UCRL-CONF-204061

An Improved N–Bit to N–Bit Reversible Haar–Like Transform

Joshua Senecal
Peter Lindstrom
Mark A. Duchaineau
Kenneth I. Joy

This paper was submitted to the 12th Pacific Conference on Computer Graphics and Applications, to be held October 6–8, 2004, in Seoul, South Korea

May 10, 2004

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

An Improved N–Bit to N–Bit Reversible Haar–Like Transform

Joshua Senecal[*‡], Peter Lindstrom[†], Mark A. Duchaineau[†]
[*]Institute for Scientific Computing Research
[†]Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

Kenneth I. Joy[‡]
[‡]Center for Image Processing and Integrated Computing
Computer Science Department
University of California, Davis

Abstract

We introduce the Piecewise–Linear Haar (PLHaar) transform, a reversible n–bit to n–bit transform that is based on the Haar wavelet transform. PLHaar is continuous, while all current n–bit to n–bit methods are not, and is therefore uniquely usable with both lossy and lossless methods (e.g. image compression). PLHaar has both integer and continuous (i.e. non-discrete) forms. By keeping the wavelet coefficients to n bits PLHaar is particularly suited for use in hardware environments where channel width is limited, such as digital video channels and graphics hardware.

1. Introduction

Integer wavelet transforms have what is termed *dynamic range expansion*. Simply put, the range over which the wavelet coefficients have their values is larger than the range over which the inputs have their values (for a discussion of dynamic range expansion and its effects see [4]). As an example, the S–Transform [1] (to be reviewed later) is an n–bit to $(n + 1)$–bit transform. This means, from a practical standpoint, that the process of designing and building custom hardware implementing this transform will incur expense and complexity from the necessity of handling the extra bit. Also, because modern computer architectures allocate storage in 8–bit increments, an implementation of this algorithm on a computer will require 16 bits to store what is essentially a 9–bit coefficient (an 8–bit magnitude plus a sign bit).

To our knowledge there are only two published lossless methods for completely eliminating dynamic range expansion. These are the TLHaar method [7] and the method of Chao and Fisher (here referred to as the CF method)[2] (to be reviewed later). PLHaar improves on these methods by supporting lossless and lossy methods.

2. The Haar Wavelet Transform

A good illustration of dynamic range expansion is found in the Haar wavelet transform [6]. The transform is a 45–degree (or one–eighth) rotation about the origin in Euclidean (or L_2) space. Points in the domain that are equidistant from the origin lie on a circle. The transform rotates all points on a particular circle.

Given two input values A and B the corresponding high–pass value H and low–pass value L can be computed by equations 1 and 2.

$$H = \frac{B - A}{\sqrt{2}} \tag{1}$$

$$L = \frac{A + B}{\sqrt{2}} \tag{2}$$

Or in matrix form

$$\begin{bmatrix} L \\ H \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} \tag{3}$$

2.1. The S–Transform

If the normalization (division) by $\sqrt{2}$ is removed from the Haar equations, the range of the transform is equivalent to the domain, expanded by a factor of $\sqrt{2}$ and rotated

[*] L–419, PO Box 808, Livermore, CA 94551, Tel: 925–422–3764, Fax: 925–422–7819, *senecal1@llnl.gov*

[†] L–561, PO Box 808, Livermore, CA 94551, *{pl,duchaine}@llnl.gov*

[‡] One Shields Ave, Davis, CA 95616, *{jgsenecal,kijoy}@ucdavis.edu*

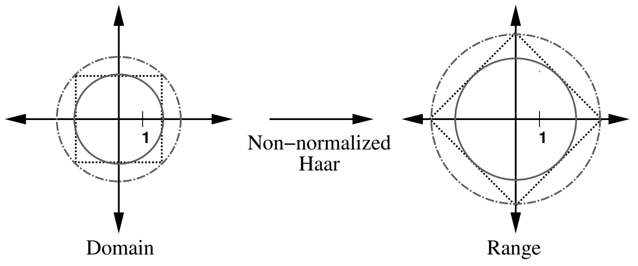

Figure 1. The non–normalized Haar transform, showing dynamic range expansion.

