SHAPE-ADAPTIVE CODING USING BINARY SET SPLITTING WITH K-D TREES

James E. Fowler
Department of Electrical and Computer Engineering
GeoResources Institute, Mississippi State ERC
Mississippi State University, Mississippi State, MS

ABSTRACT

The binary set splitting with k-d trees (BISK) algorithm is introduced. An embedded wavelet-based image coder based on the popular bitplane-coding paradigm, BISK is designed specifically for the coding of image objects with arbitrary shape. While other similar algorithms employ quadtree-based set partitioning to code significance-map information, BISK uses a simpler, and more flexible, binary decomposition via k-d trees. Additionally, aggressive discarding of transparent regions is implemented by shrinking sets to the bounding box of their constituent opaque coefficients before further partitioning. Empirical results indicate that the proposed BISK coder consistently yields efficient performance when compared to a variety of other shape-adaptive coders.

1. INTRODUCTION

The coding of imagery with shapes more general than rectangular frames has become an important issue in multimedia communications. Indeed, shape-adaptive coding of still-image and video objects is a fundamental aspect of the recent MPEG-4 standard. In other applications too, shape-adaptive coding plays an important role. For example, the oceanographic imagery considered in [1] has shapes defined by coastlines and ocean floors. Arbitrarily shaped data also arises in the medical field, e.g., chromosome imagery [2]. These and other applications can benefit from efficient coding of image objects with arbitrary shape.

The straightforward approach to handling arbitrarily shaped image objects is to consider pixels within the object as "opaque" and those outside the object as "transparent," and to adapt rectangular-frame coders to process only the opaque regions within the bounding box of the object. To this end, in modern embedded wavelet coders, one employs a shape-adaptive discrete wavelet transform (SA-DWT) [3] that transforms only opaque regions in the image.

In this paper, we introduce a new image-coding algorithm designed specifically for shape-adaptive coding. Our proposed technique, binary set splitting with k-d trees (BISK), can be considered to be a variant of the well known SPECK algorithm [4] and its shape-adaptive extension, object-based SPECK (OB-SPECK) [5]. The primary difference between our BISK coder and SPECK lies in the fact that SPECK employs a quadtree structure for coding of the significance map that arises during embedded bitplane coding, while BISK uses k-d trees [6], a somewhat simpler set-decomposition paradigm that is particularly suited to shape-adaptive coding. Additionally, our BISK coder instigates an aggressive discarding of transparent regions from consideration by "shrinking" decomposition sets to the bounding box of their opaque coefficients.

This work was funded in part by the National Science Foundation under Grant No. ACI-9982344.

In the remainder of this manuscript, we present our BISK algorithm in detail and include a variety of empirical results that indicate that the proposed BISK coder consistently offers efficient coding performance when compared to a number of existing shape-adaptive coders.

2. TECHNIQUES FOR SHAPE-ADAPTIVE CODING

In this section, we survey a variety of prior proposals for shape-adaptive coding. As we are focusing on embedded wavelet-based image coders, we consider a number of prominent approaches to significance-map coding and extensions made to them to enable shape-adaptive coding. The straightforward approach is to consider transparent regions to be permanently "insignificant" such that the significance and refinement passes of the coder can then process these transparent regions in a manner identical to that of other insignificant coefficients. Although most approaches are based on this general idea, depending on the particular method of significance-map coding involved, additional refinements are often possible to increase performance.

Zerotrees—The most prominent zerotree coder to appear in the literature is perhaps SPIHT [7]. The straightforward shape-adaptive version of SPIHT follows the approach described above for shape-adaptive coding—transparent regions are considered to be permanently insignificant, as in [8]. In this case, zerotree structures aggregate large regions of transparent coefficients into zerotree symbols along with the opaque insignificant coefficients. A further refinement on this basic approach observes that greater efficiency can be obtained by discarding all sets of coefficients that lie entirely within a transparent region from further consideration [9]. A somewhat similar approach was taken in the zerotree coder for still-image textures specified by the MPEG-4 standard [3].

Conditional Coding—A recent alternative to zerotrees is conditional coding of the significance map. Specifically, known significance states of neighboring coefficients provide a context for the coding of the significance state of the current coefficient with an adaptive arithmetic coder. Although the most prominent example of such techniques, JPEG-2000, does not support arbitrarily shaped image coding, the underlying EBCOT algorithm is easily made shape-adaptive. In shape-adaptive EBCOT [2], transparent regions are ignored and not coded in all coding passes, while anytime that the context for an opaque coefficient overlaps the object boundary, transparent coefficients in the context are treated as insignificant.

