SPIHT-Based Coding of the Shape and Texture of Arbitrarily Shaped Visual Objects

Karl Martin, *Student Member, IEEE*, Rastislav Lukac, *Member, IEEE*, and Konstantinos N. Plataniotis, *Senior Member, IEEE*

***Abstract*—A new scheme for coding both the shape and texture of arbitrarily shaped visual objects is presented. Based on set partitioning on hierarchical trees (SPIHT), the proposed Shape and Texture SPIHT (ST-SPIHT) employs a novel implementation of the shape-adaptive discrete wavelet transform (SA-DWT) using in-place lifting, along with parallel coding of texture coefficients and shape mask pixels to create a single embedded code that allows for fine-grained rate-distortion scalability. The single output code simplifies the logistics of object storage and transmission compared to previously published schemes. An input parameter provides control over the relative progression between shape and texture coding in the embedded code, allowing for adjustment of the emphasis of shape versus texture quality in low bit rate reconstructions. The combination of features provided by ST-SPIHT, namely, explicit and progressive shape coding in parallel with wavelet-based embedded coding of the object texture, is unique compared to previously published schemes. Computational complexity is minimized since the shape coding takes advantage of the decomposition and spatial orientation trees used for texture coding. Objective and subjective simulation results show that the proposed ST-SPIHT scheme has rate-distortion performance comparable or superior to MPEG-4 Visual Texture Coding for most bit rates.**

***Index Terms*—Set partitioning in hierarchical trees (SPIHT), shape-adaptive coding, shape-adaptive discrete wavelet transform (SA-DWT), shape and texture coding, visual object coding, wavelet-based coding.**

I. Introduction

THE coding of arbitrarily shaped visual objects has been increasing in importance in recent times. Many emerging applications, such as content based storage and retrieval systems [1], rely on, or are enhanced by, object-based coding. Also, object-based coding enables very low bit rate imagery by coding only specific content that is important for the application at hand. The significance of this technology is emphasized by its inclusion in the MPEG-4 standard [2]. The object-based coding paradigm differs from traditional coding techniques, such as the discrete cosine transform (DCT) based JPEG standard and the discrete wavelet transform (DWT) based JPEG2000 standard, which are limited to coding rectangular imagery [3].

Manuscript received May 19, 2005; revised March 27, 2006. This work was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under the Network for Effective Collaboration Technologies through Advanced Research (NECTAR) project. This paper was recommended by Associate Editor H. Sun.

The authors are with the Multimedia Laboratory, The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: kmartin@dsp.utoronto.ca).

Color versions of Figs. 7 and 10–18 are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2006.882388

Several shape-adaptive DCT and DWT based transforms have been proposed in the literature, along with shape-adaptive variants of popular coding schemes such as embedded zerotree wavelet coder (EZW) and set partitioning in hierarchical trees coder (SPIHT) [4]–[12]. The shape-adaptive DCT (SA-DCT) [4] and the ΔDC-SA-DCT variant [13] operate by flushing the valid pixels within an 8×8 bounding box to the top edge of the box and performing the 1-D N-DCT on each column, then flushing to the left and performing the 1-D N-DCT on each row, where $N \leq 8$ corresponds to the valid number of pixels in the given row or column. This operation has the desirable property of having the resulting number of coefficients equal to the number of valid original pixels within the 8×8 bounding box. However, the flushing operations can merge potentially unrelated and out of phase image segments, resulting in artificial frequency components.

More recently, the shape-adaptive discrete wavelet transform (SA-DWT) [7] has been proposed as a more efficient means of decomposing arbitrarily shaped objects. The SA-DWT avoids the blocking artifacts inherent to DCT based techniques by allowing the use of any FIR wavelet filter set for the decomposition of individual 1-D image segments within the object [14]. The operation involves scanning each row of the object for contiguous segments of pixels and performing an arbitrary-length wavelet decomposition on each one. The operation is then repeated on the columns. One of two subsampling strategies may be employed in the SA-DWT: 1) a *global* subsampling strategy which honors the position-parity of the segment with respect to the bounding box of the object or 2) a *local* subsampling strategy which treats each segment individually, regardless of its position within the bounding box. The global strategy respects the relative phase of the image segments whereas the local strategy ensures that there are never more high-pass samples than low-pass samples resulting from each individual segment decomposition. However, for the special case of segments of length $N = 1$, the pixel is always filtered and treated as a low-pass segment, regardless of its position and the chosen subsampling strategy.

The SA-DWT and the accompanying zerotree coding algorithm [7] have been adopted as the methodology for still visual texture coding (VTC) in the MPEG-4 standard [2]. Similar schemes have been proposed for coding the texture of video objects [10], [11] and multispectral images [12]. These schemes generally perform texture coding using adaptations of EZW and SPIHT, whereby coefficients residing outside the arbitrarily shaped objects are effectively discarded. As with the SA-DCT based schemes (e.g., [15]), only the coding of texture is addressed. This means that the object shape mask, represented as a binary image, must be: 1) coded separately using a binary

shape mask coding scheme and 2) fully decoded *prior* to the decoding of the texture. These restrictions can have a negative impact on the computational overhead for coding and decoding, complicate storage and retrieval logistics, and possibly reduce the quality of very low bit rate reconstructions. Scalable shape coding was described in [16] and adopted in MPEG-4 version 2 [2], allowing for progressive reconstruction of binary shape masks. While this allows for lossy shape reconstruction, shape and texture coding must still be implemented separately, with some shape information decoded before texture decoding in order to limit the region of the reconstructed texture upon performing the reverse wavelet transform.

The scheme proposed in [8] does provide "implicit" shape coding along with texture coding, but the object is separated from the background after a traditional—rather than shape-adaptive—transform is applied. A subsampled version of the shape mask is applied to each subband to determine which coefficients belong to the object, resulting in a greater number coefficients than original pixels. Furthermore, the original shape mask cannot subsequently be decoded accurately.

This paper presents a novel and efficient method of coding both the binary shape mask and texture of an arbitrarily shaped visual object into *one embedded code*. Called Shape and Texture SPIHT (ST-SPIHT), the proposed scheme uses a modification of the highly efficient, wavelet-based coding scheme SPIHT [17]. Compared to previously published schemes, the logistics of object storage and transmission are simplified by having only one embedded code containing both the shape and texture. The proposed procedure results in a multiresolution, progressive reconstruction of the binary shape mask as well as the texture. Subsequently, decoding does not require pre-transmission of the object shape mask. In contrast to [8], the shape mask can be accurately decoded and computational complexity is reduced by avoiding pre-processing transformations of the shape mask into the transform domain.

ST-SPIHT uses a parameter, *shape code level* (λ), to control how spread out the shape code is in the output bit stream. Equivalently, λ also determines how early in the output bit stream the complete lossless shape becomes available. This is of particular benefit to very low bit rate applications, such as high-speed object retrieval or recognition, where the emphasis on shape versus texture quality must be controlled. By choosing λ appropriately, more emphasis can be placed on low bit rate texture quality, with the trade-off of lossy shape reconstruction.

The rate-distortion performance of ST-SPIHT is comparable to MPEG-4 VTC since the texture coding is adapted from SPIHT, whereby only coefficients inside the object are considered. Additionally, the embedded coding procedure allows rate-distortion to be controlled and scaled with fine granularity during both coding and decoding. Computational complexity is minimized since the shape coding uses the same decomposition and spatial orientation tree structures as is used in the texture coding. While the lossless shape coding bit rate efficiency is below that of the best dedicated binary shape mask coders, at most bit rates the shape code represents a negligible portion of the total bit rate. In very low bit rate scenarios where the shape code becomes more significant, ST-SPIHT offers the option of increasing the spread of the shape code in the bit stream to the benefit of increased texture fidelity. This represents a simple solution for a variety of object-based applications. Furthermore, ST-SPIHT is fully backward compatible with SPIHT, generating exactly the same code when the input object fills the entire bounding box.

The proposed ST-SPIHT scheme employs a novel implementation [18] of the SA-DWT based on the in-place lifting DWT [19]. This implementation is highly efficient not only in the elementary DWT operation but because it allows the use of the spatial domain shape mask directly in the transform domain without any manipulation. This results in a one-to-one equivalence between the representation of the shape mask in the spatial domain and the transform domain, allowing transform domain coding with lossless shape reconstruction. If integer-to-integer wavelet filters are used in the SA-DWT [20], lossless reconstruction of the object texture may be achieved. With the use of advanced floating point filters, such as the CDF 9/7 biorthogonal filters [21], high-quality lossy results can be achieved. Both the objective and subjective results show that the proposed method produces results comparable to other schemes in the literature, but with the added benefit of a single implemented algorithm for both the shape and texture.

The remainder of the paper is organized as follows. The proposed ST-SPIHT object-based coding scheme is described and analyzed in Section II. Results and comparisons with other schemes are provided in Section III. The paper concludes with Section IV.