45 degrees about the origin. The range of possible high– and low–pass values, as measured along the axis, is now twice that of the domain (see figure 1). If the domain contains only integers, it should be obvious that the range of the non–normalized Haar transform will also contain only integers, and that the number of positions occupied in the range is equal to the number of positions in the domain. If the domain is $[0, N]^2$ then the range of values will be drawn from $[-N, +N]^2$, arranged in a lattice as in the left side of figure 2.

S–Transform [1], defined by equations 4 and 5, is an integer version of the Haar Transform (the S–Transform is sometimes called the Haar Integer Wavelet Transform).

$$H = B - A \tag{4}$$

$$L = \left\lfloor \frac{A + B}{2} \right\rfloor \tag{5}$$

It eliminates dynamic range expansion in the low–pass coefficients by "squashing" the low–pass range so that it is equal to the domain, as shown on the right of figure 2. Note that for any pair of inputs to the non–normalized Haar transform, H and L have the same least–significant bit (LSB). Because of this the low–pass values can be squashed without loss of information (the LSB needed to completely reconstruct L can be taken from H) and the transform is completely reversible. However, the range of high–pass coefficients is still twice the domain.

2.2. TLHaar

Because the number of occupied positions in the domain and range are equal it is possible to eliminate dynamic range expansion in the low– **and** high–pass coefficients by intelligently remapping each position in the domain to a position in the range, such that the range is equal to the domain and the resulting transform de–correlates the input data.

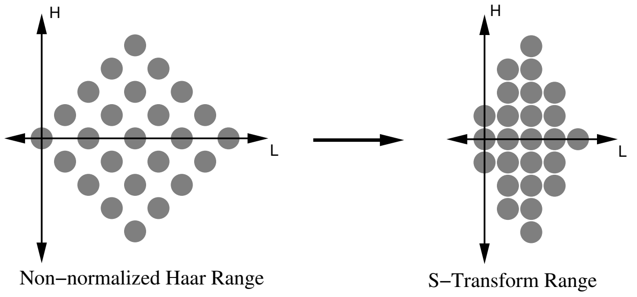

Figure 2. Eliminating low–pass dynamic range expansion by "squashing".

TLHaar [7] is an integer–to–integer transform that, given a bit width n, uses a set of two–dimensional lookup tables (each of size 2^{2n}, with an edge dimension of 2^n). One table is a mapping $(A, B) \rightarrow (H, L)$ for the forward transform, and the table for the inverse is a mapping $(H, L) \rightarrow (A, B)$. Each table is initialized with an identity transform, and then the rows and columns of the inverse transform table are sorted to agree with their true Haar counterparts. When a swap occurs in the inverse transform table, the corresponding entries in the forward transform table are also swapped.

Given two pairs of inputs (A_i, B_i), (A_j, B_j), the sort of the inverse transform table obtains the following two properties:

$$\forall L : |H_i| \leq |H_j| \iff \tilde{H}_i \leq \tilde{H}_j \tag{6}$$

$$\forall H : L_i \leq L_j \iff \tilde{L}_i \leq \tilde{L}_j \tag{7}$$

where H indicates a Haar high–pass coefficient, and $\tilde{H}$ indicates a TLhaar coefficient. The intent is to create tables such that, for any given two pairs of data values their high– and low–pass values as created by TLHaar will tend to have the same magnitude relationships as those created by Haar. Because the tables are initialized with an identity transform no table entry will have a value outside the transform domain. Thus range expansion cannot occur.

Although this method decorrelates the data, it has several shortcomings. The first is that it is not continuous and is therefore suitable only for use with lossless methods, although lossy methods are possible if the loss is confined to the subbands at the finest levels of detail. Second, for n–bit inputs the lookup tables are of size 2×2^{2n} each, which quickly becomes unwieldy at larger values of n.

2.3. The CF Method

The CF method [2] is also an integer–to–integer transform, and takes advantage of modulo arithmetic to eliminate dynamic range expansion. Given a bit width n, the range of representable signed values is $[-2^{n-1}, 2^{n-1}-1]$. Given inputs A and B the CF transform is computed as in equations 8 and 9.

$$H = (B - A) \bmod 2^n \tag{8}$$

$$L = \left(\left\lfloor \frac{H}{2} \right\rfloor + A \right) \bmod 2^n \tag{9}$$

where we use the standard convention that $-x \bmod 2^n = 2^n - x$ for $0 < x < 2^n$.

