Preliminary Tests of SPIHT and SPECK Encoders on Scientific Data Sets

Category: Research

Abstract

SPIHT and SPECK encoders were originally developed by the image processing community. They provide high image compression ratios while keep the distortions low. We investigate the application of SPIHT and SPECK on scientific data sets in this document.

1 Introduction

SPIHT [6] and SPECK [5] were originally proposed as 2D image encoders. They aim to achieve high image compression ratios while introduce a minimal amount of distortion. Emperically, SPIHT, SPECK, and JPEG 2000 [5] share similar *computational complexity* and *accuracy* on images.

Researchers later extended SPIHT and SPECK to encode three-dimensional data [3, 7]. In this document, we report results from our tests of SPIHT and SPECK on 3D scientific data sets.

2 Study Overview

Our study evaluates the reconstruction accuracy of SPIHT and SPECK on various data sets. We compare results from SPIHT and SPECK with two other compression techniques: 1) Discrete Wavelet Transform (DWT), and 2) simple truncation that casts a double-precision floating point number to single-precision. More specifically, we operate on single-precision floating point numbers when comparing with DWT, and on double-precision floating point numbers when comparing with truncation. The implementation of SPIHT and SPECK are from QccPack [2], and the DWT implementations are from VAPOR [1].

DWTs can use different wavelet kernels to perform wavelet transforms, and each wavelet kernel has its unique characteristics. We specifically choose the CDF 9/7 kernel in this study, since research by Li et al. [4] has shown that the CDF 9/7 kernel has most accuracy when compressing scientific data sets with similar properties.

We note that the SPIHT and SPECK encoders also involve wavelet transforms as a subroutine (the complete pipeline is wavelet transform $\rightarrow$ quantization $\rightarrow$ SPIHT or SPECK encoding). Multiple wavelet kernels can apply in this subroutine as well. To make a fair comparison, we keep using the CDF 9/7 kernel in this subroutine. Finally, we use the default quantization scheme from QccPack.

Our evaluation criteria are Root Mean Square Error (RMSE) and maximum point-wise difference (LMax). RMSE provides an averaged error evaluation and LMax provides the worst case bound.

3 Encoders vs. Wavelets

In this section, we compare the two encoders, SPIHT and SPECK, with discrete wavelet transform. Our evaluation compares reconstructed data with the raw data, and calculates the RMSE and LMax on each compressed form. The compression ratios are: 4:1, 8:1, 16:1, and 32:1. We use three 512^3 data sets, all from a turbulent-flow simulation: the X component of velocity (VX), the X component of vorticity (WX), and enstrophy. Figure 1 and 2 show the RMSE and LMax evaluations respectively.

Both RMSE and LMax evaluations show more accurate results from SPIHT and SPECK over wavelets. At the same time, the two encoders, SPIHT and SPECK, exhibit similar results. The advantage of SPIHT and SPECK varies on compression ratios: they are two orders of magnitude better on 4:1, but less than one order of magnitude better on 16:1 and 32:1. It is worth keep investigating

	RMSE Evaluation				
Comp.	VAPOR	SPIHT	VAPOR / SPIHT	SPECK	VAPOR / SPECK
VX component, data range (-2.8, 2.8)					
4to1	1.83E-03	1.33E-05	137.18	1.38E-05	132.33
8to1	4.14E-03	2.22E-04	18.64	2.28E-04	18.12
16to1	7.52E-03	1.07E-03	7.01	1.09E-03	6.92
32to1	1.28E-02	2.94E-03	4.37	3.00E-03	4.29
WX component, data range (-293, 332)					
4to1	5.23E-01	3.71E-03	140.80	3.84E-03	136.24
8to1	1.10E+00	6.03E-02	18.19	6.20E-02	17.68
16to1	1.81E+00	2.96E-01	6.11	3.02E-01	5.99
32to1	2.72E+00	7.64E-01	3.56	7.77E-01	3.50
Enstrophy component, data range (0, 82842)					
4to1	1.64E+01	1.46E-01	112.27	1.54E-01	106.06
8to1	3.76E+01	2.52E+00	14.88	2.62E+00	14.35
16to1	6.94E+01	1.13E+01	6.13	1.16E+01	6.00
32to1	1.12E+02	2.95E+01	3.81	2.96E+01	3.79

Figure 1: RMSE of three data sets, using various compression settings. The three compression techniques are wavelets from VAPOR, SPIHT, and SPECK. The improvements of SPIHT and SPECK against VAPOR are shown comparatively in blue background.

