

Wavelet Image Compression : A Heuristic Approach to Space Frequency Quantisation

Sara Brooks and Mohammad Tabbara

*Department of Electrical and Electronic Engineering
The University of Melbourne
m.tabbara@pgrad.unimelb.edu.au*

Abstract This report describes in detail the fourth year project work of Sara Brooks and Mohammad Tabbara. Our project specification asked us to simulate a wavelet-based image compression algorithm in software. We not only simulated such an algorithm, but used it to develop our own, fully functional image compression software of near-commercial grade. The software is called “alchimage”.

1 Introduction

What is image compression?

The goal of image compression is to remove redundancy in images and store or transmit only the minimum information necessary for reconstruction. The better this aim is achieved by signal processing, the more efficiently we can communicate with digital images. The difficulty lies in the task of coming up with a good model for an image; that is, a mathematical framework that can be used to describe every conceivable image that can be digitally captured. Then the challenge of image compression becomes reconciling two competing constraints: high image quality and low data storage/transmission rates. There is no silver bullet in this exercise. Nonetheless, wavelets have been shown to facilitate leading results.

Why do we need to compress?

Uncompressed image data requires a great deal of memory for storage or bandwidth for transmission. A single 640-by-480 pixel image, for example, can require as much as 1.2 MB for storage in 32-bit color. Sequences of images in a video stream need substantially more storage space or bandwidth. Minimising the amount of data necessary to describe an image is an important consideration for any application that works with images or sequences of images.

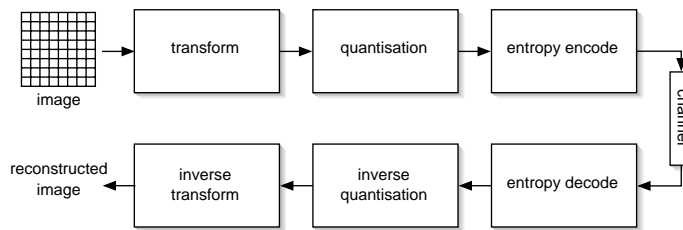


Figure 1: Components of a lossy compression scheme.

Throughout the development process in this project, we worked with sequences of images stored in an uncompressed format. Not only were they exceptionally large, their playback performance was dismal. The majority of modern desktop computers have fairly fast processors but their performance is bounded by the throughput of the I/O bus,¹ making compression a necessity for dealing with full-screen video.

2 Standard Compression Techniques

The diagram in Figure 1 shows the fundamental components in a lossy image compression scheme. First, an invertible linear transform is performed on the raw input data to de-correlate the input sequence. Commonly used transforms include the Fourier Transform, the Discrete Cosine Transform (DCT), transformations to different color spaces, and the Discrete Wavelet Transform (DWT). Second, the transform coefficients are quantised (i.e., they are rounded off to appropriate levels. These are often tailored to human psychovisual characteristics in order to minimise obvious distortion). The quantisation scheme reduces the precision of the representation of the raw data and is a lossy and irreversible process. The statistical properties of this quantised stream of data are such that entropy encoding can be used to provide effective compression. Source (entropy) encoding is a lossless and reversible procedure.

The compressed data can then be efficiently stored or transmitted over a channel.² In order to recover the original data, reconstruction takes place by applying the above procedure in reverse order. The linear transform portion of the lossy compression setup is the key step in producing good results. The idea here is to pack the majority of the input signal's energy into a small number of coefficients.

¹ATA hard-drives over a PCI bus is the typical scenario. The peak theoretical throughput is 100MBps with Ultra-ATA100, but only a tiny fraction of that is achieved on the type of sustained random accesses that digital video involves.

²The "channel" can be conceived of either in time or space, for storage or transmission respectively.

3 Wavelets

3.1 Why Wavelets?

Wavelets are ideal for processing natural images and outperform traditional DCT compression, as used in the prevailing JPEG standard, for four main reasons. First, they have non-uniform frequency spectra which facilitate multiscale analysis. Second, the multiresolution property of the wavelet transform can be used to exploit the fact that the human eye's capacity to detect noise deteriorates at high and low frequencies. This is because most of the energy of wavelet-transformed data is concentrated in the low frequency region. Third, the DWT can be applied to an entire image without imposing a block structure as used by the DCT, so there is less distortion. Fourth, the multitude of ever-more sophisticated compression strategies in the DWT domain produce better results than quantisation in the DCT domain.

Wavelets have attracted a great deal of both academic and commercial interest because their properties are simultaneously of theoretical and practical significance. The analytical fascination is explored in the section below. The practical use of wavelets is demonstrated by their incorporation into the emerging JPEG2000 standard for image compression³ and the MPEG-4 video coding standard.⁴

