UCRL-CONF-204061–REV–1

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

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

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

U.S. Department of Energy

Lawrence Livermore National Laboratory

August 4, 2004

Approved for public release; further dissemination unlimited

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

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.

Approved for public release; further dissemination unlimited

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

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

Kenneth I. Joy‡
‡Institute for Data Analysis and Visualization
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 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 number of possible output coefficient values is greater than the number of possible input values, therefore the number of bits required to represent a coefficient is greater than the number of bits required to represent an input (for a discussion of dynamic range expansion and its effects see [5]). Using the S–Transform [2] (to be reviewed later) as an example, n–bit inputs to the transform can have 2^n possible values. The resulting output coefficients can have $2^{n+1} - 1$ possible values, which requires $n + 1$ bits to represent.

Dynamic range expansion presents some problems. If the transform is being performed in hardware (such as a video frame buffer) that has a channel width limited to n bits, some data loss will result due to forced truncation of the coefficients. If custom hardware implementing the transform is to be designed and built, circuitry for handling the extra bit adds complexity and cost to the design. Because modern computer architectures allocate storage in 8–bit increments, an implementation of the transform on a computer will require 16 bits to store each 9–bit coefficient.

To our knowledge there are only two published methods for completely eliminating dynamic range expansion in a transform. These are the TLHaar method [8] and the method of Chao, Fisher, and Hua [3] (here referred to as the CFH method), both of which will be reviewed later. CFH and TLHaar are discontinuous transforms, and only suitable for use when the the transform coefficients are manipulated in a lossless manner.

Given a transform that takes a pair of function values (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 possible input pairs (A, B). The *range* of the transform is the set of all possible output coefficient pairs (L, H), given the domain. Without loss of generality, unless stated otherwise we will assume that the domain is square and centered on (0,0).

The Piecewise–Linear Haar transform is nonlinear, but it transforms the domain on a piecewise basis, the transformation of each piece being linear. PLHaar is an improvement over CFH and TLHaar because PLHaar is continuous, and is therefore suitable for both lossy and lossless methods. Our tests indicate that PLHaar is usable in a compression system, and its compression ratios should be competitive with those obtained using the other n–bit to n–bit transforms. For images that have areas of high contrast PLHaar delivers a superior lossy reconstruction. In addition, images transformed by PLHaar and reconstructed lossily have increased contrast, which helps preserve lines and edges in the image.

* 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*

2. The Haar Wavelet Transform

We begin by examining the Haar Wavelet Transform [7]. This transform takes a pair of function values (A, B) and performs a 45–degree (or one–eighth) rotation of that point about the origin in Euclidean (or L_2) space. In this space, points in the domain that are equidistant from the origin lie on a 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:

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

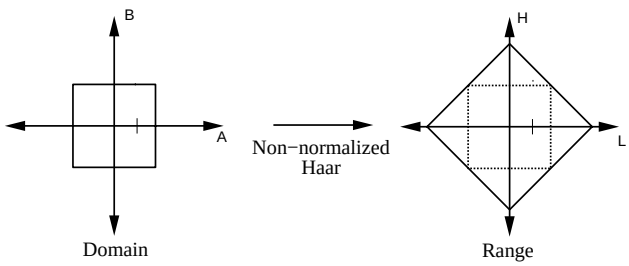

Figure 1. The non–normalized Haar transform, showing dynamic range expansion.

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 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 is integer–valued, it should be obvious that the range of the non–normalized Haar transform is also integer–valued, 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 in the range L will be drawn from $[0, 2N]$ and H from $[-N, +N]$, arranged in a lattice as in the left side of figure 2.

The S–Transform [2], defined by equations 4 and 5, is an integer version of the Haar 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. 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). However, the range of high–pass coefficients is still twice the domain.

