Investigating Lossy Image Coding Using the PLHaar Transform

Joshua G. Senecal∗‡ Peter Lindstrom† Mark A. Duchaineau† Kenneth I. Joy‡

∗Institute for Scientific Computing Research
†Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
‡Institute for Data Analysis and Visualization
Computer Science Department
University of California, Davis

Abstract

The Piecewise–Linear Haar (PLHaar) transform [6] is to our knowledge the only \( n \)-bit to \( n \)-bit reversible transform suitable for lossy and lossless coding. Here we report on our further investigations into PLHaar’s characteristics. We measure the transform’s execution time, its response to thresholding, and examine the effect that PLHaar’s increased contrast has on the visual quality of reconstructed images. We compare our results to those obtained using the transform of Chao et al. (CFH) [2]. We find that PLHaar’s computational execution time is slightly slower, but its table–lookup execution time is comparable. PLHaar is more stable than CFH, and does not produce the objectionable artifacts found when CFH coefficients are coded lossily. The increased contrast in a PLHaar reconstruction may produce an image that is more visually appealing, even when the signal–to–noise ratio is lower.

1 Introduction

Dynamic Range Expansion [4] occurs in a transform when the range of transform outputs is wider than the domain of possible inputs. Practically, this means that the number of bits required to represent a transform coefficient is larger than the number of bits required to represent an input. An example of this is the S–transform [3]. If the inputs to the S–transform are \( n \)–bit data, the high–pass coefficients are \( (n+1) \) bits each. Dynamic range expansion presents some problems. A software implementation of a transform with dynamic range expansion is going to waste system resources: modern computer architectures allocate storage in 8–bit increments. If the incoming data is 8 bits wide (e.g. a grayscale image) the output will be 9 bits wide, but it will take a datatype 16 bits wide to store it. Also, if the hardware is limited to 8 bits, then it is impossible to transform the data in a lossless manner. While this may be acceptable for everyday digital photography there are other situations, such as medical imaging, where this may be unacceptable.

∗L–419, PO Box 808, Livermore, CA 94551, Tel: 925–422–3764, Fax: 925–422–7819
senecal1@llnl.gov
†{pl,duchaine}@llnl.gov
‡kjoy@ucdavis.edu
Given a transform that takes a pair of adjacent samples \((A, B)\), where \(A\) and \(B\) are each \(n\)-bits wide and can have any one of \(2^n\) possible values, the domain of the transform is the set of all input pairs \((A, B)\). The range of the transform is the set of all possible output coefficient pairs \((L, H)\), where \(L\) is a low-pass coefficient and \(H\) is a high-pass coefficient. Without loss of generality, unless stated otherwise we will assume that the domain is square and centered on \((0, 0)\).

To our knowledge there are three published methods for performing \(n\)-bit to \(n\)-bit reversible transforms. These are the method of Chao, Fisher, and Hu a (CFH) \([2]\), the Table–Lookup Haar method (TLHaar) \([5]\), and the Piecewise–Linear Haar method (PLHaar) \([6]\). CFH and TLHaar are discontinuous transforms—there are conditions where nearby inputs \((A_i, B_i), (A_j, B_j)\) will produce outputs \((L_i, H_i), (L_j, H_j)\) that are distant from each other. PLHaar does not have this problem; it is continuous. Therefore only the PLHaar transform can be used reliably in a lossy coding scheme.

Here we provide a more detailed look at some of the characteristics of the integer PLHaar transform. Specifically, we examine how the transform behaves with thresholded coefficients, its execution time in both table–lookup and direct–computation versions, and how PLHaar's increased contrast affects image quality at lower bitrates.

As we are concerned with \(n\)-bit to \(n\)-bit transforms we compare our results primarily with the CFH transform\(^1\).

Our results show that PLHaar executes slower than the CFH transform, but the difference in execution time is only about 0.04 seconds for a 2048x2048 pixel image. If the transforms are performed by table lookup, their execution times are essentially identical. For moderate threshold levels, the PSNR of a low–contrast image reconstructed from PLHaar coefficients tends to be less than that obtained from the CFH transform. For higher threshold levels the PSNR of a reconstructed PLHaar image is less than that of a CFH reconstruction, but the increased contrast in the PLHaar reconstruction gives a more visually appealing image. If the image has areas of high contrast then at all threshold levels the CFH reconstruction contains severe artifacts due to CFH's discontinuities. PLHaar does not have this problem and is applicable to all types of images.

