World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:2, No:9, 2008

# A Parallel Quadtree Approach for Image Compression using Wavelets

Hamed Vahdat Nejad, and Hossein Deldari

***Abstract*—**Wavelet transforms are multiresolution decompositions that can be used to analyze signals and images. Image compression is one of major applications of wavelet transforms in image processing. It is considered as one of the most powerful methods that provides a high compression ratio. However, its implementation is very time-consuming. At the other hand, parallel computing technologies are an efficient method for image compression using wavelets. In this paper, we propose a parallel wavelet compression algorithm based on quadtrees. We implement the algorithm using MatlabMPI (a parallel, message passing version of Matlab), and compute its isoefficiency function, and show that it is scalable. Our experimental results confirm the efficiency of the algorithm also.

***Keywords*—**Image compression, MPI, Parallel computing, Wavelets.



## I. Introduction

WAVELETS are the foundation of a powerful new approach to signal processing and analysis, called multiresolution theory [3], [6]. Multiresolution theory incorporates and unifies techniques from a variety of disciplines, including subband coding from signal processing, quadrature mirror filtering from digital speech recognition, and pyramid image processing. As its name implies, multiresolution theory is concerned with the representation and analysis of signals (or images) at more than one resolution. The appeal of such an approach is obvious-features that might go undetected at one resolution may be easy to spot at another.

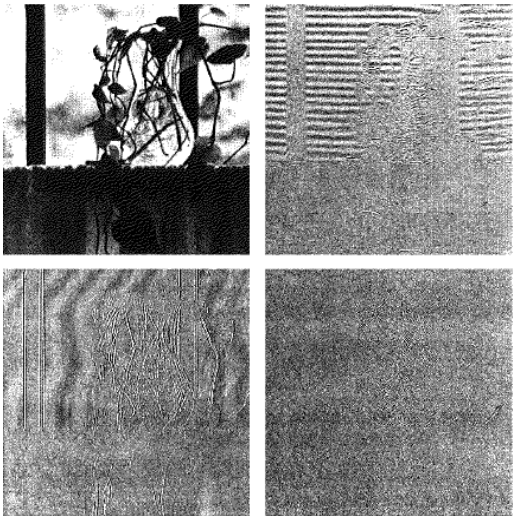
In two dimensional signals (images), at each level of wavelet decomposition, four quarter-size output images called approximation, Horizontal detail, vertical detail, and diagonal detail are produced. Fig. 1 shows an image and its wavelet decomposition. Beginning with the upper-left corner and proceeding in a clockwise manner, the subimages are approximation, horizontal detail, diagonal detail, and vertical detail. By decomposing each of the new quaternary images, we obtain the second level decomposition of the original image. This process can be repeated with respect to the application specific criteria. Fig. 2 shows a full three-level wavelet decomposition of a fingerprint image. It consists of 64 subimages. In wavelet processing after decomposition, the new subimages are processed instead of the original image.

Some new applications including image denoising and compression have been issued in wavelet processing [3]. The major problem of such applications is the long running time when they executed on one processor [4], [5]. Since wavelet decomposition is naturally adaptable to quadtrees, we use a quadtree structure for parallelizing image compression.

We implement the application using message passing model. In the world of parallel computing, the message passing interface (MPI)[7] is the de facto standard for implementing programs on multiple processors. MPI defines C and FORTRAN language functions for doing point to point communication in a parallel program. MPI has proven to be an efficient model for implementing parallel programs and is used by many of the world's most demanding applications. MatlabMPI [1] is set of Matlab scripts that implement a subset of MPI and allow any Matlab program to be run on a parallel computer. The key innovation of MatlabMPI is that it implements the widely used MPI on top of standard Matlab file I/O, resulting in a pure Matlab implementation that is exceedingly small. Thus MatlabMPI will run on any combination of computers that Matlab supports. We use MatlabMPI to implement the proposed algorithm.

The rest of this paper is organized as follows. In section 2 we introduce serial compression algorithm. Section 3 describes the proposed parallel algorithm. We evaluate the algorithm in Section 4. Section 5 concludes.

Authors are with Ferdowsi University of Mashad, Computer Department, Iran (e-mail: Hamed.vh@gmail.com, hdeldari@yahoo.com).



