Integer Klt For Lossless Hyperspectral Image Compression on BeagleBone Black Board

N.D. Mohd Ridzuan Tan[1], N.R. Mat Noor[1], W.A.F.W. Othman[1], E.A. Bakar[2], A.F. Hawary[2]

[1]School of Electrical & Electronic Eng., [2]School of Aerospace Eng., Universiti Sains Malaysia, 14300 Nibong Tebal, Pulau Pinang, Malaysia

Submitted: 23 July 2019 / Accepted: 27 September 2019 / Published online: 31 October 2019

ABSTRACT

This research concerned with the lossless hyperspectral image compression for satellite imagery using Integer Karhunen–Loève Tranform (KLT) on BeagleBone Black Board. The Integer KLT is selected as the transform for compression due to it showing superior performance in decorrelating the spectral component in hyperspectral images compared to other algorithms. The objective of this research is to develop an Integer KLT algorithm to implement it into the embedded system. The performance of the algorithm on the board in terms of execution time is investigated. The implementation of Integer KLT algorithm is executed into BeagleBone Black board and tested for its performance, profiling the execution time and comparing it with other embedded platforms which is a low-power DSP platform. The implementation shows a faster execution time compared to low-power DSP platform.

Keywords: Integer KLT; BeagleBone Black Board implementation; Execution time

Author Correspondence, e-mail: nrmn@usm.my

1. Introduction

In this current era, as technologies constantly being improved, the development of specific technologies such as cameras that are dedicated into Earth observing satellites are created and improved. These satellites are suitable for constant surveillance with no interruption as compared to airborne platforms [1]. The images captured by the satellites are called hyperspectral images. These hyperspectral images have a wide applications such as atmospheric detecting, remote sensing [1], military affairs [2] and so-on. While these images significantly helped the humankind, there are still an on-going common data-capacity related problem which are the limited amount of on-board memory and the limited speed of the communication channel between a satellite and ground station.

In hyperspectral remote sensing where the amount of data usually is in the range of hundreds of bands, the problems become even more serious [3]. In order to overcome this problem, a compression system is introduced. Integer Karhunen-Loeve Transform (KLT) was proposed in this research to achieve lossless hyperspectral image compression. Integer KLT is a modified version of the original KLT which is the lossless version of KLT. In KLT algorithm, there exists a non-integer output (floating-point) which actually leads to a lossy transformation causing some amount of data to be loss. By introducing matrix factorizations such as eigenvector and PLUS matrices, the Integer KLT is able to eliminate the floating-point output and achieves a lossless data compression process [4, 5]. In other words, there is zero missing data during the compression and decompression by using Integer KLT algorithm.

Embedded systems are often used in the recent trends. An embedded system is a specific-purpose system where it works akin to a computer. It has a combination of a computer's hardware and software function where it is capable of being programmable to be designed in a specific manor or function. Precisely, in this research, one of the embedded systems that will be used is the BeagleBone Black Board.

2. Overview on Hardware & Software Set-Up

This project involves the implementation and the optimization of the integer KLT algorithm for hyperspectral image compression using the beaglebone-black board. The hardware implementation, beaglebone-black uses Linux system with a memory of 512MB DDR3L and runs at the speed of 800MHz. The beaglebone-black is connected to the PC via USB cable where coding algorithm is able to transfer the code files into beaglebone-black.

To run the developed algorithm, a coding software Eclipse is used as it is compatible with beaglebone-black by changing the configuration set-up in the GCC and G++ Linker Build Setting. The project mainly uses clustering technique for hyperspectral compression since it is one of the optimization techniques that can be used to reduce the computation complexity of algorithm and it also helped to increase the lossless compression ratio performance.

3. Implementation of Integer KLT on BeagleBone Black Board

The basic vectors of KLT are the eigenvectors of matrix covariance where it removes the correlation of neighbouring pixels. Depending on the data, KLT is one of the best transforms that are effective in data decorrelation if we ignore the floating-point output matrix it produces. Integer KLT algorithm, a modified version of KLT that are used for this project starts with the calculation of mean of each band. Once the mean is rounded off, it is subtracted from the original hyperspectral image, H. At this point, covariance, eigenvector/eigenvalue and the P, L, U and S matrix factorization are done towards the image. When the development of Integer KLT algorithm are done, the code is then uploaded onto the beaglebone-black through Ubuntu terminal platform.