The CF method has an aliasing problem that causes incorrect behavior when the difference between A and B overflows and wraps around, causing L to have a sign opposite the expected. CF is unable to distinguish large positive numbers from small negative ones (and vice–versa) because they have the same representation. For example, if $n = 8$, the range is [-128,127]. If $A = -1$ and $B = 127$, the CF method computes $L = -65$, which is not near the expected average of 63 (recall that the L value is an average of A and B). If the wavelet coefficients are kept lossless then this aliasing does not cause any problems and the original image can be reconstructed. However, if a lossy method is used to store or encode the coefficients then artifacts can appear in the image. Most continuous–tone images do not have adjacent pixel pairs with differences wide enough to cause this behavior, so in practice this method works well for a wide range of images, reconstructed both lossy and lossless. However, this aliasing problem is a fundamental weakness in the method, and means that the CF transform is not continuous. The PLHaar transform, unlike the CF and TLHaar transforms, performs this remapping while completely preserving continuity.

3. The PLHaar Transform

Unlike the Haar transform, which is defined as a 45–degree rotation in Euclidean L_2 space, the PLHaar transform is a similar rotation in L_∞ space. In this space, points that are "equidistant" from the origin lie on the perimeter of a square. A one–eighth rotation (analogous to the 45–degree rotation of the Haar transform) about the origin in this space amounts to moving a point one–eighth of the distance along the perimeter of its square. If we divide the domain and range into octants, then as shown in figure 3 a one–eighth rotation moves all points from their positions in a given octant into the next lower octant (with wraparound).

The transform of a given point is given by

$$\begin{bmatrix} L \\ H \end{bmatrix} = U \begin{bmatrix} A \\ B \end{bmatrix} \tag{10}$$

where U is one of four matrices, depending on the octant (Oct.) where the point (A, B) is located.

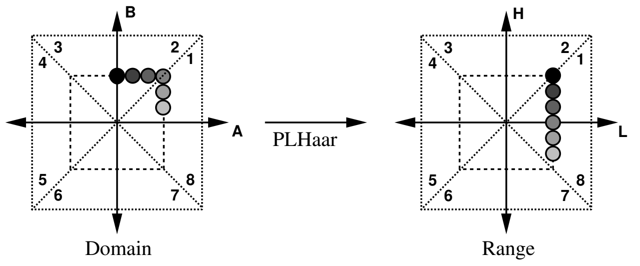

Figure 3. The PLHaar transform's rotation in L_∞ space. Numbered octants are also shown.

$$U = \begin{cases} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} & \begin{array}{ll} \text{if } 0 \le +B \le +A & \text{Oct. 1} \\ \text{or } 0 \le -B \le -A & \text{Oct. 5} \end{array} \\[2ex] \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} & \begin{array}{ll} \text{if } 0 \le +A \le +B & \text{Oct. 2} \\ \text{or } 0 \le -A \le -B & \text{Oct. 6} \end{array} \\[2ex] \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} & \begin{array}{ll} \text{if } 0 \le -A \le +B & \text{Oct. 3} \\ \text{or } 0 \le +A \le -B & \text{Oct. 7} \end{array} \\[2ex] \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix} & \begin{array}{ll} \text{if } 0 \le +B \le -A & \text{Oct. 4} \\ \text{or } 0 \le -B \le +A & \text{Oct. 8} \end{array} \end{cases} \tag{11}$$

Note that U is the non–normalized Haar matrix, but with one of the four elements zeroed out. Also, in the PLHaar transform, PL(x,x) = (x,0), decorrelating adjacent identical values, just as in the Haar transform. The PLHaar transform maps integers to integers, is an autohomeomorphism (meaning it maps the domain onto itself, is continuous, and is one–to–one), and is piecewise–linear.

PLHaar's continuity gives it an advantage over other n–bit to n–bit transforms, as this continuity makes PLHaar uniquely suitable for lossy compression. Given a pair of inputs (A, B) and their outputs (L, H), if the outputs are modified—via quantization or some other lossy procedure—to nearby values (L', H'), the reconstructed values (A', B') will be close to the original input pair. The TLHaar and CF transforms do not have this property.