Runlength Coding—The significance map forms a binary image; consequently, techniques that have long been employed for the coding of bilevel images are applicable to significance-map coding, for example, runlength coding. The Wavelet Difference Reduction (WDR) [10] algorithm combines runlength coding of the significance map with an efficient lossless representation of the runlength symbols to produce an embedded image coder. The

runlength-coding approach to embedded wavelet coding is easily made shape-adaptive by having the raster-scan processing "skip" over transparent regions by not coding any significance information for them or including them in the length of the runs [1].

Density Estimation—A unique approach to significance-map coding was proposed recently: the tarp coder [11] uses a nonadaptive arithmetic coder coupled with an explicit probability estimate of the significance map. Specifically, an arithmetic coder codes the significance state for each coefficient using the probability that that coefficient is significant as determined by a nonparametric density-estimation procedure. In [11], this density estimation is efficiently computed by a novel series of 1D filtering operations coined tarp filtering. In [12], the tarp coder was modified to handle shape-adaptive coding by "skipping" over the transparent regions while maintaining the current probability estimate unchanged.

Spatial Partitioning—Recent algorithms have dispensed with cross-scale aggregation of coefficients (e.g., zerotrees) in favor of partitioning sets of contiguous coefficients from within a single subband. A prominent algorithm of this class is SPECK [4], in which the significance-map is coded with quadtrees, a well known method of spatial partitioning. In quadtree partitioning, the significance state of an entire block of coefficients is tested and coded, the block is subdivided into four subblocks of approximately equal size, and the significance-coding process is repeated recursively on each of the subblocks. Object-based SPECK (OB-SPECK) [5] is a shape-adaptive variant which follows the methodology of shape-adaptive SPIHT [9]; that is, transparent regions are considered to be permanently insignificant, and when a set contains no opaque coefficients, it is pruned from the quadtree decomposition.

3. BINARY SET SPLITTING WITH K-D TREES (BISK)

In this section, we introduce a simplified variant of OB-SPECK designed specifically for improved shape-adaptive coding. The proposed coder, called BISK, employs two tactics to this end—aggressive discarding of transparent regions from sets after partitioning, and a spatial-partitioning structure more flexible than quadtrees. Specifically, when processing a given set, the BISK coder "shrinks" the set to the bounding box surrounding the opaque coefficients before subdividing the set into smaller blocks. Additionally, a new partitioning scheme is employed—k-d trees, rather than quadtrees, are used to recursively code the significance map.

3.1. k-d Trees

Like quadtrees, k-d trees [6] are a recursive spatial-partitioning data structure. Unlike quadtrees, which subdivide a block into four equally sized subblocks, k-d trees effectuate a binary partitioning; that is, blocks are divided in two, rather than quarters. Although there are several approaches for selecting the location and orientation of the split, we use a simple approach in which blocks are divided into approximately equally sized halves, and we alternate between splitting horizontally and vertically. For example, if a given block is halved vertically, then its two subblocks will be halved horizontally. Specifically, let a set of coefficients, $\mathcal{S}$, have decomposition level $l(\mathcal{S})$ in the k-d tree. When $\mathcal{S}$ is split, the two resulting subsets, $\mathcal{S}_1$ and $\mathcal{S}_2$, will reside at level $l(\mathcal{S})+1$ of the k-d tree. Set $\mathcal{S}$ is split either horizontally or vertically depending on whether $l(\mathcal{S})$ is even or odd, respectively. We note that a k-d tree can achieve a partition identical to that of a quadtree but will require twice as many levels of subdivision to do so.

3.2. The BISK Algorithm

The BISK algorithm starts by splitting the set of transform coefficients, $\mathcal{X}$, into individual subbands $\mathcal{S}$ which are then placed in a list of insignificant sets (LIS). Afterward, the BISK algorithm follows the common bitplane-coding paradigm consisting of sorting and refinement passes. The algorithm is as follows:

```
procedure BISK(X)
  Initialization(X)
  n ← max bitplane
  while (true)
    SortingPass()
    RefinementPass()
    n ← n − 1
procedure Initialization(X)
  for each subband S in X
    l(S) ← transform level of S
    Shrink(S)
    append S to LIS_{l(S)}
  LSP ← ∅
procedure SortingPass()
  l = number of LIS lists
  while l > 0
    for each S ∈ LIS_l
      Process(S)
    l ← l − 1
procedure RefinementPass()
  for each S ∈ LSP
    output n^th bitplane value of coefficient magnitude
```