II. Shape and Texture SPIHT Coding Scheme

The ST-SPIHT coding and decoding system is shown in Fig. 1. The scheme follows a typical transform-based image coding structure with quantization occurring implicitly during ST-SPIHT coding. The input consists of two components: 1) an $M \times N$ full color (texture) image $\mathbf{x} : Z^2 \rightarrow Z^3$ representing a 2-D matrix of three-component RGB color samples $\mathbf{x}(i,j) = [x(i,j)_1, x(i,j)_2, x(i,j)_3]$, with $i = 0, 1, \ldots, M-1$ and $j = 0, 1, \ldots, N-1$ denoting the spatial position of the pixel, and $x(i,j)_k$ denoting the component in the red $(k = 1)$, green $(k = 2)$, or blue $(k = 3)$ color channel; and 2) an $M \times N$ binary (shape mask) image $s : Z^2 \rightarrow \{0,1\}$ representing a 2-D matrix of binary values where $s(i,j) = 1$ denotes spatial positions "inside" the object, and $s(i,j) = 0$ denotes spatial positions "outside" the object.

The object is preprocessed by first converting the texture to the $\mathrm{YC_bC_r}$ color space. Subsequently, texture positions outside the object are set to zero, such that $\mathbf{x}(i,j) = [0,0,0], \forall (i,j)$ where $s(i,j) = 0$.

Each color channel of the texture is then transformed using an in-place lifting SA-DWT with global subsampling, creating the $M \times N$ vectorial field $\mathbf{x}_\mathrm{T} : Z^2 \rightarrow Z^3$ of transform coefficients $\mathbf{x}_\mathrm{T}(i,j) = [x_\mathrm{T}(i,j)_1, x_\mathrm{T}(i,j)_2, x_\mathrm{T}(i,j)_3]$. This is a modification of the SA-DWT described in [7] with a novel implementation. The result allows the spatial domain shape mask s to remain unmanipulated and coded directly. The full details can be found in Appendix A.

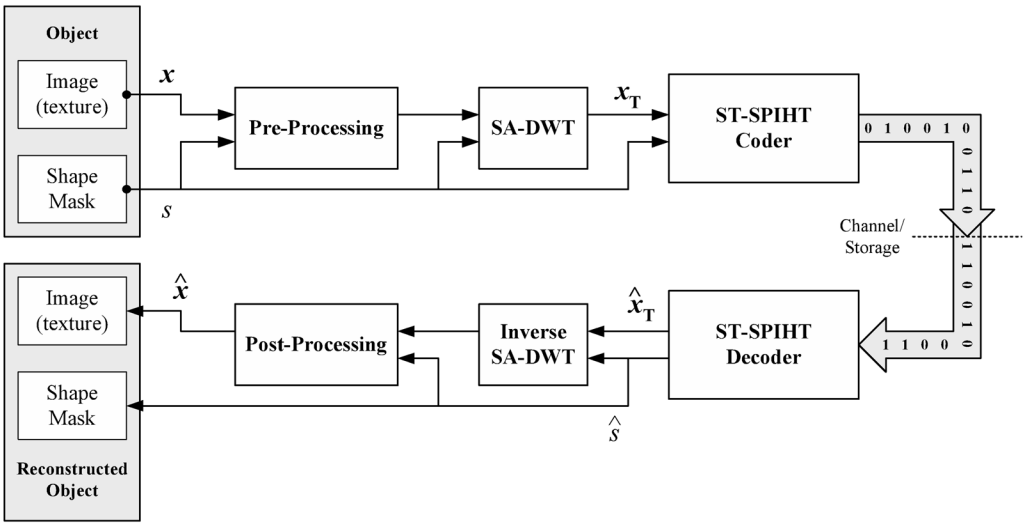


Fig. 1. System level diagram of the proposed ST-SPIHT coding and decoding scheme.

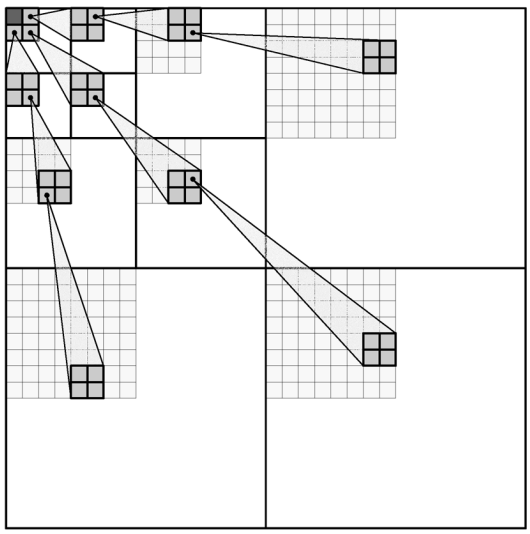


Fig. 2. SPIHT spatial orientation tree structure in a standard wavelet transform subband arrangement [17].

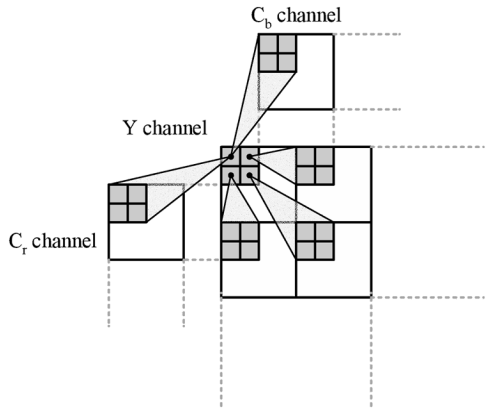


Fig. 3. Modified root parent-child structure from [22].

A. ST-SPIHT Coder and Decoder

The texture coding follows a natural extension of SPIHT, similar to the schemes proposed in [7], [9]–[11]. Namely, spatial orientation trees (SOT) and subtrees that reside outside the object are effectively discarded to avoid wasting bits coding them. The SOTs, as defined in [17] and shown in Fig. 2, are first formed using all coordinates inside the bounding box of size $M \times N$; the binary shape mask s is used to describe which nodes are inside the object and which are outside. Coding is performed in the $\mathrm{YC_bC_r}$ color space, using the SOT modification for color images proposed in [22]. As shown in Fig. 3, the chrominance LL subband coefficients become descendants of the luminance LL subband coefficients. The rest of the tree structure in the full-resolution chrominance channels is the same as in the luminance channel.

We define $\mathcal{G} = \{(i,j) \,|\, s(i,j) = 1\}$ as the set of all coordinates inside the object, and $\overline{\mathcal{G}} = \{(i,j) \,|\, s(i,j) = 0\}$ as the complementary set containing all coordinates outside the object —i.e., $\mathcal{G} \bigcup \overline{\mathcal{G}} = \{(i,j) \,|\, i = 0, 1, \ldots, M-1, j = 0, 1, \ldots, N-1\}$ and $|\mathcal{G}| + |\overline{\mathcal{G}}| = MN$. All the definitions from the standard SPIHT algorithm described in [17] remain in use with the addition of the color component index k. Briefly, the list of insignificant pixels (LIP), list of significant pixels (LSP), and list insignificant sets (LIS), store different coefficient and tree root coordinates. A "type-A" entry in the LIS refers to $\mathcal{D}(i,j)_k$, all the descendants of $(i,j)_k$; a "type-B" entry refers to $\mathcal{L}(i,j)_k = \mathcal{D}(i,j)_k - \mathcal{O}(i,j)_k$, where $\mathcal{O}(i,j)_k$ are the direct offspring of location $(i,j)_k$. $\mathcal{H}$ denotes the set of all luminance LL subband coefficient coordinates and $S_n(\cdot)$ refers to the significance test at bit plane n, as defined in [17].

We introduce a series of three "α-test" functions. The "α pixel test" function, $\alpha_p(\cdot,\cdot)$, identifies whether a coordinate is inside or outside the shape and is defined follows:

$$\alpha_p(i,j) = \begin{cases} 1, & (i,j) \in \mathcal{G} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Input: $\mathbf{x}_{\mathrm{T}}$, s, λ

