EFFICIENT IMAGE CODING FOR ACCESS TO PIXEL RANGES

Sehoon Yea[1]*,Amir Said*[2]*,William A. Pearlman*[1]

[2] HP Labs, Palo Alto, CA
[1] Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
E-mail: {yeas,pearlman}@rpi.edu[1], amir_said@hp.com[2]

ABSTRACT

Technical imaging applications such as coding "images" of digital elevation maps, require extracting regions of compressed images with the pixel values within a pre-defined range, without having to decompress the whole image. Previously, we introduced a class of nonlinear transforms which are small modifications of linear transforms to facilitate a search for the regions with pixel values below(above) a given 'threshold', without incurring any penalty in coding efficiency. However, coding efficiency had to be somewhat compromised when searching for regions with a given pixel 'range', especially at high coding rates. In this paper, we propose an improved method of pixel 'range' coding to deal with the aforementioned problem. Results show significant improvements in coding efficiency.

1. INTRODUCTION

Many technical imaging applications require extracting regions of compressed images in which the pixel values are within a pre-defined range. An example of such application would be drawing the boundaries of a lake from an elevation map. Another example is the need to isolate in MRI images the organs with a certain density. For large images there is a need for coding methods that allow finding these regions efficiently, without having to decompress the whole image.

Current image compression methods that use multiresolution representations, like wavelets, enable users to efficiently browse or extract parts of large images without the need to decompress the whole image. While this is very convenient for natural images, it cannot be directly applied to those applications in which the pixel range is needed. The problem originates from the fact that low-resolution images are commonly obtained through averages of pixels from the original image, and the range is not conserved.

The use of lifting with some nonlinear filters ("nonlinear wavelets" [1]) provides one type of solution. However, it can only facilitate a search for the min or max but not both. Also it adds a large detrimental effect on compression performance. In our previous work [5], we showed that many of the linear transforms commonly used for image compression can be modified for that purpose by proving that the inclusion of nonlinear factors (like minimum or maximum pixel value in a block) does not render the transformation irreversible, and introduced a class of nonlinear transforms to facilitate a search for the regions with pixel values below(above) a given 'threshold', *without incurring any penalty in coding efficiency*. However, a rather severe degradation in coding performance was observed at high bit-rates when finding regions with a given pixel value 'range' by decomposing an image into two sub-images. In this paper, we attempt to mitigate the degradation by a new decomposition method along with the use of the Haar wavelet to take advantage of the similarities between the decomposed sub-images. The organization of this paper is as follows: Section 2 reviews a family of nonlinear transforms based on the order-statistics preserving property of linear transforms. Section 3 proposes a new way of providing a quick access to regions with a specified pixel range using the introduced transform. Section 4 provides simulation results, and Section 5 concludes this paper.

2. ORDER-STATISTICS PRESERVING LINEAR TRANSFORMS

2.1. Order-Preserving Linear Transforms

Throughout this section we assume that all vectors are defined in an N-dimensional space, and consider the linear transformations that are defined by an $N \times N$ non-singular matrix $\mathbf{T}$. To study the properties of a special class of transforms we first need to define the N-dimensional vectors

$$\mathbf{u} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \text{and} \quad \mathbf{v} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{1}$$

We say that matrix $\mathbf{T}$ defines an *order-preserving linear transform* (OPLT) if

$$\mathbf{T}^{-1}\,\mathbf{v} = \gamma\,\mathbf{u}. \tag{2}$$

0-7803-8554-3/04/$20.00 ©2004 IEEE.

which means that all elements of the first column of $\mathbf{T}^{-1}$ are equal to a constant value $\gamma \neq 0$.

Many commonly used transforms have this property. For example, the discrete cosine transform (DCT), the Walsh-Hadamard transform, and the dyadic Haar transform [3]. Note that all the multidimensional extensions of these linear transforms have the same property. For instance, the 8×8 2-D DCT has the same property, but in a space with dimension $N = 64$.

