ACCELERATING DISCRETE WAVELET TRANSFORMS ON GPUS

David Barina and Michal Kula and Michal Matysek and Pavel Zemcik

Faculty of Information Technology
Brno University of Technology
Czech Republic

ABSTRACT

The two-dimensional discrete wavelet transform has a huge number of applications in image-processing techniques. Until now, several papers compared the performance of such transform on graphics processing units (GPUs). However, all of them only dealt with lifting and convolution computation schemes. In this paper, we show that corresponding horizontal and vertical lifting parts of the lifting scheme can be merged into non-separable lifting units, which halves the number of steps. We also discuss an optimization strategy leading to a reduction in the number of arithmetic operations. The schemes were assessed using the OpenCL and pixel shaders. The proposed non-separable lifting scheme outperforms the existing schemes in many cases, irrespective of its higher complexity.

Index Terms— Discrete wavelet transform, image processing, synchronization, graphics processors

1. INTRODUCTION

The discrete wavelet transform (DWT) has become a very popular image-processing tool in recent decades. This popularity has resulted in a development of fast algorithms on all sorts of computer systems, including graphics processing units (GPUs). Although the GPUs were originally optimized for the graphics rendering, general-purpose computing became practical with advances of programmable shaders. Later, CUDA and related general-purpose computing APIs were introduced, which brought wider possibilities and allowed to ignore the underlying graphical concepts.

So far, several studies have compared the performance of various 2-D DWT computational approaches on GPUs. All of these works are based on the most popular separable schemes (their operations are oriented horizontally or vertically) – the convolution and lifting schemes. The lifting requires fewer arithmetic operations as compared with the convolution, at the cost of introducing some data dependencies. The number of operations should be proportional to a transform performance. Interestingly, also the data dependencies may form a bottleneck, especially on parallel architectures.

We show that the optimal scheme for a given architecture can be obtained by fusing the corresponding lifting parts into a new non-separable structure. More precisely, underlying operations cannot be associated with horizontal nor vertical axes. In addition, we discuss an approach where this non-separable scheme can be adapted to a particular platform in order to reduce the number of operations. Our reasoning is supported by experiments on GPUs using OpenCL and

pixel shaders. The presented scheme is general, and it is not limited to any specific type of DWT.

The rest of this paper is organized as follows. Section 2 formally introduces the problem definition and presents the existing separable approaches. Section 3 presents the proposed non-separable scheme and discusses the optimization approach that reduces the number of operations. Section 4 evaluates the performance in the pixel shaders and OpenCL framework. Finally, Section 5 closes the paper.

2. BACKGROUND AND RELATED WORK

This section introduces some notations and definitions and also briefly reviews papers that motivated our research. The z-transform notation is used for the description of underlying one-dimensional wavelet FIR filters. The transfer function of the filter (g_k) is the polynomial

$$G(z) = \sum_k g_k \, z^{-k},$$

where the k refers to the time axis. Further, the one-dimensional transforms are used in conjunction with two-dimensional signals. For this case, the transfer function of the filter (g_{k_m,k_n}) is defined as the polynomial

$$G(z_m, z_n) = \sum_{k_m} \sum_{k_n} g_{k_m,k_n} \, z_m^{-k_m} z_n^{-k_n},$$

where the subscript m refers to the horizontal axis and n to the vertical one. The $G^*(z_m, z_n) = G(z_n, z_m)$ is a polynomial transposed to a polynomial $G(z_m, z_n)$. In this paper, a shortened notation G is only written in order to keep the notation readable.

A discrete wavelet transform is a mathematical tool which is suitable for the decomposition of a discrete signal into low-pass and high-pass frequency components. The single-scale transform splits the input signal into two components. As shown in [1], the transform can be computed by a pair of filters, referred to as G_0, G_1, followed by subsampling by a factor of two. Formally, the transform can also be represented by the polyphase matrix

$$\begin{bmatrix} G_1^{(o)} & G_1^{(e)} \\ G_0^{(o)} & G_0^{(e)} \end{bmatrix}, \tag{1}$$

where the polynomials $G^{(e)}$ and $G^{(o)}$ refer to the even and odd terms of G. This matrix defines the convolution scheme. Following the instructions by [2], the convolution scheme in (1) can be factored into a sequence

$$\prod_{K > k > 0} \begin{bmatrix} 1 & U^{(k)} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ P^{(k)} & 1 \end{bmatrix} \tag{2}$$

of short filters, known as the lifting scheme. The filters employed in (2) are referred to as the lifting steps. The first step $P^{(k)}$ in the kth pair is referred to as the predict and the second one $U^{(k)}$ to as the update. The lifting scheme reduces the number of operations by up to half. Note that the superscript (k) is omitted in the text below.

