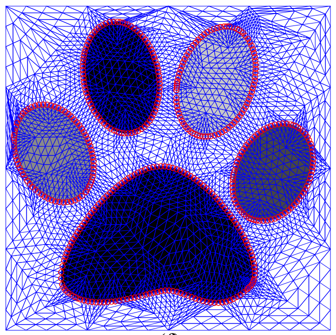
(c)



(d)



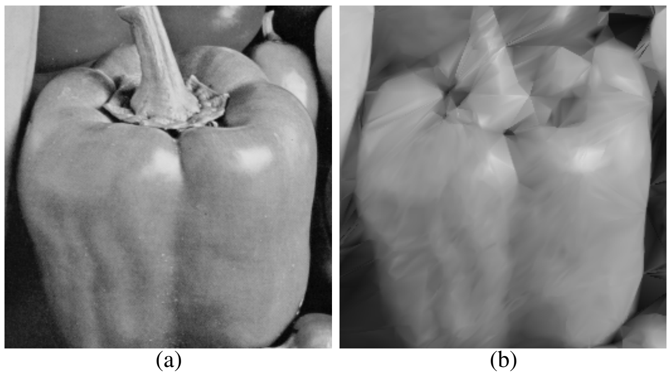
(e)



(f)

Fig. 2. Coding example for the `paw` image. Lossy reconstructions obtained at about 400:1 compression using the (a) JBL-S, (c) JBL, and (e) XA methods. The corresponding final meshes employed by the (b) JBL-S, (d) JBL, and (f) XA methods.

vertices denoted by circles. Examining the results for the `paw` image in Fig. 2, we can see that very significant edge distortions occur in the case of the JBL and JBL-S methods, while the XA scheme has little noticeable distortion. Clearly, the XA method approximates



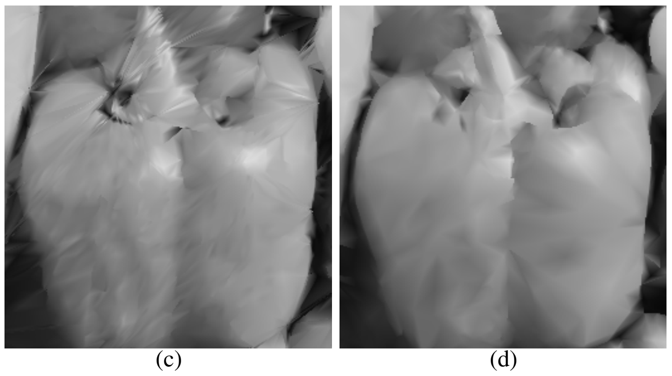


Fig. 3. Coding example for the `peppers` image. Portions of the (a) original image and the lossy reconstructions obtained at about 29:1 compression using the (b) JBL-S, (c) JBL, and (d) XA methods.

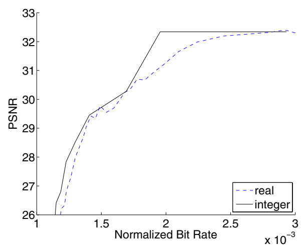


Fig. 4. Coding performance for the `paw` image using real and integer offsets.

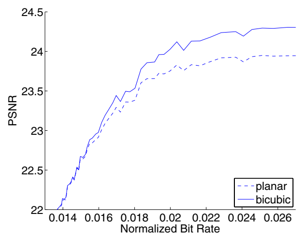


Fig. 5. Coding performance for the `peppers` image using planar and bicubic interpolation.

horizons much better than the JBL and JBL-S methods. Examining the reconstructions of the `peppers` image in Fig. 3, we can see that the results obtained with the XA method are comparable to those obtained with the JBL and JBL-S coders, in spite of the fact that the JBL and JBL-S methods have meshes with about 2.5 times more vertices than the XA case. On this basis, it is reasonable to conclude that the XA scheme is superior to the JBL and JBL-S schemes.

Let us again consider the subjective results for the `paw` image, including the final meshes produced by the various methods, as shown in Fig. 2. By examining the final meshes, we can see why the XA method is able to outperform the JBL and JBL-S methods. The mesh for the JBL-S method has some larger-area triangles that straddle horizons. This leads to very visually disturbing artifacts such as those near the rightmost pad of the paw in Fig. 2(a). By explicitly modelling horizons, the JBL and XA methods are able to locate horizon vertices faster, and reduce distortions in large regions. Furthermore, the XA method, with a data-dependent base mesh, locates horizon vertices faster and approximates horizons better than the JBL and JBL-S schemes. As an aside, we note that the small triangular teeth occurring in the reconstructed image for the XA coder can be attributed to inaccuracies in the estimation of the location and curvature of horizons.

Let us now consider our second proposed change to the JBL coder, which is to use an integer value rather than a real value to represent a normal/vertical offset. To demonstrate the effectiveness of our proposed change, we compare the coding performance obtained with real and integer offsets. Fig. 4 shows the coding performance achieved when each method is employed in the XA coder (using two levels of subdivision) for the `paw` image. From the graphs, we can see that, at low to medium bit rates, integer offsets yield better results than real offsets, with the difference being more pronounced at medium rates, while in the high bit rate case, comparable results are obtained with integer and real offsets. It is worth noting that similar results as above also hold in terms of subjective image quality (i.e., integer offsets are better than or as good as real offsets).

Lastly, we consider our proposed change to the scan conversion scheme, which is to use bicubic instead of planar interpolation. To evaluate the value of the change, both planar and bicubic interpolation were employed in the XA coder to compress the `peppers` image at various bit rates (using two levels of subdivision). The results obtained are shown in Fig. 5. From these results, we can see that bicubic interpolation outperforms planar interpolation, especially at higher bit rates. In terms of subjective image quality, bicubic interpolation also leads to superior results, as it tends to better preserve smoother regions in images, without destroying sharp intensity changes at horizons.

VI. Conclusions

In this paper, we proposed three modifications to the JBL coder and demonstrated through experimental results that these changes lead to improved coding performance. For example, we showed that good data-dependent base meshes can help to locate horizons faster and preserve edges better. Also, we showed that using a normal/vertical-offset representation based on integers (instead of real numbers) yields superior performance. Finally, we demonstrated that, by exploiting horizon information, bicubic interpolation can be made to provide smoother reconstructed images while still maintaining sharp edges. Through our work, we have helped to advance the state of the art in mesh-based image coders.

References

[1] M. Jansen, R. Baraniuk, and S. Lavu, “Multiscale approximation of piecewise smooth two-dimensional functions using normal triangulated meshes,” *Applied and Computational Harmonic Analysis*, vol. 19, pp. 92–130, 2005.
[2] I. Guskov, K. Vidimce, W. Sweldens, and P. Schroder, “Normal meshes,” in *Computer Graphics Proceedings*, 2000, pp. 95–102.
[3] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, Nov. 1986.
[4] M. Marji and P. Siy, “A new algorithm for dominant points detection and polygonization of digital curves,” *Pattern Recognition*, vol. 36, no. 10, pp. 2239–2251, Oct. 2003.
[5] J. R. Shewchuk, “Delaunay refinement algorithms for triangular mesh generation,” *Computational Geometry: Theory and Applications*, vol. 22, no. 1-3, pp. 21–74, May 2002.
[6] *ISO/IEC 15444-1: Information technology—JPEG 2000 image coding system—Part 1: Core coding system*, 2000.