Let $\Omega = \{1, 2, \ldots, N\}$ be the set of integers from 1 to N, and let the order of the elements of a vector $\mathbf{x}$ be defined by the vector function $\mathbf{s} : R^N \rightarrow \Omega^N$, such that,

$$\bigcup_{i \in \Omega} s_i(\mathbf{x}) = \Omega, \tag{3}$$

$$s_i(\mathbf{x}) < s_j(\mathbf{x}) \implies (x_i < x_j) \text{ or } ((x_i = x_j) \text{ and } (i < j)). \tag{4}$$

For instance, $s_1(\mathbf{x})$ is the index of the smallest element of $\mathbf{x}$, $s_2(\mathbf{x})$ the second smallest, up to $s_N(\mathbf{x})$ which is the index of the largest element of $\mathbf{x}$. Note that this definition of $\mathbf{s}(\mathbf{x})$ is unique, since it also specifies the order when more than one element has the same value.

The reason we call these transforms order-preserving is that changes in the first component of the transform do not change the order when the inverse transform is computed. Unless otherwise stated, all the proofs can be found in [5].

Lemma 2.1 *Let* $\mathbf{T}$ *be a non-singular matrix that defines an order-preserving linear transform. For all* α *and* $\mathbf{x}$*, if* $\mathbf{a} = \mathbf{T}\mathbf{x}$ *and* $\mathbf{y} = \mathbf{T}^{-1}(\mathbf{a} + \alpha\,\mathbf{v})$ *then* $\mathbf{x}$ *and* $\mathbf{y}$ *have the same order, i.e.,* $\mathbf{s}(\mathbf{x}) = \mathbf{s}(\mathbf{y})$.

We define an order-statistic filter (OSF) [2, 4] with coefficient vector $\mathbf{c}$ as

$$\mu_{\mathbf{c}}(\mathbf{x}) = \sum_{j=1}^{N} c_j\, x_{s_j(\mathbf{x})}. \tag{5}$$

This linear combination of the sorted elements of $\mathbf{x}$ defines several commonly used filters. For instance, if only c_1 (c_N) is different from zero, then $\mu_{\mathbf{c}}(\mathbf{x})$ is proportional to the minimum (maximum) element of $\mathbf{x}$. If N is an odd number and only $c_{(N+1)/2}$ is different from zero, then $\mu_{\mathbf{c}}(\mathbf{x})$ is proportional to the median of the elements of $\mathbf{x}$.

Lemma 2.2 *For any vector* $\mathbf{x}$ *and any order-statistic filter with coefficient vector* $\mathbf{c}$ *we have*

$$\mu_{\mathbf{c}}(\mathbf{x} + \delta\mathbf{u}) = \mu_{\mathbf{c}}(\mathbf{x}) + \delta\,\mathbf{c}'\mathbf{u}. \tag{6}$$

Using the definitions and results above we can define a family of reversible nonlinear transforms as follows.

Proposition 2.3 *Let* $\mathbf{c}$ *be any vector such that* $\mathbf{c}'\mathbf{u} \neq 0$*, and let* $\mathbf{T}$ *be a non-singular matrix that defines an order-preserving linear transform, with* $\mathbf{t}_i$ *being the* i*-th row of* $\mathbf{T}$*. The nonlinear transformation* $\mathbf{f}(\mathbf{x})$ *defined by*

$$\begin{array}{rcll} f_1(\mathbf{x}) & = & \mu_{\mathbf{c}}(\mathbf{x}) & \\ f_i(\mathbf{x}) & = & \mathbf{t}_i\,\mathbf{x}, & i = 2, 3, \ldots, N, \end{array} \tag{7}$$

is reversible, and the inverse is defined by

$$\mathbf{x} = \mathbf{T}^{-1}\mathbf{f} + \frac{f_1 - \mu_{\mathbf{c}}(\mathbf{T}^{-1}\mathbf{f})}{\mathbf{c}'\mathbf{u}}\,\mathbf{u}. \tag{8}$$

Note that the only nonlinear component of the transform defined by (7) is $f_1(\mathbf{x})$. We can use the 8×8 discrete cosine transform as a practical example, to show how the new nonlinear transform can be used. From the first part of Proposition 2.3 we know that we can replace the "DC" coefficient of the DCT with any output of an order-statistic filter (OSF), and keep the same "AC" coefficients. For this example, let us assume that the OSF gives the minimum pixel value in the 8×8 block ($\mathbf{c} = \mathbf{v}$). To compute the inverse, we first compute the (linear) inverse cosine transform. Then we apply the same OSF filter to this block, i.e., we compute the maximum recovered pixel value $\mu_{\mathbf{v}}(\mathbf{T}^{-1}\mathbf{f})$, and finally add $f_1 - \mu_{\mathbf{v}}(\mathbf{T}^{-1}\mathbf{f})$ to all pixel values.