Clustering in Integer KLT is performed by encoding a group of z bands rather than the total number of bands, Z in a hyperspectral image, where $z \leq Z$ [4]. A normal clustering which is a local decorrelation within each cluster are sufficient to be used. Encoding process of the AVIRIS and Hyperion hyperspectral image are represented by the clustering process that is repeated for a number of iterations, c, where:

$$c = \frac{Z}{z} \tag{1}$$

The clustering levels for an AVIRIS of a total 224 bands and Hyperion of a total of 196 bands has few different levels in cluster size. The minimum number of clusters, *c*, is the maximum cluster size, which is *Z/c*. So, the values of *c* are dependant of the number of bands *Z*. However, to avoid the Integer KLT performance as spectral decorrelator being ineffective, the cluster size *Z* must not be lower than four. Table 1 and Table 2 shows the AVIRIS and Hyperion clustering levels, respectively. The lowest number of clusters, *c* requires a huge volume of memory. So in certain cases of low-powered embedded platform, $c = 1$ cannot run on the platforms due to its large memory.

Table 1. Clustering Levels for AVIRIS Datasets

No. of Clusters, *c*	**Cluster Size, *Z*/*c***
1 (lowest)	224
2	112
4	56
7	32
8	28
14	16
16	14
28	8
32	7
56 (highest)	4

Table 2. Clustering Levels for Hyperion Datasets

No. of Clusters, *c*	**Cluster Size, *Z*/*c***
1 (lowest)	196
2	98
4	49
7	28
14	14
28	7
49 (highest)	4

10 images are selected from AVIRIS and Hyperion dataset respectively as test images for the purpose of this project. Each AVIRIS images are cropped to a size of 512×512×224 while each Hyperion image datasets are cropped to 256×256×196 size. Table 3 and Table 4 shows the AVIRIS and Hyperion datasets, respectively, along with the abbreviation of each hyperspectral image used for testing.

Table 3. AVIRIS Datasets

Hyperspectral Image	Abbreviation
Cuprite Scene 1	Cuprite1
Jasper Ridge Scene 1	Jasper1
Low Altitude Scene 1	Low1
Low Altitude Scene 5	Low5
Lunar Lake Scene 1	Lunar1
Yellowstone Calibrated Scene 0	YSCal0
Yellowstone Calibrated Scene 3	YSCal3
Yellowstone Calibrated Scene 10	YSCal10
Yellowstone Calibrated Scene 11	YSCal11
Yellowstone Calibrated Scene 18	YSCal18

Table 4. Hyperion Datasets

Hyperspectral Image	Abbreviation
EO1H1660512002107110PZ_SGS_01	Atturbah
EO1H1700782002055110PY_SGS_01	Benoni
EO1H0120312001129111P1_PF1_01	Boston
EO1H0920842002053311 0PY_AGS_01	Coolamon
EO1H0910822002071110PY_AGS_01	Dubbo1
EO1H0140362001127110PP_AGS_01	Edenton
EO1H0090112001140111PP_PF1_01	Greenland
EO1H1370392002032110PZ_SGS_01	Maizhokunggar
EO1H1090232002092110PZ_AKS_01	Okha
EO1H0150332001134111P1_AGS_01	Portobago

These images are used to test for Integer KLT hyperspectral image compression on beaglebone-black board while the average of the images is collected.

4. Experimental Results

The execution time of Integer KLT algoritm implementation on beaglebone-black board is evaluated for each dataset based on the number of clustering. The average execution time of the algorithm were tabulated for the average execution time of 10 images for each AVIRIS and Hyperion datasets respectively. Generally, the higher the number of cluster and the smaller the cluster size, it can be seen that clustering allows the speed up of the compression of image. The result in Table 5 and Table 6 shows that clustering technique is able to improve the execution time. This is due to the fact that the number of spectral bands that are to be encoded in each cluster are reduced significantly. The number of bands in Integer KLT are one of the main factors that contributed to the complexity of the algorithm. So, when there are a smaller number of bands per cluster, the complexity of overhead information is reduced which leads to the improvement of execution time. While referring back to Table 1 (AVIRIS) and Table 2 (Hyperion) where the clustering size starts with 224 (AVIRIS) and 196 (Hyperion), the implementation can only starts with 32 and 98 for AVIRIS and Hyperion, respectively due to the insufficient memory issue of the hardware. This is because bigger cluster size required larger volume of memory which is unable to be run on the hardware implementation.

