

# An Improved Normal-Mesh-Based Image Coder

## Un Codeur Normal-Maille-Base Ameliore d'Image

Di Xu\* and Michael D. Adams†

The normal-mesh-based image coder of Jansen, Baraniuk, and Lavu (JBL) is studied. First, the JBL coder is introduced and several of its shortcomings are identified. Then, to address these shortcomings, three modifications to this coder are proposed, namely, the use of a data-dependent base mesh, an integer representation for normal/vertical offsets, and a different scan-conversion scheme based on bicubic interpolation. Experimental results show that these proposed changes lead to improved coding performance in terms of both objective and subjective image-quality measures. In particular, the use of a data-dependent base mesh helps to locate horizons more quickly and preserve image edges better. The number of bits required to encode the normal/vertical offsets is reduced by representing this information with integers (as opposed to real numbers). Lastly, bicubic interpolation is found to yield higher-quality image reconstructions, while still maintaining sharp edges.

Le codeur normal-maille-base d'image de Jansen, de Baraniuk, et de Lavu (JBL) est etudie. D'abord, le codeur de JBL est presente et plusieurs de son des imperfections sont identifiees. Puis, pour adresser ces imperfections, on propose trois modifications a ce codeur, a savoir, l'utilisation d'a maille basse donnee-dependante, une representation de nombre entier pour la normale/verticale excentrages, et un arrangement different de balayer-conversion base sur bicubic interpolation. Les resultats experimentaux prouvent que ces changements proposes mnent a l'execution de codage amelioree en termes d'objectif et subjectif mesures d'image-qualite. En particulier, l'utilisation d'une base donnee-dependante la maille aide a localiser des horizons plus rapidement et a preserver des bords d'image meilleur. Le nombre de peu requis pour coder le normal/verticale des excentrages est reduits en representant cette information avec des nombres entiers (As oppose a de vrais nombres). Pour finir, l'interpolation bicubic s'avre pour rapporter des reconstructions plus de haute qualite d'image, tout en maintenant toujours les bords pointus.

**Keywords / Mots cles:** image coding, normal meshes, geometric image representations, triangulations

### I. Introduction

Many of today's best image coders are based on wavelet transforms [1, 2, 3]. Unfortunately, such coders cannot efficiently represent the geometric features inherent in images (i.e., edges). This has led to an interest in schemes that better exploit the geometric properties of images. Some such schemes include ridgelets [4], curvelets [5], edgelets [6], contourlets [7], wedgelets [8], and bandelets [9]. Recently, an image coder based on normal (triangle) meshes was proposed by Jansen, Baraniuk, and Lavu (JBL) [10], which we henceforth refer to as the JBL coder. Unlike wavelets, meshes are well suited to efficiently capturing the geometric information in images. For example, consider images from the so called **horizon class** (i.e., images consisting of constant intensity regions separated by smooth contours of discontinuity). It has been shown [10] that, under certain conditions, the normal-mesh representation employed by the JBL coder can more efficiently represent horizon class images than wavelet-based schemes. In particular, for a representation with  $n$  coefficients, the normal-mesh and wavelet schemes achieve asymptotic error-decay rates of  $O(n^{-1})$  and  $O(n^{-1/2})$ , respectively. Furthermore, since many real-world images behave somewhat like horizon-class images, the preceding error-decay-rate result suggests that the JBL coder has considerable promise for practical image-coding applications.

In spite of its merits, the JBL coder also has some deficiencies that unnecessarily restrict its performance. In this paper (which is an ex-

tended version of [11]), we identify some of these shortcomings, and propose three modifications to the coder aimed at addressing these weaknesses. Furthermore, we show that these modifications lead to improved coding performance.

The remainder of this paper is structured as follows. First, Section II comments on some of the notation and terminology used herein. Then, Sections III and IV, respectively, introduce the JBL coder and the implementation of it used in our work. In Section V, several shortcomings of the JBL coder are identified. This then leads us to propose, in Section VI, three modifications to the JBL coder aimed at addressing these shortcomings. The implementation of our enhanced (i.e., with our proposed modifications) coder is introduced in Section VII. Then, by way of experimental evidence provided in Section VIII, we demonstrate that each of our proposed modifications yields improved coding performance. Finally, we conclude in Section IX with a summary of our work.

### II. Notation and Terminology

Before proceeding further, a short digression is appropriate concerning some of the notation and terminology used herein. The sets of integers, odd integers, and real numbers are denoted as  $\mathbb{Z}$ ,  $\mathbb{Z}_{\text{odd}}$ , and  $\mathbb{R}$ , respectively. In what follows, we assume the reader to be familiar with basic geometric concepts, such as a triangulation, Delaunay triangulation (DT), and constrained DT. For more information on such concepts the reader is referred to [12]. For an image with  $P$  bits per sample, the peak-signal-to-noise ratio (PSNR) distortion measure is defined as  $\text{PSNR} = 20 \log_{10}([2^P - 1]/\sqrt{\text{MSE}})$ , where MSE corresponds to the mean-squared error.

\*Di Xu is with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, V6T 1Z4, Canada. E-mail: dixu@ece.ubc.ca.

†Michael Adams is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, V8W 3P6, Canada. E-mail: mdadams@ece.uvic.ca.

### III. JBL Coder

As mentioned previously, the focus of our work is the JBL coder. In the sections that follow, we provide a brief introduction to this coder. First, we describe the normal-mesh-based image representation employed by the JBL coder and explain how this representation is constructed through a process known as mesh refinement or subdivision. Then, we discuss scan conversion and a few other details relevant to this coder.

#### A. Normal-Mesh-Based Image Representation

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 for this purpose.

For our purposes here, a normal mesh [13] is a multiresolution surface representation that consists of a nested sequence of meshes  $\{M_0, M_1, \dots, M_{L-1}\}$ , generated by repeated refinement of a base mesh  $M_0$ . The base mesh consists of a small number of points from the true surface (i.e., the actual image intensity surface to be approximated). 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**). The line passing through the base point and along some search direction for locating a piercing point is called a **search line**. 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 parameter plane rather than explicit 3-D 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 3-D (i.e., the image surface). In what follows, unless otherwise noted, 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 3-D. With the JBL coder, the three points associated with the vertices of each triangle are used to form a planar interpolant. By combining these interpolants, a piecewise-planar image surface in 3-D is formed.

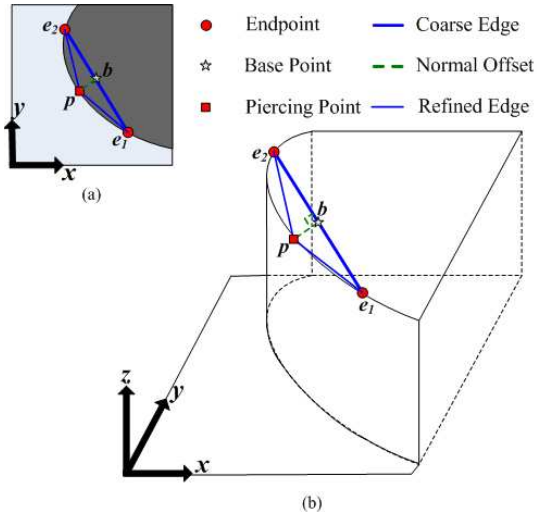
To represent discontinuities, the JBL coder models edges explicitly using the so-called **horizon model**. As a matter of terminology, a contour in the parameter plane that corresponds to a discontinuity contour (i.e., image edge) is called a **horizon**. A vertex that is on a horizon is said to be a **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.

