Example of Application for Image Compression

Amir Said,©1999

1 Example with SPIHT algorithm

Figure 1 shows the example of data in a small pyramid structure, of the type resulting from an image wavelet decomposition, that was used by J.M. Shapiro in his paper "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing,*, vol. 41, Dec. 1993, to describe his EZW image coding algorithm.

We applied the SPIHT algorithm to the same set of data, for one pass. The results are shown in Table 1, indicating the data coded and the updating on the control lists (to save space only the modifications are shown). The notation is defined in the patent description of the algorithm. For a quick reference, here are some of the important definitions.

LIS List of insignificant sets: contains sets of wavelet coefficients which are defined by tree structures, and which had been found to have magnitude smaller than a threshold (are insignificant). The sets exclude the coefficient corresponding to the tree or all subtree roots, and have at least four elements.

LIP List of insignificant pixels: contains individual coefficients that have magnitude smaller than the threshold.

LSP List of significant pixels: pixels found to have magnitude larger that the threshold (are significant).

$\mathcal{O}(i,j)$ in the tree structures, the set of offspring (direct descendants) of a tree node defined by pixel location (i,j).

$\mathcal{D}(i,j)$ set of descendants of node defined by pixel location (i,j).

$\mathcal{L}(i,j)$ set defined by $\mathcal{L}(i,j) = \mathcal{D}(i,j) - \mathcal{O}(i,j)$.

The following refer to the respective numbered entries in Table 1:

(1) These are the initial SPIHT settings. The initial threshold is set to 32. The notation (i,j)A or (i,j)B, indicates that an LIS entry is of type 'A' or 'B', respectively. Note the duplication of co-ordinates in the lists, as the sets in the LIS are trees without the roots. The coefficient (0,0) is not considered a root.

	0	1	2	3	4	5	6	7
0	63	–34	49	10	7	13	–12	7
1	–31	23	14	–13	3	4	6	–1
2	15	14	3	–12	5	–7	3	9
3	–9	–7	–14	8	4	–2	3	2
4	–5	9	–1	47	4	6	–2	2
5	3	0	–3	2	3	–2	0	4
6	2	–3	6	–4	3	6	3	6
7	5	11	5	6	0	3	–4	4

Figure 1: Set of image wavelet coefficients used by example. The numbers outside the box indicate the set of co-ordinates used.

(2) SPIHT begins coding the significance of the individual pixels in the LIP. When a coefficient is found to be significant it is moved to the LSP, and its sign is also coded. We used the notation 1+ and 1– to indicate when a bit 1 is immediately followed by a sign bit.

(3) After testing pixels it begins to test sets, following the entries in the LIS (active entry indicated by bold letters). In this example $\mathcal{D}(0,1)$ is the set of 20 coefficients {(0,2), (0,3), (1,2), (1,3), (0,4), (0,5), (0,6), (0,7), (1,4), (1,5), (1,6), (1,7), (2,4), (2,5), (2,6), (2,7), (3,4), (3,5), (3,6), (3,7)}. Because $\mathcal{D}(0,1)$ is significant SPIHT next tests the significance of the four offspring {(0,2), (0,3), (1,2), (1,3)}.

(4) After all offspring are tested, (0,1) is moved to the end of the LIS, and its type changes from 'A' to 'B', meaning that the new LIS entry meaning changed from $\mathcal{D}(0,1)$ to $\mathcal{L}(0,1)$ (i.e., from set of all descendants to set of all descendants minus offspring).

(5) Same procedure as in comments (3) and (4) applies to set $\mathcal{D}(1,0)$. Note that even though no offspring of (1,0) is significant, $\mathcal{D}(1,0)$ is significant because $\mathcal{L}(1,0)$ is significant.

(6) Since $\mathcal{D}(1,1)$ is insignificant, no action need to be taken. The algorithm moves to the next element in the LIS.

(7) The next LIS element, (0,1), is of type 'B', and thus $\mathcal{L}(0,1)$ is tested. Note that the co-ordinate (0,1) was moved from the beginning of the LIS in this pass. It is now tested again, but with another interpretation by the algorithm.