Above, $l(\mathcal{S})$ is the decomposition level of set $\mathcal{S}$ in the k-d tree, and LIS_l is the LIS for decomposition level l. As with OB-SPECK, the BISK algorithm determines the significance of a set by comparing the largest opaque-coefficient magnitude contained in the set to the current threshold. Sets without a significant coefficient are placed in an LIS, and, during the sorting pass, each set in an LIS is tested for significance against the current threshold. If the set becomes significant, it is split in two according to the k-d tree decomposition structure described above. The two new sets are placed into an LIS, recursively tested for significance, and split again if needed. At any time, if a set contains no opaque coefficients, it is removed from its LIS and discarded. The splitting and coding procedures for sets in the LIS lists are as follows:

```
procedure Process(S)
  if S = ∅
    remove S from LIS_{l(S)}
  else
    output Γ_n(S)
    if Γ_n(S) = 1
      remove S from LIS_{l(S)}
      if |S| = 1
        output sign of S
        append S to LSP
      else
        Code(S)
procedure Code(S)
  {S_1, S_2} = Partition(S)
  if S_1 ≠ ∅
    append S_1 to LIS_{l(S_1)}
    Process(S_1)
  append S_2 to LIS_{l(S_2)}
  Process(S_2)
procedure Partition(S)
  if l(S) is odd
```

split $\mathcal{S}$ into $\mathcal{S}_1$ and $\mathcal{S}_2$:
$\mathcal{S}_1$: size $y(\mathcal{S}) \times \lfloor x(\mathcal{S})/2 \rfloor$
$\mathcal{S}_2$: size $y(\mathcal{S}) \times \big(x(\mathcal{S}) - \lfloor x(\mathcal{S})/2 \rfloor\big)$
else
split $\mathcal{S}$ into $\mathcal{S}_1$ and $\mathcal{S}_2$:
$\mathcal{S}_1$: size $\lfloor y(\mathcal{S})/2 \rfloor \times x(\mathcal{S})$
$\mathcal{S}_2$: size $\big(y(\mathcal{S}) - \lfloor y(\mathcal{S})/2 \rfloor\big) \times x(\mathcal{S})$
$l(\mathcal{S}_1) \leftarrow l(\mathcal{S}) + 1$
$l(\mathcal{S}_2) \leftarrow l(\mathcal{S}) + 1$
`Shrink`($\mathcal{S}_1$)
`Shrink`($\mathcal{S}_2$)

Above, $\Gamma_n(\mathcal{S})$ is the significance state of set $\mathcal{S}$, while $y(\mathcal{S})$ and $x(\mathcal{S})$ are the number of rows and columns, respectively, of set $\mathcal{S}$.

In addition to the k-d tree decomposition structure employed, there are several other key differences between BISK and OB-SPECK. First, BISK does not employ an $\mathcal{I}$ set as does OB-SPECK. Additionally, as can be seen above, before a set is added to an LIS, the set is "shrunk" to the bounding box of the opaque coefficients in the set. Specifically, procedure `Shrink`$(\cdot)$ consults the region of the transparency mask that corresponds to set $\mathcal{S}$ and reduces the size of $\mathcal{S}$ to the bounding box of the opaque coefficients in that set. Fig. 1 illustrates how BISK decomposes an arbitrarily shaped image object using k-d trees coupled with set shrinking.

A final difference between OB-SPECK and BISK lies in the adaptive arithmetic coding used for set significance. Because BISK uses binary, rather than quaternary, partitioning of sets, the arithmetic-coding process is somewhat simpler for BISK. Specifically, when set $\mathcal{S}$ is split into sets $\mathcal{S}_1$ and $\mathcal{S}_2$, and it happens that $\mathcal{S}_1$ is empty or insignificant, then it is known that $\mathcal{S}_2$ is significant, and thus significance state $\Gamma_n(\mathcal{S}_2)$ is not coded to the bitstream.[1] Otherwise, the coding of $\Gamma_n(\mathcal{S}_2)$ is conditioned on the fact that $\mathcal{S}_1$ was significant. The contexts used for set-significance coding within the BISK algorithm are as follows:

$c(\mathcal{S}_1) \leftarrow$ `CONTEXT_S1`
if $\mathcal{S}_1 = \emptyset$ or $\Gamma_n(\mathcal{S}_1) = 0$
$c(\mathcal{S}_2) \leftarrow$ `CONTEXT_NOCODE`
else
$c(\mathcal{S}_2) \leftarrow$ `CONTEXT_S2`

Above, $c(\mathcal{S}_i)$ denotes the context that will be used to code $\Gamma_n(\mathcal{S}_i)$.