The DWT is a fast, linear, invertible and orthogonal operation, like the DCT, yet it has displaced the dominance of the DCT because techniques for compression in the DWT domain outperform compression by quantisation in the DCT domain. The DWT is useful in image processing because it can simultaneously localise signals in time and scale, whereas the discrete Fourier transform (DFT) or DCT localise only in the time, frequency or DCT domains. The foundation for all signal-processing techniques rests on the fundamentals of functional analysis and the theory of Hilbert spaces. The key point, however, is that the DWT decomposes an input signal into low frequency ("average" or "coarse") and high frequency ("detail") components as shown in Figure 2. The DWT can be practically implemented by convolving an input signal with a pair of low- and high-pass filters ("quadrature mirror filters") and downsampling by 2. Recursive DWT decomposition of a signal yields an invertible, multilevel result (different "scales" or "sub-bands" are observable). So a signal can be reconstructed by recursively applying the IDWT and upsampling by 2. For perfect reconstruction of a transformed signal, the decomposition ("analysis") filters $h(z)$ and $g(z)$ and reconstruction ("synthesis") filters $\underline{h}(z)$ and $\underline{g}(z)$ must satisfy [26]:

$$\underline{h}(z)\underline{g}(z) + h(z)g(z) = 2 \quad (1)$$

Different families of wavelets exist, the most well known being the Haar and Daubechies wavelets. Two-dimensional DWT's and IDWT's are formed by successively applying one-dimensional transforms to the rows and columns of an image

$$W_{n \times m}(x[n, m]) = W_n(W_m(x[n, m])) \text{ where } W \text{ is the DWT operator.} \quad (2)$$

³Lossy JPEG2000 uses the biorthogonal 9/7 DWT filter pair and quantises the wavelet coefficients with a different step-size for each sub-band.

⁴The wavelet transform is used for static texture coding.

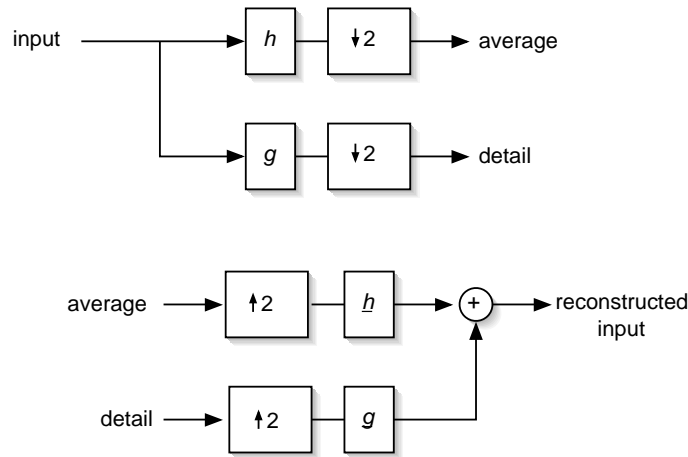


Figure 2: Diagram of 1-D decomposition and reconstruction with down/up-sampling.

This is illustrated in Figure 3. The DWT is composable in this way if we choose separable 2-D filters.

4 Wavelet Compression Algorithms

In our progress report, we explored a host of efficient compression algorithms that have been developed to exploit the properties of the wavelet transform. These include Embedded Zero-tree Wavelet (EZW) coding, Set Partitioning in Hierarchical Trees (SPIHT), Iterated Function Systems on Wavelet trees (IFSW), Footprints, and Estimation Quantisation (EQ) and derivatives. For further details of these techniques, please see the descriptions given in our earlier report. The formulae typically used to evaluate image compression algorithms are as follows:

Mean-Square Error:

$$D = \frac{1}{N} \sum_{i=0}^{N-1} \|x_i - \hat{x}_i\|^2 \quad (3)$$

SNR:

$$SNR = 10 \log_{10} \left(\frac{\sigma^2}{D} \right) \quad (4)$$

PSNR:

$$PSNR = 10 \log_{10} \left(\frac{M}{D} \right), \quad M = \max\{255, \max \|x_i\|\} \quad (5)$$

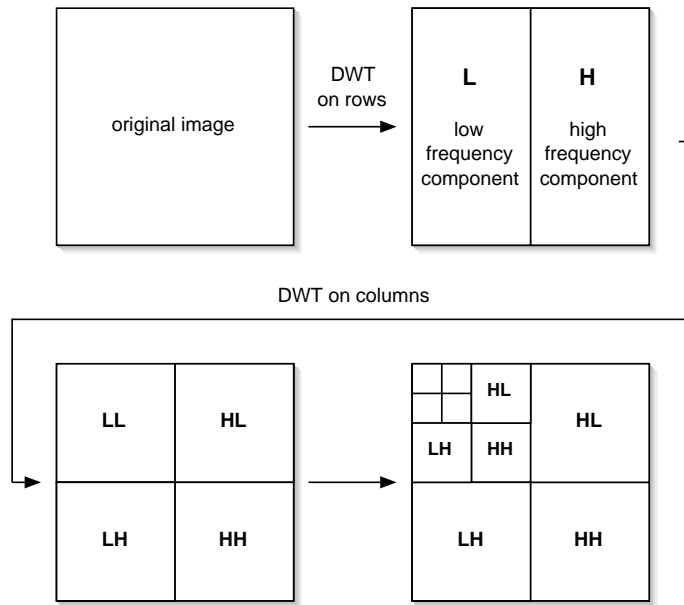


Figure 3: Diagram of 2-D (3-level) wavelet decomposition.

