Interactive Optimization of Video Compression Algorithms on the CRAY T3D

H. Nicolas, CRAY Research, Signal Processing Laboratory and
F. Jordan, EPFL, 1015 Lausanne, Switzerland

ABSTRACT: *This paper presents a new technique which allows interactive optimization of video compression algorithms using the CRAY T3D. This work aims to exploit as much as possible the parallel nature of digital image processing algorithms to obtain almost real-time computing with the flexibility of a software implementation. Thanks to this low computation time, interactive tools have been developed which allow easy and fast visual evaluation of image quality. This leads to significant productivity gain when developing new video compression techniques. Our approach has been validated on advanced region-based video compression algorithms. The interactive facilities offered by the proposed technique permit the accurate optimization of the algorithm parameters in few minutes, where several days were previously needed. Depending on the complexity of the compression algorithms, 8-12 images are compressed, decompressed and visualized per second.*

1 Introduction

One of the main difficulties for researchers when developing new digital image processing algorithms is related to the computation time required for the scientific evaluation of the algorithms, and the optimization of their parameters. As a consequence, in practice, researchers have often not enough time to optimize and to evaluate their algorithms correctly.

Due to the parallel nature of almost all image processing techniques, massively parallel computing is an attractive solution to significantly reduce the computation time. In this context, some authors have already proposed efficient parallel implementations of various video compression algorithms [1][2][3][4][5].

Unfortunately, even with such parallel implementations, the scientific evaluation of the results remains difficult due to the fact that there is no mathematical criteria to correctly evaluate the visual quality of an image. Such criteria are classically used to compare original images with their decompressed versions, but they are not sufficiently reliable. For this reason, in the undergoing development of the MPEG-4 video standard, the quality of the results is judged only visually [6]. Furthermore, image processing algorithms often have several interdependent parameters which must be optimized simultaneously. This kind of problem remains unsolved even with existing parallel approaches.

In order to overcome these problems, we present in this paper an interactive system, called *DirectView*, for real-time video processing and real broadcasting system simulations. This system is based on efficient parallel implementations of video compression algorithms on a massively parallel computer such as the CRAY T3D [7][4] with the modularity and the flexibility of a software implementation in high level languages (C, C + +). Furthermore, *DirectView* permits the visual evaluation of the processed image sequences as they are produced by an algorithm running on the parallel computer. It also allows the modification of the parameters of an algorithm in an interactive manner, thus permitting fast and efficient evaluation and optimization. The possibility of performing fast and accurate optimization is of particular importance when considering that the algorithms used in the MPEG-2 video standards were retained mainly because they were fully optimized. It should be pointed out that real-time processing can also be obtained with dedicated hardware or with DSPS, but these kinds of implementations do not offer the flexibility given by the presented system [7][8].

The outline of the paper is the following: The main characteristics of the system are presented in Section 2. Section 3 describes the parallel approach which has been used. The video compression algorithm which is implemented in *DirectView* is presented in Section 4. Experimental results are provided in

Section 5, while Section 6 draws the conclusions and perspectives of this work.

2 General description of the system

2.1 Main features of the CRAY T3D

DirectView has been implemented on a CRAY T3D. The T3D is a MIMD parallel supercomputer with a scalable architecture which allows the use of a number of processors equal to any power of two (i.e., 1, 2, 4, 8... processors). Each alpha processor (150 MHz) of the T3D has its own local memory of 64 Mbytes. There is no shared memory. Since the T3D requires a CRAY-YMP computer as front-end, all Input/Output operations are done via the YMP.

It should be pointed out that the methodology proposed in this paper does not depend directly on the characteristics of the T3D, but the T3D has been chosen because it is able to provide the computational power necessary to compress 8 and more black & White Common Intermediate Format images (CIF images, 352x288 pixels, 8bits/pixel, digitized at 25 frames/second) per second.