The proposed method also works for quantized transform coefficient cases. We just have to change definition (7) so that the OSF output is computed not from the original vector $\mathbf{x}$ but from the recovered data available at the decoder. Different from the linear transform, this requires the encoder to compute an inverse transform for every block.

3. CODING PIXEL RANGES

In the previous section we defined transforms that can efficiently represent only one extreme value (maximum or minimum, but not both). One obvious way of locating the regions with a specified pixel value 'range' is to use the Max(or Min)-OPLT while sending the minimum(or maximum) value of each block as a side-information. However, sending the side information may not be efficient since the size of side-information becomes relatively too large in the low bit rate region. In [5], we proposed a 'Left-Right Decomposition' method to facilitate the pixel 'range' search. A 'swapping' was performed as a way to guarantee the exactness of the search. However, it turned out that such a 'swapping' could aggravate the high bit-rates performances. In this section we propose a modified 'Left-Right decomposition' scheme with much reduced effect on the coding efficiency. As before, the modified scheme can search for pixel ranges of every region of size $2 \times M \times M$,where $M \times M$ is the size of an OPLT block.

3.1. LR(Left-Right) Decomposition Method

For every two adjacent pixels in the original image, the left pixel is copied to the 'left-image' and the right one is copied to the 'right-image'. See Fig. 1(a), where we illustrate an example with an image of size $N \times N$ with N=20. Since we are interested in pixel-value *ranges* of each $2 \times M \times M$ region in the original image, we have to find the maximum and the minimum. Notice that the maximum(minimum) of each $2 \times M \times M$ region can be found at the corresponding positions in the left or right block of the LR-decomposed images. For every pair of corresponding left & right blocks,we set the 'MAX_flag' bit to 0(1) when the maximum occurs in the left(right) block. Likewise, we set the 'MIN_flag' bit to 0(1) when the minimum occurs in the left(right) block. Then, we have four possible ways the maximum and the minimum can occur as illustrated in the Fig. 1(b) with a block size of 2×2(i.e.M=2). By using Max-OPLT on a block with the maximum and Min-OPLT on the one with the minimum, we can quickly decide whether the corresponding region of size $2 \times M \times M$ in the original image is within the specified pixel-range.

In case both the maximum and the minimum are achieved within the same block, we need the following Corollary to the Proposition 2.3.

Corollary 3.1 *The nonlinear transformation* $\mathbf{f}(\mathbf{x})$ *defined by*

$$\begin{aligned} f_1(\mathbf{x}) &= \mu_{\mathbf{c}}(\mathbf{x}) + \delta \\ f_i(\mathbf{x}) &= \mathbf{t}_i\,\mathbf{x}, \quad i = 2, 3, \ldots, N, \end{aligned} \tag{9}$$

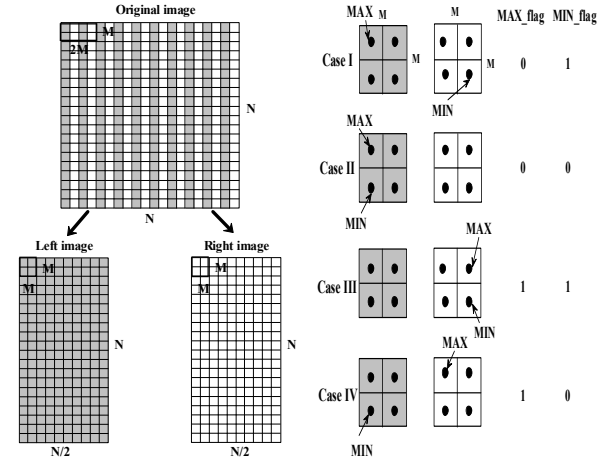
is reversible if an arbitrary constant δ *is known, and the inverse is given as*