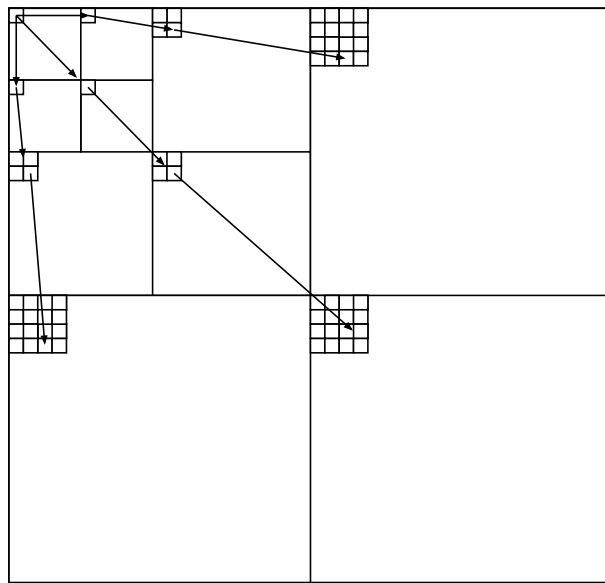


Figure 4: The hierarchical structure of a 2-D wavelet decomposition.

4.1 Space Frequency Quantisation

We decided to implement the Space-Frequency Quantisation (SFQ) algorithm for three main reasons. First, it takes advantage of the two distinctive features of wavelet multiresolution analysis: simultaneous localisation in both space and frequency. Second, SFQ is intuitive and well tailored to the wavelet decomposition because it is based on the tree structure of wavelet coefficients. The approach is provably locally optimal with respect to three variables: the final subtree (or “data map”) chosen, the quantiser used, and the operating point on the rate distortion curve. Fundamental to the algorithm is a cost inequality which decides whether to zero out coefficients depending on whether the data gain (i.e., reduced bit rate) from discarding them outweighs the energy (and hence image quality) gain of retaining them. Third, SFQ yields 1dB better PSNR performance than the reference implementation of SPIHT, the current benchmark algorithm in wavelet image compression.

SFQ is competitive with and often outperforms the best image coding algorithms in the literature⁵ [13]. This relatively simple, intuitive algorithm is our preferred candidate for project implementation. SFQ capitalises on the property of natural images clearly emphasised by a multi-level DWT decomposition: most of the energy is low-frequency information and the remaining high-frequency coefficients are larger around edges (i.e., texture and discontinuities) in an image. SFQ uses the hierarchical signal representation of the wavelet transform as a framework for exploiting statistical dependencies within an image and for selecting a quantisation strategy tailored to human psychovisual characteristics.

Two basic quantisation modes are used. First, zerotree quantisation targets the spatial compaction properties of wavelets by zeroing out trees of small wavelet coefficients and hence highlighting regions in which larger high-frequency coefficients are clustered. Second, coefficients which are not set to zero are uniformly scalar quantised⁶ within their level in the wavelet decomposition (since the variance of each frequency band is different). Despite its simplicity, uniform scalar quantisation is highly efficient because, after zerotree quantisation, the remaining coefficients are distributed fairly evenly near zero. Coupled with entropy coding, uniform quantisation achieves nearly optimal bit allocation and coding efficiency. Two key questions arise from the SFQ paradigm explained above: which spatial subsets of coefficients should be zeroed and what step-size should be used to quantise the survivors? Since each decision affects the other, their interdependence is resolved by iterative optimisation. Ultimately, this process selects the best binary zerotree “map” and quantiser. Together, the map and quantised survivor coefficients (“data” bits) form the basis for reconstruction.

SFQ minimises distortion for a given rate constraint. This goes beyond the tactic of schemes like EZW which only minimise distortion of the overall coding scheme. SFQ finds the optimal tradeoff of bits for distortion using zerotree spatial quantisers⁷ and scalar frequency quantis-

⁵Note that SFQ’s high PSNR for a given compression ratio must be taken with a grain of salt. PSNR is not an exhaustive measure of reconstructed image quality. For example, it may not take sufficient account of blurring caused by infidelity to detail.

⁶It is also possible to use non-uniform quantisation as well as vector quantisation.

⁷These reduce entire sets of coefficients to zero without significant debt to the “bit budget”.

ers.⁸ The approach is incremental. First, assuming the quantiser and R-D curves are fixed, trees are pruned by moving from leaves to roots until spatial subtrees are formed whose data cost is minimal in the rate-distortion sense. An optimal quantiser can then be chosen by exhaustively searching choices in a finite admissible list. The optimal operating point on the R-D curve is found using the bisection algorithm to successively shrink the interval containing the optimal operating point until convergence. The three optimisations described above are concisely explained by the following formula:

$$\max_{\lambda \geq 0} \min_{q \in Q} \min_{S \leq T} [D(q, S) + \lambda(R(q, S) - R_{budget})] \quad (6)$$

where R_{budget} is the bit budget, S is the optimal subtree, q is the best quantiser, $R(q, S)$ is the associated bit rate, $D(q, S)$ is distortion and λ is the optimal operating point. λ is also known as the Lagrangian multiplier, since it is the quality factor trading off distortion for rate in the Lagrangian cost function $J(q, S) = D(q, S) + \lambda R(q, S)$ ($\lambda = 0$ refers to the highest attainable quality and $\lambda = \infty$ to the lowest attainable rate).