Considering the GPUs, the processing of single or several transform samples is mapped to independent processing units. The units must use some sort of synchronization method (barrier) to avoid race conditions. In the lifting scheme, the barriers are required before the lifting steps. The barriers are indicated by the | symbol. For example, $M_2|M_1$ are two adjacent lifting steps separated by the barrier.

The 2-D transform is defined as a tensor product of 1-D transforms. Consequently, the transform splits the signal into a quadruple of wavelet coefficients. Following the paper of [1], the 1-D transforms are applied in both directions sequentially. By its nature, this scheme can be referred to as the separable convolution. The calculations in a single direction are performed in a single step (two steps in total). The scheme can be described as

$$\mathbf{N}^V \,|\, \mathbf{N}^H \,|,$$

where $\mathbf{N}^H$ is 1-D transform in horizontal direction and $\mathbf{N}^V$ is in vertical one. For the well-known Cohen-Daubechies-Feauveau (CDF) wavelet with 9/7 samples, these matrices are illustrated in Fig. 1 (horizontal part only). Particularly, the filters in the figure are of sizes 9 and 7 taps. Note that the color balls correspond to the quadruple of wavelet coefficients.
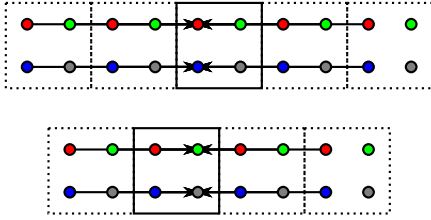
Fig. 1. Separable convolution for the CDF 9/7 wavelet (horizontal part only). Appropriately chosen pairs of matrix rows are depicted in separate subfigures. The arrows denote a multiply–accumulate operation (multiplication by a real constant) and they are pointing to the destination operand. Note that some arrows overlap the others.

Another scheme widely used for 2-D transform is the separable lifting. Similarly to the convolution, the predict and update lifting steps can be applied in both directions sequentially. Moreover, horizontal and vertical steps can be arbitrarily interleaved thanks to linear nature of the filters. In this paper, the scheme is defined as

$$S_U^V \,|\, S_U^H \,|\, T_P^V \,|\, T_P^H \,|,$$

(a) T_P^H (b) S_U^H
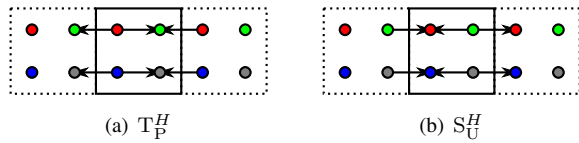
Fig. 2. Separable lifting scheme (horizontal part only) for the CDF 5/3 and 9/7 wavelets.

wherein the predict steps T always precede the update S ones. This mapping corresponds to a single P and U pair of lifting steps from (2). For multiple pairs, the scheme is separately applied to each such pair. To describe 2-D matrices, the lifting steps must be migrated into two dimensions as $G = G(z_m, z_n) = G(z_m)$. The individual steps are then defined as

$$T_P^H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ P & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & P & 1 \end{bmatrix},$$

$$T_P^V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ P^* & 0 & 1 & 0 \\ 0 & P^* & 0 & 1 \end{bmatrix},$$

$$S_U^H = \begin{bmatrix} 1 & U & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & U \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$S_U^V = \begin{bmatrix} 1 & 0 & U^* & 0 \\ 0 & 1 & 0 & U^* \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the CDF wavelets, the matrices are also illustrated in Fig. 2 (horizontal part only).