2.2 General principle

Two versions of the proposed system have been developed. The first version (*DirectView* 1.0) uses the T3D for the compression and the decompression of the video data. A workstation is used only for real-time visualization of the decompressed images. The second version (*DirectView* 2.0) is more ambitious since it simulates a complete real-time video broadcasting system. This is done by exploiting the fact that for almost all video compression algorithms, the compression is significantly more complex than the decompression in term of computational load. This characteristic allows us to decompress one image on the workstation while the parallel computer compresses the following one. As a consequence, only the compressed data must be transmitted to the workstation which is used as a decoder. Figure 1 and Table 1 give the block diagram and a synthetic comparison of each of these versions.

With the first version, a higher transmission bandwidth is necessary (8 Mbit/s with a frame rate of 10 images/s and for

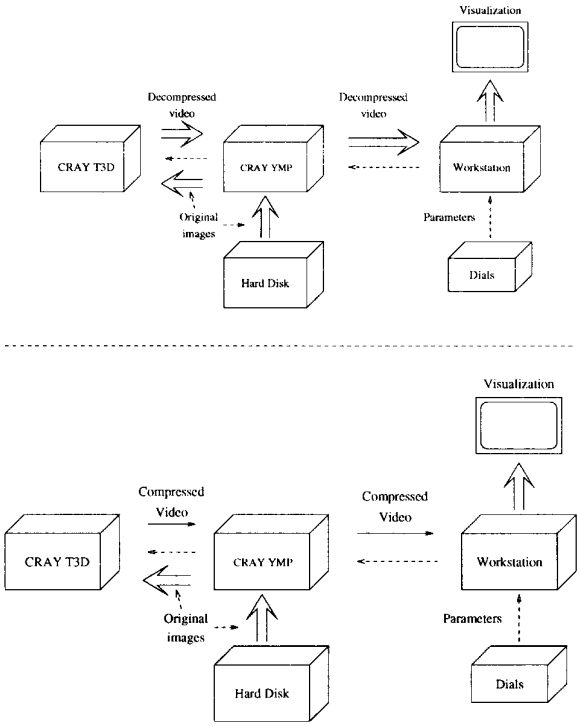

Figure 1: Top: block diagram of *DirectView* 1.0. Bottom: block diagram of *DirectView 2.0.*

black & white CIF images), but the workstation is used only to visualize the decompressed images. A lower transmission bandwidth (depending on the compression ratio) is required with the second version, but higher computational load is devoted to the workstation. *DirectView* 2.0 works correctly only if the workstation is able to decompress an image as quickly as the T3D compresses an image. If this is not the case, the computational power of the T3D is not fully exploited (it waits for the workstation). In this case, and if the transmission link capacity is sufficiently large, *DirectView* 1.0 is more efficient.

Apart from these differences between the two versions, *DirectView* works as follows: The original images are read on

Table 1. *DirectView* 1.0 versus *DirectView* 2.0

	CRAY T3D			Transmission link	Workstation	
DirectView	Compression	Bitstream generation	Decompression	Transmitted data	Decompression	Display
1.0	Yes	Yes	Yes	Decompressed images	No	Yes
2.0	Yes	Yes	No	Compressed images	Yes	Yes

the hard disk of the CRAY YMP by one processor of the T3D. The images are then broadcasted to the other processors using optimized communication routines. Each processor compresses in parallel its own partition. The bitstream (or the decompressed images) is finally collected by a processor which sends this data to a workstation via Ethernet or an FDDI (Fiber Distributed Data Interface) link. The decompressed images are immediately visualized on the screen (after decompression if *DirectView* 2.0 is used) together with the bit rate or the compression ratio. This permits an easy evaluation of the trade-offs between the quality of the images and their transmission cost. Moreover, this system allows the interactive modification of the parameters of the algorithm running on the T3D, thus permitting their optimization. This is easily done by turning a set of dials, the corresponding values being sent by the workstation to the T3D in order to modify the parameter values of the compression algorithm.

Furthermore, *DirectView* permits not only the real-time visualization of the final result (the decompressed images), but it also allows the visualization of various information used by the image processing algorithm such as the segmentation of the images, various statistical information (such as the histogram), the content of the bitstream, etc. This is of major importance for researchers since it allows fast and easy visual representation of the various data manipulated in the program.

