Fast Coding of Haar Wavelet Trees

Dylan Tarter & Dr. Brian Nutter

Texas Tech University

INTRODUCTION

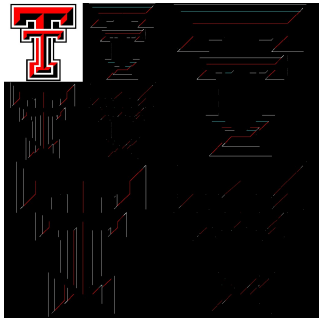
- Wavelet based compression involves taking recursive wavelet transforms on an image to generate coefficients which are then encoded and decoded using various compression techniques.
- Wavelet compression methods have high compression with high PSNR, but some wavelet transforms are slow to compute.
- On low end devices with slow clock speeds and low memory, it is desirable to have a simple algorithm with high speed and low memory cost.
- CHWT algorithm simplifies the BCWT algorithm and uses the Haar Wavelet for better compression on non-image datasets.

OBJECTIVES

- Establish an algorithm that is faster than BCWT with minimal cost to PSNR ideal for non-image datasets
- Utilize the Haar Wavelet Transform to simplify the algorithm in three key ways:
  1. No edge conditions in the transform
  2. Simplified indexing of coefficients
  3. Entirely integer-based algorithm

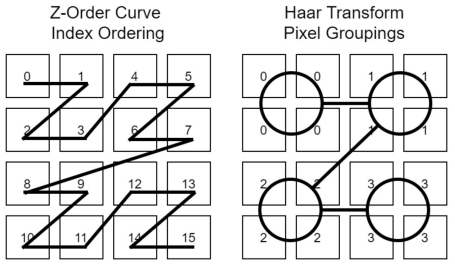
1. No Edge Conditions

- Haar wavelet transform performs calculations on groups of 2x2 pixels in an image with no overlap.
- If the image is of size 2^nx2^n, the image can be evenly divided into these groups, and recursively transformed until there is only one approximation coefficient.
- No edge conditions greatly simplifies the transform execution, especially on low end devices that are not designed to favor branching.
- Below, the Texas Tech Logo is transformed twice, and the coefficients are all contained within the original 1024x1024 set of pixels.



2. Simplified Indexing

- It was discovered while coding the algorithm that the indexes accessed for Haar wavelet transform calculations mimics the indexing of a Z-Order Curve (later referred to as a Z-Curve transform).
- If the original image is first Z-Curve transformed, the Haar wavelet transform can be performed by iterating through a 1-D array of pixels in steps of 4.
- The BCWT algorithm also visits coefficients in a manor similar to the Z-Order Curve indices. It encodes coefficients in 2x2 groups of 2x2 coefficients. The groups that need to be encoded are spatially next to each other in the 1-D array.



3. Integer-Based Algorithm

- The equations used in the algorithm operate entirely on signed and unsigned integers.
- The Haar wavelet transform equations are simply addition and subtraction, with a division by 4.
- The division by 4 can be omitted because the compression control factor Q_{min} equates to a right bit shift which equates to a division by a power of 2.
- Effectively, if Q_{min} equal 2, the Haar transform with be computed as normal, with two right bit shifts, but if Q_{min} equals 0, a lossless encoding can be done.
- Below is the equation used for the transform, where "k" is a 1-D iterator which is made possible from the Z-Curve transform.

$$A\left[\frac{k}{4}\right] = I[k] + I[k+1] + I[k+2] + I[k+3]$$
$$H\left[\frac{k}{4}\right] = I[k] + I[k+1] - I[k+2] - I[k+3]$$
$$V\left[\frac{k}{4}\right] = I[k] - I[k+1] + I[k+2] - I[k+3]$$
$$D\left[\frac{k}{4}\right] = I[k] - I[k+1] - I[k+2] + I[k+3]$$

RESULTS

Results on the Lena Test Image in Grayscale

BPP	PSNR (dB – dB)		Encoding (ms / ms)		Decoding (ms / ms)	
	KDU - CHWT	TTC – CHWT	KDU / CHWT	TTC / CHWT	KDU / CHWT	TTC / CHWT
3.88	5.13	3.73	3.33	4.26	2.96	3.29
2.67	4.76	3.46	4.80	5.33	3.17	3.90
1.644	4.34	3.6	5.09	5.77	2.55	4.47
0.993	4.73	3.96	5.58	5.96	1.85	3.38
0.6	5.36	4.46	4.19	5.03	2.34	4.68
0.36	5.35	4.87	3.78	4.21	1.87	4.20
Mean:	**4.95**	**4.01**	**4.46**	**5.09**	**2.46**	**3.99**

Results on the Lena Test Image in Color

BPP	PSNR (dB – dB)		Encoding (ms / ms)		Decoding (ms / ms)	
	KDU - CHWT	TTC – CHWT	KDU / CHWT	TTC / CHWT	KDU / CHWT	TTC / CHWT
11.94	3.60	4.52	3.19	3.93	2.80	3.17
8.28	5.09	4.10	3.45	4.49	2.51	3.07
4.83	4.34	3.48	3.30	4.03	2.15	3.28
2.54	4.43	3.60	3.19	3.58	1.47	3.16
1.39	4.80	4.56	3.33	4.99	1.47	3.22
0.79	5.13	4.64	2.30	3.84	1.06	3.35
Mean:	**4.57**	**4.15**	**3.13**	**4.14**	**1.91**	**3.21**

TTU Gray x512

TTU Gray x256

TTU Gray x128

TTU Gray Logo x512

BPP	PSNR (dB – dB)			Encoding (ms / ms)		
	CHWT	KDU	KDU - CHWT	CHWT	KDU	KDU / CHWT
0.77	55	63	8	2	9	4.5
0.46	45	52	7	2	8	4
0.19	30	39	9	2	8	4

TTU Gray Logo x512

1.37	54	60	6	1	4	4
0.83	43	45	2	0.7	3	4.28
0.47	31	38	5	0.7	3	4.28

TTU Gray Logo x512

2.25	54	51	-3	0.3	2	6.67
1.35	40	39	-1	0.17	1	5.88
1.04	34	36	2	0.18	1	5.55

CONCLUSIONS

- When implemented in C++, CHWT compresses roughly 4 times faster than BCWT (TTC) and Kakadu's JPEG2000 (KDU) but has around 5dB lower PSNR.
- On smaller, more sparse data sets such as the TTU logo in gray, CHWT
- Compression algorithm is simplified and suitable for low end devices.

ACKNOWLEDGEMENTS

This research has been supported by the Graduate Assistant in Areas of National Needs (GAANN) Scholarship from Texas Tech University