A Case for a Collaborative Computing Tool for Image Processing

Jeffrey A. Shafer, Joseph T. Fieler, Shawn T. Nichols
Frank A. Scarpino, Ph.D., John G. Weber, Ph.D.

Department of Electrical and Computer Engineering
Center for Collaborative Computing
University of Dayton
300 College Park, Dayton, Ohio 45469-0226

Keywords:

Image Processing, Reconfigurable Computing, FPGA, Wavelet Transform, Image / Video Compression, Signal Processing, Human Factors

Abstract

One of the challenging tasks in designing a superior image compression system is encouraging collaboration between the image processing researchers, system designers, and field users. To facilitate information sharing between these key players, a software package that allows the user to prototype various image compression algorithms was designed. This tool allows the users to analyze the compression results both quantitatively and qualitatively, and automatically documents the results. An overview of how the user interacts with the program and technical details of the image compression algorithms implemented are provided.

Building upon the software prototyping system, several methods are examined to transition to a final real-time hardware compression system. One method is to build a set of wavelet filter banks based upon proven FIR technology. The second method is to build an experimental processor optimized for vector and matrix calculations. This processor allows for a quick transition from software to hardware since most of the compression algorithms use matrix mathematics.

1. Introduction

One of the quintessential problems facing image/video compression designers is the constant struggle between qualitative image quality measures and hard-line quantitative data values. An outstanding image compression ratio is useless if a doctor lacks the detail to correctly diagnose a patient or an intelligence operative cannot clearly decipher the contents of a map. Emphasizing collaboration early in development can enable the development of high-quality algorithms that both meet the quantitative metrics required and, equally important, the qualitative goals of the end-users.

Therefore, a tool is needed to solve these image compression problems. The tool must provide a way to analyze the results of competing algorithms both quantitatively and qualitatively while ensuring all individuals within the design process provide input.

2. Subjectivity vs. Objectivity

The nature of image compression is extremely subjective. The interpretation of "sufficient" quality varies from application to application and more importantly, from person to person. However, the engineers actually implementing the software/hardware systems need raw data thresholds for their algorithms to function correctly. These two dilemmas yield a difficult dichotomy between subjective human interpretations and objective mathematical abstractions.

Several mathematic metrics exist to help abstract the image quality problem. Traditionally, Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) are employed by the scientific community in an attempt to numerically represent image quality. More resent developments in the image compression field have yielded another metric known as the universal quality index [10]. In the Wang and Bovik paper, the inaccuracy of MSE and PSNR was described. For instance, their research showed "the performance of MSE is extremely poor in the sense that images with nearly identical MSE are drastically different in perceived quality." They clearly demonstrate that MSE and PSNR are insufficient as standalone standards.

The universal quality index improves on the MSE and PSNR notions, but it still does not incorporate the most important image compression factor: final image quality. All of these mathematical abstractions are wonderful and provide significant value to image compression, but the end user almost always desires clear images regardless of the performance metric used.

The tool described in this paper allows the user to view both the subjective data and objective data simultaneously. Subjective data in this case is side-by-side image comparisons between, for example, the original image and final compressed image, while objective data is mathematical data values such as the MSE, PSNR, or Quality Index. By showing both on screen simultaneously,

this format allows users to attack the compression problem with a two-prong approach.

3. Importance of Collaboration

Even in today's communication-saturated society, the proliferation of critical design information is still one of the greatest challenges. Mathematicians, physicists, human factors groups, and engineers must constantly share information in order to get viable, cost-effective products to market. Furthermore, these individuals are usually not the people using the end product. Therefore, in case of image and/or video compression, the opinions and concerns of the actual field users (e.g. Military Screeners, Doctors/Medical Technicians, Internet Broadcasters, etc.) are crucial.

