Real-Time Image Compression Using Data-Parallelism

P. MORAVIE[1], H. ESSAFI[1], C. LAMBERT-NEBOUT[2], J-L. BASILLE[3]

[1] LETI (CEA – Technologies Avancées), DEIN – CE/S, F91191 Gif sur Yvette.
[2] Centre Spatial de Toulouse, 18 Av. Belin BP 1421, F31055 Toulouse Cedex.
[3] ENSEEIHT, 2 rue camichel, 31071 Toulouse Cedex.

Abstract

The purpose of this paper is to present this new parallel image compression algorithm. We present implementation results on several parallel computers. We also examine load balancing and data mapping problems. We end by presenting a well-suited architecture for Real-Time image compression.
Keywords : *Data-Parallelism, Image Compression, Wavelet Transform, Vector Quantization, Huffman Coding.*

1 Introduction

Today, in the digitized satellite image domain, the needs for high dimension images increase considerably. To transmit or to stock such images (6000 by 6000 pixels), we need to reduce their data volume and so we have to use image compression technics. In most cases, these operations have to be processed in Real–Time. But the large amount of computations required prohibits the use of common sequential processors. To solve this problem, CEA in collaboration with CNES developed and evaluated a new parallel image compression algorithm for general purpose parallel computers using data-parallelism. This paper introduces this new parallel image compression algorithm. Thus, in a first section, we briefly describe the image compression technics on which our algorithm is based. In section two, we develop our algorithm. We also present implementation results on several parallel computers and we examine load balancing and data mapping problems. As a conclusion, we present a well-suited architecture for Real-Time image compression.

2 Sequential Image Compression Algorithms

Image compression is classically achieved in three steps. The image is first transformed in a set of decorrelated coefficients. Then, the transformed coefficients are quantized. Finally, the quantized values are entropy coded.
During the past few years several design algorithms have been developed for each step. It has been shown that compression algorithms based on Wavelet Transform[1], Vector Quantization[3] and Huffman Coding[4] provide one of the best trade-off between compression rates and quality. Therefore, we based our algorithm on all these technics.

2.1 Wavelet Transform (W.T)

The Wavelet Transform is a powerful tool for signal, image processing. For image compression, using wavelet transform offers two essential advantages. First, the produced coefficients are well decorrelated due to the good localization of the wavelet function in both space and frequency. And the multi-resolution of wavelet transform is suitable with quantization: each sub-image produced by W.T can be quantized using an adapted quantizer.
In order to reduce the computation time, we used one of the fastest wavelet transform algorithm (designed by Mallat[2]). It is based on a subband coding scheme, i.e it uses linear filter convolutions and image shrinking.

2.2 Vector Quantization (V.Q)

Quantization is the most essential part of common image compression algorithms. In fact, it is the part which provides the major bits rate reduction. Indeed, this technic reduce the number of grey levels used to code the image and so reduce the number of bits required to represent each pixels. According to Shannon's rate distortion theory, better results are always obtained when vectors rather than scalars are encoded. Therefore, our algorithm is based on vector quantization.
This technic is achieved in three steps. First, the pixels are organized into k-dimensional vectors. Then, each vector is approximated by a vector (named centroid) belonging to a predefined catalogue (named codebook). And finally, each vector to be coded is replaced by the reference of its centroid.

2.3 Huffman Coding (H.C)

Huffman coding[4] is the most popular lossless compression technique. It is a statistical data compression technique which gives a reduction in the average code length used to represent the symbols of a alphabet. In fact, it assigns codes to input symbols in such a way that each code length in bits is approximatively $\log_2(\textit{symbol probability})$.
Usually, a Huffman code is built as follow. First, we rank all symbols in order of probability of occurrence. Next, we successively combine the two symbols of the lowest probability to form a new composite symbol. And finally, we trace a path to each leaf, noticing the direction at each node. The code assign to each symbol is the path.

3 Our Parallel Image Compression Algorithm

3.1 Parallelization mode

After studying implementations of all the technics described above, on MIMD, SIMD and message passing architecture, we concluded that using data-parallelism and pipeline technics are the best way to parallelize this type of algorithm. This

is due to the regularity of the processing involved. Indeed, each block of pixels is processed in the same way and all these have to be chained in good order. As our aim is to define a parallel algorithm for general purpose machine, we developed our parallel image compression algorithm solely using data-parallelism.

