Haar Wavelet Transform for Low Power Video Frames

Héctor Silva-López[1] and Sergio Suárez Guerra [1]

[1] Center of Computational Research, National Technical Institute, México, A.P. 75-476, C.P. 07738, Zacatenco, DF, México.
esilvab05@sagitario.cic.ipn.mx, ssuarez@cic.ipn.mx

Abstract. Video compression is currently a prominent topic for both military and commercial researchers, due to the rapid proliferation of digital media and the subsequent need to store and transmit it in a space and time-effective manner. Most successful compression methods have been based on mathematically transforming an image (or sequence of images) into a frequency domain representation, and then filtering that representation to obtain a form suitable for effective encoding and compression. Our framework can be targeted to multimedia applications where video frames must be encoded, decoded and processed periodically within predefined video frame-rates. The frames arrive at the system dynamically, and leaves after a number of instances executed. The frames execute in a discrete voltage/frequency processor. This framework dynamically selects the sub-bands that maximize the energy savings in the system, such that no sub-band in the system misses its deadline. The problem is presented as a dynamic optimal problem with discrete constraints. Experimental results show that our algorithm present 29% energy saving.

1 Introduction

Although wavelet have their roots in approximation theory [7] and signal processing [14], they have recently been applied to many problems in computer graphics. These graphics applications include image editing [4], image compression [12], and image querying [3]. These applications impose strict quality of service requirements in the form of timing constraints. Ignoring energy consumption, operating the CPU at its highest speed operation quickly drains the batteries. Thus there is a tradeoff between reduced energy consumption and quality of service.

Voltage scaling technology has the potential to exploit such variability in the case of meeting timing constraints. By adjusting the operating voltage of the processor, the energy consumption and speed can be controlled [1]. Power regulator and variable voltage

Control, Virtual Instrumentation and Digital Systems.
Research in Computing Science 24, 2006, pp. 21-30

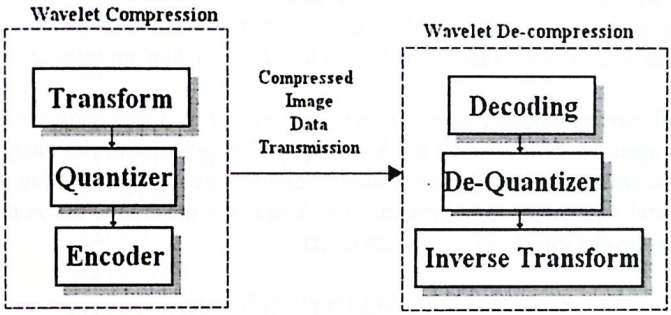
processors with response times in the microseconds range are available [10]. Fast response time makes it practical to dynamically adjust the voltage at run time.
Recently, researches have attempted to apply Dynamic Voltage Scaling (DVS) to video decoding to reduce power [5, 8, 9, 11]. These studies present approaches that predict the decoding time of incoming frames or Group of Pictures (GOPs), and reduce or increase the processor setting based on this prediction. As a result, idle processing time, which occurs when a specific frame decoding completes earlier than its playout time, is minimized. In [6] an alternative method called Dynamic is proposed as an improvement to these techniques. The Dynamic approach is designed to perform well even with high motion videos by dynamically adapting its prediction model based on the decoding experience of the particular video clip being played. The same authors present another alternative method called frame data computation aware (FDCA) in [2]. FDCA dynamically extracts useful frame characteristics while a frame is being decoded and uses this information to estimate the decoding time.
The objective of out work is to develop a novel DVS technique targeting such dynamic changing workloads. We presented a method that can be used to discern how much CPU throughout a wavelet transform is consuming. This information can be used to no deadline is missed and the energy saving is maximized for each sub-band when encoding a frame and decoding the same frame running on one processor.
This paper is structured as follows. In Section 2, we give a framework overview for the wavelets transform scheme. We then describe how is obtained energy saving for each sub-band, and the algorithm for selecting the optimal energy saving is showed in Section 3. Section 4 is an example showing how our scheme works on practical image. Section 5 presents the experimental results to demonstrate the performance of our low power video encoding/decoding scheme under different workload conditions. Section 6 summarizes our efforts.

