Fast Weighted Universal Transform Coding: Toward Optimal, Low Complexity Bases for Image Compression

Michelle Effros
Department of Electrical Engineering (136-93)
California Institute of Technology
Pasadena, CA 91125
effros@caltech.edu

Abstract

In [1], Effros and Chou introduce a two-stage universal transform code called the weighted universal transform code (WUTC). By replacing JPEG's single, non-optimal transform code with a collection of optimal transform codes, the WUTC achieves significant performance gains over JPEG. The computational and storage costs of that performance gain are effectively the computation and storage required to operate and store a collection of transform codes rather than a single transform code. We here consider two complexity- and storage-constrained variations of the WUTC. The complexity and storage of the algorithm are controlled by constraining the order of the bases. In the first algorithm, called a fast WUTC (FWUTC), complexity is controlled by controlling the *maximum* order of each transform. On a sequence of combined text and gray-scale images, the FWUTC achieves performance comparable to the WUTC at 1/32 the complexity for rates up to about 0.10 bits per pixel (bpp), 1/16 the complexity for rates up to about 0.15 bpp, 1/8 the complexity for rates up to about 0.20 bpp, and 1/4 the complexity for rates up to about 0.40 bpp. In the second algorithm, called a jointly optimized fast WUTC (JWUTC), the complexity is controlled by controlling the *average* order of the transforms. On the same data set and for the same complexity, the performance of the JWUTC always exceeds the performance of the FWUTC. On the data set considered, the performance of the JWUTC is, at each rate, virtually indistinguishable from that of the WUTC at 1/8 the complexity. The JWUTC and FWUTC algorithm are interesting both for their complexity and storage savings in data compression and for the insights that they lend into the choice of appropriate fixed- and variable-order bases for image representation.

1068-0314/97$10.00 © 1997 IEEE

I Introduction

In data compression, there exists a tradeoff between complexity and performance. To achieve the best possible rate-distortion performance, we must encode using large data vectors. Yet the complexity of a true vector code grows exponentially with the vector dimension. Transform coding represents a compromise between the performance gains achievable with high dimensional data compression and the low complexity of scalar codes. In transform coding, an incoming data set is typically coded in large vectors (e.g., 8×8 blocks of pixels). Each of the blocks undergoes a reversible *transformation* designed to decorrelate the components of each vector. The resulting transformed data is then quantized using a collection of *scalar quantizers* and each quantized component is described using an *entropy code.* Unfortunately, the optimal transform, scalar quantizers, and entropy codes are data dependent. Thus the "optimal" transform code for a given data set is the transform code whose transform, quantizers, and entropy code have been matched to the statistics of the data to be compressed.

In theory, the optimal image coding performance would be achieved if we could always use – free of charge – a transform code designed to match the statistics of the current data block to be compressed. In practice, however, optimization of the transform code to match the statistics of each data block comes at a price. The price of this optimization is paid in a combined currency of computational complexity – to design and / or choose the appropriate transform code for each data block – and rate – to describe the chosen transform code to the decoder. Until recently, this cost was thought so prohibitive that the vast majority of transform codes fixed most or all of the transform code's components at design time. For example, in JPEG, the transform is fixed a priori to be the discrete cosine transform (DCT). Further, while the JPEG algorithm allows the optimization of its scalar quantizers and entropy code (as described by the quantization matrix and Huffman tables) to match the statistics within a given image, many implementations of JPEG forgo this option in favor of a single default quantization matrix / entropy code pair. The resulting default transform code, built into many implementations of JPEG, is designed to do well on average across the class of images anticipated by the algorithm's designers. However, even with an optimized quantization matrix and entropy code, the JPEG algorithm will not achieve the performance that would be achieved if each data block of the image were quantized with a transform code truly matched to that data block's statistics.

Attempts at improving the performance of transform codes have primarily involved "adaptive" techniques, whereby the transform code is modified during the coding process to match the statistics of a given image. The resulting techniques are vulnerable to a range of pitfalls. At the two extremes of this range are codes that achieve good rate-distortion performance at prohibitive computational expense and codes that use reasonable complexity but achieve inferior performance.