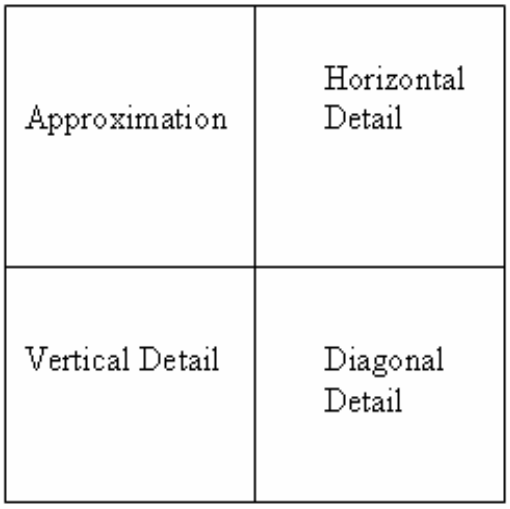



Fig. 1 Wavelet decomposition in 4 blocks

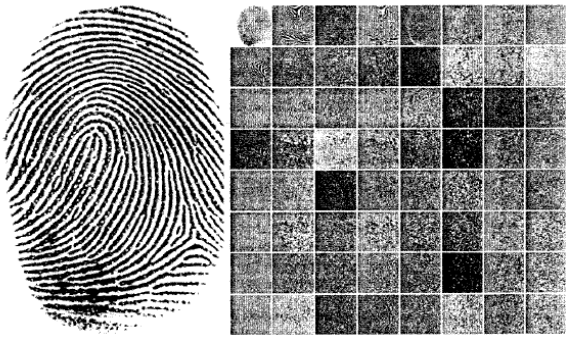


Fig. 2 Three scale, full wavelet packet decomposition

## II. Image Compression using Wavelets

The general wavelet–based procedure for image compression is as follows [2], [3]:

1. Decomposition: Choose a wavelet (e.g. Haar, Symlet…) and number of levels L for the decomposition.
2. Thresholding: Compare the wavelet coefficients of the image with ε and set them to zero if they are lower than it.
3. Quantization
4. Encoding

In the thresholding step, a threshold ε of the compression is fixed. Then wavelet coefficients of the image are compared with ε: they are set to zero, if they are lower than it.

In quantization step, compression is achieved by matching an input sequence of data samples with the entries (i.e. the codewords) in a pre-classified database known as codebook or "dictionary". The dictionary contains the sourse symbols to be coded. Clearly, the size of the dictionary is an important system parameter. If it is too small, the detection of matching gray-level sequences will be less likely; If it is too large, the size of the code words will adversely affect compression performance, and it needs much time to detect a code in the dictionary. The large computational load at the encoding process is the main problem that prevents the idea to be applied in real-time image compression systems. Many research works have been carried out to reduce the computational complexity of the quantization process. The kind of works is to limit the number of code words to be searched in the codebook.

In the final stage, symbol encoder creates a fixed or variable-length code to represent the quantizer output, and maps the output in accordance with the code. In most cases, a variable-length code is used to represent the quantized dataset. It assigns the shortest code words to the most frequently occurring output values and thus reduces coding redundancy. Huffman, arithmetic, and bit-plane coding [3] are well-known coding approaches.

## III. Parallel Compression Algorithm

Let number of processors is $p=4^i$, where i is a positive integer equal to the number of wavelet decomposition levels. We assume that initially each processor has a wavelet approximation or detail of level i. The distribution is based on the structure of quadtrees, in which any four subimages of one wavelet decomposition level are distributed among four adjacent processors. Fig. 3 shows the distribution of a 2 level decomposed image between 16 processors. The number written in each subimage indicates the rank of the processor which contains the corresponding approximation or detail. The proposed algorithm is as follows:

| | | | |
|---|---|---|---|
| 0 | 1 | 4 | 5 |
| 2 | 3 | 6 | 7 |
| 8 | 9 | 12 | 13 |
| 10 | 11 | 14 | 15 |