#### B. Mesh Refinement via Subdivision

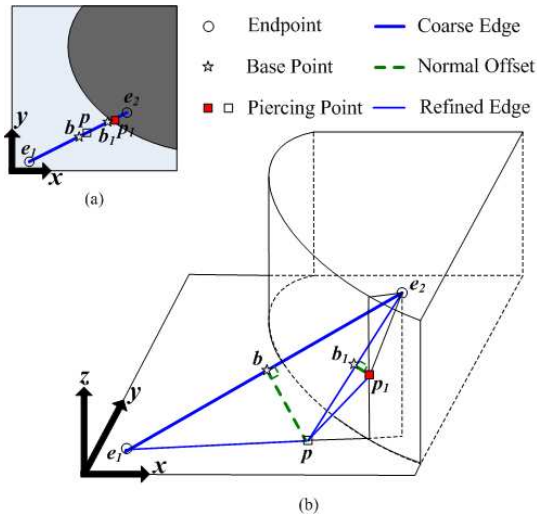
Although the JBL coder employs a normal mesh, the base mesh and its subsequent refinement are more easily described in terms of the parameter-plane triangulation (introduced above) rather than directly in terms of the 3-D mesh itself. First, the refinement process requires the notion of a true surface. Since images are essentially assumed to be piecewise constant in [10], the true surface is constructed using piecewise-constant interpolation of the original image sample data. As a matter of convenience, for the purposes of this interpolation process, the original sample values are aligned with points on the lattice  $\frac{1}{2}\mathbb{Z}_{\text{odd}}$  (rather than the lattice  $\mathbb{Z}$ ). 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. Due to the manner in which the refinement of the parameter-plane triangulation is performed (i.e., using numerous normal directions), this entire process can essentially be viewed as the refinement of a normal mesh.

When performing subdivision as described above, the location of the new vertex to be added for each edge is determined in one of two ways, depending on whether the edge is a horizon or nonhorizon edge. Horizon and nonhorizon edges are treated differently, since the goals of refining these two types of edges are not the same. In the horizon-edge case, the objective is to obtain a better polyline approximation of a horizon, whereas in the nonhorizon-edge case, the objective is to quickly locate new horizon vertices. We will now describe each of these two types of subdivision in more detail. To simplify the explanation that follows, some exceptional cases are not considered in this discussion.

**HORIZON-EDGE SUBDIVISION.** First, we consider the subdivision of a horizon edge. Since our goal in this case is to construct a refined polyline approximation of the horizon, we would like the piercing point associated with the horizon edge to be a point on horizon. To perform the subdivision, we first define the base point as the midpoint of the edge. Then, the search line for the new horizon vertex is chosen to be normal to the edge and parallel to the  $xy$  plane. The new vertex is added where the search line pierces the vertical surface through the horizon. The above process is illustrated in Figure 1, where Figures 1(a) and 1(b) are from the viewpoints of the parameter plane and 3-D space, respectively. In the diagram, a filled circle denotes an endpoint associated with a horizon vertex. The thick solid segment  $\overline{e_1e_2}$  is a coarse horizon edge. The star  $b$  at the middle of the coarse edge represents a base point. The dotted segment  $\overline{bp}$  is on the search line, which is normal to the coarse edge  $\overline{e_1e_2}$  and parallel to the parameter plane. The filled square  $p$  represents a new piercing point on horizon, the point at which the search line intersects the vertical surface through the horizon. The (signed) length of the dotted segment between the base point  $b$  and the piercing point  $p$  corresponds to the normal offset. The thin solid segments  $\overline{e_1p}$  and  $\overline{e_2p}$  connecting the endpoints of the coarse edge and the piercing point are refined horizon edges of the coarse horizon edge. The refined edges  $\overline{e_1p}$  and  $\overline{e_2p}$  form a polyline refinement of the coarse horizon edge  $\overline{e_1e_2}$ . Having located the piercing point, we also need to determine the horizon-bit and  $z$ -coordinate information associated with the newly added vertex. In this case (i.e., subdividing a horizon edge along the normal direction), the piercing point is the intersection of the search line with the vertical surface through the horizon. Therefore, the piercing point is always a point on horizon. For this reason, no horizon bit is required for the new vertex. Let  $z_1^-$  and  $z_1^+$  be the two  $z$  coordinates associated with one endpoint of the horizon edge, and  $z_2^-$  and  $z_2^+$  be the two  $z$  coordinates associated with the other endpoint of the edge. Then, the two  $z$  coordinates  $z^-$  and  $z^+$  associated with the new vertex are determined as  $\frac{1}{2}\{\min\{z_1^-, z_1^+\} + \min\{z_2^-, z_2^+\}\}$  and



**Figure 1:** Subdivision of a horizon edge along the normal direction. The (a) parameter plane view, and (b) 3-D view.



**Figure 2:** Subdivision of a nonhorizon edge along the normal direction. The (a) parameter plane view, and (b) 3-D view.

$\frac{1}{2} \{ \max\{z_1^-, z_1^+\} + \max\{z_2^-, z_2^+\} \}$ , respectively.

**NONHORIZON-EDGE SUBDIVISION.** Now, we consider the subdivision of a nonhorizon edge. Before proceeding, we need to first determine which of the  $z$  coordinates associated with the two endpoints of the nonhorizon edge are appropriate to use for determining the nonhorizon edge in 3-D. If both endpoints of the nonhorizon edge are associated with nonhorizon vertices, the  $z$  coordinates of the endpoints are unambiguously determined. Suppose now that one of the endpoints of the nonhorizon edge is associated with a horizon vertex. Since, at most, one vertex of a nonhorizon edge can be a horizon vertex, the other endpoint of the edge must be associated with a nonhorizon vertex. Let us assume that the horizon vertex is associated with the two  $z$  coordinates  $z_1^-$  and  $z_1^+$  and the nonhorizon vertex is associated with the  $z$  coordinate  $z_2$ . The  $z$  coordinate associated with the horizon vertex is chosen to be whichever of  $\{z_1^-, z_1^+\}$  is closer to  $z_2$ . Then the nonhorizon edge in 3-D is completely determined by its two endpoints and their chosen  $z$  coordinates. The base point is determined as the midpoint of this edge. The search line is through the base point, normal to the edge, and in the vertical plane containing the edge. The new point is added where the search line pierces the image surface. This process is illustrated in Figure 2, where Figures 2(a) and 2(b) are from the viewpoints of the parameter plane and 3-D space, respectively. The

unfilled circles  $e_1$  and  $e_2$  are the endpoints of an edge. The thick solid segment  $\overline{e_1 e_2}$  is a coarse nonhorizon edge. The star  $b$  represents the base point of the edge  $\overline{e_1 e_2}$ . The normal search line is perpendicular to  $\overline{e_1 e_2}$  and in the vertical plane containing  $\overline{e_1 e_2}$ . At the unfilled square  $p$ , where the search line intersects the image surface, a new nonhorizon vertex is added. Consequently, the two segments  $\overline{e_1 p}$  and  $\overline{p e_2}$ , shown by the thin solid segments, form a refinement of the coarse nonhorizon edge  $\overline{e_1 e_2}$ . To illustrate the adaptivity of nonhorizon-edge subdivision along the normal direction, Figure 2 also shows the refinement of a new nonhorizon edge  $\overline{p e_2}$  for the next level of subdivision. Through the corresponding base point  $b_1$ , the search line is perpendicular to the edge  $\overline{p e_2}$ . At the filled square  $p_1$ , the normal search line intersects the image surface. A piercing point  $p_1$  associated with a horizon vertex is located during the subdivision. By two iterations of subdivision to the coarse nonhorizon edge  $\overline{e_1 e_2}$ , a new horizon vertex  $p_1$  is found.