1. **Initialization:** Find initial quantization level $n = n_{\max} = \left\lfloor \log_2 \left(\max_{(i,j,k)} \{|x_{\mathrm{T}}(i,j)_k|\} \right) \right\rfloor$; set LSP = $\emptyset$; set LIP = $\mathcal{H}$; set LIS = $\{(i,j)_k$ "type-A" $\mid (i,j)_k \in \mathcal{H},\ \mathcal{D}(i,j)_k \neq \emptyset\}$.
2. **Sorting pass:**
   - 2.1. For each $(i,j)_k \in$ LIP:
     - 2.1.1. If $\alpha_p(i,j)$ not coded yet then output $\alpha_p(i,j)$;
     - 2.1.2. If $\alpha_p(i,j) = 1$ then:
       - Output $S_n(i,j)_k$;
       - If $S_n(i,j)_k = 1$ then move $(i,j)_k$ to the LSP and output the sign of $x_{\mathrm{T}}(i,j)_k$;
     - 2.1.3. If $\alpha_p(i,j) = 0$ then remove $(i,j)_k$ from the LIP;
   - 2.2. For each entry $(i,j)_k \in$ LIS:
     If "type-A" entry, $\mathcal{T} = \mathcal{D}(i,j)_k$; If "type-B" entry, $\mathcal{T} = \mathcal{L}(i,j)_k$
     - 2.2.1. If $n \geq \lambda$ and shape not completely coded, then:
       - If $\alpha_{SD}(\mathcal{T})$ not coded yet then output $\alpha_{SD}(\mathcal{T})$;
       - If $\alpha_{SD}(\mathcal{T}) = 0$ then remove $(i,j)_k$ from the LIS and move on to next entry in the LIS (go to Step 2.2);
       - If $\alpha_{SD}(\mathcal{T}) = 1$ then:
         - If $\alpha_{SR}(\mathcal{T})$ not coded yet then output $\alpha_{SR}(\mathcal{T})$;
         - If $\alpha_{SR}(\mathcal{T}) = 0$ and $n = \lambda$ then run SCS($\mathcal{T}$);
     - 2.2.2. If shape completely coded and $\alpha_{SD}(\mathcal{T}) = 0$ then remove $(i,j)_k$ from the LIS and move on to next entry in the LIS (go to Step 2.2);
     - 2.2.3. If "type-A" entry and $\alpha_{SD}(\mathcal{T}) = 1$:
       - Output $S_n(\mathcal{D}(i,j)_k)$;
       - If $S_n(\mathcal{D}(i,j)_k) = 1$ then:
         - For each $(p,q)_r \in \mathcal{O}(i,j)_k$:
           - Output $S_n(p,q)_r$;
           - If $S_n(p,q)_r = 1$ then add $(p,q)_r$ to the LSP and output sign of $x_{\mathrm{T}}(p,q)_r$;
           - If $S_n(p,q)_r = 0$ then add $(p,q)_r$ to the LIP;
         - If $\mathcal{L}(i,j)_k \neq \emptyset$ then move $(i,j)_k$ to the end of the LIS as "type-B" entry; else, remove $(i,j)_k$ from the LIS;
     - 2.2.4. If "type-B" entry and $\alpha_{SD}(\mathcal{T}) = 1$:
       - Output $S_n(\mathcal{L}(i,j)_k)$;
       - If $S_n(\mathcal{L}(i,j)_k) = 1$ then:
         - Add each $(p,q)_r \in \mathcal{O}(i,j)_k$ to the end of the LIS as "type-A" entry;
         - Remove $(i,j)_k$ from the LIS.
3. **Refinement pass:** For each $(i,j)_k \in$ LSP, except those found significant in the current sorting pass, output the n^{th} most significant bit of $|x_{\mathrm{T}}(i,j)_k|$;
4. **Quantization-step update:** Decrement n by 1 and go to Step 2.

Fig. 4. ST-SPIHT Coder.

Input: set $\mathcal{T}$ with root $(i,j)_k$

1. If $(i,j)_k$ is "type-A" entry:
   - 1.1. For each $(p,q)_r \in \mathcal{O}(i,j)_k$:
     - 1.1.1. If $\alpha_p(p,q)$ not coded yet then output $\alpha_p(p,q)$;
     - 1.1.2. If $\mathcal{D}(p,q)_r \neq \emptyset$ then:
       - If $\alpha_{SD}(\mathcal{D}(p,q)_r)$ not coded yet then output $\alpha_{SD}(\mathcal{D}(p,q)_r)$;
       - If $\alpha_{SD}(\mathcal{D}(p,q)_r) = 0$ terminate processing of $\mathcal{D}(p,q)_r$;
       - If $\alpha_{SD}(\mathcal{D}(p,q)_r) = 1$ then:
         - If $\alpha_{SR}(\mathcal{D}(p,q)_r)$ not coded yet then output $\alpha_{SR}(\mathcal{D}(p,q)_r)$;
         - If $\alpha_{SD}(\mathcal{D}(p,q)_r) = 0$ then go to Step 1 treating $\mathcal{D}(p,q)_r$ as new "type-A" input;
2. If $(i,j)_k$ is "type-B" entry:
   - 2.1. For each $(p,q)_r \in \mathcal{O}(i,j)_k$, go to Step 1 treating $\mathcal{D}(p,q)_r$ as new "type-A" input;

Fig. 5. SCS subroutine.

The "α set-discard test" function, $\alpha_{\mathrm{SD}}(\cdot)$, identifies sets of coefficients that are entirely outside the object

$$\alpha_{\mathrm{SD}}(\mathcal{T}) = \begin{cases} 0, & \mathcal{T} \subseteq \bar{\mathcal{G}} \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

where $\mathcal{T}$ represents a given set of coefficients. And finally, the "α set-retain test" function, $\alpha_{\mathrm{SR}}(\cdot)$, identifies sets of coefficients that are entirely inside the object

$$\alpha_{\mathrm{SR}}(\mathcal{T}) = \begin{cases} 1, & \mathcal{T} \subseteq \mathcal{G} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The ST-SPIHT coding routine requires the *shape code level* parameter, λ, to be input. This defines the quantization level at which the routine forces the coding of not-yet-coded shape mask pixels $s(i,j)$. This is done by applying the subroutine "shape code set" (SCS) to the appropriate trees. The complete description of the ST-SPIHT routine and the SCS subroutine are shown in Figs. 4 and 5, respectively.

During execution of the ST-SPIHT coder, two counters, C_1 and C_0, keep track of the number of positions coded $s(i,j) = 1$ and $s(i,j) = 0$, respectively. The counter C_1 has an expected final value of $|\mathcal{G}|$ (the number of pixels inside the object); C_0 has an expected final value of $|\overline{\mathcal{G}}|$ (the number of pixels outside the object). When either $C_1 = |\mathcal{G}|$ or $C_0 = |\overline{\mathcal{G}}|$, s is considered to be completely coded and the results of the α-test functions (1)–(3) are no longer output into the bit stream. However, the procedure can be terminated at any point, regardless of whether shape coding is complete. Typically, either a rate or distortion criterion is used to control termination of the algorithm [17], [23].

The decoder follows exactly the same execution path as the coder. To initialize, the decoder must first receive the parameters $M, N, |\mathcal{G}|$, the number of wavelet transform levels, the initial quantization level $n_{\max}$, and λ. The decoded shape mask and texture, $\hat{s}$ and $\hat{\mathbf{x}}_{\mathrm{T}}$ respectively, are initialized such that $\hat{s}(i,j) = \emptyset$ and $\hat{\mathbf{x}}_{\mathrm{T}}(i,j) = [0,0,0], \forall (i,j)$. As with the coder, the decoder maintains two counters, $\hat{C}_1$ and $\hat{C}_0$, that keep track of the number of pixels identified inside and outside the object, respectively. These counters also have expected final values $|\mathcal{G}|$ and $|\overline{\mathcal{G}}|$, respectively.

Each bit received as the result of one of the α-test functions results in the updating of the values in $\hat{s}$ and incrementing the appropriate counter. All other bits serve to gradually refine the values in $\hat{\mathbf{x}}_{\mathrm{T}}$. When either $\hat{C}_1 = |\mathcal{G}|$ or $\hat{C}_0 = |\overline{\mathcal{G}}|$, the shape decoding is complete and all remaining "uncoded" positions in $\hat{s}$ are set according to the counter which has not reached its expected final value. After this, we have lossless shape reconstruction ($\hat{s} = s$).

When decoding is terminated, if $\hat{s} \neq s$ (i.e., lossy shape reconstruction), the remaining uncoded values $\hat{s}(i,j) = \emptyset$ must be set to either 0 or 1. A variety of adaptive filtering and prediction techniques may be used to determine these values (e.g., [24]). However, this is beyond the scope of this paper. Experimental results were generated using a simple prediction scheme which involved setting values $\hat{s}(i,j) = \hat{s}(p,q)$, where $\hat{s}(i,j)$ has not been decoded, (p,q) is the parent location of (i,j) according to the SOT structure (Figs. 2 and 3), and $\hat{s}(p,q)$ has either been decoded or predicted from its parent.

As shown in Fig. 1, the final reconstruction of the object is achieved by performing the reverse SA-DWT on $\hat{\mathbf{x}}_{\mathrm{T}}$ using the completed $\hat{s}$. The texture is then converted back to the RGB domain to acquire the reconstructed texture $\hat{\mathbf{x}}$.

B. Analysis of ST-SPIHT

The ST-SPIHT coding routine is designed to code shape information in parallel with the texture coding. This occurs in two phases: *passive* shape coding, occurring during the iterations in which $n_{\max} \geq n > \lambda$, and *active* shape coding performed when $n = \lambda$, after which shape coding is complete. Hence, the parameter λ is used to control the relative balance between passive and active shape coding. By setting $\lambda = n_{\max}$, coding of the shape is completed very early in the bit stream as a result of performing active shape coding immediately during the first iteration. This may be desirable in situations where the shape information is considered to be of great importance; the decoder is only required to decode a very small portion of the bit stream to receive the complete, lossless object shape. Experimental simulations show that this also generally results in more efficient shape coding. Setting $\lambda < n_{\max}$, the shape information is coded more progressively throughout the bit stream. This may be desired in situations where texture coding is of higher priority at low bit rates. Note that setting $\lambda \ll n_{\max}$ (e.g., $\lambda < n_{\max} - 10$), shape coding may never be completed at typical bit rates. However, it should also be noted that λ has no effect on the efficiency of the texture coding since the execution path is unaffected by the progression of shape coding. A more detailed description of the passive and active shape coding follows.

