A Wavelet Toolbox for Large Scale Image Processing

Geert Uytterhoeven, Dirk Roose, and Adhemar Bultheel

Department of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium,
Geert.Uytterhoeven@cs.kuleuven.ac.be
http://www.cs.kuleuven.ac.be/~geert/

Abstract. The wavelet transform has proven to be a valuable tool for image processing applications, like image compression and noise reduction. In this paper we present a scheme to process very large images that do not fit in the memory of a single computer, based on the software library WAILI (Wavelets with Integer Lifting). Such images are divided into blocks that are processed quasi independently, allowing efficient parallel programming. The blocking is almost completely transparent to the user.

1 Introduction

Wavelet transforms have good decorrelating properties. Wavelets are localized in both the spatial domain and the frequency domain, and are based on a multi-resolution analysis. They can have vanishing moments, which means that a smoothly varying signal can be represented with a small set of basis functions. The combination of this properties makes wavelets successful for image compression.

In application areas like Geographical Information Systems (GIS), image sizes are measured in gigabytes (GB) and even terabytes (TB). For example, an aerial color image of Belgium, where each pixel corresponds to one square meter, merely consumes 90 GB of storage. It is obvious that such images need to be stored and handled in a compressed form. Wavelets can provide for this since wavelet-based techniques offer better compression rates than other techniques. Moreover, many image processing operations can be done in the wavelet domain. Some of them can even be done better in the wavelet domain than in the spatial domain.

We have developed the software library WAILI, which allows the combination of image compression and processing in one package. In this paper we describe the extension of WAILI to block-based processing, which allows for a parallel implementation.

P. Zinterhof, M. Vajteršic, A. Uhl (Eds.): ACPC'99, LNCS 1557, pp. 337–346, 1999.

2 Wavelets and the Lifting Scheme

2.1 Predict and Update

The wavelet transform of a 1D signal is a multi-resolution representation of that signal where the wavelets are the basis functions which at each resolution level give a highly decorrelated representation. At each resolution level, the signal is split into a high pass and a low pass part and the low pass part is split again etc. These high pass and low pass parts are obtained by applying certain wavelet filters.

The lifting scheme is an efficient implementation of these filtering operations. Several introductions to the lifting scheme are available [7, 6]. Suppose that the low resolution part of a signal at level $j+1$ consists of a data set λ_{j+1}. This set is transformed into two other sets at level j: the low resolution part λ_j and the high resolution part γ_j. This is obtained first by just splitting the data set λ_{j+1} into two separate data subsets λ_j and γ_j (e.g. the even samples and the odd samples). Such a splitting is sometimes referred to as the *lazy wavelet transform*. Then these two sets are recombined in several subsequent pairs of lifting steps which decorrelate the two signals.

A *dual lifting* step can be seen as a prediction: the data γ_j are 'predicted' from the data in the subset λ_j. When the signals are still highly correlated, then such a prediction will usually be very good, and thus we do not have to keep this information in both signals. That is why we can keep λ_j and store only the part of γ_j that is not predictable (the prediction error). Thus γ_j is replaced by $\gamma_j - \mathcal{P}(\lambda_j)$ where $\mathcal{P}$ represents the prediction operator. For smooth signals, the prediction error is small. This is the real decorrelating step.

However, the new representation has lost certain basic properties, which one usually wants to keep, like for example the mean value of the signal. To restore this property, one needs a *primal lifting* step, whereby the set λ_j is updated with data computed from the (new) subset γ_j. Thus λ_j is replaced by $\lambda_j + \mathcal{U}(\gamma_j)$ with $\mathcal{U}$ some updating operator.

In general, several such lifting steps can be applied in sequence to go from level $j+1$ to level j. To recapitulate, let us consider a simple lifting scheme with only one pair of lifting steps.

Splitting (*lazy wavelet transform*) Partition the data set λ_{j+1} into two distinct data sets λ_j and γ_j.

Prediction (*dual lifting*) Predict the data in the set γ_j by the data set λ_j.

$$\gamma_j \leftarrow \gamma_j - \mathcal{P}(\lambda_j).$$

Update (*primal lifting*) Update the data in the set λ_j by the data in set γ_j.

$$\lambda_j \leftarrow \lambda_j + \mathcal{U}(\gamma_j).$$

These steps can be repeated by iteration on the λ_j, creating a multi-level transform or multi-resolution decomposition.