**EXCEPTIONAL CASES DURING SUBDIVISION.** As suggested earlier, some exceptional cases can occur during the refinement process. This is because, with the approach described above, the piercing point found in a normal direction does not always lead to a valid triangulation in the parameter plane. To avoid this and other problems, we must, in some exceptional circumstances, include an offset 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 capture which combination of normal/vertical directions is employed. Due to space constraints, the exceptional cases are not discussed further here. The interested reader is referred to the first author's Master's thesis [14] for a detailed treatment of these cases.

In all of the subdivision cases, once the search line is fixed, the offset is calculated by measuring the (signed) distance between the base point and piercing point. Offsets are signed quantities since these displacements can be in either of two directions along the search line. In all of the subdivision cases other than the horizon-edge case along the normal direction, once a piercing point is found, this point is projected vertically onto the parameter plane to obtain a corresponding (triangulation) vertex. The horizon bit for the offset associated with this new vertex is then set to one if the vertex is on a horizon and zero otherwise. The  $z$  coordinate associated with the new vertex is the  $z$  coordinate of the image surface at the piercing point.

It is worth noting that the normal search direction used in the subdivision of nonhorizon edges contributes very significantly to the fast location of new horizon vertices, since the normal direction tends to point towards a nearby function discontinuity if one exists. In essence, the choice of a normal search direction makes the subdivision process adaptive to the image data (i.e., data dependent). In contrast, if we were instead to perform these searches in the vertical direction, new vertices would always be added at the midpoints of the edges in the parameter-plane triangulation, making the fast location of new horizon vertices more difficult.

### C. Other Comments on the JBL Coder

The normal-mesh-based representation produced by the JBL coder is completely characterized by the base mesh, normal/vertical offsets, horizon bits, and direction values. Using this information, the corresponding mesh can be reconstructed. Since the resulting mesh representation is a surface defined on a continuous domain, a scan-conversion process is needed to convert the mesh data from the continuous domain to points on a raster grid. It is implied in [10] that planar interpolation is being used in the scan conversion by the JBL coder. By sampling the image surface on a regular grid (aligned with the centers of pixels in the parameter plane), a rasterized image is produced.

### D. Simplified JBL Coder

Having introduced the JBL coder, we now take a moment to briefly introduce a simplified version of this coder that is implicitly suggested in [10] (in the context of natural images). We refer to this simpli-

fied version of the coder by the name JBL-S herein. Conceptually, the key difference between the JBL and JBL-S coders is in how the original image surface is formed. In the JBL-S coder, the image surface is constructed using piecewise-planar interpolation rather than piecewise-constant interpolation. Due to the use of a piecewise-planar interpolant, the image surface has no discontinuities and hence no horizons exist either. Thus, the JBL-S coder is essentially the JBL coder without the horizon model. Unlike in the JBL case, horizon bits are not needed since horizon vertices are effectively not used in the JBL-S coder. Furthermore, in the JBL-S coder, the exceptional subdivision cases mentioned earlier can never occur, and subdivision always employs a normal search direction. Since the search direction is always normal, direction values are not required. Thus, in the case of the JBL-S coder, the normal-mesh-based image representation is completely characterized by only the base mesh and normal offsets.

#### IV. Our Implementation of the JBL Coder

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. In [10], the JBL coder is applied exclusively to piecewise-constant images, most likely because the horizons in such images can be detected with a very trivial edge detector. To apply the JBL coder to a larger class of images, however, a more sophisticated edge detector is required. In our implementation of the JBL coder, we employ a Canny edge detector [15] to assist in the identification of horizons. After the horizons have been identified, a normal-mesh-based representation is constructed, the corresponding normal/vertical offsets are quantized, and a final bit rate is estimated based on a simple implicitly-assumed coding scheme.

In our implementation, the normal/vertical offsets are quantized with a separate uniform scalar 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  $\Delta_k$  for the offsets of the  $k$ th subdivision level (where  $k = 0$  corresponds to the base mesh) is chosen as  $\Delta_k = q2^{k-1}$  for  $k \geq 1$ . Here,  $k \neq 0$  since no offsets are associated with the base mesh. This choice of  $\Delta_k$  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. This is due to the fact that errors in the reconstructed vertices and their height values introduced by offset quantization propagate from coarser to finer levels of the mesh.

In [10], the authors do not make any attempt to estimate the bit rate required to code the data of the normal-mesh-based image representation. In our work, however, we assume a simple coding scheme for the data and determine the corresponding bit rate based on entropy estimates. For the remainder of this section, let  $W$ ,  $H$ , and  $P$  denote the width, height, and number of bits per sample in the image being coded, and let  $L$  denote the number of subdivision levels. Table 1 identifies the mesh data that needs to be coded for the JBL coder. The general structure of the code stream used to encode this information is as follows. The code stream begins with a simple header which includes basic information such as  $W$ ,  $H$ ,  $P$ ,  $L$ , and the quantization parameter  $q$ . This is followed by the base-mesh information. Since the base mesh simply consists of four vertices corresponding to the four corners of the image bounding box, the  $x$  and  $y$  coordinates of the vertices can be derived from  $W$  and  $H$ . Only the  $z$  coordinates need to be included in the code stream, where  $P$  bits are used for each coordinate. Finally, the data for each subdivision level is appended to the code stream in order of increasing subdivision-level index. For each level of subdivision, the new-vertex information includes offsets, horizon bits, direction values, a scale parameter  $\lambda$  for offset data (to be discussed shortly), and the actual probabilities for normal-bit and direction-value data. The offsets, horizon bits, and direction values are assumed to be entropy coded

Table 1: Summary of mesh data for the JBL coder

Data Name	Type	Range
offset	$\mathbb{R}$	$\pm \max\{2^P - 1, \frac{W-1}{2}, \frac{H-1}{2}\}$
direction value	$\mathbb{Z}$	$\{0,1,2\}$
horizon bit	boolean	$\{0,1\}$
base mesh ( $z$ coordinate)	$\mathbb{Z}$	0 to $2^P - 1$

(e.g., by using arithmetic coding [16]).

Now, we explain how we estimate the rate for the entropy-coded parts of the code stream, namely the normal/vertical offsets, horizon bits, and direction values. First, we consider the normal/vertical offsets. Since the offsets have a symmetric sharply-peaked probability distribution with zero mean, we employ a (zero-mean) Laplacian distribution to model this information. In particular, we employ a probability density function  $p_{\text{off}}$  of the form

$$p_{\text{off}}(x) = \frac{\lambda}{2} e^{-\lambda|x|}, \quad (1)$$

where  $\lambda$  is a scale parameter. To determine the scale parameter  $\lambda$ , we measure the variance  $\sigma^2$  of the offset data and then match the variance of the Laplacian distribution to  $\sigma^2$ , yielding the choice  $\lambda = \sqrt{2}/\sigma$ . Since the offsets from different subdivision levels typically have distinct distributions, the offsets are modelled on a per-subdivision-level basis. Thus, we must choose a parameter  $\lambda_l$  for each subdivision level  $l$ , and in our implementation 32 bits are employed to represent each  $\lambda_l$ . The entropy  $E_{\text{off},l}$  of the quantized offsets at level  $l$  can be estimated as

$$E_{\text{off},l} = - \sum_{k=1}^n f_k \log_2 p_k, \quad (2)$$

where  $n$  is the total number of quantization bins used,  $f_k$  is the fraction of offsets quantized to the  $k$ th bin, and  $p_k$  is the probability of an offset being in the  $k$ th bin (for the  $l$ th subdivision level). For a quantization bin associated with the interval  $[a, b]$ , the quantity  $p_k$  is simply computed as  $p_k = \int_a^b p_{\text{off}}(x) dx$ , where  $p_{\text{off}}(x)$  is as defined in (1) with  $\lambda = \lambda_l$ .