In [2], Chou, Effros and Gray introduce an alternative technique for negotiating the complexity / performance gap between the high complexity and rate associated with truly matched codes and the low complexity and rate required for practicality.

Effros and Chou generalize the described code to the transform coding domain in [1, 3]. The resulting code, called a weighted universal transform code (WUTC), uses a *fixed collection* of transform codes rather than a single transform code. By replacing a single transform code with a collection of transform codes, the WUTC algorithm allows us to compress each data block with a code matched to the statistics of that data block. By forcing the collection of codes to be fixed, the algorithm removes the on-line design step required of many adaptive codes and reduces the rate required to describe the code in operation. The resulting algorithm is relatively computation and rate efficient. Further, the code achieves good rate-distortion performance. For example, on a sequence of combined text and gray scale images, the WUTC algorithm achieves up to 3 dB performance improvement over both a JPEG-style coder and a single optimal transform code.

While use of a fixed collection of transform codes yields better rate-distortion performance than use of a single transform code, the cost of this performance is the increase in computation and storage associated with using the collection of optimal transforms and quantizers. In particular, the optimal transforms, which lack the regular structure of the DCT, must be stored by both encoder and decoder and operated on each data block.

We here consider two complexity- and storage-constrained variations of the WUTC. In both, the complexity and storage of the algorithm are jointly controlled by controlling the order of the bases in the code's collection of transforms. In the first algorithm, the available complexity and memory are divided equally among the bases. The resulting code is called a fast WUTC (FWUTC). Since the choice of a transform from the WUTC's collection requires that the data be encoded with each code in the collection, the complexity of the WUTC is governed by the *total* order of the bases in its collection. Thus in the second algorithm, we restrict the total order of the bases rather than restricting the order of each basis independently. The resulting jointly optimized fast WUTC (JWUTC) has more degrees of freedom than the FWUTC since it can allow some higher order codes at the expense of other lower order codes and thus achieves better rate-distortion performance.

In Section II we describe the weighted universal transform coding algorithm. Section III contains descriptions of both of the two fast WUTC algorithms. Finally, we summarize and discuss experimental results in Section IV.

II The Weighted Universal Transform Coding Algorithm (WUTC)

Let $x^l = (x_1, \ldots, x_l)' \in \mathcal{X}^l$ represent an l-dimensional data vector, T an $l \times l$ invertible matrix, and $(b^l = b_1 b_2 \ldots b_l)'$ a bit allocation. Suppose that we are given some generic transform coding scheme for encoding a transformed vector Tx^l with b_i bits in the ith dimension. Then associated with any transform / bit allocation pair (T, b^l) is a transform code $C = \beta \circ \alpha$ with encoder $\alpha : \mathcal{X}^l \to \mathcal{S}$ and decoder $\beta : \mathcal{S} \to \hat{\mathcal{X}}^l$ that together map the input space $\mathcal{X}^l$ of possible data vectors to the output space $\hat{\mathcal{X}}^l$ of possible

reproductions by way of a binary prefix code $\mathcal{S}$. Let $d(x^l,(T,b^l)) = d(x^l,\beta(\alpha(Tx^l)))$ be the total distortion achieved by transforming x^l with transform matrix T and then quantizing the resulting transform domain vector with bit allocation b^l. Similarly, let $r(x^l,(T,b^l)) = |\alpha(Tx^l)|$ denote the associated rate. (While $r(x^l,(T,b^l))$ equals $\sum_i b_i$ on average, $r(x^l,(T,b^l))$ will vary with x^l in a variable-rate system.)