The inversion rules are obvious: revert the order of the operations, invert the signs in the lifting steps, and replace the splitting step by a merging step. Thus, inverting the three step procedure above results in:

Inverse update $\lambda_j \leftarrow \lambda_j - \mathcal{U}(\gamma_j)$,

Inverse prediction $\gamma_j \leftarrow \gamma_j + \mathcal{P}(\lambda_j)$,

Merge $\lambda_{j+1} \leftarrow \lambda_j \cup \gamma_j$.

2.2 Integer Transforms

In practice, discrete signals are represented by integers. Doing the filtering operations on these numbers however will transform them in rational or real numbers because the filter coefficients need not be integers. To obtain an efficient implementation of the discrete wavelet transform, it is of great practical importance that the wavelet transform is represented by a set of integers as well, while the transform should still be invertible. This is easily achieved within the lifting framework.

We round the intermediates of each lifting step to integers (for example the nearest integer) and indicate this operation by square braces. Thus, we actually compute rounded values:

$$\gamma_j \leftarrow \gamma_j - [\mathcal{P}(\lambda_j)], \qquad \text{and} \qquad \lambda_j \leftarrow \lambda_j + [\mathcal{U}(\gamma_j)].$$

It is not difficult to verify that each step of the lifting scheme with rounding is perfectly invertible and thus the whole signal is perfectly reconstructible, whatever the rounding rule we use, on condition of course that the rounding is deterministic [1, 8].

2.3 Example: Cohen-Daubechies-Feauveau

The popular family of classical biorthogonal wavelets constructed by Cohen, Daubechies and Feauveau [2] fits in the above scheme. Especially its member with two vanishing moments for both the primal and dual wavelet (hence named CDF $(2, 2)$ wavelet) is widely used.

Thanks to the lifting scheme, the accompanying wavelet transform can be implemented in an efficient way [8]). From the second generation viewpoint, one transform step of a discrete signal $x = \{x_k\}$ looks like:

Splitting Split the signal x (i.e. λ_{j+1}) into even samples (i.e. λ_j) and odd samples (i.e. γ_j):

$$s_i \leftarrow x_{2i}, \qquad \text{and} \qquad d_i \leftarrow x_{2i+1}.$$

Prediction Predict the odd samples using linear interpolation:

$$d_i \leftarrow d_i - \frac{1}{2}(s_i + s_{i+1}).$$

Update Update the even samples to preserve the mean value of the samples:

$$s_i \leftarrow s_i + \frac{1}{4}(d_{i-1} + d_i).$$

As a result, the signal $s = \{s_k\}$ is a coarse representation of the original signal x, while the signal $d = \{d_k\}$ contains the high frequency information that is lost when going from resolution level $j + 1$ to resolution level j.

A whole family of lifting schemes can be constructed in this way, of which the above example is just the simplest possible case. Since lifting steps use coefficients from regions near the coefficients that are updated, one needs special care at the boundaries. This is also important for a parallel implementation.

Note that this transform works on one-dimensional data. For two-dimensional data, like images, it can be applied row and columnwise, resulting in a tensor product transform at each step. Instead of with 2 subbands (low pass and high pass), one will end up with 4 subbands: `LL` — low pass in both the horizontal and vertical directions, `LH` — low pass in the vertical, high pass in the horizontal direction, `HL` and `HH`. When iterated on the LL subband, the result is a multiresolution decomposition as shown in Figure 1. This ordering of the subbands at the different resolution levels is called the "Mallat" ordering [3].

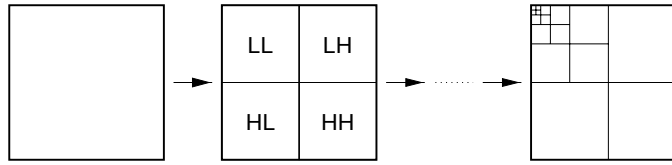

Fig. 1. The two-dimensional wavelet transform: iteration on the `LL` subband, showing the wavelet coefficients in "Mallat" ordering

3 WAILI: Wavelets with Integer Lifting

WAILI is a software library — written in C++ — providing wavelet transforms and wavelet-based operations on two-dimensional images [8]. WAILI is available in source form[1] for research purposes.

WAILI implements various integer wavelet transforms, using lifting technology. It includes some image processing operations, such as:

- Crop and merge on wavelet transformed images. This allows to cut a rectangular subimage out of a large wavelet transformed image, or to replace a rectangular area in a large wavelet transformed image.
- Noise reduction, using thresholding based on generalized cross validation.
- Simple compression, using thresholding of subbands.

[1] WWW: `http://www.cs.kuleuven.ac.be/~wavelets/`

4 Very Large Images and Tiling

We define an image to be *very large* if it does not fit in the memory of the computer system that is processing it. Note that by this definition, whether an image is very large or not depends not only on the image but also on the computer system.