The blending of technical and non-technical people poses serious drawbacks to the design phase. This is because the various groups concentrate on different portions of the problem. Image processing researchers often focus on one aspect of an image compression system. For example, a researcher might spend years developing a preprocessing filter that reduces noise in an image, which improves compression ratio [6]. This researcher may not be concerned with the rest of the system. Similarly, the human effectiveness team is trained in the physical and psychological nature of the human body. These people have little or no regard for the preprocessing filter built by the aforementioned researcher.

System engineers focus on implementing the overall system. In a video compression system the algorithm needs to be fast enough to meet a desired frame rate. System engineers are also interested in the transmission of the compressed image. Finding a balance between the bandwidth restrictions of so many bits/second, and the image integrity retained by the algorithm is on one of their greatest obstacles. According to Lt Col Kurt A. Klausner of the USAF, "Adding more bandwidth through the use of satellites is expensive at best and still might not solve all the problems." Clearly, this is an issue facing today's military [3].

In general, the end users simply desire the clearest image and the lowest possible cost. Clarity and cost are both subjective terms and vary throughout the application platforms. The end user may be interested in image quality, system latency, or compression ratios. A user in the medical field may need a lossless algorithm, but he or she is willing to sacrifice bandwidth for image quality. On the other hand, a web broadcaster needs to preserve bandwidth, and might accomplish this task at the expense of image/video quality.

4. Rapid Prototyping Interface Overview

In order to solve the aforementioned problems, a Rapid Prototyping Interface (RPI) was developed. This tool provides a means of efficiently evaluating competing compression algorithms. It directly supports the objective of this research effort, which is to both design and implement a superior hardware image/video compression system suitable for real-time embedded applications.

The RPI has two main areas of competence. First, it creates an environment allowing users to simultaneously analyze qualitative and quantitative results. Second, it provides a means of documentation that summarizes all of the qualitative and quantitative data. This fusing of information provides a common ground for both the technical and non-technical users to employ in their design discussions.

The RPI permits the user to select a source image or video file and subject it to a full set of compression and decompression routines. By coupling the RPI software with reconfigurable/adaptive computing techniques, which accelerate development-to-deployment time, users at all stages are able to contribute feedback that directly impacts the final product.

Although both academic [4] and/or commercial [5] software packages exist that provide wavelet image and video compression capabilities, these packages lack the self-documenting capabilities of the RPI that enhance its effectiveness as a collaborative tool. Further, existing algorithms often utilize floating-point arithmetic, an obstacle in transitioning to the real-time embedded hardware implementation which is this group's final objective. Finally, such pre-built implementations are typically difficult to decipher. The time spent understanding and customizing another group's specific implementation of known algorithms could be better spent developing our own algorithms and architectures tailored to specific project requirements. Such a program plan has the side benefit of building a team of researchers intimately familiar with the details of wavelet compression technologies.

5. Rapid Prototyping Interface Capabilities

The current RPI accommodates images represented as standard color planes such as red-green-blue (RGB). These can be opened from a variety of file formats including bitmap, TIFF and JPEG for images and AVI for videos. A wide range of image transforms are implemented in the RPI, including the Haar and Daubechies 4-20 wavelets [1] and a DCT. The system can convert images to luminance-chrominance representation such as YUV and YIQ which can be exploited to increase compression effectiveness. The

user may elect to employ horizontal and/or vertical down-sampling of the chrominance signals from easy to use menus contained in the RPI. Additional menu selections provide parametric selections for down-sampling by two or four (or more) in each of the chrominance signals. Further sophistication in the down-sampling process permits the user to choose up sampling approaches (resizing methods) such as linear, nearest and bi-cubic. Finally, a suite of encoders is provided, including stack-run [7] and run-length algorithms.

Figure 1 below shows the main interface for the RPI, which was designed to be simple to operate for a wide range of users.