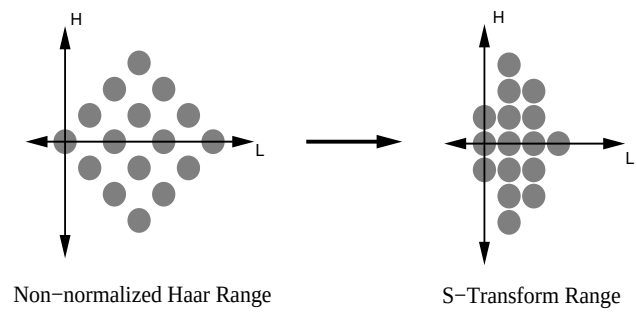

Figure 2. Eliminating low–pass dynamic range expansion by "squashing".

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 making a transform that is a permutation of the domain: intelligently remap each position in the domain such that the resulting transform de–correlates the input data.

TLHaar [8] is an integer–to–integer transform that, given a bit width n, uses a pair of square two–dimensional lookup tables (each with an edge dimension of 2^n). One table is a mapping $(A, B) \rightarrow (L, H)$ for the forward transform, and the table for the inverse is a mapping $(L, H) \rightarrow (A, B)$. Each table is initialized with an identity transform, and then the rows and columns of the inverse transform table are sorted so that for any given two pairs of inputs their high– and low–pass values will have the same magnitude relationships as those in Haar. When a swap occurs in the in-

verse transform table, the corresponding entries in the forward transform table are also swapped. Because the tables are initialized with an identity transform no table entry will have a value outside the transform domain. Thus dynamic range expansion cannot occur.

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:

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

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

where H indicates a Haar high–pass coefficient, and $\tilde{H}$ indicates a TLHaar coefficient.

Although TLHaar produces transform tables that decorrelate input data, this method has two shortcomings. The first is that the transform is not continuous and is therefore generally suitable only for use with lossless methods. Second, for n–bit inputs each table contains 2^{2n} entries of $2n$ bits per entry. This quickly becomes unwieldy at larger values of n.

2.3. The CFH Method

The CFH method [3] 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 CFH 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 CFH method has an aliasing problem that causes undesirable behavior when the difference between A and B overflows and wraps around, causing H to have a sign opposite the expected. CFH is unable to distinguish large positive numbers from small negative ones (and vice–versa) because they have the same binary representation. For example, if $n = 8$, the range is [-128,127]. If $A = -1$ and $B = 127$, the CFH method computes $H = -128$, which is nowhere near the difference between A and B. The L value is generally considered to be an "average" of A and B, but because of H being far from the expected value, L is computed as -65, which is also far from its expected value. 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 encode the coefficients then severe artifacts may appear in the reconstruction. Many continuous–tone images do not have adjacent pixel pairs with differences wide enough to cause this behavior, so in practice CFH works well for a wide range of images, reconstructed both lossily and losslessly. However, this aliasing problem is a fundamental weakness in the method, and means that the CFH transform is not continuous. The PLHaar transform, unlike the CFH and TLHaar transforms, eliminates dynamic range expansion while completely preserving continuity.

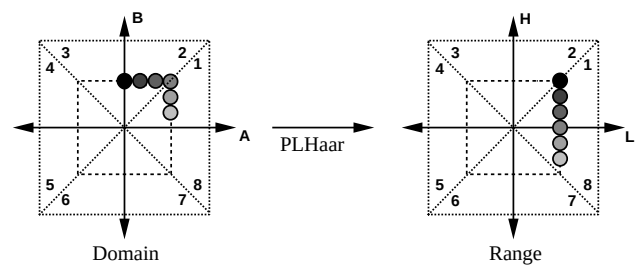

Figure 3. The PLHaar transform's rotation in L_∞ space. Numbered octants are shown.

3. The PLHaar Transform

The Haar transform 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 as a whole is nonlinear, but when taken on a piecewise (octant–by–octant) basis, the transform from octant to octant is linear. It is from this property that we derive the name "Piecewise–Linear Haar", or "PLHaar". The PLHaar transform maps integers to integers, and is an autohomeomorphism (meaning it maps the domain onto itself, is one–to–one, and is continuous).

The transform of a point at coordinates (A, B) is given by

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

where R is one of four matrices, as given in equation 11, depending on the octant (Oct.) where the point (A, B) is located.