Table 5. Execution Time Performance for AVIRIS on BeagleBone Black Board

Z/*c* (*c*)	Average Execution Time (seconds)
32 (7)	131.98
28 (8)	121.56
16 (14)	90.85
14 (16)	86.55
8 (28)	71.97
7 (32)	69.52
4 (56)	62.21

Table 6. Execution Time Performance for Hyperion on BeagleBone Black Board

Z/*c* (*c*)	Average Execution Time (seconds)
98 (2)	70.44
49 (4)	38.41
28 (7)	28.92
14 (14)	18.75
7 (28)	15.16
4 (49)	13.72

The performance analysis are obtained by comparing the performance of Integer KLT on beaglebone-black with other hardware implementation which, in this case are comparing with DSP OMAP-L137 from [4]. Table 7 and Table 8 shows the cluster size and the execution time for the low-power DSP and Beaglebone-black board for AVIRIS and Hyperion datasets, respectively.

Table 7. AVIRIS Execution Time Performance between BeagleBone Black and OMAP-L137

Cluster Size	Execution time (seconds)	
	DSP OMAP-L137	Beaglebone-Black
32	314.67	131.984
28	302.38	121.558
16	204.64	90.853
14	196.72	86.549
8	149.88	71.966
7	145.98	69.523
4	124.96	62.210

Table 8. Hyperion Execution Time Performance between BeagleBone Black and OMAP-L137

Cluster Size	Execution time (seconds)	
	DSP OMAP-L137	Beaglebone-Black
98	270.46	70.435
49	127.34	38.410
28	67.01	28.919
14	43.56	18.751
7	32.22	15.157
4	27.54	13.718

The performance are measured based on the type of drives and RAM available for the hardwares. For the low-power DSP, some of the key features taken from research paper by Noor are such that it is a dual-core DSP chip, has 64 MB of SDRAM and runs at 300MHz whereas for Beaglebone-black has 512 MB DDR3L SDRAM and runs at 800MHz. Referring to Table 5, the execution time for Beaglebone-black is better than DSP implementation. This is because of a couple of reasons which are the different type of processor used in both hardware implementation, the operating system and the coding implementation approach. Beaglebone-black runs on DDR3 which requires less power but able to run faster compared to DSPs' dual-core chipe. Besides, compared to DSP 64 MB of SDRAM, beaglebone-black has a faster and greater size at 512 MB SDRAM which allows a better processing speed. The operating system for DSP uses processor instrumentation set while beaglebone-black uses Linux based operating system which may affect the execution time.

Acknowledgements

This work was supported by Universiti Sains Malaysia Short Term Grant 304/PELECT/6315003.

References

1. M. Borengasser, W. S. Hungate, Watkins R. Hyperspectral Remote Sensing : Principles and Application. 2008:130.
2. Yuen PWT, Bishop G, Hyperspectral Algorithm Development for Military Applications: A Multiple Fusion Approach. 2006; Edinburgh, United Kingdom: Publisher.
3. Mat Noor NR, Vladimirova T, Parallel Implementation of Lossless Clustered Integer KLT Using OpenMP. 2012 25-28 June 2012; Erlangen, Germany: Publisher.
4. Mat Noor NR. Karhunen-Loève Transform based Lossless Hyperspectral Image Compression for Space Applications [eThesis]. Leicester: University of Leicester; 2015.
5. Egho C, Vladimirova T, Adaptive hyperspectral image compression using the KLT and integer KLT algorithms. 2014 14-17 July 2014; Leicester, U. Kingdom: Publisher.

How to cite this article:
N.D. Mohd Ridzuan Tan, N.R. Mat Noor, W.A.F.W. Othman, E.A. Bakar, A.F. Hawary. Integer KLT for Lossless Hyperspectral Image Compression on BeagleBone Black Board. ROBOTIKA, 2019, 1(2), 24-31