We next consider a collection $(T_1,b_1^l),(T_2,b_2^l),\ldots,(T_M,b_M^l)$ of transform / bit allocation pairs. Using the quantization interpretation of a two-stage weighted universal code [2], we consider this collection to be a *codebook* of transform codes. Thus we define a "first-stage quantizer" $\tilde{\beta}\circ\tilde{\alpha}$ with encoder $\tilde{\alpha}:\mathcal{X}^N\to\tilde{\mathcal{S}}$ and decoder $\tilde{\beta}:\tilde{\mathcal{S}}\to\mathcal{C}$ that maps the input space of possible data blocks x^N to the output space $\mathcal{C}$ of possible transform codes (T,b^l). We here assume that N is a multiple of l. The first-stage encoder chooses for each N-block a single transform code. We then use the chosen code to encode each of the l-vectors in x^N. In transform codes like JPEG, N equals the size of a single image. For many applications, we may want smaller N values to allow the transform code to change within a single image. The example of Section IV uses $N=l$, which means that we describe a new transform code for each data block.

Given some block size $N\geq l$, the total distortion associated with encoding data block x^N with transform code $\tilde{\beta}(\tilde{\alpha}(x^N))$ is

$$d(x^N,\tilde{\beta}(\tilde{\alpha}(x^N))) = \sum_{i=1}^{N/l} d(x_i^l,\tilde{\beta}(\tilde{\alpha}(x^N))).$$

The total rate associated with encoding x^N includes both the rate associated with describing the transform code $\tilde{\beta}(\tilde{\alpha}(x^N))$ and the rate associated with using the chosen code to describe the data. Thus

$$r(x^N,\tilde{\beta}(\tilde{\alpha}(x^N))) = |\tilde{\alpha}(x^N)| + \sum_{i=1}^{N/l} r(x_i^l,\tilde{\beta}(\tilde{\alpha}(x^N))).$$

Using a Lagrangian in order to minimize the distortion subject to a constraint on the rate, the optimal first-stage encoder $\tilde{\alpha}^\star$ for a given collection of transform codes $\tilde{\beta}$ is

$$\tilde{\alpha}^\star(x^N) = \arg\min_{s\in\tilde{\mathcal{S}}}[d(x^N,\tilde{\beta}(s)) + \lambda r(x^N,\tilde{\beta}(s))]$$

for every x^N. We call the optimal first-stage encoder a *nearest neighbor* encoder.

Likewise, the optimal first-stage decoder $\tilde{\beta}^\star$ for a given first-stage encoder $\tilde{\alpha}$ satisfies

$$\tilde{\beta}^\star(s) = \arg\min_{(T,b^l)\in\mathcal{C}} E\left[d(X^N,(T,b^l)) + \lambda r(X^N,(T,b^l))\middle|\,\tilde{\alpha}(X^N)=s\right]$$

for every $s\in\mathcal{S}$. We call the process of designing the optimal first-stage decoder *decoding to the centroid.*

Given that the Karhunen Loeve Transform (KLT) maximizes the coding gain over all orthogonal transform codes (e.g., [4, Appendix C]), we here set the transform in the optimal transform code to the KLT matched to the statistics of the

data that mapped to the given code. Using this choice we accomplish the optimal decorrelation and energy compaction for the source in operation. The KLT is calculated as follows. For a given index s, let V_s be the correlation matrix associated with all of the first-stage encoder's input vectors that map to index s, i.e., $V_s = (l/N)\sum_{i=1}^{N/l} E[(X_i^l)(X_i^l)'|\tilde{\alpha}(X^N) = s]$. Then the transform $T_s^\star$ has, in the first row, the eigenvector corresponding to the largest eigenvalue of V_s, in the second row, the eigenvector corresponding to the second largest eigenvalue, and so on.

Given a transform, the optimal bit allocation may be accomplished by an optimal bit-allocation design algorithm. For example, if the individual components of the transform data block are scalar quantized and then described using independent entropy codes, as in the example of [5], then each term in the quantization matrix may be chosen independently to minimize the Lagrangian performance associated with that component.

The WUTC design algorithm employs an iterative descent technique to minimize the expected Lagrangian performance. We initialize the algorithm with an arbitrary prefix code $\tilde{\mathcal{S}}$ and collection $\{\tilde{\beta}(\tilde{s}) : \tilde{s} \in \tilde{\mathcal{S}}\}$ of transform codes. Each iteration proceeds through three steps which we enumerate below.

1 *Nearest Neighbor Encoding.* Optimize the first-stage encoder $\tilde{\alpha}$ for the given first-stage decoder $\tilde{\beta}$ and prefix code $\tilde{\mathcal{S}}$.