1) Passive Shape Coding: The passive shape coding involves applying the α-test functions and transmitting the results only for individual coefficients and sets of coefficients that naturally arise during texture coefficient significance testing. Specifically, the α_{SD} and α_{SR} tests are only applied to entire sets of coefficients that are already in the LIS; subsets are only tested if the initial sets are partitioned as a result of the texture coding. Hence, if a set of coefficients $\mathcal{T}$ contains no significant texture coefficients (i.e., $S_n(\mathcal{T}) = 0$), but contains a mixture of coefficients both inside and outside the object ($\alpha_{\mathrm{SD}}(\mathcal{T}) = 1$ and $\alpha_{\mathrm{SR}}(\mathcal{T}) = 0$), the shape information for all of $\mathcal{T}$ will remain uncoded. Similarly, the α_p test is only applied to individual coefficients that are separated from trees during partitioning as a result of the texture coding.

2) Active Shape Coding: The active shape coding occurs only during the iteration $n = \lambda$ and is performed by the SCS subroutine. The SCS subroutine is applied to all sets whose shape information would have been left uncoded by the passive shape coding, i.e., any set $\mathcal{T}$ of coefficients with no significant texture elements ($S_n(\mathcal{T}) = 0$) and containing a mixture of coefficients inside and outside the object ($\alpha_{\mathrm{SD}}(\mathcal{T}) = 1$ and $\alpha_{\mathrm{SR}}(\mathcal{T}) = 0$). The SCS subroutine performs a recursive partitioning of $\mathcal{T}$, mimicking the main SPIHT algorithm, in order to isolate subsets that are completely inside or outside the object. However, it should be noted that the SCS does not permanently partition the sets. The original $\mathcal{T}$ entered into the SCS subroutine remains intact to resume normal texture coding of the coefficients inside the set. The shape information resulting from the SCS subroutine is later used to discard subsets and individual coefficients that are outside the object and encountered during texture coding.

It should be noted that both the passive and active shape coding are implemented with minimal computational overhead due to the use of the same decomposition and SOTs used in texture coding. Also, during both passive and active shape coding, it is possible to encounter coefficient coordinates that have already been coded as being inside or outside the object. This occurs when a coordinate or set of coordinates has been shape coded as a result of encountering texture coefficients from those locations in one color channel and then encountered again in the same locations but for another color channel. In these cases, the algorithm execution is not altered, simply, no bits are transmitted into the bit stream as a result of the α-tests.

During decoding, the bits received as a result of the α-tests serve to progressively reconstruct the shape information. This is in contrast to schemes like [7], [9]–[12] which assume that the shape mask is pre-transmitted and readily available to the texture decoder upon initialization. As shown in Appendix A, the shape mask can be coded directly in the transform domain due to the one-to-one equivalence between the spatial domain representation and the transform domain representation. If decoding is terminated such that lossy shape reconstruction results, the object texture will have added distortion due to the reverse SA-DWT including or excluding coefficient positions incorrectly. However, the impact is often minimal since an incorrectly included coefficient location (i,j) will have reconstructed texture coefficient value $\hat{\mathbf{x}}_{\mathrm{T}}(i,j) = [0,0,0]$. Similarly, an incorrectly excluded coefficient location (i,j) will have had its original texture coefficient $\mathbf{x}_{\mathrm{T}}(i,j)$ insignificant with respect to at least the previous quantization level $n_{\mathrm{end}} + 1$, where n_{end} is the quantization level when decoding is terminated. Note that a perfect transmission channel is assumed and hence distortion occurs solely from quantization and lossy shape reconstruction. If bit errors were to occur, it would not be possible to correctly decode any portion of the output bit stream following the error unless an error correcting code were employed.

An important property of the ST-SPIHT coder is that it is fully backward compatible with SPIHT. When the object fills the entire bounding box such that $s(i,j) = 1, \forall (i,j)$, the generated coded will be the same as in SPIHT and can be decoded

with a standard SPIHT decoder. In this case, upon initialization, $C_0 = 0 = |\overline{\mathcal{G}}|$, and the shape will be considered to have been immediately coded. Hence, at every step when an α-test is performed, the result of the test will not be output into the bit stream (i.e., the shape information for the set or individual coefficient will be assumed coded already) and the execution path will be exactly the same as in the standard SPIHT coder. This is a natural property of the algorithm and no special provisions need to be made.

C. Embedded Shape Coding Interpretation

The coding of shape information integrates seamlessly with the standard SPIHT codec because it takes advantage of the SOT structure to code a large number of pixels as being inside or outside the object (in the same way that SPIHT uses the SOT to code the insignificance of a large number of pixels). As described in Appendix A, the shape mask for each subband can be derived by performing the lazy wavelet transform on the spatial domain mask. This results in the one-to-one mapping of the coordinates in the subband mask to the spatial mask. Because of this, the shape coding procedure can be interpreted as a simplified SPIHT coder applied to the lazy wavelet transform of the shape mask.

It is useful to examine how the coordinates of the nodes in the SOTs map to the spatial domain. This is shown in Fig. 6 for a two level DWT of an 8×8 image. We first consider the simpler grayscale object case in which each 2×2 block of coefficients in the highest level LL subband creates three interleaved SOTs (the top-left coefficient is not the root of a tree). For the two level DWT shown in Fig. 6, the three middle arrays show the three interleaved SOTs and how their members' coordinates map to the spatial domain. Together, the three SOTs cover an area of size 8×8 pixels in the spatial domain. This can be generalized to say that, for an l level DWT, the three interleaved SOTs generated from a 2×2 block of coefficients in the level l LL subband cover an area of size $2^{l+1} \times 2^{l+1}$ pixels in the spatial domain.

The interpretation of this is that if the three interleaved SOTs, covering an area of $2^{l+1} \times 2^{l+1}$ block of pixels, is positioned such that all the pixels in that area are outside the object, it will require only 3 bits to encode all the shape information for that region. That is only $3/2^{2l+2}$ bits per pixel (bpp) to code the shape information for a compression ratio of $2^{2l+2}/3$ (assuming a raw bitrate of 1 bpp for the shape mask). For example, if $l = 5$, that is approximately 7.3×10^{-4} bpp or a compression ratio of approximately 1635 to 1. If the pixels were all inside the object instead of outside, the scheme, as it is designed, uses 2 bits per set for a total of 6 bits. While this is double the number of bits, it still results in a very low bpp. This bias toward coding "outside" coordinates can easily be reversed by reversing the order in which the α_{SD} and α_{SR} tests are applied.

The example described above is the idealized case. The shape coding is most efficient when inside and outside coordinates are grouped together in large blocks. Many of the SOTs will likely have mixed members with some being inside the object and some being outside. When this is the case, the SOT must be partitioned. This is shown in the right side of Fig. 6. The new smaller sets cover smaller areas in the spatial domain and so they

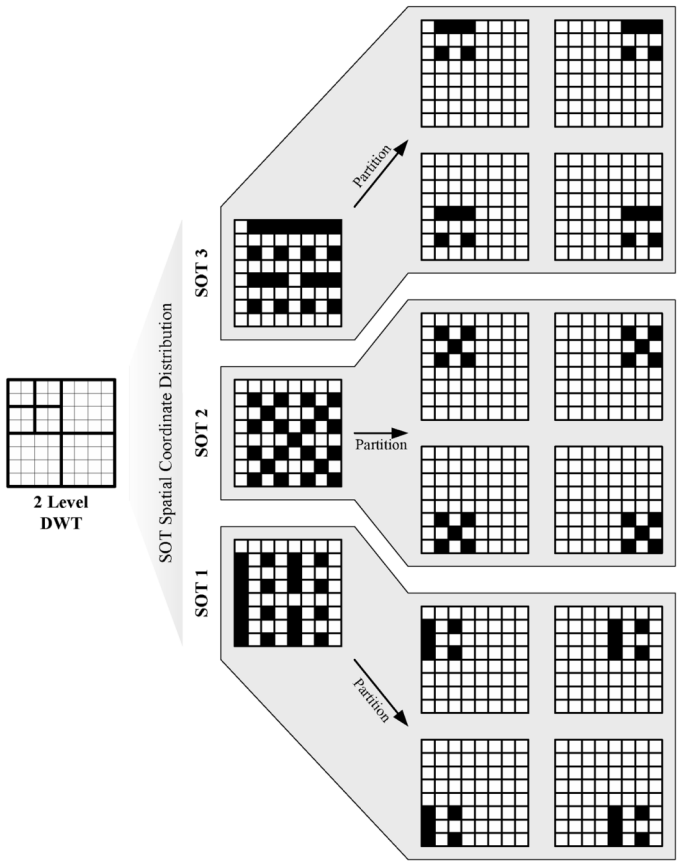


Fig. 6. Distribution of SOT members in spatial domain.

code finer shape detail. The partitioning continues until all the members are inside or outside the object. If the set happens to be centered on the edge of the object, the partitioning will likely continue until individual coordinates are isolated.

Using the SOTs to code the shape in fact results in a multiresolution binary mask coding scheme. The initial SOTs cover large image areas—if these areas are uniformly inside or outside the object, they are quickly coded. Finer details on the edge of the object require partitioning of the trees so that the new sets cover smaller areas. This continues until the finest details have been coded. Intuitively, the shape coding is most efficient with large scale features in the shape. Fine details require more bits to be spent.

In the case of color objects, the scenario is slightly different. Due to the modified SOT [22] shown in Fig. 3, the top-left coefficient of each 2×2 block of coefficients from the highest level luminance LL subband is now the root of an SOT. This one SOT, through the chrominance channels, actually covers the same area as the other three SOTs from the same 2×2 block. Because of this, the color case is actually much more efficient than the grayscale case for coding large scale shape features. As in the idealized example from before, if the entire $2^{l+1} \times 2^{l+1}$ pixel block which the SOT covers is outside the object, it will require only 1 bit to encode. If the entire block is inside the object, it will require 2 bits to encode. If the SOT is partitioned, then the new sets are the same as the luminance channel sets—the added efficiency is only realized if the entire set is found to be outside or inside before partitioning.