	LMax Evaluation				
Comp.	VAPOR	SPIHT	VAPOR / SPIHT	SPECK	VAPOR / SPECK
VX component, data range (-2.8, 2.8)					
4to1	1.49E-02	8.59E-05	173.33	8.50E-05	175.07
8to1	3.14E-02	1.48E-03	21.29	1.43E-03	21.94
16to1	6.02E-02	9.35E-03	6.44	9.33E-03	6.45
32to1	1.21E-01	2.38E-02	5.08	2.89E-02	4.19
WX component, data range (-293, 332)					
4to1	4.05E+00	2.31E-02	175.28	2.38E-02	170.23
8to1	8.48E+00	3.88E-01	21.84	3.87E-01	21.92
16to1	1.42E+01	2.42E+00	5.88	2.40E+00	5.90
32to1	2.45E+01	5.22E+00	4.68	6.55E+00	3.74
Enstrophy component, data range (0, 82842)					
4to1	1.51E+02	8.54E-01	176.28	1.11E+00	135.73
8to1	4.60E+02	2.13E+01	21.63	2.10E+01	21.95
16to1	9.25E+02	9.35E+01	9.90	9.37E+01	9.87
32to1	1.72E+03	3.35E+02	5.16	3.40E+02	5.07

Figure 2: LMax of three data sets, using various compression settings. The three compression techniques are wavelets from VAPOR, SPIHT, and SPECK. The improvements of SPIHT and SPECK against VAPOR are shown comparatively in blue background.

	LMax Evaluation		
Comp.	Truncation	SPIHT Rect. Double	SPECK Rect. Double
2to1	2.38E-07	3.35E-08	3.35E-08
3to1		3.36E-08	3.36E-08
4to1		3.48E-08	3.49E-08
6to1		1.16E-07	1.15E-07
8to1		3.58E-07	3.73E-07
	RMSE Evaluation		
Comp.	Truncation	SPIHT Rect. Double	SPECK Rect. Double
2to1	5.26E-08	3.35E-08	3.35E-08
3to1		3.35E-08	3.35E-08
4to1		3.35E-08	3.35E-08
6to1		3.71E-08	3.74E-08
8to1		7.20E-08	7.40E-08

Figure 3: LMax and RMSE errors of truncation, SPIHT, and SPECK encoders. Note that truncation only supports 2:1 compression ratio. The test data set is Marschner-Lobb data set at 256^3 resolution, ranging from 7.6E-6 to 1.

what are the scenarios to apply the two encoders with most gain in accuracy.

This test also shows how the data range affects accuracy. Our narrowest data range is 5.6 and widest is 82842. Interestingly, these three data sets show similar error rates at each compression level. This is a little counter-intuitive to us because the quantization step in SPIHT and SPECK is expected to introduce larger errors when the data range is wide. It is worth keep investigating the effects of data range when using SPIHT and SPECK.

4 Encoders vs. Truncation

In this section, SPIHT and SPECK encode and decode double-precision floating point numbers. Truncation casts a double-precision number to single-precision (this is widely used when saving the simulation results onto disk). From the standpoint of compression, truncation achieves a compression ratio of 2:1. Finally, we always use the raw data in double-precision as the baseline to compare the compressed data.

Figure 3 shows the comparison between the two encoders and truncation using the Marschner-Lobb data set. This data set has a resolution of 256^3 and data range (7.6E-6, 1). The results show that both SPIHT and SPECK introduce less error at the 2:1 level, the compression ratio that truncation achieves. Also, the two encoders reach comparable error rates with truncation at more aggressive compression ratios, between 6:1 and 8:1 in this case. We are especially pleased to see that SPIHT and SPECK have superior performance than truncation even in terms of the LMax metric.

In a second test, we compare the SPECK encoder with truncation using a climate data set. This data set contains both 2D and 3D variables, and we especially tested 157 3D variables. These 3D variables have dimensions $60 \times 384 \times 320$. This data set is also special in that it has missing values (e.g. some vertices in the mesh are not valid). Since SPECK encoder is not able to handle missing values, we take a preprocessing step to replace all the missing values in a variable with the average of all the valid values. This preprocessing step was only for SPECK to encode and decode correctly, and we calculated RMSE and LMax only on the valid data points.