Next, we consider the rate estimation for the horizon-bit and direction-value data. In this case, a first-order entropy estimate is used. The first-order entropy  $E$  of a source is calculated as

$$E = - \sum_{k=1}^n p_k \log_2 p_k, \quad (3)$$

where  $n$  is the alphabet size and  $p_k$  is the probability of the  $k$ th symbol. Let  $E_{\text{hor},l}$  and  $E_{\text{dir},l}$ , respectively, denote the entropies of the horizon-bit and direction-value data for the  $l$ th subdivision level. To compute  $E_{\text{hor},l}$  using (3), we let  $n = 2$  (since the alphabet is binary) and the probabilities  $\{p_k\}_{k=1}^2$  are set to the first-order probabilities of the actual horizon-bit data. To compute the entropy  $E_{\text{dir},l}$  using (3), we let  $n = 3$  (since the alphabet is ternary) and the probabilities  $\{p_k\}_{k=1}^3$  are set to the first-order probabilities of the actual direction-value data. In the case of both horizon-bit and direction-value data, the quantities  $\{p_k\}_{k=1}^{n-1}$  are included in the code stream, while  $p_n$  is not, as it can be deduced from the relationship  $\sum_{k=1}^n p_k = 1$ .

Given the above results, the total number  $R$  of bits required for the coded image can be computed as

$$R = \sum_{l=1}^L [(E_{\text{off},l} + E_{\text{dir},l})N_{\text{off},l} + E_{\text{hor},l}N_{\text{hor},l}] + R_{\text{overhead}}, \quad (4)$$

where  $N_{\text{off},l}$  and  $N_{\text{hor},l}$  are respectively the numbers of offsets and horizon bits at subdivision level  $l$ , and  $R_{\text{overhead}}$  is the number of bits devoted to other overhead information. The overhead information accounted for by  $R_{\text{overhead}}$  includes: a) 16 bits for each of  $W$  and  $H$ , b) 8 bits for each of  $P$  and  $L$ , c) 32 bits for the quantization parameter

$q$  (introduced above), d) 32 bits representing the parameter  $\lambda_l$  for each subdivision level  $l$ , e) 32 bits for the actual-probability information for horizon-bit data for each subdivision level, and f)  $32 \cdot 2 = 64$  bits for actual-probability information for direction-value data for each subdivision level. Since we use only first-order probabilities in the rate estimation, the rate does not depend on the data-scanning order used in the coding process.

Due to the similarities between the JBL and JBL-S coders as explained in Section III-D, the JBL-S coder is essentially implemented trivially as a special case of the JBL coder with a piecewise-planar image model. In this case, the set of horizons is empty, and consequently the horizon bits and direction values do not need to be coded.

### V. Shortcomings of the JBL Coder

Having introduced the JBL coder, we now discuss some of its shortcomings. By understanding the weaknesses of this coder, we can gain better insight into how we might improve upon them. The modifications to the coder that we propose later are motivated by the desire to overcome these deficiencies.

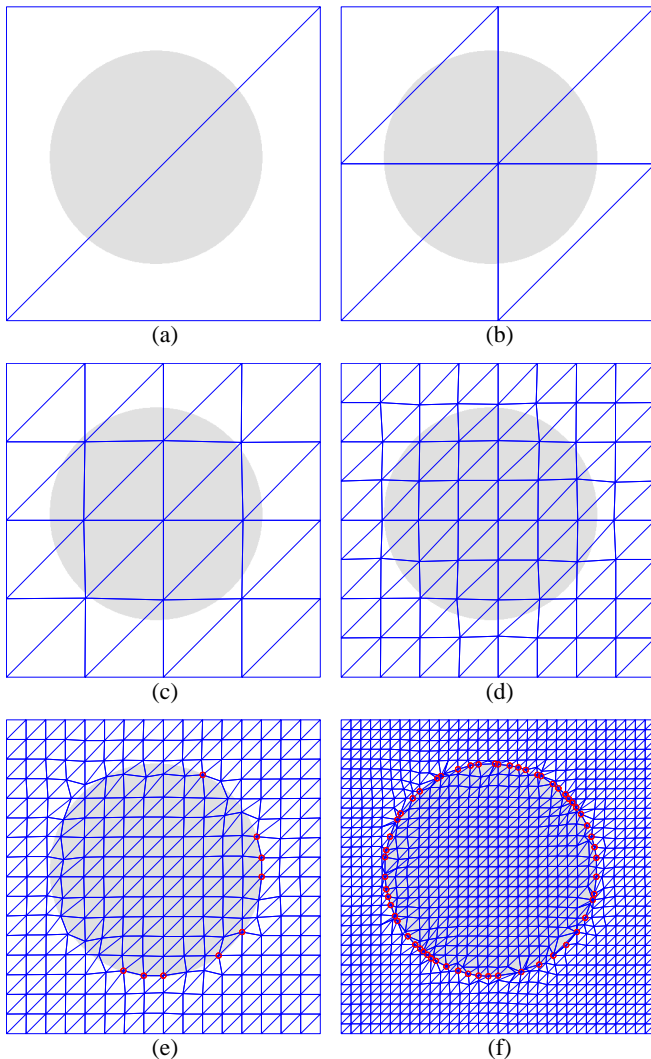
#### A. Choice of Base Mesh

One of the strengths of the JBL coder is its fast asymptotic error-decay rate for horizon-class images. This fast rate is, in part, due to the ability of the JBL coder to quickly locate new horizon vertices. Two different mechanisms are available to the JBL coder for assisting in the location of horizon vertices: 1) the choice of base mesh, and 2) the adaptivity inherent in normal subdivision. Unfortunately, the JBL coder (which uses a trivial data-independent base mesh) relies solely on the second of these mechanisms in order to quickly locate enough horizon vertices to form good polyline approximations of horizons. As a result, normal subdivision usually introduces many nonhorizon vertices and such vertices do not help to improve polyline approximations of horizons. Furthermore, many of these new nonhorizon vertices will also be positioned far away from any horizon, therefore, making a relatively smaller contribution to achieving a good surface approximation. This situation is undesirable, as it ultimately leads to reduced coding efficiency. Furthermore, this degradation in coding efficiency can be especially significant at low bit rates, where performance often depends very critically on the fast location of horizon vertices. In essence, the problem here is that, by using a data-independent base mesh (as opposed to a data-dependent one), the JBL coder severely restricts its ability to quickly locate horizons.

The above problem is illustrated by way of the example shown in Figure 3. In this example, we have the image `circle3` consisting of a single solid-gray circle that we wish to code using the JBL coder. The figure shows the mesh obtained after each of several levels of subdivision (superimposed on the original image). In a good polyline approximation of a horizon, triangle edges should not cross the horizon; they should instead be tangential to the horizon curve. Unfortunately, even after five levels of subdivision, the resulting very dense mesh does not form a particularly good polyline approximation of the circle boundary. There is, however, good reason to believe that, with an intelligently chosen data-dependent base mesh, a better approximation of horizons can be achieved.

#### B. Normal/Vertical Offset Format

The second shortcoming of the JBL coder involves the representation it employs for normal/vertical offsets. An offset measures the distance from a base point to its associated piercing point. Since neither the base point nor the piercing point falls on an integer grid, they can be anywhere on the image surface. The offset measuring the distance between the two points is a real number.



**Figure 3:** Ineffectiveness of a data-dependent base mesh for the `circle3` image. The original image superimposed on the (a) base mesh, and the mesh after (b) one, (c) two, (d) three, (e) four, and (f) five levels of subdivision.