Figure 2 shows an example of data visualized on the screen, i.e., the decompressed image, a graph of the bit rate, the current value of the parameters and the histogram of the displayed image.

2.3 Main advantages offered by DirectView

Generic implementation. All the software developed for *DirectView* has been written using C language. This permits flexible and modular software implementations. Consequently, the video compression algorithm can be easily changed or modified just by replacing the corresponding routines by new ones. This leads to a generic implementation which permits the testing of a wide range of video processing methods or algorithms in a short time. This is a definite advantage for research centers in the race for the design of new multimedia systems.

Interactive Optimization The main advantages of interactive optimization over the classical off-line one are the following:

- **Human visual evaluation:** *DirectView* permits the visualization of the decoded images as they are being produced by the T3D. Therefore, the operator can visually evaluate the spatial and temporal artifacts generated by the compression algorithm. The Peak Signal to Noise Ratio (PSNR) is classically used to compare original images with their decompressed versions, but it does not provide fair evaluation of the image quality as shown in Fig. 3. As a consequence, *DirectView* permits the evaluation of the quality which is significantly better than evaluations obtained with mathematical criteria**.**
- **Interactive multi-variable optimization:** Using this system, the operator can change interactively the parameters of the compression algorithm in order to balance the trade-off between transmission cost and the quality of the decoded images. For this task, he typically has to do a multi-variable optimization for which humans are well suited. For example, a human operator will intuitively appreciate the regularity of the algorithm and will very quickly avoid local minima in which minimization algorithms are often trapped.

  Furthermore, *DirectView* avoids all the complex operations that are needed with classical implementations to visualize the results obtained for each value of the parameters (generation of all the results, ftp transfers between machines, format conversions for visualization, etc.). Often, researchers using classical serial implementations do not perform all the exper-

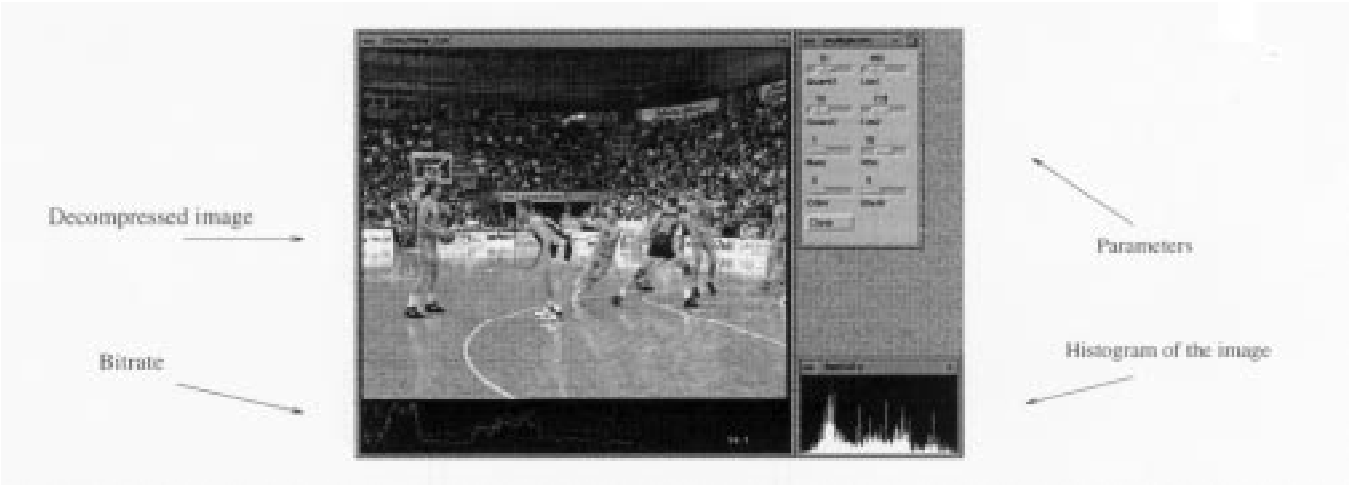

Figure 2: Example of data visualized on the screen, i.e., the decompressed image (top left), the current values of the dials (top right), the bit rate (bottom left), and the histogram of the image (bottom right).

imentations required for a correct optimization procedure for lack of time.