(8) Same as above, but $\mathcal{L}(1,0)$ is sigificant, so the set is partitioned in $\mathcal{D}(2,0)$, $\mathcal{D}(2,1)$, $\mathcal{D}(3,0)$, and $\mathcal{D}(3,1)$, and the corresponding entries are added to the LIS. At the same time, the entry (1,0)B

is removed from the LIS.

(9) The algorithm keeps evaluating the set entries as they are appended to the LIS.

(10) Each new entry is treated as in the previous cases. In this case the offspring of (2,1) are tested.

(11) In this case, because $\mathcal{L}(2,1) = \emptyset$ (no descendant other than offspring), the entry (2,1)A is removed from the LIS (instead of having its type changed to 'B').

(12) Finally, the last two entries of the LIS correspond to insignificant sets, and no action is taken. The sorting pass ends after the last entry of the LIS is tested.

(13) The final list entries in this sorting pass form the initial lists in the next sorting pass, when the threshold value is 16.

Without using any other form of entropy coding, the SPIHT algorithm used 29 bits in this first pass.

2 Example with EZW algorithm

Table 2 shows the results obtained with the EZW algorithm. The explantions, and original definition, can be found in the paper by J.M. Shapiro mentioned above. We use the abbreviation DL and SL for Shapiro's dominant and subordinate lists, respectively. The notation of **F** following a co-ordinate on the dominant list means that that an internal flag is set to indicate "significant" on that pass and its magnitude on the dominant list is set to 0 for subsequent passes.

Assuming the EZW uses (at least initially) two bits to code symbols in the alphabet {POS, NEG, ZTR, IZ}, and one bit to code the symbol Z, the EZW algorithm used 26+7=33 bits in the first pass. Since both methods are coding the same bit-plane defined by the threshold 32, both find the same set of significant coefficients, yielding images with the same mean squared error. However, SPIHT used about 10% less bits to obtain the same results because it coded different data.

The final lists may have some equal co-ordinates, but as shown in the examples, the interpretation and use of those co-ordinates by the two methods are quite different. Also, in the following passes they grow and change differently.

Comm.	Pixel or Set Tested	Output Bit	Action	Control Lists
(1)				LIS = {(0,1)A,(1,0)A,(1,1)A} LIP = {(0,0),(0,1),(1,0),(1,1)} LSP = $\emptyset$
(2)	$(0,0)$	1+	(0,0) to LSP	LIP = {(0,1),(1,0),(1,1)} LSP = {(0,0)}
	$(0,1)$	1–	(0,1) to LSP	LIP = {(1,0),(1,1)} LSP = {(0,0),(0,1)}
	$(1,0)$	0	none	
	$(1,1)$	0	none	
(3)	$\mathcal{D}(0,1)$	1	test offspring	LIS = {**(0,1)A**,(1,0)A,(1,1)A}
	$(0,2)$	1+	(0,2) to LSP	LSP = {(0,0),(0,1),(0,2)}
	$(0,3)$	0	(0,3) to LIP	LIP = {(1,0),(1,1),(0,3)}
	$(1,2)$	0	(1,2) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2)}
	$(1,3)$	0	(1,3) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3)}
(4)			type changes	LIS = {(1,0)A,(1,1)A,(0,1)B}
(5)	$\mathcal{D}(1,0)$	1	test offspring	LIS = {**(1,0)A**,(1,1)A,(0,1)B}
	$(2,0)$	0	(2,0) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0)}
	$(2,1)$	0	(2,1) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1)}
	$(3,0)$	0	(3,0) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0)}
	$(3,1)$	0	(3,1) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1)}
			type changes	LIS = {(1,1)A,(0,1)B,(1,0)B}
(6)	$\mathcal{D}(1,1)$	0	none	LIS = {**(1,1)A**,(0,1)B,(1,0)B}
(7)	$\mathcal{L}(0,1)$	0	none	LIS = {(1,1)A,**(0,1)B**,(1,0)B}
(8)	$\mathcal{L}(1,0)$	1	add new sets	LIS = {(1,1)A,(0,1)B,(2,0)A,(2,1)A,(3,0)A,(3,1)A}
(9)	$\mathcal{D}(2,0)$	0	none	LIS = {(1,1)A,(0,1)B,**(2,0)A**,(2,1)A,(3,0)A,(3,1)A}
(10)	$\mathcal{D}(2,1)$	1	test offspring	LIS = {(1,1)A,(0,1)B,(2,0)A,**(2,1)A**,(3,0)A,(3,1)A}
	$(4,2)$	0	(4,2) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1),(4,2)}
	$(4,3)$	1+	(4,3) to LSP	LSP = {(0,0),(0,1),(0,2),(4,3)}
	$(5,2)$	0	(5,2) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1),(4,2),(5,2)}
	$(5,3)$	0	(5,3) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1),(4,2),(5,2),(5,3)}
(11)			(2,1) removed	LIS = {(1,1)A,(0,1)B,(2,0)A,(3,0)A,(3,1)A}
(12)	$\mathcal{D}(3,0)$	0	none	LIS = {(1,1)A,(0,1)B,(2,0)A,**(3,0)A**,(3,1)A}
	$\mathcal{D}(3,1)$	0	none	LIS = {(1,1)A,(0,1)B,(2,0)A,(3,0)A,**(3,1)A**}
(13)				LIS = {(1,1)A,(0,1)B,(2,0)A,(3,0)A,(3,1)A} LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1),(4,2),(5,2),(5,3)} LSP = {(0,0),(0,1),(0,2),(4,3)}