3.1. Efficiency Considerations

In the definitions of the Haar and PLHaar transforms, the choice of using $H = A - B$ or $H = B - A$ is arbitrary; the choices presented thus far make Haar and PLHaar rotations. If the opposite choice is made, the result is a rotoinversion R—a rotation followed by a reflection—with the property that $R(R(A, B)) = (A, B)$, i.e. $R = R^{-1}$ is an involution.

Having PLHaar be an involution is a desirable property because it reduces the number of procedures required to compute the forward and inverse transforms–the same procedure is able to compute both. Also, if lookup tables are to be used, instead of a procedure, the same lookup table can be used for the forward and inverse transforms. Thus, compared to TLHaar and CF, the storage required for tables is cut in half. To make PLHaar an involution we negate the second row of each U matrix, previously defined in equation 11. This has no effect on the transform's continuity.

Conceptually, in order to make the transform an involution an additional step is added to the transform, where after rotation H is negated. This causes a vertical flip in the domain. If these coefficients are also rotated and flipped, they return to their original positions. This is illustrated for the discrete case in figure 4.

3.2. Implementation

Source code for the continuous PLHaar transform is given in figure 5, and for the discrete transform in figure 6. Note that both of these procedures used the modified transform (the involution) described above.

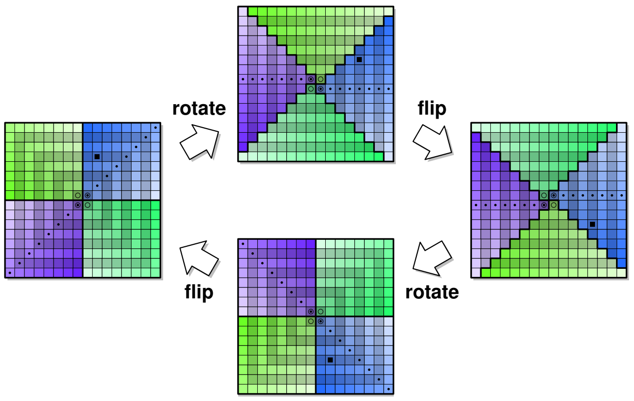

Figure 4. How PLHaar is an involution. Note that both rotations are in the same direction.

The procedure for the discrete transform 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 8–bit values from a range [0,255] the bias parameter should be set to 128 to keep the high– and low–pass coefficients in the same range. In the source, the lines marked "asymmetry" are necessary only when the domain and range contain an even number of integers (e.g. [0,255]). 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, and then translate the quadrant back.

```
#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;
  }
}
```

Figure 5. The source code for the continuous PLHaar transform.

These procedures are 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`.

4. Transform Entropy

Transforms are used because the transformed data may be more easily manipulated, a common manipulation being compression. To verify that PLHaar is usable as a transform for compression we iteratively transformed some standard test images, at each iteration applying the transform to the low–pass coefficients resulting from the previous iteration, until there was a single low–pass coefficient remaining. We then took a histogram of the coefficients and measured the normalized zero–order entropy E [8] according to equation 12

$$E = \frac{\sum_{i=1}^{m} -p_i \times \log p_i}{\log m} \tag{12}$$

where p_i is the probability of coefficient value i and m is the number of coefficient values with nonzero counts in the histogram. We compared the entropy resulting from the PLHaar transform to those of the S, TLHaar, and CF transforms. Results of the entropy measurements are given in figure 7, where "Image" is the entropy of the untransformed

```
void
plhaar_int(
  INT   *l,               // low-pass output
  INT   *h,               // high-pass output
  INT   a,                // input #1
  INT   b,                // input #2
  INT   c                 // bias
)
{
  const INT s = (a < c), t = (b < c);

  a += s; b += t;         // asymmetry: nudge
                          // origin to (+0,+0)
  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;         // asymmetry:
                          // restore origin
  *l = b; *h = a;         // store result
}
```

Figure 6. The source code for the discrete PL-Haar transform.