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

Note that R is the non–normalized Haar matrix, but with one of the four elements zeroed out. Two characteristics of the PLHaar transform are that $R(x, x) = (x, 0)$, decorrelating adjacent identical values, just as in the Haar transform. Also, when A and B are close in value, the low–pass filter satisfies $|L| = \max\{|A|, |B|\}$. This has the effect of increasing contrast in the low–pass data.

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 CFH transforms do not have this property.

Figure 4 shows images of the four transforms, interpreted as lookup tables (LUTs). The origin of each table (except the TLHaar table) is its center, with the horizontal axis corresponding to A and the vertical to B. Red corresponds to L, and green to H, biased by 128. For the TLHaar table the origin is the lower–left corner, and there is no bias on H. The discontinuities in the CFH transform are obvious. The discontinuities in the TLHaar table are less obvious, but they are present and obvious on closer inspection. The S–Transform table has no discontinuities, but many of its high–pass entries (as indicated by the blue–tinted area) contain values that are outside of the domain. Only the PLHaar table has no discontinuities and no out–of–range values.

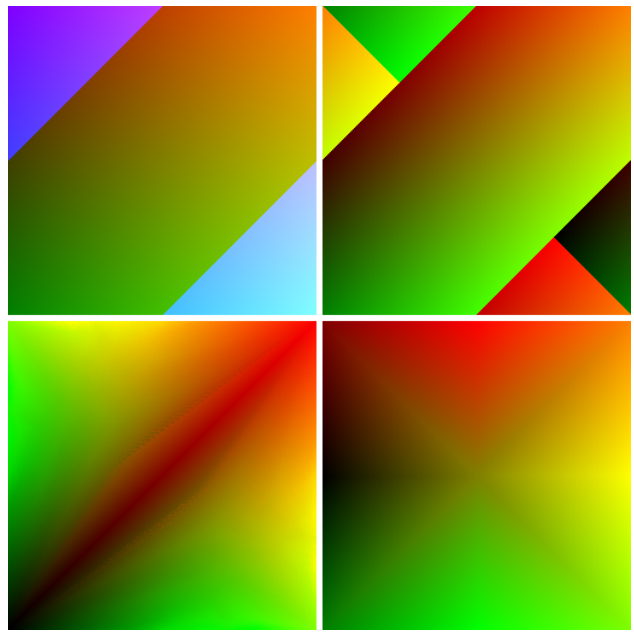

Figure 4. The transform LUTs for (clockwise from the upper–left) S–Transform, CFH, PLHaar, and TLHaar. Here we use $A - B$ (and not $B - A$) when computing H. The PLHaar table is the only one that does not have discontinuities or out–of–range values.

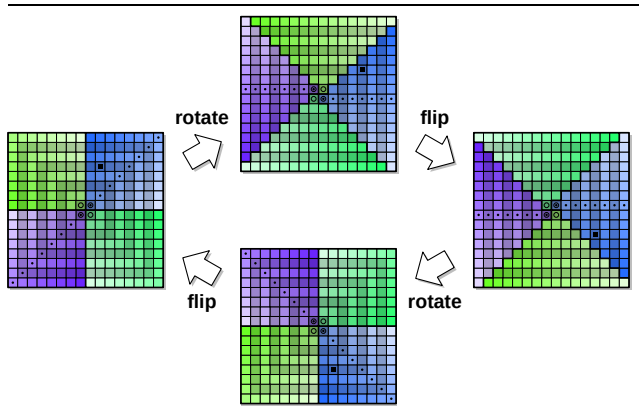

Figure 5. How PLHaar is an involution. Note that both rotations are in the same direction.

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 proper 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. To make PLHaar an involution we negate the second row of each R 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 5.

3.2. Implementation

Source code for the continuous PLHaar transform is given in figure 6, and for the discrete transform in figure 7. Note that both of these procedures implement the modified transform (the involution) described above.

```
#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 6. The source code for the continuous PLHaar transform.

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 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 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, then translate the quadrant back. If the domain and range contain an odd number

```
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;       // (**) 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 7. The source code for the discrete PLHaar transform.

of integers (e.g. [0,254]) then the lines marked `(**)` may be removed and the bias set accordingly (e.g. 127).