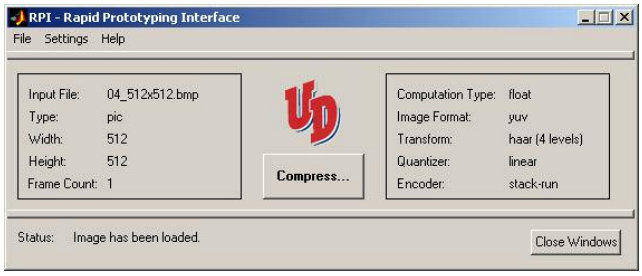

Figure 1. Main RPI Dialog

The user can access all image processing options from the 'Settings' menu. These options include the transform type, quantizer step size, encoder, image representation, down-sampling, and up-sampling. The RPI saves the selected algorithms in a preferences file so that the user is not forced to re-enter all the options each time the RPI is executed.

Once the user has selected the parameters of interest for a particular compression evaluation, the compression process is activated. A typical display corresponding to this activation is shown in Figure 2 below.

Figure 2. Typical RPI Presentation of Images

The RPI displays an array of images of interest to the evaluator. Depending upon the selection made by the user, it presents the image prior to compression (original image), the image after a pre-processing pass (e.g., a noise reduction pass), the transformed image (e.g., the image after each multi-resolution from a wavelet process) and the reconstructed image (the image after compression and decompression processing).

The RPI is designed to allow for the easy expansion of new algorithms. Each transform, quantizer, encoder, etc. is written with standard function parameters. If a researcher wishes to add a new function then it merely needs to be written according to the standardized function definition.

This modular design has already provided benefits for collaboration both inside and outside of our research team. The initial version of the RPI only had a run-length encoder. Another researcher from a different university was using a stack-run encoder and obtaining higher compression ratios. Our team was able to write a stack-run encoder function for the RPI and have the new algorithm providing results within a few days.

6. RPI as a Collaborative Tool

The RPI contains an automatic report generation facility. Evaluators at all levels of research, implementation and deployment are well aware that documentation is one of the most tedious and often neglected aspects of any evaluation. It is often in the documentation effort that a great deal of time is consumed and the most "breakage" of the required continuity for evaluation occurs. Cognizant of this, the RPI *eliminates this problem* for the evaluator by generating an HTML output report. An example output report is shown below in Figure 3.

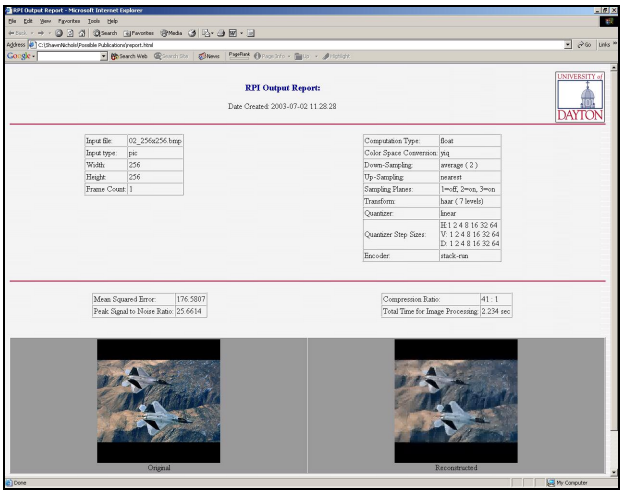

Figure 3. Example RPI Output Report

As the figure indicates, the automatically generated report contains the essential information and preserves the test and evaluation data. Not only is a great deal of time saved but the user is assured that the correct information is retained for future comparisons and collaboration. Collaboration among researchers, prime contractors, implementers and end-users is enabled and greatly facilitated. Any of the groups retaining use of the Rapid Prototyping Interface is able to confirm the accuracy and veracity of the efforts of their colleagues. Confidence among the groups responsible for development through deployment is thus facilitated and corresponding acceleration of the development through deployment cycle is enabled.