$$\mathbf{x} = \mathbf{T}^{-1}\mathbf{f} + (f_1 - \mu_{\mathbf{c}}(\mathbf{T}^{-1}\mathbf{f}))\ \mathbf{u} - \delta\ \mathbf{u}. \tag{10}$$

when $\mathbf{c}$'$\mathbf{u}$*=1 (as with the Max-OPLT or the Min-OPLT).*

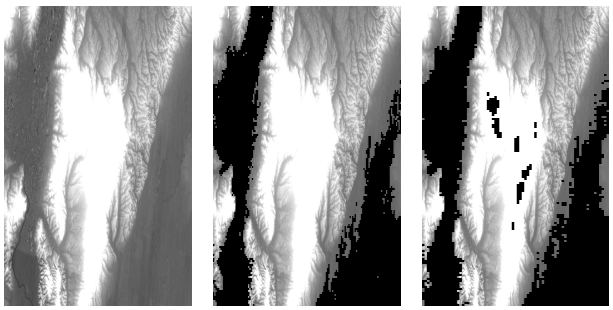
Proof: Obvious from the Proposition 2.3.

For example, suppose the maximum and the minimum of a left block are 180 and 110 and those of the corresponding right block are 176 and 112,respectively, then we have MAX_flag=0 & MIN_flag=0. In this case, we apply the Max-OPLT on the left block and the Min-OPLT(using (9)) on the right with δ =-2 ,which is the difference between the minimum values of the two blocks. Notice in (10) that the pixel values of the recovered right block will be shifted by δ from the original unless we subtract the 'correction' term $\delta\ \mathbf{u}$. We send δ as a side information when the maximum and the minimum occur in the same block as in the example. The bit rate overhead was typically 0.01$\sim$0.03 bpp using 8bits for each δ. We also need extra 2 bits for the MAX & MIN_flag for every block of size $2 \times M \times M$.(e.g. when M=8 as with a typical DCT , $2/(2 \times M^2)$=0.0156 bpp and this can be further reduced with entropy coding.)



(a) Left-Right decomposition (b) Max & Min flag bits

Fig. 1. Left-Right Decomposition of an $N \times N$ image with blocks of size $M \times M$ (N=20,M=2)



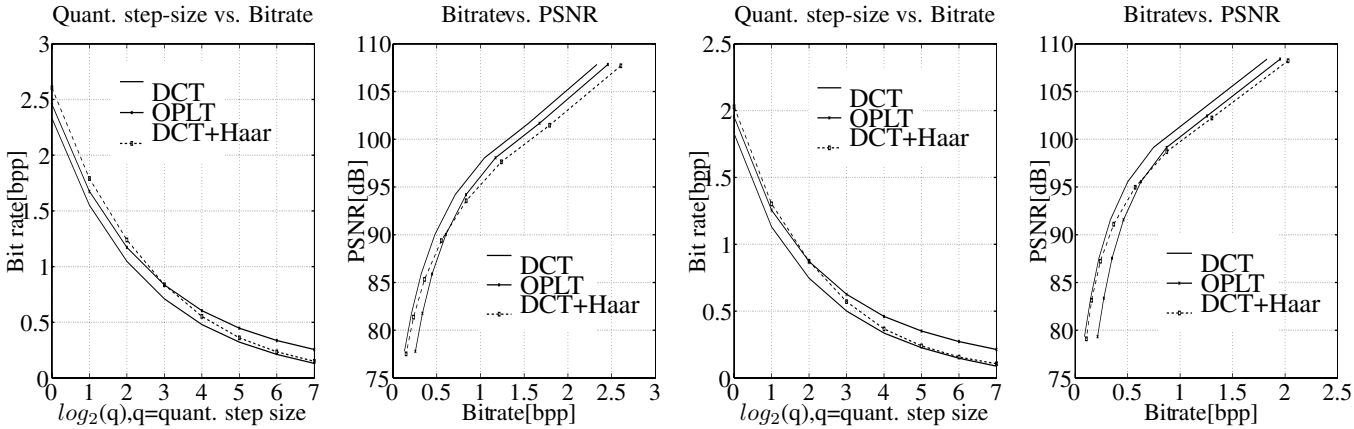
(a) Original Elevation Data (b) Regions above 100m (c) Regions above 100m and below 800m