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`.

3.3. Benefits of PLHaar

PLHaar has several key benefits over current n-bit to n-bit transforms:

- **PLHaar is continuous**. This is PLHaar's prime advantage. PLHaar is a continuous transform, while TLHaar and CFH are not. This allows PLHaar to be used with lossy and lossless methods.
- **PLHaar can be performed by direct computation or table lookup**. TLHaar is a LUT method only.
- **PLHaar requires less table storage**. Because PLHaar is an involution, when performing PLHaar via lookup table the same LUT may be used for the forward and inverse transforms. CFH and TLHaar require separate tables for each.

4. Evaluating PLHaar

4.1. Entropy

Transforms are used because the transformed data may be more efficiently manipulated, a common manipulation

being compression. To verify that PLHaar is usable as a transform for compression we iteratively transformed some 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 [9] according to

$$E = \sum_i -p_i \times \log_{256} p_i \tag{12}$$

where p_i is the probability of coefficient value i and the summation is taken over all nonzero p_i. We compared the entropy resulting from the PLHaar transform to those of the S, TLHaar, and CFH transforms.

4.2. Quantization and PSNR

To gain a basic understanding of how useful PLHaar will be when used in lossy compression or a progressive transmission scheme, we performed some quantization tests on a set of test images. We transformed the images as in section 4.1, 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 falls into the range [40,47]. Assuming a uniform distribution, to avoid biasing quantized coefficients 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 $[u, v]$ we compute the center as $\lfloor (u+v)/2 \rfloor$, 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).

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

5. Results and Discussion

5.1. Entropy

Results of the entropy measurements are given in figure 8, where "Image" is the entropy of the untransformed image. A graph of the histograms for the "Lena" image is in figure 9.

The data in figure 8 shows that PLHaar is useful for compression. For continuous–tone images (such as Lena and Bike) PLHaar performs slightly worse than CFH and the S–Transform, although the difference is such that it is not readily noticeable in the graph. PLHaar does not do as well for France (which is line art), whereas for MRI and WashSat PLHaar performs better. From these results it appears that when using PLHaar in a compression system the compression ratio may not be as good as that obtainable by using the S–Transform, but the difference is not likely to be significant.

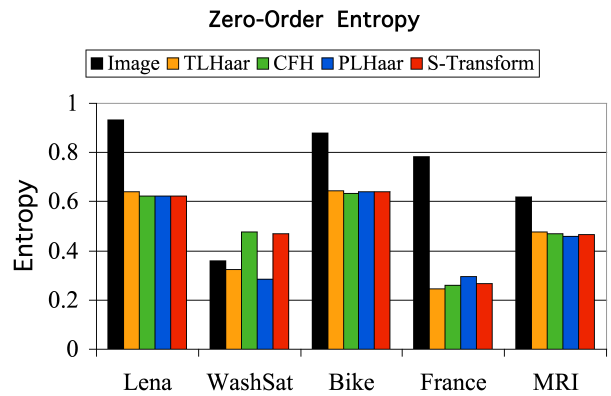

Figure 8. The zero–order entropy of the wavelet coefficients produced by the transforms.

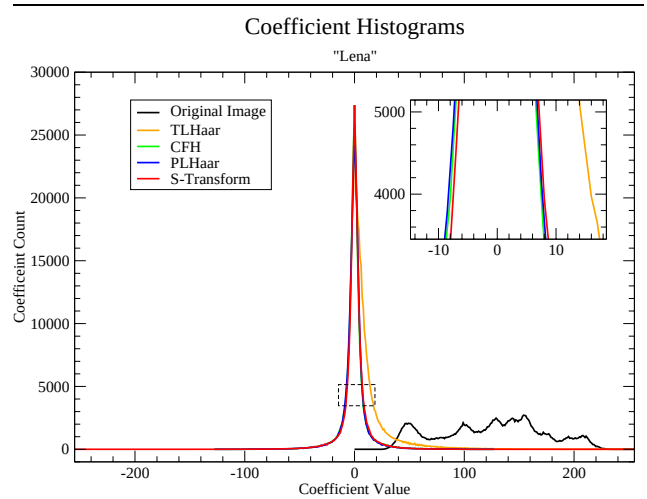

Figure 9. Histograms of the "Lena" image, and the S–Transform, PLHaar, TLHaar, and CFH coefficients.