2 *Decoding to the Centroid.* Optimize the first-stage decoder $\tilde{\beta}$ for the newly redesigned first-stage encoder and the given first-stage prefix code $\tilde{\mathcal{S}}$.

3 *Optimizing the Prefix Code.* Optimize the first-stage prefix code $\tilde{\mathcal{S}}$ for the newly redesigned first-stage encoder $\tilde{\alpha}$ and decoder $\tilde{\beta}$. The optimal prefix code s^* for a given first-stage encoder $\tilde{\alpha}$ and decoder $\tilde{\beta}$ is the entropy code matched to the probabilities $P\{\tilde{\alpha}(X^N) = s\}$, for which the ideal codelengths are

$$|s^*| = -\log P\{\tilde{\alpha}(X^N) = s^*\}.$$

Each step of the algorithm decreases the expected Lagrangian performance. Since the Lagrangian performance cannot be negative, the algorithm is guaranteed to converge.

III Two Fast Algorithms for Weighted Universal Transform Coding

In going from a traditional transform code to the WUTC, we increase both the storage and computational complexity of the given compression algorithm. The increased storage cost results from the need of both encoder and decoder to store copies of each transform code in the WUTC's collection. Thus a WUTC with 64 transform codes must include descriptions of 64 transforms and 64 bit allocations. Further, description of each transform requires greater storage space than description, for example, of the DCT, due to the fact that the matched KLTs of the previous section lack the regular structure of the DCT. The increased computational costs associated with the WUTC are absorbed almost entirely by the first-stage encoder $\tilde{\alpha}$, which effectively

encodes each data block with every transform code in the collection in order to choose the best-matched code. While *encoding* each data vector x^N with each transform code in the collection represents a significant savings in complexity when compared with algorithms that *redesign* the transform codes during the encoding process, the complexity and storage costs associated with WUTC are nontrivial, and savings in storage and computation are always welcome when they can be achieved with little or no cost in terms of rate-distortion performance.

In this section, we consider two approaches to simultaneously reducing the complexity and storage requirements of the WUTC algorithm. In both, the desired reduction is achieved by reducing the order of the bases described by the transform matrices. That is, we will reduce the ith transform matrix T_i from an $l \times l$ matrix to an $m_i \times l$ matrix. (The transformation T_i has therefore changed from a simple rotation to a rotation and projection of the input data.) As a result of this modification, we reduce the storage and complexity associated with the transformation. Further, since $T_i x^l$ is now an m_i-dimensional vector, we can now reduce our bit allocation from b_i^l to $b_i^{m_i}$. Thus by choosing m_i appropriately, we can significantly reduce the computational and storage costs associated with the algorithm.

In our first attempt at using the above approach, called a fast WUTC (FWUTC), we consider $m_i = c$ for all i, where c is some constant less than or equal to l. If $c = l$, this brings us back exactly to the WUTC. For any $c < l$, we effectively reduce the algorithm complexity and storage to c/l times the WUTC's storage and complexity.

Having chosen a value for c, we must consider how best to design the resulting order-constrained code. A number of alternatives present themselves. Given a value c, the simplest technique would be to add a fourth step to the WUTC algorithm whereby the KLT is simply truncated to discard all but the eigenvectors associated with the c highest eigenvalues. The iterative design procedure could then continue with this fourth step included. Alternative approaches would involve discarding the $l - c$ components that contribute the least to the expected Lagrangian performance, and so on. The performance of the above alternatives turns out to be almost identical. The performance of the first, whereby in every iteration of the WUTC algorithm we simply truncate each KLT matrix from order l to order c, is reported in the section that follows.