5 Software

What development languages did we use?

Our software development was done primarily in C++, though diagnostic programs were written in Objective-C and auxiliary development was done in other languages.⁹ These languages were chosen because they support high-level abstract constructs including classes, parameterised data types and exception handling, yet they also cater for low-level coding for speed. In other words, C++ and inline AltiVec assembly language intrinsics (C interface to mnemonics) combine the best of both worlds. Specifically, the C++ standard template library (STL) allows the concise representation of sophisticated constructs including hash-tables, associative containers and flexible memory allocators. Motorola's AltiVec is a Single Instruction Multiple Data (SIMD) machine designed to improve the performance of any application that can exploit data parallelism, hence it is ideal for multimedia signal processing. What this means is that multiple pixels can be processed with a single instruction rather than processing one pixel at a time. The routines written using AltiVec intrinsics perform very well on the PowerPC 7450. However, the drawback with using AltiVec in our DWT and IDWT codec is our software is platform-dependent, since AltiVec is specific to the PowerPC architecture. If our codec were extended for broader applications, this DWT and IDWT could be re-written in another language to allow wider deployability on other platforms. We decided to use it in our pilot implementation for its high speed and perfect suitability to the developing platform we used as a result of receiving an AUDF Grant.

We used a combination of Apple's ProjectBuilder, InterfaceBuilder and Metrowerks Codewarrior 8.3 for various parts of the development process. For the QuickTime component itself,

⁸These perform the exchange in proportion to their step-sizes.

⁹In addition, we used some in-line assembly language to speed up processes such as taking square roots and floating point reciprocals in the quantiser calculations.

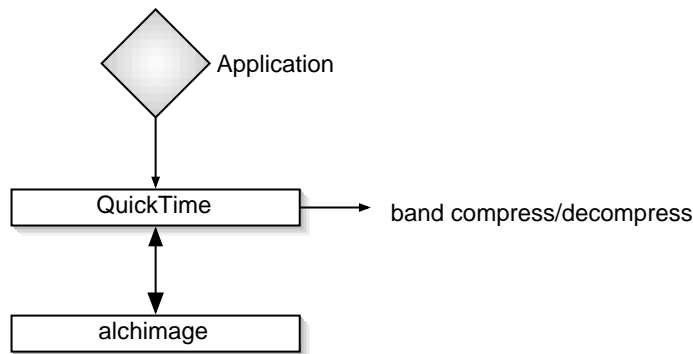


Figure 5: Communication between QuickTime and user-applications.

Codewarrior was found to produce the fastest optimised code but the Metrowerks implementation of the STL leaves something to be desired.

Where does our software sit?

Our software (i.e., the compressed image format we have defined and labelled “alchimage”) is a QuickTime component (shared library). Since it is embedded in QuickTime, it is immediately understood by any QuickTime-literate application (i.e., Microsoft Word, etc.). The diagram in Figure 5 describes how our software uses QuickTime as an interface for compressing and decompressing images and sequences of images that is independent of devices and algorithms. When an application asks QuickTime to compress or decompress an image and “alchimage” is selected as the desired codec, QuickTime calls our software on behalf of the application.

The broad reach of QuickTime means our image format has correspondingly vast scope. It ensures that “alchimage” has interoperability with a wide variety of video and still-picture formats and allows us to work seamlessly with real video and image-editing applications. This was a useful accomplishment even from the point of view of making advancements during the development of our project, since it meant we could see our progress visually.

6 Implementation

Figure 6 illustrates the way the different components in our image compression scheme fit together. Each of the procedures involved in compression (i.e., drawn as modules connected by arrows pointing to the right) is described below. Adjustments specific to the inverse process of decompression (i.e., in modules connected by arrows pointing to the left) are also highlighted.

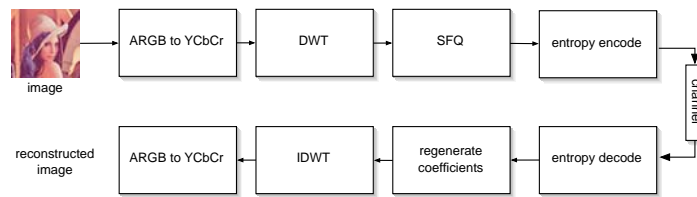


Figure 6: Interaction of components in our SFQ implementation.

6.1 ARGB to YCbCr Conversion

“Alchimage”’s first step is to convert input ARGB (Alpha, Red, Green Blue) data to the YCbCr (Luminance, Chrominance) color space. The conversion was efficiently and accurately implemented using Altivec instructions. The YCbCr domain is the most commonly used color coordinate system as it is useful for compatibility with monochrome video and interoperability with professional video processing equipment. Luminance (Y) contains most of the spatial information to which the human eye accords importance due to its sensitivity to detail. The chrominance channels (Cb, Cr, blue-difference and red-difference respectively) merely add color, so sub-sampling by up to a factor of two in the horizontal and vertical dimensions of an image results in meagre discernible loss of quality. In our project, for efficient processing, we sub-sample and use the pre-defined interleaved pixel format CbYCrY to represent each two adjacent pixels with only one set of chrominance data, a so-called 4:2:2 YCbCr pixel format. Native digital video (MiniDV and DV) are captured in this pixel format. This yields an initial primitive compression ratio of 3 : 2 over the ARGB pixel format.