Fig. 3 Distribution of blocks to different processors

- Each processor except master (processor with rank 0) applies a threshold to its subimage (Processor 0 contains the approximation of the image, and the threshold is applied to detail coefficients only). So many of detail coefficients become zero.
- Each processor quantifies its subimage with the adequate codebook.
- Encode each result using an encoder (e.g. Huffman, runlength, arithmetic…)
- Each processor sends its coded subimage to the base processor. The base processor is the processor with rank int(i/4)*4, where i is the rank of the current processor. When a base processor receives three encoded images of its three siblings, it reconstructs a larger subimage, and sends it to the second level base processor which is computed by int(i/16)*16. This operation continues until processor 0 receives the whole coded image. The following semi-code shows these steps:

```
Procedure CMP-Qtree ()
{
If (my_rank≠0)   Threshold ();
Quantify ();
Encode ();
For (i=1; i=Log4P; i++)
{
        if ((my_rank mod 4^(i-1)) =0 )
        && ((my_rank mod 4^i) ≠0 )
            Send coded image to
            int (my_rank/4^i)* 4^i
        }
        if ((my_rank mod) 4^i =0)
   {
       Receive three coded images
       Merge four coded images
   }
}
}
```

## IV. Experiments

### A. Evaluation

For the sake of simplicity, we assume that the original image is a square matrix of order $2^n \times 2^n$, and the number of processors equals to $4^i$, where i is the number of wavelet decomposition levels. Hence each processor has one image of size $2^{n-i} \times 2^{n-i}$. First all processors execute thresholding (except master), vector quantifying and encoding. The time complexity of thresholding and arithmetic coding is $o(2^{n-i} \times 2^{n-i})$. If codebook contains L words, quantifying is done in $o(2^{n-i} \times 2^{n-i} \times L)$. The communication step consists of $\log_4 P$ all to one reduction. Four processors participate in each reduction so it takes $(t_s + t_w m)\log 4$ time, where $m = 2^{n-i} \times 2^{n-i}$ is the size of message, $t_s$ is the startup time required to handle a message at the sending process, and $t_w$ is the per-word transfer time . Therefore communication time is:

$$T_{comm} = \sum_{i=1}^{\log_4 P} (t_s + t_w 4^{n-i}))\log 4$$

$$= \log_4^p t_s \log 4 + \log 4 t_w \frac{4^n - 4^{n-\log_4^p}}{3}$$

The parallel time is computed as the sum of communication and computation time, hence

$$T_p = 4^{n-i} \times L + \log_4^p t_s \log 4 + \log 4 t_w \frac{4^n - 4^{n-\log_4^p}}{3}$$

Now we can compute the overhead time as follow:

$$T_o = PT_p - W =$$

$$= p \log_4^p t_s \log 4 + p \log 4 t_w \frac{4^n - 4^{n-\log_4^p}}{3}$$

$$= k_1 t_s p \log_4^p + k_2 t_w p 4^n - k_3 t_w 4^n \qquad (1)$$

The central relation that determines the isoefficiency function of a parallel program is $W = kT_o$, where $k = E/(1-E)$ and E is the desired isoefficiency. Rewriting this equation by substituting (1) for $T_o$, we have:

$$W = kk_1 t_s p \log_4^p + kk_2 t_w p 4^n - kk_3 t_w 4^n \qquad (2)$$

Rewriting this relation first with only the $t_s$ term of $T_o$:

$$W = (kk_1 t_s) P \log_4^P \qquad (3)$$

This equation gives the isoefficiency function (plogp) with respect to message startup time. Similarly for the second term of (2), we have

$$W = kk_2 t_w P 4^n$$

$$\Rightarrow 4^n \times L = kk_2 t_w P 4^n$$
$$\Rightarrow L = kk_2 t_w P$$

Suppose $2^n$=L which means that, the number of codes is equal to number of pixels in one dimension. Therefore,

$$2^n = kk_2 t_w P$$
$$\Rightarrow W = (2^n)^3 = (kk_2 t_w)^3 P^3 \quad (4)$$

Similarly we have from third term of (1),

$$W = -kk_3 t_w 4^n$$
$$\Rightarrow 4^n \times L = (-kk_3 t_w) 4^n$$
$$\Rightarrow L = (-kk_3 t_w)$$
$$\Rightarrow W = (-kk_3 t_w)^3 \quad (5)$$

By comparing (3), (4), and (5), the overall asymptotic isoefficiency function is $\theta(p^3)$.