By further observation, we notice that the image surface is not arbitrary. Due to the piecewise-constant interpolation used to generate the image surface, any point on the surface has at least one of its  $x$ ,  $y$ , or  $z$  coordinates being integer. This suggests that the JBL coder might be improved by exploiting this special property of the image surface.

#### C. Scan Conversion

The third shortcoming of the JBL coder involves the scheme employed for scan conversion. Ideally, we desire a scan-conversion scheme that preserves both smooth regions in an image and sharp intensity changes along horizons. The piecewise-planar interpolation scheme employed by the JBL coder yields an interpolant that is smooth within each triangular domain (of the parameter-plane triangulation) but is not usually smooth at the boundaries of these domains, due to mismatches in partial derivatives along the boundaries of neighboring domains. A higher-order interpolation scheme could improve the smoothness of the interpolant along domain boundaries. For this reason, there is quite likely room for improvement in the scan-conversion method employed by the JBL coder.

## VI. Proposed Modifications to the JBL Coder

Having identified some shortcomings of the JBL coder, we now propose three modifications to it in order to overcome these weaknesses. As we will later show, each of these changes leads to improved coding performance.

### A. Choice of Base Mesh

Our first modification to the JBL coder affects the choice of base mesh. In particular, we propose the use of an intelligently chosen data-dependent base mesh (instead of a data-independent one). As mentioned earlier, a data-dependent base mesh can help to achieve good polyline approximations of horizons using relatively few vertices. In simple terms, our base-mesh generation method uses image-edge information to produce a set of horizon vertices to be employed in the base mesh. These horizon vertices along with some extra points are then triangulated to form the base mesh. In what follows, we describe our base-mesh generation method in more detail.

Since our base-mesh generation method requires image-edge information, the first step in our method is to locate all of the edge pixels in the image. This is accomplished by using a Canny edge detector. To avoid potential problems in subsequent processing, any intersecting edges are split at their intersection points. Thus, the output of the edge-detection process is always a set of edges that do not intersect each other except possibly at their endpoints.

Once the image edges have been found, for each edge, we must select a subset of its pixels that effectively capture its shape, called a critical set. To do this, curvature information for the edge is employed. Using the method of [17], we compute an estimate of the curvature of the edge at each of its constituent pixels. Initially, we select as the critical set, the first and last pixels of the edge as well as every pixel whose curvature value is above a certain threshold. In this way, an edge with sharp cusps is divided into several pieces. This approach solves the potential problem of needing many pixels to form a good polyline approximation at places with large curvature. The preceding critical set is then augmented by including more pixels such that the distance between the neighboring pixels is smaller than the reciprocal of the local curvature of the edge. In this way, we include more pixels in regions where the edge bends sharply and fewer pixels in regions where the edge is relatively straight.

Once the critical sets of pixels for the image edges have been determined, this information must be mapped into geometric structures in the continuous domain. For each edge, the critical set of pixels is converted into a polyline approximation of a horizon in the continuous domain. This is accomplished as follows. First, pixels from the critical set are mapped to points in the parameter plane. Then, the resulting points are joined by line segments in such a way as to maintain the same connectivity that these points have on the image edge from which they were derived. Since pixel centers are aligned on the lattice  $\frac{1}{2}\mathbb{Z}_{\text{odd}}$  and horizons fall on (unit-square) pixel boundaries, horizon points will always have at least one of their  $x$  or  $y$  coordinates being integer. This preceding discrete-to-continuous-domain mapping process yields a set of polyline approximations of horizons. In what follows, let  $V$  and  $E$  denote, respectively, the set of horizon points and line segments associated with the polyline approximations of horizons.

The last step in our base-mesh generation method is to produce a triangulation of the parameter plane from which the base-mesh is trivially obtained. In particular, we want to construct a triangulation containing  $V$  as triangle vertices and  $E$  as triangle edges. Since some edge constraints are imposed on the triangulation process, we cannot use a Delaunay triangulation (DT) (since a DT may not exist). In an attempt to obtain a triangulation with good angle properties, we use a constrained DT. Furthermore, for any given set of points and constrained segments, we would like to produce a constrained DT that is unique. If the triangulation is unique, we do not need to code the com-

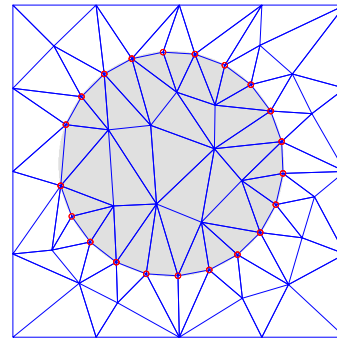


Figure 4: Example of a data-dependent base mesh for the `circle3` image.

plete connectivity information for the triangulation, as knowing only the constrained segments will suffice. This is highly desirable, as any additional connectivity information that must be coded will negatively impact coding efficiency. For this reason, we employ a constrained DT with preferred directions [18] in order to ensure the uniqueness of the triangulation.

Although we could use a constrained DT of the points  $V$  and segments  $E$  to form the base mesh, we elect not to do so. Instead, we add some extra points  $V_s$  (called Steiner points) to  $V$ . Then, we perform a constrained DT of the points  $V \cup V_s$  and segments  $E$ . This further improves the quality of the triangulation by obtaining a more uniform vertex distribution and avoiding sliver triangles. The Steiner points  $V_s$  are generated by the Triangle software [19]. Unfortunately, the Steiner points have real  $x$  and  $y$  coordinates. To overcome the inefficiency of coding real coordinates, the coordinates of Steiner points are rounded to the nearest pixel centers. In other words, the rounded Steiner points have both  $x$  and  $y$  coordinates being elements of  $\frac{1}{2}\mathbb{Z}_{\text{odd}}$ . Clearly, no Steiner points could be a horizon vertex, since none of the coordinates of the Steiner points is an integer. Therefore, the rounding operation also eliminates the need for storing horizon bits pertaining to the Steiner points. Furthermore, since the rounding of the Steiner point coordinates changes the vertex geometry only very slightly, the good vertex distribution is maintained in the data-dependent base mesh.

As it turns out, due to the geometry of the vertices in our data-dependent base mesh, many edges in the constrained DT are also edges in the DT (i.e., many constraints are inactive). The large number of inactive constraints can be partly attributed to the short segments in  $E$  and good distribution of vertices in the planar straight line graph. As an optimization, we only encode the constrained segments that affect the resulting constrained DT. In this way, the (constrained-segment) information that needs to be coded for the base mesh can be significantly reduced.

To illustrate the benefits of a data-dependent base mesh, we provide an example of a base mesh generated by our (above) method. In particular, we consider the `circle3` image from an earlier example. For this image, the base mesh produced using our method is shown in Figure 4 (superimposed on the original image). Although the mesh contains relatively few vertices, it still manages to provide a good approximation of the horizon (i.e., circle boundary). For comparison purposes, recall the earlier results from Figure 3, which show the refined meshes for the same image obtained from a data-independent base mesh. Evidently, in spite of having significantly fewer vertices, the base mesh in Figure 4 generated using our method has a much better polyline approximation of the horizon than the refined mesh in Figure 3(f) generated from a data-independent base mesh. This example clearly demonstrates that, by carefully choosing the base mesh using our method described above, superior polyline approximations of horizons can be obtained with fewer vertices (compared to when a data-independent base mesh is employed).