While it is purely a software package, the RPI directly supports the overall program goal of implementing a superior hardware image / video compression system suitable for real-time embedded civilian and military applications. This is because it is built in such a way that it simultaneously allows researchers, prime contractors, and field users to collaborate on algorithm development and analysis. This coordination of mission efforts allows everyone in the design to provide input on the final product. Figure 4 below illustrates the program plan showing how the RPI supports the hardware program objectives.

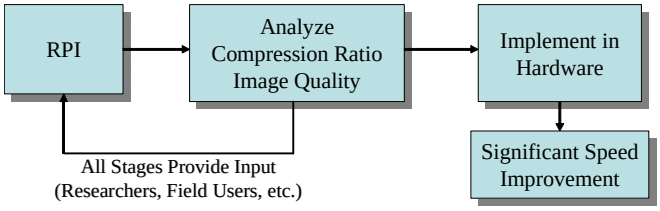

Figure 4. Development to Deployment Program Plan

The methodology behind the RPI and the program plan shown in Figure 4 have been hugely praised and accepted as a viable solution to several Air Force obstacles such as: large satellite bandwidth requirements, poor dissemination of mission critical data, and sluggish development-to-deployment times. In fact, elements of the Predator System Program Office and prime Predator contractors are currently evaluating version 1.0 of the RPI. The effectiveness of the RPI in these early real-world tests is still being determined and will be discussed in a future case study.

7. Transitioning to Hardware Implementations

The main purpose of this team's research is to produce image/video compression algorithms for real-time embedded hardware systems. The RPI furthers these goals by allowing the development and evaluation of algorithms by a wide spectrum of end-users in a purely software environment. Despite modern advances in Field-Programmable Gate Arrays (FPGAs) and hardware description languages such as Verilog or VHDL, past experience has shown that the hardware implementation is still a tedious and time-consuming process. Having a software prototyping system such as the RPI allows for several algorithms to be developed and tested in the same time it would take for a single algorithm to be built in hardware. The software environment encourages developers to explore new ideas confident that the time commitment to see a working demonstration is small.

At some point, however, working algorithms must be transitioned from the software based RPI to a hardware based real-time embedded system. Two approaches to accomplish this are considered. These approaches are both in the early stages of development, and will be discussed in depth in future publications as work continues.

The first approach in transitioning to a hardware implementation is to use wavelet filter banks that are based upon traditional Finite Impulse Response (FIR) filters. Research teams have used the approach in the past to great success [8, 9].

The second approach is to develop an experimental vector processor. Many of the RPI algorithms mentioned above are elegantly implemented using matrix mathematics (e.g. down-sampling, wavelet transforms). The proposed processor described in this section contains instructions that naturally handle vector and matrix problems. Such an instruction set allows the user to easily transition modules from the RPI to hardware.

7.1 Wavelet Filter Banks

Wavelet transformation algorithms may be represented in a two-channel filter bank as pictured in Figure 5. H_0 and H_1 represent the analysis filters while G_0 and G_1 represent the synthesis filters of the bank.

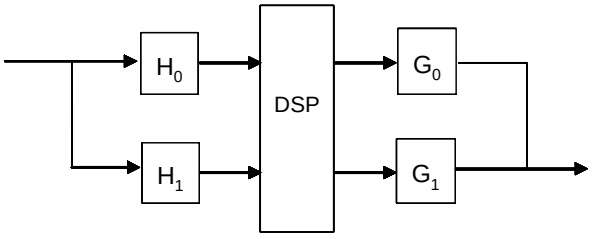

Figure 5. Two-Channel Wavelet Filter Bank [2]

The filters, H_0, H_1, G_0 and G_1 may be constructed of finite impulse response filters [11]. The FIR filters of Figure 5 are programmed in VHDL or Verilog and implemented on an FPGA. The coefficients of the filters are stored in registers on-chip.

This wavelet filter bank is part of a larger system as shown in Figure 6.