3.2 Data Organization

From implementations of these technics on several parallel machines (such as Connection–Machine CM5, Maspar, Symphonie, Sympati2) we noted that for each network and memory organizations, there are few data organizations which allow a real speed–up. This is due to the large amount of regular communications required by all these technics. In fact, these data organizations respect two essential conditions.

- Each processor must have the same amount of data in order to ensure correct load–balancing
- The amount of communications required by the processes must be minimum. It means:
  - *(V.Q)* Each vector must be stocked in one processor memory.
  - *(W.T)* Each processor only gets adjacent pixels in its memory.

An interesting point we underlined is that if a data organization is optimal for Wavelet Transform, then it is also optimal for the quantization. As the wavelet transform do not change data organization, we recommend to optimize the data organization according to wavelet transform and to architecture.

3.3 The Parallel Image Compression Algorithm

The parallel algorithm can be described as follow :

- Get an new image and map the pixels into the processors memories.
- Apply a double Wavelet Transform Decomposition on the image.
- Quantify each sub-image (vector) with the adequate codebook.
- Encode each result using a Huffman encoder.

Vector quantization and Huffman encoding consist in replacing a group of pixels (vector or reference) by a code. By using an appropriate codebook it is possible to mix this 2 process and so to speed-up the computations.

3.4 Implementing results

Each part of this algorithm has been evaluated on several parallel machine with several parameters. In table 1, we give few results for a 8:1 compression ratio and a quality of 36 DB (PSNR).

	SYMPATI–2 (32 proc.)	SYMPHONIE (32 proc.)	CM5 (32 proc.)	Cm200 (4096 proc.)
W.T (Float. point)	1502	200	250	1855
W.T (16 bits)	115	7	-	1870
V.Q + H.C (16 bits)	190	22	-	-

Table 1. *Computation time in Milliseconds (256x256 image).*

We note that when the computations is process using floating or fix point precision operations, the best results are obtained by Symphonie. This is due to its well suited architecture. In fact, Symphonie do not have floating point processors but its architecture has been designed for Real-Time Image Processing.

4 Real-Time image compression architecture

To process the satellite image compression in real–time, we need to execute our algorithm in less than 4 milliseconds. Using fix point precision, Symphonie fitted out with 32 PE processes our algorithm in 29 milliseconds. But, using floating point precision, it processes in 420 millisecond. As quality test shows that in most of cases using 16 bits fix precision to do all the calculations, we can estimate that Symphonie architecture (fitted out with 1024 PE) is well suited for process real–time image compression.
From studying all the implementations we have made, we pointed out the main characteristics of Symphonie (*in italic*) which are interesting for satellite image compression :

- Symphonie is a *Multi-SIMD* architecture where each node is a linear array of superscalar processors. Each PE consists mainly of *32 bits processor, a co-processor dedicated to memory address computation, a floating point accelerator* and a communication module. Symphonie is also fitting out with two communications networks (one asynchronous and one synchronous).

5 Conclusion

We presented here a new parallel image compression algorithm which likely to be implemented on SIMD and MIMD architecture. We evaluated this algorithm on several machines. We showed that when the computation is done using fix point precision, it allows us to process the image compression in real time.

References

1. M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies. Image coding Using Wavelets Transform. *IEEE Trans. on Image processing* 1(4):205–220, 1992.
2. S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. on Pattern Analysis and Machine Inte lligence*, 11(7):674–693, 1989.
3. A. Gersho, R.M. Gray Vector Quantization and Signal Compression. Kluwer Academic Publisher, Boston, 1992.
4. M. Nelson. The Data Compression book. Prentice Hall, Redwood Cityes, 1991.
5. M. J. Quinn. *Designing Efficient Algorithms for Parallel Computers.* McGraw-Hill, New York, NY, 1987.
6. R. W. Hockney and C. R. Jesshope. *Parallel Computers: 2 Architecture, Programming, and Algorithms, 2nd Ed.* IOP Publishing Ltd., Pennsylvania, 1988.