While the above algorithm represents a simple technique for reducing the storage and complexity of the WUTC by a desired factor, the algorithm does not take full advantage of the complexity and storage space allotted to it. Simply put, the complexity of the WUTC is not governed by the order of the largest basis in its collection. Rather, since the first-stage encoder must encode using every transform in its collection, the complexity and storage of the WUTC are governed by the total of the orders of all of the bases in the WUTC's collection. Thus, in our second algorithm, we constrain $\sum_i m_i = Kc'$, where K is the number of codes in the collection, but allow $m_i \geq 0$ to vary as a function of i. Like the FWUTC, the resulting algorithm, called a jointly optimized fast WUTC (JWUTC), may be implemented in a number of different ways. In the section that follows, we add to the WUTC design algorithm a fourth step whereby we calculate the contribution of each component to the code's

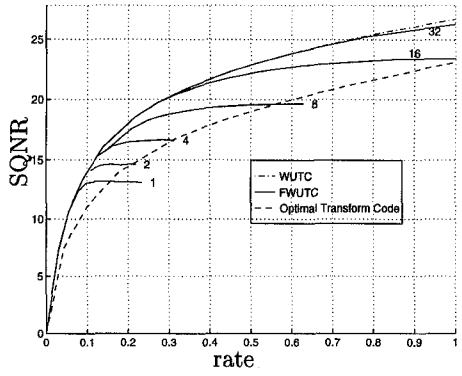

Figure 1: Comparison of SQNR results on a collection of combined text and gray scale images. The systems tested include WUTC, FWUTC at $c = 1, 2, 4, 8, 16, 32$ (labeled), and an optimal transform code. All codes operate on 8×8 data blocks. Both the WUTC and the FWUTC use a collection of 64 transform codes. The storage and complexity associated with a FWUTC at some value of c is roughly equal to $c/64$ times the storage and complexity of the WUTC.

expected rate-distortion performance and keep the Kc' components that yield that greatest benefits from a rate-distortion perspective.

IV Experimental Results

We next consider the performance of the FWUTC and JWUTC as compared to the performance of the WUTC. In all cases, discussions of complexity neglect the complexity associated with training the system. (Training complexity is roughly equivalent in all three cases and is in no case experienced by the system's end-user during actual compression of images.) All transform code implementations discussed here use independent entropy codes to losslessly encode each quantized coefficient. The WUTC, FWUTC, and JWUTC use a maximum of K=64 transform codes. The blocklengths are set at $N = l = 64$ for all experiments. Each system was trained on a single 2048 pixel by 2048 pixel image scanned from a page of *IEEE Spectrum Magazine* and tested on another page from the same issue. Each page had roughly equal amounts of text and gray scale material. All rates are reported as entropies.

In Figure 1, we compare the performance of the FWUTC at a variety of values of c with the performance of the WUTC and the performance of an optimal transform code. (The optimal transform code, which replaces JPEG's DCT with a KLT matched to the data statistics and uses an optimal bit allocation, slightly exceeds JPEG in performance at all rates. For simplicity, the JPEG curve is not included.) As the figure indicates, the FWUTC algorithm achieves performance comparable to the WUTC at 1/32 the storage and complexity for rates up to about 0.10 bits per pixel

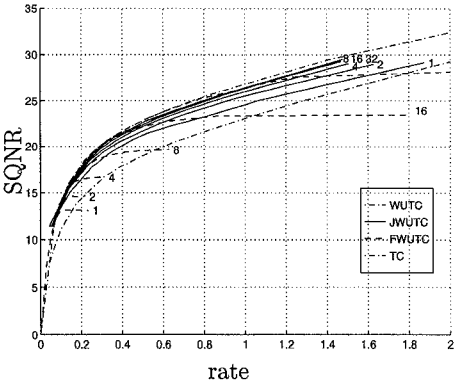

Figure 2: Comparison of SQNR results on a collection of combined text and gray scale images. The systems tested include WUTC, JWUTC and FWUTC at $c, c' \in \{1, 2, 4, 8, 16, 32\}$, and an optimal single transform code (TC). All codes operate on 8×8 data blocks. The WUTC, FWUTC, and JWUTC all use collections of up to 64 transform codes. The storage and complexity of each FWUTC and JWUTC equals $c/64$ or $c'/64$ of the storage and complexity required by the WUTC.