Fig. 7. Original test objects. (a) Akiyo. (b) Ceramic. (c) Fountain. (d) Foreman. (e) Parrots. (f) Pot.

III. Results and Comparison

Experimental results were generated by coding a variety of natural color imagery. The objects were chosen such that the object shape contained both large scale smooth features, and small scale fine details. The objects were segmented by hand, but in practice any segmentation algorithm appropriate for the application at hand may be employed. The MPEG-4 visual texture coding (VTC), SPIHT, and JPEG2000 schemes were implemented for comparison purposes. The MPEG-4 VTC coding utilized the Microsoft reference software v.2.5, based on part 2, version 2 of the standard (ISO/IEC 14496-2) [2]. Since the original SPIHT algorithm and JPEG2000 do not support true object coding, these schemes were applied to the objects with the background set to a uniform color. The color 50% gray was chosen to minimize the contrast at the object edges, in turn minimizing border distortions from the compression algorithms. Upon decoding, the object shape and texture can be extracted from these non-shape adaptive coded images using a chroma-key type operation. Additionally, the JPEG2000 scheme was implemented using the MAXSHIFT region-of-interest (ROI) coding methodology [25], [26], using the object shape as the ROI. For all schemes, the CDF 9/7 biorthogonal wavelet filters [21] with a 5–level transform were used. The 8-bit per channel RGB sample test objects are "akiyo,""ceramic,""fountain,""foreman,""parrots,"and "pot,"as shown in Fig. 7. The "ceramic" and "pot" objects were obtained from [27]. The objects contain a variety of textures and colors, as well as varied shape features. The size of the bounding box for the objects and the percentage of the bounding box occupied by the object are shown in Table I.

TABLE I
Bounding Box Sizes and Percentage of Bounding Box Occupied by Object for Test Objects

Object	Bounding Box Size	Object Percentage
'akiyo'	352×288	37.2%
'ceramic'	768×576	23.2%
'fountain'	576×448	64.2%
'foreman'	352×288	29.3%
'parrots'	768×512	42.6%
'pot'	768×576	36.1%

The peak signal-to-noise ratio (PSNR) was used as the quality metric:

$$\mathrm{PSNR} = 10 \log_{10} \frac{255^2}{\mathrm{MSE}} \tag{4}$$

with mean squared error (MSE) calculated in the RGB domain as

$$\mathrm{MSE} = \frac{1}{3|\mathcal{G}|} \sum_{(i,j)\in\mathcal{G}} \sum_{k=1}^{3} (x(i,j)_k - \hat{x}(i,j)_k)^2 \tag{5}$$

where $\mathbf{x}(i,j) = [x(i,j)_1, x(i,j)_2, x(i,j)_3]$ are the original RGB pixel components, $\hat{\mathbf{x}}(i,j) = [\hat{x}(i,j)_1, \hat{x}(i,j)_2, \hat{x}(i,j)_3]$ are the reconstructed RGB pixel components, and $|\mathcal{G}|$ is the total number of pixels inside the object.

It should be noted that in calculating PSNR for reconstructed objects, the MSE is simply calculated for the pixels inside the object. The original shape mask s is used to determine which pixels are inside the object, not the reconstructed shape mask $\hat{s}$.

The rate-distortion plots for the compared schemes applied to the test objects are shown in Fig. 8. For ST-SPIHT, $\lambda = n_{\max}$ was used so that lossless shape coding was completed after the first coding iteration. The bit rates were calculated based on the number of pixels inside the object. For mid to high bit rates, ST-SPIHT generally produced equal or superior results compared to the other schemes. At very-low bit rates, the results vary, with the object "fountain" producing nearly equal distortion for the different schemes, and the object "foreman" have several decibels greater distortion for ST-SPIHT, compared to the other schemes. This variance is the result of the overhead for coding lossless shape information; at low bit rates, the texture PSNR suffers in order to code the shape information (information not present in the SPIHT and JPEG2000 bit streams). However, in cases where higher quality texture is required at very low bit rates, λ should be chosen such that $\lambda < n_{\max}$. This scenario will be covered later in the section.

Fig. 9(a) and (b) show the effect that varying λ has on lossless shape coding. In Fig. 9(a), the bit rates for just the shape portion of the bit stream are shown. For $n_{\max} - 5 \leq \lambda \leq n_{\max}$, the bit rates show little variation. This represents a generally acceptable range for λ when lossless shape coding is required; for $\lambda \leq n_{\max} - 6$, the bit rates start to rise dramatically. In

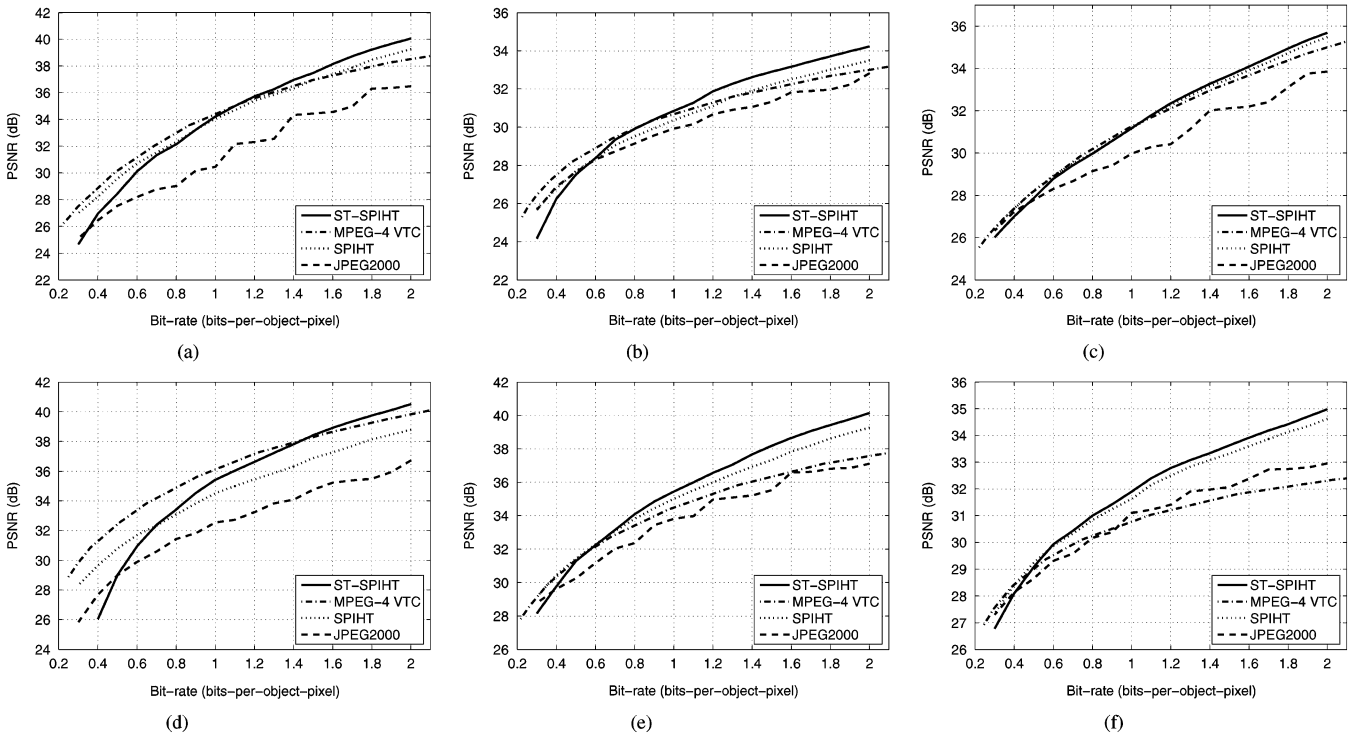


Fig. 8. PSNR versus bit rate for ST-SPIHT, MPEG-4 VTC, SPIHT, and JPEG2000. (a) Akiyo. (b) Ceramic. (c) Fountain. (d) Foreman. (e) Parrots. (f) Pot.