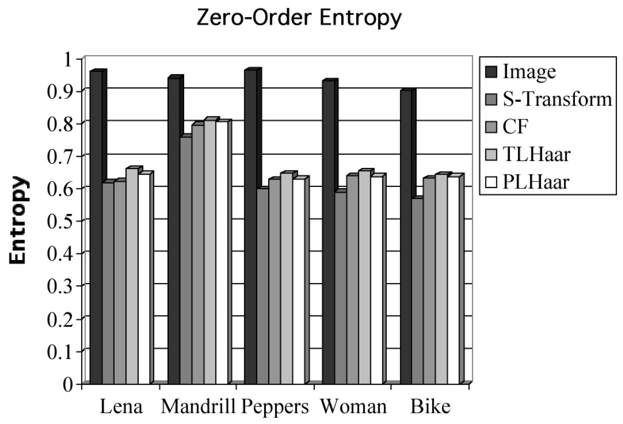

Figure 7. The zero–order entropy of the wavelet coefficients produced by the transforms.

image. A graph of the histograms for the "lena" image is in figure 8.

The data in figure 7 shows that all of the tested transforms are useful for compression. We see that the S–Transform achieves the lowest entropy, and TLHaar the highest. In general CF obtains an entropy that is less than PLHaar, although the difference is slight: from the images tested the average difference in entropy is 0.0076. Further testing on a wider range of images is needed to determine more accurately what the average difference between the two methods' entropy is.

These results also indicate that there is a trade-off with using n–bit to n–bit transforms: any advantage gained in using the transform is offset by a decrease in compression efficiency.

5. Quantization and PSNR

To gain a basic understanding of how useful PLHaar will be when used in a lossy compression or progressive transmission scheme, we performed some quantization tests on the "Lena" image. We transformed the image as in section 4, and then iteratively quantized the coefficients to shorter bit widths. We treated the S–Transform coefficients as being 9 bits wide (sign bit plus an 8–bit magnitude).

Quantized coefficients fall into a range of uncertainty. For example, if the coefficient 42 (00101010) is quantized to 5 bits of precision, it becomes 40 (00101000). This quantized coefficient could have had an original value in the range [40,47]. Assuming a uniform distribution, to avoid biasing everything towards zero, and to preserve contrast, we place the quantized coefficients near the center of the range of uncertainty. For a given non–negative uncertainty interval $[x, y]$ we compute the center as $\lfloor (x + y)/2 \rfloor$, and center the quantized coefficients accordingly. After performing this quantization and centering we applied the inverse transform and computed the peak signal–to–noise ratio (PSNR) of the result, according to equation 13 ($RMSE$ is the Root Mean–Squared Error).

$$PSNR = 20 \times \log \left(\frac{255}{RMSE} \right) \qquad (13)$$

Figure 11 gives an excellent example of the problems inherent in the CF method. It shows a grayscale photo that has been transformed by the S, PLHaar, and CF transforms and quantized to 4 bits before reconstruction. As expected many artifacts appear, particularly in areas of high contrast. PSNR values are respectively 21.88, 25.17, and 11.75.

A graph of the PSNR curve for the image "Lena" is given in figure 9. This image does not contain many areas of high contrast, and based on this and the fact that the PSNR values for CF are better than all other methods, we might expect that CF would give a better reconstruction of the im-

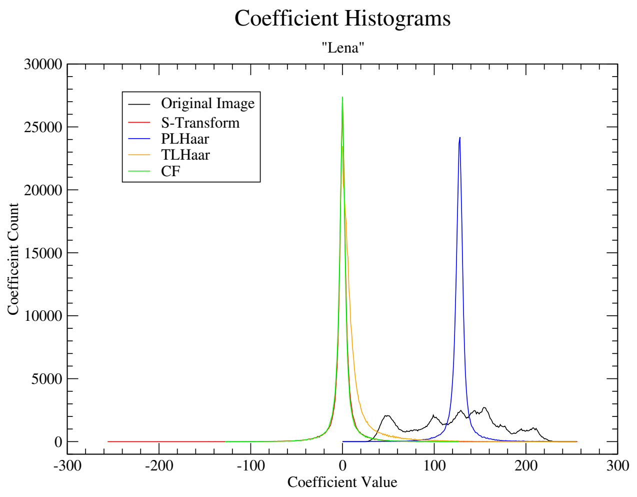

Figure 8. Histograms of the "lena" image, and the S–Transform, PLHaar, TLHaar, and CF coefficients.