2 Review of Prior Work

For a transform to be reversible it must be a 1:1 mapping between a domain and range: the number of points in the range is equal to the number of points in the domain. Therefore, to create an \(n\)-bit to \(n\)-bit transform, the task is to make the range an intelligent permutation of the domain, such that the range occupies the same space as the domain, and the transform gives desired results (e.g. data decorrelation).

The transform of Chao, Fisher, and Hua \([2]\) uses modular arithmetic to make sure that the transform coefficients require the same number of bits as the inputs. CFH is unable to distinguish large positive numbers from small negative ones (and vice versa)

\(^1\)The TLHaar transform is useless for lossy coding (see \([6]\)), and we do not include it in our comparisons.
because they have the same binary representation. This causes the transform to be discontinuous, and severe artifacts may result if lossy coding is used.

The Table–Lookup Haar (TLHaar) transform [5] uses a pair of procedurally created two–dimensional lookup tables that attempt to preserve the coefficient relationships in the Haar transform. TLHaar’s shortcomings are that it is very discontinuous, and therefore only suited for lossless coding. The second shortcoming is its lookup tables. For \( n \)-bit inputs each table contains \( 2^2n \) entries of \( 2^n \) bits per entry. This quickly becomes unwieldy at larger values of \( n \).

### 2.1 Piecewise–Linear Haar

The Haar wavelet transform is defined by the equations \( L = (A + B)/\sqrt{2} \) and \( H = (B - A)/\sqrt{2} \). In figure 1 we see that this is a 45–degree (or one–eighth) rotation of the domain about the origin in \( L_2 \) (or Euclidean) space: points in the domain that are equidistant from the origin lie on the perimeter of the same circle, and are moved along that circle. The PLHaar transform [6] is a similar one–eighth rotation about the origin, except that it happens in \( L_\infty \) space. In this space points that are “equidistant” from the origin lie on the perimeter of a square, and each point moves one–eighth along the perimeter of that square. If we divide the domain and range into octants, a one–eighth rotation moves all points from their positions in a given octant into the next octant. This is depicted in figure 2, where numbered octants are also shown. The transform as a whole is nonlinear, but when taken on a piecewise basis, the transform from octant to octant is linear. It is from this property that we derive the name “Piecewise–Linear Haar”, or “PLHaar”.

This transform has no dynamic range expansion, maintains continuity, and provides data decorrelation. PLHaar is simple to implement and is not restricted to integer domains.

PLHaar can be defined mathematically as

\[
\begin{bmatrix}
L \\
H
\end{bmatrix} = \begin{bmatrix}
f_{2,6}(+1) & f_{1,5}(+1) \\
f_{3,7}(+1) & f_{4,8}(-1)
\end{bmatrix} \begin{bmatrix}
A \\
B
\end{bmatrix}
\]

(1)

where \( f_{i,j}(x) = 0 \) if \((A, B)\) is in octants \( i \) or \( j \), and \( f_{i,j}(x) = x \) otherwise\(^2\). Note that by negating \( H \) we have here defined PLHaar as an *improper* rotation—a rotation fol-

\(^2\)Contrast this with the nonnormalized Haar transform (Haar without the division by \( \sqrt{2} \)) where \( f_{i,j}(x) = x \) everywhere
#define ABS(x) ((x) < 0 ? -(x) : (x))
#define SIGN(x) ((x) < 0 ? -1 : 1)

void
plhaar_float(
    FLOAT *l, // low-pass output
    FLOAT *h, // high-pass output
    FLOAT a,  // input #1
    FLOAT b); // input #2
);
{
    if (SIGN(a) == SIGN(b)) {
        *l = ABS(a) > ABS(b) ? a : b;
        *h = a - b;
    } else {
        *l = a + b;
        *h = ABS(a) > ABS(b) ? a : -b;
    }
}

void
plhaar_int(
    INT *l, // low-pass output
    INT *h, // high-pass output
    INT a,  // input #1
    INT b); // input #2
{
    const INT s = (a < c), t = (b < c);
    a += s; b += t; // (**) nudge origin
    if (s == t) { // A * B > 0?
        a -= b - c; // H = A - B
        if ((a < c) == s) // |A| > |B|?
            b += a - c; // L = A (replaces L = B)
    } else { // A * B < 0
        b += a - c; // L = A + B
        if ((b < c) == t) // |B| > |A|?
            a -= b - c; // H = -B (replaces H = A)
    }
    a -= s; b -= t; // (**) restore origin
    *l = b; *h = a; // store result
}