Fig. 2. Decoding examples of regions from E17N45.raw

4. RESULTS

4.1. Finding Regions with Specified Pixel Values

Figure 2(b) shows a decoding example of regions above a certain *threshold*. An 8×8 DCT was used as a Min OPLT with each DC coefficient of DCT blocks replaced by the minimum value of the corresponding region. Only the regions with minimum elevation of 100m were decoded by reading out the first component of each 8×8 DCT coefficients block. Note this process does not incur any undesirable effect on compression efficiency. Figure 2(c) shows the result of decoding only the regions with a specified pixel value *range*. The LR-Decomposition was applied to find regions with a pixel range between 100m and 800m. Unlike the *threshold* case, this process usually results in a somewhat degraded coding efficiency. We investigate it in the next subsection.



(a) bitrates for e17n45 (b) Bpp vs. PSNR for e17n45 (c) bitrates for e17n46 (d) Bpp vs. PSNR for e17n46

Fig. 3. Coding Results with Left-Right Decomposition

4.2. Effect of the Decomposition on Coding Efficiency

Even with the modified LR-Decomposition proposed in Section 3.1, there still occurs degradation in coding efficiency because the decomposition into two smaller images results in less smooth sub-images compared to the original. To mitigate this undesirable effect, we applied the Haar transform between corresponding pairs of DCT coefficients of the left and right images. This makes sense because regions in the left and right images (and their corresponding DCT coefficients) have very similar values. Figure 3 shows a comparison of coding efficiency between cases with and without the decomposition. In figures 3-(a) and 3-(c), we plotted bit rates needed to code the quantized coefficients vs. the uniform quantization step-sizes from 2^0 to 2^9. In the figures,the meaning of the bit rate curves are as follows:

1. DCT : bit rate needed to code the quantized DCT coefficients of the original image (**curve 1**)
2. OPLT : bit rate needed when the Max-OPLT is applied to the original image and the min of every block of size $2 \times M \times M$ is sent as side-information (**curve 2**)
3. DCT+Haar : bit rate needed to code the left-image(e.g. with Max-OPLT) and right-image(e.g.with Min-OPLT) after Haar transform, plus the overhead bits for side information such as MAX/MIN flag and δ (**curve 3**)

At high rates, the total bit rates by the LR decomposition(curve 3) are still a bit worse than using the Max(or Min)-OPLT along with the min(or max) values as side information (curve 2). But the negative effect of the decomposition on the bit rate becomes smaller as we go down to the mid-to-low bit rate region(bigger quantization step-size), whereas the bit rate needed for side information(included in the bit rates of curve 2) remains the same regardless of the bit rate region. Figures 3-(b) and 3-(d) show the corresponding PSNR vs. bpp results for Fig 3-(a) and 3-(c), respectively. We can confirm that comparable(less than 1dB difference) PSNR vs. bpp performances at mid-to-high bit rates were obtained with much improved PSNRs at low bit rates when compared with the side-information cases.

5. CONCLUSION

This paper described a family of nonlinear transforms based on the order-statistics preserving property of linear transforms. We presented the result of preliminary experiments with an OPLT(Order Preserving Linear Transform) derived from DCT. For fast access to regions with a pixel-value 'threshold', it does not incur any negative effect on coding efficiency. Also, for regions with a specified pixel 'range', we proposed an improved decomposition-based method with comparable PSNR performance as the original DCT transform coding.

6. REFERENCES

[1] C. Creusere, "Compression of digital elevation maps using nonlinear wavelets," in *Proc. IEEE Int. Conf. Image Processing*, Vol. 3, pp. 7–10, Oct. 2001.

[2] H. A. David, *Order Statistics*, Wiley, Toronto, 1981.

[3] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

[4] A. C. Bovik, T. S. Huang, and D. C. Munson, "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoustics, Speech, Signal Proc.*, Vol. 31(6), pp. 1342–1349, Dec. 1983.

[5] A. Said, Sehoon Yea, and W. A. Pearlman, "Coding for fast access to image regions defined by pixel range," in *Proc. IEEE Data Compression Conference*, pp. 489–497, Mar. 2004.