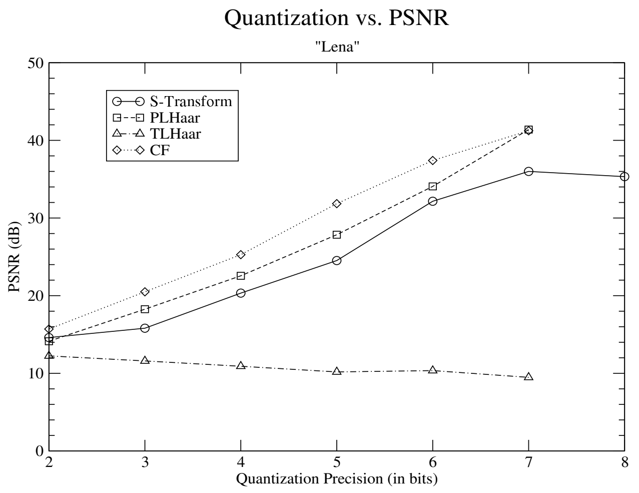

Figure 9. Lena: PSNR values for increasing quantization precision.

age regardless of quantization. However, in figure 10 a plot of the L_∞ error of each image shows that the L_∞ error of the CF reconstruction suddenly jumps from a value of 27, at a quantization precision of 5 bits, to 221 at a quantization precision of 4 bits. The L_∞ error for the PLHaar and S–Transform methods degrades more gracefully. An image of the reconstructions for the S, CF, and PLHaar transforms is given in figure 12. We note in particular that a few artifacts appear in the 3–bit image (note the inner edge of the mirror frame and along brim of her hat).

Overall, our observation is that for images that do not have areas of high contrast, the CF method is appropriate if quantization is minimal. Even with images such as Lena that do not have many areas of high contrast, if the coefficients are quantized past a certain point the resulting image will have artifacts. Regardless of computed error values, artifacts are undesirable in an image reconstruction. From this we expect that the CF method is inappropriate for use in progressive image transmission or lossy image compression schemes. It is also inappropriate for use in situations (e.g. digital photography) where a wide variety of images may be encountered. Because of its lack of aliasing artifacts, PLHaar is a more appropriate choice than CF.

6. Discussion

PLHaar shows a lot of promise. We have demonstrated through basic entropy measurements and Quantization/PSNR computations that PLHaar should be suitable for lossy and lossless image processing and manipulation. We have shown its superiority over current n–bit to n–bit transform methods–PlHaar does not have the aliasing artifacts present in CF, and unlike the TLHaar method PLHaar has both discrete and continuous forms.

From the entropy measurements it appears that PLHaar's compression ratios will not be as good as those obtainable by the S–Transform. Since the CF and TLHaar methods also have inferior compression ratios this problem is most likely part of the nature of n–bit to n–bit transforms and not a flaw in PLHaar itself. We believe that this lower compression ratio is acceptable since PLHaar was designed first to address the problem of dynamic range expansion. The loss in compression rate is mitigated by the ability to transform and manipulate data completely in a limited–width environment.

Future work will center on developing data compression methods for use with PLHaar. These methods will include embedded coding methods [3, 5, 9], suitable for progressive transmission. We would also like to compare embedded methods using PLHaar to those using CF, to see how the embedded encoding affects the aliasing problem in CF. Since PLHaar was developed for use in a limited–width environment we will also be experimenting with its use in hardware and developing prototype applications that can be run on graphics hardware.