(bpp), 1/16 the storage and complexity for rates up to about 0.15 bpp, 1/8 the storage and complexity for rates up to about 0.20 bpp, 1/4 the storage and complexity for rates up to about 0.40 bpp, and 1/2 the storage and complexity for rates up to about 1.0 bpp. Further, the FWUTC exceeds the performance of the optimal transform code and JPEG for a far greater range of coding rates.

In Figure 2, we compare the performance of the JWUTC, FWUTC, WUTC, and an optimal transform code. The JWUTC with $c' \geq 8$ achieves performance virtually indistinguishable from that of the full complexity WUTC. As the figure indicates, the performance of the JWUTC far exceeds that of the FWUTC. This gain can be explained by the greater number of degrees of freedom experienced by the JWUTC. For example, for $c = c' = 1$ FWUTC uses each of its 64 components in a different transform. In contrast, the JWUTC can use component in a different transform like FWUTC, put all of its components into a single transform like the optimal transform code, or use any variation in between. In fact, the experimental results suggest that given a fixed total order, the optimal means of allocating that order varies as a function of rate, with more lower order bases at low rates and fewer higher order bases at high rates. A sample low-rate collection of transforms is shown in Figure 3. Figure 4 shows some compressed images.

The proposed algorithms provide methods for designing collections of fixed- and variable-order bases for data compression and image representation. In data compression, the FWUTC and JWUTC algorithms yield performance comparable to their predecessor the WUTC at a fraction of the computational and storage expense. The resulting algorithms yield up to 3 dB performance improvement over the JPEG

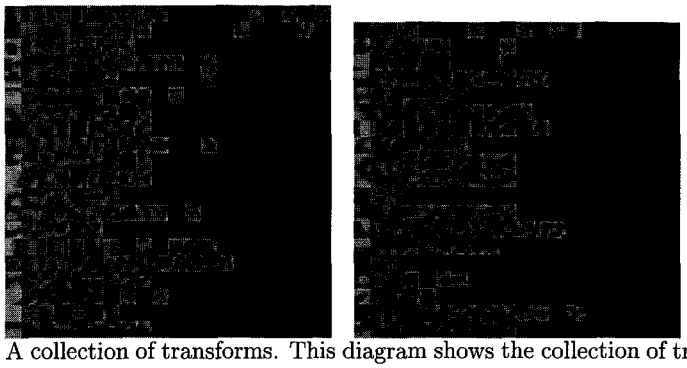

Figure 3: A collection of transforms. This diagram shows the collection of transforms used to code the $c' = 8$, rate=0.14 bpp image shown in Figure 4. The diagram contains 39 rows of 8×8 blocks – 20 in the section on the left and 19 in the section on the right. Each row contains 20 blocks and represents a single transform. The left-most block in each row shows the mean of all vectors that contributed to the design of the transform in that row. Each subsequent block shows a single eigenvector in the given transform – where eigenvectors are shown in order of decreasing eigenvalues and eigenvectors that have been removed from the system are colored solid black. The largest transform in the collection has order 13, while the smallest has order 2.

algorithm on a collection of combined text and gray-scale images, and do so with storage and complexity comparable to those of many JPEG implementations.

References

[1] M. Effros and P. A. Chou. Weighted universal transform coding: universal image compression with the Karhunen-Loeve transform. In *Proceedings of the IEEE International Conference on Image Processing*, Washington, D.C., October 1995. IEEE.

[2] P. A. Chou, M. Effros, and R. M. Gray. A vector quantization approach to universal noiseless coding and quantization. *IEEE Transactions on Information Theory*, IT-42(4):1109–1138, July 1996.

[3] M. Effros and P. A. Chou. Universal image compression. 1996. Submitted to the *IEEE Transactions on Image Processing* December 18, 1996.

[4] P. P. Vaidyanathan. *Multirate Systems and Filter Banks.* Prentice Hall Signal Processing Series, New Jersey, 1993.

[5] M. Effros and P. A. Chou. Weighted universal bit allocation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 2343–2346, Detroit,MI, May 1995. IEEE.

Figure 4: Details of compressed images. Top row: Originals; Subsequent rows – JWUTC, $c' = 8$, .14 bpp; JWUTC, $c' = 8$, .04 bpp; JPEG .14 bpp