Having converted the input data to the YCbCr color space, it is transformed to the wavelet domain, processed and stored. At some later time, a user will want to reconstruct the compressed data. Ordinarily, the data must be decompressed back to ARGB pixel format and placed in the frame-buffer, ready to be displayed on the screen. However, we avoid the final conversion from YCbCr to ARGB pixel format by writing YCbCr data directly to the frame-buffer and leaving the conversion to ARGB to be done in hardware by the graphics card. The process of writing data to the frame-buffer is called blitting. By blitting our YCbCr pixel data straight to the frame-buffer, we improve the efficiency of our software.

6.2 Discrete Wavelet Transform

“Alchimage” performs the DWT (and IDWT) using the Daubechies D4 compact wavelet. The implementation of this wavelet is simpler than the 9/7 biorthogonal Daubechies filters with which we experimented at the outset of the project because the filters are shorter (length 4 compared with 9 and 7).¹⁰ Each D4 convolution sum is extremely efficient, requiring only three vector operations per pixel.

¹⁰See explanation in our Progress Report.

6.3 Space Frequency Quantisation

Our project implements a modified version of the SFQ algorithm announced by Xiong *et al* in [13]. As explained below, our modifications improve speed and efficiency while preserving excellent results. Hence, the “proof is in the pudding” in terms of justifying the legitimacy of these variations.

The SFQ algorithm works by tailoring compression to suit a prescribed bit rate. This rate is specified by the user indirectly by indicating an image quality. For example, QuickTime presents the user with quality settings of “Best”, “High”, “Medium”, “Low” and “Least” and percentage values on a sliding scale between these settings. What these settings do in terms of bit rate is codec-dependent but in our case, for a quality of $Q\%$ and a rate of r bpp, we had

$$r = \frac{(3.6 - 0.1) \times Q}{100} + 0.1. \quad (7)$$

6.4 Optimal Subtree

Zerotree pruning tackles the innermost optimisation and finds the best subtree for a given quantiser and operating point. Once a decision is made to prune or not to prune, we proceed up the tree to parent nodes i rooted in level 1 and the coarsest level and evaluate the same inequality each time. Having completed this iteration, the next iteration begins by updating the probabilities and again applying the cost inequality to successive levels. Theoretical convergence is when no pruning occurred in an iteration, since the unchanged subtree is optimal.

6.5 Optimal Quantiser

Xiong *et al* point out the efficiency of simple, uniform scalar quantisation of retained wavelet coefficients. In our preliminary analysis of the quantisation process, we plotted a histogram of the wavelet coefficients to better understand the data distribution. We found that for natural images, coefficients range in size from about -8000 to $+8000$ and followed an approximately Laplacian distribution. This suggested to us that, although Xiong *et al* conclude that uniform scalar quantisation is adequate, we could achieve better performance by capitalising on the additional information we observed about the wavelet distribution for natural images. Experimentation with different uniform and non-uniform quantisers revealed that the best results came from non-uniform quantisation of the luminance coefficients (though, as mentioned above, we still used uniform scalar quantisation on the retained chrominance coefficients). It is worth noting that, although the luminance and chrominance channels are quantised independently using different methods, a linear relationship between the two quantisers ensures they operate at the same point on the rate-distortion curve.

6.6 Optimal Operating Point on the Rate-Distortion Curve (λ)

Operating point optimisation means choosing the optimal λ by selecting a quantiser and subtree to match a given bit budget. After examining various heuristic techniques for achieving an approximately optimal SFQ operating point, such as constrained linear optimisation using the simplex algorithm of the Golden search method described in [12], we opted for a variant of the fast bisection (Newton-Raphson) method discussed in [22]. The bisection method starts with a wide interval of the rate-distortion curve engulfing the desired operating slope. The most conservative choice for initialisation is 0 to ∞ , though a more judicious choice yields faster convergence. We found that a lower bound of $\lambda_l = 0$ and an upper bound of $\lambda_u = 16$ was adequate to ensure that

$$\sum_i R_i^*(\lambda_u) \leq R_{budget} \leq \sum_i R_i^*(\lambda_l). \quad (8)$$

If the above inequality is an equality for either λ value, the exact solution has been found. Otherwise, the search intervals are then made successively smaller, exploiting the convex relationship of both the global rate and global distortion with respect to the operating slope λ . The intervals are shrunk by defining a new λ value (λ_{next}) as the slope of the line joining the two former operating points (λ_l and λ_u) on the rate distortion curve:

$$\lambda_{next} = \frac{|\sum_i [D_i^*(\lambda_l) - D_i^*(\lambda_u)]|}{[R_i^*(\lambda_l) - R_i^*(\lambda_u)]} + \epsilon \quad (9)$$

where ϵ is a vanishingly small positive number which ensures that λ_l is selected if λ_{next} is a singular slope value.¹¹

6.7 Drawing Code

Our implementation of a compression is done at the level of the Image Compression manager portion of QuickTime. That is, the software is a service provider for layers higher up and a client of QuickTime and the kernel. Specifically:

1. Applications interact with our component only through QuickTime.
2. QuickTime defines an interface that components must implement in the form of selectors or call-backs. These are essentially functions that QuickTime expects to be named in a certain way and to take specific arguments.
3. For compression, QuickTime will supply a buffer containing a frame in the requested pixel format and expects the compressed data to be written to a pre-allocated buffer of a fixed size. The size of the buffer that QuickTime allocates is dependent on what the component (our software) suggests will be a maximum size. The actual size is then returned to QuickTime once the compressed data is written.