Table 1: Example of image coding using the SPIHT method.

Tree Root	Output Symbol	DL: dominant list SL: subordinate list
		DL = {(0,0)} SL = ∅
(0,0)	POS	DL = {(0,0)**F**,(0,1),(1,0),(1,1)} SL = {63}
(0,1)	NEG	DL = {(0,0)**F**,(1,0),(1,1),(0,2),(0,3),(1,2),(1,3),(0,1)**F**} SL = {63,34}
(1,0)	IZ	DL = {(0,0)**F**,(1,0),(1,1),(0,2),(0,3),(1,2),(1,3),(0,1)**F**,(2,0),(2,1),(3,0),(3,1)}
(1,1)	ZTR	
(0,2)	POS	DL = {(0,0)**F**,(1,0),(1,1),(0,3),(1,2),(1,3),(0,1)**F**,(2,0),(2,1),(3,0),(3,1),(0,4),(0,5),(1,4),(1,5),(0,2)**F**} SL = {63,34,49)}
(0,3)	ZTR	
(1,2)	ZTR	
(1,3)	ZTR	
(2,0)	ZTR	
(2,1)	IZ	DL = {(0,0)**F**,(1,0),(1,1),(0,3),(1,2),(1,3),(0,1)**F**, (2,0),(2,1),(3,0),(3,1),(0,4),(0,5),(1,4),(1,5),(0,2)**F**, (4,2),(4,3),(5,2),(5,3)}
(3,0)	ZTR	
(3,1)	ZTR	
(0,4)	Z	
(0,5)	Z	
(1,4)	Z	
(1,5)	Z	
(4,2)	Z	
(4,3)	POS	DL = {(0,0)**F**,(1,0),(1,1),(0,3),(1,2),(1,3),(0,1)**F**, (2,0),(2,1),(3,0), (3,1),(0,4),(0,5),(1,4),(1,5),(0,2)**F**, (4,2),(5,2),(5,3),(4,3)**F**} SL = {63,34,49,47}
(5,2)	Z	
(5,3)	Z	
		DL = {(0,0)**F**,(1,0),(1,1),(0,3),(1,2),(1,3),(0,1)**F**,(2,0),(2,1),(3,0), (3,1),(0,4),(0,5),(1,4),(1,5),(0,2)**F**, (4,2),(5,2),(5,3),(4,3)**F**} SL = {63,34,49,47}

Table 2: Example of image coding using Shapiro's EZW method.