2 Wavelet Image Compression overview.

Of the many processes available for image compression, two of the most popular transformations are the Discrete Cosine Transform (DCT) used in the common JPEG format, and the Discrete Wavelet Transform (DWT) used in the newer JPEG 2000 format. The DWT differs from the traditional DCT in several fundamental ways. The DCT operates by splitting the image into 8x8 blocks that are transformed independently [13]. Through this transformation process, the energy compaction property of the DCT ensures that the energy of the original data is concentrated in only a few of the transformed coefficients, which are used for further quantization and encoding [15]. It is the discarding of the lower-energy coefficients that result in image compression and the subsequent loss of image quality. Unfortunately, the rigid 8x8 block nature of the DCT makes it particularly susceptible to introducing compression artifacts (extraneous noise) around sharp edges in

an image. This is the "halo effect" seen in over compressed web images. Because the artifacts become more pronounced at higher compression ratios, JPEG's suitability for line drawings and cartoon-like images is significantly impaired.
In contrast to the DCT, the DWT operates over the entire image, eliminating artifacts like those caused by the 8x8 DCT blocking. Like the DCT, the fundamental wavelet transform is completely reversible, meaning that if the forward and reverse transforms are applied in sequence, the resulting data will be identical to the original. In addition, the DWT is based on sub-band coding where the image is analyzed and filtered to produce image components at different frequency sub-bands [18]. This produces significant energy compaction that is later exploited in the compression process. The wavelet's two-dimensional nature results in the image visually being divided into quarters with each pass through the wavelet transformation. A key effect of this transformation is that all of the high pass quadrants in the image contain essentially equivalent data [16]. A wavelet video compression, transmission, and decompression process that represents the target application of this vector processor is shown in Figure 1.

Figure 1: Wavelet Compression, Transmission, and Decompression Process



In this process, a single image or video frame is digitized by a camera and frame grabber. This image is then fed to a wavelet compression system. First, the compression system performs a wavelet transform of the image. This mathematical transform, which is perfectly reversible, converts the original image into a form suitable for encoding through a reversible spatial frequency separation. Next, the transformed image is quantized. This creates redundancy in the transformed image by reducing the number of allowable pixel values and thus the number of color or chrominance levels. Quantization is not reversible and is the principle cause of data loss in lossy compression. (The other cause of data loss can be attributed to floating-point rounding errors when calculating forward and inverse transformations with non-integer wavelets). The resulting bit stream from these two stages contains large blocks of redundant data that the encoder can easily locate and mathematically remove. At this point, a single image or video frame has been compressed into a bit stream that is some appreciable fraction of its original size, and is ready to be

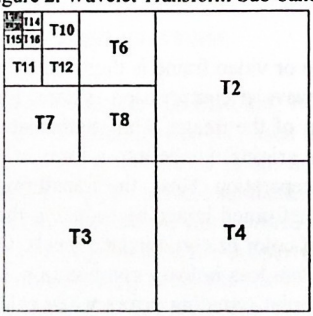
stored or transmitted to the destination system. Once at the destination, the process is reversed, whereby the image is first decoded, dequantized, and then inversed transformed to arrive at what is hopefully a convincing reproduction of the original image.

2.1) Wavelet transform.

In the field of wavelets, the modified Haar Wavelet (referred to as Haar*) is traditionally used for rudimentary image compression because of its algorithmic simplicity and low computational complexity due to an integer-based design [17]. When the wavelet is applied through its filtering process producing the scaling function coefficients (low-frequency) and wavelet coefficients (high-frequency). From the application of the Haar* wavelet, it is evident that the scaling function coefficient is simply the average of two consecutive pixel values, while the corresponding wavelet coefficient is the difference between the same two pixel values. The scaling function coefficients appear to contain all the image data, while the wavelet coefficients appear to be zero (black). If, however, the raw data was examined, it would be evident that the coefficients are only mostly zero. In reality, the wavelet coefficients contain the difference between adjacent pixel values, which is the high-frequency edge information. Because the high-frequency coefficients approach zero, the encoder is more easily able to remove redundant information from the image.