- **Debugging**: Since it is possible to interactively change the values of the algorithm parameters, almost all possible values can be investigated very quickly in order to test the robustness of the algorithms. This enables quick detection of special cases for which the algorithm has problems. In practice, the debugging process is significantly accelerated.

Figure 3: Meaningless of the PSNR. Left image: PSNR = 25.3dB, compression ratio = 12. Right image: PSNR = 27dB, compression ratio = 2. The left image has an higher visual quality for a better compression ratio although the PSNR is better for the right image.

Figure 4 synthesizes the major advantages offered by massively parallel computing in the framework of *DirectView*, i.e., generic implementation and real-time interactive optimization. These advantages open a totally new perspective for researchers working in the field of video compression and analysis. The quick prototyping and productivity gain provided by *DirectView* permit us to improve the quality of the results and to accelerate the development of new techniques.

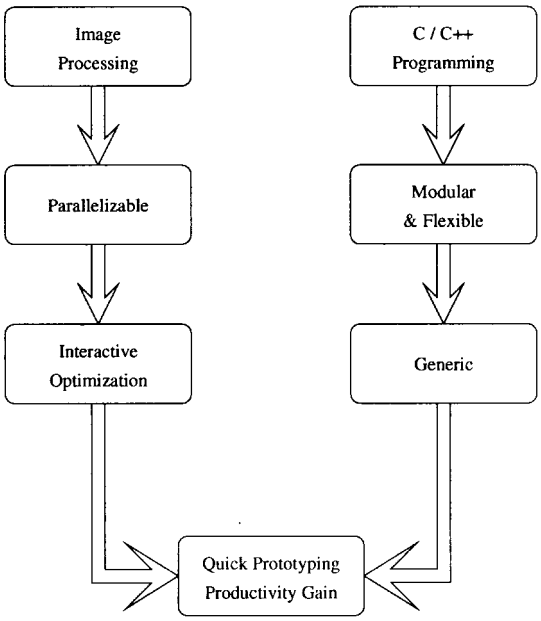

Figure 4: Main characteristics of *DirectView*.

3 Parallel implementation

To enable efficient visual evaluation of the images, they must be displayed on the screen with the highest frame rate possible. This requires optimal parallelization. The iterative algorithm running on the T3D can be schematically divided into the seven following stages:

1. **Read initial images**. The original images are initially stored on the hard disk of the YMP computer. They are read by one processor of the T3D.
2. **Data partition**. In order to optimize the load balancing as much as possible between the processors while keeping a simple data partition, the images are divided into *N* parts of equal size, where *N* is the number of processors. An example with 4 processors is given in Fig. 5. This data partition is independent from the compression algorithm.
3. **Data compression**. Each processor treats only its respective data partition. This stage represents more than 50% of the total computation time. Large speed-ups can generally be obtained for this stage since video processing algorithms are naturally highly parallelizable.
4. **Local bitstream generation**. Each processor generates its own part of the bitstream, locally compressed.
5. **Generation of the entire bitstream**. One processor collects the local bitstreams generated by each processor and builds the entire one.
6. **Image decompression (only for *DirectView* 1.0)**. The decompression is performed in parallel on the T3D, the computation load is very low.
7. **Communications with the workstation**. The sockets of the YMP are used in order to accelerate as much as possible the bidirectional communications between the T3D and the workstation. Two processors of the T3D are in charge of these communications:

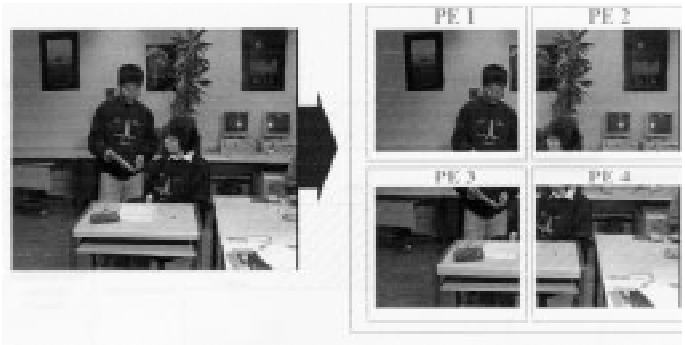

Figure 5: Data partition for 4 processors

(a) **Send bitstreams**: One processor is in charge of sending bitstreams to the workstation (one send per iteration). In order to parallelize the generation of the bitstream (stage 5) and its sending to the workstation, the bitstream of image *t* - 1 is sent while the bitstream

of image t is generated. This requires the transmission of the bitstream between these two processors. Since non-blocking communications are used, the communication time is negligible.

(b) **Receive new parameters**: Another processor tests periodically (for every other iteration) if new parameters values have been received from the workstation. In this case, these new values are broadcasted to all other processors. For *DirectView* 2.0, some of these parameters are necessary for the decompression, consequently, they must be incorporated into the bitstream to be resent to the workstation. Thus we assure good synchronization between the coder and the decoder.

Stages 2, 3, 4 and 6 are done in parallel by all processors. Each of the stages 1 and 5 is performed by one processor, while 2 processors are used for stage 7. Figure 6 gives an illustration of the work done by each processor. Strong effort has also been made to optimize single processor performance.

4 Description of the compression algorithm

The compression algorithm which has been implemented in *DirectView* is a typical example of algorithms used or proposed for the video standards MPEC2 and MPEG4, particularly in term of computational complexity. The block diagram of the proposed compression algorithm is illustrated in Fig. 7. The temporal correlation between successive images is exploited by a classical block-based motion estimation and compensation