Figure 10. The Wedding photo, transformed by the S, PLHaar, and CFH transforms, and quantized to 4 bits. Note the artifacts present in the CFH reconstruction.

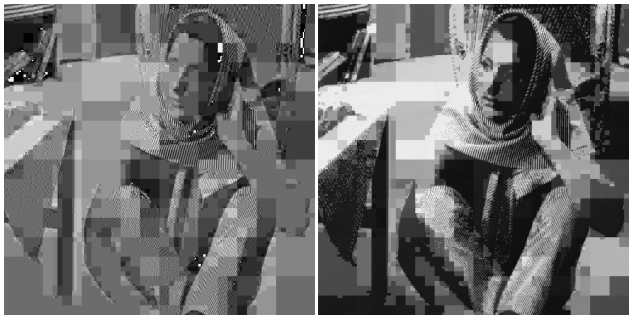

Figure 11. "Barbara" transformed by CFH (left) and PLHaar (right), and quantized to 3 bits.

5.2. PSNR

Figure 10 gives an excellent example of the problems inherent in the CFH method. It shows a grayscale photo that has been transformed by the S, PLHaar, and CFH transforms, and quantized to 4 bits before reconstruction. As expected many artifacts appear in the CFH reconstruction, particularly in areas of high contrast. PSNR values are respectively 21.88, 25.17, and 11.75.

As mentioned earlier, the low–pass values computed by PLHaar have a higher contrast than the inputs to the transform. This means that the more PLHaar coefficients are quantized, the higher the contrast in the reconstructed image. Although this increased contrast can result in a lower PSNR, it has the benefit, at a low quantization precision, of helping to bring out edges that are "blurred" away in the CFH and S–Transforms. Figure 11 shows the reconstructed "Barbara" image, after transformation by CFH and PLHaar and quantization to 3 bits. The PSNR of these images is respectively 19.06 and 16.48. An examination of the images shows that the image reconstructed from PLHaar coefficients is visually more appealing. Ignoring the artifacts present in the CFH reconstruction, we see that the PLHaar reconstruction has preserved such things as the lines in her face (note the eyes), the edges and lines on her pants, and the table covering.

A graph of the PSNR curves for the "Bike" image is given in figure 12, and the L_∞ error curves are given in figure 13. From these we see that the PLHaar transform gives a better reconstruction than all other tested methods. In particular, due to the aliasing problem discussed earlier, the CFH transform is unable to give a good reconstruction of the image. Figure 14 shows the Bike image, transformed by CFH and PLHaar, quantized to 3 bits of precision, and reconstructed. Underneath each image is a set of four details.

If figure 15 we give images of the reconstructed "Lena" image, after being transformed by the TLHaar, CFH, PLHaar, and S transforms and quantized. In the CFH reconstruction a few artifacts appear in the 3–bit image (note the

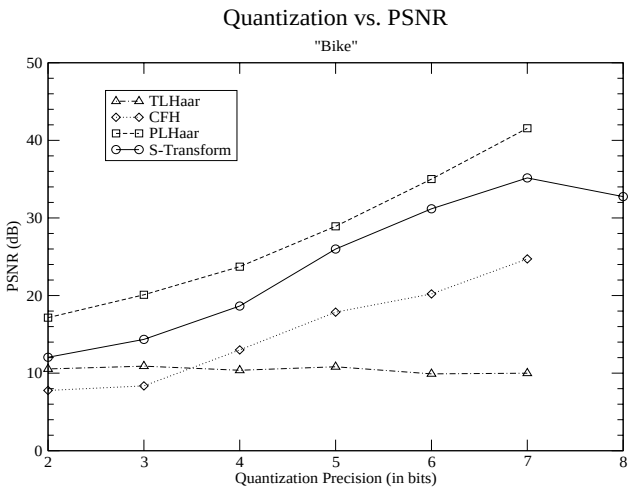

Figure 12. Bike: PSNR values for increasing quantization precision.