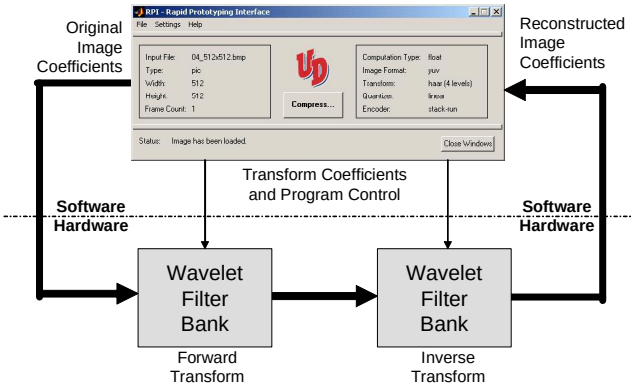

Figure 6. Wavelet Transform System

This system links two wavelet filter bank modules, performing both a forward and inverse wavelet transform, with the RPI running on a stand-alone PC. If desired, multiple filter banks and signal processors may be instantiated when concurrent processing is desired.

By linking the hardware modules with the RPI, an efficient testing and demonstration system is created. Once the hardware components are verified, the RPI dependence can be removed and the filter banks will be linked with whatever real-world system is desired.

The source image is loaded via the RPI, sent through the filter banks, and received via the RPI for final display. The transform coefficients can be readily modified through the RPI and delivered to the hardware device. The hardware has an on-chip serial port interface that receives the coefficients and image data from the PC and stores the coefficients into registers.

7.2 Vector Processor

The wavelet filter bank approach previously described is straightforward to implement and thus is highly useful for prototyping systems and gaining designer familiarity with the hardware devices. It is limited in scope, however, and can only perform forward and inverse image transformations. In contrast, the proposed matrix processor can calculate vector and matrix operations for any application. Within the realm of image processing, it can transform, up and down-sample, and quantize images.

The mathematical process of image transformations may be characterized as a matrix problem. A gray-scale image can be represented by a matrix, $\mathbf{I_m}$ consisting of pixel values, while a color image consists of a pixel matrix for each color plane. The transform of the image is accomplished by constructing a transform matrix, **W**, and forming the transformed image $\mathbf{TI_m}$ by the operation

$$\mathbf{TI_m = W\ I_m\ W^T}$$

where $\mathbf{W^T}$ is the transpose of **W.** This operation performs the two-dimensional transformation of the image matrix [2].

The RPI, a custom-written MATLAB program for video and image compression, uses matrix operations. MATLAB was chosen, among other reasons, because its native mathematical paradigm (from the user perspective) is vector/matrix operations. Characterizing the transformation algorithms as matrix manipulations has two key advantages. First, the transform (and reconstruction) matrices may be computed a` priori, reducing the amount of computation required for each image. Second, the matrix algorithms become identical and depend only upon the values in the transform matrix and the image matrix. Both of these advantages are highly significant for hardware implementations.

Following the lead of earlier, non-matrix oriented software prototypes, our previous hardware implementations mimicked the serial execution of multiply and add operations in a dedicated hardware device. By observing two key advantages of characterizing the algorithms as vector/matrix manipulations, however, a clearly beneficial new research effort has been initiated to develop a parallel, reconfigurable, hardware vector/matrix processor which supports the compression of image and video information.

We have noted that many video compression algorithms, such as wavelet transformations, can be expressed in terms of matrix manipulations. These matrix operations are easy to perform in our MATLAB software environment or even in a custom C++ class. Current hardware implementations of these algorithms, however, perform calculations in a linear element-by-element sequence. While mathematically equivalent and simple to implement, such designs are highly inflexible, and may require a total re-write to accommodate minor changes in the transformation algorithm being executed. In contrast, the vector processor allows for a near-infinite number of

different transformation matrices to be applied to a common data block with no hardware changes required in-between. Further, because the transform (and reconstruction) matrices may be computed a` priori, the system performance is further enhanced over the past sequential implementations.