4.1 Tiling

In order to process very large images, they have to be divided in blocks (tiling).

Fortunately wavelets are localized in both the spatial and frequency domains. For the division in blocks only the spatial domain is important, and this means that one needs only a limited subset of the input image to calculate a subset of the wavelet coefficients. The same is true for the inverse transform. From Figure 2, it is clear that only a fraction of the blocks in the right part of the figure are needed to reconstruct the rectangle in the left part.

Since not all blocks have to be present in main memory at the same time, this allows for the successful processing of very large images on small uni-processor machines with limited memory. Moreover, the processing of the blocks can be distributed among multiple machines and/or CPUs.

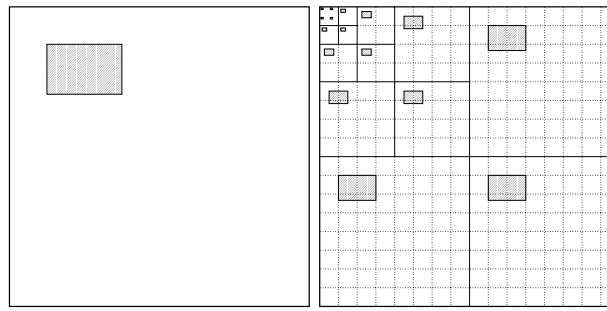

Fig. 2. Locality in the (a) *space* and (b) *wavelet domain*. The dotted lines in (b) indicate block boundaries

Within WAILI, each subband is divided separately in blocks. Blocks are square and block sizes must be powers of two (except near the borders of the image). Because small blocks cause more administrative overhead, it is recommended to use the same block size at different resolution levels.

Since we want to process both complete images and subimages that are part of a large image, the upper left corner of a (sub)image is not always located at the origin $(0, 0)$, but may have arbitrary coordinates (see Figure 3). To facilitate the wavelet transforms, the internal boundaries of the blocks must always be aligned to the block size. Hence, not only the blocks at the right and bottom

borders of the image may be rectangular and smaller than the block size, but also at the left and upper borders.

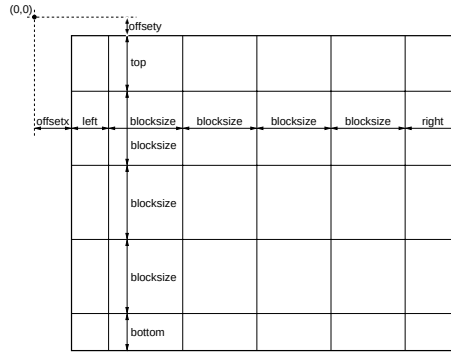

Fig. 3. Division of a subband in blocks. The margins (`top`, `bottom`, `left` and `right`) are chosen such that the coordinates of the block boundaries are always multiples of the block size

The whole wavelet decomposition is divided in blocks like shown in fig. 4. Within each channel (color images consist of multiple channels), the tree forms a "Mallat" ordering (cfr. Figure 1). A block within an image is thus uniquely defined by 5 indices: *channel* , resolution *level*, *subband*, *column* and *row*.

To implement wavelet transforms on tiled images, one has to consider the following steps:

Splitting step: To split a block in "even" and "odd" coefficients, in both the horizontal and vertical directions, one needs that block only. So this step is trivial to parallelize.

Lifting steps: To calculate the lifting operation on a border coefficient, some coefficients from an adjacent block are needed. Thus in a parallel implementation there must be communication of border coefficients (exchange of "overlap regions", with a width depending on the wavelet transform). Since in a primal lifting step the "even" blocks (i.e. the blocks with the even coefficients) are updated by the "odd" blocks, and vice versa in a dual lifting step, these updates can be done independently in each block, except for the communication just mentioned. Hence, this step can be parallelized easily. The parallel efficiency grows with increasing block size due to the "perimeter effect".

4.2 Block Management

We extended WAILI to support very large images that are divided in blocks. This is nearly completely transparent to the user: all traditional operations can be performed, just like on small non-blocked images.