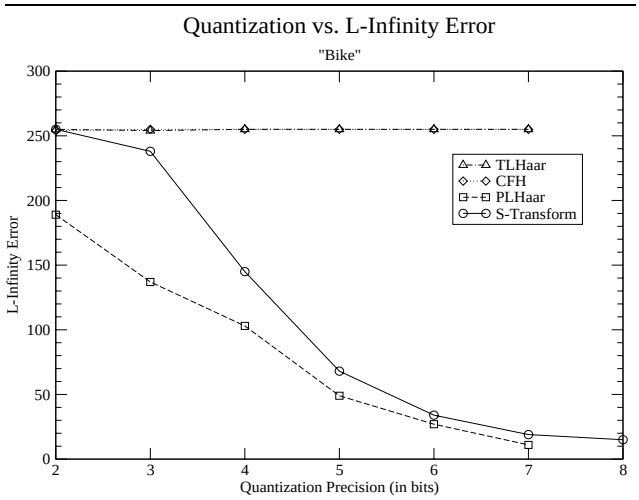

Figure 13. Bike: L_∞ error values for increasing quantization precision.

inner edge of the mirror frame, along the edge of her shoulder, and along the brim of her hat). Also, when comparing the 3–bit CFH and PLHaar reconstructions, we see again how PLHaar's increased contrast improves the visual quality of the reconstructed image. Note for example that in the PLHaar reconstruction her facial features are more intact.

6. Conclusion

We have demonstrated through basic entropy measurements and Quantization/PSNR computations that PLHaar is 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 CFH, PLHaar gives a more visually appealing reconstruction at lower bitrates, and unlike the TLHaar method PLHaar has both discrete and continuous forms.

When used in a data compression system PLHaar's coding rate is likely to be slightly worse than that obtainable by using the CFH transform. We believe that this is acceptable since the CFH transform is discontinuous, and when used in a lossy compression scheme CFH is unable to guarantee an acceptable reconstruction due to the artifacts that appear.

Further testing is warranted to better evaluate PLHaar's usefulness in compression. Future work will center on developing data compression methods for use with PLHaar. These methods will include embedded coding methods [4, 6, 10], suitable for progressive transmission. We would also like to compare embedded methods using PLHaar to those using CFH, to see how the embedded encoding affects the aliasing problem in CFH. 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

We are grateful to our anonymous reviewers for their helpful comments and suggestions.

The "MRI" test image was provided by the Department of Radiology at the University of California Medical Center. The "Bike" image is part of the ISO/IEC test image set described in ISO 12640[1].

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

[1] ISO 12640. Graphic Technology–prepress digital data exchange–standard color image data (scid), 1995.

[2] A. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Lossless image compression using integer to integer wavelet transforms. In *Proceedings. International Conference on*