Until now, several papers compared the performance of the separable lifting and separable convolution schemes on GPUs. Exemplarily, the [3] compared these schemes on GPUs using pixel shaders. The authors mapped data to 2-D textures, constituted by four floating-point elements. They concluded that the separable convolution is more efficient than the separable lifting scheme in most cases. They further noted that fusing several consecutive kernels might significantly speed up the execution, even if the complexity of the resulting fused pixel program is higher.

To illustrate the problem more widely, also several other papers are discussed. Kucis *et al.* compared the performance of several recently published schedules (either the convolution or lifting) for computing the 2-D DWT using the OpenCL framework. In more detail, the work compares a convolution-based algorithm [4] against several lifting-based methods [5, 6] in the horizontal part of the transform. The authors concluded that the lifting-based algorithm of [5] is the fastest method. Furthermore, [6] compared the performance of their separable lifting-based method against a convolution-based method. They concluded that the lifting is the fastest method. The authors also compared the performance of implementations in CUDA and pixel shaders, based on the work of [3]. The CUDA implementation proved to be the faster choice. In this regard, they noted that a speedup in CUDA occurs because the CUDA effectively makes use of on-chip memory. This use is not possible in pixel shaders, which exchange the data using off-chip memory. Without providing further details, other important approaches can be found in the papers of [7, 8, 9], and most recently in [10].

In our previous works [11, 12], we introduced several non-separable schemes for calculation of 2-D DWT. In this paper, we take the most promising scheme and adapt it to particular platforms. Moreover, differences and similarities between the non-separable scheme and their separable competitors are homogeneously discussed. All schemes are also analyzed and evaluated.

3. PROPOSED SCHEME AND OPTIMIZATION

The existing approaches did not study the possibility of a partial fusion of lifting polyphase matrices. This section presents an alternative non-separable scheme for the calculation of the 2-D DWT.

By combining the corresponding horizontal and vertical steps of the separable lifting scheme, the non-separable lifting scheme is formed. The number of operations has slightly been increased. The scheme consists of a spatial predict and spatial update step, thus two steps in total for each pair of the original lifting steps. For each pair of P and U, the scheme follows from

$$S_U \mid T_P \mid,$$

where

$$T_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ P & 1 & 0 & 0 \\ P^* & 0 & 1 & 0 \\ PP^* & P^* & P & 1 \end{bmatrix},$$

$$S_U = \begin{bmatrix} 1 & U & U^* & UU^* \\ 0 & 1 & 0 & U^* \\ 0 & 0 & 1 & U \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note that the spatial filters in PP^* and UU^* may be computationally demanding, depending on their sizes. For the CDF wavelets, the scheme is graphically illustrated in Fig. 3.
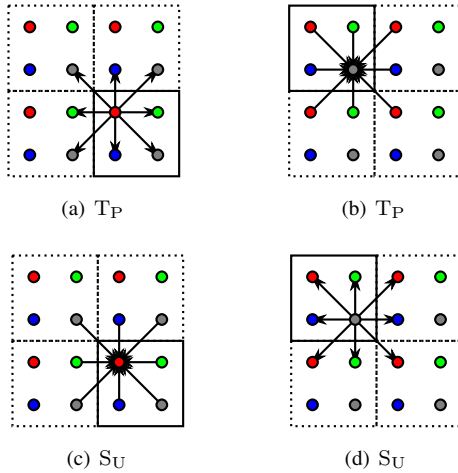

(a) T_P (b) T_P

(c) S_U (d) S_U

Fig. 3. Non-separable lifting scheme for the CDF wavelets.

An important observation can be made regardless the underlying platform. A very special form of the operations guarantees that the processing units never access the results belonging to their neighbors. These operations comprise only constants. Since the convolution is a linear operation, the polynomials can be collected out from the original matrices, and calculated in a different step. The original polynomials are split as $P = P_0 + P_1$ and $U = U_0 + U_1$. The P_0 and U_0 are the constants. As a next step, the P_0 and U_0 are substituted into the separable lifting scheme. On the contrary, the P_1 and U_1 are kept in the original non-separable lifting scheme. These two steps are then computed without any barrier. The observation is exploited to adapt the scheme for a particular platform.