## B. Normal/Vertical Offset Format

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 identify each piercing point with an integer instead of a real number. Recall that an offset is a real value measuring the distance between a base point and its corresponding piercing point. Due to the piecewise-constant interpolation process used to generate the 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 base point, all possible piercing points can be identified, by finding all intersections of the search line with planes of the form  $x = c$ ,  $y = c$ , and  $z = c$ , where  $c \in \mathbb{Z}$ . In this way, all possible piercing points can then be enumerated with an integer index. This index can be used to specify which of the possible piercing points is the actual piercing point. By using an integer index instead of a real number for each offset, coding efficiency can likely be improved.

## C. Scan Conversion

Our third modification to the JBL coder is in the interpolation scheme used for scan conversion. In short, we propose the use of a higher-order interpolant in order to improve the smoothness of the reconstructed images at the boundaries of triangular domains in the parameter-plane triangulation, while still maintaining sharp image edges. Our method is based on the bicubic interpolation technique described in [20, pp. 446-449], which yields  $C^1$ -continuous surfaces (i.e., surfaces with continuous first-order partial derivatives).

Since we wish to preserve sharp edges in the image, we cannot simply apply the above technique from [20] without modification, as this would have the undesirable effect of badly blurring edges. To avoid unnecessarily blurring edges, we modify the behavior of the preceding technique in the vicinity of horizons. In effect, this leads to two distinct cases, depending on whether or not the triangular domain being processed borders on a horizon edge. These two cases are illustrated in Figures 5(a) and 5(b). In each case, part of the parameter-plane triangulation is shown, and the triangular domain over which we wish to form an interpolant is denoted by a vertically-hatched triangle. The solid-white and solid-gray areas denote two different regions separated by a horizon.

In the first case, shown in Figure 5(a), the triangular domain being processed does not border on a horizon edge. Here, we directly apply the method of [20], which generates an interpolant that passes through the three points associated with the three vertices of the triangle in the mesh, and also has first-order partial derivatives that are continuous along the boundary of (as well as inside) the triangular domain. All 1-ring neighbors (i.e., diagonally-hatched triangles) of the triangular domain being processed are used to determine the necessary partial derivative information. That is, all vertex/height-value pairs in and on the boundary of the diagonally-hatched regions are used for interpolation.

In the second case, shown in Figure 5(b), the triangular domain being processed borders on a horizon edge. Here, when determining the interpolant for a particular triangular domain, we use only vertex/height-value pairs from the same side of the horizon as the triangular domain being processed. The region comprised of neighbouring triangles (i.e., the diagonally-hatched region) straddles the horizon. We use only vertex/height-value pairs in and on the boundary of the diagonally-hatched white-shaded regions for interpolation, since the triangular domain being processed is also from the same side of the horizon (i.e., the white-shaded as opposed to gray-shaded region).

By combining the interpolants for each of the individual triangular domains, a complete interpolated image surface is obtained. By sampling this surface on a rectangular grid aligned to the pixel centers in the parameter plane, a rasterized image is generated.

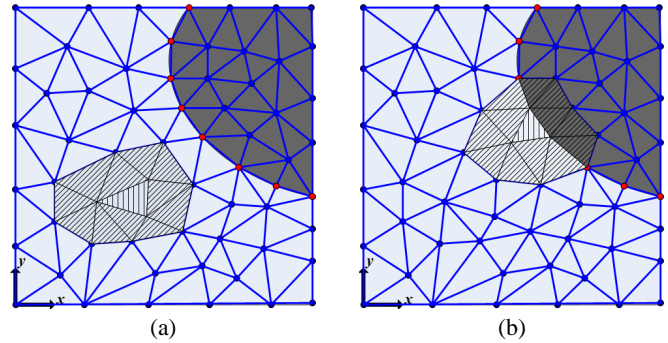


Figure 5: Two cases for the proposed interpolation scheme. Interpolation (a) away from any horizons, and (b) near a horizon.

Table 2: Summary of features for the various coders

Coder	Image Interpolant	Model	Base Mesh	Offset Format	Scan Conversion
JBL-S	piecewise planar	non-horizon	data independent	$\mathbb{R}$	piecewise planar
JBL	piecewise constant	horizon	data independent	$\mathbb{R}$	piecewise planar
Enhanced	piecewise constant	horizon	data dependent /independent	$\mathbb{R}$ or $\mathbb{Z}$	piecewise planar/bicubic

Table 3: Summary of data for the enhanced coder

Data Name		Type	Range of Value	
offset		$\mathbb{R}$ or $\mathbb{Z}$	$\pm \max\{2^P - 1, \frac{W+H}{2}\}$	
direction value		$\mathbb{Z}$	$\{0,1,2\}$	
horizon bit		boolean	$\{0,1\}$	
base mesh	number of horizon vertices <sup>†</sup>	$\mathbb{Z}$	—	
	number of Steiner points <sup>†</sup>	$\mathbb{Z}$	—	
	$x^\dagger$	horizon	$\frac{1}{2}\mathbb{Z}$	
	Steiner	$\frac{1}{2}\mathbb{Z}_{\text{odd}}$	0 to $W - 1$	
	$y^\dagger$	horizon	$\frac{1}{2}\mathbb{Z}$	0 to $H - 1$
	Steiner	$\frac{1}{2}\mathbb{Z}_{\text{odd}}$	0 to $H - 1$	
$z$		$\mathbb{Z}$	0 to $2^P - 1$	
active line segment <sup>†</sup>		$\mathbb{Z}$	—	

<sup>†</sup>only employed when using data-dependent base mesh

## VII. Enhanced Coder

In order to facilitate the further analysis of our three proposed JBL-coder modifications, we added support for these changes to our original implementation of the JBL coder, resulting in what we refer to as our enhanced coder. For convenience, we summarize the main features of the various coders (i.e., the JBL, JBL-S, and enhanced coders) in Table 2. The enhanced coder supports all combinations of data-independent/data-dependent base mesh, real/integer offsets, and planar/bicubic scan conversion. In what follows, we introduce some details regarding the enhanced coder. Since the enhanced coder is similar in many ways to the original version, we focus our attention only on the details that differ.

Table 3 shows the data that needs to be coded for the enhanced coder. The real or integer offset data is assumed to be entropy coded, where the corresponding bit rate can be estimated using (2). In the case that integer offsets are employed, the specified quantizer step size parameter  $q$  should satisfy  $q \geq 1$ . If  $q = 1$ , quantization is effectively bypassed, and no information is discarded by quantization. The bit rate corresponding to the direction-value and horizon-bit data can be estimated using (3), as in the JBL coder. The total number of bits required for the coded image can be calculated using (4) with  $R_{\text{overhead}}$  being modified to include the extra information needed for the data-

dependent base-mesh (e.g., the numbers of horizon vertices and Steiner points in the base mesh, vertex locations, height values, and active-constraint segments for the constrained DT).