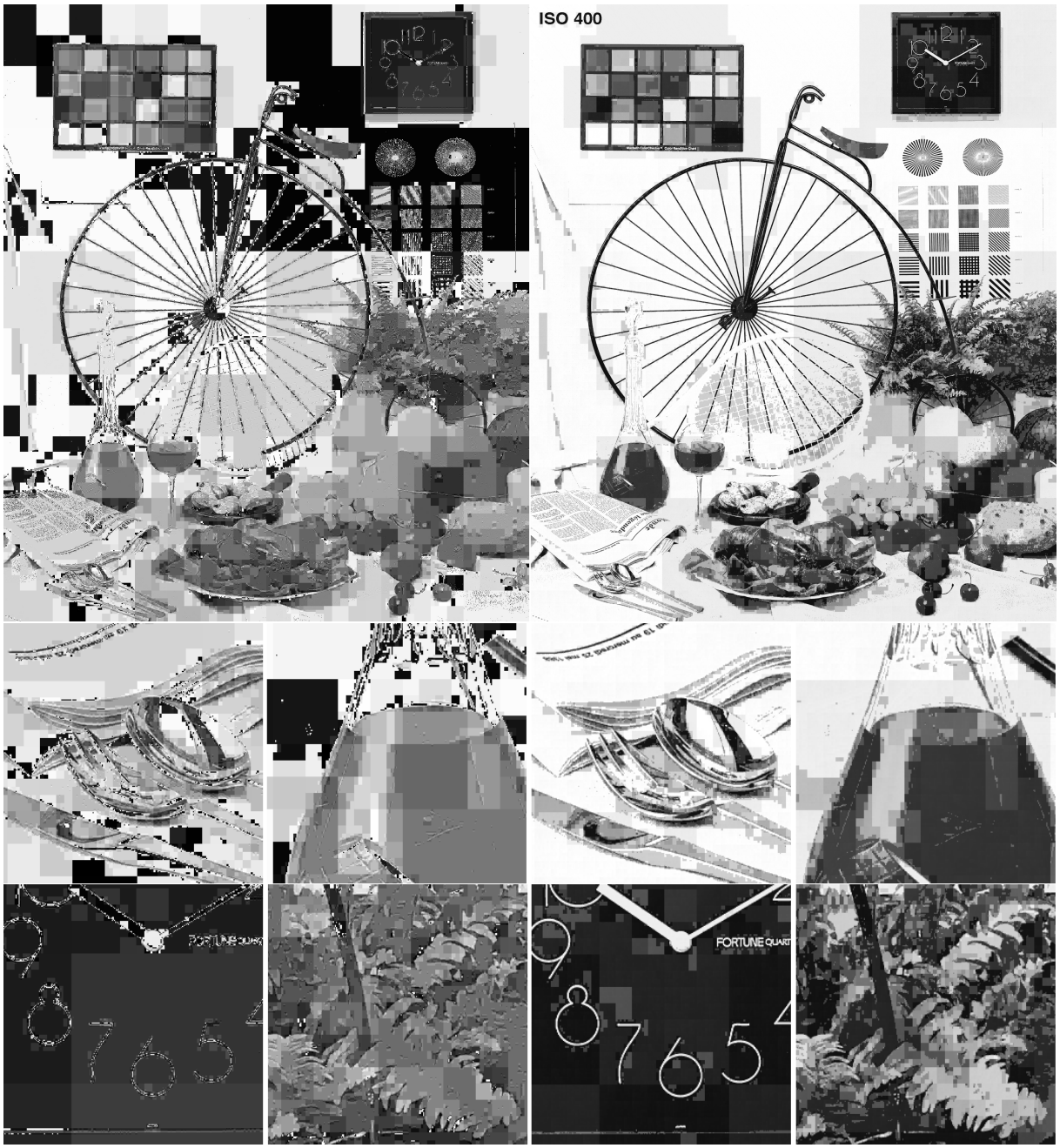

Figure 14. “Bike” image and details, transformed by CFH (left) and PLHaar (right) and quantized to 3 bits.

Image Processing, pages 596–599. IEEE Computer Society, 1997.

[3] H. Chao, P. Fisher, and Z. Hua. An approach to integer wavelet transformations for lossless image compression. In Chen, Li, Micchelli, and Xu, editors, *Advances in Computational Mathematics*, volume 202 of *Lecture Notes In Pure and Applied Mathematics*, chapter 2, pages 13–38. Marcel Dekker, Inc., 1999.

[4] 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.

[5] A. Kiely and M. Klimesh. The ICER progressive wavelet image compressor. *The Interplanetary Network Progress Re-*

Figure 15. "Lena", quantized to 5, 4, and 3 bits. From the top: TLHaar, CFH, PLHaar, and S–Transform.

port 42–155, July–September 2003, Jet Propulsion Laboratory, pages 1–46, Nov 2003.

[6] 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.

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

[8] J. G. 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.

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

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