The schemes for the shaders and OpenCL exploit the above-described observation with P_0, U_0 polynomials. Implementations

in the shaders map input and output data to 2-D textures. There is no possibility to retain results in registers, and the results are exchanged through textures in off-chip memory. Considering the OpenCL implementations, the image is divided into overlapping blocks and on-chip memory shared by all threads in a block is utilized to exchange the results. Additionally, some results are passed in registers.

Table 1. Steps and arithmetic operations for the optimized schemes.

(a) CDF 5/3

scheme		steps	operations	
			OpenCL	shaders
separable	convolution	2	20	22
separable	lifting	4	16	16
non-separable	lifting	2	18	18

(b) CDF 9/7

scheme		steps	operations	
			OpenCL	shaders
separable	convolution	2	56	58
separable	lifting	8	32	32
non-separable	lifting	4	36	36

This paper explores the performance for two frequently used wavelets, namely, CDF 5/3 and CDF 9/7 [13]. Their fundamental properties are listed in Table 1. Note that the number of operations is commonly proportional to a transform performance. Additionally, the number of steps correspond to the number of synchronizations on parallel architectures, which also form a performance bottleneck.

4. EVALUATION

The experiments were performed on GPUs of two biggest vendors using the OpenCL and pixel shaders. Only a transform performance was measured, usually in GB/s (gigabytes per second). The host system does not help in the calculation. Only results for two cards (AMD Radeon HD 6970, NVIDIA Titan X) are shown due to limited space. Their technical parameters are summarized in Table 2.

The implementations were created using the DirectX HLSL and OpenCL. The HLSL implementation is used on the NVIDIA Titan X, whereas the OpenCL implementation on the AMD 6970. The results of the performance comparison are shown in Fig. 4. The non-separable lifting schemes always outperform the separable lifting. Similar results were also achieved on other cards, especially considering the shaders. Looking at the experiments with the pixel shaders, some transients can be seen at the beginning of the plots. We concluded that these transients are caused by a suboptimal use of cache system, or alternatively by some overhead made by the graphics API.

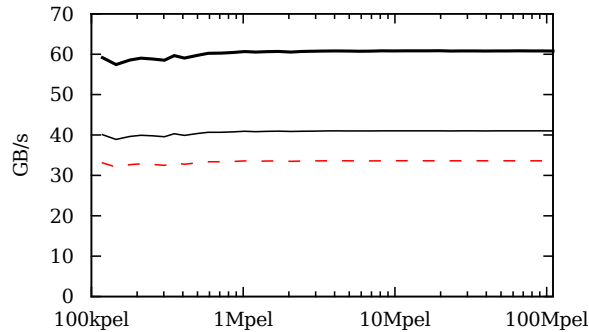**Table 2**. Description of the GPUs used for the evaluation.

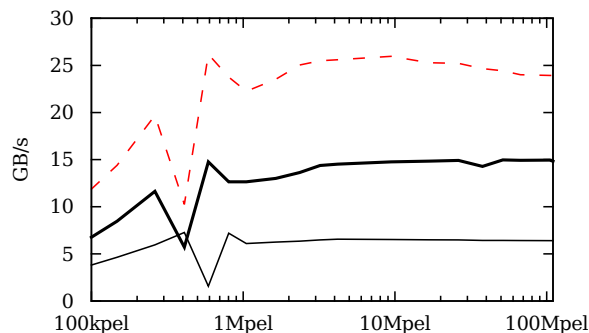| label	AMD 6970	NVIDIA Titan X
model	Radeon HD 6970	Titan X (Pascal)
multiprocessors	24	28
total processors	1 536	3 584
performance	2 703 GFLOPS	10 157 GFLOPS
bandwidth	176 GB/s	480 GB/s
on-chip memory	32 KiB	96 KiB
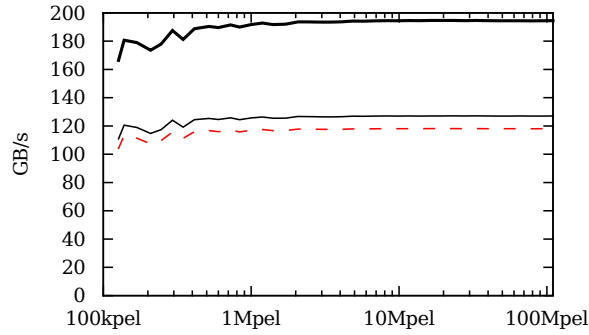
(a) CDF 5/3 in OpenCL
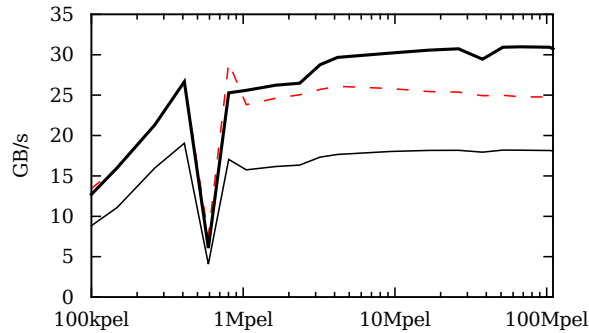
(b) CDF 5/3 in pixel shader