¹¹For a detailed treatment of singularities, see [25].

4. On the decompression end, QuickTime supplies a buffer to the called component function containing the compressed data stream as well as a pointer to a graphics buffer.
5. In our case, as the output pixel format is 2vuy (4:2:2 YCbCr), the buffer is actually mapped to device memory on the graphics card and composited¹² there with no software intervention.

6.8 Compression of Successive Video Frames

In addition to compressing individual images, our software is able to compress sequences of images as a consequence of being embedded in QuickTime. We have experimented with a host of films, including DVD excerpts and short movies we shot ourselves with a digital camera. The results are pleasing, with high image quality being preserved as well as coherent synchronisation of the audio and moving pictures. As our software does not currently perform intraframe compression, the quality exceeds MPEG-4 video streams but produces streams that are larger. Figure 7 demonstrates none of the block motion-estimation artifacts associated with intraframe compression algorithms. We did not focus on this aspect of the product as video is a significant undertaking in its own right.



Figure 7: Part of a contiguous sequence of frames captured on digital camera and compressed with our codec.

7 Performance Evaluation

There are a multitude of different ways of assessing the performance of image compression software. These include both subjective and objective analyses of the reconstructed image. The former depends on a qualitative judgement of how the image looks to a human observer, while

¹²Copied over pixels already on the screen and possibly blended should a rectangle containing an alpha channel be covering or partially covering the rectangle containing the image or sequence.



Figure 8: Left: Lena compressed with alchimage at 1.75bpp. Right: Original Lena image.

the latter can be measured quantitatively in terms of RMSE and PSNR, bit rate and computational speed. Additionally, the performance of “alchimage” can be evaluated by comparing it with other QuickTime-based image compression software.

Our results begin with the classic reference: Lena. During the design, implementation and testing of our software, we compressed a number of traditional images such as Lena, Goldhill and Barbara. A critical selection of our results, together with statistics we gathered, are presented in Figures 8 to 15. We achieve better image quality at higher compression ratios for uncompressed photographs taken with the digital camera we were loaned by the Apple University Consortium.

7.1 Subjective Results

Clearly, the subjective quality of our reconstructed images depends on the bit rate at which they are compressed. For high bit rates, “alchimage” compresses and decompresses images with high fidelity to the originals. At lower bit rates, there is more noticeable distortion such as blurring around originally clean edges and corruption of smooth regions by artefacts. These properties are demonstrated by the sample images included. It is important to reiterate that the way that a person perceives the quality of a reconstructed image depends on human psychovisual characteristics. As the eye is more sensitive to detail, blockiness which compromises the outlines in an image is more obvious and disconcerting than less noticeable background variations from the original such as inaccuracy to color. Since the eye prejudices the corruption of some aspects of an image more than others, and in order to have a more precise measure of fine discrepancies between original and reconstructed images (which the eye may not appreciate), we consider the following objective analysis methods.



Figure 9: Left: Leaves compressed with alchimage at 1.75bpp. Right: Original Leaves image.

7.2 Objective Results

PSNR and RMSE (root mean square error) offer a more objective way to compare various algorithms' performance with formulae for these metrics given in Section 5. Bit rates and PSNR values have been provided in Figures 13 to 15. The rate distortion curve for the Lena image is shown in Figure 12.

These are nominal values for bit rates of the various codecs that illustrate what might constitute typical operation in a desktop computing environment. They do not, and are not, meant to be asymptotic results and represent a sample of the data collected. They pit our software against other QuickTime-based encoders which included licensed, commercial implementations of MPEG-4, JPEG and JPEG2000. We have also experimented with MPEG-2 briefly. This is done to contextualise our results and to make a fair assessment of our software relative to other practical implementations that operate in full color (not just black-and-white). The difficulty with such comparisons is that different compression software is tailored to meet different standards, to suit different applications, and to perform optimally in different respects. Nonetheless, despite the uniqueness of the aforementioned codecs, it is instructive to situate "alchimage" alongside its counterparts because they provide a kind of measuring stick.

What the results indicate is that our codec outperforms JPEG and MPEG-4 in many scenarios in terms of both bit rate and PSNR. JPEG has long been the workhorse lossy compression algorithm and any newcomer must fare as good as JPEG. We were particularly pleased to have achieved this across every statistic that we gathered comparing our codec to JPEG. MPEG-4 represents the current state of the art in motion-compensated video but uses wavelet-based techniques to code static components of a stream including still frames. In that sense, we have not disadvantaged MPEG-4 in selecting still images for some comparisons. However, that it is a video codec and not an image codec is apparent as we do significantly better. Although statistics were not gathered, we tested against MPEG-2 for subjective quality. Nowadays, MPEG-2 has found a niche as the standard codec for video playback on DVDs in a high bit rate environment



Figure 10: Left: Goldhill compressed with alchimage at 0.875bpp. Right: Original Goldhill image.

where little motion compensation is needed. At these rates, our codec exceeds MPEG-2's PSNR but we elected not to pursue the investigation further as MPEG-2 is very restrictive on sequence formats, insisting the streams are captured at standard 16:9 PAL or NTSC aspect ratios.