7. 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

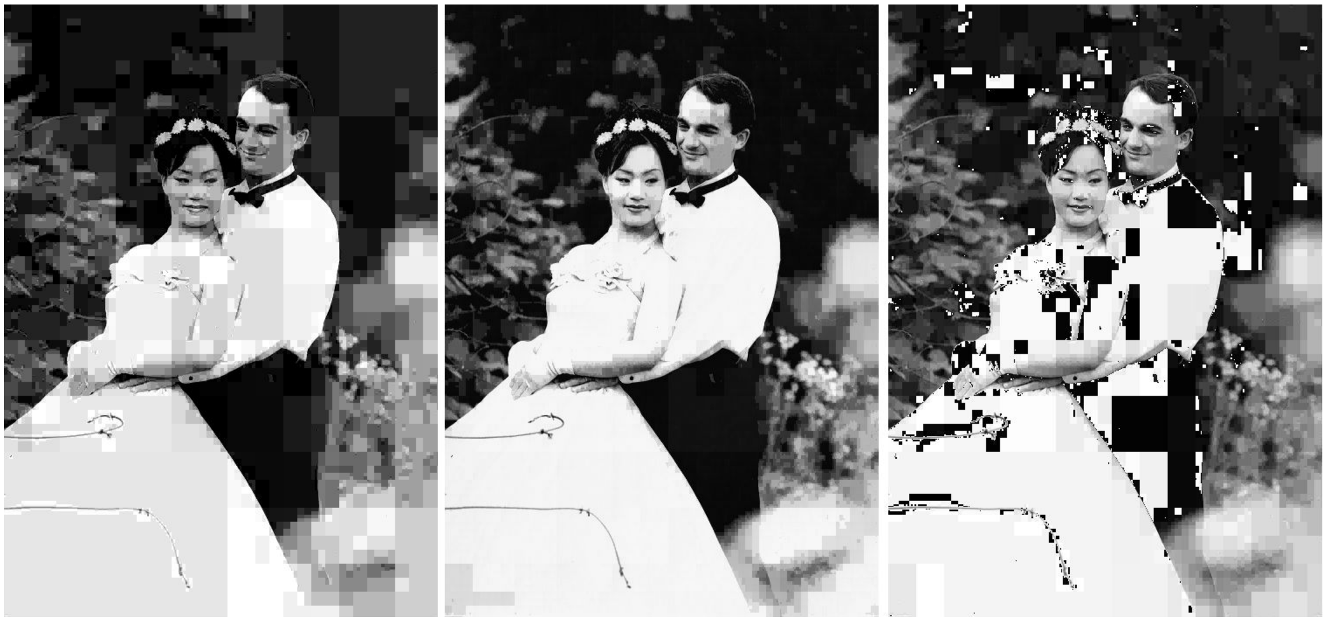

Figure 11. The Wedding photo, transformed (left to right) by the S, PLHaar, and CF transforms. Note the aliasing artifacts present in the CF reconstruction.

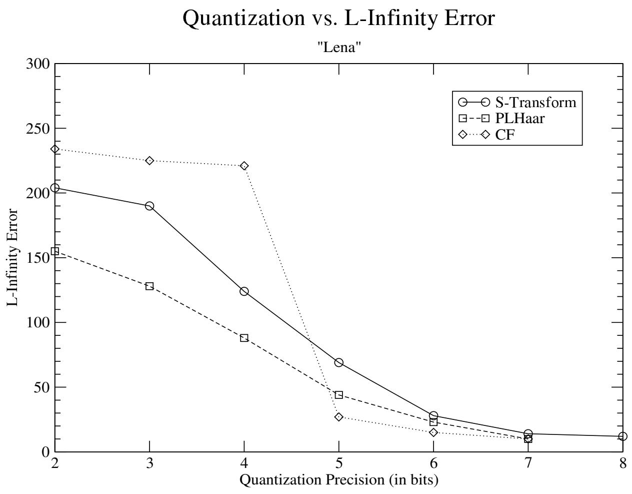

Figure 10. Lena: L_∞ error values for increasing quantization precision.

United States Department of Education Government Assistance in Areas of National Need (DOE-GAANN) grant #P200A980307.

References

[1] A. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Lossless image compression using integer to integer wavelet transforms. In *Proceedings. International Conference on Image Processing*, pages 596–599. IEEE Computer Society, 1997.

[2] H. Chao and P. Fisher. An approach to fast integer reversible wavelet transforms for image compression, 1996. Preprint, available from citeseer.ist.psu.edu.

[3] E. S. Hong and R. E. Ladner. Group testing for image compression. In J. Storer, editor, *Proceedings DCC 2000 Data Compression Conference*, pages 3–12. IEEE Computer Society, 2000.

[4] A. Kiely and M. Klimesh. The ICER progressive wavelet image compressor. *The Interplanetary Network Progress Report 42–155, July–September 2003, Jet Propulsion Laboratory*, pages 1–46, Nov 2003.

[5] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, Jun 1996.

[6] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, second edition, 2000.

[7] J. Senecal, M. A. Duchaineau, and K. I. Joy. Reversible n–bit to n–bit integer Haar–like transforms. In *Proceedings of the 7th IASTED Conference on Computer Graphics and Imaging*, 2004. To appear.