Figure 3: Source code for continuous (left) and discrete (right) PLHaar.

...we encounter a problem when the domain is a size that is a power of 2. In this case there is no central point around which to perform the rotation, which would require that points be rotated in half–point increments. The additional manipulations required to properly perform the rotation are similar (but not equivalent) to using signed one's complement arithmetic. The result is that the coefficient histogram is not centered about zero, but about positive and negative zero. We refer to to this particular PLHaar transform as PLHaar2, and the unsigned zero PLHaar transform as PLHaar1. In this paper we use both PLHaar1 and PLHaar2. When referring to the PLHaar transform in general, we use the term “PLHaar”. Signed zeros cause minor difficulties when encoding, decoding, and thresholding. We are unable to completely explore this issue here, but some of its consequences will be addressed in this paper.

Source code for the discrete and continuous PLHaar transforms is given in figure 3. This procedure uses no extra intermediate precision, and can take both signed and unsigned integers. Its use is self–explanatory, with the exception of the bias parameter, which is used to move the output range. For example, if the inputs are n–bit values from a domain [0,2^n – 1] the bias parameter should be set to 2^n – 1 to keep the high– and low–pass coefficient range equal to the domain. In the source, the lines marked (**) are necessary only when the domain is a power of two (e.g. [0,255]), as explained above. In this case there is no unique origin, so we translate
each quadrant so that its origin is at a common point, perform the transform, then translate the quadrant back. If the domain and range contain an odd number of integers (e.g. [0,254]) then the lines marked (**) may be removed and the bias set accordingly (e.g. 127).

This procedure is able to perform both the forward and inverse transforms. To perform the inverse transform, $L$ is passed as parameter $a$, $H$ as $b$, and $A$ and $B$ are taken respectively from parameters $l$ and $h$.

3 Evaluating PLHaar

3.1 Execution Time

To measure execution time we implemented the S, CFH, and PLHaar transforms in both direct–computation and table–lookup versions, and measured the time it took to transform square photographic images of edge length 256, 512, 1024, and 2048 pixels.

Because PLHaar is a transform from one octant to the next there is some additional overhead incurred during the transform process: before each point can be transformed its octant must be identified (see equation 1). In our current implementation this requires two nested if–then statements. Because of this we anticipated that PLHaar would be slower than the CFH and S transforms. However, we also anticipated that in a table–lookup version PLHaar’s execution speed will be similar to the other transforms.

Execution time was measured on a Dell Precision Workstation 530 with dual 2.2 GHz Intel Xeon processors (only one was used in the tests), 512 kilobytes of cache, and 512 megabytes of RAM. The operating system was RedHat Linux 3.2.3. All times reported are an average over 256 forward transformations.

Results for execution time using direct computation are given in figure 4, and for table lookup in figure 5. From figure 4 we see that as expected the execution times for PLHaar are slower than the S and CFH transforms, due to the additional work required for determining which octant a point is located in. Even so, PLHaar has a throughput of about 28 megabytes a second. When table lookups are used the difference in execution times between the CFH and PLHaar transforms essentially disappears.

Interestingly, for small images the direct computation execution time of the S transform is slower than the other transforms (this is not obvious from the graphs). We believe that this is because of some additional overhead incurred in the S–transform. Its coefficients are stored in 16–bit short integers, as opposed to 8–bit bytes, and therefore the coefficients do not fit in cache as well as CFH and PLHaar. For larger data this overhead is made up for by the S–transform’s fast execution speed.
3.2 Thresholding and PSNR

In previous work we examined the effect of quantization on PSNR. Thresholding (setting all coefficients whose magnitude is ≤ T to zero) is a more common operation, and we examine that here. Thresholding is typically done because the coefficients of lesser magnitude generally have less of an impact on the quality of the reconstructed data, and by eliminating these coefficients the data can be more efficiently encoded without adversely affecting the quality of the reconstruction.