4. EXPERIMENTAL RESULTS

We evaluate the performance of our proposed BISK algorithm on the still-image objects used in [12] and measure the "complexity" of the object shapes with a boundary-to-area ratio (BAR) defined as the total number of opaque pixels that lie on the boundary of the object expressed as the percentage of the total number of opaque pixels in the object. We measure fidelity of the reconstructed image objects as a PSNR distortion calculated over only opaque pixels in the image objects. All coders employ a 4-scale transform using the popular 9/7 biorthogonal filter, and we use the QccPack [13] implementations of BISK, OB-SPECK, SPIHT, WDR, and tarp. These implementations support shape-adaptive coding as described above for BISK, in [5] for OB-SPECK, in [9] for SPIHT, in [1] for WDR, and in [12] for tarp. For EBCOT, we use the shape-adaptive implementation described in [2] and code with a single quality layer.

[1] Due to how sets are partitioned (see `Partition`$(\cdot)$) $\mathcal{S}_2$ is guaranteed to be nonempty while $\mathcal{S}_1$ may or may not be empty.

First, we compare BISK to the original OB-SPECK [5] algorithm. Rate-distortion performance curves are depicted in Fig. 2 for a complex object (large BAR value) and a simple object (small BAR value), respectively. From these results, we see that the proposed BISK algorithm consistently outperforms the original OB-SPECK. While only slight performance gains are seen for objects with simple shapes, BISK outperforms OB-SPECK by a significantly wider margin (up to 0.5 dB) when the object mask is relatively complex.

Distortion values for a fixed rate for all the shape-adaptive coders under consideration are tabulated in Table 1. Although the performances of the various algorithms are, by and large, fairly similar over the set of image objects tested, we see that BISK consistently yields the highest PSNR, or close to it, for each image object. Averaged over the entire image-object set, BISK achieves the highest average PSNR.

5. CONCLUSIONS

In this paper, we have introduced BISK, a variant of OB-SPECK that improves upon its shape-adaptive coding performance. This performance improvement is due primarily to two strategies employed by the BISK coder during the coding of significance-map information. First, BISK is built upon a k-d tree decomposition structure that employs binary, rather than quaternary, partitioning. Secondly, aggressive discarding of transparent regions from sets before they are partitioned further is accomplished by shrinking sets to the bounding box of the opaque coefficients they contain. In experimental results, a performance improvement over OB-SPECK is observed, particularly for image objects with relatively complex shapes. Additionally, when compared to a variety of other shape-adaptive coders with different significance-map coding methodologies, our BISK coder consistently yields the top performance, or close to it, over a set of test image objects.

An interesting observation lies in the fact that k-d trees typically need a significantly greater number of decompositions to represent an image object than does the quadtree structure. Apparently, the binary partitioning employed by BISK permits a simpler, and more effective, entropy coding of the set-significance information which counteracts the increase in the number of set-significance bits. This hypothesis is confirmed by the fact that we have observed that BISK and SPECK achieve virtually identical performance when coding full-frame (i.e., fully opaque) images, the set-shrinking process having no effect in this situation.

6. REFERENCES

[1] J. E. Fowler and D. N. Fox, "Embedded wavelet-based coding of three-dimensional oceanographic images with land masses," *IEEE Trans. Geoscience Remote Sensing*, vol. 39, no. 2, pp. 284–290, Feb. 2001.

[2] Z. Liu, J. Hua, Z. Xiong, and K. Castleman, "Lossy-to-lossless ROI coding of chromosome images using modified SPIHT and EBCOT," in *Proc. IEEE Int. Symp. Biomedical Imaging*, Washington, DC, Jul. 2002, pp. 317–320.

[3] S. Li and W. Li, "Shape-adaptive discrete wavelet transforms for arbitrary shaped visual object coding," *IEEE Trans. Circ. Sys. Video Tech.*, vol. 10, no. 5, pp. 725–743, Aug. 2000.

[4] A. Islam and W. A. Pearlman, "An embedded and efficient low-complexity hierarchical image coder," in *Proc. VCIP*, San Jose, CA, Jan. 1999, Proc. SPIE 3653, pp. 294–305.

[5] Z. Lu and W. A. Pearlman, "Wavelet video coding of video object by object-based SPECK algorithm," in *Proc. PCS*, Seoul, Korea, Apr. 2001, pp. 413–416.