[8] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[9] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(13):3445–3462, Dec 1993.

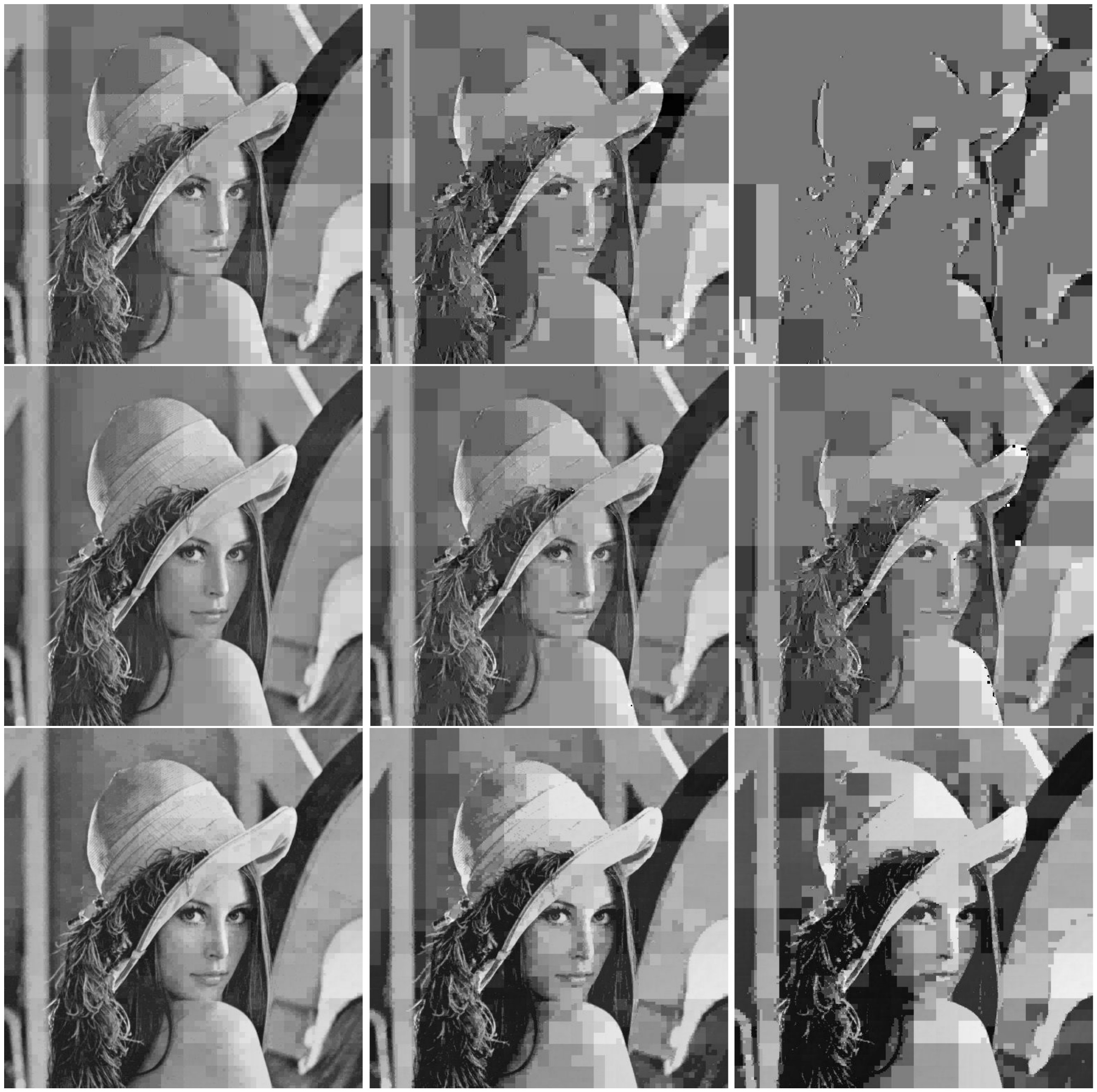

Figure 12. “Lena”, with coefficients quantized to 5, 4, and 3 bits. From the top: S–Transform, CF, PL-Haar.