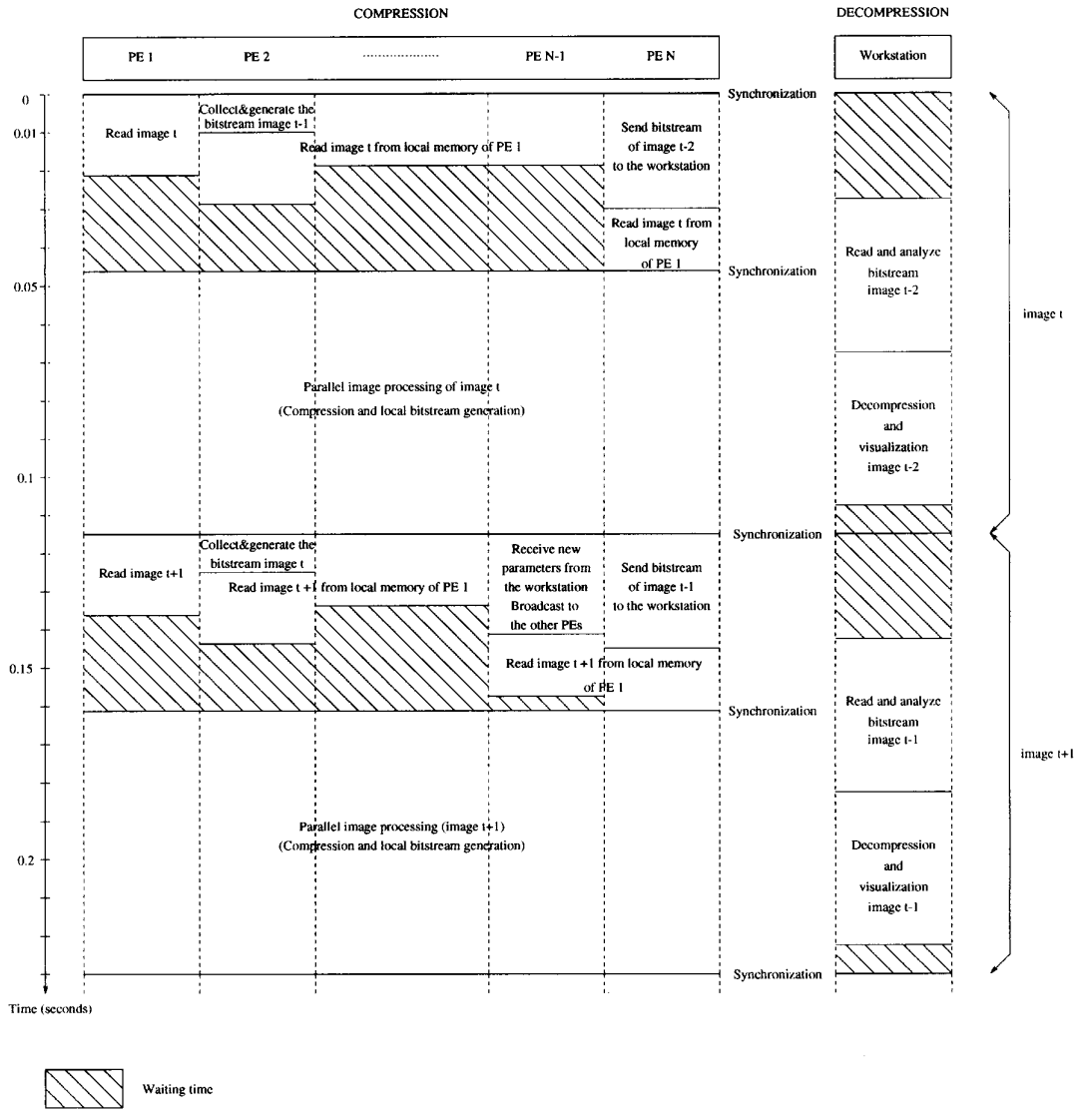

Figure 6: Load balancing representation (time measured for 64 processors)

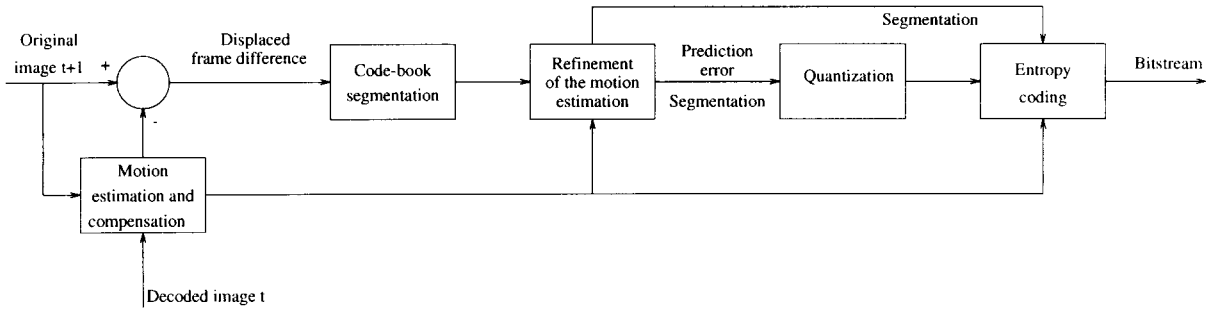

Figure 7: Simplified block diagram of the compression algorithm.

technique. A codebook-based segmentation is then applied on each block in order to refine the segmentation. This is obtained by splitting the blocks when the quality obtained after motion compensation is too low. The quality evaluation is based on the PSNR. Only the blocks which have a PSNR higher than a threshold *T* are split [9]. In the created regions, a refinement of the motion vectors is performed using the motion displacements of the neighbor blocks in order to improve the prediction quality. Finally, the prediction error for the regions where the PSNR remains higher than *T* is quantized and entropy coded.

The final bitstream contains four different kinds of information:

1. Motion parameters.
2. Description of the segmentation.
3. Quantized prediction error.
4. Flags which indicate in which regions the prediction error is coded.

The bitstream is sent to the decoder where it is decomposed. The decompressed images are finally recovered after motion compensation and by adding the quantized prediction error.

5 Experimental results

The experiments have been carried out using a CRAY T3D and black & White CIF test image sequences. The bitstream has been sent to the workstation via an FDDI link, but Ethernet can also be used.