This processor is being developed upon the "blank slate" of a System-On-Programmable Chip (SOPC) device such as the Xilinx Virtex-II Pro FPGA. Modern reconfigurable computing technology offers clock rates up to 125 MHz and up to 10 million programmable gates. With high-speed I/O and on-chip memory, these devices are ideal for data and video processing tasks. Further, they feature dozens of built-in hardware multipliers, which will allow the vector processor to calculate results in parallel.

The payoff of this research is a low-cost design that combines the performance and flexibility of both hardware and software approaches within the framework of embedded video compression. This approach provides a significant advantage in today's military climate of total information awareness, where every player in the information battlefield is clamoring for real-time video data of terrain and targets.

8. Concluding Remarks

The RPI solves two significant image processing problems. It allows for analysis of both subjective image and objective quality metrics. Also, the RPI simultaneously allows technical and non-technical people (i.e. researchers, prime contractors, and field users) to collaborate on algorithm development and analysis.

The RPI permits the user to select a source image or video file and subject it to a full set of compression and decompression routines. It gives the user the flexibility to test different compression algorithms and compare results. The modular design supports easy expansion to add more algorithms. It automatically documents the test results on a single page, which allows for easy team collaborationand eliminates this burdensome task from the user.

By coupling the RPI software with reconfigurable/adaptive computing techniques, which accelerate development-to-deployment time, users at all stages are able to contribute feedback that directly impacts the final product.

Future objectives of the RPI include increasing the RPI distribution to additional users, improving the capability of the RPI as a collaborative tool, and implementing new image/video compression algorithms. Some of the algorithms to research include preprocessing filters to reduce noise, more wavelet transforms, and different encoders such as a Huffman encoder.

To enhance the collaborative nature of the RPI further, it has been proposed to extend the RPI to be a client-server system. Each project team using the RPI would have a server (or an account on a single server) that would be responsible for distributing a common set of test images or video for processing. Using a copy of the RPI installed on client systems, the end users could perform the same image analysis tests described previously. At the end of each test, the results (both quantitative and qualitative) would be automatically uploaded back to the server, permitting the developers to easily examine results from a large number of end-users.

9. References

[1] Daubechies, I., *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, 1992

[2] Goswami, J.C. and Chan, A.K, *Fundamentals of Wavelets: Theory, Algorithms and Applications*, John Wiley & Sons, New York, 1999

[3] Klausner, K., "Command and Control of Air and Space Forces Requires Significant Attention to Bandwidth", *Air and Space Power Journal*, Vol. 17, No. 4, Winter 2002, pp. 69-77

[4] Lagendijk, I. et al, "VcDemo: Image and Video Compression Learning Tool," August 2003, http://www-ict.its.tudelft.nl/~inald/vcdemo/

[5] Pegasus Imaging Corporation, "PICVideo Wavelet2000 Codec," http://www.pegasusimaging.com/picvideowavelet.htm

[6] Pizurica, A., Philips, W., Acheroy, M, "A Joint Inter- and Intrascale Statistical Model for Bayesian Wavelet Based Image Denoising", *IEEE Transactions on Image Processing*, Vol. 11, No. 5, May 2002

[7] Tsai, M. J., Villasenor, J.D., Chen, F., "Stack-Run Image Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 5, October 1996, pp. 519-521

[8] Turri, W, "Design And Hardware Implementation Of A. Wavelet-Based Color Image Compression System," University of Dayton Masters Thesis, May 2002

[9] Villasenor, J.D., Belzer, B., Liao, J., "Wavelet Filter Evaluation for Image Compression", *IEEE Transactions on Image Processing*, Vol. 4, No. 8, August 1995, pp. 1053-1060

[10] Wang, Z. and Bovik, A.C., "A Universal Image Quality Index", *IEEE Signal Processing Letters*, Vol. 9., No. 3, March 2002, pp. 81-84

[11] Xia, X-G and Suter, B. W., "Vector-valued wavelets and vector filter banks," *IEEE Transactions on Signal Processing*, Vol. 44, No. 3, 1996, pp. 508-518