Consider now the coding of the base mesh. Here, we focus only on the case of a data-dependent base mesh, since the data-independent case is simply handled as described earlier in Section IV. We use 16 bits for each of the numbers of horizon vertices and Steiner points in the data-dependent base mesh. In the subsequent discussion, let  $W$ ,  $H$ , and  $P$  denote the width, height, and the number of bits per sample of the original image, respectively. Since the  $x$ ,  $y$  and  $z$  coordinates of vertices of the base mesh tend to have fairly uniform distributions, we choose to not use any entropy coding for this data. For the horizon vertices in the base mesh, each of the  $x$  and  $y$  coordinates is an element of  $\frac{1}{2}\mathbb{Z}$ . Furthermore, we know that if the  $x$  coordinate is an integer, the corresponding  $y$  coordinate must be an element of  $\frac{1}{2}\mathbb{Z}_{\text{odd}}$ , and vice versa. Consequently, we need only code the (single-bit) fractional part of either the  $x$  or  $y$  coordinate. In our case, we choose to code only the fractional part of the  $x$  coordinate. Thus, each of the  $x$  and  $y$  coordinates is represented using  $\lceil \log_2 W \rceil + 1$  and  $\lceil \log_2 H \rceil$  bits, respectively. In the case of Steiner points in the base mesh, we observe that such points always have  $x$  and  $y$  coordinates each being an element of  $\frac{1}{2}\mathbb{Z}_{\text{odd}}$ . Thus, we need not code the (single-bit) fractional part of these values (which is always one). So, for Steiner points, we use  $\lceil \log_2 W \rceil$  and  $\lceil \log_2 H \rceil$  bits to represent each  $x$  and  $y$  coordinate, respectively. The four corner points of the image bounding box are always chosen as vertices in the base mesh. Their  $x$  and  $y$  coordinates can be derived from the size of the image, and therefore do not need to be coded. Each  $z$  coordinate in the base mesh is represented using  $P$  bits. The (active) constrained line segments for the base mesh are coded as a list of pairs of vertex indices, where vertices are indexed according to their order of appearance in the code stream.

The code stream format employed by the enhanced coder is very similar to that of the original coder. Some basic information is included in the header (e.g.,  $W$ ,  $H$ , etc.). Then, the  $x$ ,  $y$ , and  $z$  coordinates of the base mesh are coded as explained above. Finally, the data associated with different levels of subdivision are coded, starting from data associated with the smallest subdivision-level index and proceeding to the data associated with the largest subdivision-level index.

Note that, since we do not entropy code the base mesh information, there are probably more efficient schemes for handling this data. At high bit rates, the base mesh information constitutes only a small fraction of the entire code stream. Consequently, the price paid for not entropy coding the base mesh information is relatively small. At low bit rates, however, the base mesh information can consume a more significant fraction of the entire code stream. So, the cost of not entropy coding the base mesh information is more significant in this case. In spite of our choice not to entropy code the base mesh data, however, we still obtain reasonably good performance at low rates. We leave it as a subject of future work to explore the use of more sophisticated coding schemes for the base mesh information (perhaps, by performing differential coding of the vertex coordinates).

## VIII. Experimental Results

Earlier, we suggested that our proposed modifications to the JBL coder improve its performance. In the sections that follow, we support our claim through experimental evidence. Although numerous 8-bit grayscale test images were employed in our work<sup>1</sup>, we focus our attention on the results for two representative images herein, namely, the `paw` and `peppers` images. The `peppers` image is taken from the well known USC image database [21] (and has dimensions of  $512 \times 512$ ), while the `paw` image is our own synthetic test image with dimensions of  $1024 \times 1024$ .

<sup>1</sup>For details on the other test images employed in our work, see [14].

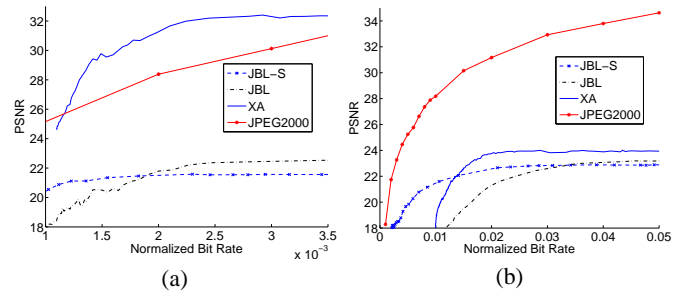


Figure 6: Coding performance for the (a) `paw` and (b) `peppers` images using the JBL-S, JBL, XA, and JPEG-2000 methods.

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 evaluate the performance of each of our three proposed modifications to the JBL coder in turn.

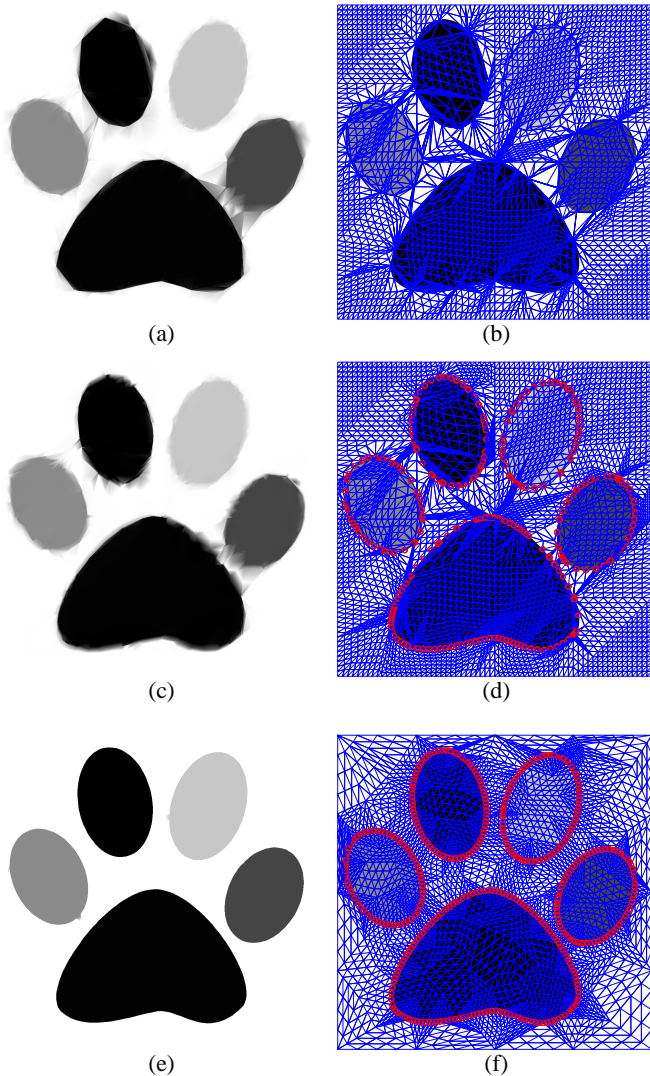
### A. Choice of Base Mesh

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 were 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) JPEG-2000 [3] 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. In what follows, 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 Figure 6. From these results, we can see that, at high bit rates, the XA coder outperforms the JBL and JBL-S coders, and the XA coder even outperforms the JPEG-2000 coder in the case of the `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). As mentioned earlier, 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 bit rates can be attributed to the higher efficiency of the horizon model, while the inferior performance at low bit 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 Figures 7 and 8, respectively. In the first case, the final mesh employed by each method is shown superimposed on the original image with the horizon vertices

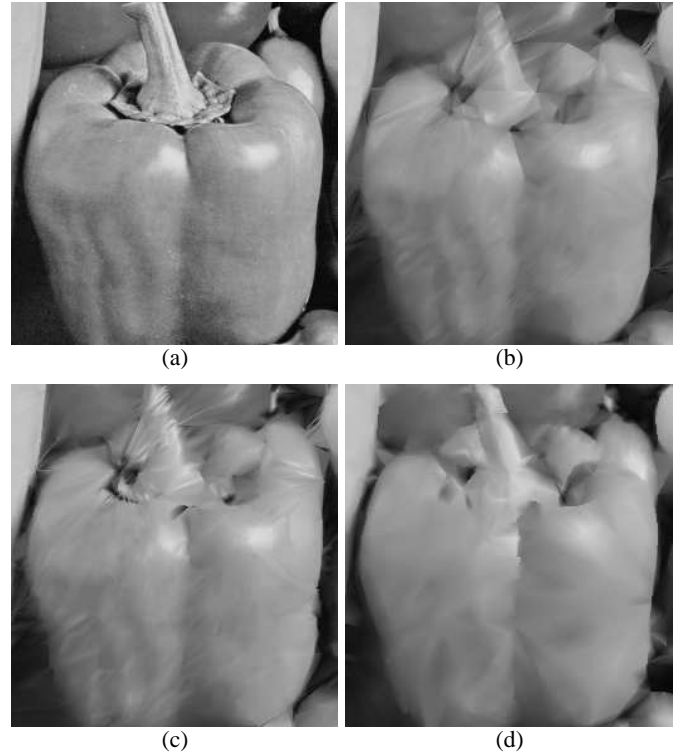




**Figure 7:** Coding example for the `paw` image. Lossy reconstructions obtained at about 400:1 compression using the (a) JBL-S, (c) JBL, and (e) XA methods. The corresponding final meshes employed by the (b) JBL-S, (d) JBL, and (f) XA methods.