(c) CDF 9/7 in OpenCL

(d) CDF 9/7 in pixel shader

lifting convolution — non-separable lifting ——

Fig. 4. Performance (gigabytes per second) for both CDF wavelets.

5. CONCLUSIONS

This paper introduced and discussed non-separable lifting scheme for computation of the 2-D discrete wavelet transform on modern GPUs. As an option, an optimization strategy leading to a reduction in the number of arithmetic operations was presented. Using this strategy, the introduced scheme was adapted on the OpenCL framework and pixel shaders. The implementations were then evaluated using GPUs of two biggest vendors. At least for the CDF wavelets, the non-separable lifting scheme exhibit mostly a better performance than their separable competitors.

We reached the following conclusion. Fusing several consecutive steps of the schemes might significantly speed up the execution, irrespective of their higher complexity. Note that all of the schemes are general and they can be used on any discrete wavelet transform. In future work, we plan to tackle with multi-scale transforms.

6. REFERENCES

[1] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 674–693, 1989.

[2] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998.

[3] C. Tenllado, J. Setoain *et al.*, "Parallel implementation of the 2D discrete wavelet transform on graphics processing units: Filter bank versus lifting," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 3, pp. 299–310, 2008.

[4] V. Galiano, O. Lopez *et al.*, "Improving the discrete wavelet transform computation from multicore to GPU-based algorithms," in *Int. Conf. on Computational and Mathematical Methods in Science and Engineering*, Jun. 2011, pp. 544–555.

[5] M. Blazewicz, M. Ciznicki *et al.*, *Two-Dimensional Discrete Wavelet Transform on Large Images for Hybrid Computing Architectures: GPU and CELL.* Springer, 2012, pp. 481–490.

[6] W. J. van der Laan, A. C. Jalba, and J. Roerdink, "Accelerating wavelet lifting on graphics hardware using CUDA," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 1, pp. 132–146, 2011.

[7] J. Matela, "GPU-based DWT acceleration for JPEG2000," in *Annual Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, 2009, pp. 136–143.

[8] V. Galiano, O. Lopez *et al.*, "Parallel strategies for 2D discrete wavelet transform in shared memory systems and GPUs," *The Journal of Supercomputing*, vol. 64, no. 1, pp. 4–16, 2013.

[9] C. Song, Y. Li *et al.*, "Block-based two-dimensional wavelet transform running on graphics processing unit," *IET Computers Digital Techniques*, vol. 8, no. 5, pp. 229–236, Sep. 2014.

[10] T. M. Quan and W.-K. Jeong, "A fast discrete wavelet transform using hybrid parallelism on GPUs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 11, pp. 3088–3100, Nov. 2016.

[11] D. Barina, M. Kula, and P. Zemcik, "Parallel wavelet schemes for images," *J. Real-Time Image Proc.*, 2016, in press.

[12] M. Kula, D. Barina, and P. Zemcik, "New non-separable lifting scheme for images," in *IEEE International Conference on Signal and Image Processing*, 2016, pp. 292–295.

[13] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Comm. on Pure and Applied Mathematics*, vol. 45, no. 5, pp. 485–560, 1992.