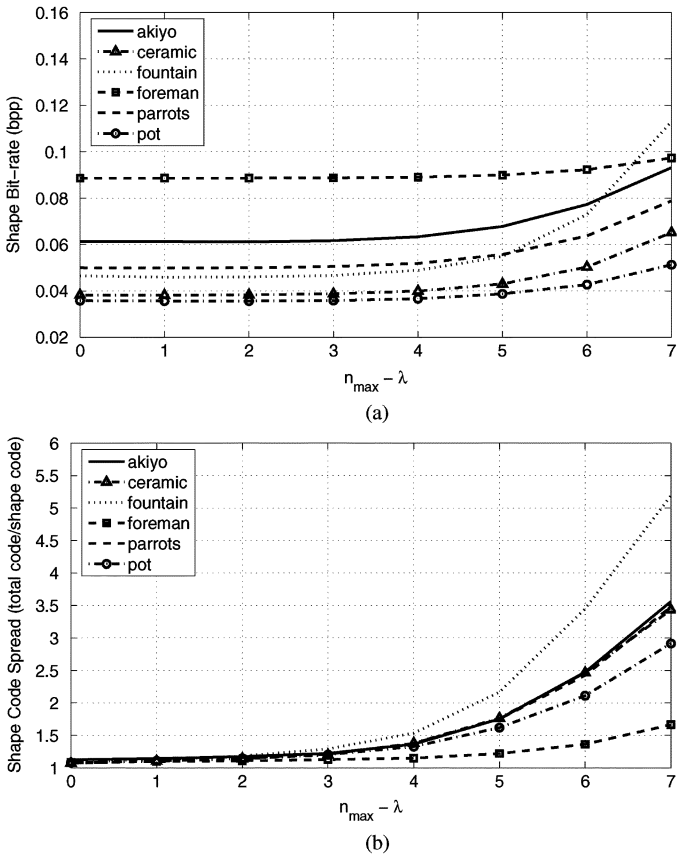


Fig. 9. Effect of λ on lossless shape code length and spread of code in full output stream. (a) Shape code bit rate versus $n_{\max} - \lambda$. (b) Shape code spread versus $n_{\max} - \lambda$.

Fig. 10. Sample results for "akiyo" at 0.5 bpp nd $\lambda = n_{\max}$. (a) ST-SPIHT. (b) MPEG-4 VTC.

Fig. 11. Sample results for "ceramic" at 0.5 bpp and $\lambda = n_{\max}$. (a) ST-SPIHT. (b) MPEG-4 VTC.

Fig. 9(b), the amount that the shape code is spread in the bit stream is shown. This is calculated as (# shape bits +# texture bits)/(# shape bits), with a higher number representing a shape code that is more spread out in the overall bit stream. The trend is similar to the shape bit rates, with the spread increasing as

(a) (b)

Fig. 12. Sample results for "fountain" at 0.5 bpp and $\lambda = n_{\max}$. (a) ST-SPIHT. (b) MPEG-4 VTC.

(a) (b)

Fig. 13. Sample results for "foreman" at 0.5 bpp and $\lambda = n_{\max}$. (a) ST-SPIHT. (b) MPEG-4 VTC.

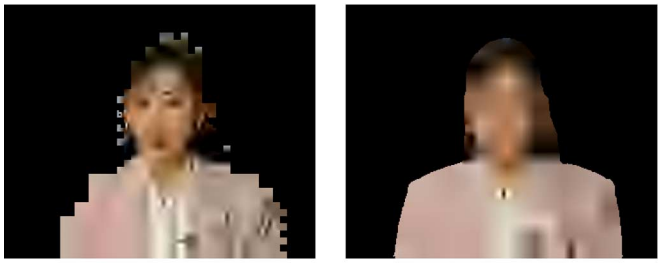
(a) (b)

Fig. 14. Sample results for "parrots" at 0.5 bpp and $\lambda = n_{\max}$. (a) ST-SPIHT. (b) MPEG-4 VTC.

(a) (b)

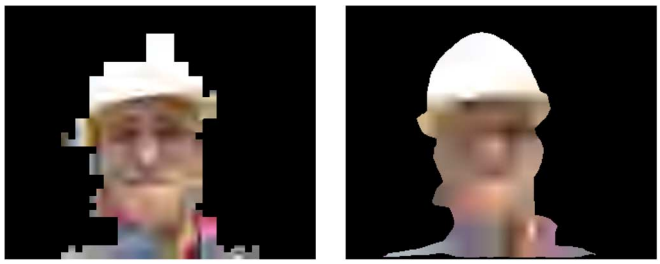
Fig. 15. Sample results for "pot" at 0.5 bpp and $\lambda = n_{\max}$. (a) ST-SPIHT. (b) MPEG-4 VTC.

λ decreases. Accordingly, if one desires a code that results in the shape being decoded slower, with more emphasis on texture early in the bit stream, a lower λ is chosen during coding. As previously noted, if $\lambda \leq n_{\max} - 6$, the compromise is that the shape bit rate increases. In these cases, inefficient passive shape coding is demonstrated as a result of the trees becoming too fragmented from the texture coding. However, λ has no effect on the efficiency of the texture coding since the execution path is unaffected by the progression of shape coding.



(a) (b)

Fig. 16. Very low bit rate results for "akiyo" at 0.0723 bpp and $\lambda = n_{\max} - 7$. (a) ST-SPIHT. (b) MPEG-4 VTC.



(a) (b)

Fig. 17. Very low bit rate results for "foreman" at 0.0949 bpp and $\lambda = n_{\max} - 7$. (a) ST-SPIHT. (b) MPEG-4 VTC.

(a) (b)

Fig. 18. Very low bit rate results for "parrots" at 0.0585 bpp and $\lambda = n_{\max} - 7$. (a) ST-SPIHT. (b) MPEG-4 VTC.

Figs. 10–15 show low bit rate subjective results using ST-SPIHT and MPEG-4 VTC. The bit rate 0.5 bpp was used for both schemes and, again, $\lambda = n_{\max}$ was used for ST-SPIHT. As demonstrated by the numerical results, at this low bit rate, "fountain,""parrots,"and "pot" produce approximately equivalent results, while "akiyo,""ceramic,"and "foreman" show marginally reduced quality for ST-SPIHT. As bit rates decrease below 0.5 bpp, if the application demands greater texture quality, λ should be set such that $\lambda < n_{\max}$. Very low bit rate reconstructions of "akiyo,""foreman,"and "parrots" are shown in Figs. 16–18, where $\lambda = n_{\max} - 7$. In all cases, ST-SPIHT produces more texture detail compared to MPEG-4 VTC, with the trade-off of lossy shape reconstruction. This can be of benefit to high-speed object recognition and retrieval applications.

Fig. 19 demonstrates the multiresolution, progressive shape coding for the example of the "parrots" objects, setting $\lambda = n_{\max} - 7$. The images on the left side of the figures show the actual decoded shape information with gray pixels representing

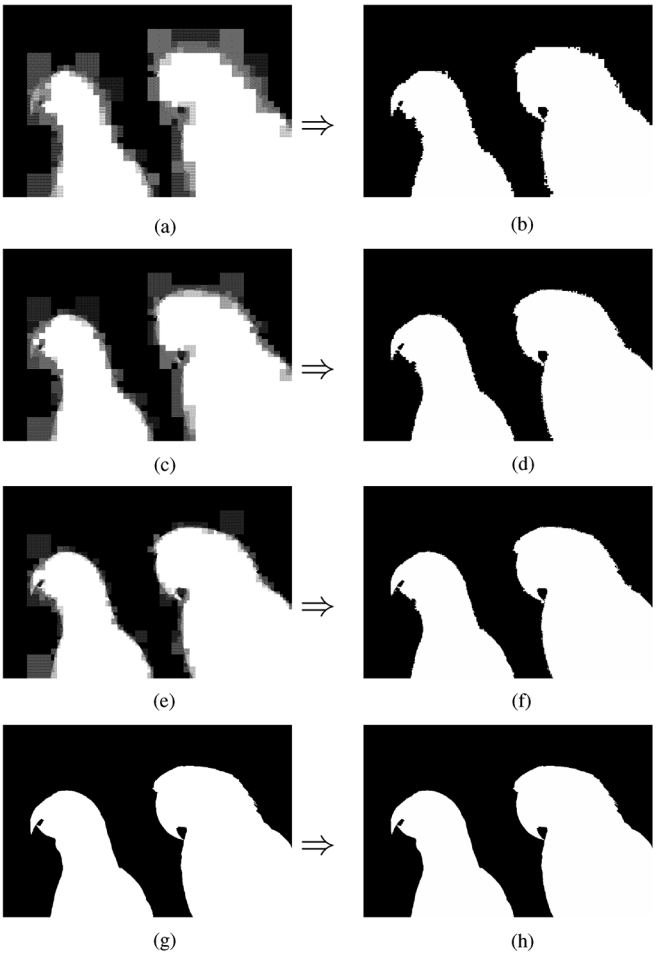


Fig. 19. Progressive shape reconstruction for "parrots" object using $\lambda = n_{\max} - 7$. Left column is actual decoded shape information with gray pixels representing unknown shape pixels. Right column is reconstructed shape based on decoded shape information and simple prediction scheme. (a) 27% shape bits/83% shape pixels. (b) 1.1% reconstruction error. (c) 50% shape bits/89% shape pixels. (d) 0.57% reconstruction error. (e) 73% shape bits/94% shape pixels. (f) 0.37% reconstruction error. (g) 100% shape bits/100% shape pixels. (h) 0% reconstruction error.

unknown values. The number of shape bits decoded is shown as a percentage of the total number of bits required to code the shape at the employed λ. Also indicated is the percentage of actual shape pixels decoded. As shown, coarse shape information is coded first with finer details filled in as more shape information is decoded.

The images on the right side of Fig. 19 show the actual reconstructed shape mask based on the decoded information and the simple prediction scheme described in Section II.A. The percentage reconstruction error describes the number of shape pixels that were incorrectly reconstructed as a percentage of the total number of shape pixels. The difference between the number of decoded shape pixels and the number of correctly reconstructed shape pixels is a result of the employed prediction scheme. This demonstrates usefulness of the multiresolution SOT representation of the shape mask for efficient lossy coding. Performance may be improved with more sophisticated shape prediction or estimation schemes.