Figure 6 illustrates the load balancing of the system and the work done by each processor and by the workstation. The time spent spending the bitstream (about 0.03 second, depending on the compression ratio), and writing a new image into the local memory of all processors (around 0.015 second per PE) explains the degradation of the speed-up. Figure 8 shows three different

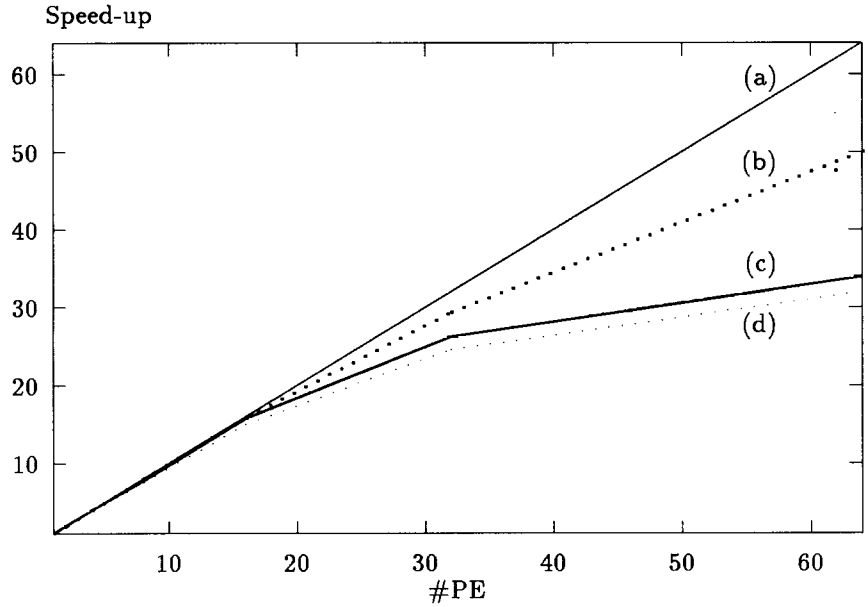

Figure 8: Speed-ups (*DirectView* 2.0). (a) Theoretical speed-up. (b) Speed-up of the compression algorithm only. (c) Speed-up obtained for the whole system, but without communication with the workstation. (d) Speed-up for the whole system.

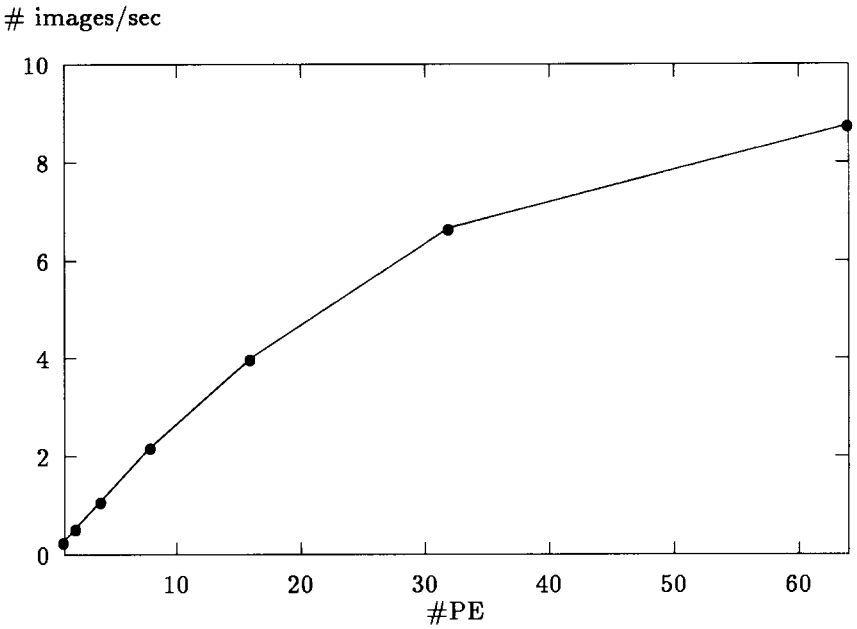

Figure 9: Number of processed images per second versus number of processors.

speed-ups. The first one (curve (b)) corresponds to the speed-up obtained for the image processing part (stages 2, 3 and 4). The curves (c) and (d) correspond to the whole system with and without the communications with the workstation, respectively. A speed-up of around 50 is reached with 64 PEs for the image processing part while the whole system gives a speed-up of around 32. No significant degradation of the speed-up is detected by taking into account the communications with the workstation.