Once the full wavelet transformation has been applied to two dimensions, the process can be repeated again by filtering only the low-frequencies quadrant of the image. This low frequency quadrant is the visible image in the rightmost frame. By repeating the filtering process several times over ever-shrinking low-frequency quadrants, the *multiresolution* aspect of the wavelet transform comes into effect.

Figure 2. Wavelet Transform Sub-bands.



Because the multiresolution wavelet transform cycles by repeatedly processing the low-frequency information, some of the image data is processed more than once. Because of

this, the lower frequency wavelet coefficients are transformed by a wavelet of differing amplitude and duration than the higher frequency wavelet coefficients.
Thus, each sub-band shown in figure 2 was generated by a different wavelet function. It gives the different coefficient sub-bands. The Level 1 correspond T2, T3, and T4. Level 2 (T6, T7 and T8). Level 3 T10, T11, and T12. Level 4 (T14, T15, and T16) and Level 5 (T17, T18, T19, and T20).

3. Formulation of the problem.

The problem can be formulated as follows. Each time a new frame arrives at or leaves the system, the problem is to determine the mode (speed) of execution of the sub-bands such that no sub-band misses its deadline and the energy savings of the system is maximized. Each frame in the system execute in a discrete voltage/frequency processor. Note that a solution to this problem must be computed each time a new frame arrives or leaves the system, therefore a solution with probably cause deadlines to be missed.

The problem is formulated as:

$$\max_{\{u(k)\}_{k=0}^{2}} J = \sum_{k=0}^{2} S_k[u(k)]$$

subject to $x(k+1) = x(k) - u(k)$

$x(0) = 1.0,$

with : $x(3) = 0,$

$u(k) \le x(k),$

$u(k) \in \{1.0, 0.8, 0.6, 0.4, 0.15\}$ *for* $k = 0,1,2$

where

k = *correspond to sub – bands*

$S_k[u(k)]$ = *energy saving*

$x(k)$ = var *iable of state*

$u(k)$ = *control* var *iable*, (*speed to select*).

Bellman's optimality principle is used by compute the state variable in each stage, as follows:

Step 1: x(3) is calculated as follow:

$$J_3^*\{x(3)\} = 0$$

Step 2: x(2) $\in \{1.0, 0.8, 0.6, 0.4, 0.15\}$ is:

$$J_2^*\{x(2)\} = \max_{u(2)}\{S_2[u(2) + J_3^*\{x(2) - u(2)\}\},$$

where:

$$x(3) = 0 = x(2) - u(2), u(2) \leq x(2), u(2) \in \{0.15,0.4,0.6,0.8,1.0\}$$

Step 3: x(1) $\in \{1.0,0.8,0.6,0.4,0.15\}$, the equation that corresponds is:

$$J_1^*\{x(1)\} = \max_{u(1)}\{S_1[u(1) + J_2^*\{x(1) - u(1)\}\},$$

where:

$$u(1) \leq x(1), u(1) \in \{0.15,0.4,0.6,0.8,1.0\}$$

Step 4: x(0)=1.0 is:

$$J_0^*\{1.0\} = \max_{u(0)}\{S_0[u(0) + J_1^*\{1.0 - u(0)\}\},$$

where:

$$u(0) \leq 5, u(0) \in \{0.15,0.4,0.6,0.8,1.0\}$$

4. Example of the algorithm.

To illustrate the execution of the proposed algorithm, we consider the image Lena and we use five discrete speed levels {1.0, 0.8, 0.6, 0.4, 0.15}, in a discrete voltage/frequency processor. The CPU utilization can be measured while the system is under various states of loading. Obviously there is no way (yet) to measure CPU utilization directly. CPU utilization must be derived from measured changes in the period of the background loop. The average background loop period should be measured under various system loads and then the CPU utilization can be obtained. The CPU utilization is defined as the time 'not' spent executing the idle task (is a task with the absolute lowest priority in a multi-tasking system). The amount of time spent executing the idle task can be represented as a ratio of the period of the idle task in an unloaded CPU to the period of the idle task under some known load.