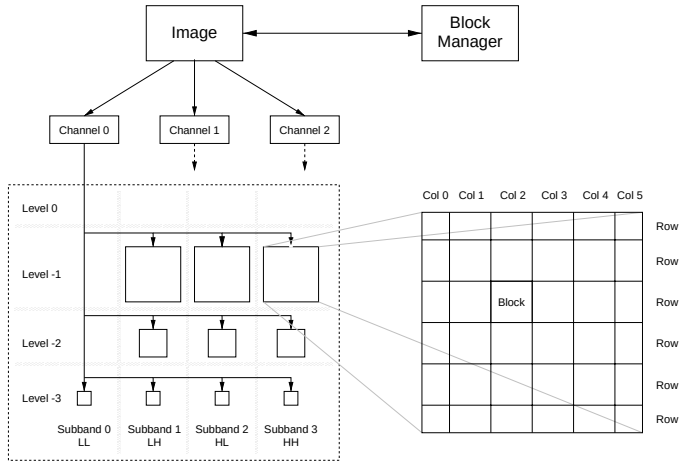

Fig. 4. Subdivision of an *image* into *channels*, *levels*, *subbands* and *blocks*

The "block manager" is responsible for managing the blocks of which a very large image is composed. To access a part of the (transformed) image, the block manager is asked for blocks and retrieves them, e.g. from disk or via a network. If a block is modified, it has to be updated on the storage device. Blocks are passed to the compression module to remove redundancy among the wavelet coefficients within each block. The block manager for sequential processing also keeps a cache of recently used blocks to speed up processing. An overview of the complete system is given in Figure 5.

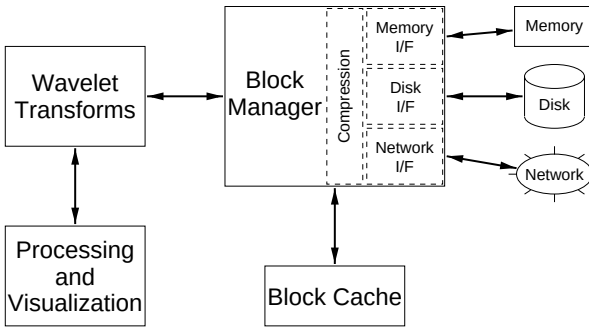

Fig. 5. Design of the large scale image processing system

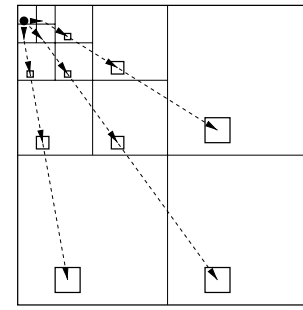

Fig. 6. Zerotree encoding

To obtain a parallel version, we only need to adapt the block manager and introduce communication of the border coefficients during the lifting steps.

4.3 Block-Based Denoising of Images

WAILI uses soft-thresholding to reduce additive, stationary noise. The threshold is selected at each transform level and for each subband separately, using generalized cross validation [8]. Since this is an asymptotical method, it only performs well for sufficiently large subbands.

For wavelet transformed images that are divided into blocks, the denoising technique is applied independently on each block of each subband. Because only the highpass subbands are processed and blocks belonging to lowpass subbands are left alone, it does not become visible where the borders between the blocks are located.

Experiments showed that one gets better results, compared to the case where complete subbands are denoised, because now the threshold is more adapted to the local properties of the image. Thus denoising is not only easily parallelizable, it also yields a better image quality.

5 Image Compression

Due to their decorrelating properties, wavelets are well suited for the compression of images and signals. If a sufficiently smooth image is transformed, most high pass coefficients will be zero or very small. Compression algorithms based on wavelets often use a variant of "zerotree encoding" [5, 4]. Most compression methods are targeted at "small" images. In that case the individual subbands are too small to get a sufficient compression gain from compressing them separately, hence zerotree encoding combines the subbands to exploit the correlation that is still present among spatially related coefficients at different resolution levels. (see Figure 6).

For very large images, each zerotree contains coefficients from a multitude of blocks. Thus zerotree encoding will cause a block access pattern that is not feasible and the application of zerotree encoding is limited. A possible solution is to use a two-level encoding strategy:

1. For the lowest resolution levels, zerotree encoding can still be used because the subbands at these levels consist of only one block.
2. For higher resolution levels, a block-based encoding can be used. At these levels the subbands contain several blocks. If the block size is sufficiently large, each block will contain enough coefficients to be well compressible. Since all blocks can be compressed independently, this step is trivially parallelizable.

6 Results

We have no parallel implementation of WAILI yet. To get an impression of the performance of our wavelet transform code, we transformed an image of 3584×3584 pixels, using a block size of 128×128 pixels, for a total of 28×28 blocks. We used the Cohen-Daubechies-Feauveau (2, 2) biorthogonal wavelets.