Figure 12 shows our codec's performance against the current state of the art *image* compression standard, JPEG2000. We find that JPEG2000 fares better at low bit rates (its target operating environment) but exhibits peculiar behavior at higher bit rates. This was part of the difficulty in assessing our codec's performance as arbitrary compression algorithms do not match arbitrary bit rates. In the case of straight JPEG, bit rate constraints are not offered by all implementations. In the case of JPEG2000, the implementation on QuickTime 6.3 will match any target output file size but the PSNR actually starts to drop when the rate is higher than 2.0 bps. The JPEG2000 standard guarantees nothing about achieving a certain PSNR for a given bit rate but only specifies that any implementation will match a rate. It is also notable that reference implementations of JPEG2000 do not achieve such high PSNRs for the Lena image, but nevertheless, even against the optimised implementation in QuickTime our codec performed well in the circumstances. However, JPEG2000 pays for its superior bit rate in decoding complexity. Under OS X 10.2.6, JPEG2000 is unable to playback a full screen full frame rate sequence in realtime and maintain audio-synchrony while our codec can do so with ease.

At the time of writing, "alchimage" fared favorably with Apple's best-of-class wavelet-based codec, Pixlet.



Figure 11: Left: Barbara compressed with alchimage at 0.875bpp. Right: Original Barbara image.

8 Conclusion

“Alchimage” is an innovative, functional product that competently stands proud next to its fore-runners. It contributes to the pursuit of better solutions to the image compression dilemma in an age when efficient multimedia applications have growing clout and are in higher demand. That two undergraduates, starting from scratch, have finalised a fully operable, well-rounded piece of software in a matter of months is no mean feat. That the software comprises the latest technology in its field and is competitive with commercial grade image compression formats is a remarkable achievement.

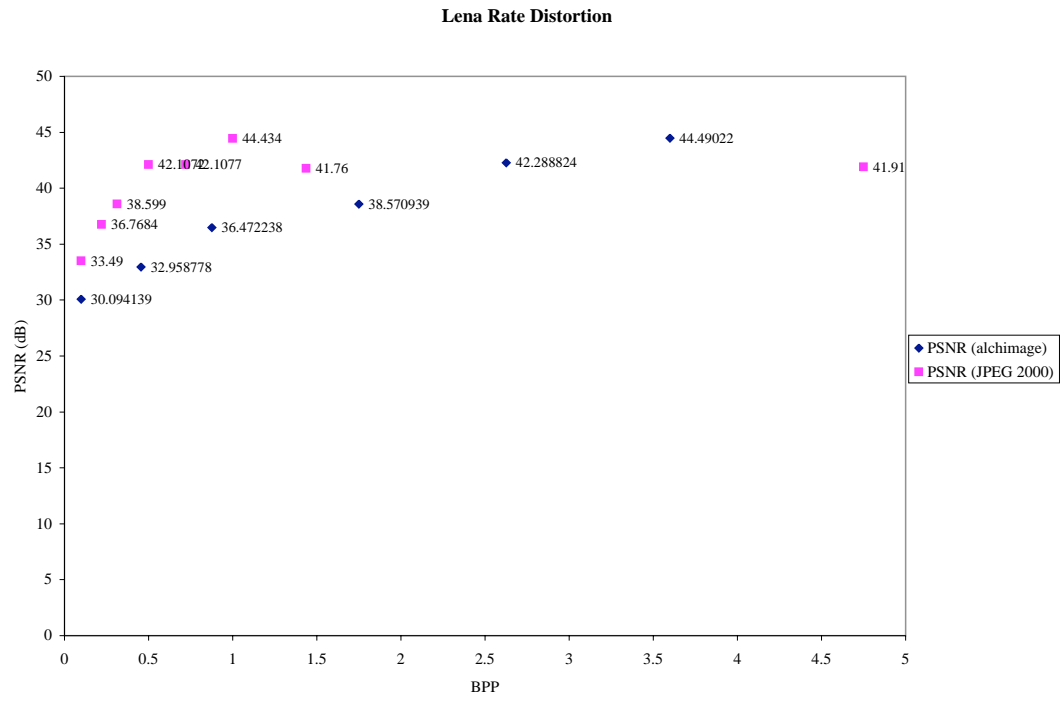


Figure 12: Rate distortion curve for Lena image for alchimage and JPEG2000 algorithms.

| Codec | BPS | PSNR (db) |
|-----------|-------|-----------|
| alchimage | 3.6 | 40.72 |
| alchimage | 1.85 | 36.43 |
| alchimage | 0.975 | 30.82 |
| alchimage | 0.1 | 25.50 |
| MPEG-4 | 2.0 | 35.52 |
| JPEG | 3.25 | 31.94 |

Figure 13: Statistics for Barbara image.