denoted by circles. Examining the results for the `paw` image in Figure 7, 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 horizons much better than the JBL and JBL-S methods. Examining the reconstructions of the `peppers` image in Figure 8, 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 Figure 7. 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 Figure 7(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



**Figure 8:** 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.

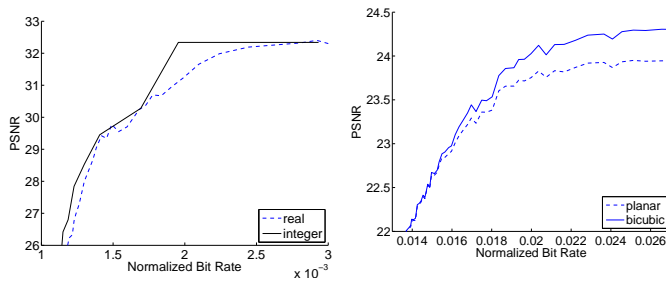
in the reconstructed image for the XA coder can be attributed to inaccuracies in the estimation of the location and curvature of horizons.

## B. Real Versus Integer Offsets

Now, we consider our second proposed change to the JBL coder, which is to use integers rather than real numbers to represent normal/vertical offsets. To evaluate the effectiveness of our proposed change, several test images were coded with the XA coder at various bit rates using both real and integer representations of offsets. Some representative results obtained in the case of the `paw` image (using two levels of subdivision) are shown in Figure 9. From this graph, 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).

## C. Planar Versus Bicubic Interpolation

Lastly, we consider our third proposed change to the coder, which is to use bicubic instead of planar interpolation for scan conversion. To assess the value of this change, both planar and bicubic interpolation were employed in the XA coder to compress numerous images at various bit rates. Some representative results obtained in the case of the `peppers` image (using two levels of subdivision) are shown in Figure 10. From these results, we can see that bicubic interpolation outperforms planar interpolation, especially at high 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.



**Figure 9:** Coding performance for the paw image using real and integer offsets.

**Figure 10:** Coding performance for the peppers image using planar and bicubic interpolation. The graph plots PSNR (y-axis, 22 to 24.5) against Normalized Bit Rate (x-axis, 0.014 to 0.026). Two curves are shown: 'planar' (dashed line) and 'bicubic' (solid line). The 'bicubic' curve shows higher PSNR for a given bit rate compared to the 'planar' curve.

## IX. 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.

## References

- [1] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.
- [2] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.
- [3] *ISO/IEC 15444-1: Information technology—JPEG 2000 image coding system—Part 1: Core coding system*, 2000.
- [4] D. L. Donoho, "Orthonormal ridgelets and linear singularities," *Journal on Mathematical Analysis of the Society for Industrial and Applied Mathematics*, vol. 31, no. 5, pp. 1062–1099, 2000.
- [5] E. Candès and D. Donoho, *Curvelets: A Surprisingly Effective Nonadaptive Representation of Objects with Edges. In Curves and Surfaces*. Nashville, TN, USA: Vanderbilt University Press, 1999.
- [6] D. Donoho and X. Huo, "Combined image representation using edgelets and wavelets," in *Wavelet Applications in Signal and Image Processing VII, in Proceedings SPIE*, vol. 3813, Denver, Colorado, Oct. 1999, pp. 468–476.
- [7] M. N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2091–2106, Dec. 2005.
- [8] D. L. Donoho, "Wedgelets: nearly minimax estimation of edges," *Annals Statistics*, vol. 27, no. 3, pp. 859–897, 1999.
- [9] E. L. Pennec and S. Mallat, "Sparse geometric image representation with bandelets," *IEEE Transactions on Image Processing*, vol. 14, no. 4, pp. 423–438, Apr. 2005.
- [10] 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, no. 1, pp. 92–130, 2005.
- [11] D. Xu and M. D. Adams, "An improved multiscale normal-mesh-based image coder," in *Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria, BC, Canada, Aug. 2007, pp. 50–53.
- [12] J. O'Rourke, *Computational Geometry in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1998.
- [13] I. Guskov, K. Vidimce, W. Sweldens, and P. Schroder, "Normal meshes," in *Computer Graphics Proceedings*, 2000, pp. 95–102.
- [14] D. Xu, "Improved subband-based and normal-mesh-based image coding," M.A.Sc. thesis, Dept. of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, Aug. 2007. [Online]. Available: <http://www.ece.ubc.ca/~dixu>
- [15] 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.
- [16] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, June 1987.
- [17] M. Marji and P. Sisy, "A new algorithm for dominant points detection and polygonization of digital curves," *Pattern Recognition*, vol. 36, no. 10, pp. 2239–2251, Oct. 2003.
- [18] C. Dyken and M. S. Floater, "Preferred directions for resolving the non-uniqueness of Delaunay triangulations," *Computational Geometry: Theory and Applications*, vol. 34, no. 2, pp. 96–101, 2006.
- [19] 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.
- [20] T. Y. Yang, *Finite Element Structural Analysis*. Englewood Cliffs: Prentice Hall, 1986.
- [21] "The USC-SIPI image database," <http://sipi.usc.edu/database/>, 2007.



**Di Xu** was born in Jilin City, Jilin, China. She received the B.S. degree in electrical engineering from Beijing Normal University, Beijing, China in 2003, and the M.A.Sc. degree in electrical and computer engineering from the University of Victoria, Victoria, BC, Canada in 2007. She is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering at the University of British Columbia, Vancouver, BC, Canada. Her research interests include multimedia signal processing, image coding, wavelets, filter banks, and digital geometric processing.



**Michael D. Adams** was born in Kitchener, ON, Canada. He received the B.A.Sc. degree in computer engineering from the University of Waterloo, Waterloo, ON, Canada in 1993, the M.A.Sc. degree in electrical engineering from the University of Victoria, Victoria, BC, Canada in 1998, and the Ph.D. degree in electrical engineering from the University of British Columbia, Vancouver, BC, Canada in 2002. From 1993 to 1995, Michael was a member of technical staff at Bell-Northern Research (now Nortel Networks) in Ottawa, ON, Canada where he developed real-time software for fiber-optic telecommunication systems. Since 2003, Michael has been Assistant Professor with the Department of Electrical and Computer Engineering at the University of Victoria, Victoria, BC, Canada.

Michael is the recipient of a Natural Sciences and Engineering Research Council (of Canada) Postgraduate Scholarship. He is a voting member of the Canadian Delegation to ISO/IEC JTC 1/SC 29 (i.e., Coding of Audio, Picture, Multimedia and Hypermedia Information), and has been an active participant in the JPEG-2000 standardization effort, serving as co-editor of the JPEG-2000 Part-5 standard and principal author of one of the first JPEG-2000 implementations (i.e., JasPer). His research interests include multimedia signal processing, wavelets, and multirate signal processing.