% time in the idle task = (average period of the background task with no load) * 100% / (average period of the background task with some load) [1]

Based on DVS technique, we adjust processor speed for each sub-band with slack reclamation.

In the table 1 present the time in micro seconds for every sub-band of the image. We consider 5 level how is presented in figure 2. The average background task execution time is 195 microseconds (because of space restrictions, we do not include here the details of how obtaining this time).

Table 1 shows too the results of applying equations 2 and 3 to the first and second columns in table 1.

Table 1. Sub-bands vs. Average background loop Period.

Sub-bands	T (microseconds)	% Idle	% CPU
1	974	20.02	79.97
2	535	36.49	63.55
3	388	50.26	49.74
4	859	22.70	77.30
5	417	46.76	53.24
6	353	55.24	44.76
7	595	32.77	67.23
8	368	52.99	47.01
9	287	67.94	32.06
10	445	43.82	56.18
11	313	62.30	37.70
12	267	73.03	26.96
13	342	57.02	42.98
14	281	69.39	30.61
15	247	78.95	21.05

After the executing the first sub-band of the level 1, to the maximum speed, we can obtain the percentage of utilization of this sub-band, the percentage idle is assigned to sub-bands 2 and 3, the executing speed the these sub-bands depending the result of first sub-band. Later, for the others levels, calculating the percentage of utilization for the first sub-band of each level and assigning the percentage idle to the two following sub-bands. In the table 2 is presented the percentage of energy saving level by level. The total energy saving of the all image is 44.19 %.

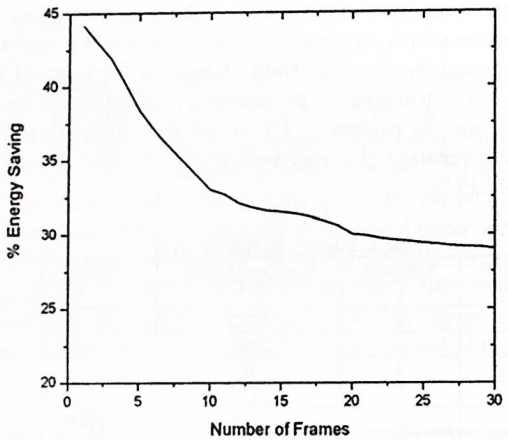
Table 2. Energy Saving for level.

Level	Sub-band	% Energy Saving	Total
1	1	0	
	2	20.02	
	3	20.02	= 37.50 %
2	4	0	
	5	22.70	
	6	22.70	= 36.40 %
3	7	0	
	8	32.77	
	9	32.77	
4	10	0	= 42.64 %
	11	43.82	
	12	43.82	
5	13	0	= 55.53 %
	14	57.02	
	15	57.02	

5. Simulation Experiments.

Our aim in this simulation experiments, was to verify the proposed algorithm in achieving our optimality criteria using different sizes of frames. The goals in this simulation experiments is to measure the energy saving over a large number of frames. Each value on figure 3 represents the results obtained from a set of 30 frames. We can observed that if the number of frames increase, the energy saving decrease until 29% of energy saving. On each frame generated, the metrics Energy Savings is obtained as a result of the execution of algorithm. The Execution Time metric denotes the execution time of each subband, which measures the physical time in microseconds, using a PC Intel PENTIUM Centrino 3.2 GHZ with 512MB of RAM and running on the Operating System SUSE Linux 9.3. The function used for the measurements is psched_get_time().