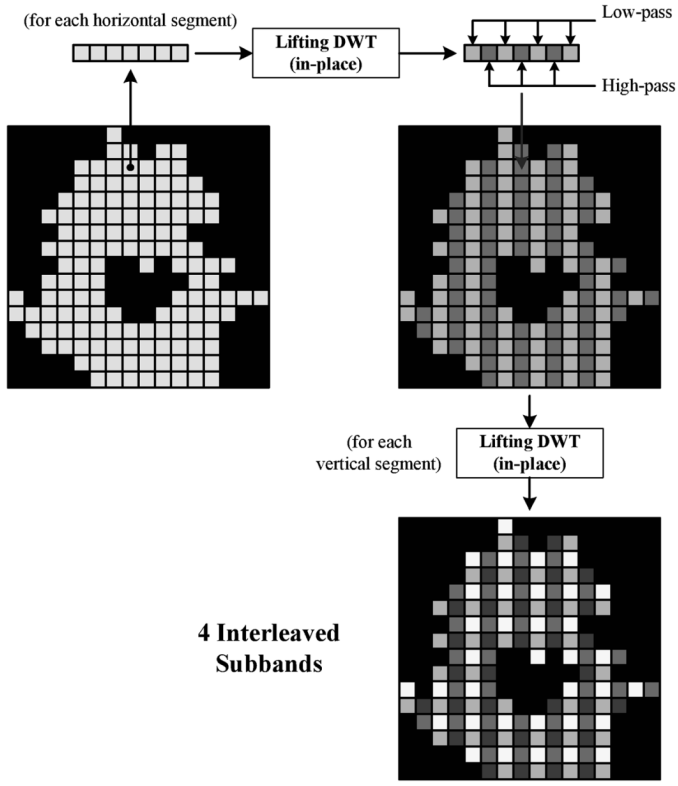


Fig. 20. One-level 2-D SA-DWT using in-place lifting DWT implementation.

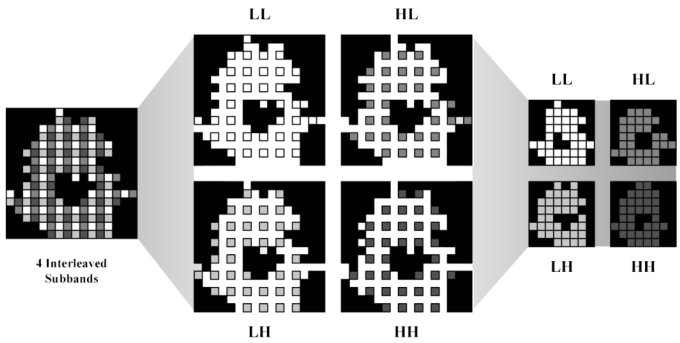


Fig. 21. Interleaved subband abstraction.

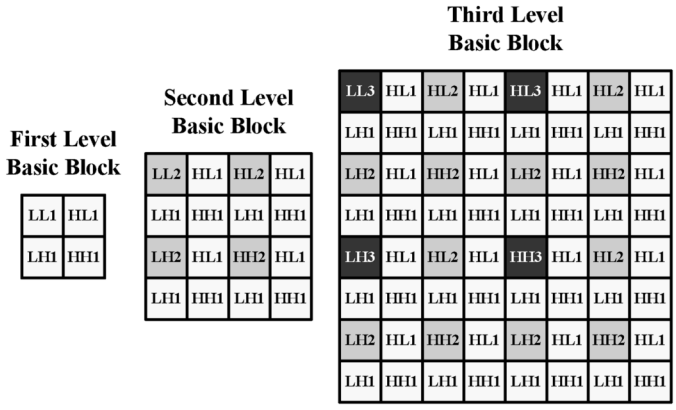


Fig. 22. Basic group of coefficients for each level of in-place DWT.

IV. Conclusion

The proposed ST-SPIHT shape-adaptive coding scheme effectively codes arbitrarily shaped visual objects and provides new functionality not present in previously published schemes. Namely, both shape and texture coding are integrated in one algorithm to create a single embedded code. This simplifies the logistics of storing visual objects, and negates the common requirement of pre-transmitting the object shape before texture decoding can commence.

The proposed scheme gives the user control over how early in the bit stream the shape becomes completely coded. By adjusting λ, a lossy shape may be reconstructed if more emphasis on texture coding is desired. In this case, care must given as to how the unknown shape mask elements are to be reconstructed. This is an open area of research not covered in this paper. This problem requires the implementation of a shape prediction or estimation scheme that may take into account the tree structure used to code the lossy shape, or may be a purely post-processing operation based on spatial domain predictive filtering. However, it was shown that the tree structure lends itself to effective prediction schemes since the simulations involved only a very simple shape mask prediction scheme which produced excellent results.

The rate-distortion performance of the shape coding in ST-SPIHT may be improved by replacing the α_{SD} and α_{SR} tests with a single test producing three possible outcomes: set entirely inside the object, set entirely outside the object, and mixed set. However, the three symbol alphabet would require arithmetic coding resulting in significant computational overhead.

The ST-SPIHT coding scheme may be extended to enable video object coding. The flexibility provided by ST-SPIHT in balancing the shape and texture coding leads to a large number of possible video coding schemes. For example, during encoding, different frames may be coded with a different λ parameter in order control the fidelity of the object shape at different points in the video sequence. Ultimately, the new features and flexibility provided by ST-SPIHT opens the door to a variety of emerging applications based on the object coding paradigm.

Appendix
In-Place Lifting SA-DWT

The SA-DWT employed in the ST-SPIHT scheme uses the *global* subsampling strategy as described by Li *et al.* in [7], with the exception of the special case of individual isolated pixels (i.e., segments of length $N = 1$). Li *et al.* propose that individual isolated pixels always be low-pass filtered; for ST-SPIHT, individual isolated pixels are made to still honor the global subsampling strategy by applying the low-pass filter for even positioned pixels, and applying the high-pass filter for odd positioned pixels.

The use of the global subsampling strategy along with the generalized treatment of individual pixels goes against the recommendations of Li *et al.* when applying the SA-DWT in zerotree coding schemes. However, this approach is used in ST-SPIHT so as to maintain an unambiguous representation of the shape mask in the transform domain. The result is a one-to-one relationship between the transform domain shape mask and the spatial domain shape mask. The implication is that the transform domain shape mask can be coded instead of the spatial domain shape mask. In contrast, if a local subsampling strategy were used, along with always treating individual pixels as low pass segments, a one-to-many relationship between the transform domain shape mask and the spatial domain shape mask would result, necessitating the coding of the original spatial domain shape mask to ensure proper decoding and placement of the object pixels. This can be demonstrated using the example of the isolated coefficient at position i of a low-pass subband, after performing a one-level decomposition. With the global subsampling strategy, it is known that the position of the original pixel that the coefficient corresponds to is $2i$; whereas, with the local subsampling strategy, it is unknown whether the position of the original pixel is $2i$ or $2i+1$. This ambiguity also occurs in the case of an isolated "hole" (i.e., coefficient position outside the object) in the high-pass subband when using the local subsampling strategy.

With the proposed global subsampling strategy, the highly memory and computationally efficient *in-place* lifting DWT implementation is employed [19]. For each one-dimensional segment that is transformed, a low-pass/high-pass interleaved 1-D segment is output and placed back into the position of the original image segment.

The in-place lifting DWT implementation has special implications for the SA-DWT, which can best be understood visually as shown in Fig. 20. As the SA-DWT is performed, the spatial domain shape mask remains intact with no requirement to derive a shape mask for each subband. How the subbands are arranged in this pseudo-spatial domain arrangement is shown in Fig. 21. Each subband can in fact be extracted from the interleaved subband arrangement using the *lazy wavelet transform* (LWT) [19]. After the one-level SA-DWT is performed, the LL1 subband can be extracted using a coordinate mapping from the interleaved subband coordinates (i, j) to the LL1 subband coordinates $(i_{\mathrm{LL1}}, j_{\mathrm{LL1}})$ using

$$(i_{\mathrm{LL1}}, j_{\mathrm{LL1}}) \leftarrow (\lfloor i/2 \rfloor, \lfloor j/2 \rfloor). \tag{6}$$

Similarly, the mapping for the HL1 subband is $(i_{\mathrm{HL1}}, j_{\mathrm{HL1}}) \leftarrow (\lfloor i/2 \rfloor + 1, \lfloor j/2 \rfloor)$; for the LH1 subband $(i_{\mathrm{LH1}}, j_{\mathrm{LH1}}) \leftarrow (\lfloor i/2 \rfloor, \lfloor j/2 \rfloor + 1)$; and for the HH1 subband $(i_{\mathrm{HH1}}, j_{\mathrm{HH1}}) \leftarrow (\lfloor i/2 \rfloor + 1, \lfloor j/2 \rfloor + 1)$. After the first level of the SA-DWT, the interleaved subband arrangement is made up of 2×2 basic blocks of coefficients. As shown in the left side of Fig. 22, the top-left coefficient of each block is an LL1 subband coefficient, the top-right coefficient is an HL1 subband coefficient, and so on.