Here we encounter the signed–zero problem described in section 2.1. If the data to be transformed has a domain that is a power of 2, then we must use PLHaar2, which creates signed zeros. Because the zeros have signs coefficients whose magnitude is ≤ T cannot be completely eliminated—they must be set to either positive or negative zero. So the problem becomes one of determining the best way to treat the signed zeros. Although we could “cheat”, and set all coefficients equal to the zero that has the same sign as the coefficient, this is not really fair. Instead we take a deterministic approach: if a high–pass coefficient is set to zero, during reconstruction we give that zero the same sign as its associated low–pass value. While this approach may not deliver the best PSNR, it works well on average.

Results for simple thresholding are given in figures 6 through 9. In these figures we plot the entropy resulting from each threshold level vs. the PSNR resulting from that threshold. From these we see that for smooth photographic images (Lena and Mandrill), where there are few areas of high contrast, PLHaar’s PSNR is not as good as that obtained from the CFH transform. For moderate threshold levels the PSNR is not that much worse, however. When there are many areas of high contrast, as found in the Wedding Photo image, CFH’s discontinuities ruin the PSNR of its reconstructed images. Images of thresholded reconstructions for the Wedding photo, Mandrill, and Woman [1] are given in figures 10, 11, and 12. Figure 10 gives an excellent example of the problems inherent in the CFH transform. The areas of high contrast found in this photograph cause considerable artifacts, even at a small threshold level. Figure 12 highlights one of the dangers of the CFH transform. Although figure 8 appears to indicate that CFH’s reconstruction is better, we see objectionable artifacts in the
reconstruction.

The major caveat when using the CFH transform is that unless one is willing to risk data corruption CFH locks a user into using lossless coding. PLHaar does not, and this is PLHaar’s main strength: it is suitable for lossy coding, while CFH is not.

### 3.3 Contrast

As mentioned in [6] when transform coefficients created by PLHaar are quantized, the reconstructed data has generally higher contrast. More quantization results in higher contrast. While this higher contrast may result in an image whose PSNR is worse than an image reconstructed from quantized CFH or S–transform coefficients, the increased contrast in the PLHaar reconstruction may be more visually appealing or provide the viewer with more information. Indeed, PSNR is not the sole criterion for determining the quality of an image.

Examining figures 10, 11, and 12 we see that as the threshold level increases, the contrast of the reconstructed image increases. Particularly with the Mandrill image we see that the increased contrast improves the visual quality of the reconstruction.
when compared to CFH. In this case, a viewer is more likely to prefer the PLHaar1 reconstruction.

4 Future Research

Future work with PLHaar will involve examining how well it performs in actual use, as the front end of an embedded coder for instance. Also, all of our work thus far has used integer PLHaar. Some study of the continuous PLHaar transform may provide additional information and insight that will prove useful.

With PLHaar we are fortunate, because the Haar transform, on which PLHaar is based, is a simple rotation. We were able to create an \(n\)-bit to \(n\)-bit transform by identifying an alternative rotation that does not result in dynamic range expansion. Further research into \(n\)-bit to \(n\)-bit transforms will examine the feasibility of developing more complex transforms. Is it possible to create for instance an \(n\)-bit to \(n\)-bit version of a linear wavelet, perhaps with lifting? Can we demonstrate that more complex wavelets do or do not exist? Or can we demonstrate that, even if such wavelets exist, implementing them would be prohibitively difficult?
Figure 11: Mandrill, thresholded to an approximate entropy of (from left) 4, 2, and 0.5 bits per pixel. Upper images are CFH, lower are PLHaar1. PSNR is given beneath each image.

5 Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Joshua Senecal’s work was supported in part by a United States Department of Education Government Assistance in Areas of National Need (DOE-GAANN) grant #P200A980307.

References


Figure 12: Woman, thresholded to an approximate entropy of (from left) 2, 1, and 0.47 bits per pixel. Upper images are CFH, lower are PLHaar1. PSNR is given beneath each image.