Figure 3. Energy vs. frames.



6. Conclusions

In this paper we proposed a power optimization method for a video encoder/decoder application running on a variable speed processor with five discrete speeds. The problem is presented as a linear problem with discrete constraints. Approximate solutions proposed are based on Bellman equation. Experimental results show that our algorithms yield near optimal performance with low complexity. For one frame is obtained 44.19% of energy saving but top 20 frames the energy saving is 29%. Also, as part of our future

work, we intend to extend our framework for running a real-time operating systems, improve the performance of our algorithm for increasing the energy saving of its performance and extend the framework to consider a large set of frames.

References

[1] A. Chandrakasan, S. Sheng and R. W. Brodersen, "Low-power CMOS digital design", IEEE Journal of Solid State Circuirs, 27, pp. 473-484, April 1992.

[2] B. Lee, E. Nurvitadhi, R. Dixit, C. Yu, and M. Kim, "Dynamic Voltage Scaling Techniques for power efficient video decoding", Journal of Systems Arquitecture, pp. 633-652, Available online 18 April 2005.

[3] Charles E. Jacobs, Adam Finkelstein, and David h. Salesin, "Fast multiresolution image querying", In Proceedings of SIGGRAPH 95, pages 277-286, ACM, New York 1995.

[4] Debora Berman, Jason Bartell, and David Salesin, "Multiresolution painting and compositing", In Proceedings of SIGGRAPH 94, pages 85-90, ACM, New York, 1994.

[5] D. Son, C. Yu, and H. Kim, "Dynamic Voltage Scaling on MPEG Decoding", International Conference of Parallel and Distributed Systems (ICPADS), June 2001.

[6] E. Nurvitadhi, B. Lee, C. Yu, and M. Kim, "A Comparative Study of Dynamic Voltage Scaling Techniques for Low-Power Video Decoding", International Conference on Embedded Systems and Applications (ESA), Jun 23-26, 2003.

[7] Ingrid Daubechies, "Orthonormal bases of compactly supported wavelets", Communications on Pure and Applied Mathematics, 41(7): 909-996, October 1988.

[8] J. Pouwelse, K. Langendoen, R. Lagendijk, and H. Sips, "Power-Aware Video Decoding", Picture Coding Symposium (PCS'01), Seoul, Korea, April 2001.

[9] J. Pouwelse, K. Langedoen, and H. Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor", 7th ACM Int. Confe, on Mobile Computing and Networking (Mobicom), pp. 251-259, Rome Italy, July 2001.

[10] M. Fleischmann, "Crusoe power management reduces the operating power with LongRun", in Hot Chips 12, Aug. 2000.

[11] M. Mesarina and Y. Turner, "Reduced Energy Decoding of MPEG Stream", ACM/SPIE Multimedia Computing and Networking 2002 (MMCN'02), San Jose CA, 18-25 January 2002.

[12] R. Devore, B. Jawerth, and B. Lucier, "Image compression Through wavelet transform coding", IEEE Transactions on Information Theory, 38(2):719-746, March 1992.

[13] Santa-Cruz, D, T. Ebrahimi, J. Askelöf, M. Larsson and C. A. Christopoulos, "JPEG 2000 Still Image Coding Versus Other Standards," *Proceedings of SPIE* 89 *45th Annual Meeting, Applications of Digital Image Processing XXIII*, Vol. 4115, San Diego, CA, July 30-August 4, 2000, pp. 446-454.

[14] Stephane Mallat, "A theory for multiresolution signal decomposition: The wavelet representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7):674-693, July 1989.

[15] Subramanya, S.R., "Image Compression Techniques," *IEEE Potentials*, Vol. 20, No. 1, February/March 2001.

[16] Topiwala, P.N., Wavelet Image and Video Compression, Kluwer Academic Publishers, 1998.

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

[18] Welstead, S, Fractal and Wavelet Image Compression Techniques, SPIE Optical Engineering Press, 1999.