| Codec | BPS | PSNR (db) |
|-----------|-------|-----------|
| alchimage | 3.6 | 40.99 |
| alchimage | 1.85 | 36.46 |
| alchimage | 0.975 | 32.94 |
| alchimage | 0.1 | 26.41 |
| MPEG-4 | 2.0 | 36.10 |
| JPEG | 3.25 | 32.00 |

Figure 14: Statistics for Goldhill image.

| Codec | BPS | PSNR (db) |
|-----------|-------|-----------|
| alchimage | 3.6 | 45.53 |
| alchimage | 1.85 | 40.21 |
| alchimage | 0.975 | 36.44 |
| alchimage | 0.1 | 30.08 |
| MPEG-4 | 2.0 | 35.99 |
| JPEG | 3.25 | 32.24 |

Figure 15: Statistics for Lena image.

References

- [1] APPLE DEVELOPER CONNECTION <http://developer.apple.com/>
- [2] ALEXANDER S.K. (2001) *Two- and three-dimensional coding schemes for wavelet and fractal-wavelet compression*. Master's thesis, Dept. of Mathematics, University of Waterloo, Ontario.
- [3] ANTONINI M., BARLAUD M., MATHIEU P. AND DAUBECHIES I. (1992) *Image coding using wavelet transforms* IEEE Trans. Image Processing **1** 205-220
- [4] BHASKARAN V. AND KONSTANTINIDES K. (1997) *Image and video compression standards: algorithms and architectures*. Kluwer.
- [5] CLARKE R.J. (1995) *Digital compression of still images and video*. Academic Press.
- [6] CRANDALL R. AND KLIVINGTON J. (1999) *Vector implementation of color-image wavelet transform* Technical report, Advanced Computation Group, Apple.
- [7] DAVIS G. (1995) *Adaptive self-quantization of wavelet subtrees: A wavelet-based theory of image compression*. SPIE Conference on Mathematical Imaging: Wavelet Applications in Signal and Image Processing. San Diego.
- [8] DEVORE R.A., JAWERTH B. AND LUCIER B.J. (1992) *Image compression through wavelet transform coding* IEEE Trans. Inform. Theory **38** 719-746.
- [9] DRAGOTTI P.L. AND VETTERLI M. (2001) *Wavelet footprints: Theory, algorithms and applications*. IEEE Trans. Signal Processing.
- [10] BARLAUD M. (ed). (1994) *Wavelets in image communication*. Elsevier.
- [11] EFFORD N. (2000) *Digital image processing: a practical introduction*. Addison-Wesley.
- [12] PRESS W.K. ET AL. (1998) *Numerical Recipes in C: The Art of Scientific Computing* Cambridge University Press.
- [13] ZIXIANG XIONG RAMCHANDRAN K. AND ORCHARD M. (1996) *Space-frequency quantization for wavelet image coding*. Technical report, Dept. of Electrical Engineering, Princeton University.
- [14] GHANBARI M. (1999) *Video coding: an introduction to standard codecs*. Institution of Electrical Engineers.
- [15] LAWSON S. AND ZHU J. (2002) *Image compression using wavelets and jpeg2000: a tutorial*. Electronics and Communication Engineering Journal.
- [16] JIN LI AND KUO C.C.J. (1999) *Image compression with a hybrid wavelet-fractal coder*. IEEE Trans. On Image Processing, **8** (6).
- [17] XIN LI AND ORCHARD M.T. (1999) *Rate-distortion optimized image coding via least square estimation quantization (ls-eq)*. Technical report, Princeton University.
- [18] MOTOROLA (1999) *AltiVec Technology Programming Interface Manual*.
- [19] MOTOROLA (2001) *MPC7450 RISC Microprocessor Family User's Manual*.
- [20] PITAS I. (2000) *Digital image processing algorithms and applications*. Wiley.

- [21] PRATT W.K. (2001) *Digital image processing*. Wiley.
- [22] RAMCHANDRAN K. AND VETTERLI M. (1998) *Best wavelet packet bases in a rate-distortion sense*. IEEE Transactions on Image Processing **2** 160-175.
- [23] SAID A. AND PEARLMAN W.A. (1996) *A new fast and efficient image codec based on set partitioning in hierarchical trees*. IEEE Trans. On Circuits and Systems for Video Technology **6** .
- [24] SHAPIRO J.M. (1993) *Embedded image coding using zerotrees of wavelet coefficients* IEEE Trans. Signal Processing **41** 3445-3462.
- [25] SHOHAN Y. AND GERSHO A. (1998) *Efficient bit allocation for an arbitrary set of quantizers* IEEE Transactions on Acoustics, Speech and Signal Processing **36** 1445-1453.
- [26] VETTERLI M. (1986) *Filter banks allowing perfect reconstruction* IEEE Trans. Signal Processing **10** 219-214.
- [27] VETTERLI M. AND KOVACEVIC J. (1995) *Wavelets and Subband Coding*. Prentice Hall Signal Processing Series.
- [28] VRSCAY E. (1995) *A hitchhiker's guide to 'fractal-based' function approximation and image compression* Math Ties.
- [29] Vrscay E. (1998) *A generalized class of fractal-wavelet transforms for image representation and compression* Canadian Journal of Electrical and Computer Engineering.