### *B. Implementation*

We test the parallel program for various numbers of processors. For this, we exploit an 1800×2115 image (Fig. 4) and a dictionary of 1024 codes. Table I shows the results in the sense of speedup. The Efficiency diagram of the algorithm for various numbers of processors is shown in Fig. 5.

## V. Conclusion

Parallel technology is an efficient method for image compression, because the encoding needs fast processing. In this paper, we have presented a parallel algorithm for image compression based on wavelet transformation. The algorithm is data parallel, and has been implemented with MPI directives. We have shown the feasibility of the parallel algorithm by computing isoefficiency function. We have also tested the algorithm for various numbers of processors. Results show the effectiveness of the parallel algorithm.

TABLE I
SPEEDUP

| Number of Processors | Speedup |
|---|---|
| 4 | 3.2 |
| 16 | 12.2 |
| 64 | 44.7 |

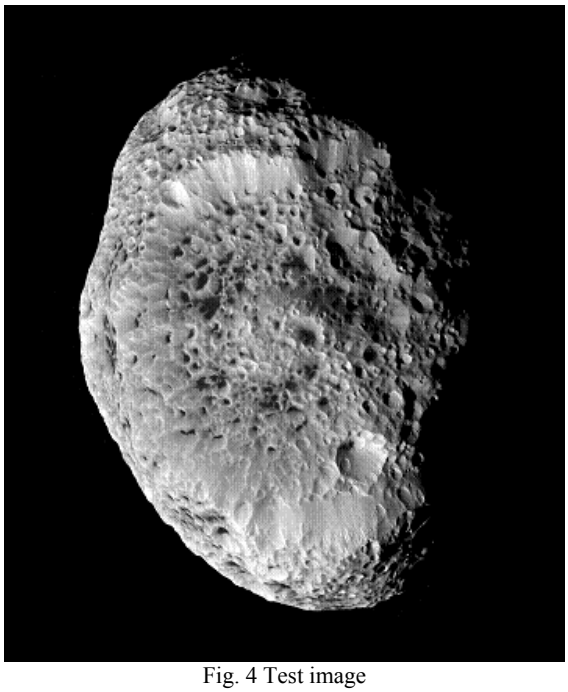


Fig. 4 Test image

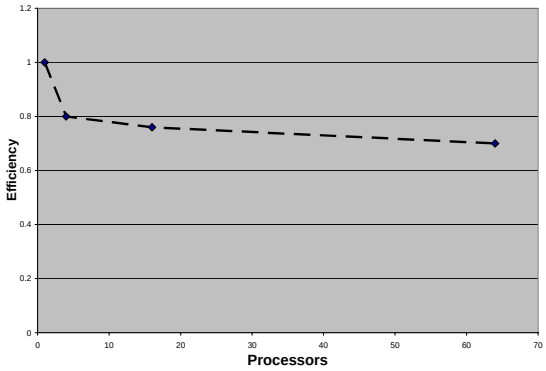



Fig. 5 Efficiency vs. Processors

## References

[1] Jeremy Kepner, "Parallel programming with MatlabMPI", 2002, High Performance Embedded Computing (HPEC) workshop, MIT Lincoln Laboratory, Lexington, MA, http://arXiv.org/abs/astro-ph/0107406.

[2] P. Moravie, H. Essafi, C. Lambert-nebout, and J-L. Basille, "Real-time image compression using SIMD architectures", In Proceedings of Computer Architectures for Machine Perception, 1995.

[3] Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing", Addison-Wesley Publishing Company.

[4] ***S***. Khanfir, M. Jemni ,and E. Ben Braiek, "Parallelization of an image compression and decompression algorithm based on 1D wavelet transformation", In Proceedings of First International Symposium on Control, communications and Signal Processing, 2004.

[5] Shi-xin Sun, Chao-yang Pang, Wen-yu Chen, "A new parallel architecture for image compression", In Proceedings of CSCW in Design, 2002.

[6] Yansun Xu , John B. Weaver, Dennis M. Healy, and Jian Lu, "Wavelet transform domain filters: A spatially selective noise filtration technique", In Proceedings of IEEE Tansactions on image processing, Vol. 3, No. 6, November 1994.

[7] "Message Passing Interface" (MPI), http://www.mpiforum.org