All calculations were done on 16 bit integer data on a single processor machine (Intel Pentium II, 350 MHz) with sufficient memory to avoid paging. Note that after 2 transform levels, the subband size is no longer divisible by the blocksize and one has to take care of rectangular blocks.

Each level of the CDF (2, 2) wavelet transform of n blocks consists of 5 steps:

- Lazy wavelet transform in both the horizontal and vertical directions. In order to maintain the blocksize, 4 blocks are jointly transformed into 4 new blocks. (in total n blocks are transformed)
- Lifting steps on the columns of the image: a dual lifting step ($n/2$ blocks) and a primal lifting step ($n/2$ blocks).
- Lifting steps on the rows of the image: a dual lifting step ($n/2$ blocks) and a primal lifting step ($n/2$ blocks).

Timings are shown in Table 1. An additional overhead of about 0.6 seconds was caused by the supply of blocks by the block manager. However, in a parallel implementation we can assume that the blocks are distributed over all processors, so this overhead can be spread over the processors too.

In a parallel implementation, the lazy wavelet transform requires no communication if the 4 neighboring blocks are transformed by the same processor. If we would use $14 \times 14 = 196$ CPUs, the lazy wavelet transform for the first level could be calculated in 1.33 ms and the calculation time for the 4 lifting steps would be 28.5 ms. The lifting steps require communication of the border coefficients. For the CDF (2, 2) transform, in each lifting step 2 strips of 128 coefficients must be exchanged between processors. On current parallel systems (e.g. IBM SP2), the communication cost for such short messages is dominated by the startup time ($\mathcal{O}(50\ \mu s)$), which is negligible compared to the calculation cost.

However, the amount of data to be transformed is divided by 4 at each level. Hence, at the second level, we can keep the blocksize equal to 128, but then only 1/4 of the CPUs can be kept busy. Thus the execution time for the second level will also be approx. 30 ms. A similar reasoning holds for the other levels, but in our example rectangular blocks will appear at level 3, causing an additional overhead.

We can conclude that a parallel implementation can be efficient if the number of transform steps is small, and the number of processors is not too large, relative to the image size.

7 Conclusion

In this paper, we have described a strategy to perform wavelet-based image compression and processing on very large images. The images are split in blocks, which can be handled nearly independently.

The (integer) wavelet transform is implemented using the lifting scheme, which can be parallelized easily and efficiently, since only communication of pixels at the borders of the blocks is required.

Transform Level	Lazy Transform # Calls	Lazy Transform Time	Lifting Steps # Calls	Lifting Steps Time	Total Time	Cumulative Total Time
1	784	0.26	1568	5.59	5.85	5.85
2	196	0.02	392	1.34	1.22	7.07
3	49	0.01	128	0.33	0.34	7.41

Table 1. Timings for the wavelet transform of an image of size 3584×3584. Each call corresponds to the processing of one elementary block. All timings are in seconds

The method is implemented as an extension to the software library WAILI. Blocks are accessed and handled under supervision of a block manager. Although the software has not been parallelized yet, the limited communication and the existing block manager allow an easy and efficient parallelization.

The division in blocks also has a positive influence on the performance of our denoising algorithm. Because denoising works independently on the blocks, it adapts better to the local properties of the image.

8 Acknowledgements

This paper presents research results of the Flemish Information Technology Action Program ("Vlaams Actieprogramma Informatietechnologie"), project number ITA/950244. The scientific responsibility rests with its authors.

References

[1] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. *Appl. Comput. Harmon. Anal.*, 5(3):332–369, 1998.

[2] A. Cohen, I. Daubechies, and J. Feauveau. Bi-orthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 45:485–560, 1992.

[3] S. G. Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Trans. Acoust. Speech Signal Process.*, 37(12):2091–2110, 1989.

[4] A. Said and W. A. Pearlman. Image compression using the spatial-orientation tree. In *Proc. IEEE Int. Symp. Circuits and Syst., Chicago, IL*, pages 279–282, May 1993.

[5] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. 41(12):3445–3462, 1993. IEEE Trans. Signal Process.

[6] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A. F. Laine and M. Unser, editors, *Wavelet Applications in Signal and Image Processing III*, pages 68–79. Proc. SPIE 2569, 1995.

[7] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2), 1997.

[8] G. Uytterhoeven, F. Van Wulpen, M. Jansen, D. Roose, and A. Bultheel. WAILI: A software library for image processing using integer wavelet transforms. In K.M. Hanson, editor, *Medical Imaging 1998: Image Processing*, volume 3338 of *SPIE Proceedings*, pages 1490–1501. The International Society for Optical Engineering, February 1998.