The second level SA-DWT is performed by first extracting the LL1 subband using the coordinate mapping (6) and then performing the one-level SA-DWT using the LL1 subband as the new input. The output is the four interleaved subbands, LL2, HL2, LH2, and HH2. This is then placed back into the original

interleaved subband arrangement where the LL1 coefficients were extracted from. This creates a two-level interleaved subband arrangement. As shown in the middle of Fig. 22, the two-level interleaved subband arrangement is made of a basic 4×4 coefficient block, with the top-left coefficient of each block being an LL2 coefficient. The coordinate mappings to extract the second and subsequent level subbands are simply derived by applying the one level coordinate mappings iteratively to the LL subband coordinate mapping from the previous level.

As shown, the in-place lifting DWT implementation allows for the shape mask to remain defined in the spatial domain with the wavelet coefficients placed in reference to the original spatial coordinates. This arrangement allows for a one-to-one mapping of wavelet domain coordinates to spatial domain pixel coordinates thus making it easy to determine if a wavelet domain coordinate is inside or outside the object. This is especially useful during coding when the shape mask is coded at the same time as the wavelet coefficients. Also, it can be seen that if a standard rectangular image is input, the SA-DWT reverts to the standard DWT. The actual wavelet filters used in the individual decompositions may be chosen for the specific application at hand. Integer-to-integer filters should be used if lossless reconstruction is required.

REFERENCES

[1] M. Mandal, T. Aboulnasr, and S. Panchanathan, "Image indexing using moments and wavelets," *IEEE Trans. Consum. Electron.*, vol. 42, no. 3, pp. 557–565, Aug. 1996.

[2] *Information Technology - Coding of Audio-Visual Objects - Part 2: Visual (14496-2)*, ISO/IEC JTC1/SC29 WG11 International Standard, Rev. 3, Dec. 2004.

[3] K. Sayood, *Introduction to Data Compression*, 2nd ed. New York: Morgan Kaufmann, 2000.

[4] T. Sikora and B. Makai, "Shape-adaptive DCT for generic coding of video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 1, pp. 59–62, Feb. 1995.

[5] O. Egger, P. Fleury, and T. Ebrahimi, "Shape-adaptive wavelet transform for zerotree coding," in *Proc. Eur. Workshop Image Anal. Coding*, 1996, vol. 1, pp. 201–208.

[6] W. K. Ng and Z. Lin, "A new shape-adaptive DCT for coding of arbitrarily shaped image segments," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 2000, vol. 4, pp. 2115–2118.

[7] S. Li and W. Li, "Shape-adaptive discrete wavelet transforms for arbitrarily shaped visual object coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 5, pp. 725–743, Aug. 2000.

[8] A. A. Kassim and L. Zhao, "Rate-scalable object-based wavelet codec with implicit shape coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 7, pp. 1068–1079, Oct. 2000.

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

[10] G. Xing, J. Li, S. Li, and Y.-Q. Zhang, "Arbitrarily shaped video-object coding by wavelet," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 10, pp. 1135–1139, Oct. 2001.

[11] H. Danyali and A. Martens, "Fully scalable texture coding of arbitrarily shaped video objects," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 2003, vol. 3, pp. 393–396.

[12] M. Cagnazzo, G. Poggi, L. Verdoliva, and A. Zinicola, "Region-oriented compression of multispectral images by shape-adaptive wavelet transform and SPIHT," in *Proc. Int. Conf. Image Process.*, 2004, pp. 2459–2462.

[13] P. Kauff and K. Schuur, "Shape-adaptive DCT with block-based DC separation and delta DC correction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 3, pp. 237–242, Jun. 1998.

[14] M. Mandal, S. Panchanathan, and T. Aboulnasr, "Wavelets for image compression," in *Proc. IEEE Int. Symp. Time-Frequency Time-Scale Anal.*, Oct. 1994, pp. 338–341.

[15] P. Kauff, B. Makai, S. Rauthenberg, U. Golz, J. L. P. De Lameillieure, and T. Sikora, "Functional coding of video using a shape-adaptive DCT algorithm and an object-based motion prediction toolbox," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 181–196, Feb. 1997.

[16] S. Li and I. Sodagar, "Generic, scalable and efficient shape coding for visual texture objects in MPEG-4," in *Proc. Int. Symp. Circuits Syst.*, 2000, vol. 1, pp. 303–306.

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

[18] K. Martin, R. Lukac, and K. N. Plataniotis, "Binary shape mask representation for zerotree-based visual object coding," in *Proc. IEEE Can. Conf. Elect. Comp. Eng.*, May 2004, pp. 2197–2200.

[19] W. Sweldens, A. F. Laine and M. Unser, Eds., "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *Proc. Wavelet Applicat. Signal Image Process.*, 1995, pp. 68–79.

[20] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Appl. Comput. Harmon. Anal.*, vol. 5, no. 3, pp. 322–369, 1998.

[21] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Process.*, vol. 1, no. 4, pp. 205–220, Apr. 1992.

[22] A. A. Kassim and W. S. Lee, "Embedded color image coding using SPIHT with partially linked spatial orientation trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 2, pp. 203–206, Feb. 2003.

[23] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

[24] J. Koplowitz and A. M. Bruckstein, "Design of perimeter estimators for digitized planar shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, pp. 611–622, Jun. 1989.

[25] *Coding of Still Pictures: JPEG 2000 Part I Final Committee Draft Version 1.0*, ISO/IEC JTC1/SC29 WG1, Apr. 2000.

[26] D. Taubman and M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice.*. Norwell, MA: Kluwer, 2002.

[27] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The Amsterdam library of object images," *Int. J. Comput. Vis.*, vol. 61, no. 1, pp. 103–112, 2005.

Karl Martin (S'00) received the B.A.Sc. degree in engineering science (electrical specialty) and the M.A.Sc. degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2001 and 2003, respectively, where he is currently working toward the Ph.D. degree in the Edward S. Rogers Sr. Department of Electrical and Computer Engineering.

His research interests include multimedia security, multimedia processing, wavelet-based image coding, object-based coding, and CFA processing.

Mr. Martin is a member of the IEEE Signal Processing Society, Communications Society, and Circuits and Systems Society. He has been a Technical Reviewer for numerous journals and conferences. Since 2003, he has held the position of Vice-Chair of the Signal Processing Chapter, IEEE Toronto Section.

Rastislav Lukac (M'02) received the M.S. (Ing.) and Ph.D. degrees in telecommunications from the Technical University of Kosice, Slovak Republic, in 1998 and 2001, respectively.

From February 2001 to August 2002, he was an Assistant Professor with the Department of Electronics and Multimedia Communications, Technical University of Kosice. From August 2002 to July 2003, he was a Researcher with the Slovak Image Processing Center, Dobsina, Slovak Republic. From January 2003 to March 2003, he was a Postdoctoral Fellow with the Artificial Intelligence and Information Analysis Laboratory, Aristotle University of Thessaloniki, Greece. Since May 2003, he has been a Postdoctoral Fellow with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. He is a contributor to four books, and he has published over 200 papers in the areas of digital camera image processing, color image and video processing, multimedia security, and microarray image processing. He is a coeditor of the book *Color Image Processing: Methods and Applications* (CRC Press,

2006). He is a Guest Coeditor of the *Real-Time Imaging*, Special Issue on Multi-Dimensional Image Processing, and of the *Computer Vision and Image Understanding*, Special Issue on Color Image Processing for Computer Vision and Image Understanding. He is an Associate Editor for the *Journal of Real-Time Image Processing*. He serves as a Technical Reviewer for various scientific journals, and he participates as a member of numerous international conference committees.

Dr. Lukac is a member of the European Association for Signal, Speech, and Image Processing (EURASIP), IEEE Circuits and Systems Society, IEEE Consumer Electronics Society, and IEEE Signal Processing Society. In 2003, he was the recipient of the North Atlantic Treaty Organization/National Sciences and Engineering Research Council of Canada (NATO/NSERC) Science Award.

Konstantinos N. Plataniotis (S'90–M'92–SM'03) received the B.Eng, degree in computer engineering from the Department of Computer Engineering and Informatics, University of Patras, Patras, Greece, in 1988 and the M.S. and Ph.D. degrees in electrical engineering from the Florida Institute of Technology, Melbourne, in 1992 and 1994, respectively.

From August 1997 to June 1999, he was an Assistant Professor with the School of Computer Science at Ryerson University. He is currently an Associate Professor at the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, where he researches and teaches image processing, adaptive systems, and multimedia signal processing. He co-authored, with A. N. Venetsanopoulos *Color Image Processing & Applications* (Springer Verlag, 2000,). He is a contributor to seven books and has published more than 300 papers in refereed journals and conference proceedings in the areas of multimedia signal processing, image processing, adaptive systems, communications systems, and stochastic estimation.

Dr. Plataniotis is a Senior Member of IEEE, an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS, a past member of the IEEE Technical Committee on Neural Networks for Signal Processing. He was the Technical Co-Chair of the Canadian Conference on Electrical and Computer Engineering (CCECE) 2001, and CCECE 2004. He is the Technical Program Chair of the 2006 IEEE International Conference in Multimedia and Expo (ICME 2006), the Vice-Chair for 2006 IEEE Intelligent Transportation Systems Conference (ITSC 2006), and the Image Processing Area Editor for the IEEE Signal Processing Society e-letter. He is the 2005 IEEE Canada "Outstanding Engineering Educator" Award recipient and the co-recipient of the 2006 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award.