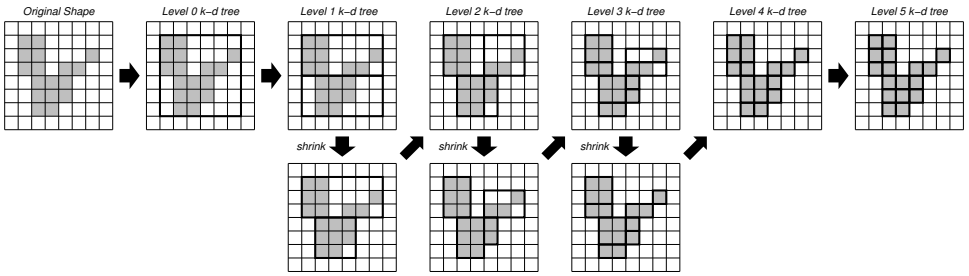
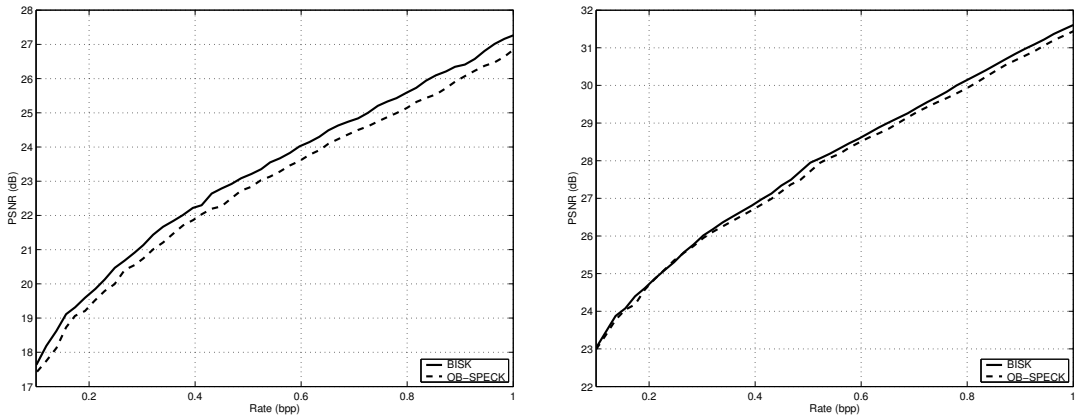


Figure 1: BISK decomposition of an arbitrarily shaped image object, with shrinking of subblocks to opaque bounding box. Gray indicates opaque coefficients.



(a) (b)

Figure 2: Rate-distortion performance of BISK and OB-SPECK. (a) hall-monitor (large BAR), (b) erik (small BAR).

Table 1: Shape complexities for the datasets and distortion performance for the shape-adaptive coders at 0.5 bpp.

		PSNR (dB)					
Image Object	BAR	BISK	OB-SPECK	SPIHT	EBCOT†	WDR	tarp
coastguard	14.6%	20.3	19.9	19.3	**20.8**	20.6	19.9
road-surveillance1	14.5%	**24.5**	23.9	22.8	23.7	24.3	23.7
hall-monitor	14.2%	**23.2**	22.8	21.9	22.7	**23.2**	22.9
penguin	5.7%	**31.5**	31.3	30.5	31.4	31.0	31.1
paris	3.3%	**28.3**	28.2	27.9	27.9	27.8	**28.3**
erik	2.0%	27.9	27.7	27.5	27.9	27.7	**28.0**
Average		**26.0**	25.6	25.0	25.7	25.8	25.7

†Rate for EBCOT is highest rate achieved not exceeding 0.5 bpp.

[6] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Comm. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[7] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circ. Sys. Video Tech.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.

[8] A. Kawanaka and V. R. Algazi, "Zerotree coding of wavelet coefficients for image data on arbitrarily shaped support," in *Proc. DCC*, Snowbird, UT, Mar. 1999, p. 534.

[9] G. Minami, Z. Xiong, A. Wang, and S. Mehrota, "3-D wavelet coding of video with arbitrary regions of support," *IEEE Trans. Circ. Sys. Video Tech.*, vol. 11, no. 9, pp. 1063–1068, Sep. 2001.

[10] J. Tian and R. Wells, Jr., "Embedded image coding using wavelet difference reduction," in *Wavelet Image and Video Compression*, P. N. Topiwala, Ed., pp. 289–301, Kluwer, Boston, MA, 1998.

[11] P. Simard, D. Steinkraus, and H. Malvar, "On-line adaptation in image coding with a 2-D tarp filter," in *Proc. DCC*, Snowbird, UT, Apr. 2002, pp. 23–32.

[12] J. E. Fowler, "Shape-adaptive tarp coding," in *Proc. ICIP*, Barcelona, Spain, September 2003, vol. 1, pp. 621–624.

[13] J. E. Fowler, "QccPack: An open-source software library for quantization, compression, and coding," in *Appl. Digital Image Proc. XXIII*, San Diego, CA, Aug. 2000, Proc. SPIE 4115, pp. 294–301.