Figure 9 shows the number of compressed and visualized images per second versus the number of processors. From 1 to 16 processors, less than five images are visualized per second. The operator has the impression of seeing a succession of still images and not a video sequence. With 32 processors, a frame rate of about 7 per second is reached. This is the limit where we began to see a video sequence. With 64 processors, the frame rate is sufficiently fast to enable a fair evaluation of the temporal characteristics and artifacts of the decompressed video sequence, thus permitting fast and efficient evaluation and optimization.

After optimization of the parameters with *DirectView* (threshold *T*, quantization step, maximal displacement for the motion estimation, ...), compression ratios of about 20 can be obtained while keeping acceptable visual quality. Only a few minutes are necessary to optimize the algorithm. Using a serial implementation on a workstation and PSNR-based optimization criterion, this optimization requires several days. Figure 3 shows two images which have been optimized using PSNR-based criterion or visual evaluation. It shows that visual evaluation provides higher compression ratios with better visual qualities.

6 Conclusions

This work proves that the T3D can be efficiently used for the fast development of new video processing algorithms by exploiting their intrinsically parallel nature. In this context, the technology developed with *DirectView* offers real-time interactive optimization facilities with the advantages of flexible and modular C software implementations. Better results are obtained both in terms of compression ratio and in terms of visual quality of the decompressed images, compared to classical approaches based on mathematical optimization criteria. Furthermore, *DirectView* can be used to develop and to optimize a wide range of video processing algorithms, thus permitting quick prototyping and significant productivity gain.

Acknowledgment This work was supported by Cray Research and the Swiss Federal Institute of Technology (EPFL) in the framework of a joint project called the “Parallel Application Technology Program (PATP)”. The authors are also grateful to Steve Williams from the EPFL who help us in the development of fast communication routines between the T3D and the workstation.

7 References

[1] G.W. Cook and E. J. Delp. An investigation of JPEG image and video compression using parallel processing. *Proc. of ICASSP*, Adelaide, Australia, April 1994.

[2] H. Jeschke, K. Gaedke, and P. Pirsch. Multiprocessor performance for real–time processing of video coding applications. In Journal *of VLSI Signal Processing, Vol. 2, No. 2, pp. 221-230*, June 1992.

[3] H. Nicolas, A. Basso, E. Reusens and M. Schutz. Parallel implementations of image sequence coding algorithms on the CRAY T3D. *Supercomputing Review*, Vol. 6, EPFL, pp. 28-32, November 1994.

[4] B. Carpentieri and J. Storer. Split-merge video displacement estimation. *Proc. of IEEE*, Vol. 82, No. 6, pp. 940-947, June 1994.

[5] H. Lee, J. Liu, A. Chant and C. Chui. Parallel vector quantization algorithm for SIMD multiprocessor systems. *Proc. of the 5th Data Compression Conference*, Snowbird, *UT, USA, pp. 479,* 1995.

[6] ISO/IEC JTC1/SC29/WGll/N999. MPEG-4 Testing and Evaluation Procedures Document Application and Operational Environments (AOE) Group. *document MPEG95/Tokyo,* July 1995.

[71 K. Shen, W. Cook, L. Jamieson and E. Delp. An Overview of Parallel Processing Approaches to Image and Video Compression. *Proc. of Image and Video Compression, SPIE, San Jose, CA, USA,* Vol. 1, pp. 197-208, 1994.

[81 C. Cheng, C. Wu, S. Pei, H. Li and B. Jeng. High speed video compression testbed. *IEEE Transactions on Consumer Electronics,* Vol. *40, No. 3, pp. 538-548,* August 1994.

[9] F. Dufaux, I. Moccagatta, F. Moscheni and H. Nicolas. Vector quantization-based motion field segmentation under the entropy criterion. *Journal of visual communication and image representation,* Vol. 5, No 4, pp. 356-369, December 1994.