We group test results from all 157 variables into four categories. Each variable falls into one of the four categories based on how many times SPECK performs better than truncation: 1) no improvement ($< 1\times$ imprv.), 2) less than ten times improvement ($< 10\times$ imprv.), 3) less than a hundred times improvement ($< 100\times$ imprv.), and 4) equal or greater than a hundred times improvement ($>= 100\times$ imprv.). Figure 4 summarizes these four categories. Still, we consider the results from the LMax metric most important and they show significant benefit of using SPECK instead of truncation.

LMax of 157 variables							
< 1X imprv.		< 10X imprv.		< 100X imprv.		>= 100X imprv.	
num.	pct.	num.	pct.	num.	pct.	num.	pct.
0	0	25	15.92%	23	14.65%	109	69.43%

RMSE of 157 variables							
< 1X imprv.		< 10X imprv.		< 100X imprv.		>= 100X imprv.	
num.	pct.	num.	pct.	num.	pct.	num.	pct.
9	5.73%	76	48.41%	42	26.75%	30	19.11%

Figure 4: The number and percentage of variables in each of the four categories. Each category has variables that gain certain amount of improvement by using SPECK over truncation. The upper figure shows results calculated based on LMax error metric, and the bottom figure shows results calculated based on RMSE error metric.

5 Calculation Time

The SPECK and SPIHT encoders take significant amount of time to perform encoding and decoding tasks. Since truncation takes almost no time to complete, I compared SPIHT and SPECK with DWT. My simple and preliminary tests show that SPECK takes $13\times$ time to encode, and $21\times$ time to decode, when both SPECK and DWT run in serial. SPIHT has similar calculation time.

6 Additional Properties

Progressive Access. Both SPIHT and SPECK encode data into bit-streams. When decoding, any prefix of the bit-stream can be used to reconstruct the rectilinear mesh, and more incoming bits will contribute to more accurate reconstructions.

Always Lossy. SPIHT and SPECK quantize data before encoding, and de-quantize it after decoding. These steps make SPIHT and SPECK always introduce some error, even when encoding to a target size that is the same as the raw data. We can also observe it from Figure 3: SPIHT and SPECK have almost the same error with compression ratios 4:1, 3:1, and 2:1.

7 Conclusion

SPIHT and SPECK are state-of-the-art encoders from the image processing community, and their use on scientific data just starts. Our preliminary experiments show both pros and cons of these encoders. Namely:

Pros:

- Up to two orders of magnitude more accurate than DWT when performing lossy compression.
- Use $3\times$ to $4\times$ less space than truncation when saving double-precision floating point data with comparable error.
- Support progressive data access.

Cons:

- Computational intensive. Could be $20\times$ slower than DWT.
- Always introduce error. Only suit for lossy compression use cases.

References

[1] J. Clyne, P. Mininni, A. Norton, and M. Rast. Interactive desktop analysis of high resolution simulations: application to turbulent plume dynamics and current sheet formation. *New Journal of Physics*, 9(8):301, 2007.

[2] J. E. Fowler. Qccpack: An open-source software library for quantization, compression, and coding. In *International Symposium on Optical Science and Technology*, pages 294–301. International Society for Optics and Photonics, 2000.

[3] B.-J. Kim, W. Pearlman, et al. An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (spiht). In *Data Compression Conference, 1997. DCC'97. Proceedings*, pages 251–260. IEEE, 1997.

[4] S. Li, K. Gruchalla, K. Potter, J. Clyne, and H. Childs. Evaluating the efficacy of wavelet configurations on turbulent-flow data. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 2015.

[5] W. Pearlman, A. Islam, N. Nagaraj, A. Said, et al. Efficient, low-complexity image coding with a set-partitioning embedded block coder. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(11):1219–1235, 2004.

[6] A. Said, W. Pearlman, et al. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(3):243–250, 1996.

[7] X. Tang and W. A. Pearlman. Three-dimensional wavelet-based compression of hyperspectral images. In *Hyperspectral Data Compression*, pages 273–308. Springer, 2006.